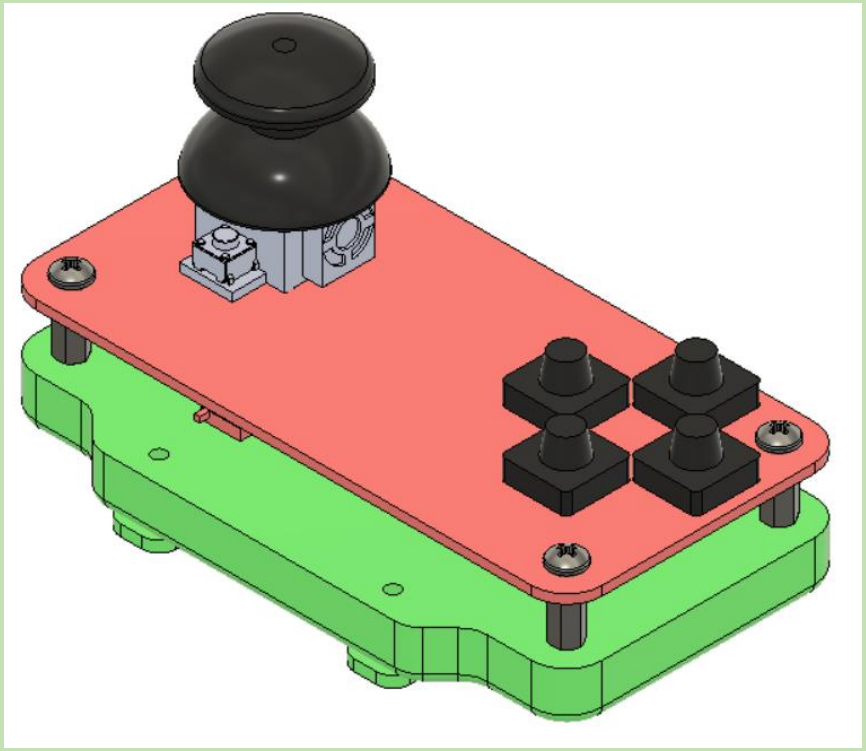
SparkFun Wireless Joystick Kit – "Generic" Code Project & Case



This Document Contains

- List of stuff to buy from sparkfun to get a sample project running with the Wireless Joystick Kit.
- High Level of overview of the code. I am assuming you are familiar with:
    - ❑ Arduino's and their programming -> https://www.arduino.cc/en/Main/Products
    - ❑ Bit manipulation -> https://playground.arduino.cc/Code/BitMath
    - ❑ Xbee Radio -> https://learn.sparkfun.com/tutorials/xbee-shield-hookup-guide
    - ❑ Sparkfun wireless Joystick Kit specs -> https://www.sparkfun.com/products/14051
- A Guide on making a 3D printed 'body' for the joystick kit.

Presentation created by Richard Firth 11/3/17

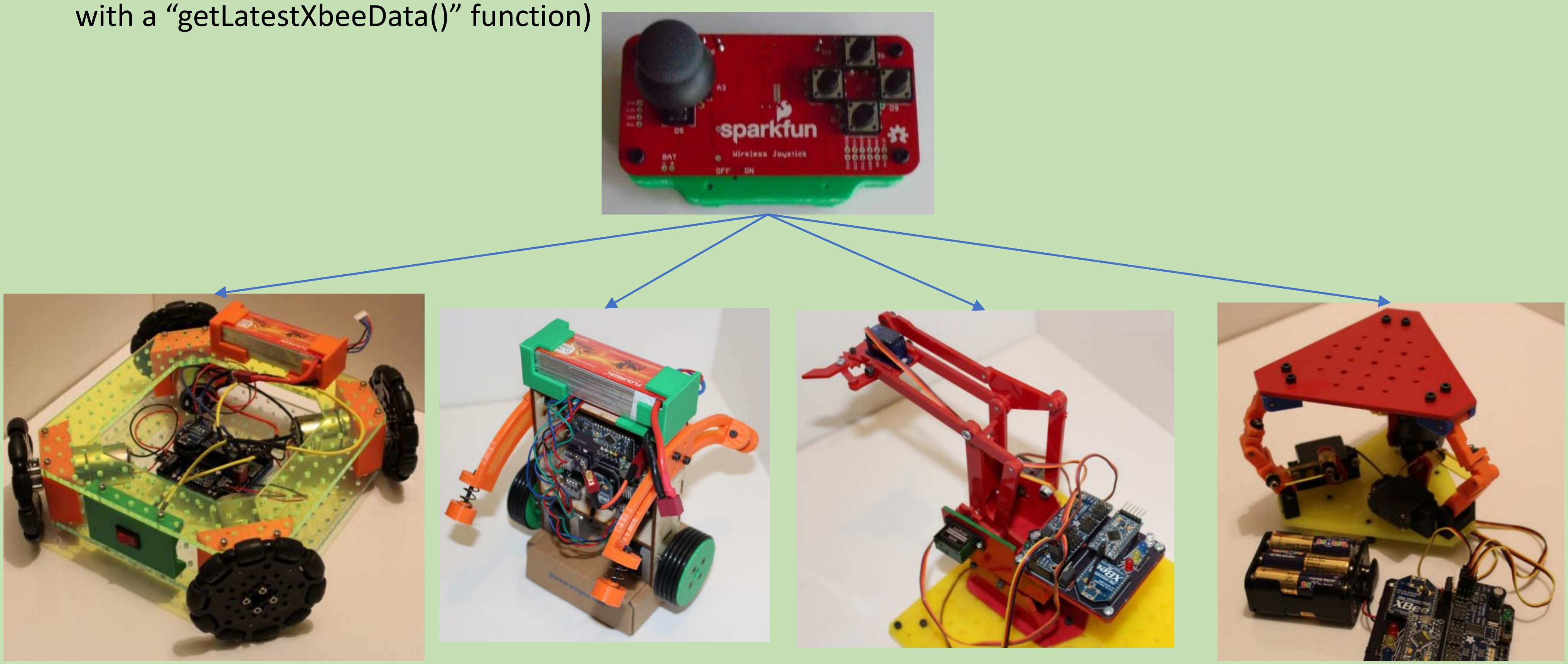
https://grabcad.com/library/wjk_body-1 <- CAD Files
https://github.com/richardFirth/WirelessJoystick <-Code

DISCLAIMER
- All the code started from open source example code that I modified, (I didn't create it from scratch.)
- I'm intending to put this presentation In a github repository and link to it from a youtube video. All relevant files will be in the repository.
- I'm not sponsored or affiliated with anyone or anything mentioned.
- All stuff I do is provided without any warranty or promise of anything, use at your own risk.

Background & Goal

I bought this Wireless Joystick Kit from spark fun, and I think it’s a really great product. It’s generally useful, and I often find myself using it with many of my different projects.

- Goal is to program the Joystick so that it broadcasts the state of the buttons and joystick in a way that can be used by any project programmed to receive the output. (Meaning I don’t have to change the code on joystick between controlling different things)
- Also have a generic “template” that Other projects can start from. (meaning I want to copypaste an Arduino tab with a “getLatestXbeeData()” function)



Bill of materials (electronics)

Picc	Desc (hyperlinked to order page)	QTY	Price (ea)
	XBee 1mW Wire Antenna - Series 1 (802.15.4)	2	$24.95
	SparkFun Wireless Joystick Kit	1	$34.95
	SparkFun XBee Explorer USB	1	$24.95
	Lithium Ion Battery - 1Ah	1	$9.95
	SparkFun XBee Shield (for your Arduino project)	1	$14.95

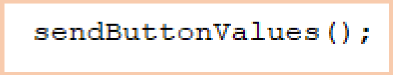
Total : $134.70

* Assuming user has an Arduino, relevant cables, computer

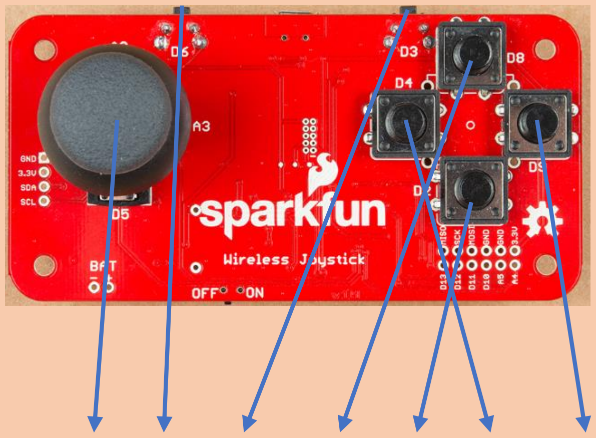
Joystick Code High Level Overview

- We can encode the state of the joystick into 14 bits if we sacrifice some resolution of the horizontal and vertical joysticks.
- These 14 bits can be sent as two bytes.
- The joystick alternate sending one or the other of these bytes every 10 milliseconds.
- We’ll use functions to manipulate the bits in in the bytes to hold our values, and then decode those values on the receiver end.

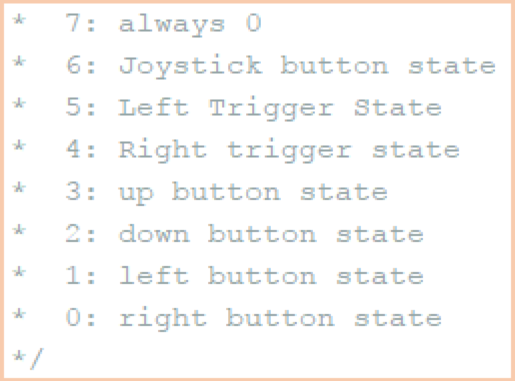
Byte A – Button Values



```
sendButtonValues();
```

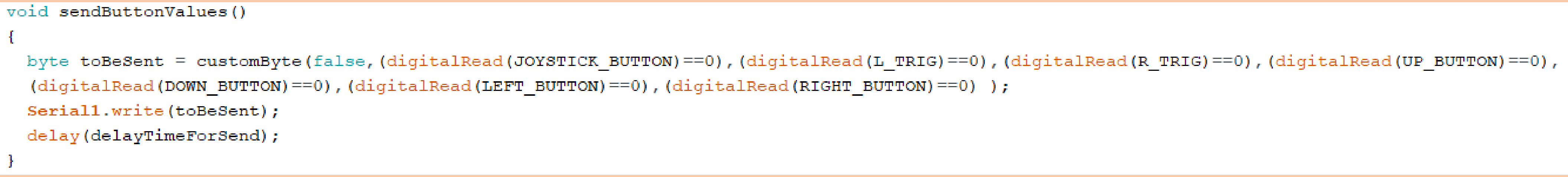


0XXXXXXX



```
*   7: always 0
*   6: Joystick button state
*   5: Left Trigger State
*   4: Right trigger state
*   3: up button state
*   2: down button state
*   1: left button state
*   0: right button state
*/
```

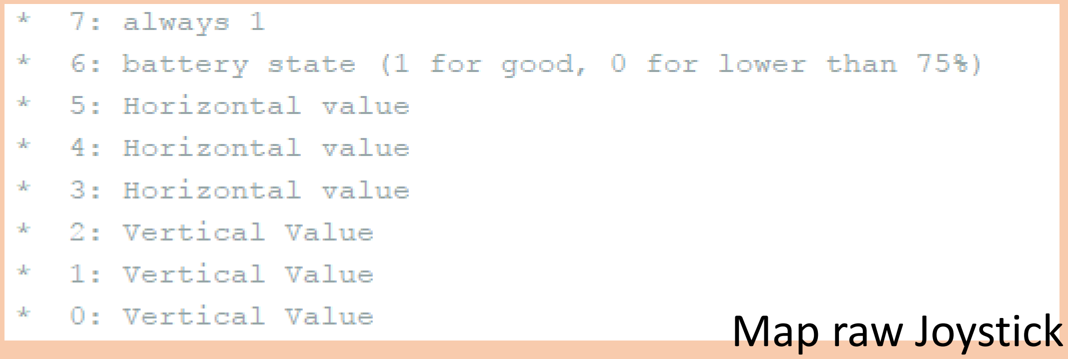
Encode state of each of the 7 buttons a bit. Leave the first bit as a ‘0’ for identification purposes.



```
void sendButtonValues()
{
  byte toBeSent = customByte(false,(digitalRead(JOYSTICK_BUTTON)==0),(digitalRead(L_TRIG)==0),(digitalRead(R_TRIG)==0),(digitalRead(UP_BUTTON)==0),
 (digitalRead(DOWN_BUTTON)==0),(digitalRead(LEFT_BUTTON)==0),(digitalRead(RIGHT_BUTTON)==0)  );
  Serial1.write(toBeSent);
  delay(delayTimeForSend);
}
```

Byte B – Joystick Values

```
sendJSValues();
```



```
*   7: always 1
*   6: battery state (1 for good, 0 for lower than 75%)
*   5: Horizontal value
*   4: Horizontal value
*   3: Horizontal value
*   2: Vertical Value
*   1: Vertical Value
*   0: Vertical Value
```

Map raw Joystick values to a value between 1 and 7

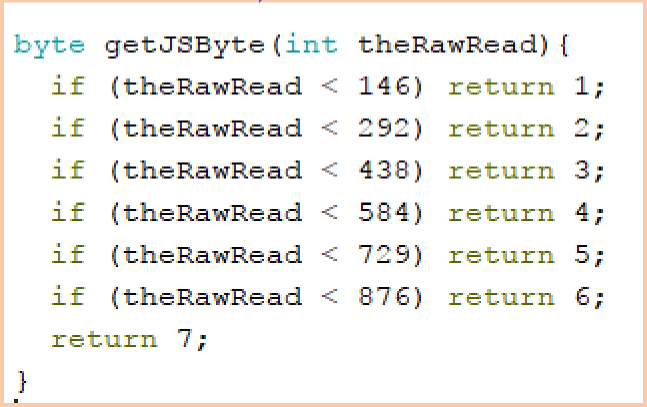
Leave the first bit as a ‘1’ for identification purposes.

$$1\,X\,\underbrace{XXX}_{\text{Horizontal Value (1 to 7)}}\,\underbrace{XXX}_{\text{Vertical Value (1 to 7)}}$$

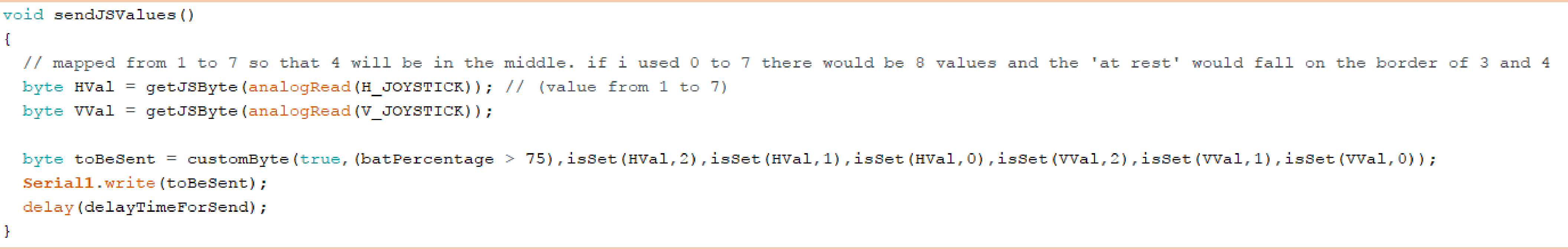
Vertical Value (1 to 7)

Horizontal Value (1 to 7)

Battery State.
Goes low when
battery < 75%



```
byte getJSByte(int theRawRead){
  if (theRawRead < 146) return 1;
  if (theRawRead < 292) return 2;
  if (theRawRead < 438) return 3;
  if (theRawRead < 584) return 4;
  if (theRawRead < 729) return 5;
  if (theRawRead < 876) return 6;
  return 7;
}
```

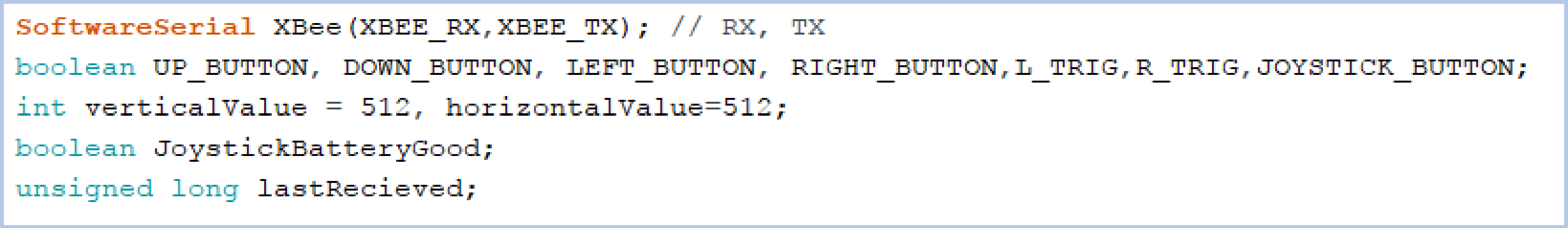


```
void sendJSValues()
{
  // mapped from 1 to 7 so that 4 will be in the middle. if i used 0 to 7 there would be 8 values and the 'at rest' would fall on the border of 3 and 4
  byte HVal = getJSByte(analogRead(H_JOYSTICK)); // (value from 1 to 7)
  byte VVal = getJSByte(analogRead(V_JOYSTICK));

  byte toBeSent = customByte(true,(batPercentage > 75),isSet(HVal,2),isSet(HVal,1),isSet(HVal,0),isSet(VVal,2),isSet(VVal,1),isSet(VVal,0));
  Serial1.write(toBeSent);
  delay(delayTimeForSend);
}
```

Receiving Code High Level Overview

- Goal is now to unpack the two bytes on the receiver
- We create global variables to hold the button states. Whatever code our project has can then reference those button states.



```
SoftwareSerial XBee(XBEE_RX,XBEE_TX); // RX, TX
boolean UP_BUTTON, DOWN_BUTTON, LEFT_BUTTON, RIGHT_BUTTON,L_TRIG,R_TRIG,JOYSTICK_BUTTON;
int verticalValue = 512, horizontalValue=512;
boolean JoystickBatteryGood;
unsigned long lastRecieved;
```

- Two possible scenarios
  - Fast – Project loop runs faster than 20millis –will only harvests one or zero bytes per call
  - Slow – project loop takes longer than 20millis – there will a backlog of bytes from while the project was running
- Same unpacking function works for both cases so that It can be copied and pasted between projects without changing stuff every time.

- Logic below has been successfully tested on both “Fast” and “slow” projects,
- See comments below for explanation:

```
void getLatestXBeeData()
{
  XBeeFlushUnilLatest(); // get rid of all the data except the most recent two bytes.
// fast projects may have an empty serial, in which case they do nothing
// fast projects may also encounter a single byte, in which case they parse it (they will get the complimentary byte in ~ 10 millisends
// a slow project has the latest two bytes, which should be no older than 30 milliseconds
  for(int x = 0;x<2;x++)
  {
      if (XBee.available() > 0)
      {
        byte incomingData;
        incomingData = XBee.read();
        processTheByte(incomingData); // send byte to be unpacked
      }
  }

    lastRecieved = millis(); // Reset all to default if we haven't heard from the remote in a while
    if (lastRecieved + 250 < millis()){
      resetRemoteInput();
    }
}
```

```
void XBeeFlushUnilLatest(){
// flush all the older data away. Keep most recent two bytes
  while(XBee.available() > 2) {
    char t = XBee.read();
  }
}
```

```
void processTheByte(byte aByte)
{
  // check each bit of the incoming byte, and unpack the joystick values.

  if (isClear(aByte,7)){            // unpacking button values
    JOYSTICK_BUTTON = isSet(aByte,6);
    L_TRIG = isSet(aByte,5);
    R_TRIG = isSet(aByte,4);
    UP_BUTTON = isSet(aByte,3);
    DOWN_BUTTON = isSet(aByte,2);
    LEFT_BUTTON = isSet(aByte,1);
    RIGHT_BUTTON = isSet(aByte,0);
   }

  if (isSet(aByte,7)){   // unpack joystick & batteries

    byte H_val = customByte(false,false,false,false,false,isSet(aByte,5),isSet(aByte,4),isSet(aByte,3));
    byte V_val = customByte(false,false,false,false,false,isSet(aByte,2),isSet(aByte,1),isSet(aByte,0));
    JoystickBatteryGood = isSet(aByte,6);
    horizontalValue = unpackJoystick(H_val);
    verticalValue = unpackJoystick(V_val);
  }

}
```

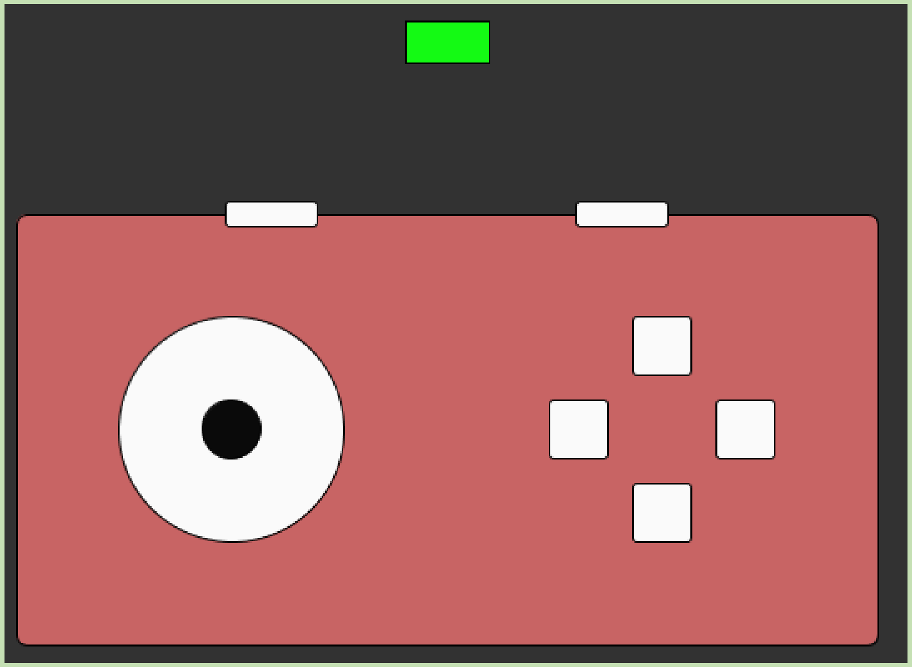
```
int unpackJoystick(byte JSVal) // map joystick back to 0 to 1023 range
{
  if (JSVal==1) return 0;
  if (JSVal==2) return 220;
  if (JSVal==3) return 365;
  if (JSVal==4)  return 512;
  if (JSVal==5) return 657;
  if (JSVal==6) return 803;
  //if (JSVal==7)
  return 1023;

}
```

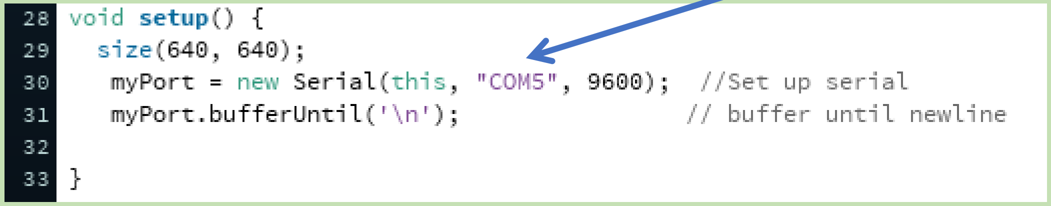
WJK_Output_Processing_Byte

WJK_Output_Processing_byte

- Intention is to use processing3 to see the joystick output in real time while testing.
- Run this on your computer with an xBee explorer attached.
- The buttons should light up when you press the corresponding button on the remote.

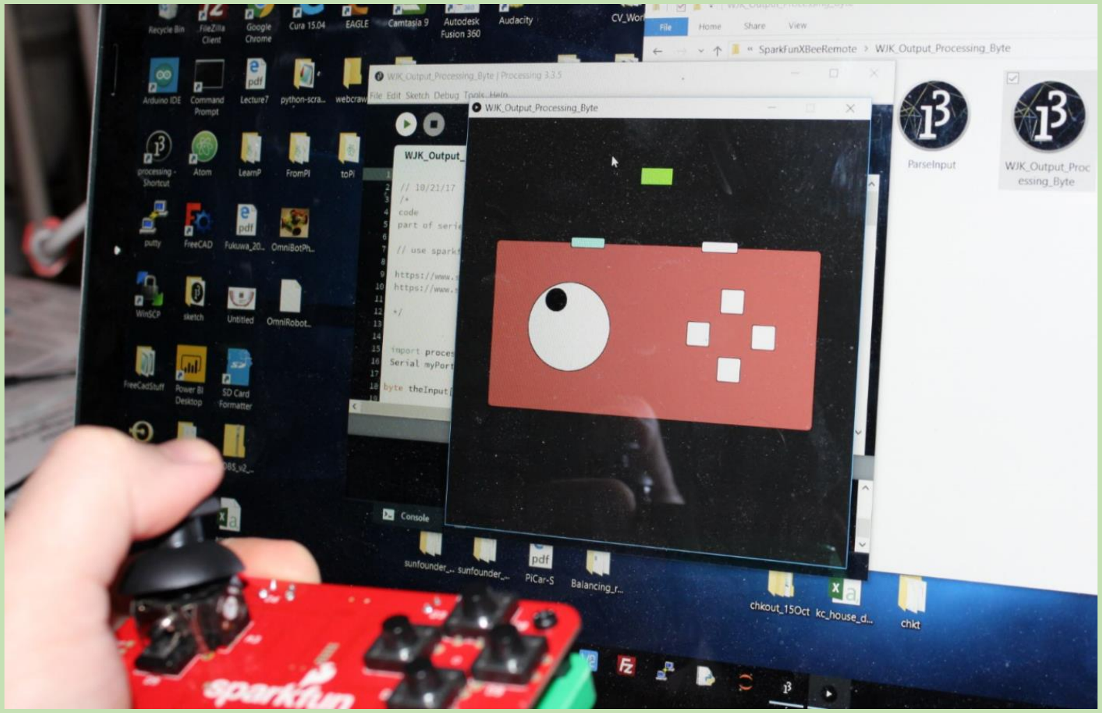


Change this to the port of your Xbee explorer before running

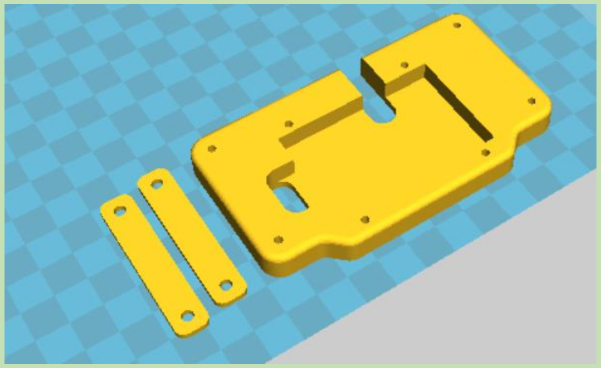


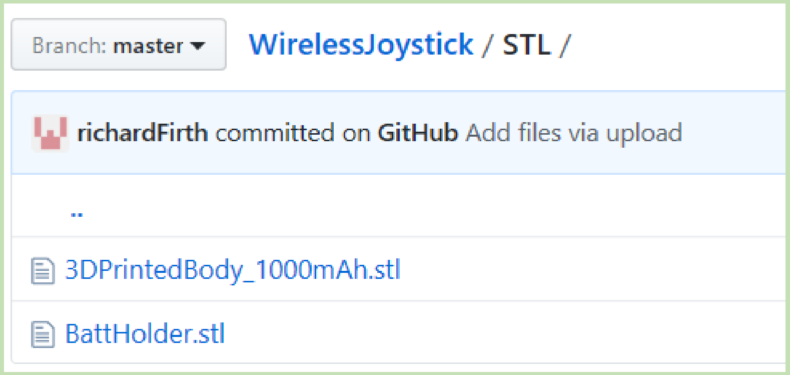
```
void setup() {
  size(640, 640);
   myPort = new Serial(this, "COM5", 9600);  //Set up serial
   myPort.bufferUntil('\n');                // buffer until newline

}
```



Making Holder





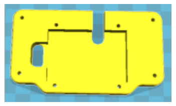
Branch: master ▾ WirelessJoystick / STL /

richardFirth committed on GitHub Add files via upload

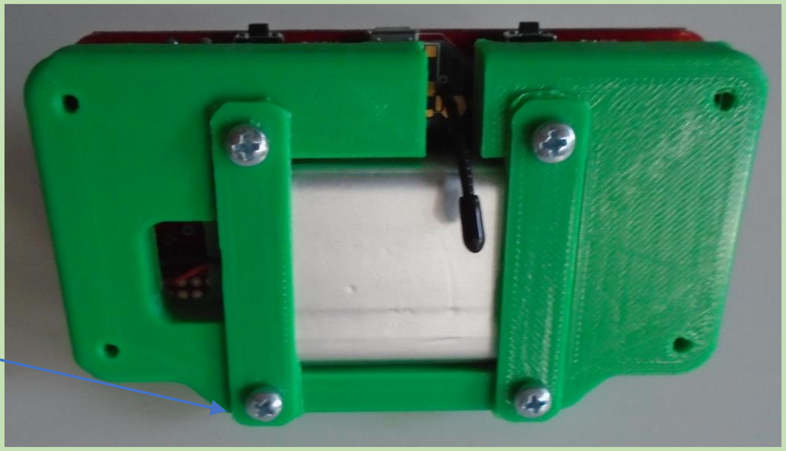
..

3DPrintedBody_1000mAh.stl

BattHolder.stl

Pic	Description	QTY	
	3DPrintedBody_1000mAh	1	
	BattHolder	2	

Hardware found:
https://www.amazon.com/gp/product/B018C19KJ0/
https://www.amazon.com/gp/product/B00MMWDYI4/
https://www.amazon.com/Tonsiki-Adjustable-T-Handle-Reversible-Threading/dp/B01M2X7P0M/
(best results from threading holes)



https://grabcad.com/library/wjk_body-1 <-.STEP file