

Project Report:
Automated Remote Sensor and Control System for Water Pumps

Submitted as partial fulfilment of the requirements of Project ELO 320 in accordance with Project EPR 400 in the Department of Electrical, Electronic and Computer Engineering University of Pretoria

November 2012

Chief Lecturer Prof S. Sinha

Teaching Assistants (TAs) [redacted]

Group 2

Initials and Surname	Student Number
F Botha	29042951
LC Short	[redacted]
JS van der Merwe	[redacted]
M van Rooyen	[redacted]

Copy 1 of 5

Summary

The end product that we as a team designed and implemented is an automated system that can be installed at farms where the dam, a water tank and some areas for irrigation are far away from each other. This system composes of four different subsystems. There is a user interface that enables the user to give personalised inputs to the system and a main controller that is the “brain” of the whole systems giving instructions to every other subsystem. Then there is the tank flow and level sensor system that informs the main controller when the tank is full or empty and also what the flow rate of the water entering the tank. Then last but not least there is the sprinkler controller system that receives instructions from the main controller to switch between different areas for irrigation and to turn on the sprinkler pump when necessary.

We designed our product to replace the conventional wired systems on industrial and agricultural levels were wires being laid are impractical because of long distances and hazardous environments between different subsystems. Our project explicitly focuses on water supply systems where a water pump is several hundred meters away from a tank that it is supposed to fill. The cost to install the wires or to replace them will be extremely high and there will also be dissipation losses in wires stretching over 500 meters.

This is where our project takes over, because we use wireless communication in the form of XBee transceivers that replaces the wires between subsystems. It is more reliable over long distances up to 1.5 km and multiple Xbees can be added to the existing system.

The four sections of our project, being the user interface and main controller in the proximity of the water pump near the dam, the tank flow and level sensor system situated at the water tank and the sprinkler controller system near the areas of irrigation was designed, built and tested together.

This report is supplemented with a copy of the approved Project Proposal (Appendix 1) and a technical documentation section (Appendix 2). The latter provides more detail on the software that was used in the experiments, including program listings, a user guide and circuit designs. This section appears on the CD that accompanies this report.
This document has been proofread by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.
My language editor was_____.

_____ _____
Language editor signature Date

I,_____ hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else. Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

_____ _____
F Botha Date

I,_____ hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else. Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

_____ _____
LC Short Date

I,_____ hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else. Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

_____ _____
JS van der Merwe Date

I,_____ hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else. Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work.

_____ _____

M van Rooyen Date

Table of Contents

List of Figures....iii

List of Tables....iii

List of Abbreviations....iii

1 Introduction....3

1.1 Identification of the design problem....3

1.2 Formulation of the design problem....3

1.3 Literature study....3

1.3.1 Long Distance Communication....3

1.3.2 Minimalist and Intuitive User Interface Design Guidelines for Consumer Electronics Devices....3

1.3.3 Controller....3

1.3.4 Level and flow sensors....3

1.3.5 Sprinkler Controller System....3

2 Method....3

2.1 Approach....3

2.1.1 User interface....3

2.1.2 Controller....3

2.1.3 Level and flow sensors....3

2.1.4 Sprinkler system....3

2.1.5 XBee communication....3

2.2 Detailed design and implementation....3

2.2.1 User interface....3

2.2.2 Controller....3

2.2.3 Level and flow sensors....3

2.2.4 Sprinkler system....3

2.2.5 XBee communication....3

3 Results....3

3.1 Summary of Results achieved....3

3.2 Experiment 1: User interface....3

3.2.1 Experimental work....3

3.2.2 Results and observations....3
3.3 Experiment 2: Controller....3
3.3.1 Experiment 2.1....3
3.3.2 Experiment 2.2....3
3.3.3 Experiment 2.3....3
3.4 Experiment 3: Tank flow and level sensors....3
3.4.1 Experiment 3.1....3
3.4.2 Experiment 3.2....3
3.5 Experiment 4: Sprinkler system....3
3.5.1 Experiment 4.1....3
3.5.2 Experiment 4.3....3
3.6 Experiment 5: XBee communication....3
3.7 Experiment 6: Integration....3
4 Discussion....3
4.1 Interpretation of results....3
4.2 Aspects to be improved....3
4.3 Strong points....3
4.4 Under which circumstances will the current system fail?....3
4.5 Design ergonomics....3
4.6 Health and safety aspects of the design....3
4.6.1 I International protection (IP) ratings....3
4.7 Social and legal impact and benefits of the design....3
4.8 Environmental impact and benefits of the design....3
5 Conclusion....3
5.1 Summary of the work....3
5.2 Summary of the observations and findings....3
5.3 Suggestions for future work....3
6 References....3
7 Appendixes....3
7.1 Project protocol....3
7.2 Technical documentation and CD....3

List of Figures

Figure 1: Menu navigation....3
Figure 2: Main controller high level flow diagram....3
Figure 3: Two different placement options for pressure sensor (opening represents the 2nd port of the pressure sensor) A and B....3
Figure 4: Two different paddlewheel flow meter options (red dot represents magnet embedded in paddle) A and B....3
Figure 5: FOCA developer board....3
Figure 6: LCD memory....3
Figure 7: Interrupt Service Routine flow diagram....3
Figure 8: The main program flow diagram....3
Figure 9: PCB layout of user interface....3
Figure 10: Main controller program flow diagram 19
Figure 11: Flow diagram to get data 3
Figure 12: Controller circuit....3
Figure 13: PCB layout of main controller....3
Figure 14: Offset compensation resistor for pressure sensor....3
Figure 15: Two amplifier instrumentation amplifier....3
Figure 16: PSpice simulations for instrumentation amplifier....3
Figure 17: Flow chart of code to process sensor data on PIC18F45K20....3
Figure 18: Circuit to switch 12V with 3.3V....3
Figure 19: SCS Circuit diagram....3
Figure 20: PCB design for SCS....3
Figure 21: Block diagram for input/output....3
Figure 22: Sprinkler controller code flow diagram....3
Figure 23: XBee UART communication....3
Figure 24: Flow diagram for wireless communication between subsystems....3
Figure 25: Diagram of how the XBees are connected to the PIC's UART ports....3
Figure 26: Table of IP-ratings from SANS 10142-1....3

List of Tables

Table 1: Subroutines written for the LCD Module....3
Table 2: Data table 1....3
Table 3: Data table 2....3
Table 4: Data table 3....3
Table 5: UART communication codes....3
Table 6: Component values for required voltage outputs....3
Table 7: Results of sensor performance in simulated extreme temperature conditions....3

List of Abbreviations

LDC	-	Long Distance Communication
GSM	-	Global System for Mobile Communication
SCS	-	Sprinkler Control System
RTCC	-	Real Time Clock/ Calendar
UART	-	Universal Asynchronous Receiver/Transmitter
RX	-	Receive
TX	-	Transmit
PCB	-	Printed Circuit Board
ADC	-	Analogue to Digital Conversion
I/O	-	Input or Output ports
KCL	-	Kirchhoff's Current Law
CCP	-	Capture/Compare/Pulse width modulation
ISR	-	Interrupt Service Routine
LCD	-	Liquid Crystal Display
EEPROM	-	Electronically Erasable and Programmable Read Only Memory
IP-Rating	-	International Protection Ratings
UI	-	User Interface

1 Introduction

1.1 Identification of the design problem

In many industrial and agricultural installations, conventionally wired control circuits are impractical either due to hazardous environments, or extreme distances between sensors, and the devices they give feedback to. This is particularly true in the case of water supply systems, where a pump is several hundred meters away from the tank that it is filling. In this case it would be too costly to install wires between the pump and the tank because of the price of copper wires. There will also be losses in any wire of 500 meters or more and in case of a veld fire the wires can get damaged. At a tank like this, there could also be sprinklers that need to be turned on or off in order to irrigate crops. There could be various different valves from the tank that need water supply at different times. In order to change the direction where the water needs to go, someone needs to physically go to change the valves.

1.2 Formulation of the design problem

In order to make life easier and more cost effective, an automated wireless system is needed. It must be a system that can turn off the water pump when the tank is full and a system that can switch between different sprinklers. It will do this all by itself and in this way remove human error and free up time for farm managers and workers. Such a system will need sensors to tell the level of a tank, some valves in order to open water to different areas and a controller to control a pump and some valves. These subsystems should communicate with each other wirelessly to overcome the problems associated with copper wires. There should also be a user interface to enable the user to add personal inputs to the system and to display the outputs.

1.3 Literature study

1.3.1 Long Distance Communication

In our group it was decided that there had two types of LDC systems we had to choose from. The first LDC system was GSM. For this system a phone with a SIM card is needed so that messages can be received on the phone. The team declined the idea because of costs, the user will be liable to buy airtime for the GSM to work and because for our project's application it is too complex. All that the project needed was a LDC wireless link.(Yahoo Voices)

After some research the team came upon the XBee/ XBee PRO RF transceivers. It uses the IEEE 802.15.4 standard network; it is low cost, requires low power and is reliable in sending and receiving data between devices. It operates in the 2.4 GHz frequency band and the devices are interchangeable with each other (XBee Datasheet).

1.3.2 Minimalist and Intuitive User Interface Design Guidelines for Consumer Electronics Devices

Duo the large number of electronic devices available there is increasingly different approaches to user interfaces. Often conflicting approaches are present even in different models from the same manufacturer. This can be very demanding on end users as they are faced with the need to constantly learn how to use new and different devices.

UI should be designed to be of an intuitive nature reducing the number of times consultation of the user manual is necessary. Often the manufacturers need to stand out from the rest adds to the confusion.

The UI is a key factor in the success of a product, especially in the competitive market of today when a well thought out and more natural UI can set a vendor apart from the competition.

One needs to take into account that not all users will be engineers or technically trained people, thus one should keep the system to an understandable level.

UI's are often neglected in the design of consumer electronics as it is not a value that can be quoted on a speck sheet like many of the other features the convince consumers to buy.

There are two aspects of a UI that need to be taken into consideration, namely the physical and the non physical. The physical is the actual hardware of UI like the buttons, switches, wheels, knobs, stylus, touch screen, keypad, microphone, monitor, speaker, digital, camera, video recorder, etc. The non physical refers to the things that happen in software, the images and sounds used to interact with the user. An important aspect of this is how good the device is at prompting or leading the user towards their end goal.

Another aspect of UI's is Aesthetics and emotions. Although the device must appeal to the user in must not be at the cost of functionality. Convenience is a very important aspect of a UI. Is the device easy to use and non-straining in the body? Is it easy to see hear and touch.

A key factor is the operating mechanism. A common operation shared by many appliances appliance such as switching on and off, menu operations and the navigation interface such as four way navigation, and alphanumeric input and the manipulation of displayed data. The other side is device specific UI's.

The some guidelines can be given for UI design. The first is to keep it minimum. Don't add unnecessary shortcuts and menu paths. Keep a common methodology to accessing the different functions of a device, and keep the most used function close at hand. Avoid giving one button more the two functions. Stick to few and standardised symbols. Avoid cluttering and unnecessary information.

The second element is intuitiveness; this comes down to looking at the world around you and developing UI such that it resembles understood and familiar ways of doing things. Arranging elements of the UI by their functions and relatedness aids an intuitive understanding.
Consistency is very important as it speeds up the users ability to learn how to use a system. (Kang & Kim:2007)

1.3.3 Controller

The main controller will be the centre of the system.

A PIC should be chosen for the application it will be used. Most regular PIC users recommend using the smallest PIC that can perform the tasks needed in a project. (Lazaridis, 2008) The team decided to make use of the PIC32 for the microcontroller, because we could get free samples from the University of Pretoria. It only has 28 pins, which is more than enough and most important the PIC32MX220F032 has two UART Ports. The PIC18 that we can sample from the University only has one UART connection.(Microchip, 2012)

The PIC32 can be programmed in either C or Assembler, but in order to simplify the logic and character handling the controller must do the C language was chosen.The peripheral library is also available to the 32 bit PICs.

In order to implement a Real Time Clock and Calendar on a PIC32, an external 32.768 KHz crystal is needed. (Digi International, 2009) The crystal needs to be low powered for the PIC to be able to drive the clock. This specific frequency is used because it will take one second to fill a 16-bit register. The PIC, however, generates a half second clock by looking at the second last bit in the 16-bit register. With this new half second clock the PIC can increment the seconds register on every rising edge of the half second clock and real time is formed. (Microchip, 2008)

1.3.4 Level and flow sensors

1.3.4.1 Level Sensors

Different methods of continuous fluid level sensing were looked at, taking into consideration that the fluid whose level to be determined is water. There were 3 different solutions that were to be chosen from. These three solutions are the eTape Liquid Level Sensor, based on hydrostatic pressure principle which changes the resistance of the eTape, secondly making use of ultrasonic transmitter and receiver to continuously determine the level of the liquid, and lastly using a pressure sensor, of the differential type, to measure the hydrostatic pressure. (Hambrice K., Hopper H., 2004)

The pressure sensor was chosen because of its low cost in comparison to the other two options and it proves to be a reliable and flexible, in the sense that only the length of pipe and a pressure sensor that can handle the pressure are needed to measure the water level as will be seen later. Lastly on this choice is that it is more suited to the application of this project with respect to the fact that external obstructions, as will be seen with the other two sensors, do not affect measurements. An integrated temperature compensated option exists which is an advantage. More specifically the MPX10DP pressure sensor was chosen.
(National Instruments, 2008)

The eTape was rejected because of cost and it is manufactured in only certain lengths, it should be noted however that custom lengths of the eTape can be ordered, and also because if anything except the water is in contact with the measurement surface, the measurements can be disrupted. For example in an open top water tank where things can fall in the tank it can be seen how this can cause problems,. One advantage is that is has integrated temperature compensation.
(Milone Technologies, 2012)

The ultrasonic measurement method was not chosen because of unnecessary complexity for the purpose of this project as well as the fact that turbulence, when water is being pumped back into the tank, moisture, and any obstructions that ultrasonic pulse encounters affect the accuracy of the measurement. Lastly temperature compensation is required.

1.3.4.2 Flow Sensors

There exists a large variety of different flow measurement methods to choose from each with certain advantages and disadvantages. It must be kept in mind that of greater importance than the actual flow rate is the detection of low rates of flow which could cause the tank to overflow. The methods assessed were the turbine method, paddlewheel method, and ultrasonic flow meters.
(The Engineering Toolbox, 2011)

The ultrasonic flow measures flow rate by measuring the time it takes for the ultrasonic pulse to propagate against and in the same direction through the water flowing in the pipe. What that results in is equal transit times, the difference is zero, at no flow and when flow occurs the pulse travelling against the flow of water is slower than the one travelling in the direction of the flow of water resulting in a non zero difference of the transit times. This is a nonintrusive method of measurement. This method fails to deliver accurate results when the water is very dirty and the pulses cannot reach the receiver which means the limiting factor here is the ability of the ultrasonic pulses to travel through the water. Air bubbles and solid particles can also affect measurements.

The turbine method like the paddlewheel method makes use of the Hall Effect. A magnet, or magnets if the turbine or paddlewheel will be unbalanced, is embedded in the blades/paddles of the turbine/paddlewheel which generate square wave output to be processed by a microcontroller when it rotates past the Hall Effect sensor. The turbine flow meter is accurate for higher flow rates it however becomes inaccurate at low flow rates when force of the water on the turbine blades is unable to cause a rotation. On the other hand the paddlewheel is more accurate at low flow rates but is limited in the range over, typically the higher flow rate range, which it remains accurate. For both these methods the speed of rotation is proportional to the volumetric flow rate. They are also both invasive measurement methods. It should be noted however that depending on the material used in the design and manufacture of a paddlewheel flow meter the accuracy at higher flow rates can be maintained. (Cartier, 2005)

The paddle wheel flow meter was chosen for the purpose of this project because of its relatively low cost, reliability at low flow rates, and ease of maintenance. The reason for the paddle wheel flow meter is able to detect lower rates of flow is because the axis about which it rotates is perpendicular to the flow of the water whereas in the turbine configuration it is parallel. A paddlewheel flow meter was constructed. (Blue-White, 2012)

1.3.4.3 Microcontroller for Sensors

As mentioned before only microcontrollers with UART pins are suitable the purpose of this project. Microchips PICs were looked at. For the sensor application the microcontroller also had to have an analogue to digital module as well as a capture capability, part of the capture/compare/pulse width modulation module. There are many different options to choose from with a 20 pin, 28 pin, and 40 pin package of the PIC16F687, PIC16F883, and PIC18F45K20 all meeting the minimum requirements. The two PIC16s can operate from a maximum supply voltage of 5.5 V whereas the PIC18 can operate for a maximum supply voltage of 3.6 V. The three PICs cost more a less the same so cost was not a factor in deciding which one to use. If there was some or other constraint on the size then one of the PIC16s could be used instead but either one of the three would perform its given task in this application. Because the PIC18 was readily available it was the chosen option.
(Microchip, 2008)

1.3.5 Sprinkler Controller System

The SCS is the last link of the system. This is where all the water gets used to irrigate different areas.

For this system to be automated some valves were needed that can opened electronically. Research was done on different electronic valves that would not be too costly and would work from 12 Volt seeing that it is the power supply we use for the project.

The valve that was chosen came from the website of Netram Technologies but the manufacturer is Aqua Tech Trading Corporation Limited. It is called an 12V solenoid valve (model AQT15SP) that uses the 12V supply to energize the solenoid and then opening it so that water can pass through it. According to the Datasheet the coil resistance is 4.75KΩ meaning it won't pull to much current.(Netram Technologies, 2012)

Research was also done on types of pumps that will be suitable as sprinkler pumps. They must give enough pressure to open the valves completely and to let the sprinklers cover a large enough area when they irrigate. Any 220Vac pump can be connected to the circuit because a 220V 16A relay is used to switch the motor. But if a bigger motor is required the relay can be replaced with a more robust one.

Some Companies that sells pumps is Wikkies Pompe in Pretoria and they have a wide range of pumps for irrigation purposes. One pump that stood out is the POWXG9402 series submersible pump that has a flow rate of 7500 litres/hour and will be suitable for the project.(Wikies Pompe, 2012)

A pump of a calibre like that cost a lot and is not financially viable for this project, so for this project only water feature pumps (already installed at home) will be used for the demo. It will be sufficient enough for demonstration purposes.

For the controlling part of the SCS a microcontroller should be chosen that will be applicable for certain usages. A decision had to be made between the PIC18F45K20 and the PIC32MX220F032B microcontrollers from Microchip. Both microcontrollers was sampled by the University of Pretoria and given to EMK 310 students for free, but the PIC32MX220F032B was chosen for controlling the sprinkler system because it is smaller in size (28 pin instead of 40) and gives the option to make the system as small as possible. The PIC32 is also more user friendly than the PIC18 because the C programming can be easily incorporated on the PIC32 and its UART communication Ports can easily be setup.(Lazaridis, 2008)

2 Method

2.1 Approach

In order to achieve wireless communication between the different subsystems of our project, we decided to use XBee PROs. The XBee PRO modules can communicate with each other over a distance of up to 1.6 kilometres. (XBee datasheet) This distance is more than enough for the requirements of our project. Next we needed to send information to the XBees in order to communicate with other subsystems. The XBees can receive and send data over its embedded UART pins.

To implement the different controllers at each subsystem, PICs will be used. PIC18s will be used at the user interface and the sensor controller and PIC32s at the main controller and the sprinkler controller. The PICs have UART communication ports and can thus communicate with the XBees. Next we needed a set of UART codes to enable the different subsystems to communicate with each other.

Time also needs to be incorporated to enable the user to set different switch on times for different sprinklers. Time will be implemented on the main controller and also displayed on the user interface.

The main controller will communicate with for example the SCS to give it certain instructions (code words) that will be converted on the PIC32 and then drive certain outputs.

2.1.1 User interface

To set the parameters governing the system the user needs to give it some form of input. There are many possible ways of doing this raging from a simple console to a graphical user interface on a computer enabled for remote configuration. For this project it was designed so that we would not use remote configuration due to the data costs associated.

When one is faced with an electronic system one is often shielded to how it actually functions, one is simply faced with the user interface that enables one to operate the system. The user interface is thus of great importance since the ease of use and effectiveness of a system is to a large part determined by it. A difficult to use interface is often the downfall of many technically sound systems.

The ideal is to have a system which does not require connection to a pc or other device to for configuration, and does not require the user to have technical experience. A common trend in modern devices is the use of a graphical display (such as a LCD display) and a hierarchical menu structure. This menu is usually navigated by arrows and an enter buttons type structure as shown. A numerical pad is also often included for numerical input.

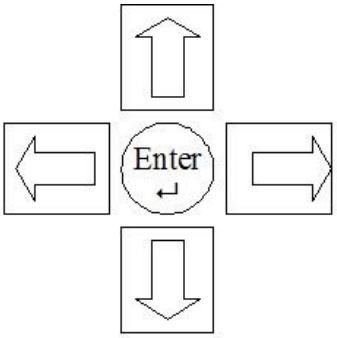


Figure 1: Menu navigation

Such a menu structure enables the user to intuitively understand the functionality of the system and make changes to the operational parameters. It removes the need for the user to constantly refer to a manual to decipher codes produced by the system.

Often when a system is too complicated to configure it is set up by an installation technetium and never altered again. The intention was to give the user the freedom to alter the system settings as needed without the assistance of a technician. Self installation would also be encouraged.

It was decided that such an interface would be well suited to the system. A LCD Display together with a numerical display was used. The menu navigation buttons were combined with the numerical keypad as a cost saving measure.

The operational parameters also needed to be stored into a non volatile memory and be readily available to the main controller.

A PIC18F45K20 microcontroller was chosen to implement the system due to the large number of physically implemented I/O ports which would be needed to implement the keypad and to drive the LCD Display. This microcontroller has 256 bits of EEPROM suitable for storing the operational parameters of the system. The Microprocessor was programmed using a Microchip native PIC18 assembly language. (Microchip,2010)

The coding was done in such a way that the menu structure would be entered into data table. The use of such a data structure strengthens the choice of a PIC18 Microcontroller as it has good support for Data tables entered into program memory. The data table also served to associate EEPROM addresses to operational parameters and for defining communication codes for the controller to retrieve and alter operational parameters. This was done such that alterations in the menu structure would not require recoding the entire program easing the integration of the User interface into the system. This approach allowed the table to be created in a spread sheet program and imported into the code, greatly easing the definition of the data table and future alterations and error corrections.

The User interface would be joined to the rest of the system by means of a serial port using the on board UART module of the microcontroller, adhering to the RS232 standard.

2.1.2 Controller

A PIC32 was chosen for the main controller because of the UART ports and to be able to use the PIC32 Peripheral Library. The Peripheral Library is used to make it easier to work with the real time clock and to be able to use the UART functions. Although the controller could have been implemented on a PIC18 or a PIC24 with two UART ports and a RTCC, the PIC32MX220F032B is available for students at the University of Pretoria and will be used to implement the main controller. The team also has previous experience with this PIC32, which will make it easier to implement it.

The first UART port will be connected to the XBee and the second to the user interface. The main controller and the user interface will be next to each other.

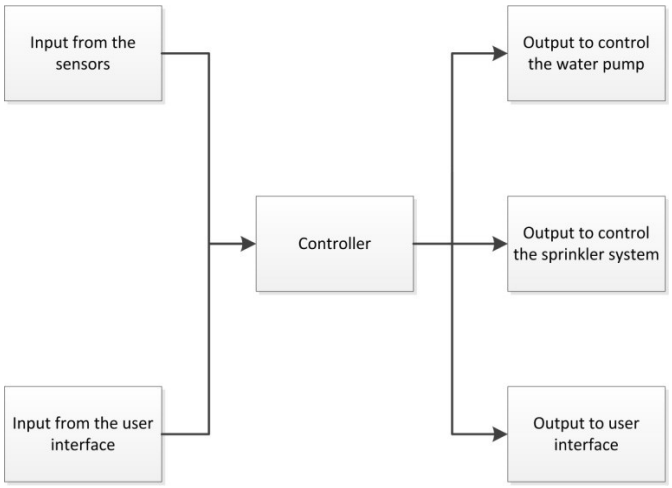


Figure 2: Main controller high level flow diagram

Through the wireless XBee modules, the main controller will receive inputs from the tank sensor system and give outputs to the SCS but also receive inputs from it. It will also get inputs and send outputs from and to the user interface as in Figure 1. All this communication will be according to the UART codes set up in the beginning of the project.

The PIC was programmed in C using MPLAB X. MPLAB X is much more user friendly than MPLAB 8. It is also much easier to debug in MPLAB X. The software for both these programs is available on the microchip website.(Microchip, 2012)The PIC was programmed using a Pic-kit 3.

The code for the controller is entirely new.

2.1.3 Level and flow sensors

A 12 V power supply (a car battery) is used to power the sensors, operational amplifiers, and PIC18 after it has been regulated to 3.2 V and 5.3 V. From the literature study done the clear choice, as stated there, was to make use of a pressure sensor. For the purpose of this project the MPX10DP, maximum pressure rating of 10 kPa (roughly 1 m depth of water) was used but in reality a 40 kPa (roughly 4 m depth of water) rated sensor can be chosen. The differential output signal had to be amplified as it is in the mV range and there was an offset that had to be compensated for. Other alternatives such as the MPX5010DP were assessed which already has its offset compensated for but it has a maximum output of 4.5 V which exceeds the allowable input of the PIC18F45K20. The output of the MPX10DP will be fed through an instrumentation amplifier, whose performance will be simulated in PSpice, before it arrives at the ADC of the PIC. Instrumentation amplifier ICs exist such as the INA122 from Texas Instruments. However the instrumentation amplifier was implemented using operational amplifier ICs.

Two different methods were analysed on how and where to place the sensor. Keeping in mind that the sensor chosen is for use only in dry media, such as air, as anything else can potentially damage the piezoelectric transducer inside. That meant that there had to be a column of air between the piezoelectric transducer of the sensor and the water. Shown below are the two methods considered.

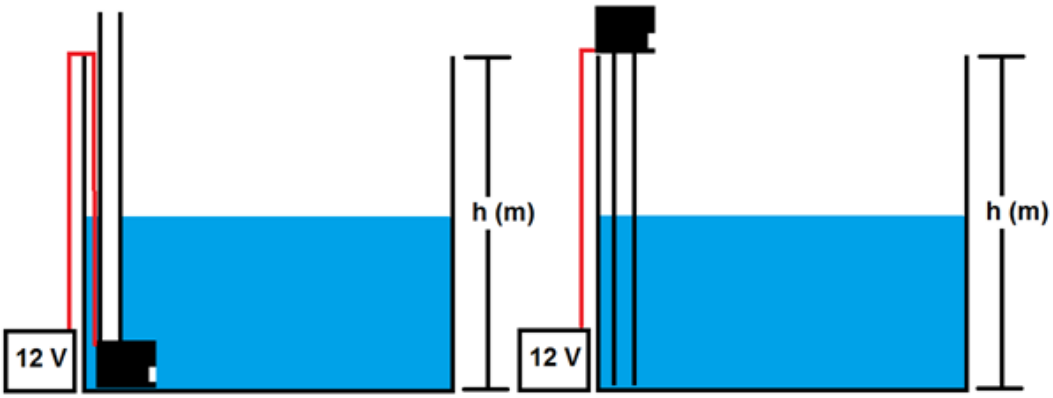


Figure 3: Two different placement options for pressure sensor (opening represents the 2nd port of the pressure sensor) A and B

In figure 2A above where the enclosure housing the pressure sensor and other electronics, voltage regulators and operational amplifiers and PIC18, is submersed in the water and placed at the bottom of the tank is not a suitable option in that if the electronics fail the tank has to be emptied to repair something as well as having the power supply wiring run through the water, can be very dangerous, means that it will have to be well sealed to be leak proof. Another thing that could become a problem is that the enclosure should then be able to withstand the pressure at the bottom of the tank. For the aforementioned reasons option A was not used, instead option B was used.

In figure 2B the enclosure housing the electronics is mounted onto the brim of the tank which makes setup and maintenance a whole lot easier. There is also always a column of air in the pipe separating the water and sensor; it is this air that is pressurised. In this setup the power supply cable does not have to run through water and the requirements for the enclosure are less stringent to ensure the electronics are kept in working condition.
Next the different paddle wheel flow meter options are assessed, as mentioned in the literature study this was the best option. Two different configurations are looked at, it should be noted that these figures are just to illustrate the concept of the actual design that will be implemented.

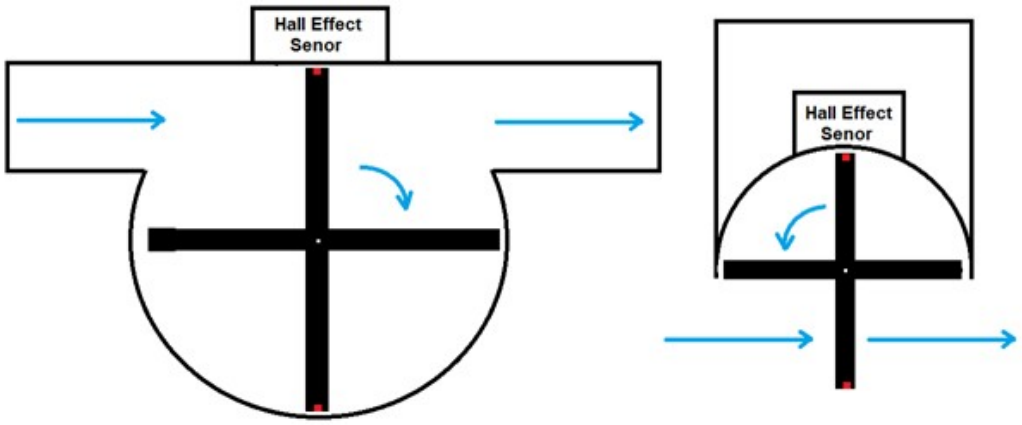


Figure 4: Two different paddlewheel flow meter options (red dot represents magnet embedded in paddle) A and B

As can be seen both methods are intrusive in that they come into contact with the water to measure the rate of flow. The method represented by figure 3A is the slightly less desirable option in that it cannot measure lower rates of flow as accurately as the method shown by figure 3B. The design of figure 3B is chosen as it is easier to setup and maintain, in that it is just inserted into the pipe where a opening is made for it whereas the other option must be inserted between two sections of pipe, with the only requirement being that it must be positioned upright and the paddle wheel axis of rotation must be in the centre of the pipe where in the water is flowing. Lastly option A can become problematic in that particles, such as dirt and such, can accumulate in the bottom of the device.

The Hall Effect sensor used was the OHN3020U Unipolar Hall Effect switch which can operate from 4.5 – 24 V with an output, a high level, being equal to the supplied voltage meaning that the output would be have to be fed through a voltage divider to get it at a level acceptable by the PIC. An alternative was the A3290 from Allegro which comes in the latch configuration, although not a problem when using two magnets each with opposite polarity embedded in the paddle, when one magnet is used the device cannot become unlatched as the unlatching polarity is not present. All things considered most of the Hall Effect sensors operate the same, some with lower operating voltages, such as the TLI4906 from Infineon which can operate from as low as 2.7 V, which are harder to come by and that being the only reason why the OHN3020U is chosen. The output is a square wave with vary duty cycle depending how fast the paddle wheel is spinning.

The PIC will make use of its ADC to determine the water level and send one of the predetermined communication codes over the XBee to the controller whenever there is a change in water level of 12.5 %. The PIC can also accept a communication code that requests the water level or the flow rate. The rate of flow is determined by capturing every falling edge of the output from the Hall Effect sensor and using the period to determine if there is flow or not and then send the applicable communication code corresponding to the flow rate to the controller, if requested to do so. Coding was done in assembler.

2.1.4 Sprinkler system

The reason why the PIC32 was chosen for the controlling of the SCS was of its UART ports that can easily be setup using the peripheral library functions. The output PORTS was also easy to setup. Code for the PIC32 was implemented by using MPLABX with its C-language compiler. The PicKit3 connects to the laptop using a USB cable. The UART1 RX and TX pins of the PIC32 are connected to the XBee transceiver's Data-Out and Data-In pins respectively that acts as the communication link between SCS and the main controller.

As mentioned in the literature study the AQT 15 SP 12V solenoid valve were used to switch between different areas for irrigation but in order to switch 12V to the valve a transistor circuit is needed because the PIC32 output pins only give out 3.3V and that is way too little to switch the valve.

To drive the 220 Vac sprinkler pump a 220Vac 16A relay were used. The live wire of the sprinkler pump will be split so that it can be connected over the relay making the option possible to switch the pump on and off by energizing the relay coil with 12V. Also to energize the relay to put the 220Vac through, 12V was needed in the same way the valves were switched.

The whole circuit was etched on a PCB so that the circuit will look as neat as possible.

2.1.5 XBee communication

A developer board called FOCA (FT232RL) was purchased to establish communication between the laptop and the XBees. It is a USB to serial UART interface and is compatible with X-CTU programming software.(Micro Robotics, 2012)



Figure 5: FOCA developer board

The XBees were programmed using X-CTU software (freely available on the internet), a program designed specifically to interface with the XBees. With this program you can program the XBees by choosing different options listed when you read the XBee's information, but you can also use the program as a terminal to send UART codes between the XBees. As well as range tests to help to determine what the transmit and receive range of the XBees. (Digi, 2012)

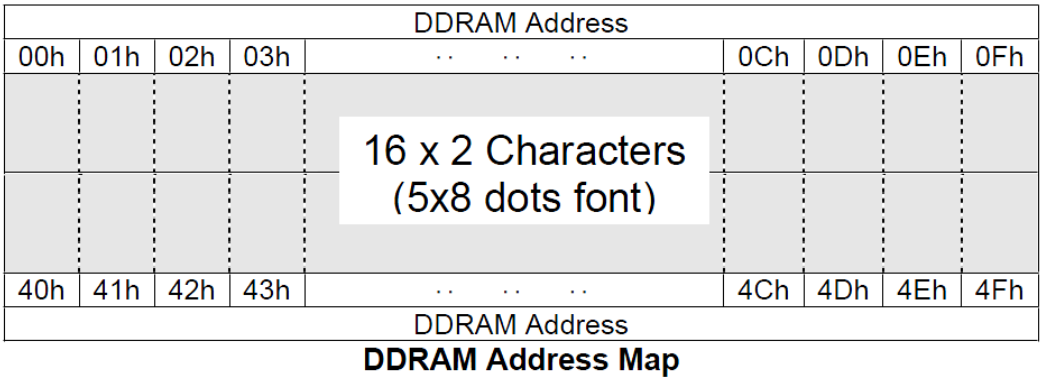
2.2 Detailed design and implementation

2.2.1 User interface

2.2.1.1 LCD Module

A 16 by 2 character LCD module (LMB162ABC) produced by Shenzhen TOPWAY Technology was used. It is controlled by three controls Bit and has an 8-bit data bus for transferring data into the memory. These pins were connected to the microcontroller through buffer chips to convert the 3.3V output of the microcontroller to 5V as required by the LCD module. (Topway,2005)

The LCD module has a 80 character element memory arranged into a 4 by 40 , displaying only a 2 by 16 window. This allows the LCD to be scrolled for messages longer than the viewable window. The memory elements could be addressed directly or accessed in a linear fashion by using commands for moving the cursor. A write to the module moves the cursor automatically.



DDRAM Address Map

Figure 6: LCD memory

Short subroutines were written for the basic operations of the LCD such that it could be used one by a simple call statement in the code. The module would only be accessed in a linear fashion as this avoids having to change from write to address mode.

The functions subroutines written for the LCD Module are summarised in the table below:

Table 1: Subroutines written for the LCD Module

Function	Description	Subroutine label
Setup	Set the LCD into the desired display mode	setup_LCD
Clear	Cleared the LCD memory and Display	clear_LCD
Home	Returned the cursor to the top left corner and the viewing window to the start of the memory	home_LCD
New line	Moves the cursor to the second line on the LCD	new_line_LCD
Shift left	Sets the cursor to move from left to right	shift_a_l_LCD
Shift right	Sets the cursor to move from right to left	shift_a_r_LCD
On	turns the LCD on	on_LCD
Cursor on	Makes the cursor visible	on_c_LCD
Cursor off	Turns the cursor off	off_LCD
Shift cursor left	Shift the cursor left	shift_l_LCD
Shift cursor right	Shift the cursor right	shift_r_LCD
Shift screen left	Shifts the display window to the left	shift_ls_LCD
Shift screen right	Shifts the display window to the right	shift_rs_LCD
Write	Writes the 8 bit character currently in the variable “LCD” to the LCD module in the current cursor position and shifts the cursor on one position.	write_LCD

An extract from the code for the Write subroutine operation follows, all other subroutines are similar.

```
;_____________write_LCD__________
write_LCD
movf LCD,0 ;data
movwf PORTD
nop
bsf PORTC,2 ;RS (select data transfer)
bcf PORTC,3 ;RW (select write mode)
bsf PORTC,4 ;E (just for safety)
bcf PORTC,4 ;E (start write)
bsf PORTC,4 ;E (complete write)
call delay
return;write_LCD
```

2.2.1.2 Data table

The menu structure was defined in the data table that was added to the end of the code implementation. This makes the menu easy to alter and debug. It also makes adapting to changes in the other subsystems easier to deal with as well as easing future improvement. The table was created in a spread sheet program to making it easy to define the menu structures. The first seven elements define the relationships between the menu items. The next three the EEPROM addresses for storing data associated to the memory element. The next three define the maximum limit placed on data entered. The next three define communication codes; for editing data, for requesting data and the returned code respectively. The last two contains the string elements to be displayed on the LCD. An example of the table is shown in appendix A. The first row is repeated here.

Table 2: Data table 1

;Lable		Sub_menu(right)	next(down)	previous(up)	Previos_sub_menu
_start	data	_sprinkler_settings	_start	_start	_start

Table 3: Data table 2

next_in_tabel	Data_1	Data_2	Data_3	Limit_1	Limit_2	Limit_3
_sprinkler_settings	0x01	0x02	0xFF	d'24'	d'60'	0x00

Table 4: Data table 3

Code_Read	Code_Write	Code_return	Display_1st_line (must be odd number of char before <.>)	Display_2nd_line
"ooooo"	"ooooo"	"ooooo"	"Standby > for more ."	"# : #.";

The relationships between the elements are defined by the use of program memory address labels. The labels are found in the first column of Pic18 assembly language to mark locations in memory. When used after opcodes such as bra (break) or goto these labels are replaced by a memory address embedded in the op-code by the compiler. These labels can also be used to store program memory address lists created with the “data” assembler directive for the purpose of lookup tables.

The relationships between menu elements or data table rows are defined as follows.

- The first element in the table (Label) is a label that defines the program memory address for the start of the row.

- Next element is the opcode “data” used to create a data list in this case a row element in the table.

The next five elements in the data table contain address labels that will be converted by the compiler to two byte long program memory addresses pointing to the start of another data table rows.

- Sub_menu(right) points to the row containing the first element in a sub menu. The character “>” will be displayed at the end of the LCD top line If sub menu element exists. Pressing the rightward pointing navigation button will display the addressed element. If there is no sub menu for the current row the address label of the current row is repeated in this element causing it to be redisplayed.
- Next (down) points to the next row in the current menu level. If the current element is the last in the current menu level its address label is repeated here, or alternatively the first address element of the current menu level can be placed here to create a circular menu flow.
- Next (down) points to the previous row in the current menu level. If the current element is the first in the current menu level last its address label is repeated here, or alternatively the last address element of the current menu level can be placed here to create a circular menu flow.
- Previos_sub_menu moves one menu level up, the current element is in the root menu, and its address label is repeated here.
- next_in_tabel points the subroutine used to check the communication codes to the next table row as row length is not fixed.

The line Display_1st_line element contains a string to be displayed on the first line of the LCD the, while the Display_2nd_line element can contain special characters to direct the program to display associated data elements from EEPROM. The character ‘#’ directs the program to format the data as an integer, while the “*<text if true>*<text if false>*” directs a data element of Boolean type. The text between the ‘*’ characters are placed on the display. For example “*on*off*” is replaced with on if the data element is ‘1’ , or off when the data element is ‘0’.
The data table is exported from the spread sheet program to a CSV file that is then opened in a text editor The export process generates additional character that are removed using the replace function of the text editor. Some tabs used for the assembly file need to be added manually. This Text is then copied (using copy and paste) into the assembly file at the location after the directive definition for the origin of the table “org 0x006000”

2.2.1.3 Keypad

A 10 digit keypad is implemented for the user to input data and doubles up as the menu navigation element. It common ground type of keypad used, that is every button is connected to ground when pressed. A scanning technique as described by Huang (2007) is use to detect and de-bounce button presses. Debouncing is a technique used to remove the transients of a mechanical switch that could cause multiple presses to be detected. The

keypad is continually scanned until a press is detected the first time. A 10ms delay is the implemented after which the button is rescanned. if a press is still detected the program continues with a response else it returns to scanning the keypad button for button.

2.2.1.4 Program Flow

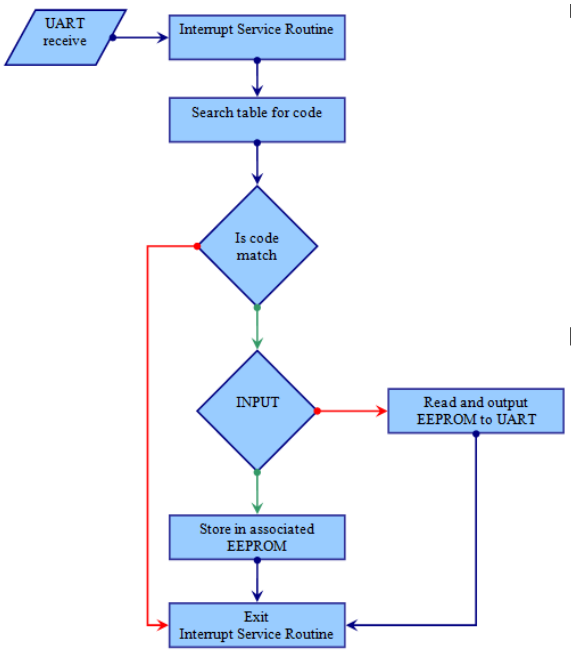


Figure 7: Interrupt Service Routine flow diagram

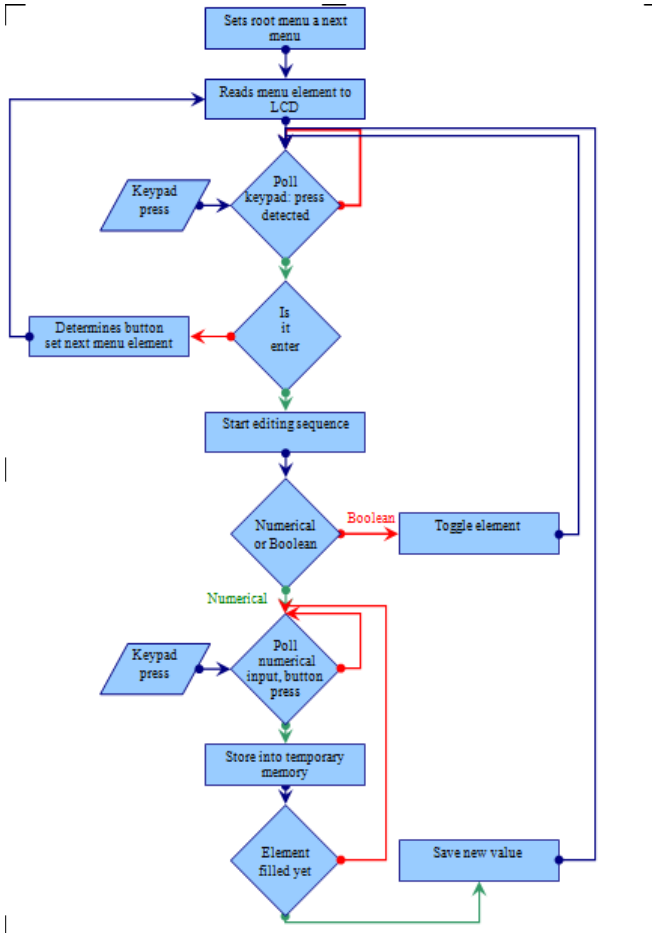


Figure 8: The main program flow diagram

2.2.1.5 Circuit layout

The actual circuit was laid out as follows with headers allowing for the connection of the LCD and keypad. It is supplied from an external 12V supply that is regulated by two voltage (LM317) regulators tuned to operate at 5V and 3.3V. Refer to section2.2.3 for more detail on the regulator. 10kΩ resistors are used as pull up resistors at the keypad.

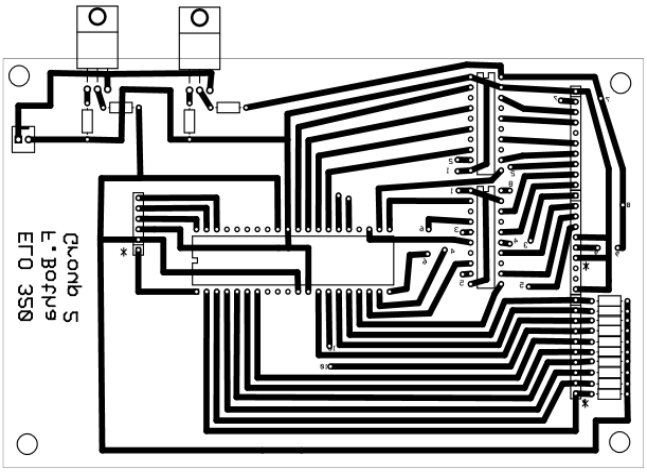


Figure 9: PCB layout of user interface

The following PCB artwork was designed with ExpressPCB for the circuit created by using positive pre-treated photosensitive PCB and developer. Ferric Chloride Liquid was used for etching the copper.

2.2.2 Controller

The code programmed on the PIC32 has the following structure:

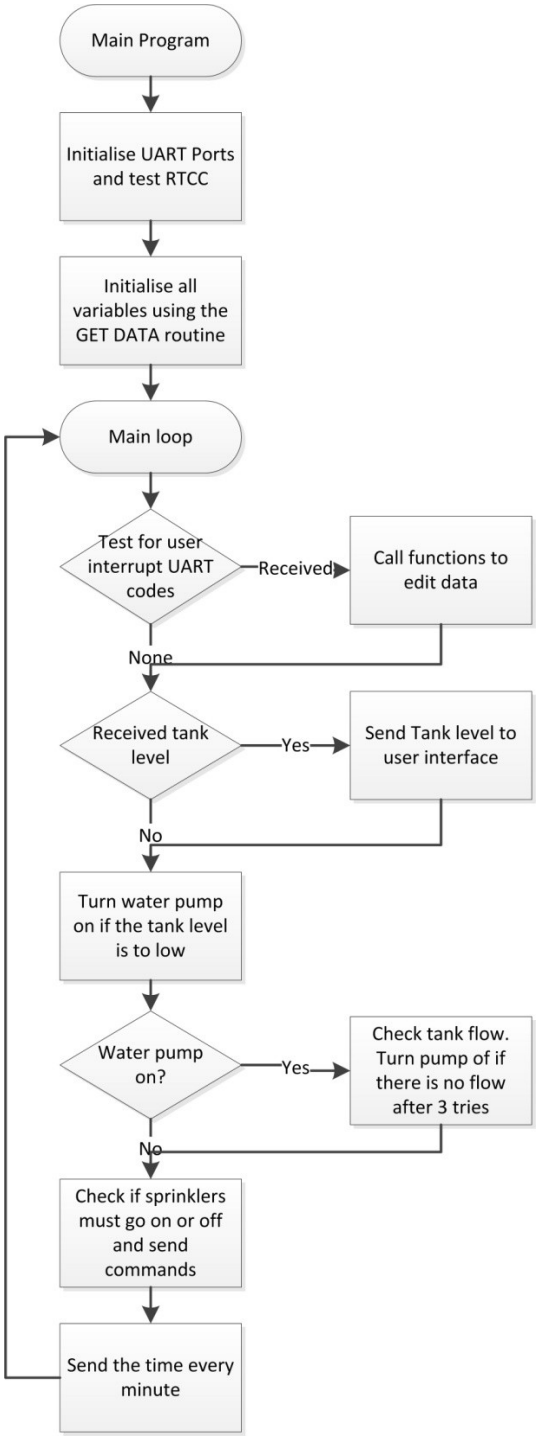


Figure 10: Main controller program flow diagram

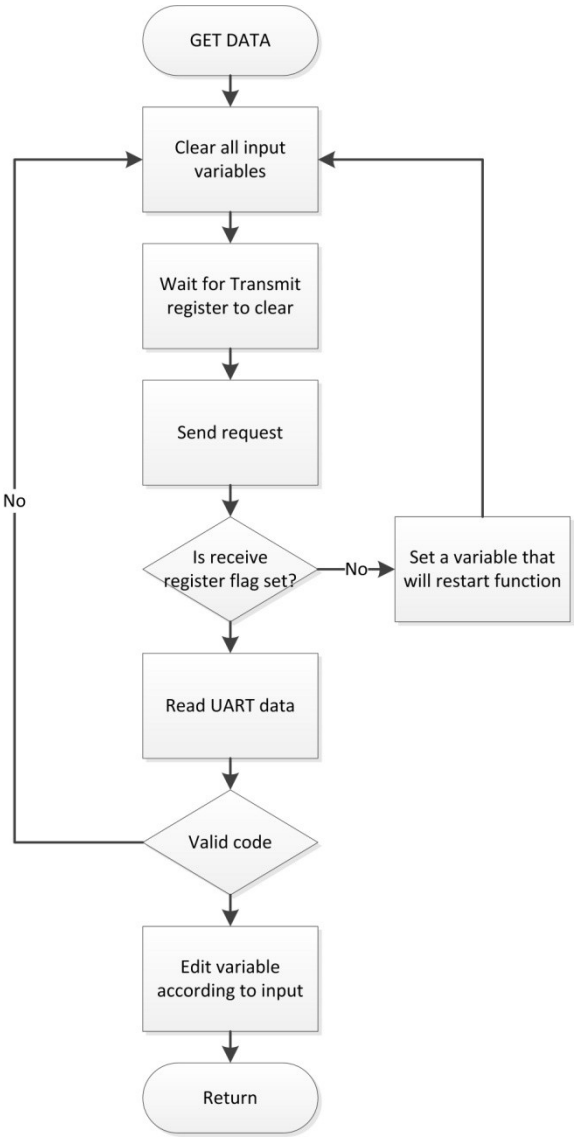


Figure 11: Flow diagram to get data

Apart from the main function in the code, as see in Figure 10, there will be a whole lot of other functions that will be called in the main program. These functions are used to get sent codes and decode replies to set variables.

The routine to get data is show in Figure 11.

The UART communication codes were set to enable the different subsystems to communicate with each other.

Table 5: UART communication codes

Code word	Description	How does it work
TLx	Tank level full	This code is sent to the controller from the tank subsystem to tell the tank level (x can be 0 to 7) .
>TF	Get tank flow status	This code is sent from the controller to the tank subsystem to ask for the flow rate at the tank.
SP7	Sprinkler pump on	This code is sent to the SCS and received again by the controller to turn on the sprinkler pump
SP0	Sprinkler pump off	This code is sent to the SCS and received again by the controller to turn off the sprinkler pump
S1V7	Sprinkler 1 Valve on	This code is sent to the SCS and received again by the controller to open sprinkler valve 1.
S1V0	Sprinkler 1 Valve off	This code is sent to the SCS and received again by the controller to close sprinkler valve 1.
TMxx	Set time min	This code is sent to the user interface every minute to enable the user interface to display the time. The minutes are indicated by xx.
THxx	Set time hours	This code is sent to the user interface every minute to enable the user interface to display the time. The hour is indicated by xx.

The function that is used to check for incoming data:

```
void UART1get(void){
for(y=1;y<10000000;y++){
if (U1STAbits.URXDA!=0) {
y=10000000; x=1;}
else x=0;};
```

If this function is called a for loop is started. Every time the for loop runs through, the URXDA flag is checked. This flag tells if there is something in the receive register. If this flag is set, the for loop is ended by making y=1000000 and the code will return to the function that called this function. A global variable x is also set if something is received. The function that called the UART1get function will then check x. If no data was received at the end of the for loop, x will stay 0 and the function that called UART1get will restart.

The flow check is done in the following way:

```
if (WaterPump==1){ //Check tankflow
z++;
if (z>10000){
GetTankFlow();
z=0;
if (TankFlow==0){
x++;
if (x>=3){
while(WaterPump==1) WaterPumpOff();
x=0;}
}}}
```

The z variable will ensure that the controller only checks the tank flow every 10000 cycles of the main loop. It can take the pump a while to start up or for the water to reach the tank. The amount of cycles can be increased to compensate for a bigger system. If the tank flow is 0 after 3 tries, the water pump will turn off. The main controller will then wait for 10 minutes before turning on the pump again.

To turn on a sprinkler, the following code is needed:

```
tm.l=RtccGetTime();
if (tm.min==Sprinkler1Time && date.wday==Sprinkler1Day && TankLevel>3){
//tm.hour
if (Sprinkler1On==0){
S1ont=tm.min + Sprinkler1OnTime; //tm.hour
if (Sprinkler2On==0 && Sprinkler3On==0 && Sprinkler4On==0)
while(Sprinkler1On==0) Sprinkler1ON();}}
```

In the first line the current time is fetched and saved into a real time clock class. The sprinkler will only be switched on if the time and day is right and the tank is more than halfway full. For demonstrational purposes the minutes and not the hours is used. The current time and the time a sprinkler must be on is added and saved in a variable. This variable will then later be used to turn the sprinkler off. Only one sprinkler can be on at a time.

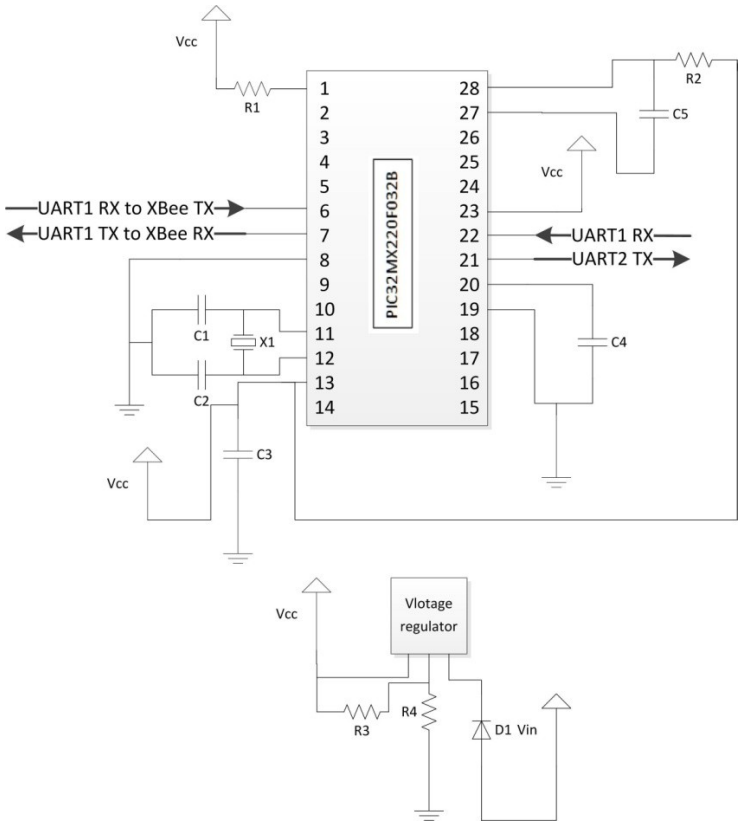


Figure 12: Controller circuit

With reference to Figure 12, all the power for the circuit will enter at Vin and then be regulated to 3.3V to power the PIC and the XBee module. Vin can range from 5 to 25 V. This feature enables the system to be flexible and can easily be powered by a solar system that charges a battery. More on the LM1085IS-ADJ voltage regulator used to regulate the input voltage is explained in section 2.2.3

Diode D1 is added to protect the hardware from faulty connections. In case the polarity of Vin is the wrong way around, the diode will block the current flow and no power will be given to the circuit.

The circuit has been built on a PCB. The design(Figure 13) and build process for this can be found in the technical report.

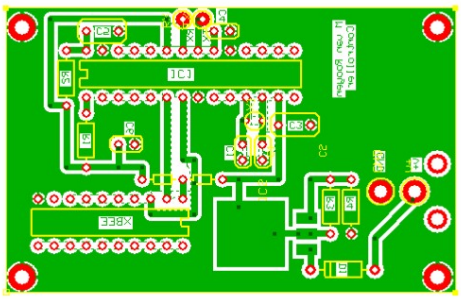


Figure 13: PCB layout of main controller

Finally the circuit will be powered from a 12VDC power supply. The whole circuit fits into an 85x56x30mm enclosure.

2.2.3 Level and flow sensors

The LM1085IS-ADJ regulator IC is used to get voltages of 3.2 V and 5.3 V from the 12V supply. The formula used to calculate the required resistor values to get the specified voltages is shown along with the component values. The equation is

$$V_{out} = 1.25 \times \left(1 + \frac{R_2}{R_1}\right) \quad (1)$$

Solving for R2 yields the table that follows which shows the component values.

Table 6: Component values for required voltage outputs

Output Voltage	R1	R2
3.2 V	120 Ω	186 Ω
5.3 V	120 Ω	390 Ω

The 3.2 V was used to supply power to the PIC18F45K20, the 5.3 V to supply the Hall Effect sensor and Pressure sensor, and the 12V is used to power the operational amplifiers used to make the instrumentation amplifier.

The offset of the pressure sensor was removed by connecting a variable resistor between its Vs, supply, and –Vout, shown in figure 3, pins and tuning it until the differential output between pins + Vout and –Vout was equal to zero. The resistance value at which the offset is made zero is 50 kΩ and because a variable resistor is large and takes up space it was replaced by two 100 kΩ resistors in parallel to achieve the same result. This was done when both ports of the pressure sensor were at the same atmospheric pressure; this was used as the tank empty pressure reference.

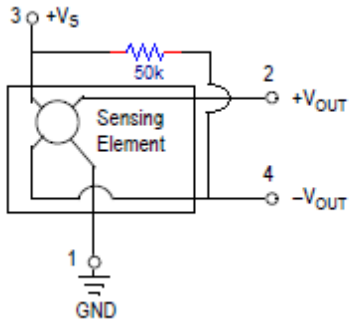


Figure 14: Offset compensation resistor for pressure sensor

For the purpose of this project the tank was simulated using a bucket with a height of 20 cm which means that the pressure for a full tank corresponds to 2 kPa, it should be noted that these values can be scaled up and down as long as the accompanying code is also adjusted accordingly. According to the datasheet at a pressure of 2 kPa the differential output should be roughly about 7 mV. This was then measured and turned out to be relatively close coming in at 7.35 mV. This value then had to be amplified by an instrumentation amplifier to a voltage that can be effectively used by the ADC on the PIC18F45K20.

The instrumentation amplifier was implemented using two operational amplifiers in which both of the operational amplifiers provide signal gain and the second operational amplifier performs the difference function. The transfer function of the instrumentation amplifier shown below is

$$V_{out} = (V_1 - V_2)\left(1 + \frac{R_1}{R_2} + \frac{2R_1}{R_G}\right) \quad (2)$$

The equation can be derived by performing KCL at nodes A and B which would yield the following

$$\frac{V_2}{R_1} + \frac{V_2 - V_{o2}}{R_2} + \frac{V_2 - V_1}{R_G} = 0 \quad (3)$$

$$\frac{V_1 - V_{o2}}{R_2} + \frac{V_1 - V_2}{R_G} + \frac{V_1 - V_{o1}}{R_1} = 0 \quad (4)$$

The above two equations can then be solved for V01 to get the result shown in equation 2.

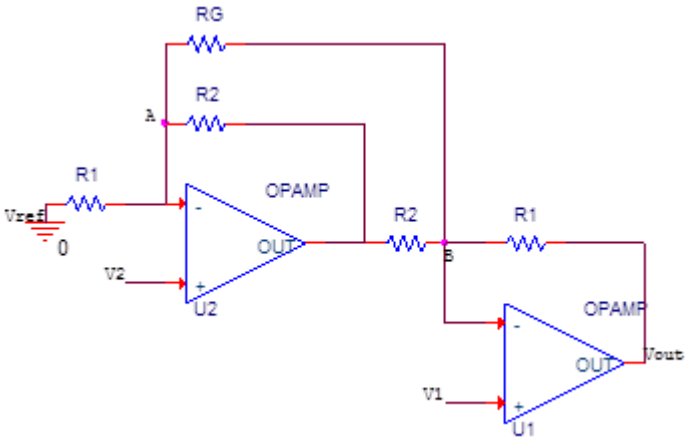


Figure 15: Two amplifier instrumentation amplifier

Choosing component values for R1 and R2 as 10 kΩ and 1 kΩ made the gain, the gain can be obtained by dividing equation 2 by the term (V1 – V2), dependant on RG only. Because 7 mV represents the tank full limit the gain required to amplify the signal to 3.2 V, which is the reference voltage of the ADC, is calculated as 457 V/V. Solving for RG a value was found as 44 Ω. Simulation were performed in PSpice, omitted here for this case, and this value provided to much gain, the output was also measured with a multimeter and the PSpice results were confirmed. A calculated guess was made and a gain of 100 V/V less was used, 357 V/V, which yielded a value of 58 Ω, the nearest realisable resistor value was chosen which is 60 Ω which corresponds to a gain of 344 V/V. Those results were then used in the PSpice simulations whose results are shown below.

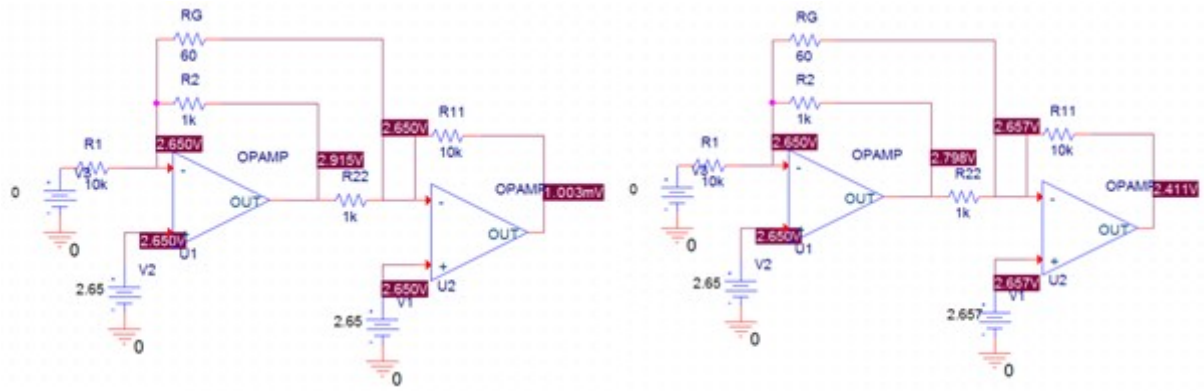


Figure 16: PSpice simulations for instrumentation amplifier

In figure 4 above V2 and V1 represent –Vout and +Vout of the pressure sensor with respect to ground, it should be noted that the voltages are at half of Vs. The circuit on the left represents the output for an empty tank and the circuit on the right represents the output for a full tank when the difference between V2 and V1 is 7 mV. To verify the simulations results the output was measured with a multimeter and for the tank empty state a voltage of 1.35 V was measured and for the tank full state a voltage of 3.18 V was measured. This amplified signal was then fed into the ADC of the PIC18F45K20.

Next the Hall Effect sensor was set up. The output was equal to the supply voltage of 5.3 V when no magnetic field is present and drops to about 0.3 mV in the presence of a magnetic field. There was a pull up resistor from the output to supply and this had to be taken into account when calculating the component values for the voltage divider network to output 3.2 V for input into the PICs CCP pin. As this was such a trivial calculation it was omitted and the result yields a value of 1560 Ω, this resistor is connected from the output to ground making the output voltage the wanted 3.2 V. A square wave will be generated by the sensor as the paddle wheel starts to rotate as a result of water flowing in the pipe. This square wave was fed into the capture/compare/pulse width modulation pin on the PIC and the capture capability was used to determine the period of the square wave and hence determine if water was flowing.

The appropriate pins on the PIC18F45K20 had to be setup to process the signals received from the sensors and communicate with the controller. The CCP1 pin was enabled for input from the Hall Effect sensor, the ADC was enabled and setup to process the signal from the pressure sensor, and finally the UART pins had to be setup and enabled for communication via the XBee. The timer module, specifically Timer 0, also had to be setup as it is used in conjunction with the CCP module on the PIC. The flow diagram of code that will be running on the PIC18F45K20 to process the signals from the sensors as well as communicate with the controller are shown and explained next.

The program basically loops continuously polling the ADC to continuously measure the water level and transmit the communication code corresponding to the water level and then transmit that code only once per level and again only once per level when a level change occurs. The level changes occur at 12.5% increments/decrements to correspond to the communication codes for example TL0 - 0% (empty tank), TL1 - 12.5% and so forth up until TL7 represent a full tank. While continuously polling the ADC the PIC can attend to a request made from the controller and calculates the flow rate, this was addressed in the ISR of the program. The communication codes for the flow detection work similarly to the communication codes for the tank level with TF0 representing no flow and the only difference being that the upper limit regarded as high rate of flow was represented by the code TF3.

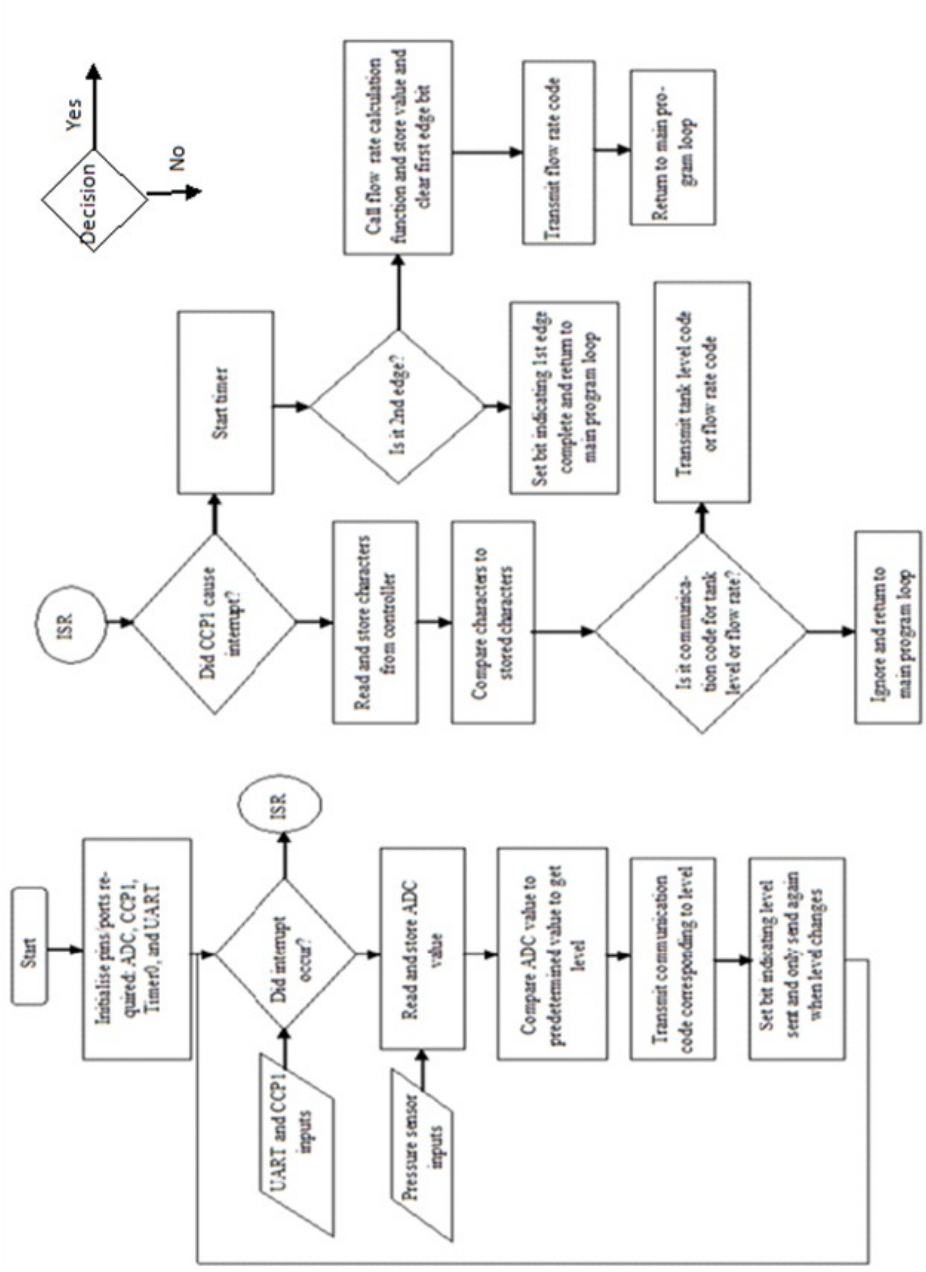


Figure 17: Flow chart of code to process sensor data on PIC18F45K20

2.2.4 Sprinkler system

2.2.4.1 Hardware Design

To use the 3.3V output from the PIC32's pins to drive the valves and relay using 12V, the following circuit was designed. For the transistor to switch on completely a base current (Ib) of 8mA was chosen. This also falls into the range that the PIC32 can source.

Vf_{doide} = 0.7V

$V_{on\text{-}tarnsistor}$ = 1V

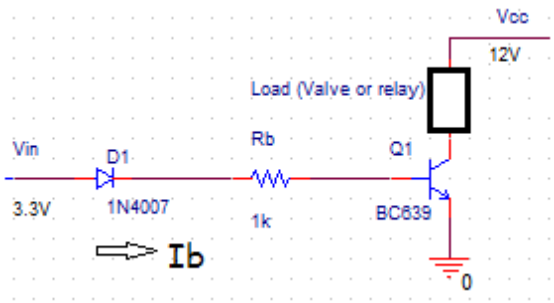


Figure 18: Circuit to switch 12V with 3.3V

$-V_{in} + Vf_{doide} + IbRb + V_{on\text{-}transistor} = 0$

$-3.3 + 0.7 + IbRb + 1 = 0$

$Rb = 3.3 - 0.7 - 10.008$

$= 200$

2.2.4.2 Hardware Implementation

The implementation of the SCS on the breadboard or PCB has the following setup. The following figure shows the layout:

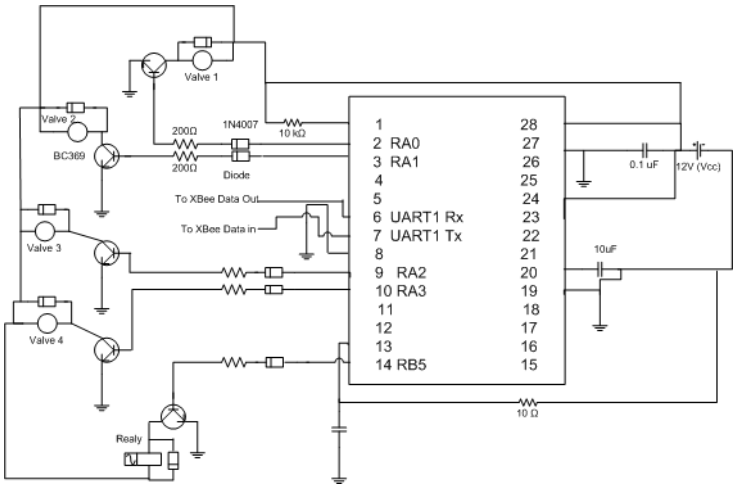


Figure 19: SCS Circuit diagram

The whole system as show in the figure above is powered by a 12Vdc power supply that is then connected at Vin from where it is then regulated down to 3.3V using a surface mount AMS117 3.3V regulator to power the PIC32 and the XBee PRO module. Vin must be in the region of 12V to be able to switch the pump and valves, but this means that the SCS can be power by an 12V car battery or a solar panel system if the push comes to shove. A diode (D1) has been incorporated in the circuit to prevent the circuit to get damaged when the input voltage polarity is swapped around. The trade off losing 0.7V on input voltage had to be accepted to protect the circuit.

Each output pin of the PIC32 that is being used is supplied with a diode to protect the PIC32 when the transistor is switched on. The valves and relay for the pump is connected between Vcc and the collector of the transistor as shown in Figure 12. Figure 12 is incorporated in every output pin used.

The whole circuit was designed using ExpressPCB and then etched on PCB. Figure 1 shows the PCB design for the SCS. The process and design can be seen in the technical document.

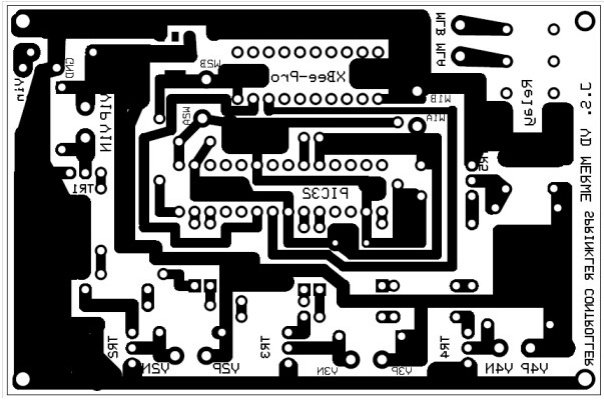


Figure 20: PCB design for SCS

The UART1 RX and TX pin in Figure 13 above will be connected to the XBee PRO transceiver module's RX and TX pins as shown in Figure 18 in section 2.2.5.

2.2.4.3 Software Implementation

The code that is implemented on the PIC32 was typed in such a way that it will always run in a never ending loop, always pending and waiting for the receive register to receive specific UART codes.

The following code snippet shows the never ending loop the code run in.

```
while (loop==1)
{
while (U1STAbits.URXDA==0);
getsUART1(4,inset1,1600);
//and to keep the never ending loop:
loop = 1;
```

A safety feature was also incorporated in the SCS's code that prevents the sprinkler pump to turn on before any of the valves are switched on. This will prevent the pump from burning out or the valves' seals from wearing out fast. The following code snippet shows what was just discussed:

```
if (inset1[0]=='S' && inset1[1]=='P' && inset1[2]=='7'){ //if this is the
received UART code //then do the following
if(SV1==0 && SV2==0 && SV3==0 && SV4==0){ // make sure that if all the
valves are off to //keep the pump off also
mPORTBClearBits(BIT_5);
for(y=1;y<1000;y++);
putsUART1("SP0");}
else{ // if one valve is open switch the pump on
mPORTBSetBits(BIT_5);
for(y=1;y<1600;y++);
putsUART1("SP7");
for(y=1;y<1000;y++);
SP1=1;
inset1[0]='0'; inset1[1]='0'; inset1[2]='0'; inset1[3]='0';
}};
```

In the code for the SCS provision was made for four valves that can be chosen for irrigation, but for demonstration purposes only two will be used. The SCS receives a specific UART code on the receive port of UART1 from the main controller i.e. S1V7 meaning that the controller wants valve one to switch on. After the valve is switched on the SCS will reply with the same code telling the main controller that the status of the controller is on. This can also be done for the other valves (S2V7) and as well as for the sprinkler pump (SP7). The main controller will also want the status of the valves and pump during irrigation by sending the following codes to the SCS: >S1V, >S2V and >SP. The output pins of the PIC32 will drive the valves and relay for the motor as described in the hardware design.
Figure 1 shows the inputs and outputs for the PIC32:

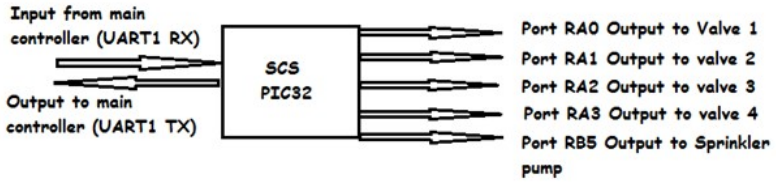


Figure 21: Block diagram for input/output

If you want to switch between different valves without switching the pump of provision was made in the code that the valve you want to switch on, switches on while the first one remains on for a second or two to relieve pressure from both valves and then the first valve will switch off. Finally if the main controller sends SP0 the SCS will switch the pump off first and then a few second later closes the valve that was open. The following figure shows the flow diagram of the software coded on the PIC 32:

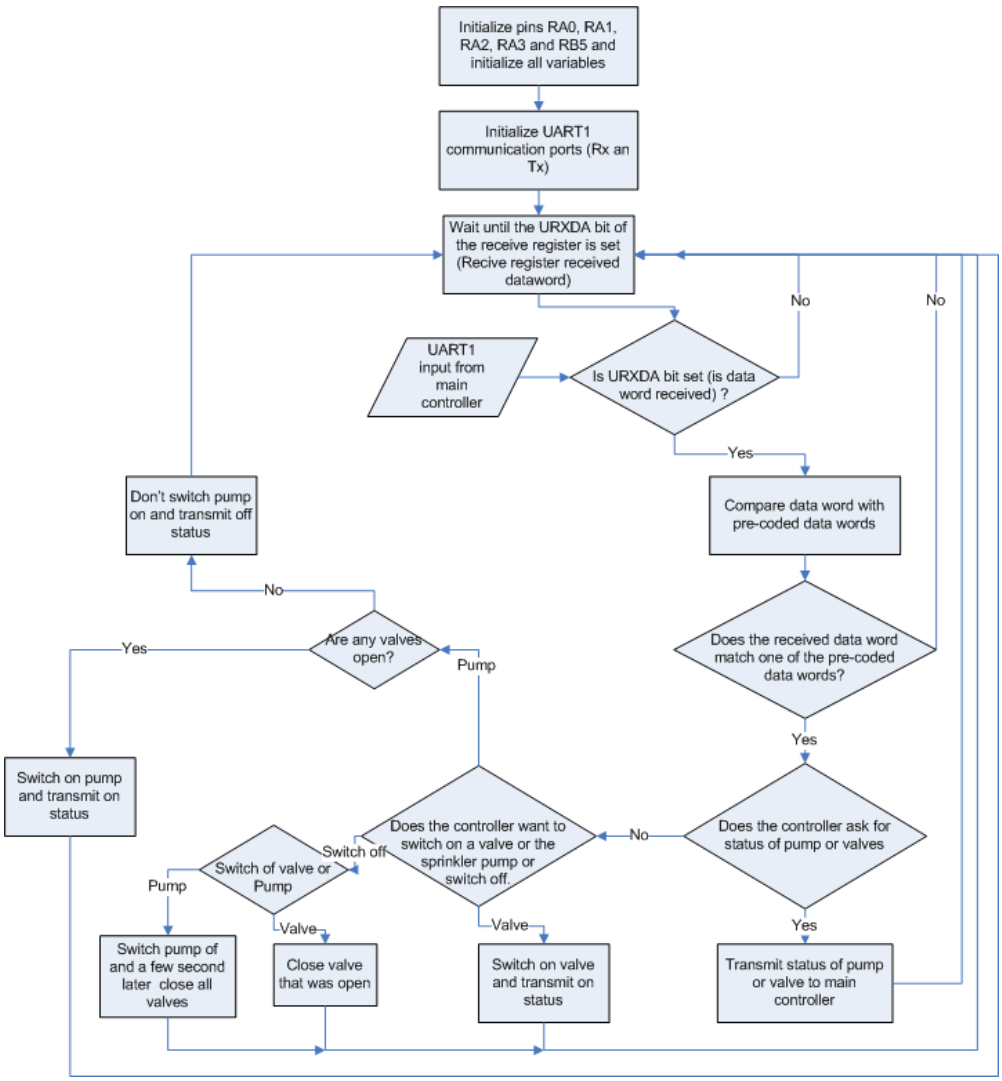


Figure 22: Sprinkler controller code flow diagram

2.2.5 XBee communication

The XBee PRO transceiver module was used as you buy it in store or online, but alterations had to be made for the pins because the XBee's pins are spaced 2 mm apart were the commonly found pin headers' pins are 2.54 mm apart. An adaptor that takes the 2 mm spacing and converts it to 2.54mm was designed on ExpressPCB and then PCB was made of the adaptor design. The following block diagram shows the connection between micro controllers and the XBees, as well as the wireless link between the XBees.

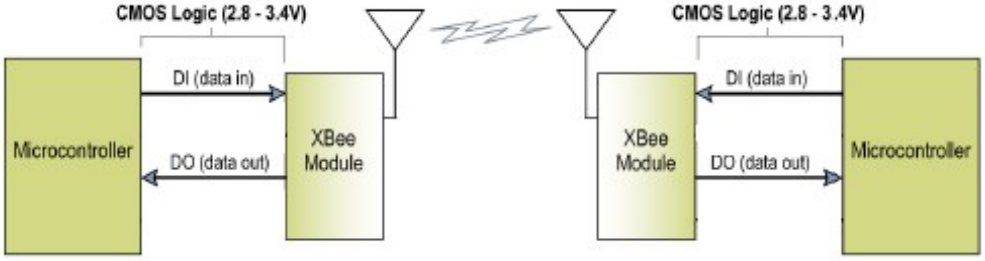


Figure 23: XBee UART communication

In our project we are using three Xbee transceivers. The first module at the main controller, the second at the level and flow sensor system and the third at the SCS. If the main controller transmits a data word, all the other subsystems will receive that code word. The separate subsystems each has a code word recognition part to see if the code word is for the specific system, otherwise it will just be ignored. The communication link between the subsystems is shown in the following flow diagram.

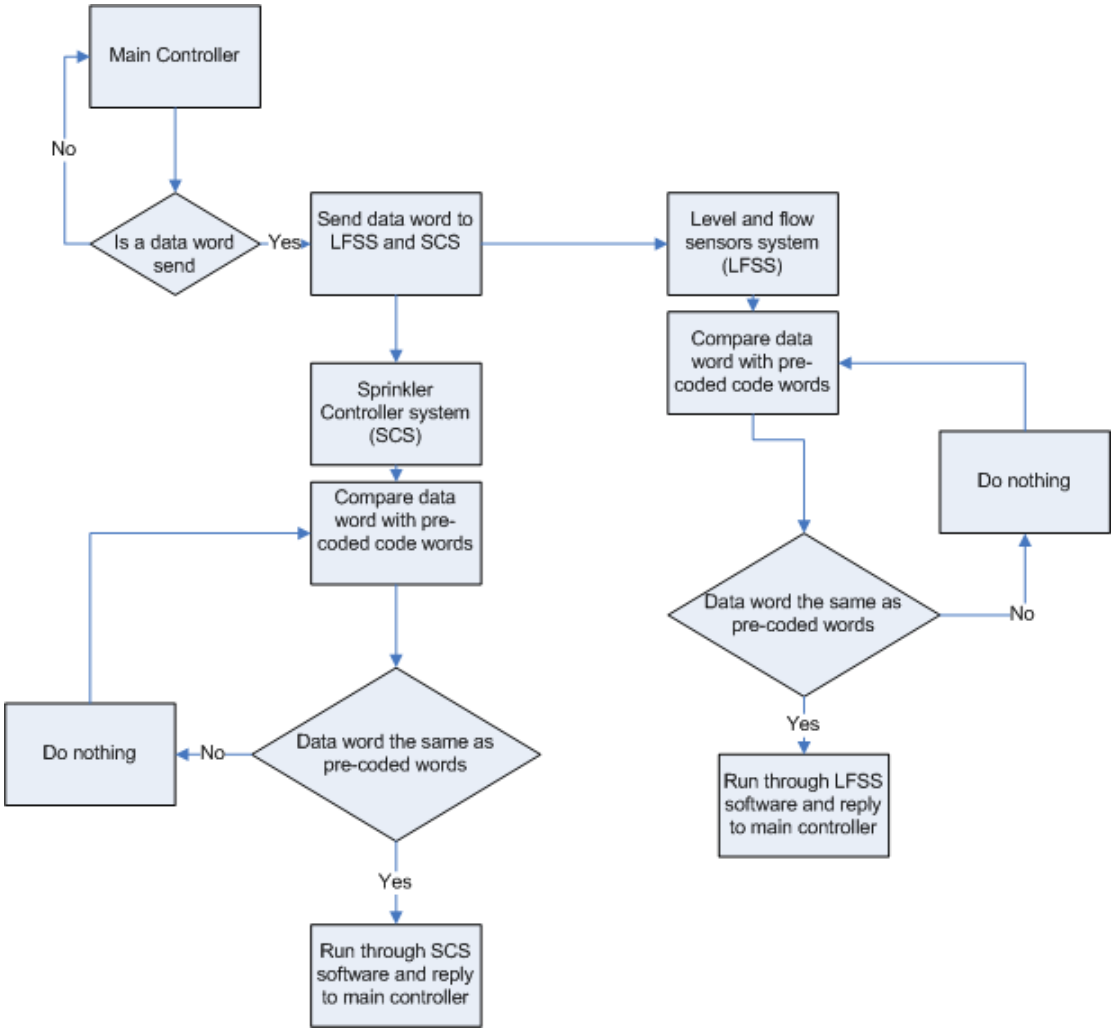


Figure 24: Flow diagram for wireless communication between subsystems

Figure 25 below shows how the XBees are connected to the PIC's UART ports.

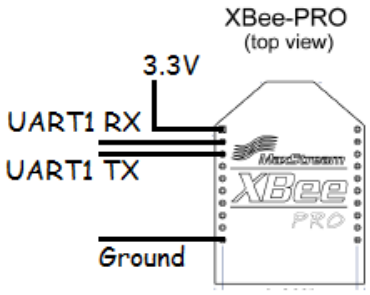


Figure 25: Diagram of how the XBees are connected to the PIC's UART ports

3 Results

3.1 Summary of Results achieved

Expected results	Real results
The UART communication works between the PIC and a computer	The UART communication works good between the PIC and a computer, except that the team had to add some delays to clearly see the response.
The RTCC works	After changing the crystal, the RTCC works.
The PIC circuits work on homemade PCBs	The assembled and soldered circuits work.
Test solenoid valves with transistor circuit and water feature pump	Transistor circuit works with valve but pump does not give enough pressure to open valve completely. Good enough for demo purposes.
Level sensor functions correctly, not affected by simulated weather changes, and transmits correct communication code	Simulated weather conditions do not affect the sensor, a false tank empty/full signal will not be sent to the controller
Flow meter transmits correct data for rate of flow detected	The flow meter functions correctly, however has to be mounted in the upright position
SCS receives code words correctly over wireless link from main controller, reacts and transmits code word back.	Xbees was connected to the main controller and SCS. Program was ran from the main controller and the SCS reacted automatically as expected.
XBee communication between subsystems work	After sorting out the problem of power to the XBees the communication to the subsystems work perfectly
Whole system works together automatically without glitches	Connected subsystems and tested it. A few code word errors was made in the software code that first had to be fixed and then everything worked fine.

3.2 Experiment 1: User interface

3.2.1 Experimental work

To test the communication codes the circuit was connected to a UART to usb bridge converter. A console program was connected And the codes typed manually to study the results. Some minor debugging of timing was also done for the LDC.

3.2.2 Results and observations

Except for some minor debugging and error correcting everything functioned as expected.

3.3 Experiment 2: Controller

3.3.1 Experiment 2.1

3.3.1.1 Experimental work

The first experiment conducted on the controller was to see if the UART communication works. This was done using two USB-to-serial bridges implemented on PIC18F13K50. The PICs, circuitry and the code programmed on the these PIC18s are from a previous subject, EMK310, the team had. Two terminals were used to see and send on the UART ports. The terminal program is called Termite.exe, easy to download and use. Termite is more user friendly than for example Putty and also does not need any installation, therefore it is used. The program on the PIC32 should be able to send the right codes and decode the replies correctly. This experiment was, however, repeated every time the code was changed or an error occurred in the test.

3.3.1.2 Results and observations

Seeing that the test was conducted several times, it is clear that there were a lot of errors and changes along the way.
It was easy to pick up a problem in the code using this experiment. If a reply was wrong or no reply occurred it was clear where a problem in the code was. The problem could then be solved fairly fast most of the time.
In the end the required results were obtained. When specific codes were sent to the PIC it replied with the correct codes.

3.3.2 Experiment 2.2

3.3.2.1 Experimental work

The second experiment conducted on the controller was to test the RTCC. The controller will send the time to the user interface every minute over UART. Through using a USB-to-serial bridge the time it takes for one minute to pass on the PIC can be measured. The time can be measured accurately by letting the PIC and a stopwatch run for an hour or longer and then check if the times are still synchronised.

3.3.2.2 Results and observations

At first the clock did not work at all and the PIC only picked up a crystal clock when someone touched the crystal. After research and various measurements in the lab, a conclusion of the multifunctional clock was made. There was an oscillation of around 70 KHz when someone touched the crystal. This caused the RTCC to work, but at double the speed of real time. After tests with other crystals and other PICs, a possible cause for the problem possibly was that the crystal used was not low power and the PIC needed a low power crystal. This conclusion was proved when a low power crystal was installed and the RTCC worked perfectly. Measurements were made and the RTCC was exactly in synchronisation with real time. It took one minute for one minute to pass and one hour for one hour.

3.3.3 Experiment 2.3

3.3.3.1 Experimental work

In this experiment the main controller was tested on the PCB after the PCB has been soldered and the power tested at various points. Two UART connections were connected to the PIC32 and the UART codes and the time were tested.

3.3.3.2 Results and observations

Everything worked perfectly. The UART communications and the RTCC functioned well.

3.4 Experiment 3: Tank flow and level sensors

3.4.1 Experiment 3.1

3.4.1.1 Experimental Work

The level sensor was tested in a bucket of water of height 20 cm which would equate to a pressure of 2 KPa for a full tank. To determine if changing weather conditions would affect the accuracy of the level measurement of send false tank empty/full signals to the controller the weather conditions were simulated. For hot weather conditions the water was heated up and the room temperature increased to simulate a hot day. For a cold day, cold water, after being in the fridge for an hour or two, and a colder room was used to simulate a cold day. The output of the level sensor was then measured for the two simulated weather conditions to determine how far from the normal values the readings deviate. If large deviations are discovered there will have to be either hardware or software compensation implemented.

3.4.1.2 Results and Observations

The signal was measured at the output of the instrumentation amplifier and is tabulated below against the measured values taken at room temperature, represent a normal day.

Table 7: Results of sensor performance in simulated extreme temperature conditions

Tank Level	Normal Conditions	Hot Conditions	Cold Conditions
0% (Tank empty)	1.35 V	1.35 V	1.35 V
50%	2.23 V	2.23 V	2.24 V
100% (Tank full)	3.18 V	3.17 V	3.18 V

The results indicate that there is no need for compensation, either hardware or software, as false tank empty/full alarms will not be triggered by changing weather conditions. It should however be noted that the sensor is more prone to be inaccurate with temperature changes at its upper limit of the pressure it can handle, but even then the inaccuracies only occur at extreme high temperatures (over 100 degrees Celsius even at 50 degrees Celsius the sensor remain accurate), temperatures unlikely to be experienced.

3.4.2 Experiment 3.2

3.4.2.1 Experimental Work

The purpose of this experiment was to ensure the correct working of the flow meter and to ensure that the correct communication code corresponding to the rate of flow is sent to the controller. For this experiment the controller was a desktop pc which will just receive the codes from the flow meter and display them to ensure they are correct. Lastly it was used to confirm if low rates of flow could be detected so as to prevent the tank from overflowing, as a result of low flow rates being detected inaccurately.

3.4.2.2 Results and Observations

The flow meter functioned as it should have but only when mounted upright. The one thing that was noticed is that the level of the water flowing in the pipe should not be below half as this will result in the water passing under the paddlewheel without causing rotation. As long as the water level in the pipe is above half, flow can be detected. The correct communication codes relating to flow were transmitted.

3.5 Experiment 4: Sprinkler system

3.5.1 Experiment 4.1

3.5.1.1 Experimental Work

The first experiment that was done was to see if the SCS UART communication worked as it was coded. The experiment was conducted using a USB to Serial bridge implemented on a PIC18F13K50. From the serial bridge a USB cable is connected to a PC or Laptop. A program called Termite was used as a terminal port from where data word could be send and receive. The experiment was conducted but only the data word could be send to the PIC32. There was no reply.

3.5.1.2 Results and Observations

The problem with the code was that a line of code was missing that needed to activate the transmitting port. The end result was that after that line of code was added the UART1 communication was working properly.

3.5.2 Experiment 4.2

3.5.1.3 Experimental Work

The second experiment conducted was to see if the code that was written for different UART input codes was working. Every time an error popped up, the code needed to be changed and the experiment had to be repeated.

3.5.1.4 Results and Observations

The experiment was repeated a lot. This meant that the system had a lot of error that needed to be eliminated before any coding could be done any further. It was easy to correct the error because there are different sections for each code word in the code. At the end all the errors was eliminated and every time a code was sent the correct reply was received.

3.5.2 Experiment 4.3

3.5.2.1 Experimental Work

The third experiment conducted was to test the valves with the transistor circuit, to see if it opens and to see what kind of pressure is needed to open it completely. The transistor circuit was powered and the valves were opened. A water feature pump was used but it didn't give enough pressure, but it still was able to push water through the valves after it was open. So a hose pipe and tap was used to oen the valves completely because a tap can give a great amount pressure. Luckily both the valves purchased worked as it was supposed to.

3.5.2.2 Results and observations

After conducting the experiment it was very clear that the transistor circuit does what it was designed for and that the valves needed a lot of pressure to open completely. It was decided that the water feature pump will be more than sufficient to be used in the final demonstration because it still give enough pressure for water to flow through the valves and it cost much less than a pump that can give the same pressure as a tap.

3.5.4 Experiment 4.4

3.5.2.3 Experimental Work

This experiment was conducted after the PCB design was done and etched. Before the PIC and Xbee was plugged in the pin headers, all the points where voltages needed to be was tested, as well as the ground connection. Then the PIC and XBee was connected, as well as the valves and the sprinkler pump. The PCB was powered and X-CTU was used as a terminal to send code words to the SCS.

3.5.2.4 Results and observations

The result of the experiment was that the circuit responded to all the correct code words sent and it replied with the correct code words. One thing that was observed was that the AMS1117 3.3V regulator got very hot. It is not because of to much current but because it is used to step 12V down to 3.3V, meaning there will be a lot of heat dissipation. After a few repetitions the regulator still worked and will be fine to demonstrate with.

3.6 Experiment 5: XBee communication

3.6.1.1 Experimental Work

As soon as the XBees (three of them) and their developer board were here, loopback tests were done on them to see that they are working and what range they have. Next they were connected to the subsystems that would use them. The activity on the "wireless UART" could still be monitored.

3.6.1.2 Results and observations

The loopback tests worked on all 3 XBees and they could easily receive data from more than 30 metres indoors. Another line-of-sight test was done and the XBees could receive data from 500m.

The XBee communication between the different subsystems worked wonderfully. The only problems we had were simple things like no power to the XBee.

3.7 Experiment 6: Integration

3.7.1.1 Experimental Work

This experiment was conducted in order to see if the communication between the different subsystems and the main controller will work when they are connected to each other. Each subsystem was connected to a USB-to-serial bridge (of course, the main controller has two bridges). Each of the subsystems was programmed with the code to implement the UART communication. All the subsystems were powered and the first code was sent by the main controller. This was visible in the portal, or in our case Termite. Then everybody who should receive this code typed it into their subsystem's portal. These systems then replied on the codes sent and the replied code words showed in the terminals. The replies were then typed into the controller's portals. The controller should recognise the codes and move on to the next set of codes. Afterwards the XBees was plugged into each subsystem and the main controller started transmitting data and automatically receiving data as well.

3.7.1.2 Results and Observations

The result of this experiment was that by using the terminals everything considering the interfacing worked, but the real results was awaited for by transmitting over the XBees. To the teams surprize the transmitting of data over the XBees, as well as the Receiving worked. This meant that the automation of the project was falling into place. If a code word received is not applicable to the one subsystem it is ignored allowing the other subsystem to respond to the controller's request.

4 Discussion

4.1 Interpretation of results

Our results are acceptable. All the requirements that were stated in the project protocol were adhered to and met in the end. It, however, took a lot of experimentation and research to achieve these results. The time the team spent on the project and the amount of team work done is proportional to the results of the end product. The UART codes together with the wireless communication work well.

4.2 Aspects to be improved

- The RTCC can be calibrated in more detail.
- The menu and controller can be improved with regards to the sprinkler settings.
- More accurate settings for the time the sprinkler should go on and how long the sprinkler should be on can be implemented.
- The menu buttons can be simplified.
- The XBees were ordered from overseas, but if the team had them earlier, they could have implemented more of the subsystems directly on them, seeing that they also have ADCs and I/O ports.
- All the UART codes could have been chosen to be the same length.
- Addition of remote configuration of settings.
- XBees could be programmed in a way that enables the main controller to communicate with the sensor system and SCS individually and vice versa.

4.3 Strong points

The system has secure communication between the subsystems. In the UART communications there has to be a 3-bit error to make an error to turn a pump or a sprinkler on. This feature will protect the pumps from damage. In case a system does not reply at a first request from the main controller, it will resend the request. A resend will also occur if the reply is incorrect because of noise or interference.
[Mention XBee encryption]

4.4 Under which circumstances will the current system fail?

- Seeing that the system might be placed in the middle of nowhere, the chance to get hit by lighting is larger than if it was situated indoors.
- No electronics can work with water. If an enclosure is not properly sealed and rain enters it, the system might fail.
- If one of the subsystems fail to answer a request from the main controller, the controller will keep on sending the request and may end up in an infinite loop if the other subsystem is broken for some reason.

- If the pipe of the pressure sensor gets clogged the level of water in the tank will go undetected and the tank will most likely run dry.

4.5 Design ergonomics

This project is there to make life easier. Someone does not have to go and turn pumps on and off and turn taps, because everything is done automatically.
The controller and the user interface are both at the water pump in the team's project. This can, however, easily be changed. The user interface can even be moved to a fourth position by using one more XBee, seeing that all the communication is via UART.
The user can edit the time, day and sprinkler times at any instance according to the user specifications.

4.6 Health and safety aspects of the design

- No unsafe materials have been used in the design.
- No open connections visible.
- The user should take care when connecting electrical connections, for example the motors. The main power should always be switched off to decrease the shock hazard.

4.6.1 I International protection (IP) ratings

International protection ratings quantify the protection provided by the enclosures or housings of electric equipment. It is applicable form transformer boxes to consumer electronics like cell phones. The IP-ratings consist of two parameter namely: the penetration of solids and dust and the protection against water penetration. A table of IP-ratings from SANS 10142-1(Standards South Africa:2003) is reproduced below:

Table J.1 — International protection ratings — IP ratings

1	2	3		4	5	6
First characteristic numeral				Second characteristic numeral		
Protection against solid foreign objects		Meaning for the protection of persons against access to hazardous parts with:		Protection against harmful ingress of water		Meaning for the protection against water:
IP	Tests			IP	Tests	
0	No protection	Non-protected		0	No protection	Non-protected
1	Full penetration of 50 mm diameter of sphere not allowed. Contact with hazardous parts not permitted.	Back of hand		1	Protected against vertically falling drops of water	Vertical dripping
2	Full penetration of 12,5 mm diameter of sphere not allowed. The jointed test finger shall have adequate clearance from hazardous parts	Finger		2	Protected against vertically falling drops of water with enclosure tilted 15 ° from the vertical	Dripping up to 15 ° from the vertical
3	The access probe of 2,5 mm diameter shall not penetrate	Tool		3	Protected against sprays to 60 ° from the vertical	Limited spraying
4	The access probe of 1,0 mm diameter shall not penetrate	Wire		4	Protected against water splashed from all directions – limited ingress permitted	Splashing from all directions
5	Limited ingress of dust permitted (no harmful deposit)	Dust-protected	Wire	5	Protected against low-pressure jets of water from all directions – limited ingress permitted	Hosing jets from all directions
6	No ingress of dust	Dust-tight		6	Protected against strong jets of water, e.g. for use on ship decks – limited ingress permitted.	Strong hosing jets from all directions
				7	Protected against the effects of immersion between 150 mm and 1 m	Temporary immersion
				8	Protected against long periods of immersion under pressure > 1m	Continuous immersion

SANS 10142-1:2003
Edition 1.1

323

Figure 26: Table of IP-ratings from SANS 10142-1

Considering that the level sensor will be mounted outside near water supply lines one would expect it to have an IP rating of IP55 i.e. protection from dust, and protections from strong jets of water typical to burst water pipes. The rest of the equipment is expected to be mounted together with switchgear and other electric equipment in locations already protected from water and uncontrolled access. Thus an IP rating similar to that required by distribution boards would be adequate, IP20 protection against contact with a finger.

4.7 Social and legal impact and benefits of the design

By increasing the productivity of farmers and farming entities more ensure the quality of the crop. Better crops and improved efficiency could lead to reduced food prices. This leads to consumer household savings, which in turn strengthens the economy by reducing household debt and improving the consumer's ability to spend on other products.

4.8 Environmental impact and benefits of the design

In most cases people will wait for a tank to overflow, because they can hear the tank overflowing, and then turn off the pump. Our project will save this overflow water seeing that the pump will turn off before the tank overflows.

5 Conclusion

5.1 Summary of the work

The work done includes the design and implementation of sensors and the programming of the microcontrollers for their respective applications within the system. The implementation of valves and pumps (for water and sprinklers) were also included. As well as setting up the XBee devices for communications between the subsystems and implementation of a user interface. Designing circuits on PCBs or Vero board (strip board) was also one of the phases of the project. Lastly the integration of the subsystems to function as a complete system was implemented.

5.2 Summary of the observations and findings

- All individual subsystems delivered the expected results after minor problems were sorted out.
- Improvements can be made as mentioned in section 4.2.
- UART communication between of the different subsystems as well as the communication between subsystems worked as was planned.
- False tank empty/full signals were not sent to the controller in simulated weather conditions.
- The flow of the system from the user interface module to the SCS worked well as expected.

5.3 Suggestions for future work

Future work to be done can include incorporating a solar power system to be used as a supply of power for the system. Such a system will, however, need a battery to power 300mA transmission current for the XBees.
The user interface can also be improved to fit into a watertight enclosure and to add more user-changeable variables.
More sprinklers could be added to the system.
Plug in modules could be developed for the system to enable direct switching and starting of Three-phase pumps. The system could be expanded to start up diesel engine driven pumps. Additional sensors such as pump temperature sensors, rain sensors and ground moisture sensors can also be included.

6 References

- Blue-White. 2012.*Paddlwwheel Flowmeters And why they are popular*. [Online]. Available at: http://www.bluwhite.com/artpaddlewheel.htm [Accessed 26 October 2012]
- Cartier M., 2005. *Paddle Wheel Flow Technology: When Do You Use It*. [Online]. Available at: http://www.gfsignet.com/go/?action=GF_DocumentDownload&doc_uuid=4A718CC019993E1D34E4F1B46CD121CD [Accessed 29 October 2012]
- Digi International, 2009. *XBee/XBee PRO RF Modules datasheet*, Minnetonka: s.n.
- Digi, 2012 . X-CTU software . [Online]. Available: (http://www.digi.com/support/productdetail?pid=3352&osvid=57&type=utilitie) [Accessed 6 October 2012].
- Hambrice K., Hopper H., 2004, *A Dozen Ways to Measure Fluid Level and How They Work*. [Online]. Available at: http://www.sensorsmag.com/sensors/leak-level/a-dozen-ways-measure-fluid-level-and-how-they-work-1067 [Accessed 31 October 2012]
- Huang, H. 2007 Pic(c) Microcontroller: An Introduction to Software & Hardware interfacing. New York: Delmar Cengage Learning
- Kang, S. & Kim, W. 2007. *Minimalist and Intuitive User Interface Design Guidelines for Consumer Electronics Devices*. Journal o Object Technology Vol. 6, No. 3, [online] available from http://www.jot.fm. [3 November 2012]
- Lazaridis, G., 2008. *PCB Heaven: Choosing the right PIC*. [Online] Available at: http://pcbheaven.com/picpages/Choosing_the_right_PIC/ [Accessed 30 October 2012].
- Micro Robotics, 2012. FOCA USB to Serial + Xbee board (FTDI).[Online]. Available at: http://robotics.org.za/index.php?route=product/product&path=46&product_id=462 [Accessed 23 October 2012]
- Microchip, 2012. *MPLAB Integrated Development Environment*. [Online] Available at: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002 [Accessed 31 October 2012].
- Microchip, 2012. *PIC18 MCU Products*. [Online] Available at: http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1004&mid=10&lang=en&pageId=74 [Accessed 31 10 2012].
- Microchip. 2008. *8-Bit PIC Microcontrollers*. [Online]. Available at: http://ww1.microchip.com/downloads/en/devicedoc/39630f.pdf [Accessed 30 October 2012]
- Microchip. 2010. DS41303G PIC18F23K20 /24K20 /25K20 /26K20 /43K20 /44K20 /45K20 /46K20 Data Sheet.

- Milone Technologies. 2012. *eTape Datasheet*. [Online]. Available at: http://www.milonetech.com/uploads/eTape_Datasheet_12110215TC-8.pdf
- National Instruments. 2011. *Pressure Measurement Basics*. [Online]. Available at: http://www.ni.com/white-paper/13034/en [Accessed 31 October 2012]
- Netram Technologies, 20 August 2012. 12V Solenoid Valve. [Online]. Available:http://www.netram.co.za/Electronics/Mechanical/12V-Solenoid-Valve-3/4.htm [Accessed 20 August 2012]
- Standards South Africa. 2003 South African National standard, The wiring of premises, Part 1: Low-voltage installations. SANS 10142-1:2003. Pretoria: Standards of South Africa (a division of the SABS)
- The Engineering Toolbox. 2011. *Types of Fluid Flow Meters*. [Online]. Available at: http://www.engineeringtoolbox.com/flow-meters-d_493.html [Accessed 28 October 2012]
- Topway,Shenzhen Topway Technologies Co.,Ltd.. 2005. LMB163ABC-manual-Rev0.2.DOC [Online]. Available at: http://www.topwaydisplay.com [Accessed 1 November 2012]
- Wikkies Pompe, 2012. submersible pumps. [Online]. Available: http://www.wikkiespumps.com/#!__industrial/submersible-pumps [Accessed 29 October 2012].
- Yahoo voices, What is GSM Cellular Signal, and How Does it Work? [Online]. Available: http://voices.yahoo.com/what-gsm-cellular-signal-does-work-2269266.html?cat=15. [Accessed 29 October 2012].

7 Appendixes

7.1 Project protocol

Project Proposal – ELO 320 **Your Group Number: 2**_____

Student number: 29042951_____ Full name: Francois Botha_____

Student number: [redacted]_____ Full name: Lemuel Curtis Short_____

Student number: [redacted]_____ Full name: Jacobus Stephan v/d Merwe

Student number: [redacted]_____ Full name: Marno van Rooyen _____

 We confirm that we have read and abide to the plagiarism statement as provided on the website:

http://www.ee.up.ac.za/main/en/undergrad/guides#plagiarism

 Member contribution remains equal, unless otherwise stated (indicate next to student number).

Project category: **Remote Sensing and Control**_____

Project title: **Automated Remote Sensor and Control System for Water**

Word count: _285__

(less than 300 words?)

Introduction and impact

In many industrial and agricultural installations, conventionally wired control circuits are impractical either due to hazardous environments, or extreme distances between sensors and the devices they control. In an attempt to meet this need a wireless sensor and universal controller will be implemented. As an example of such a system (see Figure1) will be designed and build, namely an automated remotely activated water pumping system. It will consist of a tank with a level detection sensor, and a controller that can be used to activate a range of devices namely single phase motor, three-phase motor or engine. To achieve the wireless connection XBee transceiver modules, which uses IEEE 802.15.4, will be used. Other standards included in the project are IEEE p1451.2-RS232 for communication between the microchips and XBee transceiver modules.

Hardware to be delivered

At a tank there will be different sensors, one will measure the water level in the tank by means of a float switch and the other the flow of water from the dam towards the tank with a flow sensor. The data from the sensors will be processed by a pic microchip. The remote sensor system, at the tank, will communicate with the controller via an XBee transceiver module.[Lemuel] At the sprinkler system there will be a water pump and valves to different sprinkling areas, in order to irrigate different areas at different times.[Jaco] Other sensors are optional (like pump temperature) to the consumer, but will be excluded in this project.

The wireless modules will consist of XBees transceivers and will act as the communication link between the remote sensor and sprinkler system and the controller.[Jaco]
The controller module will be split into 2 sub-sections consisting of a section dealing with the functional action of the controller, including sensor/communication test routines and the control of the motor/engine. The controller will be implemented on a pic microchip.[Marno]
The other sub-section deals with the user interface for programming the adjustable parameters of the system(LCD display and Keypad). This will include a structured programming menu implemented on a pic microchip.[Francois]
This system is aimed at the agricultural industry, any environment where large distances exist between controllers and sensors.

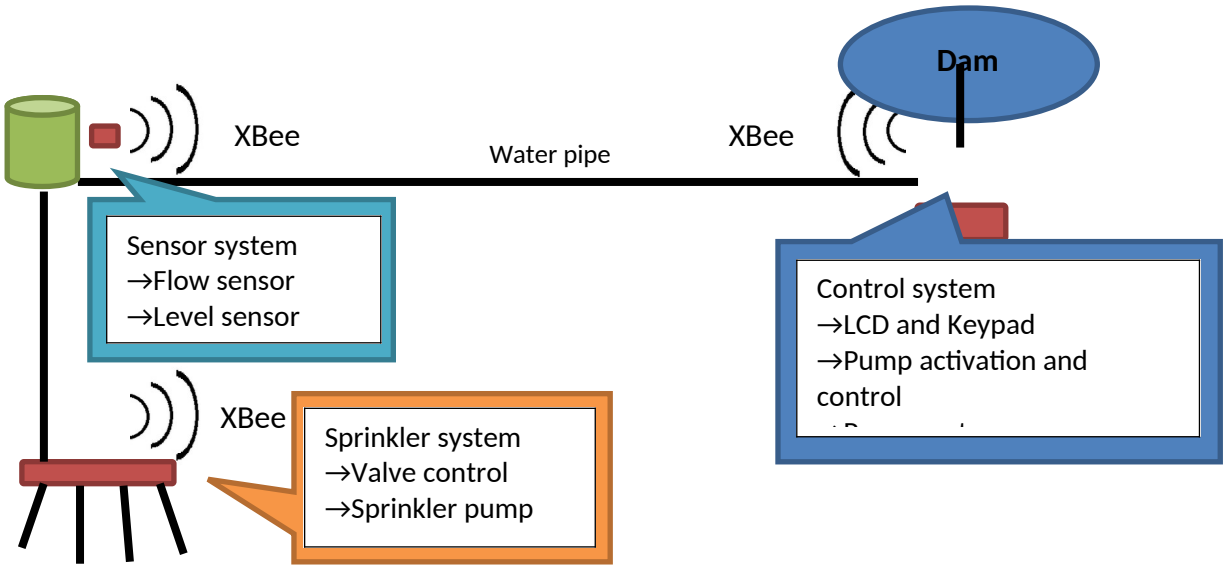


Figure 1. Project Proposal

7.2 Technical documentation and CD

The technical documentation section of the final report is included on the CD that accompanies this report. The CD is organized as follows: