LAB REPORT: SENSORS INTERFACE WITH ARDUINO

Course: Embedded Systems

Name: Venkata Siva Sai Krishna Kumar Reddyboina

Student ID: 12503460

Instructor: Prof. Tobias Schaffer

Date: 05-07-2025

1 Introduction

The objective of this lab was to introduce students to sensor interfacing using the Arduino Nano 33 BLE Sense Rev2. The session focused on reading and analyzing data from the onboard IMU and temperature/humidity sensors. This activity strengthened my understanding of how to collect and interpret sensor data—an essential skill for working with embedded systems. Such strategies are particularly relevant in contemporary pacakages like IoT, robotics, and clever devices, wherein sensor enter is crucial for smart machine conduct

2 Methodology

We wrote and deployed Arduino code that accessed and displayed sensor readings using the Arduino IDE. The technique included initializing the serial connection, configuring the sensors, analyzing data from them, and then outputting the statistics to the serial reveal for actual-time commentary.

Software and Hardware Used

Programming Language:

- C++ using the Arduino framework

Libraries:

- Arduino BMI270 BMM150 (for detecting IMU data)
- Arduino HS300x (for temperature and humidity measurements)

Hardware:

- Arduino Nano 33 BLE Sense Rev2
- Micro-USB cable
- Computer with Arduino IDE installed

Code Repository:

`https://github.com/saikumar374/Embedded_Systems`

3 Code Implementation

Reading IMU Sensor Data

```
#include "Arduino_BMI270_BMM150.h"

void setup() {
    Serial.begin(9600);
    while (!Serial);
    if (!IMU.begin()) {
        Serial.println("IMU␣initialization␣failed!");
        while (1);
    }

}

void loop() {
    float x, y, z;
    if (IMU.accelerationAvailable()) {
        IMU.readAcceleration(x, y, z);
        Serial.print("Acceleration:␣X="); Serial.print(x);
        Serial.print("␣Y="); Serial.print(y);
        Serial.print("␣Z="); Serial.println(z);
    }
    delay(500);
}
```

Figure 1:

Reading Temperature and Humidity Data

```
#include <Arduino_HS300x.h>

void setup() {
    Serial.begin(9600);
    while (!Serial);
    if (!HS300x.begin()) {
        Serial.println("HS300␣sensor␣initialization␣failed!");
        while (1);
    }
}

void loop() {
    float temp = HS300x.readTemperature();
    float humidity = HS300x.readHumidity();
    Serial.print("Temp:␣"); Serial.print(temp); Serial.print("C␣␣");
    Serial.print("Humidity:␣"); Serial.print(humidity); Serial.println("%");
    delay(2000);
}
```

Figure 2:

4 Results

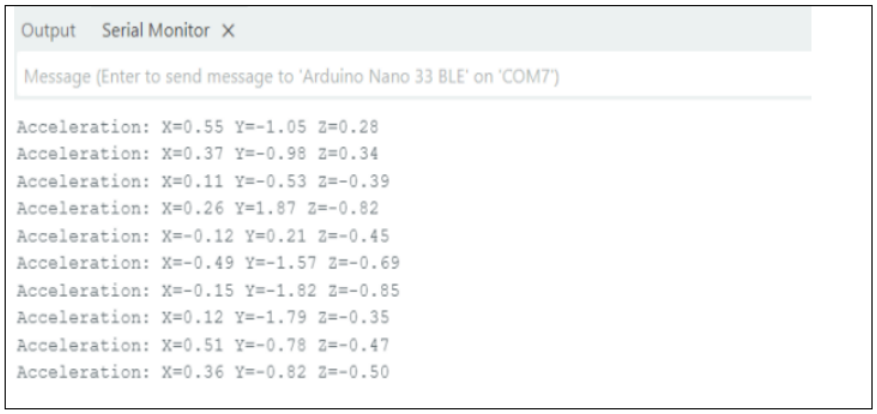

Figure 3: Readings of Real-time acceleration data from the onboard IM

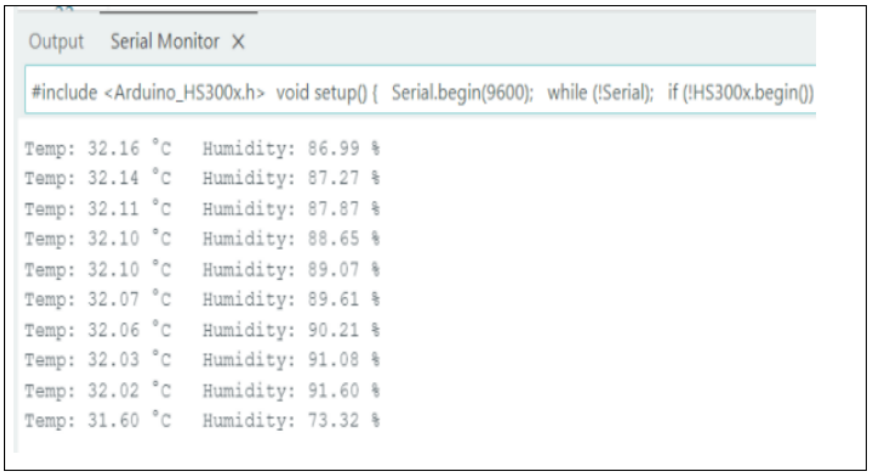

Figure 4: Temperature and humidity values displayed on the Serial Monitor

5 Challenges, Limitations, and Error Analysis

Challenges Encountered

- Installing the IMU and sensor libraries correctly.
- To make suring that all required libraries are properly added to the Arduino IDE.
- Decoding and formatting raw sensor values into readable output.

Error Analysis

- Occasionally, sensor data was incomplete or missing due to communication lags.
- Compilation problems were created by early code misreferences.
- A logic error where the program tried to read sensor values before checking their availability, led to runtime faults.

Limitations Noted

- Only acceleration data was captured; other IMU capabilities like gyroscope and magnetometer weren't used.

- Temperature and humidity readings were obtained but not fully integrated with IMU data.
- No noise filtering was applied, which may have affected the clarity of real-time readings.

6 Discussion

This experiment clearly showed how to work with the built-in sensors on the Arduino Nano 33 BLE Sense Rev2 and use them to collect real-time data. It gave hands-on exposure to highlighted the importance of real-time monitoring in embedded applications and demonstrated how to read and interpret live sensor data. The sensor values responded appropriately to environmental changes and motion, validating the setup.

7 Conclusion

This lab exercise provided foundational experience in sensor interfacing within embedded systems. Students gained practical knowledge in coding, sensor communication, and interpreting physical data via the Arduino platform. Future iterations could enhance functionality by adding features like data logging, wireless communication, and real-time processing with filters or Machine Learning models.