BRAC UNIVERSITY

Inspiring Excellence

Department of Computer Science and Engineering

Course Code: CSE461	Credits: 1.5
Course Name: Introduction to Robotics Lab	Semester: Fall '22

Lab 2

Introducing servo motor using along with push buttons and LEDs with Raspberry Pi

I. Topic Overview:

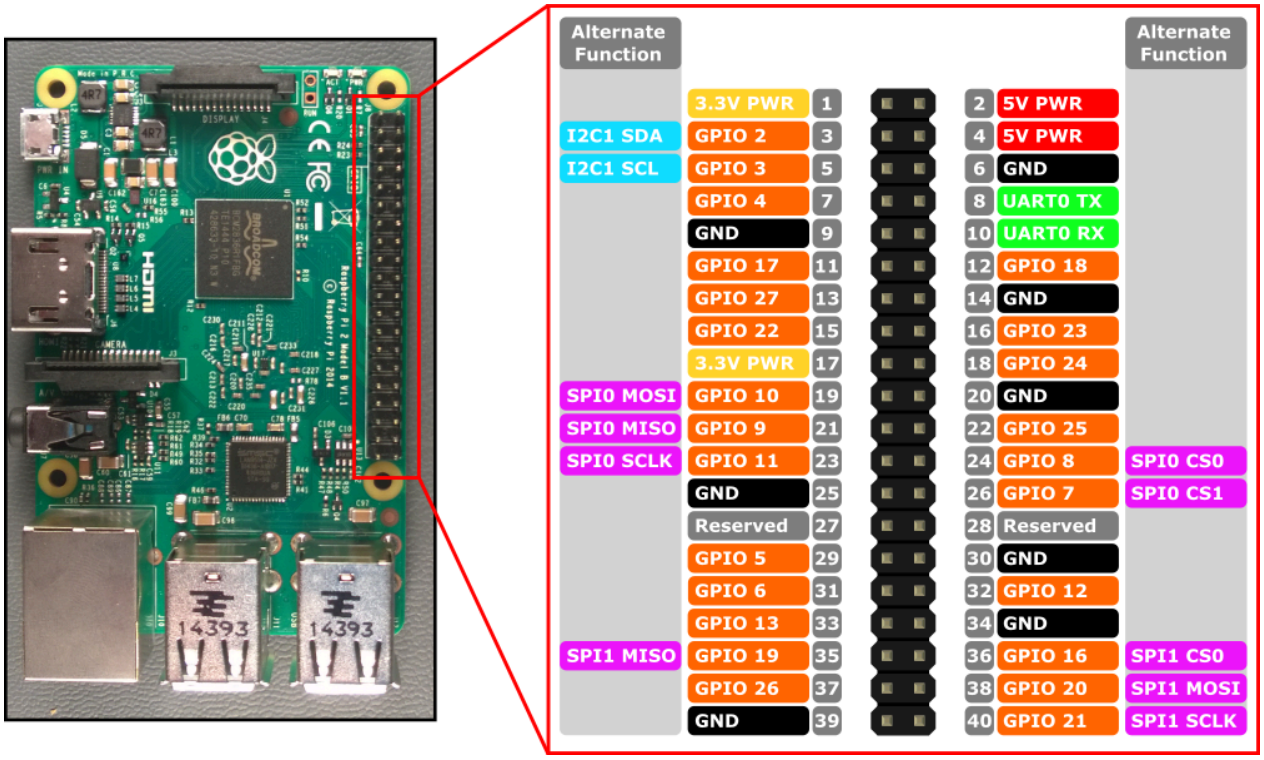
The lab is designed to introduce the students to get the basic idea on motor about how they function. Students will be given an introduction on servo motors, duty cycle and working mechanism. Also they will incorporate the last lab's task which was control LED with a push button.

III. Learning Outcome:

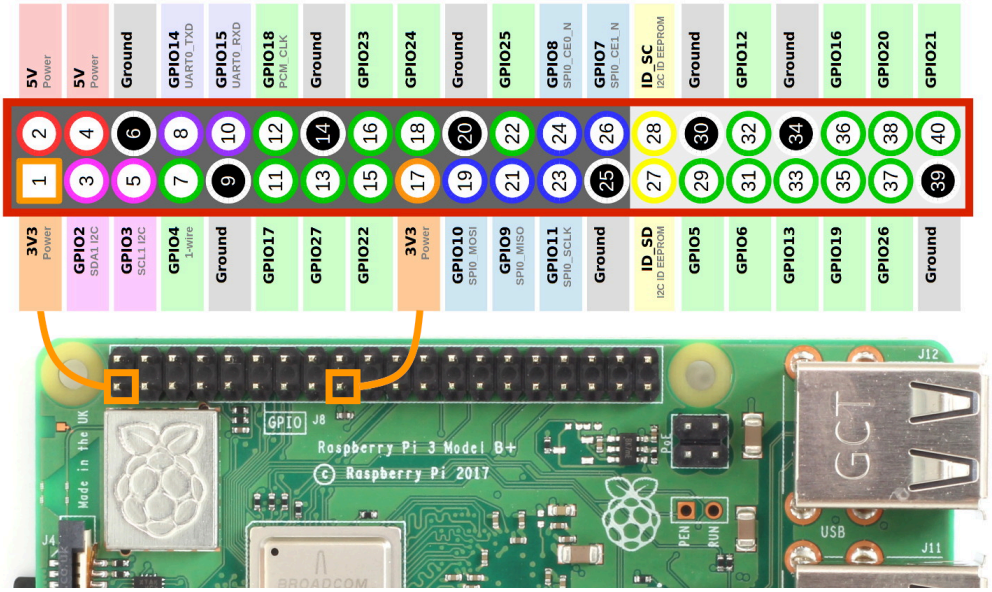
After this lecture, the students will be able to:

a. Know the functionality of servo motor
b. Rotate the motor to a specific position
c. Program the GPIO pins to control different circuits for different usages
d. Use the GPIO pins in unison with push buttons and LEDs to create an elementary user-controlled circuit

The Raspberry Pi 4 has 40 GPIO pins that can be easily configured to read inputs or write outputs.

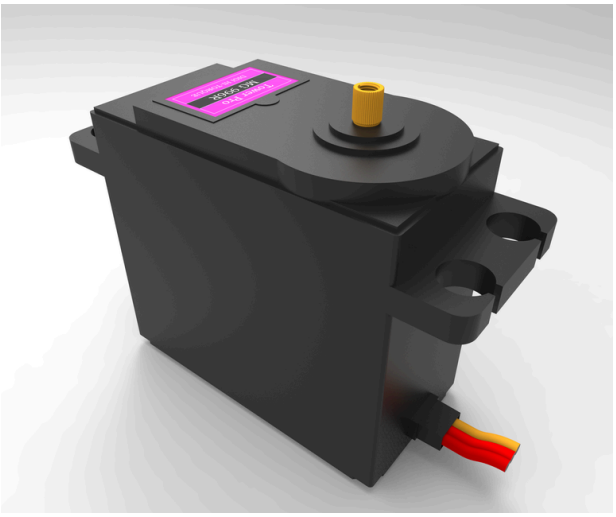


Here, you will be able to learn the functioning of each pin, which helps you to do things on your Raspberry Pi 4 easily. There are 40 pins in this model and among them 26 are GPIO pins.



Servo Motor :

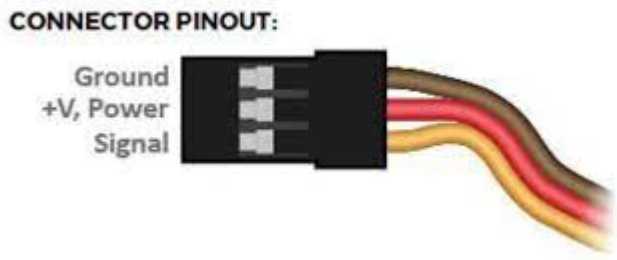
We are using the MG996r Model.



The **MG996R** is a metal gear servo motor with a maximum stall torque of 11 kg/cm. Like other RC servos the motor rotates from 0 to 180 degree based on the duty cycle of the PWM wave supplied to its signal pin

Wire Configuration

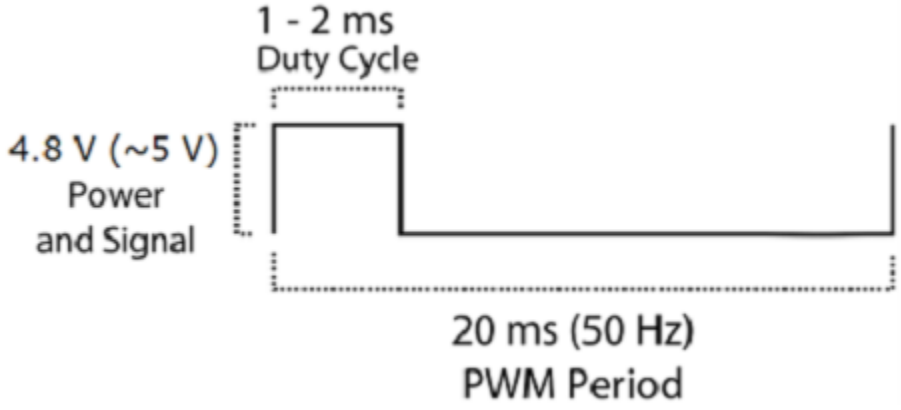
Wire number	Wire color	Description
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor



How to use a Servo Motor:

After selecting the right Servo motor for the project, comes the question of how to use it. As we know there are three wires coming out of this motor. The description of the same is given in the previous section. To make this motor rotate, we have to power the motor by connecting the Red

wire to +5V DC power supply, Brown wire to the Ground and sending the PWM signals to the Orange colored wire. Hence we need something that could generate PWM signals to make this motor work, this something could be anything like a 555 Timer or other Microcontroller platforms like Arduino, PIC, ARM or even a microprocessor like Raspberry Pi. Now, how to control the direction of the motor? To understand that let us look at the picture :



From the picture we can understand that the PWM signal produced should have a frequency of 50Hz, that is the PWM period should be 20ms.

The position of the servo depends on the "ON" part of the cycle. The "ON" time can vary from 1ms to 2ms. So when the "ON" time is 1ms the motor will be at 0° (fully right position) and when 1.5ms the motor will be 90° (center position), similarly when it is 2ms it will be 180°(fully left position). So, by varying the "ON" time from 1ms to 2ms the motor can be controlled from 0° to 180°.

The "ON" time can also be represented in terms of Duty Cycle. The Higher the Duty Cycle of a signal, the greater the "ON" time of that signal.

Relationship between the Duty Cycle and Pulse Width:

DutyCycle = PulseWidth*frequency

∴ DC = 0.001 *50 = 0.05 = 5% (0 degree, full right position)

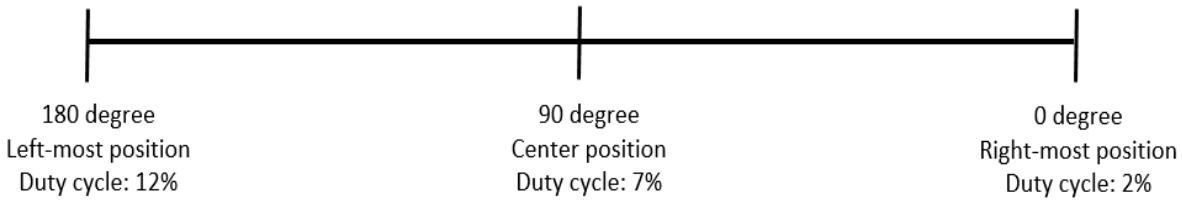
DC = 0.0015 *50 = 0.075 = 7.5% (90 degree, center position)

DC = 0.002 *50 = 0.1 = 10% (180 degree, full left position)

To sum up, to move the servo full left we need a Duty Cycle of 10%, for the center position a Duty Cycle of 7.5%, and for the full right position a Duty Cycle of 5%. This is all based on a 50 Hz signal. Intermediate values would linearly scale between these values.

The values of the Duty Cycles stated above may vary slightly from one servo to another. Thus, in the experimentation to test whether these values of duty cycles are the same for our servo or not, we found out that for our servo:

- A Duty Cycle of 12% yielded the full left position (180 degrees)
- A Duty Cycle of 7% yielded the center position (90 degrees)
- A Duty Cycle of 2% yielded the full left position (0 degree)



Now, how do we find the Duty Cycle for our desired angle? For example, what will be the Duty Cycle if we want the servo to rotate 60 degrees?

For that, we can linearly calibrate our duty cycle from 2% (0 degree, full left position) to 12% (180 degree, full right position), so these 2 extreme points can be written in the coordinate system form like this : (0,2) and (180,12)

with these two points we can calculate the slope of the line:

$m = (y2-y1) / (x2-x1) = (12-2) / (180-0) = 10 / 180 = 1/18$

Next, we will be representing the relationship between duty cycle and angle in the form of an equation, so finding the equation:

$y - y1 = m (x - x1)$

for the point (0,2)

$y-2 = 1/18 (x - 0)$

so the relation between duty cycle and the angle can be represented in the following way:

$y = 1/18x + 2$

where, x = desired angle
y = duty cycle

$\therefore$ Duty Cycle = (1/18) * desired angle +2

Therefore, for rotating the servo to 60 degrees, we can calculate the duty cycle in the following way:

Duty Cycle = (1/18) * 60 +2

= 5.3 %

Steps for basic servo connection:

- First, connect the power wire of the servo (red for us), to a 5v power pin of the raspberry pi (Pin 2 or 4).
- Then connect the ground wire of the servo (brown), to any ground pin of the raspberry pi.
- Lastly, connect the pwm wire of the servo (orange), to any GPIO pin of the raspberry pi. We will use the GPIO pin number in the code below.

Code for Basic Servo Connection:

*(First go to the Terminal and type: **sudo pigpiod,** then open the python IDE and write the code.)

```
from gpiozero.pins.pigpio import PiGPIOFactory

from gpiozero import Device, Servo, AngularServo

from time import sleep

Device.pin_factory = PiGPIOFactory()

s = AngularServo(18,min_angle = 0, max_angle =
180,min_pulse_width=0.5/1000,max_pulse_width = 25/10000)

while True:

    s.angle=120# (120 degree to the left)

    sleep(1)

    #right

    s.angle=60 # 60 degree to the right

    sleep(1)
```

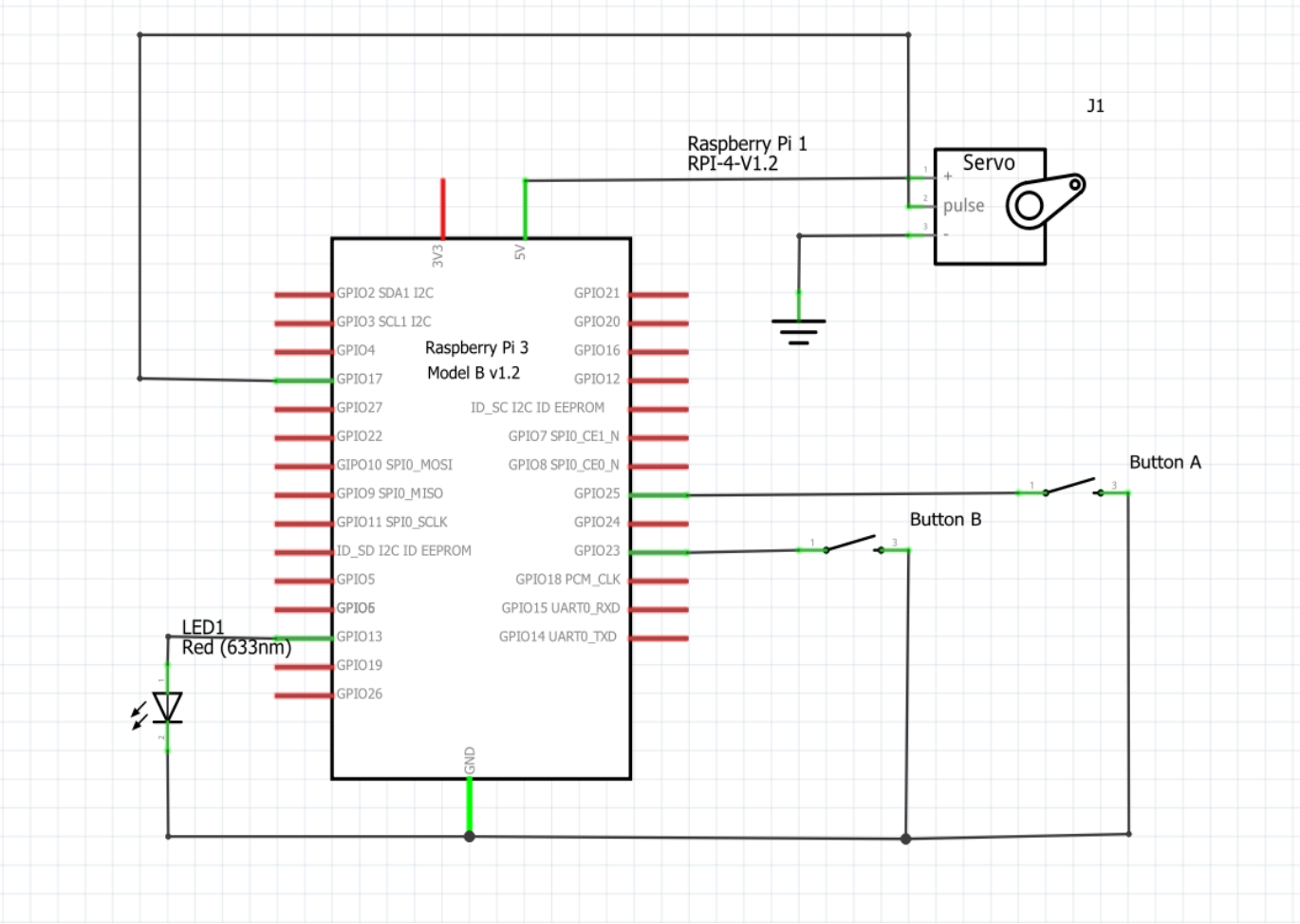
Part II: Controlling an LED with the help of a push button

Components required for the setup:

For controlling the LED with a push button on the Raspberry Pi 4, we need the following electronic components:

- **Raspberry Pi 4**
- **Servo Motor**
- **LED**
- **A resistor of 220 ohms**
- **Push-button**
- **Connecting wires (female to male)**

Circuit diagram of the setup:



To start the pigpio daemon

sudo pigpiod

To stop the pigpio daemon

sudo killall pigpiod

The code:

```
from gpiozero.pins.pigpio import PiGPIOFactory
from gpiozero import Device, LED, Button, Servo, AngularServo
from time import sleep

button1= Button(27) #pin 13
button2= Button(22) #pin 15
led=LED(4) #pin 7


Device.pin_factory = PiGPIOFactory()


s = AngularServo(17,min_angle = 0, max_angle =
180,min_pulse_width=0.5/1000,max_pulse_width = 25/10000)


while True:

    button1.wait_for_press()
    s.angle=120# (120 degree to the left)
    led.on()
    sleep(1)
    led.off()
    #right

    button2.wait_for_press()
    s.angle=60 # 60 degree to the right
    led.on()
    sleep(1)
    led.off()
```

Code with duty cycle control:

```
import RPi.GPIO as GPIO
from time import sleep

from gpiozero import Button
```

```
from gpiozero import LED
button1= Button(27)
button2= Button(22)
led1=LED(4)



GPIO.setmode(GPIO.BCM)
GPIO.setup(17,GPIO.OUT)
pwm=GPIO.PWM(17,50)
pwm.start(7)  #center(90 degrees)


while True:

    button1.wait_for_press()
    pwm.ChangeDutyCycle(5.3) # (left)
    led1.on()
    sleep(1)
    led1.off()


    button2.wait_for_press()
    pwm.ChangeDutyCycle(8.7) # right
    led1.on()
    sleep(1)
    led1.off()
```

Lab Task:

Now your task is to update the circuit and modify the code in such a way that you can implement following task:

- Implement 2 LED
- Press Button_A and one LED will turn on and servo motor will rotate for a specific angle
- Press Button_B and another LED will turn on and servo motor will rotate for a specific angle