ELEC1601 Week 5 Lab Report

SID: 500425334

Lab GROUP: M18_08B

Online student: Yes

Introduction:

The propose of this lab session was

1. learning how to use the accelerometer and servo.
2. learning how to use a library.
3. learning what is PWM.

What I have done:

1. Using the accelerometer to output the acceleration in three axes.
2. Control the servo in two different way, `Servo.write()` and `digitalWrite()`.
3. Distinguish the difference between `analogWrite()` and `Servo.write`.
4. Understand how PWM works.

Material:

Component list

Task	Component	Quantity
1, 2, 3	Arduino Uno 3	1
1, 3	1 $k\Omega$ Resistor	1
1, 3	Accelerometer	1
2, 3	Micro Servo	1

introduction of Accelerometer

Brief introduction of Accelerometer

- A device consists of polysilicon, the change in distance of a movable beam and two fixed beams is a measure of **acceleration**.
- Take 3.3V as input, the analog signal of x, y, z as output.
- **Example of real world**: Aircraft's orientation, fall detect device

Brief introduction of Accelerometer's pins

Pin	Usage
XOUT, YOUT, ZOUT	Output the acceleration in x, y, z axis as analog signal.
G-select	Select the sensitive of accelerometer, 1.5g or 6g. When this pin is unconnected, select 1.5g as default.
Sleep	When HIGH is input, work in normal mode, when LOW is input, work in sleep mode.

introduction of Servo

Brief introduction of Servo

- A device allows for precise control of angular, work with the library `Servo.h`.
- Take 5V as input, rotate the Servo's shaft by a certain angular according to the code.
- **Example of real world**: Robot, Conveyor belt

Brief introduction of Servo's pins

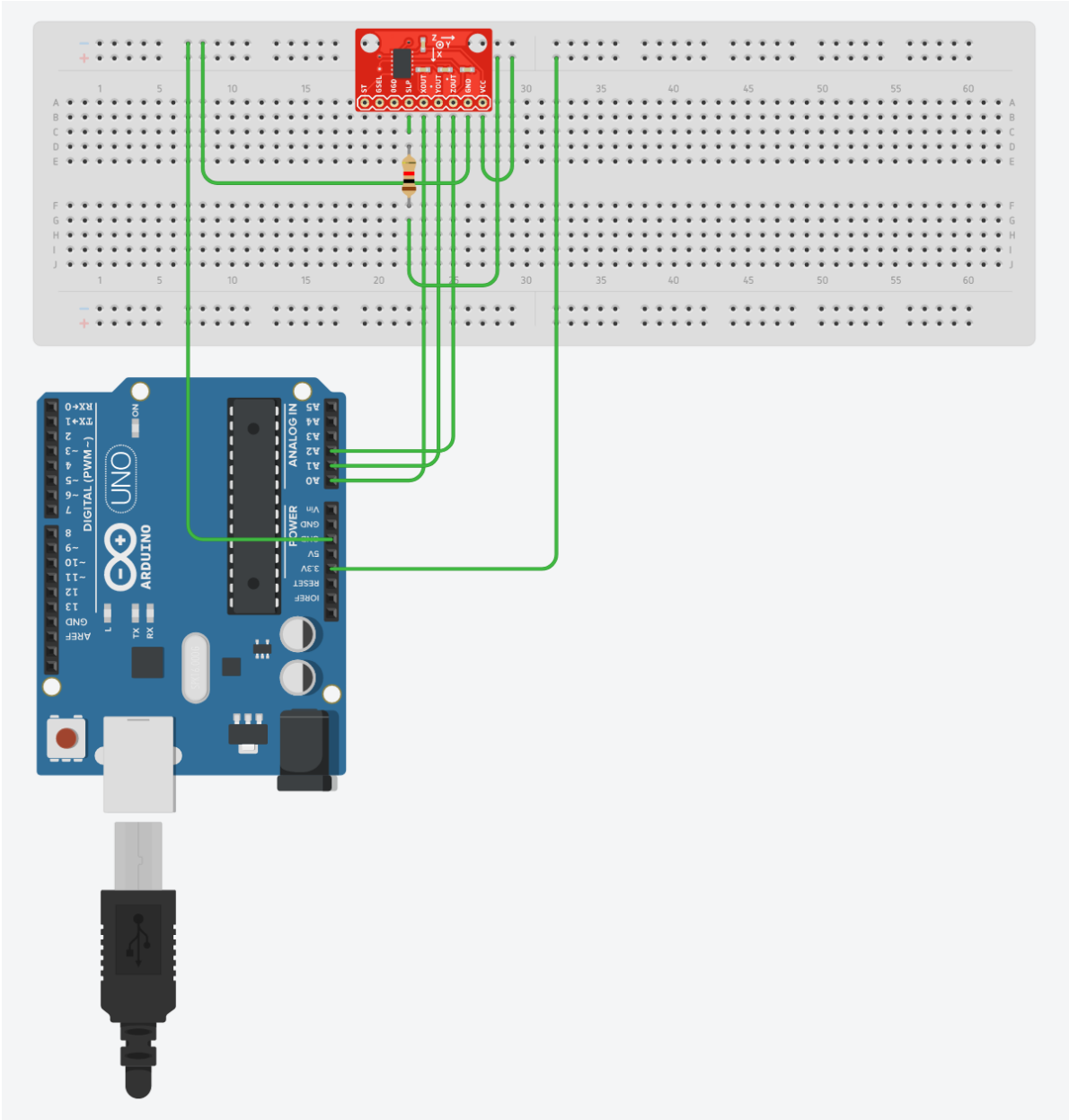
GND	5V	PWM signal
BLACK	RED	WHITE
BROWN	ORANGE	YELLOW
BLACK	RED	BLUE

Brief introduction of `Servo.h`

Function name	Parameters	Description
attach()	pin: int, pin 9 or pin 10	Attach the Servo variable to a pin.
write()	angle: int, from 0 to 180	Controlling the shaft according to the angle directly.

TASK 1: Accelerometer

Connection:



- According to the data sheet:

  > By placing a high input signal on pin 7, the device will resume to normal mode of operation.

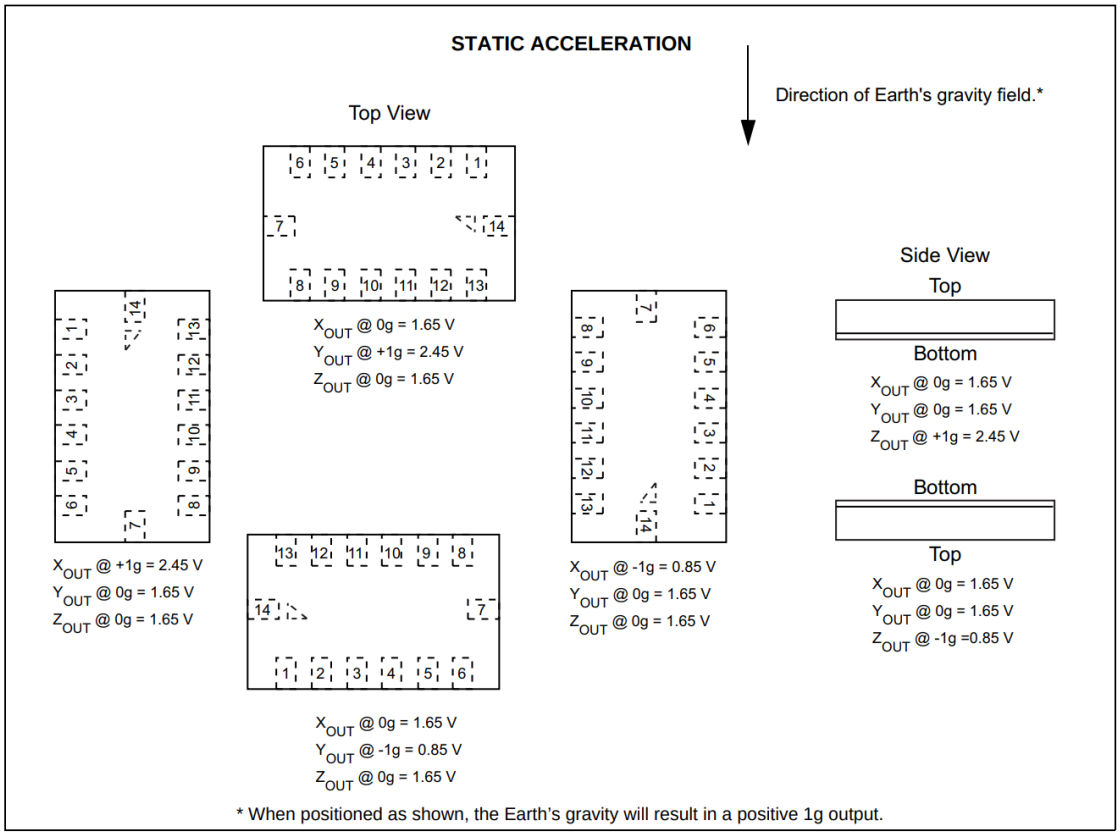
  the input of pin SLP is HIGH, so the accelerometer is working under the normal mode.

Pseudo-code:

```
1. Set ANALOG IN pin A0, A1, A2 to XOUT, YOUT, ZOUT respectively.
LOOP:
2. Read values from XOUT, YOUT, ZOUT.
3. Convert each analog signal into voltage with [VOLTAGE = OUT / 1023.0 *5000] in
mV.
4. Adjust the value of voltage with the data sheet, determine the value of the
offset.
5. Convert the voltage into g-scale with [G-SCALE = VOLTAGE / 800].
```

Explanation of pseudo-code:

1. The outputs of the accelerometer is a set of analog signals, which range from 0 to 1023, so use ANALOG PIN to read values.
2. The range of value is corresponding to the voltage, from 0V to 5V, no matter which power pin is used. Because function `analogRead()` is built into Arduino board, the reference is always 5V.
3. As shown by the data sheet, when placing the accelerometer in a certain way, the output voltages are fixed.



For example, when placing the accelerometer horizontally, the voltage of X and Y both should be 1.65 V, the voltage of Z should be 2.45 V.

4. As shown by the connection graph, the GSEL pin is unconnected, indicated by the data sheet,

> The g-Select pin can be left unconnected for applications requiring only a 1.5 g sensitivity as the device has an internal pull-down to keep it at that sensitivity (800 mV/g)).

So the relationship between g and voltage is `1g = 800mV`

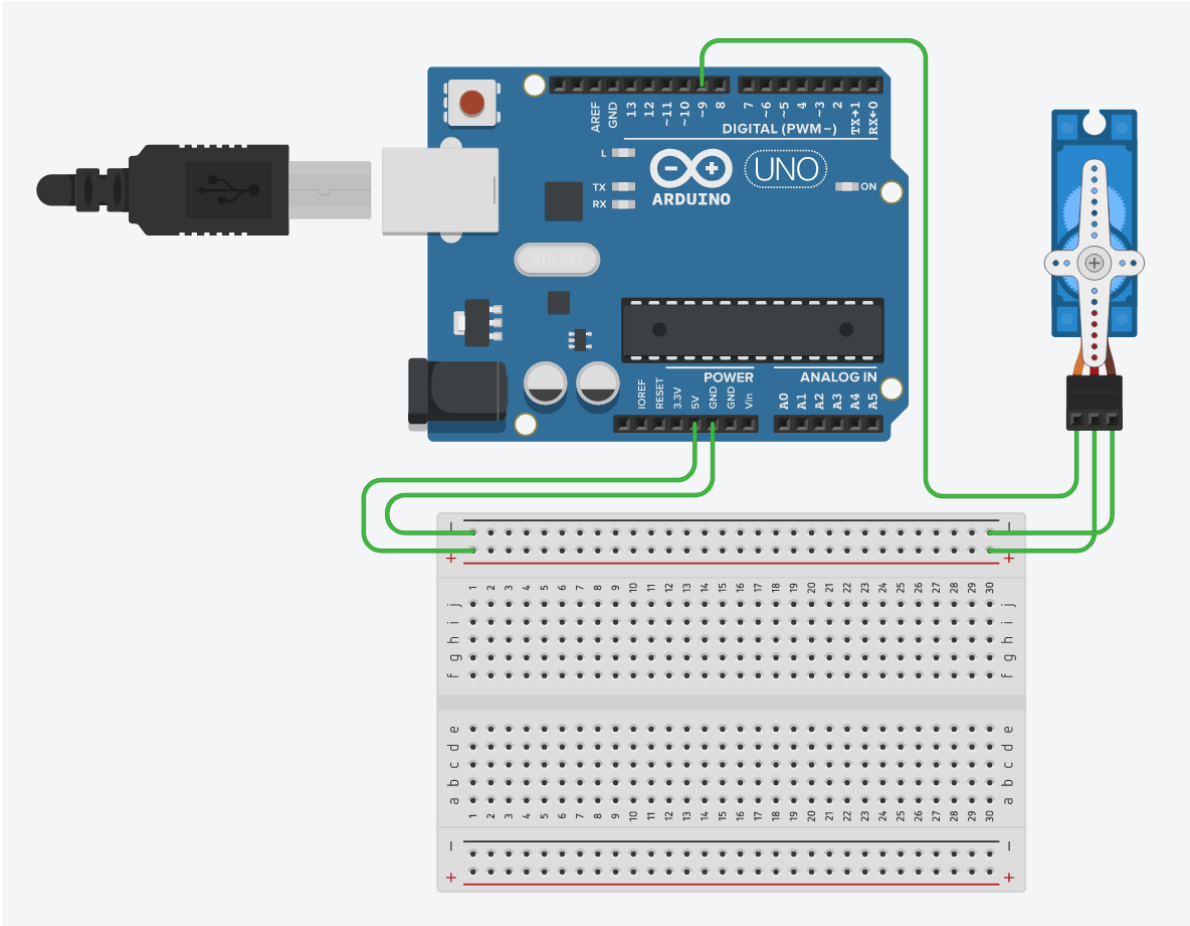
Expected result:

- When placing the accelerometer horizontally, the output should be

```
X_gscale:2.06 Y_gscale:2.06 Z_gscale:3.06
```

TASK 2: Servo motor

Connection:



Pseudo-code:

```
1. Include the servo library.
2. Instantiate a Servo Object.
3. Attach the Servo to Pin 9 using function attach().
LOOP:
4. when the Servo's shaft hasn't rotated to 150 degrees, increase 30 degrees
using function write(), delay 1000ms.
5. when the Servo's shaft already reach 150 degrees but hasn't rotated to 0
degrees, decrease 30 degrees using function write(), delay 1000ms.
```

Explanation of pseudo-code:

1. The `servo.h` allows the Arduino board to control a servo.
2. The `servo.h` only support using pin 9 or pin 10, when `servo.h` is included, `analogWrite()` couldn't be used in pin 9 or pin 10, because the timer 1 is occupied.
3. This task requires the Servo's shaft rotate +30 degree per second when the Servo hasn't reach 150 degrees, so after rotation, delay 30 degrees.
4. This task requires the Servo's shaft rotate -30 degree per second after the Servo reach 150 degrees, so after rotation, delay 30 degrees.

Expected result:

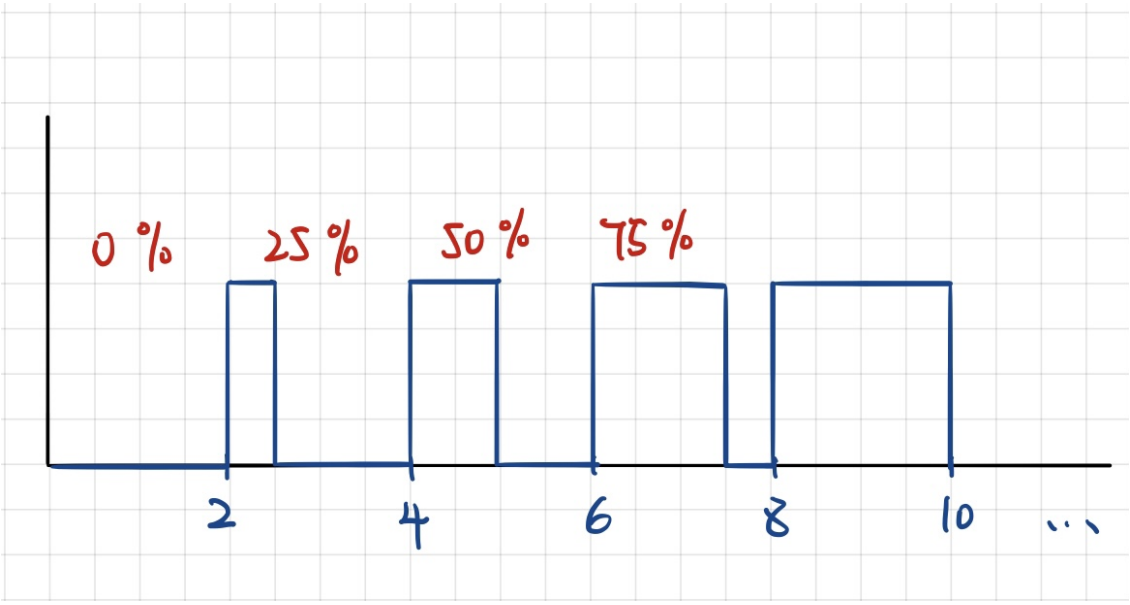
- rotate 30 degrees until reaching 150 degrees, decease to 0 degrees by the same step.

About `Servo.h`

The third question of this task asks to use `analogWrite()` instead to complete this task, however, this function can power the Servo, but can't control the Servo as expect, the reasons are as follows.

1. As shown on the official website (website: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/), the frequency of PWM output of `analogWrite` is 490 Hz (980 HZ in pin 5 and pin 6), which means every 2 ms ($interval(ms) = \frac{1000ms}{frequency}$), an analog signal is outputted, and the analog signal from 0 V to 5 V is mapping to digit from 0 to 255.

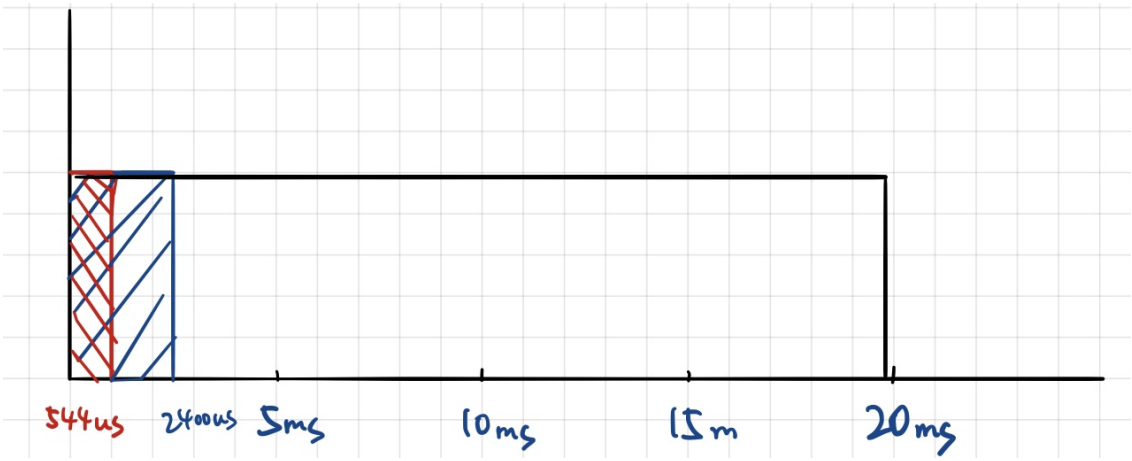
   In each 2 ms, follow formula is applied, $PWN\ duty\ cycle = \frac{high\ signal\ time}{2ms} * 100\%$, digit 0 represents 0%, digit 255 represents 100%.

2. As shown in the source code of `Servo.h` (website: https://github.com/arduino-libraries/Servo/blob/master/src/Servo.h)

   ```
   #define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
   #define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
   #define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is
   attached
   #define REFRESH_INTERVAL    20000     // minumim time to refresh servos in
   microseconds
   ```

   which means each 20 ms ($1ms = 1000us$), the Servos refresh, 20 ms can be converted to 50 Hz, and in each 20 ms, a high-level output of 544 us is corresponding to 0 degrees, a high-level output of 2400 us is corresponding to 180 degrees, the HIGH signal has to be continuous, verification as the third point.



3. Because the frequency of `analogWrite()` and `Servo.write()` are different, so it is impossible to instead `Servo.write()` by `analogWrite()`, however, use `digitalWrite()` is feasible.

```
int pinOut = 9;
void simulateServoWrite(int angle){
  int pulseWidth = map(angle, 0, 180, 544, 2400);
  digitalWrite(pinOut,HIGH);
  delayMicroseconds(pulseWidth);
  digitalWrite(pinOut,LOW);
  delayMicroseconds(20000-pulseWidth);
}
void setup()
{
  pinMode(9, OUTPUT);
}

void loop()
{
  for(int angle = 0; angle < 150; angle += 30){
    for(int i = 0; i < 100; i++){
      simulateServoWrite(angle);
    }
    delay(1000);
  }
  for(int angle = 150; angle > 0; angle -= 30){
    for(int i = 0; i < 100; i++){
      simulateServoWrite(angle);
    }
    delay(1000);
  }
}
```

Explanation of the code:

1. `SimulateServoWrite`:
    1. In `Servo.h`, every 20 ms, a high-level output of 544 us will set the servo angle to 0 degrees, a high-level output of 2400 us will set the servo angle to 180 degrees. Using `map()` function to convert the degree to microseconds.
    2. The PWM output interval is 20 ms, using the high-level output to rotate the Servo's shaft, the rest of the time should be set to low-level output.
2. `setup`:

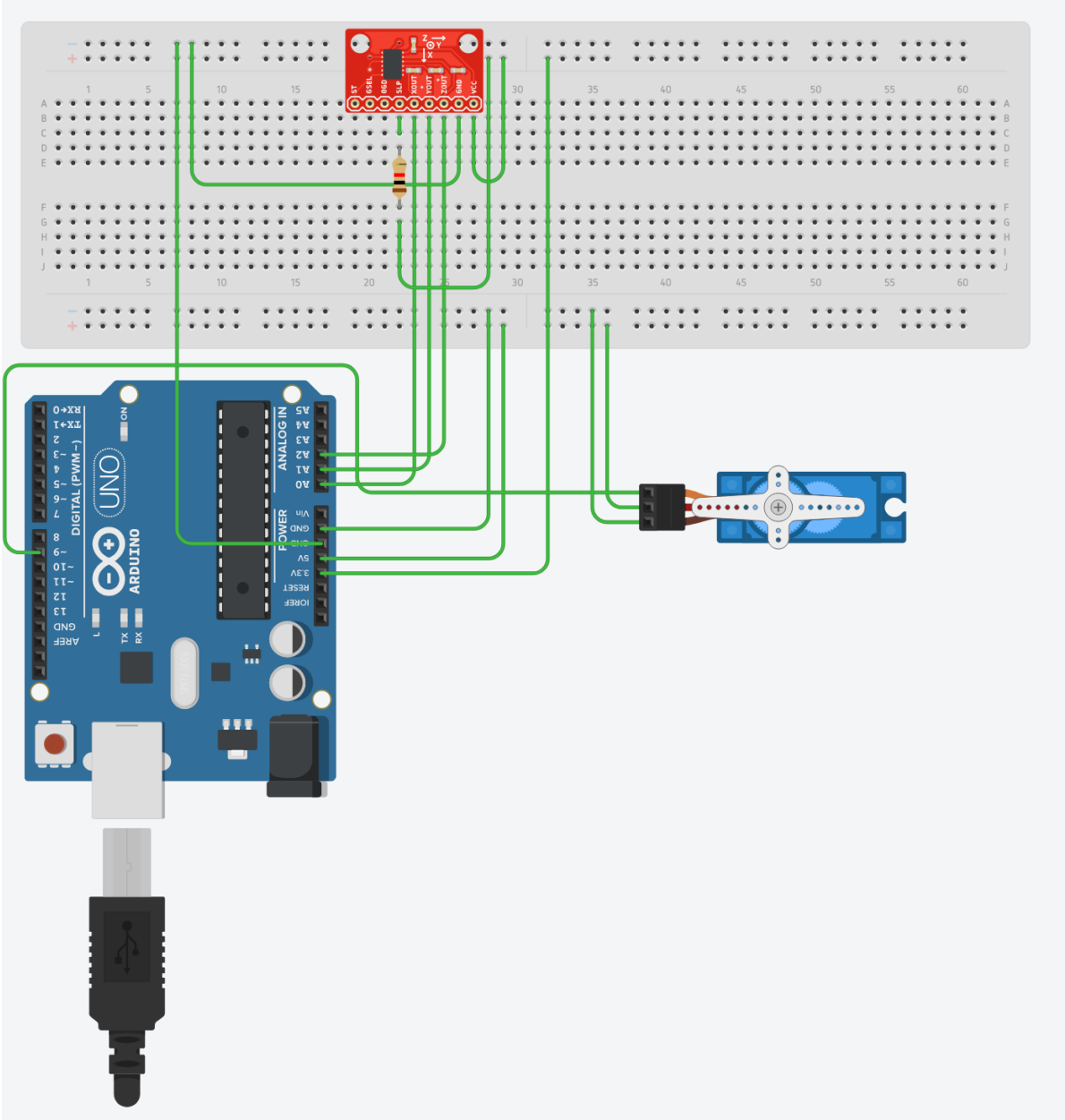
Set pin 9 mode as output.

3. `loop`:

   The logic of `loop` is the same as previous task 2 pseudo-code, calling 100 times `simulateServoWrite` function is to make sure the Servo have enough time to rotate, the maximum speed of this servo is 0.10 sec/60 degrees.

Task3: Control a Servo with an Accelerometer

Connection:



pseudo-code:

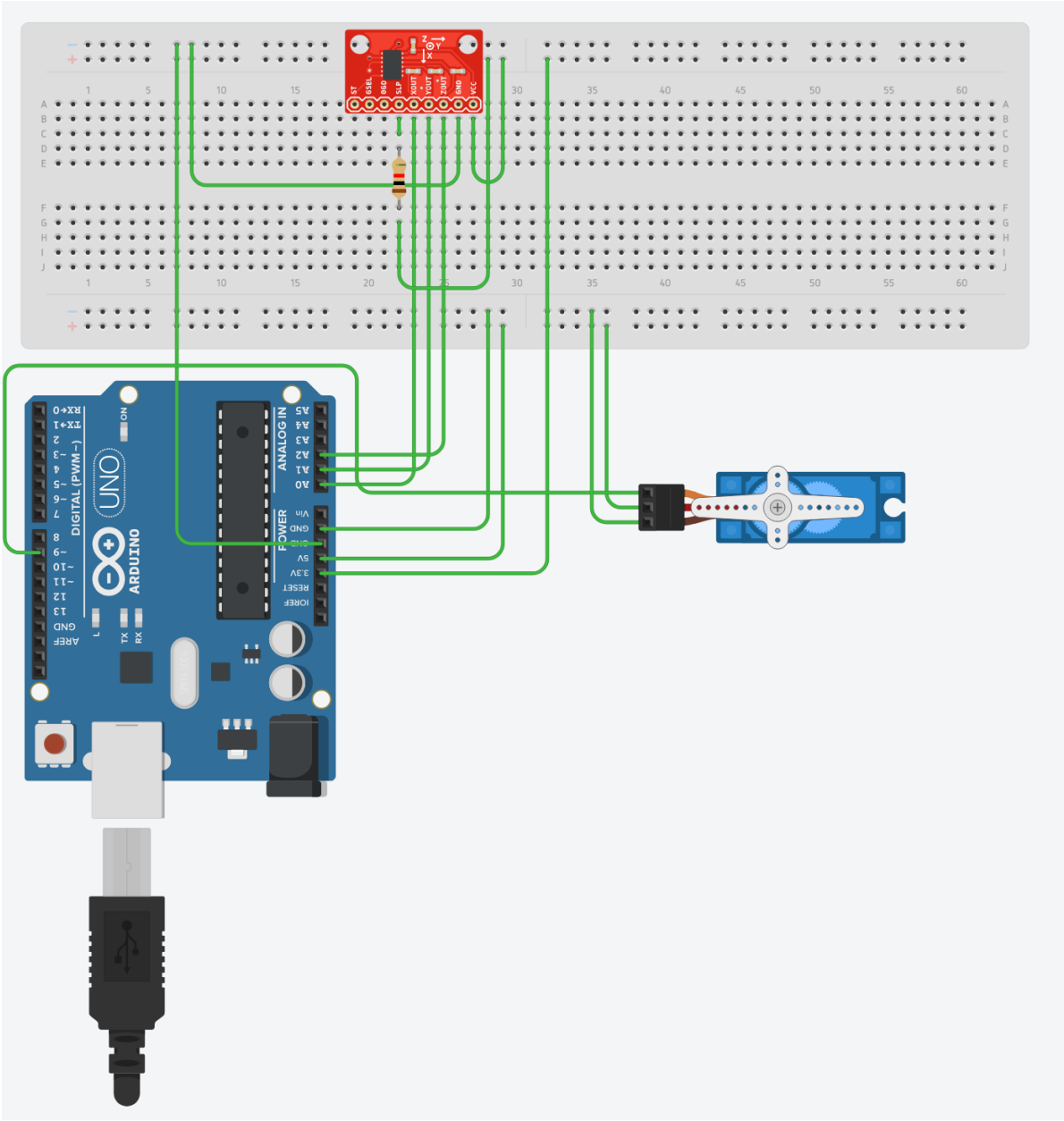
```
1. Include the servo library, the math library.
2. Set ANALOG IN pin A0, A1, A2 to XOUT, YOUT, ZOUT respectively.
3. Instantiate a Servo Object.
SETUP:
4. Attach the Servo to Pin 9 using function attach().
LOOP:
5. Read values from XOUT, YOUT, ZOUT.
6. Convert each analog signal into voltage with [VOLTAGE = OUT / 1023.0 *5000] in
mV.
7. Adjust the value of voltage with the data sheet, determine the value of the
offset.
8. Convert the voltage into g-scale with [G-SCALE = VOLTAGE / 800].
9. Compare these g-scales to the g-scales when the accelerometer is balanced
given by datasheet.
10. if the axes change by 10%, rotate the Servo's shaft to 0 degrees, the
original position, otherwise, it will rotate by 2 degrees.
11. Delay 100 ms.
```

Explanation of pseudo-code:

1. Point 1, 2, 3, 4, 5, 6, 7, 8 are already explained in previous tasks.
2. When the accelerometer is balanced, the output voltage of three axis is given by data sheet, convert it into g-scale.
3. Compare the output in g-scale with balanced state g-scale using `fabs(Balanced g-scale - output g-scale)/Balanced g-scale`.
4. Read outputs of accelerometer every 0.1 second, so delay 100 ms.

Relation to real-world electronics: Auto door

Connection:




```
1. Include the servo library, the math library.
2. Set ANALOG IN pin A0 to ZOUT respectively.
3. Instantiate a Servo Object.
SETUP:
4. Attach the Servo to Pin 9 using function attach().
LOOP:
5. Read values from ZOUT.
6. Convert each analog signal into voltage with [VOLTAGE = OUT / 1023.0 *5000] in
mV.
7. Adjust the value of voltage with the data sheet, determine the value of the
offset.
8. Convert the voltage into g-scale with [G-SCALE = VOLTAGE / 800].
9. Compare this g-scale to the g-scale when the accelerometer is balanced given
by datasheet.
10. if the axes changes, rotate the Servo's shaft to 90 degrees, otherwise, it
will rotate 10 degrees per second until it reaches 0 degrees.
11. Delay 10000 ms.
```

Explanation of pseudo-code:

1. Point 1 to 9 are the same as task 3.
2. When someone stands on the blocks, the ZOUT will change, then the door will open.
3. After 10 seconds, the door will close slowly, so as not to hurt people.

Limitation:

1. No matter who is in front of the door, the door will open.
2. If someone still standing on the blocks after the door open, the door will close after 10 second.

Reference

- MMA7361L document: https://www.nxp.com/docs/en/data-sheet/MMA7361L.pdf
- Sparkfun website of Servo: https://www.sparkfun.com/products/9065
- Arduino Servo API: https://www.arduino.cc/reference/en/libraries/servo/
- Arduino Servo Source code: https://github.com/arduino-libraries/Servo/blob/master/src/Servo.h and https://github.com/arduino-libraries/Servo/blob/master/src/avr/Servo.cpp
- Arduino Map function: https://www.arduino.cc/reference/en/language/functions/math/map/