Plant Guardian

Steven Keller, Tony Chau, Luz Romero, and Keven Hyppolite

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The Plant Guardian system provides an easy-to-use solution for maintaining indoor plants by automating essential watering processes. Designed to monitor and optimize plant health, this device combines a set of advanced sensors to track soil moisture, temperature, and pH balance, ensuring each plant remains in its ideal growing conditions. Plant Guardian includes a mobile app interface that enables users to view real-time data, receive updates, and control watering schedules remotely. By employing cost-effective and energy-efficient components, our system remains affordable and accessible, promoting eco-friendly practices for a broad audience.**

***Index Terms* — Sensors, sustainable technologies, mobile app, smart gardening.**

I. Introduction

Plants play an essential role in supporting life by providing food, medicine, oxygen, and improving air quality. With the growing popularity of houseplants, especially among millennials, people increasingly recognize the benefits plants bring to indoor environments, such as reducing stress, boosting mood, and enhancing creativity. However, maintaining houseplants can be a challenge, especially for individuals with busy schedules or frequent travel. Problems like overwatering, underwatering, and inconsistent care routines can arise, affecting the health and longevity of the plants. Overwatering leads to root rot, while underwatering causes wilting—detrimental to plant well-being. Our project, the Plant Guardian, addresses these issues by providing a smart, automated watering solution designed to simplify plant care. The Plant Guardian integrates sensors that monitor soil moisture, temperature, and humidity, ensuring plants receive the right amount of water at the optimal time. This system helps prevent common plant care problems by automating watering based on real-time conditions, enhancing both plant health and ease of care. With a user-friendly mobile app, the Plant Guardian enables users to monitor and manage their plants from anywhere. The app allows users to set watering schedules, view sensor data, receive alerts for potential issues, and even remotely check on plants via a live-streaming feature. This functionality makes it ideal for frequent travelers or busy individuals who want to ensure their plants thrive without constant supervision. By merging technology with plant care, the Plant Guardian fosters sustainable indoor gardening practices and provides a convenient solution for houseplant enthusiasts. This innovation empowers users to enjoy the many benefits of houseplants while simplifying daily care routines and enhancing their connection with nature.

This innovation empowers plant owners to maintain healthy, vibrant plants while fitting seamlessly into their routines, promoting sustainable living and enhancing well-being.

II. System Components

Our smart water planter project aims to deliver reliable, efficient plant care through advanced, integrated features. Key functionalities include a remote-controlled water pump, operated via Bluetooth or Wi-Fi, which allows users to water plants from a distance.

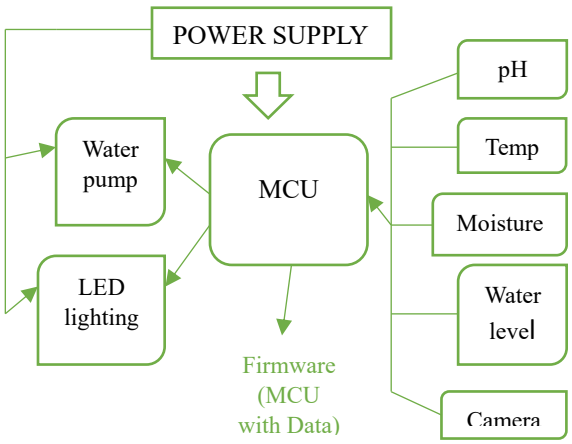

Fig. 1. Hardware Diagram

Additionally, energy-efficient LED lights designed for plant growth provide the necessary light spectrum for photosynthesis, supporting healthy development even in low-light indoor environments.

The block diagram in Fig. 1 provides an overview of the hardware system architecture, highlighting the main components and their connections within the smart plant watering system. At the center is the microcontroller unit (MCU), which serves as the

primary control hub, receiving data from various sensors, including pH, temperature, moisture, water level, and a camera. Each sensor feeds real-time data to the MCU, enabling accurate monitoring of the plant's environment.

The power supply provides the necessary energy to all components, ensuring reliable operation. The MCU processes the data and activates output devices, such as the water pump and LED lighting, based on the plant's needs. The firmware in the MCU controls the data flow and automates the responses, ensuring optimal plant care by adjusting water and light levels as required. This integrated hardware setup forms an efficient and responsive system for maintaining plant health.

A. Microcontroller

The ESP32-S microcontroller was selected to support core functionalities, including data processing, sensor integration, and remote communication. Initially, we evaluated alternatives like the MSP430 and Raspberry Pi 0 W; however, each presented limitations for our requirements. The MSP430 lacked sufficient processing power for image streaming, while the Raspberry Pi 0 W's high power consumption and integration complexities made it unsuitable for our PCB design. This evaluation clarified the need for a microcontroller that could efficiently manage real-time data and provide reliable connectivity for remote access. The ESP32-S met these needs with its dual 32-bit processors, providing ample power to manage streaming from a connected camera and handling data from multiple sensors without lag. With Wi-Fi, Bluetooth, and BLE capabilities built in, the ESP32-S allows seamless remote control and monitoring through our mobile app. Its extensive GPIO pins simplify integration with the system's components, including the water pump, LED lights, and environmental sensors, ensuring a flexible and streamlined design.

B. Sensors

The sensors integrated into the PCB include a range of components designed for reliable plant care. The pH sensor used is the GAOHOU PH0-14, selected for its accuracy and cost-effectiveness in measuring soil acidity. The temperature monitoring is handled by the LM35DZ, chosen for its precision in tracking environmental conditions. The water level sensor, A02YYUW, is an ultrasonic sensor that helps monitor water levels in the reservoir, ensuring the plants receive the right amount of water. These sensors are carefully integrated into the PCB to ensure compactness and seamless operation. The moisture sensor chosen is the Adafruit STEMMA, which is known for its accuracy in measuring soil moisture levels. All these sensors are connected directly to the PCB, allowing for easy data collection and integration with the microcontroller. Each sensor has been calibrated for accuracy, with the data from the moisture sensor mapped to real-world moisture levels, ensuring effective plant care.

C. LED lighting

LED lights were chosen for their efficiency and ability to support optimal plant growth. We selected the Commercial Electric LED strip light kit, which includes four 12-inch linkable segments and operates at just 8.5 watts. This energy-efficient lighting system is ideal for indoor plant care, providing the right light spectrum for photosynthesis while minimizing energy consumption. Unlike traditional bulbs, the flexibility of LED strip lights allows for custom placement around plants, ensuring even light distribution and improved coverage for healthy growth. The warm white light produced by the LED strips mimics natural sunlight, creating an environment that supports plant health without generating excessive heat.

D. Water pump

The water pump chosen was DC 3-5V Micro Submersible Mini Water Pump, serves as the core mechanism for delivering water to plants with precision and efficiency. This pump's low voltage and current range (150-250mA) enable it to operate with minimal energy consumption, aligning well with the system's goal of sustainability. Its quiet operation is particularly suited for indoor use, ensuring it doesn't disrupt the indoor environment. Additionally, the pump's compact design fits neatly into the 1-gallon reservoir, saving space while maintaining an unobtrusive aesthetic within the planter. Incorporating this water pump enables automated plant hydration, controlled through real-time data from sensors and a mobile app. The pump responds directly to soil moisture readings, activating only when water is needed, thus avoiding overwatering or underwatering. Users can also manually control the pump remotely via Bluetooth or Wi-Fi, which is particularly helpful for those with busy schedules or frequent travel.

E. Wireless Communication

WiFi (802.11n) was selected as the primary communication technology due to its high data rate, making it ideal for real-time streaming of images and sensor data. Its moderate range suits indoor environments well, and integration with internet connectivity enables remote monitoring and control from any location with internet access, adding a layer of convenience for plant care management even during extended absences.

F. Camera

The OV2640 camera module was selected for this project due to its balanced performance and practical advantages. Offering a 1600p resolution, it meets the requirements for clear, detailed plant imaging while maintaining manageable bandwidth demands. This module is also widely available in development kits, minimizing both prototyping costs and setup time, and providing comprehensive documentation and community support. While the OV5640 was considered for its versatility across resolutions and frame rates, the OV2640's efficient power usage and lower data rate requirements made it the better fit for this application. Its specifications support consistent image quality with minimal strain on the system's processing and communication resources, ensuring reliable operation in a power-sensitive environment.

III. System Concepts

The User Initiation & Navigation process begins when the user opens the "Plant Guardian" app, which takes them to the main dashboard or homepage. Here, they can use navigation icons to access specific sections, including the Data Page, Camera Page, Water Page, and Settings Page. This setup provides a user-friendly interface for quick access to key plant management functions, allowing the user to navigate between pages efficiently and adjust as needed.

At the system's core are the Server (MongoDB/Atlas) and Firmware (MCU with Data), which work together to store and process information. The server stores all plant and sensor data as well as the app's settings, providing a central repository for the application's data. Meanwhile, the firmware on the microcontroller unit (MCU) gathers real-time data from sensors attached to the plant and manages hardware functions, such as lighting and watering. Together, these components enable reliable data storage and device control, forming the technical backbone of the system.

The Data Page displays current and historical sensor data that has been collected by the firmware and stored in the server. This page is vital for the user, as it provides them with essential information on the plant's environment, including soil moisture, temperature, and pH levels. By monitoring these statistics, the user can gain insights into the plant's needs and make informed decisions to maintain optimal growing conditions.

The Camera Page allows users to view a live video feed of the plant, thanks to the Camera Capture feature. This page offers a visual monitoring function, letting users check the plant's condition directly. Additionally, the Toggle Lighting Button within this page lets users control the lighting setup remotely, which is particularly useful for adjusting light exposure in real time to support the plant's growth requirements.

On the Water Page, users can manually activate watering through the Manually Click to Water Button. This feature gives users control over hydration by allowing them to water the plant based on the sensor feedback they observe on other pages. The Water Page works with the Light/Pump Control feature in the firmware, providing manual and automated water management options to support a flexible watering routine.

The Settings Page enables users to customize sensor thresholds for factors such as soil moisture and temperature. With the User Set Values for Each Sensor function, users can define specific values for each environmental factor based on the plant's requirements. This customization ensures that the automated responses, like turning on lights or activating the water pump, are tailored to meet the needs of different plant types, enhancing the system's versatility.

The Control Functions (Hardware) include essential hardware actions managed by the firmware, such as toggling the lighting and activating the water pump. These functions respond to both user commands and sensor readings, allowing the app to execute automated responses when certain thresholds are reached. This setup bridges the gap between software and physical devices, enabling the system to create a balanced environment for the plant without requiring constant user oversight.

Finally, the Real-Time Feedback loop ensures that all actions taken in the app, such as adjusting settings or manually watering, are immediately reflected in the system. Every interaction with the hardware sends updates back to the Server and Firmware, which then refreshes the sensor data displayed on the Data Page. This continuous feedback loop allows the user to stay

informed about the plant's status and verify that their actions have been successfully implemented, supporting an interactive, real-time plant care experience.

IV. Hardware Details

This section goes into further detail the PCB and integration of components into it. We designed our PCB using EasyEDA. We chose EasyEDA due to easy web access as well as their integration with JLCPCB. Our PCB has gone through 3 revisions with 2 PCBs being created. Both were manufactured by JLCPCB with minor components sourced from their catalogue. Our first design had issues due to the camera connector being too small to be soldered. So, we had to move to a design with the camera and ESP32 on a dev board.

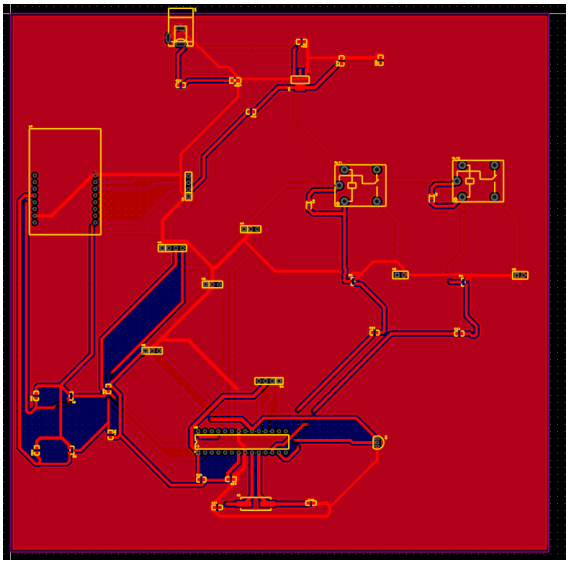

Fig. 2. First Manufactured PCB

The second PCB, which was the first to be manufactured, worked but required modifications. The ESP32 dev board pins had 3.3v routed to the wrong pin. It also had a reset pin hard stuck to ground causing infinite resetting loop. Lastly, it suffered from needing a bypass capacitor on the temperature sensor and a low pass filter on the data pin for the temperature sensor. Our last PCB design successfully rectified all issues faced in the previous versions and is discussed further below.

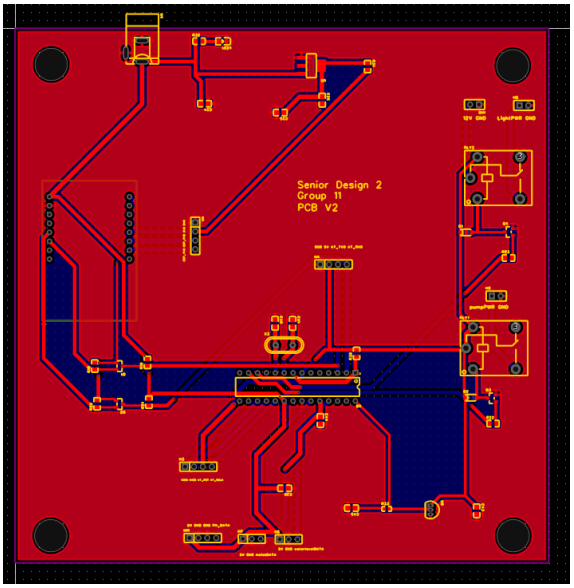

Fig. 3. Final PCB

A. MCU

Our MCU subsystem consists of two microcontrollers. The ESP32 and the ATMEGA328-PU. The ATMEGA328-PU is in charge of handling most io as well as the processing required for cleaning up this data. In order to use this processor at 16Mhz an external 16Mhz oscillator is required. This oscillator. It can be used in an 8Mhz configuration without the external oscillator, however for our use case we need the additional speed. The ATMEGA328 also takes 5v as an input. Our ATMEGA328 is mounted via a DIP connection so that we can flash it via an Arduino UNO for ease of development. In order to do this the bootloader must already be flashed.

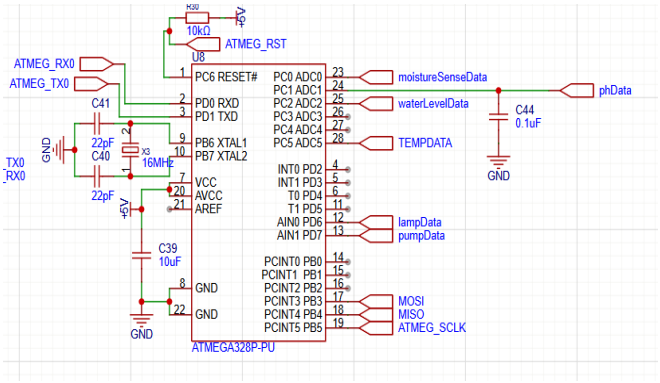

Fig. 4. ATMEGA328-PU and sensor connections

The ESP32 is used for our camera connection and for interfacing with Wi-Fi. Due to our soldering limitations the ESP32 is on a dev board with the

camera. To communicate between the two they are connected via UART. The ESP32 takes a 3.3V input. Due to the ESP32's GPIO being 3.3V and the ATMEGA328-PU 5v, we must convert the signal voltages. This is done via a bi direction logic level shifter made up of 2 10k ohm resistors and a mosfet.

There are also header pins to access the UART connection to the ESP32 and ATMEGA328 for debugging as well as their reset pins.

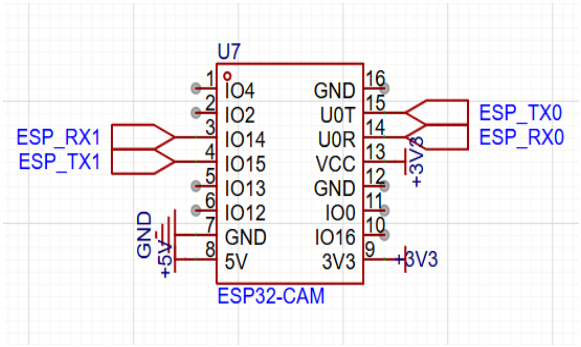

Fig. 5. ESP32 Dev Board

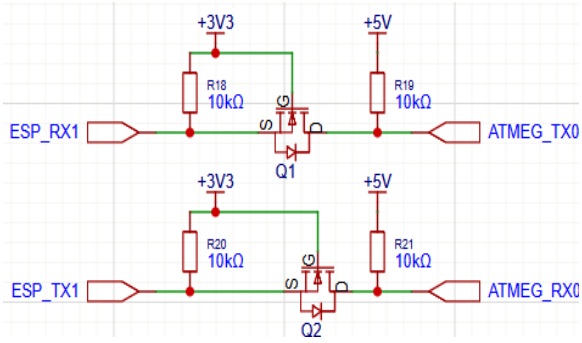

Fig. 6. 3.3v to 5v Bidirectional Level Shifter

B. Sensors

The moisture sensor is connected to our ATMEGA328-PU via our ADC0 pin. As well as takes a 5v and ground connection. This sensor goes into the soil of the plant and informs us the moisture content. This sensor was tested to see what water levels correspond with dry and wet to offset and sclae soil moisture. The next sensor is the water level sensor which is similar to the moisture sensor in implementation. It connects into the ADC2 port of the ATMEGA328 and takes in a 5v and ground connection. This gives us an analog value that can be mapped to the water level's height by measuring when its at its lowest and when it is at its highest measurement. Then the PH sensor is connected to the ADC1 port. We added a 0.1 microFarad capacitor to filter the input due to a known issue from the manufacturer regarding noise. We also needed to calibrate this sensor via the adjustable resistors on the sensor itself to ensure proper readings. This is done by first having a neutral PH solution and putting the sensor in it and adjusting until it measures 7. Then we get an acidic solution of 4 and calibrate the sensor until it is at 4. Finally we have the temperature sensor, this one is connected to the ADC5 port on the ATMEGA328. This also has a bypass capacitor across the 5v input and ground input to ensure stable power as well as a low pass filter to keep noise out. We also found that this sensor needed to be offset by 2 degrees. The last thing worth mentioning is that the ATMEGA ADC ports are actually just one multiplexed ADC. This means that high impedance sensors can have issues reaching a proper reading if switching between them too quickly. This leads to issues with our temperature sensor, to rectify this we wait a second before reading the temperature sensor.

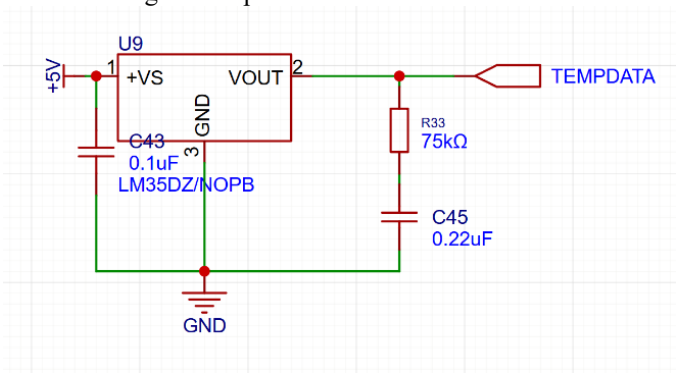

Fig. 7. Temperature Sensor

C. Power

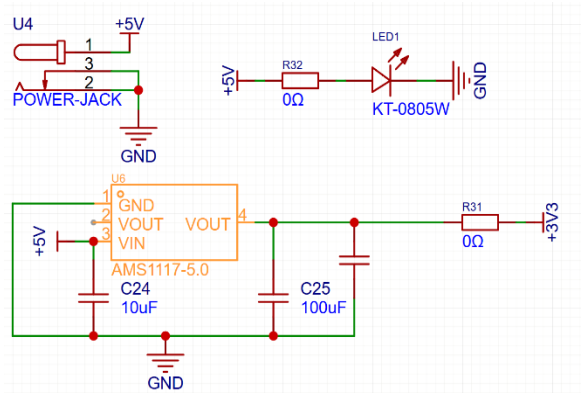

Fig. 8. Power jack, status, and linear regulator

The voltages required for our circuit are 5v and 3.3v. Most of our circuit uses 5v with only the ESP32 and the level shifter converter using 3.3v. For this reason

our AC to DC wall adapter inputs 5v 2Amp power via a barrel jack connector. This leaves us with the requirement of 3.3v. To achieve this we used an AMS1117 linear regulator. While these regulators are inefficient for high voltages our system pulls at max 600mA at 3.3v and our linear regulator can supply 1A.

D. Outputs

Our two outputs are the light and the water pump. They are both implemented via a relay circuit. This allows us to take our digital signal from the ATMEGA328 and use it to turn on and off a source with more power. Our circuit uses a BJT to trigger the relay on and off. It also has a flywheel diode to prevent damage. The water pump uses 5v so it pulls its power from the 5v on the circuit. However the light circuit needs 12v. So we have an external input for 12v.

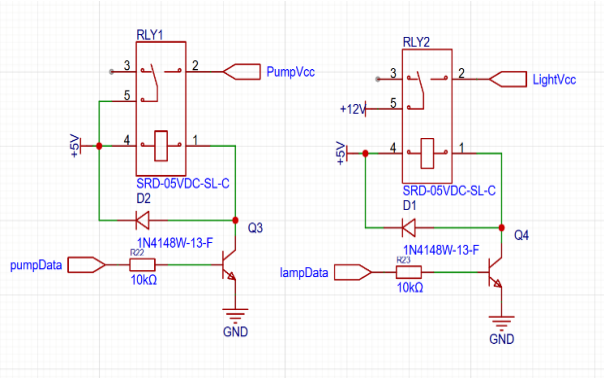

Fig. 9. Light and Pump relay circuits

V. Software Overview

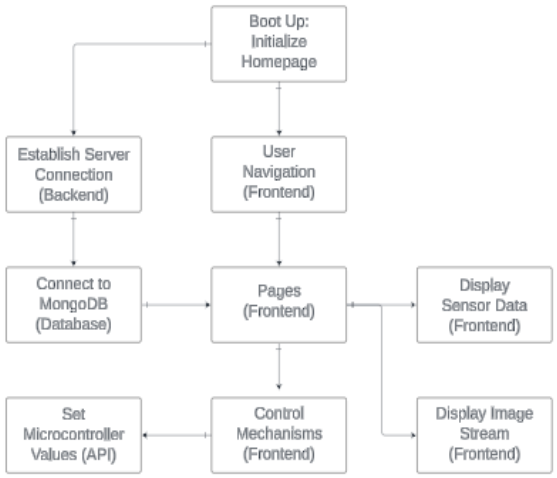

Fig. 10. Software Flowchart

The Smart Plant Guardian System is powered by a MERN (MongoDB, Express, React Native, Node.js) stack architecture, designed to enable real-time monitoring, user control, and automated management of plant care parameters. The software's primary objective is to provide a responsive and interactive mobile application interface that allows users to oversee and adjust critical environmental factors affecting plant health, including soil moisture, temperature, water level, and pH levels. The figure below introduces the project's software flowchart and visualizes the interactions between multiple aspects. In the following sections, each part of the MERN stack that the flow chart is comprised of is explained in further detail.

A. The MERN Stack

The MERN stack provides an efficient and cohesive framework for real-time plant monitoring and control. MongoDB serves as the database, handling data related to environmental conditions: moisture, temperature, water level, and pH. Express and Node.js form the backend API layer, which facilitates communication between the frontend and both the database and the hardware (ESP32 microcontroller). The backend handles data storage, retrieval, and control commands, providing endpoints for monitoring sensor data and managing manual controls for the water pump and lighting with built-in safety features. React Native is used to build the mobile frontend, enabling a responsive and cross-platform interface where users can monitor conditions and adjust control parameters.

B. Frontend: React Native Application

The frontend of the Smart Plant Guardian System is built with React Native, enabling a cross-platform mobile application that runs on both Android and iOS devices. React Native's component-based architecture facilitated the creation of a modular, efficient, and maintainable codebase, making it easier to optimize performance. The design process for the frontend was initiated in Figma, a versatile UI/UX design tool. Figma was used to create high-quality mockups of the application's interface, focusing on user-centric design principles to ensure the app is both visually appealing and functional. The Figma prototypes allowed for thorough design iteration, enabling feedback and revisions before development began. The result was a

mobile interface that offers a range of intuitive functionalities that allow users to monitor real-time data, view desired levels, and control the physical system.

C. Backend: Node.js and Express API

The backend, built with Node.js and Express, acts as the intermediary between the frontend and the MongoDB database, as well as between the frontend and hardware commands sent to the ESP32 microcontroller. The backend's core functions include:

Data Handling and Storage: The backend retrieves real-time data from the ESP32 microcontroller and stores it in MongoDB, where it can be accessed and processed. Each environmental reading—soil moisture, temperature, water level, and pH—is stored updated routinely every minute.

API Endpoints for Data Retrieval and Control: The backend provides a RESTful API with endpoints for retrieving sensor data from MongoDB, ensuring users always see the latest values.

Manual Control of Hardware Components: The backend includes endpoints to toggle the water pump and lighting. Specifically, each hardware component can be managed with a state variable that switches between two values to represent on and off. Each endpoint handles a toggle command and applies logic to ensure that the device automatically resets to an "off" state after a specified time or condition, providing safe control functionality without needing the user to manually turn it off.

D. Database: MongoDB

The data bank MongoDB serves as the primary storage for real-time data related to the plant's environmental conditions. MongoDB's flexible, schema-less design makes it a vital component for handling the dynamic data requirements of this project. It excels in managing groups of data, which is crucial for storing frequent sensor readings in a way that remains efficient over time. This design enables the system to track and maintain large volumes of data without compromising performance, as MongoDB is optimized for rapid write and read operations. The collections in MongoDB include: sensor data collections, control state collection, and an images collection. Each collection is grouped by their roles for efficient API management.

E. Functionalities Enabled by Software

The software provides a variety of functionalities aimed at enhancing plant care management. Key functionalities include real-time monitoring: through continuous data retrieval from the ESP32 microcontroller and periodic updates to the frontend, users are always presented with the latest environmental readings. Configurable thresholds: users can set desired levels for each parameter, allowing the system to adjust its operation based on individual plant needs. For instance, if a user sets moisture to "High," they can expect the app to recommend or automate watering actions as needed. Manual controls with safety automation: users can toggle the water pump and lighting manually through the app. The backend enforces safety by automatically reverting these settings after a specified duration. This helps prevent unintentional overwatering or excessive lighting.

F. Firmware

The firmware is made up of two parts. The ESP32 firmware and the ATMEGA328 firmware. They are both programmed in C++ via the Arduino IDE. The ESP32 is responsible for Wi-Fi communication and handling the images. The ATMEGA handles all other sensors and outputs including averaging data.

First the ESP32 initializes two serial ports. One to the ATMEGA and another for debugging. Then it connects to Wi-Fi and then connects to our websocket server on the backend. After setting up a heart beat pulse to ensure its still connected it initializes the camera. This is where our image quality is decided, we initialize for an XGA image with compression. Then it runs our websocket switch statement that processes inputs from the server. If it recieves any it passes it along to the ATMEGA via UART. Finally it takes an image and uploads the image as a binary and the sensor data from the ATMEGA as text to the backend.

The ATMEGA328 firmware first sets up a serial connection to the ESP32. Then it initializes all the output pins for use as output. Then it checks if any serial information telling it to trigger the light or pump has been sent yet. If so it'll set the corresponding pin to high. Then it goes through all the sensors averaging their data for better consistency. Due to the ADC limitation before reading the temperature sensor it waits a second after the multiplexer changes to allow the voltage to hit steady state. Otherwise, it will report inaccurate readings. Finally, it sends this data to the ESP32.

VI. Conclusion

The "Plant Guardian" system demonstrates a practical and effective solution for automated plant care, combining hardware and software elements to streamline monitoring and maintenance. By leveraging sensors, microcontrollers, and a user-friendly app interface, this system provides real-time data on soil moisture, temperature, lighting, and watering needs, allowing users to optimize plant health with minimal effort. The integration of a MongoDB server ensures reliable data storage and access, while customizable thresholds allow for flexibility across different plant types and conditions.

This project highlights the potential of IoT in smart agriculture, where automation can reduce manual workload and improve outcomes for plant growth. With features like remote camera monitoring, adjustable lighting, and manual watering controls, users can tailor the plant environment to specific needs, whether they are at home or away. Overall, the "Plant Guardian" system exemplifies how modern technology can support sustainable and efficient plant care, with implications for broader applications in agricultural and home gardening contexts. Future developments could explore additional sensor types, data analytics for predictive care, and scalability for larger plant collections or agricultural fields.

BIOGRAPHY

Luz Romero is a 24-year-old Electrical Engineering student who recently completed a 4-month internship as a Project Engineer at Urban Electric. She aims to gain further hands-on experience in the field and pursue a master's degree to advance her expertise.

Steven Keller is a 22-year-old Electrical Engineering student with a minor in computer science. After graduation he is working on embedded software for spacecraft at the Johns Hopkins Applied Physics Laboratory and eventually get a masters in robotics.

Tony Chau is a 23-year-old Computer Engineering Student who recently swapped from his goal to be an electrical engineer. With a newfound interest in software design, he aims to expand his personal project experience, make up for the late start, and pursue a career in the field at a large company like Google and Microsoft.

Keven Hyppolite is a driven electrical engineer graduating from the University of Central Florida in December 2024, specializing in power and renewable energy. With a passion for sustainable innovation, he aims to gain industry experience, pursue a master's degree, and establish his own engineering firm to shape the future of renewable energy.

REFERENCES

"Digikey." *ESP32-CAM Development Board*, media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0602_Web.pdf. Accessed 23 July 2024.

ESP32 series datasheet. (n.d.). https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

ATMEGA328P. (n.d.-b). https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf