IoT Monitoring and Control System for Plant Irrigation

1st Hian de Almeida Santiago
IT department
Federal University of Paraiba
João Pessoa, Brazil
hianalmeidasantiago@gmail.com

2nd Georgia Pereira Brito
IT department
Federal University of Paraiba
João Pessoa, Brazil
georgiapereirab@gmail.com

3rd José Venancio de Oliveira Silva
IT department
Federal University of Paraiba
João Pessoa, Brazil
josevenancioliveira@gmail.com

4th Mirele da Silva Costa
IT department
Federal University of Paraiba
João Pessoa, Brazil
mirelecosta@eng.ci.ufpb.br

5th Raissa da Silva Vieira
IT department
Federal University of Paraiba
João Pessoa, Brazil
raissavieira@eng.ci.ufpb.br

6th Veronica Maria Lima Silva
Department of Computer Systems (DSC)
Federal University of Paraiba
João Pessoa, Brazil
veronica.lima@ci.ufpb.br

***Abstract*—This article describes an automatic irrigation system that allows the user to monitor several parameters related to the care of a plant, including humidity level, temperature and environmental lighting data. These ambient levels are measured using sensors. An irrigation system controls soil moisture and can be triggered automatically when a significant drop in these values is detected. Furthermore, the system includes the option to manually activate the irrigation or fertilizer dispenser if requested by the user, in addition to emitting ultraviolet radiation if the plant is not receiving enough natural light. The *IoT* system stores this information and can be controlled from anywhere. All of this aims to ensure a favorable environment to plant growth and ease for those who use it. Through a series of tests, the system to be consistent and easy to use, both independently and manually activated.**

***Index Terms*—IoT, Esp32, Firebase, Smart, Garden**

I. Introduction

In the busy daily lives of contemporary societies, connectivity and automation play increasingly essential roles, transforming not only humans, but also the way we interact with the environment. In this context, the relevance of projects that explore the integration of embedded systems in domestic environments stands out, especially in plant care. The balance between the practicality of automation and environmental responsibility becomes crucial as we seek innovative solutions to optimize cultivation in residential spaces.

In this work, an embedded monitoring and care system was developed for plants to grow in a domestic environment. The system proposes an integrated solution for automatic supervision and monitoring of various settings related to plant care, such as soil humidity, temperature and environmental lighting. The system also provides additional functionalities, such as manual watering control, fertilizer dosing and ultraviolet radiation emission, if necessary. The developed system contains a web platform, which user interface was built using the React [1] library developed in JavaScript. The Firebase [2] platform was used to store and synchronize data. Furthermore, the physical components of the project were initially tested using the Tinkercad [3] modeling tool before developing a final prototype with the ESP32 board.

II. Related work

Table I presents a comparative analysis of related work on smart gardens projects, highlighting smart irrigation systems that include sensor networks and automation in the plant irrigation process, among other components.

Choudhari et al. [4] presents a smart gardening system based on the Internet of Things (IoT), employing several sensors to monitor soil moisture, air temperature, lighting conditions and air humidity. The system incorporates a fertilizer dispenser controllable remotely via a mobile app or web panel. The irrigation system consists of a relay connected to the microcontroller, water pump, water source and separate power supply. The fertilizer doser is controlled by a servo motor, also connected to the microcontroller.

Song et al. [5] proposes a system to automatically monitor the conditions of the plant's environment. Using the NodeMCU open source platform, it incorporates a relay module to control high-current appliances, an air quality sensor sensitive to various gases and smoke, a soil moisture sensor to trigger irrigation based on a reading of the water and a DHT11 sensor to adjust environmental conditions. around the plant. The project seeks to explore automation in gardening, offering an integrated and efficient solution for monitoring and caring for residential plants, but does not cover fertilizer dispensers.

Singh et al. [6] also explores soil moisture and solar intensity monitoring but does not have automatic irrigation solutions that rely on the user to carry out actions, and for this it uses alerts via SMS through the Twilio service.

Pereira et al. [7] addresses the evolution of agriculture with the integration of the Internet of Things (IoT). This article presents an IoT-enabled smart drip irrigation system using ESP32 microcontroller. The proposed system connects to the

TABLE I
RELATED WORK

Smart Garden Items	Specifications
IoT-based Smart Gardening System [4]	1 - Air humidity sensor 2 - Soil moisture sensor 3 - Brightness sensor 4 - Web application 5 - Mobile application 6 - Water dispenser pump 7 - Servo motor for fertilizer dispenser
A Study on IoT based Real-Time Plants Growth Monitoring for Smart Garden [5]	1 - Soil moisture sensor 2 - Servo motor 3 - LED 4 - Gas sensor 5 - Water dispenser pump
Smart Garden with IoT based Plant Monitoring System [6]	1 - Soil moisture sensor 2 - Remote data analysis
ioT-Enabled Smart Drip Irrigation System Using ESP32 [7]	1 - Soil moisture sensor 2 - Remote data analysis 3 - ESP32 4 - Temperature Sensor 5 - Air Humidity Sensor 6 - Water Flow Sensor 7 - Solenoid Valve
Iot and Artificial Intelligence Based Smart Gardening and Irrigation System [8]	1 - Rain sensor 2 - Temperature and humidity sensor (DHT11) 3 - Soil moisture sensor (V1.2)
Proposed system	1 - Air humidity and temperature sensor 2 - Soil moisture sensor 3 - Brightness sensor 4 - Web application 5 - Water dispenser with peristaltic pump 6 - Servo motor for fertilizer dispenser 7 - Reservoir water level monitoring

Blynk application, allowing the collection of irrigation data, manual irrigation control and visualization of graphs based on soil moisture, temperature, air humidity and water flow sensors. The ESP32 activates irrigation when necessary, ensuring an adequate supply of water to the plants. Additionally, the system alerts the user to extreme humidity conditions and offers the option to turn off automatic watering as desired.

Samira et al. [8] involves building a smart irrigation system using multiple hardware components like NodeMCU ESP8266, a hydraulic motor, battery, relay module, rain sensor, and multiple soil moisture sensors. An Android application and a NodeMCU framework control the system. Key goals include remote monitoring, streamlining gardening tasks, time-saving irrigation methods, frequent system updates for accuracy, and expansion to IoT, microcontrollers, AI, and other technologies for enhanced functionality. The system aims for efficient irrigation, minimizing water loss and improving the effectiveness of smart gardening.

The main focus of our system is to create a favorable environment for the plant and display the detailed sensors and actuators in a simple and user-friendly way. The web application provides information on the current state of the plant and options to activate the watering and fertilizing directly. As for the actuators and system architecture, we offer as many simple-to-find and inexpensive components as possible, delivering most of the functionality of the related article while being relatively simple and easy to replicate.

III. METHODOLOGY

This study was developed in distinct stages, aiming to achieve the proposed objectives of development and integration of the automatic irrigation system. In Fig. 1, we represent an information flow diagram in the architecture of our system, in order to represent how readings are taken in the plant and how the actuators act on it. The numbers in the image show how the data flow and the system behaves, as follows:

1) Data Collection: The sensors read the data from the plant and water reservoir.
2) Data Processing: The ESP32 processes the data into a JSON string.
3) Data Upload: The ESP32 sends the JSON string to the Firebase instance.
4) Data Visualization: When a new document is added to the database, the Web Application updates the graphs and real-time readings, and sends updates the data on the database if any of the actuators are manually activated.
5) Actuation Decision: The ESP32 checks if any of the actuator flags are set, and sends the signal to the respective actuators.
6) Actuation: Each actuator that had its flag set on the database goes through a code routine based on its functionality.

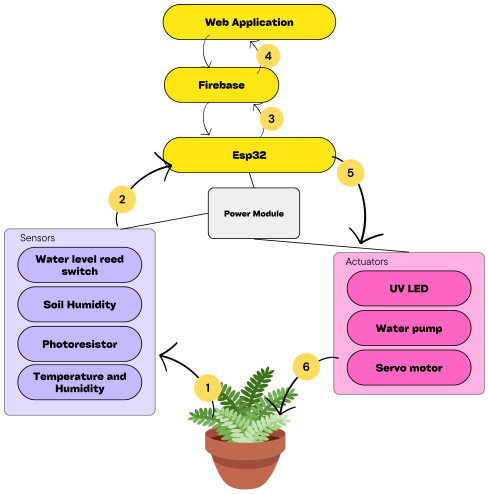

Fig. 1. System architecture

The flow of activities developed is explained in Fig. 2 with a diagram of all the steps from prototyping to the final integration of the Web platform with the hardware. Initially, a prototype of the irrigation system was developed using the Tinkercad online simulation platform. This step allowed the visualization and preliminary validation of the integration of

sensors and some actuators, serving as a testing stage for the following implementations.

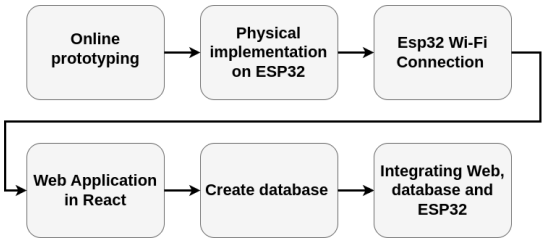

Fig. 2. System implementation flow diagram

The next step involves implementing the system's physical components on a breadboard and making appropriate connections using jumper wires. The components used were: ESP-WROOM-32, Micro Servo 9g SG90, Mini Water Pump 12V DC Motor - RS-385, Reed Switch Water Level Sensor, UV LED, 12v power supply, XL6009 Step Up Voltage Regulator Module, L298N Double H Bridge, DHT22 sensor and GC-58 Soil Moisture Sensor.

The communication protocol between the ESP32 and the Micro Servo, L298N Double H Bridge and LED components is PWM [9] (Pulse Width Modulation) which can send a specific value to the components based on its duty cycle. The DHT22 and GC-58 components use 1-Wire Serial Bus [10] as a protocol, which uses only one communication wire between a device and a peripheral.

Subsequently, we connected the ESP32 board to a local Access Point, and ran a few tests on how to upload the sensor data to a Firebase instance. This step was essential to enable communication between the system and the internet.

The web application was developed using the React library. This stage included creating a user-friendly interface for users to monitor and control the irrigation system remotely. The application allowed for visualizing sensor data, manually activating irrigation, fertilizer dispenser, and the LEDs.

The database used was Firebase's Real Time Database. It is a NoSQL cloud database that stores data in JSON format. The data was divided into two blocks, the first named "readings", as can be seen in Fig. 3, is responsible for storing the sensor readings. The second block of information, represented in Fig. 4, in the block named "leituras", records the actuator activation data.

The connectivity of the ESP32 with the database was done through the FirebaseESPClient library. The microcontroller sends the readings obtained by the sensors to the "readings" data set, and reads from the "leituras" data set to check if any of the actuators must be triggered, calling the actuator routine.

The Web Application is integrated to database through HTTPS calls to the Firebase API. Data from sensor readings are used to create monitoring graphs and tables. When the user activates some of the buttons to activate the actuator functions, a call is made to the bank to save the value 1 in "leituras", which will be interpreted and executed by the ESP32.

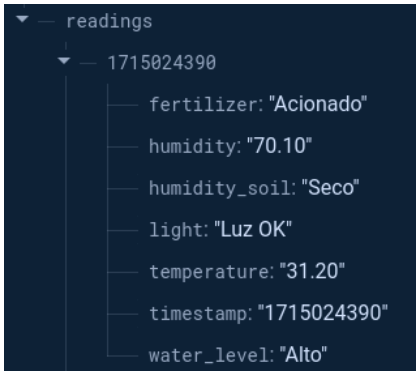

Fig. 3. JSON structure of the database with sensor reading data

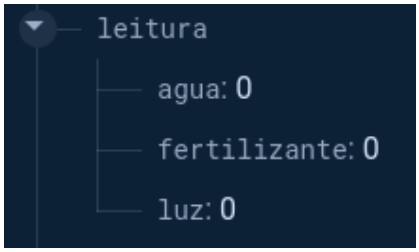

Fig. 4. Database JSON structure with actuator activation data

IV. Results

The final implemented system is represented in Fig. 5 in which it is possible to see the sensor connections on the NodeMCU microcontroller with the help of the breadboard. Fig. 6 shows a usage view with a real plant.

In Fig. 7 the first screen of the application is shown, most prominently in the "Minha Planta" section it is possible to see the last reading made by the sensors with data on temperature, ambient light level and humidity of the soil and air. In the "Cuidados Essenciais" section we have three buttons that trigger physical system functionalities, namely watering, fertilizing and turning on the UV light. An alert modal is displayed for the user informing the critical level of the water reservoir, the screen stops displaying this modal when the reservoir is full again.

The second screen of the web application is used to present sensor reading metrics. In Fig. 8 we have a graph that shows the temperature and humidity of the environment over time. In Fig. 9, a table is presented with records of soil moisture and ambient light level.

Tests of the smart gardening system were carried out at different times of the day to evaluate the variation in light and its influence on the system's automatic actions.

Additionally, we used different soil samples with varying humidity, from dry soil to completely wet soil, to test the system's ability to detect and respond to specific soil conditions by activating irrigation or not.

Readings were taken at 30-minute time intervals, with the aim of testing data saving and correct representation on the platform.