Mechatronics System Integration (MCTA3203)

Week **4b**: Serial and USB interfacing with microcontroller and computer based system (2): Sensors and actuators.

To link your computer to the RFID card reader through a USB cable, a distinct method is required for your Python code to engage with it. Most USB-connected RFID card readers act as USB Human Interface Devices (HID), and you'll need a library or module to handle USB HID communication. To set up a lab experiment that involves RFID card authentication and controlling a servo motor with Python and Arduino, you will need a variety of materials.

Materials Needed:

- 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

Hardware Setup:

- Servo Motor Wiring:

```
  Arduino                 Servo Motor
+-----------+           +----------------+
|           |           |                |
|    5V     |-----------| Power (Red)    |
|           |           |                |
|    GND    |-----------| Ground (Brown) |
|           |           |                |
|    D9     |-----------| Signal (Yellow)|
|           |           |                |
+-----------+           +----------------+
```

  - Connect the servo's power wire (usually red) to the 5V output on the Arduino.

    - Connect the servo's ground wire (usually brown or black) to one of the ground (GND) pins on the Arduino.
    - Connect the servo's signal wire (usually orange or yellow) to one of the PWM pins on the Arduino (e.g., pin 9).
    - Ensure that you have a common ground connection between the Arduino and the servo motor to complete the circuit.

- As for the USB RFID reader, it's usually powered via the USB connection, and you don't need any additional wiring for power. The communication with the RFID reader is handled through the USB cable.

Arduino Code:

- Here's a *simplified example* of Arduino code for servo control:

```
#include <Servo.h>

Servo servo;
int servoPosition = 90;  // Initial servo position

void setup() {
  servo.attach(9);  // Attach the servo to pin 9
  servo.write(servoPosition);
  Serial.begin(9600);
}

void loop() {
  // Listen for signals from Python
  if (Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'A') {
      // Allow servo control
      servoPosition = 180;  // Set the servo to a specific angle
      servo.write(servoPosition);
    } else if (command == 'D') {
      // Disallow servo control
      servoPosition = 90;  // Set the servo back to the default angle
      servo.write(servoPosition);
    }
  }
}
```

- In this example, when Python authorizes a card, it sends the signal "A" to the Arduino to allow servo control, and when the card is denied, it sends the signal "D" to disallow control. Make sure you adapt the code to your specific RFID reader and servo motor, including wiring and library dependencies.

Python Script:

- One common library for USB HID devices in Python is *pyusb*. You can install the *pyusb* library using *pip*, if you haven’t it:

```
pip install pyusb
```

- Find your RFID reader's vendor ID and product ID. You can use tools like *lsusb* on Linux or *Device Manager* on Windows to get this information. Here's a Python code *example* using *pyusb*:

```
import usb.core
import usb.util

# Define your RFID reader's vendor and product IDs
vendor_id = 0x1234  # Replace with your RFID reader's vendor ID
product_id = 0x5678  # Replace with your RFID reader's product ID

# Authorized card IDs
authorized_cards = ["YourCardID1", "YourCardID2"]

# Initialize the USB RFID reader
dev = usb.core.find(idVendor=vendor_id, idProduct=product_id)
if dev is None:
    raise ValueError("RFID reader not found")

# Detach kernel driver if it's attached
if dev.is_kernel_driver_active(0):
    dev.detach_kernel_driver(0)

# Set the configuration of the device
dev.set_configuration()

# Define the endpoint
endpoint = dev[0][(0, 0)][0]

while True:
    try:
        data = dev.read(endpoint.bEndpointAddress,
endpoint.wMaxPacketSize)
        card_id = ''.join([chr(byte) for byte in data])

        if card_id in authorized_cards:
            print("Access granted. You can now control the servo.")
            # Send the 'A' signal to the Arduino
            ser.write(b'A')
        else:
            print("Access denied. Unauthorized card.")
            ser.write(b'D')
    except usb.core.USBError:
        pass
```

Run the Python Script:

- Save the above Python script with a *.py* extension.
- Run the Python script, and when you place an RFID card near the reader, the card's UID will be sent from the Arduino to the Python script, which will then display it in the terminal.

With this setup, you can extend your Python script to perform actions or decisions based on the RFID card data you receive.

Task

Enhance the existing code to introduce a visual indicator, such as illuminating a green LED, when a recognized UID is detected by the RFID reader, and conversely, activate a red LED when an unrecognized card is read. Incorporate structured JSON data handling within your code for better organization and flexibility. Add some options for the user to freely set the angle position of the servo.