CO2 BODY WORN LOGGER

Prepared by:

TUMISANG LEQELE

LQLTUM001

Department of Electrical Engineering
University of Cape Town

Prepared for:

SAMUEL GINSBERG

****Department of Electrical Engineering****
University of Cape Town

January 2016

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for MSc in Electrical Engineering

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This final year project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof

Name: **TUMISANG LEQELE**

Signature: Signed by candidate **Date:** 12 June 2016

Terms of Reference

In February 2014, Samuel Ginsberg of the department of Electrical Engineering initiated this thesis together with Prof Robin Wood of the Faculty of Health Sciences, University of Cape Town. The project requires me to design, build and test the CO_2 body worn device (CBWD) and a piece of software to communicate with the device.

The device is equipped with the GPS to identify its current location, and COZir to measure the CO_2 level, the relative humidity and the Temperature. The device also have the SD card for log data storage.

The CBWD software provides the interface for the user to be able to communicate with the device. This includes:

- The user being able to download and delete log data from the Device.
- The user being able to synchronise device date and time with the computer‘s date and time.
- The user being able to set the reference value for the CO_2 level.
- The user being able to graphically see device live results.

The project requires the following skills:

- Electronics design or hardware design, primarily digital
- Programming skills (Especially in C/C++ and Visual basic or C#, Java)

The deliverables of this project are:

- A working device which can provide log data for the CO_2 level, the humidity, Temperature, GPS location coordinates and time
- A piece of software to communicate with the device to download log data, erase log data, synchronize device date with the computer, and zero the device CO_2 level.

Acknowledgements

I would like to express my deep gratitude to Samuel Ginsberg, my thesis supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this research work.

I would also like to extend my gratitude to Prof Robin Wood and Carl Morrow, for guiding me on the project specification and helping me obtain the scholarship for the project.

I would like to thank the Desmond Tutu HIV foundations for the financial support.

I wish to thank my family for their support and encouragement throughout my studies. I would also like to thank all my friends and people around me who helped with this work.

Finally I would like to thank my God for giving me power to finish this thesis.

Abstract

Tuberculosis is the second most killing disease in the world, although cure is being provided for the disease in South Africa, prevention becomes the most important part [1]. This calls for a device which can be used to gather statistical data to be used in verifying where TB transmission mostly takes place, and proper conditions be implemented to minimise the TB transmission.

Carbon dioxide Body Worn Device (CBWD) is a device which relies on the battery supply for its function. Its function includes gathering the CO_2 level, humidity and Temperature as well as the location coordinates with time, and this data stored on the device memory, to allow for downloading, for the user to be able to analyse log data. The device stores its log data on the SD card, and its live results can be viewed using the software or on the device screen.

A way to communicate with the device calls for a piece of software which the user can interact with as to issue commands to the device. CBWD software is a piece of software which provides the interface for the user to be able to interact with the device. It provides an easy user menu for the user to download log data, erase log data, synchronise device date and time, and zero the CO_2 level. The software also provides the storage for last data downloaded for the device.

The system can be improved to integrate with other devices as a control system, for instance on the ventilation system, the device can be integrated to regulate the ventilation rate factors based on the current conditions on the area.

Table of Contents

List of Figures

List of Illustrations

List of Tables

List of Abbreviations

CO_2	Carbon Dioxide
GPS	Global Positioning System
LCD	liquid Crystal Displays
MCU	Microcontroller
NMEA	National Marine Electronics Association
OOP	Object Oriented Programming
SD	Secure Digital
TB	Tuberculosis
WHO	World Health Organisation

1. Introduction

CO_2 Body Worn Device refers to the device that can be able to log on the CO_2 level, temperature and humidity together with location coordinates and time.

1.1 Background to the study

The bacterium that causes Tuberculosis (TB) was discovered in the year 1882 by the German physician and scientist named Robert Kotch. TB is an air borne disease, thus it spread from one person to the other only if one person inhales the air which contains the TB bacteria which has been exhaled by another person. [3]

Although TB cure improves every year, the statistical records show that TB cure has improved from 40% to 70% between the year 1995 and 2011, but TB incidents in South Africa continues to increase. [2]. TB transmission occurs mostly in areas where people are packed and it affects mostly young people in the age range 15 to 25. High schools are identified as the most vulnerable places as young people in the age range 15 to 19 are gathered together for an extensive amount of time.

Carbon dioxide Body worn device is a mobile device that can gather the statistical data in relation to the CO_2 level, the temperature and the relative humidity. And this data can be used in calculation to identify places where TB is mostly being transmitted. The device also provides location coordinates, hence makes it easy to identify and group together the results based on the locations.

1.2 Objectives of this study

1.2.1 Problems to be investigated

The objectives of the report are therefore the following:

- To design build and test CO_2 body worn device which can:
  - Measure Carbon dioxide in the atmosphere
  - Measure temperature and humidity
  - Log on the current location coordinates and time
  - Store log data on a log file

- To build a portable device which can rely on battery supply for at least two days without losing its log data

- To design, develop and test a piece of software to communicate with the CO_2 device, the software should provide the following specification:
  - Be able to automatically connect to CO_2 body worn device
  - Be able to download data from the device
  - Be able to command the device to delete log data
  - Be able to synchronize computer date and time to the device date and time
  - Be able to set reference to the device variable such as zero the CO_2 level.
  - Be able to get live data results from the device
  - Be able to communicate with the previous version of the device(CO_2 device generation one)

- Build the device and the software that is easy to use, thus even unskilled people should be able to use the device as well as its computer interface software.

.

1.2.2 Purpose of the study

The purpose of this study is to design, build and test a portable CO_2 body worn device, which can log on its current location, as well as the CO_2 level, humidity and temperature. This device should be able to store safely its log data on memory card. The device will be supplied from the battery supply and last for at least two days without any failure. The study also includes maintaining the generation one CO_2 devices.

1.3 Scope and Limitations

The scope of this thesis is to initially build the working prototype of the CO_2 Body worn device. Then develop software which will communicate with the devices.

The following limits have been imposed on the project:

- The device will not auto detect number of people within its working environment
- The device may send data at intervals to the server, but the device should not work as a tracking device
- Microcontrollers have limited memory and speed which limits the response of the device.
- The CO_2 transduce and GPS unit has the defined logging accuracy limitations, hence the obtained results cannot be defined as 100% accurate

1.4 1.4 Plan of development

1.4.1 Chapter 2: Literature Review

The project starts with discussing the historical background of Tuberculosis. This includes the TB discovery, TB types and the factors which contribute to the TB transmission. Finally the GPS positioning technology is discussed, including the GPS history and applications. The GPS satellites configuration and, how GPS receiver determines its location will also be included.

1.4.2 Chapter 3: Specification

The device specification will be discussed in details in the chapter, as well as the CBWD software specification.

1.4.3 Chapter 4: High Level System Design

The Device high level system design will be discussed in this chapter, this will include the system main component modules and their interconnection. The components select criteria is also discussed in this chapter.

1.4.4 Chapter 5: Hardware detailed Design

The detailed description of the hardware design will be provided in this chapter. This concentrates on the circuit design and components values and their calculations.

1.4.5 Chapter 6: Hardware assembly

The way to assemble the hardware of the device is provided in this chapter. The guide on how the components should be assembled and soldered will be provided in this chapter

1.4.6 Chapter 7: Graphical User Interface Design

The CBWD software design will be described in details in this chapter. This includes the use case diagrams as well as the code pieces to show the main device functions.

1.4.7 Chapter 8: Firmware

The device MCU programming will be discussed in this chapter. All the program flow diagrams as well as the code functions to perform the main functions of the device will be provided.

1.4.8 Chapter 9: Testing and Results

All the tests performed on the device will be explained in details, and all the results obtained for all the test will be noted.

1.4.9 Chapter 10: Discussion

The results obtained will be analysed in this chapter.

1.4.10 Chapter 11: Conclusion

The discussed results implications will be concluded in this chapter.

1.4.11 Chapter 12: Recommendations

Based on the conclusions reached about the analysis of the results, the recommendations and improvements to the system will be drawn.

2. Literature Review

This chapter establishes the theory behind the transmission of TB. Thus this chapter addresses the environmental factors that contribute to the transmission of TB. Other factors may include the bacteria and the host, but this review is mostly based on the environmental factors. This chapter additionally discusses the definition and historical background of TB.

2.1 Tuberculosis

Tuberculosis (TB) is a multisystem infectious disease caused by Mycobacterium tuberculosis bacteria. [3] Although there are many types of TB, the main two types are classified as active or latent TB. Active TB refers to when the disease is still showing symptoms and can be transmitted to other people, while latent TB refers to when the disease is not producing any symptoms, thus people with latent TB cannot infect other people. The currently known hosts for TB are human [4]

All other types of TB exist as either active or latent, and these types are classified based on the signs and body system the Mycobacterium Tuberculosis infected. For instance, pulmonary TB infects the pulmonary system, cutaneous TB infects the skin, etc. It is said to be rare for a human to be infected with more than one TB type. [4]

Human beings can catch TB through infection of human tissues by the bacterium. Human beings that are mostly vulnerable to TB are people who work at the hospitals where TB patients are being hosted, human with weak immune systems, or people who travel at places reported to possess high incidence of TB, also patients which HIV. [4]

In most cases TB affects the lungs, but this does not limit it to only the lungs, it can affect other areas like the brain, the kidneys and the skeleton. Persistent coughing regularly of blood, and chest pain are the obvious symptoms of TB. Other symptoms depend mainly on the organ being affected. [5]

The figure below shows the magnified TB bacteria in the lungs of a human being:

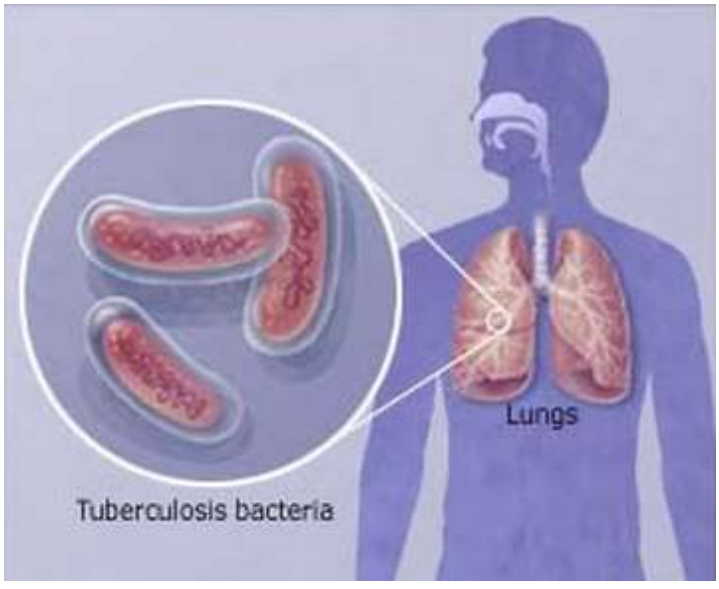


Figure 1: Magnified view of TB bacteria inside lungs [6]

The other environmental factors that contributes to TB transmission includes the ventilation, thus TB is highly transmitted at highly crowded places with poor ventilation.

2.1.1 Background

The German physician and scientist named Robert Koch discovered the bacterium that causes TB in the year 1882. He stated then that if the disease was being prioritised by the number of patients it kills, by then TB would have been the mostly paid attention disease, based on his statement that one in seven dies from the disease. TB was concluded to be an air borne disease, thus it can spread from one person to another if one person inhales the air in which the TB person has sneezed or coughed into. [5]

After the identification of TB, good ventilation was believed to be the better way to reduce TB transmission hence the standard medical therapy was predicted for the coming decades to be on rest, the sanatorium and good climate with fresh cold air. [5]

The first human was cured in 1944, using the drug manufactured from the antibiotic substance named streptomycin, which is produced by the fungus called Streptomyces griseus. TB was discovered to develop resistivity with time for the mentioned drug, hence two additional drugs were manufactured namely **para-aminosalicylic acid (PAS)** and **isoniazid** between the years 1944 to 1954, and a combination of this drugs was used as a treatment for TB. Therefore it is still believed that up to today a multidrug therapy is the only way to cure TB. [5]

The statistics estimates that in the two centuries from 1700 to 1900 before TB was discovered, TB was responsible for approximately one billion human beings, and the annual death rate in the year of discovery was seven million people. [5]

2.1.2 TB Transmission in the World

The world health organisation(WHO) Global tuberculosis report 2014 states that TB is regarded as the second only to HIV/AIDS as the most killer globally due to a single infection. The statistical results show that in the year 2013, 1.5 million people died from TB, and the approximate total number of people who were ill is 9 million. Most TB deaths occur at developing countries, [2] thus over 95% deaths in the year 2013 occurred in developing countries [1]. TB infect mostly children in the age from 15 years, is has been stated that in the year 2013, 550 000 children were ill with TB from which 80 000 where HIV negative, and it is regarded as the leading killer of HIV positive patients. [1]

TB death rate is declining , with an estimate of 45% from the year 1990 to 2013, and the number of lives save in the year between 2000 and 2013 amount to 37 million [1]. Although the reports show a magnificent decline in the rate of TB, most infections occur in less developed countries due to failure to access to proper and good ventilation places. Schools where most children spend most of their time are believed to be the places where TB can be mostly transmitted. [2]

The graph below shows the statistical report from world health organisation for the year 2014. It analyses the TB fatality rate.

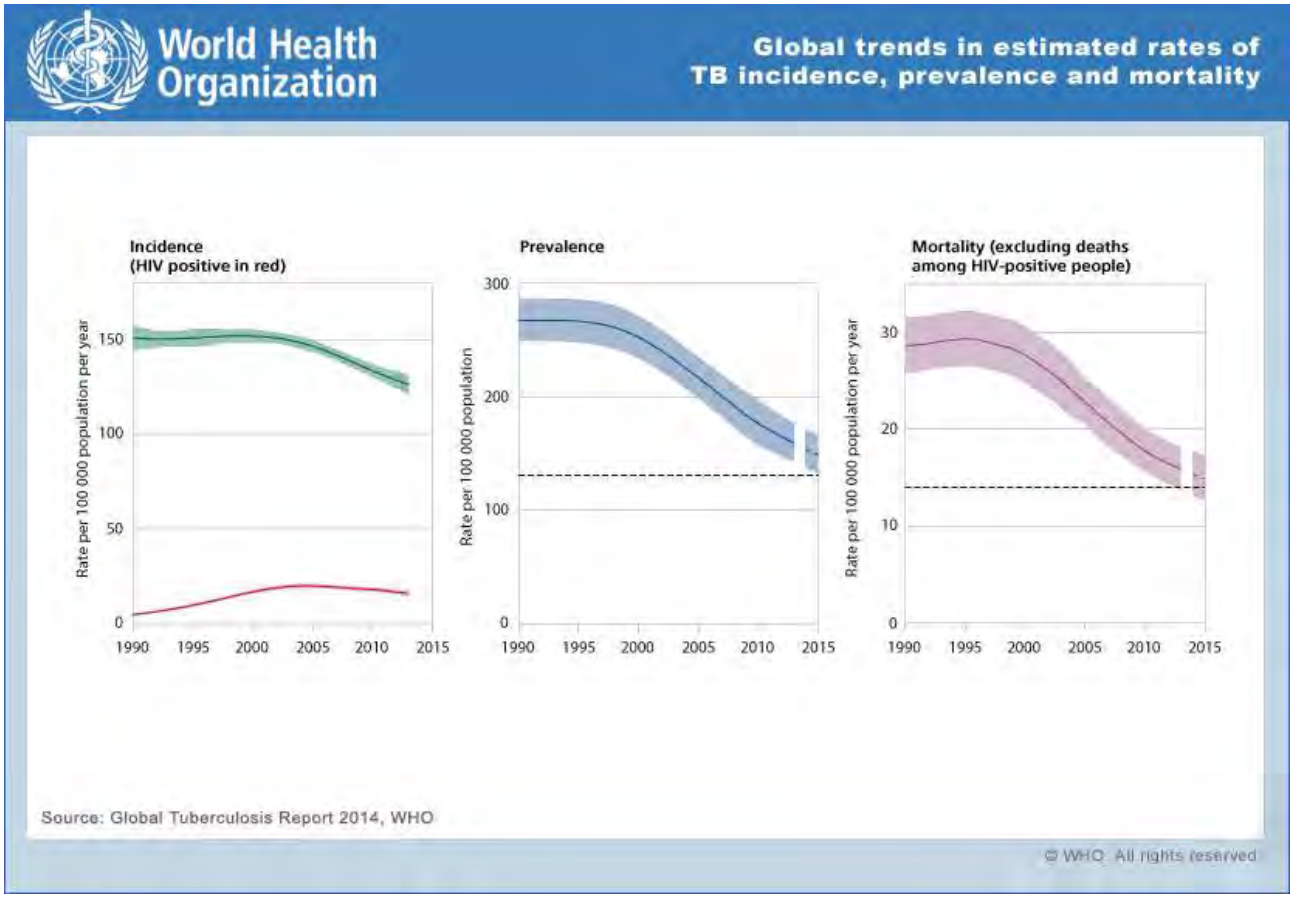


Figure 2: WHO Data and analysis

2.1.3 TB Transmission in South Africa

South Africa achieved a significant percentage of TB cure of about 40- 74% in the year between 1995 and 2011. Despite this fact, most children in the age 15 – 19 are vulnerable to TB infection as they gather in schools for quite a significant amount of time, and since South Africa is one of the less Developed countries, it lacks proper ventilation at schools, hence putting children at high risk of catching TB. [2] It is believed that 427 children among 10 0000 per year are catching TB in South Africa. [2]

The World health Organisation report states that South Africa still possess TB incidence greater than 1000 per 100 000 people in the year 2012. In the same year, 17.2 % of young people in the age on 15-25 caught TB in South Africa [1] . Most People in South Africa still leave in informal settlements, the HDA informal settlements in South Africa 2013 research shows that in the year 2001, only 68 percent of SA population lived in formal settlement, while in 2011, 78 % of the population lives in formal settlement, showing that 22% of South Africa population still leaven in informal settlement with poor ventilation, thus making the 22% of the population mostly vulnerable to TB.

The graph below shows type of dwelling in South Africa, with the findings from 2001 and 2011 census about the percentages of each dwelling type population constitution.

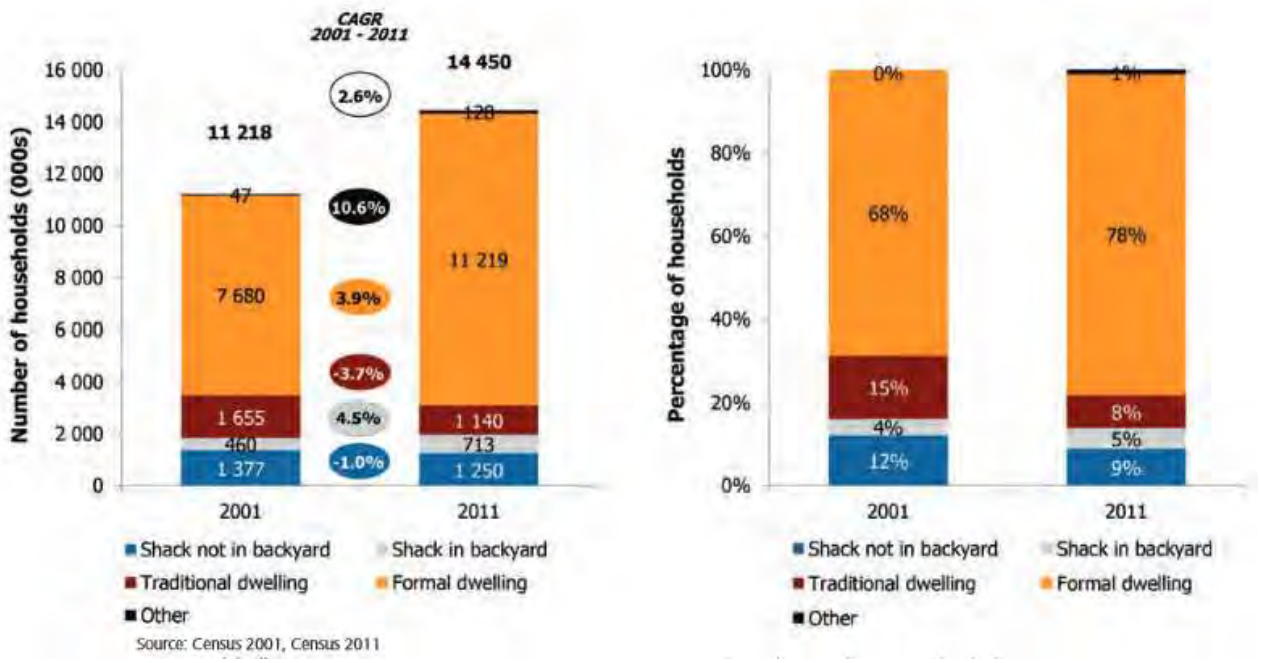


Figure 3: South Africa dwelling types [7]

2.1.4 2.3 TB risk factors in South Africa

TB can be caught by any human being anywhere, but certain factors increase the probability of one catching TB. The first factor to consider which increases TB chances is the week immune system. A healthy immune system often successfully fights TB bacteria, but the immune system may be weakened numerous diseases including HIV/AIDS, Diabetes etc. The other important factor to consider is the environmental condition. It is believed that people who work at prisons, immigration centres, hospitals or clinic where there is overcrowding and poor ventilation are highly likely to catch TB. [8]

2.2 GPS Technology

2.2.1 GPS History

The satellite navigation system was initially investigated in the early 1970. The navigation systems are of three types namely:

1) The Navy navigation system (Transit)
2) The US navy's Timation (Time navigation)
3) US Air Force project 621B. [9]

The development of GPS was initiated in 1973(Known as NAVSTAR GPS) by the US Department of Defence. The first GPS satellite was launched in 1978 and the 24 satellites were in obit in the year 1995, thus they are currently at least 24 operational GPS satellites in orbit [10]. These satellites are situated approximately 20,200 km altitude and each satellite cycles the earth in 12 hours. The GPS receiver system uses the signals from these satellites to calculate its coordinates as well as time. [10]

The diagram below show the GPS satellite in orbit:

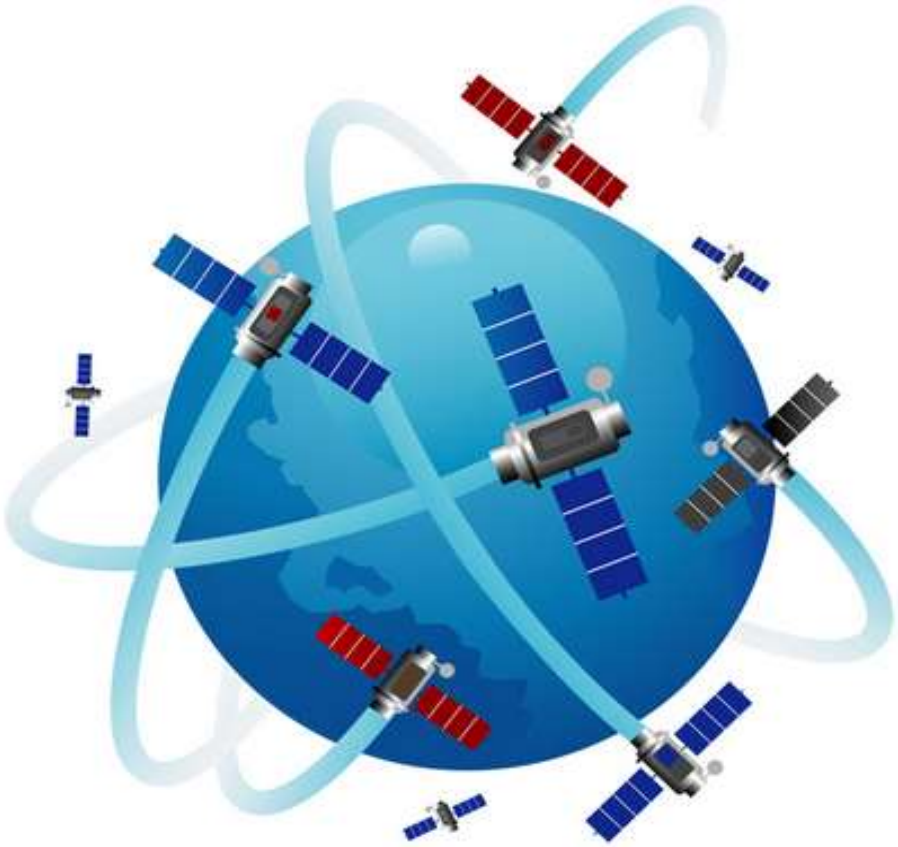


Figure 4: GPS Satellites in obit [11]

2.2.2 How GPS Works

The GPS utilises the three main parts to accomplish it functionality, these part are namely:

- **The GPS satellites** that broadcast position information data
- **The ground stations** that controls and updates the satellite information
- **The GPS receivers** which received the data from the satellites and performs GPS calculations

The GPS receivers require a clear view to the sky, therefore it perform better outdoors than indoors. [10]

2.2.3 How GPS Receiver computes Position

i. Geometrical Representation

To compute the GPS fix, the GPS receiver measures its distance from the satellite by looking at the length of the time it takes for the signal to reach the receiver, and the knowledge that the signal travels at the speed of light. Since the satellites are in motion, the fix calculated results in a sphere (Radius = fix).

The same information is received for numerous satellites, and at least three satellites can be used to compute a fix. The radius information obtained from the first satellite ant the information obtained from the second satellite results in a sphere that cuts the first satellite sphere at a plane (Circle), therefore this enable us to assume that the fix is within the resulting circle. Obtaining the radius from the third satellite will produce a sphere that intersects the circular plane at two points. Hence the

three circles are required to give the point in two dimensions (2D). A forth sphere can be used to calculate the location in three dimensions (3D). [12]

The diagram below illustrates the geometrical computation of a fix:

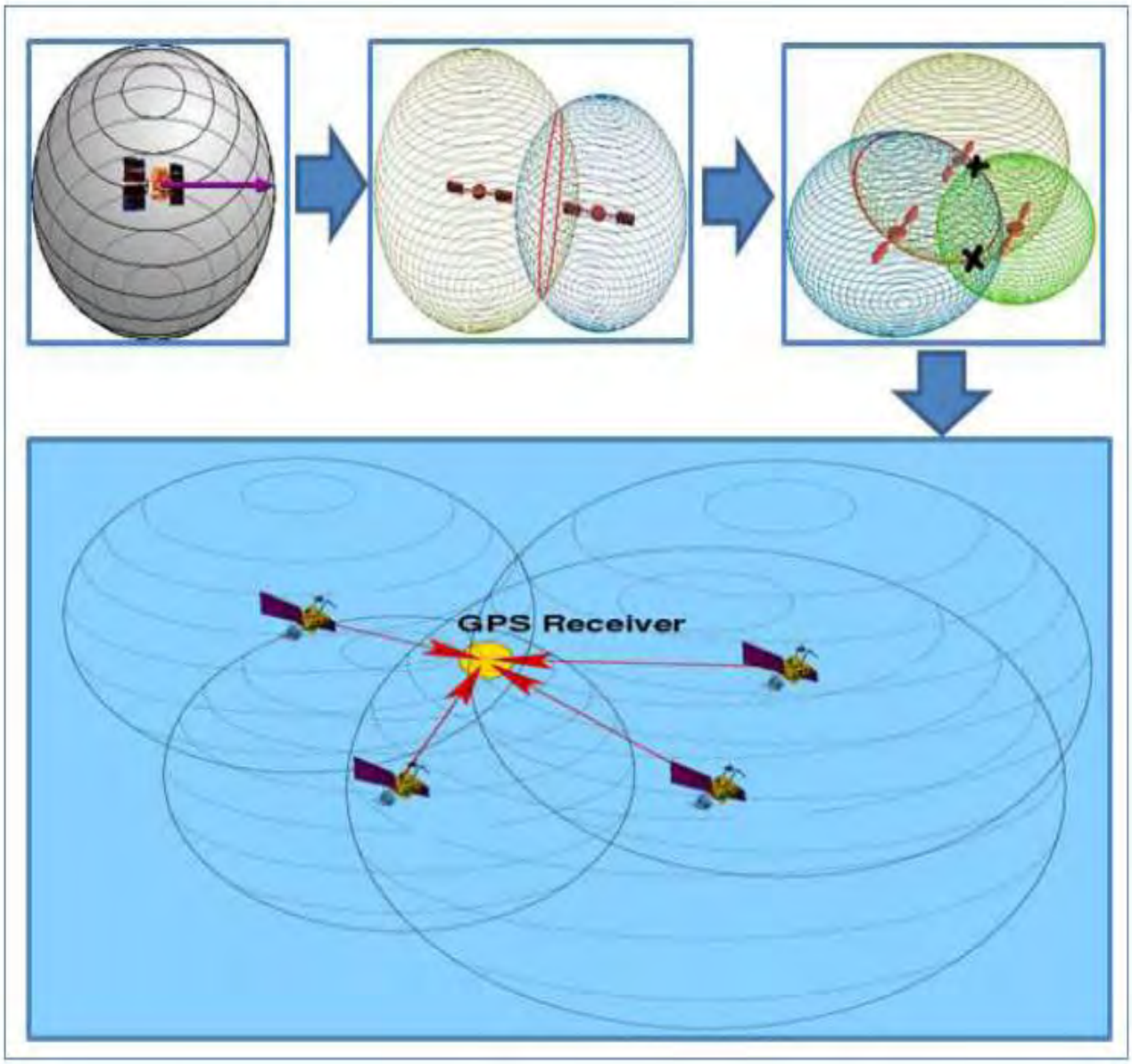


Figure 5: Geometrical Representation of GPS position calculation

ii. 2.3.4 GPS data

There are numerous types of GPS data used, the common types being the Receiver Independent Exchange Format (RINEX) data, National Marine Electronics Association (NMEA) data and Radio Technical Commission for Maritime Services (RTCM) data. RINEX focuses on combining data from different manufactures recovers, for statistical data processing and archive. RTCM uses a binary file for transmission of data between GPS receivers for real time DGPS/RTK corrections. [13]

The type of data of interest in this thesis is NMEA, which its main purpose is transmission of data between GPS receivers and other devices for real time positioning. It uses ASCII file which is easily readable to human eye.

The common NMEA-0183 sentences are shown in a table below:

Sentence	Description
$GPGGA	Global Positioning system fixed data
$GPGLL	Geographic position – Latitude/Longitude
$GPGSV	GNSS satellites in view
$GPGSA	GNSS DOP and active satellites
$GPRMC	Recommended minimum specific GPS data
$GPVTG	Course over ground and ground speed

Figure 6: NMEA data sentences

The sentence data type used in this thesis is the $GPRMC data, the following fields are found in the RMC data, separated by commas:

Fields	**Comments**
Sentence ID	$GPRMC
UTC Time	hhmmss:sss
A	Status
Latitude	ddmm.mmmm
N/S Indicator	N= North, S = South
Longitude	dddmm.mmmm
E/W indicator	E = East, W = West
S	Speed over the ground in knots
o	Track angle in degrees True
230394	Date - 23rd of March 1994
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

Figure 7: GPRMC data sentence [4]

3. Device Specification

In this chapter, the device full specification is discussed. Initially the device requirements are discussed and the specification drawn and each component of the specification priority weighted.

3.1 Overview

A body worn CO2 logger device is a small, portable electronic device which can be used to identify the areas/places where the spreading of TB is likely to occur by monitoring the level of Carbon dioxide in the atmosphere, as well as the humidity and temperature. This device will provide a log file which contains the following fields, the location of the device (coordinates) and time, the CO2 level, Temperature and humidity. The device may also have a kind of diary to keep records of the number of people (limited to 10) in place while still recording the samples of carbon dioxide. This diary will form the part of the fields in the log file if successfully implemented.

The device log file data will be easily downloadable to the PC, and the log file fields will be easily imported into a database. To achieve this, a piece of software to manipulate the downloading and clearing of the device data will be designed, developed and tested.

The device requires a reliable source of energy. It is expected to survive on a battery supply for at least 48 hours. The device is intended to be used by any person sans difficulties, thus even unskilled people will be able to use this device.

This device will have the following indicators to assure its normal functionality:

- Battery charging indicator to show charging and when the battery is full.
- Coordinates log indicator on the screen (to show if it's able to identify its location or not)
- Display Carbon dioxide, temperature and humidity on the screen in real time.
- Display screen to show current time.

3.2 Product/Service Description

This is a revised redesign of the CO2 Logger, based on the performance of the first generation of the device, the factors affecting the functionality and performance of the device includes:

- The battery life: The generation 1 device battery could not take up to 24 hence the log data was lost at some stage.
- The indicators: The generation 1 device could not provide adequate indicators to show that the device is performing normally. Hence more attention has been paid to ensuring that the device is user friendly enough to signify when it is not functioning normally.
- Screen Display: Screen to display time, location coordinates, carbon dioxide level, temperature and humidity on the new device has been implemented.
- Data logging : device location coordinates , carbon dioxide level, temperature and humidity log file can be easily downloaded from the device
- Software: A user friendly piece of software that can also show graphically the real-time behaviour of carbon dioxide level, humidity and temperature, as well as manipulating generation 1 and 2 device.
- Auto identification port software : the generation one device could not auto connect to the downloading data software, hence more attention has been paid to ensuring that the device can auto connect to the PC

3.3 Constraints

Device accuracy: The device is expected to operate normally with minimum errors on the log data, therefore these calls for:

- A GPS which will be able to give best performance indoors and outdoors
- A rechargeable battery supply which can last longer than 1 day
- A memory enough to store log data before its downloading interval
- A display or form of indicators to show the device when it is behaving abnormally

Time frame: This device is expected to retain all the functionality of the first generation of the device, while still improving its accuracy and friendliness, and the time allocated for the project design and implementation phase is 2 years.

Size, weight and Portability: The device is expected to be small and portable therefore the components used should adhere to the size and portability requirements.

3.4 Dependencies

In order to determine the device design and development feasibility, the main design components that were required were identified and listed in the following sections.

i. Software Packages

The following software packages are required to facilitate the design of the project:

- Solid-works package 2013
- Eagle CAD 6.6.0
- Visual studio 2010 (for the software design)
- Altera (micro controller software programming environment

ii. Hardware

The following hardware components are required:

- GPS
- CO2 Transducer
- Microcontroller
- Flash Memory
- Display
- Rechargeable battery
- Battery charger and protector
- Laptop charger
- Phone charger

iii. Other

Other tools required in this project include:

- Personal Computer
- Bench Power supply
- Bench Multi-meter
- Soldering station
- Oscilloscope

3.5 Requirements

The following definitions are used to guide on the priorities of the agreed requirements.

- Priority 1 : The system must have this requirement
- Priority 2: The requirement is needed to improve the processing and fulfilment will create immediate benefit
- Priority 3: The requirement is a "nice to have" which may include new functionality

i. Functional Requirements

The following functional requirements were agreed upon:

#	Function	Description	Priority
1.	Data Log	The device system should be able to log on the carbon dioxide level, temperature, humidity and current location coordinates. This log file should be stored within the device and easily be accessible using the device associated software. Thus the log file should provide the following fields: • Location coordinates • Time • Carbon dioxide level • Humidity • Temperature	1
		The system should be able to provide a diary to record the number of people in the area of interest.	3
2	Battery supply	The system should be able to work reliable on battery supply for at least 48 hours.	1
		This device should use rechargeable batteries	1
		The system should provide a means to show its battery level	2
		The system should indicate the battery life, thus it should be able to tell when a battery replacement is required	3
3	Device Memory	The device should provide memory to store its log data	1
		It should also show the indication of the memory capacity, as well as showing available space.	2
4	Screen Display	The system should have screen to display the messages as well as real time data log	1
		Touch screen to allow typing of location name or number of people at a place	3
5	Ports	The system should provide the communication port to allow for data accessing and downloading, as well as the device calibration	1
		The system should have two charging ports to allow for laptop of phone/PC charger	1
6	Device portability	The device should be portable, thus its weight should not exit 120g. The device should also be easy to carry around (Thus it can have a form of a handle or grip)	1
7	Data transmission	The device should be able to transmit its log data which includes its current location wirelessly	3

Table 1: Device Specification

ii. Software Requirements

A graphical user interface should be developed, which can automatically connect and communicate with the new device, as well as being able to communicate with generation 1 device. The software system should provide the following functionality:

1) Automatic connection:

Anytime a device is connected to the PC, the software should automatically connect to the device.

2) Data downloading and clearing

A user friendly piece of software to allow for data downloading. Thus the user should be able to spot the download button and click on it to perform data download. Clearing of data button should also be easily spotted and on click, the device log data should be cleared.

3) Device Synchronisation and calibration

A piece of software to allow for changing of device log variables, such as synchronising device time with a computer clock should be easily accomplished using the software system. Also device calibrations such as resetting the reference level for carbon dioxide and humidity should be easily implemented using the software system.

4) Live log

The software to graphically show real time data log on the device. This software system should be able to also store the recent received data for future reference.

5) Installation environment

The software system will run on window operating systems. The target operating system is windows 7 and higher versions (e.g. Windows 10).

4. High Level System Design

This chapter discusses the overall hardware and software design of CO_2 body worn device, identifying its components at conceptual level. It starts with the overall description of the hardware design, showing the power supply design and the control system design. It then describes the software piece of the device. Components selection for the device is also discussed in this chapter. The detailed description of the device is discussed in the next chapter.

4.1 System Overview

CO_2 Body worn device is a system which logs on the Carbon dioxide level, humidity, temperature and its location. The system relies on battery supply for its functionality. The log data access and downloading is manipulated using a piece of software. The power supply network and the control network is discussed below, as well as the software high level design.

4.2 System Hardware Design

The main components which constitute the system hardware are the power supply network and the control system network.

4.2.1 Power Supply Network

The CO_2 Body worn device power is supplied by rechargeable batteries. These batteries can be either charged using normal phone charger or from the computer ports or it can be charged from a laptop charger. While charging the device from a phone charger or computer ports, a way to boost the supply to a high level, to accommodate other components supply and charging is required. Also a way to regulate down to a constant voltage supply to the control system components is required.

The diagram below shows the power supply network:

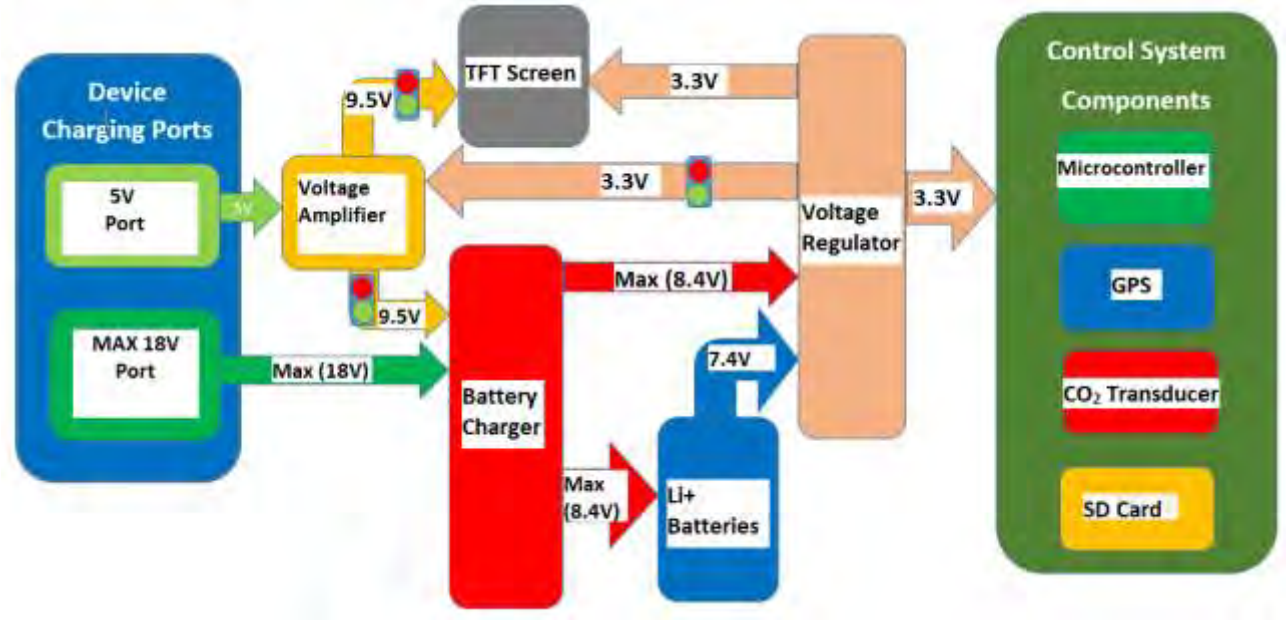


Figure 8: Power supply network of CO2 Body Worn Device

From the network above, the device can be either charged using Computer/Phone charger, in this scenario, the 5 Volts from the charger port is being boosted to 9.5V, of which is used to feed the LCD screen in a controlled fashion, the same voltage is being supplied to the charger module also in a controlled fashion. This is controlled to ensure that no back feed or self-charging of the batteries occurs when the charger is disconnected

The charger module then charges the batteries as well as supplying the voltage regulator to ensure that the control system continues to perform their function immediately when the charger is connected, even if the batteries did not have enough voltage to power up the control circuit. Once the charger is disconnected the battery supply takes over.

If charging is performed from the laptop port, the phone/computer charger port supply to the charging module is shut down. To turn on the screen, the supply to the 9.5V output booster module is being feed from the voltage regulator. The screen is turned on and off in a controlled fashion as to save power.

4.2.2 Control System network

The control system network is composed of the microcontroller (MCU), GPS unit, CO_2 transducer and SD card. The latter three modules are all being controlled by the micro controller. GPS and CO_2 transducer supply data to the MCU, of which the MCU manipulates the storage of data to the SD card. Also this data can be accessed through the device port from the computer using USB cable. MCU also handles all the commands from the Computer software to either access or reset data etc. The diagram below shows the control system network with data flow and control lines.

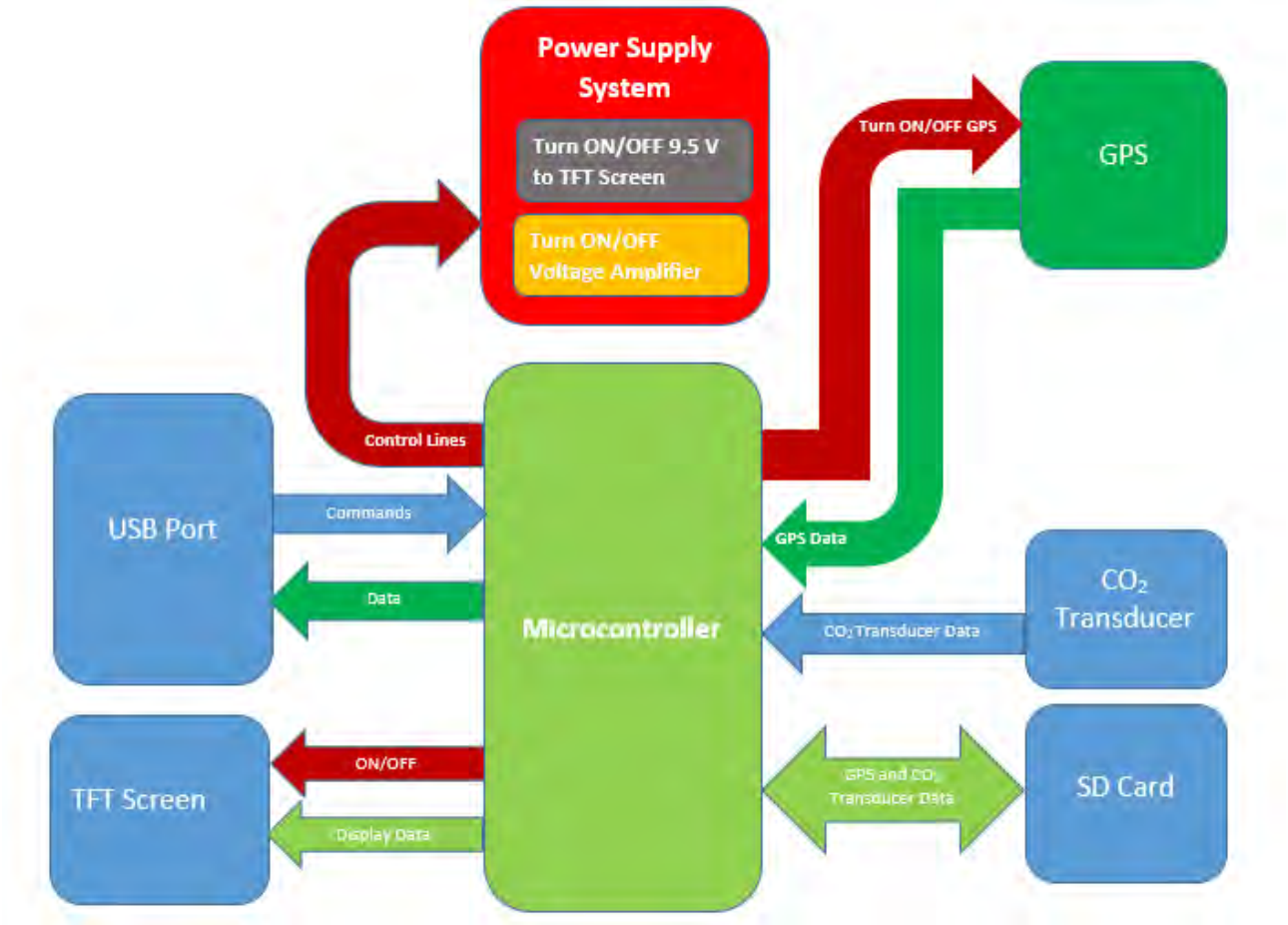


Figure 9: CO_2 Body Worn Device Control system network

The diagram above shows the "heart" of the CO_2 Device. The MCU controls when should the screen turns on and off by turning on/off the power supply line to the screen back lights. The MCU also controls when charging to be performed by turning ON/OFF the voltage amplifier module. Additionally, based on the motion of the device, GPS is turned ON/OFF. All this turning ON/OFF of device components modules is performed to save power.

MCU also receives commands from the computer CO_2 piece of Software, this commands includes device settings and device data manipulation such as when to download and delete data. For instance, if a download data command is received by the MCU, then the MCU will access the required data from the SD Card and send the data back via the port. Furthermore the MCU is responsible for receiving GPS and CO_2 Transduce data, and making proper selection of the required data and storing this data on the SD card.

4.3 System Software Design

The Graphical User Interface (GUI) is required to facilitate manipulation of the CO_2 Body Worn Device. The device will be connected on the computer, and the commands to the device send via the Software. The diagram below shows the overall device connection and what should happen once the device is connected. The use case diagram and other detailed description of the software system design are elaborated on the next chapters.

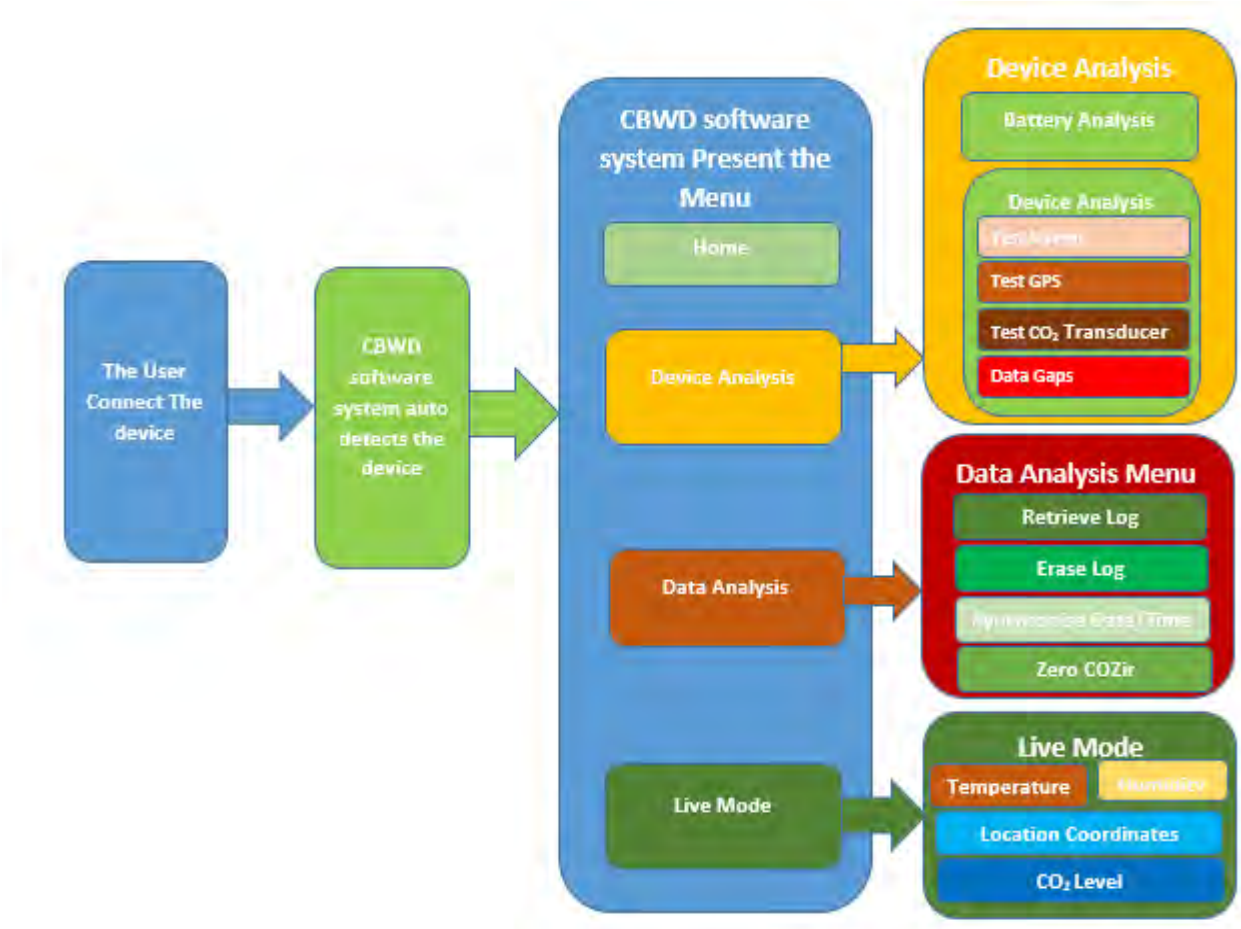


Figure 10: Basic Functions to occur on CBWD Software

On the diagram above, once the device is connected to the computer and the CBWD software is started, the software system should be able to auto identify the device connected port and get the device name, then present the menu to the user. On the menu the user can be able to analyse the device in terms of the batteries or the device performance. The device performance is analysed based on the GPS, CO2 transducer and the actual gaps on the data to download, thus the device will check if it is receiving any data from the GPS and CO2 Transducer, at specific interval, hence the overall device life will be concluded.

Data manipulation is performed under data analysis menu. Under this menu, the device data can be downloaded and deleted. Also the device time and date can be successfully synchronised to the computer time and date. The CO_2 Level reference for the device is also performed under this menu.

Live results of the device can be found under the live menu, thus the current temperature, humidity, CO_2 Level and the device current location shown in real time, and the graphical behaviour of the temperature, humidity and CO_2 shown.

4.4 Component Selection

In this section, the components of the CO_2 Body worn device are identified, and the selection criteria is based on the specification functional requirements

4.4.1 Hardware Components

i. Microcontroller

The system requires to be able to get its current location, thus the first communication transmission line, and also the system requires to be able to receive data related to the humidity, Temperature and CO_2 level. The microcontroller also needs to be able to send data to and receive data from the PC. Based on the information above**, three RX** ports are required and **one TX** port. Furthermore, the MCU need to be able to store data, also to be able to display the current data on the screen, thus **two SPI** ports required to fulfil this functionality. A way to program and update the device firmware is required, thus the MCU which is easy to program is required. Control lines to switch ON/OFF other device components accordingly are also required. The MCU also should be cost effective and possess enough memory to perform its function.

Based on the functional requirements, the selected MCU is STM32F107V. The diagram below shows the selected STM32F107VC:

Figure 11: STM32F107VC diagram [14]

The MCU possesses 256 Kbytes of Flash memory, 5 USARTs and 3 SPIs (18 Mbit/s). The total number of Input/output pins for this MCU is 100, enough to accommodate control lines, communication lines and power lines.

The diagram below shows the summary of STM32F107VC properties:

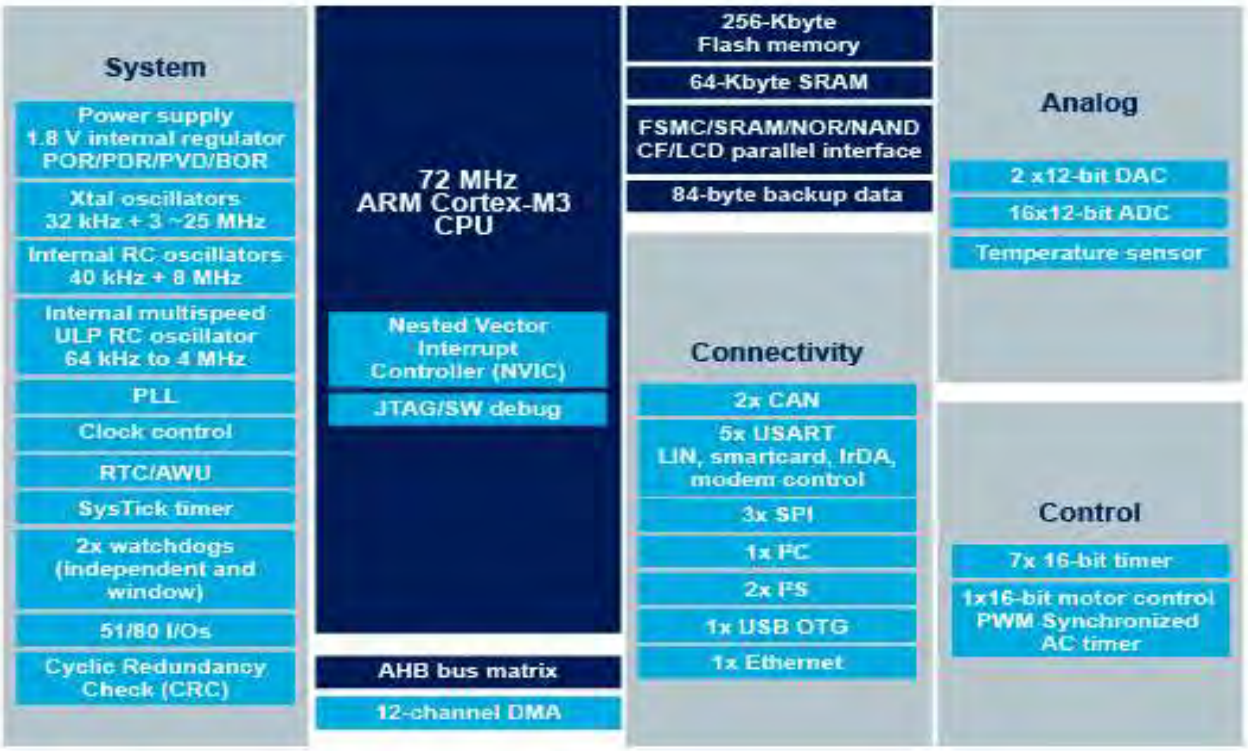


Figure 12: STM32F107 properties [15]

ii. GPS Unit

The main modules that constitute the GPS unit includes the GPS receiver, and the GPS antenna. The GPS receiver is the one responsible for performing the calculation of the location coordinates, while the antenna assist the GPS receiver in logging onto GPS satellites.

GPS Receiver

The system is expected to be able to log its current location coordinates. As a means to safe power, the GPS could be turned ON/OFF hence a GPS receiver with minimum cold, warm and hot start is required. Device availability is the other factor considered when selecting the GPS receiver module. The module should also be low power consuming.

The selected GPS receiver module is Ublox NEO-6Q. It can be supplied from the voltage range of 2.7V to 3.6V, additionally the module possesses the following interfaces;

- UART
- USB
- SPI
- DDC (I2C compliant).

This module offers many connectivity options in a miniature 16 x 12.2 x 2.4 mm package, which allows for the portability of the system.
The GPS module performance is analysed in the table below:

Parameter	Specification	Value
Time-To-First-Fix	• Cold start and • Warm start • Hot start	• 26 s • <26s • < 1s
Sensitivity	• Tracking and Navigating	• -162 dBm
Horizontal Position Accuracy	• GPS	• 2.5 m
Operational Limits	• Dynamics • Altitude	• ≤ 4 g • 50 000 m

Table 2: NEO-6Q GPS performance [16]

The diagram below shows the GPS receiver module used:

Figure 13: GPS Receiver NEO-6Q

GPS Antenna

GPS antenna selection was based on availability, size and its performance. The device portability and dimensions dictated the type selected namely **"The Cloud" Flexible Polymer GPS/GLONASS /COMPASS Cloud Shape Antenna**. The antenna operates within the frequency band 1559-1610MHz, with a peak gain of 3dBi. The maximum power input to the antenna is 5 Watts. [17].

The table below summarises the performance properties of the antenna:

ANTENNA	
STANDARD	GPS-GLONASS-COMPASS
Operation Frequency (MHz)	1559-1610
Polarization	Linear
Impedance (Ohms)	50
Max VSWR	1.2:1
Peak Gain (dBi)	3
Efficiency (%)	80
Average Gain (dB)	-1
Radiation Properties	Omni-directional
Max Input Power (Watts)	5

Table 3: GPS Antenna characteristics [17]

The antenna offers the lower return loss at it operating frequency. The diagram below show the return loss of the antenna:

Figure 14: Return loss of FXP611 GPS/GLONASS/COMPASS Antenna [17]

It also offers a consistent efficiency of approximately 80% within its operating bandwidth. The efficiency drops slightly down outside its operating frequency. [17]

The diagram below shows the efficiency of the antenna:

Figure 15: Efficiency of FXP611 GPS/GLONASS/COMPASS Antenna [17]

Therefore the antenna was selected based on its powers consumptions, its efficiency and performance as outlined above.

The diagram below shows the antenna used on the device:

Figure 16: Taoglas GPS Passive antenna

iii. Carbon Dioxide Transducer

Based on the specification, the device should be able to log on the CO_2 level, the temperature and humidity. The selected CO_2 transduce module solely is able to provide this functionality, thus it is able to provide the CO_2 levels, humidity and temperature.

The selected CO_2 transducer is the COZir, which is an ultra-low power (3.5mW), high performance CO2 sensor, ideally suited for battery operation, portable instruments and measuring ambient CO2 for HVAC control. [18] It can be supplied from a voltage range 3.25V – 5.5V, and it peak current 33mA, with a power consumption 3.5mW.

The table below gives a summary of the general performance of the COZIR:

General Performance	
Warm-up Time	< 10s. 1.2 secs to first reading.
Operating Conditions	0°C to 50°C (Standard), -25°C to 55°C (Extended range) 0 to 95% RH, non-condensing
Recommended Storage	-30°C to +70°C
CO2 Measurement	
Sensing Method	Non-dispersive infrared (NDIR) absorption Patented Gold-plated optics, Solid-state source and detector
Sample Method	Diffusion
Measurement Range	0-2000ppm, 0-5000ppm, 0-1%
Accuracy	±50 ppm +/- 3% of reading[1]
Calibration	Autocalibration[6]
Non Linearity	< 1% of FS
Pressure Dependence	0.13% of reading per mm Hg in normal atmospheric conditions.
Operating Pressure Range	950 mbar to 1050 mbar[2]
Response Time	30 secs to 3 mins (Configurable via filter type and application)[3] Reading refreshed twice per second.[3]

Table 4: COZir specification [19]

The diagram below shows the selected COZir:

Figure 17: COZir Ambient Sensor

iv. Power Supply

Based on the system power requirements, which includes the components modules power requirements, the following voltage level requirements were drawn:

- 3.3V to supply MCU, GPS, COZir
- 9.5V to supply TFT Screen backlights

Therefore the system requires to be supplied from the battery, and a means to achieve all the power requirements to different modules be implemented. The power supply includes the following components:

Batteries

Based on the casing requirements and taking into consideration the recharging factors, two lithium ion AA batteries were selected, with the rating 800mAh at a voltage 3.7V each.

Voltage Amplifier and voltage regulator

The TFT Screen display requires 9.5V for its back lights, hence a voltage booster to supply the required voltage is needed. The selected voltage booster module is MAX1790, which is a boost converter that incorporate high-performance, current-mode, fixed-frequency, pulse-width modulation (PWM) circuitry with a built-in 0.21 Ω n-channel MOSFET to provide a highly efficient regulator with fast response. It offers 90% efficiency, and voltage output range can be adjusted from Input voltage (V_{in}) to 12V. Additionally its small size (4.78 X 2.95 X 1.1mm) makes it more convenient to be used in the device. [20]

Charger Module

The selected battery charger module is the MAX1758, which is a switch-mode lithium-ion (Li+) battery charger that charges one-to-four cells. It offers ±0.8% Battery Regulation Voltage Accuracy at the battery terminals, with a low dropout 98% duty cycle. This module also offers voltage and temperature monitoring, and it can also prevent overcharging. The input voltage can range to a maximum of 28V, making it possible to use both cell phone and laptop charger system. [21]

v. Debugging and Charging Ports

The device needs a way to connect to the computer for debugging, charging and data transfer, additionally communication port to the device. To achieve all these, a single mini B USB connector port has been selected based on the availability and low cost.

The diagram below shows the USB mini B port:

Figure 18: USB mini B connector

vi. Display and Indicators

The way to show when a device is connected to external power supply, or when it is charging is required, hence LEDs have been incorporated for this functionality. Additionally, to display device data, a display screen is required, and the selected display screen is DT024CTFT LCD colour screen. The screen selection was based on its low cost and power consumption:

The diagram below shows the selected LCD screen:

Figure 19: DT024CTFT LCD colour screen [22]

5. Hardware Detailed Design

In this chapter, the hardware design and implementation is described in details. The hardware design is composed of two main blocks namely the power supply block and the control system block. These two blocks are described in details with the circuit diagrams included. Initially the power system block design is elaborated followed by the control system block. The complete circuit diagrams and board layout of the entire system are included in the appendix A.

5.1 Power System Block

5.1.1 Overview

The CO_2 body worn device relies on rechargeable battery power supply. These batteries can be charged either from a small charger (Phone charger or Computer USB port), or it can be charged from a laptop charger (Max voltage 20V). This section describes the power supply including the two charging system design.

While charging, if a laptop charger is connected, the power from the mini B port charger is shut down and the power from the laptop charger line is feed directly to the charger module MAX1790. The MCU then becomes responsible for turning on or off the LCD display. In case the system is charged solemnly from the phone charger, the voltage is being feed to the booster, from which it is being feed to the charger module and display screen. The charger module is responsible for charging the Li+ batteries, as well as supplying the voltage to the voltage regulator. The batteries take over when the charger is disconnected.

The diagram below shows the process flow diagram for the power supply system including charging:

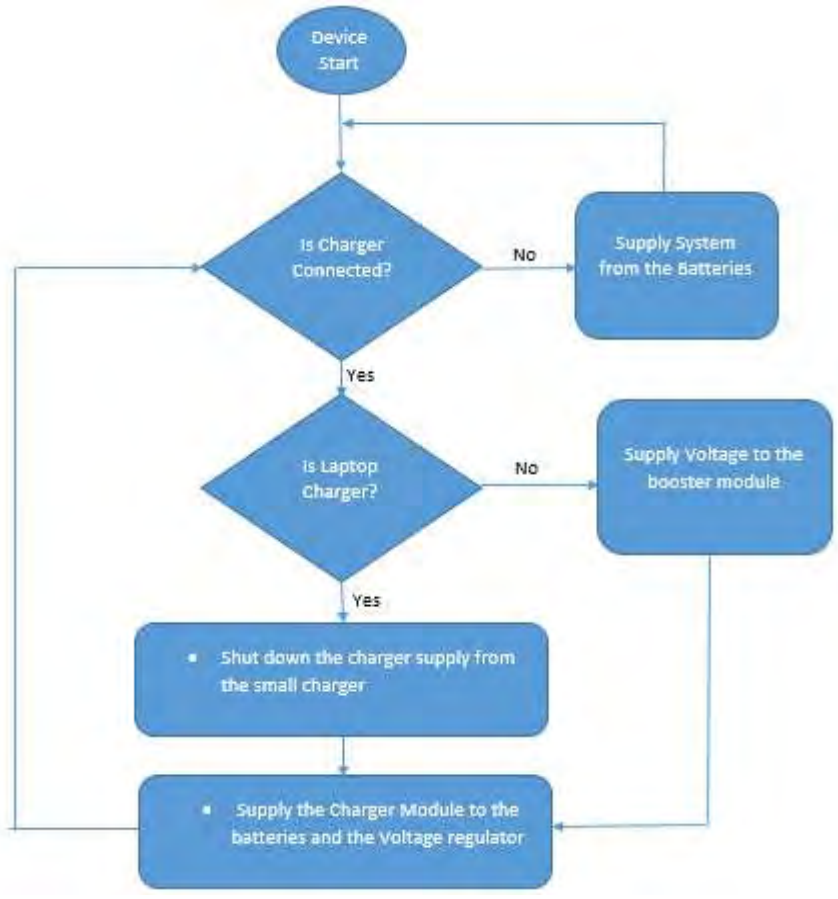


Figure 20: Flow diagram for the Power supply

5.1.2 *Laptop and Phone Charger / Computer USB port Charger design*

While charging from a laptop charger (max 20V), the voltage is supplied directly to the charger module. The charger module circuit is discussed in the following charger section.

While charging the system from the mini B port, the voltage at the port which is 5V max, is initially feed to the voltage booster MAX1790. The booster amplifies the voltage to 9.5 volts, and the 9.5V is feed to either the LCD colour display and/or the battery charger module in a controlled fashion. The switching of this voltage to either the LCD and/or the Charger module is done through an ideal diode namely LTC4353 module.

The block diagram below shows the main modules involved in charging the battery supply from the phone charger:

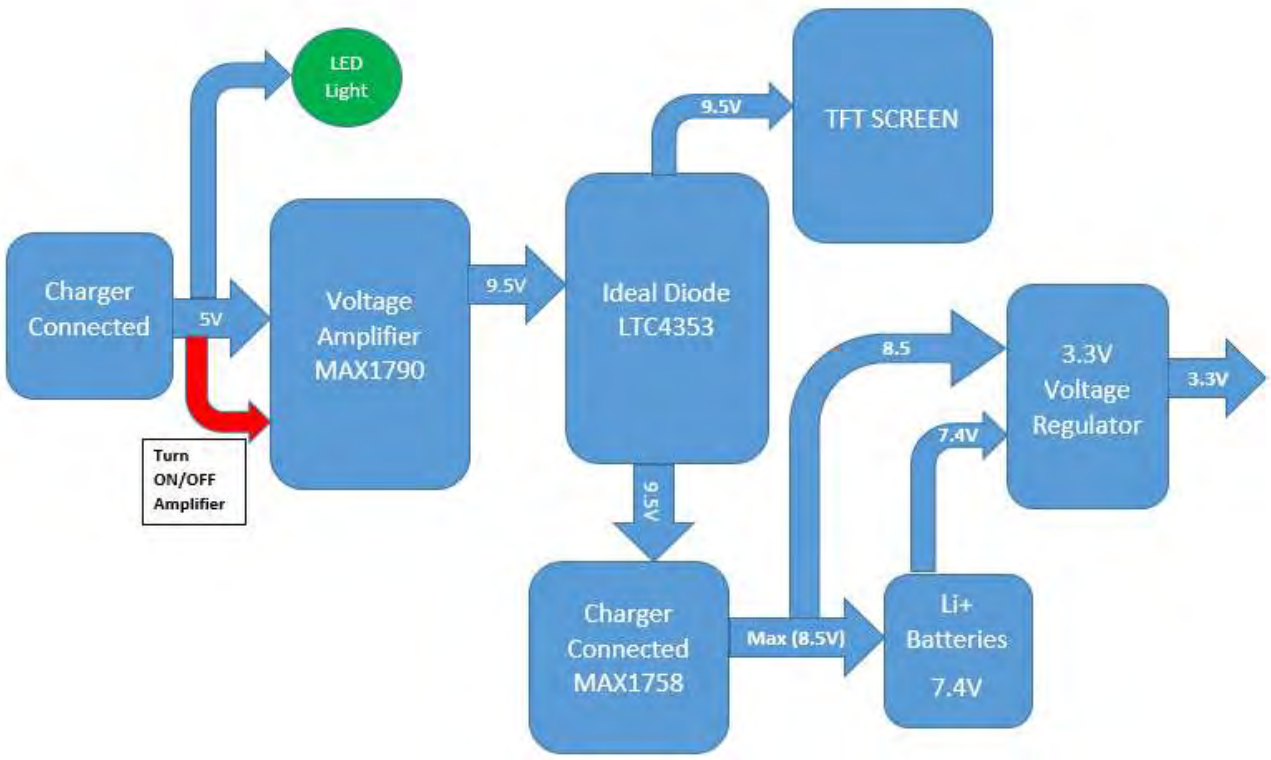


Figure 21: Mini B port charging block diagram

5.1.3 *The Circuit Diagram*

i. *Mini B type USB socket*

Mini B USB connector has a total of 5 pins, and the pins connection is described in the table below:

Pin No	Signal	Description
1	V_{BUS}	Power Nominal 5V
2	D-	Data -
3	D+	Data +
4	ID	NC
5	GND	Ground
Shell	GND	Ground

Table 5: Mini B connector pins description [23]

The power line is connected from pin 1, and this 5V is feed to an LED with two Resistors 1K, thus limiting current through the LED:

$$I = V/R$$
$$\Rightarrow I = (5V - 2.1V)/(1K + 1K)$$
$$\Rightarrow I = 1.45mA$$

In this case the voltage drop across the LED (Green LED) is approximately 2.1V, therefore it gives the limiting current through the LED of about 1.45mA. This gives enough brightness on the LED to be recognised by human eye.

The circuit diagram below show the connection configuration of the mini B,and the highlighted line is the line of interest (5V line):

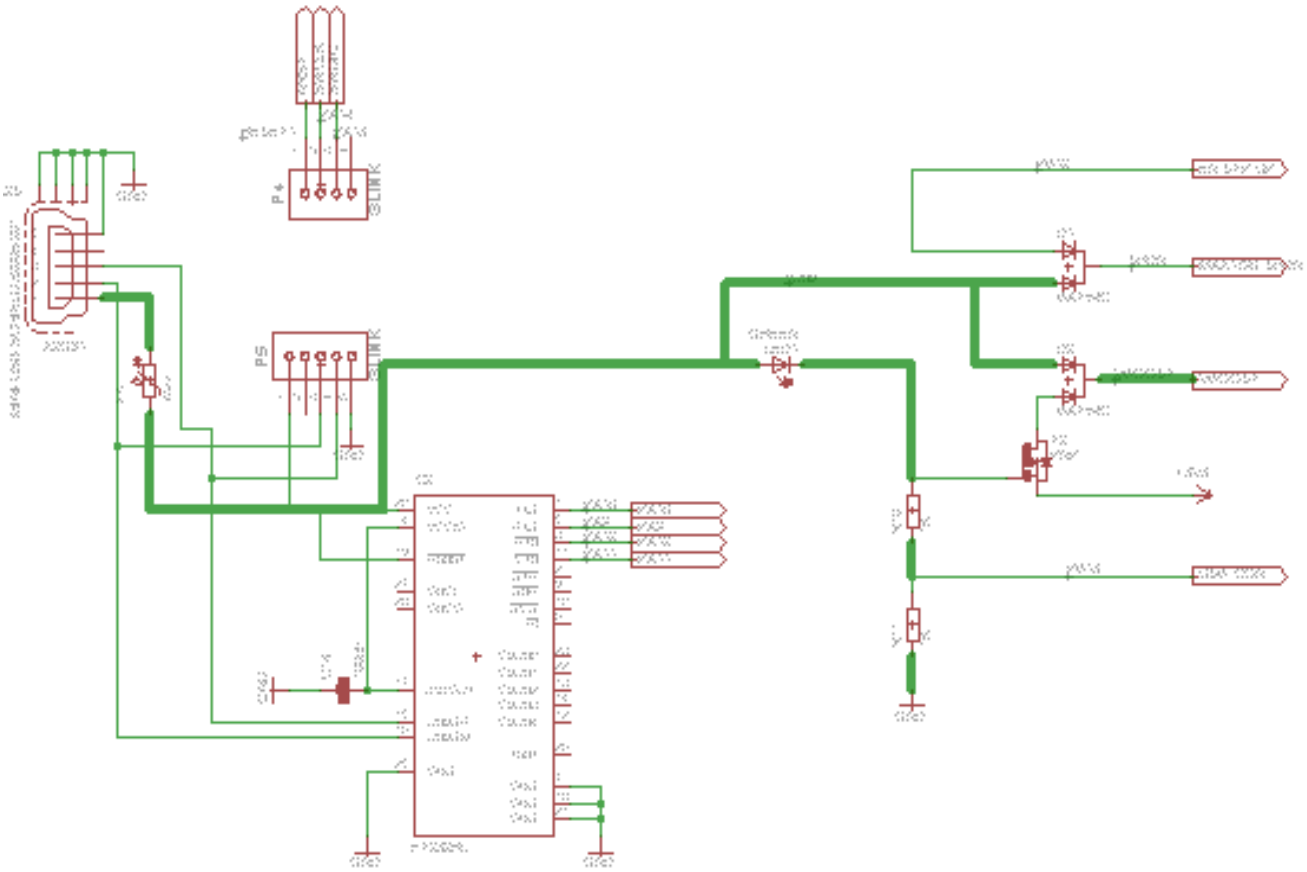


Figure 22:5V Power Supply line

From the circuit diagram above, the 5V line is also feed to the voltage booster through a dual diode BAT54C, this is the voltage to be boosted, and the booster module circuit configuration is discussed in the next sections. The same power is used to turn on the voltage booster (MAX1790 to be discussed in detail on the next chapters) through dual diode BAT54C, this ensures controlling voltage booster module from the microcontroller, separates it from controlling it from the mini B port, it should be noted that once the charger is connected to the port, the booster automatically turns on.

ii. ***Voltage Booster***

The voltage booster used in the circuit design is MAX1790. The discussion of internal structure of the booster is beyond the scope of this thesis, but the block diagram has been provided below to give an overview of what is within the booster module.

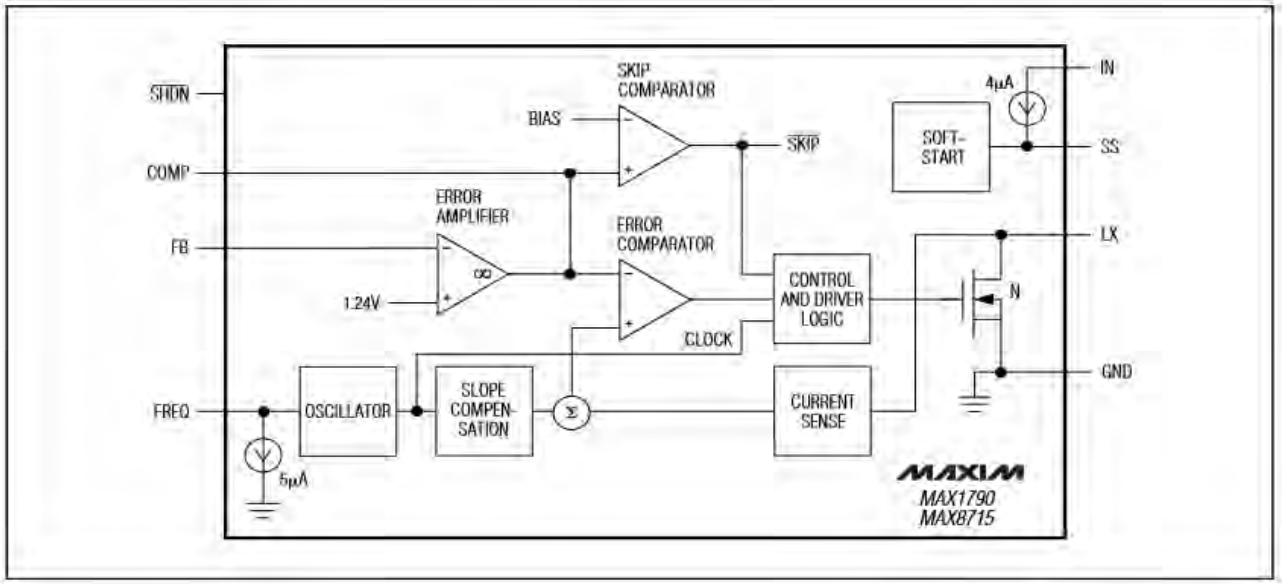


Figure 23: Max1790 block diagram [20]

This module has 8 pins, and the table below shows the pins description and their function:

Pin	Name	Function
1	COMP	Compensation pin for error amplifier, connect a series RC from COMP to ground
2	FB	Feedback pin with reference voltage 1.24V nominal. Connect an external resistor-divider tap to FB and minimise the trace area. Vout = 1.24(1 + R1/R2)
3	!SHDN	Shutdown control pin. Drive it low to turn the module off
4	GND	Ground
5	LX	Switch pin, connects inductor and minimise trace area for lowest EMI
6	IN	Supply pin (Requires at least 1uF capacitor to GND)
7	FREQ	Frequency select input (1 = 1.2MHz, 0=640KHz)
8	SS	Soft start control pin

Table 6: Max1790 pins description and their functions [20]

The circuit diagram below shows the connections on the booster:

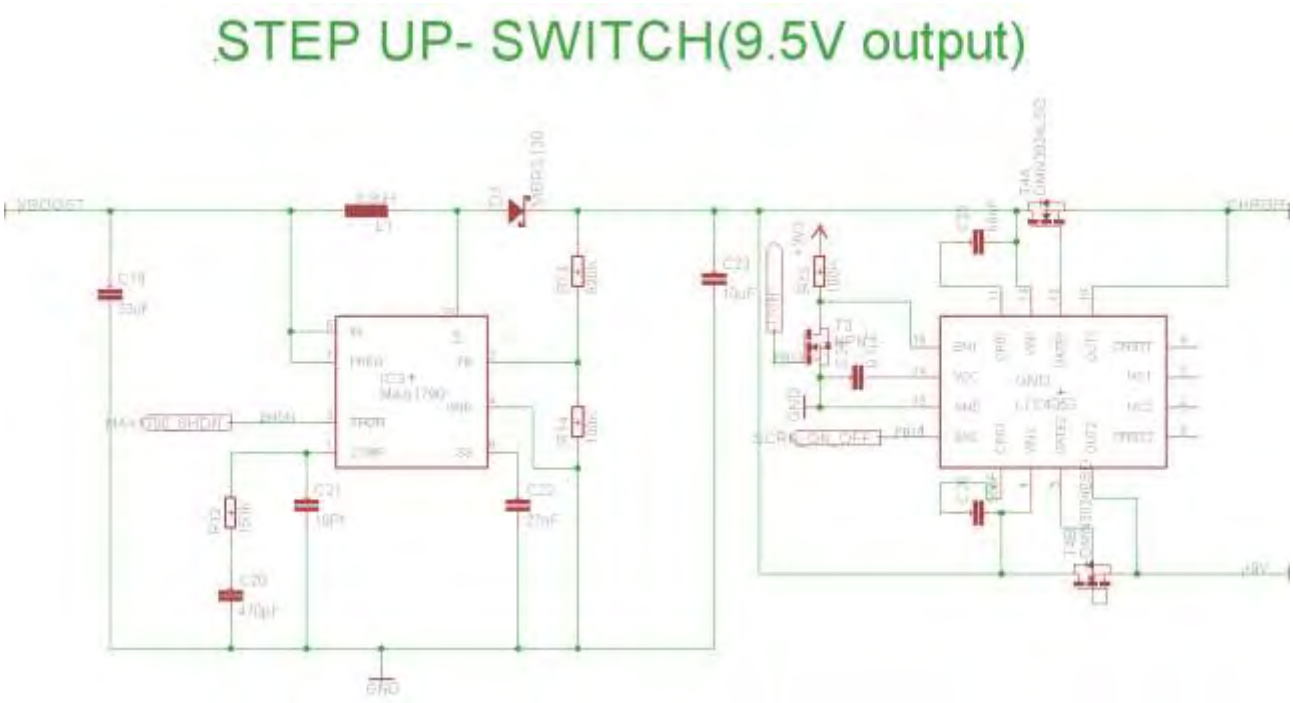


Figure 24: Voltage Booster circuit diagram

Based on the pins functional description on the table above, it can be recognised on the above diagram that a 33uF capacitor has been connected on the VIN line (Pin 6), also Frequency pin (Pin 7) is held high to ensure the module operates at a frequency 1.2MHz.

The inductor also connected from the LX pin, and the value for this inductor has been found using the formulae below:

$$L = (V_{in}/V_{main})^2((V_{main}-V_{in})/(I_{main} \times f_{osc}))(ntyp/LIR)\ [24] \quad ntyp = 0.85,\ LIR = 0.5\ [20]$$

On the worst case, when the screen is controlled from the microcontroller, V_{in} used will be 3.3V from the microcontroller, hence using 3.3V to achieve V_{out} 9.5V:

$$L= (3.3V/9.5V)((9.5 - 3.3)/(0.15A \times 1.2MHZ))(0.85/0.5)\ [20]$$
$$L= 7.06\ uH$$

Picking the standard size that is close to the inductor value: L = 6.8uH

From this inductor value, the ripple current can be calculated using the formula:

$$I_{IN}= (I_{MAIN\ (max)} \times V_{MAIN})/ (V_{IN\ (MIN)} \times eff)\ [20] \text{where eff is efficiency [20]}$$

So assuming the efficiency of 80%:

$$I_{IN} = (0.15A \times 9.5V)/(3V \times 0.8) \text{ assuming the worst case of } V_{in} \text{ minimum of 3V}$$

$$I_{IN} = 0.59A$$

And therefore the ripple current can be obtained from the formula:

$$I_{RIPLE} = (V_{in(min)} \times (V_{Main} - V_{min}))/(L \times V_{main} \times f_{osc})\ [20]$$
$$I_{RIPLE} = (3V \times (9.5V - 3V))/(6.8uH \times 9.5V \times 1.2MHz) = 0{,}251A$$

Hence peak current can be computed as:

$$I_{PEAK} = I_{IN} \times I_{RIPLE}/2 \quad [20]$$

$$I_{PEAK} = 0.59 \times 0.251 / 2 = 0.72A$$

Therefore the diode connected in series with the inductor was chosen to be MBRS130, which can be able to handle the output voltage of 9.5V and the peak current calculated above.

In front of the diode, are the two dividing resistors R13 and R14, this resistors are said to obey the formula:

$$Vout = 1.24(1 + R13/R14) \quad [20]$$

$$\Rightarrow \quad R13 = R14 * (Vout - 1.24)/1.24$$

Choosing R14 as 100K give R13 = 666K. So the standard value of R13 was selected experimental as 680K which gives the voltage approximately 9.6V.

The output voltage is then being feed to ideal diode LTC4353 which is responsible for switching ON/OFF the screen and charger module supply.

iii. ***Ideal Diode***

From the circuit diagram above, the ideal diode has 16 input output pins. The pins labelled EN1 and EN2 are responsible for turning off the supply to the charger and the screen respectively. Once a charger is connected from the mini B port, the diode is switched ON automatically using an NPN transistor connected to EN1 pin, thus when the charger is connected, the EN1 pin is pulled down to ground using the transistor, which leaves the EN1 high (at 3.3V) when the charger is not connected. The EN2 pin is controlled by the MCU.

The block diagram below is provided to give an overview of the functional diagram of an LTC4353 ideal diode, but the internal description is beyond the scope of this report:

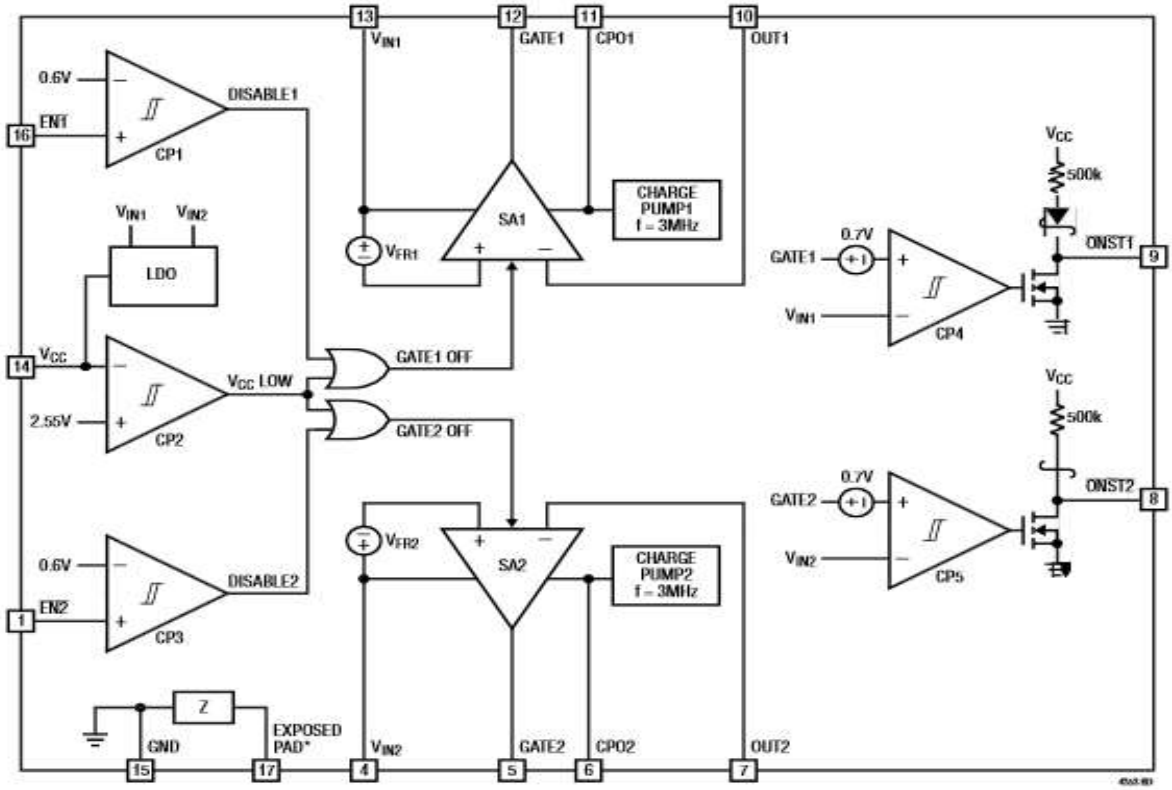


Figure 25: LTC4353 block diagram [25]

iv. ***The charger***

The charger module used is MAX1758 which is a stand-alone charger which can charge up to 4 batteries. This PIC has 28 pins, the input voltage can range from 6V to 28V, making it appropriate to be used for the phone (5V which is boosted to 9.5V) or laptop charger (20V). The voltage comes from either laptop charger or phone charger, the forward bias of this voltages are ensured using two MBRS130 diodes. The line with the thicker width on the circuit diagram below shows the laptop charger voltage line and phone charger voltage line.

The charger circuit diagram is shown below:

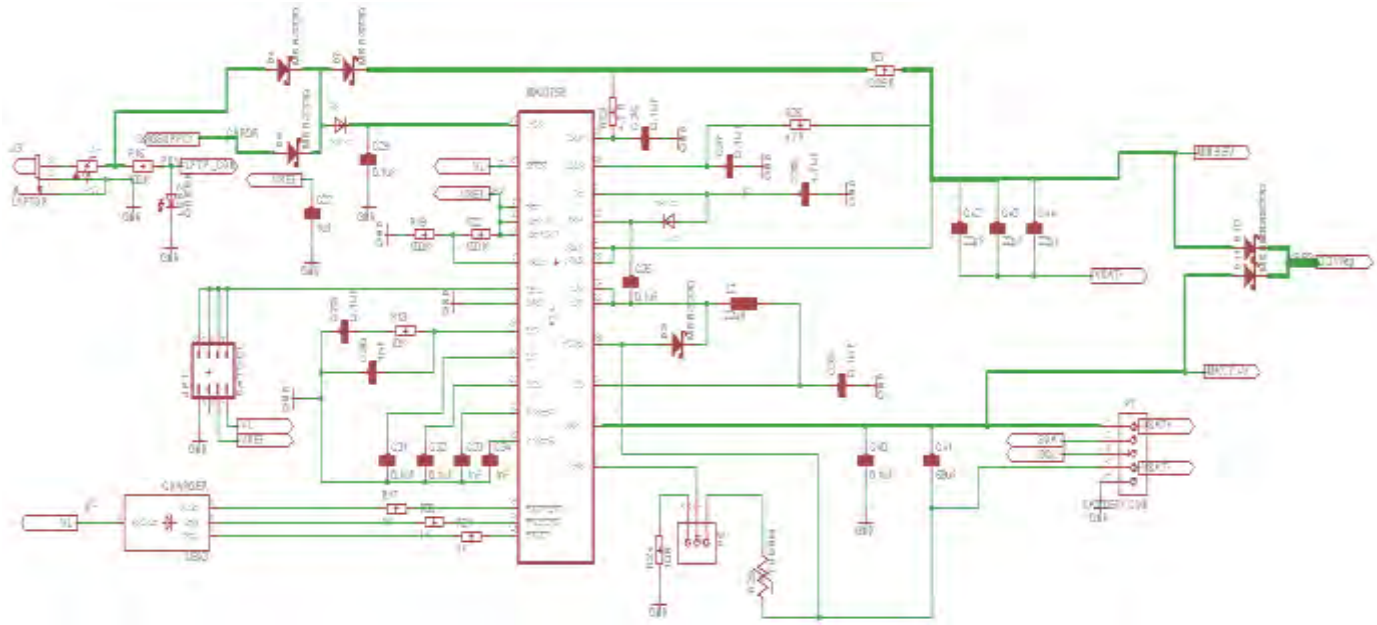


Figure 26: Charger module circuit connection

The description of how the pins are connected on the circuit diagram above can be found on the table below:

Pin No	Name	Function	Connection
1	VL	Power supply	4.7uF capacitor coupling capacitor connected
2	ISETIN	Input current limit Adjust	This is connected to REF to get the full scale input current limit.
3	ISETOUT	Battery Current Adjust	This is connected to REF to get the full scale input current limit.
4	THM	Thermistor Input	10K resistor connected.
5	REF	4,2V reference voltage output	1uF capacitor connected
6	GND	Analog Ground	Connected to GND
7	VADJ	Voltage Adjustment	Two 100K resistors connected from VREF to act as a voltage divider to VADJ.
8	BATT	Battery voltage sensor	Connects to Positive terminal of the battery
9,10	HSD	High side drain	Connected together
11	CELL	Cell count programming Input	Left hanging to allow for 2 batteries charging (AA batteries)
12	TIMER1	Timer 1 Adjustment	1nF capacitor connected
13	TIMER2	Time 2 Adjustment	1nF capacitor connected
14	!FAULT	Charge fault indicator	Red LED with 1k current limiting resistor connected
15	!FASTCHG	Fast charge indicator	Blue LED with 1k current limiting resistor connected
16	!FULLCHG	Full Charger indicator	Green LED with 1k current limiting resistor connected
17	!SHDN	Shutdown Input	Maintained high by connecting it to VL
18	PGND	Power Ground	Connected to batteries negative terminal
19,20	LX	Power Inductor	Connected together in series with 22uH inductor
21	CS	Battery current sense	Connected to ground via 1uF
22	BST	High-Side MOSFET Gate Drive Bia	1uF capacitor connected from this pin to LX
23	CCS	Battery Current-Sense Positive Input	Connected 1uF capacitor from this pin to ground
24	CCI	Battery Charge Current Regulation Loop Compensation Point.	Connected 1uF capacitor from this pin to ground
25	CCV	Battery Charge Current Regulation Loop Compensation Point.	Connected 1uF capacitor from this pin to ground
26	CSSN	Source current sense negative input	Connected 1uF capacitor from this pin to ground
27	CSSP	Source current sense positive input	Connected 1uF capacitor from this pin to ground
28	DCIN	Power supply input	Connected to the voltage supply

Table 7: Charger module connection [21]

Connections on the table above and the resistors, capacitors and inductor values used have been achieved with the aid of Max1758 datasheet.
The charger is able to signify any errors encountered while charging, the error state diagram is shown below:

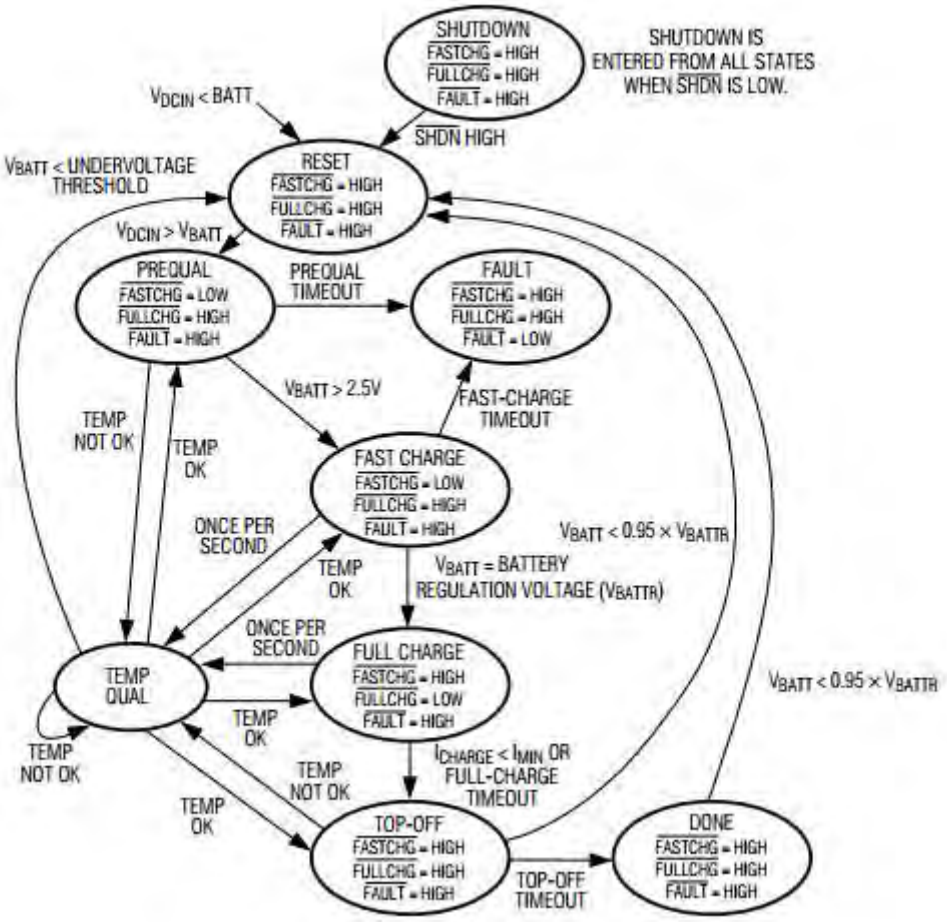


Figure 27: Charger state diagram [21]

v. Voltage Regulator

To supply the control system modules of the device, a 3.3V is required, and a switch voltage regulator is used. The circuit diagram below show the voltage regulator circuit diagram:

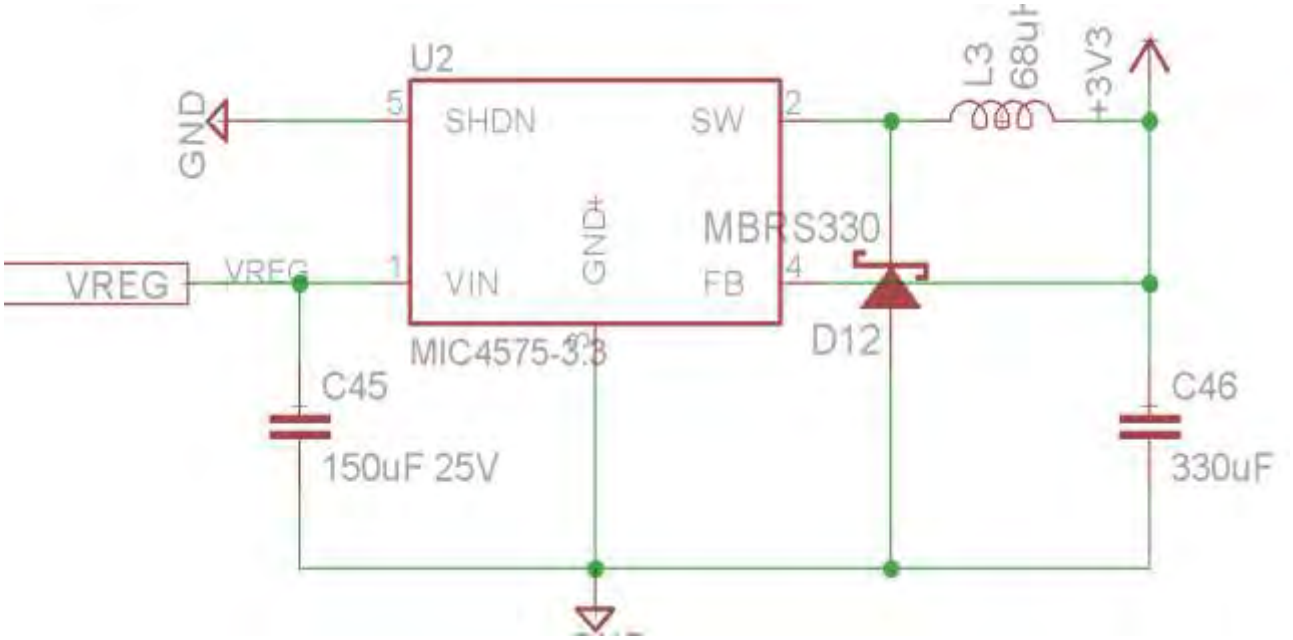


Figure 28: Voltage regulator circuit connection

5.2 The Control system Block

The control system block main components include the microcontroller STM32F107, the GPS and the CO_2 transducer module. The microcontroller receives data coordinates from the GPS receiver NEO6Q and humidity, temperature and CO_2 level from the transducer, and then the MCU stores this data to the SD Card. The microcontroller also awaits the commands from the USB port, to react accordingly. More information about the commands can be found under the software development chapter.

5.2.1 GPS Receiver

The GPS receiver used is NEO6Q. This module has 24 input/output pins and the maximum voltage that can be supplied to this module is 3.6V. Currently it is being supplied from 3.3V via V_{cc} pin, and a 10uF decoupling capacitor to accommodate for voltage ripples. Pins 10,12,13,24 are connected to ground. The V_BCKP pin 22 acts as a backup voltage for the module to enable warm start, since power supplied to this module is being regulated by the microcontroller, thus the MCU turns off and ON this module by turning off the supply to V_{cc} pin, hence a 3.3V is also supplied to the V_BCKP pin with a 100uF decoupling capacitor in place. The passive antenna is connected straight to the RF_IN pin 11. The TX pin is connected directly to the microcontroller RX to allow for communication. Other pins are left unconnected as they are not relevant to this design.

The circuit diagram below shows the GPS module connection:

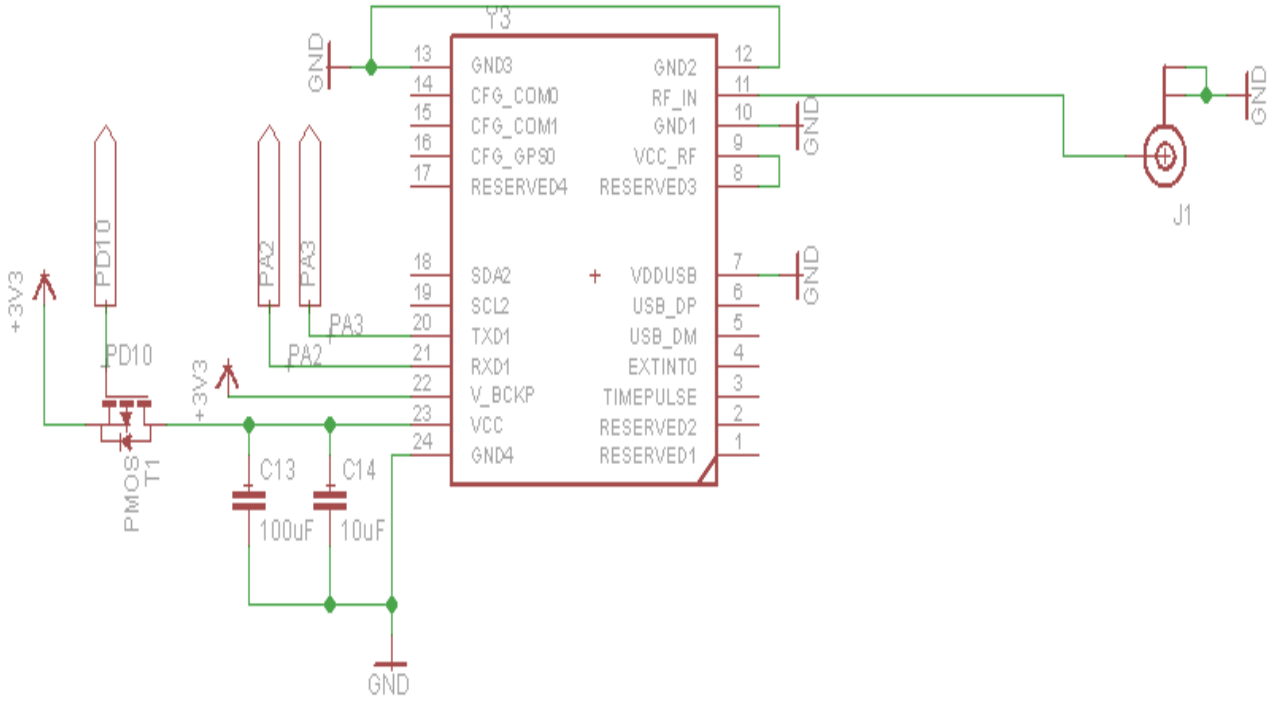


Figure 29: GPS Circuit connection

5.2.2 CO_2 transducer

The module used here is the COZir, which can provide the CO_2 level, the temperature and the humidity. The COZir has 10 pins, of which pins 2, 4, 8, 9 and 10 are not used. The TX pin is connected directly to the MCU (STM32F107) to allow for data transmission. Pin one is connected to GND.

The circuit diagram below show the COZIR circuit diagram:

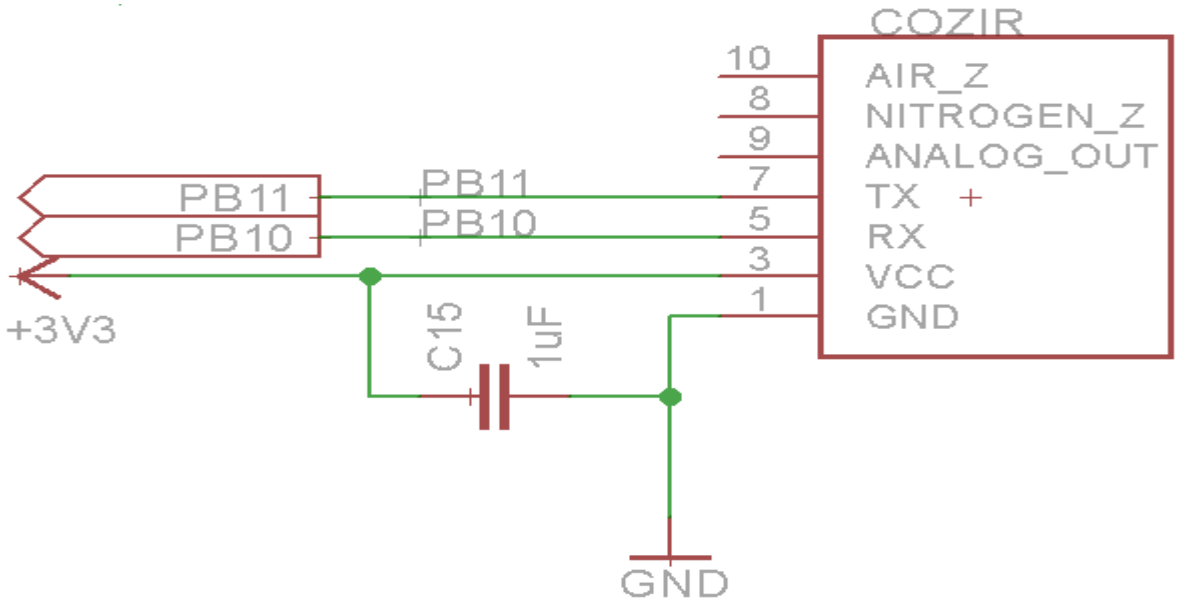


Figure 30: COZir circuit connection

5.2.3 The Microcontroller (MCU)

The microcontroller used is STM32F107VC with 100pins. The MCU is classified under high performance ARM-Cotex-M3 32 bit RISC core, it operates at 72MHz frequency, and it has high speed embedded memory 256Kb. The power supply to this module can range from 2V to 3.6V. It possesses 5 USARTs. [15]

The circuit diagram below shows how the power supply is connected to the MCU:

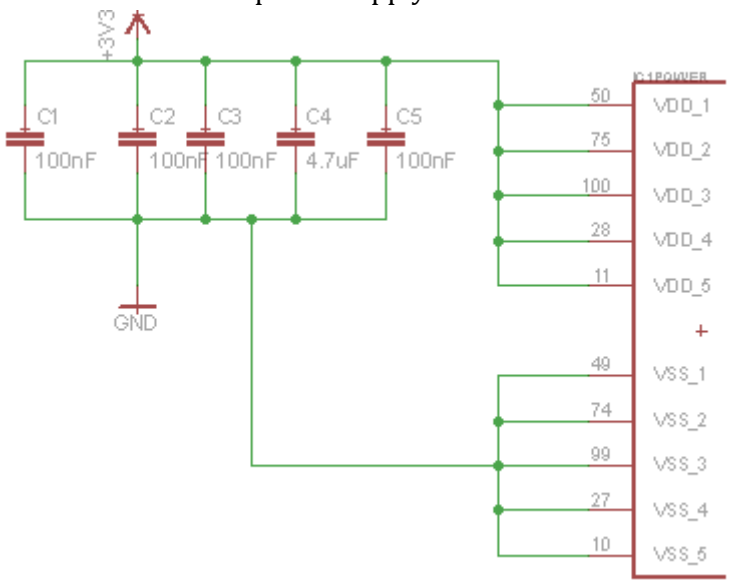
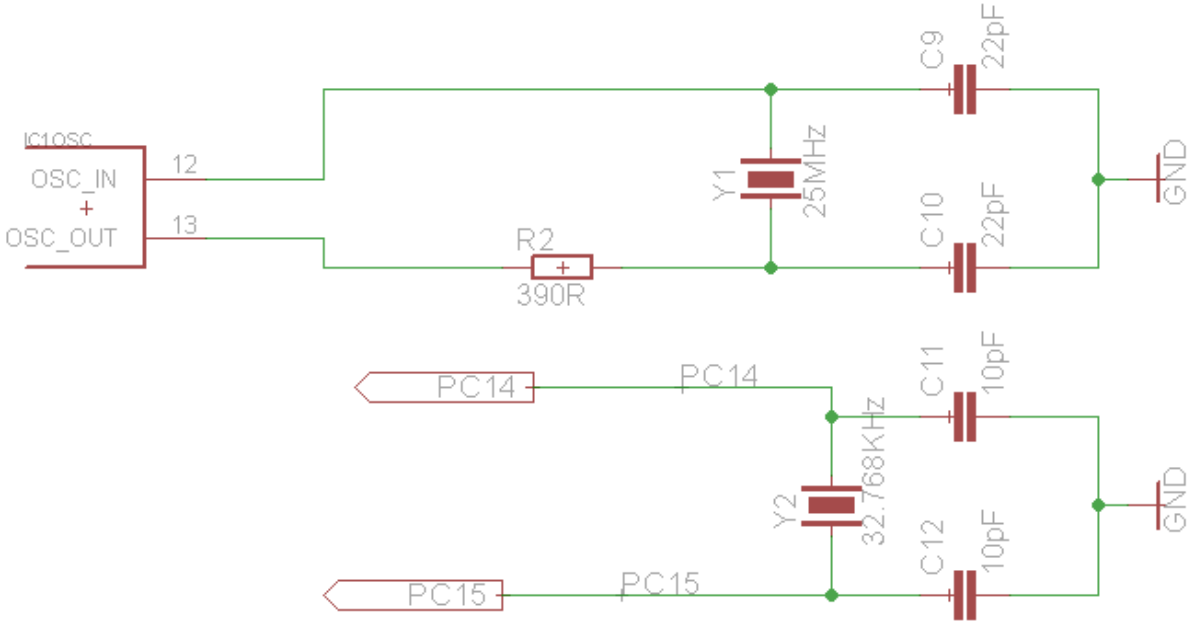


Figure 31:STM32F107VC power connections

From the diagram above, Vss pins (49, 74, 99, 27, and 10) are connected to ground. VDD pins 1, 2, 4 and 5 are connected from 3.3V line, each with a 100nF decoupling capacitor to accommodate for voltage ripples. VDD3 is connected to 3.3V line, and a 4.7uF decoupling capacitor used along. [15]

The two oscillators 25MHz and 32.678 KHz have been used on the device, and are connected to the MCU as shown on the diagram below:



The oscillator 25MHz acts as an internal oscillator for the program most operations, while the external oscillator 32.768 KHz is used in low power mode and to keep the MCU data time.

The MCU also connect to the screen, on which a parallel data mode transmission is being performed. All pin E I/O ports (PE0- PE15) act as data transmission lines, while pin D I/O ports are used as the LCD control lines, more information on how controls lines are used can be found under the firmware chapter. The circuit diagram below shows how the LCD connector is connected to MCU.

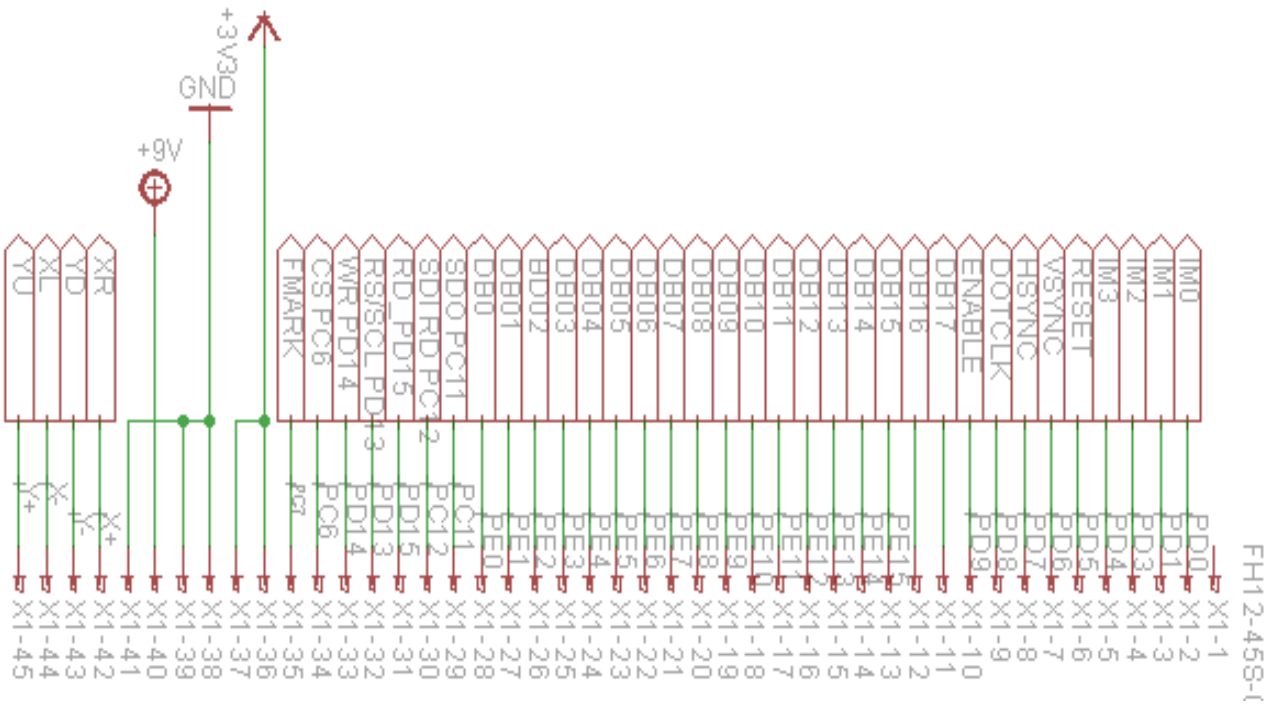


Figure 32: STM32F107 connection to LCD Touch Screen

Additionally an interaction button has been connected to the MCU, the vibration sensor has also been connected to the MCU. These two components aid in the power saving functionality of the device.

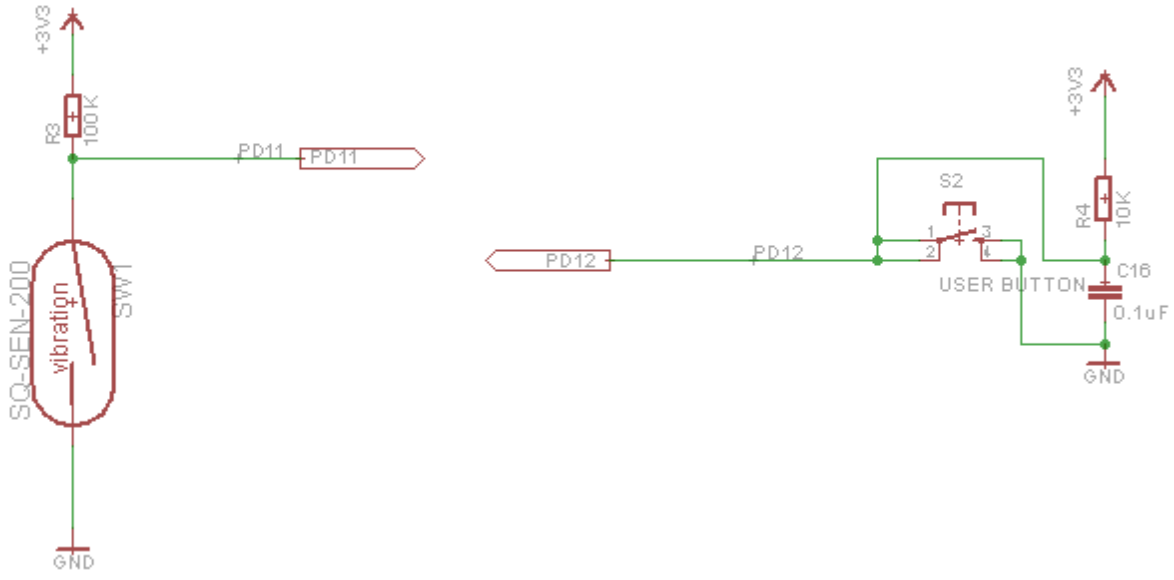


Figure 33: Vibration sensor and Interaction button connection to STM32F107VC

6. Hardware Assembly and Casing

In this chapter system hardware components assembly is described in detail. It initially starts by giving a list of all the components required for the system, then it describes the order at which all the items are being soldered into the board.

6.1 Overview

The device software need to be assembled in the order described in this chapter as a way to ensure that one can be able to spot quickly any component which might bring on unanticipated problems.

The table below shows all the components (Excluding the main components which have already been discussed in the previous chapter) required to build the CO_2 Body worn device:

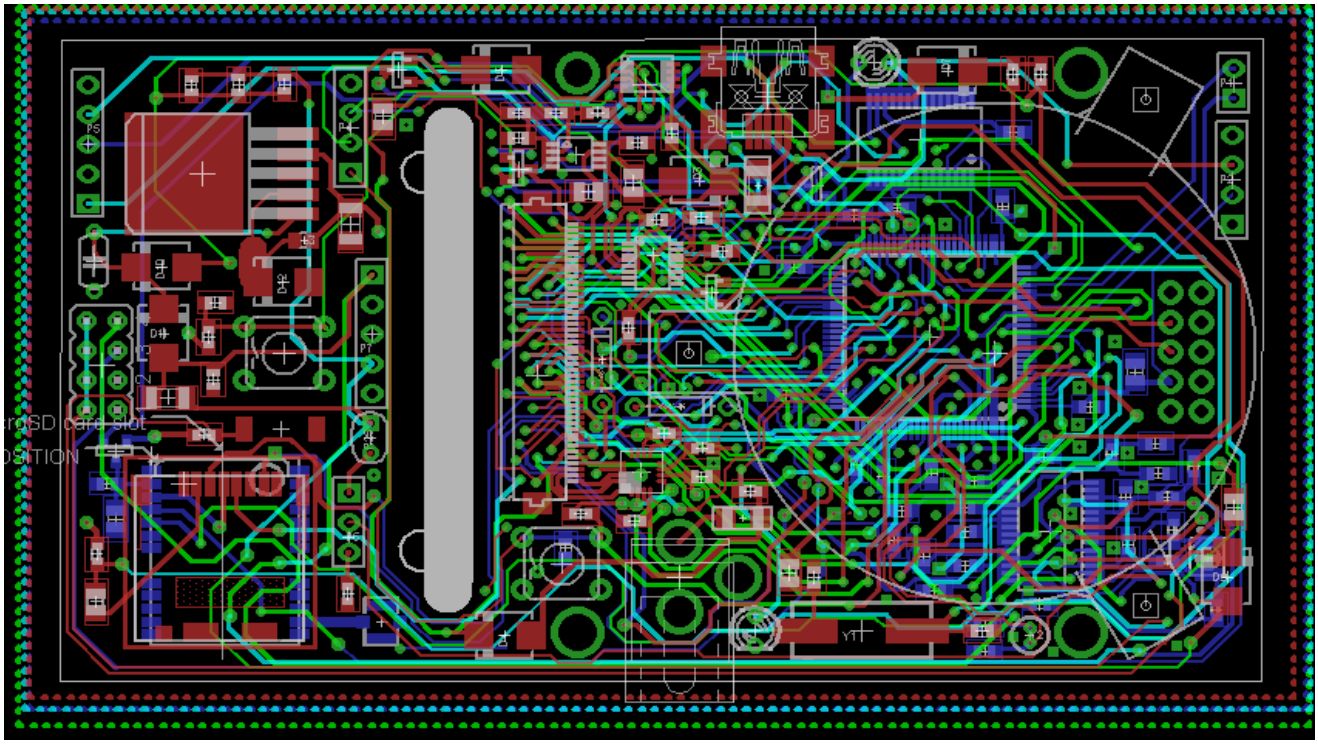
CAPACITORS		
Value	Quantity	Device
150uF 25V	1	C-EU050-024X044
0.1uF	20	C-EUC0603
10pF	2	C-EUC0603
10uF	2	C-EUC0603
18Pf	1	C-EUC0603
1nF	3	C-EUC0603
1uF	1	C-EUC0603
22pF	5	C-EUC0603
27nF	1	C-EUC0603
4.7uF	2	C-EUC0603
56nF	2	C-EUC0603
100uF	1	C-EUC0805
1uF	1	C-EUC0805
33uF	1	C-EUC0805
470pF	1	C-EUC0805
330uF	1	C-EUC1206
68uF	1	C-EUC1206

Table 8: CO2 Body worn device Capacitors list

Resistors		
Value	Quantity	Device
0.05R	1	R-EU_R0603
100K	6	R-EU_R0603
10K	7	R-EU_R0603
10R	1	R-EU_R0603
150K	1	R-EU_R0603
1K	5	R-EU_R0603
4.7R	2	R-EU_R0603
820K	1	R-EU_R0603
0R	1	R-EU_R0805
390R	1	R-EU_R0805

Table 9: CO_2 Body worn device Resistors

The diagram below shows the complete device Board layout and the complete circuit diagram of the device can be obtained in Appendix A:



6.2 Device Casing

The device case was obtained off the shelf. Therefore the device board layout has been designed to fit successfully in the device case. To achieve this, the dimensions of the case have been taken, and the shape of the case modelled using solid works. Then the board layout also modelled on solid works and the simulation performed to ensure that the board will fit in the case, together with the components.

The diagram below shows the case top and bottom:

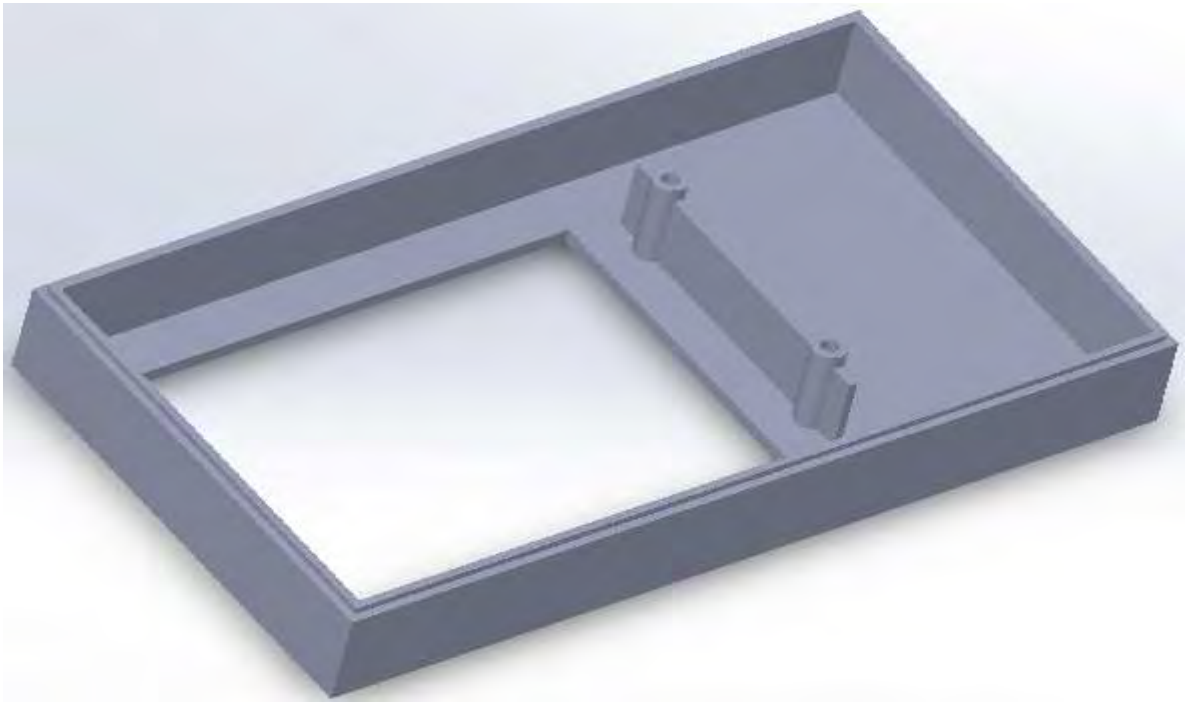
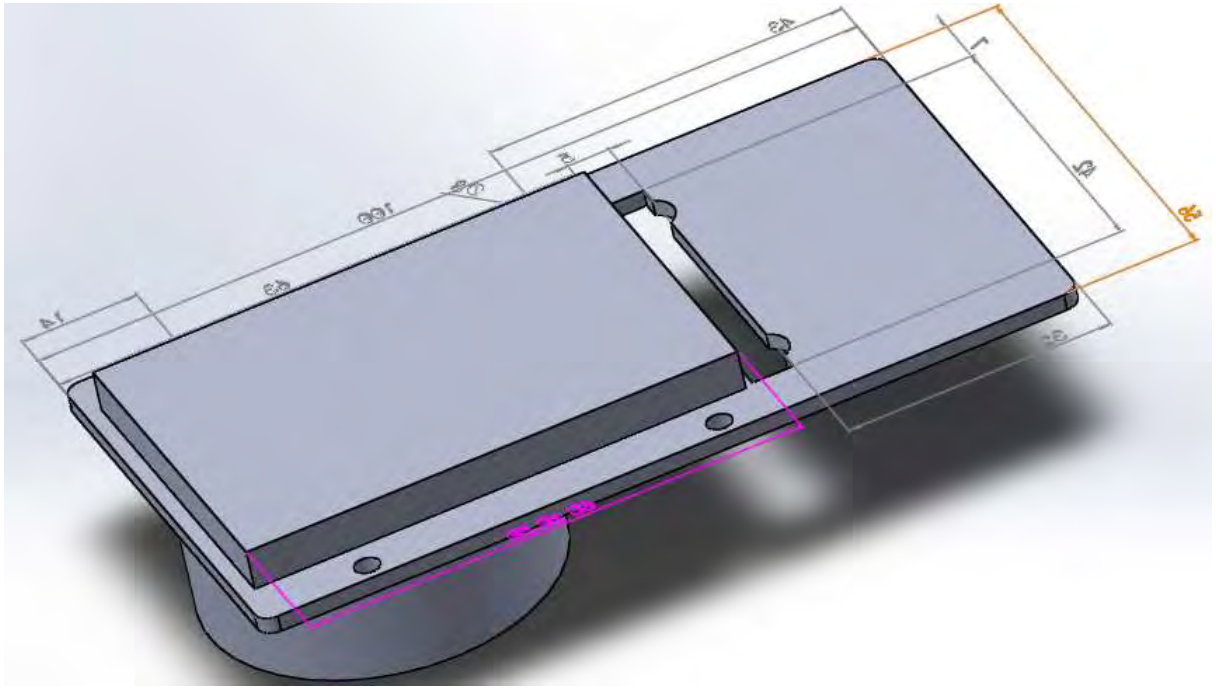


Figure 34: CBWD Top Case cover

Figure 35: CBWD bottom case cover

The PC Board thickness was also modelled on the CAD, and all the components that contribute to the width and thickness also included on the board model. These components are the COZir sensor and the LCD.

The diagram below shows the PC Board layout model with the Coir and LCD on the opposite sites of the board:



Once these components have been positioned on the board, the board is being fit into the case as shown in the diagram below:

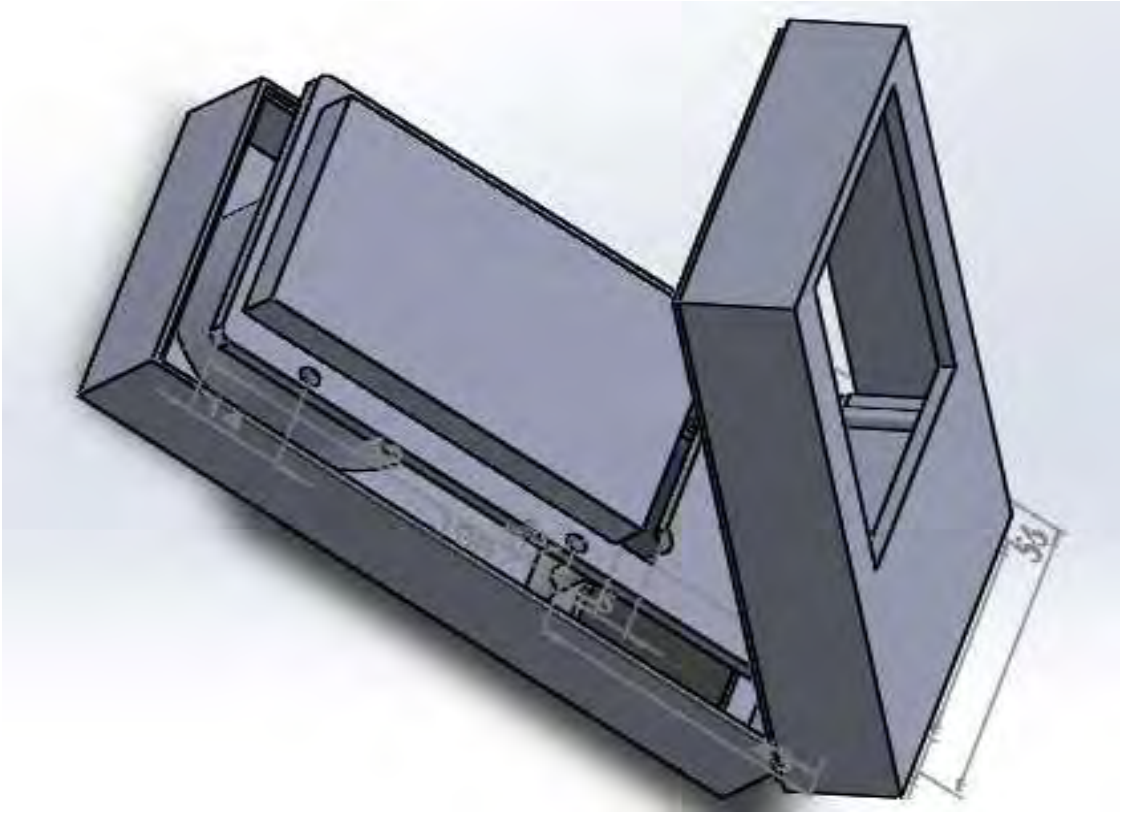


Figure 36: CBWD open top and bottom case with board inside

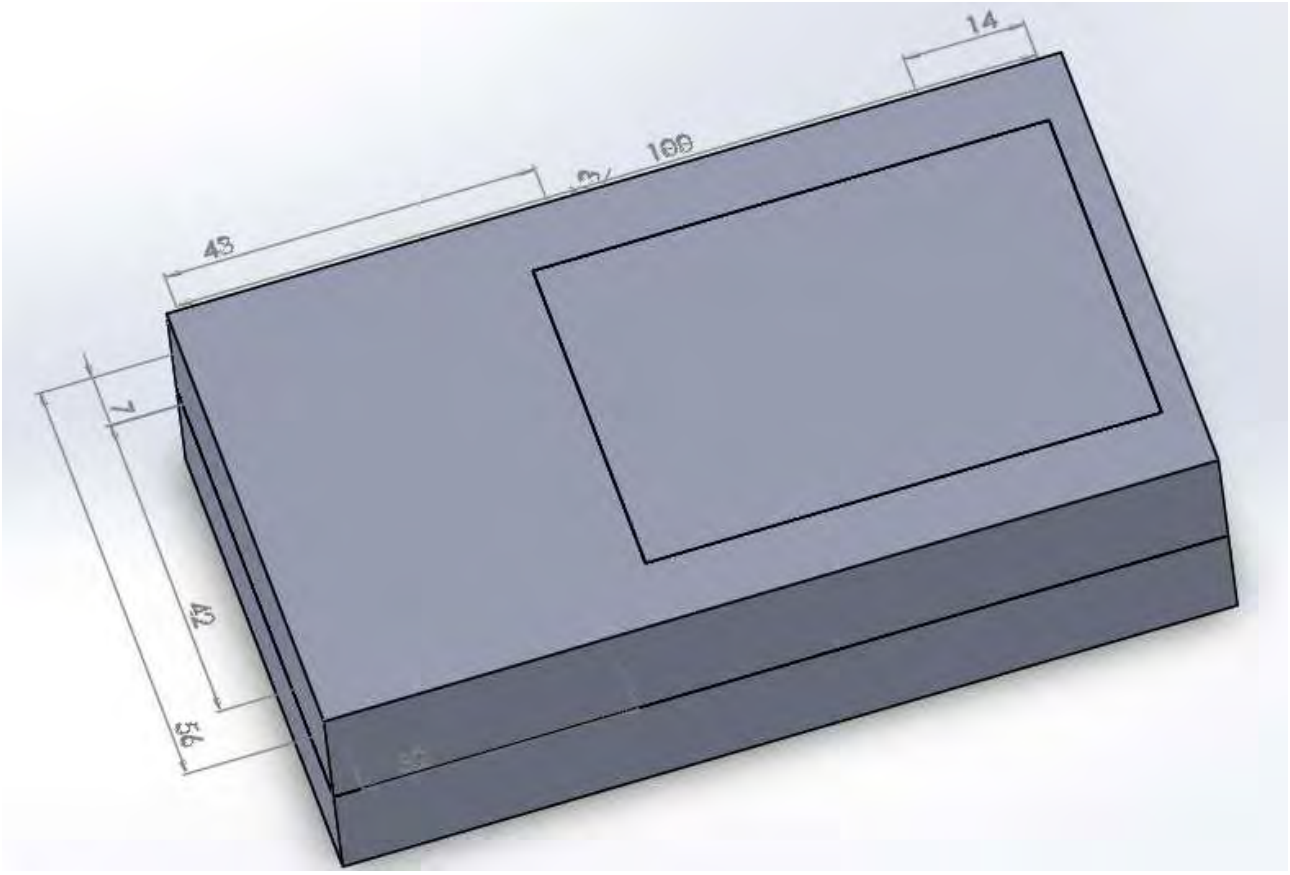


Figure 37 : CDBW Device case closed

6.3 Device modelling, Soldering, and Assembling

To achieve the final version, three sample boards where manufactured, the circuit diagrams of sample 1, sample 2 and the final version are included in the appendix B, C and A respectively. The diagram below shows the trend evolved from sample 1 board to final version board:

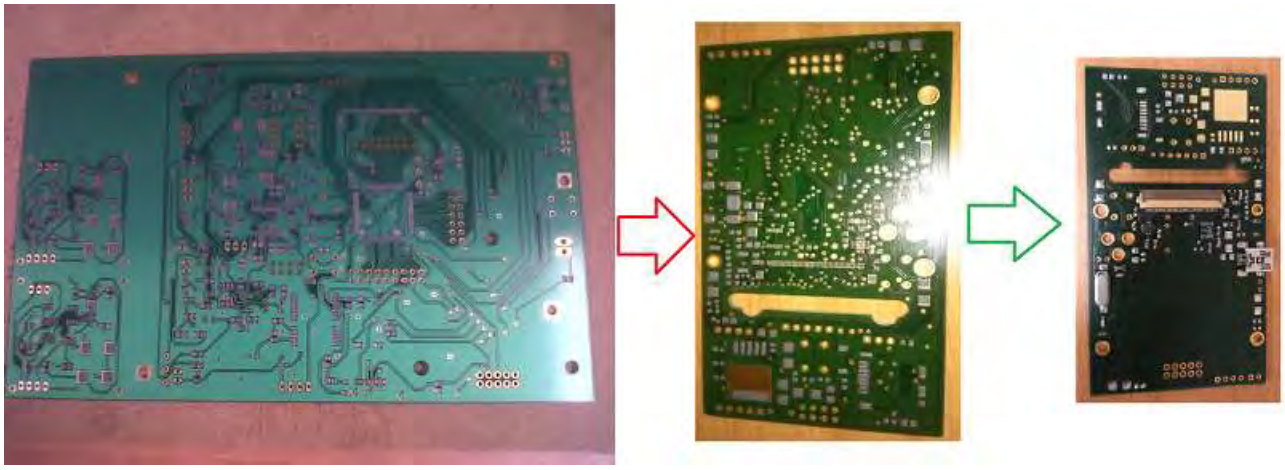


Figure 38: $C0_2$ Body worn device Board trend

It can be noted from the above picture that the board size decreases in the next version.

Sample 1 main purpose was to get all the components onto one board and built the prototype to program. All the hardware bugs obtained on sample 1 were rectified in sample two design. The Sample two design was to put as close as enough components. Sample 2 consisted of the power board separated from the control board. The diagram for sample 2 power board is shown below:

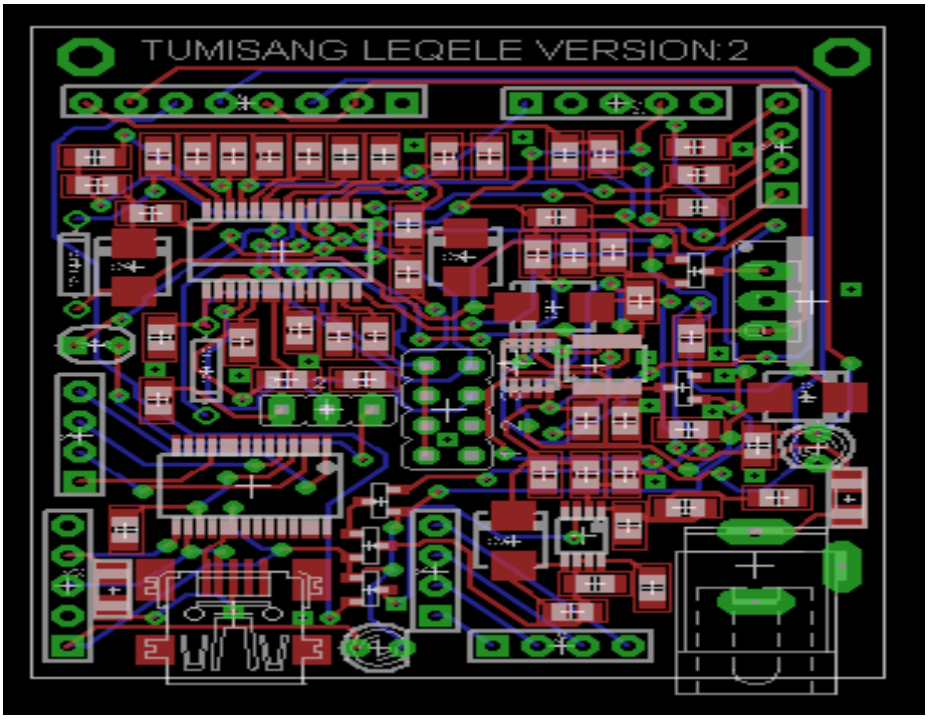


Figure 39: CBWD Model 2 power board layout

The circuit diagram for the Sample 2 power board can be found in Appendix D.

i. Device Soldering

Initially critical components are identified, these are the very small component with close pins, and the board side which has these components is the first side to perform soldering on. In this case, the size with 45 pins Molex screen connector was the first side to be soldered. Initially solder paste is being sprinkled evenly over the board with the help of the board stencil: the diagram below shows the first step:

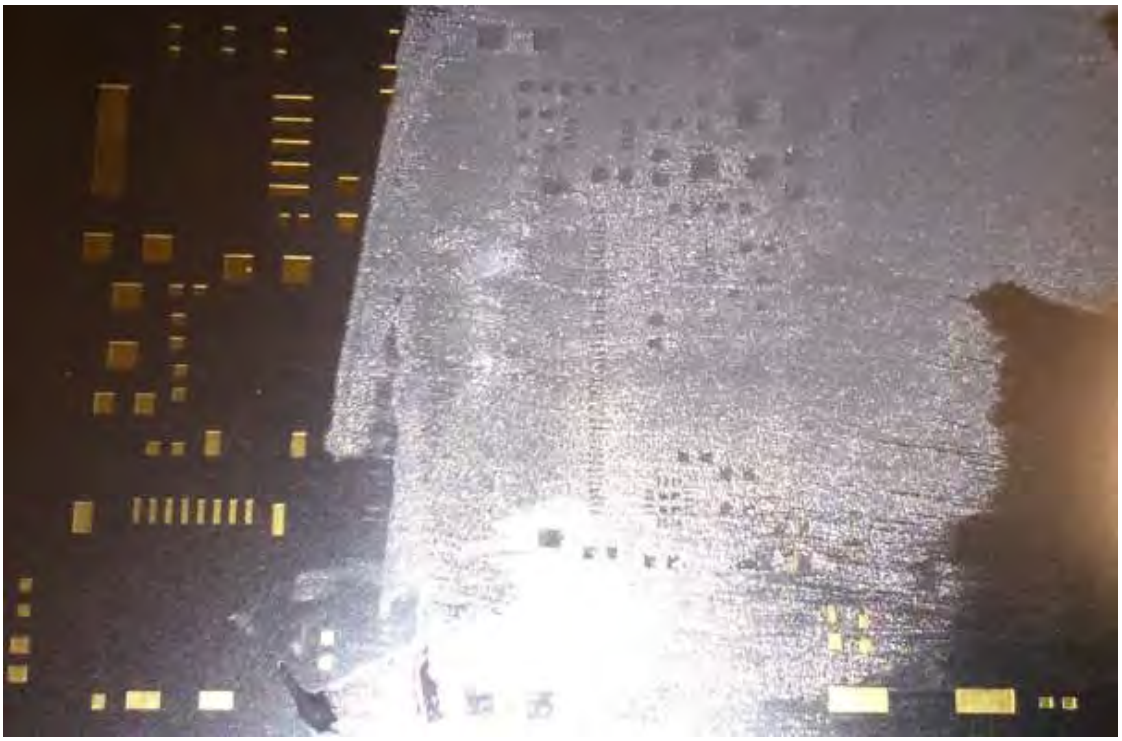
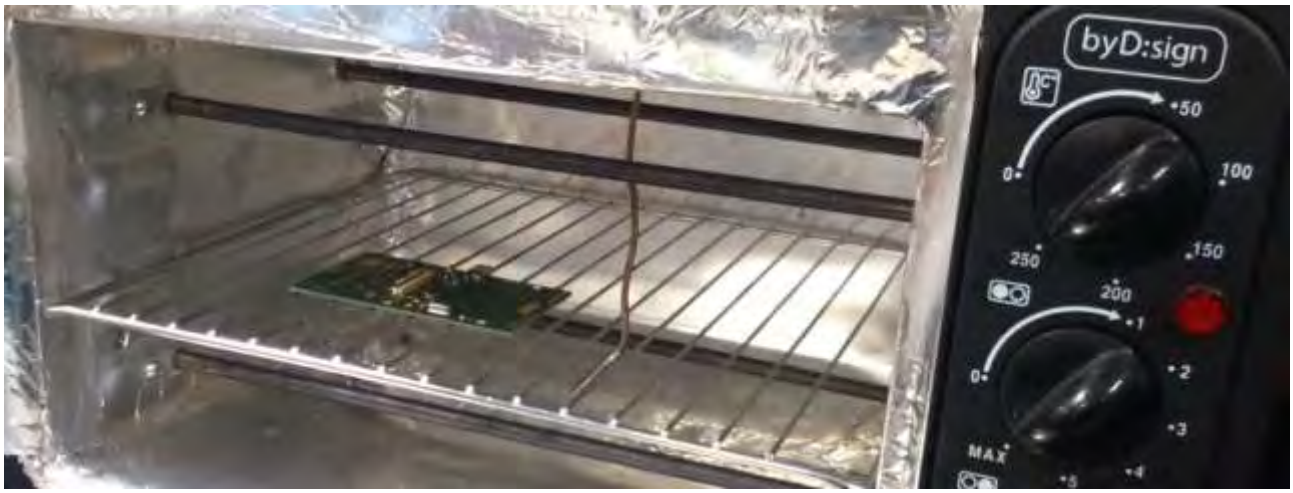


Figure 40: Sprinkling evenly solder paste on CO_2 Body worn device board

Then the components are positioned precisely on their foot print, and an oven is used to melt the solder, with the precaution not to damage the components. The diagram below show the oven about to being heat to approximately 140 ^{0}C:



Then the microscope is used to check on any pins that might be connected together and extra solder is being removed. The critical points or tracks are the power lines, therefore the continuity between these lines is tested to ensure that after soldering these first components there is no earth fault.

There after the main modules are solder and, the testing is performed after soldering each module to ensure that it behaves as expected

ii. ***Step up and switching***

In this circuit, voltage amplifier module is used to amplify input voltage from USB port (5V) to about 9.5V. The complete circuit diagram for these modules can be found in the appendix A. All the components required for this module are stated in the table below:

Component	Value	Quantity	Circuit Symbol
Capacitor	10uF	2	C19, C23
	470pF	1	C20
	18pF	1	C21
	27nF	1	C22
	56nF	2	C25, C26
	0.1uF	1	C24
MAX1790		1	IC3
LTC4353		1	LTC4353
IRF7501		1	IRF7501

These components are all soldered onto the board and the output voltage be measured to ensure that this module behaves as expected.

The diagram below shows the above module components on the board:

Figure 41: Device amplifier and switch

Once this module components soldering is complete, the output voltage has to be measured to ensure that no error has been introduced while soldering.

iii. ***Charger***

The charger module is used to charger Li+ cells, and the device number of cells to charge can be programmed.

The table below shows all the components used to build the charger module:

Component	Value	Quantity	Circuit Symbol
Capacitors	0.1uF	9	C1,C2,C4,C5,C6,C7,C8,C16,C17,C18,C24,C28,C29,C31,C32 ,C35,C36,C37,C39,C40,
	1nF	3	C30,C33,C34,
	1uF	1	C15,C27,
	22pF	3	C9,C10,C42,C43,C44,
	4.7uF	1	C3,C38,
	0.1uF	1	C24
Diodes	1N4148	2	D6,D8,
	MBRS330	6	D4,D5,D7,D9,D10,D11,D12,
MAX1758		1	IC4
Inductor	22uH	1	L2
Resistors	0.05R	1	R27,
	100K	3	R3,R14,R15,R16,R19,R22,
	10K	1	R1,R4,R6,R7,R8,R9,R18,
	10R	1	R24,
	1K	3	R10,R11,R17,R20,R21,
	4.7R	2	R23,R26,
RGB LED		1	LED3

The diagram below show the Charger module components soldered:

Figure 42 : Device Charger module components

Once the charger is properly soldered, if power is supplied, blue LED will turn on.

iv. ***The GPS, COZir and SD Card Holder***

The GPS assists in locating the device, while the COZir assist in measuring the CO_2 level, relative humidity and temperature. The complete circuit diagram for these modules can be found in the appendix A.

The diagram below shows the SD Card holder, Voltage Regulator and COZir soldered in position.

Figure 43: SD Card, COZir and Voltage Regulator position layout

Once these components have been soldered, LCD Screen can be connected and the device is ready to be commissioned.

The diagram below show the complete device:

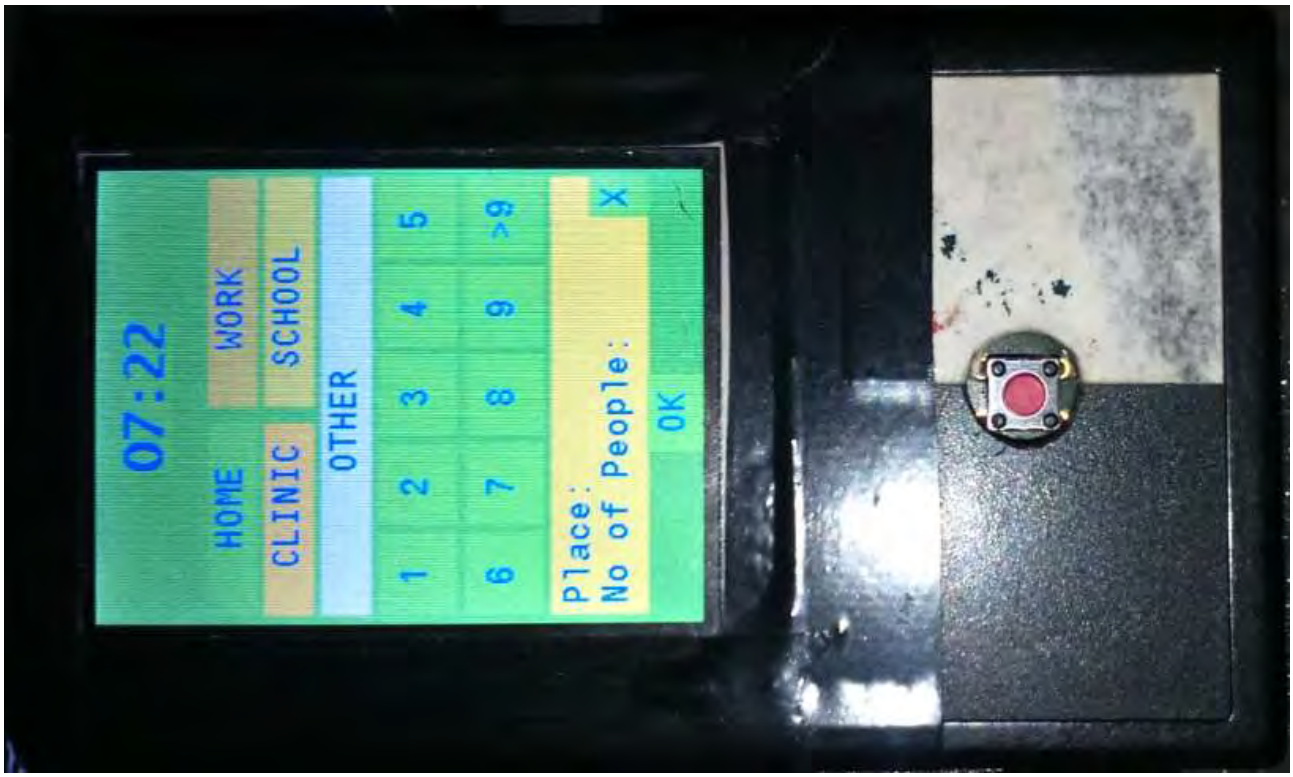


Figure 44: CO2 Body Worn Device

7. Graphical User Interface Design

In this chapter, a graphical user interface (GUI) is described in details. All the user interactions with the interface and outputs are explained in this chapter.

7.1 System Overview

The main purpose of the CO_2 Body Worn Device software (CBWD System) is to allow the device to communicate with the computer, allowing users to manipulate data on the device, which includes downloading and erasing data log on the device memory. Other functionality includes setting reference values of the device, such as CO_2 level reference.

The diagram below shows the overall system program flow.

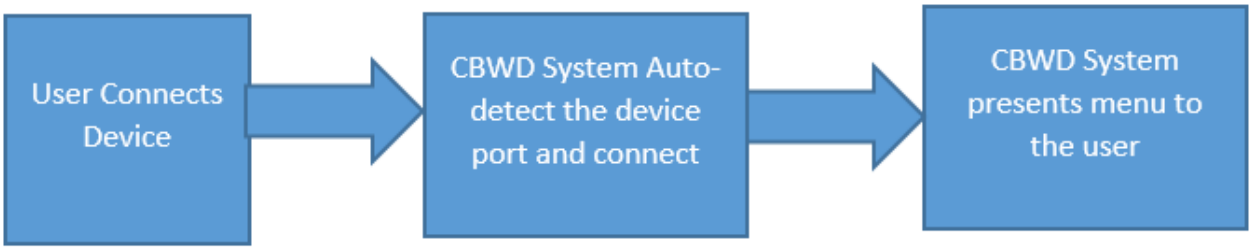


Figure 45: CBWD System program flow diagram

From the diagram above, the user connects the device to the computer, and then starts the CBWD system software, which auto detects the connected device and presents the menu to the user. Then from this menu the user can choose what he wants to do on the device (download log data, set device time, etc.). This is a version 2 of the software, the generation one devices did have the software to help in manipulating their data, therefore all the functionality from the generation 1 software are all retained, with other functionality added. This software can also work with generation 1 device.

7.2 System Architecture

7.2.1 Architectural Design

The main modules which compose the system include:

- Device connection module
- Data analysis
- Device Status
- Live mode

i. Device Connection module

This module is responsible for connection of the device to the computer. When it is started, it scans all the available ports and picks the CO_2 Device if connected. It then presents the menu, which leads to other modules specified above. If the CBWD System fails to recognize the device, it then keeps the list of available ports, and asks the user to manual configure the device connection by simply picking the port at which the device is connected. The other modules are available once this module has been validated.

ii. Data analysis

This module contains the most essential functionality of this device. Initially on this module the command are being issued, this command includes downloading data, setting reference for CO_2 level, synchronizing device time with computer time and resetting all other data values of the device. This module exist in parallel with the other two modules namely device status and live mode, the only difference is all the functions performed on this module need to complete first before one can move to other modules.

iii. Device Status

This module is responsible for checking the device performance status based on historical data and live performance. The battery performance is analysed based on how quickly it discharges. Also system log interval gaps are being analysed to see the processor performance. Also the commands to get GPS data or CO_2 transducer data straight way are passed here, with the purpose of identifying the faults in the system blocks. Therefore this module is responsible for analysing the life of the device. This module also runs in parallel with the data analysis and live mode module.

iv. Live Mode

This module is designed to handle real time GPS and COZir data, therefore this module can be used in performing experiments. The real time graphs of CO_2, humidity and temperature are shown, and the temperature and GPS location coordinates are also displayed in real time on this module. Therefore the commands to derive real time data are being sent on this module.

7.2.2 Decomposition Description

i. Use Case of CBWD System

For the system to perform its functions, they are a number of interactions required from the operator. Initially the user connects the device the computer, thereafter starts the CBWD system software. Once the software has loaded, it will automatically detect the port of the connected device and communicate with it accordingly, in case it fails to connect to the device, it will provide the user with a list of communication ports available. The users can then pic from this list the device port.

The diagram below show the use case diagram of the device:

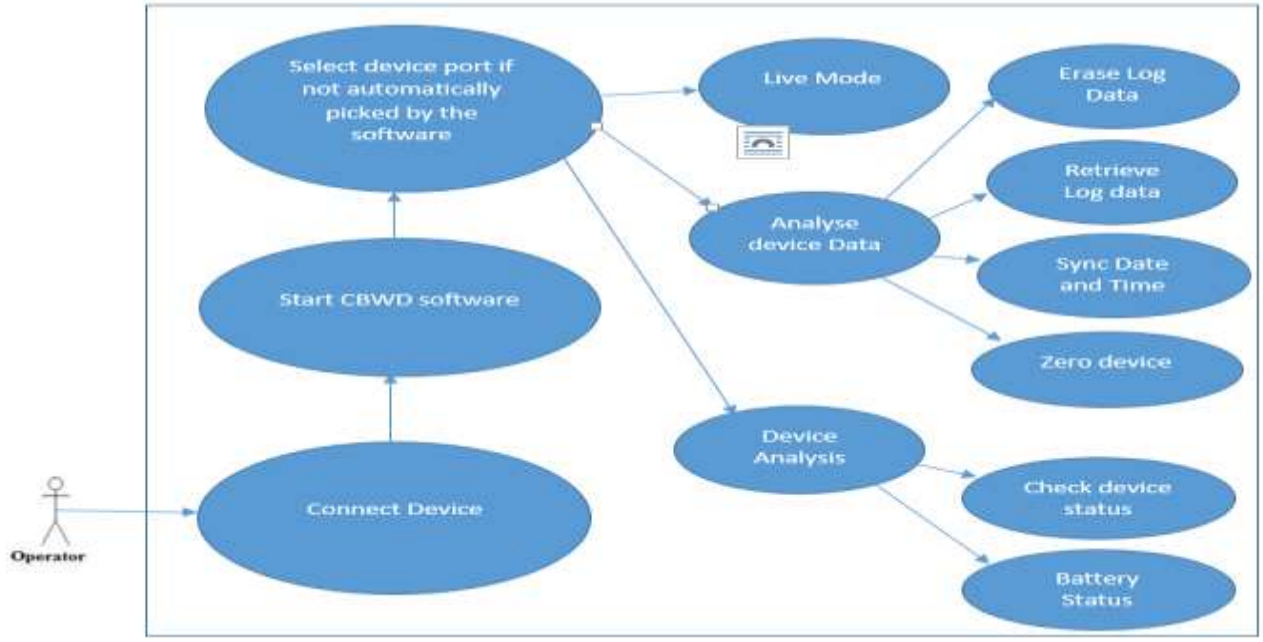


Figure 46: CBWD use case diagram

It can be recognised from the diagram above that the user can perform numerous functions on the software.

The diagram below shows the main menu shown once the device connection has been established.

Figure 47: CBWD System main window

From the above diagram, the user can click on the buttons to exercise the device functionality.

If the user decides to perform data analysis, a window with the option buttons to retrieve log, erase log, synchronise computer date and time with the device date and time, or set the COZir CO_2 reference point. The diagram below show the interface of data analysis:

Figure 48: CBWD System Data Analysis Window

To perform live mode functions, once a user has clicked on the live mode button, the window with an option to start live view, pause and download data is presented to the user.

The diagram below shows the live mode window:

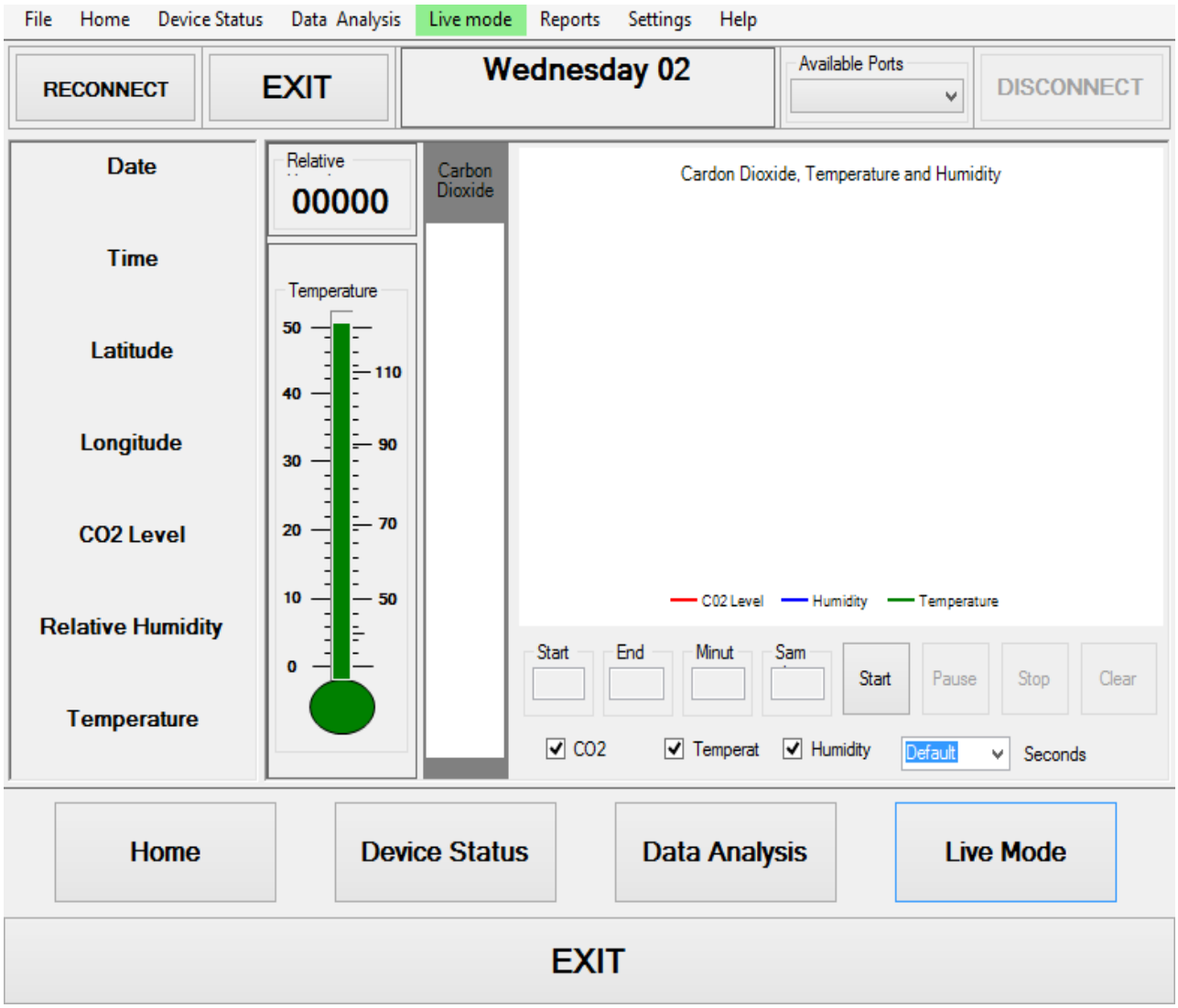


Figure 49: CBWD System Live mode window

7.3 Component Design

This section takes a closer look to the components modules, and provides the components description with pseudo codes and program snip help. The complete code for the software can be found in on the CD attached to this Thesis.

7.3.1 Software Structure

Once the software is started, it checks on the availability of the device, from which if the device is available, it presents the menu to the user to interact with as a way to issue commands to the device. The menu includes data analysis, device analysis and live mode, and the functional structure of these modules has been discussed in the sections above.

The activity diagram of the software system is shown below:

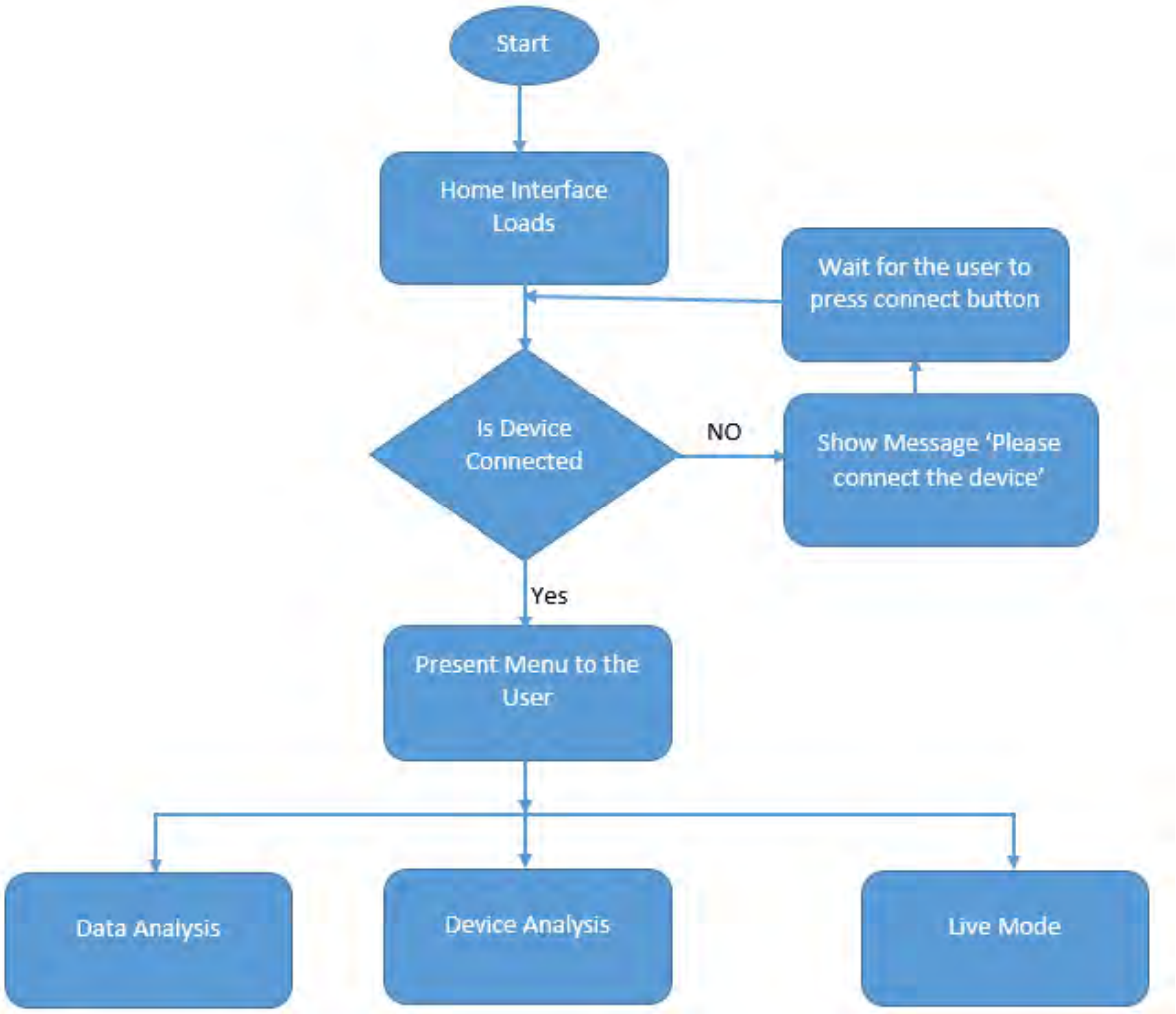


Figure 50: CBWD Software activity diagram

7.3.2 Device Connection

Initially the device is connected, and the CBWD software is started, then the software auto detects the connected device port.
The following pseudo code describes how the system achieves this:

Program start:

- Scan all available ports and load them on list box
- If ports are available
  {
  - For all the ports available
    - Connect
    - If connected
      - Get Device name
      - If Device is known
        - Exit and present menu
  - If not connected to any port
    - Allow user to select the port from the list, connect to the device and present menu

  }
- Else Show error, ask user to configure ports

The following visual basic function shows how to perform the connection of the device

```
Private Function reconnect() As Integer
        Try
            myPortNames = IO.Ports.SerialPort.GetPortNames

            If myPortNames.Length = 0 Then
                MessageBox.Show("There is no device connected, Please connect the device
and click on RECONNECT")
                Reconectbutton.Enabled = True
            Else
                AvailablePorts.Items.AddRange(myPortNames)

                If myPortNames.Length = 1 Then
                    connectedPort = myPortNames(0)
                    SerialPort1.PortName = connectedPort
                    Try
                        SerialPort1.Open()
                    Catch ex As Exception
                        MessageBox.Show(ex.Message)
                    End Try

                    MessageBox.Show("Device connected at " & connectedPort)
                    Reconectbutton.Enabled = False
                    DisconnectButton.Enabled = True
                Else
                    Dim connected As Boolean = False

                    For i = 0 To myPortNames.Length - 1
                        Try
                            connected = False
                            connectedPort = myPortNames(i)
                            SerialPort1.PortName = connectedPort
                            SerialPort1.Open()
                            If SerialPort1.IsOpen Then
                                connected = True
                                Exit For
                            End If
                        Catch ex As Exception
                        End Try
                    Next

                    If connected = True Then
                        MessageBox.Show("Device connected at " & connectedPort)
                        Reconectbutton.Enabled = False
                        DisconnectButton.Enabled = True
                    Else
                        MessageBox.Show("The device failed to connect to any of the
available ports")
                    End If
                End If
            End If
        Catch ex As Exception
            Reconectbutton.Enabled = True
            MessageBox.Show(ex.Message)
        End Try
        Return 0
    End Function
```

Figure 51: Device Connection function

7.3.3 Data Analysis

This menu offers a user the ability to interact with the device to perform the following functions:

- Download data
- Erase log data
- Synchronise date and time
- Set COZir zero reference

The data analysis window waits for the user to press any of the functions, from which it sends the appropriate commands to the device to perform a requested function. The diagram below shows the activity diagram for the data analysis menu:

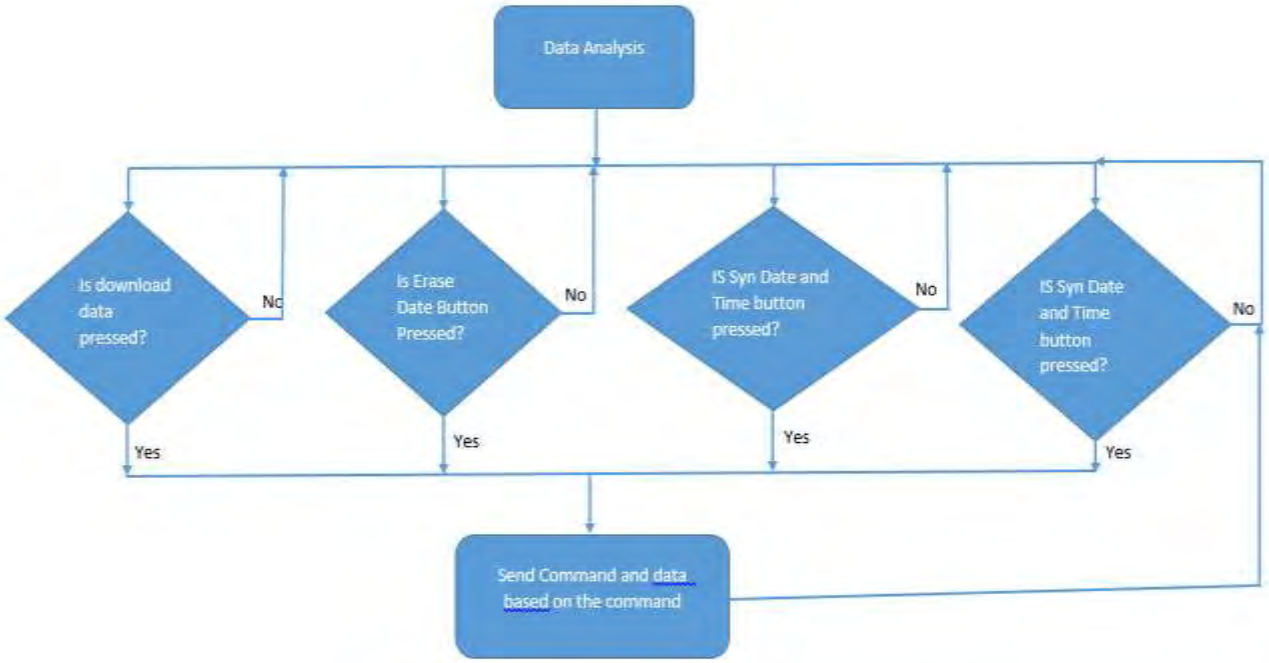


Figure 52: Device Data analysis activity diagram

Each command is being manipulate via single function which is responsible for identifying commands and data to the device. The pseudo code below describes how the commands are being selected and send to the device:

Function SendCommand:

- Enable transmission port and wait for it to be ready to transmit
- Send the command
- Turn off the transmission port
- Return false if error, else return true
- Use timers to regulate the number or times the command is send depending on the command being send.

The Code function below performs the above mentioned pseudo code:

```
Private Function SendCommand(ByVal num As String) As Boolean
        Dim suc As Boolean = False
        Try
            SerialPort1.RtsEnable = True
            Do
                Application.DoEvents()
            Loop Until SerialPort1.CtsHolding = True
            SerialPort1.Write(num)
            SerialPort1.RtsEnable = False
            suc = True
        Catch ex As Exception
            MessageBox.Show(ex.Message)
            Timer1.Enabled = False
            suc = False
        End Try
        If passvalue = "5" Then
            Timer2.Enabled = True
        End If
        Return suc
    End Function
```

Figure 53: Send command and data function

For every command send to the device, the software listens for the feedback from the device and informs the user of the end results. It also logs the end results on its local database.

The function below performs this functionality:

```
Private Function ReceivedText(ByVal [text] As String) As Integer
        Try
            If Me.CO2Level.InvokeRequired Then
                Dim x As New SetTextCallBack(AddressOf ReceivedText)
                Me.Invoke(x, New Object() {(text)})
            Else
                If passvalue = "2" Then
                    If [text] <> "" Then
                        actionresults = "Log data erased sucessfully"
                        commnts = ""
                        LogActionLogTableTableAdapter.Insert(deviceName, DateTime.Now,
actionperformed, actionresults, commnts, "DESMOND TUTU")
                        MessageBox.Show("Data erased successfully.")
                        enable_buttons()
                    End If
                ElseIf passvalue = "C" Then
                    If [text] <> "" Then
                        actionresults = "COZIR set to zero reference sucessfully"
                        commnts = ""
                        LogActionLogTableTableAdapter.Insert(deviceName, DateTime.Now,
actionperformed, actionresults, commnts, "DESMOND TUTU")
                        MessageBox.Show("COZIR has been set to zero reference.")
                    End If
                Else
                    If [text] <> "" Then
                        contentsbuffer.Items.Add([text])
                    End If
                End If
            End If
        Catch ex As Exception
        End Try
        Return 0
    End Function
```

Figure 54: CBWD End results function

7.3.4 Live Mode

In this mode, the real time results of the device log data are been recorded and graphically displayed on the system to the user. To achieve this, the system continuously sends a request live data at specific intervals until the user issues a stop or pause function.

The pseudo code is provided below:
Live mode:

- Do
  - Send get live data results
  - Listen to the data and display it on the screen
- While there is no error in device connection and communication and the user has not pressed stop or pause button.

The activity diagram for the live mode is provided below:

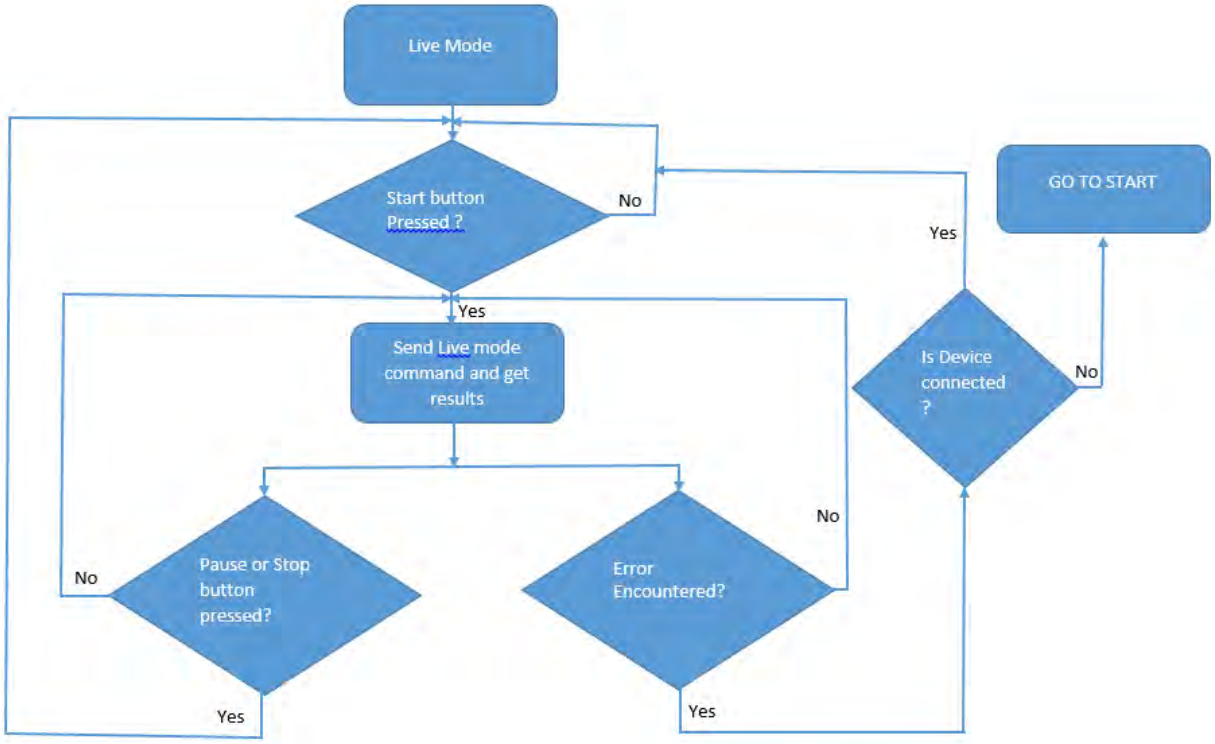


Figure 55: Live mode Activity diagram

8. Firmware

In this chapter, the MCU programming is described in details. It initially start with describing the system architecture, thus the architectural design and the system decomposition is described in details. The complete firmware code can be found on the CD attached to this Thesis.

8.1 System Overview

The main purpose of the MCU (STM32F107) is to control all the functions performed by the device. The main function of the device includes:

- Logging CO_2 Level, relative humidity and temperature
- Log on location coordinates
- Storing and retrieving log data
- Listening to the commands from CBWD software and act accordingly
- Display live data

The system program flow is provided below:

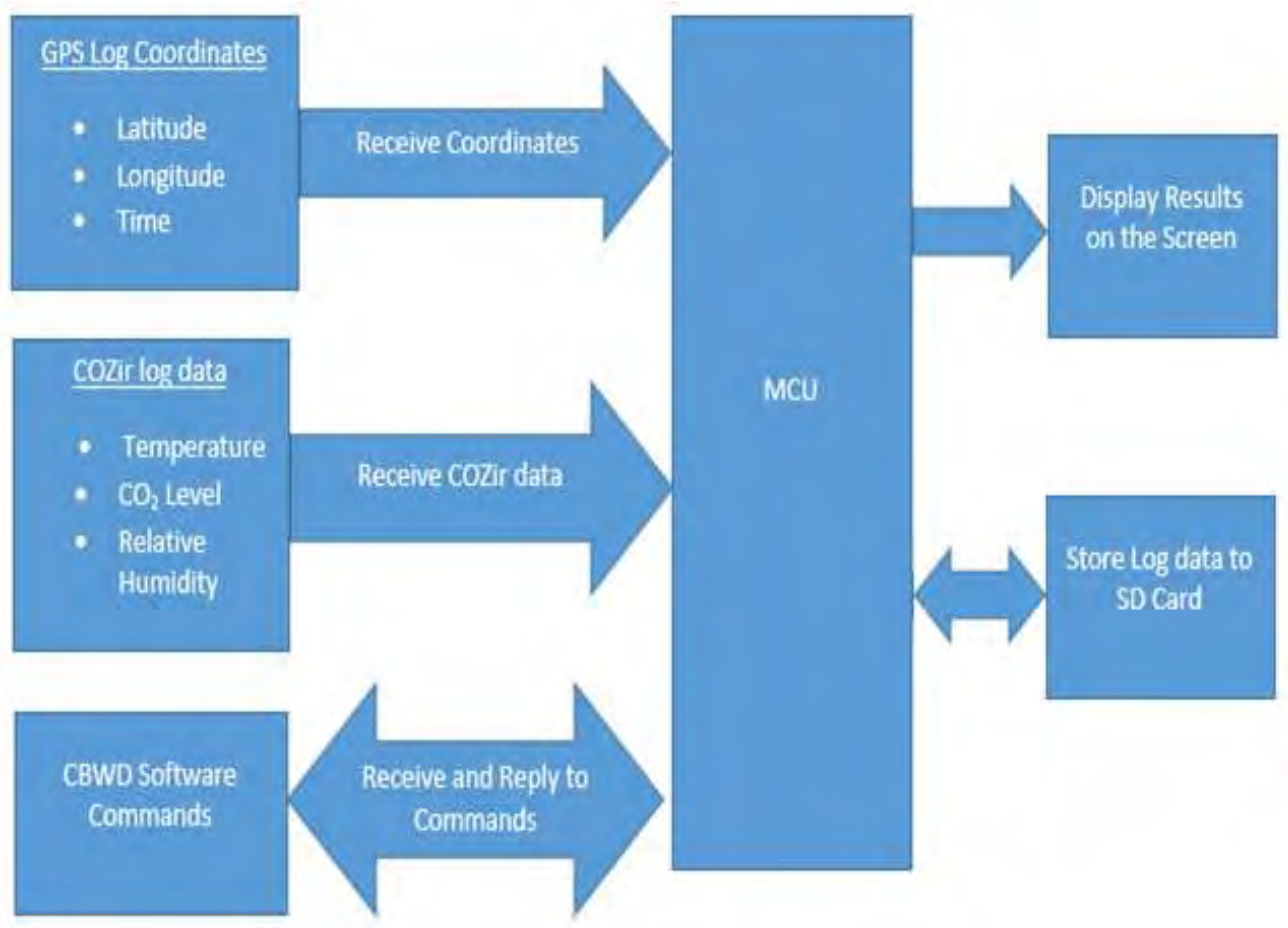


Figure 56: MCU Flow diagram

8.2 SYSTEM ARCHITECTURE

8.2.1 3.1 Architectural Design

In this section, the device is broken into modules from which each module is being described and how the modules combine to make the entire device.

i. Microcontroller

The MCU is responsible for coordinating all the functions which take place on the device. The functions that occur in the MCU take place as either interrupts or normal. These functions include reading the data from the GPS and the COZir, then writing the data to the SD Card.

The other responsibilities for the MCU includes manipulating power consumption of the device, by turning ON and off the LCD, and also by turning ON and off the GPS based on the device movement. All the commands before being processed are handled by the MCU, and the MCU decides on the response to the commands. Therefore the MCU acts as a "heart" of the device by regulating every action which takes place on the device

ii. GPS Module

The GPS module is responsible for locating the device. The GPS data is being send to the MCU, from which the MCU extracts the appropriate GPS data, thus the GPS sentence (GPRMC) which contains the latitude and the longitude as well as time and speed.

The diagram below show the GPS sentences:

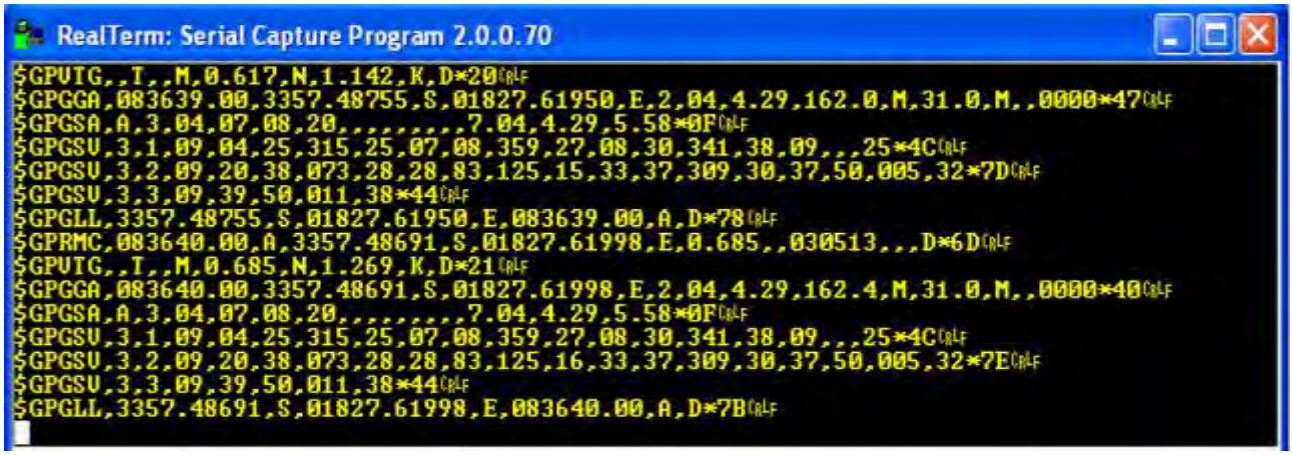


Figure 57: GPS Sentences

iii. COZir module

The COZir module is responsible for providing the CO_2 level, the humidity and the Temperature. The data from this module is being sent to the MCU, from which the MCU adjusts the log CO_2 value based on the set reference value. The MCU also ensures that it extracts the correct data sentence from this component.

iv. SD Card

The SD card is responsible for storing device log data. The MCU writes and reads data from the SD card. The data format is stored on the SD cards as follows:

CO_2 level, Relative Humidity, Temperature, Latitude, Longitude, GPS Time, Device Time

v. LCD Screen Display

The LCD screen is incorporated on the device to show real time log data, and display all the necessary messages for the device. The messages includes when the device is charging etc. Additionally the LCD provides the keypad to the user where he can be able to input number of people a certain location a specific time. The locations are categorised as home, work, clinic and school.

The diagram below shows the user menu on the device:

Figure 58: CBWD User interaction menu

vi. Communication Ports to CBWD software

The commands are being received, and the data required being send back via the communication ports to the CBWD software.

8.3 Decomposition Description

8.3.1 Microcontroller

The MCU reads the COZir data through the interrupt. The data refreshes twice per second [19]. Then the MCU extracts the $C0_2$ level, the relative humidity and the temperature. Then it adjusts the CO_2 level to store on the SD Card based on the reference value set (Value stored = Value Read – Reference value).

The MCU reads the GPS data through an interrupt, but the GPS data refreshes every second, hence it the GPS interrupt is given high priority than the COZir data to allow data coordinates update to occur. From the GPS sentences, the MCU only reads the GPRMC data and ignores any other type of data, from this sentence it extracts the longitude, latitude and time, which is being send to the SD Card for storage. The MCU also turns off or on the GPS based on whether the device is at the same location or it is moving.

The device time is being handled by the external clock. Thus the time values are only read by the MCU from the external clock, while all other modules run on the internal clock.

The MCU sends formatted log data to the SD card for storage.

The MCU is also responsible for turning ON and OFF of the LCD screen. The LCD screen is turned on based on the user requirements, thus if the user presses the button, the MCU turns on the LCD screen for a limited time (60 seconds),unless the user is still interacting with the device.

The MCU also handles commands from the CBWD software. The commands are shown in the table below including the response from the MCU:

Command	MCU Response
Download data	• Stops all the interrupts and sends all the log data to the CBWD system software • It also check for errors in connections to the software, from which it stops transmitting data if the device is disconnected
Get Live data	• Sends the live data from the GPS and the COZir and waits for the next command
Set time and date	• Set the time and data to the required values
Zero the COZir	• Updates the COZir reference value to the current value measured from the COZir
Erase Log data	• Deletes the log file from the SD card and start new log

Table 10: CBWD Commands

The Use case diagram of the device is provided below:

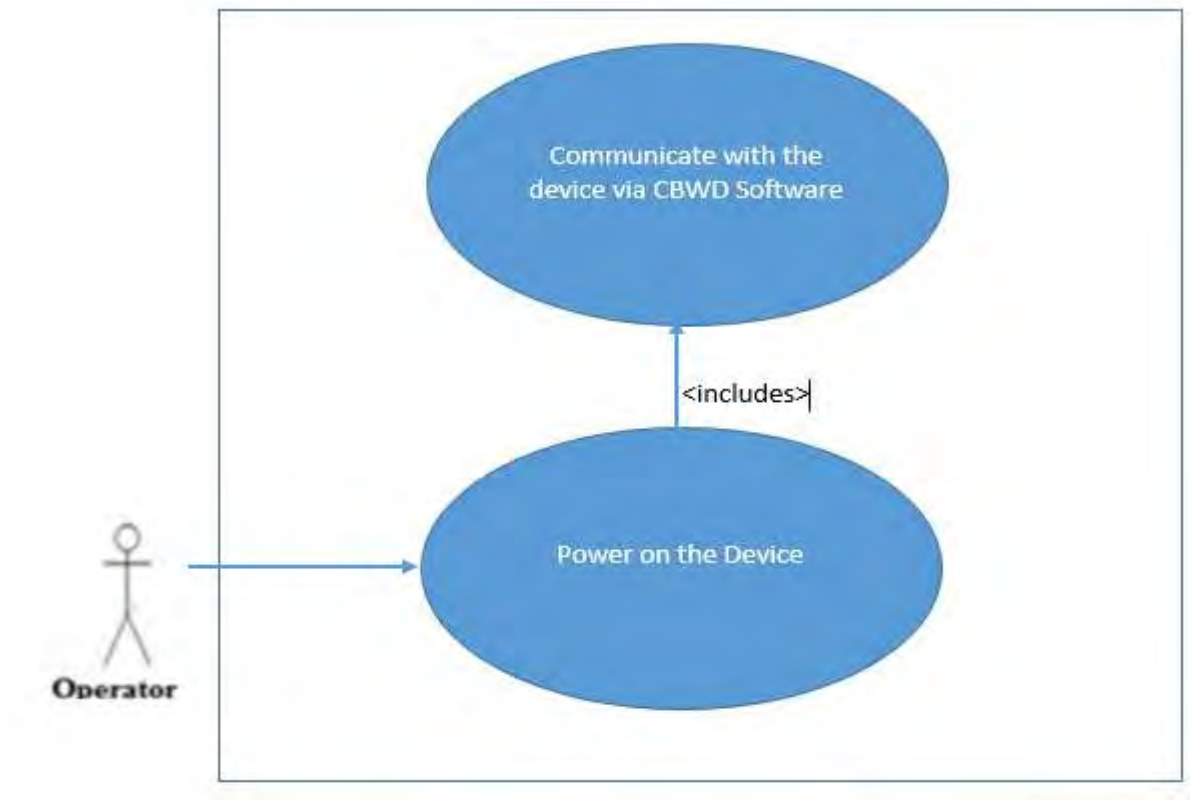


Figure 59: Device Use case diagram

8.3.2 GPS Module

The GPS updates the latitude, Longitude and time value via an interrupt. The transmission ports are initialised, and the interrupt schedule of the GPS line activated. The complete code for the GPS class with all the functions can be found in the firmware package on the CD attached to this thesis. When the interrupt is initiated, the GPS class updates the GPS and sets it ready to be read by the MCU class.

The OOP diagram of the GPS class is provided below:

GPS

`int GPRMC;`
`int i;`
`char GPSTempData[255];`
`char  GPSFinalData[255];`
`int GPS_siz;`

`char * getGPSData(void);  // get the GPS data`
`int getGPSDataSize(void);`
`extern void initialiseVariablesGPS(void);`
`extern void RCC_Configuration_GPS(void);`
`extern void GPIO_Configuration_GPS(void);`
`extern void USART_Configuration_GPS(void);`
`extern void NVIC_Configuration_GPS(void);`
`extern void USART2_IRQHandler(void);  //GPS interrupt handler`

Table 11: GPS OOP Diagram

8.3.3 COZir Module

The COZir class prepares and updates the CO_2 level, the temperature and the relative humidity values for reading by the MCU. It update these values via an interrupt. To achieve this functionality, the COZir is initialised and the TX and RX communication ports configured. The COZir class is responsible for extracting the required variables from the COZir data sentence and gets it ready for the MCU reading.

The OOP diagram for the COZir class is provided below:

COZir

`char COZIRTempData[255];`
`char COZIRFinalData[255];`
`int COZIR_siz;`
`int k;`

`extern void initialiseVariables_COZIR(void);`
`extern void RCC_Configuration_COZIR(void);`
`extern void GPIO_Configuration_COZIR(void);`
`extern void USART_Configuration_COZIR(void);`
`extern void NVIC_Configuration_COZIR(void);`
`extern int getCOZIRDataSize(void);`
`extern char * getCOZIRData(void);`

Table 12: COZir OOP Diagram

8.3.4 SD Card module

The SD card stores the device data for future reference and data retrieval. To accomplish this functionality, the SD card communication ports are being initialised, and data transmission and acknowledgements being configured. The details of how the configuration takes place is explained on the next section, the OOP diagram shown below has been provided to give an idea of the class composition of variables and functions which do SD card manipulation:

SD Card

u8 MSD_Init(void);
u8 MSD_WriteBlock(u8* pBuffer, u32 WriteAddr, u16 NumByteToWrite);
u8 MSD_ReadBlock(u8* pBuffer, u32 ReadAddr, u16 NumByteToRead);
u8 MSD_WriteBuffer(u8* pBuffer, u32 WriteAddr, u32 NumByteToWrite);
u8 MSD_ReadBuffer(u8* pBuffer, u32 ReadAddr, u32 NumByteToRead);
u8 MSD_GetCSDRegister(sMSD_CSD* MSD_csd);
u8 MSD_GetCIDRegister(sMSD_CID* MSD_cid);
u8 SD_INFORMATION(sMSD_SDCARD * SDCard);
void MSD_SendCmd(u8 Cmd, u32 Arg, u8 Crc);
u8 MSD_GetResponse(u8 Response);
u8 MSD_GetDataResponse(void);
u8 MSD_GoIdleState(void);
u16 MSD_GetStatus(void);
u8 writeDeviceDataToMemory(sMSD_SDCARD * SDCard);
void MSD_WriteByte(u8 byte);
u8 MSD_ReadByte(void);

Table 13: COZIR OOP Diagram

8.3.5 LCD Screen

For the LCD screen to display the data, the LCD screen mode of transmission needs to be set first, thus if its data is being send through RX TX ports or in parallel mode. On this device the parallel mode data transmission is used. The explanation on how the configurations are performed is provided on the next section. Additionally the user can be able to input the number of people at a specific location using the LCD Touch screen.

The OOP diagram for the LCD screen is shown below:

LCD

`extern void LCD_init(void); //initialises the LCD ports`
`extern void LCD_Clear (unsigned short color);`
`extern void GLCD_init(void);`
`extern void LCD_PutChr(unsigned short x, unsigned short y,unsigned char c,LCD_FontDef_t *font,unsigned short foreground,unsigned short background);`
`extern void LCD_DrawChar(unsigned int ln, unsigned int col, unsigned char c);`
`extern void LCD_PutStr(uint16_t x, uint16_t y, char *str, LCD_FontDef_t *font, uint16_t foreground, uint16_t background,unsigned short str_siz) ;`
`extern void LCD_UpdateDateTime(uint8_t sec);`
`extern void LCD_UpdateCOZIR(char * COZIR,int siz);`
`extern void LCD_UpdateGPS(char * COZIR, int siz);`
`extern void LCD_UpdateScreen(uint8_t sec,LCD_Data_t GPS, LCD_Data_t COZIR);`
`extern void LCD_init_Vars(void);`
`extern void LCD_PutTime(char * Mytime, uint8_t siz);`
`extern void LCD_PutDate(char * Mytime,uint8_t siz);`
`extern void LCD_DisplayGPSData(void);`

Table 14: LCD OOP Diagram

8.4 COMPONENT DESIGN

8.4.1 The Microcontroller

The main function of the MCU is to coordinate all the functions that occur on the device. First the MCU initialises the SD Card, the GPS, the LCD screen and finally the COZir. Then the COZir and the GPS are let to run separately and generate interrupts when the have data. During this time, the MCU checks on the screen interaction as well as ensuring that it writes the log data to the SD card every second, and also it regularly checks on the commands from the communication ports. To achieve this, the MCU performs the following pseudo code in the main function:

Program Start:

- Initialise SD Card
- Initialise LCD Screen
- Initialise GPS
- Initialise COZir
- Forever
  - Check that the SD Card is present
  - Check the interactions on the screen
  - Update the log data and Store data every second
  - Check for commands from the communication ports
    - If commands available disable interrupts accordingly and service the command

The main function is show below:

```
int main(void)
{
	vu32 TH = 0,TM = 0,TS=0;
	u8 j = 0;
	u8 sd_present=0;
	sMSD_SDCARD sdCard[1];

	sd_present = MSD_Init();
	initialiseSD(sdCard);
	if (sd_present == 0) {
			sd_present = SD_INFORMATION(sdCard);
			sdCard->SD_CON ='1';
	}
	int countP = 0;

	STMPE811_Init();// Init  touch screen
	TM_STMPE811_TouchData structdata[1];

	RCC_Configuration();
	Delayms(5);;
    initialiseLCD();	// Init display
    delay(10);
	initialiseGPS();	// Init GPS
	delay(10);
	initialiseTime();	// Init Clock
	delay(10);
	initialiseCOZIR();	// Init COZir
	delay(10);

    int LCDCountDown = 100;
	GPIO_WriteBit(GPIOD , GPIO_Pin_10 ,Bit_SET); //GPS OFF
```

Figure 60: Device Initialisation

```
    while(1)
    {       if (ScreenON == 1)   // Check the user touch where screen is ON
                  if(STMPE811_ReadTouch(structdata) == STMPE811_State_Pressed)
                  {
                   STMPE811_ReadData();      // read user input and update log file
                  }
```

Figure 61: Touch screen function

```
     while(!GPIO_ReadInputDataBit(GPIOD, GPIO_Pin_12)) ScreenON++;  // Check the
interact button if pressed and wait for release
      if (ScreenON > 30)     // Turn on the screen the button was pressed
      {
            if (DeviceData.LCD_ON = '0')  //  check if screen was not already on
            {
                  DeviceData.LCD_ON = '1';
                  LCDCountDown =100;
                  ScreenON = 1;          // Screen ON Flag
                  GPIO_WriteBit(GPIOD , GPIO_Pin_10 ,Bit_SET); //Turn off GPS
momentarily to manage power since screen is on
                  GPIO_WriteBit(GPIOB , GPIO_Pin_15, Bit_RESET); // Screen ON
                  GPIO_WriteBit(GPIOB , GPIO_Pin_12, Bit_SET);
            }
            else {
                  if(DeviceData.LCD_ON  == '1' && LCDCountDown <= 0)  // If screen
time out, turn off screen
                  {
                        GPIO_WriteBit(GPIOD , GPIO_Pin_10 ,Bit_RESET); //GPS ON
                        GPIO_WriteBit(GPIOB , GPIO_Pin_15, Bit_SET);   // Screen
OFf
                        GPIO_WriteBit(GPIOB , GPIO_Pin_12, Bit_SET);
                        DeviceData.LCD_ON = '0';
                        btnStatus = 0;
                        delay(10);
                        ScreenON = 0;
                        LCDCountDown = 0;
                  }
                  else { if (LCDCountDown > 0)     // count down displa
                              LCDCountDown --;
                  }
            }
            if (DeviceData.LCD_ON  == '1')  // Change screen window if the interact
button pressed when the screen is on
            {     // button pressed
                  if (ScreenON > 10)  {
                        if (btnStatus == 0) //Show SD card
                        {
                              DisplaySDCard(sdCard);
                              btnStatus = 1;
                              DeviceData.LCD_ON = '0';
                        }
                        else if (btnStatus == 1)
                              DeviceData.LCD_ON ='1';
                              initialiseLCD();
                              displayedDate = 0;
                              btnStatus = 0;
                        }
              }
      }
```

Figure 62: Read TFT Screen

```
  if (USART_GetFlagStatus(USART1, USART_FLAG_RXNE) != RESET)  //  Check connection
to the software and respond accordingly
              {
                            char rdata = USART_ReceiveData(USART1) ;

                            if (DateComm == 0)
                            {
                                  if (rdata== DATE_COMMAND)
                                  {
                                        if (DateComm == 1  )
                                        {
                                              if (DateIndex < 10)
                                              {
                                                    dateT[DateIndex] = rdata;
                                                DateIndex ++;
                                              }
                                              else
                                              {
                                                    //ReCalculateDate();
                                                    DateComm = 0;
                                                    DateIndex = 0;
                                              }
                                        }
                                        DateComm = 1;
                                  }
                                  else
                                  {
                                        ReceiveCommands(rdata, sdCard);
                                  }
                            }
                            else
                            {

                            }

              }
```

Figure 63: CBWD Software connection detection

From the diagrams above, the main function continuously monitors the connection from the computer for the commands, if there are no commands, it also check on whether the user has pressed the wake up button, from which it will check on the touch screen interaction in case the screen is ON. The wake up button is used to turn on the screen.

The flow diagram for the above code is shown below:

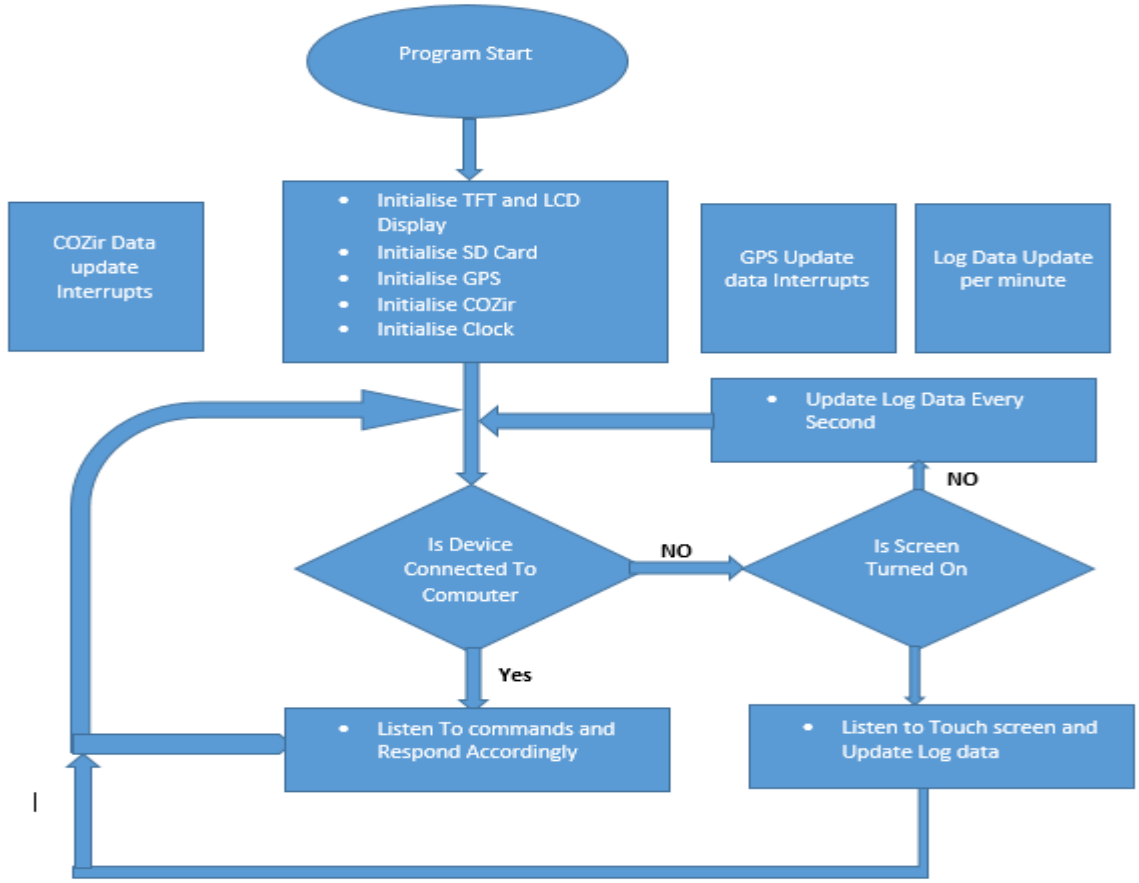


Figure 64: Main Function flow diagram

8.4.2 The GPS

To GPS refreshes its data every second, therefore the GPS module generates the interrupts every second. During this interrupt the GPS class extracts the GPRMC data and keeps it in a variable that will be read by the MCU once it is ready. The GPS TX pin is connected to the RX pin A3 of the MCU. The pseudo code to perform port initialisation and configuration is provided below:

- GPS initialisation and configuration
  - Configure the GPS ON/Off port for power manipulation
  - Configure MCU RX pin to GPS baud rate (9600), and other data variables
  - Enable GPS interrupt

The function to perform the above pseudo code is provided below:

```
void GPIO_Configuration_GPS(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    /*=================================================================
     * GPS
     * =================================================================
    * Configure GPS ON OFF SWITCH */
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;  // PD10
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
     GPIO_Init(GPIOD, &GPIO_InitStructure);
    /* Configure GPS USART Rx as input floating */
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3; // PA3 USART2.RX
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
     GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

Figure 65: GPS MCU ports initialisation

On the function above, the pin PD10 is initialised as an output pin, which is used to turn ON/OFF the GPS module. Also the RX pin PA3 is set as the input pin to receive the data from the GPS module. Once the RX pin has been configured as an input commutation ports, the port baud rate, data length, stop bits and parity and mode of commutation need to be configured.

The following configuration has been performed:

- Baud rate : 9600
- Word Length : 8 bits
- Stop bit : 1
- Parity : 1
- Mode: RX pin.

The function below performs the above configuration:

```
void USART_Configuration_GPS(void)
{
  USART_InitTypeDef USART_InitStructure;
  /* USART resources configuration (Clock, GPIO pins and USART registers) ----*/
  /* USART configured as follow:
        - BaudRate = 9600 baud
        - Word Length = 8 Bits
        - One Stop Bit
        - No parity
        - Hardware flow control disabled (RTS and CTS signals)
        - Receive and transmit enabled */
  USART_InitStructure.USART_BaudRate = 9600;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx;
  USART_Init(USART2, &USART_InitStructure); /* USART configuration */
  /* Here the USART2 receive interrupt is enabled
   * and the interrupt controller is configured
   * to jump to the USART2_IRQHandler() function */
  USART_ITConfig(USART2, USART_IT_RXNE, ENABLE); // enable the USART2 receive
interrupt
  /* Enable the USART2 */
  USART_Cmd(USART2, ENABLE);
}
```

Figure 66: GPS RX port configuration

Once all the configurations have been completed, the interrupt configuration is performed and the GPS interrupt enabled. The code below activates the GPS receive data interrupt:

```
void NVIC_Configuration_GPS(void)
{     NVIC_InitTypeDef NVIC_InitStructure;
            /* Configure the NVIC Preemption Priority Bits */
      NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
             /* Enable the USART2 Interrupt */
      NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
      NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
      NVIC_Init(&NVIC_InitStructure);
      NVIC_Init(&NVIC_InitStructure);
}
```

Figure 67: GPS Receive data interrupt enable function

When the interrupt is experienced, the GPS class updates the GPS data variables with the GPRMC data, therefore the interrupt only extracts the only required data. The pseudo code to extract the GPRMC data only is provided below:

- GPS interrupt generated
  - Check if this is a new line sentence (Every new line starts with $ character)
  - Check if the next letters are GPRMC
    - If not GPRMC ignore all other characters until you get a new line and start the steps from new line
    - If GPRMC
      - Store the sentence and make it available for the MCU to read

The function to perform the above code is provided below:

```
void USART2_IRQHandler(void)
/* Reads GPS Data  GPRMC data */
{
	char c;
	if(USART_GetFlagStatus(USART2, USART_FLAG_RXNE) != RESET)
	{
		if (i < 255)
		{
			c = USART_ReceiveData(USART2);

			if (c == '$')
			{
				if (GPRMC > 0)
					{
						//CopyArrays(GPSFinalData, GPSTempData, i);
						CopyGPSData(GPSTempData,i);
						//GPS_siz = i;
						GPRMC= 0;
					}
				//intitialiseArrays(i,GPSTempData);
				i = 0;
				GPSTempData[i]= '$';
				i++;
			}
			else if (i >= 0)		{
				GPSTempData[i] = c;
				i++;
			}

			if (i == 6)  //Check if GPRMC data
			{
		if(GPSTempData[3] == 'R' && GPSTempData[4] == 'M' &&   GPSTempData[5] == 'C')
				{
					GPRMC = 1;
				}
				else
				{
					i = -1;
					GPRMC = 0;			}
	}
		}
		else		{
			i = -1;
		}
	}
}
```

Figure 68: GPS interrupt handler function

8.4.3 The COZir

The COZir provides the CO_2 level, the relative humidity and the Temperature. This data is being refreshed twice per second, and the GPS values are also updated via interrupts. To accomplish this functionality, the MCU ports to the MCU are configured, and the pseudo code to configure the COZir class is provided below:

- COZir
  - Initialise input RX pin PA11
  - Configure the communication data parameters (Baud rate, word size, stop bit , etc)
  - Enable interrupts

The function to configure the RX pin to the COZir is provided below:

```
void GPIO_Configuration_COZIR(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /*=====================================================================
     * SENSOR
     * =====================================================================
     */

    /* Configure SENSOR USART Rx as input floating */
     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11; // PB11 USART3.RX
     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
     GPIO_Init(GPIOA, &GPIO_InitStructure);

    /*=====================================================================
         * COZIR
     *=====================================================================
    */
     GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

Figure 69: COZir ports initialisation function

From the function above, the COZir is connected to PA11 (MCU RX pin), and this port is configured as the input port. Additionally the communication variables are configured same as the GPS communication variables above and the function to perform this function is provided below:

```
void USART_Configuration_COZIR(void)
{
  USART_InitTypeDef USART_InitStructure;

  /* USART resources configuration (Clock, GPIO pins and USART registers) ----*/
  /* USART configured as follow:
        - BaudRate = 9600 baud
        - Word Length = 8 Bits
        - One Stop Bit
        - No parity
        - Hardware flow control disabled (RTS and CTS signals)
        - Receive and transmit enabled
  */
  USART_InitStructure.USART_BaudRate = 9600;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx;

  /* USART configuration */
  USART_Init(USART3, &USART_InitStructure);


  /* Here the USART3 receive interrupt is enabled
   * and the interrupt controller is configured
   * to jump to the USART2_IRQHandler() function
     */
  USART_ITConfig(USART3, USART_IT_RXNE, ENABLE); // enable the USART3 receive
interrupt

  /* Enable the USART3 */
  USART_Cmd(USART3, ENABLE);
}
```

Figure 70: COZir communication port initialisation

Once the communication variables have been configured, the interrupt is then enabled, and the function to enable the interrupt is shown below:

```
void NVIC_Configuration_COZIR(void)
{
                    NVIC_InitTypeDef NVIC_InitStructure;

               /* Configure the NVIC Preemption Priority Bits */
               NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

               /* Enable the USART3 Interrupt */
               NVIC_InitStructure.NVIC_IRQChannel = USART3_IRQn;
               NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;
               NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
               NVIC_Init(&NVIC_InitStructure);
}
```

Figure 71: COZir interrupt enable function

When the interrupt is experienced, the required sentence is being extracted, the pseudo code is provided below to give an overview of how the COZir data is extracted:

- COZir data
  - Read the sentence and check for the new line character
  - If new line make the read data available for the MCU and read new data in

The function shown below reads the COZir sentence to extract the temperature, the CO_2 level and the relative humidity:

```
void USART3_IRQHandler(void)
{
	while(USART_GetFlagStatus(USART3, USART_FLAG_RXNE) == RESET);

	char c = USART_ReceiveData(USART3) ;
	if (c =='\r' || c =='\n')
	{
		if (completeCOZIRData(COZIRTempData,k) == 1)
		{
			//CopyArrays(COZIRFinalData ,COZIRTempData,k);
			//COZIR_siz = k;
			CopyCOZIRData(COZIRTempData,k);
		}
	//	intitialiseArrays(k,COZIRTempData);
		k = 0;
	}
	else
	{
		if (k < 255)
		{
			COZIRTempData[k] = c;
			k++;
		}
	}
}
```

Figure 72: COZir data extraction function

8.4.4 The SD Card

The SD card provides the storage for the device. It is connected to the MCU in a SPI mode (SP1). To successfully communicate with the SD card, the following pseudo code should be implemented:

- Initialise SD card communication ports and mode
- Check if the SD card is available
- Get the SD card size and other properties
- Write and read from the SD card.

The SD card is initialised in an SPI mode, the code below provides a way to initialise SPI1 for the micro SD card:

```
void SPI_Config(void)
{
  uint32_t delay;
  GPIO_InitTypeDef  GPIO_InitStructure;
  SPI_InitTypeDef   SPI_InitStructure;

  /* GPIOC Periph clock enable */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_SPI1 |
RCC_APB2Periph_AFIO, ENABLE);
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);

  /* Configure SPI1 pins: SCK, MISO and MOSI */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* Configure PA4 pin: CS pin */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStructure);

  /* SPI1 Config */
  SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;
  SPI_InitStructure.SPI_Mode = SPI_Mode_Master;
  SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;
  SPI_InitStructure.SPI_CPOL = SPI_CPOL_High;
  SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;
  SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;
  SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;
  SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;
  SPI_InitStructure.SPI_CRCPolynomial = 7;
  SPI_Init(MSD_SPI, &SPI_InitStructure);

  /* SPI enable */
  SPI_Cmd(MSD_SPI, ENABLE);

  //Delay
  for(delay = 0; delay < 0xfffff; delay++);
}
```

Figure 73: SD Card initialisation (SPI 1)

From the code snip above, the SD card is connected to MCU via pins PA4, PA5, PA6 and PA7 which connects to the SD card pins CS, SCK, MISO and MOSI respectively.

To identify is the SD card is connected, a bite is send to the SD card and if no response in a specific time, the card is not inserted, and the following code gets the response from the SD card.

```
u8 MSD_GetResponse(u8 Response)
{
  u32 Count = 0xFFF;
  /* Check if response is got or a timeout is happen */
  while ((MSD_ReadByte() != Response) && Count)
  {
    Count--;
  }
  if (Count == 0) {
     return MSD_RESPONSE_FAILURE; /* After time out */
  }
  else {
        return MSD_RESPONSE_NO_ERROR; /* Right response got */
  }
}
```

Figure 74: Get Micro SD card response

To write data to the SD card, first a Write command is send, and if the SD card responds with no error, the block of data characters are being written to the SD card, once all data has been send through, the data response is being requested from the SD card, to acknowledge if data has been successfully accepted.

The code snipe below achieves SD card writing:

```
u8 MSD_WriteBlock(u8* pBuffer, u32 WriteAddr, u16 NumByteToWrite)
{
  u32 i = 0;
  u8 rvalue = MSD_RESPONSE_FAILURE;
  MSD_CS_LOW();/* MSD chip select low */
  MSD_SendCmd(MSD_WRITE_BLOCK, WriteAddr, 0xFF); /* Send CMD24 (MSD_WRITE_BLOCK) to
write multiple block */

/* Check if the MSD acknowledged the write block command:
       R1 response (0x00: no errors) */
  if (!MSD_GetResponse(MSD_RESPONSE_NO_ERROR))
  {
     MSD_WriteByte(DUMMY); /* Send a dummy byte */
     MSD_WriteByte(0xFE); /* Send the data token to signify the start of the data */

    /* Write the block data to MSD : write count data by block */
    for (i = 0; i < NumByteToWrite; i++)
    {
      MSD_WriteByte(*pBuffer); /* Send the pointed byte */
      /* Point to the next location where the byte read will be saved */
      pBuffer++;
    }
    MSD_ReadByte();/* Put CRC bytes (not really needed by us, but required by MSD)*/
    MSD_ReadByte();
    /* Read data response*/
    if (!MSD_GetResponse(MSD_RESPONSE_NO_ERROR))
    {
      rvalue = MSD_RESPONSE_NO_ERROR;
    }
  }
  MSD_CS_HIGH();/* MSD chip select high */
  MSD_WriteByte(DUMMY); /* Send dummy byte: 8 Clock pulses of delay */
  return rvalue; /* Returns the reponse */
}
```

Figure 75: SD card write function

To read data from the SD card, a command to signify data reading is send to the SD card, then a response from the SD card is awaited, once received, data read from a specific address is initiated.

The code below shows how data read is performed on the SD card:

```
u8 MSD_ReadBlock(u8* pBuffer, u32 ReadAddr, u16 NumByteToRead)
{
  u32 i = 0;
  u8 rvalue = MSD_RESPONSE_FAILURE;
  MSD_CS_LOW();/* MSD chip select low */
  /* Send CMD17 (MSD_READ_SINGLE_BLOCK) to read one block */
  MSD_SendCmd(MSD_READ_SINGLE_BLOCK, ReadAddr, 0xFF);

  /* Check if the MSD acknowledged the read block command: R1 response (0x00: no
errors) */
  if (!MSD_GetResponse(MSD_RESPONSE_NO_ERROR))
  {
    /* Now look for the data token to signify the start of the data */
    if (!MSD_GetResponse(MSD_START_DATA_SINGLE_BLOCK_READ))
   {
      /* Read the MSD block data : read NumByteToRead data */
      for (i = 0; i < NumByteToRead; i++)
      {
        /* Save the received data */
       *pBuffer = MSD_ReadByte();
        /* Point to the next location where the byte read will be saved */
        pBuffer++;
      }
      /* Get CRC bytes (not really needed by us, but required by MSD) */
      MSD_ReadByte();
      MSD_ReadByte();
      /* Set response value to success */
      rvalue = MSD_RESPONSE_NO_ERROR;
    }
  }
  /* MSD chip select high */
  MSD_CS_HIGH();
  /* Send dummy byte: 8 Clock pulses of delay */
  MSD_WriteByte(DUMMY);
  /* Returns the reponse */
  return rvalue;
}
```

Figure 76: Reading data from the SD card

8.4.5 The LCD screen

The LCD screen is responsible for displaying data, it is connected to the MCU to work in parallel data transmission on pin Es. The physical connections of the pins to the MCU have been explained under the hardware design chapter, and can also be viewed from the device circuit diagram on the appendix.

The mode of communication is selected via pins PD0, PD1, PD3 and PD4 (IM0, IM1, IM2, IM3), which for this device is '0001' respectively to allow for 16 bits parallel data transmission. Port E pins are all initialised as output pins. The complete code on how LCD prints characters is provided in the appendix G.

The LCD initialisation function is provided below:

```
static __inline  void LCD_init_Pins()
{
       // Hardware interface
       // nCS          PC6
       // RS           PD13
       // nWR          PD14
       // nRD          PD15
       // DATA[15:0]   PE[15:0]
       // IM0          PD0
       // IM1          PD1
       // IM2          PD3
       // IM3          PD4
        GPIO_InitTypeDef GPIO_InitStructure;
    /* Configure GPS ON OFF SWITCH */
       GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15;  // PB15
       GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
       GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
       GPIO_Init(GPIOB, &GPIO_InitStructure);

       GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;  // PB12
       GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
       GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
       GPIO_Init(GPIOB, &GPIO_InitStructure);

        /* Configure LCD data PINS*/
        GPIO_InitStructure.GPIO_Pin =GPIO_Pin_0| GPIO_Pin_1 | GPIO_Pin_3| GPIO_Pin_4
|GPIO_Pin_5;
     GPIO_Init(GPIOD, &GPIO_InitStructure);

    //Set the memory mode M3 M2 M1 M0 0001
       GPIO_WriteBit(GPIOD , GPIO_Pin_0 ,Bit_SET);
       GPIO_WriteBit(GPIOD , GPIO_Pin_1 ,Bit_RESET);
       GPIO_WriteBit(GPIOD , GPIO_Pin_3 ,Bit_RESET);
       GPIO_WriteBit(GPIOD , GPIO_Pin_4 ,Bit_RESET);

    // RESET PIN SET
       GPIO_WriteBit(GPIOD , GPIO_Pin_5 ,Bit_SET);

       RCC->APB2ENR |= 0x0000007D;
       /* PE output set to high. */
       GPIOE->CRL = 0x33333333;       // output push-pull
       GPIOE->CRH = 0x33333333;         // output push-pull
       GPIOC->CRL &= 0xf0ffffff;      // PC6 output push-pull 50MHz
       GPIOC->CRL |= 0x03000000;
       GPIOD->CRH &= 0x000fffff;      //PD13,14,15 output push-pull 50MHz
       GPIOD->CRH |= 0x33300000;
       GPIOC->BSRR = 0x00000040;         //cs = 1;
       GPIOD->BSRR = 0x0000E000;         //rs,wr,rd = 1;
       delay(10);
}
```

Figure 77: LCD parallel communication initialisation function

9. Results

In this chapter, the experimental results are being gathered and the methods used to gather data are explained in detail. Initially the software testing results are presented, then the hardware results. The software was tested on windows 7 and windows 8 operating system.

9.1 System Software Testing

This section provides Carbon dioxide body worn software test results and the method used to collect data.

9.1.1 Device Detection and Auto connection

Different number of devices were connected to the laptop which has 3 USB ports, and the software started to see whether it can be able to detect and connected to the CBWD successfully. The following results were obtained:

Initially the CBWD Software was started without any device connected and the following error was raised:

The error message raised reads as “There is no device connected. Please connect the device and click on RECONNECT”. The message explains that the user can connect a device and click on reconnect and the device will be automatically connected.

Connecting only the CO2 device and then starting the software generated the following results:

Figure 78: CBWD Software error message if no device is found

From the image above, it can be confirmed that the software automatically detected the device communication port and connected to it, and assures the user that the device has been connected to which port, in this case COM3.

The table below summarises the results obtained by connecting different number of the devices to the computer:

Number of devices connected	Devices Connected	COM Port	Connect immediately
0	No device Connected		Error message
1	Carbon dioxide body worn device only	COM 3	Yes
2	Carbon dioxide body worn device only	COM 3	Yes
	Version 1 COZir device	COM 4	Not connected
2	Version 1 COZir device	COM 3	Yes
	Carbon dioxide body worn device only	COM 4	Not connected
2	Version 1 COZir device	COM 3	Yes
	Stm32F4 Board	COM 4	Not Connected
2	Stm32F4 Board	COM 3	Yes
	Version 1 COZir device	COM 4	Not Connected
3	Carbon dioxide body worn device only	COM 3	Yes
	Version 1 COZir device	COM 4	Not connected
	Stm32F4 Board	COM 5	Not Connected

Table 15: Number of devices connected to PC and their connection results

From the table above, when the new device is connected together with other devices, the software is able to identify and connect to the new version CBWD. This is due to the fact that once the software has been connected, it requests device name, if not received it moves to the next available communication port. If all the ports are scanned with no connection success, it tries to connect to the first available port, if the port is used, then to the next port. This can be notified from the table above, that if the old version of the device is connected on the first port COM3 and STM32F4 board to COM4, the device connects to COM3, even if the devices are swapped, the software still connects to COM3. It can also be sported from the interface that there is the disconnect button, from which the user can click and select the correct COM port. If the device fails to connect to any device, it provides the list of available ports on the drop box from which the user can select the correct com port to connect to.

9.1.2 Live Communication

In this experiment, CO_2 level, humidity and temperature were measured using the device and the live results displayed on the CBWD software, then the obtained results downloaded. The live result data download sample can be found in Appendix E.

The diagram below shows the CO_2 level, temperature and humidity graphs as displayed by the device software:

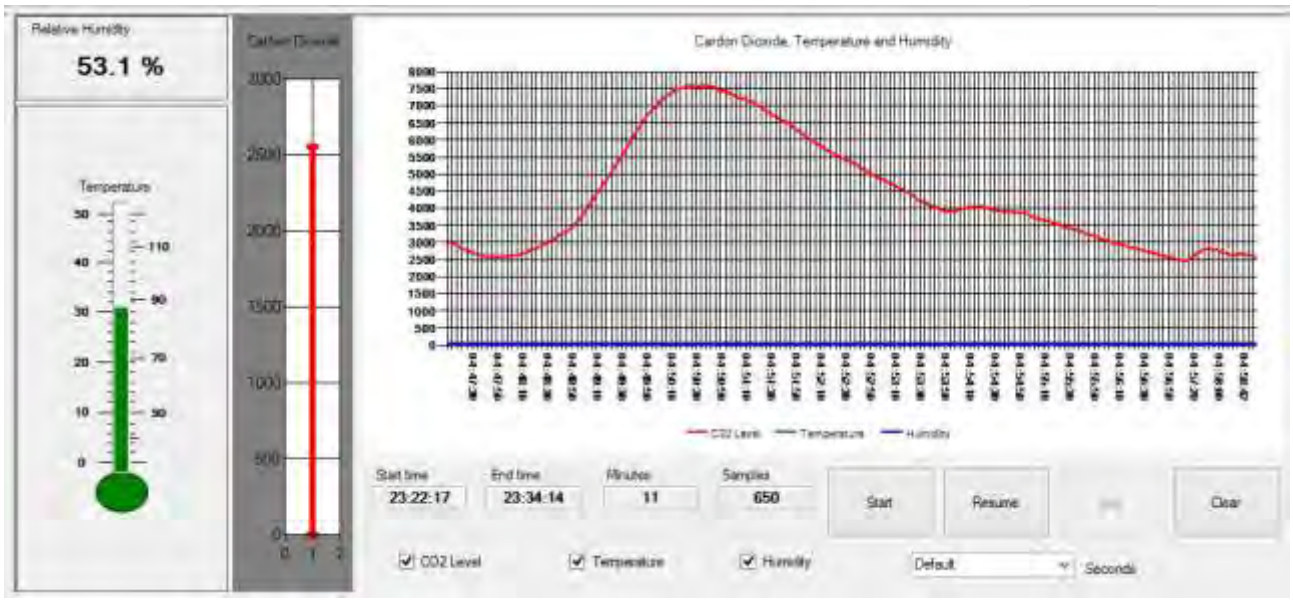


Figure 79: CO_2 level, temperature and humidity plotted by CBWD software

From the graph above, the CO_2 level has been measured with the device reference level not set, thus before zeroing the monitor. The red line shows the CO_2 level behaviour, the graph rises up while the user was breathing onto the device. 650 samples were taken. Towards the bottom of the software, the user is able to tick on which graphs he wants to see.

It can be recognised that the temperature and humidity graph are approximately constant, the relative humidity setting at around 53.1% and the temperature at around 31^0C.

The diagram below shows the humidity and temperature graph:

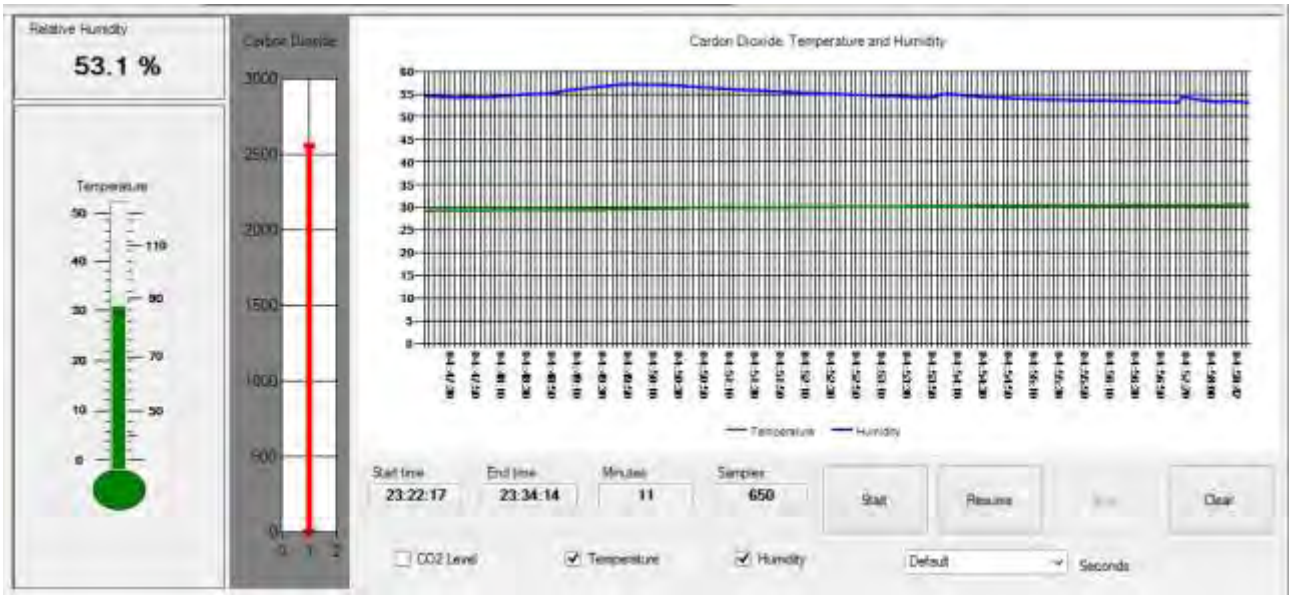


Figure 80: Humidity and Temperature VS time graph plotted by CBWD Software

9.1.3 Log Data Erasing

In this experiment, the device was connected and the delete button clicked. The following notification message is shown:

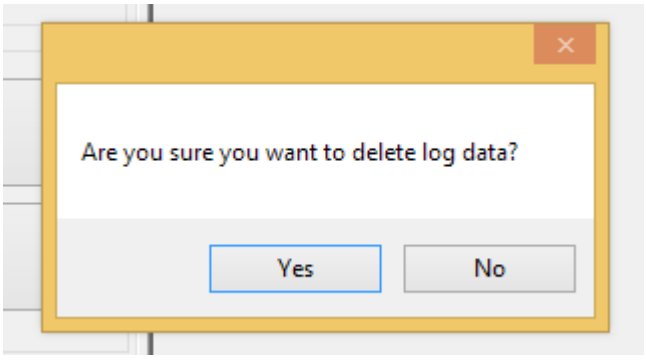


Figure 81: CBWD Software user confirm message

Once data deletion has been confirmed. The user gets the assurance message shown below:

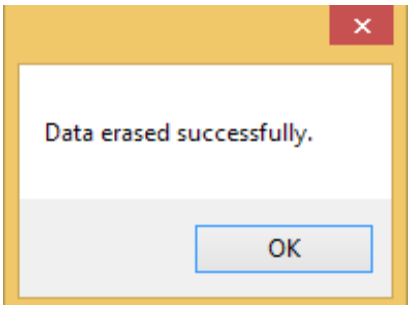


Figure 82: CBDW Software data deletion assurance message

Data downloading was performed right after this process, and the obtained results are explained below.

9.1.4 Log Data Downloading and Error Analysis

For data downloading to occur, the user click on the retrieve log button, and the user validation message shown from which the user confirms his request.
The diagram below shows the confirmation message:

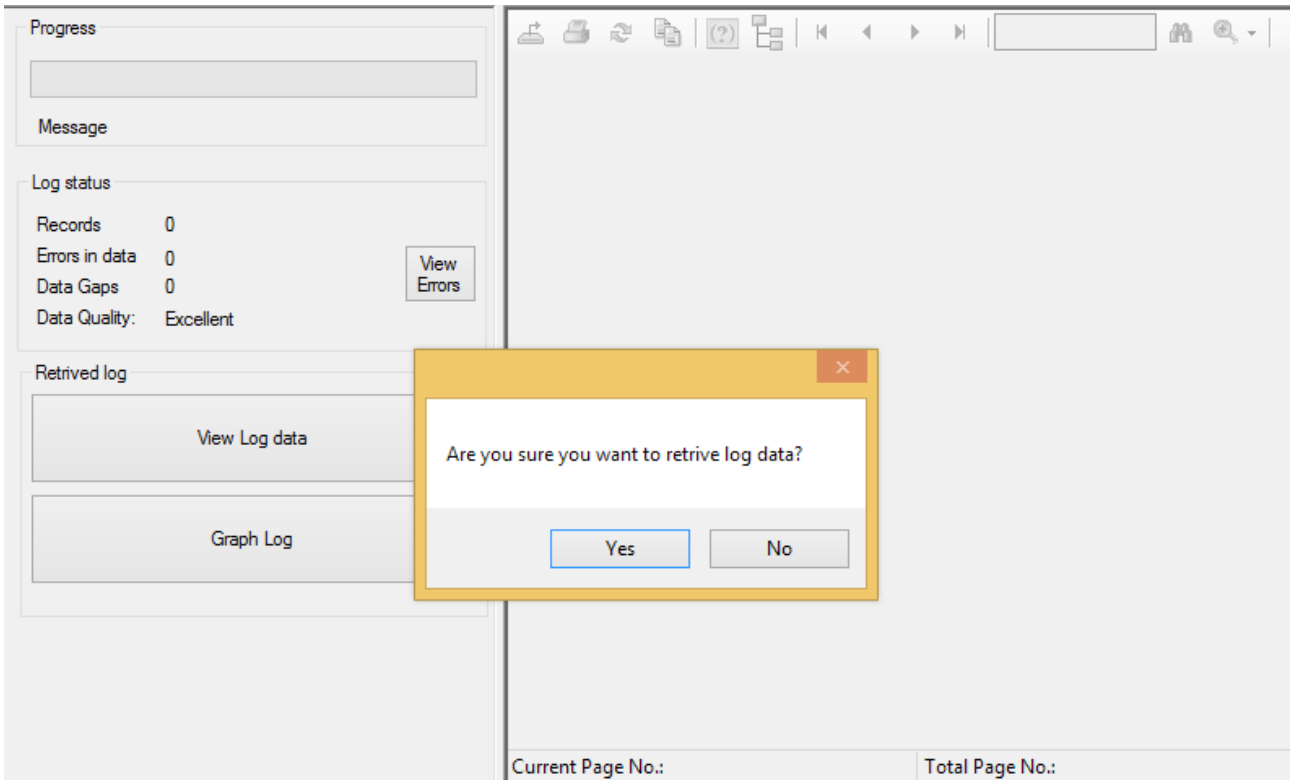


Figure 83: CBDW Software Data downloading

Once the user has confirmed data retrieval, the data is being downloaded, and any errors on the data, such as data gaps are analysed and the software provides the results. The diagram below shows the results obtained after downloading data.

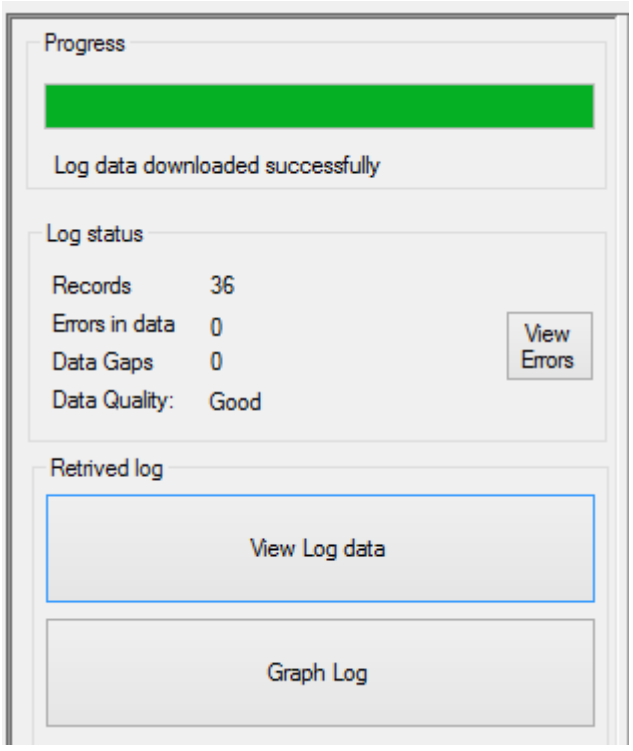


Figure 84: CBWD Software data download completion message

From the diagram above, 36 records have been downloaded, and there are no errors in the data download. The errors includes missing CO_2 values since the device is supposed to log data at a second interval, hence data gaps refers to when the device skipped some time without logging. The user can click on view error to get the error details, such as between which time did the data gap occur. The user can also click on the view data to actually see the data structure before exporting it. The export option can be found on the top left menu as shown in the diagram below:

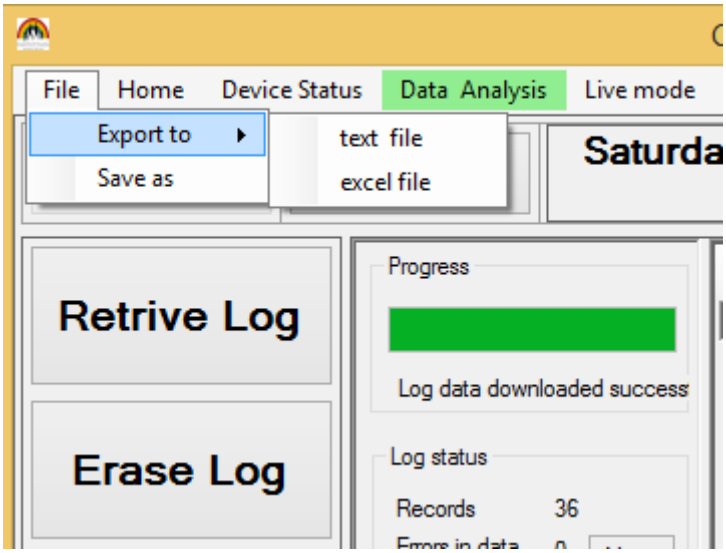


Figure 85: CBDW Software export menu

From the diagram the quick files to export to includes, the text or the excel file, other file formats shown below can be found under save as option:

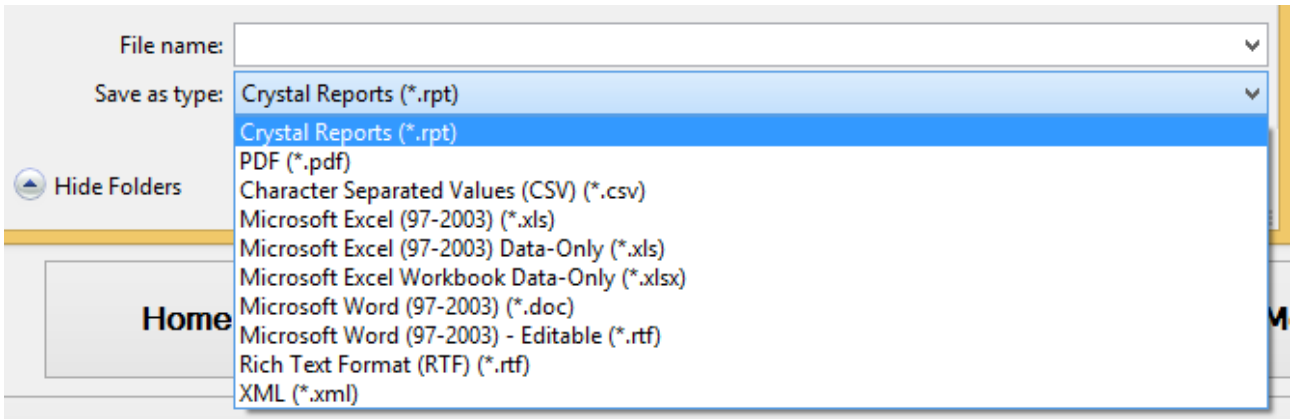


Figure 86: CBDW Software export file options

9.1.5 Time Synchronizing

To perform this test, a computer time and data was switched by changing time zone and the device time and date synchronized, then to confirm the device time and date, the live results were obtained as explain on the above live testing results. When the user clicks on the synchronise data and time button, the confirmation message is shown:

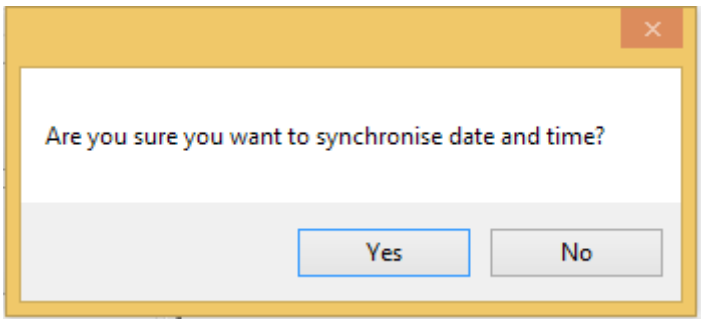


Figure 87: CBWD Software date and time synchronization user confirm message

From which he confirms on the synchronising date and time. The diagram below shows the results from the device and the computer time and date:

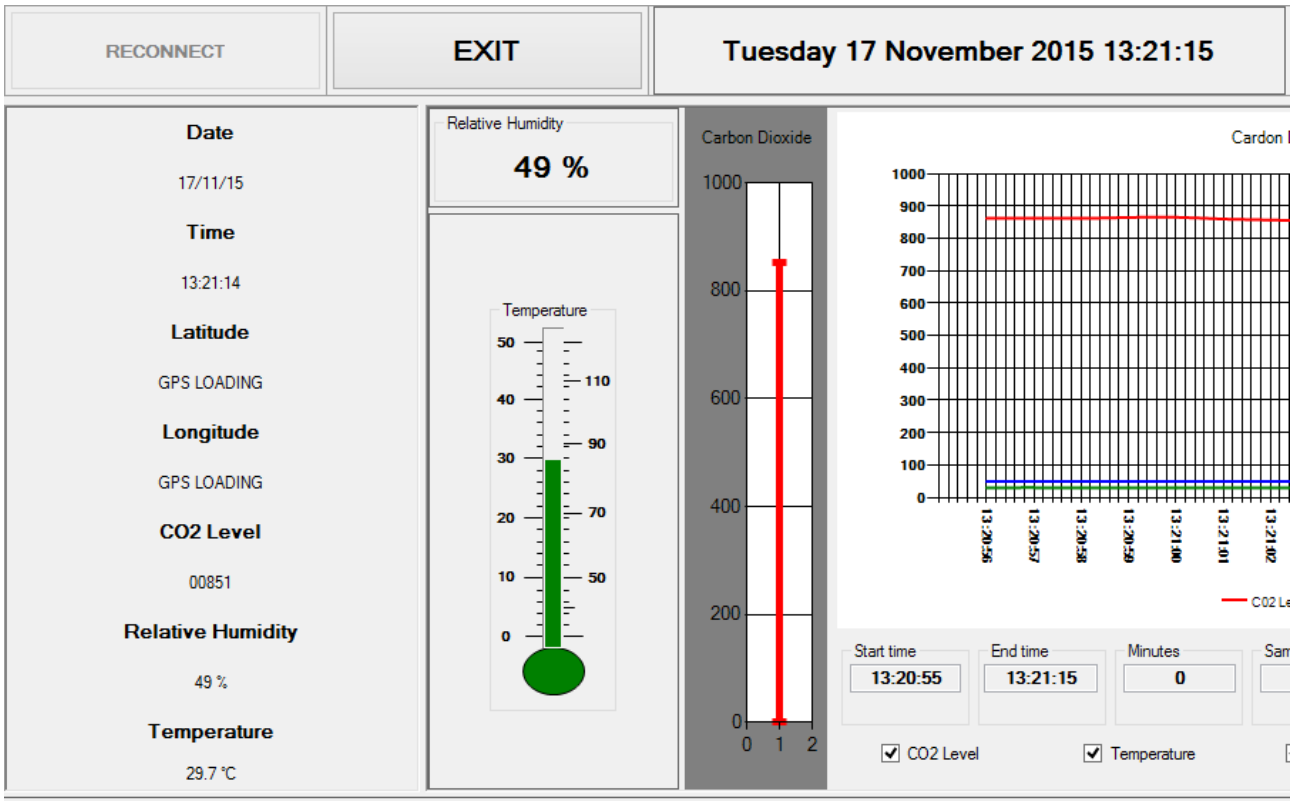


Figure 88: CBDW Software interface showing time and date

From the graph above, the computer date and time reads as:

Figure 89: Date and time from computer

And the time and date obtained from device live data reads as:

Figure 90: Date and time from the device

The two values are out of sync by 1 second as shown on the diagrams above.

9.1.6 *Zero COZir*

In this experiment, the COZir reference values was set to the room CO_2 rpm and the live results obtained. To Zero the COZir, one click on "Set COZir zero button and confirms on zero the monitor by clicking yes on the message shown below:

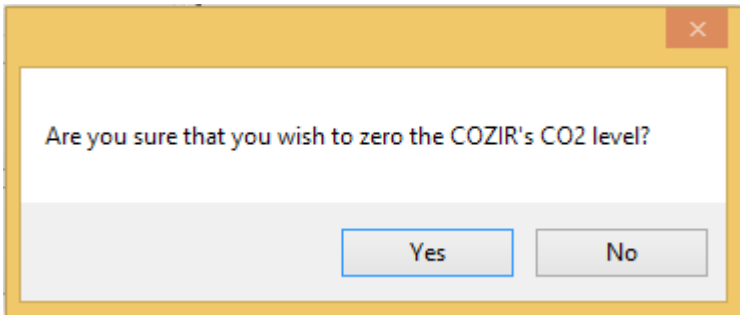


Figure 91: CBWD Software COZir zeroing user confirm message

After confirming on zeroing the device, the following live results are obtained:

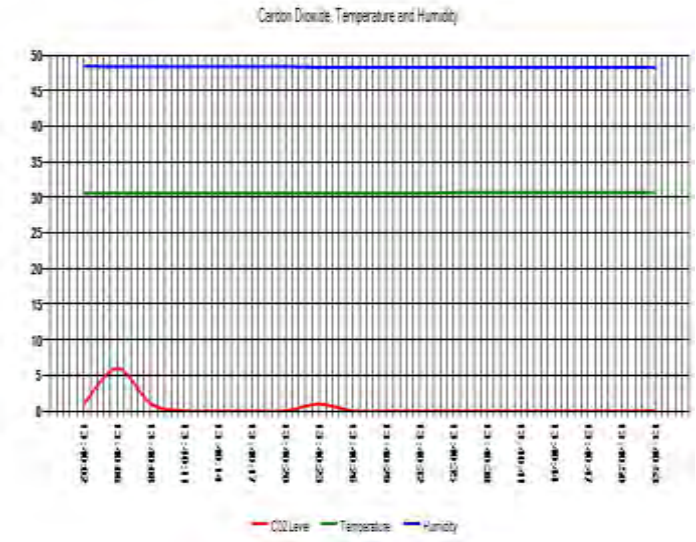


Figure 92: CO_2 level, temperature and humidity after zeroing the monitor

From the graph above, CO_2 level (in red) show the results approximately zero, showing that the device reference level has been set close to the room COZir level successfully.

9.2 Hardware Testing at Component Level

9.2.1 *GPS Accuracy at fixed point*

The device provides the location coordinates. In this experiment, the device was held at a fixed position outdoors during the day, and its GPS stopped from turning off, and the NMEA data collected using visualGPS software programme. Then test results on GPS accuracy at fixed point were performed by surveying it location coordinates using visualGPS software programme.

The diagram below shows the position plot for the GPS:

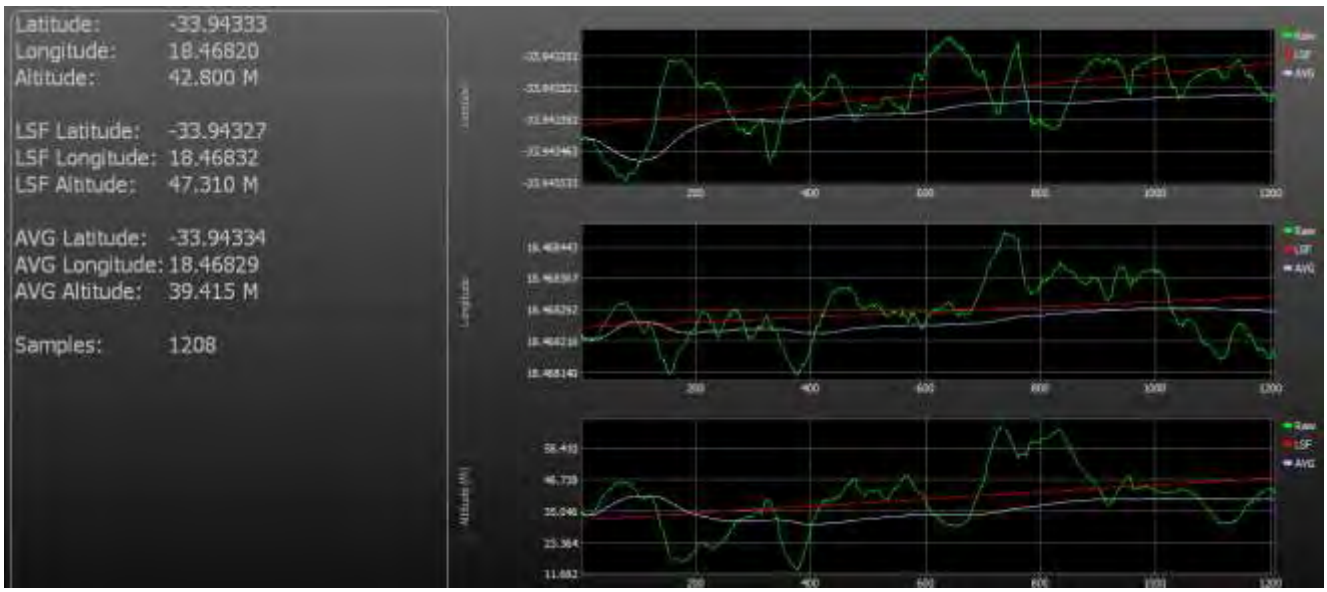


Figure 93: CBWD location coordinates analysis at fixed point

For the graph above, the green graph shows the raw GPS coordinates which varies significantly, but the average plot is more constant.

The diagram below shows scatter plot for the GPS:

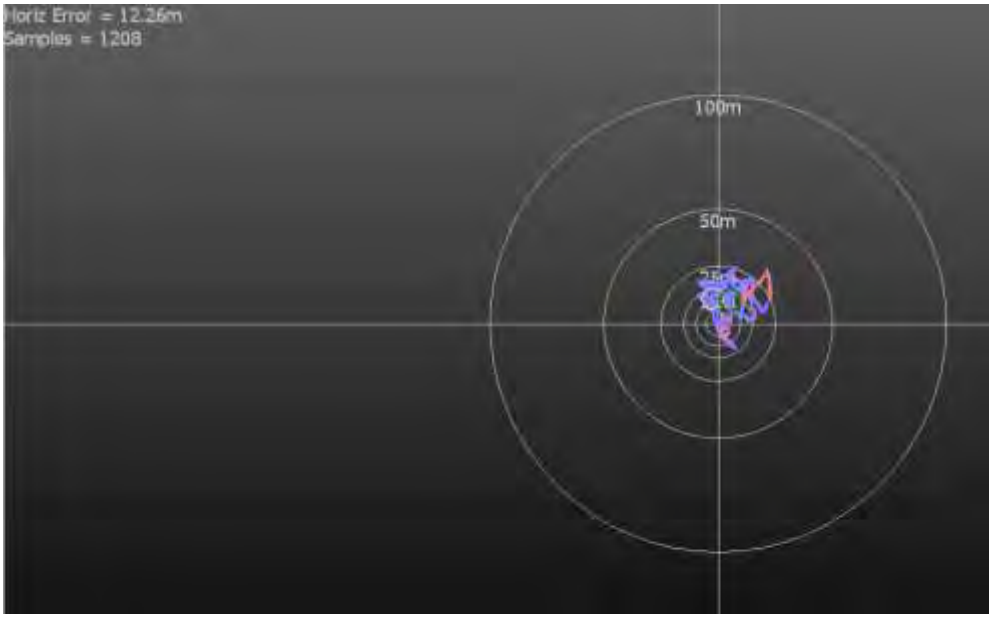


Figure 94: GPS Accuracy at fixed point Scatter plot

The diagram below shows the number of satellites visible and the ones tracked:

Figure 95: Number of GPS satellites used in coordinates calculations

From the above diagram, there are 11 satellites in view, and 9 of this satellites have been used to calculate a fix. The horizontal error calculated on the diagrams above is about 12.26m which can be seen from the GPS Accuracy at fixed point Scatter plot graph above.

The **HDOP** = 1.5 is a measure of how the satellites that have been connected to are arranged. The satellites that are closely arranged results in a high value of **HDOP**. If the satellites are close together, triangulation is not defined, and it results in lower position accuracy.

9.2.2 Power Consumption

The device power consumption was analysed by measuring the amount of current it consumes per voltage. The results have been with the GPS and LCD screen off, then with GPS on and LCD screen of, finally with all the components turned on. Also the device was left to operate on its own for a day and the voltage measured after a day and it was found to be 3.5V.

The table below shows the amount of current consumed by the device components excluding the GPS and LEC screen:

GPS OFF and LCD OFF		
Voltage	Current	Device Status
7.4	80	Operating Normal
7	79.8	Operating Normal
6.5	80.2	Operating Normal
6	75	Operating Normal
5.5	60	Operating Normal
5	40	Operates Normally
4.5	45	Misses some log data
4	35	Misses some log data
3.5	10	not working
3	0	not working
2	0	not working
1	0	Not Working

Table 16: Device Power Consumption with GPS and LCD off

From the diagram above, the device operates normally when the voltage is above 4.5V, thus the battery voltage should be kept above this voltage.

The table below shows the results obtained with the screen and the GPS ON:

GPS ON and LCD OFF		
Voltage	Current	Device Status
7.4	130	Operating Normal
7	132	Operating Normal
6.5	120	Operating Normal
6	113	Operating Normal
5.5	85	Operating Normal
5	83	Operates Normally
4.5	12	Not working
4	0	Not working
3.5	0	not working
3	0	not working
2	0	not working
1	0	Not Working

Table 17: Device Power with GPS ON

From the table above, the device operates normally with the GPS permanently ON with the voltage above 4.5V.

The table below shows the results obtained with the GPS ON and LCD screen ON

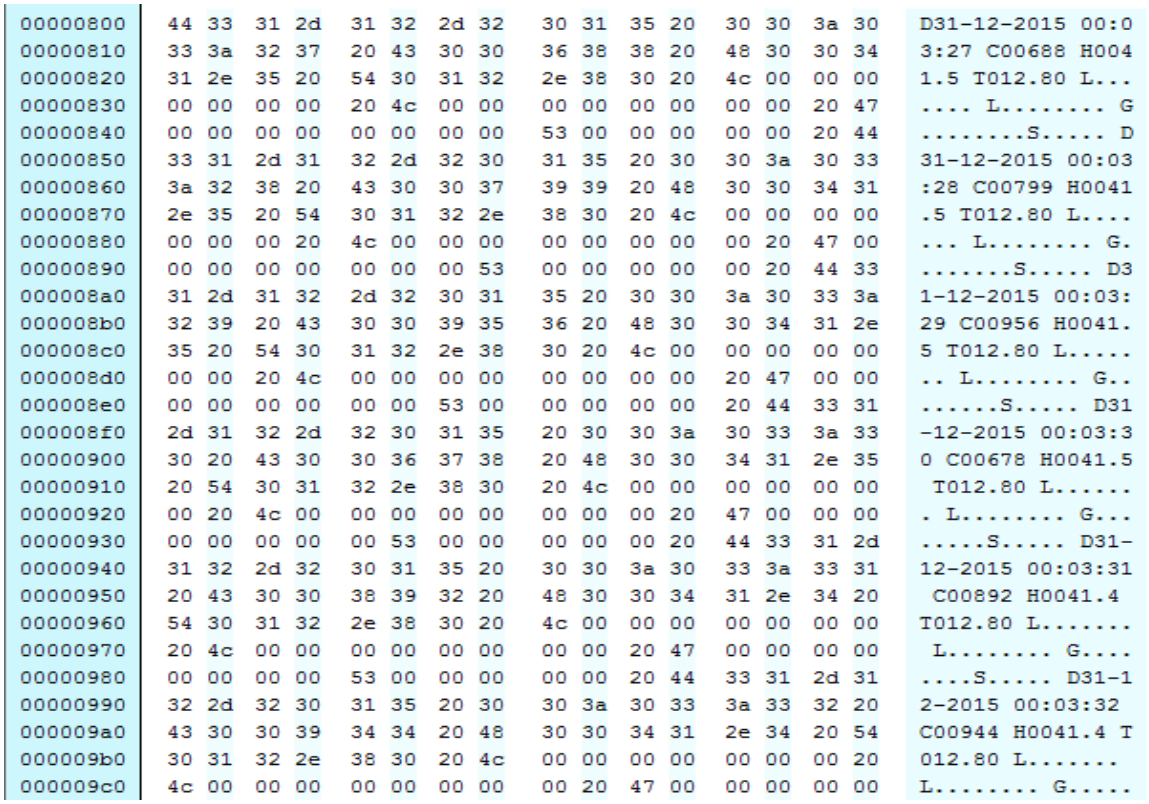
GPS ON and LCD OFF		
Voltage	Current	Device Status
7.4	145	Operating Normal
7	143	Operating Normal
6.5	135	Operating Normal
6	120	Operating Normal
5.5	100	Operating Normal
5	43	Operates Normally
4.5	20	Not working
4	0	Not working
3.5	0	not working
3	0	not working
2	0	not working
1	0	Not Working

Table 18: Device Power consumption when operating in full capacity

From the table below, it can be confirmed that if the GPS and LCD screen are kept on at the same time, the device can operate at a voltage greater or equal 5V, therefore the best mechanism to keep the device in operation is to coordinate the LCD and GPS , thus when to turn on and off.

9.2.3 Device Log file in memory

The log file was read using the hex editor file reader, so as to confirm the data log onto the SD card. The diagram below shows the log file data in binary form and in human readable form on the SD card memory:



```
00000800  44 33 31 2d 31 32 2d 32 30 31 35 20 30 30 3a 30  D31-12-2015 00:0
00000810  33 3a 32 37 20 43 30 30 36 38 38 20 48 30 30 34  3:27 C00688 H004
00000820  31 2e 35 20 54 30 31 32 2e 38 30 20 4c 00 00 00  1.5 T012.80 L...
00000830  00 00 00 00 20 4c 00 00 00 00 00 00 00 00 20 47  .... L........ G
00000840  00 00 00 00 00 00 00 00 53 00 00 00 00 00 20 44  ........S..... D
00000850  33 31 2d 31 32 2d 32 30 31 35 20 30 30 3a 30 33  31-12-2015 00:03
00000860  3a 32 38 20 43 30 30 37 39 39 20 48 30 30 34 31  :28 C00799 H0041
00000870  2e 35 20 54 30 31 32 2e 38 30 20 4c 00 00 00 00  .5 T012.80 L....
00000880  00 00 00 20 4c 00 00 00 00 00 00 00 00 20 47 00  ... L........ G.
00000890  00 00 00 00 00 00 00 53 00 00 00 00 00 20 44 33  .......S..... D3
000008a0  31 2d 31 32 2d 32 30 31 35 20 30 30 3a 30 33 3a  1-12-2015 00:03:
000008b0  32 39 20 43 30 30 39 35 36 20 48 30 30 34 31 2e  29 C00956 H0041.
000008c0  35 20 54 30 31 32 2e 38 30 20 4c 00 00 00 00 00  5 T012.80 L.....
000008d0  00 00 20 4c 00 00 00 00 00 00 00 00 20 47 00 00  .. L........ G..
000008e0  00 00 00 00 00 00 53 00 00 00 00 00 20 44 33 31  ......S..... D31
000008f0  2d 31 32 2d 32 30 31 35 20 30 30 3a 30 33 3a 33  -12-2015 00:03:3
00000900  30 20 43 30 30 36 37 38 20 48 30 30 34 31 2e 35  0 C00678 H0041.5
00000910  20 54 30 31 32 2e 38 30 20 4c 00 00 00 00 00 00   T012.80 L......
00000920  00 20 4c 00 00 00 00 00 00 00 00 20 47 00 00 00  . L........ G...
00000930  00 00 00 00 00 53 00 00 00 00 00 20 44 33 31 2d  .....S..... D31-
00000940  31 32 2d 32 30 31 35 20 30 30 3a 30 33 3a 33 31  12-2015 00:03:31
00000950  20 43 30 30 38 39 32 20 48 30 30 34 31 2e 34 20   C00892 H0041.4
00000960  54 30 31 32 2e 38 30 20 4c 00 00 00 00 00 00 00  T012.80 L.......
00000970  20 4c 00 00 00 00 00 00 00 00 20 47 00 00 00 00   L........ G....
00000980  00 00 00 00 53 00 00 00 00 00 20 44 33 31 2d 31  ....S..... D31-1
00000990  32 2d 32 30 31 35 20 30 30 3a 30 33 3a 33 32 20  2-2015 00:03:32
000009a0  43 30 30 39 34 34 20 48 30 30 34 31 2e 34 20 54  C00944 H0041.4 T
000009b0  30 31 32 2e 38 30 20 4c 00 00 00 00 00 00 00 20  012.80 L.......
000009c0  4c 00 00 00 00 00 00 00 00 20 47 00 00 00 00 00  L........ G.....
```

Figure 96: Device log Data in memory

From the above diagram, the device log data has been formats to show the CO_2 level, Humidity, Temperature, GPS latitude, longitude, GPS time, synchronised computer date and time, the battery level and location. This data is being transferred to the CBWD software while downloading.

9.2.4 Device Charging

In this experiment, 15V power supply was feed through the laptop charger port, and current consumed by the device measured, also 5V power supply was feed to mini B port and the current consumed while charging the batteries measured. Additionally voltage was measured at the charging terminals before the battery protector. All this was performed with the LCD screen and GPS off as they will mostly be switched off while charging. The device uses two 3.6V 800mA lithium ion batteries. The table below shows the amount of current obtained while charging through two ports:

Charger Type	Supply Voltage	Voltage at battery Terminal	Current
• Laptop charger	15V	2.68V	140mA
• Mini B charger	5V	2.68V	100mA

From the table above, the charger module supplies constant voltage to the batteries. Current varies slightly.

10. Discussion

In this chapter, the results gathered in the previous chapter are discussed.

10.1 Device Software

10.1.1 Device Detection and Auto connection

The obtained results confirms that the system software can be able to gather a list of all the communication devices on a computer, from which it can be able to detect the Carbon dioxide body worn device. In case of auto connection failure, it minimises the user intense in trouble shooting by providing a list of all available communication ports from which a user can linearly search for the right port and click a connect button. The software is also able to pick the new version of the device regardless of the number of other devices connected along with the device.

10.1.2 Live Communication

The device software is able to automatically get the CO_2 level, the relative humidity and temperature, the device time and location coordinates in real time. It also creates the graphs for this live data which can be analysed, making this device suitable for performing experiments where live data is required. The software also provides the ability to download the gathered data, and the user is able to see the number of samples gathered in how much time.

10.1.3 Log Data Erasing

The gathered results confirms that the device can be able to erase the log data on the device. It also provides warning and assurance messages to the user to confirm success in log data deletion.

10.1.4 Log Data Downloading and Error Analysis

The system software is able to download data from the software and provides the information about the number of samples downloaded. It also helps the user by identifying minimum errors on the downloaded data, which includes data gaps and CO_2 values omission. The software also offers different file systems from which a user can export the downloaded data to. Additionally clear warning and confirmation messages are presented to the user to confirm and assure him of success in data downloading.

10.1.5 Time Synchronizing

The obtained results confirms that the device date and time can successfully be set from the software, thus synchronising the computer time with the device date and time. The confirmation message is offered to the user, and the live mode offers the user the way to actually confirm that the device time has been set accurately. The device time was found to be out of synchronisation by one second, which is not real an enormous deviation, therefore the device can successfully be synchronised with the device. This error is due to time taken for the device to successfully get the information from the software.

10.1.6 Zero COZir

The gathered results show that the device CO_2 value can be successfully set to zero reference. The software offers the confirmation message and the live mode can additionally be used to confirm on the success in zeroing the CO_2 monitor.

10.1.7 GPS Accuracy at fixed point

The GPS module at fixed point generated HDOP of 1.5 which is acceptable. The device is not a tracking device hence the precision of the device can be tolerated. This value was obtained while outdoor and since there is a mechanism to turn off the device while it is at the same location for a long time, there was no need to do indoor testing as the device will spent most of the time with GPS OFF indoors.

10.1.8 Power Consumption

The device is supplied from two AA cells each 3.6V 800mA, and from the experiment it can be verified that a high capacity power cells are required, although the batteries are able to meet the required specification of one day.

The power saving mechanism, which involves turning on and off of the GPS and the LCD screen is very important to this device as the mechanism failure will affect the power consumption of the device, thus it has been found that the device works better and longer when this device components are turned off, therefore this mechanism should always work at its best.

10.1.9 Device Log file in memory

The device results confirm that data is successfully being logged into the SD card. Raw data on this card has been formatted to be readable by human eye to allow direct copying of data.

10.1.10 Device Charging

The device uses two 3.6V 800mA lithium ion batteries charge, therefore offering 2 X 800mAh = 1600mAH. The chargers each provide 120mAH and 100mAH at a constant voltage of 2.68V. It can be argued that laptop charger will charge much faster that the USB port (mini B) charger.

11. Conclusions

Based on the following results above, the following conclusions were made:

- The device software can auto connect to the CDBW device
- The software can successfully download log data, erase log data, synchronise date time with PC date time and zero the COZir
- Device can graphically represent device real time data
- Device can work for at least a day with the current batteries
- Device can successfully log the CO_2 level, humidity , temperature and location coordinates
- Device batteries are not good enough to withstand more than 2 days.

12. Recommendations

Based on the result obtained and the conclusion drawn on the results, the following recommendations have been drawn:

- Device casing should be revised or redesigned and manufactured, as the casing limits the components used.
- Device software should be improved to allow entire device data manipulation on the software, such as drawing reports and storing data for future reference
- Device mode of downloading data should be improved to allow wireless downloading of data.
- Device batteries should be improved, 11000mA batteries are available hence they can be used, although they might need casing revision.
- Device microprocessor should be improved to accommodate future or evolution, use latest microprocessors with better clock frequency such as STM32F4

13. List of References

[1] World Health Organization, "Global tuberculosis report 2014," World health Organisation, Geneva, 2014.

[2] Carl D. Morrow, Darryl B.Eugene T. Richardson, Kalil,Linda-Gail Bekke,Robin Wood,Samuel Ginsberg, "Shared Air: A Renewed Focus on Ventilation for the Prevention of Tuberculosis Transmission," 7 May 2014. [Online]. Available: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0096334. [Accessed 25 July 2015].

[3] T. E. Herchiline, "Tuberculosis," 4 November 2014. [Online]. Available: http://emedicine.medscape.com/article/230802-overview#aw2aab6b2b2. [Accessed 2015 March 10].

[4] C. P. Davis, "Tuberculosis (TB)," MedicineNet, 1996. [Online]. Available: http://www.medicinenet.com/script/main/art.asp?articlekey=12510. [Accessed 20 March 2015].

[5] "Robert Koch and Tuberculosis," 9 December 2003. [Online]. Available: http://www.nobelprize.org/educational/medicine/tuberculosis/readmore.html. [Accessed 25 March 2015].

[6] Weebly, "The Effects of Poverty on the Health of Those Living In It," Weebly, [Online]. Available: http://theeffectsofpovertyonhealth.weebly.com/tuberculosis.html. [Accessed 20 July 2015].

[7] The Housing Development Agency (HDA), "South Africa: Informal Settlement Status (2013)," August 2013. [Online]. Available: http://www.thehda.co.za/uploads/images/HDA_South_Africa_Report_lr.pdf. [Accessed 26 June 2015].

[8] Mayo Clinic, "Tuberculosis (Diseases and Conditions)," Mayo Clinic, 1 August 2014. [Online]. Available: http://www.mayoclinic.org/diseases-conditions/tuberculosis/basics/risk-factors/con-20021761. [Accessed 23 July 2015].

[9] J. B.-Y. Tsui, Fundamentals of Global Positioning System Receivers, 2nd Edition ed., New Jersey: Werley, 2005.

[10] Smithsonian, "GPS: A New Constellation," Smithsonian Air and Space Museum, 12 December 1998. [Online]. Available: http://airandspace.si.edu/exhibitions/gps/. [Accessed 2015 May 25].

[11] Ordnance, "GPS and Satellites - Third Level," [Online]. Available: http://www.osi.ie/Education/Third-Level-Academic/GPS-and-Satellites.aspx. [Accessed 15 July 2014].

[12] D. DePriest, "Geometry view," in *How GPS Works*, 2002, pp. 1-2.

[13] D. Priest, "NMEA Data," [Online]. Available: http://www.gpsinformation.org/dale/nmea.htm. [Accessed 20 May 2015].

[14] AliExpress, "STM32F107VCT6 ORIGINAL MCU ARM 256KB FLASH MEM 100-LQFP STM32F107VCT6TR STM32F107," AliExpress, [Online]. Available: http://www.aliexpress.com/item/AVR-AVR-MCU-STM32F107VCT6-ORIGINAL-STOCK-LQFP100-32F107VC/1673366144.html. [Accessed 30 July 2015].

[15] STMicroelectronics, "STM32F107VC," STMicroelectronics, [Online]. Available: http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1564/PF221020. [Accessed 3 March 2015].

[16] u-blox, “NEO-6 ublox 6 GPS Modules,” [Online]. Available: https://www2.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf. [Accessed 1 July 2015].

[17] taoglas antenna solutions, “SPECIFICATION FXP611.07.0092C,” [Online]. Available: http://taoglas.com/images/product_images/original_images/FXP611.07.0092C.pdf. [Accessed 30 June 2015].

[18] CO2Meter, “COZIR Ambient 2/5/10K CO2 Sensor,” CO2Meter, [Online]. Available: http://www.co2meter.com/products/cozir-0-2-co2-sensor. [Accessed 10 June 2015].

[19] CO2Meter, “COZIR Datasheet,” CO2Meter, [Online]. Available: http://www.co2meters.com/Documentation/Datasheets/DS-GC-0010-COZIR-Ambient.pdf. [Accessed 10 June 2015].

[20] Maxim, “Low-Noise Step-Up DC-DC Converters,” Maxim, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX1790-MAX8715.pdf. [Accessed 15 June 2015].

[21] Maxim, “MAX1758 Stand-Alone,Switch-Mode Li+ Battery Charger with internal 28V Switch,” Maxim Integrated, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX1758.pdf. [Accessed 15 June 2015].

[22] RS Components, “Displaytech DT024CTFT-TS TFT LCD Colour Display,” RS Components, [Online]. Available: http://za.rs-online.com/web/p/lcd-colour-displays/7812992/. [Accessed 25 June 2015].

[23] ZYTRAX, “Micro-USB Type B and A/B Connectors,” ZYTRAX, [Online]. Available: http://www.zytrax.com/tech/pc/serial.html. [Accessed 30 June 2015].

[24] MAXIM, “Low-Noise Step-Up DC-DC Converters,” MAXIM, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX1790-MAX8715.pdf. [Accessed 10 July 2015].

[25] Linear Technology, “Dual Low Voltage Ideal Diode Controller,” Linear Technology, [Online]. Available: http://cds.linear.com/docs/en/datasheet/4353f.pdf. [Accessed 30 June 2014].

[26] J. D. Kramer and C. Jacky, “How to write biblos,” vol. 1, no. 1, 2006.

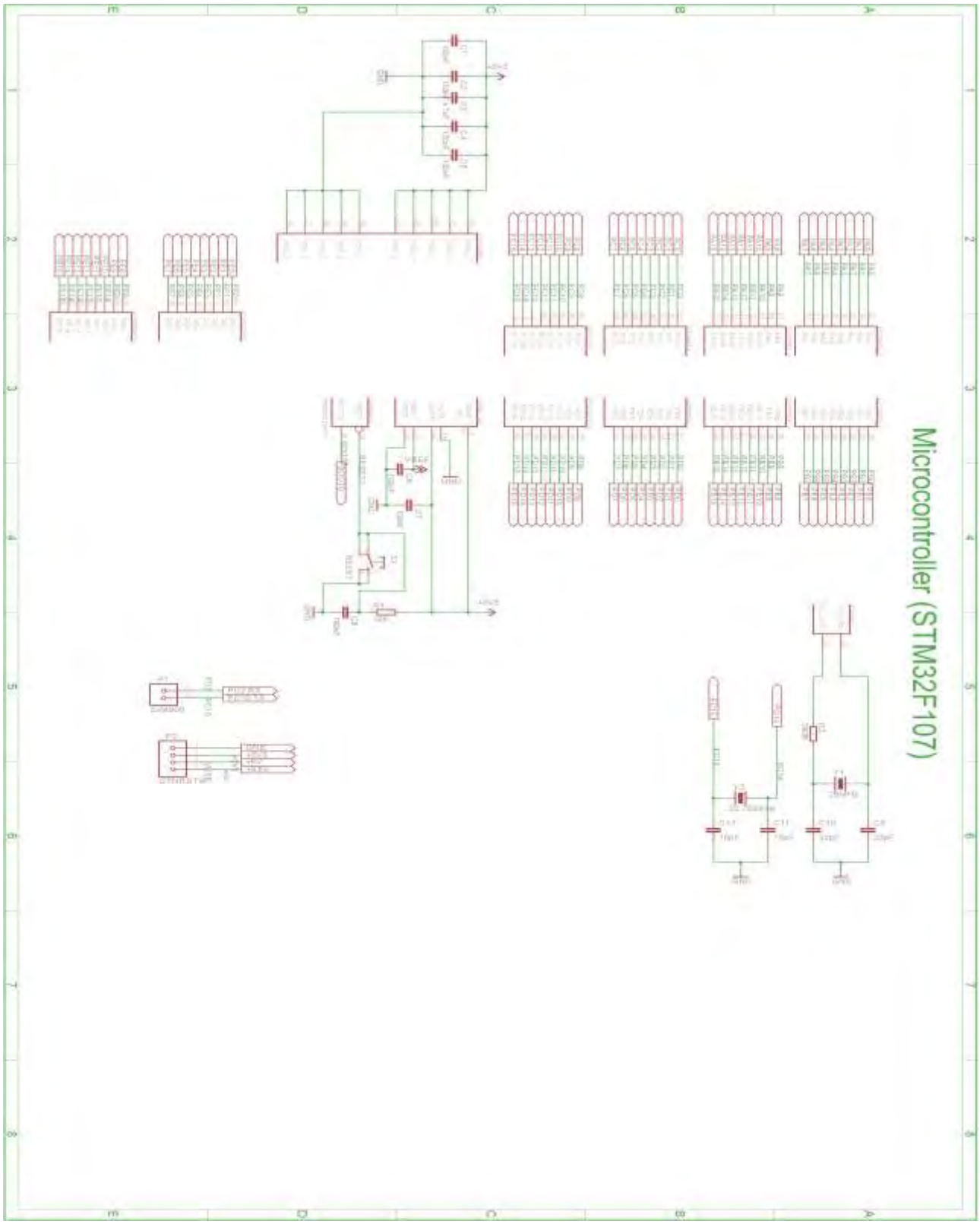
[27] Federal Aviation, “GNSS,” Federal Aviation, [Online]. Available: http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/faq/gps/. [Accessed 30 June 2015].

14. Appendices

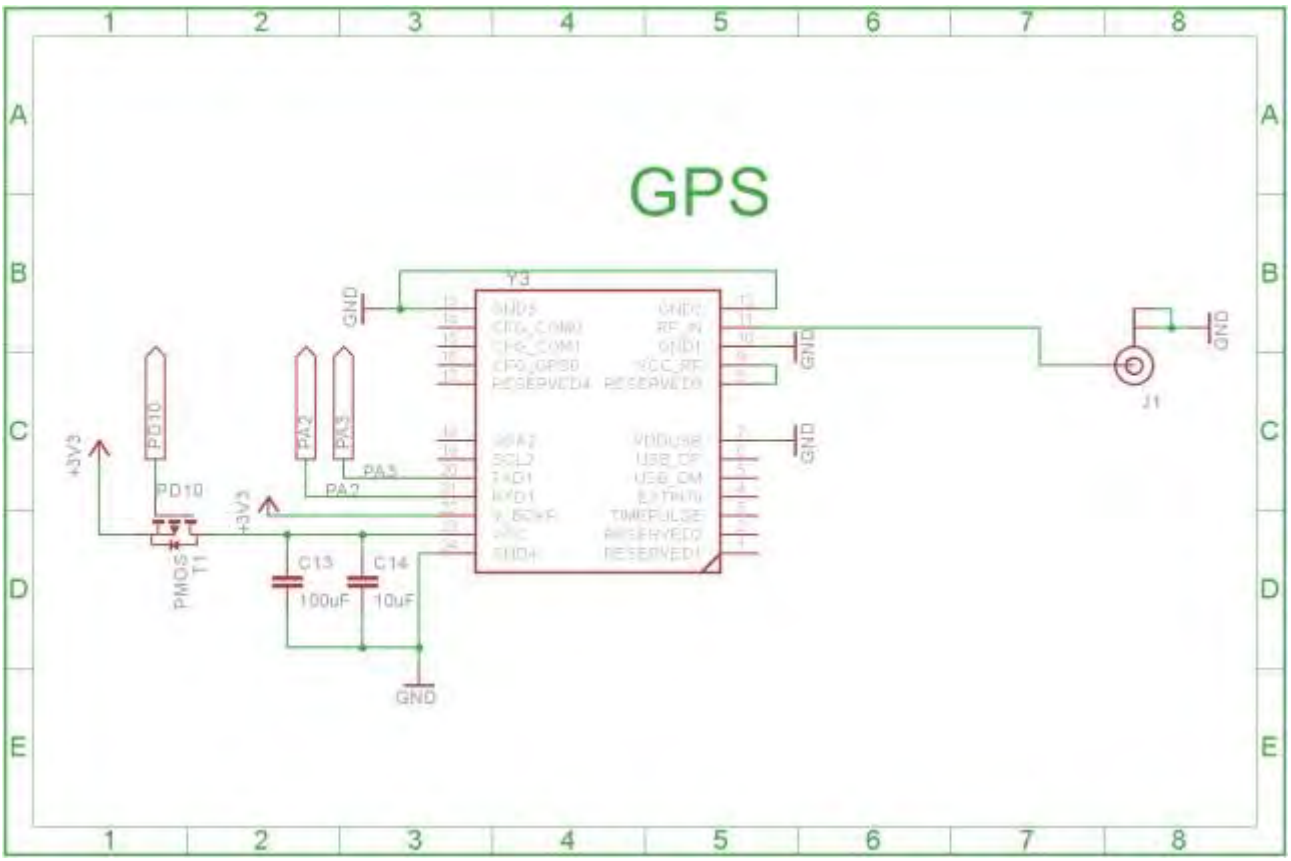
14.1 APPENDIX A: Schematic diagrams and board layout of Body Worn Carbon Dioxide Logger

In this appendix, the schematic diagrams and the board layout for the device are shown.

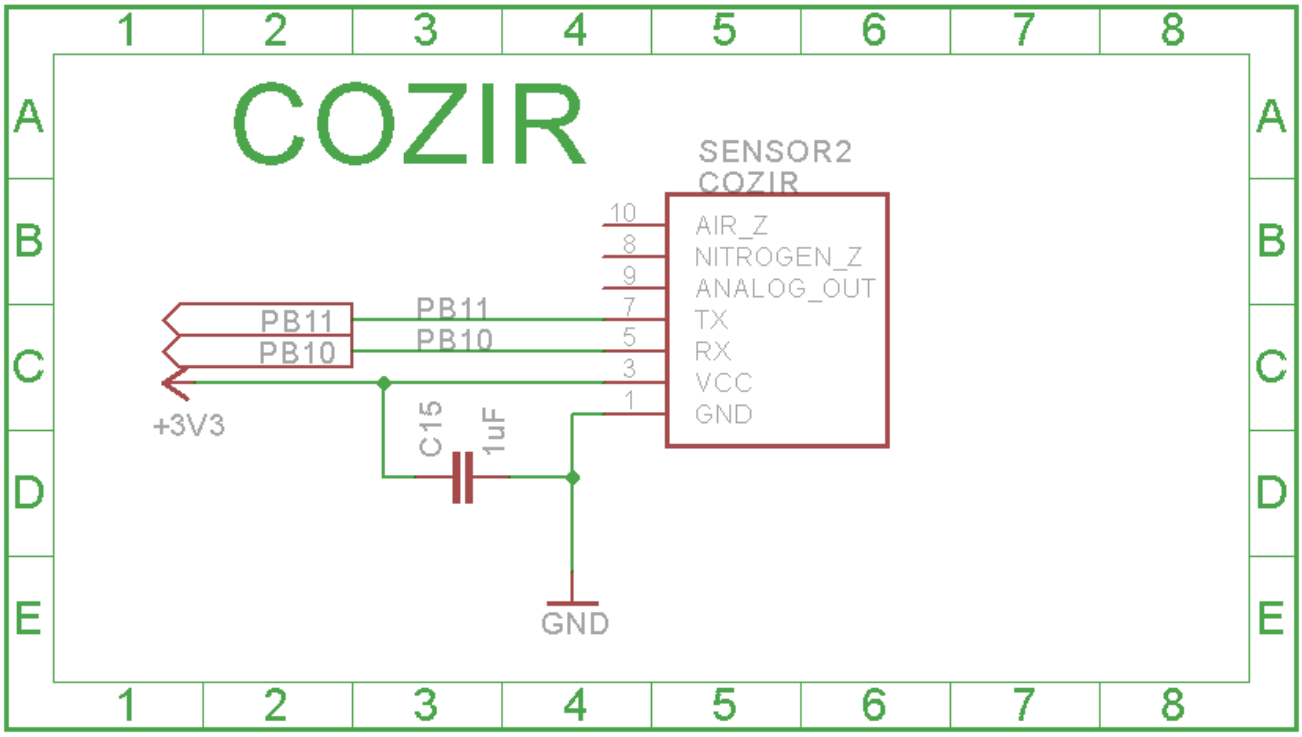
14.1.1 Microcontroller Connection



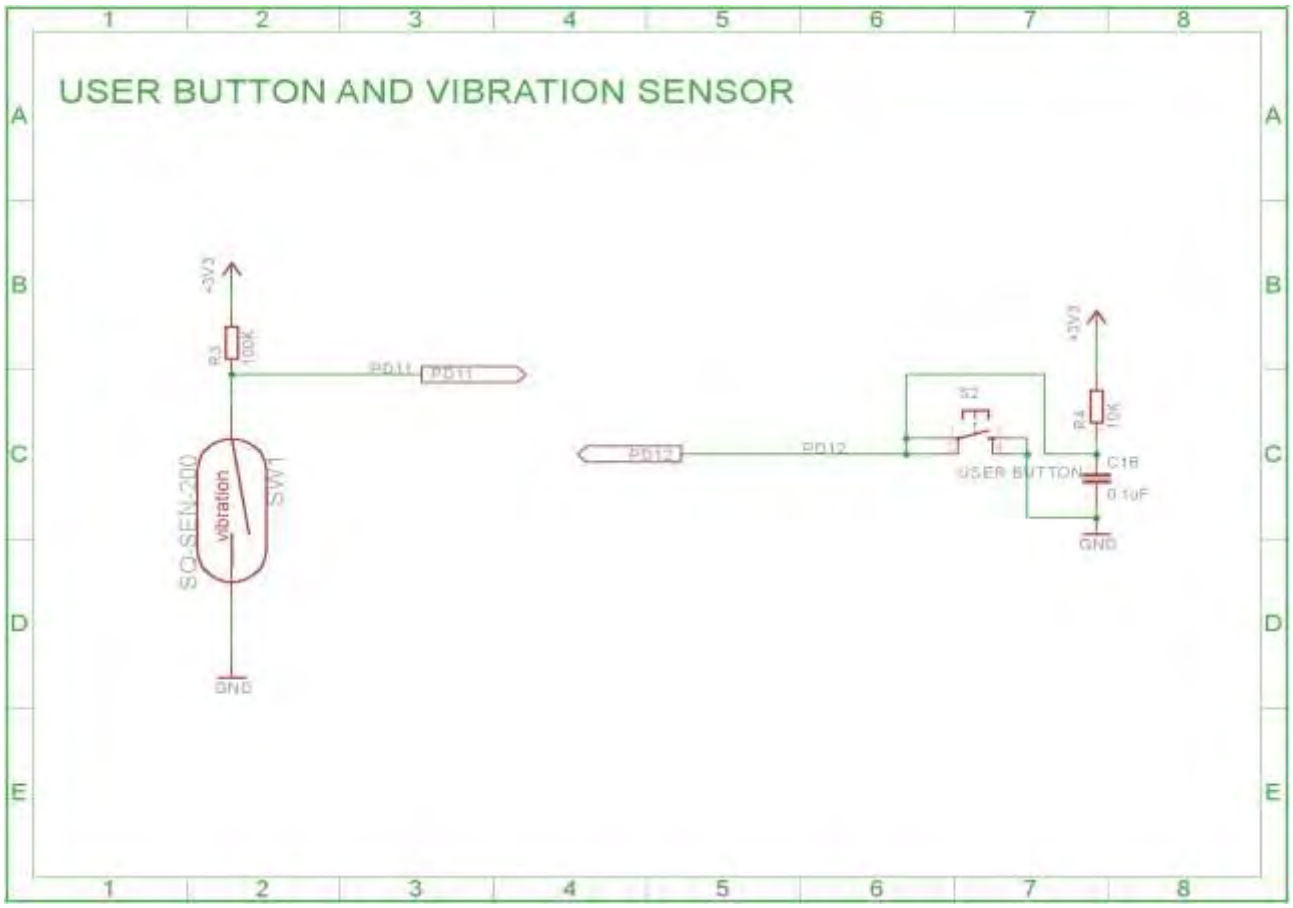
14.1.2 The GPS connection



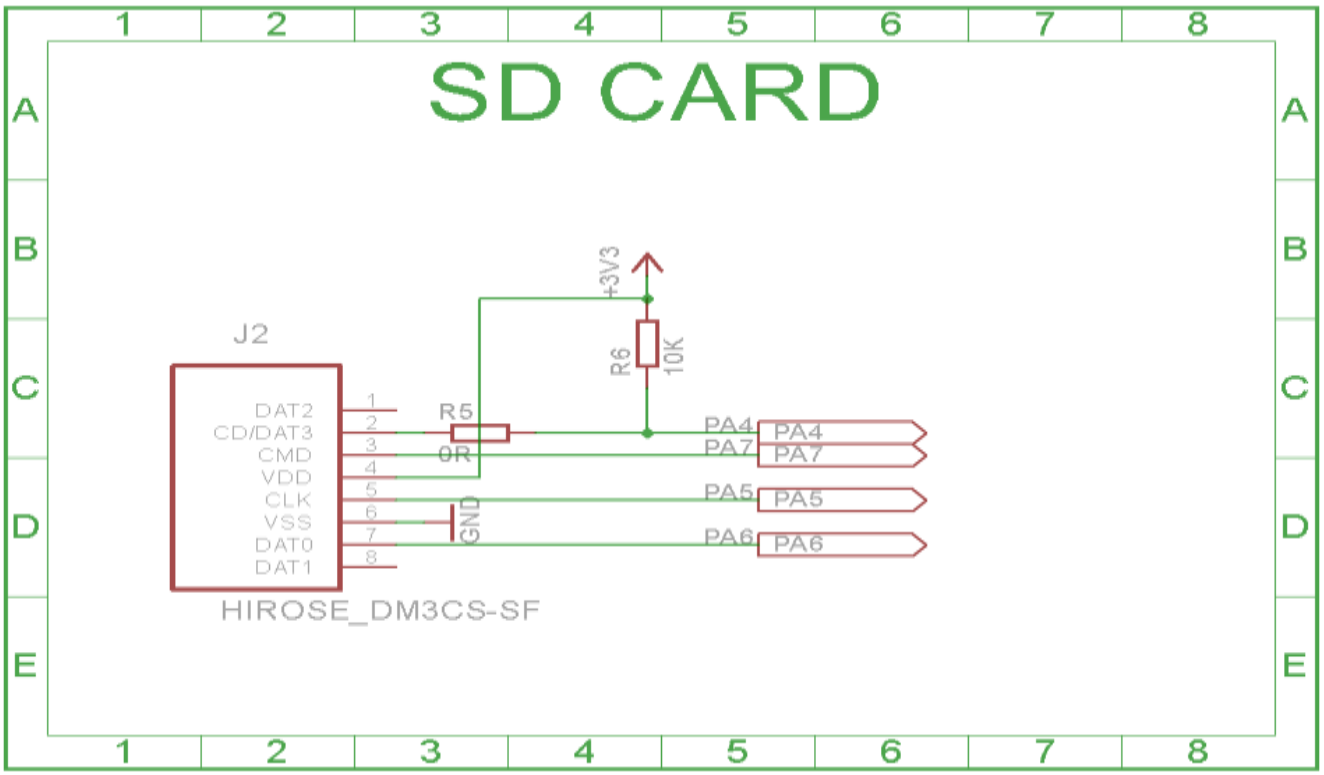
14.1.3 The carbon dioxide Transducer connection



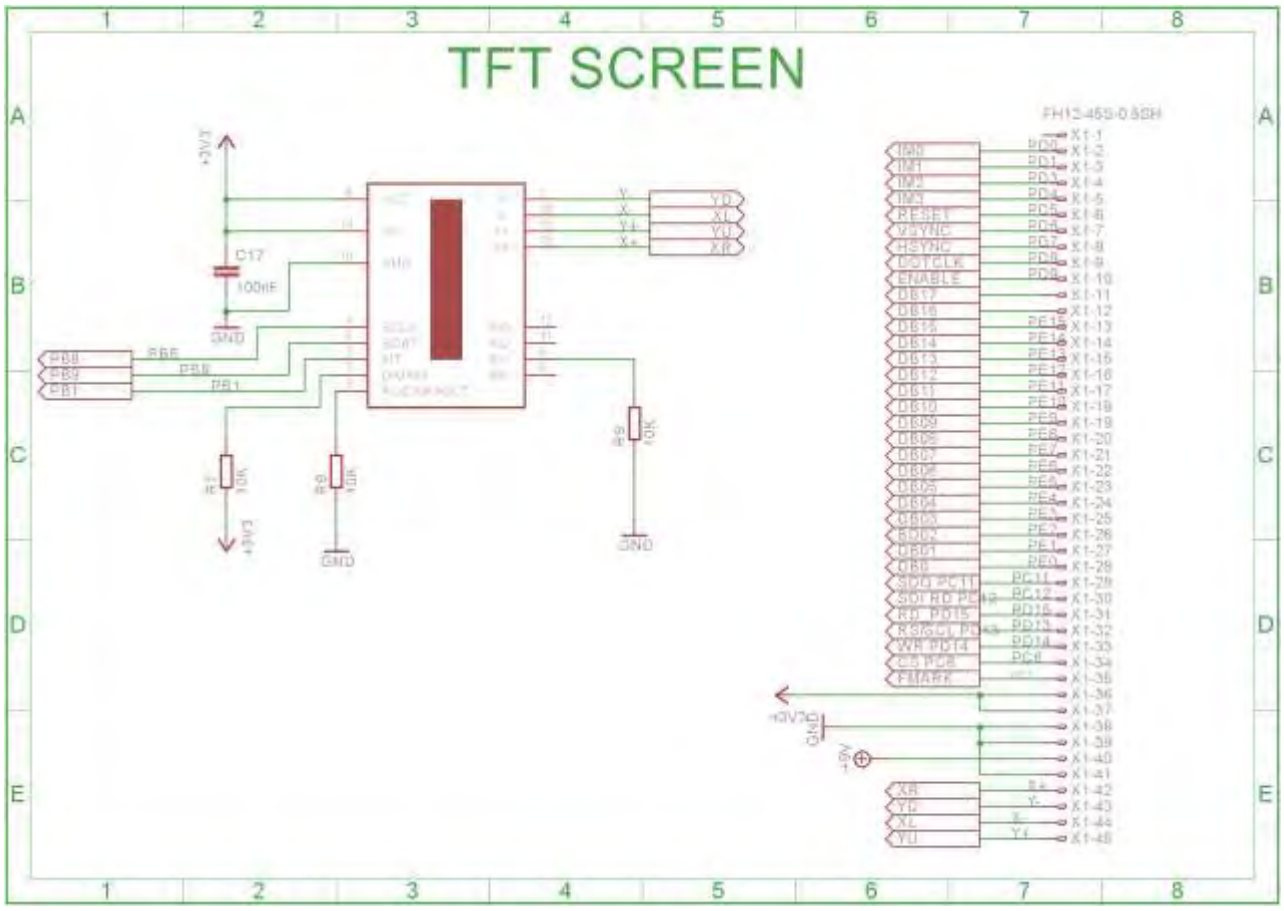
14.1.4 User Interaction button and Control Sensor



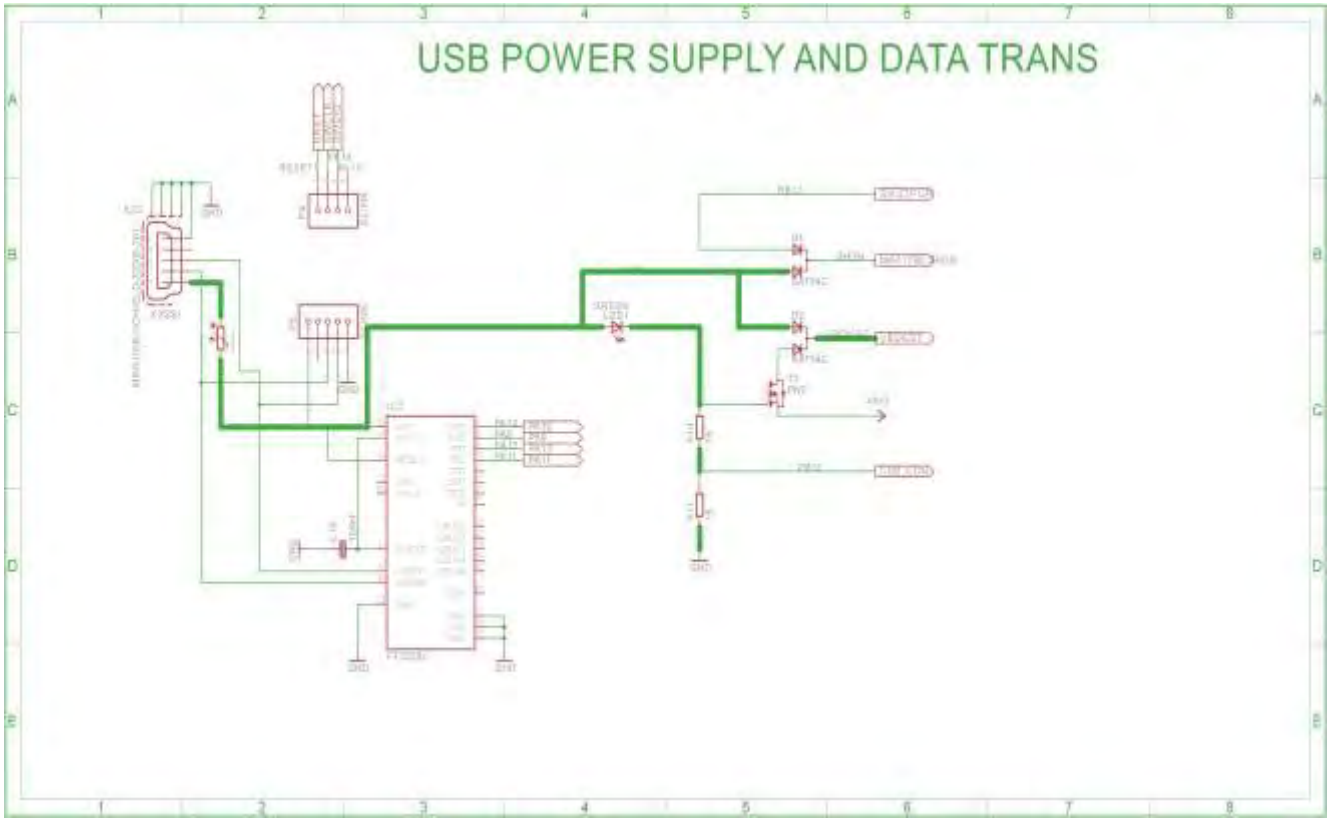
14.1.5 SD Card Connection



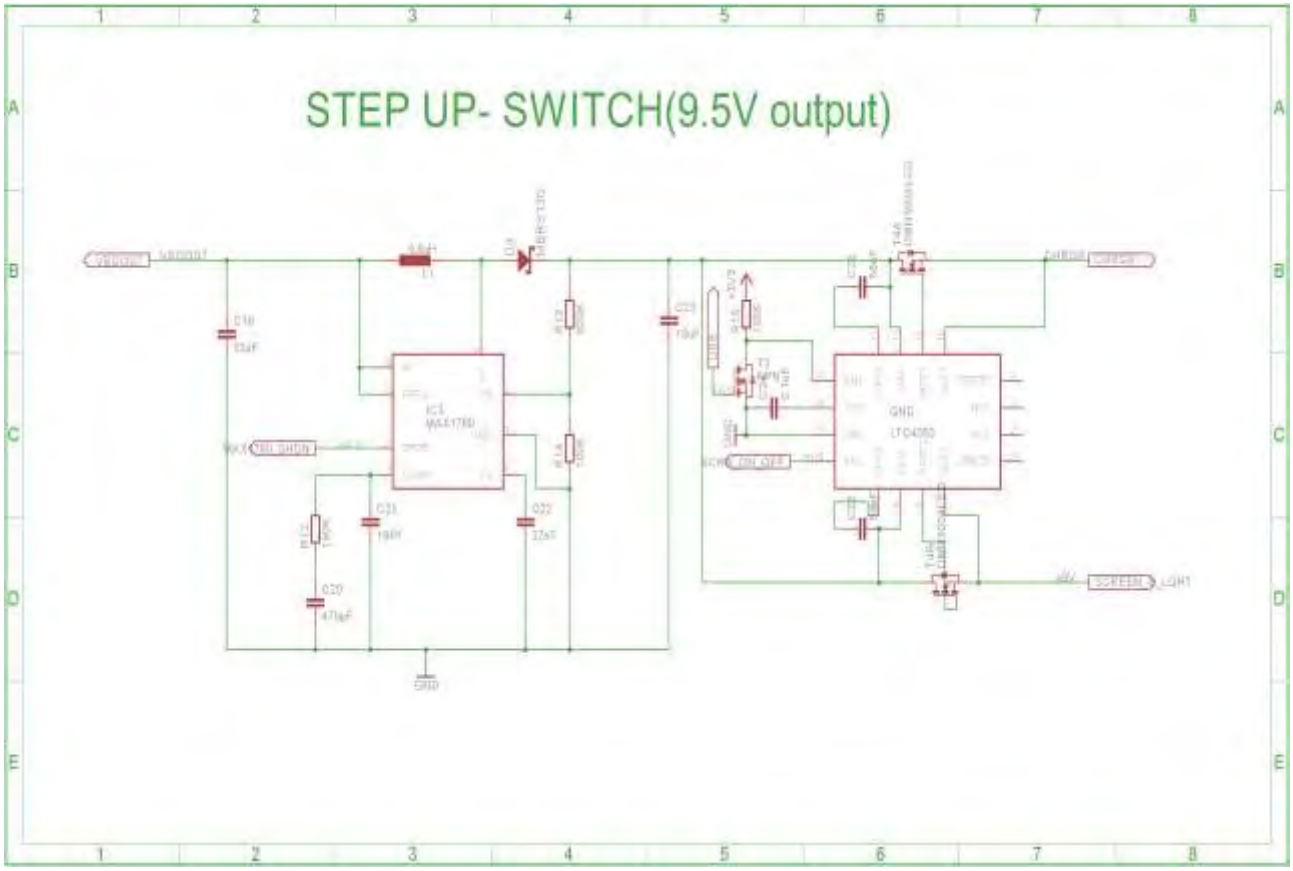
14.1.6 Touch Screen Connection



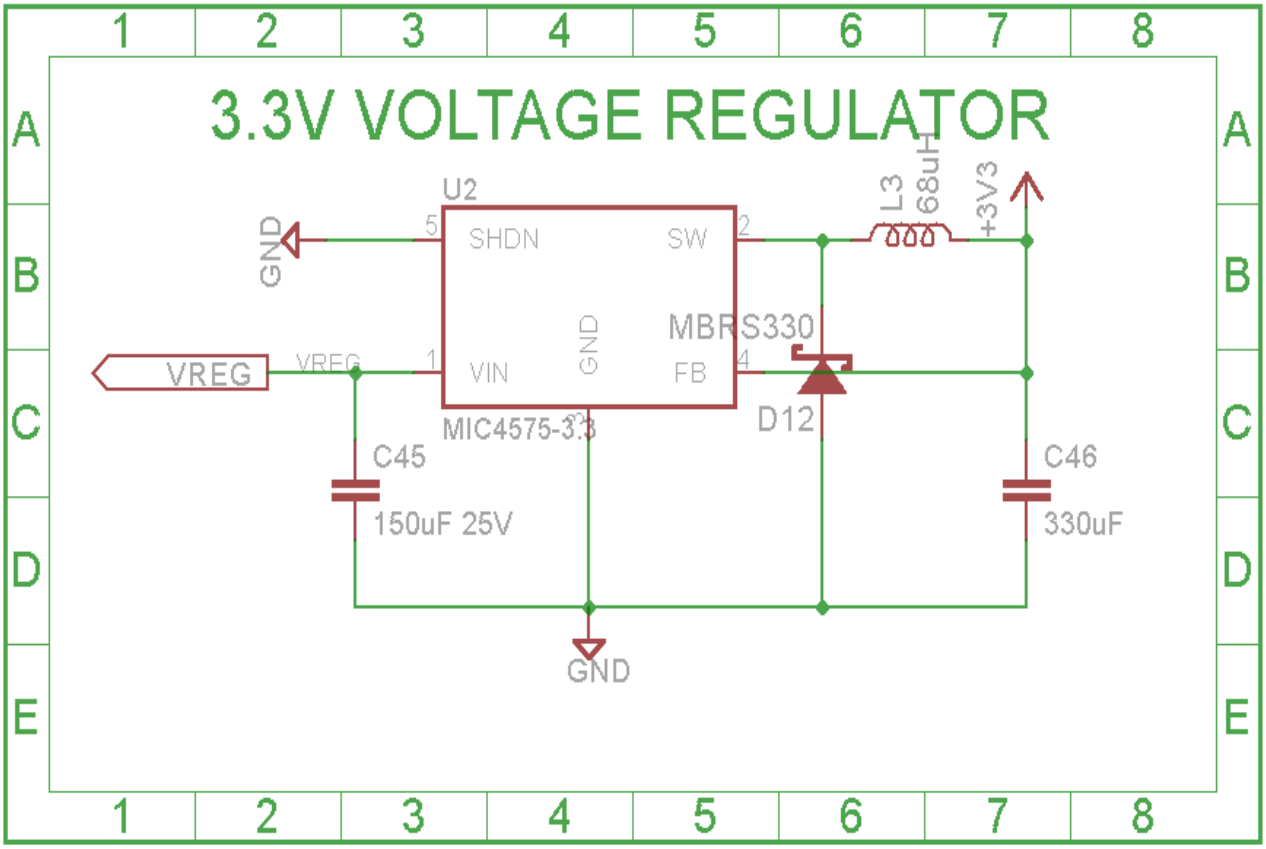
14.1.7 The Power Supply connection



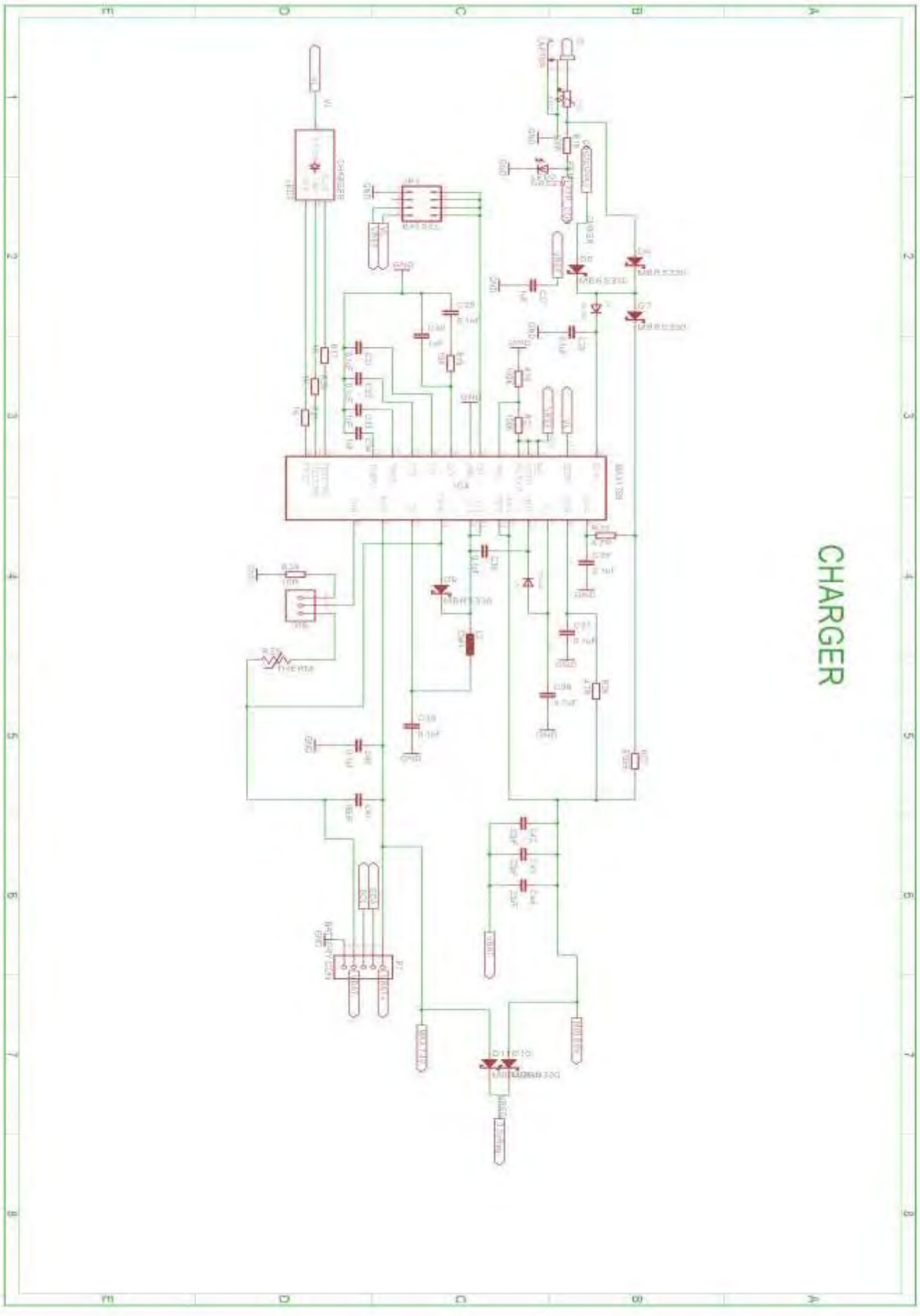
14.1.8 9.5 Voltage Regulator



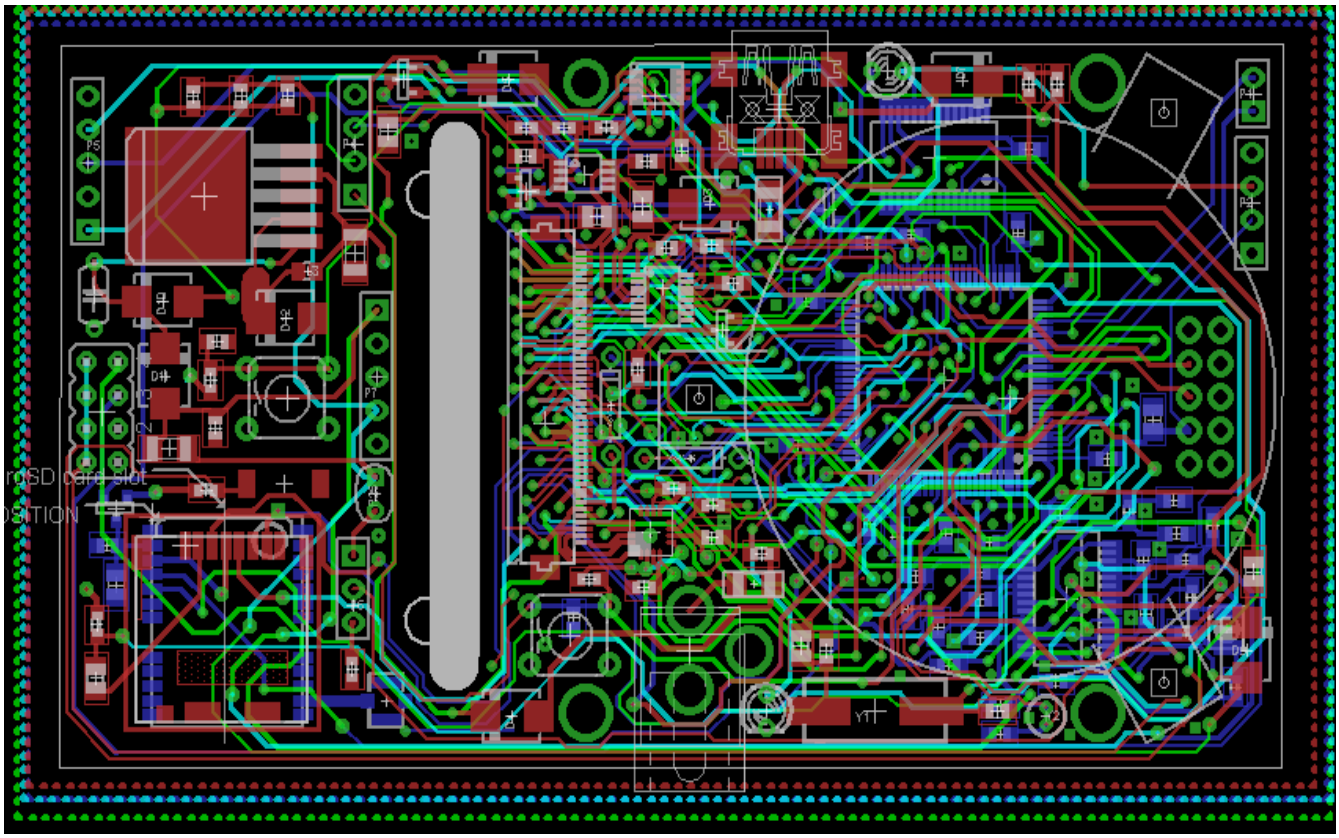
14.1.9 3.3V Switch Regulator

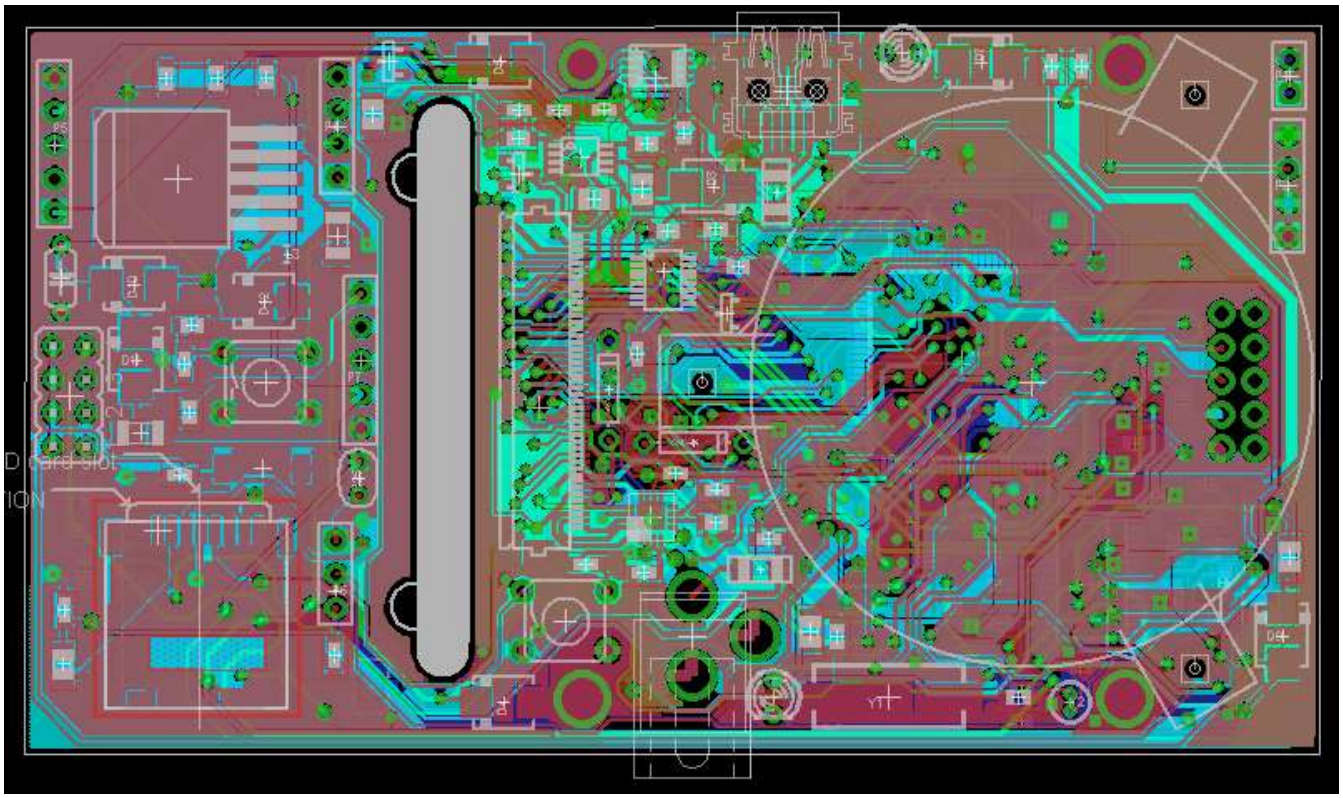


14.1.10 The Charger Circuit diagram



14.1.11 4 Layers Device Board Diagram

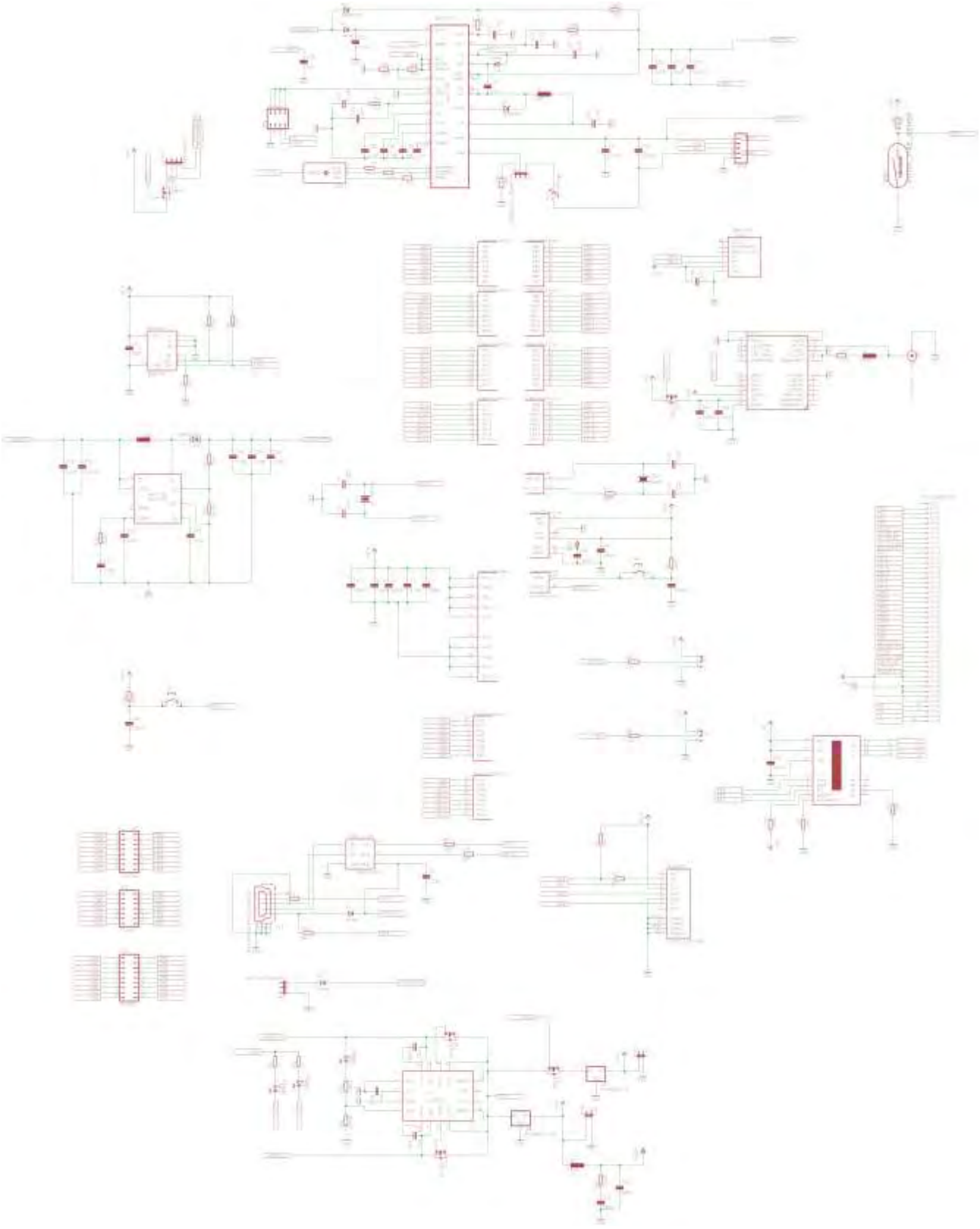




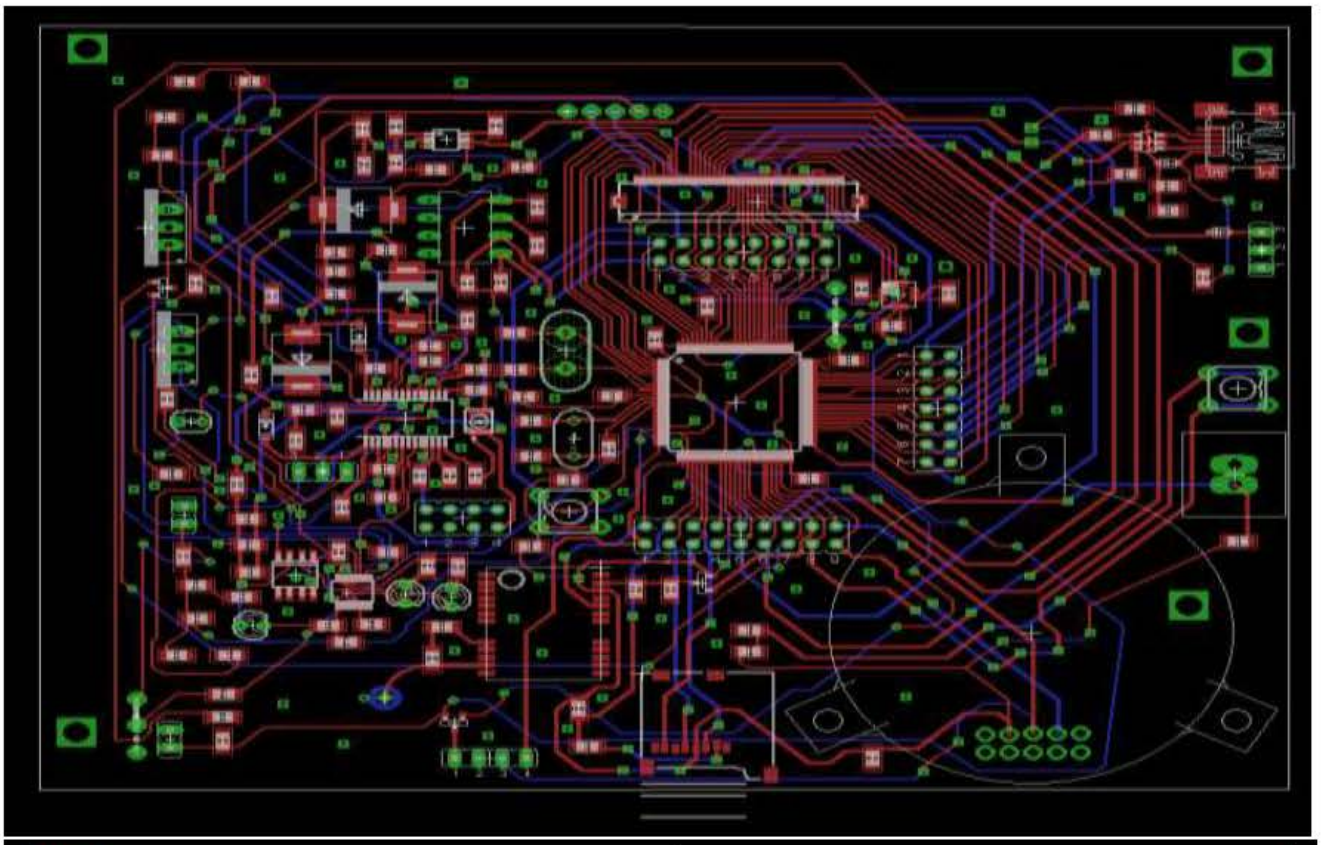
14.2 APPENDIX B: Schematic diagrams and board layout of Carbon Dioxide Body Worn Device Model 1

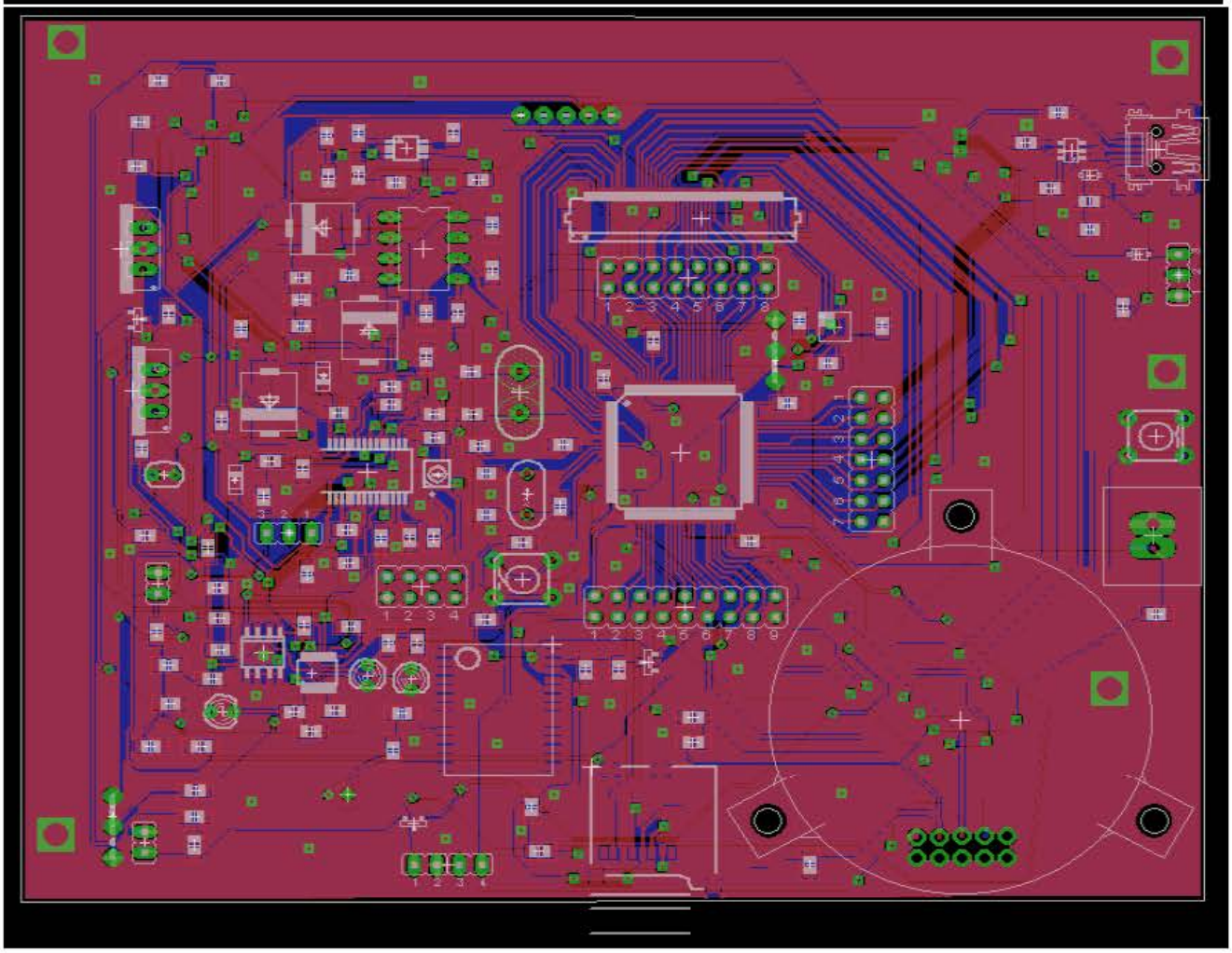
In this appendix, the circuit diagram and the board layout of the initial design of the device are shown:

14.2.1 Initial Model Circuit diagram



14.2.2 Initial Model Board Layout

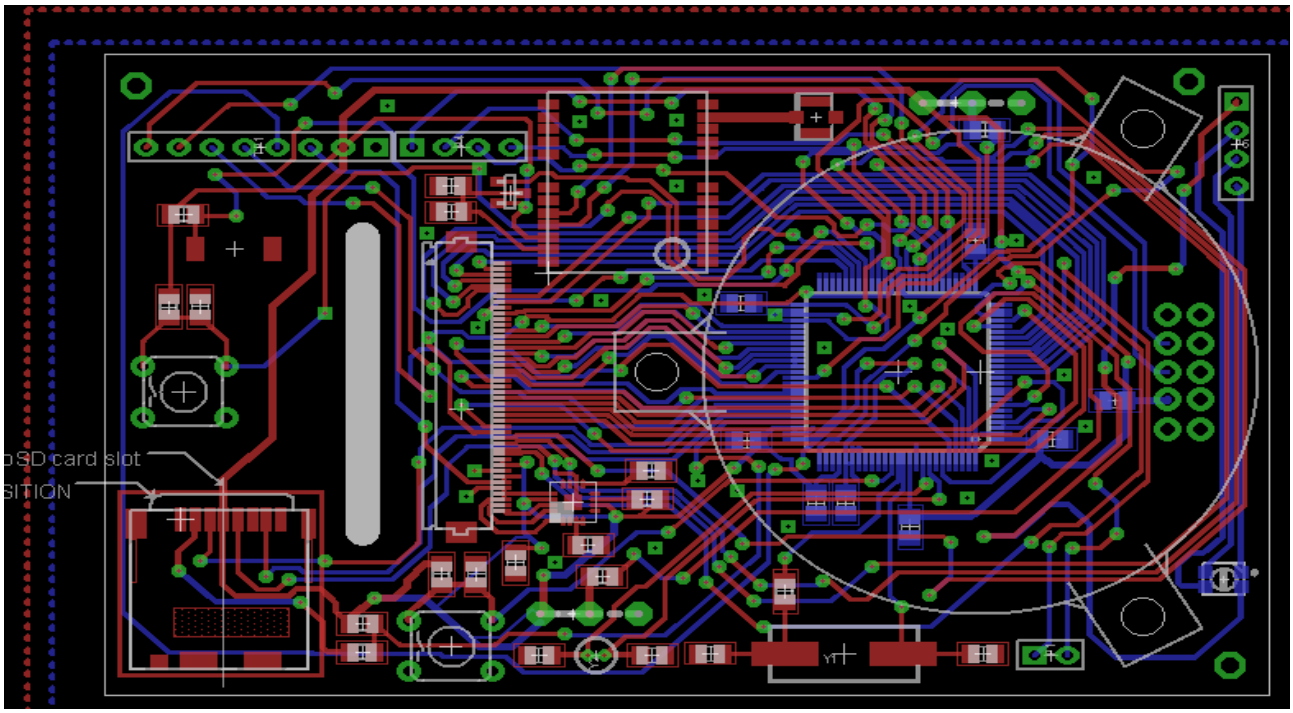


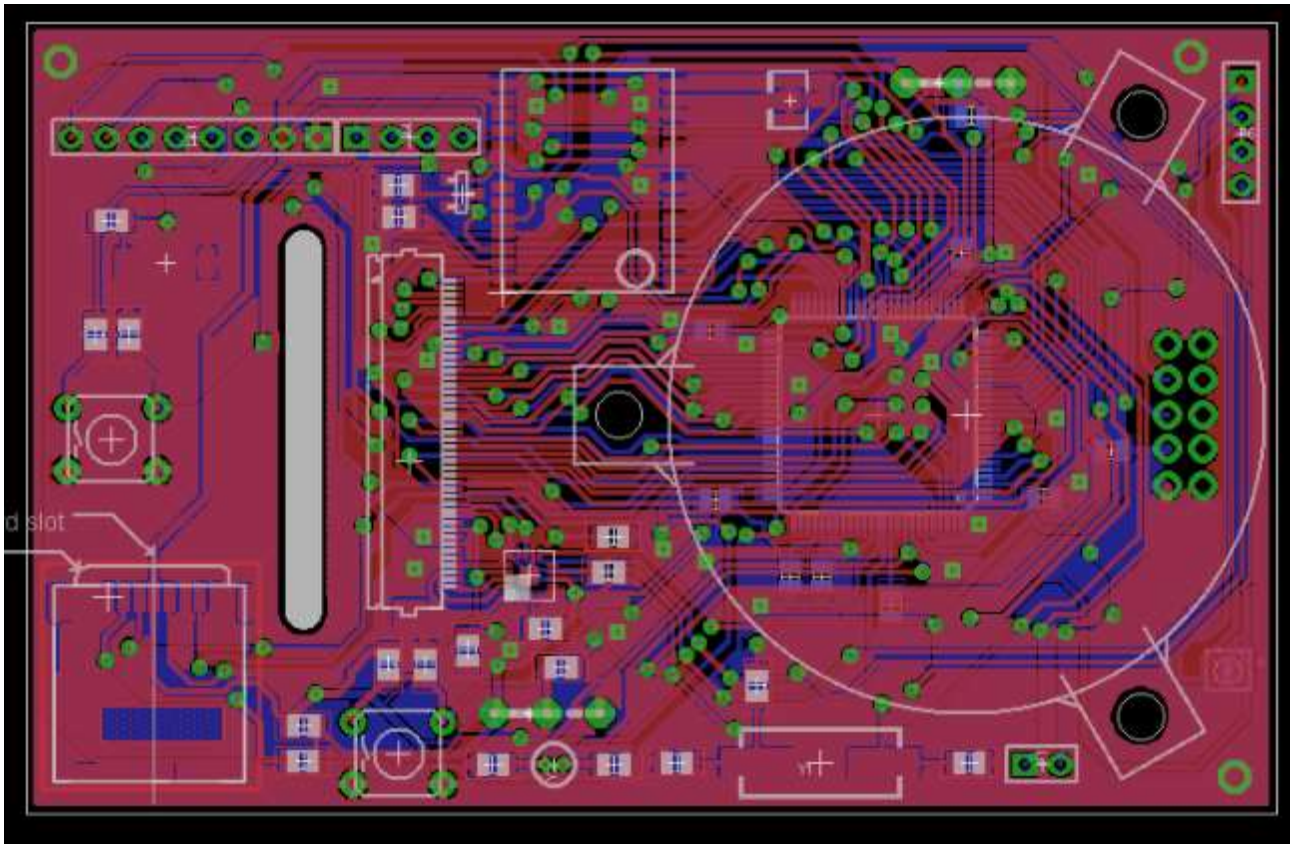


14.3 APPENDIX C: Schematic diagrams and board layout of Carbon Dioxide Body Worn Device Model 2

The schematic diagrams for this model for individual models are the same as model 3 board layout. The power board layout has been included in Appendix D.

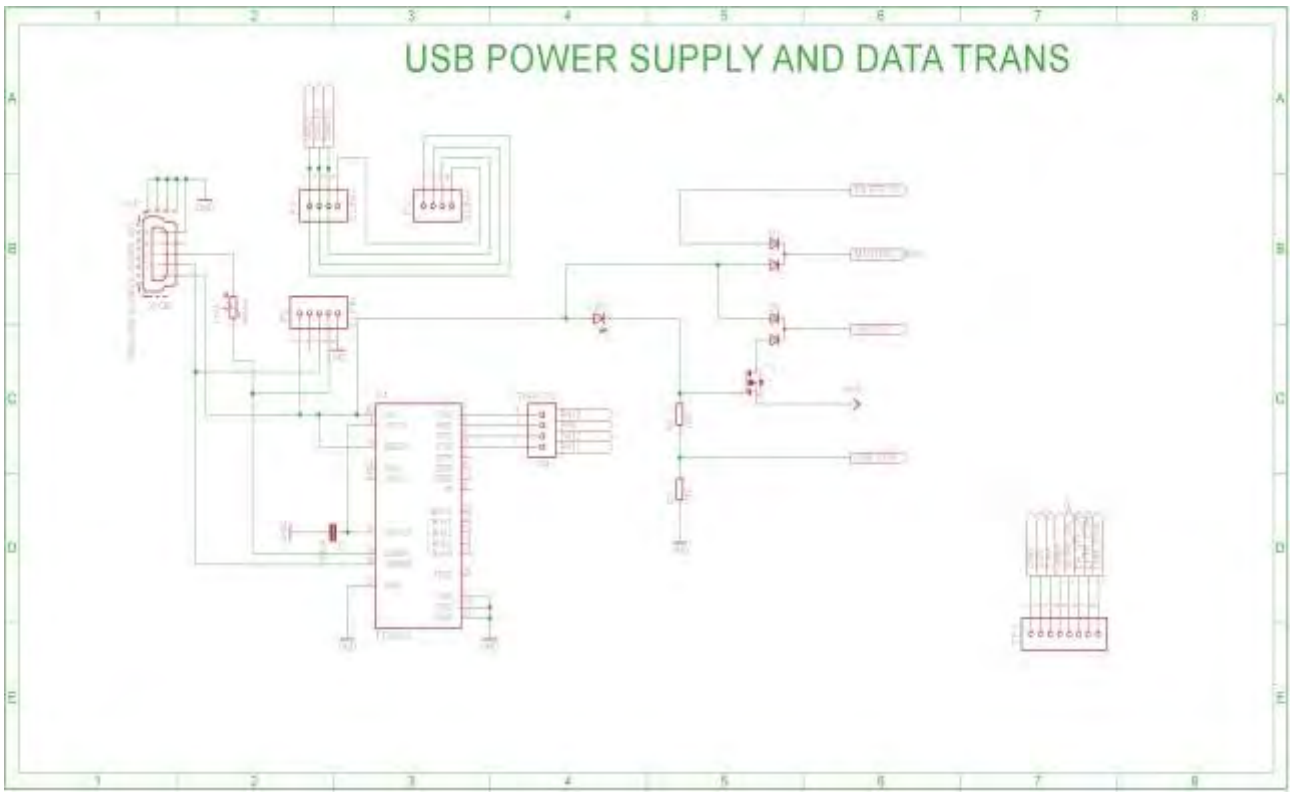
14.3.1 Model 2 Board layout

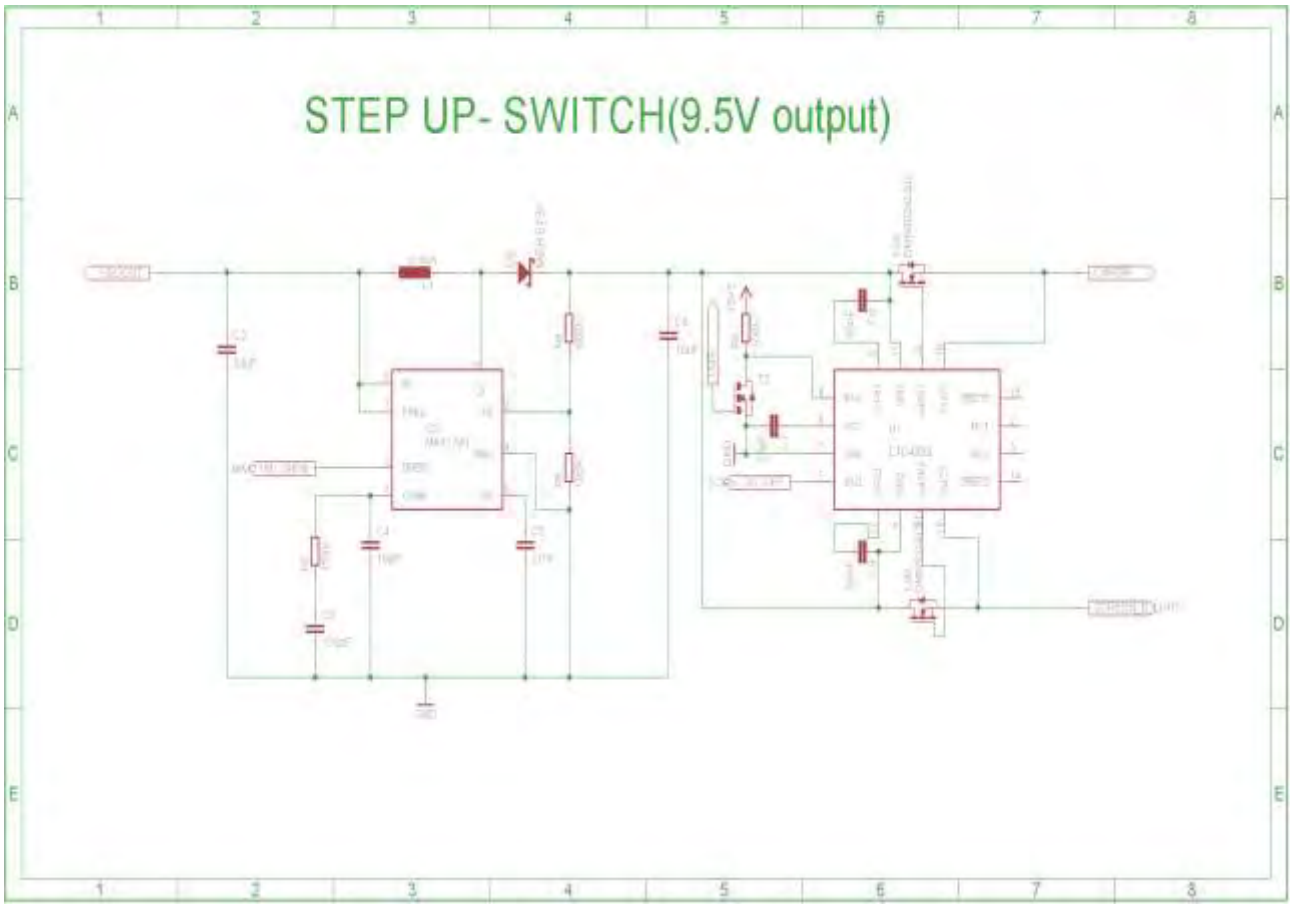


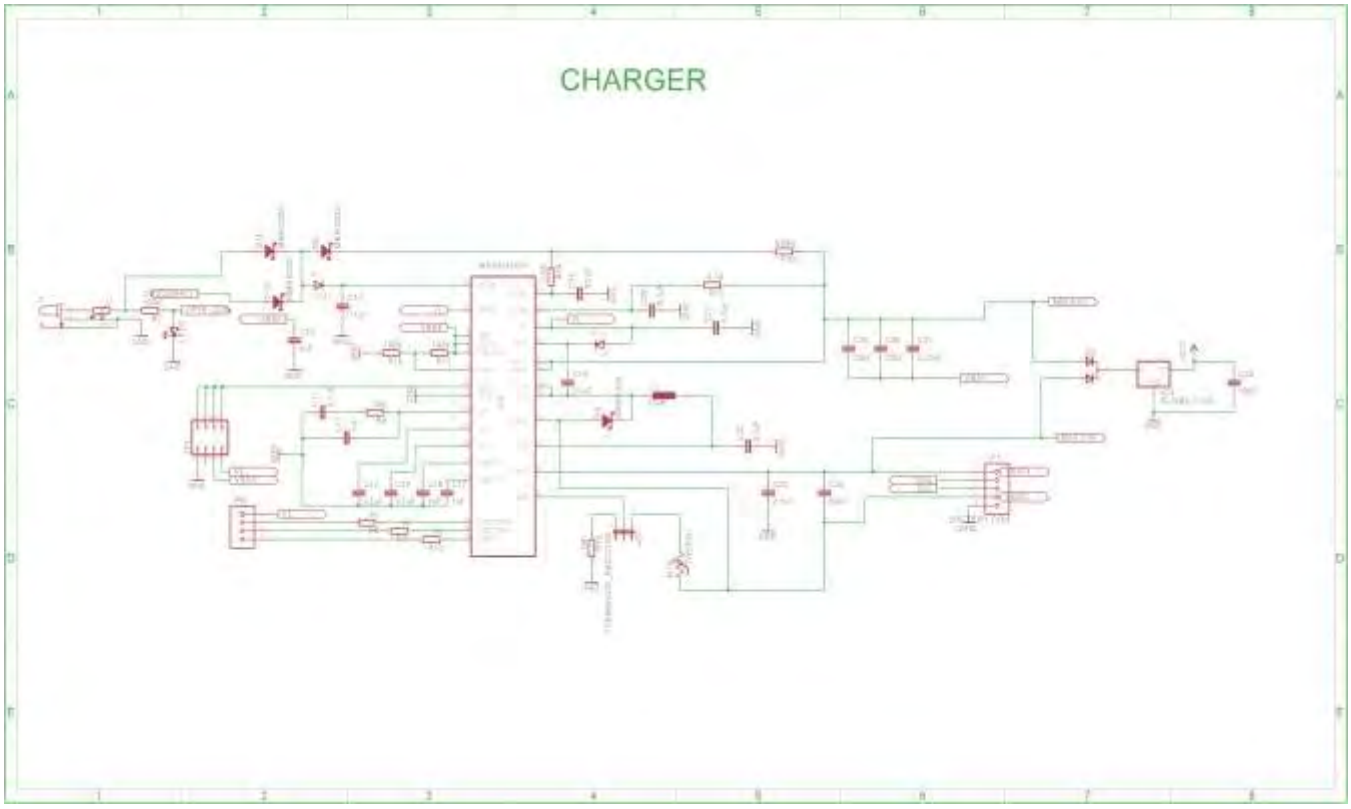


14.4 APPENDIX D: Schematic diagrams and board layout of Carbon Dioxide Body Worn Device Model 2 Power board

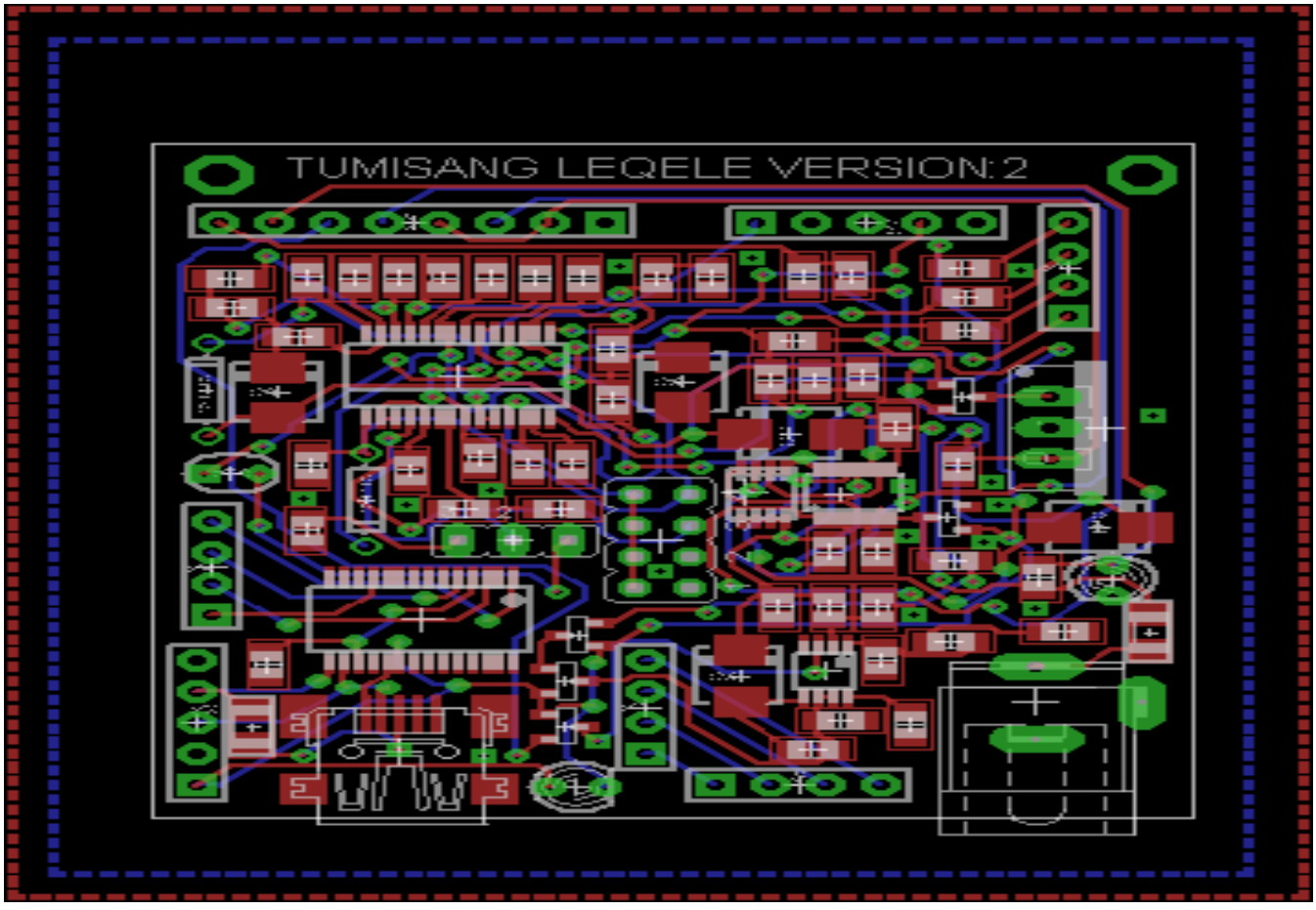
14.4.1 Model 2 Power supply circuit diagram







14.4.2 Model 2 Power Board



14.5 APPENDIX E: Device Live data download

TIME	HUMIDITY	TEMPERATURE	C02 LEVEL
04:47:11	54.70%	29.2 °C	3033 ppm
04:47:11	54.60%	29.2 °C	3010 ppm
04:47:12	54.60%	29.2 °C	2987 ppm
04:47:12	54.60%	29.2 °C	2971 ppm
04:47:13	54.60%	29.3 °C	2965 ppm
04:47:13	54.60%	29.2 °C	2948 ppm
04:47:14	54.50%	29.3 °C	2920 ppm
04:47:14	54.50%	29.3 °C	2915 ppm
04:47:15	54.50%	29.3 °C	2887 ppm
04:47:15	54.50%	29.3 °C	2864 ppm
04:47:16	54.50%	29.3 °C	2853 ppm
04:47:16	54.40%	29.3 °C	2831 ppm
04:47:17	54.40%	29.3 °C	2814 ppm
04:47:17	54.40%	29.3 °C	2798 ppm
04:47:18	54.40%	29.3 °C	2765 ppm
04:47:18	54.40%	29.3 °C	2760 ppm
04:47:19	54.40%	29.3 °C	2749 ppm
04:47:19	54.30%	29.3 °C	2733 ppm
04:47:20	54.30%	29.3 °C	2722 ppm
04:47:20	54.30%	29.3 °C	2706 ppm
04:47:21	54.30%	29.3 °C	2690 ppm
04:47:21	54.30%	29.3 °C	2674 ppm
04:47:22	54.30%	29.3 °C	2657 ppm
04:47:22	54.20%	29.3 °C	2652 ppm
04:47:23	54.20%	29.3 °C	2641 ppm
04:47:23	54.20%	29.3 °C	2636 ppm
04:47:24	54.20%	29.3 °C	2621 ppm
04:47:24	54.30%	29.3 °C	2615 ppm
04:47:25	54.30%	29.3 °C	2610 ppm
04:47:25	54.30%	29.3 °C	2600 ppm
04:47:26	54.40%	29.3 °C	2589 ppm
04:47:26	54.40%	29.3 °C	2589 ppm
04:47:27	54.40%	29.3 °C	2584 ppm
04:47:27	54.40%	29.3 °C	2589 ppm
04:47:28	54.40%	29.3 °C	2585 ppm
04:47:28	54.40%	29.3 °C	2580 ppm
04:47:29	54.30%	29.3 °C	2580 ppm
04:47:29	54.30%	29.3 °C	2575 ppm
04:47:30	54.30%	29.3 °C	2580 ppm
04:47:30	54.30%	29.3 °C	2585 ppm
04:47:31	54.30%	29.3 °C	2585 ppm
04:47:31	54.30%	29.3 °C	2580 ppm
04:47:32	54.30%	29.3 °C	2580 ppm

04:47:32	54.30%	29.3 °C	2590 ppm
04:47:33	54.30%	29.3 °C	2590 ppm
04:47:33	54.30%	29.3 °C	2596 ppm
04:47:34	54.30%	29.3 °C	2601 ppm
04:47:34	54.30%	29.3 °C	2596 ppm
04:47:35	54.30%	29.3 °C	2596 ppm
04:47:35	54.30%	29.3 °C	2601 ppm
04:47:36	54.30%	29.3 °C	2611 ppm
04:47:36	54.40%	29.3 °C	2611 ppm
04:47:37	54.40%	29.3 °C	2616 ppm
04:47:37	54.40%	29.3 °C	2627 ppm
04:47:38	54.40%	29.3 °C	2632 ppm
04:47:38	54.40%	29.3 °C	2632 ppm
04:47:39	54.40%	29.4 °C	2642 ppm
04:47:39	54.50%	29.4 °C	2653 ppm
04:47:40	54.50%	29.4 °C	2669 ppm
04:47:40	54.50%	29.4 °C	2675 ppm
04:47:41	54.50%	29.4 °C	2680 ppm
04:47:41	54.50%	29.4 °C	2696 ppm
04:47:42	54.60%	29.4 °C	2712 ppm
04:47:42	54.60%	29.4 °C	2729 ppm
04:47:43	54.60%	29.4 °C	2745 ppm
04:47:43	54.60%	29.4 °C	2762 ppm
04:47:44	54.70%	29.4 °C	2778 ppm
04:47:44	54.70%	29.4 °C	2789 ppm
04:47:45	54.70%	29.4 °C	2805 ppm
04:47:45	54.80%	29.4 °C	2822 ppm
04:47:46	54.80%	29.4 °C	2833 ppm
04:47:46	54.80%	29.4 °C	2849 ppm
04:47:47	54.80%	29.4 °C	2872 ppm
04:47:47	54.80%	29.4 °C	2894 ppm
04:47:48	54.80%	29.4 °C	2911 ppm
04:47:48	54.90%	29.4 °C	2922 ppm
04:47:49	54.90%	29.4 °C	2933 ppm
04:47:49	54.90%	29.4 °C	2956 ppm
04:47:50	54.90%	29.4 °C	2973 ppm
04:47:50	54.90%	29.4 °C	2984 ppm
04:47:51	54.90%	29.4 °C	3001 ppm
04:47:51	54.90%	29.4 °C	3030 ppm
04:47:52	55%	29.4 °C	3054 ppm
04:47:52	55%	29.4 °C	3083 ppm
04:47:53	55%	29.4 °C	3100 ppm
04:47:53	55%	29.4 °C	3124 ppm
04:47:54	55%	29.4 °C	3135 ppm
04:47:54	55%	29.4 °C	3170 ppm
04:47:55	55%	29.4 °C	3200 ppm

04:47:55	55%	29.4 °C	3224 ppm
04:47:56	55%	29.4 °C	3242 ppm
04:47:56	55%	29.4 °C	3255 ppm
04:47:57	55.10%	29.4 °C	3274 ppm
04:47:57	55.10%	29.4 °C	3292 ppm
04:47:58	55.10%	29.4 °C	3323 ppm
04:47:58	55.10%	29.4 °C	3341 ppm
04:47:59	55.10%	29.4 °C	3365 ppm
04:47:59	55.20%	29.4 °C	3390 ppm
04:48:00	55.20%	29.4 °C	3415 ppm
04:48:00	55.20%	29.4 °C	3447 ppm
04:48:01	55.30%	29.4 °C	3478 ppm
04:48:01	55.30%	29.4 °C	3517 ppm
04:48:02	55.40%	29.4 °C	3561 ppm
04:48:02	55.40%	29.4 °C	3600 ppm
04:48:03	55.40%	29.4 °C	3639 ppm
04:48:03	55.50%	29.4 °C	3684 ppm
04:48:04	55.50%	29.4 °C	3730 ppm
04:48:04	55.60%	29.4 °C	3777 ppm
04:48:05	55.60%	29.4 °C	3824 ppm
04:48:05	55.70%	29.4 °C	3878 ppm
04:48:06	55.70%	29.4 °C	3919 ppm
04:48:06	55.80%	29.4 °C	3973 ppm
04:48:07	55.80%	29.4 °C	4035 ppm
04:48:07	55.90%	29.4 °C	4099 ppm
04:48:08	55.90%	29.4 °C	4156 ppm
04:48:08	55.90%	29.4 °C	4205 ppm
04:48:09	56%	29.4 °C	4262 ppm
04:48:09	56%	29.4 °C	4307 ppm
04:48:10	56%	29.4 °C	4373 ppm
04:48:10	56.10%	29.4 °C	4431 ppm
04:48:11	56.10%	29.4 °C	4475 ppm
04:48:11	56.10%	29.4 °C	4528 ppm
04:48:12	56.20%	29.5 °C	4581 ppm
04:48:12	56.20%	29.5 °C	4627 ppm
04:48:13	56.20%	29.5 °C	4695 ppm
04:48:13	56.30%	29.5 °C	4749 ppm
04:48:14	56.30%	29.5 °C	4796 ppm
04:48:14	56.30%	29.5 °C	4851 ppm
04:48:15	56.30%	29.5 °C	4906 ppm
04:48:15	56.40%	29.5 °C	4946 ppm
04:48:16	56.40%	29.5 °C	5002 ppm
04:48:16	56.40%	29.5 °C	5067 ppm
04:48:17	56.50%	29.5 °C	5116 ppm
04:48:17	56.50%	29.5 °C	5189 ppm
04:48:18	56.50%	29.5 °C	5255 ppm