



MECHATRONICS SYSTEM INTEGRATION (MCTA 3203)

SECTION 2, SEMESTER 1, 2024/2025

ACTIVITY REPORT

Week: 4 (b)

Serial and USB interfacing with microcontroller and computer-based system (2):

Sensors and actuators.

No	Name (Group 6)	Matric No.
1	MUHAMMAD BASIL BIN ABDUL HAKIM	2215315
2	MUHAMMAD SYAFIQ BIN NOR AZMAN	2213187
3	MUHAMMAD HAFIZ HAMZAH BIN FANSURI	2212803
4	MUHAMMAD AMMAR ZUHAIR BIN NOR AZMAN SHAH	2110259
5	MUHAMMAD RAZIQ BIN KAHARUDDIN	2120225

Table of content

Table of content.....	2
Abstract.....	3
Material and equipment.....	4
Experimental Setup and Methodology.....	6
Results.....	7
Discussions.....	8
Recommendation.....	17
Conclusion.....	17
Student Declaration.....	18

Abstract

In a typical automatic gate system, RFID technology is frequently utilized to permit authorized vehicles or individuals to enter without the need for manual involvement. Every permitted individual or car is equipped with an RFID tag, typically located on a card or installed in the vehicle. As they get closer to the gate, an RFID reader located by the gate emits a signal which activates the RFID tag, causing it to transmit its specific ID.

If the system identifies the ID as valid, it will activate the gate to open without manual intervention. This arrangement offers a convenient, touchless method for managing entry, making it perfect for secure locations such as parking lots, gated communities, and workplaces. Due to the fact that RFID tags do not require direct visibility, the gate is able to identify them when they are within range, enabling a seamless and effective entry process.

Thus, in this project, we have explored how RFID works as a serial communication, especially in controlling systems.

Introduction

This project explores interfacing microcontrollers with computer-based systems via USB and serial connections, focusing on the integration of sensors and actuators. Specifically, it addresses the process of connecting a computer to an RFID card reader through USB, typically requiring USB Human Interface Device (HID) communication to interact effectively with the RFID device.

An experiment is designed to demonstrate RFID-based authentication and servo motor control using Python and Arduino, highlighting the necessary components and steps to achieve a functional setup.

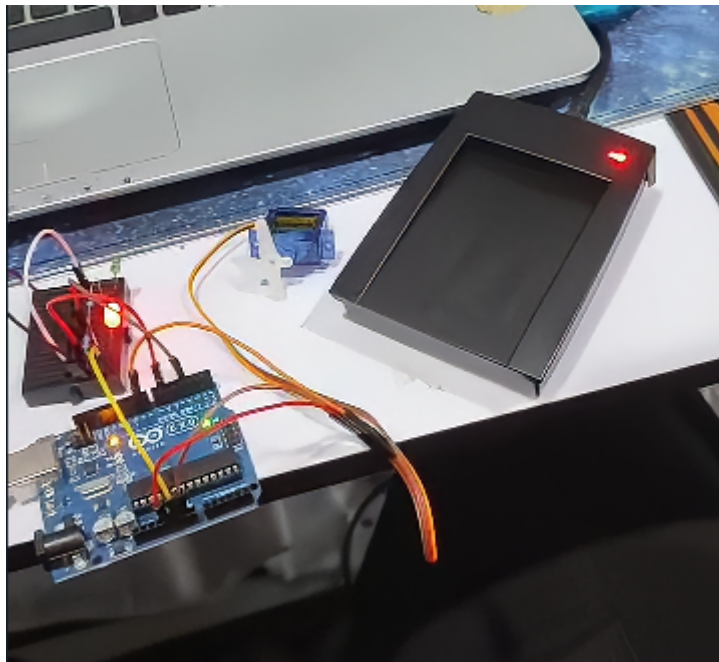
Material and equipment

1. 1, Arduino board
2. 1, RFID card reader with USB connectivity
3. 1, RFID tags or cards that can be used for authentication
4. 1, Servo Motor
5. Jumper wires
6. 1, Breadboard
7. 2, LEDs

Experimental Setup and Methodology

Here is how we set up the hardware:

1. We connected the servo's power wire (red colour) to the 5V output on the Arduino.
2. We connected the servo's ground wire (brown colour) to one of the ground (GND) pins on the Arduino.
3. We connected the servo's signal wire (orange colour) to one of the PWM pins on the Arduino (pin no. 9).
4. We ensured a common ground connection between the Arduino and the servo motor to complete the circuit.
5. The USB RFID reader was powered via the USB connection, so we didn't use any additional wiring for power.
6. Communication with the RFID reader was handled through the USB cable.



Here is how we set up the software incorporated into the hardware:

1. We wrote the Arduino code to control the servo using RFID
2. From the code, we assumed that when Python authorized a card, it sent the signal "A" to the Arduino to allow servo control, and when the card was denied, it sent the signal "D" to disallow control.
3. We made sure to adapt the code to our specific RFID reader and servo motor, including wiring and library dependencies.
4. We ran the Python script.
5. We saved the Python script with a .py extension.
6. We ran the Python script, and when we placed an RFID card near the reader, the card's UID was sent from the Arduino to the Python script, which then displayed it in the terminal.
7. With this setup, we could extend the Python script to perform actions or make decisions based on the RFID card data received.

Results

From this experiment, we managed to build a simple ID card authentication system by using an RFID card reader. The USB-connected RFID card reader is able to authenticate the ID card when being touched by the card reader. Then the Python script reads the card's UID and displays the authentication on the display.

Next, it will send a signal to the Arduino board to allow the servo and the colour of the LED to light on. If the card is authorised, the servo will move 90 degrees and the green LED will light up. If the card is unauthorised, the servo will not rotate and the red LED will light up.

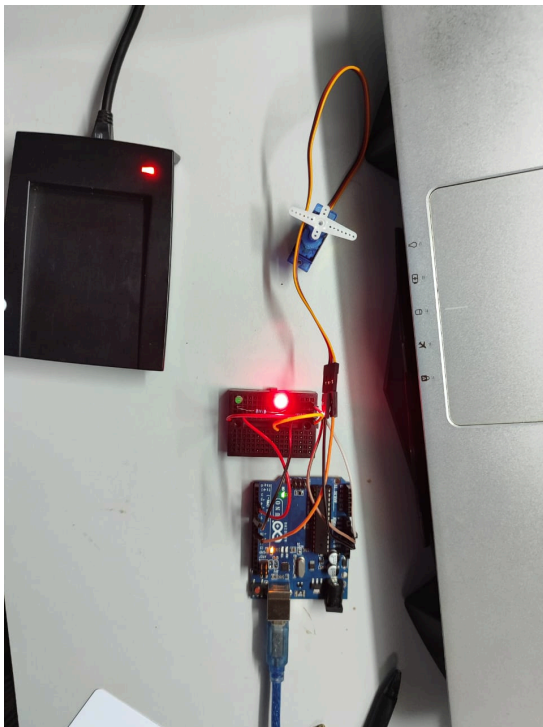


Figure 1: Red LED lights up when the card is unauthorised

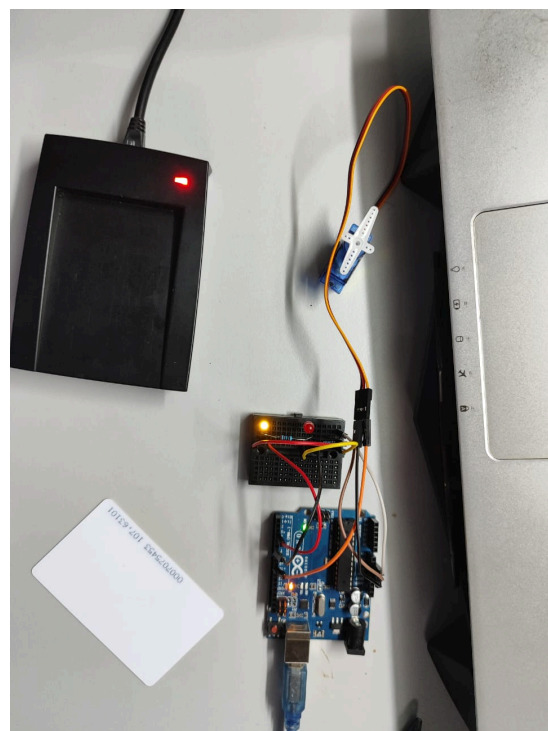


Figure 2: Green LED lights up when the card is authorised and the servo rotates 90 degrees

Discussions

Hardware Discussion

1. Breadboard

Breadboards are used to connect electronic components without the need of soldering.

2. Arduino board

Arduino, which comes in many forms, is a microcontroller that can be integrated into many electronics projects by utilising its pin and be controlled by programming

3. Wire/ Jumper cable male-to-male

Wires are used to connect electrical components.

4. RFID Reader:

Reads unique IDs from RFID cards and sends them to the Arduino, which checks if the ID is authorized.

5. LEDs (Green and Red):

Provide a visual indication of access status, with the green LED signalling access granted and the red LED signalling access denied.

6. Resistors

Protect the LEDs by limiting the current, ensuring safe operation

Software Discussion

ARDUINO CODE


```

#include <Servo.h>

const int servoPin = 9;

const int greenLEDPin = 3;

const int redLEDPin = 6;

Servo myServo;

void setup() {
  Serial.begin(9600);    // Begin serial
communication
  myServo.attach(servoPin); // Attach
servo to pin 9
  pinMode(greenLEDPin, OUTPUT);
  pinMode(redLEDPin, OUTPUT);
  Serial.println("Arduino ready.");
}

void loop() {
  if (Serial.available() > 0) {
    char accessStatus = Serial.read();

    delay(10); // Allow time for angle data
to arrive

```

```

    if (Serial.available() > 0) {
      int angle = Serial.parseInt();

      if (angle >= 0 && angle <= 180) { //
Ensure angle is within valid range
        if (accessStatus == 'G') {
          digitalWrite(greenLEDPin, HIGH);
          digitalWrite(redLEDPin, LOW);
          myServo.write(angle);
        } else if (accessStatus == 'D') {
          digitalWrite(greenLEDPin, LOW);
          digitalWrite(redLEDPin, HIGH);
          myServo.write(angle);
        } else if (accessStatus == 'I') {
          myServo.write(angle);
          digitalWrite(greenLEDPin, LOW);
          digitalWrite(redLEDPin, LOW);
        }
      }

      delay(2000); // Hold state for 2
seconds
      digitalWrite(greenLEDPin, LOW);
      digitalWrite(redLEDPin, LOW);
    } else {

```

```

        Serial.println("Invalid angle          }
received.");                                }
    }
}

```

1) **Library and Pin Setup:**

The code begins by including the Servo library, which enables control over servo motors. It also defines constants for the servo and LED pins, ensuring easy reference throughout the program.

2) **setup() Function:**

This function initializes the Arduino environment by starting serial communication at 9600 baud, attaching the servo to the specified pin, and setting the LED pins as outputs. It also sends a message to the Serial Monitor to confirm that the Arduino is ready to receive commands.

3) **loop() Function:**

The loop() function runs continuously, checking for incoming data from the Python program. It reads the access status character and an associated angle for the servo. Depending on the access status (G, D, or I), it controls the LEDs and moves the servo to the specified position. After processing the commands, it holds the current state for 2 seconds before resetting the LEDs.

PYTHON CODE

```

import serial

import time

import json

# Serial port settings

RFID_PORT = "COM6" # Update with
RFID COM port

ARDUINO_PORT = "COM4" # Update
with Arduino COM port

BAUD_RATE_RFID = 2400 # RFID
reader baud rate

BAUD_RATE_ARDUINO = 9600 #
Arduino baud rate

# Load configuration

def load_config():

    try:

        with open("config.json", "r") as file:

            return json.load(file)

    except (FileNotFoundError,
json.JSONDecodeError) as e:

        print(f"Error loading config: {e}")

        return None

```

```

# Initialize RFID and Arduino connections

try:

    # Set up RFID serial connection

    with serial.Serial(RFID_PORT,
BAUD_RATE_RFID, timeout=1) as
rfid_serial, \

        serial.Serial(ARDUINO_PORT,
BAUD_RATE_ARDUINO, timeout=1) as
arduino_serial:

        print(f"Connected to RFID on
{RFID_PORT}, baud rate
{BAUD_RATE_RFID}")

        print(f"Connected to Arduino on
{ARDUINO_PORT}, baud rate
{BAUD_RATE_ARDUINO}")

# Load config settings

config = load_config()

if not config:

    raise Exception("Failed to load
configuration.")

```

```

        authorized_cards =
        last_read_time = current_time

config["authorized_cards"]
        print(f"Card ID:

        servo_angle_granted =
        {card_data}")

config["servo_angle_granted"]

        servo_angle_denied =
        if card_data in

config["servo_angle_denied"]
        authorized_cards:

        servo_initial_position = 0
        print("Access granted.")

        servo_reset_delay = 3
        # Send a single command

        for granted access

        last_card_id = None

        last_read_time = 0
        arduino_serial.write(f"G,{servo_angle_gra

        debounce_time = 3
        nted}\n".encode())

# Main loop
        time.sleep(servo_reset_delay)

while True:
        # Reset servo to initial

        if rfid_serial.in_waiting > 0:
        position

        card_data =

rfid_serial.read(rfid_serial.in_waiting).hex
        arduino_serial.write(f"I,{servo_initial_pos

()
        ition}\n".encode())

        current_time = time.time()
        else:

        print("Access denied.")

        if (card_data != last_card_id or
        # Send a single command

(current_time - last_read_time) >=
        for denied access

debounce_time):

        last_card_id = card_data

```

```

except Exception as e:
    print(f"Error: {e}")
finally:
    print("Program terminated.")

time.sleep(0.5) # Short delay to
avoid continuous reads

```

1. Imports and Serial Port Setup:

The Python script imports necessary libraries (serial, time, and json) to handle serial communication, timing, and JSON file operations. It defines constants for the RFID and Arduino serial ports and their baud rates to facilitate communication.

2. Configuration Loading:

The `load_config()` function reads the `config.json` file, extracting the list of authorized card IDs, the servo angle for granted access, and the angle for denied access. This configuration enables the script to operate based on the current settings without needing hardcoded values.

3. Main Loop:

The main loop establishes serial connections with the RFID reader and Arduino, and then continuously checks for incoming card data. When a card is detected, it verifies whether it is different from the last read and if enough time has passed since the last read to prevent multiple detections. The script processes the card data, sending grant or deny commands to the Arduino based on whether the card is authorized, along with the appropriate servo angle.

Json file

```
{  
  "authorized_cards": ["f8", "f0", "ff"],  
  "servo_angle_granted": 90,  
  "servo_angle_denied": 0  
}
```

1) **authorized_cards Key:**

The `authorized_cards` key contains an array of strings, specifically `["f8", "f0", "ff"]`, representing valid RFID card IDs. The program checks this array against scanned card IDs to determine access rights. If a scanned card ID matches one of the entries in this array, it is granted access, triggering the appropriate actions in the system.

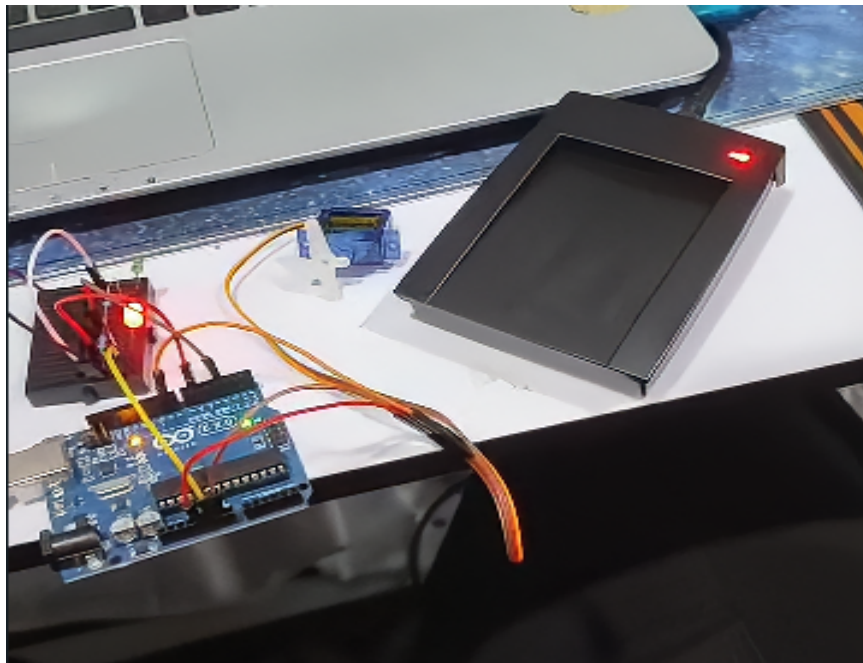
2) **servo_angle_granted Key:**

The `servo_angle_granted` key is a numerical value set to 90. This value indicates the angle to which the servo motor should move when access is granted. By sending this angle from the Python script to the Arduino, the system can control the servo's position, typically representing an open or unlocked state.

3) **servo_angle_denied Key:**

The `servo_angle_denied` key holds a numerical value of 0. This specifies the angle the servo should move to when access is denied. By sending this angle to the Arduino in response to an unauthorized card scan, the system moves the servo to a locked position, effectively denying access.

Electrical discussion



In our RFID-based access control experiment, we interconnected various components to create a cohesive system. The Green LED, connected to pin 3, serves as an indicator of granted access, lighting up when an authorized RFID card is scanned. In contrast, the Red LED, connected to pin 6, illuminates to signify denied access when an unauthorized card is detected. The Servo Motor is wired with its brown wire connected to the Ground (GND) for the negative connection, the red wire to the 5V pin for power, and the orange wire to pin 9, allowing the Arduino to send control signals to move the servo to specific angles based on access status.

Additionally, the RFID Reader is connected through COM4 on our computer, facilitating serial communication with the Arduino to transmit data regarding scanned RFID cards. This setup enables the Arduino to determine whether access should be granted or denied effectively. Together, these connections allow for a user-friendly experience, providing visual

feedback through the LEDs and physical control via the servo motor, ensuring that authorized users can easily gain access while unauthorized attempts are clearly indicated.

Recommendation

This experiment can further improve its security by encrypting the UID data stored. Simple UID recognition may be susceptible to spoofing. Incorporating two-factor authentication can improve the security system.

Other than that, in order to account for differences in hand gestures between users, calibration routines or adaptive learning methods can be implemented. By customizing the system's gesture recognition features, user experience and accuracy can be enhanced, making interactions more reliable and intuitive for a wider range of users.

Another enhancement that can be done for security is to introduce more sensors such as camera modules or biometric sensors to be validated alongside the RFID card. This approach can be useful for high-security applications where multi-factor authentication is required.

The usage of JSON files allows for more flexibility and scalability as the file is easily readable and modifiable, making it easier to be configured in the future.


Conclusion


To summarise, the MPU6050 sensor, along with Arduino and Python scripts, provides a simple approach to recognizing hand gestures. By using accelerometer and gyroscope data, the system detects specific gestures based on set threshold values. This setup enables real-time tracking and reaction to hand movements. Although the current system is basic, it has the potential for expansion to support more complex tasks and applications. Gesture recognition technology shows promise for enhancing user interaction and accessibility in various fields.


Student Declaration

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 taken or done by unspecified sources or person. 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 have been verified by us.

NAME: Muhammad Basil bin Abdul Hakim	READ	✓
MATRIC NO: 2215315	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: Muhammad Syafiq Bin Nor Azman	READ	✓
MATRIC NO: 2213187	UNDERSTAND	✓
SIGNATURE 	AGREE	✓

NAME: Muhammad Hafiz Hamzah Bin Fansuri	READ	✓
MATRIC NO: 2212803	UNDERSTAND	✓
SIGNATURE : 	AGREE	✓

NAME: Muhammad Ammar Zuhair Bon Nor Azman Shah	READ	✓
---	------	---

MATRIC NO: 2110259	UNDERSTAND	✓
SIGNATURE: 	AGREE	✓

NAME: MUHAMMAD RAZIQ BIN KAHARUDDIN	READ	✓
MATRIC NO: 2120225	UNDERSTAND	✓
SIGNATURE	AGREE	✓