Appendix B – ESP8266 Code

```
/*************************************************
name: espcode_3_8_17.ino
author: E. Hunckler & S. McAndrew
adapted from: R. Schafer
date: 3-18-2017
comments: This expects a serial transmission in the form
receiver_indicator<data_packet>.  Ex: R1<125.2>
This then uplaods the data to the MQTT server in RM213 of Stinson-Remick.
*************************************************/
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
/*************************************************
Defintions
*************************************************/
#define WLAN_SSID       "SDNet"
#define WLAN_PASS       "CapstoneProject"
#define SERVER_ADDRESS  "192.168.1.136"  // server in 213 SR
#define SERVER_PORT     1883        // standard port
/*************************************************/

/*************************************************
 Global Variables
*************************************************/
String data_string;
String distance_topic;
char newcharacter;
int len;
boolean exitstringin = false;
/*************************************************/

/*************************************************
   MQTT Callback
*************************************************/
void callback(const MQTT::Publish& pub) {
  // handle message arrived
  Serial.print("Message arrived [");
  Serial.print(pub.topic());
  Serial.print("] ");

  Serial.println(pub.payload_string());

  Serial.println();
}// end of callback function
/*************************************************/
// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient wf_client; // instantiate wifi client
PubSubClient client(wf_client, SERVER_ADDRESS); // pass to pubsub

void setup() {
// Setup console
  Serial.begin(115200);
  delay(10);
```

```
  client.set_callback(callback);
// Connect to WiFi access point.
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);
//
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
//Connect to Client & Subscriptions
if (WiFi.status() == WL_CONNECTED) {
    if (!client.connected()) {
      client.connect("sender1");
      //Add topics to subscribe to
      //client.subscribe("Control Info");
    }
  }
  Serial.flush();
}

void loop() {

//Read String In from Serial Port
  char tagnumber;
  exitstringin = false;
  while (Serial.read() != 'R') {ESP.wdtFeed();}
  while (Serial.available() == 0) {ESP.wdtFeed();}
  tagnumber = (char)Serial.read();
  while (Serial.read() != '<') {ESP.wdtFeed();}
  //The incoming string starts here
  data_string = String("");
  while (!exitstringin)
  {
    while (Serial.available() == 0) {ESP.wdtFeed();}
    newcharacter = Serial.read();
    if (newcharacter == '>')
    {
      exitstringin = true;
      break;
    }
    else
    {
        data_string += newcharacter;
    }
  }

//Print Received Data
  Serial.print(tagnumber);
  Serial.print(": ");
  Serial.println(data_string);
```

```
 //Determine the Topic
  switch(tagnumber)
  {
    case '1':
      distance_topic = "R1";
      break;
    case '2':
      distance_topic = "R2";
      break;
    case '3':
      distance_topic = "R3";
        break;
    case '4':
      distance_topic = "R4";
      break;
    case 'X':
    distance_topic = "RX";
      break;
    case 'Y':
    distance_topic = "RY";
      break;
    case 'Z':
    distance_topic = "RZ";
      break;
    default:
      distance_topic = "Error";
      break;
  }

//Transmit MQTT data
  if (WiFi.status() == WL_CONNECTED)
  {
    if (client.connected())
    {
      client.publish(distance_topic,data_string);
      Serial.println("Sent to MQTT");
      client.loop();
    }
    else
    {
    client.connect("sender1");
    }
  }
}
```