Design and Implementation of a Modular Home Monitoring System Using Arduino and Home Assistant

Prepared for: Professor Christopher Hockley

Prepared by: Joel Adkins, Daniel Betts, Ethan Billette, John Collins, Evan Miceli

Date of Submission: August 11, 2025

Abstract

This report covers the design and construction of a home monitoring system for our ME 313 project. Our goal was to use an Arduino Mega to collect data from various sensors and display it on a dashboard. The system uses a Raspberry Pi running Home Assistant to show real-time temperature, humidity, light, motion, air quality, and RFID scans. The project was divided between the group, with each member responsible for integrating a specific component. We successfully built and tested the system, with the main challenge being an undocumented voltage issue with the RFID sensor that required troubleshooting to solve.

Table of Contents

1. Introduction
    1.1. Project Motivation and Goals
    1.2. Scope and Objectives
    1.3. Report Organization
2. Literature Review
    2.1. Overview of Smart Home Technology
    2.2. Microcontroller Platforms for IoT
    2.3. Principles of Key Sensors
3. Design Description and Analysis
    3.1. System Architecture
    3.2. Hardware Description
    3.3. Software Design
    3.4. Design Analysis
4. Implementation and Testing
    4.1. Hardware Assembly
    4.2. Testing Procedures
5. Results and Discussion
    5.1. System Performance and Results
    5.2. Challenges, Limitations, and Future Work
6. Conclusion
7. References
8. Appendix
    - Appendix A: Full Arduino Source Code
    - Appendix B: Home Assistant Configuration (YAML)

Table of Figures

- **Figure 1:** Overall project setup showing the integrated system in operation.
- **Figure 2:** System architecture block diagram.
- **Figure 3:** The fully assembled sensor circuit on a breadboard, connected to the Arduino Mega.
- **Figure 4:** Close-up of the circuit showing the SGP40, DHT11, and photoresistor.
- **Figure 5:** The Raspberry Pi 4 serving as the Home Assistant host.
- **Figure 6:** Screenshot of the Arduino IDE Serial Monitor displaying the JSON data string.
- **Figure 7:** The final Home Assistant dashboard displaying real-time sensor data.

1. Introduction

1.1 Project Motivation and Goals

The goal of this project was to build a functional smart home monitoring system using an Arduino Mega and a variety of sensors. The system needed to measure environmental data, detect events like motion, and display the information in a user-friendly way. This project put our instrumentation skills to the test by requiring us to integrate multiple hardware components and make them communicate with a software interface.

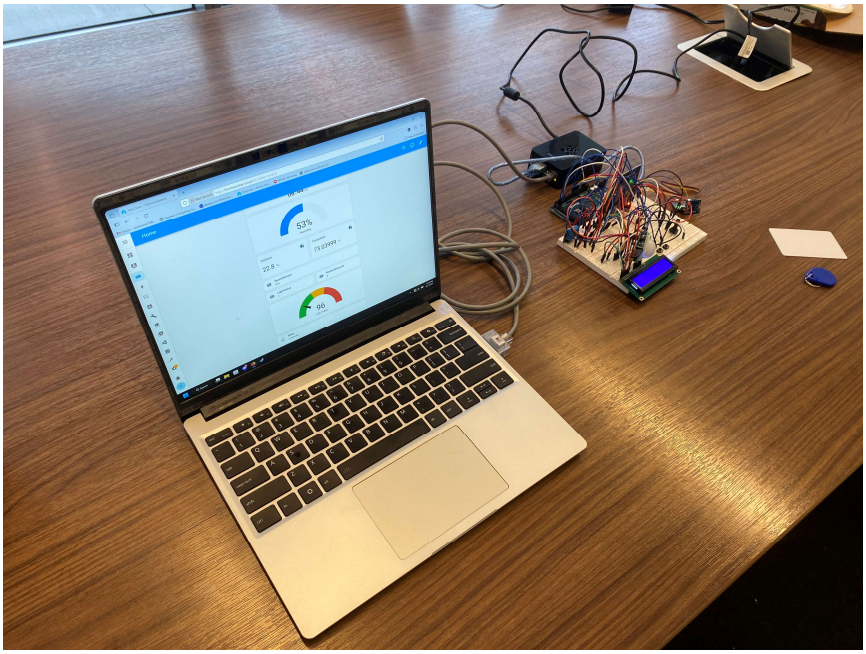

Figure 1 Overall project setup showing the integrated system in operation.

1.2 Scope and Objectives

The project was broken down into modular tasks, with each team member taking responsibility for the integration of specific sensors and subsystems, as outlined below:

- **Ethan Billette:** Home Assistant integration, DHT11 temperature/humidity sensor, LED status light.
- **Evan Miceli:** HC-SR501 motion sensor and buzzer integration.
- **Daniel Betts:** SGP40 air quality sensor integration.
- **Joel Adkins:** RC522 RFID module and Grove Sound Sensor integration.
- **John Collins:** 16x2 LCD display and photoresistor (light sensor) integration.

The system is designed to be both functional and extensible, providing a solid foundation for future development in home automation. This report documents the entire engineering process, from the initial design and component selection to final testing and results.

1.3 Report Organization

This report begins with a review of the underlying technologies, followed by a detailed description of the system's hardware and software design. The implementation and testing phases are then documented, leading into a discussion of the results and challenges encountered. The report concludes with a summary of the project's achievements and suggestions for future work.

2. Literature Review

The core of the system is the Arduino Mega 2560, a microcontroller board based on the ATmega2560 processor. It is well-suited for this application due to its large number of digital and analog input/output (I/O) pins (54 digital, 16 analog), multiple hardware serial ports, and extensive community support [1].

The user interface and data aggregation are handled by Home Assistant, an open-source automation platform designed to be a central control system for smart home devices [2]. Its ability to integrate with a vast array of devices and services makes it an ideal choice. For this project, its serial sensor platform is critical, allowing it to listen for and process data sent over a USB connection from the Arduino.

Each sensor operates on a distinct physical principle. The DHT11 uses a capacitive humidity sensor and a thermistor. The photoresistor, or Light Dependent Resistor (LDR), is

a variable resistor whose resistance decreases with increasing incident light intensity. The SGP40 is a metal-oxide (MOx) sensor that measures volatile organic compounds (VOCs) to provide a general air quality index [3]. The RC522 module operates on the principle of near-field communication (NFC) at 13.56 MHz to read RFID tags [4]. Finally, the HC-SR501 is a passive infrared (PIR) sensor that detects motion by measuring changes in infrared radiation levels.

3. Design Description and Analysis

3.1 System Architecture

The system has two main parts: an Arduino Mega for data collection and a Raspberry Pi for the user interface. The Arduino is connected to all the sensors and continuously reads their values. The Raspberry Pi runs Home Assistant, an open-source smart home software.

The Arduino packages all sensor data into a JSON text string and sends it to the Raspberry Pi over a USB cable. Home Assistant receives this string, unpacks the data, and updates the dashboard. This setup lets the Arduino handle the real-time hardware tasks while the more powerful Raspberry Pi manages the software interface.

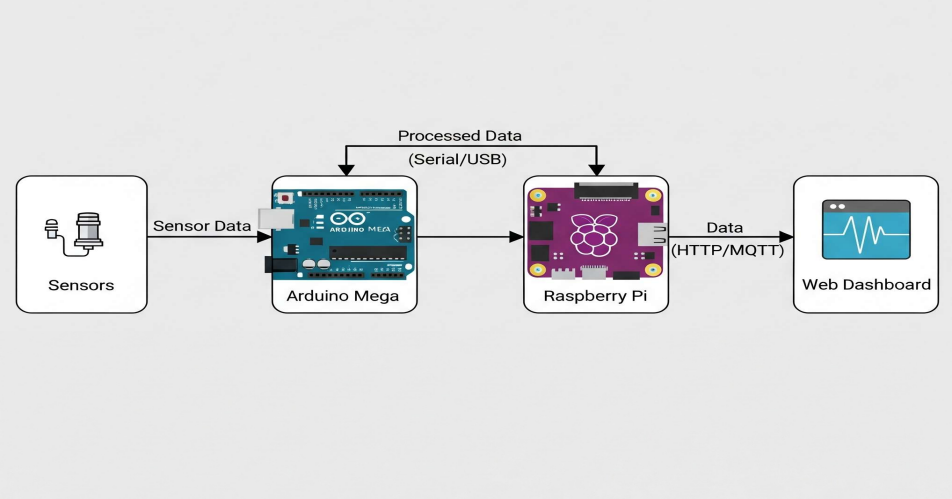

Figure 2 System architecture block diagram illustrating the flow of data from sensors to the user dashboard.

3.2 Hardware Description

The hardware consists of the central microcontroller and a suite of sensors and output devices wired on a solderless breadboard.

- **Arduino Mega 2560:** The central processing unit for data acquisition.
- **Raspberry Pi:** A single-board computer dedicated to running the Home Assistant operating system and server.
- **Sensor Suite:** DHT11 (Temperature/Humidity), Photoresistor (Light), SGP40 (VOC), HC-SR501 (Motion), Grove Sound Sensor, and MFRC522 (RFID).
- **Output Devices:** 16x2 LCD Display and a Piezo Buzzer for audible alerts.

3.3 Software Design

The system's software is split between the Arduino firmware and the Home Assistant configuration.

- **Arduino Firmware:** Written in C++ using the Arduino IDE, the firmware is responsible for initializing all sensor libraries, reading data from each sensor in the main loop, formatting the data into a single JSON string, and printing it to the serial port at a baud rate of 9600. The full code is available in Appendix A.
- **Home Assistant Configuration:** Configured using YAML files, Home Assistant uses a serial sensor to read the raw JSON string from the Arduino. A series of template sensors then parse this string to create individual, usable entities for each metric (e.g., "Humidity," "Celsius"). This configuration is detailed in Appendix B.

3.4 Design Analysis

We made several key design choices. We used a JSON string over a serial connection to send data because it is simple and reliable for a wired prototype. While MQTT is common in IoT, it would have required extra hardware for the Arduino. The Arduino Mega was a good choice because it has dedicated hardware for different communication types like I^2C and SPI, which let us connect all our sensors without issues. Finally, we decided to have the Arduino only collect data and let the more powerful Raspberry Pi handle the processing and user interface, which is an efficient way to design an IoT system.

4. Implementation and Testing

4.1 Hardware Assembly

All components were assembled on solderless breadboards. This method was chosen for its flexibility, allowing for rapid prototyping and easy modification of the circuit during development. The Arduino provided 5V and 3.3V power rails for the various sensors and

modules. While the initial plan included transferring the final circuit to a more permanent protoboard for stability, this step was omitted due to time constraints. The final breadboard implementation, while fully functional, remains susceptible to loose connections.

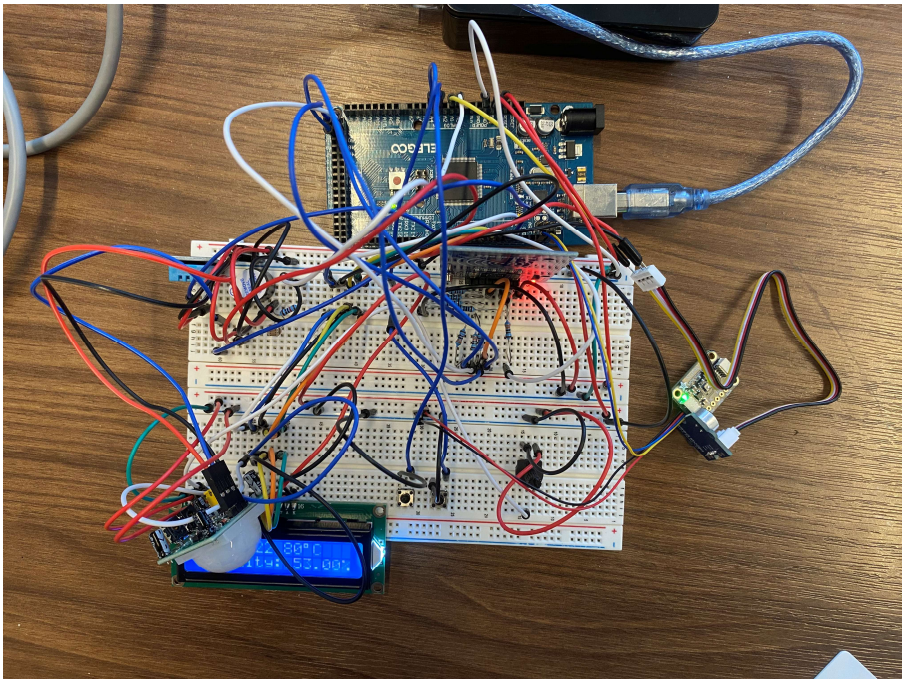

Figure 3 The fully assembled sensor circuit on a breadboard, connected to the Arduino Mega.

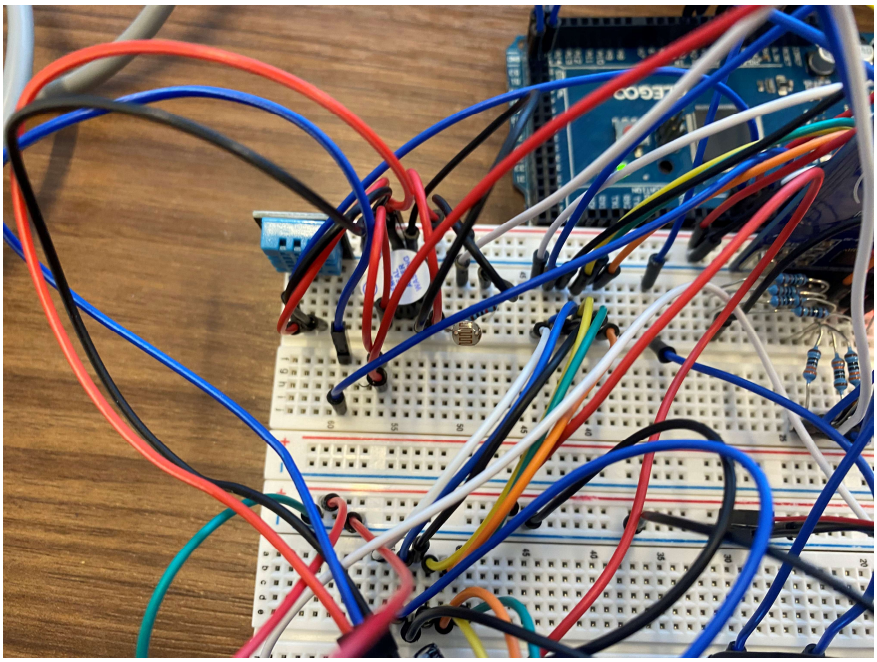

Figure 4 Close-up of the circuit showing the SGP40 (black chip), DHT11 (blue), and photoresistor.

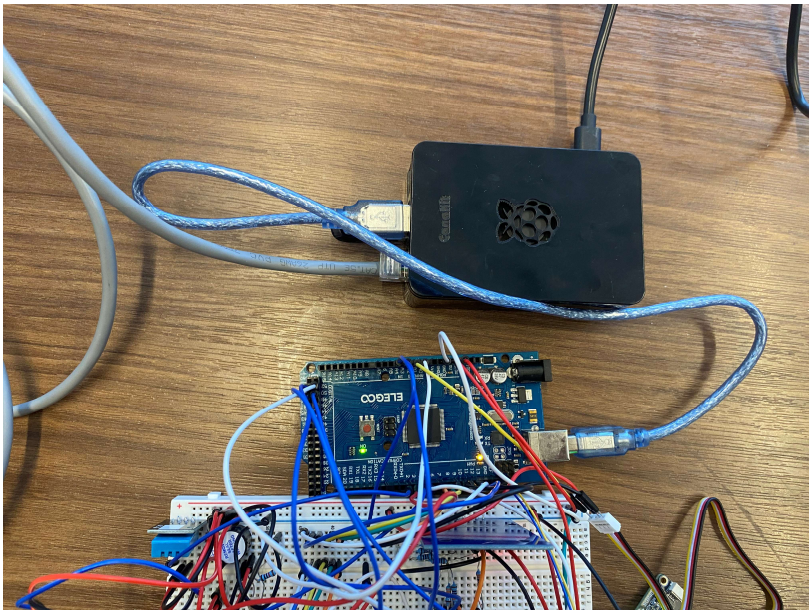

Figure 5 The Raspberry Pi 4 (black case) serving as the Home Assistant host, connected to the Arduino Mega.

4.2 Testing Procedures

Testing was conducted in two phases: individual component testing and full-system integration testing.

1. **Component Testing:** Each sensor was wired and tested individually using simple Arduino sketches. The Arduino IDE's Serial Monitor was the primary debugging tool to verify that sensors were outputting sane and responsive values. This modular testing ensured each component was functional before integration.
2. **System Integration Testing:** Once all components were on the breadboard, the full system was tested. The Arduino was connected to the Raspberry Pi, and the Home Assistant logs and dashboard were monitored to ensure data was being received and parsed correctly. The screenshot in Figure 6 confirms the correct data format and transmission.

Figure 6 Screenshot of the Arduino IDE Serial Monitor displaying the JSON data string being sent to the Raspberry Pi.

5. Results and Discussion

5.1 System Performance and Results

The integrated system performed as expected. The Arduino successfully polled all sensors, and Home Assistant displayed the data in near real-time. The final dashboard, shown in Figure 7, provides a clean visualization of all monitored parameters. The system was stable during extended testing, with the dashboard updating reliably every few seconds. The local LCD and buzzer also functioned correctly, providing immediate local feedback.

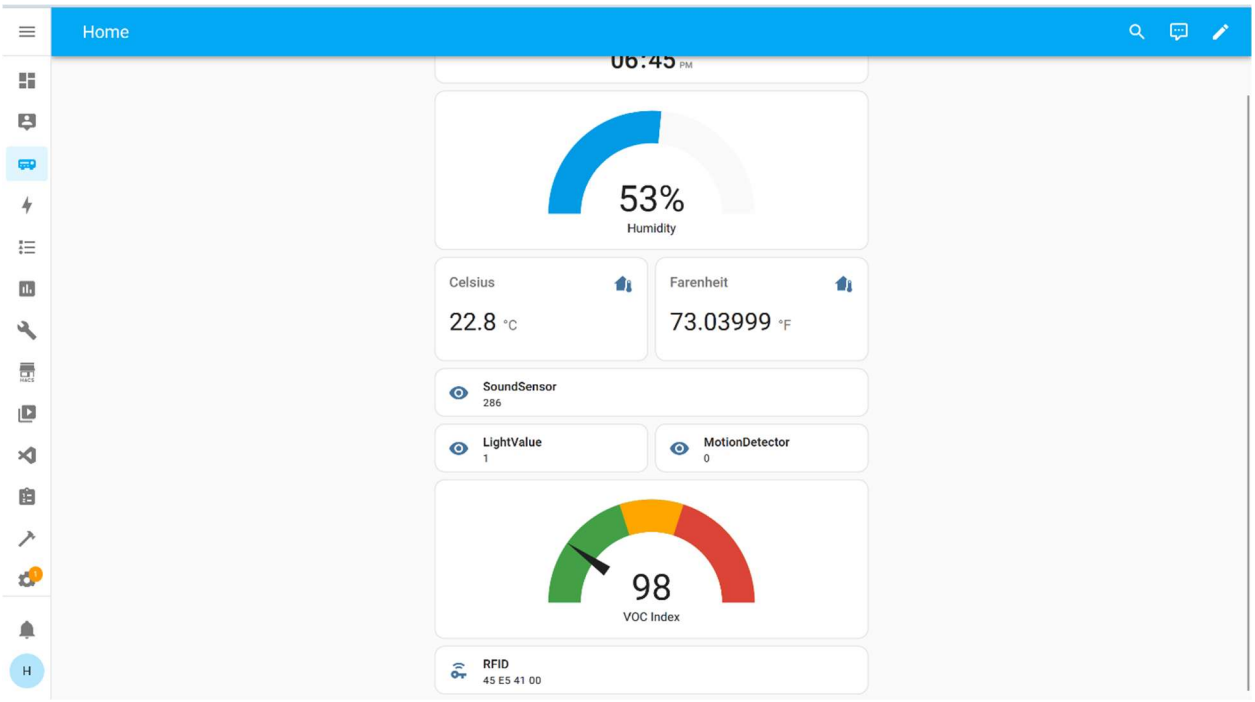

Figure 7 The final Home Assistant dashboard displaying real-time sensor data.

5.2 Challenges, Limitations, and Future Work

While the system was successful, it has several limitations and areas for improvement. The sensors used are hobby-grade and are not calibrated for high-precision scientific work. The DHT11, for example, has a known accuracy of ±2°C. The system's reliance on a breadboard makes it unsuitable for long-term use, as wires can easily come loose.

The most significant limitation is the wired USB connection between the Arduino and the Raspberry Pi, which restricts where the sensor module can be placed.

For future work, the first step would be to design a custom Printed Circuit Board (PCB) to create a compact and reliable sensor unit. To solve the placement issue, the Arduino Mega could be replaced with a microcontroller that has built-in Wi-Fi, such as an ESP32. This would allow the sensor module to communicate with Home Assistant wirelessly over the network using the MQTT protocol. Finally, more complex automations could be built in Home Assistant, such as sending a mobile notification when motion is detected.

6. Conclusion

Our group successfully built a home monitoring system that met all project goals. We integrated multiple sensors with an Arduino and displayed the data on a Home Assistant dashboard. The project provided practical experience in circuit building, microcontroller coding, and hardware debugging. The main takeaway was learning to troubleshoot unexpected problems like the RFID sensor's voltage incompatibility. While the final product is a prototype, it serves as a strong proof-of-concept for a low-cost, customizable smart home system.

7. References

[1] Arduino, "Arduino Mega 2560 REV3 Datasheet," 2023. [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf.

[2] H. Assistant, "Home Assistant Documentation," 2023. [Online]. Available: https://www.home-assistant.io/docs/.

[3] Sensirion, "Datasheet SGP40," 2021. [Online]. Available: https://sensirion.com/media/documents/4AD44315/61644B84/Datasheet_SGP40.pdf.

[4] N. Semiconductors, "MFRC522: Standard MIFARE IC Card Reader/Writer," 2016. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf.

8. Appendix

Appendix A: Full Arduino Source Code

```
#include "DHT.h"
#include "ArduinoJson.h"
#include "Adafruit_SGP40.h"
#include "LiquidCrystal.h"
#include <SPI.h>          // Required library for SPI communication
#include <MFRC522.h>
```

```
#define DHTPIN 2
#define DHTTYPE DHT11
#define SS_PIN 53
#define RST_PIN 5
MFRC522 rfid(SS_PIN, RST_PIN);
DHT dht(DHTPIN, DHTTYPE);
Adafruit_SGP40 sgp;
int buzPin = 3;
int pirPin = 4;
int pirValue;
const int sound_sensor = A1;
const int threshold = 500;

const int rs = 12, en = 11, d4 = 10, d5 = 9, d6 = 7, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
void setup() {

 //dht11 setup
 Serial.begin(9600);
 dht.begin();
 sgp.begin();
 lcd.begin(16, 2);
 //motion detector setup
 pinMode(pirPin, INPUT);
 SPI.begin();           // Begin SPI communication (uses Mega's default SPI pins)
 rfid.PCD_Init();       // Initialize the RFID reader
}

void loop() {
 // put your main code here, to run repeatedly:
 JsonDocument Sensors; //Create JsonDocument named Sensors

 //light sensor
 int lightValueFinal = 0; //0-Dark,1-dim,2-bright
```

```
int lightValue = analogRead(A0);//read anaolgue value from photoresistor
if (lightValue > 800) {
  lightValueFinal = 2;
} else if (lightValue > 400) {
  lightValueFinal = 1;
} else {
  lightValueFinal = 0;
}

//motion sensor
pirValue = digitalRead(pirPin);

//gas sensor
uint16_t sraw;
int32_t voc_index;
float t = dht.readTemperature();
float h = dht.readHumidity();
voc_index = sgp.measureVocIndex(t, h);

//sound sensor
int soundValue = 0;
for (int i = 0; i < 32; i++) {
  soundValue += analogRead(sound_sensor);
}
// Divide sum by 32 using bitshift (faster than division)
soundValue >>= 5;  // Equivalent to: soundValue = soundValue / 32;

//RFID reader
// Check if a new RFID card is present
byte UID[4];
int test;
if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) {
}
else
```

```
 {
 for (byte i = 0; i < rfid.uid.size; i++) {
  // Add leading zero if needed for readability
  //Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");
  //Serial.print(rfid.uid.uidByte[i], HEX); // Print UID byte in HEX format
  UID[i] = rfid.uid.uidByte[i];
 }
 //Serial.println();
 for(byte i = 0; i<4; i++)
 {
 Serial.print(UID[i]);
 Serial.print('-');
 }
 delay(10);  // Small delay to prevent multiple reads of the same tag
 }

//JSON out
 Sensors["Humidity"] = dht.readHumidity(); //read humidity and write to JSON
 Sensors["Celsius"] = dht.readTemperature(); //read temperature in celsius and write to JSON
 Sensors["Farenheit"] = dht.readTemperature(true); //read temperature in celsius and write to JSON
 Sensors["LightValue"] = lightValueFinal;
 Sensors["MotionDetector"] = pirValue;
 Sensors["GasSensor"] = voc_index;
 Sensors["SoundSensor"] = soundValue;

 serializeJson(Sensors, Serial); //Send JSON to serial
 Serial.println();

//Display
 lcd.clear();
 int humidity = dht.readHumidity();
 char hum[10];
 itoa(humidity,hum,10);
 lcd.write(hum);
```

```
  //analogWrite(3, 255);
  delay(1000);
}
```

Appendix B: Home Assistant Configuration (YAML)