REPORT 3 : SERIAL AND USB INTERFACING WITH MICROCONTROLLER AND COMPUTER BASED SYSTEM (2): SENSORS AND ACTUATORS)

GROUP 4

MCTA 3203

SEMESTER 1 2024/2025

MECHATRONICS SYSTEM INTEGRATION

DATE OF SUBMISSION: 6 NOVEMBER 2024

NO	NAME	MATRIC NUMBER
1.	IRDINA NABIHAH BINTI MOHD NAZRI	2214772
2.	KHALISAH AMANI BINTI YAHYA AZMI	2218184
3.	LUQMAN AZFAR BIN AZMI	2219857
4.	MOHAMAD NASRI BIN MOHAMAD NAZRI	2219879
5.	MUHAMAD NURHAKIMIE THAQIF BIN ABDULLAH	2213217

TABLE OF CONTENT

INTRODUCTION

In the mechatronics area, the use of sensors combined with microcontrollers allows the development of systems that can interact with the environment. The present manual focuses on the MPU6050 sensor and, owing to its small size, low price, and easy interfacing, it can be applied to a wide range of projects requiring motion and orientation data. In a nutshell, the main steps towards connecting a personal computer with an MPU 6050 via an Arduino board start with hardware setup and coding. While using both Arduino and Python, one can effectively read data and process it from an MPU 6050 for innovation projects such as gesture recognition systems .

ABSTRACT

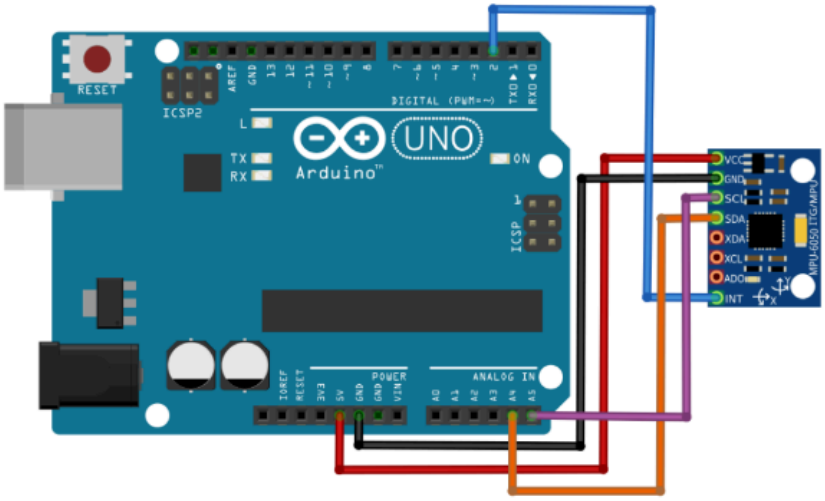
This experiment explores the integration of the MPU6050 sensor with microcontrollers, specifically focusing on its application in motion and orientation data acquisition. The MPU6050, due to its compact size, affordability, and ease of interfacing, is an ideal choice for a wide range of mechatronics projects. The study outlines the process of connecting a personal computer to the MPU6050 via an Arduino board, detailing the necessary hardware setup and coding procedures. By utilizing both Arduino and Python, the system effectively reads and processes data from the MPU6050, enabling its use in innovative projects, such as gesture recognition systems.

EXPERIMENT 4A: <u>Demonstrating a connection between a personal computer and an IMU (Inertial Measurement Unit) MPU6050 through an Arduino board,</u>

MATERIALS AND EQUIPMENT

- Arduino board
- MPU6050 sensor
- Computer with Arduino IDE and Python installed
- Connecting wires: Jumper wires or breadboard wires to establish the connections between the
- Arduino, MPU6050, and the power source.
- USB cable: A USB cable to connect the Arduino board to your personal computer. This will be used
- for uploading the Arduino code and serial communication.
- Power supply: If your Arduino board and MPU6050 require an external power source, make sure to
- have the appropriate power supply.
- LEDs of different colours.

EXPERIMENTAL SETUP



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

METHODOLOGY

1. Hardware Setup

- The MPU6050 sensor was connected to the Arduino via I2C, using the SDA and SCL pins (typically pin 20 and 21 on the Arduino).
- Power and ground lines were connected between the MPU6050 and Arduino's 5V and GND pins.
- The Arduino board was connected to the PC via a USB cable for both code uploading and serial communication

2. Software Programming and Execution

- Arduino Code Development:
    - Code was written in the Arduino IDE to initialize the MPU6050, read accelerometer and gyroscope data, and send this data to the PC via serial communication at a baud rate of 9600.
    - A separate Arduino code version included gesture detection based on threshold values in the accelerometer data to identify specific hand movements (gestures).

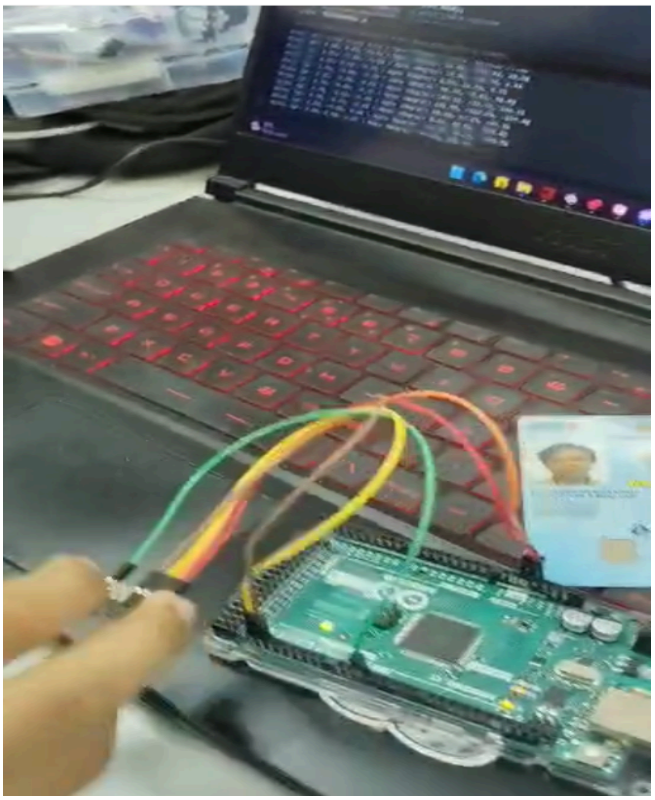
- Running the Python Script
    - To execute the script, ensure that any necessary libraries, such as pyserial are installed and display serial data from the Arduino.
    - The data was continuously printed to the console, allowing for real-time observation of accelerometer and gyroscope values.

- In the gesture recognition setup, the script was enhanced to identify specific gestures based on detected data patterns and to print corresponding actions.

- Gesture Recognition Task
  - To recognize gestures, threshold values were defined within the Arduino code to classify different hand movements.
  - The Python script on the PC processed the data to provide feedback or perform actions associated with each gesture.

- Execution and Visualization
  - The setup was tested by executing predefined hand movements. The accelerometer and gyroscope data were collected, processed, and displayed on the PC console.
  - Visualization of hand paths on an x-y coordinate system was suggested as an enhancement for tracking gesture movement paths.

RESULTS



Connection of Arduino

Video Link: https://github.com/nasrinazri/Group-4-lab-/blob/main/Week%204/A/Gyro_Acc.mp4

```
#include <Wire.h>
#include <MPU6050.h>

// Class to handle gesture detection using MPU6050
class GestureDetector {
private:
    MPU6050 mpu;
    int previousGesture;

    // Gesture detection thresholds
    static const int ACCELERATION_THRESHOLD = 800;  // Sensitivity

    // Gesture types
    static const int NO_GESTURE = 0;
    static const int GESTURE_ONE = 1;  // Forward tilt
    static const int GESTURE_TWO = 2;  // Backward tilt

public:
    GestureDetector() : previousGesture(-1) {}

    void begin() {
        Wire.begin();
        mpu.initialize();

        // Verify connection
        if (!mpu.testConnection()) {
            Serial.println("MPU6050 connection failed!");
            while (1);
        }
        Serial.println("MPU6050 connection successful!");
    }

    void update() {
        int currentGesture = detectGesture();

        // Gesture Update
        if (currentGesture != previousGesture) {
            printGesture(currentGesture);
```

```
            performGestureAction(currentGesture);
            previousGesture = currentGesture;
        }
    }

private:
    int detectGesture() {
        int16_t ax, ay, az;    // Acceleration values
        int16_t gx, gy, gz;    // Gyroscope values

        // Get raw sensor data
        mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

        // Gesture recognition logic
        if (ax > ACCELERATION_THRESHOLD && ay <
ACCELERATION_THRESHOLD) {
            return GESTURE_ONE;
        }
        else if (ax < -ACCELERATION_THRESHOLD && ay >
ACCELERATION_THRESHOLD) {
            return GESTURE_TWO;
        }

        return NO_GESTURE;
    }

    void printGesture(int gesture) {
        Serial.print("Detected Gesture: ");
        switch (gesture) {
            case GESTURE_ONE:
                Serial.println("Forward Tilt");
                break;
            case GESTURE_TWO:
                Serial.println("Backward Tilt");
                break;
            case NO_GESTURE:
                Serial.println("None");
                break;
        }
```

```
    }
};

// Global instance of gesture detector
GestureDetector gestureDetector;

void setup() {
    Serial.begin(9600);
    gestureDetector.begin();
}

void loop() {
    gestureDetector.update();
    delay(50);  // Delay to prevent serial clot
}
```

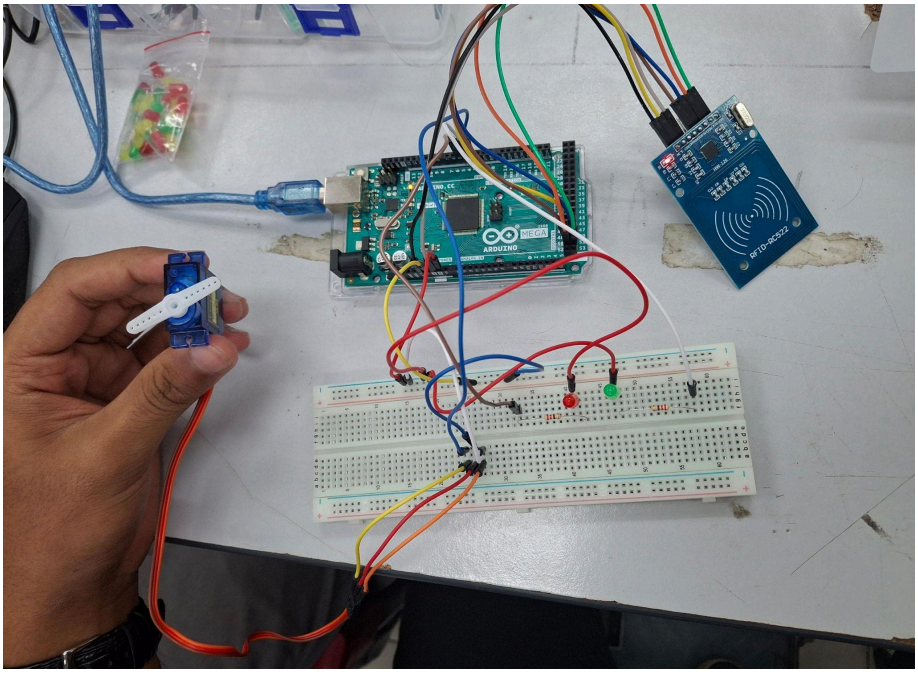
By using the MPU6050 sensor we detect the tilt gestures. The GestureDetector class initializes the sensor and defines gesture thresholds. detectGesture() reads accelerometer data, identifying "forward" or "backward" tilts if acceleration exceeds a set threshold. Detected gestures are printed and trigger corresponding actions. In setup(), the serial connection and sensor are initialized. The loop() function continuously checks for gestures, updating only when a new one is detected. printGesture() displays the current gesture in the serial monitor

EXPERIMENT 4B: Proving RFID card authentication and controlling a servo motor with Python and Arduino,

MATERIALS AND EQUIPMENT

- Arduino board
- RFID card reader with USB connectivity
- RFID tags or cards that can be used for authentication
- Servo Motor: A standard servo motor to control the angle
- Jumper wires
- Breadboard
- LEDs of various colours
- USB cables to connect the Arduino board and the RFID reader to your computer.
- Computer with Arduino IDE and Python installed
- Datasheets and Manuals: Make sure you have the datasheets or manuals for the RFID reader, servo motor, and any other components you are using. Most of them can be downloaded from the internet. Before starting the experiment, carefully read the documentation for each component and understand the electrical and mechanical requirements. Also, consider safety protocols and guidelines to ensure a safe working environment in the lab.
- Power Supply (optional): If the servo motor requires a power supply other than what the Arduino can provide, you'll need the appropriate power supply.
- Mounting Hardware (for the servo): If you want to mount the servo in a specific orientation or location, you might need screws, brackets, or other mounting hardware.

EXPERIMENTAL SETUP



METHODOLOGY

1. Hardware Setup

- Servo Motor Wiring:
- Connect the servo motor's power wire to the 5V pin on the Arduino, ground to a GND pin, and signal wire to a PWM pin (e.g., pin 9).
- Ensure a common ground connection between Arduino and the servo motor.
- RFID Reader Wiring:
- Connect the RFID reader to the computer via USB for power and communication, using the USB HID protocol.

2. Software Programming and Execution

- Arduino Code Development:

- Code was written in the Arduino IDE to initialize the servo on pin 9and set a default angle position.
- Listen for signals ("A" or "D") from Python to control the servo angle based on RFID card authentication.
- Load and upload the code to the Arduino

- Running the Python Script
  - To execute the script, ensure that any necessary libraries, such as pyusbl are installed.
  - In Python script, the RFID reader using its vendor and product IDs will be detected and reads the RFID card data and matches it to predefined, authorized card IDs.
  - A signal ('A' for access granted, 'D' for access denied) to the Arduino to adjust the servo position will be send.
  - A green LED if the card is authorized or a red LED if unauthorized will be activated.

- Running the Experiment
  - Place RFID cards near the reader and observe the system's response.
  - For authorized cards, the servo motor should adjust to a specific angle, and the green LED should illuminate.
  - For unauthorized cards, the red LED should illuminate, and the servo should reset to its default position.

- Additional Enhancements
  - Introduce visual indicators (LEDs) and JSON data handling in the code for structured data management.
  - Allow user-defined angle positioning for the servo through Python, providing more flexibility in the control mechanism.

RESULTS

```
#include <SPI.h>
#include <MFRC522.h>
#include <Servo.h>

// Pin definitions
#define RST_PIN         49          // Reset pin for RFID
#define SS_PIN          53          // SS/SDA pin for RFID
#define SERVO_PIN       3           // Servo control pin
#define GREEN_LED_PIN   4           // Green LED
#define RED_LED_PIN     5           // Red LED

// Instances
MFRC522 rfid(SS_PIN, RST_PIN);     // Create MFRC522 instance
Servo accessServo;                 // Create servo instance

// Array of authorized cards
byte authorizedCards[][4] = {
  {0x33, 0x36, 0x87, 0x1A} // Card for granted and tag for decline
(Not Here Easier to code then making another function)
};

const int NUM_CARDS = sizeof(authorizedCards) /
sizeof(authorizedCards[0]);

void setup() {
  Serial.begin(9600);
  SPI.begin();                      // Initialize SPI bus
  rfid.PCD_Init();                  // Initialize RFID reader

  // Initialize servo
  accessServo.attach(SERVO_PIN);
  accessServo.write(0);             // Set initial position

  // Initialize LEDs
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);
```

```
  Serial.println("System Ready");
}

void loop() {
  // Reset the loop if no new card is present
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
    return;

  // Check if card is authorized
  if (isAuthorizedCard(rfid.uid.uidByte)) {
    grantAccess();
  } else {
    denyAccess();
  }

  // Halt PICC and stop encryption
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}

bool isAuthorizedCard(byte *cardUID) {
  for (int i = 0; i < NUM_CARDS; i++) {
    if (compareUIDs(cardUID, authorizedCards[i])) {
      return true;
    }
  }
  return false;
}

bool compareUIDs(byte *uid1, byte *uid2) { //comparison algorithm
  for (int i = 0; i < 4; i++) {
    if (uid1[i] != uid2[i]) {
      return false;
    }
  }
  return true;
}
```

```
void grantAccess() {
  Serial.println("Access granted");
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(RED_LED_PIN, LOW);

  // Move servo to open position
  accessServo.write(90);
  delay(1000);                      // Keep open for 1 seconds

  // Return to closed position
  accessServo.write(0);
  digitalWrite(GREEN_LED_PIN, LOW);
}

void denyAccess() {
  Serial.println("Access denied");
  digitalWrite(RED_LED_PIN, HIGH);
  delay(1000);
  digitalWrite(RED_LED_PIN, LOW);
}
```

By using a RFID reader and servo motor we can control access based on card authorization. The MFRC522 library handles RFID operations, while Servo controls the servo motor, which serves as a gate or door lock. The system checks the card's unique identifier (UID) against a predefined authorized list.

In setup(), it initializes the RFID reader, servo, and LED pins for feedback (green for access granted, red for denied). When a card is detected in loop(), isAuthorizedCard() checks if the UID matches any authorized card UID stored in authorizedCards.

If authorized, grantAccess() is called, which turns on the green LED, moves the servo to open the "door" for 1 second, then resets it. If unauthorized, denyAccess() lights the red LED to indicate a rejection. The compareUIDs() function compares the detected UID with stored UIDs to validate access.

DISCUSSION

Two applications were shown in this experiment: RFID-based access control with the MFRC522 module and a servo motor and gesture detection using the MPU6050 sensor. The MPU6050 accelerometer and gyroscope could identify forward and backward tilt gestures by keeping an eye on acceleration thresholds. Video observations validate the system's potential for basic gesture-based controls by confirming that it consistently recognizes and shows these gestures on the serial monitor. Applications in robotics and other domains where simple movement-based commands could simplify interactions are promising.

The RFID-based access system reliably distinguished between authorized and unauthorized cards, activating the servo motor to open for authorized access and signaling denial with a red LED for unauthorized attempts. This performance confirms the system’s capacity for secure access control, making it viable for environments requiring restricted access, such as labs or offices. There were some differences between the expected and actual results. There were still a few instances where missed detections or false positives resulted from slight misalignments or ambient vibrations. These findings highlight how crucial it is to align and stabilize the MPU6050 to improve detection accuracy. Similar to this, irregular errors between the RFID card and reader resulted in missed reads even though the RFID system generally functioned as planned. These differences imply that more improvement is required for reliable performance, particularly in environments that are used frequently.

Gesture recognition may be impacted by sensor noise and power variations, and the fixed sensitivity limit may not be optimal in various settings. Reliability could be increased by modifying the MPU6050 calibration and threshold levels according to particular applications. The errors related to card orientation and interference with the reader were observed. Additionally, power fluctuations affected the servo motor’s reliability. Incorporating a more adaptable authorization system, such as a dynamic database for authorized cards, and adding feedback for the servo’s position would enhance scalability and responsiveness.

Overall, both parts of the experiment were functional but limited. Gesture detection was restricted to only two gestures and was sensitive to environmental shocks, which could result in unintended detections. The RFID system’s fixed list of authorized cards and lack of remote update options restricted its scalability, while the servo’s fixed open/close duration limited its adaptability. Overall, This experiment demonstrated the basic ideas of secure access control and gesture detection, which may find wider use in robotics and security systems with minor improvements.

CONCLUSION

It means that, with an MPU6050 sensor integrated with an Arduino board and a personal computer, one can develop sophisticated applications to read the data derived from motion and orientation. The given code in Arduino and Python will successfully capture and analyze accelerometer and gyroscope data, thus helping users in creating gesture recognition systems. This hands-on approach not only enhances understanding in serial communication but also opens up avenues for further exploration in the field of mechatronics, which will encourage users to extend their projects and provide more functionalities related to gesture recognition.

RECOMMENDATION

To enhance the smoothness and effectiveness of the experiments, students should discuss the interfacing experiment of the MPU6050 IMU, covering all the details necessary for serial communication between the PC and Arduino. Additionally, the student should know the procedure used for sensor calibration, the methods adopted to acquire data, and give an analysis of the accelerometer and gyroscope readings. The implementation of motion detection, plotting of real-time data, and sensor accuracy testing, using appropriate experimental results and graphs can show the clear data output.

For RFID-RC522 authentication system experiment. Students are able to further extend the servo motor control system by adding acceleration profiles for smoother movement. Adding a number of authentication states with LED indication will provide clear system feedback. The students can implement a Python multithreading approach for handling both RFID reading and servo control operations simultaneously. Several test runs with different RFID cards and servo movements will help optimize the system to have better accuracy and reliability. Consequently, this will make these suggestions help in conducting the experiment more efficiently and free of errors.

ACKNOWLEDGEMENTS

A special thanks goes out to Dr. Wahju Sediono and Dr. Zulkifli Bin Zainal Abidin, my teaching assistant, and my peers for their invaluable help and support in finishing this report. Their advice, feedback, and experience have greatly influenced the level of quality and understanding of this work. Their time, patience, and commitment to supporting my academic success are greatly appreciated.

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**.

Signature: 
Name: IRDINA NABIHAH BINTI MOHD NAZRI
Matric Number: 2214772

Read ☑
Understand ☑
Agree ☑

Signature: 
Name: KHALISAH AMANI BINTI YAHYA AZMI
Matric Number: 2218184

Read ☑
Understand ☑
Agree ☑

Signature: 
Name: LUQMAN AZFAR BIN AZMI
Matric Number: 2219857

Read ☑
Understand ☑
Agree ☑

Signature: 

Name: MOHAMAD NASRI BIN MOHAMAD NAZRI

Matric Number: 2219879

Read ☑
Understand ☑
Agree ☑

Signature: 

Name: MUHAMAS NURHAKIMIE THAQIF BIN ABDULLAH

Matric Number: 2213217

Read ☑
Understand ☑
Agree ☑