**How To Use: Controllers**

For this project, we are using an Arduino to control the joy cons. Today I’m going show you how to do a basic set up of the breadboard and later we will get into the code.

Everything we will need-

1 breadboard

2 thumb sticks

1 Arduino Leonardo pro micro

Dupont connectors

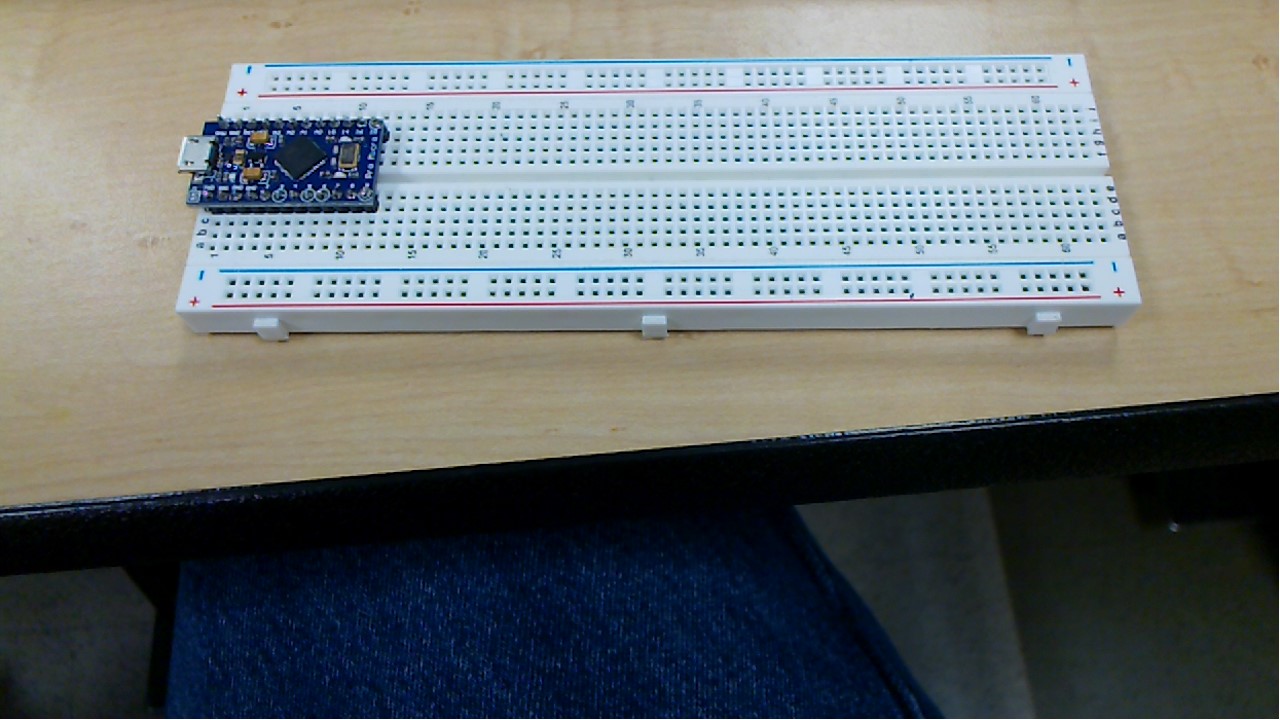
Micro USB cable

Get your breadboard and thumb sticks out and place them on your work surface.

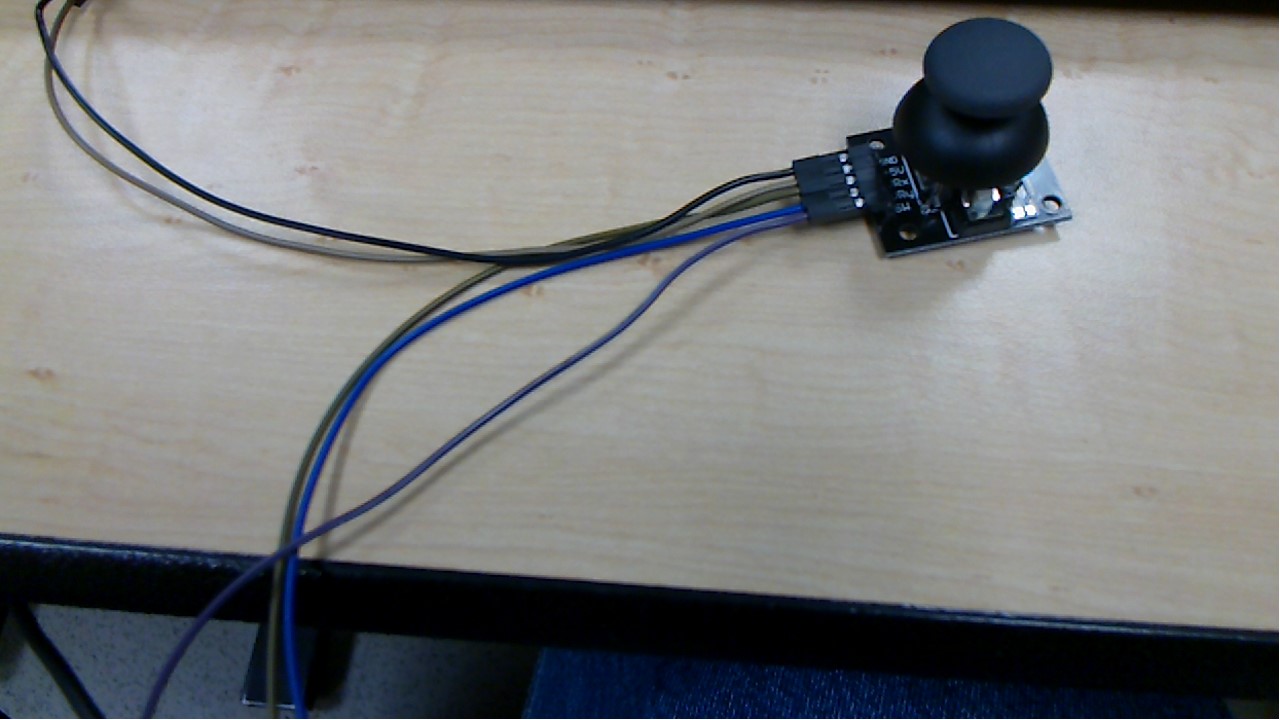
#picture of everything we will need#

On both thumb sticks you will notice they have pins with words next to them. GND is ground which grounds the pins. +5V gives the thumb stick 5 volts of power. VRx and VRy are the x and y axis respectively. SW is the pin that sends out the actual data that the thumb switch generates.

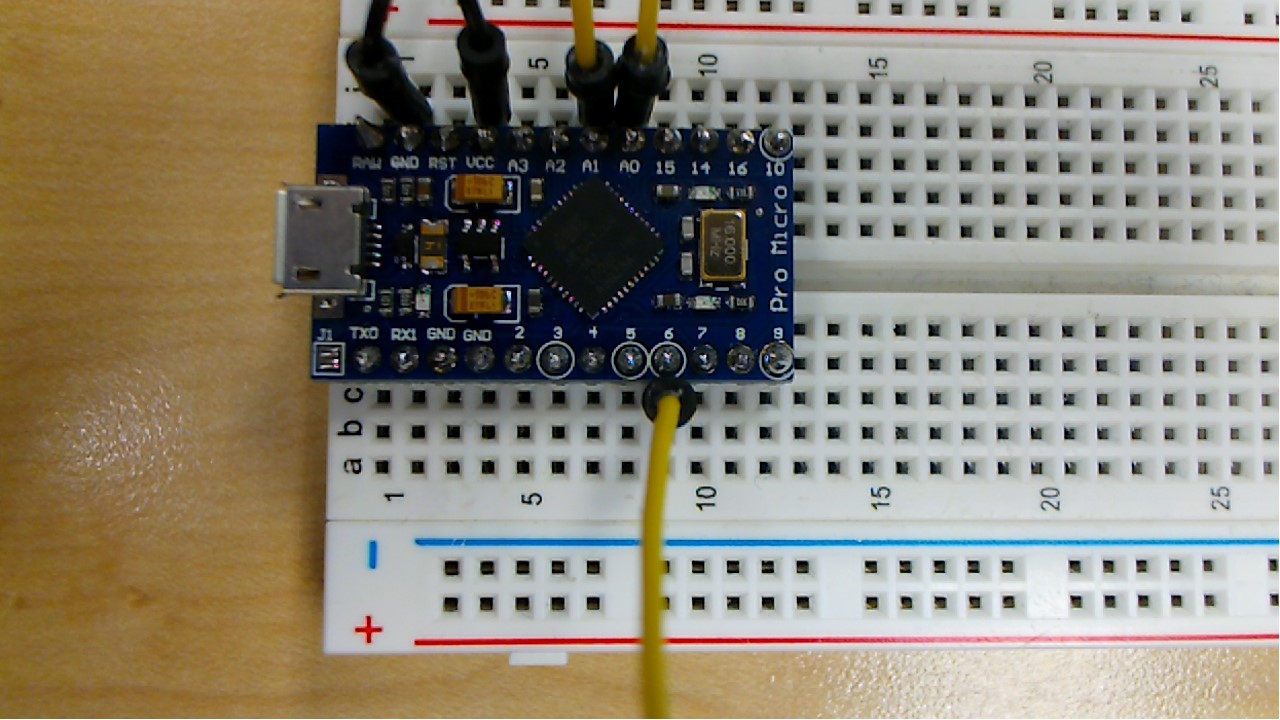
Step 1- At the end of the breadboard, insert the arduino Leonardo into the board. The charging port needs to face away from the board. The board should also be in the center of the ditch as possible.



Step 2- on every pin on the thumb switches, insert a wire that has a single pinout and a single pin in. you will need 10 wires to complete this step.



Step 3- on the breadboard, place a double sided pinout wire in pins GND at the top of the arduino, VCC at the top of the arduino, and a pinout wire on pin 6 at the bottom of the board. Two more wires need to be inserted into the A0 and A1 ports. I have the other ends of the wires under my keyboard to make it easier to take a picture.



Step 4- obtain another double pinout wire and insert one end into negative side of the breadboard.

#picture of new pinout ground wire#

Step 5- The ground wire that is not plugged in will be plugged into hole 40 on the breadboard. A1 will be plugged into hole 35. A0 will be plugged into hole 37. VCC will be plugged into hole 45. Pin 6 will be plugged into hole 50. Remember which pin is which on the board, we will need to know that later.

#picture of all double sided wires connected to the board.#

Step 6- Grab one of the thumb sticks with the wires coming out of it. When looking at the switch, the wires should be coming out of the left side of the switch. Now we are going to plug the switch into the breadboard. Ground goes to the bottom of hole 40, +5V goes into the bottom of hole 45, VRx goes into the bottom of hole 27, VRy goes into the bottom of hole 34, SW goes into the bottom of hole 50. If you are using both thumb sticks the other wires will plug into the holes below the ones you just inserted.

#step6.jpg#

Step 7- open the Arduino IDE on your workstation. The following code we did not create ourselves so all claims go to the original creator. Plug the micro USB cable into the end of the Arduino that faces the outside of the breadboard, plug the other end of the cable into the USB slot in your workstation. You can also find the code in the Code folder on the repo. Copy the following code into the Arduino IDE.

\*step7-1.jpg\*

|  |
| --- |
| /\*  JoystickMouseControl  Controls the mouse from a joystick on an Arduino Leonardo, Micro or Due.  Uses a pushbutton to turn on and off mouse control, and a second pushbutton  to click the left mouse button.  Hardware:  - 2-axis joystick connected to pins A0 and A1  - pushbuttons connected to pin D2 and D3  The mouse movement is always relative. This sketch reads two analog inputs  that range from 0 to 1023 (or less on either end) and translates them into  ranges of -6 to 6.  The sketch assumes that the joystick resting values are around the middle of  the range, but that they vary within a threshold.  WARNING: When you use the Mouse.move() command, the Arduino takes over your  mouse! Make sure you have control before you use the command. This sketch  includes a pushbutton to toggle the mouse control state, so you can turn on  and off mouse control.  created 15 Sep 2011  updated 28 Mar 2012  by Tom Igoe  This example code is in the public domain.  https://www.arduino.cc/en/Tutorial/BuiltInExamples/JoystickMouseControl  \*/  #include "Mouse.h"  // set pin numbers for switch, joystick axes, and LED:  const int switchPin = 2; // switch to turn on and off mouse control  const int mouseButton = 3; // input pin for the mouse pushButton  const int xAxis = A0; // joystick X axis  const int yAxis = A1; // joystick Y axis  const int ledPin = 5; // Mouse control LED  // parameters for reading the joystick:  int range = 12; // output range of X or Y movement  int responseDelay = 5; // response delay of the mouse, in ms  int threshold = range / 4; // resting threshold  int center = range / 2; // resting position value  bool mouseIsActive = false; // whether or not to control the mouse  int lastSwitchState = LOW; // previous switch state  void setup() {  pinMode(switchPin, INPUT); // the switch pin  pinMode(ledPin, OUTPUT); // the LED pin  // take control of the mouse:  Mouse.begin();  }  void loop() {  // read the switch:  int switchState = digitalRead(switchPin);  // if it's changed and it's high, toggle the mouse state:  if (switchState != lastSwitchState) {  if (switchState == HIGH) {  mouseIsActive = !mouseIsActive;  // turn on LED to indicate mouse state:  digitalWrite(ledPin, mouseIsActive);  }  }  // save switch state for next comparison:  lastSwitchState = switchState;  // read and scale the two axes:  int xReading = readAxis(A0);  int yReading = readAxis(A1);  // if the mouse control state is active, move the mouse:  if (mouseIsActive) {  Mouse.move(xReading, yReading, 0);  }  // read the mouse button and click or not click:  // if the mouse button is pressed:  if (digitalRead(mouseButton) == HIGH) {  // if the mouse is not pressed, press it:  if (!Mouse.isPressed(MOUSE\_LEFT)) {  Mouse.press(MOUSE\_LEFT);  }  }  // else the mouse button is not pressed:  else {  // if the mouse is pressed, release it:  if (Mouse.isPressed(MOUSE\_LEFT)) {  Mouse.release(MOUSE\_LEFT);  }  }  delay(responseDelay);  }  /\*  reads an axis (0 or 1 for x or y) and scales the analog input range to a range  from 0 to <range>  \*/  int readAxis(int thisAxis) {  // read the analog input:  int reading = analogRead(thisAxis);  // map the reading from the analog input range to the output range:  reading = map(reading, 0, 1023, 0, range);  // if the output reading is outside from the rest position threshold, use it:  int distance = reading - center;  if (abs(distance) < threshold) {  distance = 0;  }  // return the distance for this axis:  return distance;  } |

Next you will go up to Tools, select Board, select Arduino Micro. Next you will go back to Tools, select Port, select whichever Port has Arduino Micro in the name.

\*step7-2.jpg\*