Development of a Cable Odometer with a Network Interface

prepared by

Teboho N. Nyareli

A dissertation submitted to the Department of Electrical Engineering,
University of Cape Town, in fulfillment of the requirements
for the degree of Master of Science in Engineering.

Cape Town, July 2003

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in any other university.

Signature of Author .....

Cape Town

July 2003

Abstract

The dissertation involves the design of a Cable Odometer that determines the position , direction and speed of a Radar as it surveys the subsurface. The Radar moves horizontally across the surface or up and down a borehole.

The distance moved by the Radar as it surveys the subsurface is determined from the rotations of a wheel, over which a cable passes. The Radar is attached to this cable. Information on the distance moved is collected by Odometer and used for further analysis on the Radar data.

The user requirement states that in some cases data collected by Odometer is used when triggering Radar directly to make surveys at fixed interval. In other cases the Radar operates autonomously. For the latter case the data collected by both Radar and Odometer is related by a time-stamp.

The Odometer is controlled on a web page. Depending on the mode of operation chosen the user can transfer the time-stamped data to remote data storage via a network or store it on board (i.e. in Dataflash) for later downloading.

Acknowledgements

I would like to acknowledge all the people who helped and supported me in completing this project.

- My supervisor Prof. Michael Inggs for the guidance and patience.
- Alan Langman for his advice.
- The AVR-GCC mail list and ICC-AVR mail list for the technical support.
- Special thanks to Judd Bard from Atmel AVR development centre.
- My parents and family for the inspiration and moral support.

Finally, I would like to thank my friends for the moral support they have given.

Contents

List of Figures

List of Tables

List of Symbols

B_{14} — A block of 14 file
f_{clk} — frequency of clock oscillator
t_p — Access time for a page in Dataflash

Nomenclature

CPLD — Complex Programmable Logic Device.

Frame — Refers to a portion of a packet.

Packet— Refers to the entire serial string transfered over the Ethernet network.

eCos— an open source real time operating system intended for embedded applications.

Time-stamp — The time that is appended to data.

Odometer— The processor board that does all the processing of the data from encoder.

GPR— Ground penetrating radar.

AEM— Airborne Electromagnetic: Airborne exploration technology.

RRSG— Remote Radar Sensing Group: Group in the Department of Electrical Engineering at University of Cape Town.

BCD— Binary Coded Decimal format.

PCB— Printed Circuit Board.

NTP— Network Time Protocol.

RISC— Reduced Instruction Set Computer.

RTOS— Real Time Operating System

Chapter 1

Introduction

Radar can be used for almost any subsurface detection problem. It can either be deployed on the surface or in a bore-hole, to survey the immediate surroundings. Some of its applications are in mining geophysics, agriculture, concrete non destructive testing, inspection of tunnels and pipelines and in archeology. The goal of this dissertation is the development of a cable odometer for a subsurface radar. The odometer should be able to determine the position and direction when a radar antenna echoed a signal. Figure 1.1 shows Radar deployed on surface while Figure 1.2 shows the Radar deployed in a bore-hole.

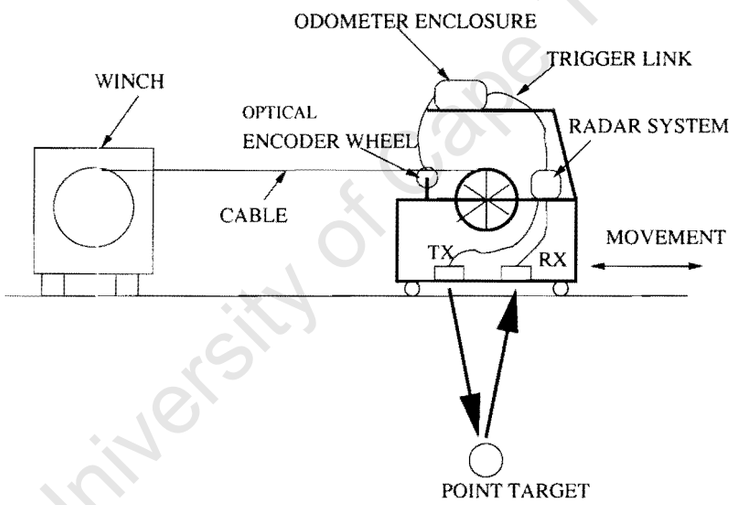

Figure 1.1: Radar deployed on surface

In this dissertation, the distance moved by the Radar antennas is measured from the rotations made by an optical encoder wheel over which the cable passes (as shown in Figure 1.1 and 1.2). The encoder outputs quadrature signals to the Odometer [6]. These signals represent the rotations made by the encoder wheel.

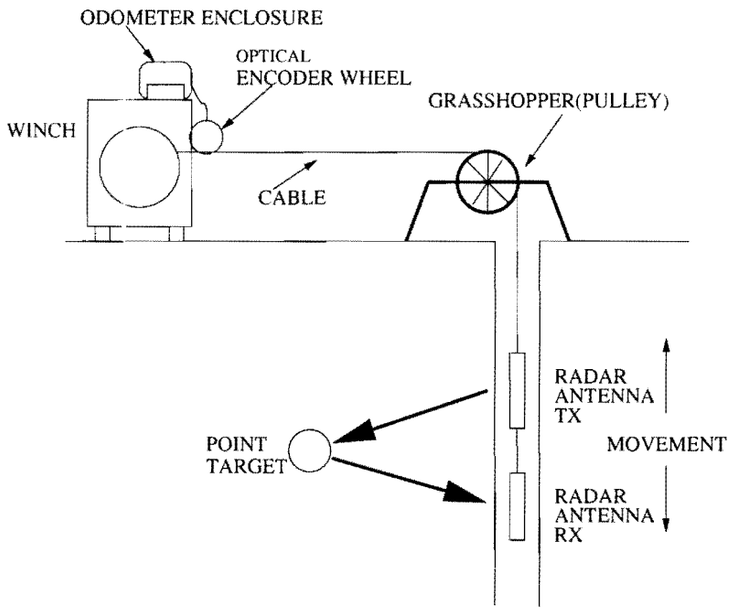

Figure 1.2: Radar deployed in bore-hole

The user requirements determine the modes of operation supported by Odometer and are discussed in detail in Chapter 3. All modes of operation are derived from two main modes discussed below. One of the main modes is when Odometer triggers Radar to make a survey at fixed intervals. The Radar and Odometer are in close proximity in this case (as shown in Figure 1.1). The other mode is when the Odometer and the Radar run autonomously. In the latter mode both sensors take measurements independently from each other. The sensors communicate with other devices via an Ethernet connection (as shown in Figure 1.3). The data collected from the sensors and is related by a time-stamp. The time-stamp source is a Real Time Clock (RTC) on each sensor. The accuracy of all of the clocks is maintained by implementing an Network Time Protocol (NTP) client [RFC 1305] on each sensor. In this project, the NTP client was not implemented on the Odometer and therefore, the RTC on board is set manually by a program. The data that is timestamped is transfered in real time to a remote database or stored on a local storage in cases when there is no network connection.

This project is a module of Geophysical Recording and Processing system that involves sensors (i.e. Odometer, Radar and Orientation Sensors), databases, NTP server, processors and an operator [17]. Figure 1.3 shows the setup of the Geophysical Recording and Processing system. All components communicate via an Ethernet connection.

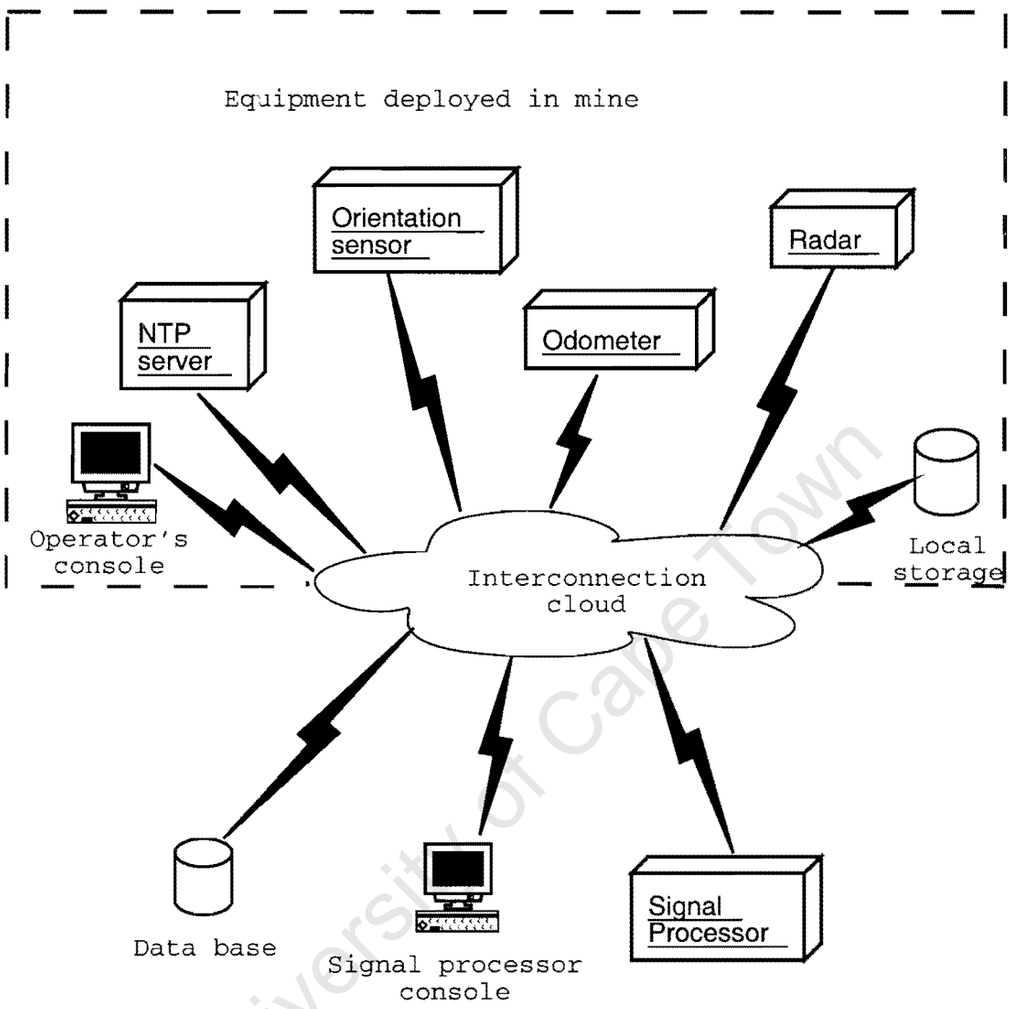

Figure 1.3: Setup of Geophysical Recording and Processing system

1.1 Objectives of Project

The main aim of the project is to design a processor board that can determine the position, direction and speed of radar antennas. A summary of the functionality of the Odometer is stated below:

- The Odometer should calculate the position, direction and speed from the digital output of the optical shaft encoder.
- The Odometer should accommodate all modes of operation.
- The Odometer should be able to operate in harsh environments (e.g. in a mine).

1.2 Procedure taken in completion of project

1.2.1 Concept study

A pre-study was undergone to evaluate the user requirements of the project. The problems encountered during a Kleinzee Trial Borehole Radar survey were taken into consideration when reviewing the user requirements [20].

An investigation into the feasibility of the project was performed. The investigation includes selection of suitable microcontrollers, compatible peripherals and a suitable optical shaft encoder. The optical shaft encoder is a position feedback device that converts real time shaft angle and direction into digital quadrature output. A concept study is made on how to reduce errors that result from the equipment used.

The feasibility of using open source software tools was made. An investigation into developing software that runs independent from an operating system yet meeting the user requirements was made.

1.2.2 System Design

The system block diagram was drawn up including only the main sections of Odometer. The design was based on an embedded web server development board designed by Atmel corporation.

The new design uses an 8 bit AVR Atmega128 microcontroller as compared to the Atmega103 used in the development board. A real time clock is used as on board time source. An Ethernet controller is used to communicate with the rest of the world. Dataflash memory is provided for local storage in cases when there is a network problem. A serial communication interface is also provided. Since there will be a TCP/IP stack on the processor board an external 32K SRAM is included.

1.2.3 Schematic and PCB design

Due to the rate at which technology advances (as in the case of the Atmel Atmega103 which is now being replaced by the Atmega128), it was advisable to obtain the components required before making any final decision on the schematic design.

The PCB design included four layers of which two were power planes. The board was manually routed. An expansion connector was included to enable debugging of the board and future expansions to be made. Most of the components used were surface mounted type so as to conserve board space and keep cost low.

1.2.4 Software implementation

The software used on the Odometer board was ported from the development board. Even though the Odometer board used an Atmega128 processor which is pin compatible with the Atmega103, drastic changes had to be made during software development. The extended functionality of the Atmega128 which is necessary in order to accomplish the user requirements necessitated the changes.

The option of using a free compiler such as an AVR-GCC compiler was considered. Since it is still being developed, some functions (i.e. sprintf and sscanf) which were used in the TCP/IP stacks [RFC 1180], ymodem protocol, FTP [RFC 959] and other protocols have not yet been implemented in the compiler. Due to the complexity of implementing these functions and the time limitation for the project, it was not feasible to implement the functions on AVR-GCC compiler. An Imagecraft compiler [1] was used as an alternative.

1.3 Project limitations

Taking into considerations that it was my first experience of C language, using an operating system such as eCos was beyond the time scope of this project. The eCos would be the ideal operating system for the processor board. It was therefore advisable to use TCP/IP stack that run independently from an operating system. The TCP/IP stack used was ported from the AVR embedded web server reference design. This TCP/IP stack is not a complete stack [5].

Another setback is the limited supply of components that could be used in the project. Most of the components had to be ordered from outside South Africa. It was difficult to obtain most component as companies were reluctant to sell in small quantities. It took months to get some of the components from overseas distributors. For this reason, one had to make sure that the components are available before making any final decisions on the schematic and PCB designs.

1.4 Outline of Dissertation

The dissertation consists of the following chapters outlined below:

Chapter 2 discusses the following issues:

- theoretical background of the project. The factors that motivated the project are discussed here. The Kleinzee Trial Borehole Radar survey is one of the factors. This project is a module of the Geophysical recording and processing system which is also discussed in this chapter.
- states the user requirements of the project. This determines the modes and states that should be supported by the system. A review on how the user requirements can be achieved is explained. The specifications of the project are also stated.
- describes the pre-study that was carried out. The errors that might affect the performance of the system are explained. The feasibility of the project is done taking into consideration the errors that might be involved.

Chapter 3 discusses the hardware design of the project. A thorough explanation of the components used and how they interface to each other is made in this section. The PCB design and the protection of Odometer (when used in harsh environments) is also discussed.

Chapter 4 discusses the software design. Most of the software used has been ported from the Atmel Embedded web server development board. An explanation on how this software interacts with hardware and other programs in order to improve the performance of system is made.

Chapter 5 describes the testing of the prototype board. The results of the test are discussed in this section.

Chapter 6 is the final chapter where conclusions and recommendations for the project are made.

Chapter 2

Concept study

2.1 Theoretical background

For this project to be a success, it is crucial to understand how other equipment that will be used with it operates. Applications of the Odometer can be two projects discussed in the following sections.

2.1.1 Kleinzee Trial Borehole Radar System Survey

On the 22nd March 2001 part of the Remote Radar Sensing Group (RRSG) members from University of Cape Town joined a team from Stellenbosch University to do a Trial Borehole Radar Survey [20]. The survey was conducted at Kleinzee, a diamond-mining region on the northern coast of South Africa. There are different layers of sand and clay found in this region. Between this layers diamond ore may be found. The bedrock that contains diamonds is very small, and its topography is of interest in the analysis of diamond concentrated regions. These regions can be at depths of 40 - 60 metres [20]. If these can be identified, the mining process would be selective and as a result costs and destructions to the environment would be reduced.

There are different methods that can be used such as Microgravity, GPR and AEM to evaluate the bedrock channels. Bore-holes are drilled at 50m grids. At any region that is promising, 25m grids were made. AEM has been the most reliable amongst these methods even though it cannot be used for grids less than 25m. It is speculated that Borehole Radar can be used in such cases[20].

As part of the exploration programme, the Borehole Radar (BHR) system was used. Figure 2.1 shows the diagram of the BHR. Boreholes of different depths were made at approximately 10m apart [20].

Measurements are taken at different heights as the Radar moves up and down within the bore-hole. To achieve this movement a winch, pulley system and grasshopper are used as shown in Figure 2.2. The heights at which the Radar antennas transmit or receive signals are important during the analysis of the data collected. Cable markings of 0.5m spacing

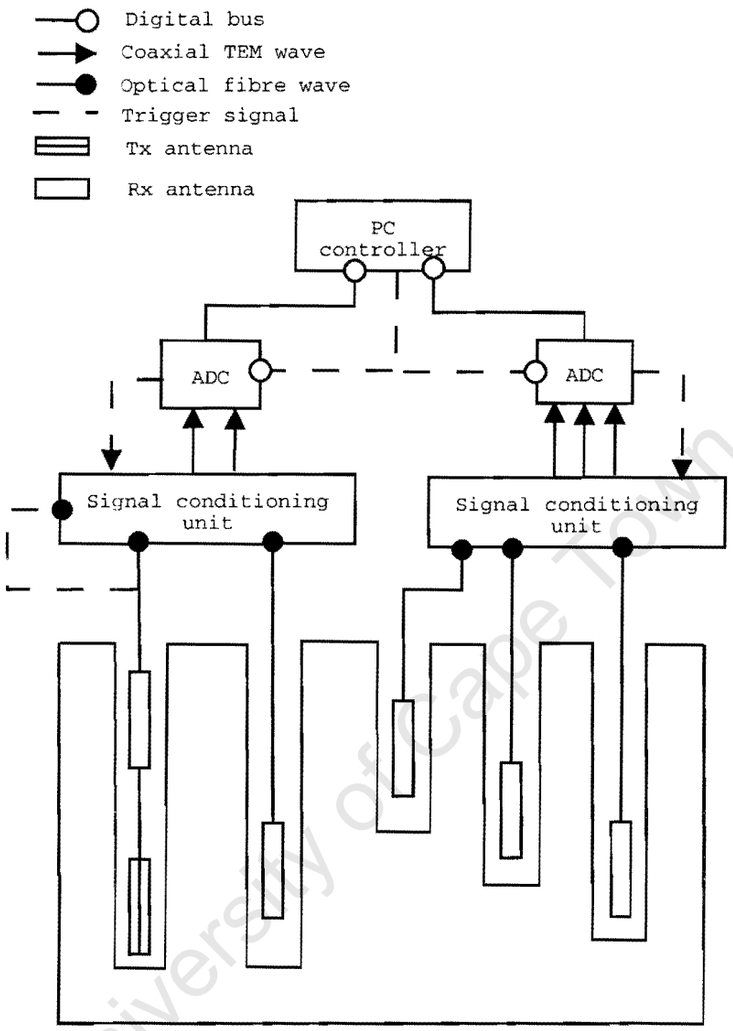

Figure 2.1: Block diagram of BHR (with reference to [20])

were made on the optical fibre cable. These markings were used to determine the depth of the antennas.

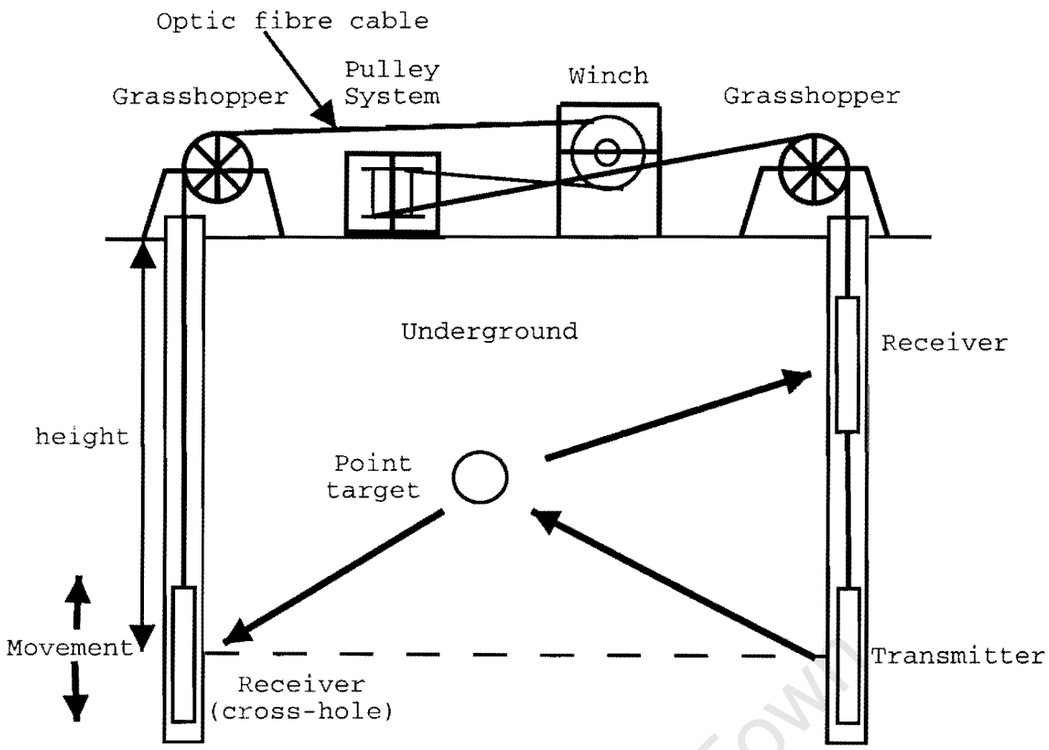

Figure 2.2: Geometry of bore-hole survey (with reference to [20])

The following problems were encountered when using the above-mentioned method:

- the markings on cable became unclear after taking a few readings;
- jittering movements could not be detected;
- as the Radar moves down the bore-hole, before it takes a reading the mark on the cable had to be checked that it coincides with the reference point. Therefore, the method was time consuming;
- the speed at which the antennas in the different boreholes moved up and down could not be monitored.

All these factors affected the accuracy of the whole system. Solving these problems constitute some of the user requirements that will be discuss later.

The Odometer should communicate with other devices included in Geophysical Recording and Processing System. The following section states how the devices involved relate to each other.

2.1.2 Geophysical Recording and Processing System

The system consists of sensors (i.e. Odometer, Orientation Sensor and Radar), databases and processor which all communicate via an interconnection cloud (i.e. Ethernet) [17]. Figure 2.3 shows the block diagram of the components involved. The sensor's data will be related by a time-stamp. Each of the sensors has a Real Time Clock (RTC) that is used to time-stamp the data. A Network Time Protocol (NTP) server, which is accessible to all subsystems via the interconnection cloud, is used to correct and synchronize all of the RTCs. The database is the central storage for all data produced by the sensors. The system console allows full access to all on line sensors and databases [17]. The Signal processor provides processing as requested by system console. All the equipment deployed in the mine can be controlled by an operator's console.

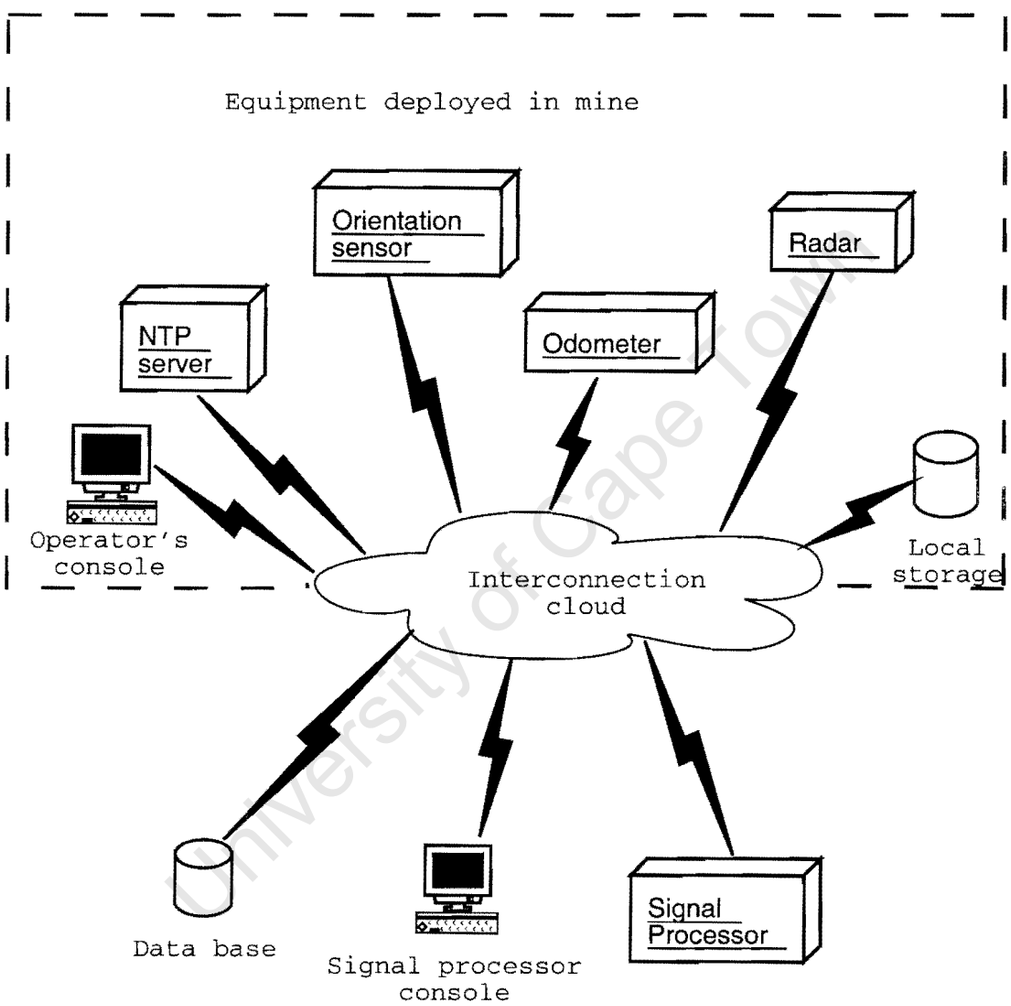

Figure 2.3: Geophysical recording and processing system (with reference to [17])

2.2 User Requirements, Requirements Review and Specification

This section discusses the user requirements that were drawn from the problems encountered at Kleinzee Trial Borehole Radar survey and the requirements of the Geophysical Recording and Processing system. The requirements are reviewed and specifications for the system are made.

2.2.1 User Requirements

The University of Cape Town and University of Stellenbosch have designed a Borehole Radar system that was used in a Trial Borehole Radar survey at the Kleinzee alluvial diamond mine. In order to determine the position of the Borehole Radar as it moves up and down the bore-hole, the optical cable was marked at 0.5m spacing on the cross-hole receiver. The transmitter and receiver pair cable were marked at 1m spacing [20]. The cable markings were sometimes worn off completely by the cable guide on the winch. The procedure used to transfer data from the Psion work-about [20] to the laptop PCs delayed the survey.

The Trial Borehole Radar Survey is an example of an application of this dissertation. Therefore, solving some of the problems experienced during survey is part of the user requirements. The project involves the design of a controller board (called Odometer) that will determine position, direction and speed of the Radar at any time as it move up or down the bore-hole. The system should also be applicable for cases whereby the Radar surveys by moving horizontally above surface. In the latter case the Radar will be in the proximity of the Odometer. Therefore, methods of triggering the radar directly can be used. The data collected should be transfered via a network connection to some remote data storage in real time or store on board in cases where there is no network connection.

Considering that the system can be used in a mine, conditions such as safety, ruggedness, EMI, reliability and transportability should be taken into consideration when designing the system.

2.2.2 Requirements Review

The user requirements are analyzed below. The analysis is divided into two sections, the data collection section and the data processing section.

Data collection section

1. The system must be able to determine position, speed and direction of the Radar at any instant.

2. The distance measured should be accurate to centimetres.

3. The interval of taking readings should be set by user and should be in centimetres.

4. The smallest interval at which the Radar makes a survey is 5cm [11]. Therefore, a system that can achieve a resolution of 1cm is required.

5. Considering the fact that the equipment can be deployed in a mine, a rugged and portable device should be used.

6. An optical shaft encoder that is designed for heavy-duty applications should be used as the system can be used in a mine. The optical shaft encoder should serve as a position feedback that converts real time shaft angle, speed and direction into a digital output signal.

Data processing section

1. There should be an option of selecting the size of the wheel that the optical shaft encoder is attached to.

2. The microcontroller used should have a high throughput such that there are no losses in the data received and transfered via a network connection. There should be enough RAM for a TCP/IP stack to run on.

3. The Ethernet controller used should comply with the network protocols needed to create a network connection (i.e. TCP, IP, UDP, etc).

4. In cases when there is no network connection (i.e. in mines) data storage should be provided on board.

5. The data should be time-stamped by time from a clock source (i.e. RTC) on board that is corrected by another clock source (i.e. NTP server), which is standard for all systems that need a time-stamp.

6. Provide serial communication (i.e. via RS232).

7. System should accommodate these modes of operation:

- Sound of buzzer should alert user to trigger Radar or there should be a Microswitch connection that triggers radar directly.
- There should be an optical fibre connection that is used when triggering Radar directly.
- An LED switching on and off should alert user to trigger Radar.

2.2.3 Specification

1. The optical encoder should operate from a 5V power supply since it is powered by the Odometer.
2. The resolution of the optical shaft encoder should be at least 1cm.
3. The microcontroller should have at least one 16-bit counter used as an interface for the quadrature output signals from optical shaft encoder.
4. The RTC used should have a backup power supply such that it maintains time in the absence of primary power on Odometer.
5. An NTP client should be implemented to correct the clock source on board.
6. A web page should be used as a user interface when selecting modes of operation.
7. An SRAM chip should be included on board as to increase the RAM needed for running a TCP/IP stack.
8. A dataflash memory chip should be provided on board as non-volatile memory storage.

2.3 Literature review

In most cases a GPR or BHR are used to survey the ground. The Radar antenna is either moved horizontally across the surface to be surveyed or up and down a bore-hole (as shown in Figure ??). The position at which results were taken by Radar is very important when making analysis. In this dissertation a processor board is designed to keep track of the position of the Radar at any time. The errors that will affect the accuracy of the system are discussed in this section.

2.3.1 Distance Error

The distance measurement that is obtained will consist of errors that result from the methods and components used to determine distance. The different types of errors are discussed below.

Digitization error

A Radar antenna moving in the x direction surveys the ground at intervals Δxcm. The Odometer records each interval where Radar made a survey. The accuracy of the Odometer can be improved by using a high-resolution optical encoder and therefore, reducing quantization of distance errors.

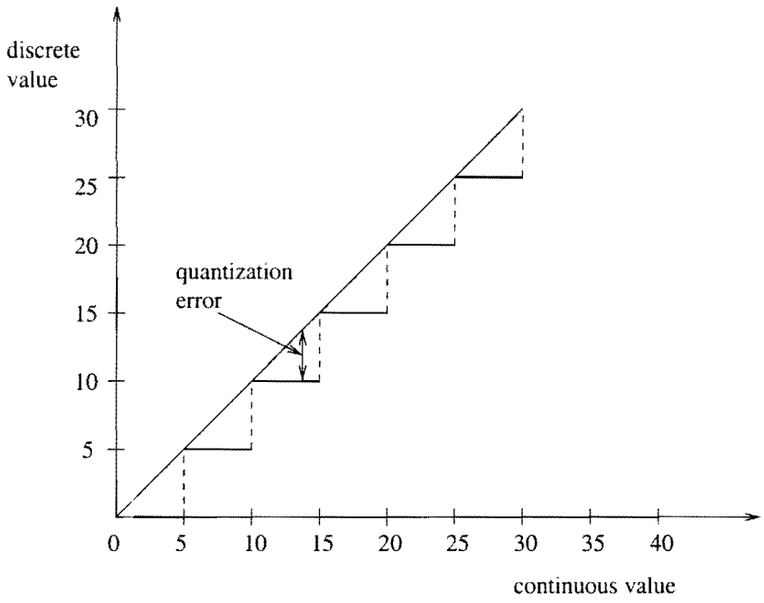

Figure 2.4: Digitization error

Considering Figure 2.4, if a distance resolution of 5cm is used, any survey taken between 5cm and 10cm will be considered to be at 5cm. If the resolution is increased to 1cm, a survey between 10cm and 11cm will be considered to be at 10cm. When considering the worst case, the quantization error for a 1cm and 0.5 cm resolution is 1cm and 0.5 cm respectively.

Initialization error

When the optical shaft encoder is turned on, the slot (that result in a pulse) on the disk of the encoder may sometimes not be aligned with the initial position (as shown in the Figure 2.5). This may result in an error that depends on the resolution of the encoder. As the resolution increases the error decreases.

Time-stamp error

The positive or negative edge of the pulses (representing distance moved) produced by the encoder can sometimes not be synchronous to the time assigned to it (referred to as time-stamp). This result in an inaccurate measure of the time taken between pulses (as shown in Figure 2.6).

When considering the worst case, an RTC with a resolution of 1s has a time-stamp error of 1s. Implementing a software clock with a high resolution on a microcontroller can reduce the error. The software clock resolution is calculated as shown below with the following assumptions [14]:

clock frequency = 16 MHz

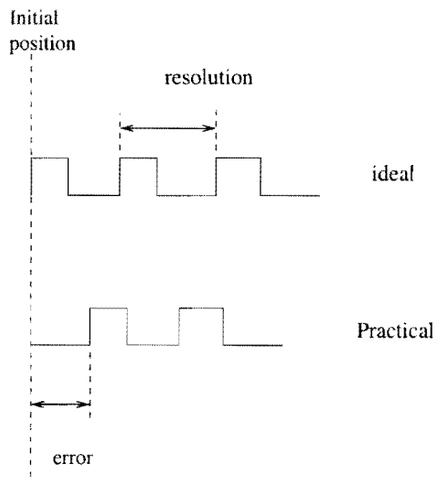

Figure 2.5: actual position of slot on disk of encoder

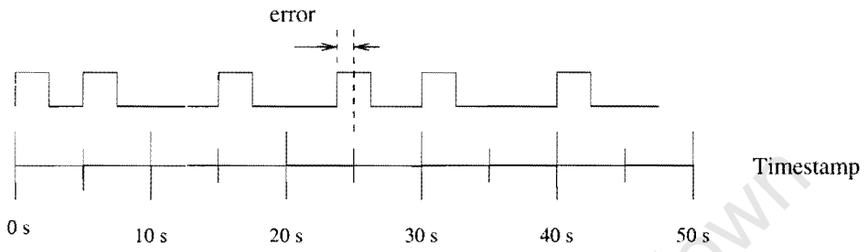

Figure 2.6: time-stamp error

8-bit timer

timer interrupts rate t:

$$\begin{aligned} t &= \frac{16 \times 10^6}{2^8} \\ &= 62500 ticks/s \end{aligned} \tag{2.1}$$

resolution r:

$$\begin{aligned} r &= \frac{1}{62500} \\ &= 16us \end{aligned} \tag{2.2}$$

Therefore, the time-stamp error will be 16 us when considering the worst case.

Other errors

The mechanical setup of the system can result in the following errors. The slippage between the shaft wheel and the cable tied to the Radar, the stretching of the cable due to the weight of the Radar can result in an inaccurate measurement.

The local clock has to be synchronized to other more accurate clocks on the Internet using NTP. This is done to correct the drift error that depends on the type of oscillator used and environmental factors (i.e. temperature) [22].

Another error that results is the synchronization error due to the delay when synchronizing a local clock to an NTP server.

2.3.2 Distance and Time Resolution

In order to be able track the movement of the Borehole Radar as it moves up or down the bore-hole, the time sampling interval should be as small as possible. In Figure 2.7, between 5 - 25 ms the Radar is expected to move smoothly without jittering through the bore-hole. The dotted line represents a smooth movement. But, due to the irregularities in the bore-hole, the Radar can follow the trace shown by the solid line. The latter can only be noticed if the resolutions of both distance and time are as small as possible. Distance resolution depends on the settings on the optical shaft encoder and most encoders can achieve a resolution of fractions of millimetres when a wheel attached to the shaft encoder has a diameter of less than 20 cm (as shown in Figure 2.9).

Time resolution depends on the clock used. Most RTC has a resolution of 1 second. To improve this resolution a software clock that can attain resolution of microseconds can be implemented.

The data from the shaft encoder needs to be timestamped in order to determine time taken and speed. The test made by the team involved in the Kleinzee survey showed that the radar could move at maximum speed of 0.25 m/s (25 cm/s) up and down a bore-hole. For a distance resolution of 0.5 cm, time resolution should be at least 0.5/25 seconds (20 milliseconds) in order to time-stamp each pulse produced by the encoder. Most RTCs have a resolution of 1 second. This implies that a software clock with a resolution less than 20 ms will have to be implemented.

The software clock can achieve time resolution of less than 20 ms but will be limited by NTP server resolution on LAN, which are a few milliseconds. NTP server is used to synchronize the clock with other accurate clocks on the Internet. An NTP secondary (stratum-2) time server can achieve an accuracy of 3 ms when installed on distant LAN [22]. In the reconstruction of the image formed by the Radar survey, tolerance of up to 10 cm [11] in distance error is acceptable. This implies that the distance resolution of 1 cm can be used. The time resolution should be less than 40 ms in this case and considering that the NTP server has an accuracy of 3 ms, a time resolution of 10ms will be used. The above analysis applies for the case when an NTP client is implement on the Odometer.

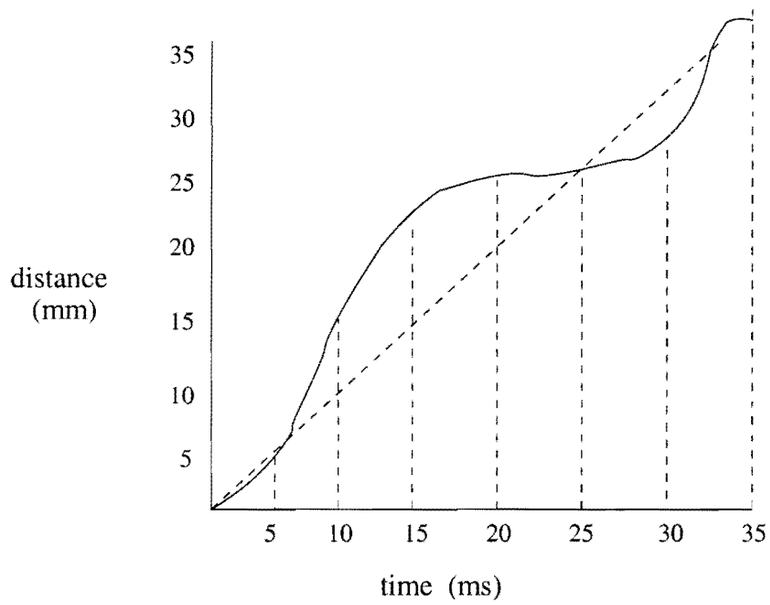

Figure 2.7: Movement of radar

2.3.3 Shaft encoder

The number of revolutions that the shaft make is used to calculate the position of anything tied to the cable that rolls around the shaft wheel. Changing the diameter of the shaft wheel can vary the resolution of the position measurement. As shown in Figure 2.8 and 2.9 when the diameter increases the resolution decreases. Another factor that affects resolution is the number of electrical cycles per revolution (CPR). As the number of CPR increase the resolution increases.

Choosing the shaft encoder

The following factors were considered when choosing a shaft encoder:

- resolution
- cost
- robustness

Table 2.1 shows different types of encoders from different companies.

The shaft encoder designed by USDigital Corporation satisfied most of the above-mentioned features. Other companies that design incremental shaft encoders are Dynamics Research Corporation (DRC), Hohner Corporation, BEI Duncan Electronics Division.

Figure 2.8 shows the number of CPR as 100 while Figure 2.9 shows the number of CPR as 2048. This is the range for the HD25 optical encoder. When using HD25, a resolution

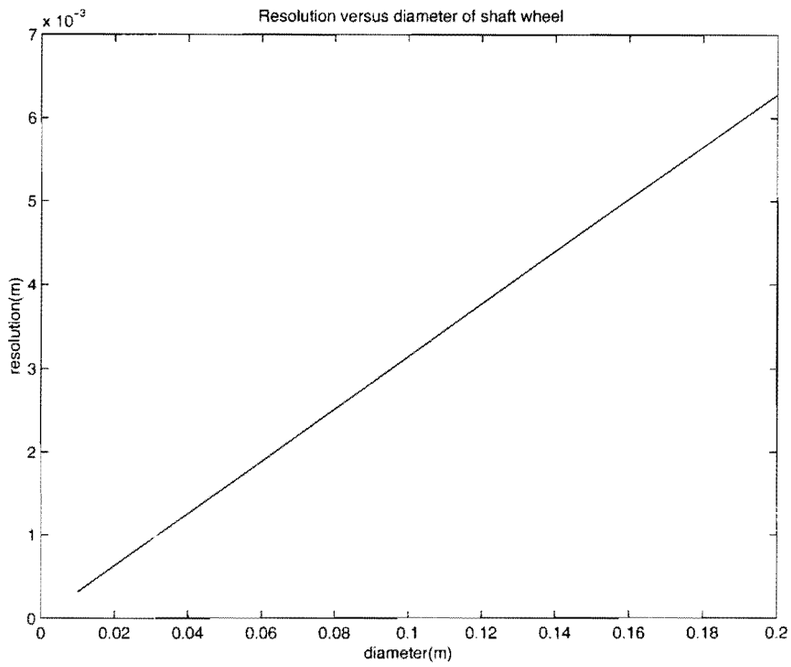

Figure 2.8: Resolution when CPR is 100

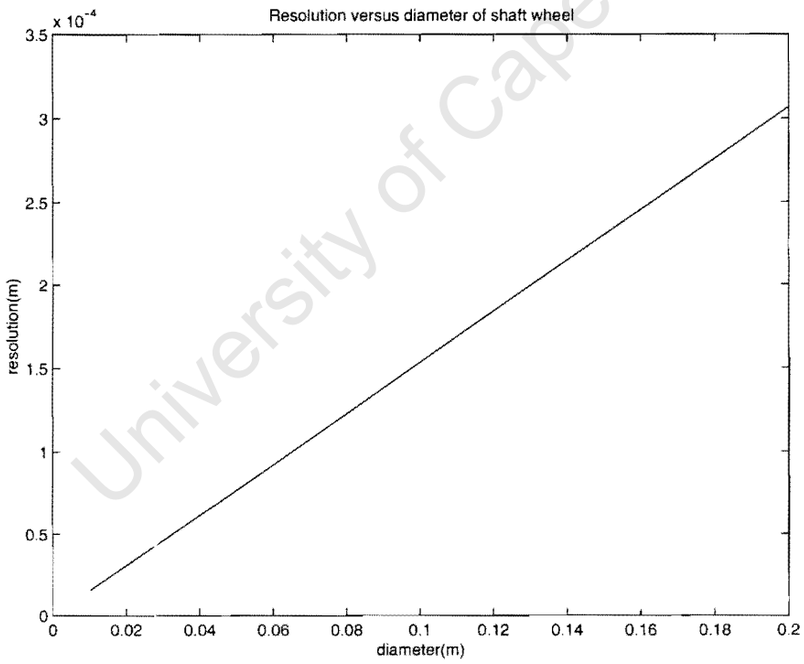

Figure 2.9: Resolution when CPR is 2048

Table 2.1: Incremental Encoder

Features	US digital HD25	D.R.C. Model 25D	Hohner series 02
CPR (max)	2048	5000	
PPR(max)	8192	20,000	1024
Supply voltage	+5, 9.5 -32V	+5,12,15V	+5 - 24V
Price/unit	$280	$425	$321
Shaft sealed	yes	yes	no
Operating temp	-40C- 100C	0C- 70C	-20-100C
Housing and cover material	anodized aluminum	die cast aluminum	die cast zinc alloy

of 5 mm can be obtained even if the diameter of the shaft wheel is increased to about 30 cm. The radius of curvature of the optic fibre cable used is 15 cm. This implies that for the cable to be rolled around the whole circumference of the wheel (i.e. to reduce slippage), the diameter of wheel should not be less than 30 cm. A resolution of 5 mm can still be achieved considering that for 2048 CPR at 20 cm the resolution is 0.3 mm as shown in Figure 2.9.

The housing and cover material is made of anodized aluminum to protect it against shock. The encoder is also sealed to protect it against fluids. It is the cheapest as compared to the two shaft encoders shown in Table 2.1.

2.3.4 Investigating a suitable Microcontroller and peripherals

There are a number of microcontrollers and peripherals that can be used. The section below discusses the microcontroller that is suitable for this project.

Atmel microcontroller family

Atmel products are easily available. An Atmel ARM Thumb microcontrollers would be suitable if a Real Time Operating System (RTOS) is to be implemented. A suitable RTOS would be eCos since there are ports to some of the ARM microcontrollers. Due to the time constraints, designing a PCB and implementing eCos was not feasible for someone who is still learning C language. A better option was to use an Atmel 8-bit AVR microcontroller since there is a lot of support in software development (i.e. AVR mailing list for support). A TCP/IP protocol suite that runs independently from an operating system can also be implemented on an AVR microcontroller [5]. The Atmega128 is suitable for this application. It is a CMOS 8-bit microcontroller based on the AVR RISC architecture. It has a 128K bytes ISP flash, 4K bytes EEPROM, 4K bytes SRAM, real time clock (RTC), four timers/counters and other features.

Memory

The system had to provide temporary storage and non-volatile data storage. The types of memory used are discussed below.

In cases when there is a problem with the network and data cannot be transfered to the data storage, dataflash memory will be provided.

The size of the dataflash memory that will be used to store data is 2 Mbits. The calculations below show how the size was determined. For 60 minutes of continuous data processing, 1.15 Mbits would be required. Since the web page and files that contains MAC and IP address are also stored in dataflash, 2 Mbits storage would be suitable.

size of position data (unsigned long) = 4 bytes
size of timestamp (unsigned long) = 4 bytes
size of timestamped data per radar reading = 8 bytes/reading
minimum interval of taking readings = 5cm
maximum speed of radar = 25 cm/s
rate of taking readings = 5 readings/s
timestamped data stored per second = 5 readings/s * 8 bytes/reading
= 40 bytes/s
no. of bytes in an hour = 40 bytes/s * 3600s
= 144 Kbytes

The dataflash will be used to store frequently changing data. Therefore, small sectored flash will be desirable. An Atmel AT45D021B dataflash was used.

SRAM will be used for temporary storage of data, stacks, and program workspace. A capacity of 32K bytes is sufficient. An IDT71256S CMOS SRAM can be used. It has a 32K * 8 memory organization.

Ethernet controller

When considering the mode of operation of system data should be time-stamped and transfered via an Ethernet connection to some data storage on another location. To achieve this transfer an Ethernet controller is interfaced to the microcontroller.

There are two models of Ethernet controller that were considered. These are CS8900A and a LAN91C111. Both these controllers can achieve the transfer. They have both been used in the RRSG group (which I am member of) and are readily available. CS8900A has been used on the development board that will be used for software development, therefore it is the suitable Ethernet controller in this case. The network interface is a 10base-T physical layer (PHY). It uses a 20 MHz reference clock for both PHY and media access control (MAC).

2.3.5 Real time clock

In order to relate the data obtain from the shaft encoder to some data obtained on a remote sensor (i.e. the radar system), the data is time-stamped. This data will be time-stamped by time obtained from a software clock on board. An on board clock is used during bootup as the NTP server compares the time on board to its time which is much more accurate. The Atmega128's RTC can be implemented. The disadvantage of this RTC is that when the Atmega128 is not powered it looses time. These will happen several times considering the cases when the board and other equipment are transfered into a mine. A solution to this problem would be to use a DS1286 Dallas Semiconductor chip that can be used to provide time to the Atmega128 [2]. It consists of an embedded lithium energy source and a 32.768 KHz crystal and drift by less than 1 minute per month at 25 degree celsius. The embedded lithium energy maintains time for over 10 years in the absence of external power.

2.3.6 Network Time

The geographical locations of the sensors, signal processors and data storages (i.e. shown in Figure 2.3) indicate that these devices may sometimes not be close to each other. To relate the information collected from all this devices an accurate time-stamp is appended to the data collected. When the time-stamp source (i.e. an RTC on odometer) is accurate, the data collected can be integrated precisely with data from other sensors. To achieve this, the time-stamp source must be synchronized to the other time-stamp sources.

A method that can be used is to distribute the time whilst maintaining local time source on each device [18]. This can be achieved by requesting time from a server (i.e. Network Time Server) and using it to correct the local time source (i.e. RTC) taking into considerations the delays involved. Network Time Protocol (NTP) uses this principle. NTP is implemented on LAN therefore, it is subjected to problems that data using network encounter. Factors such as packet collision and random delay due to network traffic result in delaying of time. Apart from the delay due to transferring a time-stamp from one source to another, there are some effects related to the oscillator of the time source. Factors known to affect the oscillator are physical vibrations, crystal aging, fluctuations in temperature and power supplied [18].

The time of any time source at time t can be represented as

$$T(t) = T(t_0) + Rt(t - t_0) + \frac{D}{2}t^2(t - t_0) + x(t) \tag{2.3}$$

where T is time offset at time t_0, R is the frequency offset, D is the drift due to aging and x is jittering effect [18]. T(t) includes the effects of propagation delay and x(t) includes the uncertainty due to jitter in the network and measurement process.

There are other protocols such as Probabilistic Clock Synchronization (PCS) and Digital

Time Service that can be used to correct the timing errors illustrated in the equation above [18]. Network Time Protocol has an edge over these two protocols, as it is able to correct most of the timing errors in equation above [18].

NTP operation

The NTP client resides on the Odometer and is used to synchronize the time of the RTC on Odometer to the time of an NTP server which could be on a Global Positioning Service (GPS) receiver. The NTP program measures round-trip delay jitter and oscillator frequency offset and then adjust the RTC. The RTC is adjusted in small steps in order to keep the timescale continuous. NTP client exchanges information with one or more servers at certain poll intervals. Under normal operations the poll interval is between 64s and 1024s [19].

2.4 Conclusions

The radius of curvature of the optical fibre cable is 15cm. A HD25 encoder with 2048 CPR has a resolution of 0.05cm when attached to a wheel of diameter 30cm. This is more than the required 1cm.

DS1286 RTC has a resolution of 10ms. Therefore, a software clock will not be implemented since the RTC achieves the required resolution for timestamping each pulse produced by the encoder.

Most of the distance errors depend on the distance resolution. A higher resolution will reduce the distance error. The error due to slippage was not quantified in this project.

Chapter 3

Hardware design

This chapter explains the functionality of the main devices. Details of how the devices communicate with each other are discussed. The PCB design is also discussed.

3.1 Encoder hardware

An incremental shaft encoder of type HD25 (which is an industrial rugged metal optical encoder) is used in this project. A HD25 encoder's CPR ranges from 100 to 2048 [6].

Output signals of shaft encoder

HD25 encoder's output signals options are either two quadrature signals (CH A and CH B), clock and direction signals (Clock and Up/Dn) or up and down clock signals (UpClk and DnClk) as shown in Figure 3.1. In this case the option of clock and direction as output signals was selected.

Encoder to Microcontroller interface

The HD25 signals cannot be interpreted by the Atmega128, therefore some logic gate circuitry is introduced as an interface between the encoder and microcontroller. The logic interface outputs signals T1 and IC1 (shown in Figure 3.2) are inputs to a microcontroller. The pulses on T1 in Figure 3.2 represent the distance moved while that on IC1 represent a change in direction. The logic gate circuit diagram is included in Appendix B.

3.2 Processor hardware

Figure 3.3 shows a block diagram of the main components. A detail discussion of the components functionality and how they interconnect is made in the following section.

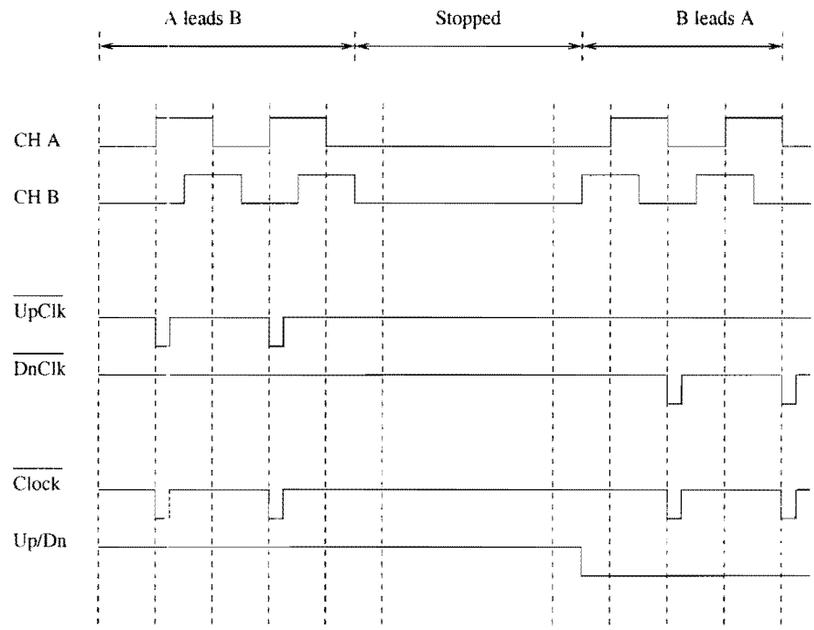

Figure 3.1: Timing diagram for the encoder to microcontroller interface

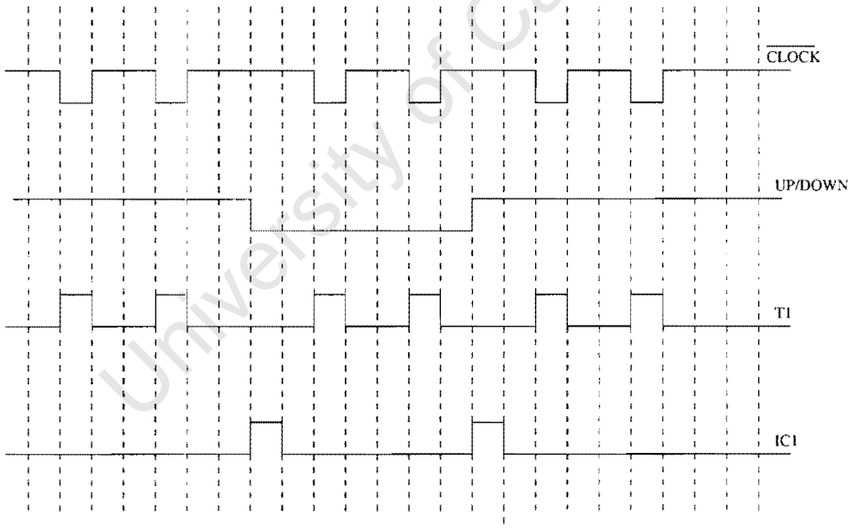

Figure 3.2: Encoder signals converted to input signals for Atmega128 counter

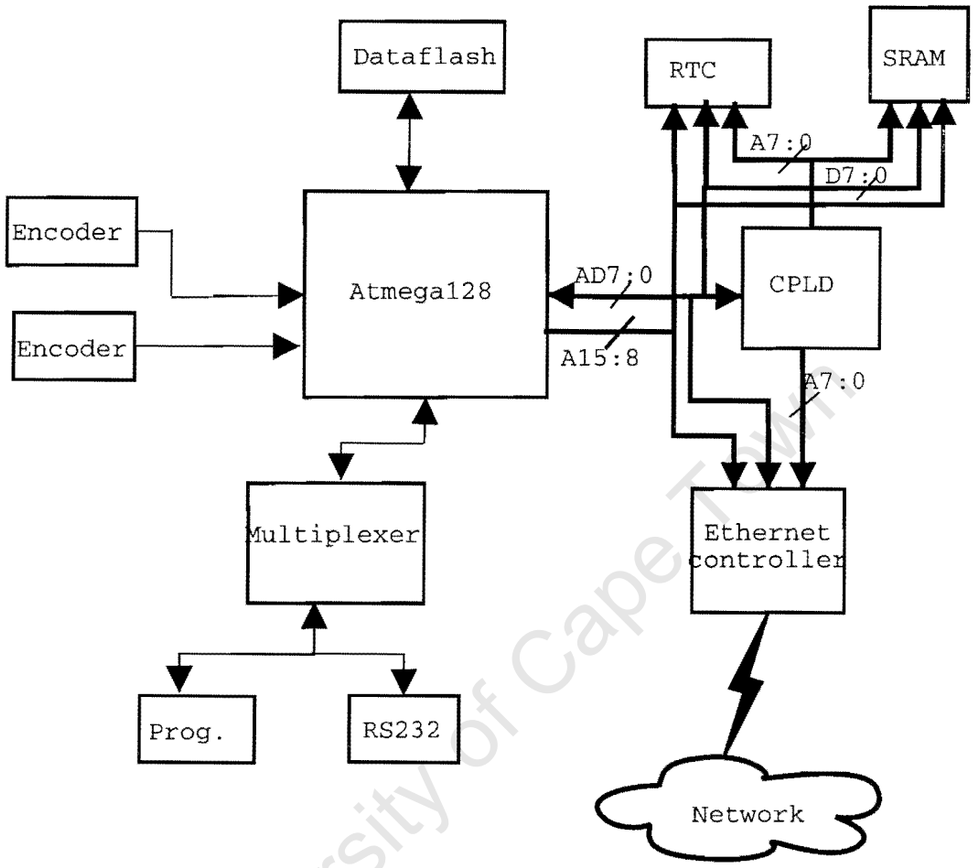

Figure 3.3: Block diagram of Odometer

3.2.1 Microcontroller

The Atmel Atmega128 is a high performance, low power 8-bit AVR microcontroller. It has an advanced RISC architecture with up to 16 MIPS throughput at 16 MHz [10]. It also has up to 64K bytes of optional external memory space. There are two 8-bit timers/counters, two 16-bit timers/counters with separate prescaler, compare mode and input capture mode [10]. Some of its peripheral features that will be used in this project are the serial USARTs, master/slave SPI serial interface and byte-oriented two-wire serial interface[10].

Functionality of the Atmega128

The Atmega128 obtains two signals T1 and IC1 from encoder. T1 is an external clock source for the Atmega128 16-bit counter. The value of the counter is stored in the TCNT1 as shown in Figure 3.4. TCNT1 can be accessed by the Atmega128 CPU for further processing (i.e. determining position of the Radar) [10]. The IC1 signal is applied to the Input capture pin (ICP) of the counter. Figure 3.5 shows the input capture unit. When there is an IC1 pulse on the ICP1 pin a capture is triggered. The 16-bit counter value in TCNT1 is written to ICR1 register. An input capture interrupt is generated at this stage. The interrupt routine is used to time-stamp the counter value. Hence, the position and time when there was a change in direction will be known.

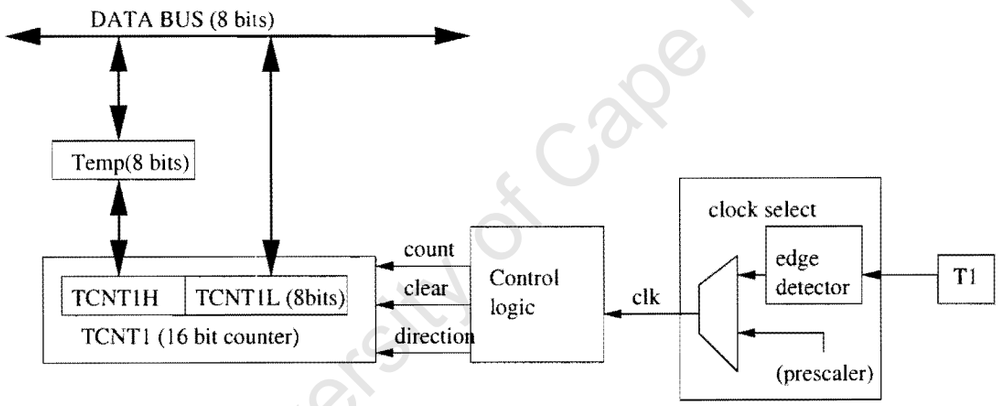

Figure 3.4: Counter unit (with reference to [10])

3.2.2 CPLD

The external memory interface of an Atmega128 is designed in compliance with the 74HC series latch [10]. When the Atmega128 operates at a frequency above 8 MHz @ 4V or 4 MHz @ 2.7V the 74HC latch cannot be used [10]. As an alternative, an Altera EPM 7064S 44-pin PLCC CPLD device is used. This CPLD complies with the timing parameters for a suitable address latch for the microcontroller. It has a total of 36 I/O pins of which

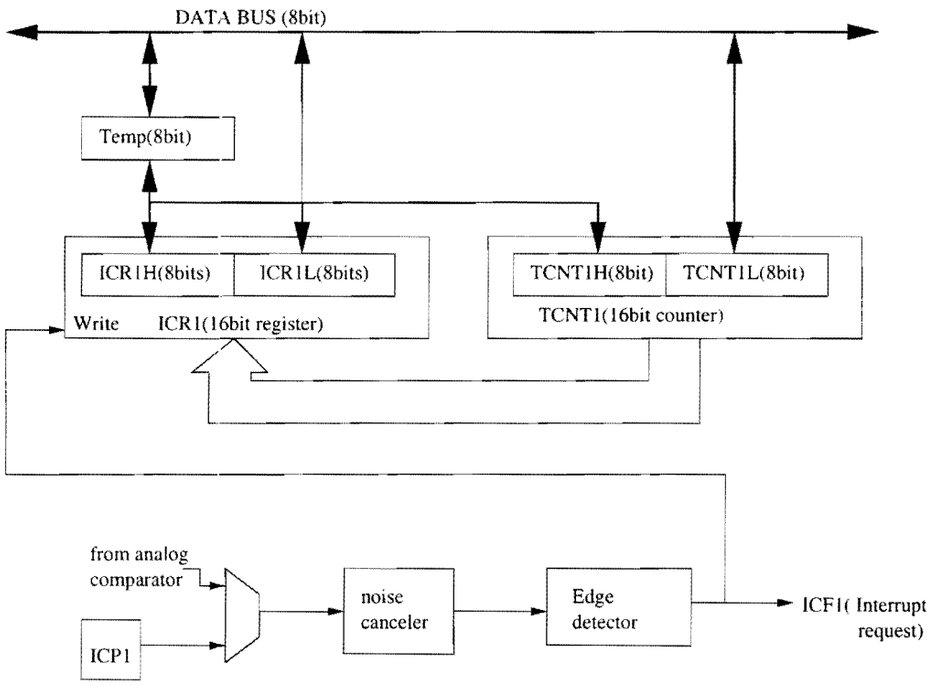

Figure 3.5: Input capture unit (with reference to [10])

four are dedicated for JTAG interfacing [7]. The JTAG interface is used for in-system programming of the device.

In this project, the SRAM is memory mapped to the Atmega128 by the CPLD as shown in Figure 3.7 [10]. The RTC is also memory mapped to the Atmega128 from base address 0xA000 by the CPLD.

CS8900A operates in I/O and memory mode [4]. The decode logic in the CPLD maps the upper memory space of Atmega128 to the CS8900A. Address lines A15, A14, A13 and signals nRD and nWR from the Atmega128 are used for decoding the I/O and memory mode. A15 and A14 select the base address for the two modes as follows:

Mode	A15	A14	Base
I/O	0	1	0x8000
memory	1	1	0xC000

The default values of the I/O base and memory base addresses are 0x0300 and 0x0000 respectively [4]. In this project, memory base address 0x0000 is changed to 0x1000. For the two modes, the base address are actually accessed in Atmega128 memory space at the following addresses.

Mode	Base address decoded by CPLD	Offset	Actual base address in Atmega128
I/O	0x8000	0x0300	0x8300
memory	0xC000	0x1000	0xD000

3.2.3 Ethernet controller

CS8900A is a full duplex Ethernet controller consisting of all analog and digital circuitry needed for communicating via Ethernet. It is configured for a 10Base-T interface. The built-in transceiver includes filters that eliminate the need for external filters or filter/transformer hybrid. Since the CS8900A is supplied with 5V, the ratio between primary and secondary windings for the isolation transformer is 1:1 on the receiver side and 1:1.414 on the transmit line [4].

The CS8900A is suitable for 16-bit data transfer, operating in either I/O space, memory space or DMA slave. It can also be used in 8-bit mode. The 8-bit mode will be used in this project. Since the DMA engine only uses 16-bit memory access, it is not support in 8-bit mode [13]. Interrupts are also not supported in 8-bit mode [13].

In this project, CS8900A will be used in I/O and memory mode. The default mode of operation of the CS8900A is I/O mode. Memory mode is enabled by configuring CS8900A through the I/O registers during initialization.

The Ethernet controller has 4K bytes of internal RAM (known as PacketPage memory) for its internal registers and for buffering the received and transmitted frames [4]. Its architecture uses a method called PacketPage to access its internal registers or for buffering the frames. The internal registers are accessed directly using memory mode or indirectly using I/O mode.

I/O mode

The PacketPage memory is accessed through eight 16-bit I/O ports that are mapped to 16 I/O addresses from 0x8300 - 0x830F in the Atmega128 I/O space [4]. For an I/O read or write operation, the AEN pin must be low, the I/O address of the Atmega128 must match the I/O address space of the CS8900A and either nIOW or nIOR must be low.

Memory mode

When the CS8900A is configured for memory mode, the PacketPage memory is mapped directly into 4K bytes of the Atmega128 memory space [4]. Since 4K bytes of address space is needed, only lines SA0 - SA12 are used. Memory mode operations are mapped into addresses 0xD000 - 0xDFFF. Memory mode access is achieved when AEN pin is low, the address of the Atmega128 matches the memory address space of the CS8900A and either nMEMR or nMEMW pin is low [4].

3.2.4 Memory

External SRAM

The microcontroller will be performing a lot of applications and to compliment this, an IDT71256 SRAM is used as an external memory. The IDT71256 is a high speed static RAM organized as 32K x 8. It can accomplish an address access time of up to 20 ns [3].

The external SRAM is connected to the Atmega128 as shown in Figure 3.6. It is mapped above the internal SRAM as shown in Figure 3.7.

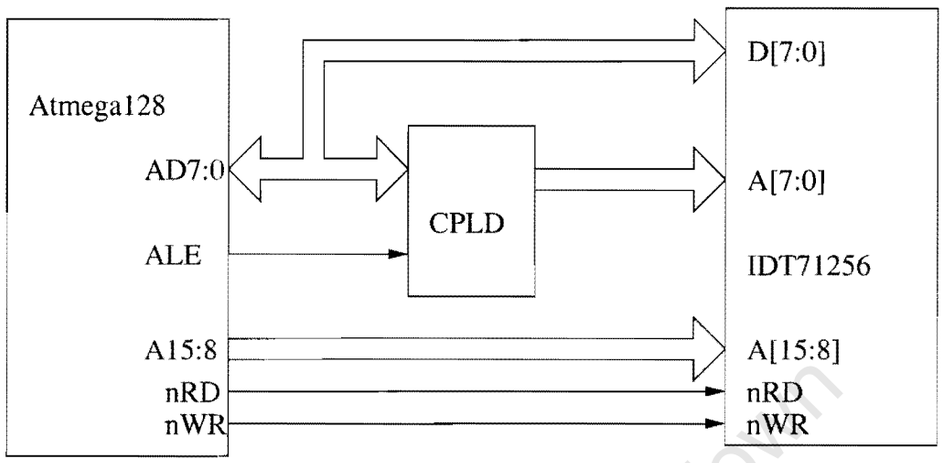

Figure 3.6: Atmega128 interface with SRAM

Dataflash

An AT45DB021B is an Atmel 2 megabits Dataflash memory [9]. The memory is organized as 1024 pages of 264 bytes each. It has two SRAM data buffers of 264 bytes each which allow data to be received while the main flash memory is being reprogrammed.

The Dataflash is used to store configuration files (i.e. files that contain MAC and IP address), web-pages and time-stamped data. The web-page is used when remotely communicating with the Odometer via the Ethernet.

3.2.5 RTC

The DS1286 watchdog timekeeper is a real time clock, alarm, watchdog timer and interval timer in 28-pin JEDEC DIP package [2]. The 64 eight bit registers can be accessed in the same manner as a byte-wide SRAM. The lithium energy source of the timekeeper enables it to retain real time for 10 years in the absence of external power supply. Information that can be obtained include hundredths of seconds, seconds, minutes, hours, day, month and year.

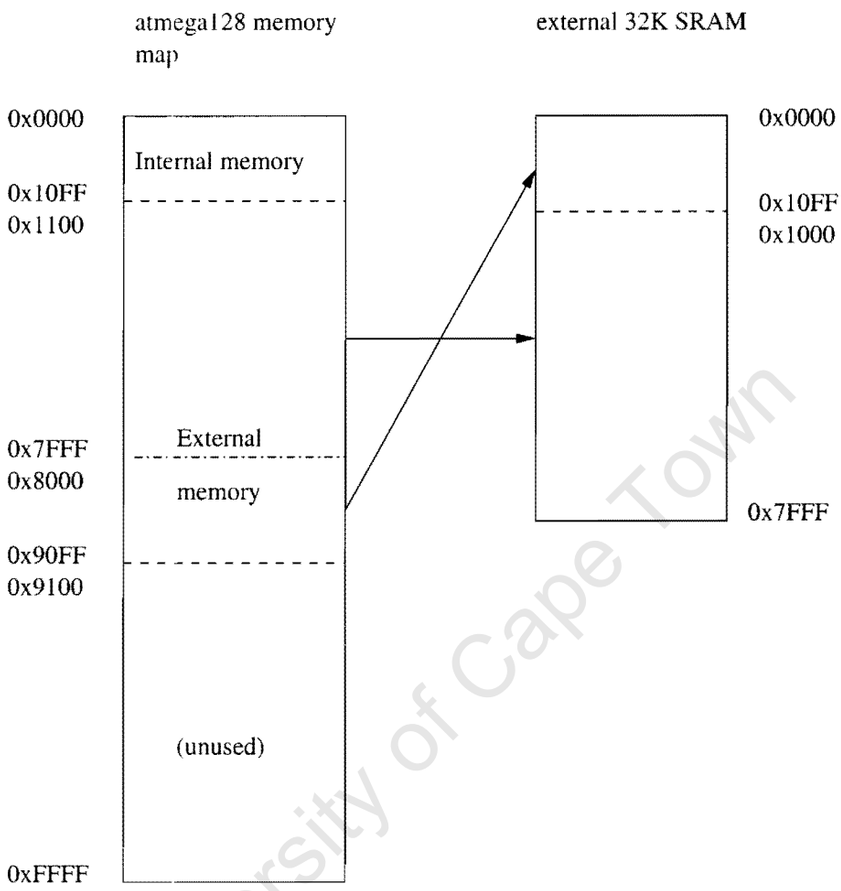

Figure 3.7: Atmega128 memory map(with reference to [10])

Figure 3.8 shows some of the 64 registers in the DS1286 [2]. The time of the day information is stored in BCD format. This registers are directly accessed by the Atmega128 microcontroller since they have been memory mapped to it at base address 0xA000.

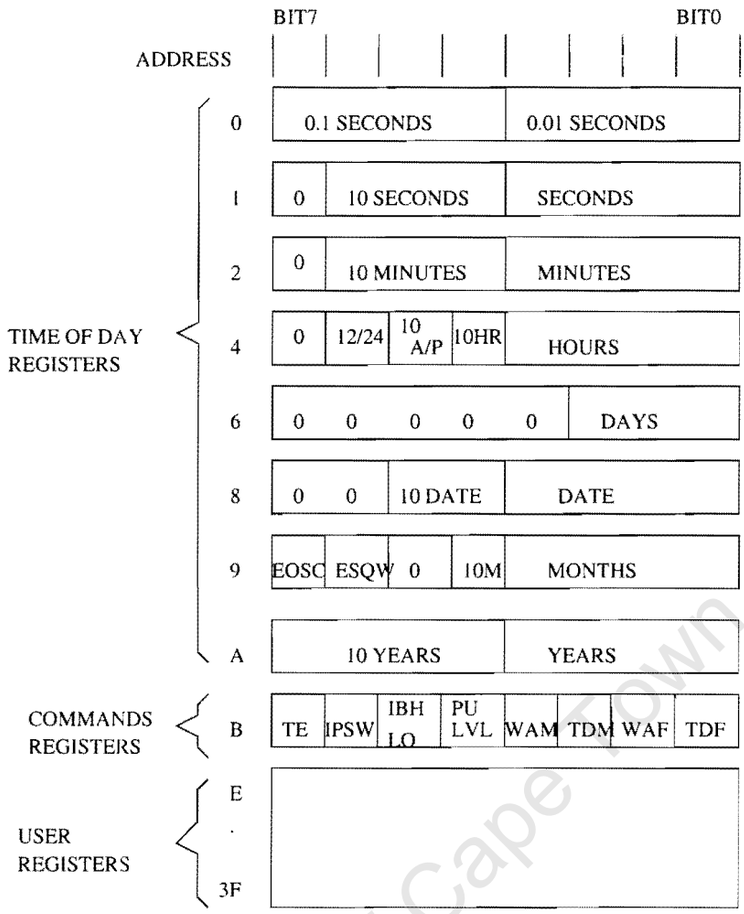

Figure 3.8: DS1286 registers

The minimum read and write access time of DS1286 is 150 ns. This implies that Atmega128 access time to DS1286 should be at least 150 ns or greater.

3.2.6 Serial Interface

The Atmega128 has two USARTs, USART0 and USART1[10]. In this project only USART0 will be used for serial communication. Asynchronous mode of operation is used and therefore only two signals (receive and transmit signals on PE0 pin and PE1pin respectively) are used for serial communication. Both this signal together with a clock signal on PB1 pin are alternatively used for serially programming the Atmega128. To accomplish both tasks, a multiplexer is placed between the Atmega128, in-system programmer and RS232 transceiver as shown in Figure 3.9.

For the initial configuration of the Odometer, the web-page and file that contains the MAC

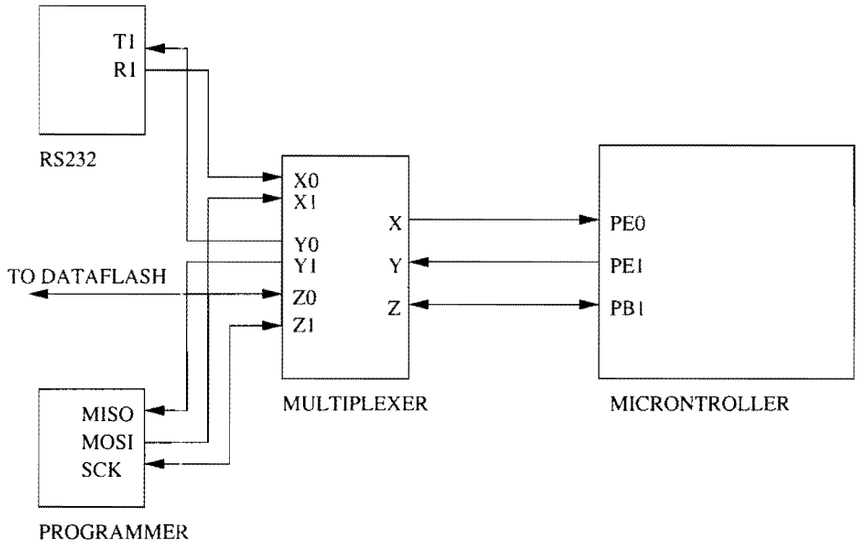

Figure 3.9: Serial interface

and IP address of the CS8900A are transfered to Dataflash via the RS232 transceiver. The protocol that is used to transfer the files is y-modem protocol [5]. Any x-terminal that supports this protocol can be used.

3.3 PCB design

The PCB design was done using the Protel Design Explorer 99. The board is 4 layered and two of the layers are power planes. Most components are surface mounted (SMT) packages. The schematics and PCB layouts are shown in Appendix B.

3.3.1 Routing Nets

The PCB board was manually routed as to reduce any loops that might result in Electro-Magnetic Interference (EMI). The address and data bus nets were aligned together making it easier to trace the signals between components. The track size for the power nets from the power connector up to the regulator is 40 mils. The other nets track size were 8 mils and 10 mils. The vias pad size was 25 mils while the hole size was 15 mils. Figure 3.10 and 3.11 shows top and bottom layers of an unpopulated board.

3.3.2 Component Placement

Most components were placed on the top layer. Some decoupling capacitors were placed on the bottom layer. This was to ensure that the capacitors are as close as possible to the components they are decoupling. The microcontroller was placed towards the middle of the board with Ethernet controller, SRAM, RTC and CPLD close by as they shared the

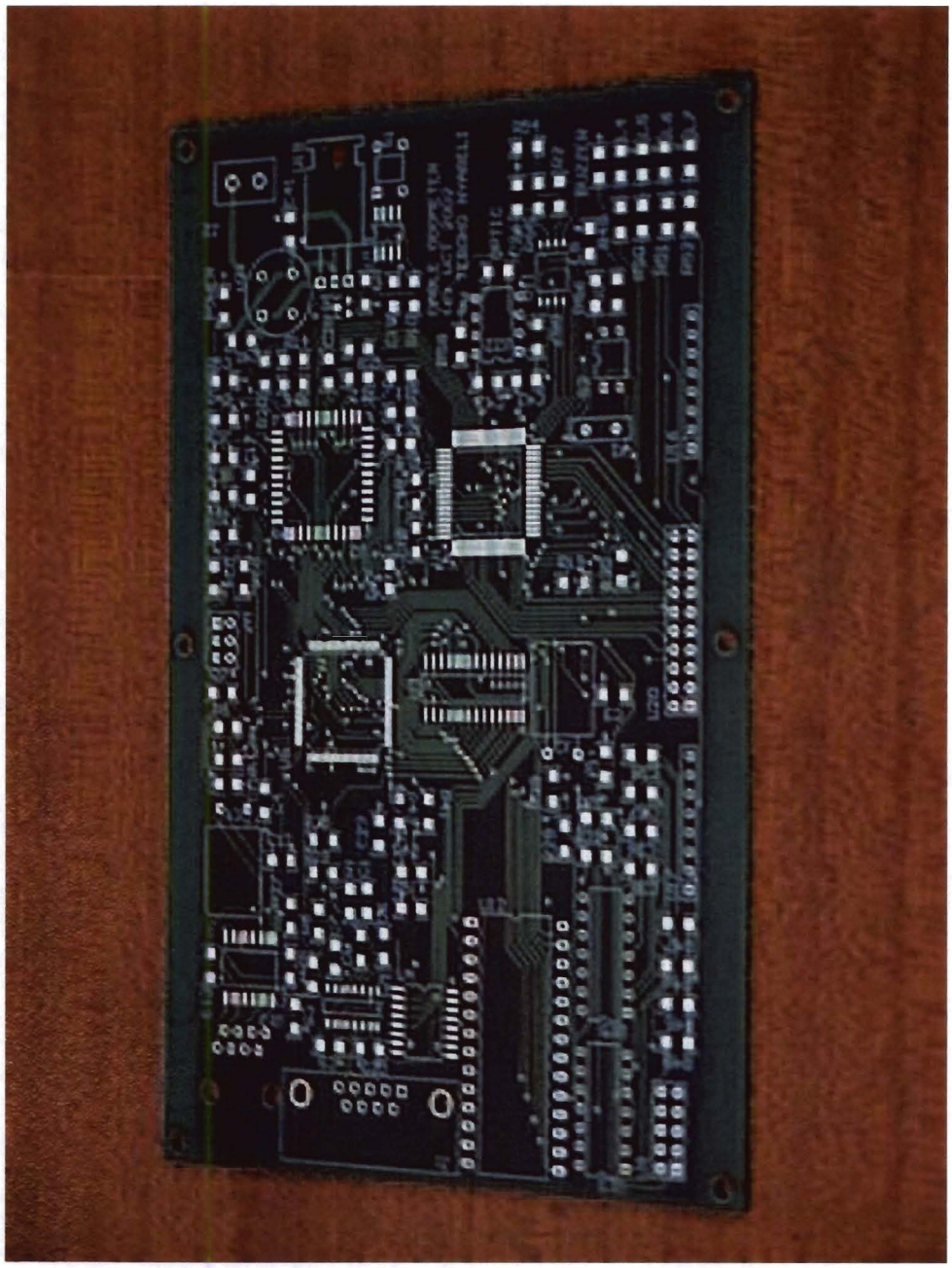

Figure 3.10: Unpopulated PCB (top layer)

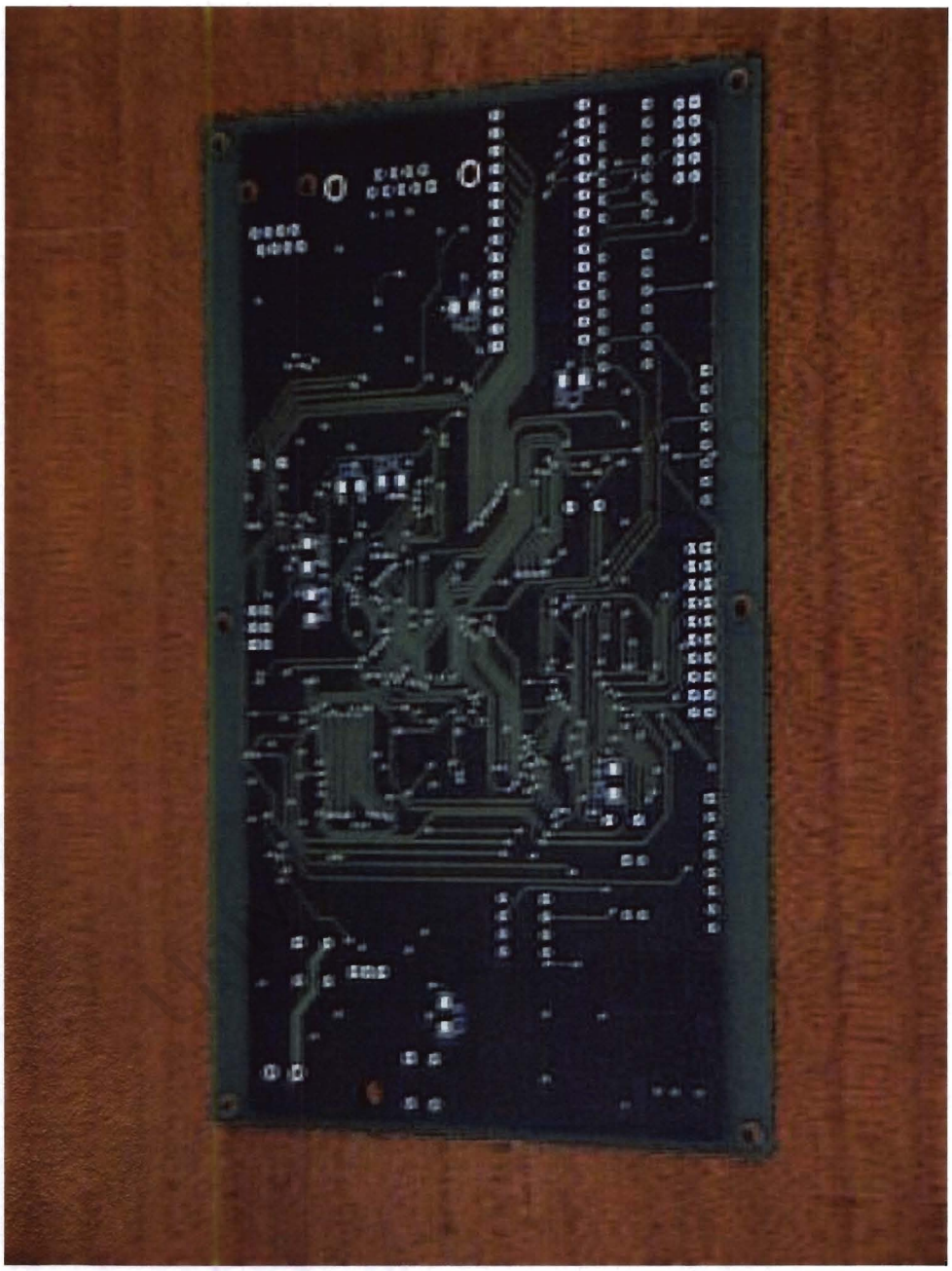

Figure 3.11: Unpopulated PCB (bottom layer)

address and data lines. Figure 3.12 shows the PCB board. The Ethernet analogue circuitry was placed as far as possible from the Ethernet digital circuitry and other digital devices so as to reduce noise.

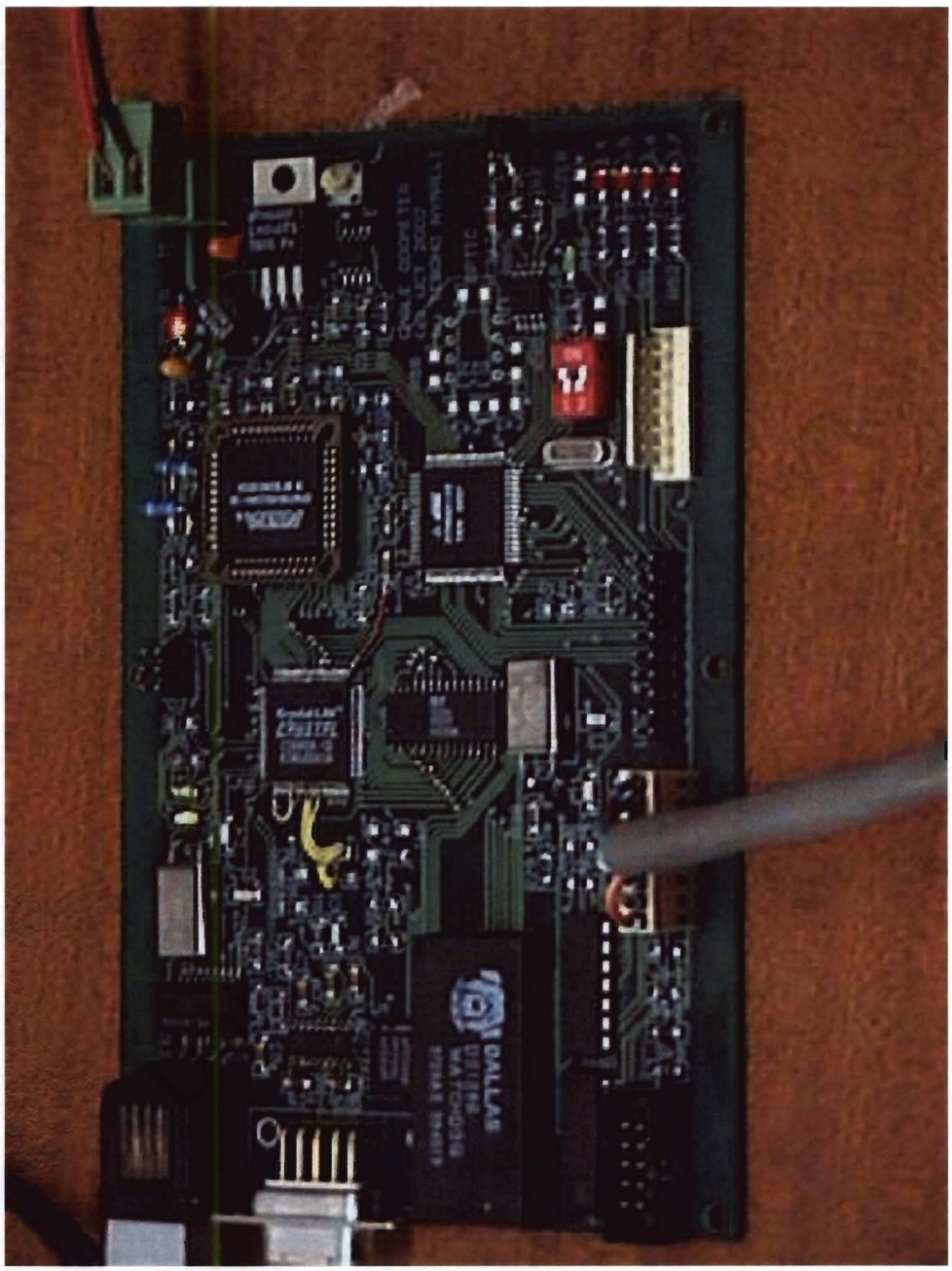

Figure 3.12: Populated PCB

3.4 Protection of Odometer

There are some cases when the Odometer will be used in mines where it could come into contact with dust and fluids. In such cases the Odometer needs to be protected from

such an environment for it to operate correctly. Figure 3.13 shows how and where the Odometer is mounted on the box which is used for protection.

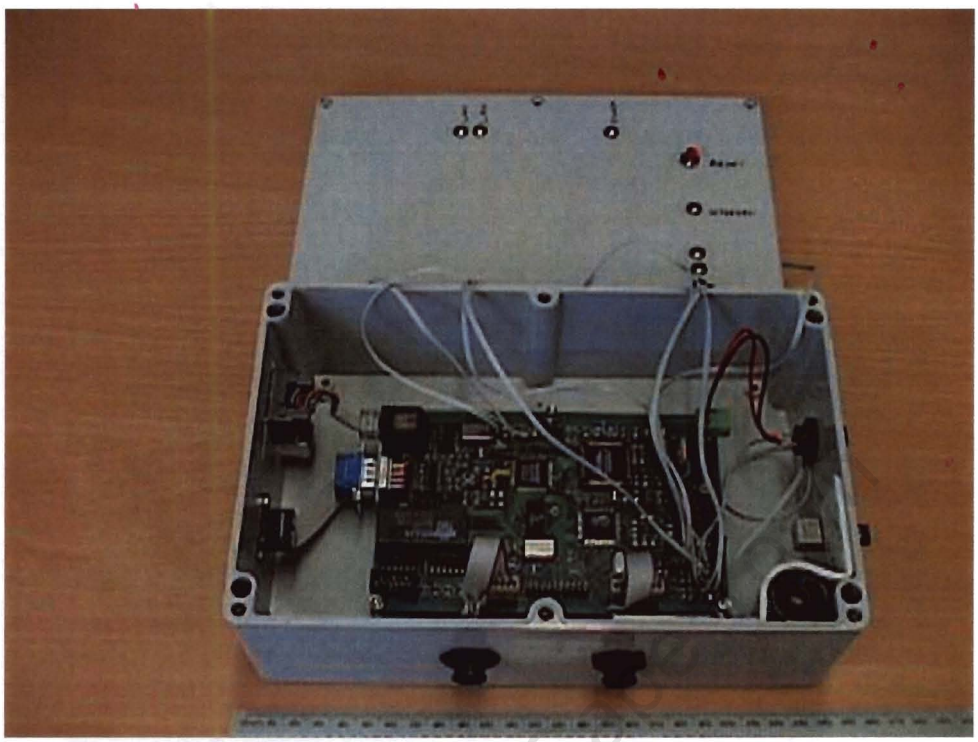

Figure 3.13: Enclosure of Odometer

The PCB is of dimensions 150mm x 100mm. A Polycarbonate box of dimensions 240mm x 160mm x 90mm fits the PCB board with enough allowance for the coaxial connector mounted on the walls of the box as shown in Figure 3.13 .

In order for the box to be shower proof, all the connectors on the box must be shower proof and connected firmly to the box.

3.5 Conclusions

The analogue circuitry is placed as far as possible from the digital circuitry to avoid electro-magnetic interference and therefore performance is increased.

The distance moved by radar is determined from the encoder pulses. The 16-bit counter counts these pulses. It covers a distance of 32m (when the resolution of the encoder is 0.05cm) before it overflows. The Odometer will be used for much longer distances. This problem will be solved in software.

Chapter 4

Software design

The software developed in this project was mostly written in C programming language. VHDL was also used for the CPLD chip. The C code was compiled using an Imagecraft compiler while VHDL was compiled by the Altera max plus 2 version 10.2 compiler. No operating system was used in this project. The PC software was written in HTML language. The modes of operation and software design are discussed in this chapter.

4.1 Modes of operation

The modes of operation of the system are the core specifications for software design. There are two main modes of which the others are derived from. The source code that implements the modes discussed below is included in Appendix A under /Odo/EIT128a directory in the main.c and script.c file.

4.1.1 Timestamped data

A flow chart shown in Figure 4.1 applies for both modes discussed below. *Count* is the value obtained from the 16-bit counter. *Direction* is chosen as up or down when there is an input capture interrupt. *Max. down* and *up* are counter values at which directions changed. x is the interval at which Radar makes a survey.

Transfer to remote data storage

In this mode, the encoder data is used to determine position and direction the Radar is moving in. The data is then timestamped by RTC on board and then transfered to a remote data storage. The data storage could be the database shown in Figure 2.3.

Data stored on board

This mode is similar to the above mode except that the timestamped data is stored on board in the Dataflash. This mode is used for cases when there is no network connection.

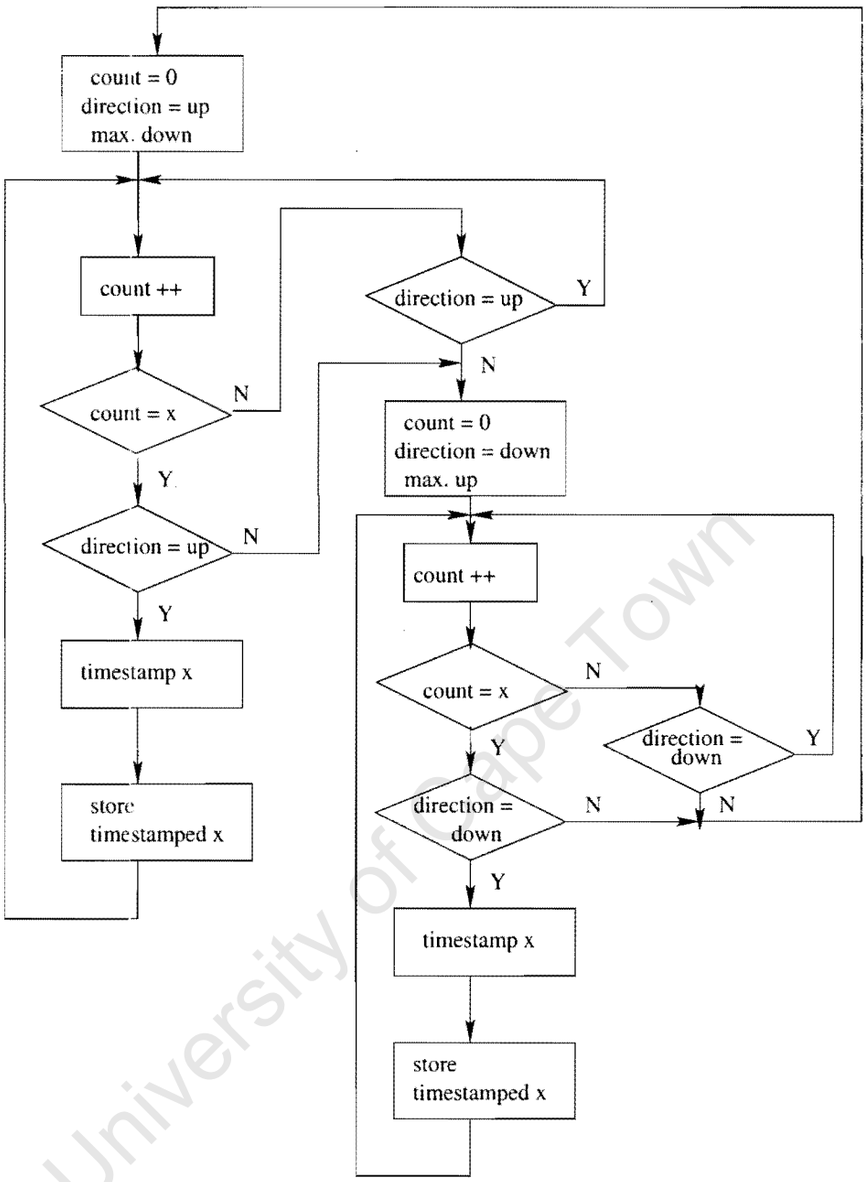

Figure 4.1: Flow chart for time-stamping the data

When buffer gets full the timestamped data is transfered to Dataflash.

4.1.2 Non timestamped data

The flow chart shown in Figure 4.2 applies for the three modes discussed below. In these modes, the data is not timestamped as it is not stored or transfered to a remote database. The data is used to trigger the Radar directly.

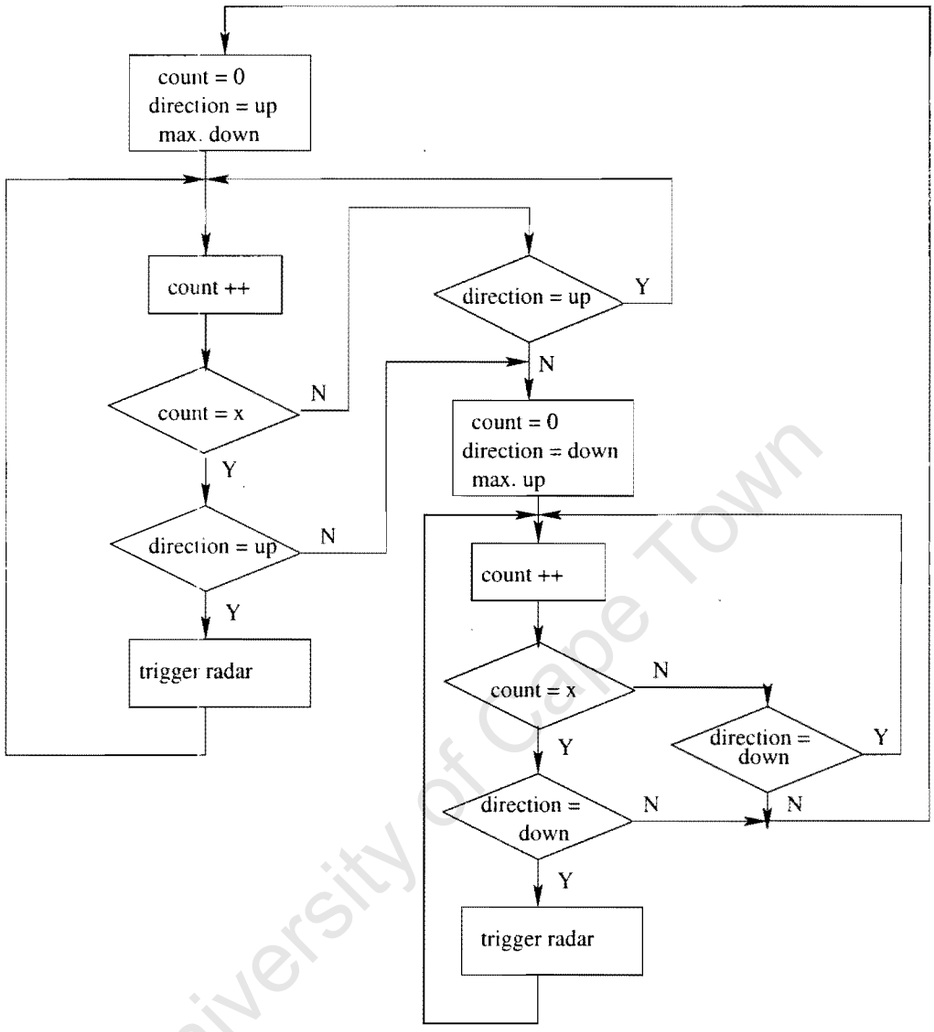

Figure 4.2: Flow chart for non timestamped data

Trigger Radar via optic fibre

In this mode there is a direct link between the Odometer and Radar. The Odometer triggers the Radar via an optic fibre link.

Trigger Radar by switching LED on/off

An LED switches on and off to signal the person operating the Radar system to trigger it. This mode is used when Radar and Odometer are close to each other (i.e. when Radar surveys a horizontal surface as shown in Figure ??(b)).

Trigger radar by Microswitch or Buzzer

In this mode a Microswitch (i.e. relay) connected to Odometer is used to trigger Radar. Another option is to connect a Buzzer to the Odometer. The sound of Buzzer signals the person operating Radar to trigger it.

4.2 Odometer software

All of the software discussed in this section resides in the memory space of the Atmega128. The subsections below discuss the softwares implemented on main components of the Odometer. The source code for the subsections below is also available in Appendix A under the EIT128a directory.

4.2.1 Encoder software

Signals T1 and IC1 shown in Figure 3.2 are used to determine the position and direction of Radar as it moves up or down a bore-hole or horizontally above the surface. The encoder outputs pulse T1 that depend on the CPR of encoder. The resolution of encoder is determined from its CPR. Knowing the resolution of encoder and number of pulses generated by encoder, the exact position of Radar can be determined. IC1 signal triggers the input capture interrupt to signal a change in direction.

The position at which Radar surveys the ground of interest is stored in an array called ring buffer shown in Figure 4.3. This position is stored with a time-stamp obtained from a time source (DS1286 chip) on board. The ring buffer is used because the rate at which data is received from encoder might differ from the rate it is transfered to a data storage. Received data is appended at the end of ring buffer while data being transfered is removed from the start of ring buffer. All the above operations are done in software.

4.2.2 Ethernet software

The TCP/IP protocol suite used in this project consists of four main layers [5]. These are the application, transport, network and link layer. A brief explanation of these layers is given below:

- **Application layer**: handles the details of certain applications such as telnet, browser, file transfer application and email applications.

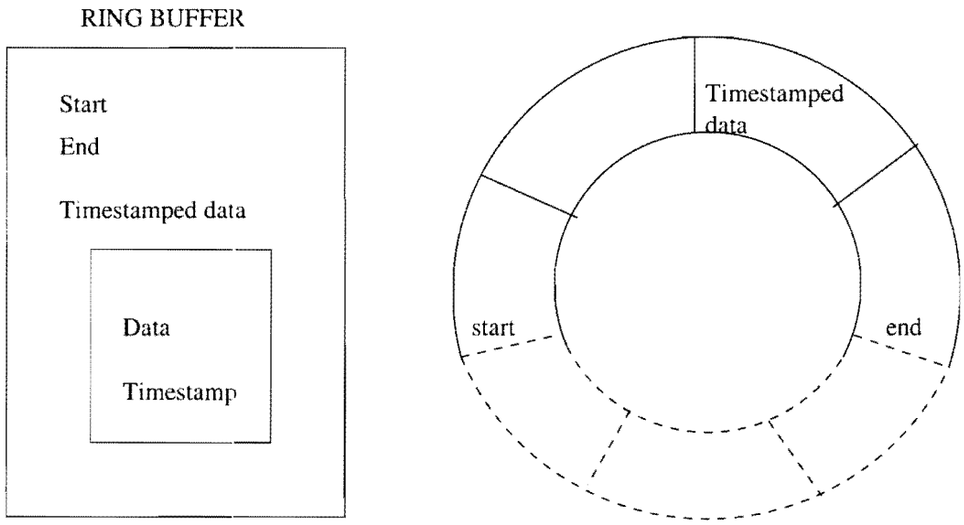

Figure 4.3: Structure of a ring buffer

- **Transport layer**: TCP insures that there is a reliable flow of data between two hosts. The data is divided into small TCP segments that can be transfered over the network. It also acknowledges received packets and retransmits lost packets. UDP sends packets, with no guarantee that the other host receives them. In this case, the application layer above is responsible for insuring that packets are received.
- **Network layer**: These layer is responsible for routing the frames through the physical network. The mapping of the IP address to the physical address is done in these layer.
- **Link layer**: These layer includes the device drivers (i.e. CS8900A driver in these case) used for connecting to the network and the physical wires or medium in which data is transferred.

Figure 4.4 shows the four layers used in these project with the various protocols implemented. It also shows the dependency of the protocols on others.

CS8900A driver

The CS8900A Ethernet driver is included in the link layer as part of the TCP/IP stack. Since the Ethernet controller is used in 8-bit mode, interrupts are not supported [13]. Therefore, polling mode is used. But before the CS8900A can be in this mode, the driver has to initialize it to its default configuration or load configuration from an external EEPROM. Figure 4.5 shows flow chart of how the CS8900A is initialized. The calibration of the on chip analog circuitry takes approximately 10 ms [4]. In this project there is no external EEPROM, therefore the default configuration is used.

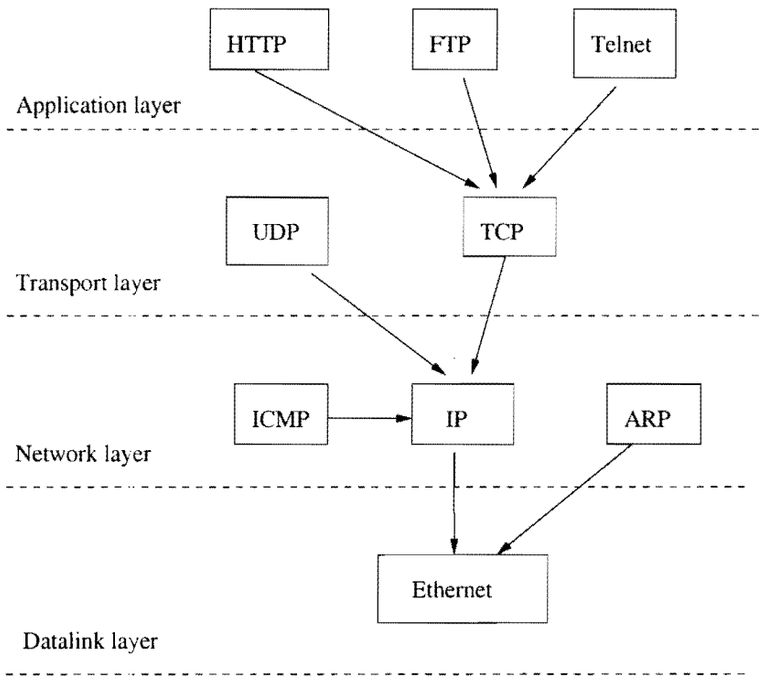

Figure 4.4: TCP/IP suite

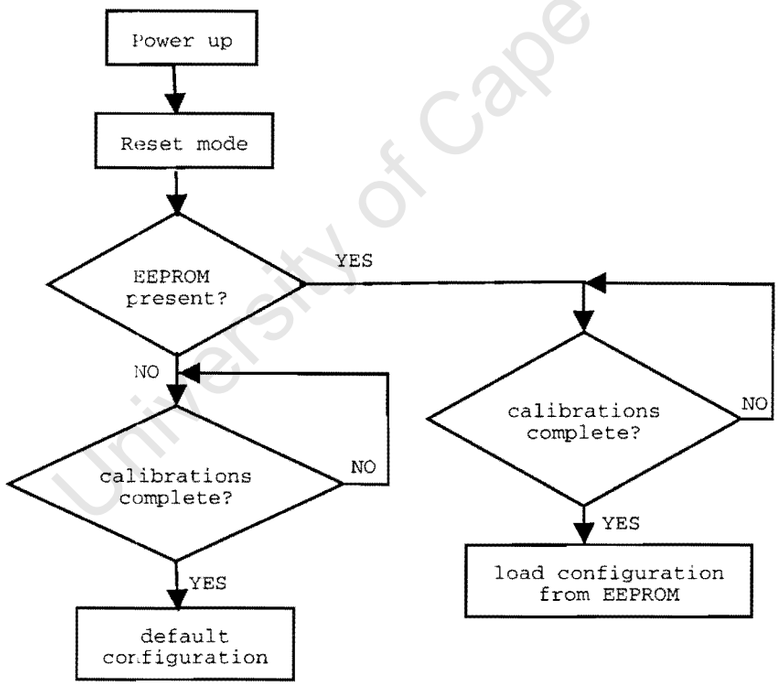

Figure 4.5: Initialization of CS8900A

When the CS8900A comes out of reset with default configuration, the following steps are taken as part of the initialization routine:

1. wait for the RESET bit in Self control register to be cleared;
2. wait for initialization done bit in Self status register to be set;
3. configure the MAC engine and physical interface;
4. set the test modes of the CS8900A;
5. determine what type of frames to accept and configure how to filter destination address;
6. write Ethernet address;
7. enable I/O and memory mode.

When initialization is complete the Ethernet controller is ready to enter the polling mode. Figure 4.6 shows a flow chart that describes the polling mode up to the stage control is transfered to the Network layer (i.e received IP).

Control of board over network

The Odometer is controlled via a web-page stored in the Dataflash on board. The HTTP client (i.e. a web browser on PC) requests services from the server (i.e. the odometer) via network using HTTP protocol. Table 4.1 shows commonly used request methods implemented on a HTTP daemon.

Table 4.1: HTTP request methods

Method	Description
GET	Request to read a web page on server (i.e Odo.htm) or any HTTP URI
POST	Transmits data or any form input information to the requested web page on server
HEAD	Request to read web page's header
PUT	Request to store web page on server
DELETE	Remove web page from server
LINK	connect two existing web pages
UNLINK	disconnect an existing connection between two web pages

The GET and POST request are available in the HTTP daemon implemented in the application layer. In this project they are used for providing the diameter of the wheel for the optical shaft encoder, the interval at which radar takes readings, selecting the mode of operation of odometer and transferring the file that contain timestamped data to client.

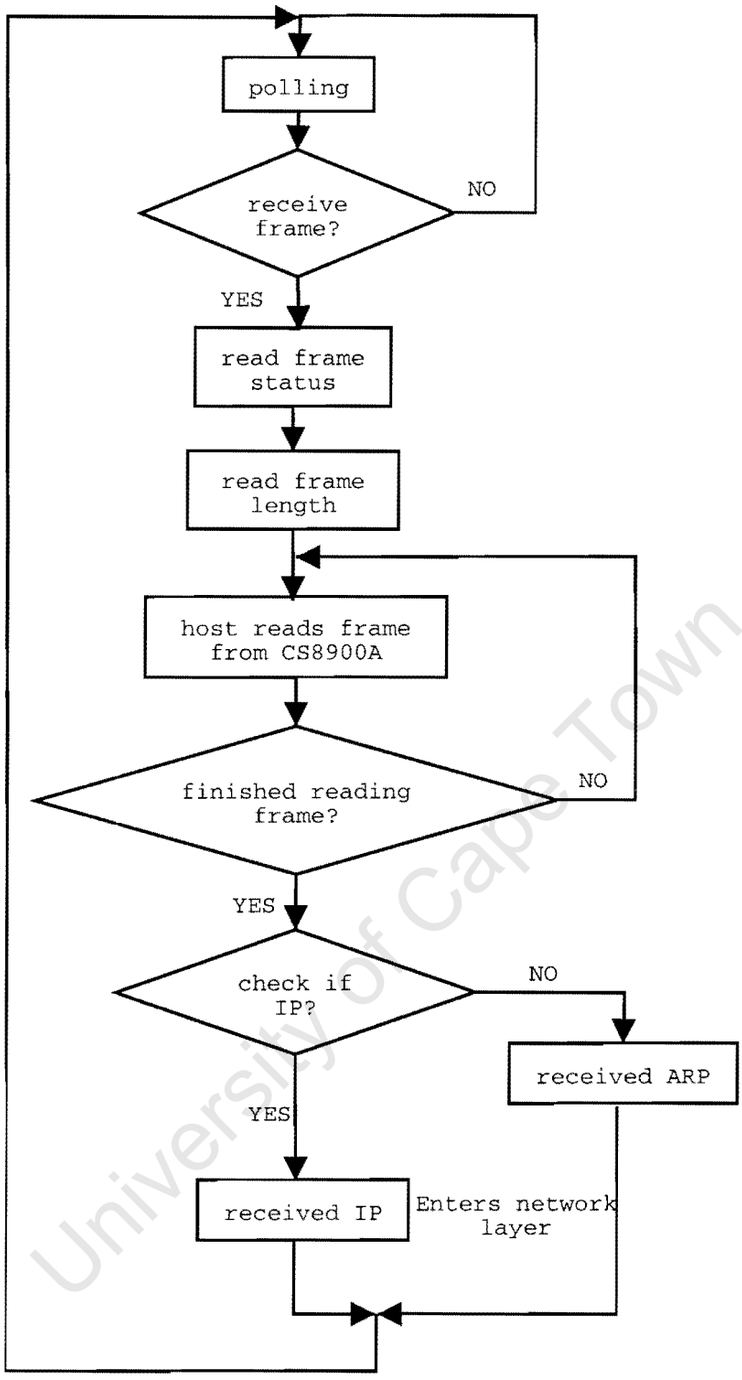

Figure 4.6: Polling mode

4.2.3 CPLD software

A CPLD is used in this project for memory mapping other devices to the Atmega128. The devices mapped are CS8900A Ethernet controller, IDT71256S SRAM and DS1286 RTC. The CPLD acts as logic gates which select the device to be mapped to the Atmega128. Max+Plus 2 is a development software that was used for designing the logic. The language that is used for designing this logic is VHDL. VHDL is completely integrated into the max+plus 2 software package.

Address lines A15, A14, A13 with signals $\overline{WR}$ and $\overline{RD}$ from the Atmega128 are used as inputs to the logic when selecting the devices to map. Since CS8900A operates in two modes (I/O and memory mode) the modes are used to select the device instead of the chip select input pin. For the SRAM and RTC, their chip select input pins are used when selecting the devices. The following logic is implemented in the CPLD and used to select the devices.

$$\overline{IOR} = A15 \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{RD} \quad (4.1)$$

$$\overline{IOW} = A15 \cdot \overline{A14} \cdot \overline{A13} \cdot \overline{WR} \quad (4.2)$$

$$\overline{MEMR} = A15 \cdot A14 \cdot \overline{A13} \cdot \overline{RD} \quad (4.3)$$

$$\overline{MEMW} = A15 \cdot A14 \cdot \overline{A13} \cdot \overline{WR} \quad (4.4)$$

$$\overline{CE} = \overline{A15}\left(\overline{RD} + \overline{WR}\right) \quad (4.5)$$

$$\overline{CE2} = A15 \cdot \overline{A14} \cdot A13\left(\overline{RD} + \overline{WR}\right) \quad (4.6)$$

All three devices are active low. $\overline{IOR}$ and $\overline{IOW}$ select I/O mode while $\overline{MEMR}$ and $\overline{MEMW}$ select memory mode operations for the CS8900A. $\overline{CE}$ selects the SRAM and $\overline{CE2}$ selects the RTC. From the above equations the base addresses at which the devices are map to the Atmega128 are determined by the the three address lines. The VHDL source code that is used to implements this is included in Appendix A in directory /Odo/cpld.

4.3 XML and HTML

In this project either XML or HTML could be used to view the web-page used as an interface for controlling the Odometer and the timestamped data stored on the Odometer. Since the data has already been processed (i.e. position and direction determined) on the odometer, a method of only displaying this data was needed. HTML was designed to display data, therefore it is used in this project. The source code for the web-page is included in Appendix A under directory /Odo/webpage.

4.4 Conclusions

All modes of operation have been implemented independent from an operating system. When there is a network connection, timestamped data is transfered to a remote database in real time. When there is no network connection or buffer is full, the timestamped data is transfered to local data storage. The tests performed are discussed in detail in the next chapter.

Chapter 5

Testing the prototype Odometer board

All the tests that have been performed are discussed here. These tests include power supply level and testing functionality of components individually. To verify that the design fulfills that user requirements an acceptance test procedure is performed. The amount of data the board can store, speed at which the radar can move and the maximum distance that the Odometer can measure are also discussed in this chapter.

5.1 Hardware and Software verification

5.1.1 Power supply

The first step was to inspect the PCB. The continuity of address and data bus, control nets with the relevant components was tested. A 5V voltage regulator was mounted and the voltage level at all components being supplied by 5V was measured with an multimeter. The tests were made before any of this components was placed on the PCB.

Table 5.1 show the components that consume most of the current supplied to the Odometer board. The current is approximately 550mA. This is less than the 1A current rating of the voltage regulator. A heat sink may be attached to the regulator to reduce the heat dissipation.

5.1.2 Testing individual components

The functionality of most of the components was tested but only the tests performed on the main components will be discussed below. The main components were mounted to PCB by a company called Rhomco that specializes in assembling PCB. The programs used to test the main components are available in Appendix A under directory /Odo/test. There is a README file in this directory for more information on directories involved.

Table 5.1: Current consumption

Components	Current(mA)
Atmega128	200
CS8900A	65
IDT71256	150
EPM7064S	65
AT45DB021B	4
DS1286	15
HD25	20
MAX202	8
MAX707	20

Microcontroller

After the components had been placed, the voltage levels at the power connections of Atmega128 were measured. A simple program that switches LEDs on and off was programmed to the Atmega128. Other tests included a program that transfered a string of characters to a PC x-terminal via the USART interface.

CPLD

The EPM7064S is used to memory map the SRAM, RTC and Ethernet controller to the microcontroller. To test the functionality of the CPLD all of the above above mention devices should be accessed by the microcontroller.

Dataflash and SRAM

The functionality of the Dataflash was verified by writing and reading to a specific memory location on the AT45DB021B. The value read was transfered via the USART to the PC x-terminal. The source code is available in Appendix A under directory /Odo/test/Dataflash.

When the external memory control registers are enabled, the external SRAM is written or read automatically by accessing an address location above 0x10FF [10]. A test program (included in Appendix A under directory /Odo/test/sram) that writes and reads from a memory location above 0x10FF was used.

Ethernet controller

To test that the CS8900A has been mounted properly, a boundary scan test was performed. This test checks the orientation and if there are any open or short circuits [4]. Boundary scan test consist of two cycles, output and input cycles. Both tests were performed. The next test done on the CS8900A was to check if the internal registers were accessible. To verify these, the product identification register should contain a unique 32 bit code that

identifies the chip as a CS8900A. The first two bytes of this register contain the signature code (i.e. 0x630C) and the last two bytes contain the product ID number [4]. The "ping" program was then run as a test that the CS8900A can communicate over the Ethernet.

RTC

The time and day of the year read from the DS1286 RTC chip was incorrect. A program (included in Appendix A under directory /Odo/RTC) was designed to correct the time and day of the DS1286 chip.

5.2 Acceptance test Procedure

To verify that the odometer fulfills the user requirements the following tests were carried out. Figure 5.1 shows the set-up of the whole system. The Odometer on one side is connected to an optical encoder mounted on a stand with a wheel while on the other side the Odometer is connected to the network.

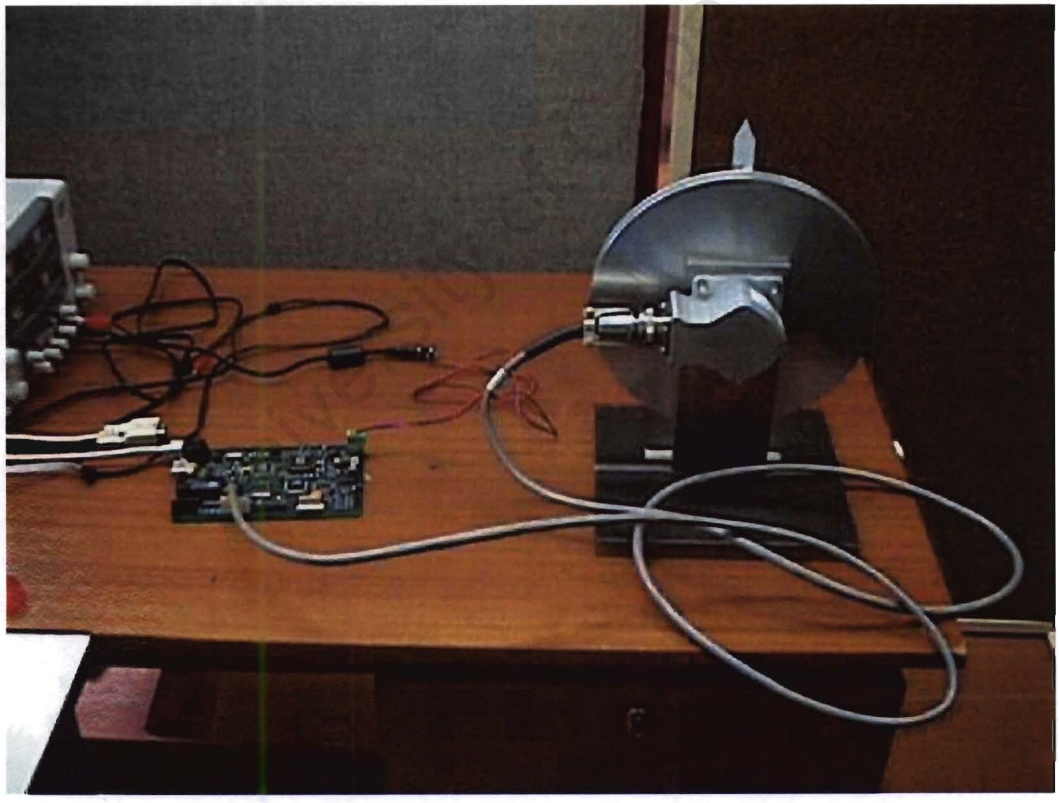

Figure 5.1: Setup of system during testing

5.2.1 ATP table

Table 5.2 verifies that the user requirements mentioned in Chapter 3 have been fulfilled. In Table 5.2 the numbers in first column refer to the section in Chapter 3 where the user requirements were stated.

Table 5.2: ATP table

user requirements	performance/achieved	comments
3.2.1.1	yes	when the radar changes direction the distance is reset to zero at that instant
3.2.1.2	yes	accurate to mm.
3.2.1.3	yes	
3.2.1.4	yes	resolution is 0.03 cm.
3.2.1.5	yes	protection of odometer discussed in section 5.5 .
3.2.2.1	yes	diameter of wheel selected on web page.
3.2.2.2	yes	external 32K SRAM included.
3.2.2.4	yes	2 Mb Dataflash included on board.
3.2.2.5	partly	NTP client not implemented. RTC used for times-tamping data.
3.2.2.8	yes	all mode implemented. Only LED option tested . The others use the same principle.

5.2.2 Accuracy

The Table 5.3 and a graph in Figure 5.2 show by how much the values determined by the odometer deviate from the actual values.

Table 5.3: Circumference of encoder wheel (i.e. diameter is 20cm)

Frequency	Calculated distance (cm)
2	62.73
2	62.77
2	62.80
6	62.83
5	62.86
2	62.89
1	62.95

The actual circumference of encoder wheel used during the tests is 62.8318cm. The mean which is calculated using the values in Table 5.3 is 62.83 cm. The standard deviation is 0.05 cm. Figure 5.2 shows the Gaussian distribution and histogram of the values shown in Table 5.3.

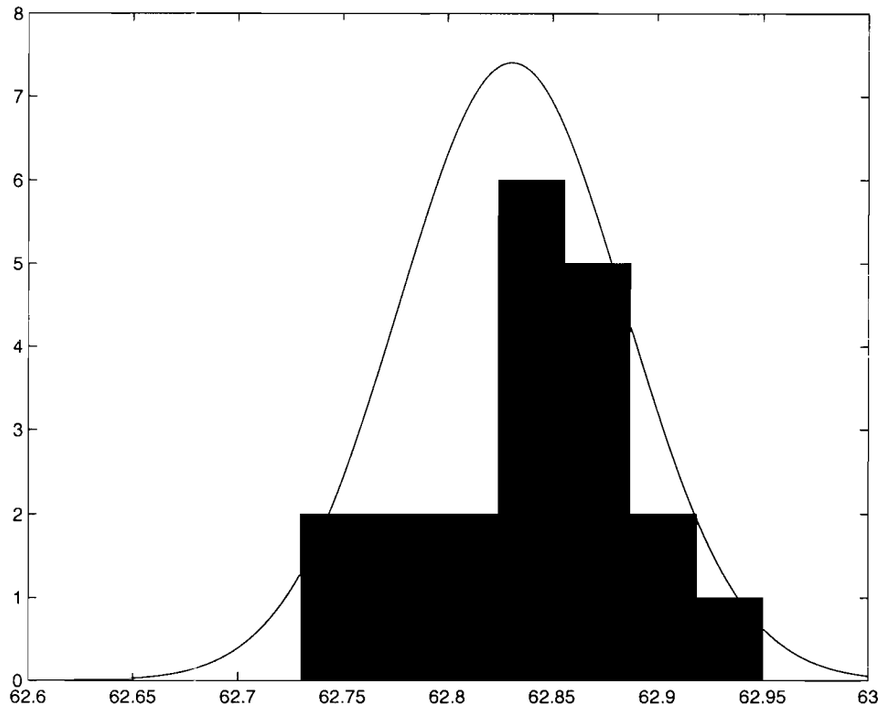

Figure 5.2: Gaussian distribution and histogram of encoder wheel circumference

5.3 Performance

The Odometer board can be controlled remotely via a network connection. When the Odometer is connected to network, the user (i.e. operator in Figure 2.3) can upload the web-page by entering the IP number 137.158.131.7 on a web browser (as shown in Figure 5.3). The desired parameters should be entered else the default wheel diameter and interval of taking readings which are 20cm and 5cm respectively will be used. The mode of operation should also be chosen. Any request to the Odometer is made by pressing the start button.

5.3.1 Remote data storage

In the presence of a network connection, the data timestamped is transfered to a remote data storage from a temporary buffer which can contain up to 600 values. If the radar takes readings at 5cm interval, the distance that can be moved before the buffer gets full is 30m. Due to the limited SRAM size the HTML page that contains timestamped data is limited to 1500 bytes. The page transfers an average of 45 timestamped data values at once. At the top of each page the time and day of the year the data was recorded is displayed (e.g. Figure 5.4). The time-stamp for each data on that page is relative to that time.

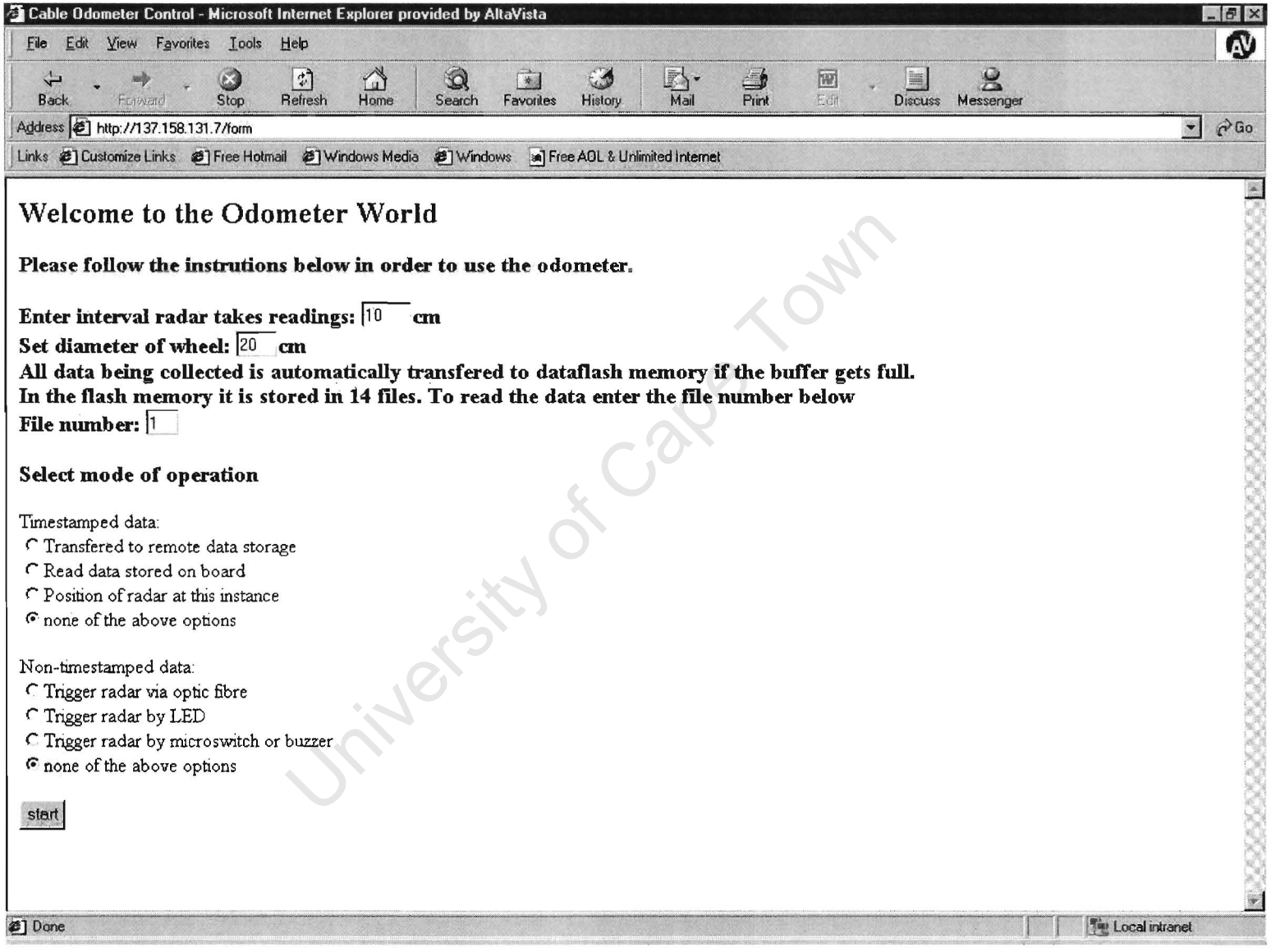

Figure 5.3: Odometer web page.

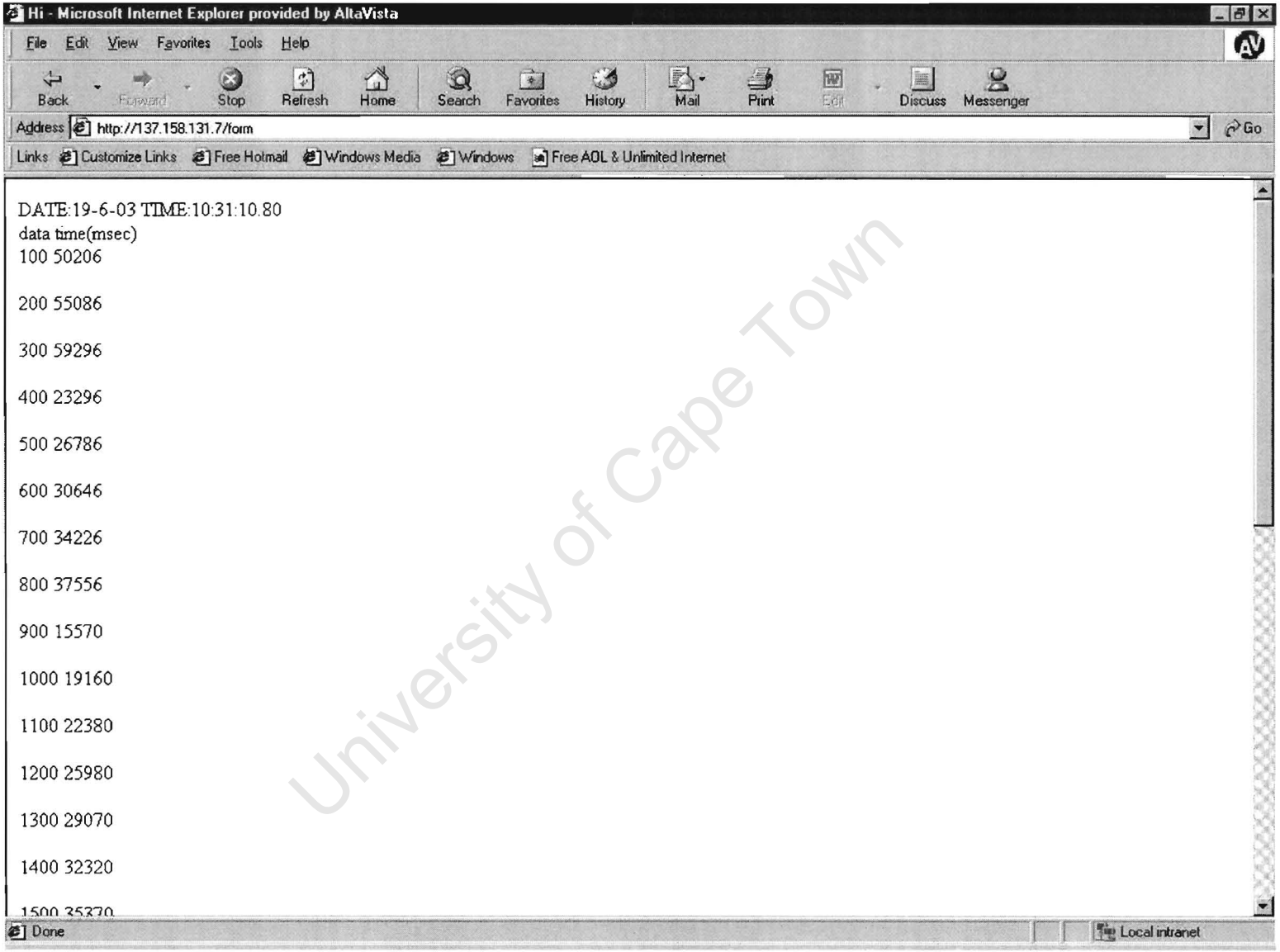

Figure 5.4: Timestamped data at 100cm interval.

5.3.2 On-board data storage (Dataflash)

A file system exit on the Dataflash. Any data that is stored on the Dataflash is stored in a file format. Figure 5.5 shows how the files are organized. To access any file in Dataflash, the file system makes a linear search for the file [5]. One read access is needed for each file. For N number of files, average number of files accesses is n:

$$n = \frac{N}{2} \tag{5.1}$$

One page size is 264 bytes. The number of pages for one file n_p:

$$\begin{aligned} n_p &= 1 + \left\{ \frac{size}{pagesize} \right\} \\ &= 1 + \left\{ \frac{size}{264} \right\} \end{aligned} \tag{5.2}$$

SPI uses one fourth of the frequency of the microcontroller[10]. For each SPI clock cycle one bit is read or written. Access time t_p [5] for one page is:

$$t_p = \frac{f_{clk} \cdot 264 \cdot 8}{4} \tag{5.3}$$

Average read time T [5] for one file :

$$T = (n + n_p)\, t_p \tag{5.4}$$

The file size is 1500 bytes. Average number of timestamped data per file is 45. When the temporary buffer in SRAM get full the data is automatically transfered to the Dataflash. The number of files needed to store this data is:

$$\begin{aligned} n &= \frac{600}{45} \\ &= 13.3 \\ &\approx 14 files \end{aligned} \tag{5.5}$$

For 14 files of size 14 x 1500 bytes = 21Kbytes, $f_{clk} = 4MHz$:

$$n_p = 80.545 \tag{5.6}$$

$$t_p = 2.11ms \tag{5.7}$$

Therefore, 14 files are written in:

$$T = 28.86ms \tag{5.8}$$

The size of the Dataflash is 2,162,688 bits which is 270 336 bytes. The size of the file (server.ini) that contains the MAC, IP, password for FTP and other parameters is 223 bytes. The size of the two other files (odo.htm and odo.syn) that contain the web pages is 2746 bytes. Free space for data storage is therefore 267367 bytes. One full buffer occupies :

$$\begin{aligned} Buffer_{600} &= 1500 * 14 \\ &= 21000bytes \end{aligned} \tag{5.9}$$

The number of blocks B_{14} (i.e. one block is 14 files) that can be put into Dataflash before it fills up is:

$$B_{14} = 12 \tag{5.10}$$

When considering interval taking readings as 5cm, the buffer gets full when a distance of 30m is covered. Data that covers a distance of 360m can be stored in Dataflash.

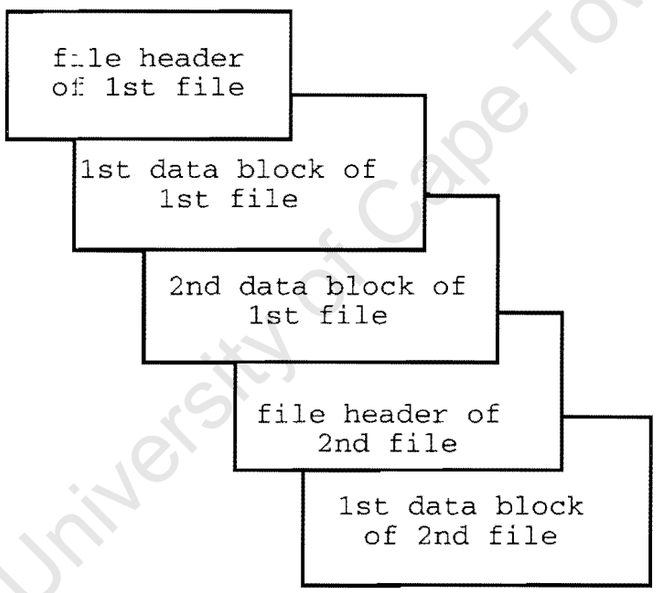

Figure 5.5: Block organization in file system (with reference to [5])

5.3.3 Maximum distance

The maximum distance that can be measured is limited by the 16 bit counter (i.e $2^{16} = 65536$). When the counter reaches its maximum value 65536 an overflow interrupt oc-

curs and the counter starts from zero again. For a wheel diameter of 20cm, the distance resolution of encoder r_{20}:

$$\begin{aligned} r_{20} &= \frac{\pi \cdot d}{2048} \\ &= 0.03cm \end{aligned} \tag{5.11}$$

This implies that for every pulse produced, a distance of 0.03 cm is moved. One count is made by the 16-bit counter (shown in Figure 3.4). The 65536 counts (result in an overflow) cover a distance of 19.66m. This is not sufficient if the system is to be used in mines. To overcome this problem a simple algorithm shown in Figure 5.6 was implemented where:

count - value used in calculating distance.

count_16 - counter value obtained from 16-bit counter.

count_over - value incremented by one when overflow interrupt occurs (i.e. when counter reaches 65536).

where increment count is:

$$count = count\_over + count\_16 \tag{5.12}$$

and increment overflow count is:

$$count\_over = count + 1 \tag{5.13}$$

The maximum distance now depends on the integral type of count (i.e. char, int or long). In this case count is unsigned long which its maximum value is 2^{32}. To obtain distance, count is multiplied by resolution. Therefore maximum distance D:

$$\begin{aligned} D &= 2^{32} * 0.03 \\ &= 128849018.9cm \\ &= 1288km \end{aligned} \tag{5.14}$$

Figure5.7 shows timestamped data at 100cm interval of taking readings. Because of limited space where measurement were taken, only a distance of 32m could be measured.

5.3.4 Speed

The maximum speed at which a Radar can moves up and down bore-hole has a limit. Apart from the main factor mentioned in Section 4.2.1 of this dissertation, other factors

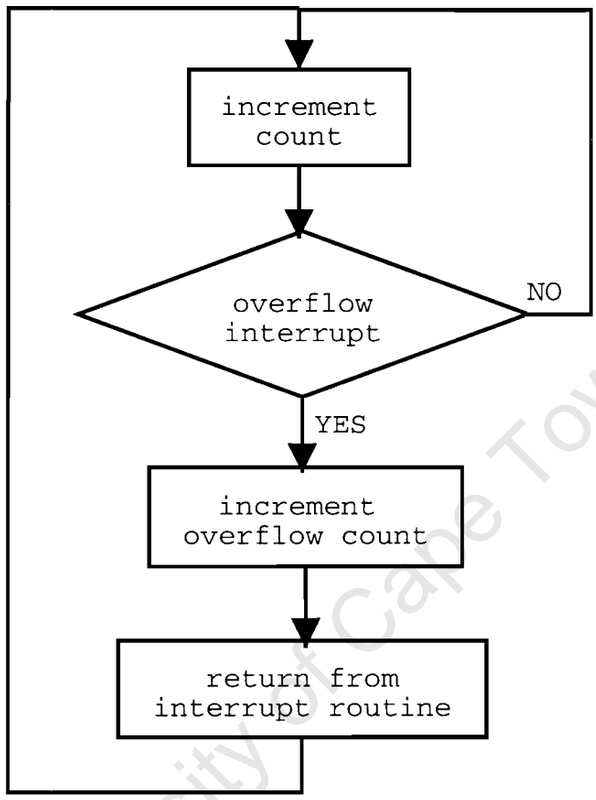

Figure 5.6: Extending the maximum distance that the Odometer can cover

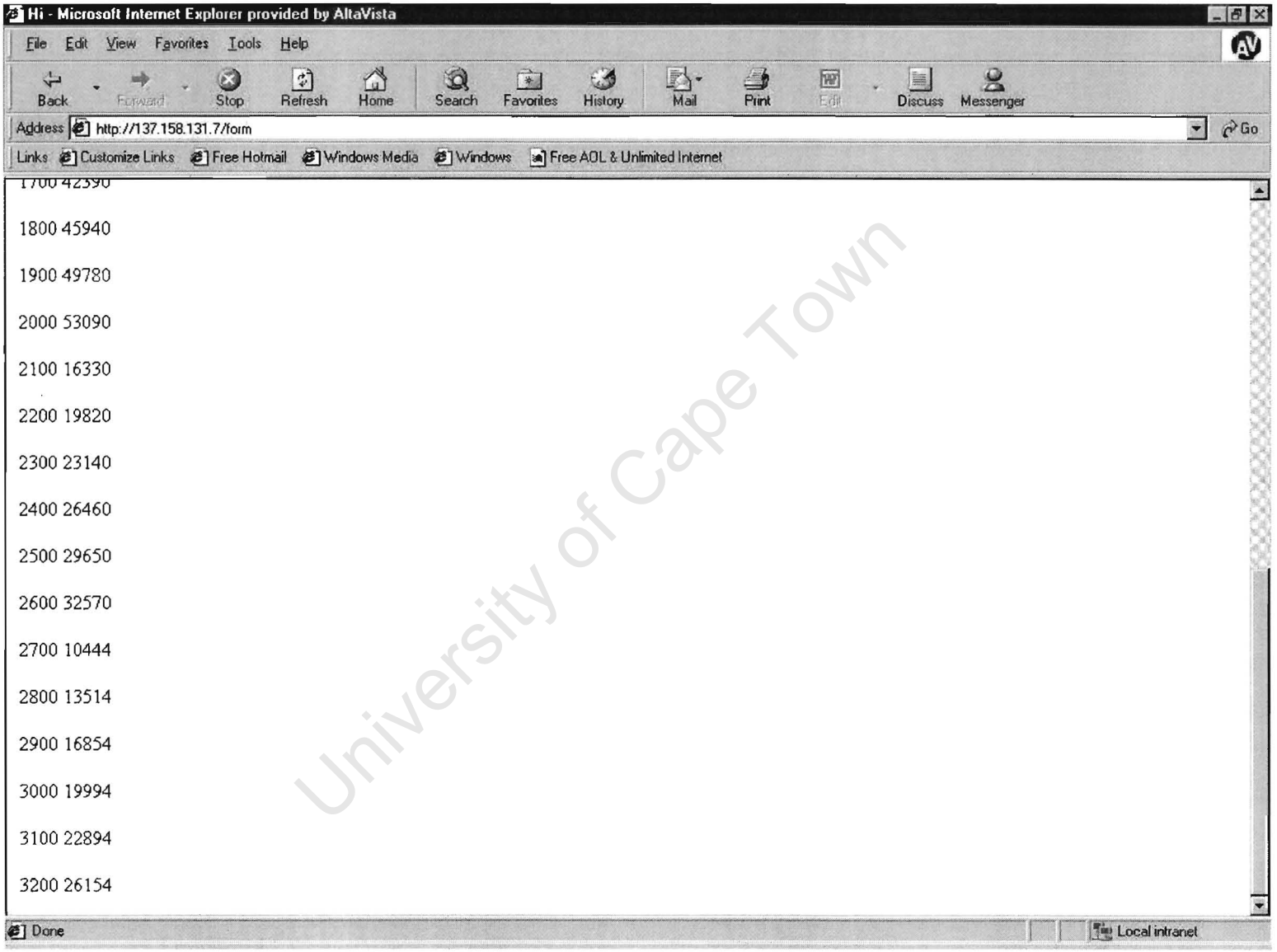

Figure 5.7: Maximum distance measured

that affected speed are the interval at which Radar takes readings and the diameter of encoder wheel. Tables 5.4 and 5.5 and graphs in figure 5.8 shows the relationship.

Table 5.4: Diameter of wheel is 20cm

interval (cm)	max. speed(m/s)
5	0.33
10	0.34
15	0.37
20	0.40
25	0.44
30	0.48
35	0.52
40	0.53
45	0.58
50	0.63
55	0.78
60	0.88

From the graphs in figure 5.8 it is clear that a diameter greater than 23 cm has a greater influence on the maximum speed at which the Radar can move up or down a bore-hole.

Table 5.5: Interval of taking reading is 10cm

diameter (cm)	max. speed (m/s)
10	0.26
20	0.36
30	0.61
40	0.82
50	1.08
60	1.25
70	1.37
80	1.49

5.4 Conclusions

The power supplies for the different components were correct. The microcontroller and its peripherals performed as expected. Most of the user requirements have been accomplished. When there is a network connection, an operator at remote location can have full control of the Odometer. Diameter of the encoder wheel influence the maximum speed of Radar more than the interval at which the Radar takes readings.

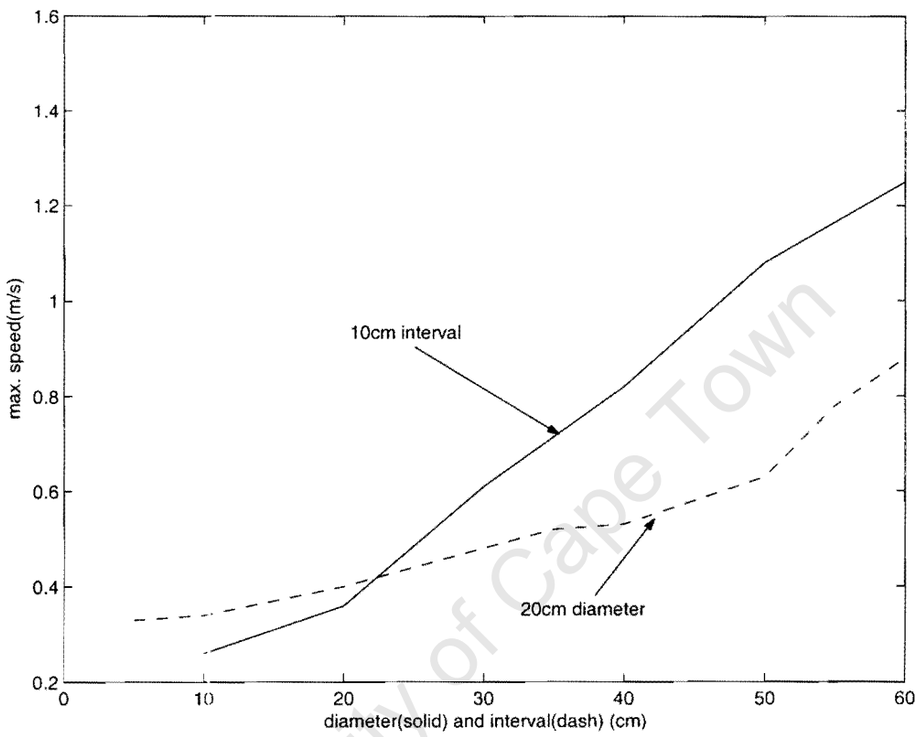

Figure 5 8: Speed at which radar moves up or down bore-hole

Chapter 6

Conclusions and Future work

In this chapter a conclusion of the project is made considering the user requirements and results obtained from concept study, hardware and software design and tests performed. This is followed by a discussion of future work that can be done to improve the odometer design.

6.1 Conclusions

The HD25 encoder provides a distance resolution of 0.03 cm that is much higher than the required 1 cm. This reduces distance errors such as digitization and initialization error that are associated with the resolution. The time resolution should be at least 20 ms and DS1286 has a resolution of 10 ms. Therefore, it was unnecessary to implement a software clock in the Atmega128.

The NTP client was not implemented in this project and DS1286 RTC drift by less than 1 minute per month at 25 degree celsius. This implies that the RTC had to be corrected manually.

The Atmega128 has a throughput of up-to 16 MIPS at 16 MHz. But due to the slow access time of the DS1286, the Atmega128 had to be used at a frequency of 4 MHz. This was still sufficient for the processes running on the Atmega128.

The external 32K SRAM was sufficient for the temporary storage of the TCP/IP stack and the buffer that holds up-to 600 timestamped data values. The buffer is large enough to give time for data to be transfered via network to a remote data storage or else be stored on board when buffer gets full.

The 2M bits Dataflash is sufficient for on board storage of timestamped data. When a full buffer is transfered, it occupies 21Kbytes in Dataflash. With regard to the size of Dataflash 12 blocks can be stored. This implies that 7200 timestamped data values can be stored in Dataflash in the absence of a network connection. There is a file system on the Dataflash. One full buffer is stored in 14 files in Dataflash. This implies that 12 blocks contain 168 files. Transferring these files to remote data storage one at a time would be

time consuming. FTP applications implemented during software development enable all files to be transfered using a single command (i.e. mget).

The speed of the Radar can be calculated using the time-stamped data.

6.2 Future work

The aim of the project was to develop a system that fulfills the user requirements. Most of the requirements have been fulfilled. But, the system can still be improved. Some points that can be addressed for future work are:

- There is no operating system implemented in this project. Implementing an operating system such as eCos can solve some of the problems encountered in this project. Most NTP clients need an operating system platform. Therefore with the presence of eCos an NTP client can be ported to the odometer.
- NTP client should be implemented to correct the on board RTC.
- A complete TCP/IP stack should be implemented.
- The DS1286 has an access time of 150 ns. This implies that the Atmega128 cannot use an oscillator greater than 6 MHz. An option would be to divide the SRAM into sectors such that the external SRAM is configure for different wait states at different sections of memory. The disadvantage of the latter is that DS1286 use only 64 bytes whereas the smallest sector is 4 Kbytes. More than 3K bytes of SRAM will not be used.
- The data stored in the data base of the Geophysical Recording and Processing System is in XML format. In this project data is stored in HTML format. Therefore, this data should be converted to XML format in future work.

Bibliography

[1] Imagecraft compiler. Technical report, http://www.imagecraft.com.

[2] Watchdog timekeeper. Technical report, Dallas semiconductors, November 1999.

[3] CMOS Static RAM. Technical report, Intergrated Device Technology, February 2001.

[4] Crystal LAN ISA Ethernet Controller, Product Datasheet. Technical report, Cirrus Logic, April 2001.

[5] Embedded Web Server, application notes. Technical report, Atmel Corporation, November 2001.

[6] Industrial rugged metal optical encoder, Product datasheet. Technical report, USDigital corporation. December 2001.

[7] MAX 7000 Programmable logic device family. Technical report, Altera, November 2001.

[8] Quick start guide for the embedded internet toolkit. Technical report, Atmel Corporation, October 2001.

[9] 2 megabit 2.7 volt only Datafalsh. Technical report, Atmel Corporation, May 2002.

[10] 8 bit avr microcontroller with 128K bytes in-system Programmable flash. Technical report, Atmel corporation, September 2002.

[11] Private conversation with G. Doyle and P. Mukhopadyay as geophysicist who were involved in the Kleinzee survey, 2002.

[12] Real Time Clock(RTC) using the Asynchronous Timer. Technical report, Atmel Corporation, May 2002.

[13] J. Ayres. Using the crystal CS8900A in 8-bit mode. Technical report, Cirrus Logic, January 2000.

[14] G. T. Becker. RighTime tm A Real Time Clock correcting program for MS-DOS-Based computer system. Technical report, Air System Technologies, Inc, December 1992.

[15] G. E. Blight. Guide for the Preparation of Theses, Dissertations and Project Reports. Technical report, University of the Witwatersrand, Faculty of Engineering, Johannesburg, December 1990.

[16] D. Bodas. CS8900A Ethernet controller technical reference manual. Technical report, Cirrus logic, June 2001.

[17] M. Inggs. Geophysical Recording and Processing System. Technical report, Remote Radar Sensing Group, Department of Electrical engineering, UCT, November 2001.

[18] D. Mills. Time and time interval measurement application to computer and network performance evaluation. Technical report, Department of Electrical engineering, University of Delawar, January 1996.

[19] D. Mills. Network Time Protocol daemon. Technical report, Department of Electrical engineering, University of Delawar, January 2001.

[20] Pual Herselman, Wessel van Brakel, Wessel J. Croukamp, Mark Rutchlin and Gavin Doyle. Kleinzee Trial Borehole Radar Survey on 22-27 March 2001. Technical report, Stellenbosch University and University of the Cape Town, July 2001.

[21] W. H. Pual Horowitz. *The art of electronics*. Cambridge University Press, Trumpington Street, Cambridge CB2 1RP, 1980.

[22] V. Smotlacha. Measurement of Time Servers. Technical report, CESNET association, Czech republic, December 2001.

[23] F. G. Stremler. *Introduction to Communication Systems*. University of Wiscousin, Madison, 1977.

[24] D. R. Wehner. *High Resolution Radar*. Artech House, Norwood, MA 02062, 1987.

Chapter 7

Appendix A

Software Source Code and Datasheets

Due to the size of the source code and number of Datasheets used, all of the Datasheets and source code have been omitted from this report. All this information is available on attached CDROM. The main directory has a README file used to explain briefly the contents of the other directories.

Chapter 8

Appendix B

Schematics and PCB layout

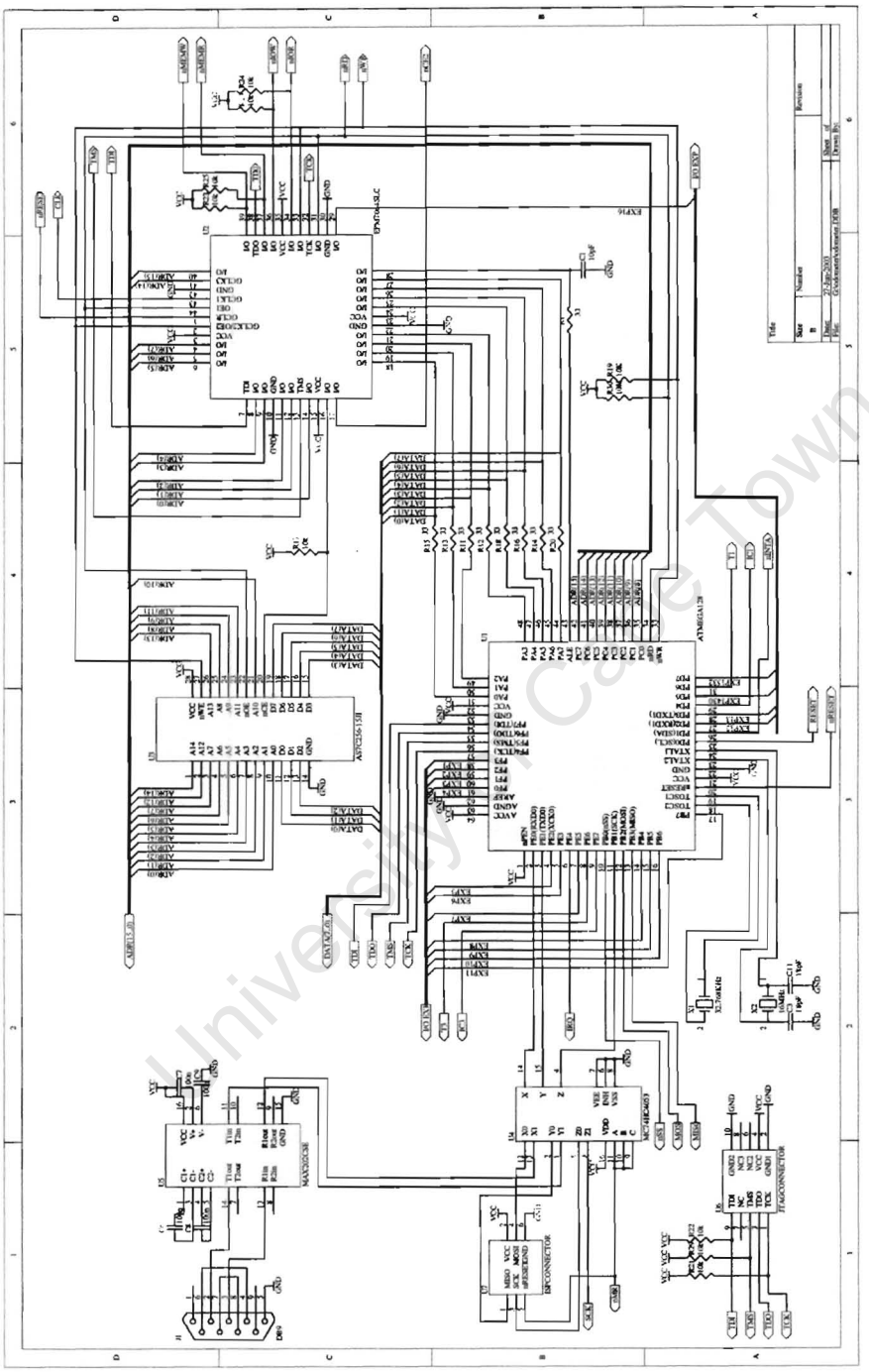

Figure 8.1: Microcontroller and peripherals schematic

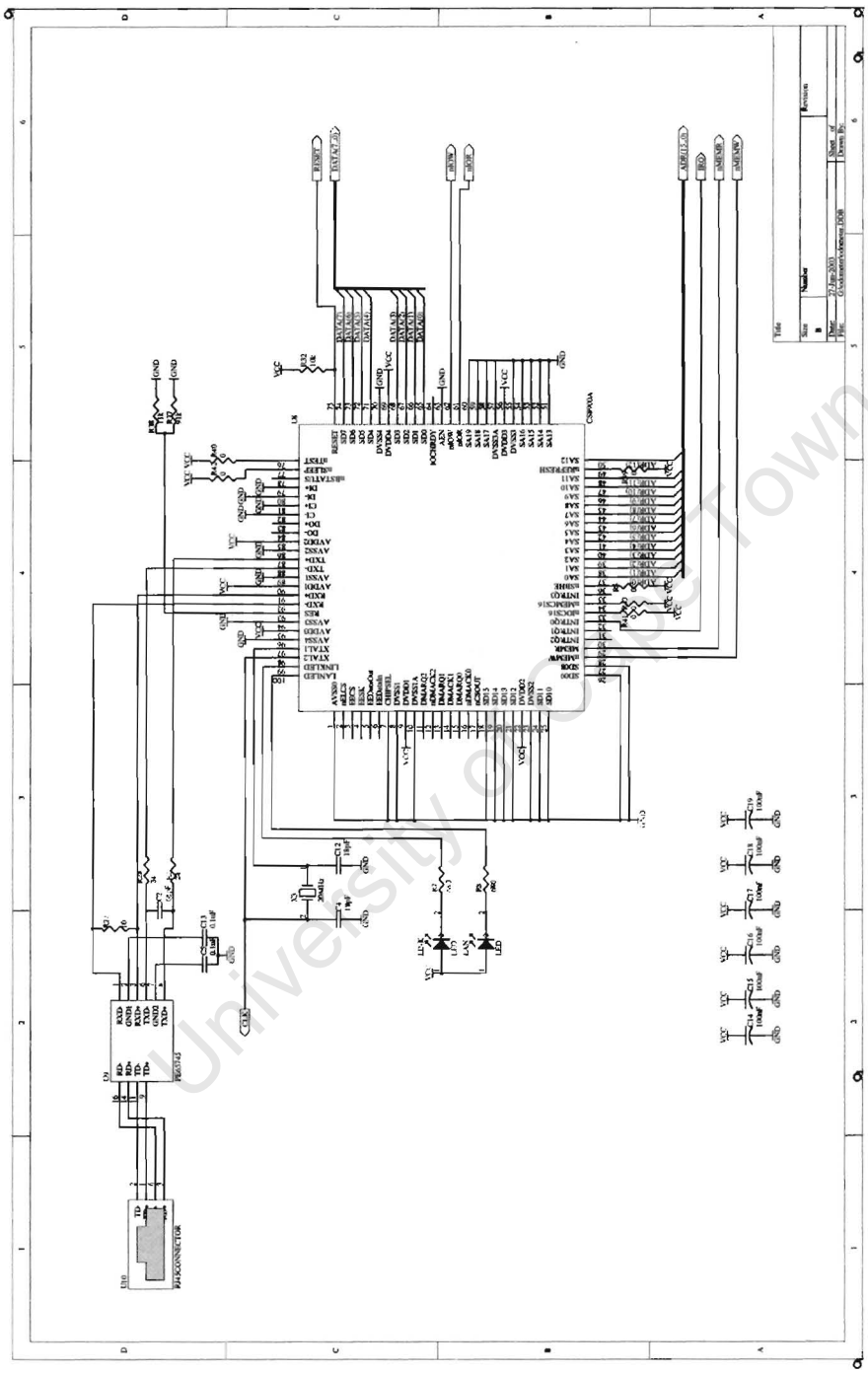

Figure 8.2: Ethernet controller schematic

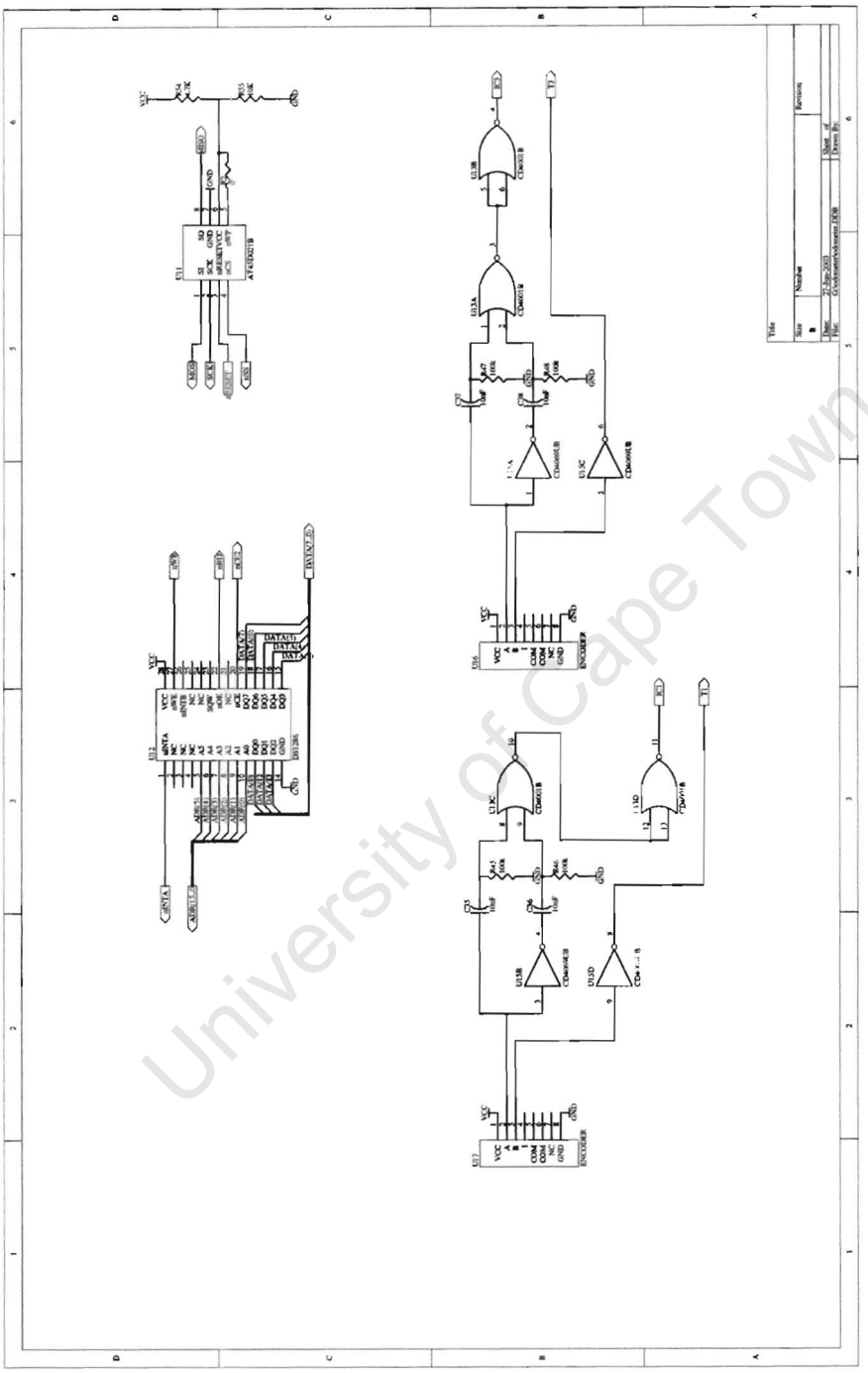

Figure 8.3: Encoder, RTC and Dataflash schematic

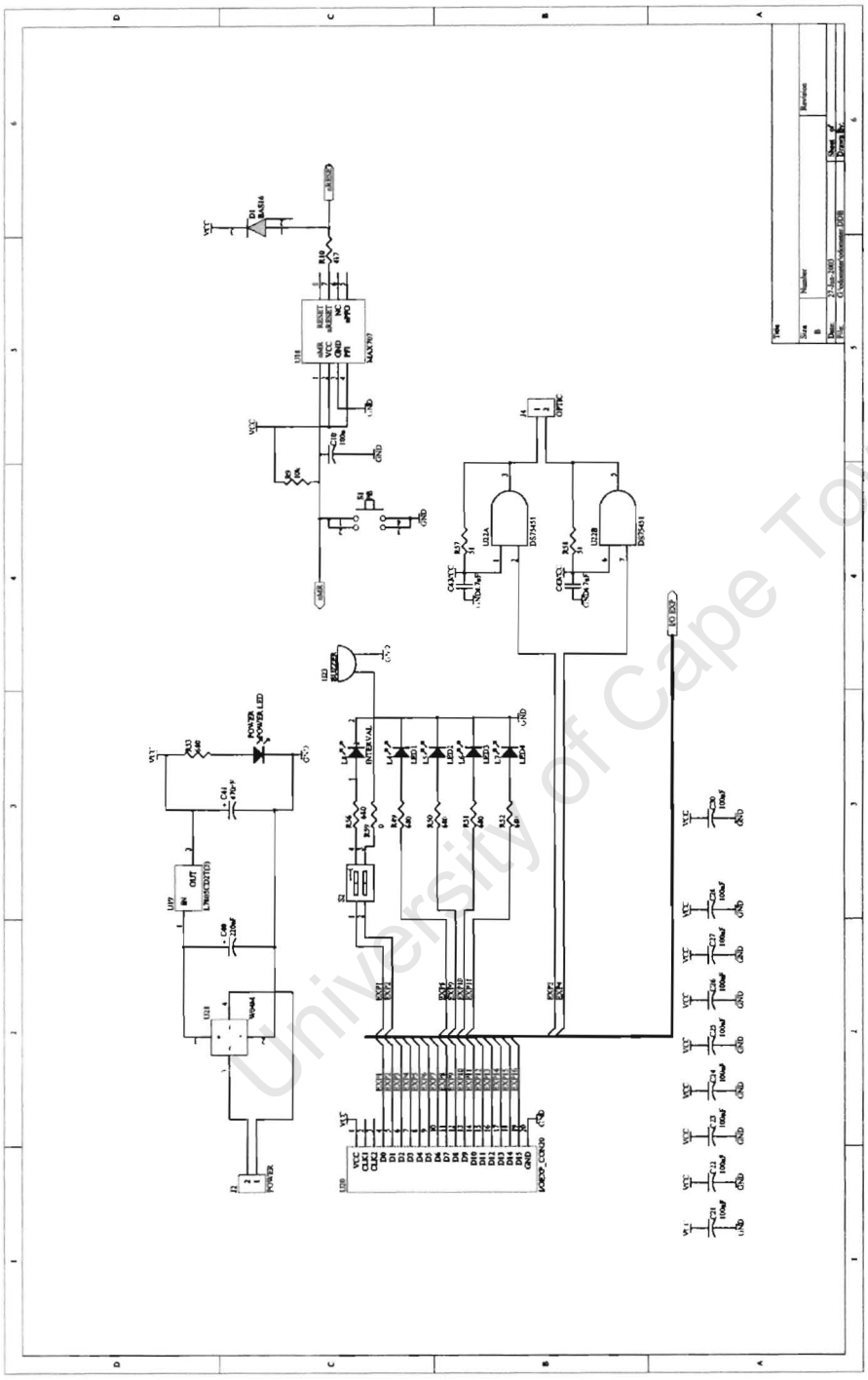

Figure 8.4: Expansion connector schematic

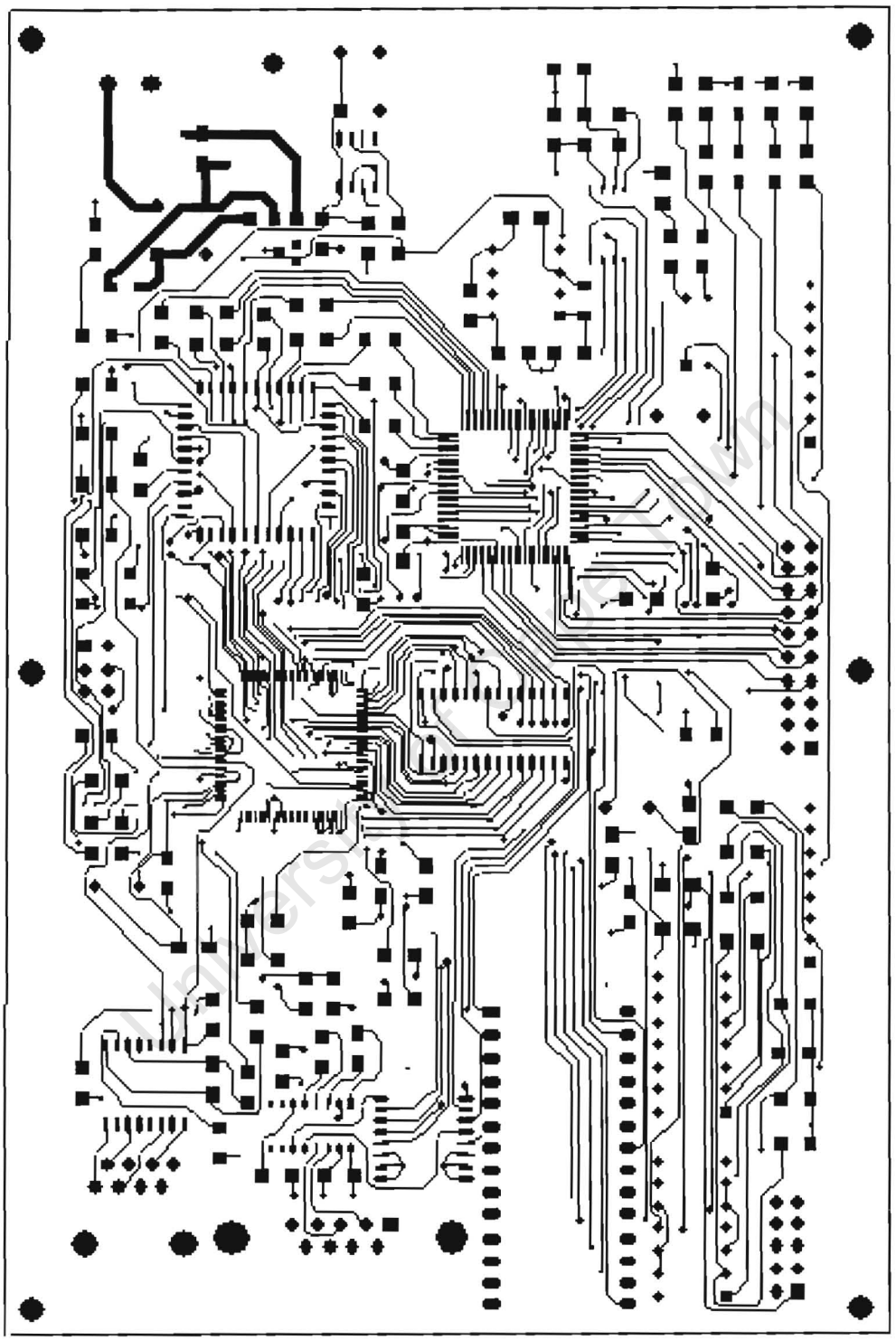

Figure 8.5: Top layer for PCB

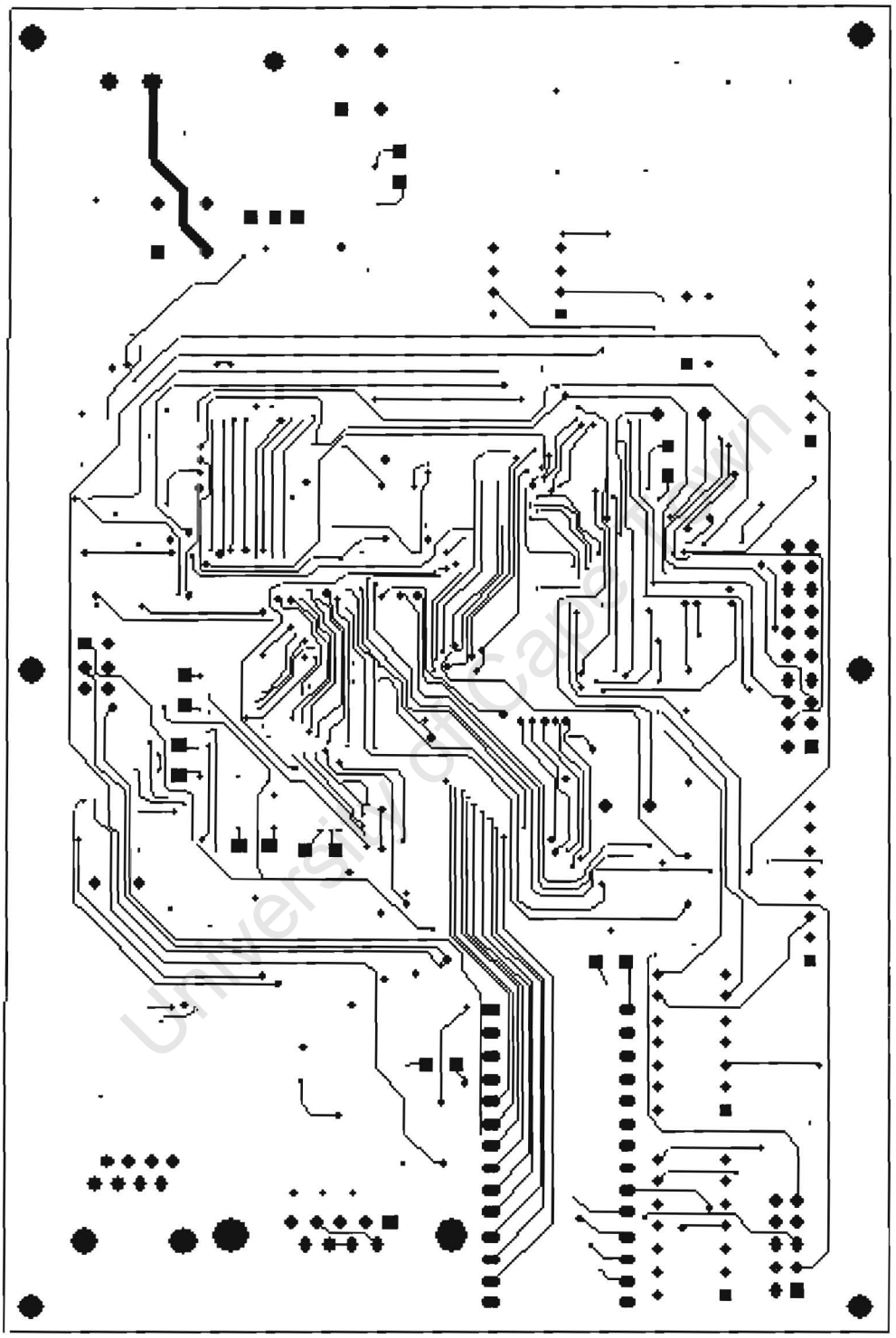

Figure 8.6: Bottom layer for PCB

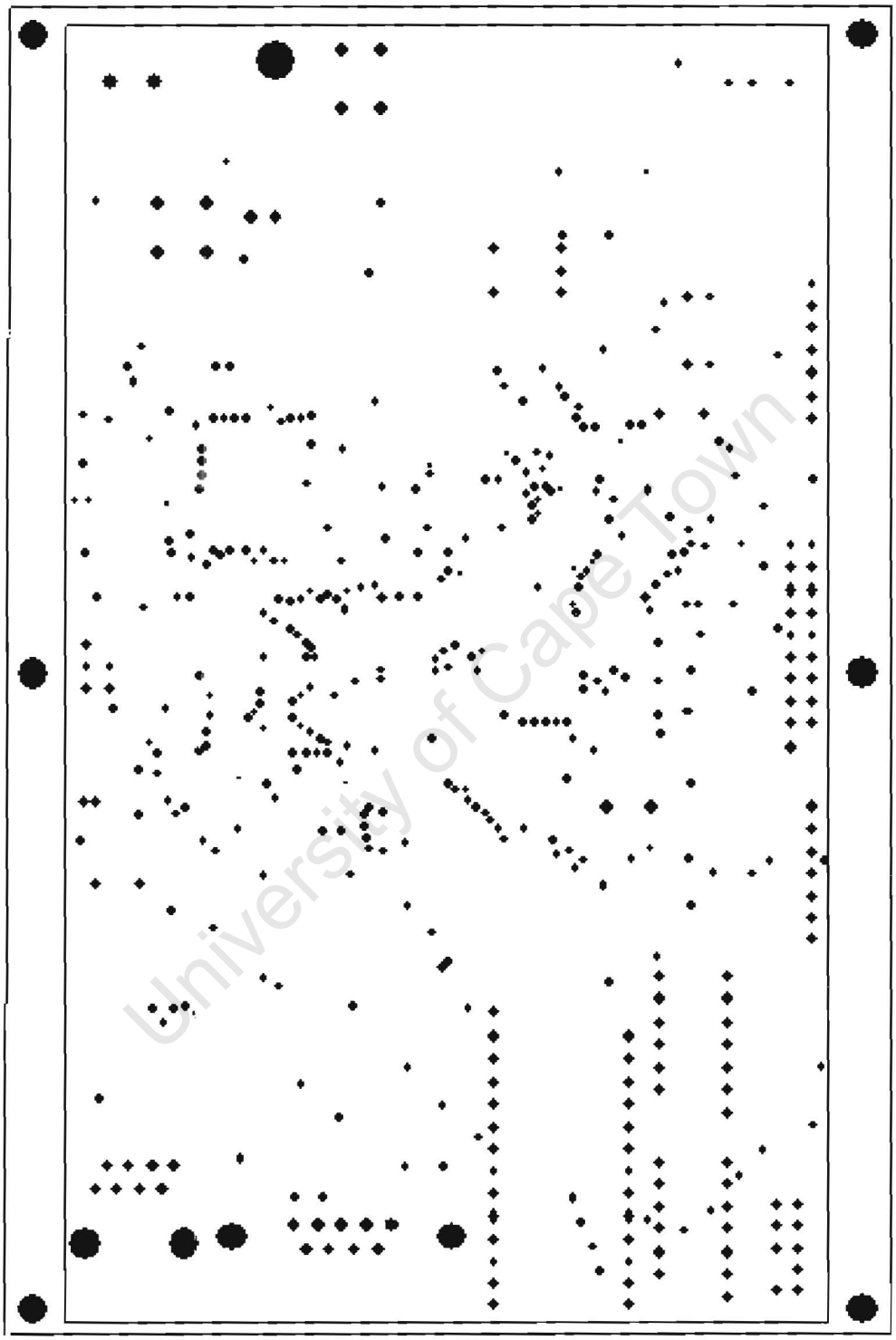

Figure 8.7: Ground plane for PCB

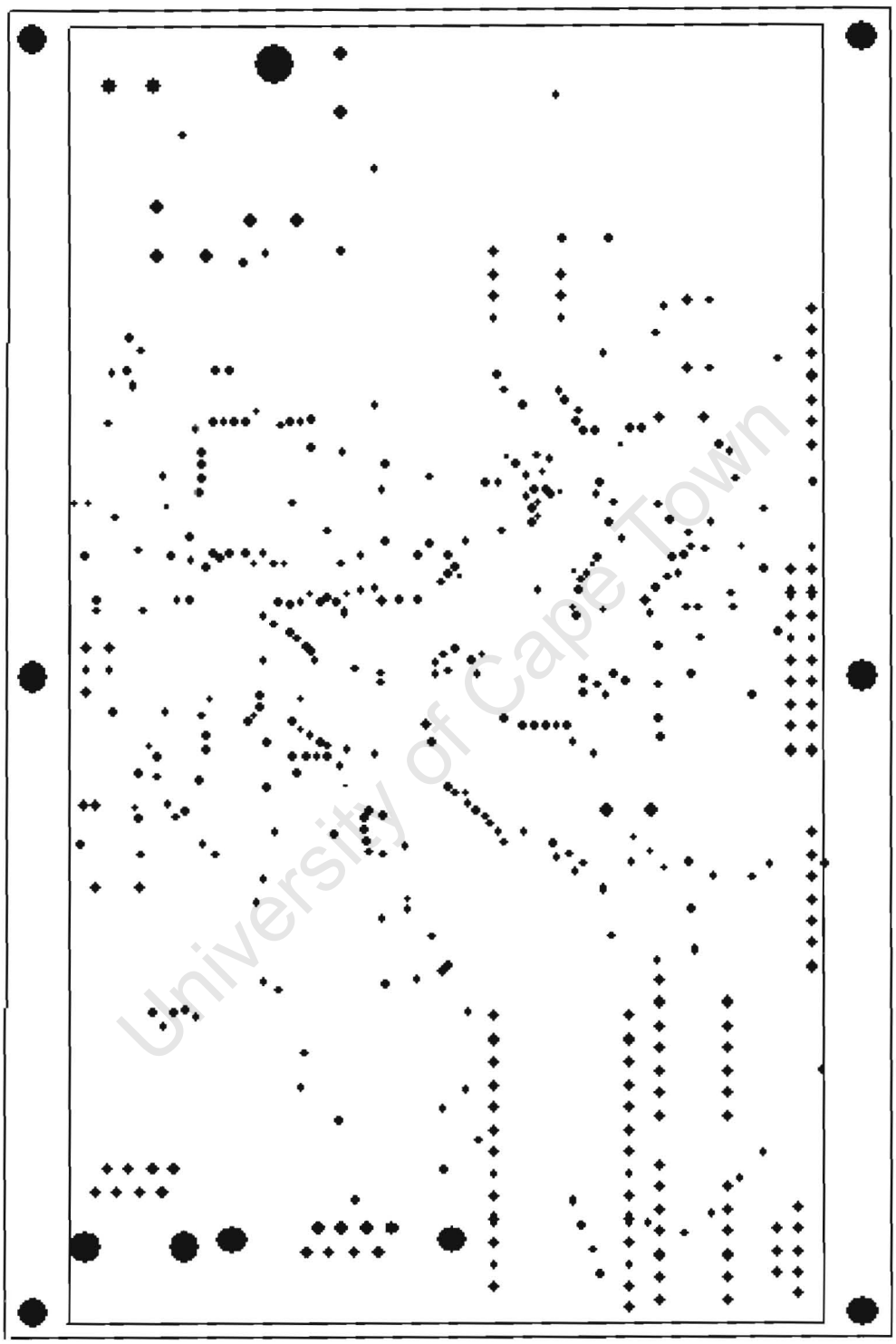

Figure 8.8: Power plane for PCB

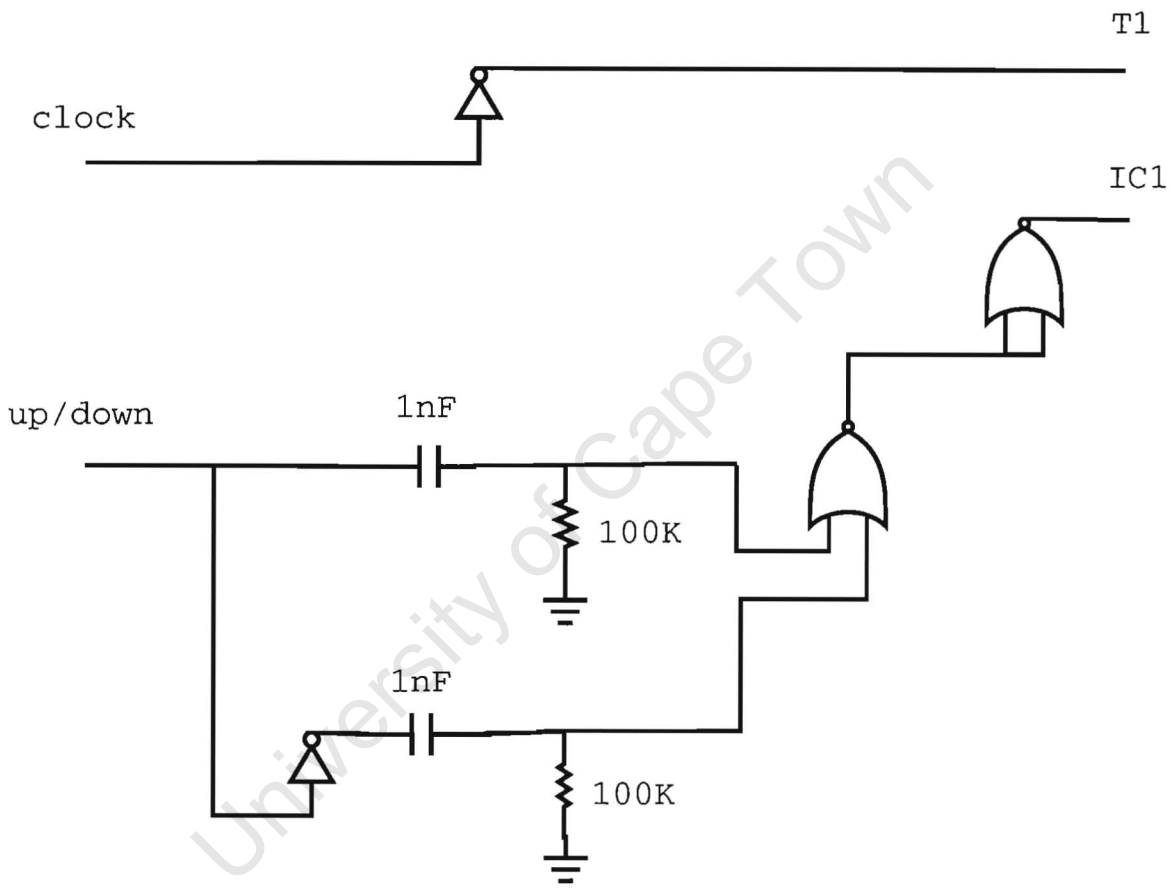

Figure 8.9: Circuit diagram of encoder to microcontroller interface

Chapter 9

Appendix C

9.1 Interface specification document

This document discusses how the odometer is interfaced to other sensors, database and processors included in the Geographical Recording and Processing System.

9.1.1 Hardware interface

This section discusses how the odometer hardware interfaces with other devices.

Encoder connectors

There are two EN3PF 8 ways waterproof coaxial female connectors (labeled Encoder 1 and 2) mounted on the side of the polycarbonate box. Two EN3CM 8 ways male connectors interface the odometer to two optical shaft encoders. This encoders could be attached to a wheel on the winch or the grasshopper wheel if used in the Kleinzee Trial Borehole Radar Survey.

In this project only the female connector labelled Encoder 1 was used during the tests. The 8 way male connector interfaces an optical shaft encoder mounted to a wheel on a stand.

Power supply connector

An EN3PF 2 ways waterproof coaxial female connector (labeled power) is mounted on the side of the box. A cable with an EN3CM 2 ways male connector interfaces a 12V DC power supply from a battery or main supply.

RS232 connector

An FDE9S DB9 connector is mounted on the side of the box. This is used to interface a PC to the odometer via an RS232 cable.

Ethernet connector

A standard RJ45 splash proof socket is mounted on the side of the box. This provides an Ethernet connection to the odometer.

LED and Micro-switch connectors

There are two EN3PF 2 ways waterproof female connectors (labeled LED and Micro-switch) mounted on the side of the box. The connectors provide 5V pulses that are used to trigger radar system directly. Their pulse width is 20ms.

LEDs and reset button

On top of the box are LEDs and a reset button. The LEDs labeled LINK and LAN flicker when there is a network connection to the odometer. The LED labeled Power indicates that there is a power supply to the odometer while the one labeled Interval flashes at fixed interval set when selecting the mode of operation. LEDs L1 to L4 are used during debugging. The reset button resets the odometer.

Optical fibre connector

A HFBR 1414T optical fibre transmitter is mounted on the side of the box. It can attain data rates of up to 160 megabaud. This is used to provide an optical fibre connection when the radar is trigger directly.

9.1.2 Software interface

This section explains how the software implemented on the odometer communicates with other devices.

Web page interface

A web page is used as a user interface when controlling or monitoring the odometer. When odometer is connected to the network, the web page can be uploaded from a PC connected to the network by entering the IP address 137.158.137.7 on a web browser.

On the web page there is an option of entering the interval at which the radar makes a survey. The diameter of the encoder wheel used can also be set.

The mode of operation are discussed below.

Transfer to remote data storage

This option is used to upload data from the odometer to the PC once the data has been processed. A maximum of 45 timestamped data values can be transfered at once.

Read data stored on board

Data stored on odometer during the absence of a network connection is transfered to PC using this option. A file with a maximum of 45 timestamped data values is transfered at once. A file stored on odometer can be uploaded by entering its file number.

Position of radar at this instance

This mode is used to determine position of the radar system at any instance. It is displayed in centimetres.

The following modes trigger the radar when it has moved the preset interval. The network can be disconnected once a mode is selected.

Trigger radar via optical fibre

In this mode an optical fibre pulse is transmitted to radar system.

Trigger radar by LED

In this mode an LED labelled Interval flashes and a 5V pulse is transfered through the connector labelled LED. Both actions occur simultaneously.

Trigger radar by micro-switch or buzzer

For this option, a buzzer is pulsed and a pulse transfered through the connector labeled micro-switch. Both actions also occur simultaneously.

To initiate any of the above modes a start button must be pressed.

9.1.3 FTP transfer

There are 168 files that can be stored on board. Transferring one file at a time would be time consuming. FTP can be used to transfer all the files at once. The user name is odometer and password is odo when logging into the odometer.

9.1.4 RS232 interface

Data can be transfered serially between the odometer and PC. Any terminal on the PC that utilises the ymodem protocol can be used.

In this project the odometer web page, IP and MAC address are transfered to the odometer using a window based HyperTerminal package.