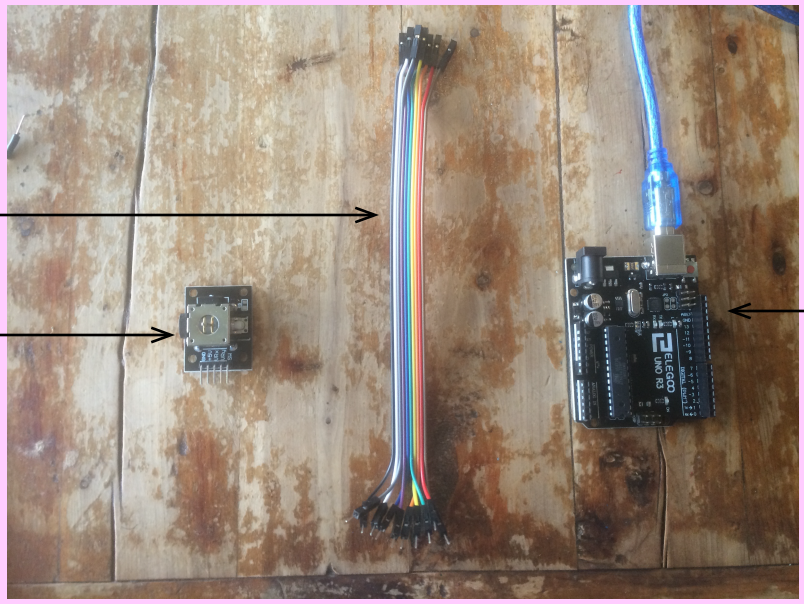
Joystick

By Elie
for Grayson Earle's
Art Tech 2018

Materials:

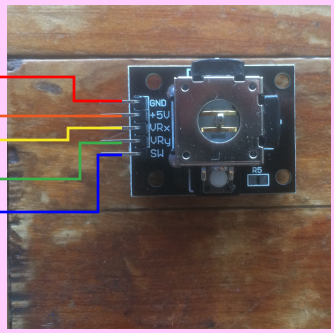
(1) Arduino
(1) Joystick Module
(5) Jumper cables with a pin on one end and a slot for a pin on the other

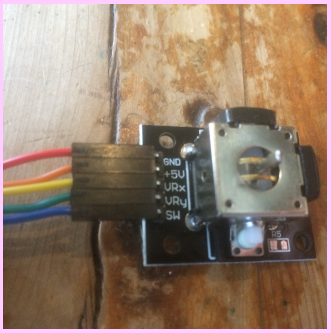


****

The Joystick Module has 5 pins:

1. Ground
2. 5v
3. X value
4. Y value
5. SW (switch)



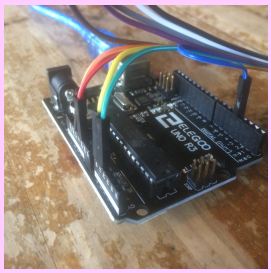


There is a Push-Button Momentary Switch (SW) built into the Joystick (push in the joystick). The Joystick still works without the Switch connected. If you do not need it, just dont hook up the SW pin.

****

GND -- goes to Ground pin
+5v -- goes to +5v pin
VRx -- go to an ANALOG pin on the Arduino
VRy -- go to an ANALOG pin on the Arduino
SW -- needs to go to a DIGITAL pin

SW needs a 10k< ohm Pull-up resistor. To see how to do that, go to the next page



Now simply analogRead( ) the analog pins to get the values of the Joystick. See my posted code to test if things are working right! You'll probably have to map( ) or scale in some way the input, since top-left is (0,0) - like in Processing.

TO SEE HOW TO HOOK UP THE SWITCH (SW) see next page

Joystick (SW)

The SW pin on the Joystick Module is connected internally to GND. In order to get the switch to read, you must have the Digital Pin it go to have a 10k (or more) “pull-up resistor”. Luckily, Arduinos have built in pull-up and pull-down resistors.
You can add one with code by writing “pinMode(4,INPUT_PULLUP)” instead of “pinMode(4,INPUT)”.

IMPORTANT

This makes it so that when the Button is pressed, it returns a 0 and when it’s not pressed it returns a 1

Pull-up Resistor Reference

Pullup Resistors with pins configured as INPUT

Often it is useful to steer an input pin to a known state if no input is present. This can be done by adding a pullup resistor (to +5V), or a pulldown resistor (resistor to ground) on the input. A 10K resistor is a good value for a pullup or pulldown resistor.

Properties of Pins Configured as INPUT_PULLUP

There are 20K pullup resistors built into the Atmega chip that can be accessed from software. These built-in pullup resistors are accessed by setting the pinMode() as INPUT_PULLUP. This effectively inverts the behavior of the INPUT mode, where HIGH means the sensor is off, and LOW means the sensor is on.
When connecting a sensor to a pin configured with INPUT_PULLUP, the other end should be connected to ground. In the case of a simple switch, this causes the pin to read HIGH when the switch is open, and LOW when the switch is pressed

NOTE: Digital pin 13 is harder to use as a digital input than the other digital pins because it has an LED and resistor attached to it that's soldered to the board on most boards. If you enable its internal 20k pull-up resistor, it will hang at around 1.7V instead of the expected 5V because the onboard LED and series resistor pull the voltage level down, meaning it always returns LOW. If you must use pin 13 as a digital input, set its pinMode() to INPUT and use an external pull down resistor.

Taken from:
https://www.arduino.cc/en/Tutorial/DigitalPins

THE CODE --

Joystick_Module_-_elie §

```
int x = 0; //choose the ANALOG pin that is hooked up to the X value (pin VRx)from the Joystick Module
int y = 1; //choose the ANALOG pin that is hooked up to the Y value (pin VRy) from the Joystick Module
int sw = 4; // choose the DIGITAL pin that is hooked up to the switch (pin SW)

//Note: Like in Processing - The Joystick's
//Top-left is (0,0)... Bottom-right is (maxValue,maxValue)
//when joystick is pressed in - it returns a 0

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

  pinMode(sw,INPUT_PULLUP);

}

void loop() {

  //Create a variable that reads the values coming from the Joystick
  //they must be FLOATs because the Joystick is ANALOG and returns a continuous spectrum of values
  float xVal = analogRead(x);
  float yVal = analogRead(y);
  //now an INT to see the button state
  int swVal = digitalRead(sw);

  //Print the values to the console
  Serial.print("X: ");                                    //name these whatever you want!
  Serial.println(xVal); // println to go to the next line
  Serial.print("Y: ");
  Serial.println(yVal);

  //now the button
  //you'll notice that it is "1" when off, and "0" when pressed.
  //This is bc of the Pullup resistor! Make sure to remember this when writing your code!
  Serial.print("Button: ");
  Serial.println(swVal);

  //print a BLANK LINE (a return) in order to keep the data readable
  Serial.print("\n");

  //Slow it down so it's readable in our console
  delay(500);

}
```