University of Colorado
Boulder

# EMBEDDED INTERFACE DESIGN (ECEN 5783)
# SPRING 2021

## PROJECT 8

## TITLE:
## PLANT AUTOMATED WATERING SYSTEM (PAWS)


## PROJECT BY:
## MICHAEL FRUGE
## &
## BRYAN CISNEROS

## PROFESSOR:
## PROF. BRUCE MONTGOMERY

# Work Breakdown Structure:

Completed  Not Completed

## 1.1 Design

| | | | |
|---|---|---|---|
| 1.1.1 | WBS | Team | M |
| 1.1.2 | UML Use Cases | Team | M |
| 1.1.3 | UI Wireframes | Mike | L |
| 1.1.4 | Architecture Diagram | Bryan | S |
| 1.1.5 | Wizard of Oz | Team | L |

## 1.2 Subsystem Implementation and Integration

### 1.2.1 Hardware Interfaces

| | | | |
|---|---|---|---|
| 1.2.1.1 | Humidity Sensor Functionality | Bryan | L |
| 1.2.1.2 | Sensor → Xbee Dev Board | Bryan | L |
| 1.2.1.3 | Xbee → Raspberry Pi | Bryan | L |
| 1.2.1.4 | Raspberry Pi IOT Thing | Mike | L |

### 1.2.2 Software Interfaces

| | | | |
|---|---|---|---|
| 1.2.2.1 | Thing Rules | Team | M |
| 1.2.2.2 | DynamoDB Setup | Mike | M |
| 1.2.2.3 | AWS SNS Notification | Bryan | S |
| 1.2.2.4 | API Gateway | | |
| 1.2.2.4.1 | RESTful API | Bryan | M |
| 1.2.2.4.2 | HTML User Interface | Mike | M |
| 1.2.2.5 | Qt User Interface | Mike | L |

1.3 System Test

| | | | | |
|---|---|---|---|---|
| 1.3.1 | | Subsystem Tests | Team | L |
| | 1.3.1.1 | UI Customer Review | Team | M |
| | 1.3.1.2 | Moisture Tests | Bryan | L |
| | 1.3.1.3 | Xbee Communication | Bryan | M |
| | 1.3.1.4 | IOT rules verification | Team | M |
| | 1.3.1.5 | AWS Sensor Data Storage | Mike | M |
| | 1.3.1.6 | UI -> API Gateway Tests | Mike | M |
| | 1.3.1.7 | Data Collection | Team | L |
| 1.3.2 | | System Test | Team | L |

1.4 System Rollout

| | | | | |
|---|---|---|---|---|
| 1.4.1 | | Project Documentation | Team | L |

# Project Reflection Statement:

Project Successes:

Our biggest concern with this project was getting 2 way communication between the UI, AWS, and the sensors/gateway. We ended up accomplishing this by creating separate databases for configuration data obtained from the UI, and sensor data obtained from each of the sensors. This architecture, along with publishing and subscribing to different MQTT topics allowed us to configure sensor data from the UI, which was one of our biggest goals with this project.

Our API gateway ended up coming together very well, which really eased the worries of communication between the UI and AWS. This is something we were not familiar with coming into the project, and both members of the team learned quite a bit about how to properly set up and utilize a RESTful API with AWS.

Project Shortcomings:

We decided not to implement sensor groups for this project, as we felt the value it added was not enough for us to focus on that, and we decided instead to focus on the 2 way communication mentioned in the Project Successes.

We also were unable to implement the 2 separate UI's required for this project. Instead we worked with the QT UI and decided to explore more aspects of PyQT such as using the QNetwork library, Layouts, and adding widgets dynamically. Since we had already implemented an HTML/JQuery UI in project 5, we made this decision to become more experienced with PyQT as well as developing other aspects of our system.

Lastly, we have not implemented any graphing functionality for the system as intended in the original project scope. This was due to us simply running out of time to implement the feature, and in the future we would certainly implement this if we had another day or 2.

# Third Party Code Used Statement:

- W3schools.com
    - Assistance on python requests library and other basic python syntax consulting
- Docs.aws.amazon.com

    - Provided background information and tutorials regarding AWS services utilized in this project such as DynamoDB, API Gateway, AWS SDK, Lambda
- EID Project 4
    - Helped develop the structure of our QT UI and used methods learned through that project to help us make this UI better
- https://zetcode.com/gui/pyqt5/eventssignals/
    - Taught us how to define our own error objects in PyQT5 so we could create a custom error class that assists in error handling and displaying message pop-ups
- https://learn.adafruit.com/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor?view=all
    - Example code for soil moisture sensor. We based our main XBee code on this example code.
- https://github.com/adafruit/Adafruit_CircuitPython_seesaw/blob/master/adafruit_seesaw/seesaw.py
    - Library for communicating with the I2C sensor. We had to make some modifications but largely we used that code as is.