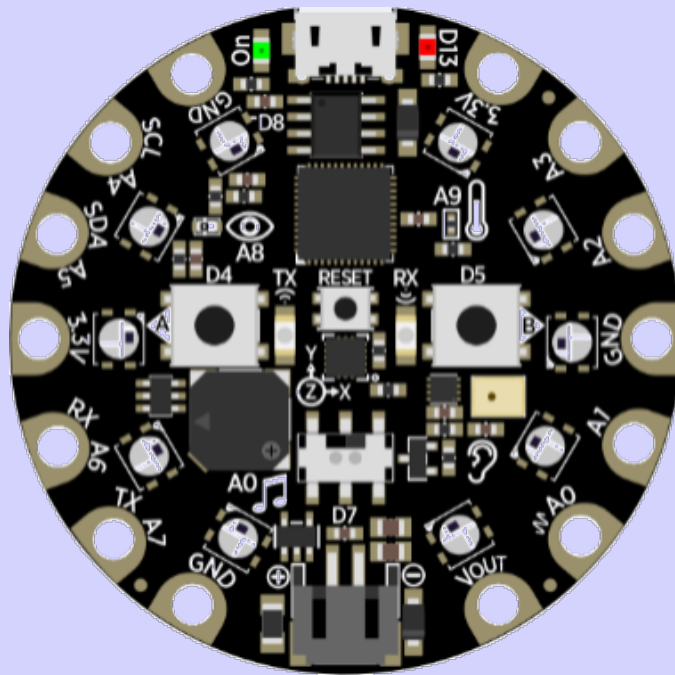


Digital Technology

CPX / CircuitPython Introduction



Version 2
2021

Barry Butler
bbutl58@eq.edu.au

ROBOCOAST
Sunshine Coast Robotics
www.robocoast.tech



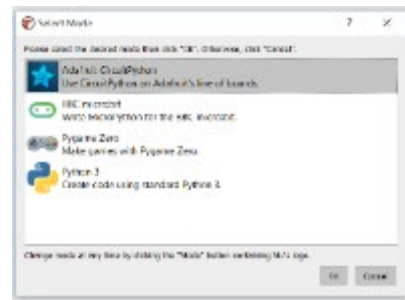
Contents

Section	Content
A	Introduction and Setup, Serial, Debugging Blinking LED's – single, all, tone, brightness
B	Buttons, Touchpads, Tap, Shake Conditional statements (if, elif, else)
C	Variables and Loops (repeat)
D	Read light sensor, map_range() function More conditional statements
E	Connect sensors – show as led or sound output Digital: Tracker, Break beam Analog: Potentiometer, Soil Moisture, Ultrasonic
F	Connect actuators Servos, motors
G	Connect a display
H	Other Useful Functions (including Color Wheel)

A. Introduction and Setup

Getting Started with the CPX board and CircuitPython

- Open the Mu IDE
- Click Mode, and select Adafruit CircuitPython
- Click New to start a new file (but do not save it yet)



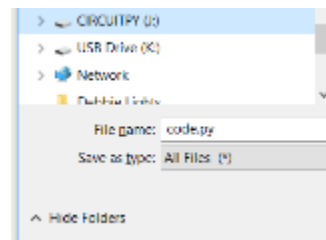
Our First Program – Blinking LED's

```
import time
from adafruit_circuitplayground.express import cpx

while True:
    cpx.pixels.fill((50, 0, 0))
    time.sleep(0.5)
    cpx.pixels.fill((0, 0, 0))
    time.sleep(0.5)
```

Save the File to the CPX

- Click the Save button
- Select the CIRCUITPY folder (that should be automatic)
- Enter the filename code.py.



Have a go at these to alter your code:

- set the led brightness `cpx.pixels.brightness = 0.2` (a value between 0 and 1)
- turn on only one led `cpx.pixels[6] = (255, 0, 0)` (index is between 0 and 9)
- play a tone `cpx.play_tone(400,0.5)` (frequency: 50 – 5000, seconds)

B. Buttons, Touchpads, Tap and Shake (Conditional Statements)

Buttons

Instead of the lights being turned on when code is downloaded, let's turn on the lights when we press the left button (A). We use an if statement to test if the button has been pressed.

```
import time
from adafruit_circuitplayground.express import cpx

cpx.pixels.brightness = 0.2
while True:
    if cpx.button_a:
        cpx.pixels.fill((50, 0, 0))
    if cpx.button_b:
        cpx.pixels.fill((0, 0, 0))
```

Use the Connection Pads as Switches

The CPX connection pads act as capacitance touch pads, that is, they respond to being touched. This code uses pin A7 to turn the lights on when it is touched and A1 to turn them off.

```
cpx.pixels.brightness = 0.3
while True:
    if cpx.touch_A7:
        cpx.pixels.fill((50, 0, 0))
    if cpx.touch_A2:
        cpx.pixels.fill((0, 0, 0))
```

Use the Serial Connection for Errors and Debug

When errors occur in the CPX code, you can be notified of the error. Click the **Serial** button to open a connection with the CPX. A print statement is used to give feedback from the program.

```
while True:
    if cpx.touch_A7:
        cpx.pixels.fill((50, 0, 0))
        print("Touched A7!")
    if cpx.touch_A2:
        cpx.pixels.fill((0, 0, 0))
        print("Touched A2!")
```

Shake It or Tap to Turn Lights On

We can use the motion sensor (accelerometer) to turn on the lights using a shake or a tap.

```
while True:
    if cpx.tapped:
        cpx.pixels.fill((50, 0, 0))
        time.sleep(0.5)
```

```
while True:
    if cpx.shake(10):
        cpx.pixels.fill((50, 0, 0))
        time.sleep(0.5)
        #between 1 and 20
```

C. Variables and Loops (Repeat and While)

Use a For Loop to Repeat Actions

There are several types of loops that can be used for repetitive actions. The first is a **for** loop. Use this to make the lights blink a fixed number of times depending on the button or touch pad pressed.

```
while True:
    if cpx.button_a:
        for i in range(4):
            cpx.pixels.fill((50, 0, 0))
            time.sleep(0.1)
            cpx.pixels.fill((0, 0, 0))
            time.sleep(0.1)
```

Use a While Loop to Turn on Individual Neopixels

For loops are great for very simple repetitive tasks. **While** loops give a lot more flexibility to many tasks.

```
while True:
    if cpx.button_a:
        i = 0 #set the count variable to the first value
        while i <= 9: #test the count against the final value
            cpx.pixels[i] = (255, 0, 0) #use the loop count variable (i)
            time.sleep(0.2)
            i = i + 1 #add one to the count
```

Turn on Neopixels in Reverse

A while loop can start at a high number and end at a low number.

```
while True:
    if cpx.button_a:
        i = 9 #set the count variable to the first value
        while i >= 0: #test the count against the final value
            cpx.pixels[i] = (255, 0, 0) #use the loop count variable (i)
            time.sleep(0.2)
            i = i - 1 #deduct one from the count
```

Try adding or subtracting different values to i

Turn on Alternate Colours

```
while True:
    if cpx.button_a:
        i = 0
        while i <= 9:
            if (i % 2 == 0): # i%2 tests for a remainder (0 or 1)
                cpx.pixels[i] = (0,80,150)
            else:
                cpx.pixels[i] = (100,80,0)
            i += 1
```

D. Read the Light Sensor and Map to Output

To read and information from sensors, variables must be used to store the sensor values. Information can be displayed in the serial window or on the plotter.

```
import time
from adafruit_circuitplayground.express import cpx

cpx.pixels.brightness = 0.3
while True:
    light = cpx.light
    print(light)                #print((light,)) for plotter
    time.sleep(0.05)
```

Pass your phone light over, then away from, the CPX light sensor.

Map Sensor Values to Outputs

Use the `map_range()` function to change (or map) the range of input values to the range of the output values. A variable stores the output values.

```
import time
from adafruit_circuitplayground.express import cpx
import simpleio

while True:
    tone = simpleio.map_range(cpx.light, 0, 320, 50, 5000)
    cpx.play_tone(tone,0.2)
    time.sleep(0.1)
```

0, 320 are light sensor values;

50, 5000 are the output values (tones)

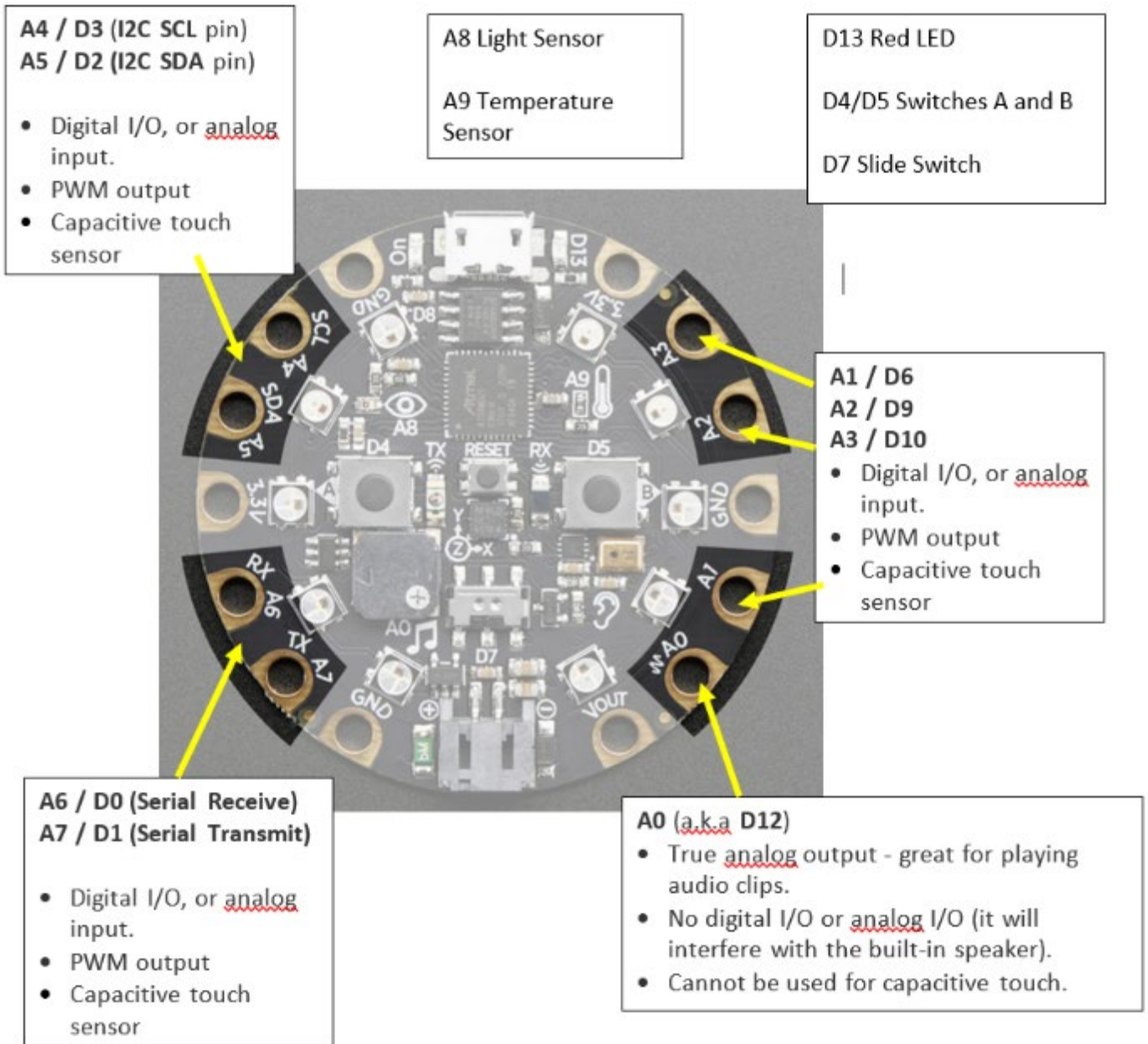
Conditional Statements (if-elif-else)

Conditional statements are decision statements. If a condition is True then a section of code is executed. If it is False, a different sequence of code is executed (or no code is executed).

```
while True:
    light = cpx.light
    if light < 50:
        cpx.pixels.fill((250, 250, 250))
    elif light < 100:
        cpx.pixels.fill((0, 0, 250))
    elif light < 150:
        cpx.pixels.fill((0, 250, 0))
    else:
        cpx.pixels.fill((250, 0, 0))
    time.sleep(0.1)
```

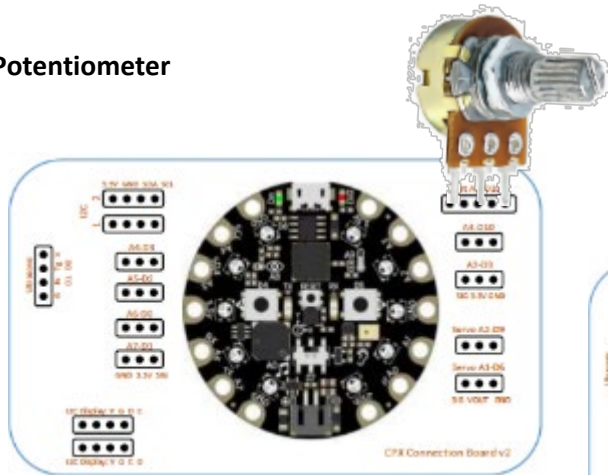
E. Connect Sensors

The CPX board has a combination of analog and digital pins.

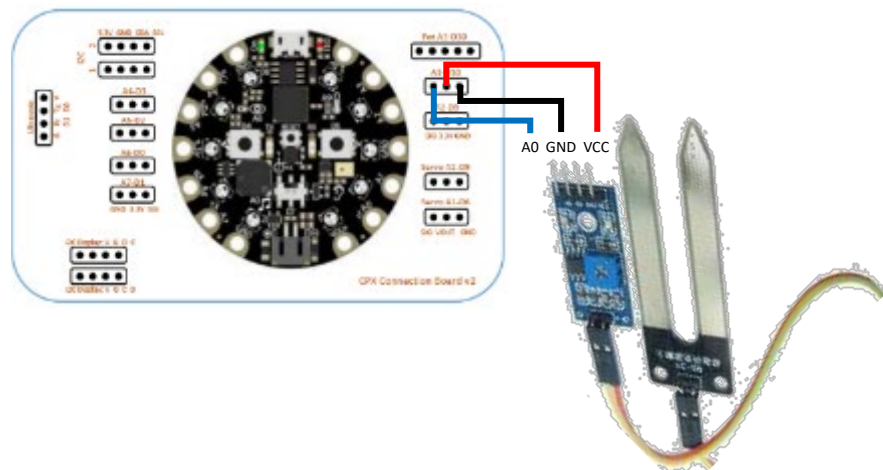


Analog Sensors (Potentiometers, Soil Moisture etc)

Potentiometer



Soil Moisture, Break Beam Sensor



```
#IMPORTS-----
import time
import board
from analogio import AnalogIn

#SETUP-----
analog_in = AnalogIn(board.A3)

#MAIN LOOP-----
while True:
    value = round(analog_in.value/65.536)
    print((value,))
    time.sleep(0.1)
```

Use Potentiometer to Change Pixel Brightness

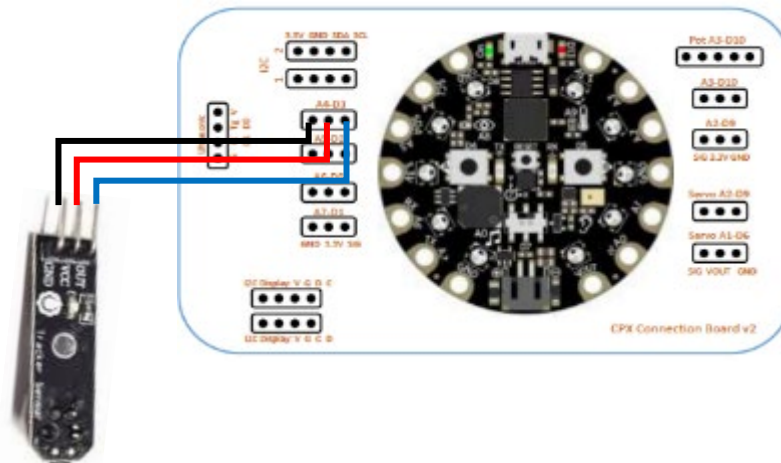
```
from adafruit_circuitplayground.express import cpx
import time
import board
import simpleio
from analogio import AnalogIn

analog_in = AnalogIn(board.A3)

while True:
    potvalue = round(analog_in.value/65.536)
    print((potvalue,))

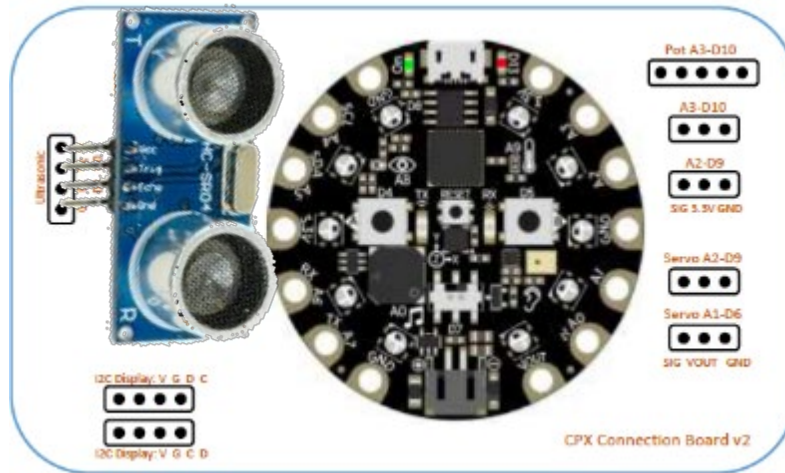
    bright = simpleio.map_range(potvalue, 0, 1000, 0, 1)
    cpx.pixels.brightness = bright
    cpx.pixels.fill((0,0,150))
    time.sleep(0.1)
```


Digital Sensors (e.g. Tracker, Collision Switch, Tilt Switch)



```
#IMPORTS-----  
import time  
import board  
from digitalio import DigitalInOut, Direction, Pull  
  
#SETUP-----  
sensor = DigitalInOut(board.D3)  
sensor.direction = Direction.INPUT  
sensor.pull = Pull.UP  
  
#MAIN LOOP-----  
while True:  
    value = sensor.value  
    print((value,))  
    time.sleep(0.1)
```

Ultrasonic Sensor



```
#IMPORTS-----
import time
import board
import adafruit_hcsr04

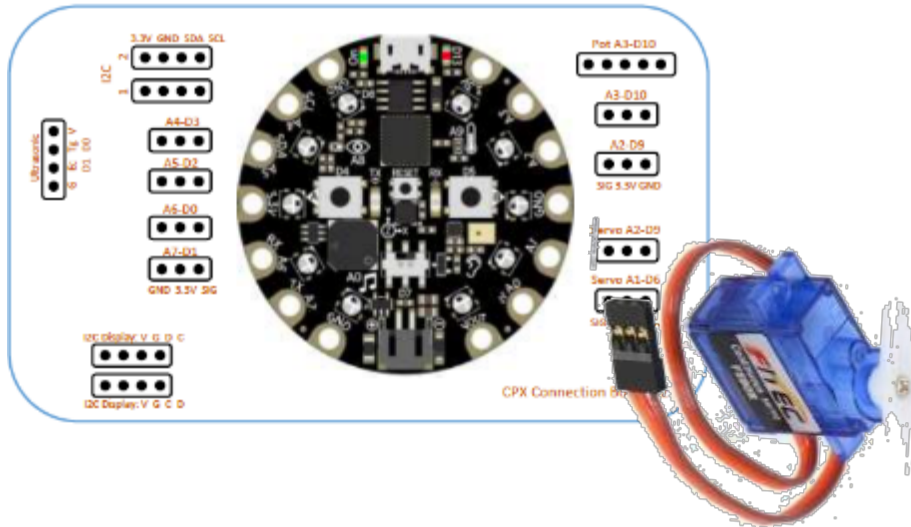
#SETUP-----
sonar = adafruit_hcsr04.HCSR04(trigger_pin=board.D0, echo_pin=board.D1)

#FUNCTIONS-----
def get_sonar_distance():
    d = 9999
    try:
        d = sonar.distance
        print((d,))
    except RuntimeError:
        print("Retrying!")
    return d

#MAIN LOOP-----
while True:
    distance = get_sonar_distance()
    time.sleep(0.1)
```

F. Connect Servos and Motors

Servos



Servo

0-180 angle of servo

Continuous Servo

0-85 Backward
90 Stop
95-180 Forward

```
#IMPORTS-----
import time
import board
import pulseio
from adafruit_motor import servo

#SETUP-----
pwm1 = pulseio.PWMOut(board.D6, duty_cycle=2**15, frequency=50)
servo1 = servo.Servo(pwm1)

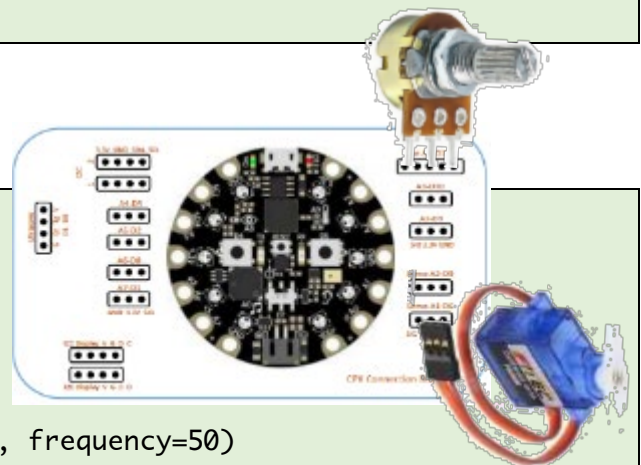
#MAIN LOOP-----
while True:
    servo1.angle = 10
    time.sleep(2)
    servo1.angle = 170
    time.sleep(2)
```

Use a Potentiometer to Move a Servo

```
import time
import board
from analogio import AnalogIn
import pulseio
from adafruit_motor import servo

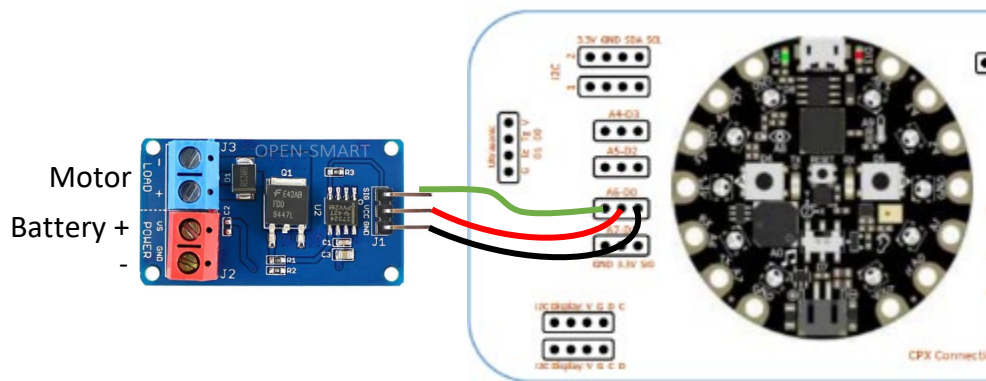
analog_in = AnalogIn(board.A3)
pwm1 = pulseio.PWMOut(board.D6, duty_cycle=2**15, frequency=50)
servo1 = servo.Servo(pwm1)

while True:
    value = round(analog_in.value/65.536)
    print((value,))
    servo1.angle = round(value/5.9)          #1000/180 = 5.6
    time.sleep(0.3)
```



Motors

Motors cannot be powered using the same power source as the CPX board and other sensors. A motor driver board (e.g. OPEN-SMART Single MOS Switch) is required to supply battery power to the motor, and control the motor.



Turn the motor on and off (full power)

```
#IMPORTS-----
import time
import board
from digitalio import DigitalInOut, Direction

#SETUP-----
motor = DigitalInOut(board.D0) #A6
motor.direction = Direction.OUTPUT

#MAIN LOOP-----
while True:
    motor.value = True
    time.sleep(2)
    motor.value = False
    time.sleep(2)
```

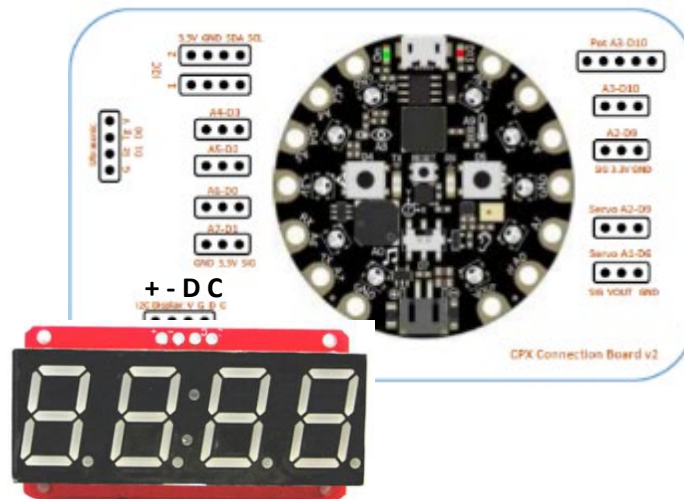
Control Power to the Motor (using PWM– D0, D6, D9, D10 only)

```
#IMPORTS-----
import time
import board
import pulseio

#SETUP-----
motor = pulseio.PWMOut(board.D0, frequency=5000, duty_cycle=0) #A6

#MAIN LOOP-----
while True:
    for i in range(40000,65001,5000):
        motor.duty_cycle = i
        time.sleep(2)
```

G. 7-Segment LED Display



```
#IMPORTS-----  
import time  
import board  
import busio  
from adafruit_ht16k33 import segments  
  
#SETUP-----  
i2c = busio.I2C(board.SCL, board.SDA)  
display = segments.Seg7x4(i2c)  
display.fill(0)  
  
#MAIN LOOP-----  
while True:  
    display.print(42)  
    time.sleep(2)  
    display[0] = '1'  
    display[1] = '2'  
    display[2] = 'A'  
    display[3] = 'B'  
    time.sleep(2)
```

H. Other Useful Functions

Use a Colour Wheel to Select 256 colours

```
def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if (pos < 0):
        return [0, 0, 0]
    if (pos > 255):
        return [0, 0, 0]
    if (pos < 85):
        return [int(pos * 3), int(255 - (pos*3)), 0]
    elif (pos < 170):
        pos -= 85
        return [int(255 - pos*3), 0, int(pos*3)]
    else:
        pos -= 170
        return [0, int(pos*3), int(255 - pos*3)]
```