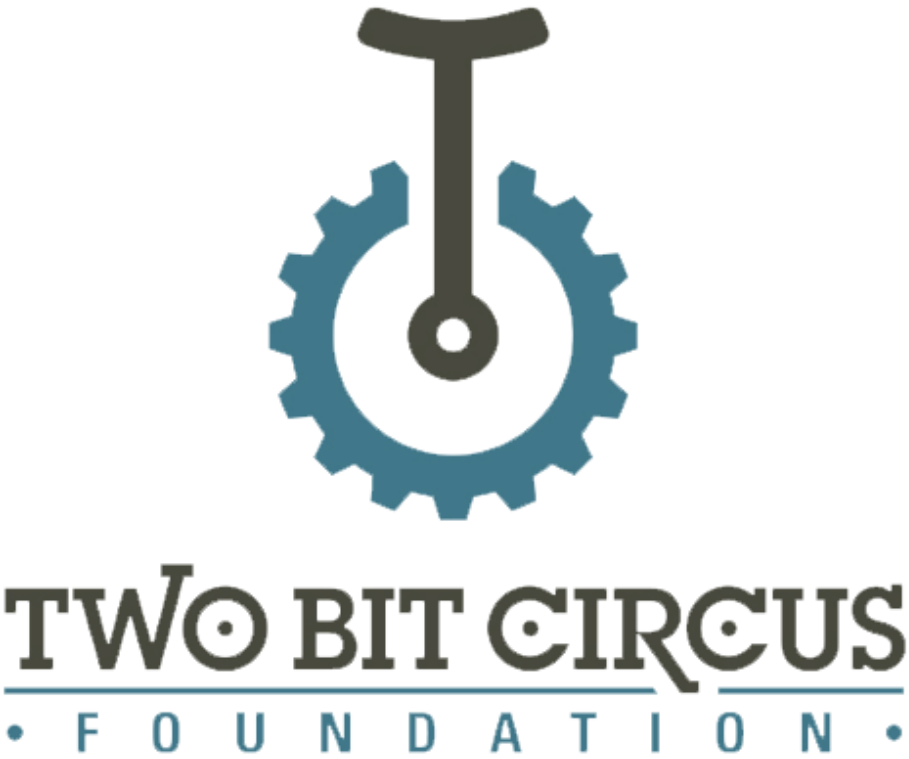


Water Gun Game
Alt Controller Setup Guide

Written by Aubrey Isaacman

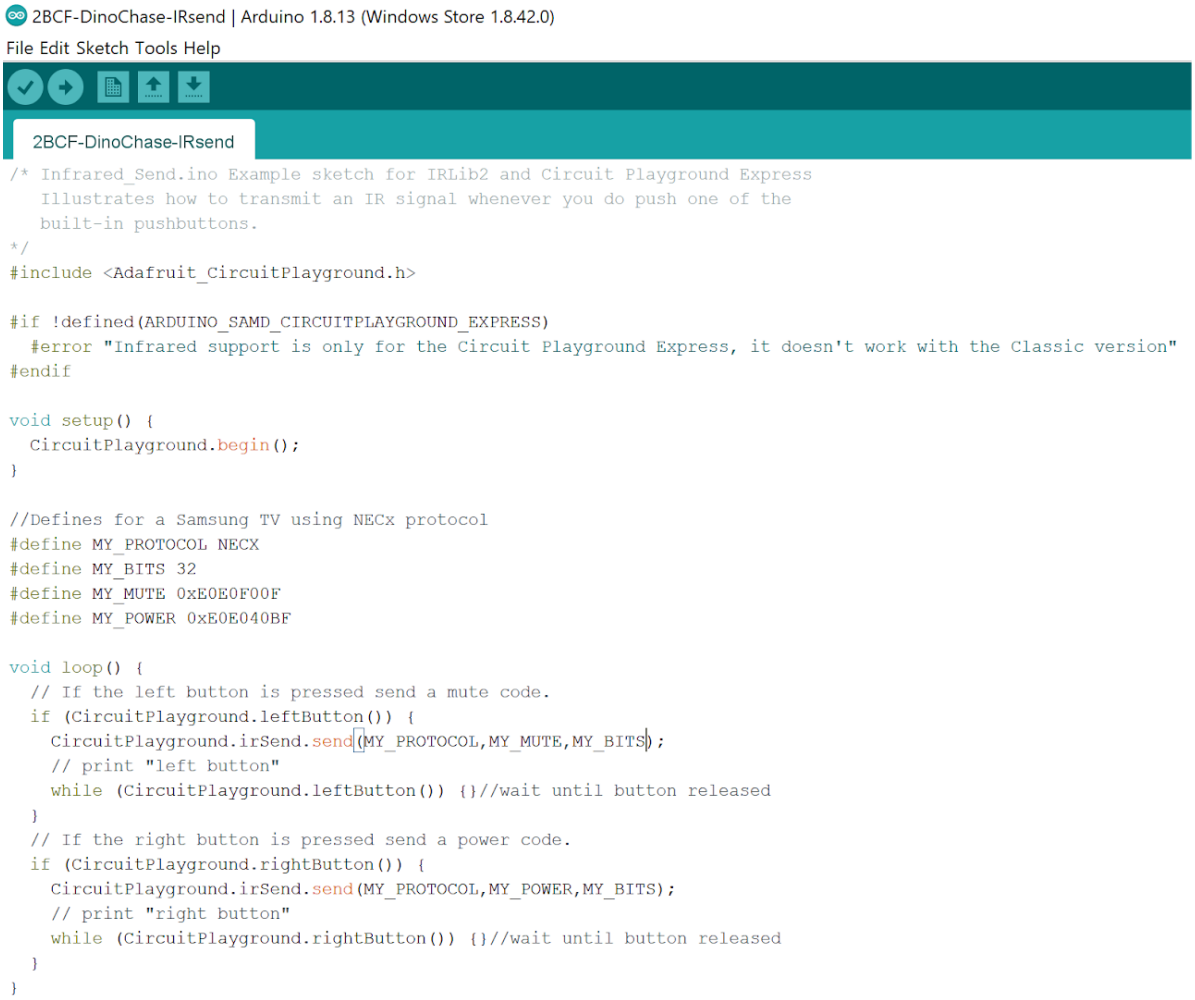
Setup

1. Install the Ardity Unity asset to communicate between the Adafruit Circuit Playground Express (aka CPX) and Unity. Follow the directions of their setup guide first. It can be found in the install package and will show up in the “Ardity” folder in your “Assets folder” after you install it.

2. You will need **two** CPX. One will need a battery pack on it; this one will be the player’s. The other one will be the target. It will be connected to the computer USB port. It will read input from the player and send that data to Unity via Ardity.

3. Follow this tutorial by Adafruit to set up your Arduino IDE with CPX.

4. Connect the CPX that will be used by the player to your computer USB port and upload the following code. It can also be found by going to File -> Examples -> Adafruit Circuit Playground -> Infrared Demos




```
/* Infrared_Send.ino Example sketch for IRLib2 and Circuit Playground Express
   Illustrates how to transmit an IR signal whenever you do push one of the
   built-in pushbuttons.
*/
#include <Adafruit_CircuitPlayground.h>

#if !defined(ARDUINO_SAMD_CIRCUITPLAYGROUND_EXPRESS)
  #error "Infrared support is only for the Circuit Playground Express, it doesn't work with the Classic version"
#endif

void setup() {
  CircuitPlayground.begin();
}

//Defines for a Samsung TV using NECx protocol
#define MY_PROTOCOL NECX
#define MY_BITS 32
#define MY_MUTE 0xE0E0F00F
#define MY_POWER 0xE0E040BF

void loop() {
  // If the left button is pressed send a mute code.
  if (CircuitPlayground.leftButton()) {
    CircuitPlayground.irSend.send(MY_PROTOCOL,MY_MUTE,MY_BITS);
    // print "left button"
    while (CircuitPlayground.leftButton()) {}//wait until button released
  }
  // If the right button is pressed send a power code.
  if (CircuitPlayground.rightButton()) {
    CircuitPlayground.irSend.send(MY_PROTOCOL,MY_POWER,MY_BITS);
    // print "right button"
    while (CircuitPlayground.rightButton()) {}//wait until button released
  }
}
```

5. Disconnect this CPX and plug it into a battery pack. Then, plug in the other CPX (the target) and upload the following code to it. This code can be found in the same place as the previous code. This CPX will remain plugged into your computer.

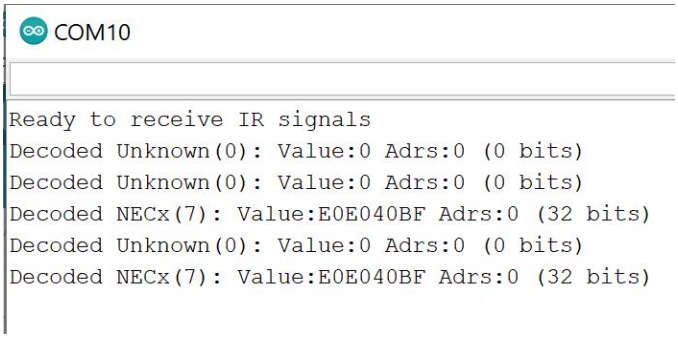
2BCF-DinoChase-IRread | Arduino 1.8.13 (Windows Store 1.8.42.0)

File Edit Sketch Tools Help

2BCF-DinoChase-IRread

```
/* Infrared_Read.ino Example sketch for IRLib2 and Circuit Playground Express
   Illustrates how to receive an IR signal, decode it and print
   information about it to the serial monitor.
*/
#include <Adafruit_CircuitPlayground.h>

#if !defined(ARDUINO_SAMD_CIRCUITPLAYGROUND_EXPRESS)
  #error "Infrared support is only for the Circuit Playground Express, it doesn't work with the Classic version"
#endif


void setup() {
  CircuitPlayground.begin();
  Serial.begin(9600);

  while (!Serial); // Wait until serial console is opened

  CircuitPlayground.irReceiver.enableIRIn(); // Start the receiver
  Serial.println("Ready to receive IR signals");
}

void loop() {
  //Continue looping until you get a complete signal received
  if (CircuitPlayground.irReceiver.getResults()) {
    CircuitPlayground.irDecoder.decode();           //Decode it
    CircuitPlayground.irDecoder.dumpResults(false);  //Now print results. Use false for less detail
    CircuitPlayground.irReceiver.enableIRIn();      //Restart receiver
  }

  // close the serial port
  Serial.end();
}
```

6. Open the Serial Monitor under the “Tools” menu. Turn on the battery pack and aim the player’s CPX at the target CPX. Press the left button, then press the right button. It should look something like this.



7. Open your Unity project and open the “PlayerMovementAltCtrl” script. Scroll down to the Update function

PlayerMovementAltCtrl.cs

C: > Users > Aubs > OneDrive > Documents > Two Bit > Summer 2020 > DinoChase > Assets > Scripts > PlayerMovementAltCtrl.cs

```
51      // Update is called once per frame
52      void Update()
53      {
54          // receive data
55          string message = serialController.ReadSerialMessage();
56          if (message == null)
57          {
58              return;
59          }
60
61          // Check if the message is plain data or a connect/disconnect event.
62          if (ReferenceEquals(message, SerialController.SERIAL_DEVICE_CONNECTED))
63              Debug.Log("Connection established");
64          else if (ReferenceEquals(message, SerialController.SERIAL_DEVICE_DISCONNECTED))
65              Debug.Log("Connection attempt failed or disconnection detected");
66          else
67              Debug.Log("Message arrived: " + message);
68
69
70          // player moves up on left button press on CPX
71
72          if(message == "Decoded Unknown(0): Value:0 Adrs:0 (0 bits) ")
73          {
74              moveUp = gameObject.transform.position;
75
76              moveUp.y += moveUp.y * -speed * Time.deltaTime;
77              gameObject.transform.position = moveUp;
78
79          }
80
81          // players moves down on right button press on CPX
82          if(message == "Decoded NECx(7): Value:E0E040BF Adrs:0 (32 bits)")
83          {
84              moveUp.y += moveUp.y * speed * Time.deltaTime;
85              gameObject.transform.position = moveUp;
86          }
87
88      }
89
```

8. Go to your Serial Monitor in Arduino and copy the message that printed when you pressed the left button. Paste that into the if statement on line 72. Then, go back to your Serial Monitor and copy the message that printed when you pressed the right button. Paste that into the if statement on line 81.

9. Now, hit play and test your controller!

To Do

- Add sound effects
- Add background music
- Replace temporary sprites and backgrounds
- General polish/clean up

Attributes

- All sprites are from flaticon.com
- All background photos are Adobe Stock Images

Resources

- Adafruit Learning System, Infrared
- Adafruit, Infrared Receive and Send
- Adafruit, Infrared Transmit and Receive