UTM UNIVERSITI TEKNOLOGI MALAYSIA

Plantation Monitoring System

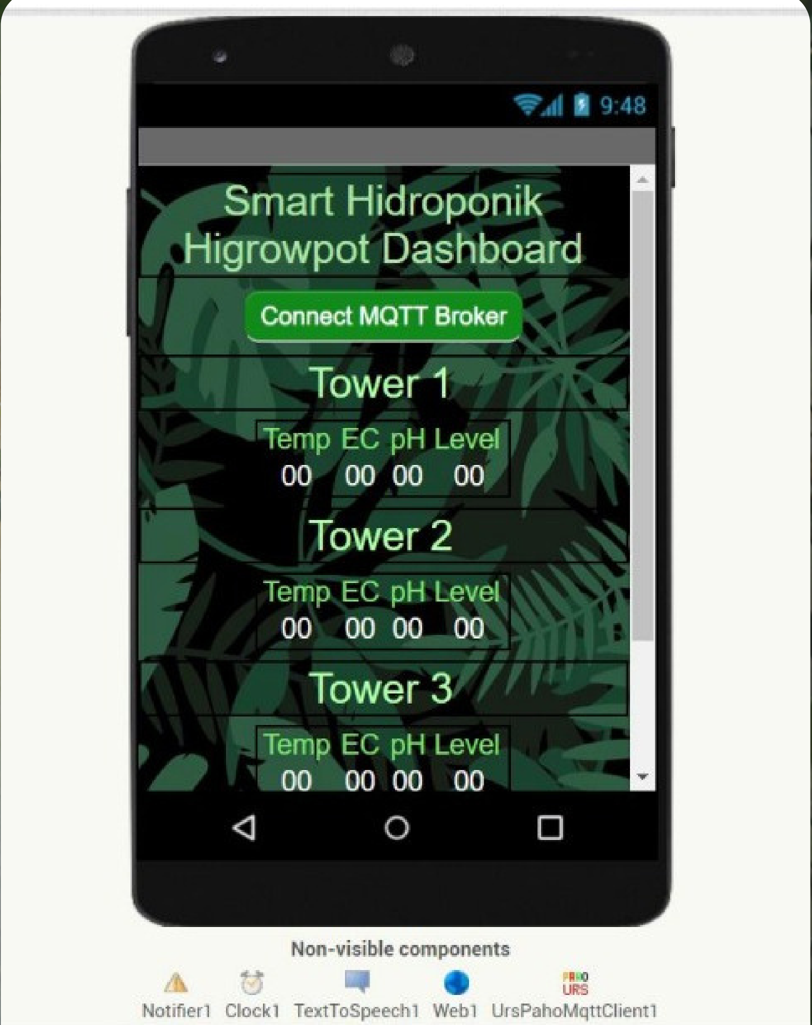

Figure 4.12 MIT App Inventor dashboard.

Mazen Omar Mohamed
A20EM4015
Ahmed MohamedAli
A20EM3003

ACKNOWLEDGEMENT

To begin, I must express my gratitude to Allah (SWT) for granting me this internship opportunity, and the will and dedication to complete its tasks successfully in the Smart Hidroponik Higrowpot Project in collaboration with UTM.

I present my deepest thanks to Universiti Teknologi Malaysia for accepting me into this internship and assisting me along my internship hunting process. I extend this gratitude to my Latihan Industri coordinator, Dr Hasbullah.

Moreover, I would like to thank my organization supervisor, Dr Zainab Binti Asus for baring with me along the internship process and assisting me in decision making by approving my presented ideas and results. My gratitude is also expressed towards Dr Nur Safwati for always being there to aid me with her knowledge in the field. I would also like to thank my friend and brother, Mazen Omar for helping me in the execution of the project and assisting me when needed.

I present my deepest thanks to SK Taman Tun Aminah for their continuing hospitality and cooperation. In particular, I must extend my gratitude for Mr Zul at SK TTA for sharing his extremely useful insights regarding the sensors used in the project.

Lastly, I must thank Dr Rozaimi Mohd Saad, my supervisor at UTM for having me at his office for my meeting with him and his cooperation.

ABSTRACT

The paper is a report that summarizes my work for my industrial training course over the span of 12 weeks. My internship was part of the Smart Hidroponik Higrowpot project that is a collaboration between UTM and SK Taman Tun Aminah.

The main objective was to develop and build the monitoring system of the hydroponic systems installed at the school. After exploring the various concepts and simulating the circuit, the execution of the project was commenced. Purchasing the components and writing the programming code for the sensors were completed. Then, the circuit was built and tested to monitor the temperature, EC and pH of the tank water. The monitoring system featured an application that displays these values in a simple way.

The aim of this internship is to strengthen my existing skills, exercise my theoretical knowledge, and utilize and hone my soft skills to ultimately prepare me for a real engineering position.

ABSTRAK

Kertas kerja itu adalah laporan yang meringkaskan kerja saya untuk kursus latihan industri saya dalam tempoh 12 minggu. Internship saya adalah sebahagian daripada projek Smart Hidroponik Higrowpot yang merupakan kerjasama antara UTM dan SK Taman Tun Aminah.

Objektif utama adalah untuk membangun dan membina sistem pemantauan sistem hidroponik yang dipasang di sekolah. Selepas meneroka pelbagai konsep dan mensimulasikan litar, pelaksanaan projek telah dimulakan. Membeli komponen dan menulis kod pengaturcaraan untuk penderia telah selesai. Kemudian, litar dibina dan diuji untuk memantau suhu, EC dan pH air tangki. Sistem pemantauan menampilkan aplikasi yang memaparkan nilai ini dengan cara yang mudah.

Matlamat latihan ini adalah untuk mengukuhkan kemahiran sedia ada saya, menggunakan pengetahuan teori saya, dan menggunakan dan mengasah kemahiran insaniah saya untuk akhirnya menyediakan saya untuk jawatan kejuruteraan sebenar.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

Aspiring engineers around the globe are required to go through multiple checkpoints that assess their knowledge and understanding. After gaining the theoretical knowledge, technical and cognitive skills from completing all the prerequisite courses; they are put into action in their work field in the third of fourth year of their study. This ensures that fresh graduates are calibrated to the outperform in the competitive nature of today's workspace. Industrial training was designed to provide the means to challenge undergraduate engineering students in a new, professional setting to give them a strong sense of what the real-world work environment, ethics and obstacles look like. For instance, the academic nature of the university cannot replicate a "9 to 5" job after which an employee is not expected to work. Learning to separate their normal life from work life is somewhat challenging as a student.

Furthermore, the exposure of students to the pressure and atmosphere of organizations aids in the development of their interpersonal skills; which include effective communication, collaboration and professional discipline. Interacting with teammates and properly communicating their ideas to them can sometimes be the decider after a trial period of a real job.

Students who have completed industrial training are also expected to have acquired the habit of reporting their moves at constant intervals. This is simulated by the requiring students to submit a log book containing all their daily or weekly activities along with the final report. This, and many other habits are adopted by almost every company, hence it is crucial that graduates are accustomed to the conventions and norms of the professional work-field.

1.2 Objectives

Industrial Training aims to instill crucial lessons in future engineers in their third or fourth year of study. The objectives this course aim to achieve are as follows:

(a) Prepare soon-to-be engineers to the real industrial world.

(b) Test and access the student's ability to convert the theoretical and practical knowledge gained throughout the years into practical, valuable work in a proper way.

(c) Condition the student's mentality to know what to expect from working a real job.

(d) Putting the student's "engineer mindset" to work by solving work's challenges.

(e) Reveal to students the company's expectations of an achieving member.

(f) Provide the means to exercise their interpersonal skills and make meaningful connections that might benefit them during and after their industrial training as an intern engineer.

(g) Give the student's a hands-on experience on professional equipment and machinery commonly used in the work field.

1.3 Scope

Industrial training is considered to be a great preview of a real job as its scope provides a wide range of activities that help achieve its agenda. The scope of this internship revolved around the following:

(a) Communicating to help understand the job objectives.

(b) Project planning and execution

- i) Concept Ideation and selection
- ii) Process planning and optimization
- iii) Task description
- iv) Problem management and resolution

(c) Extensive and intensive research

(d) Study of feasibility

(e) Plan execution

(f) Testing and Troubleshooting

(g) Project submission

1.4 Summary

In this internship, provided by the university, a plethora of tangible and intangible technical, mental and soft skills were exercised and/or established. Ensuring graduates of Universiti Teknologi Malaysia are ready to enter the job market with the experience of a junior employee, a highly competitive pool of acquired skills and the mindset to grow and innovate. The industrial training course is a mandatory course of 5 credits to graduate as an engineer. It provides irreplaceable knowledge and experience that can only be obtained from the workspace.

Being allowed to take on the role of a project manager, planner and execution officer all at once is truly a one-time experience opportunity to be grateful of. This internship demanded the development of a control system of a hydroponics system situated in an educational level. Planning and deciding the project's task alone was an invaluable experience that will surely help develop project managerial skills and an achievement-oriented mind. Researching and self-learning was a huge part of this internship as it exposed the student to unfamiliar devices and induced innovative solutions.

The purpose of industrial training is to forge an aspiring graduate into an effective, competitive engineer to ensure them a high chance in the similarly competitive job market.

CHAPTER 2

ORGANIZATIONAL BACKGROUND

2.1 Organization Profile

The Smart Hidroponik Higrowpot project was initiated at Sekolah Kebangsaan Taman Tun Aminah as an entry to the third edition of the Iskandar Puteri Low Carbon Community Grant offering contest after being approved in the finalist pitching list in 25th of May 2023. The direct funding for this project was from UTM.

Figure 2.1 The Smart Hidroponic Higrowpot Project securing a spot in the final pitching stage of the grant offering.

This initiative is part of the Iskandar Puteri City Council 's response to the community's need for low-carbon ideas and solutions to promote research and collaboration through innovative sustainable low carbon project initiatives. It was meant to aid the community to adopt a low-carbon lifestyle and face the economic challenges presented by COVID-19. It now continues based on the positive impact and communities' response.

Figure 2.2 All the collaborating parties involved in the grant offering run.

This grant offer is a colossal result of the collaborative efforts of Universiti Teknologi Malaysia, The Ministry of Education in Johor, Iskandar Puteri City Council & Regional Development Authority and SWM Environment (waste management company) together with the Public Private Partnership Unit - Prime Minister's Department. Founded in 2021, the first edition offered 200,000 Ringgit while the second and third (current) edition offered half a million Ringgit. The Iskandar Puteri Low Carbon Community Grant 2.0 was proudly received by Universiti Teknologi Malaysia last year; being awarded an amount of 83,000 Ringgit.

The Smart Hidroponik Higrowpot project located at the backyard of SK Taman Tun Aminah aims to establish a Nutrient Film Technique hydroponics system that features solar power renewable energy while injecting it with an innovative Internet of Things control system.

2.1.1 Organization Information

Figure 2.3 Universiti Teknologi Malaysia & Smart Hidroponik Higrowpot logos

(a) Company Name: Smart Hidroponik Higrowpot Project for IPRK 3.0 MBIP hosted by SK Taman Tun Aminah, funded & managed by UTM.

(b) Location: SK TAMAN TUN AMINAH, JLN PAHLAWAN 4, Johor Bahru, Malaysia, 81300

(c) Email: hidroponikskttautm@gmail.com, sktta.shh@gmail.com

(d) Contact No.: +60 11-1633 4296

2.2 Industrial Training

The industrial training provided by the Smart Hidroponik Higrowpot project delegates the development, execution and testing of the monitoring and control IoT system of the hydroponic system while maintaining the clean solar power integration provided by the panels installed.

This training is aimed to develop my project management, research and life-long learning skills. It allows me to go head-first to take on all the challenges that might emerge in the journey to the completion of the project. Certainly, the lack of a person to guide me through the project requires great demonstration of self-discipline, commitment, perseverance and communication skills to be able to satisfy the objectives of the industrial training.

The system is required to be installed by the end of the internship; and continue to be upgraded until January. However, it was delayed due to unforeseen issues with the final system despite prior functional testing.

2.3 Organizational Structure

Figure 2.2 UTM FKM Organizational Structure.

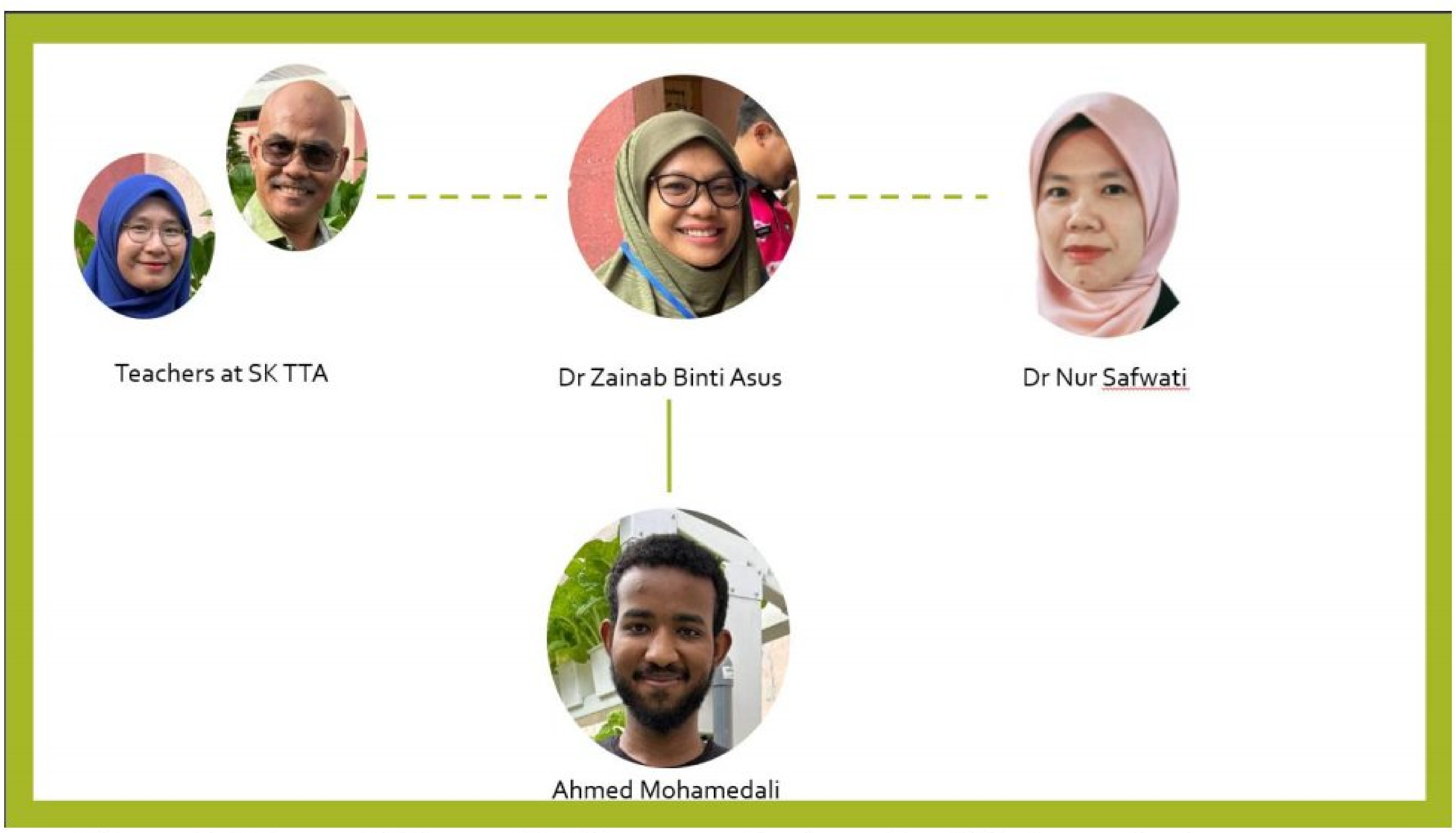

Figure 2.3 Smart Hidroponik Higrowpot Project simplified member's structure.

CHAPTER 3

DESCRIPTION & EXPERIENCE

3.1 Description

Figure 3.1 Four NFT Hydroponic systems installed on day one of training.

Four towers of Nutrient-Film-Technique-type (NFT) hydroponics system were installed at SK TTA. Throughout the internship, working on developing this monitoring system has proven to be the best way to learn about microcontroller boards, several types of sensors, optimizing circuits and the code needed inject the project with a dose of IoT. The project supervisor, Dr Zainab Binti Asus has allowed me to freely utilize all my skills, and experiment by giving me the space to plan the entire project, hence leading the transformation of our ideal concept into reality. Exercising my interpersonal skills with Dr Nur Safwati was pivotal to benefit from her experience in electronics components. Through her guidance, the concept selection process was simplified and the limitations of the various components were identified swiftly.

After visiting the school and inspecting the hydroponics system, I immediately started brainstorming concepts that would satisfy the given objectives. It was essential to ask a plethora of questions to draft guidelines and criteria for my concept selection process as well as understand more about the limitations in different aspects of the project such as budget, client's preference, etc…

The first two weeks of my internship involved deep research on the topics of microcontroller boards, hydroponic systems and the various sensors that will be used to obtain the monitoring data. Additionally, a quick refresher on Arduino coding - based on C plus plus programming language - has proved helpful. A plethora of websites, research papers and FYP reports were scoured for relevant information. A system featuring four hydroponic towers was anything but common.

The first step I took towards building the monitoring system was to sketch flowcharts outlining all the possible concepts. Attending a short course on Embedded IoT systems at the electrical engineering faculty, presented by Dr Waleed Ejaz - a professor at Lakehead University, Canada - have helped me tremendously towards my exploration of concepts for the hydroponics system. By applying the knowledge from Introduction to design course, a morphological chart was created to generate more ideas from the initial concepts. Following that, the selection process was simply to list out the advantages and disadvantages of every concept to filter out the absurd ones. The screened concepts were then presented to my supervisor - Dr Zainab - following my consultation with Dr Nur Safwati.

The approved concept was then transformed into a circuit to further verify its feasibility and reveal the components that should be ordered. Finding suitable quality parts for cheap prices proved to be unexpectedly time-consuming; but it was required to ensure the project remains within the budget range. While I awaited the arrival of the ordered components, I started writing the code that collects and processes the sensor data.

Finding a suitable and free website to display the data in a mobile app form was truly challenging. A friend of mine was of great help in finding the MIT application inventor that was free, highly customizable and had plenty of features. All the other means of uploading the data in a presentable way were either paid, had limited features or was unfortunately put to its End of Life recently (Cayenne IoT).

3.2 Gained Experience

This internship was definitely a journey packed full of new and invaluable experience until the last bit of it. The project called for extensive research and hands-on with new components. It featured a “Frankenstein-like” board joining an Arduino Mega 2560 and ESP 8266 Wi-Fi chips into one. The decision to use this board, although highly beneficial, came with its own set of challenges due to the scarcity of documentation on how to harvest its features. However, it proved to be worth the time and effort once it was thoroughly explored.

Figure 3.2 Arduino Mega 2560 + ESP 8266 development board.

I have also gained highly beneficial C/C++ programming experience. Most of the sensors required the use of specific libraries to help manipulate and process the raw data obtained. Some had readily available documentation and examples, while others demanded heavy scavenging around the internet to understand the operation of different functions in the library and be able to integrate it efficiently and effectively in the Arduino code.

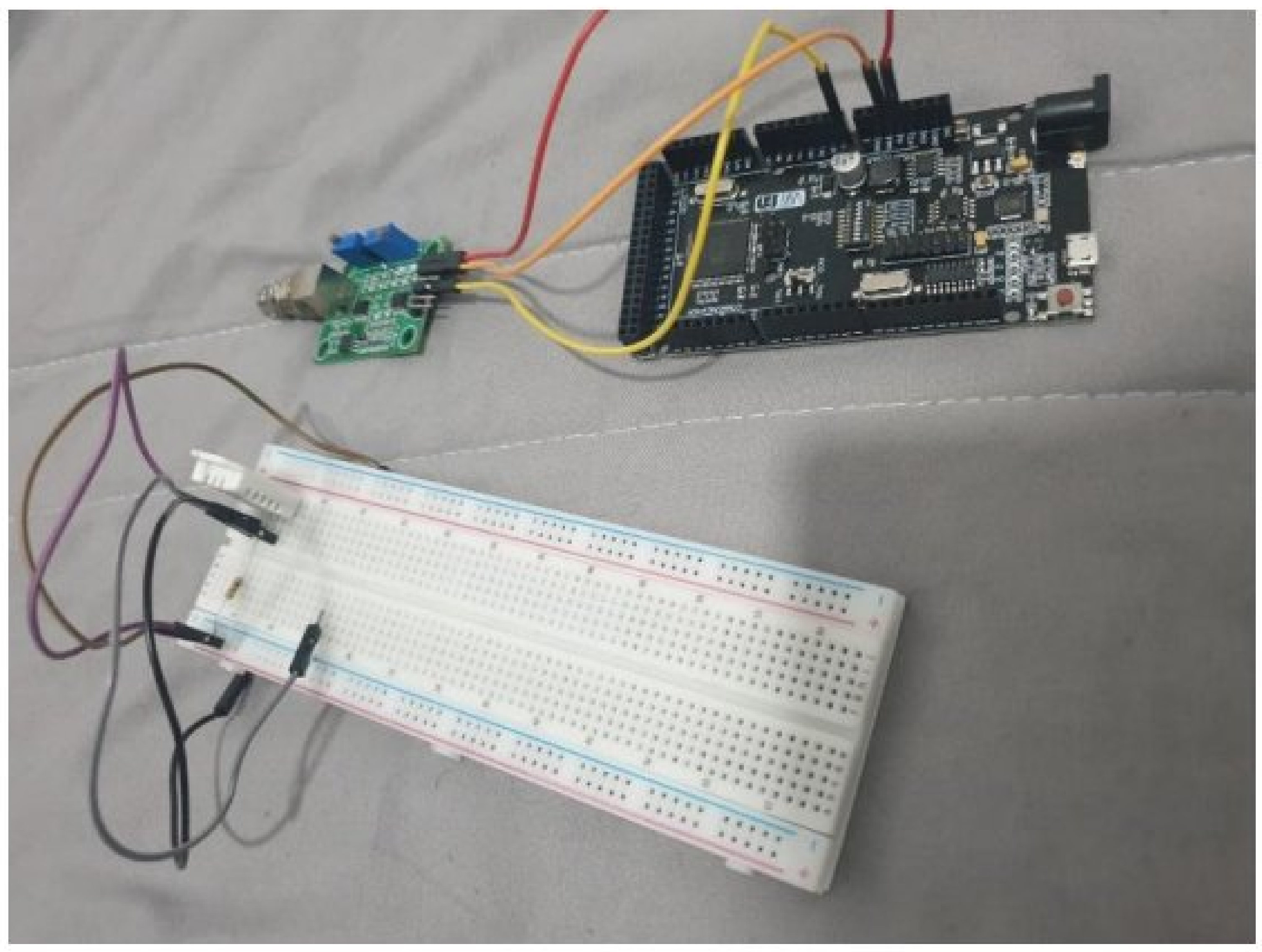

Figure 3.3 pH module connected to the Arduino Mega + ESP board for calibration.

Working with analogue sensors further solidified the knowledge obtained from the Instrumentation course; as calibration and error offset were implemented. Moreover, the process of presenting the data in a user friendly, easy to understand application required more learning and researching. Receiving assistance from a friend of mine; the means by which the data can be relayed to the app could be listed and filtered for the most feasible option. MQTT was the method of choice as, while being quite challenging, it was found to be of lower load to the processor, which will be running continuously.

In addition, through communicating with one of the teachers at SK TTA, a valuable piece of knowledge was revealed. Mr Zul has alerted me to the issue of rapid rusting of the water level sensor. This seemingly minor detail has saved both money and time by avoiding the component damage and implementing his suggested solution.

Figure 3.4 Meeting and chatting with Mr Zul at SK TTA.

Being tasked to design posters for the project has further strengthened my graphical designing skills. Finding high resolution free-use images and vectors was almost impossible; which required me to design them manually on Adobe Illustrator.

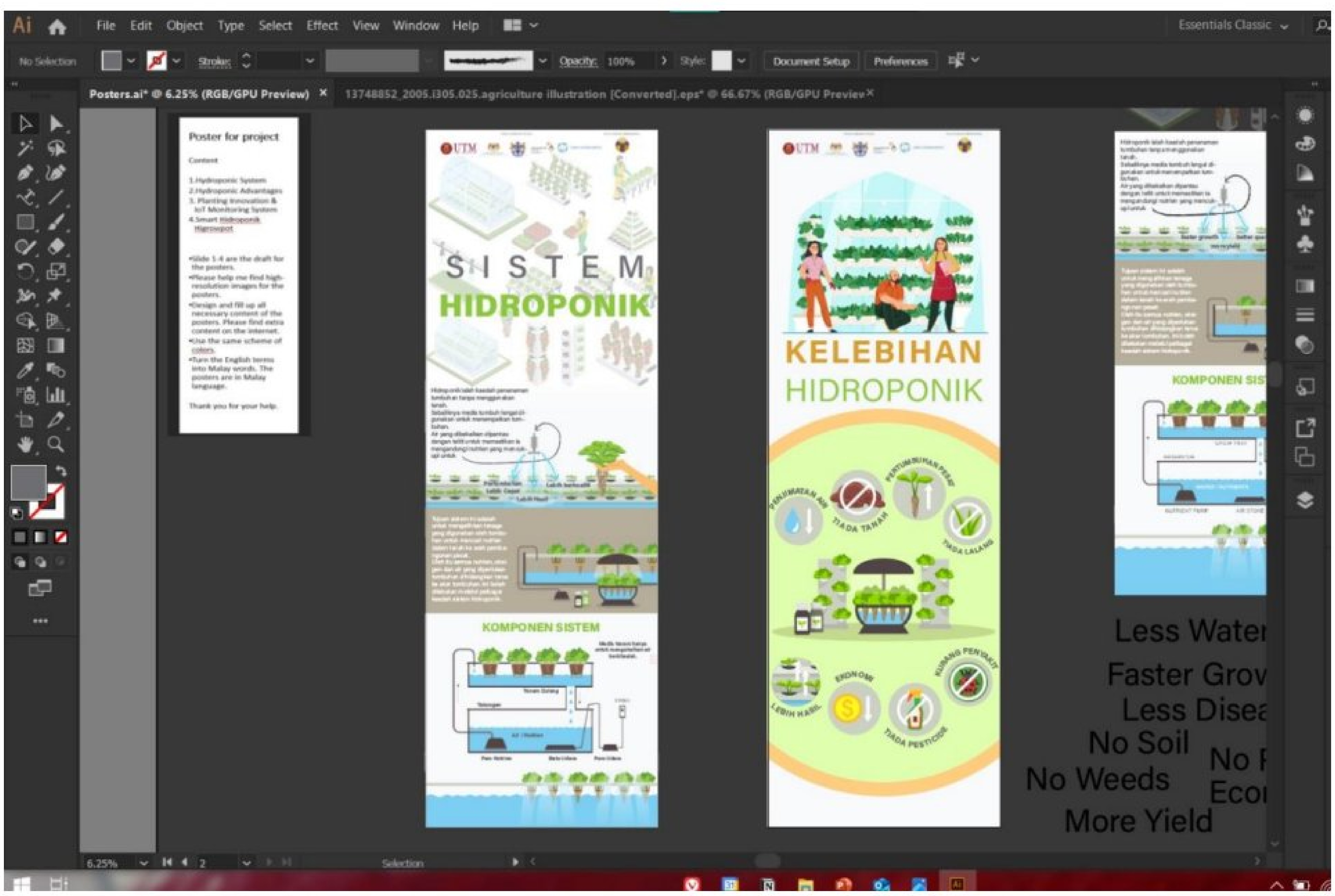

Figure 3.5 Designing of project posters on Adobe Illustrator.

CHAPTER 4

DETAILS & DISCUSSION

4.1 Introduction

As previously mentioned, the project is essentially an entry to the Low Carbon Community grant offering competition from Iskandar Puteri City Council. The project revolves around installing and innovating a hydroponics system at SK Taman Tun Aminah while using clean energy. Solar panels are installed in all four systems which provide the power to all the pumps.

A hydroponics system is a way of planting vegetation without the use of soil. This means the medium in which the plant roots grow in is neutral and hence requires intricate monitoring and care to ensure the growing conditions are suitable. The parameters that has to be monitored are the temperature of the water, the electrical conductivity (EC), the acidity (pH) and the tank water level. The electrical conductivity of the water is measured in parts per million to determine the concentration of fertilizer dissolved in the water tank. This parameter is extremely important to ensure the plants are not deprived of any nutrients and the hydroponics system is able to produce it's expected high yield. Furthermore, the pH of the water pumped into the system must also be maintained within an acceptable range of 6 to 7. Lastly, the water amount in the tank is monitored to prevent the pumps from getting damaged by operating at a low water level; which will increase the cost and reduce the reliability of the system.

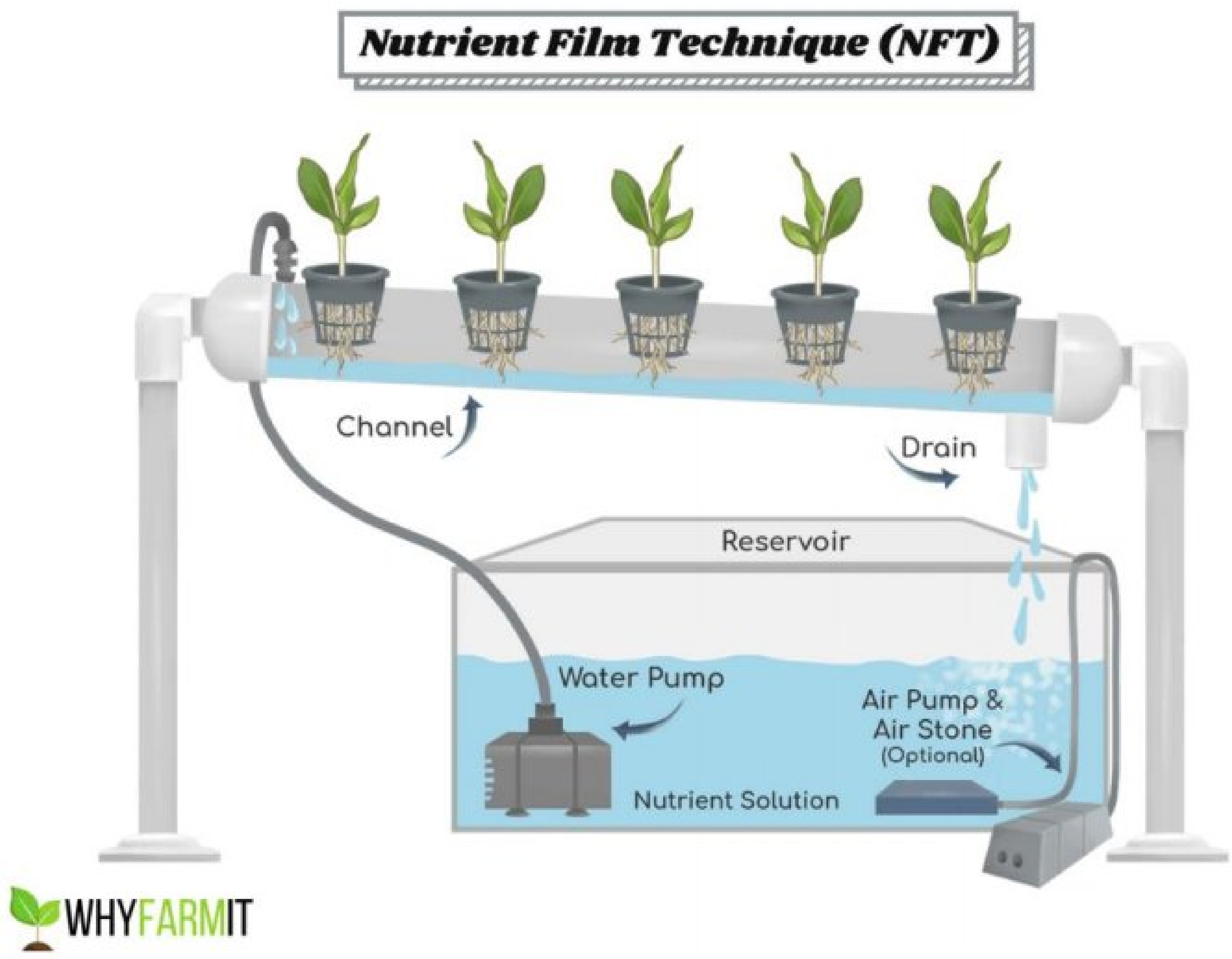

Figure 4.1 NFT Hydroponics system diagram.

There are many types of hydroponics systems; Deep water culture, Aeroponics, Wick and Nutrient Film Technique (NFT) Hydroponic systems to name a few. The last example, the NFT Hydroponics system, is the type used in this project. The pump raises the nutrient-rich water to the highest tray over which the plants are suspended. The water flows through the top row of plants and drops down to the one below it until it reaches back to the tank where it is cycled again into the system. This process ensures all the plants are exposed to sufficient nutrients and water to grow at a rapid rate.


Figure 4.2 Fast growth and high yield from Hydroponic system at SK TTA.

The hydroponics system is expected to present a plethora of benefits that count towards more sustainable cultivating methods. According to online resources, the use of hydroponics systems saves plenty of water since it is constantly circulated. Secondly, it completely avoids the growth of intrusive, unwanted weeds that normally would hinder the growth of the cultivated plants. In addition, the harvested products obtained from hydroponics systems are proven to be less prone to diseases. This is in direct correlation to the lack of pests and hence the omittance of pesticide use on the crops. All of these benefits are wrapped up with higher yields and faster growth, thus making it economically feasible and highly beneficial if executed correctly.

As aforementioned, hydroponics systems require monitoring to avoid issues and ensure effectiveness. This part in specific was the main topic of this internship. The following part will discuss the detailed objectives of the project.

4.2 Objectives

The aim of the monitoring system was explained in my first visit to the project site (at SK TTA). They can be summarized as follows:

(a) To obtain and prepare the water temperature, EC, pH and water level sensor values. The outside temperature and humidity is considered a good addition.

(b) To display the sensor data in the form of a user-friendly application targeted at junior pupils at the school.

(c) To enable the monitoring of the sensors from anywhere.

(d) To notify the user if an issue arises; be it the pump, fertilizer (EC) level, etc..

(e) To allow the manual control of the pumps from the application.

It must be noted that the first 3 objectives are the main ones; while the remaining are considered future upgrades that must be accounted for in the system development process.

4.3 Project Details

4.3.1 Concept Ideation

The monitoring system is required to collect the data from four sensors from four hydroponic system towers. This means the data from about sixteen sensors is expected to be managed, analysed, stored and displayed on the application.

According to the instrumentation method, the sensors data are sent to a microcontroller board to be processed before a usable output is obtained. There are several ways this can be achieved. As mentioned before, a morphological chart was constructed to compare and contrast between the concepts as well as generate more optimized variations.

Activities/Options:	A	B	C	D	E
Collect the data	Sensors	Sensors	Sensors	Sensors	Sensors
Transfer the data	Wires	Wires	Wires	Wires	Wires
Receive the data	1 Arduino Mega2560 & 1 ESP8266 wifi board	1 (2in1) Arduino Mega + ESP WiFi board	1 Arduino Giga board	Raspberry Pi board	4 Arduino Unos & 1 ESP 8266 WiFi board
Process the data	Arduino Mega 2560	1 Arduino Mega + ESP WiFi board	1 Arduino Giga board	Raspberry Pi board	ESP 8266 WiFi board
Store the data	SD Card	Arduino Cloud	Google Sheets	Blynk Local Server	Google Sheets
Display Medium	MIT app inventor	Arduino IoT app	Grafana	Blynk	Google Sheets

Table 4.1 Morphological chart.

As can be seen from the chart, there are 5 concepts that were brainstormed outlining the way the data is collected, transferred, received, processed, stored and finally displayed.

Concept A featured connecting all the wired sensors to an Arduino Mega 2560 board with an ESP 8266 Wi-Fi module board attached. This allows the data to be transferred to the cloud by the ESP board after getting processed at the Arduino Mega. Having two microcontroller boards is beneficial as the load is somewhat shared between them; which is a factor worth considering since the monitoring system is expected to be running almost 24/7. The storage method was suggested to be on an SD Card module to serve as local backup and future reference to further analyze the data if needed. The display medium could be the MIT App Inventor 2 application as it features a wide range of features, almost complete customizability and ease of use. It can receive data from an MQTT server, Firebase account or via a webpage.

The second concept B can almost be called a variant of A. The board that could be used to receive and process the sensors raw data is the “Frankenstein-like” board joining an Arduino Mega 2560 and an ESP 8266 Wi-Fi. The concept utilizes the IoT cloud service provided from Arduino. However, this method of storing data is hindered by the freemium nature of the Arduino Cloud platform as it will not retain it more than a day.

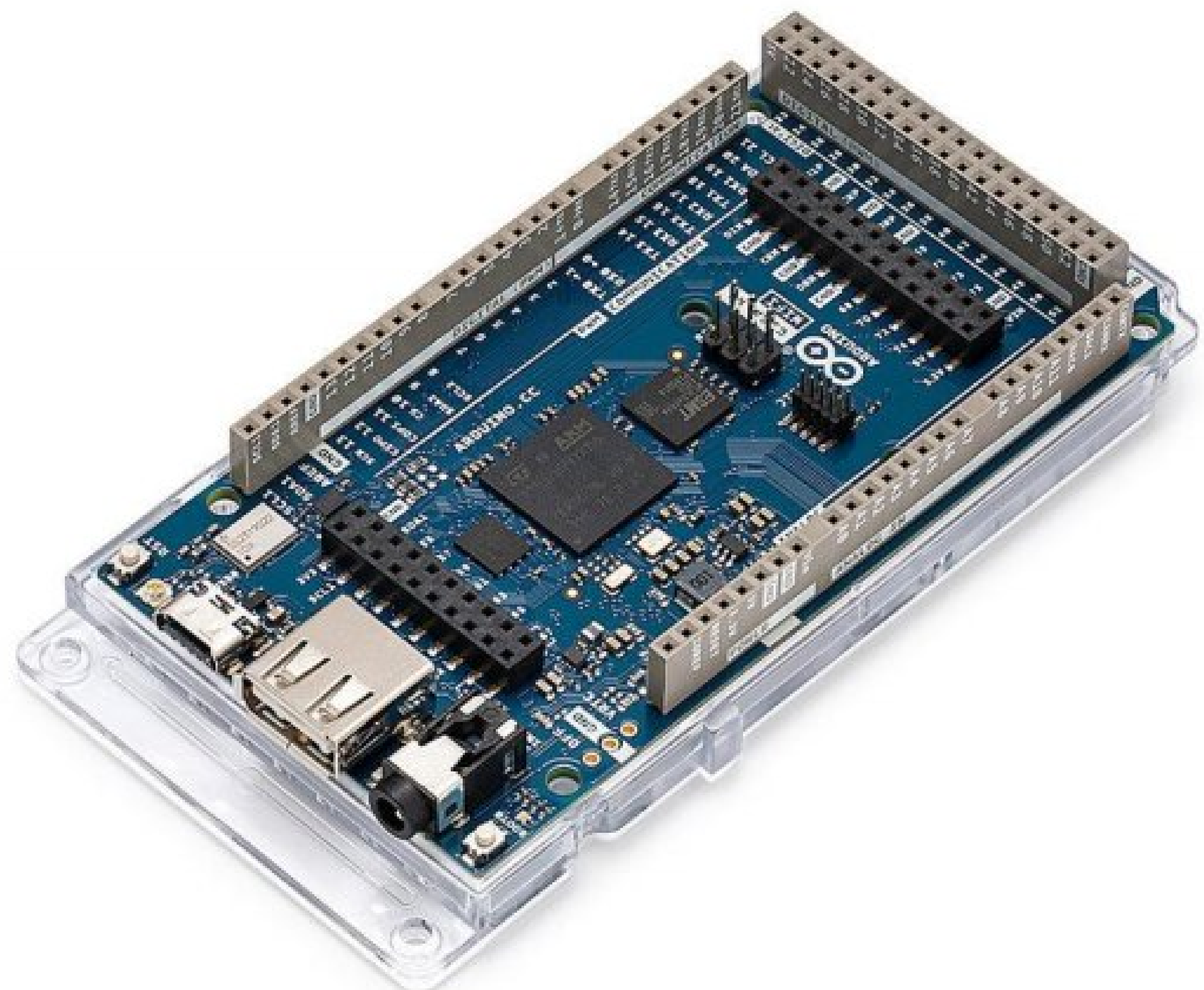

Figure 4.3 Arduino GIGA Q1 Wi-Fi

The next concept (C) features the all-new board that was recently released by Arduino - the Arduino Giga R1 Wi-Fi. This colossal board integrates all the required functionality into one board. It was suggested to be used to receive, process and upload the sensors data to Google Sheets and Display the monitored parameters in a more visual way using GrafanaLabs.

Concept D suggests the use of Raspberry Pi board with Blynk’s pre-built application. This board is able to handle a local Blynk server on which the processed sensor output can be stored and relayed to the application in a user-friendly way.

The fifth concept (E) is was one of the first that came into mind when I was inspecting the hydroponics system on site. The idea is to connect the sensors from each of the four towers to a separate Arduino Uno board. Then connect all the Uno boards to a single ESP 8266 board that will act as a Wi-Fi module to all of the Arduino's of each tower; allowing it to send the data to Google Sheets. The display medium can also be Google Sheets as it provides many graphical and visual widgets that can be designed in a similar way to an application.

4.3.2 Concept Selection

Concept C and D utilize the Arduino Giga R1 Wi-Fi and the Raspberry Pi respectively, both of which are of high cost. The budget in hand for the development of the monitoring and control system was found to be unaccommodating for such investments. This meant the disqualification of these concepts.

Moreover, researching the feasibility of concept E, the idea of using a separate Arduino Uno for each tower seemed logical and straightforward, however, connecting all the boards together was found to be a complicated and unreliable route for this project.

Furthermore, using Google Sheets and Grafana to present the output data appears to be certainly suitable, as it is free and cloud based. However, these methods ignore the fact that the system has to be able to control the pump; which requires a two-way connection: sensors output and user control input.

As mentioned before, using the Blynk IoT or Arduino app, although popular among projects of this kind, it is not viable for this application due to the high sample size and low number of variables allowed on both IoT platforms.

The morphological chart allows for more concepts to be generated by mixing the best of different concepts. Concept B appeared to be almost fitting to the project criteria except for the Arduino Cloud limitation. This can be resolved by simply using an SD card module instead to store the data locally. Since the use of Arduino Cloud is now omitted, the Arduino IoT application cannot be used now. The MIT app inventor, as aforementioned, has an arsenal of extremely useful features; and best of all: it is free to use!

Following discussions with my supervisor, the final concept was approved, hence I was cleared to begin the execution of the project.

4.3.3 Project Execution

The commencement of the project execution involved more research into the selected concept. The first step was to understand and visualize the flow of data from the sensors all the way to the display medium. A flow chart was constructed to aid in that.

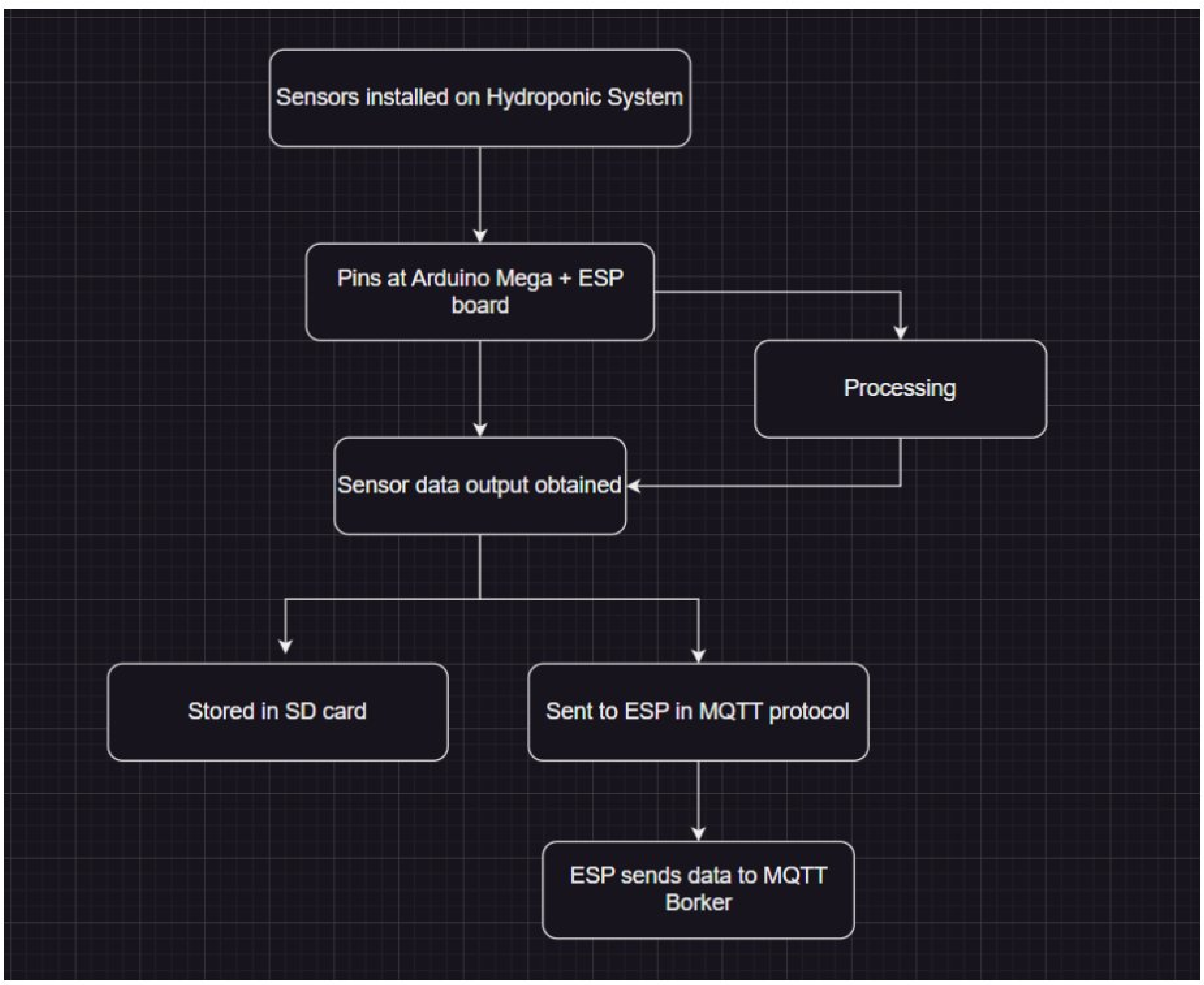

Figure 4.4 Sensor Data flow chart.

The following step was to determine all the components of the system and the circuitry that connects them together down to the pins expected to be used. A computer program called Fritzing was used to visualize and simulate the circuit.

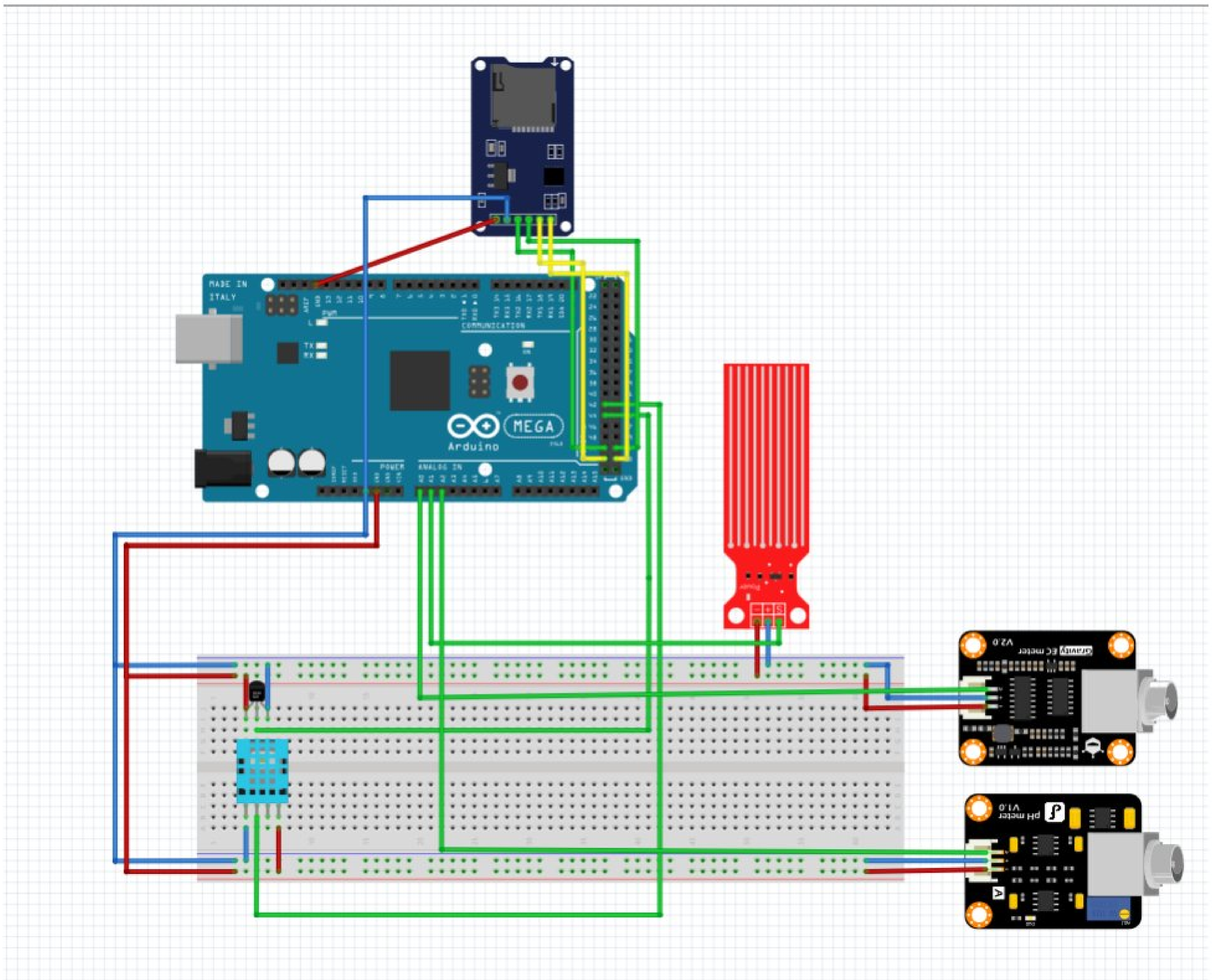

Figure 4.5 Fritzing simulated circuit for one tower

Referring to the figure above, the black modules shown on the bottom right corner are the analogue EC and pH sensors. The SD card module on the top was found to require an SPI connection to the Arduino board; which involved 4 data wires apart from the power and ground.

After determining the required components and their connections and presenting them to my SV, the parts can now be ordered. Finding the suitable parts was not the hardest task, however finding them cheap was somewhat of a challenge. It took some time to find cheap but well-reviewed parts; buying the pH sensor probe from one seller and the controller module from another to save money.

List of Components		
Units	**Name**	**Price per unit (RM)**
1	Arduino Mega 2560 + ESP 8266 Wi-Fi	52.00
4	TDS (EC) Sensor	19.99
2	pH Sensor Module	14.99
2	pH Electrode Probe	22.65
2	Breadboard	3.80
2	40 Pcs connector jumper wires M2M & M2F	3.70
3	10 Pcs 4.7kOhm resistors	1.00
1	Kingston 16GB SD card	11.9
4	5 Meters Triple-core Electronic Wire	6.50
4	Water Level sensor	2.37
4	Waterproof DS18B20 Temperature Digital sensor	7.10
1	SD card module	3.20
1	DHT digital sensor	10.00
4	Float switch water level sensor (tank)	9.60
1	pH buffer solutions (4, 7 & 10)	21.00

Table 4.2 List of components

After ordering all the parts from Shopee, I proceeded to start writing the code that will program the Arduino Mega+ESP board into life.

4.3.3.1 Coding

Programming the arduino board was the most tedious and time-consuming task in the project. The flow chart explaining the general the flow of the code is presented in the figure below:

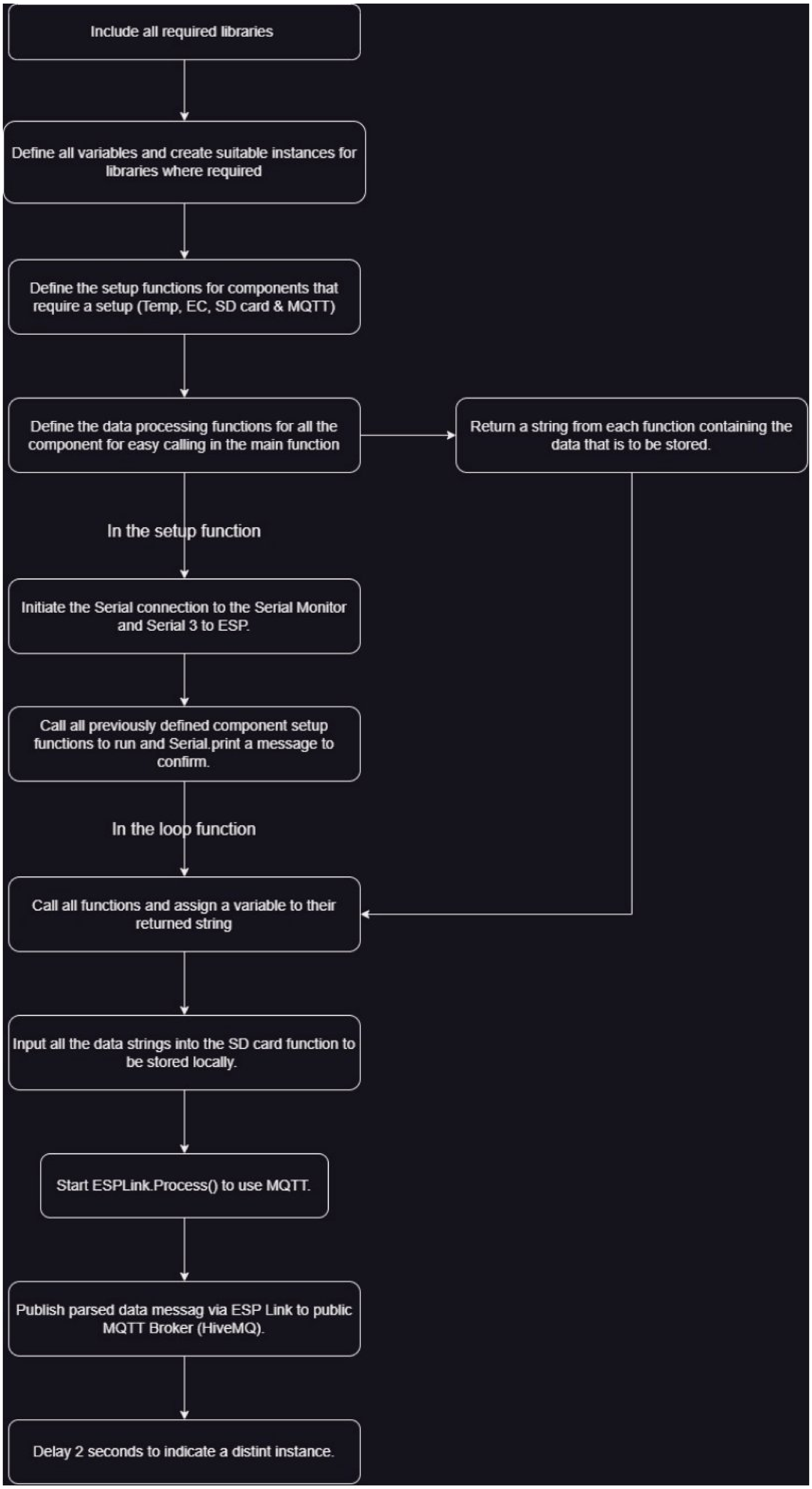
Include all required libraries
Define all variables and create suitable instances for libraries where required
Define the setup functions for components that require a setup (Temp, EC, SD card & MQTT)
Define the data processing functions for all the component for easy calling in the main function
Return a string from each function containing the data that is to be stored.
In the setup function
Initiate the Serial connection to the Serial Monitor and Serial 3 to ESP.
Call all previously defined component setup functions to run and Serial.print a message to confirm.
In the loop function
Call all functions and assign a variable to their returned string
Input all the data strings into the SD card function to be stored locally.
Start ESPLink.Process() to use MQTT.
Publish parsed data messag via ESP Link to public MQTT Broker (HiveMQ).
Delay 2 seconds to indicate a distint instance.

Figure 4.6 Arduino Code flow chart.

The code consists of functions that each perform a certain task. All the functions are then called in the setup and/or the loop functions (main Arduino functions). The sensor output variables were all made global in order to make them usable outside of their process functions.

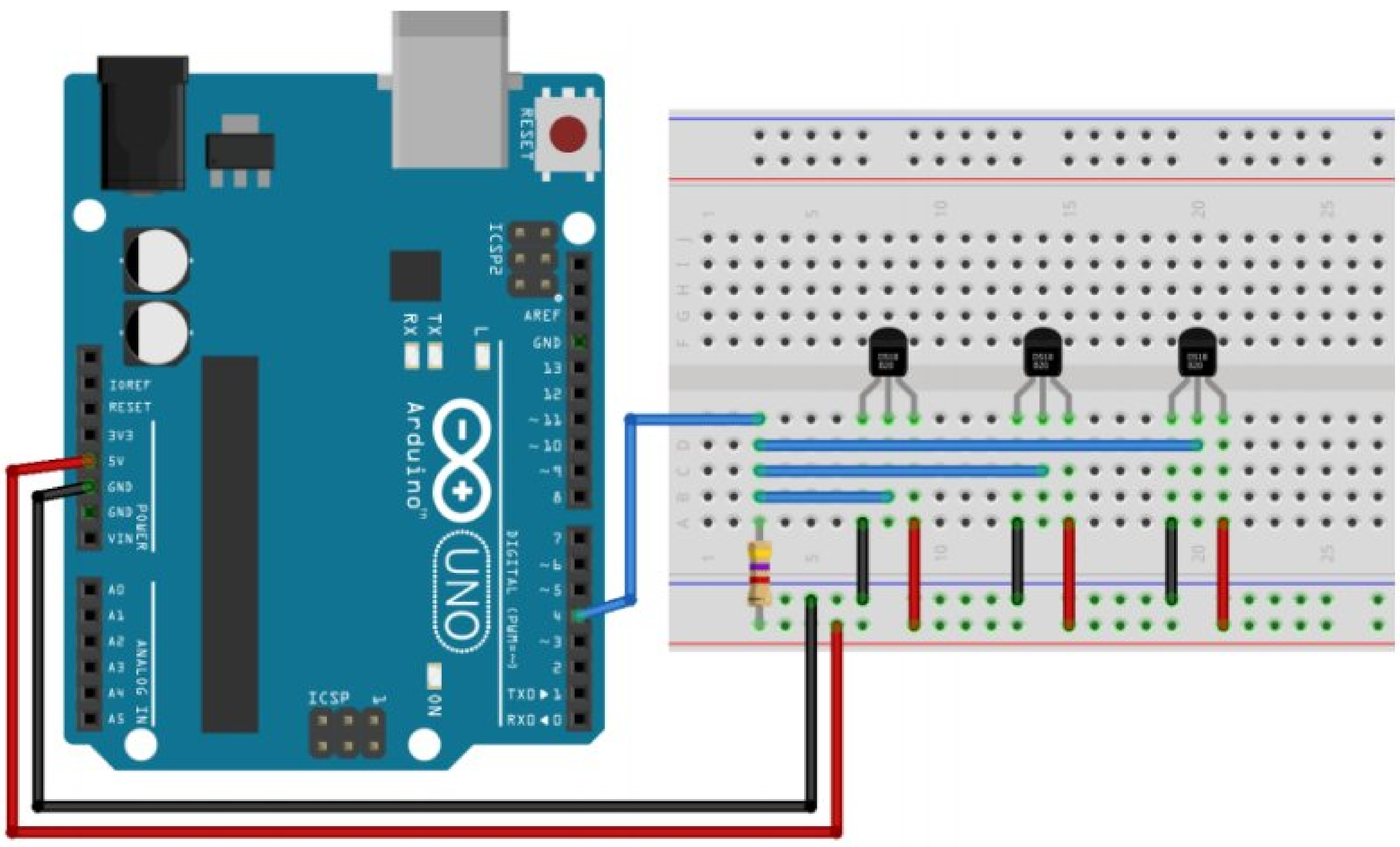

Figure 4.7 Onewire I2C bus circuit.

The temperature sensor was found to be utilizing an I2C bus connection in order to send all the data to a single pin on the Arduino. The main reason the Arduino Mega was chosen was due to the required number of analogue pins which are limited in the Arduino Uno variant of the board. The temperature sensors used were of the waterproof model DS18B20; which had plenty of documentation on the arduino library suitable to obtain the temperature output but more importantly required no signal conditioning as it is a digital sensor. The library used to process the data - Onewire.h and DallasTemperature.h - called every sensor in the bus and stored the value to an index in the Temp list variable.

The EC sensor also came with sufficient documentation to understand the way by which the raw analogue data is manipulated by the supported arduino library. The GravityTDS.h library, which was made for the Gravity EC meter kit from DFRobot, has single-handedly taken care of processing the data despite the sensor not being of the same brand.

Moving on to the pH sensor, the case was a bit more challenging compared to the previous two sensors as there were little to no information on the module. There were no libraries found to work with the module in hand, which was of lower quality compared to other options due to the vast price difference. The raw input data from the pH probe was to be calibrated and processed to output the pH value of the tank water. After further research and more YouTube videos watched, the procedure to calibrate the the module was determined. The BNC connector must be first short circuited by connecting the positive and negative to each other. Then, the module is connected to the arduino and the output value is observed on the serial monitor. The idea is that the value must be centered in the middle of the 5V range of the output.

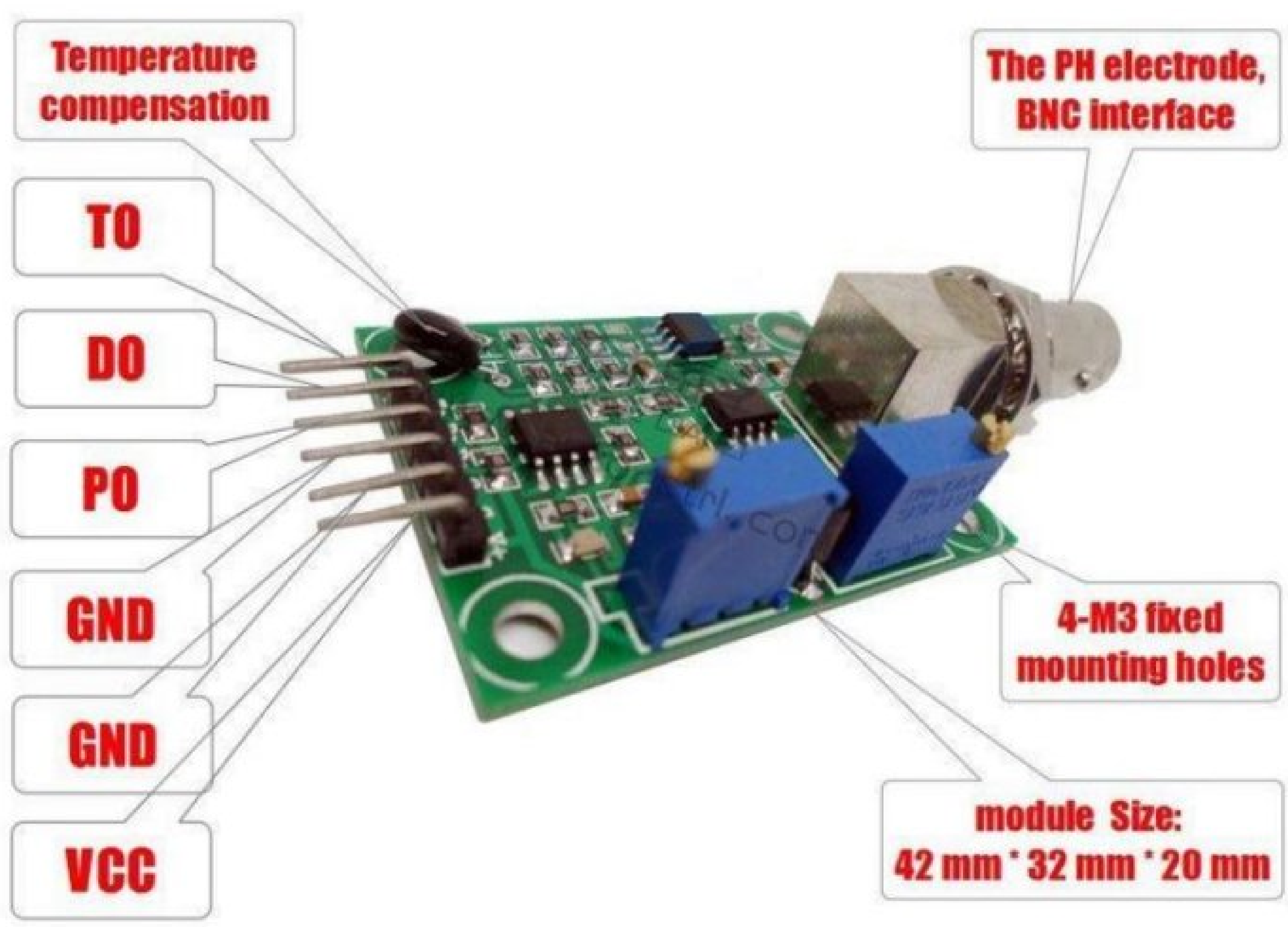

Figure 4.8 pH module.

The module features 2 potentiometers; the one near the BNC connector controls the offset while the other is to configure a limit feature, which wasn't particularly of our interest. The first potentiometer is turned until the output voltage value on the serial monitor reads 2.5 V - half of 5V. This is where an issue arises; upon turning the potentiometer, it has reached its end while the value has only reached 2.7 V. The solution was to simply adapt to the error by applying an offset and adding it to the calibration offset in the code. After connecting the pH probe to the module; the pH is put into a buffer solution of a known pH and the calibration offset is known by the difference between the buffer solution pH and the output value from the sensor.

The coding of the water level sensor was delegated to my friend. He has written the function that will receive the analogue data from the water level sensor and process it to return a low or high level. The water level sensor here is not meant to indicate the level of the water in the tank as it is small in size and cannot be fully submerged. The sensor will be used to indicate the water level inside the tube through which the nutrient-rich water flows.

As indicated in the BOM table, a DHT22 sensor is used to determine the outside temperature and humidity at the hydroponics system. The connection of this sensor required a pull-up resistor to ensure reliable readings. This component too was delegated to my friend to program. It had quite decent documentation and several libraries to choose from to obtain the temperature and humidity values.

The SD card saving function demanded more learning of the SD card functions from Arduino built-in library in order to be able to manipulate the saved files as desired. Saving the data file and reading from it is ensured as implementing error catching methods in the code meant that the data is successfully stored; otherwise, it would give an error on the serial monitor.

After finishing the code functions for each of the sensors, which were made to also return a string that can be saved to the SD card in a proper format, the next step was to send the data to be displayed.

The ESP 8266 that is imbedded in the board should be configured to relay the output data via Wi-Fi to the chosen destination where it will be displayed. The commonly known way to use the ESP board as a Wi-Fi module was to utilize the ESP8266WiFi.h library to connect to the Wi-Fi while receiving and parsing the incoming sensor output data from the Arduino Mega. Attempting this method has resulted in many issues and plenty of hours wasted on troubleshooting to no avail. Therefore, the ESP-Link firmware was found to be an easy way to connect to the Wi-Fi seamlessly and act as a serial monitor for the Arduino Mega thanks to its SLIP protocol. This firmware also allows for an easy integration of MQTT connection and can send the data to a web page; which is a great upgrade route in the future. As a result, the decision to flash the ESP 8266 with the ESP-Link firmware was executed.

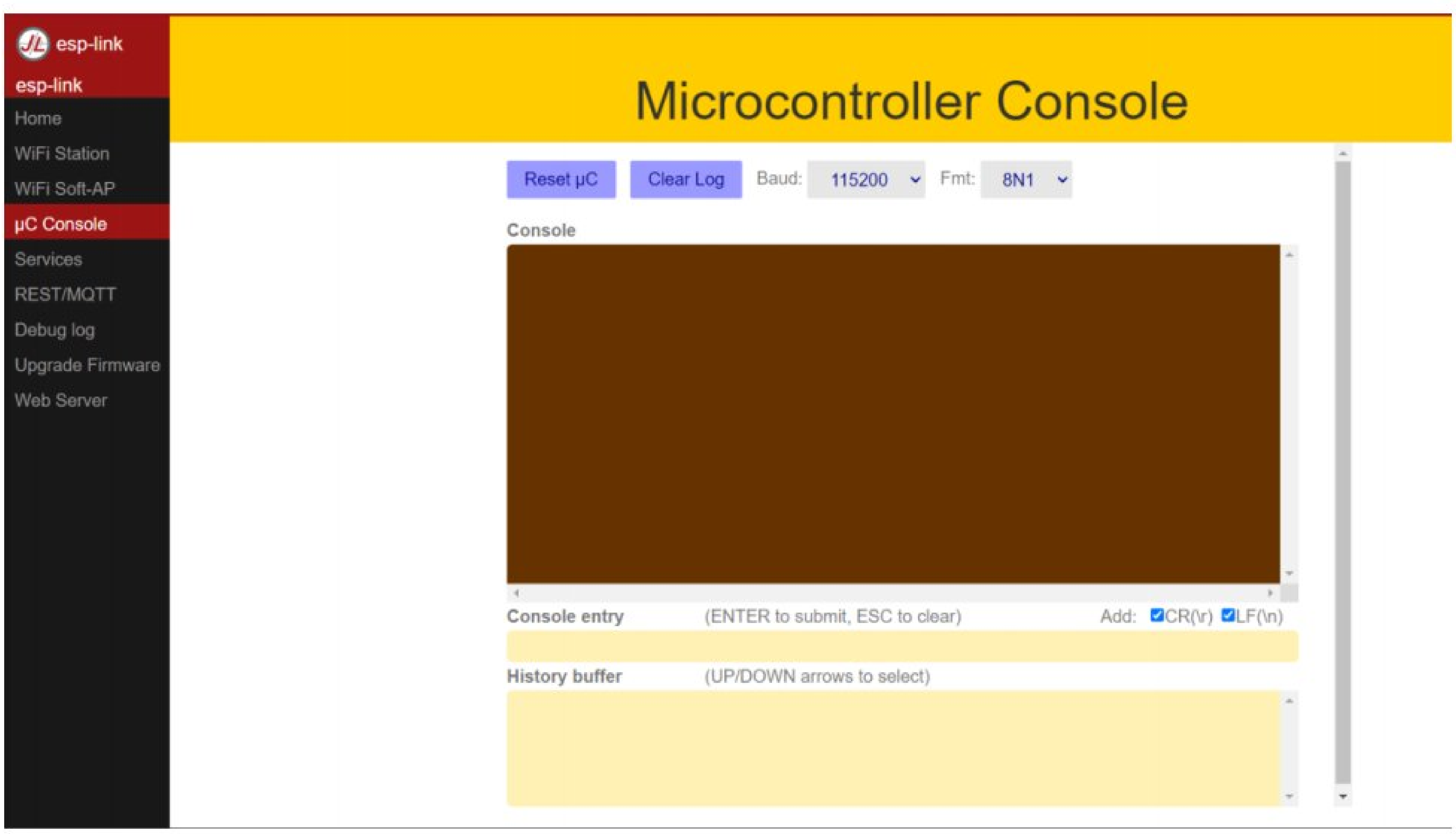

Figure 4.9 ESP Link relayed serial monitor from the Arduino Mega.

There were several options on the table when it came to sending the data to the MIT App Inventor. The first being via updating the data to a web path that can be retrieved by the application. This method inhibited the same issue of using Google Sheets in the sense that it is only a one-way connection; where a two-way connection is needed to support the control of the pump later on.

The second option was to use Firebase; which is a back-end mobile application service from Google. This platform presented a similar solution to the first option but the "tag" (similar to a web path) can also be updated by MIT App Inventor application, meaning it becomes a two-way connection that allows the pump control feature upgrade to the hydroponics system. This method is not supported by ESP-Link firmware and is considered to be resource heavy in terms of network connection speed requirement and microcontroller load to operate smoothly.

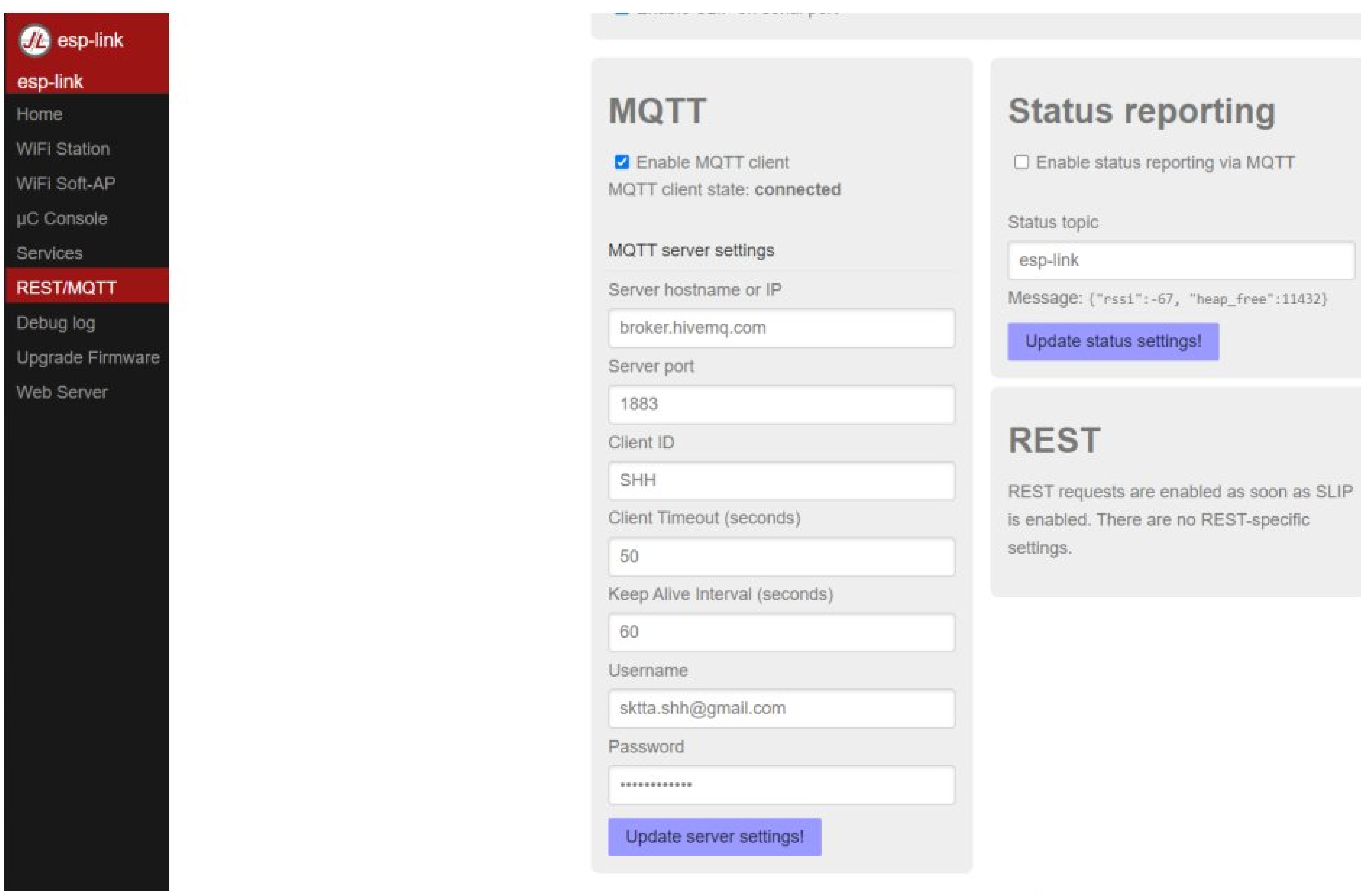

Figure 4.10 MQTT integration via ESP Link.

The third option was to use MQTT which is considered a standard in the world of IoT. MQTT or Message Queuing Telemetry Transport is a simple, publish-subscribe network protocol for message queues and message queuing services. This protocol is lightweight, which suitable considering the location of the hydroponics system is in an area with extremely low Wi-Fi signal and high traffic being at a school. This network consists of clients who subscribe and/or publish to a topic. This different topics and messages are all managed and kept up-to-date by the MQTT broker. For this project, a free broker was used ensuring a unique topic was chosen to avoid any disturbance. The chosen broker is HiveMQ.

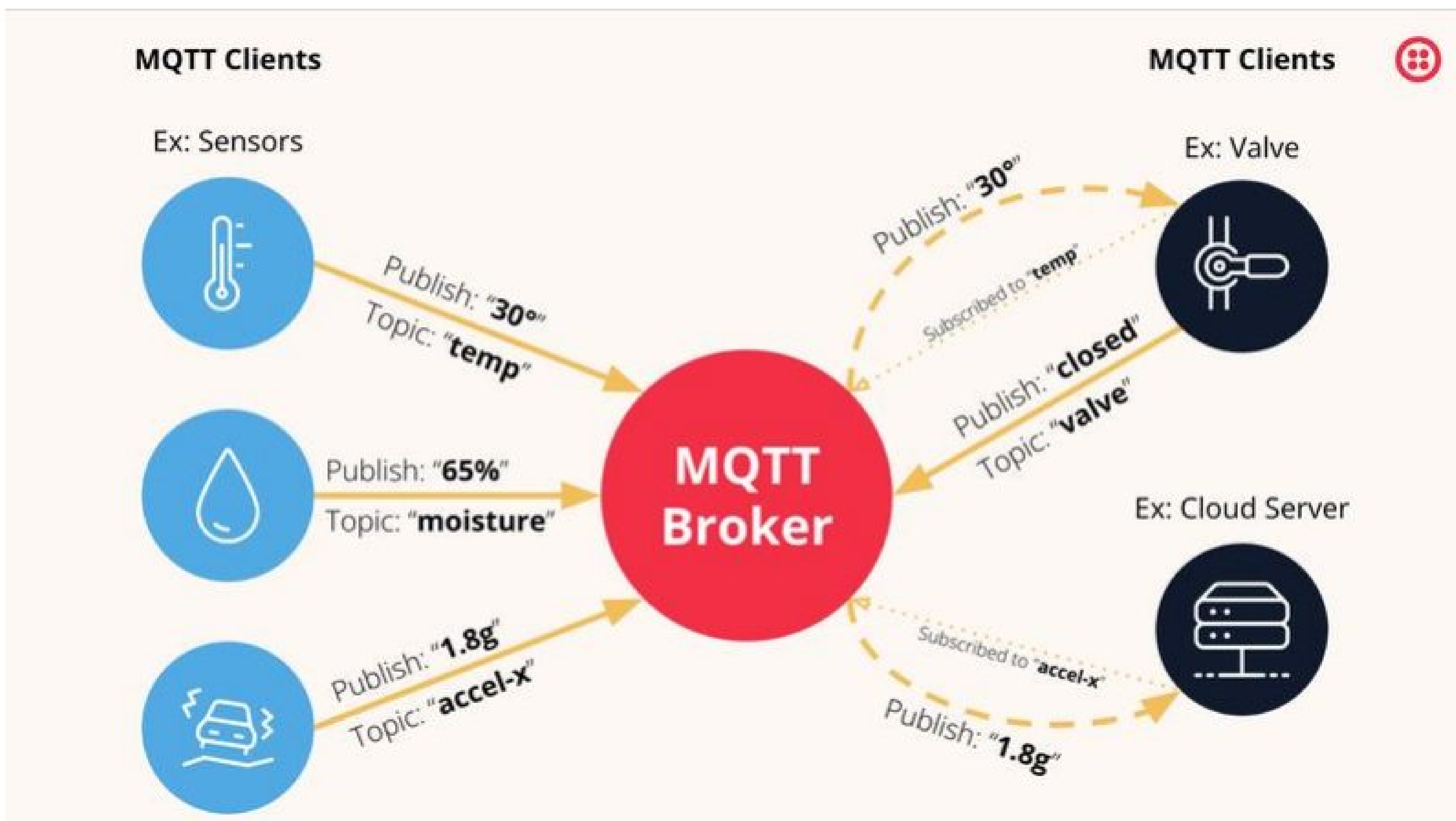

Figure 4.11 MQTT Concept visualization diagram.

The integration of MQTT protocol into the code was largely assisted by a website from GrowBoxGuy - a DIY Guide for Indoor gardening. The MQTT code was largely inspired by his project that utilizes the ESP-Link's MQTT support. The topic that stores the sensor output data was named SensorsData/SensorsData/ on Quality of Service 0. After testing the code and ensuring the board is updating the data message to the topic; it was time to "invent" the mobile application that will retrieve the data from the topic and display it.

4.3.3.2 Mobile Application

The MIT App inventor provides two modes of editing. The first mode is the designer mode, where the application appearance is made as desired. The second mode is the back-end Blocks mode, where all the behind-the-scene operations of the app is programmed.

The design was made to be simple a straight forward, indicating the temperature, EC, pH and water level of each hydroponics system tower. A button to connect to the MQTT broker to update the data in real time is presented at the top. The text on the button changes to “Disconnect MQTT Broker” while the broker is connected.

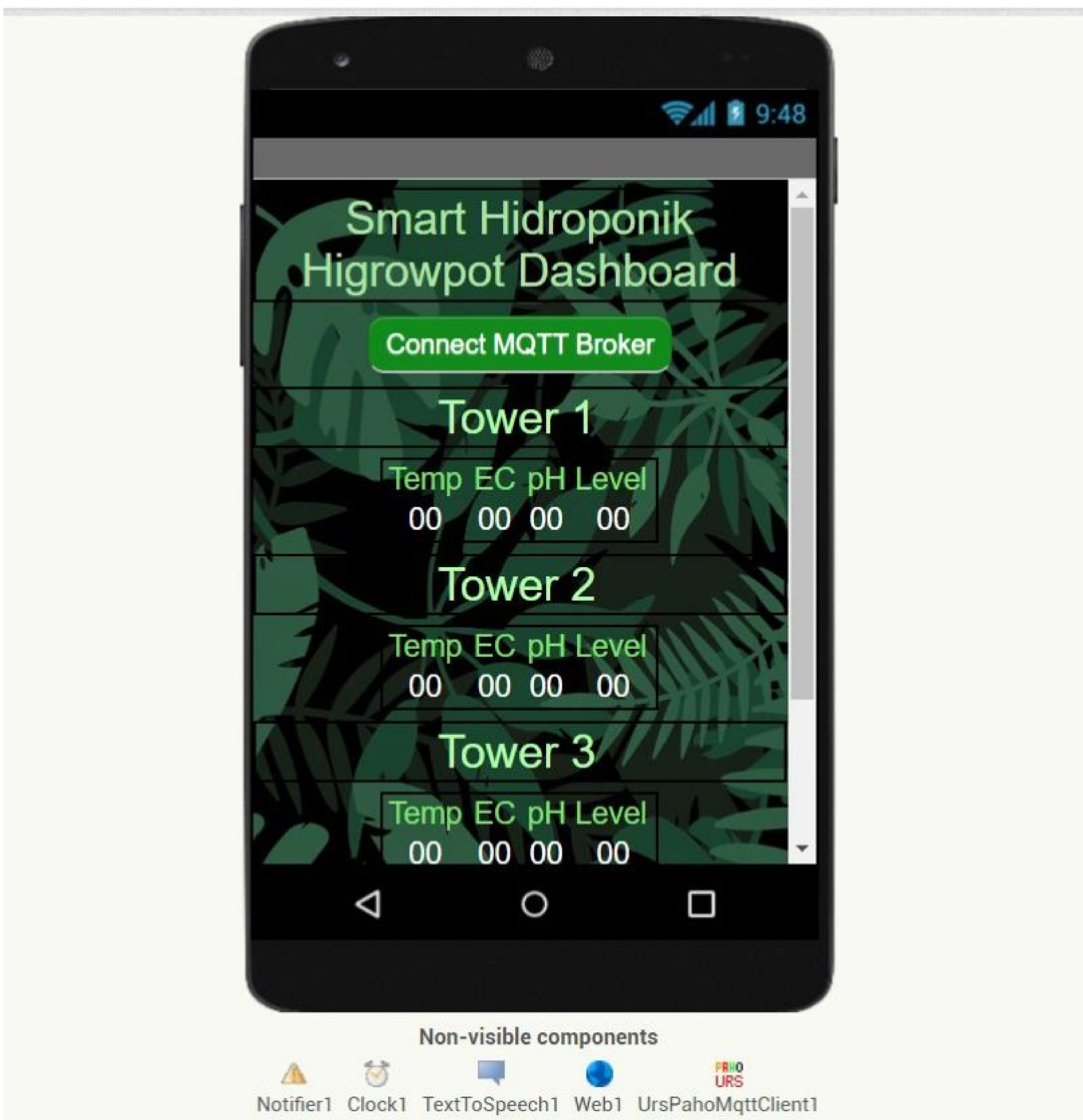

Figure 4.12 MIT App Inventor dashboard.

In the other side, the blocks mode showed a 3 clusters of code blocks. The first two being to connect to the MQTT broker and subscribe to the SensorsData/SensorsData/ topic via an extension called "UrsPahoMqttClient" and update the button label when it is connected or disconnected. The last and largest of the three clusters of blocks essentially split and parse the MQTT message received into a global list inside the application code. Then assigning each index of the list to its corresponding label on the application screen.

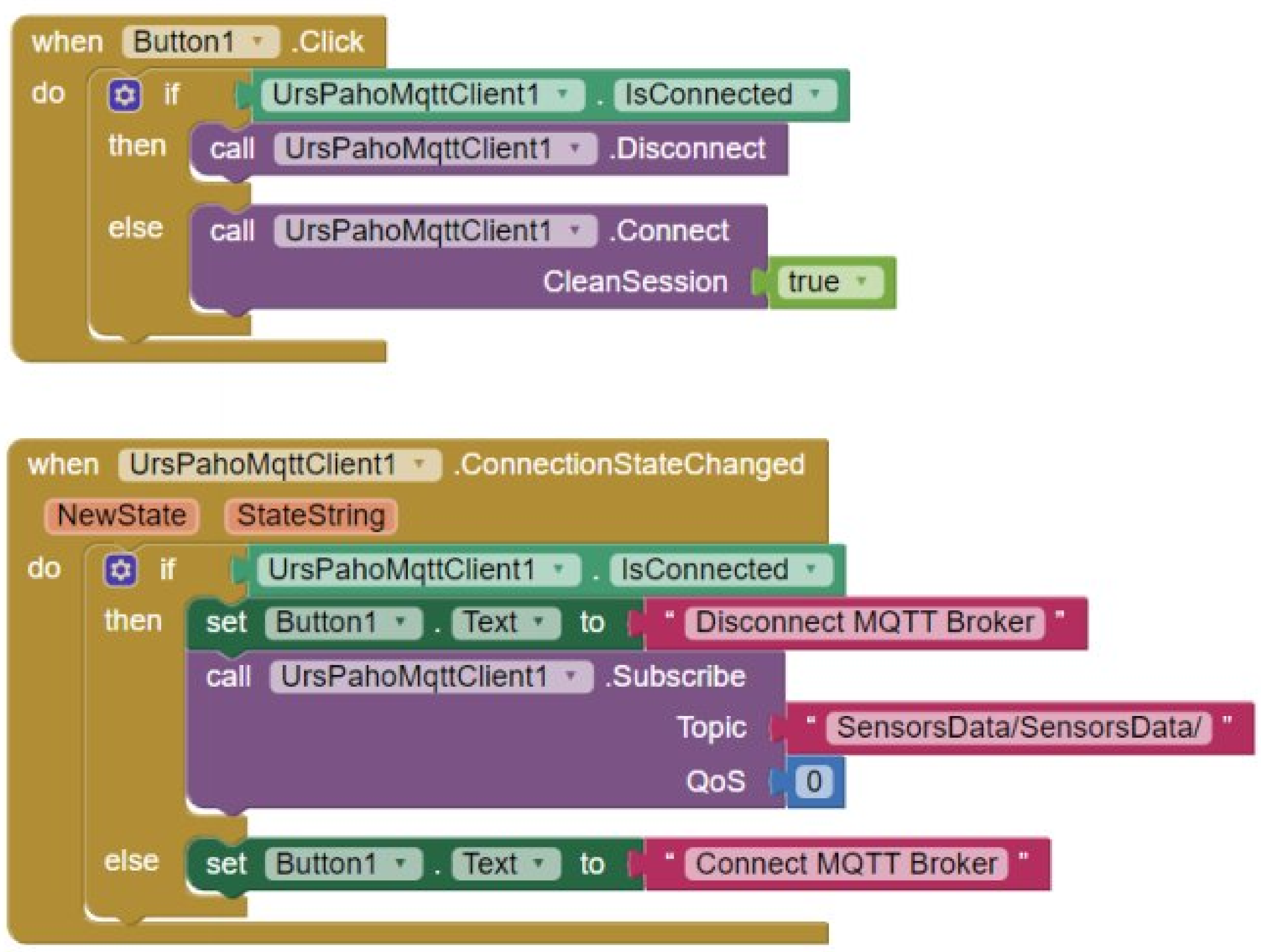

Figure 4.13 Button functionality and MQTT code blocks on MIT App Inventor.

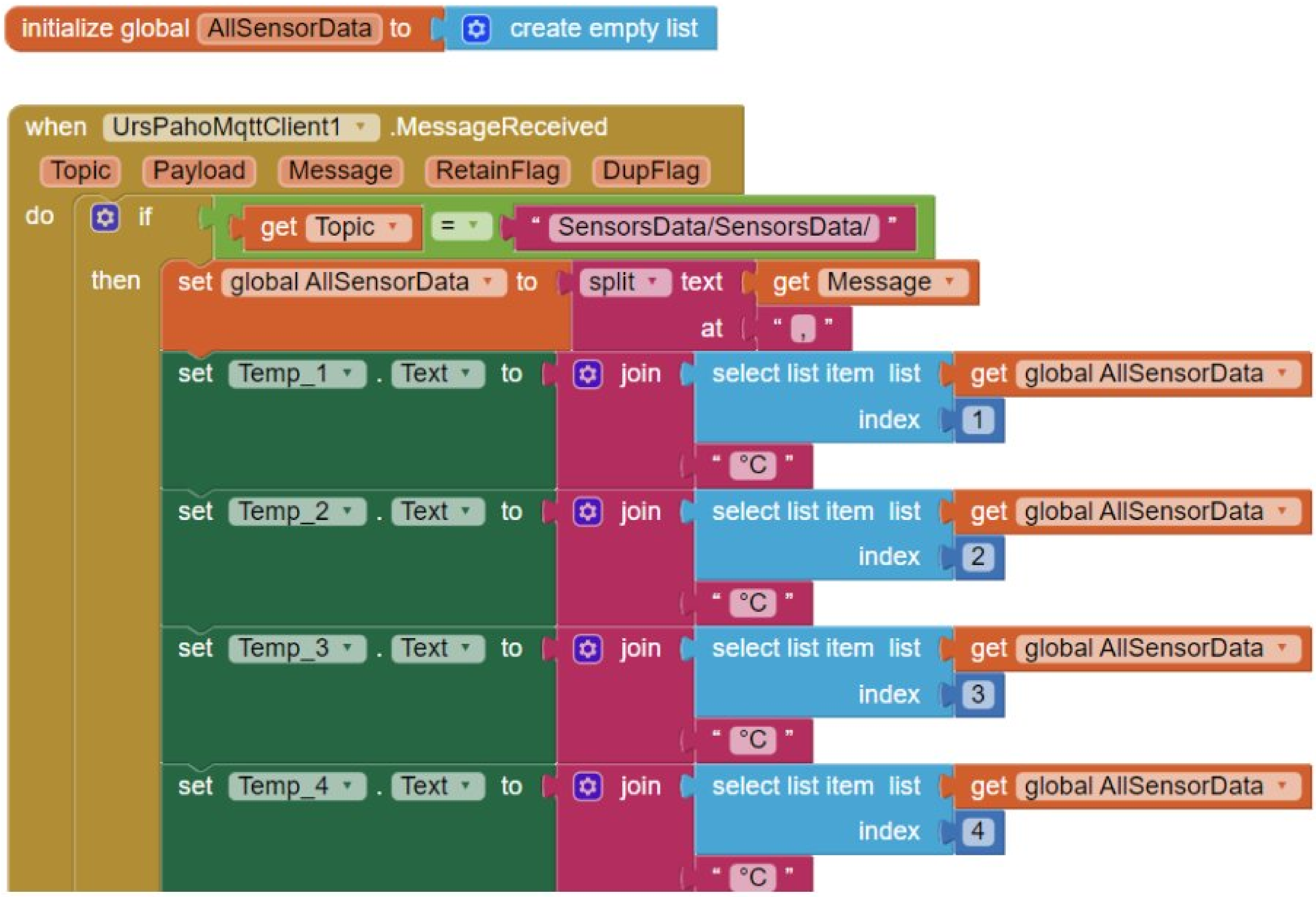

Figure 4.14 MQTT message retrieval and parsing into global list variable.

CHAPTER 5

CONCLUSION

5.1 Conclusion

The opportunity that this internship has provided has allowed me to heavily exercise my coding skills, researching skills and project management skills to be able to execute the project successfully. I have learnt plenty programming knowledge and techniques throughout this internship. Acquiring this knowledge and equipped with this experience has strengthened my electronics side of my Mechanical Engineering career; certain to prove to be a boost as I aspire to seek my Masters postgraduate education in Mechatronics Engineering.

The project main objectives were certainly achieved as testing all the sensors while connected to the board and observing the data on the phone application was successful.

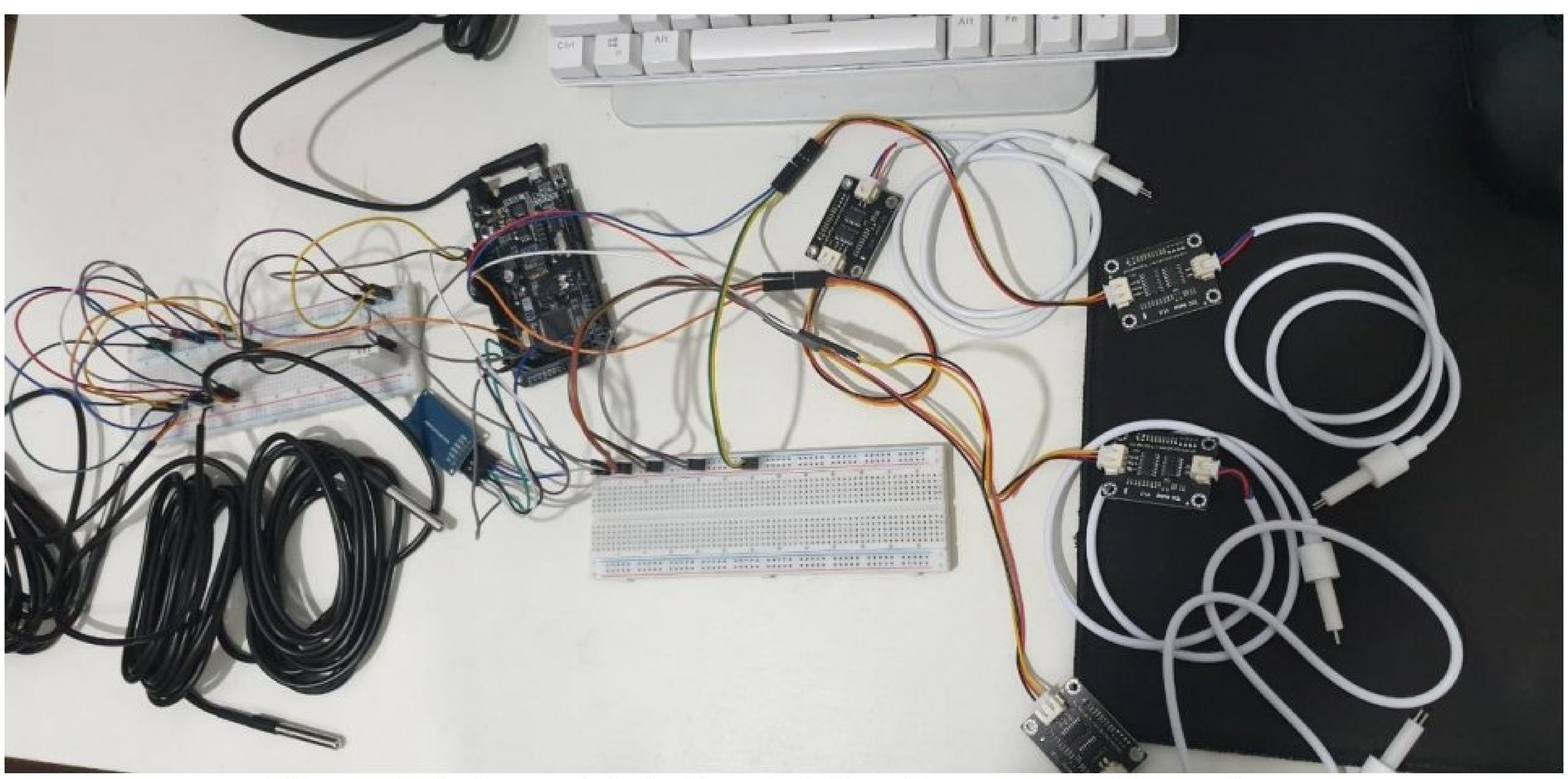

Figure 5.1 Assembled circuit following successful testing.

The monitoring system rig is expected to be installed at SK Taman Tun Aminah on Monday the 23rd of October 2023. This system will allow the users to monitor the hydroponics system from anywhere as long as they are connected to a Wi-Fi network.

5.2 Challenges & Final Thoughts

Moreover, there were uncountable challenges faced during the execution of this project, which are typically expected. For example, the pH sensor module offset potentiometer could only be set to 2.7V, which meant a systematic error of 0.2V was to be expected always. This problem was simply resolved by offsetting the error in the pH function of the Arduino code. Another issue arose when the ESP 8266 module showed persistent, uncommon and random errors every time a sketch code was uploaded to it; which ultimately resulted in the use of the ESP-Link firmware instead.

As is the case for any project, it must be admitted that the monitoring system did not turn out to be the best it could have been; but it has at least successfully fulfilled its purpose. For instance, it would have been more suitable to use 4 Wi-Fi enabled Arduino Unos for each of the towers; allowing each of them to publish their data in a sperate topic. The current set-up, although perfectly functional, is a bit limiting if any of the towers were desired to be operated in a different area than the other two.

It is without a doubt that the experience gained from working with all of the different sensors, software applications and microcontroller boards have widened my perspective on IoT systems and their highly beneficial affect when integrated into innovative projects such as the Smart Hidroponik Higrowpot system.

REFERENCES

Azimi, F. H., Aw, S. R., Syafie, L. S., Nur Adiimah Raja Aris, R. S., Fikri, K. I., & Wan Shafie, W. R. (2020, July 1). IOT monitoring in NFT hydroponic system using blynk- an android platform | International Journal of Synergy in Engineering and Technology. IOT Monitoring in NFT Hydroponic System Using Blynk- an Android Platform | International Journal of Synergy in Engineering and Technology. https://tatiuc.edu.my/ijset/index.php/ijset/article/view/18

Helmy, H., Nursyahid, A., Setyawan, T. A., & Hasan, A. (2016, October 19). Nutrient Film Technique (NFT) Hydroponic Monitoring System. JAICT, 1(1). https://doi.org/10.32497/jaict.v1i1.425

Abu Bakar Shedek, M. A., & Mohamed Ali, R. A. (2023, May 3). Smart Hydroponic System Using IoT | Evolution in Electrical and Electronic Engineering. Smart Hydroponic System Using IoT | Evolution in Electrical and Electronic Engineering. https://publisher.uthm.edu.my/periodicals/index.php/eeee/article/view/10727

DIY Guide - Mega2560 WiFi. (n.d.). DIY Guide - Mega2560 WiFi. https://sites.google.com/site/growboxguy/esp-link/esp8266

DIY Guide - MQTT. (n.d.). DIY Guide - MQTT. https://sites.google.com/site/growboxguy/esp-link/mqtt

Guide for DS18B20 Temperature Sensor with Arduino | Random Nerd Tutorials. (2019, July 2). Random Nerd Tutorials. https://randomnerdtutorials.com/guide-for-ds18b20-temperature-sensor-with-arduino/

Geran Komuniti Iskandar Puteri Rendah Karbon. (n.d.). Geran Komuniti Iskandar Puteri Rendah Karbon. https://sites.google.com/utm.my/iskandarputerirendahkarbon

Achievement In Receiving Of Iskandar Puteri Low-Carbon Community Grant 2.0 | Faculty of Science. (n.d.). Achievement in Receiving of Iskandar Puteri Low-Carbon Community Grant 2.0 | Faculty of Science. https://science.utm.my/blog/achievement-in-receiving-of-iskandar-puteri-low-carbon-community-grant-2-0/

MQTT Android Application | MIT app inventor. (2021, December 18). YouTube. https://www.youtube.com/watch?v=WAimZhU5phs

Unleash Your Inner Developer: Learn to Build MQTT Apps with MIT App Inventor | Part-1. (2023, May 5). YouTube. https://www.youtube.com/watch?v=0P1TMM1EgB0

Unleash Your Inner Developer: Easy Steps To Learn To Build MQTT Apps With MIT App Inventor 2 | Innovate Yourself. (2023, May 26). Innovate Yourself. https://innovationyourself.com/build-mqtt-apps-with-mit-app-inventor/

Smart Hidroponik Higrowpot. (n.d.). Smart Hidroponik Higrowpot | Johor Bahru. https://eo-eo.facebook.com/people/Smart-Hidroponik-Higrowpot/100094549659709/

APPENDIX

Arduino Code

```
// ---- INCLUDED LIBRARIES ---- //

//  1. DHT sensor:
#include <dht.h>

//  2. Temperature sensor:
#include <OneWire.h>
#include <DallasTemperature.h>

//  3. EC sensor:
#include <EEPROM.h>
#include "GravityTDS.h"

//  4. pH sensor: N/A

//  5. SD card:
#include <SD.h>
#include <SPI.h>

//  6. MQTT:
#include "ELClient.h"
#include "ELClientMqtt.h"

// -------------------------------- //

```

```
// ---- VARIABLE DEFINITIONS ---- //

//  1. DHT sensor:
dht DHT;
#define DHTPin 42
float Humidity = 0.00;
float Temperature = 0.00;

//  2. Temperature sensor:
const int TempsPin = 44;
float Temp[] = {0, 0, 0, 0};
OneWire oneWire(TempsPin);
DallasTemperature sensors(&oneWire);

//  3. EC sensor:
GravityTDS gravityTds;
const int ECPin[] = {A0,A1,A2,A3};
float temperature = 25;
float tdsValue = 0;
float ECValue[] = {0, 0, 0, 0};
int n = 0;

//  4. pH sensor:
float calibration[] = {0.00,0.00}; //change this value to calibrate
const int analogInPin[] = {A4,A5};
int sensorValue = 0;
unsigned long int avgValue;
float b;
int buf[10],temp;
float phValue[] = {0.00 ,0.00, 0.00, 0.00};

//  5. SD card:
const int chipSelect = 10;
int pos = 0;

//  6. MQTT:
const char *MqttTopic = "SensorsData/SensorsData/";  //Topic of every M
const char *MqttLwtTopic = "SHH/LWT/";    //When the connection is lost
const char *MqttLwtMessage = "Arduino Offline, Connection Lost!"; //thi

char MqttPath[64];
unsigned long LastPublish;
int PublishedCounter = 0;

ELClient ESPLink(&Serial3);
ELClientMqtt Mqtt(&ESPLink);

// -------------------------------- //
```

```
// ---- SETUP FUNCTIONS ---- //

//  1. DHT sensor: N/A

//  2. Temperature sensor:
void Temp_setup() {
    sensors.begin();
    pinMode(TempsPin,INPUT);
    Serial.println("Temp setup complete.");  //Confirm message
}

//  3. EC sensor:
void EC_setup() {
    gravityTds.setAref(5.0);  //reference voltage on ADC, default 5.0V on Arduino UNO
    gravityTds.setAdcRange(1024);  //1024 for 10bit ADC;4096 for 12bit ADC
    gravityTds.begin();  //initialization
    Serial.println("EC setup complete.");    //Confirm message
}

//  4. pH sensor: N/A

//  5. SD card:
void SD_setup() {
  File SD_Check;
  //  Initializing SD card:
  Serial.println("Initializing SD card...");

  if (!SD.begin()) {
    Serial.println("Initialization Failed!");
    return;
  }
  Serial.println("Initialization Complete!");
```

```
  //  Delete previous check file:
  if (SD_Check.available()) {
    SD.remove("SD_Check.txt");
  }

  //  Checking SD file create & write functions:
  SD_Check = SD.open("SD_Check.txt", FILE_WRITE);

  if (SD_Check) {
    Serial.println("Check: Writing to SD_Check.txt ... ");
    SD_Check.println("Status: Available");
    SD_Check.close();
  }
  else {
    Serial.println("Failed: Cannot access file!");
    return;
  }

  //  Checking SD file read:
  SD_Check = SD.open("SD_Check.txt");
  if (SD_Check) {
    while (SD_Check.available()) {
      Serial.write(SD_Check.read());
    }
    SD_Check.close();
  }
  else {
    Serial.println("error opening SD_Check.txt");
    return;
  }
}
```

```
//  6. MQTT:
void MQTT_setup() {
    Serial.print(F("WebServer starting.."));
    while (!ESPLink.Sync()) {
      Serial.print(".");
      Serial3.print(".");
      delay(500);
    }
    Mqtt.connectedCb.attach(mqttConnected);
    Mqtt.disconnectedCb.attach(mqttDisconnected);
    Mqtt.publishedCb.attach(mqttPublished);
    // Mqtt.dataCb.attach(mqttReceived);
    memset(&MqttPath[0], 0, sizeof(MqttPath)); //reset variable to store the Publish to path
    strcat(MqttPath, MqttLwtTopic);
    Mqtt.lwt(MqttPath, MqttLwtMessage, 0, 1); //(topic,message,QoS[only 0 supported],retain)
    Mqtt.setup();
    Serial.println(F("MQTT initializing.."));
    delay(5000); //allow 5s for MQTT to connect
}

// --------------------------------- //
```

```
// ---- PROCESS FUNCTIONS ---- //

//  1. DHT sensor:
String DHT_Process() {
    //DHT sensor Type:
    Serial.print("DHT22, \t");

    //  DHT sensor status:
    int chk = DHT.read22(DHTPin);
    bool Failed = false;
    switch (chk) {
    case DHTLIB_OK:
        Serial.print("OK,\t");
        Failed = false;
        break;
    case DHTLIB_ERROR_CHECKSUM:
        Serial.print("Checksum error,\t");
        Failed = true;
        break;
    case DHTLIB_ERROR_TIMEOUT:
        Serial.print("Time out error,\t");
        Failed = true;
        break;
    default:
        Serial.print("Unknown error,\t");
        Failed = true;
        break;
    }

    if (Failed == false) {
        //Obtain values:
        Humidity = DHT.humidity;
        Temperature = DHT.temperature;

        //Print to Serial:
        Serial.print(Humidity, 1);
        Serial.print(",\t");
        Serial.println(Temperature, 1);

        //Push to Blynk:


        //Return output to save:
        return "Temperature = " + String(Temperature) + " °C | Humidity = " + String(Humidity) + "%\n";
    }
    else return "N/A";
}
```

```
//  2. Temperature sensor:
String Temp_Process() {
    //  Request temp data from all sensors in bus:
    sensors.requestTemperatures();
    // Print temps to serial:
    for (int i=0; i<4; i++) {
      Temp[i] = sensors.getTempCByIndex(i);
      //Serial.print("Temperature from tower ");
      //Serial.print(i+1);
      //Serial.print(" = ");
      //Serial.print(Temp[i]);
      //Serial.println(" C");
    }
    Serial.println();

    //  Return to save:
    return "Water Temperature:\nTank 1 = " + String(Temp[0]) + "C\nTank 2 = "
}
```

```
//  3. EC sensor:
String EC_Process() {
    for (n=0; n<4; n++) {
        gravityTds.setPin(ECPin[n]);
        gravityTds.setTemperature(Temp[n]);  // set the temperature a
        gravityTds.update();  //sample and calculate
        tdsValue = gravityTds.getTdsValue();  // then get the value
        ECValue[n] = tdsValue/2;
        //Serial.print("EC Value ");
        //Serial.print(n);
        //Serial.print("= ");
        //Serial.print(ECValue[n]);
        //Serial.println(" ppm");
    }

    //  Return to save:
    return "EC Value:\nTank 1 = " + String(ECValue[0]) + "ppm\nTank 2
}
```

```
//  4. pH sensor:
String pH_Process() {
    for (int k=0; k<2; k++) {
      for (int i=0; i<10; i++) {
      buf[i]=analogRead(analogInPin[k]);
      delay(30);
      }
      for (int i=0; i<9; i++)  {
        for (int j=i+1; j<10; j++) {
          if (buf[i] > buf[j]) {
          temp = buf[i];
          buf[i] = buf[j];
          buf[j] = temp;
          }
        }
      }
      avgValue=0;
      for (int i=2; i<8; i++)  {
        avgValue+=buf[i];
      }
      float pHVol = (float)avgValue * 5.0/1024/6;
      phValue[k] = -5.70 * pHVol + calibration[k];
    }
    return "pH Value 1: " + String(phValue[0]) + "\npH Value 2: " + String(phValue[1]);
}
```

```
//  5. SD card:
void SD_Saving(String DataFileName, String SensorOutput) {
  //  Open SD card:
  File DataFile;
  DataFile = SD.open(DataFileName, FILE_WRITE);
  pos = DataFile.position();

  //  Write sensor data:
  if (DataFile) {
    DataFile.println(SensorOutput);
  }

  //  Close SD card:
  DataFile.close();

  //  Read saved data to Serial:
  DataFile = SD.open(DataFileName);
  if (DataFile) {
    DataFile.seek(pos);
    while (DataFile.available()) {
      Serial.write(DataFile.read());
    }
    DataFile.close();
  }
  else {
    Serial.println("error opening DataFile");
    return;
  }
}
```

```
//  6. MQTT:
void mqttPublish() {
    char WebMessage[512];                          //buffer for MQTT API
    // const char SensorData[512] = {floatToChar(Temp[0]),",",floatToChar
    memset(&WebMessage[0], 0, sizeof(WebMessage)); //clear variable
    // strcat(WebMessage, SensorData);
    strcat(WebMessage, floatToChar(Temp[0]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(Temp[1]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(Temp[2]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(Temp[3]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(ECValue[0]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(ECValue[1]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(ECValue[2]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(ECValue[3]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(phValue[0]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(phValue[2]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(phValue[1]));
    strcat(WebMessage, ",");
    strcat(WebMessage, floatToChar(phValue[3]));
    strcat(WebMessage, ",");
    strcat(WebMessage, "Low");
    strcat(WebMessage, ",");
    strcat(WebMessage, "Low");
    strcat(WebMessage, ",");
    strcat(WebMessage, "Low");
    strcat(WebMessage, ",");
    strcat(WebMessage, "Low");
```

```
    strcat(WebMessage, "Low");
    strcat(WebMessage, ",");

    memset(&MqttPath[0], 0, sizeof(MqttPath)); //reset variable to store the Publish to path
    strcat(MqttPath, MqttTopic);
    Serial.print(PublishedCounter);
    Serial.print(". publish to: ");
    Serial.print(MqttPath);
    Serial.print(" - ");
    Serial.println(WebMessage);
    Mqtt.publish(MqttPath, WebMessage, 0, 1); //(topic,message,QoS[only 0 supported],retain)
}

void mqttConnected(void *response) {
    memset(&MqttPath[0], 0, sizeof(MqttPath)); //reset variable
    strcat(MqttPath, MqttTopic);
    strcat(MqttPath, "#"); //subscribe to all sub-topics
    Mqtt.subscribe(MqttPath);
    Serial.println(F("MQTT connected!"));
    Serial.print(F("Listening to: "));
    Serial.println(MqttPath);
}

void mqttDisconnected(void *response) {
    Serial.println(F("MQTT disconnected"));
}

void mqttPublished(void *response) {
    Serial.println(F("MQTT published"));
}

//  7. Number to constant char:
char *floatToChar(float Number) {
    static char ReturnFloatChar[8] = ""; //4 digits + decimal sign + 2 decimals + null terminator
    dtostrf(Number, 4, 2, ReturnFloatChar);
    return ReturnFloatChar;
}

char *intToChar(int Number) {
    static char ReturnChar[8] = ""; //7 digits + null terminator max
    itoa(Number, ReturnChar, 10);
    return ReturnChar;
}

// --------------------------------- //
```

```
// ---- MAIN FUNCTIONS ---- //

void setup() {
    Serial.begin(115200);
    Serial3.begin(115200);
    Temp_setup();
    EC_setup();
    SD_setup();
    MQTT_setup();
    Serial.println("\nSENSORS SETUP COMPLETE!\n");
}

void loop() {
    //  Get output from sensor functions:
    String DHT_Output = DHT_Process();
    String Temp_Output = Temp_Process();
    String EC_Output = EC_Process();
    String pH_Output = pH_Process();

    //  Saving output to SD card:
    SD_Saving("DHT_Data.txt", DHT_Output);
    SD_Saving("Tmp_Data.txt", Temp_Output);
    SD_Saving("EC_Data.txt", EC_Output);
    SD_Saving("pH_Data.txt", pH_Output);

    //  Publishing data to MQTT broker:
    ESPLink.Process();
    if (PublishedCounter == 0 || (millis() - LastPublish) > 5000) {
      PublishedCounter++;
      mqttPublish();
      LastPublish = millis();
    }

    //  Delay 2s to create a detectable instance:
    delay(2000);
}
```