

Make sure you have the correct libraries included. These should all be included in the Arduino IDE.

```
//Include the necessary libraries.  
#include <SPI.h>  
#include <WiFi.h>  
#include <WiFiUdp.h>
```

1

First, we define our Wifi variables. *Make sure to add your information for your ssid, password, keyIndex, and localPort or else you will not be able to connect to your wireless network.*

```
//Wifi Variables  
int status = WL_IDLE_STATUS;  
char ssid[] = "YOURNETWORK"; // your network SSID (name)  
char pass[] = "YOURPASSWORD"; // your network password (use for WPA, or use as key for WEP)  
int keyIndex = 0; // your network key Index number (needed only for WEP)  
  
unsigned int localPort = 2390; // local port to listen on  
  
char packetBuffer[255]; //buffer to hold incoming packet  
  
WiFiUDP Udp;
```

2

Next, we define our motor control variables. Each motor needs a PWM and enable pin. Make sure you clearly label each pin for each motor so you don't get them confused. Variables lX, lY, rX, and rY are integers for later when we parse the wireless packet.

```
// Motor Control Variables  
int PWM1 = 9;  
int ENABLE1 = 8;  
int PWM2 = 5;  
int ENABLE2 = 1;  
int PWM3 = 3;  
int ENABLE3 = 0;  
int PWM4 = 6;  
int ENABLE4 = 2;  
  
int lX, lY, rX, rY;
```

3

In the setup function, we set the enable pins for each motor as output. Then we check for the wifi shield, and if it is connected, we begin the Udp protocol.

```

void setup(){

    //IF YOU WANT TO TEST YOUR SHIELD AND SEE INFORMATION ON THE SERIAL MONITOR,
    //UNCOMMENT THE FOLLOWING FUNCTION.
    //IF YOU WANT TO RUN THE MOTORS, COMMENT IT OUT AGAIN.
    //Serial.begin(9600);

    //Set pinMode for the enable pins
    pinMode (ENABLE1, OUTPUT);
    pinMode (ENABLE2, OUTPUT);
    pinMode (ENABLE3, OUTPUT);
    pinMode (ENABLE4, OUTPUT);

    //UDP Configuration
    // check for the presence of the shield:
    if (WiFi.status() == WL_NO_SHIELD) {
        Serial.println("WiFi shield not present");
        // don't continue:
        while(true);
    }

    // attempt to connect to Wifi network:
    while ( status != WL_CONNECTED) {
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(ssid);
        // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
        status = WiFi.begin(ssid,pass);

        // wait 10 seconds for connection:
        delay(10000);
    }
    Serial.println("Connected to wifi");
    //printWifiStatus();

    Serial.println("\nStarting connection to server...");
    // if you get a connection, report back via serial:
    Udp.begin(localPort);
}

```

In the loop function, the wifi shield will read a packet when it is received. Four numbers, one for each of the X and Y axes of the left and right joysticks, should be received with each packet. The packet is then tokenized, using the space (" ") as the delimiter. The code takes the four numbers and assigns them to a variable (lX, lY, rX, and rY) based on which token the numbers are. This information is then used to command the motors.

```

void loop() {
  // if there's data available, read a packet
  int packetSize = Udp.parsePacket();
  if(packetSize)
  {
    Serial.print("Received packet of size ");
    Serial.println(packetSize);
    Serial.print("From ");
    IPAddress remoteIp = Udp.remoteIP();
    Serial.print(remoteIp);
    Serial.print(", port ");
    Serial.println(Udp.remotePort());

    // read the packet into packetBuffer
    int len = Udp.read(packetBuffer,255);
    if (len >0) packetBuffer[len]=0;
    Serial.println("Contents:");
    Serial.println(packetBuffer);

    char* pch;
    pch = strtok (packetBuffer," ");

    for(int i = 0; i < 4; i++)
    {
      if(i == 0) lX = atoi(pch);
      if(i == 1) lY = atoi(pch);
      if(i == 2) rX = atoi(pch);
      if(i == 3) rY = atoi(pch);
      pch = strtok (NULL," ");
    }
    CommandMotors(lY,rY);
  }
}

```

5

CommandMotors is a helper function. Each joystick has a range from 0 to 255 in the X and Y direction. For tank-style steering, only the Y values are used. Each direction has a “dead zone,” the center where the robot will stop. We found that the joystick does not necessarily always go back to the exact center (127), so a range of 120-135 is used to ensure that the robot stops when the joystick is near the center.

The digitalWrite function is used to turn the motor forward (HIGH) and backward (LOW).

When the joystick is pushed up ( $Y < 120$ ), the formula  $(Y-120)*-2.125$  is used because the speed should increase as Y decreases. When  $Y = 0$ , the speed should equal 255, and when  $Y = 120$ , the speed should equal zero. When the joystick is pushed down ( $Y > 135$ ), the formula  $(Y-135)*2.125$  is used because the speed should increase as Y increases. When  $Y = 255$ , the speed

should equal 255, and when  $Y = 135$ , the speed should equal zero. The robot will stop when the joystick is in the dead zone, so the speed is set to zero.

```
void CommandMotors(int LY, int RY)
{
    if(LY < 120) {
        digitalWrite (ENABLE1, HIGH);
        digitalWrite (ENABLE4, HIGH);
        analogWrite(PWM1, (LY-120)*-2.125);
        analogWrite(PWM4, (LY-120)*-2.125);
    }
    if(LY > 135) {
        digitalWrite (ENABLE1, LOW);
        digitalWrite (ENABLE4, LOW);
        analogWrite(PWM1, (LY-135)*2.125);
        analogWrite(PWM4, (LY-135)*2.125);
    }
    if(RY < 120) {
        digitalWrite (ENABLE2, HIGH);
        digitalWrite (ENABLE3, HIGH);
        analogWrite(PWM2, (RY-120)*-2.125);
        analogWrite(PWM3, (RY-120)*-2.125);
    }
    if(RY > 135) {
        digitalWrite (ENABLE2, LOW);
        digitalWrite (ENABLE3, LOW);
        analogWrite(PWM2, (RY-135)*2.125);
        analogWrite(PWM3, (RY-135)*2.125);
    }

    //STOP
    if(LY > 120 && LY < 135) {
        analogWrite(PWM1, 0);
        analogWrite(PWM4, 0);
    }
    if(RY > 120 && RY < 135) {
        analogWrite(PWM2, 0);
        analogWrite(PWM3, 0);
    }
}
```

6

printWifiStatus() is another helper function that prints information about the network on the serial monitor and is used when testing out the wireless network connection. This code comes

straight from the samples in the Arduino libraries.

```
void printWifiStatus() {  
  // print the SSID of the network you're attached to:  
  Serial.print("SSID: ");  
  Serial.println(WiFi.SSID());  
  
  // print your WiFi shield's IP address:  
  IPAddress ip = WiFi.localIP();  
  Serial.print("IP Address: ");  
  Serial.println(ip);  
  
  // print the received signal strength:  
  long rssi = WiFi.RSSI();  
  Serial.print("signal strength (RSSI):");  
  Serial.print(rssi);  
  Serial.println(" dBm");  
}
```

7