Expertise
and insight
for the future

Naveed Awais

IOT Based Hydroponic Controller Prototype

Metropolia University of Applied Sciences

Bachelor of Engineering

Electronics

Bachelor's Thesis

21 April 2020

Author Title Number of Pages Date	Naveed Awais IOT Based hydroponic controller Prototype 30 pages + 2 appendices 21 April 2020
Degree	Bachelor of Engineering
Degree Programme	Electronics
Professional Major	
Instructors	Matti Fischer, Principal Lecturer Erkki Räsänen, Principal lecturer

The goal of this project was to develop a hydroponic controller prototype which can control the environment of a hydroponic system by using open source microcontrollers. The controller will monitor the environmental elements that affect the growth of plants in system like pH, electrical conductivity, humidity and temperature by using respective sensors. The user can set desired threshold values. The system maintains the saved parameters and uploads the data to the data base.

An Arduino 2560 board was used to develop the device along with particle photon board to upload the data to mongoDB database. The device uses two analog sensors, one to measure pH of water and one for electrical conductivity and a DHT11 sensor to measure humidity and temperature.

The controller unit is designed in a way that it is easy to repair and easy to copy so that similar units could be developed without difficulty. User interface usability study was done and applied during writing the software for device to make it easy to use and avoid possible accidents by user. The safety of device operation was taken in account during designing its workflow to keep electronics in device and the equipment being controlled safe. The system architecture was designed flexible which allows interfaces that integrates with both current and future components. A description of how such system can be build is given. The workflow of system hardware and software is presented. This study can be used to develop similar systems.

The result of study is that, the performance of prototype was tested, and results were obtained through the data cloud. The device performed as expected and according to requirement of system. The standard of hardware of device was lower than the commercial standard. The device operation can be further improved by using industrial grade sensors in future.

Keywords	Hydroponics, Arduino, IOT,

Contents

List of Abbreviations

AC	Alternating current
DC	Direct current
EC	Electrical conductivity
EEPROM	Electrically erasable programmable read-only memory
GND	Ground signal
IO	Input and Output
IOT	Internet of things
IOT	Internet of things
LCD	Liquid crystal display
MCU	Micro controller unit
MISO	Master in slave out
MOSI	Master out Slave In
Op-amp	Operational Amplifier

pH	Power of hydrogen Ions – The amount of acidity of basicity of a solution
SPI	Serial peripheral interface
VCC	Voltage signal

1 Introduction

This thesis study has been focused on development of a prototype for a hydroponic controller unit. The goal of this whole project was to develop a hydroponic controller prototype based on open source platforms and microcontrollers. There are already hydroponic controllers in market, but they are expensive and not open source.

This device is a prototype of IOT based controller, developed to monitor and control the environment of a hydroponic system. The device monitors the live data from sensors including temperature, humidity, pH and electric conductivity of solvent water and maintains the desired parameters set by the user. The user can set desired parameters for pH and electrical conductivity and the device maintains the parameters by using four peristaltic pumps. The data is displayed on an LCD display panel and is uploaded to a cloud system. An Arduino mega microcontroller was used to maintain and monitor the environment. A particle photon device was used to upload data to server. The development of device was done in a way that device should be easy to assemble so that further similar models could be developed. The device has space to add more sensors in future if needed. The device sends the collected data to a database. The security of system was prioritized, so that user can safely use device without causing possible accidents and errors.

In a human operated hydroponic system, the factors affecting the growth of plants must be continuously monitored by using precise equipment on frequent basis. Minerals are added by hand in precise amounts. pH of water is maintained by using solutions called pH up solution and pH down solution. Farmers have to constantly look after the plants just like traditional land farming. To operate a hydroponic system, respective knowledge is also required which discourages most of indoor farmers to invest in such systems. In case of an error one plant can affect all plants in system. But in an automated setup the system monitors the environment and by remotely accessing the state of plants it saves a lot of resources and time. Farmers can use that data to compare the growth of plants with the obtained results and optimize the parameters to get desired results in future.

2 Hydroponics

Hydroponics is the process of growing plants based on urban farming techniques. Hydroponics is a subset of hydroculture, where plants use water solution mixed with nutrients to get minerals instead of from ground.

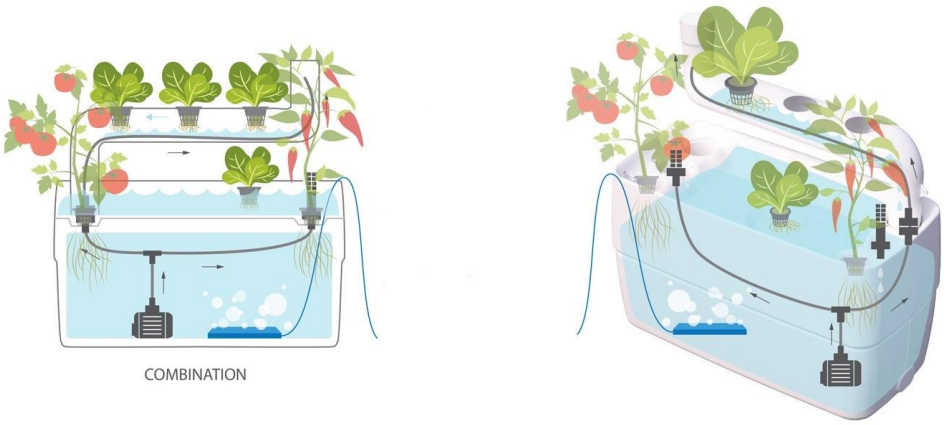

Figure 1. A Hydroponics System

In this process, as shown in figure 1[1.] the plants are provided with optimum conditions for their growth, which includes the amount of nutrients in water, pH of water, the intensity of light and surrounding atmosphere of plants. By optimizing the parameters, we can get maximum yield. This process is highly useful in the countries where weather conditions are not good for plants growth.

Finland is an example of such case where growing vegetables and fruits depending on natural weather conditions gives a very limited time slot throughout the year. It results in discouraging the farmers. It also increases number of imports and prices of food items. According to Wikipedia [1] the advantages of using hydroponics include:

- No pesticides are used in this process
- No soil is needed for hydroponics

- The water stays in the system and can be used again - thus, a lower water requirement
- Pesticides are not used in this process
- Easy harvesting process
- Plants grow more healthy
- Not season dependent

It is an environmentally sound growing method where water and nutrients are recycled until they are used up by the plants. Nothing is wasted, and nothing ends up in our rivers and lakes [2.].

This capability to wirelessly control the quality of water is very important. In natural technique of growing by sand, the plants get the nutrients from the ground. The plants grown by hydroponics get their required minerals from water full of nutrients and salts, which farmers have to watch so that mineral levels not increase too much to make it poisonous for plant life.

Installing a well-regulated hydroponic system can take a lot of time and needs knowledge of hydroponics before investing. Some indoor growers find it difficult and complicated to constantly monitor the pH and taking readings but having an online accessible system would ease this process.

The process can be made easy with an IOT based process with all the components which can run the hydroponic system reliably. It can be one-time investment but for a long term it will reduce the workload, recourses and maintenance of the system.

3 System Architecture

The working structure of device is shown in figure 2. The Arduino MCU simply collects the data and it is responsible for maintaining the parameters set by user and particle photon is receiving the data from MCU by serial communication. The particle photon converts the data into Json format and uploads it to particle mobile app, Mongo DB database [3], and Heroku cloud [4]. Flexible architecture consents to add further sensors in the system easily.

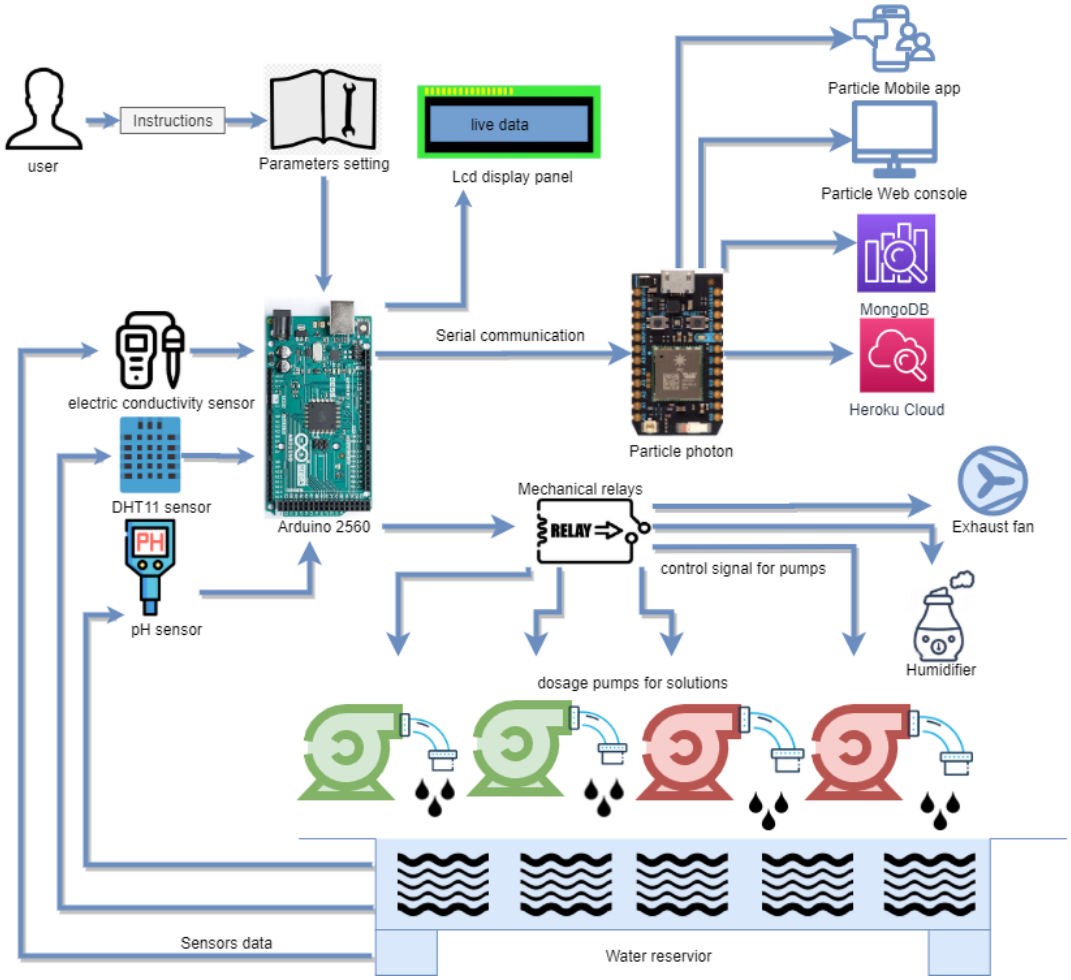

Figure 2. System Architecture

The MCU controls

- 2 dosage pumps to maintain pH
- Exhaust fan to maintain humidity
- 2 dosage pumps to maintain electrical conductivity

The device can maintain the humidity using a DC fan used as exhaust and a humidifier. The MCU monitors if the current humidity of surroundings is less than the parameters set by user and it turns on the humidifier until the humidity level reaches the desired limit. The device always tries to reach parameters limit which is dependent on saved parameters. For example, if the user sets 60% humidity level, the device will try to reach the limit that is 60% ± 5%.

In case of pH and electrical conductivity maintaining, the MCU compares the live readings with the saved parameters and turns on the respective pump to increase and decrease respective entity. Maintaining the pH is a bit hard since a minor change in water, can make big difference in hydrogen ion concentration. An air pump continuously shakes and mixes the water in reservoir so the added solution or minerals can be mixed quickly. Each dosing pump runs for 5 milliseconds and waits for 20 seconds to get the dosage mix. After that sensor probe again takes new measurement and compares it with required pH to check if more dosage is needed, and the process continues until desired pH limit is achieved.

The Device always maintain the pH between the pH limits. It is almost impossible to achieve the pinpoint desired figures due to sensitivity of the probe.

4 Hardware

4.1 Temperature and Humidity sensor

The DHT11 is a commonly used temperature and humidity sensor. Low cost and appropriate accuracy of this sensor was suitable enough for use in this project. The sensor can get data every 3-5 seconds to perform at best. This should not be a problem for hydroponics. Having a big temperature difference in this much time is not probable. The sensor was first tested to understand connections on a prototype breadboard, as shown in Figure 3 Below [5].

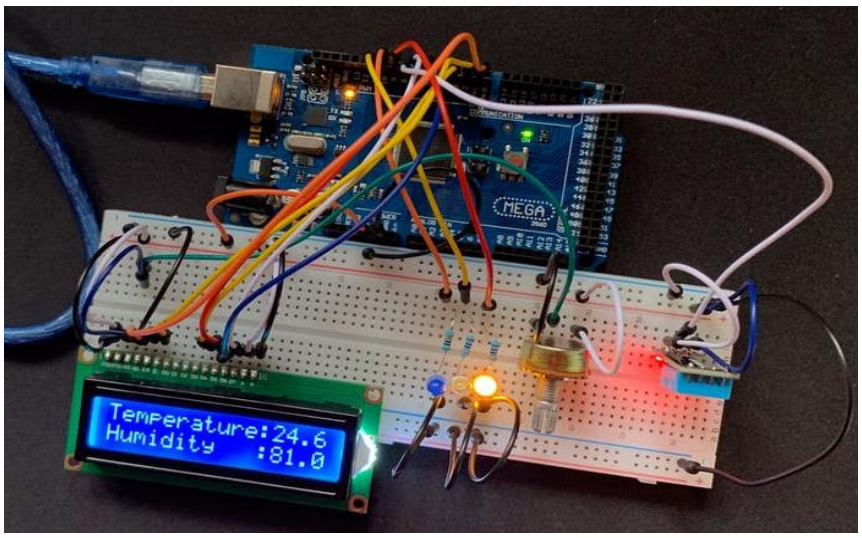

Figure 3. Testing DHT11 Sensor

The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is factory calibrated and hence easy to interface with other microcontrollers [6].

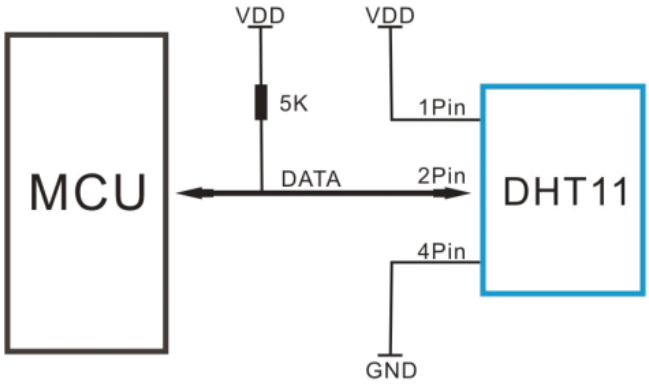

Figure 4. DHT11 Sensor Interface with MCU

As shown in Figure 4, the sensor has one data pin, which sends humidity and temperature values as serial data to arduino MCU. It is recommended by manufacturer to use a pull-up resistor of 5K if we use connection wire longer than 20 meters.
Single-bus data format is used for communication and synchronization between MCU and DHT11 sensor. One communication process is about 4 milliseconds.

4.1.1 Communication process:

The communication is serial interfaced and uses decimal and integral data. For communication the MCU first sends a start signal to the sensor. The sensor stays in low power mood unless it receives a start signal from MCU. Once the start signal is completed, DHT11 sends a response signal of 40-bit data that include the relative humidity and temperature information to MCU. A complete transmission of data is 40 bit, and the sensor sends higher data bit first.

When DHT11 detects the start signal from MCU, it replies with a low-voltage level response signal, which is 80 µs. Then DHT11 sets the data single bus from low to high to send the data bit to MCU. In this case the length of signal determines the bit. The figure 5, demonstrates the transmission of “0” bit from sensor. [6.]

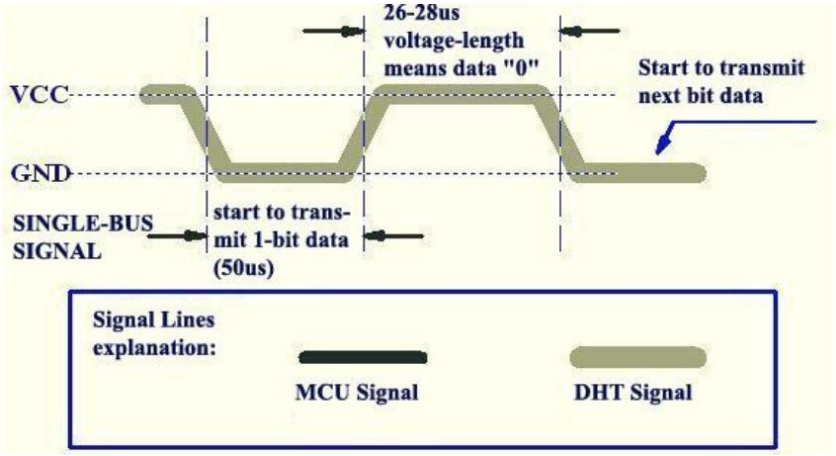

Figure 5. Representation of Bit "0" transmission

The length of signal between 26 µs to 28 µs represents the “0” bit and 70 µs voltage length means data bit “1” is received as shown in figure 6 below.

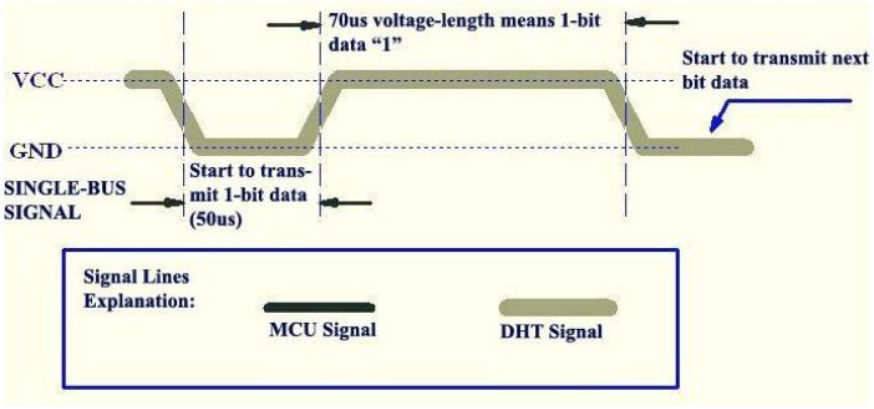

Figure 6. Representation of Bit "1" transmission

If the response signal from sensor is always, high it is taken as error and suggests DHT11 is not working properly. After the last bit is sent the sensor pulls down the voltage level for 50 µs and the single bus voltage will be pulled up by the resistor to set it back to free status.[6].

The electrical properties of sensor are mentioned in table 1.

Table 1. Electrical properties of DHT11 Sensor

	Conditions	minimum	typical	Maximum
supply	DC	3 V	5 V	5.5 V
Current supply	During operation	0.5mA		2.5 mA
	average	0.2mA		1 mA
	Standby	100 µA		150 µA
Sampling time	second	1		
humidity	measure	20%		90%
Temperature	measure	0°C		50°C

The above parameters were measured at room temperature.

4.2 pH sensor

pH is the measure of the hydrogen ion concentration of a solution. Solutions with a high concentration of hydrogen ions have a low pH and solutions with a low concentration of H+ ions have a high pH. To measure the pH of solvent the following pH sensor was used as shown in figure 7.

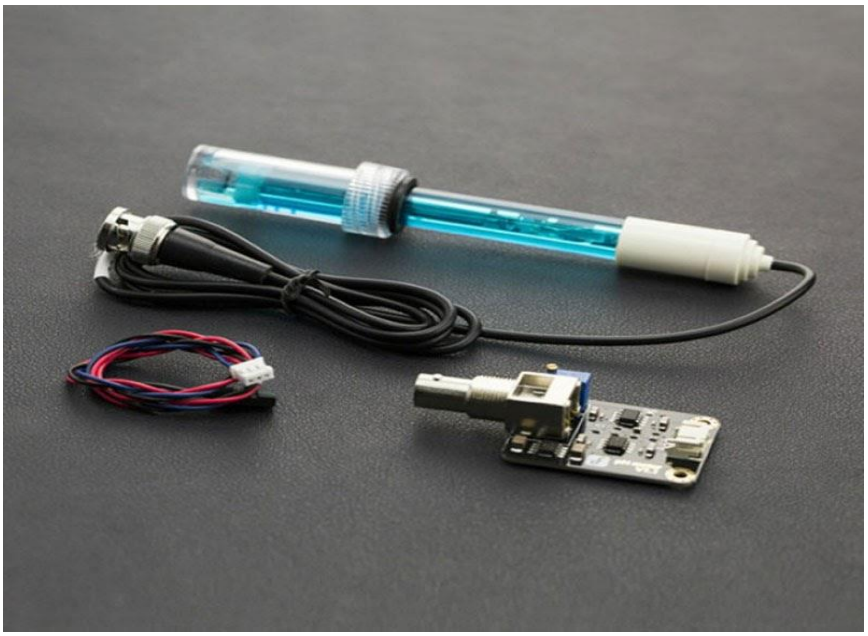

Figure 7. The pH Sensor Probe

It is an analog sensor, which means it outputs analog voltage which is dependent on pH solution. The sensor detects the presence of hydrogen ions in a solution with highly sensitive electrodes. As a relation, a difference of one pH unit (i.e. from pH 2 to pH 3) is a ten-fold (10X) difference in H+ ion concentration so, the concentration of hydrogen ions in a pH 7 solution is ten times less than the pH 6 solution.

The sensor can be used by using the “Arduino library for pH sensor by dfrobot” which can be downloaded from the manufacturer’s website [7]. In this case the sensor was not calibrated by the manufacturer, so the pH value was calculated by reading analog signals from sensor. The sensor was tested with different pH calibration solutions and results

were saved. The Figure 8 shows the output analog signals (Series 2) from MCU for different calibration solutions with pH values (Series 1) and illustrates the linearity of the sensor.

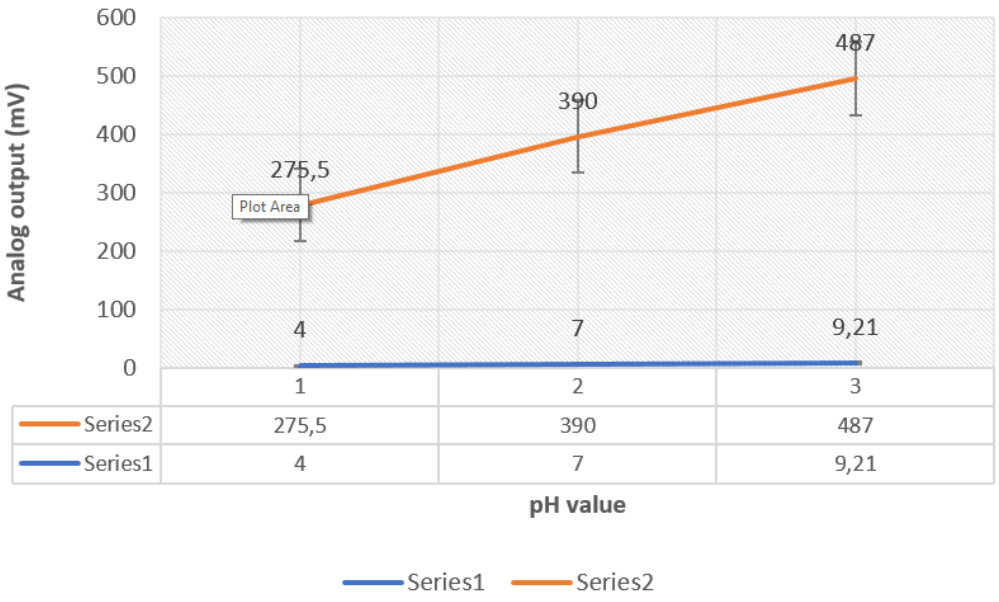

Figure 8. Analog voltage as a function of pH

From the above graph, by using mathematical slope calculations we can calculate the pH for any solution at room temperature.

$$Y = mx + b \qquad (1)$$

In equation (1) if 'm' is slope, 'x' is analog voltage reading from sensor and 'b' is called intercept.

$$m = (4 - 7)\,(275 - 400)$$

$$\mathrm{m} = \mathrm{Slope} = 0.024 \qquad (2)$$

Intercept calculation:

The intercept, b equation (3) has been calculated using pH 7 solution. For pH 7 The analog reading, x = 400 approximately.

$$intercept = b = y - m$$
$$b = (7) - (0.024 * 400)$$
$$b = -2.6 \quad (3)$$

By using the above calculation method, the following output voltage values were calculated as shown in table 2.

Table 2. Analog readings for pH sensor

pH values	Analog output (mV)	pH values	Analog output (mV)
01	150,0	08	441,6
02	191,6	09	483,3
03	233,3	10	525,0
04	275,0	11	566,6
05	316,6	12	608,3
06	358,3	13	650,0
07	400,0	14	691,6

The readings were measured at room temperature.

4.3 Electrical conductivity sensor

Conductivity is the ability of a substance to carry the current. It is the reciprocal of resistivity. [16.] In fluids, to calculate the conductivity we use reciprocal of resistivity. It is important to know quality of water used in hydroponic system because it represents the amount of salts and minerals present in water. The water with more dissolved minerals would have more conductivity and vice versa.

In the International system of Units, the unit of conductivity is Siemens / meter (S/m), and the other units are: S/m, mS/cm, µS/cm. Conversion relationship is: 1S/m = 1000mS/m = 1000000µS/m = 10mS/cm = 10000µS/cm.

Figure 9: The electrical conductivity sensor probe

The following sensor in figure 9 [16.] is an analog sensor like the pH sensor, containing two sensitive electrodes that detect the conductivity of water for a specific temperature. The given buffer solutions were accurate enough at 25 centigrade temperature. Before measuring a different solution, water was used to wash the conductivity electrode, in

order to prevent contamination and inaccurate results. The deionized water is recommended to wash the conductive electrodes of the sensor.

The sensor was first tested with water using table salt with water to detect increase and decrease in the conductivity of water and was found that this sensor is suitable to measure electrical conductivity from 1 mS to 20 mS. Since conductivity of water was dependent on water temperature, the sensor needed a temperature compensation. However, the device was supposed to operate in a closed and maintained temperature environment, temperature compensation was neglected. The sensor operated at 5 VDC and needs a stable power supply. The more stable is supply voltage the more accurate measurements we get.

4.3.1 Electrical conductivity calculations

The conductivity is related to resistivity of system. Figure 10 shows the schematic [8.] for the U3B chip on left hand side. R10 is a feedback resistance and its value is 820 Ω. The transfer function is

$$Vo = R10/R \times Vi \qquad (4)$$

$$\frac{R10}{R} = gain \qquad (5)$$

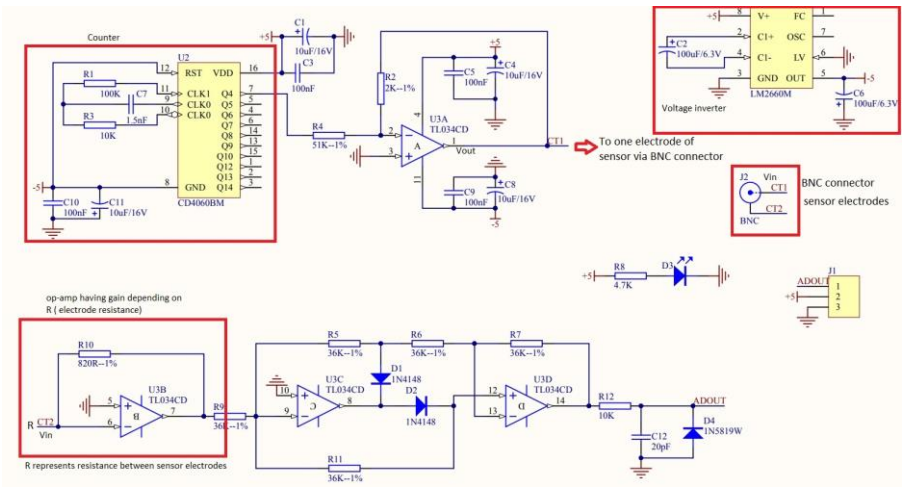

Figure 10 Electrical Conductivity Probe Schematic for Chip U3B

The output of op-amp at U3B circuit depends on the Vin and R. Here R in equation (4) is the resistance value when the electrode is placed inside a solution and its value is depended on conductivity of the solution under test and equation (5) represents the gain value and Vin is voltage from op-amp U3A output. So, when R is changed the gain also changes and ultimately Vo as well. The CT2 is connected to BNC connector terminal as shown in figure 10 which connects the conductivity probe. LM2660M represents the voltage inverter used to convert +5 VDC to -5 VDC.

On the right of the U3B chip circuit, there is an absolute-value circuit. Its transfer function is Vo=|Vi|. MCU samples the output of the absolute-value circuit to calculate conductivity.

$$R = \rho\, L/A \qquad (6)$$

We can define resistance as equation (6). Here ρ is the resistivity, L is the length of the resistor element and in this case, it is spacing between the electrodes of electrical conductivity probe, A is the cross section of the resistor, which in this case is the area of electrodes of the probe that conducts. Conductivity is defined in equation (7) as follows

$$\sigma = 1/\rho \qquad (7)$$

From equation 6 and 7 we can derive equation (8). In this equation 1/R is reciprocal equation which is basically conductance G and L/A is vessel constant Q.

$$\sigma = \frac{1}{R} * \frac{L}{A} \qquad (8)$$

The transfer function of the measurement circuit is given as follows

$$Vout = \frac{R10}{R} * |Vin| \qquad (9)$$

Here |Vin| is also a constant depending on the signal generating circuit and its value according to manufacturer [8] is about 200 mV. In conclusion we can derive conductivity as equation (10)

$$\sigma = \frac{Q*Vout}{(R10*|Vin|)} \qquad (10)$$

4.4 Arduino mega 2650

The arduino mega in figure 11 is microcontroller board based on ATmega2650 micro-processor chip. This microcontroller unit has 256 kilobytes of internal programmable flash memory, 4 kilobytes of EEPROM. [9.] It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator. The board operates at 6 VDC to 12 VDC and can also be used as DC to DC converter with an output of 5 VDC and 3.3 VDC which can be used to Power low power sensors during testing.

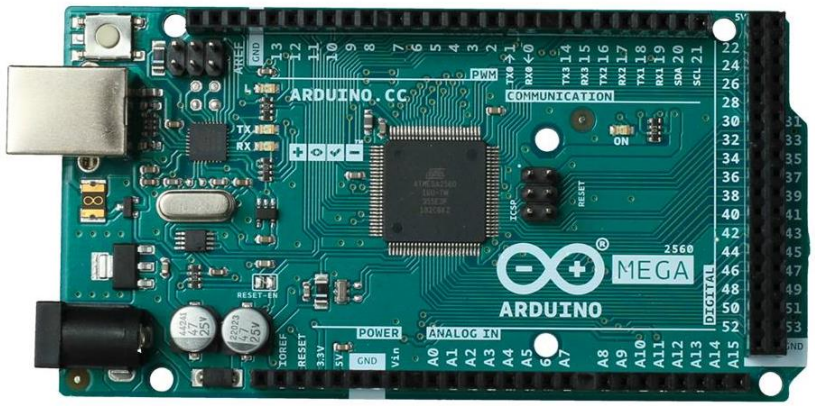

Figure 11 An Arduino mega 2650 Microcontroller

Some of the pins on the board serve additional functions (Arduino.cc, 2013):

• Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data.

• PWM: 2 to 13 and 44 to 46. Provides the 8-bit PWM output.

• SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS)

• LED: 13. There is a built-in LED connected to Digital pin 13.

• TWI: 20 (SDA) and 21 (SCL). The USB connector is used to programme the MCU or open the serial monitor to see liv software operation.

The MCU was choosed to use in the prototype for having enough IO pins to support current sensors and have room for future addition of more sensors in the system. Serial communication was used to share data between Arduino and particle photon MCUs.

4.5 Particle photon

Particle is an IOT based MCU having powerful ARM cortex M3 chip and a built-it WIFI module. A Mobile app by manufacturer is used to configure and setup WIFI credentials [10]. The photon operates at 3.6 VDC and 5.5 VDC and is responsible was sending the sensors data to the cloud in this project. The photon receives the data from Arduino MCU for all sensors through the Serial communication and uploads it to cloud.

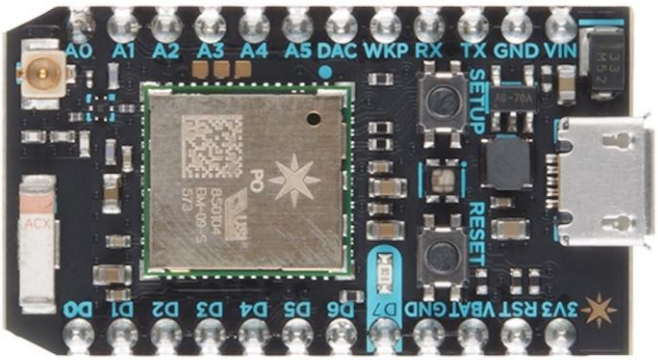

Figure 3: A Particle Photon Device

This particle photon MCU can also be used by using micro USB as shown in figure 12. The compact size of device makes it very useful to use in projects using very little surface area or in light weight devices. The device also has an antenna connection which can be used to extend the antenna range in open spaces.

4.6 Peristaltic Pumps

Four peristaltic pumps as shown in figure 13 were used of which, two for supplying of mineral solutions and two for maintaining the pH of the water reservoir. The pump contains a DC motor operating at 6 VDC to 12 VDC which draws the liquid using a suction mechanism. During pump selection, suction power is important to care about so that at given voltage the pump should be able to draw the liquid from height of 3 meters and supply as needed. The nutrients dosage should be very small during pumps each delivery which helps the device during balancing the pH.

Figure 13 A Peristaltic Pump

4.7 RFID Security

The figure 14 shows An RFID card reader MFRC522 chip-based board being tested. It was installed at the bottom left of the device to give a security feature to device. The card reader operates at 13.56 MHz. The device parameters can be edited by using a RFID key tag. In this situation a bus card was used and saved as RFID key to demonstrate the function.

The security of device is a vital step since any mishap in a hydroponic system affects the whole system. In our case most of sources controlling the system are disabled and can only be activated with saved key.

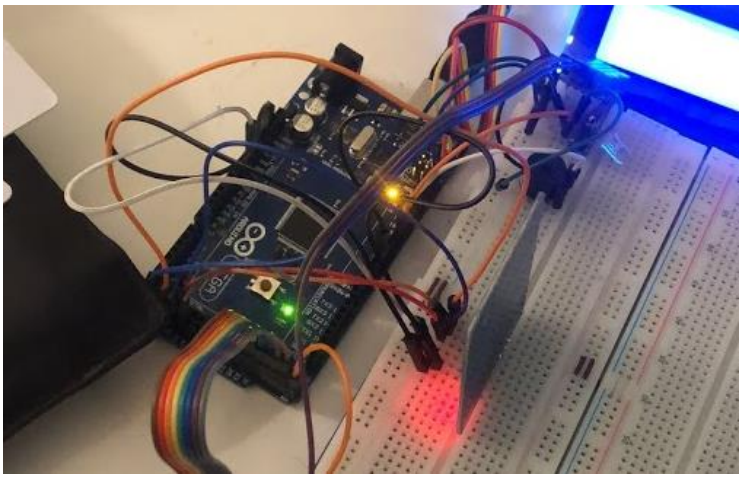

Figure 14 Testing the RFID card Reader

The card reader operates at 3.3 VDC and have SPI interface. The card reader can be used after adding the MCU library by manufacturer for MFRC522 [11].

4.7.1 Arduino wiring for card reader

Following table 3 shows the wiring instruction while using arduino mega MCU. It can be different for using other MCUs than mega.

Table 3 MFRC522 wiring with arduino mega

MFRC522 pins	MCU pins
SDA (ss)	D53
MOSI	D51
SCK	D52
MISO	D50
GND	GND
RST	D9
3.3	3.3 VDC sharply!

4.8 Relay strip

A strip of 8 - mechanical relays was used for switching of AC and DC power operating devices used in project. Mechanical relays are devices that behaves like a switch and can be turned on and off by applying a small voltage. They are used for low speed switching. The relays have electromagnet which can be activated by low power like 5 VDC. The magnetic coil is used to connect high voltage contacts.

These relays can be used for switching of 120 VAC at 10 A and 30 VDC at 10 A. In this project, the relays were used as shown in figure 15, for switching of both AC source like Humidifier and DC sources including DC motor fan and peristaltic pumps.

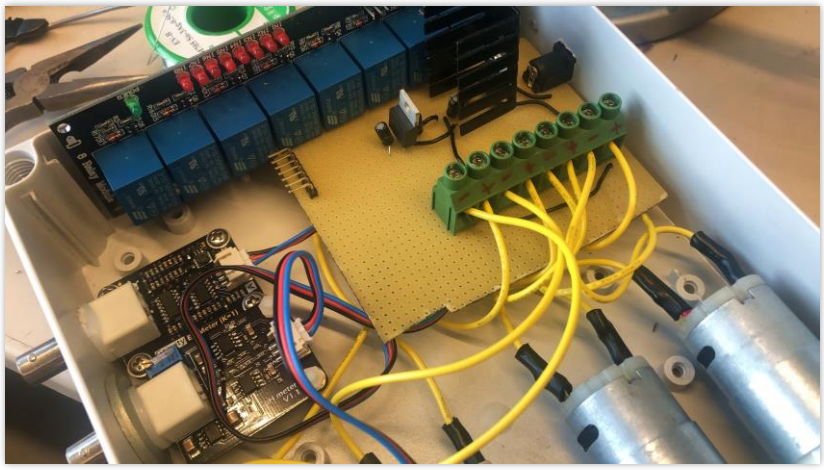

Figure 15 Strip of 8- Relay modules

4.9 Other hardware

Other hardware used in this project includes

- A 20 × 4 LCD display
- A 4 × 4 alpha numeric keypad
- 12VDC fan
- Connection Wires
- Plastic enclosure to install all electronics

5 Device Assembly

The electronics were enclosed in a plastic enclosure shown in figure 16. The MCU is placed at the bottom of device in case of future changing in the code and debugging it can be easily accessed through USB-C port. The user cannot access the particle photon device and reset it. It can only be edited through particle photon web console. The device is programmed to measure and upload figures to cloud during normal operation and user can see live figures on LCD panel. If the user wants to change the default parameters, it can be done by using the RFID card key. The alpha numeric keypad will be enabled with correct key credentials. Keypad is disabled by default and user cannot manipulate settings without activating keypad.

Main power circuit was made by using prototyping PCBs. A 2.1-millimeter female DC barrel jack was used for main power supply source.

Figure 16 Assembled prototype front view

For safety reasons all AC and DC operated sources are turned off by default by relays and user can turn them on after accessing settings by using RFID key. After settings open, user can see instructions to change the parameters and can easily save new parameters.

In the bottom front, A reset button is Installed in case of emergency which can reset the MCU and set the parameters to default and turns off all relay operated sources.

The EC and pH probes are connected through two BNC-connectors on the left side of device and can be easily removed for deionizing the probes as needed.

6 Software and Programming

6.1 Arduino IDE

The Arduino IDE [12] is an open source software used to Programme the MCU used in this project. It contains a text editor for writing the code for MCU which can be in C or C++ programming language. A computer USB is used to monitor the live data and performance through its serial monitor. It contains a toolbar as shown in figure 17 through which libraries can be added.The main function of arduino IDE is to convert the C progarmming language into machine language that the MCU can understand and save.

Once MCU is programmed the C code cannot be achieved from MCU, since it only saves the machine langauge in its memory.It have a serial plot function which can be used to draw plots and see live performance for example reading the analog voltage and ploting a graph.

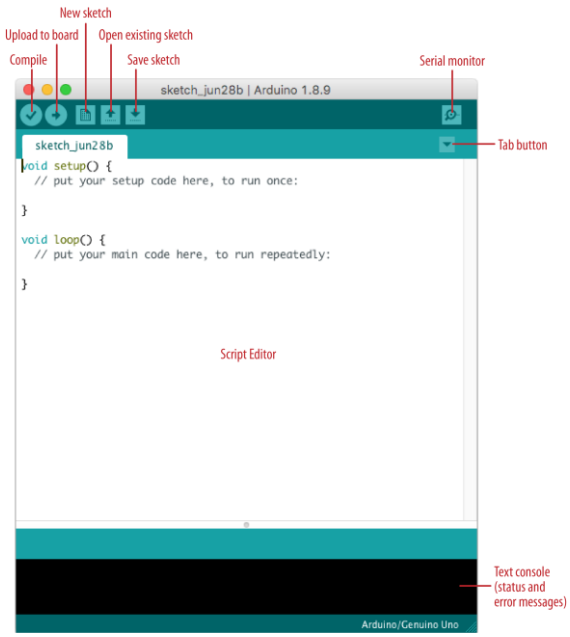

Figure 17 Arduino IDE

6.2 Particle Web IDE

Particle photon has WEB based IDE [13] as shown in figure 18. It is very advanced IDE where we can update the code for MCU and monitor the data wirelessly. The photon is connected through internet and can be controlled from anyplace in world. In this project the IDE was used to write the code to receive the data from Arduino MCU and upload it to cloud.

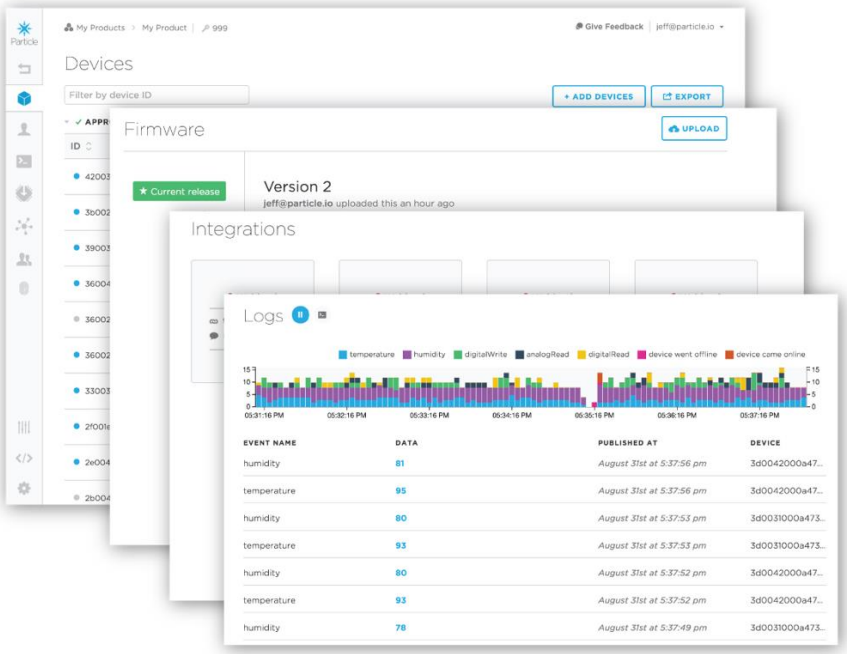

Figure 18 Particle IDE and Console

6.3 Flow chart:

The figure 19 shows the working flow of software which demonstrates one sequence of programme during which the parameters are measured, and respective actions are done.

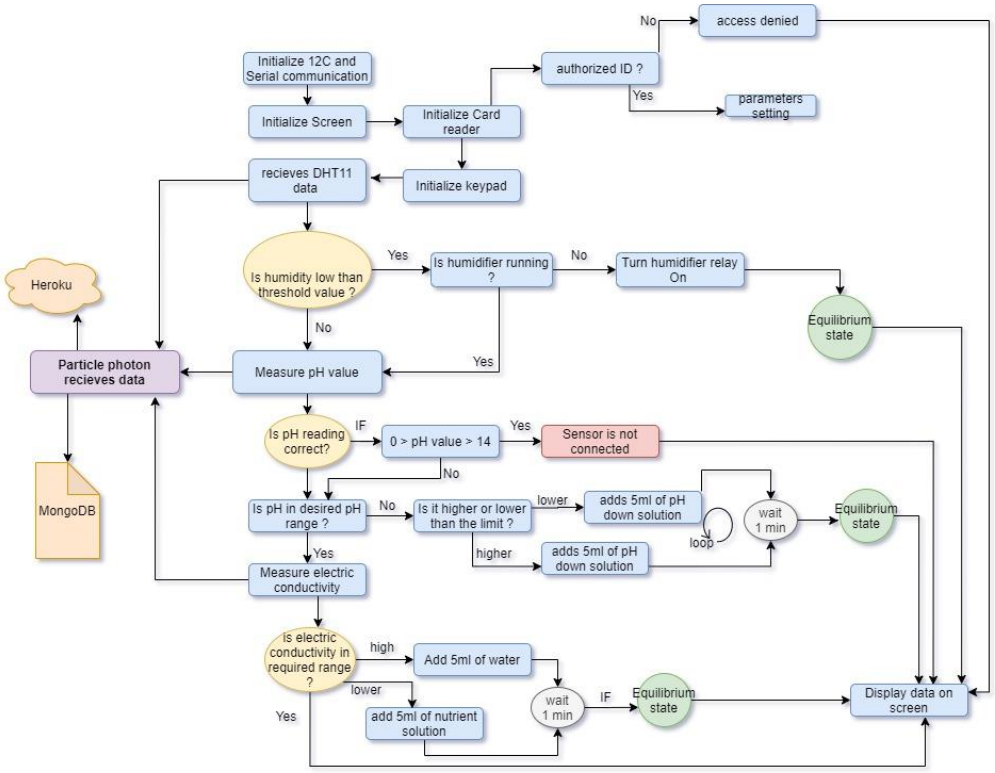

Figure 19 software flow chart

The Arduino code for this software is provided in appendix 1 with instructions and helping texts.

7 Cloud and Data collection

7.1 Heroku cloud and deploying app

Heroku [4] is a commercial cloud platform which companies use to monitor and access their data. It allows the processes of deploying, configuring, scaling, tuning, and managing the Virtual apps. In this project the App was built using Node.js app. Following are the prerequisites before starting to build the node.js app.

- Node.js and npm installed (node package manager)
- an existing Node.js app.
- a free Heroku account
- the Heroku CLI (command line interface)

The deployment of app was done using Node.js CLI. The start creating the app we make package.Json file by using following command.

```
npm init
```

The generated package.Json file in this case looks like this in figure 20

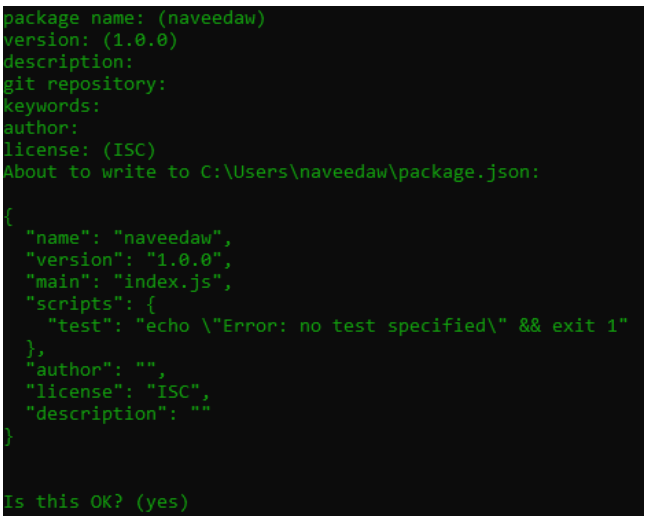

Figure 20 Package.json file generated

After that the NPM install command is used in your local app directory to install the dependencies that you declared in your package.json file. Following are the commands to deploy the app to Heroku cloud.

```
Git add .
Git commit -m ”Text to commit”
Heroku login
Git push Heroku master
```

During running the following commands each space and dot should be followed to avoid errors in deployment. Once the deployment is done the data logs can be accessed through following commands [14].

```
Heroku logs
Heroku logs –tail
```

Figure 21 Heroku logs showing received data

The figure 21 is showing the data received and connection status with mongoDB database. The data figures are for demonstration purpose representing humidity, temperature, pH and electrical conductivity respectively.

7.2 MongoDB ATLAS

MongoDB is a commercial data base used to visualize the data [3]. In this project the service was used to obtain and visualize the data by charts and graphs. The Figure 22 shows the data feed coming from particle photon device.

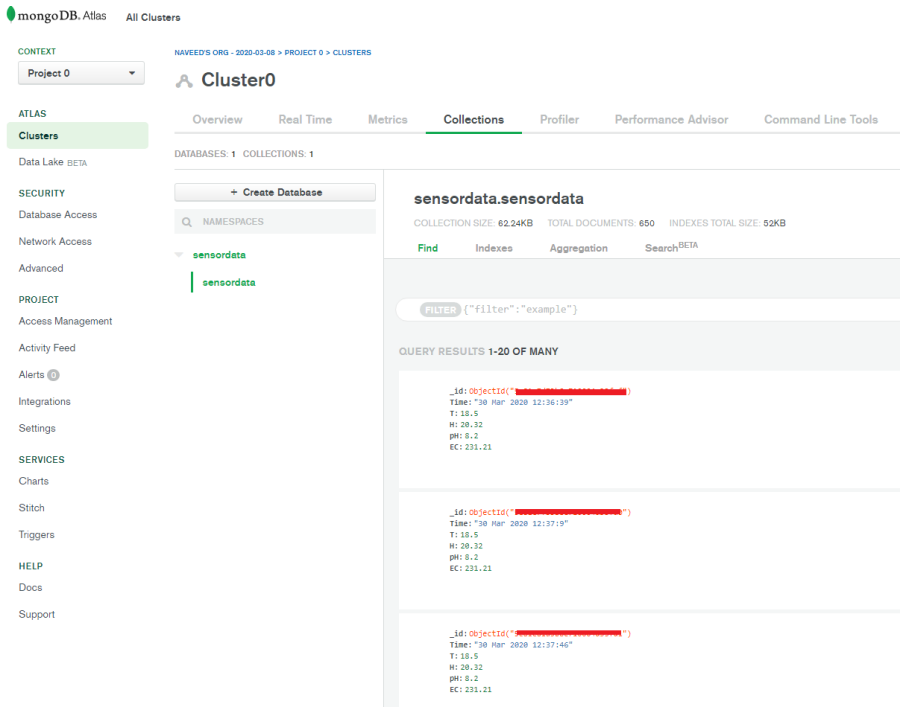

Figure 22 MongoDB collections dashboard

The figure 22 shows the data received which is used to make charts and data visualizations to understand it better.

MongoDB Charts allows for a secure way to create and share visualization dashboards with everyone, or just targeted team members. Similarly, the data source being used behind the scenes can be shared securely as well. [3].

8 Conclusion

To sum up the whole project, the goal was to design a system for a hydroponic controller device based on open source platforms and microcontrollers. The device should be easy to develop such that more copies of hardware could be produced. The device can read data and save in database. The data can be accessed by mobile app as well as by cloud.

The device was developed by considering the requirements of a hydroponic system and was tested in lab by using pH solutions. The device was able to maintain pH. But in case of uneven power supply the pH probe becomes highly unstable. Such big fluctuations in pH cause over dosage of solution which can be fetal for plants.

The device still needs improvements in sensors. The sensors used in this project were not very precise and were unstable. During this project the price of whole hardware was kept as low as possible. The MCU used, is good for prototyping but cannot be used on commercial scale applications due to security and safety standard issues. Improvement can be done by replacing the MCU with the Industrial PLC. The manufacturer of the sensors used in this project does not recommend using them for commercial purposes. The life of sensor probes used in this project is about 6 months. Replacing the sensor probes with commercial ones can solve the problem with the stability of measurements data.

References:

[1] **Supara gardens**. hydroponic company product. [online] [cited: March 4, 2020.] https://www.supragarden.com/epages/supragarden.sf/en_GB/?ObjectPath=/Shops/2016122606/Categories/how_it_works/Hydroponics_works

[2] **Hydroponics** [online] https://en.wikipedia.org/wiki/Hydroponics [cited: March 2, 2020.]

[3] **MongoDB.** [online] database. [cited: April 02, 2020] https://www.mongodb.com/what-is-mongodb

[4] **Heroku.** [online] Data cloud platform [cited: April 06, 2020] https://dashboard.heroku.com/auth/heroku/callback?code=aa68f9f2-6356-469f-beb3-1eeedc02ded3

[5] **sensor Image** [online] [cited: March 26, 2020.] https://create.arduino.cc/projecthub/r4lph127/temperature-and-humidity-sensor-with-led-lights-bf4ce6

[6] **DHT11**. datasheet [online] sensor manufacturer [cited: March 26, 2020.] https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf

[7] **dfrobot**. [online] pH sensor probe [cited: April 15,2020] https://www.dfrobot.com/product-1025.html?search=ph%20meter&description=true

[8] **Dfrobot**. [online] Analog electric conductivity meter. [cited: March 12,2020.] https://wiki.dfrobot.com/Analog_EC_Meter_SKU_DFR0300

[9] **Arduino.cc**. [online] microcontroller Arduino mega 2560 [cited: March 23, 2020.] https://store.arduino.cc/arduino-mega-2560-rev3

[10] **Particle.io**. [online] datasheet [cited: April 15, 2020] https://docs.particle.io/datasheets/wi-fi/photon-datasheet/

[11] **Software library**. [online] MFRC522 RFID card reader. [cited: April 05, 2020] https://www.arduinolibraries.info/libraries/mfrc522

[12] **Arduino cc**. [online] Arduino ide [cited: April 17, 2020] https://www.arduino.cc/en/Main/Software

[13] **Particle IDE**. [online] online console [cited: April 12, 2020] https://build.particle.io/build

[14] **Heroku**. [online] Heroku cli commands [cited: April 12,2020] https://devcenter.heroku.com/articles/heroku-cli-commands

[15] **GitHub**. [online] Liquid crystal display library. [cited: March 21, 2020.] https://github.com/arduino-libraries/LiquidCrystal

[16] **Dfrobot.** [online store] electric conductivity sensor. [cited: March 23,2020.] https://wiki.dfrobot.com/Analog_EC_Meter_SKU_DFR0300

[17] **Dfrobot.** [Online] Circuit schematic [cited: April 05, 2020] http://image.dfrobot.com/image/data/DFR0300/DFR0300_v1.0_schematic.pdf

[18] **phidgets**. [online] mechanical relays working [cited: march 14, 2020] https://www.phidgets.com/docs/Mechanical_Relay_Primer

Arduino MCU code:

```
#include <SoftwareSerial.h>      //library
#include <EduIntro.h>  // include the EduIntro library for dht11 update
#include <Wire.h>               //library
#include <LiquidCrystal_I2C.h>//library
#include <SPI.h>//library
#include <MFRC522.h>//library
#include <Keypad.h>//library
#include "DFRobot_PH.h"//library
#include "DFRobot_EC.h"//library
#include <EEPROM.h>//library


#define EC_PIN A0
  int buzzer = 11;
  int fanrelay= 42;
  int PHuprelay = 47;  //P1 PHup
int PHdownrelay = 44; //P2 PHdown
int ECuprelay = 48;   //P3 ECup
int ECdownrelay = 43; //P4 ECdown

DHT11 dht11(7);
int Temperature;
float F;
String Humidity;
float m = 0.024;
float b = -2.6;

// HUMIDITY VARIABLES
String humidityy="80"; //Variable to store the current humidity
String temphumidityy =""; //Variable to store the input humidity
int doublecheckhumidityy;    //Check twice the entered humidity value
boolean armed = false;  //Variable for system state (armed:true / un-
armed:false)
boolean input_humidityy;   //Variable for input (correct:true / wrong:false)
boolean storedhumidityy = true;
boolean changedhumidityy = false;
boolean checkhumidityy = false;
int i = 1; //variable to index an array

float x = analogRead(A1); //int

float y = m * x + b  ;
//PH VARIABLES
String initialPH= "7.0"; //Variable to store the current ph
String tempPH=""; //Variable to store the input ph
int doublecheckPH;    //Check twice the entered ph value
boolean armedPH = false;  //Variable for system state (armed:true / un-
armed:false)
boolean input_PH;   //Variable for input (correct:true / wrong:false)
boolean storedPH = true;
boolean changedPH = false;
boolean checkPH = false;
float desiredPH= (initialPH.toFloat());
float upperPHlimit= (desiredPH + 0.10);       //ph limits defining
float lowerPHlimit= (desiredPH - 0.10);
```

```
//-------------------------------------------------------------------------------------------//
//EC VARIABLES
String initialEC= "0.2"; //Variable to store the current Ec value
String tempEC=""; //Variable to store the input Ec values
int doublecheckEC;    //Check twice the entered Ec value and compares
boolean armedEC = false;  //Variable for system state (armed:true / un-
armed:false)
boolean input_EC;   //Variable for input (correct:true / wrong:false)
boolean storedEC = true;
boolean changedEC = false;
boolean checkEC = false;
float desiredEC= (initialEC.toFloat());
float upperEClimit= (desiredEC + 0.10);
float lowerEClimit= (desiredEC - 0.10);

LiquidCrystal_I2C lcd(0x27, 20, 4); // i2c adress for lcd
#define SS_PIN 53
#define RST_PIN 5

float  voltagePH, voltageEC, phValue, ecValue, temperature = 25;
DFRobot_PH ph;
DFRobot_EC ec;

MFRC522 mfrc522(SS_PIN, RST_PIN);
byte degree[8] = {
  0b01100,
  0b10010,
  0b10010,
  0b01100,
  0b00000,
  0b00000,
  0b00000,
  0b00000
};

const byte rows = 4; //number of the keypad's rows and columns
const byte cols = 4;

char keyMap [rows] [cols] = { //define the symbols on the buttons of the key-
pad

  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', '.'}
};

byte rowPins [rows] = {37, 35, 33, 31}; //pins of the keypad
byte colPins [cols] = {29, 27, 25, 23};



Keypad myKeypad = Keypad( makeKeymap(keyMap), rowPins, colPins, rows, cols);




void commands(){
  float desiredPH= (initialPH.toFloat());
```

```
  float desiredEC= (initialEC.toFloat());
  Serial.print("Desired PH value:");
  Serial.println(desiredPH);
  Serial.print("Desired ec value:");
  Serial.println(desiredEC);
  Serial.println("---------------------------------");
}
void mainscreen()
{
main:


  lcd.createChar(1, degree);
  dht11.update();
  Temperature = dht11.readCelsius();       // Reading the temperature in Cel-
sius degrees and store in the C variable
  Humidity = dht11.readHumidity();

  // Print the collected data in a row on the Serial Monitor

  lcd.setCursor(0, 0);
  lcd.print("Humidity: ");
  lcd.print(Humidity);
  lcd.print("%");
  Serial.print("Humidity: ");
  Serial.print(Humidity);
  Serial.println("%");
  if(Humidity > humidityy) {
    digitalWrite(42,LOW);
    Serial.println("humidity fan Runing!");
  }

 if( Humidity < humidityy)
  {
    digitalWrite(42,HIGH);
 }

delay(500);

  lcd.setCursor(0, 1);
  lcd.print("Temperature: ");
  lcd.print(Temperature);
  lcd.write(1);
  lcd.print("C");
  Serial.print("Temperature: ");
  Serial.print(Temperature);
  Serial.println(" Centigrade");
  Serial.println("---------------------------------");



//  digitalWrite(47, HIGH);
//  digitalWrite(48, HIGH);
//  digitalWrite(44, HIGH);
//  digitalWrite(43, HIGH);

char   keypressed = myKeypad.getKey();
  if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
```

```
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == 'A' || key-
pressed == 'B' ||
      keypressed == 'C' || keypressed == '.' || keypressed == '*' || key-
pressed == '#' ) {
    lcd.clear();
    buzzerfalse();
    lcd.setCursor(0, 0);
    lcd.print("Keypad not Activated");
    lcd.setCursor(0, 2);
    lcd.print("Please read card!");
    delay(1500);
    lcd.clear();
    goto main;
  }
}
void PHEC()

{
  float upperPHlimit= (desiredPH + 0.10);
  float lowerPHlimit= (desiredPH - 0.10);
  Serial.println("USE RFID CARD TO UNLOCK KEYPAD");
  Serial.println("Press * to save data");
  Serial.println("Press # to go back");
  float desiredEC= (initialEC.toFloat());
  float desiredPH= (initialPH.toFloat());
  float upperEClimit= (desiredEC + 0.10);
float lowerEClimit= (desiredEC - 0.10);
  // PH eC calculations
  char cmd[10];
  static unsigned long timepoint = millis();
  if (millis() - timepoint > 700U) {                    //time interval: 1s
    /*timepoint = millis();
      //temperature = readTemperature();                  // read your tem-
perature sensor to execute temperature compensation
      voltagePH = analogRead(PH_PIN) / 1024.0 * 5000;       // read the ph
voltage
      phValue    = ph.readPH(voltagePH, temperature);       // convert voltage
to pH with temperature compensation
      Serial.print("pH:");
      Serial.print(phValue, 2);
      lcd.setCursor(0, 3);
      lcd.print("pH:");
      lcd.print(phValue, 2);
      delay(100);            */
    PH:
    float x = analogRead(A1); // float
    float y = m * x + b  ;
    Serial.println("PH and EC calculations and figures");
    Serial.print("pH:");
    Serial.println(y, 3);
    lcd.setCursor(0, 3);
    lcd.print("pH:");
    lcd.print(y, 2);
    delay(700);
    if(y>14 && y > upperPHlimit){                               // y is calcu-
lated PH Value
      lcd.setCursor(9,3);
      lcd.print("No sensor!");
      Serial.println("PH sensor not connected!");
      delay(1000);
```

```
      lcd.setCursor(9,3);
      lcd.print("           ");
      delay(100);
    }
    if(y<0 && y < lowerPHlimit ){
      lcd.setCursor(10,3);
      lcd.print("No sensor!");
      delay(1000);
      lcd.setCursor(10,3);
      lcd.print("          ");
      delay(100);
       }
    if (y > desiredPH && y > upperPHlimit && y < 14 )
         {

      digitalWrite(44, LOW);
      delay(500);
      digitalWrite(44, HIGH);
      delay(10000);
      Serial.println("pump 2 is Lowering the PH");

 }
 if (y < desiredPH && y < lowerPHlimit  )
    {
      digitalWrite(47, LOW);
      delay(500);
      digitalWrite(47, HIGH);
      delay(10000);
      Serial.println("pump 1 is Raising the PH");
      Serial.println(initialPH);
    }
   if (lowerPHlimit<= y <= upperPHlimit)
    {
      digitalWrite(47, HIGH);

      digitalWrite(44, HIGH);
      Serial.print("Desired PH limit reached");
    }

    }

  //Ec calculations:

    voltageEC = analogRead(EC_PIN) / 1024.0 * 5000;
    ecValue    = ec.readEC(voltageEC, temperature);      // convert voltage to
EC with temperature compensation
    Serial.print("EC:");
    Serial.print(ecValue, 2);
    Serial.println("ms/cm");
    lcd.setCursor(0, 2);
    lcd.print("EC:");
    lcd.print(ecValue, 2);
    lcd.print("mS/cm");
    Serial.println("------------------------------");
    Serial.print("lower eC limit: ");
    Serial.println(lowerEClimit);
     Serial.print("upper eC limit: ");
    Serial.println(upperEClimit);


    if ( ecValue < desiredEC && ecValue < lowerEClimit    )
    {
```

```
      digitalWrite(48, LOW);
      delay(500);                    //pump 3
      digitalWrite(48, HIGH);
      delay(10000);
      Serial.println("Pump 3 is raising EC");
    }

    if (ecValue == 0 ) {
      digitalWrite(47, HIGH);
                                        // both pumps off
      digitalWrite(43, HIGH);
      Serial.println("EC probe not connected!");
      Serial.print("Ec pumps off");


    }
    if ( lowerEClimit <= ecValue,2 <= upperEClimit  ) {

      digitalWrite(47, HIGH);  // both pumps off

      digitalWrite(43, HIGH);
      Serial.println("Desired EC limit Reached");

    }
    if (ecValue > desiredEC && ecValue > upperEClimit   )
    {
      digitalWrite(43, LOW);   //pump 4
      delay(500);
      digitalWrite(43, HIGH);
      delay(10000);
      Serial.println("pump 4 is reducing EC");
    }
    else{
       digitalWrite(47, HIGH);  // both EC pumps off

      digitalWrite(43, HIGH);

    }
  if (readSerial(cmd)) {
    strupr(cmd);
    if (strstr(cmd, "PH")) {
      ph.calibration(voltagePH, temperature, cmd);     //PH calibration pro-
cess by Serail CMD
    }
    if (strstr(cmd, "EC")) {
      ec.calibration(voltageEC, temperature, cmd);     //EC calibration pro-
cess by Serail CMD
    }
  }
}

 // int i = 0;
bool readSerial(char result[]) {
  while (Serial.available() > 0) {
    char inChar = Serial.read();
    if (inChar == '\n') {
      result[i] = '\0';
      Serial.flush();
      i = 0;
      return true;
    }
```

```
    if (inChar != '\r') {
      result[i] = inChar;
      i++;
    }
    delay(1);
  }
  return false;

}
  // RFID READING AND PRINTING
void rfid()
{
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.println("Message : setting opened ");
  content.toUpperCase();

  if (content.substring(1) ==  "xxxxxxxxxxxx" ) // only card can access for
now
  {
    buzzertrue();
    lcd.clear();
    delay(300);
    do {
      Control();
    } while (content.substring(1) ==  "xxxxxxxxxx"  );
  }
  else   {

    Serial.println(" Access denied");
    lcd.clear();
    buzzerfalse();
    lcd.setCursor(0, 0);
    lcd.print("Access Denied!");
    delay(900);
    lcd.clear();
    return; //goto label

  }
}
//MAIN SETTING SCREEN
void Control()
```

```
{
controlmain:
  lcd.setCursor(0, 0);
  lcd.print("Press 'A' for PH");
  lcd.setCursor(0, 1);
  lcd.print("Press 'B' for EC");
  lcd.setCursor(0, 2);
  lcd.print("Press 'C' for others");
  lcd.setCursor(0, 3);
  lcd.print("Press '#' to Return");

  char keypressed = myKeypad.getKey();

  if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ||
      keypressed == '*') {

    lcd.clear();
    buzzerfalse();//Button tone
    lcd.setCursor(0, 0);
    lcd.print("Invalid Selection");
    lcd.setCursor(0, 2);
    lcd.print("Please select Again");
    delay(1500);
    lcd.clear();
    goto controlmain;

  }
  else if (keypressed == 'A') {
    lcd.clear();
    Serial.println("'A' pressed");
    lcd.setCursor(0, 0);
    tone(buzzer,800,200);
    lcd.print("PH selected!");
    Serial.println("PH control opened!");
    delay(800);
    lcd.clear();
    do {
      PHcontrol();
    } while (keypressed == 'A');

  }
  else if (keypressed == 'B') {
    lcd.clear();
     Serial.println("'B' pressed");
    tone(buzzer,800,200);
    lcd.setCursor(0, 0);
    lcd.print("EC selected!");
    Serial.println("EC control opened!");
    delay(800);
    lcd.clear();

    do {
      ECcontrol();
    } while (keypressed == 'B');
 }
 else if (keypressed == 'C') {
```

```
    lcd.clear();
     Serial.println("'C' pressed");
    tone(buzzer,800,200);
    lcd.setCursor(0, 0);
    lcd.print("others selected!");
    Serial.println("others setting opened!");
    delay(800);
    lcd.clear();

    do {
      Others();
    } while (keypressed == 'C');
 }
  else if (keypressed == '#') {
    lcd.clear();
    Serial.println("# pressed");
    Serial.println("back to main screen");
    do {
      loop();
    }
    while (keypressed  == '#');

  }
}
//PH changing CODE//
void PHcontrol()
{
    char keypressed = myKeypad.getKey();

  do {
  moto: //goto label
  tempPH="";
  lcd.clear();
  i=6;
  noTone(buzzer);



  while(!checkPH){
    lcd.setCursor(0,0);
    lcd.print("Press B to change");
    lcd.setCursor(0,1);
    lcd.print("PH settings!");

    float m = 0.024;
    float b = -2.6;
       int x = analogRead(A1);

     int y = m * x + b ;

  lcd.setCursor(0, 3);
  lcd.print("Live PH is: ");
  lcd.print(y,1);
  delay(10);


 char keypressed = myKeypad.getKey();    //Read pressed keys
    if (keypressed != NO_KEY){     //Accept only numbers and * from keypad
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
```

```
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ){
        tempPH += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);       //printing
        i++;
        tone(buzzer,800,200);   //Button tone
      }
      else if (keypressed == 'B'){
        changePH();
        goto moto;
      }
      else if (keypressed=='#'){
      lcd.clear();
     Serial.println("# pressed");
    Serial.println("back to main screen");
    do {
        loop();
    }
    while (keypressed  == '#');
    }
      else if (keypressed == '*'){  //Check for ph
        if (initialPH==tempPH){//If it's correct...
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Accepted");
          lcd.setCursor(0,1);
          lcd.print("PH saved");
          tone(buzzer,100);    //Play a tone while door is unlocked
          //unlock the door for 5 seconds
          delay(5000);
          goto moto;
        }
        else{           //if it's false, retry
          tempPH="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto moto;
        }
      }
    }
  }

    lcd.setCursor(0, 3);
    lcd.print("Press '#' to Return");


  }
  while(keypressed == 'A');

}
//CONTINUES
// PH changing CODE
//Change current PH
void changePH(){
  retry: //label for goto
  tempPH="";
  lcd.clear();
  i=1;
```

```
  while(!changedPH){        //Waiting for current ph
   char keypressed = myKeypad.getKey();   //Read pressed keys
    lcd.setCursor(0,0);
    lcd.print("Enter old PH");
    lcd.setCursor(0,1);
    lcd.print(">");
    lcd.setCursor(0,3);
    lcd.print("previous PH:");
    lcd.setCursor(13,3);
    lcd.print(initialPH);


    if (keypressed != NO_KEY){
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ){
        tempPH += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
        tone(buzzer,800,200);
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
        i=1;
        if (initialPH==tempPH){

          storedPH=false;
          tone(buzzer,500,200);
          newPH();
          break;
        }
        else{               //Try again
          tempPH="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto retry;
        }
      }
    }
  }
}
//CONTINUES
String firstPH;
void newPH(){
  tempPH="";
  changedPH=false;
  lcd.clear();
  i=1;
  while(!storedPH){
    char keypressed = myKeypad.getKey();   //Read pressed keys
    if (doublecheckPH==0){
      lcd.setCursor(0,0);
      lcd.print("Enter required PH");
      lcd.setCursor(0,1);
      lcd.print(">");
```

```
    }
    else{
      lcd.setCursor(0,0);
      lcd.print("One more time...");
      lcd.setCursor(0,1);
      lcd.print(">");
    }
    if (keypressed != NO_KEY){
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9'  || keypressed == '.'){
        tempPH += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
          tone(buzzer,800,200);
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
        if (doublecheckPH == 0){
          firstPH=tempPH;
          doublecheckPH=1;
          newPH();
        }
        if (doublecheckPH==1){
          doublecheckPH=0;
          if (firstPH==tempPH){
            i=1;
            firstPH="";
            initialPH = tempPH; // New ph saved
            tempPH="";//erase temp ph

            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Saving");
            lcd.setCursor(7,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(8,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(9,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(0,2);
            lcd.print("New PH saved!");

              storedPH=true;
              tone(buzzer,500,400);
              delay(2000);
              lcd.clear();
              break;
          }
          else{
            firstPH="";
            newPH();
          }
        }
```

```
      }
    }
  }
}
void ECcontrol()
{
  char keypressed = myKeypad.getKey();
    voltageEC = analogRead(EC_PIN) / 1024.0 * 5000;
    ecValue    = ec.readEC(voltageEC, temperature);

  do {
    moto: //goto label
  tempEC="";
  lcd.clear();
  i=6;
  noTone(buzzer);



  while(!checkEC){
    lcd.setCursor(0,0);
    lcd.print("Press B to change");
    lcd.setCursor(0,1);
    lcd.print("EC settings!");


    voltageEC = analogRead(EC_PIN) / 1024.0 * 5000;
    ecValue    = ec.readEC(voltageEC, temperature);

  lcd.setCursor(0, 3);
  lcd.print("Live EC is: ");
  lcd.print(ecValue,1);
  delay(10);


 char keypressed = myKeypad.getKey();    //Read pressed keys
    if (keypressed != NO_KEY){     //Accept only numbers and * from keypad
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ){
        tempEC += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);       //printing
        i++;
        tone(buzzer,800,200);   //Button tone
      }
      else if (keypressed == 'B'){
        changeEC();
        goto moto;
      }
      else if (keypressed=='#'){
      lcd.clear();
     Serial.println("# pressed");
    Serial.println("back to main screen");
    do {
        loop();
    }
    while (keypressed  == '#');
    }
      else if (keypressed == '*'){  //Check for key
```

```
        if (initialEC==tempEC){//If it's correct...
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Accepted");
          lcd.setCursor(0,1);
          lcd.print("New Ec saved");
          tone(buzzer,100);    //Play a tone while door is unlocked
          //unlock the door for 5 seconds
          delay(5000);
          goto moto;
        }
        else{           //if it's false, retry
          tempPH="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto moto;
        }
      }
    }
  }

    lcd.setCursor(0, 3);
    lcd.print("Press '#' to Return");
  }
  while (keypressed == 'B');

}

//CONTINUES
// Ec changing CODE
//Change current Ec
void changeEC(){
  retry: //label for goto
  tempEC="";
  lcd.clear();
  i=1;
  while(!changedEC){        //Waiting for current ec
   char keypressed = myKeypad.getKey();   //Read pressed keys
    lcd.setCursor(0,0);
    lcd.print("Enter old EC");
    lcd.setCursor(0,1);
    lcd.print(">");
    lcd.setCursor(0,3);
    lcd.print("previous EC:");
    lcd.setCursor(13,3);
    lcd.print(initialEC);


    if (keypressed != NO_KEY){
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ){
        tempEC += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
        tone(buzzer,800,200);
      }
```

```
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
        i=1;
        if (initialEC==tempEC){

          storedEC=false;
          tone(buzzer,500,200);
          newEC();          //ec is corrent, so call the newec function
          break;
        }
        else{               //Try again
          tempEC="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto retry;
        }
      }
    }
  }
}
//CONTINUES
String firstEC;
//Setup new ec
void newEC(){
  tempEC="";
  changedEC=false;
  lcd.clear();
  i=1;
  while(!storedEC){
    char keypressed = myKeypad.getKey();   //Read pressed keys
    if (doublecheckEC==0){
      lcd.setCursor(0,0);
      lcd.print("Enter required EC");
      lcd.setCursor(0,1);
      lcd.print(">");
    }
    else{
      lcd.setCursor(0,0);
      lcd.print("One more time...");
      lcd.setCursor(0,1);
      lcd.print(">");
    }
    if (keypressed != NO_KEY){
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9'  || keypressed == '.'){
        tempEC += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
          tone(buzzer,800,200);
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
```

```
        if (doublecheckEC == 0){
          firstEC=tempEC;
          doublecheckEC=1;
          newEC();
        }
        if (doublecheckEC==1){
          doublecheckEC=0;
          if (firstEC==tempEC){
            i=1;
            firstEC="";
            initialEC = tempEC; // New ec saved
            tempEC="";//erase temp EC
            float desiredEC= (initialEC.toFloat());
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Saving");
            lcd.setCursor(7,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(8,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(9,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(0,2);
            lcd.print("New EC Saved!");

              storedEC=true;
              tone(buzzer,500,400);
              delay(2000);
              lcd.clear();
              break;
          }
          else{
            firstEC="";
            newEC();
          }
        }
      }
    }
  }
}

//HUMIDITY CHANGING CODE
void changehumidityy(){
  retry: //label for goto
  temphumidityy="";
  lcd.clear();
  i=1;
  int j=i+1;
  while(!changedhumidityy){        //Waiting for current humidityy
   char keypressed = myKeypad.getKey();   //Read pressed keys
    lcd.setCursor(0,0);
    lcd.print("Enter old value:");
    lcd.setCursor(0,1);
    lcd.print(">");
    lcd.setCursor(0,3);
    lcd.print("old setting:");
    lcd.setCursor(13,3);
    lcd.print(humidityy);
```

```
    if (keypressed != NO_KEY){
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.'){
        temphumidityy += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
        tone(buzzer,800,200);
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
        i=1;
        if (humidityy==temphumidityy){
          Serial.print(humidityy);
          storedhumidityy=false;
          tone(buzzer,500,200);
          newhumidityy ();          //humidityy is corrent, so call the newhu-
midityy function
          break;
        }
        else{               //Try again
          temphumidityy="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto retry;
        }
      }
    }
  }
}
//CONTINUES
String firstpass;
//Setup new humidityy
void newhumidityy (){
  temphumidityy="";
  changedhumidityy=false;
  lcd.clear();
  i=1;
  while(!storedhumidityy){
    char keypressed = myKeypad.getKey();   //Read pressed keys
    if (doublecheck==0){
      lcd.setCursor(0,0);
      lcd.print("Enter Required Value");
      lcd.setCursor(0,1);
      lcd.print(">");
    }
    else{
      lcd.setCursor(0,0);
      lcd.print("One more time...");
      lcd.setCursor(0,1);
      lcd.print(">");
    }
    if (keypressed != NO_KEY){
```

```
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.' ){
        temphumidityy += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);
        i++;
          tone(buzzer,800,200);
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){
        if (doublecheck == 0){
          firstpass=temphumidityy;
          doublecheck=1;
          newhumidityy ();
        }
        if (doublecheck==1){
          doublecheck=0;
          if (firstpass==temphumidityy){
            i=1;
            firstpass="";
            humidityy = temphumidityy; // New humidityy saved
            temphumidityy="";//erase temp
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Saving");
            lcd.setCursor(7,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(8,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(9,0);
            lcd.print(".");
            delay(400);
            lcd.setCursor(0,2);
            lcd.print("New Humidity Saved!");

              storedhumidityy=true;
              tone(buzzer,500,400);
              delay(2000);
              lcd.clear();
              break;
          }
          else{
            firstpass="";
            newhumidityy ();
          }
        }
      }
    }
  }
}
//CONTINUES
  void Others()
{
  char keypressed = myKeypad.getKey();
```

```
  dht11.update();
  moto: //goto label
  temphumidityy="";
  lcd.clear();
  i=6;
  noTone(buzzer);



  while(!checkhumidityy){
    lcd.setCursor(0,0);
    lcd.print("Press A to change");
    lcd.setCursor(0,1);
    lcd.print("humidity settings");



    dht11.update();
  Temperature = dht11.readCelsius();       // Reading the temperature in Cel-
sius degrees and store in the C variable
  Humidity = dht11.readHumidity();



  lcd.setCursor(0, 3);
  lcd.print("Live Humidity: ");
  lcd.print(Humidity);
  lcd.print("%");
  delay(10);


    keypressed = myKeypad.getKey();    //Read pressed keys
    if (keypressed != NO_KEY){     //Accept only numbers and * from keypad
      if (keypressed == '0' || keypressed == '1' || keypressed == '2' || key-
pressed == '3' ||
      keypressed == '4' || keypressed == '5' || keypressed == '6' || key-
pressed == '7' ||
      keypressed == '8' || keypressed == '9' || keypressed == '.'  ){
        temphumidityy += keypressed;
        lcd.setCursor(i,1);
        lcd.print(keypressed);       //printing
        i++;
        tone(buzzer,800,200);   //Button tone
      }
      else if (keypressed == 'A'){
        changehumidityy ();
        goto moto;
      }
      else if (keypressed=='#'){
        break;
      }
      else if (keypressed == '*'){  //Check for humidityy
        if (humidityy==temphumidityy){//If it's correct...
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("Accepted");
          lcd.setCursor(0,1);
          lcd.print("humidity saved");
          tone(buzzer,100);
          //unlock the door for 5 seconds
          delay(5000);
          goto moto;
```

```
        }
        else{           //if it's false, retry
          temphumidityy="";
          tone(buzzer,500,200);
          delay(300);
          tone(buzzer,500,200);
          delay(300);
          goto moto;
        }
      }
    }
  }

  if (keypressed == '#')
    {
      lcd.clear();
     Serial.println("# pressed");
    Serial.println("back to main screen");
    do {
      loop();
    }
    while (keypressed  == '#');
    }
}
//tune indication
void buzzertrue()
{
  tone(buzzer, 800);
  delay(75);
  noTone(buzzer);
  delay(75);
  tone(buzzer, 800);
  delay(75);
  noTone(buzzer);
  delay(75);
}

void buzzerfalse()
{
  tone(buzzer, 500);
  delay(75);
  noTone(buzzer);
  delay(75);
  tone(buzzer, 500);
  delay(75);
  noTone(buzzer);
  delay(75);
  tone(buzzer, 500);
  delay(75);
  noTone(buzzer);
  delay(75);
}

//SETUP Settings
void setup()
{

  // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus

  pinMode(9, OUTPUT);
```

```
  pinMode(7, INPUT); //dht11
  pinMode(10, OUTPUT);
  pinMode(47, OUTPUT); //p1
  pinMode(44, OUTPUT); //p2
  pinMode(48, OUTPUT); //p3
  pinMode(43, OUTPUT); //p4
  pinMode(42, OUTPUT); // humidity fan
   pinMode(11, OUTPUT); // buzzer

 digitalWrite(PHuprelay,HIGH);
 digitalWrite(PHdownrelay,HIGH);
 digitalWrite(ECuprelay,HIGH);
 digitalWrite(ECdownrelay,HIGH);
 digitalWrite(fanrelay,HIGH);

  lcd.init();
  lcd.backlight();
  mfrc522.PCD_Init();
  Serial.begin(115200);
  ph.begin();
  ec.begin();

}
void loop()
{
  mfrc522.PCD_Init();
  rfid();
  mainscreen();
  PHEC();
 commands();
 Writing();
 humiwriting();
}
```

Code for Mongo dB:

```
const MongoClient = require('mongodb').MongoClient;

var Particle = require('particle-api-js');
var particle = new Particle();

//Get the following parameters form your Particle account
var token = '**************************';  //cli command    access token ex-
pires on Wed Jun 24 2020 17:32:39 GMT+0300
var id = '********************';  // my personal particle device ID

var obj; // data object for received mqtt data
var obj_m; // data object for mongodb

var myCollection;
var db;
var client_m;


function save_data() {
     obj_m = {Time: timeConverter(Date.now()), T: obj.T, H: obj.H, pH: obj.pH,
EC: obj.EC};
      console.log(obj);
        createConnection(function(){
          addDocument(function(){
                });
        });
}

function addDocument(onAdded){
     myCollection.insert(obj_m, function(err, result) {
      if(err)
    throw err;
         console.log("entry saved");
        onAdded();
     });
}

function createConnection(onCreate){
  MongoClient.connect('mongodb+srv://***username****:***password**@cluster0-
qduyr.mongodb.net/***appname**?retryWrites=true&w=majority', function(err,
client_m) {
     db = client_m.db('**appname**');
        if(err)
      throw err;
                console.log("connected to the mongoDB !");
                myCollection = db.collection('**appname**');
          onCreate();
        client_m.close();
  });
}

particle.getEventStream({ deviceId: id, auth: token }).then(function(stream) {
  var obj_;
    stream.on('event', function(obj_) {
                console.log("Event: ", obj_);
                     if(obj_.name=="data") obj = JSON.parse(obj_.data);
                     console.log(obj);
                          save_data();
          });
});
```

```
if (!Date.now) {
    Date.now = function() { return new Date().getTime(); }
}

function timeConverter(UNIX_timestamp){
  var a = new Date(UNIX_timestamp);
  var months =
['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'];
  var year = a.getFullYear();
  var month = months[a.getMonth()];
  var date = a.getDate();
  var hour = a.getHours();
  var min = a.getMinutes();
  var sec = a.getSeconds();
  var time = date + ' ' + month + ' ' + year + ' ' + hour + ':' + min + ':' +
sec ;
  return time;
}
```