



# Group 25: IoT Environmental Sensors

## Senior Design – Final Presentation

CSULA Team Members:

Juan Avila, Anthony Huynh, Jose Rodriguez, David Hernandez,  
Jeffrey Espinoza, Vincent Oviedo, Akbar Rizvi

Advisor: Airs Lin



Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---



# Background

- Network of physical devices embedded with sensors or software that help interexchange information over the internet
- Each device supports easy connectivity to our sensor network without much consumer configuration



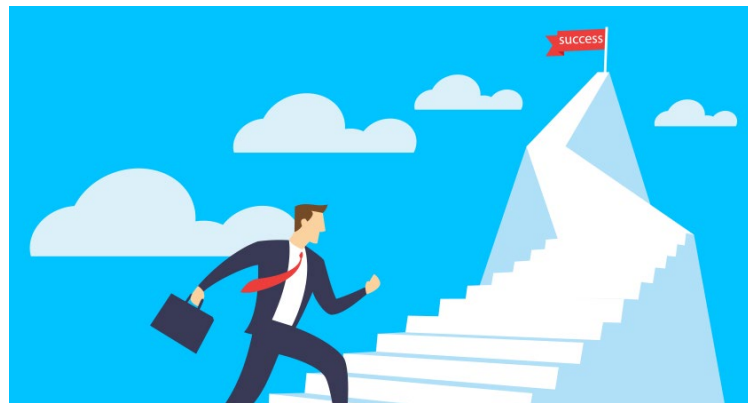
# Problem

- Using IoT to detect CO<sub>2</sub> in any environment
- Reduce human dependency
- Design sensors to collect environmental data and upload to our servers



# Objective

- Build a 4 stationed system that can:
  1. Detect both indoor and outdoor environment changes
  2. Gather data from stations and upload it to AWS server analyze
  3. Easy user deployment and scalability

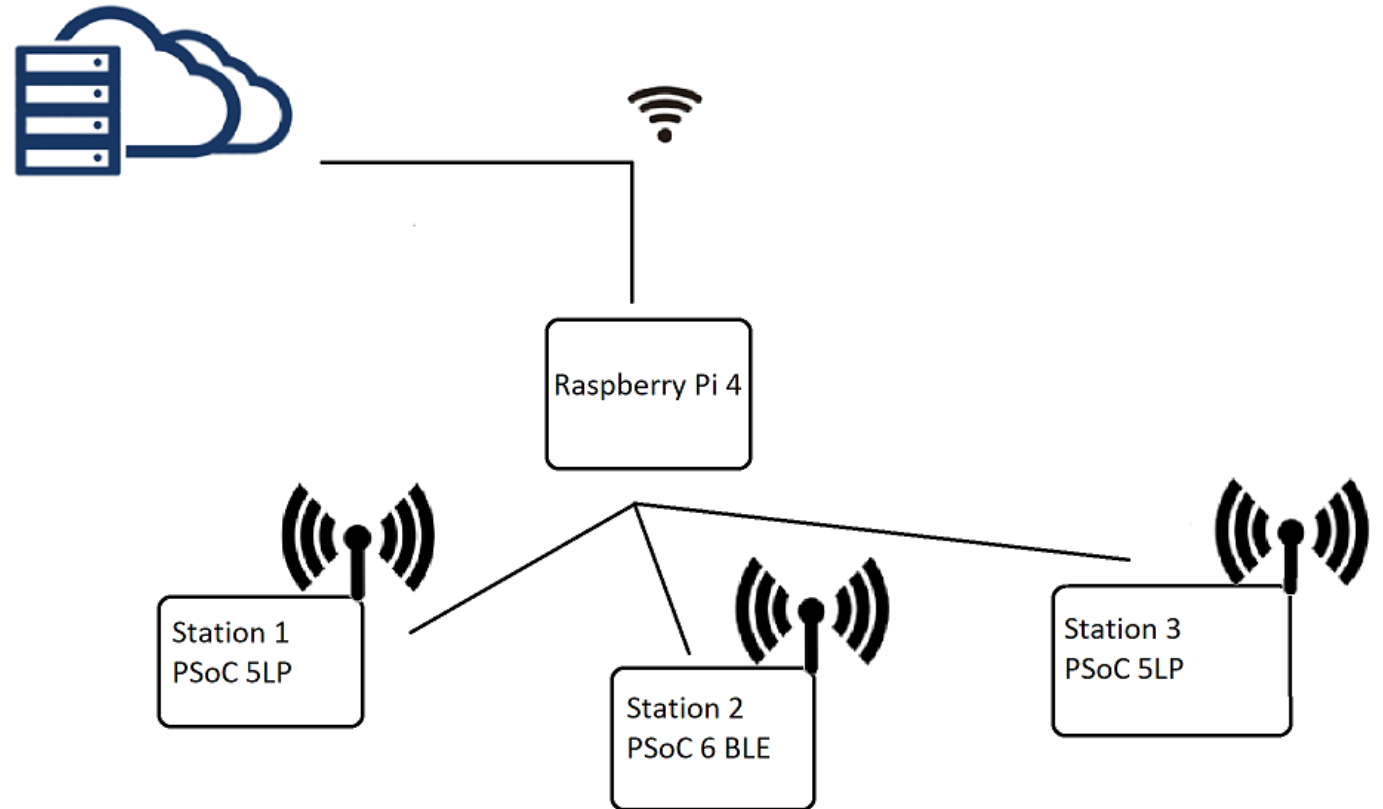


Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
<b>System Overview</b>	<b>David Hernandez</b>
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---




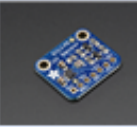








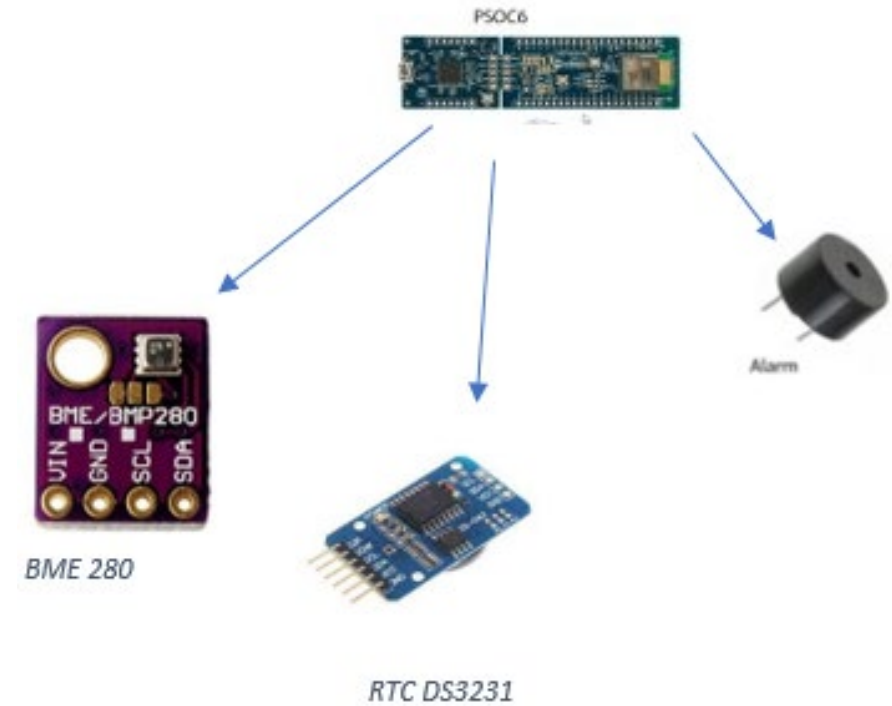
# Concept Design

- **Core station**
  - Internet connectivity
  - and RF connectivity
- **Sensor stations**
  - Scalable
  - Small footprint
- **Server**
  - Storage
  - Remote accessible
  - Affordable



# Environmental Sensors

Station 1	Weather sensors	Station 2	Indoor Temp/Humidity	Station 3	Indoor Gases/outdoor Gases
BME 280 Temp Humidity Pressure		BME 280 Temp Humidity Pressure		PPD42NJ Particle sensor unit	
SI1145 Digital UV light/index Infrared light				MQ-4 Methane Gas sensor	
SEN-15901 Weather meter kit 3 Parts:				CJMCU-811-CCS811 Gas sensor	
Rain Gauge Rain intensity				CO2 VOC MOX	
Anemometer Wind speed					
Wind Vane Wind direction					

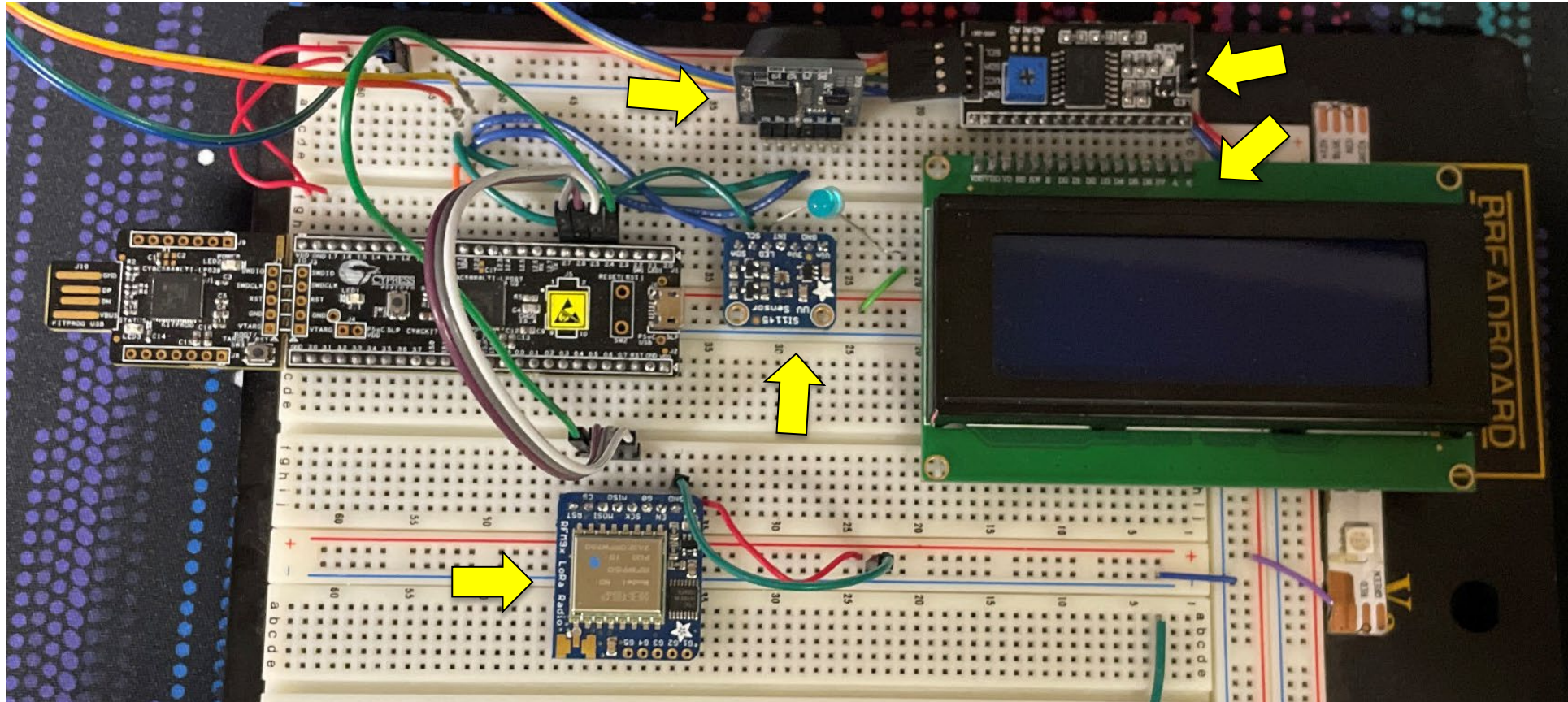




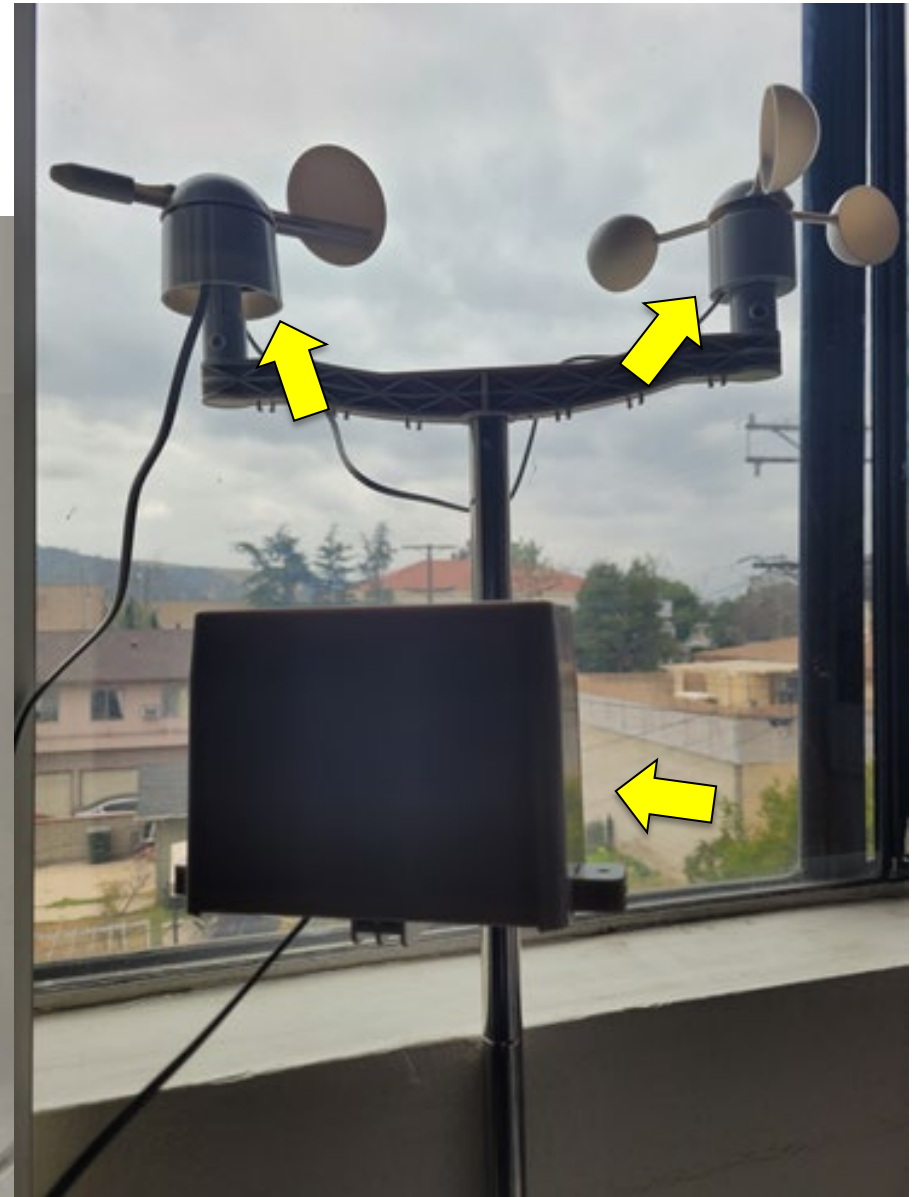
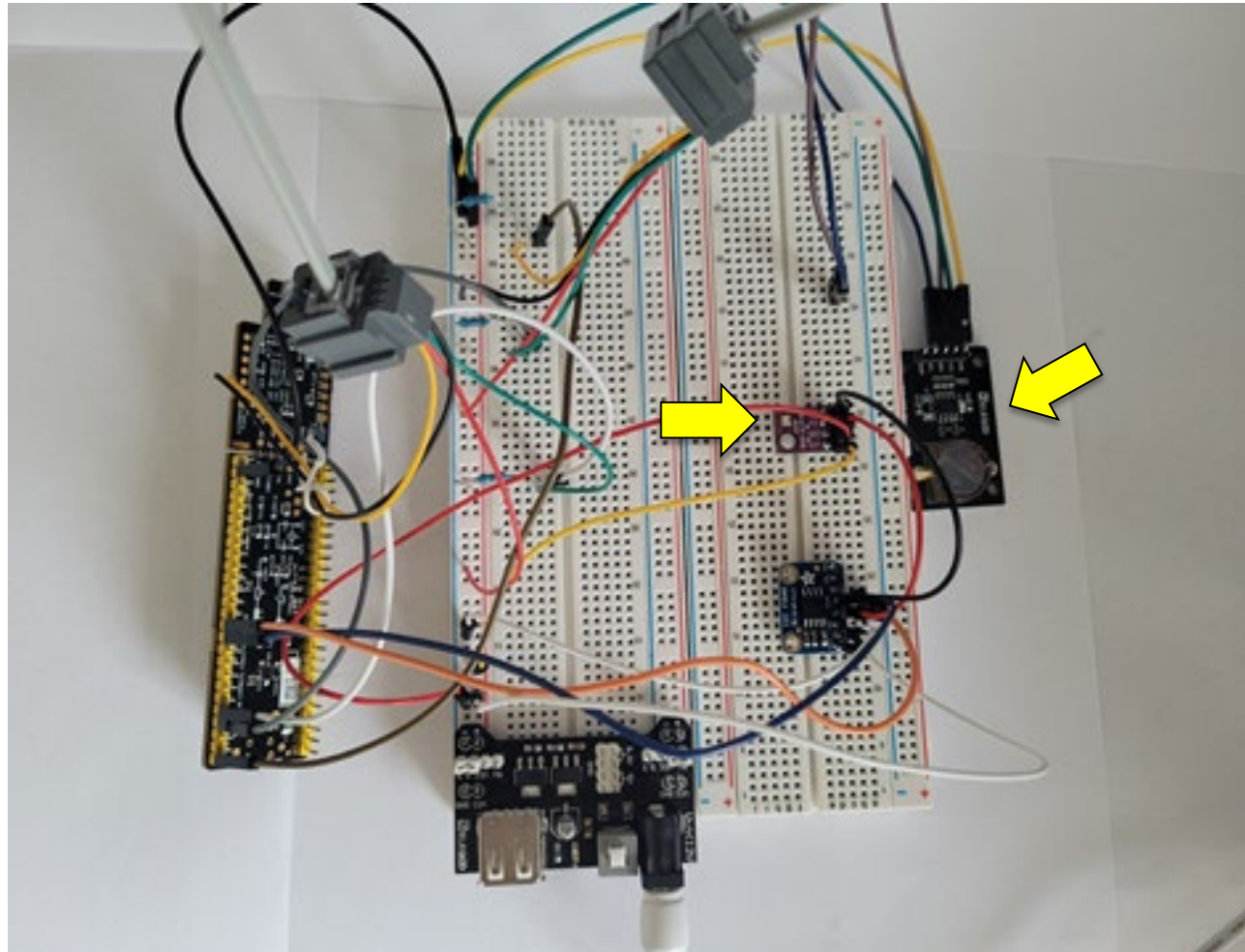
Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---



# Station 1 – Wired up



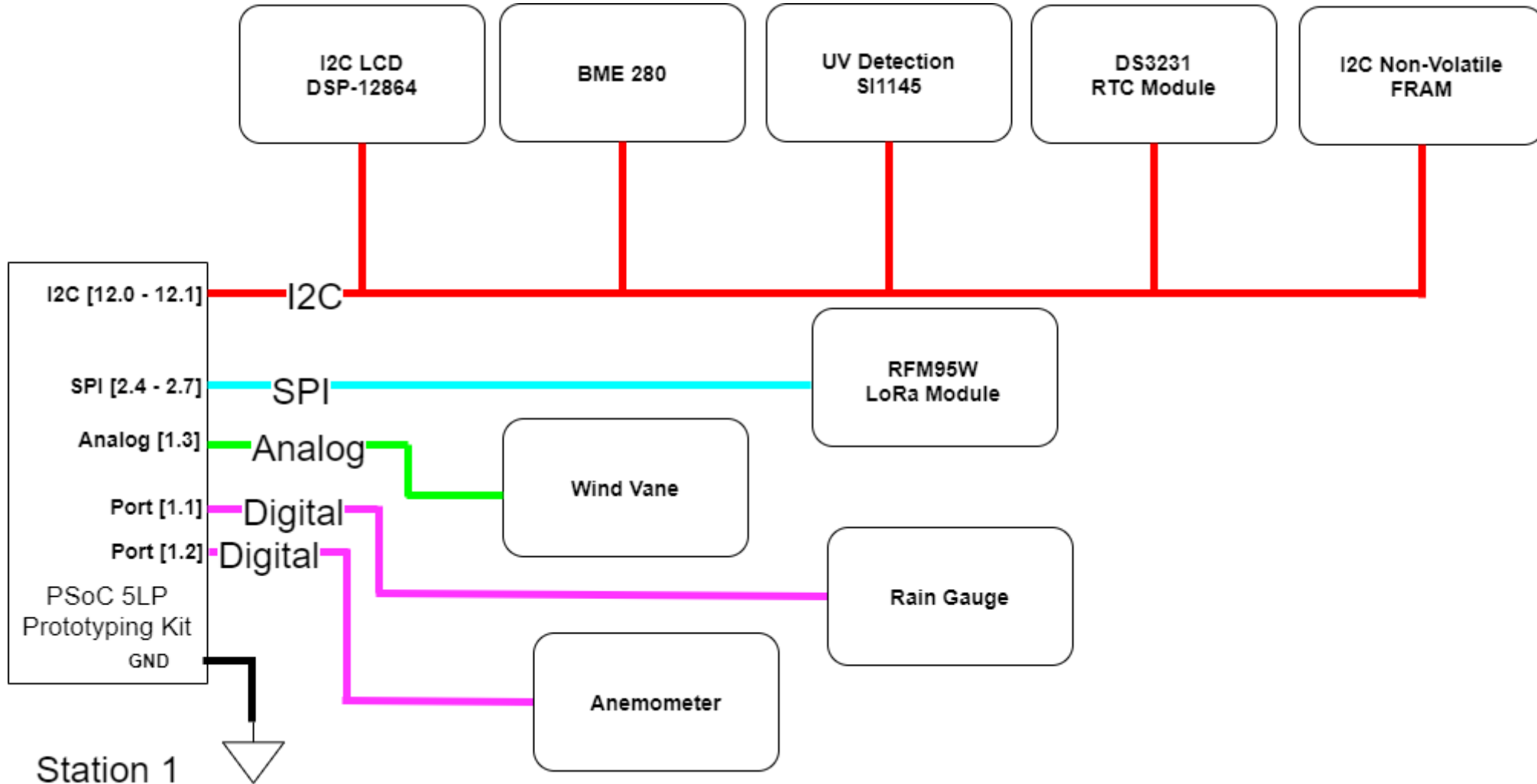
# Station 1 – Wired up



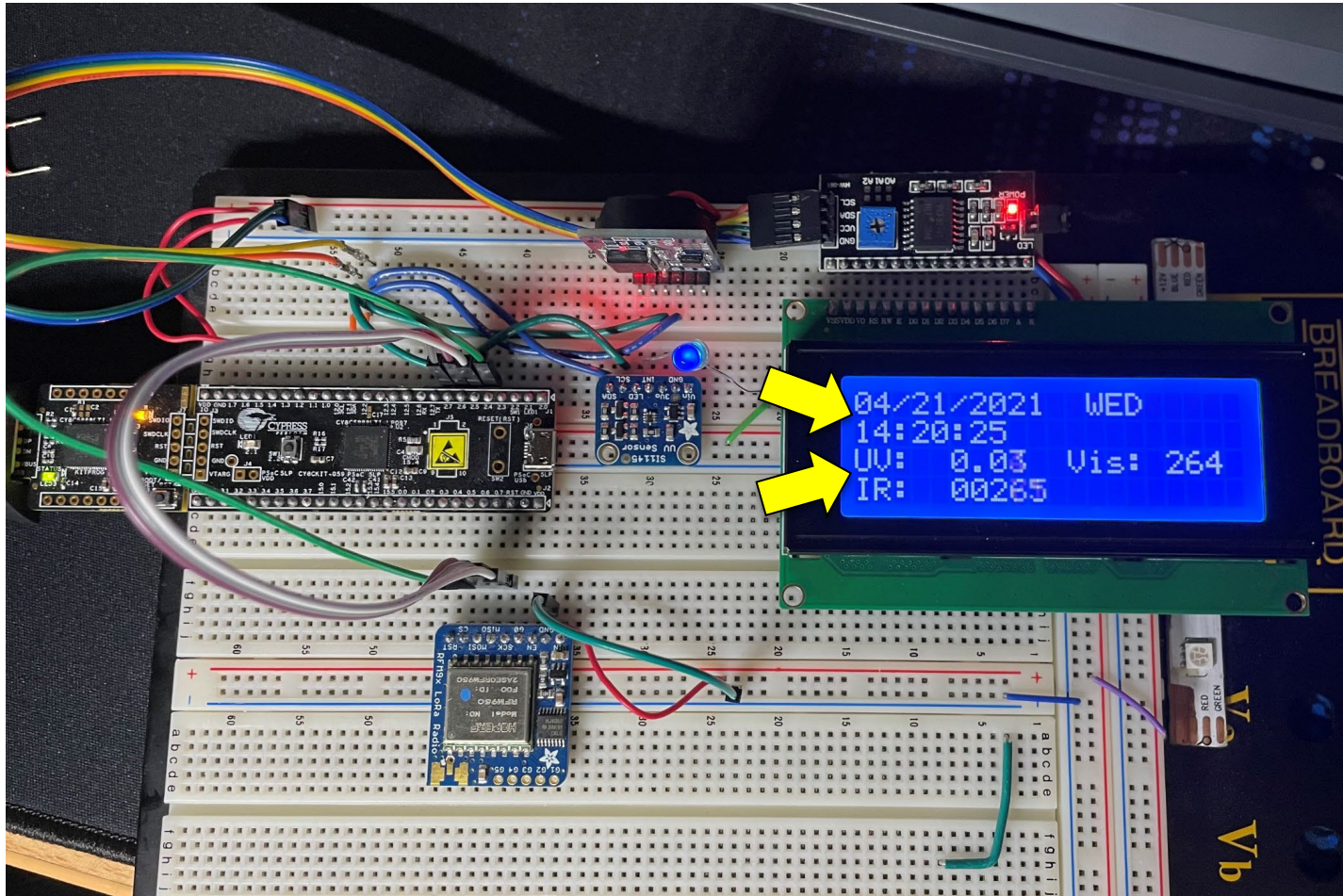
# Station 1 – Sensor Information

Microcontroller				
PSoC 5LP Prototyping Kit (CY8CKIT-059)				
Sensors	Model	Voltage	Max Current	Interface
LCD screen	2004A LCD Display Module & I2C Graphic LCD Adapter	3.3 – 5 V	66 mA	I2C
Temperature, Pressure & Humidity	BME 280	3.3 – 5 V	3.6 $\mu$ A	I2C
Real-Time Clock	DS3231 RTC	2.3 – 5.5 V	200 $\mu$ A	I2C
Ferroelectric RAM	I2C Non-Volatile FRAM	2.7 – 5.5 V	200 $\mu$ A	I2C
LoRa Radio Transceiver	RFM95W LoRa Module	3.3 – 5 V	130 mA	SPI
Rain amount	Rain Gauge	5 V	0.5 mA	GPIO
Wind speed	Anemometer	5 V	0.5 mA	GPIO
Wind direction	Wind vane	5 V	0.5 mA	Analog(ADC)

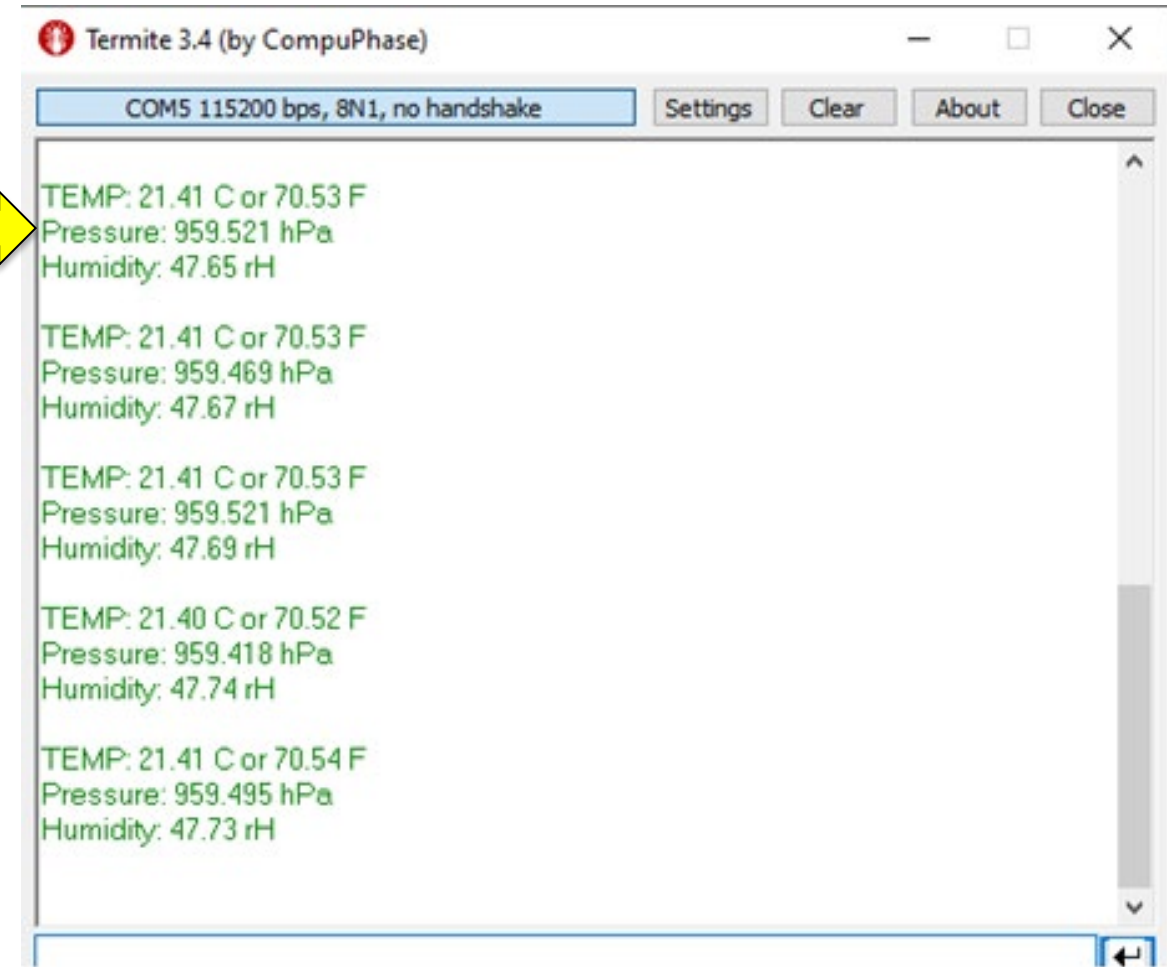
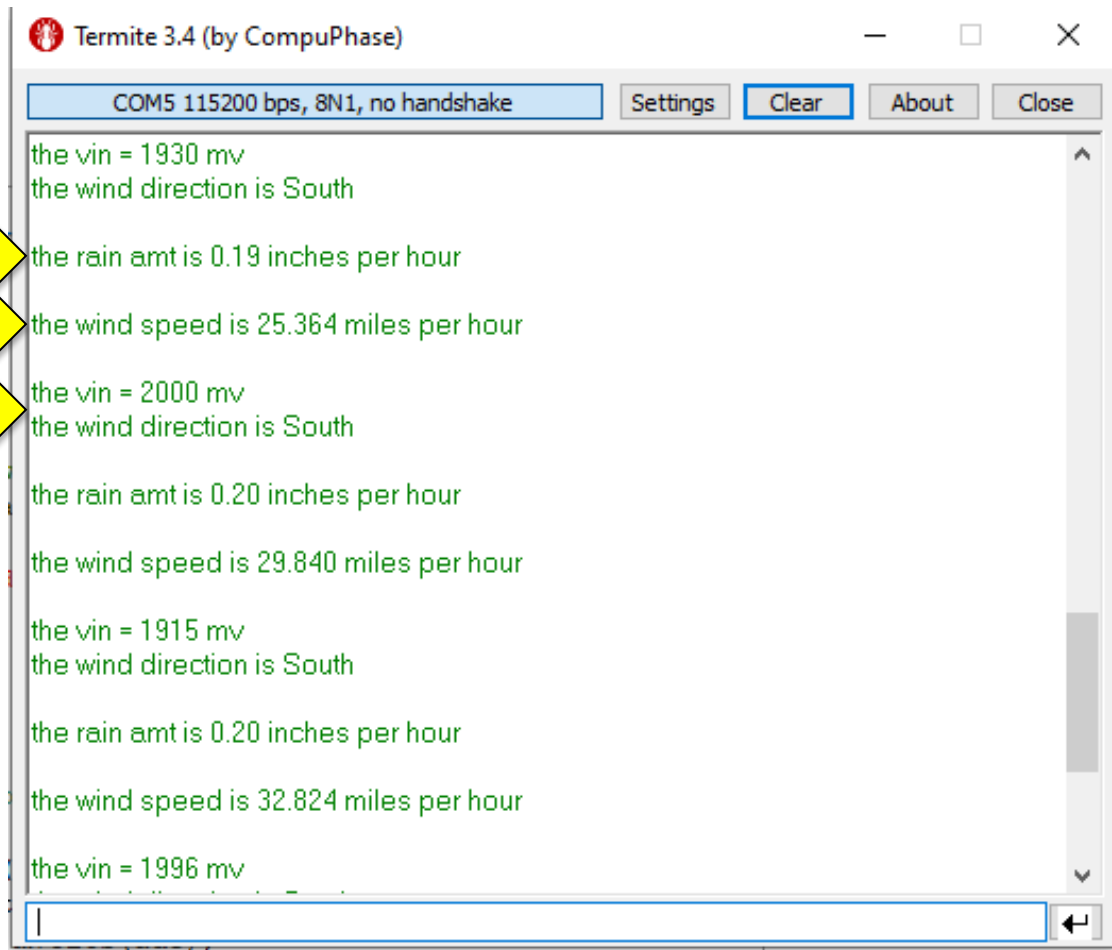
# Station 1 - Connection Diagram



# Station 1 – Testing results



# Station 1 – Testing Results cont.

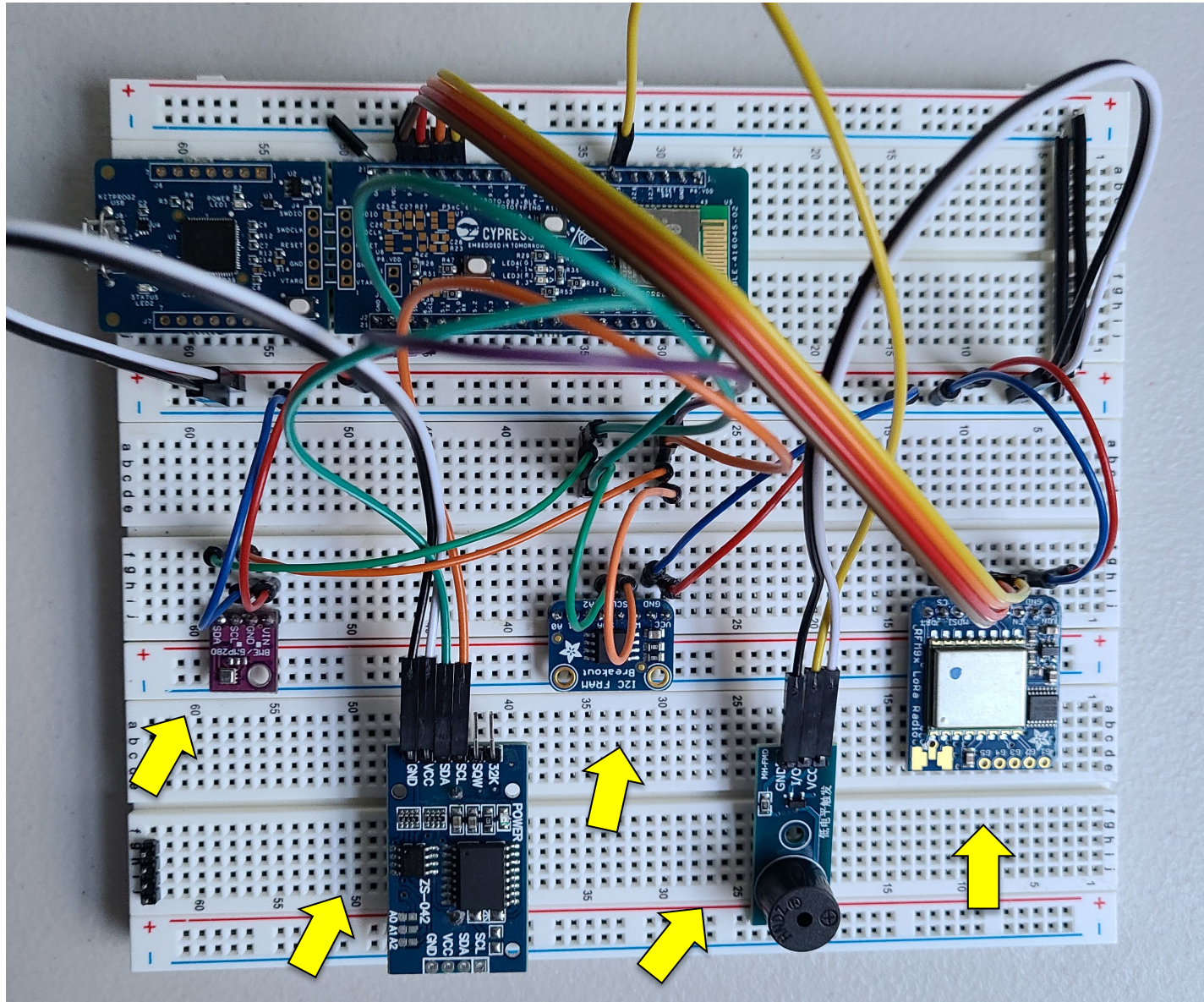


Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---





# Station 2



# Station 2 – Details

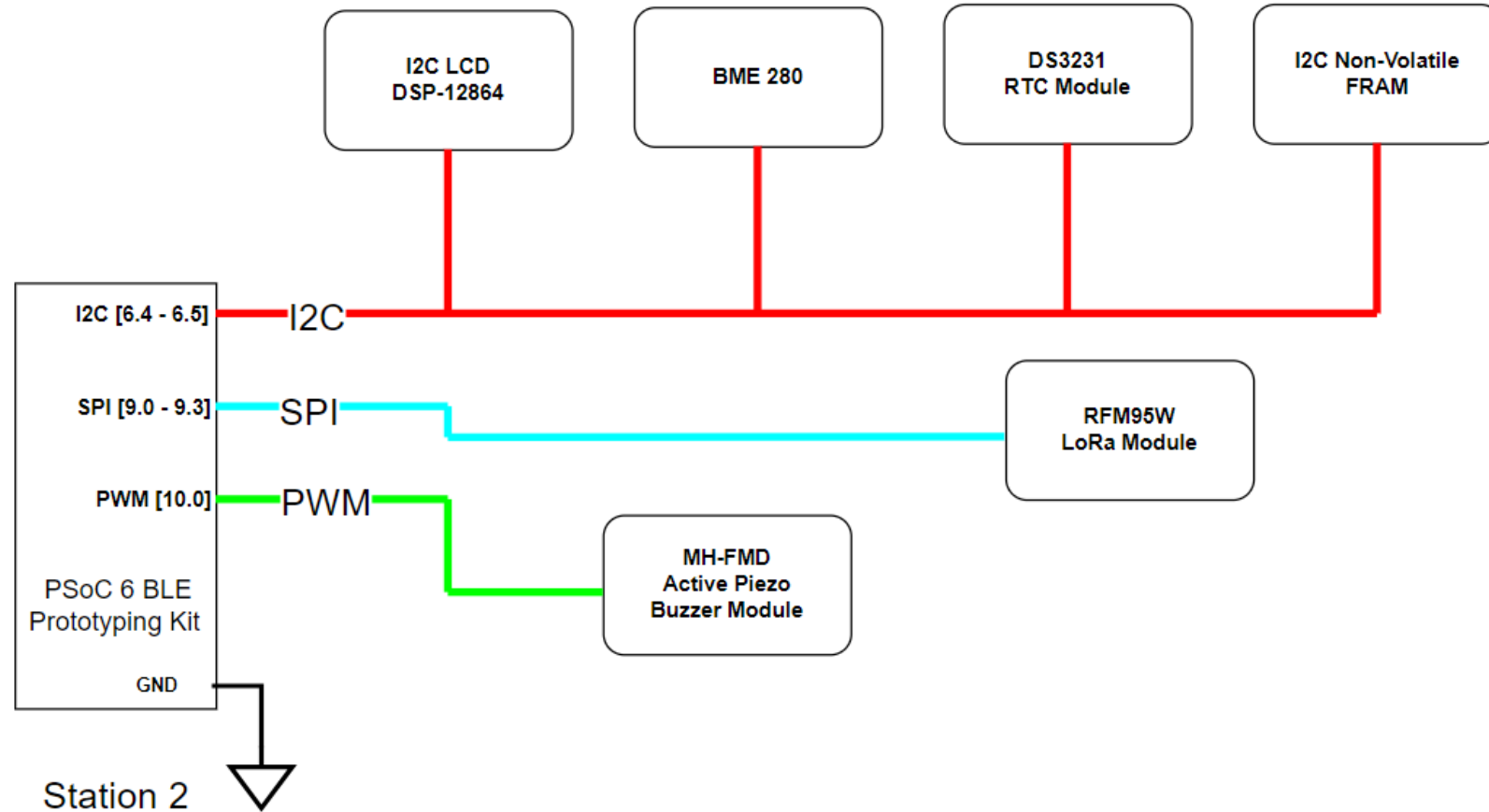
- Location:
  - Indoors Environments
- Data Displayed by LCD Screen
  - Set One:
    - Data from station 2 sensors
  - Set Two:
    - Data from Station 1 and 3
    - Sent from Central Station (Raspberry Pi) via LoRa Module



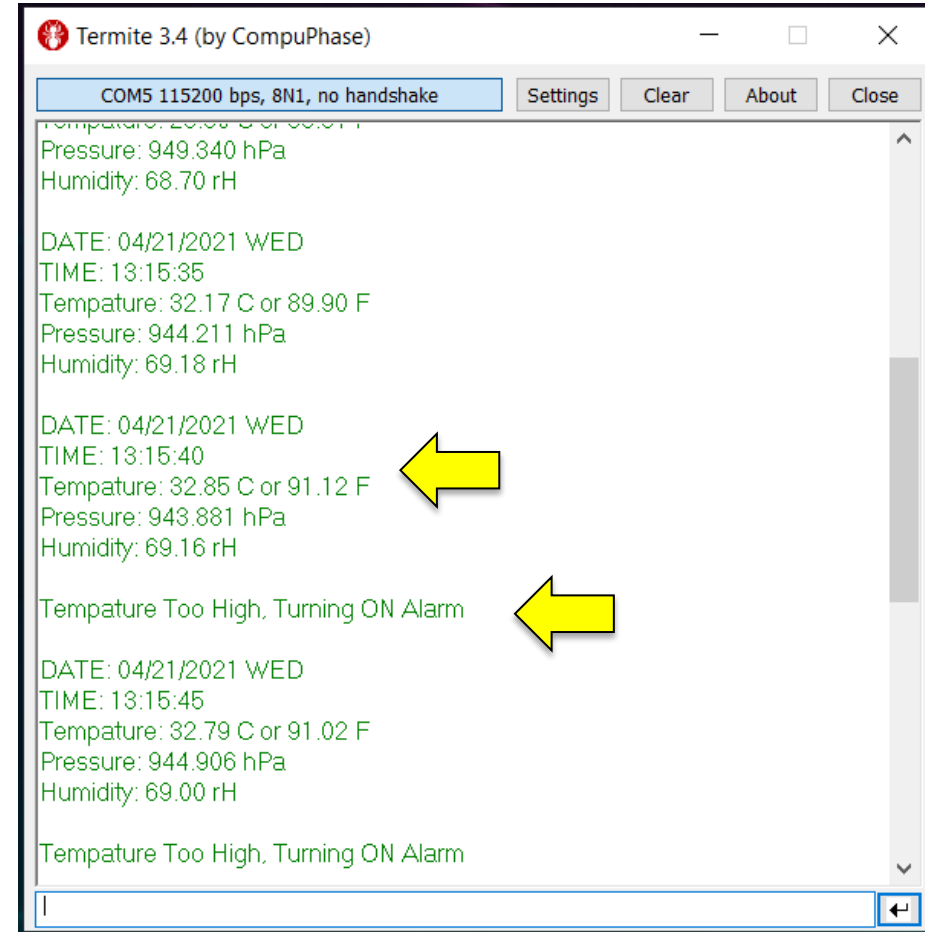
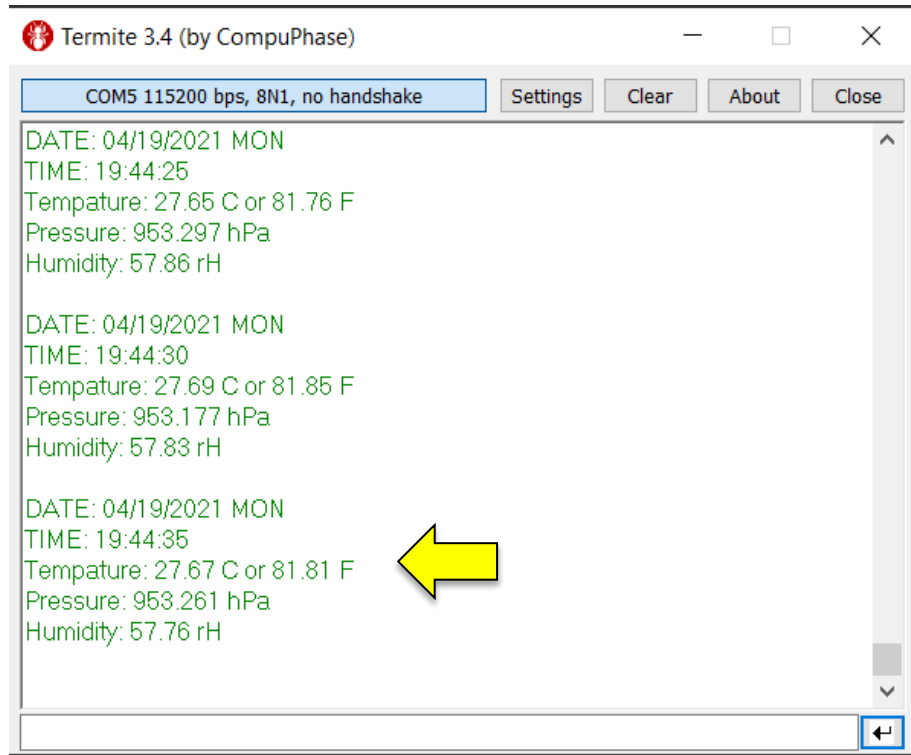
# Station 2 – Sensor Information

Microcontroller				
PSoC 6 BLE Prototyping Kit (CY8CPROTO-063-BLE)				
Sensor	Model	Voltage	Max Current	Interface
LCD Display	12864 LCD Display Module	3.3 – 5 V	66 mA	I2C
Temperature, Pressure & Humidity	BME 280	3.3 – 5 V	3.6 $\mu$ A	I2C
Buzzer Alarm Module	MH-FMD	3.3 – 5 V	30 mA	PWM
Real-time Clock	DS3231 RTC	2.3 – 5.5 V	200 $\mu$ A	I2C
Ferroelectric RAM	MB85RC256V FRAM Chip	2.7 – 5.5 V	200 $\mu$ A	I2C
LoRa Radio Transceiver	RFM95W LoRa Module	3.3 – 5 V	130 mA	SPI

# Station 2 – Connection Diagram



# Station 2 – Results



Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---



# Station 3

Micro controller	PSoC 5LP Prototyping Kit (Cy8CKIT-059)			
Component	Voltage	Current	Interface Type	Sensor Features
CJMCU-811 CCS811	1.8 - 3.6V	30mA	I2C	CO2 Sensor
MQ-4	5.0V	0.25mA	Analog	Gas, CNG Sensor
PPD42NJ	5V	2.5µA	ADC	Dust Particle Sensor
DSP-1182 LCD Screen	3.3 - 5V	66 mA	I2C	20x4 LCD Display
MB85RC256V	2.7 - 5.5V	200 µA	I2C	256K-Bit FRAM
RFM95W Lora Module	3.3-5V	130mA	SPI	RF long range transceiver

Figure 1: Station 3 system specifications

# Station 3 - Connection Diagram

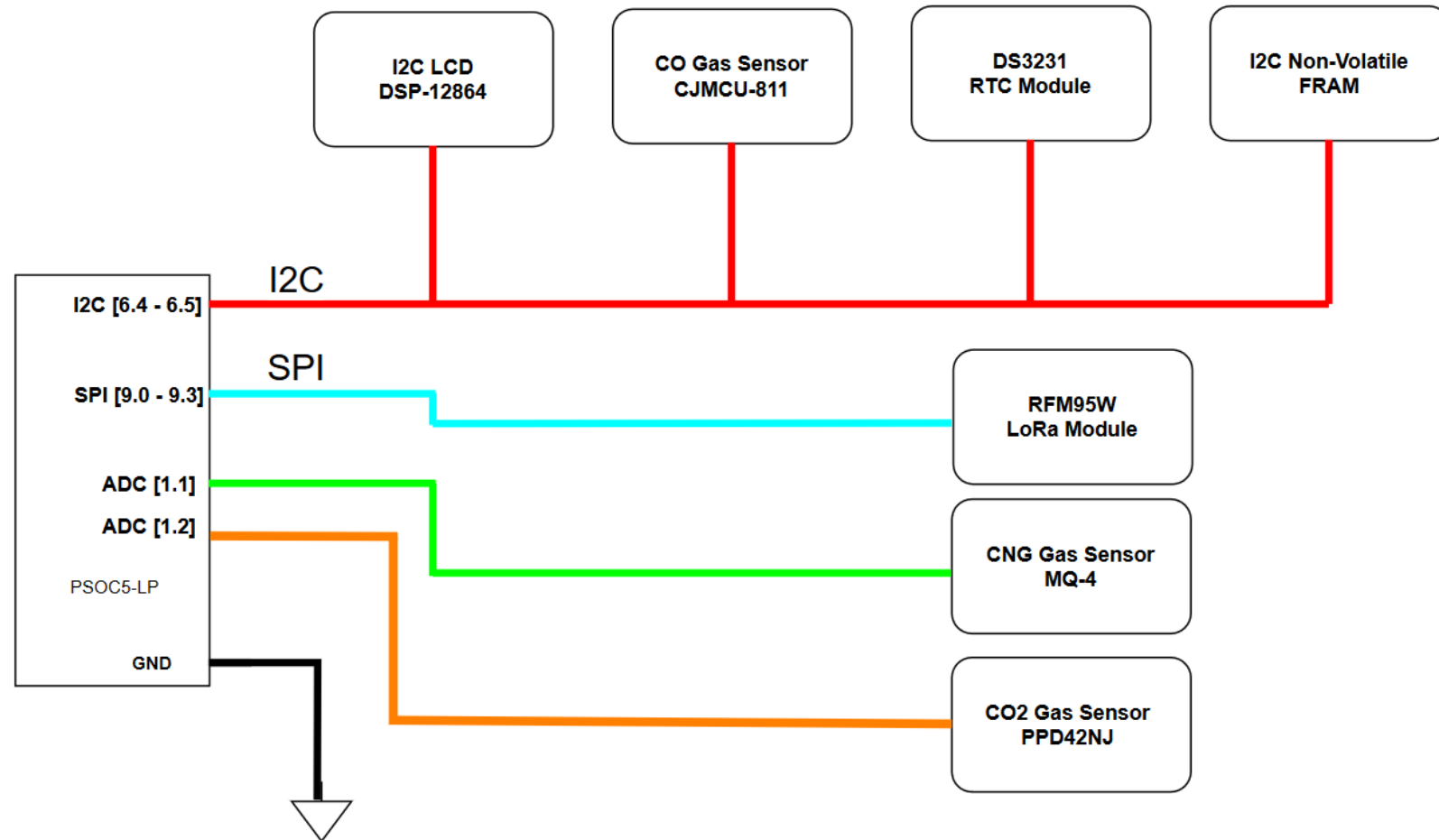


Figure 1: Station 3 Schematics



# Station 3 – Test Bench

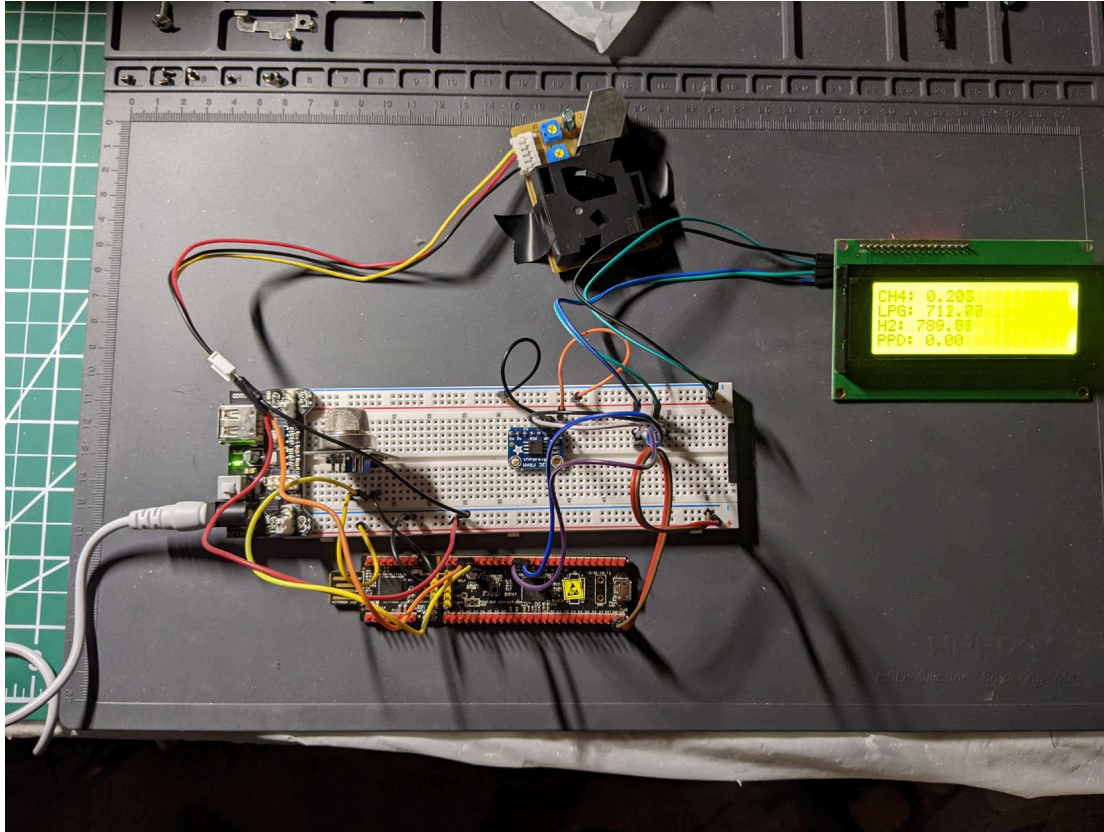


Figure 1: Station 3 test station

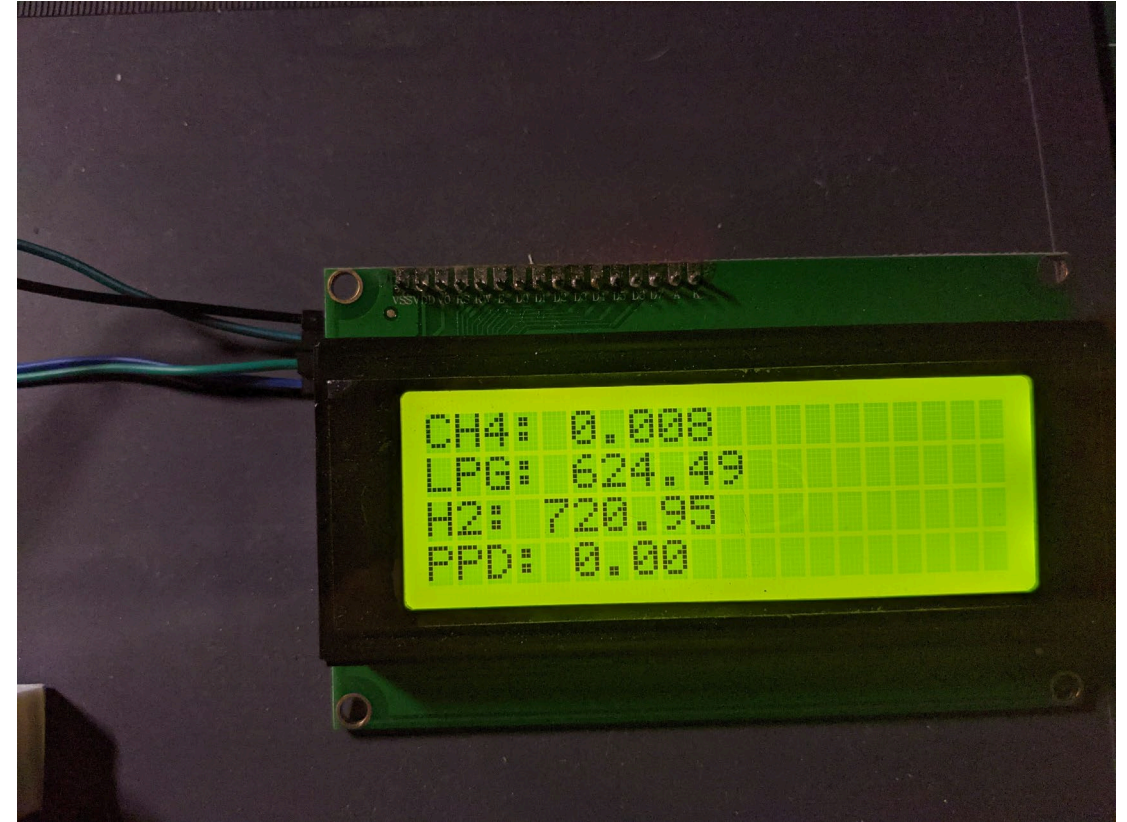
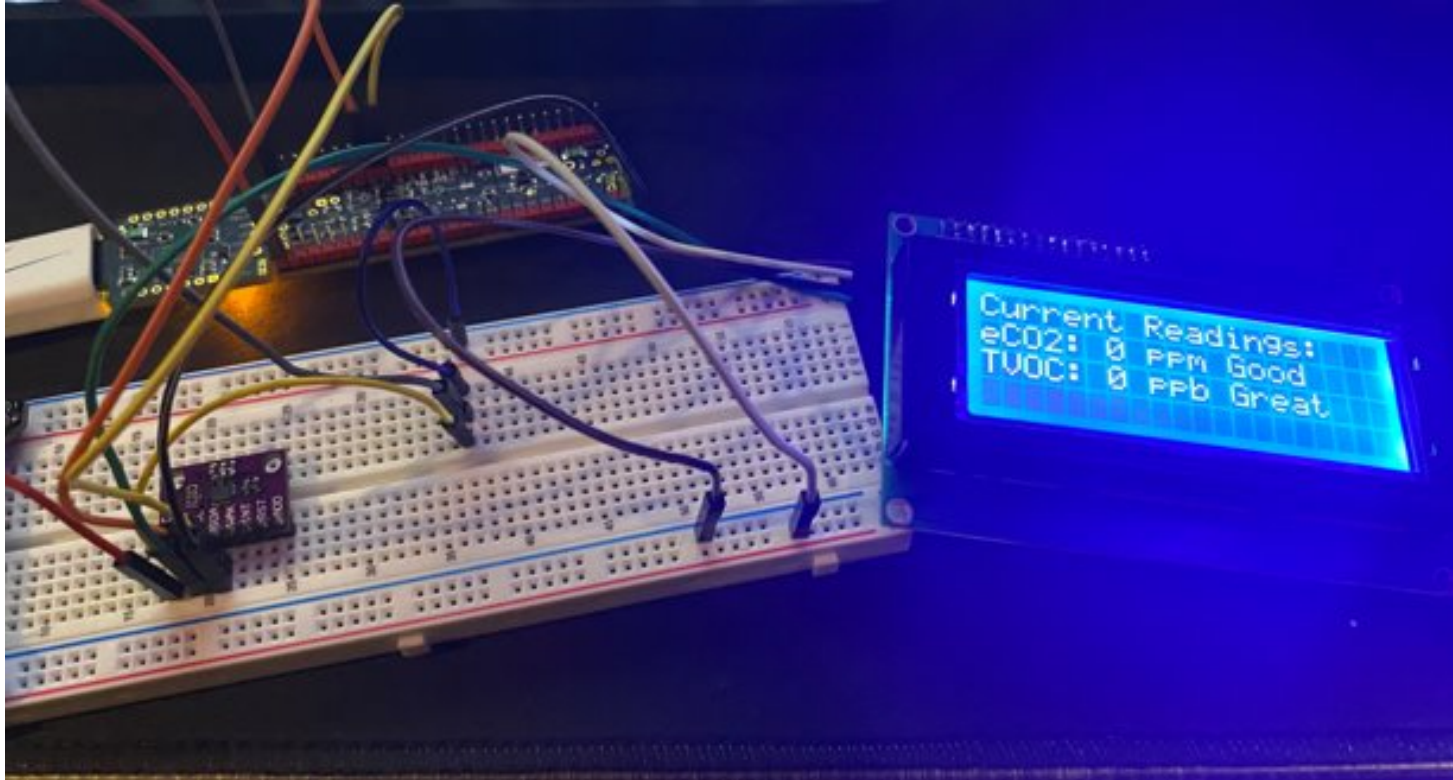
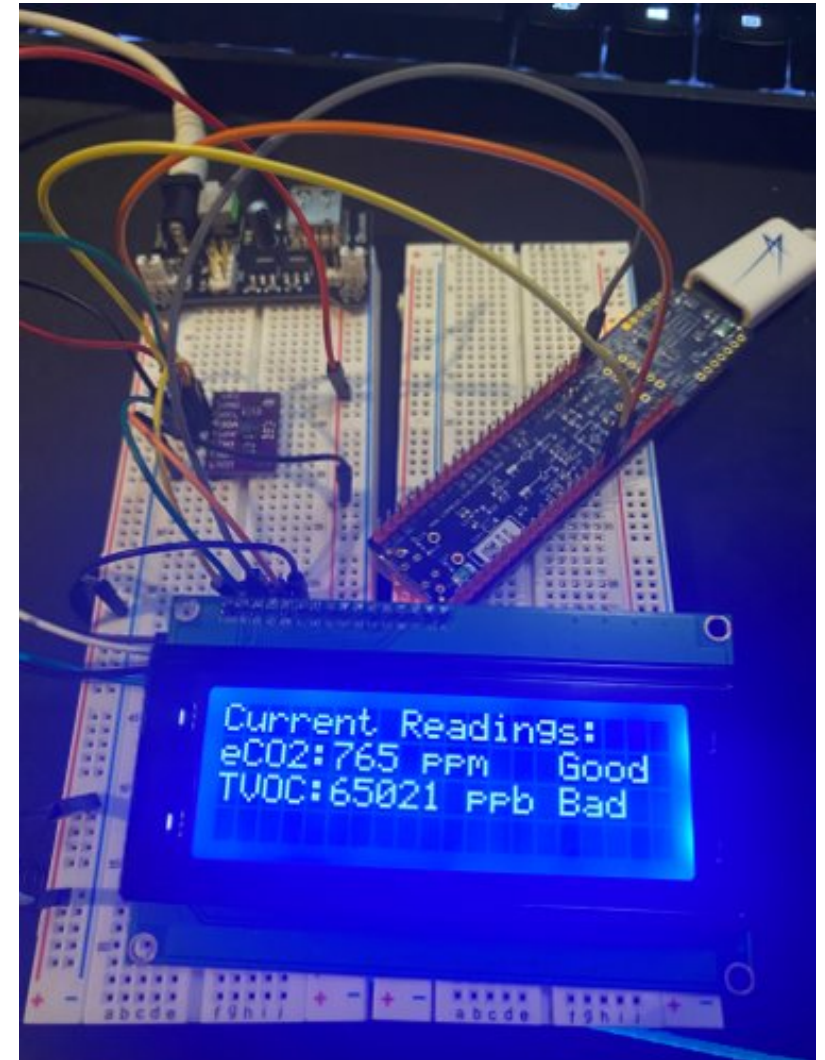


Figure 2: Close up of LCD output from station 3

## Station 3 – Test Bench



*Figure 1: CJMCU-811 sensor running with output*



*Figure 2: CJMCU-811 test bench*

# Station 3 - Results

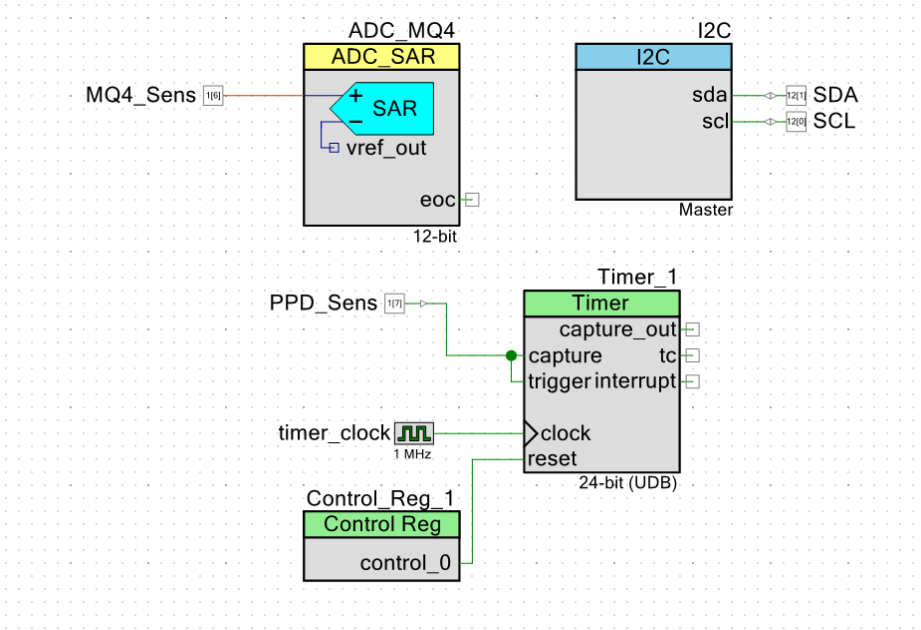


Figure 1: Top Design of Station 3

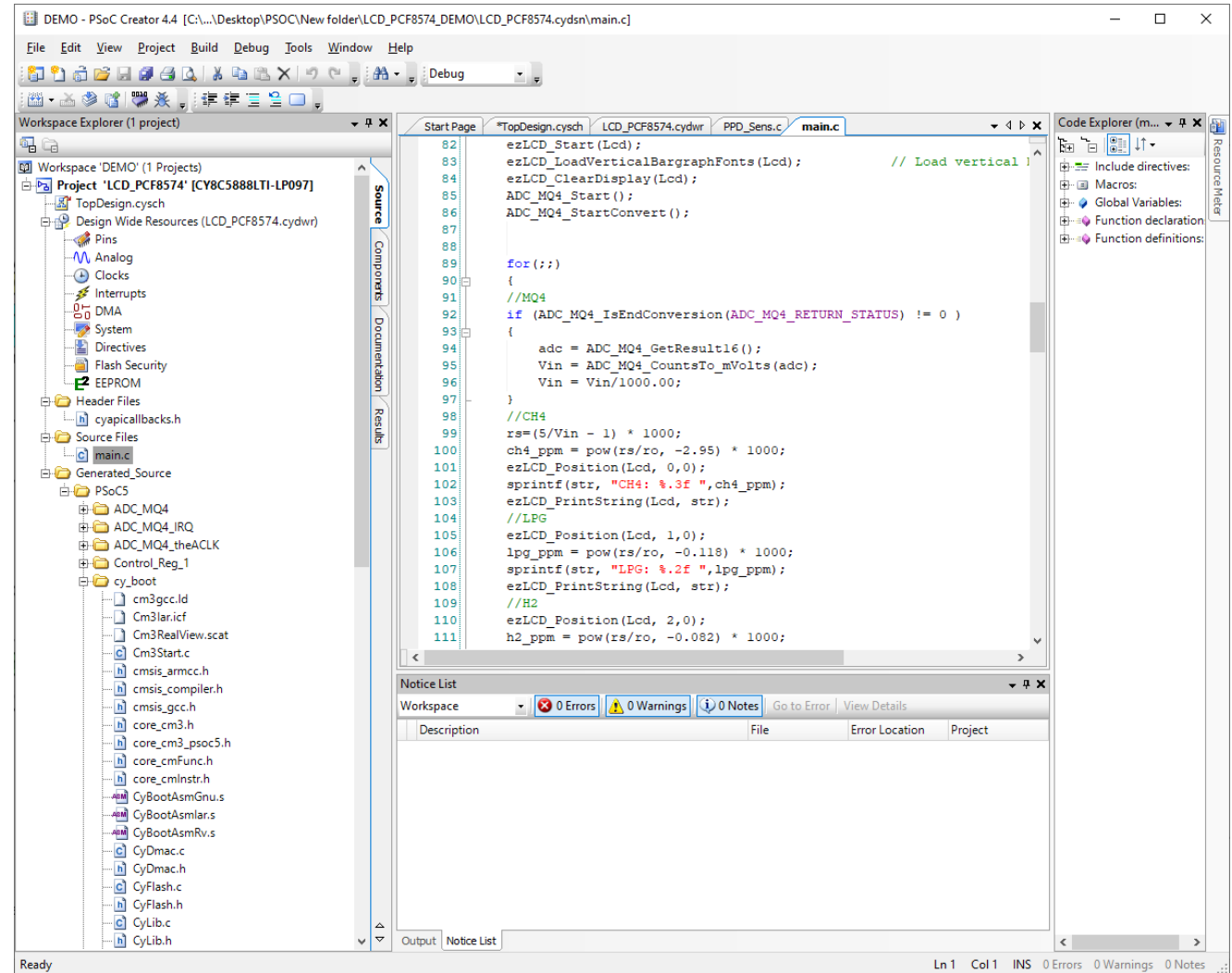


Figure 2: Runtime code of station 3

Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---



# AWS Amazon Web Services (A.W.S)

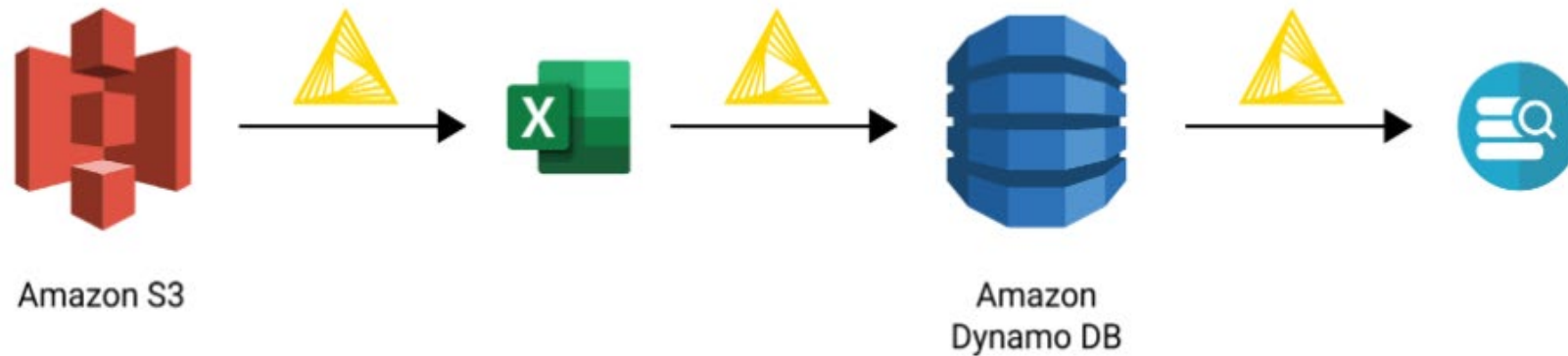
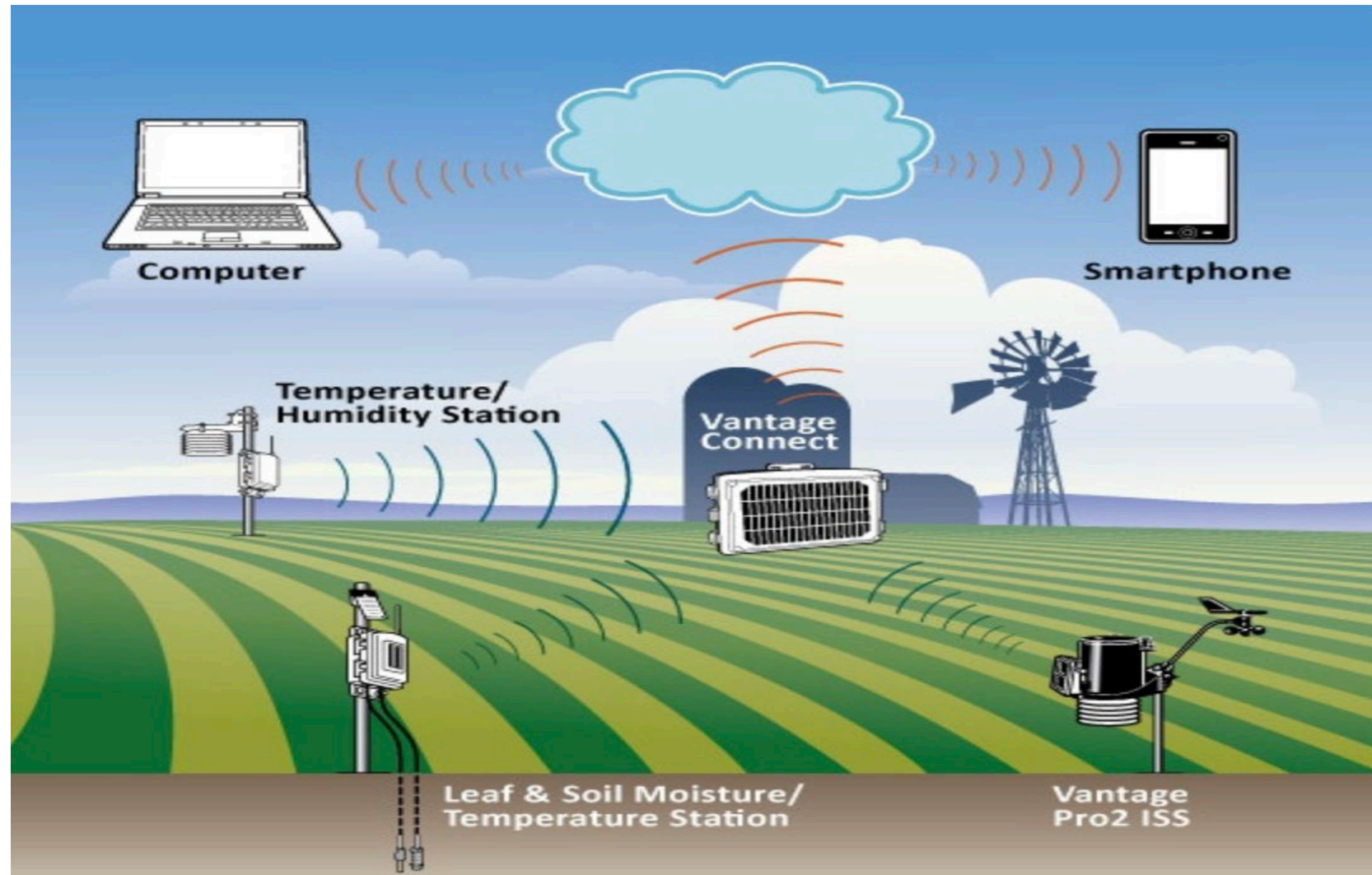


Figure 1

The image below (*figure1*) explains how data can be retrieved in real-time and can be stored in the database like Amazon Dynamo DB to help notify the humans about disastrous weather situations.

# INTERNET OF THINGS IOT



- ❑ The **Internet of Things (IoT)** refers to a system of interrelated, internet-connected objects that are able to collect and transfer data over a wireless network without human intervention.

# Storing data to Database

```
regions.py x _init_.py x client.py x session.py x
led import boto3

dynamodb = boto3.resource('dynamodb')
dynamodbTable = dynamodb.Table('test_table')

dynamodbTable.put_item(
#boto3.session('dynamodb', region name='us-west-2', endpoint url='https://18.191.139.28/2021/02/22/hello-world/')
# storing sensor data in the form of dictionary
Item={
'WeatherStation': 'Cal State Los Angeles',
'Station number': 1,
'Location': 'Los Angeles CA'
}
```

```
Text v [ ] [ ] DynamoDB JSON
1 {
2 "weather Station": "CalsateLa",
3 "temperature": "60 degrees",
4 "station number": "station 1"
5 }
```

- ❑ Storing data to AWS by using python SDK.
- ❑ Data storage format are structured in strings.

Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
Conclusion	---





# Raspberry Pi 4

- In order to process the data set packets we used Python with Visual Studio Code.
- In order to connect to AWS IoT Core, we download necessary libraries from Raspberry Pi command terminal.
- Create certificates public and private keys.

```
# For TLS mutual authentication
myMQTTClient = AWSIoTMQTTClient("DesignID")
myMQTTClient.configureEndpoint("a5e1cu78wbs20-ats.iot.us-west-1.amazonaws.com", 8883) #Provide your AWS IoT Core endpoint (Example: "abcdef1
myMQTTClient.configureCredentials("/home/pi/certs/Amazon-root-CA-1.pem", "/home/pi/certs/private.pem.key", "/home/pi/certs/device.pem.crt")
myMQTTClient.configureOfflinePublishQueueing(-1)
myMQTTClient.configureDrainingFrequency(2)
myMQTTClient.configureConnectDisconnectTimeout(10)
myMQTTClient.configureMQTTOperationTimeout(5)
```

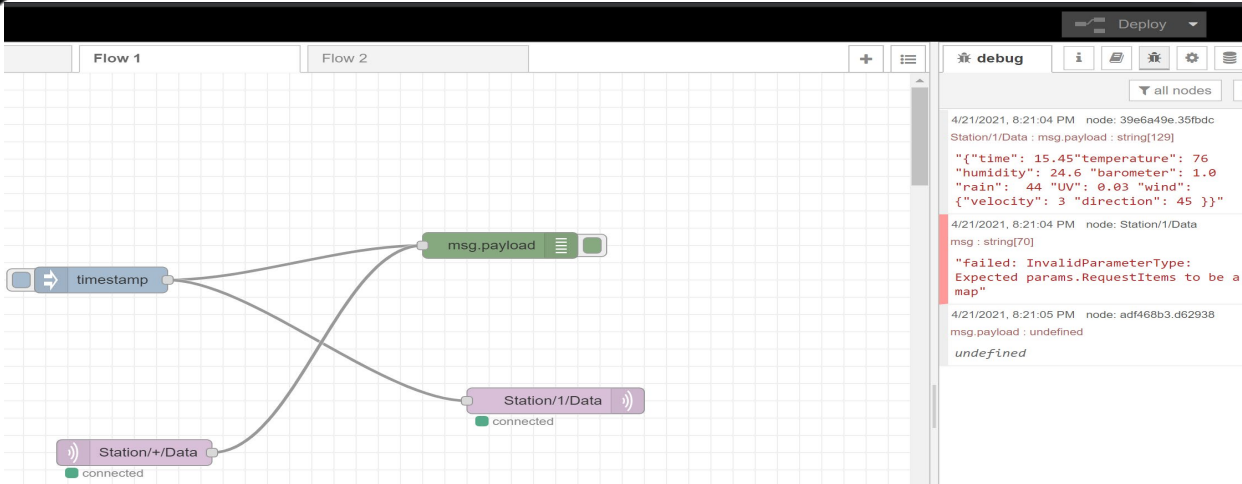
# Data Set Notation

```
test_1 = 'Station_1| "time": 15.45, "velocity": 3 , "direction": 45 , "rain": 44 , "UV": ' \
| '0.03 , "temperature": 76 , "humidity": 24.6 , "barometer": 1.0 '
test_2 = 'Station_2| "time": 15.46 , "temperature": 76 , "humidity": 27.6 , "barometer": 2.0 '
test_3 = 'Station_3| "time": 15.47 , "methane": 23 , "particle": 66 , "cs811": -127 '
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
pi@raspberrypi:~/aws-iot-device-sdk-python-v2 $ cd /home/pi/aws-iot-device-sdk-python-v2 ; /usr/bin/env /
b/python/debugpy/launcher 33191 -- /home/pi/aws-iot-device-sdk-python-v2/topics2.py
Initiating IoT Core Topic ...
Station_1
"time": 15.45
"velocity": 3 km/hr
"direction": 45 degrees
"rain": 44 mm
"UV": 0.03
"temperature": 76 C
"humidity": 24.6 %
"barometer": 1.0 psi
Publishing Message from Raspberry PI
pi@raspberrypi:~/aws-iot-device-sdk-python-v2 $
```

- The string must be formatted in a particular structure.
- Quotation marks on the environmental variable.
- String will be sent without variable units.

# Sending Data Sets



```
debug
all nodes
4/21/2021, 8:21:04 PM node: 39e6a49e.35fbd
Station/1/Data : msg.payload : string[129]
{"time": 15.45 "temperature": 76
"humidity": 24.6 "barometer": 1.0
"rain": 44 "UV": 0.03 "wind":
{"velocity": 3 "direction": 45 }}
4/21/2021, 8:21:04 PM node: Station/1/Data
msg : string[70]
"failed: InvalidParameterType:
Expected params.RequestItems to be a
map"
4/21/2021, 8:21:05 PM node: adf468b3.d62938
msg.payload : undefined
undefined
```

## MQTT test client [info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

[Subscribe to a topic](#) | [Publish to a topic](#)

### Topic filter [Info](#)

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

Station/+Data

► Additional configuration

Subscribe

### Subscriptions

Station/+Data

Pause Clear Export Edit

Station/+Data

▼ Station/1/Data

April 21, 2021, 20:54:18 (UTC-0700)

Message cannot be displayed in specified format.

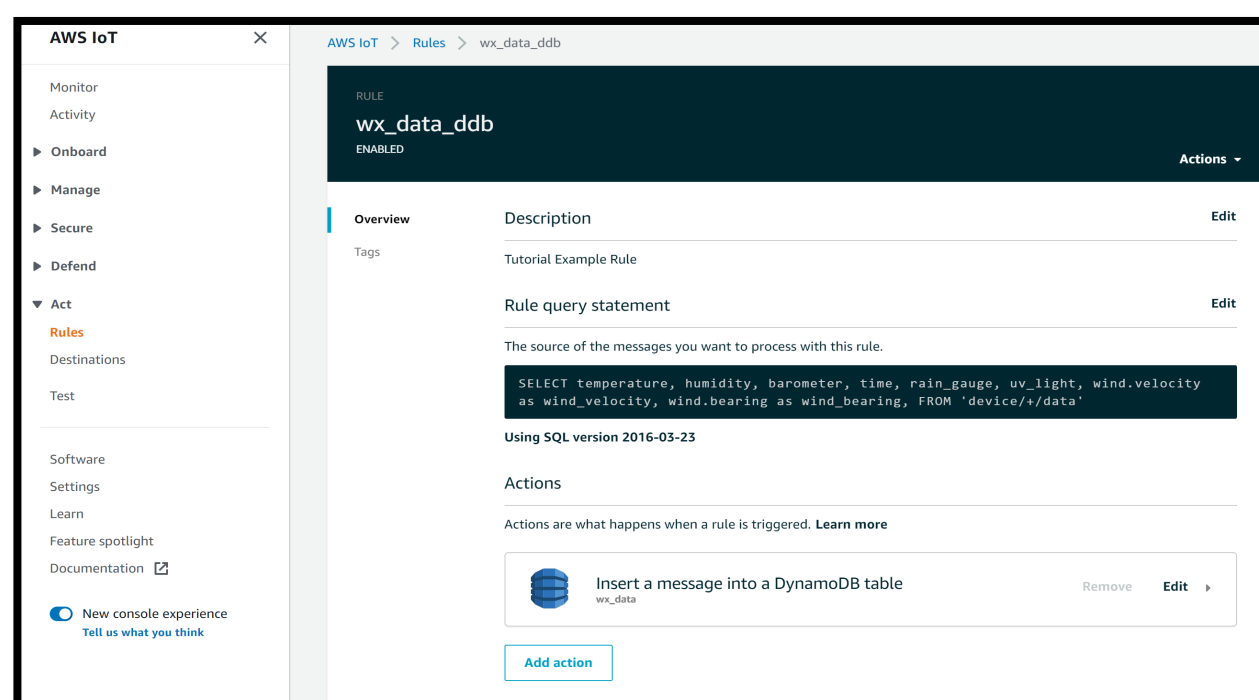
```
{"time": 15.45 "temperature": 76 "humidity": 24.6 "barometer": 1.0 "rain": 44 "UV": 0.03 "wind":{"velocity": 3 "direction": 45 }}
```

- To successfully identify if a data set was sent, we used Node Red.
- In order to receive the data sets, we created topics for the stations to subscribe to and publish data.
- Within these topics there are rules that detect key words or variables.



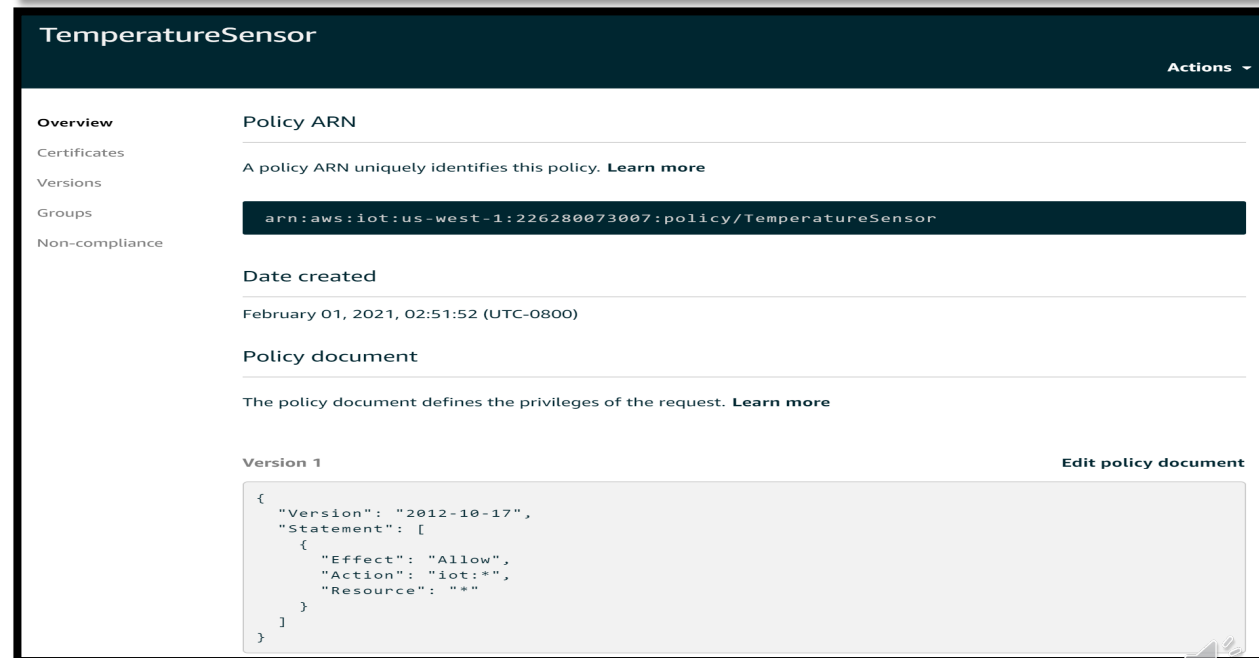
# IoT Core

- IoT Core allows our team to create rules which can identify environmental variables from a particular topic.
- Once variables are identified from MQTT client the rules in place will store the data sets into DynamoDB.
- In order for the Raspberry Pi to subscribe to IoT Core a policy must be created.



The screenshot shows the AWS IoT console interface for configuring a rule. The left sidebar contains navigation options: Monitor, Activity, Onboard, Manage, Secure, Defend, Act (with 'Rules' highlighted), Destinations, and Test. Below these are links for Software, Settings, Learn, Feature spotlight, and Documentation. A 'New console experience' toggle is also visible.

The main content area displays the configuration for the rule 'wx\_data\_ddb', which is currently 'ENABLED'. It includes an 'Overview' section with a 'Description' (Tutorial Example Rule) and 'Tags'. The 'Rule query statement' section shows a SQL query: `SELECT temperature, humidity, barometer, time, rain_gauge, uv_light, wind.velocity as wind_velocity, wind.bearing as wind_bearing, FROM 'device+/data'`. Below the query, it specifies 'Using SQL version 2016-03-23'. The 'Actions' section shows a single action: 'Insert a message into a DynamoDB table' with the identifier 'wx\_data'. An 'Add action' button is located at the bottom of the actions list.



The screenshot shows the AWS IAM console configuration for an IoT policy named 'TemperatureSensor'. The left sidebar lists 'Overview', 'Certificates', 'Versions', 'Groups', and 'Non-compliance'. The main content area shows the 'Policy ARN' as `arn:aws:iot:us-west-1:226280073007:policy/TemperatureSensor`. It also displays the 'Date created' as 'February 01, 2021, 02:51:52 (UTC-0800)'. The 'Policy document' section shows the JSON policy document for 'Version 1':

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

An 'Edit policy document' link is provided next to the version information.



# Data Base

- DynamoDB is used to store the sets according to the station number and environmental variables.

Stations Close

Overview Items Metrics Alarms Capacity Indexes Global Tables Backups Contributor Insights More

Create item Actions

Scan: [Table] Stations: Sample\_Time, Station Viewing 1 to 26

Scan [Table] Stations: Sample\_Time, Station Add filter Start search

Sample_Time	Station	Station_Data
1618784612749	1	{"barometer": {"N": "1"}, "humidity": {"N": "24.6"}, "rain": {"N": "44"}, ...
1618784648604	1	{"barometer": {"N": "1"}, "humidity": {"N": "24.6"}, "rain": {"N": "44"}, ...
1618977171620	2	{"barometer": {"N": "2"}, "humidity": {"N": "27.6"}, "temperature": {"N": ...
1618994787095	3	{"methane": {"N": "23"}, "particle": {"N": "66"}, "time": {"N": "15.47"}}
1618994898641	3	{"cs811": {"N": "-127"}, "methane": {"N": "23"}, "particle": {"N": "66"}, ...
1618994919774	3	{"cs811": {"N": "-127"}, "methane": {"N": "23"}, "particle": {"N": "66"}, ...

Edit item

Tree

- Item {3}
- Sample\_Time Number : 1618784648604
- Station Number : 1
- Station\_Data Map {8}
- barometer Number : 1
- humidity Number : 24.6
- rain Number : 44
- temperature Number : 76
- time Number : 15.45
- UV Number : 0.03
- wind\_direction Number : 45
- wind\_velocity Number : 3

Edit item

Tree

- Item {3}
- Sample\_Time Number : 1618977171620
- Station Number : 2
- Station\_Data Map {4}
- barometer Number : 2
- humidity Number : 27.6
- temperature Number : 76
- time Number : 15.46

Edit item

Tree

- Item {3}
- Sample\_Time Number : 1618994787095
- Station Number : 3
- Station\_Data Map {3}
- methane Number : 23
- particle Number : 66
- time Number : 15.47

Agenda	Team Member
Backgrounds, Problem, and Objectives	Juan Avila
System Overview	David Hernandez
Implementation – Station 1	Anthony Huynh
Implementation – Station 2	Jeffrey Espinoza
Implementation – Station 3	Vincent Oviedo
Implementation – AWS database/Raspberry Pi Part 1	Akbar Rizvi
Implementation – AWS database/Raspberry Pi Part 2	Jose Rodriguez
<b>Conclusion</b>	<b>Juan Avila</b>



# Conclusion – Accomplishments

- Built 4 stations and a database
- Build a wireless network of sensors
- Collect indoor and outdoor environmental data



# Conclusion – Issues

- Limited due to Covid19 and remote learning, we had some limitations on what we could implement





# Conclusion – Going Forward

- With the way we designed it we can add more different sensor types or more stations for scalability





# Group 25: IoT Environmental Sensors

# Thank you

Any Questions?

