OULU UNIVERSITY OF
APPLIED SCIENCES

# THE DESIGN AND IMPLEMENTATION OF A HYDROPONICS CONTROL SYSTEM

Mark Griffiths

# THE DESIGN AND IMPLEMENTATION OF A HYDROPONICS CONTROL SYSTEM

Mark Griffiths
Master's thesis
Autumn 2014
Information Technology
Oulu University of Applied Sciences

## ABSTRACT

Oulu University of Applied Sciences
MEng Information Technology

---

Author: Mark Griffiths
Title of thesis: The Design and Implementation of a Hydroponics Control System.
Supervisor: Dr. Kari Laitinen
Term and year when the thesis was submitted: Autumn 2014 Number of pages: 74

---

This thesis was born out of the idea to make hydroponic food growing easier and cheaper, hydroponics is essentially the growing of plants without the use of soil. The idea of it being open source is also in keeping with the general community feeling surrounded by the hydroponic movement.

The objective of the thesis was to create a working hydroponic controller, which is cheap and simple enough to build. It will monitor and control the key environmental ingredients needed for successful hydroponic growing, chiefly the pH, EC, air and water levels. Visual alarms will be raised if these go outside of predefined ranges. The controller will also be able to control external HW, such as lights, water heater and a water pump.

A circuit board was developed, which will act as a shield for an Arduino 2560 board. The circuit board will help in the ease of building the system. It acts too, as a way of keeping the HW more stable, as wires will not come loose and interference will be minimised. The SW and HW design will be available for everyone to download.

Knowledge of what is important in hydroponic growing was considered and these factors were used as inputs into driving the SW and HW requirements. UI usability studies were carried out to ensure that the controller would be enjoyable to use. Software design and software patterns were taken into use to make the SW more module and expandable. Investigation into any third party SW libraries was done and consideration into licensing was carried out.

This thesis details the HW and SW and why certain design decisions were made. The design of the SW classes are also described as well as the overall functioning of the SW. An explanation of how to build the system is also given.

---

**Keywords:** Hydroponics, Open Source, Software Design, Hardware Design, Arduino

# PREFACE

I would like to thank Niina Kuokkanen for participating in the UI usability study and Kari Laitinen, my project supervisor, for giving feedback on the project.

Mark Griffiths, Oulu, October 2014

## CONTENTS


ABSTRACT 3

PREFACE 4

CONTENTS 5

TERMS AND ABBREVIATIONS 7

1 INTRODUCTION 8

2 WHAT IS HYDROPONICS? 11

- 2.1 Nutrient Solutions 12
- 2.2 pH Level 13
- 2.3 Electrical Conductivity: EC Level 13
- 2.4 Water Temperature 14
- 2.5 Air Temperature 15
- 2.6 Light Level 15

3 CONTROLLER HW REQUIREMENTS & DESIGN 16

- 3.1 DHT11 Temperature sensor 16
- 3.2 DS18B20 Water Temperature Sensor 17
- 3.3 Photo-resistor 18
- 3.4 DS3234 Real Time Clock 18
- 3.5 TFT Touchscreen 20
- 3.6 Transmitter & Plugs 21
- 3.7 EC Circuit 22
- 3.8 pH Circuit 23
- 3.9 Enclosure 25
- 3.10 HW Layout 25
- 3.11 Overall View of the System 27

4 SW REQUIREMENTS AND DESIGN 28

- 4.1 UML Overall Package Diagram 29
- 4.2 UI Package 29
- 4.3 UTFT Library 37
- 4.4 Engine 40
- 4.5 Plug Receiver and Transmitter Library 41
- 4.6 DS3234 RTC Library 42
- 4.7 Dallas Water Temperature Library 43
- 4.8 DHT11 Air Temperature Library 43

4.9 EC Probe 44
4.10 pH Probe 50
4.11 The Main Loop Controller 54
5 DISTRIBUTION OF THE SW 56
6 UI USABILITY STUDY 57
6.1 Usage Scenarios 58
6.2 Description of Test Procedure 64
6.3 Observations of the Execution 64
6.4 Recommendations 65
6.5 Usability Conclusion 66
7 BUILD MANUAL 67
8 PROBLEMS FACED 70
9 FUTURE DEVELOPMENT POSSIBILITIES 71
10 CONCLUSIONS 72
REFERENCES 73

## TERMS AND ABBREVIATIONS

| | |
|---|---|
| **HW** | Hardware |
| **EEPROM** | Electrically Erasable Programmable Read-Only Memory |
| **RTC** | Real Time Clock |
| **Data** | Used to get the temperature. |
| **pH** | power of Hydrogen - The measure of the acidity or basicity of an aqueous solution. |
| **EC** | Electrical Conductivity |
| **SW** | Software |
| **UI** | User Interface |
| **GND** | Ground |
| **PCB** | Printed Circuit Board |
| **VCC** | Voltage |
| **BT** | Bluetooth |
| **TDS** | Total Disolved Salts |
| **API** | Application Programming Interface |

# 1 INTRODUCTION

This master's thesis focuses on making an open source hydroponics controller. The significance of hydroponics is in providing a way for the average person to grow their own food without the need of soil, for example, for people living in flats and inner city areas. Correct pH, EC and temperature levels of the water are critically important in hydroponics. Therefore, the help of a controller that monitors these factors is invaluable and will ensure higher success and efficiency rates of the grower.

At the moment, however, such controllers are expensive and none of them are open source. Occasionally people will say they have made a controller themselves, but they never give the code or plans away. The end goal of the project is to design and build a hydroponics controller in which the hardware and software are open source. There are many places to learn how to build the actual hydroponics system, the main one being our.Windowfarms [10]. The controller would complement this.

The controller itself controls lights, a water pump and a fan to aid in the growing of food.  The use of appropriate lighting would make it possible for people living in conditions of little light, e.g. Finland, to grow fresh food all year round too.  This would be especially useful now that food prices are increasing.

A general overview of hydroponics and its advantages over soil based planting is given by Hector Munoz [1].

By providing an open source controller it will allow more people to access hydroponic growing. This will be achieved as the controller will be cheaper and will allow busy people to grow the food as the controller will take care of most of the aspects involved.  For example, making sure the pH/EC level or water/air temperature is correct.

It is also in keeping with the Window Farms project in which the infrastructure for the growing is open source.

As food gets more expensive, it will allow people to grow their own food more cheaply especially in areas where soil, light or temperature can be problematic for growing.

According to Wikipedia[2] , the advantages of hydroponics are the following.

- No soil is needed for hydroponics
- The water stays in the system and can be reused - thus, a lower water requirement
- It is possible to control the nutrition levels in their entirety - thus, lower nutrition requirements
- No nutrition pollution is released into the environment because of the controlled system
- Stable and high yields
- Pests and diseases are easier to get rid of than in soil because of the container's mobility
- Ease of harvesting
- No pesticide damage
- Plants grow healthier
- It is better for consumption

Today, hydroponics is an established branch of agronomy. Progress has been rapid, and results obtained in various countries have proved it to be thoroughly practical and to have very definite advantages over conventional methods of horticulture.

There are two chief merits of the soil-less cultivation of plants. First, hydroponics may potentially produce much higher crop yields. Also, hydroponics can be used in places where in-ground agriculture or gardening are not possible.

In essence, the question to be answered in this Master's Thesis is:

*What engineering solution can be made to make it easier for people to successfully engage in hydroponics at a reasonable price?*

The purpose in doing this Masters Thesis would be to build a prototype hydroponics controller. It would be functional and ready to use although the UI and look and feel may not be perfected. A circuit will be designed, which will be a shield for an Arduino Mega 2560. The circuit will be open source and available for download along with the code.

A manual for the system will be written, which would allow people to easily put the HW together and flash the SW. The project will be hosted on GitHub.

## 2 WHAT IS HYDROPONICS?

Before the SW and HW were designed, a study into the methods and usage of hydroponics was carried out. This way it is ensured that the controller will offer the maximum usage to the end user.

Essentially hydroponics can be summarised as “gardening without soil”. Instead plants get the extra nutrients they need from a water based solution, which is passed over their roots.

Passing the water over the roots is possible by hand. However it is generally impractical when you have more than a couple of plants. An automated system gives the advantage of working the pumps and timers for you, ensuring that the plants will always get the required nutrients and raising the success rate of the plants. The controller offers the advantage that the plants can be watered, even if the user is away from the house.

The controller is used in conjunction with a system. The system is the actual structure in which the plants will sit. There are four common types of system:

**Flood/Drain system**

In this system, the reservoir of water sits directly below the growing area. At certain intervals a pump will fill the growing area with water. When the pump is switched off, the water will drop back down into the reservoir.

**Drip system**

In the drip system the water pump will pump the water up over the plants in small drips, this may be continuously or at timed intervals. The water, via gravity, will drip over the plants and back into the reservoir.

**Nutrient Film Technique (NFT) System**

Essentially the same system as the drip system, except the plants are set-up in troughs and the water pumped into the troff and directly to the roots.

**Aeroponic System**

The same as the NFT system except you use a high pressure pump with holes cut out every six inches or so in the tube. This way the water sprays up at the roots.

The aeroponic system, is of current interest as it may be used to feed people on space missions. It has also been proven to be the best, in terms of nutrients, for growing food. According to Wikipedia [2]

> *"NASA research has shown that aeroponically grown plants have an 80% increase in dry weight biomass (essential minerals) compared to hydroponically grown plants. Aeroponics used 65% less water than hydroponics. NASA also concluded that aeroponically grown plants requires ¼ the nutrient input compared to hydroponics. Unlike hydroponically grown plants, aeroponically grown plants will not suffer transplant shock when transplanted to soil, and offers growers the ability to reduce the spread of disease and pathogens. Aeroponics is also widely used in laboratory studies of plant physiology and plant pathology. Aeroponic techniques have been given special attention from NASA since a mist is easier to handle than a liquid in a zero gravity environment."*

All systems have the same basic premise as their underlying methodology, namely to keep the roots wet and exposed to air. The hydroponics controller considered in this thesis will support all of the above systems, making it versatile over a large range of growing methods.

## 2.1 Nutrient Solutions

Nutrient solutions can be bought directly from shops. However there are some free open source tools created by professional chemists that allow you to make your own. These are:

HydroBuddy[11]
HydroCal [12]

The controller system will measure the following properties of the environment/nutrient solution, which are important when hydroponically growing plants. These are discussed in the next sections.

**2.2 pH Level**

The control of pH is extremely important, not only in hydroponics but in soil as well. Plants lose the ability to absorb different nutrients when the pH varies. Different plants have a particular pH that is optimal for them, generally though most plants prefer a slightly acid growing environment. An ideal pH level is between 5.5 and 7. Changing the pH level too quickly is not a good idea as this will stress the plant out too much. Generally, just make sure that the pH level is between the range above.

The controller has the benefits that the pH level of the water is constantly being reported ( every five seconds ). The user can set limits on the pH levels and so there will be a visual cue on the main screen if the pH level fluctuates outside of the predefined levels. Without the controller the user would need to use an external device, which may be carried out only a few times a day. By which time the pH could be too out of range causing damage to the plants.

**2.3 Electrical Conductivity: EC Level**

After pH level, the second most important element to measure in the water solution is the EC level. EC is a measure of the concentration of nutrients in the solution. However as each nutrient has a different salt content, you could have a high concentration of one nutrient and a lack of another. According to Fernandez [4]

> *“Of paramount importance are the ions that determine pH which have conductivities hundreds of times larger than other ions”.*

Therefore, EC should always be measured at a constant pH. The controller allows the user to know the pH when measuring the EC and so this requirement will always be met.

The reason why the EC level is so important is outlined here:

*"The electrical conductivity can tell you if your solution has lost nutrients or water due to evaporation, if measurements are done at the exact same pH value. The EC should be measured when the solution is prepared and three times each day after then. If your solution's EC becomes too high, you can add water to lower it to the original value. If EC becomes too low (70% of original value), you should not add nutrients. This means that your solution has been substantially changed in composition by the plant and it needs to be disposed off and a fresh one needs to be prepared."*[4]

Just like the pH level the EC is constantly being reported back to the user ( every five seconds ), with the ability for them to set alarms at predefined levels. A visual cue will be given if the EC level falls out of some predefined range. Again, without the controller, the user would need some external device to do this and ensure it is carried out a few times a day.

Why should we use EC instead of TDS?

*"EC stands for Electrical Conductivity and is measured in mS/cm or millisiemens per centimeter. TDS stands for Total Dissolved Solids and is measured in PPM or parts per million. TDS is acquired by taking the EC value and performing a calculation to determine the TDS value. Because TDS is actually a calculation it is really only a guess at what the nutrient concentration is. On top of that, there are three different conversion factors to determine TDS and different manufacturers use different conversion factors. In other words you could test the same solution with two different meters and get two totally different readings. But the EC is read the same by all meters the only difference is the conversion factor."[3]*

### 2.4 Water Temperature

In order for the nutrients to be properly absorbed by the plants the temperature of the water should be in the range of 18-26°C. If the water temperature is too low, then a water heater can be used to warm it back up again.

The controller reports the water temperature in real time back to the user on the main screen. The user has the ability to set the lower and upper limits of the water temperature. The user, if they are using a water heater, can set it so that if the temperature falls too low, then the water heater will switch on. It switches off if the water becomes too warm, ensuring that the temperature remains constant. This feature is more useful if the user is using the controller outside, like in a greenhouse.

## 2.5 Air Temperature

The controller reports the air temperature in real time back to the user on the main screen.  Like the other main screen reports, if the value falls outside of a user specified range, a red light will show on the screen.  If the temperature is within the range it will be green. The controller has currently no way of controlling the air temperature.  A possible extension of the project would be that a fan could be controlled if the air temperature is too high.

## 2.6 Light Level

Not just a hydroponics problem, but light level is a general problem of growers.  If light levels are too low, the controller can be used to turn on a grow light. As the system will control external remote plugs, it means that the user has flexibility in the devices they choose.

As shown above, the main advantages of the controller are that it allows the most important aspects of growing to be automated and alerts the user immediately if some element is out of a preset range.

# 3 CONTROLLER HW REQUIREMENTS & DESIGN

In order for the controller to be used for growing, a study was made on how people actually use hydroponic systems and these will be the requirements with which I will base the HW and SW on. In addition to this, thought has been given to the people that will actually make the hydroponic system and how the system can be made easy for them. The justification for the usage of the HW and SW is described in this and the following chapter.

The HW platform will be the Arduino Mega 2560. This will be used as it is open source and it provides enough pins to support a 3.2” TFT screen and numerous sensors. In addition, it provides access to pins which support interrupts making it ideal for this project. The available space for SW is 250KB, which is more than enough for this project. It supports EEPROM storage meaning that any user data can be permanently stored.

### 3.1 DHT11 Temperature sensor

The controller must be able to measure the air temperature. For this a DHT11 sensor will be used. This was chosen as its temperature range falls well into the range required for growing food, which is 0-50°C It also has a temperature accuracy of ±2°C. However, this can be improved by using an offset in the SW to configure it to the actual temperature using a mercury based thermometer. The sensor can only get new data once every 2 seconds. This should not be a problem though for hydroponics. The chances of a big fluctuation in air temperature within two seconds is not very likely.

The full technical details are[13]:

- Operates between 3 to 5V
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-50°C temperature readings ±2°C accuracy
- No more than 1 Hz sampling rate (once every second)

Long time exposure to strong sunlight and ultraviolet light may debase the performance. For this reason, the DHT11 will be within the enclosure itself.  The circuit does not dissipate much heat and tests done with a mercury thermometer show that the temperature is equal to that outside the enclosure. This way we can get the actual temperature and not that, which would be higher, for example if the sun was shining directly onto it.

When soldering the DHT11 to the circuit, the temperature should be below 260°C.  Also, the sensor will not work well under dew conditions.

The pins of this sensor are described in Table 1.

*TABLE 1.  DHT11 Pin Descriptions*

| Pin | Description |
|---|---|
| GND | Ground |
| VCC | Between 3 to 5.0v |
| Data | Used to get the temperature. |

The DHT11 used in this project came with its own resistors attached. This has the benefit as it will use less space on the circuit board and makes it easier for end users to build.

### 3.2 DS18B20 Water Temperature Sensor

The temperature sensor used for the water will be a DS18B20 [14].  It will be encased in plastic to allow it to be submerged indefinitely. These are widely sold and so any user will be able to get hold of these. Many come with a 3m long cable.  This is handy as then the controller does not need to be directly next to the water tank, giving extra flexibility to the user. Its temperature range is -55°C - 125°C, so more than enough for growing food.  The accuracy is ±0.5°C over the range -10°C to 85°C.

The resolution of the temperature sensor is user configurable to 9,10,11 or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C and 0.0625°C. In this project 9bit resolution is used. A half a degree was considered enough accuracy for hydroponic food growing.

The sensor communicates over a 1-wire bus and so by definition requires only one data line. The used pins are given in table 2.

*TABLE 2. DS18B20 Pin Descriptions*

| Pin | Description |
| --- | --- |
| GND | Ground |
| VCC | Between 3 to 5.5v |
| Data | Used to get the temperature |

**3.3 Photo-resistor**

A photo-resistor is used to measure the amount of light available. If the light is too low then the grow lamp can be signalled to switch on. The photo-resistor will not be attached to the circuit board but will use the pins of the Arduino board. The reason for this is that attaching the resistor to the circuit board will save some space. Other benefits include the fact that the resistor will need to be on the outside of the enclosure and so it is easy to just attach some wires to it.

**3.4 DS3234 Real Time Clock**

The DS3234 is the RTC and is controlled via a SPI interface. It comes with a built in power sense circuit that detects power failures and automatically switches to battery backup, which is ideal if the system is powered down. The clock is a central component of the controller as the user will want to power up the pumps at certain intervals and leave them on for a required time. The accuracy of the timer is ±2ppm between 0°C and 40°C. It also supports alarms. They are not used in this project but this ability gives some extra flexibility when expanding the features in the future. The primary driver for selecting this device was its accuracy, achieved via its temperature compensated crystal oscillator as described in the data sheet [15]. The pins on the RTC are shown in table 3.

*TABLE 3. DS3234 Pin Descriptions*

| Pin | Description |
|---|---|
| GND | Ground |
| VCC | Between 3 to 5.5v |
| SS | (Slave Select) - the pin on each device that the master can use to enable and disable specific devices. |
| MISO | (Master In Slave Out) - The Slave line for sending data to the master |
| MOSI | (Master Out Slave In) - The Master line for sending data to the RTC. |
| SCK | (Serial Clock) - The clock pulses which synchronize data transmission generated by the master |
| SQW | (Square Wave) Not used in this project. |

The Arduino SPI interface guidelines were followed [16], when programming the device to ensure robust usage.

In an earlier design of the controller, a cheaper DS1307 RTC was used, the time was not kept properly.  A few minutes per day were lost irregularly, which made it difficult to compensate for. Because the timer operation is important for this device, it was decided it would be better to buy a more expensive and reliable one.  The DS1307 RTC had the advantage that it used the I2C to control it, which meant that less pins were needed. However, timer accuracy is more important.

### 3.5 TFT Touchscreen

The user will interact with the controller through this touchscreen.  A 3.2" Sainsmart touchscreen was chosen for this purpose.  The size is not too small so that it would be difficult to control and it is not too big as to make the controller itself too big.

The compatible shield that comes with this has been taken into use.  This helps in the building as then the pins will connect easier with the Arduino.  It has support for a SD card too, so that images can be kept on there for making the UI look better.  This project does not make use of the SD card, but this is something that will be used as further study to make the UI more pleasurable to use.

Other options were to control the system only via a mobile phone or computer.  However this makes it less independent and means that many versions of SW would need to be written for numerous OSs.  By using the TFT it makes the HW available to a far wider audience.  The TFT does not cost too much and so overall it adds value to the product. However a future development may be to let the device connect to a network, in which a computer could be used to analyse the data which the controller has generated.  It could also be controlled remotely, via the net, when the user is away from home.

The graphics libraries I will be using are provided by Henning Karlsen [17].  These libraries come with decent documentation and give me just the functionality that I need.  Using external libraries will lower the development time and increase quality as these libraries are used by numerous people and are open sourced.

More information about the SW libraries used in conjunction with the TFT HW are discussed in the SW section.

### 3.6 Transmitter & Plugs

A transmitter will be required to send information to the wireless plugs.  The transmitter will need to operate at 433.92 MHz as this is an open frequency in Europe ( and many other parts of the world too) for data transmission. As stated by the ECC Recommendation 70-03 [5].

It is also the frequency that wireless plugs operate at in Europe and so in order for the controller to be compatible with the greatest arrange of plugs, this frequency must be used.

Other options could be used over a transmitter and wireless plugs, for example using a relay.  It was decided not to use this approach however as it puts a lot of people off building their own devices.  One of the principals of the project is to increase the use of and success rate of hydroponics.  In many cases people may not be familiar with electronics and so messing around with mains electricity may be too scary.

Using wireless plugs also gives the advantage of positioning the controller itself in more locations, so not necessarily right next to the system itself.

There are no real drawbacks to this system, except if it is to be used in some countries, most notably the USA, as the 433 MHz is more restricted there.  However, the transmitter used can broadcast on the 315 MHz frequency, too.  One future development could be to switch the frequency of the transmitter based on a user's setting.

The wireless plugs used in this project are the NEXA PBR-2300, although any self programmable plugs that operate at the 433 MHz frequency can be used.

The plugs will be connected to lights, a water pump and a water heater.  The controller will turn them on and off when needed based on the environment or user settings.

When testing indoors and with no antenna attached, the plugs could be switched on and off at a range of 10m.  With an antenna, distances of up to 100m could be achieved. This is more than enough range for this project.  During the testing of this project, no antenna was used.

### 3.7 EC Circuit

For measuring the EC levels of the water an Atlas Scientific stamp 3.1 [18] has been used. This is used as their level of quality is high and there is a lot of documentation on how to use them. The circuit is small and fits easily into a breadboard or a small shield that would be used on an Arduino. There are also some debugging LEDs, which can be used as an indication that commands are indeed being sent to the circuit. The end user will not see these but they are useful in the development life cycle.

As described in table four, Atlas Scientific sell three different types of circuits, each are especially designed for certain kinds of water. This project will use the sensor K0.1 as drinking water will be used for watering the plants. The pins used are described in table 5.

*TABLE 4. EC Circuit Sensor Types*

| Sensor Type | Type of water | Sensor Range |
|---|---|---|
| K0.1 | Pure water and drinking water | $11 \mu s / cm$ to $3{,}000 \mu s / cm$ |
| K1.0 | Fresh water to brackish water | $1{,}300 \mu s / cm$ to $40{,}000 \mu s / cm$ |
| K10 | Salt water | $36{,}000 \mu s / cm$ to $92{,}000 \mu s / cm$ |

*TABLE 5. EC Pin Descriptions*

| Pin | Description |
|---|---|
| GND | Ground |
| TX | The output delivers asynchronous serial data in TTL Rs-232 format. |
| RX | The input receives asynchronous serial data in TTL Rs-232 format. |
| VCC | Operates at 5.0v |
| PRB | Connects to the BNC probe data connection |
| PRB GND | Connects to the BNC probe GND connection |

The circuit itself requires only two data lines and operates at 5v, which ensures accurate readings. The circuit can be stored at ranges between -20°C up to 125°C.

It is important to note that when testing, the BNC connector and the circuit must be connected directly on the breadboard and not using any wires, otherwise the noise will cause inaccurate readings.  As this will be soldered onto the circuit board, the device should not suffer from any of these problems.

### 3.8 pH Circuit

The 5.0 version of the pH circuit from Atlas Scientific has been used [19].  It has support for full pH readings (.01 to 14.00) and the accuracy is within two significant figures.  It also offers temperature dependent or independent readings. Even though, only temperature readings will be used as the controller comes with a water temperature sensor. The circuit itself has a storage temperature range of -40°C to 125°C. The pins used are described in Table 6.

*TABLE 6. pH Pin Descriptions*

| Pin | Description |
|---|---|
| GND | Ground |
| TX | The output delivers asynchronous serial data in TTL RS-232 format. |
| RX | The input receives asynchronous serial data in TTL RS-232 format. |
| VCC | Operates on 3.3v to 5.5v |
| PRB | Connects to the BNC probe data connection |
| PRB GND | Connects to the BNC probe GND connection |

The default baud rate is 38400, 8bits no parity.  However, the circuit supports eight different baud rates:

1. 300
2. 1200
3. 2400
4. 9600
5. 19.2k
6. 38.4k
7. 57.6k
8. 115.2k

The pH circuit is very sensitive and touching it can cause an inaccurate reading for about five minutes. As the circuit will be enclosed with the hydroponics controller, this should not pose any problems.

### 3.9 Enclosure

The enclosure is only there for the look and feel and to protect the equipment from any water that might splash onto the circuit. This project has used a basic plastic enclosure, which will have holes cut out in the front and back for the TFT screen and any wires.

### 3.10 HW Layout

An Arduino shield will be designed and made that will connect all the parts of the HW together and hold them firmly in place. Below is a diagram for that. Everything will be designed using Fritzing, as it is free with no restrictions and allows the manufacture of PCB boards. The PCB board used in this thesis was made by Fritzing.

A circuit board was made up as this will make the hydroponic controller more stable and less likely for the reading to be affected by interference. It also eliminates the need for messy wires and for any HW errors to occur; such as a wire coming lose.

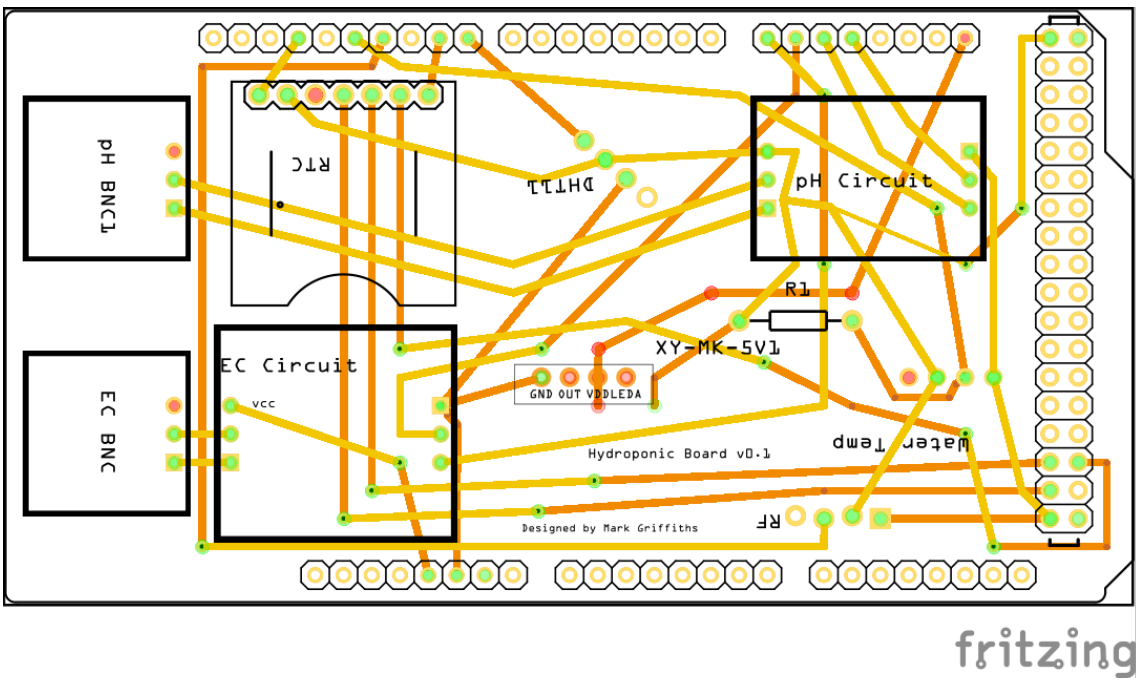


*FIGURE 1. The Hydroponic Circuit Board*

The circuit used in this project is shown in Figure 1. It uses two copper layers, with a copper layering on top to minimise the risk of interference from the transmitter and receiver.  The TFT will just connect using a molex connector to the pins that it needs.  The circuit was designed so that the connections would not cross too much and would be as short as possible. Thought was also given to the build of the system, for this reason the pH and EC BNC connectors are at the back. This way the system will look better as all the connections will be at the back of the device.  The power lead for the Arduino will sit just below the BNC connectors.

One of the main challenges when designing the HW, was that of space. All of the components just about fit on the board.  If any extra HW would be needed then a bigger board would be needed too.  This may not cause too much problems if the enclosure has enough space to accommodate it.  However a bigger board will also cost more, one goal of the project was to make the controller as cheap as possible.

Almost all of the Arduino interrupt and digital pins are now in use.  Any extra expansion to the HW would require a rethink into which processor and board should power the device.  Since the creation of the project, much more powerful Arduino boards and other similar boards have entered the market. In this sense a rethink of the HW layout would need to be reconsidered.

### 3.11 Overall View of the System

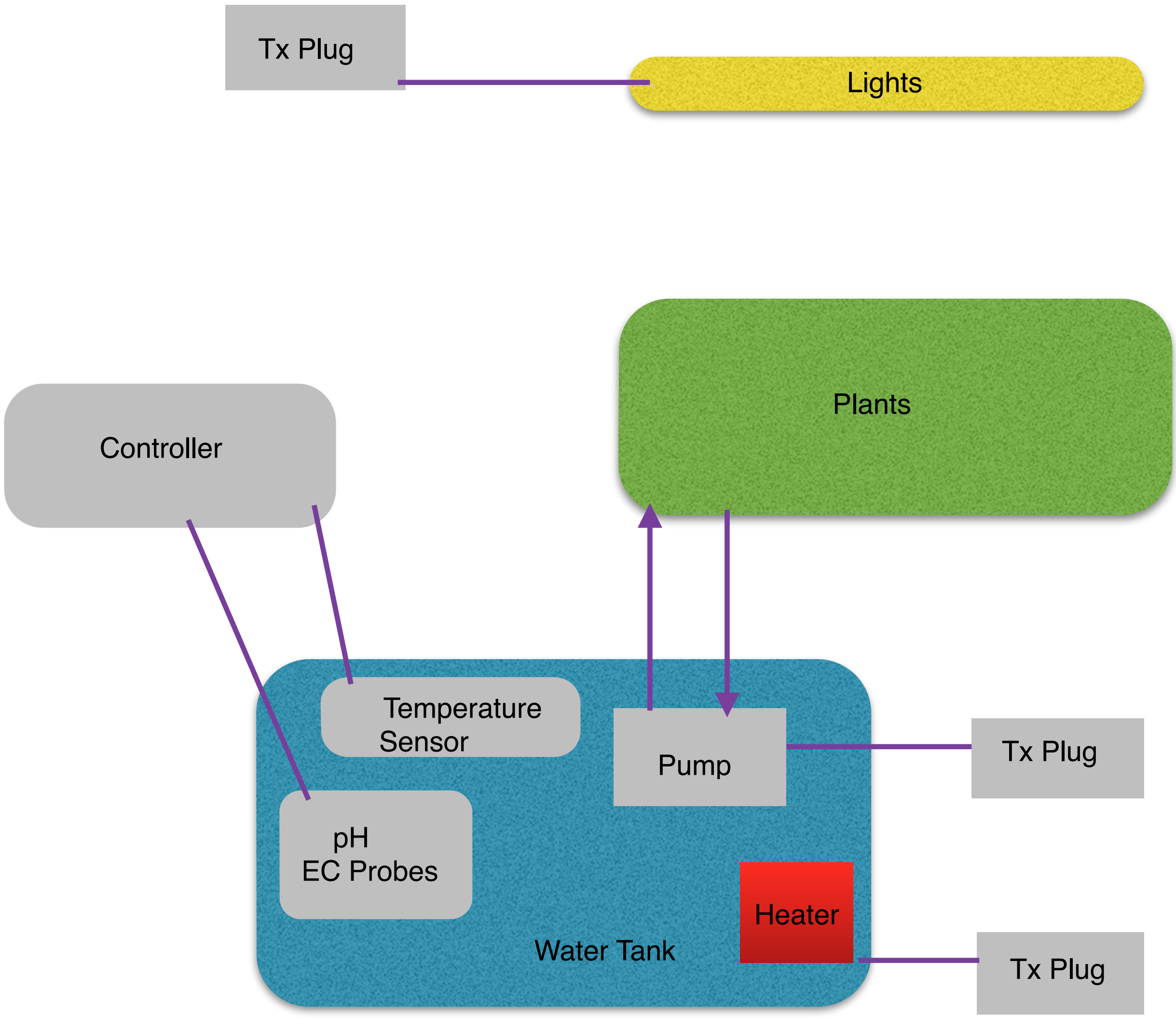


*FIGURE 2. Overall view of the system*

Figure 2 shows how the Controller fits into the system as a whole. As can be seen the external HW like the water pump, lights and water heater are not directly connected to the controller adding flexibility into the system. The pump is used to pump water across the roots of the plants.

# 4 SW REQUIREMENTS AND DESIGN

Developing the software was the most time consuming part of making the controller, as this is where all the logic is. The language used is C++ and the Arduino IDE was used as the compiler to flash the SW on to the HW. The bulk of the SW has been built from scratch although some open source libraries have been used when interfacing with the HW.

The software not only needs to read the sensor information but also needs to be used to calibrate the pH and EC probes as well as the other sensors, to provide details of the information to the user, to function as an alert to the user if some setting e.g the pH level is too high or low and to control certain other devices, like lights and water pumps.

The controller itself works without the aid of a computer and all inputs come from an intuitive touch screen which gives warnings to the user when necessary.

The SW has been designed in a way that each HW component has its own library and the UI, controller and engine have been separated out from each other. The controller is simply the loop function which is the main function of the Arduino.

The reason for this split is that in the future a different TFT screen may be used. Then only the UI section would need to be rewritten. Likewise for the HW libraries, if in the future a different HW device is used, then only that library will need to be rewritten. If needed, just an interface wrapper would need to be constructed.

### 4.1 UML Overall Package Diagram

Figure 3 shows the libraries used in the SW and their relationships with each other. The packages Engine, Main Loop Controller and UI are self made for this project. All other packages are third party packages, all external packages are used to control some specific HW. The packages are discussed in more detail in the next few chapters.

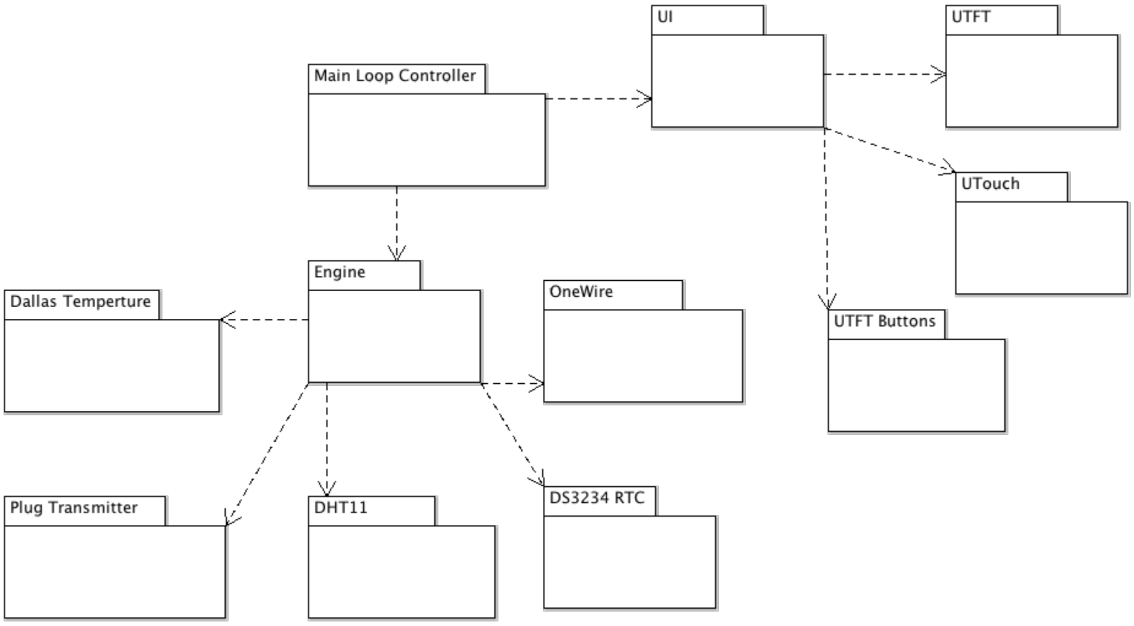


*FIGURE 3. Package Diagram*

### 4.2 UI Package

The UI package is where most of the code and logic for the controller is. This package is responsible for drawing the UI, as well as capturing the touches from the screen. It will raise an alarm to the UI if some values are out of range. The values are passed to the UI via the controller from the engine.

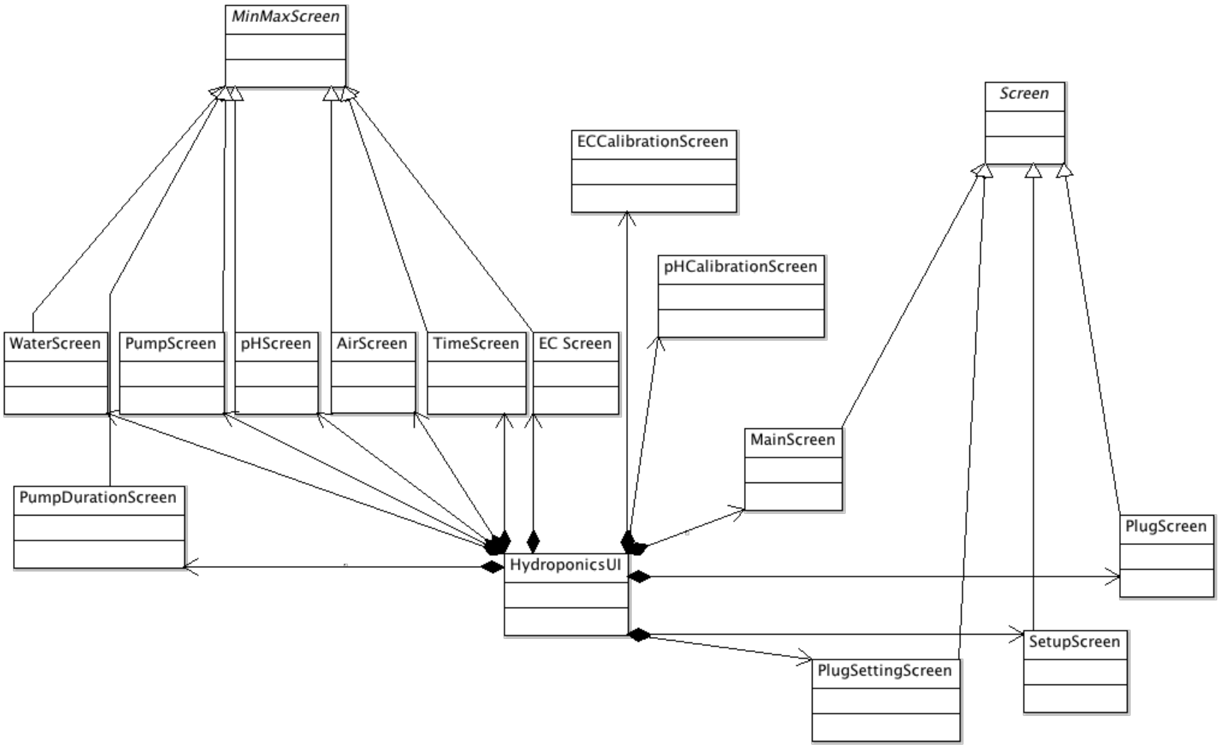


*FIGURE 4. The UI Class Diagram*

A UML class diagram of the UI package is shown in Figure 4. Each of the classes, except the base classes represents one screen. Therefore it has one specific job to do related to the functionality. The UI classes are described in more detail below.

First, there are two base classes. MinMaxScreen and Screen.

**MinMaxScreen**

This is a base class to many other screens. It can be used when the screen to be used requires that only a min and max value are given. The screens in those cases are pretty generic and it is this generic data and layout that is encapsulated within this class. The derived classes then only need to give the specific information, like the text, which shall be on the screen.

**Screen**

This base class encapsulates the button handling logic of screens that will handle settings or in which a lot of buttons will be displayed on the screen.  It makes it easier for derived classes to handle the button logic,

The following classes represent a screen and will draw to the screen and perform some function.

**Water Screen**

This screen allows the user to set the lower and higher thresholds for the water temperature. Upon exiting from the screen, the min and max values are saved to the EEPROM.

**Air Screen**

This screen allows the user to set the lower and higher thresholds for the air temperature.  Upon exiting from the screen, the min and max values are saved to the EEPROM.

**pH Screen**

This screen allows the user to set the lower and higher thresholds for the pH levels.  Upon exiting from the screen the min and max values are saved to the EEPROM.  This screen also draws the pH calibration button. When the user presses this, they will open the pHCalibrationScreen.

**EC Screen**

This screen allows the user to set the lower and higher thresholds for the EC levels.  Upon exiting from the screen, the min and max values are saved to the EEPROM.  This screen also draws the EC calibration button. When the user presses this, they will open the ECCalibrationScreen.

**Time Screen**

From this screen the user can change the time. They are given two options, hour and min, In this way it is similar to min and max. This is the reason why this screen derives from MinMaxScreen. On exit the values are not saved to the EEPROM but the time is saved to the DS3234 circuit.

**Pump Screen**

The user sets on this screen the interval at which the pump will come on. Again hours and mins are the options. If the user selects 0 hours and 8mins, this will mean that the pump will switch on every 8 minutes.

**Pump Duration Screen**

The user sets on this screen the interval at which the pump will stay on. Again hours and mins are the options. If the user selects 0 hours and 4mins, this will mean that the pump will stay on for 4 minutes. If the duration period is longer than the interval, then the pumps will never switch off.

**pH Calibration Screen**

This screen presents the user with the following options:

- **Reset**

    This will send a reset command to the device and all saved data will be lost. The device will need to be re-calibrated for it to function properly.

- **Info**

    This will query the device firmware version and firmware date.

- **Read**

    This will return one read from the sensor.

- **TRead**

    This will return one temperature calibrated read from the sensor.

- **Stop**

    This will stop the circuit from asking the pH probe for any data.

- **Start**

    The circuit will start asking the pH probe for data again. If it has been already started, then this button has no effect.

- **Calibrate**

    This screen walks the user through calibrating the pH circuit. They are given instructions for each stage. Each stage takes a few minutes and the library simpleTimer is used to achieve this. The screen is asynchronous and does not block the updating of other HW devices, e.g. turning on the pump.

**EC Calibration Screen**

This screen presents the user with the following options:

- **Reset**

    This will send a reset command to the device and all saved data will be lost. The device will need to be re-calibrated for it to function properly.

- **Info**

  This will query the device firmware version and firmware date.

- **Read**

  This will return one read from the sensor.

- **TRead**

  This will return one temperature calibrated read from the sensor.

- **Stop**

  This will stop the circuit from asking the EC probe for any data.

- **Start**

  The circuit will start asking the EC probe for data again. If it has been already started then, this button has no effect.

- **Calibrate**

  This screen walks the user through calibrating the EC circuit. They are given instructions for each stage. Each stage takes a few minutes and the library simpleTimer is used to achieve this. The screen is asynchronous and does not block the updating of other HW devices, e.g. turning on the pump.

### Plug Screen

This screen allows the user to calibrate the plugs for the lights, pump or heater. Upon pressing one of the buttons, the user is asked to press a button on the plugs remote. The device will then capture the plugs ID and will use this when transmitting information to the plug to switch on or off. The user can also access the screen PlugSettingsScreen.

### Plug Settings Screen

This screen allows the user switch the pump, lights and heater on manually. This is useful if the user wants the lights off despite there being too little light etc.. This is just a way to give the user some extra control over the device.

### Main Screen

This is where the controller will spend most of its time. The main screen displays information about the environment, like the EC and pH levels as well as the air and water temperature. The alerts are shown here too, for example if one of the levels is out of range. The time is also displayed. From here the user can access the SetupScreen, WaterScreen, AirScreen, ECScreen or pHScreen.

### Setup Screen

The setup screen is just a menu screen where the user can navigate to other screens. It accesses all of the above mentioned screens.

### Hydroponics UI

This class is the brains of the UI package. This is where the touch screen captures are passed over so that the right screen is activated. The controller will initialise this at boot and in turn this class will initialise all of the other classes in the package.

The following piece of source code shows how each screen is activated, `CheckForScreenTouch` is called every 100ms from the controller. The UTouch library is used to see if the TFT screen has been touched.

```
void HydroponicsUI::CheckForScreenTouch()
{
  if (myTouch.dataAvailable())
    {
      switch(whichScreen)
      {
        case MAIN_SCREEN:
        {
          handleButtons( iMainScreen->handleScreen() );
          break;
        }
         case ABOUT_SCREEN:
        {
          handleButtons( iAboutScreen->handleScreen() );
          break;
        }
          case SETUP_SCREEN:
        {
          handleButtons( iSetupScreen->handleScreen() );
          break;
        }
```

```
in the following function the actual screen is drawn.
```

```
void HydroponicsUI::handleButtons( int aHandleWhichButton )
{
     switch(aHandleWhichButton)
     {
     case AIR_BUTTON:
```

```
        {
        iAirScreen->drawScreen(whichScreen);
        whichScreen = AIR_SCREEN;
        break;
        }
    case PH_BUTTON:
        {
        ipHScreen->drawScreen(whichScreen);
        whichScreen = PH_SCREEN;
        break;
        }
```

**4.3 UTFT Library**

A freely available open sourced UTFT library was used to put text and graphics onto the screen [5].

From the SW point of view, we use some global variables throughout the whole code. For the TFT screen these are defined as:

```
//Global definition for whole of the UI
UTFT myUTFT(ITDB32S, 38,39,40,41);
UTouch myTouch(6,5,4,3,2);
UTFT_Buttons myButtons(&myUTFT, &myTouch);
```

Those numbers represent the pins that are used by the screen. After doing this, the libraries for the TFT screen are ready to use.

With this library I can initialise the screen , set its colour, clear and fill it. The full API can be found in [5] and it comes as a pdf file along with the code.

The following code example shows how this SW library is used.

```
myUTFT.clrScr();
myUTFT.setBackColor(0, 0, 255);
myUTFT.print("Reset", 55, 30);
```

In the above example the screen was cleared of all other text and pictures, the background colour was set to blue and the text “Reset” was printed at the given x,y coordinates.

**UTouch Library**

The UTouch library [20] is used for telling when the screen has been touched and the location of the touch. Using this library, we tell the orientation of the screen; in this project it is always landscape.

An example of setting up the screen is

```
myTouch.InitTouch();
myTouch.setPrecision(PREC_MEDIUM);
```

Here the myTouch object was initialised and the precision of the touch screen has been set to medium. After this, the UTouch library is not used directly but via the UButtons library as described below.

**UButtons Library**

The UButtons library helps simplify the UI code by easily drawing buttons on the screen. It notifies when a certain button has been pressed by returning the ID of the button back. It can enable/disable buttons and set text and icons to them. This is the most used library in the UI code.

An example would be the following:

```
resetButton = myButtons.addButton( 10, 10, MAIN_BUTTON_X,
MAIN_BUTTON_Y, mainButton);
```

This code creates a button at x 10 and y 10 of size MAIN_BUTTON and uses the image mainButton to make the button. resetButton is the ID of this button and can be used to know when the button has been pressed.

```
int pressed_button = myButtons.checkButtons();

if (pressed_button==backButton)
          {
          handleExitScreen();
          return EC_BUTTON;
          }
if (pressed_button==resetButton)
          {
          factoryReset();
          }
```

In the code snippet above we just check which button has been pressed and preform some action based on this. Using this library significantly improved the readability and maintainability of the code.

### 4.4 Engine

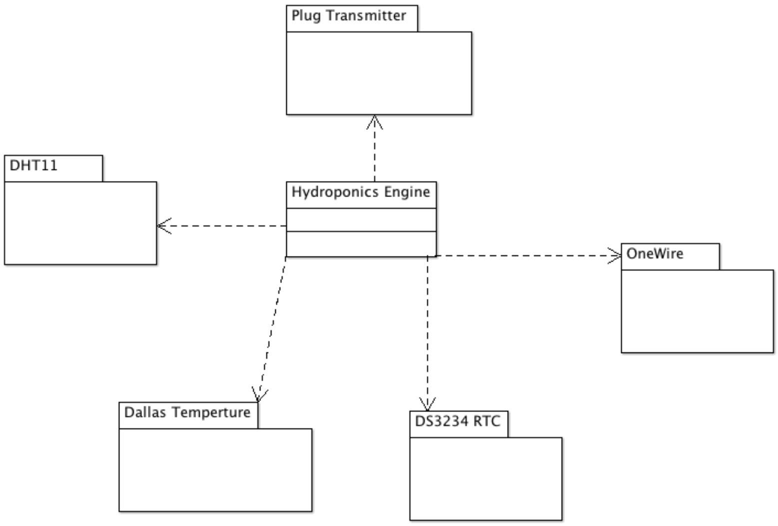


*FIGURE 5. The Engine External Libraries*

The main task of the Engine is to get the data from the HW sensors, e.g. the water temperature and update the sensor struct. It is the sensor struct that is passed from the Engine to the UI, via the controller that is used when updating the UI. The sensor struct is only passed if there is some updated information from the HW, like if the temperature has changed. The controller does this check as described here.

```
if( Hydro_Eng.doesScreenNeedUpdating() )
   {
   //Update the UI of the latest Sensor Info

Hydro_UI.updateInfoFromSensors( Hydro_Eng.getSensorInformat
ion() );
   //Update the screen with the new info.
   Hydro_UI.refreshScreen();
   //Inform the engine that the UI has been updated.
   Hydro_Eng.screenUpdated();
   }
```

The struct containing the updated sensor data is given to the UI in the following.

```
Hydro_UI.updateInfoFromSensors( Hydro_Eng.getSensorInformat
ion() );
```

A refresh of the screen is then done to ensure that the UI always displays the most up to date information.

The Engine Library is made up of just one class that interfaces with the HW libraries. As can be seen in Figure 5, each HW component has its own library and is described in more detail in the following sections. Each of the libraries is external and all are open source. [21]

### 4.5 Plug Receiver and Transmitter Library

The library used to control the receiver and transmitter is called 433MHz for Arduino [22].This package contains several small libraries for Arduino 1.0 which add communication capability with some radio controlled (433MHz / 434MHz) domestic appliances. The libraries can be used for easy home automation using cheap, off-the-shelf components.

This library was used in order to turn on and off the remote plugs. Each plug must be set to a function, for example a water pump. This can be done by the user via the UI.

To get the details of the plug, the remote of the plugs needs to be used. When on the plug screen, you just need to press a button and the device will capture the plug details. In code this is done via a callback. When the transmitter picks up information, it will capture and store that, as shown in the code example.

First the receiver is set up for the correct pin and a callback waitForReceiver is given. The number one, given as the second parameter below, means that this will be done only once. No repeats are allowed.

```
NewRemoteReceiver::init(RECEIVER_PIN, 1, waitForReceiver);
```

When the user presses the button, the callback is called. In the callback the information is stored in EEPROM for later use.

```
setAddresstoEEPROM(receivedCode.address);
     if(plugDevice.lights)
     {

EEPROM.write(EEPROM_LEARNED_LIGHTS_UNIT,receivedCode.unit);
          plugDevice.lights = false;
     }
     if(plugDevice.pump)
     {

EEPROM.write(EEPROM_LEARNED_PUMP_UNIT,receivedCode.unit);
          plugDevice.pump = false;
     }
     if(plugDevice.heater)
     {

EEPROM.write(EEPROM_LEARNED_HEATER_UNIT,receivedCode.unit);
          plugDevice.heater = false;
     }
```

**4.6 DS3234 RTC Library**

Using this library makes time management very easy and through an intuitive API, which is now described. The DS3234 library [23] is initialised by passing the SS pin number.

```
DS3234 rtc(RTC_PIN);
```

After this, the time can be set based on the user input as such. The zero means how many seconds. The user cannot choose this and so it is set to zero.

```
rtc.setTime(aHour, aMin, 0);
```

When the time is queried, the code is the following.

```
void HydroponicsEngine::GetTime(int& aHour, int& aMin )
{
     clock = rtc.getTime();
     aHour = clock.hour;
     aMin = clock.min;
}
```

**4.7 Dallas Water Temperature Library**

The water temperature library [24] is first asked to get the temperature using the following command:

```
waterTempSensor.requestTemperatures();
```

After this, we can get the current temperature of the water by calling

```
int temp = waterTempSensor.getTempCByIndex(0)+
waterTempCal;
```

Here waterTempCal is a calibration constant. During testing, I called this function and compared it with a mercury thermometer and noticed that it was out by a degree or two. Trusting the mercury thermometer more I added a constant to it which would change the temperature based on tests. For each different HW that is used, this constant will need to be changed based on the results of testing.

**4.8 DHT11 Air Temperature Library**

The DHT11 library [25] is very straight forward to use. In order to get the temperature we just need to call the read function, with the pin that the DHT11 is connected to.

As in this example:

```
int chk = DHT11.read(DHT11_PIN);
sensorInfo.airTemp = DHT11.temperature();
```

After this call, we have the temperature in Celsius located in sensorInfo.airTemp and we can update the UI and anything else as needed. If we wanted the temperature in Fahrenheit, we would call

```
DHT11.fahrenheit();
```

The library also supports getting the temperature in Kelvin and for getting the dew point. This SW does not make use of those features.

### 4.9 EC Probe

The EC probe does not use its own library as all communication with it is done over the serial port. In essence, this is its library. The baud rate is 38400.
The full manual and data sheet are available here [18].

The following commands listed in Table 7 are used to control the EC circuit. For example, the command **R** - This tells the EC circuit to return a single EC reading. This instruction takes 1000ms to complete. An example is the following.

```
inputstring = "R\r";  //Command to get reading
Serial3.print(inputstring); //send command to sensor.
```

All commands must end with a carriage return. This tells the EC circuit that we have sent our command.

For most commands the circuit will respond with the serial ports callback function. Assuming we are using serial port 3, this would be `void serialEvent3()`. This callback function is only called every time the main loop function has ended.

After sending the command **R**, the circuit will respond with:

EC,TDS,SAL<CR>

Where:

EC is the electrical conductivity in µs/cm

TDS is Total Dissolved Solids (referenced to Kcl)

SAL is Salinity (Practical Salinity scale 1978) expressed as a whole number only.

As with sending the command, each response back ends with a carriage return (<CR>). This way my software knows that the full response has been received.

In the serialEvent3 function we get the results of what the circuit has done. In some cases it gives us the data we asked for and in others it is just a conformation that everything went as expected. That function is shown here:

```
void serialEvent3()
{
sensorstring="";
while(Serial3.available())
        {
        char inchar = (char)Serial3.read();
        sensorstring += inchar;
        if(inchar == '\r') {sensor_stringcomplete = true;}
        }



Serial.print(sensorstring.length());

if(sensorstring.length() == 9) //length of EC response.
{
     ECLevelString = sensorstring.substring(0,3);
}
```

```
//check which data we are waiting for.
if(waiting_for_info && sensor_stringcomplete)
     {
     drawInfoScreen();
     waiting_for_info = false;
     sensor_stringcomplete = false;
     }
if(waiting_for_single_reading && sensor_stringcomplete)
     {
     drawReadingScreen();
     waiting_for_single_reading = false;
     sensor_stringcomplete = false;
     }

}
```

*TABLE 7. EC Circuit Commands*

| Command | Function | Callback | Description |
|---|---|---|---|
| R | Return a single EC/TDS/salinity reading. Temp assumed to be 23C | EC,TDS,SAL | EC is the electrical conductivity in µs/cm<br>TDS is Total Dissolved Solids (referenced to Kcl)<br>SAL is Salinity (Practical Salinity scale 1978) expressed as a whole number only. |
| TT.T | Like R but returns a temperature compensated reading. | EC,TDS,SAL | EC is the electrical conductivity in µs/cm<br>TDS is Total Dissolved Solids (referenced to Kcl)<br>SAL is Salinity (Practical Salinity scale 1978) expressed as a whole number only. |

| Command | Function | Callback | Description |
|---|---|---|---|
| C | Returns a continuous reading every 1000ms at the temperature previously used. | EC,TDS,SAL | EC is the electrical conductivity in µs/cm<br>TDS is Total Dissolved Solids (referenced to Kcl)<br>SAL is Salinity (Practical Salinity scale 1978) expressed as a whole number only. |
| E | Stops all readings and the circuit enters standby mode. | None | The E.C. Circuit will respond by ceasing data transmission. There is no ASCII response to this instruction |
| X | Does a factory reset of the circuit. | Factory reset | |
| I | Returns Information about the circuit. | E,V3.0,4/11 | E =E.C. Circuit<br>V3.0= Firmware version<br>4/12= Date firmware was written |

The EC circuit must be calibrated by the SW in the following order:

1. Set Sensor type
2. Calibrate for a dry sensor
3. Calibrate for a high side reading
4. Calibrate for a low side reading.

All of the data is stored on the circuits EEPROM memory and will not be lost during power down.

*TABLE 8. EC Circuit Calibration Commands*

| Command | Function | Callback | Description |
|---|---|---|---|
| P,1 | Tell the circuit it is of type 0.1 | K0.1 | All OK |
| Z0 | Informs the E.C. Circuit to calibrate for a dry Sensor. | Dry Cal | Calibration Done |
| Z30<CR> | High side calibration for 3,000 solution | 90,000 µs/cm cal | Calibration Done |
| Z2 | Low side calibration for 220 solution | 220 µs/cm cal | Calibration Done |

The commands used only during calibration are listed in Table 8. When calibrating we wait five minutes before sending the reading to the circuit. It takes this amount of time for the EC readings to stabilise.

The calibration of the EC circuit is conducted in the class ECCalibrateScreen. During the calibration stage, access to calibration solutions will be needed.
The waiting of the five minute period is asynchronous from a HW and SW point of view. During this time the UI will be updated every three seconds informing the user how long is left of the calibration process. The user will not be able to do anything except wait for the calibration to be done. The updating of the UI is done so that the user does not think that the device has frozen.

During this time though, the SW still polls the HW in order to get the EC level and it will only set the EC level for that calibration when the time has expired.

**4.10 pH Probe**

The pH probe [19], like the EC probe does not use its own library as all communication with it is done over the serial port. The baud rate is 38400.

The following commands are used to control the pH circuit. For example **R** - tells the pH circuit to return a single pH reading. This instruction takes 378ms to complete. An example is the following.

```
inputstring = "R\r";  //Command to get reading
Serial2.print(inputstring); //send command to sensor.
```

The pH circuit and the EC circuit are both made by the same manufacturer. As can been seen from the section on the EC probe, there are many similarities between these two probes.

Again all commands must end with a carriage return. This tells the pH circuit that we have sent our command.

For most commands the circuit will respond with the serial ports callback function. Assuming we are using serial port 2, this would be `void serialEvent2().` This callback function is only called every time the main loop function has ended.

After sending the command **R**, the circuit will respond with “XX.XX”. This is the current pH value, e.g 7.5. If the pH circuit reads a pH that is out of range, it will respond with the error message “check probe”.

As with sending the command, each response ends with a carriage return (<CR>). This way my software knows that the full response has been received.

In the `serialEvent2` function we get the results of what the circuit has done. In some cases it gives us the data we asked for and in others it is just a conformation that everything went as expected.

The other commands and responses are summarised in the table below.

*TABLE 9. pH Circuit Commands*

| Command | Function | Callback | Description |
|---|---|---|---|
| R | Instructs the pH circuit to return a single pH reading. | xx.xx | Where xx.xx is some pH value. |
| TT.TT | By transmitting a temperature to the pH circuit a temperature compensated reading will be returned. | xx.xx | Where xx.xx is some pH value.<br>Temperature data will be lost if the circuit loses power. |

| Command | Function | Callback | Description |
|---|---|---|---|
| C | The pH circuit will operate in continuous mode and deliver a pH reading every 378ms until the “e” command is transmitted. | xx.xx | If the pH circuit detects that a pH sensor is not connected or damaged it will respond with the error message “check probe” |
| E | Stops all readings and the circuit enters standby mode. | None | The pH circuit will respond by ceasing data transmission. There is no ASCII response to this instruction |
| X | Does a factory reset of the circuit. | reset | The pH circuit will respond “reset” |
| I | Returns Information about the circuit. | P,V5.0,5/13 | P =pH Circuit<br>V5.0= Firmware version<br>5/13= Date firmware was written |

The pH circuit must be calibrated by the SW in the following order:

1. Calibrate in a pH7 solution.

2. Calibrate in a pH4 solution
3. Calibrate in a pH10 solution

All of the data is stored on the circuits EEPROM memory and will not be lost during power down.

*TABLE 10. pH Circuit Calibration Commands*

| Command | Function | Callback | Description |
| --- | --- | --- | --- |
| S | Instructs the circuit to calibrate for pH7 solution. | 7.00 | All OK |
| F | Instructs the circuit to calibrate for pH4 solution. | 4.00 | Calibration Done |
| T | Instructs the circuit to calibrate for pH10 solution. | 10.00 | Calibration Done |

The commands used only during calibration are listed in Table 10. Calling any of the commands when the pH sensor is not immersed in the correct pH solution will calibrate the pH circuit to an arbitrary value and can lead to significant errors.

When calibrating, we wait three minutes before sending the reading to the circuit. It takes this amount of time for the pH readings to stabilise.

The calibration of the pH circuit is conducted in the class pHCalibrateScreen. During the calibration stage, access to calibration solutions will be needed.
The waiting of the three minute period is asynchronous from a HW and SW point of view. During this time the UI will be updated every three seconds informing the user how long is left of the calibration process. The user will not be able to do anything though except wait for the calibration to be done. The updating of the UI is done so that the user does not think that the device has frozen.

During this time however, the SW still polls the HW in order to get the pH level and it will only set the pH level for that calibration when the time has expired.

### 4.11 The Main Loop Controller

The controller is the main loop of an Arduino sketch.  It acts as a way to link the UI and the engine together.  The best way to explain it is by showing it:

```
void loop()
{
 simpleTimer.run();
 loopCounter++;
 //Check sensor information every 5s ( 0.1s*50 )
 if( (loopCounter%50) == 0  )
   {
   loopCounter = 0;
   //Get/update the sensors information
   Hydro_Eng.pHLevel();
   Hydro_Eng.airTemperatureHumidity();
   Hydro_Eng.dateAndTime();
   Hydro_Eng.waterTemp();
   Hydro_Eng.lightValue();
   Hydro_Eng.ECLevel();
   }

 //If the sensors or time have updated some values we may
need to show
 //those on the UI
 if( Hydro_Eng.doesScreenNeedUpdating() )
   {
   //Update the UI of the latest Sensor Info
```

```
Hydro_UI.updateInfoFromSensors( Hydro_Eng.getSensorInformat
ion() );
   //Update the screen with the new info.
   Hydro_UI.refreshScreen();
   //Inform the engine that the UI has been updated.
   Hydro_Eng.screenUpdated();
   }

 //Any bigger delay and touch screen does not respond.
 //Quick enough.
 delay(100); //loop every 100ms/0.1s
 //If the screen is touched.
 Hydro_UI.CheckForScreenTouch();
 }
```

The controller is pretty straight forward. It gets the latest values from the HW sensors and if they have changed from the last update, then the UI will be updated with that information.

The sensors are checked every five seconds as this is often enough to get an impression of live continuous updates.  The timer, which is shown on the screen only has to the minute accuracy and so poses no problem.

The delay is added as this will put less stress on the HW and will reduce the power output of the device somewhat.  If the value is any bigger, then there is a noticeable affect on the UI in that it does not respond to touches quick enough.

The value updates for the EC and pH level do not come direct from the HW libraries but via the serial port callback `void serialEvent2()` for the pH level and `void serialEvent3()` for the EC level.

# 5 DISTRIBUTION OF THE SW

The SW license used for the whole project is the GNU  v3 license.  This was chosen as it is full copyleft.  This means that all derivative works coming from this, must themselves be open sourced and free to be modified.

The lesser GNU license was disregarded as it would mean that the propriety SW could link to one of my libraries, so being a disadvantage to free SW in general.

This SW license is well used and endorsed by numerous organisations involved in the open source and free SW movement.  Therefore, it seems like a safe option from a legal point of view.

The project will be hosted on GitHub at this location [26]

This includes the SW as well as the HW layout files.  In this way the whole project is open sourced and not just the SW.

# 6 UI USABILITY STUDY

The UI will be designed to make the product enjoyable to use.  The graphics used will be limited however due to the restrictions of the HW. This is something that will form the basis of future work and is out of scope for this thesis. For example, making use of the SD card to load more fancy graphics.

In order to ensure that the UI would be easy to use, a UI test plan was made and people, who knew nothing of the design, were used to try it out.

This chapter details the evaluation of the UI which has been created for the hydroponics controller and needs usability testing.  When tested here, the UI was in early phase development and the results were used to see how easily the user can accomplish two tasks.  Based on the results, the UI can be changed in order to increase the usability of the system. The two tasks that we will focus on here are,

- Calibrating the EC ( Electrical Conductivity ) HW ready for use
- Setting which pH levels are required

These two tasks have been chosen as they very closely match other operations that must be carried out on the device.  For example, setting up the pH HW and setting the air, water and EC levels are all pretty much similar.  So by executing these two tests a large part of the UI is tested.

The final goal of this study is to make the device more pleasurable to use.  The feedback gathered from the usability tests will be put back into the development of the system. It is hoped that this cycle will continue until the development is complete.  Only one cycle is detailed here but the methodology will remain the same.

### 6.1 Usage Scenarios

In this usability study we will detail two use cases.  Many of the operations in the system resemble each other and for this reason our use cases have been chosen to cover as much of the UI usability as possible.

Typically, the user will use the controller when they have renewed the water in the tank or when they have changed the plants that they are growing.  Again our choices of use cases to examine will cover both of these scenarios and so hopefully covering the maximum functions of the system.

**The first test: Calibrate the EC level**

In this use case, the user has just changed the water for use when growing.  Typically, a user would change the water every few days in order to ensure that the quality remains good enough for growing plants. The first step is to navigate to the EC screen and find the calibration screen. We assume in each use case that the user will start from the home screen.

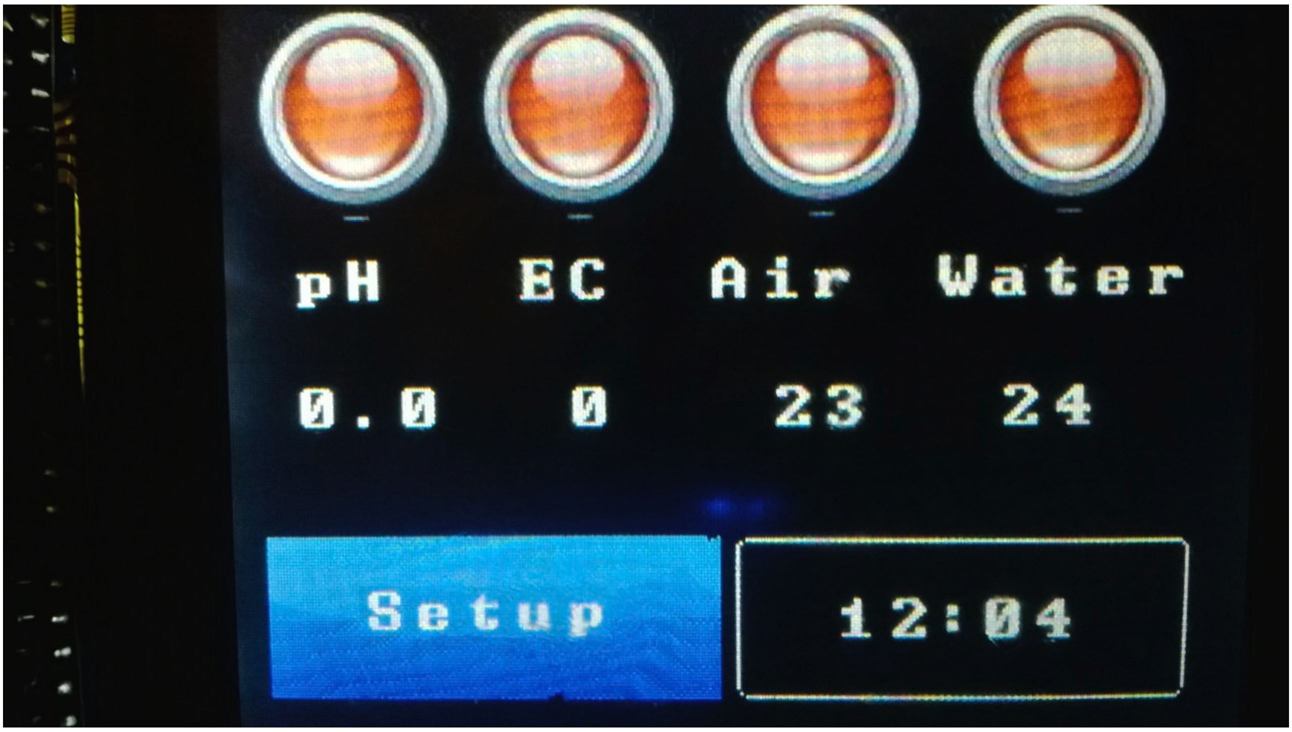


FIGURE 6. The Home Screen

This is a pretty reasonable thing to assume as the home screen is where most of the device will spend its time as this is the screen where the user gets details of the current levels and where warnings will be shown if a certain level becomes too high or low.

As can be seen in Figure 6:  There are four red circles, which tell the user that the given levels are out of range for what they have set and when in range, they are green. They also double up as buttons to navigate to the setup screens for that feature.  By clicking the setup button, the user can also navigate to the setup screens.  The setup menu takes the user to Figure 7, and from there, you can access the same features as the home screen plus a few more.

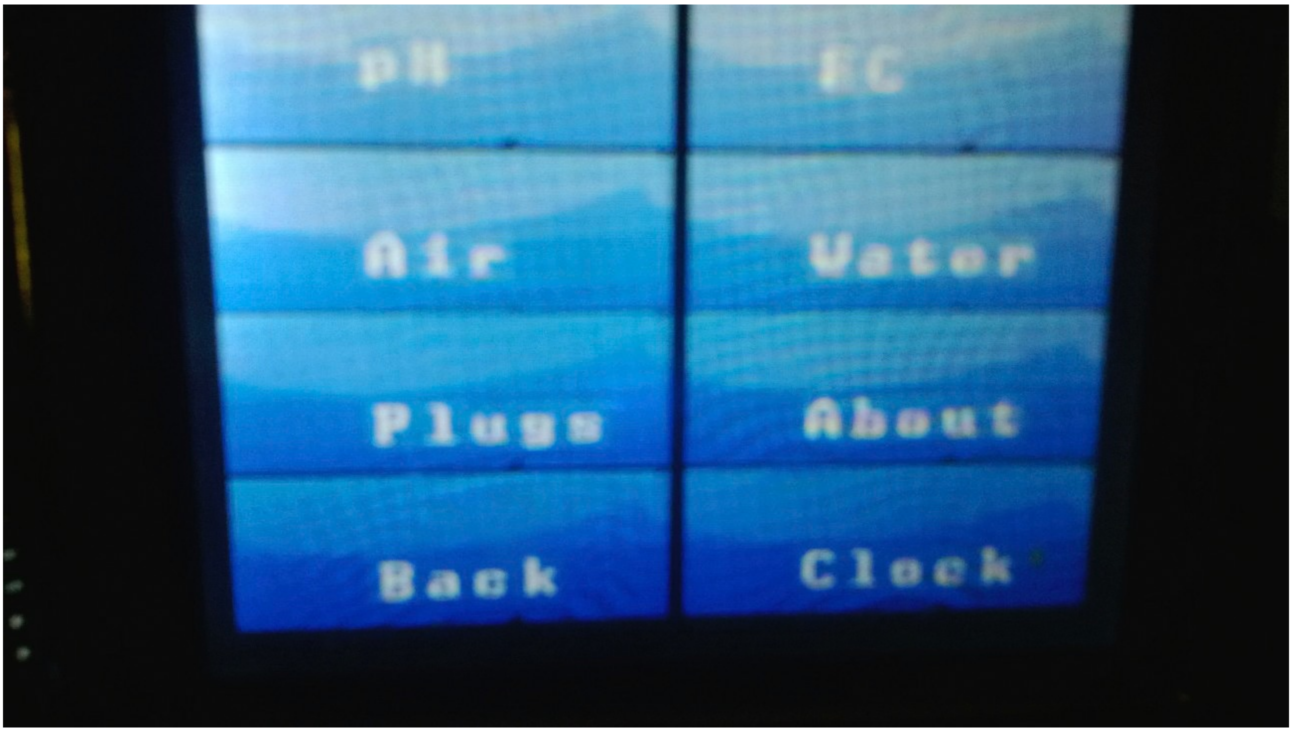


*FIGURE 7. The Settings Screen*

Starting from the home screen the user has to then go to the EC screen. As said, there are two ways to do this:

1. Just click on the EC circle from the home screen
2. Click on Settings and then EC

In this scenario we will see if the user finds it easy to understand that they can click on the red circle to get to the EC screen.

Once on the EC screen, the user then has to press the settings button and then the calibrate button.

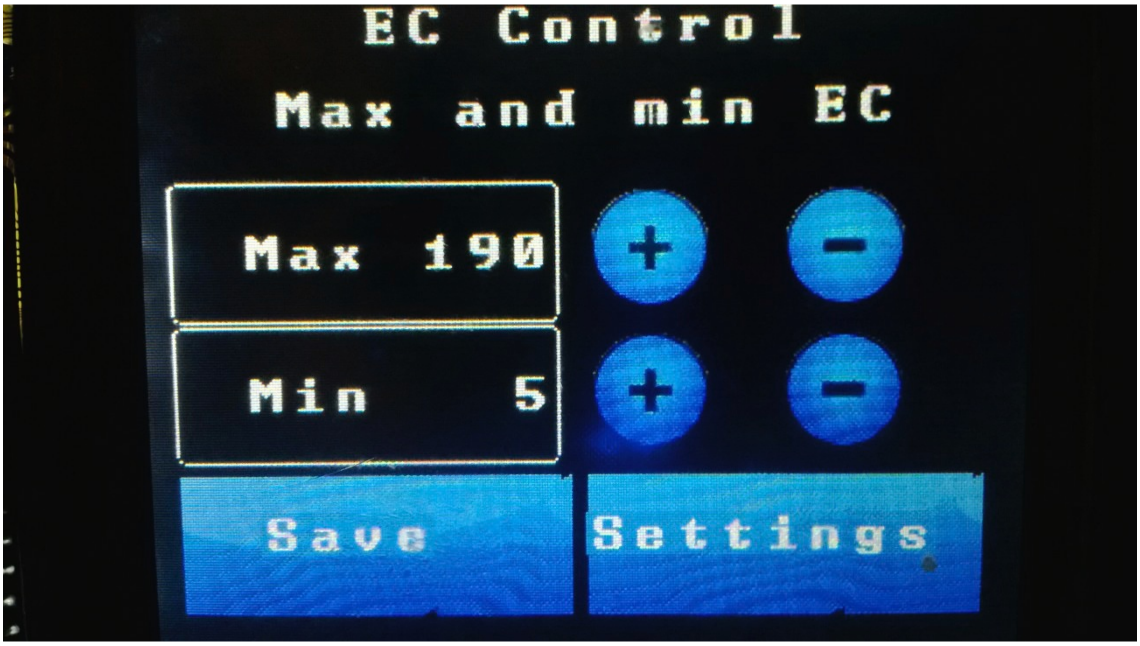


*FIGURE 8. The EC Screen — Press Settings*

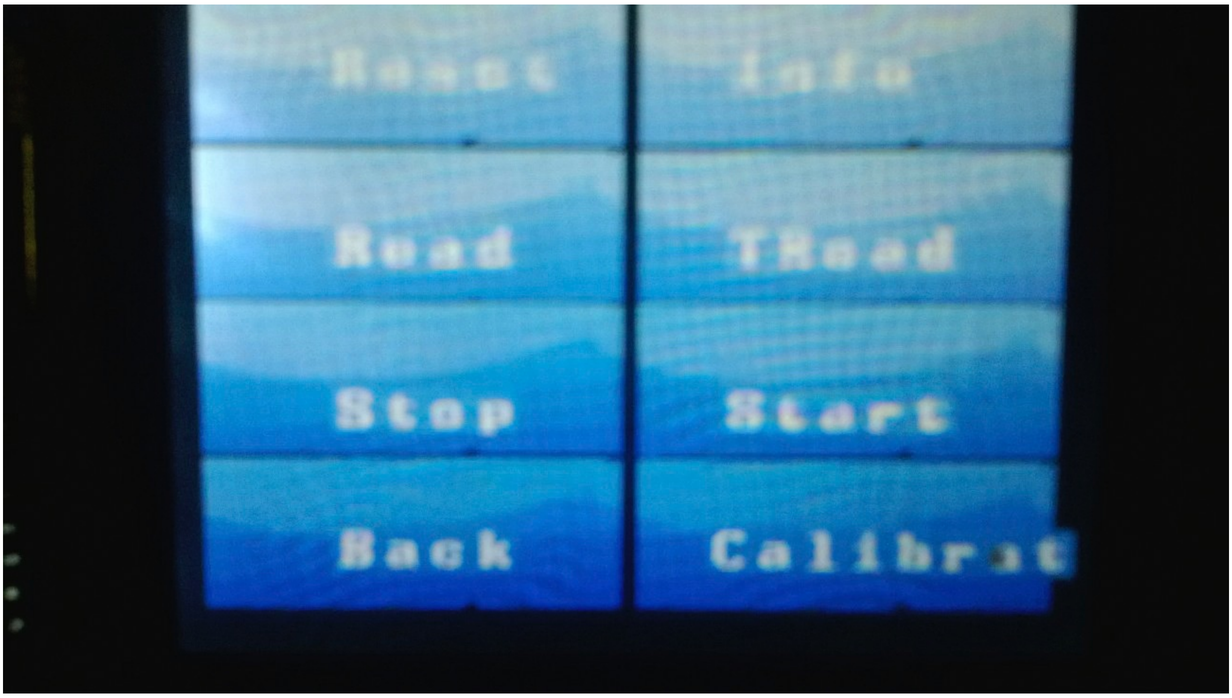


*FIGURE 9. The EC Settings screen - The calibrate button is bottom right*

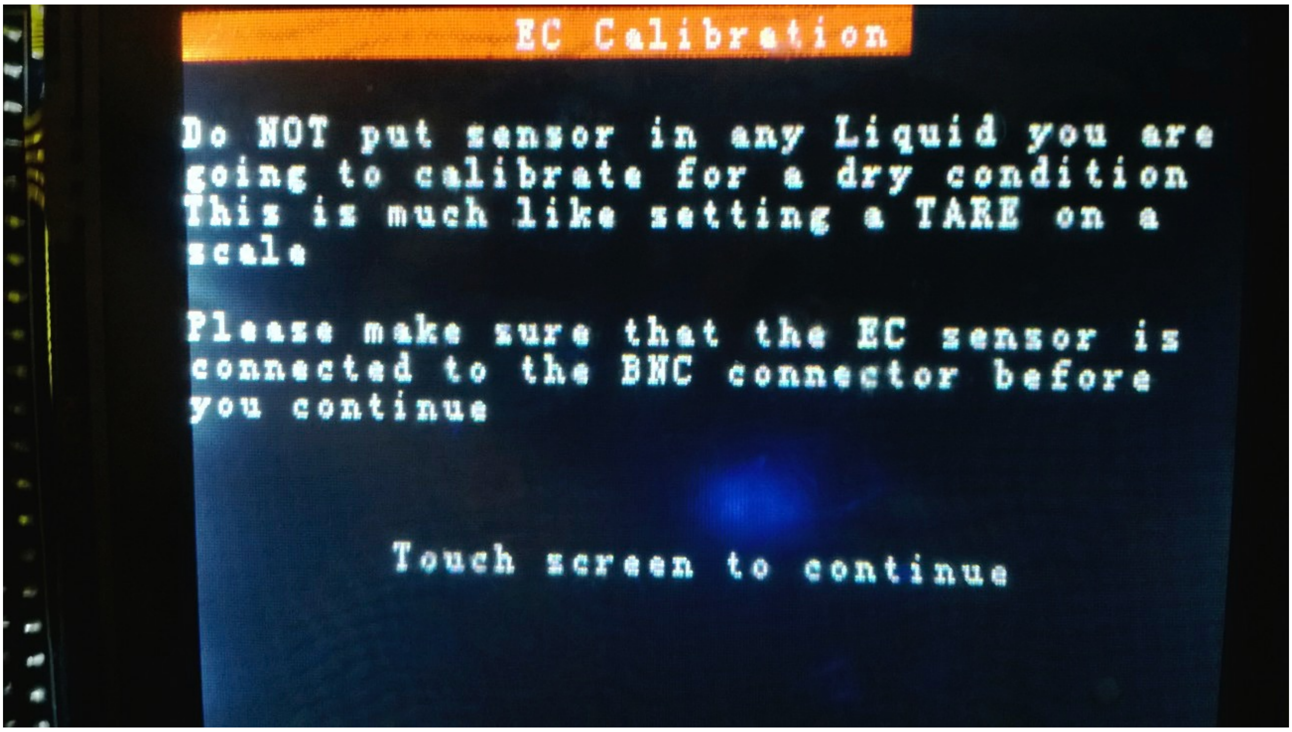


FIGURE 10. The start of the EC calibration

After this, there is a calibration screen with instructions on what the user needs to do in order to perform the calibration of the EC.  Each screen will wait until the user presses the screen, at which point the device will automatically save the data from the EC sensor and move on to the next phase.  The order of the phases and the time the user has to wait has been determined by the HW sensor and so this part of the test case is fixed and cannot be changed even if it affects usability.  Each screen will give feedback on how much time is left until it is complete. It displays a message " x seconds left unit completion". The note is updated every three seconds.  As it takes some time for the readings to stabilise, each phase of the calibration system will take some time, usually around five minutes.  At the end of the process the user is informed and the user may exit. There are four phases in total, phase one is shown in Figure 10.

In addition to using the device, the system will give instructions on how the HW probe is to be used to ensure that the calibration goes smoothly.  In the usability study, how easily the user can follow the instructions and put the HW probe into the right position was monitored.

This is the same process as when calibrating the pH level. The actual details given in the phases of the screens are different but the process and layout of the calibration from the UI point of view is the same.  So, here we can see how both of these processes will work.

**Test 2: Setting adequate pH level settings**

In this use case scenario the user has just changed which plants are growing and they need a different pH level than the one which were there before.  In order to change the pH level, the user needs to navigate to the pH screen.

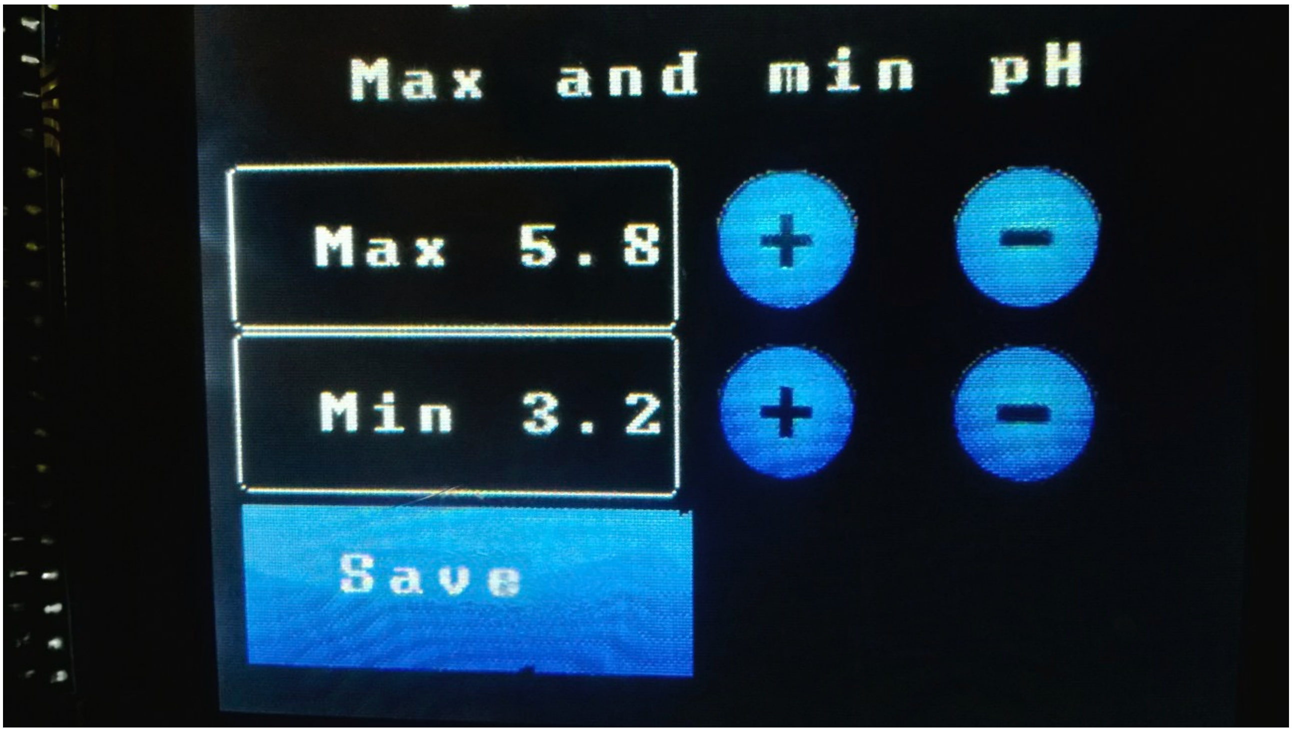


*FIGURE 11. The pH control screen*

So, starting from the home screen, the user has to then go to the pH screen. There are two ways to do this:

1. Just click on the pH circle from the home screen
2. Click on Settings and then pH

Once on the pH screen the user will be presented with two boxes, one that says min and the other max. These boxes display the maximum and minimum at which the pH level is allowed to go before the system will give an alert to the user informing them that they need to raise or lower the pH level. The user can change the levels by pressing the plus and minus icons to the right. The user then saves the changes by clicking on the Save button.

The same process applies when setting the air and water temperature as well as the EC level. So, this one test case in essence tests the usability of all of these options.

Again the main focus here will be to see how easily the user can understand the screens meanings and do the required tasks.

### 6.2 Description of Test Procedure

As I am the person who has created the UI, I was not going to be the person who tested the system.  Instead a third party, who has never used the system did that and I made observations based on their usage.  This way it was ensured that a real test of the system took place as I am too familiar with the system myself.

I told the person which task is required and I made notes on how the task was accomplished and noted down any problems that were raised.  The user was able to ask questions although I did not answer them but just deferred them until after the test was over.

I sat with the user during the test just to make notes on how the user was trying to figure out what to do and where they might get stuck or find something difficult to use.  In addition, the user chosen was not a computer guru and generally finds technical things daunting.   From this point of view too, they represented an average user.

### 6.3 Observations of the Execution

From the tests carried out I made the follow observations:

- The red/green circles that represent if the levels are in the predefined range made sense as short cut keys.
- The circle buttons for min and max were confusing. The user thought the rectangle squares, which represent the values were the buttons as all the other buttons were rectangles except for those which are circles here. It took some time for the meanings of those buttons to be fully understood.
- The save button was confusing as the user did not understand that it means go back.
- The calibration screens made sense and the feedback telling the user how much time is left worked well and the user just waited for the process to finish.

- The user did not fully understand what calibration was. Although a manual would clear this up, it may be useful just to add a few lines in the start process to ensure that the user actually wants to do this.
- After exiting the calibration screen, you go back to the EC screen and not to the home screen. Although this is part of the hierarchy, the user did not remember this.

From these observations I have selected the three most important ones based on the amount of confusion that it caused to the user.

1. I need to make sure that all the push buttons look the same.
2. As I am restricted in space, I need to use icons that would better represent, for example Save and Exit together.
3. I could add some text as to what calibration means to the start of the process as this way the user knows what is about to happen. Although a manual may solve this, many people will not read that.

**6.4 Recommendations**

The first problem can be cleared up by following the “Golden rules of interface Design” rule 1 Strive for consistency, as outlined in [8]. In this case I need to ensure that all the buttons have the same look and feel and do not just surprise the user with a change in this convention on a few screens.

The second problem is fixed by following the rule “Consistency and standards” as stated in [9]. I need to use words as they are defined and not assume that people will realise that saving means closing the dialogue.

The third problem too, can be solved by following the rule “Help and documentation” as outlined in [9]. Although documentation will be provided with the device, giving some help within the device itself will ease the burden on the user in using the device.

### 6.5 Usability Conclusion

A lot of lessons were learnt from doing this test. Being too familiar with a system can bring negative side effects from a UI point of view, when you are developing it as well. Trying to make the UI a bit different in terms of adding some different buttons seemed like a good idea as I thought it would make the UI more interesting but all it did was make it more confusing. Assuming too much was a mistake I also made, when guessing that people would think saving a screen would therefore mean exiting it too.

Some lessons learned were positive though, like giving feedback to the user informing how much time a process will take was well received.  It seems like those golden rules outlined in the references really are golden.  In the future I will make sure that my UI design adheres to those rules as they have been built up over many projects and years, they most certainly are worth keeping in mind.

I need to be creative with some of my icons. I cannot afford to have a lot of text, yet I need to give the user some indication on what the buttons do.  I will need to study more in these areas to make my UI more standard in order to communicate meanings better.

# 7 BUILD MANUAL

This section details how the HW can be built and the SW installed. It has been made relatively easy. The only difficult stage is that of soldering. If the soldering is not done correctly, then the device will not work at all or will be unstable. A soldering iron, solder and a screw driver are the only tools needed.

Before building the device you will need to have the circuit board made.

The best way to do this is to get the Fritzing file from [26] and then order the PCB directly from them. This has been tested and is guaranteed to work. Other alternatives are to generate some Gerber files and have the circuit board manufactured elsewhere or print off the version as a PDF and etch your own version of the printed circuit board.

The bill of materials, with recommended places to buy the HW is shown in table 11 below. The Arduino Mega can also be replaced with a clone. This is legal and can be much cheaper. Some clones are on sale for only 10 Euros.

If you choose not to buy from the location example, you must still buy the actual HW devices, for example a DHT11 temperature circuit. Other similar circuits like DHT12 may or may not work. Likewise for the pH and EC circuits, other versions may work just fine from Atlas Scientific but be sure to check that first.

*TABLE 11. Bill Of Materials*

| Quantity | Component | Price in Euros | Location (Example) |
|---|---|---|---|
| 1 | pH Stamp | 21 | https://www.atlas-scientific.com/product_pages/embedded/ph.html |
| 1 | EC Stamp | 32 | https://www.atlas-scientific.com/product_pages/embedded/ec.html |
| 1 | TFT Screen + Shield | 17 | http://www.sainsmart.com/sainsmart-3-2-tft-lcd-display-sainsmart-tft-lcd-adjustable-shield-for-arduino-2560-r3-1280-a082-plug.html |
| 1 | DHT11 | 1.86 | http://www.fasttech.com/products/0/10000007/1144101-dht11-arduino-compatible-digital-temperature-humid |
| 1 | RTC DS1307 | 13 | http://www.adafruit.com/products/255#Distributors |
| 3 | Wireless Plugs | 27,90 | http://www.verkkokauppa.com/fi/product/9502/cctjg/Nexa-PB3-kauko-ohjattava-virtakytkin-3-x-2300-W-230-V |
| 1 | Transmitter/ Receiver | 1 | http://www.aliexpress.com/item/RF-wireless-receiver-module-transmitter-module-board-Ordinary-super-regeneration-315-433MHZ-DC5V-ASK-OOK-10pair/705027555.html |
| 1 | Arduino Mega | 48 | http://www.lawicel-shop.se/prod/Arduino-Mega2560-R3_854098/Arduino_63894/ENG/EUR |
| 1 | Enclosure | 5 e | |
| 1 | Water Temp | 3 e | |
| 2 | BNC Adapters | 12 | https://www.atlas-scientific.com/product_pages/components/bnc.html |
| 20 | Dupont wires | 5 e | http://www.fasttech.com/products/0/10002017/1144700-female-to-female-40-pin-dupont-breadboard-jumper-w |
| 1 | PCB Board | 25e | Fritzing |
| 1 | Arduino Stackable Header set | 2e | |
| | TOTAL | 266,76 | |

Once the circuit has been made up, the software can be downloaded as described in the chapter 9.If you do not have the Arduino IDE installed on your computer then download it from the Arduino website.Next you will need to make sure that the following external libraries are in the Arduino Libraries folder [7], [17],[20],[21],[22],[23],[24],[25]

After this open the file HydroControl_UI.ino and then click verify.  If there are any errors, most likely due to a missing package then solve them.  Once it compiles, attach the Arduino to the computer and click upload.

After this you should see the main screen.  The first steps to do are to set the time and calibrate the pH and EC circuits.

# 8 PROBLEMS FACED

The use of the DS3017 RTC caused too many problems with timing issues. The timings were irregular and it was difficult to use SW to counter balance them. As stated earlier, the DS3234 was used instead.

A Phidgets pH board was used first as this was a few Euros cheaper. However the quality was much lower as the calculations for the exact pH level had to be done in the SW. The Atlas Scientific circuits do all the calculations on the HW and they come with a lot more features. The calibration is also much easier with the Atlas Scientific, using the Phidgets board requires the SW to do all of the calculation. This would have increased the code complexity and size too much, therefore it was considered better from a quality point of view to just buy the better HW.

# 9 FUTURE DEVELOPMENT POSSIBILITIES

There are numerous enhancements that could be made to the HW and SW. It is possible to:

1. Add internet support and update the readings to a server.
2. Create a computer application that can be used to read the readings from the HW and display them nicely to the user.
3. Give BT support so that the controller can interact with phones and computers.
4. Improve the UI graphics.
5. Provide multi-language support.

These are some of the things that can be developed in later versions of the SW and HW.

# 10 CONCLUSIONS

This thesis considered what hydroponics is and what are the key ingredients to successful hydroponics growing. The findings from this study were used as inputs to the SW and HW design of the controller.

The HW was then sourced so that it could meet these requirements — for example, by answering questions such as whether they tolerate the right temperatures? Do they have enough accuracy? How do you communicate with them? and What is their price?

The controller was chosen as the Arduino Mega 2560 due to its support, open nature, large array of pins and interrupts as well as its large space for SW. A circuit was made up after prototyping on breadboards as this will help others to build a more robust system.  The plans for this have been shared with the code and others can easily make up their own circuit from this.

The SW was then developed based on the HW features, for example, the pH and EC circuits use the serial ports, and the RTC the SPI interface, etc.  First, any existing open sourced SW that could drive these devices were sought.  The benefit being increased quality and development time.  The UI and the HW code was not mixed together to ensure that at a later stage the HW can be updated with little work to the UI.

A UI usability report was conducted to ensure that the device is easy and intuitive to use.  The feedback from this report was fed back into the development of the SW. Further development of the project could include developing the interface even more.

The last stage was that of testing, the main testing was done by emulating real world scenarios and seeing if the device would work as expected.  Ideally unit tests would be developed to ensure that all of the code is covered as it is being written.  If the project was done from the start again, I would start the project as a test driven development project.

## REFERENCES


[1] Dr. Hector Munoz. (2010). Home Based Vegetable Production System. Date of retrieval 28.01.2014

http://www.iica.int/Eng/regiones/caribe/guyana/IICA%20Office%20Documents/Hydroponics%20Manual/Hydroponics%20Manual.pdf.

[2] Hydroponics. Date of retrieval 22.02.2014

http://en.wikipedia.org/wiki/Hydroponics

[3] EC vs TDS. Date of retrieval 19.03.2014

http://www.hydroponics.net/learn/debate_over_ec_and_tds.asp

[4] Fernandez, D. 2009. FAQ - Electrical Conductivity (EC) in Hydroponics. Date of retrieval 19.03.2014

http://scienceinhydroponics.com/2009/02/faq-electrical-conductivity-ec-in-hydroponics.html

[6] Short Range Devices. Date of retrieval 18.06.2014

https://en.wikipedia.org/wiki/Short_Range_Devices

[7] Karlsen, H 2014. UTFT Library. Date of retrieval 09.04.2014

http://www.henningkarlsen.com/electronics/library.php?id=51

[8] Shneiderman, B. The Eight Golden rules of Interface Design. Date of retrieval 06.08.2014

http://www.cs.umd.edu/~ben/goldenrules.html

[9] Nielsen, J. 1995. 10 Usability Heuristics for User Interface Design. Date of retrieval 06.08.2014

http://www.nngroup.com/articles/ten-usability-heuristics/

[10] our.Windowfarms. Date of retrieval 26.02.2014

http://our.windowfarms.org

[11] Fernandez, D. 2013. HydroBuddy. Date of retrieval 22.02.2014

http://scienceinhydroponics.com

[12] 2013. HydroCal. Date of retrieval 22.02.2014

http://hydrocal.sourceforge.net

[13] DHT11 Humidity & Temperature Sensor. Date of retrieval 01.02.2014

http://www.micro4you.com/files/sensor/DHT11.pdf

[14] DS18B20 Programmable Resolution 1-Wire Digital Thermometer. Date of retrieval 12.02.2014

http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[15] Extremely Accurate SPI Bus RTC with Integrated Crystal and SRAM. Date of retrieval 16.05.2014
http://datasheets.maximintegrated.com/en/ds/DS3234.pdf

[16] A Brief Introduction to the Serial Peripheral Interface (SPI). Date of retrieval 16.05.2014
http://www.arduino.cc/en/Reference/SPI

[17] Karlsen, H 2014. UTFT Library. Date of retrieval 08.03.2014
http://henningkarlsen.com/electronics/library.php?id=51

[18] Micro footprint electrical conductivity total dissolved solids salinity monitoring subsystem. Date of retrieval 12.04.2014
https://www.atlas-scientific.com/_files/_datasheets/_circuit/EC_Circuit_3.0.pdf

[19] Micro footprint pH monitoring subsystem. Date of retrieval 12.04.2014
https://www.atlas-scientific.com/_files/_datasheets/_circuit/pH_Circuit_5.0.pdf

[20] Karlsen, H 2014. UTouch Library. Date of retrieval 03.02.2014
http://henningkarlsen.com/electronics/library.php?id=55

[21] Karlsen, H 2014. UTFT_Buttons Library. Date of retrieval 03.02.2014
http://henningkarlsen.com/electronics/library.php?id=61

[22] 2013. 433MHz for Arduino. Date of retrieval 03.02.2014
https://bitbucket.org/fuzzillogic/433mhzforarduino/wiki/Home

[23] Karlsen, H 2014. DS3234 SW Library. Date of retrieval 19.06.2014
http://henningkarlsen.com/electronics/library.php?id=71

[24] Dallas Temperature Control Library. Date of retrieval 14.05.2014
http://www.milesburton.com/?title=Dallas_Temperature_Control_Library

[25] Robtillaart. 2013. A DHT11 Class for Arduino. Date of retrieval 14.05.2014
http://playground.arduino.cc/Main/DHT11Lib

[26] Griffiths, M. 2014. HydroController. Date of retrieval 22.09.2014
https://github.com/markcgriffiths/hydrocontroller

[27]Norman, N.A.(2002) The Design of Everyday Things. Basic Books