TITTLE: Introduction to Servos

DURATION: 60 Mins

CC STANDARDS: CCSS.ELA-LITERACY.RST.6-8.3, CCSS.ELA-LITERACY.RST.9-10.3, CCSS.ELA-LITERACY.RST.6-8, 9-10, 11-12. 4, CCSS.ELA-LITERACY.RST.9-10.5

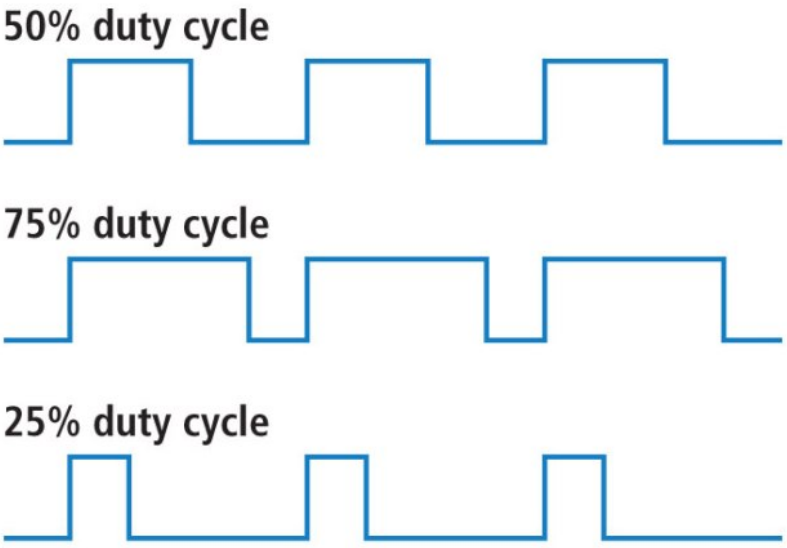
MATERIALS: PowerPoint Presentation, Laptops (students can be grouped if not enough laptops are present), Internet, Raspberry Pi, jumper wires, servos, AeroWeb's Website

INTRODUCTION:

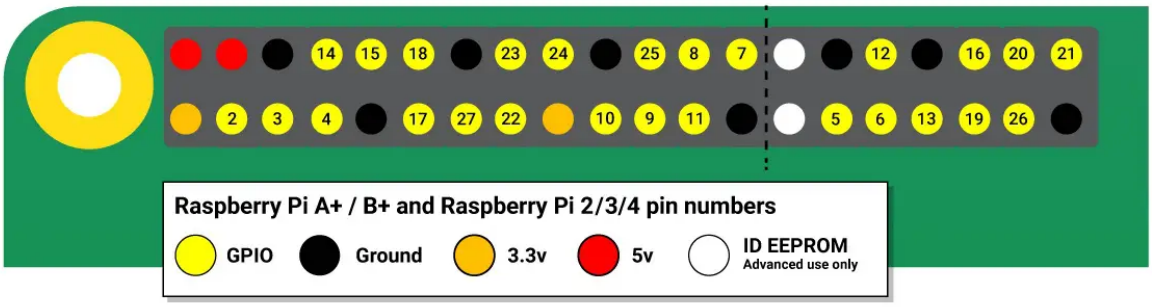
- Teacher reviews prior lesson that covered Python Loops by asking leading questions such as "What is a python loop? What does a "while" do? Can you give me an example?"
- Teacher introduces the topic and sets the learning goals which are:
    - Learn what are servos
    - Learn what a Pulse Width Modulation is
    - Learn how driving a servo is associated with a duty cycle
    - Learn how to use preconfigured code required for Raspberry Pi
- Teacher reminds the class of the opportunities, knowing how to use a Raspberry Pi can teach them how to control different sensors in robotics, for example, moving the wings of an aircraft up or down using servos. (AeroWeb recommends using the following exercise https://www.aeroweb.info/servos/)

LESSON PART 1:

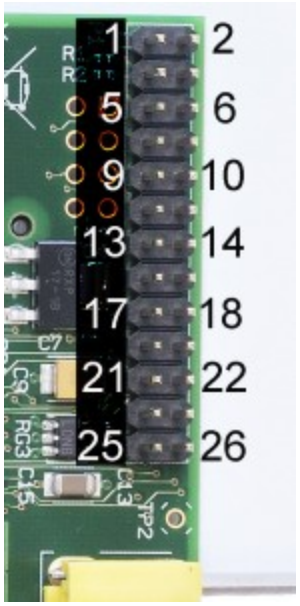
- Teacher introduces the concept of a servo. Reminds the students that a servo does not spin 360 degrees like other motors. Instead, a servo allows precise rotation within 180 degrees.
- The teacher explains how a pulse width modulation (PWM) drives the servo shaft. Reminds the students that a PWM is a type of digital signal to control the direction of a servo.
- Teacher introduces the concept of Duty Cycle which is measured in percentage. If a digital signal spends half of the time on and the other half off, we would say the digital signal has a duty cycle of 50% and resembles an ideal square wave.



- For example, in our servo code, 2% duty cycle corresponds to 0 degrees and 12% duty cycle corresponds to 180 degrees.
- Teacher explains the code for the Raspberry Pi, starting with the need to import the python libraries and the 'time' library
- Teacher explains the Raspberry Pi Pinout and introduces the concept of GPIOs or General-Purpose Input & Output pins



- Teacher explains that the servo has three leads, one for voltage (5V), one for ground, and one for the PWM signal
- Teacher explains that the Raspberry Pi has 40 pins and in this lesson, we name the pins by counting them from left to right:



- The teacher clarifies the slightly confusing part that each pin also has a name, according to what it does. The best way to see which pin number does what is with a diagram:

3V3	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7

GPIO pinouts for Rev 2 Pi

- The teacher explains how to set up BOARD and GPIO numbering schemes. In RPi.GPIO you can use either pin numbers (BOARD) or the Broadcom GPIO numbers (BCM), but only one system can be used in each program.

- The teacher explains that at the top of every script, after importing the RPi.GPIO module, the GPIO numbering mode is set

```
import RPi.GPIO as GPIO

# for GPIO numbering, choose BCM
GPIO.setmode(GPIO.BCM)

# or, for pin numbering, choose BOARD
GPIO.setmode(GPIO.BOARD)

# but you can't have both, so only use one!!!
```

- The teacher explains how to set up a GPIO port as an input by using the following line of code:

  GPIO.setup(Port_or_pin, GPIO.IN)

  Changing Port_or_pin to the number of the GPIO port or pin they want to use.

  For example, to use the BCM GPIO numbering and port GPIO25 use:

```
import RPi.GPIO as GPIO

1. GPIO.setmode(GPIO.BCM)  # set up BCM GPIO numbering
2. GPIO.setup(25, GPIO.IN) # set GPIO 25 as input
```

- The teacher explains how to read inputs. Inputs are Boolean values: 1 or 0, GPIO.HIGH or GPIO.LOW, True or False (this corresponds to the voltage on the port: 0V=0 or 3.3V=1). You can read the value of a port with this code:

  GPIO.input(25)

```
if GPIO.input(25): # if port 25 == 1
1.     print "Port 25 is 1/GPIO.HIGH/True"
```

- The teacher explains how to set pin 11 as output and for PWM for our servo demonstration. The teacher creates a variable named 'servo1' and assigns the value as follows:

```
GPIO.setup(11, GPIO.OUT)
```

```
servo1 = GPIO.PWM(11,50) # Note 11 is pin, 50 = 50Hz pulse
```

- The teacher explains that the PWM will start running with an initial value of 0 (pulse off)

```
servo1.start(0)
```

- The teacher shows how to wait for two seconds:

```
print ("Waiting for 2 seconds")
time.sleep(2)
```

- The teacher shows the students how to declare a duty cycle variable and use python loops to drive the servo shaft 180 degrees:

```
# Define variable duty
duty = 2
# Loop for duty values from 2 to 12 (0 to 180 degrees)
while duty <= 12:
    servo1.ChangeDutyCycle(duty)
    time.sleep(1)
    duty = duty + 1
```

- The teacher then explains how to wrap things up at the end of a code:

```
#Clean things up at the end
servo1.stop()
GPIO.cleanup()
print ("Goodbye")
```

- After discussion, the teacher have the students practice what they learned using the servos exercise link where they will drag the proper command from the right hand side into the rectangle

LESSON PART 2:

- The teacher opens the class to a group discussion as the students see how they can use python loops and commands to drive a servo shaft
- Teacher runs a demonstration by setting up the Raspberry Pi and connecting the servo leads. The teacher shows the code that allows a servo shaft to rotate 180 degrees.
- The teacher shows how they can change the duty cycle to move the servo shaft to different angles.
- Teacher asks the students what happens at different duty cycles?
- During this demo, the teacher asks the students what can be done next. Once the teacher finishes the group-led demo, the students proceed to work on their own computers, raspberry pi, and servos by themselves.
- Teacher walks around the room monitoring students.

CONCLUSION:

- At the end of the lesson, the teacher reviews what has been learned and what they accomplished.
- Asks students if there is something they did not understand, or they would like for the teacher to cover in the next class.
- Teacher moves the rocket up the path and reminds them they are getting closer to the first milestone, controlling an RC car with Python.
- Teacher concludes by giving them a challenge. Use python loops to move a servo shaft left and right for 5 consecutive times.