Riley Mark Gatto 224554695

Project Artifact Report Plant Watering System

Project Overview:

Background:
Due to the evolving technological world smart watering systems have become more accessible to consumers. For many people around the world a smart way to water and take care of their plants is thought after. Manual watering systems have become outdated and time consuming so people are looking for alternatives to take care of their plants. So the main point of having a smart watering system is the plants do not need constant supervision and if owners are away they can still be sustained and properly taken care of. Having a watering system allows us to minimise the human input meaning we can avoid errors which could arise from over watering or under watering.

Existing work:
Many of the existing implementations are single device solutions which only use one device such as the arduino. This usually means that there is no graphical interface for flexible control and environment visualization. Due to most implementations only using a single device this usually means it is unlikely there will be manual and automatic watering modes with real time solid moisture feedback. This is where my system shines as it provides a GUI interface which can be accessed from a long distance away from the watering system. So for example if you have the watering system set up outside or a hard to get to area you can then connect the raspberry pi device Up to a monitor in your house and control it in a more comfortable environment. This takes away the hassle of having to go out and actually be in the same environment as the system.

Problem Statement:
Manual watering systems can lead to negligence and incorrect watering of the plant. So my system is designed to use environment collected data to decide the correct amount of water to give to the plant. Through My GUI I will be able to display the data and also switch between auto and manual modes based on the scenario which will lead to a more effective plant watering system. So minimising human input will lead to a computer deciding and calculating how much water a plant needs as this is more fault tolerant than a human manual watering a plant. Through the communication between two devices I will be able to effectively store and monitor my environment and provide real world actions to succeed in the goal of maintaining a plant's health.

Requirements:
These are the following requirements to ensure my system will effectively care for a plant by providing the correct amount of water. First I will need a soil moisture sensor which will detect the moisture of the soil. I will need a water pump, when turned on by the arduino, pumps the correct amount of water to the plant. I will need an arduino microcontroller which can decide when to turn the relay on to turn the pump on and also use MQTT to send and retrieve data from the MQTT server. I will need a Relay which can turn on and off the water pump depending on the signal from the arduino (HIGH / LOW). Next I will need a raspberry pi which can run the MQTT server and the GUI which has the automatic and manual options, as well as showing the moisture level. I need to use the MQTT protocol to send data between the two devices. Lastly I need to add a constraint so my system does not water under any circumstance if the soil is too moist even if on manual mode.

Riley Mark Gatto 224554695

Design Principles:

The first design principle I have implemented is Modular programming. I have made clear separations between the tasks that are done by the arduino and the tasks done by the raspberry pi. Further I have created individual functions for key components of my systems. This was important as it made my program code more clear and modularized. Through separating my system up into individual functions I was able to focus on one section at a time and did not have to change my whole code to add new features. Overall modular programming allowed me to effectively implement constraints such as checking the moisture level before watering which insured plant would not be over watered.

The second design principle I implemented was the Polling programming. I implemented polling as I believed it benefited my system the most. Through pollign I was able to check the moisture level and then based on this decide what actions to take. This worked best for me as I could take a more simple approach to my program and keep everything running in a loop. Polling in my system positively impacts my system as it is simple, comparable and predictable which can make my system more cross compatible when running in different environments.

The last Design principle I implemented was the constraints and fault tolerant aspects. My System implements many constraints to ensure the system runs as expected and does not result in incorrect watering patterns or readings that will impact our domain. The main constraints I have implemented is if we can not connect to wifi or the MQTT server I can run the system just on the arduino. This is done by switching to local automatic mode in which we read the data in from the sensor and water the plant based on this. This introduces fault tolerance as we do not need to rely on the communication between the two devices for example if the raspberry pi breaks or is removed from the environment. Further I have implemented a constraint on the arduino to check before watering that the soil moisture is not below the threshold, which stops any chance of over watering. Lastly each time the user switches between auto and manual mode I check to make sure the pump is off by turning the relay off every switch to avoid watering exceeding the time limit.

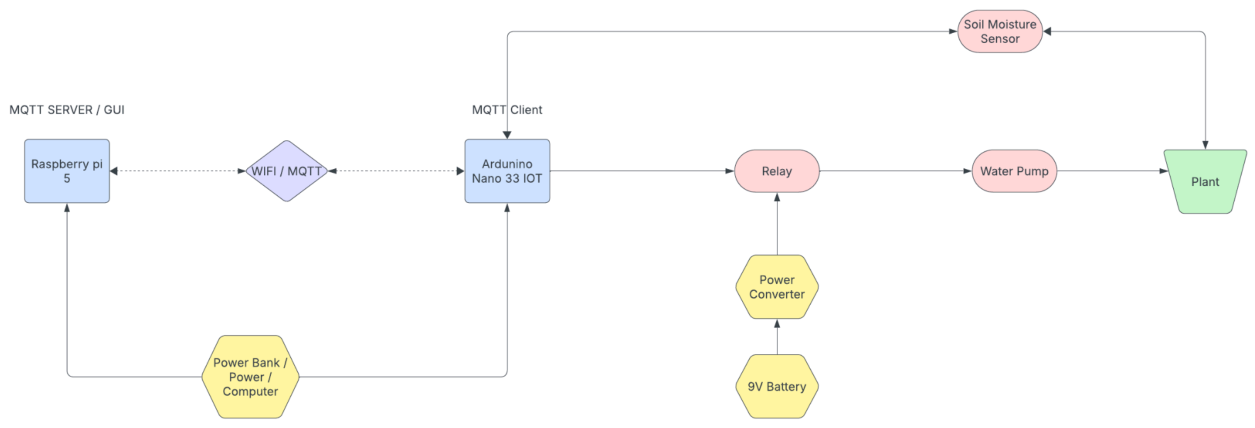
Prototype Architecture:

Hardware:
For my prototype I am using an Arduino nano 33 IOT and Raspberry pi 5 for my devices. The Arudino will be used to handle the sensor and pump through a relay as the pi will be responsible for hosting the MQTT server and GUI. I then have my soil moisture sensor and water pump, which will be responsible for reading the moisture levels and pumping water into the plants soil. I then have my relay which will be controlled by my arduino which either turns the water pump on or off by cutting the connection between the water pump and 9v Battery. Next I have My power sources which can either be the power bank, wall outlet, 9V battery connected to power converter module or other power source to power both the arduino ,pi and water pump. So these are the hardware architectures which are necessary for my project to run efficiently.

Riley Mark Gatto 224554695

Software:
For my software architecture my arduino needs to read the moisture and then send it the raspberry pi using MQTT. The raspberry pi must also connect the GUI to the MQTT server running on the same device. Further the user needs to be able to interact with the GUI and switch between auto and manual mode and also receive information such as moisture levels. Data is transferred by each device by sending messages using the MQTT protocol to the MQTT server either to get information or receive information. This software architecture is important as it allows both devices to subscribe to the MQTT server and query or add information to each topic of the server. This way the server acts as the mediatry passing and storing information for each device to communicate with each other. Through this I am able to control my system through the GUI running on my raspberry pi.

Below is a diagram of my prototype architecture:



Link To Prototype Code On Github:

Task Link:
https://github.com/RileyGatto/SIT210-Riley-Gatto/tree/main/Task11.1Project

Unit Link (all tasks):
https://github.com/RileyGatto/SIT210-Riley-Gatto/tree/main

Testing Approach:

Within my implementation of my system I have integrated tests and edge cases to make sure my system runs as expected even in unusual situations. I have also completed extensive testing of my own to ensure everything is working as expected. To begin I added multiple serial.print lines in my arduino program. This was used primarily to ensure that my arduino was successfully connecting to wifi and MQTT broker. Through this I was able to debug and make sure my arduino was communicating with my raspberry pi effectively and applying the changes made by the user in the GUI.

Further I did my own extensive testing such as making sure even if the user selected manual mode if the moisture level was below the threshold it would not water. This test was important as it made sure the user could not accidently over water the plant without knowing. Further I made sure that if the user spammed the water button it would only register one as I used polling, it will first need to check the moisture level before deciding to water so it can not be overwater in this way. Lastly if the sensor disconnects the program will not allow the user to water the plant as the serial communication will be broken, so the program will state the moisture is too high so the plant will not be overwatered. So through these tests I was able to efficiently test my system against these edge cases.

User Manual:

Hardware Setup:

1. First Connect the moisture sensor to the arduino by connecting the sensor AOUT to A0 of arduino, sensor GND to arduino GND and sensor VCC to arduino Vin power pin. Then insert the moisture sensor into the plant's soil.

2. Connect the relay to the arduino by connecting it to the arduino 3.3V power pin, breadboard ground rail and arduino digital pin 2.

3. Connect the water pump power cable to the relay and connect the water pump's ground cable to the ground rail on the breadboard.

4. Connect the 9V battery to the power converter and snap onto the breadboard power rail. Then use a cable to connect the power rail to the relay which will supply the pump with 9v power when the relay is set to HIGH.

5. Connect the arduino to power such as the powerbank or usb from your computer if you want to see debugging information.

6. Connect the raspberry pi up to power and connect monitor, keyboard and mouse.

Software Setup:

1. Clone my github repository as you will need the program for the arduino and raspberry pi. Open a folder and copy the arduino.ino and secrets.h into the folder then open this folder in arduino IDE. Then change the secrets header to contain your wifi network credentials and raspberry pi IP address on your local network might be different from mine. Then plug your arduino into your usb port and flash the program to your arduino nano 33 IOT. You can then see in the serial monitor if it is connecting to wifi and MQTT server.

2. Copy the python.py file onto your raspberry pi. Then we will need to install the following libraries on your raspberry pi both paho-mqtt, Tkinter. You can do this by running pip install paho-mqtt and Tkinter comes preinstalled on raspberry pi.

3. We then need to set up the MQTT server. We do this by running “sudo apt install mosquitto mosquitto-clients -y”. Which will install the MQTT server then run sudo “systemctl start mosquitto” and “sudo systemctl enable mosquitto”. These will start the MQTT server and run it on your raspberry pi and also enable it to start on runtime each time you turn on your raspberry pi so you don't need to set it up each time.

4. Lastly we then need to run our GUI by going to the directory the python file is located in and running sudo python program.py which will open the GUI for user interaction. We also have to provide power to our arduino which will now connect to the MQTT server on the raspberry pi.

5. Lastly you will need to provide the IP address of the arduino in the python program you can get the IP address by adding the following Serial.print line in the arduino code and checking it in the serial monitor "Serial.println(WiFi.localIP());".

Operation:
Once The arduino and raspberry pi are powered on then you can use the GUI interface and manipulate how the plant is watered and see the moisture level. You can also adjust the moisture threshold depending on your environment; you just have to change the constant variables in the code. If you switch to manual mode you can click a button to water the plant. If you select automatic mode the system will automatically detect when to water.

How to Set Up in Your Environment:
Once you have set it up and compiled everything together and have the hardware set up in its box you can then follow these steps to add it to your environment. First insert the soil moisture sensor into your plants soil. Put the water pump in a tub with water filled so it can pump water to the plant. Place the tubing of the pump on the soil so it can pump water from the tub to the soil. Then connect the water pump to the hardware box which will allow us to control and monitor when to water the plant. If you follow the setup guide and this information you can then view the GUI on your raspberry pi and manipulate when the system waters the plant based on the soil moisture.

Link To Demonstration Video:

https://youtu.be/jdsvf8JXWDo

Conclusion:

Through this project I feel I have gained the experience and understanding of how to build my own embedded system. Through trial and error I was able to build a plant watering system which implements a range of unique features. I have also used many theoretical concepts I have learned in this unit such as modular and polling programming. This project was very important as it allowed me to showcase my skills and understanding of embedded systems. This unit was all about growing our understanding of how embedded systems work and in this last task we implemented this knowledge to build a project which has real world application. Overall this task has allowed me to reflect on my experience over the unit and apply these skills to build a final project to showcase my unique skills.

Riley Mark Gatto 224554695

I faced many challenges throughout my project and here are a few of them. The first challenge was being able to distribute the correct amount of power from my arduino. I originally found the relay connected to the VIN power pin but this was resulting in the relay not getting the necessary power voltage. So to fix this I changed it to the 3.3V pin and connected the sensor to Vin which worked as both had the correct amount of power to execute its actions. Another challenge I faced was syncing the data transmission between the arduino and raspberry pi. I found it challenging in the beginning to use MQTT to send data between the two devices. To overcome this I made sure both devices were communicating on the correct ports and not overlapping each other when communicating. If I had the chance to do this task again I would have expanded my system. I could have done this by including additional water pumps and sensors to water multiple plants. This would have allowed me to present to a larger audience as I could have advertised a larger scale of my project. Lastly I would have also included more fault tolerant features to make sure my project ran more efficiently. But overall I believe I have made a successful project which aligns with the marking rubric.

Image Of Project: