ECEN430: S.T.in CE (Internet of Things)

Final Project Report

Let it Grow (Smart Irrigation IoT System)

Presented to

Eng. Bishoy

Eng. Salma Zakzok

Prepared by:

Salma Hossam

Anas Ayman

Salma AlWardany

Table of Contents

Table of Figures and Tables

Idea:

Problem Statement:

Water is a critical determinant of plant health, playing a pivotal role in physiological processes such as photosynthesis, nutrient transport, and cell turgidity. However, the dichotomy of water provision—overwatering and underwatering—poses a significant challenge to plant survival and productivity [1]. Overwatering, characterized by excessive soil moisture beyond field capacity, leads to hypoxic conditions, impairing root respiration and facilitating the proliferation of root pathogens. In contrast, underwatering, marked by insufficient soil hydration, results in turgor loss, stomatal closure, and diminished photosynthetic activity, ultimately triggering desiccation and plant demise. Despite the prevalence of these issues among both domestic and agricultural settings, there remains a substantial gap in the empirical quantification of plant mortality attributable to suboptimal watering practices. Table 1 shows how each plant water need is different.

Table 1: water need for some plants.

Pumpkin	Tomato	Potato	Cactus	Peppermint	Strawberry
31.15 to 46.725 liters per 0.5 m^2 per week	31.15 to 46.725 liters per 0.5 m^2 per week	31.15 to 62.3 liters per 0.5 m^2 per week	15.575 liters per 0.5 m^2, applied every 4 weeks.	31.15 to 62.3 liters per 0.5 m^2 per week	31.15 liters per 0.5 m^2 per week

Project Idea:

Using the concepts of IoT, we worked on creating a smart irrigation project to solve the problem of underwatering or overwatering plants. The project depends on creating a connection between an open-source platform and an open-source application to monitor the moisture and temperature levels of plants and give signals to open the water pump if needed. This will save the plant from overwatering, in addition to decreasing wasting of water which happened in using certain intervals of time for watering in the normal watering systems as the sensor was stopped the pump when the plant is well moisture.

Methodology:

Main Components:

The smart irrigation system incorporates three key components: the Raspberry Pi with accompanying hardware, the Adafruit web dashboard, and the Blynk mobile app. This system offers two modes of operation, selectable via a mode button in the Blynk app. The automatic mode enables the pump to operate based on the soil moisture sensor readings, activating, or deactivating the pump when the moisture level crosses a predetermined threshold. In contrast, the manual mode allows users to control the pump directly through a button in the Blynk app as per their requirements.

On the Adafruit dashboard, users find two indicators: one displaying the current operation mode (manual/automatic) and the other showing the pump's status (ON/OFF). The dashboard also features a temperature gauge that changes color to alert users when the temperature exceeds or falls below set thresholds. Additionally, there are dual line gauges to monitor humidity and soil moisture levels. The system also includes a temperature indicator to manage the system's fan, turning it on or off as needed. Moreover, a battery level indicator is present to display the current charge level and to manage automatic charging.

The components used in this project include: RaspberryPi zero - Battery 12v 10000mAh - DS1307 RTC (Real Time Clock) for Raspberry Pi - BMS 3S (12.6V – 100A) Lithium Battery Protection Module - dc step down converter variable 3a 24v 12v to 5v 5a power module - Relay Module (12V-10A) 1Channel (Low level trigger) - Voltage and current sensor module consume - Waterproof Electrical Box IP65 (30(L) x 25(W) x 12(H) cm) - Battery Level Indicator Module 1S-8S (General Version) – pump - DHT11 – Soil moisture sensor.

Communication protocols:

The communication protocols used are https for requesting the button values from the Blynk application to the python script, and MQTT (Message Queuing Telemetry Transport) to communicate between the Adafruit server and the python script controlling the hardware. Also, the message queues IPC was used to send the button values between Blynk and Adafruit scripts to connect the 2 platforms together.

- **HTTP:**
  The python script initiates a connection with the Blynk server by using the IP after resolving it from the DNS, it then establishes a TCP connection with the Blynk server. After establishing the connection, the request of the python script is sent as packets to the Blynk server which then authenticates the sender and performs the required action (returns the button state) in our case.
- **MQTT:**
  MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol, in this project we used it in communicating between Adafruit and the python script controlling the hardware. This protocol was used to send data (sensor values and indicators) from the Python script (client) to Adafruit IO broker which is a server that implements the MQTT protocol and is hosted by Adafruit. The script publishes messages/values to a specific topic and each topics act as channels within feeds. The subscriber is used to receive data from the Adafruit to the python script.
  The TLS (Transport Layer Security) is used to encrypt data transmitted between devices and the broker. The MQTT uses both TCP/IP and utilizes a client-server model for communication. The python script acts as a MQTT client and initiates a connection to the MQTT broker, then handshaking and authentication are applied to authenticate the client using the username, password, and token. Then publishing and subscribing messages can take place where the broker is used to forward the messages to the correct place. The broker stores the sent messages temporarily and forwards them to the subscribed clients.
- **Message Queues:**

The message queues provide a way to exchange data without directly sharing memory. In this project it is used to communicate the value of 2 buttons from the blynk application python script to the Adafruit python script. The first step is queue creation, which is identified by a unique key and provides mechanisms for sending and receiving messages. The placing of data into the message queue initializes the sending of data. The receiver reads messages from the queue using the same key. Message queues are managed by the OS kernel, it handles the queuing, buffering, and synchronization of data between the processes.

Workflow:

We started by connecting the hardware sensors and pump to the Raspberry Pi, then we designed the Blynk and Adafruit platforms then we wrote 2 python scripts to link the platforms with the hardware. The first one interacts between Blynk (using an http request) and sends the button state (using sysvmq from the ipcqueue library) to the other python script which controls the hardware and communicates with Adafruit using MQTT, to end up with a fully connected system as shown in figure 2, and the flowchart is shown in figure 3.

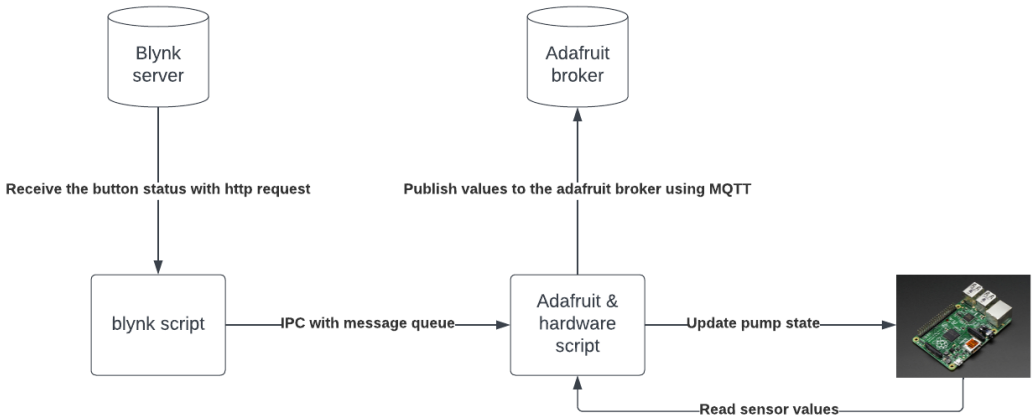

Figure 1 Project Workflow

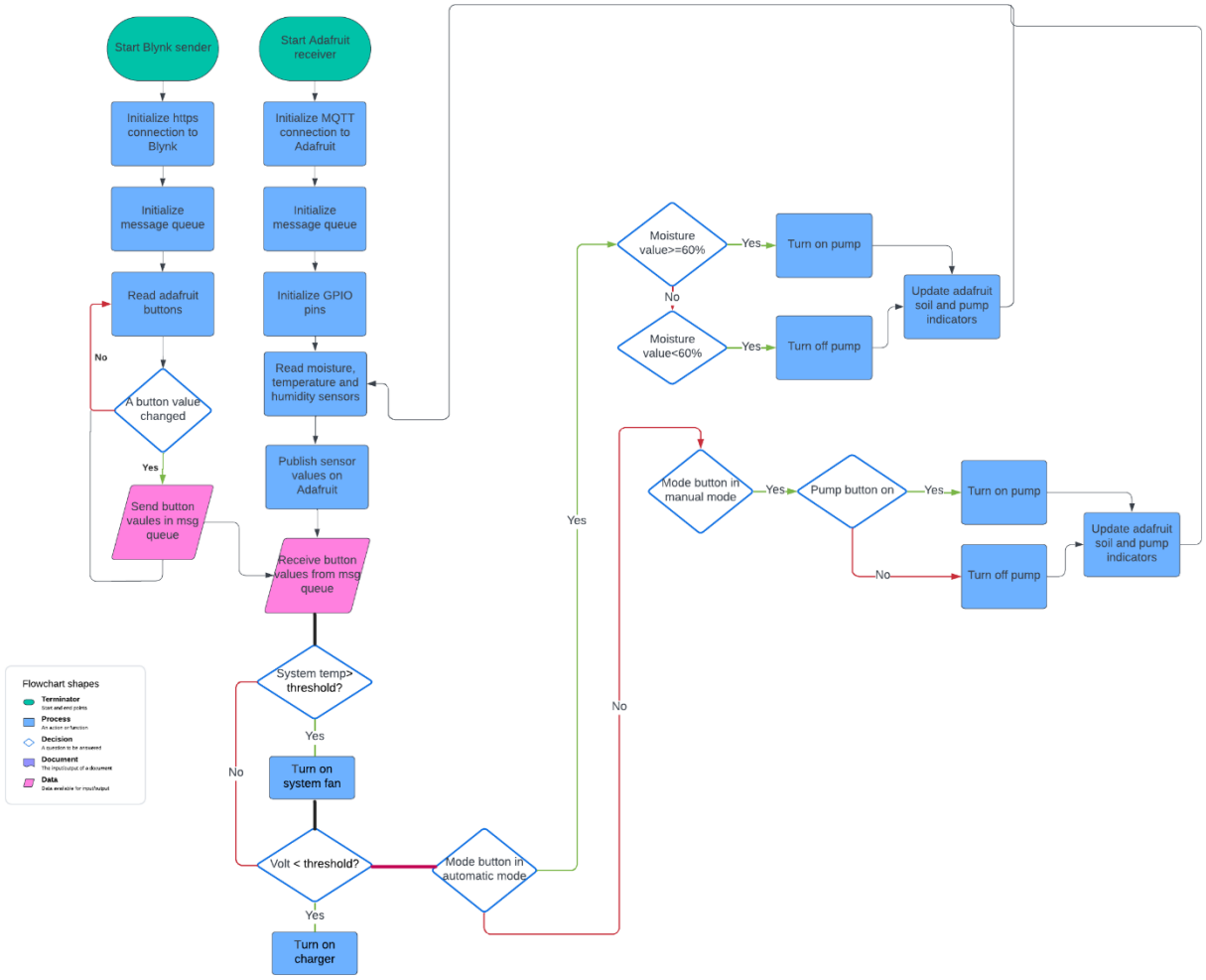

Figure 2 Flow chart

Results:

The following figures 4, 5 show the Adafruit and blynk designs used. And here is a link to the video of the working hardware: [ECEN430_Lab_Project_Smart_Irrigation_System.mp4]

The feed names in AdaFruit are: AutoCharge, BatteryCapacity, humidity, PlantType, Temperature, cpu_temp, fan_indicator, mode_indicator, pump_indicator, and soil_moisture.

In addition, figures 6, 7 show the real circuit and the tree of the files on Linux system.

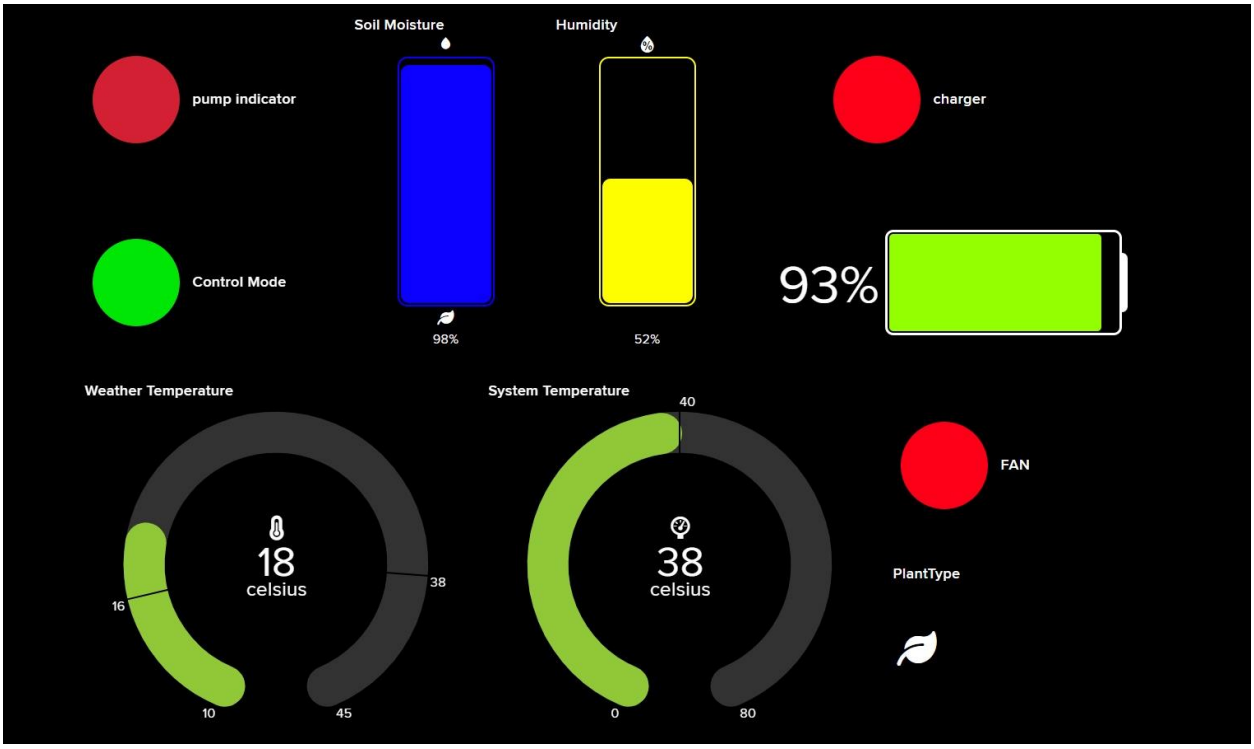

Figure 3 Adafruit Web Dashboard for Monitoring

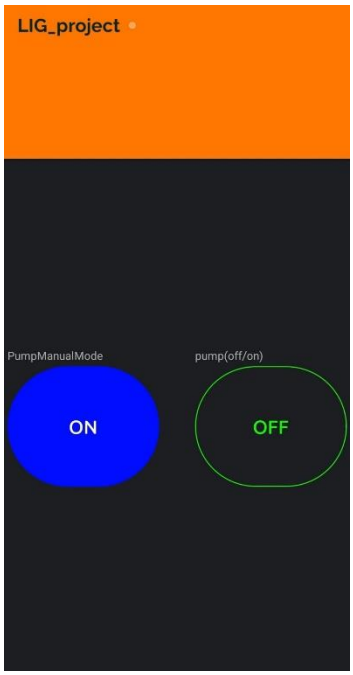

Figure 4 Blynk Mobile Application for Control

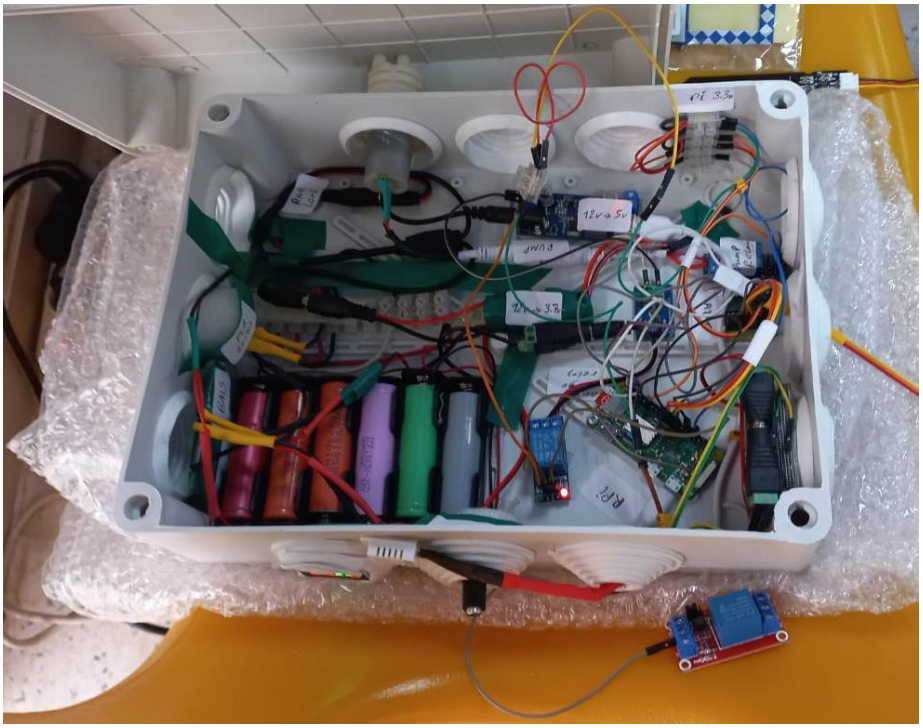

Figure 5: Project circuit

```
zonzo@pi:~ $ tree Smart_Irrigation_System/
Smart_Irrigation_System/
├── Actuators
│   ├── Act.sh
│   ├── charger.py
│   ├── fan.py
│   ├── pump.py
│   └── relay_f.py
├── Adafruit
│   ├── Adafruit_f.py
│   ├── Adafruit_main.py
│   ├── Indicators_send.py
│   └── Sensor_send.py
├── Blynk
│   └── Blynk_sender.py
├── run.sh
└── Sensors
    ├── ADC_sensors
    │   ├── ADC_f.py
    │   ├── ADC_main.py
    │   ├── Moisture_f.py
    │   └── Volt_f.py
    └── Digital_Sensors
        ├── Cpu_Temp.py
        ├── DHT_f.py
        └── DHT_main.py

6 directories, 18 files
```

Figure 6: tree of the files

The following table presents the cost of the components:

Table 2: Cost of Components

Components	Price
RaspberryPi Zero	2200 L.E
Battery 12v 10000mAh	250 L.E
DS1307 RTC (Real Time Clock) for Raspberry Pi	
BMS 3S (12.6V – 100A) Lithium Battery Protection Module	210 L.E
dc step down converter variable 3a 24v 12v to 5v 5a power module	150 L.E
Relay Module (12V-10A) 1Channel (Low level trigger)	30 L.E
Voltage and current sensor module consume	100 L.E
Waterproof Electrical Box IP65 (30(L) x 25(W) x 12(H) cm)	110 L.E
Battery Level Indicator Module 1S-8S (General Version)	60 L.E
pump	75 L.E
DHT11	48 L.E
wiring	70 L.E
Soil moisture sensor	75 L.E
Total	3378 L.E = 110 $

In the following table you can find a comparison between this example of irrigation project and others in the market:

Table 3: comparison between our project and the similar products in the market

	Our project	Kollea Reliable Automatic Watering System [2]	RAINTPOINT Automatic Watering System [3]	MARS HYDRO Auto Drip Irrigation Kits [4]
Cost	140 $ (this is the cost + the profit)	50 $	55 $	140$
Number of plants	8 plants with some modifications	8	8	8
Automatic	✓	✓ It waters on intervals of time	✓ It waters on intervals of time	✓ It waters on intervals of time
Manual	✓	✗	✓	✗
Sensors	✓	✗	✓	✗

As result, we can say that our project can be used efficiently to save plants, water, and cost, and it also has the ability for scaling and covering soils indoor and outdoor as it doesn't have restrictions on working outdoor, as the watering depends on the reading of the sensor.

References

[1] B. website, "The five most common reasons plants die," 2020. [Online]. Available: https://botanium.se/blogs/the-growing-guide/the-five-most-common-reasons-plants-die.

[2] "Kollea Reliable Automatic Watering System," [Online]. Available: https://www.amazon.com/Kollea-Automatic-Watering-Irrigation-Programmable/dp/B08LK7VH9P/ref=sr_1_8?crid=8CC9VS0T9SBJ&keywords=self+watering&qid=1702693276&sprefix=self+watering%2Caps%2C935&sr=8-8.

[3] "RAINTPOINT Automatic Watering System, Plant Self Watering System Automatic Drip Irrigation Kit with Pump, Indoor Irrigation System for Potted Plants, APP & Voice Remote Control (2023 Release, V2)," [Online]. Available: https://www.amazon.com/RAINPOINT-Automatic-Watering-Irrigation-Control/dp/B0BN8VDS3G/ref=sr_1_29?crid=8CC9VS0T9SBJ&keywords=self%2Bwatering&qid=1702693276&sprefix=self%2Bwatering%2Caps%2C935&sr=8-29&th=1.

[4] "MARS HYDRO Auto Drip Irrigation Kits Garden Watering System for Indoor, Lawn, Greenhouse, Yard, 5-Gallon Bucket 22W Water Pump with 8 Drip Emitters," [Online]. Available: https://www.amazon.com/dp/B0BPM3PDK1/ref=sspa_dk_detail_0?psc=1&pd_rd_i=B0BPM3PDK1&pd_rd_w=ee1Kh&content-id=amzn1.sym.eb7c1ac5-7c51-4df5-ba34-ca810f1f119a&pf_rd_p=eb7c1ac5-7c51-4df5-ba34-ca810f1f119a&pf_rd_r=2EYR080KCJCB2YTRMVE2&pd_rd_wg=mkUTH&pd_rd_r=d49.