Thao, Phoeuk (2256614)
420-321-VA Unix Sect. 02

Weekly Journal Progress

November 10th – 17th :

Monday, November 11th :

Learned more about Git. Mainly how to use Git Bash and how console commands work with Git.

Website used: https://www.geeksforgeeks.org/working-on-git-bash/
https://www.datacamp.com/tutorial/git-push-pull

Wednesday, November 13th :

Watched a video on Python, how to use python scripts:

https://www.youtube.com/watch?v=dQlw1Cdd3pw&pp=ygUYaG93IHRvIGRvIHB5dGhvbiBzY3JpcHRz

Asked ChatGPT to give me some examples of scripts that can be run using python and what would we use for a humidity sensor project.

Saturday, November 16th :

Looking at Nginx and how we will use it to visualize our presentation. Mainly analyzing how we can mix it with HTML.

Links used: https://nginx.org/en/

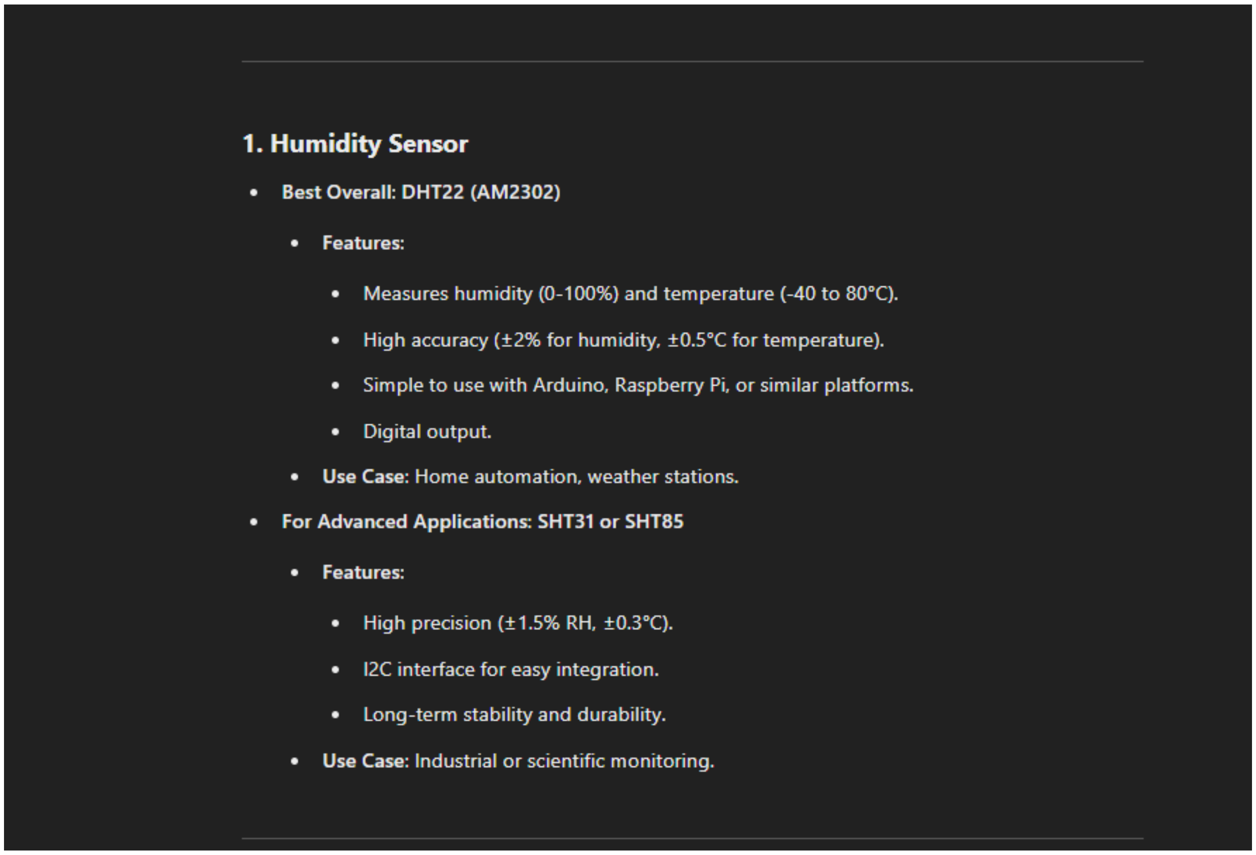
https://www.youtube.com/watch?v=q0tSIv22rgA

November 18th – 24th :

Monday, November 18th :

I used ChatGPT to give me some more insight on our sensors and what model would be perfect for our constrained budgets and lack of expertise in finding material for Tuesday.

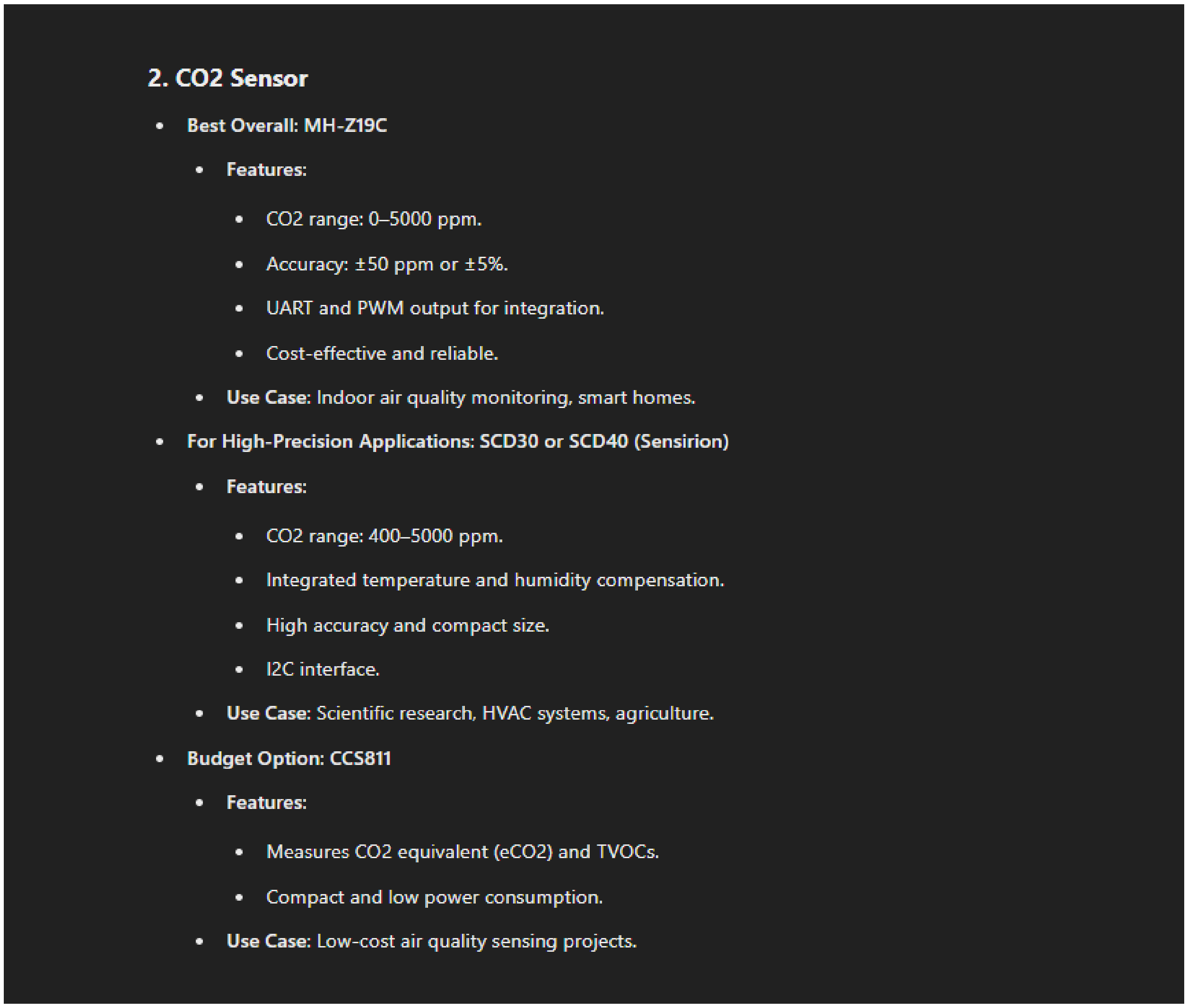
GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git



1. Humidity Sensor

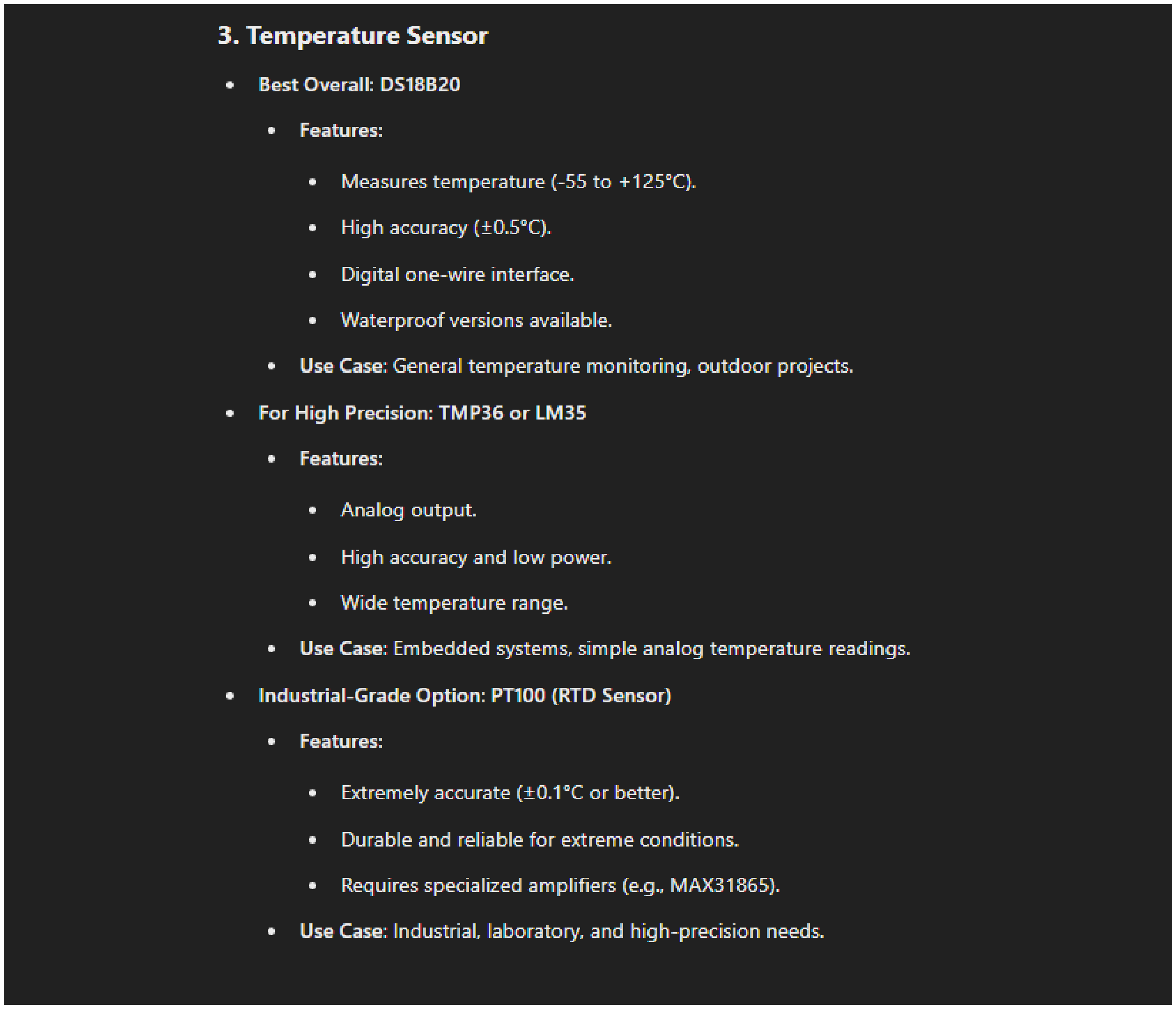
- **Best Overall: DHT22 (AM2302)**
  - **Features:**
    - Measures humidity (0-100%) and temperature (-40 to 80°C).
    - High accuracy (±2% for humidity, ±0.5°C for temperature).
    - Simple to use with Arduino, Raspberry Pi, or similar platforms.
    - Digital output.
  - **Use Case:** Home automation, weather stations.
- **For Advanced Applications: SHT31 or SHT85**
  - **Features:**
    - High precision (±1.5% RH, ±0.3°C).
    - I2C interface for easy integration.
    - Long-term stability and durability.
  - **Use Case:** Industrial or scientific monitoring.

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git



2. CO2 Sensor

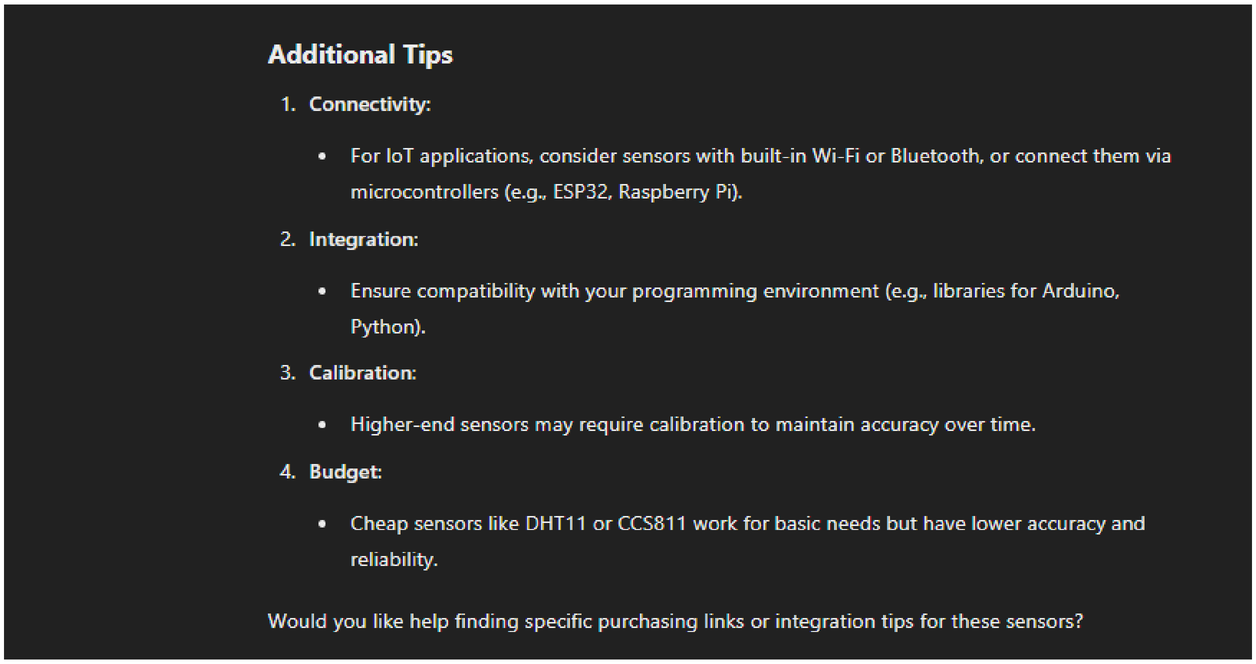
- **Best Overall: MH-Z19C**
  - **Features:**
    - CO2 range: 0–5000 ppm.
    - Accuracy: ±50 ppm or ±5%.
    - UART and PWM output for integration.
    - Cost-effective and reliable.
  - **Use Case:** Indoor air quality monitoring, smart homes.
- **For High-Precision Applications: SCD30 or SCD40 (Sensirion)**
  - **Features:**
    - CO2 range: 400–5000 ppm.
    - Integrated temperature and humidity compensation.
    - High accuracy and compact size.
    - I2C interface.
  - **Use Case:** Scientific research, HVAC systems, agriculture.
- **Budget Option: CCS811**
  - **Features:**
    - Measures CO2 equivalent (eCO2) and TVOCs.
    - Compact and low power consumption.
  - **Use Case:** Low-cost air quality sensing projects.



3. Temperature Sensor

- **Best Overall: DS18B20**
  - **Features:**
    - Measures temperature (-55 to +125°C).
    - High accuracy (±0.5°C).
    - Digital one-wire interface.
    - Waterproof versions available.
  - **Use Case:** General temperature monitoring, outdoor projects.
- **For High Precision: TMP36 or LM35**
  - **Features:**
    - Analog output.
    - High accuracy and low power.
    - Wide temperature range.
  - **Use Case:** Embedded systems, simple analog temperature readings.
- **Industrial-Grade Option: PT100 (RTD Sensor)**
  - **Features:**
    - Extremely accurate (±0.1°C or better).
    - Durable and reliable for extreme conditions.
    - Requires specialized amplifiers (e.g., MAX31865).
  - **Use Case:** Industrial, laboratory, and high-precision needs.

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git



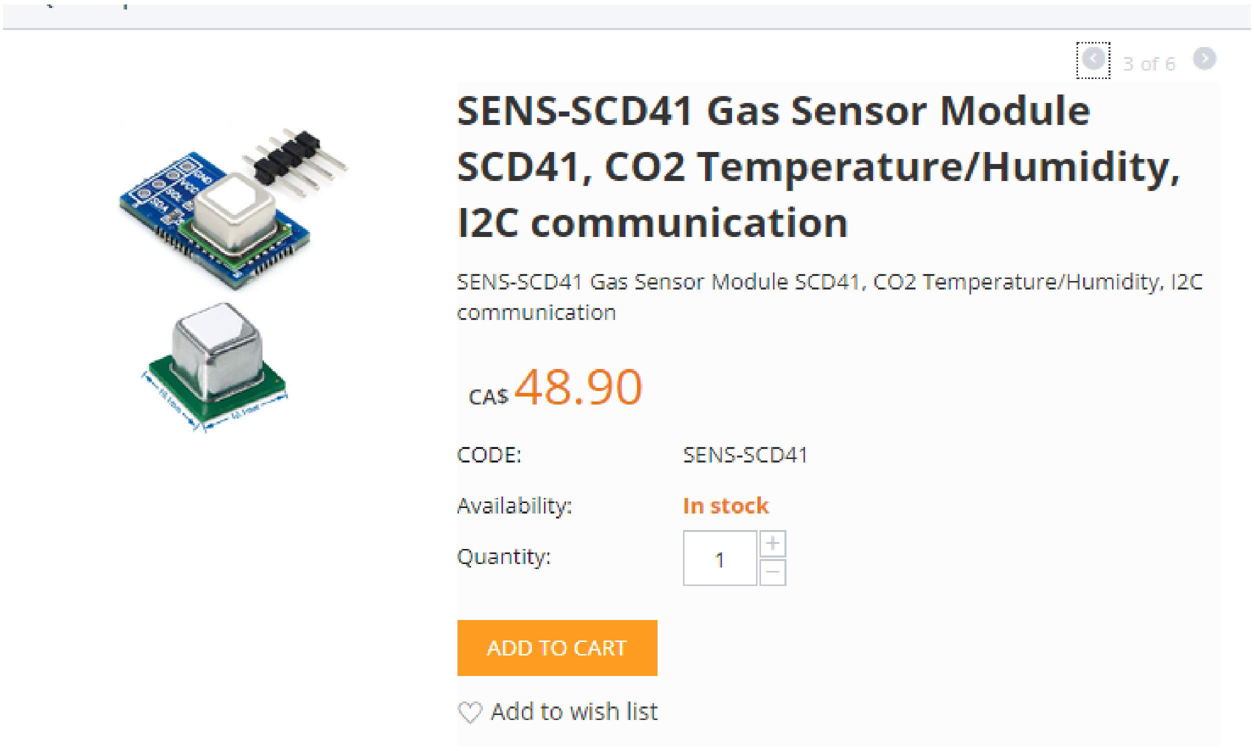
Additional Tips

1. **Connectivity:**
   - For IoT applications, consider sensors with built-in Wi-Fi or Bluetooth, or connect them via microcontrollers (e.g., ESP32, Raspberry Pi).
2. **Integration:**
   - Ensure compatibility with your programming environment (e.g., libraries for Arduino, Python).
3. **Calibration:**
   - Higher-end sensors may require calibration to maintain accuracy over time.
4. **Budget:**
   - Cheap sensors like DHT11 or CCS811 work for basic needs but have lower accuracy and reliability.

Would you like help finding specific purchasing links or integration tips for these sensors?

Tuesday, November 19th:

After the morning class, my partners and I went to search for the parts required to build our project. We found the sensor we wanted to purchase as well as a few other equipments like a board and wires to make sure we can have our project work.



SENS-SCD41 Gas Sensor Module SCD41, CO2 Temperature/Humidity, I2C communication

SENS-SCD41 Gas Sensor Module SCD41, CO2 Temperature/Humidity, I2C communication

CA$ 48.90

CODE:	SENS-SCD41
Availability:	In stock
Quantity:	1

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git

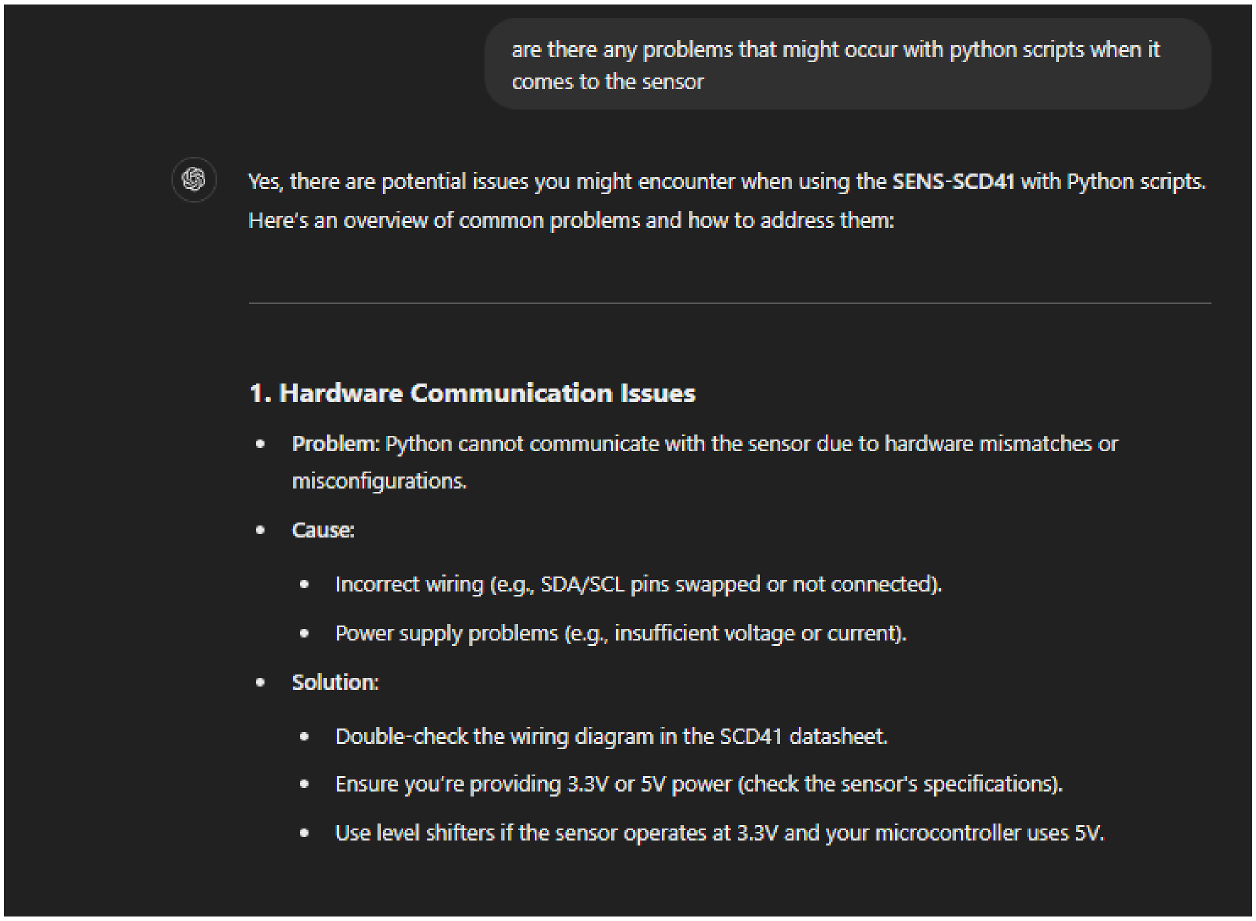
Thao, Phoeuk (2256614)
420-321-VA Unix Sect. 02

This is the sensor we bought: The SENS-SCD41 Gas Sensor. As for why I did not go with the options that ChatGPT brought forward is because after discussing it with my teammates, we realized that either the options were too expensive or too niche for the project we wanted to build.

Link to the ABRA website where we purchased it: https://abra-electronics.com/sensors/sensors-gas/scd40-scd41-gas-sensor-module.html

Wednesday, November 20th :

After purchasing the pieces with my teammates, I decided to go ask ChatGPT, if there were any constraints or eventual problems that could arise with a python scripts and the sensor chosen.



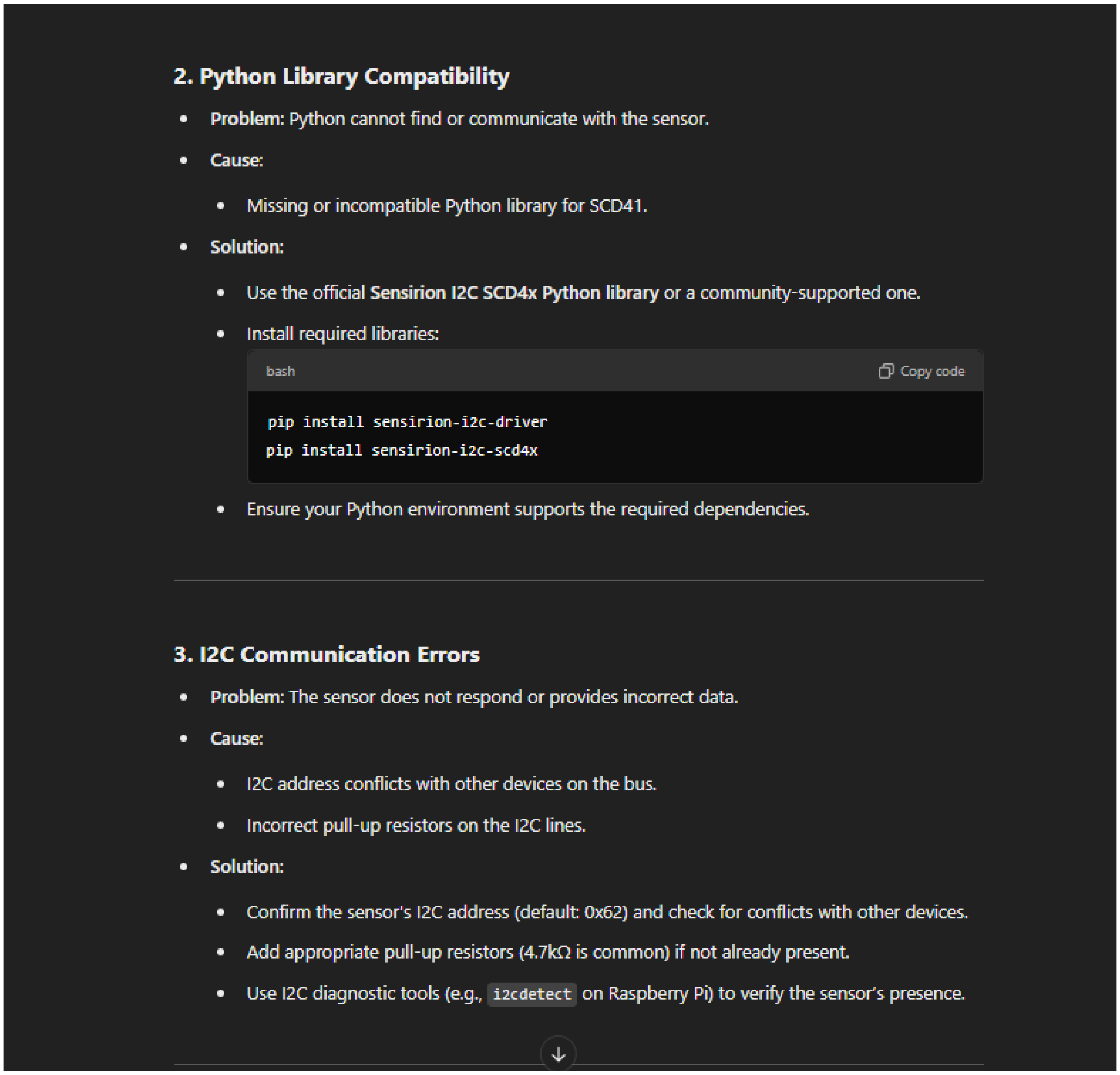
are there any problems that might occur with python scripts when it comes to the sensor

Yes, there are potential issues you might encounter when using the **SENS-SCD41** with Python scripts. Here's an overview of common problems and how to address them:

---

1. Hardware Communication Issues

- **Problem:** Python cannot communicate with the sensor due to hardware mismatches or misconfigurations.
- **Cause:**
  - Incorrect wiring (e.g., SDA/SCL pins swapped or not connected).
  - Power supply problems (e.g., insufficient voltage or current).
- **Solution:**
  - Double-check the wiring diagram in the SCD41 datasheet.
  - Ensure you're providing 3.3V or 5V power (check the sensor's specifications).
  - Use level shifters if the sensor operates at 3.3V and your microcontroller uses 5V.



2. Python Library Compatibility

- **Problem:** Python cannot find or communicate with the sensor.
- **Cause:**
  - Missing or incompatible Python library for SCD41.
- **Solution:**
  - Use the official **Sensirion I2C SCD4x Python library** or a community-supported one.
  - Install required libraries:

```bash
pip install sensirion-i2c-driver
pip install sensirion-i2c-scd4x
```

  - Ensure your Python environment supports the required dependencies.

---

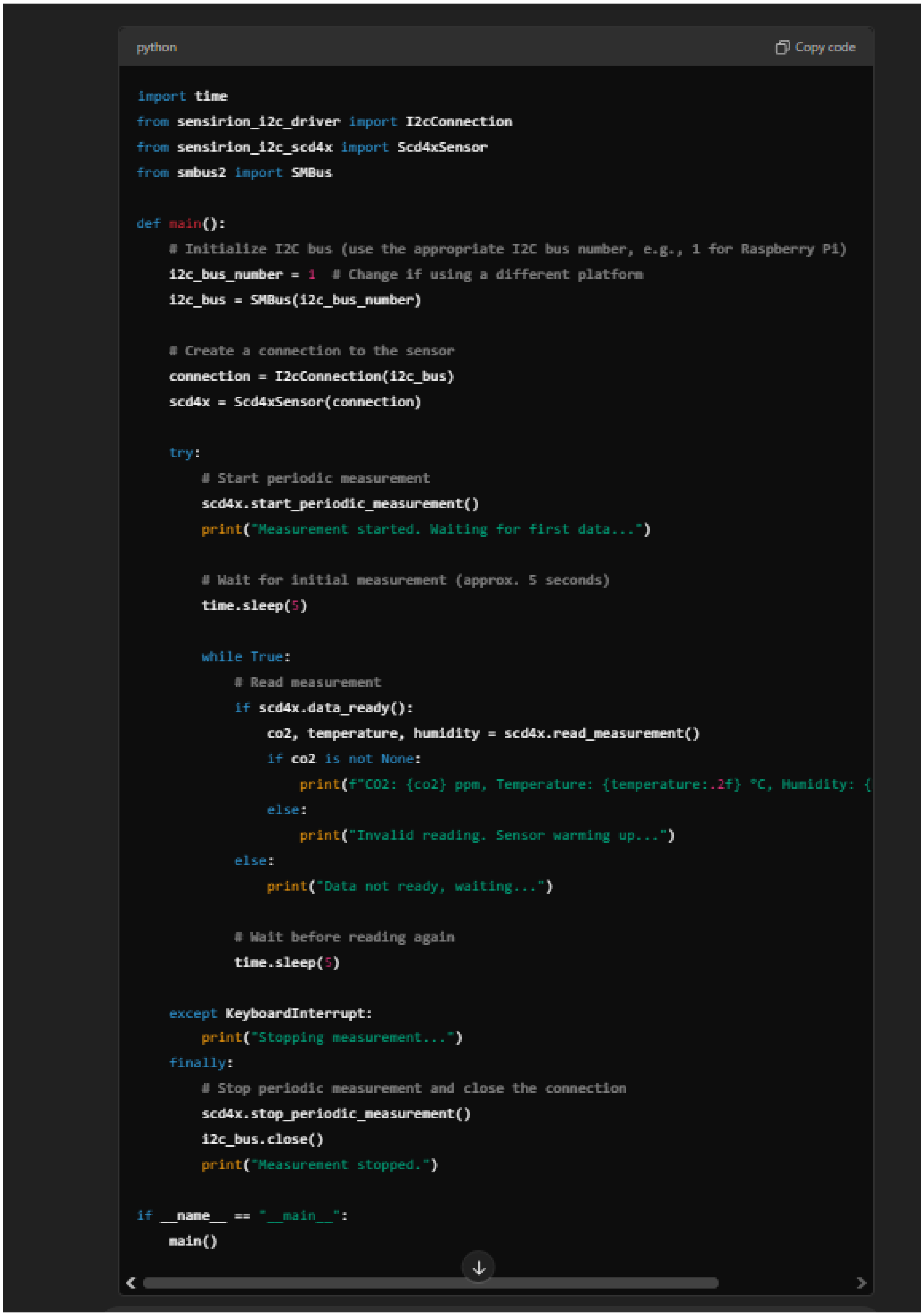
3. I2C Communication Errors

- **Problem:** The sensor does not respond or provides incorrect data.
- **Cause:**
  - I2C address conflicts with other devices on the bus.
  - Incorrect pull-up resistors on the I2C lines.
- **Solution:**
  - Confirm the sensor's I2C address (default: 0x62) and check for conflicts with other devices.
  - Add appropriate pull-up resistors (4.7kΩ is common) if not already present.
  - Use I2C diagnostic tools (e.g., `i2cdetect` on Raspberry Pi) to verify the sensor's presence.

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git

I also asked it to see with these probable errors if it can make us a basic python script and it came up with this.

Thao, Phoeuk (2256614)
420-321-VA Unix Sect. 02



```python
import time
from sensirion_i2c_driver import I2cConnection
from sensirion_i2c_scd4x import Scd4xSensor
from smbus2 import SMBus

def main():
    # Initialize I2C bus (use the appropriate I2C bus number, e.g., 1 for Raspberry Pi)
    i2c_bus_number = 1  # Change if using a different platform
    i2c_bus = SMBus(i2c_bus_number)

    # Create a connection to the sensor
    connection = I2cConnection(i2c_bus)
    scd4x = Scd4xSensor(connection)

    try:
        # Start periodic measurement
        scd4x.start_periodic_measurement()
        print("Measurement started. Waiting for first data...")

        # Wait for initial measurement (approx. 5 seconds)
        time.sleep(5)

        while True:
            # Read measurement
            if scd4x.data_ready():
                co2, temperature, humidity = scd4x.read_measurement()
                if co2 is not None:
                    print(f"CO2: {co2} ppm, Temperature: {temperature:.2f} °C, Humidity: {
                else:
                    print("Invalid reading. Sensor warming up...")
            else:
                print("Data not ready, waiting...")

            # Wait before reading again
            time.sleep(5)

    except KeyboardInterrupt:
        print("Stopping measurement...")
    finally:
        # Stop periodic measurement and close the connection
        scd4x.stop_periodic_measurement()
        i2c_bus.close()
        print("Measurement stopped.")

if __name__ == "__main__":
    main()
```

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git

Thao, Phoeuk (2256614)
420-321-VA Unix Sect. 02

Thursday, November 21st :

I went online to search for videos on how to install sensors and how to link it with a raspberry pi. Although I did not find what I was looking for exactly, here is what I watched:

https://www.youtube.com/watch?v=6UOiGkbp6w8
https://www.youtube.com/watch?v=BpJCAafw2qE
https://www.youtube.com/watch?v=ELznPFK1JJE

https://www.youtube.com/watch?v=Gl9HS7-H0mI&t=2s

I find these videos pertinent to our project because they make me understand what I am dealing with, how I would start using the raspberry pi and how said raspberry pi can run scripts.

Friday, November 22nd :

On Friday I decided to look up some more details about our sensor. (Where it should be placed in a room for the best and most accurate readings) I found that placing it on the higher part of a wall or the ceiling would make the readings more accurate. Next to a window is not preferred due to us wanting the reading in a room with as little outside influence as possible.

https://learn.pimoroni.com/article/co2-detection-with-scd41#:~:text=You%20should%20position%20detectors%20away,can%20cause%20disproportionately%20high%20readings.

https://download.mikroe.com/documents/datasheets/SCD41%20Datasheet.pdf

https://sensirion.com/media/documents/0D0C9129/623B1183/Sensirion_CO2_Sensors_SCD4x_design-in_guide.pdf

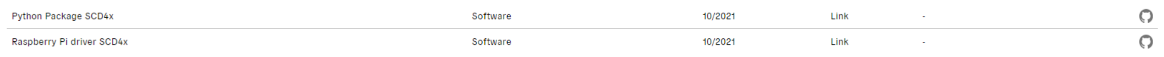
November 25th – December 2nd :

Sorry, I'm so late. I was waiting for a group up with my teammates today to discuss this sensor project.

Monday, November 25th :

Monday, I went and did some more research on the types of scripts the AI did for me. I saw that all if not most of them came from something called sensirion. I went to check if they had anything on their page but to no avail. However, when I went on the page of the sensor we bought (SCD41), I found some interesting data about said device. Among the data that caught my attention were some downloadable files called "Python Package SCD4x" and Raspberry Pi Driver SCD4x. They linked to a git repo and that's where I found the right things we needed.

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git



Python Package SCD4x	Software	10/2021	Link	-	
Raspberry Pi driver SCD4x	Software	10/2021	Link	-	

Link: https://sensirion.com/products/catalog/SCD41

https://github.com/Sensirion/raspberry-pi-i2c-scd4x

https://github.com/Sensirion/python-i2c-scd

Wednesday, November 27th :

After I found out about the git repositories, I went to tell my teammates. Turns out, they found them much earlier than I did. I then waited for Tarek, Rafea who had the components to assemble the project and run the scripts we found. This turned out to be very long as we had to wait until we were all together to talk face to face. This is why I waited so long to send this journal. I thought to myself this is too short for a third journal entry.

Monday, December 2nd:

We scheduled a meeting for this Monday, we gathered around together to discuss the sensor. Tarek brought the sensor connected to the raspberry pi and we saw that it worked. It sent the output to a .log file after running a .py script and a .service script. We then wanted to visualize the data to a Nginx file using Flask. That turned out to be very difficult. We spent almost 4 hours on trying to make a webpage work. We still are not able to make it work as of today. Me and Tarek, Abou Chahin are going to try to make a working webpage with data.

Outcome and expectations:

I went into this project with a solid plan, week 1 get the information and resources then, week 2 assemble the project and have some tools and a basic script working and week 3 being the week where we finish the project by adding a webpage and cleaning up the data. This, however, proved to be not what we had imagined at all. First, our basic idea was to have oxygen sensor. We went to buy a CO2 sensor, similar but very different. Second, the data visualization, while sounds simple on paper, was a bigger challenge than we anticipated. Third, the tentative week plans went to the dump, since we realized that only one person can work on the actual thing at one time, I had to mostly do research. This put our progress at a halt. All in all, the project presented major hurdles as a team. A hard thing this was and I hope that our presentation can show this hard and rough work we put in.

GitHub link: https://github.com/Tarek-rf/UnixFinalProject.git