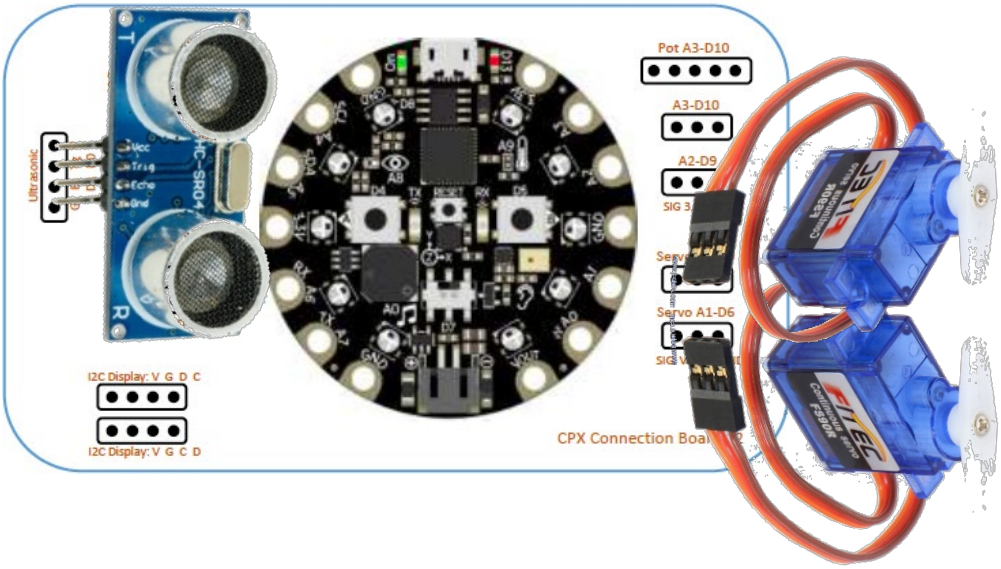
# Collision Avoidance Vehicle

CPX with Servos/Ultrasonic Sensor

Barry Butler
bbutl58@eq.edu.au

## A. CPX Connections

Connect the servo's and ultrasonic sensor to the CPX as shown.




## B. CircuitPython Basic Code

The following code is used for all CircuitPython CPX Code. Type this code into the Mu editor (in **CircuitPython mode**).

```
#IMPORTS----------------------------------------
from adafruit_circuitplayground.express import cpx
import time

#CONNECTIONS------------------------------------

#GLOBAL VARIABLES-------------------------------

#FUNCTIONS--------------------------------------

#PRESS BUTTON TO START--------------------------

# MAIN LOOP-------------------------------------
print('Main loop')
while True:

   time.sleep(0.1)
```

**EACH TIME YOU ENTER A SECTION OF CODE, CLICK THE "SAVE" BUTTON TO TEST IT, AND FIX ANY ERRORS.**

- **Select the CURCUITPY Folder**
- **Always Save Using the Filename – code.py**

Coolum State High School
Care • Respect • Excellence

ROBOCOAST
Sunshine Coast Robotics
www.robocoast.tech

## C. Import the Required Libraries

We want the vehicle to run two servos and the ultrasonic sensor, so we import all the libraries required.

```
#IMPORTS----------------------------------
from adafruit_circuitplayground.express import cpx
import time
import simpleio
import board
import pulseio
from adafruit_motor import servo
import adafruit_hcsr04
```

## D. Set the CPX Board Connections

We must set up the connections of the servos and ultrasonic sensor to the correct pins on the CPX.

```
#CONNECTIONS-----------------------------
pwmL = pulseio.PWMOut(board.D6, duty_cycle=0,frequency=50)
servoL = servo.Servo(pwmL)
pwmR = pulseio.PWMOut(board.D9, duty_cycle=0,frequency=50)
servoR = servo.Servo(pwmR)

sonar = adafruit_hcsr04.HCSR04(trigger_pin=board.D0, echo_pin=board.D1)
```

## E. Global Variables

Next, we create two global variables – one to store whether the vehicle is on or off (by pressing buttons A and B), and the other to store the ultrasonic sensor distance. We also set the brightness of the cpx neopixels.

```
#GLOBAL VARIABLES------------------------
vehicle_on = False
distance = 999
cpx.pixels.brightness = 0.2
```

## F. Read and Test the Buttons and Ultrasonic Sensor

Before we turn on the servos and move the vehicle, let's write the code to read the ultrasonic sensor and print the result. We will also write the code to read the button presses and turn the vehicle on and off.

```
#FUNCTIONS-------------------------------
def read_sonar():                          #read the ultrasonic sensor
    try:
        d = sonar.distance
    except RuntimeError:
        d = 999
    return d

def read_all_sensors(show_all):
    global distance, vehicle_on

    if cpx.button_a: vehicle_on = True      #read on/off buttons
    elif cpx.button_b: vehicle_on = False

    distance = read_sonar()                 #store the sonar distance
    if show_all: print(distance)
```

Then, test the code we have written in the main loop.

```
#MAIN LOOP-------------------------------
print('Main loop')
while True:
    read_all_sensors(True)

    if vehicle_on:                          #button A is pressed - turn on servos
        cpx.pixels.fill((0,20,0))
    else:                                   #button B is pressed - turn off servos
        cpx.pixels.fill((20,0,0))

    time.sleep(0.1)
```

Press the **Serial** button in Mu to check for any errors. The line number of the error will be shown. Be aware that the error may be at the end of the previous line (e.g. missing closing bracket).

If the code is correct, the ultrasonic sensor distance will be displayed in the serial output. The CPX neopixels should turn red and pressing button A will turn them green (vehicle on). Pressing button B will turn them back to red (vehicle off).

## G. Wait for a Button Press to Start

We don't want the servo's to start running as soon as we download the code the CPX. Sometimes, we might want a delay after pressing the button. So, write the code to check if CPX button A has been pressed.

```
#PRESS BUTTON TO START--------------------
print('Press button A to start')
cpx.pixels.fill((0,0,20))
while not cpx.button_a:
    time.sleep(0.1)
vehicle_on = True
```

## H. Make the Vehicle Move using Servos

The vehicle has two servos. We need to:

1. Write a function called *drive()* to run both servos at the same time
2. Call the *drive()* function in the main loop

**Write a function to run both servos at the same time**

You already have code in the functions section. Put the new code beneath the existing functions.

```
#FUNCTIONS---------------------------------------

def drive(lspeed, rspeed):     #values between 0 (off) and 90 (full on)
    if lspeed == 0: pwmL.duty_cycle = 0
    else:
        pwmL.duty_cycle = 2 ** 15
        servoL.angle = 90 + lspeed
    if rspeed == 0: pwmR.duty_cycle = 0
    else:
        pwmR.duty_cycle = 2 ** 15
        servoR.angle = 90 - rspeed
```

**Call the servo drive() function in the main loop**

```
#MAIN LOOP-------------------------------
print('Main loop')
while True:
    read_all_sensors(True)
    if vehicle_on:
        cpx.pixels.fill((0,20,0))
        drive(20,20)                      #drive forward
    else:
        cpx.pixels.fill((20,0,0))
        drive(0,0)                         #stop the vehicle

    time.sleep(0.1)
```

## I. Use the Ultrasonic Sensor Distance to Avoid Collisions

Now we must use the sensor distance to stop the vehicle hitting objects, and move in a different direction.

```
#MAIN LOOP-------------------------------
print('Main loop')
while True:
    read_all_sensors(True)
    if vehicle_on:
        cpx.pixels.fill((0,20,0))

        if distance <= 10:              #check if sonar distance <= 10cm:
            drive(0,0)                     #stop servos
            drive(0,20)                    #turn for 0.7 seconds
            time.sleep(0.7)
            drive(20,20)                   #drive forward

        elif distance < 999:            #if sonar distance > 10 cm:
            drive(20,20)                   #drive forward

    else:
        cpx.pixels.fill((20,0,0))
        drive(0,0)

    time.sleep(0.1)
```

**Try:**

1. **Drive in a square or figure of 8**
2. **Locate an object and drive to it**
3. **Light signals when turning left or right using individual pixels** e.g. cpx.pixels[0] = (0,0,100)
4. **Slow down when getting closer to an object**