
# LABORATORY PROJECT REPORT

SERIAL AND USB INTERFACING WITH MICROCONTROLLER

## EXPERIMENT 4A

**DATE : 31TH MARCH 2025**

**SECTION : 1**

**GROUP : 1**

**SEMESTER 2, 2024/2025**


| NO | NAME | MATRIC NO |
|---|---|---|
| **1.** | AKMAL FAUZAN BIN AZHAR | 2319531 |
| **2.** | AIMAN SAIFULLAH BIN AMINNURLLAH | 2319185 |
| **3.** | SYAFIQ HELMIE BIN SAIFULLAH ADHA | 2316049 |

**Table of Contents**

## 1.0 Introduction

This experiment aims to demonstrate the principles of inertial sensing and I2C communication by establishing a data exchange link between an MPU6050 Inertial Measurement Unit (IMU), an Arduino Uno, and a personal computer. The objective is to collect real-time motion and orientation data—specifically acceleration and angular velocity—from the MPU6050 sensor and transmit this data to a Python script running on the computer. Through this experiment, students will gain practical experience with microcontroller interfacing, sensor data acquisition, and communication protocols commonly used in embedded systems.

The MPU6050 integrates a 3-axis accelerometer and a 3-axis gyroscope into a single compact and cost-effective module, making it a versatile component in applications involving motion tracking, gesture recognition, and orientation sensing. In this experiment, the sensor communicates with the Arduino via the I2C protocol, enabling synchronized data collection with minimal wiring complexity. The Arduino processes the raw data from the MPU6050 and transmits it via serial communication to a Python script utilizing the pyserial library, where the data can be monitored and further analyzed.

The hypothesis for this experiment is that, when the sensor is properly connected and the respective Arduino and Python codes are correctly implemented, the MPU6050 will provide accurate real-time acceleration and gyroscopic data to the computer terminal. This successful transmission and display of data will validate the integration of hardware and software systems, while reinforcing core embedded system concepts such as I2C communication, sensor calibration, and real-time data handling.

## 2.0 Materials and Equipments

### 2.1 Electronic Components

- Arduino Board – Microcontroller board for controlling the display
- MPU6050 Sensor - to read data such as detecting hand gestures
- Connecting Wires - establish connections between components

### 2.2  Equipment and Tools

- Arduino IDE – Software for writing and uploading code to the Arduino Uno
- USB cable – For connecting the Arduino Uno to a computer
- Power Supply  – To power the circuit

# 3.0 Experimental Setup

## 3.1 Diagram of the Arduino Setup

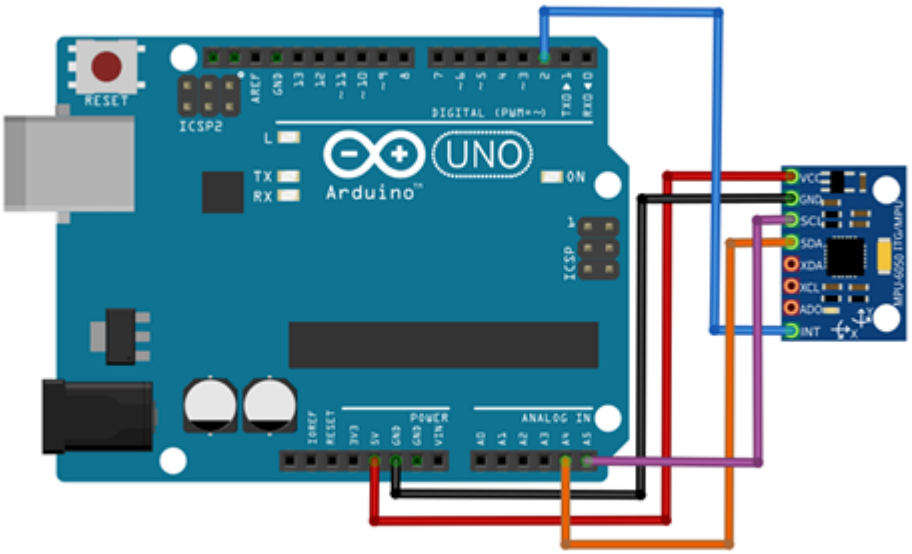



*Figure 1 : Arduino-MPU6050 Connections*

## 3.2 Circuit Setup

1. The MPU6050 sensor was connected to the Arduino board using the appropriate pins. The MPU6050 typically used I2C communication, so the SDA and SCL pins of the MPU6050 were connected to the corresponding pins on the Arduino (usually A4 and A5 for most Arduino boards).
2. The power supply and ground of the MPU6050 were connected to the Arduino's 5V and GND pins.
3. The Arduino board was connected to the PC via USB.

# 4.0 Methodology

## 4.1 Implementation and Testing

1. The circuit was built using an arduino board, MPU6050 sensor and connecting wires.
2. The MPU6050 sensor was connected with:
    - VCC pin (power supply) to 5V on Arduino.
    - GND pin to GND on Arduino.
    - SCL pin to A5 (Analog in).
    - SDA pin to A4 (Analog in).
3. Then, the arduino board was connected to the PC via USB.
4. The Arduino code was uploaded via the Arduino IDE to enable readings of the accelerometer and gyroscope data from the MPU6050 sensor and it was sent to the PC via serial port at a baud rate of 9600.
5. A python code using the pyserial library was run to receive the data of the accelerometer and gyroscope data from the MPU6050 sensor and printed on the PC's console.

## 4.2 Control Algorithm

The Arduino Uno was programmed using the Arduino IDE to read analog values from the MPU6050 sensor which was previously hand gestures data and send them via serial communication. The Python script handled the reception and display of this data. The control algorithm on the Arduino followed these steps:

1. Initialize the serial communication at a baud rate of 9600.
2. I2c communication was initialized using **wire.begin()** and activated the MPU6050 sensor with **mpu.initialize()**.
3. The motion data is monitored from the sensor and performs gestures recognition in real time.
4. When a new gesture was detected, the system printed the corresponding gesture label to the serial motor.

5. If predefined gesture conditions threshold was detected, it will return the specific value such as “Gesture 1”.
6. If other conditions were detected, it will also return values to identify additional gestures.
7. If no known gesture was detected, the function will return zero, indicating no gesture was detected.

The Python script followed these steps:

1. Establish a serial connection using the pyserial library.
2. Continuously read incoming data from the serial port.
3. Decoded each line of received data from bytes to a UTF-8 string and remove any leading or trailing whitespace.
4. Checked if the incoming data contained a gesture notification by identifying lines that began with the phrase “Detected Gestures”.
5. Extracted the specific gesture label from the message and executed a corresponding action for each recognized gesture. For example, when "Gesture 1" was detected, the script printed “Action for Gesture 1” to the terminal.
6. And also other gesture types, such as "Gesture 2", triggered their respective responses.

## 7.0 Results

The experiment successfully demonstrated a simple yet effective hand gesture recognition system by utilizing real-time motion data captured from the MPU6050 sensor. During execution, the Arduino Uno accurately read accelerometer and gyroscope values from the MPU6050 via I2C communication and employed a threshold-based algorithm to identify predefined hand gestures. These gestures were then transmitted over a serial connection to a Python script, where they were correctly recognized and triggered corresponding responses.

As each gesture was performed, the system consistently detected the intended motion pattern and output the appropriate gesture label—such as "Gesture 1" or "Gesture 2"—to the Python terminal. The serial communication between the Arduino and the computer was stable and responsive, with gesture updates reflected in near real-time. The baud rate of 9600 on both devices ensured smooth data transmission and proper synchronization between hardware and software components.

In addition to terminal output, the experiment also allowed for the potential visualization of motion paths in an x-y coordinate system. While the code provided focused primarily on gesture recognition, the continuous stream of accelerometer data enabled further development toward graphical representation, making this system suitable for motion tracking and basic movement mapping.

Overall, the results confirmed the successful integration of the MPU6050 sensor, Arduino control logic, and Python-based serial communication. The system remained stable throughout the experiment and effectively demonstrated the practical application of embedded motion sensing and real-time data interaction in gesture-based interfaces.

## 8.0 Discussion

This experiment successfully shows a simple hand gesture recognition system using the MPU6050 sensor, an Arduino Uno, and a Python script. By capturing real-time motion data from the accelerometer and gyroscope, the system was able to detect and respond to specific

hand movements. When a gesture was recognized by the Arduino, it was sent to the computer via serial communication and displayed clearly in the Python terminal.

The gestures were consistently identified, and the Python script reacted as predicted, confirming that both the hardware and code were working together smoothly. This also showed the effectiveness of using threshold-based conditions for gesture classification and highlighted how easy it is to link embedded systems with a PC for interactive applications.

Some challenges included occasional missed or incorrect gestures, likely due to quick or inconsistent hand movements or minor sensor noise. These could be improved with better filtering or fine-tuning the thresholds. Ensuring stable wiring and secure sensor placement also helped reduce errors.

Overall, the experiment was a great introduction to real-time motion tracking and serial communication. It not only reinforced key concepts in embedded systems but also opened the door for more advanced projects like gesture-based controls or motion visualization.

## 9.0 Conclusion

To conclude this experiment, we successfully implemented a basic hand gesture recognition system using the MPU6050 sensor, an Arduino board, and a Python script. By capturing real-time accelerometer and gyroscope data, we were able to identify and categorize specific hand gestures based on predefined motion patterns. The recognized gestures were accurately transmitted to the computer via serial communication and processed using conditional logic in Python, confirming the effective integration of sensor input, microcontroller processing, and software-based response handling.

This project reinforced key embedded systems concepts, including I2C communication, motion sensing, serial data exchange, and gesture classification. The system's responsiveness to different hand movements validated both the hardware connections and the correctness of the implemented algorithms. Furthermore, the experiment lays a strong foundation for more advanced gesture-based applications, with potential enhancements such as dynamic gesture detection, real-time x-y coordinate path visualization, and more robust machine learning approaches for improved accuracy and flexibility in gesture interpretation.

## 10.0 Recommendations

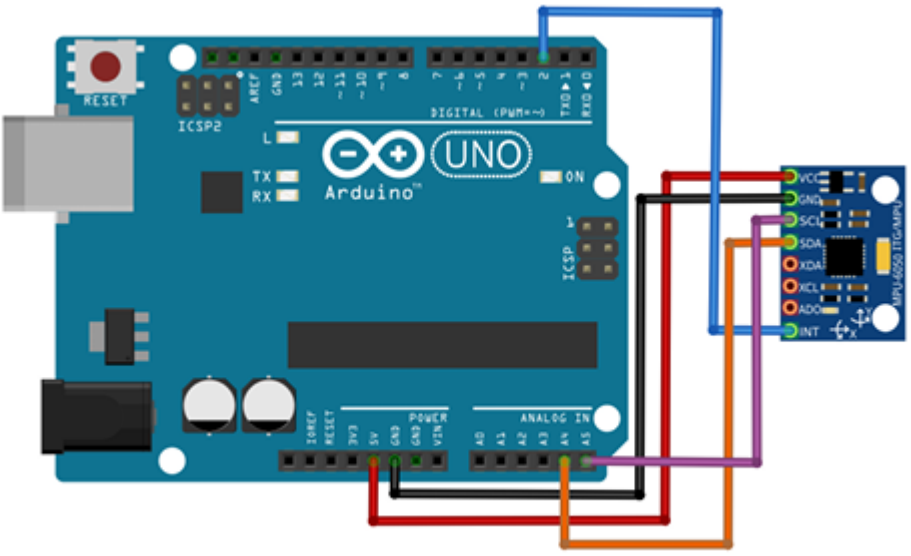
To make the hand gesture recognition system more reliable and user-friendly, one of the first steps would be to clean up the code by using functions—both in the Arduino and Python scripts. This helps organize repetitive tasks, makes the code easier to read, and saves time when making future updates or debugging issues. During development, using the Serial Monitor as a debugging tool can be incredibly helpful for spotting communication problems or unexpected sensor behavior early on.

It's also a good idea to double-check the hardware setup. Sometimes, something as simple as a loose jumper wire or a slightly damaged sensor can throw off your readings. Making sure all components—especially the MPU6050 and any wiring—are in good condition will go a long way in improving the system's accuracy and consistency.

Looking ahead, adding features like real-time visualization can take the project to the next level. For example, using Python libraries to plot motion data on an x-y coordinate system can help better understand the gestures and fine-tune detection. A small OLED or I2C display connected to the Arduino could also provide instant on-device feedback, making the system feel more interactive and polished.

Lastly, moving away from a USB connection by integrating wireless communication—like Bluetooth or WiFi—could make the setup more flexible and portable. This opens the door to more creative applications, such as wearable gesture controllers or remote-controlled systems. With a few thoughtful upgrades, this project has the potential to evolve into a powerful and versatile gesture recognition platform.

## 11.0 References




- Figure 1 (above) : *Arduino-MPU6050 Connections*
- *Weekly Module - Google Drive*. (n.d.). Google Drive. https://drive.google.com/drive/folders/1rq0wLF6mA7jEoNsPWyAbWR9n0X3SQsQz?usp=sharing

## 12.0 Acknowledgement

Special thanks to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, as well as the teaching assistants and peers, for their guidance and support in completing this experiment.

## 13.0 Student's Declaration

Certificate of Originality and Authenticity

This is to certify that we are responsible for the work submitted in this report, that the original work is our own except as specified in the references and acknowledgement, and that the original work contained herein have not been untaken or done by unspecified sources or persons.

We hereby certify that this report has not been done by only one individual and all of us have contributed to the report. The length of contribution to the reports by each individual is noted within this certificate.

We also hereby certify that we have read and understand the content of the total report and no further improvement on the reports is needed from any of the individual's contributors to the report.

We therefore, agreed unanimously that this report shall be submitted for marking and this final printed report has been verified by us.

| | | |
|---|---|---|
| / | Name: Aiman Saifullah bin Aminnurllah | Read |
| / | Matric Number: 2319185 | Understand |
| / | Signature: | Agree |

| | | |
|---|---|---|
| / | Name: Akmal Fauzan Bin Azhar | Read |
| / | Matric Number:2319531 | Understand |
| / | Signature: | Agree |

| | | |
|---|---|---|
| / | Name: Syafiq Helmie bin Saifullah Adha | Read |
| / | Matric Number: 2316049 | Understand |
| / | Signature: | Agree |