Servo Motor Test Notes

Control Method:

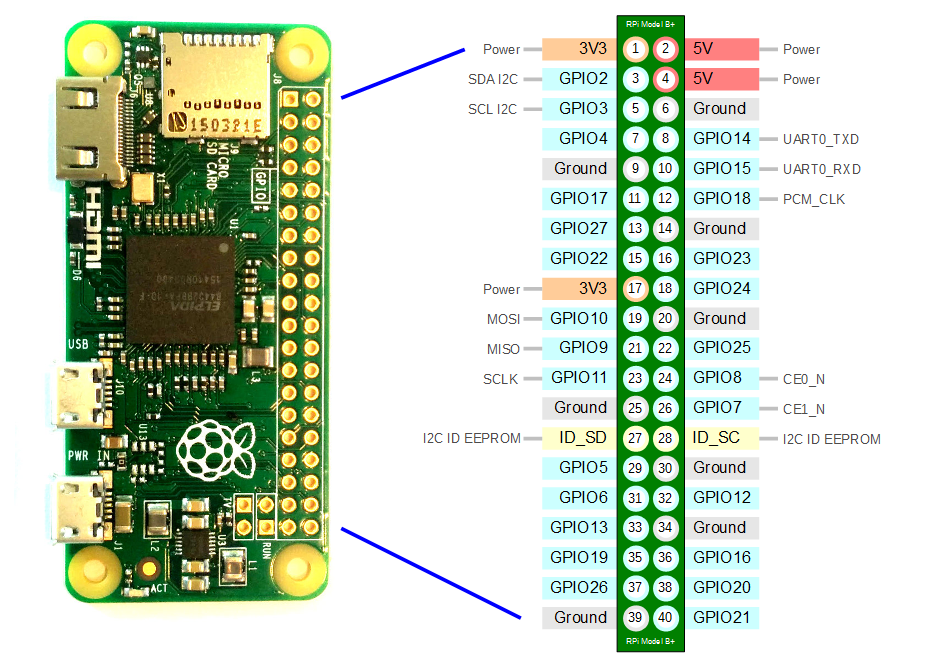
Library: Rpi.GPIO

This allows access to pin control and pwm setup in python.

Code Implementation:

To initialize the pins, use:

GPIO.setup(pinNum, GPIO.OUT)



Replace pinNum with the number of the pin you would like to configure. They are listed as 1-40 in the diagram above.

To configure a Pulse Width Modulation (PWM) output, use:

pwmFormat = GPIO.PWM(pinNum, frequency)

This will setup a pin to be a PWM output. Replace pinNum with the pin number as described previously. Replace frequency with the value you wish to use in Hertz (Hz). The value is not scaled so GPIO.PWM(15, 100) sets pin 15 to output a 100Hz PWM. Change pwmFormat to whatever variable name will help organize the code.

To run and modify the pwm signal:

pwmFormat.start(dutyCycle) #Initiates pwm signal for pwmFormat

pwmFormat.ChangeDutyCycle(newDutyCycle) #Changes duty cycle to new value

pwmFormat.stop() #Ends pwm output

The arguments of .start and .ChangeDutyCycle are the percentage of duty cycle. pwmFormat.start(50) outputs a 50% duty cycle.

Translation of PWM to Servo Control:

The servos used in this project are continuous, meaning their rotating speed is controlled by their PWM input. Input out of bounds of their parameter will yield no rotation. A spectrum of Clockwise (CW) and Counterclockwise (CCW) parameters are provided. These parameters are given in microseconds (µs).

To translate this to frequency and duty cycle, one must convert frequency to time:

pwmTest = GPIO.PWM(15, 1000)

Pin 15 is set to output a 1000Hz PWM signal. The period of its cycle is: T = 1 / f

This equation gives us T = 1ms. If we begin at a duty cycle of 25%:

pwmTest.start(25)

We do not output a full 1ms signal, but a fraction of one. Specifically: TPWM = T \* PDuty

For PDuty = 25: TPWM = 1ms \* 0.25 = 0.25ms or 250µs.

Hence, we can use the PWM range given in the servo specification to determine what range of frequency and duty cycle would work.

Turret Servo Specifications Link: <https://www.digikey.com/en/products/detail/pololu-corporation/2820/10450037>

Movement Servo Specifications Link: <https://www.sparkfun.com/datasheets/Robotics/servo-360_e.pdf>

Turret Servo Specifications:

CW for PWM 700-1500µs

CCW for PWM 1500-2300µs

Turret Servo Tested Values:

Frequency: 400Hz -> T = 2.5ms = 2500µs

Duty Cycle: Duty<54 -> (Faster CW)

Duty Cycle: 54 -> 1350µs (Slowest CW)

Duty Cycle: 55-57 -> Stopped or Stuttering (Better to output 0 Duty Cycle to stop)

Duty Cycle: 58 -> 1450µs (Slowest CCW)

Duty Cycle: Duty>58 -> (Faster CCW)

Fractional duty values work inconsistently. It may be better to use a larger frequency to get better precision.

It was determined that 625Hz is not a usable frequency. Nor is 500Hz or 300Hz. The servos may expect a pulse at a given frequency. Deviating from this may confuse them. Especially when 1LS operates faster and at a different frequency to 1RS.

Operational Values for Turret Servo:

The values to be used for Turret Servo operation:

Frequency = 400Hz, Duty Cycle CW: 54, Duty Cycle CCW: 58

Note that these values may not yield equal speeds and may have to be calibrated.

Movement Servo Tested Values:

Frequency: 400Hz -> T = 2.5ms = 2500µs

Duty Cycle: Duty<54 -> (Faster CW)

Duty Cycle: 54 -> 1350µs (Slowest CW)

Duty Cycle: 55-57 -> Stopped or Stuttering (Better to output 0 Duty Cycle to stop)

Duty Cycle: 58 -> 1450µs (Slowest CCW)

Duty Cycle: Duty>58 -> (Faster CCW)

Movement servo delivered very similar results, though with a lower RPM and far less stuttering.

Movement Servo Synchonization:

1LS and 1RS operate at different speeds. Efforts to synchronize must be made. Stepping through 1RS frequencies will likely be most productive. So far:

Frequency = 400Hz, D1LS = 60.0, D1RS = ~69.8 +- 0.05

Further testing needs to be done, but this should be acceptable for the moment.