Thesis

Real-time crop/plant monitoring and maintenance system

Author: Joseph Laithwaite
Student ID: 201109273

Project Supervisor: Dr. Kirsty McKay
Project Assessor: Dr. Mohammad Hasan

1 - Abstract

The human population on earth is expected to rise to 9.772 billion by 2050 which will demand an increase in food production of over 30%. The system built in this project may play a part in tackling this issue.

Through a structured process of development, a functioning smart hydroponic greenhouse has been built with the ability to observe and corrector environmental conditions including light, temperature, humidity, water flow and pH. The system is interacted with via a web based user interface allowing the remote deployment of these system atop flat buildings. This has the potential to be a hugely impactful area of economical growth in the future as land, water and fossil fuels all become more scare [ref].

The advantage of a closed hydroponic system means water use can be reduced by up to 90%, by stacking or installing on roofs land use is reduced and with little use for fertiliser and transport when produce is farmed in cities, fossil fuel dependency can reduce too.

2 - Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.
I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.
I confirm that I have not copied material from another source nor committed plagiarism nor fabricated, falsified or embellished data when completing the attached piece of work.
I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

Signature Joseph Laithwaite

Date 13/04/18

3 - Contents

4 - Introduction

The report to follow is the second and final report for the “Real-time crop/plant monitoring and maintenance system” BEng project following on from the first preliminary report [1], which gave the initial plan for the project and a brief introduction to it’s relevance in the agriculture field. This report is more of an explanation of the work carried out and an analysis of the results found.

The system built in this project is a prototype of a smart hydroponic greenhouse equipped with an array of sensors, from temperature to pH, used to monitor and store environmental data inside and outside of the greenhouse. Hydroponics is simply the cultivation of plants with no soil [2]. The greenhouse also utilises components to change the environmental conditions, known as correctors in this project, these are things such as LED (light emitting diode) lights, fans and valves to release pH equaliser. By utilising correctors to change conditions when the sensors realise it’s needed, the system can maintain a more consistent growing environment as close as possible to optimum levels set by the farmer. The farmer interacts with the system through an online interface giving farmers the ability to configure the system by selecting what sensors and correctors are connected to which Arduino pins. Once the system is configured a famer can then set desired conditions linking a sensor such as luminosity to a threshold level, say 100 lux, with a corrector, such as LEDs. This allows a farmer to customise desired conditions for whatever plant is being grown as well as, monitor live sensor readings and graphically analyse historic data. By pairing this with the ability to input crop yields comparisons can be made between success of a harvest and conditions used, hopefully allowing link to be spotted to improve desired conditions.

The industrial applications of the system developed are also explored finding a range of applications from an educational tool to commercial food production and tackling issues as diverse as land and water use and childhood obesity.

The rest of the report goes into detail of the theory underpinning this project, why certain pieces of hardware were chosen, how hardware had to be calibrated and a detailed explanation of software design. The software comprises of four key sections, an SQL (structured query language) database which acts as memory for the whole system, PHP (PHP : hypertext pre-processor) scripts used for a web interface, python scripts for the interface between Arduino and Raspberry Pi and finally C++ program running on the Arduino to carry out sensor and corrector operations.

Finally, results are displayed comprising of graphs displayed on the web interface, screenshots of the user interface and images of the built system. A comprehensive discussion follows explaining the significance of each result. [3]

5 - Industrial relevance, real-world applicability and scientific/societal impact

The crop monitoring and maintenance system developed in this project has wide reaching real-world applications ranging from commercial farming to an education tool. These two very different applications solve two very different problems but by making use of the systems self configuration, both are equally plausible.

The worldwide agriculture industry currently services 7.550 billion people and by 2050 the world population is expected to rise to 9.772 Billion [4], as food is a vital resource for human survival the industry must find a way to provide over 30% more food for these extra people. This is no small task as agriculture already exploits 37.3% [5]of the worlds land area (though 77% of this is used for livestock [6]) which may sound low but when forest land accounts for an all time low of 30.8% [7] and 29% is either baron or covered in glacier [6] there's not much extra space to grow the required food. The production of food must therefore become a lot more land efficient.

Another issue the agriculture industry must overcome is it's use of water; agriculture is the largest consumer using 70% of global fresh water used each year [8] .As a result of this exploitation and climate change, 1/3 of the worlds largest ground water supplies are already running out [8] and [9]. of the largest river basins on earth are expected to experience, at the very least, economic water scarcity by 2025 [9]. This coupled with the industry being the largest polluter of fresh water through nutrients and organic loads [10] seeping into water supplies, means much more care has to be taken over how fresh water resources are used in the future.

Finally, the current size of the agriculture industry worldwide is $(US)3.18 Trillion [11], such a huge industry could therefore make vast savings, thus profits, by improving efficiencies in the whole process.

As a commercial system the project aims to reduce or remove many of these inefficiencies and negative externalities. By closely monitoring and recording precise environmental conditions over a crops growth and correcting these to their optimum, resources are therefore only consumed when absolutely required and allow for a larger crop to grow more quickly. The design of the system would allow it's use with traditional soil based farms as well as hydroponics with only small additions to the code base to add sensors and correctors facilitating more efficient irrigation. In the scope of this project however only the hydroponic system developed will be considered.

The system developed in this project is a prototype testing the functionality of various sensors and developing a software platform to allow multiple configurations. This means the system is very scalable so as the system grows the cost per crop decreases. A simple costing (using no bulk buying discounts) shows how the current 8 pant system costs £20.23 per plant, whereas for a 10m by 10m system growing 10,000 plants, the cost is just 28p per plant [Table shown in appendex B) System type cost analysis]. This demonstrates the economic opportunity for large scale food production with this system.

As well as costs reducing, land use issues can be mitigated by the system, as its small height and lightweight design (requiring no soil) means it could easily be placed on flat roof tops in urban areas. This would be a lot less obtrusive than conventional rooftop greenhouses which are designed for humans to operate inside, by allowing planting and harvesting to be carried out by opening the greenhouse top instead the height can be much smaller. Though a small air volume of a greenhouse can cause larger temperature swings between day and night [12], the thermal inertia of the building underneath helps maintain a much more consistent temperature than tradition greenhouses [13]. Another benefit of the system being placed inside cities the concentration of CO_2 in the air is much higher, as CO_2 is vital for a plant to photosynthesise having greater access to it means it doesn't require to be created by burning fuels. In terms of land use by placing these systems in urban areas no extra land is required.

This system would be especially attractive for large supermarkets as a growing system could be installed on their roofs meaning once the system is paid off the cost of production would be

negligible compared to the current method of paying for produce and paying for its transportation. The selling point of it's freshness could also be used as a great marketing tool for the shop. As well as supermarkets, factories may be interested as a similar installation could be used which would run carbon dioxide through the system and effectively offset a portion of the CO_2 produced on the site, this will become even more appealing as carbon taxes increase.

Finally, use of a Hydroponic system removes the strains on water as they have some of the highest water efficiencies using 90-95% less water than soil based farming [14] [15]so can be utilized even in the driest areas on earth. This will allow people in poor arid countries the opportunity to consistently produce food from the small sporadic rains. These water savings can be improved even further as some systems use treated sewage as nutrients and water to grow barley fodder as cow feed [16], this method requires no fresh water. Due to the nature of the closed system no nutrients (salts), herbicides, pesticides or sediments [10] are washed away to rivers, reducing pollution almost entirely plus saving money for the farmer with reduced chemical use. This closed loop system will also allow for specially tailored nutrients to to be given to the plants coupled with specific light wavelengths which together can produce a more nutritional crop in less time than conventional methods.

As an educational tool the small size and possible configuration with a low cost (by using only cheap sensor and correctors) would allow compact greenhouses to be practical for almost every class room. The reason this could be so valuable in classrooms is to try and fight the obesity crisis, just under one third of all youths aged between 2 and 18 are either overweight or obese [17] and some studies show that 77% of overweight children become overweight adults [18].

Studies have shown how introducing vegetable gardens to school can drastically change young people's relationship with food increasing youth's enjoyment of vegetables by 57% [19], increase youth's acceptance and desire of fruit or vegetable as a snack by 37% [19] and how a nutrition class involving gardening is more likely to keep result of increased food awareness after 6 months [20]. The fact the system developed in this project is self contained and can therefore be run continuously throughout the year would allow for a much longer interaction with fresh food as the youths can watch food grow day in day out. Other benefits of this system as apposed to a traditional garden is the possibility to group multiple subjects together, such as biology, nutrition and even computer science. With the ability for conditions to be set, competition between classes could be held to grow the biggest veg, youths may be competitive and thus more involved so enjoy learning how plants grow depending on their environment. This approach may aid the development of scientific methods as well as teach nutritional understanding to youths.

6 - Project Specification Report and Gantt Chart

Project Specification

Project Description and Methodology:

This project aims to develop a hydroponic crop monitoring and maintenance system large enough to hold 4 small salad plants. This will consist of a small greenhouse equipped with light, temperature & humidity sensors as well as equipment to correct these conditions such as grow lights, a fan and a heater. The Greenhouse will also house a growing bed, which will have water pumped to it and drain into a reservoir bellow at programmed intervals. Finally, the water reservoir will monitor the water level as well as contain a PH monitor and the ability to add either bicarbonate of soda or lemon juice to maintain the desired PH.

The whole system will be interacted with through a web interface served up by a Raspberry Pi, this will give the user the ability to set desired watering times, PH, light level, Temperature & Humidity. The system will then self correct itself using the above components and the ambient conditions.

The system will be designed to be as scalable and adaptable as possible. The system should be easily set up by a user with little technical knowledge and allow them to configure any combination of different sensors and correctors.

Milestones:

1) 3rd October - Electrical Hardware ordered
2) 11th October - Interim Report finished
3) 10th November - Hardware components working as expected on an individual basis, ie. lights and light sensor working together but not linked to the rest of the system.
4) 15th November - Installation of electrical hardware into growing environment (greenhouse)
5) 1st December - First draft of project presentation created
6) 12th February - Basic prototype with the web interface controlling the system

Tasks:

1.1) 2nd October - Decide upon hardware plan and circuitry.
1.2) 3rd October - Source hardware components required online.
2.1) 3rd October - Complete Specification
2.2) 6th October - Gantt Chart
2.3) 6th October - Draw up plans on computer software
2.4) 9th October - Literature Review
3.1) 10th October – Build LED Strip circuitry so strips can be controlled by the raspberry Pi.
3.2) 13th October – Light level sensors calibrated and can be used to turn on LEDs.
3.3) 20th October - Water pump and valve can be turned on & off/ Open/ closed using relays controlled by the raspberry Pi
3.4) 25th October - Temperature & Humidity sensor data measured to Arduino and passed to Raspberry Pi.
3.5) 27th October – Fan controlled by raspberry Pi
3.6) 30th October - PH Meter connected to Arduino and PH reading taken and passed to Raspberry Pi.
3.7) 5th November – Water level sensor created
3.8) 8th November – PH Equalizer addition components working

3.9) 10th November – PH Module working together.
4.1) 1st November - Design/ find green house and water reservoir suitable for the project & Order
4.2) 12th November - Install lights, temp, humidity & light sensors to greenhouse
4.3) 15th November - Install water level sensor, PH module, pump & valve to reservoir. 6.1) 22nd November - Plan the software, how the problems split up, what data needs storing and querying.
6.2) 6th December - Code individual methods and classes for each section
6.3) 5th February - Design web interface
6.4) 12th February - Raspberry Pi server running with the basic control functionality
8.1) 20th February – Incremental addition of functionality

Project deliverables:

1) Web interface with the ability to set and save different desired climate conditions
2) Live read outs of the current conditions in the greenhouse on the web interface. This will include all sensor information as well as which components are on ie. lights or fan.
3) Historic database of conditions and component usage. This will be queried to calculate information such as average lamp time used per crop.
4) PH of water reservoir messaged and automatically corrected by the addition of organic chemical ie. lemon juice to lower PH or baking soda to increase PH.
5) LED grow lights automatically turn on when it gets bellow desired light level.
6) If humidity or temperature higher in greenhouse than outside and above desired level fan turns on.

Gantt Chart

The figures 6.1 to 6.4 show the original Gantt chart for the project. Some sections took longer than expected, such as the Arduino code and by extension python interface, which had to be fully redesigned half way though the project due to using too much dynamic memory being used by the Arduino. This pushed back other sections such as building the physical greenhouse and user interface design meaning the system wasn't running and finished in time for a full plant crop to be grown and harvested with the system running. Instead, plants were grown inside the system but, often times the fully functionality of the system wasn't running, or sections were being tested.

ID	Task Name	Duration	Start	Finish
1	**Order Electrical Hardwar**	**7 days**	**Thu 28/09/17**	**Fri 06/10/17**
2	Decide upon hardware plan and circuitry	2 days	Fri 29/09/17	Mon 02/10/17
3	Source hardware components required online	1 day	Tue 03/10/17	Tue 03/10/17
4	**Finish Interim Report**	**7 days**	**Sat 30/09/17**	**Mon 09/10/17**
5	Complete Specificatior	2 days	Mon 02/10/17	Tue 03/10/17
6	Gantt Chart	4 days	Tue 03/10/17	Fri 06/10/17
7	Draw up plans on computer software	4 days	Tue 03/10/17	Fri 06/10/17
8	Literature Review	4 days	Wed 04/10/17	Mon 09/10/17
9	**Hardware components working as expected on an individual basis**	**26 days**	**Tue 10/10/17**	**Tue 14/11/17**
10	Build LED Strip circuitry so strips can be controlled by the raspberry Pi	3 days	Tue 10/10/17	Thu 12/10/17
11	Light level sensors calibrated and can be used to turn on LEDs	3 days	Fri 13/10/17	Tue 17/10/17
12	Water pump can be turned on & off using relays controlled by the raspberry Pi	3 days	Wed 18/10/17	Fri 20/10/17
13	Temperature & Humidity sensor data measured by Arduino and passed to	3 days	Mon 23/10/17	Wed 25/10/17
14	Fan controlled by raspberry Pi	2 days	Thu 26/10/17	Fri 27/10/17
15	PH Meter connected to Arduino and PH reading taken and passed to Raspberry	5 days	Mon 30/10/17	Fri 03/11/17
16	Water level sensor created	2 days	Mon 06/11/17	Tue 07/11/17

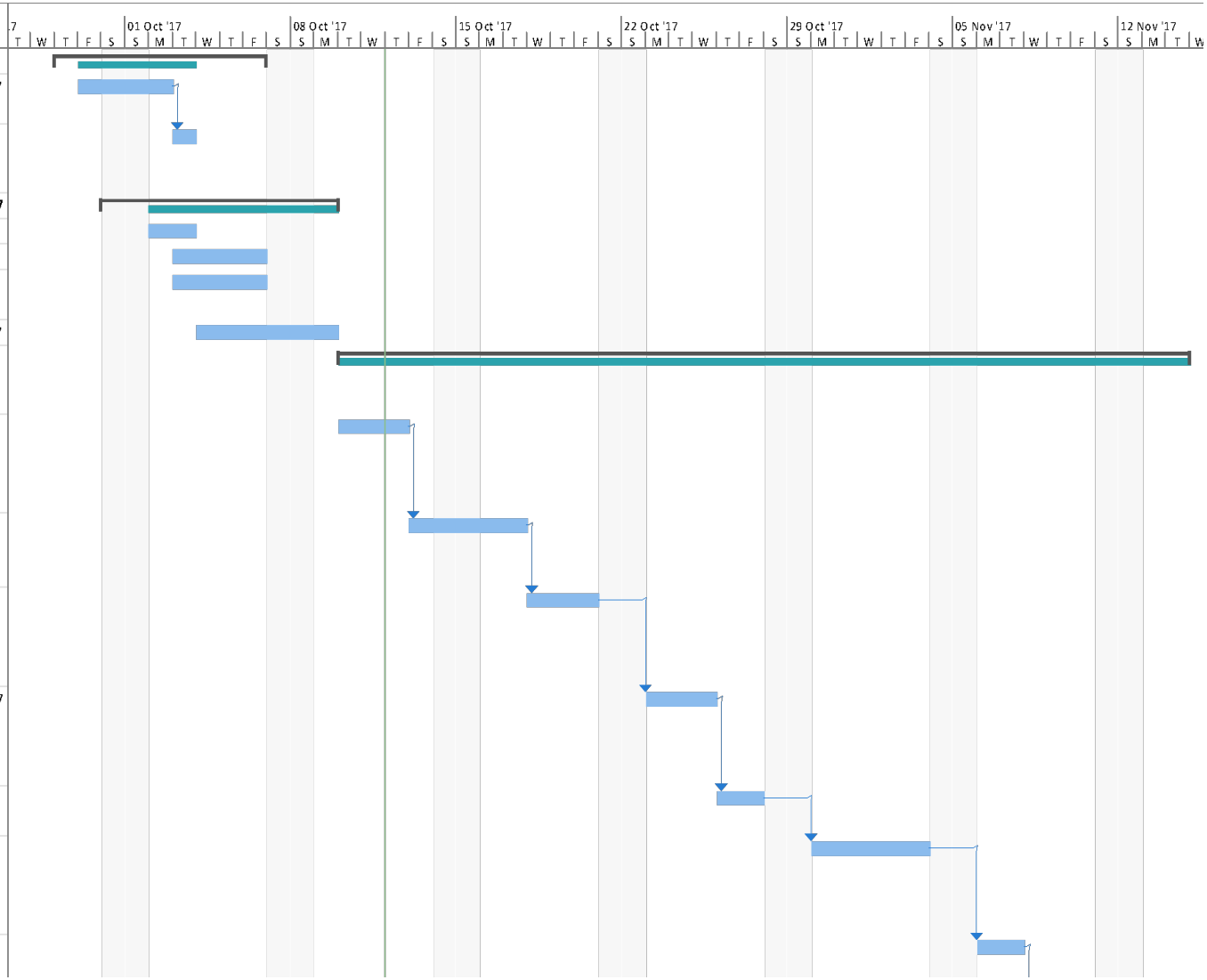

Figure 6.1 - Gantt chart part 1

ID	Task Name	Duration	Start	Finish
15	PII Meter connected to Arduino and PII reading taken and passed to Raspberry	5 days	Mon 30/10/17	Fri 03/11/17
16	Water level sensor created	2 days	Mon 06/11/17	Tue 07/11/17
17	PII Equalizer addition components working	3 days	Wed 08/11/17	Fri 10/11/17
18	PII Module working together	2 days	Mon 13/11/17	Tue 14/11/17
19	**Install elctronic hardware into**	**27 days**	**Thu 12/10/17**	**Fri 17/11/17**
20	Design/ find green house and water reservoir suitable for the project & Order	15 days	Thu 12/10/17	Wed 01/11/17
21	Install lights, temp, humidity & light sensors to greenhouse	1 day	Wed 15/11/17	Wed 15/11/17
22	Install water level sensor, PII module, pump & valve to	2 days	Thu 16/11/17	Fri 17/11/17

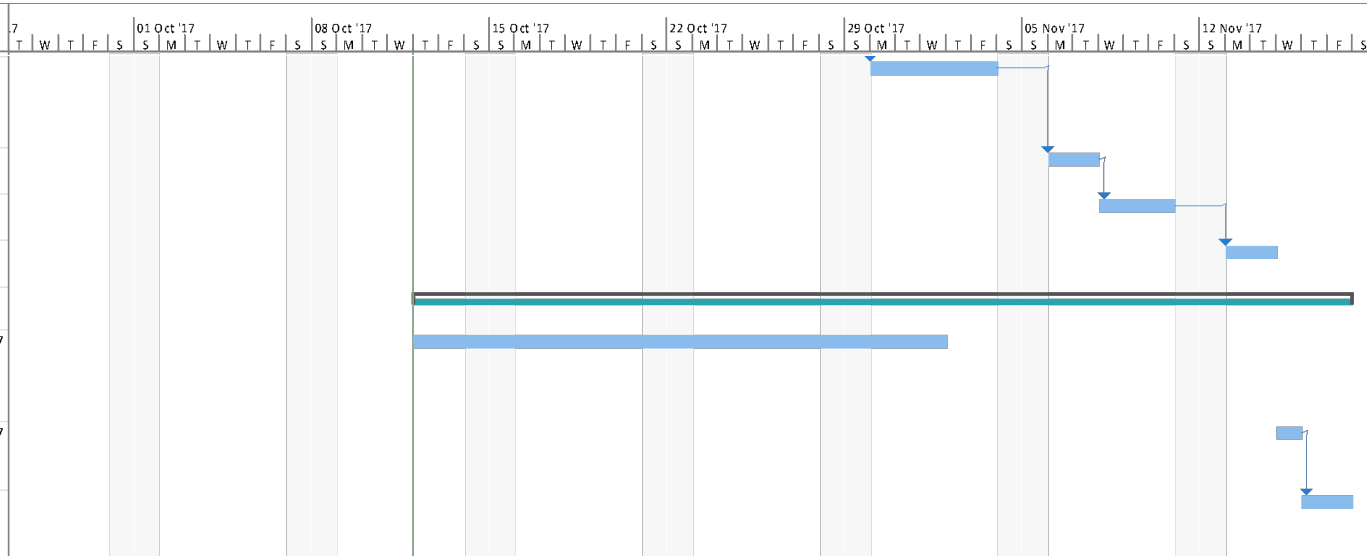

Figure 6.2 - Gantt chart part 2

ID	Task Name	Duration	Start	Finish
23	**Project presentation Ready**	**21 days**	**Sat 18/11/17**	**Thu 14/12/17**
24	First draft	10 days	Sat 18/11/17	Thu 30/11/17
25	Edited, improved and finalised	10 days?	Fri 01/12/17	Wed 13/12/17
26	**Prototype with the web interface controlling the system**	**66 days**	**Wed 15/11/17**	**Mon 12/02/18**
27	Plan the software, how the problems split up, what data needs storing and querying.	7 days	Wed 15/11/17	Wed 22/11/17
28	Code individual methods and classes for each section	10 days	Thu 23/11/17	Tue 05/12/17
29	Design web interface	43 days	Thu 07/12/17	Mon 05/02/18
30	Raspberry Pi server running with the basic control functionality	5 days	Tue 06/02/18	Mon 12/02/18
31	**Plants placed into system**	**10 days**	**Thu 01/02/18**	**Wed 14/02/18**
32	Germinate seeds	9 days	Thu 01/02/18	Tue 13/02/18
33	Move sprouted seeds to greenhouse. Add water & nutrients	1 day	Wed 14/02/18	Wed 14/02/18

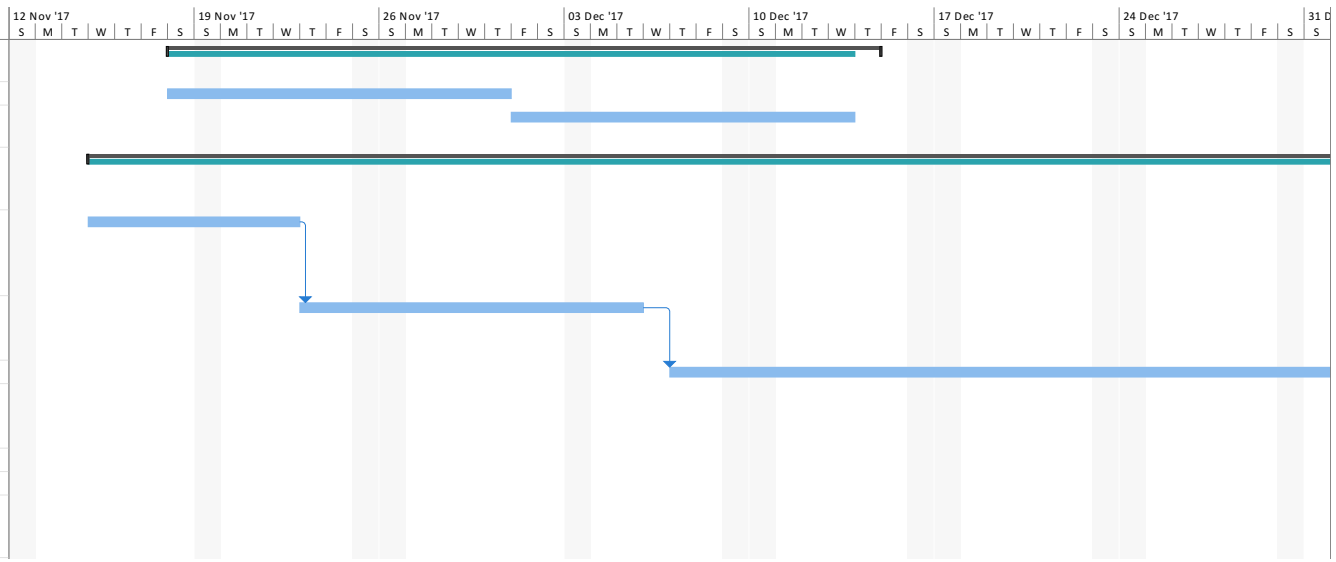

Figure 6.3 - Gantt chart part 3

ID	Task Name	Duration	Start	Finish
23	**Project presentation Ready**	**21 days**	**Sat 18/11/17**	**Thu 14/12/17**
24	First draft	10 days	Sat 18/11/17	Thu 30/11/17
25	Edited, improved and finalised	10 days?	Fri 01/12/17	Wed 13/12/17
26	**Prototype with the web interface controlling the system**	**66 days**	**Wed 15/11/17**	**Mon 12/02/18**
27	Plan the software, how the problems split up, what data needs storing and querying.	7 days	Wed 15/11/17	Wed 22/11/17
28	Code individual methods and classes for each section	10 days	Thu 23/11/17	Tue 05/12/17
29	Design web interface	43 days	Thu 07/12/17	Mon 05/02/18
30	Raspberry Pi server running with the basic control functionality	5 days	Tue 06/02/18	Mon 12/02/18
31	**Plants placed into system**	**10 days**	**Thu 01/02/18**	**Wed 14/02/18**
32	Germinate seeds	9 days	Thu 01/02/18	Tue 13/02/18
33	Move sprouted seeds to greenhouse. Add water & nutrients	1 day	Wed 14/02/18	Wed 14/02/18

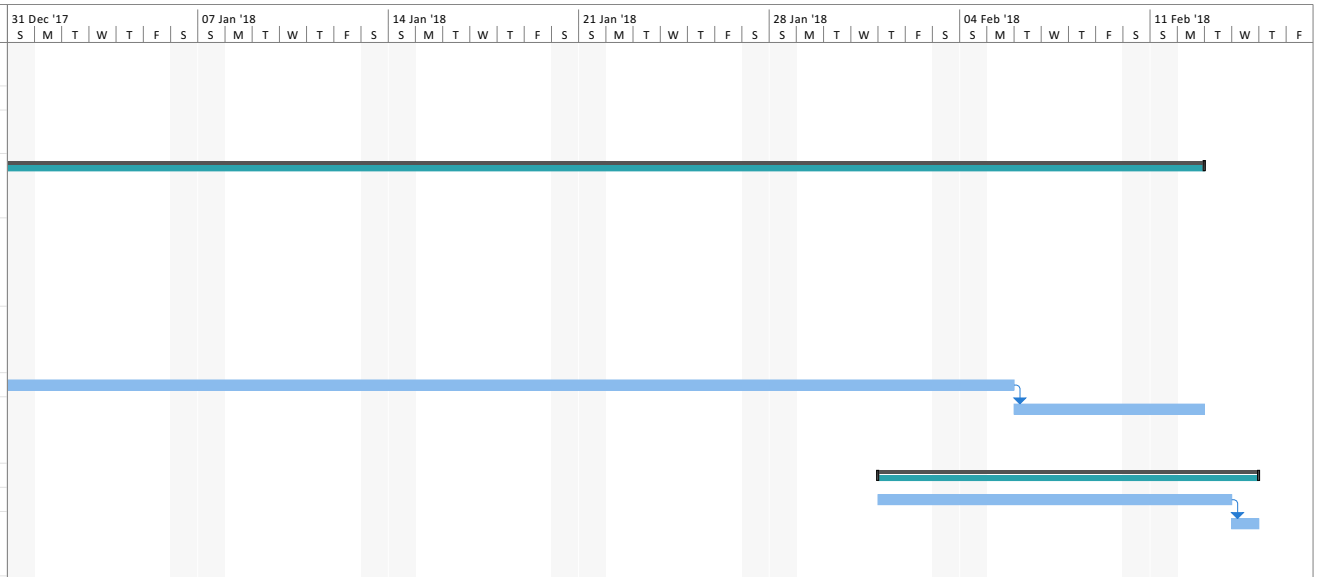

Figure 6.5 - Gantt chart part 4

ID	Task Name	Duration	Start	Finish
34	Testing complete with system fully operational	11 days	Thu 15/02/18	Thu 01/03/18
35	**Prject bench inspection prepared**	**6 days**	**Fri 02/03/18**	**Fri 09/03/18**
36	Poster Designed and printed	3 days	Fri 02/03/18	Tue 06/03/18
37	Demonstration video shot & edited	3 days	Wed 07/03/18	Fri 09/03/18
38	Bench Inspection Week	5 days	Mon 12/03/18	Fri 16/03/18
39	**Complete Report**	**17 days**	**Sat 17/03/18**	**Sun 08/04/18**
40	Complete first draft	7 days	Sat 17/03/18	Mon 26/03/18
41	Get Feedback on first draft	4 days	Tue 27/03/18	Fri 30/03/18
42	Write 2nd draft	3 days	Mon 02/04/18	Wed 04/04/18
43	Finish Final Report	3 days	Thu 05/04/18	Sun 08/04/18

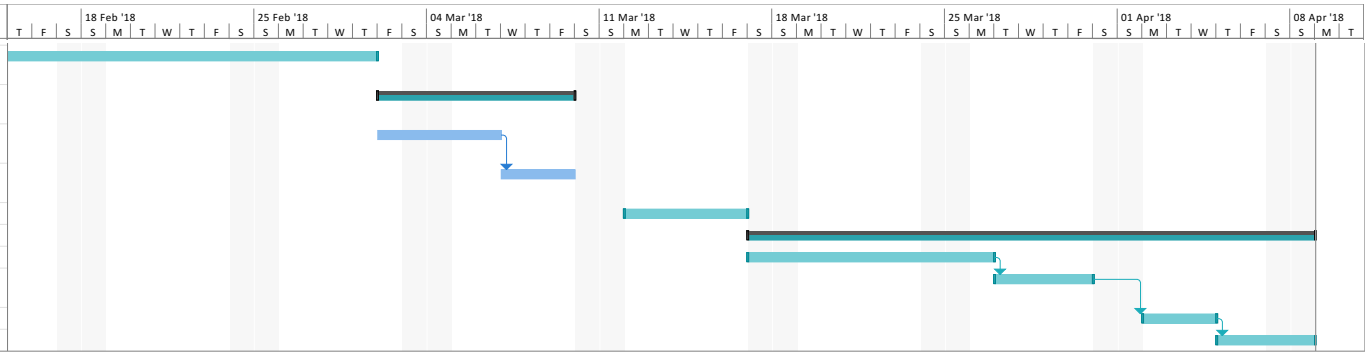

Figure 6.4 - Gantt chart part 5

7 - Theory

Importance of pH control

Firstly, to understand why it's important to plant growth and how it's measured, pH must be explained. pH is a measure of concentration of H^+ (hydrogen ions) in a solution compared to pure water (set at pH 7), an acidic solution contains a high concentration of hydrogen ions, whereas an alkali solution contains a low concentration of H^+, as it forms OH^- (hydroxide ions) which accept H^+ ions (to make water). The pH scale is a way to simply understand these concentrations by using the negative logarithm of H^+ concentration [21] ie. $pH = -\log[H^+]$. So for pure water with concentration of H^+ equal to concentration of OH^- at 10^{-7} Mol/Litter, the pH = -log[10^{-7}] = 7 and an acid, say lemon juice (citric acid) has a H^+ concentration of 0.006 Mol/Litter giving it a pH = -log[0.006] = 2.22 [22].

With respect to plants, as hydroponic plants grow they absorb inorganic ions and some organic compounds from the nutrient solution continuously run over its roots. The uptake of positive and negative ions causes the plant to release H^+ or OH^- ions respectively to keep a contestant ratio between positive and negative ions. This causes the pH of the solution to change which then affects the availability of nutrients. The make up of these solutions varies slightly but will always contain at least nitrogen, phosphorus, potassium, calcium, magnesium and sulphur [23] and at different pH levels, different ions are more common, some being easier than others for the plant to absorb. The diagram in figure 7.1 shows the availability of nutrients to a plant with varying pH levels. This demonstrates how, the extremes of the pH scale are completely unusable, half the nutrients are fairly consistently available at ±1.5 around neutral, and the remaining nutrients favours slightly acidic solution. Therefore, the optimum pH value for a nutrient solution is between 5.5 and 6.5 [24].

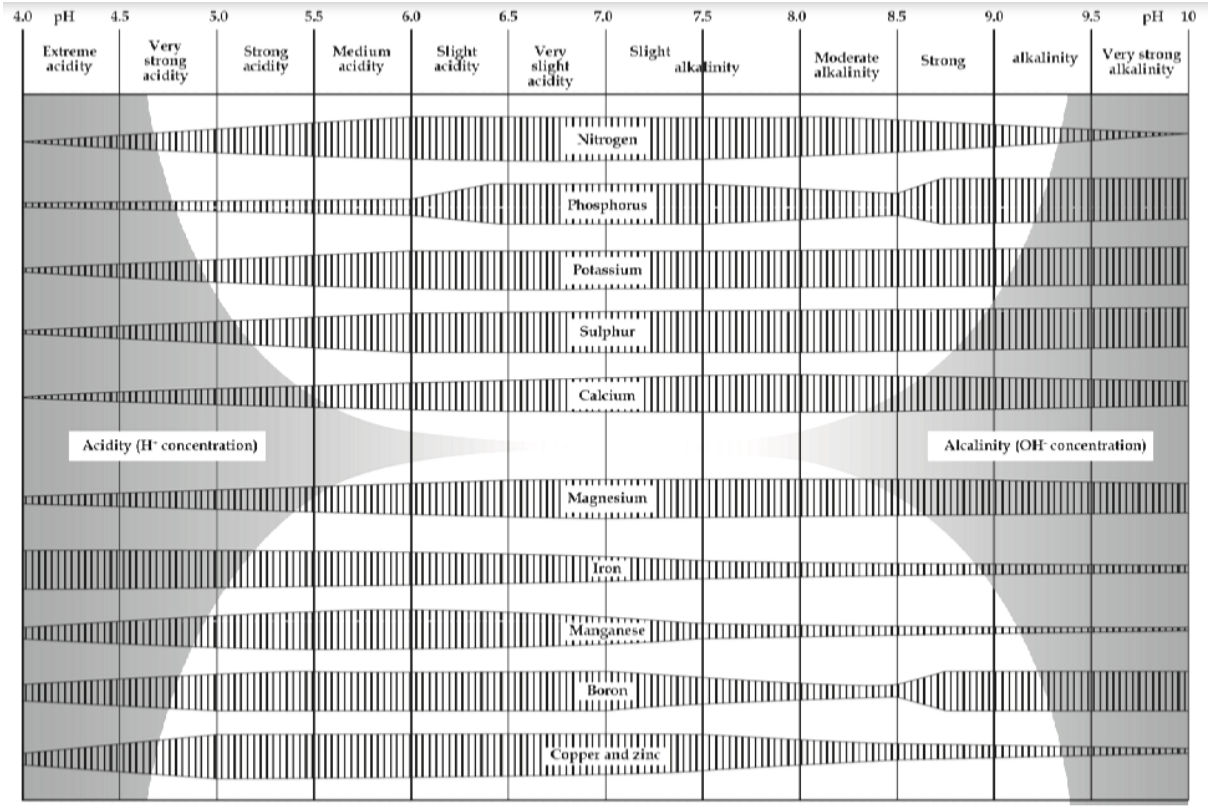

Figure 7.1 - Troug diagram of nutrient availability at varying pH levels. Each nutrient is represented by a band with each bands thickness proportional to its availaility. Graph taken from [23]

How the pH meter works

A pH meter is in effect a custom calibrated voltmeter with two specialised electrodes. Modern pH meters (such as the Haoshi SEN0169 [25] used in this system) integrates the two electrodes into a single device for ease and using an IC then returns an analogue voltage representing the potential difference between the two electrode. The returned voltage has a linear relationship with the solutions pH with pH calculated as

$$pH = 3.5 \times \text{voltage} + \text{offset}$$

with the offset determined during calibration. The two electrodes consist of either a silver or potassium metal wire suspended in a reference solution of KCl (potassium chloride), this reference solution has a neutral pH of 7, inside a glass tube. The first probe, known as the glass electrode has a small bulb at the end, this is bulb is made out of a special silica compound which allows ion exchange, this means the boundaries of the bulb collect different amounts of H^+ ions and therefore hold a slightly different charge. The charge inside the bulb is measured by the bulb electrode and the outer potential is measured by the reference electrode. The reference electrode does not allow for ion swapping so simply completes the circuit. The difference between these two electrodes can now be compared to find the potential difference and thus the pH.

Importance of humidity control

Humidity is a measure of how much water is held as water vapour in the air, there's two main measurements of which are absolute humidity, which is a measure of the mass of water in a given volume or mass of air and relative humidity, which is the same as absolute but compared to the saturated weight of water vapour in the air at that temperature. In this system we'll be using relative humidity as this allows controls to be carried out at the same thresholds independent of temperature.

Plants require CO_2 (Carbon Dioxide) and water to photosynthesise and uses pores on it's leafs to take in CO_2. At the same time water is drawn up the plant through the roots, some is used for photosynthesis then the rest is removed as water vapour again through it's pores, known as transpiration. In an environment with high relative humidity the air is already full of water so very little is transpired meaning the pores close up and can starve the plant of CO_2. On the other hand, if the relative humidity is very low, the plant puts too much energy into transpiration and causes other functions (such as growth) to slow. The ideal humidity for hydroponics is therefore in a range of 50% to 70%, this removes risk of mould from high humidity and requires enough transpiration that nutrients is transported from the roots to the leaves [26].

Importance of temperature control

Plants require temperature to facilitate respiration, as it's a reaction using enzymes it has an ideal range they can work in. At low temperatures between 0℃ & 10℃ the enzymes which carry out photosynthesis do so very inefficiently [27]. The ideal temperatures for most photosynthesizing enzymes is between 10℃- 20℃ as shown in figure 7.1.

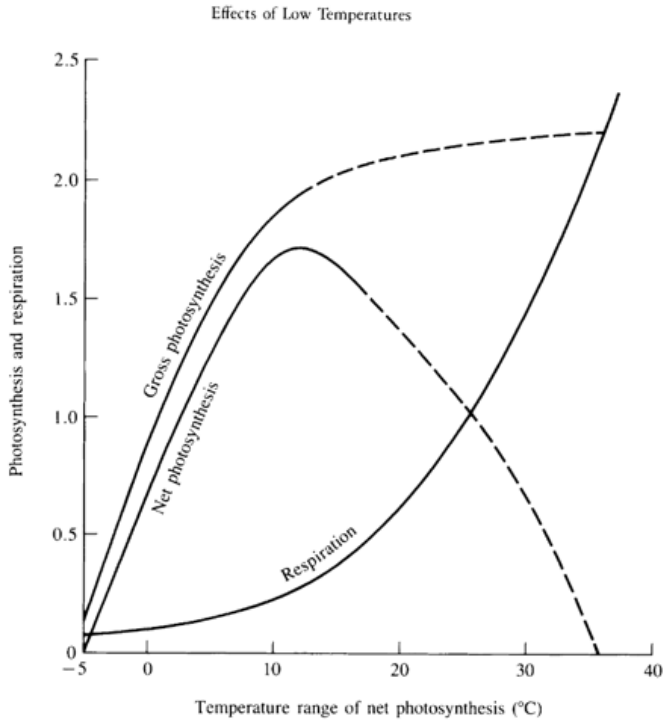

Figure 7.1 - Effects of low temperatures on photosynthesis, taken from [39]

How the DHT22 sensor works

The DHT22 sensor (Shown in figure 7.2) is a pre-calibrated combined temperature and humidity sensor.

The sensor utilises an NTC (negative temperature coefficient) thermistor for temperature [28], this works as a variable resistor reducing the resistance as the temperature increases. The IC (integrated circuit) can then calculate the resistance and ready this to be transmitted to the microcontroller.

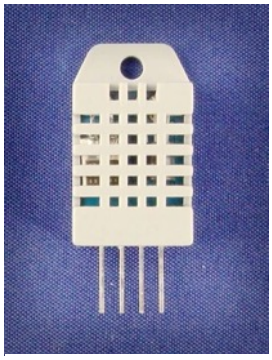

Figure 7.2 - DHT22 Sensor from datasheet [29]

For humidity a humidity sensing component [28] comprising of a moisture absorbing substrate sandwiched between two electrodes is used. When the substrate holds a large amount of moisture, it's conductivity is high and likewise, a low moisture content has a low conductivity. The IC can then prepare this resistance to be transmitted.

Even though the DHT22 has 4 pins, only 3 of them have functionality, the pins (left to right in fig 7.2) are VDD power supply, data signal, not connected & ground. The sensor communicates with the MCU (micro-controller) entirely over a single bus (the data signal pin) using a precisely timed pre-programmed protocol [29] as described below:

1. The MCU sends an initial start signal (data signal goes low from high) to the DHT22 and waits for a response.
2. On detection of the start signal the DHT22 changes from low-power-consumption-mode to running-mode then returns a response signal (again a low voltage).
3. DHT22 sends data to MCU. This data is 40 bits comprising of; 16 bits humidity data, 16 bits temperature data and 8 bits for a check sum.

The DHT22 does have its own Arduino library so this detail isn't required for it's use as the simpleDHT library can be included in any Arduino program. This allows an object to be created for the sensor with a given pin then to get the readings simply call the read2() function.

Importance of light control

Light is a fundamental part of how plants generate glucose through photosynthesis and therefore necessary for the plants growth. There's multiple ways of measuring light quantity and quality but in this system Lux is being used. This is the SI unit of luminosity where one Lux equals one lumen per square meter [30].

It's know the minimum luminosity required for most plants to photosynthesise is 1075Lux. It's also known that the wavelengths of light have a large effect on photosynthesis and thus plant growth. Research has demonstrated that under LED grow lights the best combination of light producing the largest yield of lettuce with the highest nutritional content is a combination of 90% red light (640 nm) and 10% of blue light (365 nm) [31].

How the TSL2561 works

The TSL2561 (shown in figure 7.3) is a digital light sensor utilising two photodiodes and an I^2C bus connection with up two 3 addresses [32].

Figure 7.3 - TSL2561 sensor [32].

The main issues with using a single full spectrum photodiode is that it responds strongly to IR (infrared) light [33], this means that with many light sources the photodiodes response will suggest bright conditions, when the visible spectrum may be dark. For this project this is vital given IR light does not aid in plant growth (it only brings heat). It's for this reason the TSL2561 utilises a second photodiode which is specifically for the IR spectrum, by integrating the output voltages and subtracting the IR response from the full spectrum response an accurate visible spectrum luminance can be calculated [33]. As shown in figure 7.4 the channel 0 photodiode (full spectrum) has a response from before 300nm to past 1100nm, the visible range for humans is from 380nm to 750nm meaning for over a third of channel 0s' response no visible light is present.

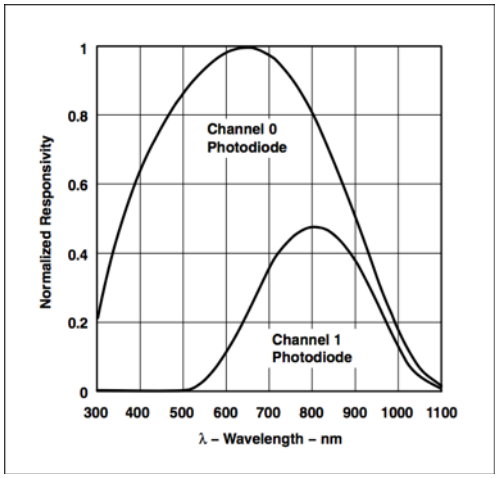

Figure 7.4 - TSL2561 Photodiode normalised responses [33]

The sensor also houses an ADC (analogue to digital converter) which converts the integrated response from an analogue voltage to a digital signal representing the luminosity in binary.

The binary string can now be transferred from the TSL2561 to the Arduino via I^2C connection. This standard two-wire bus protocol requires an SDA and

SCL connection to ensure no crosstalk occurs on the bus and allows the master (Arduino) to request readings when necessary. An Addr pin is also included on the component to allocate up to 3 unique addresses to the sensor, meaning three can be attached to the same I^2C bus at any one time. It manages this by setting the address as 0x39, 0x29 or 0x49 when Addr pin is no connected, connected to ground and connected to VDD respectively.

Choosing control components

The Arduino Nano

The Arduino is a cheap, easy to program microcontroller with a versatile range of functionality making it great for quick prototyping as well as use in open source hardware products. The wide range of built in circuitry greatly simplifies the development process allowing for use of a great range of pins:

- Analogue input pins allow input voltages from 0v to 5v to be measured with a precision 1024 (ie. 1 unit is 5/1023 = 4.888 mV). This means no additional ADCs (analogue to digital converters) are required meaning analogue sensors such as the pH meter can be used directly.
- PWM (pulse width modulation) output pins, these pins output a digital (square) wave alternating from on (5v) and off (0v) with a given frequency. This can be used to induce current in a transistor to control the voltage given to a component such as the LED strips.
- Digital output, outputs a voltage either high (5v) or low (0v). This is useful when switching devises such as relays or the power to a sensor.
- Finally, the I^2C (inter-interconnected circuit) bus connection which consists of the SDA (serial data) pin and SCL (clock stretching) pin allows 112 devices to communicate with the Arduino. This has been utilised for the lux meter, though could be utilised for many other sensors.

Developing code on the Arduino is also very straight forward. The standard Arduino IDE (integrated development environment) is a great programing environment for beginners and super fast prototyping, it was incredibly useful in the early stages of the project to allow for single functions to be tested, though it quickly became too basic to efficiently write code in one file. As a result, Eclipse IDE for C/C++ developers [34] was used with the sloeber plugin [35]. The sloeber plugin integrated normal object oriented C++ programming with the basic Arduino libraries and facilitated the compilation and build of the binaries to be uploaded to the Arduino. Unfortunately, my installation required a separate piece of software called Freematics Arduino Builder [36] to upload the binary to the Arduino.

As a whole this development system worked incredibly well to allow very efficient code to be written with full utilisation of inheritance, polymorphism and more complete debugging.

The Raspberry Pi

The Raspberry Pi 2 model B is an incredibly small, cheap Linux computer. It comes with most of the functionality you'd expect to get on a normal desktop including HDMI, Audio, Ethernet and 4 USB ports with 1GB RAM and a 900MHz quad-core ARM Cortex-A7 CPU, but it's the size of a credit card and only £30. With computing power it's more than capable of running a database, web server and act as a master to control the 'slave' Arduino(s) which is all that's required on the software side.

8 - Design

System design

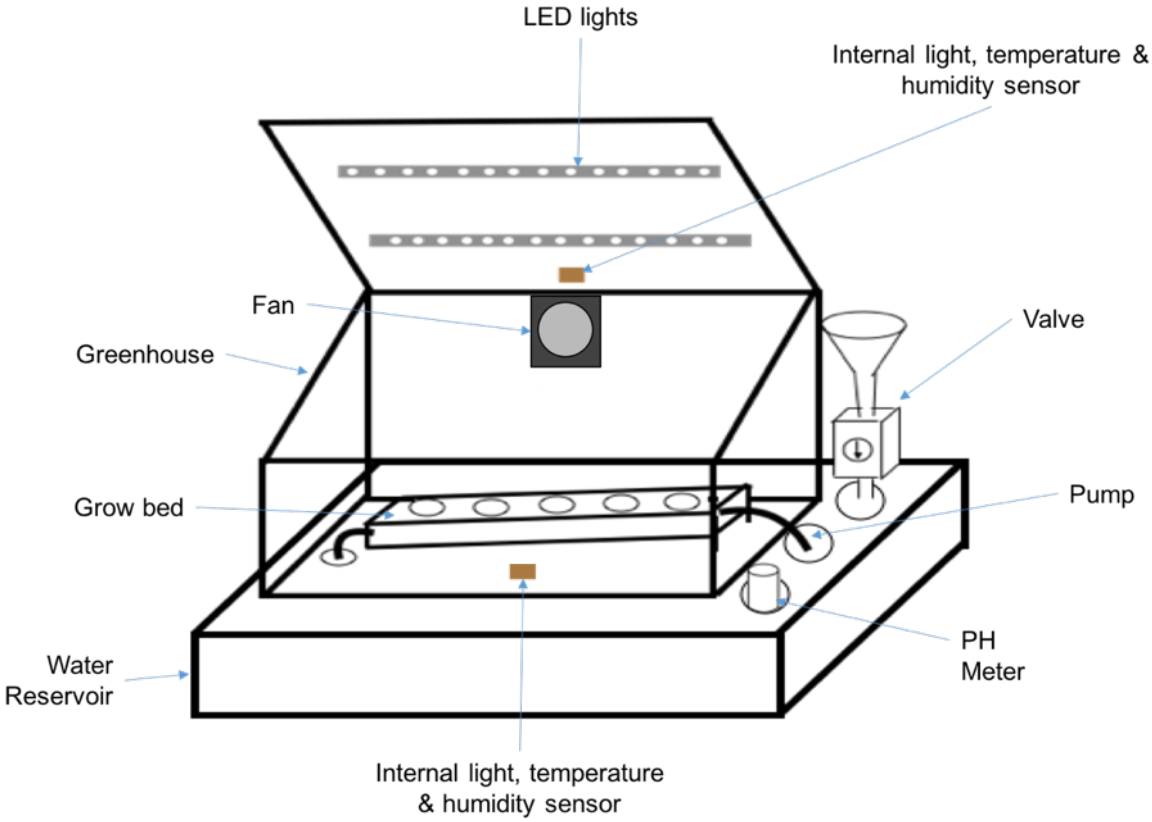

Figure 8.1 - Greenhouse system design

The system built in this project is a prototype testing what functionality is possible and how effective the various sensor and correctors can be. With this in mind the design had to include a large variety of sensor and correctors, though due size constraints of the lab a small foot print was necessary.

The main three growing elements of the design are, a storage container acting as a water reservoir with a small pre-built ornamental green house (45 cm/22 cm/35 cm) sat on top, with a grow bed sat inside. The grow bed holds 8 cages which house the plants with a tray around it on a slight slope causing water to flow out of the lower end, at the other end a pump pumps water up from the reservoir. The other two elements in the reservoir is a pH meter, a valve which holds lemon juice to increase the solutions acidity and the final sensor is a water level sensor measuring the volume of water inside the container. Inside and on top of the greenhouse is a pair of light, temperature & humidity sensors, which control a fan on the back and led strip lights on the lid.

All these sensors and correctors will the be connected directly to the Arduino Nano and this intern the Arduino with the Raspberry Pi [10] via a USB cable. The prototype will have lots of wires connecting hardware together and this must be secured with tape as to not cause hazard, in a commercial system this would be less of an issue due to the electronics being more spread out and in a consumer system the design would include tubes for wires to travel through inside the greenhouse frame.

Although the prototype is small and all it's hardware was predefined, the code is such that a large combination of hardware can be used. It's with this in mind that the Raspberry Pi, has been used as a master to up to 4 (or more using USB hubs) Arduino Nanos. By connecting the USB (Universal Serial BUS) connector of the Arduino to the Raspberry Pi, a communication link is formed allowing instructions to be sent to the Arduino to either take a sensor reading or make a correction and return data using a bespoke protocol. The Arduinos can then be set up by plugging in each components wires into the desired pins and, through the web interface, add the components to the database and program them to work as desired.

Hardware

The hardware used (as shown in the full circuit diagram in figure 8.1) is not incredibly complex, the sensors and correctors are all self contained requiring little additional circuitry in order to integrated them with the Arduino Nano. When additional circuity is required it's a simple MOSFET transistor, Relay, diode or resistor. These will be explained for each section below.

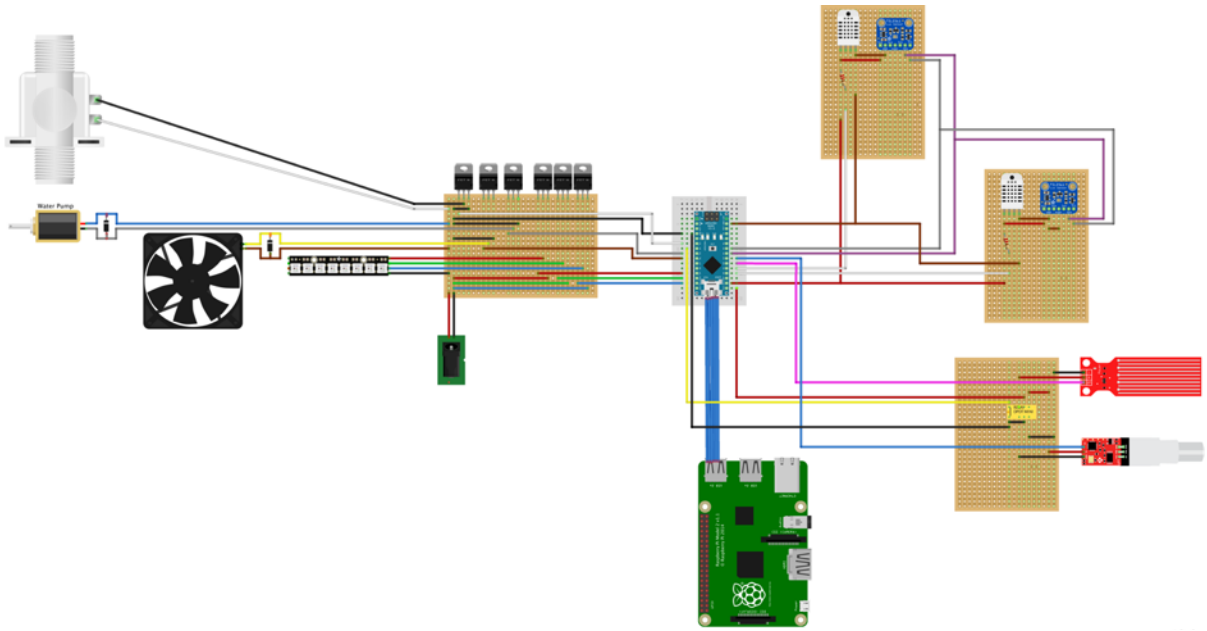

Figure 8.2 Full final hardware design

Correctors

All of the correctors in the system are powered externally from a 12v DC power supply. It was decided these would be externally powered as the max output of the Arduino Nano is just 5V with a maximum current output of 200mA, this is obviously no where near powerful enough to usefully run any of the components let alone all of them, so instead the Arduino controls 5V MOSFETS to control the external corrector power.

A close up image of the component circuit diagram is shown in figure 8.2, here you can see the solenoid valve in the top left, this is set up with it's positive terminal connected directly to the external power supplies positive and the valves negative to the drain of its transistor, with the corresponding source and gate going to ground and pin D3 on the Arduino (or whatever pin is set up as the valve pin in the database) respectively.

The water pump is shown as a DC motor in 8.2, as the pump works using a motor to expand and constrict a rubber tube to move the liquid. As a DC motor acts as an inductive load, when the transistor switches off the current to the motor, a small inductive voltage spike is created to keep the motor running. Without a fly back diode placed between the positive and negative terminals of the motor, this inductive current would flow backwards causing damage to the transistor.

The fan acts in an identical manor to the water pump requiring a fly-back diode due to the inductive load made by the motor.

Finally, the LED strip utilises 3 transistors to control the amount of current each LED colour (red, green and blue) receives. This allows for varying brightness and colour (wave length) of light to be produced in order to most efficiently complement and substitute sun light.

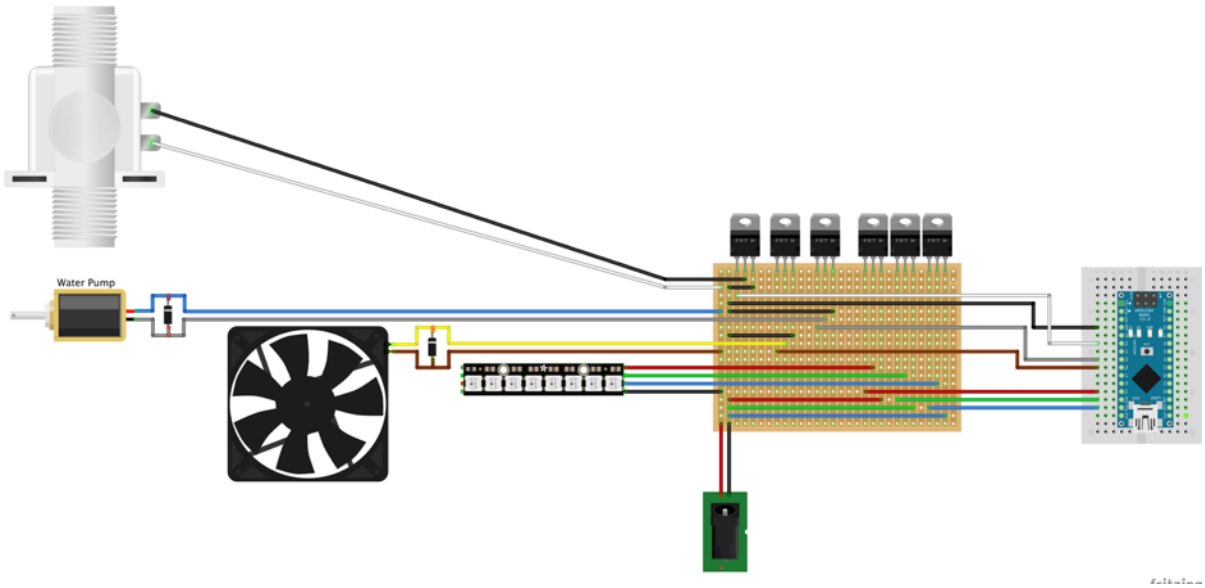

Figure 8.3 - Circuit diagram of correctors

Luminosity, Temperature & Humidity Sensors

The humidity, temperature & light level must be taken both internally and externally, this is due to the fact when a correction is working the inside conditions become the desired condition and with only internal sensors they would instantly shut off, then realise it needs the corrector and turn it back on, getting stuck in a loop. By adding external sensors, the corrections won't turn off until for example the external light is above the desired so LEDs may be switched off or lowered. It's for this reason one DHT22 (temperature and humidity) and one TSL2561 (luminosity) are paired up and soldered to a two separate Vero board strips so they can be placed one internally and one externally. Both sensors can run on 3.3V or 5V but to reduce wires we used 3.3v for both, the only difference between the internal and external sensor boards is the Addr connection on the TSL2561, the internal sensor is left floating, whereas the external sensor has the Addr pin connected to ground, this lets the two identical sensor have a different address on the I^2C bus allowing for each to be individually addressed and communicated with. The only other notable part of this circuit is the resistor between the DHT22s' sensor pin and its power pin. This is used as a pull up resistor to ensure full swing of the logic level when the sensor pin changes from output to input

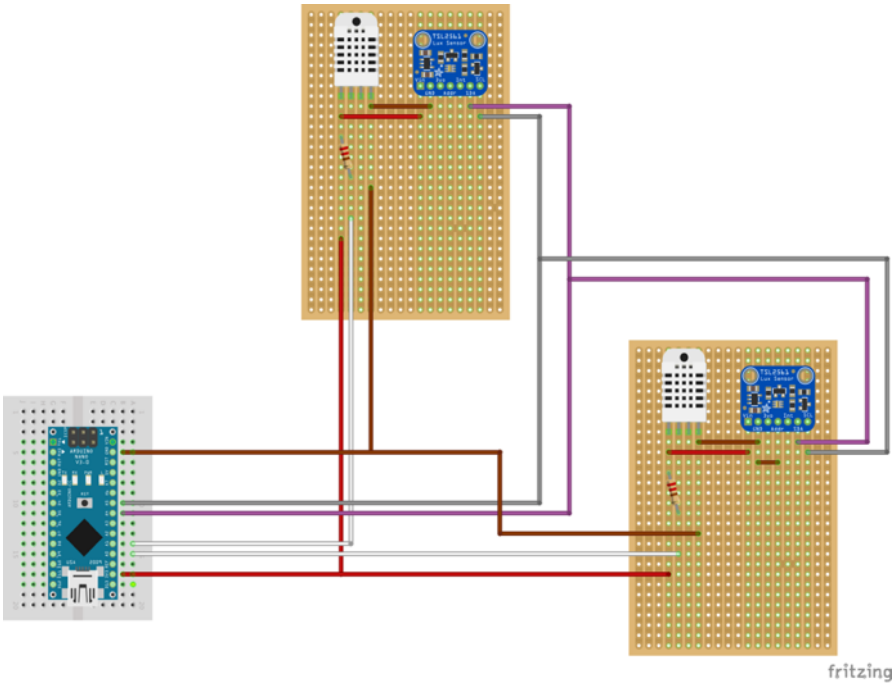

Figure 8.4 - Circuit design of external and internal luminosity, humidity & temperature

pH & Water level sensors

The final section of hardware is the pH sensor and the water level sensor. These two both work by measuring a current or change in potential in the water and therefore cannot be run together at the same time as one sensors voltage will interfere with the other. To avoid this the sensors are fully disconnected from the electronics using a relay. The DPDT (double pole double throw) relay is used for this task, by connecting both the Vcc and ground as the input to the relay on each pole, then use

one throw for the water sensor connection and the other throw for the pH connection. This means it is physically impossible for both sensors to have power going to them at the same time, thus removing the risk of interference.

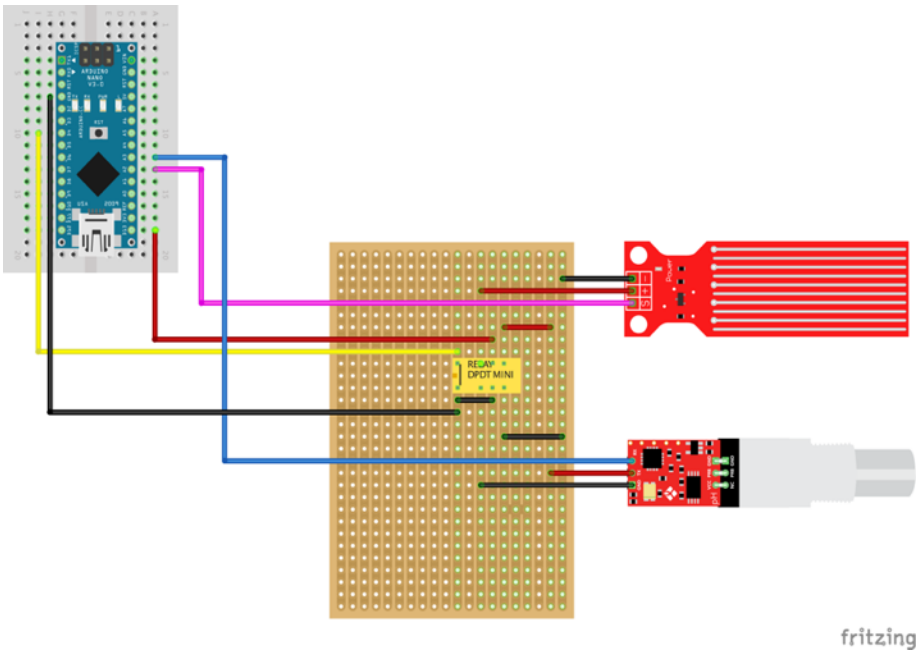

Figure 8.5 - Circuit diagram of water level and pH meter

Software

The software for this project is much more complex than its hardware as it must efficiently monitor, control and store all the environmental conditions while running a user friendly interface with the ability to customise the hardware. A long process of research and planning was carried out in order to come to the most effective way of amalgamating the Arduino, the raspberry pi and the web interface. The diagram below figure 8.5 shows the abstracted architecture and flow of instructions (white arrows) and flow of data (black & white checked arrows), between the various devices

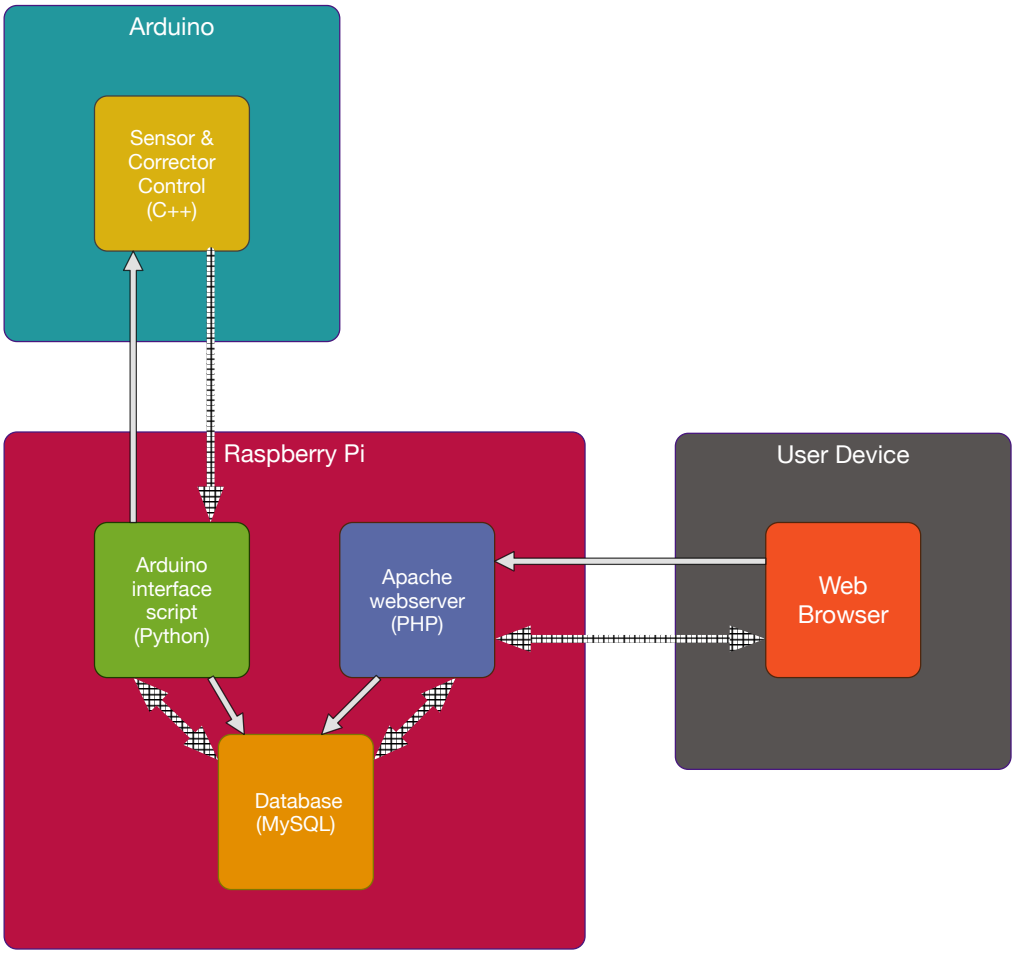

Figure 8.6 - Full system overview, solid white arrows indicate flow of instructions and checked arrows show flow of data

(Arduino, Raspberry Pi, User device) and the various modules running on each device. Effectively the brain of the system is the Raspberry Pi and its most important module is the database, this database holds all the data about the system including user login information, sensor readings and hardware configuration. The database is then accessed and queried by the apache webserver here the html websites are dynamically created integrating the data from the database and the user input which can then be sent to the user's web browser. A python script is also employed to act as an intermediary between the database and the Arduino, here the commands are created to instruct the Arduino to make sensor readings and carry out corrections according to the data stored in the database, it also saves the information returned to the database for later analysis. Finally, on the other side of this interface is the Arduino which acts as a slave simply carrying out instructions sent to it from the python script.

SQL Database design

The database is the most important component of the whole system and therefore required a great deal of planning and trial and error to come to the final design. The idea of a well designed database is one with no repeated or redundant data as this reduces efficiency of searches and can cause errors when updating or deleting data. So for example if the whole database consisted of just one table with each row including all the possible combinations of data would mean that all the information about the farmer, the farm and the region would be saved for every single sensor reading as well as null being saved for the remaining cells with no relevant information. This would obviously be incredible inefficient, slowing down searches and updates so instead the data being saved has been split into 22 different tables as shown in the full ER (entity relationship) diagram in figure 8.6. Unfortunately, due to the complexity of this diagram it's very difficult to understand so will go into depth in the individual sections.

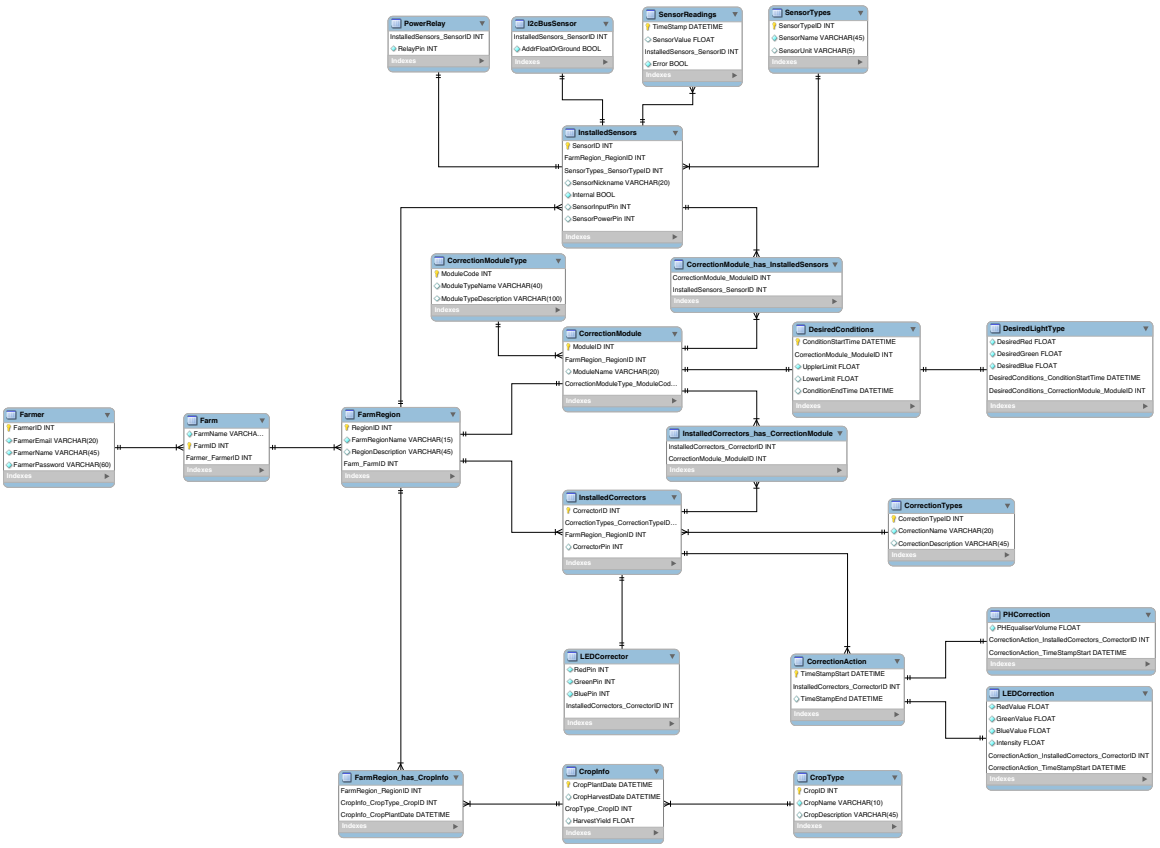

Figure 8.7 - Full entity relationship diagram for the database

The first section of the ER diagram (figure 8.7) shows the interactions and data of the Farmer, Farm and FarmRegion tables. The first two tables, Farmer and Farm, are slightly redundant in this system as the limitations of the hardware mean the Raspberry Pi would not be suitable as a

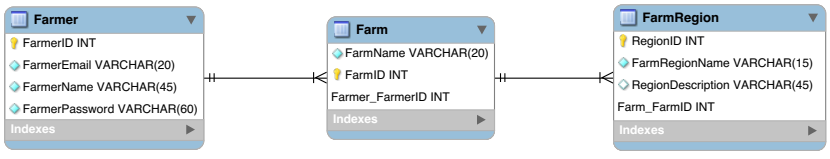

Figure 8.8 - Farmer, Farm, Region ER diagram section

server for multiple farms due to it's computing power constraints and the current interface between the Arduino and the Raspberry Pi. Planning for future scalability of the product, where the Raspberry Pi is replaced by dedicated offsite servers and the USB cable swapped for Wi-Fi modules, the importance of user information becomes clear. So a user, known in this system as a Farmer, has a name, email, ID and password associated with it.

The password is stored as a salted hash utilising the crypt_blowfish hashing algorithm, this algorithm generates a 22bit salt, a salt being a randomly generated selection of characters used to lengthen the password and add randomness in effect removing the power of rainbow tables to reverse engineer the plain text password from the hash. The algorithm creates a 60-character output which contains both the salt and the result of the cryptography (the hash), this is then saved in the database and the plain text password can be discarded. When logging in the next time, the salt can be taken from the saved password, then use this to hash the password being used to login and finally compare the saved hashed password to the user inputted hashed password [37].

Moving to the Farm table, each farmer can have many farms but each farm can only be run by one farmer. The Farm simply stores data about its name, ID and the farmer which runs it, in the future this may be expanded to store information about the farms location and company contact details but for now the notion of a farm is all that's required.

Each farm is then made up of one or more regions which also has a name, ID and a description. The region is really where the system built in this project begins as the small greenhouse built is only big enough to act as a single region.

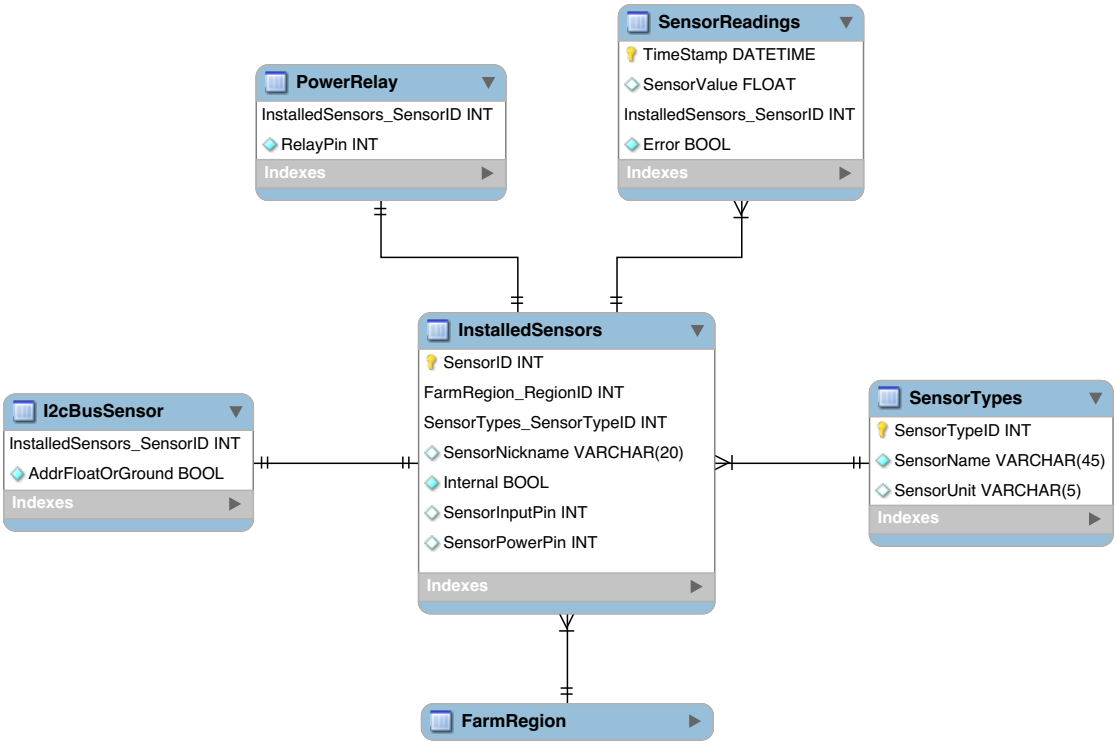

Figure 8.9 - Installed Sensors ER Diagram

One of the main purposes of including regions in the database is so all the sensors and correctors can be related directly to a given specified area, different regions may house different

crops or it may just be different greenhouses on a farm. But the next section of the ER diagram (figure 8.8) shows how sensor data is saved in relation to a region. The most important aspect of this section is how the SensorTypes table is used, this can't be changed by users of the system as the sensor type ID is used throughout the system, as part of the interface script, so the only way new types of sensors can be added will be through a software update where additional sensor types are added to the database and code to handle these sensors is added to the python interface script.

Once a region is created and the sensor types are saved, the user can then add multiple InstalledSensor, in this table the regionID and SensorTypeID are saved as foreign keys to link the tables and additional information such aas sensorID, SensorNickname, whether the sensor is internal or external and finally the pin numbers on the Arduino for the input and if necessary the sensors power.

As well as this basic sensor information there's two special case sensors which have a one to one relationship with an installed sensor, these are a PowerRelay sensor which is used with the water level sensor to fully disconnect its electronics from the water and an I2cBusSensor which utilises the I2C connection and is used for the luminosity sensors, the only additional information saved is about if the sensors ADDR pin is grounded or if it's unconnected.

Finally when a sensor reading is taken the data must be saved and this is done in the SensorReadings table, here the current date time, sensor reading and an error flag is saved along with the sensorID to link it to the required sensor.

The storage of sensor info and corrector info is very similar with an installed corrector being a link between a region and a corrector type with special cases and a way to record any correction action as shown in figure 8.9. The only difference between the InstalledSensors and InstalledCorrectors table is that the only pin number saved is the CorrectorPin though the only special corrector case is the LEDCorrector table which holds pin numbers for all 3 transistors for red, green and blue leds. Saving correction actions which is for example turning on or off the lights is saved with a start time stamp of when the corrector is turned on and an end time stamp for when the correction is finished. As well as this there are two specials cases for the LED corrections, which saves the colour of light used as well as it's intensity, and a pH correction which saves the volume of pH equaliser used.

Continuing to the next section where information is stored about which sensors cause changes to what correctors and at what desired condition, which is stored in the correction module and its associated tables shown in figure 8.10. The CorrectionModule table is the heart of this sub-section, it has a ModuleCode (whose data is saved in the CorrectionModuleType table) and this is used throughout the system to explain what kind of correction it is eg. pH correction or light Correction. Each region can have many corrections and each correction can link multiple sensors and correctors by employing two junction tables (CorrectionModule_has_InstalledSensors and InstalledCorrectors_has_CorrectionModule) these allow many to many relationships between the InstalledSensors table (& InstalledCorrectors table) with a correction. After a correction module is created the desired conditions can be set in the DesiredConditions table, the upper and lower limits for sensor readings are set and a start time must be added, the end time can be left blank for an open ended correction eg.pH or the end time may be set in order to schedule things eg. only 16 hours of light are wanted a day.

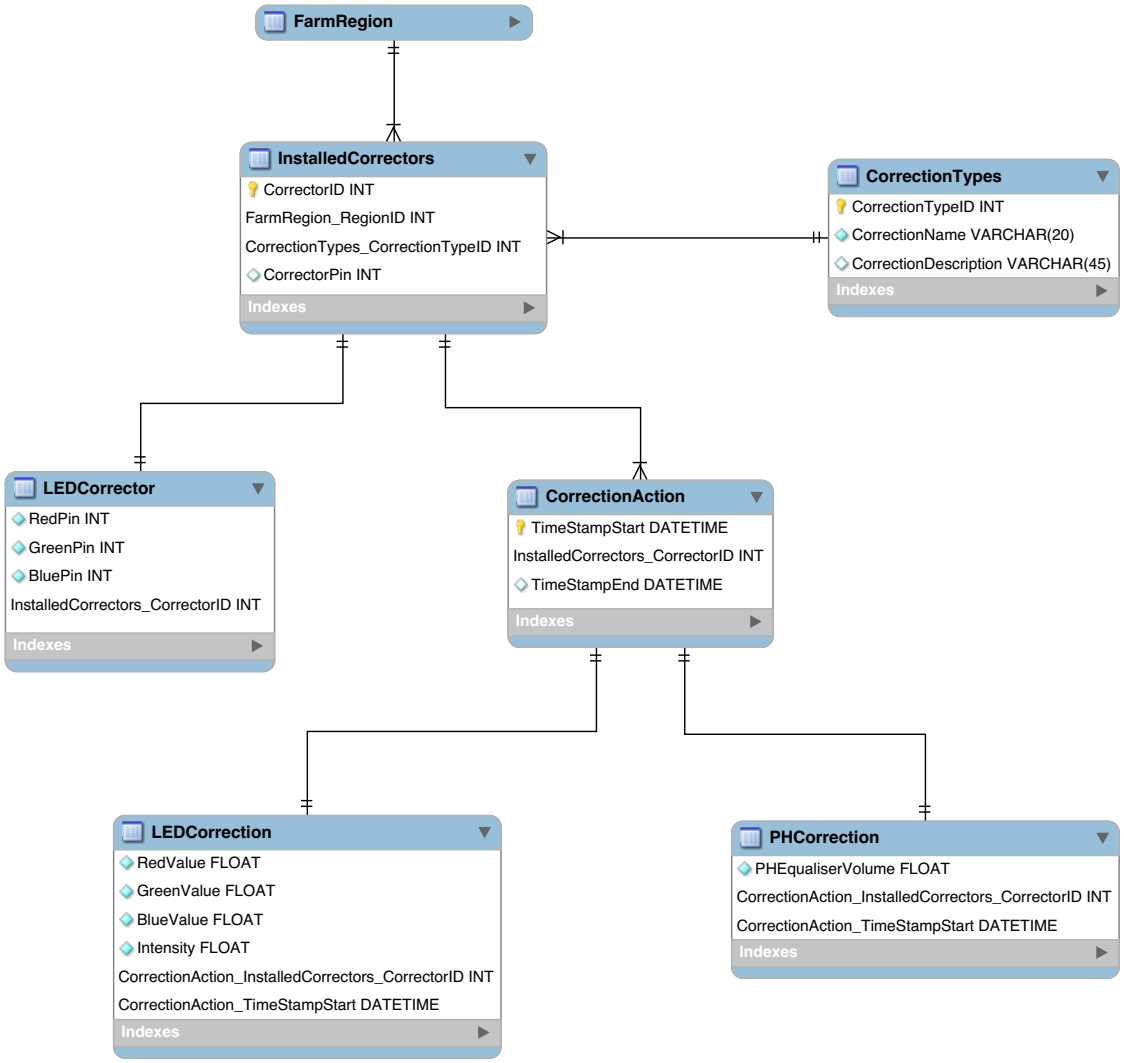

Figure 8.10 - Corrections section of ER diagram

The final section of the ER diagram is used in order to analyse the yields and success of a crop giving the opportunity to compare yields with different conditions set. As a crop can be planted over multiple regions on the same day for large systems or multiple crops may be added to the same region (or planting staggered over time) for small systems, the FarmRegion table to CropInfo table must have a many-to-many relationship, thus requiring the junction table FarmRegion_has_CropInfo. The junction table uses the Region ID from the FarmRegion table and The crop plant time and the crop type ID from the CropInfo table in order to create a unique relationship. Other information stored about the crop is a harvest date and a yield in kilograms.

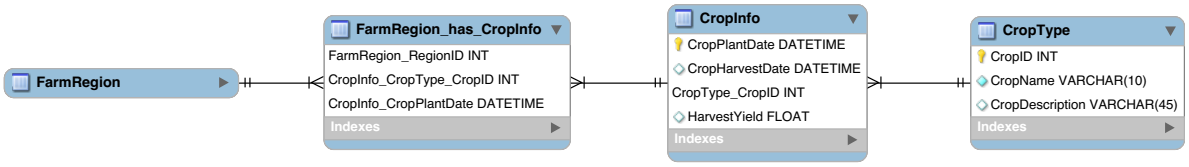

Figure 8.11 - Crop Info section of ER diagram

PHP user interface design

The user interface is the link between user and data stored in the database and, by extension, the interactions of sensor and correctors in the greenhouse. The user interface is built on a LAMP server stack, this is a commonly used combination of Linux operating system, Apache web server, MySQL server and PHP programming. The Linux operating system runs the hardware, in this case a Raspberry Pi running Raspbian, a special release of the Debian operating system. The Apache web server is what allows interaction of users external to the machine to access website files remotely. MySQL database holds data as explained above and PHP code is used to dynamically make files which are sent to users' computers, this process permits information specific to that user, their farms and crops.

PHP (PHP: hypertext pre-processor) [38] is a programing language in it's own right but can incorporate standard html mark up to create html documents including dynamically created data. This is possible through an Apache server paired with an intermediate software library like libapache2-mod-php7.0 [39]. In this program the PDO (php data object) extension has been used [40], which allows a unified connection between php and multiple types of databases., in this system the MySQL database.

The selection of pages built allows for a user to sign up, login and access their data. The most important section off the user interface is the graphing. This has been made possible using libraries and php wrapper from fusion charts [41] which are available free for non commercial usage. These require a json (JavaScript Object Notation) [41] dictionary as data input as this is a standard data-interchange protocol. This is dictionary is easily attained by encoding an array of array utilising the php function json_encode(). The difficulty of displaying graphs is getting the correct data out of the databse, as 5 different lines must be plotted for some module (ie. internal and external readings, upper and lower limits and corrector state). This wouldn't be too difficult on its own as all 5 can separately be queried using a simple SELECT WHERE. This issue is that the JSON dictionary starts with a category array containing all the labels for the x-axis, the next array after category is dataset which is full of arrays itself, each one holding information about one line, each line follows the same x-axis so in order to plot all the data they must all be on the same x-axis. Unfortunately this system is designed to save data as and when it is received meaning it's likely all 5 lines will have data save at a time which isn't the same. To get around this issue a selection of joins are required to get all of the data into one database on one timeline.

For example to get both start correction time and end correction time in the same time axis columns you'd require this queery (for the fan with correctionID 4):

```
SELECT
 TimeStampStart AS `TimeStamp`
FROM
 CorrectionAction
```

```
UNION ALL
SELECT
  TimeStampEnd AS `TimeStamp`
FROM
  CorrectionAction
WHERE
  InstalledCorrectors_CorrectorID = '4'
```

This method of joining can be propagated along joining all 5 bits of data into one table which can then be analysed separately by the PHP code ensuring the correct data is added to the correct array to be charted.

Python database/Arduino interface design

The design of the interface python script is described in the UML class diagram shown below (figure 8.12). The first class, ArduinoController, is the entry class to the system, here the serial connection is formed between the Arduino and Raspberry Pi using the serial standard class. Once the connection is made the script waits until an input string is detected, only if the input is 'Arduino Ready' will the script continue on to create a connection with the mysql database (using mysql.connector class) then ask the Arduino for the region ID finally entering a infinite loop (whilst the USBconnection maintains) to repeatedly call the two functions in the arduino_sensor_corrector_control class.

The arduino_sensor_corrector_control holds just two functions. The first, get_sensor_data_in_region is illustrated in the flow chart in figure 8.13, the general flow is that all

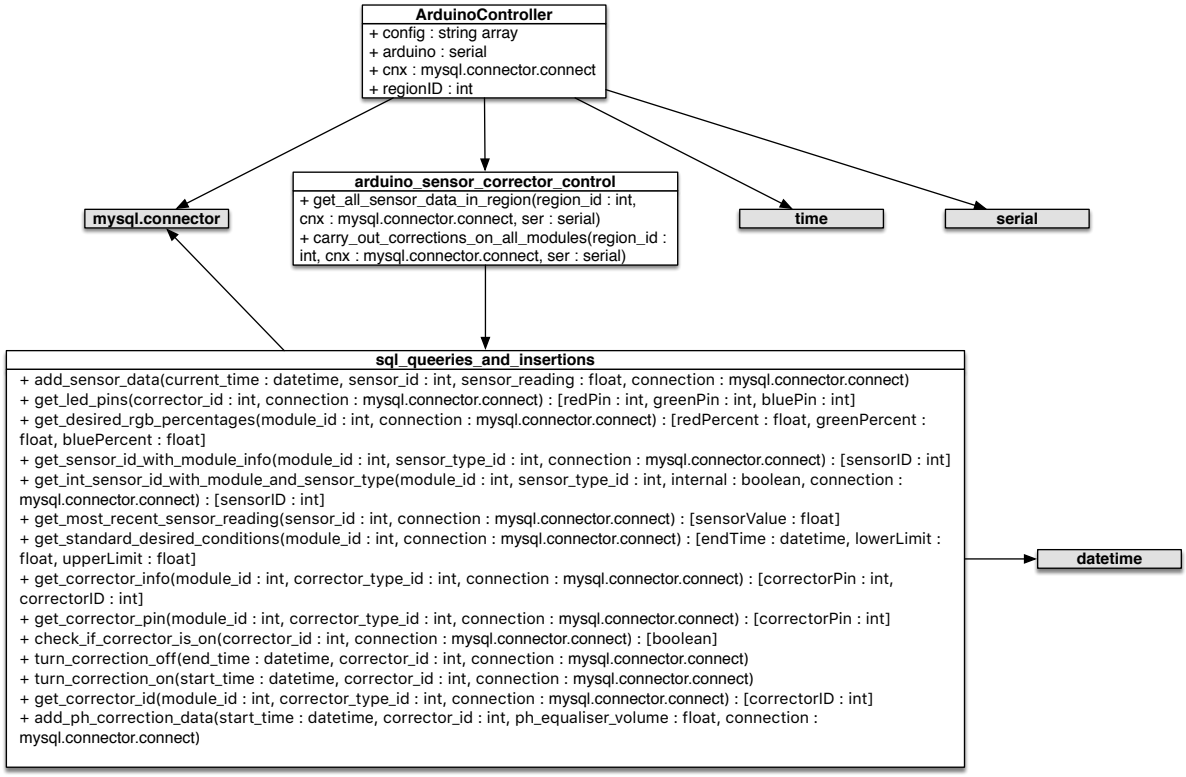

Figure 8.12 - Python code design for Arduino/ Raspberry Pi interface
white classes are user defined and grey classes are standard classes

the information about installed sensors, in the given region, are taken from the database, any null values are replaced by 255 so when sent to the Arduino it can ignore them, then depending on the sensor type ID a slightly different command is sent to the Arduino. The command consists of the letter 's', denoting the instruction is a sensor read, followed by a selection of numbers made up of 3 character, allowing the highest values of uint8_t data type (255) can be used, or Booleans which are represented as a single character (0 or 1). The script then waits for a response from the Arduino, converts the returned data to the sensor reading and saves this reading to the SQL database. The data returned most often is the raw analogue read from the Arduino so the algorithm developed in the calibration is used, other sensors such as the luminosity returns the exact reading in lux and the

DHT22 returns the reading multiplied by 10 (as this removes the need for floats on the arduino).

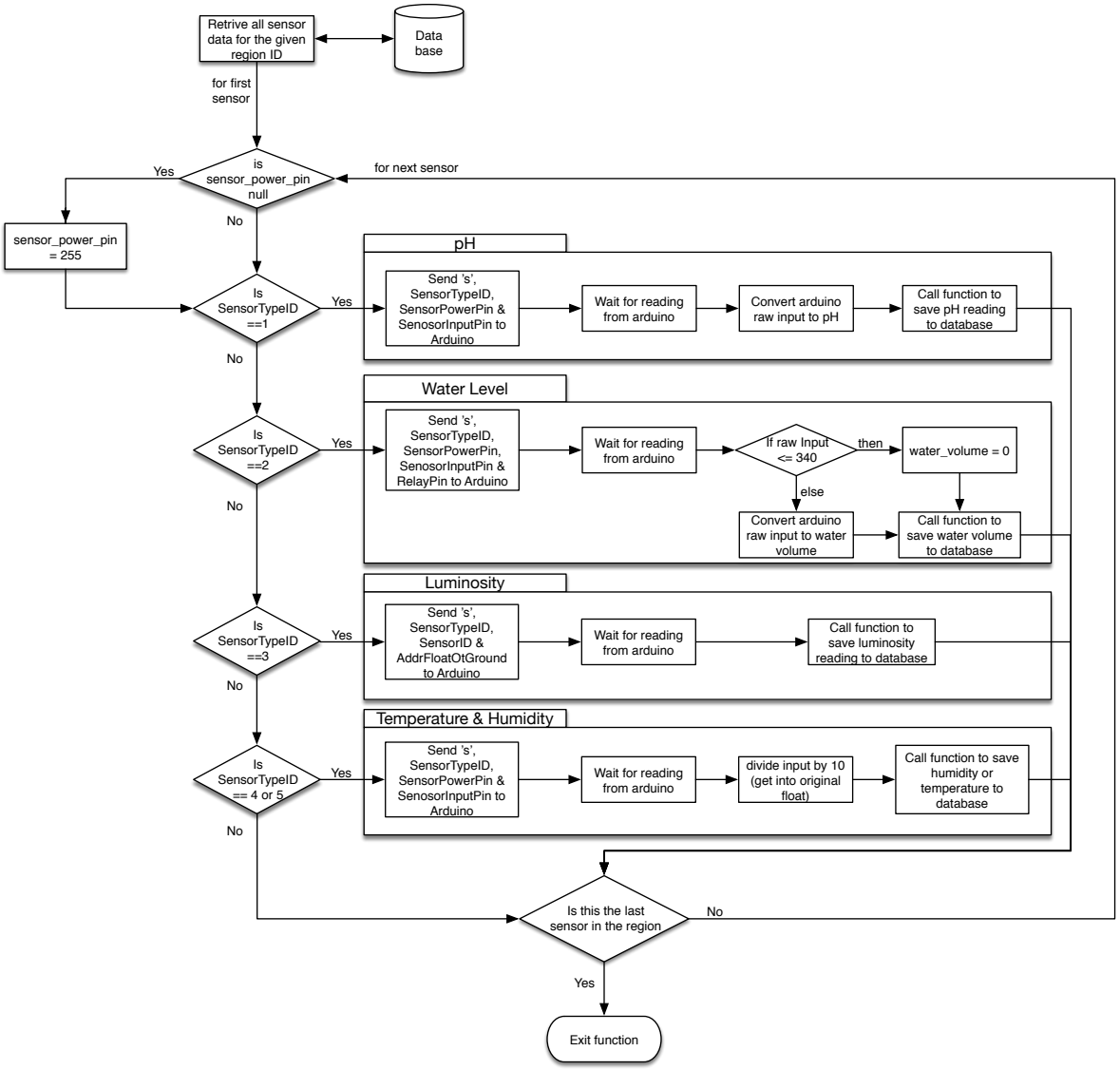

Figure 8.13 - Flow chart of the python function get_all_sensor_data_in_region

The second function, carry_out_corrections_on_all_modules, accesses the database for correction module data, it then uses each corrections ModuleCode to undertand what to do next. This is split up similarly to the sensor reads into a pH module, a water module, a light module and a temperature & humidity module. These modules all interact slightly differently with their sensor and correctors as shown in the flow chart in figures 9.14.a and 9.14.b.

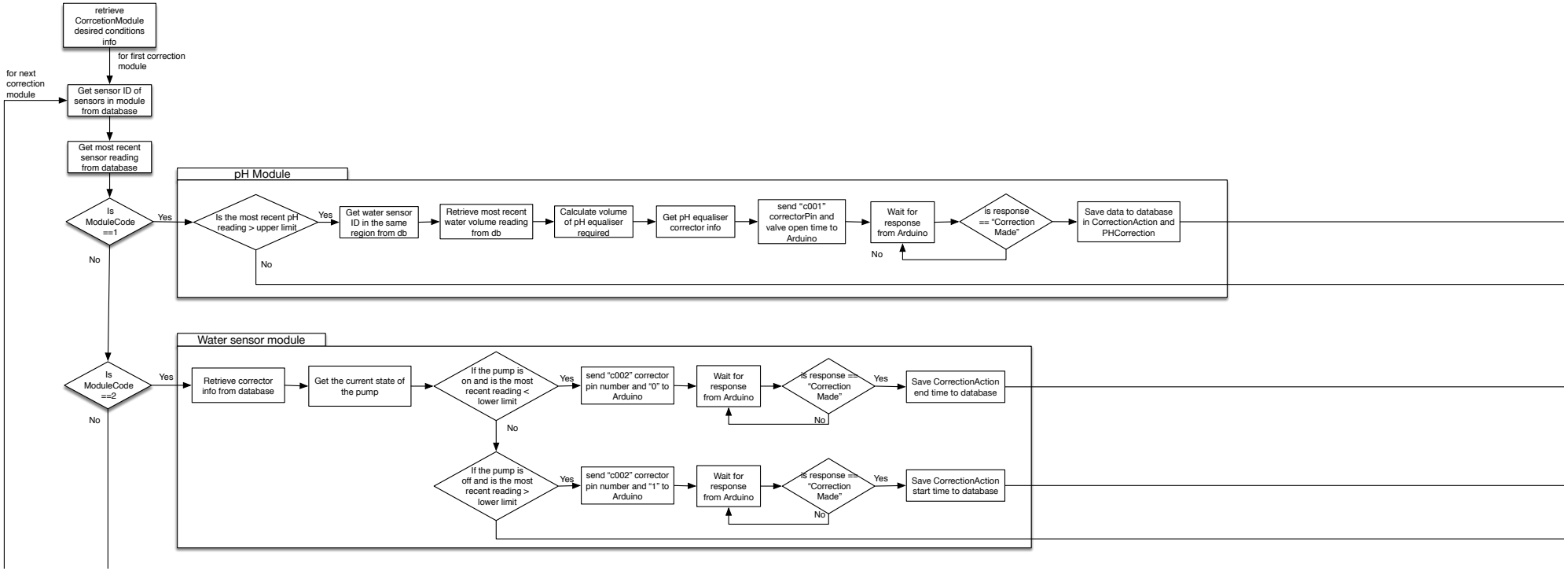

Figure 8.14.a - part a of a flow diagram depicting the carry_out_corrections_on_all_modules class

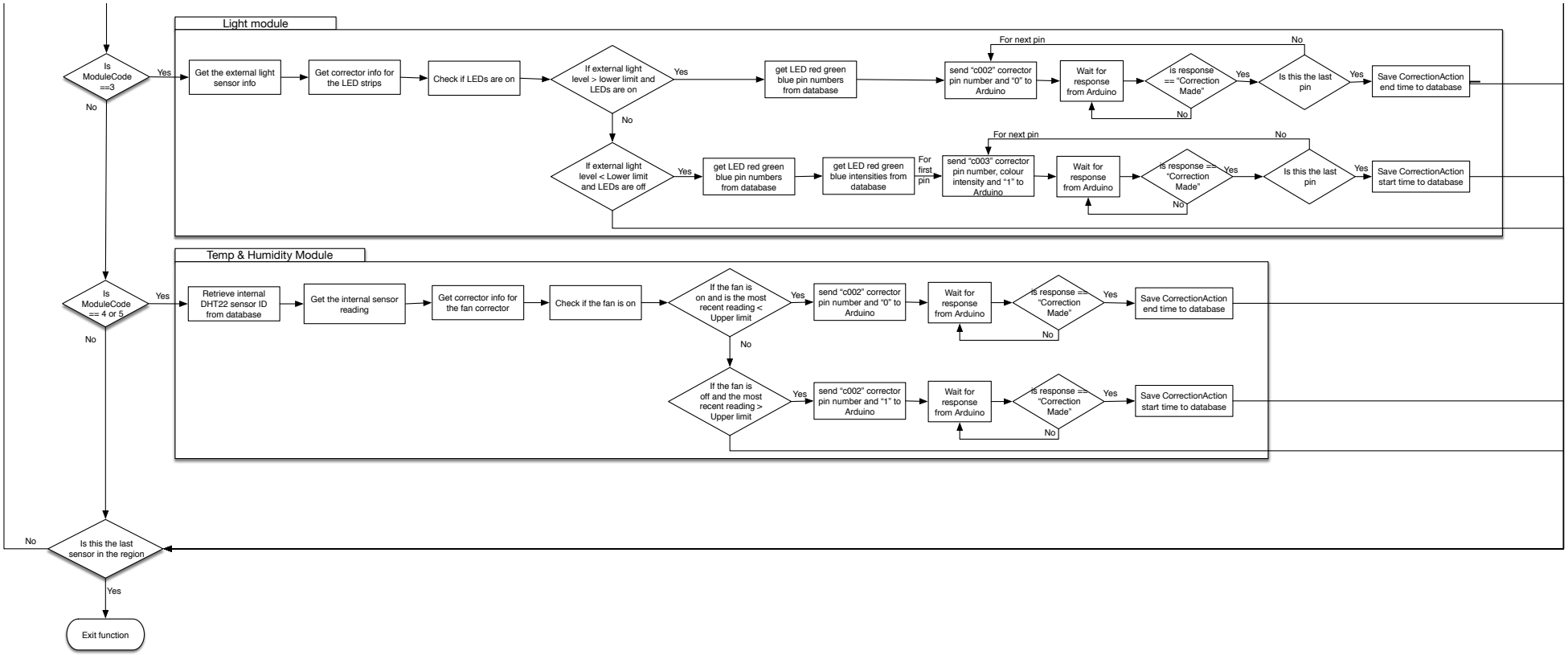

Figure 8.14.b .a - part b of a flow diagram depicting the carry_out_corrections_on_all_modules class

C++ Arduino design

The code for the Arduino changed quite dramatically throughout the design and build process. The initial plan was for the Arudino to take on much more of the decision making and simply report back to the Raspberry Pi when a sensor reading was made or a corrective action was undertaken. The issue with this approach however was the very limited dynamic memory of the Arduino so although the program code fitted on the Arduinos' static memory, in order to maintain the customisability of the system, such that the user can chose what and how hardware is used, all the data to the right of FarmRegion in the full ER diagram figure 8.6 (minus crop information) had to be saved dynamically (during run time) to the Arduino. This lead to the system appearing to work until all the functionality was running at once, at which point it would crash.

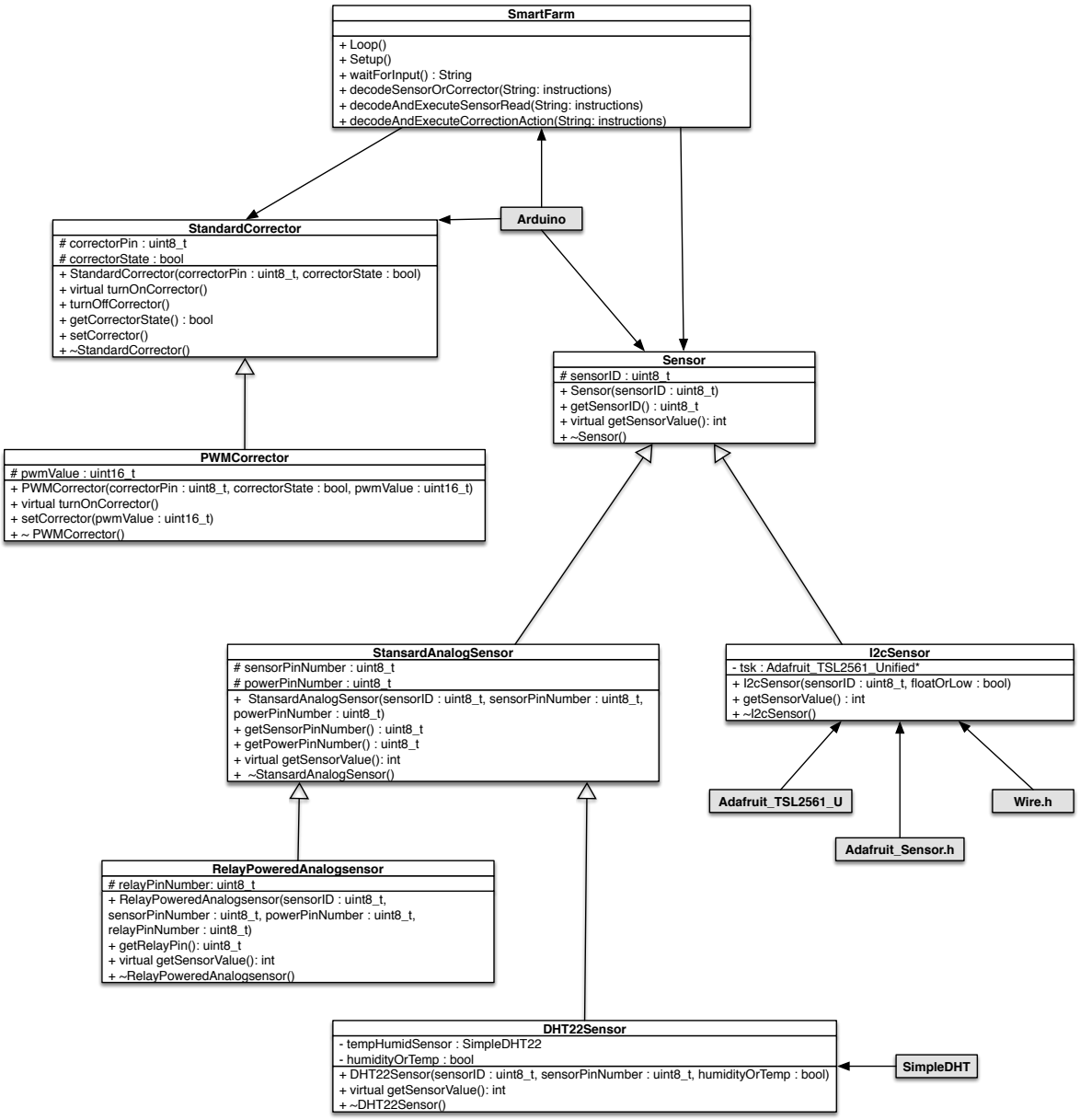

Figure 8.15 - UML Class diagram for Arduino code with custom classes in white and standard libraries in grey

So instead, the Arduino is now designed as a slave simply following instructions sent to it, creating new instances of objects every time it's asked to carry out a task but deleting this object as soon as the action is carried out. The UML (unified modelling language) class diagram for the

Arduino is shown bellow in figure 8.12, the SmartFarm class is the entry point of the code and acts in the same way as a traditional Arduino script including a Setup function which executes when the Arudino is powered on and a Loop function which is continually repeated until the Arduino is powered off. In this system the Setup function is utilised to create a serial link with the raspberry pi using a basic handshake where the raspberry pi waits for the Arduino to send the message 'Arduino Ready' and then the Pi will start sending commands. The Loop function call just two functions, the first is the waitForInput function, this pauses the flow of instructions until a character return is detected on the serial input, the preceding string of characters is the returned. This string can now be sent to the decodeSensorOrCorrector function which checks the strings first letter and if it's an 's' then calls decodeAndExecuteSensorRead or if it's a 'c' calls decodeAndExecuteCorrectorAction, passing on the sting to these functions.

The basic premise of the two decode and execute functions is that they read the first 3 characters which represent either the sensor type ID or corrector action code and then uses this data to add context to the remaining data.

So for the decodeAndExecuteSensorRead function a switch statement is used on the Sensor ID (first 3 characters after the initial 's') and different cases are carried out for each of the 5 currently programmed sensors (pH, water level, light, temperature and humidity), up to 256 different sensors can be programmed with the current data types. These cases simply instantiate the appropriate sensor object with the correct data taken from the input string and converted to the desired data types. The returned value from the getSensorValue function on the sensor object is then printed to the serial connection, returning the sensor reading to the Raspberry Pi.

The decodeAndExecuteCorrectorAction function acts in a very similar way to its sensor equivalent, though the initial three characters represent the three types of possible corrections to be carried out which are:

1. Turn on the given pin for short period of time (less than 30 seconds), wait the given time then turn the corrector off. This is useful for the solenoid valve used to add pH equaliser and a water mixer.
2. Set the given pins output state to the Boolean value passed, this can be used by any correction which doesn't require a varied output level, for example a fan.
3. Turn on the given PWM pin with a given level (from 0 to 256).

The rest of figure 8.12 displays the hierarchical nature of the specific sensor and corrector objects. The filed in arrow from the SmartFarm class to the StandardCorrector class shows that SmartFarm includes or uses the StandardCorrector class and any of it's sub-classes. The StandardCorrector saves the correctorPin to be used and the current state of the corrector, as well as functions to turn on and off the corrector, return the current corrector state and a virtual function set up the correction (not used in the standard corrector, included for polymorphism). The PWMCorrector class inherits all of the StandardCorrector class' functions and variables, depicted by an open arrow. The PWMCorrector class then adds the functionality to set the pwmValue variable and replaces the turn on function to correctly set the pins state.

The sensor classes are setup in a similar way, exploiting inheritance and polymorphism to create 4 specialised sensors which are:

1. A standarAnalogSensor which simply returns the analogue read from the input pin and can turn on a power pin if necessary to power the sensor.

2. A RelayPoweredAnalogSensor which has to turn on a relay pin before taking the sensor reading.
3. A DHT22Sensor which utilises the simpleDHT standard library to get the actual temperature or humidity reading and return this. The returns value is 10 times the actual reading as integers are used to transfer sensor values though the DHT22 sensor takes readings to one decimal place.
4. A I2CSensor specifically setup for the TSL2561 sensor. This class utilises three standard libraries:
    - Wire, this library holds the I^2C communication protocol and allows for the connectiuon between the sensors and the Arduino
    - Adafruit_TSL2561_U, this library holds the conversion algorithms to convert the output from the sensor into a useful measurement.
    - Adafruit_Sensor, this final library makes the sensor output a unified SI output.

9 - Experimental method

Sensor Calibration

pH Calibration

The standard way to calibrate a pH meter is with the use of a standard solution, this is a pre-mixed solution set at a specific pH with a high accuracy, most common pH solutions to use are 4.01, 7.02 & 10.01 [38]. The standard solutions used should technically surround the pH you're trying to measure; as the desired pH we're using is 5.5 to 6.5 a 4.01 & 7.02 standard solution could be used. The 7.02 solution is used to calculate an offset which would be the difference between the reading the pH meter outputs and the actual 7.02 value. The 4.01 solution would then be used and the gain should be adjusted to make the reading match.

Unfortunately, due to the hazards associated with using strong chemicals in an electronics lab, this method of calibration couldn't be carried out. Instead it was attempted to use pH indicator paper, by measuring tap water with a strip of pH indicator paper then using this measure to adjust the offset, followed by testing lemon juice with pH paper then adjusting the gain accordingly. This however also proved unsuccessful as the pH paper only had defined increments of 1 pH and any estimation of decimal points between the colours were just estimates. The readings the pH meter made without calibration (offset = 0.0 and gain potential device untouched) were within the pH paper readings so no accuracy could be gained and the pH meter is used as is, with an accuracy of ±0.5pH.

Photodiode/ LUX Meter calibration

The initial lux meter was a self calibrated simple voltage divider, implemented with one resistor and one LDR (light dependent resistor) as shown in figure 9.1. The circuit had to be calibrated by comparing the voltage reading taken from the centre of the circuit to a reading of the light level from an external luminosity sensor, in this case a phone running "Physics Toolbox Light Meter" [39].

Firstly, the exact resistance of the resistor should be measured using a multi-meter, for the most accurate calculations. Then, by running a simple Arduino sketch to capture analogue inputs (0-1023), a csv (comma separated values) can be outputted to the serial monitor of time & analogRead value. Simultaneously the luminosity is being measured on the Physics Toolbox Light Meter app on a phone, this also saves the data as a csv file with time and luminosity. The two csv files can then be combined to compare resistance to luminosity with a graph drawn and trend line equation calculated for the conversion equation. The graph of this is displayed in figure 9.2, and the trend line calculated has the equation:

$$Y=125047 \bullet \exp(-0.01x)$$

$$\text{Luminosity} = 125047 \bullet \exp(-0.01 \bullet \text{AnalogRead})$$

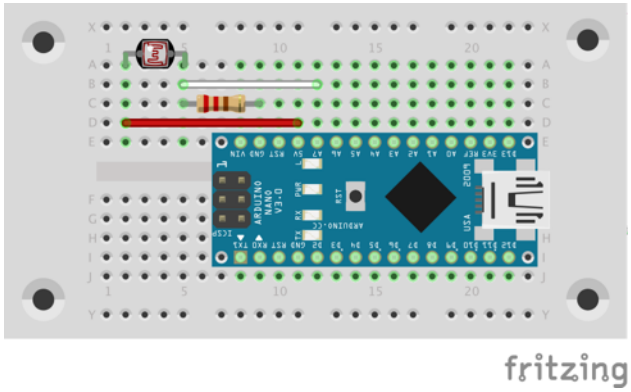

Figure 9.1 - LDR luminosity sensor circuit diagram

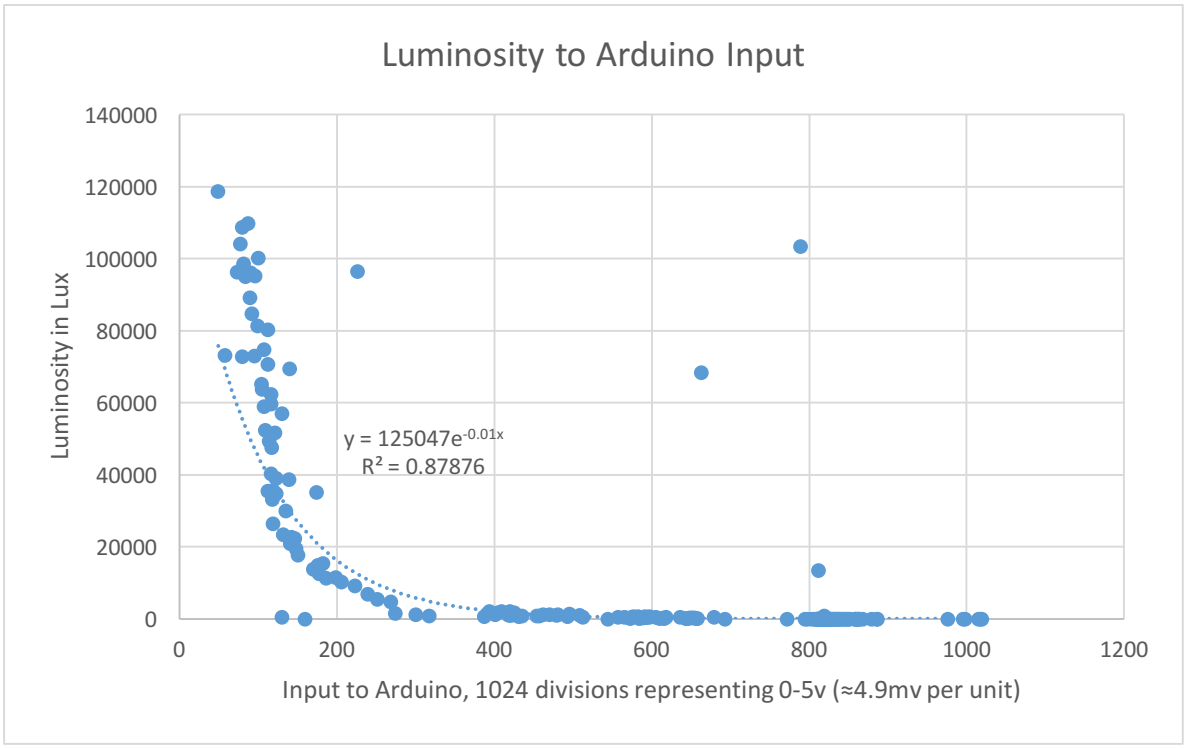

Figure 9.2 - LDR Light circuit calibration. Graph of Luminosity verses the analogue read of the Arduino input

TSL2561 comparisons

Due to concerns over the LDR lux meters' reliability and accuracy a TSL2561 luminosity sensor was tested against the LDR. The TSL2561 is a pre-calibrated sensor and communicates through the I^2C interface as shown in figure 9.3. In the testing between the two sensors the TSL2561 was taken as the true value and the LDR value was taken to see it's accuracy. By running an Arduino script returning the two values separated by a comma, these values can be entered into excel and displayed against each other on a graph [this data can be found in appendix LDR & TSL2561 comparison data]. This determines how accurate the LDR sensor is over a large range of lighting conditions.

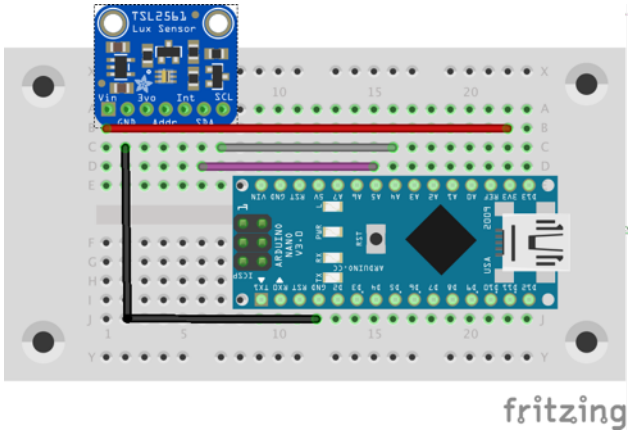

Figure 9.3 - TSL2561 sensor diagram

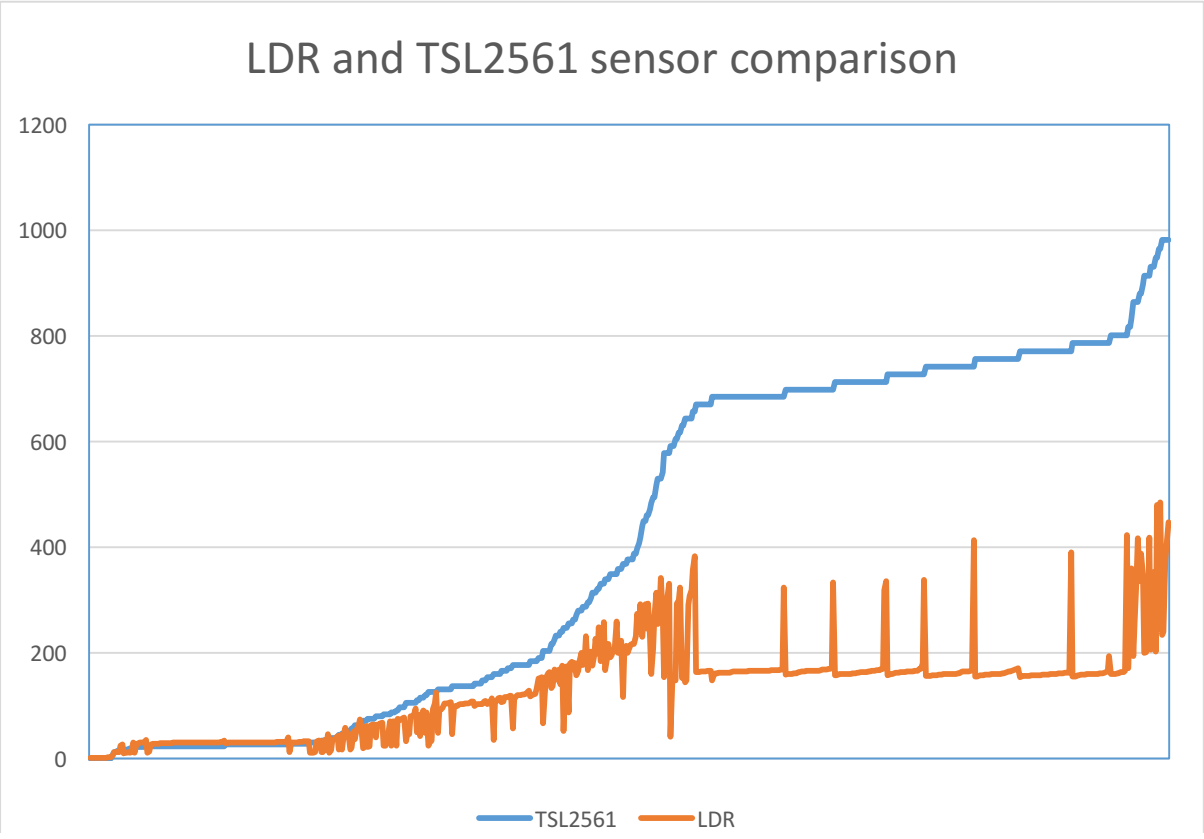

Figure 9.4 - LDR and TSL2561 sensor comparisons. Line chart showing the luminosity readings from the LDR compared to TSL2561

The luminosity sensors comparison graph is shown in figure 11.2. The LDR appears to match the TSL2561 quite accurate in the low ranges up to about 150lux, but from here the two readings separate dramatically, making the LDR unusable in a high light changing environment.

Water level sensor calibration

The water level sensor shown in figure 9.5 works by the fact that when more water surrounds the sensor the conductivity increases between the vertical strips meaning more current flows through them and a higher output voltage from the signal pin (S) is observed. To calibrate this, sensor lines were drawn on the sensor every 5mm (shown on the diagram figure 9.5 to the right of the sensor), the sensor was then lowered into water noting the analogue input to the Arduino at every 5mm interval [table of all data in appendix a) water level calibration data]. This was repeated with both a 5v input and 3.3v input. The graphs of these could then be potted, the trend line calculated and this equation used to convert analogue input to a water height.

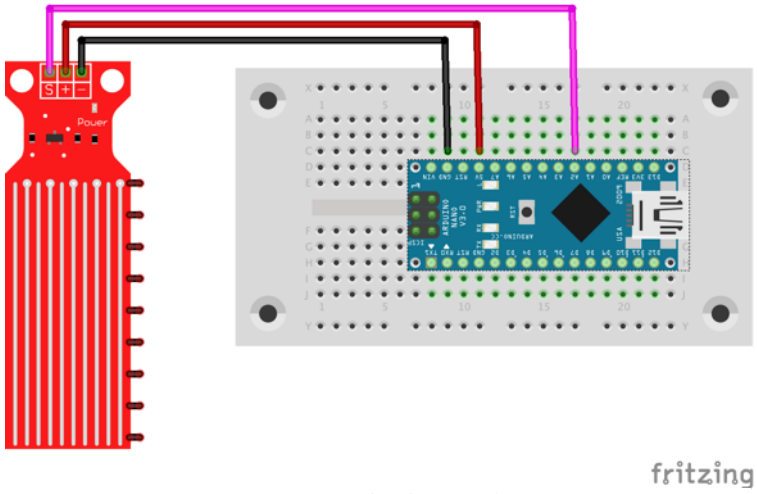

Figure 9.5 - Water level sensor diagram

The voltage was chosen at 5v so it could be the same as that for the pH meter (making it easier as they share a relay) so the output characteristic was plotted and the trend line with the least variance was chosen and plotted in figure 9.6. The equation for the conversion is as follows:

$$WaterDepth = 9 \bullet E\text{-}25 \bullet AnalogRead^{9.1084}$$

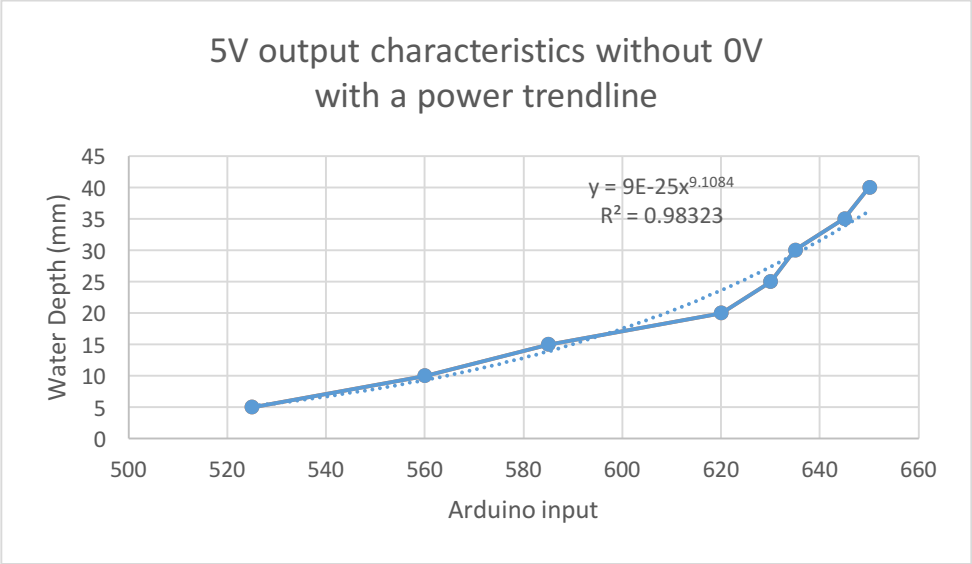

Figure 9.3 - Output characteristic of water sensor calibration with 5v input

Corrector calibration

Valve flow rate

The solenoid works by opening and closing a valve depending on whether an electric field is applied to its contacts or not. The amount of liquid to pass through the valve per second (its flow rate) differs depending on the pressure of liquid entering the valve. As in this system only a small amount of liquid is to be gravity fed through the valve, it's flow rate must be fairly slow. To calculate the flow rate a funnel is attached to the valve feeding in water, underneath the valve sits a measuring jug and the time taken to release 50ml of water is recorded (See figure 9.7). Now the average time taken for 50ml can be calculated and from this the average amount of liquid per second.

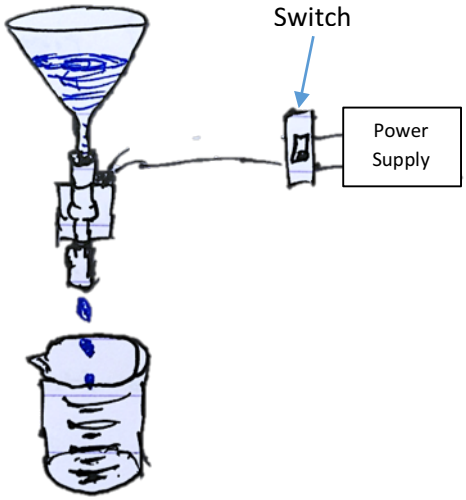

Figure 9.7 - Valve flow rate configuration

Table 9.1 – Valve calibration chart for flow rate as change in time and change in water volume is measured

Volume (ml)	**Volume change (ml)**	**Elapsed Time (Minutes)**	**Time change (seconds)**
50	50	1:50	90
100	50	3:50	120
150	50	5:50	120
200	50	6:55	65
250	50	8:35	100
300	50	10:10	95
350	50	11:40	90
400	50	13:30	110
Average volume change	**50**	**Average time per 50ml**	**98.75**

The measurements taken for this calibration are shown in table 9.1 with the flow rate calculated as:

Average volume change/ Average time per 50ml = flow rate
50/98.75 = 0.5063291139 ≈ 0.5ml/sec

10 - Results and Calculations

The physical system

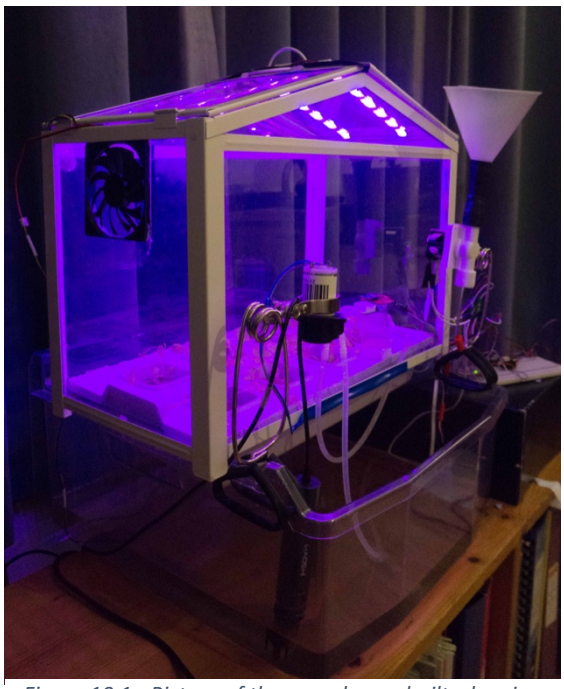

Figure 10.1 - Picture of the greenhouse built, showing the fan, pump and pH meter

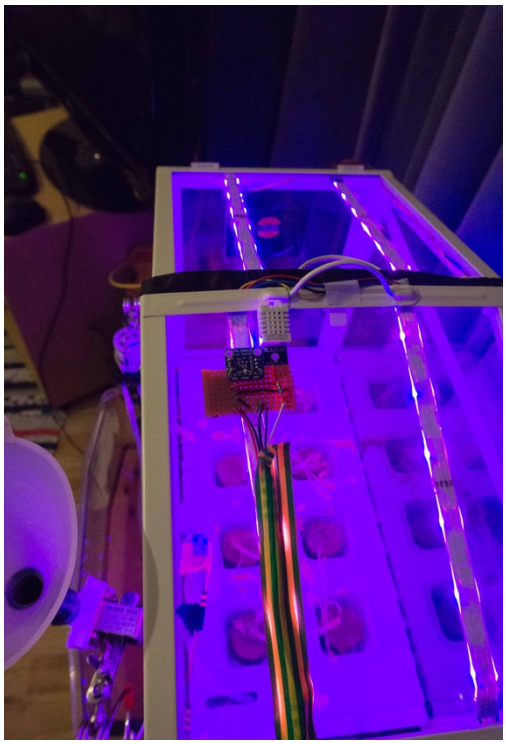

Figure 10.3 - Picture of the greenhouse built, showing the external DHT22 temperature and humidity sensor and the TSL2561 luminosity sensor

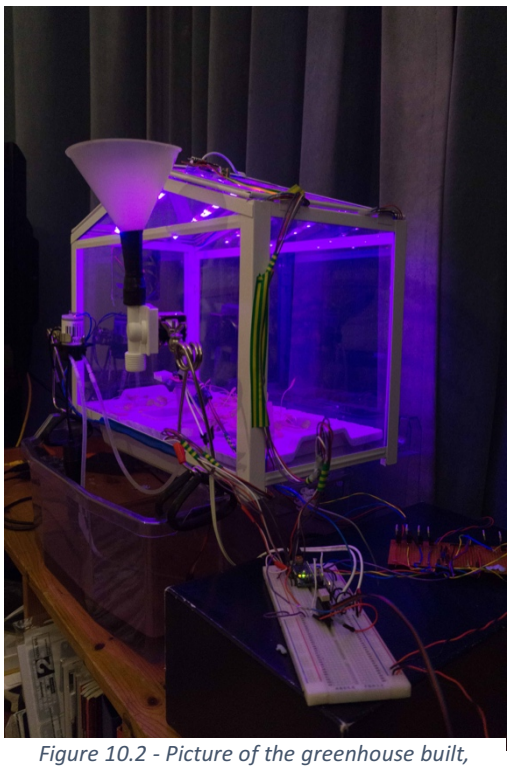

Figure 10.2 - Picture of the greenhouse built, showing the Arduino, Solenoid valve and light control circuitry

The systems user interface

Sign up

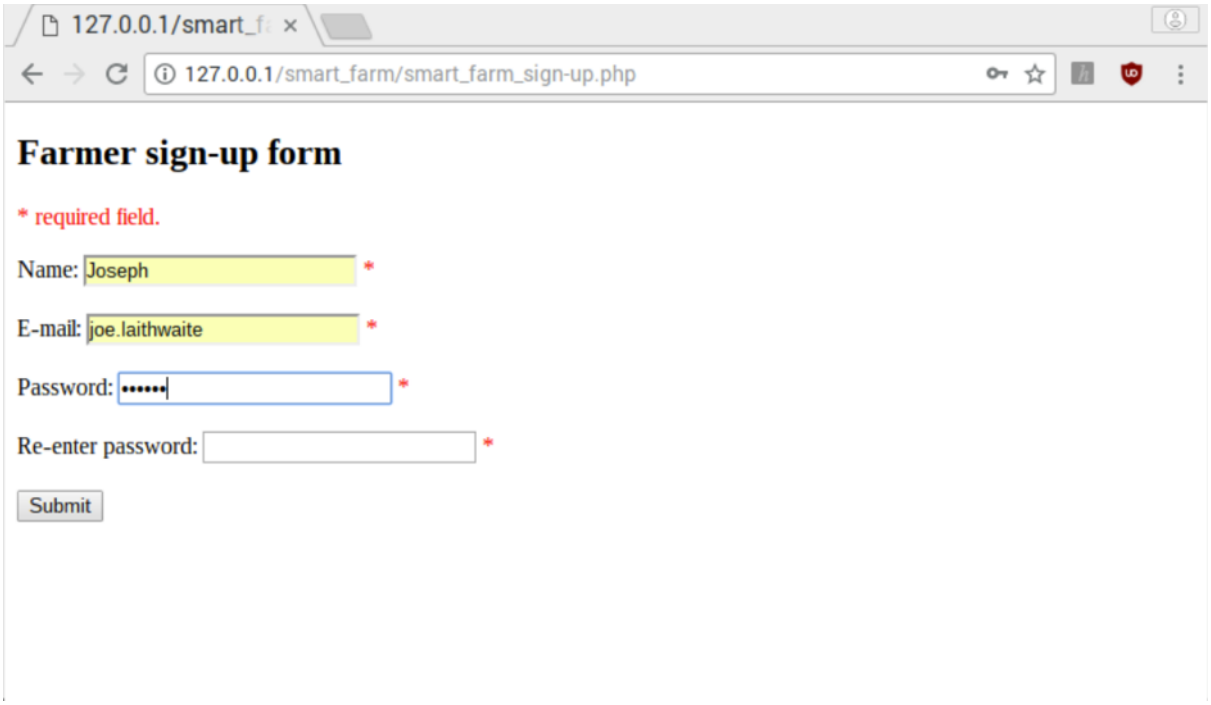

Figure 10.4 - Screenshot of the sign-up page of the user interface with an invalid email address being tried

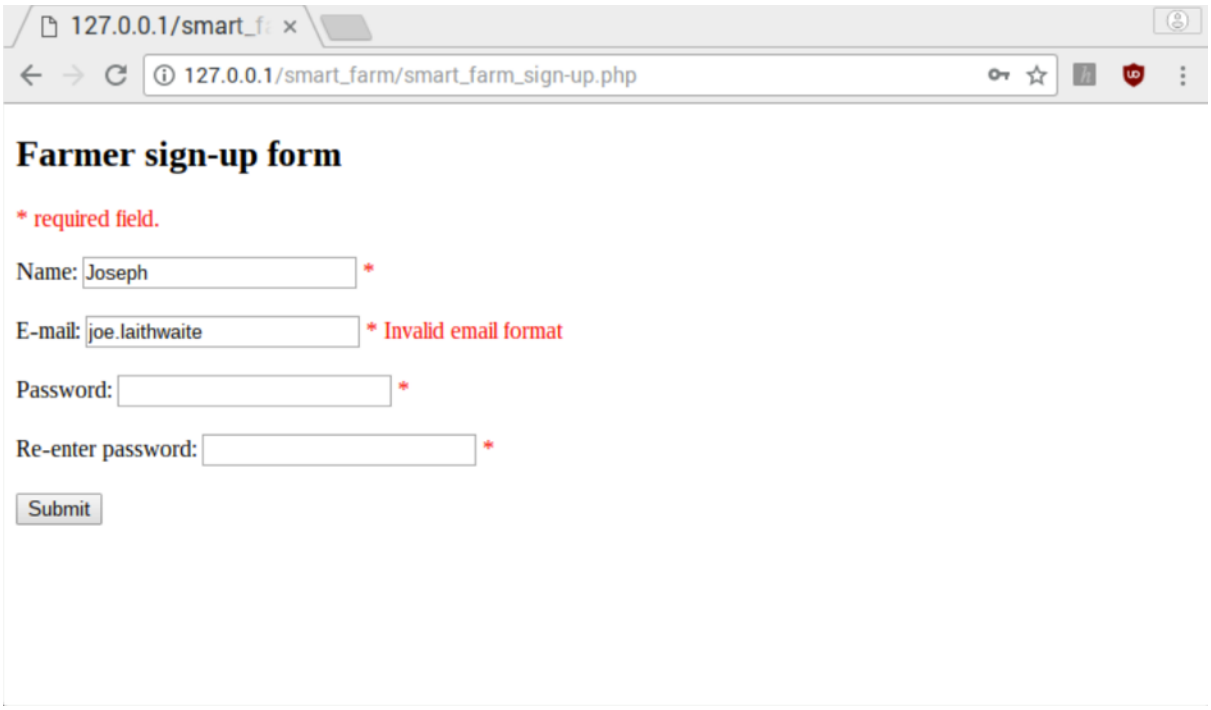

Figure 10.5 - Screenshot of the sign-up page of the user interface showing an error message after an ncorrcetly entered email

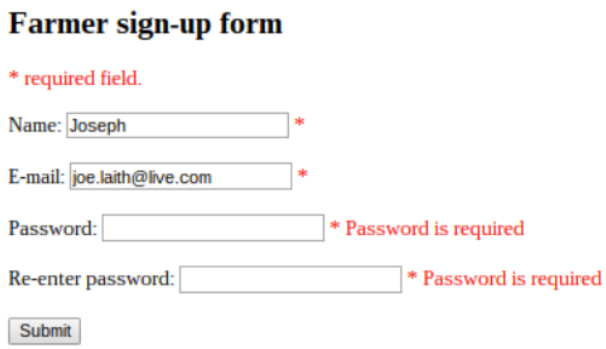

Figure 10.6 - Screenshot of the sign-up page of the user interface showing error messgaes when no passwords are submitted

Login

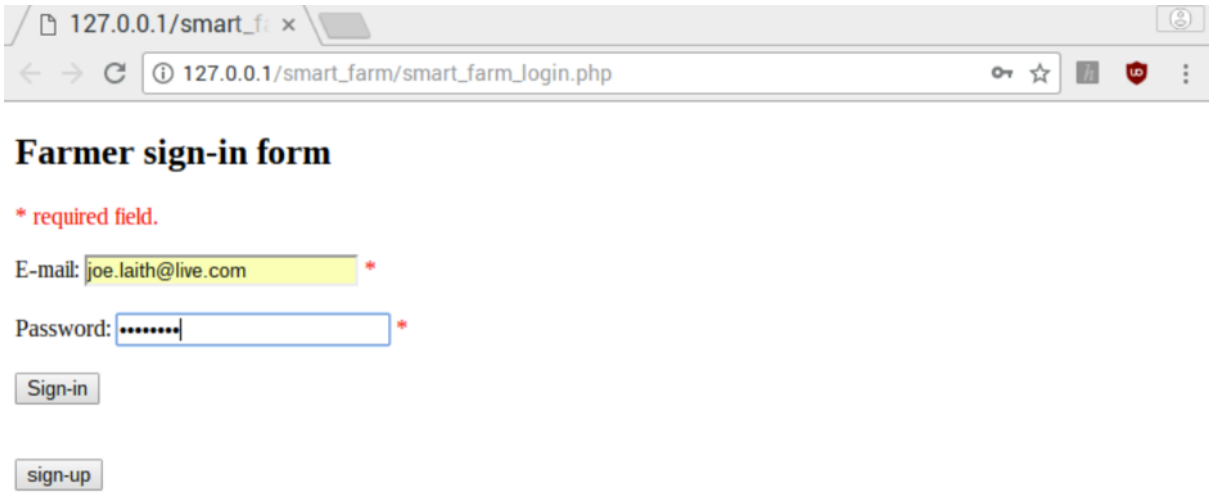

Figure 10.7- Login screen of the user interface with a valid email and password

Figure 10.8 - Welcome screen of the user interface after successfully signing up or logging in

Editing hardware

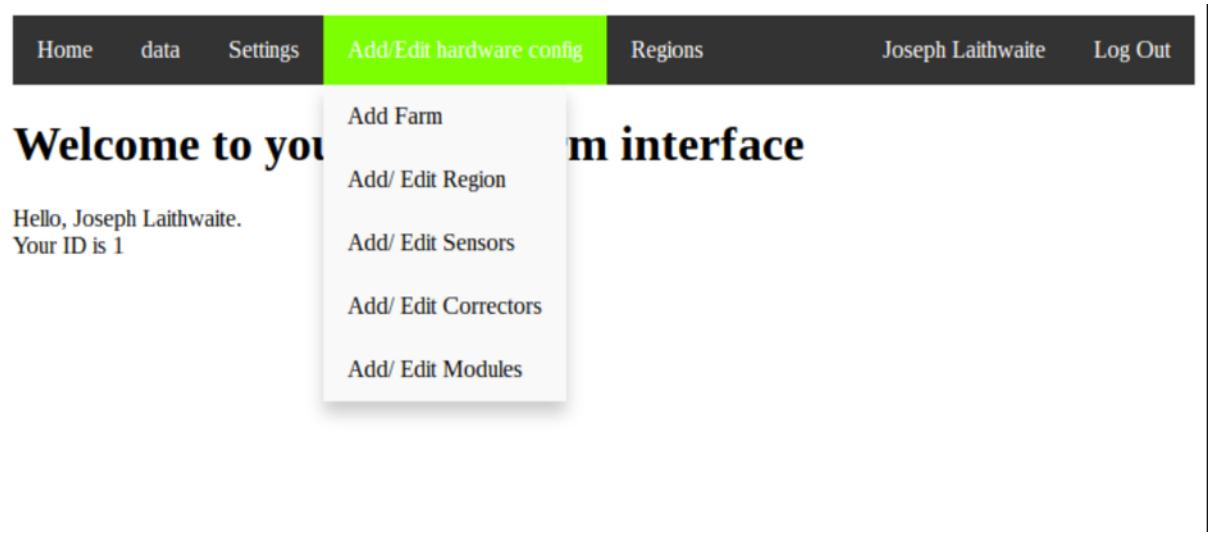

Figure 10.9 - Screenhot of user interface showing Add/ edit hardware menu options

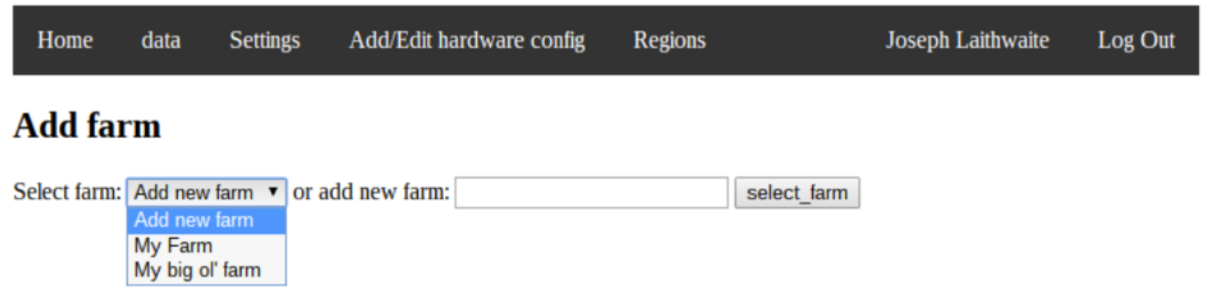

Figure 10.10 - Screenshot of user interface showing options to add or select a farm to edit, it's hardware

Figure 10.11 - Screenshot of the user interface showing the ability to select or add a new region with the options to edit sensor or correctors

User interface data output

Figure 10.12 - Screenshot of the user interface showing the menu options of data to look at

Luminosity Module

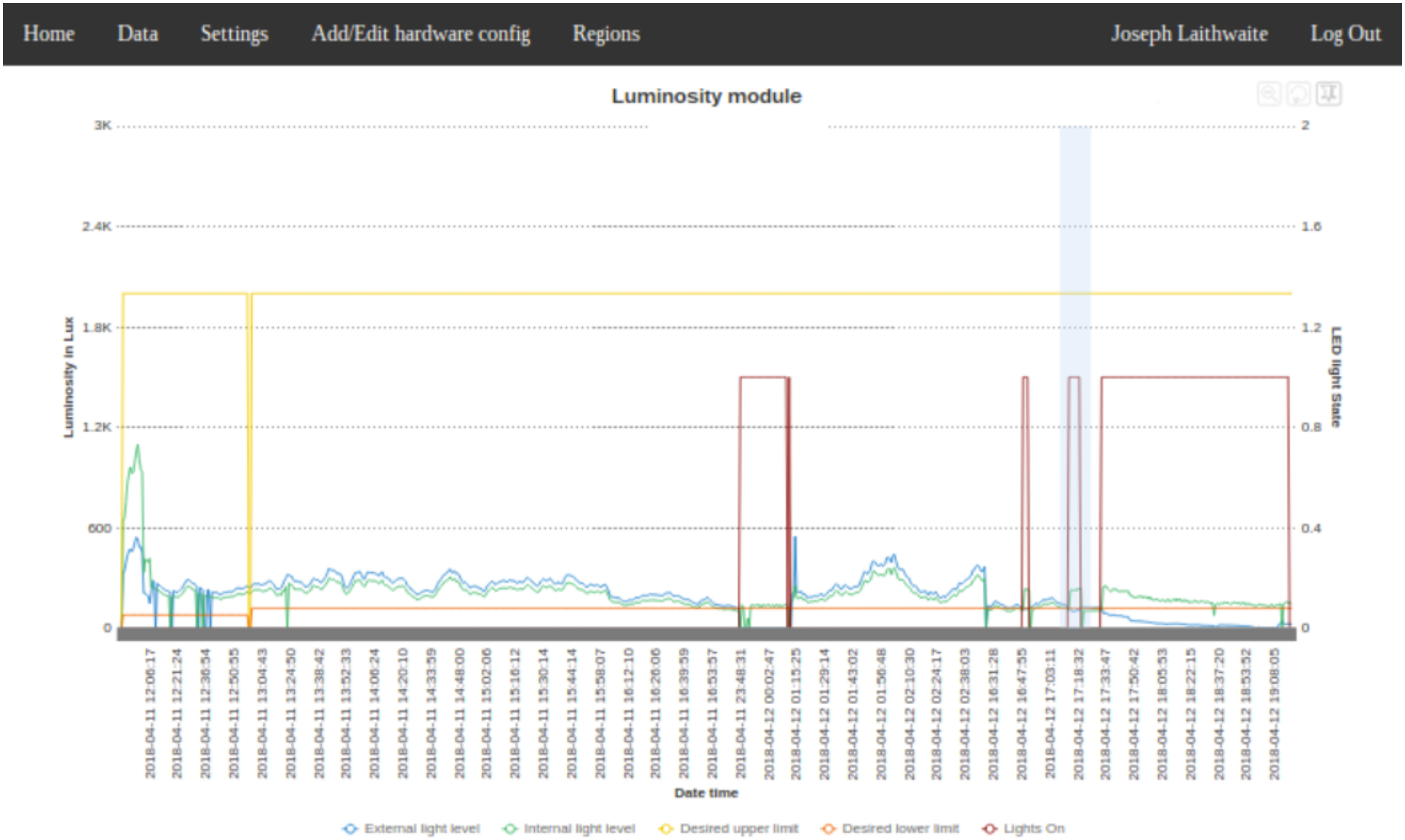

Figure 10.13 - Screenshot of the full data recorded by the Light module including desired conditions, sensor reading and LED actions

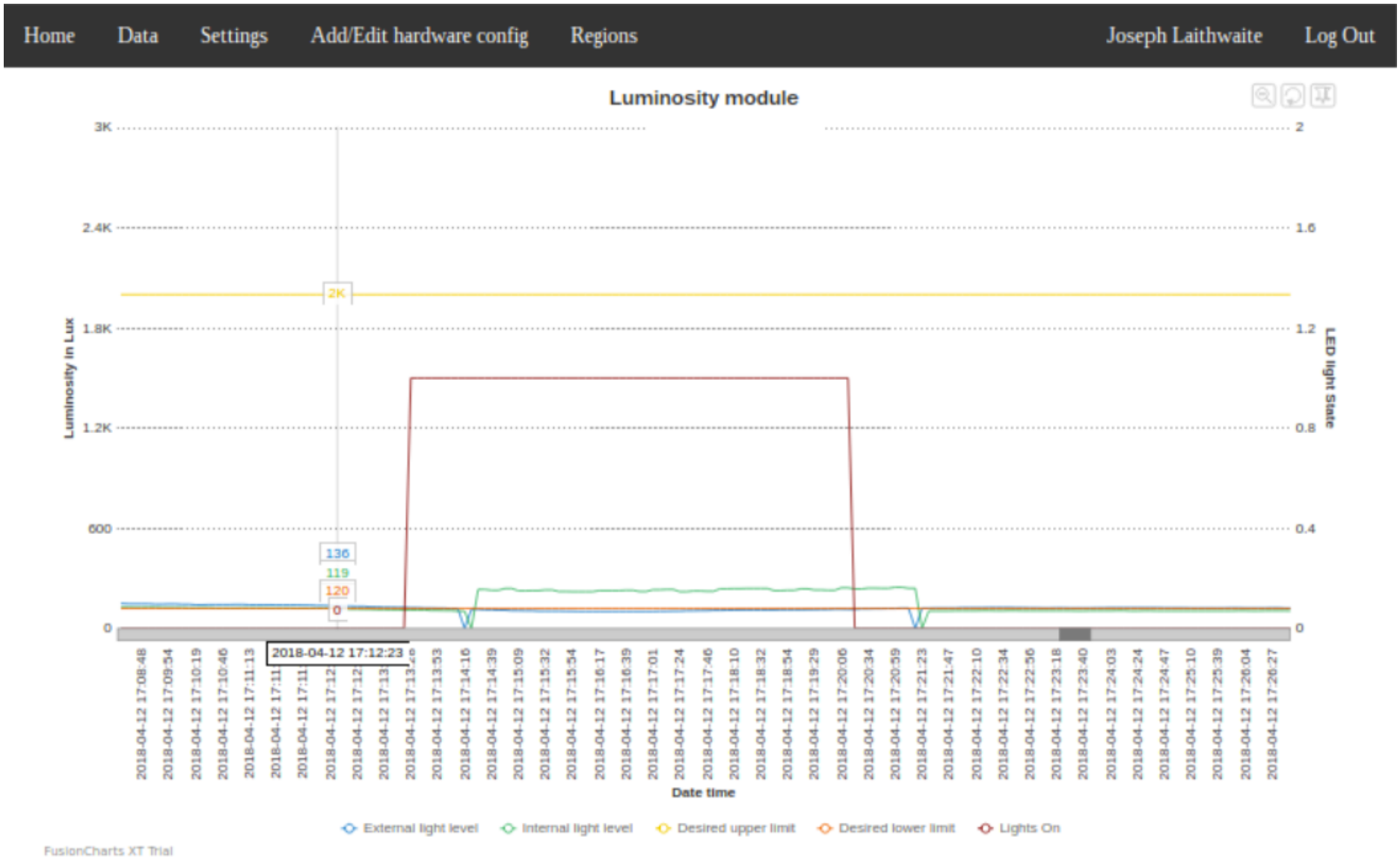

Figure 10.14 - By highlighting a section from the graph in figure 11.13, a zoomed in graph is displayed. Here the data surrounding a turn on and off

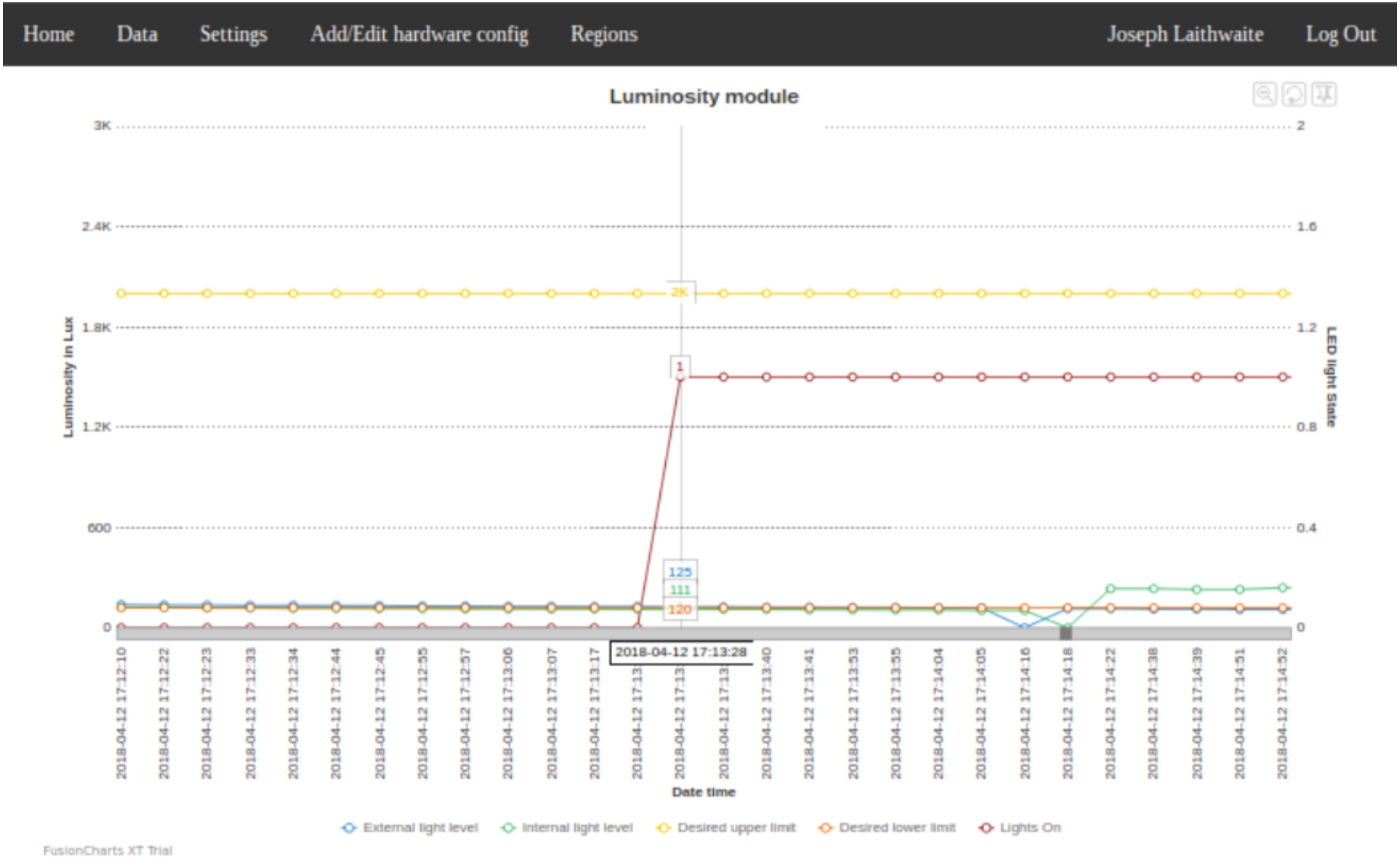

Figure 10.15 - screenshot of a zoomed in section of the luminosity module graph with values of the time on time for the Lights

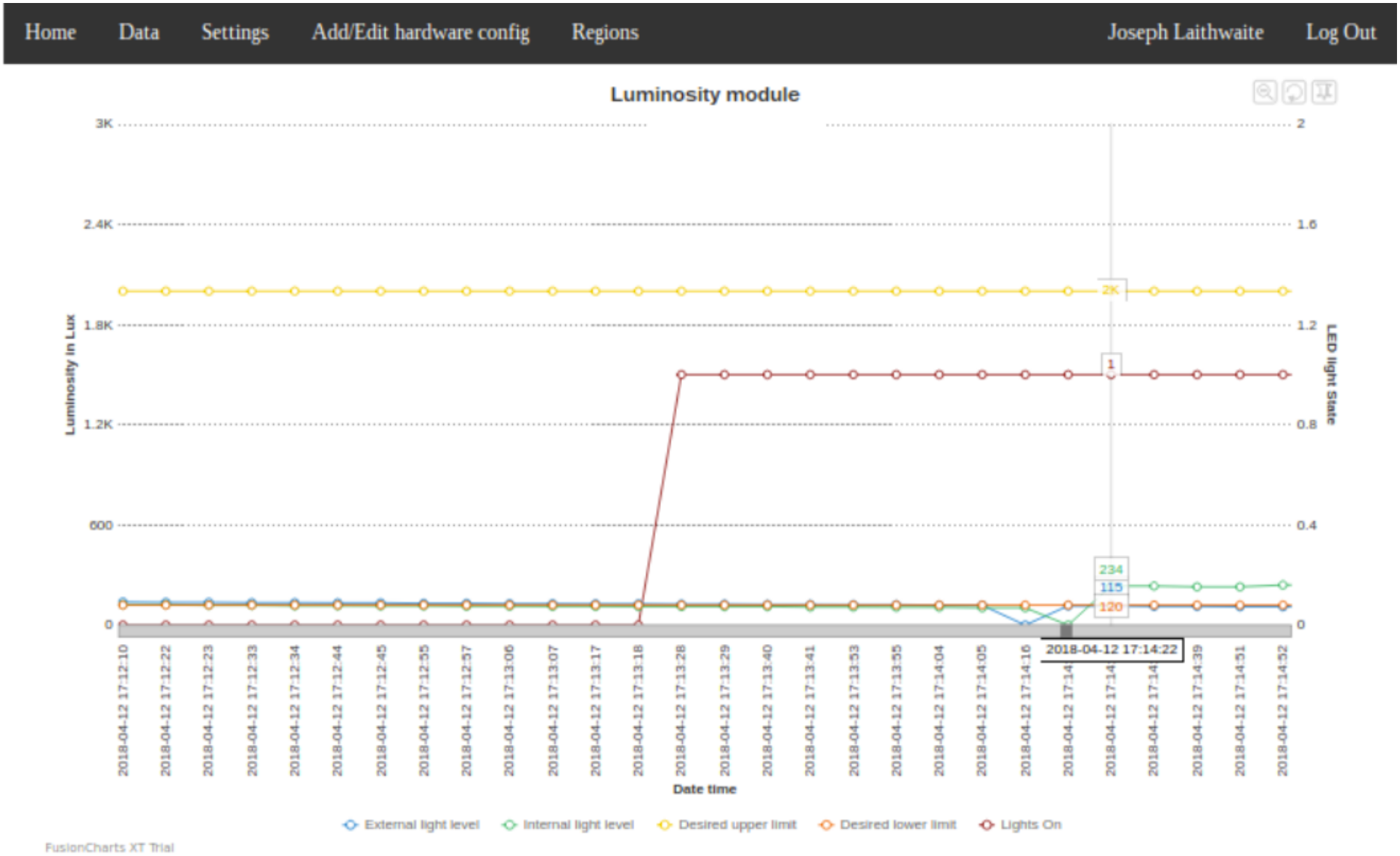

Figure 10.16 - Screenshot of luminosity module graph showing the time at which a change was noticed in light level after LEDs were turned on

Temperature Module

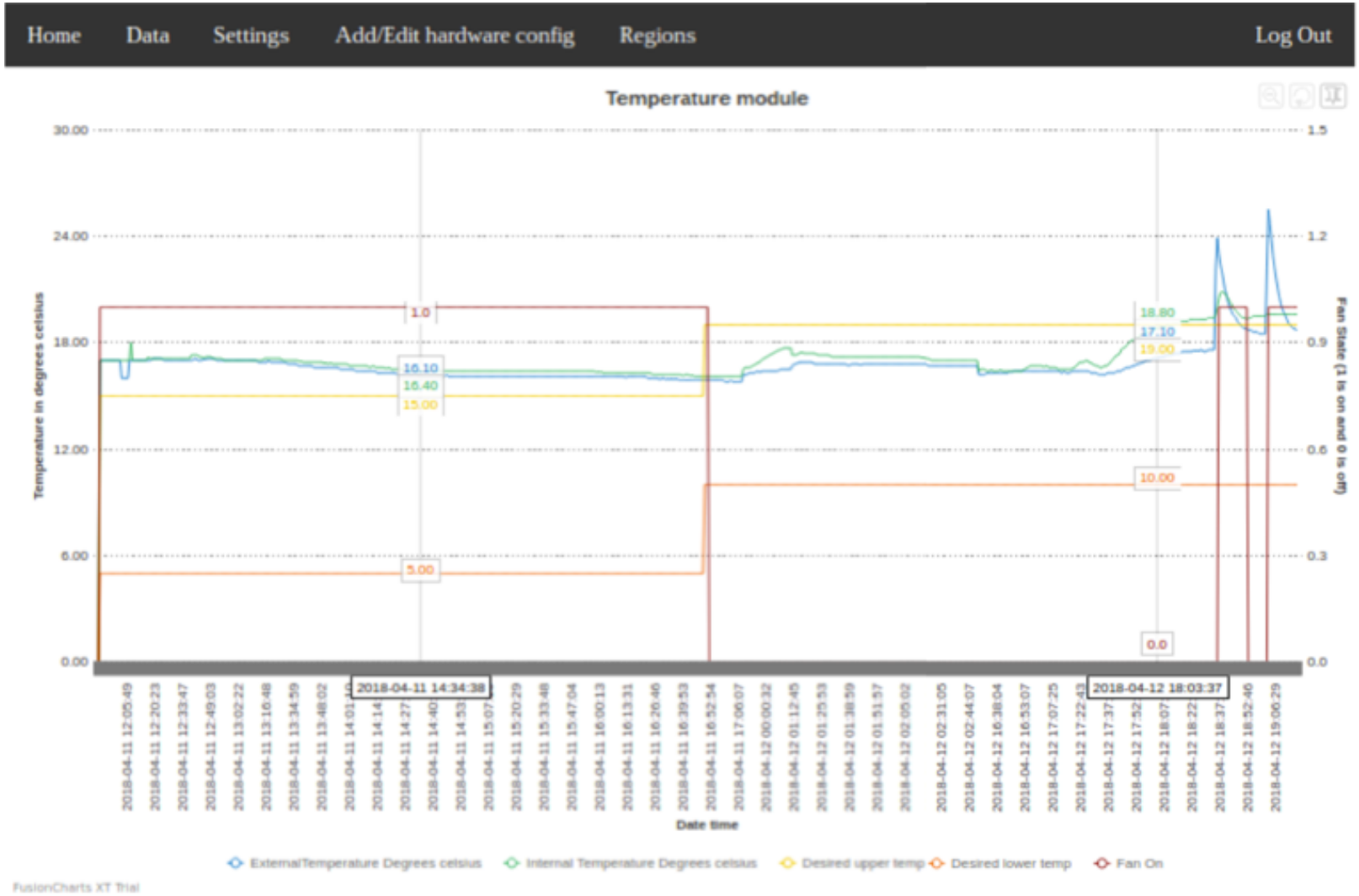

Figure 10.17 - Full graph of temperature module data

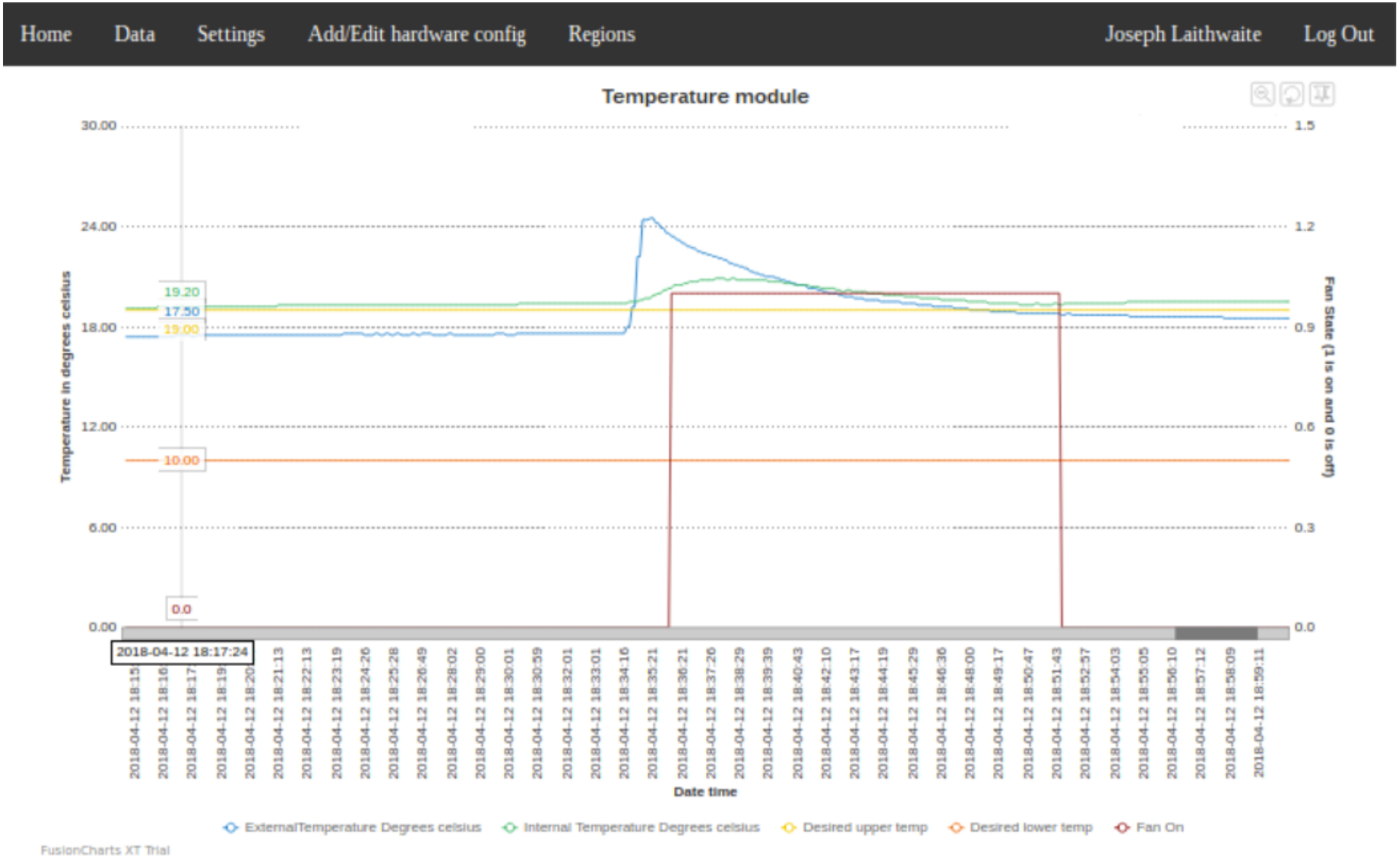

Figure 10.18 - Zoomed in graph of temperature module whilst a heat source was applied to the system

Relative Humidity

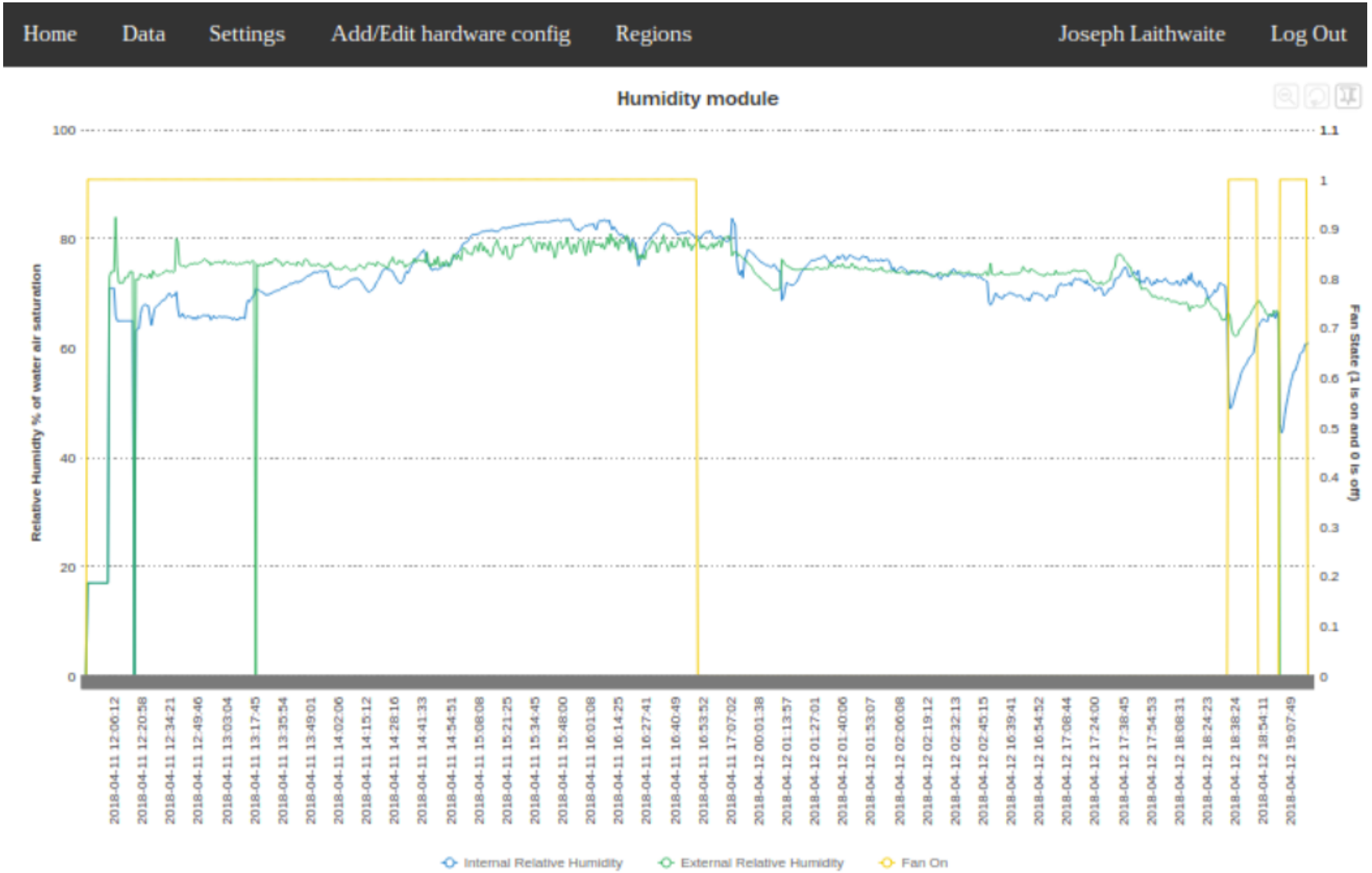

Figure 10.19 - Full graph for the humidity modules data, this has no desired levels set and the fan's state is set for the temperature module

Water Module

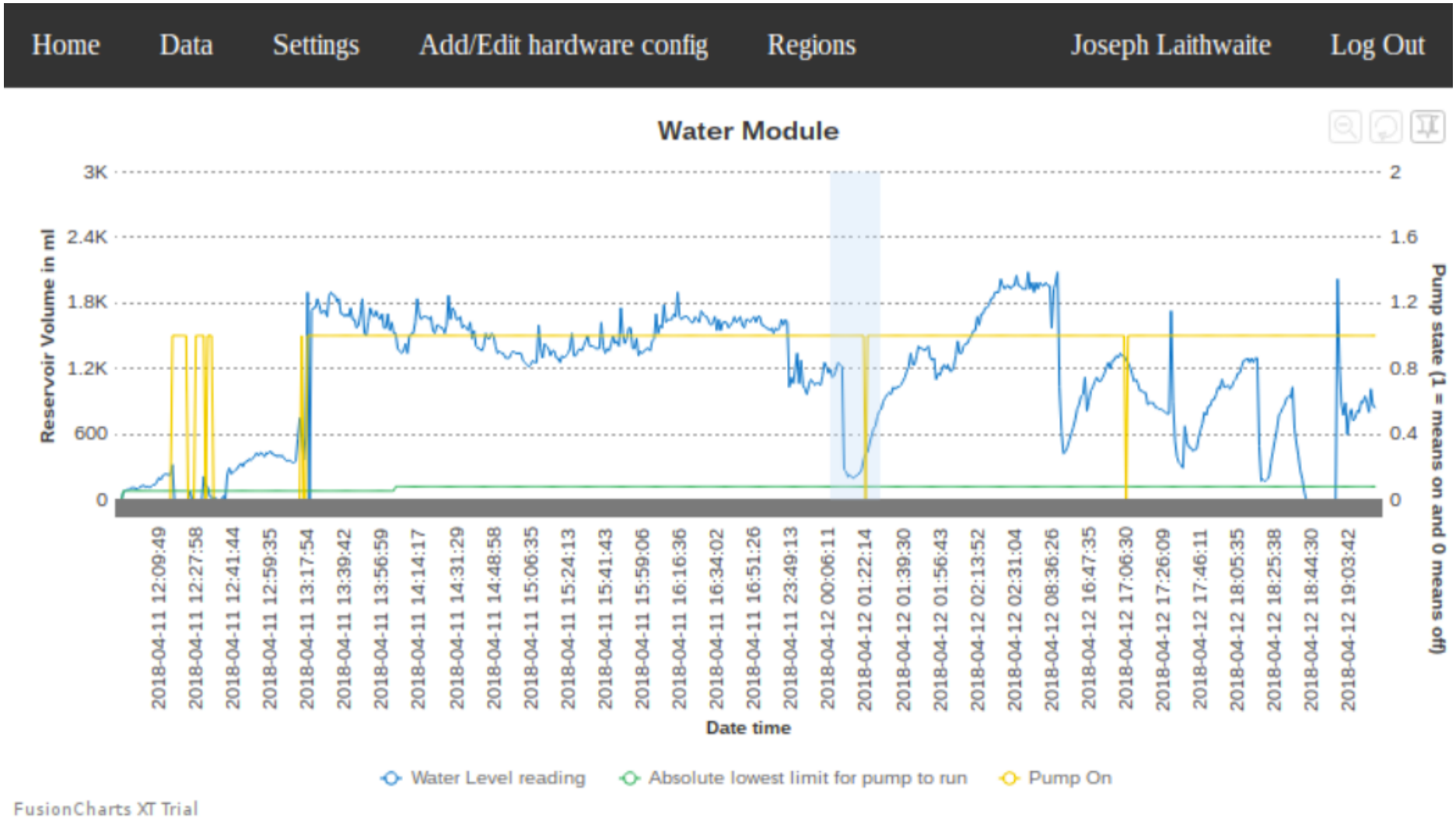

Figure 10.20 - Full graph of water module data

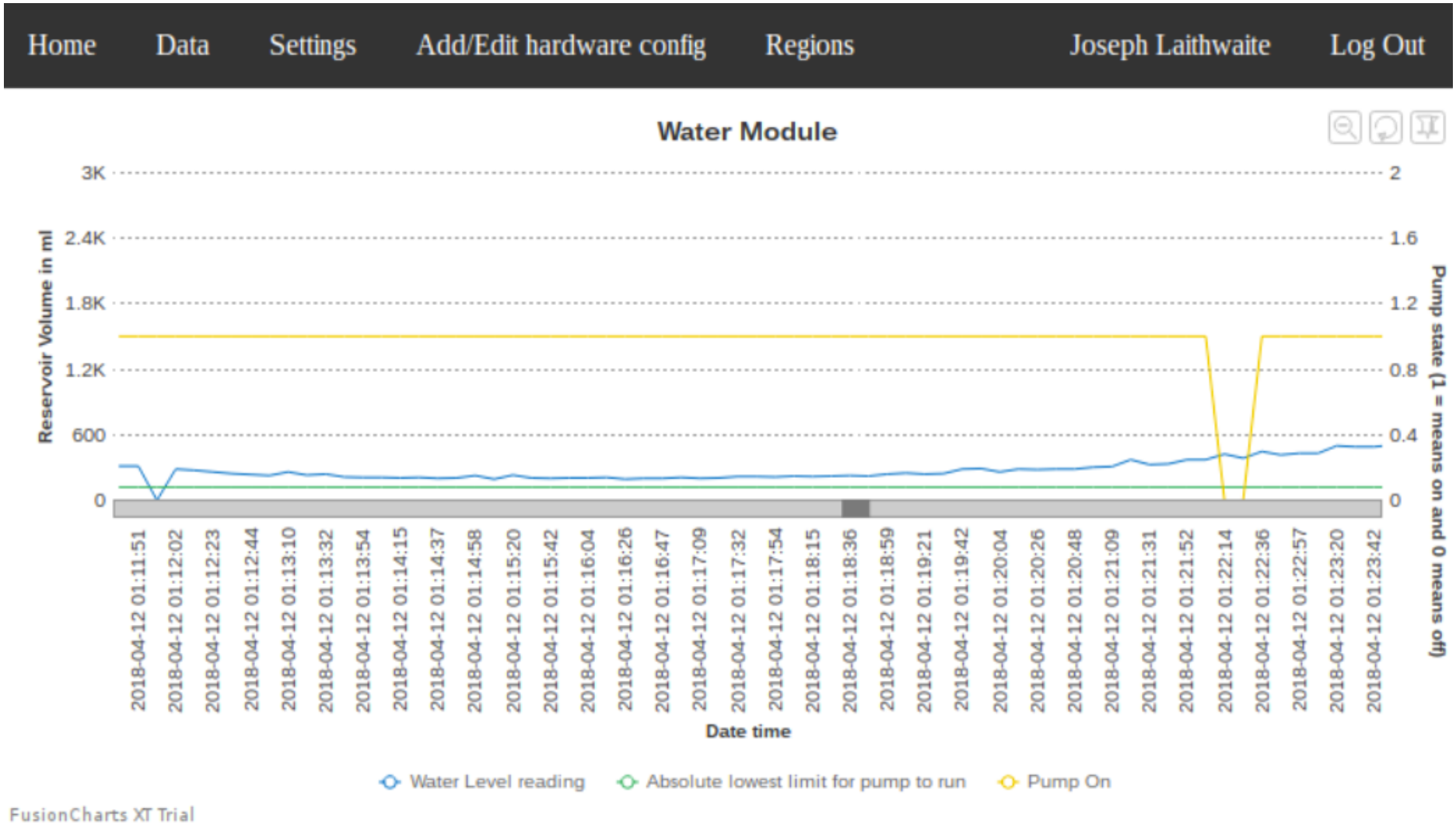

Figure 10.21 - zoomed in graph of water module showing the pumps response to the water sensors removal from water

pH module results

```
MariaDB [SmartGreenhouse_2]> SELECT AVG(SensorValue) FROM SensorReadings
WHERE InstalledSensors_SensorID = 4;
+------------------+
| AVG(SensorValue) |
+------------------+
|                0 |
+------------------+
1 row in set (0.00 sec)
```

Figure 10.22 - MySQL command line output of average sensor values from the pH meter (with SensorID 4)

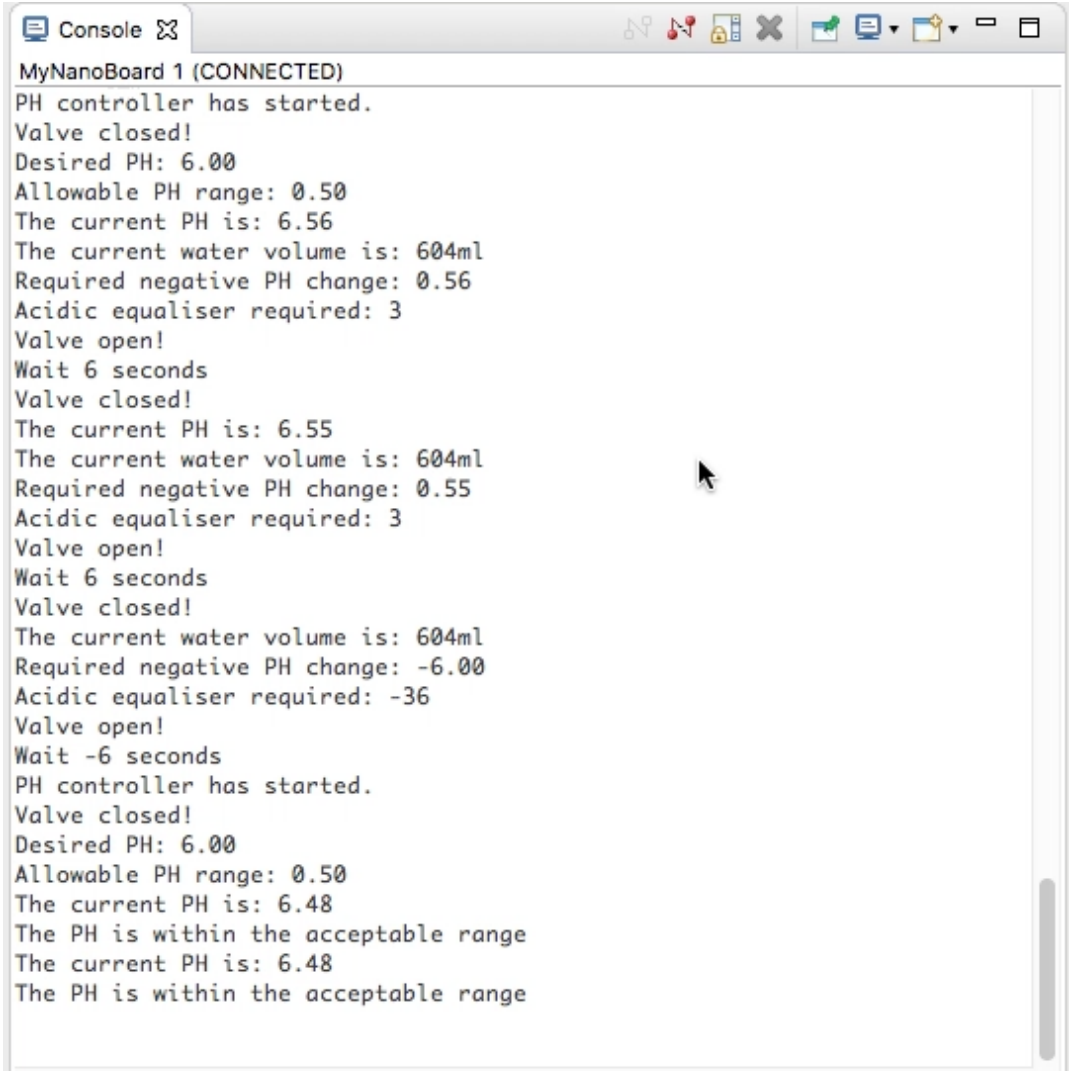

Figure 10.23 - Eclipse IDE serial connection output from Arduino running a pH controller module (basis of that used in the final system)

11 - Discussion

The results displayed in the last section show the full range of the work carried out and the scope of the work completed in this project. Here, these results will be explained and the varied importance of them discussed.

The Physical System

The physical system has been built to the specifications set out in the design section. The only small adjustments have been the use of vices, to grip and suspend the pump, pH meter and pH equalizer valve (which can be seen in figure. 10.1). Additionally, instead of sitting atop the container the grow bed is slightly submerged in the contained being held in place by the lips at each end, this allowed for a smaller system with much less risk of water accidentally leaking on electronics, whilst still allowing the flow of water through the grow bed figure 10.2. Finally, the combined sensor board comprising of the TSL561 and the DHT22 had to be slightly moved due to the temperature and humidity sensor being to close to the hole at the top meaning the internal conditions where heavily impacting the external readings. To counteract this the sensor board was moved further down the lid and placed more central as far from gaps in the greenhouse as possible. In addition to this tape was added to the other lid allowing for a slightly tighter seal this is shown in figure 1.3.

The wire management can also be seen in all three images (figure 10.1 – 10.3), this was a bit of an after thought and could be confusing when trying to follow a wire, the colour code from the circuit diagram in figure 8.1 was used as a guide, but wires still had to be taped down to keep them out the way.

The systems user interface

The system user interface isn't as complete as was hoped during the design stage, though it has come a long way in providing some of the functionality desired. Parts of the system have been successfully completed such as the user login system, though due to the complexity involved in creating a secure, validated and hashed password system the build took longer than anticipated and other functionality was missed.

The screenshots in the results section demonstrate the flow of the user interface starting with the sign up process. Figure 10.4 displays an attempt to sign up using an invalid email, one without an '@' or a '.', This is noticed by the system once submit has been pressed and the user is given a new screen with the error message informing of the invalid email figure 10.5. The last validation is the screenshot in figure 10.6 showing the error messages when no password is entered.

The next screenshot figure 10.7 shows an attempt of signing in with a valid password and email combination which brings the user to the welcome screen in figure 10.8. From here the user is able to use the menu to navigate through adding farms, regions and hardware (as shown in figures 10.9 – 10.11) as well as the option to logout, which suspends the users' session and protects their data. The most interest area of the menu however is in data tab figure 10.12. When the mouse is hovered over this tab 5 options appear for the; light module, temperature module, humidity module, water module and pH module.

The data for these modules are displayed using fusion charts plugins and allows for a great deal of user analysis to be undertaken with ease. By using a zoomable, dual y-axis graph, information about the multiple sensor readings, desired conditions and the state of the correctors can be displayed. All this data can be viewed at any time scale allowing for a smooth intuitive interface.

The user interface data output

Luminosity module output data

Once the graph loads 5 lines are displayed over the full time span the system has been running, these five lines represent; external light level (blue), Internal light level (green), desired upper limit (yellow), desired lower limit (orange) and the LEDs state (red). It's worth noting the first four lines are mapped to the left y-axis measuring luminosity and the last LED state line is mapped to the right y-axis with 1 being on and 0 being off.

The general overview of data displayed on the graph (shown in figure 10.13), exemplifies the tendency for external light level to be higher than internal light level with a difference of approximately 40 lux. This is understandable as an external sensor placed on top of the greenhouse is at the highest point with no obstacles to shade it. On the other hand, an internal sensor has glass surrounding it which may cause a slight loss of luminosity, plus obstacles such as the metal structure, sensors and LED strips which at points may all cast slight shade onto the sensor. As plants grow these two could cause shading on the sensor so a small support is required to slightly elevate the sensor. A note about all the results is that the experiment was carried out in a functioning bedroom, lights were turned on and off, curtains drawn and humans shading the sensors occurred. This means that some of the changes, which would appear very drastic in an external setting, are indeed accurate and not necessarily outliers.

Moving to a shorter time frame, by selecting the blue section in figure 10.13, causes a new zoomed in graph to be loaded figure 10.14. The cursor is set at the first point the internal light level (in green) drops bellow the desired lower limit (in orange), this is at 17:12:23 where the internal, external and lower limit light levels are 119 lux, 236 lux and 120 lux respectively. Zooming in further (to figure 10.15) it can be observed that LEDs turn on at 17:13:28, though no instant effect is observed in the light level readings. It's not until 17:14:22 (figure 10.16) when a reaction to the LEDs is noticed, at this point the internal and external light levels are 234lux and 115lux respectively. Using this data delays between events can be calculated such that:

- The delay between detection of the lower limit being breached and the response from the LEDs as

$$\text{response time - detection time} = \text{delay}_{\text{detection to response}}$$

$$17:13:28 - 17:12:23 = 65 \text{ seconds}$$

- The delay between the LEDs response and the observed increase in internal light level is

$$\text{observed effect time} - \text{response time} = \text{delay}_{\text{Response to effect}}$$

$$17:14:22 - 17:13:28 = 54 \text{ seconds}$$

- Finally, the total delay is the detection to response plus the response to effect delay

$$\text{delay}_{\text{detection to response}} + \text{delay}_{\text{Response to effect}} = \text{delay}_{\text{total}}$$

$$65 + 54 = 119 \text{ seconds} \approx 2 \text{ minutes}$$

This shows a surprisingly long delay between cause and effect though as nothing in the system is time critical, this slow response is an efficient way to minimise computational load, slightly reduce memory requirements and uses less energy than a system with a shorter delay.

Although the functionality of the lighting module works just as desired, turning on and off to maintain a more optimum growing environment, research shows the light output of the LED strips just isn't sufficient for most plants to thrive. The minimum desired luminosity for most plants is stated to be above 1000lux, as the internal luminosity sensor detect just 150lux to 200lux the plants require at least 5 times more light. Another aspect of light to consider is using PAR (photosynthetically active radiation) sensors, this measures the light a plant can absorb rather than the light humans see so would be much more appropriate. However, these sensors are very expensive and would like not be able to be installed in the same number. One option would be attaching it to a small track allowing the sensor to move taking readings from a much larger area and making it much more cost effective.

Temperature module output data

The temperature module graph follows a similar setup as the luminosity module graph. At full view (figure 10.17), the trend of temperature readings is such that internal temperature is often slightly above external temperature. During cooling (when the fan is on) readings are very close with roughly just 0.3 °C between them, once the fan is off and the system begins to warm, the range grows larger to between 0.5°C and 1.5°C.

Desired conditions can be set up to have different conditions at different times which can be seen by following the upper limit (yellow) and lower limit (orange) lines, a jump from 15°C and 5°C to 19°C and 10 °C respectively is noticed. This has an instant effect on the fan which is continuously on, trying to cool the air bellow 15°C, until the lower limit changes. At this point the temperatures begin to rise.

Towards the right hand side of the graph a hair drier was used to simulate a temperature increase, as can be seen by two spikes and subsequent falls as the fan was very effective in reducing temperature. Expanding the graph at the first spike a graph in figure 10.18 is created. This shows a fairly long delay period between internal temperature reaching 19°C and the fan turning on, but once on the rapidly increasing temperatures are quickly slowed. The external temperature spike is much more sharp than internal spike for two possible reasons, firstly the hair drier was outside trying to push air in gaps though largely missed and landed on the external sensor. Secondly, much as hot air rises a large portion of hot air rushes up to where the external sensor is, whereas the internal sensor, situated on the bottom, doesn't measure as much of the rising hot air. This observation suggests the external sensor may need to be physically separate to the greenhouse, whether this is fully separated by utilising local weather stations' data, or if it's just attaching a sensor board further from the greenhouse.

Relative humidity output data

The humidity module is a little different to the others as the fan was set up to correct temperature change and not humidity change, the graph therefore only displays three lines for; fans state (yellow), internal humidity (blue) and external humidity (green) figure 10.19.

As explained in the theory section, humidity reading taken is relative to temperature, as various air temperatures can hold different amounts of water. It's therefore quite interesting to note the first section of the graph, with the fan on and undergoing cooling, caused a substantial increase in relative humidity of around 15%, this as colder air holds less water meaning water already in the air takes up a larger portion relative to this lower saturation point. The opposite effect can be

observed once the fan is switched off and the greenhouse gradually heats, this effect is exacerbated when the hair drier is used showing almost instantaneous drops and recovery once the fan is on.

Though not the best method possible to control humidity, as consistent temperature is more beneficial to growth as well as the energy required to heat an area, this is an interesting tool when trying to counteract constant changes and with deeper investigations could be useful in developing a more complex algorithm to decide what correctors should be used when.

Water module output data

The water module is the most basic module and serves simply as a reminder to fill up the reservoir and a safety measure so the pump doesn't run dry (without water going in). It's for this reason the pump is on for the majority of running time. Off times may look fairly random in the full time view (figure 10.20), but when on a smaller time scale, such as in figure 10.21, the turn offs become clear. An instant drop in the water readings to zero can be identified with the pump turning off shortly after. This sudden zero water reading is due to the sensor being removed from water to test the systems reaction.

This graph does raise an issue that the correction may turn off or on with such a delay that the sensor reading has already raised back into the ideal zone. As well as the issue of a delay, even if a reaction is instantaneous, this may be too quick if the lights are turning on because the light level dropped 1 lux under, chances are within the next few seconds it'll bounce back above the limit. It's for this reason a more complex method of turning on correctors should be used, one which takes averages over a period of time to ensure a passing object doesn't cause corrections. This could also aid in reducing memory usage as if an average is saved every couple of minutes instead of an exact read every 30 seconds, less data is stored without losing any important data.

pH Module data

Unfortunately, graphed results for the pH module can't be obtained due to a fault with the pH meter causing a zero output to be displayed at all times. Extensive troubleshooting has been attempted using various different connection wires, power sources and Arduino input pins but all with no success. This is demonstrated with the simple sql query in figure 10.22 retrieving the average sensor reading from the pH meter. All zero.

Results were however produced earlier in the development process (which are shown in figure 10.23) using Eclipse IDE's serial console interface. The programs flow is it; gets current pH reading, checks this against desired pH and it's range, then if it's too alkali, takes the water level, calculates volume of equalizer required (as water volume * pH change *0.1) then opens the valve for two times the number of milliliters (as it's flow rate is 1ml/2s). This process worked effectively, as shown in the output, though it can take multiple attempts to get the pH correct. The approach of little and often is preferred however as with a constant flow of water in the system, the possibility of localised highly acidic water and localized alkali water is foreseeable.

12 - Conclusions

To conclude, a prototype smart hydroponic greenhouse has been developed with the ability to monitor and correct light, temperature, humidity, water flow and somewhat pH. The user interface developed along side it also has reasonable functionality allowing a user to securely log on, begin to add hardware and delve deep into the historic and current data of each module. This allowed for 5 out of the 6 deliverables set out in the early stages of the project (and in section 6 of this report) to be achieved.

This success acts as a proof of concept, demonstrating the software behind this system is working well and leads to the possibility of scaling up its functionality for small urban farms. A must to achieve this would be understanding the effects of various conditions on a crops growth. By carrying out a multiple plant growth tests with a variety of sensor the most cost effective systems would be developed. Starting small by improving the temperature module would be a good place to start. Currently the module has a fan to correct temperature when it's too warm, but by adding a heating element to this the system could also correct for lower than optimum temperatures. Other hardware to consider may include CO_2 sensors and injectors, a sprinkle/ misting valve for humidity and a EC (electrical conductivity) sensor to measure nutrients with corresponding nutrient equalisers similar to the pH equalisers.

The luminosity module is another piece of hardware which may require upgrading, the output of the LED strips provide s maximum of 200 lumens to the greenhouse base, this being 5 times less than the desired amount for optimum photosynthesis means higher wattage lights are needed. The other issue with the luminosity module is the light sensor itself. Luminosity is a measure of visible light to the human eye, a third of this spectrum (green) doesn't have any part of photosynthesis so using human light levels is pointless. By using a PAR sensor, the light level plant can actually absorb is picked up.

The integration of local weather station data instead of (or in addition to) the external sensors may facilitate a much more accurate representation of the controlled and the external conditions. By separating the internal and external readings, correction actions inside the greenhouse should not have any affect on the external readings. Also this may reduce the cost of a system if external sensors aren't required. Finally the possibility of future developments utilising forecasted data to primitively set corrections instead of reactively set corrections. This may allow for greater energy efficiency as there's no point turning a heater on in a greenhouse if ou know in an hour the suns going to be heating it up.

Bibliography

[1] J. Laithwaite, “Preliminary report for “Real-time crop/plant monitoring and maintenance system”,” N/A, Liverpool, 2017.

[2] dictionary.com, “Dictionary.com "hydroponic," in Dictionary.com Unabridged,” Source location: Random House, Inc., [Online]. Available: http://www.dictionary.com/browse/hydroponic. [Accessed 11 10 2017].

[3] [Online].

[4] United Nations, “World Population Prospects: The 2017 Revision, Key Findings and Advance Tables,” 2017.

[5] Food and Agriculture Organization, “Agricultural land (% of land area),” 2017. [Online]. Available: https://data.worldbank.org/indicator/AG.LND.AGRI.ZS. [Accessed 11 04 2018].

[6] M. Roser and H. Ritchie, “Land Cover,” 2017. [Online]. Available: https://ourworldindata.org/land-cover. [Accessed 11 04 2018].

[7] Food and Agriculture Organization, “Forest area (% of land area),” 2016. [Online]. Available: https://data.worldbank.org/indicator/AG.LND.FRST.ZS. [Accessed 11 04 2018].

[8] L. Williams, “The water watchers Tech-assisted irrigation,” *Engineering & Technology,* vol. 11, no. 12, pp. 36-39, 2017.

[9] T. D. R. d. Nooy, ““Agriculture Water Use and River Basin Conservation” Adapted from Water Use for Agriculture in Priority River Basins, a study prepared for WWF,” WWF–World Wide Fund for Nature, 2003.

[10] V. J. Charles and P. B. McIntyre , “Global threats to human water security and river biodiversity,” *Nature,* vol. 467, no. 467, pp. 555-561 , 09 2010.

[11] Bank, The World, “Agriculture, value added (current US$),” 2017. [Online]. Available: https://data.worldbank.org/indicator/NV.AGR.TOTL.CD . [Accessed 11 04 2018].

[12] A. Rodriguez, “How Important is the Height of a Greenhouse,” [Online]. Available: http://homeguides.sfgate.com/important-height-greenhouse-47826.html. [Accessed 11 04 18].

[13] O. Pons, A. Nadal, E. Sanyé-Mengual, P. Llorach-Massana, E. Cuerva, D. Sanjuan-Delmàs, P. Muñoz, J. Oliver-Solà, C. Planas and M. R. Rovira, “Roofs of the Future: Rooftop Greenhouses to Improve Buildings Metabolism,” *Procedia Engineering,* vol. 123, pp. 441-448, 2015.

[14] P. Bradley and C. Marulanda, “SIMPLIFIED HYDROPONICS TO REDUCE GLOBAL HUNGER,” no. 554_31, p. 289_296, 2001.

[15] R. Northfield, “Take it to the extreme Farming in unlikely places,” *Engineering & Technology,* vol. 11, no. 12, pp. 40-43, 2017.

[16] A. A. Ajmi, A. A. Salih, I. Kadim and Y. Othman, “yield and water use efficiency of barley fodder produced under hydroponic system in GCC countries using tertiary treated sewage effluents,” *Journal of Phytology,* vol. 1, no. 5, p. 342–348, 2009.

[17] D. S. Freedman, L. K. Khan, W. H. Dietz, S. R. Srinivasan and G. S. Berenson, “Relationship of Childhood Obesity to Coronary Heart Disease Risk Factors in Adulthood: The Bogalusa Heart Study,” *the American Academy of Pediatrics,* vol. 108, no. 3, pp. 712-718, 2001.

[18] D. M. Allcock, M. J. Gardner and J. R. Sowers, “Relation between Childhood Obesity and Adult Cardiovascular Risk,” *International Journal of Pediatric Endocrinology,* 2009.

[19] S. E. Lineberger and J. M. Zajicek, “School Gardens: Can a Hands-on Teaching Tool Affect Students’ Attitudes and Behaviors Regarding Fruit and Vegetables?,” *HortTechnology,* vol. 10, no. 3, pp. 593-597, 2000.

[20] J. L. Morris, M. Briggs and S. Zidenberg-Cherr, “Development and Evaluation of a Garden-Enhanced Nutrition Education Curriculum for Elementary Schoolchildren,” *The Journal of child nutrition and management,* vol. 26, no. 1, 2002.

[21] C. Woodford, “pH meters,” 05 01 2018. [Online]. Available: http://www.explainthatstuff.com/how-ph-meters-work.html. [Accessed 11 04 2018].

[22] Science Buddies, “Acids, Bases, & the pH Scale,” [Online]. Available: https://www.sciencebuddies.org/science-fair-projects/references/acids-bases-the-ph-scale. [Accessed 11 04 18].

[23] Gómez-Merino, L. I. Trejo-Téllez and F. C., “Nutrient Solutions for Hydroponic Systems,” in *Hydroponics - A Standard Methodology for Plant Biological Researches, Dr. Toshiki Asao (Ed.)*, D. T. Asao, Ed., 2012, pp. 1-22.

[24] L. Morgan, “Perfecting the pH of Your Hydroponic Nutrient Solution,” 01 12 2016. [Online]. Available: https://www.maximumyield.com/perfecting-ph/2/1212. [Accessed 11 04 2018].

[25] DFRotbot, “PH meter(SKU: SEN0161),” 2017. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/SEN0161_SEN0169_Web.pdf. [Accessed 11 04 2018].

[26] M. Brechner and A. Both, “Hydroponic Lettuce Handbook”.

[27] S. Markings, “The Effect of Temperature on the Rate of Photosynthesis,” 09 03 2018. [Online]. Available: https://sciencing.com/effect-temperature-rate-photosynthesis-19595.html. [Accessed 11 04 2018].

[28] D. Nedelkovski, “DHT11 & DHT22 Sensors Temperature and Humidity Tutorial using Arduino,” 03 2016. [Online]. Available: https://howtomechatronics.com/tutorials/arduino/dht11-dht22-sensors-temperature-and-humidity-tutorial-using-arduino/. [Accessed 11 04 2018].

[29] T. Liu, “DHT22 datasheet,” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf. [Accessed 11 04 2018].

[30] physics.nist.gov, “International System of Units (SI),” [Online]. Available: https://physics.nist.gov/cuu/Units/units.html . [Accessed 11 04 2018].

[31] A. Urbonavičiu¯tė, P. Pinho, G. Samuolienė, P. Duchovskis, P. Vitta, A. Stonkus, G. Tamulaitis, A. Žukauskas and L. Halonen, “Effect of short-wavelength light on lettuce growth and nutritional quality.,” *Sodininkystė ir Daržininkystė,* vol. 26, no. 1, pp. 157-165, 2007.

[32] l. ada, “TSL2561 Luminosity Sensor,” 27 01 2018. [Online]. Available: https://cdn-learn.adafruit.com/downloads/pdf/tsl2561.pdf. [Accessed 11 04 2018].

[33] AMS, “TSL2560, TSL2561 Datasheet,” 2018. [Online]. Available: http://ams.com/eng/Products/Light-Sensors/Ambient-Light-Sensors/TSL2561/TSL2560-TSL2561-Datasheet. [Accessed 11 04 2018].

[34] Eclipse, “Eclipse IDEE Downloads,” [Online]. Available: https://www.eclipse.org/downloads/eclipse-packages/.

[35] Sloeber, “install Advice,” [Online]. Available: http://eclipse.baeyens.it/installAdvice.shtml. [Accessed 11 04 2018].

[36] Freematics, “Freematics Arduino Builder,” [Online]. Available: https://freematics.com/pages/software/arduino-builder/.

[37] php, “php password hash function,” [Online]. Available: http://php.net/manual/en/function.password-hash.php. [Accessed 11 04 2018].

[38] F. B. Salisbury, Plant physiology, Belmont, Calif: Wadsworth Pub. Co., 1978.

[39] T. T. Kozlowski, P. J. Kramer and S. G. Pallardy, The Physiological Ecology of Woody Plants, San Diego: Academic Press inc., 1991.

13 - Appendices

A) Water Level calibration data

Water Depth mm	5V Raw data	3.3V Raw data
0	0	0
5	525	275
10	560	310
15	585	320
20	620	340
25	630	345
30	635	355
35	645	370
40	650	375

5V without 0	
Slope	0.257268722
Y-intercept	-133.469163

3.3V without 0	
Slope	0.356340289
Y-intercept	-97.31942215

Raw Data	Approximate depth with 3.3V no 0 linear
0	-97.319
275	0.674
310	13.146
320	16.709
340	23.836
345	25.618
355	29.181
370	34.526
375	36.308

Approximate depths with various trendlines, using 5V				
5V Raw data	Linear 0.2573x - 133.47	Exponential 0.0016e^0.0155x	Power $9E\text{-}25x^{9.1084}$	Polynomial y = 0.0022x^2 - 2.3269x + 624.3
525	1.597	5.47	5.38	9.0525
560	10.601	9.41	9.68	11.156
585	17.033	13.87	14.41	15.9585
620	26.037	23.86	24.46	27.302
630	28.610	27.86	28.30	31.533
635	29.896	30.11	30.41	33.8135
645	32.469	35.15	35.06	38.7045
650	33.756	37.99	37.62	41.315
r^2	0.89432	0.983	0.98323	0.96299

B) System type cost analysis

Component info		This system		Basic Eductaional/home system		10m^2 Commercial system	
Component name	Component Price	Quantity	System Price	Quantity2	System Price2	Quantity3	System Price3
Light Sensor	£7.40	2	£14.80	2	£14.80	8	£59.20
Humidty & Temp	£4.00	2	£8.00	2	£8.00	8	£32.00
PH	£43.03	1	£43.03	0	£-	1	£43.03
Water Level	£1.98	1	£1.98	1	£1.98	1	£1.98
Pump	£25.83	1	£25.83	0	£-	2	£51.66
LED strip (per10cm)	£0.23	8	£1.84	8	£1.84	10000	£2,300.00
DC mixer motor	£2.99	1	£2.99	0	£-	4	£11.96
Solenoid valve	£7.22	1	£7.22	0	£-	3	£21.66
Fan	£3.99	1	£3.99	1	£3.99	8	£31.92
Transistor	£0.64	6	£3.83	4	£2.55	300	£191.40
relay	£1.02	1	£1.02	0	£-	1	£1.02
							£-
RaspberryPi	£29.99	1	£29.99	0	£-	1	£29.99
Arduino Nano 2.0	£17.31	1	£17.31	1	£17.31	4	£69.24
Total			£161.83		£50.47		£2,845.06

Plants in Configuration			8		8		10000
Price per plant			£20.23		£6.31		£0.28

C) LDR & TSL2561 comparison data

LDR	TSL2561
2	0
2	0
2	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
1	0
6	7
12	12
12	12
26	14
24	14
12	14
12	14
9	16
11	16
11	17
12	17
30	19
10	19
12	19
30	21
27	21
30	21
30	21
30	21
35	21
27	21
11	21
25	23
27	23
28	23
29	23
29	23
29	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
31	23
31	23
33	23
29	23
29	23
30	23
30	23
30	23
30	23
29	23
28	23
27	23
28	23
29	23
29	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
30	23
11	23
13	23
29	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
30	26
31	26
31	26
31	26
31	26
31	26
31	26
30	26
30	26
30	26
30	26
30	26
30	26
31	26
39	26
29	26
30	26
30	26
30	26
30	26
30	26
30	28
30	28
30	28
32	28
31	28
30	28
30	28
31	28
32	28
32	28
31	28
32	28
12	28
33	30
31	30
33	30
10	30
11	30
11	30
12	30
35	33
12	33
12	33
46	35
15	35
38	38
36	38
35	38
11	38
15	38
42	41
44	44
16	44
45	47
58	47
17	47
46	50

47	50
17	53
51	56
20	56
55	63
57	63
73	63
36	63
60	67
61	71
19	71
21	71
62	75
64	75
64	75
23	75
21	75
64	79
64	79
66	79
67	79
40	79
70	83
24	83
44	83
41	83
24	83
70	87
70	87
24	87
75	91
24	91
72	96
74	96
76	96
77	96
15564	100
79	105
81	105
80	105
94	105
45	105

45	105
32	105
82	110
49	110
84	115
90	115
42	115
87	120
48	120
93	125
104	125
124	125
30	125
34	125
24	125
93	130
95	130
104	130
104	130
104	130
105	130
106	130
91	130
48	130
96	136
99	136
101	136
102	136
103	136
104	136
105	136
104	136
103	136
104	136
107	136
107	136
99	136
45	136
102	141
102	141
102	141
101	141

99	141
108	147
109	147
103	147
113	153
107	153
105	153
102	153
115	159
113	159
109	159
108	159
35	159
116	165
116	165
106	165
108	165
116	171
118	171
118	171
119	177
121	177
122	177
119	177
119	177
121	177
117	177
128	177
124	177
112	177
56	177
122	184
121	184
120	184
117	184
132	184
151	190
151	190
153	190
126	203
163	203
154	203

161	203
66	203
133	217
136	217
168	225
168	232
149	232
150	232
140	239
175	239
150	247
174	247
52	247
179	255
183	255
219	255
87	255
180	263
168	263
157	271
184	279
166	279
200	279
176	287
231	287
233	287
188	287
167	295
191	295
202	304
175	313
226	313
195	313
198	321
248	321
184	330
258	330
190	330
189	339
216	339
167	339
191	349

195	349
209	349
205	349
259	349
97	349
199	358
211	358
223	358
203	368
213	368
116	368
209	377
215	377
217	377
199	377
216	387
231	387
274	397
239	407
15635	418
292	418
230	439
261	449
292	449
246	460
293	460
231	471
159	483
270	494
208	494
313	517
284	529
254	529
341	529
271	541
302	578
267	578
330	578
153	578
155	591
41	591
146	591

293	604
148	604
323	617
299	617
152	630
157	630
318	643
308	643
290	643
144	643
147	643
357	657
383	657
163	670
163	670
165	670
164	670
163	670
164	670
164	670
164	670
166	670
165	670
323	684
162	684
164	684
164	684
161	684
162	684
162	684
148	684
168	684
165	684
167	684
166	684
168	684
167	684
166	684
167	684
167	684
165	684
167	684

164	684
166	684
164	684
167	684
165	684
166	684
164	684
166	684
164	684
163	684
165	684
164	684
162	684
162	684
165	684
165	684
162	684
165	684
164	684
161	684
165	684
164	684
162	684
166	684
164	684
162	684
158	684
333	698
164	698
168	698
169	698
169	698
168	698
171	698
167	698
164	698
165	698
168	698
164	698
168	698
163	698
165	698

161	698
166	698
166	698
166	698
161	698
165	698
165	698
165	698
159	698
158	698
161	698
159	698
160	698
160	698
163	698
165	698
335	712
318	712
164	712
157	712
160	712
161	712
157	712
158	712
162	712
165	712
169	712
166	712
163	712
161	712
166	712
167	712
163	712
167	712
162	712
167	712
161	712
160	712
160	712
163	712
159	712
163	712

163	712
159	712
159	712
169	712
164	712
160	712
160	712
338	727
157	727
162	727
162	727
169	727
169	727
164	727
165	727
179	727
160	727
164	727
165	727
161	727
159	727
162	727
164	727
165	727
163	727
163	727
164	727
164	727
163	727
163	727
158	727
413	741
156	741
159	741
163	741
167	741
161	741
165	741
159	741
159	741
164	741
160	741

164	741
160	741
164	741
159	741
159	741
164	741
158	741
164	741
158	741
157	741
156	741
157	741
160	741
156	741
156	741
157	741
161	741
157	741
158	741
160	741
155	756
161	756
159	756
158	756
155	756
169	756
168	756
162	756
167	756
165	756
170	756
164	756
163	756
164	756
158	756
159	756
160	756
159	756
157	756
160	756
157	756
160	756

158	756
161	756
158	756
160	756
157	756
156	756
390	771
153	771
158	771
155	771
156	771
156	771
162	771
159	771
162	771
159	771
162	771
159	771
158	771
161	771
157	771
162	771
159	771
161	771
161	771
161	771
158	771
157	771
157	771
156	771
157	771
160	771
156	771
158	771
157	771
157	771
156	771
157	771
158	771
155	786
157	786
155	786

155	786
163	786
161	786
164	786
193	786
160	786
159	786
161	786
161	786
159	786
162	786
159	786
159	786
159	786
158	786
159	786
158	786
159	786
158	786
158	786
157	786
165	801
161	801
161	801
163	801
163	801
159	801
163	801
159	801
159	801
159	801
422	801
171	817
259	817
359	832
417	864
340	864
193	864
256	864
335	880
387	880
347	897

418	913
199	913
201	913
213	913
349	930
352	930
206	930
480	947
202	947
484	964
246	964
406	981
447	981
376	981
240	981
233	981
242	1017
447	1035
245	1035
234	1035
450	1053
236	1053
469	1071
399	1090
386	1090
462	1090
260	1090
474	1108
466	1108
247	1108
269	1108
449	1127
482	1127
474	1127
539	1146
541	1146
15879	1150
319	1166
251	1166
245	1166
265	1166
503	1185

255	1185
476	1185
479	1205
494	1205
499	1205
448	1205
449	1205
312	1205
283	1205
293	1205
422	1205
492	1225
496	1225
568	1225
452	1225
457	1225
458	1225
455	1225
458	1245
505	1245
265	1245
298	1245
467	1266
273	1266
564	1266
461	1266
505	1286
551	1286
533	1307
500	1307
496	1307
335	1307
286	1307
438	1328
477	1328
593	1328
537	1349
513	1349
496	1349
507	1349
581	1349
436	1349

543	1371
482	1371
573	1392
513	1392
500	1392
354	1414
337	1414
518	1437
416	1437
300	1437
633	1459
355	1482
495	1504
379	1504
612	1551
471	1551
620	1598
589	1598
454	1598
419	1598
650	1646
592	1646
365	1646
640	1671
471	1671
654	1695
492	1720
540	1745
649	1745
709	1745
685	1771
398	1796
583	1822
617	1822
564	1848
707	1875
682	1875
559	1875
619	1875
652	1902
541	1902
624	1902

490	1928
531	1956
406	1956
596	1956
539	1956
741	2011
521	2011
225	2011
739	2039
724	2039
609	2039
604	2039
608	2039
605	2067
555	2067
646	2067
681	2067
629	2067
604	2067
416	2095
836	2095
774	2153
782	2153
814	2212
567	2212
563	2212
630	2212
795	2242
736	2242
612	2242
851	2242
750	2272
591	2333
622	2333
924	2364
684	2395
890	2395
845	2426
796	2426
779	2426
881	2522
1025	2654

860	2654
853	2654
1053	2688
798	2688
1042	2722
764	2757
860	2757
845	2757
1059	2791
864	2791
897	2862
1061	2897
989	2897
987	2897
1093	3043
981	3043
1116	3080
1073	3117
850	3155
1123	3310
1196	3349
865	3389
1016	3429
1136	3469
1140	3469
955	3635
1144	3635
1227	3677
1235	3719
1179	3805
1330	3893
1309	4027
1378	4118
1087	4164
1420	4258
1452	4401
1609	4548
1467	4597
1618	4597
1230	4647
1481	4647
1428	4698

14028	4895
14905	4895
1580	4904
1330	5116
1403	5225
1466	5280
1620	5336
1486	5392
1696	5392
1675	5448
20376	5525
1574	5620
1660	5620
1872	5737
1639	5855
1840	6037
1862	6037
1949	6160
1746	6285
2020	6285
2139	6413
2270	6607
1998	6607
2033	6673
2344	6673
1885	6740
2459	7010
1989	7432
2082	7432
2185	7432
2753	8102
2529	8737
2716	9239
3129	9498
2801	9853
2974	10692
2848	10692
14230	11241
3740	11902
3318	12221
3821	14294
4396	15570

21657	15750
4324	15972
11675	16141
13326	16141
4395	16245
4956	17087
5054	17376
4826	17818
11430	19158
5359	19850
12322	20597
9832	20855
10164	20950
6159	21375
8623	22450
6206	22450
12671	22819
15830	23193
10701	23343
6822	25560
6491	25767
7283	26398
10758	26545
8369	26611
19324	28834
7568	29065
8086	29771
9267	31181
9379	31181

Arduino Code

Sloeber.ino.cpp

```
#ifdef __IN_ECLIPSE__
//This is a automatic generated file
//Please do not modify this file
//If you touch this file your change will be overwritten during the next build
//This file has been generated on 2018-04-10 19:04:31
#include "Arduino.h"
#include "Arduino.h"
#include "SensorLibraries/Sensor.h"
#include "SensorLibraries/DHT22Sensor.h"
#include "SensorLibraries/I2cSensor.h"
#include "SensorLibraries/RelayPoweredAnalogSensor.h"
#include "SensorLibraries/StandardAnalogSensor.h"
#include "CorrectorLibraries/StandardCorrector.h"
#include "CorrectorLibraries/PWMCorrector.h"
String waitForInput();
void decodeAndExecuteSensorRead(String instruction);
void decodeAndExecuteCorrectorAction(String instruction);
int freeMemory() ;
void decodeSensorOrCorrector(String instruction);
void setup() ;
void loop() ;

#include "Smart_Farm.ino"
#endif
```

Smart_Farm.ino

```
#include "Arduino.h"
#include "SensorLibraries/Sensor.h"
#include "SensorLibraries/DHT22Sensor.h"
#include "SensorLibraries/I2cSensor.h"
#include "SensorLibraries/RelayPoweredAnalogSensor.h"
#include "SensorLibraries/StandardAnalogSensor.h"
#include "CorrectorLibraries/StandardCorrector.h"
#include "CorrectorLibraries/PWMCorrector.h"
String waitForInput(){
while(!Serial.available());
return Serial.readString();
}
void decodeAndExecuteSensorRead(String instruction){
        //Serial.println("Sensor read about to occur");
        //String sensorTypeID = instruction.substring(1,4);
        switch((instruction.substring(1,4)).toInt()){                //switch between sensor types
                case(1): //PH
                        {
                                uint8_t sensorPin = uint8_t(instruction.substring(4,7).toInt()); //sensor
pin
                                uint8_t powerPin = uint8_t(instruction.substring(7,10).toInt()); //power
pin
                                StandardAnalogSensor phSensor =
StandardAnalogSensor(sensorPin, powerPin);
```

```
                                        //Serial.print("PH: ");
                                        Serial.println(phSensor.getSensorValue());
//                                      delete phSensor;
                                break;
                                }
                case(2): //WaterLevel
                                        {
                                        RelayPoweredAnalogSensor waterLevelSensor =
RelayPoweredAnalogSensor(uint8_t(instruction.substring(4,7).toInt()),
uint8_t(instruction.substring(7,10).toInt()), uint8_t(instruction.substring(10,13).toInt())) ;
                                        //Serial.print("Water Level (cm): ");
                                        Serial.println(waterLevelSensor.getSensorValue());
                                        }
                break;
                case(3): //Light
                                        {
                                I2cSensor lightSensor = I2cSensor(uint8_t(instruction.substring(4,7).toInt()),
bool(instruction.substring(7,8).toInt()));
                                //Serial.print("Light: ");
                                Serial.println(lightSensor.getSensorValue());
                                }
                break;
                case(4):        //temp
                                        {
                                //Serial.println("Temp");
                                //Serial.print("Sensor Pin ");
                                //Serial.println(instruction.substring(4,7).toInt());
                                //Serial.print("Power Pin ");
                                //uint8_t powerPin = uint8_t(instruction.substring(7,10).toInt());
                                //Serial.println(powerPin);
                                //Serial.println(uint8_t(instruction.substring(4,7).toInt()));
                                        DHT22Sensor tempHumidSensor =
DHT22Sensor(uint8_t(instruction.substring(4,7).toInt()), uint8_t(instruction.substring(7,10).toInt()),
false) ;
                                        //Serial.print("Temp (or humidity): ");
                                        Serial.println(tempHumidSensor.getSensorValue());
                                        }
                break;
                case(5): //Humidity
                                        {
                                Serial.println("Humidity");
                                        DHT22Sensor tempHumidSensor =
DHT22Sensor(uint8_t(instruction.substring(4,7).toInt()), uint8_t(instruction.substring(7,10).toInt()),
true) ;
                                        Serial.println(tempHumidSensor.getSensorValue());
                                        }
                break;
                //default:
                        //Serial.println("Unknown sensor");
                        }
}
void decodeAndExecuteCorrectorAction(String instruction){
```

```
        //Serial.println("Correction action about to take place");
        switch((instruction.substring(1,4)).toInt()){
        case(1):{       //on for a short period of time eg.PH Equaliser valve or water mixer
//                  Serial.print("Pin number ");
                    uint8_t pinNum = uint8_t(instruction.substring(4,7).toInt());
//                  Serial.print(pinNum);
                    int onTime = instruction.substring(7,12).toInt();
//                  Serial.print(" on for ");
//                  Serial.print(String(onTime));
//                  Serial.println(" m seconds");
                    StandardCorrector corrector =
StandardCorrector(uint8_t(instruction.substring(4,7).toInt()), true);       //initialise and turn on
equaliser valve
                    delay(instruction.substring(7,12).toInt());
                    corrector.turnOffCorrector();
                    Serial.println("Correction Made");
                    break;
        }
        case(2): {      //On or Off, eg. fan & pump
//                  Serial.print("Corrector at pin number ");
//                  Serial.print(uint8_t(instruction.substring(4,7).toInt()));
//                  if (bool(instruction.substring(7,8).toInt())==true){
//                      Serial.print(" turned on");
//                  }else{
//                      Serial.print(" turned off");
//                  }
                    StandardCorrector corrector =
StandardCorrector(uint8_t(instruction.substring(4,7).toInt()), bool(instruction.substring(7,8).toInt()));
        //initialise and turn on equaliser valve
                    Serial.println("Correction Made");
                    break;
        }
        case(3):{                       //LEDs PWM corrector On
                    //Serial.println("PWM corrector on");
                    PWMCorrector pwmCorrector =
PWMCorrector(uint8_t(instruction.substring(4,7).toInt()), bool(instruction.substring(12,13).toInt()),
uint16_t(instruction.substring(7,12).toInt()));
                    Serial.println("Correction Made");
        }
        }
}
#ifdef __arm__
// should use uinstd.h to define sbrk but Due causes a conflict
extern "C" char* sbrk(int incr);
#else // __ARM__
extern char *__brkval;
#endif // __arm__
int freeMemory() {
char top;
#ifdef __arm__
return &top - reinterpret_cast<char*>(sbrk(0));
#elif defined(CORE_TEENSY) || (ARDUINO > 103 && ARDUINO != 151)
```

```
return &top - __brkval;
#else // __arm__
return __brkval ? &top - __brkval : &top - __malloc_heap_start;
#endif // __arm__
}
void decodeSensorOrCorrector(String instruction){
if(instruction.substring(0,1) == "s"){                    //communication code is s (sensor read)
        decodeAndExecuteSensorRead(instruction);
}else if(instruction.substring(0,1) == "c"){                    //communication code is (correction action)
        decodeAndExecuteCorrectorAction(instruction);
}else if(instruction.substring(0,1) == "m"){
        Serial.println(freeMemory());
}
}
void setup() {
Serial.begin(9600);
Serial.print("Arduino Ready\n");
}

//int i;
static String input;
void loop() {
input = waitForInput();
decodeSensorOrCorrector(input);
}
```

Sensor.h

```
/*
* Sensor.h
*
* Created on: 19 Feb 2018
* Author: josephlaithwaite
*/
#include <Arduino.h>
#ifndef SENSOR_H_
#define SENSOR_H_
class Sensor {
public:
        Sensor(uint8_t sensorID);
        uint8_t  getSensorID();
        virtual int getSensorValue();
        virtual ~Sensor();
protected:
        uint8_t sensorID;
};
#endif /* SENSOR_H_ */
```

Sensor.cpp

```
/*
* Sensor.cpp
*
* Created on: 19 Feb 2018
* Author: josephlaithwaite
*/
```

```
#include "Sensor.h"
Sensor::Sensor(uint8_t sensorID) {
	this->sensorID = sensorID;
}
Sensor::~Sensor() {
	// TODO Auto-generated destructor stub
}

uint8_t  Sensor::getSensorID(){
	return sensorID;
}

int Sensor::getSensorValue(){
	return 0;
}
```

StandardAnalogSensor.h

```
/*
* StandardAnalogSensor.h
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
*/
#ifndef STANDARDANALOGSENSOR_H_
#define STANDARDANALOGSENSOR_H_
#include "Sensor.h"
class StandardAnalogSensor {
public:
	StandardAnalogSensor(uint8_t sensorPinNumber, uint8_t powerPinNumber = 255);
	virtual ~StandardAnalogSensor();
	uint8_t getSensorPinNumber();
	uint8_t getPowerPinNumber();
	virtual int getSensorValue();
protected:
	uint8_t sensorPinNumber;
	uint8_t powerPinNumber;
};
#endif /* STANDARDANALOGSENSOR_H_ */
```

StandardAnalogSensor.cpp

```
/*
* StandardAnalogSensor.cpp
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
*/
#include "StandardAnalogSensor.h"
StandardAnalogSensor::StandardAnalogSensor(uint8_t sensorPinNumber, uint8_t
powerPinNumber) {
	// TODO Auto-generated constructor stub
	this->sensorPinNumber = sensorPinNumber;
	this->powerPinNumber = powerPinNumber;
	pinMode(sensorPinNumber,INPUT);
}
```

```
StandardAnalogSensor::~StandardAnalogSensor() {
	// TODO Auto-generated destructor stub
}
int StandardAnalogSensor::getSensorValue(){
	int currentRawInput;
	if (powerPinNumber==255){		//code for no power pin used
		currentRawInput = analogRead(sensorPinNumber);
	}else{
		pinMode(powerPinNumber,OUTPUT);
		digitalWrite(powerPinNumber, HIGH);
		delay(500);
		currentRawInput = analogRead(sensorPinNumber);
		delay(300);
		pinMode(powerPinNumber, OUTPUT);
		pinMode(powerPinNumber, INPUT);
		digitalWrite(powerPinNumber, LOW);
	}
	return currentRawInput;
}
uint8_t StandardAnalogSensor::getSensorPinNumber(){
	return sensorPinNumber;
}
uint8_t StandardAnalogSensor::getPowerPinNumber(){
	return powerPinNumber;
}
```

RelayPoweredAnalogSensor.h

```
/*
* RelayPoweredAnalogSensor.h
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
*/
#ifndef RELAYPOWEREDANALOGSENSOR_H_
#define RELAYPOWEREDANALOGSENSOR_H_
#include "StandardAnalogSensor.h"
class RelayPoweredAnalogSensor: public StandardAnalogSensor {
public:
	RelayPoweredAnalogSensor(uint8_t sensorPinNumber, uint8_t powerPinNumber, uint8_t
relayPinNumber);
	virtual ~RelayPoweredAnalogSensor();
	uint8_t getRelayPin();
	int getSensorValue();
protected:
	uint8_t relayPinNumber;
};
#endif /* RELAYPOWEREDANALOGSENSOR_H_ */
```

RelayPoweredAnalogSensor.cpp

```
/*
* RelayPoweredAnalogSensor.cpp
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
```

```
*/
#include "RelayPoweredAnalogSensor.h"
RelayPoweredAnalogSensor::RelayPoweredAnalogSensor(uint8_t sensorPinNumber, uint8_t
powerPinNumber, uint8_t relayPinNumber) : StandardAnalogSensor(sensorPinNumber,
powerPinNumber) {
	this -> relayPinNumber = relayPinNumber;
}
uint8_t RelayPoweredAnalogSensor::getRelayPin(){
	return relayPinNumber;
}
int RelayPoweredAnalogSensor::getSensorValue(){
	pinMode(relayPinNumber,OUTPUT);
	digitalWrite(relayPinNumber, HIGH);
	delay(100);
	int currentRawInput = StandardAnalogSensor::getSensorValue();
	digitalWrite(relayPinNumber, LOW);
	pinMode(relayPinNumber, OUTPUT);
	pinMode(relayPinNumber, INPUT);
	digitalWrite(relayPinNumber, LOW);
	return currentRawInput;
}
RelayPoweredAnalogSensor::~RelayPoweredAnalogSensor() {
	// TODO Auto-generated destructor stub
}
```

DHT22Sensor.h

```
/*
* DHT22Sensor.h
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
*/
#ifndef DHT22SENSOR_H_
#define DHT22SENSOR_H_
#include "SimpleDHT.h"
#include "StandardAnalogSensor.h"
class DHT22Sensor: public StandardAnalogSensor {
public:
	DHT22Sensor(uint8_t sensorPinNumber, uint8_t powerPinNumber, bool humidityOrTemp);
	virtual ~DHT22Sensor();
	virtual int getSensorValue();
private:
	SimpleDHT22 tempHumidSensor;
	bool humidityOrTemp;
};
#endif /* DHT22SENSOR_H_ */
```

DHT22Sensor.cpp

```
/*
* DHT22Sensor.cpp
*
* Created on: 9 Mar 2018
* Author: josephlaithwaite
*/
```

```
#include "DHT22Sensor.h"
DHT22Sensor::DHT22Sensor(uint8_t sensorPinNumber, uint8_t powerPinNumber, bool
humidityOrTemp) : StandardAnalogSensor(sensorPinNumber, powerPinNumber) {
	// TODO Auto-generated constructor stub
	//Serial.println(String(sensorPinNumber) + " " + String(powerPinNumber) + " " +
String(humidityOrTemp));
	this->humidityOrTemp = humidityOrTemp;
}
DHT22Sensor::~DHT22Sensor() {
	// TODO Auto-generated destructor stub
}
int DHT22Sensor::getSensorValue(){
	//int err = 0;
	if (humidityOrTemp==true){
		//Serial.println("Getting humidity ");
		float humidity;
		//err =
		tempHumidSensor.read2(getSensorPinNumber(), NULL, &humidity, NULL);
		return int(humidity*10);
	}else{
		//Serial.println("Getting temperature ");
		float temperature;
		//err =
		tempHumidSensor.read2(getSensorPinNumber(), &temperature, NULL, NULL);
		return int(temperature*10);
	}
}
```

I2cSensor.h

```
/*
* I2cSensor.h
*
* Created on: Mar 9, 2018
* Author: Joseph Laithwaite
*/
#ifndef I2CSENSOR_H_
#define I2CSENSOR_H_
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include "Sensor.h"
class I2cSensor{
public:
	I2cSensor(uint8_t sensorID, bool floatOrLow);
	int getSensorValue();
	virtual ~I2cSensor();
private:
	Adafruit_TSL2561_Unified * tsl;
};
#endif /* I2CSENSOR_H_ */
```

I2cSensor.cpp

```
/*
* I2cSensor.cpp
*
* Created on: Mar 9, 2018
* Author: Joseph Laithwaite
*/
#include "I2cSensor.h"
I2cSensor::I2cSensor(uint8_t sensorID, bool floatOrLow){
	//Serial.println(String(sensorID) + " " + String(floatOrLow));
	if (floatOrLow==true){
		tsl = new Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, sensorID);
	}else{
		tsl = new Adafruit_TSL2561_Unified(TSL2561_ADDR_LOW, sensorID);
	}
	tsl->enableAutoRange(true); /* Auto-gain ... switches automatically between 1x and 16x */
	tsl->setIntegrationTime(TSL2561_INTEGRATIONTIME_13MS); /* fast but low resolution */
}
int I2cSensor::getSensorValue(){
	sensors_event_t event;
	tsl->getEvent(&event);
	return event.light;
}
I2cSensor::~I2cSensor() {
	delete tsl;
	// TODO Auto-generated destructor stub
}
```

StandarCorrector.h

```
/*
* StandardCorrector.h
```

```
*
* Created on: Mar 10, 2018
* Author: Joseph Laithwaite
*/
#ifndef STANDARDCORRECTOR_H_
#define STANDARDCORRECTOR_H_
#include "Arduino.h"
class StandardCorrector {
public:
	StandardCorrector(uint8_t correctorPin, bool correctorState = false);
	virtual ~StandardCorrector();
	virtual void turnOnCorrector();
	void turnOffCorrector();
	bool getCorrectorState();
	virtual void setCorrector();
protected:
	uint8_t correctorPin;
	bool correctorState;
};
#endif /* STANDARDCORRECTOR_H_ */
```

StandardCorrector.cpp

```
/*
* StandardCorrector.cpp
*
* Created on: Mar 10, 2018
* Author: Joseph Laithwaite
*/
#include "../CorrectorLibraries/StandardCorrector.h"
StandardCorrector::StandardCorrector(uint8_t correctorPin, bool correctorState) {
	this->correctorPin = correctorPin;
	this->correctorState = correctorState;
	if (correctorState==true){
		turnOnCorrector();
	}else{
		turnOffCorrector();
	}
}
void StandardCorrector::turnOnCorrector(){
	pinMode(correctorPin, OUTPUT);
	digitalWrite(correctorPin,HIGH);
//	Serial.println("Corrector ON");
}
void StandardCorrector::turnOffCorrector(){
	pinMode(correctorPin, INPUT);
	pinMode(correctorPin, OUTPUT);
	digitalWrite(correctorPin,LOW);
//	Serial.println("Corrector OFF");
}
bool StandardCorrector::getCorrectorState(){
	return correctorState;
}
void StandardCorrector::setCorrector(){}
```

```
StandardCorrector::~StandardCorrector() {
	// TODO Auto-generated destructor stub
}
```

PWMCorrector.h

```
/*
* PWMCorrector.h
*
* Created on: Mar 10, 2018
* Author: Joseph Laithwaite
*/
#ifndef PWMCORRECTOR_H_
#define PWMCORRECTOR_H_
#include "../CorrectorLibraries/StandardCorrector.h"
class PWMCorrector: public StandardCorrector {
public:
	PWMCorrector(uint8_t correctorPin, bool correctorState, uint16_t pwmValue);
	void setCorrector(uint16_t pwmValue);
	void turnOnCorrector();
	virtual ~PWMCorrector();
private:
	uint16_t pwmValue;
};
#endif /* PWMCORRECTOR_H_ */
```

PWMCorrector.cpp

```
/*
* PWMCorrector.cpp
*
* Created on: Mar 10, 2018
* Author: Joseph Laithwaite
*/
#include "../CorrectorLibraries/PWMCorrector.h"
PWMCorrector::PWMCorrector(uint8_t correctorPin, bool correctorState, uint16_t pwmValue) :
StandardCorrector(correctorPin, correctorState) {
	setCorrector(pwmValue);
	if (correctorState==true){
		turnOnCorrector();
	}
}
void PWMCorrector::setCorrector(uint16_t pwmValue){
	this->pwmValue = pwmValue;
}
void PWMCorrector::turnOnCorrector(){
//	Serial.print("Turning corrector pwm on");
	analogWrite(correctorPin, pwmValue);
}

PWMCorrector::~PWMCorrector() {
	// TODO Auto-generated destructor stub
}
```

Python Arduino, Raspberry Pi interface code

ArduinoController.py

```
# ---Import Libraries
import mysql.connector
# Database connection
import serial
# Connection with arduino
import time
# Sleep function
import arduino_sensor_corrector_control as ascc
config = {
  'user': 'joseph',
  'password': 'passcode',
  'host': 'localhost',
  'database': 'SmartGreenhouse_2',
  'raise_on_warnings': True,
}
arduino = serial.Serial("/dev/ttyUSB0", 9600)
# change ACM number as found from ls /dev/tty*
arduino.baudrate = 9600
time.sleep(2)
# Wait for arduino to restart after connecting
cnx = mysql.connector.connect(**config)
read_ser = arduino.readline()
# get input from arduino
while read_ser != b'Arduino Ready\n':
  read_ser = arduino.readline()
  print("Arduino not ready")
  # print error if arduino isn't ready
print("Arduino ready")
region_id = 1
# region_id = ser.readline()   Get region ID from Arduino
ascc.turn_all_correctors_off(region_id, cnx, arduino)
# As all correctors are on on startup, they must be turned off.
while arduino.isOpen():
  ascc.get_all_sensor_data_in_region(region_id, cnx, arduino)
  ascc.carry_out_corrections_on_all_modules(region_id, cnx, arduino)
  print("")
  # arduino.write("m")
  # Requests the dynamic memory space left on the Arduino, used for debugging
  # print(arduino.readline())
  # Displays the free memory
arduino.close()
# close serial connection when the program ends
```

arduino_sensor_corrector_control.py

```
import mysql.connector
# Database connection
import datetime
# Gets curent datetime
import sql_queeries_and_insertions as sql
# file of often used sql queeries and insertion code
def turn_all_correctors_off(region_id, cnx, ser):
  cursor = cnx.cursor()
  querry_get_corrector_info = ("""
    SELECT
      IC.CorrectorID,
      IC.CorrectorPin
```

```
        FROM
            InstalledCorrectors IC
        WHERE
            IC.FarmRegion_RegionID = '%(RegionID)s'
        """)
    data_region_id = {
        'RegionID': region_id
    }
    cursor.execute(querry_get_corrector_info, data_region_id)
    for (CorrectorID, CorrectorPin) in cursor:
        if CorrectorPin is None:
            # LED corrector
            cnx_2 = mysql.connector.connect(**sql.config)
            cursor_2 = cnx_2.cursor()
            querry_get_all_led_pins = ("""
                SELECT
                    RedPin,
                    GreenPin,
                    BluePin
                FROM
                    LEDCorrector
                WHERE
                    InstalledCorrectors_CorrectorID = '%(CorrectorID)s'
                """)
            data_get_all_led_pins = {
                'CorrectorID': CorrectorID
            }
            cursor_2.execute(querry_get_all_led_pins, data_get_all_led_pins)
            for (RedPin, GreenPin, BluePin) in cursor_2:
                ser.write("c002{:03}0".format(RedPin))
                correction_response = ser.readline()
                while correction_response != 'Correction Made\r\n':
                    correction_response = ser.readline()
                ser.write("c002{:03}0".format(GreenPin))
                correction_response = ser.readline()
                while correction_response != 'Correction Made\r\n':
                    correction_response = ser.readline()
                ser.write("c002{:03}0".format(BluePin))
                correction_response = ser.readline()
                while correction_response != 'Correction Made\r\n':
                    correction_response = ser.readline()
            cursor_2.close()
            cnx_2.close()
        else:  # Other corrector
            ser.write("c002{:03}0".format(CorrectorPin))
            correction_response = ser.readline()
            while correction_response != 'Correction Made\r\n':
                correction_response = ser.readline()
        # Save change to db
        if bool(sql.check_if_corrector_is_on(CorrectorID, cnx)):
            sql.turn_correction_off(datetime.datetime.now(), int(CorrectorID), cnx)
def get_all_sensor_data_in_region(region_id, cnx, ser):
    cursor = cnx.cursor()
    querry_get_all_sensor_data_in_region = (
        """
        SELECT
            I_S.SensorID,
            I_S.SensorTypes_SensorTypeID,
```

```
        I_S.Internal,
        I_S.SensorInputPin,
        I_S.sensorPowerPin,
        PR.RelayPin,
        I2C.AddrFloatOrGround
    FROM
        InstalledSensors I_S
            LEFT JOIN
        PowerRelay PR ON I_S.SensorID = PR.InstalledSensors_SensorID
            LEFT JOIN
        I2cBusSensor I2C ON I_S.SensorID = I2C.InstalledSensors_SensorID
    WHERE
        I_S.FarmRegion_RegionID = 1
    """)
  data_get_all_sensor_data_in_region = {
    'RegionID': region_id
  }
  cursor.execute(querry_get_all_sensor_data_in_region, data_get_all_sensor_data_in_region)
  for (SensorID, SensorTypes_SensorTypeID, Internal, SensorInputPin, sensor_power_pin, RelayPin,
       AddrFloatOrGround) in cursor:
      # Convert None (no power pin) to 255 (Arduino will ignore 255)
      if sensor_power_pin is None:
          sensor_power_pin = 255
      # PH Sensor
      if SensorTypes_SensorTypeID == 1:
          ser.write('s{:03}{:03}{:03}'.format(SensorTypes_SensorTypeID, SensorInputPin, sensor_power_pin))
          print('s{:03}{:03}{:03}'.format(SensorTypes_SensorTypeID, SensorInputPin, sensor_power_pin))
          raw_ph_reading = ser.readline()
          ph_value = 17.5 * (int(raw_ph_reading) / 1024)
          print(ph_value)
          sql.add_sensor_data(datetime.datetime.now(), SensorID, ph_value, cnx)
      # Water Level
      elif SensorTypes_SensorTypeID == 2:
          print("s{:03}{:03}{:03}{:03}".format(SensorTypes_SensorTypeID, SensorInputPin, sensor_power_pin,
RelayPin))
          ser.write(
              "s{:03}{:03}{:03}{:03}".format(SensorTypes_SensorTypeID, SensorInputPin, sensor_power_pin,
RelayPin))
          print("sent to arduino")
          try:
              raw_water_height_data = ser.readline()
          except:
              print("exception reached")
              raw_water_height_data = 0
          print("Water level read")
          if int(raw_water_height_data) <= 340:
              # if input lower than 340 equation creates below 0.1 and negatives.
              water_volume = 0
          else:
              # 1296 = container base area or ml per mm height
              # water_volume = int(((int(raw_water_height_data) * 0.356) - 97.319) * 1296)
              water_volume = int((((9 * pow(10, -25)) * pow((int(raw_water_height_data)), 9.1084)) * 0.1) * 1296)
          print(int(water_volume))
          sql.add_sensor_data(datetime.datetime.now(), SensorID, water_volume, cnx)
      # Lux
      elif SensorTypes_SensorTypeID == 3:
          print("s{:03}{:03}{:1}".format(SensorTypes_SensorTypeID, SensorID, AddrFloatOrGround))
          ser.write("s{:03}{:03}{:1}".format(SensorTypes_SensorTypeID, SensorID, AddrFloatOrGround))
```

```
            print("waiting for lux reading")
            lux_reading = ser.readline()
            print(int(lux_reading))
            sql.add_sensor_data(datetime.datetime.now(), SensorID, lux_reading, cnx)
        # DHT22 sensor
        # type = 4 for temperature and type = 5 for humidity
        elif SensorTypes_SensorTypeID == 4 or SensorTypes_SensorTypeID == 5:
            print("s{:03}{:03}{:03}".format(SensorTypes_SensorTypeID, SensorInputPin, sensor_power_pin))
            ser.write("s004{:03}{:03}".format(SensorInputPin, sensor_power_pin))
            sensor_read = ser.readline()
            print(float(int(sensor_read) / 10))
            sql.add_sensor_data(datetime.datetime.now(), SensorID, float(int(sensor_read) / 10), cnx)
    cursor.close()
def carry_out_corrections_on_all_modules(region_id, cnx, ser):
    cursor = cnx.cursor()
    querry_get_module_conditions_info = ("""
        SELECT
            CM.ModuleID,
            CM.ModuleCode,
            DC.UpperLimit,
            DC.LowerLimit
        FROM
            CorrectionModule CM
                INNER JOIN
            DesiredConditions DC ON CM.ModuleID = DC.CorrectionModule_ModuleID
        WHERE
            CM.FarmRegion_RegionID = '%(RegionID)s'
                AND
            (
            DC.ConditionEndTime IS NULL
                OR
            DC.ConditionEndTime >= NOW()
            )
            AND DC.ConditionStartTime <= NOW()
    """)
    data_get_module_conditions_info = {
        'RegionID': region_id
    }
    cursor.execute(querry_get_module_conditions_info, data_get_module_conditions_info)
    for (ModuleID, ModuleCode, UpperLimit, LowerLimit) in cursor:
        print("Condition upper limit: " + str(UpperLimit) + " condition lower limit: " + str(LowerLimit))
        # get sensorID for sensor
        sensor_id = sql.get_sensor_id_with_module_info(ModuleID, ModuleCode, cnx)
        print ("Sensor ID: " + str(sensor_id[0]))
        # get sensor reading
        sensor_reading = sql.get_most_recent_sensor_reading(int(sensor_id[0]), cnx)
        print ("Most recent sensor reading with sensor id " + str(sensor_id[0]) + ": " + str(sensor_reading[0]))
        if ModuleCode == 1:
            # PH Module
            # compare sensor reading to limits
            if float(sensor_reading[0]) > float(UpperLimit):
                # Water more basic than desired
                print ("Corrrection required, ph>upper limit")
                # get sensorID for water level using Water volume sensor type id = 2
                water_sensor_id = sql.get_sensor_id_with_module_info(ModuleID, 2, cnx)
                # get water volume sensor reading
                water_volume = sql.get_most_recent_sensor_reading(water_sensor_id[0], cnx)
                print ("Most recent water level with sensor id " + water_sensor_id + ": " + str(water_volume[0]))
```

```
        # calculate ph required
        desired_ph_decrease = float(sensor_reading[0]) - (
                (float(LowerLimit)) + (float(UpperLimit) - float(LowerLimit)) / 2)
        equaliser_required = float((water_volume[0] * desired_ph_decrease) * 0.01)
        print("For a ph decrease of " + str(desired_ph_decrease) + ", " + str(
            equaliser_required) + " ml of equaliser is needed")
        # get ph corrector info using PH correctoion type 1
        corrector_info = sql.get_corrector_info(ModuleID, 1, cnx)
        print("valve corrector pin is: " + str(corrector_info[0]))
        # send corrector info to arduino, flow rate is 2ml/s and x1000 as Arduino delay in ms
        ser.write("c001{:03}{:05}".format(int(corrector_info[0]), int(
            equaliser_required * 2000)))
        corrcetion_response = ser.readline()
        while corrcetion_response != 'Correction Made\r\n':
            corrcetion_response = ser.readline()
        start_time = datetime.datetime.now() - datetime.timedelta(seconds=equaliser_required * 2)
        sql.turn_correction_on(start_time, corrector_info[1], cnx)
        sql.add_ph_correction_data(start_time, corrector_info[1], equaliser_required, cnx)
        sql.turn_correction_off(datetime.datetime.now(), int(corrector_info[1]), cnx)
    # WaterSensorModule
    elif ModuleCode == 2:
        # Get correcor pump pin water pump has corrector type 2
        corrector_info = sql.get_corrector_info(ModuleID, 2, cnx)
        # Get if pump is on or off
        pump_on = bool(sql.check_if_corrector_is_on(corrector_info[0], cnx))
        print("pump on is " + str(pump_on))
        if float(sensor_reading[0]) < float(LowerLimit) and pump_on:
            # Turn pump off
            print("Turning off pump")
            ser.write("c002{:03}0".format(int(corrector_info[0])))
            corrcetion_response = ser.readline()
            while corrcetion_response != 'Correction Made\r\n':
                corrcetion_response = ser.readline()
            # Save change to db
            sql.turn_correction_off(datetime.datetime.now(), int(corrector_info[1]), cnx)
        if float(sensor_reading[0]) > float(LowerLimit) and not pump_on:
            print("Turning on pump")
            # Turn pump on
            ser.write("c002{:03}1".format(int(corrector_info[0])))
            print("c002{:03}1".format(int(corrector_info[0])))
            corrcetion_response = ser.readline()
            while corrcetion_response != 'Correction Made\r\n':
                corrcetion_response = ser.readline()
            # Save change to db
            sql.turn_correction_on(datetime.datetime.now(), corrector_info[1], cnx)
    elif ModuleCode == 4 or ModuleCode == 5:
        # temp module & humid module
        print("Temp module, upper temp level: " + str(UpperLimit))
        # get sensorID for internal dht22 sensor
        sensor_id = sql.get_int_sensor_id_with_module_and_sensor_type(ModuleID, ModuleCode, True, cnx)
        print ("Sensor ID: " + str(sensor_id[0]))
        # get sensor reading
        int_temp = sql.get_most_recent_sensor_reading(int(sensor_id[0]), cnx)
        print ("Most recent temp/humidity reading: " + str(int_temp[0]))
        # Get correcor pin for fan
        corrector_info = sql.get_corrector_info(ModuleID, ModuleCode, cnx)
        # Get if fan is on or off
        fan_on = bool(sql.check_if_corrector_is_on(corrector_info[1], cnx))
```

```
        print("Fan on: " + str(fan_on))
        if (float(int_temp[0]) < float(UpperLimit)) and fan_on == 1:
            # Turn pump off
            print("turning off fan")
            ser.write("c002{:03}0".format(int(corrector_info[0])))
            print("c002{:03}0".format(int(corrector_info[0])))
            corrcetion_response = ser.readline()
            while corrcetion_response != 'Correction Made\r\n':
                corrcetion_response = ser.readline()
            # Save change to db
            sql.turn_correction_off(datetime.datetime.now(), int(corrector_info[1]), cnx)
        if (float(int_temp[0]) > float(UpperLimit)) and fan_on == 0:
            print("Turning on fan")
            # Turn fan on
            ser.write("c002{:03}1".format(int(corrector_info[0])))
            print("c002{:03}1".format(int(corrector_info[0])))
            corrcetion_response = ser.readline()
            print(corrcetion_response)
            while corrcetion_response != 'Correction Made\r\n':
                corrcetion_response = ser.readline()
            print("fan correction made")
            # Save change to db
            # Fan has corrcetion id 4
            sql.turn_correction_on(datetime.datetime.now(), corrector_info[1], cnx)
    elif ModuleCode == 3:
        # Light module
        print("Light")
        # update sensor id with external id, light senor type id = 3
        ext_sensor_id = sql.get_int_sensor_id_with_module_and_sensor_type(ModuleID, 3, False, cnx)
        # check if the sensor ID alreayd attained is the same as the external
        if ext_sensor_id != sensor_id:
            ext_sensor_reading = sql.get_most_recent_sensor_reading(sensor_id[0], cnx)
        else:
            ext_sensor_reading = sensor_reading
        # LED strip has corrector ID 3
        corrector_info = sql.get_corrector_info(ModuleID, 3, cnx)
        leds_on = bool(sql.check_if_corrector_is_on(corrector_info[1], cnx))
        if ext_sensor_reading[0] > float(LowerLimit) and leds_on == 1:
            # Light enough without light so turn them off
            print("Light doesn't need to be on")
            # if leds_on:
            print("Light was on and is about to turn off")
            pins_array = sql.get_led_pins(int(corrector_info[1]), cnx)
            for i in range(0, 3):
                ser.write("c002{:03}0".format(int(pins_array[i])))
                print("c002{:03}0".format(int(pins_array[i])))
                corrcetion_response = ser.readline()
                while corrcetion_response != 'Correction Made\r\n':
                    corrcetion_response = ser.readline()
            sql.turn_correction_off(datetime.datetime.now(), int(corrector_info[1]), cnx)
        if ext_sensor_reading[0] < float(LowerLimit) and leds_on == 0:
            print("Light should be on")
            intensity = 255
            print("Light was off and about to turn on")
            rgb_array = sql.get_desired_rgb_percentages(ModuleID, cnx)
            pins_array = sql.get_led_pins(int(corrector_info[1]), cnx)
            print("red pin: " + str(pins_array[0]) + " green pin: " + str(pins_array[1]) + " blue pin: " + str(
                pins_array[2]))
```

```
            print("red percent: " + str(rgb_array[0]) + "green percent: " + str(
                rgb_array[1]) + "blue percent: " + str(rgb_array[2]))
            for i in range(0, 3):
                colour_output = int(intensity * (float(rgb_array[i]) / 100))
                ser.write("c003{:03}{:05}1".format(int(pins_array[i]), colour_output))
                print("c003{:03}{:05}1".format(int(pins_array[i]), colour_output));
                corrcetion_response = ser.readline()
                while corrcetion_response != 'Correction Made\r\n':
                    corrcetion_response = ser.readline()
            sql.turn_correction_on(datetime.datetime.now(), corrector_info[1], cnx)
    cursor.close()
```

sql_squeeries_and_insertions.py

```
import mysql.connector
# Database connection
config = {
  'user': 'joseph',
  'password': 'passcode',
  'host': 'localhost',
  'database': 'SmartGreenhouse_2',
  'raise_on_warnings': True,
}
def add_sensor_data(current_time, sensor_id, sensor_reading):
  cnx = mysql.connector.connect(**config)
  crs = cnx.cursor()
  add_sensor_reading = (
    """
    INSERT INTO SmartGreenhouse_2.SensorReadings (TimeStamp, InstalledSensors_SensorID,
SensorValue)
    VALUES(%(TimeStamp)s, %(InstalledSensors_SensorID)s, %(SensorValue)s)
    """)
  data_sensor_reading = {
    'TimeStamp': current_time.strftime("%Y-%m-%d %H:%M:%S"),
    'InstalledSensors_SensorID': sensor_id,
    'SensorValue': float(sensor_reading)
  }
  crs.execute(add_sensor_reading, data_sensor_reading)
  cnx.commit()
  crs.close()
  cnx.close()
def get_led_pins(corrector_id):
  cnx = mysql.connector.connect(**config)
  crs = cnx.cursor()
  querry_get_led_pins = (
    """
    SELECT
      RedPin,
      GreenPin,
      BluePin
    FROM
      LEDCorrector
    WHERE
      LEDCorrector.InstalledCorrectors_CorrectorID = '%(CorrectorID)s'
    """)
  data_get_led_pins = {
    'CorrectorID': corrector_id
  }
  crs.execute(querry_get_led_pins, data_get_led_pins)
  for (RedPin, GreenPin, BluePin) in crs:
```

```
        return [RedPin, GreenPin, BluePin]
    crs.close()
    cnx.close()
def get_desired_rgb_percentages(module_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_get_rgb_desired_conditions = (
        """
        SELECT
          DLT.DesiredRedPercentage,
          DLT.DesiredGreenPercentage,
          DLT.DesiredBluePercentage
        FROM
          DesiredConditions DC
            INNER JOIN
          DesiredLightType DLT
            ON
            DC.CorrectionModule_ModuleID = DLT.DesiredConditions_CorrectionModule_ModuleID
              AND
            DC.ConditionStartTime = DLT.DesiredConditions_ConditionStartTime
        WHERE
          DC.CorrectionModule_ModuleID = '%(ModuleID)s'
            AND
            (
                DC.ConditionEndTime IS NULL
                OR DC.ConditionEndTime >= NOW()
            )
            AND DC.ConditionStartTime <= NOW()
      """
    )
    data_get_rgb_desired_conditions = {
        'ModuleID': module_id
    }
    crs.execute(querry_get_rgb_desired_conditions, data_get_rgb_desired_conditions)
    for (DesiredRedPercentage, DesiredGreenPercentage, DesiredBluePercentage) in crs:
        return [DesiredRedPercentage, DesiredGreenPercentage, DesiredBluePercentage]
    crs.close()
    cnx.close()
def get_sensor_id_with_module_info(module_id, sensor_type_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_get_sensor_id_with_module_info = (
        """
        SELECT
          I_S.SensorID
        FROM
          CorrectionModule_has_InstalledSensors CMIS
            INNER JOIN
          InstalledSensors I_S ON CMIS.InstalledSensors_SensorID = I_S.SensorID
        WHERE
          I_S.SensorTypes_SensorTypeID = '%(SensorType)s'
            AND
          CMIS.CorrectionModule_ModuleID = '%(ModuleID)s'
        """)
    data_get_get_sensor_id_with_module_info = {
        'SensorType': int(sensor_type_id),
        'ModuleID': int(module_id)
    }
```

```
    crs.execute(querry_get_sensor_id_with_module_info, data_get_get_sensor_id_with_module_info)
    for (SensorID) in crs:
        return int(SensorID)
    crs.close()
    cnx.close()
def get_int_sensor_id_with_module_and_sensor_type(module_id, sensor_type_id, internal):
    print(str(module_id) + " " + str(sensor_type_id))
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_get_sensor_is_with_module_and_sensor_type = (
        """
        SELECT
            I_S.SensorID
        FROM
            CorrectionModule_has_InstalledSensors CMIS
                INNER JOIN
            InstalledSensors I_S ON CMIS.InstalledSensors_SensorID = I_S.SensorID
        WHERE
            I_S.SensorTypes_SensorTypeID = '%(SensorType)s'
                AND
            I_S.Internal = '%(Internal)s'
                AND
            CMIS.CorrectionModule_ModuleID = '%(ModuleID)s'
        """)
    data_get_sensor_id_with_module_and_sensor_type = {
        'SensorType': int(sensor_type_id),
        'ModuleID': int(module_id),
        'Internal': int(internal)
    }
    crs.execute(querry_get_sensor_is_with_module_and_sensor_type,
data_get_sensor_id_with_module_and_sensor_type)
    for (SensorID) in crs:
        return SensorID
    # return cursor.fetchone()
    crs.close()
    cnx.close()
def get_most_recent_sensor_reading(sensor_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_get_sensor_readings = (
        """
        SELECT
            SR.SensorValue
        FROM
            SensorReadings SR
        WHERE
            SR.InstalledSensors_SensorID = '%(SensorID)s'
        ORDER BY
            SR.TimeStamp DESC
        LIMIT
            1
        """)
    data_get_sensor_readings = {
        'SensorID': int(sensor_id)
    }
    crs.execute(querry_get_sensor_readings, data_get_sensor_readings)
    for (SensorValue) in crs:
        return SensorValue
```

```
    crs.close()
    cnx.close()
def get_standard_desired_conditions(module_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_get_standard_desired_conditions = (
        """SELECT
              DC.UpperLimit,
              DC.LowerLimit
            FROM
              DesiredConditions DC
            WHERE
              DC.CorrectionModule_ModuleID = '%(ModuleID)s'
              AND
              (
                DC.ConditionEndTime IS NULL
                OR DC.ConditionEndTime >= NOW()
              )
              AND DC.ConditionStartTime <= NOW()"""
    )
    data_get_standard_desired_conditions = {
        'ModuleID': module_id
    }
    crs.execute(querry_get_standard_desired_conditions, data_get_standard_desired_conditions)
    for (ConditionEndTime, UpperLimit, LowerLimit) in crs:
        print("ConditionEndTime {}, UpperLimit {}, LowerLimit {}".format(ConditionEndTime, UpperLimit,
LowerLimit))
        return[ConditionEndTime, UpperLimit, LowerLimit]
    crs.close()
    cnx.close()
def get_corrector_info(module_id, corrector_type_id):
    querry_get_corrector_info = ("""
        SELECT
          IC.CorrectorPin,
          IC.CorrectorID
        FROM
          InstalledCorrectors_has_CorrectionModule ICMC
            INNER JOIN
          InstalledCorrectors IC ON ICMC.InstalledCorrectors_CorrectorID = IC.CorrectorID
        WHERE
          ICMC.CorrectionModule_ModuleID = '%(ModuleID)s'
            AND
          IC.CorrectionTypes_CorrectionTypeID = '%(CorrectionTypeID)s'
         ;""")
    data_get_corrector_info = {
        'ModuleID': module_id,
        'CorrectionTypeID': corrector_type_id
    }
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    crs.execute(querry_get_corrector_info, data_get_corrector_info)
    for (CorrectorPin, CorrectorID) in crs:
        return [CorrectorPin, CorrectorID]
    crs.close()
    cnx.close()
def get_corrector_pin(module_id, corrector_type_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
```

```
    querryget_corrector_info = (
        "SELECT "
        "    IC.CorrectorPin "
        "FROM "
        "    InstalledCorrectors_has_CorrectionModule ICMC "
        "        INNER JOIN "
        "    InstalledCorrectors IC ON ICMC.InstalledCorrectors_CorrectorID = IC.CorrectorID "
        "WHERE "
        "    ICMC.CorrectionModule_ModuleID = '%(ModuleID)s'"
        "        AND"
        "   IC.CorrectionTypes_CorrectionTypeID = '%(CorrectionTypeID)s' "
        " "
    )
    dataget_corrector_info = {
        'ModuleID': module_id,
        'CorrectionTypeID': corrector_type_id
    }
    crs.execute(querryget_corrector_info, dataget_corrector_info)
    for (CorrectorPin) in crs:
        return CorrectorPin
    crs.close()
    cnx.close()
def check_if_corrector_is_on(corrector_id):
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
    querry_check_if_corrector_is_on = (
        """SELECT
             EXISTS
          (
            SELECT
              *
            FROM
               CorrectionAction CA
            WHERE
              CA.InstalledCorrectors_CorrectorID = '%(CorrectorID)s'
                AND
              CA.TimeStampStart<=NOW()
                AND
              CA.TimeStampEnd IS NULL
          )
          AS CorrectorOn """
    )
    data_check_if_corrector_is_on = {
        'CorrectorID': corrector_id
    }
    crs.execute(querry_check_if_corrector_is_on, data_check_if_corrector_is_on)
    for (CorrectorOn) in crs:
        return int(CorrectorOn[0])
        # if int(CorrectorOn[0]) == 1:
        #    return True
        # else:
        #    return False
    crs.close()
    cnx.close()
def turn_correction_off(end_time, corrector_id):
    # print("turning correction off " + str(corrector_id) + "  " + str(end_time))
    cnx = mysql.connector.connect(**config)
    crs = cnx.cursor()
```

```
  add_update_correction = ("UPDATE CorrectionAction "
                           "SET TimeStampEnd = %(end_time)s "
                           "WHERE InstalledCorrectors_CorrectorID= %(corrector_id)s AND TimeStampEnd IS
NULL"
                           )
  data_update_correction = {
    'end_time': end_time,
    'corrector_id': corrector_id,
  }
  crs.execute(add_update_correction, data_update_correction)
  cnx.commit()
  crs.close()
  cnx.close()
def turn_correction_on(start_time, corrector_id):
  cnx = mysql.connector.connect(**config)
  crs = cnx.cursor()
  add_start_correction_action = ("INSERT INTO SmartGreenhouse_2.CorrectionAction"
                                 "(TimeStampStart, InstalledCorrectors_CorrectorID)"
                                 "VALUES(%(TimeStampStart)s, %(InstalledCorrectors_CorrectorID)s)"
                                 )
  data_start_correction_action = {
    'TimeStampStart': start_time,
    'InstalledCorrectors_CorrectorID': corrector_id
  }
  crs.execute(add_start_correction_action, data_start_correction_action)
  cnx.commit()
  crs.close()
  cnx.close()
def get_corrector_id(module_id, corrector_type_id):
  cnx = mysql.connector.connect(**config)
  crs = cnx.cursor()
  querry_get_corrector_id = (
    """
    SELECT
      IC.CorrectorID
    FROM
      InstalledCorrectors_has_CorrectionModule ICMC
        INNER JOIN
      InstalledCorrectors IC ON ICMC.InstalledCorrectors_CorrectorID = IC.CorrectorID
    WHERE
      ICMC.CorrectionModule_ModuleID = '%(ModuleID)s'
        AND
      IC.CorrectionTypes_CorrectionTypeID = '%(CorrectorTypeID)s'
    """
  )
  data_get_corrector_id = {
    'ModuleID': module_id,
    'CorrectorTypeID': corrector_type_id
  }
  crs.execute(querry_get_corrector_id, data_get_corrector_id)
  for (CorrectorID) in crs:
    return CorrectorID
  crs.close()
  cnx.close()
def add_ph_correction_data(start_time, corrector_id, ph_equaliser_volume):
  # add_start_correction_action_data(start_time, corrector_id)
  cnx = mysql.connector.connect(**config)
  crs = cnx.cursor()
```

```
  add_ph_correction = ("""
  INSERT INTO SmartGreenhouse_2.PHCorrection
     (PHEqualiserVolume, CorrectionAction_TimeStampStart,
     CorrectionAction_InstalledCorrectors_CorrectorID)
  VALUES(%(PHEqualiserVolume)s, %(CorrectionAction_TimeStampStart)s,
  %(CorrectionAction_InstalledCorrectors_CorrectorID)s)"
  """)
  data_ph_correction = {
     'PHEqualiserVolume': ph_equaliser_volume,
     'CorrectionAction_TimeStampStart': start_time,
     'CorrectionAction_InstalledCorrectors_CorrectorID': corrector_id,
  }
  crs.execute(add_ph_correction, data_ph_correction)
  cnx.commit()
  crs.close()
  cnx.close()
```

MySQL code for creation of the database

```
-- MySQL Script generated by MySQL Workbench
-- Tue Apr 10 21:04:52 2018
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @old_unique_checks=@@unique_checks, unique_checks=0;SET @old_foreign_ke
y_checks=@@foreign_key_checks, foreign_key_checks=0;SET @old_sql_mode=@@sql
_mode, sql_mode='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema SmartGreenhouse_2
-- -----------------------------------------------------
DROP SCHEMA IF EXISTS `smartgreenhouse_2` ;

-- -----------------------------------------------------
-- Schema SmartGreenhouse_2
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `smartgreenhouse_2` DEFAULT CHARACTER SET utf8
;USE `smartgreenhouse_2` ;

-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`Farmer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`farmer` ;CREATE TABLE IF NOT EXIS
TS `smartgreenhouse_2`.`farmer` (
  `farmerid` INT NOT NULL auto_increment,
  `farmeremail` VARCHAR(20) NOT NULL,
  `farmername` VARCHAR(45) NOT NULL,
  `farmerpassword` VARCHAR(60) NOT NULL,
  PRIMARY KEY (`farmerid`))
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`Farm`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`farm` ;CREATE TABLE IF NOT EXISTS
 `smartgreenhouse_2`.`farm` (
  `farmname` VARCHAR(20) NOT NULL,
  `farmid` INT NOT NULL auto_increment,
  `farmer_farmerid` INT NOT NULL,
  PRIMARY KEY (`farmid`, `farmer_farmerid`),
```

```
  INDEX `fk_farm_farmer_idx` (`farmer_farmerid` asc),
  CONSTRAINT `fk_farm_farmer`
    FOREIGN KEY (`farmer_farmerid`)
    REFERENCES `smartgreenhouse_2`.`farmer` (`farmerid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`FarmRegion`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`farmregion` ;CREATE TABLE IF NOT
EXISTS `smartgreenhouse_2`.`farmregion` (
  `regionid` INT NOT NULL auto_increment,
  `farmregionname` VARCHAR(15) NOT NULL,
  `regiondescription` VARCHAR(45) NULL,
  `farm_farmid` INT NOT NULL,
  PRIMARY KEY (`regionid`, `farm_farmid`),
  INDEX `fk_farmregion_farm1_idx` (`farm_farmid` asc),
  CONSTRAINT `fk_farmregion_farm1`
    FOREIGN KEY (`farm_farmid`)
    REFERENCES `smartgreenhouse_2`.`farm` (`farmid`)
    ON DELETE RESTRICT
    ON UPDATE RESTRICT)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`SensorTypes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`sensortypes` ;CREATE TABLE IF NOT
 EXISTS `smartgreenhouse_2`.`sensortypes` (
  `sensortypeid` INT NOT NULL auto_increment,
  `sensorname` VARCHAR(45) NOT NULL,
  `sensorunit` VARCHAR(5) NULL,
  PRIMARY KEY (`sensortypeid`))
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`InstalledSensors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`installedsensors` ;CREATE TABLE I
F NOT EXISTS `smartgreenhouse_2`.`installedsensors` (
  `sensorid` INT NOT NULL auto_increment,
  `farmregion_regionid` INT NOT NULL,
  `sensortypes_sensortypeid` INT NOT NULL,
  `sensornickname` VARCHAR(20) NULL,
  `internal` TINYINT(1) NOT NULL DEFAULT 1,
  `sensorinputpin` INT NULL,
  `sensorpowerpin` INT NULL,
  PRIMARY KEY (`sensorid`, `farmregion_regionid`, `sensortypes_sensortypeid
`),
  INDEX `fk_installedsensors_sensortypes1_idx` (`sensortypes_sensortypeid`
asc),
  INDEX `fk_sensor_farmregion_idx` (`farmregion_regionid` ASC),
  CONSTRAINT `fk_sensor_farmregion`
    FOREIGN KEY (`farmregion_regionid`)
    REFERENCES `smartgreenhouse_2`.`farmregion` (`regionid`)
    ON DELETE no action
```

```
    ON UPDATE no action,
  CONSTRAINT `fk_installedsensors_sensortypes1`
    FOREIGN KEY (`sensortypes_sensortypeid`)
    REFERENCES `smartgreenhouse_2`.`sensortypes` (`sensortypeid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CorrectionTypes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`correctiontypes` ;CREATE TABLE IF
 NOT EXISTS `smartgreenhouse_2`.`correctiontypes` (
  `correctiontypeid` INT NOT NULL auto_increment,
  `correctionname` VARCHAR(20) NOT NULL,
  `correctiondescription` VARCHAR(45) NULL,
  PRIMARY KEY (`correctiontypeid`))
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`InstalledCorrectors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`installedcorrectors` ;CREATE TABL
E IF NOT EXISTS `smartgreenhouse_2`.`installedcorrectors` (
  `correctorid` INT NOT NULL auto_increment,
  `correctiontypes_correctiontypeid` INT NOT NULL,
  `farmregion_regionid` INT NOT NULL,
  `correctorpin` INT NULL,
  PRIMARY KEY (`correctorid`, `correctiontypes_correctiontypeid`, `farmregi
on_regionid`),
  INDEX `fk_installedcorrectors_correctiontypes1_idx` (`correctiontypes_cor
rectiontypeid` asc),
  INDEX `fk_installedcorrectors_farmregion1_idx` (`farmregion_regionid` ASC
),
  CONSTRAINT `fk_installedcorrectors_correctiontypes1`
    FOREIGN KEY (`correctiontypes_correctiontypeid`)
    REFERENCES `smartgreenhouse_2`.`correctiontypes` (`correctiontypeid`)
    ON DELETE no action
    ON UPDATE no action,
  CONSTRAINT `fk_installedcorrectors_farmregion1`
    FOREIGN KEY (`farmregion_regionid`)
    REFERENCES `smartgreenhouse_2`.`farmregion` (`regionid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`SensorReadings`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`sensorreadings` ;CREATE TABLE IF
NOT EXISTS `smartgreenhouse_2`.`sensorreadings` (
  `timestamp` DATETIME NOT NULL,
  `sensorvalue` FLOAT NULL,
  `installedsensors_sensorid` INT NOT NULL,
  `error` TINYINT(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`timestamp`, `installedsensors_sensorid`),
  INDEX `fk_sensorreadings_installedsensors1_idx` (`installedsensors_sensor
id` asc),
```

```
  CONSTRAINT `fk_sensorreadings_installedsensors1`
    FOREIGN KEY (`installedsensors_sensorid`)
    REFERENCES `smartgreenhouse_2`.`installedsensors` (`sensorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CorrectionAction`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`correctionaction` ;CREATE TABLE I
F NOT EXISTS `smartgreenhouse_2`.`correctionaction` (
  `timestampstart` DATETIME NOT NULL,
  `installedcorrectors_correctorid` INT NOT NULL,
  `timestampend` DATETIME NULL,
  PRIMARY KEY (`installedcorrectors_correctorid`, `timestampstart`),
  INDEX `fk_sensorreadings_copy1_installedcorrectors1_idx` (`installedcorre
ctors_correctorid` asc),
  CONSTRAINT `fk_sensorreadings_copy1_installedcorrectors1`
    FOREIGN KEY (`installedcorrectors_correctorid`)
    REFERENCES `smartgreenhouse_2`.`installedcorrectors` (`correctorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`PHCorrection`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`phcorrection` ;CREATE TABLE IF NO
T EXISTS `smartgreenhouse_2`.`phcorrection` (
  `phequaliservolume` FLOAT NOT NULL,
  `correctionaction_installedcorrectors_correctorid` INT NOT NULL,
  `correctionaction_timestampstart` DATETIME NOT NULL,
  PRIMARY KEY (`correctionaction_installedcorrectors_correctorid`, `correct
ionaction_timestampstart`),
  CONSTRAINT `fk_phcorrection_correctionaction1`
    FOREIGN KEY (`correctionaction_installedcorrectors_correctorid` , `corr
ectionaction_timestampstart`)
    REFERENCES `smartgreenhouse_2`.`correctionaction` (`installedcorrectors
_correctorid` , `timestampstart`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`LEDCorrection`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`ledcorrection` ;CREATE TABLE IF N
OT EXISTS `smartgreenhouse_2`.`ledcorrection` (
  `redvalue` FLOAT NOT NULL,
  `greenvalue` FLOAT NOT NULL,
  `bluevalue` FLOAT NOT NULL,
  `intensity` FLOAT NOT NULL,
  `correctionaction_installedcorrectors_correctorid` INT NOT NULL,
  `correctionaction_timestampstart` DATETIME NOT NULL,
  PRIMARY KEY (`correctionaction_installedcorrectors_correctorid`, `correct
ionaction_timestampstart`),
  CONSTRAINT `fk_ledcorrection_correctionaction1`
```

```
    FOREIGN KEY (`correctionaction_installedcorrectors_correctorid` , `corr
ectionaction_timestampstart`)
    REFERENCES `smartgreenhouse_2`.`correctionaction` (`installedcorrectors
_correctorid` , `timestampstart`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CorrectionModuleType`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`correctionmoduletype` ;CREATE TAB
LE IF NOT EXISTS `smartgreenhouse_2`.`correctionmoduletype` (
  `modulecode` INT NOT NULL,
  `moduletypename` VARCHAR(40) NULL,
  `moduletypedescription` VARCHAR(100) NULL,
  PRIMARY KEY (`modulecode`))
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CorrectionModule`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`correctionmodule` ;CREATE TABLE I
F NOT EXISTS `smartgreenhouse_2`.`correctionmodule` (
  `moduleid` INT NOT NULL auto_increment,
  `farmregion_regionid` INT NOT NULL,
  `modulename` VARCHAR(20) NULL,
  `correctionmoduletype_modulecode` INT NOT NULL,
  PRIMARY KEY (`moduleid`, `farmregion_regionid`, `correctionmoduletype_mod
ulecode`),
  INDEX `fk_correctionmodule_correctionmoduletype1_idx` (`correctionmodulet
ype_modulecode` asc),
  CONSTRAINT `fk_correctionmodule_farmregion1`
    FOREIGN KEY (`farmregion_regionid`)
    REFERENCES `smartgreenhouse_2`.`farmregion` (`regionid`)
    ON DELETE no action
    ON UPDATE no action,
  CONSTRAINT `fk_correctionmodule_correctionmoduletype1`
    FOREIGN KEY (`correctionmoduletype_modulecode`)
    REFERENCES `smartgreenhouse_2`.`correctionmoduletype` (`modulecode`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`DesiredConditions`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`desiredconditions` ;CREATE TABLE
IF NOT EXISTS `smartgreenhouse_2`.`desiredconditions` (
  `conditionstarttime` DATETIME NOT NULL,
  `correctionmodule_moduleid` INT NOT NULL,
  `upplerlimit` FLOAT NOT NULL,
  `lowerlimit` FLOAT NULL,
  `conditionendtime` DATETIME NULL,
  PRIMARY KEY (`conditionstarttime`, `correctionmodule_moduleid`),
  INDEX `fk_desiredconditions_correctionmodule1_idx` (`correctionmodule_mod
uleid` asc),
  CONSTRAINT `fk_desiredconditions_correctionmodule1`
```

```
    FOREIGN KEY (`correctionmodule_moduleid`)
    REFERENCES `smartgreenhouse_2`.`correctionmodule` (`moduleid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`DesiredLightType`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`desiredlighttype` ;CREATE TABLE I
F NOT EXISTS `smartgreenhouse_2`.`desiredlighttype` (
  `desiredred` FLOAT NOT NULL,
  `desiredgreen` FLOAT NOT NULL,
  `desiredblue` FLOAT NOT NULL,
  `desiredconditions_conditionstarttime` DATETIME NOT NULL,
  `desiredconditions_correctionmodule_moduleid` INT NOT NULL,
  PRIMARY KEY (`desiredconditions_conditionstarttime`, `desiredconditions_c
orrectionmodule_moduleid`),
  CONSTRAINT `fk_desiredlighttype_desiredconditions1`
    FOREIGN KEY (`desiredconditions_conditionstarttime` , `desiredcondition
s_correctionmodule_moduleid`)
    REFERENCES `smartgreenhouse_2`.`desiredconditions` (`conditionstarttime
` , `correctionmodule_moduleid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CropType`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`croptype` ;CREATE TABLE IF NOT EX
ISTS `smartgreenhouse_2`.`croptype` (
  `cropid` INT NOT NULL auto_increment,
  `cropname` VARCHAR(10) NOT NULL,
  `cropdescription` VARCHAR(45) NULL,
  PRIMARY KEY (`cropid`))
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CropInfo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`cropinfo` ;CREATE TABLE IF NOT EX
ISTS `smartgreenhouse_2`.`cropinfo` (
  `cropplantdate` DATETIME NOT NULL,
  `cropharvestdate` DATETIME NULL,
  `croptype_cropid` INT NOT NULL,
  `harvestyield` FLOAT NULL,
  PRIMARY KEY (`croptype_cropid`, `cropplantdate`),
  INDEX `fk_cropinfo_planttype1_idx` (`croptype_cropid` asc),
  CONSTRAINT `fk_cropinfo_planttype1`
    FOREIGN KEY (`croptype_cropid`)
    REFERENCES `smartgreenhouse_2`.`croptype` (`cropid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
```

```
-- Table `SmartGreenhouse_2`.`FarmRegion_has_CropInfo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`farmregion_has_cropinfo` ;CREATE
TABLE IF NOT EXISTS `smartgreenhouse_2`.`farmregion_has_cropinfo` (
  `farmregion_regionid` INT NOT NULL,
  `cropinfo_croptype_cropid` INT NOT NULL,
  `cropinfo_cropplantdate` DATETIME NOT NULL,
  PRIMARY KEY (`farmregion_regionid`, `cropinfo_croptype_cropid`, `cropinfo
_cropplantdate`),
  INDEX `fk_farmregion_has_cropinfo_cropinfo1_idx` (`cropinfo_croptype_crop
id` asc, `cropinfo_cropplantdate` ASC),
  INDEX `fk_farmregion_has_cropinfo_farmregion1_idx` (`farmregion_regionid`
 ASC),
  CONSTRAINT `fk_farmregion_has_cropinfo_farmregion1`
    FOREIGN KEY (`farmregion_regionid`)
    REFERENCES `smartgreenhouse_2`.`farmregion` (`regionid`)
    ON DELETE no action
    ON UPDATE no action,
  CONSTRAINT `fk_farmregion_has_cropinfo_cropinfo1`
    FOREIGN KEY (`cropinfo_croptype_cropid` , `cropinfo_cropplantdate`)
    REFERENCES `smartgreenhouse_2`.`cropinfo` (`croptype_cropid` , `croppla
ntdate`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`LEDCorrector`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`ledcorrector` ;CREATE TABLE IF NO
T EXISTS `smartgreenhouse_2`.`ledcorrector` (
  `redpin` INT NOT NULL,
  `greenpin` INT NOT NULL,
  `bluepin` INT NOT NULL,
  `installedcorrectors_correctorid` INT NOT NULL,
  PRIMARY KEY (`installedcorrectors_correctorid`),
  CONSTRAINT `fk_ledcorrector_installedcorrectors1`
    FOREIGN KEY (`installedcorrectors_correctorid`)
    REFERENCES `smartgreenhouse_2`.`installedcorrectors` (`correctorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`PowerRelay`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`powerrelay` ;CREATE TABLE IF NOT
EXISTS `smartgreenhouse_2`.`powerrelay` (
  `installedsensors_sensorid` INT NOT NULL,
  `relaypin` INT NOT NULL,
  PRIMARY KEY (`installedsensors_sensorid`),
  CONSTRAINT `fk_relay_installedsensors1`
    FOREIGN KEY (`installedsensors_sensorid`)
    REFERENCES `smartgreenhouse_2`.`installedsensors` (`sensorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;
```

```
-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`I2cBusSensor`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`i2cbussensor` ;CREATE TABLE IF NO
T EXISTS `smartgreenhouse_2`.`i2cbussensor` (
  `installedsensors_sensorid` INT NOT NULL,
  `addrfloatorground` TINYINT(1) NOT NULL,
  PRIMARY KEY (`installedsensors_sensorid`),
  CONSTRAINT `fk_relay_installedsensors10`
    FOREIGN KEY (`installedsensors_sensorid`)
    REFERENCES `smartgreenhouse_2`.`installedsensors` (`sensorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`InstalledCorrectors_has_CorrectionModule`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`installedcorrectors_has_correctio
nmodule` ;CREATE TABLE IF NOT EXISTS `smartgreenhouse_2`.`installedcorrecto
rs_has_correctionmodule` (
  `installedcorrectors_correctorid` INT NOT NULL,
  `correctionmodule_moduleid` INT NOT NULL,
  PRIMARY KEY (`installedcorrectors_correctorid`, `correctionmodule_modulei
d`),
  INDEX `fk_installedcorrectors_has_correctionmodule_correctionmodul_idx` (
`correctionmodule_moduleid` asc),
  INDEX `fk_installedcorrectors_has_correctionmodule_installedcorrec_idx` (
`installedcorrectors_correctorid` ASC),
  CONSTRAINT `fk_installedcorrectors_has_correctionmodule_installedcorrecto
1`
    FOREIGN KEY (`installedcorrectors_correctorid`)
    REFERENCES `smartgreenhouse_2`.`installedcorrectors` (`correctorid`)
    ON DELETE no action
    ON UPDATE no action,
  CONSTRAINT `fk_installedcorrectors_has_correctionmodule_correctionmodule1
`
    FOREIGN KEY (`correctionmodule_moduleid`)
    REFERENCES `smartgreenhouse_2`.`correctionmodule` (`moduleid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;


-- -----------------------------------------------------
-- Table `SmartGreenhouse_2`.`CorrectionModule_has_InstalledSensors`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `smartgreenhouse_2`.`correctionmodule_has_installedsen
sors` ;CREATE TABLE IF NOT EXISTS `smartgreenhouse_2`.`correctionmodule_has
_installedsensors` (
  `correctionmodule_moduleid` INT NOT NULL,
  `installedsensors_sensorid` INT NOT NULL,
  PRIMARY KEY (`correctionmodule_moduleid`, `installedsensors_sensorid`),
  INDEX `fk_correctionmodule_has_installedsensors_installedsensors1_idx` (`
installedsensors_sensorid` asc),
  INDEX `fk_correctionmodule_has_installedsensors_correctionmodule1_idx` (`
correctionmodule_moduleid` ASC),
  CONSTRAINT `fk_correctionmodule_has_installedsensors_correctionmodule1`
    FOREIGN KEY (`correctionmodule_moduleid`)
    REFERENCES `smartgreenhouse_2`.`correctionmodule` (`moduleid`)
```

```
    ON DELETE no action
    ON UPDATE no action,
  CONSTRAINT `fk_correctionmodule_has_installedsensors_installedsensors1`
    FOREIGN KEY (`installedsensors_sensorid`)
    REFERENCES `smartgreenhouse_2`.`installedsensors` (`sensorid`)
    ON DELETE no action
    ON UPDATE no action)
engine = innodb;SET sql_mode=@old_sql_mode;SET foreign_key_checks=@old_fore
ign_key_checks;SET unique_checks=@old_unique_checks;
```

PHP code

Multi_Series_Luminosity.php

```
<?php
/* Include the `include/fusioncharts.php` file that contains functions to embed the charts.*/
require  '../menu.php';
include("wrappers 2/php-wrapper/fusioncharts.php");
/* The following 4 code lines contains the database connection information. Alternatively, you can move these code lines to a separate file and include the file here. You can also modify this code based on your database connection.   */
$hostdb = "localhost";  // MySQl host
$userdb = "joseph";  // MySQL username
$passdb = "passcode";  // MySQL password
$namedb = "SmartGreenhouse_2";  // MySQL database name
// Establish a connection to the database
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
/*Render an error message, to avoid abrupt failure, if the database connection parameters are incorrect */
if ($dbhandle->connect_error) {
  exit("There was an error with your connection: ".$dbhandle->connect_error);
}
?>
<html>
<head>
  <title>FusionCharts | Multi-Series Chart using PHP and MySQL</title>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.charts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/themes/fusioncharts.theme.zune.js"></script>
</head>
<body>
<?php
$categoryArray=array();
$arrData = array(
  "chart" => array(
//      "caption"=> "Luminosity readings",
//      "yAxisName"=> "Luminosity in Lux",
//      "legendItemFontColor"=> "#666666",
//      "theme"=> "zune",
//      "drawAnchors"=>"0",
//      "showValues"=>"0"
//      "chart": {
      "caption"=> "Luminosity module",
      "pYAxisName"=> "Luminosity in Lux",
      "sYAxisName"=> "LED light State",
      "xAxisname"=> "Date time",
      "compactDataMode"=> "1",
      "pixelsPerPoint"=> "0",
      "lineThickness"=> "1",
//      "dataSeparator"=> "|",
      "sYAxisMaxValue"=> "2",
      "sYAxisMinValue"=> "0",
      "theme"=> "zune"
  )
);
$strXAxisQuery = "SELECT   TimeStampStart AS `TimeStamp` from   CorrectionAction UNION ALL SELECT   TimeStampEnd AS `TimeStamp` from   CorrectionAction WHERE InstalledCorrectors_CorrectorID = 3        UNION ALL      SELECT     `TimeStamp`    FROM SensorReadings    WHERE     InstalledSensors_SensorID = '2'     OR InstalledSensors_SensorID = '3'
```

```
UNION ALL  SELECT  DISTINCT *  FROM  (  SELECT  ConditionStartTime  from
DesiredConditions  WHERE  CorrectionModule_ModuleID = 3  UNION ALL  SELECT
ConditionEndTime  from  DesiredConditions  WHERE  CorrectionModule_ModuleID = 3
) correctionTimes order by  `TimeStamp` ASC";
$result = $dbhandle->query($strXAxisQuery.";") or exit("Error code ({$dbhandle->errno}): {$dbhandle-
>error}");
if ($result) {
  // pushing category array values
  $count = 0;
  while($row = mysqli_fetch_array($result)) {
     array_push($categoryArray, array(
        "label" => $row["TimeStamp"]));
  }
}
$sensor1Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strXAxisQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '2'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["SensorValue"] === null){
        array_push($sensor1Readings, array("value" => ((int)$lastValue )));
     }
     else{
        array_push($sensor1Readings, array("value" => ((int)$row["SensorValue"])));
     }
     $lastValue = $row["SensorValue"];
  }
}
$sensor2Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strXAxisQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
```

```
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '3'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["SensorValue"] === null){
        array_push($sensor2Readings, array("value" => ((int)$lastValue )));
     }
     else{
        array_push($sensor2Readings, array("value" => ((int)$row["SensorValue"])));
     }
     $lastValue = $row["SensorValue"];
  }
};
$correctionActions=array();
$strQuery = "SELECT
 caOn.TimeStampStart AS CorrectionStart,
 caOff.TimeStampEnd AS CorrectionEnd
FROM
 (
  (".$strXAxisQuery.") xAxis
  LEFT JOIN (
    SELECT
      *
    FROM
      CorrectionAction
    WHERE
      InstalledCorrectors_CorrectorID = '3'
  ) caOn ON xAxis.`TimeStamp` = caOn.TimeStampStart
 )
 LEFT JOIN (
  SELECT
    *
  FROM
    CorrectionAction
  WHERE
    InstalledCorrectors_CorrectorID = '3'
 ) caOff ON xAxis.`TimeStamp` = caOff.TimeStampEnd;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["CorrectionStart"] !== null){
        array_push($CorrectionActions, array("value" => 1 ));
        $lastValue = 1;
     }
     elseif ($row["CorrectionEnd"] !== null){
        array_push($correctionActions, array("value" => 0));
```

```
        $lastValue = 0;
    }
    else{
        array_push($correctionActions, array("value" => $lastValue));
    }
  }
};
$desiredLowerLimit=array();
$desiredUpperLimit=array();
$strQuerry1 = "
SELECT
 conditionOff.ConditionEndTime AS condition_end,
 conditionOn.ConditionStartTime AS condition_start,
 conditionOn.LowerLimit AS lower_limit,
 conditionOn.UpperLimit AS upper_limit
FROM
 (
  xAxis
  LEFT JOIN (
   SELECT
    dc.ConditionStartTime,
    dc.LowerLimit,
    dc.UpperLimit
   From
    DesiredConditions dc
   WHERE
    CorrectionModule_ModuleID = '3'
  ) conditionOn ON xAxis.`TimeStamp` = conditionOn.ConditionStartTime
 )
 LEFT JOIN (
  SELECT
   dc1.ConditionEndTime
  From
   DesiredConditions dc1
  WHERE
   CorrectionModule_ModuleID = 3
 ) conditionOff ON xAxis.`TimeStamp` = conditionOff.ConditionEndTime
order by
 `TimeStamp` asc;
";
$result1 = $dbhandle->query($strQuerry1) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result1) {
  $upperValue=0;
  $lowerValue=0;
  while($row = mysqli_fetch_array($result1)) {
    if ($row["condition_start"] !== null){
        array_push($desiredLowerLimit, array("value" => $row["lower_limit"]));
        array_push($desiredUpperLimit, array("value" => $row["upper_limit"]));
        $upperValue=$row["upper_limit"];
        $lowerValue=$row["lower_limit"];
    }
    elseif ($row["condition_end"] !== null){
        array_push($correctionActions, array("value" => 0));
        $upperValue=0;
        $lowerValue=0;
    }
    else{
        array_push($desiredLowerLimit, array("value" => $lowerValue ));
```

```
        array_push($desiredUpperLimit, array("value" => $upperValue ));
    }
  }
};
$arrData["categories"]=array(array("category"=>$categoryArray));
// creating dataset object
$arrData["dataset"] = array(
    array(
        "seriesName"=> "External light level",
        "parentyaxis"=> "p",
        "renderAs"=>"line",
        "data"=>$sensor1Readings),
    array(
        "seriesName"=> "Internal light level",
        "parentyaxis"=> "P",
        "renderAs"=>"line",
        "data"=>$sensor2Readings),
    array(
        "seriesName"=> "Desired upper limit",
        "parentyaxis"=> "p",
        "renderAs"=>"line",
        "data"=>$desiredUpperLimit),
    array(
        "seriesName"=> "Desired lower limit",
        "parentyaxis"=> "p",
        "renderAs"=>"line",
        "data"=>$desiredLowerLimit),
    array(
        "seriesName"=> "Lights On",
        "parentyaxis"=> "S",
        "renderAs"=>"line",
        "data"=>$correctionActions)
);
/*JSON Encode the data to retrieve the string containing the JSON representation of the data in the array. */
$jsonEncodedData = json_encode($arrData);
//echo $jsonEncodedData;
// chart object
$msChart = new FusionCharts(
    "zoomlinedy",
    "chart1" ,
    1000, 600,
    "chart-container",
    "json",
    $jsonEncodedData);
// Render the chart
$msChart->render();
$dbhandle->close();
?>
<center>
  <div id="chart-container">Chart will render here!</div>
</center>
</body>
</html>
```

Multi_Series_Temperature.php

```
<?php
/* Include the `include/fusioncharts.php` file that contains functions to embed the charts.*/
```

```
require  '../menu.php';
include("wrappers 2/php-wrapper/fusioncharts.php");
/* The following 4 code lines contains the database connection information. Alternatively, you can move these code lines to a separate file and include the file here. You can also modify this code based on your database connection.   */
$hostdb = "localhost";  // MySQl host
$userdb = "joseph";  // MySQL username
$passdb = "passcode";  // MySQL password
$namedb = "SmartGreenhouse_2";  // MySQL database name
// Establish a connection to the database
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
/*Render an error message, to avoid abrupt failure, if the database connection parameters are incorrect */
if ($dbhandle->connect_error) {
   exit("There was an error with your connection: ".$dbhandle->connect_error);
}
?>
<html>
<head>
   <title>FusionCharts | Multi-Series Chart using PHP and MySQL</title>
   <script src="http://static.fusioncharts.com/code/latest/fusioncharts.js"></script>
   <script src="http://static.fusioncharts.com/code/latest/fusioncharts.charts.js"></script>
   <script src="http://static.fusioncharts.com/code/latest/themes/fusioncharts.theme.zune.js"></script>
</head>
<body>
<?php
$categoryArray=array();
$arrData = array(
   "chart" => array(
      "caption"=> "Temperature module",
      "pYAxisName"=> "Temperature in degrees celsius",
      "sYAxisName"=> "Fan State (1 is on and 0 is off)",
      "xAxisname"=> "Date time",
      "compactDataMode"=> "1",
      "pixelsPerPoint"=> "0",
      "lineThickness"=> "1",
  "pYAxisMaxValue"=> "30",
      "pYAxisMinValue"=> "10",
  "forceYAxisValueDecimals" =>"1",
  "forcesYAxisValueDecimals" =>"0",
  "sYAxisValueDecimals" =>"0",
      "sYAxisMaxValue"=> "1.5",
      "sYAxisMinValue"=> "0",
      "theme"=> "zune"
   )
);
$strXAxisQuery = "SELECT   TimeStampStart AS `TimeStamp` from   CorrectionAction UNION ALL SELECT   TimeStampEnd AS `TimeStamp` from   CorrectionAction WHERE InstalledCorrectors_CorrectorID = '4'        UNION ALL      SELECT      `TimeStamp`    FROM SensorReadings    WHERE      InstalledSensors_SensorID = '1'     OR InstalledSensors_SensorID = '6' UNION ALL    SELECT      DISTINCT *    FROM      (       SELECT         ConditionStartTime        from DesiredConditions        WHERE         CorrectionModule_ModuleID = '4'       UNION ALL        SELECT ConditionEndTime        from         DesiredConditions        WHERE         CorrectionModule_ModuleID = '4' ) correctionTimes order by  `TimeStamp` ASC";
$result = $dbhandle->query($strXAxisQuery.";") or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
   // pushing category array values
   $count = 0;
```

```
  while($row = mysqli_fetch_array($result)) {
      array_push($categoryArray, array(
          "label" => $row["TimeStamp"]));
  }
}
$sensor1Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strXAxisQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '1'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
      if ($row["SensorValue"] === null or $row["SensorValue"] == 0){
          array_push($sensor1Readings, array("value" => ((float)$lastValue )));
      }
      else{
          array_push($sensor1Readings, array("value" => ((float)$row["SensorValue"])));
          $lastValue = $row["SensorValue"];
      }
  }
}
$dbhandle->close();
$sensor2Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strXAxisQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '6'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
```

```
 sr1.`TimeStamp` ASC;
";
$dbhandle->close();
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["SensorValue"] === null or $row["SensorValue"] == 0){
        array_push($sensor2Readings, array("value" => ((float)$lastValue )));
     }
     else{
        array_push($sensor2Readings, array("value" => ((float)$row["SensorValue"])));
        $lastValue = $row["SensorValue"];
     }
  }
};
$correctionActions=array();
$dbhandle->close();
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$strQuery = "SELECT
 caOn.TimeStampStart AS CorrectionStart,
 caOff.TimeStampEnd AS CorrectionEnd
FROM
 (
  xAxisTemp
  LEFT JOIN (
   SELECT
    *
   FROM
    CorrectionAction
   WHERE
    InstalledCorrectors_CorrectorID = '4'
  ) caOn ON xAxisTemp.`TimeStamp` = caOn.TimeStampStart
 )
 LEFT JOIN (
  SELECT
   *
  FROM
   CorrectionAction
  WHERE
   InstalledCorrectors_CorrectorID = '4'
 ) caOff ON xAxisTemp.`TimeStamp` = caOff.TimeStampEnd;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["CorrectionStart"] !== null){
        array_push($CorrectionActions, array("value" => 1 ));
        $lastValue = 1;
     }
     elseif ($row["CorrectionEnd"] !== null){
        array_push($correctionActions, array("value" => 0));
        $lastValue = 0;
     }
     else{
        array_push($correctionActions, array("value" => $lastValue));
```

```
    }
  }
};
$dbhandle->close();
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$desiredLowerLimit=array();
$desiredUpperLimit=array();
$strQuerry1 = "
SELECT
 conditionOff.ConditionEndTime AS condition_end,
 conditionOn.ConditionStartTime AS condition_start,
 conditionOn.LowerLimit AS lower_limit,
 conditionOn.UpperLimit AS upper_limit
FROM
 (
  xAxisTemp
  LEFT JOIN (
   SELECT
    dc.ConditionStartTime,
    dc.LowerLimit,
    dc.UpperLimit
   From
    DesiredConditions dc
   WHERE
    CorrectionModule_ModuleID = '4'
  ) conditionOn ON xAxisTemp.`TimeStamp` = conditionOn.ConditionStartTime
 )
 LEFT JOIN (
  SELECT
   dc1.ConditionEndTime
  From
   DesiredConditions dc1
  WHERE
   CorrectionModule_ModuleID = '4'
 ) conditionOff ON xAxisTemp.`TimeStamp` = conditionOff.ConditionEndTime
order by
 `TimeStamp` asc;
";
$result1 = $dbhandle->query($strQuerry1) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result1) {
  $upperValue=0;
  $lowerValue=0;
  while($row = mysqli_fetch_array($result1)) {
     if ($row["condition_start"] !== null){
        array_push($desiredLowerLimit, array("value" => $row["lower_limit"]));
        array_push($desiredUpperLimit, array("value" => $row["upper_limit"]));
        $upperValue=$row["upper_limit"];
        $lowerValue=$row["lower_limit"];
     }
     elseif ($row["condition_end"] !== null){
        array_push($correctionActions, array("value" => 0));
        $upperValue=0;
        $lowerValue=0;
     }
     else{
        array_push($desiredLowerLimit, array("value" => $lowerValue ));
        array_push($desiredUpperLimit, array("value" => $upperValue ));
     }
```

```
  }
};
$arrData["categories"]=array(array("category"=>$categoryArray));
// creating dataset object
$arrData["dataset"] = array(
      array(
         "seriesName"=> "ExternalTemperature Degrees celsius",
         "parentyaxis"=> "p",
         "renderAs"=>"line",
         "data"=>$sensor1Readings),
      array(
         "seriesName"=> "Internal  Temperature Degrees celsius",
         "parentyaxis"=> "P",
         "renderAs"=>"line",
         "data"=>$sensor2Readings),
      array(
         "seriesName"=> "Desired upper temp",
         "parentyaxis"=> "p",
         "renderAs"=>"line",
         "data"=>$desiredUpperLimit),
      array(
         "seriesName"=> "Desired lower temp",
         "parentyaxis"=> "p",
         "renderAs"=>"line",
         "data"=>$desiredLowerLimit),
      array(
         "seriesName"=> "Fan On",
         "parentyaxis"=> "S",
         "renderAs"=>"line",
         "data"=>$correctionActions)
);
/*JSON Encode the data to retrieve the string containing the JSON representation of the data in the array. */
$jsonEncodedData = json_encode($arrData);
//echo $jsonEncodedData;
// chart object
$msChart = new FusionCharts(
      "zoomlinedy",
      "chart1" ,
      1000, 600,
      "chart-container",
      "json",
      $jsonEncodedData);
// Render the chart
$msChart->render();
$dbhandle->close();
?>
<center>
   <div id="chart-container">Chart will render here!</div>
</center>
</body>
</html>
```

Multi_Series_Humidity.php

```
<?php
/* Include the `include/fusioncharts.php` file that contains functions to embed the charts.*/
require  '../menu.php';
include("wrappers 2/php-wrapper/fusioncharts.php");
```

```
/* The following 4 code lines contains the database connection information. Alternatively, you can move these code lines to a separate file and include the file here. You can also modify this code based on your database connection.  */
$hostdb = "localhost";  // MySQl host
$userdb = "joseph";  // MySQL username
$passdb = "passcode";  // MySQL password
$namedb = "SmartGreenhouse_2";  // MySQL database name
// Establish a connection to the database
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
/*Render an error message, to avoid abrupt failure, if the database connection parameters are incorrect */
if ($dbhandle->connect_error) {
  exit("There was an error with your connection: ".$dbhandle->connect_error);
}
?>
<html>
<head>
  <title>FusionCharts | Multi-Series Chart using PHP and MySQL</title>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.charts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/themes/fusioncharts.theme.zune.js"></script>
</head>
<body>
<?php
$categoryArray=array();
$arrData = array(
  "chart" => array(
"caption"=> "Humidity module",
      "pYAxisName"=> "Relative Humidty % of water air saturation",
      "sYAxisName"=> "Fan State (1 is on and 0 is off)",
      "xAxisHumidname"=> "Date time",
      "pixelsPerPoint"=> "0",
      "lineThickness"=> "1",
       "pAxisMinValue"=>"0",
      "pAxisMaxValue"=>"100",
      "sYAxisMinValue"=> "0",
      "theme"=> "zune"
  )
);
$strxAxisHumidQuery = "
SELECT
 TimeStampStart AS `TimeStamp`
 from  CorrectionAction UNION ALL SELECT  TimeStampEnd AS `TimeStamp` from  CorrectionAction
WHERE
InstalledCorrectors_CorrectorID = '4'       UNION ALL      SELECT     `TimeStamp`
FROM     SensorReadings   WHERE     InstalledSensors_SensorID = '7'     OR
InstalledSensors_SensorID = '8'
  UNION ALL   SELECT     DISTINCT *   FROM     (      SELECT        ConditionStartTime
from        DesiredConditions       WHERE        CorrectionModule_ModuleID = '4'       UNION ALL
     SELECT        ConditionEndTime       from        DesiredConditions       WHERE
  CorrectionModule_ModuleID = '4'     ) correctionTimes order by  `TimeStamp` ASC";
$result = $dbhandle->query($strxAxisHumidQuery.";") or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $count = 0;
  while($row = mysqli_fetch_array($result)) {
     array_push($categoryArray, array(
        "label" => $row["TimeStamp"]));
```

```
  }
}
$sensor1Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strxAxisHumidQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '7'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
      if ($row["SensorValue"] === null or (float)$row["SensorValue"] < 0){
          array_push($sensor1Readings, array("value" => ((float)$lastValue )));
      }
      else{
          array_push($sensor1Readings, array("value" => ((float)$row["SensorValue"])));
          $lastValue = $row["SensorValue"];
      }
  }
}
$dbhandle->close();
$sensor2Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strxAxisHumidQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '8'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$dbhandle->close();
```

```
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["SensorValue"] === null or (float)$row["SensorValue"] < 0){
        array_push($sensor2Readings, array("value" => ((float)$lastValue )));
     }
     else{
        array_push($sensor2Readings, array("value" => ((float)$row["SensorValue"])));
        $lastValue = $row["SensorValue"];
     }
  }
};
$correctionActions=array();
$strQuery = "SELECT
 caOn.TimeStampStart AS CorrectionStart,
 caOff.TimeStampEnd AS CorrectionEnd
FROM
 (
  xAxisHumid
  LEFT JOIN (
   SELECT
    *
   FROM
    CorrectionAction
   WHERE
    InstalledCorrectors_CorrectorID = '4'
  ) caOn ON xAxisHumid.`TimeStamp` = caOn.TimeStampStart
 )
 LEFT JOIN (
  SELECT
   *
  FROM
   CorrectionAction
  WHERE
   InstalledCorrectors_CorrectorID = '4'
 ) caOff ON xAxisHumid.`TimeStamp` = caOff.TimeStampEnd;
";
$dbhandle->close();
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["CorrectionStart"] !== null){
        array_push($CorrectionActions, array("value" => 1 ));
        $lastValue = 1;
     }
     elseif ($row["CorrectionEnd"] !== null){
        array_push($correctionActions, array("value" => 0));
        $lastValue = 0;
     }
     else{
        array_push($correctionActions, array("value" => $lastValue));
     }
  }
};
```

```
$dbhandle->close();
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
$arrData["categories"]=array(array("category"=>$categoryArray));
// creating dataset object
$arrData["dataset"] = array(
      array(
         "seriesName"=> "Internal Relative Humidity",
         "parentyaxis"=> "p",
         "renderAs"=>"line",
         "data"=>$sensor1Readings),
      array(
         "seriesName"=> "External Relative Humidity",
         "parentyaxis"=> "P",
         "renderAs"=>"line",
         "data"=>$sensor2Readings),
      array(
         "seriesName"=> "Fan On",
         "parentyaxis"=> "S",
         "renderAs"=>"line",
         "data"=>$correctionActions)
);
/*JSON Encode the data to retrieve the string containing the JSON representation of the data in the array. */
$jsonEncodedData = json_encode($arrData);
//echo $jsonEncodedData;
// chart object
$msChart = new FusionCharts(
      "zoomlinedy",
      "chart1" ,
      1000, 600,
      "chart-container",
      "json",
      $jsonEncodedData);
// Render the chart
$msChart->render();
$dbhandle->close();
?>
<center>
   <div id="chart-container">Chart will render here!</div>
</center>
</body>
</html>
```

Multi_Series_Water_Module.php

```
<?php
/* Include the `include/fusioncharts.php` file that contains functions to embed the charts.*/
require  '../menu.php';
include("wrappers 2/php-wrapper/fusioncharts.php");
/* The following 4 code lines contains the database connection information. Alternatively, you can move these
code lines to a separate file and include the file here. You can also modify this code based on your database
connection.   */
$hostdb = "localhost";  // MySQl host
$userdb = "joseph";  // MySQL username
$passdb = "passcode";  // MySQL password
$namedb = "SmartGreenhouse_2";  // MySQL database name
// Establish a connection to the database
$dbhandle = new mysqli($hostdb, $userdb, $passdb, $namedb);
/*Render an error message, to avoid abrupt failure, if the database connection parameters are incorrect */
```

```
if ($dbhandle->connect_error) {
  exit("There was an error with your connection: ".$dbhandle->connect_error);
}
?>
<html>
<head>
  <title>FusionCharts | Multi-Series Chart using PHP and MySQL</title>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/fusioncharts.charts.js"></script>
  <script src="http://static.fusioncharts.com/code/latest/themes/fusioncharts.theme.zune.js"></script>
</head>
<body>
<?php
$categoryArray=array();
$arrData = array(
  "chart" => array(
     "caption"=> "Water Module",
     "pYAxisName"=> "Reservoir Volume in ml",
     "sYAxisName"=> "Pump state (1 = means on and 0 means off)",
     "xAxisname"=> "Date time",
     "compactDataMode"=> "1",
     "pixelsPerPoint"=> "0",
     "lineThickness"=> "1",
     "sYAxisMaxValue"=> "2",
     "sYAxisMinValue"=> "0",
     "theme"=> "zune"
  )
);
$strXAxisQuery = "
SELECT
 TimeStampStart AS `TimeStamp`
from
 CorrectionAction
UNION ALL
SELECT
 TimeStampEnd AS `TimeStamp`
from
 CorrectionAction
WHERE
 InstalledCorrectors_CorrectorID = '2'
UNION ALL
SELECT
 `TimeStamp`
FROM
 SensorReadings
WHERE
 InstalledSensors_SensorID = '5'
UNION ALL
SELECT
 DISTINCT *
FROM
 (
  SELECT
    ConditionStartTime
  from
    DesiredConditions
  WHERE
    CorrectionModule_ModuleID = 2
```

```
  UNION ALL
  SELECT
    ConditionEndTime
  from
    DesiredConditions
  WHERE
    CorrectionModule_ModuleID = 2
 ) correctionTimes
order by
 `TimeStamp` ASC
";
$result = $dbhandle->query($strXAxisQuery.";") or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $count = 0;
  while($row = mysqli_fetch_array($result)) {
      array_push($categoryArray, array(
          "label" => $row["TimeStamp"]));
  }
}
$sensor1Readings=array();
$strQuery = "
SELECT
 SensorValue
FROM
 (
  (".$strXAxisQuery.") sr1
  LEFT JOIN (
    SELECT
      `TimeStamp`,
      SensorValue
    FROM
      SensorReadings
    WHERE
      InstalledSensors_SensorID = '5'
  ) sr2 ON sr1.`TimeStamp` = sr2.`TimeStamp`
 )
ORDER BY
 sr1.`TimeStamp` ASC;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  // pushing category array values
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
      if ($row["SensorValue"] === null){
          array_push($sensor1Readings, array("value" => ((float)$lastValue )));
      }
      else{
          array_push($sensor1Readings, array("value" => ((float)$row["SensorValue"])));
      }
      $lastValue = $row["SensorValue"];
  }
}
$correctionActions=array();
$strQuery = "SELECT
 caOn.TimeStampStart AS CorrectionStart,
```

```
 caOff.TimeStampEnd AS CorrectionEnd
FROM
 (
  (".$strXAxisQuery.") xAxis
  LEFT JOIN (
    SELECT
      *
    FROM
      CorrectionAction
    WHERE
      InstalledCorrectors_CorrectorID = '2'
  ) caOn ON xAxis.`TimeStamp` = caOn.TimeStampStart
 )
 LEFT JOIN (
  SELECT
    *
  FROM
    CorrectionAction
  WHERE
    InstalledCorrectors_CorrectorID = '2'
 ) caOff ON xAxis.`TimeStamp` = caOff.TimeStampEnd;
";
$result = $dbhandle->query($strQuery) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result) {
  $lastValue=0;
  while($row = mysqli_fetch_array($result)) {
     if ($row["CorrectionStart"] !== null){
        array_push($CorrectionActions, array("value" => 1 ));
        $lastValue = 1;
     }
     elseif ($row["CorrectionEnd"] !== null){
        array_push($correctionActions, array("value" => 0));
        $lastValue = 0;
     }
     else{
        array_push($correctionActions, array("value" => $lastValue));
     }
  }
};
$desiredLowerLimit=array();
$desiredUpperLimit=array();
$strQuerry1 = "
SELECT
 conditionOff.ConditionEndTime AS condition_end,
 conditionOn.ConditionStartTime AS condition_start,
 conditionOn.LowerLimit AS lower_limit,
 conditionOn.UpperLimit AS upper_limit
FROM
 (
  xAxis
  LEFT JOIN (
    SELECT
      dc.ConditionStartTime,
      dc.LowerLimit,
      dc.UpperLimit
    From
      DesiredConditions dc
    WHERE
```

```
      CorrectionModule_ModuleID = '3'
  ) conditionOn ON xAxis.`TimeStamp` = conditionOn.ConditionStartTime
 )
 LEFT JOIN (
  SELECT
    dc1.ConditionEndTime
  From
    DesiredConditions dc1
  WHERE
    CorrectionModule_ModuleID = 2
 ) conditionOff ON xAxis.`TimeStamp` = conditionOff.ConditionEndTime
order by
 `TimeStamp` asc;
";
$result1 = $dbhandle->query($strQuerry1) or exit("Error code ({$dbhandle->errno}): {$dbhandle->error}");
if ($result1) {
  $upperValue=0;
  $lowerValue=0;
  while($row = mysqli_fetch_array($result1)) {
     if ($row["condition_start"] !== null){
        array_push($desiredLowerLimit, array("value" => $row["lower_limit"]));
        array_push($desiredUpperLimit, array("value" =>  $row["upper_limit"]));
        $upperValue=$row["upper_limit"];
        $lowerValue=$row["lower_limit"];
     }
     elseif ($row["condition_end"] !== null){
        array_push($correctionActions, array("value" => 0));
        $upperValue=0;
        $lowerValue=0;
     }
     else{
        array_push($desiredLowerLimit, array("value" => $lowerValue ));
        array_push($desiredUpperLimit, array("value" => $upperValue ));
     }
  }
};
$arrData["categories"]=array(array("category"=>$categoryArray));
// creating dataset object
$arrData["dataset"] = array(
     array(
        "seriesName"=> "Water Level reading",
        "parentyaxis"=> "p",
        "renderAs"=>"line",
        "data"=>$sensor1Readings),
     array(
        "seriesName"=> "Absolute lowest limit for pump to run",
        "parentyaxis"=> "p",
        "renderAs"=>"line",
        "data"=>$desiredLowerLimit),
     array(
        "seriesName"=> "Pump On",
        "parentyaxis"=> "S",
        "renderAs"=>"line",
        "data"=>$correctionActions)
);
/*JSON Encode the data to retrieve the string containing the JSON representation of the data in the array. */
$jsonEncodedData = json_encode($arrData);
//echo $jsonEncodedData;
```

```
// chart object
$msChart = new FusionCharts(
      "zoomlinedy",
      "chart1" ,
      800, 400,
      "chart-container",
      "json",
      $jsonEncodedData);
// Render the chart
$msChart->render();
$dbhandle->close();
?>
<center>
   <div id="chart-container">Chart will render here!</div>
</center>
</body>
</html>
```

smart_farm_sing_up.php

```
<?php
session_start();
require 'sql_query_and_insertion.php';
require 'useful_functions.php';
?>
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
// define variables and set to empty values
$farmerEmail = $farmerName = $farmerPassword = $farmerPassword1 = $farmerPassword2 = "";
$nameErr = $emailErr = $passwordErr = $tempPassword ="";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 if (empty($_POST["farmerName"])) {
   $nameErr = "Name is required";
 } else {
   $farmerName = test_input($_POST["farmerName"]);
   // check if farmerName only contains letters and whitespace
   if (!preg_match("/^[a-zA-Z ]*$/",$farmerName)) {
     $nameErr = "Only letters and white space allowed";
   }
 }
 if (empty($_POST["farmerEmail"])) {
   $emailErr = "Email is required";
 } else {
   $farmerEmail = test_input($_POST["farmerEmail"]);
   // check if e-mail address is well-formed
   if (!filter_var($farmerEmail, FILTER_VALIDATE_EMAIL)) {
     $emailErr = "Invalid email format";
   }
 }
 if (empty($_POST["farmerPassword1"])) {
   $passwordErr = "Password is required";
```

```
} else {
 $tempPassword = preg_replace('/\s+/', '', test_input($_POST["farmerPassword1"]));
 if($_POST["farmerPassword1"] == $tempPassword){
       if ($farmerPassword1 == $farmerPassword2){
         //echo password_hash($tempPassword, PASSWORD_BCRYPT);
         $farmerPassword = password_hash($tempPassword, PASSWORD_BCRYPT);
       }else{
         $passwordErr = $passwordErr = "Passwords must match";
        }
 }else{
       $passwordErr = "No white spaces allowed";
 }
}
//echo "Ready to write to db";
if (($nameErr=="") && ($emailErr=="") && ($passwordErr=="")){

 echo "Ready to write to db<br>";
 //Add the submitted data to the Farmer tuple in SmartGreenhouse SQL db

 try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
 echo "Connection made<br>";

    // prepare sql and bind parameters
    $insertIntoFarmers = "INSERT INTO Farmer (FarmerName, FarmerEmail, farmerPassword)
    VALUES (:farmerName, :farmerEmail, :farmerPassword)";

    $stmt = $conn->prepare($insertIntoFarmers);

    echo "Insert Prepared<br>";

    $stmt->bindParam(':farmerName', $farmerName);
    $stmt->bindParam(':farmerEmail', $farmerEmail);
    $stmt->bindParam(':farmerPassword', $farmerPassword);

    echo "parameters bound<br>";

    $stmt->execute();

    echo "Insert Executed<br>";

    $last_id = $conn->lastInsertId();
    echo "New record created successfully. Last inserted ID is: " . $last_id;

 $_SESSION["farmerID"] = $last_id;
 $_SESSION["farmerName"] = $farmerName;
 header('location: /smart_farm/welcome_page.php');

 }
  catch(PDOException $e)
    {
    echo "Error: " . $e->getMessage();
    }
 $conn = null;

}
```

```
}
?>
<h2>Farmer sign-up form</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
 Name: <input type="text" name="farmerName" value="<?php echo $farmerName;?>">
 <span class="error">* <?php echo $nameErr;?></span>
 <br><br>
 E-mail: <input type="text" name="farmerEmail" value="<?php echo $farmerEmail;?>">
 <span class="error">* <?php echo $emailErr;?></span>
 <br><br>
 Password: <input type="password"  name="farmerPassword1" value="<?php echo
$farmerPassword1;?>">
 <span class="error">* <?php echo $passwordErr;?></span>
 <br><br>
 Re-enter password: <input type="password"  name="farmerPassword2" value="<?php echo
$farmerPassword2;?>">
 <span class="error">* <?php echo $passwordErr;?></span>
 <br><br>
 <input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

smart_farm_login.php

```
<?php
session_start();
require 'sql_query_and_insertion.php';
require 'useful_functions.php';
$_SESSION["farmerID"] = 0;
$_SESSION["farmerName"] ="";
?>
<!DOCTYPE HTML>
<html>
<head>
  <style>
      .error {color: #FF0000;}
  </style>
</head>
<body>
<?php
// define variables and set to empty values
$farmerEmail = $farmerPasswordInput = "";
$emailErr = $passwordErr ="";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  switch ($_POST["submit"]) {
     case "Sign-in":
        if (empty($_POST["farmerPasswordInput"])) {
           $passwordErr = "Password is required";
        }
        if (empty($_POST["farmerEmail"])) {
           $emailErr = "Email is required";
        }
        //echo "Ready to write to db";
        if (($emailErr=="") && ($passwordErr=="")){
           $farmerEmail = test_input($_POST["farmerEmail"]);
           $farmerPasswordInput = test_input($_POST["farmerPasswordInput"]);
```

```
            echo "Ready to get user info from db<br>";
            //Get the users data corresponding to the given email from the Farmer table in SmartGreenhouse SQL
db
            try {
                $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
                // set the PDO error mode to exception
                $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
                echo "Connection made<br>";
                // prepare sql and bind parameters
                echo "SQL Select statement made<br>";
                $stmt = $conn->prepare($selectAllFromFarmerWHereEmail);
                echo "parameters Prepared<br>";
                $stmt->execute([':farmerEmail' => $farmerEmail]);
                echo "querry executed <br>";
                while ($userInfo = $stmt->fetch(PDO::FETCH_ASSOC)){
                    $_SESSION["farmerID"] = $userInfo[farmerID];
                    $_SESSION["farmerName"] = $userInfo[FarmerName];
                    $hash = $userInfo[farmerPassword];
                }
                if (password_verify($farmerPasswordInput, $hash)){
                    header('location: /smart_farm/welcome_page.php');
                }else{
                    $emailErr = $passwordErr = "password & email do not match";
                }
            }
            catch(PDOException $e)
            {
                echo "Error: " . $e->getMessage();
            }
            $conn = null;
        }
        break;
    case "sign-up":
        header('location: /smart_farm/smart_farm_sign-up.php');
        break;
  }
}
?>
<h2>Farmer sign-in form</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  E-mail: <input type="text" name="farmerEmail" value="<?php echo $farmerEmail;?>">
  <span class="error">* <?php echo $emailErr;?></span>
  <br><br>
  Password: <input type="password"  name="farmerPasswordInput" value="<?php echo
$farmerPasswordInput;?>">
  <span class="error">* <?php echo $passwordErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Sign-in">
</form>
<br><br>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  <input type="submit" name="submit" value="sign-up">
</form>
</body>
</html>
```

welcome_page.php

```
<?php
```

```
session_start();
include 'menu.php';
include 'sql_query_and_insertion.php';
include 'useful_functions.php';
?>
<!DOCTYPE HTML>
<html>
<head>
<h1>Welcome to your smart farm interface</h1>
</head>
<body>
<?php
echo "Hello, " . $_SESSION["farmerName"] . ". <br>Your ID is " . $_SESSION["farmerID"];
//get_sensor_data('1', '2018-02-13 09:20:00');
//get_last_sensor_data('1');
?>
</body>
</html>
```

menu.php

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}
li {
  float: left;
}
li a, .dropbtn {
  display: inline-block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}
li a:hover, .dropdown:hover .dropbtn {
  background-color: #69ff33;
}
li.dropdown {
  display: inline-block;
}
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
```

```
}
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
  text-align: left;
}
.dropdown-content a:hover {background-color: #f1f1f1}
.dropdown:hover .dropdown-content {
  display: block;
}
</style>
</head>
<body>
<ul>
 <li><a href="/smart_farm/welcome_page.php">Home</a></li>
  <li class="dropdown">
     <a href="javascript:void(0)" class="dropbtn">Data</a>
     <div class="dropdown-content">
        <a href="/smart_farm/data/Multi_Series_Luminosity.php">Light Module</a>
        <a href="/smart_farm/data/Multi_Series_Temperature.php">Temperature Module</a>
        <a href="/smart_farm/data/Multi_Series_Humidity.php">Humidity Module</a>
        <a href="/smart_farm/data/Multi_Series_Water_Module.php">Water Module</a>
        <a href="">pH Module</a>
     </div>
 <li><a href="Add_Installed_Sensors_2.php">Settings</a></li>
 <li style="float:right"><a href="/smart_farm/smart_farm_login.php">Log Out</a></li>
 <li class="dropdown">
  <a href="javascript:void(0)" class="dropbtn">Add/Edit hardware config</a>
  <div class="dropdown-content">
    <a href="/smart_farm/edit_hardware/add_farm.php">Add Farm</a>
    <a href="/smart_farm/edit_hardware/edit_region.php">Add/ Edit Region</a>
    <a href="/smart_farm/edit_hardware/edit_sensors.php">Add/ Edit Sensors</a>
    <a href="/smart_farm/edit_hardware/edit_correctors.php">Add/ Edit Correctors</a>
    <a href="/smart_farm/edit_hardware/edit_modules.php">Add/ Edit Modules</a>
  </div>
 <li class="dropdown">
  <a href="javascript:void(0)" class="dropbtn">Regions</a>
  <div class="dropdown-content">
    <a href="#">Region 1</a>
    <a href="#">Region 2</a>
    <a href="#">Region 3</a>
  </div>
 </li>
 <li style="float:right"><a class="active"><?php echo $_SESSION["farmerName"] ?></a></li>
</ul>
</body>
</html>
```

useful_functions.php

```
<?php
function test_input($data) {
 $data = trim($data);
 $data = stripslashes($data);
 $data = htmlspecialchars($data);
 return $data;
```

```
}
function get_db_connection(){
  require 'sql_query_and_insertion.php';
  try{
    //echo "About to attempt to connect<br>";
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    //echo "Connected<br>";
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    //echo "Set up <br>";
  return $conn;
  }
  catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
    return NULL;
  }
}
function get_sensor_data($sensor_id, $start_time){
  echo "Getting sensor data<br>";
  require 'sql_query_and_insertion.php';


  echo "<table style='border: solid 1px black;'>";
  echo "<tr><th>Time</th><th>Sensor Reading</th></tr>";
  class TableRows extends RecursiveIteratorIterator {
     function __construct($it) {
        parent::__construct($it, self::LEAVES_ONLY);
     }
     function current() {
        return "<td style='width:150px;border:1px solid black;'>" . parent::current(). "</td>";
     }
     function beginChildren() {
        echo "<tr>";
     }
     function endChildren() {
        echo "</tr>" . "\n";
     }
  }


  try {
     $conn = get_db_connection();
     $stmt = $conn->prepare($querry_for_sensor_data);
     $stmt->execute([':sensor_id' => $sensor_id, ':start_time' => $start_time]);
     // set the resulting array to associative
     $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
     foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
     }
  }
  catch(PDOException $e) {
     echo "Error: " . $e->getMessage();
  }
  $conn = null;
  echo "</table>";
}
function get_last_sensor_data($sensor_id){
  require 'sql_query_and_insertion.php';
  $conn = get_db_connection();
```

```
  $stmt = $conn->prepare($querry_last_sensor_data);
  $stmt->execute([':sensor_id' => $sensor_id]);
  $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);

  echo $result;
}


  function display_farm_combo($farmer_id, $selectedID=0) {
  require 'sql_query_and_insertion.php';
  $conn = get_db_connection();
  $stmt = $conn->prepare($query_for_farms);
  $stmt -> execute([':FarmerID' => $farmer_id]);
  echo "<select name = \"farmNameSelect\" >";
  echo "  <option  value = \"Add new farm\">Add new farm</option>";
  while ($farmInfo = $stmt->fetch(PDO::FETCH_ASSOC)){
    echo "<option value = \"" ;
    echo $farmInfo["FarmID"];
         echo "\"";
    if ($farmInfo["FarmID"] == $selectedID){
      echo " selected ";
    }
    echo ">";
    echo $farmInfo["FarmName"];
    echo "</option>";
  }
  echo "</select>";
}
function display_region_combo($farm_id, $selectedID=0) {
  require 'sql_query_and_insertion.php';
  $conn = get_db_connection();
  $stmt = $conn->prepare($query_for_regions);
  $stmt -> execute([':FarmID' => $farm_id]);
  echo "<br>";
  echo "<select name = \"regionNameSelect\" >";
  echo "  <option  value = \"Add new region\">Add new region</option>";
  while ($region_info = $stmt->fetch(PDO::FETCH_ASSOC)){
    echo "<option value = \"" ;
    echo $region_info["RegionID"];
    echo "\"";
    if ($region_info["RegionID"] == $selectedID){
      echo " selected ";
    }
    echo ">";
    echo $region_info["FarmRegionName"];
    echo "</option>";
  }
  echo "</select>";
}
function display_sensor_combo($farm_id, $region_id) {
   require 'sql_query_and_insertion.php';
   $conn = get_db_connection();
   $stmt = $conn->prepare($query_for_sensors);
   $stmt -> execute([':FarmID' => $farm_id, ':RegionID' => $region_id]);
   echo "<br>";
   echo "<select name = \"sensorNameSelect\" >";
   echo "  <option  value = \"Add new sensor\">Add new sensor</option>";
   while ($sensor_info = $stmt->fetch(PDO::FETCH_ASSOC)){
```

```
      echo "<option value = \"" ;
      echo $sensor_info["RegionID"];
      echo "\"";
//       if ($sensor_info["RegionID"] == $selectedID){
//           echo " selected ";
//       }
      echo ">";
      echo $sensor_info["FarmRegionName"];
      echo "</option>";
   }
   echo "</select>";
}
function add_farm($farmName, $farmerID){
  require 'sql_query_and_insertion.php';
  try {
     $conn = get_db_connection();
    //echo "Connection made<br>";
    $stmt = $conn->prepare($insert_into_farms);
     //echo "Insert Prepared<br>";
     $stmt->execute([':farmName' => $farmName, ':farmerID' => $farmerID]);
     //echo "Insert Executed<br>";
     return $FarmID = $conn->lastInsertId();
  }
  catch(PDOException $e){
     echo "Error: " . $e->getMessage();
   }
  $conn = null;
}
function add_region($regionName, $regionDescription, $farmID){
  require 'sql_query_and_insertion.php';
  try {
     $conn = get_db_connection();
    //echo "Connection made<br>";
    $insert_into_region = "
       INSERT INTO FarmRegion(FarmRegionName, RegionDescription, Farm_FarmID)
       VALUES (:region_name, :regionDescription, :farm_id)
       ";
    $stmt = $conn->prepare($insert_into_region);
     //echo "Insert Prepared<br>";
     $stmt->execute([':region_name' => $regionName, ':regionDescription' => $regionDescription, ':farm_id'
=> $farmID]);
     //echo "Insert Executed<br>";
     $RegionID = $conn->lastInsertId();
     return $RegionID;
  }
  catch(PDOException $e){
     echo "Error: " . $e->getMessage();
   }
  $conn = null;
}
?>
```

sql_queeries_and_insertions.php

```
<?php
 $servername = "localhost";
 $username = "joseph";
 $password = "passcode";
```

```php
$dbname = "SmartGreenhouse_2";
$insertIntoFarmers =
    "INSERT INTO Farmer (FarmerName, FarmerEmail, farmerPassword)
    VALUES (:farmerName, :farmerEmail, :farmerPassword)";
$selectAllFromFarmerWHereEmail =
    "SELECT farmerID, FarmerEmail, FarmerName, farmerPassword
     FROM Farmer
      WHERE FarmerEmail = :farmerEmail";
$querry_for_sensor_data =
     "SELECT TimeStamp, SensorValue
      FROM SensorReadings
      WHERE InstalledSensors_SensorID = :sensor_id AND TimeStamp >= :start_time
      ORDER BY TimeStamp DESC
     ";
$querry_last_sensor_data =
     "SELECT SensorValue
      FROM SensorReadings
      WHERE InstalledSensors_SensorID = :sensor_id
      ORDER BY TimeStamp DESC
      LIMIT 1
     ";
$query_for_regions=
    "SELECT FarmRegionName, RegionID
     FROM FarmRegion
     WHERE Farm_FarmID = :FarmID
    ";
$query_for_farms=
   "SELECT FarmName, FarmID
    FROM Farm
    WHERE Farmer_FarmerID = :FarmerID
    ";
$insert_into_farms =
    "INSERT INTO Farm (FarmName, Farmer_FarmerID)
     VALUES (:farmName, :farmerID)
    ";
  $query_for_sensors=
     "SELECT SensorNickname, SensorID
     FROM InstalledSensors
     WHERE RegionID = :RegionID
     ";
?>
```