Week 7 Lab Activities:
Interfacing with the Computer

Preparation:

1. Reading:
    a. Review: “Connecting Arduino to Processing”
    https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing
    b. Review: “Asynchronous Serial Communication: The Basics”
    https://itp.nyu.edu/physcomp/lessons/serial-communication-the-basics/
    c. Review: “Arduino Servo Motors”
    https://www.instructables.com/Arduino-Servo-Motors/
2. Install processing 3.0.1+

Exercises:

What are we going to use in this workshop?

- **Solderless breadboard**
  !!! When you start to put components on your breadboard, avoid adding, removing, or changing components on a breadboard whenever the board is powered. You risk shocking yourself and damaging your components.
- **Arduino board**
- **USB cable**
- **LEDs**
- **Wires**
- **Resistors**
- **Servo motor**

Circuit 1: Dimmer

Connect an LED to digital pin 9 of the Arduino board. Use a 390 Ohm resistor in series with the LED to limit the current flowing through the LED.

Task: Build the circuit as described above.
Task: Write code that allows you to dim the LED by sending data over the serial port.

Circuit 2: Servo motors

Servomotors is an actuator that produces movement. It works well for precise movement and can be positioned from 0-180 degree, or they can have 360-degree rotation. They have an easy to use three-wire **PWM** 5V interface. They can be controlled by pulse (1000ms for 0 degrees to 2000ms for 180 degree) or by angle between 0 and 180 degrees (followed by a non-digit character). The servo library is included in the Arduino IDE. See code examples below. The Arduino uno supports up to 12 servo motors.

Task: Build a circuit in which a servo motor and a potentiometer are connected to the Arduino.
Task: Write the code that allows you to move the servo motor with the potentiometer. Look at the example code.

Circuit 3: Servo motor – reading data from the serial monitor

We are going to create a circuit in which we can control the servo motor with keyboard commands over the serial monitor. Review the code example below. Try to understand what is happening before moving on to the tasks.

Task: Connect the servo motor to the Arduino board and control the motor.
Answer: Why do we subtract '0' from command?

Circuit 4: Serial Dimmer – Processing and Arduino

Demonstrates the sending of data from the computer to the Arduino board, in this case to control the brightness of an LED. The data corresponds to the current position of the mouse along the X axis of the processing screen. This data is sent in individual bytes, each of which ranges from 0 to 255. Arduino reads these bytes and uses them to set the brightness of the LED. A single LED is connected to digital pin 9 of Arduino to the ground through a 390-Ohm resistor.

Task: Build the circuit as described above.
Task: Start with writing a sender in processing:

1. Import the serial communication library
2. Use the write() function
3. Use a width of 256 so you can use mouseX as the data being send, otherwise you need to map the mouseX value to a value in the [0,255] range.
4. Connect your Arduino board to the PC before running your serial write code in processing, otherwise it will throw an IndexOutOfBoundsExcption.

Task: Write the Arduino codes that receives the data and controls the LED.

Circuit 5: Blink - Reading data from the serial monitor

Connect a yellow LED to pin 12 and a red LED to pin 11 of the Arduino board. Connect each of them to a 390 Ohm resistor. In this exercise, Arduino will make use of data coming from the serial port. The data will be typed in the Serial monitor in the Arduino IDE. Your code will read the value of the data. If the data is between 0 and 9, then the yellow LED should blink the same number of times as the value that is read from the serial port. For example, if 8 is typed, the yellow LED should blink 8 times. Also, display a message each time the LED blinks (for example, the word 'blink'). If any other character is typed, the red LED should turn ON and blink until a value between 0 and 9 is typed again.

Task: Build the circuit as described above.
Task: Write the code as described above. Look at the code of circuit 3 for guidance.

Servo motors

1. Include library:

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

2. Create servo object to control a servo:

   ```
   Servo myservo;
   ```

3. Attach the servo on pin 9 to the servo object:

   ```
   void setup() {
     myservo.attach(9);
   }
   ```

4. Set the servo position

   ```
   myservo.write(val);
   ```

Servo motors: Example code

```
int command = 0;
int pulseWidth = 1500;

// this will be the center that the servo will go to if
// no commands are sent from the keyboard
int centerServo = 1500;
int servoPin = 9;

void setup(){
        pinMode(servoPin, OUTPUT);
        Serial.begin(9600);
        pulseWidth = centerServo; //Give the servo a starting point (or it floats)
}

void loop(){
        if(Serial.available()){
                command = Serial.read(); // get the keyboard input
                // let's assume the user or another control
                // is sending a number between 1 and 9 to rotate the servo
                if(command > '0' && command <= '9') {
                        command = command - '0';
                        // now turn that number into a pulsewidth microsecond value
                        // you might need to change this based on the servo that you're using
                        pulseWidth = (command * 100) + 1000;
                        sendPulse();
                }
        } else {
        // if no data is coming then just send the servo to the last position
        pulseWidth = (command * 100) + 1000; // you could also put it to the center, right,
        sendPulse();
        }
}

void sendPulse(){
        digitalWrite(servoPin, HIGH);
        delayMicroseconds(pulseWidth);
        digitalWrite(servoPin, LOW);
        // make sure the send pulse isn't getting called too
        // soon after the last time
        delay(20);
}
```