INTERNET OF THINGS

CSCE 838

By

Group 1

Milestone 4: Final Paper

Ebuka Philip Oguchi

Junxiao Zhang

Md Didarul Islam

COURSE INSTRUCTOR: Dr. Mehmet C. Vuran

LAB INSTRUCTOR: Mohammad Mosiur Lunar

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

SCHOOL: UNIVERSITY OF NEBRASKA, LINCOLN

DATE: 12/17/21

Project Topic: A Sensor Data Gathering, and Cloud based Transmitter Prototype System

Project Description:

This project focuses on how to collect data from different sensors (like Temperature sensors and Soil moisture sensors) and detect the sensor types based on their different identification voltages. The main goal for our project is to develop a prototype system that could automatically detect the sensor type and send the correct value and unit (V or A) to the remote end via cloud. Also, the sensor reading uploaded to the cloud must be from the identified transmitter to avoid interferences from other transmitters.

Table of changes:

MS	Comments	Addressing comments	Change made in	Date
MS0	Vague title	Changed to: A Sensor Data Gathering, and Cloud based Transmission Prototype System	MS1	10/29/21
MS0	Label each requirement with unique identifier	Done	MS1	10/29/21
MS0	Including table for tracked changes	Done	MS1	10/29/21
MS0	Clarify the project scope	Done	MS1	10/29/21
MS1	Most engineering requirements should be derived from customer requirements. Each CR needs to be referred to in at least one ER. There may be ERs that are internal to the system and don't refer to any CR. In that case, it should refer to another ER. Please update it in the final paper.	All ERs are linked with relevant CRs	MS4	12/09/21
MS1	Style sheet should include a portion for the software code. For example, how and when are comments included? What are the conventions for variable, function, etc, names? Other considerations? Please update in your final paper.	Addressed	MS4	12/09/21
MS1	(Test plan) Need an owner for each test. Who will conduct the test? Who will approve pass? I don't see any papers produced, or owners and consumers of papers. Please update in the final paper.	Test is assigned to group members based on the work distribution (staffing plan). A test is considered 'pass' when all the group members agreed on that.	MS4	12/09/21
MS1	Schedule and staffing plan should include more than just milestones. The team needs intermediate milestones to	Task is assigned to specific group members. However, there are some joint tasks which must be conducted by	MS2	11/19/21

	ensure progress. I do not see a staffing plan; it is not clear who is responsible for which task. Please update me in the final report.	collaboration of group members		
MS3	No changes are listed here, even though a few changes with respect to MS2 can be observed. Table changes need a column for date of change	Table has been updated as instructed	MS4	12/09/21

1. Customer Requirements

Customer Expectations:

The customer is a researcher performing research in an agricultural field.

CR-1: Customer expects a **plug and play device** that can take temperature, soil moisture readings once every week, upload it to Microsoft Azure cloud and then notify the researcher if messages are missing.

CR-2: Device must have batteries that are easily **replaceable** and **rechargeable and have a good life span.**

CR-3: The device should be **durable,** i.e., it is resistance to dust and splash of water (IP67 rating)**, reliable** (i.e., up to 95 percent of the measurements get to the clouds).

CR-4: The device should be easy to use i.e., the user can more easily update most functionalities in the system whenever he chooses to. The device should be able to work with various types of soil moisture and temperature sensors.

CR-5: This device should be **modular in size, easy to adjust to (**sensors can be easily swapped when necessary**)** and takes **fast frequency readings**.

CR-6: This device should be able to work **indoors** and **outdoors**.

CR-7: The device and related maintenance should be **cost efficient**.

Constraints on the system

C1: Devices can only adapt with known sensors

C2: The system is only able to detect voltages between 0-5V

C3: Those data are going to be sent to Microsoft Azure only

Success criteria:

Serial No.	Success Criteria	Definition
SC-1	Usability	Customers should be able to gather data at a fast-receiving speed (real-time) from Microsoft Azure.
SC-2	Stability	The device can be powered by batteries.
SC-3	Cost	The Cost of total should be restricted to $100 or less.
SC-4	Long range communication	The device can communicate at a distance above 50m using the omnidirectional antenna. [2], [3]
SC-5	Universality	The device can identify temperature and soil moisture sensor automatically.

Timeline: Customer expects a working plug and play sensor value transmitter that can upload data readings to the cloud by 12/17/2021(Final Demo Presentation).

User Guide

A Sensor Data Gathering, and Cloud based Transmitter Prototype System

Disclaimer

This manual is for the operational setup and maintenance of “Sensor Data Gathering, and Cloud based Transmission Prototype System” which is a course project for CSCE 838. It is created for educational purposes as it is only a prototype of the main design. Please read and adhere to the setup up procedures to ensure proper transmission and capture of the required information from this machine.

All inquiries regarding this handbook should be addressed to the project team (Group 1)

User Guide Table of Contents

Setup Procedure:

1. Connect batteries to the Transmission Device (SparkFun SAMD21) and the SparkFun ESP32 gateway
2. Connect the antennas of the transmitter and the SparkFun ESP32 gateway
3. Connect the required sensor (Temperature sensor or soil Moisture sensor) one at a time to the transmitter
4. Turn on the transmitter by pressing the power button
5. Connect battery to the sensor
7. Register and Login into your Microsoft Azure cloud service
8. Set up your device (on cloud) to receive data from the transmitter.

Changing Batteries:

1. Check if the battery is connected.
2. Turn off the transmitter
3. Ensure that the devices are properly powered off
4. Remove the old battery
5. Replace the new battery (Ensure that the battery replacement is of the same specification and layout of the original battery).

Maintenance:

1. Always place equipment in an undamped, clean, and clear place.
2. Always clean with dry clothes
3. Always unplug the battery and sensors when not in use to save energy

Troubleshooting:

Device Unresponsive:

1. Check whether the antennas are properly connected to the device
2. Unplug and plug again
3. Checky whether the sensor devices are properly connected to the universal sensor value transmitter
4. Check the connection string used
5. Check the Wi-Fi connectivity
6. Check the battery to see whether it still has energy left and replace it if the battery has drain

No Sensor value in Microsoft Azure cloud:

1. Reload the Microsoft Azure webpage
2. Power on and off the transmitter
3. Check if the devices in the IoT Central App are accurately configured to receive the required data
4. Check whether the sensors connected are the appropriate one (e.g., temperature, and soil moisture sensor).

Engineering Requirements

Overview:

Constructing a data collection and transmission device that will be compatible with most sensors in the field. Specifically, soil moisture sensor and temperature. The main purpose of the device will be to; Get Sensor reading as input and transmit the sensor data to the remote cloud server.

The secondary objective of the device will be to get one sensor reading at a time as we are expecting to use multiple sensors in the field. The device will be able to tell what type of sensor is being used once the sensor is plugged into the device and pass that information to the remote end. This will eliminate the human error of not knowing what type of sensor is plugged in the field side. Also, the technical knowledge of explicitly programming the system will not be required to see (from the remote end) what type of sensor is plugged in every time, as this part will be handled by hardware circuitry and will be a plug and play device.

The third function of the device is error logging, the device should be able to send error information to users when there is a crash or error happening.

Engineering Requirements Table:

Serial No.	Title	Derived from CR	Description
ER-1	Cost	CR-7	Based on the BOM Cost of finished goods Budget for major components
ER-2	Software requirements	CR-1, CR-4	The Prototype System will be programmed in C++ or python, The cloud service is Microsoft Azure, Watchdog timer, Error logging and Transmit functions
ER-3	Usage Environment	CR-6	Indoors and outdoors, specifically in the Laboratory.
ER-4	Battery life span	CR-2	0V – 5V DC battery [1] 5 years battery life [1]
ER-5	Communication	CR-5, CR-6	Antennas, SparkFun ESP32 gateway,

			SparkFun SAMD21/transmitter (Battery, push power button, battery case, antenna, sensor device ports)
ER-6	Fast response	CR-5	Should be Less than 5 minutes to read and upload data [4]
ER-7	Robustness	CR-1, CR-4, CR-5	Built in watch dog timer
ER-8	Continuous improvement	CR-1, CR-4, CR-5	Run time error logging
ER-9	Safety	CR-3	Data gathering and transmitting device placed in a proper package with correct layout. Use Identification Code in packet to avoid interferences from other transmitters

List of Error Logging tests are as follows:

1. Time stamp
2. System resets
3. Runtime errors
4. Assertion violations
5. Stack status
6. Hardware failures
7. Peripheral equipment failures
8. Operating conditions

1. Implementation – style sheet

Style sheet for software code:

Code header

All variables and functions need to be defined at the head of code, including all the dictionaries used in the code. All these variables and functions need to have an appropriate comment.

Code

Each function needs to be simplified and avoid including too many functions. Also, leave a blank line between each function to keep the program easy to read and understand.

Comments style sheet:

How to comment:

Before each segment of code, a comment section has been used. Also, inside the code segment, after each line comments are used to clarify the function of that code line.

When to comment:

We have used comments after almost every line of code, so that it is understandable to other group members as well as to the course instructor.

1. Header comment
    a. It needs to include the authors' names, dates, and the course number
    b. A brief description of the use of the program
    c. Introduce the version of the program and any bug is fixed
    d. A list of variables and definitions are using in the program
2. Function
    a. A brief description before the function to illustrate the function
    b. If there is something unfinished or hard to understand, make a comment after that to remind team members

Convention for names:

Most of the cases, we have used variable and function names that are self-explanatory. For names with multiple words, we have either used underscore or capitalized each word.

Convention in circuit diagrams:

a. x.y DC voltage represented as $V_{x.y}$
b. Ground represented as GND

Test plan:

We know that the system by carrying out the various tests outlined below and the strategy for this test can be:

1. **Design verification or compliance Test:** This test will be done during the development or approval stage of the Prototype System.

2. **Manufacturing or production test:** This test will be done during the preparation or assembly of the Prototype System.

3. **Acceptance or commissioning test:** This will be done during the time of delivery or installation of the Prototype System.

4. **Regression test:** This will be done to ensure the new functionality of the platform is not affecting other aspects of the Prototype System (e.g., upgrading the platform on which the application runs).

During the design and construction of the Prototype System (A Sensor Data Gathering, and Cloud based Transmitter Prototype System), we are going to carry out the various tests below to ensure that. The engineering requirements and the customer requirements are met.

Engineering Requirements Tests:

Planned Date	**Setup Test**	**Assigned group member**	**No. of trials**	**Passed**	**Failed**	**Not yet run**	**Comment**
12/02/21	Power on Test	Md Didarul Islam/ Ebuka Philip Oguchi/ Junxiao Zhang	20	20	0		Passed
12/03/21	Error logging Test	Ebuka Philip Oguchi	4	4	0		Passed
12/04/21	Communication Test	Junxiao Zhang	30	30	0		Passed
12/05/21	Universal sensor test	Md Didarul Islam	20	20	0		Passed

12/10/21	Identification Test	Junxiao Zhang	10	10	0		Passed

Customer Requirement Tests:

Planned Date	Setup Test	Assigned group member	No. of trials	Passed	Failed	Not yet run	Comment
12/09/21	Outdoor and Indoor test	Ebuka Philip Oguchi / Junxiao Zhang/Md Didarul	10	10	0		Passed
12/09/21	Ease of use test	Junxiao Zhang	20	20	0		Passed
12/09/21	Durability and Reliability test	Ebuka Philip Oguchi	20	20	0		Passed
12/09/21	Plug and Play Test	Md Didarul Islam	20	20	0		Passed

Scheduling

Assignments	Planned date	Actual date
Milestone 1: Gate Review	10/29/2021	10/29/2021
Milestone 2: Mid-term Gate Review	11/19/2021	11/19/2021
Milestone 3: Pre-Demo	12/03/2021	12/03/2021
Milestone 4: Final Demo	12/14/2021	12/14/2021
Milestone 5: Final paper	12/17/2021	12/17/2021

Staffing plan

Group Members	Role
Ebuka Philip Oguchi	Watchdog Timer, Error Logging, and Physical packaging
Junxiao Zhang	Transmission and cloud Service, and Package identification
Md Didarul Islam	Hardware design, Hardware implementation and sensor data gathering, Observation of the effect of identification circuit on sensor performance

2. Architecture

The architecture of this system comprises of the hardware part and the software part. The hardware part contains various components for the transmitter module which takes inputs from the sensors and sends it to the cloud. There are two major hardware modules in this system. The hardware modules are the Sensor value transmitter and the gateway. The diagrammatic illustration of this setup is shown below.

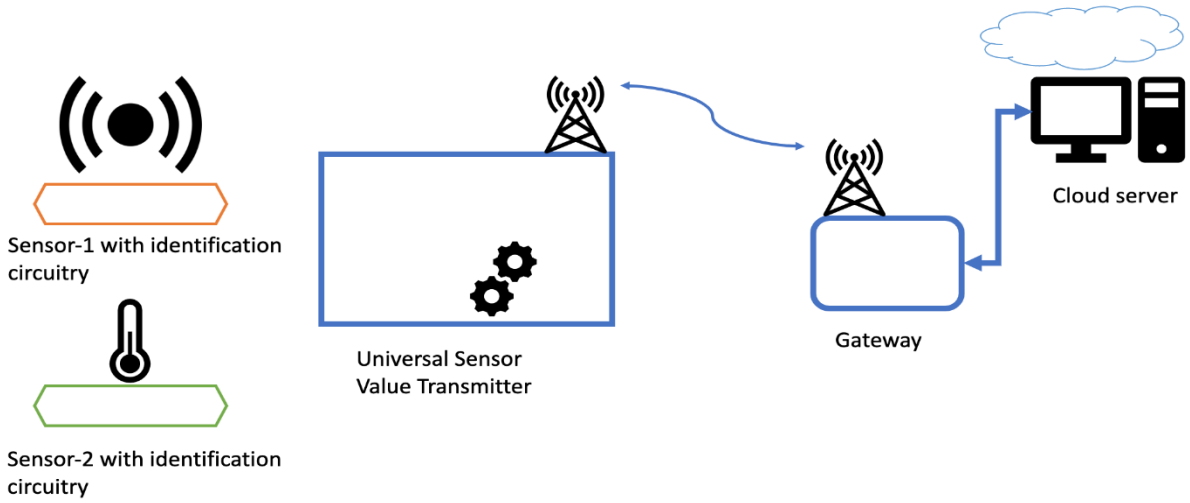

Figure 1: Schematic diagram of the overall setup.

The hardware part will contain the actual transmitter which is a SparkFun SAMD21/transmitter which contains two ports where the sensors can be plugged in. It has an antenna and a battery connection. The sensors are also designed to output an identification voltage using resistors. The gateway part only contains the antenna which received the signal communication from the transmitter and then uploads it to the cloud. Most of the functionalities of this system are controlled by the software. The battery style for this project is using the 3.3 V DC battery. The sensor and how it is designed for detection is shown below.

Sensor type detection using the system:

To transmit the sensor output voltage only to the cloud, the connection between the sensor output voltage and the transmitter (analog pin) is sufficient (not to mention the power supply connections). But to detect the sensor type, another additional connection between the sensor and the transmitter is required which is the connection between the sensor identification voltage and transmitter's another analog pin. So, from the sensor's perspective, the connections will look like something like this:

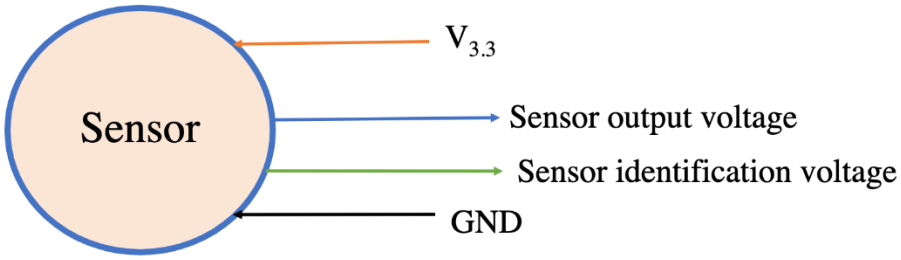

Figure 2: Sensor pins and meaning.

Instead of using a separate (precise) voltage source for the identification voltage, we can utilize the existing power supply (between Vcc and GND) and the Kirchhoff's Voltage Division Law to create specific voltage by selecting the appropriate resistances. The concept of creating sensor identification voltage is illustrated below:

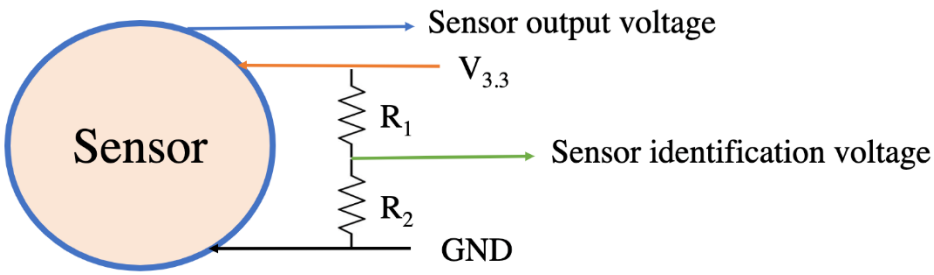

Figure 3: Concept of generating sensor identification voltage.

One the sensor is powered up and running, and the identification voltage has been generated, we need to connect these two voltages to the designated analog input pins of the transmitter. These connections are shown in the following figure:

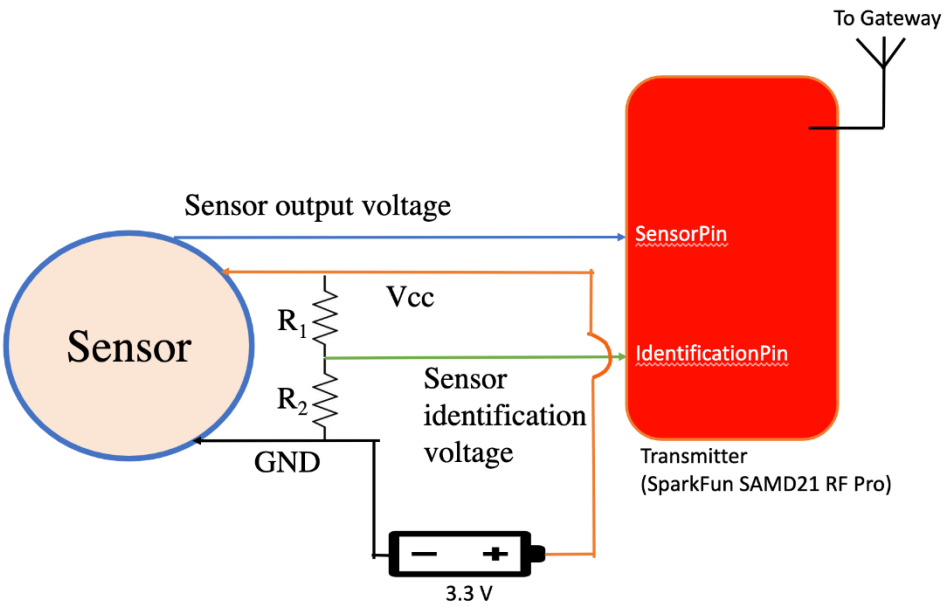

Figure 4: Connection between sensor and transmitter.

For the software part of the system, it will contain the transmission code which gets the sensor ID and Sensor values in real time and transmit it to the cloud through the gateway. To maintain the proper operational functioning of these processes, we build a watch dog timer. We also build an error logging function, reception and sending function for the gateway. One importance component of the software part is making the gateway to be able to differentiate between the right transmitter packet from the wrong packet from another transmitter.

List of objects, functions, or other software elements with interface info

Software connection and interface information

The transmission between the transmitter and the gateway will use LoRa with RFM95W LoRa Radio Chip on Spark fun Pro Rf. Due to the LoRa frequency band for America, the team decided to use 903.9MHz as the transmitting frequency. The users can transmit data in 1-mile line of sight with an Antenna.

To connect the Microsoft Azure Cloud, the website's connection string needs to be provided and added to the program.

Connection string example:

“HostName=Junxiaohub.azuredevices.net;DeviceId=sparkfunlora;SharedAccessKey=Ys/6ctrXKt+5HkX8rjmNcBA NLYfM+MPsuNi2F7EvQYw=”

After that, the cloud can gather data from the gateway and do further analysis.

3. Architecture – Message dictionary

Transmitting Package:

The package sent from the transmitter will be an integer array containing information of Identification code, Node ID, Sensor Type, Packet ID, Timestamp, Sensor value and Error code.

Table 1: Architecture of Transmitting Package

Message Position	Comment
1	An identification code to help gateway to upload correct package to the cloud
2	Timestamp: Document the time passes since the program starts running
3	Node ID: 1, 2, and 3
4	“Temperature” for temperature sensor, and “Soil Moisture” for soil moisture sensor
5	Packet ID: Counting from 1, 2, 3, and so on
6	Sensor value: the reading converted from analog voltage
8	First part of error code
9	Reserved extra space for the package

Error logging:

In our lab experiments, we already have experience of logging the run time error in a 16-bit error code. We used a similar kind of approach but improved it as per the engineering requirement.

The reason we incorporated the watch dog time with error log in is to track the program flow of our system and for easy debugging. Some of the errors to log in are run time errors, communication errors that track lost packets due to wireless channel and faulty sensor readings. Key points to note in the configuration of the Error logging are:

I. Configuring the Device Service unit registers
II. Resetting the error codes each time we want to send and error code
III. Including the flash storage library to store errors at every instant
IV. Assigning the packets for the error log and setting the timestamp for each log.
V. Assigning bits position and using switch statement to move through each bit's position
VI. Setting the packet counter and the millis() timestamp

For the watch dog timer, we implemented it to monitor correct program operation and recover from error situation. Before configuring the WDT, we first disable the WDT and configure the generic clock registers, configure the WDT early interrupt warning and then enable the watch dog time then kick. The watch dog timer works with error logging to ensure the proper program flow or operations. We assigned 16-bit positions for the respective errors.

The 16-bit error code’s description is provided below:

Table 2: 16 Bits of Error Code

Bit position	Corresponding error code
0	power on reset
1	WDT reset
2	External Reset
3	Protection Error
4	DSU Failure
5	Bus Error
6	CPU Reset Phase
7	Hot Plugging Enable
8	Debug Communication 1
9	Debug Communication 0
10	Debugger Present
11	missing packet
12	reception failure
13	transmit failure
14	Value above 100 or below 0
15	reserved

4. Design

What does each component do?

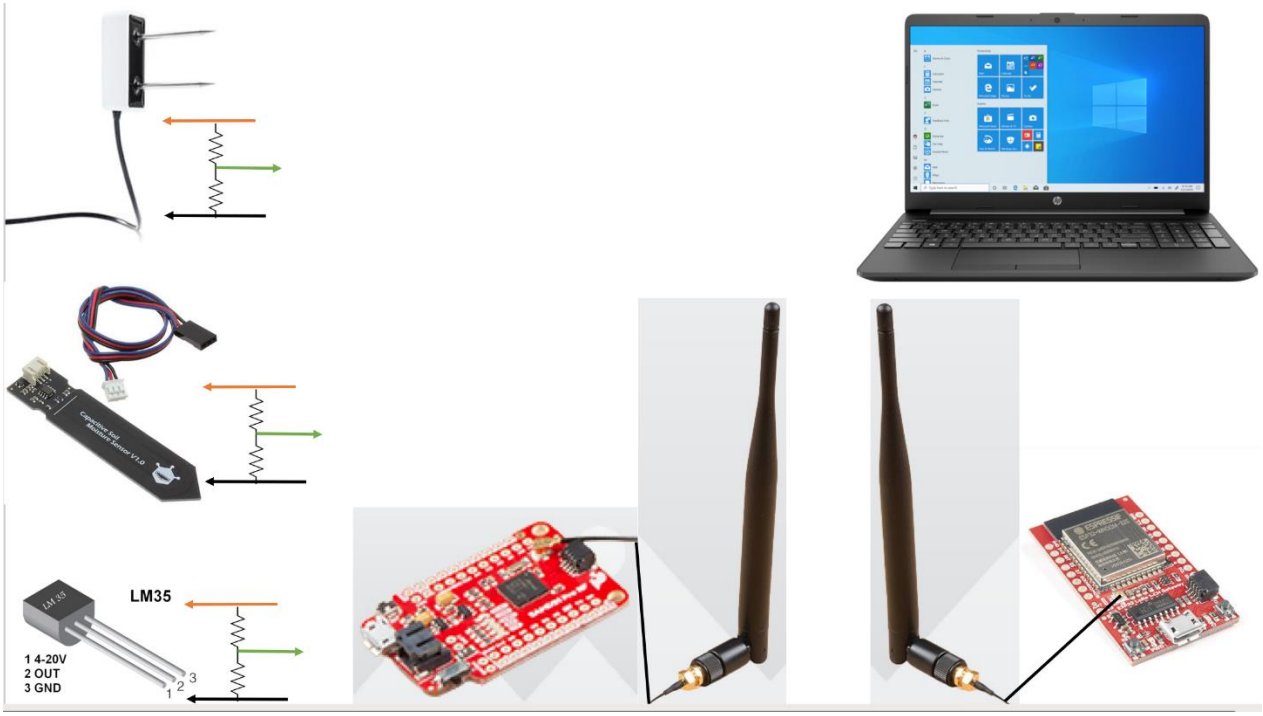

Figure 5: Overall setup of the project (dummy/draft version).

In our proposed system, each sensor will come with its own identification voltage (eventually with own R_1 and R_2).

To illustrate the scenario better, let us consider the following example. For both sensor-1 (e.g., TMP36) and sensor-2 (e.g., SKU: SEN0193) if we use 3.3V DC as Vcc, and for sensor-1, let us say $sensor_1.R_1$ = 517 Ω and $sensor_1.R_2$ = 1200 Ω, the sensor identification voltage for sensor-1 will be 2.3 V. If, $sensor_2.R_1$ = 1200 Ω and $sensor_2.R_2$ = 517 Ω, then the sensor identification voltage for sensor-2 will be 0.99 V.

Now, if the transmitter detects any voltage around 2.3 V in its identification pin, it will detect the sensor as sensor 1. Similar cases will happen for sensor-2 (identification voltage close to 0.99 V).

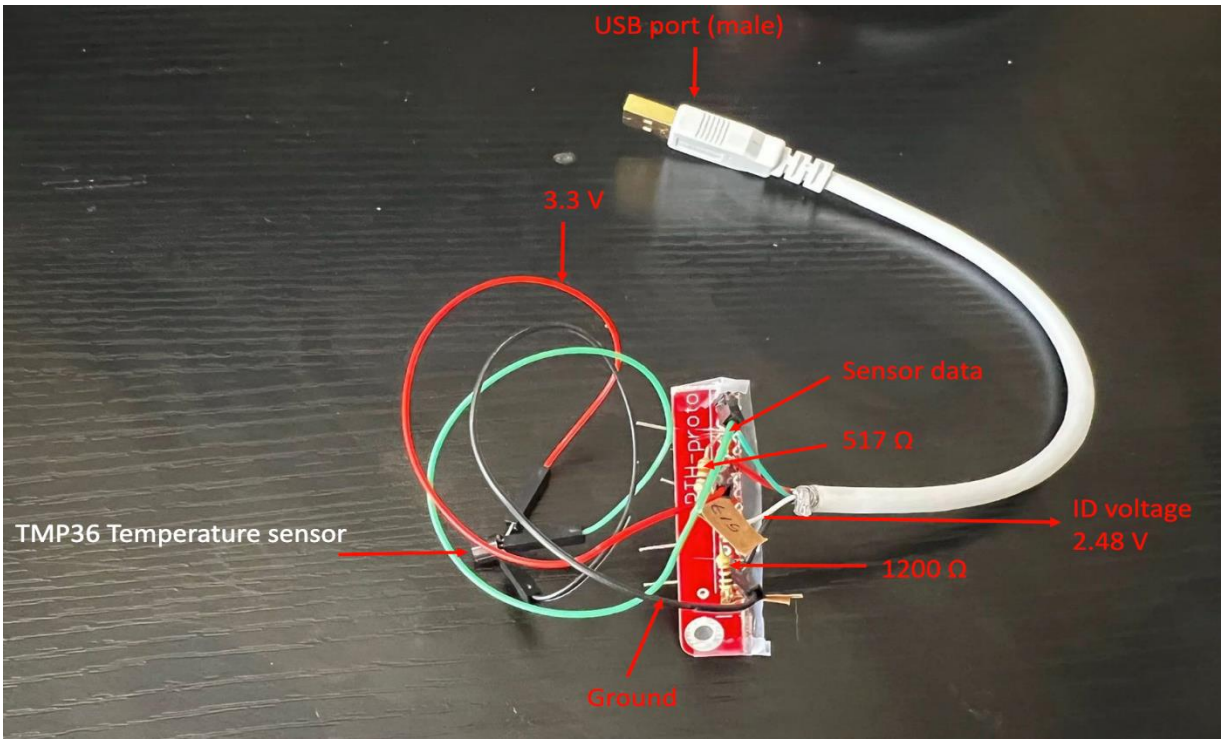

Figure 6: TMP36 temperature sensor with identification circuit.

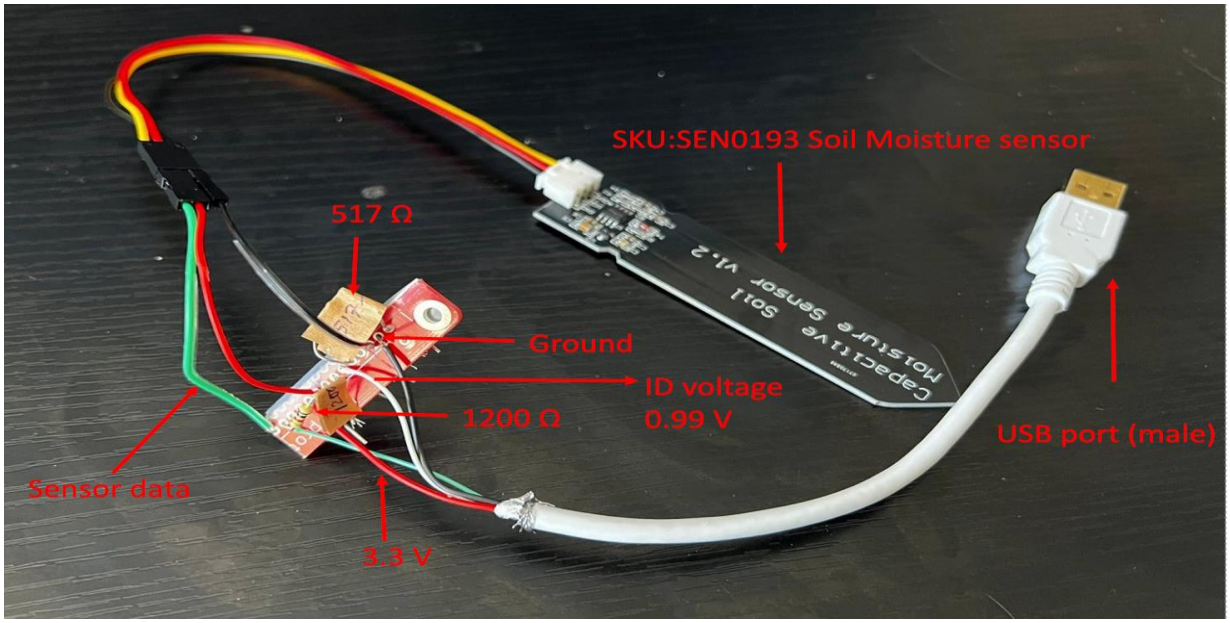

Figure 7: SKU: SEN0193 soil moisture sensor with identification circuit.

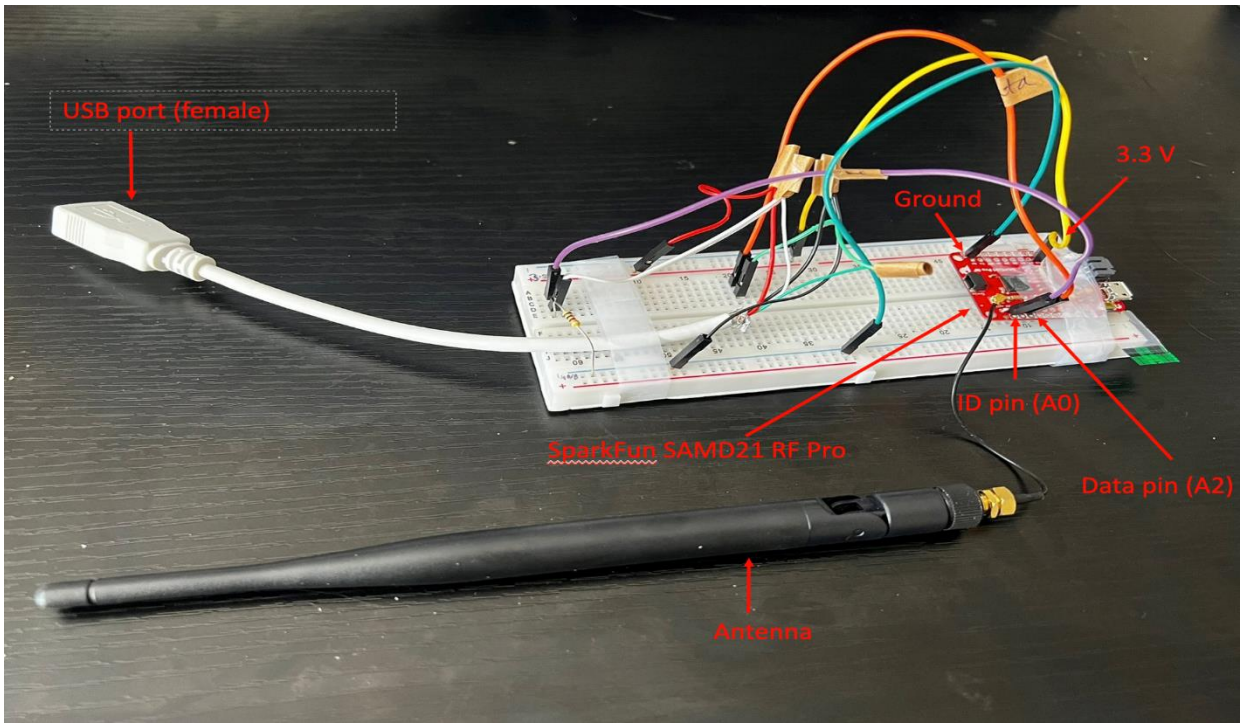

Figure 8: Plug and play transmitter circuit.

Temperature sensor TMP36 has the following temperature voltage conversion with a 10 mV/^{o}C scale factor.

Table 1: Temperature to output voltage conversion TMP36

$Temp_{in}$	**V_{out}**
150°C (max)	2V (max)
25°C	0.75 V
–40°C (min)	0.1 V (min)

The conversion from voltage to temperature can be formulated as follows:

$$Temp\ (^{o}C) = [\{V(mV) - 100\}/10] - 40$$

Soil Moisture (Gravimetric Water content) and voltage relationship for low-cost SM sensor, SKU: SEN0193 sensor is as follows:

$$SM = [55.7 - SQRT\{55.7^2 - (3000\text{-}mV)\}]/(2*1.3)$$

Effect of identification circuit on sensor performance

The system voltage we have used is 3.3 V. When we have used the identification circuit, system voltage dropped to 3.299 V. This voltage is close to 3.3 V and good enough to power up the sensors we have used. For that reason, we did not notice any significant changes in the performance of the sensors.

A current analysis of the circuit must have been done shown as follows. First, it should be measured that how much current the sensor is drawing.

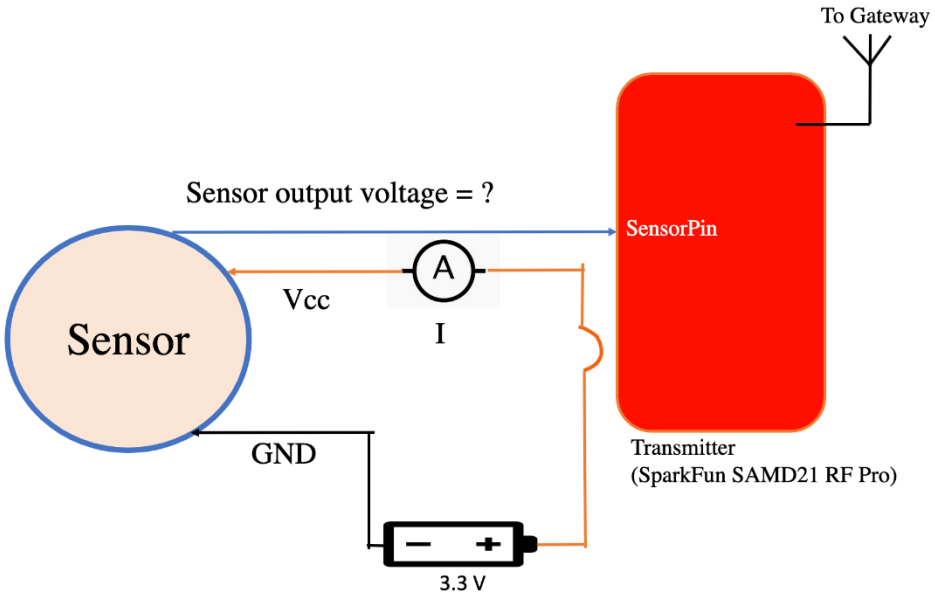

Figure 9: Measuring the current drawn by the sensor.

After that, when the ID circuit is added in the system, it can be measured that how much current the ID circuit is drawing and how much current the sensor is drawing. From there we can have an idea of effect of ID circuit in sensor performance. However, due to time constraint, it was not possible to finish the current analysis.

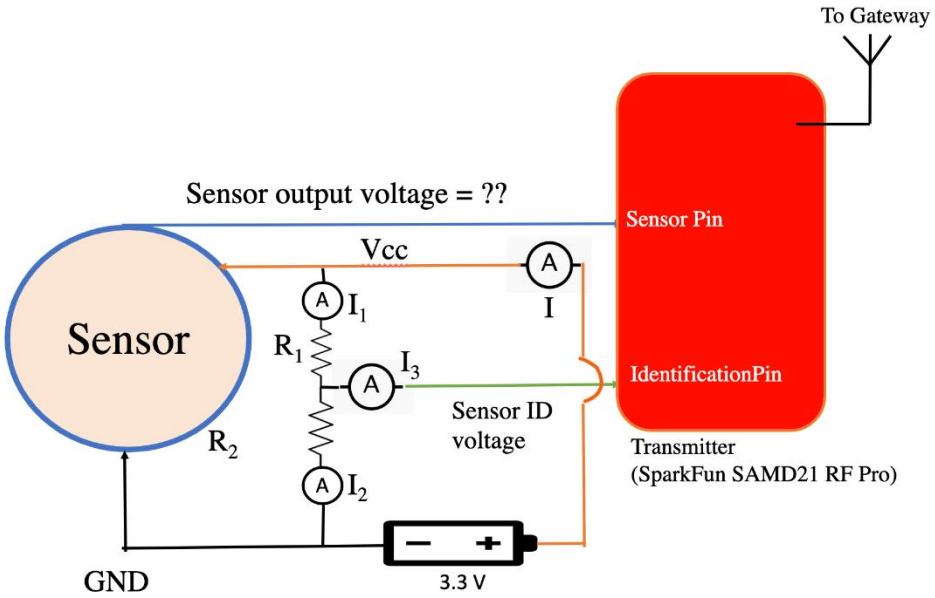

Figure 10: Measuring the current drawn by the ID circuit and the sensor once the ID circuit has been introduced.

The following 2 figures show the sensor output voltages with and without ID circuit.

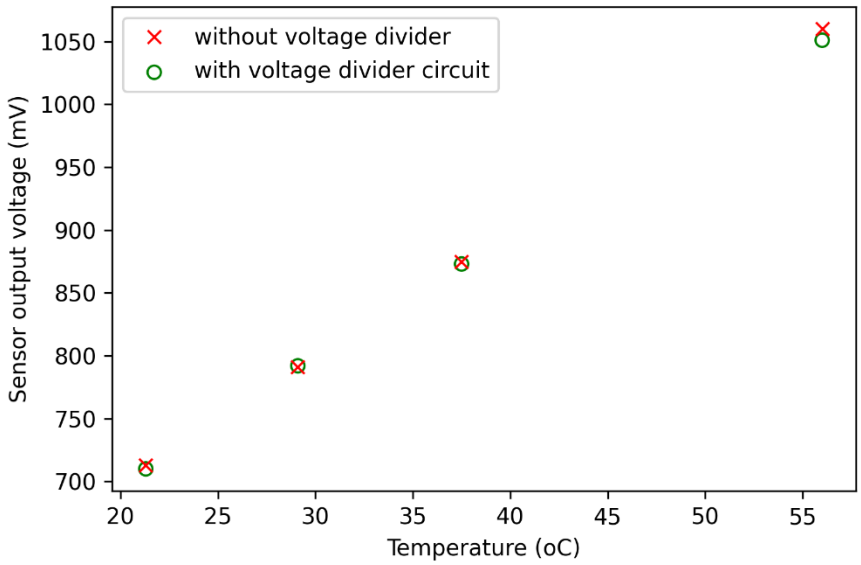

Figure 11: Temperature sensor TMP36's output voltage without and with the identification circuit.

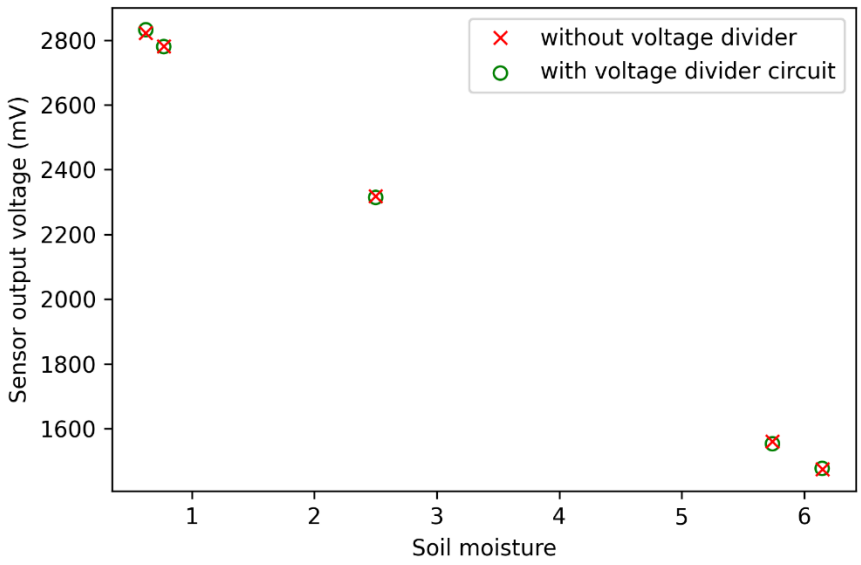

Figure 12: Soil moisture sensor SKU:SEN0193's output voltage without and with the identification circuit.

The transmitter will get the sensor type when its power on, and get the analog input every 4 seconds. The analog input will be converted to temperature or soil moisture and generate in the a packet with the format (Identification ID, Timestamp, Node ID, Sensor type, Packet ID, Sensor reading and error code) and send to gateway.

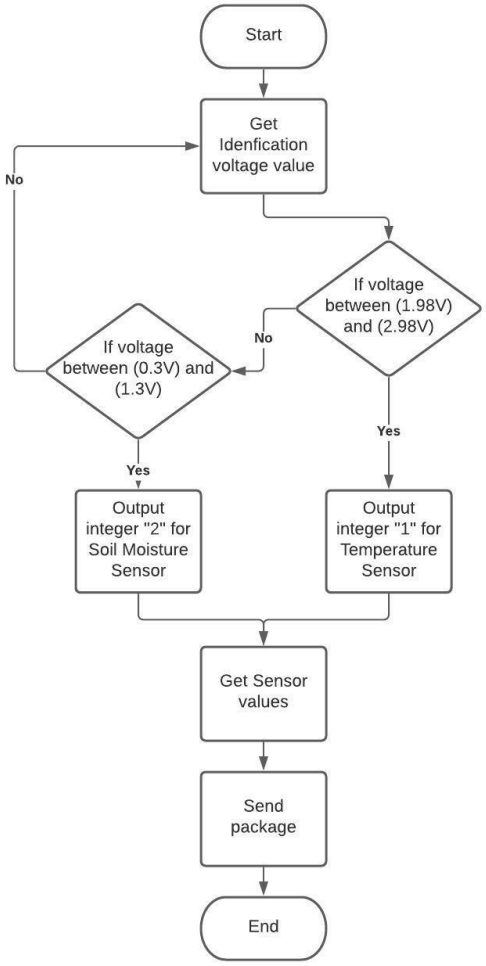

Figure 13: Transmitter Flowchart

Pseudocode:

```
Vidx=2.3 // set an identification voltage for sensor-1
Vidy = 0.99 // identification voltage for sensor-2
Th=0.5 // setting up a +/- threshold up to which we are willing to accept/recognize sensor
id_value = analogRead(IdentificationPin);
if (id_value<=(Vidx+Th) && id_value>=(Vidx-Th))
```

```
        {SensorType = Sensorx;}
else if (id_value<=(Vidx+Th) && id_value>=(Vidx-Th))
        {SensorType = Sensory;}
else SensorType = Unidentified;
tx("SensorType"); // transmit what type of sensor it is if detected or unidentified otherwise

while (1){
        sensor_value = analogRead(SensorPin);
        tx("sensor_value"); // transmit the output voltage of the sensor continuously
        delay(time); // within a certain interval
        }
```

After Gateway gets the package from transmitter, it will check the identification code in the packet and then decide whether send to Cloud or not.

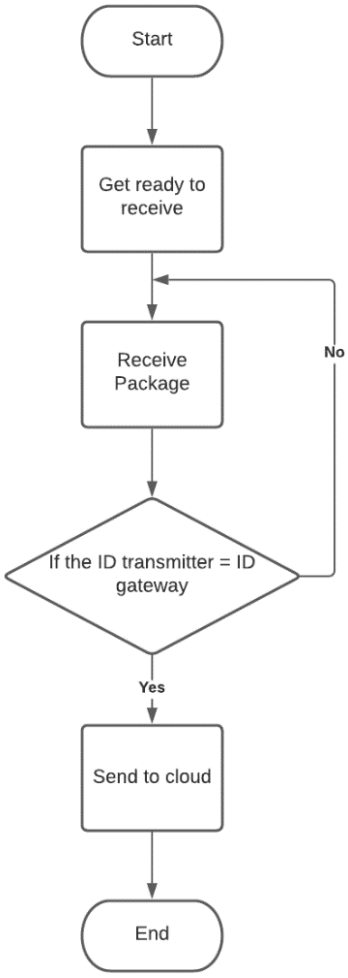

Figure 14: Gateway Flowchart

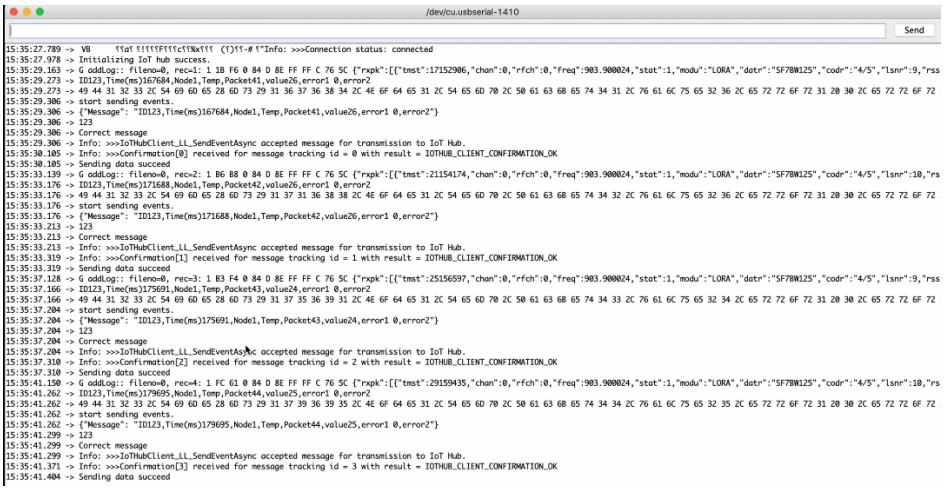

Figure 15: Serial Monitor from Gateway Side (Correct ID)

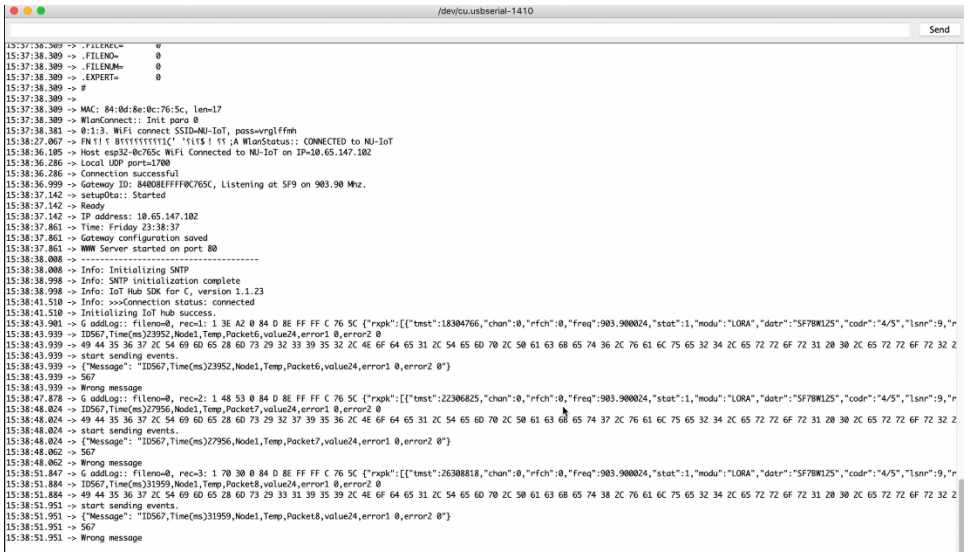

Figure 16: Serial Monitor from Gateway Side (Wrong ID)

Pseudocode:

```
if (res.substring(ID position) == Identification Code)
 {
 Serial.println("Correct message");
 if (Esp32MQTTClient_SendEvent(buff))
  {
   Serial.println("Sending data succeed");
  }
 else
```

```
   {
    Serial.println("Failure...");
   }
  }
else
{
 Serial.println("Wrong message");
}
```

The screenshot of Azure cloud shown following is json file gathered on the platform and available for users to download and convert into CSV file for future analysis.

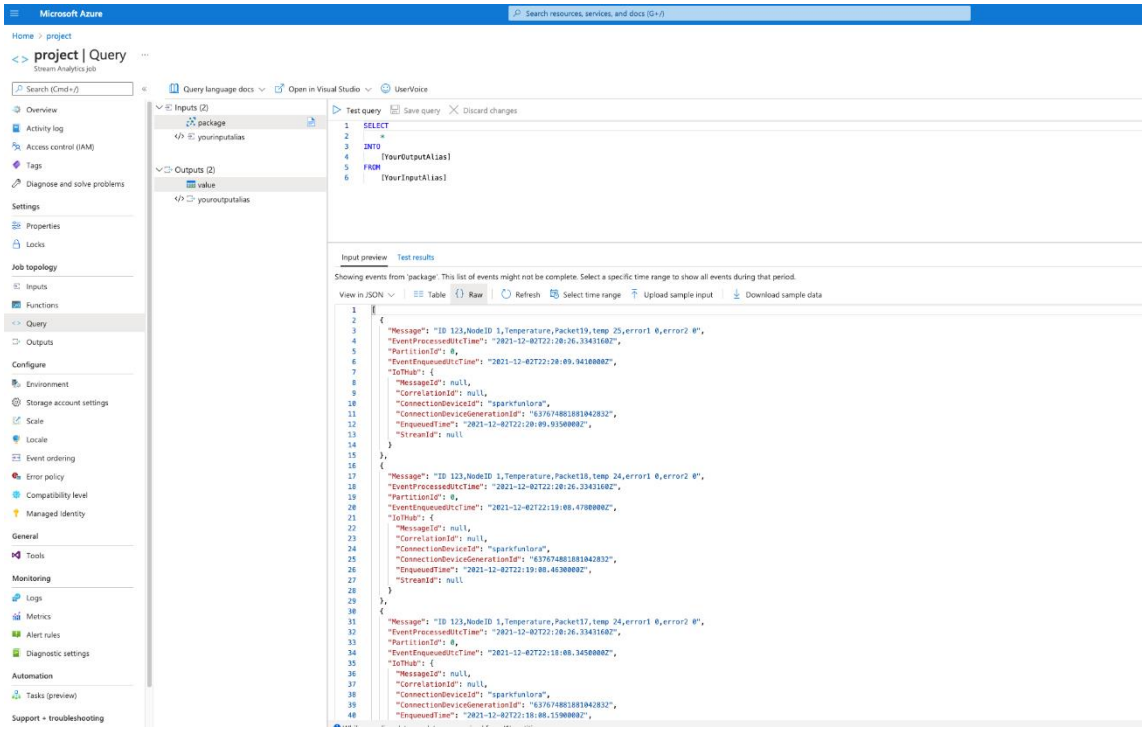

Figure 17: Received packet shown as json file in Microsoft Azure cloud

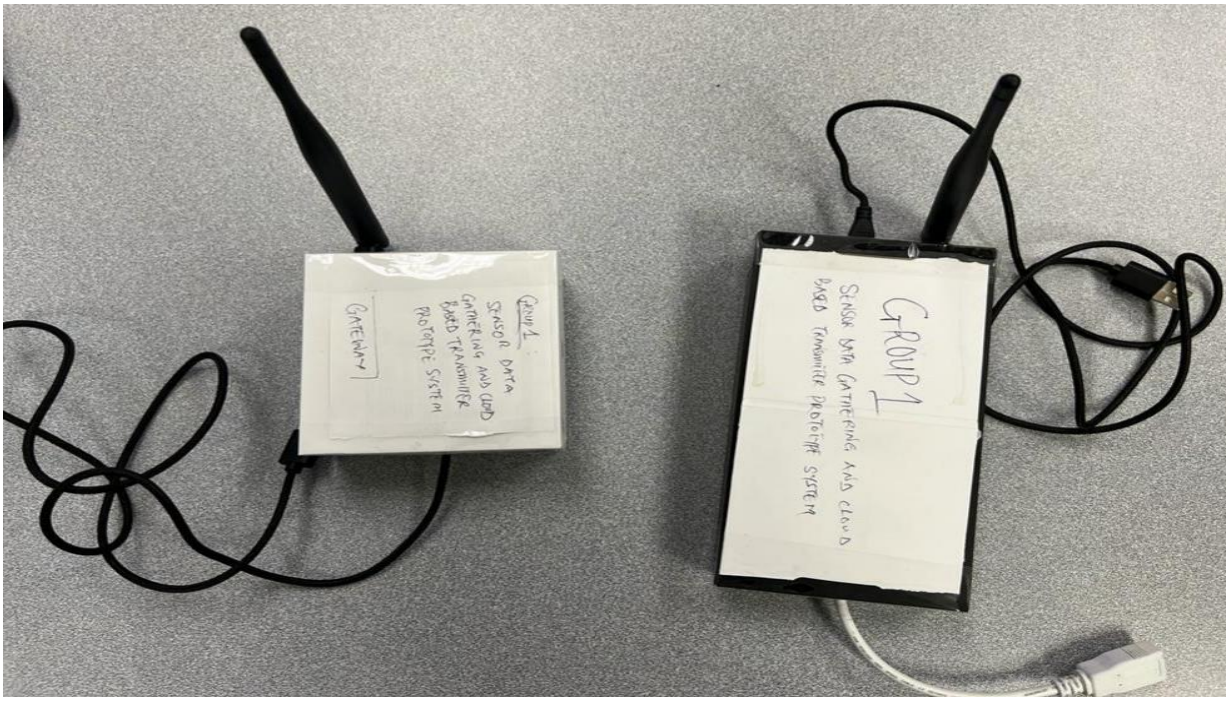

Fig 19: Final packaged Transmitter-Gateway Prototype

5. Bug List

Bug	Priority	Status	Comment
Sending temperature or soil moisture is not working with “double”	fixed	medium	Used “int” to send integer type data right now and will try some other way next time.
Keep receiving other messages from unknown transmitter	deferred	low	An identification code on gate way has been added to identify the message from authentic transmitter
There is a bit limit for package sending from gateway to cloud	fixed	medium	Decrease the length of the packet and add some empty position

6. Software Development Plan

General approach

A Sensor Data Gathering, and Cloud based Transmitter Prototype System

Ebuka Philip Oguchi

Junxiao Zhang

Md Didarul Islam

Project Start: Mon, 11/22/2021

Display Week: 1

TASK	ASSIGNED TO	START	END
Sensor Identification and Reading			
Sensor connection	Md Didraul Islam	11/22/21	11/23/21
Data Reading	Md Didraul Islam	11/23/21	11/25/21
Sensor Identify	Md Didraul Islam	11/23/21	11/25/21
Observation of the effect of identification circuit on sensor performance	Md Didraul Islam	12/3/21	12/6/21
Test	Md Didraul Islam	11/25/21	11/26/21
Tranmisitting system			
Transmitter Code	Junxiao Zhang	11/27/21	12/1/21
Gateway Code	Junxiao Zhang	11/27/21	12/1/21
Cloud Setup	Junxiao Zhang	12/1/21	12/3/21
Package Identificatio	Junxiao Zhang	12/3/21	12/6/21
Test	Junxiao Zhang	12/1/21	12/3/21
Package Identification Test	Junxiao Zhang	12/7/21	12/10/21
Safety system			
WDT	Ebuka Philip Oguchi	11/27/21	12/1/21
Error Logging	Ebuka Philip Oguchi	12/1/21	12/3/21
Test	Ebuka Philip Oguchi	12/1/21	12/3/21

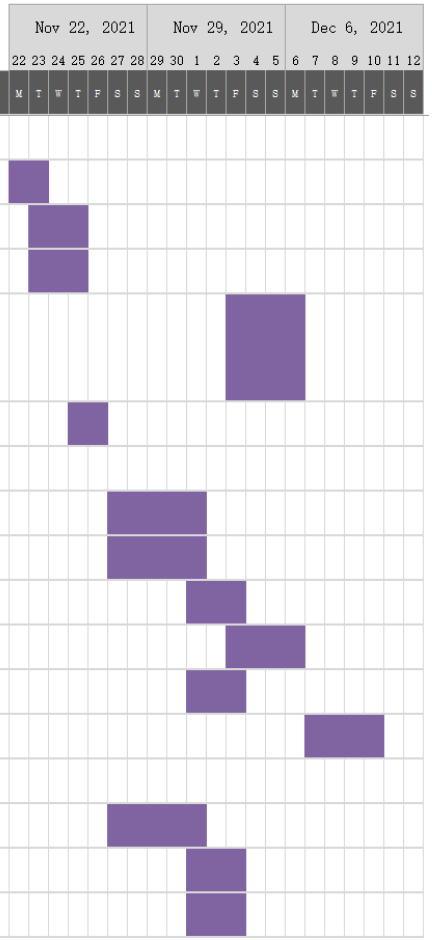

Figure 18: Gantt Chart

Names of documents to be produced

1. Customer requirements
    a. Functions and services that are expected for the design
    b. Constraints on the system
    c. Success criteria
2. User Guide
    a. Introduction to users about the system and how to use it
    b. List of requirements maintenance procedures

    c. Troubleshooting and problem resolution guide
3. Engineering requirements
    a. Functions of hardware and software
    b. Performance criteria
4. Implementation
    a. Include code (header, code itself, and code comments)
    b. A style sheet to define how and when to comment
5. Test plan
    a. List of tests (spreadsheet)
    b. List of test results
    c. Records of peer review results
    d. Test every requirement (engineering & customer)
6. Schedule and staffing plan
    a. Staff assignments
    b. Schedule and milestones
7. Architecture
    a. Describe the components and objects in the software and hardware
8. Message dictionary
    a. Contains all message formats and meanings transmitted through the network.
9. Design
    a. Describe the relationship between each component by flowcharts, state charts, and pseudocode
10. Bug list
    a. List of bugs and other issues
    b. Status and priority of each bug
11. Software Development Plan
    a. General approach
    b. Documents and the requirements

7. Experimental Challenges encountered and practical solutions

Experimental Challenges	Possible Solutions
1. Message Interference and collision	We Introduced a mechanism for the gateway to differential between the expected Packets from the transmitter and the wrong transmitter packets
2. Physical Packaging	i. Getting the correct form factor for the packaging. ii. It is still not waterproof yet iii. The packaging looks modular and easy to open.
3. Time required for the data to get to cloud	Messages sometimes get queued up. We tried reducing the time gap between 2 transmissions. Shows Improvement.
4. Transmitting float temperature value	We could only transmit temperature as an Integer.

8. Conclusion

In this project, we studied and implemented a Sensor Data Gathering, and Cloud based Transmitter Prototype System which is a plug and play device to data collection, transmission and logging errors encountered. We had a hands-on experience of the implementation of this system from the project design to the actual implementation. Our system is built to protect against interferences from other unknown transmitters. This prototype system is modular, easy to use and scalable.

9. Future Project Directions

FP-1: **Universality:** For this project, we have tested our prototype system with only two types of sensors: temperature sensor and soil moisture sensor. In the future, we want to test our system to be compatible with any sensor devices.

FP-2: **Scalability:** For this project, we limit our sensors to only one temperature sensor and one soil moisture. In the future, we want our Prototype System to be able to incorporate more sensor devices.

FP-3: **Data storing:** Storing the sensor values in an SD card connected to the transmitter, so that if for some reason, the communication between the transmitter and the gateway is interrupted, the values collected during this time are not lost.

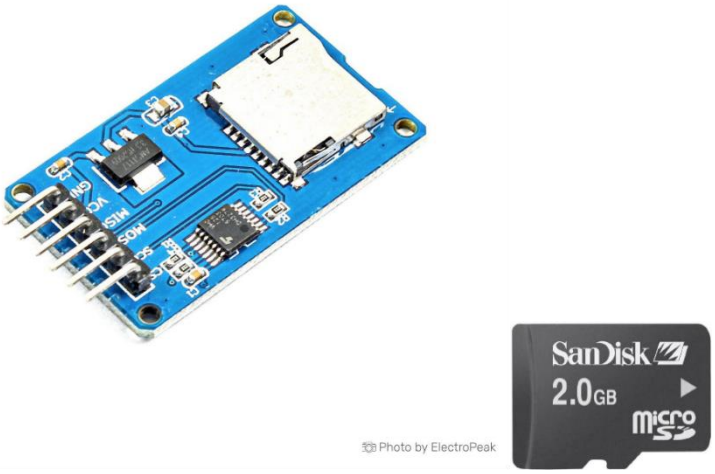

Figure 20: Micro SD card adapter with micro-SD card.

FP-4: **Real time clock:** In remote areas, where communication channels are not robust enough, transmitted messages are often delayed or discarded sometimes. At the receiving end, Microsoft Azure uses its own clock to timestamp the message when it was received. But this is not referring to the time when the message was generated. To get an accurate insight of the data generation time, we can use RTC at the sensor side to timestamp the data when it was collected.

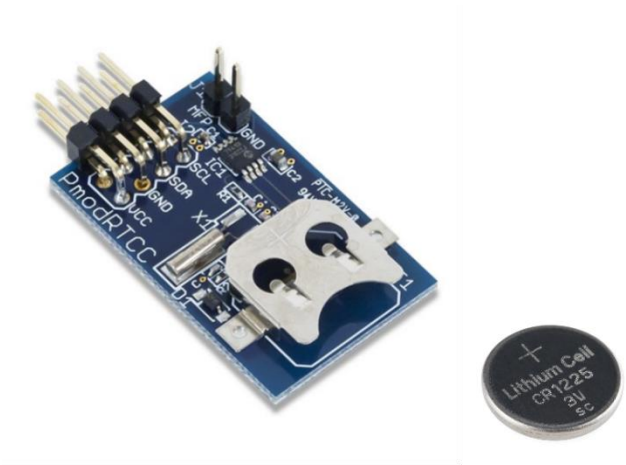

Figure 21: Real-Time Clock with Lithium coin cell.

FP-5: **Power saving by periodically awaking sensor:** We can use a common or separate power source for the transmitter and the sensor(s). No matter how frequently we are collecting (or

accepting) data from the sensor, the sensor is continuously transmitting its output. That will drain the sensor battery rapidly, which is not suitable for field applications since it is not convenient to recharge/change batteries by going to the remote fields very often. For that case, if we have any sensor of which we do not need any continuous output, a periodical output (let us say every 15 minutes or every hour) will also serve the purpose, we can turn on that sensor only at that specific time using MOSFET. As the following figure illustrates, Spark Fun's digital output pin is giving a (preset/ expected) periodic signal to the gate of the MOSFET. That will turn on the MOSFET, and only at that time will the sensor get power supply from the battery. That simple trick will significantly extend the sensor's battery life.

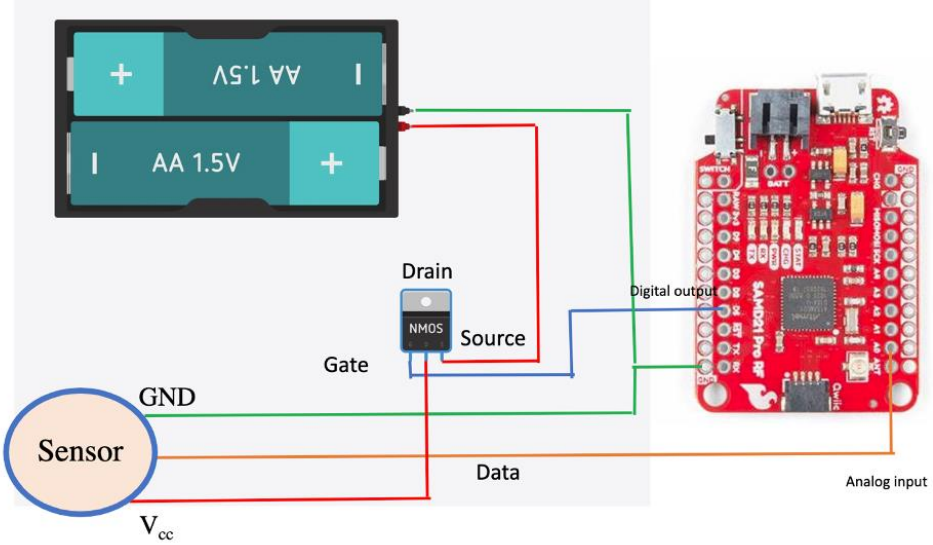

Figure 22: (Sensor) Power saving mechanism using MOSFET.

FP-6: **Power saving by detaching sensor identification voltage:** Once the sensor has been identified by the system, there is no need to continue this voltage. So, after a certain specific time (let's say 15-20 seconds) of the sensor identification, the identification voltage can be discontinued. This action can be performed in a similar fashion as described in the previous section. A high-level idea is sketched below:

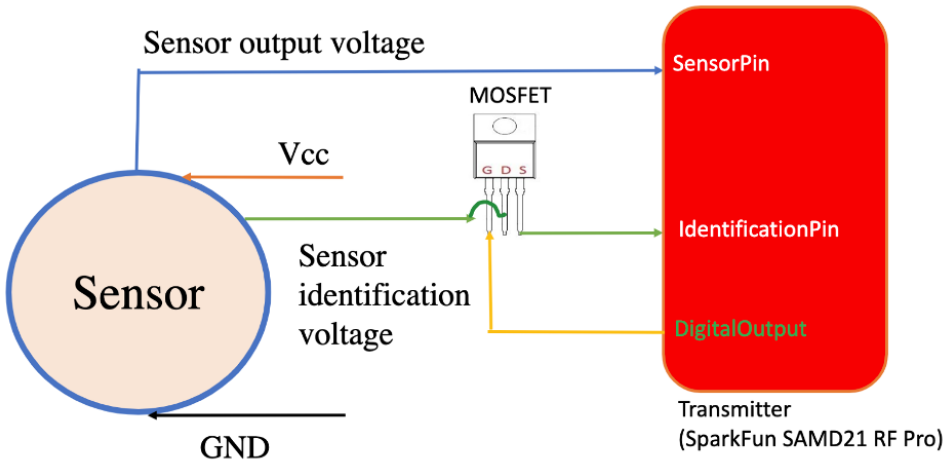

Figure 23: Disconnecting the sensor identification voltage using MOSFET.

FP-7: IP v 67: Water, splash, and dust Resistance for the packaging. The current packaging does not have any IP rating.

FP-8: Incorporating a system to do data visualization and analysis on the cloud

FP-9: Converting Json to excel file from the Microsoft Azure cloud and saving it

References:

[1] https://www.walmart.com/ip/Energizer-Rechargeable-AA-Batteries-4-Pack-Double-A-Batteries/887736?wl13=1943&selectedSellerId=0

[2] https://en.wikipedia.org/wiki/LoRa

[3] https://www.sparkfun.com/products/15836

[4] https://www.howtogeek.com/200728/why-does-it-take-so-long-to-upload-data-to-the-cloud/