SDDEC21-08: IoT Sensor Project

Client: Mark Easley, Texas Instruments
Faculty Advisor: Daji Qiao

Team Members:
- *Walter Gilbert - Frontend*
- *Mason Gil - Frontend/Backend*
- *Zach Kauffman - Firmware/Backend*
- *Thomas Smeed - Hardware*
- *Daniel Phalen - Hardware*

Motivation

Problem:

Caring for plants can take quite a bit of time and commitment.
IoT sensors connected to a website can take the guesswork and memory out of caring for plants.

Solution:

A smart-plant ecosystem that will read various sensors connected to a common houseplant and relay the information to a website where users can view their plant's status and setup alerts for when sensor values go out of the defined bounds.

Intended Users:

- Hobbyists with an interest in keeping plants alive
- Large-scale greenhouses that want an easy way to monitor their operation

Design Requirements

Functional

- Sensor status visible on front-end
  - Text format
  - Graph format
- Visual indicator on the sensor module of if the sensors are in range
- Users can add multiple sensor modules
- Users can add up to 8 sensors per sensor module

Non-functional

- If a sensor reading goes out of bounds, a user should get an email with 5 minutes stating that fact.
- The hardware can run for upwards of a week without user interaction

Engineering Constraints

- Low-power
- Cost-effective

Operating Environment

- Indoor usage
- Outdoors: waterproofing out of scope

Standards

- Agile development standards
- Javascript coding standards
- Functional programming standards
- Git conventions
- 2-layer PCB

Technologies used

Hardware:

- Custom PCB
- Off-the-shelf analog sensors

Firmware: TI CC3220SF Launchpad

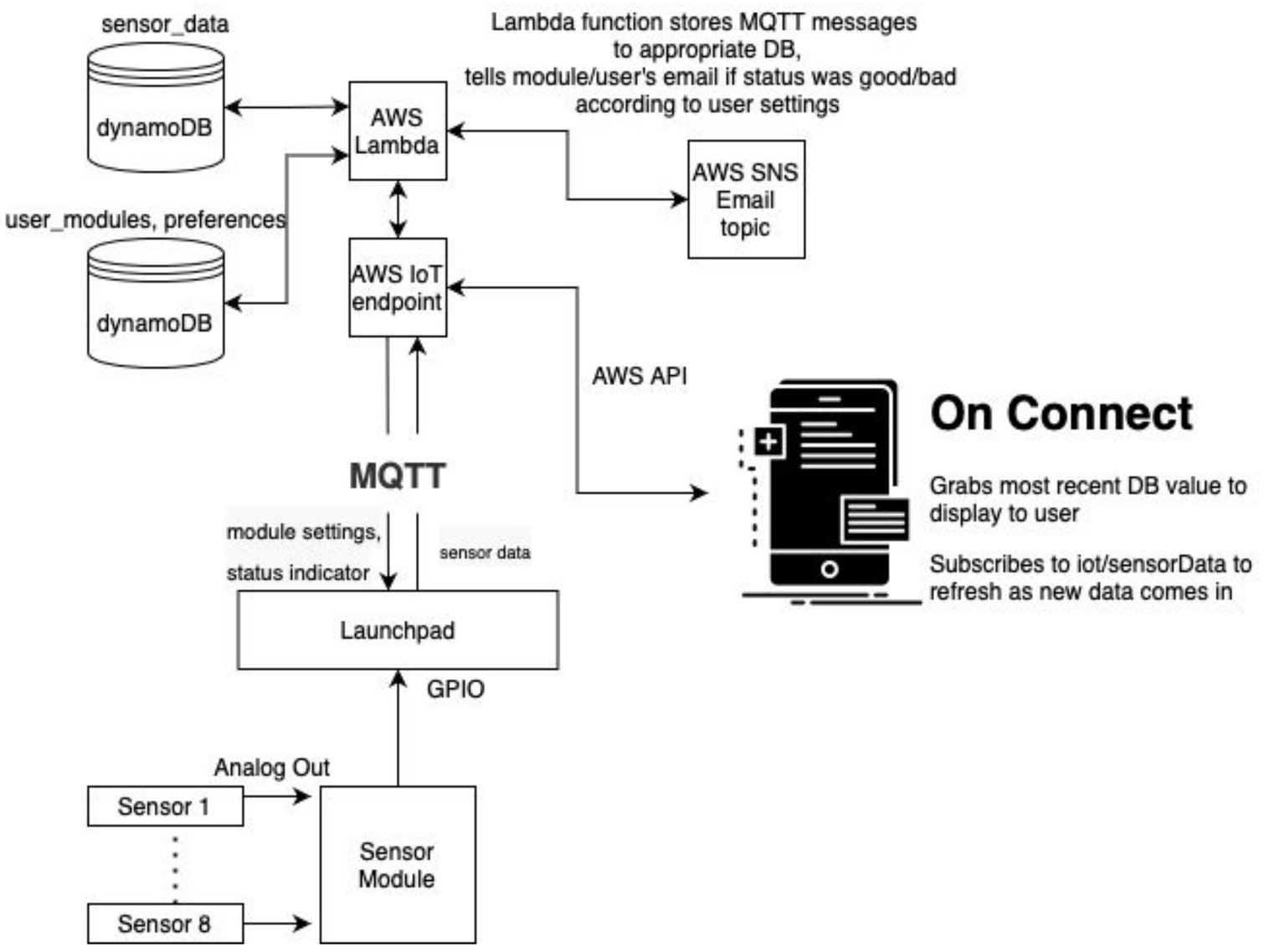
- Programmed in C
- FreeRTOS
- Connects to our AWS IoT endpoint
- Communicates via MQTT

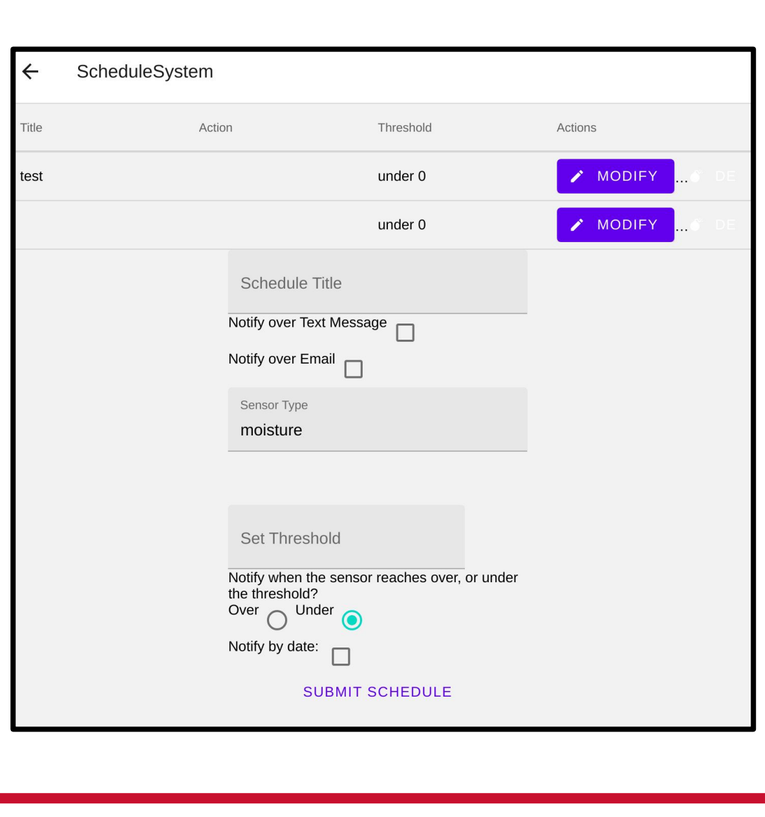
Backend: AWS

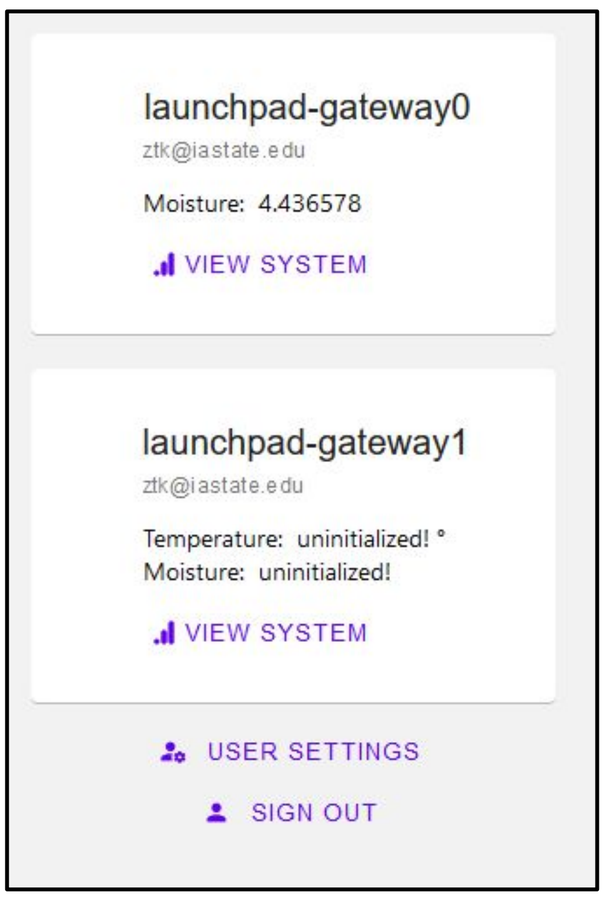
- AWS IoT Core
- AWS Lambda
- AWS DynamoDB
- AWS Amplify

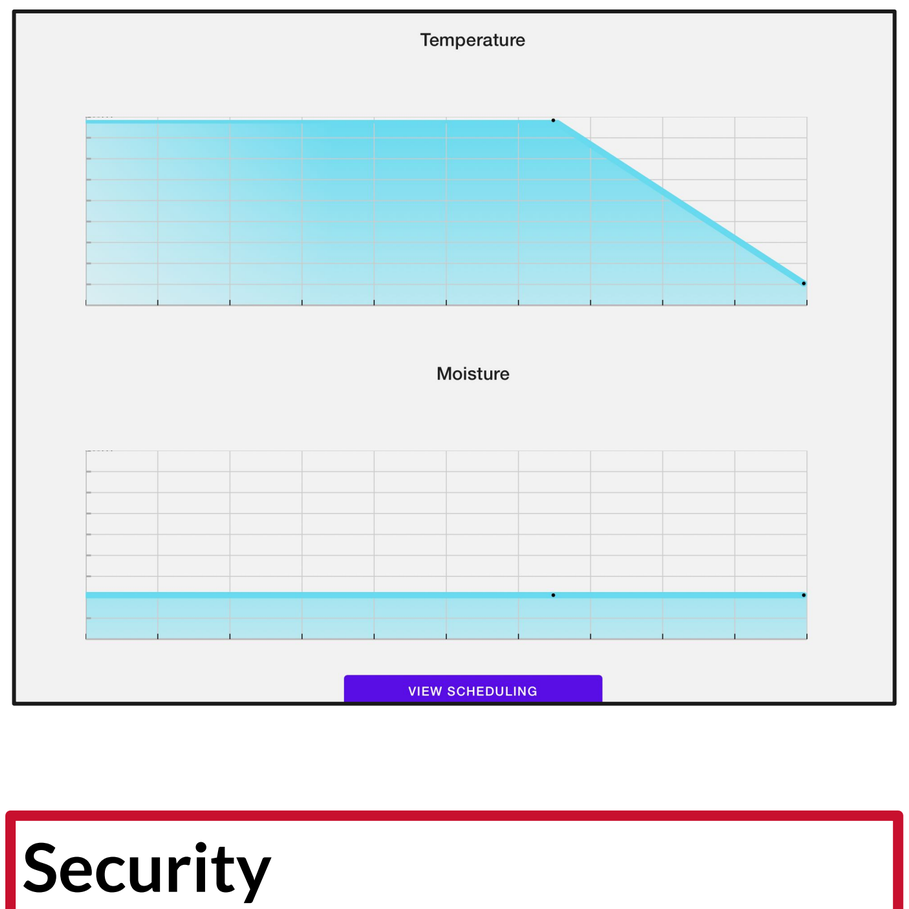
Frontend:

- React Native
- Amplify









Testing

Front-end integration testing
- AWS backend and mocked data
- Firmware control loop to frontend display

Backend unit tests:
- Mocked MQTT data makes it to database
- Lambda function tests

Firmware unit tests:
- Test connection to AWS with fake data
- Test reading data from 2 analog sensors without AWS

This testing allowed us to present a final product in line with our functional and non-functional requirements.

Security

- One of the main benefits of AWS was that it handles our authentication
- Amplify handles making accounts on the website
- AWS IAM handles giving those accounts the proper access to...
  - Read from our sensor_data database
  - Write to our preferences database

Results

Our project achieved our functional and non-functional requirements, although our hardware layout suffered from longer shipping times, and we were unable to get our final PCB revision with vital fixes in time. A successful demo was prepared using a breadboarded version of that PCB.