acrt AI, Computer Science and Robotics Technology

IntechOpen Journals

RESEARCH PAPER

A Machine Learning Method on a Tiny Hardware for Monitoring and Controlling a Hydroponic System

Arpit Sharma[1], Anahita Taherkhani[2], Ezekiel Orba[3] and Aboozar Taherkhani[1,*]

1 School of Computer Science and Informatics, De Montfort University, Leicester, UK
2 Takestan Azad University, Iran
3 Roots Out CIC, Nottingham, UK
*Corresponding author: E-mail: aboozar.taherkhani@dmu.ac.uk

Citation
Arpit Sharma, Anahita Taherkhani, Ezekiel Orba, Aboozar Taherkhani (2025), A Machine Learning Method on a Tiny Hardware for Monitoring and Controlling a Hydroponic System. *AI, Computer Science and Robotics Technology* 4(1), 1–35.

DOI
https://doi.org/10.5772/acrt.20240016

Received: 19 March 2024
Accepted: 18 December 2024
Published: 24 January 2025

Abstract

The implementation of artificial intelligence on very tiny chips plays an important role in the future of IoT. Typically, these chips do not conduct artificial intelligence operations locally. They mostly send collected data to a cloud service, where artificial intelligence is implemented for decision making. This leads to a time lag and significant dependency of the system on an internet connection, making it unsuitable for systems requiring immediate action. In a hydroponic system, there is an immediate need to control the speed of a pump to maintain pH level. However, there are many challenges in designing an intelligent system using low- powered chips with low computational power. Therefore, achieving high AI accuracy is very difficult for these devices. Additionally, tiny devices need to communicate with the user to execute IoT operations. To overcome these challenges, a hydroponic system was designed in this study to incorporate an ESP32 chip-based microcontroller with sensors and actuators attached to it, so that AI on edge and IoT tasks can be executed simultaneously. A dedicated android app was implemented to monitor and control the system remotely via IoT. Results show that the predicted pump speed just falls behind the expected speed by an average of 2.94%. The overall designed system was stable and reliable. Komatsuna plants were grown in a hydroponic system and the yield was compared with the plants grown in standard potting compost.

The hydroponic system was monitored by the proposed method to produce a higher yield compared to the potting compost.

Keywords: edge computing, hydroponic system, internet of things, machine learning

1. Introduction

Food security has become a major concern amidst crises in the energy sectors, various geopolitical issues, and global warming in many countries. For instance, in the UK after Brexit, there was a significant "emphasis on the food security and self-sufficiency" [1] as produce which was previously imported from other European countries had to be largely grown by the local farmers. The COVID-19 pandemic had further adversely impacted food imports giving rise to the need to encourage local food production in the country. Since the climatic conditions are not very suitable for traditional soil farming in most parts of England, plants are mostly grown in artificially created indoor environments using techniques like "hydroponic irrigation" [2].

A Hydroponic System (HS) has several "advantages" [3] over traditional soil-based irrigation methods, lower water consumption, in particular. Water is recirculated in an HS as opposed to 99.9% wastage due to evapotranspiration in soil-based irrigation. HS uses just "10% of the water" [4] required for soil-based irrigation. The other advantages are higher crop growth rate (30–50% faster than soil-based technique), higher-quality crop, lower 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. Setting up a fully functional HS requires a large investment 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 circuitry that are selected according to the preferred mode of operation, which is a major research topic.

The main challenges of HS are related to the maintenance of the parameters like pH and Electrical Conductivity (EC) [5]. Since the plants growing in the environment of HS 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. Thus, an intelligent automatic control is required to monitor and control the systems in real-time.

The automatic control needs to be self-adaptive as the behavior of HS can vary from time to time due to temperature, lighting, and other environmental factors.

Therefore, an intelligent system is needed to control the system. Moreover, it is preferable to have the artificial intelligence (AI) 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 maintain the system reliability by eliminating the need for an uninterrupted internet connection and saving the time cost incurred in sending the request, and waiting for decisions over the internet, especially for the decisions which have to be made quickly, for instance, deciding the pump speed for a specific pH. However, AI models are generally run on an attached computer like Raspberry Pi [6, 7], which is expensive, bigger in size, and has a high-power requirement compared to small microcontrollers.

Machine learning (ML) on a very small and low-powered device, referred to as tinyML, has allowed handling ML activities on devices running on small microcontrollers like Arduino Nano 33 BLE Sense [8], ESP32 development board [9], NodeMCU development board [10], etc. A lighter version of an end-to-end open-source ML platform such as TensorFlow Lite [11] has been made for training light neural network models, whose learning parameters can be quantified to be small enough to be added to a tiny chip. In this study, ESP32 microcontroller was used.

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 if the system behaves abnormally. Internet of Things (IoT) allows the system to report the its status to the user remotely through the cloud. It also allows controlling the system remotely if the user detects any anomaly in the system. Therefore, a user interface such as a dedicated mobile app is needed for the system. The app and the microcontroller running the HS, must be connected to the IoT cloud server and publish and subscribe to various feeds to send and receive messages, respectively. The feed to which the microcontroller publishes, the app must subscribe, and vice-versa for successful message transmission.

In this study, an IoT-based, machine learning (ML) on edge is proposed for an HS to continuously monitor and control pH levels. Additionally, a mobile app and IoT control panel are implemented for managing the HS. The proposed model is shown in Figure 1.

In this study, the Adafruit IoT platform [12] functioned as a middleman between the ESP32 development board and the dedicated mobile application. The sensors used with the microcontroller were liquid level sensors and a pH sensor. A DC pump was used to pump lower pH liquid from the nutrient tank to the water in the HS to regulate the pH. As UDP was not considered to be a safe protocol, MQTT was selected as an alternative.

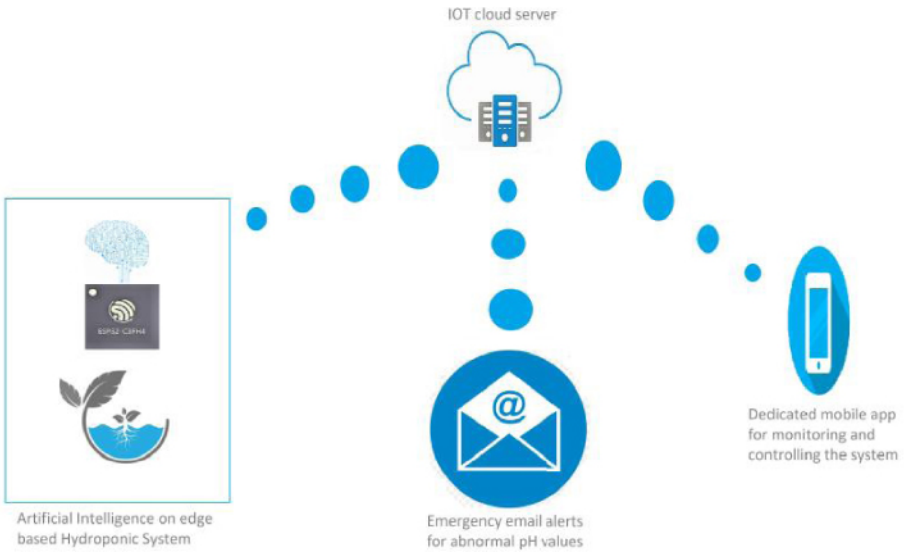

Figure 1. The proposed 'Intelligent HS'.

The main contributions of this study include the development and deployment of ML on a low- power edge device for real-time monitoring and controlling of a hydroponic system, the creation of a mobile app for IoT integration, achieving high system performance with a response time under three milliseconds, and addressing practical challenges such as sensor calibration and data pre-processing.

2. Literature review

Nutrient concentration and temperature significantly impact pH levels in hydroponic systems. For instance, variations in nutrient concentration can alter the solution's pH, while temperature fluctuations can affect nutrient uptake and pH stability. Although the main parameter that controls the pH level is the nutrient level, other factors such as water contamination during growth period may affect the pH level. These factors were excluded from the current study due to the complexity they add to pH management, but future studies will incorporate them to enhance the robustness of the system [13].

Megantoro *et al.*. [14] presented an IoT-based telemetry system using ESP32 microcontroller for real-time aquaculture monitoring, measuring parameters like temperature, pH, and dissolved oxygen (DO). The system offered high portability and accuracy through Google Firebase cloud storage, and a dedicated Android app was implemented for remote monitoring. The system improved the efficiency of aquaculture by providing real- time data access, making it suitable for both laboratory and field conditions.

The development of an IoT-based weather station embedded with O2, CO2, and CO sensor readings to monitor atmospheric conditions was presented in a study [15]. Arduino was used for data acquisition, with transmission to a cloud server via Wemos D1 Mini. The system provided real-time access to weather and air quality data, with an Android app for user monitoring. The study demonstrated how weather stations contribute to environmental assessments and energy utilization by leveraging accurate, remote telemetry systems.

An IoT-based monitoring system for hydroponic farming using ESP32 microcontroller, focusing on water pH, temperature, turbidity, and ambient air conditions was tested [16]. The system continuously monitored these variables and uploaded real-time data to Google Firebase. The role of weather conditions, particularly temperature and humidity, was crucial in maintaining optimal nutrient levels and ensuring healthy plant growth in hydroponic systems, directly influencing the quality and quantity of yields. Literature review substantiates the importance of IoT in hydroponic farming and monitoring of the system.

3. Materials and methods

3.1. Hardware and physical components

The following components were procured from online electronic stores, tested and assembled to build the physical prototype of the proposed system.

3.1.1. pH sensor

This sensor operated within a pH range of 0–14 with an accuracy of ±0.1 pH and a resolution of 0.01 pH, ensuring precise monitoring of nutrient solutions. The response time was less than 10 seconds, making it suitable for real-time applications. Monthly calibration using standard buffer solutions (pH 4.0, 7.0, and 10.0) is recommended to maintain accuracy. Regular maintenance involved cleaning the electrode with distilled water and storing it in a storage solution when not in use. This sensor integrated with microcontrollers via an analog output and required a 5 V power supply. During testing, it provided consistent and accurate readings. Isolated grounding was necessary to avoid interference.

An analog pH sensor/meter kit V2, shown in Figure 2, was used to measure the pH of the solution intended for the growth of plants. Being a very delicate and sensitive instrument, based on electrodes, the sensor was handled with “care” [17]. It was necessary to change the sensor before the “expiry date” of the sensor.

It was found out that the sensor malfunctioned when there were terminals, especially ground, of other sensors, or actuators in direct contact with the solution.

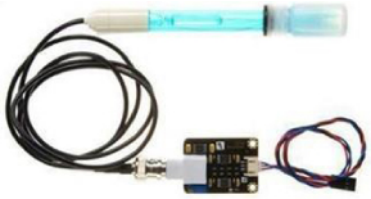

Figure 2. Analog pH sensor meter kit, shielding probe board cable, meter monitoring Analog pH sensor kit for Arduino [18].

The problem was significant enough to halt further experimentation. It was later determined that a ground loop problem was the reason for malfunctioning.

This was a major hurdle as the pH readings were unreliable as the pump was not properly isolated from the solution. This faulty mechanical problem could have been resolved by the following:

(1) Finding a pump with no leakage current (a pump with perfectly insulated terminals).
(2) Creating a completely independent power supply for the pump using pulse width modulated (PWM) signal [19] via optocouplers so that the pH sensor and the pump do not have a common ground.

With the former option, the pump could still malfunction due to wear and tear of the insulation leading to complete system failure, eventually damaging the plants being grown in the HS. The latter solution was more suitable.

Another issue faced was due to the contact level sensor, which also shared common ground with the pH sensor and few terminals in direct contact with the solution leading to ground loop [20] formations. This also had to be replaced with its contactless alternative, ultrasonic distance sensor.

Finally, the problem was resolved and it was ensured that no sensor was in contact with the liquid except the pH sensors. The ultrasonic level sensor was the only option for the level sensor as other sensors were expected to be in contact with the solution leading to the ground loop problem. The HC-SR04 ultrasonic distance measuring sensor was finalized as the level sensor for the experiment. The pump was controlled by optocouplers using PWM signal, given 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.

Sensing the pH level in an HS is an essential task. The optimal pH range for hydroponic systems is typically between 5.5 and 6.5. This range is crucial for maximizing nutrient uptake and ensuring the healthy growth of various crops, including leafy greens, tomatoes, and other vegetables [21].

3.1.2. Water level sensor

The HC-SR04 Ultrasonic Level Sensor was used for measuring distances and liquid levels in the HS. It has a measurement range of 2 cm to 400 cm with an accuracy of ±3 mm and a response time of approximately 20 ms. This sensor requires minimal maintenance, such as keeping the surface clean and ensuring proper connections. It outputs digital signals through trigger and echo pins, interfacing easily with microcontrollers, and operates at 5 V. During testing, the HC-SR04 demonstrated stable and accurate measurements, although it is sensitive to environmental noise and obstacles.

Figure 3. HC-SR04 ultrasonic sensor distance measuring module [19].

The sensor shown in Figure 5 was tried due to the ground loop problem discussed in the previous section.

When the nutrient solution was up to the brim of the tank, the value read by the microcontroller was 1000 mm (the height of the nutrient tank) which decreased gradually as the solution was pumped from the nutrient tank to the tank where plants are growing. The precision and the accuracy of the sensor were good and were also used in many similar experiments as evidenced in literature review.

3.1.3. Pump, and motor driver module

To control the speed of the pump to bring the pH of the solution in the desired range for plants to grow, in the desired range, a DC pump [22] is suitable, as the speed of DC pumps is proportional to the supplied voltage which can be controlled using PWM by the microcontrollers. DC 3–6 V Mini Micro Submersible Water Pump was

used in our study. An ideal DC pump would have zero leakage of current, i.e., circuitry would be completely insulated from the solution, however, these pumps are expensive and can cause a complete system failure, if they malfunction. These pumps also 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.

A hose is necessary for the assembly of an HS, which needs to be leak-proof to preserve the entire data collection process and the general operation of the system. Leaks also lead to wastage of the expensive nutrient solution, which can increase the running cost of the system. That would even make the HS no longer less expensive than traditional irrigation methods, which is otherwise supposed to be a major advantage of HS over traditional irrigation methods.

The thickness of the hose should also be appropriate for the pump. If it is thicker than the pump can deliver the solution through, it can 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 inefficient.

Two different DC motor drivers were tested for the experiment. The first was an H-bridge-based module. 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 replaced with the 1 Channel IRF540 MOSFET motor driver [23].

3.1.4. ESP32 microcontroller

ESP32 microcontroller's compatibility with Arduino IDE [24] and other IDEs has made it accessible for many users including professionals, students, and hobbyists.

A study by Babiuch [9] discussed the capabilities and different variants of the ESP32 development board with their advantages and disadvantages. The study also focused on the type of applications that the microcontroller can be used for. The boards have the ability to run a lighter version of python, made for microcontrollers known as microPython.

The microcontroller is supported by the Arduino IDE and has several libraries in Arduino IDE for IoT and machine learning such as 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 to handle complex code. It also supports 802.11 b/g/n Wi-Fi connectivity with speeds up to 150 Mbps, thus 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 × SPI, 2× I2C, 2× I2S, and 3 × UART which allows the sensor values to be printed on the console monitor of the computer for printing sensor values dynamically for debugging, and data collection for creating neural network As it has PWM 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 determined that for dataset creation, a local display attached to it was not required 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. The ESP32 microcontroller, used in our system, demonstrated varying power consumption levels depending on its operational mode. In active mode, the ESP32 consumed 78.32 mW, while in deep sleep mode, the consumption dropped to 10–150 μW, depending on the features activated. This efficient power management makes the ESP32 suitable for energy-sensitive applications in IoT environments [25]. Therefore, it was used to collect the data from the sensors to build the dataset to be used for training the neural networks.

3.1.5. Sliding potentiometer

The 10K Logarithmic Slide Potentiometer was used to control and measure variable resistance in the hydroponic system. This potentiometer offers a resistance range of 0–10 kΩ and provides instantaneous response time, making it ideal for real-time adjustments. Maintenance involves regular cleaning to ensure smooth operation and inspecting for wear and tear. It integrates with microcontrollers via an analog output and is compatible with both 3.3 V and 5 V systems. Testing showed that the potentiometer offered smooth variable resistance and stable performance, making it suitable for precise control applications.

A sliding potentiometer [26] 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 microcontroller takes the potentiometer reading through the analog pin the potentiometer is attached to, and then controls the speed of the pump by writing the reading to the digital pin the pump is attached to, using PWM. The potentiometer collected pH data to train a neural network as described in Section '3.6 Data Collection and Preparation'.

3.2. Circuit diagram of ESP32, sensor and the pump

The circuit was fitted as shown in Figure 12 with the motor driver pin connected to digital pin 17. The pH sensor pin was connected to analog pin 33. The trigger pin, and

the echo pin of the HC-SR04 ultrasonic sensor (Figure 3) were connected to pin 5 and pin 18, respectively, of the microcontroller.

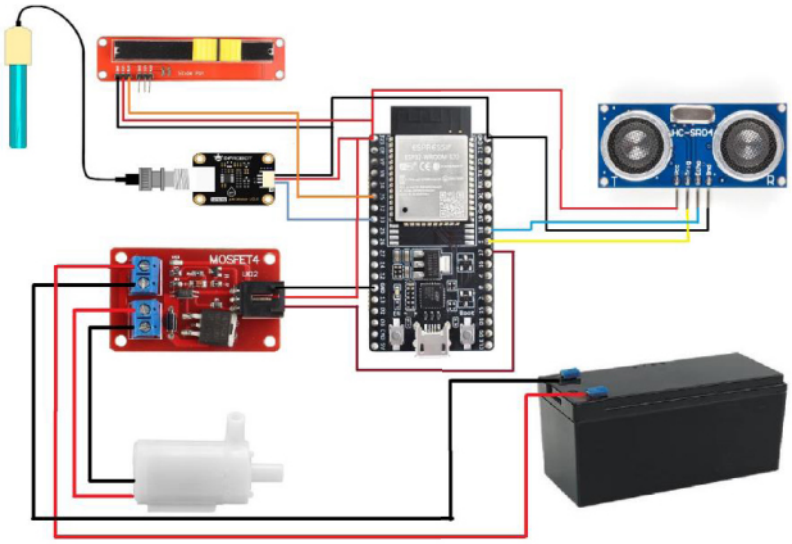

Figure 4. Final circuit diagram used in the project.

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.

3.3. Android app development

The app was created with Android Studio IDE in Java programming language. The user interface was carefully designed to make it aesthetically appealing. It used MQTT Android Client class to publish and subscribe to the feeds for monitoring and controlling the system via IoT. A dialog box was used to display the developer's details. Following are the components that make up the control panel screen of the app.

1. The battery component

The battery size was selected based on the pump power requirements. The sensors were connected to the analog pins of the microcontroller. A component displaying the battery level for the system was created in the app's user interface.

The app subscribed to the battery feed of the IoT server so that whenever the microcontroller published the latest battery status, the battery level was updated inside the battery component in the app, as shown in Figure 5. The setBatteryTv

method inside the BatteryFrag.java was called whenever the data on the battery feed changed, and an integer value(the battery level in percentage between zero and hundred) was received as a parameter, which updated the Textview for displaying the battery level. The entire layout was designed in XML format. This feature enables the future extension of the project to make the system, solar powered. At the time of the experiment, the system was plugged with the 5 V DC adapter plugged into the wall, and so the battery levels were always 100%.

2. The weather component

Weather information was retrieved using the API of openweathermap.org, the name of the city and country was selected to output the weather in JSON format. The JSON file was retrieved and parsed to extract individual parameters like current weather, wind direction, wind speed, outside temperature, and outside humidity. These parameters were used when the weather component in the control panel screen of the app (as shown in Figure 5) was updated with the latest weather details. Weather of the locality where the HS was located, outside wind direction and speed, outside temperature, and humidity were displayed on the app's user interface. The significance of displaying weather for controlling water plants in indoor hydroponic systems is described in Appendix A.

3. The Liquid-level component

This component displayed the liquid level left in the nutrient tank as detected by the ultrasonic range sensor. When the solution containing nutrients necessary for plants' growth was pumped to the main tank, the level sensor readings were continuously sent via IoT to the app for monitoring purposes. The values were in millimeters and based on the precision of the sensor. The setLiquidLevelTv method inside the LiquidLevelFrag.java was called and a double value (the liquid level in mm) was received as a parameter, which was used to update the Textview for displaying the liquid levels. It also helped the user to ensure that the nutrient tank was not empty.

4. The pH component

The pH component (as shown in Figure 5) showed the existing pH value and pH status of the tank inside which the plants are growing, which was the most important part of the project. This component (like other components) was updated when the pH value was published by the microcontroller to the cloud and received by the microcontroller. The callback methods, setPhValueTv and setPhStatusTv, were called for updating pH value and pH status, respectively.

5. The pump speed component

The speed of the pump was calculated by AI on the microcontroller and published to the IoT server, the slider of this component, as shown in Figure 5, also slid automatically according to the existing speed of the pump. The speed could also be controlled manually by sliding the slider which invoked the setOnSeekBarChangeListener callback method of the SeekBar class inside PumpSliderFrag.java, setting the text of the Textview, and the slider of the component to the current pump speed of the system.

The screenshots shown in Figure 5 display the main screen, and the control panel screen of the developed mobile app. There were many challenges and problems faced during the making of the app. However, buoyed by the massive online community support, the app was completed successfully.

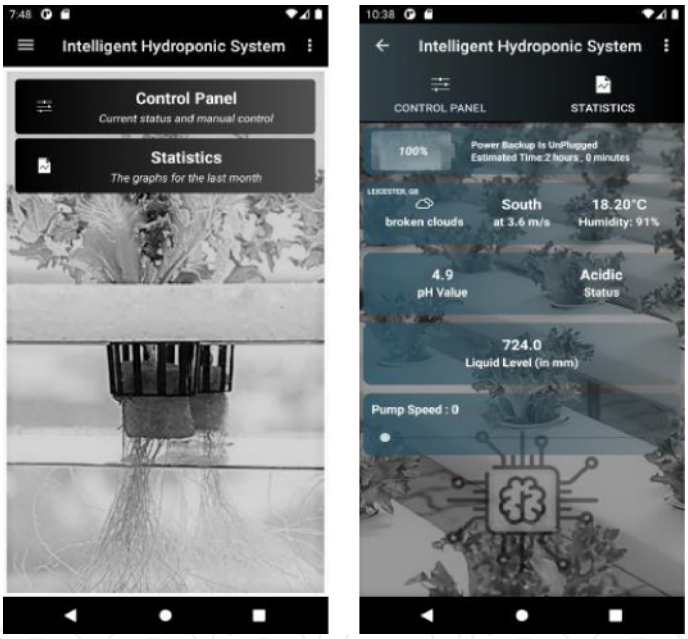

Figure 5. Screenshots of the developed app.

3.4. IoT cloud server setup

3.4.1. Adafruit dashboard

For IoT cloud server, the free tier of adafruit.io was chosen, which provided sufficient services and features for this study. 30 data points per minute, included in the free tier was sufficient as pH value, pH status liquid level, pump speed, and battery could be updated even after a minute for the user to monitor the system whenever required. As shown in Figure 6, the current value of every single feed used in the project was displayed in adafruit's cloud server dashboard.

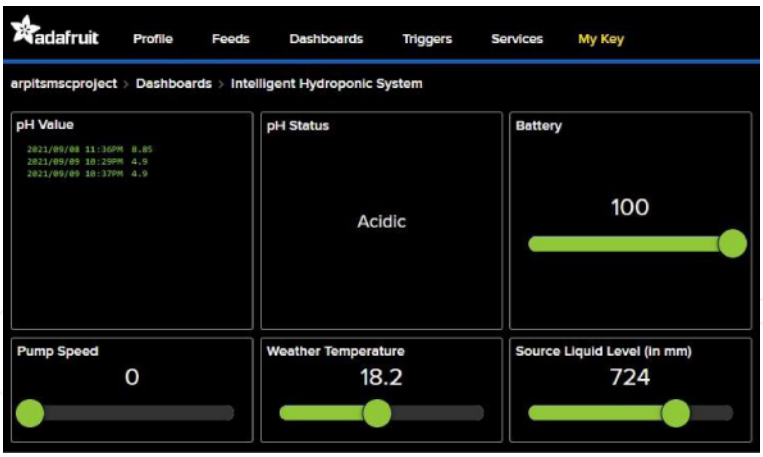

Figure 6. Adafruit Dashboard made for the project.

3.4.2. Adafruit feeds

Six feeds were used in the project as shown in Figure 7. Battery and weather temperature feeds were made for the future use cases and not for the study.

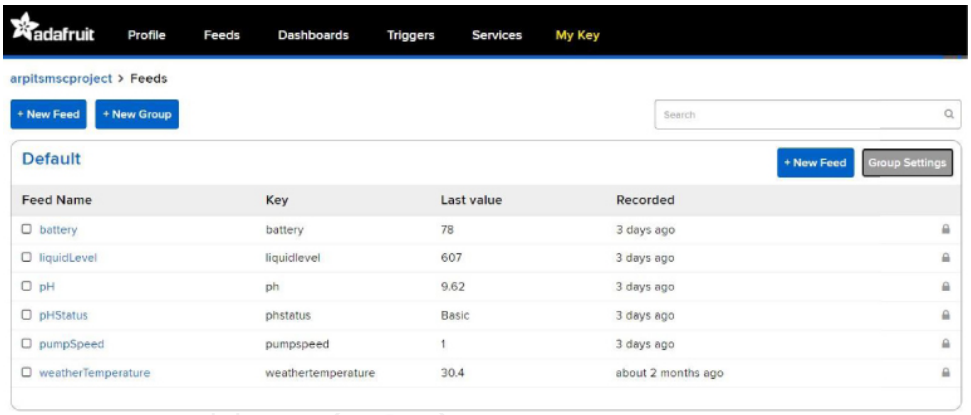

Figure 7. Total number of feeds used in the project.

3.4.3. Notifying users by email when pH is not in the desired range

Whenever the pH was not in the required range i.e., whenever the system behaved abnormally, an email alert was sent to the users automatically with the abnormal pH value at that instance and prompted to take action, a safety feature that was added to protect the cultivated plants. The system was made using Adafruit.io's integration with IFTTT which enabled all possible 'recipes' that were made when a feed value crossed a certain threshold set by the user. Therefore, for the system, an email alert was chosen as the notification method. The setup used at the IFTTT's and the adafruit.io's end is shown in Figures 8 and 9 respectively. If the pH increased beyond

the range, an email was received by the user almost immediately. if the pH decreased below the set lower limit, a corresponding email was sent as well.

Figure 8. Setup at IFTTT's end.

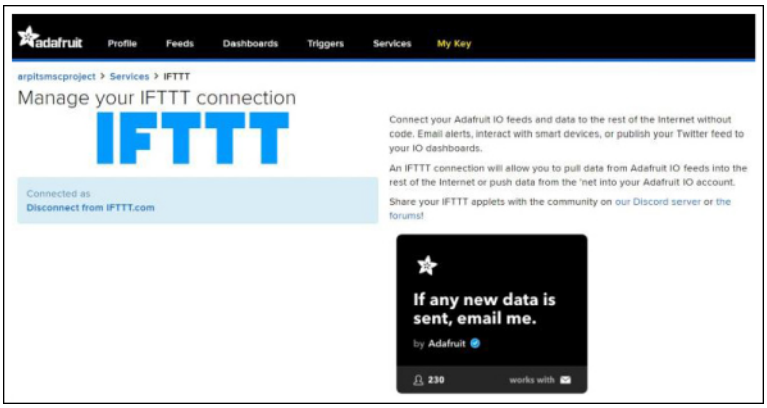

Figure 9. Setup at adafruit.io's end.

3.5. Dataset collection and preparation

The dataset was created by collecting pH sensor data for different pump speeds from the serial monitor of the Arduino IDE. A sliding potentiometer was used to set different speeds for the pump by manually sliding the potentiometer. A program was written to read the set value from the pin connected to the sliding potentiometer so that the pump could be controlled using PWM [27] technique through another pin as shown in Figure 4. At the same time, pH levels were recorded using the pH sensor.

An expert manually adjusted the pH to the required range using the sliding potentiometer to control the pump speed, which pumped the nutrient solution to the main tank where pH was measured. The sensor data was collected continuously while the user was doing the maneuver. The data was later processed to prepare training data for the neural network. The neural network then learned to mimic the user's action. This implemented the user's preference to control the parameters (pH in the project) of the HS, rather than a fixed controller like a PID controller.

A sample of the data set stored in a CSV file is shown in Figure 10, which is used for training the neural network model.

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 10. A sample of the dataset collected from pH sensor.

All the columns in the data set were stored in individual arrays. Then, (X, y) pairs were formed to prepare the data for training. The data was plotted to visualize its nature as shown in Figure 11. X-axis shows the pH values from 0 to 14 (the axis is inverted since the pH was decreased in the experiment as the nutrient solution was added. The nutrient was acidic in nature). Y-axis shows the pump speed, which was modified based on the pH value.

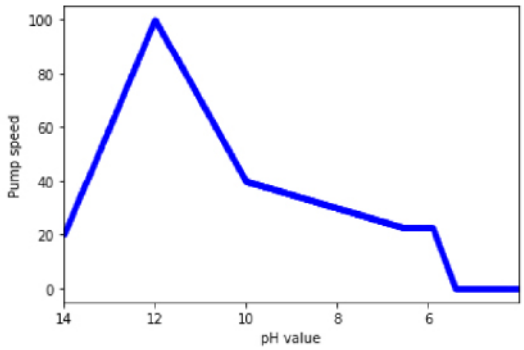

Figure 11. The nature of the dataset.

The dataset of 1000 samples was split into sixty percent for training, twenty percent for validation, and twenty percent for testing. 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 12.

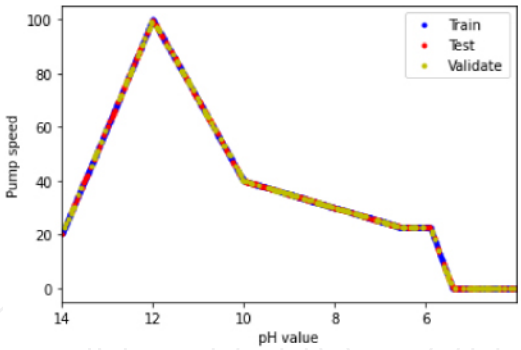

Figure 12. The dataset split into three sections.

3.6. The structure of the neural network

Neural network regression was done with a sequential model using the Keras library. Considering the importance of the size of the neural network to be deployed on an embedded system with low memory capacity and computation power, a balance was necessary between the size and accuracy of the model. Moreover, a large model with a high computation cost could increase the processing time and reduce the reliability of the model for an online application. To determine the appropriate size for the model, different models with different sizes were considered in this study to find an optimal model structure. The evaluation started with a small model having a single hidden layer used with only 8 neurons (as shown in Figure 13) as the minimum sized model for the system. Additionally, it had a low response time. However, accuracy was an issue. The largest model that was evaluated in this study is shown in Figure 14. The network had five hidden layers with ReLU activation function. The number of neurons in the five hidden layers were 32, 64, 128, 64, and 32. There was a single neuron at the input layer to obtain the pH value, and one output neuron to predict the pump speed. Different networks with different sizes between the smallest and the largest models were tested, and the results are presented in the result section.

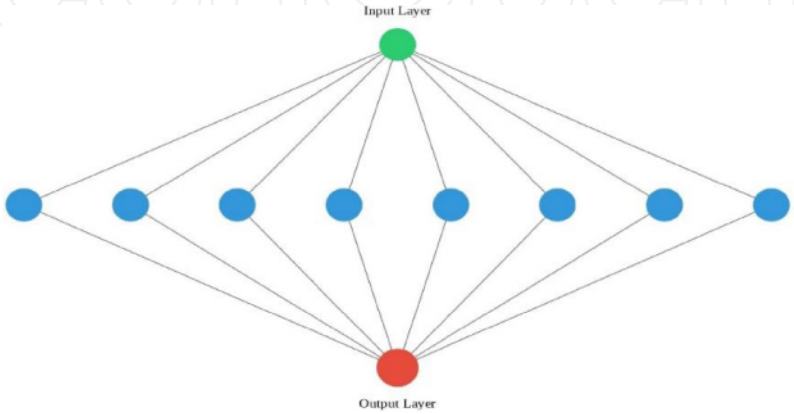

Figure 13. Smaller model configuration.

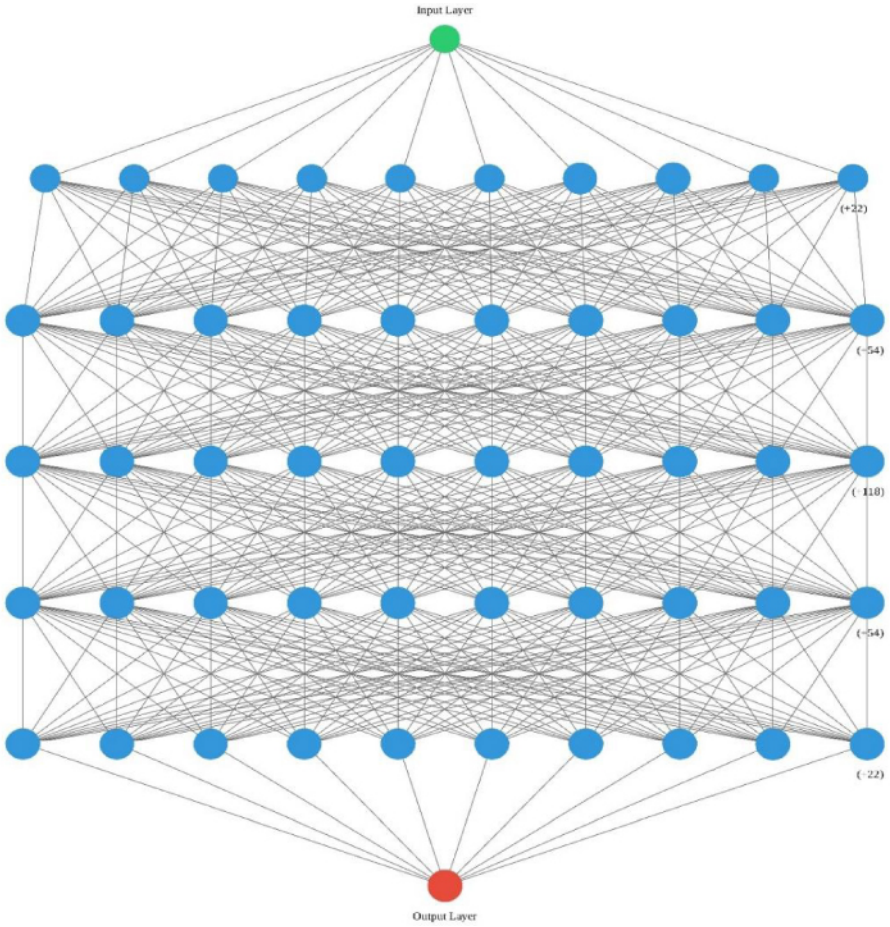

Figure 14. The configurations of the optimum model.

3.7. Trials

A trial run was executed between February and March 2022 to grow Komatsuna (*Brassica rapa* var. *perividis*) plant. An organic liquid nutrient was used to make up the hydroponic liquid growth medium, with seven Komatsuna seedlings, each grown in a separate spongy solid growth medium. The HS was situated in a small unheated building. For comparison, seven Komatsuna seedlings were grown in a standard potting compost in an unheated polytunnel, in seed modules with a similar volume of soil to the spongy solid growth medium used in the HS. Another control in an unheated glasshouse was conducted in addition to the polytunnel control. In HS, plants were grown under vegetative growth lighting settings in a 14-hour daylight cycle. The hydroponic liquid growth medium was topped up with tap water twice a week to the maximum level in the hydroponic tank, with nutrients only being added to the solution on day 1 to have a similar situation to the control ones. During plant growth, photographs and measurements of plant height from spongy solid growth medium (for the plants in the HS) or soil level were taken twice a week.

Seedlings from both groups were harvested and final measurements were taken when the majority of plants growing in the hydroponic unit had reached a maximum height (i.e. were touching the LED lighting unit at its fullest height extension setting). This took 43 days in this trial. All individual plants were cut at soil/'spongy solid growth medium' level, and each group had its total fresh yield measured in fresh above- ground biomass. The dried yield was then measured as dried above-ground biomass after drying samples for 1 h at 150 °C.

Measurement Errors

1. **Sensor Accuracy and Precision**:
    - **The pH Sensors** have an accuracy of ±0.1 pH, and require regular calibration. Errors can occur due to drift and contamination.
    - **The Ultrasonic Sensor** has an accuracy of ±3 mm, that can be affected by temperature, humidity, and obstacles.
2. **Calibration Errors**:
    - **The pH Sensor**, during its calibration with buffer solutions, can introduce errors if done improperly.
    - **The Ultrasonic Sensors**, during initial setup, can introduce errors leading to inaccurate measurements.
3. **Environmental Interference**:
    - **The pH Sensor** can be affected by electrical noise and ground loops.
    - **The Ultrasonic Sensor** can be affected by environmental noise and air conditions.

Control Errors

1. **Response Time Lag**:
    - Delays in sensor response and control adjustments can lead to inaccurate pH maintenance.
2. **Actuator Accuracy**:
    - **In Pumps and Valves**, inconsistent flow rates and leaks can cause over or under-correction.
3. **Data Processing and Algorithm Efficiency**:
    - Inefficient algorithms and poor data accuracy can lead to incorrect system adjustments.

4. **System Integration**:
    - Errors due to poor sensor integration and software bugs can affect system performance.

Error Mitigation Strategies include the following:

- Regular calibration and maintenance.
- Implementing robust error-checking algorithms.
- Ensuring proper sensor placement and environmental control.
- Using high-quality actuators and refining control algorithms.

By addressing these potential sources of error, the reliability and effectiveness of hydroponic systems was achieved.

3.8. Dependent and independent variables

In our experiment, the dependent and independent variables were defined as follows:

- Dependent Variables:
    - pH Level: The pH of the nutrient solution, which we aimed to control through the pump speed.
    - Plant Growth/Yield: The growth and yield of the plants, measured to assess the effectiveness of the hydroponic system.
- Independent Variables:
    - Pump Speed: This was varied to control the pH levels of the nutrient solution.
    - Nutrient Composition: The nutrient solution's composition was kept constant to ensure that it did not influence the pH or plant growth independently.

By clearly defining these variables, we could systematically study the effects of pump speed on pH control and, consequently, on plant growth and yield. This approach ensured that our experiment was structured and focused on the primary factors influencing the hydroponic system's performance.

4. Results

4.1. The evaluation of the model with different sizes

The training mean absolute error, training loss, validation loss and validation mean absolute error for the small network shown in Figure 13 were 322.52, 13.73, 332.07, and 14.34, respectively, which were extremely high.

The mean square error versus epochs graph for both validation loss and training loss were plotted to determine if 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 15 to see the detailed difference between the actual data, and the model's predictions. The mean absolute error was significantly higher than the mean squared error.

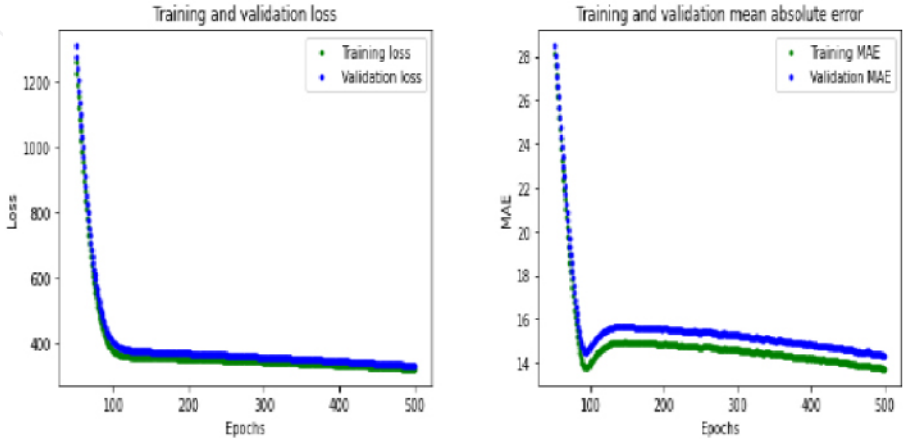

Figure 15. The graph of the smaller model's mean absolute error.

The actual data and the predicted values on the input versus output graph (as shown in Figure 16) reveal the true picture of how apart the predicted values are from the actual ones. Hence, the system's need was unmet because of its inefficiency to meet the accuracy required for maintaining pH conducive to the growth of healthy plants in the HS.

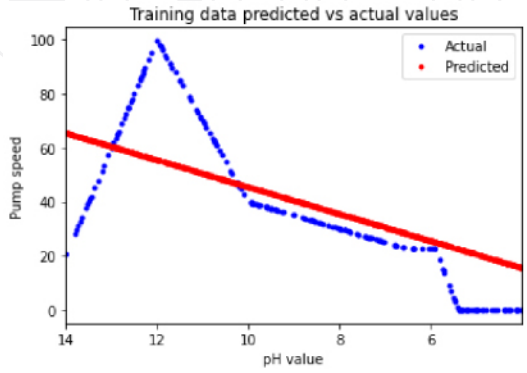

Figure 16. The graph of the smaller model's actual (blue) against the predicted values (red).

Therefore, we increased the size of the model by increasing the number of neurons, hidden layers, and epochs and adjusting other parameters to fine-tune the model.

The graphs of predicted and actual values for the intermittent models experimented during the tuning of the neural network model are shown in Figure 17. It is evident that due to the complexity (sharp and rapid changes in the graph) of the dataset, the output cannot follow the actual values for smaller model configurations. Therefore, the number of layers and the number of neurons were increased.

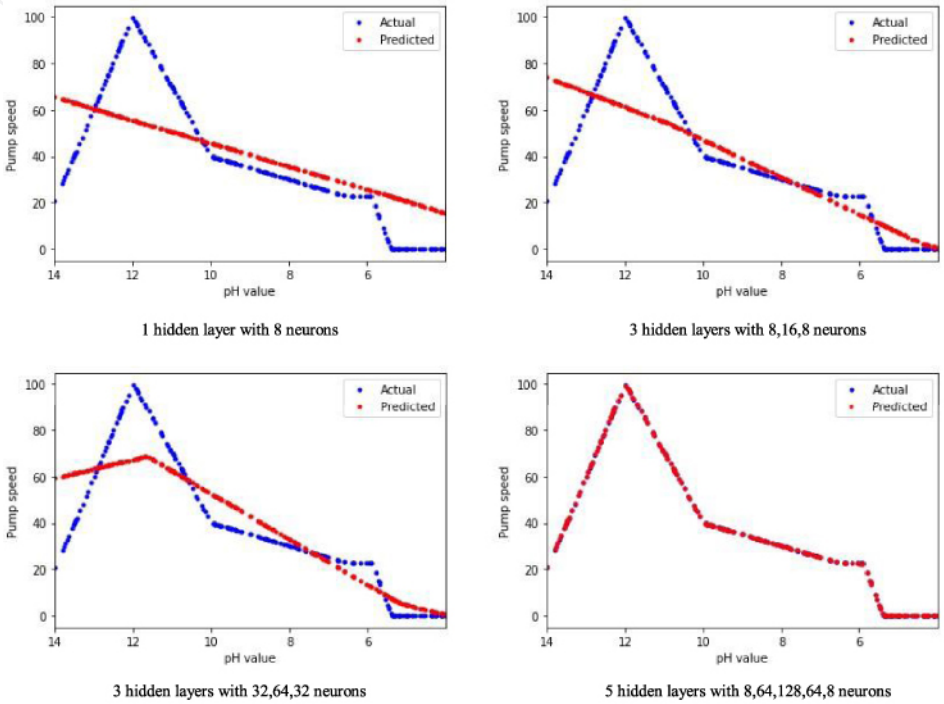

Figure 17. Comparison of predicted and actual values in intermediate models tried during fine-tuning of the neural model.

Ultimately, a larger model (see Figure 14) with five ReLU activated hidden layers with the number of neurons equal to 32, 64, 128, 64, and 32, with one neuron at the input layer for pH value, and one neuron at the output layer for pump speed was assigned.

As the number of epochs increased, the loss decreased and flattened at 270 epochs (as shown in Figure 18), but continued to decrease gradually. Hence, increasing the number of epochs is desirable as accuracy is important for HS. It was also for this reason, the early stopping feature was not preferred while training the model.

The mean absolute error (shown in Figure 19) was higher than the mean squared error, however, the inferences are the same as made earlier.

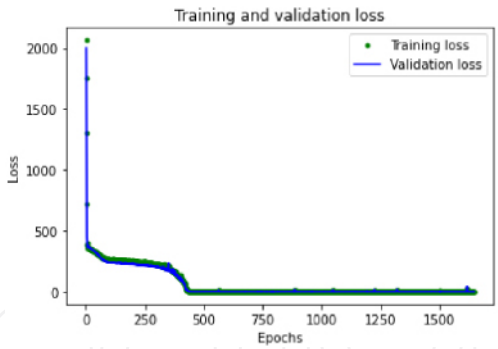

Figure 18. The graph of the optimum model's mean squared error.

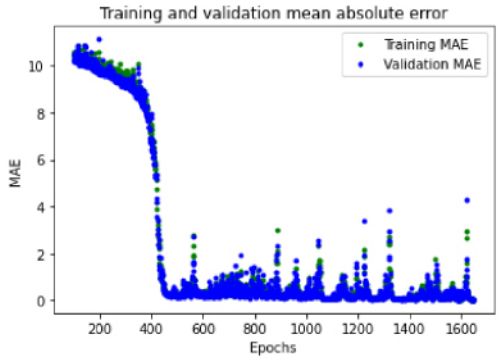

Figure 19. The graph of the optimum model's mean absolute error.

A reliable and minimum error became evident at 1650 epochs (shown in Figure 19) implying that a minimum of 1650 epochs are needed to train the system. The model reached the training loss value, training MAE, validation loss value, and validation MAE of 0.0038, 0.0445, 0.0015, and 0.0280, respectively.

4.2. *K*-fold cross-validation technique

In a further attempt, instead of taking a fixed data split for training the neural modal, *K*-fold cross-validation [28] was used. In the final model, five folds with 300 epochs each were attempted in an effort to find a better data split. The model was trained on the data split that had the best results for 1639 epochs resulting in 0.0445 and 0.028, MAE training and validation losses, respectively. It was ensured that the validation loss was not more than the training loss. Otherwise, the model would have stopped generalizing to new data.

4.3. Optimizers and learning rate

Different optimizers like 'Adam' optimizer, 'RMSprop', and 'Adadelta' optimizers were tried during the experimentation. The best results were obtained with 'Adam'

optimizer and the extremely poor results were seen with 'RMSprop' optimizer. Eventually, 'Adam' optimizer with a learning rate of 0.001 was selected for training the system.

Finally, the actual data and the predicted values depicted in the input vs output graph (shown in Figure 20) describes how accurately the predicted values are on the mark, and the system was predicting reliable results. The only drawback of this model was its immense size making it very difficult to run it on a tiny low powered chip. A method to reduce the size of the model was yet to be determined. The

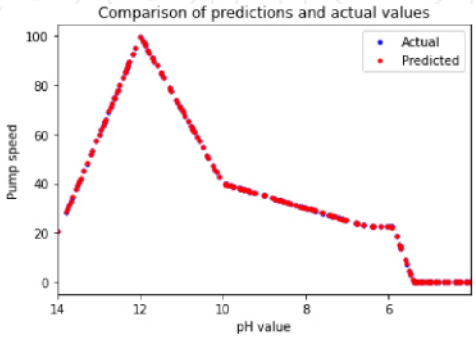

Figure 20. The graph of optimum model's actual vs. predicted values.

model was then quantized to reduce its size, to be easily accommodated by the microcontroller with its limited memory space and processing power. The accuracy of the quantized and the unquantized models are compared on the graph to confirm that the performance of the quantized model was not reduced significantly. The graphs of the actual data, and the predictions made by the quantized and unquantized models are plotted (Figure 21); it can be inferred that there was no significant difference in their performances and the quantized model suitable to be run on the microcontroller. Lastly, the trained model is converted to a C source file to be exported to a microcontroller.

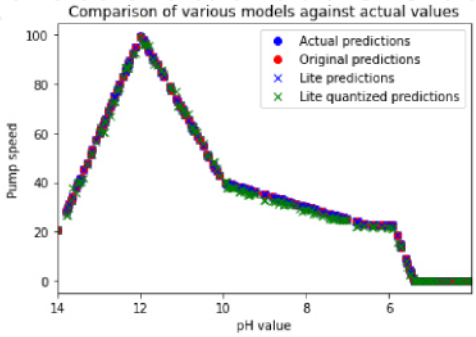

Figure 21. The comparison of the quantized model with the unquantized model.

4.4. 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 module 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 determine the accuracy of AI on edge predicted the speed of the pump according to the pH values.

4.5. 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 unseen data, were determined according to mean squared error and mean absolute error (Table 1). The model with the lowest overall loss was chosen as the optimum model for the experiment.

Table 1. Training and validation losses for the prominent models experimented in this study with the *K*-fold cross-validation technique delivering minimum error.

Model number	Model configurations	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	281.0879	11.3358	273.5565	11.5603
3	3 hidden layers with 8, 16, 8 neurons	260.8473	10.4382	242.9051	10.2793
4	3 hidden layers with 16, 32, 16 neurons	247.3403	10.1742	229.8249	9.9623
5	3 hidden layers with 32, 64, 32 neurons	160.5955	8.9749	149.886	8.9081
6	3 hidden layers with 32, 64, 33 neurons	0.0121	0.0797	0.0161	0.1048
7	3 hidden layers with 64, 128, 64 neurons	0.0417	0.1685	0.0385	0.1328
8	5 hidden layers with 8, 64, 128, 64, 8 neurons	0.011	0.074	0.0073	0.0617
9	5 hidden layers with 16, 64, 128, 64, 16 neurons	0.0844	0.2216	0.0095	0.0624
10	5 hidden layers with 32, 64, 128, 64, 32 neurons	0.0581	0.1969	0.0451	0.1934
11	7 hidden layers with 8, 32, 64, 128, 64, 32, 8 neurons	0.0294	0.1172	0.0554	0.174
12	**5 hidden layers with 8, 64, 128, 64, 8 neurons using *K*-fold cross validation**	**0.0038**	**0.0445**	**0.0015**	**0.028**

Model number 12 (shown in Table 1) showed the minimal values for every loss calculation method with 0.0038 and 0.0015, training and validation losses, respectively, based on mean squared error; and 0.0445 and 0.0280, training and validation losses, respectively, based on mean absolute error, making it the 'fittest' model. The model had 5 hidden layers with 8, 64, 128, 64, 8 neurons respectively, Adam optimizer with a learning rate of 0.001, ReLU activated hidden layers with 64

batch size, and 1639 epochs and used *K*-fold cross-validation configurations discussed before.

4.6. The physical apparatus of the proposed system

The physical prototype of the HS used for the experiment including the collection of data for the neural model, and testing AI and IoT operations are as follows. The system is well connected wirelessly with the mobile app running on a tablet and a laptop via IoT. The laptop and the tablet receive the live feed of the system remotely from any part of the world as long as there is internet access. Another configuration was used to do trials for growing Komatsuna as described in Section '4.8 Results of the hydroponic trial for growing Komatsuna'.

4.7. Model size versus time per inference

The size of the model being run on Arduino IDE is the number of comma-separated hexadecimal numbers required to represent it. The time taken per AI inference on the microcontroller/edge against the size of the model is plotted in Figure 22 to elucidate the effect of model's size on the processing time required for each inference by the microcontroller for conducting machine learning operations on edge. The inference time and the sizes of different-sized models are noted in Table 2.

Figure 22. The graph of model size vs the time taken by the microcontroller per inference.

As depicted in Figure 22 and Table 2 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 2. 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. Moreover, it can be inferred that the KtensorArenaSize should be kept as minimum as possible to reduce the latency of the system.

Table 2. Training and validation losses for the prominent models experimented during this study with the *K*-fold cross-validation technique delivering minimum error.

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	500	2872	230	2.2 × 1024
3 hidden layers with 8, 16, 8 neurons	500	3128	268	2.2 × 1024
3 hidden layers with 16, 32, 16 neurons	500	4040	375	2.2 × 1024
3 hidden layers with 32, 64, 32 neurons	500	7400	779	2.2 × 1024
3 hidden layers with 32, 64, 33 neurons	1650	7400	779	2.2 × 1024
3 hidden layers with 64, 128, 64 neurons	1650	20264	2315	2.2 × 1024
5 hidden layers with 8, 64, 128, 64, 8 neurons	1650	22240	2623	3.2 × 1024
5 hidden layers with 16, 64, 128, 64, 16 neurons	1650	23344	2808	3.2 × 1024
5 hidden layers with 32, 64, 128, 64, 32 neurons	1650	25552	3161	3.2 × 1024
7 hidden layers with 8, 32, 64, 128, 64, 32, 8 neurons	1650	27080	3313	4.0 × 1024
5 hidden layers with 8, 64, 128, 64, 8 neurons using *K*-fold cross validation	1639	22352	2606	3.2 × 1024

From the fifth and sixth models in Table 2, 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 repeatedly for the same models to determine the precision. The time was almost the same for the same models implying that the microcontroller can be relied for the time, it will be taking for each inference for a fixed model size.

The predicted speed was reliable as depicted in Figure 23 and the graphs of expected speed and the predicted speed almost overlap. The predicted speed just fall behind 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 was zero (that is when the pump is off). The maximum error was 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 is not enough for the model to be trained properly in that area.

The most important time was when the pump needed to be in the off state. It was very dangerous for the system to have the pump false triggered as it can keep slowly pumping the nutrient solution to the main tank eventually making the main tank's solution highly acidic, and unfit for plant growth. However, simulation results showed that the speed predictions at low- speed levels were very accurate, and the system was free from this danger.

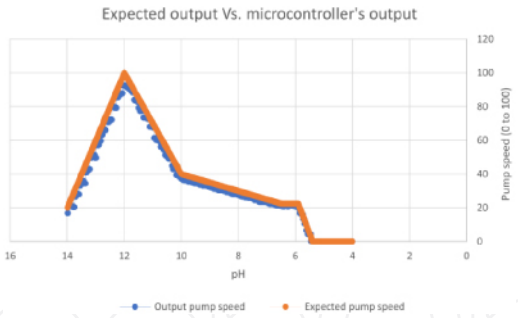

Figure 23. Expected pump speed Vs. physical system's pump speed according to pH values.

4.8. Results of the hydroponic trial for growing Komatsuna

Shown in Figure 24 are a selection of images demonstrating the differences in growth of Komatsuna between the glasshouse control group and the HS. Images of the second control group (polytunnel) are not included, as their appearance was visually similar to the glasshouse control. Both crop height and crop mass were measured for plants grown in the HS and the soil-based control groups.

4.8.1. Crop growth comparison and system feasibility

In this study, crop growth and yield were compared between hydroponic and soil-based cultivation systems under field trial conditions. The trials revealed differences in crop performance across the systems, although they were not designed to be directly comparable due to fundamentally different cultivation methods and uncontrolled environmental factors, such as temperature, lighting intensity, light spectrum, and daylight length.

Crop Height: The average plant height in the hydroponic group was 207 mm, which was greater than the average height in the glasshouse and polytunnel control groups, both of which were 47 mm.

Crop Mass: The total fresh yield of the six plants in the hydroponic system was 196.9 g, exceeding that of the soil-based controls. The glasshouse control group produced a cumulative fresh yield of just 2.0 g, while the polytunnel control group produced 3.5 g. After drying, the hydroponic group achieved a total dried yield of 13.1 g, compared to 0.2 g and 0.4 g in the glasshouse and polytunnel groups, respectively.

The inclusion of soil-based cultivation groups, such as the glasshouse and polytunnel setups, provided a useful reference point to demonstrate the performance of the hydroponic system under real-world agricultural conditions on the same premises. While the glasshouse and polytunnel conditions were not

Figure 24. Visual comparison of Komatsuna growth in (a) glasshouse and (b) the hydroponic system with pH and tiny AI monitor after 22, 29, 36, 40 and 43 days.

intended as direct controls, they highlight the stark contrast between soil-based and hydroponic cultivation methods in terms of both growth and yield.

Although indoor hydroponic growth is generally expected to outperform soil-cultivated salad crops in a glasshouse, the primary goal of this study was to validate the feasibility of the hydroponic growth method and assess the utility of the equipment—specifically, a compact AI-driven pH monitoring system—for agricultural workers in the field. This system holds potential as a practical tool for improving crop management practices in hydroponic setups.

5. Discussion

The pH value in our HS was dependent on pump speed because the pump controlled the addition of the nutrient solution to the tank, which directly impacted the pH level. By adjusting the pump speed, we can precisely control the amount of nutrient solution added to the tank, thereby stabilizing the pH at the desired level.

The following are the additional parameters that may have a significant impact on pH:

- Temperature: Temperature fluctuations can alter the chemical equilibrium of the nutrient solution, impacting pH levels.

- Humidity: While humidity has a less direct effect, it can influence the overall environment, potentially affecting the nutrient solution's properties.

- Nutrient Concentration: Variations in the concentration of nutrients can significantly change the pH, as different nutrients have different acidifying or alkalizing effects.

In this experiment, we focused primarily on the relationship between pump speed and pH as a simple, manageable approach. This approach allowed us to focus on the pH control mechanism. Other parameters like temperature and humidity were not controlled in this study but are acknowledged as factors that can impact pH. In future studies, these parameters will be monitored and managed to understand their effects on pH in detail. This focused approach allowed us to demonstrate the fundamental concept of using pump speed to control pH effectively, providing a foundation for more complex future research.

Our project primarily was focused on controlling the pH value with pump speed as the independent parameter. Given the simplicity of this project, we used a known nutrient solution with a stable pH value. When an acceptable level of the nutrient solution was added to the tank, the pH value in the tank is at a specific level. However, it is not a precise method and another factor or component may impact the relationship between the pH level and the appropriate level of nutrition. In future research, additional sensors will be used to measure the nutrient level.

The nutrient solution's composition was kept constant and was the same for both the glasshouse and hydroponic setups. In both setups, the pH of the nutrient solution was periodically measured to ensure consistency. Nutrient levels were indirectly maintained by ensuring that the solution's pH remained unchanged. For future projects, more complex nutrient solutions can be measured both directly and indirectly, ensuring comprehensive monitoring and control.

Our findings emphasize the importance of smooth adjustments during data collection process to avoid irregularities which can be identified in pH versus pump speed graph, and may complicate model training. Proper data preprocessing, including removing outliers and filling gaps, is critical for creating a high-quality dataset for neural network training. Isolated circuitry for the pH sensor is necessary to avoid ground loop issues, and selecting compatible modules is crucial to prevent interference.

For optimal performance, maintaining a fixed shape for containers and accessories is vital, as the AI model is trained on specific configurations. Changes in hardware, such as replacing pumps, may require retraining the AI model. The system currently uses a single acidic nutrient tank, limiting pH control. Future improvements could include incorporating both acidic and alkaline tanks with separate pumps for better control.

The mobile app is robust, retaining its last state upon reopening and providing reliable notifications via email alerts. This feature ensures that the system remains functional even without a continuous internet connection, making it suitable for remote farms reliant on limited solar power.

6. Conclusion

Food security has become a major concern due to crises in the energy sectors, geopolitical matters, and global warming. Implementing intelligent systems to monitor and control Hydroponic Systems (HS) is a promising method for securing food production. In our study, a machine learning method was developed and deployed on an edge device in the HS. A mobile app was created to connect to the edge device, acting as an IoT interface for monitoring and control. All computations for the edge device were performed on small, low-powered hardware.

The overall performance of the system was evaluated and the system achieved a very low response time of less than three milliseconds, which is significantly faster than typical "AI in the cloud" systems. This rapid response time ensures system reliability and robustness, crucial for real-time applications. However, the system exhibited a higher error rate in practical applications, which can be addressed through fine-tuning, advanced models, and rigorous testing. Reducing the model size further could enhance response time at the cost of accuracy.

7. Future work

The energy consumption of an IoT system is important. Although ESP32 consumes relatively little energy, it is worth considering further studies to reduce the system's energy consumption by recording appropriate energy data and analyzing the data.

Future work will focus on further optimization of energy consumption, enhancing system scalability, and integrating additional parameters for comprehensive control. Incorporating more advanced control algorithms and exploring alternative hardware configurations will also be pursued to improve system accuracy and reliability. Additionally, expanding the mobile app's functionality to include interactive features for users and stakeholders to enhance its utility and user experience is also considered.

After investigating the hydroponic system's feasibility, other study area of focus are: (1) assessing how cultivators perceive the system's ease of use and effectiveness in field conditions, and (2) measuring whether the system improves crop yield and quality in real-life agricultural applications when compared to other pH monitoring devices. By integrating both user feedback and measurable outcomes, subsequent studies could provide a comprehensive evaluation of the system's practicality and impact.

Authors' Contribution

Sharma, Arpit: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing; **Taherkhani, Anahita:** Investigation, Methodology, Validation; **Orba, Ezekiel:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Validation, Visualization, Writing – review & editing; **Taherkhani, Aboozar:** Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Project administration, Resources, Supervision, Validation, Visualization, Writing – review & editing.

Funding

This research did not receive external funding from any agencies.

Ethical Statement

Not Applicable.

Data Availability Statement

Source data is not available for this article.

Conflict of Interest

The authors declare no conflict of interest.

Appendix A. The significance of displaying weather for controlling water plants in indoor hydroponic systems

Even though our experiment was conducted entirely indoors, displaying weather information was still significant due to indirect influence on the system's environmental control mechanisms. To ensure optimal plant growth, the following precautions were taken.

A.1. Temperature management

External Temperature Influence: External temperature can impact indoor environment through Heat Ventilation and Air Conditioning (HVAC) systems. We monitored outdoor temperatures to pre- adjust HVAC settings, to ensure a stable indoor temperature, prevent any drastic changes that could affect the plants, and maintain the ideal temperature for plant growth.

A.2. Humidity control

Ventilation Impact: External humidity levels can affect indoor ventilation efficiency. Outdoor humidity data was used to adjust the indoor humidity and to maintain the desired humidity levels to prevent plant stress and disease.

Dehumidification Needs: High external humidity necessitated the increase in indoor dehumidification efforts to keep the environment within optimal ranges for hydroponic functioning.

A.3. Energy efficiency

Predictive Heating/Cooling: Weather forecasts helped us predict and adjust heating and cooling needs in advance, optimizing energy use and ensuring a stable environment.

Resource Planning: Understanding weather patterns allowed us to plan resource use more efficiently, ensuring minimal energy consumption while maintaining ideal conditions.

A.4. Water management

Evaporation Rates: Although the system was indoors, external weather could influence indoor evaporation rates via ventilation. Monitoring weather helped us adjust water levels and nutrient delivery systems accordingly.

A.5. System stress prevention

Proactive Measures: By monitoring weather changes, we took proactive measures to prevent system stress. For example, we increased cooling efforts before expected heatwaves or adjusted humidity controls ahead of humid periods.

In summary, even though our experiment was conducted in a fully controlled indoor environment, we adhered to these precautions, leveraging external weather data to optimize temperature, humidity, and energy management. This ensured that our hydroponic system operated efficiently and effectively, maintaining ideal growing conditions for the plants.

Appendix B

The photos of the implementation setup show electrical and non-electrical components used during lab implementation and testing.

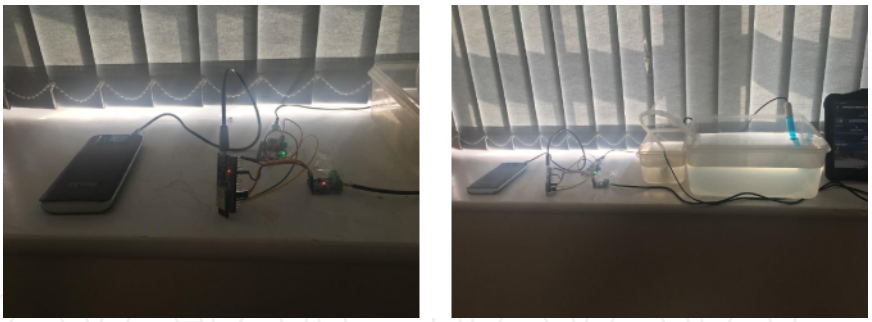

Figure B.1. Lab implementation set up for the project.

References

1 Dieter H. Agriculture after brexit. *Oxford Rev Econ Policy*. 2017;**33**(suppl 1):S124–S133.

2 Jurga A, Pacak A, Pandelidis D, Kamierczak B. A long-term analysis of the possibility of water recovery for hydroponic lettuce irrigation in an indoor vertical farm. Part 2: rainwater harvesting [English] [Internet]. *Appl Sci*. 2021;**11**(1):310, doi:10.3390/app11010310. Available from: https://go.exlibris.link/Cqbh5RXR.

3 Tripp T. *Hydroponics advantages and disadvantages: Pros and cons of having a hydroponic garden*. UK: Speedy Publishing LLC; 2014.

4 Souza SV, Gimenes RMT, Binotto E. Economic viability for deploying hydroponic system in emerging countries: a differentiated risk adjustment proposal. *Land Use Policy*. 2019;**83**: 357–369.

5 Abou-Hadid AF, Abd-Elmoniem EM, El-Shinawy MZ, Abou-Elsoud M. Electrical conductivity effect on growth and mineral composition of lettuce plants in hydroponic system. *Strateg Market Orient Greenh Product*. 1995;**434**: 59–66.

6 Richardson M, Wallace S. *Getting started with raspberry PI*. California: O'Reilly Media, Inc; 2012.

7 Mehra M, Saxena S, Sankaranarayanan S, Tom RJ, Veeramanikandan M. IoT based hydroponics system using deep neural networks. *Comput Electron Agric*. 2018;**155**: 473–486.

8 Kurniawan A. Arduino nano 33 BLE sense board development. In: *IoT projects with Arduino Nano 33 BLE sense*. Heidelberg: Springer; 2021. p. 21–74.

9 Babiuch M, Foltnek P, Smutn P. Using the ESP32 microcontroller for data processing. In: *2019 20th International Carpathian Control Conference (ICCC)*. Piscataway, NJ: IEEE; 2019. p. 1–6.

10 Parihar YS. Internet of things and NodeMCU. *J Emerg Technol Innovat Res*. 2019;**6**(6):1085.

11 Li S. Tensorflow lite: on-device machine learning framework. *J Comput Res Dev*. 2020;**57**(9):1839.

12 Singh D, Sandhu A, Sharma Thakur A, Priyank N. An overview of IoT hardware development platforms. *Int J Emerg Technol*. 2020;**11**: 155–163.

13 Hendrickson T, Dunn BL, Goad C, Hu B, Singh H. Effects of elevated water temperature on growth of basil using nutrient film technique. *HortScience*. 2022;**57**(8):925–932. Retrieved 2024 Sept 19, doi:10.21273/HORTSCI16690-22.

14 Megantoro P, Anugrah A, Abdillah M, Kustanto B, Fadhilah M, Vigneshwaran P. Smart measurement and monitoring system for aquaculture fisheries with IoT-based telemetry system. *Bull Electr Eng Inf*. 2024;**13**(3):1555–1565. doi:10.11591/eei.v13i3.6900.

15 Megantoro P, Saud Al-Humairi S, Kustiawan A, Arsalan M, Prastio R, Awalin L, et al. Development of an internet of things-based weather station device embedded with O2, CO2, and CO sensor readings. *Int J Electr Comput Eng*. 2024;**14**(1):1122–1134. doi:10.11591/ijece.v14i1.pp1122-1134.

16 Megantoro P, Prastio R, Kusuma H, Abror A, Vigneshwaran P, Priambodo D, et al. Instrumentation system for data acquisition and monitoring of hydroponic farming using ESP32 via Google firebase. *Indones J Electr Eng Comput Sci*. 2022;**27**(1):52–61. doi:10.11591/ijeecs.v27.i1.pp52-61.

17 Martin . ThermoWorks [Internet]. pH Meter Care and Common Mistakes; 2019 Dec 10 [cited 2021 Sept 5]. Available from: https://blog.thermoworks.com/ph-meter-care-and-common-mistakes/.

18 Analog pH sensor meter kit, cable of the shielding probe board, meter monitoring Analog pH sensor kit for Arduino: Amazon.co.uk: Home & Kitchen; [cited 2021 Aug 25]. Available from: https://www.amazon.co.uk/gp/product/B083XTLH3L/ref=ppx_yo_dt_b_asin_title_009_s00?ie=UTF8&psc=1.

19 HALJIA HC-SR04 Ultrasonic Sensor Distance Measuring Module Compatible with Arduino, Amazon.co.uk: Ultrasonic Proximity Sensors, [cited 2025 Jan.], Available from: https://www.amazon.co.uk/gp/product/B01DM8MRTS/ref=ox_sc_act_title_1?smid=AWLU8WJU8SoVS&psc=1.

20 Martin TH. Troubleshooting pH/ORP controllers. *PRODUCTS FINISHING-CINCINNATI-*. 1995;**59**: 56.

21 Gillespie DP, Papio G, Kubota C. High nutrient concentrations of hydroponic solution can improve growth and nutrient uptake of spinach (Spinacia oleracea L.) grown in acidic nutrient solution. *HortScience Horts*. 2021;**56**(6):687–694. doi:10.21273/HORTSCI15777-21, Retrieved 2024 Sept 19.

22 RUNCCI-YUN Automatic Irrigation DIY Kit Self Watering System with Capacitive Soil Moisture Sensor 1 Channel 5 V Relay Module and Water Pump + 1M Vinyl Tubing for Garden Plant Flower Herb Potted: Amazon.co.uk: Pet Supplies; [cited 2021 Aug 25]. Available from: https://www.amazon.co.uk/gp/product/B0814HXWVV/ref=ppx_yo_dt_b_asin_title_008_s00?ie=UTF8&psc=1.

23 HALJIA 1 Channel MOSFET Switch IRF540 Isolated Power Compatible with Arduino DIY etc.: Amazon.co.uk: Business, Industry & Science; [cited 2021 Aug 25]. Available from: https://www.amazon.co.uk/gp/product/B06XB5TPVG/ref=ppx_yo_dt_b_asin_title_000_s00?ie=UTF8&psc=1.

24 Arduino SA. *Arduino*. Monza: Arduino LLC; 2015. 372 p.

25 Cheddadi Y, Cheddadi H, Cheddadi F, Errahimi F, Es-sbai N. Design and implementation of an intelligent low-cost IoT solution for energy monitoring of photovoltaic stations. *SN Appl Sci*. 2020;**2**: 1165, doi:10.1007/s42452-020-2997-4.

26 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; [cited 2021 Aug 25]. Available from: https://www.amazon.co.uk/gp/product/B076PYGFFP/ref=ppx_yo_dt_b_asin_title_007_s00?ie=UTF8&psc=1.

27 Kherroubi ZEA, Akel F, Kermadi M, Berkouk EM. Real time implementation of space vector pulse width modulation using Arduino DUE board. In: *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. Piscataway, NJ: IEEE; 2016 Oct. p. 3576–3581.

28 Anguita D, Ghelardoni L, Ghio A, Oneto L, Ridella S. The 'K' in *K*-fold cross validation. In: *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) [Internet]*. i6doc.com publ.; 2012. p. 441–446. Available from: http://www.i6doc.com/en/livre/?GCOI=28001100967420.