OpenRov Assembly:
http://openrov.dozuki.com/Guide/How+to+Assemble+OpenROV+2.6/6
OpenRov Parts:
https://docs.google.com/spreadsheets/d/1DdteDIJqQQ5VRS-0wDRqXkb3rrWe9QZ-iHLwQIVYGBA/edit#gid=640628591

Motor:

- Turnigy Aerodrive DST-1000 Brushless Outrunner Motor 700kv
- Prop:
  - 3 Propeller 57 or this - 4.8mm
  - OpenRov Propeller - too expensive!!! But should do this.
  - 3mm - 4mm Couplink
  - Prop shaft
  - 4M Thread Rod
- Wire:
  - Heat Shrink Small & Heat Shrink Large
  - Motor Wire
- Mounting:
  - Motor Mount Screws - M2.5 Head Hex - Length based off of Mounting base thickness.
  - Motor Mount Nuts - M2.5 Locknut
  - Motor Mount Washers - M3
  - Motor Housing:
    - Tube - ***Go to Home Depot***
    - Mounting Base - ***Go to Home Depot***
- Other:
  - Motor Shaft C-Clip
- Websites:
  - Waterproofing (OpenRov Assembly): http://openrov.dozuki.com/Guide/How+to+Assemble+OpenROV+2.6/6

ESCs: (SimonK or BLheli)

- Afro esc 12A
- Bullet Connectors

Arduino:

- Master Arduino: Micro
- Or
- MultiWii Nano

- Slave Arduino: Uno

Battery/Power:

- LiPO: **1800mAh 3S 11.1V**
  - 1800mAh 2S 7.6V
  - 1500mAh 3S 11.1V
- Power Distribution:

        - Wire
        - Detachable Connectors - already have.

Waterproofing:
- Epoxy - ***Go to Home Depot***
- Waterproof Connectors (4 wires)

Tether:
- Wire:
    - 26 awg 10m
- Mesh:
    - 3mm 5m
    - 6mm 5m
- Flaotation:

X-Box Controler:
- XBox 360 USB Controller
- Arduino USB Host Shield 2

Camera System:
- Gopro
- Gopro video cable
- Modified Waterproof Cas:
    - Skeleton Case
    - Epoxy
    - Sheet Metal/Plastic - to fill in the gaps.
    - Website:
    - https://www.youtube.com/watch?annotation_id=annotation_1916814555&feature=iv&src_vid=OWafhM0vAhc&v=munjTdY6LP8
- Wire/Connector
- A/V Plug for LCD FPV Screen

Frame and Buoyancy System:
- PVC - Should go to the store to get a real feeling for measurements.
- Foam
- Weights

Electronics Case:
- PVC Tube

Lights:
- LEDs
- Relay

Other:
- Solder - gottem.

Steps/Process:

1. **Control esc and motors with arduino (You have to reprogram the ESCs to go Forwards and Backwards.)**
    - Materials:
        - Arduino Uno
        - Jumper wires
        - ESC
        - Brushless motor
        - Battery
    - (1) learn how to use the ArduinoUSBlinker at: https://github.com/c---/ArduinoUSBLinker
    - Run the code on the arduino and use pins: grd and D2 for Uno ONLY!!!
    - Be aware of: When applying power to the Arduino and the ESC at the same time the ESC will arm before we are able to set the signal pin HIGH. It will still work but be careful. Best is to connect and power the Arduino first then power up the ESC to ensure that it is held in the bootloader (there should be no beeps from the ESC).
    - (2)Follow instructions to flash the ESCs with forward reverse SimonK firmware: https://forum.openrov.com/t/flashing-a-blank-atmega8-with-simonk-firmware/3125
    - Ignore steps 1-4. Start on 5.
    - Use the "KKmulticopter Flash Tool 0.80 Beta 6" application.
    - Insert:
        - Programmer: ArduinoUSBLinker (arduinousblinker)
        - Controller: atmega 8-based brushless ESC (8kb flash)
    - Continue following the steps.
    - (3)Program the esc with forward/reverse with an Arduino:
    - To get a feel, run the sample code you wrote: file name: ESCForwardReverseSample.ion (esc signal = D3, esc grd = grd).
        - When you run:
            - Plug in arduino and send code.
            - Connect the battery to esc.
            - Open the Serial Monitor in the Arduino Application.
            - Key "2" is plus pulse signal, key "1" is minus pulse signal, and every other key is revert pulse signal to initial signal.
    - Helpful Websites:
        - Control SimonK ESC with Arduino: http://experimentaltechnik.com/simonk-esc-arduino/
        - Potentiometer Sample Code (writemicroseconds): http://rztronics.com/control-brushless-motor-using-arduino/
        - Arduino Controlled ESC Full Class (has a lot of good methods): http://experimentaltechnik.com/simonk-esc-arduino/
        - OpenRov SimonK ESC Code: https://www.youtube.com/watch?v=5sV6oylQTi0
        - Helpful Page Made by SimonK: http://0x.ca/tgy/

2. **Program X-Box Controller to be input:**
    - Materials:
        - Arduino Uno
        - Arduino USB Host Shield 2
        - X-Box 360 Controller with a USB

- (1) Download USB Host Shield 2 Library from Github: https://github.com/felis/USB_Host_Shield_2.0
- Add the Library to the Arduino IDE.
- Go to Examples-> USB Host Shield 2 -> XBox -> XBOXUSB
- IMPORTANT: For the Example Code to Compile you need to use the Arduino IDE 1.8.1!!!!
- (2) Change the "XBOXUSB" code to get input of Joysticks and buttons: Sketch name: "XBoxUSBSample.ion"
- Test the Input in the Serial Monitor.

3. **Program I2C tether communication with motor control. So the X-Box input gets to the Master Arduino: (Slave arduino is controller and Master arduino is ROV)**
    - Materials:
        - Arduino Micro and Uno
        - Arduino USb Host Shield 2
        - Jumper Wires
        - XBox Controller
    - (1) Run the Programs "ROV2Slave1" on a Uno (controller) and "ROV2Master1" or "ROV2Master2" on a Micro (ROV).
    - Great Website: https://www.arduino.cc/en/Tutorial/MasterWriter

4. **Program motor Algorithms that take the X-Box input and turn it into the motions of the ROV.**
    - Run the Programs: "ROV2Master2" and "ROV2SlaveXBox2" with the ESCs connected to the Master Arduino, Slave Arduino Connected to the Master Arduino, and XBox connected to the Slave Arduino.

5. **Test Everything Together -- X-Box Controller, I2C Communication, ESCs/Motors to create the motions of the ROV.**
    - **Must turn on the ROV Arduino First Before the Controller Arduino!!!**

6. **GoPro video send to Eye-Piece-Screen or to Computer Screen or to HD LCD Screen:**
7. **Build the E-Chassi, where all the parts have plugs; least amount of soldering:**
8. **Waterproof Motors and modify wires:**
9. **Build Frame:**
10. **Waterproof frame:**
11. **Add waterproof wire connectors to the frame and a detachable tether connector:**
12. **Make the tether (Floating is better than sinking):**
13. **…**
14. **Test ROV in the Field:**
15. **Make Modifications:**
16. **Repeat Steps 14 and 15 until ROV complete.**
17. **Add sensors, maybe? (depth, temp, IMU):**

Notes:

- When you document all of this, talk about how you were thinking about making a underwater glider (submarine ROV). Explain the amazingness of the science that allows this type of ROV work. That it uses a ballast to control its depth -- the science is due to pressure and density/bouyancy.

```
////////////////////////////////////////////////////////////////////////////////
//README:
//12c Code from: https://www.youtube.com/watch?v=FGjyUSrfx-k
//ESC Code from: https://www.youtube.com/watch?v=DHDOAocEpqU
//
//This program recieves data from the Controller Arduino through i2c communication. Then it processes
//the data from joystick and button values into values such as thrust, vertical, and yaw. Then it has
//algorithms that tell the ESCs their correct values based off the new values(thrust, vertical, yaw).
//The algorithms may not work at first and will need to be tweeked for perfection!!!
//
//*****************************************************************************************************************
//**THIS VERSION IS DIFFERENT THAN "ROV2MASTER1" BECAUSE IT HAS DIFFERENT
OUTCOMES FOR "FULL RIGHT" AND "FULL LEFT".**
//*****************************************************************************************************************
//
//PROGRAM A SENSITIVITY VARIABLE!!!
//
//Arduino: in ROV.
//Connected to: ESCs, i2c wires.
//
//PAY ATTENTION TO ANYTHING WITH ASTRICS:******...
//
//PINS: ("<->" means the two pins on the two arduinos are connected)("|" means that the two different
pins on the two arduinos DO NOT connect)
//  Type      |  Master-Micro  |  Type      |  Slave-Uno
//  i2c SCL       D3          <-> i2c SCL      A5
//  i2c SDA       D2          <-> i2c SDA      A4
//  escR          D5           |  RJoyX        A3
//  escL          D6           |  RJoyY        A2
//  escV          D9           |  LJoyX        A1
//                             |  LJoyY        A0
//
//NOTE: Could work on Right Forward & Right Reverse and Left Forward & Left Reverse body codes.
There are notes thier that describe an objective.
////////////////////////////////////////////////////////////////////////////////
/*
  This Program is for a ROV with a Gopro Camera and a 3 thruster mobility system. (not for a submerine
with a ballast.)
*/

#include <Wire.h>
#include <Servo.h>

//Input Data Variables:
  int yInputJoyL, xInputJoyL; //Input data
  int yInputJoyR, xInputJoyR; //Input data (on/off data)(not sure of the Output yet)(Should be 1 & 0)
  int button1, button2, button3, button4; //Input data.
//Main Data Variables:
  int yaw             = 0; //A range: -189 -- 0 -- 180  //Ex. if range=90, then escL+power and escR-power, as
long as min < escL&escR < max.
  int thrust          = 0; //A range: -180 -- 0 -- 180
  int vertical        = 0; //A range: -180 -- 0 -- 180
//Parts Pin Variables: (need to assign real PWM pins)
  int escLPin         = 9; Servo escL; //PWM
  int escRPin         = 10; Servo escR; //PWM
  int escVPin         = 5; Servo escV; //PWM
  int escValR;
  int escValL;
//Other Variables:
  const int id              = 5;
  const int messageLength    = 12; //if problems, revert to "10"
  const int minDeadZone      = 1450;
  const int maxDeadZone      = 1549;
  const int s               = 7; //*****Must be between: 0 and 8 inclusive. "s" changes the Range between
"minESC" and "maxESC" to be Shorter: (increase s)The amount of movement of the joy affects the ESC
signal LESS OR Longer:(decrease s) The amount of movement of the joy affects the ESC signal
MORE.*****
  int minESC                = 1000 + s*50;
  const int medESC          = 1500;
  int maxESC                = 2000 - s*50;

void setup() {
  Wire.begin();
  Serial.begin(115200);//regular: 9600.

  escL.attach(escLPin);
  escR.attach(escRPin);
  escV.attach(escVPin);
  //Arming ESCs:
  escL.writeMicroseconds(1520);
  escR.writeMicroseconds(1520);
  escV.writeMicroseconds(1520);

  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  Wire.beginTransmission(id);
  int available = Wire.requestFrom(id, messageLength);

  if(available == messageLength) {
    //Blink LED to show it is connected:
        //digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
        Serial.print("Connected");
```

```
   //Read Slave Data: (Needs to correlate with the Data placement in the Array!)
//////////////////////////////////////////////////
        // -What this does is it scans through the array and after reading a piece of data it deletes it to read
the next one.
        yInputJoyL  = (Wire.read() << 8) | (Wire.read()); //Index 0,1
        xInputJoyL  = (Wire.read() << 8) | (Wire.read()); //Index 2,3
        yInputJoyR  = (Wire.read() << 8) | (Wire.read()); //Index 4,5
        xInputJoyR  = (Wire.read() << 8) | (Wire.read()); //Index 6,7
        button1     = Wire.read(); //Index 8
        button2     = Wire.read(); //Index 9
        button3     = Wire.read(); //Index 10
        button4     = Wire.read(); //Index 11

   //Processing Data / Conversion of Input to Main Data / Conversion of Main to Output Data:
////////////////////////////////////////
        //Input to Servo/Print Data:
        //original: 0<->1023 not -32770<->32770
        yInputJoyL  = map(yInputJoyL, -32770, 32770, minESC, maxESC);          //--------PROGRAM
SENSITIVITY---------// the movement of the
        yInputJoyR  = map(yInputJoyR, -32770, 32770, minESC, maxESC);          //joy stick is more or less
impacful on the ESC signals.
        xInputJoyR  = map(xInputJoyR, -32770, 32770, minESC, maxESC);          //Right now the
Sensitivity = 1. (ratio, joy:ESC is 1:1)
                                                        //******Do this by making the Range less than
1000-2000.**
                                                        //EX. Make "minESC": 1250 and "maxESC": 1750. This
cuts the
        //Input to Main Data:                                   //Range in half (1000-2000 = 1000 to 1250-1750 =
500).
        yaw          = xInputJoyR;  //Value: 1000-2000  //(L->R Increase)Left is 1000 and Right is 2000.
        vertical     = yInputJoyL;  //Value: 1000-2000  //^^^    //****Write Now, I am assuming 1000
correlates to Right and 2000 correlates to Left. May need to reverse based on tests of actual Joys.****
        thrust       = yInputJoyR;  //Value: 1000-2000  //^^^

  //DO: ALGORITHMS//////////////////////////////////////////////////////////////////////////////////////////////////////
    //ESC (vertical):
        escValR = thrust;
        escValL = thrust;
        if(vertical < minDeadZone || vertical > maxDeadZone){
          escV.writeMicroseconds(vertical);
          Serial.print("   UP/DOWN");
          Serial.print("   escV = "); Serial.print(vertical);
        }
        else{
          escV.writeMicroseconds(medESC); //Stop movement
          Serial.print("   UP/DOWN"); Serial.print("   escV = "); Serial.print(medESC);
        }
    //ESCs (Thrust)(Yaw):
        //This makes the Right and Left ESCs equal to the "thrust" but then sub or add to their val
according to the change on "yaw"(n).
        //****To Visualize this, draw a graph and make yaw equal to x-values and thrust equal to y-values. It
should surround the origin and create a symmetrical shape. There should be 1 section and 2 half sections
per
        //   Quadrant of the Graph (in whole graph there should be 8 sections). Sections should be (clock
wise): Forward, Right Forward, Full Right, Right Reverse, Reverse, Left Reverse, Full Left, Left
Forward.*****
        int nL = yaw - minESC; //yaw - 1000. //"nL" is the amount of change in yaw from the midpoint.
        int nR = (maxESC - minESC) - nL; //Inverse direction of nL.

        //If in DeadZone, then stop:
        if((thrust > minDeadZone && thrust < maxDeadZone) && (yaw > minDeadZone && yaw <
maxDeadZone)){
          escL.writeMicroseconds(medESC);
          escR.writeMicroseconds(medESC);
          Serial.print("   Off");
          Serial.print("   escR = "); Serial.print(medESC);
          Serial.print("   escl = "); Serial.print(medESC);
        }                                                       //Vars to Use: minDeadZone/maxDeadZone, thrust,
n, yaw, minESC/maxESC/medESC
        //Right:
        else if(yaw < minDeadZone){ //yaw > 1550.    *******IF the joy Left and Right data is flipped, then
switch this "if statement" with the Left one.********
            //Forward: yaw < 1450 and thrust >= 1550 (Doesn't include full forward and full Right)(Includes
area between full forward and full Right)
            if(thrust >= maxDeadZone){
              escL.writeMicroseconds(escValL);
              escR.writeMicroseconds(medESC);
              //****Change Code Here that changes escR val to make ROV turn forwards while still moving
forwards. Now it is just turning and not moving.****
              Serial.print("   Right Forward");
              Serial.print("   escR = "); Serial.print(medESC);
              Serial.print("   escL = "); Serial.print(escValL);
            }
            //Full Left: yaw < 1450 and 1450 < thrust < 1550
            else if(thrust > minDeadZone && thrust < maxDeadZone){
              escR.writeMicroseconds(medESC);
              escL.writeMicroseconds(medESC + (nR - ((maxESC - minESC) / 2)));
              Serial.print("   Full Right");
              Serial.print("   escR = "); Serial.print(medESC);
              Serial.print("   escL = "); Serial.print(medESC + (nR - ((maxESC - minESC) / 2)));
            }
            //Reverse: yaw < 1450 and thrust <= 1450 (Doesn't include full reverse and full Right)(Includes
area between full reverse and full Right)
            else if(thrust <= minDeadZone){
              escR.writeMicroseconds(escValR);
              escL.writeMicroseconds(medESC);
          //****Add Code Here that changes escL val to make ROV turn reverse while still moving
reverse. Now it is just turning and not moving.****
          Serial.print("   Right Reverse");
          Serial.print("   escR = "); Serial.print(escValR);
          Serial.print("   escL = "); Serial.print(medESC);
         }
        }
      //Left:
      else if(yaw > maxDeadZone){ //yaw < 1450.
         //Forward: yaw < 1450 and thrust >= 1550 (Doesn't include full forward and full Left)(Includes
area between full forward and full Left)
         if(thrust >= maxDeadZone){
          escL.writeMicroseconds(medESC);
          escR.writeMicroseconds(escValR);
          //****Change Code Here that changes escL val to make ROV turn forwards while still moving
forwards. Now it is just turning and not moving.****
          Serial.print("   Left Forward");
          Serial.print("   escR = "); Serial.print(escValR);
          Serial.print("   escL = "); Serial.print(medESC);
         }
         //Full Left: yaw < 1450 and 1450 < thrust < 1550
         else if(thrust > minDeadZone && thrust < maxDeadZone){
          escR.writeMicroseconds(medESC + (nL - ((maxESC - minESC) / 2))); //(maxESC - minESC) /
2)) == 500.
          escL.writeMicroseconds(medESC);
          Serial.print("   Full Left");
          Serial.print("   escR = "); Serial.print(medESC + (nL - ((maxESC - minESC) / 2)));
          Serial.print("   escL = "); Serial.print(medESC);
         }
         //Reverse: yaw < 1450 and thrust <= 1450 (Doesn't include full reverse and full Left)(Includes
area between full reverse and full Left)
         else if(thrust <= minDeadZone){
          escR.writeMicroseconds(medESC);
          escL.writeMicroseconds(escValL);
          //****Add Code Here that changes escR val to make ROV turn reverse while still moving
reverse. Now it is just turning and not moving.****
          Serial.print("   Left Reverse");
          Serial.print("   escR = "); Serial.print(medESC);
          Serial.print("   escL = "); Serial.print(escValL);
         }
      }
      //Med (Straight or Reverse):
      else{ //1450 < yaw < 1550.
        escR.writeMicroseconds(escValR);
        escL.writeMicroseconds(escValL);
        Serial.print("   Forwards/Reverse");
        Serial.print("   escR = "); Serial.print(escValR);
        Serial.print("   escL = "); Serial.print(escValL);
```

```
          }

      //Print: ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
          Serial.print("  Button1 = ");       Serial.print(button1);
          Serial.print("  Button2 = ");       Serial.print(button2);
          Serial.print("  Button3 = ");       Serial.print(button3);
          Serial.print("  Button4 = ");       Serial.print(button4);
          //Serial.print("  |  ");

          //Serial.print("  JoyLeftY = ");      Serial.print(yInputJoyL);
          //Serial.print("  Vertical = ");      Serial.print(vertical);
          //Serial.print("  |  ");

          //Serial.print("  JoyRightX = ");     Serial.print(xInputJoyR);
          //Serial.print("  JoyRightY = ");     Serial.print(yInputJoyR);
          //Serial.print("  Yaw = ");           Serial.print(yaw);
          //Serial.print("  Thrust = ");        Serial.print(thrust);

          //Serial.print("  nR = ");            Serial.print(nR);
          //Serial.print("  nL = ");            Serial.print(nL);

          Serial.println();
    }
    else{
      //Default Sub Mode / Failsafe:
      Serial.println("Signal Disconnected");
      digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    }

    Wire.endTransmission();
    delay(50);

  }
```

```
/*
 Example sketch for the Xbox 360 USB library - developed by Kristian Lauszus
 For more information visit my blog: http://blog.tkjelectronics.dk/ or
 send me an e-mail:  kristianl@tkjelectronics.com
 */
 ///////////////////////////////////////////////////////////////////////////////////////////////////
 //THIS CODE IS A DERIVATIVE OF THE CODE FROM "ROV2Slave2XBox" but modified for the Xbox
controller.
 ///////////////////////////////////////////////////////////////////////////////////////////////////

#include <XBOXUSB.h>
#include <Wire.h>

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#include <SPI.h>
#endif

USB Usb;
XBOXUSB Xbox(&Usb);

int leftJoyX;
int leftJoyY;
int rightJoyX;
int rightJoyY;
byte button1 = 0;
byte button2 = 0;
byte button3 = 0;
byte button4 = 0;

int id = 5;
int messageLength = 12;
byte messageArray[12];

void setup() {
  Serial.begin(115200);
#if !defined(__MIPSEL__)
  while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy and other boards with
built-in USB CDC serial connection
#endif
  if (Usb.Init() == -1) {
    Serial.print(F("\r\nOSC did not start"));
    while (1); //halt
  }
  Serial.print(F("\r\nXBOX USB Library Started"));

  Wire.begin(id);
  Wire.onRequest(requestEvent);
}

void loop() {
  Usb.Task();
  if (Xbox.Xbox360Connected) {

    if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500 ||
Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500 ||
Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500 ||
Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
      if (Xbox.getAnalogHat(LeftHatX) > 7500 || Xbox.getAnalogHat(LeftHatX) < -7500) {
        Serial.print(F("LeftHatX: "));
        Serial.print(Xbox.getAnalogHat(LeftHatX));
        Serial.print("\t");
        leftJoyX = Xbox.getAnalogHat(LeftHatX);
      }
      if (Xbox.getAnalogHat(LeftHatY) > 7500 || Xbox.getAnalogHat(LeftHatY) < -7500) {
        Serial.print(F("LeftHatY: "));
        Serial.print(Xbox.getAnalogHat(LeftHatY));
        Serial.print("\t");
        leftJoyY = Xbox.getAnalogHat(LeftHatY);
      }
      if (Xbox.getAnalogHat(RightHatX) > 7500 || Xbox.getAnalogHat(RightHatX) < -7500) {
        Serial.print(F("RightHatX: "));
        Serial.print(Xbox.getAnalogHat(RightHatX));
        Serial.print("\t");
        rightJoyX = Xbox.getAnalogHat(RightHatX);
      }
      if (Xbox.getAnalogHat(RightHatY) > 7500 || Xbox.getAnalogHat(RightHatY) < -7500) {
        Serial.print(F("RightHatY: "));
        Serial.print(Xbox.getAnalogHat(RightHatY));
        rightJoyY = Xbox.getAnalogHat(RightHatY);
      }
      Serial.println();

      //Vibrating Intensity for horizontal thrusters:
      int numf = abs(map(rightJoyY, -32780, 32770, -200, 200));
      int numk = abs(map(rightJoyX, -32780, 32770, -200, 200));
      int numt = abs(map(leftJoyY, -32780, 32770, -225, 225));
      Serial.print(numf); Serial.print("  "); Serial.print(numk); Serial.print("  "); Serial.print(numt); Serial.print("
");
      if(numf < 255 && numf > 70){
        Xbox.setRumbleOn(0, numf); //(Left: 0-255 , Right: 0-255)
        if(numt < 255 && numt > 70){
          Xbox.setRumbleOn(numt, numf);
        }
      }
      else if(numk < 255 && numk > 70){
        Xbox.setRumbleOn(0, numk);
        if(numt < 255 && numt > 70){
          Xbox.setRumbleOn(numt, numf);
        }
      }
      else if(numt < 255 && numt > 80){
          Xbox.setRumbleOn(numt, 0);
      }
      else{ Xbox.setRumbleOn(0,0); }

    }

    if (Xbox.getButtonClick(A)){
      Serial.println(F("A"));
      button1 = 1;
    } else{button1 = 0; }


    if (Xbox.getButtonClick(B)){
      Serial.println(F("B"));
      button2 = 1;
    } else{button2 = 0;}
    if (Xbox.getButtonClick(X)){
      Serial.println(F("X"));
      button3 = 1;
    } else{button3 = 0;}
    if (Xbox.getButtonClick(Y)){
      Serial.println(F("Y"));
      button4 = 1;
    } else{button4 = 0;}

  }
  delay(1);
}



void requestEvent() {
  //JoyL Array Assignment:
  messageArray[0]  = highByte(leftJoyY);
  messageArray[1]  = lowByte(leftJoyY);
  messageArray[2]  = highByte(leftJoyX);
  messageArray[3]  = lowByte(leftJoyX);
  //JoyR Array Assignment:
  messageArray[4]  = highByte(rightJoyY);
  messageArray[5]  = lowByte(rightJoyY);
  messageArray[6]  = highByte(rightJoyX);
  messageArray[7]  = lowByte(rightJoyX);
  //ElevBut Array Assignment:
  messageArray[8]  = button1;
  messageArray[9]  = button2;
  messageArray[10] = button3;
  messageArray[11] = button4;

  //Message Send:
  Wire.write(messageArray, messageLength); //change later to only send Data: Roll/Pitch/Yaw/Elevation
}
```