Lab 3: External Sensors for Arduino with CircuitPython

A. Trahan, P.E.

October 20, 2019

Introduction

At this point, we have covered most of the important concepts and setup for designing and developing on Arduino with Circuit Python. This lab focuses on using external sensors (input devices), which is very similar to using external actuators. As with actuators, the CPX has many built-in sensors, but there may be additional sensors (like flow meters) that need to be attached for a given project. In addition, most other Arduino devices don't have as many built-in sensors, so all sensors for a project must be attached externally. We will then provide examples combining internal and external sensors and actuators.

In this lab we will:

1. Connect an external sensor to the CPX
2. Try examples using combinations of internal and external sensors and actuators

Required materials (per group):

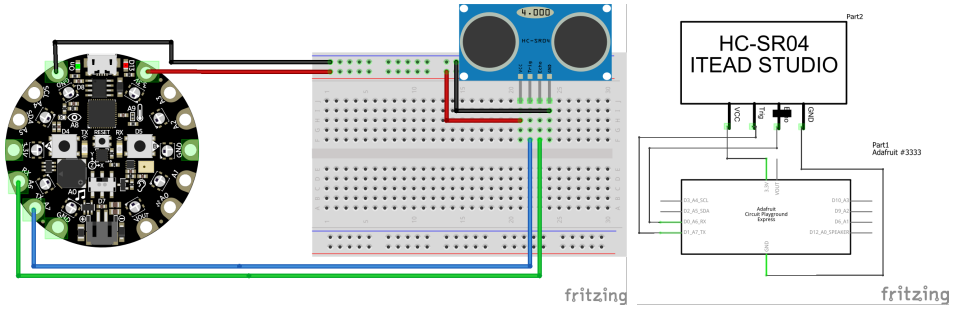
- (1) Circuit Playground Express (CPX)
- (1) Micro USB cable
- (1) Breadboard
- (1) Ultrasonic Proximity Sensor (RCWL-1601 or similar)

- (1) Servo (SG90)
- Assorted jumpers and alligator clips

1 Connecting an External Sensor to the CPX

As discussed in the previous lab, the CPX has a variety of "pads" for inputs and outputs. The CPX is a 3.3V Arduino device, so we use the RCWL-1601 proximity sensor (Figure **??**), which is a 3.3V version of the older HC-SR04 proximity sensor. That background is only necessary because it explains why the python package for the proximity sensor is called `HCSR04` and why it appears in the schematics. The RCWL-1601 has four pins: power in, ground, trigger (ultrasonic ping out), and echo (ultrasonic ping in). To attach it to the CPX:

1. Attach the proximity sensor to a breadboard, using four rows
2. Attach the **3.3V** and **GND** pads to the power and ground columns of a breadboard and use jumpers to attach those to the VCC and GND rows for the proximity sensor
3. Attach pad **A7** to the TRIG row for the proximity sensor
4. Attach pad **A6** to the ECHO row for the proximity sensor



(a) Breadboard layout (b) Circuit diagram

Figure 1: Diagrams for attaching external proximity sensor to CPX

5. Enter the code below in Mu, and save to the CPX as `code.py`, and note the following:

(a) We import `HCSR04` to control the proximity sensor. This requires an external library - `adafruit_hcsr04.mpy` - which must be copied from the library bundle (see Lab 2) onto the CPX.

(b) Sometimes the sensor has trouble getting a signal and throws a RuntimeError, so a `try/except` statement is used to catch this and continue the loop.

6. Open the Serial console or the Plotter pane to see the distance measured from the sensor (cm)

```
#-- ExternalProximitySensor.py - CircuitPython code for CPX --#

# Import Section
import board
from adafruit_hcsr04 import HCSR04
from time import sleep

# Setup Section
sonar = HCSR04(trigger_pin=board.A7, echo_pin=board.A6)

# Function Section

# Loop Section
while True:
    try:
        print((sonar.distance,))
    except RuntimeError:
        print("Retrying!")
    sleep(0.5)
```

2 Bringing it Together

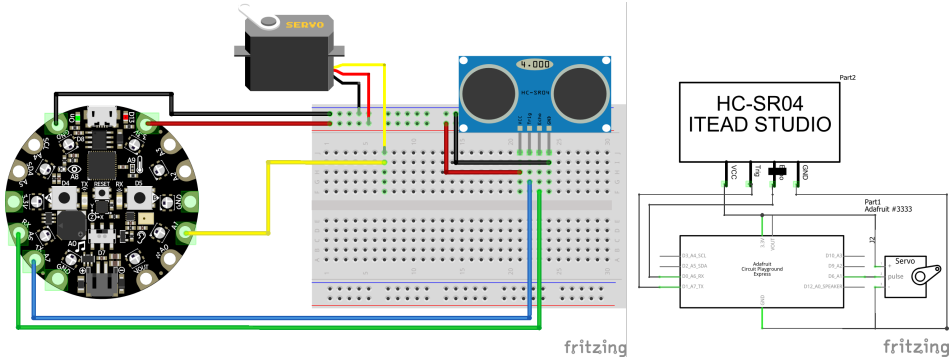
With a firm grasp of how to attach and control internal and external actuators and sensors with the CPX, it's time to try combining those elements and designing your own projects. Here are a few more examples.

2.1 Add a needle gauge to the proximity sensor

This example combines the proximity sensor from this lab with the light gauge from Lab 2. (*external sensor*, *external actuator*)

1. Connect the proximity sensor and servo to the CPX as shown in Figure 2.

2. Enter the code below in Mu, and save to the CPX as `code.py`. Note how it directly combines two previous examples, except for the function mapping proximity to servo position.



(a) Breadboard layout (b) Circuit diagram

Figure 2: Diagrams for attaching external proximity sensor and servo to CPX

```
#-- ExternalProximitySensorWithOutputs.py - CircuitPython code for CPX --#

# Import Section
import board
import pulseio
from adafruit_circuitplayground.express import cpx
from adafruit_hcsr04 import HCSR04
import adafruit_motor.servo
from time import sleep

# Setup Section
```

```
sonar = HCSR04(trigger_pin=board.A7, echo_pin=board.A6)
pwm = pulseio.PWMOut(board.A1, frequency=50)
servo = adafruit_motor.servo.Servo(pwm, min_pulse=750, max_pulse=2600)

# Function Section
def servo_pos(dist):
    dist_min = 5  # Low distance servo  [cm]
    dist_max = 50 # High distance servo [cm]
    dist = min((max((dist, dist_min)), dist_max))
    return (((dist-dist_min)/(dist_max-dist_min)) * 180)

# Loop Section
while True:
    try:
        servo.angle = servo_pos(sonar.distance)
        print((sonar.distance, servo.angle))
    except RuntimeError:
        print("Retrying!")
    sleep(0.5)
```

2.2 Add warning lights/sounds to the proximity sensor

This example combines the proximity sensor from this lab with the neopixels on the CPX from Lab 1 and the internal buzzer. (*external sensor*, *internal actuator*)

1. Connect the proximity sensor to the CPX as shown in Figure 1.
2. Enter the code below in Mu, and save to the CPX as `code.py`, noting:
   (a) The helper function used to flip the neopixels on and off
   (b) The string of conditionals used to respond to different proximity, which needs to be in reverse order
   (c) The use of `t` and `dt` to blink or buzz at different frequencies from the loop frequency

```
#-- ExternalProximitySensorWithWarnings.py - CircuitPython code for CPX --#
```

```
# Import Section
import board
from adafruit_circuitplayground.express import cpx
from adafruit_hcsr04 import HCSR04
from time import sleep

# Setup Section
led_brightness = 0.25
t = 0
dt = 0.25
sonar = HCSR04(trigger_pin=board.A7, echo_pin=board.A6)
cpx.pixels.fill((255,0,0))
cpx.pixels.brightness = 0

# Function Section
def pixel_flip():
    if cpx.pixels.brightness > 0:
        cpx.pixels.brightness = 0
    else:
        cpx.pixels.brightness = led_brightness

# Loop Section
while True:
    try:
        d = sonar.distance

        # Closer than 15 cm is Dangerously Close
        if d <=5:
            pixel_flip()
            if t >= 1.0:
                cpx.play_tone(440, 0.25)
                t = 0

        # Closer than 15 cm is Very Close
        elif d <= 15:
            pixel_flip()

        # Closer than 30 cm is Close
```

```
        elif d <= 30:
            if t >= 1.0:
                pixel_flip()
                t = 0

        # Farther than 25 cm is Safe
        else:
            cpx.pixels.brightness = 0

        print((d,))
        t+=dt
    except RuntimeError:
        print("Retrying!")
    sleep(dt)
```