

Transition to novel Internet of Things technology for measurement in hydrology – case study in Cape Town, South Africa



Prepared by:

Roberto Agostinho De Oliveira, BSc. Civil Eng

Supervised by:

Dr. John Okedi, Pr. Eng, PhD

Research report submitted in fulfilment of the requirements for the degree of Master of Science
in Civil Engineering MSc. (Eng)

Department of Civil Engineering
University of Cape Town, Private Bag Rondebosch, 7700
Cape Town, South Africa

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Plagiarism declaration

- i. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own.
- ii. I have used the Harvard-UCT convention for citation and referencing. Each contribution to, and quotation in, this body of work, from the work(s) of other people has been attributed and has been cited and referenced.
- iii. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.
- iv. I have not allowed and will not allow anyone to copy my work with the intention of passing it off as his or her own work.
- v. I acknowledge that copying someone else's work, or part of it, is wrong, and declare that this is my own work.

Name and Surname: Roberto De Oliveira

Student Number: DLVROB001

Date: 01/02/2023

Signature:

Signed by candidate

Acknowledgements

Firstly, I would like to thank my amazing fiancé and life partner, Brogan, for all of her support and patience throughout this entire journey of completing my dissertation. She has been there for me always and has celebrated my academic wins as well as helped deal with the losses. I would also like to thank her for saying “YES” to marrying me during this journey and adding so much joy to it by making me the happiest man alive. I would also like to extend my thanks to her mother, Almaree, her father Theo, and her three brothers, Dylan, Connor, and Nevan for being an amazing and supportive family.

To my dad, Jose, my mom Vitalina, my sister Andrea, my brother-in-law Jarred and my niece and Goddaughter Lina, thank you for being my comfort and my strength and for always reminding me of how proud you are of me. I would not be where I am today without the support, love and care I received from you during my research and my life. I am very blessed to have you all as family. To all my extended family and friends, thank you for your support and encouragement throughout my studies.

I would like to tremendously thank and register my great appreciation to my supervisor, Dr. John Okedi, for all of his support and expertise throughout the completion of my dissertation. He was always available and willing to lend an ear and provided highly knowledgeable advice and feedback. I am truly grateful to have had him as a supervisor. Thank you, John, for giving me an opportunity to step out of comfort zone and to learn and grow academically.

To all the wonderful people that formed part of the “Internet of Drops” project, thank you for welcoming me into your IoT community and for all the expertise and knowledge you shared. A warm thanks to Professor Jan Diels, who was an integral member of the community, for all his help and expertise whenever required. These zoom calls to Belgium provided massive project related breakthroughs for which I am very grateful. To the Future Water folks who attended the monthly stormwater webinars, thank you for listening to my progress and for your invaluable advice and support. To Ameen and Curwin, thank you for all your electronics-based assistance and for lending an ear and giving advice on all things to do with microelectronics - your help was greatly appreciated. A vote of thanks to SAWS for providing weather data used in this project, this was very helpful.

To my best mate Aaron and my fellow IoT ground breaker, Sam, thank you guys for all your support and help in setting up my sensors, amongst many other things. I couldn’t have done it without you two. To all my fellow student colleagues up on the 6th floor – Geordie, Aaron, Sam, Nadiya, Meghan and Owen - who were with me during different stages throughout this journey, thank you for making it enjoyable and relating to all the struggles that come with academic research. Your company and support were great and appreciated.

Lastly, to my funders, the University of Cape Town, and the National Research Fund, thank you for your financial generosity and support in helping me to complete my dissertation. I believe

that this has given me the platform to use my skills and knowledge to actively work towards a better future in the field of Civil Engineering through sustainable design and net positive solutions.

Abstract

In this study, an investigation was undertaken to determine the viability of novel Internet of Things (IoT) technology application in hydrology with a case study in Cape Town, South Africa. The IoT system that collected hydrological data was built with the use of low-cost, off-the-shelf components and was investigated as a pilot study in a catchment around the University of Cape Town's (UCT) Upper Campus. The demand for this type of monitoring system in South Africa is largely due to poor and unreliable hydrological data collection and monitoring.

The main aim of this study was to show that novel IoT technology for hydrological data collection can replace conventional systems and determine the viability of transitioning to novel IoT technology for hydrological measurements. In the study, hydrological monitoring systems were built using sensors and a gateway that collected data from various points of measurement which were then conveyed from the network to an internet database. The data was accessed and visually displayed to an end user in a user-friendly manner.

The systems contained multiple layers of architecture that connected with one another through the different hardware and software components used in the design. The hardware of the system was involved in controlling and managing the sensors to transmit data to the internet. The software throughout each layer prepared the data and sent it to the internet where it could be displayed to an end-user. The hardware components of the system were selected based on cost, power consumption and applicability to hydrological monitoring. The hardware and software components were integrated to create two separate hydrological monitoring nodes, namely the Water Monitoring Node (WMN) and the Soil-Air Monitoring Node (SAMN). The WMN measured water level and water temperature, while the SAMN measured soil moisture, soil temperature, air temperature and relative humidity.

Once created, the systems were experimentally tested to determine its ability to provide real-time hydrological data. Using these results, they were evaluated based on reliability and performance for application as a real-time hydrological monitoring system. The systems were housed in weatherproof containers and mounted on a level gauge within the UCT dam and on a tree located above the New Engineering Building (NEB) on the university's Upper Campus. The testing period for the sensor nodes in totality lasted 4 months and 6 days with the last data point retrieved on the 01 of November 2022.

The WMN transmitted 4049 messages across a total of 42 days, while the SAMN transmitted 10366 messages across 128 days. The data outputted by the nodes were visually displayed through the Things Speak platform and the results produced from the experimental deployment were used to investigate the reliability and performance of the nodes. The reliability analysis looked at the robustness and network communication of the system while performance analysis looked at load testing and power consumption. Initial deployment of the node was disrupted by theft and vandalism. The WMN and SAMN were then deployed and proved to be robust towards harsh environmental conditions during the 170 days and communicated with the

internet efficiently. The enclosures housing the nodes successfully protected all components while communication was done within the 1% duty cycle, with an average packet loss rate of 4.56%. Abnormalities and flaws in the systems were highlighted during deployment and this included erroneous data points and irregularities with the precision of recordings. The actual and projected lifespan of the WMN were determined as 24 and 36 days, respectively, while the projected life span for the SAMN was 183 days. The output from the sensors were validated using multiple methods including spot checks and comparisons with hydrological data from the nearby Kirstenbosch South African Weather Services (SAWS) weather station. According to spot checks, the water level data produced had an average accuracy error of 0.0005% while, when compared to the SAWS station, the temperature and humidity data produced had an average accuracy error of 7%. Based on the findings of the study, there is potential for using IoT-based solutions for hydrological monitoring in Cape Town, South Africa.

Table of Contents

Plagiarism declaration	i
Acknowledgements	ii
Abstract	iv
Table of contents	vi
List of Figures	xi
List of Tables	xiv
Abbreviations	xv
1. Introduction	1-1
2. Literature Review	2-1
2.1 Current Hydrological Collection & Monitoring Methods	2-1
2.2 Internet of Things (IoT)	2-2
2.2.1 Basic Concept of IoT	2-2
2.2.2 IoT Architecture	2-3
2.2.3 Opportunities & Challenges for IoT in South Africa	2-5
2.3 Wireless Sensor Networks	2-6
2.3.1 WSN Technology	2-6
2.3.2 Network Topology	2-7
2.4 Mac Network Protocols	2-11
2.5 Low-Powered Wide Area Networks (LPWAN)	2-12
2.5.1 LoRa	2-13
2.5.2 LoRaWAN	2-15
2.5.2.1 <i>Network Topology & Architecture</i>	2-15
2.5.2.2 <i>Device Classes</i>	2-16
2.5.2.3 <i>Security</i>	2-17
2.5.2.4 <i>LoRaWAN in Africa and South Africa</i>	2-18
2.5.3 The Things Network	2-19
2.5.4 Things Speak	2-19
2.5.5 Sigfox Communication	2-20
2.5.6 NB-IoT Communication	2-20
2.6 IoT Hardware	2-20
2.6.1 Microcontroller Unit (MCU)	2-20
2.6.2 Sensor Devices	2-21
2.6.3 Sensor Nodes	2-22

2.7	Power Consumption in IoT Systems	2-23
2.7.1	Battery Powered	2-23
2.7.2	Energy Harvesting	2-24
2.7.3	Energy Efficient Techniques	2-24
2.8	Application of IoT in Hydrology	2-26
2.9	Summary	2-28
3.	Conceptual Model	3-1
3.1	Perception Layer	3-1
3.1.1	Sensor Node	3-2
3.1.2	LoRa Gateway	3-2
3.2	Middleware Layer	3-3
3.3	LoRa Network	3-4
3.3.1	LoRa Network Software	3-4
3.3.2	LoRa Application	3-4
3.4	Summary	3-5
4.	Methodology – System Design	4-1
4.1	Overview – System Design	4-1
4.2	Statement of Method – System Design	4-2
4.3	Project Requirements	4-2
4.4	LoRa Communication	4-3
4.4.1	Network Implementation	4-3
4.4.2	Network Configuration	4-4
4.4.3	Message Structure	4-5
4.5	Hardware Selection and Application	4-6
4.5.1	Microcontroller Device	4-6
4.5.2	Printed Circuit Board (PCB)	4-7
4.5.2.1	<i>Headers and Terminal Blocks</i>	4-7
4.5.2.2	<i>JST & Voltage Divider</i>	4-8
4.5.2.3	<i>MIC2545A -2YM Power Switch</i>	4-8
4.5.2.4	<i>ADS1015IDGST Analog to Digital Convertor</i>	4-8
4.5.2.5	<i>CD74HC4052M96 4 Channel Analog Multiplexer & Demultiplexer</i>	4-9
4.5.3	Sensing Implementation	4-9
4.5.3.1	<i>Temperature Sensing</i>	4-9
4.5.3.2	<i>Atmospheric Temperature & Relative Humidity Sensing</i>	4-11

4.5.3.3	<i>Soil Moisture Sensing</i>	4-12
4.5.3.4	<i>Water Level Sensing</i>	4-15
4.5.4	Node Setup	4-18
4.5.5	Power Considerations	4-21
4.6	Software Application	4-25
4.6.1	Firmware Design for Pycom	4-25
4.6.1.1	<i>Onboard ADC</i>	4-25
4.6.1.2	<i>I2C</i>	4-26
4.6.1.3	<i>Pin Selection</i>	4-26
4.6.1.4	<i>LoRa Setup</i>	4-28
4.6.2	Micro Python Libraries	4-28
4.6.2.1	<i>adcR.py</i>	4-28
4.6.2.2	<i>adsx1x15.py</i>	4-29
4.6.2.3	<i>adsx15read.py</i>	4-29
4.6.2.4	<i>BME280_float.py</i>	4-29
4.6.2.5	<i>onewire.py</i>	4-30
4.6.2.6	<i>sht31.py</i>	4-30
4.6.2.7	<i>watermark.py</i>	4-30
4.6.2.8	<i>measureSensors.py</i>	4-32
4.6.3	Sensor Setup	4-33
4.6.3.1	<i>DS18B20 Submersible Water Temperature Sensor</i>	4-33
4.6.3.2	<i>DF Robot – SHT31 Temperature and Humidity Sensor</i>	4-34
4.6.3.3	<i>Hafei WNK Submersible Water Level Sensor</i>	4-34
4.6.3.4	<i>Watermark Soil Moisture Sensor</i>	4-35
4.6.4	The Things Network	4-36
4.6.5	Things Speak	4-42
4.7	Summary of Method – System Design	4-43
5.	Methodology – System Implementation	5-1
5.1	Overview – System Implementation	5-1
5.2	Statement of Method – System Implementation	5-1
5.3	Selection of Study Area	5-2
5.4	Practical Field Experimentation	5-5
5.4.1	Setup of Field Experiment	5-5
5.4.2	Period and Duration of Experiment	5-8
5.4.3	Accuracy and Precision Testing	5-8
5.5	Analysis From Experiment	5-9

5.5.1	Data Output of IoT System	5-10
5.5.1.1	<i>Water Level Data</i>	5-11
5.5.1.2	<i>Water Temperature Data</i>	5-11
5.5.1.3	<i>Soil Moisture Data</i>	5-12
5.5.1.4	<i>Air Temperature & Humidity Data</i>	5-13
5.5.2	Reliability Analysis	5-14
5.5.2.1	<i>Robustness of System</i>	5-14
5.5.2.2	<i>Communication of System</i>	5-14
5.5.3	Performance Analysis	5-15
5.5.3.1	<i>Load Testing</i>	5-15
5.5.3.2	<i>Power Consumption Testing</i>	5-15
5.6	Summary of Method – System Implementation	5-15
6.	Results and Analysis from Experiment	6-1
6.1	Analysis of Data Output from IoT Nodes	6-1
6.1.1	Water Level Data	6-1
6.1.2	Water Temperature Data	6-3
6.1.3	Soil Moisture Data	6-5
6.1.4	Air Temperature & Humidity Data	6-8
6.2	Robustness Analysis	6-12
6.2.1	Robustness – Water Monitoring Node	6-13
6.2.2	Robustness – Soil-Air Monitoring Node	6-14
6.3	Network and Communication Analysis	6-14
6.3.1	Duty Cycle	6-15
6.3.2	Percentage Packets Lost	6-16
6.4	Load Testing Analysis	6-17
6.4.1	Load Testing Analysis – Water Monitoring Node	6-17
6.4.2	Load Testing Analysis – Soil-Air Monitoring Node	6-20
6.5	Power Consumption Analysis	6-23
6.5.1	Power Consumption Analysis – Water Monitoring Node	6-23
6.5.2	Power Consumption Analysis – Soil-Air Monitoring Node	6-24
7.	Discussion of Results	7-1
7.1	Validation of Data	7-1
7.1.1	Water Level	7-1
7.1.2	Water Temperature	7-3

7.1.2	Soil Moisture	7-4
7.1.2	Air Temperature & Humidity	7-4
7.2	Data Output of the WMN and SAMN	7-6
7.2.1	WMN	7-6
7.2.1	SAMN	7-7
7.3	Reliability of System	7-7
7.4	Performance of System	7-9
7.5	Cost and Maintenance	7-11
7.5.1	Cost Analysis	7-11
7.5.2	Maintenance Plan	7-11
7.6	Summary	7-12
8.	Conclusion and Recommendations	8-1
8.1	Conclusion	8-1
8.2	Recommendations for Future Work	8-3
9.	References	9-1
	Appendices	
A	Stepwise Summary for Sensor Deployment	A-1
B	Bit-Packing Examples	B-1
C	MCU Specifications	C-1
D	Water Sensor Expansion Board 1.1 Diagram	D-1
E	Sensor Product Sheets	E-1
F	Relationship Between Voltage & Water Level	F-1
G	Detailed Drawings of Radiation Shield	G-1
H	Diagrams of Setup for Power Consideration Testing	H-1
I	Python Script for Measuring Battery Voltage	I-1
J	Node Experimental Field Setups Micro	J-1
K	Accuracy and Precision Testing	K-1
L	Extracts from WMN and SAMN	L-1
M	Daily Min, Max and Averages for the WMN and SAMN Data	M-1
N	SAWS Kirstenbosch Weather Station Data Comparison to the WMN and SAMN data	N-1
O	Summary of Enclosure Temperature and Humidity Data Collected by the WMN	O-1
P	Detailed Cost Analysis of Sensor Nodes	P-1

List of Figures

Figure 2-1: 3- & 5-Step Layered Architecture of an IoT System (Jamali, et al., 2020)	2-3
Figure 2-3: Star Topology	2-8
Figure 2-4: Tree Topology	2-8
Figure 2-5: Mesh Topology	2-9
Figure 2-6: Rate vs Range of Communication Technologies (Tsavalos & Hashem, 2018)	2-12
Figure 2-7: Typical LoRaWAN network architecture (The Things Network, 2021)	2-16
Figure 2-8: Tiered Structure of LoRaWAN communication (The Things Network, 2021)	2-19
Figure 3-1: Operational Flow Diagram for Conceptual IoT Model	3-1
Figure 3-2: Proposed Sensor Node Setup of the System at the Perception Layer	3-2
Figure 3-3: Locality of Western Cape TTN community in S.A. (The Things Network, 2021)	3-3
Figure 3-4: LoRa coverage across the City of Cape Town (The Things Network, 2021)	3-3
Figure 4-1: Overview of Methodology – System Design	4-1
Figure 4-2: IoT Testing (Data Flair Team, 2018)	4-3
Figure 4-3: LoRa coverage over the UCT area (The Things Network, 2021)	4-4
Figure 4-4: Uplink Message Structure	4-5
Figure 4-5: Downlink Message Structure	4-6
Figure 4-6: Annotated Diagram of the Pycom LoPy4 board (Core Electronics, 2020)	4-7
Figure 4-7: Voltage Divider Circuit on the PCB	4-8
Figure 4-8: DS18B20 Submersible Temperature Sensor.	4-9
Figure 4-9: Circuit Diagram for the DS18B20 Sensor	4-10
Figure 4-10: Circuit Built to Read Data from the DS18B20 Sensors	4-11
Figure 4-11: DF Robot – SHT31 Weatherproof Temperature & Humidity Sensor	4-11
Figure 4-12: Pin out Diagram of the SHT31 Sensor	4-12
Figure 4-13: The Watermark Soil Moisture Sensor	4-12
Figure 4-14: Basic voltage divider circuits (Blom, 2021)	4-13
Figure 4-15: Circuit Diagram for Watermark Sensor	4-14
Figure 4-16: The Hafei WNK Submersible Water Level Sensor	4-15
Figure 4-17: Relationship Between Output Voltage of Sensor and the Depths of a Liquid	4-17
Figure 4-18: Circuit Built to Measure the Current Consumption of the Nodes	4-22
Figure 4-19: LoPy4 Mounted on PCB with Sensors Connected	4-18
Figure 4-20: Pictorial Representation of WNM Setup	4-19
Figure 4-21: Pictorial Representation of SAMN Setup	4-20
Figure 4-22: Weatherproof Enclosure with L-bracket Housing the Sensor Nodes	4-20
Figure 4-23: Radiation Shield Built to House Temperature and Humidity Sensor	4-21
Figure 4-24: Micro Python Script to Limit Power Consumption of LoPy4 and PCB	4-32
Figure 4-25: Micro Python Script that Reads the DS18B20 Sensor	4-33
Figure 4-26: Micro Python Script that Reads the SHT31 Sensor	4-34
Figure 4-27: Micro Python Script that Reads the Water Level Sensor	4-35
Figure 4-28: Micro Python Script that Reads the Watermark Sensor	4-36

Figure 4-29: Micro Python Script that Connects the Device to TTN.	4-38
Figure 4-30: Micro Python Script that Packages the Data and Sends it to TTN	4-39
Figure 4-31: Hexadecimal Payload Converted using a Payload Formatter	4-40
Figure 4-32: Decoder Function for the WMN	4-41
Figure 4-33: Decoder Function for the SAMN	4-42
Figure 5-1: Overview of Methodology – System Implementation	5-1
Figure 5-2: Locality Map of Cape Town, South Africa (ESRI, 2021)	5-2
Figure 5-3: Locality Map of UCT, Cape Town (Google Earth, 2022)	5-3
Figure 5-4: Locality Map of Sensors on UCT Upper Campus (ESRI 2022)	5-4
Figure 5-5: System Setup of the WMN	5-5
Figure 5-6: System Setup of the SAMN	5-6
Figure 5-7: Example of Soil Moisture Sensor Deployment (Irrometer, 2020)	5-7
Figure 5-8: Locality Map of SAWS Kirstenbosch station in comparison to the Nodes (ESRI,2021)	5-10
Figure 5-9: Water Level Data Visualisation for Things Speak	5-11
Figure 5-10: Water Temperature Data Visualisation for Things Speak	5-12
Figure 5-11: Soil Moisture Data Visualisation for Things Speak	5-12
Figure 5-12: Air Temperature & Humidity Data Visualisation	5-13
Figure 6-1: Time Series of Water Level Data Measured from the UCT Dam	6-2
Figure 6-2: Gauge Displaying Water Level from Things Speak (Things Speak, 2022)	6-2
Figure 6-3: Time Series of Water Temperature Data Measured from the UCT Dam	6-3
Figure 6-4: Daily Min, Max and Average Water Temperatures	6-4
Figure 6-5: Gauge Setup on Things Speak to Display Real-time UCT Water Temperature	6-5
Figure 6-6: Time Series of Soil Moisture Data Measured from UCT Upper Campus	6-6
Figure 6-7: Time Series of Soil Temperature Data Measured from UCT Upper Campus	6-7
Figure 6-8: Daily Min, Max and Average Soil Temperatures Across the SAMN Deployment	6-8
Figure 6-9: Gauge Setup on Things Speak to Display Soil Moisture and Temperature	6-9
Figure 6-10: Time Series of Air Temperature Data Measured from UCT Upper Campus	6-9
Figure 6-11: Min, Max and Average Air Temperatures Across the SAMN Deployment	6-10
Figure 6-12: Time Series of Air Humidity Data Measured from UCT Upper Campus	6-11
Figure 6-13: Daily Min, Max and Average Relative Humidity	6-11
Figure 6-14: Relationship between Air Temperature (°C) and Relative from SAMN	6-12
Figure 6-15: Gauge on Things Speak to Display Real-time Air Temperature and Humidity	6-13
Figure 6-16: Deployment 0 - System Vandalised	6-13
Figure 6-17: Graph showing battery voltage during power outages for the WMN	6-14
Figure 6-18: Graph showing battery voltage during power outages for SAMN	6-15
Figure 6-19: Irregularities in Dam Level Readings due to Changes in Water Level	6-19
Figure 6-20: Irregularities in Dam Level Readings due to Battery Life	6-20
Figure 6-21: Irregularities in Water Temperature Readings for WMN1	6-21
Figure 6-22: Water temperature recorded during WMN2 with no Flaws or Irregularities	6-21
Figure 6-23: Unknown irregularities in SAMN1 data	6-22

Figure 6-24: Time Series of Battery Voltage for Iteration 1 and 2 of the WMN	6-24
Figure 6-25: Node Battery Life vs Number of Readings Produced by WMN1	6-25
Figure 6-26: Time Series of Battery Voltage for Iteration 1 and 2 of the SAMN	6-26
Figure 6-27: Node Battery Life vs Number of Readings Produced by SAMN2	6-26
Figure 7-1: Comparison of WMN Water Level data with Water Level Spot Checks	7-2
Figure 7-2: Comparison between the Dam Water Level and Kirstenbosch SAWS Station	7-3
Figure 7-3: Comparison between Soil Moisture and Kirstenbosch SAWS Weather Station	7-4
Figure 7-4: Comparison of Air Temperature (°C) recorded by the SAWS and the SAMN	7-5
Figure 7-5: Comparison of Relative Humidity recorded by the SAWS the SAMN	7-6
Figure 7-6: Condition of the WMN and SAMN after the End of Deployment	7-8

List of Tables

Table 2-1: Advantages & Disadvantages of Network Topologies (Pandya, 2013; Mathews Modieginyane, et al., 2017; IntellegenInc., 2015; Inst Tools, 2020)	2-10
Table 2-2: Parameters for LoRa spreading factors (Semtech, 2019)	2-14
Table 2-3: LoRa common frequency plans (The Things Network, 2021)	2-14
Table 2-4: LoRa common frequency plans (Semtech, 2019)	2-15
Table 2-5: LoRaWAN Security Keys (LoRa Alliance, 2020)	2-18
Table 2-6: Impact of energy efficient techniques (Soua & Minet, 2011)	2-25
Table 4-1: Colour coded pin out description for DS18B20	4-10
Table 4-2: Colour coded pin out description for SHT31	4-12
Table 4-3: Colour coded pin out description for water level sensor	4-16
Table 4-4: Average Voltage Reading of Nodes While in Active and Deep Sleep Modes	4-23
Table 4-5: Description of I2C parameters (Pycom , 2021)	4-26
Table 4-6: Description of Pin methods (Pycom , 2021)	4-26
Table 4-7: Description of default pin modes (Pycom , 2021)	4-27
Table 4-8: Description of keyword arguments for adcR.py library	4-28
Table 4-9: Description of keyword arguments for ads1x15.py library	4-29
Table 4-10: Description of keyword arguments for adsx15read.py library	4-29
Table 4-11: Description of keyword arguments for __init__ function	4-30
Table 4-12: Description of keyword arguments for readWM# function	4-31
Table 4-13: Description of keyword arguments for read function	4-31
Table 4-14: Description of keyword arguments for ShockkPa function	4-31
Table 4-15: : TTN parameters selected for the Nodes	4-37
Table 6-1: Monthly Average, Min and Max Values for Water Temperature Across WMN Deployment	6-4
Table 6-2: Monthly Soil Moisture (kPa) data across SAMN deployment	6-6
Table 6-3: Soil Temperature (°C) data Across WMN Deployment	6-8
Table 6-4: Monthly Air Temperature (°C) data across SAMN deployment	6-10
Table 6-5: Average, Min and Max Values for Air Humidity Across SAMN Deployment	6-12
Table 6-6: Breakdown of water monitoring nodes payload	6-16
Table 6-7: Breakdown of soil & air monitoring nodes payload	6-16
Table 6-8: Results derived from online airtime calculator for LoRaWAN	6-17
Table 6-9: Results of Percentage Packets Lost for WMN	6-18
Table 6-10: Results of Percentage Packets Lost for SAMN	6-18
Table 7-1: Spot Checks of the Water Level in the UCT Dam.	7-1
Table 7-2: Enclosure Temperatures and Humidity Across WMN1 and WMN2	7-10
Table 7-3: Summary of Cost Analysis for the Systems	7-11
Table 7-4: Maintenance Plans for the Systems Based on a 5000 mAh Li-Po Battery	7-12

Abbreviations

6LoWPAN - IPv6 over Low -Power Wireless Personal Area Networks

ABP - Authentication By Personalisation

AC – Alternating Current

ADC - Analog-to-Digital Converter

AI – Artificial Intelligence

ALOHA - Advocates of Linux Open-source Hawaii Association

API - Application Programming Interface

CoCT – City of Cape Town

CPU – Central Processing Unit

CRC - Cyclic Redundancy Check

CSMA - Carrier Sense Multiple Access

CSV - Comma Separated Values

GND – Ground

GPIO – General Purpose Input Output

GSM - Global System for Mobile

HMN - Hydrological Monitoring Node

I2C - Inter-Integrated Circuit

ICASA - Independent Communications Authority of South Africa

IDE – Integrated Development Environment

IERC - IoT European Research Center

IoT – Internet of Things

IP – Internet Protocol

ISM – Industrial, Scientific and Medical

JST - Japan Solderless Terminal

LiPo - Lithium-ion Polymer

LoRa – Long Range Radio

LoRaWAN – LoRa Wide Area Network

LPWAN – Low-Powered Wide Area Network

LTE - Long Term Evolution
M2M – Machine-to-Machine
MAC – Medium Access Control
MCU - Microcontroller Unit
MPU – Microprocessor Unit
MQTT - Message Queuing Telemetry Transport
NB-IoT - Narrowband Internet of things
NEB - New Engineering Building
NiMH - Nickel Metal Hydride
OOP - Object Oriented Programming
OTAA - Over The Air Authentication
PCB – Printed Circuit Board
PHDR – Physical Header
QR - Quick Response
RF – Radio Frequency
RFID - Radio Frequency Identification
RAM – Read Access Memory
REST - Representational State Transfer
ROM – Read Only Memory
RTC – Real Time Clock
SAEON – South African Environmental Observation Network
SAMN – Soil-Air Monitoring Node
SAWS – South African Weather Services
SCL – Serial Clock
SDA – Serial Data
SDG – Sustainable Development Goals
SEMAT - Smart Environmental Monitoring and Assessment Technologies
SF – Spreading Factor
SPI - Serial Peripheral Interface
TTN – The Things Network

UCT - University of Cape Town

UN – United Nations

UNB – Ultra Narrow Band

UPS – Uninterrupted Power Supply

VCC - Voltage Common Collector

WC – Wireless Communication

Wi-Fi - Wireless Fidelity

WMN – Water Monitoring Node

WRM – Water Resource Management

WSN – Wireless Sensor Network

1. Introduction

The current available methods of collecting hydrological data are labour intensive while the results obtained from these methods are unable to provide real-time data. The Internet of Things (IoT) is a contemporary technology that can be used in the development of low-cost, reliable, and user-friendly systems that provide real-time hydrological data. Hydrological data forms a major component of Water Resource Management (WRM). Collecting this data involves processes that are known to be either, expensive, specialised, resource intensive, unreliable or all the above. The methods used to collect hydrological data need to be altered and improved to provide the ability to monitor highly accurate data in real-time.

WRM has an impact on several key issues in society, such as the environment, water consumption, food production, sewage treatment, purification, irrigation, energy balance, etc. (Robles, et al., 2014). These impacts have a major effect on the socio-economic state of a region, and if not managed correctly can have a detrimental impact on the efficiency of a region's economy (Meintjies, 2001). The continued exponential growth of the world's population and, rapid urbanisation in developing countries, burdens the already limited water resources available. In 2015, The United Nations (UN) developed 17 Sustainable Development Goals (SDG) in order to provide a blueprint on ending poverty by improving health & education, spurring economic growth, reducing inequality, while simultaneously combating climate change and protecting oceans and forests (United Nations, 2016). The aim of the 17 SDG's set out by the UN is to ensure sustainable development by 2030. This research is done with the aim of moving towards achieving the goals related to sustainable WRM.

IoT is a combination of sensor and network technology which works together to create a Wireless Sensor Network (WSN) that generates accurate, precise and timely data from the surrounding environment (Han, et al., 2012). Recent developments in the application of IoT technology for the automation of data collection has provided the opportunity to collect hydrological data remotely, which diminishes the need to use existing outdated methods. IoT technology, when utilised correctly, provides accurate real-time data. The data provided allows for greater sustainable WRM, as decisions are based on data that is accurate, precise, and recorded in real-time. The main advantages of this method of hydrological data collection involve low cost, remote access and providing a more direct, precise way to closely monitor hydrological information in real-time.

IoT technology includes many applications designed to assist in many sectors, such as transportation, healthcare, industry, markets, environment, agriculture, and smart cities (Pellicer, et al., 2013; Arasteh, et al., 2016; Sumi & Ranga, 2016; Raaijen & Daneva, 2017). Although it is a widely used method of data acquisition, not much research has been done in the application of these technologies in the field of hydrology and WRM. Recent advances in the IoT field have allowed for the implementation of low cost, lower power consumption systems that are robust and reliable. This system would be applicable in the field of hydrology and fits all the characteristics needed for an efficient and effective data collection method. The typical IoT

system consists of various components that are needed to collect, monitor, connect, manage, analyse and display things data (Ismail, et al., 2019). These components consist of sensors, processors, gateways, and applications (Das & Jain, 2017). These components are widely available, with many different product ranges. Despite the value of having many different options when choosing an IoT component, choosing the appropriate option that suites the specified application is troublesome. This was particularly evident in literature which either suggests favourable options or evaluates particular options (Ismail, et al., 2019; Turcu, et al., 2018; Robles, et al., 2014; Lazarescu, 2013).

The focus of this project was to develop an IoT platform which utilises a WSN in order to collect hydrological data in real-time. The project acted as a case study for the implementation of an IoT platform in South Africa. The scope of the project involved building an IoT system by choosing hardware and software components that are compatible and can be integrated to create an IoT architecture that allows it to function at the required standard. The proposed system needed to be low-cost, energy efficient, flexible, robust, and reliable. It also needed to be interoperable and adaptable with already existing technology and required simplified implementation.

The proposed system with the above-mentioned attributes was developed to collect and monitor hydrological data on air temperature, air humidity, soil moisture, water temperature and water level. The overall objectives of the research and the systems implementation were:

- i. Identifying existing low-cost sensor node hardware components.
- ii. Adapting the selected hardware components, through programming and microelectronics, to the hydrological requirements of the project.
- iii. Utilising an open-source network and cloud service for transmitting and processing hydrological data.
- iv. Integrating the sensor nodes and open-source cloud service to develop a system that allows the transmission of hydrological data measured in-situ through the internet to an end-user in a timely and user-friendly manner.
- v. Ensuring the system selected is low-cost, low-power consumption, reliable, robust, and fit for purpose.

Despite the value of having several different IoT platforms as well as, several different hardware and software components, selecting the appropriate combination is challenging due to there being no significant benchmark or systematic evaluation criteria that exists to evaluate and compare the various platforms and systems (Ismail, et al., 2019). This resulted in an increase in emphasis on the platform and component selection, with a lot of research needed in this area to make the right decision. Successful selection of the IoT platform and hardware/software components would aid in the overall success of the system. The success of an IoT based hydrological monitoring system would result in an increased and improved research output following improved data available. With IoT technology being a new technology, the ability of local

authorities in South Africa to embrace and benefit from IoT remains a challenge. Local authorities lack the knowledge and skills to make adequate use of IoT technology to monitor hydrological data. This project looks to provide the evidence on how to successfully implement an IoT based hydrological monitoring system in South Africa.

The location selected for this study was in and around the University of Cape Town's (UCT) upper campus, specifically the UCT dam and surroundings for the water temperature and water level, while the soil moisture, air temperature and air humidity monitoring occurred near the New Engineering Building (NEB) which is located on UCT upper campus. The location was selected based on the accessibility to the water body, as well as the low risk to theft of equipment. The location was of low importance for this study given that the study is mainly centred around the viability of developing an efficient hydrological data monitoring system.

2. Literature Review

This chapter provides an overview on IoT, and its application based on existing literature. The chapter begins with an introduction into the current hydrological methods used to collect and monitor data. There is an in-depth analysis of IoT technology and its different applications, as well as how it can be applied in the collection and real-time monitoring of hydrological data. The chapter then ends off with a detailed analyses of various software and hardware components used in IoT systems.

2.1 Current Hydrological Collection and Monitoring Methods

South Africa has a long history of hydrological monitoring which dates to the late 19th and early 20th century (Nomqophu, 2018). Hydrological monitoring intends to measure and map where and in what quantities water is stored. Older methods of hydrological data collection involve specialised and expensive equipment and are labour intensive. These monitoring methods do not have the ability to display real-time data and decisions are made on data that may be outdated or inaccurate due to real-time changes of the already collected data.

Rainfall data collected in South Africa involves the use of rain gauges which come in several different models, but all share a common functionality. The South African Environmental Observation Network (SAEON) (2017) states that South Africa has over 100 rain gauges and weather stations which monitor rainfall in the region. Based on data in 2017, only five stations could provide real-time data which is published online every five minutes. Current methods of measuring and collecting water temperature involves either recording instantaneous measurements at different times of the day and generating an average or utilising water temperature loggers which measure temperatures on an hourly frequency or less (Dallas, 2008). Spot measurements are available for many rivers in South Africa, but water temperatures are not routinely measured. This method of measurement means there is little to no understanding of the diurnal variation or maximum/minimum temperature values (Dallas, 2008). Monitoring the water level in water bodies across South Africa is done by gauging stations which use river flows and water levels. The number of active gauging stations has been declining due to various reasons, which provides a challenge in terms of hydrological monitoring (Water Resource Information Management, 2012). Many of these gauging stations involve the construction of weirs with known dimensions which may hinder the natural progression and flow of the water body (Unidata, 2017). Larger water bodies tend to use mechanical sensors to measure the depths at different points along the water body. These sensors record the data but require manual collection. This can create problems ranging from compromised data sets, human error and decreased resolution of measurements (Peacock, 2017). Soil moisture can be measured both in-situ and through laboratory testing of grab samples. Both options are currently labour intensive and require specialised personnel and equipment in order to take accurate measurements (Brodie, 2019; Kramarenko, et al., 2016).

Current in-situ testing methods of soil moisture is quicker, tests a larger volume and are therefore more representative of the soil, but the interpretation of the data is generally empirical and crude (Sivakugan, 2016). These methods, particularly in South Africa, are labour and resource intensive and this results in either outdated or missing hydrological data.

2.2 Internet of Things (IoT)

2.2.1 Basic Concept of IoT

The basic concept of IoT involves connecting physical things to computers and the internet. The term ‘Internet of Things’ was first mentioned in 1991 by Kevin Ashton who defined it as uniquely identifiable objects and their virtual representation in an ‘internet-like’ structure (Ashton, 2009). The term ‘things’ refers to information, people, or objects in the physical environment. The definition of IoT has developed over the years with several clearer definitions which tend to differ based on the context in which the technology is being used. The IoT European Research Centre (IERC) (2013) defined IoT as “A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual “things’ have identities, physical attributes, and virtual personalities and use intelligence, and are seamlessly integrated into the information network”. This definition covers the entire basis of IoT, and many other definitions relate to this.

IoT technology creates a symbiotic relationship between the internet and the surrounding environments. This relationship allows the physical world to communicate directly to humans through the internet. Physical things can see, listen, think, and perform tasks by sharing information and coordinating decision-making using IoT technology. IoT can transform physical objects into smart objects by utilising its fundamental technologies such as sensor networks, communication technologies, embedded devices, and pervasive and ubiquitous computing (Yelamarthi, et al., 2017). Incorporating IoT has many advantages which can help individuals, businesses, and society daily. The integrated application of these technologies creates a “Smart Environment”. A “Smart Environment” is an environment that is thoroughly and extensively interwoven with sensors, actuators, displays, and computational elements, embedded coherently in everyday objects, and connected through a continuous network (Weiser, et al., 1999). This enables the accurate and real-time monitoring of a specific environment. Monitoring these environments generate useful data that can be used to make important decisions and can also create algorithms that predict future environmental events. IoT systems are influenced by unpredictable events and physical characteristics that generate unreliable results during simulation (Yelamarthi, et al., 2017). The ability to monitor and predict otherwise unpredictable events is the main driver of the implementation of IoT in many fields, including hydrological monitoring.

IoT is made up of several wireless technologies. Wireless Communication (WC) provides the benefits of mobility, connectivity between devices as well as connectivity between devices and the internet, all without the use of cables (Khutsoane, et al., 2017).

The wireless technologies include WSNs, Low-Powered Wide Area Networks (LPWAN), Radio Frequency Identification (RFID), Internet Protocol (IP), Electronic Product Code, Barcode, Wireless Fidelity (Wi-Fi), Bluetooth, ZigBee, Near Field Communication, Actuators, and Artificial Intelligence (AI) (Madakam, et al., 2015). WC technologies can be split into short range and medium to long range communication, with Wi-Fi, Bluetooth, Zigbee and IPv6 over Low - Power Wireless Personal Area Networks (6LoWPAN) the most notable short-range communications, and RFID, Sigfox, Long Range Radio (LoRa), and Narrowband Internet of things (NB-IoT), the most notable medium to long range communications (Das & Jain, 2017; Khutsoane, et al., 2017). WSNs are one of the major technologies used for IoT deployment. Its success in IoT is due to the technologies ability to integrate physical world objects with cyberspace (Khutsoane, et al., 2017). A wireless IoT system has specific architecture which makes up the components of the system. IoT architecture is a diverse subject with no specific standard. The IoT architecture is implemented based on both context and designer's choice in a project, and this results in architecture divergence (IEC, 2014). Consequently, this is the main reason as to why there is a lack of standardisation in IoT systems.

2.2.2 IoT Architecture

IoT architecture is the backbone of these systems. It is made up of a collection of physical objects, actuators, sensors, cloud services, communication layers, developers, users, business layers, and IoT protocols (Jamali, et al., 2020). Developing an architecture is important in the design process of an IoT system as it is responsible for creating the high-level structure of the IoT system (Ghapar, et al., 2018).

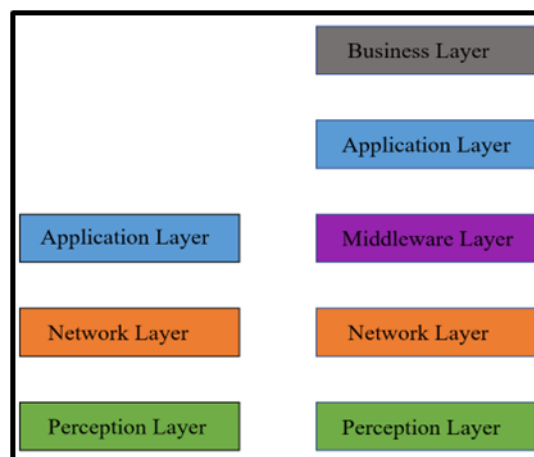


Figure 2-1: 3- & 5-Step Layered Architecture of an IoT System (Jamali, et al., 2020)

IoT can be seen as a layered architecture with each layer defined by its function and the devices used. There is not a defined number of layers in an IoT architecture. Basic systems are said to have only three layers while slightly more complex systems tend to have five or more. Figure 2-1 shows a basic system architecture with a perception, network, and application layer, next to a

more complex system made up of a perception, network, middleware, application and business layer (Ghapar, et al., 2018; Pardini, et al., 2019).

The perception layer or sensing layer contains devices and things that have sensors/actuators connected to gather information about their environments (Singh, 2020). Data in an IoT system originates in the perception layer (Ghapar, et al., 2018). The layer detects, collects, and then processes data to obtain useful information which is then sent to the network layer through a transmitting device. It applies technologies that can detect relevant parameters of physical characterisation, such as flow, weight, temperature, humidity, etc., in addition to the collection of identification data of objects, such as RFID and Quick Response (QR) codes (Pardini, et al., 2019). It is at this layer where most of the variation comes about with the use of different sensor/actuator devices to decrease power consumption and reduce cost of the overall system.

The network layer connects smart things, networks, devices, and servers to the internet and to each other (Singh, 2020). The function of this layer is to transmit, and process data collected in the perception layer. The information is transmitted via the internet, mobile communication, and wireless networks (Ghapar, et al., 2018). This layer acts as the connection between the perception layer and the upper layers and uses short, medium and long-range communication technologies (Pardini, et al., 2019). The network technology used is based on the range at which information needs to be transmitted as well as the power consumption during transmission. Along with the basic assignments, the network layer also conducts cloud computing procedures and data management procedures (Pardini, et al., 2019).

The middleware layer, also known as the process layer, stores, analyses, and processes large amounts of data that comes from the network layer (Singh, 2020). It manages and provides a diverse set of services to lower layers of the IoT architecture. The layer employs technologies, such as databases, cloud computing, and big data processing modules. The middleware layer is made up of software or even a set of sublayers that work to integrate components of the IoT architecture that would not communicate otherwise (Pardini, et al., 2019). It also allows the application layer to interact with the perception layer to ensure effective communication and plays a vital role in the development of new technologies.

The application layer is responsible for delivering application-specific services to users, which it obtained from the sensors/actuators in the perception layer (Singh, 2020). It defines various applications in which IoT can be utilised by interpreting the collected data and making the information available for interested parties. The information from the lower layers is received, integrated, and then stored by this layer (Ghapar, et al., 2018).

The business layer manages and controls the whole IoT system and includes applications, business and profit models and user privacy (Singh, 2020). It provides high level analysis reports of the lower layers and has the responsibility of creating graphs and flow charts (Pardini, et al., 2019).

2.2.3 Opportunities & Challenges for Hydrological Application of IoT – South Africa

The implementation of IoT technology in the developing world provides endless opportunities, ranging from environmental monitoring to smart city application (Miazi, et al., 2016). However, these opportunities are coupled with several challenges that need to be overcome to successfully implement the technology in these countries. South Africa, like many other countries, has a high demand for the diverse application of IoT technology, including the real-time monitoring of hydrological data.

High quality hydrological-related data is fundamental to WRM, water and sanitation infrastructure design and scientific research (Fell, et al., 2019). This data is vital in ensuring the sustainable development of water-related infrastructure. Smart monitoring equipment can be used to collect this data, but this equipment is often very costly and susceptible to vandalism and theft. The idea behind the implementation of IoT in South Africa and other developing countries is to rather create innovative ways to utilise the current technology as opposed to adopting technology verbatim. IoT innovation has the potential to solve the many problems that are unique to developing countries (Masinde, 2014). It is important to develop an IoT system that uses low-cost “off-the-shelf” components that are suitable for use in the South African context.

Hydrological data in developing countries are fragmented, inaccurate, and are largely based on isolated measurements (Fell, et al., 2019). This provides a need for accurate, real-time hydrological data that is measured in-situ. The system must account for the many challenges faced in developing countries to perform accurate, real-time hydrological monitoring in South Africa. These can be categorised as technical challenges, device reliability, financial challenges and security, privacy and trust (Miazi, et al., 2016). The technical challenges faced are factored around internet connectivity, data centres, power resources and human resources. Internet connectivity and human resources in developing countries are the major issue. For IoT to be successful, reliable and adequate internet access needs to be available. Unfortunately, ensuring the provision of internet connectivity is particularly challenging in rural/remote areas where most of the implementation will take place. Thus, solving this problem would involve a hybrid internet connection system. The continent of Africa has a mobile penetration of higher than 63% while developing countries host more than 70% of the world’s mobile sim cards (Miazi, et al., 2016; Masinde, 2014). Global System for Mobile (GSM) based solutions are, thus, one of many feasible options. The issue of human resources is due to the lack of technically knowledgeable personnel. State of the art IoT technology requires Scientists, Engineers and Technicians that are well versed in the field. It is important to utilise the long-term data generated from existing projects for educational purposes and to promote innovation and funding in the field of IoT (Miazi, et al., 2016; Zennaro, et al., 2012).

In South Africa, WRM is underfunded compared to other components of water management programs (Fell, et al., 2019). Researchers tend to get demotivated due to lack of funds, theft and vandalism. Thus, the idea behind the implementation of IoT technology in

developing countries, like South Africa, is to adopt existing IoT technology in an innovative way. By doing this, the unique problems present in developing countries can be considered with unique systems that are reliable and efficient.

2.3 Wireless Sensor Networks

2.3.1 WSN Technology

The idea of IoT was developed in parallel to WSNs (IEC, 2014). While IoT does not take on a specific WC technology, these technologies play a vital role in the make-up of IoT systems. WSNs have an application in many industries and can bring IoT to the smallest of physical objects, in any kind of environment, with its small, rugged, inexpensive and low powered sensors. WSN is defined as a network of nodes that together sense and can control the environment, while allowing interaction between humans or computers and the surrounding environment (Bröring, et al., 2011). The network of nodes in a WSN are devices that communicate with each other in an integrated system, allowing them to control the environment and the interaction between humans and the environment through computers and the internet. The main aim of WSN technology is to detect any events occurring in an area of interest and reporting it to the relevant person/authority. They support a vast range of applications, such as environmental monitoring, health monitoring, target tracking, system control, or exploration in dangerous environments (Soua & Minet, 2011). There are several different applications of WSNs, with the three most common being periodic sampling, event driven and store-and-forwards WSNs (Rhee, et al., 2004). Periodic sampling is used in applications that require continuous monitoring through sensor data acquired from multiple sensor nodes which are forwarded to a data collection centre on a periodical basis. Event driven monitoring requires the monitoring of one or more crucial variables and only sends transmissions when a certain threshold is reached. Store-and-forward monitoring occurs when sensor data is captured and stored, sometimes processed, by a remote node and is then transmitted to a central base-station. The critical requirements for the application of WSNs include:

- i. Long Battery Life
- ii. Small size
- iii. Low-data Rate
- iv. Low Cost
- v. Centralised Architecture.

WSNs have tiered structures through which data is transmitted until it reaches the intended recipient (Lazarescu, 2013). A simplified representation of a WSN tiered structure can be seen in Figure 2-2.

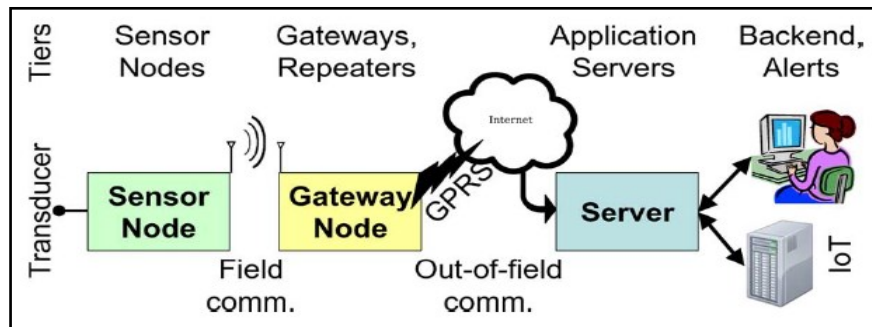


Figure 2-2: Simplified Representation of a WSN Tiered Structure (Lazarescu, 2013)

The tiered structure is a representation of a long-term outdoor WSN experiment. The sensor nodes take in-situ measurements of the surrounding environment and transmits those in the form of data packets to the gateway node through field communications. The gateway node takes the received packets of data and transmits these to the internet through out-of-field communications where it can be stored in a cloud server and then accessed by the end user through an application.

The past decade has witnessed a large reduction in the cost and power consumption of WC. This has opened new frontiers, especially in environmental monitoring (Bartos, et al., 2018). Periodic sampling WSNs are ideal for hydrological monitoring which requires low-cost, low-power, and low-maintenance systems which are flexible, reliable and robust. The application of WSNs in hydrological monitoring has been to great success – however, current implementations are experimental, and this results in a lack of interoperability, discoverability and standardisation. These are important attributes that are needed for these systems to be accepted as “critical infrastructure” by the hydrological resource and WRM community at large.

2.3.2 Network Topology

WSNs are made up of several nodes and they can be connected in different ways to form different network topologies. The four topologies that are discussed in this section include Star, Tree, Mesh and Hybrid.

One of the simplest topologies that sensors nodes can form is a star topology seen in Figure 2-3. The topology is formed by connecting the sensor nodes directly to a central connection point or gateway, with each gateway autonomously connected to the server (Lazarescu, 2013). The nodes only connect through the central gateway even if they are close in proximity. The star topology network setup allows single nodes to fail without disrupting the entire system and making the failed nodes easily identifiable due to the simplistic network (Pandya, 2013). Nodes are easy to replace, install or remove as it can be done without disturbing the entire system. The system is not expensive to implement due to the architecture sharing one central device, but it is heavily dependent on the central device and failure of this will result in the failure of the network (Mathews Modieginiane, et al., 2017).

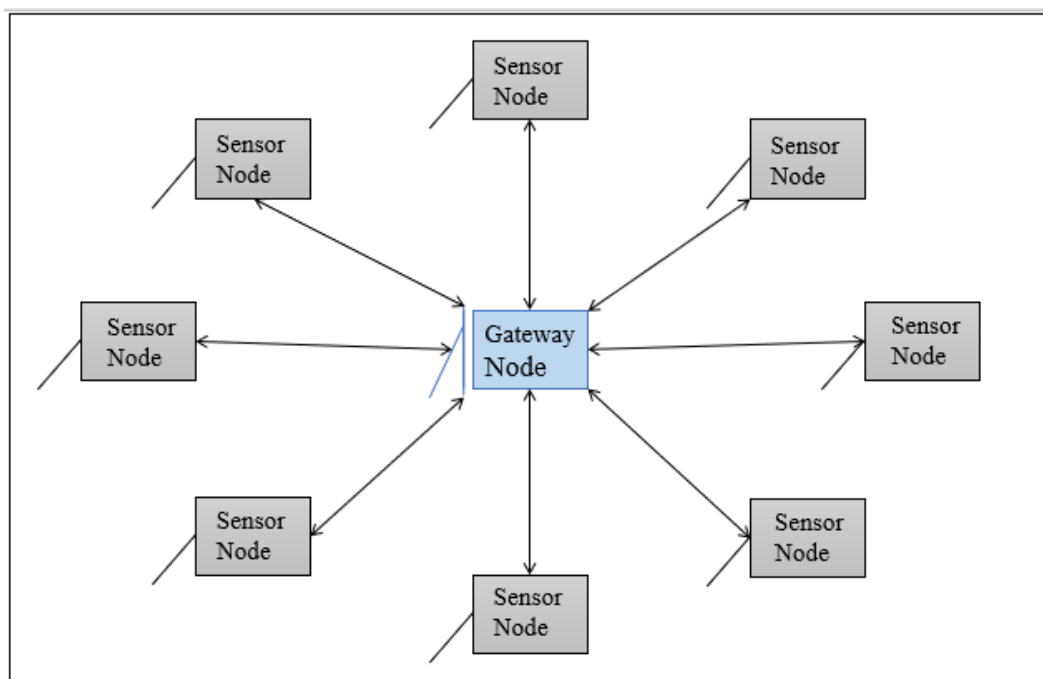


Figure 2-3: Star Topology

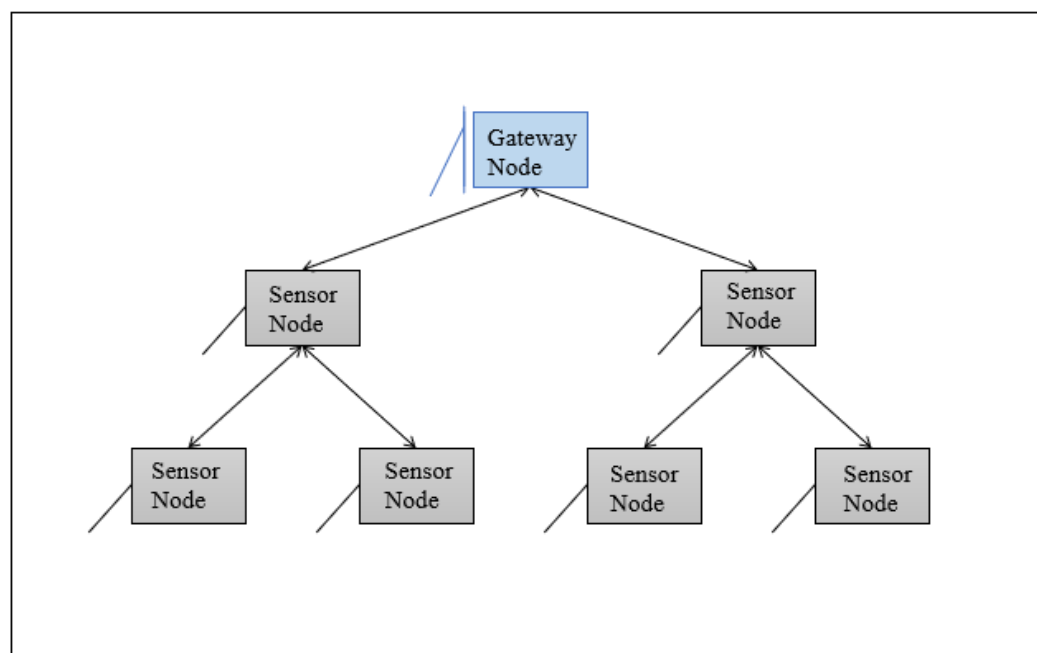


Figure 2-4: Tree Topology

The tree topology is more complex than the star topology and it gets its name from its branched structure, as seen in Figure 2-4. This topology is ideal for optimising power consumption or to extend the communication range of the network (Mathews Modieginnyane, et al., 2017). Scalability can be achieved in this system at the right level and the architecture can be easily managed and maintained since the network topology is segmented into branches. The lower structure of this topology is dependent on the root nodes. If the root nodes fail the lower structure also fails.

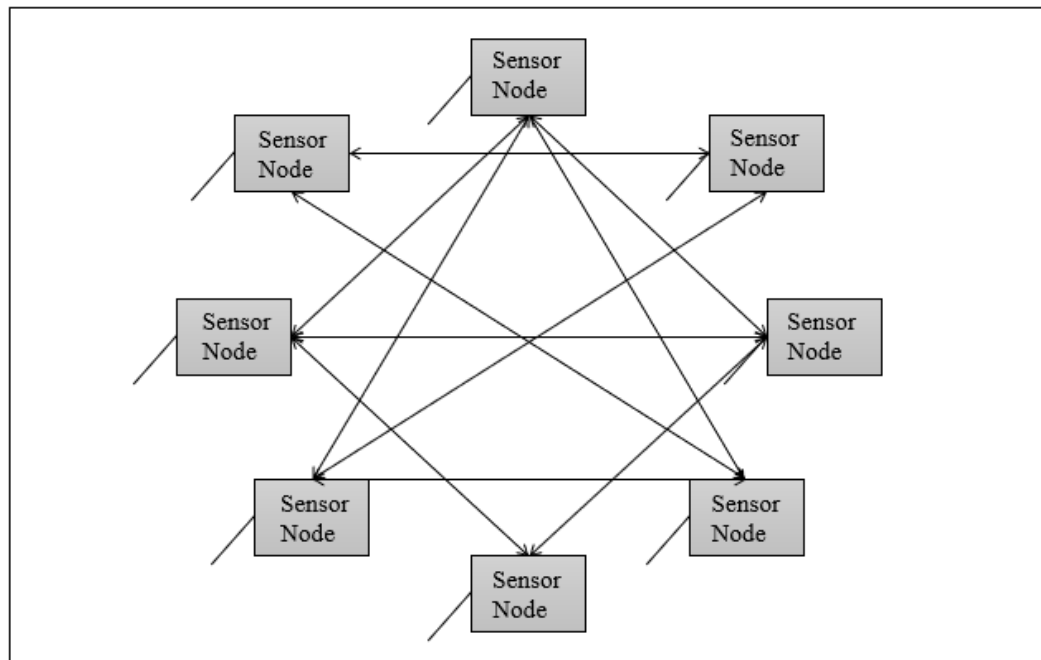


Figure 2-5: Mesh Topology

There are more complicated networks with the most notable being a mesh topology. A mesh topology provides alternate paths through which transmission of data can occur to balance the load of nodes in the network. The alternate paths are provided by node-to-node connection and can be seen in Figure 2-5. The increased number of paths yields the greatest amount of redundancy in the system due to transmissions having the option of taking several paths to its required destination (Pandya, 2013).

Hybrid topologies result from the combination of two or more of the above topologies (Mathews Modieginnyane, et al., 2017). The network implementation is generally reliable, scalable, flexible, and effective in its operation. The system does tend to be more expensive than the other network topologies since it is the combination of two or more network topologies that have different capabilities. Table 2-1 discusses the advantages and disadvantages of the different topologies mentioned.

Table 2-1: Advantages & Disadvantages of Network Topologies (Pandya, 2013; Mathews Modieginyane, et al., 2017; IntellegenInc., 2015; Inst Tools, 2020)

Topology Type	Advantages	Disadvantages
Star	Reliable transmission – data packets do not need to pass through many nodes	Highly dependent on a central node
	Centralised management aids in monitoring networks	
	Flexible	Number of nodes that can be added depends on capacity of central node
	Low-power consumption	
	Simple implementation	
Tree	High transmission range	Highly dependent on central node
	Secondary hubs allow more devices to be attached to central hub	Maintenance is difficult and expensive due to scalability factor
	Faults in network easily traced	Uneven power consumption across nodes
	Low-power consumption	Complex topology
Mesh	Network can handle heavy traffic	High power consumption
	Network problems easy to diagnose	Network is complex and hard to deploy
	Good system in terms of privacy	Yield a large amount of redundancy
	One node can send data to many nodes at the same time	Large number of input/output pins required
Hybrid	Fault tolerant	Expensive to deploy
	Low-power consumption	
	Reliable communication	Complex Design

2.4 MAC Network Protocols

Sensor nodes communicate through multi-hop paths over the wireless medium in a scattered dense and rough sensor field. Medium Access Control (MAC) protocol manages communication traffic on a shared medium and creates a simple network infrastructure for nodes to communicate with one another (Althobaiti & Abdullah, 2015). MAC protocol provides a self-organising capability to nodes which enforces singularity in the network by letting sender and receiver in node systems communicate with each other in a collision and error-free fashion. Basic functions that MAC is responsible for is channel access policies, scheduling, buffer management and error control (Alkhatib, et al., 2012). In WSN, a MAC protocol needs to consider energy efficiency, reliability, low access delay and high throughput as major priorities to accommodate the sensors' limited resources and to avoid redundant power consumption. Wasted Energy in MAC protocols are attributed to idle listening, collisions, overhearing, control packet overhead, over-emitting and complexity (Althobaiti & Abdullah, 2015).

Idle listening occurs when a node in a WSN system keeps its radio in ready-to-receive mode. This is due to the node/s not knowing when it will be the receiver of a message and this action is what consumes almost as much energy as when it is receiving a message. In low data traffic applications, this is considered the result of the frames being discarded because the receiver must drop the overlapped information. Collisions tend to result in retransmission which consumes most of the energy. Overhearing occurs when a node receives and processes a packet that is not addressed to it on the wireless broadcast medium. In dense networks and under heavy traffic situations, this scenario could lead to serious power consumption problems. Control packet overhead is caused by an increase in the size and number of control packets which results in an overhead and unnecessary energy waste for the WSNs, especially if there are low amounts of data transmitted in each message. These controls also tend to decrease channel capacity and a balanced approach is required to keep packets at a minimal. Over-emitting occurs due to the transmission of messages when the node that is supposed to receive it is not ready. Complexity of a system through computationally expensive algorithms may decrease the time that nodes spend in sleep mode. They may limit processing time available for the application and other functionalities of the protocol. Energy saving can be done by implementing an overly simple MAC algorithm compared to a complex algorithm. There is a trade-off between energy saving and the ability of the system to provide complex functions, such as adaption to traffic and topology conditions, clustering, or data aggregation.

MAC has two types of families of protocols, namely schedule based and contention based. Schedule based protocol uses Time Division Multiple Access (Alkhatib, et al., 2012). This type of protocol needs previous knowledge of network topology to establish. The advantages of this protocol include no collision, predictable delay and increase in overall throughput; while the disadvantages include the protocol not being good for large networks, not scalable, needs precise synchronisation, and requires expensive hardware and large overhead.

Contention based protocol uses Carrier Sense Multiple Access (CSMA) or Advocates of Linux Open-source Hawaii Association (ALOHA) and there is no need for synchronisation or topology knowledge because nodes compete to access the channel and only the winner succeeds. The advantages of the protocol include being good for large scale systems while the disadvantages include less performance for high load traffic, data packet sizes are usually small, and Request to Send/Clear to send are more energy consuming.

2.5 Low-Powered Wide Area Network (LPWAN)

IoT systems require measured data to be transmitted to a database for storage purposes and this is done through communication technology. The communication technology used in a WSN are wireless transceivers and they provide connectivity to transmit and receive data between different components in a WSN system (Yelamarthi, et al., 2017). These communication devices have different ranges and can be grouped as either short range or long-range transmission. Short range technologies include ZigBee and 6LowPAN, while long-range technologies include LoRa, Sigfox, NB-IoT, weightless, Long-Term Evolution (LTE) - Machine type communication etc. (Khutsoane, et al., 2017). Figure 2-6 shows a graph of the relationship between the rate of transmission and the range of transmission of the different communication technologies. Many WSN applications require flexibility and mobility. To ensure these characteristics are present in the system, the preferred wireless technology applied should be low-cost and low-power consumption (Fernandes Carvalho, et al., 2019).

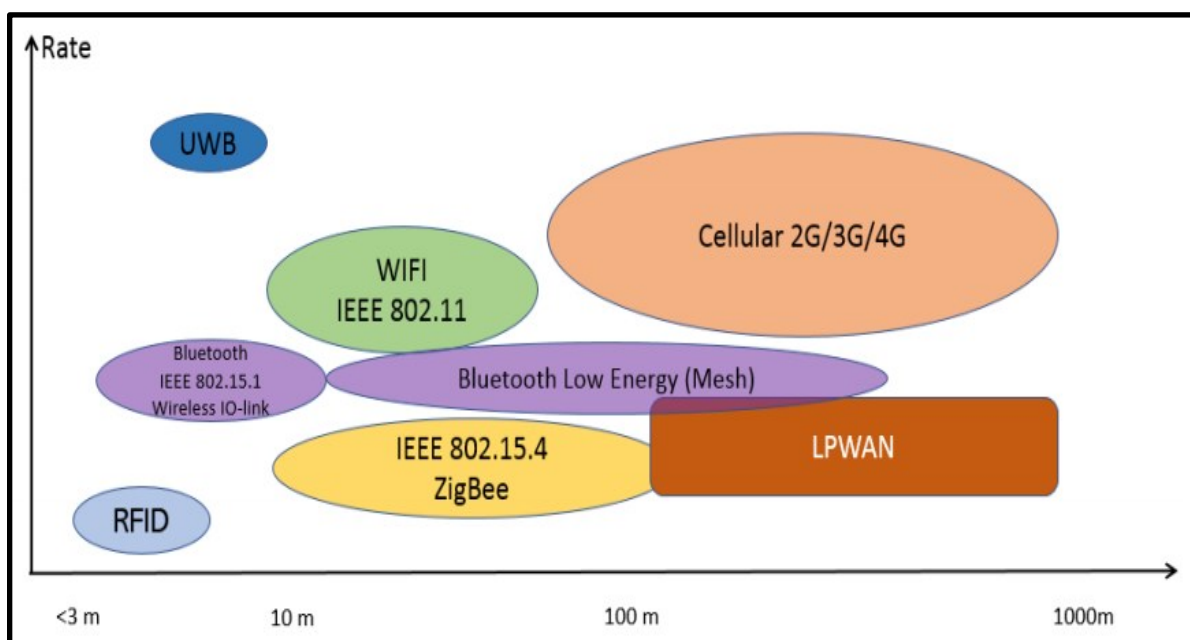


Figure 2-6: Rate vs Range of Communication Technologies (Tsavalos & Hashem, 2018)

Low-Powered Wide Area Network (LPWAN) is a low-power consumption network that has a long transmission range. LPWANs operate in wireless bands that are both licensed and unlicensed. According to Khutsoane *et al* (2017), the main characteristics of a LPWAN that guide the design of a WSN include the following:

- i. Low-power consumption
- ii. Low-cost devices for low-cost network deployment
- iii. Secure
- iv. Extended coverage
- v. Easy to deploy network infrastructure.

LPWAN technologies are ideal for devices that are required to send small quantities of data over a long range with specific constraints around power consumption and computational power (Augustin, et al., 2016). Many of the technologies perform well in less dense environments which make them ideal for hydrological monitoring in remote areas (Saravanan, et al., 2017). LPWAN include technologies such as LoRa, Sigfox and NB-IoT.

2.5.1 LoRa

LoRa is a recently introduced LPWAN technology. It provides lower throughput than previously mentioned technologies which makes it ideal for WSN systems that require minimal data transmission and low-power consumption (Saravanan, et al., 2017). LoRa is a long range, low power wireless platform that is popularly used in the IoT space (Semtech, 2021). Base stations or single gateways can cover entire cities or hundreds of square kilometres (LoRa Alliance, 2020). The long range and lower power capabilities make LoRa applicable to deployments that require long range data transmission while maintaining power efficiency of end-devices (Augustin, et al., 2016). These deployments generally involve Machine to Machine (M2M) and IoT networks.

LoRa protocol has been developed by Semtech and is a proprietary protocol. It is the physical layer protocol that is often used together with LoRa Wide Area Network (LoRaWAN) MAC-layer protocol (Olsson & Finnsson, 2017). LoRa technology uses a proprietary spread-spectrum modulation technique derived from the existing Chirp Spread Spectrum technology (Semtech, 2019). It operates in a fixed-bandwidth channel of either 125 KHz or 500 KHz (uplink/downlink). This type of modulation is very effective in achieving low power consumption. Orthogonal Spreading Factor (SF) allows the network to preserve battery life of end nodes by making adaptive optimizations of individual end node data rates and power levels (Semtech, 2019). This optimization is done by transmitting data from end nodes close to gateways at a low SF since little budget is needed.

The opposite can be said for nodes far away from the gateway. Optimising these nodes would result in a high SF that increases processing power and reception sensitivity. The SF and parameters for each factor can be found in Table 2-2.

Table 2-2: Parameters for LoRa spreading factors (Semtech, 2019)

Spreading Factor (For UL at 125 KHz)	Bit Rate	Range (Depends on Terrain)	Time on Air for an 11-byte payload
SF10	980 bps	8 km	371 ms
SF9	1760 bps	6 km	185 ms
SF8	3125 bps	4 km	103 ms
SF7	5470 bps	2 km	61 ms

LoRa channel frequencies can operate on 433-, 868- or 915 MHz Industrial, Scientific, and Medical (ISM) bands. Table 2-3 provides the channel frequency plan per region for the commonly used plans.

Table 2-3: LoRa common frequency plans (The Things Network, 2021)

Channel Plan	Common Name	Region
EU863-870	EU868	Europe
US902-928	US915	USA
CN779-787	CN779	China
EU433	EU433	Europe
AU915-928	AU915	Australia
CN470-510	CN470	China
AS923	AS923	Asia
KR920-923	KR920	-
IN865-867	IN865	-
RU864-870	RU864	-

LoRa modulation characteristics differ based on the region of deployment (Semtech, 2019). Table 2-4 gives the different LoRa modulation characteristics. The data rate can reach 50 Kbps when channel aggregation is called while the payload size of each transmission can range from 2-255 octets (Augustin, et al., 2016).

Table 2-4: LoRa common frequency plans (Semtech, 2019)

Data Rate (DR)	Spreading Factor (SF)	Channel Frequency	Uplink or Downlink	Bitrate (Bits/Sec)	Maximum user payload size (Bytes)
0	SF10	125 kHz	Uplink	980	11
1	SF9	125 kHz	Uplink	1760	53
2	SF8	125 kHz	Uplink	3125	125
3	SF7	125 kHz	Uplink	5470	242
4	SF8	500 kHz	Uplink	12500	242
5-7	-	-	-	-	-
8	SF12	500 kHz	Downlink	980	53
9	SF11	500 kHz	Downlink	1760	129
10	SF10	500 kHz	Downlink	3125	242
11	SF9	500 kHz	Downlink	5470	242
12	SF8	500 kHz	Downlink	12500	242
13	SF8	500 kHz	Downlink	21900	242

The modulation technique is a proprietary technology from Semtech (2019) and, thus, LoRa is not open source. The networks have two main advantages, namely signal propagation and minimal power consumption (Khutsoane, et al., 2017). Signal propagation gives the network the ability to alter signal frequency, optimising the signal for the exact scenario needed, while minimal power consumption provides the system with years' worth of life.

2.5.2 LoRaWAN

LoRaWAN protocol is used to wirelessly connect battery powered devices to the internet in regional, national or global networks (LoRa Alliance, 2020). The protocol aims to meet key IoT requirements, such as bi-directional communication, end-to-end security, mobility, and localisation services. LoRaWAN is an open-source protocol and was developed by the LoRa Alliance. It runs on top of the LoRa physical layer and provides the MAC mechanism that provides communication between multiple nodes and network gateways (Haxhibeqiri, et al., 2018).

2.5.2.1 Network Topology & Architecture

LoRaWAN networks consist of star-of-stars topology through which gateways relay messages between node devices and a central network server (LoRa Alliance, 2020). This type of network topology enables connection between end-devices and LoRaWAN gateways but does not allow direct communication between end-devices.

A LoRaWAN network must have at least a network server, a gateway, and an end-device (Ram, 2018). The end-device is a sensor or actuator that measures a specific parameter from the surrounding environment and connects to one or many gateways over LoRa. Gateways receive data from the end-devices and are connected to the network server through Internet Protocol (IP) links such as 2G, 3G, 4G dongles or Wi-Fi, while end-devices use single-hop LoRaWAN communication to one or many gateways in MAC protocol.

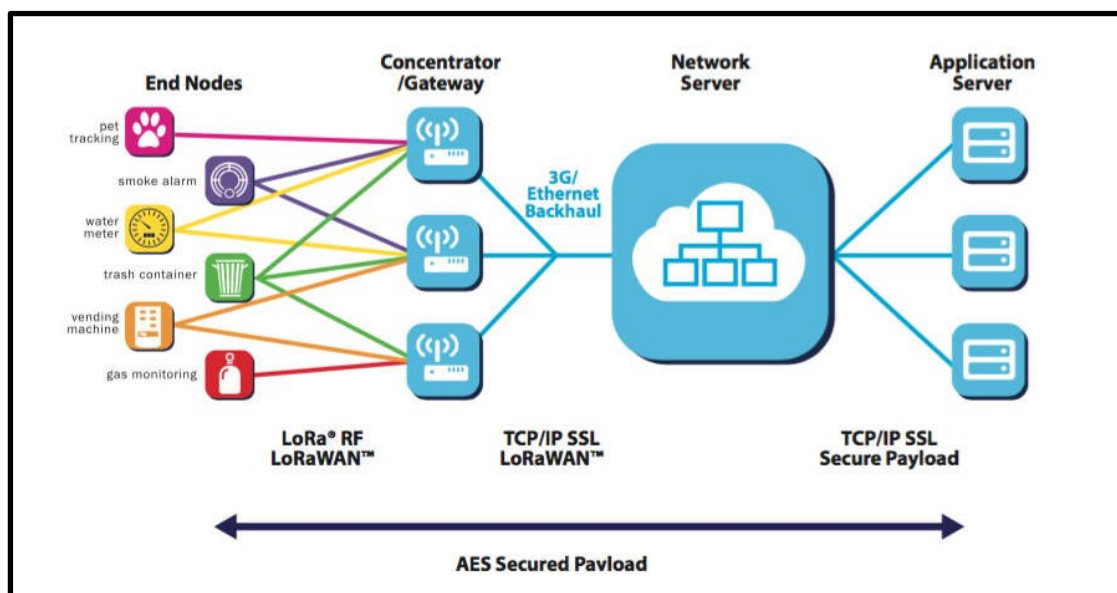


Figure 2-7: Typical LoRaWAN network architecture (The Things Network, 2021)

Using these IP links, gateways forward packets of data to a cloud server and the data is made available to an end-user/application by the network server (LoRa Alliance, 2020). This makes the network structure simple, efficient, and robust, making it ideal for IoT communication. Figure 2-7 depicts a typical LoRaWAN network architecture.

The protocol and network architecture have a major influence in determining node battery lifetime, network capacity, quality of service, security and several applications served by the network (Khutsoane, et al., 2017). Ideal WSN systems provide connectivity at low operational and maintenance costs, and LoRaWAN networks satisfy these constraints. Solar panels can also be connected to the LoRa devices for energy harvesting providing the devices with virtually unlimited life spans.

2.5.2.2 Device Classes

LoRaWAN devices can be split into three different classes. These three classes allow end point devices to address the specific needs required by the wide range of applications (LoRa Alliance, 2020). The three device classes can be classified as follows (LoRa Alliance, 2020; Semtech, 2019; Shiferaw, et al., 2020):

Class A is the default class supported by all LoRaWAN end-devices. This class has the lowest operating power consumption and allows bi-directional communication or network control commands if necessary. The network operates with an ALOHA type protocol and class A communication is fully asynchronous. The communication is always initiated by the end-device. Class A devices have the lowest power operating modes while still allowing uplink communication at any time. The end-devices can enter low-power sleep mode as defined by its own application. This makes this class the most energy efficient and the most suitable for battery powered sensors.

Class B devices are bi-directional end-devices with deterministic downlink latency. The devices use periodic beacons to synchronise with the network to open downlink periodic reception windows, also known as ping slots, at scheduled times (LoRa Alliance, 2020; Shiferaw, et al., 2020). This gives the devices the ability to send downlink communications with a deterministic latency. However, this requires additional power consumption from the end-device. The increase in battery power consumption is still low enough to effectively use battery powered nodes while the latency of these devices is programmable up to 128 seconds. This ensures the compatibility of these devices in different applications.

Class C devices are also bi-directional end-devices but have the lowest latency. Class C devices are constantly listening for downlink messages unless they are transmitting an uplink message. To allow the device to constantly listen for downlink messages with no latency, the end-device receiver is always open. This requires a large power consumption and thus class C devices are advised to be powered with a continuous power supply to obtain a workable life span. Battery powered devices can be used but temporary mode switching between class A & C is required to keep the end-device energy efficient.

2.5.2.3 *Security*

Security is very important when it comes to IoT deployment using LPWAN. LoRaWAN specifications define two cryptography layers (LoRa Alliance, 2020):

- i. 128-bit Network Session Key (NwkSKey) shared between end-device and network server
- ii. 128-bit Application Session Key (AppSKey) shared end-to-end at application level.

The NwkSKey is used to verify the integrity and authenticity of packets sent to the LoRaWAN network. The AppSKey is used to encrypt or decrypt the application payload once the data has reached the application server. By providing these two levels of security, ‘multi-tenant’ shared networks can be implemented without the network operator having visual access to the user’s payload data. Including the NwkSKey and the AppSKey, there are multiple security keys used in LoRaWAN and the keys and their explanations can be found in Table 2-5.

Table 2-5: LoRaWAN Security Keys (LoRa Alliance, 2020)

Key Name	Explanation
DevEUI	64-bit unique ID assigned to end-device by the manufacturer
DevAddr	32-bit ID which identifies the end device within the current network
NwkAddr	End device address within the network
AppEUI	Unique 64-bit ID of the application server
AppKey	128-bit number - Encryption key between the source of the message and the destination of the message (Unique for each device).
NwkSKey	Network Session Key - Verifies integrity and authentication of packets sent to LoRaWAN network
AppSKey	Application Session Key - Encrypt/decrypt application payloads once data reaches application server

The security level of a LoRaWAN network depends on the activation method. A LoRaWAN network can be activated by Activation By Personalisation (ABP) or by Over-The-Air-Activation (OTAA). ABP allows activation on the production line or during commissioning, while OTAA allows devices to be re-keyed if necessary.

ABP activation allows end devices to skip the join procedure by hard coding a fixed DevAddr and session keys for a pre-selected network into an end device. These keys remain the same throughout the ABP devices lifetime (The Things Stack, 2021). ABP end devices use a fixed security session, and these session keys cannot be changed or rotated which poses security threats if the keys get leaked. ABP end devices also use fixed network parameters and when registering an end device, a RX1 Delay, a Data Rate Offset, a RX2 Data Rate Index, RX2 Frequency and a list of Factory Preset Frequencies are required.

OTAA activation involves a join procedure with a LoRaWAN network, during which a dynamic DevAddr is assigned to an end device. OTAA end devices are provisioned with root keys and these are used to derive session keys (The Things Stack, 2021). DevAddr and sessions keys are dynamic and are changing constantly when establishing each new session. OTAA devices require the end device to be within coverage of its registered network when attempting to join. This is because end device needs to be able to receive the join accept downlink message from the network server to join.

2.5.2.4 *LoRaWAN in Africa and South Africa*

LoRaWAN specifications differs from region to region. These specifications have been defined for Europe, North America, Australia, and Asia but are yet to be defined for the continent of Africa. Africa's LoRa frequency plan is EU863-870/EU433 and according to legislation in South Africa, there is a 1% duty cycle placed on networks operating within the unlicensed frequency spectrum (ICASA, 2015: 279). According to The Things Network (2021), there are 182

LoRaWAN gateways connected all around South Africa with 7 active communities in the country and over 200 active people in these communities.

2.5.3 The Things Network

The Things Network (TTN) is an open-source public network equipped with infrastructure that allows LoRa devices to transmit messages through the network. Access is available to anyone and, thus, awareness needs to be raised in terms of security related risks. The backend systems of TTN are responsible for routing IoT data between devices and application. This connection between device and application is made possible through LoRaWAN technology.

LoRaWAN allows the connection to the internet without 3G/4G/5G or Wi-Fi, while also providing low battery usage, low bandwidth, and long-range transmission. These characteristics make the technology suitable for IoT application. TTN connection begins with low-power devices connecting to long range LoRa gateways which are connected to the internet through an open-source network. The network exchanges data with the users' applications as illustrated in Figure 2-8.

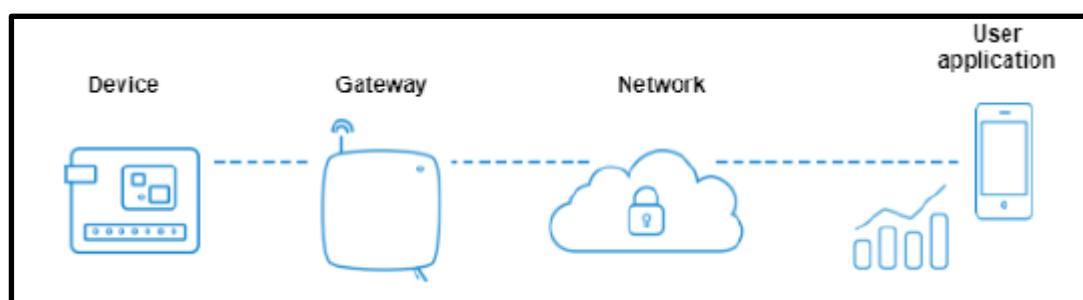


Figure 2-8: Tiered Structure of LoRaWAN communication (The Things Network, 2021)

2.5.4 Things Speak

Things Speak is an IoT analytics platform that allows the user to collect, aggregate, visualise, and analyse live data streams in the cloud. The platform collects data measured by sensors and uses MATLAB to analyse the data to discover relationships, patterns, and trends in the data.

Things Speak uses an Application Programming Interface (API) to update their channels and to chart numeric data stored in the channels. An API allows the creation of an application using a set of functions and procedures in the form of code (Sagdeo, 2018). The application can then access the features or data of an application, operating system or other service. Things Speak allows for the use of two APIs to update its channels, namely Representational State Transfer (REST) API and Message Queuing Telemetry Transport (MQTT) API (MathWorks, 2020).

REST is an architectural style which functions as a request-response model using HTTP communication. GET, POST, PUT and DELETE are used by Things Speak to create and delete

channels, read and write channel data, and clear the data in a channel. MQTT utilises publish/subscribe communication protocol that makes use of transmission control protocol/IP sockets or WebSocket's. MQTT allows a client device to connect to the broker and publish to channels or subscribe to receive updates from different channels.

2.5.5 Sigfox Communication

Sigfox offers an end-to-end LPWAN based on its patented Ultra Narrow Band (UNB) technologies. Sigfox network operators function with numerous proprietary gateways that are equipped with intelligent software-defined radios and connects them to the backend server using an IP-based network (Raza, et al., 2017). The base stations are deployed in many countries and operate in the unlicensed sub-GHz ISM bands (Mekki, et al., 2018). The employment of UNB technologies in the sub-GHz spectrum allows Sigfox to efficiently use the frequency band and has very low noise levels (Mekki, et al., 2018; Raza, et al., 2017). The low noise levels lead to a lower power consumption, higher receiver sensitivity, and low-cost antenna design. However, these benefits come at an expense with the maximum throughput of 100 bits per second. This reduces the number of use-cases of Sigfox as it falls at the lower end of throughput offered by most LPWAN technologies. Sigfox initially started with an uplink transmission only and only later it was developed to incorporate bidirectional communication with downlink transmission only occurring after uplink transmission occurs (Mekki, et al., 2018). In South Africa, Sigfox network is managed by Squidnet and comes at an annual cost while the network is limited to 140 messages per day with a maximum payload length for each downlink message being 8 bytes (Pretorius, 2019).

2.5.6 NB-IoT Communication

NB-IoT is a narrow band LPWAN technology which coexists in LTE or GSM under licensed frequency bands (Mekki, et al., 2018). NB-IoT functions at a frequency bandwidth of 200 KHz, which allows it to operate under stand-alone, guard band and in-band operation. Stand-alone operations use GSM frequencies bands while guard bands and in-band operations use LTE. The preferred integration technology is LTE with most NB-IoT communication protocol being based on this protocol. NB-IoT offers different trade-offs between cost, coverage, data rate and power consumption (Raza, et al., 2017). These trade-offs make this technology application specific.

2.6 IoT Hardware

2.6.1 Microcontroller Unit

Microcontroller Units (MCU) are tiny self-contained computers hosted on a single integrated circuit or microchip (Bhavsar, 2017). The MCU in a WSN system is usually a low-power embedded processor whose primary role is to capture sensor data, perform data analysis, provide

feedback to the user as necessary, and relay the information to the internet gateway (Yelamarthi, et al., 2017). It has several functions, such as collecting and managing data from sensors, performing power management functions, interfacing the sensor data to the physical communication layer and managing the communication network protocol (Martin & Islam, 2012). Processors that are used in WSN systems are often based on node size and power consumption more than processing capacity due to the lower processing power needed in most WSN applications. They can be powered by a battery or an energy saving module such as a solar panel.

The basic structure of an MCU includes a Central Processing Unit (CPU), memory and circuitry that works together to implement a number of functionalities (InternalZero, 2018). The CPU is the “thinker” as it performs the arithmetic, logic and data-orientated functions; the memory stores information in the form of Read Only Memory (ROM) and Read Access Memory (RAM); and the circuitry executes the Inputs/Outputs (I/O). MCUs used in WSN application are characterised by their CPU, Analog-to-Digital Converter (ADC), Clock speed (MHz), Bus width (bits), Memory (kB), Input/Output connectivity and battery operation (Yelamarthi, et al., 2017). These characteristics are used to decide on the type of MCU to use in WSN application to ensure optimum performance of the system.

2.6.2 Sensor Devices

From an engineering point of view, sensors are defined as devices that transform physical, chemical, or biological parameters into an electrical signal (Bröring, et al., 2011). Examples of sensors range from temperature measuring sensors (thermometers) to solar or radiation measuring sensors. They form the most basic unit of sensor technology while a sensor system is the formation of a network of sensors by connecting a collection node to a single platform (Stasch, et al., 2009). There has been continuous improvement in sensor technology which has allowed it to become cheaper, smaller, more power efficient and more intelligent (Bröring, et al., 2011). This has increased the applicability of sensors as more fields are able to utilise this technology. The aim of a sensor device is to detect events or changes in an environment, and then supply a corresponding electrical current (Das & Jain, 2017). The most important characteristics of a sensor device when detecting events or changes are precision, resolution, linearity, and speed. They are particularly useful in environmental monitoring due to its low cost, long maintenance-free service time and its simple and reliable deployment procedure (Bartos, et al., 2018).

Hydrological data that can be obtained through sensor devices include *inter alia* water levels, precipitation, evaporation, surface soil moisture, surface soil temperature, atmospheric temperature, relative humidity, snow water equivalent, and water quality (Holzbecher, et al., 2019). Sensors are application-specific and there are different sensors that are utilised across several fields. These include healthcare, building ambience, structural health and

agricultural/hydrological sensor (Yelamarthi, et al., 2017). Bartos et al. (2017) gives the following examples of hydrological sensors:

- i. Ultrasonic and pressure-based water level sensors
- ii. Soil moisture sensors
- iii. Tipping-bucket and optical rain gauges
- iv. Automated grab samplers for pollutant loads
- v. In-situ water quality sensors, including probes for DO, pH, temperature, conductivity, dissolved solids, and oxidation-reduction potential.

These also include internal sensors that monitor the health statistics of the sensor. These internal sensors aid in diagnosing network problems and allow remote configuration of the sensors through the internet. This capability allows users to remotely switch sensors on/off and change sampling frequencies without having to reprogram the device in the field (Bartos, et al., 2018).

2.6.3 Sensor Nodes

Sensor nodes in a WSN are miniature devices that consist of four main components: a sensing device for data collection; an MCU or Microprocessor (MPU) for data processing and memory functions; a communication device which allows the transmission and reception of data to and from connected devices; and, lastly, a power source which is usually battery powered (Soua & Minet, 2011). The sensing device oversees collecting and transforming vibration, chemical and light signals in an environment into electrical signals and transmitting them to an MCU or MPU (IEC, 2014). The MCU/MPU then receives the data from the sensor and processes the data before it is transferred by the Radio Frequency (RF) communication, so that the data can become a physical, tangible form of communication to be displayed to the end-user on a particular application.

Size and weight are particularly important to ease deployment while energy consumption is the major factor to consider when trying to increase the maintenance-free service of the system. Lazarescu (2013) divides the energy consumption of sensor nodes into five parts:

- i. RF communication, for data and network maintenance.
- ii. Processing, *e.g.*, transducer data, self-checks, Real Time Clock (RTC).
- iii. Sensing, *e.g.*, transducer supply, calibration.
- iv. Safety devices, *e.g.*, watchdog timer, brown-out detector.
- v. Power down energy required by the node components in low power consumption mode.

The last three depend mostly on component selection while RF communication and processing depend mostly on the protocol and the application, respectively. Battery powered nodes may improve cost and reliability if energy consumption can be satisfied with a small battery that lasts the duration of the node's lifetime.

2.7 Powering IoT Systems

The application scenarios of IoT, specifically WSNs, often involve battery powered nodes that need to be active for long periods of time without external human control after initial deployment (Soua & Minet, 2011). Without energy efficient techniques, a node would drain a battery within a couple of days. The most challenging concern when implementing a WSN design is power consumption. A WSN design is considered viable if the system can save node energy while maintaining the desirable network behaviour. Power consumption is based on both the hardware and software of a system, with the choice of hardware components and programming protocol playing a major role (Al-Kofahi, et al., 2019).

2.7.1 Battery Powered

Energy stored in the form of batteries has been the presiding source of energy when providing power to IoT-driven WSNs (Raj & Steingart, 2018). The popularity of battery-based power solutions comes from the early applications of IoT systems. These early systems spent most of their life cycle in sleep mode, only waking up to perform a measurement or transmit messages across low-power wireless connections, such as Bluetooth low energy (Parker, 2016). The applications of IoT have diversified, resulting in traditional power supply options becoming inadequate. This has forced the rapid progression of stored energy in the form of batteries to meet newer, more diverse applications of IoT.

There are many different batteries that can be used to power IoT-driven devices. Alkaline, Lithium and Nickel Metal Hydride (NiMH) batteries are three alternative battery options for supplying IoT devices with power (Young, 2008). Alkaline batteries are low-cost workhorse batteries that are widely available and are ideal for low current applications. The drawbacks of the battery are its inability to operate under cold conditions and its inability to perform under high current draws. The NiMH battery is similar to alkaline batteries and is the powerhouse of rechargeable batteries. They tend to maintain their characteristics and performance after many charges. The drawbacks of the battery are that the power to material ratio is small, making the battery relatively heavy while it also has a lower density level.

The Lithium battery is used as a replacement for alkaline batteries where longer life, higher current draw and improved temperature performance is required. There are many different types of Lithium batteries and, while these batteries share certain common characteristics, it is difficult to put them in the same class. The Lithium battery's ability to offer high performance and reliability has made it the commonly used energy storage option to power IoT devices. IoT devices require batteries that are light and compact also having a high voltage and high energy density (Joannin, 2020).

2.7.2 Energy Harvesting

Another power supply option for powering IoT devices involves energy harvesting from the environment. Energy harvesting is applicable to IoT systems that monitor the environment and are placed outdoors. These systems, with the right components, can continuously harvest energy from the surrounding environment. Many of the harvesting techniques have been developed for long term IoT node operation using solar cells, flow systems and mechanically and thermally based harvesting systems (Raj & Steingart, 2018).

Solar energy is harvested using solar cells and it is considered one of the most efficient energy harvesting techniques to power IoT devices (Shirvanimoghaddam, et al., 2019). The silicon based solar cells utilise the energy from sunlight to power the IoT devices. These systems must be exposed to sunlight during periods of the day to generate power. This can make power generation inconsistent. Implementing solar energy-based power supply in regions susceptible to theft and vandalism will require these systems to be out of sight or inaccessible to the public. Flow systems draw their power from rotational movements from air or water (Raj & Steingart, 2018). These energy harvesting techniques are generally utilised at large scales through mega structures like wind turbines or hydroelectric dams. Like in solar cells, the major drawback of the system is related to the inconsistent power generation. This, coupled with the large-scale nature of the energy harvesting technique, limits its applicability to small scale IoT systems. Thermal energy uses a thermal energy generator to convert temperature gradients into energy to generate power (Raj & Steingart, 2018; Shirvanimoghaddam, et al., 2019). Devices powered by thermal energy require cyclical temporal variations to generate power supply, making the power supply limited and inconsistent (Shirvanimoghaddam, et al., 2019). Mechanical energy-based power generation utilises the energy generated from vibrations, pressure, and stress-strain to produce power (Shirvanimoghaddam, et al., 2019). Powering nodes that require continuous power would require continuous vibrations, pressure or stress-strain (Raj & Steingart, 2018).

Energy harvesting in IoT systems is based on the requirements and application of the system. The use of renewable energy roughly doubles the overall cost of the basic IoT node and deployment of the systems may not be possible in all scenarios (Cerchecci, et al., 2018). Implementing an IoT-driven system powered by energy harvesting requires thorough investigation of the benefits and limitations of the system in its specified use-case.

2.7.3 Energy Efficient Techniques

Energy represents the main constraint in all WSNs as the sensor nodes are limited to energy from a limited power battery. In order to reduce power consumption in WSNs, the focus needs to be put on the data transmission and data processing of the sensors in the system (Baadache & Adouane, 2015). The main reason for sensor power consumption is due to the electrical communication processes. The amount of energy consumed in each communication increases by an increase in the amount of data transmitted, and an increase in the distance between nodes and the collision between nodes. A network's lifetime has become a key characteristic for evaluating

WSNs and their ability to deliver real-time monitoring of systems. Energy efficient techniques have been developed to reduce energy consumption and, in-turn, improve network lifetime. Classifications of these techniques have given five main classes, namely data reduction, protocol overhead reduction, energy efficient routing, duty cycling and topology control (Soua & Minet, 2011).

Data reduction aims to limit the amount of data produced, processed and transmitted. Protocol overhead reduction is a technique which looks to increase the protocol efficiency by reducing the transmitted overhead messages. Energy efficient routing is designed with the aim of enhancing the network lifetime by reducing energy consumption of the end-to-end transmission and avoiding nodes with low residual energy. Duty cycling aims to minimise the amount of time the node is unnecessarily active during its lifetime. Duty cycling is implemented by developing sleep/active schedules for the nodes based on specific application requirements. Topology control focuses on minimising energy consumption through the adjustment of transmission power while ensuring network connectivity. Table 2-6 gives an indication of the impact the five energy efficient techniques have on sources of energy wastage in a WSN. The symbol ‘M’ indicates main impact while the symbol ‘S’ indicates secondary impact.

Table 2-6: Impact of energy efficient techniques (Soua & Minet, 2011)

Reasons of Energy Waste	Sensing-Process	Communication	Collision	Overhearing	Control – Packets	Idle - Listening	Interference
Techniques							
Data Reduction	M	M	S	S	-	-	S
Protocol Overhead Reduction	-	M	S	S	M	-	S
Energy Efficient Routing	-	M	S	M	S	-	M
Duty Cycling	M	M	M	S	M	M	M
Topology Control	-	M	M	M	-	-	M

2.8 Application of IoT in Hydrology

Hydrologists are focused on data collection and producing models based on data collected to predict future hydrological related events (British Hydrological Society, 2014). The application of IoT can be the link in providing reliable hydrological data that can be used to develop accurate predictive models that help in improving WRM. Advances in IoT technology have improved the efficiency and decreased the cost of implementing IoT devices for hydrological monitoring. These low cost, low power devices built and developed with off-the-shelf components have been developed in multiple regions across the world.

Low-cost sensors were deployed in the Dodder River catchment in Dublin, Ireland (Zhang, et al., 2019). The Dodder River flows through Churchtown and Dundrum before joining the River Liffey at the Dublin port before entering Dublin Bay. The sensors deployed were off-the-shelf Kingspan Watchman Anywhere Pro ultra-sonic sensors that allowed two-way communication for sending data to the internet and receiving instructions remotely. The sensor node was made up of a GSM/GPRS module for wireless communication, four alkaline batteries for power, a control board and the nodes were housed in UV stabilised housing. 11 sensors were originally deployed in the South Dublin region with water level readings taken every 15 minutes. Due to vandalism, only eight sensors operated for the duration of the deployment with none of the eight sensor nodes requiring maintenance. The study found that the implementation of these sensors provided a way to monitor water level of an open water body at a level that is almost as accurate as traditional stations, but at a fraction of the cost.

In Colima, Mexico, a sensor node network was setup in the Colima River catchment located in the north of the state (Mendoza-Cano, et al., 2021; Moreno, et al., 2019). The network consisted of multiple building blocks that collected and measured hydrological data, including water level node network which was named RiverCore. RiverCore sensor nodes consisted of an ultrasonic sensor, a 32-bit MCU, 3G cellular modem electronic board for communication, a regulated power supply, a solar charge controller and a 12 V 80 Ah battery. The nodes were housed in a IP65 enclosure, and these nodes were deployed at 14 different points along the Colima River and its tributaries. There were three major findings from this research. Firstly, the study highlighted the importance of reducing the measuring and storing time to allow for enough time for the system to establish connection between the hardware and data acquisition web platform. This significantly reduces the data loss of the system. Secondly, during this study, 3G/2G communication was found to suffer from interference and strong variations in signal in remote locations. Data pools or data files to store the measured data in the device's memory were recommended to combat this issue. Thirdly, further recommendations looked at improving power efficiency by implementing stand-by and sleep modes for the nodes during dry periods of the year and making the nodes compatible with commercial weather stations to ensure improved data transmission.

Ragnoli *et al* (2020) developed and implemented a flood-monitoring system based on LoRa technology. The sensor node of the system consisted of a custom electronic board with a

Atmel SAMD21 MCU, a RFM95W LoRa module, the water level sensor and a 3.3V battery interface housed in a wall mount sensor enclosure. The system was tested using flood laboratory simulations and researchers found that LoRa as the choice of wireless technology and the sensor node developed proved to be an energy efficient and effective IoT flood monitoring system.

PROMAS, a meteorological monitoring station, measured parameters such as direction and wind speed, rainfall, temperature, humidity, barometric pressure, and water level of rivers in the national park EL Cajas, in southern Ecuador (Guaman, et al., 2018). The aim of the research was to integrate a water level monitoring system to improve the accuracy of the prediction station by implementing real-time transmission, low-energy consumption, user friendly presentation and handling of data, programmable sampling period, portability, and variable sampling frequency. The proposed system was built using the LoPy4 board from Pycom with the board having a Espressif ESP32 dual core chipset and LoRa, Sigfox, Wi-Fi and Bluetooth transmission capabilities. The sensor used to measure water level was a WL400 submersible pressure transducer and the system was powered by a 7200 mAh – 12 V solid acid battery coupled with a 30 W solar panel. According to several tests, the design and implementation of the water level sensor integrated to the PROMAS station was successful with tests being carried out both in the laboratory and in the field.

The Smart Environmental Monitoring and Assessment Technologies (SEMAT) project aimed at developing and creating an open standard for low-cost, near real-time, remote aquatic environmental monitoring systems based on IoT (Trevathan & Johnstone, 2018). The sensor node, housed in a buoy, measured multiple parameters including temperature, turbidity, and water level. Communication occurred between the node and base station using 3G networks and GSM communication. The sensor node consisted of an Arduino Pro Mini ATmega328P MCU and was powered by 3.3V Lithium-ion Polymer (LiPo) batteries that were recharged with a 6 V 1 W solar panel. As part of the project, the buoys were deployed in Lake Ellerslie, an artificial lake forming part of Griffiths University's Logan Campus in Brisbane, Australia. The study suggested that all the goals set to date were met. The field tests demonstrated that the latest developments for the SEMAT project were well suited for this type of application, and this bodes well for future IoT application in water quality monitoring. The low-cost of the sensors provided potential for wider scale community involvement through school initiatives and other associated community networks.

IoT application in hydrology is scarce in South Africa. A study looked at developing and deploying low-cost flow sensors to measure water level and flow rates of a river downstream of informal settlement in Franschhoek, South Africa (Fell, et al., 2019). The aim of the study was to develop a low-cost flow sensor that produces accurate and actionable water data continuously. The low-cost sensor was built using an MCU, a communication device, battery, battery management system, and an ultra-sonic distance sensor. The sensor communicated using GSM and was powered by a Li-Ion battery.

The results from the study suggest that the sensor provided high-quality continuous data and provided new insights into the dynamics of flow and pollutants in a river that drains an informally settled catchment.

2.9 Summary

Conventional methods of hydrological monitoring do not allow for real-time decision making and this can cause errors in hydrological models and design decisions. IoT is a novel technology that connects all types of things to the internet. IoT systems are built on an architecture consisting of five layers – perception, network, middleware, application, and business layer. There are endless opportunities for IoT implementation in Africa, and specifically South Africa, as there is a need for data that is not fragmented, inaccurate and isolated to one measurement. WSNs form the basis of IoT systems and operate in a tiered structure that transmits data measured by the sensor to the internet. The formation of a WSN consists of multiple nodes communicating through a network protocol. The nodes can be arranged in varying topologies with the most effective depending on the application. LPWAN communication is the ideal communication technology for IoT application as it provides nodes with the ability to send small packets of data over long distances with the most popular options for IoT application being LoRa, Sigfox and NB-IoT. LoRa is a low-power, long range wireless network that uses LoRaWAN protocol for multi-node communication and sends data to the internet using TTN, an open IoT ecosystem that allows LoRa messages to be sent to the internet. Once messages are sent to the internet, they can be stored and visualised on applications such as Things Speak. IoT hardware includes MCUs and sensor devices that are integrated to create sensor nodes. The MCU is the brain of a node and performs most of the processing while the sensor devices measure the physical world. Powering IoT devices is a major point of interest for researchers. IoT devices require power for long periods without interruption. The most common power supply options in IoT include battery power which can be coupled with energy harvesting and energy efficient techniques to increase battery life. Studies suggest that there are many examples of IoT application for hydrological monitoring, including water level monitoring of rivers, flood monitoring systems and multi parameter hydrological monitoring systems.

3. Conceptual Model

IoT in hydrological monitoring requires a system that is reliable, robust and energy efficient (Trevathan & Johnstone, 2018; Ragnoli, et al., 2020; Mendoza-Cano, et al., 2021; Moreno, et al., 2019; Guaman, et al., 2018). This section details the representation of the system and how the system operates at the architectural level. It provides an overview of the functions at every layer of the system. Figure 3-1 depicts the operational flow diagram of the conceptual model.

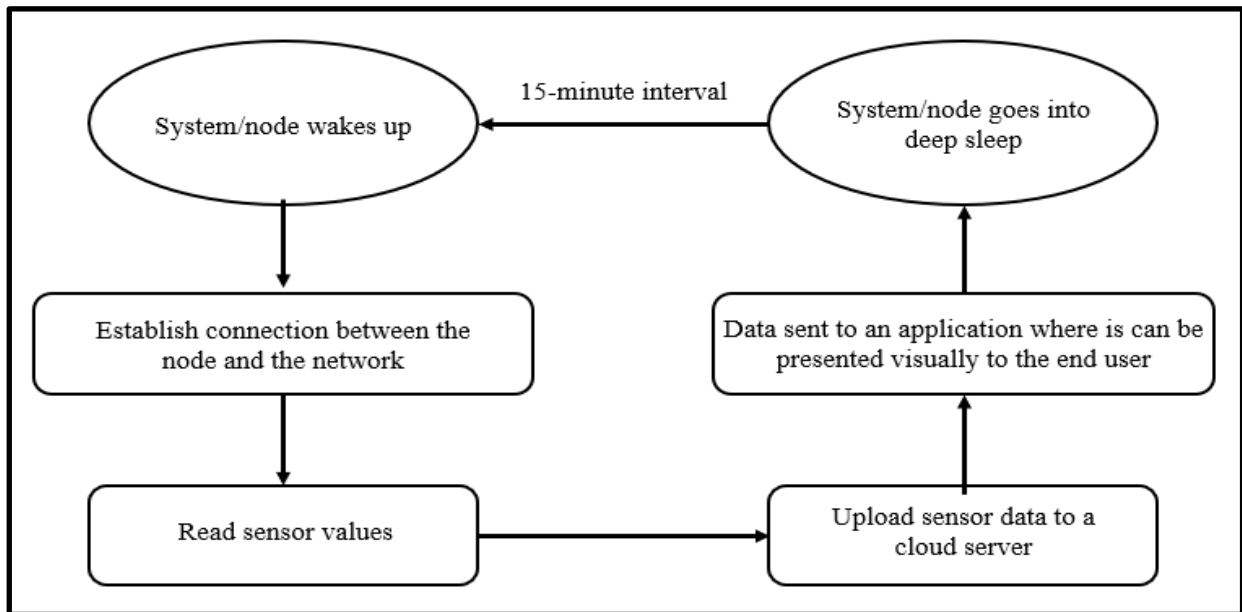


Figure 3-1: Operational Flow Diagram for Conceptual IoT Model

3.1 Perception Layer

The system begins at the perception layer, also known as the physical layer, which involves devices that sense the surrounding environment. The hydrological parameters measured are based on the available technology. In this study, the following hydrological measurements were investigated using the following sensor devices: water temperature, water level, soil moisture, soil temperature, air temperature, and relative humidity. The devices were connected to sensor nodes comprising of MCUs. Each sensor has a specific connection protocol which governs how they are integrated to the MCU. These MCUs are the processing power of the system. They are programmed to convert the signals received from the connected sensors into readable data that can be transmitted, stored, analysed, and visually displayed at later stages within the architectural layers. Figure 3-2 shows sensor node setup of the system at the perception layer.

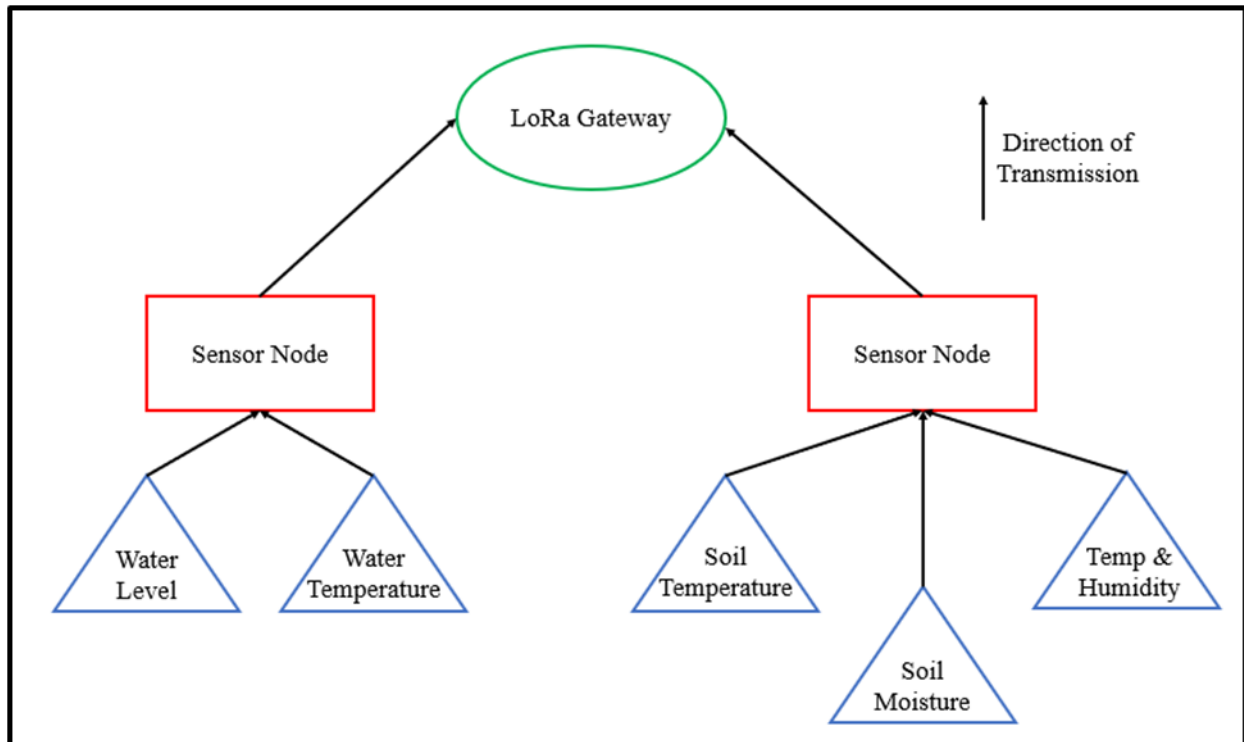


Figure 3-2: Proposed Sensor Node Setup of the System at the Perception Layer

3.1.1 Sensor Node

The sensor node is made up of all the hardware components of the system. This is where all hardware components are connected and integrated with one another. The MCU functions as the brain of the node as all the software programming will be stored within its memory. The MCU is mounted on a customised Printed Circuit Board (PCB) which allows integration between the sensors and the MCU. The PCB provides terminals through which the sensors measuring hydrological data can be connected and read by the MCU. The nodes, driven by the MCU and using a LoRa antenna, will packet the data collected from the sensors and send this in the form of payloads to any listening LoRa gateways in the area every 15 minutes. The selection of the MCU and PCB is dictated by energy efficiency and whether these components are compatible with the available sensors that are measuring the hydrological data required for this research.

3.1.2 LoRa Gateway

LoRa gateways form part of the perception layer as they are physical devices that receive data from the sensor node and send the data to the internet via TTN. The gateways used in this project are setup by TTN community members located in the City of Cape Town (CoCT), South Africa. These members form part of the Western Cape TTN community, and their locality can be seen in Figure 3-3. There are 54 active gateways in the community with most of them offering coverage over CoCT and its surrounds.

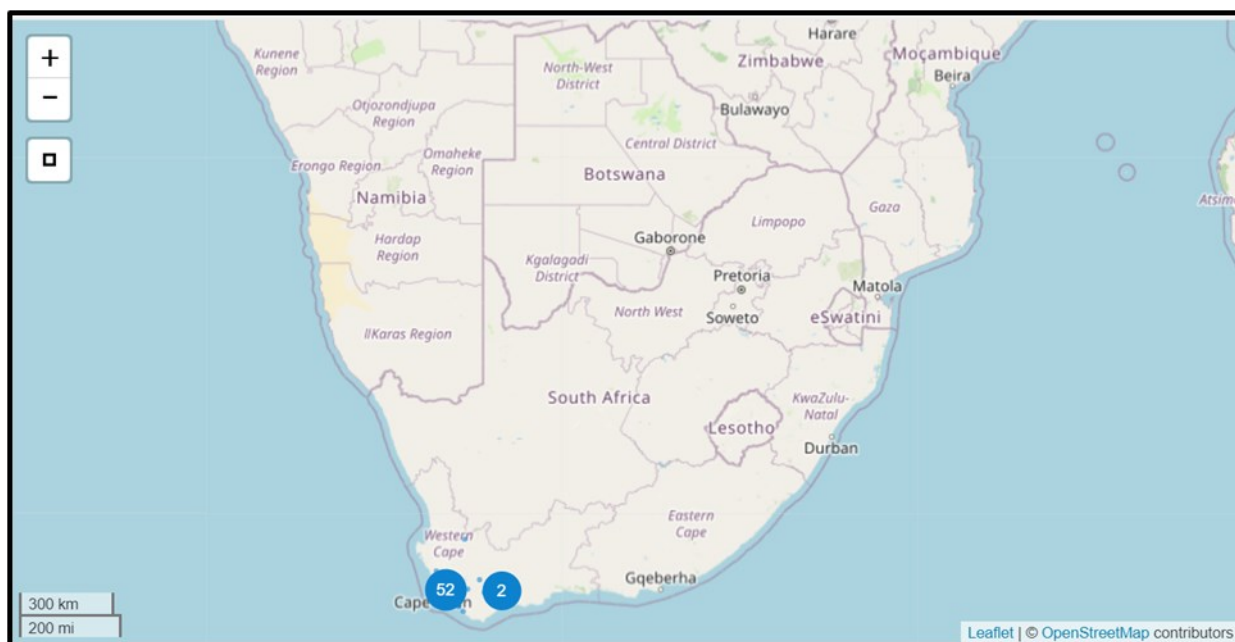


Figure 3-3: Locality of Western Cape TTN community in S.A. (The Things Network, 2021)

Existing LoRa gateways, including a gateway set up at UCT, will be used to transmit data to the internet. Figure 3-3 shows the LoRa coverage available in CoCT and its surrounds. These gateways provide coverage for the transmission of data. Their purpose is to receive incoming data in the form of LoRa messages and forward it to a database that can decrypt information and store the data. The process is visualised by the TTN website, and this application allows the user to track any node activity, which includes uplink and downlink transmissions.

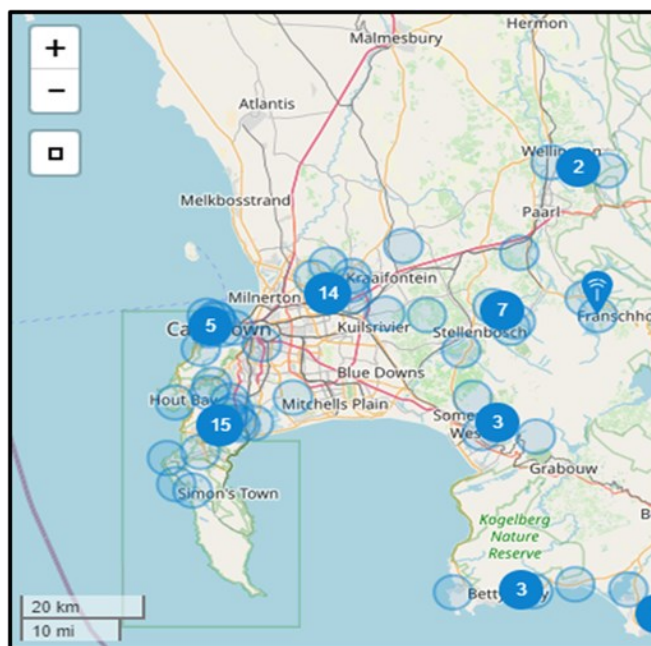


Figure 3-4: LoRa coverage across the City of Cape Town (The Things Network, 2021)

3.2 Middleware Layer

The Middleware Layer of the system is made up of software that operates the system and allows it to function as required. Software is used to prepare and send data to the internet through a network. The middleware layer is governed by Micro Python, which is a condensed version of Python specifically used for programming microelectronics. Micro Python is particularly useful for smaller computers, such as MCUs, as it ensures the MCU performance is optimised and fit for purpose for the application it will serve (Zennaro, 2018). The MCU for the sensor node is programmed so that the system measures data, packages the data and then forwards the data to a gateway which transmits the data to the internet. All scripts written for this project are loaded onto the MCU and govern the function of the node. An open-source text and source code editor, referred to as an Integrated Development Environment (IDE), allows the user to edit and change programming scripts in multiple different languages using extension packages.

The software that controls the sensors and the sensor nodes is known as Object Oriented Programming (OOP) and this was written using Micro Python. To read the data measured by the sensors, multiple already-existing micro python libraries need to be referenced and where there were no available libraries, new libraries are developed. The software developed and used prompts the sensor to wake up, take and send measurement readings to the MCU. The script is edited and developed on the chosen IDE and once the script is finalised it is uploaded to the MCU of the sensor node using the IDE.

3.3 LoRa Network

For this research, the network layer of the system is driven by LoRa communication. The LoRa network operates using LoRaWAN protocol that is built on top of LoRa modulation. LoRa communication uses LoRaWAN protocol which is a cloud-based MAC layer protocol that manages communication between LPWAN gateways and end-node devices by acting as a network layer. The LoRaWAN protocol of the system manages the format in which messages are sent. The protocol allows the system to transmit small size payloads over long distances while using ultra low power.

The MAC for the network uses a contention-based protocol. The two possible contention-based technologies are ALOHA and CSMA. These protocols allow nodes to transmit messages to the gateway without considering the availability of the channel (Pretorius, 2019). LoRaWAN networks make use of an ALOHA based protocol. This protocol allows for the implementation of a simple star topology network that has multiple nodes transmitting data to an active listener in the form of a gateway node. The contention-based protocol is scalable, making the protocol effective in large-scale systems.

3.3.1 LoRa Network Software

The software that governs the sending of data via LoRa encrypted messages to TTN (LoRa Cloud Server) is written in Micro Python and uses LoRaWAN protocol. Before data can be sent to TTN, the end device needs to connect to the network. The connection process involves Micro Python script that allows the end device to join TTN. Once joined, the data that has been packaged in payloads is sent to the TTN server.

3.3.2 LoRa Application

Once data has been successfully sent to TTN, the data is decrypted using JavaScript programming, stored, and displayed via a LoRa compatible application called Things Speak. Things speak is an IoT analytics platform and gives the user the opportunity to aggregate, visualise and analyse data that is sent from the sensor nodes of the system. Things Speak is integrated with TTN and receives data directly from the network. The data is displayed in interactive graphs and exported for further analysis using excel.

3.4 Summary

The conceptual model outlines the functions of the system throughout each architectural layer. The sensor node formed the perception layer and was responsible for measuring and transmitting hydrological data. The programming language that formed the basis of the middleware layer is Micro Python. Micro Python scripts prompts sensors to measure hydrological parameters and package them to send through to the internet. Once measurements are taken, the messages are sent to the internet using LoRaWAN protocol along with LoRa gateways that are situated in and around the CoCT. These messages were sent to TTN and then onto Things Speak where they were stored, visualised, and exported to other supported software for analysis.

4. Methodology – System Design

4.1 Overview – System Design

The aim of the study was to design a system that collects real-time hydrological data using IoT and WSN. The study implemented an IoT driven monitoring system that used LoRa communication to capture and display data in a user friendly and easily accessible manner. Figure 4-1 shows an overview of the method of the system design used in the study based on various studies (Pretorius, 2019; Rmsm, et al., 2019 ; Akhila & Senthil Prabha, 2019; Fell, et al., 2019; Guaman, et al., 2018; Mendoza-Cano, et al., 2021; Moreno, et al., 2019; Zhang, et al., 2019; Ragnoli, et al., 2020; Trevathan & Johnstone, 2018). Appendix A provides a step wise summary of the system design.

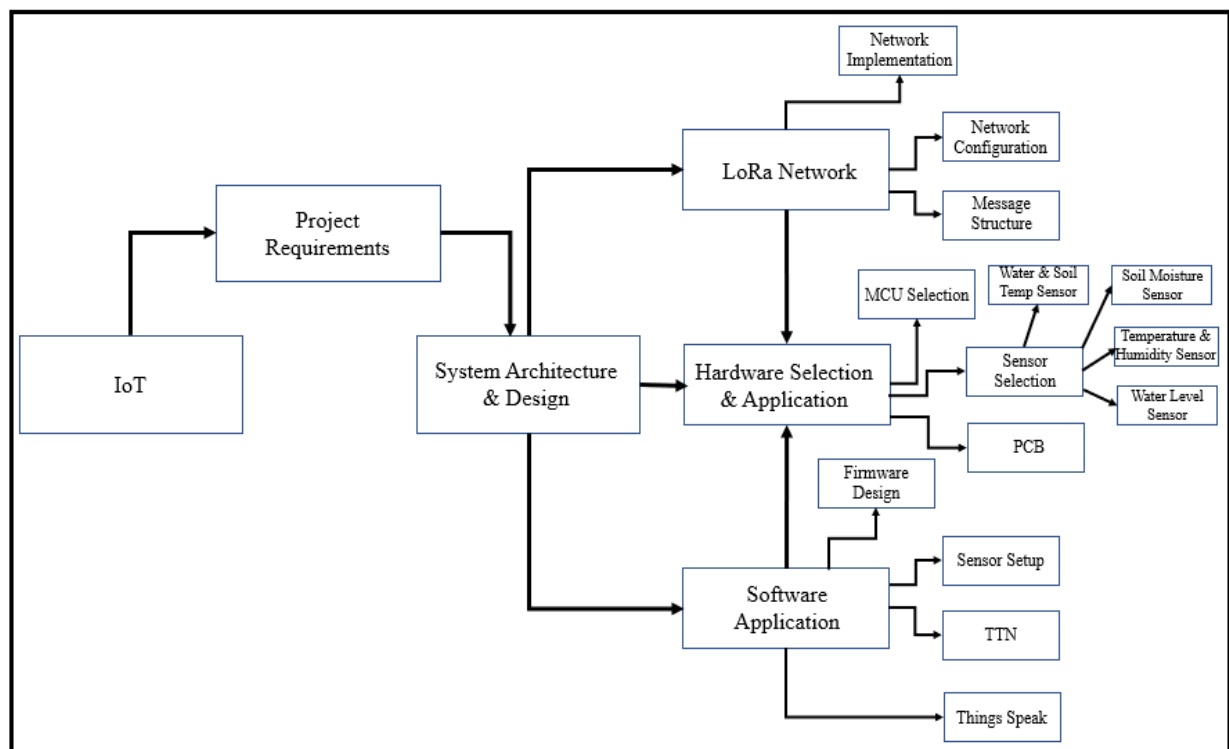


Figure 4-1: Overview of Methodology – System Design

This chapter consists of six sections including an overview, statement of method, project requirements, LoRa communication, hardware selection/application, and software application.

4.2 Statement of Method – System Design

The study investigated the design of two IoT-driven hydrological monitoring systems that measured data in real-time using LoRa communication. The design of the systems included developing requirements and selection of suitable architecture.

Developing the system requirements of the project involved ensuring that the set-up accounted for key aspects needed to collect and convey hydrological data. The key aspects include *inter alia* availability, reliability, performance, scalability, and interoperability. The study mainly focused on reliability, performance, and data output and integrity. These aspects govern the design of developing a system that meets the requirements necessary for an IoT driven hydrological monitoring system. The adopting of the system architecture involves integrating the aspects of the design, the network application, hardware selection/application, and software application.

4.3 Project Requirements

The study integrated the aspect of reliability to account for proper working of the system based on its specifications (Macedo, et al., 2014). The aim of a reliable system is to increase the success rate of IoT service delivery. It is important for this to be implemented in both software and hardware throughout the IoT layers (Al-Fuqaha, et al., 2015). An unreliable system can lead to long delays, loss of data, and eventually wrong decisions, which can lead to possible disastrous scenarios. This is of particular importance when utilising IoT systems for real-time hydrological monitoring. Figure 4-2 provides the key aspects relevant to IoT testing for this research.

The performance of an IoT system is based on several different components and can be difficult to evaluate (Al-Fuqaha, et al., 2015). It can be based on its ability to obtain the required data at a level that meets the user or customer's satisfaction. There are many metrics available to assess the performance of these systems, including processing speed, communication speed, device form factor and cost (Al-Fuqaha, et al., 2015). In terms of hydrological monitoring, the performance of the system must consider its ability to produce data in a timely manner that can be visually interpreted through user-friendly applications.

Data Output and integrity testing forms the basis of an IoT system. An IoT system applicable to hydrological monitoring is analysed based on the type of data it can measure as well as the integrity of the data measured. As mentioned in Chapter 2, hydrological monitoring involves measuring many different hydrological parameters and the integrity of the data is measured by validating the accuracy and precision.

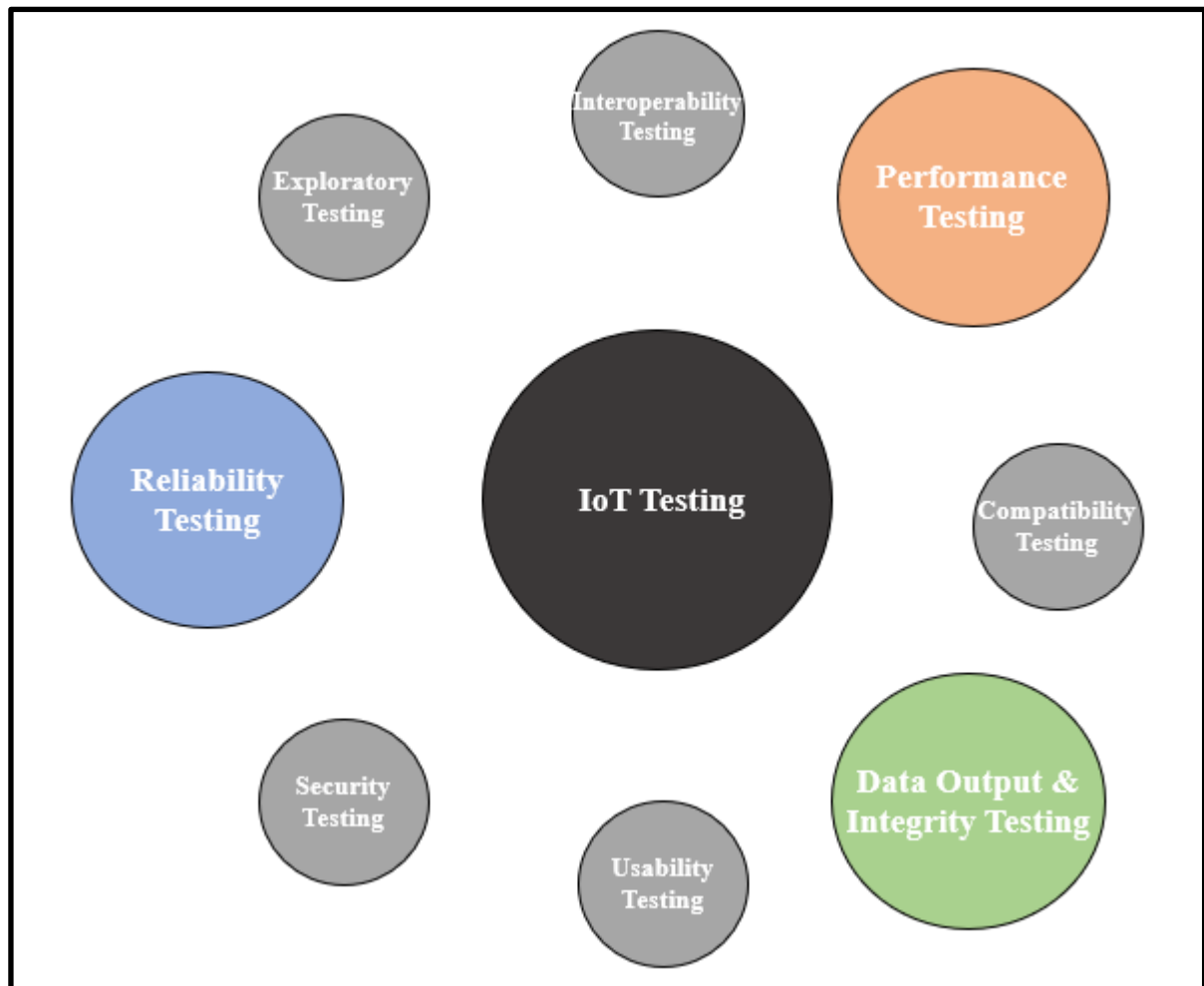


Figure 4-2: IoT Testing (Data Flair Team, 2018)

4.4 LoRa Communication

4.4.1 Network Implementation

LoRa implementation begins with the development of a LoRa node. An antenna was used to transmit data to and from the node. The antenna allowed the node to send and receive data in the unlicensed band and operated in a frequency range of 137-1020 MHz. The LoRa node needed to be equipped with a LoRa transceiver which allowed the node to interact directly with the LoRa network.

To transmit data from sensor node to the internet, the node first needed to connect to a LoRa gateway. The gateway acted as a communication bridge between the node and the cloud storage. In this study, the LoRa node connected to the LoRa network using existing LoRa gateways that cover the upper campus area of UCT which can be seen in Figure 4-3. Further LoRa coverage in the CoCT can be seen in section 3.1.1. The gateway setup on UCT upper campus had the ability to send and receive LoRa data. Once the data reached the LoRa network, it was forwarded to a backend platform or application that was used to store and then access the data.

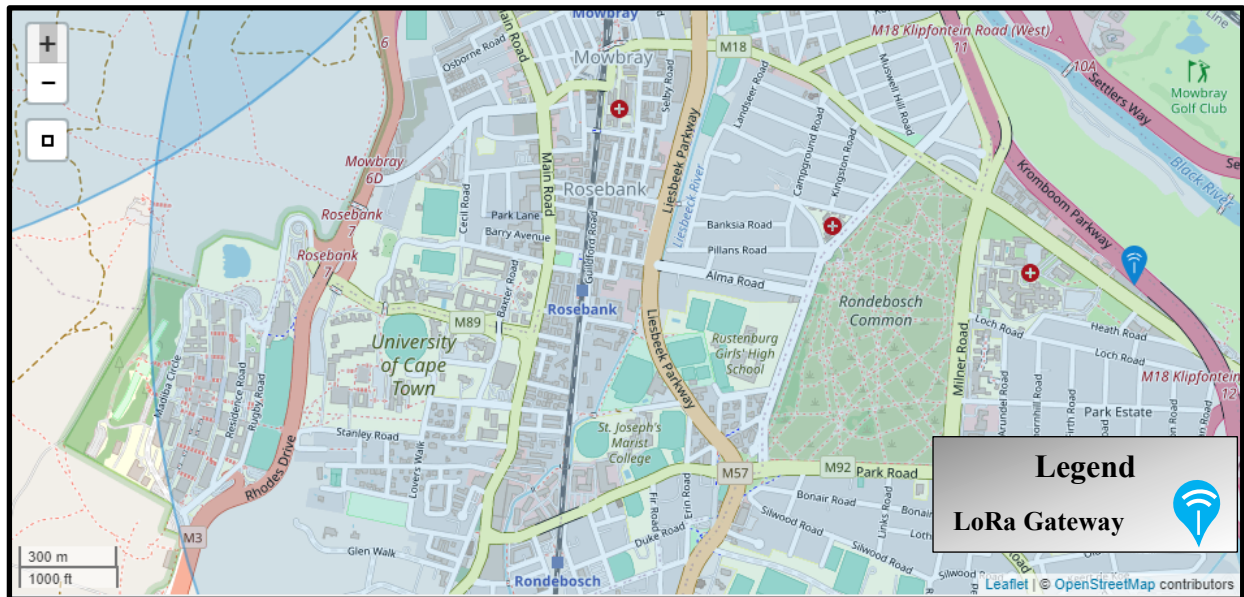


Figure 4-3: LoRa coverage over the UCT area (The Things Network, 2021)

4.4.2 Network Configuration

The network configuration is governed by a form of MAC applied to the system. The LoRaWAN networking protocol was implemented in this study to wirelessly connect the battery-operated systems to the internet. The network can transmit messages between end-devices and a central network server through a gateway. The network architecture is applicable to a single node, as well as multi-node deployment using a star-of-stars topology (LoRa Alliance, 2020). The LoRa gateways were connected to the network server via standard IP connections and behaved as transmission bridges that converted RF packets to IP packets and vice versa. The LoRaWAN end device utilised the 1.0.2 LoRaWAN version available on The Things Stack section of TTN. The end device was implemented as the default class A type of device, allowing the end device to only initiate communication with uplink messages.

The end device ran on the Europe 863-870 MHz (SF9 for RX2) frequency plan. This frequency band is available across Africa and South Africa. As per the Independent Communications Authority of South Africa (ICASA) - The Radio Frequency Spectrum Regulations (2015), the device had to have a duty cycle below 1%. The device used OTAA by storing the AppEUI, DevEUI and AppKey on the end device and sending join requests using these keys to the network until the device was connected to the server.

To maximise efficiency during end-to-end application integration, the LoRaWAN protocol offers several helpful practices for LoRa application. These practices involved limiting the payload lengths of messages to a maximum of 20 bytes and using a SF of 9. Using SF9 during device initialization maximised communication efficiency, even under difficult radio conditions. The repetition mechanism allowed up to three transmissions to increase the radio performance.

4.4.3 Message Structure

LoRa Alliance (2020) indicates that the message structure should be governed by the following 4 factors:

- i. How often data should be sent
- ii. The data format.
- iii. Precision in digits required
- iv. Kind of Payloads that should not be sent.

The aim of the system was to sense data and transmit the measurements to the LoRa network every 15 minutes. The type of messages that can be transmitted by the LoRa network include uplink and downlink messages. Uplink messages are sent from the end-device to the network. Uplink messages use the LoRa radio packet explicit mode in which both the LoRa physical header (PHDR) and the Cyclic Redundancy Check (PHDR_CRC) are included (LoRa Alliance, 2020). The CRC protects the integrity of the payload. The PHDR, PHDR_CRC and the payload CRC fields are inserted by the radio transceiver. Figure 4-4 gives an example of the uplinks physical message structure.

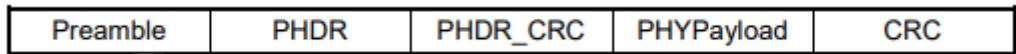


Figure 4-4: Uplink Message Structure

Downlink messages are sent by the network server to the end-device and if there are multiple end-devices, it is sent to one end-device which broadcasts the message to the other devices by a single gateway (LoRa Alliance, 2020). Downlink messages use the radio packet explicit mode in which the LoRa physical header (PHDR) and a header CRC (PHDR_CRC) are included. Figure 4-5 gives an example of the downlink message structure.

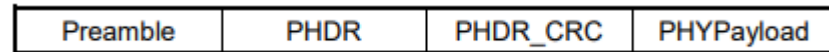


Figure 4-5: Downlink Message Structure

To maximise efficiency of the data packet transmissions, packet optimisation best practices were implemented when transmitting messages. By synchronising sensors and sensor transmissions, the systems could send multiple measurements in one message (LoRa Alliance, 2020). Group measurement reduced the number of messages in the system which reduced power consumption of the system. Bit packing was another critical aspect in maximising data transfer, which involved transferring data using the smallest number of bits. To bit pack effectively, the appropriate scale and precision was provided. Appendix B shows a variety of bit-packing examples for common physical and other measurements. As seen in Table A-1, bit packing for temperature requires 8 bits. The aim of the system was to use bit packing to reduce redundant information in the system which, in turn, improves efficiency for data transmission.

4.5 Hardware Selection and Application

This section presents the selection criteria of the hardware components and application in this study.

4.5.1 Microcontroller Device

Before the selection of the development board, many different boards were considered for this study. The main aim in selecting a board was based on its compatibility with LoRa communication. Many boards were analysed and then narrowed down to two mainstream development boards, namely the Arduino MKR WAN 1300 and the Pycom LoPy 4. The Pycom LoPy 4 is compatible with both LoRa and Sigfox communication, while the Arduino MKR WAN 1300 is only compatible with LoRa. Both these boards have potential for further node development because of their ability to offer major functionality and power capabilities. Side by side comparison of the two development boards can be seen in Appendix C-1.

The Arduino MKR WAN 1300 has an embedded Murata CMWX1ZZABZ LoRa module and supporting libraries to interface with the LoRa network. The Pycom LoPy 4 has an embedded Semtech SX1272 LoRa chip which also supports libraries to interface with the LoRa network. Programming language for Arduino is written in C++ while the programming language for Pycom is written in Micro Python. Due to its compatibility to LoRa as well as its programming language being Micro Python, the Pycom LoPy 4 board was chosen. Pycom has an active online community and many open source Pycom support libraries which aided in the development of this study. Figure 4-6 depicts an annotated diagram of the Pycom LoPy4 development board.

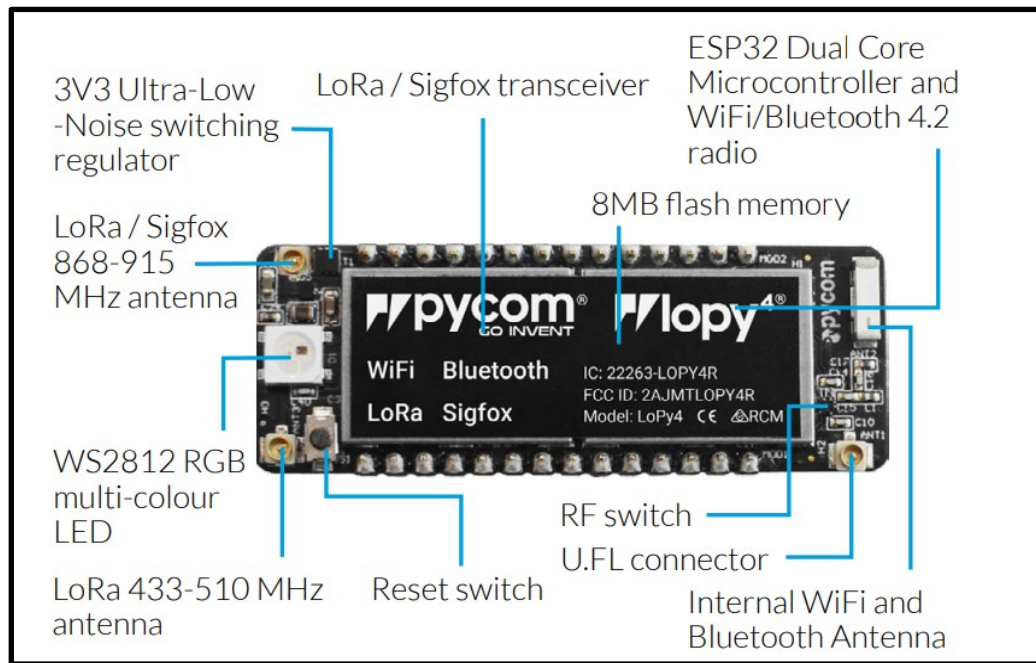


Figure 4-6: Annotated diagram of the Pycom LoPy4 board (Core Electronics, 2020)

4.5.2 Printed Circuit Board

The Water Sensor Expansion Board 1.1 was the PCB used in this project. It has been adopted from the design developed by Professor Jan Diels, who formed part of the “Internet of Drops” project which was a multi-national project that aimed to provide training in the use of IoT technologies for WRM. The project included contributors from Ecuador (Escuela Politecnica Nacional), Ethiopia (Arba Minch University), South Africa (UCT), and Belgium (KU Leuven). The PCB has several different electronic components mounted on the board and these components perform important functions for the overall system. A pictorial diagram of the sensor can be seen in Appendix D.

4.5.2.1 Analog-to-Digital Converter Headers and Terminal Blocks

The PCB was equipped with Headers for the LoPy4 Pycom board and terminal blocks that allowed the connection of multiple sensors. They provided access to all 23 General Purpose Input Output (GPIO) pins and the 5V, Ground (GND), 3v3 and RST pins on the LoPy4. There are four terminal blocks mounted on the PCB. The terminal block for the watermark sensors provided connection for four watermark sensors with two pins available for each sensor. These sensors connected to the LoPy4 via the Analog-to-Digital Converter (ADC) mounted on the PCB. The OneWire terminal block provided connection for two DS18B20 sensors, with each sensor connecting to Voltage Common Collector (VCC), data and GND pins. The Inter-Integrated Circuit (I2C) terminal block supported the connection of two I2C sensors with VCC, Serial Clock Line (SCL), Serial Data Line (SDA) and GND pins while the last terminal block consisted of

unused general pins and ADC pins. The unused pins allowed for integration between the water level sensor and the LoPy4 via the ADC.

4.5.2.2 JST & Voltage Divider

To power the sensor node, the PCB was equipped with a Japan Solderless Terminal (JST) connector for LiPo batteries. The JST connector is connected to a voltage divider which measures the battery voltage. Figure 4-7 depicts the voltage divider circuit adapted from the PCB schematic.

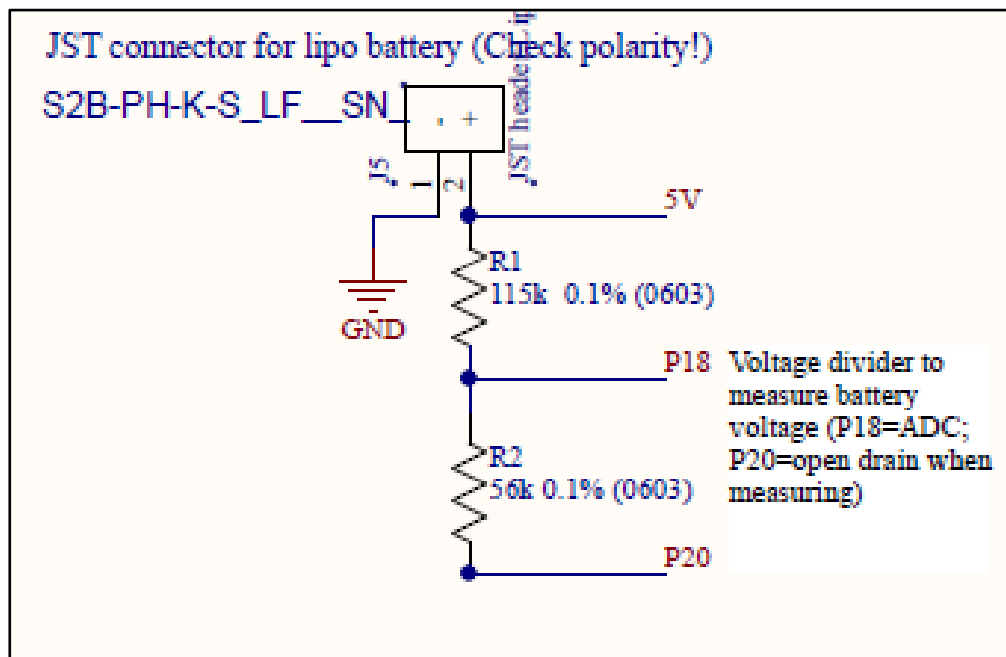


Figure 4-7: Voltage Divider Circuit on the PCB

4.5.2.3 MIC2545A -2YM Power Switch

The MIC2545A -2YM power switch was mounted on the PCB to provide an electrical connection from the voltage source (LiPo battery) or ground to the sensors and multiple other components on the PCB. The power switch protects subsystems from damage and saves power across multiple voltage rails (Huang & Roth, 2017). The power switch also provided improved component protection, inrush current protection and reduces PCB size.

4.5.2.4 ADS1015IDGST Analog to Digital Convertor

To convert analog signals sent by the watermark sensor and the water level sensor into digital signals that the LoPy4 could read, the PCB was equipped with an ADS1015 ADC. The LoPy4 has an onboard ADC, but, according to the Pycom Forums, the onboard ADC is not reliable

(Pycom , 2021). The ADS1015 has a 12-Bit noise-free resolution and requires a low current consumption of 150 μ A during operation.

4.5.2.5 CD74HC4052M96 4 Channel Analog Multiplexer & Demultiplexer

The CD74HC4052M96 4 Channel Analog Multiplexer & Demultiplexer was equipped to the PCB to allow four watermark sensors to connect to the system. The 12-bit ADS1015 ADC does not have enough outputs to read more than two watermark sensors and so the multiplexer connects one signal from several inputs to a single output. This multiplies the number of outputs available and allows the ADC to read signals from four different watermark sensors. This component was not used in this project as only one watermark sensor was connected to the system.

4.5.3 Sensing Implementation

To generate hydrological data from the environment, sensors needed to be deployed. The data types that the systems captured were water temperature, soil temperature, water level, soil moisture, air temperature and relative air humidity. These data sets were captured through the implementation of four different types of sensors.

4.5.3.1 Temperature Sensing

A DS18B20 submersible temperature sensor is a digital thermo sensor that uses a unique Onewire interface to allow for easy communication between sensor and devices. The sensor seen in Figure 4-8 is compatible with the Pycom LoPy 4 device. It has a power supply range of 3.0V to 5.5V and provides 9 to 12-bit temperature readings. The sensor has an operating temperature range of -55°C to +125°C and has an accuracy range between 10°C to +85°C of $\pm 0.5^\circ\text{C}$. Two 5m long DS18B20 sensors were used to measure water temperature and soil temperature. Table 4-1 shows the colour coded pin layout for the sensor.

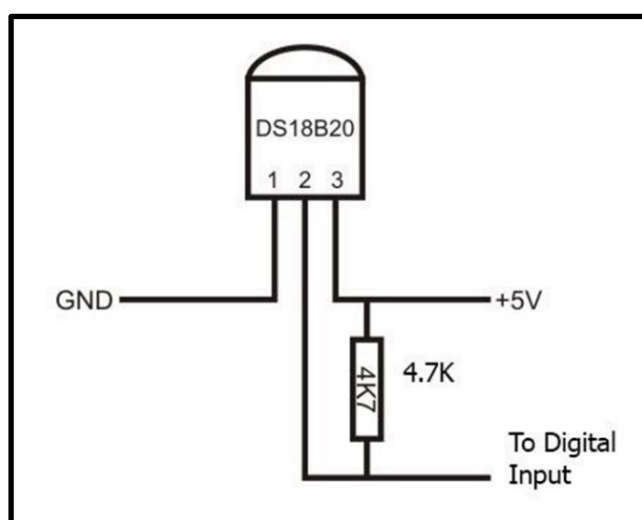


Figure 4-8: DS18B20 Submersible Temperature Sensor.

Table 4-1: Colour coded pin out description for DS18B20

Colour	Name	Description
Red	VCC	Power
Black	GND	Ground
Yellow	data.	Data line

The circuit diagram for the DS18B20 sensor can be seen in Figure 4-9. The GND pin was connected to the GND pin of the PCB while the 1-wire data bus was connected to a digital pin on the board. The VDD pin supplied power to the sensor and was connected to the 3v3 pin on the board.

**Figure 4-9: Circuit Diagram for the DS18B20 Sensor**

The sensors were integrated with the LoPy4 and the PCB using a circuit which adds a 4.7 Kohm pull up resistor between the data line and the power (VCC) line and allowed for stable reading and transmitting from the sensor to the MCU. Figure 4-10 depicts the circuitry built to read data from the DS18B20 sensors. Two DS18B20 sensors were deployed to measure water temperature of the dam and to measure soil temperature that was used in determining soil moisture. The data sheet for the DS18B20 Waterproof Temperature Sensor can be found in Appendix E-1.

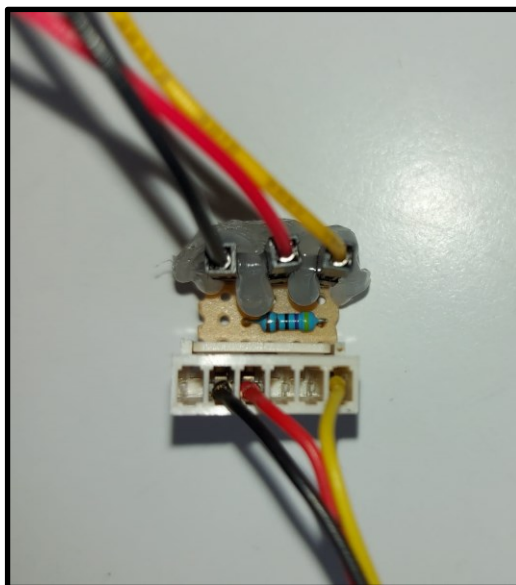


Figure 4-10: Circuit Built to Read Data from the DS18B20 Sensors

4.5.3.2 *Atmospheric Temperature & Humidity Sensing*

Figure 4-11 shows the DF Robot – SHT31 Temperature & Humidity sensor that was used to measure air temperature and relative humidity. The sensor is housed in a weatherproof enclosure and offered high reliability with low power consumption. The sensor is compatible with the Pycom LoPy4 and supports I2C communication. According to the products specifications sheet, the operating voltage of the sensor is in the range of 3.3 - 5V, and it has an operating current of <1.5 mA. The temperature detection range is between -40°C - 125°C with an accuracy of $\pm 0.2^{\circ}\text{C}$. The full specification sheet can be found in Appendix E-2.



Figure 4-11: DF Robot – SHT31 Weatherproof Temperature & Humidity Sensor

Figure 4-12 depicts the pin out of the SHT31 which is easily integrated with the LoPy4 and PCB combination. Table 4-2 provides a colour coded description of the four pin out wires of the SHT31. The VCC, GND, SDA and SCL are connected to the corresponding pin terminals of the PCB with no need for any extra circuitry.

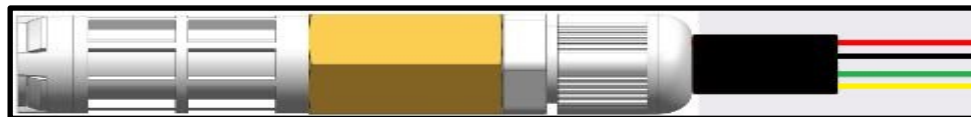


Figure 4-12: Pin out Diagram of the SHT31 Sensor

Table 4-2: Colour coded pin out description for SHT31

Colour	Name	Description
Red	VCC	+
Black	GND	-
Green	SDA	Data line
Yellow	SCL.	Clock line

4.5.3.3 Soil Moisture Sensing

The system took soil moisture readings in kPa using the Watermark Soil Moisture Sensor and the DS18B20 Temperature Sensor. The Watermark Soil Moisture Sensor, in Figure 4-13, is an electrical resistance type sensor that indirectly measures soil water content. The specification sheet for the irrometer watermark sensor can be found in Appendix E-3.



Figure 4-13: The Watermark Soil Moisture Sensor

The sensor measured soil moisture by converting the electrical resistance readings into a calibrated reading of centibars (cb) of soil suction with a range from 0 (wettest) to 200 (driest) centibars (Irrometer, 2020). The sensor was driven by a voltage divider circuit. Basic voltage divider circuits can be seen in Figure 4-14. These circuits are wired differently but they all follow the same electrical principles. A voltage divider circuit always involves applying a voltage across a series of two resistors. The voltage divider equation is derived from Ohms law to give Equation 4.1 below (Blom, 2021):

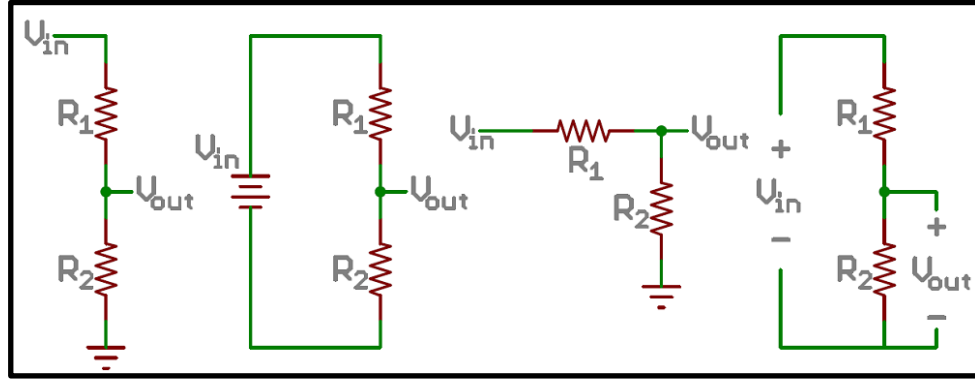


Figure 4-14: Basic voltage divider circuits (Blom, 2021)

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2} \quad (4.1)$$

Where:

V_{out} – Output Voltage

V_{in} – Input Voltage

R_1 – Resistor 1 (Closest to the input voltage)

R_2 – Resistor 2 (Closest to the output voltage)

In Equation 4.2, output voltage is directly proportional to the input voltage and the ratio of resistor 1 and resistor 2. There are generalisations that need to be known when using voltage dividers.

$$\text{if } R_1 = R_2 : V_{out} = V_{in} \times \frac{R}{2R} = \frac{V_{in}}{2} \quad (4.2)$$

If R_2 and R_1 are equal, then the output voltage is half the input voltage, and this remains true regardless of the value of the resistors as seen in Equation 4.3:

$$\text{if } R_2 \gg R_1 : V_{out} \approx V_{in} \times \frac{R_2}{R_2} = V_{in} \quad (4.3)$$

If R_2 is at least an order of magnitude bigger than R_1 , the output voltage will have a similar value to the input voltage. As seen in Equation 4.4 below, the voltage across R_1 will be minimal:

$$\text{if } R_2 \ll R_1 : V_{out} \approx V_{in} \times \frac{0}{R_1} = 0 \quad (4.4)$$

Conversely, if R_2 is much smaller than R_1 , the output voltage will be very small when compared to the input voltage. Most of the input voltage will be across R_1 .

The watermark sensor required a circuit that measures the electrical resistance of the sensor. The sensor's electrical resistance decreased with increasing soil moisture. The circuit allowed for Alternating Current (AC) excitation by alternating the polarity of the sensor. This ensured that there was no net reaction taking place at the electrodes and thus protected the integrity of the sensor's readings and prevented degradation of the gypsum block. Figure 4-15 depicts the circuit diagram for the watermark sensor. The circuit was assembled using two diodes, two precision resistors and an external ADC. These components were embedded on the PCB on which the LoPy4 board is mounted.

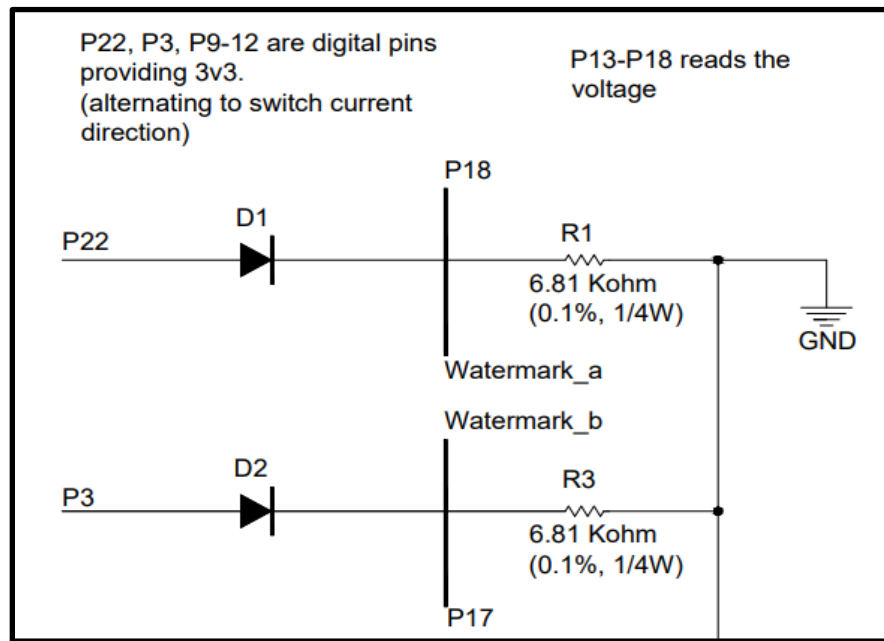


Figure 4-15: Circuit Diagram for Watermark Sensor

The precision resistors were used as the known resistors in the circuit. Voltage measured over the two resistors connected in series were ratioed based on their respective resistances. The watermark sensor was the unknown resistor and by measuring the voltage over the known resistor, the resistance value of the unknown resistor was determined. The voltage was measured using an external ADC due to the onboard LoPy4 ADC not producing reliable readings. The two diodes were used to ensure bidirectional current conduction. This allowed current flow in both

directions through the sensor. The diodes cause voltage loss when conducting current and thus the voltage of the circuit was measured at points directly after the diode to ensure accurate measurements.

Using the known value of the output voltage, the resistance of the sensor was calculated using an ADC and was then converted to a value between 0-200 cb. The temperature of the soil was measured using the DS18B20 temperature sensor. The temperature sensor coupled with the centibar reading was converted to a pressure reading indicative of soil moisture using an equation developed by Shock *et al* (1998). Equation 4.5 developed by Shock *et al* (1998) can be seen below:

$$S = -\left(\frac{4.093+3.213R}{1-0.009733-0.01205T}\right) \quad (4.5)$$

Where:

S – Sensor output (kPa)

R – Resistance of Resistor in Voltage Divider (Ω)

T – Temperature of Soil ($^{\circ}\text{C}$)

4.5.3.4 Water Level Sensing

Figure 4-16 shows the Hafei WNK submersible water level sensor used in this study to monitor water level. The sensor is a throw-in type sensor which receives different pressures at different depths of a liquid.



Figure 4-16: The Hafei WNK Submersible Water Level Sensor

The sensor produces an output voltage of between 0.33-2.97 V.dc and has a range of 0-5m. The resistive sensor has an on-board voltage divider and, similarly to the watermark soil moisture sensor, it converts electrical resistance into a water level depth. The sensor produces an analog voltage output signal which was converted by an ADC and then read by an MCU. According to the manufacturer, the sensor has an accuracy of $\pm 0.25\%$ FS/year, $\pm 0.5\%$ FS/year. The specification sheet for the sensor can be found in Appendix E-4.

Table 4-3 shows the colour coded pinout of the water level sensor. The GND and VCCs pins were connected to available GND and VCC pin terminals on the PCB, while the data pin was connected to the ADC3 pin terminal on the PCB. This allowed the sensor to send the analog output voltage through the ADS1015 where it was converted to a digital signal that was read by the MCU.

Table 4-3: Colour coded pin out description for water level sensor

Colour	Name	Description
Red	VCC	Power
Black	GND	Ground
Yellow	Data	Data/ADC line

The sensor measured water level by calculating pressure in a liquid using Equation 4.6:

$$P = \rho gh + P_0 \quad (4.6)$$

Where:

P – Pressure of liquid on the sensor

ρ - Density of liquid

g – Gravity acceleration

P_0 – Atmospheric pressure at surface of the liquid

h – Depth between the sensor and the surface of the liquid

Equation 4.6 was rearranged to make height the subject of equation 4.7 below:

$$h = \frac{P - P_0}{\rho g} \quad (4.7)$$

The sensor measured the atmospheric pressure and the pressure of the liquid as the liquid level changed, and this was amplified and compensated by the voltage divider circuit. Based off the voltage signal, the known values of density and gravitational acceleration, the system generated a water level measurement.

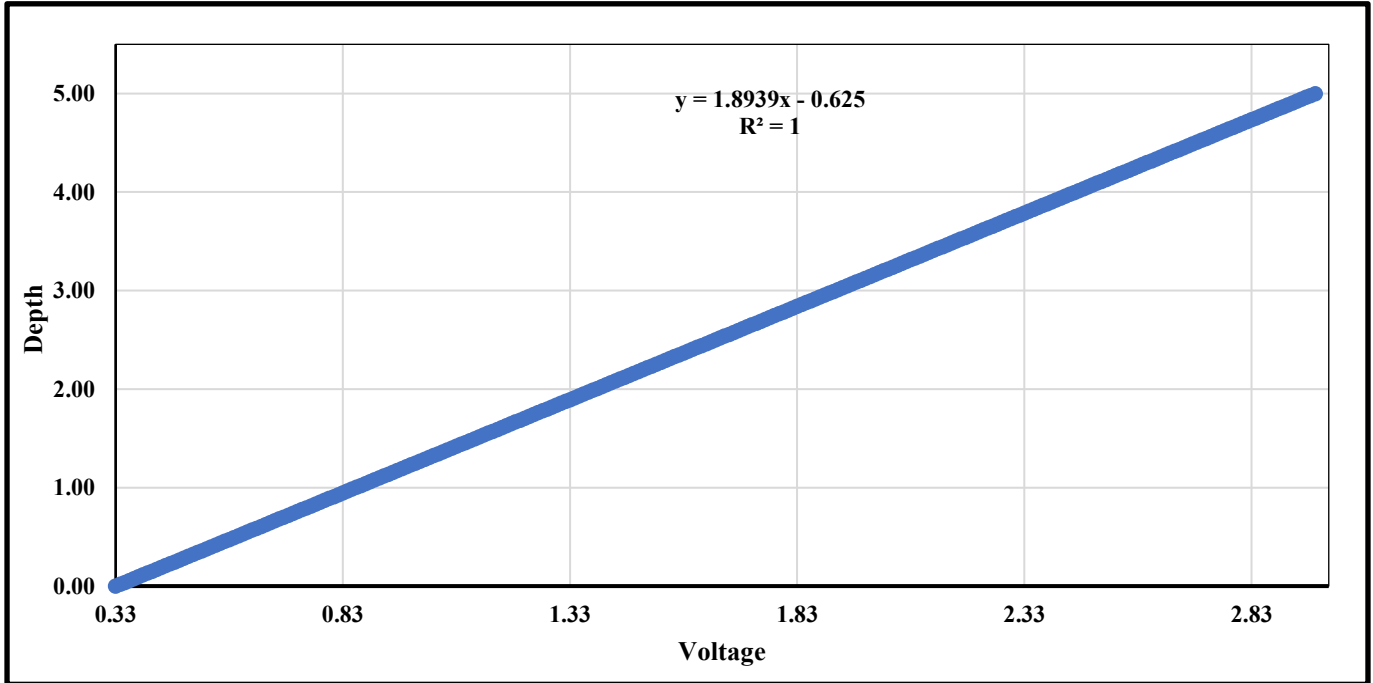


Figure 4-17: Relationship Between Output Voltage of the Sensor and the Depths of a Liquid

To convert the measured voltage to a water level reading, the voltage value was converted to a corresponding depth value according to Table E-1 in Appendix F. The values of Voltage vs Depth are plotted on a line graph in Figure 4-17 and the line of best fit equation was used to determine the water level. The line of best fit was manipulated to generate Equation 4.8 that gives the depth of the liquid being measured:

$$h = (V - V_{h_0}) \times \left(\frac{R}{\rho(V_{h_5} - V_{h_0})} \right) \quad (4.8)$$

Where:

h – Liquid level (m)

V – Voltage output signal (Volts)

V_{h_0} - Voltage output corresponding to a depth = 0m (Volts)

V_{h_5} - Voltage output corresponding to a depth = 5m (Volts)

R – Maximum range of height (m)

ρ – Density of liquid (kg/m³)

4.5.4 Node Setup

The sensor node comprised of four main components, namely the MCU, PCB, sensor device and the power supply. These four components were integrated to form the node. The MCU comprised of the Pycom LoPy4 development board coupled with a LoRa antenna. For this research, two sensor nodes were deployed. Water level and water temperature were measured by the Water Monitoring Node (WMN) and soil moisture, soil temperature, air temperature and relative humidity were measured by the Soil-Air Monitoring Node (SAMN).

The LoPy4 fits onto the PCB, and this allowed for seamless integration with peripherals such as actuators and sensors. Figure 4-18 depicts the LoPy4 mounted on the PCB with the antenna, battery and sensors connected. The antenna was connected to the LoPy4 and allows LoRa transmission to take place over the 868 MHz frequency band.

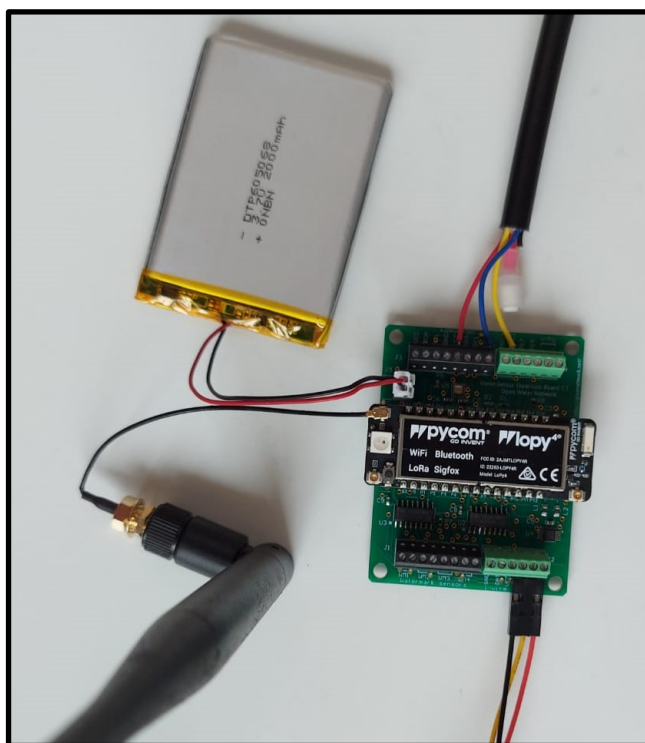


Figure 4-18: LoPy4 Mounted on PCB with Sensors Connected

The power for the system was supplied through a battery. The battery provided power to the system through the JST connector on the PCB. The PCB is equipped with an internal voltage divider which uses a simple series resistor circuit. This ensured the output voltage was a fraction of the input voltage with the divide-down ratio determined by the two resistors used. The sensors were connected to the device through the Water Sensor Expansion Board 1.1 to create a functional circuit that allowed the sensors, coupled with the development board, to perform the required functions.

The WMN and the SAMN had similar configurations, but different types of sensors connected to the nodes. The WMN had a 5m DS18B20 sensor and a Hafei WNK submersible water level sensor while the SAMN had a 5m DS18B20 sensor, an Irrometer watermark sensor and a DF Robot – SHT31 temperature & humidity sensor connected via the PCB. The DS18B20 sensor had the same configuration for both nodes with the sensor connecting to the node via the GND, VCC and data terminal blocks on the PCB.

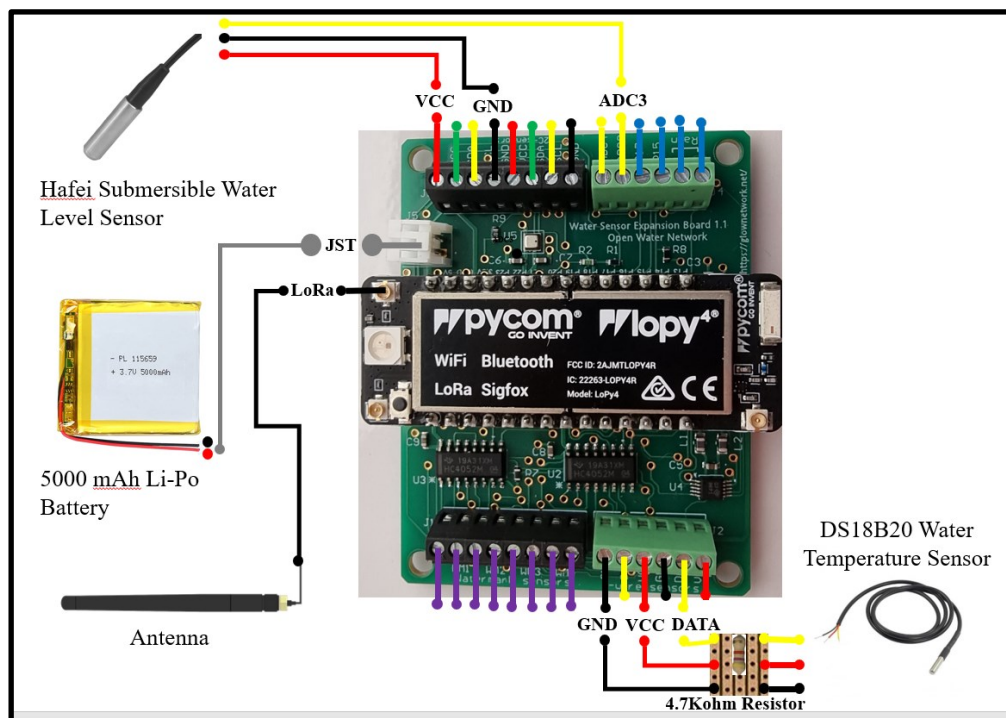


Figure 4-19: Pictorial Representation of WNM Setup

The water level sensor was connected to the node via the ADC3, GND, and VCC terminal blocks and the watermark sensor was connected via the two WM1 terminals. The SHT31 sensor was connected to the PCB via the GND, VCC, SDA and SCL terminals blocks. Figures 4-19 and 4-20 show a pictorial diagram of the electronic setup of WMN and SAMN, respectively.

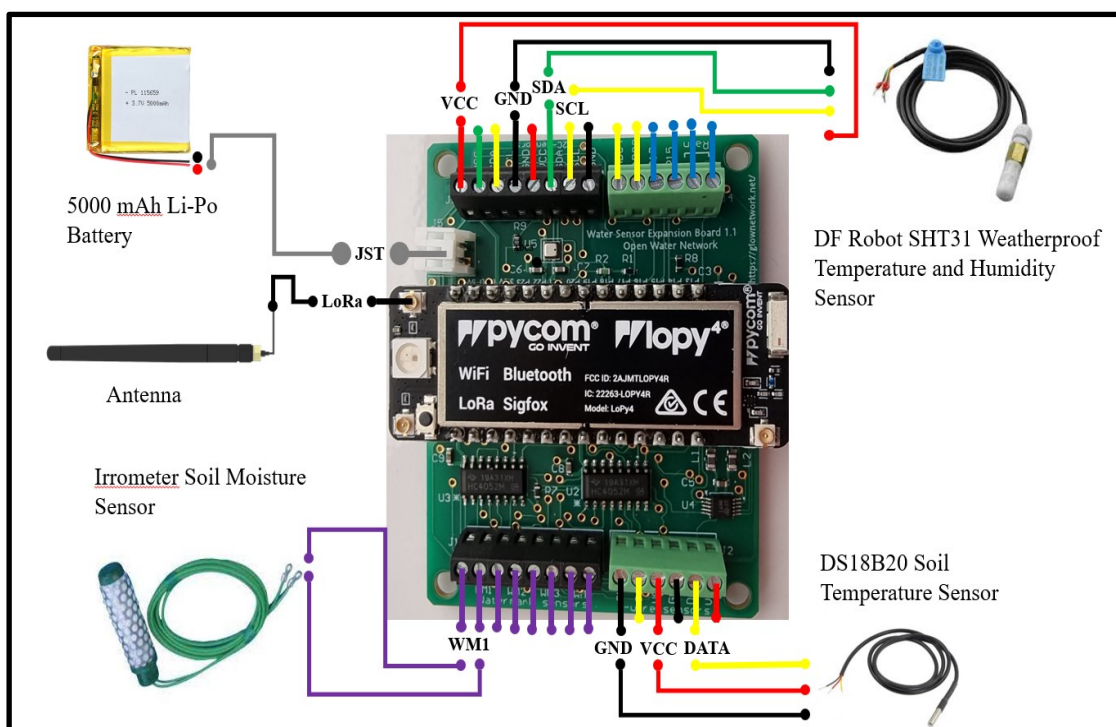


Figure 4-20: Pictorial Representation of SAMN Setup

Both sensor nodes were encased in a waterproof enclosure and connected to a metal L-bracket as seen in Figure 4-21. The sensors ran down a PVC pipe into the water or the ground depending on the node. The WMN's waterproof enclosure was mounted onto a concrete level gauge in the UCT dam, and the SAMN was mounted on a tree on UCT upper campus.



Figure 4-21: Weatherproof Enclosure with L-bracket Housing the Sensor Nodes

The SHT31 temperature and humidity sensor was housed in a DIY solar radiation shield which protected the sensor from direct UV rays that may have affected readings. Figure 4-22 shows an image of the solar radiation shield, and a detailed drawing can be found in Appendix G. The LoRa antenna protruded out the lid of the utility box so that message transmission was possible for both nodes. The waterproof container housing the node protected the node from damage due to rain/dust intrusion into the device. The enclosure provided a dry compartment within which the node could operate without any problems.



Figure 4-22: Radiation Shield Built to House Temperature and Humidity Sensor

4.5.5 Power Considerations

IoT systems that continuously monitor environments in real-time require a system and power supply that is energy efficient. Section 2.3.7 indicates that there are three main battery technologies to consider for IoT-driven wireless sensor networks. Alkaline, lithium and NiMH batteries can support the system, but, based on the system requirements, the options were narrowed down to lithium batteries. Lithium batteries are regularly used as a replacement for alkaline batteries where longer life, higher current draw or improved temperature performance is required. There are many varieties of lithium batteries, and the type of lithium battery was based on the results from testing the current consumption of the system.

To make power considerations for the systems, the systems were measured by a multi-meter. This gave an indication of the power consumption of the system when the system was in active and deep sleep mode. Active mode included the system waking up, taking measurements, and transmitting these messages. Once the system has run through this process, it moved into deep sleep mode where all peripherals were powered down and power consumption was extremely low. Once the overall power consumption of the system was determined, the power supply was considered. The power supply came from battery power and the systems were required to have a life cycle of at least 1 month before it requires any maintenance in the form of battery replacements.

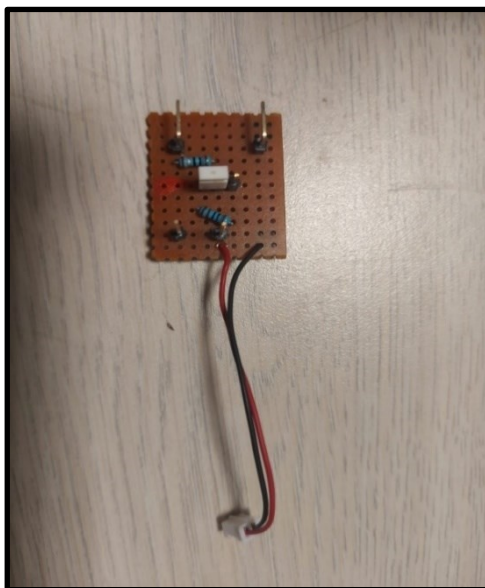


Figure 4-23: Circuit Built to Measure the Current Consumption of the Nodes

Figure 4-23 shows the circuit built to measure the current consumption of the system and the full experimental setup can be seen in Appendix H. The circuit had a 1 Ohm resistor and 1 kOhm resistor in series. While the system is in active mode, the 1 kOhm resistor was bypassed using a switch and the voltage drop across the 1-ohm resistor was measured using a voltmeter. According to Ohms law, the voltage drop across the sensor was quantified by Equation 4.9.

$$1 \text{ mV} = 1\text{mA} \quad (4.9)$$

When the system was in deep sleep mode, the 1 kOhm resistor was reintroduced to the circuit and the voltage drop across the resistors in series was measured. The voltage drop across the sensor was quantified by Equation 4.10 below:

$$1\text{mV} = 1\mu\text{A} \quad (4.10)$$

Once the current consumption was quantified in active and deep sleep mode, the battery life of the system was calculated. Equation 4.11 to calculate the battery life of the system is:

$$\text{battery life in days} = \frac{\text{Battery Capacity}}{\text{Active mAh} \times N \frac{\text{cycle}}{\text{day}} + \text{deep sleep mAh} \times \frac{24\text{h}}{\text{day}}} \quad (4.11)$$

With N the number of measurements per day.

The testing of the system involved taking 10 voltage measurements while the systems were in active and in deep sleep mode. Table 4-4 shows the results from the tests run on the system for the WMN and SAMN.

Table 4-4: Average Voltage Reading of Nodes While in Active and Deep Sleep Modes

No. (#)	WMN			SAMN		
	Active (mA)	Deep Sleep (µA)	Cycles (s)	Active (mA)	Deep Sleep (µA)	Cycles (s)
1	243	21	22	44	21	22
2	249	21	22	48	21	22
3	243	21	22	50	21	22
4	245	21	22	45	21	22
5	239	21	22	40	21	22
6	240	21	22	49	21	22
7	242	21	22	45	21	22
8	241	21	22	46	21	22
9	246	21	22	60	21	22
10	242	21	22	43	21	22
Average	243	21	22	47	21	22

The average voltage drop during active mode for WMN was 243.0 mV and 21.0 mV during deep sleep mode. The current consumption during active mode was 243.0 mA while the current consumption during deep sleep mode was 21.0 µA. To convert the current measured into milliampere-hours, Equation 4.12 was used:

$$\text{mAh} = \text{mA} \times \text{hours per cycle} \quad (4.12)$$

The hours per cycle was measured by timing the seconds taken per active mode cycle and converting this into hours. The average cycle time for the system was 22 seconds per cycle and thus the milliampere-hours per cycle were as follows:

$$243 \times \frac{22}{60 \times 60} = 1.485 \text{ mAh}$$

By re-arranging Equation 4.11, the battery capacity required to power the WMN for a minimum of three months (taken as 92 days for this project) was determined as follows:

$$\text{Battery Capacity} = \text{battery life in days} \times (\text{Active mAh} \times N \frac{\text{cycle}}{\text{day}} \times \text{deep sleep mAh} \times \frac{24\text{h}}{\text{day}})$$

$$\text{Battery Capacity} = 92 \times (1.485 \times 96 + 0.021 \times 24)$$

$$\text{Battery Capacity} = 13\,161.89 \text{ mAH}$$

Based on the power consumption experiments and the requirements of the system, the WMN would require a battery capacity of 13 166.30 mAh to power the system for 3 months. Due to limitations based on the size of the compartment and maximum voltage required to power the MCU, the number of days to power the node was reduced to 34 days which would require a battery capacity of 4865.81 mAh. Based on these power considerations, a 5000 mAh LiPo battery was chosen for the WMN.

The average voltage drop across the SAMN system was 47.0 mV during active mode and 21 mV during deep sleep mode. Using Equation 4.12, and an average cycle time of 22 seconds, milliampere-hours per cycle were as follows:

$$47 \times \frac{22}{60 \times 60} = 0.287 \text{ mAh}$$

Re-arranging Equation 4.11 gave battery capacity to power the node for 3 months (92 days):

$$\text{Battery Capacity} = \text{battery life in days} \times (\text{Active mAh} \times N \frac{\text{cycle}}{\text{day}} \times \text{deep sleep mAh} \times \frac{24\text{h}}{\text{day}})$$

$$\text{Battery Capacity} = 92 \times (0.287 \times 96 + 0.021 \times 24)$$

$$\text{Battery Capacity} = 2581.15 \text{ mAH}$$

According to the power consideration experiment, the SAMN would require a 2581.15 mAh battery capacity to have a theoretical node-life of 3 months. Based on these results and with the aim of extending the node-life past 3 months, a 5000 mAh LiPo battery was chosen to power the SAMN. The onboard ADC of the LoPy4, which will be detailed in Section 4.6.1, measured the battery voltage of the system throughout the deployment of the two nodes. This monitored the battery life of the node in real-time and gave the user an indication of when maintenance was required.

4.6 Software Application

The software of the system was responsible for preparing the acquired data and sending it through the network of the system (Pretorius, 2019). There are existing libraries and scripts that have been written for similar projects. These libraries and scripts were analysed and, if applicable, were manipulated and used in this project. The programming language used in this study was Micro Python, which is a condensed form of Python that can be used to programme MCUs that are compatible with Micro Python, like the Pycom range. The programming of the system was split into various sections which were responsible for the functionality of the different architectural layers in the system. This section discusses the software development required for the layers in this study.

4.6.1 Firmware Design for Pycom

The Pycom LoPy 4 device was the central component of the IoT system and allowed for the easy integration of all peripherals to measure, process and transmit data to the LoRa network. The LoPy4 was embedded with Pycom modules that are specific to Pycom devices and these include modules that support access to underlying hardware, such as I2C, Serial Peripheral Interface (SPI), WLAN, Bluetooth etc (Pycom, 2021). The embedded modules that were used in this project include the machine library and the network library. The embedded function used from the network library was the LoRa function while the embedded functions used from the machine library include ADC, I2C, PIN and RTC.

The nodes also made use of the `machine.deepsleep` function which takes time as a keyword argument. This function puts the LoPy4 into deep sleep once readings have been processed and sent to the listening gateway. Deep sleep reduces overall power consumption of the system by reducing the current draw when the node is inactive.

4.6.1.1 Onboard ADC

The onboard ADC of the Pycom LoPy4 was used to monitor the battery voltage of the nodes. The function creates an ADC object using the pins on the LoPy4 board and the resolution of the bits can be set to an integer value between 9 and 12 (Pycom, 2021). An ADC object was created on P18, and the bits resolution was set to 12. At the start of active mode of the nodes, P18 takes 100 samples of the battery voltage. An average of the 100 values was assigned to the variable battery voltage. The number of samples was to ensure accuracy when reading battery voltage with the onboard ADC. The Micro Python script for reading battery voltage can be found in Appendix I.

4.6.1.2 I2C

I2C protocol provides communication between devices using two wires. At a physical level, SDA represents the data line while SCL represents the clock line (Pycom , 2021). Creating an I2C object involves attaching the object to a specific bus and initialising the object when created or initialising it later. Initialising an I2C bus is done with the parameters seen in Table 4-5.

Table 4-4: Description of I2C parameters (Pycom , 2021)

Parameter	Description
Mode	must be I2C.MASTER
Baudrate	SCL clock rate
Pins	optional tuple with the pins to assign to the I2C bus. The default I2C pins are P9 (SDA) and P10 (SCL)

4.6.1.3 Pin Selection

The pin is an object used to control the GPIO of the MCU. Pins are used to drive an output voltage and read input voltages amongst other functions. These physical pins are connected to the MCU, and the mode of these pins can be set using different methods. Table 4-6 gives a description of the different methods available for Pycom boards (Pycom , 2021).

Table 4-5: Description of Pin methods (Pycom, 2021)

Parameter	Description
pin.init(mode, pull, *, alt)	Initialises the pin using four arguments and returns None
pin.id()	Retrieves pin id
pin.value([value])	Only works in Pin.OUT mode – sets or gets digital logic level of pin
pin([value])	Provides a shortcut to set or get a pins value
pin.toggle()	Toggle value of a pin
pin.mode([mode])	Set or get mode of a pin
pin.pull([pull])	Set or get pin pull
pin.hold([hold])	Set or get a pin hold - Commonly used to hold pins state after deepsleep

The PCB used in this research was connected to all 28 pins of the LoPy4 via 28 terminal block connectors. Communication with these pins can happen through multiple protocols. The protocols used in this research included ADC, OneWire and I2C.

Analog control of these pins is done through the ADC software module or an external ADC that allows the pin to convert the analog signals into digital signals that can be read by the electronic components used in the system. This conversion creates an interface between the physical world and the electronic system. OneWire communication, as is in the name, initiates communication using a single data line and a ground reference (Linke, 2008). Referred to as the OneWire bus, the One-Wire master controls multiple slave devices on the bus. All 23 general pins on the LoPy4 device can be used for OneWire protocol control. I2C provides communication between devices by using a two-wire protocol (Pycom, 2021). The Pycom LoPy4 device has assigned pins P9 and P10 which provide I2C control for integration of devices that communicate with this protocol.

The LoPy4 coupled with the PCB has 28 GPIO pins which, through software modules, can be controlled to serve the specific purpose required. Table 4-7 gives a description of the default modes of some of the pins on the LoPy4 device.

Table 4-6: Description of default pin modes (Pycom, 2021)

Module Pin	Pin Name	Default Function	Description
2	P0	RX0 (Programming)	Used to program the module by the bootloader
3	P1	TX0 (Programming)	Used to program the module by the bootloader
5	P3	TX1	-
6	P4	RX1	JTAG TDO, SD card CMD
11	P9	SDA	Data line (I2C)
12	P10	SCL (I2C) / CLK (SPI)	Clock line (I2C)
13	P11	MOSI	Master Output Slave Input (SPI)
16	P14	MISO	Master Input Slave Output (SPI)
26	3v3	Regulated 3v3 supply	Output only
27	GND	Ground	-
28	Vin	Voltage input	Voltage input between 3.5V and 5.5V

The full pinout diagram with the default modes of the pins on the LoPy4 board can be seen in Appendix C-2. The system, where possible, was setup to utilise pins based on their default mode. However, because of the Pycom board's flexibility, pins are not limited to one mode, allowing pins to be configured based on the user's requirements.

4.6.1.4 LoRa Setup

For the system to communicate using the LoRa network, the Pycom LoPy 4 needs to operate in LoRaWAN mode. This mode allows the device to run a full LoRaWAN stack for a class A device. The mode supports both the Over The Air Authentication (OTAA) and Authentication By Personalisation (ABP) connection methods. In this study, the OTAA connection method was used to connect to the LoRa network. The system runs on the things stack platform developed by TTN. The device operated within the 863-870 MHz (SF9 for RX2) frequency plan while the LoRaWAN version and the regional parameters of the device were set by the user. The connection procedure involved the device performing a join procedure with the network, during which a dynamic device address was assigned and security keys were negotiated with the device. The default script for setting up OTAA connection can be found at <https://docs.pycom.io/tutorials/networks/lora/lorawan-otaa/>.

4.6.2 Micro Python Libraries

Multiple Micro Python Libraries were used in this research to program the systems. Most of the libraries exist and only need to be adapted to meet the output requirements of the specific study. The systems make use of seven micro python libraires to function i.e., `adcR.py`, `ads1x15.py`, `adsx15read.py`, `BME280_float.py`, `onewire.py`, `watermark.py`, `measureSensors.py`, and `sht31.py`. All the MicroPython libraries below can be found in the following GitHub repository: [RobYNWA/Hydrological-Monitoring \(github.com\)](https://github.com/RobYNWA/Hydrological-Monitoring)

4.6.2.1 *adcR.py*

The `adcR.py` library was developed for the ‘internet of drops’ project (Diels, 2019). The library initialises the onboard LoPy4 ADC by calling the `adcRead` function. The function takes ADC readings and converts the signals to voltage readings. Keyword arguments for the library are presented in Table 4-8. The function returns a mean ADC value of the 100 samples which represents a voltage that is approximately equal to the voltage of the power supply to the system.

Table 4-7: Description of keyword arguments for `adcR.py` library

Keyword Argument	Description
Pain	pin object as created with <code>adc.channel()</code> call
Bits	integer value between 9 and 12 = the resolution set for the ADC with <code>machine.ADC()</code> call
nSamples	(optional) number of times the ADC reading is repeated to calculate an average (default=100)

4.6.2.2 *ads1x15.py*

The *ads1x15.py* library was developed by Dopieralski and Hammelrath (2017) with further modification on the ‘internet of drops’ project. The library initialises the external ADS1015 mounted on the PCB. The library contains multiple classes with multiple functions that allow the user to initialise the ADS1015 by setting keyword arguments as seen in Table 4-9.

Table 4-8: Description of keyword arguments for ads1x15.py library

Keyword Argument	Description
i2c	initialised by the machine library and defaults to Pin 9 = SDA and Pin 10 = SCL with a maximum baud rate of 400kHz for pycom devices.
Addr	set as 72
gain	the programmable gain is set to 1 for a 2-mV resolution and +/- 4.096V range.

Once the ADS1015 has been initialised it can convert analog sensor signals to digital values that can be read by the LoPy4.

4.6.2.3 *adsx15read.py*

The *adsx15read.py* library is used to create a class called conversion that allows the program to reference a variable in the global space directly from an interrupt call back (Diels, 2019). The conversion class creates an instance, and an integer is used to send the ADC reading from the interrupt call back to the main programme. The sample auto function drives the alert process by taking the keyword arguments seen in Table 4-10.

Table 4-9: Description of keyword arguments for adsx15read.py library

Keyword Argument	Description
x	variable integer which is used to send the ADC readings from an interrupt call back
adc	set at <code>ads.alert_read</code> which prompts the function to be called when the alert pin has fallen at the end of the conversion. This will prompt the ADC to take voltage measurement readings.

4.6.2.4 *BME280_float.py*

The *BME280_float.py* library drives the function of the BME280 I2C/SPI temperature & humidity sensor. Cunnane *et al* (2022) developed this Micro Python library by borrowing from the Adafruit BME280 Python library which has been written for Raspberry Pi modules but

modified for Micro Python and esp8266 boards. Through OOP the library allows the BME280 sensor mounted on the PCB to measure temperature and humidity through I2C/SPI protocol.

4.6.2.5 *onewire.py*

The *onewire.py* library is the driver for the DS18B20 water temperature sensor. Similarly to the BME280 library, the library programmes the sensor to measure temperature through OOP using onewire protocol.

4.6.2.6 *sht31.py*

The *sht31.py* library is the driver for the I2C DF Robot – SHT31 temperature & humidity sensor. The library uses OOP to program the sensor to measure air temperature and relative air humidity using I2C protocol.

4.6.2.7 *watermark.py*

The *watermark.py* library is the driver for the irrometer watermark soil moisture sensor. Written by Diels (2019), the library imports the *adsx15read.py* library to create a watermark class. The `__init__` function initialises the objects state and is commonly used in OOP. Table 4-11 shows descriptions of the keyword arguments taken by the function.

Table 4-10: Description of keyword arguments for `__init__` function

Keyword Argument	Description
<code>self</code>	represents the instance of the class and allows the function to access the attributes and methods of the class.
<code>S0pin</code>	S0 pin used for selecting a channel on the CD74HC52.
<code>S1pin</code>	S1 pin used for selecting a channel on the CD74HC52.
<code>enableA2Bpin</code>	prompts the current to move from pin A to pin B of the watermark sensor.
<code>enableB2Apin</code>	prompts the current to move from pin B to pin A of the watermark sensor
<code>powerWmpin</code>	pin for powering the voltage divider that is used to measure the watermark sensor.
<code>r1ohms</code>	value of the resistor used in the voltage divider.

The class has 6 functions with four of the functions called `readWM#` and these functions are identical and power the four different watermark sensors that can be connected to the node. These functions take the following keyword arguments shown in Table 4-12.

Table 4-11: Description of keyword arguments for readWM# function

Keyword Argument	Description
self	represents the instance of the class and allows the function to access the attributes and methods of the class.
ads	instance of an ADS1015 object
irq_pin	interrupt pin which prompts the ads to measure voltage
n	number of samples measured by the watermark sensor (defaults at 1)

The read function reads the watermark sensor by powering the voltage divider. Table 4-13 gives a description of the keyword arguments taken by the function.

Table 4-12: Description of keyword arguments for read function

Keyword Argument	Description
self	represents the instance of the class and allows the function to access the attributes and methods of the class.
ads	instance of an ADS1015 object
irq_pin	interrupt pin which prompts the ads to measure voltage
n	number of samples measured by the watermark sensor (defaults at 1)

The read function lets current move from A to B of the watermark sensor and reads the voltage, and then moves the current from B to A and reads the voltage again. The final function in the `watermark.py` library is the ShockkPa function and is based off the equation developed by Shock *et al.*, (1998). This function calibrates the moisture reading and converts the values to water potential in kPa. Table 4-14 describes the keyword arguments taken by the function.

Table 4-13: Description of keyword arguments for ShockkPa function

Keyword Argument	Description
Rkohm	value of the resistor
tempDegC	temperature in degrees Celsius (defaults at 20 °C)

Once this programme runs through the library, it prompts the system to read the resistance of the watermark sensor.

4.6.2.8 *measureSensors.py*

The `measureSensors.py` library, developed for the ‘internet of drops’ project, imports all the above libraries mentioned above and acts as the prompt for the sensors to take measurements and return those measurements to the MCU. The library also plays an important role in improving the energy efficiency of the system. The entire script runs under the `measure` function which takes a `messageNumber` as its argument. `messageNumber` is programmed in the main script and assigns each measurement package a message number starting at 1 and increases in increments of 1. The message number saves to the non-volatile storage of the LoPy4. The `measure` function imports all libraries mentioned above and then uses the built-in `Pin` library to control the functions of the GPIO on the LoPy4. Pin 20 is set to open drain which grounds and powers the sensor once the system is in active mode. The script then prompts the LoPy4 device to measure the battery voltage using the onboard ADC. Once the battery voltage has been measured, the script initialises the ADS1015 and the onboard I2C to create the variables `i2c` and `ads` which will function as instances of the I2C and ADS1015 objects respectively.

Within the `measure` function there are additional functions that programme the relevant sensors to measure the required data. When all sensors have taken measurements, the script converts the measured data to Comma Separated Value (CSV) format using the `f.write` function and this data is saved to a MicroSD memory card mounted on the PCB. Once the data has been successfully written to the MicroSD memory card, all active pins on the LoPy4 are set to input mode as seen in Figure 4-24.

```
# Limit power consumption during deep sleep by pin settings
# I2C and data pin DS18B20 to input mode to minimize power use during
deepsleep
# this is also their default at startup
p=Pin("P21",mode=Pin.IN,pull=Pin.PULL_DOWN,alt=-1) # data pin DS18B20
p9=Pin("P9",mode=Pin.IN,pull=Pin.PULL_DOWN,alt=-1) # SDA pin for I2C
p10=Pin("P10",mode=Pin.IN,pull=Pin.PULL_DOWN,alt=-1) # SCL pin for I2C

pinSD_SLCK=Pin("P23",mode=Pin.IN,pull=Pin.PULL_DOWN,alt=-1) # put pull-down
resistor on P23
# this is the SLCK pin for the microSD card. Default definition is no
pullup or pulldown
# That pin turned out to increase power consumption from 22µA to 80µA
during deepsleep
# when P23 is connected to microSD.
pinSD_DAT0=Pin('P8', mode=Pin.IN, pull=Pin.PULL_DOWN) # also needed to
limit power consumption
pinSD_CMD =Pin('P4', mode=Pin.IN, pull=Pin.PULL_DOWN) # during deep sleep
p20.value(1) # set pin floating to disconnect sensors from ground during
deep sleep
p20.hold(1) # hold pin in same status during deep sleep and wakeup
```

Figure 4-24: Micro Python Script to Limit Power Consumption of LoPy4 and PCB

This limits the power consumption of the sensors and other peripherals during deep sleep and increases the energy efficiency of the system. Pin 20's value is set as 1 which disconnects the sensors from ground during deep sleep and the hold function sets Pin 20 as 1 to hold the pin in the same state throughout the deep sleep mode. Lastly, the script returns all the measurements taken by the sensors and these values are processed by the `main.py` script.

4.6.3 Sensor Setup

4.6.3.1 DS18B20 Submersible Temperature Sensor

The DS18B20 sensors connected with the device using a one wire bus. This is a serial bus that uses one wire for data communication. The sensor is driven by two separate open-source modules found in the one wire library, namely, `OneWire` and `DS18X20`. These modules are an example of OOP and aid in simplifying a program while also making the code reusable. The portion of the script that reads the DS18B20 in the `measureSensors.py` library can be found in Figure 4-25.

```
ow = OneWire(Pin('P21'))

# Define function to measure the temperature with the DS18X20 sensor
# on a one-wire bus (owBUS). If several sensors are connected to this bus
# you need to specify the ROM address of the sensor as well
# If only one DS18x20 device is attached to the bus you may omit the rom
parameter.
def measureTemperature(owBus,rom=None):
    while True: # we loop until we get a valid measurement
        temp = DS18X20(owBus)
        temp.start_conversion(rom)
        time.sleep(1) # wait for one second
        TempCelsius=temp.read_temp_async(rom)
        if TempCelsius is not None:
            return TempCelsius # TempCelsius exit loop and return result
# single temperature measurement omitting the ROM parameter
while True:
    soilTempCelsius=measureTemperature(ow)
    if not soilTempCelsius==85:
        break # jump out of loop
    print('soilTempCelsius=85°C which means it is equal to the power-on
reset value of the temperature register. Repeat the measurement!')

print("Soil temperature (degrees C) = %7.1f" % soilTempCelsius)
```

Figure 4-25: Micro Python Script that Reads the DS18B20 Sensor

Initially, the OneWire bus was assigned to Pin 21 through the variable `ow`. The script then prompted the MCU to scan for all DS18B20 sensors connected to the OneWire bus by searching for their unique 64-bit (=8 bytes) address in the sensors ROM memory. The `measureTemperature` function was then called which takes two arguments, `owBus` and `rom`. The function initiates the sensors to take temperature measurements and assigns the measurements of the sensor to variables `waterTempCelsius` and `soilTempCelsius` dependent on the node. The modules are available at <https://github.com/RobYNWA/Hydrological-Monitoring-Water>.(Github repository)

4.6.3.2 DF Robot – SHT31 temperature & humidity Sensor

The SHT31 sensor connects with the device using I2C protocol. The sensor is driven by the `sht31.py` library. The library was imported to the `measureSensors.py` module and the portion of the script that reads the SHT31 sensor can be found in Figure 4-26.

```
i2c=I2C(baudrate=400000)
sensor = sht31.SHT31(i2c = i2c, addr=0x44)
(airTempCelsius,airRelHumidity)=sensor.get_temp_humi()
print(airTempCelsius)
print(airRelHumidity)
return(airTempCelsius,airRelHumidity)
```

Figure 4-26: Micro Python Script that Reads the SHT31 Sensor

The portion of the script begins with initialising the I2C of the MCU. The variable `sensor` was created and assigned a unique I2C address. The variables `airTempCelsius` and `airRelHumidity` are assigned the values measured by the sensor using the `get_temp_humi()` function found in the `sht31.py` library. Values for temperature and humidity measured by the sensor were then returned and assigned to variables `airTempCelsius` and `airRelHumidity` respectively. The modules are available at: [RobYNWA/Hydrological-Monitoring---Soil-Air: Monitoring node \(github.com\)](https://github.com/RobYNWA/Hydrological-Monitoring---Soil-Air-Monitoring-node)

4.6.3.3 Hafei WNK submersible water level sensor

The Hafei WNK submersible water level sensor operates by converting analog current into digital signals which the MCU can process. The conversion was done by an ADC that interfaces the sensor and the MCU.

```

# Measurement of Water Level Sensor
# *****#
def measureWaterLevel():
    adsDataRate=6 # for ADS reading
    RANGE = 5000 # Depth measuring range 5000mm (for water)
    VOLTAGE_INIT=0.33 # Voltage @ 0mm (uint: Vdc)
    DENSITY_WATER=1 # Pure water density normalized to
    (ADCreading,voltage)=adsx15read(ads,irq_pin,adsDataRate,adsChannel=2)
    while True:
        depth = (voltage - VOLTAGE_INIT) * (RANGE/ DENSITY_WATER / 2.64)
#Calculate depth from voltage readings
        if depth is not None:
            return depth # TempCelsius exit loop and return result
    WaterLevel=measureWaterLevel()

print("Water level= %.3f" % WaterLevel)

```

Figure 4-27: Micro Python Script that Reads the Water Level Sensor

The sensor used to measure water level has no libraries that are compatible with Pycom devices. Therefore, the code had to be developed using the help of members on the Pycom forums. The code obtained from the forums was altered and modified to produce a script that works within the required context. The portion of the `measureSensors.py` library that reads the water level sensor can be found in Figure 4-27. The `measureWaterLevel` function takes no set argument and begins by defining four fixed variables namely, `adsDataRate` set as 6, `RANGE` set at 5000 (maximum sensor range), `VOLTAGE_INIT` set at 0.33 which is the voltage output at 0 mm depth, and `DENSITY_WATER` set at 1. The variable `voltage` is assigned an ADC reading from the ADC3 pin of the ADS1015 using the `adsx15read.py` and `ads1x15.py` libraries. A while loop takes the output voltages of the sensor and converts them into a depth based on Equation 4.8 in section 4.5.3.4. The modified code can be found at <https://github.com/RobYNWA/Hydrological-Monitoring-Water>. (Github repository)

4.6.3.4 Watermark Soil Moisture Sensor

The watermark sensor requires modules imported from existing libraries to function. The script to read the watermark sensors used the `watermark.py`, `adsx15read.py` and the `ads1x15.py` libraries. The three libraries enabled the watermark sensor to measure the resistance of the sensor and convert the analog signal to a digital signal that can be read by the MCU using the ADS1015. The portion of the script that reads the watermark sensor can be found in Figure 4-28.

```

# Read Watermark sensors
# *****
# Set up watermark sensors (each sensor requires two analog and two digital pins)
#
#*****
*****
wm=watermark.watermark(S0pin=Pin('P11'),S1pin=Pin('P12'),enableA2Bpin=Pin('P3')
,enableB2Apin=Pin('P19'),powerWmpin=Pin('P22'),r1ohms=7870)

# The rON resistance of the mux is about 8ohm when the watermark resistance varies
between 4k and 94k (so in-out current is low).
# This 80-ohm resistance is twice in series with the watermark resistance. So
we need to subtract 160 ohm.

wm1kohm=(wm.readWM1(ads,irq_pin,n=1)-160)/1000 #subtract 160 ohm
print("WM1: rWatermark (kohm)= %7.3f" % wm1kohm)
kPa1=watermark.ShockkPa(wm1kohm,soilTempCelsius) # calibration Shock et al.
(1989)

```

Figure 4-28: Micro Python Script that Reads the Watermark Sensor

The watermark sensor requires four pins, two analog and two digitals. For this, Pin 11, Pin 12, Pin 3 and Pin 19 were selected. Pin 22 was set as the pin that powers the watermark sensor when it is time to take a measurement. These pins were selected based on the integration of the PCB with the GPIO of the LoPy4. The ohms of the resistor used in the voltage divider circuit that allows the watermark sensor to read resistance was assigned to the variable `r1ohms` and this value was set as 7870. The variable `wm1kohm` was created and assigned the `readWM1` function which prompts the watermark sensor to read the resistance. The value read was subtracted by 160 and divided by 1000 to get the value into the right units. Once this process had finished, using the `ShockkPa` function, the value assigned to `wm1kohm` was converted to a pressure and assigned to the variable `kPa1`. This was done by using the value assigned to `soilTempCelsius` and based on Equation 4.5. The modified code can be found at <https://github.com/RobYNWA/Hydrological-Monitoring---Soil-Air>. (Github repository)

4.6.4 The Things Network (TTN)

To use TTN to capture sensor data, the sensor system needs to be connected and integrated to TTN. The connection and integration process are two-fold and involves creating an application on TTN and then programming the MCU to integrate and connect with this application. Creating an application on TTN involved visiting the TTN website, creating an account and then visiting the Things Stack Community Edition section of the website. The Things Stack Community Edition allows the user to create and name an application. Once the application has been created, end devices can be named and added to the application. To create an end device, a frequency

plan, LoRaWAN version, Regional Parameters version, Activation mode, and LoRaWAN Class were chosen and the DevEUI, AppEUI, and AppKey were generated. Table 4-15 has a breakdown of the parameters selected for the end device created.

Table 4-14: TTN parameters selected for the Nodes

TTN Parameters	
Frequency Plan	Europe 863-870 MHz (SF9 for RX2)
LoRaWAN Version	LoRaWAN Specifications 1.0.2
Regional Parameters	RP001 Regional Parameters 1.0.2
Activation Mode	Over the air activation (OTAA)
LoRaWAN Class	Class A

Once generated, the DevEUI, AppEUI and AppKey were used to connect the end device on TTN to the sensor node. The AppEUI and AppKey were used to send a join request to a gateway from the end device and this established a connection between the sensor node and the network. Once this connection was established, the sensor node can send data to the TTN. Figure 4-29 depicts the code found in the main.py script that prompts the sensor node to establish the connection with TTN. The frequency plan was assigned to the variable `lora` and the LoRa state, consisting of the join status, network keys, packet counters etc., was restored from the devices non-volatile memory. The OTAA authentication parameters were used to begin the join process which uses a while loop to loop over the join process until the device successfully joins the LoRa network.

```
# Initialise LoRa in LORAWAN mode.
# Please pick the region that matches where you are using the device:
# Asia = LoRa.AS923
# Australia = LoRa.AU915
# Europe = LoRa.EU868
# United States = LoRa.US915
lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)

# Restore the LoRaWAN state (joined status, network keys, packet counters, etc)
# from non-volatile memory.
lora.nvram_restore()

# create an OTAA authentication parameters, change them to the provided
# credentials
app_eui = ubinascii.unhexlify('#####')
app_key = ubinascii.unhexlify('#####')
dev_eui = ubinascii.unhexlify('#####')

print('lora.has_joined()=', lora.has_joined())
```

```

if not lora.has_joined():
    # join a network using OTAA (Over the Air Activation)
    lora.join(activation=LoRa.OTAA, auth=(dev_eui, app_eui, app_key),
timeout=0)
    # wait until the module has joined the network
    while not lora.has_joined():
        time.sleep(2.5)
        print('Not yet joined...')

    print('Joined')

# create a LoRa socket
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)

# set the LoRaWAN data rate
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 3)

# make the socket non-blocking
# (because if there's no data received it will block forever...)
s.setblocking(False)

```

Figure 4-29: Micro Python Script that Connects the Device to TTN.

Once successfully joined, a LoRa socket was created which allows the user to set the LoRaWAN data rate and the LoRa socket was set to non-blocking to avoid the device being blocked. Once all measurements were taken, the data obtained was packaged into a payload using the lines of script seen in Figure 4-30.

```

# next encode in bytes
# '>'=big endian;
# 'b'=signed integer 1 byte = int:8; 'B'= unsigned integer 1 byte = uint:8;
# 'h'=short signed integer 2 bytes = int:16 'H'=short unsigned integer 2 bytes = uint:16
# 'i'=long signed integer 4 bytes = int:32 'I'=long unsigned integer 4 bytes = int:32
# 'f'=float (single precision real number) 4 bytes
# 'd'=double (double precision real number) 8 bytes
payload=struct.pack(">HhHHhHHI",messageNumber,batmV,hPa1,soilTempCentigradeCelsius,airTempCentigradeCelsius,airRelHumiditytypermil,TempCentigrade,unixtimesec
s)

def sendpayload(payload):
    print('Sending:', payload)
    s.send(payload)

```

```

time.sleep(4)
rx, port = s.recvfrom(256)
if rx:
    print('Received: {}, on port: {}'.format(rx, port))
    stats=lora.stats()
    print(stats)
    # counterfrom,counterto=struct.unpack(">hh",rx) # example on how to
decode
    # Signal successful receipt of downlink with green led for 10 seconds
    pycom.heartbeat(False) # stop the heartbeat
    pycom.rgbled(0x007f00) # green
    time.sleep(10)
    pycom.rgbled(0) # switch led off
    pycom.heartbeat(True) # start the heartbeat
    time.sleep(6)

# Now send the payload
sendpayload(payload)

# Save the LoRaWAN state (joined status, network keys, packet counters, etc)
# in non-volatile memory in order to be able to restore the state when coming
# out of deepsleep or a power cycle
lora.nvram_save()

```

Figure 4-30: Micro Python Script that Packages the Data and Sends it to TTN

The payload message structure consisted of an 18-byte hexadecimal payload. The payload contained the measured data captured by the sensor and was converted from an integer value to a hexadecimal format that can be accepted by the TTN network. Payloads sent from the two monitoring systems contained the following data:

- Message number,
- Battery voltage,
- Water level,
- Water temperature,
- Soil moisture,
- Soil temperature,
- Atmospheric Temperature,
- Atmospheric Humidity,
- Temperature in the electronic compartment,
- Relative humidity in the electronic compartment,
- Time to transmit message successfully.

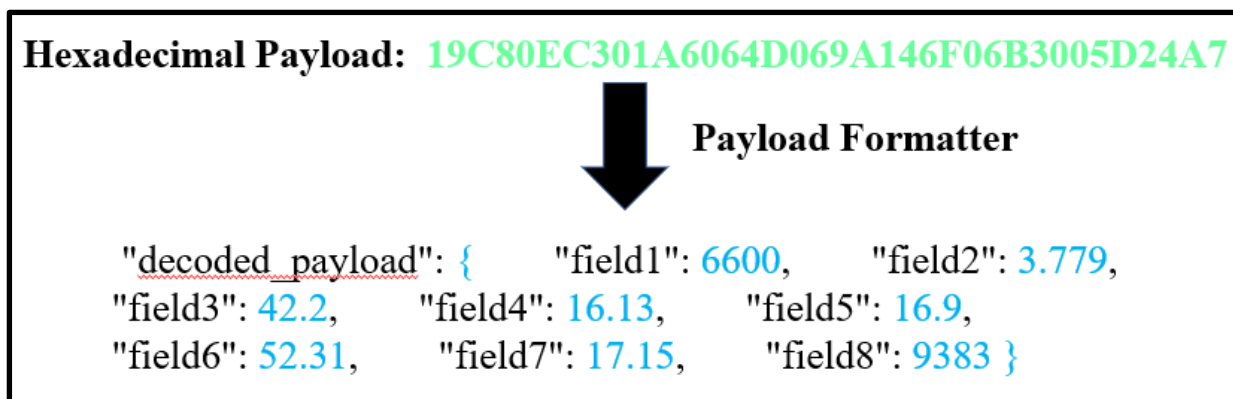


Figure 4-31: Hexadecimal Payload Converted using a Payload Formatter

Once the payload was sent to TTN, the hexadecimal payload was decoded using a payload formatter. Figure 4-31 depicts an 18-byte payload that has been decoded using the payload formatter. The payload formatter allowed for the conversion of binary payloads to human readable data in the form of fields. The payload formatter script was written in JavaScript and for this project there were two variations, one for the WMN and one for the SAMN. The script for the WMN can be seen in Figure 4-32.

```

function decodeUplink(input) {

    // This is the code in Javascript to use in the TTN decoder function for
    // the Thingspeak Integration
    // It decodes an uplink message from a buffer (array) of bytes to JSON
    // formatted object
    // with at least one of the allowed fields. ThingSpeak can accept up to 8
    // fields of data per channel,
    // a status field, and three position fields including latitude, longitude,
    // and elevation.

    var bytes = input.bytes;
    var MessageNumber = bytes[0]<<8 | bytes[1];
    //unsigned short integer (2bytes); bytes in big endian order
    var batmV = (bytes[2]<<24) >> 16 | bytes[3];
    // signed short integer (2bytes); bytes in big endian order
    var WaterLevelMeters = (bytes[4]<<24) >> 16 | bytes[5];
    // signed short integer (2bytes); bytes in big endian order
    var waterTempCentigradeCelsius = (bytes[6]<<24) >> 16 | bytes[7];
    // signed short integer (2bytes); bytes in big endian order
    var TempCentigrade = (bytes[8]<<24) >> 16 | bytes[9];
    // signed short integer (2bytes); bytes in big endian order
    var pressurehPa = (bytes[10]<<24) >> 16 | bytes[11];
    // signed short integer (2bytes); bytes in big endian order
    var relHumiditypermil = (bytes[12]<<24) >> 16 | bytes[13];
  }

```

```

// signed short integer (2bytes); bytes in big endian order
var unixtimesecs = (bytes[16]<<24) >> 16 | bytes[17];
// long signed integer 4 bytes = int:32 'I'=long unsigned integer 4 bytes =
int:32
return {
  data: {
    field1: MessageNumber,
    field2: batmV/1000,
    field3: (WaterLevelMeters/100000)+9,
    field4: waterTempCentigradeCelsius/100,
    field5: TempCentigrade/100,
    field6: pressurehPa/10,
    field7: relHumiditypermil/10,
    field8: unixtimesecs
  },
  warnings: [],
  errors: []
};
}

```

Figure 4-32: Decoder Function for the WMN

The script for the SAMN can be seen in Figure 4-33.

```

function decodeUplink(input) {

  // This is the code in Javascript to use in the TTN decoder function for
  the Thingspeak Integration
  // It decodes an uplink message from a buffer (array) of bytes to JSON
  formatted object
  // with at least one of the allowed fields. ThingSpeak can accept up to 8
  fields of data per channel,
  // a status field, and three position fields including latitude, longitude,
  and elevation.
  var bytes = input.bytes;
  var MessageNumber = bytes[0]<<8 | bytes[1];
  //unsigned short integer (2bytes); bytes in big endian order
  var batmV = (bytes[2]<<24) >> 16 | bytes[3];
  // signed short integer (2bytes); bytes in big endian order
  var hPa1 = (bytes[4]<<24) >> 16 | bytes[5];
  // signed short integer (2bytes); bytes in big endian order
  var soilTempCelsius = (bytes[6]<<24) >> 16 | bytes[7];
  // signed short integer (2bytes); bytes in big endian order
  var airTempCentigradeCelsius = (bytes[8]<<24) >> 16 | bytes[9];
  // signed short integer (2bytes); bytes in big endian order
  var airRelHumiditytypermil = (bytes[10]<<24) >> 16 | bytes[11];

```

```

// signed short integer (2bytes); bytes in big endian order
var TempCentigrade = (bytes[12]<<24) >> 16 | bytes[13];
// signed short integer (2bytes); bytes in big endian order
var unixtimesecs = (bytes[16]<<24) >> 16 | bytes[17]
// 'i'=long signed integer 4 bytes = int:32 'I'=long unsigned integer 4 bytes =
int:32
return {
  data: {
    field1: MessageNumber,
    field2: batmV/1000,
    field3: hPa1/10,
    field4: soilTempCelsius/100,
    field5: airTempCentigradeCelsius/100,
    field6: airRelHumiditytypermil/100,
    field7: TempCentigrade/100,
    field8: unixtimesecs,
  },
  warnings: [],
  errors: []
};
}

```

Figure 4-33: Decoder Function for the SAMN

The `decodeUplink` function for both nodes returned 8 fields of data containing the measurements taken by the systems. The `var` function created a variable for the bits in which the data was stored and the function implemented bit shifting through the payload to decode the payload. The function then returned the 8 fields of data which corresponded to different measurements that were sent to TTN.

4.6.5 Things Speak

This study utilised the Things Speak platform to convert raw data collected by sensors into data that can be visually displayed and analysed. Things Speak collects data from TTN and presents the data visually to an end-user. Once data was successfully sent to TTN and then decoded using the payload formatter, a webhook integration was setup which allowed the data sent to TTN to be collected and stored on the Things Speak application. To integrate TTN and Things Speak, the network and the application were connected through a write API key.

To generate a write API key, a channel needs to be created on the Things Speak website. Two channels were created for this study which captured data from both the WMN and the SAMN. Both channels adopted the REST API which displays the data in a user-friendly manner on the Things Speak application. The REST API has all the functions and procedures required to meet the requirements with regards to displaying the hydrological measurements, and it was

chosen over MQTT as the study did not require the publish/subscribe communication protocol. Data sent to Things Speak was ordered in terms of fields as seen in the payload formatters in Figure 4-32 and 4-33. The fields were labelled and displayed graphically on the channel, were monitored in real-time and were downloaded for further data analysis at any point.

4.7 Summary of Method – System Design

A summary of the method of the System Design is as follows:

- i. The project requirements were discussed and outlined with reliability, performance and data output deemed as the most important for IoT-driven hydrological monitoring.
- ii. The LoRa communication methods were outlined where the network implementation, network configuration and message structure were detailed.
- iii. The hardware selection was detailed as well as the reason for the selection of the hardware components. MCU, sensor and power supply selection were based on the requirements of the system. The implementation and integration of the hardware components selected to create the sensor node were outlined and depicted.
- iv. Details of the software application were outlined by identifying the firmware design and sensor setup of the system. The firmware and setup were both based on code found in Pycom and Micro Python libraries. The backend servers for LoRa communication were detailed with LoRa using the TTN and Things Speak platforms.

5. Methodology – System Implementation

5.1 Overview – System Implementation

The system implementation of this research involved selecting a suitable site and performing practical field experimentations to analyse the system based on its performance as an IoT-driven hydrological monitoring system. Performing the practical field experiment involved selecting a time period and duration of the experiment, performing accuracy testing on the sensors that were measuring hydrological data as well as deploying the systems within the selected study site. Once the systems had been deployed, analysis was done with respect to the reliability, performance, and data output of the systems. Figure 5-1 displays an overview of the method followed by the system implementation for this research.

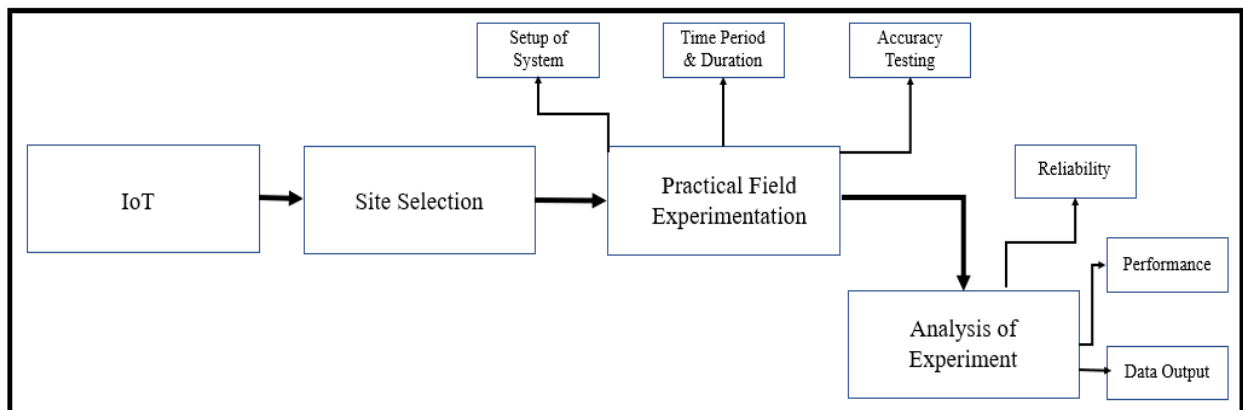


Figure 5-1: Overview of Methodology – System Implementation

5.2 Statement of Method – System Implementation

The study investigated the implementation of an IoT-driven hydrological monitoring system that measured data in real-time using LoRa communication. The assessment of the system involved practical experimentation in which the developed system was tested in the field. Before the experimentation begun, a study site was selected, accuracy testing was undertaken for the sensor and a duration and time period was determined for the experiment.

The practical experimentation involved setting up two hydrological nodes on the study site. The WMN was setup at the UCT dam, and the SAMN was setup on a tree above a parking lot near the NEB building. Once the practical experiment had been performed, the reliability and performance of the system was analysed based on the data collected during the field experiment.

5.3 Selection of Study Area

The requirements of the project influenced criteria for study site selection. The selection of a suitable catchment for this project was based on three critical aspects, namely safety, accessibility, and application. The application of the system was explored in South Africa and more specifically the CoCT and a locality map can be seen in Figure 5-2.



Figure 5-2: Locality Map of Cape Town, South Africa (ESRI, 2021)

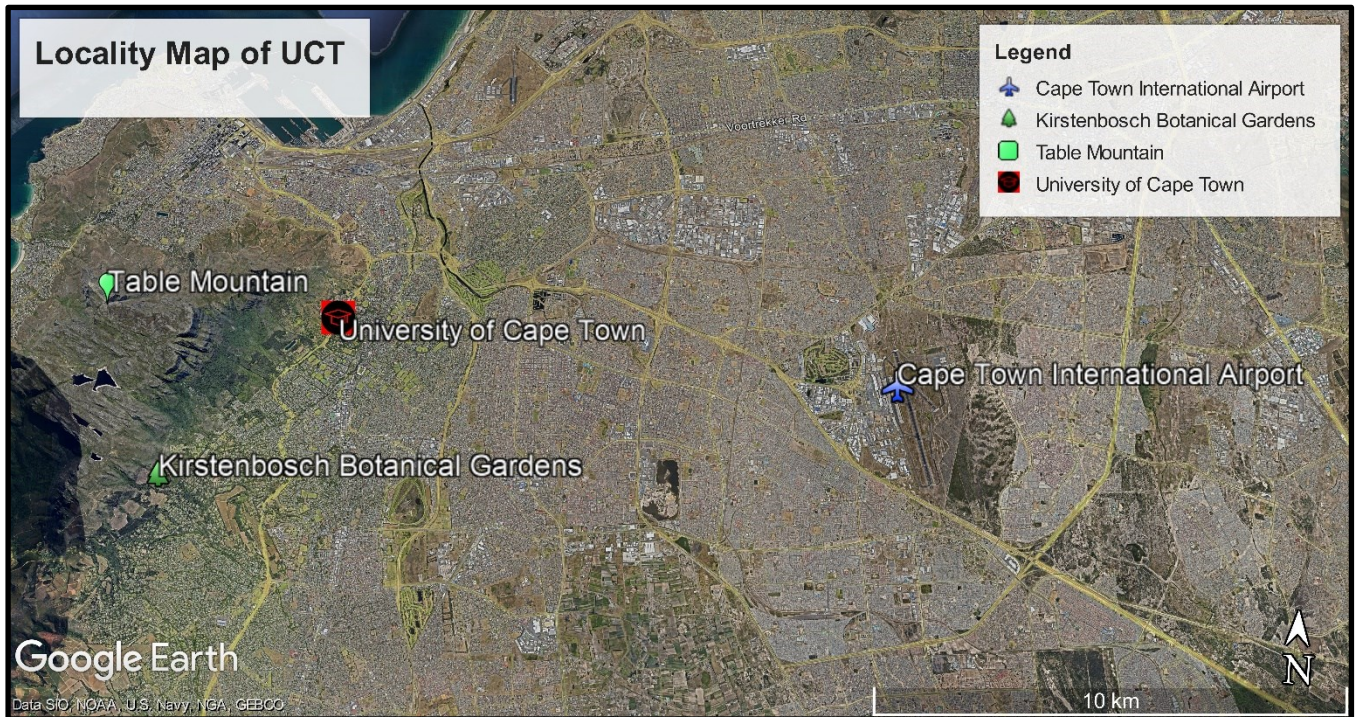


Figure 5-3: Locality Map of UCT, Cape Town (Google Earth, 2022)

Figure 5-3 shows the location of UCT with respect to other notable landmarks and areas in the CoCT. The UCT dam has the largest water storage capacity on campus and receives runoff from the down sloping section of Table Mountain above UCT. The dam provides the campus with an emergency water supply and amenity in the form of recreational activities, such as fly fishing and a focal point for students to gather. Figure 5-4 shows a locality map that references the placement of the WMN and SAMN on UCT's Upper Campus.

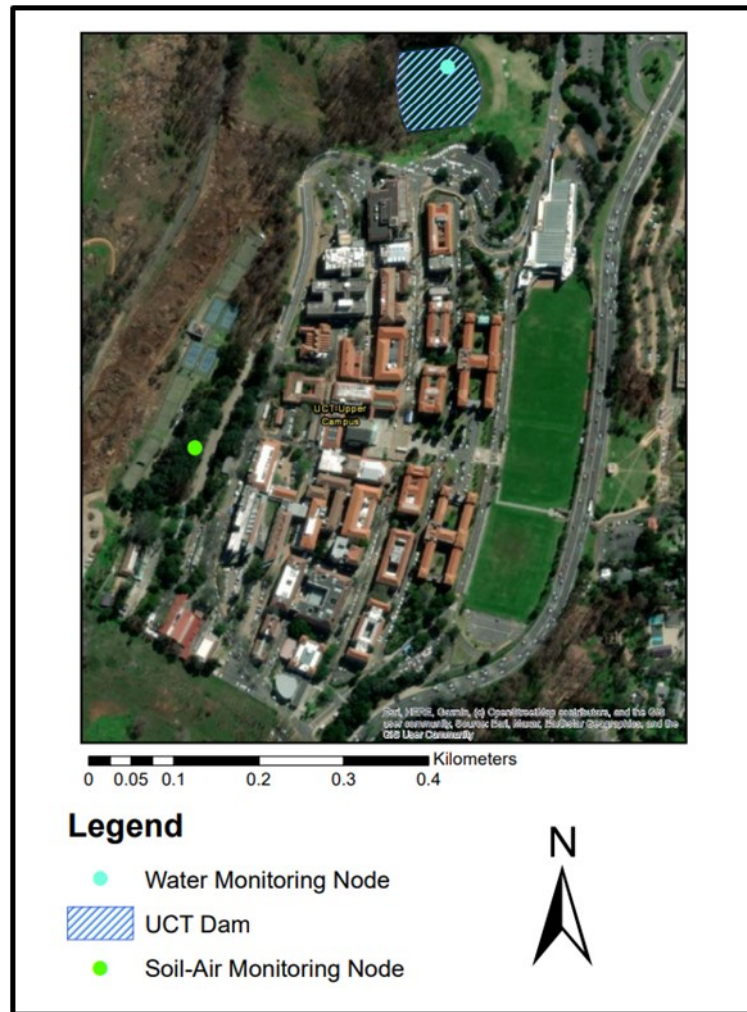


Figure 5-4: Locality Map of Sensors on UCT Upper Campus (ESRI, 2021)

The safety of the site is dependent on the likelihood of the equipment being tampered with or stolen. For this project, it was important to choose a site that had a low likelihood of these events occurring in order to minimise risks and allow the study to proceed uninterrupted. Furthermore, accessibility was important to ensure that the site was easy ad hoc visits. The application of the WMN requires a water body where water level and water temperature can be monitored while the SAMN requires a natural piece of land with soil layers.

The locality of the site as well as the on-site terrain and conditions are very important aspects to consider when deciding on an accessible site. An accessible site is one that is easy to travel to and easy to navigate through. Although these factors are used to choose a testing site, they should not be limitations and the system would be applicable to any situation and site in South Africa. The first site option for this project was the dam located on UCT upper campus. The earth dam has a varying water level and temperature which were both measured. The location of the SAMN testing site was located behind the NEB on UCT 's Upper Campus.

5.4 Practical Field Experimentation

The hydrological monitoring systems were monitored and assessed based on its efficiency and functionality regarding transmitting data at a rate that is acceptable for real-time monitoring. The systems were monitored over a 4-month period with varying weather conditions, allowing for more accurate testing conditions when compared to the actual implementation of the system.

5.4.1 Setup of Field Experiment

The setup of the field experiment involved strategic placing of the system to ensure safety from theft or vandalism, while remaining accessible in case of unexpected hardware or software failure. Reducing the system's susceptibility to theft and vandalism involved strategically placing the system out of site and reach whilst ensuring that the system remained within reach of the dam and ground to take data measurements. The placement was done at the most inaccessible position within the area of the dam. This was to deter individuals from tampering with or moving the system. The dam has two access points and, thus, placing the node within the dam, on the level gauge, made it inaccessible to the public. The WMN was mounted on a concrete slab that functions as a level gauge in the dam. Figure 5-5 shows the WMN setup on the level gauge in the dam. The electronics were housed in a waterproof container and secured to the gauge using an L-bracket. The water level sensor and water temperature sensor were enclosed within a PVC pipe and secured along the gauge using galvanised saddles. A detailed AutoCAD drawing of the WMN setup can be found in Appendix J-1.



Figure 5-5: System Setup of the WMN

The SAMN was setup on a tree located behind the NEB building on UCT's Upper Campus. The location of the tree can be seen on the UCT Upper Campus locality map as seen in Figure 5-4. Like the WMN, the SAMN was encased in a weatherproof enclosure and mounted on a tree. The soil moisture and soil temperature sensors were run down the tree into the soil at the base of the tree. An example diagram of how the soil moisture sensor and soil temperature sensor were placed into the ground can be seen in Figure 5-6. The sensors were secured in a PVC pipe with the sensors' heads submerged into the surrounding soil at a depth of 500 mm, and the cables were secured with saddles along the trunk of the tree. The temperature and humidity sensor, which was enclosed in the solar radiation shield, was mounted next to the weatherproof enclosure on the same tree. The radiation shield was mounted using an L-bracket and the setup of the SAMN can be seen in Figure 5-7. A detailed drawing of the SAMN can be found in Appendix J-2.

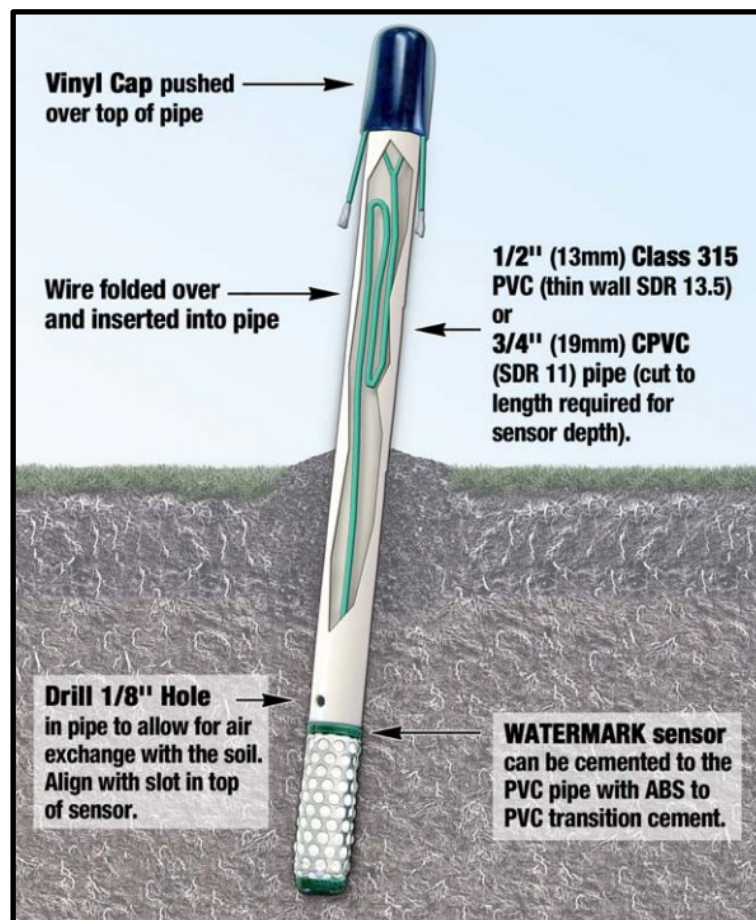


Figure 5-6: Example of Soil Moisture Sensor Deployment (Irrometer, 2020)



Figure 5-7: System Setup of the SAMN

5.4.2 Period and Duration of Experiment

The duration of the experiment was based on the context in which the system would be applied. Since the system was designed to constantly measure hydrological data in real-time, it needed to operate for several uninterrupted months and avoid unnecessary downtime that require unplanned visits for maintenance. For this reason, the system was tested for a minimum of 3 months, starting in June, which happened to be during the winter rainfall season in the CoCT. This provided an indication of the system's robustness and durability, over five months, in fairly harsh environmental conditions as Cape Town is known for its torrential rainstorms and high wind speeds during this time of the year.

5.4.3 Accuracy and Precision Testing

Before the IoT system was deployed in the field for testing, the accuracy and precision of the sensors used were tested to ensure reliability of collected data from the sensors. These tests were done under laboratory conditions using various equipment and methods explained in this section.

The accuracy and precision were tested by determining the percentage error of the data collected from the sensor. The percentage error was calculated by determining the difference

between the reported known measurement and the measurement taken by the sensor and then dividing the difference by the sensors full scale range. The equation for the percentage error of accuracy and precision for a sensor can be seen in Equation 5.1:

$$\% \text{ error} = \frac{\text{Known Measurement} - \text{Sensor Measurement}}{\text{Full Scale Range}} \quad (5.1)$$

The accuracy and precision testing setup can be found in Appendix K-1.

Testing the accuracy and precision of the Hafei WNK water level sensor involved placing the sensor in a 1000 mm transparent Perspex cylinder filled with water. 100 measurements were taken with the sensor and the depth readings obtained from the sensor were compared to spot height checks of the water in the cylinder using a measuring tape. When testing accuracy, the water level was altered after each iteration; and when measuring precision, the water level was kept constant for each iteration. Based on the accuracy testing, it was determined that the sensor had an average error range of ± 5 mm and a 0.10% error. The results for the precision testing showed that the average error range was 0 mm, and the sensor had a 0% error. The results of the accuracy tests indicated that in the range of 0-1000 mm the sensor had a 0.15% better accuracy than that specified by the manufacturer.

Determining the accuracy and precision of the DS18B20 water and soil temperature sensor involved taking temperature readings of water and soil respectively in a bucket with a known temperature obtained from a laboratory sensor probe. To determine the known water temperature and known soil temperature, a Center 307 Type K temperature probe was used. The probe is used in the UCT civil laboratory to measure the temperature of the water baths for concrete cube testing and has a specified accuracy of $\pm 0.3^\circ\text{C}$ by the manufacturer. For accuracy testing, the temperature of the water and soil was changed after each iteration, while for precision testing the temperature was kept constant. The temperature measured on the probe was compared to the temperature measured by the sensor for 100 samples. The results for the accuracy tests of the water temperature sensor indicated that the sensor had an average temperature error range of $\pm 0.026^\circ\text{C}$ and a 0.014% error. The results for the precision testing of the water temperature sensor indicated an average error range of $\pm 0.012^\circ\text{C}$ and a percentage error of 0.000067%. The results for the accuracy testing of the soil temperature sensor had an average temperature error range of $\pm 0.3^\circ\text{C}$ and the percentage error was 0.17% when compared with the sensor probe. The results for the precision of the soil temperature sensor indicated the sensor had an average temperature error of $\pm 0.24^\circ\text{C}$ and a percentage error of 0.0014%. The accuracy results for the DS18B20 sensor align with the accuracy specified by the manufacturer.

The SHT31 air temperature sensors accuracy and precision were also measured by obtaining a known temperature with the aforementioned sensor probe and then comparing the readings. To measure accuracy of the air temperature sensor, the ambient air temperature of the enclosure housing the temperature was changed after each iteration, and for precision testing the

ambient air temperature in the enclosure was kept constant. The testing involved taking 100 samples and the accuracy readings for the air temperature indicated that the average temperature error range was $\pm 0.099^{\circ}\text{C}$ and the percentage error of the sensor was 0.0006%. The precision testing indicated the sensor had an average temperature error range of $\pm 0.015^{\circ}\text{C}$ and a percentage error of 0.0001%. This aligned with the specified accuracy outlined by the manufacturer. Appendix K-2 has an extract of the results from the accuracy and precision tests. After precision and accuracy were tested, the nodes were ready for deployment.

5.5 Analysis from Experiment

The entire IoT system (sensing data to visual representation of data) and the data produced by the experiment were analysed in terms of its usability with respect to hydrology. Pilot testing of the system provided information on the system's performance under real-world operating conditions. The analysis focused on key characteristics, such as robustness of the system, reliability, performance, and user-friendliness. The analysis considered the process of monitoring data remotely, beginning with sensor devices taking measurements and transmitting them through a node to a gateway. The gateway connected the nodes to the internet where it can be stored via cloud storage and/or analysed using a backend application. The data produced by the system was analysed to determine accuracy and efficiency in providing real-time hydrological monitoring using IoT (Akhila & Senthil Prabha, 2019). Once the output data was analysed, feedback was provided to guide decisions needed to improve the system for the use-case.

5.5.1 Data Output of IoT System

The main requirement of the system was its ability to produce hydrological data in real-time. This section of the study considered the measurement data generated from the system, as well as the way in which the system displayed data to the end user. The data output of the systems was also checked for accuracy and consistency. Accuracy was measured using multiple methods prior to and during field deployment, while the consistency of the systems data output was checked using a nearby South African Weather Services (SAWS) weather station in Kirstenbosch gardens.

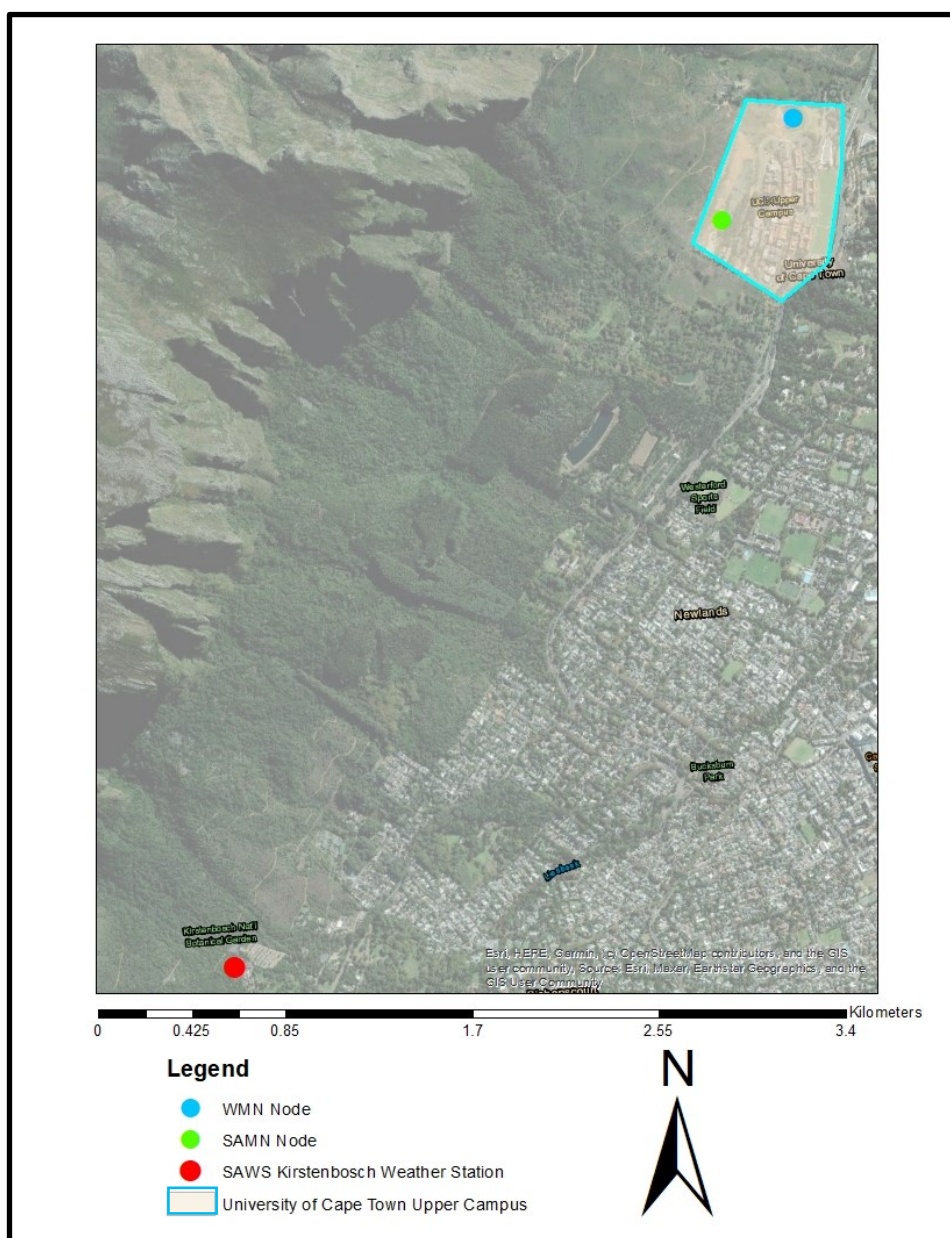


Figure 5-8: Locality Map of SAWS Kirstenbosch station in comparison to the Nodes (ESRI, 2021)

Figure 5-8 shows a locality map of the SAWS Kirstenbosch weather station in comparison to the monitoring nodes. Rainfall, air temperature and relative humidity data that was measured during the deployment of the WMN and SAMN were obtained from the SAWS station to make comparisons and check consistency.

5.5.1.1 Water Level Data

The water level data was displayed graphically on the Things Speak website. The data displayed on a scatter plot graph was updated in 15-minute intervals after every successful transmission of data. The data was stored on the Things Speak application where it could be exported to Excel as a Comma Separated Values (CSV) file and the data could be graphically represented and analysed. Figure 5-9 shows how the water level data was displayed on the Things Speak website during deployment.

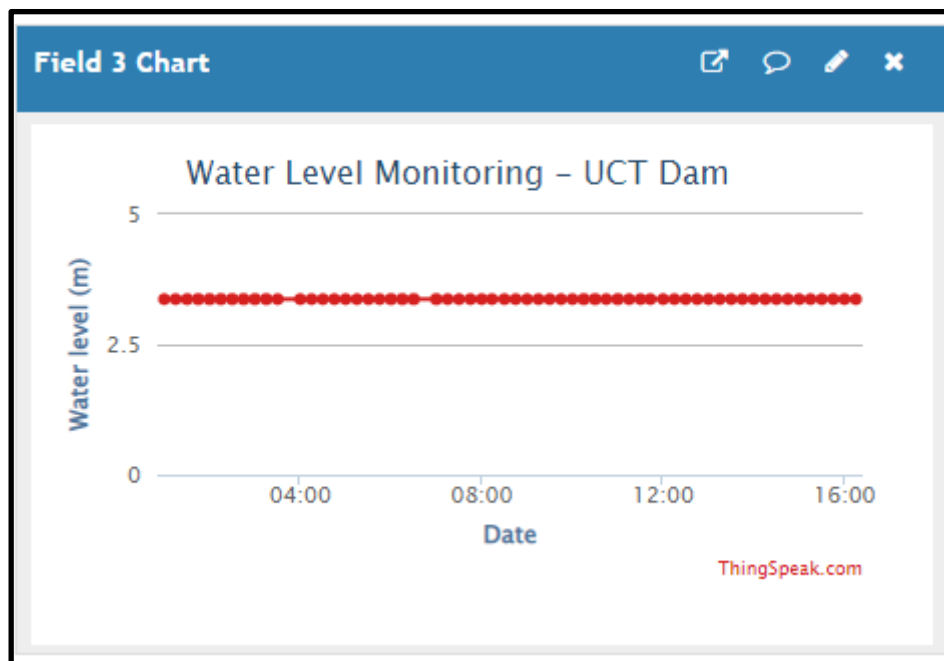


Figure 5-9: Water Level Data Visualisation for Things Speak

To check the validity and consistency of the data, spot checks were done weekly and sometimes bi-weekly by visiting the dam and checking the level gauge of the dam. The data output was also compared with the rainfall data collected by the nearby SAWS station.

5.5.1.2 Water Temperature Data

Water temperature data was displayed via graphs on the Things Speak platform. The data was recorded in degrees Celsius and plotted in a scatter graph that was updated in 15-minute intervals. The stored data was accessed at any time on Things Speak by exporting a CSV file to Excel. Figure 5-10 is an example of how the data was displayed on Things Speak.

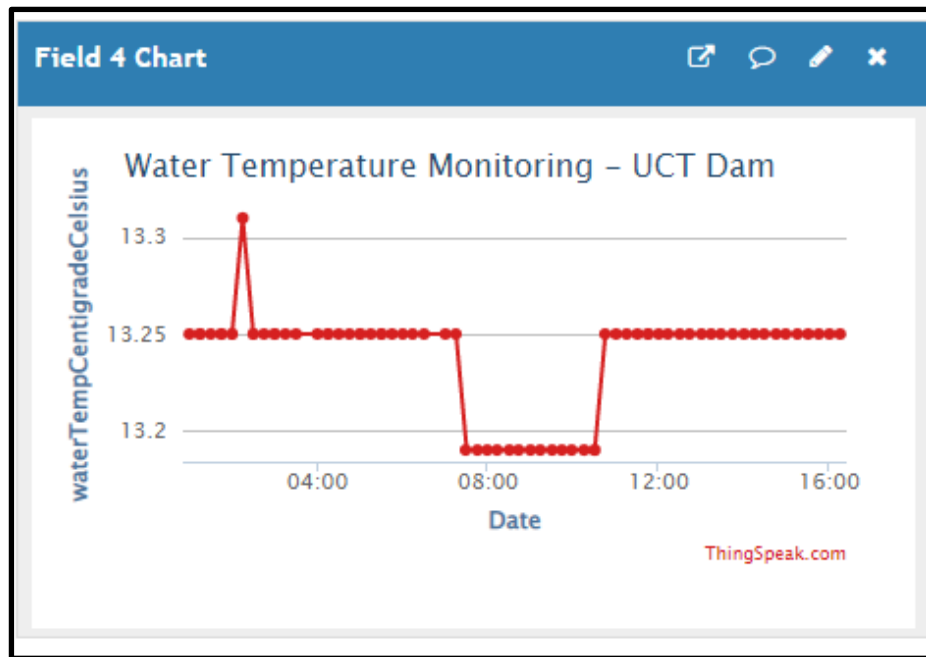


Figure 5-10: Water Temperature Data Visualisation for Things Speak

Due to the positioning of the WMN, accurate spot checks were not possible for the water temperature node. Thus, the accuracy of the sensor was determined using accuracy checks done in Section 5.4.3.

5.5.1.3 Soil Moisture Data

The soil moisture data was displayed on scatter plots through the Things Speak platform. Soil moisture data was measured in kPa, the soil temperature measured in degrees Celsius and displayed alongside soil moisture. Figure 5-11 is an example of how the data was displayed on Things Speak.

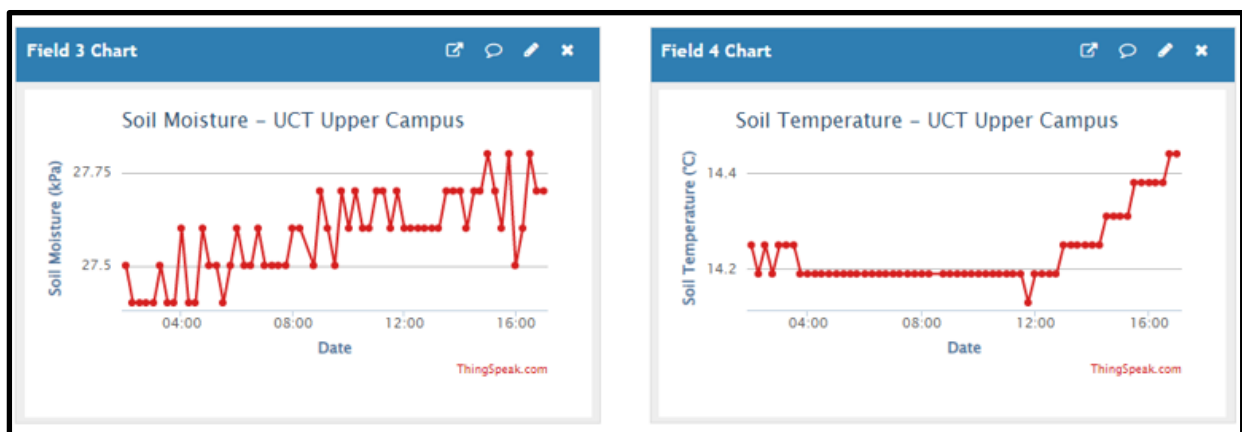


Figure 5-11: Soil Moisture Data Visualisation for Things Speak

Spot checks were not possible for the soil moisture data as the relevant equipment was not available during this research. However, the soil moisture data was compared to the rainfall data collected by a nearby station to check consistency of the sensor during rainfall events and the accuracy of the soil temperature sensor was confirmed in section 5.4.3.

5.5.1.4 Air Temperature & Humidity Data

The air temperature and relative humidity data was displayed on Things Speak via a series of scatter plots and gauges. The air temperature was measured and displayed in degrees Celsius while the air humidity was measured as a relative value and displayed accordingly. Figure 5-12 shows how the data displayed on the Things Speak platform.

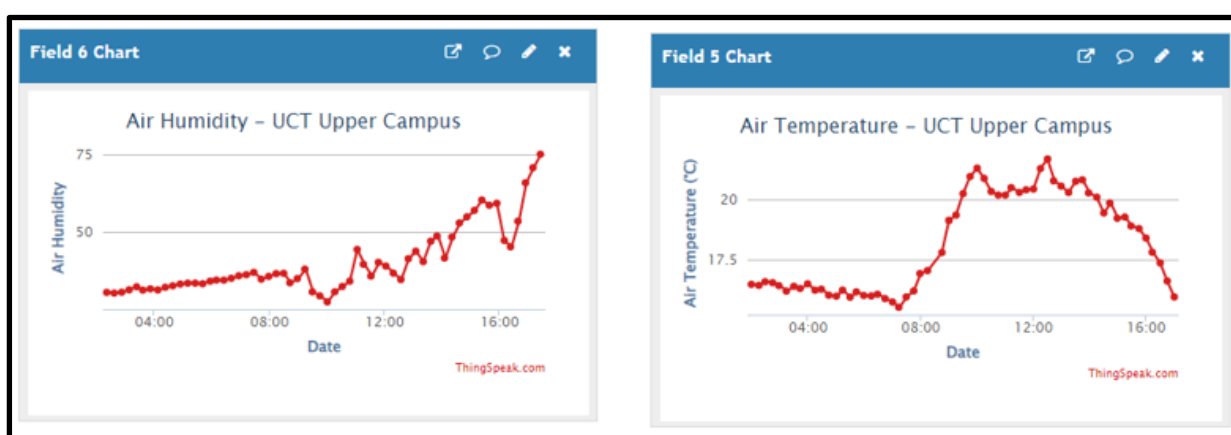


Figure 5-12: Air Temperature & Humidity Data Visualisation

Data produced by the sensor was compared to air temperature and relative humidity data produced by the SAWS Kirstenbosch weather station to check for accuracy and consistency.

5.5.2 Reliability Analysis

Testing the reliability of an IoT-driven system involved analysing parameters that ensure reliability during operation. The parameters that were tested and analysed in this study were based on the communication and robustness of the system. This section provides instructions on what methods were used to test these parameters.

5.5.2.1 Robustness of System

The physical components of the IoT systems were monitored on a regular basis and the reaction to the natural environment was analysed at the end of the deployment with the findings documented. The study investigated the durability of the systems when exposed to natural conditions.

Abnormalities during general operation of the system were investigated as well as how the system reacted to these abnormalities. Abnormalities are considered occurrences or events that are unplanned and may interfere with the general operation of the system.

5.5.2.2 Communication of System

To ensure that the systems were running within the regulations specified by ICASA (2015), the duty cycle of the systems needed to be checked and analysed. Determining duty cycle involved using an online airtime calculator for LoRaWAN communication. Testing the communication of the system involved monitoring the rate at which the system successfully transmits messages to the relevant backend applications. The system aimed to transmit hydrological measurements through the system every 15 minutes. The communication reliability for the LoRa network was analysed by tracking the number of packets lost during transmission. Each time a node wakes up, a message number that goes up in chronological order was assigned to the packet of data. Lost data packets show up as gaps in the chronological number order and the number of packets lost was tallied up. Using Equation 5.1, the percentage of packets lost by the system was determined as follows:

$$\% \text{ Packets Loss} = \frac{\text{Packets Lost}}{\text{Packets Created}} \quad (5.1)$$

Where:

% Packet loss – Percentage packets lost for the duration of the experiment

Packets lost – Number of packets lost for the duration of the experiment

Packets created – Number of packets created by the node for the duration of the experiment

Once the experiment had come to an end, the packets lost and created were calculated for both nodes and the % packets lost was determined and compared to IoT standards.

5.5.3 Performance Analysis

Performance testing looked at how well the internal integrated system and the external applications were functioning to achieve the desired objective of the IoT system (Kingatua, 2018). To analyse the performance of the IoT system, load and power consumption parameters were tested in this study. This section provides instruction on the methods used to test these parameters.

5.5.3.1 *Load Testing*

The load testing evaluated the software and hardware of the system in a practical environment. The aim was to yield flaws to provide feedback which was then used to improve the implementation of the system so that it may be better suited to meet the performance requirements in future projects.

5.5.3.2 *Power Consumption Testing*

Power consumption testing involved measuring the amount of time the system operated without maintenance. The aim was to extend the life cycle by increasing energy efficiency. The number of days operating without maintenance was documented and compared with the theoretical prediction of power consumption as discussed in section 4.5.5. If the experiment ran for 3 months without the need to replace or recharge the battery, the system life was estimated by determining the projected battery life of the system if the node completed its life cycle. This gave a projected estimation of the node's life cycle and allowed for the planning of required maintenance where necessary.

5.6 Summary of Method – System Implementation

A summary of the method is as follows:

- i. The study site selection procedure was outlined and discussed. The UCT dam and UCT Upper Campus were chosen as study sites due to their applicability, safety, and accessibility for the study while the main requirements for the project were based on reliability and performance of the system.
- ii. The practical field experiment was outlined, and the method used was detailed. The setup and duration were explained while the time period of the experiment was said to occur during Cape Town's rainy season. The details of accuracy and precision testing done for each sensor was documented.
- iii. The analysis of the experiment based on the reliability, performance and output of the system was outlined and detailed. Detailed analysis on reliability, performance and output was done based on communication, robustness, load, power consumption and output testing procedures.

6. Results from Experiment

In this chapter, the IoT system was assessed based on the data collected, reliability, and performance. The reliability analysis involved assessing the efficiency and reliability of the communication protocol of the nodes and documenting the robustness of the nodes throughout the experimental deployment. Performance analysis assessed the load testing of the system in which the hardware and software of the system were analysed in a practical environment. It also incorporated power consumption analysis of the system wherein the life span of the node was analysed.

6.1 Assessment of Data Output from IoT Nodes

This section presents an assessment of the data output from the IoT nodes and how the data was collected and displayed visually to the end user. The data assessed included water level, water temperature, soil moisture, air temperature and relative humidity. Other measured data used to assess monitoring performance and reliability of the system included battery voltage, enclosure temperature, enclosure humidity, and enclosure pressure.

Due to power outage problems in the CoCT, the deployment of the sensor nodes was split into two phases i.e., Water Monitoring Node 1 (WMN1), Water Monitoring Node 2 (WMN2), Soil-Air Monitoring Node 1 (SAMN1) and Soil-Air Monitoring Node 2 (SAMN2). The WMN deployment had two phases that lasted a total of 45 days with WMN1 running for 22 days and WMN2 running for 23 days. The SAMN deployment also had two phases, with SAMN1 lasting 10 days and SAMN2 lasting 111 days. SAMN2 deployment for this research was ended on the 01/11/2022 with all data collected up to that point included in the research. Extracts of the data collected by WMN can be found in Appendix L-1 and extracts of data collected by SAMN can be found in Appendix L-2.

6.1.1 Water Level Data

The water level data for this research was measured using a submersible water level sensor which collected depth values for the UCT dam, as can be seen in Figure 6-1. The range of the submersible water level sensor was 5m and, thus, it was not possible to get a reading of the actual dam level as it fell out of the sensor's range. To mitigate this, the sensor was mounted on the level gauge with the probe fixed to the 9m point of the gauge. To determine the actual level of the dam, 9m was added to the readings produced by the sensor. According to the data, there was a gradual rise in dam level. More specifically, the dam level increased from 9.150 to 10.727 m between the 12th and 15th of June during WMN1. After this, there was a steady increase to the dam level due to scattered rainfall events in the UCT area, and the final reading of the dam level before WMN1 ended was 11.221 m. WMN2 had a steady increase in depth with the time period being drier than the previous deployment. The initial dam level at the start of WMN2 was

11.799m and the last viable reading read 12.083m. As per the capacity of the dam, the water level reading did not rise pass 12.100m and this was consistent with the height of the spillway of the dam which is documented as 12.000m.

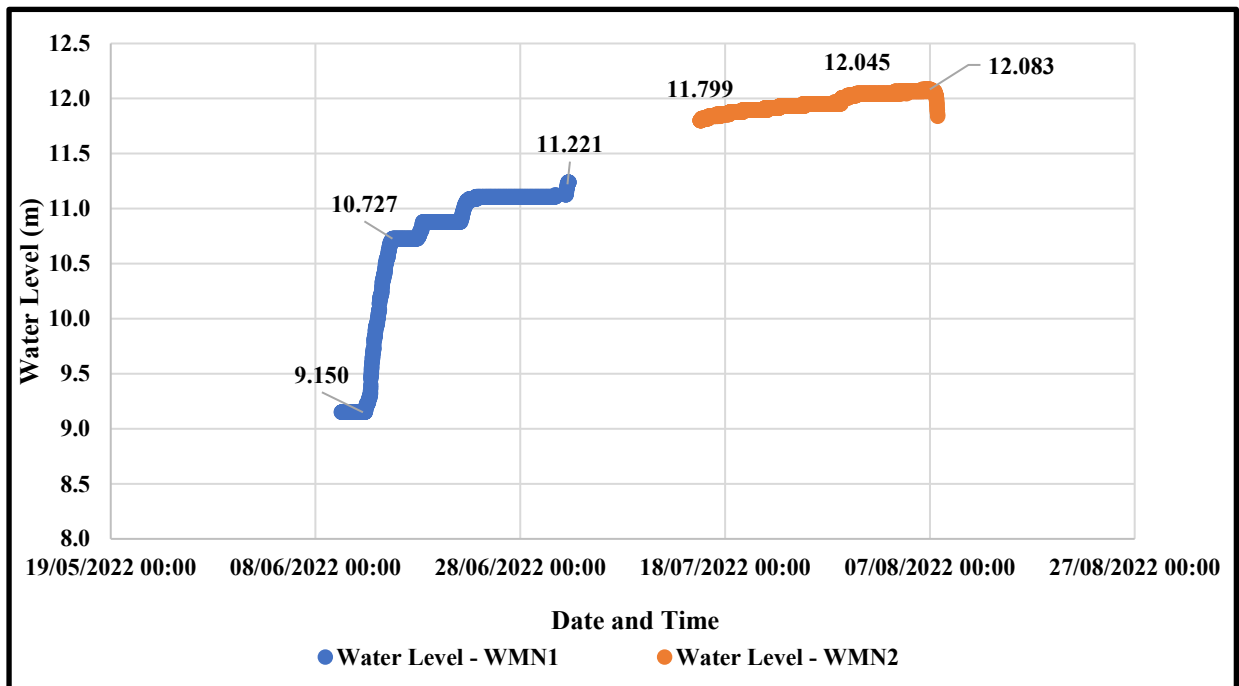


Figure 6-1: Time Series of Water Level Data Measured from the UCT Dam

Real-time display of the data involved creating a widget on Things Speak, which produced a gauge that displays the real-time water level of the UCT dam as seen in Figure 6-2. The gauge updated after every measurement was successfully sent to Things Speak and this method was the best way to visually represent the data alongside the graphical representations presented in section 5.5.1.

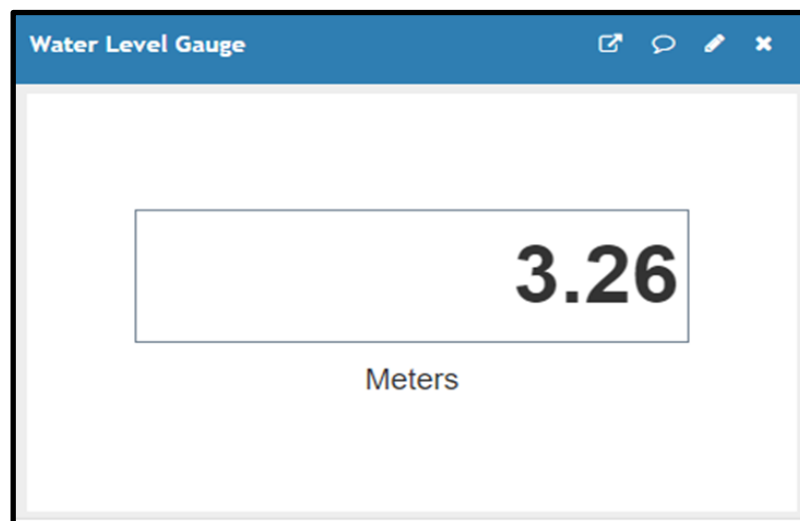


Figure 6-2: Gauge Displaying Water Level from Things

6.1.2 Water Temperature Data

Water temperature data for the UCT dam was collected using a 5m DS18B20 water temperature sensor and the data collected over the two iterations of deployment can be seen graphically in Figure 6-3. Water temperature at the start of WMN1 was 15.75°C.

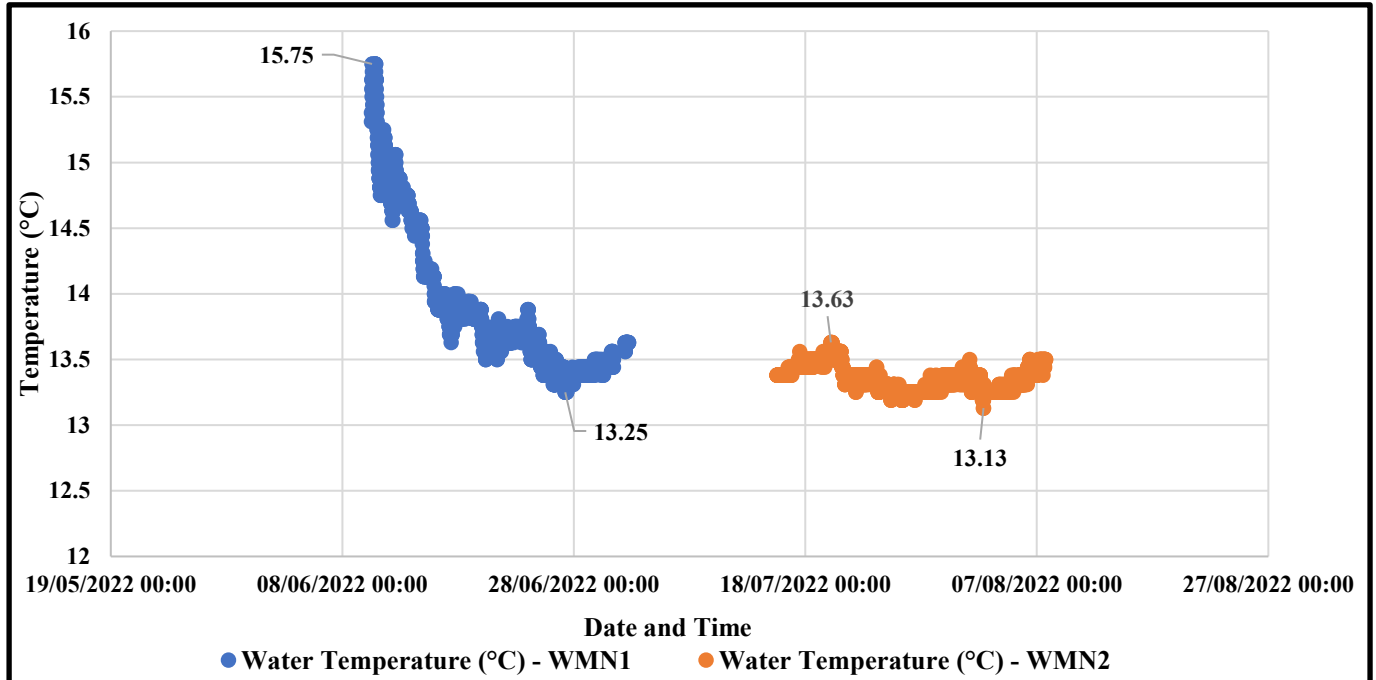


Figure 6-3: Time Series of Water Temperature Data Measured from the UCT Dam

It began to drop exponentially until it reached 13.25°C and, from this point, the temperature of the water remained between the range of 13.13 and 13.63°C throughout the remainder of WMN1 and for the duration of WMN2. The gradual drop in temperature came after the major rainfall events over the 13th and 14th of June which, as seen in Section 6.1.1, increased the dam level by 1.577 m. Figure 6-4 displays the daily minimum, maximum and average values for the full duration of the deployment. Over the course of the deployment, the highest recorded water temperature was 15.75°C and the lowest recorded water temperature was 13.13°C. The average across the entire deployment was 13.60°C. August had the lowest average temperature recordings (13.33°C) and the lowest single temperature (13.19 °C) while June had the highest recording(15.75°C). Table 6-1 shows the daily min, max and average water temperatures of the dam for each month during deployment. The daily outputs of min, max and average water temperature values for the WMN deployment can be seen in Appendix M-1.

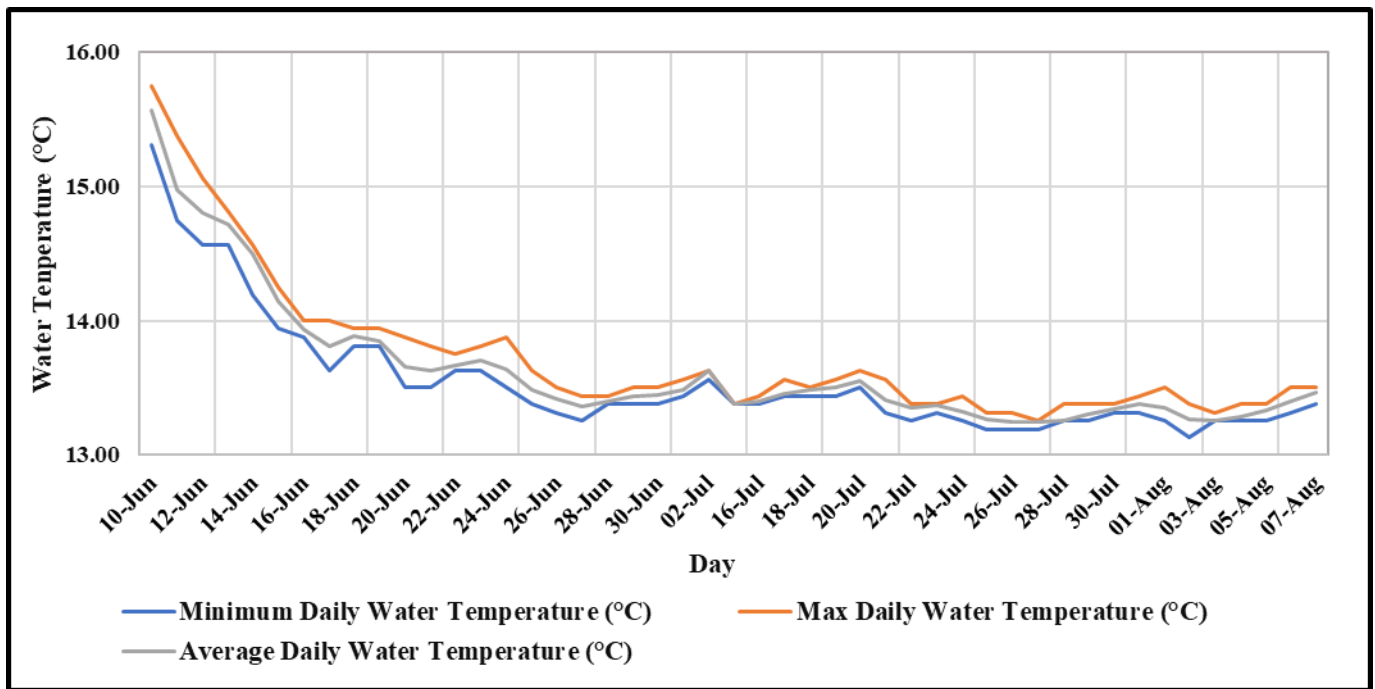


Figure 6-4: Daily Min, Max and Average Water Temperatures Across the WMN Deployment

Table 6-1: Monthly Average, Min and Max Values for Water Temperature Across WMN Deployment

Month	Average	Min	Max
June	13.90	13.25	15.75
July	13.38	13.19	13.63
August	13.33	13.13	13.50
WMN Total	13.60	13.13	15.75

In Things Speak, there was a water temperature gauge which displayed the real-time temperature at the UCT dam with updates occurring in 15-minute intervals. A screenshot of the temperature gauge displaying the dam water temperature can be seen in Figure 6-5

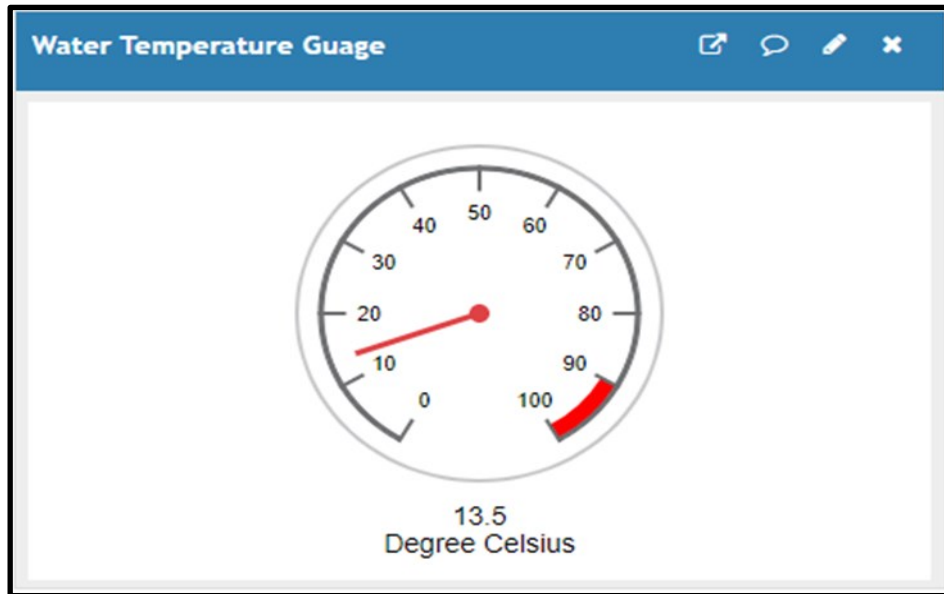


Figure 6-5: Gauge Setup on Things Speak to Display Real-time UCT Water Temperature

6.1.3 Soil Moisture Data

The soil moisture data was collected using two sensors to obtain a pressure reading for the soil. The irrometer watermark sensor was used to measure soil moisture while the DS18B20 temperature sensor was used to increase the accuracy of the measurement by considering the temperature of the soil. During SAMN1 the initial recording of soil moisture was 10.4 kPa. The soil moisture increased until it reached 37.8 kPa and then dropped sharply to 23.3 kPa. This trend of gradual increasing of soil moisture with a sudden drop could be seen throughout the deployment of SAMN2. Towards the end of the deployment, the soil moisture began to rise exponentially due to the increase in ambient air temperature and reduction in rainfall events. During this exponential increase, there were minor decreases in soil moisture due to intermittent rainfall events for the months of September and October. The final reading produced by the soil moisture sensor was 206.4 kPa indicating a dry soil. Figure 6-6 provides a time series of the soil moisture data collected during the deployment.

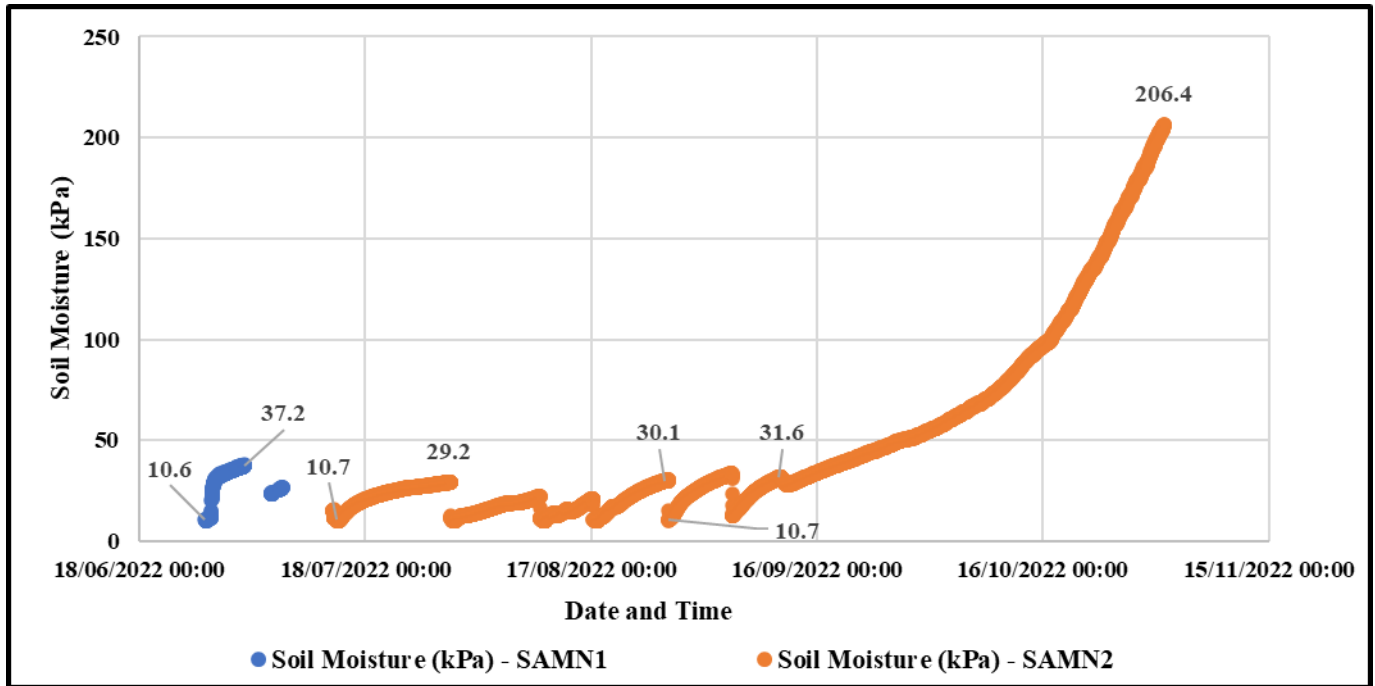


Figure 6-6: Time Series of Soil Moisture Data Measured from UCT Upper Campus

Table 6-2 gives a monthly statistical summary of the soil moisture data across the SAMN deployment. November values were only for a single day of recording and, thus, are not considered viable when analysing the average monthly data. The average soil moisture during the SAMN deployment was 49.07 kPa while the minimum and maximum were 9.80 kPa and 206 kPa, respectively. The lowest monthly average (18.76 kPa) was recorded during August while the lowest soil moisture recording (9.80 kPa) was in July. The highest (206.60 kPa) moisture reading was recorded in November with the highest monthly average (110.83 kPa) recorded in October.

Table 6-2: Monthly Soil Moisture (kPa) data across SAMN deployment

Month	Average	Min	Max
June	31.04	10.30	36.70
July	22.24	9.80	37.80
August	18.76	10.20	30.40
September	36.17	12.40	55.20
October	110.83	54.80	206.00
November	206.04	205.40	206.60
SAMN Total	49.07	9.80	206,60

Figure 6-7 displays the soil temperature data produced during the deployment of the SAMN. The soil temperature data followed a similar trend to the soil moisture data and this was expected considering there is a positive relationship between soil moisture (kPa) and soil temperature (°C). During periods that temperatures increased, there was an increase in moisture and this can be seen from the positive correlation in Equation 4.5 between soil moisture (cb) and soil temperature.

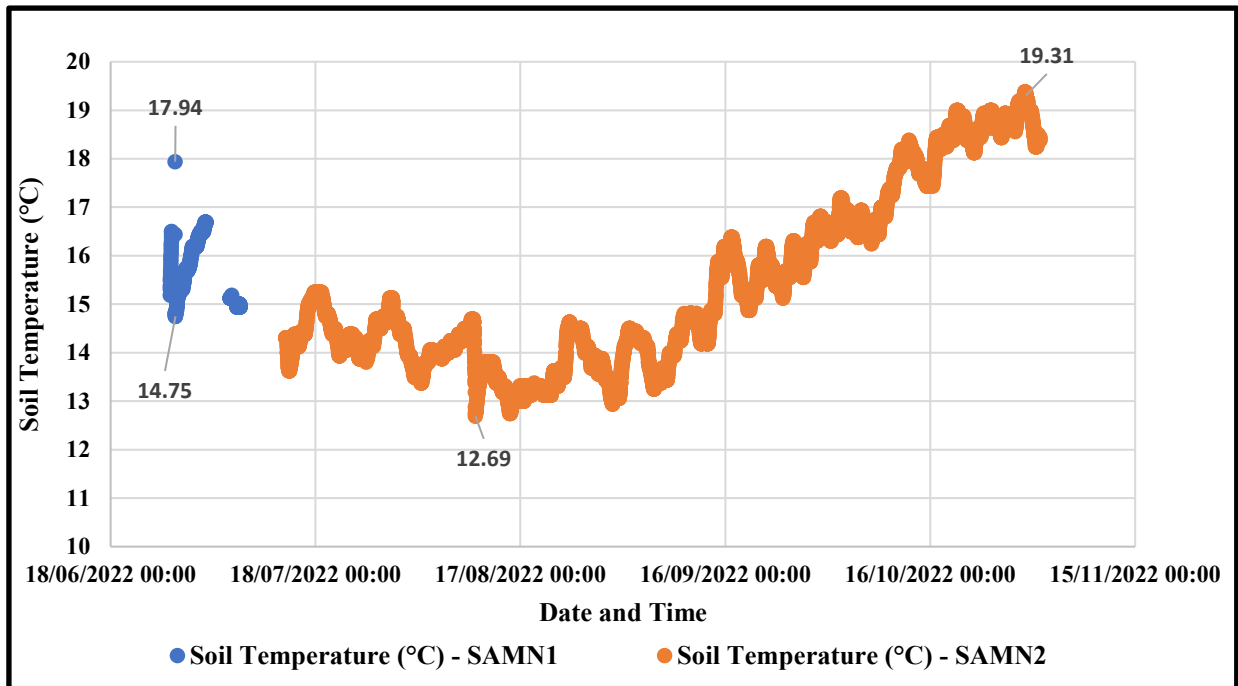


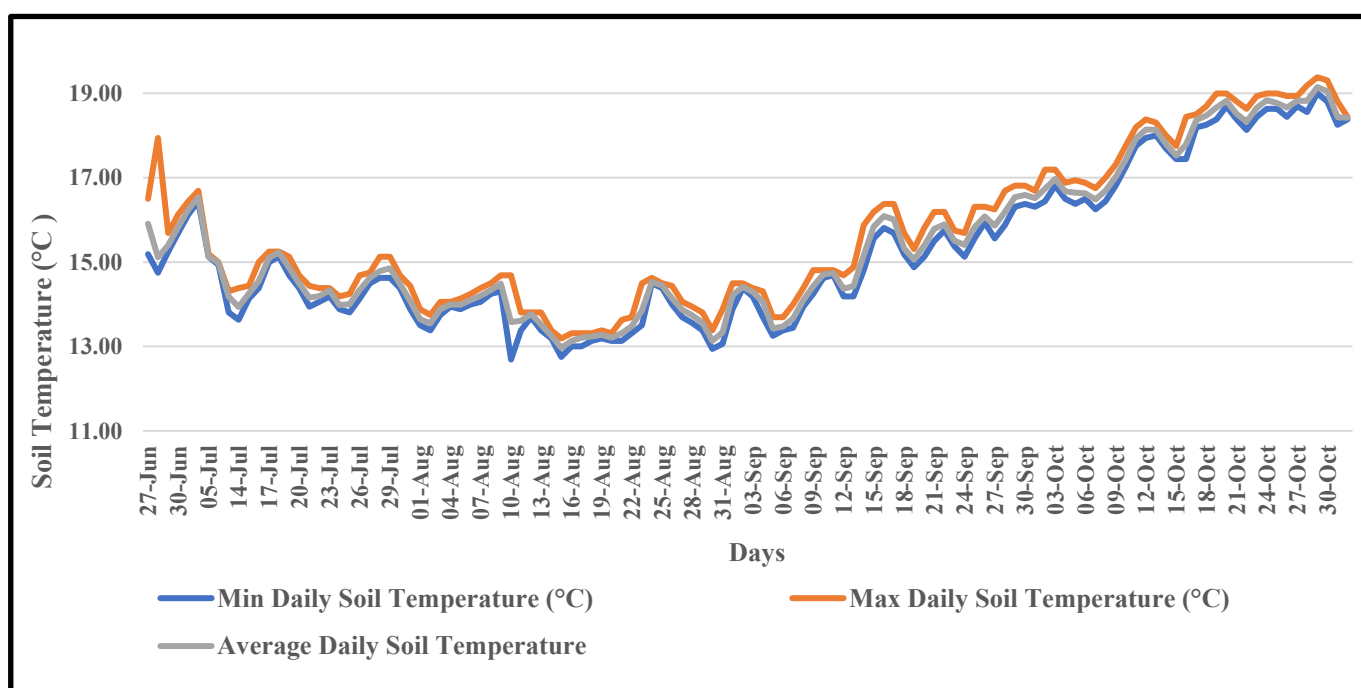
Figure 6-7: Time Series of Soil Temperature Data Measured from UCT Upper Campus

The total average temperature for the duration of the SAMN deployment was 14.30°C, while the minimum and maximum temperatures were 12.69°C and 17.94°C, respectively. Table 6-3 gives the monthly minimum, maximum and average values for the SAMN deployment. August had the lowest recorded temperature reading (12.69°C) while October had the highest (19.38°C). The lowest average soil temperature recording (13.69°C) was in the month of August.

Figure 6-8 depicts the daily min, max and averages for the soil temperature through the deployment of SAMN. The daily min, max, and averages for soil temperature can be found in Appendix M-2.

Table 6-3: Soil Temperature (°C) data Across WMN Deployment

Month	Average	Min	Max
June	15,73	14,75	17,94
July	14,57	13,63	16,69
August	13,69	12,69	14,69
September	15,10	13,25	16,81
October	17,92	16,25	19,38
November	18,42	18,38	18,44
SAMN Total	15.41	12,69	19,38

**Figure 6-8: Daily Min, Max and Average Soil Temperatures Across the SAMN Deployment**

Displaying the data visually on Things Speak can be seen in Figure 6-9. The soil moisture and soil temperature were displayed on gauges that updated every 15 minutes alongside time series graphs of the data.

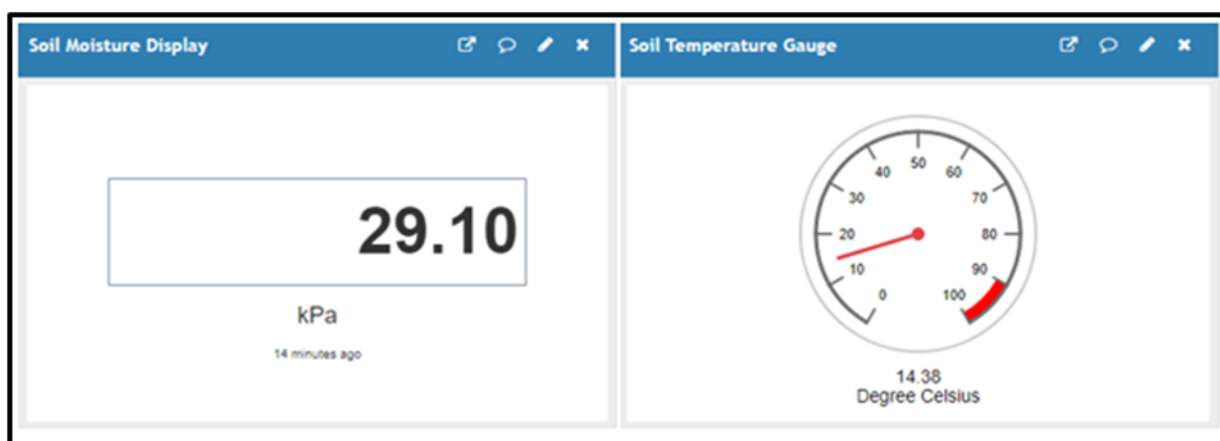


Figure 6-9: Gauge Setup on Things Speak to Display Soil Moisture and Temperature

6.1.4 Air Temperature & Humidity Data

The air temperature and relative humidity data were measured using a weatherproof temperature and humidity sensor encased in a protective radiation shield. The air temperature (°C) data produced by the sensor can be seen in Figure 6-10.

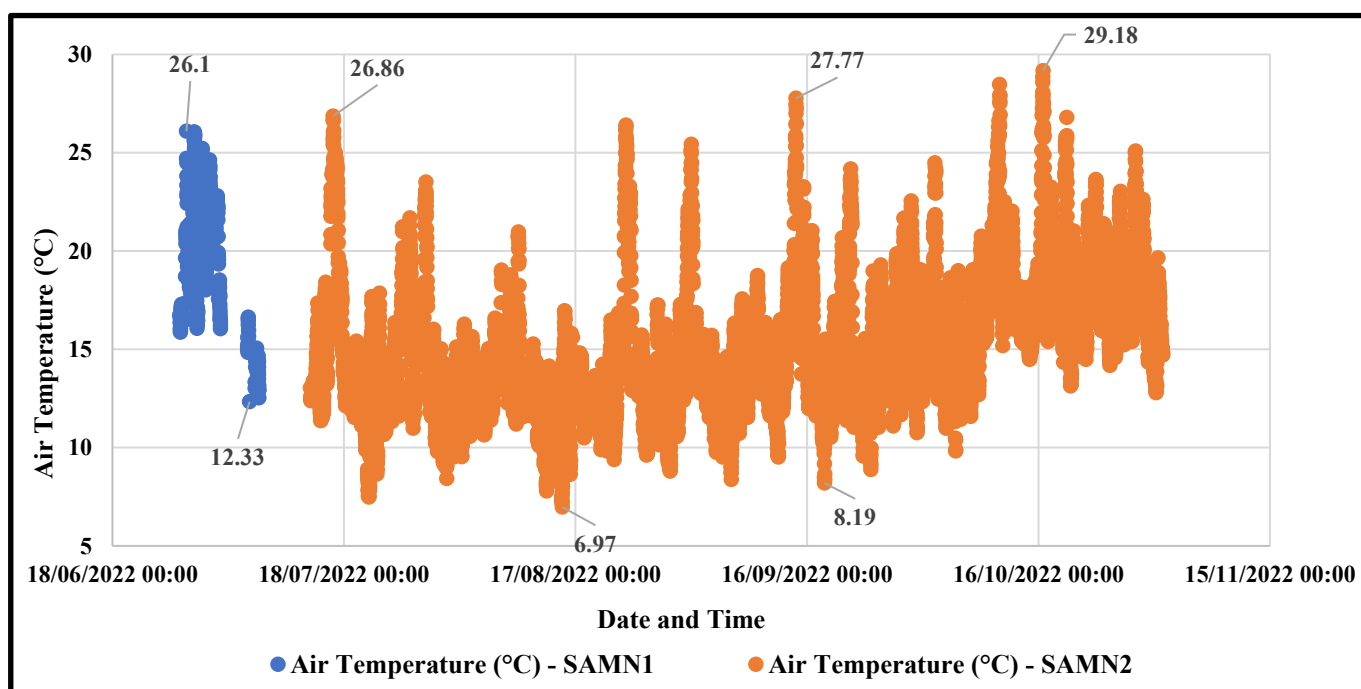


Figure 6-10: Time Series of Air Temperature Data Measured from UCT Upper Campus

In general, it was observed that the overall temperature began to decrease as the time series moved through the Winter months and increased again through the Spring months. According to

the data, the highest temperature (29.18°C) was recorded in October and the lowest temperature (6.97°C) was recorded in August. The lowest average was recorded throughout the month of August while the highest (21.17°C) was recorded throughout June. The reason for the high temperatures in June is because of the lack of reliable data due to constant disruptions in transmission due to power outages. Other than June, October had the highest monthly average (17.37°C) across the 6 months of deployment. Table 6-4 gives an overview of the monthly minimum, maximum and average temperature for the SAMN deployment. Figure 6-11 provides a representation of the daily min, max and average air temperature during the SAMN deployment. Tabulated data of the daily min, max and averages for air temperature can be found in Appendix M-3.

Table 6-4: Monthly Air Temperature (°C) data across SAMN deployment

Month	Average	Min	Max
June	21,17	15,87	26,10
July	14,42	7,48	26,86
August	13,09	6,97	26,40
September	14,42	8,19	27,77
October	17,37	9,83	29,18
November	14,87	14,71	15,03
SAMN Total	15.07	6.97	29.18

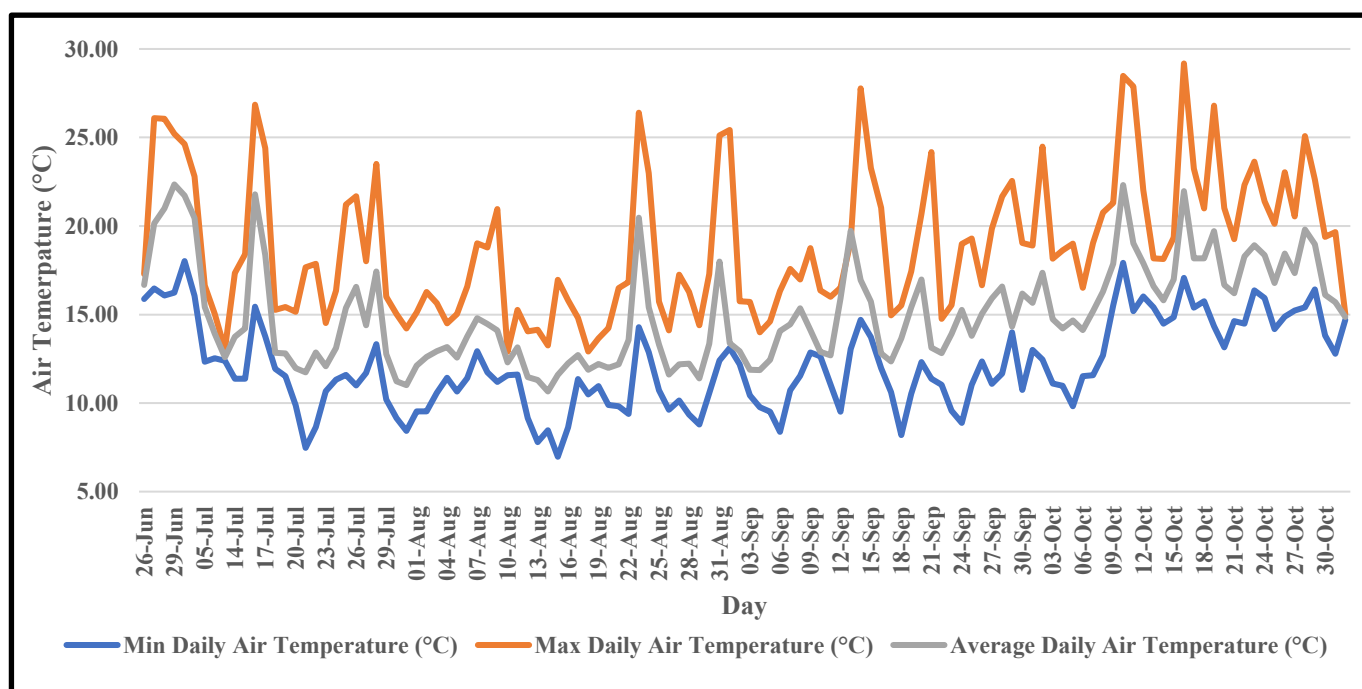


Figure 6-11: Min, Max and Average Air Temperatures Across the SAMN Deployment

The relative air humidity data produced by the sensor can be seen in Figure 6-12. The humidity data increased as the time series moved through the winter months and then decreased through the spring months. The lowest humidity recording (19.54) was in September and the highest (100) recording was in July and August.

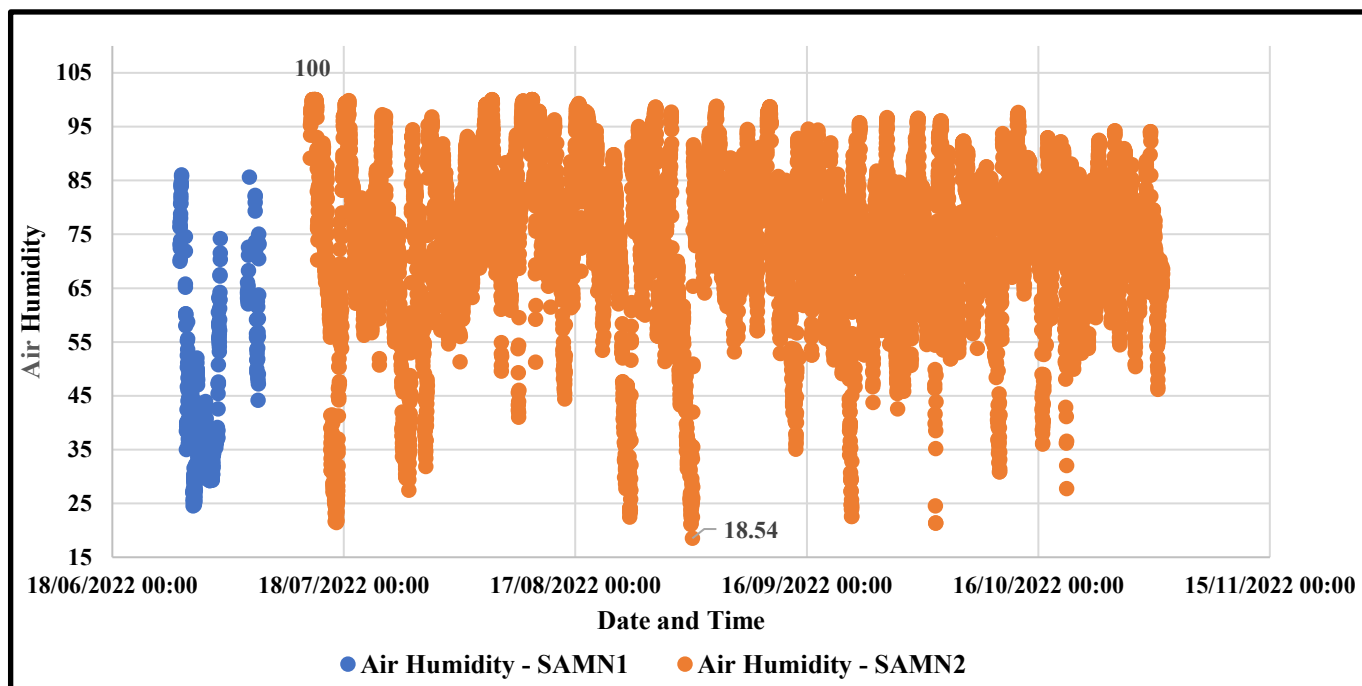


Figure 6-12: Time Series of Air Humidity Data Measured from the UCT Upper Campus Daily

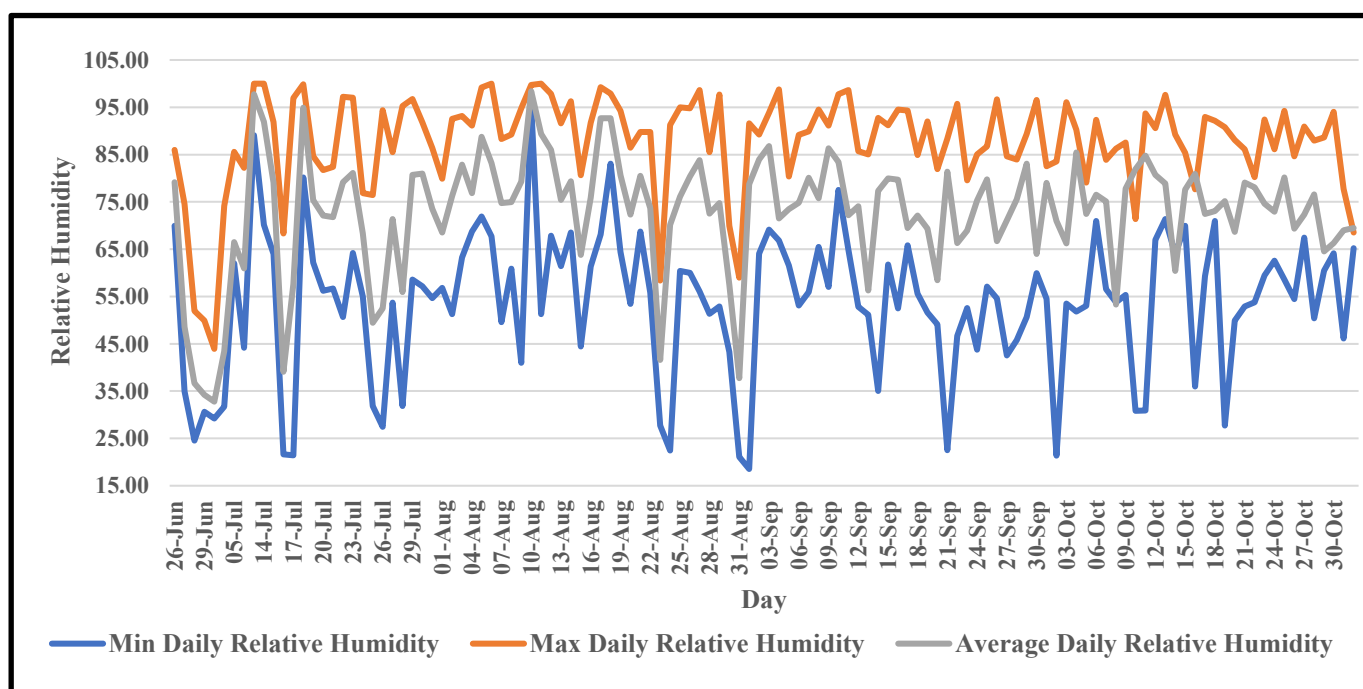


Figure 6-13: Daily Min, Max and Average Relative Humidity Across the SAMN Deployment

This indicates that, as expected, when temperature decreases, humidity increases – thus, confirming the inversely proportional relationship between the two parameters.

The average humidity across the SAMN deployment was 72.88. The lowest humidity (39.23) reading was recorded in June and the highest (100) was recorded in July and August. As with air temperature, the June readings are inconsistent and disjointed and, thus, November is taken as the lowest average (66.28). Table 6-5 gives an overview of the average, minimum and maximum relative humidity measured during SAMN deployment. Figure 6-13 provides a graphical representation of the daily min, max and averages for the relative humidity measured during the SAMN deployment. The tabulated data can be found in Appendix M-3.

Table 6-5: Average, Min and Max Values for Air Humidity Across SAMN Deployment

Month	Average	Min	Max
June	39,23	24,55	86,02
July	69,70	21,42	100,00
August	76,15	21,10	100,00
September	74,73	18,54	98,77
October	73,94	21,34	97,62
November	66,28	65,21	68,54
SAMN Total	72.88	18.54	100.00

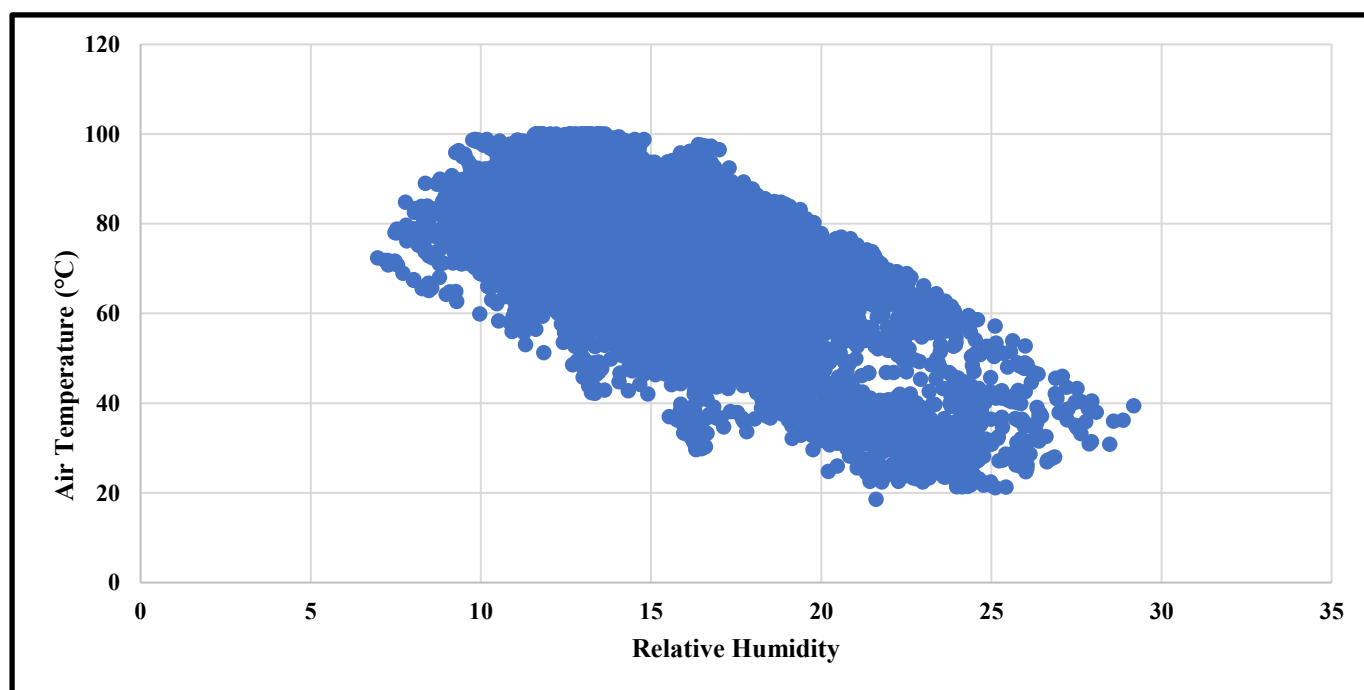


Figure 6-14: Relationship between Air Temperature (°C) and Relative Humidity Collected by the SAMN

The air temperature and relative humidity data collected during the SAMN deployment was plotted to confirm the expected relationship between the parameters. Figure 6-14 displays the inverse relationship between the two parameters and confirms the expected outputs measured by the SHT31 temperature and humidity sensor.

Figure 6-15 shows how the data is displayed on Things Speak alongside time series graphs of the data. The gauge updated in 15-minute intervals when the node successfully transmitted the measured data to Things Speak via TTN.

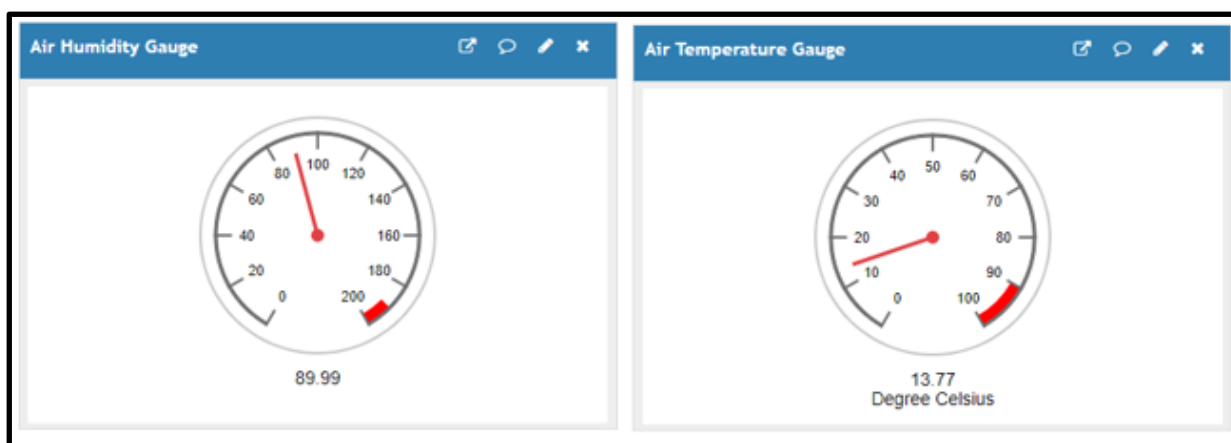


Figure 6-15: Gauge on Things Speak for Real-time Air Temperature and Humidity

6.2 Robustness Analysis

Robustness analysis assessed the system's resilience and ability to adapt to abnormal conditions during general operation as well as how the physical components cope in the environment.



Figure 6-16: Deployment 0 - System Vandalised

The initial setup of the system (referred to as deployment 0), involved one single node that measured all the required data and was housed in a concrete utility box that was placed underground on the bank of the UCT dam. After one day of deployment, the system was dug up and vandalised as seen in Figure 6-16. To prevent future vandalism to the system, a new approach was taken wherein the systems were split into two nodes and setup in a manner that made it inaccessible and, thus, prevented and deterred any theft or vandalism.

6.2.1 Robustness - Water Monitoring Node

The WMN was setup on a level gauge in the UCT dam and the node was operational for 19 days before transmission stopped due to power outages in the CoCT interrupting the system. The power outages resulted in areas around UCT experiencing three power outages a day, with two outages lasting two hours and one outage lasting four hours. This caused surrounding gateways in the area to switch off for four and two hours at a time, resulting in no available listening gateways and an interruption in the transmission of data, as seen in Figure 6-17. To address this problem, an Uninterruptible Power Supply (UPS) was purchased and setup in the NEB building on UCT's Upper Campus for IoT related projects. Once the UCT gateway was operational, the node remained connected to the TTN network despite power outages in the area.

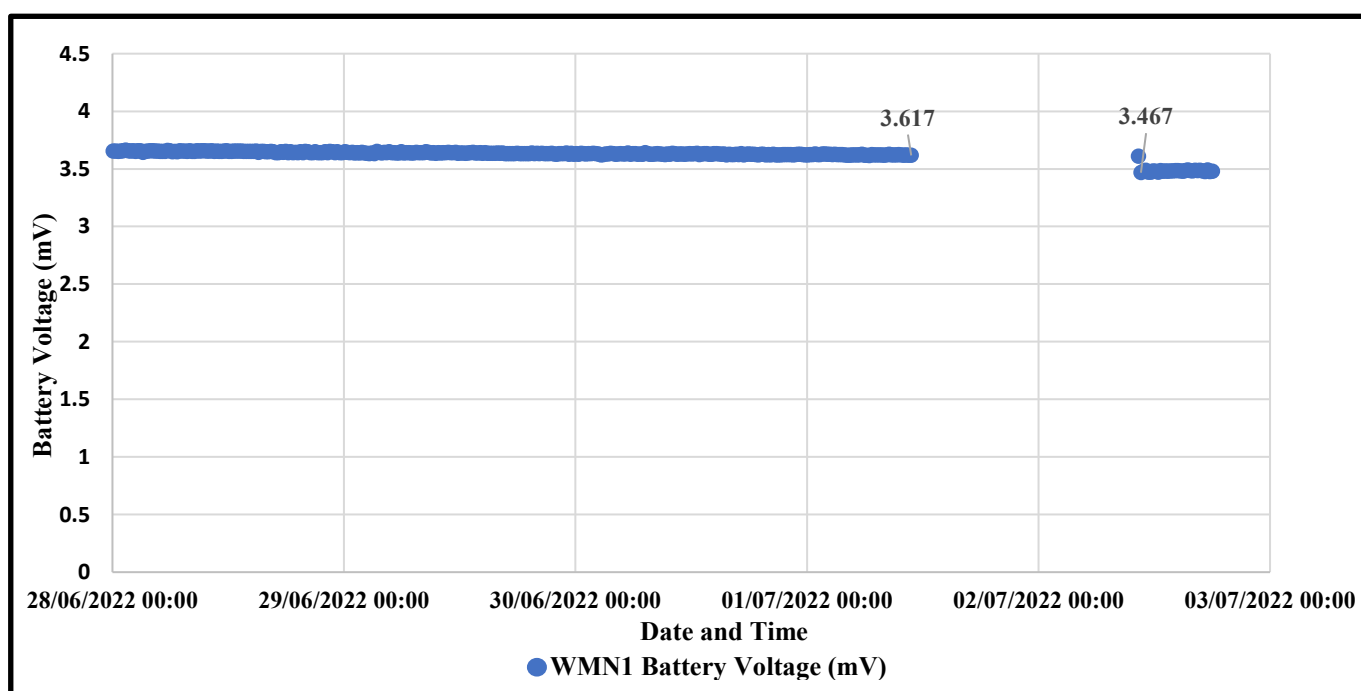


Figure 6-17: Graph showing battery voltage during power outages for the WMN

The physical components of the system were protected for the 5-month duration of the nodes' deployment and are still currently operational in terms of successfully housing and protecting the electronics of the node. The weatherproof enclosure has experienced multiple storm events and

has prevented water intrusion. The placement of the node prevented any theft or vandalism to the system due to the inaccessibility of the node to the public.

6.2.2 Robustness – Soil-Air Monitoring Node

The SAMN was setup on a tree above the parking lot near the NEB building. For the duration of the first deployment, the system also suffered connection issues in the form of periods of connection loss during power outages as seen in Figure 6-18. The gateway installed at UCT prevented these interruptions and improved connection reliability. The components were elevated above head height on a tree, and this made it difficult to access. The inaccessibility of the node successfully prevented theft and vandalism for the duration of the experiment, and the weatherproof enclosure protected the electronics from water and dust inclusion.

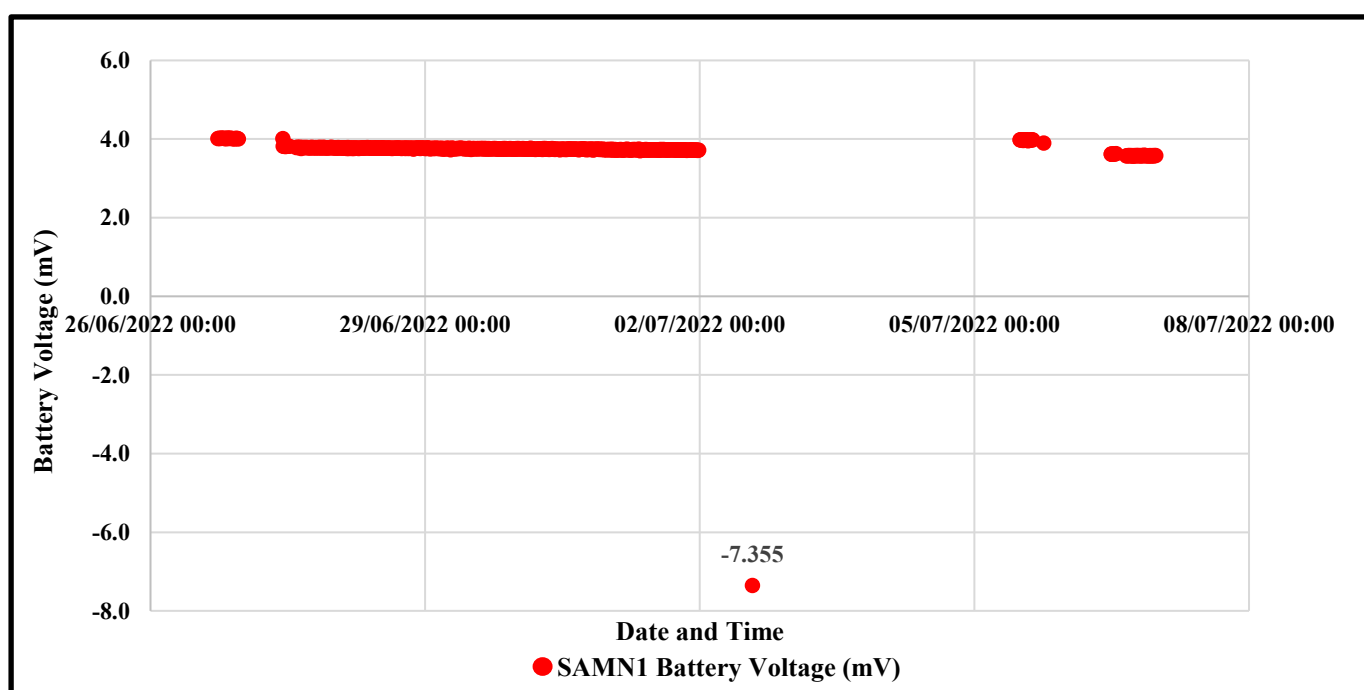


Figure 6-18: Graph showing battery voltage during power outages for SAMN

6.3 Network and Communication Analysis

By adopting the LoRa network for wireless transmission, the nodes operated in the unlicensed frequency spectrum with the frequency centring around 868 MHz. For the systems to be viable, the nodes must meet the legal duty cycle of 1% in accordance with the ICASA (2015) regulations. The packet losses during the run-time of the nodes were analysed to determine the percentage packets lost over the node's lifespan. This gave an indication of the node's reliability and efficiency in terms of communication.

6.3.1 Duty Cycle

The duty cycle of the system was determined by using an online airtime calculator for LoRaWAN developed in 2021 by a github user that goes by the alias ‘avbentem’ and can be accessed via the following link: [Airtime calculator for LoRaWAN \(avbentem.github.io\)](https://github.com/avbentem/airtime-calculator) (Arjan, 2021). The online calculator required two inputs to calculate airtime for the nodes. The first input was overhead size, and this consisted of the messages MHDR, DevAddr, FCtrl, FCnt and MIC and was estimated to be on average 13 bytes. The second input was payload size, and this describes the size of the observed or messaged data by the node in bytes. As specified in section 4.6.4, the size of the payload for both nodes were 18-bytes and the data points associated to each payload for each node can be found in Tables 6-6 and 6-7.

Table 6-6: Breakdown of water monitoring nodes payload

Data	Bytes
Message Number	2
Battery Voltage	2
Water Level	2
Water Temperature	2
Enclosure Temperature	2
Enclosure Pressure	2
Enclosure Humidity	2
Transmission time	4
Total	18

Table 6-7: Breakdown of soil & air monitoring nodes payload

Data	Bytes
Message Number	2
Battery Voltage	2
Soil Moisture	2
Soil Temperature	2
Air Temperature	2
Air Humidity	2
Enclosure Temperature	2
Transmission time	4
Total	18

Both nodes had the same total packet size, thus only one calculation was needed for the two systems. After inputting 13 and 18 for overhead size and payload size respectively, the online airtime calculator provided the results seen in Table 6-8 for nodes operating with DR between 9 and 6 and an SF between 7 and 12. Based on the results, both nodes, which operated at a DR of 3 and SF of 9, should not send more than 145 messages per hour to be within 1% duty cycle. The nodes sent data every 15 minutes, which is 4 messages per hour, falling within the 1% legal duty cycle specified by ICASA (2015).

Table 6-8: Results derived from online airtime calculator for LoRaWAN

Data rate	Airtime	1% Max Duty Cycle	Fair Access Policy
DR6 SF7 BW	36.0 ms	3.6 sec 1000 msg/hour	34.8 avg/hour 834 msg/24h
DR5 SF7 BW125	71.9 ms	7.2 sec 500 msg/hour	17.4 avg/hour 417 msg/24h
DR4 SF8 BW125	133.6 ms	13.4 sec 269 msg/hour	9.4 avg/hour 224 msg/24h
DR3 SF9 BW125	246.8 ms	24.7 sec 145 msg/hour	5.1 avg/hour 121 msg/hour
DR2 SF10 BW125	452.6 ms	45.3 sec 79 msg/hour	2.8 avg/hour 66 msg/24h
DR1 SF11 BW125	905.2 ms	90.5 sec 39 msg/hour	1.4 avg/hour 33 msg/24hr
DR0 SF12 BW125	1810.4 ms	181.0 sec 19 msg/hour	0.7 avg/hour 16 msg/24hr

6.3.2 Percentage Packets Lost

To determine the efficiency and reliability of the wireless communication of the systems, the percentage packets lost for each node was determined. Both nodes had two iterations of deployment with the first iteration using nearby gateways in the area and the second iteration using a private gateway setup on UCT's Upper Campus for IoT related projects. The percentage packets lost were calculated for the entire run-time of the experiment for the two nodes using Equation 5.1 and the results for the WMN and SAMN can be seen in Tables 6-9 and 6-10, respectively.

Table 6-9: Results of Percentage Packets Lost for WMN

Iteration	Successful Transmissions	Message Number	% Packets sent	% Packets lost
WMN1	1887	1999	94.40%	5.60%
WMN2	2162	2178	99.30%	0.70%
Total	4049	4177	96.90%	3.10%

The WMN had a total packet loss of 3.10%, with 4049 out of the 4177 measurements taken being successfully sent to TTN and Things Speak. When looking at the two iterations of deployment, deployment one had a significantly higher percentage packet loss than deployment two with a difference of 4.90% between WMN1 and WMN2.

Table 6-10: Results of Percentage Packets Lost for SAMN

Iteration	Successful Transmissions	Message Number	% Packets sent	% Packets lost
SAMN1	498	502	99.20%	0.80%
SAMN2	9720	10371	93.72%	6.28%
Total	10366	10873	93.98%	6.02%

The SAMN had a total packet loss of 6.02%, with 10366 out of the 10873 measurements taken being successfully transmitted to TTN and Things Speak. Comparison between the two deployments show that there was a 5.48% packet loss difference between the two iterations with SAMN2 having a 6.28% packet loss while transmitting 9720 messages.

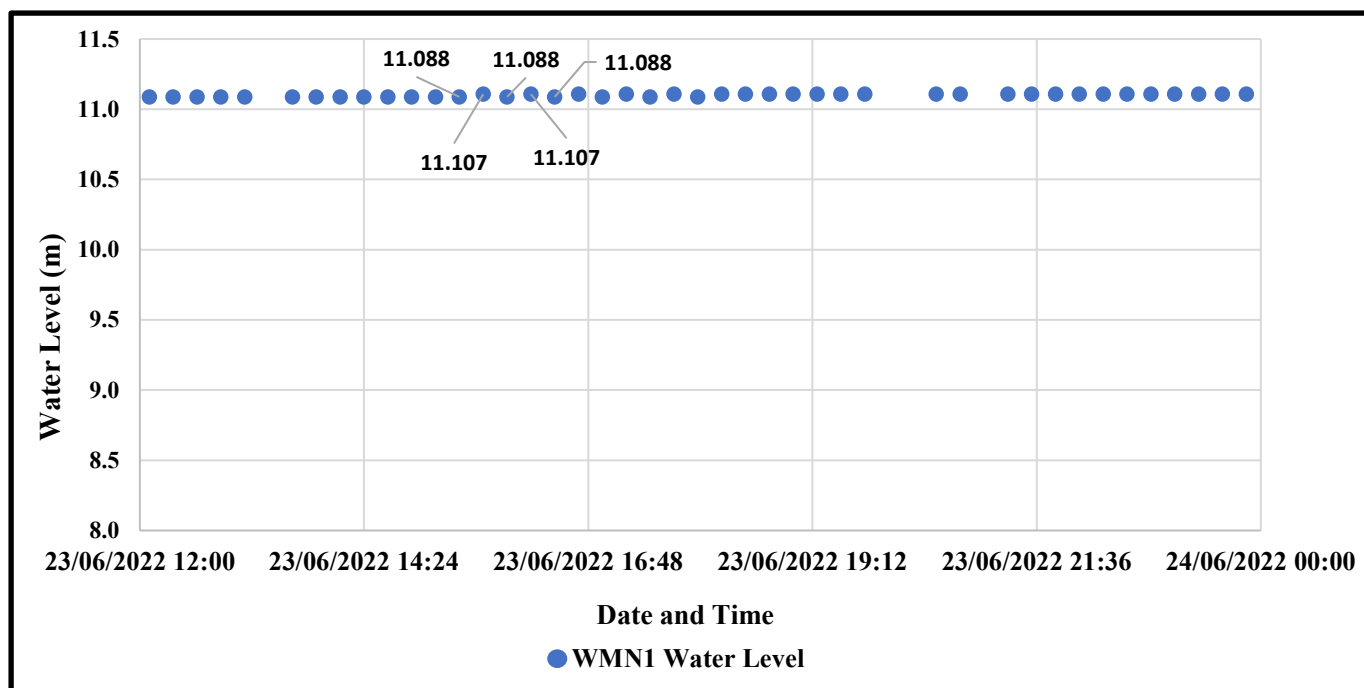
6.4 Load Testing Analysis

To determine flaws in the system, the output data was analysed and irregularities in the data were investigated. The data output from the nodes were investigated separately, with flaws/irregularities being found in both nodes.

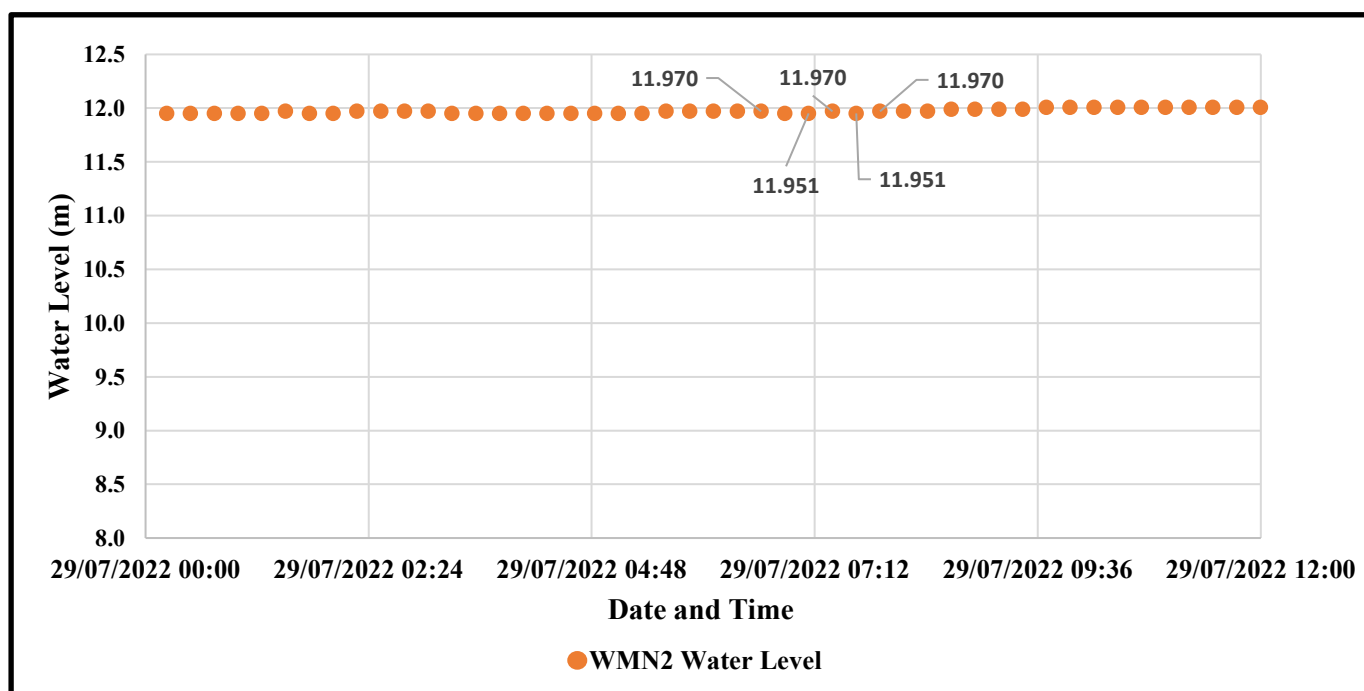
6.4.1 Load Testing Analysis – Water Monitoring Node

During the deployment of the water monitoring node, irregularities in the data were observed when there was a rise in water level. Figure 6-19a and Figure 6-19b show fluctuations in water level for WMN1 and WMN2, respectively. These fluctuations occurred during an increase in water level due to storm events and oscillated between two measurements of water level until the readings began to stabilise by outputting the higher value for water level. The water level

increased in increments of 0.016 m and, thus, changes were only visible when this threshold was passed.



(a)



(b)

Figure 6-19: Irregularities in Dam Level Readings due to Changes in Water Level

These irregularities present an issue with the accuracy and precision of the water level sensor during periods of measuring when the water level is changing.

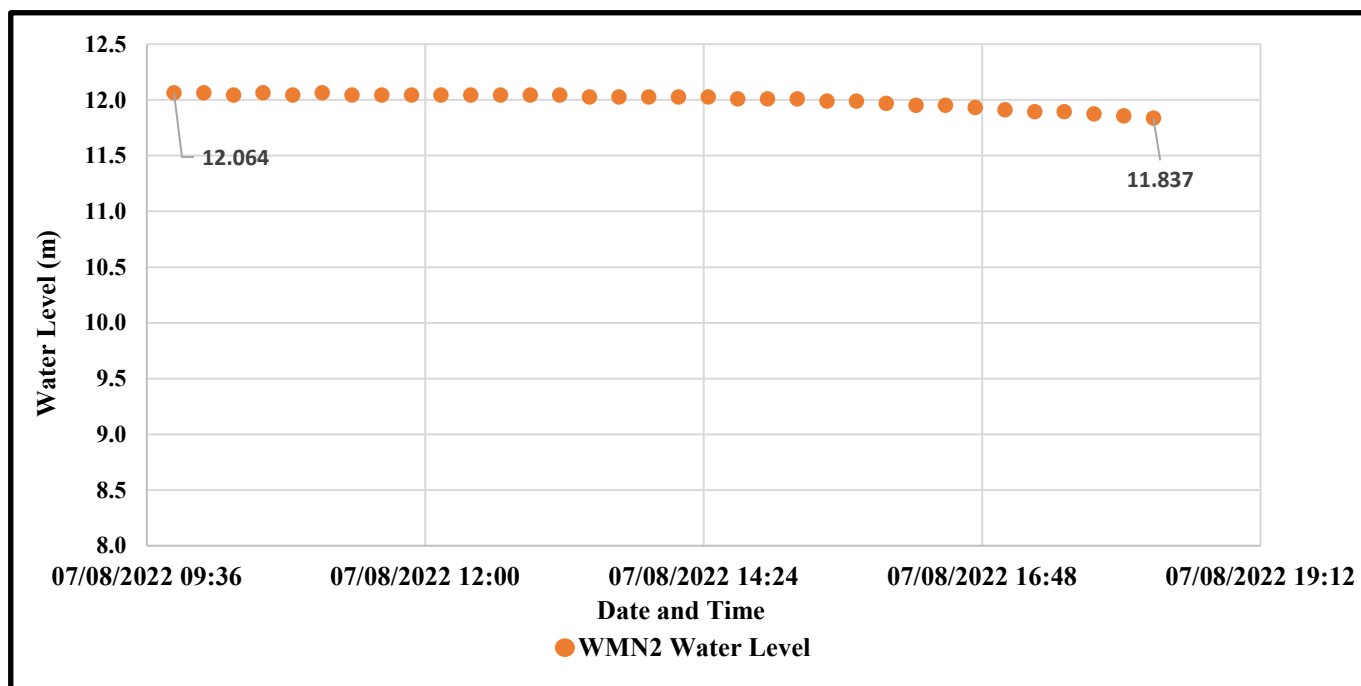


Figure 6-20: Irregularities in Dam Level Readings due to Battery Life

Another flaw found in the water level data occurred when the battery life of the node was about to deplete. Resultantly, the sensor began to output irregular water level data which showed the water level reducing over a roughly 11-hour period, as seen in Figure 6-20. Upon observation of the dam level, the water level had not changed according to the data outputted by the water level sensor.

During the deployment of WMN1, irregularities were found in the data measured by the DS18B20 temperature probe. Over the course of the first deployment, the sensor outputted -0.05 for water temperature three times, as seen from a snippet of the water temperature data for WMN1 in Figure 6-21. In comparison to WMN2, the water temperature outputted logical and regular data points throughout the second iteration as seen in Figure 6-22.

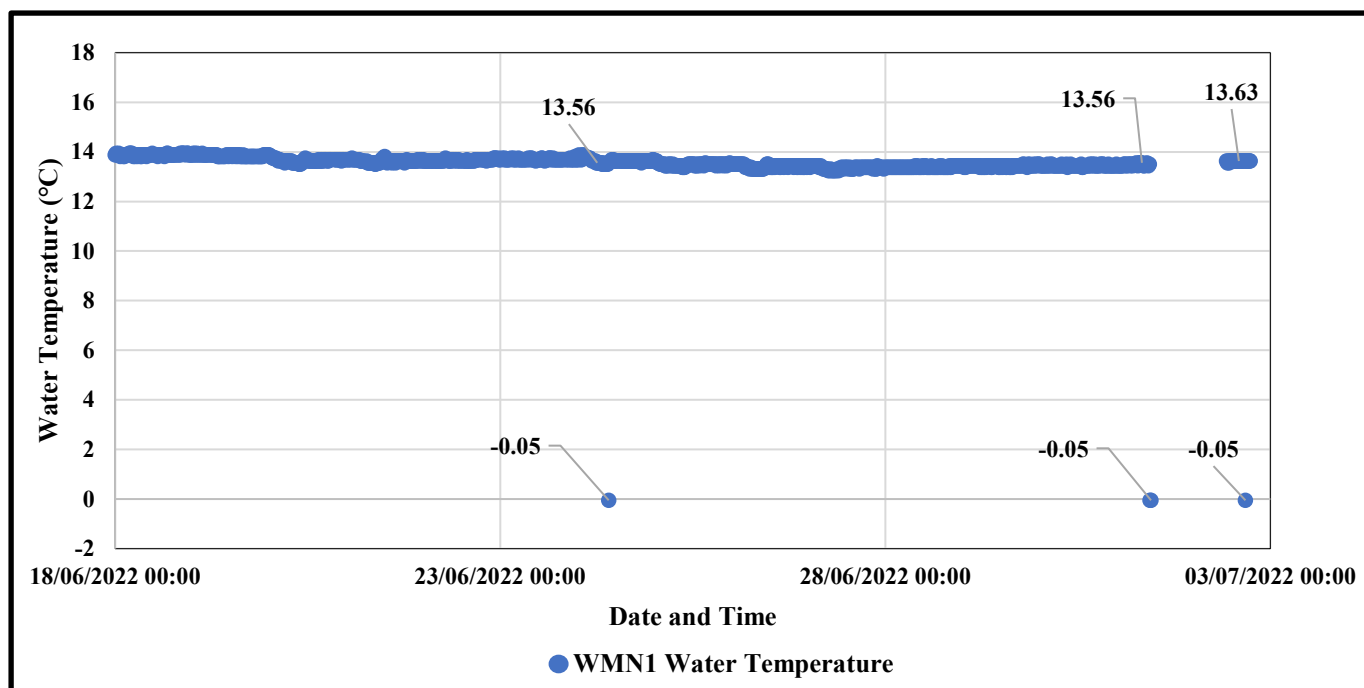


Figure 6-21: Irregularities in Water Temperature Readings for WMN1

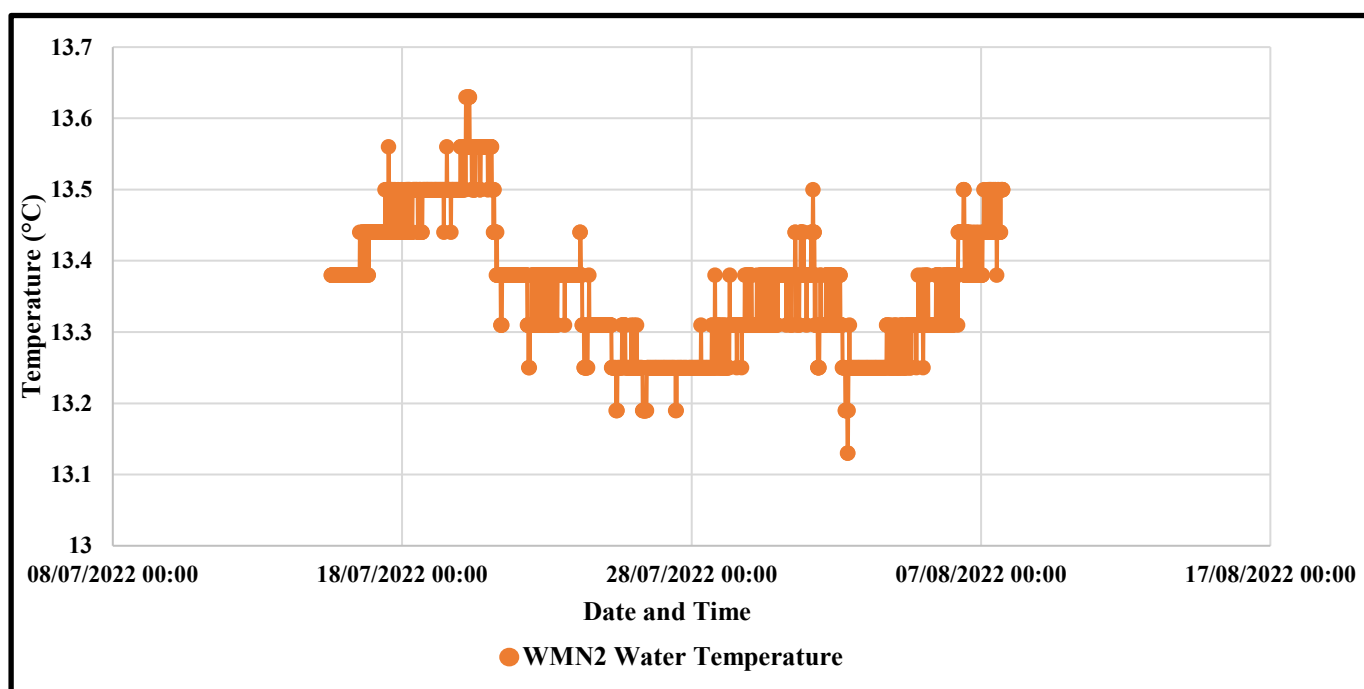
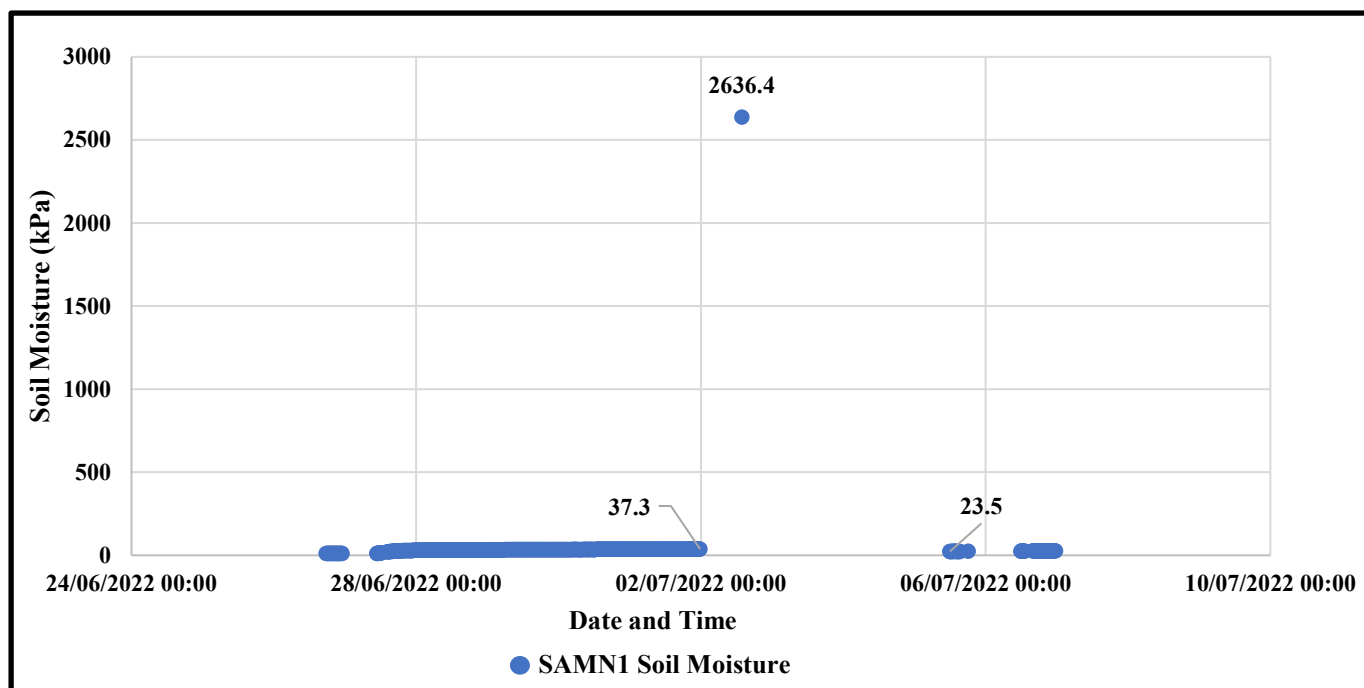


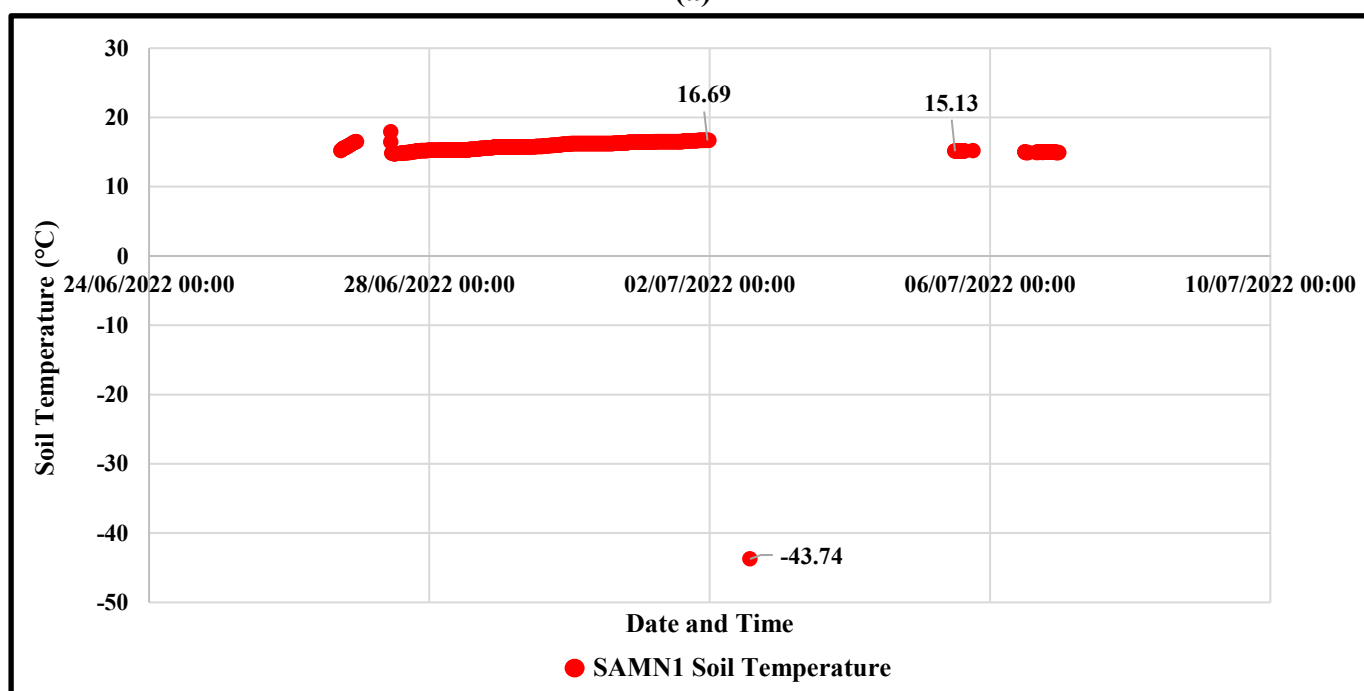
Figure 6-22: Water temperature Data obtained during WMN2 with no Flaws or Irregularities

6.4.2 Load Testing Analysis – Soil & Air Monitoring Node

When analysing flaws and irregularities for the SAMN node, noticeable irregularities and flaws were found only for one reading during SAMN1. The causes of these flaws in the data are unknown, but the results can be seen in Figures 6-23a, b, c, and d.

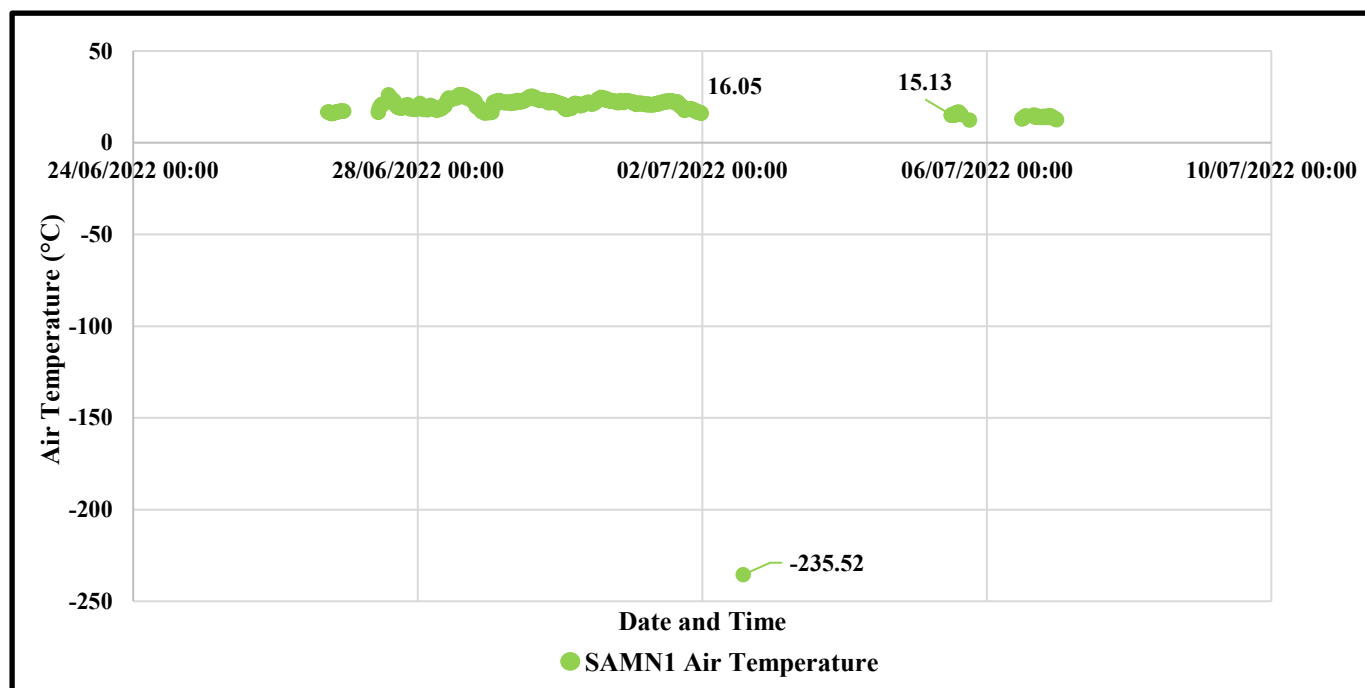


(a)

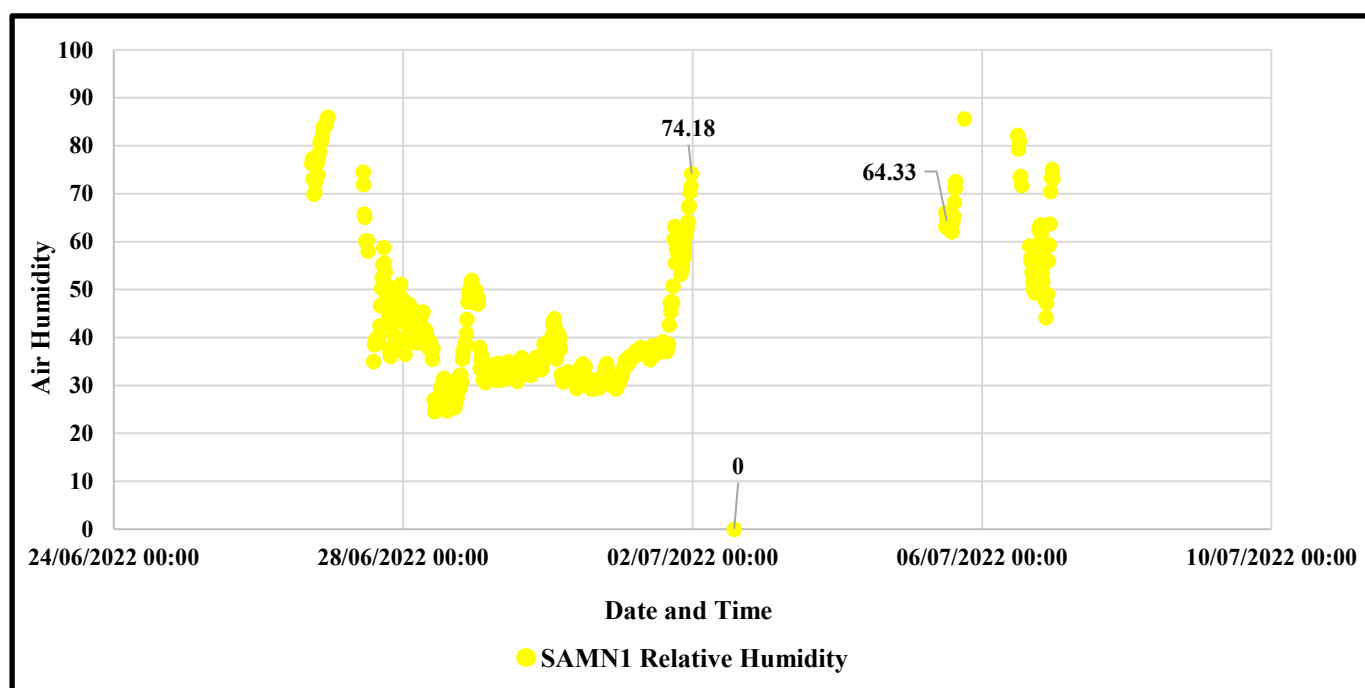


(b)

Figure 6-23: Unknown irregularities in SAMN1 data



(c)



(d)

Figure 6 23: Unknown irregularities in SAMN1 data

6.5 Power Consumption Analysis

Power consumption analysis assessed the performance of the battery and the power consumption of the two nodes. The data was then compared to the theoretical calculations done before deployment as well as comparisons made between the different iterations of deployment for the WMN and SAMN.

6.5.1 Power Consumption Analysis – Water Monitoring Node

Figure 6-24 shows the battery voltage of WMN1 and WMN2. During WMN1 the battery voltage monitoring was interrupted due to power outages and a projected battery life was calculated. When comparing the node life of the two deployments, there was a one-day difference with WMN2 lasting 23 days and WMN1 lasting 22 days. Although the hardware and software of the devices were identical across the two iterations, the projected battery life of WMN1 and the actual battery life of WMN2 differed significantly.

To calculate the projected battery voltage, an equation for the linear battery voltage decline was developed for WMN1. According to WMN2, once the battery voltage dropped below 3.43 mV, the battery voltage decline became unsteady and exponential and, thus, the projected battery life of the node was determined using 3.43 mV as the minimum voltage.

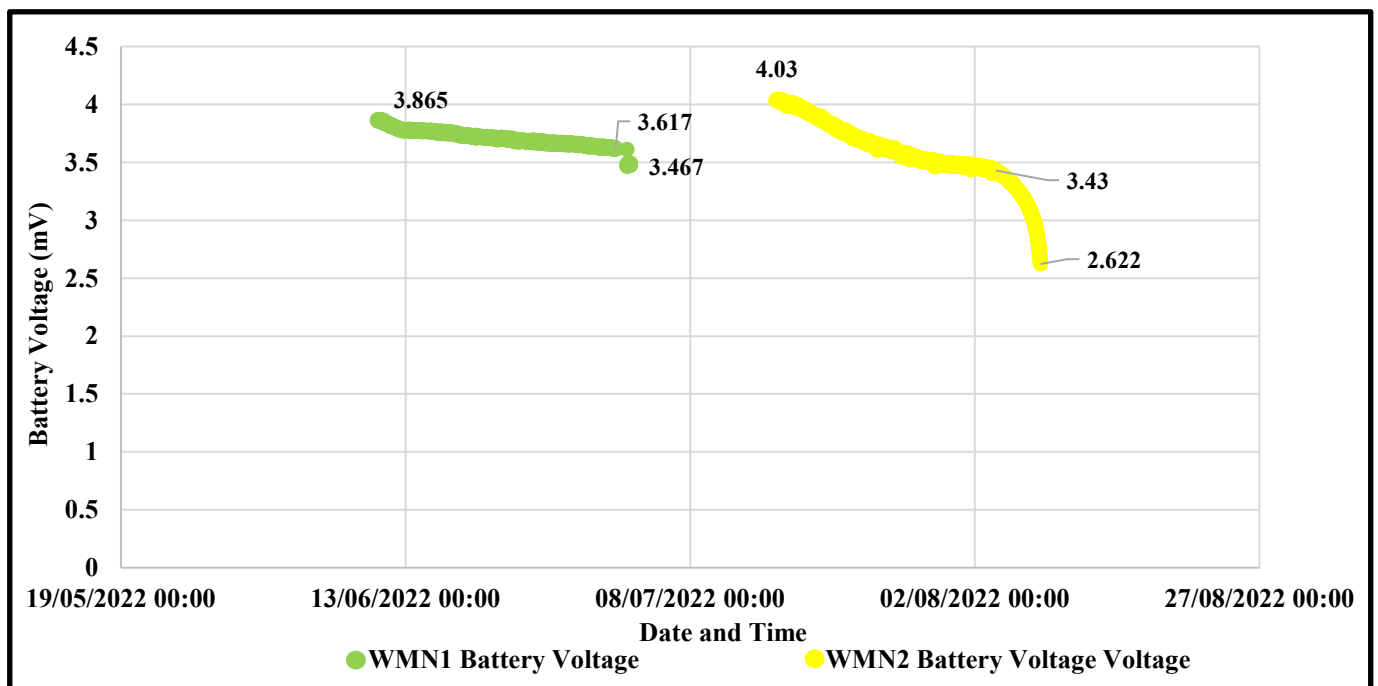


Figure 6-24: Time Series of Battery Voltage for Iteration 1 and 2 of the WMN

Equation 6.1 was developed for projected battery life based on the relationship between battery voltage decline and No. of readings for WMN1 as seen in Figure 6-25:

$$\text{Projected WMN Battery Life} = \frac{-\text{Min Battery Voltage} + 3.81}{(0.00011 \times \text{readings per day})} \quad (6.1)$$

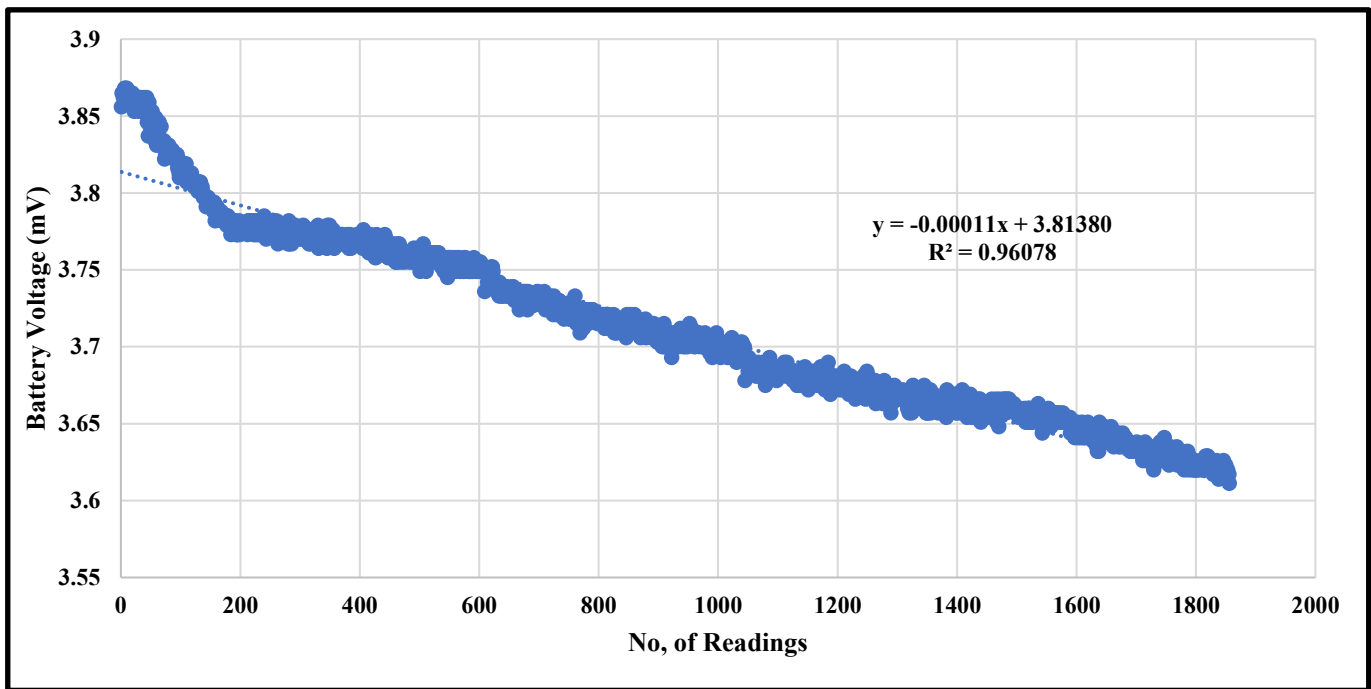


Figure 6-25: Node Battery Life vs Number of Readings Produced by WMN1

Using Equation 6.1, inputting the minimum battery voltage of 3.43 mV and the readings per day at 96, the projected battery life for WMN1 was calculated to be $35.98 \approx 36$ days. The projected battery life for WMN1 was 13 days longer than the actual battery life of WMN2 which was 23 days. The comparison between the theoretical battery life calculated in section 4.5.1 and the actual battery life observed during the WMN2 indicated that the WMN2 under performed in terms of battery life by nine days while the projected battery life of WMN1 was on course to outperform the theoretical battery life by two days.

6.5.2 Power Consumption Analysis – Soil & Air Monitoring Node

The interferences during SAMN1 made it difficult to perform any type of analysis on the power consumption during this period. Due to constant losses of connection throughout the deployment and changing of battery during the deployment, there was no possibility of obtaining a projected battery life for SAMN1. SAMN2 deployment was stable, and a linear decline of the battery can be seen in Figure 6-26. The data for SAMN2 was obtained on November 1st and this was 110 days into the deployment of SAMN2. The node was still active at the end of the experimental

testing, thus obtaining the projected battery life involved a similar process carried out in section 6.5.1.

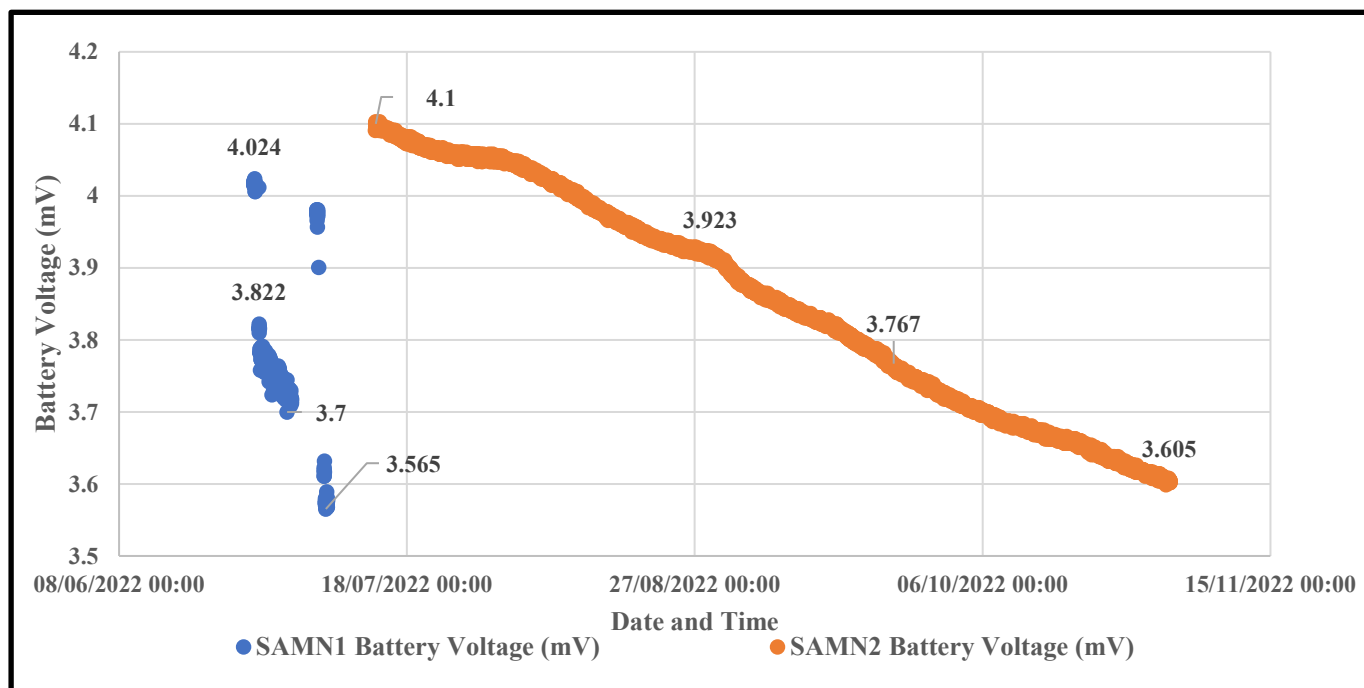


Figure 6-26: Time Series of Battery Voltage for Iteration 1 and 2 of the SAMN

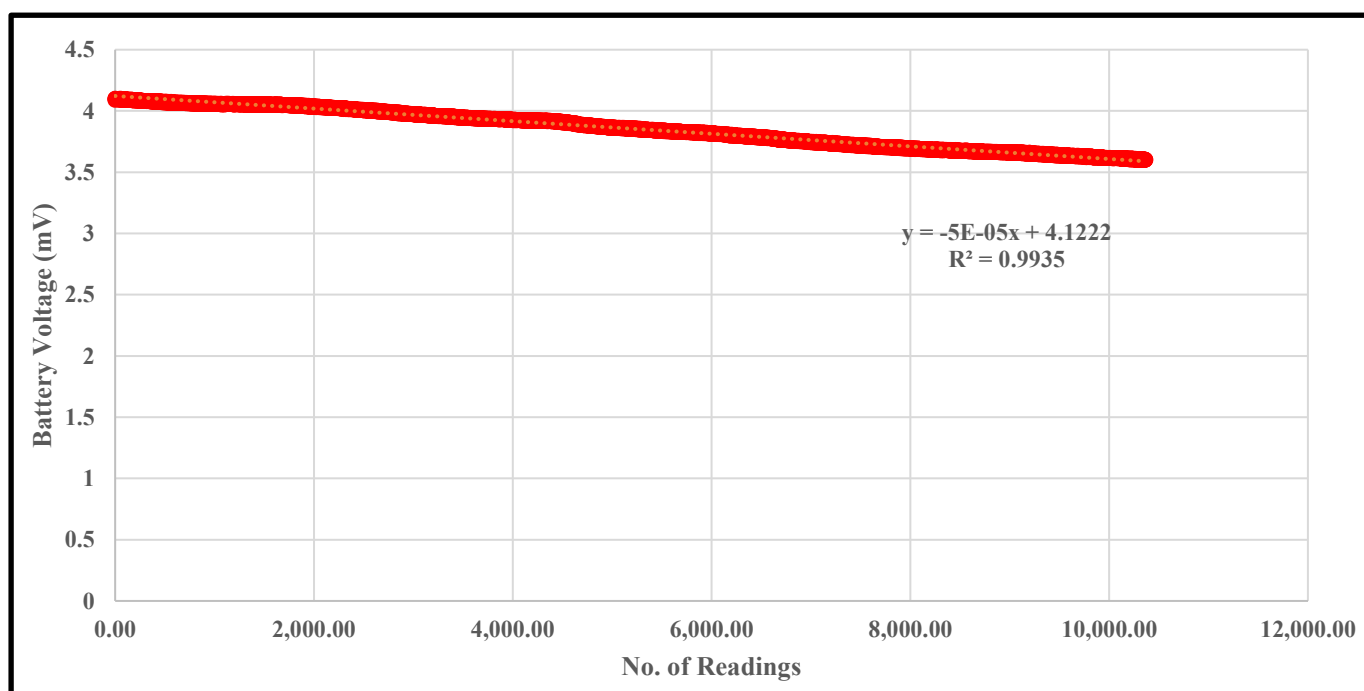


Figure 6-27: Node Battery Life vs Number of Readings Produced by SAMN2

Equation 6.2 is used to calculate the projected battery life of the node according to the straight-line equation obtained from the relationship between battery voltage and no. of readings for SAMN2, as seen in Figure 6-27:

$$\text{Projected SAMN Battery Life} = \frac{-\text{Min Battery Voltage} + 4.1}{(0.000038 \times \text{readings per day})} \quad (6.2)$$

Inputting 3.43 mV and 96 for minimum voltage and readings per day respectively, Equation 6.2 gave a projected battery life of $144.21 \approx 144$ days. The projected battery performance indicated a life span of $6.034 \approx 6$ months if one considers the average days per month throughout the year to be 30.437 days. The power consumption considerations carried out in section 4.5.1 predicted that the system would need a 2581 mAh battery to power the system for 3 months. Thus, the projected battery life of roughly 6 months for SAMN2 while using a 5000 mAh battery is comparable to that of the theoretical calculations performed before the deployment.

7. Validation of Data and Discussion

This section highlights the validation of the data produced by the systems and presents a discussion on the data output, reliability and performance of the hydrological monitoring nodes based on the analysis carried out in Section 6. The analysis of the data produced by the two systems involved highlighting the validity, accuracy, and precision. Visual representations of the SAWS data used to check these aspects can be found in Appendix N-1. The systems' ability to produce the data visually to the end-user and in user-friendly manner was discussed. The reliability in terms of the systems' robustness and communication efficiency to transmit the measured data is highlighted and discussed in terms of the systems' viability in successfully transmitting hydrological data. Lastly, the performance of the system is discussed based on the results and analysis performed on the load testing and power consumption.

7.1 Validation of Data

7.1.1 Water Level

Two methods of validation were used to check the accuracy and the expected behaviour of the water level sensor. The accuracy was checked using both pre-deployment accuracy checks as seen in section 5.4.3 and by checking the dam level gauge multiple times throughout the field deployment.

Table 7-1: Spot Checks of the Water Level in the UCT Dam.

Date and Time	Level Gauge Reading	Sensor Readings	Difference
10/06/2022 14:15	9.200	9.150	0.050
15/06/2022 13:15	10.700	10.727	-0.027
19/06/2022 13:10	10.900	10.879	0.021
24/06/2022 14:40	11.100	11.107	-0.007
29/06/2022 11:30	11.100	11.107	-0.007
02/07/2022 17:20	11.200	11.240	-0.04
15/07/2022 13:30	11.800	11.799	0.001
20/07/2022 10:45	11.900	11.894	0.006
27/07/2022 11:30	12.000	11.951	0.049
03/08/2022 17:00	12.100	12.045	0.055
07/08/2022 16:30	12.100	11.932	0.168
08/08/2022 10:00	12.100	11.837	0.263

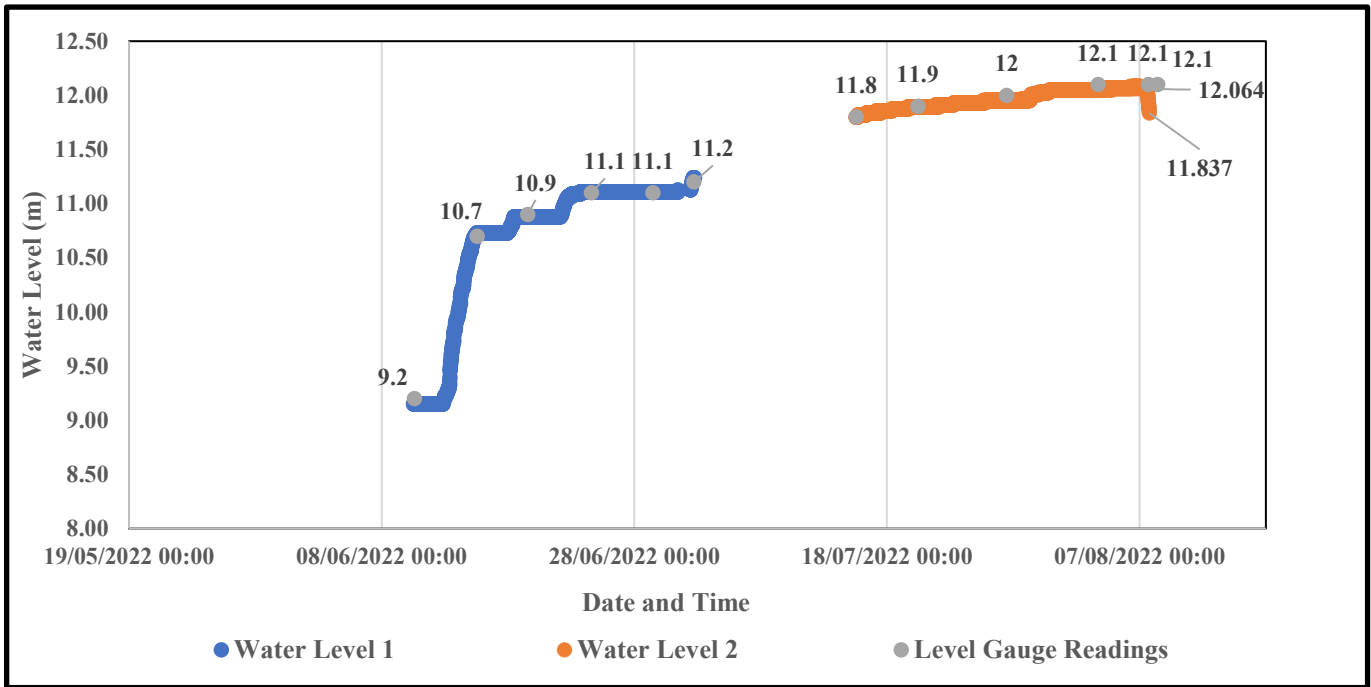


Figure 7-1: Comparison of WMN Water Level data with Water Level Spot Checks

Spot checks of the gauge in the dam were undertaken once or twice a week during the deployment of both WMN1 and WMN2 to compare the difference of the level gauge reading in the dam to that of the sensor readings. The results of the level gauge readings compared to the most recent sensor reading at the time of the check for WMN1 and WMN2 can be found in Table 7-1. The comparison between the level gauge and the water level sensor readings can be seen in Figure 7-1. Based on the comparison and using Equation 5.1, the water level sensor had an accuracy of 0.0005% when compared with the manual level gauge readings.

Further checks were carried out using data collected by the SAWS weather station in Kirstenbosch. The data collected by the Kirstenbosch rain gauge was used to check the dam's increase in water level. The daily rainfall data was plotted alongside the dam water level as seen in Figure 7-2 between the 11/06/2022 and the 22/06/2022 - this provided evidence of an increase in dam level during and after major daily rainfall events. This is evident on the 13/06/2022 where there was a cumulative daily rainfall of 131.4 mm, and this corresponded with an increase in dam level from 9.150 on the 11/06/2022 to 10.727 on the 16/06/2022. This indicated a trend of rising dam levels after rainfall events which is an indication that the water level sensor is operating as expected. The pair plot representing the full deployment of the WMN can be seen in Appendix N-2.

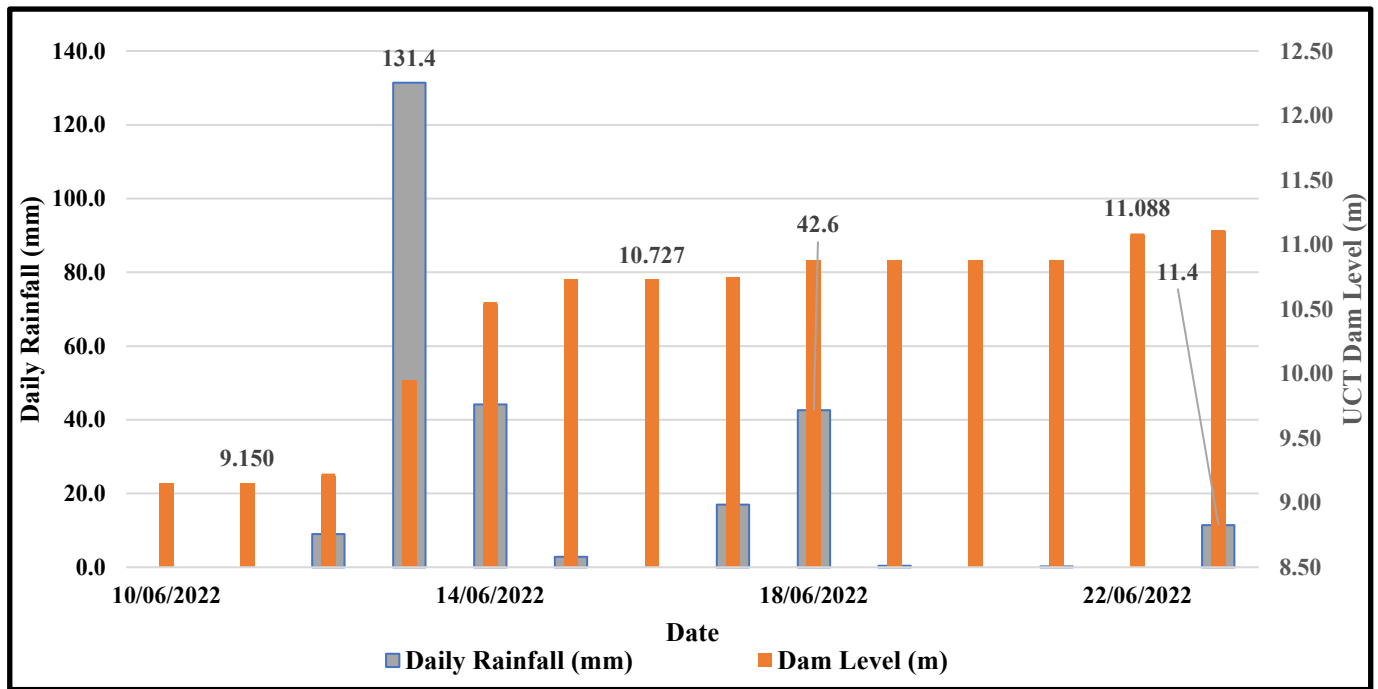


Figure 7-2: Comparison between the Dam Water Level and Kirstenbosch SAWS Weather Station

7.1.2 Water Temperature

Validation of the water temperature data produced during field deployment was not possible due to the position of the water temperature sensor in the UCT dam. Due to this, the temperature was validated by using pre-deployment accuracy testing. The data produced by the sensor was assumed to have an accuracy and precision error of ± 0.026 °C and ± 0.012 °C, respectively, as determined in section 5.4.3.

7.1.3 Soil Moisture

The validation of the soil moisture data involved using both data from the SAWS Kirstenbosch weather station and using pre-deployment accuracy tests.

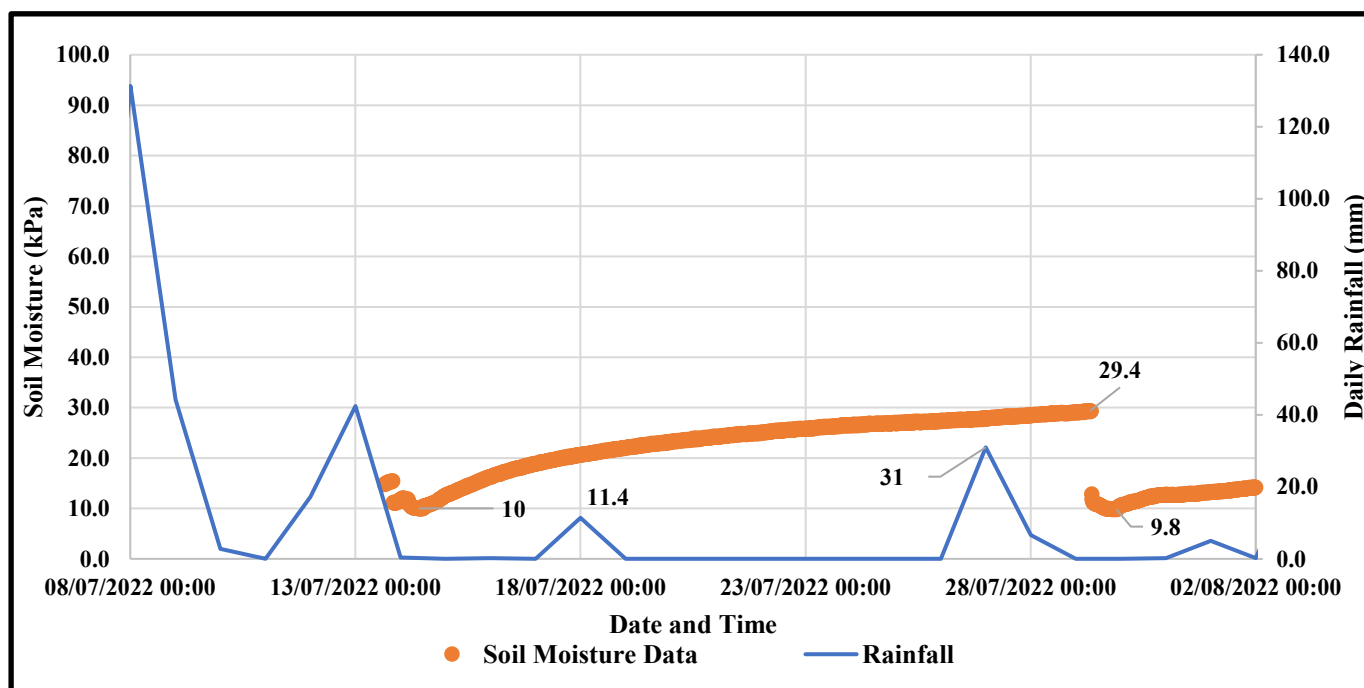


Figure 7-3: Comparison between the Soil Moisture and Kirstenbosch SAWS Weather Station

Figure 7-3 provides a pair plot of the daily rainfall of the weather station and the soil moisture data produced by the SAMN between 08/07/2022 – 02/08/2022. Comparing the two data sets highlighted the consistent relationship between a rainfall event and the soil moisture recordings. During drier periods, soil moisture readings increased until there was a rainfall event which caused a drop in the soil moisture that was dependent on the severity of the rainfall event. This validated the sensor's expected behaviour during rainfall events. The pair plot representing the full deployment of the SAMN can be seen in Appendix N-3.

The accuracy of the soil temperature sensor used to determine soil moisture was validated using pre-deployment accuracy and precision tests. As seen in section 5.4.3 the accuracy and precision error of the sensor was ± 0.3 °C and ± 0.24 °C, respectively. It was assumed that the sensor accuracy and precision were at pre-deployment levels during the field deployment.

7.1.4 Air Temperature & Humidity

The validation of the air temperature and relative humidity data was assessed using air temperature and relative humidity data obtained from the Kirstenbosch SAWS station. Figures 7-4 and 7-5 display the comparison of air temperature and relative humidity readings of the SAMN and the Kirstenbosch SAWS station, respectively. Figure 7-4 shows that, on average, the temperatures collected by the SAMN was comparable to the SAWS station while a significant

drop-off can be seen when temperatures drop below 10 °C. The difference in temperatures can be attributed to the difference in elevation of the two data collecting systems, with the SAWS station having an elevation of 156m and the SAMN 88 m. Using Equation 5.1, the average accuracy error of the air temperature measured by the SAMN in comparison to the SAWS station was 3%. This was compared using hourly data points collected by the node and the station on the hour every hour throughout the deployment of the SAMN.

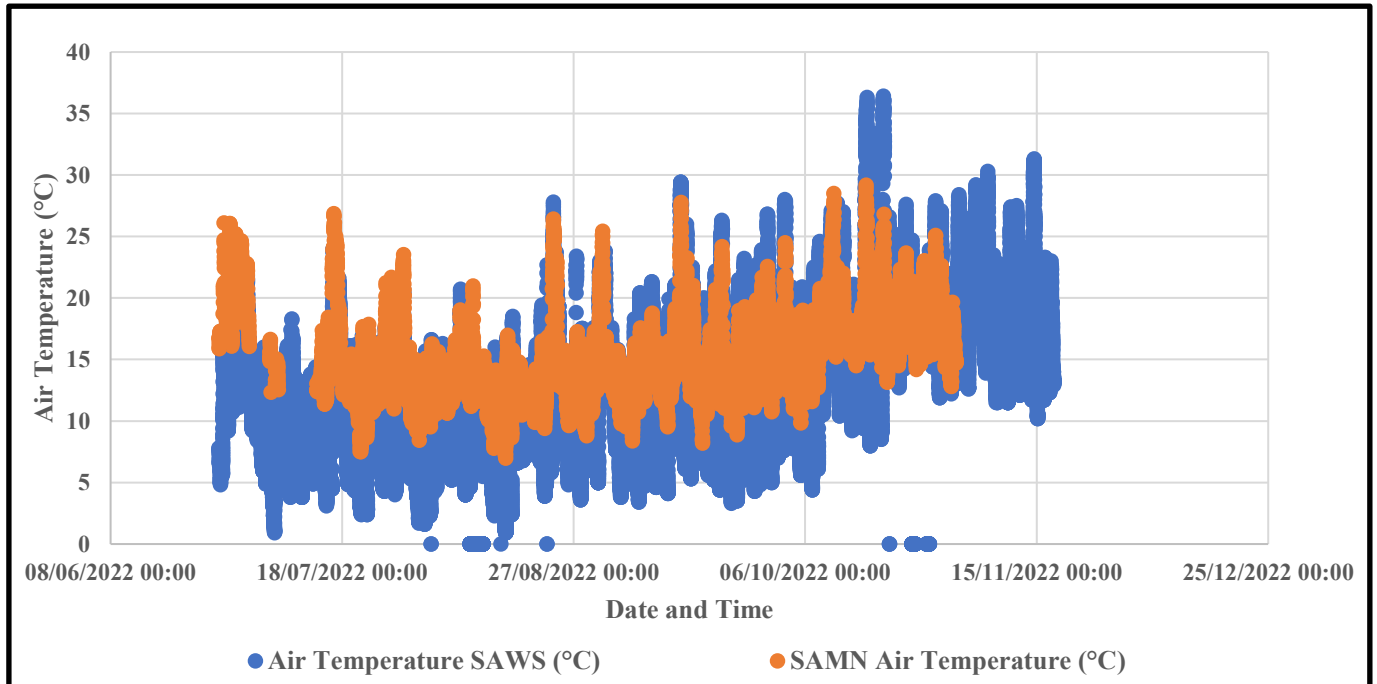


Figure 7-4: Comparison of Air Temperature (°C) Data Collected by the SAWS Station and the SAMN

When comparing the relative humidity readings based on Figure 7-5, the relative humidity data produced by the SAMN is much closer to the relative humidity data produced by the SAWS station in comparison to the air temperature data. However, when the data was reduced to hourly readings and the accuracy measured using Equation 5.1, it was found that the average accuracy error of the relative humidity measured by the SAMN in comparison to the SAWS station is 11%. This was 8% higher than the air temperature average accuracy error.

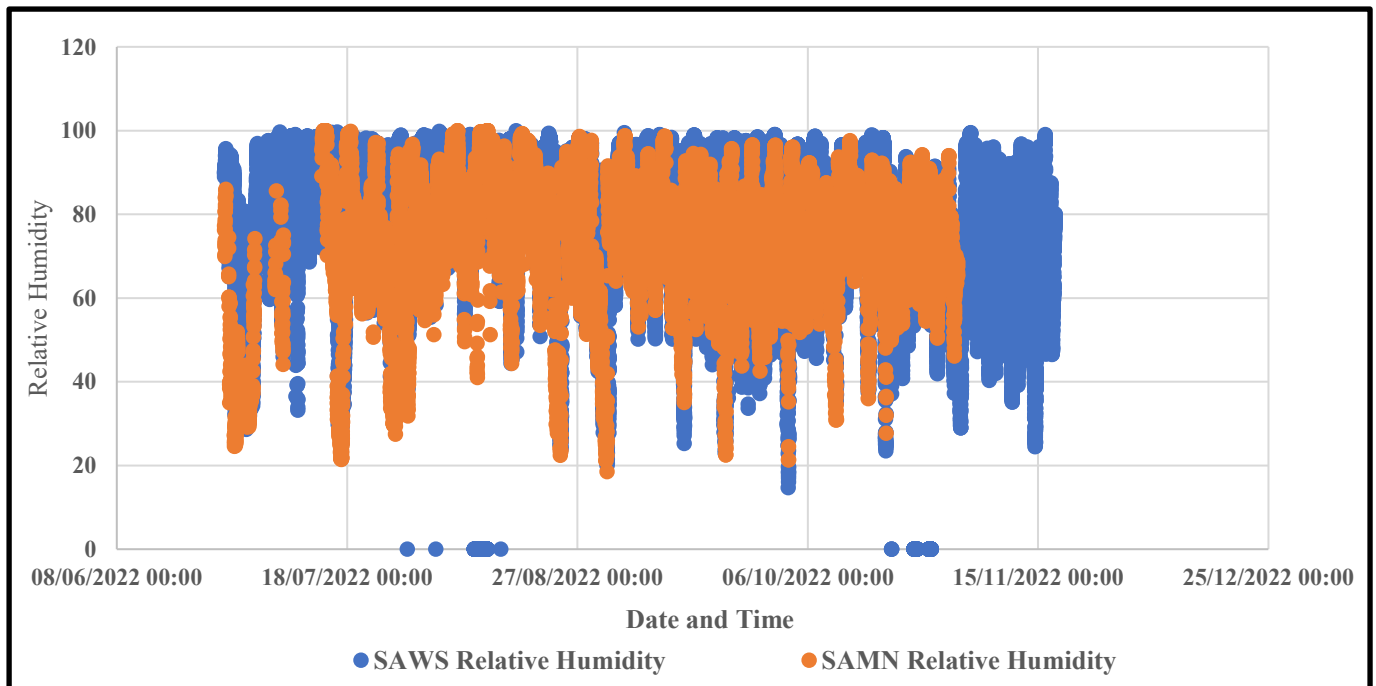


Figure 7-5: Comparison of Relative Humidity Data Collected by the SAWS Station and the SAMN

7.2 Data Output of the WMN and SAMN

In this section, the data output of the two hydrological monitoring nodes are discussed based on the accuracy, precision, and ability to present data visually in a user-friendly manner.

7.2.1 WMN

The WMN successfully transmitted 4049 messages during a period of 45 days. The messages contained data for water level and water temperature that was presented visually using the Things Speak platform and Excel. The data produced by the water level sensor was compared with spot checks done using the level gauge. Comparing the water level sensor data to the spot checks, the only major discrepancies in the data were towards the end of WMN2 where the level readings of the sensor became unstable due to the node running out of battery. Before this point, the largest difference between the level gauge reading and the sensor reading was 0.055m while the average accuracy error was 0.0005%. The water level data was also compared with rain data captured by the SAWS Kirstenbosch weather station to verify rising dam level during rainfall events. The data produced by the water level sensor and the rain gauge data had corresponding trends with the major rainfall events during the 13th and 14th of June significantly increasing the water level during this period. At the start of the deployment, the water temperature sensor produced fairly high temperatures above 15°C. There was a significant drop in water temperatures after the significant increase in dam level due to the major rainfall events that occurred on the 13th and 14th of June. The temperature data collected by the WMN followed the expected trend of

decreasing in temperature due to an increase in water level above the sensor. The increased water levels above the temperature sensor prevented little heat transfer from solar radiation to where the sensor was located. Based on the data output, water temperature followed the natural trend of decreasing in temperature with colder months and increasing during warmer months. Regular spot checks of water temperature were not possible for this research due to the position of the sensor, which was inaccessible, unless in possession of a raft. Therefore, the accuracy and precision of the sensor was determined as per the results in Section 5.4.3.

7.2.2 SAMN

During the deployment of the SAMN, the nodes successfully transmitted 10366 messages during a period of 121 days. The messages contained measured soil moisture, soil temperature, air temperature and relative humidity data that was visually presented and analysed to the end-user through the Things Speak platform and Excel. The soil moisture data followed the expected trend of increasing during dry periods and decreasing suddenly during rainfall events, as validated with the data produced by the SAWS Kirstenbosch rain gauge. Soil temperature and soil moisture had a positive relationship with an increase in soil temperature relating to an increase in soil moisture as per Equation 4.5.

As expected, the air temperature and relative humidity data were negatively related with an increase in temperature resulting in a decrease in humidity and vice versa. The air temperature followed the trend of the ambient temperature in Cape Town and decreased from June to August and began to increase from September to November. The relative humidity followed the inverse to the air temperature, with humidity dropping from June to August and increasing from September to November. The accuracy of the air temperature and relative humidity results were validated by using data collected by the SAWS Kirstenbosch weather station which indicated a combined average accuracy error of 7% for air temperature and relative humidity.

7.3 Reliability of System

The reliability of the system had two components, namely robustness and network communication. Robustness was measured by the system's ability to deal with unforeseen events and operate in the natural environment for the five month testing period. The systems experienced issues relating to power outages in the CoCT, resulting in both nodes losing connection to TTN and, in turn, losing battery life due to long periods of active mode trying to search for a listening gateway. This issue was resolved by purchasing a gateway and a UPS which ensured that there was always a listening gateway for nodes to transmit to during power outages. Deployment 0, mentioned in Section 6.2, was the first iteration of the system and was highly susceptible to theft and vandalism despite precautions being taken. The WMN and SAMN were designed and deployed to further minimize the risk of theft or vandalism and, during the deployment of the nodes, they operated without such occurrences. The physical components of the system

successfully protected the micro-electronics from the natural environment, with the systems operating successfully during multiple storm events from June to the start of November. Figure 7-6 shows the conditions of the node after deployment.

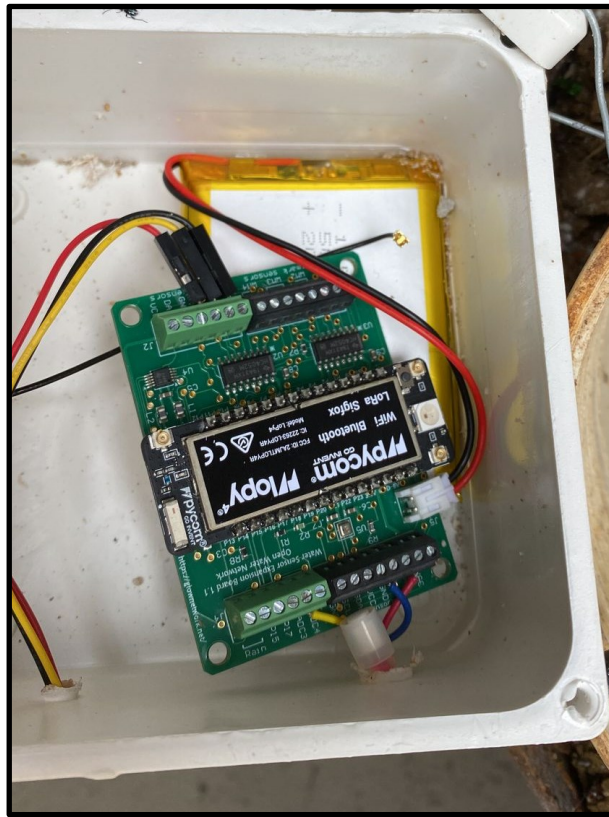


Figure 7-6: Condition of the Node after the End of Deployment

The network and communication analysis assessed the duty cycle and packet losses during deployment of the systems. Both systems met the regulations set out by ICASA (2015) which required the nodes to have a duty cycle below 1%. In terms of communication, this proves that the system is viable and meets regulations set by South African regulators. The packet losses for the WMN and SAMN deployments differed significantly between nodes and the different deployments of the nodes. Comparing packet losses between WMN and SAMN, there was a 2.92% greater packet loss for the SAMN node even though both nodes used the same communication protocol. The differences could be attributed to distance, elevation, and line of sight, but the main attribute is the length of deployment of the nodes. Over the 45 days of deployment, WMN had a packet loss of 3.10% while SAMN had a packet loss of 6.02% over 121 days. According to this metric, the SAMN had a better packet loss per day performance losing only 0.05% per day while the WMN lost 0.07% per day. According to Google Earth (2022), the WMN node, located at the dam was roughly 600 m away from the listening gateway in the NEB building while the SAMN node was roughly 100 m away from the listening gateway. The SAMN node had an elevation of 162 m while the WMN node had an elevation of 88 m. This indicates that a node with a greater elevation and a closer distance to listening gateways

experiences less packet losses per day during deployment. The packet loss percentage difference between SAMN1 and SAMN2 was 5.48% and this difference is not considered reliable due to the short period for which SAMN1 was deployed. In contrast, the percentage packet loss difference for WMN1 and WMN2 was 4.90%, with WMN1 having the greater percentage packets lost over a similar deployment period. During the deployment of WMN1, there was no listening gateway setup in the NEB building, thus the node had to transmit to listening gateways in the area. According to Google Earth (2022) and The Things Network (2021), the closest listening gateway during this deployment was in Rondebosch, roughly 3 km away. This further proves the drop off in communication reliability as the distance between the node and the listening gateway increases.

The overall reliability of the two systems ensured that a total of 14 415 data packages containing hydrological data were transmitted to TTN and then to Things Speak over a 166-day period while maintaining a duty cycle of below 1%. Packet losses for both systems improved after implementing a listening gateway at UCT and these improvements brought down the packet loss percentage average of the two nodes to below 5%. It was noted that opting for inaccessible spots for deployment reduced the risk of vandalism and theft significantly during the 166-day period.

7.4 Performance of System

Analysing performance of the system involved investigating the load testing and power consumption of the systems. The load testing observed any irregularities or flaws in the data produced by the two systems. The major irregularity found in the data produced by the WMN was with the water level node data. During changes in water level, fluctuations occurred between values until the water level stabilised on the higher reading and changes in the water level were only recorded once it had surpassed the 0.016 m threshold of the previous water level reading. This created issues with water level accuracy and precision when using the water level sensor. Additionally, the readings of the water level became unstable and unreliable when the battery voltage dropped below 2.575 mV. The water temperature sensor produced three unreliable data points during WMN1, while there were no irregular data points in WMN2. The sensor outputted -0.05 three times which, based on testing done on prototypes, usually results from wired connection issues. Thus, it is theorised that these readings came about due to hardware faults. The SAMN ran smoothly with only one single data point irregularity during deployment. The data point produced unrealistic data for all measurements taken by the node, and this could have been a result of a software malfunction during the reading and/or transmission of the data packet.

The power consumption of the system was analysed by first determining the theoretical battery life of the system and what capacity battery would be required to power the systems for the specified time. The battery life of the node was then monitored during deployment and the battery performance during WMN1 and WMN2 and SAMN1 and SAMN2 were compared. For deployments where battery capacity was not fully drained, a projected battery life was

determined using parameters of the deployment that did run until the end of its node-life. During the deployment of WMN, the WMN1 lost connection to TTN before the battery drained completely while WMN2 ran until the battery was depleted. WMN2 lasted for a duration of 23 days while WMN1 lasted a duration of 22 days with remaining battery power still available. The projected battery life based on the linear decline of the battery for WMN1 was 36 days, thus there was a 13-day difference between the actual battery life of WMN2 and the projected battery life of WMN1. The hardware, software, and setup of these systems were identical, and this indicates that the reasons for the changes in the power consumption were due to external factors. Looking at enclosure temperature and enclosure humidity during the deployment of the WMN1 and WMN2, there were differences in the averages across the two deployments as seen in Table 7-2. The average enclosure temperature during WMN1 deployment was 1.36°C higher than WMN2, while the average enclosure humidity was 3.38 higher for WMN1. A statistical summary report of the enclosure temperature and humidity data can be found in Appendix O.

Table 7-2: Enclosure Temperatures and Humidity Across WMN1 and WMN2

Deployment	Average Enclosure Temperature (°C)	Average Enclosure Humidity
WMN1	14.62	66.63
WMN2	13.26	70.01

Based on the power consumption comparisons and the comparisons between average enclosure temperature and humidity across WMN1 and WMN2, the lower temperature and higher humidity in WMN2 had a negative effect on battery performance, thus reducing the life cycle of the node. The theoretical battery life for WMN was deemed to be 34 days when using a theoretical battery capacity of 4865.81 mAh. The actual battery life for WMN2 when using a 5000 mAh battery was nine days shorter than the theoretical battery life, while the projected battery life of WMN1 was two days longer. These results indicated high variations in battery performance when using Li-Po batteries in varying temperature and humidity conditions. A node operating for 35 days, although adequate, is not ideal for IoT deployment when collecting hydrological data, especially in remote and difficult-to-access areas. Ideally, periods of operation for monitoring nodes should exceed 6 months to be deemed acceptable.

The SAMN deployment had a theoretical battery life of 3 months if powered using a theoretical battery capacity of 2581.15 mAh. The SAMN node was powered by a 5000 mAh Li-Po battery and the node was still producing data when last checked on the 01/11/2022. SAMN1 ran for a short period of time, thus effective battery performance and power consumption analysis was not possible. SAMN2 did not have an actual battery life value, thus the projected battery life using the linear decline equation of the battery was used. The projected battery life of the node was calculated to be 144 days, which equates to around 6 months of uninterrupted data. The

theoretical calculations verify these results and the projected lifespan of the node is deemed acceptable for hydrological monitoring.

7.5 Costs and maintenance

The cost analysis was done for the WMN and SAMN separately, as well as for the Hydrological Monitoring Node (HMN), also referred to as deployment 0, which was a single system that measured a combination of all the WMN and SAMN hydrological parameters. This allowed for a price comparison between the different variations of the deployment. The maintenance plan was developed using the power consumption analysis and information obtained during the deployment of the node. The maintenance plan was governed solely by battery life of the node which is the first hardware component that required replacing.

7.5.1 Cost Analysis

A summary of the cost analysis for the different systems can be seen in Table 7-3. The combination of the WMN and SAMN was used and tested in this research, and this involved the setup of two sensor nodes which increased overall costs. If it were possible, deploying the HMN would have reduced the overall cost of the project by R3559.81 while still having the ability to measure the same parameters as the WMN and SAMN combined. The variations of node deployment are project specific and depend on the location of deployment and the required parameters to be measured. A detailed cost analysis can be found in Appendix P for each of the systems.

Table 7-3: Summary of Cost Analysis for the Systems

Node	Total Cost (ZAR)
WMN	4277.93
SAMN	6016.09
HMN	6735.02
WMN + SAMN	10294.83

7.5.2 Maintenance Plan

Similarly to the costs analysis, a summary of the maintenance plan for each system can be seen in Table 7-4. The maintenance plan of the system is based on the projected battery life of the nodes calculated in Section 6.5 when using a 5000 mAh Li-Po battery. The maintenance of the WMN and the HMN require 10.1 scheduled maintenance visits a year for battery change while the SAMN requires 2.5 scheduled maintenance visits a year. The combination deployment of the SAMN and the WMN requires 10.1 scheduled maintenance visits per year. These maintenance plans are dependent on the energy efficiency of the system as well as the node life of system

which can differ from system to system. Changing battery capacity changes the required maintenance of each system.

Table 7-4: Maintenance Plans for the Systems Based on the use of a 5000 mAh Battery Li-Po Battery

Node	Node life	Maintenance visits per year
WMN	36	10.1
SAMN	144	2.5
HMN	36	10.1
WMN + SAMN	36	10.1

7.6 Summary

Validation of the data included using spot checks and comparisons with the nearby Kirstenbosch SAWS weather station. The spot checks done for the water level data indicated an average accuracy error of 0.0005% while the comparison with the SAWS rainfall data confirmed that the increases in water level corresponded with rainfall events during the deployment of the WMN. Similarly to the water level data, the soil moisture data was validated by comparing sensor output to rainfall events documented by the SAWS station which showed a correspondence between sensor output and rainfall events. The data produced by the air temperature and humidity sensor was compared with data produced by the station, and an average accuracy error of 7% was established for the SAMN. The experimental data output by the WMN and SAMN was documented with key findings highlighted and discussed. The reliability and performance analysis of the systems were discussed with significant findings from the analysis being highlighted. The WMN and SAMN nodes were found to be robust to the surrounding environment during the 5-month testing period, with communication efficiency being deemed acceptable with a packet loss average of 4.56% while the duty cycle of both nodes was below 1%. The projected life spans of the WMN and SAMN were 36 and 144 days, respectively, while unreliable, inaccurate, and erroneous data points were highlighted and addressed. The total cost of deploying the two systems was R10294.83 with other variations also costed for different applications. The maintenance plan, dependent on the battery life, for the WMN indicated scheduled maintenance 10.1 times a year while the SAMN only requires 2.5 scheduled visits a year.

8. Conclusion and Recommendations

8.1 Conclusion

The primary objective of this study was to determine transitional pathways from conventional hydrological data collection to IoT-driven collection using novel IoT technology. In this study, a setup of two hydrological monitoring nodes measuring water level, water temperature, soil moisture, air temperature and relative humidity was presented alongside a demonstration of hardware and software design. The setup of the monitoring node used low-cost, existing sensor node components and adapted and integrated these to measure the required hydrological data. Once the nodes were setup, they were integrated with an open network server, known as TTN, where the hydrological data was measured and then transmitted to the Things Speak application where the data was visually displayed and accessible to the end-user.

Inaccurate and unreliable hydrological data is a major problem when modelling and making decisions for WRM. It is necessary to develop ways to ensure high quality and reliable hydrological data that would vastly improve hydrological models and decision making done in WRM. This research shows the viability of developing a low cost IoT-driven hydrological monitoring system that produces reliable and accurate data. The monitoring systems investigated in this research were built using Pycom LoPy4 devices as the main processing unit of the setup, coupled with the water sensor expansion board 1.1 PCB that allowed for the integration of the sensors measuring hydrological data. The sensors were checked for accuracy and precision in line with their specifications set out by the manufacturer before deployment. The data was transmitted over TTN and made available to end-users using the Things Speak platform. The experimental assessment of the setups began with deploying the sensor nodes and exposing them to field conditions. The data collected by the nodes were assessed based on their data output, integrity, reliability, and performance.

The output data produced by the WMN and SAMN produced data for water level, water temperature, soil moisture, air temperature and relative humidity. The data produced was displayed visually through gauges and scatter plots that updated in real-time on Things Speak and the data was downloaded for further analyse using Excel. The data produced was validated using spot checks done during deployment as well as rainfall data produced by the SAWS Kirstenbosch rain gauge to determine the accuracy and consistency of the data. In comparison to spot checks, water level had an average accuracy error of 0.0005% and the sensor output corresponded to rainfall data from the SAWS station. The water temperature and soil temperature sensors' accuracy and precision were validated during pre-deployment tests, while the soil moisture sensor's output corresponded with the rainfall data produced by the SAWS station. The SAWS station data was also used to validate the air temperature and relative humidity data captured for this study, with there being an average accuracy error of 7% for the air temperature and relative humidity. The results based on the output data produced by the nodes validated the viability of producing hydrological data using IoT technology.

The reliability of the nodes was tested by analysing the robustness, communication and network analysis of the nodes. The robustness of the nodes was assessed by analysing the systems' response to abnormalities or unforeseen circumstances during operation as well as the nodes' reaction and adaption to the natural environment during the 5 months of deployment. The WMN and SAMN had issues during power outages that caused losses in connection with TTN for extended periods of time. This resulted in poor battery performance and high-power consumption. To combat this, a LoRa gateway was purchased and setup in the NEB building with a UPS. Once the gateway was setup, robustness of the nodes improved as they were then unaffected by power outages. The components that housed the microelectronics of the nodes successfully protected the nodes and provided robustness to natural conditions, such as rainfall events, high winds, and solar radiation during the 5 months of deployment. Further proof of the robustness of the system was highlighted by the absence of theft or vandalism to the two nodes throughout the deployment period. This highlighted the importance of positioning nodes in inaccessible spaces and areas where possible. The results from the analysis of the network and communication supported the viability of the hydrological monitoring nodes. The duty cycles of the systems were both below 1% which adhered to South African regulations and the overall average packet losses for both systems were below 5%. It was noted that packet losses increased with an increase in distance of a node from a listening gateway while there was a decrease in packet loss with an increase in elevation of a node.

The performance of the nodes was analysed in this research by investigating the results from load testing of the nodes and by assessing the results from the power consumption analysis of both nodes. The load testing of the nodes, which looked at irregularities and flaws in the hardware and software operation, found issues of precision when analysing results from the water level sensor. The water level sensor also produced abnormal and unreliable data towards the end of the WMN lifespan. These flaws and abnormalities can be corrected through improved and more efficient programming of the water level sensor. The water temperature sensor produced three unreliable data points while the rest of the points were deemed valid. These error points in the data could be attributed to random hardware or software faults during operation – however, the effects of these faults are deemed insignificant due to the frequency at which the faults occurred during deployment. The SAMN node had a single unreliable data point during the full duration of deployment, and this could be attributed to a software fault during operation due to all sensors of the node producing the error point at the same time. Due to the low frequency of error points during SAMN deployment, this was deemed insignificant.

The results from the power consumption analysis created a scenario where WMN was split into two iterations of deployment due to power outages. The difference between the projected battery life of WMN1 and the actual battery life of WMN2 was 13 days and this difference could be attributed to the difference in enclosure conditions across WMN1 and WMN2. There was roughly a 2-day difference between the theoretical battery life of WMN and the projected battery life of WMN1. This difference is considered negligible due to the variations between theoretical testing and deployment. The actual battery life of WMN2 had a 9-day difference to the theoretical battery life of WMN1. Variations in enclosure conditions during

deployment could be the main reason for this difference. For hydrological monitoring, maintenance once a month is not viable and hence the WMN would need to improve energy efficiency and power consumption to be deemed viable for hydrological monitoring. This could be achieved by increasing the intervals between readings to reduce power consumption. The power consumption analysis of the SAMN node indicated that the projected battery life of the node would be roughly 144 days which equates to around 6 months of uninterrupted data. Comparisons between the projected and theoretical battery life of the node showed negligible difference and further supported the findings for the projected battery life of SAMN. A lifespan of 6 months was deemed acceptable for hydrological monitoring. This provides significant time for uninterrupted data collection and the development of a maintenance plan that can optimise operational costs. Further optimisation of operational costs would involve the implementation of energy harvesting techniques.

Overall, the two hydrological monitoring nodes built for this research produced hydrological data and transmitted this data using IoT technology to an end-user. Based on the findings of this research, there is potential for the implementation of IoT technology to build a system that can produce hydrological data in South Africa. Based on literature, the nodes met requirements needed to measure hydrological data effectively and reliably for extended periods of time while maintaining low-cost, low-power consumption, and fit-for-purpose performance.

8.2 Recommendations for Future Work

This research undertook a pilot study that investigated the viability of using IoT technology to create a system that measures hydrological data remotely in South Africa. Although a concept was developed during this research, there is still work to be done to establish this preposition. Sensor accuracy can only be determined by tried and tested methods while hydrological nodes require a lifespan of 6 months or greater for optimal efficiency, especially in remote areas. Therefore, future work will be based on the findings and results of this research.

Due to theft and vandalism during this research, two nodes were deployed instead of the initially planned single node. The single node would have produced all the measurements the WMN and SAMN produced and would have operated off one Pycom LoPy4 device, one PCB and one battery. If possible, it is recommended that only one node be deployed that measures all the relevant hydrological data as this would cut the cost of the system significantly. This would also simultaneously make the system more efficient by providing the system with the ability to measure multiple parameters, making it a more suitable hydrological monitoring station. Adding a rain gauge and water quality sensors to the system should also be investigated as these are important parameters for hydrological monitoring.

Although the systems' network and communication reliability improved when a LoRa gateway was purchased and setup, for future work a cheaper alternative gateway can be built for IoT projects and used in remote areas where the installation of a gateway is required. It is also recommended that these types of systems are placed in inaccessible areas or with the required

security to prevent theft and vandalism. Furthermore, future work would benefit from increasing the testing and deployment duration of the nodes to greater than 5 months to better understand the reliability, robustness and durability of the nodes when deployed for extended periods of time.

Based on the results of the data produced by the water level sensor, it was evident that future deployments of this sensor would require an increase in precision during data collection to improve performance. This could be achieved by programming the sensor to take multiple readings and obtain an average across the readings. Another way to improve performance of the system and increase the lifespan of a node is to improve energy efficiency and reduce power consumption. Power consumption could be reduced by using power efficient components which would reduce deep sleep current, while energy harvesting techniques, such as using solar power and/or hydropower to recharge the batteries, would create a system with an infinite battery life. Due to the low frequency of change in water level when there is no rainfall, the WMN could be programmed to only transmit measurements when there has been a change in water level or, alternatively, the time interval between the node waking up and taking a measurement could be increased to 1 hour. Lastly, it is recommended to insulate the battery powering the node to allow the battery to operate at optimum performance under optimal temperature and humidity conditions.

Future work in this field would benefit from the full development of an MCU, a PCB and an application to which the data is sent for visualisation and analysis to mitigate any limitations that may come about when utilising retail components and applications for hydrological monitoring. This would also increase security of the system and it would give future researchers more freedom in the creation and modification of these systems. Furthermore, it would prevent the reliance on retail components that could be discontinued. An example of this was experienced during the latter stages of this research when Pycom Ltd indicated to their customers and community that they will no longer be manufacturing LoPy4 devices.

9. References

- Akhila, P. & Senthil Prabha, R., 2019. Water Monitoring System Based on Internet of Things. *International Journal of Software and Computer Science Engineering*, 4(2), pp. 11-19.
- Al-Fuqaha, A. et al., 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys and Tutorials*, 17(4), pp. 2347-2376.
- Alkhatib, A., Abed, A., Alkhatib, A. & Singh Baicher, G., 2012. *MAC Layer Overview for Wireless Sensor Networks*. Singapore, Computer Networks and Communication Systems.
- Al-Kofahi, M. M., Al-Shorman, M. Y. & Al-Kofahi, O. M., 2019. Toward energy efficient microcontrollers and Internet-of-Things systems. *Computers and Electrical Engineering*, Volume 79.
- Althobaiti, A. S. & Abdullah, M., 2015. Medium Access Control Protocols for Wireless Sensor Networks Classifications and Cross-Layering. *Procedia Computer Science*, Volume 65, pp. 4-16.
- Arasteh, H. et al., 2016. *IoT-based smart cities: A survey*. Florence, Institute of Electrical and Electronic Engineers.
- Arjan, 2021. *Airtime calculator for LoRaWAN*. [Online] Available at: <https://avbentem.github.io/airtime-calculator/ttn/eu868> [Accessed 15 September 2022].
- Ashton, K., 2009. *RFID Journal*. [Online] Available at: <https://www.rfidjournal.com/articles/view?4986> [Accessed 19 March 2020].
- Augustin, A., Yi, J., Clausen, T. & Townsley, W. M., 2016. A study of Lora: Long range & low power networks for the internet of things. *Sensors (Switzerland)*, 16(9).
- Baadache, A. & Adouane, R., 2015. Minimizing the energy consumption in wireless sensor networks. *Ad-Hoc and Sensor Wireless Networks*, 27(3-4), pp. 223-237.
- Bartos, M., Wong, B. & Kerkez, B., 2018. Open storm: A complete framework for sensing and control of urban watersheds. *Environmental Science: Water Research and Technology*, 4(3), pp. 346-358.
- Bhavsar, A., 2017. *IIoT World*. [Online] Available at: <https://iiot-world.com/connected-industry/a-guide-for-selecting-the-right-microcontroller-for-your-iiot-project/> [Accessed 20 April 2020].
- Blom, J., 2021. *Voltage Dividers - learn.sparkfun.com*. [Online] Available at: <https://learn.sparkfun.com/tutorials/voltage-dividers/all> [Accessed 15 September 2021].
- British Hydrological Society, 2014. *British Hydrological Society*. [Online] Available at: http://www.hydrology.org.uk/Hydrological_data.php [Accessed 10 April 2020].

Brodie, L., 2019. *Methods of Measuring Soil Moisture*. [Online] Available at: <http://southafrica.co.za/methods-measuring-soil-moisture.html> [Accessed 14 April 2020].

Bröring, A. et al., 2011. New generation Sensor Web Enablement. *Sensors*, 11(3), pp. 2652-2699.

Cerchecci, M. et al., 2018. A low power IoT sensor node architecture for waste management within smart cities context. *Sensors (Switzerland)*, 18(4).

Core Electronics, 2020. *Pycom LoPy Overview*. [Online] Available at: <https://core-electronics.com.au/guides/pycom-lopy-overview/> [Accessed 23 June 2020].

Dallas, H., 2008. Water temperature and riverine ecosystems: An overview of knowledge and approaches for assessing biotic responses, with special reference to South Africa. *Water SA*, 34(3), pp. 393-404.

Das, B. & Jain, P., 2017. *Real-Time Water Quality Monitoring System using Internet of Things*. Jaipur, Institute of Electrical and Electronic Engineers .

Data Flair Team , 2018. *Data Flair*. [Online] Available at: <https://data-flair.training/blogs/iot-testing/> [Accessed 4 August 2020].

ESRI 2021. ArcGIS Desktop: Release 10.8.1. Cape Town, Western Cape: University of Cape Town, Upper Campus.

Fell, J., Pead, J. & Winter, K., 2019. Low-cost flow sensors: Making smart water monitoring technology affordable. *IEEE Consumer Electronics Magazine*, 8(1), pp. 72-77.

Fernandes Carvalho, D. et al., 2019. A test methodology for evaluating architectural delays of LoRaWAN implementations. *Pervasive and Mobile Computing*, Volume 56, pp. 1-17.

Ghapar, A. A., Yussof, S. & Bakar, A. A., 2018. Internet of Things (IoT) Architecture for Flood Data Management. *International Journal of Future Generation Communication and Networking*, 11(1), pp. 55-62.

Google Earth 7.3.6.9345, University of Cape Town, Upper Campus 33°57'27.55"S, 18°27'40.32"E, elevation 105M. 2D map, viewed 20 April 2022, <<http://www.google.com/earth/index.html>>.

Guaman, J. et al., 2018. *Water Level Monitoring System Based on LoPy4 Microcontroller with LoRa technology*. Lima, INTERCON.

Han, K., Zhang, D., Bo, J. & Zhang, Z., 2012. Hydrological Monitoring System Design and Implementation Based on IoT. *Physics Procedia*, Volume 33, pp. 449-454.

Haxhibeqiri, J., De Poorter, E., Moerman, I. & Hoebeke, J., 2018. A survey of LoRaWAN for IoT: From technology to application. *Sensors (Switzerland)*, 18(11).

Holzbecher, E., Hadidi, A., Barghash, H. & Al Balushi, K., 2019. *Application of Big Data and Technologies for Integrated Water Resources Management - A Survey*. Muscat, Oman, Institute of Electrical and Electronic Engineers.

Huang, A. & Roth, E., 2017. *Basics of Power Switches Application Report Basics of Power Switches*, Dallas: Texas Instruments.

IEC, 2014. *Internet of Things: Wireless Sensor Networks*, Geneva : International Electrotechnical Commission

Independent Comm. Authority of S. Africa. 2015. National Policy for Health Act, 1990 (Act No. 279 of 2015). The Radio Frequency Spectrum Regulations 2015. Government Gazette No. 38641:597 30 Mar.

Inst Tools, 2020. *Advantages and Disadvantages of Network Topologies*. [Online] Available at: <https://instrumentationtools.com/advantages-and-disadvantages-of-network-topologies/> [Accessed 8 April 2020].

IntellegenInc., 2015. *Pros and Cons of Different Types of Network Topologies*. [Online] Available at: <https://intellegensinc.com/resources/blog/pros-and-cons-of-different-types-of-network-topologies/> [Accessed 8 April 2020].

InternalZero, 2018. *InternalZero*. [Online] Available at: https://www.intervalzero.com/hardware/basic-structure-operation_microcontrollers/ [Accessed 30 March 2020].

IoT European Research Centre, 2014. *Internet of Things*. [Online] Available at: internet-of-things-research.eu/about_iot.htm [Accessed 10 March 2020].

Irrrometer, 2020. *Reading WATERMARK Soil Moisture Sensors*. [Online] Available at: <https://www.irrometer.com/200ss.html> [Accessed 14 September 2021].

Ismail, A. A., Hamza, H. S. & Kotb, A. M., 2019. *Performance Evaluation of Open Source IoT Platforms*. Cairo, Institute of Electrical and Electronics Engineers Inc..

Jamali, M. A. J. et al., 2020. IoT Architecture: Towards the Internet of Things. In: C. Springer, ed. *EAI/Springer Innovations in Communication and Computing*. Geneva: Springer Nature, pp. 9-31.

Joannin, C., 2020. *Which types of batteries for your IoT device?*. [Online] Available at: <https://www.saftbatteries.com/energizing-iot/types-batteries-iot-devices> [Accessed 27 August 2020].

Khutsoane, O., Isong, B. & Adnan Abu-Mahfouz, n. M., 2017. *IoT Devices and Applications based on LoRa/LoRaWAN*, Mafikeng: Institute of Electrical and Electronic Engineers .

Kingatua, A., 2018. *Medium*. [Online] Available at: <https://medium.com/supplyframe-hardware/iot-system-tests-checking-for-failure-c146d2ebb8ef> [Accessed 12 August 2020].

Kramarenko, V. V. et al., 2016. *Determination of water content in clay and organic soil using microwave oven*. Tomsk, Institute of Physics Publishing.

Lazarescu, M. T., 2013. Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 3(1), pp. 45-54.

Linke, B., 2008. *Maxim Integrated*. [Online] Available at: https://www.maximintegrated.com/en/design/technical_documents/tutorials/1/1796.html [Accessed 8 December 2021].

LoRa Alliance, 2020. *LoRa*. [Online] Available at: <https://lora-developers.semtech.com/library/tech-papers-and-guides/the-book/data-packet-transmissions/> [Accessed 16 August 2020].

LoRa Alliance, 2020. *LoRa Alliance*. [Online] Available at: <https://loralliance.org/about-lorawan> [Accessed 11 August 2020].

Macedo, D., Guedes, L. A. & Silva, I., 2014. *A dependability evaluation for Internet of Things incorporating redundancy aspects*. Miami, FL, USA, IEEE.

Madakam, S., Ramaswamy, R. & Tripathi, S., 2015. Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, 3(5), pp. 164-173.

Martin, M. & Islam, M., 2012. *Overview of Wireless Sensor Network*. [Online] Available at: <https://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/overview-of-wireless-sensor-network> [Accessed 30 March 2020].

Masinde, M., 2014. *IoT applications that work for the African continent: Innovation or adoption?*. Porto Alegre, Brazil, Institute of Electrical and Electronics Engineers Inc..

Mathews Modieginiane, K., Letswamotse, B. B., Malekian, R. & Abu-Mahfouz, A. M., 2017. *Software Defined Wireless Sensor Networks Application Opportunities for Efficient Network Management: A Survey*, Pretoria: Council for Scientific and Industrial Research.

MathWorks, 2020. *API References*. [Online] Available at: <https://www.mathworks.com/help/thingspeak/channels-and-charts-api.html> [Accessed 28 August 2020].

Meintjies, C. J., 2001. *Guidelines to regional socio-economic analysis*, Johannesburg: Development Bank of Southern Africa.

Mekki, K., Bajic, E., Chaxel, F. & Meyer, F., 2018. *Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT*. Athens, Institute of Electrical and Electronics Engineers.

Mendoza-Cano, O. et al., 2021. Experiments of an IoT-based wireless sensor network for flood monitoring in Colima, Mexico. *Journal of Hydroinformatics*, 23(3), pp. 385-401.

Miazi, M. N. S. et al., 2016. *Enabling the Internet of Things in developing countries: Opportunities and challenges*. Dhaka, Bangladesh, Institute of Electrical and Electronics Engineers Inc..

Moreno, C. et al., 2019. Rivercore: IoT device for river water level monitoring over cellular communications. *Sensors (Switzerland)*, 19(1).

Nomqophu, W., 2018. *Potential Remote Sensing Technologies to Enhance the Monitoring and Reporting of Water Flows*, Pretoria: Water Research Commission.

Olsson, K. & Finnsson, S., 2017. *Exploring LoRa and LoRaWAN A suitable protocol for IoT weather stations?*, Gothenburg, Sweden: Chalmers University of Technology .

Pandya, K., 2013. Network Structure or Topology. *International Journal of Advance Research in Computer Science and Management Studies*, 1(2), pp. 22-27.

Pardini, K. et al., 2019. IoT-based solid waste management solutions: A survey. *Journal of Sensor and Actuator Networks*, 8(1), pp. 1-25.

Parker, T., 2016. *Avent Abacus - Design trends: Batteries in IoT*. [Online] Available at: <https://www.avnet.com/wps/portal/abacus/resources/article/batteries-for-iot/> [Accessed 26 August 2020].

Peacock, C., 2017. *Aquaread*. [Online] Available at: <https://www.aquaread.com/why-water-level-monitoring-is-so-important/> [Accessed 18 March 2020].

Pellicer, S. et al., 2013. *A Global Perspective of Smart Cities: A Survey*. Taichung, Institute of Electrical and Electronic Engineers, pp. 439-444.

Pretorius, D. D., 2019. *Wireless Telemetry based River Level Sensing*, Stellenbosch: Department of Electrical and Electronic Engineering at the University of Stellenbosch.

Pycom, 2021. *Pycom docs - Firmware & API reference - Pycom Modules*. [Online] Available at: <https://docs.pycom.io/firmwareapi/pycom/> [Accessed 24 June 2021].

Raaijen, T. & Daneva, M., 2017. *Depicting the smarter cities of the future: A systematic literature review & field*. Prague, Institute of Electrical and Electronic Engineers.

Ragnoli, M. et al., 2020. An autonomous low-power lora-based flood-monitoring system. *Journal of Low Power Electronics and Applications*, 10(2).

Raj, A. & Steingart, D., 2018. Review—Power Sources for the Internet of Things. *Journal of The Electrochemical Society*, 165(8), pp. B3130-B3136.

Ram, P., 2018. *LPWAN, LoRa, LoRaWAN and the Internet of Things*. [Online] Available at: <https://medium.com/coinmonks/lpwan-lora-lorawan-and-the-internet-of-things-aed7d5975d5d> [Accessed 18 August 2021].

Raza, U., Kulkarni, P. & Sooriyabandara, M., 2017. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys and Tutorials*, 19(2), pp. 855-873.

Rhee, S., Seetharam, D. & Liu, S., 2004. *Techniques for Minimizing Power Consumption in Low Data-Rate Wireless Sensor Networks*, Cambridge : Institute of Electrical and Electronic Engineers.

Rmsm, D., Wickramaarachchi, H. & Whc, W., 2019 . *IoT Based Water Level Monitoring System Using Nodemcu*. Sri Lanka, Applied Science, Business & Industrial Research Symposium.

Robles, T. et al., 2014. *An Internet of Things-based model for smart water management*. San Jose, California , Institute of Electrical and Electronics Engineers. .

Sagdeo, P., 2018. Application Programing Interfaces and the Standardization Value Appropriation Problem. *Harvard Journal of Law & Technology*, 32(1), pp. 236-262.

Saravanan, M., Das, A. & Iyer, V., 2017. *Smart Water Grid Management using LPWAN IoT Technology*, Chennai: Institute of Electrical and Electronic Engineers .

Semtech, 2019. *LoRa and LoRaWAN: A Technical Overview*, s.l.: Semtech.

Semtech, 2021. *Lora - Platform for IoT*. [Online] Available at: <https://www.semtech.com/lora> [Accessed 11 August 2021].

Shiferaw, Y., Arora, A. & Kuipers, F., 2020. *LoRaWAN Class B Multicast Scalability*, Delft, Netherlands: Delft University of Technology.

Shirvanimoghaddam, M. et al., 2019. Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting. *IEEE Access*, Volume 7, pp. 94533-94556.

Shock, C. C., Barnum, M. J. & Seddigh, M., 1998. *Calibration of Wermark Soil Moisture Sensors for Irrigation Management* , Ontario: Malheur Experiment Station.

Singh, K., 2020. *Smart Supply Chains Blog*. [Online] Available at: <https://smartsupplychains.ai/2020/01/05/a-comprehensive-introduction-to-reinforcement-learning/> [Accessed 23 March 2020].

Sivakugan, S., 2016. *LinkedIn*. [Online] Available at: <https://www.linkedin.com/pulse/laboratory-vs-situ-testing-siva-sivakugan/> [Accessed 14 April 2019].

Soua, R. & Minet, P., 2011. *A survey on energy efficient techniques in wireless sensor networks*. Toulouse, Institute of Electrical and Electronic Engineers.

South African Environmental Observation Network (SAEON), 2017. *SAEON Annual Report 2016/2017*, Pretoria: Dept. of Science and Technology .

Stasch, C. et al., 2009. A Stimulus-Centric Algebraic Approach to Sensors and Observations. In: N. Tringoni, A. Markham & S. Nawaz, eds. *Lecture Notes in Computer Science*. Berlin: Springer, Berlin, Heidelberg, pp. 169-179.

Sumi, L. & Ranga, V., 2016. *Sensor enabled Internet of Things for smart cities*. Wagnaghat, Institute of Electrical and Electronic Engineers .

The Things Network, 2021. *The Things Network*. [Online] Available at: <https://www.thethingsnetwork.org/> [Accessed 7 September 2021].

The Things Stack, 2021. *ABP vs OTAA*. [Online] Available at: <https://www.thethingsindustries.com/docs/devices/abp-vs-otaa/> [Accessed 23 August 2021].

Trevathan, J. & Johnstone, R., 2018. Smart environmental monitoring and assessment technologies (Semat)—a new paradigm for low-cost, remote aquatic environmental monitoring. *Sensors (Switzerland)*, 18(7).

Tsavalos, N. & Hashem, A. A., 2018. *Low Power Wide Area Network (LPWAN) Technologies for Industrial IoT Application*, Lund, Sweden: Lund University.

Turcu, C., Turcu, C. & Gaitan, V., 2018. *An Internet of Things Oriented Approach for Water Utility Monitoring and Control*, Suceava: World Scientific and Engineering Academy and Society.

Unidata, 2017. *Unidata*. [Online] Available at: <https://www.unidata.com.au/application-notes/rivers-streams-monitoring/> [Accessed 18 March 2020].

United Nations, 2016. *Sustainable Development Goals*. [Online] Available at: <https://www.un.org/sustainabledevelopment/sustainable-development-goals/> [Accessed July 2019].

Water Resource Information Management, 2012. *THE Annual National State of Water Resource Report*, Pretoria : Department of Water Affairs.

Weiser, M., Gold, R. & Brown, J. S., 1999. The origins of ubiquitous computing research at PARC in the late 1980s. *IBM Systems Journal*, 38(4), pp. 693-696.

Yelamarthi, K., Aman, M. S. & Abdelgawad, A., 2017. An application-driven modular IoT architecture. *Wireless Communications and Mobile Computing*, pp. 1-16.

Young, J. K., 2008. *A Practical Guide to Battery Technologies for Wireless Sensor Networking*. [Online] Available at: <https://www.fierceelectronics.com/components/a-practical-guide-to-battery-technologies-for-wireless-sensor-networking> [Accessed 27 August 2020].

Zennaro, M., 2018. *ICTP: Why we will use microPython*. [Online] Available at: http://wireless.ictp.it/school_2018/Slides/whymicropython.pdf [Accessed 18 May 2020].

Zennaro, M., Bagula, A. & Nkoloma, M., 2012. *From training to projects: Wireless sensor networks in Africa*. Washinton, United States, Institute of Electrical and Electronics Engineers Inc..

Zhang, D. et al., 2019. A low-cost smart sensor network for catchment monitoring. *Sensors (Switzerland)*, 19(10).

Appendices

Appendix A - Stepwise Summary for Sensor Deployment

• Step 1 – Supplies:

Hardware:

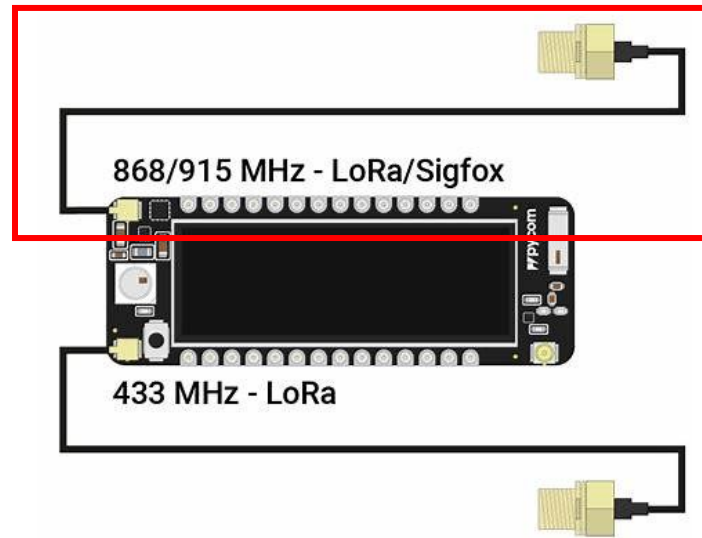
- 2 x Pycom LoPy4
- 2 x LoRa/Sigfox Antenna
- Pycom Expansion Board
- 2 x Water Sensor Expansion Board 1.1 (Open Water Network)
- 2 x 5 m DS18B20 submersible temperature sensor + Veroboard, 4.7 kohm resistor
- 1 x Hafei WNK submersible water level sensor
- 1 x Irrometer watermark soil moisture sensor
- 1 x DF Robot SHT31 Weather-Proof Temperature & Humidity Sensor
- 2 x 3.3 V 5000 mAh Li-Po Battery
- 2 x 8 GB Micro SD card
- 2 x IP65 Weatherproof enclosure
- 2 x L brackets
- 8 x galvanised saddles
- 2 m 20 mm PVC pipe
- Nuts and bolts
- PVC Male Adaptor 20 mm
- Jumper cables
- Wall bracket
- 6 x 16 diameter plastic plates
- Washers
- 3 x 15 cm stainless steel rods

Software:

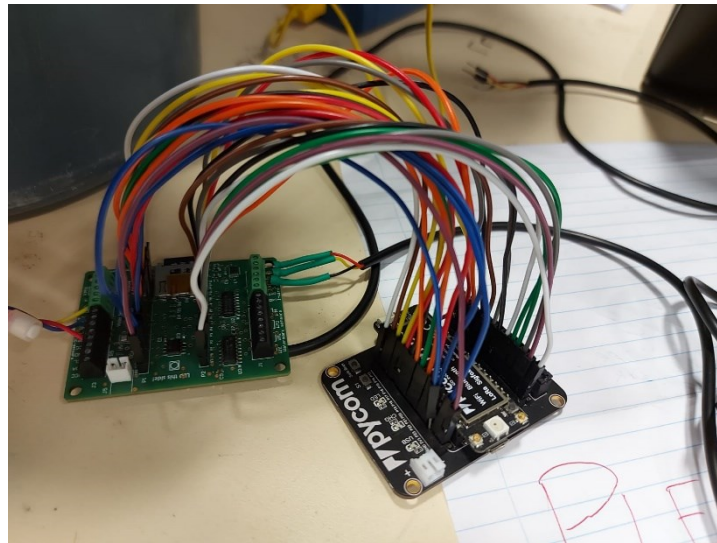
- VS Code IDE - [Download Visual Studio Code - Mac, Linux, Windows](#) and follow [instructions for download](#).
- Pymakr extension for VS Code

- **Step 2 – Wiring**

- LoPy4 mounted on Pycom Expansion board and Antenna connected 868/915 MHz LoRa/Sigfox connection.

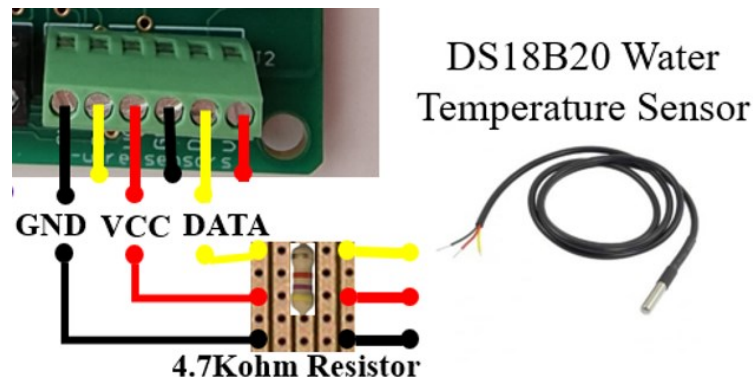


- Connect Pycom Expansion Board to Water Sensor Expansion board using jumper cables

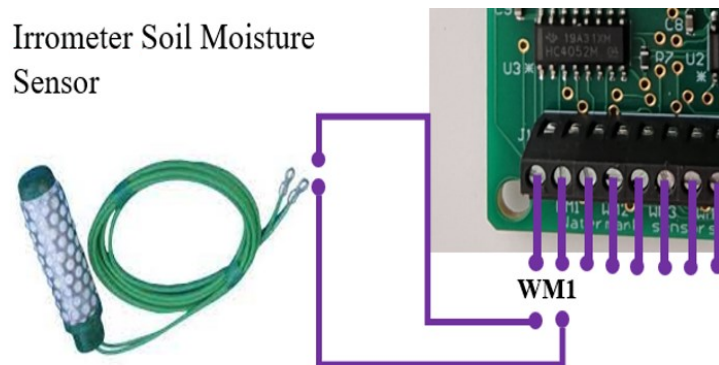


- Connect Sensors to relevant terminals as seen below:

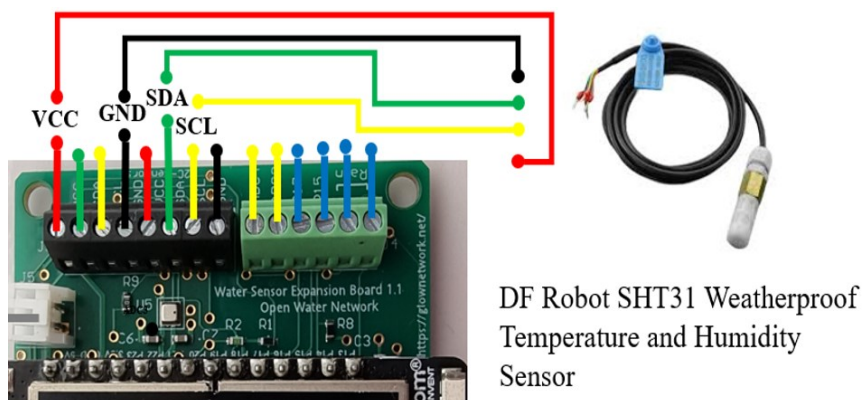
DS18B20 sensors connected at one-wire terminal – Red to VCC, Black to GND and yellow to data terminal with 4.7 kohm pull-up resistor over the VCC and data line of the sensor.



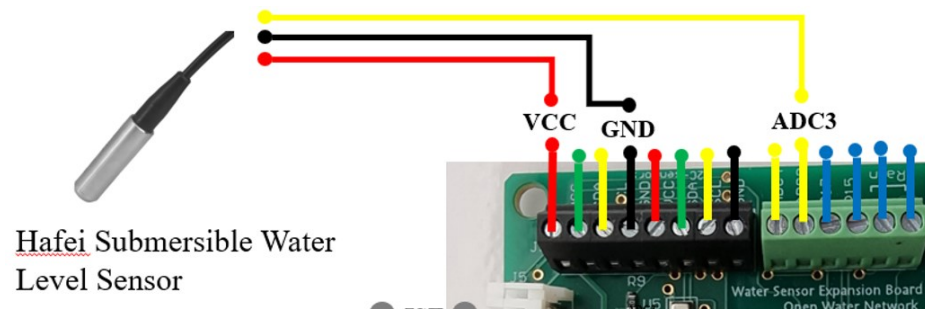
Watermark sensor connected at watermark terminal to WM1 terminal. Order does not matter.



DF Robot SHT31 Weather-Proof Temperature & Humidity Sensor connected to I2C terminal – Red to VCC, black to GND, green to SDA and yellow to SCL.



Hafei WNK submersible water level sensor connected at I2C sensor and terminal for extra pin connection – Red to VCC, black to GND and yellow to ADC3.



- Once all sensors connected, connect Pycom Expansion board to computer/Laptop using USB.
- **Step 3 – VS Code Directions for Use**
 - Introduction to VS Code - [Get Started with Visual Studio Code](#)
 - Click link to open page and follow the instructions on getting started with VS Code
 - Ensure that you download the pymkr extension in order to communicate with Pycom devices.
- **Step 4 – Downloading the Script**
 - Download the script for the hydrological monitoring station from the following relevant github repositories – [RobYNWA \(github.com\)](#)
 - Changes can be made to the scripts through VS Code IDE.
 - Once finished working on the script – upload to device using pymkr extension as seen below:
- **Step 5 – Connecting to TTN**
 - Create a TTN account
 - If no existing gateway, register a gateway in your area.
 - Add your application and register your device - [Getting Started on The Things Network - Tutorial Australia \(core-electronics.com.au\)](#)
 - Join the network with Over The Air Activation
 - Send data from your device to the TTN Application
- **Step 6 – Integrate TTN data with Things Speak**
 Detailed instructions of how to link TTN with Things Speak - [ThingSpeak | The Things Network](#)
- **Step 7 – Build Sensor Node Enclosure**

- Connect L-bracket to IP65 weatherproof enclosure using nuts and bolts
 - Connect PVC to enclosure using PVC male adaptor.
 - Run sensors through PVC
 - DF Robot SHT31 Weather-Proof Temperature & Humidity Sensor enclosed with a radiation shield to protect from exposure to direct sunlight - [DS18B20 Radiation Shield : 10 Steps \(with Pictures\) - Instructables](#)
- **Step 8 – Deployment of Node**
 - Deploy Sensor node by mounting sensor enclosure on a mountable surface, building or tree.
 - Fix PVC pipe to the mountable surface using galvanised saddles, nuts and bolts and run sensors through the PVC pipe into soil and or water medium to be measured.

Appendix B – Bit-Packing Examples

Table B-1: Bit-Packing Examples (LoRa Alliance, 2020)

Quantity	Desired quantity	Bit packing formula	Bits required	Range	Precision
Time	Arbitrary date and time to second	$T = \text{seconds from Jan 1, 1970, 00:00:00}$	32	Jan 1, 1970, 00:00:00 to Jan 19 2038 03:14:07	1 second
Temperature	Temperature wide range	$T = 0.5 * (t - 80) \{t:0-255\}$	8	-40°C to 87.5°C	0.5°C
Temperature	Temperature narrow range	$T = 0.25 * (t - 40) \{t:0-255\}$	8	-10°C to 53.75°C	0.25°C
Temperature	Temperature course	$T = 4 * (t - 1.25) \{t:0-15\}$	4	-5°C to 55°C	4°C
Voltage	Voltage wide range	$V = 0.1 * v \{v:0-0.225\}$	8	0 to 25.5V	0.1V
Voltage	Voltage narrow range	$V = 0.1 * (v * 26) \{v:0-15\}$	4	2.6V to 4.1V	0.1V
Battery level	Battery % precise	$P = 100\% * p/15 \{p:0-15\}$	4	0-100%	6.67%
Battery level	Battery % course	$P = 100\% * p/3 \{p:0-3\}$	2	0-100%	33.33%

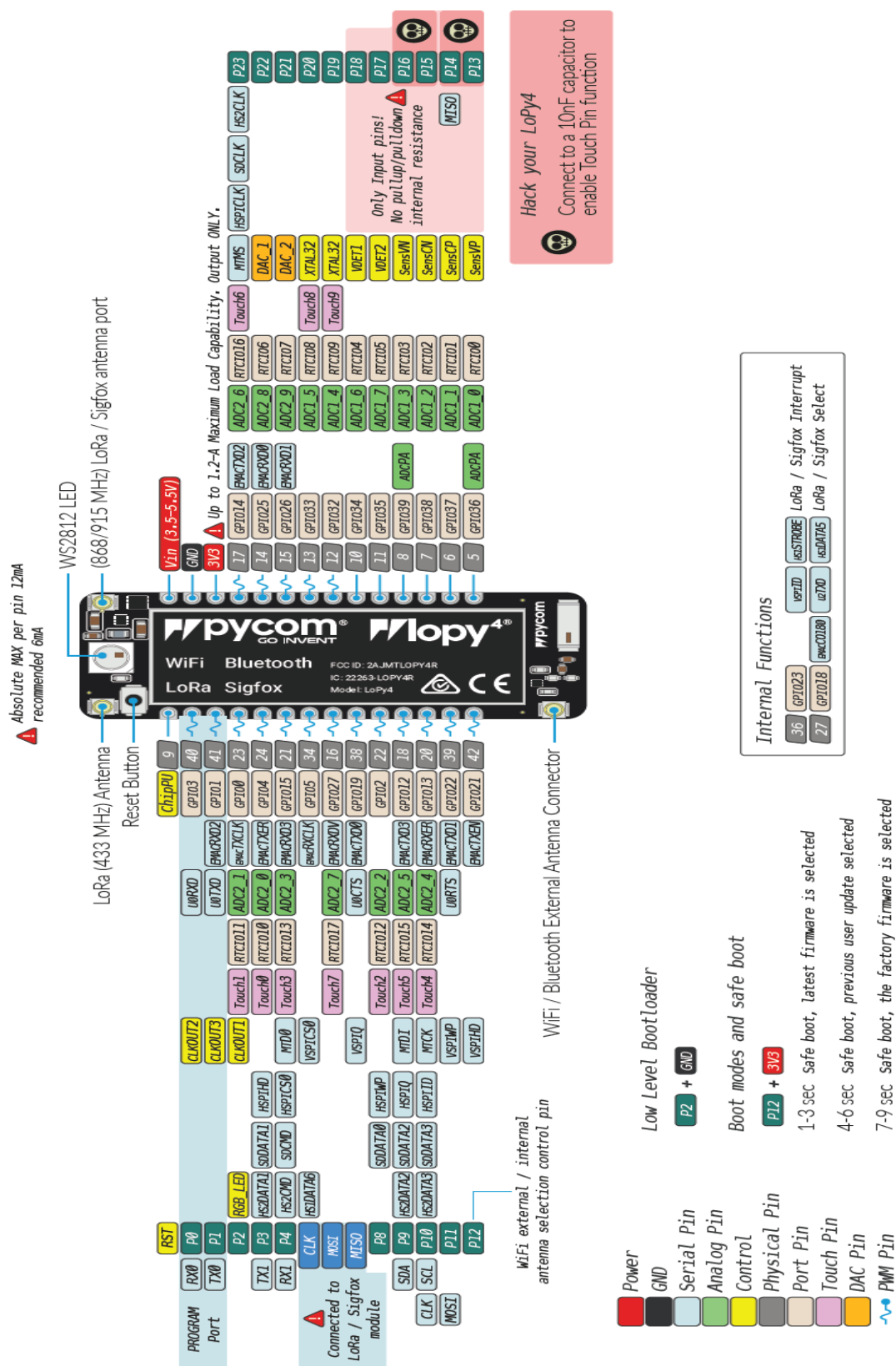
Appendix C – MCU Specifications

C-1 – Comparison of Arduino and Pycom Boards

Table B-1: MCU Specifications

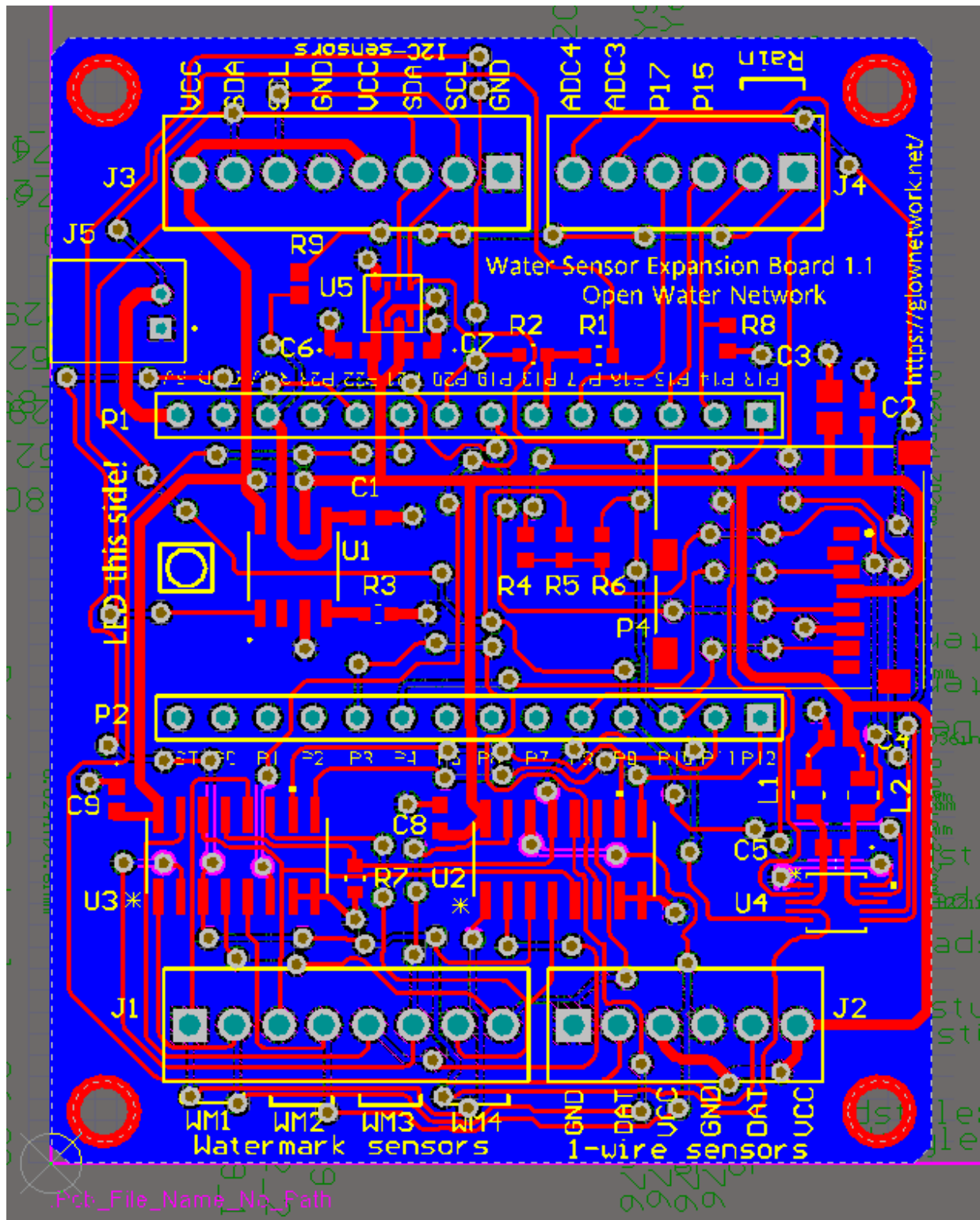
Characteristic	arduino mkr wan 1300	Pycom LoPy 4	STM32 B-L072Z-LRWAN1
Price [R]	R876	R802,70	R817,47
Module	Murata CMWX1ZZABZ	Semtech SX1276	CMWX1ZZABZ-091 LoRa®/Sigfox™ module (Murata)
CPU	SAMD21 Cortex-M0+ 32bit low power ARM	Xtensa® dual-core 32-bit LX6	STM32L072CZ Series
Frequency Range (MHz)	433/868/915	137–1020	860-930
Communication Protocols	UART, I2C and SPI	UART, I2C and SPI	LP-UART, I2C and SPI
Flash Memory	256 kB	4 MB	192 kB
ADC	7(12 bit)	8(12 bit)	4(12-bit)
Digital Output	8	18	6
Power Supply	3.3V	3,5V - 5,5V	1.65 V to 3.6 V
Current Consumption	7 mA per I/O pin	19,5 mA - 107 mA	Low RX current of 10 mA, 200 nA register retention
Weight	32g	31g	126g

C-2 – Pin Layout of LoPy 4



22/03/18

Appendix D – Water Sensor Expansion Board 1.1 Pictorial Diagram



Appendix E – Sensor Product Sheets

E-1 – DS18B20 Sensor

DS18B20 Waterproof Temperature Sensor Cable



HK Shan Hai Group Limited
Room 620, Yutian building, songling road, Futian district, Shenzhen

Product Description

This Maxim-made item is a digital thermo probe or sensor that employs DALLAS DS18B20. Its unique 1-wire interface makes it easy to communicate with devices. It can convert temperature to a 12-bit digital word in 750ms (max). Besides, it can measure temperatures from -55°C to +125°C (-67F to +257F). In addition, this thermo probe doesn't require any external power supply since it draws power from data line. Last but not least, like other common thermo probe, its stainless-steel probe head makes it suitable for any wet or harsh environment.

The datasheet of this DS18B20 Sensor can be found from:
<https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Temp/DS18B20.pdf>

Feature:

Power supply range:	3.0V to 5.5V
Operating temperature range:	-55°C to +125°C (-67F to +257F)
Storage temperature range:	-55°C to +125°C (-67F to +257F)
Accuracy over the range of 10°C to +85°C:	±0.5°C
3-pin 2510 Female Header Housing	
Waterproof Stainless steel sheath	
Stainless steel sheath	
Size of Sheath:	6*50mm
Connector:	RJ11/RJ12, 3P-2510, USB.

Pin Definition:	RED: VCC Yellow: DATA Black: GND
Cable length:	1meter, 2m, 3m, 4m are available upon request.

Application:

The DS18B20 Digital Temperature Probe provides 9-to-12-bit (configurable) temperature readings which indicate the temperature of the device. Information is sent to/from the DS18B20 over a 1Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

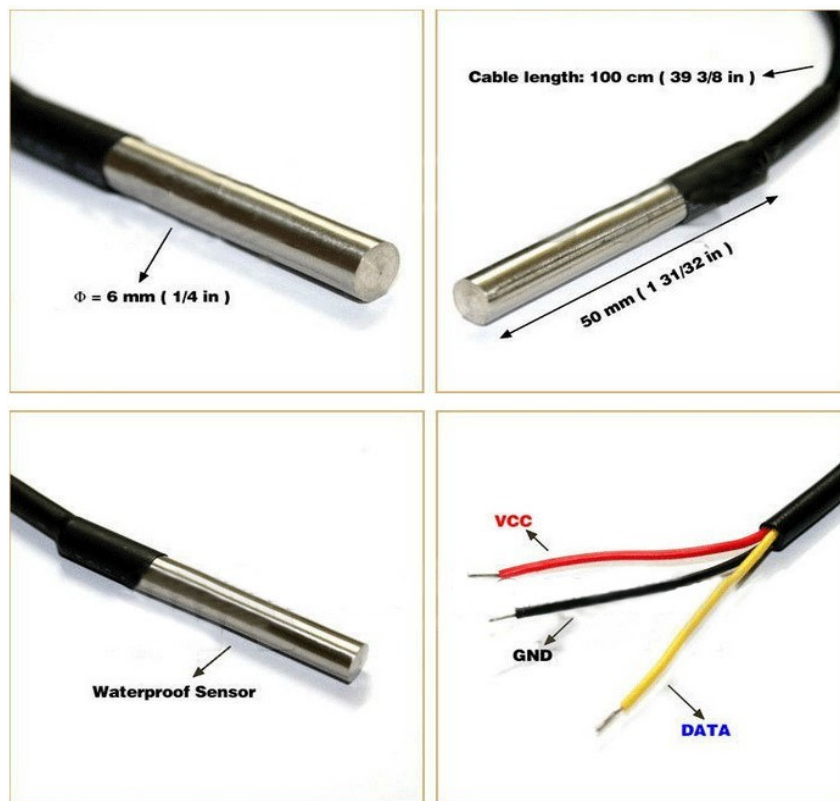
Because each DS18B20 contains a unique silicon serial number, multiple DS 18B20s can exist on the same 1Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.



Details :



Quick-teck Electronics Components datasheet



E-2 – DF Robot SHT31 Temperature & Humidity sensor

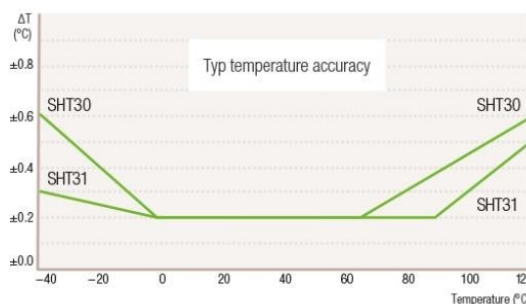
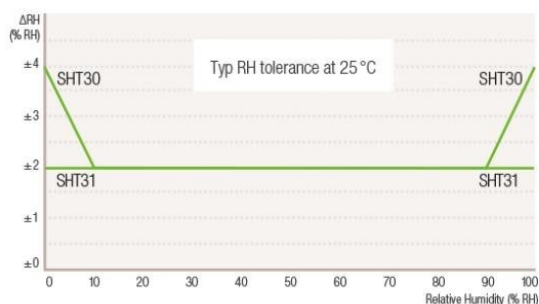


Introduction

This is a SHT31 Temperature & Humidity sensor housed in a weather-proof enclosure. The sensor is designed to be waterproof so you can use it safely in outdoor projects. Please note that although the sensor is waterproof, it is not suggested to submerge it in water. For in-water measurement, [DS18B20](#) Sensor is highly recommended.

Thanks to Sensirion's CMOSens® technology, highly integrated capacitive humidity sensing components and band-gap temperature sensing components, the SHT31 offers high reliability and long-term stability with low power consumption, fast response and strong anti-interference ability. The sensor supports I2C communication, and is compatible with 3.3V/5V controllers like Arduino, micro:bit, ESP32. SHT31 is the standard version of the SHT3x series. It provides humidity accuracy $\pm 2\%RH@0\% RH \sim 100\% RH$ (at $25^{\circ}C$), and temperature accuracy $\pm 0.2^{\circ}C@0^{\circ}C \sim 90^{\circ}C$ (typical).

Note: Although the temperature measurement range of the sensor is $-40 \sim 125^{\circ}C$, we recommend not to place the sensor at a temperature higher than $80^{\circ}C$ since its shell is made of PE material.



Specification

- Operating Voltage: 3.3~5V
- Operating Current: <1.5mA
- Humidity Detection Range: 0%RH~100%RH
- Humidity Accuracy: $\pm 2\%$ RH
- Temperature Detection Range: $-40^{\circ}\text{C} \sim 125^{\circ}\text{C}$
- Temperature Accuracy: $\pm 0.2^{\circ}\text{C}$
- Communication: I2C
- Cable Length: about 1m

PinOut

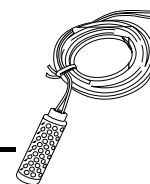


Color	Name	Description
Red	VCC	+
Black	GND	-
Green	SDA	Data line
Yellow	SCL	Clock line

E-3 – Irrrometer Watermark Soil Moisture sensor

Soil Moisture Sensor

For Wireless Soil Moisture/Temperature Station (# 6345)



6440

Vantage Pro² Accessories

The Watermark[®] Soil Moisture Sensor is an indirect, calibrated method of measuring soil water content. It is an electrical resistance type sensor. The Soil Moisture/Temperature Station converts the electrical resistance reading from the sensor into a calibrated reading of centibars of soil water suction with a range from 0 (wettest) to 200 (driest) centibars.

The Watermark Soil Moisture Sensor is a product of the Irrrometer Company, Inc.

General

Sensor Type	Electrical resistance
Attached Cable Length	15' (4.6 m)
Cable Type	Watermark standard, two-wire, wires stripped and tinned
Maximum Cable Length	800' (243 m) (18 AWG UF recommended)
Housing Dimensions	7/8" diameter x 3" (22 mm diameter x 76 mm)
Weight	3.6 oz. (103 g)

Sensor Output

Resolution	1 cb
Range	0 to 200 cb
Update Interval	77 to 90 seconds

Package Dimensions

Product #	Package Dimensions (Length x Width x Height)	Package Weight	UPC Codes
6440	6.00" x 5.63" x 1.25" (152 mm x 143 mm x 32 mm)	9 oz. (.3 kg)	011698 00330 3

DAVIS

Davis Instruments

3466 Diablo Ave., Hayward, CA 94545-4873
(510) 732-9229 FAX (510) 670-0589 sales@davisnet.com www.davisnet.com

DS6440

/Rev

1

F

Transition to novel Internet of Things technology for measurement in hydrology – case study in Cape Town, South Africa

E-4 – Hafei WNK8010-TT Submersible Level Transmitter



WNK8010-TT Submersible Level Transmitter

Features

Wide range application, using high reliable and stable oil filled diffused silicon sensor.

Full stainless steel sealed construction.

IP68 protection.

High precision, high frequency response, long term stability.



Introduction

WNK8010-TT water level transmitter adopts high performance diffused silicon pressure sensor as the measuring element, which is high reliability, high accuracy and small in volume. It is widely used in measurement of liquid pressure and height in small inside diameter pump as well as level height of liquid reservoir, such as water, oil and mild corrosive liquid. The cable which has vented tube in connects with housing sealed, making sensor connecting to the atmosphere to make sure measurement precise.

Specifications

Pressure range: 0-1M- 200M H₂O

Max overload: 150% of sensor range

Burst pressure: 300% of sensor range

Output: 0.5-4.5V, I2C, 1-5V, 4-20mA, 0-5V, 0-10V

Power supply: 5V, 3.3-5V, 12-30V, 10V, 24V

Accuracy: 0.5%FS/year

Stability: 0.25%FS/year

Housing material: Stainless steel 304

Cable material: Polyethylene, PTFE

Ambient temperature: -20~85°C

Process temperature: -20~85°C

Compensation temperature: -20~85°C EMC standard Electromagnetic radiation: EN50081-1/-2

Electromagnetic susceptibility: EN50082-2

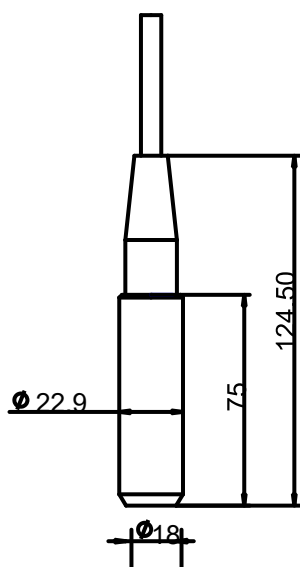
Protection class: IP68 Cable diameter: 1.4mm/wire, 7.4mm for outside diameter

Insulation resistance: 50MΩ Polarity protection in the electronic

Circuit: Yes



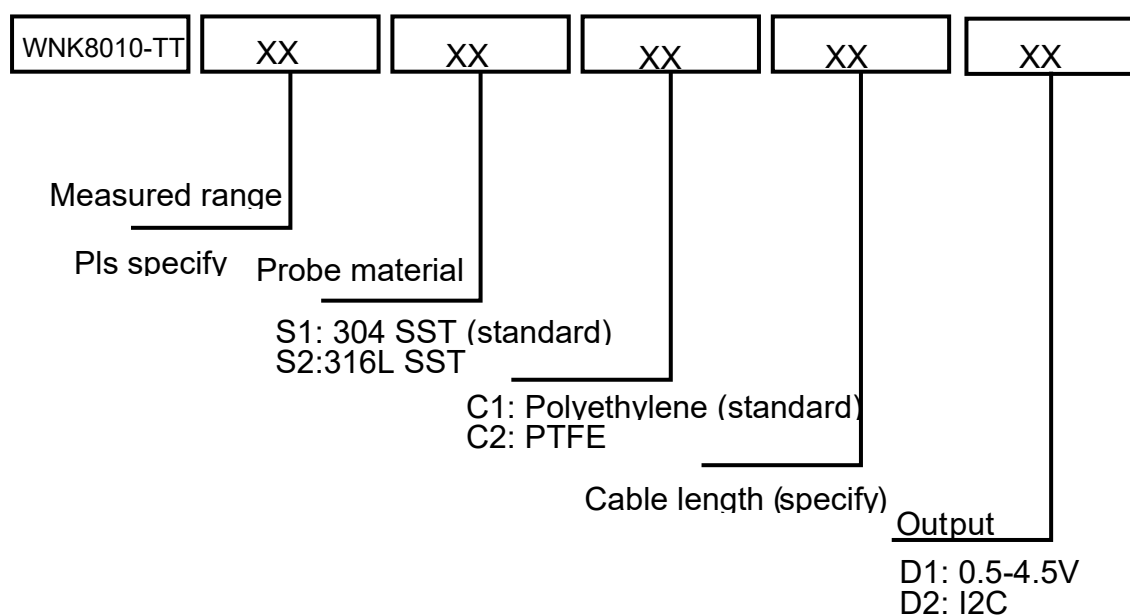
Dimension



Wire Connection

4-20mA	red +	blue -		
V output	red +	blue -	yellow signal	
I2C	red VCC	yellow SCL	blue GND	black SDA

Ordering Code



www.wnksensor.com | Email: sales@wnksensor.com | Office: +86 551 65360650

Appendix F – Data on Relationship between Voltage Output and Depth

Table F-1: Extract of Data displaying Relationship between Voltage Output and Depth

Voltage (Volts)	Depth (m)
0.33	0.00
0.34	0.02
0.35	0.04
0.36	0.06
0.37	0.08
0.38	0.09
0.39	0.11
0.4	0.13
0.41	0.15
0.42	0.17
0.43	0.19
0.44	0.21
0.45	0.23
0.46	0.25
0.47	0.27
0.48	0.28
0.49	0.30
0.5	0.32
0.51	0.34
0.52	0.36
0.53	0.38
0.54	0.40
0.55	0.42
0.56	0.44
0.57	0.45
0.58	0.47
0.59	0.49
0.6	0.51
0.61	0.53
0.62	0.55

Appendix G - Detailed drawing of Radiation Shield

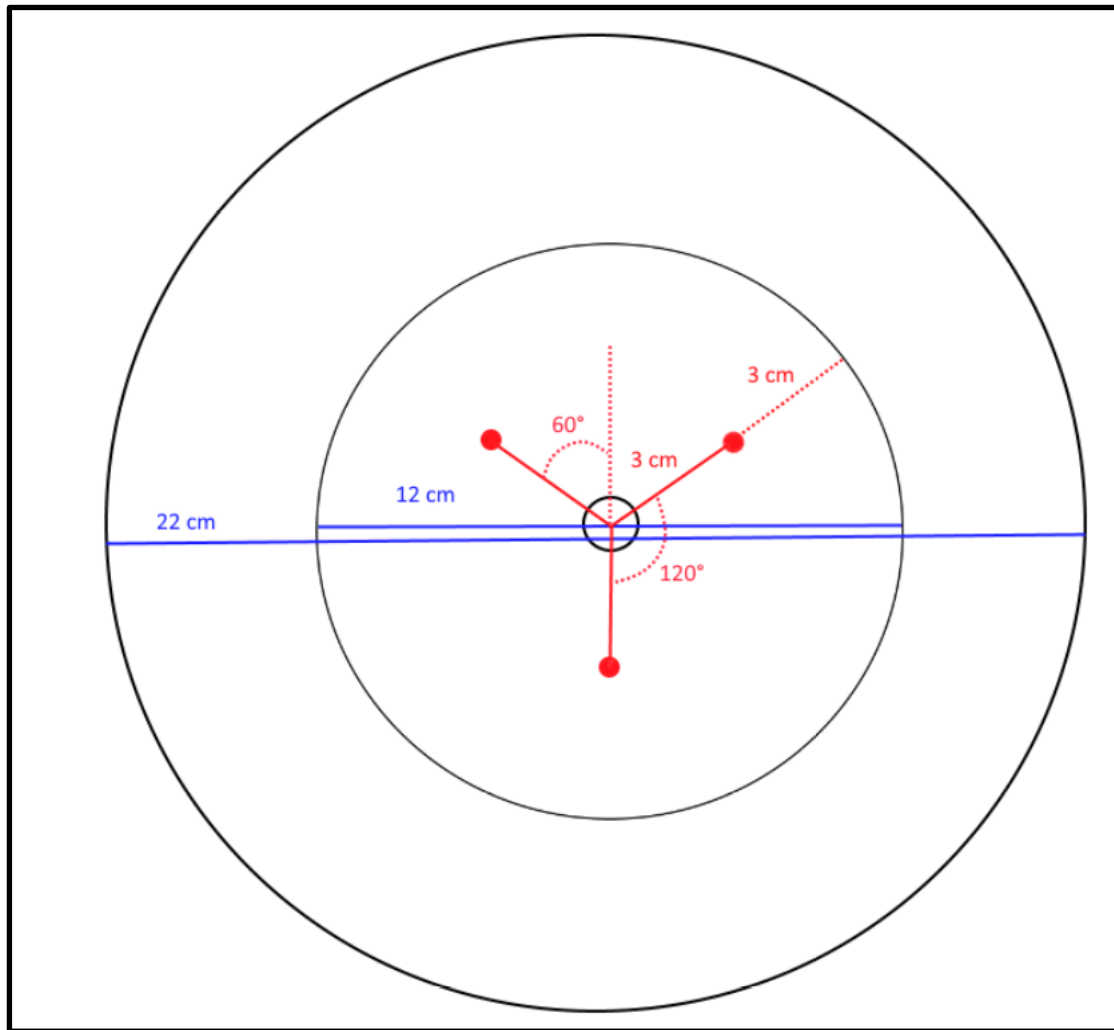


Figure F-1: Detailing of Radiation Shield

Appendix H – Diagrams of Setup for Power Consideration Testing

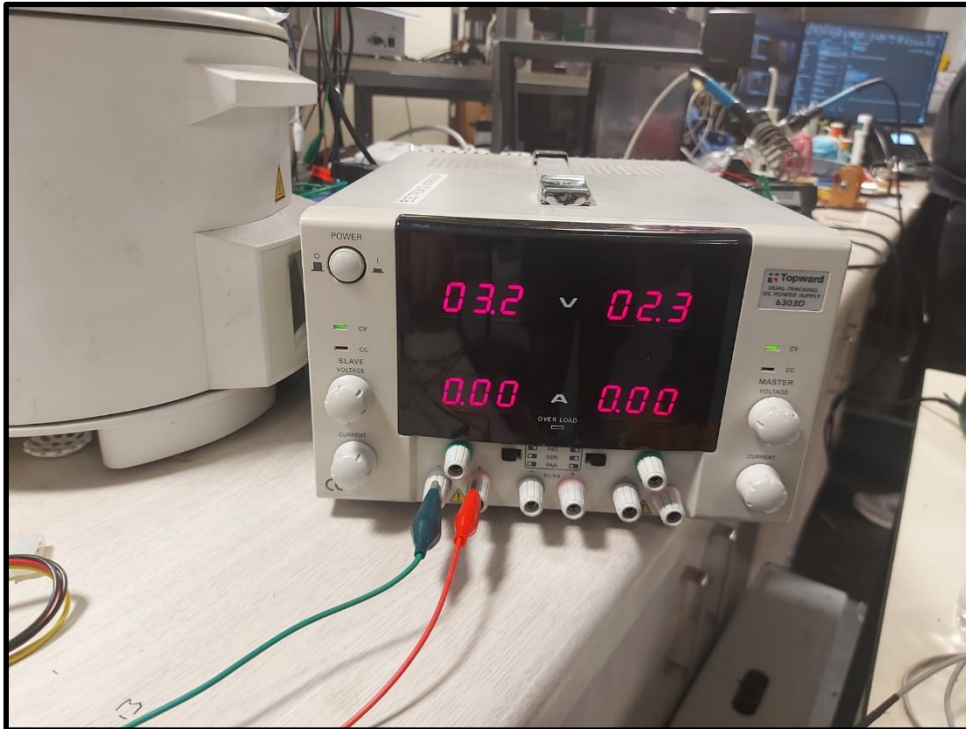


Figure H-1: Power Supply used during Power Consideration Testing

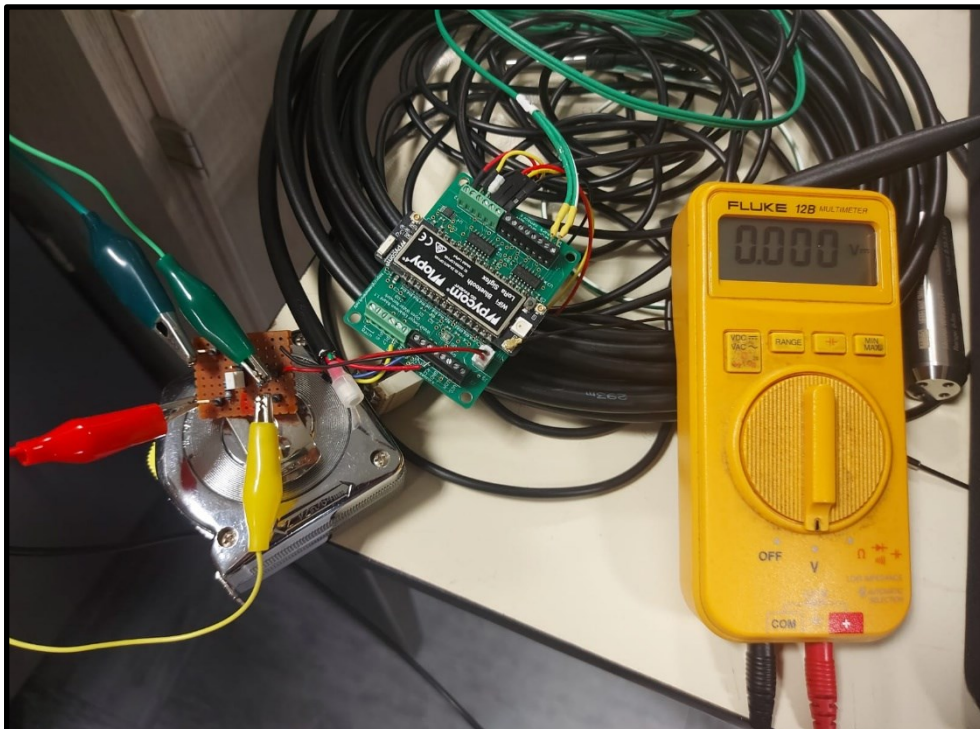


Figure H-2: Setup for Power Consideration Testing

Appendix I – Micro Python Script for Measuring Battery Voltage

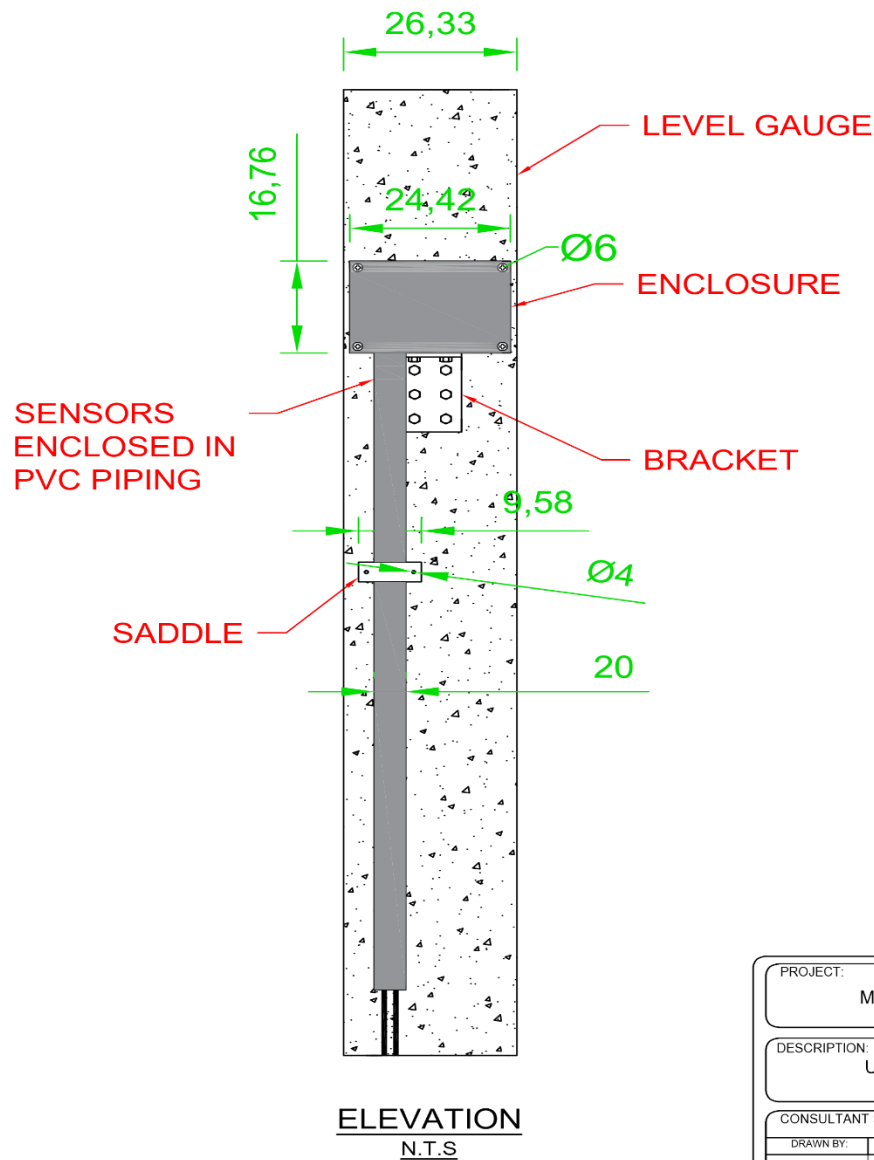
```

# Measurement of Vbat
# *****
# using built-in ADC calibration apin.value_to_voltage() and adjusted vref
of ADC
attn=2 # chose attenuation of ADC
# attn=0 (default) is 0 dB (0-1.1V) with internal reference voltage of ADC
being (about) 1.1V
# attn=1 is 2.5 dB          (0-1.5V)
# attn=2 is 6 dB           (0-2.2V)
# attn=3 is 11 dB          (0-3.9V)
bits=12 #'Bits' can take integer values between 9 and 12 and selects the
number of bits of resolution of the ADC
adc = machine.ADC(bits=bits) # create an ADC object and specify
resolution.
# Adjust vref of ADC with the value stored in NVRAM memory if available
(see script in 'Calibrate ADC' script folder)
try:
    vrefmV=pycom.nvs_get('vref_mV')
except Exception as e:
    vrefmV = None
if vrefmV==None : # when no value is stored
    adc.vref(1100) # new functionality in ADC, works in release
'1.10.2.b1' but not in older releases!
    print("ADC vref set to default value 1100 mV because no value was
stored in NVRAM memory of board")
elif vrefmV<900 or vrefmV>1300 : # when value is out of range
    adc.vref(1100) # same
    print("ADC vref set to default value 1100 mV")
    print("because vrefmV from NVRAM is equal to %.0f" % vrefmV,"mV (that
is outside [900,1300])")
else :
    adc.vref(vrefmV) # same
    print("ADC vref set to %.0f" % vrefmV,"mV (data that was stored in
NVRAM of board)")
    apin = adc.channel(pin='P18',attn=attn) # Create an analog pin on P18,
that is the pin for reading VBatt
    nSamples=100
    ADCreading,Voltage=adcR.adcRead(apin,bits,attn,nSamples) # read ADC and
convert reading to voltage
    batteryVoltage=(Voltage-0.002)*3.060 # correct for the voltage divider,
the 0.002V is the offset voltage on P20
    if printForDebugging is True:
        print("ADCreading = %.4f" % ADCreading, "    Battery voltage = %.3f"
% batteryVoltage

```

Appendix J – Node Experimental Field Setups

J-1 – WMN Setup



PROJECT:

MSc. ENG THESIS

DESCRIPTION:

UCT DAM SENSOR
SETUP

CONSULTANT SIGNATURE:

DRAWN BY:

CHECKED BY:

DESIGNED BY:

ROBERTO

DLVROB001

DE OLIVEIRA

SCALE:

N.T.S

DATE:

OCT. 2022

APPROVAL

CONSTRUCTION

TENDER

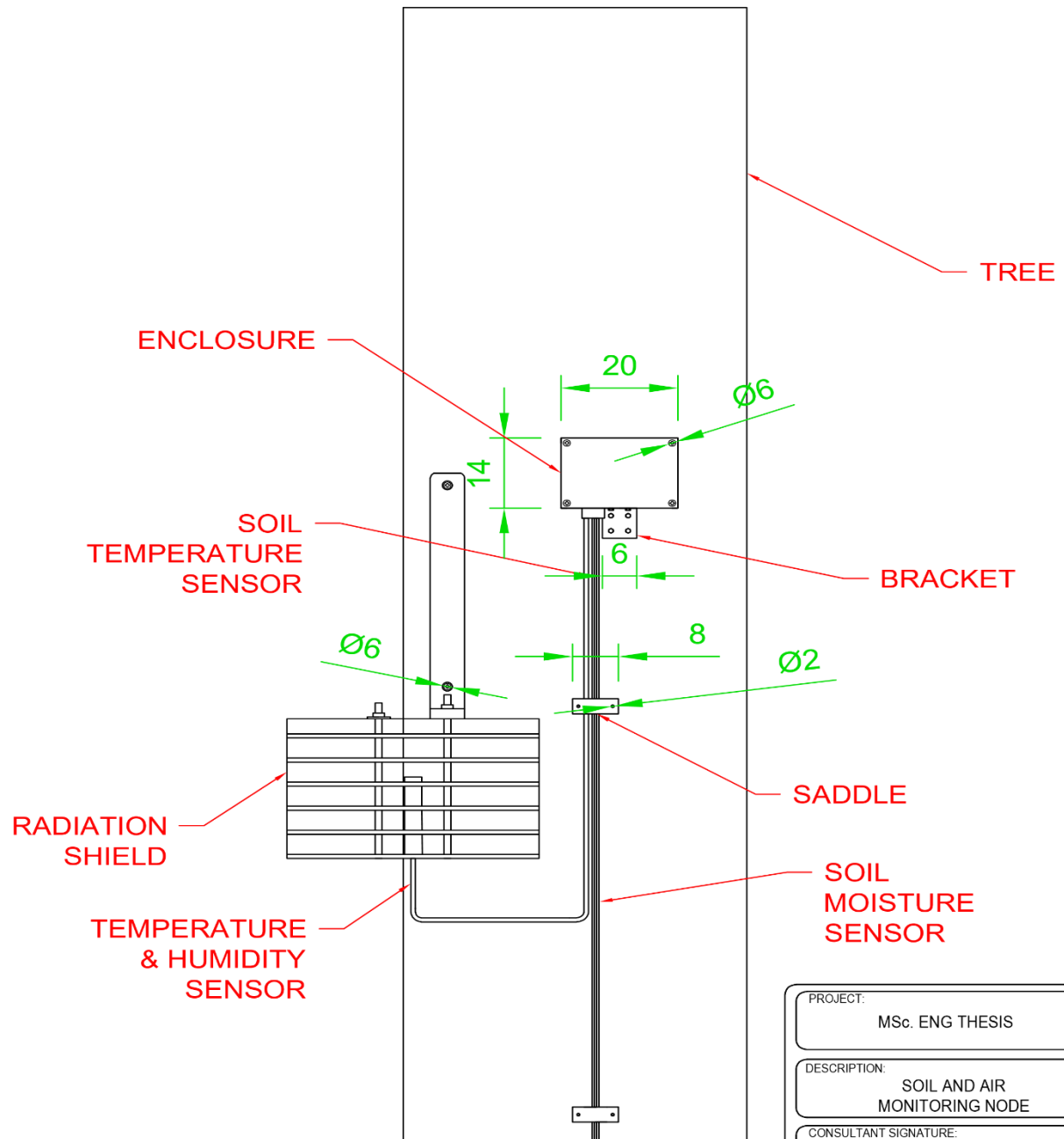
RECORD

X

CONSULTANT DRAWING NUMBER:

CIV5000W - DWG01

J-2 – SAMN Setup



PROJECT:		
MSc. ENG THESIS		
DESCRIPTION:		
SOIL AND AIR MONITORING NODE		
CONSULTANT SIGNATURE:		
DRAWN BY:	CHECKED BY:	DESIGNED BY:
ROBERTO DE OLIVEIRA		DLVRO8001
SCALE:	N.T.S	DATE: OCT. 2022
APPROVAL:	CONSTRUCTION:	
TENDER:	RECORD:	X
CONSULTANT DRAWING NUMBER:		
CIV5000W - DWG02		

*

Appendix K – Accuracy and Precision Testing

K-1 – Setups of the Accuracy and Precision Testing

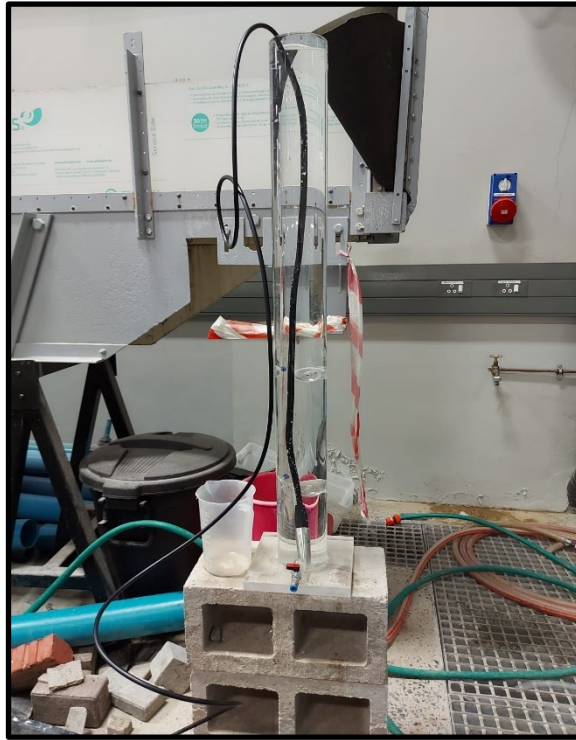


Figure K-1: Setup for Accuracy and Precision Testing for Water Level Sensor



Figure K-2: Setup for Accuracy and Precision Testing for DS18B20

K-2 – Results from the Accuracy and Precision Testing

Table K-1: Results Extracted from Accuracy testing of Water Temperature Sensor

Iteration	DS18B20 - Water	Center 307 Type K	Difference
1	26	26	0
2	27.2	27.2	0
3	28.1	28	0.1
4	27.9	27.8	0.1
5	30.4	30.4	0
6	31.8	31.7	0.1
7	33	33	0
8	34.2	34.1	0.1
9	33.6	33.5	0.1
10	32.7	32.7	0
11	32	32	0
12	31.5	31.6	0.1
13	30.8	30.8	0
14	30.2	30.2	0
15	29.6	29.5	0.1
16	29	29	0
17	28.5	28.5	0
18	28.1	28.1	0
19	27.9	27.9	0
20	27.6	27.6	0
21	27.3	27.4	0.1
22	27.2	27.2	0
23	26.5	26.5	0
24	26	26	0
25	25.6	25.6	0
26	25.4	25.5	0.1
27	18.2	18.2	0
28	20.4	20.5	0.1
29	22.3	22.3	0
30	25.1	25.1	0

Table K-2: Results Extracted from Accuracy testing of Soil Temperature Sensor

Iteration	DS18B20 - Soil	Center 307 Type K	Difference
1	23.70	24.00	0.3
2	23.70	24.00	0.3
3	29.40	29.70	0.3
4	32.00	32.30	0.3
5	32.70	33.00	0.3
6	31.60	31.80	0.2
7	31.30	31.60	0.3
8	31.10	31.40	0.3
9	30.90	31.20	0.3
10	30.70	31.00	0.3
11	30.60	30.80	0.2
12	30.30	30.50	0.2
13	30.10	30.40	0.3
14	29.90	30.20	0.3
15	29.70	29.90	0.2
16	29.50	29.70	0.2
17	29.60	29.90	0.3
18	29.10	29.40	0.3
19	28.50	28.90	0.4
20	28.40	28.80	0.4
21	27.80	28.20	0.4
22	27.00	27.40	0.4
23	27.00	27.30	0.3
24	26.70	27.10	0.4
25	22.80	23.00	0.2
26	24.10	24.40	0.3
27	25.70	26.00	0.3
28	27.00	27.30	0.3
29	28.10	28.40	0.3
30	29.10	29.40	0.3

Table K-3: Results Extracted from Accuracy testing of Water Level Sensor

Iteration	Sensor	Manual	Difference
1	909	913	4
2	841	845	4
3	841	846	5
4	792	800	8
5	708	711	3
6	606	610	4
7	492	500	8
8	348	354	6
9	182	185	3
10	114	117	3
11	72	75	3
12	34	34	0
13	566	570	4
14	909	911	2
15	852	859	7
16	795	795	0
17	757	755	2
18	701	701	0
19	644	642	2
20	549	544	5
21	492	501	9
22	474	474	0
23	456	450	6
24	417	418	1
25	530	534	4
26	492	500	8
27	473	471	2
28	455	460	5
29	417	425	8
30	852	860	8

Table K-4: Results Extracted from Precision testing of Water Temperature Sensor

Iteration	DS18B20 - Water	Center 307 Type K	Difference
1	23.3	23.2	0.1
2	23.3	23.3	0
3	23.5	23.4	0.1
4	23.4	23.4	0
5	23.4	23.4	0
6	23.4	23.4	0
7	23.4	23.4	0
8	23.4	23.4	0
9	23.4	23.4	0
10	23.4	23.4	0
11	23.4	23.4	0
12	23.4	23.4	0
13	23.5	23.4	0.1
14	23.4	23.4	0
15	23.4	23.4	0
16	23.4	23.4	0
17	23.4	23.4	0
18	23.5	23.4	0.1
19	23.4	23.4	0
20	23.5	23.4	0.1
21	23.5	23.4	0.1
22	23.4	23.4	0
23	23.5	23.4	0.1
24	23.5	23.4	0.1
25	23.5	23.4	0.1
26	23.4	23.4	0
27	23.4	23.4	0
28	23.4	23.4	0
29	23.4	23.4	0
30	23.4	23.4	0

Table K-5: Results Extracted from Precision testing of Soil Temperature Sensor

Iteration	DS18B20 - Soil	Center 307 Type K	Difference
1	16	16.3	0.3
2	16.1	16.3	0.2
3	16.1	16.3	0.2
4	16.1	16.3	0.2
5	16.1	16.3	0.2
6	16.1	16.3	0.2
7	16.1	16.3	0.2
8	16.1	16.3	0.2
9	16.1	16.3	0.2
10	16.1	16.3	0.2
11	16.1	16.3	0.2
12	16.1	16.3	0.2
13	16.1	16.3	0.2
14	16.1	16.3	0.2
15	16.1	16.3	0.2
16	16.1	16.3	0.2
17	16.1	16.3	0.2
18	16.1	16.3	0.2
19	16.1	16.3	0.2
20	16.1	16.3	0.2
21	16.1	16.3	0.2
22	16.1	16.3	0.2
23	16.1	16.3	0.2
24	16.1	16.3	0.2
25	16.1	16.3	0.2
26	16.1	16.3	0.2
27	16	16.3	0.3
28	16	16.3	0.3
29	16	16.3	0.3
30	16	16.3	0.3

Table K-6: Results Extracted from Precision testing of Water Level Sensor

Iteration	Sensor	Manual	Difference
1	190 mm	190 mm	0
2	190 mm	190 mm	0
3	190 mm	190 mm	0
4	190 mm	190 mm	0
5	190 mm	190 mm	0
6	190 mm	190 mm	0
7	190 mm	190 mm	0
8	190 mm	190 mm	0
9	190 mm	190 mm	0
10	190 mm	190 mm	0
11	190 mm	190 mm	0
12	190 mm	190 mm	0
13	190 mm	190 mm	0
14	190 mm	190 mm	0
15	190 mm	190 mm	0
16	190 mm	190 mm	0
17	190 mm	190 mm	0
18	190 mm	190 mm	0
19	190 mm	190 mm	0
20	190 mm	190 mm	0
21	190 mm	190 mm	0
22	190 mm	190 mm	0
23	190 mm	190 mm	0
24	190 mm	190 mm	0
25	190 mm	190 mm	0
26	190 mm	190 mm	0
27	190 mm	190 mm	0
28	190 mm	190 mm	0
29	190 mm	190 mm	0
30	190 mm	190 mm	0

Table K-7: Results Extracted from Accuracy testing of Air Temperature Sensor

Iteration	DF Robot - SHT 31	Center 307 Type K	Difference
1	18.02	18	0.02
2	18.06	17.94	0.12
3	18.02	18	0.02
4	17.96	18.06	0.1
5	18.16	18.13	0.03
6	18.07	18.03	0.04
7	18.16	18.14	0.02
8	18.69	18.74	0.05
9	17.69	17.62	0.07
10	17.25	17.21	0.04
11	17.06	16.96	0.1
12	16.88	16.78	0.1
13	16.75	16.61	0.14
14	16.63	16.54	0.09
15	16.44	16.33	0.11
16	16.25	16.17	0.08
17	16.13	16.06	0.07
18	16	15.93	0.07
19	15.94	15.85	0.09
20	15.81	15.76	0.05
21	15.75	15.69	0.06
22	15.63	15.59	0.04
23	15.63	15.55	0.08
24	15.31	15.35	0.04
25	15.31	15.28	0.03
26	15.31	15.29	0.02
27	15.31	15.29	0.02
28	15.31	15.25	0.06
29	15.25	15.21	0.04
30	15.25	15.25	0

Table K-8: Results Extracted from Precision testing of Air Temperature Sensor

Iteration	DF Robot - SHT 31	Center 307 Type K	Difference
1	23	23	0
2	23.2	23.1	0.1
3	23.4	23.4	0
4	23.5	23.4	0.1
5	23.5	23.4	0.1
6	23.5	23.4	0.1
7	23.5	23.4	0.1
8	23.4	23.4	0
9	23.4	23.4	0
10	23.4	23.4	0
11	23.4	23.4	0
12	23.4	23.4	0
13	23.4	23.4	0
14	23.4	23.4	0
15	23.4	23.4	0
16	23.4	23.4	0
17	23.4	23.4	0
18	23.4	23.4	0
19	23.4	23.4	0
20	23.4	23.4	0
21	23.4	23.4	0
22	23.4	23.4	0
23	23.4	23.4	0
24	23.5	23.4	0.1
25	23.4	23.4	0
26	23.4	23.4	0
27	23.4	23.4	0
28	23.4	23.4	0
29	23.4	23.4	0
30	23.4	23.4	0

Appendix L – Data Extracts from WMN and SAMN

L-1 – Extract of WMN data

created_at	entry_id	Message Number	Battery Voltage (mV)	Sensor Water Level (m)	Adjusted Water Level (m)	Water Temp (°C)	Enclosure Temp (°C)	Enclosure Pressure (hPa)	Enclosure Humidity	Seconds
2022/06/10 12:36		6	21	3.865	0.57	9.150	15.31	24.08	100.9	33.6
2022/06/10 12:51		7	22	3.865	0.57	9.150	15.38	24.96	100.9	30.1
2022/06/10 13:06		8	23	3.862	0.57	9.150	15.56	25.34	100.8	29.4
2022/06/10 13:22		9	24	3.862	0.57	9.150	15.63	25.73	100.8	27.5
2022/06/10 13:37		10	25	3.868	0.57	9.150	15.75	26.18	100.8	28.1
2022/06/10 13:52		11	26	3.862	0.57	9.150	15.63	26.54	100.8	27.2
2022/06/10 14:08		12	27	3.868	0.57	9.150	15.5	26.94	100.7	27.5
2022/06/10 14:38		13	29	3.865	0.57	9.150	15.75	27.44	100.7	25.8
2022/06/10 14:54		14	30	3.868	0.57	9.150	15.69	27.56	100.7	24.2
2022/06/10 15:09		15	31	3.865	0.57	9.150	15.56	27.7	100.7	23
2022/06/10 15:24		16	32	3.865	0.57	9.150	15.44	27.42	100.6	22.6
2022/06/10 15:40		17	33	3.865	0.57	9.150	15.5	26.33	100.6	23.4
2022/06/10 15:55		18	34	3.862	0.57	9.150	15.5	25.98	100.6	22.5
2022/06/10 16:26		19	36	3.865	0.57	9.150	15.44	24.33	100.6	24

L-2 – Extracts of SAMN data

Created at	entry_id	Message Number	Battery Voltage (mV)	Soil Moisture (kPa)	Soil Temperature (°C)	Air Temperature (°C)	Air Humidity	Enclosure Temperature	Seconds
2022/06/26 17:40		27	4.015	10.6	15.19	16.75	76.29	18.42	950
2022/06/26 17:56		28	4.018	10.6	15.31	16.73	77.24	17.52	1877
2022/06/26 18:11		29	4.015	10.5	15.38	16.62	73.13	17.1	2804
2022/06/26 18:27		30	4.018	10.7	15.5	16.35	69.94	16.63	3730
2022/06/26 18:42		31	4.021	10.4	15.56	16.04	70.41	16.22	4657
2022/06/26 18:57		32	4.021	10.3	15.56	15.87	72.41	15.93	5584
2022/06/26 19:13		33	4.021	10.4	15.63	15.91	76.51	15.82	6511
2022/06/26 19:28		34	4.015	10.4	15.69	16.07	76.42	15.91	7437
2022/06/26 19:43		35	4.015	10.9	15.75	16.22	73.9	16	8364
2022/06/26 19:59		36	4.015	10.6	15.81	16.38	78	16.13	9291
2022/06/26 20:14		37	4.024	10.8	15.88	16.54	78.75	16.32	10217
2022/06/26 20:29		38	4.015	10.3	15.94	16.7	80.65	16.49	11144
2022/06/26 20:45		39	4.018	10.4	16	16.79	80.71	16.6	12071
2022/06/26 21:00		40	4.021	10.7	16.06	16.86	81.63	16.71	12998

Appendix M – Daily Min, Max and Averages for the WMN and SAMN

M-1 - Daily Min, Max and Averages for Water Temperature

Dates	Min of Water Temp (°C)	Max of Water Temp (°C)2	Average of Water Temp (°C)
Jun	13,25	15,75	13,90
10-Jun	15,31	15,75	15,57
11-Jun	14,75	15,38	14,97
12-Jun	14,56	15,06	14,80
13-Jun	14,56	14,81	14,72
14-Jun	14,19	14,56	14,50
15-Jun	13,94	14,25	14,14
16-Jun	13,88	14,00	13,94
17-Jun	13,63	14,00	13,81
18-Jun	13,81	13,94	13,88
19-Jun	13,81	13,94	13,85
20-Jun	13,50	13,88	13,65
21-Jun	13,50	13,81	13,63
22-Jun	13,63	13,75	13,66
23-Jun	13,63	13,81	13,70
24-Jun	13,50	13,88	13,64
25-Jun	13,38	13,63	13,49
26-Jun	13,31	13,50	13,41
27-Jun	13,25	13,44	13,36
28-Jun	13,38	13,44	13,39
29-Jun	13,38	13,50	13,43
30-Jun	13,38	13,50	13,45
Jul	13,19	13,63	13,38
01-Jul	13,44	13,56	13,49
02-Jul	13,56	13,63	13,63
15-Jul	13,38	13,38	13,38
16-Jul	13,38	13,44	13,40
17-Jul	13,44	13,56	13,45
18-Jul	13,44	13,50	13,48
19-Jul	13,44	13,56	13,50
20-Jul	13,50	13,63	13,55
21-Jul	13,31	13,56	13,41
22-Jul	13,25	13,38	13,35
23-Jul	13,31	13,38	13,37
24-Jul	13,25	13,44	13,32
25-Jul	13,19	13,31	13,27
26-Jul	13,19	13,31	13,24
27-Jul	13,19	13,25	13,25

28-Jul	13,25	13,38	13,26
29-Jul	13,25	13,38	13,30
30-Jul	13,31	13,38	13,34
31-Jul	13,31	13,44	13,38
Aug	13,13	13,50	13,33
01-Aug	13,25	13,50	13,35
02-Aug	13,13	13,38	13,26
03-Aug	13,25	13,31	13,25
04-Aug	13,25	13,38	13,29
05-Aug	13,25	13,38	13,33
06-Aug	13,31	13,50	13,40
07-Aug	13,38	13,50	13,46
Grand Total	13,13	15,75	13,60

M-2 – Daily Min, Max and Averages for the Soil Moisture and Temperature

Row Labels	Min of Soil Moisture (kPa)	Max of Soil Moisture (kPa)	Average of Soil Moisture (kPa)	Min of Soil Temperature (°C)	Max of Soil Temperature (°C)	Average of Soil Temperature (°C)
Jun	10.30	36.70	31.04	14.75	17.94	15.73
26-Jun	10.30	10.90	10.55	15.19	16.50	15.91
27-Jun	11.10	30.10	24.33	14.75	17.94	15.11
28-Jun	29.90	33.40	31.91	15.25	15.69	15.40
29-Jun	32.90	35.00	33.91	15.69	16.13	15.84
30-Jun	34.60	36.70	35.46	16.13	16.44	16.24
Jul	9.80	37.80	22.24	13.63	16.69	14.57
01-Jul	36.10	37.80	36.82	16.44	16.69	16.54
05-Jul	23.30	24.10	23.68	15.13	15.19	15.13
06-Jul	25.50	26.60	26.03	14.94	15.00	14.98
13-Jul	11.10	15.50	13.42	13.81	14.31	14.18
14-Jul	9.90	12.50	11.02	13.63	14.38	13.94
15-Jul	12.50	16.30	14.45	14.13	14.44	14.26
16-Jul	16.30	18.90	17.67	14.38	15.00	14.55
17-Jul	18.80	20.70	19.81	15.00	15.25	15.11
18-Jul	20.60	22.10	21.36	15.13	15.25	15.23
19-Jul	22.00	23.30	22.64	14.69	15.13	14.86
20-Jul	23.10	24.30	23.73	14.38	14.69	14.50
21-Jul	24.10	25.10	24.65	13.94	14.44	14.15
22-Jul	25.00	25.90	25.48	14.06	14.38	14.19
23-Jul	25.80	26.60	26.21	14.19	14.38	14.33
24-Jul	26.50	27.10	26.75	13.88	14.19	13.98
25-Jul	26.80	27.40	27.14	13.81	14.25	14.00
26-Jul	27.30	28.00	27.62	14.13	14.69	14.34

27-Jul	27.80	28.60	28.19	14.50	14.75	14.63
28-Jul	28.40	29.20	28.80	14.63	15.13	14.79
29-Jul	9.80	29.40	16.97	14.63	15.13	14.85
30-Jul	10.50	12.80	11.84	14.38	14.69	14.49
31-Jul	12.60	13.30	12.89	13.88	14.44	14.07
Aug	10.20	30.40	18.76	12.69	14.69	13.69
01-Aug	13.30	14.30	13.70	13.50	13.88	13.65
02-Aug	14.10	15.40	14.78	13.38	13.75	13.55
03-Aug	15.30	16.70	15.99	13.75	14.06	13.88
04-Aug	16.50	17.90	17.22	13.94	14.06	13.99
05-Aug	17.80	18.80	18.40	13.88	14.13	13.99
06-Aug	18.60	18.90	18.75	14.00	14.25	14.11
07-Aug	18.80	19.70	19.21	14.06	14.38	14.20
08-Aug	19.60	20.60	20.08	14.25	14.50	14.34
09-Aug	20.50	22.00	21.28	14.31	14.69	14.49
10-Aug	10.20	22.30	12.99	12.69	14.69	13.58
11-Aug	11.10	13.70	12.42	13.38	13.81	13.61
12-Aug	12.80	13.90	13.27	13.69	13.81	13.77
13-Aug	13.70	15.80	14.83	13.38	13.81	13.52
14-Aug	14.80	15.60	15.02	13.19	13.38	13.25
15-Aug	15.70	18.10	16.86	12.75	13.19	12.97
16-Aug	18.20	20.70	19.46	13.00	13.31	13.13
17-Aug	10.30	21.10	13.46	13.00	13.31	13.20
18-Aug	11.50	13.50	12.35	13.13	13.31	13.24
19-Aug	13.60	17.10	15.57	13.19	13.38	13.28
20-Aug	16.70	18.90	17.56	13.13	13.31	13.21
21-Aug	18.90	21.70	20.34	13.13	13.63	13.31
22-Aug	21.60	24.10	22.86	13.31	13.69	13.48
23-Aug	23.90	25.90	24.96	13.50	14.50	13.85
24-Aug	25.80	27.60	26.72	14.50	14.63	14.53
25-Aug	27.40	29.00	28.22	14.38	14.50	14.45
26-Aug	28.90	30.30	29.58	14.00	14.44	14.14
27-Aug	10.40	30.40	17.17	13.69	14.06	13.86
28-Aug	13.60	19.40	16.60	13.56	13.94	13.74
29-Aug	19.40	22.70	21.24	13.38	13.81	13.56
30-Aug	22.80	25.40	24.09	12.94	13.38	13.13
31-Aug	25.40	27.70	26.58	13.06	13.88	13.34
Sep	12.40	55.20	36.17	13.25	16.81	15.10
01-Sep	27.60	29.60	28.62	13.88	14.50	14.22
02-Sep	29.50	31.30	30.43	14.38	14.50	14.42
03-Sep	31.20	32.60	31.93	14.19	14.38	14.30
04-Sep	12.40	33.60	27.88	13.69	14.31	14.06
05-Sep	13.90	17.70	15.82	13.25	13.69	13.42
06-Sep	17.80	22.00	19.95	13.38	13.69	13.47

07-Sep	22.00	25.30	23.73	13.44	14.00	13.66
08-Sep	25.30	27.80	26.56	13.94	14.38	14.09
09-Sep	27.70	29.90	28.79	14.25	14.81	14.43
10-Sep	29.80	31.70	30.74	14.63	14.81	14.71
11-Sep	28.10	31.90	30.08	14.69	14.81	14.74
12-Sep	28.00	29.20	28.43	14.19	14.69	14.37
13-Sep	29.10	30.90	30.02	14.19	14.88	14.43
14-Sep	30.80	32.40	31.65	14.81	15.88	15.16
15-Sep	32.30	34.10	33.17	15.56	16.19	15.84
16-Sep	33.90	35.40	34.73	15.81	16.38	16.09
17-Sep	35.40	37.00	36.23	15.69	16.38	16.01
18-Sep	36.80	38.30	37.58	15.19	15.69	15.33
19-Sep	38.20	39.70	38.98	14.88	15.31	15.06
20-Sep	39.40	40.90	40.23	15.13	15.81	15.39
21-Sep	40.80	42.50	41.71	15.50	16.19	15.79
22-Sep	42.40	44.10	43.26	15.75	16.19	15.89
23-Sep	43.90	45.30	44.60	15.38	15.75	15.50
24-Sep	45.20	46.70	45.93	15.13	15.69	15.41
25-Sep	46.50	48.20	47.34	15.56	16.31	15.83
26-Sep	48.00	50.10	49.08	15.94	16.31	16.09
27-Sep	49.80	50.50	50.17	15.56	16.25	15.87
28-Sep	50.30	51.80	51.04	15.88	16.69	16.19
29-Sep	51.50	53.20	52.44	16.31	16.81	16.54
30-Sep	53.10	55.20	54.14	16.38	16.81	16.59
Oct	54.80	206.00	110.83	16.25	19.38	17.92
01-Oct	54.80	56.50	55.71	16.31	16.69	16.52
02-Oct	56.40	58.50	57.50	16.44	17.19	16.73
03-Oct	58.40	61.00	59.75	16.81	17.19	16.97
04-Oct	60.60	62.90	61.80	16.50	16.88	16.67
05-Oct	62.70	64.90	63.82	16.38	16.94	16.64
06-Oct	64.70	67.40	66.22	16.50	16.88	16.63
07-Oct	67.20	69.20	68.12	16.25	16.75	16.49
08-Oct	69.00	71.70	70.33	16.44	17.00	16.69
09-Oct	71.60	74.40	73.07	16.81	17.31	17.02
10-Oct	74.30	78.00	76.20	17.25	17.75	17.43
11-Oct	77.90	82.20	79.89	17.75	18.19	17.92
12-Oct	81.60	86.00	83.88	17.94	18.38	18.14
13-Oct	85.80	90.50	88.27	18.00	18.31	18.14
14-Oct	90.30	94.20	92.12	17.69	18.00	17.80
15-Oct	93.70	97.20	95.41	17.44	17.75	17.53
16-Oct	96.50	100.00	98.18	17.44	18.44	17.79
17-Oct	99.70	105.90	102.98	18.19	18.50	18.37
18-Oct	105.50	111.40	108.55	18.25	18.69	18.47
19-Oct	111.30	117.10	114.30	18.38	19.00	18.66
20-Oct	117.00	124.30	121.00	18.69	19.00	18.82

21-Oct	124.30	131.40	128.10	18.38	18.81	18.51
22-Oct	131.20	137.00	134.30	18.13	18.63	18.33
23-Oct	136.10	144.10	140.27	18.44	18.94	18.66
24-Oct	143.20	151.30	147.61	18.63	19.00	18.84
25-Oct	151.30	159.60	155.71	18.63	19.00	18.77
26-Oct	159.30	166.30	163.10	18.44	18.94	18.66
27-Oct	166.60	174.30	170.35	18.69	18.94	18.83
28-Oct	173.70	181.60	178.04	18.56	19.19	18.82
29-Oct	180.50	189.60	185.11	19.00	19.38	19.15
30-Oct	188.80	198.60	193.79	18.81	19.31	19.05
31-Oct	197.90	206.00	201.91	18.25	18.81	18.44
Nov	205.40	206.60	206.04	18.38	18.44	18.42
01-Nov	205.40	206.60	206.04	18.38	18.44	18.42
Grand Total	9.80	206.60	49.07	12.69	19.38	15.41

M-3 – Daily Min, Max and Averages for the Air Temperature and Relative Humidity

Row Labels	Min of Air Temperature (°C)	Max of Air Temperature (°C)	Average of Air Temperature (°C)	Min of Air Humidity	Max of Air Humidity	Average of Air Humidity
Jun	15,87	26,10	21,17	24,55	86,02	39,23
26-Jun	15,87	17,28	16,68	69,94	86,02	79,14
27-Jun	16,48	26,10	20,14	34,98	74,55	48,57
28-Jun	16,07	26,05	21,00	24,55	51,98	36,66
29-Jun	16,24	25,21	22,35	30,61	49,95	34,23
30-Jun	18,02	24,63	21,71	29,21	43,93	32,78
Jul	7,48	26,86	14,42	21,42	100,00	69,70
01-Jul	16,05	22,80	20,45	31,75	74,18	43,43
05-Jul	12,33	16,63	15,42	62,04	85,59	66,48
06-Jul	12,53	15,05	13,92	44,17	82,22	60,93
13-Jul	12,39	13,04	12,58	89,11	100,00	97,80
14-Jul	11,38	17,35	13,75	70,18	100,00	91,93
15-Jul	11,38	18,41	14,22	64,00	91,89	79,12
16-Jul	15,45	26,86	21,80	21,61	68,34	39,08
17-Jul	13,76	24,40	18,35	21,42	96,93	57,66
18-Jul	11,94	15,27	12,84	80,18	99,84	94,91
19-Jul	11,53	15,42	12,81	62,19	84,64	75,34
20-Jul	9,87	15,15	11,98	56,24	81,75	72,10
21-Jul	7,48	17,67	11,74	56,71	82,38	71,78
22-Jul	8,65	17,86	12,85	50,68	97,19	79,12
23-Jul	10,68	14,52	12,08	64,20	97,00	81,13
24-Jul	11,33	16,35	13,11	54,97	76,88	68,34
25-Jul	11,60	21,22	15,37	31,83	76,46	49,43
26-Jul	10,99	21,68	16,57	27,46	94,40	52,50
27-Jul	11,70	18,02	14,40	53,70	85,54	71,35
28-Jul	13,33	23,51	17,43	31,87	95,26	55,93
29-Jul	10,20	16,01	12,77	58,61	96,77	80,73
30-Jul	9,16	15,02	11,24	57,18	91,78	80,98
31-Jul	8,43	14,21	11,01	54,65	86,38	73,55
Aug	6,97	26,40	13,09	21,10	100,00	76,15
01-Aug	9,53	15,12	12,12	56,81	79,92	68,55
02-Aug	9,53	16,27	12,60	51,28	92,55	76,23
03-Aug	10,58	15,64	12,93	63,24	93,19	82,86
04-Aug	11,43	14,51	13,16	68,73	91,10	76,87
05-Aug	10,65	15,03	12,57	71,89	99,19	88,73
06-Aug	11,41	16,59	13,76	67,70	100,00	83,21
07-Aug	12,93	19,02	14,79	49,57	88,27	74,73
08-Aug	11,74	18,78	14,46	60,86	89,25	74,95

09-Aug	11,20	20,96	14,10	40,99	94,81	79,29
10-Aug	11,57	12,90	12,29	95,72	99,75	98,40
11-Aug	11,61	15,27	13,14	51,26	100,00	89,36
12-Aug	9,16	14,05	11,46	67,88	97,85	86,00
13-Aug	7,79	14,14	11,30	61,44	91,64	75,47
14-Aug	8,47	13,25	10,66	68,56	96,28	79,33
15-Aug	6,97	16,96	11,60	44,41	80,64	63,81
16-Aug	8,63	15,80	12,24	61,33	91,76	75,91
17-Aug	11,35	14,81	12,71	68,11	99,27	92,67
18-Aug	10,49	12,91	11,88	83,05	97,88	92,72
19-Aug	10,96	13,66	12,21	64,40	94,26	80,63
20-Aug	9,89	14,24	11,99	53,46	86,45	72,33
21-Aug	9,82	16,49	12,19	68,72	89,80	80,49
22-Aug	9,38	16,84	13,57	55,80	89,76	73,68
23-Aug	14,28	26,40	20,47	27,74	58,37	41,57
24-Aug	12,87	22,99	15,40	22,42	91,29	70,18
25-Aug	10,73	15,73	13,31	60,40	95,00	76,10
26-Aug	9,62	14,10	11,60	59,98	94,81	80,36
27-Aug	10,14	17,26	12,20	56,05	98,63	83,82
28-Aug	9,36	16,27	12,23	51,36	85,52	72,49
29-Aug	8,80	14,40	11,40	52,91	97,68	74,73
30-Aug	10,56	17,28	13,35	43,27	69,98	57,50
31-Aug	12,40	25,12	18,00	21,10	59,00	37,76
Sep	8,19	27,77	14,42	18,54	98,77	74,73
01-Sep	13,11	25,43	16,19	18,54	91,60	69,48
02-Sep	12,19	15,76	13,39	64,06	89,26	78,78
03-Sep	10,44	15,71	12,92	69,14	93,81	83,95
04-Sep	9,76	13,99	11,88	66,85	98,77	86,78
05-Sep	9,52	14,61	11,87	61,57	80,36	71,54
06-Sep	8,37	16,33	12,45	53,13	89,19	73,46
07-Sep	10,73	17,57	14,07	55,92	89,91	74,83
08-Sep	11,54	16,99	14,45	65,51	94,49	80,08
09-Sep	12,86	18,75	15,36	57,06	91,14	75,80
10-Sep	12,64	16,36	14,12	77,52	97,79	86,36
11-Sep	11,06	16,00	12,89	64,85	98,67	83,40
12-Sep	9,52	16,51	12,69	52,83	85,75	72,19
13-Sep	13,05	19,15	15,89	51,16	85,03	74,11
14-Sep	14,70	27,77	19,73	35,06	92,78	56,31
15-Sep	13,73	23,27	16,94	61,78	91,23	77,34
16-Sep	11,98	21,03	15,74	52,52	94,55	79,98
17-Sep	10,62	14,96	12,80	65,81	94,33	79,69
18-Sep	8,19	15,51	12,34	55,62	84,93	69,51
19-Sep	10,52	17,44	13,65	51,64	91,98	72,15
20-Sep	12,32	20,68	15,42	49,12	81,96	69,36
21-Sep	11,37	24,17	16,98	22,54	88,39	58,43

22-Sep	11,03	14,75	13,13	46,68	95,73	81,42
23-Sep	9,57	15,54	12,82	52,59	79,58	66,32
24-Sep	8,88	18,98	13,90	43,76	85,05	68,91
25-Sep	11,03	19,29	15,26	57,10	86,73	75,30
26-Sep	12,36	16,65	13,79	54,56	96,65	79,75
27-Sep	11,09	19,86	15,05	42,55	84,66	66,68
28-Sep	11,67	21,67	15,91	45,80	83,93	71,13
29-Sep	13,99	22,55	16,59	50,65	89,35	75,60
30-Sep	10,75	19,05	14,32	59,93	96,55	83,11
Oct	9,83	29,18	17,37	21,34	97,62	73,94
01-Oct	13,00	18,90	15,67	54,51	82,57	68,97
02-Oct	12,47	24,49	17,35	21,34	83,53	64,03
03-Oct	11,10	18,15	14,74	53,52	96,07	79,05
04-Oct	10,98	18,63	14,22	51,84	90,23	70,98
05-Oct	9,83	19,00	14,67	53,01	79,09	66,21
06-Oct	11,53	16,52	14,13	70,98	92,35	85,45
07-Oct	11,58	19,04	15,20	56,52	83,90	72,45
08-Oct	12,70	20,76	16,29	53,80	86,28	76,55
09-Oct	15,51	21,30	17,86	55,34	87,52	75,06
10-Oct	17,93	28,48	22,31	30,86	71,38	53,29
11-Oct	15,19	27,88	19,03	30,91	93,72	77,80
12-Oct	16,03	22,02	17,90	66,97	90,57	81,96
13-Oct	15,40	18,17	16,60	71,35	97,62	84,76
14-Oct	14,49	18,13	15,81	63,93	89,26	80,73
15-Oct	14,85	19,35	16,98	69,95	85,35	78,90
16-Oct	17,07	29,18	21,98	36,00	77,69	60,44
17-Oct	15,39	23,24	18,18	59,32	92,95	77,57
18-Oct	15,76	20,99	18,17	70,96	92,17	80,95
19-Oct	14,34	26,79	19,72	27,72	90,80	72,48
20-Oct	13,14	21,02	16,67	49,90	88,06	73,04
21-Oct	14,63	19,26	16,21	52,91	86,12	75,14
22-Oct	14,49	22,32	18,29	53,76	80,21	68,65
23-Oct	16,36	23,64	18,91	59,47	92,43	79,10
24-Oct	15,94	21,39	18,35	62,60	86,14	78,08
25-Oct	14,17	20,13	16,79	58,56	94,26	74,62
26-Oct	14,89	23,04	18,44	54,45	84,62	72,92
27-Oct	15,23	20,54	17,34	67,44	90,93	80,14
28-Oct	15,40	25,08	19,80	50,39	87,93	69,32
29-Oct	16,42	22,63	18,97	60,50	88,63	72,35
30-Oct	13,82	19,38	16,11	64,10	94,06	76,56
31-Oct	12,79	19,65	15,69	46,16	77,75	64,57
Nov	14,71	15,03	14,87	65,21	68,54	66,28
01-Nov	14,71	15,03	14,87	65,21	68,54	66,28
Grand Total	6,97	29,18	15,07	18,54	100,00	72,88

Appendix N – SAWS Kirstenbosch Weather Station Data Comparison to the WMN and SAMN data

N-1 - Kirstenbosch Rainfall Data

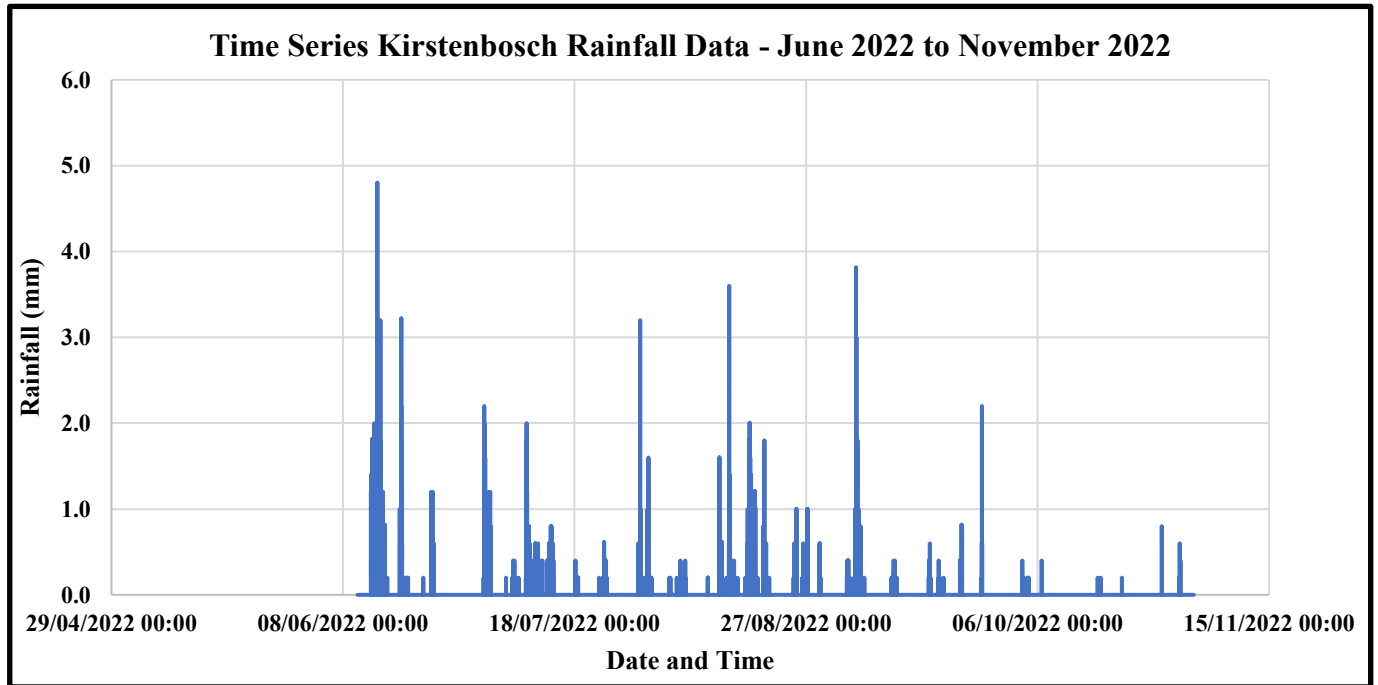


Figure M-1: Time Series of Kirstenbosch Rainfall data collected by SAWS

N-2 - Water Level Data

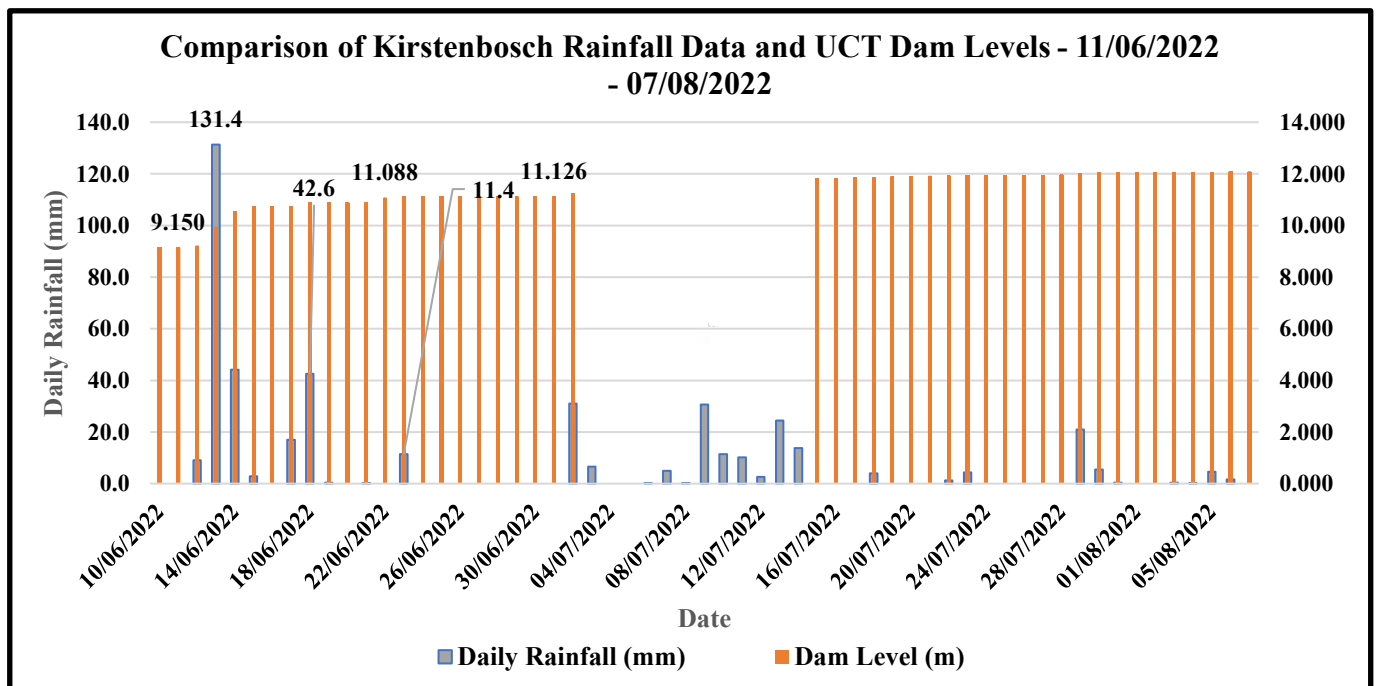


Figure M-2: Comparison between Kirstenbosch Rainfall Data and UCT Dam Levels

N-3 - Soil Moisture Data

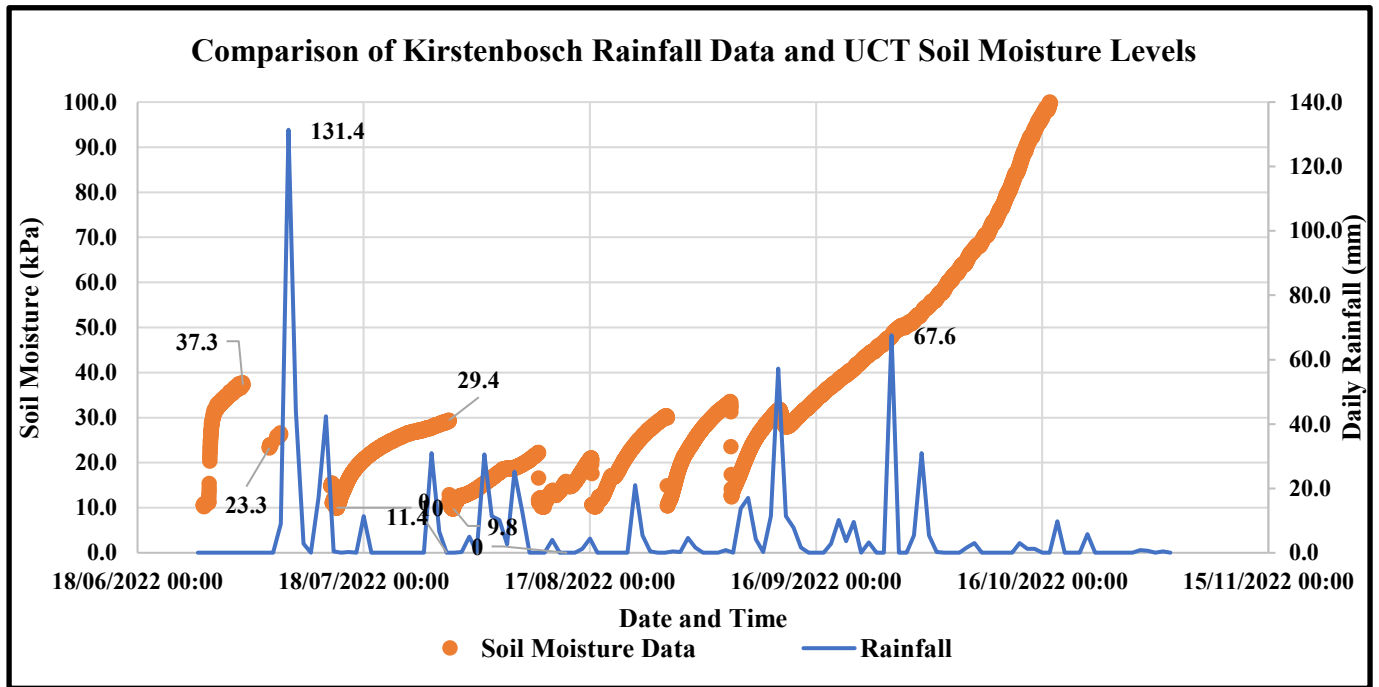


Figure M-3: Comparison between Kirstenbosch Rainfall Data and UCT Soil Moisture

Appendix O – Summary of Enclosure Temperature and Humidity Data Collected by the WMN

Table O-1 Summary Enclosure Temperature and Humidity for WMN1

Row Labels	Average of Enclosure Temp (°C)	Min of Enclosure Temp (°C)	Max of Enclosure Temp (°C)	Average of Enclosure Humidity	Min of Enclosure Humidity	Max of Enclosure Humidity
Jun						
10-Jun	21.01	15.75	27.70	31.94	22.50	40.40
11-Jun	21.38	16.14	25.59	28.97	19.90	50.70
12-Jun	16.62	13.07	20.66	54.53	31.60	89.80
13-Jun	12.04	10.32	13.47	95.44	90.10	98.10
14-Jun	9.93	7.50	11.91	99.12	98.00	100.40
15-Jun	9.37	6.14	17.30	96.38	89.90	100.00
16-Jun	11.63	7.10	18.89	90.20	85.00	94.90
17-Jun	15.05	11.56	19.08	69.89	43.00	89.30
18-Jun	13.29	9.99	18.90	75.95	51.30	94.00
19-Jun	11.97	7.49	19.25	72.95	47.80	87.00
20-Jun	13.60	10.76	20.43	64.75	43.40	75.40
21-Jun	14.61	12.32	18.55	69.15	57.80	81.90
22-Jun	15.36	11.06	25.61	77.01	54.00	86.00
23-Jun	12.33	10.80	16.95	84.09	74.30	90.20
24-Jun	13.20	10.46	18.43	67.56	50.30	80.20
25-Jun	12.73	9.17	19.17	68.84	46.90	78.80
26-Jun	13.21	9.58	18.02	69.48	53.60	81.70
27-Jun	16.34	10.45	27.55	53.64	36.40	67.70
28-Jun	18.51	12.41	27.06	41.05	23.10	54.60
29-Jun	18.91	12.90	26.52	39.70	28.30	52.80
30-Jun	18.30	12.24	25.45	38.40	25.00	54.20
Jul						
01-Jul	18.36	16.65	19.50	37.82	32.70	42.20
02-Jul	12.45	12.05	14.70	88.24	78.60	90.40
Grand Total	14.62	6.14	27.70	66.63	19.90	100.40

Table O-2 Summary Enclosure Temperature and Humidity for WMN2

Row Labels	Average of Enclosure Temp (°C)	Min of Enclosure Temp (°C)	Max of Enclosure Temp (°C)	Average of Enclosure Pressure (hPa)	Min of Enclosure Humidity	Max of Enclosure Humidity
Jul						
15-Jul	16.18	13.31	28.69	101.21	33.40	77.10
16-Jul	17.90	10.35	27.34	100.90	30.70	75.20
17-Jul	17.06	13.65	20.62	100.64	28.30	77.20
18-Jul	12.98	11.44	17.12	100.91	75.90	83.30
19-Jul	13.41	10.72	19.02	101.17	55.60	79.30
20-Jul	11.91	7.72	18.28	101.62	53.90	76.30
21-Jul	11.63	6.48	21.33	101.56	59.40	78.80
22-Jul	12.12	7.19	18.04	101.02	59.40	82.80
23-Jul	12.30	9.72	16.93	101.47	64.10	87.60
24-Jul	13.29	10.33	19.00	101.80	49.70	70.70
25-Jul	14.84	10.88	23.11	101.07	37.20	72.20
26-Jul	15.17	10.56	23.54	100.64	40.40	80.70
27-Jul	14.14	9.16	20.58	100.98	45.30	81.90
28-Jul	15.83	9.08	24.93	100.16	35.70	74.80
29-Jul	12.31	8.26	19.51	100.55	58.80	88.20
30-Jul	10.38	7.15	17.01	101.21	60.40	85.90
31-Jul	10.79	6.50	17.86	101.81	56.90	86.80
Aug						
01-Aug	11.81	7.99	17.99	101.61	52.00	76.60
02-Aug	12.34	7.43	18.55	100.98	60.10	86.60
03-Aug	12.11	9.08	17.17	100.90	70.40	89.70
04-Aug	12.21	10.02	15.37	101.10	67.00	85.00
05-Aug	11.51	9.54	16.39	101.18	76.40	85.40
06-Aug	13.54	9.42	18.08	100.93	59.20	90.10
07-Aug	14.46	12.03	18.27	100.28	58.30	75.80
Grand Total	13.26	6.48	28.69	101.08	28.30	90.10

Appendix P – Detailed Cost Analysis of Sensor Nodes

P-1 – WMN Cost Analysis

Electronic Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
Pycom LoPy 4	1385,74	1	1385,74
Water Sensor Expansion Board 1.1	1204,54	1	1204,54
Antenna - LoRa/Sigfox	271,93	1	271,93
DS18B20 - 5 meters	58,85	1	58,85
Hafei WNK submersible water level sensor	660,08	1	660,08
3.3 V 5000 mAh Li-Po Battery	550,00	1	550,00
Total			4131,14

Enclosure Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
IP56 Weatherproof PVC Enclosure	39,79	1	39,79
1000 mm x 20 mm PVC Pipe	39,00	1	39,00
20 mm Galvanised saddle	2,00	4	8,00
L-Bracket	30,00	2	60,00
Total			146,79

P-2 – SAMN Cost Analysis

Electronic Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
Pycom LoPy 4	1385,74	1	1385,74
Water Sensor Expansion Board 1.1	1204,54	1	1204,54
Antenna - LoRa/Sigfox	271,93	1	271,93
DS18B20 - 5 meters	58,85	1	58,85
Irrrometer Watermark Soil Moisture Sensor	1899,00	1	1899,00
DF Robot SHT31 Sensor	358,30	1	358,30
3.3 V 5000 mAh Li-Po Battery	550,00	1	550,00
Total			5728,36

Enclosure Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
IP56 Weatherproof PVC Enclosure	39,79	1	39,79
1000 mm x 20 mm PVC Pipe	39,00	1	39,00
20 mm Galvanised saddle	2,00	4	8,00
L-Bracket	30,00	2	60,00
16 diameter plastic plates	15,99	6	95,94
15 mm Stainless Steel Rod	15,00	3	45,00
Total			287,73

P-3 – HMN Cost Analysis

Electronic Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
Pycom LoPy 4	1385,74	1	1385,74
Water Sensor Expansion Board 1.1	1204,54	1	1204,54
Antenna - LoRa/Sigfox	271,93	1	271,93
DS18B20 - 5 meters	58,85	2	117,70
Irrrometer Watermark Soil Moisture Sensor	1899,00	1	1899,00
DF Robot SHT31 Sensor	358,30	1	358,30
Hafei WNK submersible water level sensor	660,08	1	660,08
3.3 V 5000 mAh Li-Po Battery	550,00	1	550,00
Total			6447,29

Enclosure Components			
Components	Price per Unit (ZAR)	Quantity	Price (ZAR)
IP56 Weatherproof PVC Enclosure	39,79	1	39,79
1000 mm x 20 mm PVC Pipe	39,00	1	39,00
20 mm Galvanised saddle	2,00	4	8,00
L-Bracket	30,00	2	60,00
16 diameter plastic plates	15,99	6	95,94
15 mm Stainless Steel Rod	15,00	3	45,00
Total			287,73