Lab Worksheet 5

CSE461: Introduction to Robotics

Department of Computer Science and Engineering

Lab 05: Controlling a servo motor using Raspberry PI microcontroller based on the input of an ultrasonic sensor.

I. Topic Overview

The lab is designed to introduce servo motors so that the students get the basic idea on controlling motors and how they function. Students will be given an introduction on servo motors, duty cycle and working mechanism. Also they will incorporate an ultrasonic sensor in a task which should be covered in one of the previous labs. Depending on the sensor input they will control the servo motor.

II. Learning Outcome

After this lab, students will be able to:

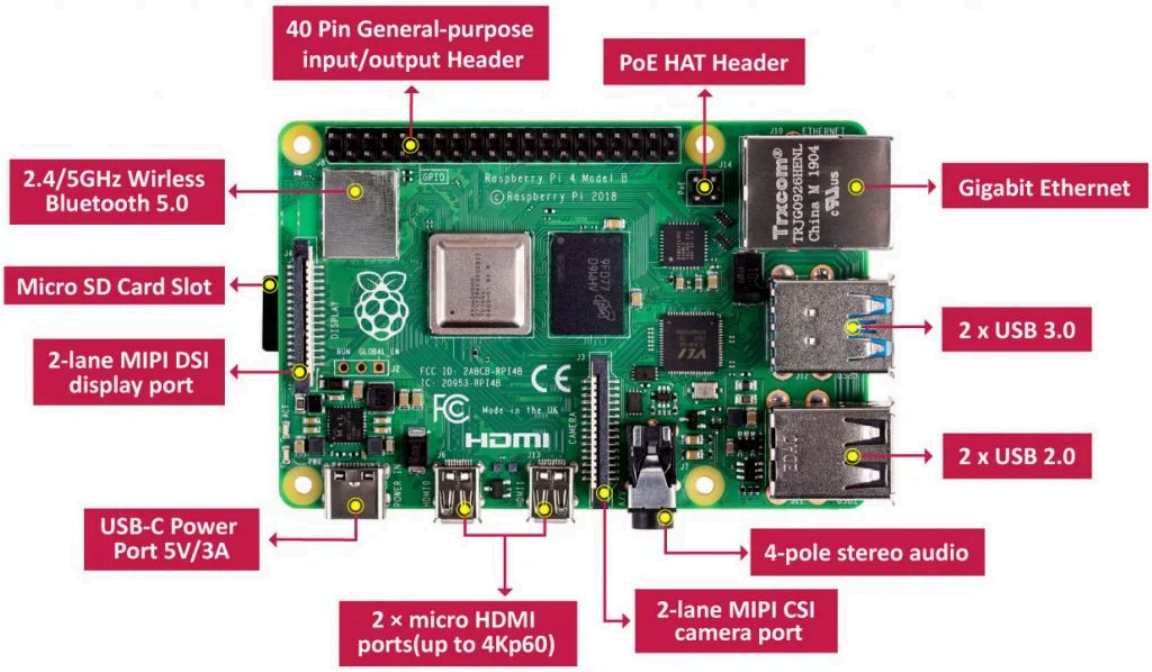
1. Know the functionality of servo motors.
2. Rotate the motor to a specific position using PWM.
3. Program the GPIO pins to control different circuits for different usages.
4. Control the servo motor based on the real-time sensor data.
5. Learn how to build a simple automated system that interacts with the environment and reacts in real-time.

III. Materials

- Raspberry PI development board
- Servo Motor
- Ultrasonic Sensor(HC-SR04)
- Resistor (appropriate value for current limiting, typically around 220 ohms)

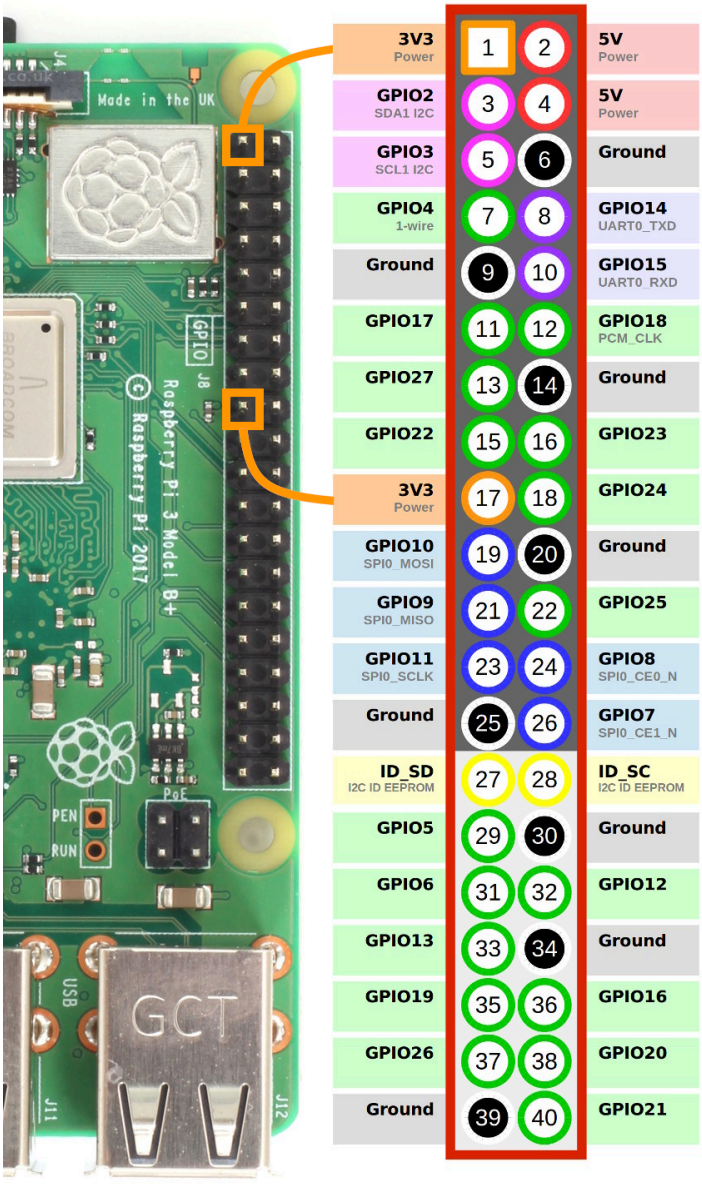
- Jumper wires
- Breadboard
- USB pen drive containing OS

IV. Raspberry PI Overview



Raspberry Pi 4 Model B features a high-performance 64-bit quad-core processor, dual-display support at resolutions up to 4K via a pair of micro HDMI ports, hardware video decode at up to 4Kp60, up to 8GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on). For the end user, the Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems. Raspberry Pi OS (previously called Raspbian) is the recommended operating system for normal use on a Raspberry Pi. Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card or a pendrive, ready to use with your Raspberry Pi. Then the board can be programmed using user-friendly IDEs; for example, Python programming can be done using Thonny IDE.

V. Raspberry Pi Pins:



The Raspberry PI 4B has the following types of pins:

1. **Power Pins:**
    a. **5V and 3.3V:** Provide regulated voltage output.
    b. **GND:** Ground pins.

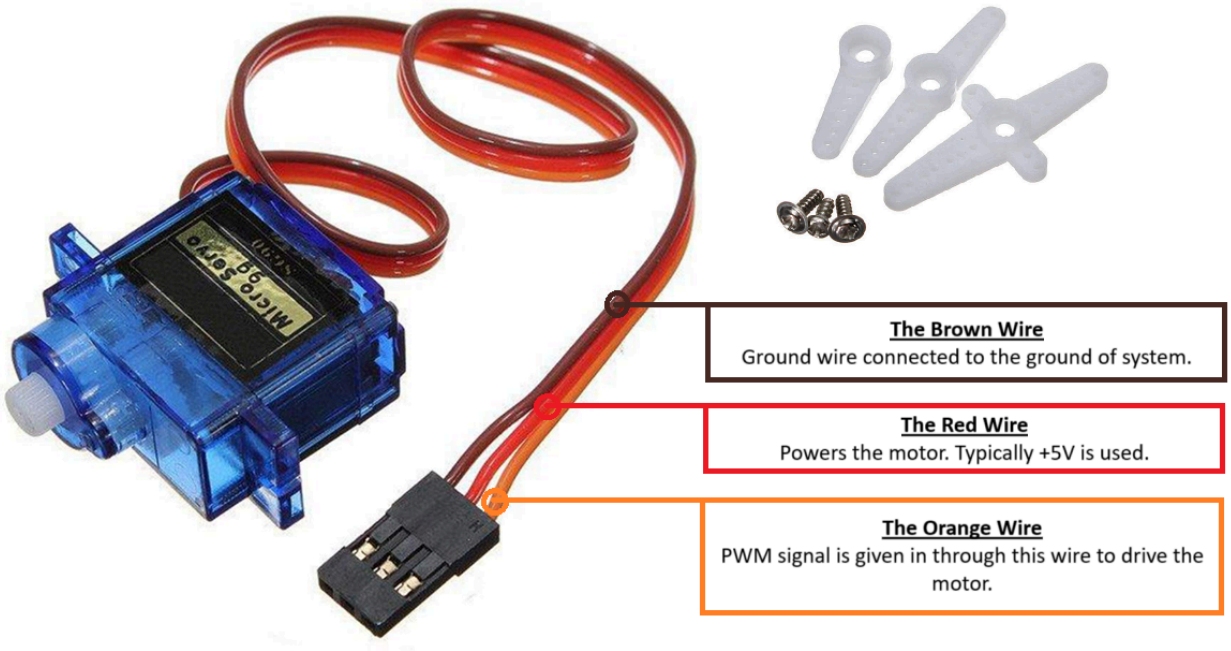
2. **GPIO Pins:** The general purpose input and output pins. Some of these GPIO pins can be used for following different communication protocols like UART, I2C and SPI to communicate with other devices. The GPIO pins work at a high voltage of 1.8V-3.3V maximum.

V. Raspberry PI OS Installation Guideline: Setting up Raspberry PI

VII. Experiment 1: How to use a servo motor.

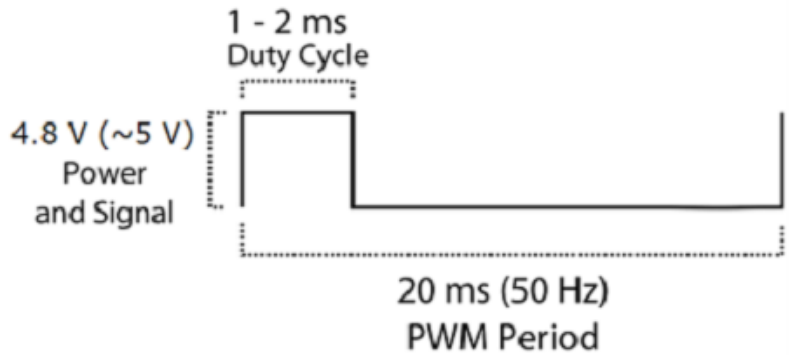
1. **Description the components: Servo Motor**

   We are using the SG90 Model. The Servo Motor SG90 is a versatile and high performance motor designed for a range of applications. Weighing just 9g, this compact servo motor operates efficiently at a voltage range of 3.0V to 7.2V. It provides an impressive operating speed of 0.12sec/60° at 4.8V (no load) and offers a stall torque of 17.5oz/in (1.2 kg/cm) at 4.8V. 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.



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 which is the percentage of a signal's "ON" time. The Higher the Duty Cycle of a signal, the greater the "ON" time of that signal.

Relationship between the Duty Cycle and Pulse Width:

$$DutyCycle = PulseWidth \times frequency$$

Keeping the frequency at 50Hz, by modulating the pulse width we can achieve different duty cycles ultimately achieving different angle of rotation for the servo motor.

$\therefore$ DC = 0.001 × 50 = 0.05 = 5% (0 degree, full right position)

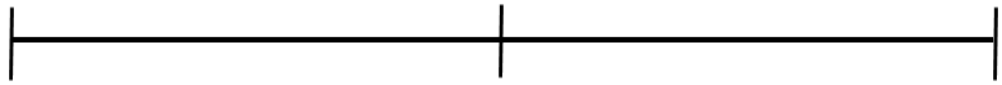
DC = 0.0015 × 50 = 0.075 = 7.5% (90 degree, center position)

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 50Hz 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 right position (0 degree)



180 degree
Left-most position
Duty cycle: 12%

90 degree
Center position
Duty cycle: 7%

0 degree
Right-most position
Duty cycle: 2%

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:

$$y - y1 = m (x - x1)$$
$$\Rightarrow y-2 = 1/18 (x - 0) \text{ [for the point (0, 2)]}$$

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

$$y = (1/18) x + 2$$
where, x = desired angle and 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:

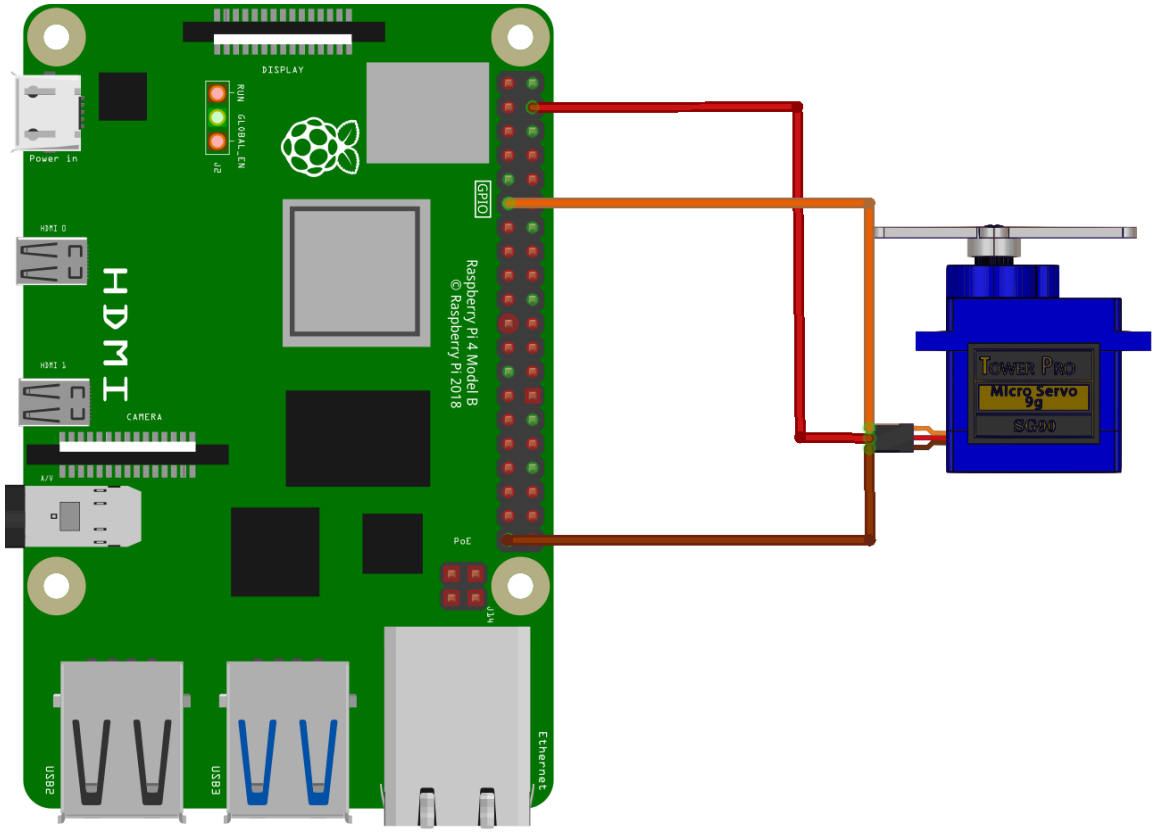
$$\text{Duty Cycle} = (1/18) * 60 + 2 = 5.3\ \%$$

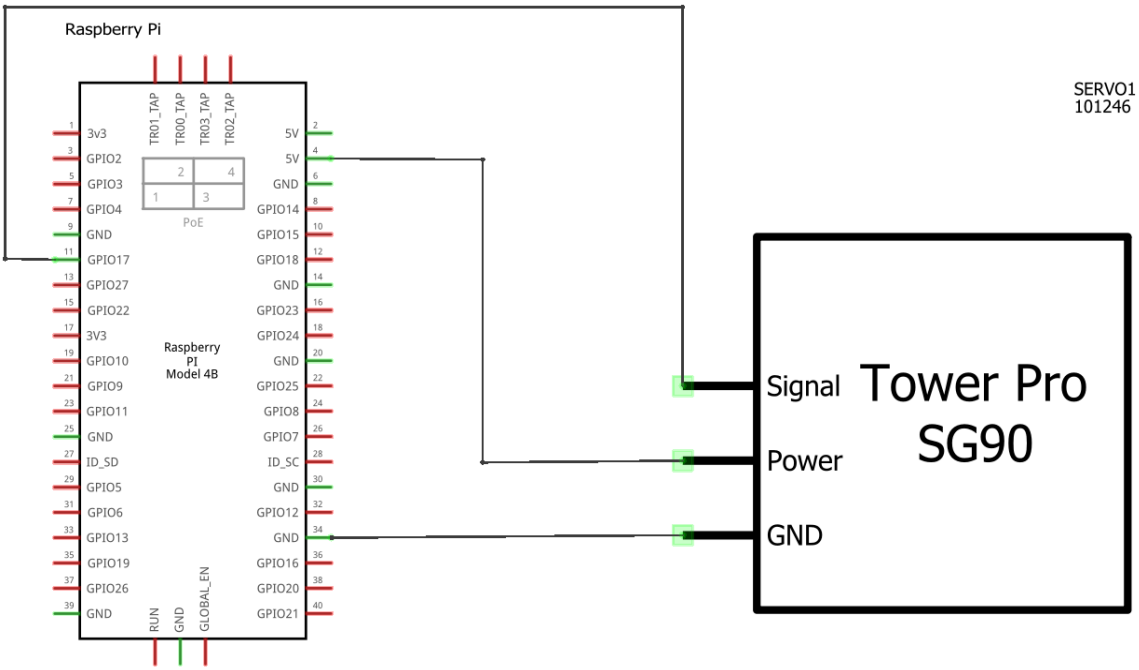
Terminologies:

- **PWM:** Stands for Pulse Width Modulation. A technique of modulating a signal by varying the width of pulses while maintaining a constant frequency.
- **Duty cycle:** The percentage of the time a signal was ON or High.

2. **Setting up the circuit:**
    - Connect the power wire of the servo (red), to a 5v power pin of the raspberry pi.
    - Connect the ground wire of the servo (brown), to a ground pin of the raspberry pi.
    - Lastly, connect the pwm wire of the servo (orange), to any GPIO pin of the raspberry pi. Pin GPIO17 is used in the code.

3. **Circuit diagram:**





4. **Code:**

We can control the servo either by changing the duty cycle through the code or we can use a pre-built class from `gpiozero` library named `PiGPIOFactory` which is a pin factory that provides low-level access to the raspberry pi GPIO pins offering features like PWM.

- First, go to the terminal and type: **sudo pigpiod** to start the pigpio daemon.
- Then, open a .py file in a folder on the RPI OS. Open Thonny IDE and save the code in the previously created file. Then run the code.
- To stop the pigpio daemon go to the terminal and type: **sudo killall pigpiod**

pigpiod is a daemon, meaning it runs in the background, listening for commands and requests related to GPIO control.

We are using another class called `AngularServo` from the same library to represent the PWM controlled servo motor in the program. Following are the parameters we have used:

pin – The GPIO pin that the servo is connected to.

min_angle – Sets the minimum angle that the servo can rotate to.

max_angle – Sets the maximum angle that the servo can rotate to.

min_pulse_width – The pulse width corresponding to the servo's minimum position.

max_pulse_width – The pulse width corresponding to the servo's maximum position.

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

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:
    s.angle=120# (120 degree to the left)
    sleep(1)
    #right
    s.angle=60 # (60 degree to the right)
    sleep(1)
```

If you want to learn more about the library, you can check out their documentation from here.

VIII. Lab Task: Rotate a servo motor to 180 degree if the distance measured by the ultrasonic sensor is less than 5CM otherwise rotate the motor to 90 degree. [Hurray! You have just made an automated parking gate which opens the gate if a car is nearing the gate otherwise remains close]

IX. References:

1. https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
2. https://www.raspberrypi.com/documentation/computers/getting-started.html
3. https://gpiozero.readthedocs.io/en/stable/api_output.html