GEAR currently has two Adafruit Serov HATs for RPi. You can find a tutorial on using the HAT here:
https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi

In this lesson you will learn how to control a servo connected to the HAT. You will need the following parts:

1. Raspberry Pi with the Adafruit library for the HAT installed
2. Servo HAT
3. Hitec model HS-485HB servo
4. 5.2 volt DC power supply for the HAT (four AA rechargeable batteries in a battery holder)

## Attach the Hat to the Raspberry Pi

The HAT has a 40-pin header that fits over the GPIO pins of the RPi. Connect the HAT to the RPi and use some standoffs to secure the HAT to the RPI (if you do not add the standoffs, there is a possibility of shorting out the HAT on the HDMI housing of the RPi).

## Configure your Pi to use I2C devices

You probably have the software below installed, but run the commands to confirm.

```
1. sudo apt-get install python-smbus
2. sudo apt-get install i2c-tools
```

## Enable I2C on your RPi

In the RPi **Preferences** menu select **RPi Configuration**. Then select the **Interfaces** tab. Set **I2C** to **enable**.

## Check functioning of HAT

In the **Terminal** enter command below.

```
sudo i2cdetect -y 1
```

A table of i2c devices is displayed. In the first column (labeled 0), on the row labeled 40, there should be the number 40. In the same column on the row labeled 70, there should be the number 70. If you see these two numbers listed in the first column, then the RPi is properly connected to the HAT.

## Download the software for the HAT

In the **Terminal** enter the command in step 1 on next page, which will download the software. Steps 2 and 3 will change your current directory to the location of a sample Python program for controlling a servo (Servo_Example.py). You must save your own Python programs in this folder. Otherwise, you will get an error stating the library can't be found when you try running the program.

```
1. git clone -b legacy https://github.com/adafruit/Adafruit-
   Raspberry-Pi-Python-Code.git
2. cd Adafruit-Raspberry-Pi-Python-Code
3. cd Adafruit_PWM_Servo_Driver
```

**Writing the Python program to control servo**

Write the code below in Python version 2.

```
from Adafruit_PWM_Servo_Driver import PWM
import time

# Initialise the PWM device using the default address

pwm = PWM(0x40)

def getInput():

  while True:

    position = raw_input('enter number between 123 and 492...')
    if (position > "122") and (position < "493"):
      position = int(position)
      return position

    else:

      print 'you did not enter a proper number, try again'

pwm.setPWMFreq(50)          # Set frequency to 50 Hz

while (True):

  # Change position of servo on channel O

  position = getInput()

  pwm.setPWM(0, 0, position)

  time.sleep(1)
```

**Synopsis of above program**

The program asks for user input, a number between 123 and 492. The number entered by the user causes the servo to rotate to a specific position. The **raw_input** method is used to avoid errors that will happen if a user enters letter characters instead of numbers (if you use the **input** method, a letter will result in an error). The **if** statement just below the raw_input line determines if the user input falls between 123 and 492. If it does, the string stored in the object named **position** is converted to an integer, then passed back to the line calling the **getInput()** function.

The **.setPWM()** method takes three arguments: channel, start, stop. The channel is the connection point on the HAT. There are 16 connection points, numbered 0 though 15. For timing, this library uses the number 4096, which represents the number of clock ticks which occur for each cycle of the PWM. If the start argument is set at 0, as in our example, then the signal rises to HIGH at the very start of the cycle. In the code above, the stop argument is determined by the contents of the object named **position**. According to the **getInput()** function, the number can vary between 123 and 492. We need to convert those numbers to microseconds so that we can understand the code. Let us work on that now.

Note that the frequency of the PWM is set to 50 Hz. That means that each cycle has a period of 20 milliseconds (1/50) or 20,000 microseconds. Therefore, 4096 clock ticks equal 20,000 microseconds and each clock tick is 20,000/4096 = 4.88 microseconds/tick. Therefore, 123 ticks is equivalent to 123 x 4.88 = 600 microseconds. A pulse width of 600 microseconds will set our particular servo very near its limit in counterclockwise rotation. Let us calculate the other number, 492 ticks, 492 x 4.88 = 2400 microseconds. This pulse width is near the limit of full rotation in the clockwise direction for our servo.

**Testing the HAT**

You are now ready to try out your program.

1. Connect the servo to channel 0 on the HAT. It is important that you attach the servo correctly. The black wire should be toward the edge of the HAT and the yellow wire on the inside (if you are not sure about this, please ask Mr. La Favre – we do not want to damage the HAT electrically).
2. Connect the power wires from the battery supply to the HAT – MAKE SURE to connect the wires properly, positive to positive and negative to negative.
3. Insert the batteries into the battery holder. If you do this properly, the green LED on the HAT will start to glow.

You are now ready to test your program. Run the program and enter various numbers between 123 and 492. Observe the results of servo rotation. Try entering a letter or number outside the range of 123 to 492 and observe the message returned on the Python console.