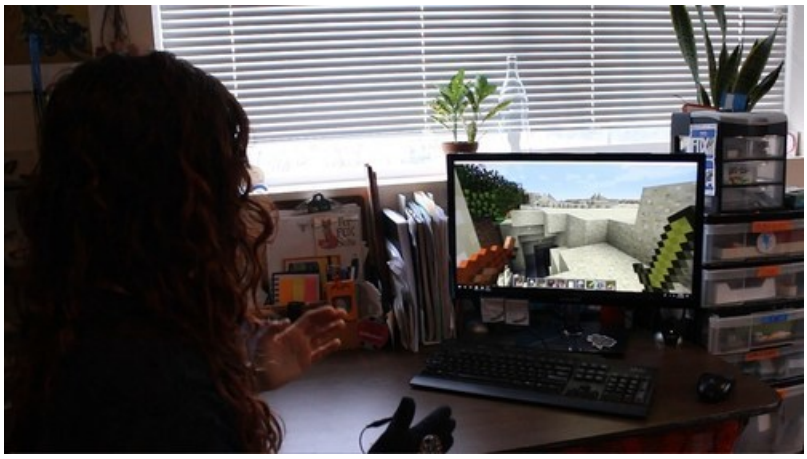
adafruit learning system

CircuitPlayground Minecraft Gesture Controller

Created by Jen Fox



Last updated on 2018-08-22 04:03:44 PM UTC

Guide Contents

Overview

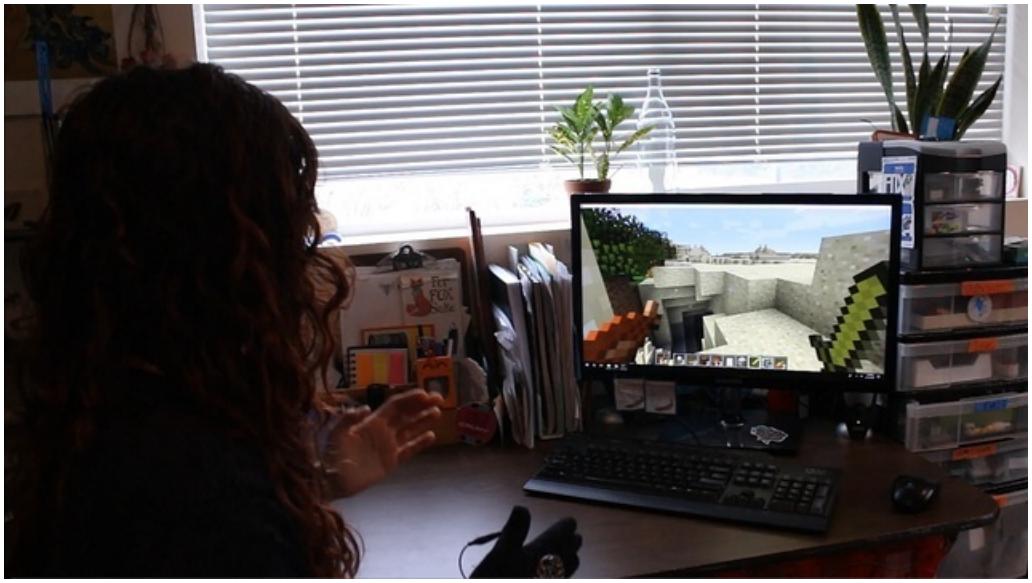
Move your body to play Minecraft! What!! Yes. Check the video for a demo :)

This tutorial will show you how to make your very own gesture game controller for Minecraft (or your other fav. computer game). Move your hand(s) to walk/run/jump, look around, and attack* all the things!

Let's get started! Grab yourself a Circuit Playground Express (https://adafru.it/wpF), snag my program code (https://adafru.it/Btb), and get shakin' to play Minecraft in (srsly) the most fun way ever! :D

- **Read time**: 20 min
- **Build Time**: ~ 2 hours
- **Cost**: ~$30

**It is a biiiiit tricky to attack moving things (like monsters), so be careful in survival mode! Or use this to challenge your skills :)*

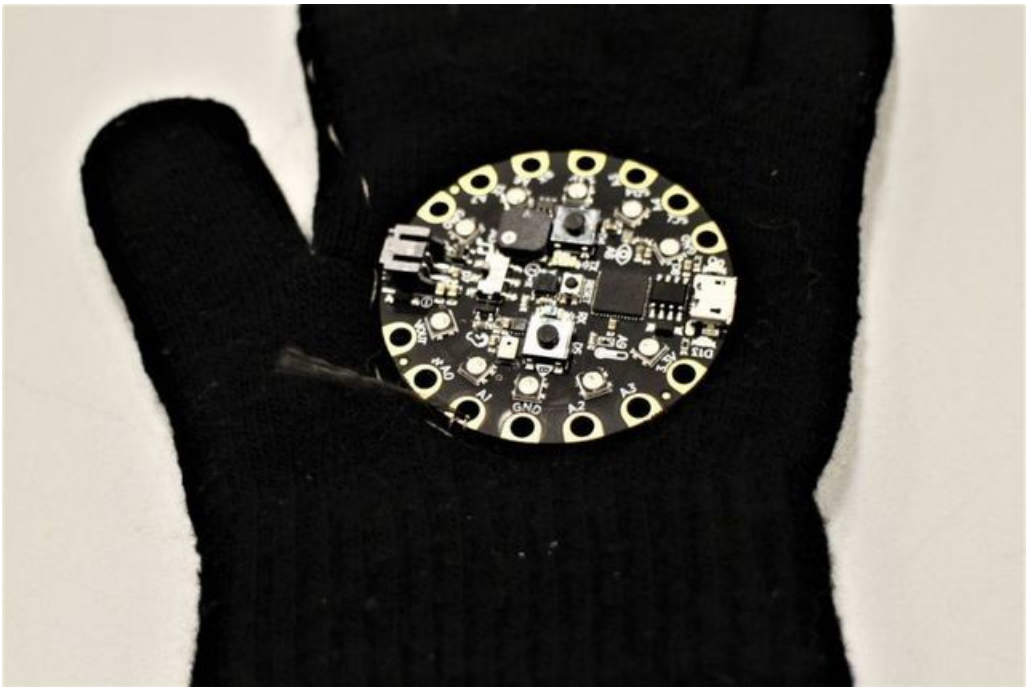


Helpful Background Info

To keep things short as possible, set up your Circuit Playground Express to program it in CircuitPython, add libraries, and use the Serial Monitor.

If you're like "what are those words", here are some tutorials to get ya started!

1. Setting up the Circuit Playground Express (https://adafru.it/AFI)
2. Installing (all the) libraries for CPX (https://adafru.it/C9M)
3. Using the Serial Monitor (https://adafru.it/CgR)
4. More info on CircuitPython! (https://adafru.it/CgS)



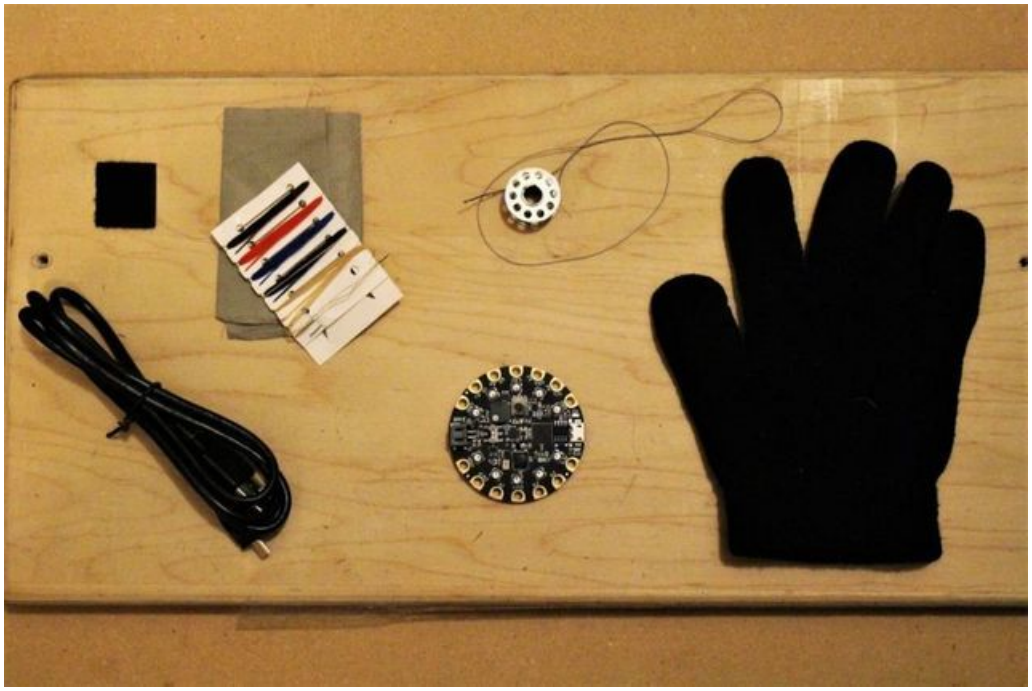
Materials

Materials

- Circuit Playground Express (https://adafru.it/wpF) (FYI: gonna call this the "CPX" to save typing)
- MicroUSB to USB cable (https://adafru.it/iia)
- **Glove** -- use a thick glove or one with multiple layers (to avoid shorting the conductive thread)
- Conductive Fabric (https://adafru.it/yBm) (~ 6 in. x 6 in.)
- Conductive Thread (https://adafru.it/dME) (~ 24 in.)
- Regular Thread (~ 24 in.)
- Velcro Strips (two 1 in. x 1 in.)

Tools

- Sewing Needle
- Scissors



Build the Glove Controller! (Pt. 1)

You can make the gesture controller without the glove, but the glove controller makes it easier to play, keeps the CPX in the same orientation (very important), and means you can use your fingers as added controls!

1. Cut rectangles of conductive fabric for the finger pads (~ 0.5 in. x 1 in.).

2. Use regular thread to sew the conductive fabric pads onto each of the glove fingers.

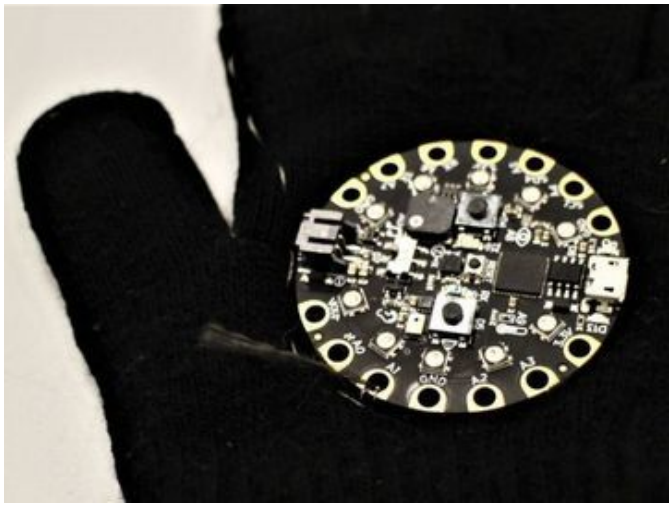
Suggested to use a highlighter or other pen to avoid sewing the two sides of the glove together (learn from my mistakes bbies).

3. Attach CPX to the glove with velcro squares.

Build the Glove Controller! (Pt. 2)

Use an alligator clip or insulated wire to connect the CPX ground ("GND") to the thumb pad.

Stitch conductive thread from the CPX capacitive touch pads (A1, A2, A3 & A4) to each of the four fingers.



Stitch conductive thread from the CPX capacitive touch pads (**A1**, **A2**, **A3** & **A4**) to each of the four fingers.

If you have a multimeter, check continuity between the CPX pins and the conductive thread pads.

Plan out your Controller!

First! What do we need to do to control Minecraft (or another awesome game)?

This is a super helpful & fun lesson in Design Thinking, but you can skip this if you want to just use my controls. You can always come back here later if you want to make changes later :D

Determine (crucial) game controls.

Note: Start simple! Figure out the most important controls for the game and start there. You can always add more later.

Here are the controls that I wanted to use while playing Minecraft.. in creative mode :) (you can use the same ones or customize your own controller!):

Movement:

- Walk forward: **W key**
- Run: **Ctrl** + **W**
- Jump: **Space bar**
- Look Left & Right: **Mouse rotate**
- Walk backward: **S key**

Actions:

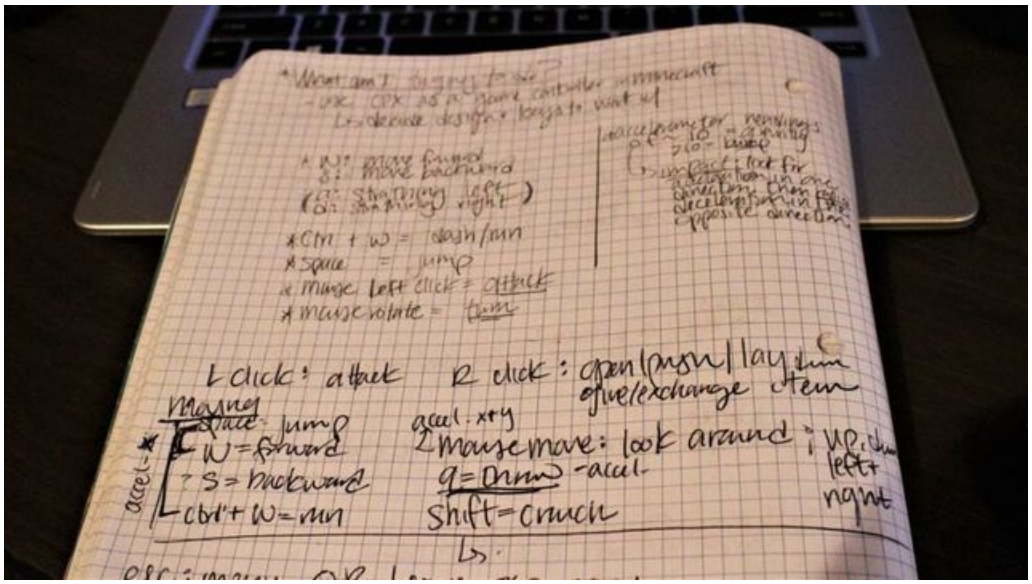
- Attack: **Mouse Left Click**
- Place Block/Push/Open: **Mouse Right Click**
- Inventory: **E key**
- Escape: **ESC key**

Decide how you want to use gestures and/or the finger pads to trigger these controls. Recommended to sketch out your plan.

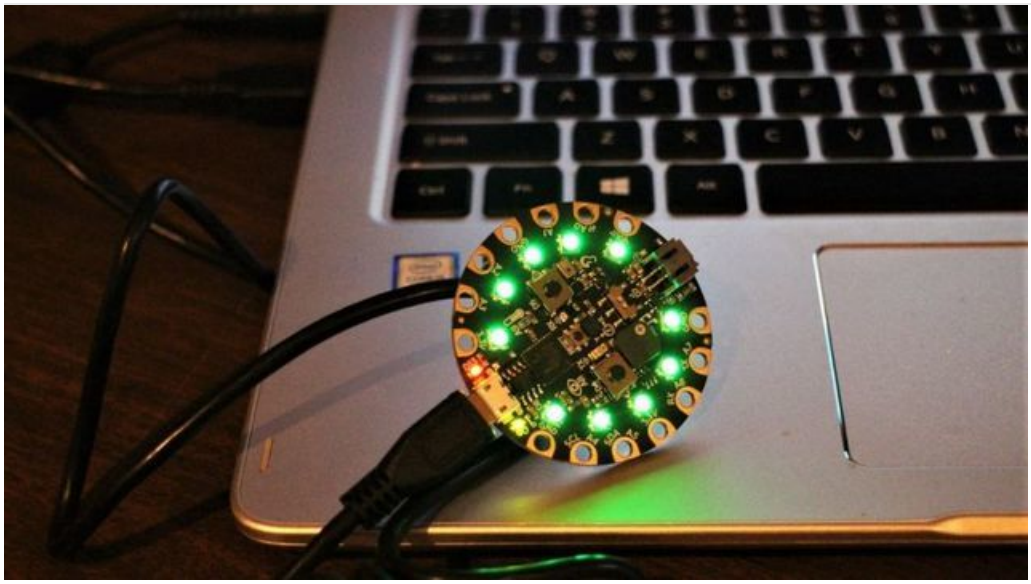
Here is my design thought process:

> *I've always wanted to feel like I was actually *in* a game, so I went the "cheap VR" route and used gestures to control basic movements. For walking, I went the "let's move my arms like I'm walking" route, which easily transitioned into running and jumping by increasing the speed of motion.*
>
> *To make it easy to place a block or exchange items, I decided to use an "awkward handshake" motion.*
>
> *Turning was a bit of a challenge, but my goal was to be able to look around by moving my hands in the direction I wanted to look.*
>
> *Attack became the pointer finger pad, inventory the middle finger pad (which I ended up removing), Escape the ring finger pad, and the pinky finger pad to let me to walk backwards.*

Again, you can keep these same controls or design your own :D


L click: attack
R click:
Moving
Space: jump
W = forward
S = backward
Ctrl + W = run
accel: x+y
Mouse move: look around
q = Dmm
Shift = crouch

Program the Circuit Playground Express!



The CPX has an on-board compiler, which means you can program it in (pretty much) any language you want! I opted for CircuitPython, a version of Python for microcontrollers, 'cause Python is awesome.

Here's the GitHub repository (https://adafru.it/Btg) that has the full code. The same code is embeded below, simply click the **Download** link to save as **main.py** on your CIRCUITPY drive on the CPX

```
# Minecraft Gesture Controller
#
# Written by <jenfoxbot@gmail.com>
# MIT License V.asof2018
# Also coffee/beer ware license :)
# Super awesome thanks to:
# Richard Albritton, Tony DiCola, John Parker
# All the awesome people who wrote the libraries

# Libraries


import time

# Libraries for accelerometer
import adafruit_lis3dh
import board
import busio
# General purpose libraries
import touchio
# Libraries for HID Keyboard & Mouse
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.keycode import Keycode
from adafruit_hid.mouse import Mouse

# CPX Setup
touch_a1 = touchio.TouchIn(board.A1)
touch_a1.threshold = 2000
touch_a2 = touchio.TouchIn(board.A2)
```

```
touch_a2.threshold = 2000
touch_a3 = touchio.TouchIn(board.A3)
touch_a3.threshold = 2000
touch_a4 = touchio.TouchIn(board.A4)
touch_a4.threshold = 2000

# Keyboard & Mouse Setup

# The keyboard object!
kbd = Keyboard()
# we're americans :)
layout = KeyboardLayoutUS(kbd)
# The mouse object!
mouse = Mouse()

# Accelerometer Setup

# Initialize Accelerometer
i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c, address=25)
# Set range of accelerometer
# (can be RANGE_2_G, RANGE_4_G, RANGE_8_G or RANGE_16_G).
lis3dh.range = adafruit_lis3dh.RANGE_8_G


# Controller Functions

# A helper to 'remap' an input range to an output range
def Map(x, in_min, in_max, out_min, out_max):
    return int(
        (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min
    )


def Move(upDown_axis, isBackward):
    axis_new = abs(upDown_axis)
    # If you are touching A4, walk backwards, else walk forwards
    if isBackward:
        print("backwards")  # Debugging
        if axis_new > 1.2:  # walk threshold
            if axis_new > 2.5:  # run threshold
                kbd.press(Keycode.LEFT_CONTROL, Keycode.S)
                time.sleep(0.1)
                kbd.release_all()
            else:
                kbd.press(Keycode.S)
                time.sleep(0.1)
            kbd.release_all()
    else:
        if axis_new > 1.2:  # walk threshold
            if axis_new > 2.5:  # run threshold
                kbd.press(Keycode.LEFT_CONTROL)
                time.sleep(0.1)
                kbd.release_all()
            else:
                kbd.press(Keycode.W)
                time.sleep(0.1)
            kbd.release_all()
```

```
def Turn(upDown_axis, leftRight_axis, lookUp):
    leftRight_adj = int(leftRight_axis)  # currently z_axis
    upDown_adj = int(upDown_axis)  # currently y_axis

    leftRight_new = Map(leftRight_adj, -3, 3, -100, 100)
    if lookUp and abs(upDown_adj) < 1.2:
        upDown_new = Map(upDown_adj, -1, 1, -100, 100)
    else:
        upDown_new = 0
    if abs(leftRight_new) < 127:
        mouse.move(-leftRight_new, upDown_new)
    else:
        mouse.move(0, 0)


def Jump(upDown_axis):
    upDown = abs(upDown_axis)
    if upDown > 3:
        kbd.press(Keycode.SPACE, Keycode.W)
    kbd.release_all()


def Give(upDown_axis, frontBack_axis):
    frontBack_new = abs(frontBack_axis)
    if abs(upDown_axis) < 1:
        if frontBack_new > 2:
            print("give")
            mouse.click(Mouse.RIGHT_BUTTON)
            mouse.release_all()


def Attack():
    """Attack! By clicking the left mouse button"""
    print("attack")
    mouse.click(Mouse.LEFT_BUTTON)
    time.sleep(0.1)
    mouse.release_all()


def Inventory():
    """Open up inventory, press the E keyboard key"""
    print("inventory")  # Debugging -- view in serial monitor
    kbd.press(Keycode.E)
    time.sleep(0.01)
    kbd.release_all()


def ESC():
    """Escape by pressing the ESCape key"""
    print("ESC")  # Debugging -- view in serial monitor
    kbd.press(Keycode.ESCAPE)
    time.sleep(0.01)
    kbd.release_all()


def readAxes():
    """Convert acceleration from m/s^2 to G, with a delay"""
    x, y, z = lis3dh.acceleration
    time.sleep(0.01)
```

```
    return (x / 9.806, y / 9.806, z / 9.806)  # 9.806 m/s^2 per G


# Main Function
while True:
    # Read accelerometer values (in G).  Returns a 3-tuple of x, y, x axis
    pos_x, pos_y, pos_z = readAxes()

    # Read finger pads and act accordingly
    if touch_a1.value:
        Attack()

    if touch_a2.value:
        Inventory()

    if touch_a3.value:
        ESC()

    is_backward = touch_a4.value
    look_up = touch_a4.value

    # Run through the motions! .. literally :)
    Move(pos_y, is_backward)
    Turn(pos_y, pos_z, look_up)
    Jump(pos_y)
    Give(pos_y, pos_x)

    # Small delay to keep things responsive but
    # give time for interrupt processing.
    time.sleep(0.01)

    # Debugging Ahead!!
    # Use the following 2 lines to figure out which
    # axis is upDown, frontBack, or LeftRight
    # and also for debugging!
    # print('x = {}G, y = {}G, z = {}G'.format(x, y, z))
    # time.sleep(0.3)
```

Keep readin' if you want to understand how the program works (definitely suggested) or if you want to modify the code.

Required Libraries

To do the things we want with our controller, we need the following CircuitPython libraries. All are available in the bundle, you can learn how to install it here (https://adafru.it/C9M).

- **LIS3DH accelerometer**
    - This allows us to use motion to trigger various things.
- **Human Interface Device ("HID") keyboard**
    - This library allows us to control the keyboard!
- **HID mouse**
    - This library means we can control the mouse!

- **CPX capacitive touch**
    - This library lets us use the capacitive touch feature on the CPX, hooray!
- A couple of other libraries to make our lives easier: **time**, **busio**, and **board**.

```
#########################
##        Libraries     ##
#########################
import touchio
import board
import busio
import time
#Libraries for HID Keyboard & Mouse
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.mouse import Mouse
#Libraries for accelerometer
import adafruit_lis3dh
```

Configure and initialize the libraries

Assign variables for the keyboard, mouse, and accelerometer objects. Select a range for the accelerometer.

```
######################
#     CPX Setup      #
######################
touch_a1 = touchio.TouchIn(board.A1)
touch_a1.threshold = 2000
touch_a2 = touchio.TouchIn(board.A2)
touch_a2.threshold = 2000
touch_a3 = touchio.TouchIn(board.A3)
touch_a3.threshold = 2000
touch_a4 = touchio.TouchIn(board.A4)
touch_a4.threshold = 2000

############################
# Keyboard & Mouse Setup   #
############################
# The keyboard object!
kbd = Keyboard()
# we're americans :)
layout = KeyboardLayoutUS(kbd)
#The mouse object!
mouse = Mouse()

######################
# Accelerometer Setup #
######################
#Initialize Accelerometer
i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c, address=25)
# Set range of accelerometer (can be RANGE_2_G, RANGE_4_G, RANGE_8_G or RANGE_16_G).
lis3dh.range = adafruit_lis3dh.RANGE_8_G
```

Write short functions for each of the controls

The motion controls can be tricky. We did some initial testing with the accelerometer by printing the values in a serial monitor (in the source code, go to the 'main loop' `while True:` section and uncomment the two debugging lines). This will help you to determine thresholds for walking, running and jumping, looking left and right, and placing objects.

The touch pad triggers are much easier as you are only looking for a capacitive trigger (`True` / `False`) when you read the `touch_an.value`

Remember to release all of the keyboard and mouse keys at the end of each function! Otherwise you'll end up with stuck keys, in which case you can always unplug the USB cable

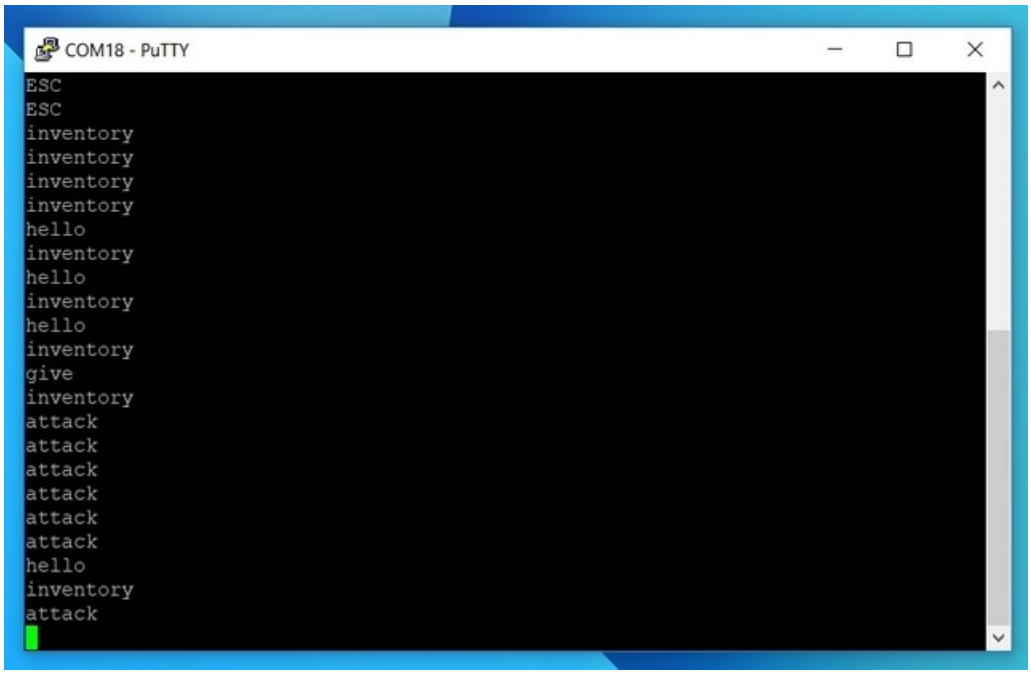
Test & Adjust

Don't forget you have to load the program onto the CPX by dragging and dropping the python file onto the **CIRCUITPY** drive, then rename the file as `code.py` or `main.py` (and back up your previous program first)

Like pretty much every project, this one will likely be a little wonky when you first get it running. If the touch pads are acting strange, reset the CPX (this recalibrates the capacitive input pins) by clicking the **RESET** mini button

Test #1

1. Open up the serial monitor with Mu (https://adafru.it/CgR) (or PuTTY or any other serial monitor) and run the program (CTRL + D)
2. Test each of the movement controls (you'll see the mouse moving on the screen and make sure the program doesn't crash as well as the touch pads (which should display relevant text on the serial monitor).



Test #2

Deploy in Minecraft creative mode! Test the movement and action controls to see if anything breaks or doesn't work as expected (plz keep in mind that this is a prototype!)

Update the program based on your testing. Remember, it's OK if it's not perfect, there's always time to make it better! :)

Run Wild!

You're ready to run through Minecraft!! Just be wary of monsters, it might be a bit tricky to protect yourself.. dun dun dunnnnn!!

Supplementing your gesture controller with a keyboard is a good idea if you want play for reals.

Please like and/or leave a comment if you enjoyed the tutorial! And of course, let me know if you have any comments or questions!
Happy Building!

<3, jenfoxbot