```
#include<SabertoothSimplified.h>

/*
 Joystick Control of two Sabertooth 2x5 modules
 Horizontal control is X1, Y1
 Vertical/Crabbing control is X2, Y2
 */
 // Include the Software Serial library
#include <SoftwareSerial.h>

// include the DISPLAY library
#include <LiquidCrystal.h>
//
// Define your pins and variables
// initialize the library with the numbers of the interface pins -
//LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
//int STSerialOut = 8;           // software serial output pin
//int STSerialIn = 7;            // routines need an input pin but
//int ST_S2H = 9;                // horizontal S2 select pin
//int ST_S2V = 10;               // Vertical S2 select pin

// initialize the library with the numbers of the interface pins -
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
int STSerialOut = 11;           // software serial output pin
int STSerialIn = 10;            // routines need an input pin but
int ST_S2H = 12;                // horizontal S2 select pin
int ST_S2V = 13;               // Vertical S2 select pin

// Initialize the Software Serial now that the pins were declared a
SoftwareSerialStSerial(STSerialIn, STSerialOut); // RX, TX

// ROV Type
int VECTORED = 0;
int ORTHO = 1;
int ROV_Type = VECTORED;   // define the configuration of the ROV V

// Horizontal Control
```

```
int POT_X1 = A1;  // where the X POTENTIOMETER analog signal is con
int POT_Y1 = A2;  // where the Y POTENTIOMETER analog signal is con
int CountsX1;     // a location to save the result of the analog re
int CountsY1;     // a location to save the result of the analog re
int Val_X1 = 0;   // a location to save the result calculations for
int Val_Y1 = 0;   // a location to save the result calculations for
int MtrR = 0;     // value that will be used for the right thruster
int MtrL = 0;     // value that will be used for the left thruster
//
// Vertical Control
int POT_X2 = A3;  // where the X POTENTIOMETER analog signal is con
int POT_Y2 = A4;  // where the Y POTENTIOMETER analog signal is con
int CountsX2;     // a location to save the result of the analog re
int CountsY2;     // a location to save the result of the analog re
int Val_X2 = 0;   // a location to save the result calculations for
int Val_Y2 = 0;   // a location to save the result calculations for
int MtrVR = 0;    // value that will be used for the right thruster
int MtrVL = 0;    // value that will be used for the left thruster
//
// Sabertooth Values
byte SaberToothH;   // output value for the Horizontal Sabertooth
byte SaberToothV;    // output value for the Vertical Sabertooth


void setup() {
   // set up the LCD's number of columns and rows:
   lcd.begin(16, 2);
   // Print a message to the LCD.
   lcd.print("Simplified Ser.");
   lcd.setCursor(0, 1);    // set the cursor to column 0, line 1
   if (ROV_Type == VECTORED){
       lcd.print("  VECTORED  ");
   }
   else {
       lcd.print("  ORTHO     ");
   }
   pinMode(STSerialOut,OUTPUT);       // sabertooth control pin
   pinMode(ST_S2H,OUTPUT);            // sabertooth control pin
```

```
   pinMode(ST_S2V,OUTPUT);             // sabertooth control pin
   digitalWrite(ST_S2H,LOW);           // set to disable
   digitalWrite(ST_S2V,LOW);           // set to disable
   delay(10);   // delay a bit

  // set the data rate for the SoftwareSerial port
   StSerial.begin(9600);
   MtrR = 0;
   digitalWrite(ST_S2H,HIGH);     // enable sabertooth
   digitalWrite(ST_S2V,HIGH);     // enable sabertooth (enable both
   StSerial.write(MtrR);             // send a command to shut off
   delay(1);                      // delay a bit
   digitalWrite(ST_S2H,LOW);     // disable sabertooth
   digitalWrite(ST_S2V,LOW);     // disable sabertooth

   delay(2000);
   lcd.setCursor(0, 0);
   lcd.print("                ");
   lcd.setCursor(0, 1);    // set the cursor to column 0, line 1
   lcd.print("                ");
}

void loop() {
  // read the Horizontal Joystick
  CountsX1 = analogRead(POT_X1);  // go read the analog input of X
  CountsY1 = analogRead(POT_Y1);  // go read the analog    of Y
  Val_X1 = CountsX1;              // keep the raw counts for later
  Val_Y1 = CountsY1;              // keep the raw counts for later
  Val_X1 -= 511;                  // center the value around 0, max
  Val_Y1 -= 511;                  // center the value around 0, max

  MtrR = Val_Y1 + Val_X1;      // create the X & Y mixed values
  MtrL = Val_Y1 - Val_X1;      // create the X & Y mixed values

  if (MtrR > 511) {          // handle the large positive values an
     MtrR = 511;
  }
  if (MtrR < -511) {         // handle the large negative values an
```

```
    MtrR = -511;
  }
  if (MtrL > 511) {          // handle the large positive values an
    MtrL = 511;
  }
  if (MtrL < -511) {         // handle the large negative values an
    MtrL = -511;
  }

  // read the Vertical Joystick
  CountsX2 = analogRead(POT_X2);  // go read the analog input of X
  CountsY2 = analogRead(POT_Y2);  // go read the analog input of Y
  Val_X2 = CountsX2;              // keep the raw counts for later
  Val_Y2 = CountsY2;              // keep the raw counts for later
  Val_X2 -= 511;                  // center the value around 0 max
  Val_Y2 -= 511;                  // center the value around 0 max

  if (ROV_Type == VECTORED) {
      if (abs(Val_X2) < 100) {    // check for no crabbing input
          MtrVR = Val_Y2;         // no crabbing then put both vert
          MtrVL = Val_Y2;
      }
      else {
          MtrVR = Val_X2;         // crabbing input then switch to
          MtrVL = -Val_X2;        // left is opposite from right.
      }
  }
   else {                        // for Orthogonal design
       MtrVR = Val_Y2;           // vertical thruster = Y input
       MtrVL = Val_X2;           // crabbing thruster = X input
   }
//
// Before scaling the values, send status to the display
//

  // Display the Results
  lcd.setCursor(0, 0);
  if (MtrL == 0) {
```

```
        lcd.print("OFF ");
        lcd.print(MtrL);
 }
 else if (MtrL > 0){
        lcd.print("FD ");
        lcd.print(MtrL);
 }
 else{
        lcd.print("RV ");
        lcd.print(MtrL);
 }
 if (MtrR == 0) {
        lcd.print(" OFF ");
        lcd.print(MtrR);
 }
 else if (MtrR > 0){
        lcd.print(" FD ");
        lcd.print(MtrR);
 }
 else{
        lcd.print(" RV ");
        lcd.print(MtrR);
 }
 lcd.print("        ");
 lcd.setCursor(0, 1);    // set the cursor to column 0, line 1
 if (MtrVL == 0) {
        lcd.print("OFF ");
        lcd.print(MtrVL);
 }
 else if (MtrVL > 0){
        lcd.print("FD ");
        lcd.print(MtrVL);
 }
 else{
        lcd.print("RV ");
         lcd.print(MtrVL);
}
 if (MtrVR == 0) {
```

```
        lcd.print(" OFF ");
        lcd.print(MtrVR);
 }
 else if (MtrVR > 0){
        lcd.print(" FD ");
        lcd.print(MtrVR);
 }
 else{
        lcd.print(" RV ");
        lcd.print(MtrVR);
 }
 lcd.print("        ");
//
//
// Create the values to send to the SaberTooth Controllers
// Horizontal
   MtrR /= 8;                      // divide +-511 by 8
   MtrR += 64;                     // shift from -64 to 64 to 0 to
   if (MtrR == 0) {                // full reverse = 1, 0 is emerge
         MtrR = 1;
   }
   if (MtrR > 127){                // full foward = 128
         MtrR = 127;
   }
   MtrL /= 8;                      // divide +-511 by 8
   MtrL += 192;                     // shift from -64 to 64 to 128
   if (MtrL < 128){                // full reverse = 128
         MtrL = 128;
   }
   if (MtrL > 255){                // full forward = 255
         MtrL = 255;
   }

// Vertical
   MtrVR /= 8;                      // divide +-511 by 8
   MtrVR += 64;                     // shift from -64 to 64 to 0 to
   if (MtrVR == 0) {                // full reverse = 1, 0 is emerg
         MtrVR = 1;
```

```
    }
    if (MtrVR > 127){                // full foward = 128
          MtrVR = 127;
    }
    MtrVL /= 8;                      // divide +-511 by 8
    MtrVL += 192;                     // shift from -64 to 64 to 128
    if (MtrVL < 128){                // full reverse = 128
          MtrVL = 128;
    }
    if (MtrVL > 255){                // full forward = 255
          MtrVL = 255;
    }

  // send the right speed and direction commands to the SaberTooth
   digitalWrite(ST_S2H,LOW);           // set to disable
   digitalWrite(ST_S2V,LOW);           // set to disable
   delay(1);
   digitalWrite(ST_S2H,HIGH);      // enable Horizontal sabertooth
   StSerial.write(MtrR);           // send the right horizontal spee
   digitalWrite(ST_S2H,LOW);       // set to disable
   delay(1);                       // delay a bit
   digitalWrite(ST_S2H,HIGH);      // enable Horizontal sabertooth
   StSerial.write(MtrL);           // send the right horizontal spee
   digitalWrite(ST_S2H,LOW);       // set to disable
   delay(1);
   digitalWrite(ST_S2V,HIGH);     // enable vertical sabertooth
   StSerial.write(MtrVR);           // send the right vertical speed
   delay(1);                       // delay a bit
   StSerial.write(MtrVL);           // send the right vertical speed
   delay(1);
   digitalWrite(ST_S2V,LOW);           // set to disable
//
// slow the loop down, you can add other tasks below as needed
  delay(50);
}
```