Project Thesis
Intelligent Hydroponic System

Arpit Sharma
P Number: P2613237,
Intelligent Systems and Robotics MSc.

Under the supervision of
Dr. Aboozar Taherkhani, De Montfort University, England.

Acknowledgements

I would like to thank Dr. Aboozar Taherkhani for being my supervisor for the project. It was because of his incessant guidance and valuable feedback throughout the period that such a complex system could be created for the project. Despite the COVID-19 outbreak, he was always available via MS Teams meetings and email. His deep knowledge about the hydroponic systems and machine learning made it possible to make the hydroponic system, intelligent by implementing and testing the state of art, TinyML, and IOT on it. To conclude, I am very grateful to my family and friends for their love and support during this tough academic year.

Contents

List of Figures

List of Tables

Abstract

The future of artificial intelligence lies in implementing artificial intelligence on very tiny chips so that they can be placed in "ultra-low-power" [1] systems , so that they can also be made solar-powered and their size would not be an issue. Generally, these chips do not conduct artificial intelligence operations locally. They just send the sensor data directly to the cloud, where artificial intelligence is located to make the decisions for them according to the received sensor readings. This leads to time lag and the system becomes highly dependent on the internet connection, making the chips unsuitable for systems that need immediate action and high reliability. One of those systems is hydroponic systems[2], which need to control the speed of the pump immediately to maintain the pH value [3] in the main tank where plants are growing. But there are many challenges as these devices do not offer much computational power. Therefore, achieving high AI accuracy is very difficult for them. Also, the tiny devices need to communicate with the user for which they must be able to conduct IOT operations. For the IOT operations, a user interface is necessary for the user to monitor and control the system from any part of the world. To solve these, problems, the proposed system includes the physical apparatus for the hydroponic system, incorporating the tanks and the required fittings, an ESP32 chip-based microcontroller with sensors and actuators attached to it to conduct AI on edge and IOT tasks simultaneously, a dedicated android [4] app for the user to monitor and control the system remotely via IOT.

1 Introduction

After Brexit, there is a lot of "emphasis on the food security and self-sufficiency" [5] as the food which was previously imported from other European countries now must be grown largely by the local farmers . The COVID-19 pandemic has further adversely impacted food imports giving rise to the need to encourage local food production in the country. But as the climatic conditions are not very suitable for traditional soil farming in most parts of England, plants must be grown in artificially created indoor environments using techniques like "hydroponic irrigation" [6].

The other reason for the adoption of the hydroponic system is its high number of "advantages"[7] over the traditional soil based irrigation methods such as lower water consumption as the water is recirculated rather than 99.9% getting wasted in evapotranspiration in soil-based irrigation.Infact, it uses just "10% of the water"[8] used in soil based irrigation. The other advantages are higher crop growth rate (30-50 percent faster than soil-based technique), higher-quality crop, less space requirement, freedom from most of the pests, and almost no reliance on seasonal changes.

This has led to an increasing number of farmers setting up hydroponic farms in the UK. Setting up the hydroponic systems needs high investments because of the cost of the shed, corrosion-resistant tanks, pipes, battery backup, high-quality pumps, reliable sensors, and circuitry. Almost everything is standard except the sensors and the circuitry which must be selected according to the preferred mode of operation, which is a major research topic.

The main challenge faced by the hydroponic systems is the maintenance of the parameters like pH and Electrical Conductivity(EC)[9] as the plants growing in hydroponic systems are much more sensitive to these parameters as compared to plants growing in soil. Therefore, there is a need to continuously monitor and control the system which is not manually possible. Therefore, intelligent automatic control is needed which can monitor and control the systems in real-time.

The automatic control must be self-adaptive as the behaviour of the hydroponic system can vary from time to time because of the uncertain environmental factors like temperature, lighting, etc.. Therefore, an intelligent system is needed to control the system. Also, it is preferable to have the artificial intelligence model run on 'edge' rather than remotely on the cloud because the response time has to be very low to accurately maintain and monitor the parameters, and also to make the system reliable by eliminating the need for an uninterrupted internet connection and saving the time wasted in sending the request, and waiting for the decisions over the internet especially for the decisions which have to be made quickly like deciding the pump speed for a specific pH. But the problem is that the deep learning models are generally run on an attached computer like Raspberry Pie [10], which is expensive, bigger in size, and has a high-power requirement as compared to small microcontrollers.

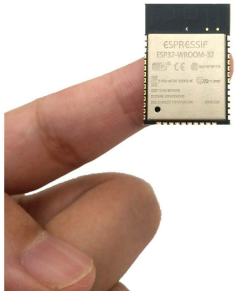

Figure 1: A 3.3 volt ESP32 chip used for controlling the physical apparatus of the hydroponic system and conducting IOT and 'AI on edge' operations simultaneously in the experiment.

Machine learning on a very small and low-powered device, tiny machine

learning, commonly referred to as tinyML recently has allowed handling machine learning devices on devices running on small microcontrollers like Arduino Nano 33 BLE Sense [11], ESP32 development board [12], NodeMCU development board [13], etc. A lighter version of an end-to-end open-source platform for machine learning such as TensorFlow Lite[14] has been made for training lighter neural network models, which can be quantified for it to be small enough to be transported to a tiny chip.

Occasionally, the status and performance of these automated systems also have to be manually monitored, preferably remotely by the farmers. The farmer must receive a notification of some sort in case the system behaves abnormally. The Internet of Things allows the system to report the system's status to the user remotely through the cloud. It also allows controlling the system over the internet in case the user detects any anomaly in the system. Therefore, a user interface such as a dedicated mobile app is needed for the system. Both the app and the microcontroller running the hydroponic system, must connect with the IOT cloud server and publish and subscribe to various feeds to send and receive messages respectively. The feed to which microcontroller publishes, the app must subscribe, and vice-versa for a successful message transmission.

One of the important parameters is pH. Every crop has its pH range requirement for growth like potatoes need a range of 5.8 to 6.2, whereas, for tomatoes, it is 6.0 to 6.5. The pH must be kept in their respective range for the safe growth of the plant. If the pH is lower (relatively acidic) or higher (relatively alkaline), the chances of the plant dying or the yield getting very adversely affected are very high. Bringing the pH in the required range and then maintaining it within that range is very crucial and challenging.

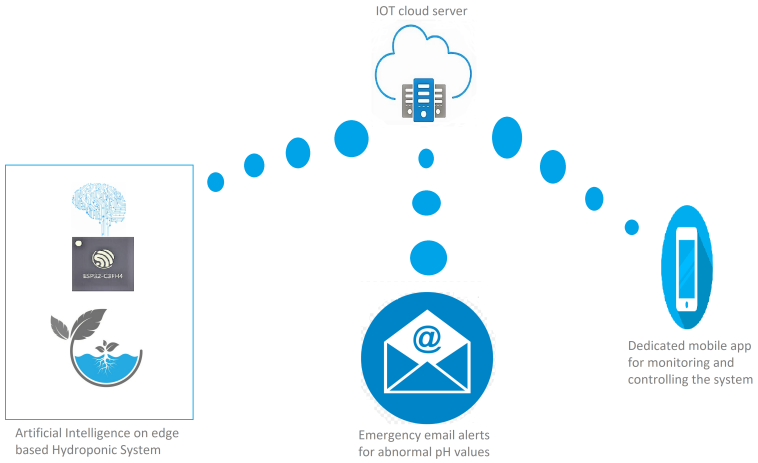

Figure 2: The proposed 'Intelligent Hydroponic system'

The proposed system (as shown in figure 2) comprises of a physical prototype of the hydroponic system, comprising of liquid tanks, pumps, potentiometer, sensors, and fitting pipes. They will have a fixed structure to give a reliable output. An IOT platform for the two-way communication between the dedicated app and the microcontroller using MQ Telemetry Transport (MQTT) messaging protocol[15]. An email alert message is sent to the user in case the pH value goes outside the set pH range (4.5 to 5.5 has been used in the experiment). A dedicated Jupyter Notebook for training the neural modal using TensorFlow Lite and Keras [16]. A ESP32 microcontroller (as shown in figure 1) programmed to handle IOT and machine learning on edge tasks simultaneously. A dedicated android mobile application built from scratch to control and monitor the system.

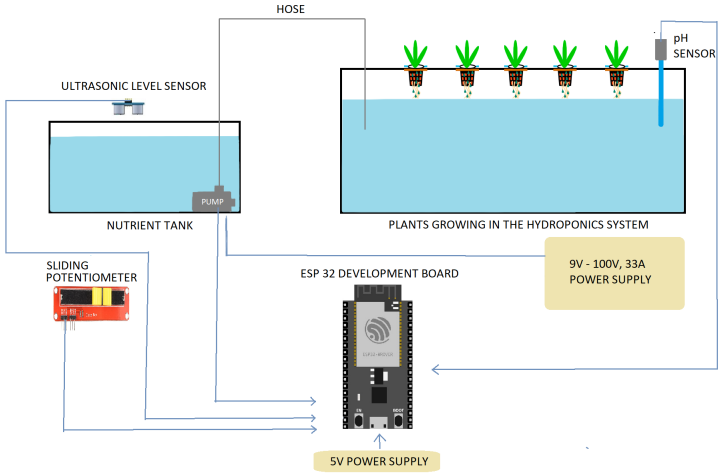

Figure 3: System architecture of the proposed, 'Intelligent hydroponic system'

The system architecture of the proposed system has been described in figure 3. All the sensors like analog pH sensor and the actuators like pump are directly connected to a single microcontroller, ESP32 development board.

1.1 Academic Objectives

The Academic objectives of the project can be summarised as follows:

1. Building the apparatus for the hydroponic system by trying and testing different actuators, sensors, and fittings.
2. Run machine learning models and Internet of Things operations simultaneously on a tiny low powered device.
3. Study the performance of the machine learning models on these devices in terms of accuracy and time taken per inference.
4. To gain experience in app development that can be used for intelligent systems.
5. To gain experience and skills working with TensorFlow Lite, TinyML, Arduino IDE[17], IOT using MQTT protocol and Android Studio[18].

6. To realize the development life cycle of a project related to Machine Learning.
7. Learning project management skills.
8. Learning the optimization of neural network models for them to reliably run on microcontrollers.
9. Studying artificial neural network architectures and their suitability to specific type of data and objective.
10. Learning the tools for handling and transforming data.

1.2 Research Questions

The following questions have been considered in this research:

1. Is it feasible and reliable to automate hydroponic systems using artificial intelligence and IOT?
2. Is it possible to reliably run IOT and Deep Learning on a very small and low powered microcontroller?
3. Which level sensor is best suited out of many available in market for hydroponic systems?
4. What are the limitations of the pH sensor in hydroponic systems?
5. What methods can be used to collect the data for training the model?
6. How to make the system learn to mimic a human adjusting the potentiometer to maintain the pH value in the tank in which the plants are growing.
7. What are the different methods in which pH can be controlled within a range using deep learning?
8. How to train the model, on the edge or have a pre-trained model on a computer and then transfer it to the microcontroller by making the required changes.
9. How do these methods compare against each other in terms of accuracy and efficiency?
10. How can the models be further extended into more applications like aquariums?
11. How can the success of IOT and neural network,measured individually?

1.3 Background Research Objectives

1. Investigation into the various microcontrollers capable of IOT and deep learning.
2. Investigation into the shortcomings of the systems running on mini-computers like Raspberry Pi.
3. Investigation into data pre-processing and building pipelines for feeding data to the model
4. Study into the optimization parameters of Neural Networks.
5. Study into the existing methods for data collection and optimising the model, with their advantages and shortcomings.
6. Study different end-to-end open-source platform for machine learning which can be used to train the model for the microcontroller.
7. Study into the various deployment and development platforms available which can be used to train and evaluate the models.

1.4 Product Objectives

1. Different types of actuators and sensors will be tried, and the best ones will be selected during the experiment.
2. The app made will be capable of communicating with the microcontroller and should have high accuracy and reliability. Initially, the app will be built for the Android platform, but is expected to be open for extending it to other platforms like IOS, Windows, Linux etc in the future.
3. Different messaging protocols will be considered for the two-way communication between the dedicated app and the microcontroller like MQ Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP)[19], Constrained Application Protocol (CoAP)[20], and Extensible Messaging Presence Protocol (XMPP)[21].
4. Several microcontrollers will be shortlisted which are capable of handling IOT and machine learning operations. Their configurations like processor type, RAM, physical size, and cost will be compared and then the best one will be selected. Finally, the selected microcontroller will be programmed using Arduino IDE to handle IOT and ML on edge tasks.
5. A dedicated Jupyter Notebook for training the neural modal using TensorFlow Lite and Keras will be made.
6. The final challenge will be to put the individual parts together and make them work in sync with each other to have a fully functional prototype of the system.

1.5 Resources Required

The following resources are required for the project:

1. Sensors including different types of liquid level detectors and pH sensors.
2. Actuators like different DC pumps, potentiometer, etc.
3. A range of microcontrollers for simultaneous IOT and machine learning operations like NodeMCU, ESP32, Arduino Nano BLE Sense, etc..
4. Apparatus including tanks and hose
5. Google Collaboratory
6. Local installation of Python with the following installed dependencies:
   (a) TensorFlow Lite
   (b) Keras Framework
   (c) NumPy
7. An account of adafruit.io to get access to the Adafruit IOT server
8. Local installation of Arduino IDE
9. Local installation of Android studio along with the software development kit (SDK)[22].

2 Literature Review

The research related to IOT-based, ML on edge, Intelligent hydroponic systems with continuous pH monitoring and controlling with a dedicated mobile app and IOT control panel is discussed here. So, we would be examining in depth several pieces of literature for the development of our system. They are discussed below.

2.1 pH monitoring in hydroponic systems

An study by Hardeep Singh illustrates the importance of pH monitoring in hydroponic systems along with the pH requirement of each crop capable of growing in hydroponic systems. It says that pH value determines "the availability of nutrients, so it should be maintained in the optimum range"[23]. The pH requirement for most of the plants is "between 5 to 6 (usually 5.5), so the pH in the root environment is maintained between 6 to 6.5". This pH makes the nutrients easily available to the plants. It also gives examples of the acidic fertilizers which can be used to remove the alkalinity (pH higher than the required pH) like those containing "phosphoric acid, citric acid or vinegar; or by reverse osmosis".

Another study by Hanif Fakhrurroja also states that by "keeping the pH at the optimum level, the crops will be easier to absorb the nutrient from the solution" [24]. Additionally, it illustrates the difficulties faced while controlling the pH value as the value is also dependent on the amount of "ion absorption by the crop roots". This makes the control much fuzzier and more complex.

Theeramet Kaewwiset conducted a study that focuses on the importance and indicator of the pH value in the hydroponic solution. According to it, pH value "is an indicator to define the relationship between the concentration of hydrogen ions H+ and OH- present in range 0 to 14"[25]. The pH measuring process is necessary for plants because every plant has different acidic or alkalinity needs to absorb nutrients in the chemical form.

Through this literature review, the true meaning of pH value along with its relevance in the hydroponic systems was found out. It was also realised that the pH value requirement for each plant is different. So, for the experiment, a bit lower pH range was chosen to get some time for the system to reach from pH value of 7(of pure water) to the required range to get a larger variation in the data. Therefore, a range of 4.5 to 5.5 was chosen, which intersects with the pH range for growing blueberries [26].

2.2 IOT in hydroponics

IOT refers to the Internet of things towards connecting people, and things using the internet, and also to store the data in the cloud for analysis. Farmers have recently started automating hydroponic culture since the emergence of this technology. Many parameters can be monitored and regulated using it remotely like pH, light intensity, flow, liquid level for both water in which the plants are growing and the liquid nutrients, etc.

An experimental study[27] by JV Gosavi has developed a prototype of a hydroponic system based on IOT which monitors the pH, water luminosity, and water conductivity by using sensors like Lumen's meter, pH, Electrical conductivity. It uses ARM 7 Microcontroller which continuously monitors the hydroponic system. The purpose is to ensure optimal plant growth. To make sure that the plants are in sixteen hours of light and eight hours of darkness, the microcontroller has a Real-Time Clock and a relay attached to it for controlling the lighting system that is attached to it. It also has a local LCD to display the status of the system.

Chanya Peuchpanngarm conducted an experiment[28], and developed an IOT-based autonomous control android/IOS mobile application for automating the hydroponic system. For this prototype, the sensor used is the temperature sensor, light intensity sensor, and humidity sensor, which are attached to the Arduino board. Between the cloud and the microcontroller, Raspberry Pi 3

is the interface. It controls the system according to the data captured from the sensors. The mobile application is used for remotely harvest recording and planning and monitoring the system.

S Charumathi developed a system [29] to monitor the hydroponic system's environment through various sensors in real-time focusing on the stability and accuracy of the results. The system automatically transmits the data captured from sensors like pH sensor, light intensity, humidity, the temperature to the cloud via IOT in real-time.

A study by Oleksii Barybin investigates the security of the systems made using devices based on ESP32 development board for IOT tasks. It focuses on the "vulnerabilities of wireless Wi-Fi networks to compromise data transmitted from devices based on ESP32"[30]. It is important as it impacts the security of the entire system. The study concludes that the use of User Datagram Protocol("UDP") is not safe for IOT purposes with this microcontroller.

An experimental study by Nico Surantha developed a "fuzzy"[31] logic-based intelligent system using the Internet of Things to control water and nutrients required by the plants automatically. IOT technology enables the sensors to interact with the IOT cloud server for the monitoring and processing of the captured data in real-time. The sensors are connected to the Arduino board to serve the needs of the plants using fuzzy logic. For instance, if the level of nutrients in the installation is low, the electrical conductivity sensors (EC) detect it, and the system automatically adds the nutrients. Farmers can use the processed data to improve their production.

For the project, the Adafruit IOT platform will act as a middleman between the ESP32 development board and the dedicated mobile application. The sensors used with the microcontroller will be liquid level sensors and a pH sensor. A DC pump will be used to pump lower pH liquid from the nutrient tank to the water in the hydroponic system to regulate the pH. Also, since the UDP was found not a safe protocol, MQTT was selected as an alternative.

2.3 Devices used for TinyML

Machine learning has been done for a very long time on computers. For building size, cost, and power constraint systems, Raspberry Pi [Raspberry Pi user guide], which is a low cost, of the size of a visiting card, to which keyboard and mouse can be attached just like a regular computer. The python language can be installed on the device and the machine learning tasks can be conducted like on any other computer. Plus, like microcontrollers, different types of sensors can be attached to them. But, when compared to microcontrollers with the same constraints, it has no advantage over it except higher computational power. But recently microcontrollers have become computationally powerful and that has

leads to the creation of libraries like TensorFlow Lite enabling these tiny devices to conduct machine learning tasks. This has revolutionized the field of Artificial Intelligence and opened a way for unlimited possibilities.

The book, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers [32] states the definition and importance of TinyML. It mainly focuses on the practical applications of the system. It is a quick start guide for people new to the world of microcontrollers but have experience in machine learning. The book is widely used by hobbyists to "build stuff they care about" and also by professionals. The book is a must-read and was one of the major factors in the coining and popularisation of the term, TinyML itself. This book was used as a guide for the project.

An experimental study by Colby R Banbury made four "hardware benchmarks"[1] for the TinyML microcontrollers namely, Caremark, Mlmark, Mlperf Inference, and Tinyml Requirements to study the landscape of the TinyML. It also discusses the problems faced during the making of these benchmarks, and at last, depicts TinyML in a positive light in the future by saying that it is "an important and rapidly evolving field that requires comparability amongst hardware innovations to enable continued progress and stability".

Another study by Eric Flamand talks about expanding the scope of edge ML to microcontroller class devices. In its words, '' enabling the next generation of edge devices to process data from richer sensors such as image, video, audio, or multi-axial motion/vibration has huge application potential''[33]. It proposes a new very low powered "edge computing engine" for the microcontrollers, GAP-8.

This section of the literature review helped in shortlisting the microcontrollers that could be used for the project by understanding the specifications of the microcontrollers given in their respective manual. It was found that NodeMCU development board, ESP32 development board and Arduino Nano 33 BLE Sense microcontroller were the perfect candidates for the project.

2.4 Machine Learning in hydroponics

The future of innovation is believed to be in the field of Artificial Intelligence. Intelligent systems are replacing manually run systems in almost every area of society with a much higher success rate. Machine learning is a subpart of Artificial intelligence which enables the computer to do the job autonomously after getting trained for a dedicated task. If the dataset used for training is made from human activities and is good, the computer has the capability of thinking and learning like a human being. ML algorithms have "various uses in the field of hydroponics towards controlling the plant growth" [34].

An experimental study by F. Ludwig uses a "Bayesian"[35] network for controlling and monitoring different parameters like levels of potenz-Hydrogen and light intensity, humidity, water level, electrical conductivity. One month was allocated for the collection of data to make the Bayesian network for predictive analysis for automatically controlling the hydroponic system. Additionally, the system has two websites for controlling, monitoring, displaying, and the actuators like DC pumps and solenoid valves that are part of the hydroponic systems.

KP Ferentinos conducted an experiment[36] that uses Artificial Neural Networks in Hydroponic systems to control levels of Electrical-Conductivity and potenz-Hydrogen. Feed-Forward Neural Model is employed with nine inputs outputting pH and Electrical Conductivity values. It is verified from the outputs that the neural network is accurate in predicting the values for predictive analysis.

An experimental study[37] by R Vadivel developed a machine learning-based model of the hydroponic systems and also employing IOT with intrusion detection capability. The data can be sent to the cloud in both ways, via wired connection or wirelessly. Instead of the MQTT protocol used in the project, it uses the Transmission Control Protocol (TCP) protocol. The pH sensor reads the values of the water in the Aquaculture and controls the water pump according to the readings of the soil moisture. The Machine Learning part happens on the cloud instead of locally(on edge), by receiving the inferences using IOT technology.

An experimental study[38] by Andreas Komninos developed an embedded network of sensors operating wirelessly. Uniquely, it was used in a big commercial setup. The whole network is inside a greenhouse and transmits the captured data to the Internet of Things platform. It has an ongoing machine learning project to predict the amount of light needed by the plant. But, it also uses machine learning on the cloud instead of on edge.

This part of the literature review helped in understanding the kinds of techniques used for training the neural networks. Although for these projects, either the machine learning happened on the cloud instead of locally or on a raspberry pi, which made them dependent on the internet connectivity at all times, or on a attached computer instead of a small low-powered microcontroller, respectively.

2.5 ESP32 microcontroller

Because of the various powerful features of the microcontroller as discussed in the previous sections, it has become a hot topic in recent times. Its compatibility with Arduino IDE and other IDEs has made it accessible for many users including professionals, students, and hobbyists. The following literature was found and reviewed on this microcontroller, which finalized its selection for the project.

An study[12] by Marek Babiuch discussed the capabilities and different variants of the ESP32 development board with their advantages and disadvantages. It has also focused on the type of applications, the microcontroller can be used for. It states the ability of the boards to run a lighter version of python, made for microcontrollers known as microPython.

Following the Literature review, a deep knowledge about the microcontroller was gained. The microcontroller is supported by the Arduino IDE and has plenty of libraries in Arduino IDE for both IOT and machine learning like Adafruit_MQTT.h and TensorFlowLite_ESP32.h, respectively. It is a 32-bit LX6 dual-core microprocessor with clock frequency up to 240 MHz, and Cryptographic Hardware Acceleration for AES, Hash (SHA-2), RSA, ECC, and RNG, it can handle complex code. As it supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps, implementing IOT was possible without connection breaks. It also has 34 Programmable GPIOs, up to 18 channels of 12-bit SAR ADC, and 2 channels of 8-bit DAC for the sensor and pump pins to be connected to the microcontroller. The serial connectivity includes 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART which allows the sensor values to be printed on the console monitor of the computer for printing sensor values dynamically for debugging, and for collecting the data for creating the dataset for the neural network. Because it has PWM (pulse width modulation) capability with up to 16-channels of PWM, the pump speed can be controlled by the microcontroller itself by signaling the sensing pin of the motor driver, driving the pump. it was found that for the collection of the dataset, there is no need to have a local display attached to it as it already has serial ports to communicate with the computer attached to it to display the sensor readings directly to the screen of the serial monitor of the Arduino IDE, via a micro-USB data cable. Therefore, it could be used to collect the data from the sensors to build the dataset to be used for training the neural networks.

2.6 Tools and Platforms

The following tools have been used during the project to build the system.

2.6.1 Android Studio IDE

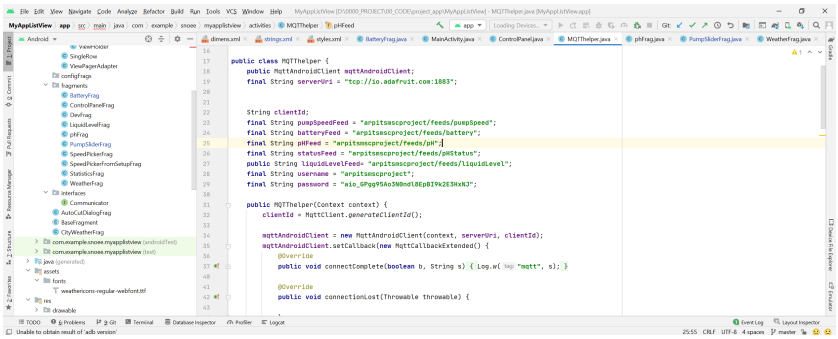

Figure 4: The code for the app being edited in the Android Studio IDE

Android Studio IDE as shown in figure 4, is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It offers coding in two languages, java, and Kotlin. Both have their pros and cons. Kotlin 1.0 was introduced much later on May 7, 2019, and makes the code more concise and readable, whereas java is an old and has inefficient null pointer exception handling, but has much bigger community support. Java was used in the project to build the app from scratch incorporating the MQTT protocol libraries written in java for IOT features in the app.

2.6.2 The Android Emulator

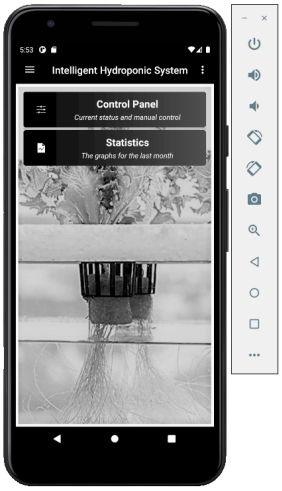

Figure 5: The app being tested on the Android Emulator

The android emulator as shown in figure 5, is used for testing and visualising the user interface of the the app without the need for a physical android device. The emulator was used extensively for building the app for the system.

2.6.3 Arduino IDE

Figure 6: The code for the system being edited in the Arduino IDE

The open-source Arduino Software (IDE) as shown in figure 6, makes it easier to write code and upload it to the board. This software can be used with any

Arduino board. With extended libraries, it can even be used with other compatible boards like ESP32, NodeMCU, Arduino clones, etc. It uses the C language for coding the microcontrollers, which was only two main void functions, setup, and loop. The first one runs only once when the microcontroller is powered and the second one runs continuously. Therefore, the initialization of the variables and the onboard pins (declaring them as inputs or outputs), goes in the setup function and the main code is part goes in the loop function. Custom functions and libraries can be written for making the code more readable. The IDE has been used for programming the ESP32 development board using the MQTT Adafruit library, TensorFlow Lite library, ESP32 library, etc. to making it able to manage the hydroponic system.

2.6.4 Adafruit IOT Platform

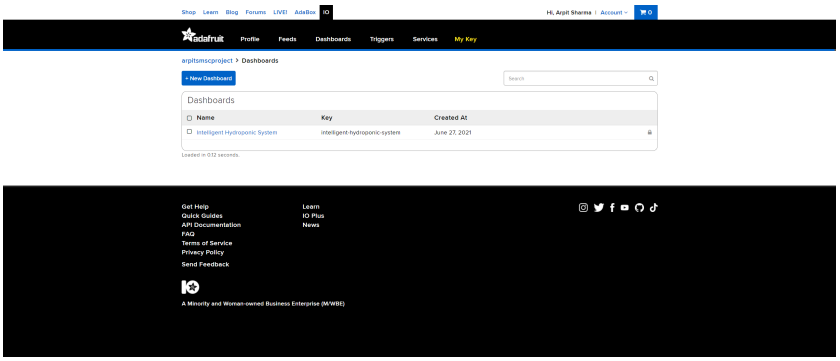

Figure 7: Creation of a dashboard on the Adafruit IOT Platform for the system

Adafruit IO platform as shown in figure 7, is a cloud provider focusing more on IOT deployments on the cloud. Adafruit IO supports different hardware like Raspberry PI, ESP2866, and Arduino. The user is required to make an account to be able to use it. Then a unique key is given to the user which must be used on all devices communicating with the platform. Several feeds/topics can be created to which the communicating device can either subscribe to receive a message or simply publish a message using the unique key. It has a customizable control panel in which several feeds can be added for visualization of the transmission of the IOT messages. The panel can be used for sending messages as well. But there is a limitation on the number of messages that can be sent in a minute. The free version offers 30 data points per minute but the paid version, Adafruit IO Plus grants sixty data points per minute. For the project, the free version was used the data was being sent just for monitoring purposes.

2.6.5 VS Code IDE

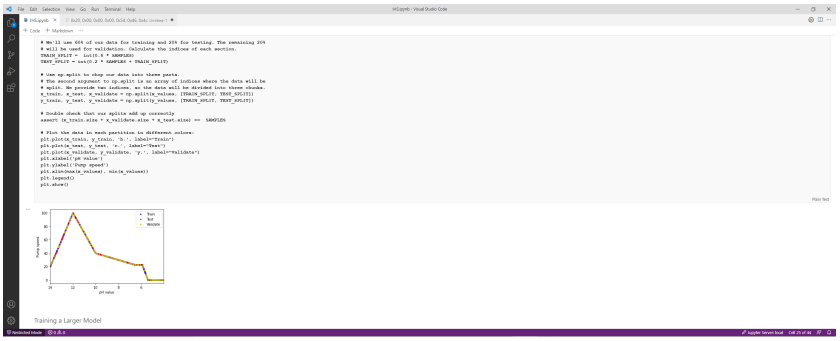

Figure 8: The Jupyer notebook being run on the VS Code IDE

VS Code IDE[39] as shown in figure 8, offers a stable and secure development platform for Python including development tools like plugins installer, debugging, and a simple structure for the project. This IDE was used to locally train the model.

2.6.6 Google Colab Platform

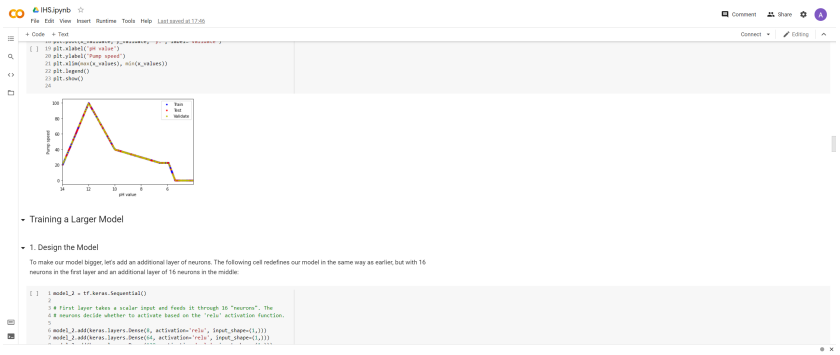

Figure 9: The Jupyer notebook being run on the Google Colab

Google Colab[40] or Collaboratory as shown in figure 9, is a free Jupyter Notebook-based development environment provided by google in which the code runs on the cloud instead of a local computer. The hosted resources include both TPU and GPU. For large datasets or computers with low-end configurations, Colab is very useful as everything is run on the cloud without utilizing local resources. It is also very handy if the user doesn't want to install all the required pre-installations of the python and the supporting libraries on his/her computer.

This environment is used for training the model with the dataset collected from the microcontroller, optimizing the model, visualizing the performance of the model by calculating the mean square error and mean absolute error, and finally converting the model to C language for it to be able to run on the microcontroller.

2.6.7 Python Programming Environment

Python programming language is widely used for machine learning thanks to several libraries and frameworks available for ML and data processing and analytic realms like Pandas, Seaborn, NumPy, Keras, etc. The language is much more readable and concise than the old languages like java or C because it's an interpreted language. The automatic garbage collection capability is very useful and is supported by flexible data structuring. It is platform-independent which allows to extend it across different platforms. Since it is a very popular language, it has great documentation and community support on forums like quora.com and stackoverflow.com. There is a very high chance of an error that one is looking for a solution for, being raised and solved, by someone else already on these forums. The language was used to program the neural network in a jupyter notebook.

2.6.8 Numpy library

NumPy is a mathematical library offered by the python language. It contains data structures and functions used for machine learning. It makes numerical processing faster and well represented because of the data structures and multi-dimensional array. The functions can be used for a large variety of operations like algebra, statistics, trigonometry, transforming matrix, generating random numbers, etc. These functions are in turn used in the construction of other frameworks like Keras, Pandas, and OpenCV as well. It is also used for scientific computing and simulation because of its versatility outperforming legacy platforms like MATLAB.It was used extensively in programming the Jupyter notebook

2.6.9 TensorFlow Lite

TensorFlow Lite is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and IOT devices. As the name suggests, it is a lighter version of TensorFlow, made especially for enabling the running of deep learning on small devices. It has multiple platform support for IOS, Windows, Android, Linux, and microcontrollers. It is also available in a lot of languages like Java, Python, Objective-C, C++, and Swift. with model optimization and hardware acceleration, it achieves high performance. It can be used for common ML tasks like text classification, pose

estimation, etc. on multiple platforms. For the project, it was used both on the computer in python programming language, and the microcontroller in C programming language to train and run the model respectively.

3 Methodology

3.1 Apparatus Selection

The following components was bought from online electronic stores, tested and connected with each other to build the physical prototype of the proposed system.

3.1.1 pH sensor

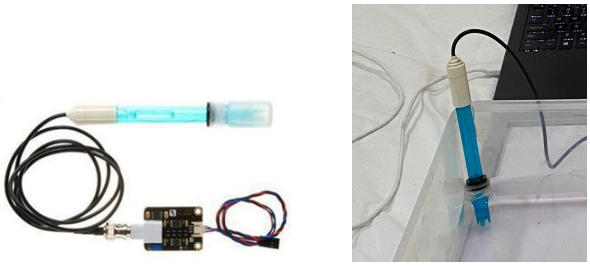

Figure 10: Analog pH sensor meter kit, cable of the shielding probe board, meter monitoring Analog pH sensor kit for Arduino [41]

An analog pH sensor/meter kit V2 as shown in figure 10, was used to measure the pH of the solution in which plants are supposed to grow. The sensor was taken good "care" [42] because being based on electrodes, it is a very delicate and sensitive instrument. It is also very important to change the sensor after the passage of the "expiry date" of the sensor.

It was found out that the sensor didn't behave properly when there were terminals, especially ground, of other sensors, or actuators in direct contact with the solution. The problem was severe and seemed to halt further experimentation. The problem was later found out to be known as a ground loop problem.

This was a major problem as the pH readings became completely unreliable when the pump started as the pump had some leakage current because of the terminals of the pump, not being properly isolated from the solution because of its improper construction. The problem because of the pump could be solved by

1. Finding a pump with no leakage current (a pump with perfectly insulated terminals).

2. Making a completely independent power supply for the pump using PWM[43] via optocouplers so that the pH sensor and the pump don't even have a common ground.

But the problem with the first option was that the pump could still malfunction because of the wear and tear of the insulation leading to complete system failure, which could lead to a very severe damage to the health of the plants being grown in the system. The second solution was more suitable as it would also make the selection of pump, flexible.

Another problem was caused because of the contact level sensor as it also shared common ground with the pH sensor and some terminals in direct contact with the solution leading to ground loop[44] formations. it also had to be replaced with its contactless alternatives, ultrasonic distance sensor.

Finally, the problem was solved by making sure that no sensor was in contact with the liquid except the pH sensors. So, the ultrasonic level sensor was the only option for the level sensor as other sensors were supposed to be used with contact with the solution leading to the ground loop problem. Hence, the HC-SR04 ultrasonic distance measuring sensor was finalized as the level sensor for the project. The pump was controlled by optocouplers using PWM, giving the need for the 1 Channel IRF540 MOSFET motor driver. It also had another advantage that with this circuitry, a wide range of voltage supply from 9 v to 100 V could be used for the pump according to the pump specifications. It also would save the microcontroller from negative voltage peaks from the pump that could destroy the circuit.

3.1.2 Water Level sensor

1. Hall sensor-based flowmeter

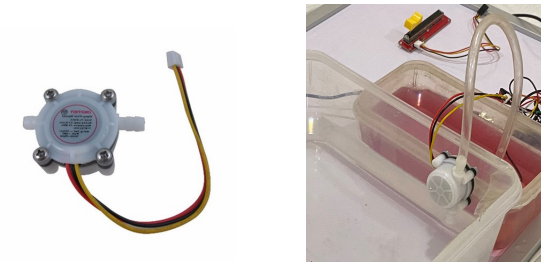

Figure 11: 3. 0.3-6L/min G1/4" Water Coffee Flow Hall Sensor Switch Meter Flowmeter Counter Connect Hose [45]

The flowmeter shown in figure 11 was tried which was attached to the output of the pump. The principle of the sensor is very simple. The flowing water makes its turbine rotate and the voltage produced by the hall sensor[46] whenever the turbine completely a full rotation is read by an analog pin of ESP32 microcontroller to calculate the flowrate. But the problem faced was that the sensor wouldn't function at lower flow rates because of the inertia of its turbine. Moreover, the inertia of the turbine would further lower the flow rate of the pump making it very difficult to properly calibrate the sensor.

2. DollaTek water level sensor

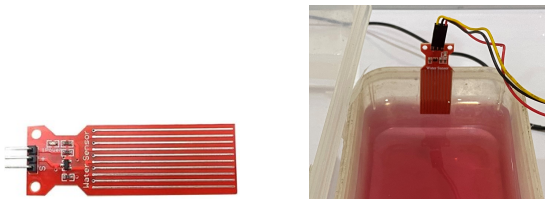

Figure 12: Depth Water Level Sensor [47]

The other sensor tried was the DollaTek water level sensor as shown in figure 12. Initially, it seemed to serve all the requirements as the readings were very precise and accurate, and the length of the sensor matched the height of the nutrient tank.

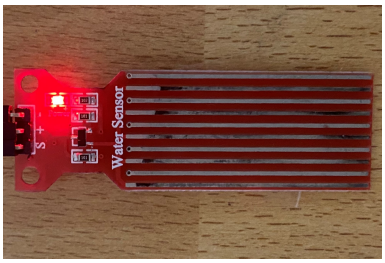

Figure 13: Sensor damaged after few hours of operation

But after few hours of experimentation, the electrode of the sensor started to corrode (as can be seen in the picture taken of the sensor during the experimentation shown in figure 13), and the sensor began to give garbage readings. One workaround was made to power on the sensor only while readings were being taken. This would minimize the reaction between the ions of the solution and the electrodes of the sensor. Hence, the power

pin of the sensor was connected to the digital pin of the microcontroller, which was made high only when the sensor was supposed to be active. This measure improved the results and prolonged the life of the sensor. But, still, the corrosion was not completely prevented, which still meant the replacement of the sensor after two to three days of experimentation.

The other option was to replace it wih a sensor with platinum electrodes as they are much less prone to corrosion than copper ones, but was very expensive, hence, not feasible for the experiment.

3. Contact water/liquid level sensor

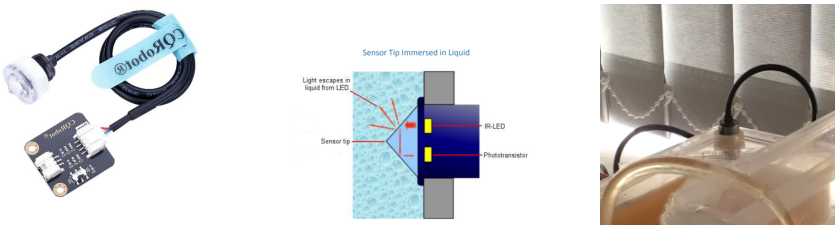

Figure 14: Contact sensor and its mode of operation [47]

4. Water Level Detection Sensor Depth Water Level Sensor

   CQRobot Ocean's contact water/liquid level sensor as shown in figure 14 was initially tried as it gave a very precise and accurate reading. It had to be always kept in contact with the solution. This was very challenging at first, but the problem was solved by mounting the sensor on the bottom of a plastic container that was made to float in the nutrient tank.

5. HC-SR04 ultrasonic level sensor

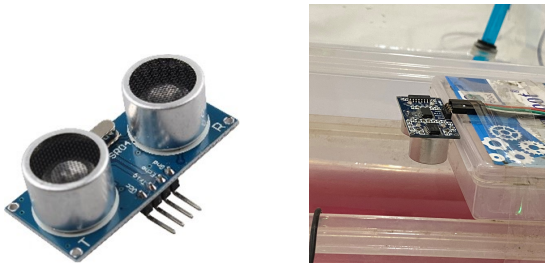

Figure 15: HC-SR04 Ultrasonic Sensor Distance Measuring Module [48]

HC-SR04 ultrasonic sensor as shown in figure 15 had to be tried due to the ground loop problem discussed in the previous section because of previously tried sensors.

When the nutrient solution is up to the brim of the tank, the value read by the microcontroller is 1000 mm (the height of the nutrient tank) which decreases gradually as the pump pores solution from the nutrient tank to the tank where plants are growing. The precision and the accuracy of the sensor are great and are also used in many similar experiments found during the literature review. Hence, it was finalized for the project.

3.1.3 Pump, hose and motor driver module

(a) DC 3-6 V Mini Micro Submersible Water Pump

Figure 16: The pump used for the experiment [49]

As it was needed to control the speed of the pump to bring the pH of the solution in which plants were supposed to grow, in the desired range, it would be easiest to do so by using a DC pump (as shown in figure 16) as the speed of these pumps is proportional to the supplied voltage which can be controlled using pulse width modulation by the microcontrollers. An ideal DC pump would have absolutely zero leakage current meaning that its circuitry would be completely insulated from the solution, but the pumps are expensive and can cause a complete system failure, if they malfunction. The pump is also supposed to have sufficient torque to pull the liquid from the nutrient tank to the tank where plants are growing. It is also important to have flowrate as low as possible so that reliable readings can be taken at low pump speed as well. The movement of the pump's motor should be jerk-free to have a smooth flow. The pump shown on the right side in figure 16 was chosen for the experiment.

(b) Hose

Figure 17: The hose used for the experiment [49]

The hose is very important for the assembly of the system. It is required to be leak-proof because if it leaks, it can mess up the entire data collection process and the general operation of the system. It also can lead to wastage of the expensive nutrient solution, which can increase the running cost of the system. That would even make the hydroponic system no longer less expensive than traditional irrigation methods, which is otherwise supposed to be a major advantage of hydroponic system over traditional irrigation methods.

The thickness of the hose should also be according to the pump because if it is thicker than the pump can pump the solution through, it will lead to system failure. It is also important to maintain the shape of the pipe throughout the experiment as any change would affect the flowrate, making the trained neural network highly inefficient. An inexpensive clear hose of the specifications given in figure 17 was selected for the experiment.

1. DC motor driver

Figure 18: DC motor driver [50]

Two different DC motor drivers were tried for the experiment. The first was an H -bridge-based module. But, since it was required for the pump's motor to rotate in just one direction, and it shared ground with the microcontroller's circuit, it was soon replaced with the 1 Channel IRF540

MOSFET motor driver (as shown in figure 18) because of the reasons discussed in earlier sections.

3.1.4 Microcontroller selection

1. Arduino Nano 33 BLE Sense [ABX00031]

Figure 19: Arduino Nano 33 BLE Sense [51]

This microcontroller (as shown in figure 19) was the first microcontroller for which machine Learning was officially launched on Arduino IDE. It had huge documentation provided by arduino.cc itself. A lot of examples of machine learning on the Tensorflow Lite platform in Arduino IDE like 'Hello world' and 'Magic wand' were studied to understand the basics of machine learning on edge. The only problem was that the board has no WIFI capability, which didn't allow the system to be monitored remotely via IOT. Hence, another microcontroller had to be looked for.

2. NodeMCU

Figure 20: 1. SP-COW Development Board WiFi WLAN Wireless Module for ESP8266 for NodeMCU for ESP-12E compatible with Arduino [13]

The next microcontroller tried was an ESP8266-based microcontroller (as shown in figure 20). All the IOT operations were successfully executed. It also offers machine learning, but with Arduino IDE, it has very little community support. Plus, it has a very limited RAM and processor size to manage machine learning operations in needed time. Therefore, another microcontroller was needed for the experiment, which could handle IOT and machine learning operations simultaneously on the fly, comfortably

and without much lag.

3. ESP32 Development Board

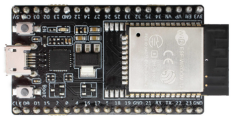

Figure 21: ESP32-DevKitC core Board ESP32 Development Board ESP32-WROOM-32U compatible with Arduino IDE [52]

The final microcontroller was the ESP32 development board (as shown in figure 21). It possesses a very powerful microprocessor, and a large RAM, and a WIFI capability. Hence, it was found to be the smallest-sized microcontroller for conducting machine learning operations reliably for the project. It conducted both IOT and machine learning operations using Arduino IDE without any error.

3.1.5 Sliding potentiometer

Figure 22: Sliding potentiometer to manually set the pH value for the AI to mimic [53]

A sliding potentiometer (as shown in figure 22), was used to collect data for building the dataset for machine learning. It is a logarithmic potentiometer, meaning that its resistance increases on a logarithmic scale unlike linearly for a linear potentiometer. These potentiometers are used for achieving high resistance with lesser sliding distance.

The potentiometer is used to control the speed of the pump using PWM. An expert is first supposed to manually adjust the pH to the required range using the sliding potentiometer to control the pump speed, which pumps the nutrient solution to the main tank in which pH is being measured. The sensor data is collected continuously while the user is doing the maneuver. The data is later processed to prepare training data for feeding it to the neural network. The neural network then learns to mimic the user's action. This gives the user to have his/her preference to control the parameters (pH in the project) of the hydroponic system, rather than a fixed controller like a PID controller doing

the job.

3.1.6 Circuit diagram

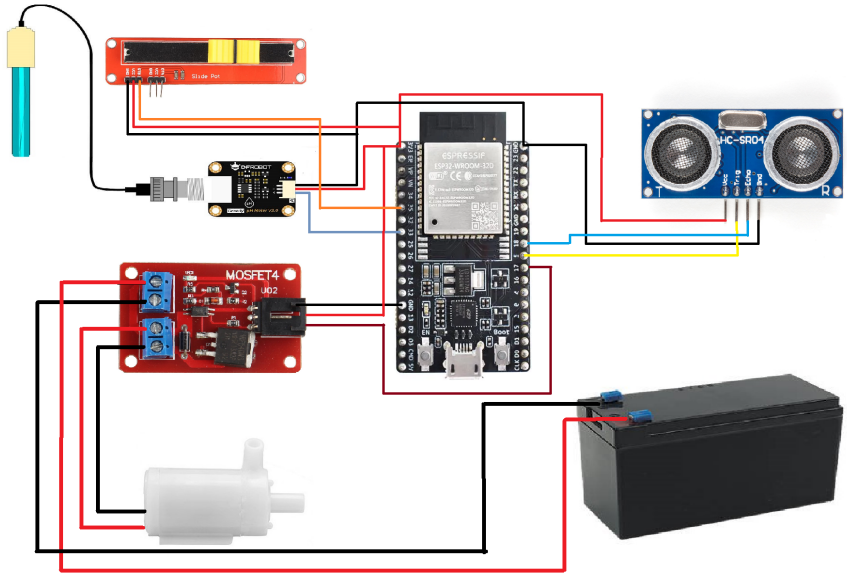

Figure 23: Final circuit diagram made for the project

Finally, the circuit was rigged (as shown in figure 23), with the motor driver pin connected to digital pin 17. The pH sensor pin is connected to analog pin 33. The trigger pin, and the echo pin of the HC-SR04 ultrasonic sensor are connected to to pin 5 and pin 18, respectively, of the microcontroller.

The battery size is to be selected according to the pump power requirements. The sensors must be connected to the analog pins of the microcontroller. The ESP 32 microcontroller, being a 32-bit device has higher precision, and hence, can read sensor values between O to 4095 instead of just between 0 and 1024 like an 8-bit microcontroller like Arduino UNO microcontroller.

The circuit must be connected as shown in figure 23 for the successful construction of the proposed hydroponic system.

3.2 Android App Development

The app was made using the Android Studio IDE in java programming language by devoting a major time of the experiment. The entire code of the app is very long. So, the working of the app has been very briefly explained in this section. The user interface was carefully designed to make it aesthetically beautiful at the same time. It uses MqttAndroidClient class to publish and subscribe to the feeds for monitoring and controlling the system via Internet of Things. A dialog box was used to display the developer's details.

Following are the fragments that make up the control panel screen of the app.

1. The battery fragment

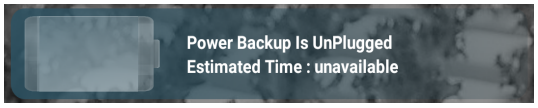

Figure 24: Fragment displaying the battery level for the system in the app's user interface

```
public void setBatteryTv(int batt) {
        Toast.makeText(getActivity(), "changed", Toast.LENGTH_LONG);
        battery = batt;
        if (batteryTv != null&&battery!=-1) {
            batteryTv.setText(String.valueOf(battery) + "%");
            layoutParams = battLevel.getLayoutParams();
            layoutParams.width = (int) (Double.valueOf(batteryLevelWidth / 100.0)
            * Double.valueOf(batt));
            battLevel.setLayoutParams(layoutParams);
            estimatedTimeRemaining = (int) (batt * battToTimeConversionFactor * 60);
            estimatedRunningTimeTv.setText(estimatedTimeRemainingPrefix
            + String.valueOf(estimatedTimeRemaining / 60)
            + " hours" + " , " + String.valueOf(estimatedTimeRemaining % 60)
            + " minutes");
        }
}
```

The app subscribes to the battery feed of the IOT server so that whenever the microcontroller publishes the latest battery status, the battery level is updated inside the battery fragment in the app, as shown in figure 24. The setBatteryTv method(as shown above) inside the BatteryFrag.java gets called whenever the data on the battery feed changes, and an integer value(the battery level in percentage between zero and hundred) is received as a parameter, which is used to update the Textview for displaying the battery level. The entire layout was designed in the XML format. This feature is for the future extension of the project to make the

system, solar powered. At this stage, the system is plugged with the 5V DC adapter plugged into the wall, and so the battery levels are always 100 percent.

2. The weather fragment

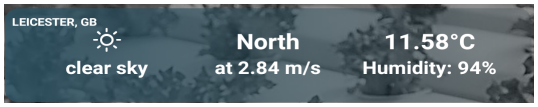

Figure 25: Fragment displaying the system's location's weather, outside wind direction, and speed, outside temperature, and humidity in the app's user interface

```
class DownloadWeather extends AsyncTask<String, Void, String> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            loader. setVisibility (View.VISIBLE);

        }

        protected String doInBackground(String... args) {
            String xml = excuteGet("http://api.openweathermap.org/data/2.5/weather?q="
                    + args[0] + "&units=metric&appid=" + OPEN_WEATHER_MAP_API);
            return xml;
        }

        @Override
        protected void onPostExecute(String xml) {

            try {
                JSONObject json = new JSONObject(xml);
                if (json != null) {
                    JSONObject details = json.getJSONArray("weather").getJSONObject(0);
                    JSONObject main = json.getJSONObject("main");
                    JSONObject wind = json.getJSONObject("wind");
                    DateFormat df = DateFormat.getDateTimeInstance();

                    Log.i("test", wind.getString("speed"));
                    Log.i("test", wind.getString("deg"));
                    direction = degToDir(wind.getDouble("deg"));
                    cityField.setText(json.getString("name").toUpperCase(Locale.US)
                    + ", " + json.getJSONObject("sys").getString("country"));
                    detailsField.setText(details.getString("description").toLowerCase(Locale.US));
                    currentTemperatureField.setText(String.format("%.2f", main.getDouble("temp"))
                    + " C");
                    humidity_field.setText("Humidity: "
                    + main.getString("humidity") + "%");
                    windDirection.setText(direction);
                    windSpeed.setText("at "
                    + wind.getString("speed") + " m/s");
                    weatherIcon.setText(Html.fromHtml(setWeatherIcon(details.getInt("id"),
                            json.getJSONObject("sys").getLong("sunrise") * 1000,
                            json.getJSONObject("sys").getLong("sunset") * 1000)));

                    loader. setVisibility (View.GONE);

                }
            } catch (JSONException e) {
                Toast.makeText(getActivity(), "Error, Check City", Toast.LENGTH_SHORT).show();
            }


        }
```

All the information about the weather is got retrieved the API of openweathermap.org. It takes the name of the city and country to output the weather in JSON format. The JSON file retrieved is then parsed to

get the individual parameters like current weather, wind direction, wind speed, outside temperature, and outside humidity. The parameters are used while updating the weather fragment in the control panel screen of the app (as shown in figure 25) with the latest weather details.

3. The Liquid-level fragment

Figure 26: Fragment displaying the nutrient tank's liquid level in the app's user interface

```
public void setLiquidLevelTv(double ll) {
    liquidLevel = ll;
    if (liquidLevelTv != null) {
        liquidLevelTv.setText(String.valueOf(ll));
    }
}
```

This fragment is made for displaying the liquid level left in the nutrient tank detected by the ultrasonic range sensor that is selected as the level sensor for the system. As the pump pumps the nutrients from the nutrient tank to the main tank in which the plants are supposed to grow, the liquid level decreases and the level sensor readings are continuously sent via IOT to the app for monitoring purposes. The values are in millimeters because of the precision of the sensor. The setLiquidLevelTv method (as shown in figure 26) inside the LiquidLevelFrag.java gets called and a double value(the liquid level in mm) is received as a parameter, which is used to update the Textview for displaying the liquid levels. It also helps the user to ensure that the nutrient tank is not empty.

4. The pH fragment

Figure 27: Fragment displaying the pH value and pH status in the app's user interface

```
public void setPhValueTv(double ph) {
    phValue = ph;
    if (phValueTv != null) {
```

```
            phValueTv.setText(String.valueOf(ph));
        }
    }

    public void setPhStatusTv(String status) {
        phStatus = status;
        if (phStatusTv != null) {
            phStatusTv.setText(String.valueOf(status));
        }
    }
```

The pH fragment (as shown in figure 27) shows the current pH value and pH status of the tank inside which the plants are growing, which is the most important part of the project. The fragment (like other fragments) gets updated when the pH value is published by the microcontroller to the cloud and is received by the microcontroller. The callback methods, setPhValueTv and setPhStatusTv as shown above, get called for updating pH value and pH status, respectively.

5. The pump fragment

Figure 28: Fragment displaying the pump speed in the app's user interface

```
public void pumpSliderInit() {

        pumpSlider.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                seekPin = progress;
                pumpSpeed.setText("Pump Speed : " + progress);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                communicator.onDialogMessage("PumpSliderFrag", "" + seekPin);
            }
        });
    }
```

When the speed of the pump is calculated by the AI on the microcontroller and is published to the IOT server, the slider in this fragment shown in figure 28, also slides automatically according to the current speed of the pump. The speed can also be controlled manually by sliding the slider. Whenever the slider slides, the setOnSeekBarChangeListener callback method (as shown above) of the SeekBar class inside PumpSliderFrag.java gets called, setting the text of the Textview, and the slider of the fragment to the current pump speed of the system.

6. Screenshots of the app

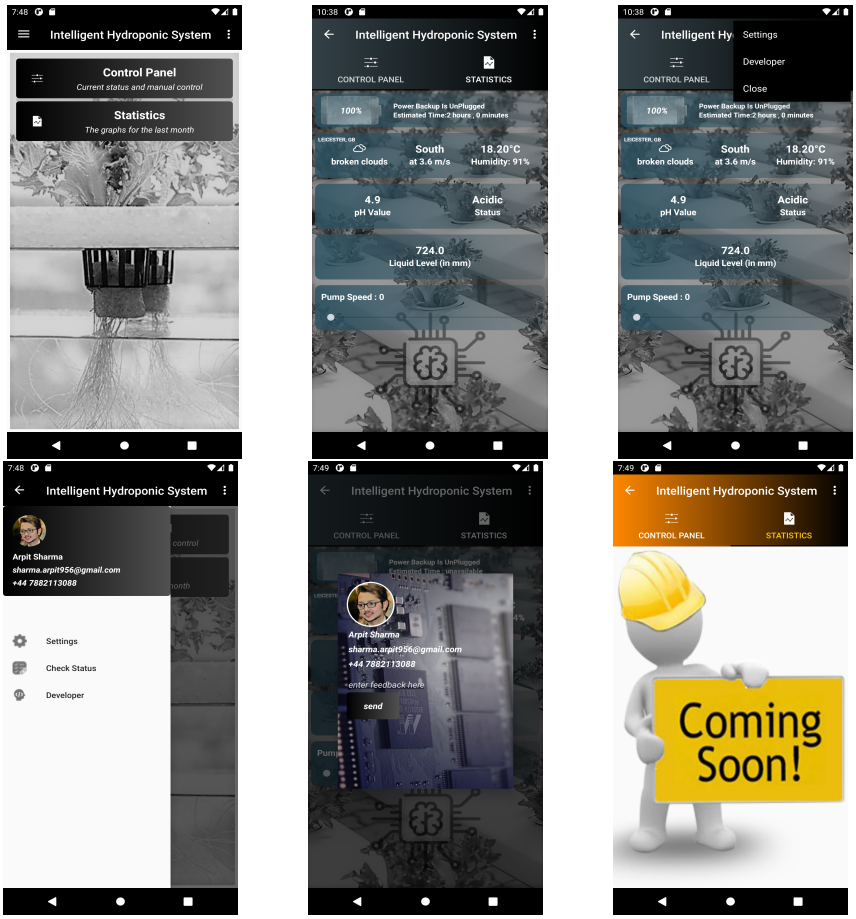

Figure 29: Screenshots of the app

The screenshots in figure 29 are from the app, displaying the main screen, control panel screen, statistics screen, developer screen, drop-down menu, two scrolling tabs, and a side navigation window. The statistics screen will be made for displaying the performance of the system over time ,and is for extending the project in the future.

There were many challenges and problems faced during the making of the app. Because of the massive community support on the internet, the complete app could be made successfully.

3.3 IOT Cloud Server Setup

3.3.1 Adafruit Dashboard

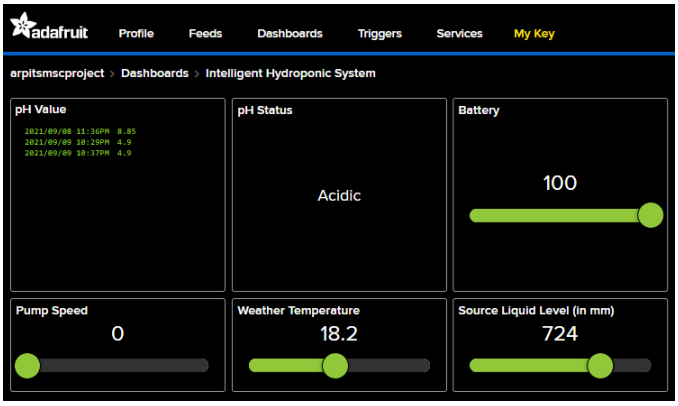

Figure 30: Adafruit Dashboard made for the project

For the IOT cloud server, the free subscription of the adafruit.io was chosen as it offered sufficient services and features for the purpose. 30 data points per minute, included in the free subscription were enough for the project as the pH value, pH status liquid level, pump speed, battery could be updated even after a minute for the user to monitor the system whenever he/she wants. As shown in figure 30, the current value of every single feed used in the project is displayed in the adafruit's cloud server dashboard.

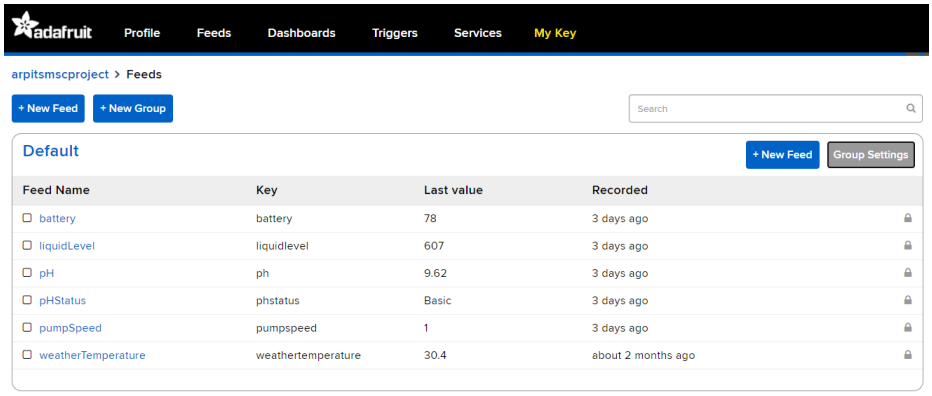

Figure 31: Total number of feeds used in the project

3.3.2 Adafruit feeds

There are six feeds used in the project as shown in the figure 31. The feeds, battery and weatherTemperature are made for the future. They are not being used actively for now.

3.3.3 Notify users by an email when pH not in required range

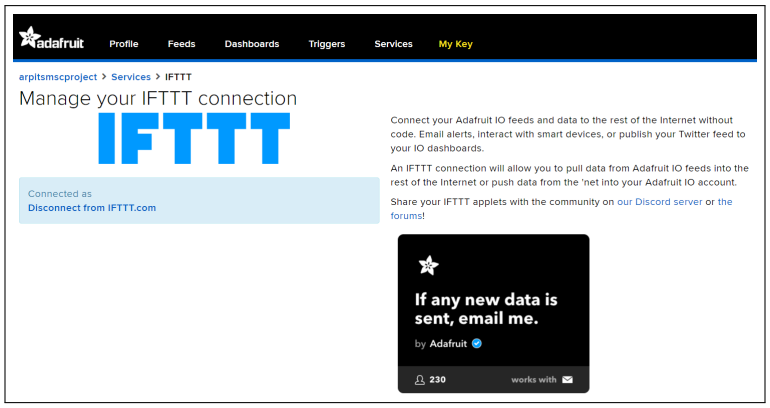

Figure 32: setup at the adafruit.io's end

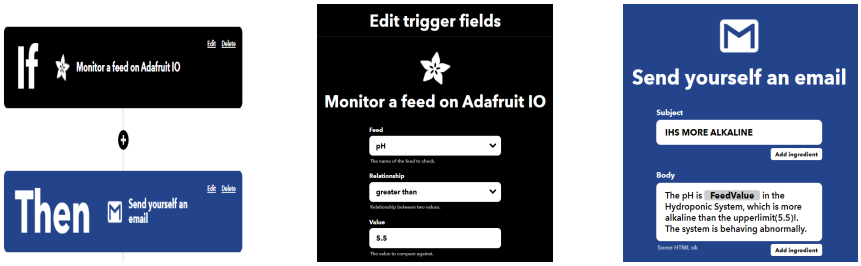

Figure 33: setup at the IFTTT's end

Whenever the pH is not in the required range i.e., whenever the system behaves abnormally, an email alert is sent to the users automatically with the abnormal pH value at that time and asks them to take control of the system. It is a safety feature that has been added to save the growing plants in the main tank of the system in time, if the system doesn't behave as expected. The system has been made using the Adafruit.io 's integration with IFTTT which enables all set of possible 'recipes' that can be made when a feed value crosses a certain threshold set by the user. Therefore, for the system, email alert was chosen as a notification method, as it is a one of the best ways to make the users, aware.

The setup used at the IFTTT's and the adafruit.io's end is shown in figure 33 and figure 32 respectively.

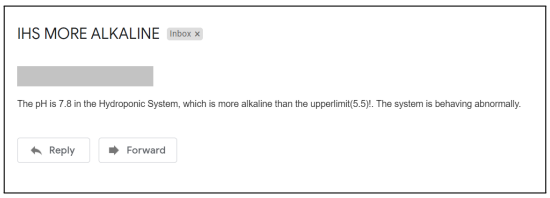

Figure 34: Email alert received when pH not in required range

The subject and the body of the mail was written as follows. So, if the pH increases beyond the range, an email is received as shown in figure 34 by the user almost immediately. if the pH decreases below set lower limit, a corresponding email is sent as well.

3.4 Microcontroller Programming

The Arduino code was written in twenty different files for readability using the Arduino IDE. Some of the important files have been briefly explained in this section.

3.4.1 main.ino

This file contains the main code, which consists of two main methods for programming the microcontroller: void setup and void loop. The declaration of variables happens outside these methods.

The void setup function is used to initiate the microcontroller pins as inputs or outputs depending on whether they are used for sensors or acuators, respectively. It also prepares for the IOT and the AI operations, which have to be later carried out in the void loop function. This function runs just once when the microcontroller is powered on.

The void loop function itself calls most of the functions written across all other files. The functions are written in such a way that each 'send' functions take, time interval in seconds, as a parameter to update their respective latest values after that time interval regularly. Therefore, the pH values are updated after every sixteen seconds, liquid level values after every seventeen seconds, battery Level after eighteen seconds, weather temperature after every nineteen seconds, and pump speed after every twenty seconds. The 'get' functions are used to get the latest values from the sensors or the APIs. The 'set' functions

are all used to set the actuators (pump) at those values(speed in case of the pump).

3.4.2 MQTT.ino

The file is the main file for conducting the IOT operations from the microcontroller using the MQTT protocol. for the simplicity of the project, the WIFI's username and the password have been hard-coded, which are used by the WIFI class to connect the microcontroller to the internet. The IOT cloud server's name, port, username, and password(unique key) are hard-coded as well, which are taken up by an instance of Adafruit_MQTT_client class to connect to the cloud server. All the feeds are initialized as instances of Adafruit_MQTT_Subscribe class or Adafruit_MQTT_Publish class depending on whether the microcontroller subscribes or publishes to those feeds, respectively.

The MOTT_setup function is written to connect to the WIFI and subscribe to the feeds be like pump speed. The MQTT-loop fruition is used to connect and stay connected to the IOT Server. When a message is sent by the cloud, it is received by the respective callback functions of that feed declared in other files, described later in this section.

3.4.3 set_speed.ino

The set_pumpSpeed method receives the speed of the pump as an integer. The integer value is mapped to a value between 0 31 on a scale of 0 to 4095 using the inbuilt map function. Then the speed of the pump is controlled by using Pulse Width modulation by using another inbuilt function, analogWrite.

3.4.4 get_pump_speed.ino

The method is used to get the latest speed of the pump in case the pump is being controlled manually from the app by moving the slider in the pump fragment in the control panel screen of the app via IOT.

The subscribed feed, 'pumpSpeed' is continuously read after every 50 milliseconds using the readsubscription method of the Adafruit_MOTT_ Client Class to get the latest speed in form of a character array, which is later converted to an integer using an inbuilt function, atol. The values are printed to the serial monitor for debugging and troubleshooting purposes using the static function, print of the class, Serial.

3.4.5 send_pump_speed.ino

The method publishes the pump speed calculated by the AI of the microcontroller to the IOT cloud server for it to be displayed in the pump speed fragment by changing the position of the slider according to the speed in the control panel screen of the app.

3.4.6 send_pH.ino

The file is responsible for calculating, and publishing the latest pH values, and pH status (how much acidic or alkaline is the solution) to the Adafruit cloud server. The values are finally printed on the screen of the app. The sensor is calibrated in such a way that the pH values lie between 0 and 14. A neutral solution like pure water is supposed to have a pH of 7.0 and 'pH status', neutral.

3.4.7 read_ultrasonic.ino

The code in this file is takes the HC-SR04 ultrasonic level sensor readings. The time between when the emitter sends the sound-wave, which after getting reflected from the surface of the solution of the nutrient tank is deleted by the receiver is noted and is used to calculate the distance between the sensor and the liquid surface.

The trigger pin is made high for a short period of ten milliseconds, and then made high to send sound waves from the emitter of the sensor. when the receiver detects the wave, the echo pin turns high which is detected by the function, pulseIn. The method returns the needed time duration. Further calculations are made to calculate the exact solution level in the nutrient tank.

3.4.8 send_liquid_level.ino

The file contains the code for sending the latest liquid level in the nutrient tank measured by the ultrasonic sensor to the IOT serves for it to publish it in the liquid level fragment of the control panel screen of the app.

As the values read by the ultrasonic sensor can vary a little, a running median is taken of ten consecutive samples to get rid of the outliers. 'getMedian' function of the 'RunningMedian' Class is used for this purpose.

The latest liquid level is published to the IOT server using the publish method of the Adafruit_MQTT_ Client class after every Seventeen Seconds.

3.4.9 ai.ino

The training of the neural model happens on the computer in a Jupyter Notebook, not on the microcontroller. This file is just responsible for running the pre-trained model to get the required pump speed according to the current pH value.

It also contains the code to calculate the time it took for the microcontroller to conduct one AI inference. It depends upon the complexity of the pre-trained model like the number of hidden layers, number of neurons, etc.

The code has been divided into two functions: ai_loop and ai_setup. Like other files, all libraries and variables are declared and initialized outside these functions. The 'KtensorArenaSize' is the space given to the TensorFlow Lite to do its part by allotting memory to the tensors. its value should be the minimum value for which the model runs successfully, and is found by hit and trial. For the project, the value between 2.2 x 1024 to 4.0 * 1024 were used depending on the model size as the bigger models needed more memory for tensors.

The ai_setup function contains the code for the one-time setup of the error reporters and getting the handle of the neural network model. It also ensures that the scheme version of TensorFlow Lite and the model version are equal, and reports an error, otherwise. The interpreter for the model is also initialized in this section to interpret the inputs and the outputs. Some memory is allocated from the previously defined, ktensorArena to the tensors that are supposed to be used for inputting and outputting data from the model. Lastly, the variables for the inputs and the outputs are initialized with the help of the interpreter.

The ai_loop function is responsible for running an AI inference after every void loop call. For the simple model, the input (X) is the current pH value, and the output (y) is the speed of the pump at that pH. The input is copied to the input tensor, and an error is thrown if there is a size mismatch between them. The inference is run using the invoke method, and the error reporter reports an error if the inference fails. The output is retrieved from the output tensor into a float variable for it to be used by the 'send_pumpSpeed' and 'set_pumpSpeed' functions.

3.5 Training The Neural Network

Pump speed (from 0-100)	pH
20	14
20.4	13.99
20.8	13.98
21.2	13.97
21.6	13.96
22	13.95
22.4	13.94
22.8	13.93
23.2	13.92
23.6	13.91
24	13.9
24.4	13.89
24.8	13.88
25.2	13.87
25.6	13.86
26	13.85
26.4	13.84
26.8	13.83
27.2	13.82
27.6	13.81
28	13.8

Figure 35: The dataset

The training of the neural network model happens on the computer. The dataset is made by sensor data collected from the serial monitor of the Arduino IDE by manually sliding the potentiometer to control the pump speed to achieve the desired pH. The data set is stored in a CSV file (as shown in figure 35) which is used for training the neural network model.

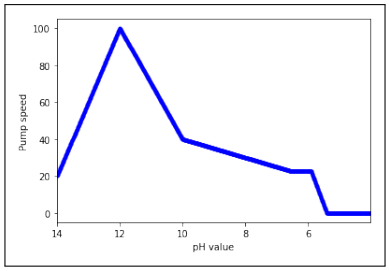

Figure 36: The nature of the dataset

After importing all the important libraries like the ones discussed in the introduction section, the data is read from the dataset file using a custom method, readfile. All the columns in the data set are stored in individual arrays. Then, (X, y) pairs are formed to prepare the data for training. Next, the data is plotted to visualize its nature as shown in figure 36. The x-axis is the pH values from 0 to 14 (the axis is inverted because the pH is being decreased in the experiment as the nutrient solution being added is acidic in nature). The y-axis is the pump speed, which is dependent on the pH value.

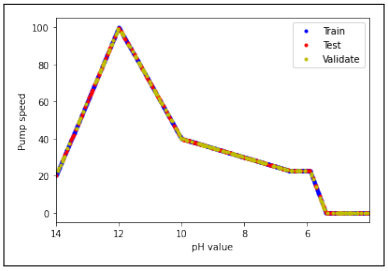

Figure 37: The dataset split in three sections for training

Initially, the data was split into sixty percent for training, twenty percent for validation, and twenty percent for testing. After that, it was made sure that the total number of samples was equal to their sum, to be on a safer side. All the sections of the datasets were individually plotted on the same graph to ensure the random assignment of data to each category as shown in figure 37.

```
# We'll use Keras to create a simple model architecture
model_1 = tf.keras.Sequential()
# First layer takes a scalar input and feeds it through 8 "neurons". The
# neurons decide whether to activate based on the 'relu' activation function.
model_1.add(keras.layers.Dense(8, activation='relu', input_shape=(1,)))
# Final layer is a single neuron, since we want to output a single value
model_1.add(keras.layers.Dense(1))

# Compile the model using a standard optimizer and loss function for regression
model_1.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Figure 38: Smaller model configuration

This regression problem was tried to solve by constructing a sequential model using the Keras library. Initially, there was just a single hidden layer used with only 8 neurons (as shown in figure 38) to have the minimum sized model for the system as the microcontroller has very limited memory and processing power. Also, have the response time as low as possible, which is possible with a small-sized model.

```
Epoch 482/500
10/10 [==============================] - 0s 5ms/step - loss: 324.5599 - mae: 13.8076 - val_loss: 334.7316 - val_mae: 14.4254
Epoch 483/500
10/10 [==============================] - 0s 5ms/step - loss: 324.5530 - mae: 13.7891 - val_loss: 334.8164 - val_mae: 14.4015
Epoch 484/500
10/10 [==============================] - 0s 6ms/step - loss: 324.3248 - mae: 13.7848 - val_loss: 334.5498 - val_mae: 14.4076
Epoch 485/500
10/10 [==============================] - 0s 6ms/step - loss: 324.2207 - mae: 13.7840 - val_loss: 334.3743 - val_mae: 14.4052
Epoch 486/500
10/10 [==============================] - 0s 7ms/step - loss: 324.1176 - mae: 13.7798 - val_loss: 334.2890 - val_mae: 14.3948
Epoch 487/500
10/10 [==============================] - 0s 6ms/step - loss: 323.9589 - mae: 13.7809 - val_loss: 333.9080 - val_mae: 14.4105
Epoch 488/500
10/10 [==============================] - 0s 8ms/step - loss: 323.8824 - mae: 13.7846 - val_loss: 333.8738 - val_mae: 14.3965
Epoch 489/500
10/10 [==============================] - 0s 7ms/step - loss: 323.7616 - mae: 13.7776 - val_loss: 333.7290 - val_mae: 14.3910
Epoch 490/500
10/10 [==============================] - 0s 4ms/step - loss: 323.6549 - mae: 13.7717 - val_loss: 333.5967 - val_mae: 14.3851
Epoch 491/500
10/10 [==============================] - 0s 5ms/step - loss: 323.5499 - mae: 13.7606 - val_loss: 333.5950 - val_mae: 14.3689
Epoch 492/500
10/10 [==============================] - 0s 4ms/step - loss: 323.4972 - mae: 13.7605 - val_loss: 333.2658 - val_mae: 14.3802
Epoch 493/500
10/10 [==============================] - 0s 5ms/step - loss: 323.4026 - mae: 13.7762 - val_loss: 332.8932 - val_mae: 14.3955
Epoch 494/500
10/10 [==============================] - 0s 5ms/step - loss: 323.1864 - mae: 13.7745 - val_loss: 332.8244 - val_mae: 14.3840
Epoch 495/500
10/10 [==============================] - 0s 5ms/step - loss: 323.1655 - mae: 13.7819 - val_loss: 332.4503 - val_mae: 14.3995
Epoch 496/500
10/10 [==============================] - 0s 5ms/step - loss: 322.9927 - mae: 13.7756 - val_loss: 332.5065 - val_mae: 14.3768
Epoch 497/500
10/10 [==============================] - 0s 5ms/step - loss: 322.9759 - mae: 13.7479 - val_loss: 332.6951 - val_mae: 14.3429
Epoch 498/500
10/10 [==============================] - 0s 5ms/step - loss: 322.7970 - mae: 13.7228 - val_loss: 332.5739 - val_mae: 14.3360
Epoch 499/500
10/10 [==============================] - 0s 6ms/step - loss: 322.6393 - mae: 13.7291 - val_loss: 332.2199 - val_mae: 14.3493
Epoch 500/500
10/10 [==============================] - 0s 6ms/step - loss: 322.5226 - mae: 13.7306 - val_loss: 332.0695 - val_mae: 14.3448
```

Figure 39: The epoch training of the smaller model

The mean absolute error, training loss and the validation loss were both extremely high at 1373 percent and 1434 percent, respectively at this configuration (as shown in figure 39).

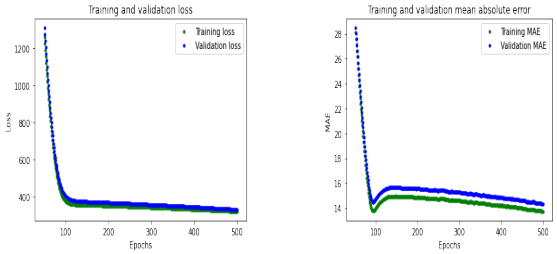

Figure 40: The graph of smaller model's mean absolute error

The mean square error versus epochs graph for both validation loss and training loss was plotted to see whether validation loss catches up with the training loss or not. The graphs depict the difference between the actual date and the model's predictions. we also calculated and plotted the mean absolute error as shown in figure 40 to see the detailed difference between the actual data, and the model's predictions. The mean absolute error turned up to be much higher than the mean squared error.

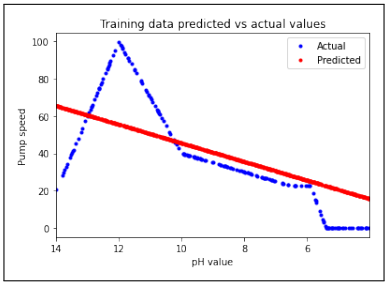

Figure 41: The graph of smaller model's actual Vs. predicted values

The actual data and the predicted values on the input versus output graph (as shown in figure 41) gives the true picture of how off the predicted values are from actual ones. Hence, it couldn't satisfy the need of the system because of its inefficiency to meet the accuracy needed for maintaining pH for growing healthy plants in the hydroponic system.

Hence, we had to increase the size of the model by adding the number of neurons, hidden layers, epochs and adjusting other parameters as well to properly tune the model.

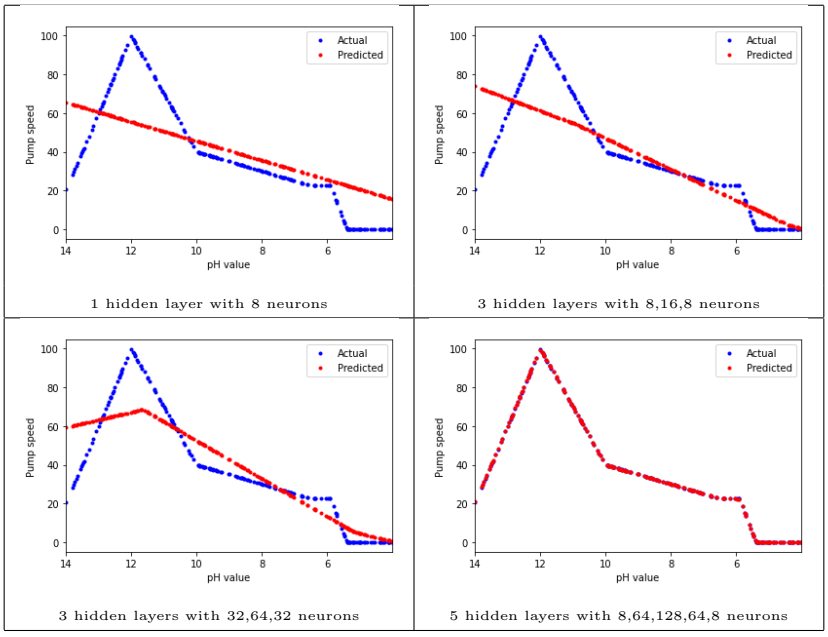

Table 1: Comparison of predicted and actual values in intermittent models tried during the tuning of the neural modal

The graphs of predicted and actual values for the intermittent models tried during the tuning of the neural network modal are given in table 1. It can be seen that due to the complexity (sharp and rapid changes in the graph) of the dataset, the output cannot keep up with the actual values at lower model configurations. Therefore, number of layers and number of neurons had to be increased.

```
model_2 = tf.keras.Sequential()

# First layer takes a scalar input and feeds it through 16 "neurons". The
# neurons decide whether to activate based on the 'relu' activation function.

model_2.add(keras.layers.Dense(32, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(64, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(128, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(64, activation='relu', input_shape=(1,)))
model_2.add(keras.layers.Dense(32, activation='relu', input_shape=(1,)))

# Final layer is a single neuron, since we want to output a single value
model_2.add(keras.layers.Dense(1))

# Compile the model using a standard optimizer and loss function for regression
model_2.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Figure 42: The configurations of the optimum model

Ultimately, a larger model (as shown in the figure 42) with five 'Relu' activated hidden layers with the number of neurons equal to 32, 64, 128, 64, 32 respectively with one neuron at the input layer for the pH value, and one neuron at the output layer for the pump speed was made.

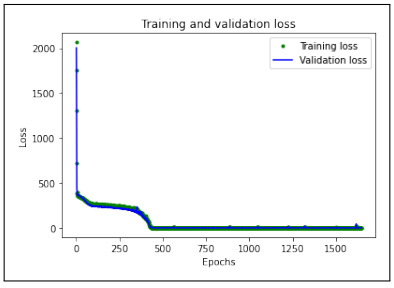

Figure 43: The graph of the optimum model's mean squared error

As the number of epochs increases, the loss decreases and flattens around

270 epochs (as shown in figure 43), but still continues to decrease gradually. So, increasing the number of epochs is desirable as accuracy is paramount for hydroponic systems. It is also the reason, early stopping feature was not preferred while training the model.

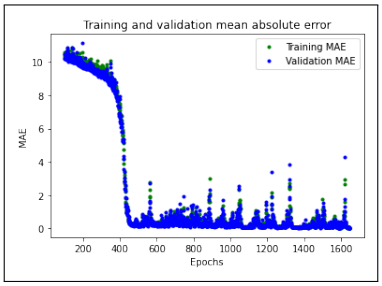

Figure 44: The graph of the optimum model's mean absolute error

The mean absolute error (as shown in figure 44) turns out to be a bit higher than the mean squared error but almost paints the same picture.

3.6 K-fold cross-validation technique

```
Epoch 1628/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0089 - mae: 0.0721 - val_loss: 0.0084 - val_mae: 0.0679
Epoch 1629/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0062 - mae: 0.0588 - val_loss: 0.0058 - val_mae: 0.0635
Epoch 1630/1639
10/10 [==============================] - 0s 8ms/step - loss: 0.0062 - mae: 0.0562 - val_loss: 0.0127 - val_mae: 0.0748
Epoch 1631/1639
10/10 [==============================] - 0s 8ms/step - loss: 0.0134 - mae: 0.0793 - val_loss: 0.0102 - val_mae: 0.0689
Epoch 1632/1639
10/10 [==============================] - 0s 9ms/step - loss: 0.0104 - mae: 0.0762 - val_loss: 0.0098 - val_mae: 0.0847
Epoch 1633/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0042 - mae: 0.0515 - val_loss: 0.0069 - val_mae: 0.0672
Epoch 1634/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0050 - mae: 0.0581 - val_loss: 0.0126 - val_mae: 0.1003
Epoch 1635/1639
10/10 [==============================] - 0s 8ms/step - loss: 0.0047 - mae: 0.0539 - val_loss: 0.0044 - val_mae: 0.0407
Epoch 1636/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0041 - mae: 0.0475 - val_loss: 0.0086 - val_mae: 0.0694
Epoch 1637/1639
10/10 [==============================] - 0s 9ms/step - loss: 0.0054 - mae: 0.0526 - val_loss: 0.0046 - val_mae: 0.0518
Epoch 1638/1639
10/10 [==============================] - 0s 7ms/step - loss: 0.0059 - mae: 0.0536 - val_loss: 0.0021 - val_mae: 0.0367
Epoch 1639/1639
10/10 [==============================] - 0s 8ms/step - loss: 0.0038 - mae: 0.0445 - val_loss: 0.0015 - val_mae: 0.0280
```

Figure 45: The epoch training of the optimum model using K-fold cross-validation technique

A reliable and minimum error starts showing up around 1650 epochs (as shown in figure 45) implying that a minimum of 1650 epochs approximately are needed to train the system. In attempt to further increase the accuracy of the model,

instead of taking a fixed data split for training the neural modal, K-fold cross-validation [54] was used to find the best data split that would give the highest accuracy. Then the model was trained on the best K-fold to achieve the lowest mean absolute loss. This increased the performance of the model and made the training process more robust. In the final model, five folds with 300 epochs each were used to find the best data split, and 1639 epochs for finally training the model on that data split.Again, it is also made sure that the validation loss is not more than the training loss. Otherwise, the model stops generalizing to new data.

3.7 Optimizers and learning rate

Different optimizers like 'Adam' optimizer, 'RMSprop', and 'Adadelta' optimizers were tried during the experimentation. The best results were gained by the 'Adam' optimizers at 6 percent validation loss and extremely bad by the 'RMSprop' optimizer at 116 percent validation loss. This meant that the 'RMSprop' optimizer doesn't work for this system at all. Different learning rates were tried but the 'Adam' optimizer surpassed all other optimizers every time. Finally 'Adam' optimizer with a 0.001 learning rate was selected for training the system.

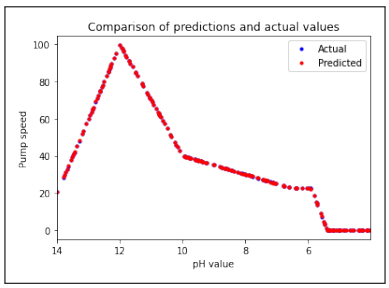

Figure 46: The graph of optimum model's actual Vs. predicted values

Finally, the actual data and the predicted values on the input vs output graph (as shown in figure 46) describes how accurately the predicted values are on the mark, and the system is predicting reliable results. The only drawback of this model is its immense size making it very difficult to run it on a tiny low powered chip. A method of reducing the size of the model was supposed to be found.

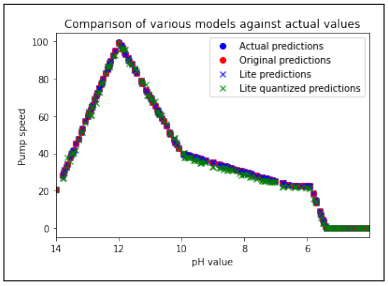

Figure 47: The comparison of the quantized model with the unquantized model

Hence, the model is then quantized to reduce its size so it to be easily accommodated by the microcontroller with its limited memory space and processing power. The accuracy of the quantized and the unquantified models are compared on the graph visually to verify that the performance of the quantized doesn't lag significantly behind the quantized model. The graphs of the actual data, and the predictions made by the unquantized model, and the quantized model are plotted (as shown in figure 47), for making the comparison. It turns up that there is no significant difference in their performances making the quantized model ideal to be run on the microcontroller.

```
 # Install xxd if it is not available
!apt-get update && apt-get -qq install xxd
# Convert to a C source file
!xxd -i {MODEL_TFLITE} > {MODEL_TFLITE_MICRO}
# Update variable names
REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
!sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
# Print the C source file
!cat {MODEL_TFLITE_MICRO}
```

Lastly, the trained model is converted to a C source file (using the above code snippet) for it to be exported to a micrcontroller.

4 Testing On The Physical System

To test the model on the real apparatus to see the practicality of the system, the pH values in the dataset were fed to the AI on the microcontroller, instead of getting them from the pH sensor, and the outputs were compared with the expected values.

The error between the expected values and the original values was calculated to see how accurate the AI on edge is in predicting the speed of the pump according to the pH values.

```
double pHValue=14;
```

```
void loop(){
        ..............
        ..............
    // Sets the speed of the pump through AI according to pH value
    ai_loop();
    // print space separated pairs of pH value and pump speed
    Serial.print(pHValue);
    Serial.print(" ");
    Serial.println(pumpSpeed);
    // decrement 0.01 pH value every cycle
    pHValue -= 0.01;
    // when the pH value reaches the minimum, exit from the void loop
    if (pHValue < 0) {
        exit(0);
    }
}
```

The above code snippet was added for this data collection purpose and was uploaded on the microcontroller. The data was collected from the serial monitor, which was later analysed to calculate the error and plot graphs.

4.1 Model size versus Time per inference

The size of the model being run in the Arduino IDE is the number of comma-separated hexadecimal numbers required to represent it. The time taken per each AI inference on the microcontroller/edge is plotted with the size of the model to study the effect of the size of the model on the processing time require for each inference by the microcontroller for conducting machine learning operations on edge. The inference time and the sizes of different-sized models were noted to study the pattern.

4.2 Training and validation losses for various models

The training loss, which determines the efficiency of the model in fitting in the training data, and the validation loss, which describes its efficiency in fitting to an unseen data, were noted according to both mean squared error and mean absolute error. The model with the lowest overall loss was chosen as the optimum model for the experiment.

Model Number	Model Configerations	Training Loss (MSE)	Training Loss (MAE)	Validation Loss (MSE)	Validation Loss (MAE)
1	1 hidden layer with 8 neurons	322.5226	13.7306	332.0695	14.3448
2	3 hidden layers with 4,8,4 neurons respectively	281.0879	11.3358	273.5565	11.5603
3	3 hidden layers with 8,16,8 neurons respectively	260.8473	10.4382	242.9051	10.2793
4	3 hidden layers with 16,32,16 neurons respectively	247.3403	10.1742	229.8249	9.9623
5	3 hidden layers with 32,64,32 neurons respectively	160.5955	8.9749	149.886	8.9081
6	3 hidden layers with 32,64,33 neurons respectively	0.0121	0.0797	0.0161	0.1048
7	3 hidden layers with 64,128,64 neurons respectively	0.0417	0.1685	0.0385	0.1328
8	5 hidden layers with 8,64,128,64,8 neurons respectively	0.011	0.074	0.0073	0.0617
9	5 hidden layers with 16,64,128,64,16 neurons respectively	0.0844	0.2216	0.0095	0.0624
10	5 hidden layers with 32,64,128,64,32 neurons respectively	0.0581	0.1969	0.0451	0.1934
11	7 hidden layers with 8,32,64,128,64,32,8 neurons respectively	0.0294	0.1172	0.0554	0.174
12	5 hidden layers with 8,64,128,64,8 neurons respectively using K-fold cross validation	**0.0038**	**0.0445**	**0.0015**	**0.028**

Figure 48: Training and Validation losses for the prominent models tried during the experiment with the K-fold cross validation technique delivering minimum error

Model number, twelve (as shown in table 48) showed the lowest loss for every loss calculation method with 0.38 percent and 0.15 percent, training loss and validation loss respectively based on mean squared error, and 4.45 percent and 2.80 percent training and validation loss respectively based on mean absolute error, making it the 'fittest' model. The model has 5 hidden layers with 8,64,128,64,8 neurons respectively, 'Adam' optimizer with the 'learning rate' of 0.001, 'Relu'-activated hidden layers with 64 batch size, and 1639 epochs and uses K-fold cross validation configurations discussed before.

5 The Physical Apparatus Of The Proposed System

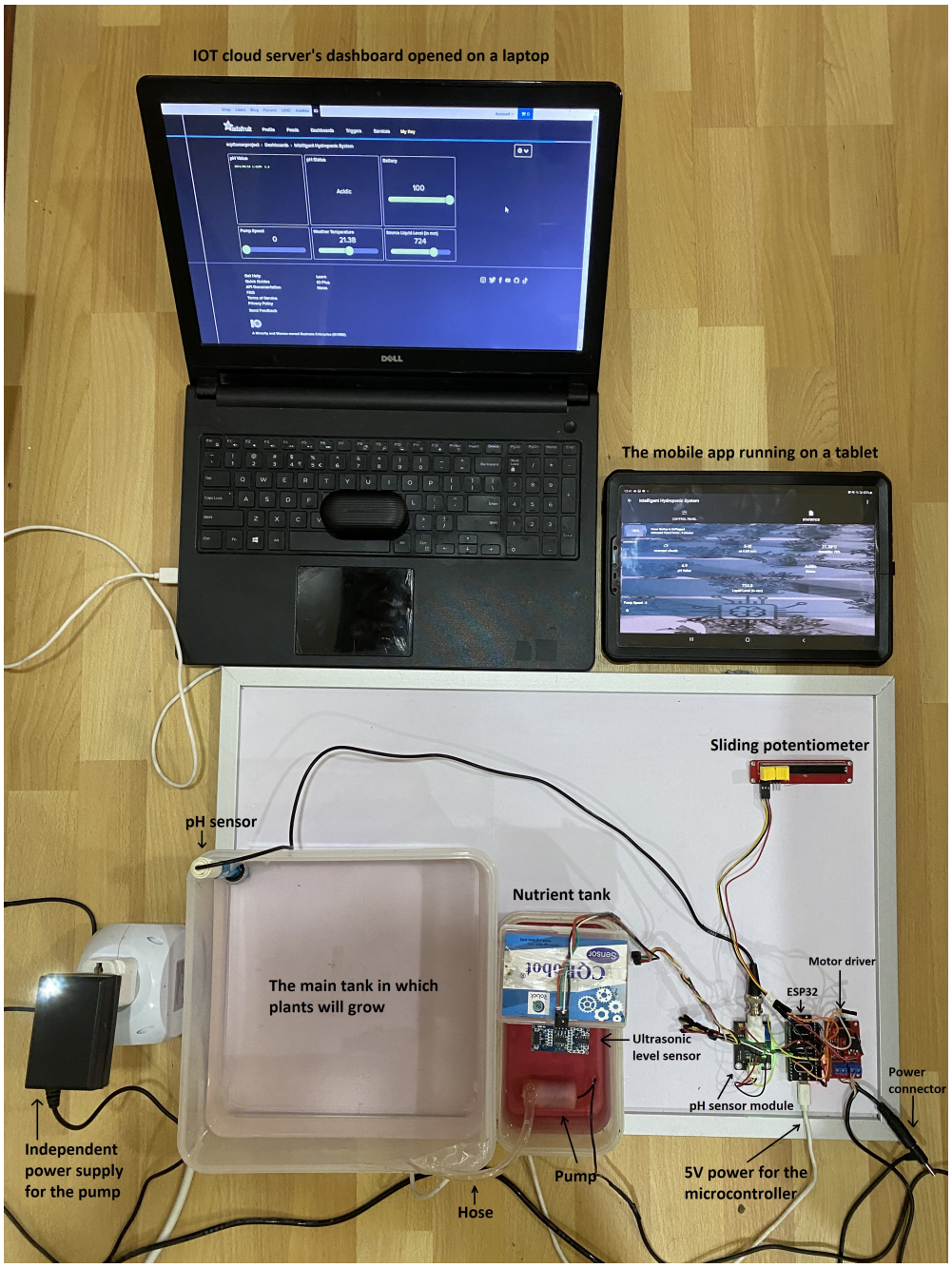

Figure 49: The labeled diagram of the physical apparatus

The physical apparatus of the system is shown in figure 49. This prototype of the hydroponic system was used for the experiment including the collection of data for the neural model, and for testing the artificial intelligence and IOT operations. The system is well connected wirelessly with the mobile app running on the tablet and the laptop via IOT and is showing the current system status as shown in the figure. The laptop and the tablet can get the live feed of the system remotely from any part of the world as long as they have internet access.

6 Results

6.1 Model size versus Time per inference

Model Configurations	Epochs	Model Size	Time per inference on the microcontroller (in microseconds)	KtensorArenaSize
1 hidden layer with 8 neurons	500	1784	131	2.2 * 1024
3 hidden layers with 4,8,4 neurons respectively	500	2872	230	2.2 * 1024
3 hidden layers with 8,16,8 neurons respectively	500	3128	268	2.2 * 1024
3 hidden layers with 16,32,16 neurons respectively	500	4040	375	2.2 * 1024
3 hidden layers with 32,64,32 neurons respectively	500	7400	779	2.2 * 1024
3 hidden layers with 32,64,33 neurons respectively	1650	7400	779	2.2 * 1024
3 hidden layers with 64,128,64 neurons respectively	1650	20264	2315	2.2 * 1024
5 hidden layers with 8,64,128,64,8 neurons respectively	1650	22240	2623	3.2 * 1024
5 hidden layers with 16,64,128,64,16 neurons respectively	1650	23344	2808	3.2 * 1024
5 hidden layers with 32,64,128,64,32 neurons respectively	1650	25552	3161	3.2 * 1024
7 hidden layers with 8,32,64,128,64,32,8 neurons respectively	1650	27080	3313	4.0 * 1024
5 hidden layers with 8,64,128,64,8 neurons respectively using K-fold cross validation	1639	22352	2606	3.2 * 1024

Figure 50: The table of model size vs the time taken by the microcontroller per inference

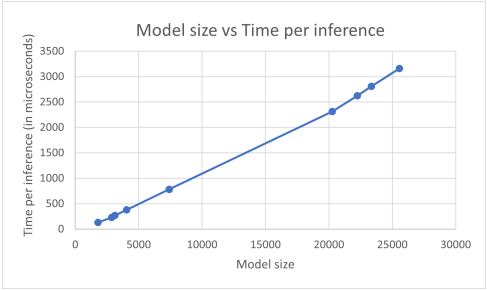

Figure 51: The graph of model size vs the time taken by the microcontroller per inference

As it can be seen in table 50 and figure 51 that the graph of model size versus time per inference is almost a straight line implying that time per inference is directly proportional to the model size. A little bend appears after 25552 sized model, because the KtensorArenaSize was increased from 3.2 *1024 to 4.0 * 1024 from there on, as can be seen in table 50. As the KtensorArenaSize is the space allotted to the tensors for ML tasks by the microcontroller, it also positively contributes to the time per inference of the model. Again, proving the principle that the KtensorArenaSize should be kept as minimum as possible to reduce the latency of the system.

From the fifth and sixth models in the table, it is clear that the number of epochs has no effect on the model size and the time take per inference by the microcontroller

The time per inference was tested again and again for the same models to see the precision. The time was almost the same for the same models implying that the microcontroller can be trusted for the time, it will be taking for each inference for a fixed model size.

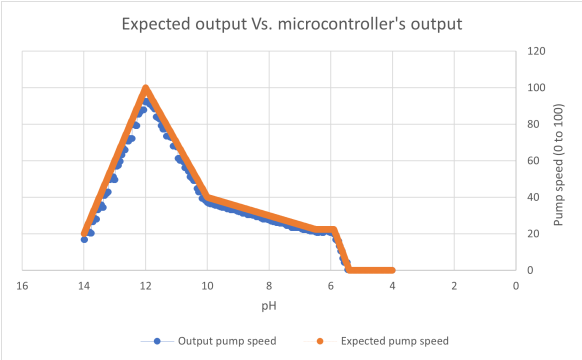

Figure 52: Expected pump speed Vs. physical system's pump speed according to pH values

The predicted speed turned out to be quite reliable as it can be seen in figure 52 that the graphs of expected speed and the predicted speed almost overlap. The predicted speed just lags the expected speed by an average of 2.94 and a median of 2.56 difference in the speed (speed being in range 0-100), but mostly they are equal as the mode of the absolute errors turns out to be zero (that is when the pump is off). The maximum error was found out to be 10.81, which is a bit high, but still, it was on the sharpest slope in the graph as the number of data points in that area are not enough for the model to be trained properly in that area.

Most important is the pump to be in the off state when it is expected to be off as it is very dangerous for the system to have the pump false triggered as it can keep slowly pouring the nutrient solution in the main tank eventually making the main tank's solution highly acidic, and unfit for plant growth. But It can be seen from table in the appendices that the speed predictions at low-speed levels are very accurate making the system free from this danger.

7 Conclusion

Following conclusions were drawn after the completion of the experiment.

7.1 About the overall performance of the system

A very low response time of fewer than three milliseconds is achieved, which is good as compared to what it would take for an "AI in the cloud" system. Most importantly, as can be seen in the table in the appendix that the inference time is very precise. If it wasn't the case, it could lead to system failure. Hence, the

system is very reliable and robust.

The only challenge is a bit high error of 2.56 percent on the real system. It can be further improved by fine-tuning and rigorous testing.

It is advised to the expert controlling the potentiometer while the data collection process, to make smooth adjustments. It is very difficult to train the model with high accuracy if there are irregular sharp edge on the pH versus pump speed graph. It would take a very big modal to adapt to it, which may not fit on the ESP32 chip.

After the data collection, it is also very important to pre-process the data, to get rid of outliers, add some fillers where data points are missing to make a good dataset out of it for the neural network to be able to get trained on it and achieve high accuracy.

7.2 About the apparatus

An isolated circuitry for the pH sensor is required for the pH sensor to avoid the problem of ground loop formation. Therefore, the other modules like motor drivers should be selected such as their circuitry doesn't interfere with that of the pH sensor.

It is vital for the success of the system, to have a fixed shape for the physical containers and fittings, so that the AI which is trained on a particular shape and size of the apparatus can perform properly. Similarly, if the pump is replaced with a pump of different specifications, it may be required to retrain the AI on edge model according to the new dataset.

The apparatus just incorporates a single nutrient tank which being itself acidic, and hence, can only make the main tank more acidic. There is no way of restoring the alkalinity of the main tank if it falls below the desired range. It is believed that two tanks, one acidic and the other alkaline with two pumps(one each) will be needed for making a proper controller.

7.3 About the mobile app

The mobile app is robust and doesn't crash. It also has the functionality to open with the last state with which it was previously closed. Therefore, the app needs no major further improvements except a login page for offering security to the users.

7.4 About the ESP32 microcontroller and IOT

The ESP32 development board was used on large sized AI models, and it functions properly without failing anytime. Because of the board's immense processing speed despite its tiny physical size and extremely low power consumption, it is possible to try a wide range of model sizes on it along with IOT operations, very easily.

If the response time must be further reduced, the model size can be reduced further, but it can be a trade-off with the accuracy of the system.

It is a completely automated system, and doesn't necessarily need any human intervention, but has a reliable mechanism of notifying the user through email alerts for them to stay cautious of the system status. All the other functionalities of the system still work when the internet connection is not available or is interrupted.

8 Next Steps

A deeper tuning and testing of the AI on edge can be done to optimise the system's performance even more.

For the time being, the data must be collected by physically attaching the computer to the system. It is believed that it should be also possible in the future to send the sensor data as training data via IOT to the remote computer for updating the neural models by retraining it on the new data, and sends the model back via internet in small data packets. This would make the system more robust and up to date.

The pH monitoring is also very important in other systems like aquariums for maintaining a safe environment for aquatic lives. And not just pH monitoring, other parameters like electrical conductivity, temperature, etc. are also needed to be monitored and controlled in these types of systems. Therefore, the system can be easily extended to other similar systems by making a few hardware and software changes.

9 Final Thoughts

Building different components of the system, troubleshooting them, and then making them work in sync with each other was an extremely challenging task. Sometimes, there would be problems with the apparatus like a sensor or an actuator would stop working properly or an issue in the mobile application's

development process in the Android Studio would appear, or the AI of the microcontroller would get stuck. These problems sometimes halted the entire experimentation, but with the support and guidance of Dr. Aboozar Taherkhani, all those problems were solved in time, leading to the successful completion of the project. Being a roboticist myself with two patent applications in the field including "Robotic Gardening Device", working on this project was very enjoyable, and was working tirelessly day and night on it.

Completing the project has given me a good experience in mobile app development with Android Studio IDE in the java programming language, microcontroller programming with Arduino IDE in C programming language, IOT using Adafruit cloud server, AI in Google Colab, and VS code editor in the python programming language, and in state of art, TinyML. I hope these skills will be beneficial in the future as well. I would hope to get an opportunity to implement the proposed system by extending it with more vigorous testing and sophisticated equipment on a real farm.

References

[1] Colby R Banbury et al. "Benchmarking TinyML systems: Challenges and direction". In: *arXiv preprint arXiv:2003.04821* (2020).

[2] Mamta D Sardare and Shraddha V Admane. "A review on plant without soil-hydroponics". In: *International Journal of Research in Engineering and Technology* 2.3 (2013), pp. 299–304.

[3] VC Spinu, RW Langhans, and LD Albright. "Electrochemical pH control in hydroponic systems". In: *II Modelling Plant Growth, Environmental Control and Farm Management in Protected Cultivation 456* (1997), pp. 275–282.

[4] Android Developers. "What is android?" In: *Dosegljivo: http://www. academia. edu/download/30551848/andoid–tech. pdf* (2011).

[5] Dieter Helm. "Agriculture after brexit". In: *Oxford Review of Economic Policy* 33.suppl_1 (2017), S124–S133.

[6] Anna Jurga et al. "A Long-Term Analysis of the Possibility of Water Recovery for Hydroponic Lettuce Irrigation in an Indoor Vertical Farm. Part 2: Rainwater Harvesting". English. In: *Applied sciences* 11.310 (2021). Publisher: MDPI AG, p. 310. DOI: 10.3390/app11010310. URL: https://go.exlibris.link/Cqbh5RXR.

[7] Timothy Tripp. *Hydroponics advantages and disadvantages: Pros and cons of having a hydroponic garden*. Speedy Publishing LLC, 2014.

[8] Sulma Vanessa Souza, Régio Marcio Toesca Gimenes, and Erlaine Binotto. "Economic viability for deploying hydroponic system in emerging countries: A differentiated risk adjustment proposal". In: *Land Use Policy* 83 (2019), pp. 357–369.

[9] Ayman F Abou-Hadid et al. "Electrical conductivity effect on growth and mineral composition of lettuce plants in hydroponic system". In: *Strategies for Market Oriented Greenhouse Production 434* (1995), pp. 59–66.

[10] Matt Richardson and Shawn Wallace. *Getting started with raspberry PI.* " O'Reilly Media, Inc.", 2012.

[11] Agus Kurniawan. "Arduino Nano 33 BLE Sense Board Development". In: *IoT Projects with Arduino Nano 33 BLE Sense*. Springer, 2021, pp. 21–74.

[12] Marek Babiuch, Petr Foltnek, and Pavel Smutn. "Using the ESP32 microcontroller for data processing". In: *2019 20th International Carpathian Control Conference (ICCC)*. IEEE. 2019, pp. 1–6.

[13] Yogendra Singh Parihar. "Internet of Things and Nodemcu". In: *Journal of Emerging Technologies and Innovative Research* 6.6 (2019), p. 1085.

[14] Shuangfeng Li. "Tensorflow lite: On-device machine learning framework". In: *Journal of Computer Research and Development* 57.9 (2020), p. 1839.

[15] Meena Singh et al. "Secure mqtt for internet of things (iot)". In: *2015 fifth international conference on communication systems and network technologies*. IEEE. 2015, pp. 746–751.

[16] Nikhil Ketkar. "Introduction to keras". In: *Deep learning with Python*. Springer, 2017, pp. 97–111.

[17] Mohamed Fezari and Ali Al Dahoud. "Integrated Development Environment "IDE" For Arduino". In: *WSN applications* (2018), pp. 1–12.

[18] Jerome DiMarzio. *Beginning Android Programming with Android Studio*. John Wiley & Sons, 2016.

[19] Steve Vinoski. "Advanced message queuing protocol". In: *IEEE Internet Computing* 10.6 (2006), pp. 87–89.

[20] Zach Shelby, Klaus Hartke, and Carsten Bormann. "The constrained application protocol (CoAP)". In: (2014).

[21] Peter Saint-Andre et al. "Extensible messaging and presence protocol (XMPP): Core". In: (2004).

[22] James Steele and Nelson To. *The Android developer's cookbook: building applications with the Android SDK*. Pearson Education, 2010.

[23] Hardeep Singh and Dunn Bruce. "Electrical conductivity and pH guide for hydroponics". In: *Oklahoma Cooperative Extension Fact Sheets, HLA-6722. Oklahoma State University, Division of Agricultural Sciences and Natural Resources* (2016), p. 5.

[24] Hanif Fakhrurroja et al. "Automatic ph and humidity control system for hydroponics using fuzzy logic". In: *2019 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*. IEEE. 2019, pp. 156–161.

[25] Theeramet Kaewwiset and Thongchai Yooyativong. "Electrical conductivity and pH adjusting system for hydroponics by using linear regression". In: *2017 14th International Conference on Electrical Engineering/-Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. IEEE. 2017, pp. 761–764.

[26] Judith. *pH in Hydroponics: How to Maintain the pH Levels of Hydroponic Systems*. en-us. URL: `https://blog.jencoi.com/ph-in-hydroponics-how-to-maintain-the-ph-levels-of-hydroponic-systems` (visited on 08/27/2021).

[27] JV Gosavi et al. "Water monitoring system for hydroponics agriculture." In: *International Journal for Research in Applied Science and Engineering Technology* 5.7 (2017), pp. 234–238.

[28] Chanya Peuchpanngarm et al. "DIY sensor-based automatic control mobile application for hydroponics". In: *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*. IEEE. 2016, pp. 57–60.

[29] S Charumathi et al. "Optimization and control of hydroponics agriculture using IOT". In: *Asian J. Appl. Sci. Technol* 1.2 (2017), pp. 96–98.

[30] Oleksii Barybin, Elina Zaitseva, and Volodymyr Brazhnyi. "Testing the security ESP32 internet of things devices". In: *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*. IEEE. 2019, pp. 143–146.

[31] Nico Surantha et al. "Intelligent monitoring and controlling system for hydroponics precision agriculture". In: *2019 7th international conference on information and communication technology (ICoICT)*. IEEE. 2019, pp. 1–6.

[32] Pete Warden and Daniel Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

[33] Eric Flamand et al. "GAP-8: A RISC-V SoC for AI at the Edge of the IoT". In: *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE. 2018, pp. 1–4.

[34] Manav Mehra et al. "IoT based hydroponics system using Deep Neural Networks". In: *Computers and electronics in agriculture* 155 (2018), pp. 473–486.

[35] F. Ludwig et al. “Electrical conductivity and pH of the substrate solution in gerbera cultivars under fertigation [Condutividade elétrica e pH da solução do substrato em cultivares de gérbera fertirrigadas]”. In: *Horticultura Brasileira* 31.3 (2013). cited By 8, pp. 356–360. DOI: 10.1590/S0102-05362013000300003. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84886286761&doi=10.1590%2fS0102-05362013000300003&partnerID=40&md5=3c767250fc9fbbab7c856eb470b6fb61.

[36] KP Ferentinos and LD Albright. “Predictive neural network modeling of pH and electrical conductivity in deep–trough hydroponics”. In: *Transactions of the ASAE* 45.6 (2002), p. 2007.

[37] R Vadivel et al. “Hypaponics-monitoring and controlling using Internet of Things and machine learning”. In: *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*. IEEE. 2019, pp. 1–6.

[38] Andreas Komninos et al. “Improving Hydroponic Agriculture through IoT-enabled Collaborative Machine Learning”. In: ().

[39] *Visual Studio Code - Code Editing. Redefined*. URL: https://code.visualstudio.com/ (visited on 09/16/2021).

[40] Tatiana Sergeevna Volokitina. “Analysis of Google Colab possibilities”. In: 2 (2021), pp. 12–12.

[41] *Analog pH sensor meter kit, cable of the shielding probe board, meter monitoring Analog pH sensor kit for Arduino : Amazon.co.uk: Home & Kitchen*. URL: https://www.amazon.co.uk/gp/product/B083XTLH3L/ref=ppx_yo_dt_b_asin_title_o09_s00?ie=UTF8&psc=1 (visited on 08/25/2021).

[42] *pH Meter Care and Common Mistakes — ThermoWorks*. URL: https://blog.thermoworks.com/thermometer/ph-meter-care-and-common-mistakes/ (visited on 09/05/2021).

[43] Michael Barr. “Pulse width modulation”. In: *Embedded Systems Programming* 14.10 (2001), pp. 103–104.

[44] Thomas H Martin and Delta Chemicals. “Troubleshooting pH/ORP Controllers”. In: *PRODUCTS FINISHING-CINCINNATI-* 59 (1995), pp. 56–56.

[45] *DIGITEN 0.3-6L/min G1/4” Water Coffee Flow Hall Sensor Switch Meter Flowmeter Counter Connect Hose : Amazon.co.uk: Business, Industry & Science*. URL: https://www.amazon.co.uk/gp/product/B072JVL5VG/ref=ppx_yo_dt_b_asin_title_o02_s01?ie=UTF8&psc=1 (visited on 08/25/2021).

[46] Michał Urbański et al. “Flowmeter Converter Based on Hall Effect Sensor”. In: *Progress in Automation, Robotics and Measuring Techniques*. Springer, 2015, pp. 265–276.

[47] *CQRobot Ocean: Contact Water/Liquid Level Sensor Compatible with Raspberry Pi/Arduino. for Automatic Irrigation Systems, Aquariums, Plants, in the Garden, in Agriculture etc. : Amazon.co.uk: Business, Industry & Science*. URL: https://www.amazon.co.uk/gp/product/B07ZMGW3QJ/ref=ppx_yo_dt_b_asin_title_o02_s01?ie=UTF8&psc=1 (visited on 08/25/2021).

[48] VA Zhmud et al. "Application of ultrasonic sensor for measuring distances in robotics". In: *Journal of Physics: Conference Series*. Vol. 1015. 3. IOP Publishing. 2018, p. 032189.

[49] *RUNCCI-YUN Automatic Irrigation DIY Kit Self Watering System with Capacitive Soil Moisture Sensor 1 Channel 5V Relay Module and Water Pump + 1M Vinyl Tubing for Garden Plant Flower Herb Potted : Amazon.co.uk: Pet Supplies*. URL: https://www.amazon.co.uk/gp/product/B0814HXWVV/ref=ppx_yo_dt_b_asin_title_o08_s00?ie=UTF8&psc=1 (visited on 08/25/2021).

[50] *HALJIA 1 Channel MOSFET Switch IRF540 Isolated Power Compatible with Arduino DIY etc. : Amazon.co.uk: Business, Industry & Science*. URL: https://www.amazon.co.uk/gp/product/B06XB5TPVG/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1 (visited on 08/25/2021).

[51] Kristian Dokic, Marko Martinovic, and Dubravka Mandusic. "Inference speed and quantisation of neural networks with TensorFlow Lite for Microcontrollers framework". In: *2020 5th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. IEEE. 2020, pp. 1–6.

[52] Joshua Kim. "Analysis and optimization of DC supply range for the ESP32 development board". In: (2020).

[53] *HALJIA 10K Logarithmic Slide Potentiometer Log Potentiometer Dual Output Linear Trim Pot Module Compatible with Arduino AVR Electronic Block : Amazon.co.uk: Business, Industry & Science*. URL: https://www.amazon.co.uk/gp/product/B076PYGFFP/ref=ppx_yo_dt_b_asin_title_o07_s00?ie=UTF8&psc=1 (visited on 08/25/2021).

[54] Davide Anguita et al. "The 'K'in K-fold cross validation". In: *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. i6doc. com publ. 2012, pp. 441–446.

Appendix

9.1 Model outputs and errors

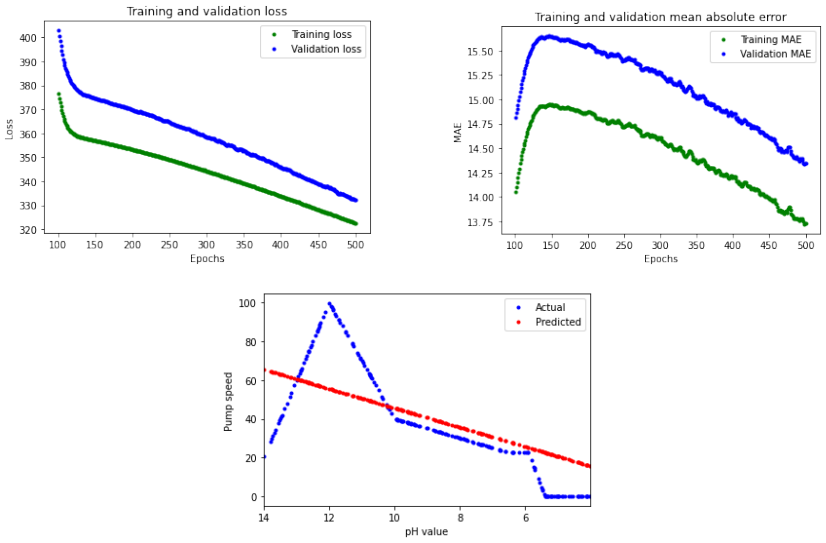

Figure 53: MSE, MAE and the output for the model with 1 hidden layer with 8 neurons, batch size = 64,adam optimiser, 500 epochs

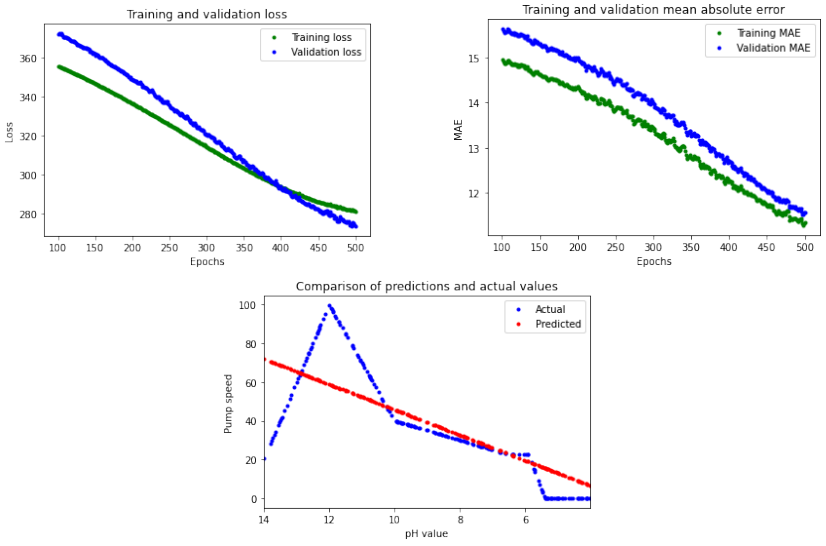

Figure 54: MSE, MAE and the output for the model with 3 hidden layers with 4,8,4 neurons respectively, batch size = 64,adam optimiser, 500 epochs

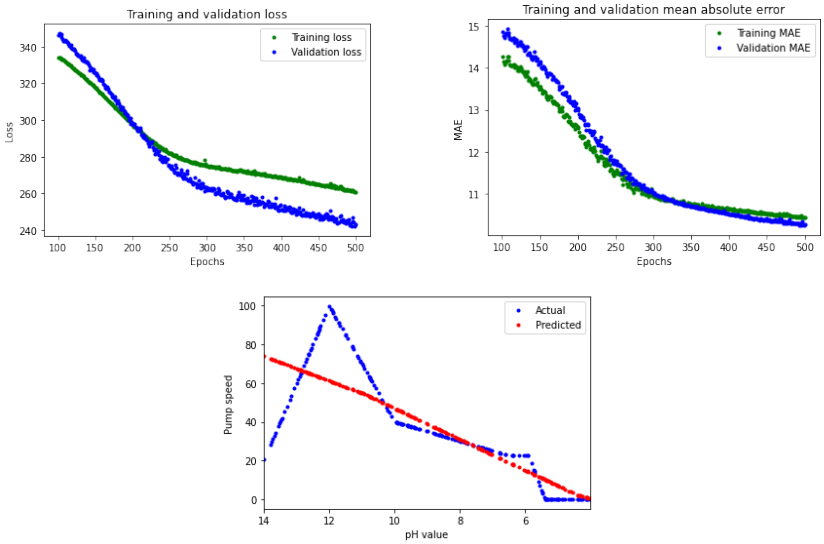

Figure 55: MSE, MAE and the output for the model with 3 hidden layers with 8,16,8 neurons respectively, batch size = 64,adam optimiser, 500 epochs

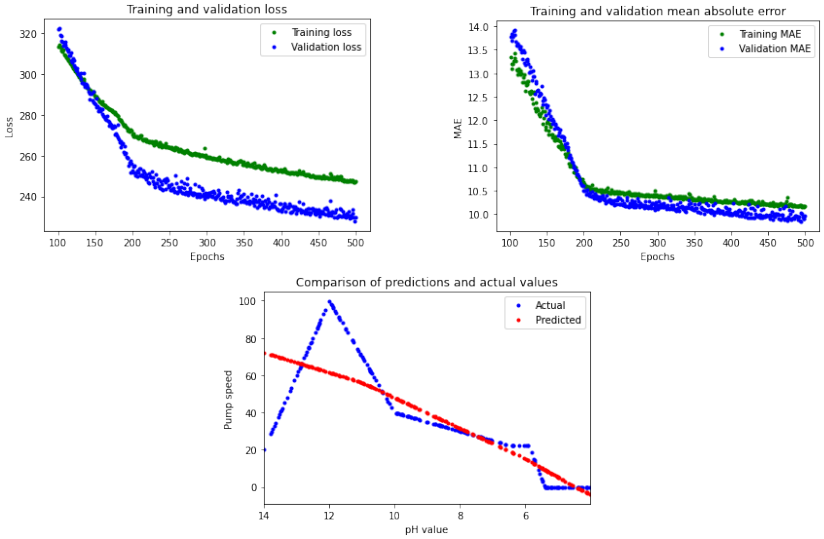

Figure 56: MSE, MAE and the output for the model with 3 hidden layers with 16,32,16 neurons respectively, batch size = 64,adam optimiser, 500 epochs

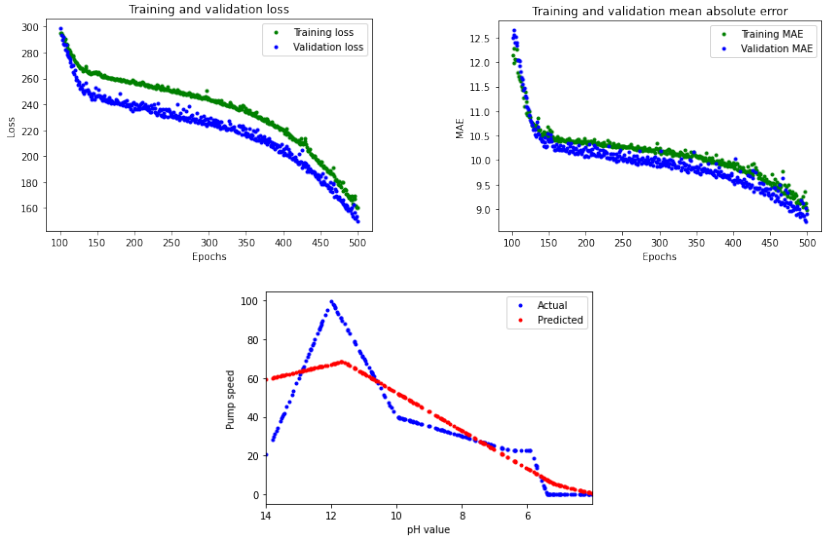

Figure 57: MSE, MAE and the output for the model with 3 hidden layers with 32,64,32 neurons respectively, batch size = 64,adam optimiser, 500 epochs

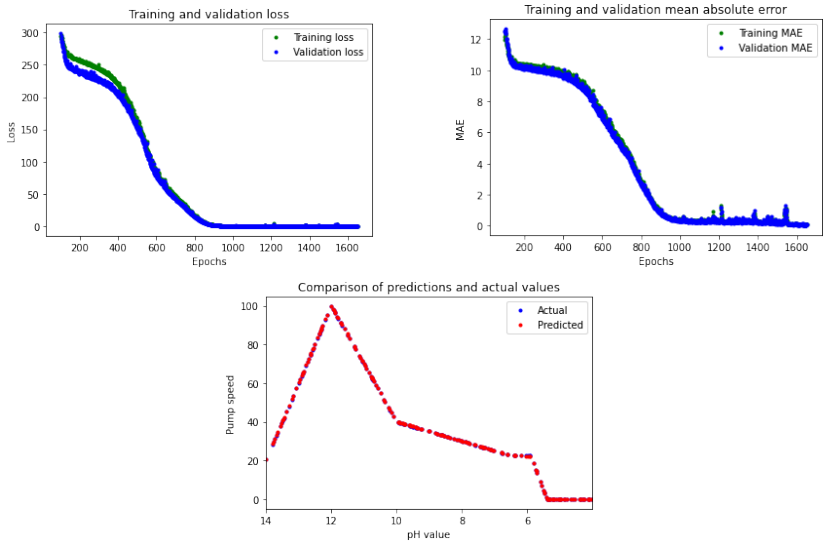

Figure 58: MSE, MAE and the output for the model with 3 hidden layers with 32,64,33 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

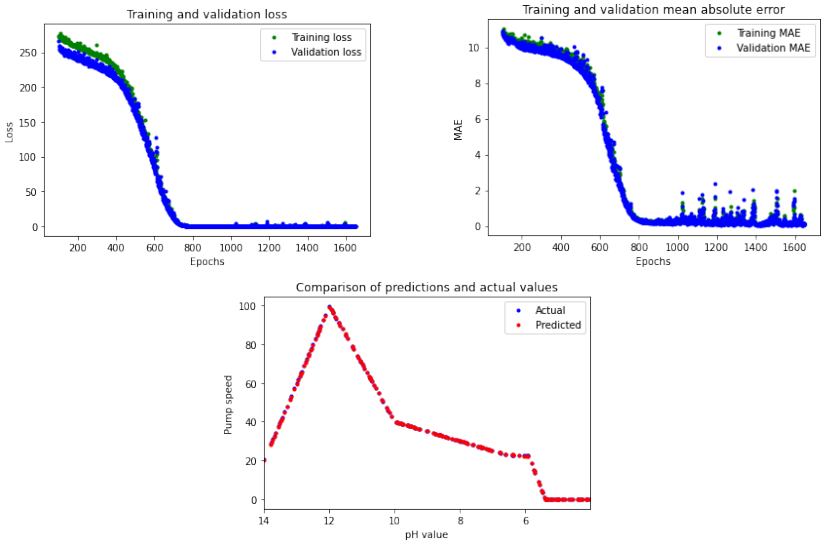

Figure 59: MSE, MAE and the output for the model with 3 hidden layers with 64,128,64 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

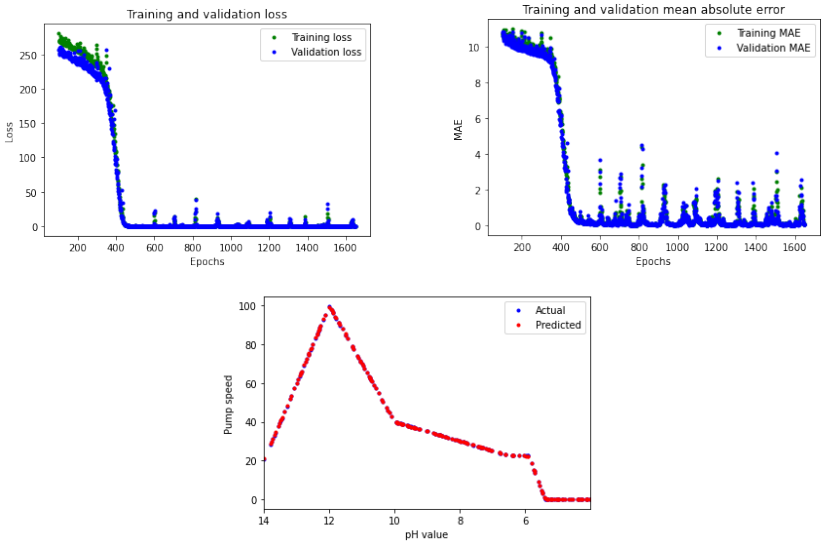

Figure 60: MSE, MAE and the output for the model with 5 hidden layers with 8,64,128,64,8 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

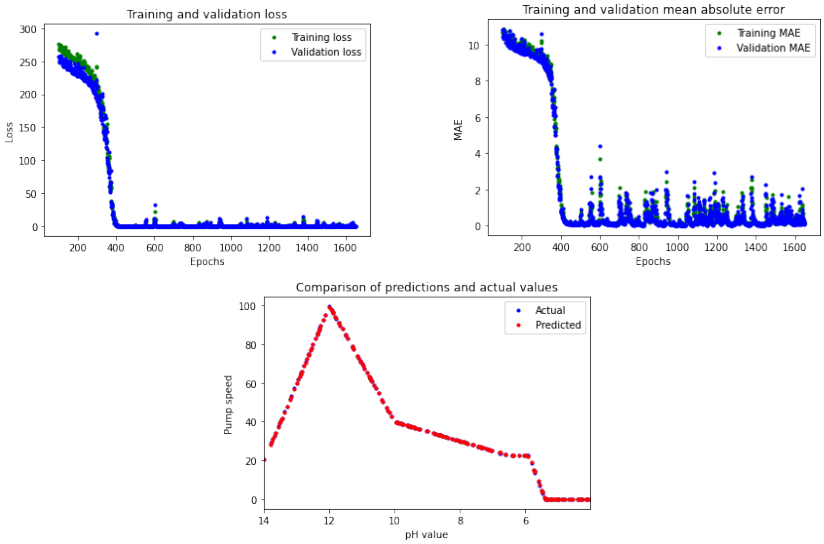

Figure 61: MSE, MAE and the output for the model with 5 hidden layers with 16,64,128,64,16 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

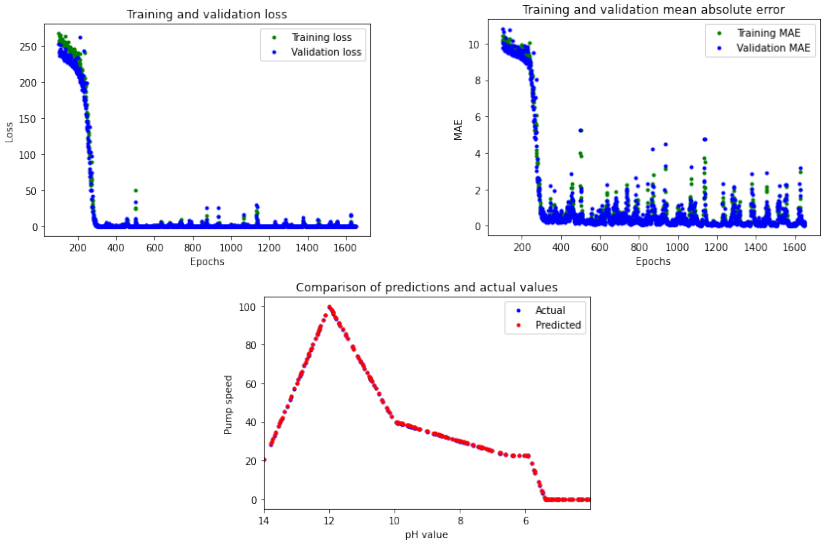

Figure 62: MSE, MAE and the output for the model with 5 hidden layers with 32,64,128,64,32 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

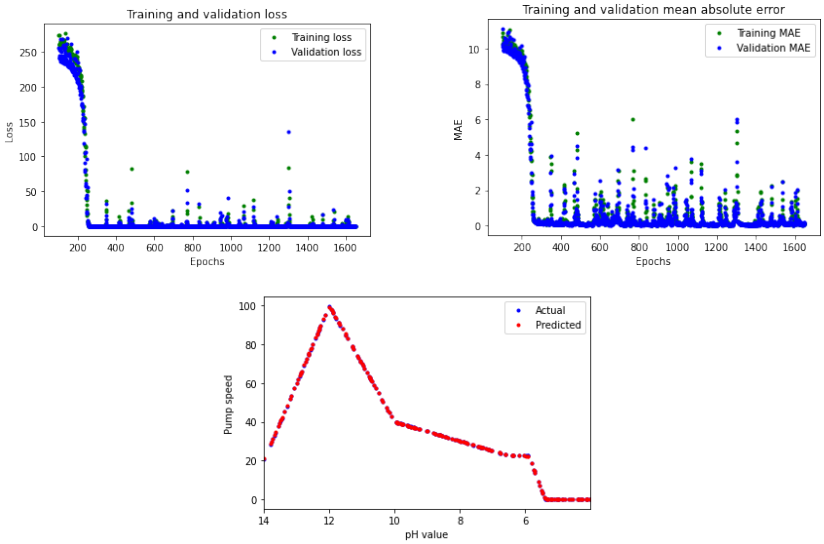

Figure 63: MSE, MAE and the output for the model with 7 hidden layers with 8,32,64,128,64,32,8 neurons respectively, batch size = 64,adam optimiser, 1650 epochs

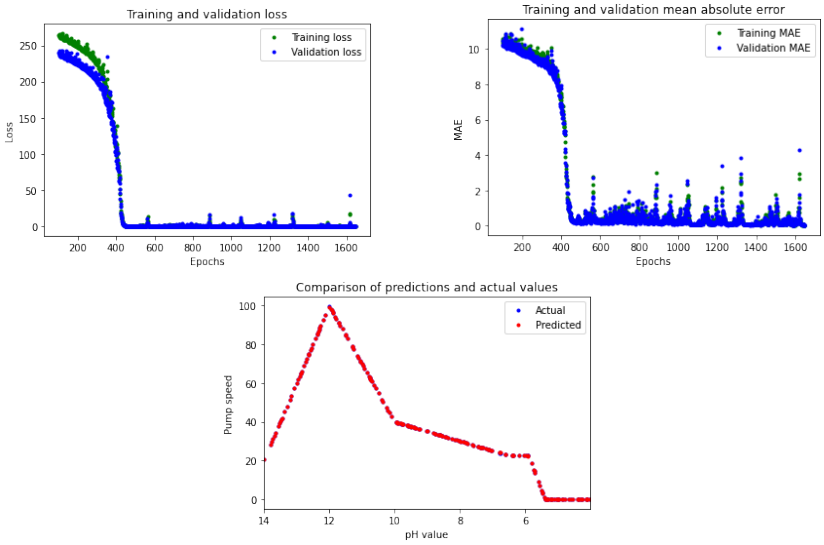

Figure 64: MSE, MAE and the output for the model with 5 hidden layers with 8,64,128,64,8 neurons respectively, batch size = 64,adam optimiser, 1639 epochs using K-fold cross validation technique

9.2 Testing the performance of AI on the real device

pH	Output pump speed	Expected pump speed	Absolute Error (pump speed)	*Time per inference (in microseconds)*
14	16.8	20	3.2	*2599*
13.99	16.8	20.4	3.6	*2606*
13.98	16.8	20.8	4	*2595*
13.97	21.1	21.2	0.1	*2613*
13.96	21.1	21.6	0.5	*2596*
13.95	21.1	22	0.9	*2606*
13.94	21.1	22.4	1.3	*2595*
13.93	21.1	22.8	1.7	*2613*
13.92	21.1	23.2	2.1	*2597*
13.91	20.71	23.6	2.89	*2614*
13.9	20.71	24	3.29	*2597*
13.89	20.71	24.4	3.69	*2632*
13.88	20.71	24.8	4.09	*2697*
13.87	20.71	25.2	4.49	*2607*
13.86	21.1	25.6	4.5	*2596*
13.85	21.1	26	4.9	*2614*
13.84	21.1	26.4	5.3	*2597*
13.83	21.1	26.8	5.7	*2607*
13.82	21.1	27.2	6.1	*2596*
13.81	21.1	27.6	6.5	*2614*
13.8	20.32	28	7.68	*2597*
13.79	20.32	28.4	8.08	*2606*
13.78	20.32	28.8	8.48	*2596*
13.77	20.32	29.2	8.88	*2614*
13.76	20.32	29.6	9.28	*2597*
13.75	26.57	30	3.43	*2618*
13.74	26.57	30.4	3.83	*2597*
13.73	26.57	30.8	4.23	*2614*
13.72	26.57	31.2	4.63	*2596*
13.71	26.57	31.6	5.03	*2606*
13.7	26.57	32	5.43	*2596*
13.69	28.53	32.4	3.87	*2614*
13.68	28.53	32.8	4.27	*2596*
13.67	28.53	33.2	4.67	*2606*
13.66	28.53	33.6	5.07	*2596*
13.65	28.53	34	5.47	*2614*
13.64	28.14	34.4	6.26	*2596*
13.63	28.14	34.8	6.66	*2606*
13.62	28.14	35.2	7.06	*2596*
13.61	28.14	35.6	7.46	*2614*
13.6	28.14	36	7.86	*2596*
13.59	28.14	36.4	8.26	*2614*
13.58	33.22	36.8	3.58	*2597*
13.57	33.22	37.2	3.98	*2612*
13.56	33.22	37.6	4.38	*2665*

13.55	33.22	38	4.78	*2606*
13.54	33.22	38.4	5.18	*2596*
13.53	34.78	38.8	4.02	*2612*
13.52	34.78	39.2	4.42	*2596*
13.51	34.78	39.6	4.82	*2606*
13.5	34.78	40	5.22	*2596*
13.49	34.78	40.4	5.62	*2612*
13.48	34.78	40.8	6.02	*2596*
13.47	35.95	41.2	5.25	*2606*
13.46	35.95	41.6	5.65	*2596*
13.45	35.95	42	6.05	*2612*
13.44	35.95	42.4	6.45	*2596*
13.43	35.95	42.8	6.85	*2616*
13.42	34.39	43.2	8.81	*2597*
13.41	34.39	43.6	9.21	*2612*
13.4	34.39	44	9.61	*2603*
13.39	34.39	44.4	10.01	*2709*
13.38	34.39	44.8	10.41	*2596*
13.37	34.39	45.2	10.81	*2612*
13.36	41.03	45.6	4.57	*2596*
13.35	41.03	46	4.97	*2606*
13.34	41.03	46.4	5.37	*2596*
13.33	41.03	46.8	5.77	*2612*
13.32	41.03	47.2	6.17	*2596*
13.31	42.2	47.6	5.4	*2605*
13.3	42.2	48	5.8	*2595*
13.29	42.2	48.4	6.2	*2612*
13.28	42.2	48.8	6.6	*2596*
13.27	42.2	49.2	7	*2615*
13.26	42.2	49.6	7.4	*2598*
13.25	42.99	50	7.01	*2613*
13.24	42.99	50.4	7.41	*2596*
13.23	42.99	50.8	7.81	*2607*
13.22	42.99	51.2	8.21	*2597*
13.21	42.99	51.6	8.61	*2613*
13.2	49.63	52	2.37	*2597*
13.19	49.63	52.4	2.77	*2607*
13.18	49.63	52.8	3.17	*2596*
13.17	49.63	53.2	3.57	*2613*
13.16	49.63	53.6	3.97	*2596*
13.15	49.63	54	4.37	*2606*
13.14	50.41	54.4	3.99	*2596*
13.13	50.41	54.8	4.39	*2613*
13.12	50.41	55.2	4.79	*2596*
13.11	50.41	55.6	5.19	*2618*
13.1	50.41	56	5.59	*2598*

13.09	51.19	56.4	5.21	*2613*
13.08	51.19	56.8	5.61	*2596*
13.07	51.19	57.2	6.01	*2671*
13.06	51.19	57.6	6.41	*2689*
13.05	51.19	58	6.81	*2613*
13.04	51.19	58.4	7.21	*2596*
13.03	49.63	58.8	9.17	*2606*
13.02	49.63	59.2	9.57	*2596*
13.01	49.63	59.6	9.97	*2613*
13	49.63	60	10.37	*2596*
12.99	49.63	60.4	10.77	*2606*
12.98	57.05	60.8	3.75	*2596*
12.97	57.05	61.2	4.15	*2613*
12.96	57.05	61.6	4.55	*2596*
12.95	57.05	62	4.95	*2617*
12.94	57.05	62.4	5.35	*2598*
12.93	57.05	62.8	5.75	*2613*
12.92	57.45	63.2	5.75	*2596*
12.91	57.45	63.6	6.15	*2606*
12.9	57.45	64	6.55	*2627*
12.89	57.45	64.4	6.95	*2613*
12.88	57.45	64.8	7.35	*2596*
12.87	59.79	65.2	5.41	*2606*
12.86	59.79	65.6	5.81	*2596*
12.85	59.79	66	6.21	*2613*
12.84	59.79	66.4	6.61	*2596*
12.83	59.79	66.8	7.01	*2606*
12.82	59.79	67.2	7.41	*2596*
12.81	63.31	67.6	4.29	*2613*
12.8	63.31	68	4.69	*2596*
12.79	63.31	68.4	5.09	*2622*
12.78	63.31	68.8	5.49	*2597*
12.77	63.31	69.2	5.89	*2613*
12.76	66.82	69.6	2.78	*2596*
12.75	66.82	70	3.18	*2606*
12.74	66.82	70.4	3.58	*2596*
12.73	66.82	70.8	3.98	*2613*
12.72	66.82	71.2	4.38	*2596*
12.71	66.82	71.6	4.78	*2606*
12.7	66.04	72	5.96	*2596*
12.69	66.04	72.4	6.36	*2613*
12.68	66.04	72.8	6.76	*2596*
12.67	66.04	73.2	7.16	*2606*
12.66	66.04	73.6	7.56	*2596*
12.65	72.3	74	1.7	*2613*
12.64	72.3	74.4	2.1	*2596*

12.63	72.3	74.8	2.5	*2618*
12.62	72.3	75.2	2.9	*2597*
12.61	72.3	75.6	3.3	*2614*
12.6	72.3	76	3.7	*2596*
12.59	70.73	76.4	5.67	*2607*
12.58	70.73	76.8	6.07	*2638*
12.57	70.73	77.2	6.47	*2714*
12.56	70.73	77.6	6.87	*2597*
12.55	70.73	78	7.27	*2607*
12.54	72.3	78.4	6.1	*2596*
12.53	72.3	78.8	6.5	*2614*
12.52	72.3	79.2	6.9	*2596*
12.51	72.3	79.6	7.3	*2607*
12.5	72.3	80	7.7	*2597*
12.49	72.3	80.4	8.1	*2614*
12.48	72.3	80.8	8.5	*2596*
12.47	72.3	81.2	8.9	*2621*
12.46	72.3	81.6	9.3	*2598*
12.45	72.3	82	9.7	*2614*
12.44	72.3	82.4	10.1	*2596*
12.43	79.72	82.8	3.08	*2606*
12.42	79.72	83.2	3.48	*2596*
12.41	79.72	83.6	3.88	*2614*
12.4	79.72	84	4.28	*2596*
12.39	79.72	84.4	4.68	*2606*
12.38	79.72	84.8	5.08	*2596*
12.37	79.72	85.2	5.48	*2614*
12.36	79.72	85.6	5.88	*2596*
12.35	79.72	86	6.28	*2606*
12.34	79.72	86.4	6.68	*2596*
12.33	79.72	86.8	7.08	*2614*
12.32	79.33	87.2	7.87	*2596*
12.31	79.33	87.6	8.27	*2618*
12.3	79.33	88	8.67	*2597*
12.29	79.33	88.4	9.07	*2614*
12.28	79.33	88.8	9.47	*2596*
12.27	85.58	89.2	3.62	*2606*
12.26	85.58	89.6	4.02	*2596*
12.25	85.58	90	4.42	*2647*
12.24	85.58	90.4	4.82	*2675*
12.23	85.58	90.8	5.22	*2606*
12.22	85.58	91.2	5.62	*2596*
12.21	86.75	91.6	4.85	*2614*
12.2	86.75	92	5.25	*2596*
12.19	86.75	92.4	5.65	*2606*
12.18	86.75	92.8	6.05	*2596*

12.17	86.75	93.2	6.45	*2614*
12.16	87.93	93.6	5.67	*2596*
12.15	87.93	94	6.07	*2619*
12.14	87.93	94.4	6.47	*2597*
12.13	87.93	94.8	6.87	*2614*
12.12	87.93	95.2	7.27	*2596*
12.11	87.93	95.6	7.67	*2606*
12.1	87.93	96	8.07	*2596*
12.09	87.93	96.4	8.47	*2614*
12.08	87.93	96.8	8.87	*2658*
12.07	87.93	97.2	9.27	*2606*
12.06	87.93	97.6	9.67	*2596*
12.05	92.62	98	5.38	*2614*
12.04	92.62	98.4	5.78	*2596*
12.03	92.62	98.8	6.18	*2606*
12.02	92.62	99.2	6.58	*2596*
12.01	92.62	99.6	6.98	*2614*
12	92.62	100	7.38	*2596*
11.99	92.23	99.7	7.47	*2618*
11.98	92.23	99.4	7.17	*2598*
11.97	92.23	99.1	6.87	*2614*
11.96	92.23	98.8	6.57	*2596*
11.95	92.23	98.5	6.27	*2606*
11.94	92.62	98.2	5.58	*2596*
11.93	92.62	97.9	5.28	*2615*
11.92	92.62	97.6	4.98	*2596*
11.91	92.62	97.3	4.68	*2606*
11.9	92.62	97	4.38	*2597*
11.89	92.62	96.7	4.08	*2615*
11.88	91.05	96.4	5.35	*2596*
11.87	91.05	96.1	5.05	*2606*
11.86	91.05	95.8	4.75	*2596*
11.85	91.05	95.5	4.45	*2614*
11.84	91.05	95.2	4.15	*2596*
11.83	90.66	94.9	4.24	*2622*
11.82	90.66	94.6	3.94	*2598*
11.81	90.66	94.3	3.64	*2614*
11.8	90.66	94	3.34	*2596*
11.79	90.66	93.7	3.04	*2606*
11.78	90.66	93.4	2.74	*2596*
11.77	89.49	93.1	3.61	*2614*
11.76	89.49	92.8	3.31	*2625*
11.75	89.49	92.5	3.01	*2678*
11.74	89.49	92.2	2.71	*2596*
11.73	89.49	91.9	2.41	*2612*
11.72	88.32	91.6	3.28	*2596*

11.71	88.32	91.3	2.98	*2606*
11.7	88.32	91	2.68	*2596*
11.69	88.32	90.7	2.38	*2612*
11.68	88.32	90.4	2.08	*2596*
11.67	88.32	90.1	1.78	*2620*
11.66	84.02	89.8	5.78	*2598*
11.65	84.02	89.5	5.48	*2612*
11.64	84.02	89.2	5.18	*2596*
11.63	84.02	88.9	4.88	*2606*
11.62	84.02	88.6	4.58	*2594*
11.61	83.63	88.3	4.67	*2612*
11.6	83.63	88	4.37	*2596*
11.59	83.63	87.7	4.07	*2606*
11.58	83.63	87.4	3.77	*2594*
11.57	83.63	87.1	3.47	*2611*
11.56	83.63	86.8	3.17	*2596*
11.55	82.85	86.5	3.65	*2606*
11.54	82.85	86.2	3.35	*2594*
11.53	82.85	85.9	3.05	*2612*
11.52	82.85	85.6	2.75	*2596*
11.51	82.85	85.3	2.45	*2621*
11.5	79.33	85	5.67	*2594*
11.49	79.33	84.7	5.37	*2612*
11.48	79.33	84.4	5.07	*2596*
11.47	79.33	84.1	4.77	*2606*
11.46	79.33	83.8	4.47	*2594*
11.45	79.33	83.5	4.17	*2612*
11.44	77.38	83.2	5.82	*2596*
11.43	77.38	82.9	5.52	*2664*
11.42	77.38	82.6	5.22	*2594*
11.41	77.38	82.3	4.92	*2612*
11.4	77.38	82	4.62	*2596*
11.39	78.94	81.7	2.76	*2606*
11.38	78.94	81.4	2.46	*2594*
11.37	78.94	81.1	2.16	*2611*
11.36	78.94	80.8	1.86	*2596*
11.35	78.94	80.5	1.56	*2620*
11.34	78.94	80.2	1.26	*2595*
11.33	73.47	79.9	6.43	*2613*
11.32	73.47	79.6	6.13	*2597*
11.31	73.47	79.3	5.83	*2606*
11.3	73.47	79	5.53	*2595*
11.29	73.47	78.7	5.23	*2613*
11.28	73.47	78.4	4.93	*2596*
11.27	73.47	78.1	4.63	*2615*
11.26	73.47	77.8	4.33	*2709*

11.25	73.47	77.5	4.03	*2613*
11.24	73.47	77.2	3.73	*2597*
11.23	73.47	76.9	3.43	*2607*
11.22	73.08	76.6	3.52	*2595*
11.21	73.08	76.3	3.22	*2613*
11.2	73.08	76	2.92	*2597*
11.19	73.08	75.7	2.62	*2622*
11.18	73.08	75.4	2.32	*2595*
11.17	72.69	75.1	2.41	*2613*
11.16	72.69	74.8	2.11	*2596*
11.15	72.69	74.5	1.81	*2606*
11.14	72.69	74.2	1.51	*2595*
11.13	72.69	73.9	1.21	*2613*
11.12	72.69	73.6	0.91	*2596*
11.11	68	73.3	5.3	*2606*
11.1	68	73	5	*2595*
11.09	68	72.7	4.7	*2613*
11.08	68	72.4	4.4	*2596*
11.07	68	72.1	4.1	*2606*
11.06	68	71.8	3.8	*2595*
11.05	68	71.5	3.5	*2612*
11.04	68	71.2	3.2	*2596*
11.03	68	70.9	2.9	*2621*
11.02	68	70.6	2.6	*2595*
11.01	68	70.3	2.3	*2612*
11	67.61	70	2.39	*2596*
10.99	67.61	69.7	2.09	*2606*
10.98	67.61	69.4	1.79	*2595*
10.97	67.61	69.1	1.49	*2613*
10.96	67.61	68.8	1.19	*2596*
10.95	61.35	68.5	7.15	*2606*
10.94	61.35	68.2	6.85	*2655*
10.93	61.35	67.9	6.55	*2703*
10.92	61.35	67.6	6.25	*2596*
10.91	61.35	67.3	5.95	*2606*
10.9	61.35	67	5.65	*2595*
10.89	60.18	66.7	6.52	*2613*
10.88	60.18	66.4	6.22	*2596*
10.87	60.18	66.1	5.92	*2622*
10.86	60.18	65.8	5.62	*2595*
10.85	60.18	65.5	5.32	*2613*
10.84	60.57	65.2	4.63	*2596*
10.83	60.57	64.9	4.33	*2606*
10.82	60.57	64.6	4.03	*2595*
10.81	60.57	64.3	3.73	*2613*
10.8	60.57	64	3.43	*2596*

10.79	60.57	63.7	3.13	*2606*
10.78	59.4	63.4	4	*2595*
10.77	59.4	63.1	3.7	*2659*
10.76	59.4	62.8	3.4	*2596*
10.75	59.4	62.5	3.1	*2606*
10.74	59.4	62.2	2.8	*2595*
10.73	56.27	61.9	5.63	*2613*
10.72	56.27	61.6	5.33	*2596*
10.71	56.27	61.3	5.03	*2621*
10.7	56.27	61	4.73	*2595*
10.69	56.27	60.7	4.43	*2614*
10.68	56.27	60.4	4.13	*2597*
10.67	55.88	60.1	4.22	*2607*
10.66	55.88	59.8	3.92	*2596*
10.65	55.88	59.5	3.62	*2614*
10.64	55.88	59.2	3.32	*2596*
10.63	55.88	58.9	3.02	*2607*
10.62	54.32	58.6	4.28	*2596*
10.61	54.32	58.3	3.98	*2614*
10.6	54.32	58	3.68	*2596*
10.59	54.32	57.7	3.38	*2606*
10.58	54.32	57.4	3.08	*2596*
10.57	54.32	57.1	2.78	*2614*
10.56	51.19	56.8	5.61	*2596*
10.55	51.19	56.5	5.31	*2623*
10.54	51.19	56.2	5.01	*2596*
10.53	51.19	55.9	4.71	*2614*
10.52	51.19	55.6	4.41	*2596*
10.51	50.41	55.3	4.89	*2606*
10.5	50.41	55	4.59	*2596*
10.49	50.41	54.7	4.29	*2613*
10.48	50.41	54.4	3.99	*2596*
10.47	50.41	54.1	3.69	*2606*
10.46	50.41	53.8	3.39	*2596*
10.45	49.24	53.5	4.26	*2651*
10.44	49.24	53.2	3.96	*2711*
10.43	49.24	52.9	3.66	*2606*
10.42	49.24	52.6	3.36	*2596*
10.41	49.24	52.3	3.06	*2613*
10.4	49.24	52	2.76	*2596*
10.39	49.24	51.7	2.46	*2622*
10.38	49.24	51.4	2.16	*2596*
10.37	49.24	51.1	1.86	*2614*
10.36	49.24	50.8	1.56	*2596*
10.35	49.24	50.5	1.26	*2606*
10.34	44.94	50.2	5.26	*2596*

10.33	44.94	49.9	4.96	*2614*
10.32	44.94	49.6	4.66	*2596*
10.31	44.94	49.3	4.36	*2606*
10.3	44.94	49	4.06	*2596*
10.29	42.99	48.7	5.71	*2614*
10.28	42.99	48.4	5.41	*2596*
10.27	42.99	48.1	5.11	*2606*
10.26	42.99	47.8	4.81	*2596*
10.25	42.99	47.5	4.51	*2614*
10.24	42.99	47.2	4.21	*2596*
10.23	43.38	46.9	3.52	*2622*
10.22	43.38	46.6	3.22	*2596*
10.21	43.38	46.3	2.92	*2614*
10.2	43.38	46	2.62	*2596*
10.19	43.38	45.7	2.32	*2606*
10.18	39.47	45.4	5.93	*2596*
10.17	39.47	45.1	5.63	*2614*
10.16	39.47	44.8	5.33	*2596*
10.15	39.47	44.5	5.03	*2606*
10.14	39.47	44.2	4.73	*2596*
10.13	39.47	43.9	4.43	*2614*
10.12	39.47	43.6	4.13	*2630*
10.11	39.47	43.3	3.83	*2680*
10.1	39.47	43	3.53	*2596*
10.09	39.47	42.7	3.23	*2614*
10.08	39.47	42.4	2.93	*2597*
10.07	38.3	42.1	3.8	*2621*
10.06	38.3	41.8	3.5	*2597*
10.05	38.3	41.5	3.2	*2615*
10.04	38.3	41.2	2.9	*2597*
10.03	38.3	40.9	2.6	*2607*
10.02	38.3	40.6	2.3	*2596*
10.01	37.12	40.3	3.18	*2615*
10	37.12	40	2.88	*2597*
9.99	37.12	39.95	2.83	*2607*
9.98	37.12	39.9	2.78	*2597*
9.97	37.12	39.85	2.73	*2615*
9.96	36.73	39.8	3.07	*2596*
9.95	36.73	39.75	3.02	*2688*
9.94	36.73	39.7	2.97	*2596*
9.93	36.73	39.65	2.92	*2615*
9.92	36.73	39.6	2.87	*2596*
9.91	36.73	39.55	2.82	*2622*
9.9	36.34	39.5	3.16	*2596*
9.89	36.34	39.45	3.11	*2615*
9.88	36.34	39.4	3.06	*2596*

9.87	36.34	39.35	3.01	*2606*
9.86	36.34	39.3	2.96	*2596*
9.85	36.34	39.25	2.91	*2614*
9.84	36.34	39.2	2.86	*2596*
9.83	36.34	39.15	2.81	*2606*
9.82	36.34	39.1	2.76	*2596*
9.81	36.34	39.05	2.71	*2614*
9.8	36.34	39	2.66	*2596*
9.79	35.95	38.95	3	*2606*
9.78	35.95	38.9	2.95	*2596*
9.77	35.95	38.85	2.9	*2614*
9.76	35.95	38.8	2.85	*2596*
9.75	35.95	38.75	2.8	*2621*
9.74	35.56	38.7	3.14	*2596*
9.73	35.56	38.65	3.09	*2614*
9.72	35.56	38.6	3.04	*2596*
9.71	35.56	38.55	2.99	*2606*
9.7	35.56	38.5	2.94	*2596*
9.69	35.56	38.45	2.89	*2614*
9.68	35.56	38.4	2.84	*2596*
9.67	35.56	38.35	2.79	*2606*
9.66	35.56	38.3	2.74	*2596*
9.65	35.56	38.25	2.69	*2612*
9.64	35.56	38.2	2.64	*2596*
9.63	35.17	38.15	2.98	*2652*
9.62	35.17	38.1	2.93	*2675*
9.61	35.17	38.05	2.88	*2612*
9.6	35.17	38	2.83	*2596*
9.59	35.17	37.95	2.78	*2623*
9.58	34.78	37.9	3.12	*2596*
9.57	34.78	37.85	3.07	*2612*
9.56	34.78	37.8	3.02	*2596*
9.55	34.78	37.75	2.97	*2606*
9.54	34.78	37.7	2.92	*2596*
9.53	34.78	37.65	2.87	*2612*
9.52	34.39	37.6	3.21	*2596*
9.51	34.39	37.55	3.16	*2606*
9.5	34.39	37.5	3.11	*2596*
9.49	34.39	37.45	3.06	*2612*
9.48	34.39	37.4	3.01	*2596*
9.47	34.39	37.35	2.96	*2606*
9.46	34.39	37.3	2.91	*2596*
9.45	34.39	37.25	2.86	*2612*
9.44	34.39	37.2	2.81	*2597*
9.43	34.39	37.15	2.76	*2621*
9.42	34.39	37.1	2.71	*2596*

9.41	34.39	37.05	2.66	*2613*
9.4	34.39	37	2.61	*2597*
9.39	34.39	36.95	2.56	*2607*
9.38	34.39	36.9	2.51	*2597*
9.37	34.39	36.85	2.46	*2613*
9.36	33.61	36.8	3.19	*2597*
9.35	33.61	36.75	3.14	*2607*
9.34	33.61	36.7	3.09	*2596*
9.33	33.61	36.65	3.04	*2613*
9.32	33.61	36.6	2.99	*2596*
9.31	33.61	36.55	2.94	*2607*
9.3	34.39	36.5	2.11	*2645*
9.29	34.39	36.45	2.06	*2613*
9.28	34.39	36.4	2.01	*2596*
9.27	34.39	36.35	1.96	*2623*
9.26	34.39	36.3	1.91	*2596*
9.25	33.22	36.25	3.03	*2613*
9.24	33.22	36.2	2.98	*2596*
9.23	33.22	36.15	2.93	*2606*
9.22	33.22	36.1	2.88	*2596*
9.21	33.22	36.05	2.83	*2613*
9.2	33.22	36	2.78	*2596*
9.19	33.22	35.95	2.73	*2607*
9.18	33.22	35.9	2.68	*2596*
9.17	33.22	35.85	2.63	*2613*
9.16	33.22	35.8	2.58	*2596*
9.15	33.22	35.75	2.53	*2606*
9.14	33.22	35.7	2.48	*2615*
9.13	33.22	35.65	2.43	*2682*
9.12	33.22	35.6	2.38	*2596*
9.11	33.22	35.55	2.33	*2621*
9.1	33.22	35.5	2.28	*2596*
9.09	33.22	35.45	2.23	*2613*
9.08	32.83	35.4	2.57	*2596*
9.07	32.83	35.35	2.52	*2606*
9.06	32.83	35.3	2.47	*2596*
9.05	32.83	35.25	2.42	*2613*
9.04	32.83	35.2	2.37	*2596*
9.03	32.44	35.15	2.71	*2606*
9.02	32.44	35.1	2.66	*2596*
9.01	32.44	35.05	2.61	*2613*
9	32.44	35	2.56	*2596*
8.99	32.44	34.95	2.51	*2606*
8.98	32.44	34.9	2.46	*2596*
8.97	32.04	34.85	2.81	*2613*
8.96	32.04	34.8	2.76	*2596*

8.95	32.04	34.75	2.71	*2624*
8.94	32.04	34.7	2.66	*2596*
8.93	32.04	34.65	2.61	*2613*
8.92	32.04	34.6	2.56	*2596*
8.91	32.04	34.55	2.51	*2606*
8.9	32.04	34.5	2.46	*2596*
8.89	32.04	34.45	2.41	*2613*
8.88	32.04	34.4	2.36	*2596*
8.87	32.04	34.35	2.31	*2606*
8.86	31.26	34.3	3.04	*2596*
8.85	31.26	34.25	2.99	*2613*
8.84	31.26	34.2	2.94	*2596*
8.83	31.26	34.15	2.89	*2606*
8.82	31.26	34.1	2.84	*2596*
8.81	31.65	34.05	2.4	*2650*
8.8	31.65	34	2.35	*2692*
8.79	31.65	33.95	2.3	*2622*
8.78	31.65	33.9	2.25	*2596*
8.77	31.65	33.85	2.2	*2614*
8.76	31.65	33.8	2.15	*2597*
8.75	31.26	33.75	2.49	*2607*
8.74	31.26	33.7	2.44	*2596*
8.73	31.26	33.65	2.39	*2614*
8.72	31.26	33.6	2.34	*2596*
8.71	31.26	33.55	2.29	*2606*
8.7	30.48	33.5	3.02	*2597*
8.69	30.48	33.45	2.97	*2613*
8.68	30.48	33.4	2.92	*2596*
8.67	30.48	33.35	2.87	*2606*
8.66	30.48	33.3	2.82	*2596*
8.65	30.48	33.25	2.77	*2613*
8.64	30.48	33.2	2.72	*2635*
8.63	30.48	33.15	2.67	*2623*
8.62	30.48	33.1	2.62	*2597*
8.61	30.48	33.05	2.57	*2614*
8.6	30.48	33	2.52	*2596*
8.59	30.48	32.95	2.47	*2606*
8.58	30.48	32.9	2.42	*2594*
8.57	30.48	32.85	2.37	*2613*
8.56	30.48	32.8	2.32	*2596*
8.55	30.48	32.75	2.27	*2606*
8.54	30.48	32.7	2.22	*2594*
8.53	30.09	32.65	2.56	*2613*
8.52	30.09	32.6	2.51	*2596*
8.51	30.09	32.55	2.46	*2606*
8.5	30.09	32.5	2.41	*2594*

8.49	30.09	32.45	2.36	*2613*
8.48	30.09	32.4	2.31	*2596*
8.47	30.09	32.35	2.26	*2620*
8.46	30.09	32.3	2.21	*2596*
8.45	30.09	32.25	2.16	*2613*
8.44	30.09	32.2	2.11	*2596*
8.43	30.09	32.15	2.06	*2606*
8.42	29.31	32.1	2.79	*2594*
8.41	29.31	32.05	2.74	*2613*
8.4	29.31	32	2.69	*2596*
8.39	29.31	31.95	2.64	*2606*
8.38	29.31	31.9	2.59	*2594*
8.37	29.7	31.85	2.15	*2613*
8.36	29.7	31.8	2.1	*2596*
8.35	29.7	31.75	2.05	*2606*
8.34	29.7	31.7	2	*2594*
8.33	29.7	31.65	1.95	*2613*
8.32	29.7	31.6	1.9	*2636*
8.31	29.7	31.55	1.85	*2710*
8.3	29.7	31.5	1.8	*2611*
8.29	29.7	31.45	1.75	*2613*
8.28	29.7	31.4	1.7	*2596*
8.27	29.7	31.35	1.65	*2606*
8.26	28.53	31.3	2.77	*2594*
8.25	28.53	31.25	2.72	*2613*
8.24	28.53	31.2	2.67	*2596*
8.23	28.53	31.15	2.62	*2606*
8.22	28.53	31.1	2.57	*2594*
8.21	28.53	31.05	2.52	*2613*
8.2	28.14	31	2.86	*2596*
8.19	28.14	30.95	2.81	*2606*
8.18	28.14	30.9	2.76	*2594*
8.17	28.14	30.85	2.71	*2613*
8.16	28.14	30.8	2.66	*2597*
8.15	28.53	30.75	2.22	*2611*
8.14	28.53	30.7	2.17	*2609*
8.13	28.53	30.65	2.12	*2614*
8.12	28.53	30.6	2.07	*2597*
8.11	28.53	30.55	2.02	*2607*
8.1	28.53	30.5	1.97	*2595*
8.09	28.14	30.45	2.31	*2614*
8.08	28.14	30.4	2.26	*2597*
8.07	28.14	30.35	2.21	*2607*
8.06	28.14	30.3	2.16	*2595*
8.05	28.14	30.25	2.11	*2614*
8.04	27.75	30.2	2.45	*2596*

8.03	27.75	30.15	2.4	*2606*
8.02	27.75	30.1	2.35	*2595*
8.01	27.75	30.05	2.3	*2614*
8	27.75	30	2.25	*2596*
7.99	27.75	29.95	2.2	*2648*
7.98	27.36	29.9	2.54	*2660*
7.97	27.36	29.85	2.49	*2614*
7.96	27.36	29.8	2.44	*2596*
7.95	27.36	29.75	2.39	*2606*
7.94	27.36	29.7	2.34	*2595*
7.93	26.96	29.65	2.69	*2614*
7.92	26.96	29.6	2.64	*2596*
7.91	26.96	29.55	2.59	*2606*
7.9	26.96	29.5	2.54	*2595*
7.89	26.96	29.45	2.49	*2614*
7.88	26.96	29.4	2.44	*2596*
7.87	26.57	29.35	2.78	*2606*
7.86	26.57	29.3	2.73	*2595*
7.85	26.57	29.25	2.68	*2614*
7.84	26.57	29.2	2.63	*2596*
7.83	26.57	29.15	2.58	*2610*
7.82	26.57	29.1	2.53	*2661*
7.81	26.57	29.05	2.48	*2614*
7.8	26.57	29	2.43	*2596*
7.79	26.57	28.95	2.38	*2606*
7.78	26.57	28.9	2.33	*2595*
7.77	26.57	28.85	2.28	*2614*
7.76	26.18	28.8	2.62	*2596*
7.75	26.18	28.75	2.57	*2606*
7.74	26.18	28.7	2.52	*2595*
7.73	26.18	28.65	2.47	*2613*
7.72	26.18	28.6	2.42	*2596*
7.71	26.18	28.55	2.37	*2606*
7.7	26.18	28.5	2.32	*2595*
7.69	26.18	28.45	2.27	*2614*
7.68	26.18	28.4	2.22	*2596*
7.67	26.18	28.35	2.17	*2608*
7.66	26.18	28.3	2.12	*2600*
7.65	25.79	28.25	2.46	*2614*
7.64	25.79	28.2	2.41	*2596*
7.63	25.79	28.15	2.36	*2606*
7.62	25.79	28.1	2.31	*2595*
7.61	25.79	28.05	2.26	*2614*
7.6	25.79	28	2.21	*2596*
7.59	25.79	27.95	2.16	*2606*
7.58	25.79	27.9	2.11	*2595*

7.57	25.79	27.85	2.06	*2614*
7.56	25.79	27.8	2.01	*2596*
7.55	25.79	27.75	1.96	*2606*
7.54	25.79	27.7	1.91	*2595*
7.53	25.79	27.65	1.86	*2611*
7.52	25.79	27.6	1.81	*2597*
7.51	25.79	27.55	1.76	*2624*
7.5	25.79	27.5	1.71	*2649*
7.49	25.4	27.45	2.05	*2710*
7.48	25.4	27.4	2	*2597*
7.47	25.4	27.35	1.95	*2607*
7.46	25.4	27.3	1.9	*2596*
7.45	25.4	27.25	1.85	*2612*
7.44	25.4	27.2	1.8	*2596*
7.43	24.62	27.15	2.53	*2607*
7.42	24.62	27.1	2.48	*2596*
7.41	24.62	27.05	2.43	*2612*
7.4	24.62	27	2.38	*2596*
7.39	24.62	26.95	2.33	*2607*
7.38	24.62	26.9	2.28	*2596*
7.37	24.62	26.85	2.23	*2612*
7.36	24.62	26.8	2.18	*2596*
7.35	24.62	26.75	2.13	*2630*
7.34	24.62	26.7	2.08	*2598*
7.33	24.62	26.65	2.03	*2612*
7.32	24.62	26.6	1.98	*2596*
7.31	24.62	26.55	1.93	*2606*
7.3	24.62	26.5	1.88	*2596*
7.29	24.62	26.45	1.83	*2612*
7.28	24.62	26.4	1.78	*2596*
7.27	23.45	26.35	2.9	*2606*
7.26	23.45	26.3	2.85	*2596*
7.25	23.45	26.25	2.8	*2612*
7.24	23.45	26.2	2.75	*2596*
7.23	23.45	26.15	2.7	*2606*
7.22	23.45	26.1	2.65	*2596*
7.21	23.84	26.05	2.21	*2612*
7.2	23.84	26	2.16	*2596*
7.19	23.84	25.95	2.11	*2640*
7.18	23.84	25.9	2.06	*2604*
7.17	23.84	25.85	2.01	*2679*
7.16	23.45	25.8	2.35	*2596*
7.15	23.45	25.75	2.3	*2606*
7.14	23.45	25.7	2.25	*2596*
7.13	23.45	25.65	2.2	*2612*
7.12	23.45	25.6	2.15	*2596*

7.11	23.45	25.55	2.1	*2606*
7.1	23.45	25.5	2.05	*2596*
7.09	23.45	25.45	2	*2612*
7.08	23.45	25.4	1.95	*2596*
7.07	23.45	25.35	1.9	*2606*
7.06	23.45	25.3	1.85	*2596*
7.05	23.45	25.25	1.8	*2612*
7.04	23.45	25.2	1.75	*2596*
7.03	23.45	25.15	1.7	*2626*
7.02	23.45	25.1	1.65	*2599*
7.01	23.45	25.05	1.6	*2648*
7	23.45	25	1.55	*2682*
6.99	23.45	24.95	1.5	*2606*
6.98	23.45	24.9	1.45	*2596*
6.97	23.45	24.85	1.4	*2612*
6.96	23.45	24.8	1.35	*2596*
6.95	23.45	24.75	1.3	*2606*
6.94	22.67	24.7	2.03	*2596*
6.93	22.67	24.65	1.98	*2612*
6.92	22.67	24.6	1.93	*2596*
6.91	22.67	24.55	1.88	*2606*
6.9	22.67	24.5	1.83	*2596*
6.89	22.27	24.45	2.18	*2612*
6.88	22.27	24.4	2.13	*2596*
6.87	22.27	24.35	2.08	*2624*
6.86	22.27	24.3	2.03	*2598*
6.85	22.27	24.25	1.98	*2613*
6.84	22.27	24.2	1.93	*2597*
6.83	22.27	24.15	1.88	*2607*
6.82	22.27	24.1	1.83	*2597*
6.81	22.27	24.05	1.78	*2613*
6.8	22.27	24	1.73	*2597*
6.79	22.27	23.95	1.68	*2607*
6.78	22.27	23.9	1.63	*2596*
6.77	22.27	23.85	1.58	*2613*
6.76	22.27	23.8	1.53	*2597*
6.75	22.27	23.75	1.48	*2606*
6.74	22.27	23.7	1.43	*2596*
6.73	22.27	23.65	1.38	*2613*
6.72	21.88	23.6	1.72	*2596*
6.71	21.88	23.55	1.67	*2624*
6.7	21.88	23.5	1.62	*2599*
6.69	21.88	23.45	1.57	*2613*
6.68	21.88	23.4	1.52	*2633*
6.67	21.49	23.35	1.86	*2728*
6.66	21.49	23.3	1.81	*2596*

6.65	21.49	23.25	1.76	*2613*
6.64	21.49	23.2	1.71	*2597*
6.63	21.49	23.15	1.66	*2606*
6.62	21.49	23.1	1.61	*2596*
6.61	22.27	23.05	0.78	*2613*
6.6	22.27	23	0.73	*2597*
6.59	22.27	22.95	0.68	*2606*
6.58	22.27	22.9	0.63	*2596*
6.57	22.27	22.85	0.58	*2613*
6.56	21.49	22.8	1.31	*2597*
6.55	21.49	22.75	1.26	*2631*
6.54	21.49	22.7	1.21	*2599*
6.53	21.49	22.65	1.16	*2613*
6.52	21.49	22.6	1.11	*2597*
6.51	21.49	22.55	1.06	*2643*
6.5	21.1	22.5	1.4	*2596*
6.49	21.1	22.5	1.4	*2613*
6.48	21.1	22.5	1.4	*2596*
6.47	21.1	22.5	1.4	*2606*
6.46	21.1	22.5	1.4	*2596*
6.45	20.71	22.5	1.79	*2613*
6.44	20.71	22.5	1.79	*2596*
6.43	20.71	22.5	1.79	*2606*
6.42	20.71	22.5	1.79	*2596*
6.41	20.71	22.5	1.79	*2613*
6.4	20.71	22.5	1.79	*2596*
6.39	21.49	22.5	1.01	*2626*
6.38	21.49	22.5	1.01	*2599*
6.37	21.49	22.5	1.01	*2613*
6.36	21.49	22.5	1.01	*2596*
6.35	21.49	22.5	1.01	*2606*
6.34	21.49	22.5	1.01	*2596*
6.33	21.49	22.5	1.01	*2613*
6.32	21.49	22.5	1.01	*2596*
6.31	21.49	22.5	1.01	*2606*
6.3	21.49	22.5	1.01	*2596*
6.29	21.49	22.5	1.01	*2613*
6.28	20.71	22.5	1.79	*2596*
6.27	20.71	22.5	1.79	*2606*
6.26	20.71	22.5	1.79	*2596*
6.25	20.71	22.5	1.79	*2613*
6.24	20.71	22.5	1.79	*2597*
6.23	21.49	22.5	1.01	*2625*
6.22	21.49	22.5	1.01	*2600*
6.21	21.49	22.5	1.01	*2614*
6.2	21.49	22.5	1.01	*2597*

6.19	21.49	22.5	1.01	*2654*
6.18	21.49	22.5	1.01	*2702*
6.17	21.49	22.5	1.01	*2614*
6.16	21.49	22.5	1.01	*2597*
6.15	21.49	22.5	1.01	*2606*
6.14	21.49	22.5	1.01	*2597*
6.13	21.49	22.5	1.01	*2614*
6.12	21.49	22.5	1.01	*2596*
6.11	21.49	22.5	1.01	*2606*
6.1	21.49	22.5	1.01	*2597*
6.09	21.49	22.5	1.01	*2614*
6.08	21.49	22.5	1.01	*2596*
6.07	21.49	22.5	1.01	*2624*
6.06	21.1	22.5	1.4	*2599*
6.05	21.1	22.5	1.4	*2614*
6.04	21.1	22.5	1.4	*2596*
6.03	21.1	22.5	1.4	*2607*
6.02	21.1	22.5	1.4	*2596*
6.01	20.71	22.5	1.79	*2614*
6	20.71	22.5	1.79	*2596*
5.99	20.71	22.5	1.79	*2606*
5.98	20.71	22.5	1.79	*2596*
5.97	20.71	22.5	1.79	*2614*
5.96	20.71	22.5	1.79	*2596*
5.95	20.71	22.5	1.79	*2607*
5.94	20.71	22.5	1.79	*2596*
5.93	20.71	22.5	1.79	*2614*
5.92	20.71	22.5	1.79	*2596*
5.91	20.71	22.5	1.79	*2626*
5.9	19.93	22.5	2.57	*2599*
5.89	19.93	22.05	2.12	*2612*
5.88	19.93	21.6	1.67	*2596*
5.87	19.93	21.15	1.22	*2606*
5.86	19.93	20.7	0.77	*2639*
5.85	19.93	20.25	0.32	*2656*
5.84	16.8	19.8	3	*2596*
5.83	16.8	19.35	2.55	*2606*
5.82	16.8	18.9	2.1	*2596*
5.81	16.8	18.45	1.65	*2612*
5.8	16.8	18	1.2	*2596*
5.79	16.02	17.55	1.53	*2606*
5.78	16.02	17.1	1.08	*2596*
5.77	16.02	16.65	0.63	*2612*
5.76	16.02	16.2	0.18	*2596*
5.75	16.02	15.75	0.27	*2631*
5.74	16.02	15.3	0.72	*2599*

5.73	13.29	14.85	1.56	*2612*
5.72	13.29	14.4	1.11	*2596*
5.71	13.29	13.95	0.66	*2606*
5.7	13.29	13.5	0.21	*2596*
5.69	13.29	13.05	0.24	*2693*
5.68	10.55	12.6	2.05	*2596*
5.67	10.55	12.15	1.6	*2606*
5.66	10.55	11.7	1.15	*2596*
5.65	10.55	11.25	0.7	*2612*
5.64	10.55	10.8	0.25	*2596*
5.63	10.55	10.35	0.2	*2606*
5.62	6.64	9.9	3.26	*2596*
5.61	6.64	9.45	2.81	*2612*
5.6	6.64	9	2.36	*2596*
5.59	6.64	8.55	1.91	*2625*
5.58	6.64	8.1	1.46	*2598*
5.57	4.3	7.65	3.35	*2613*
5.56	4.3	7.2	2.9	*2597*
5.55	4.3	6.75	2.45	*2607*
5.54	4.3	6.3	2	*2595*
5.53	4.3	5.85	1.55	*2613*
5.52	4.3	5.4	1.1	*2596*
5.51	4.3	4.95	0.65	*2607*
5.5	4.3	4.5	0.2	*2595*
5.49	4.3	4.05	0.25	*2613*
5.48	4.3	3.6	0.7	*2597*
5.47	4.3	3.15	1.15	*2607*
5.46	0.39	2.7	2.31	*2594*
5.45	0.39	2.25	1.86	*2611*
5.44	0.39	1.8	1.41	*2596*
5.43	0.39	1.35	0.96	*2622*
5.42	0.39	0.9	0.51	*2598*
5.41	0.39	0.45	0.06	*2611*
5.4	0	0	0	*2596*
5.39	0	0	0	*2606*
5.38	0	0	0	*2594*
5.37	0	0	0	*2683*
5.36	0	0	0	*2690*
5.35	0	0	0	*2606*
5.34	0	0	0	*2594*
5.33	0	0	0	*2611*
5.32	0	0	0	*2596*
5.31	0	0	0	*2606*
5.3	0	0	0	*2594*
5.29	0	0	0	*2611*
5.28	0	0	0	*2596*

5.27	0	0	0	*2625*
5.26	0	0	0	*2598*
5.25	0	0	0	*2611*
5.24	0	0	0	*2596*
5.23	0	0	0	*2606*
5.22	0	0	0	*2594*
5.21	0	0	0	*2611*
5.2	0	0	0	*2596*
5.19	0	0	0	*2606*
5.18	0	0	0	*2594*
5.17	0	0	0	*2611*
5.16	0	0	0	*2596*
5.15	0	0	0	*2606*
5.14	0	0	0	*2594*
5.13	0	0	0	*2611*
5.12	0	0	0	*2596*
5.11	0	0	0	*2621*
5.1	0	0	0	*2597*
5.09	0	0	0	*2611*
5.08	0	0	0	*2596*
5.07	0	0	0	*2606*
5.06	0	0	0	*2594*
5.05	0	0	0	*2611*
5.04	0	0	0	*2643*
5.03	0	0	0	*2606*
5.02	0	0	0	*2594*
5.01	0	0	0	*2611*
5	0	0	0	*2595*
4.99	0	0	0	*2606*
4.98	0	0	0	*2594*
4.97	0	0	0	*2611*
4.96	0	0	0	*2596*
4.95	0	0	0	*2630*
4.94	0	0	0	*2597*
4.93	0	0	0	*2612*
4.92	0	0	0	*2596*
4.91	0	0	0	*2607*
4.9	0	0	0	*2595*
4.89	0	0	0	*2612*
4.88	0	0	0	*2636*
4.87	0	0	0	*2682*
4.86	0	0	0	*2595*
4.85	0	0	0	*2612*
4.84	0	0	0	*2596*
4.83	0	0	0	*2606*
4.82	0	0	0	*2595*

4.81	0	0	0	*2612*
4.8	0	0	0	*2596*
4.79	0	0	0	*2625*
4.78	0	0	0	*2599*
4.77	0	0	0	*2612*
4.76	0	0	0	*2596*
4.75	0	0	0	*2606*
4.74	0	0	0	*2595*
4.73	0	0	0	*2612*
4.72	0	0	0	*2596*
4.71	0	0	0	*2606*
4.7	0	0	0	*2595*
4.69	0	0	0	*2612*
4.68	0	0	0	*2596*
4.67	0	0	0	*2606*
4.66	0	0	0	*2595*
4.65	0	0	0	*2612*
4.64	0	0	0	*2596*
4.63	0	0	0	*2625*
4.62	0	0	0	*2599*
4.61	0	0	0	*2612*
4.6	0	0	0	*2596*
4.59	0	0	0	*2606*
4.58	0	0	0	*2595*
4.57	0	0	0	*2612*
4.56	0	0	0	*2596*
4.55	0	0	0	*2634*
4.54	0	0	0	*2713*
4.53	0	0	0	*2612*
4.52	0	0	0	*2596*
4.51	0	0	0	*2606*
4.5	0	0	0	*2595*
4.49	0	0	0	*2612*
4.48	0	0	0	*2596*
4.47	0	0	0	*2622*
4.46	0	0	0	*2599*
4.45	0	0	0	*2612*
4.44	0	0	0	*2596*
4.43	0	0	0	*2606*
4.42	0	0	0	*2595*
4.41	0	0	0	*2612*
4.4	0	0	0	*2595*
4.39	0	0	0	*2606*
4.38	0	0	0	*2649*
4.37	0	0	0	*2612*
4.36	0	0	0	*2595*

4.35	0	0	0	*2606*
4.34	0	0	0	*2595*
4.33	0	0	0	*2612*
4.32	0	0	0	*2596*
4.31	0	0	0	*2624*
4.3	0	0	0	*2600*
4.29	0	0	0	*2613*
4.28	0	0	0	*2596*
4.27	0	0	0	*2607*
4.26	0	0	0	*2596*
4.25	0	0	0	*2613*
4.24	0	0	0	*2596*
4.23	0	0	0	*2606*
4.22	0	0	0	*2596*
4.21	0	0	0	*2613*
4.2	0	0	0	*2596*
4.19	0	0	0	*2606*
4.18	0	0	0	*2596*
4.17	0	0	0	*2613*
4.16	0	0	0	*2596*
4.15	0	0	0	*2626*
4.14	0	0	0	*2599*
4.13	0	0	0	*2613*
4.12	0	0	0	*2596*
4.11	0	0	0	*2606*
4.1	0	0	0	*2596*
4.09	0	0	0	*2613*
4.08	0	0	0	*2596*
4.07	0	0	0	*2606*
4.06	0	0	0	*2632*
4.05	0	0	0	*2717*
4.04	0	0	0	*2596*
4.03	0	0	0	*2606*
4.02	0	0	0	*2596*
4.01	0	0	0	*2611*