Helium Localized Environmental Alerts and Forecasting (LEAF)

EEC 193 A/B Senior Design Project

Team Omega

Brandon Alba, Jaskirat Singh, & Jierui Yang

Team Members

Name: Brandon Alba
Major: Electrical Engineering
Worked on: Circuit/firmware, mobile app

Name: Jierui Yang
Major: Computer Engineering
Worked on: Mobile app functionality

Name: Jaskirat Singh
Major: Computer Engineering
Worked on: Cloud infrastructure, cloud computing, mobile app

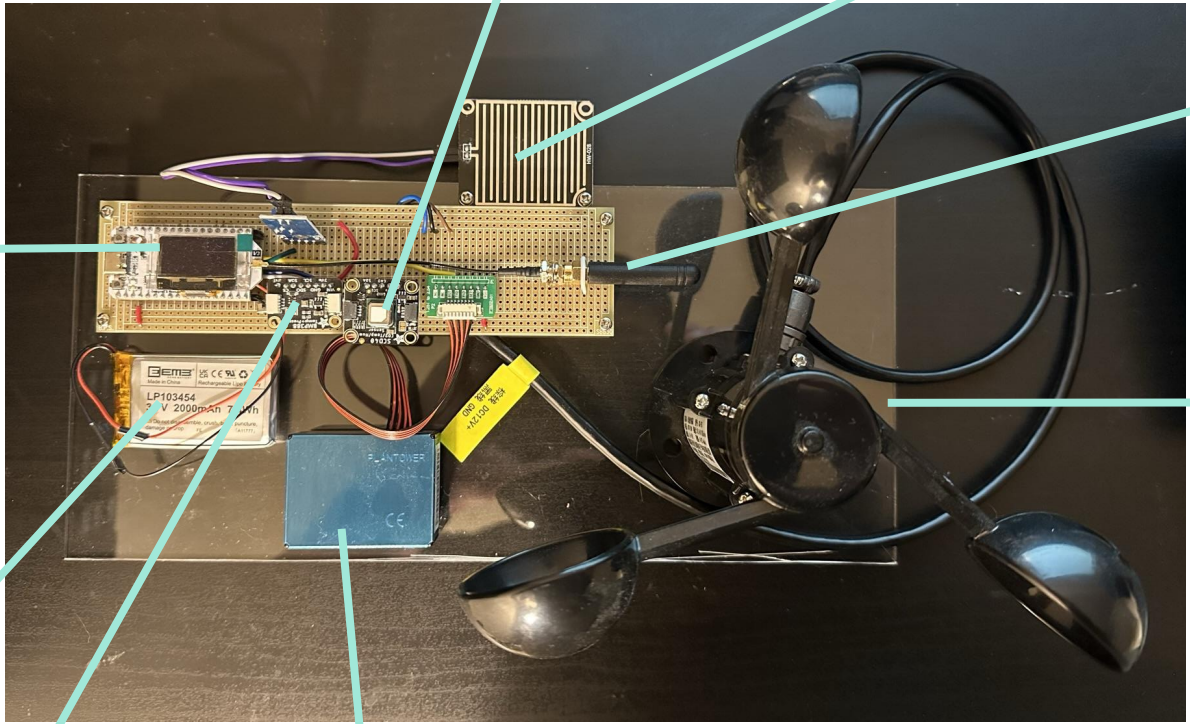
Project Overview

What Does the Device Do? (Why is it an IoT Device?)

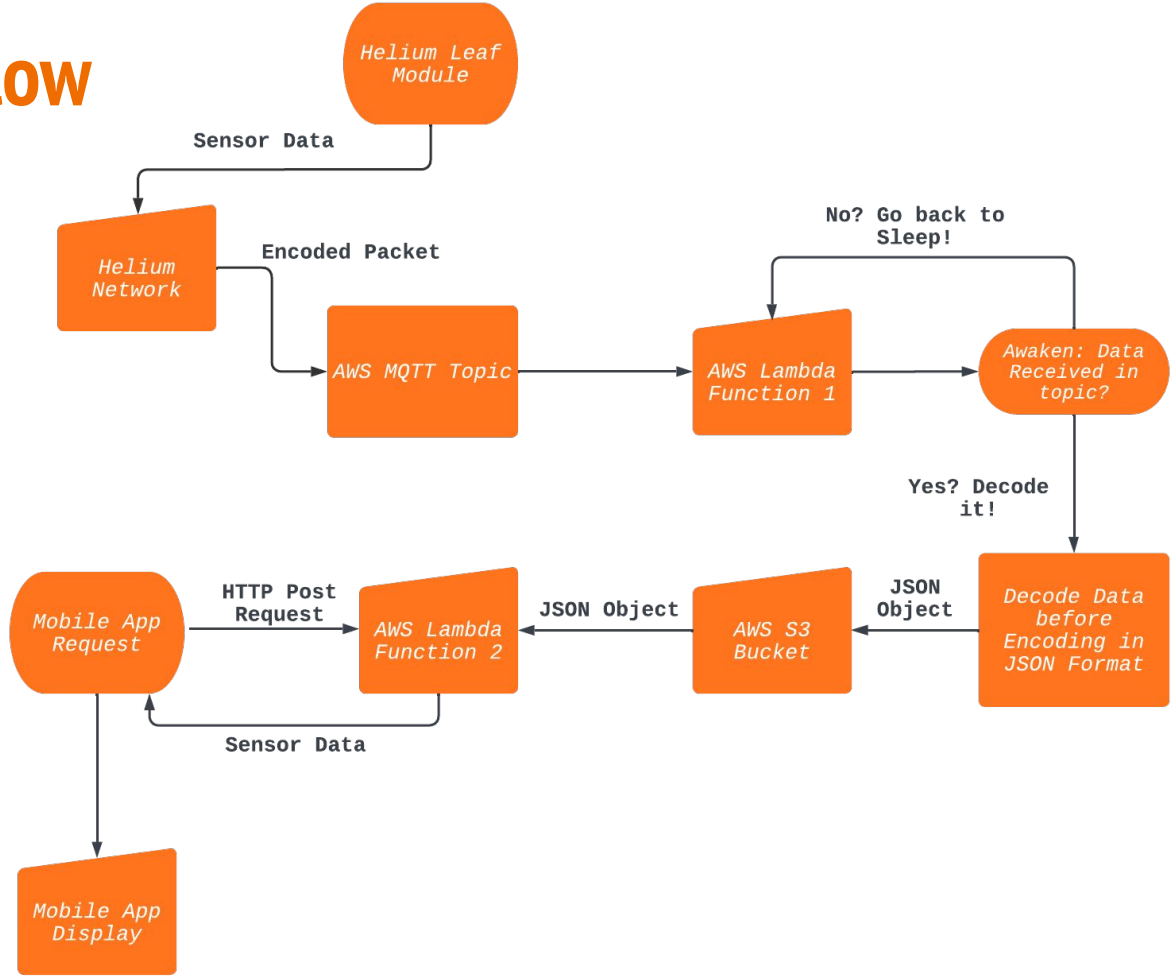
- Battery-powered (onboard charging) and various **sensors**:
  - Wind speed
  - Temperature/humidity/pressure
  - CO2 concentration/AQI
  - Rain level
- Transmits real-time **L**ocalized **E**nvironmental data to mobile app, which provides **A**lerts and **F**orecasting.
  - With multiple instances of these devices distributed over populated areas, can **compute** risk factors more accurately and quickly for: heavy pollution, forest fires, floods, and more
- Onboard LoRaWAN transmitter to **communicate** data over Helium network, allows cloud communication without cellular
- **Mobile** app with GPS support that can show user's proximity to devices and also display detailed real-time + historical data
- **Cloud computation** involving decoding and encoding

Part	Price	Function
WiFi LoRa 32 (V3) IoT Dev Board	$18.90	Central microcontroller for the system
BMP388	$15.77	Air pressure/altitude sensor
SCD42-D-R2	$53.75	CO2 sensor (ppm)
PM 2.5 Air Quality Sensor	$39.95	Particulate matter (PM) sensor
LM393 Rain Sensor	$6.49	Rain sensor
CALT Anemometers	$44.95	Wind Speed Monitoring Sensor
Voltage Regulators (x2)	$9.99 for 10	Power for wind speed/AQI sensors
Total	$189.80	

Prototype Diagram



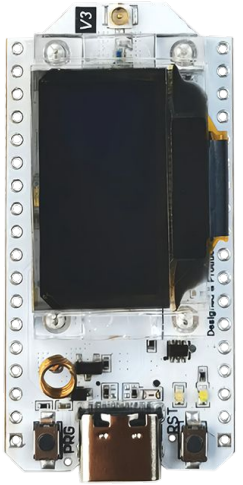
Data Flow


Helium Leaf Module
Sensor Data
Helium Network
Encoded Packet
AWS MQTT Topic
AWS Lambda Function 1
No? Go back to Sleep!
Awaken: Data Received in topic?
Yes? Decode it!
Decode Data before Encoding in JSON Format
JSON Object
AWS S3 Bucket
JSON Object
AWS Lambda Function 2
HTTP Post Request
Mobile App Request
Sensor Data
Mobile App Display

Circuit and Software (Brandon)

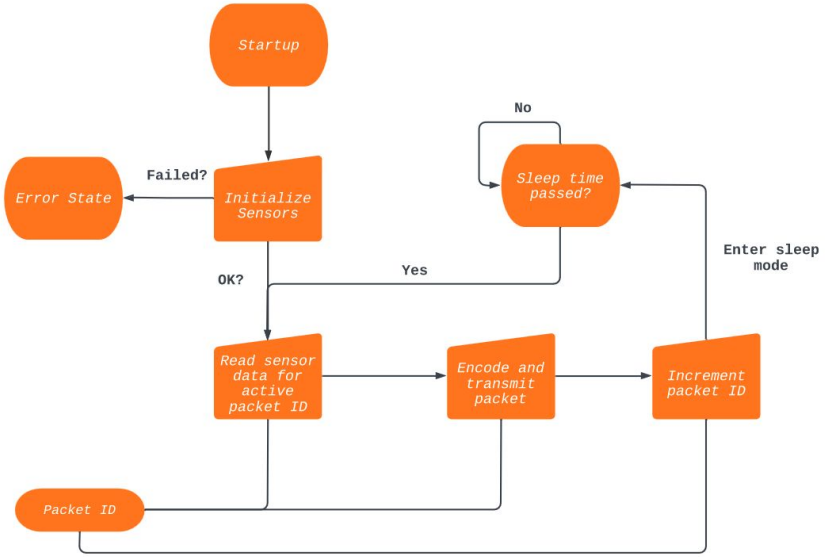
Development Board: Heltec Wifi LoRa v3

- Utilized following hardware:
  - I2C (BMP388 pressure sensor and SDC40 CO2 sensor)
  - UART (Rx only, AQI sensor)
  - Analog inputs (x2, wind speed and rain level sensor)
  - Onboard LoRa transceiver + antenna
  - Onboard battery management system (BMS) for battery power/charging
  - OLED display for diagnostics/debug
- Heltec provides Arduino board translation layer for use in Arduino IDE (installed manually)
- Supports sleep mode for better battery life



Arduino Software Flowchart

- Helium packet size limit 24 bytes - multiple packet fragments required, labeled with ID
  - Device sends one packet fragment at a time, sleeps between packets
  - Advantage is only certain sensors need to be captured at a time (saves power)
- Packets are encoded with Google **protobuf**, API for converting C structures to serialized data



Cloud Computing on AWS (Jaskirat)

AWS Infrastructure

Utility	Function
AWS IoT MQTT Topic	Catalyst for receiving Helium raw packet
AWS Lambda Functions	Cloud computational resource for decoding and encoding packets w/ watchdog functionalities on other AWS Utilities
AWS S3	Mass cloud storage utility
AWS API Gateway	Efficient API gateway utility for HTTP functions
AWS Lambda Layers	Asset of AWS Lambda used to import custom decoding algorithms

AWS Integration

Helium packets are posted to an IoT MQTT Topic (Google ProtoBuf)

Lambda function triggers when it is posted

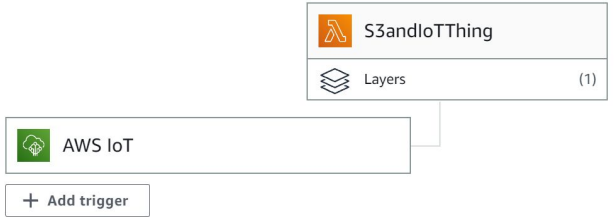
- Decodes packet and stores in a AWS S3 bucket

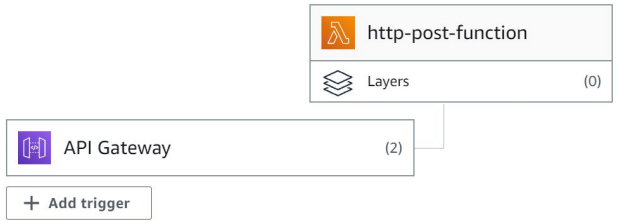
AWS API Gateway creates link for app to request data

Lambda function triggers when app requests data

- Sends S3 bucket object to app

IAM roles created to give permissions to each one of these tasks

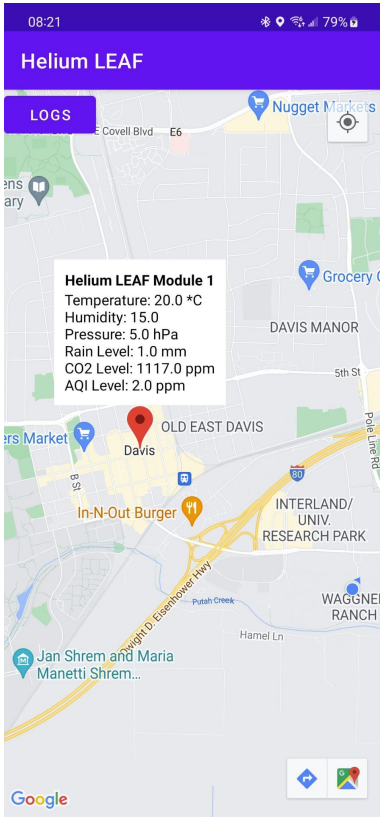


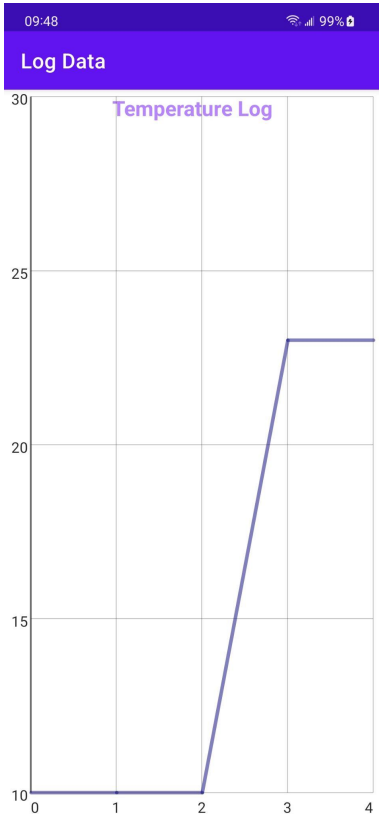


Mobile App (Jierui)

Dev. Platform: Android Studio

Google Maps API	Show the map on main screen
JJoe64 GraphView	Plot Log data graph (line graph view)
getDeviceLocation	Allow the app to access user location
HTTP POST	Read sensor data from AWS
data.GetString()	Get different sensor value from JSON
MarkerOptions()	Set the Module as a marker on the map, present data in snippet
myDataArray	Store data for plotting log graphs by updating y-axis value
buttonClick	Log button





Communication with AWS

- Uses HTTP POST request
  - Advantages: simple, cross-platform, no need to learn complicated library
  - Also allows other scripts/programs to request same data from AWS for easy testing

Steps for communication:

1. Every few seconds (Android coroutine) app sends its own location in JSON via POST
2. AWS lambda function can use location info and return JSON containing device IDs + associated sensor data

```
with(url.openConnection() as HttpURLConnection) { this: HttpURLConnection
    requestMethod = "POST"  // optional default is GET

    inputStream.bufferedReader().use { it: BufferedReader
        val response = StringBuffer()
        var inputLine = it.readLine()
        while (inputLine != null) {
            response.append(inputLine)
            inputLine = it.readLine()
        }
        println("Response : $response")
        val data = JSONObject( json: """$response""")

        return data
    } ^with
```

Code snippet that reads the AWS sensor data into a JSON object for further use in app

Questions?