```
#include  <LiquidCrystal_I2C.h>
#include  <Adafruit_MotorShield.h>
#include   "utility/Adafruit_MS_PWMServoDriver.h"
#include <Wire.h>
#include  <hcsr04.h>
#include <sbus.h>

#define MODE_TYPE 4    // Pin for the mode type
#define TRIG_PIN 7     // Trigger pin for ultrasonic
#define ECHO_PIN 8     // Echo pin for ultrasonic
#define IR_RIGHT A0    // Pin for Right IR sensor
#define IR_LEFT A1     // Pin for Left IR sensor
#define RX 0           // Receiver for wireless transmission


const int quick = 55 ;                      // Speed of the
motors


int distance = 800;         // Variable for distance from
ultrasonic sensor
int IRRight = 0;            // Variable for value of the right
IR sensor
int IRLeft = 0;             // Variable for value of the left I
sensor



float Right = 0;            // Adjusted values for the IR input
float Left = 0;

HCSR04 sonic(TRIG_PIN, ECHO_PIN, 20, 2000);   // Create
ultrasonic object with above definitions
                                              // as well as
minimum distance of 20mm and max of 20cm
                                              // as that is th
```

```
diameter of the ring

Adafruit_MotorShield AFMS = Adafruit_MotorShield(); // Create
motorshield object.

Adafruit_DCMotor *leftMotor = AFMS.getMotor(4);      // Sets
the two motors to two objects so we can use them
Adafruit_DCMotor *rightMotor = AFMS.getMotor(3);

SBUS sbus;  // Create sbus object
int comfb;  // Command value for forward/backward
int comrl;  // Command value for left/right
int prevfb; // Previous comfb value
int prevrl; // Previous comrl value

LiquidCrystal_I2C lcd(0x27, 20, 4); // set up the lcd object

void setup()
{
  Serial.begin(9600); // Standard baud rate set

  pinMode(12, OUTPUT);                // Check and use manual
input if switch is on
   pinMode(MODE_TYPE, INPUT);
   digitalWrite(12, HIGH);
   if(digitalRead(MODE_TYPE) == LOW) // Read mode for user
input or autonomous
   {
     userInput();
   }


  AFMS.begin();                  // Start motorshield functions
   leftMotor->setSpeed(quick + 5);    // Set rate for motor
speeds, left motor moves a little slower than right
   rightMotor->setSpeed(quick);       // adjusted for that
```

```
  sbus.begin(RX, sbusNonBlocking);   // Start sbus receiver in
nonblocking mode

  pinMode(IR_RIGHT, INPUT);    // Set up the IR readers for
input
   pinMode(IR_LEFT, INPUT);

  lcd.init();           // Initialize lcd and turn the backlight
on
   lcd.backlight();

  delay(5000); // Per sumobot official rules, bots may not
start until after a five second delay
}

void loop()
{
  distance = sonic.distanceInMillimeters();    // Read in
values from sensors
   IRRight = analogRead(IR_RIGHT);
   IRLeft = analogRead(IR_LEFT);

  Right = (float)IRRight * 5 / 1023.0;          //Find adjusted
values for IR sensors
  Left = (float)IRLeft * 5 / 1023.0;

  lcd.setCursor(0, 0);          // Output debugging info
  lcd.print("Distance: " + String(distance) + "
");
   lcd.setCursor(0,1);
  lcd.print("RIR Sensor: " + String(Right) + "    ");
   lcd.setCursor(0,2);
  lcd.print("LIR Sensor: " + String(Left) + "    ");
   lcd.setCursor(0,3);
```

```
  if((Right < 2.0 || Left < 2.0) && distance <
150)               // First two statements check right and lef
IR
  {                                              // sensors to
move away from the edge for 1 second
    moveBack();
    int s = 0 - quick;
    lcd.print("LW: " + String(s) + " RW: " + String(s) + "
");     // Print strng to PCB

delay(1000);
  //Delay so that the bot is a fair distance from edge before
starting again.
  }
  else if(distance < 720 && distance != -1)             // Chec
for object in ring in front of it if there
  {                                                     // is
one move toward it (ring is 720 max 80 buffer)
    moveForward();
    lcd.print("LW: " + String(quick) + " RW: " + String(quick
+ "     ");   // Print strng to PCB
  }
  else if(distance != -1)                        // If no other
inputs cause something, rotate until
  {                                              // we find
something. Ignore the case of -1 as the ultrasonic sensor
    turnLeft();                                  // throws this
when either too close or too far

    lcd.print("LW: " + String(0) + " RW: " + String(quick) +
"     ");   // Print strng to PCB
  }

  delay(20);                                    // Delay 20ms for
fluidity
```

```
}

/*
 * Function to set motors to move forward
 * Note: Due to physical contraints, the left motor is wired
backwards, so all inputs are opposite.
 */
void  moveForward()
{
   leftMotor->run(BACKWARD);
   rightMotor->run(FORWARD);
}

/*
 * Function to set motors to reverse
 */
void  moveBack()
{
   leftMotor->run(FORWARD);
   rightMotor->run(BACKWARD);
}

/*
 * Function to set motors to turn right
 */
void  turnLeft()
{
   leftMotor->run(RELEASE);
   rightMotor->run(FORWARD);
}

/*
 * Function to set motor to turn left
 */
void  turnRight()
{
```

```
   leftMotor->run(RELEASE);
   rightMotor->run(FORWARD);
}

/*
 * Function to set motors to stop
 */
void halt()
{
   leftMotor->run(RELEASE);
   rightMotor->run(RELEASE);
}

/*
 * Transmitter input commands
 * This function will be executed and loop continuously for
our alternative
 * user input command instead of autonomous mode.
 *
 */
void userInput()
{
  while(true)
  {
    comfb = sbus.getChannel(3);    //Store width of PWM signal
from channel 3
    comrl = sbus.getChannel(2);    //Store width of PWM signal
from channel 2
    if(comrl==957)                 //Error handling. Width of
957 will be read periodically
      comrl=prevrl;                //so previous comfb/comrl
values are restored when this happens
    if(comfb==957)                 //to not have incorrect
commands sent
      comfb=prevfb;
      if((comfb>1100)&&(comfb<2000)&&(comrl>1100)&&(comrl<2000)
```

```
    {
       rightMotor->run(RELEASE);
       leftMotor->run(RELEASE);
      lcd.setCursor(0, 0);          // Output debugging info
     lcd.print("     Stop         ");
    }
    if(comfb<1100)
    {                  //Width of Ch.3 pulse when stick is in
reverse position
       rightMotor->run(BACKWARD);
       leftMotor->run(FORWARD);
      lcd.setCursor(0, 0);          // Output debugging info
     lcd.print("     Back         ");
    }
    if(comfb>2000)
    {                 //Width of Ch.3 pulse when stick is in
forward position
       rightMotor->run(FORWARD);
       leftMotor->run(BACKWARD);
      lcd.setCursor(0, 0);          // Output debugging info
     lcd.print("     Forward         ");
    }
    if(comrl<1100)
    {                  //Width of Ch.2 pulse when stick is in
left  position
       rightMotor->run(FORWARD);
       leftMotor->run(FORWARD);
      lcd.setCursor(0, 0);          // Output debugging info
     lcd.print("     Left         ");
    }
    if(comrl>2000)
    {                 //Width of Ch.2 pulse when stick is in
right  position
       rightMotor->run(BACKWARD);
       leftMotor->run(BACKWARD);
      lcd.setCursor(0, 0);          // Output debugging info
```

```
      lcd.print("     Right         ");
    }
     if (sbus.signalLossActive())
    {  //Stops wheels if signal is lost between reciever and
transmitter
       rightMotor->run(RELEASE);
       leftMotor->run(RELEASE);
      lcd.setCursor(0, 0);          // Output debugging info
      lcd.print("     Signal Lost      ");
    }

     if (sbus.failsafeActive())
    {     //Stops wheels if reciever failsafe triggers
       rightMotor->run(RELEASE);
       leftMotor->run(RELEASE);
      lcd.setCursor(0, 0);          // Output debugging info
      lcd.print("     Failsafe      ");
    }

    prevfb = comfb;                 //Store comfb value for
error handling on next loop
    prevrl = comrl;                 //Store comrl value for
error handling on next loop
     delay(200);
  }
}
```