IOT Design Jury

•••

Soil Moisture Tracker.

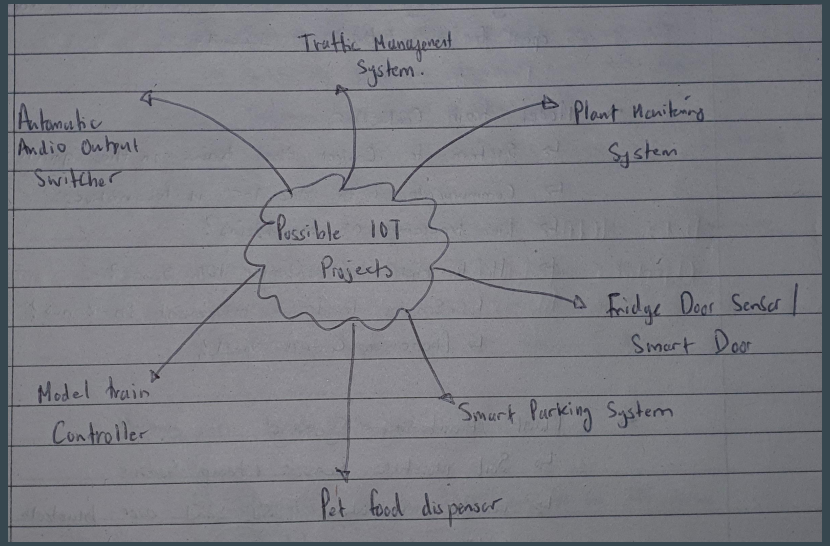
Intro To Project

The project was selected through complete coincidence, I had a list of possible projects and one day my flatmates came home with a bunch of plants, which decided on the project for me.

The main purpose of the system was to remind us when to water the plants as in the past where have been located, they have been placed somewhere out of sight and out of mind and often get neglected and I did not want a repeat of previous instances.

I had thought to use either BLE or WiFi to send a notification to devices at a certain point to water the plants, with WiFi being the final choice, although due to authentication issues with my hall's network, I had to connect to my phones hotspot for most of the time.

The device works as follows: reads the soil moisture and temperature from one of the plants, calculates an average of those values over a 10 second period and then sends a email to a recipient when the soil moisture reaches a certain threshold indicating watering is needed.

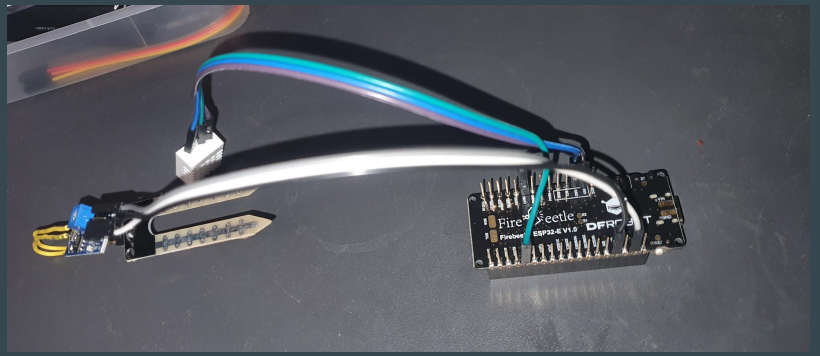


IOT Projects Mind Map

Development

Stages of Development:

- Wire up the soil moisture sensor and DHT22 to the Arduino
- Ensure that readings were given from both sensors, calculating averages for improved accuracy, which had complications within itself.
- Establish baseline readings for wet and dry soil
- Create a new email account and establish WiFi communication on the Arduino
- Install the ESP_MailClient library and add Gmail's SMTP settings to the code
- Build the emails body with values recorded from the sensors
- Add the logic to only send an email when a threshold is met
- Make use of the MailClinet library and send the email using the settings provided.



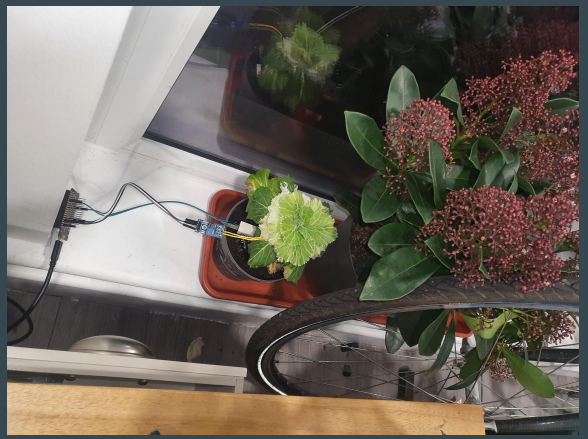


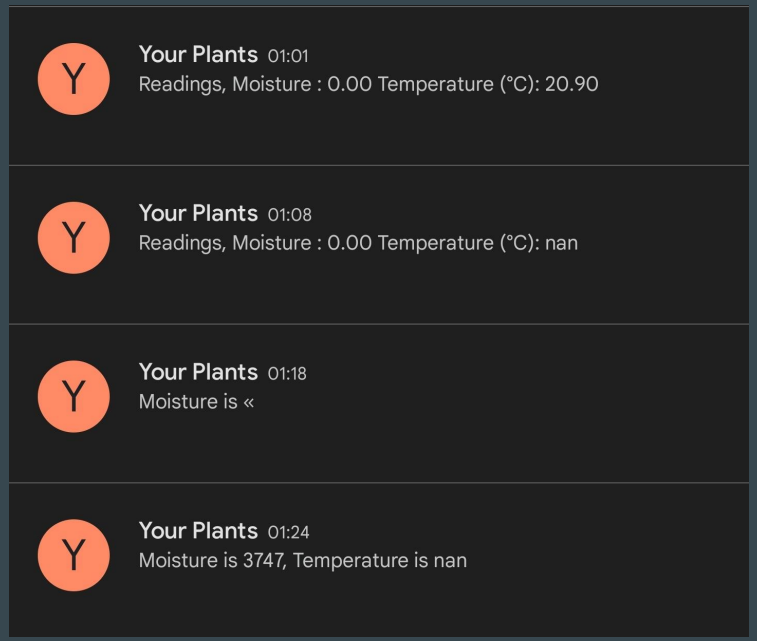
Image above: IOT device

Image adjacent: IOT device deployed in the kitchen

Issues During Development

As mentioned in the previous slide there were a few complications when developing this project, some highlights were:

- The DHT22 sensor did not want to read values at all, with many libraries being outdated and no longer functioning correctly for my usage. Eventually one was found which gave the desired functionality.
- The values from the sensors not displaying correctly in the email body, often they would not show at all, or be recorded as ‘nan’, a solution for this was to add a considerable delay to the code to allow the sensors to start up and start recording data before the check to see if the data had reached the threshold, in addition the float values obtained from the sensors were converted into strings before attached to the email body as there were problems with building the body of the email with various data types even with the use of sprintf and snprintf
- There also were instances where multiple emails would be sent out, to prevent this a bool value was used to track if an email was already sent out when the threshold was hit.



Examples of unsuccessful emails

Reflection

Overall, I deem the project to be a success, the simple nature of the project didn't leave much opportunity for failure of completion which was a real concern with other modules work also increasing.

So far the project has made a positive impact around the flat, the plants are watered daily and have lasted longer than their predecessors.

Looking back against the original concept there are some areas where there could be some further development over time, such as having a 24 hours view of previous data graphed to see consumption patterns or even dispensing water automatically when the threshold is hit to ensure the plants stay alive as long as possible.

Example of successful email

```
const char *SSID = "GalaxyNote10+";
const char *PWD = "123456789";
bool sent = false;
int x = 1;

#define GMAIL_SMTP_SERVER "smtp.gmail.com"
#define GMAIL_SMTP_USERNAME "COMP6570@gmail.com"
#define GMAIL_SMTP_PASSWORD "smtp_password"
#define GMAIL_SMTP_PORT 465
```

WiFi and Gmail SMTP settings