#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#include <EthernetUdp.h> *//* *UDP* *library* *from:* *bjoern@cs.stanford.edu* *12/30/2008*

#include <stdio.h>

*/\*Connect* *a* *ESP8266* *to* *a* *network* *and* *assign* *itself* *a* *static* *IP* *IPAddress*

*then* *listen* *send* *a* *UDP* *packet* *with* *a* *delay* *interval* *in* *between*

*Sabrina* *Flemming*

*November* *2017* *\*/*

*/\*To* *upload* *onto* *specific* *port,* *type* *the* *following* *into* *terminal:*

*platformio* *run* *--target* *upload* *--upload-port* *COM8*

*-* *Example* *for* *port* *8\*/*

*//Don't* *forget* *to* *choose* *upload* *port* *under* *platformio.ini* *or* *in* *the* *terminal*

*/\*This* *code* *utilizes* *the* *code* *from* *ESP8266\_Connect\_to\_network* *and* *adds* *Code*

*based* *on* *the* *sketch:*

*http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-examples.html*

*\*/*

*//assign* *static* *IP* *address*

IPAddress ip(192, 168, 42, 2); *//raspberry* *pi* *IP* *address* *is* *192.168.42.1*

IPAddress gateway(192, 168, 42, 21);

IPAddress subnet(255, 255, 255, 0);

IPAddress broadcastIP(192, 168, 42, 255);

*//* *WiFi* *parameters* *to* *be* *configured*

const char\* ssid = "PI\_ME310";

const char\* password = "ME310\_criticalprototype";

*//WiFiUDP* *object* *called* *Udp* *(used* *to* *handle* *programming* *UDP* *over* *wifi)*

WiFiUDP Udp;

unsigned int localUdpPort = 4210; *//port* *4210* *is* *unregistered*

*//https://en.wikipedia.org/wiki/List\_of\_TCP\_and\_UDP\_port\_numbers*

*//sending* *and* *receiving* *port* *must* *be* *the* *same*

bool signalsent = false;

bool signalrecieved = false;

unsigned long rssi = 0;

unsigned long total\_rssi = 0;

unsigned long average\_rssi = 0;

unsigned int t\_sentSignal = 0;

unsigned int t\_returnSignal = 0;

unsigned long timelapsed\_signal = 0;

unsigned long total\_timelapsed\_signal = 0;

unsigned long average\_timelapsed\_signal = 0;

unsigned int exp\_dist = 0;

unsigned int exp\_dist1 = 0;

unsigned int loop\_count = 0;

unsigned int message\_count = 0;

char\* receive(const char replyPacket[]);

void send(const char outgoingMessage[]);

void setup() {

Serial.begin(115200);

delay(10);

*//* *Set* *static* *IP* *address*

WiFi.config(ip, gateway, subnet);

*//* *Connect* *to* *WiFi*

WiFi.begin(ssid, password);

*//* *while* *wifi* *not* *connected* *yet,* *print* *'.'*

*//* *then* *after* *it* *connected,* *get* *out* *of* *the* *loop*

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.println(WiFi.status());

*/\*Serial.println("\n");*

*Serial.println(WL\_CONNECTED);* *//3*

*Serial.println(WL\_CONNECT\_FAILED);* *//4*

*Serial.println(WL\_CONNECTION\_LOST);* *//5*

*Serial.println(WL\_NO\_SSID\_AVAIL);* *//1*

*Serial.println(WL\_SCAN\_COMPLETED);* *//2*

*Serial.println(WL\_DISCONNECTED);* *//6*

*Serial.println(WL\_NO\_SHIELD);* *//255*

*Serial.println(WL\_IDLE\_STATUS);* *//0* *\*/*

}

*//print* *a* *new* *line,* *then* *print* *WiFi* *connected* *and* *the* *IP* *address*

Serial.println("");

Serial.println("WiFi connected");

*//* *Print* *the* *IP* *address*

Serial.println(WiFi.localIP());

*//start* *listening* *to* *incoming* *UDP* *packets*

Udp.begin(localUdpPort);

}

void loop() {

char\* incoming = receive("Signal returned");

*//* *if* *a* *signal* *has* *been* *recieved* *update* *signal* *sent* *to* *allow* *another* *to* *be* *sent*

if (signalrecieved == true){

*//delay(1000);*

signalsent = false;

signalrecieved == false;

}

*//* *sends* *off* *a* *signal* *and* *waits* *until* *response* *is* *received* *before* *trying* *again*

if (signalsent == false) {

message\_count++;

*//char* *outgoingPacket[100];*

*//sprintf(outgoingPacket,"Hello* *World;* *loop* *counter* *=* *%d;* *message* *=* *%d\0",*

*//* *loop\_count,* *message\_count);*

*//outgoingPacket* *=* *"Hello* *World".c\_str();*

send("Hello World");

*//t\_sentSignal* *=* *micros();*

t\_sentSignal = ESP.getCycleCount();

signalsent == true;

}

if (message\_count < 10000)

total\_timelapsed\_signal += timelapsed\_signal;

total\_rssi += rssi;

if (message\_count >= 10000) {

average\_timelapsed\_signal = total\_timelapsed\_signal/message\_count;

average\_rssi = total\_rssi/message\_count;

exp\_dist = (average\_rssi - 62.476)/0.3115;

exp\_dist1 = (average\_rssi - 65.487)/0.2358;

printf("Average Time of signal %d microseconds with an RSSI strength of %d at an expected distance of %d, %d\n", average\_timelapsed\_signal, average\_rssi, exp\_dist, exp\_dist1);

message\_count = 0;

total\_timelapsed\_signal = 0;

average\_timelapsed\_signal = 0;

total\_rssi = 0;

average\_rssi = 0;

timelapsed\_signal = 0;

exp\_dist = 0;

exp\_dist1 = 0;

delay(1000);

}

*//* *incase* *nothing* *happens*

*//if* *((micros()* *-* *t\_sentSignal)* *>* *10000000)*

*//* *signalsent* *=* *FALSE;*

loop\_count++;

}

void send(const char outgoingMessage[]) {

uint8\_t ipBuffer[4] = {broadcastIP[0], broadcastIP[1], broadcastIP[2], broadcastIP[3]};

Udp.beginPacket(ipBuffer, localUdpPort);

*//Starts* *a* *connection* *to* *write* *UDP* *data* *to* *the* *remote* *connection*

*//Udp.remoteIP* *gets* *the* *IP* *address* *of* *the* *remote* *connection.*

*//Udp.MremotePort* *get* *the* *port* *of* *the* *remote* *connection.*

*//create* *the* *outgoingPacket* *which* *will* *include* *the* *counter* *number*

char outgoingPacket[100];

sprintf(outgoingPacket, "%s\0", outgoingMessage);

*//Serial.println(outgoingPacket);*

*//Send* *out* *the* *Udp* *packet*

Udp.write(outgoingPacket);

Udp.endPacket();

}

char\* receive(const char replyPacket[]) {

char incomingPacket[255]; *//character* *array* *of* *size* *255*

*//this* *needs* *to* *be* *reinitialized* *every* *time* *the* *receive* *function* *is* *called*

*//Udp.begin(localUdpPort);* *//here* *or* *in* *setup...*

*/\*Check* *for* *the* *presence* *of* *a* *UDP* *packet,* *and* *report* *the*

*size.* *parsePacket()* *must* *be* *called* *before* *reading* *the* *buffer* *with*

*UDP.read()* *\*/*

int packetSize = Udp.parsePacket(); *//*

*//long* *udprssi* *=* *Udp.RSSI();*

*//if* *the* *packet* *has* *a* *non-zero* *size* *(i.e.* *evaluates* *to* *TRUE)*

if (packetSize){

signalrecieved = true;

*//t\_returnSignal* *=* *micros();*

t\_returnSignal = ESP.getCycleCount();

rssi = abs(WiFi.RSSI());

timelapsed\_signal = 6.25\*(t\_returnSignal - t\_sentSignal);

*//unsigned* *int* *tof* *=* *6.25\*(t\_returnSignal* *-* *t\_sentSignal);*

*//Serial.printf("Sender:* *Received* *return* *signal* *(%d)* *after* *%d* *microseconds* *\n",* *rssi,* *timelapsed\_signal);*

int len = Udp.read(incomingPacket, 255);

*//read* *up* *to* *the* *255th* *place* *(the* *maximum* *length)*

*/\*Packet* *Length* *specify* *the* *size* *of* *the* *whole* *packet* *include* *header,*

*trailer* *and* *the* *data* *that* *send* *on* *that* *packet.* *But* *Packet* *Size* *specify*

*size* *of* *the* *header* *on* *the* *packet.* *\*/*

*//add* *a* *zero* *to* *the* *end* *of* *the* *incomingPacket* *to* *allow* *for* *proper* *printing*

if (len > 0){

incomingPacket[len] = 0;

}

*//Serial.printf("UDP* *packet* *contents:* *%s\n",* *incomingPacket);*

*/\*Once* *a* *packet* *is* *received,* *the* *code* *will* *printing* *out* *the* *IP* *address*

*and* *port* *of* *the* *sender* *as* *well* *as* *the* *length* *of* *received* *packet.* *If* *the*

*packet* *is* *not* *empty,* *its* *contents* *will* *be* *printed* *out* *as* *well.\*/*

*/\*To* *prevent* *an* *infinte* *loop* *or* *"Message* *Received"* *going* *back* *and* *forth,*

*we* *say* *that* *if* *the* *incoming* *message* *is* *"Message* *Received",* *do* *not* *reply\*/*

String packetString((char \*)incomingPacket);

if (!(packetString == "Message Received..")) {

Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

*//Starts* *a* *connection* *to* *write* *UDP* *data* *to* *the* *remote* *connection*

*//Udp.remoteIP* *gets* *the* *IP* *address* *of* *the* *remote* *connection.*

*//Udp.remotePort* *get* *the* *port* *of* *the* *remote* *connection.*

Udp.write(replyPacket);

Udp.endPacket();

}

}

*//return* *(const* *char)* *\*Udp.remoteIP().toString().c\_str();*

return incomingPacket;

}

------------------------------------------------------------------------------------------------------------------------------

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#include <EthernetUdp.h> *//* *UDP* *library* *from:* *bjoern@cs.stanford.edu* *12/30/2008*

#include <stdio.h>

*/\*Connect* *a* *ESP8266* *to* *a* *network* *and* *assign* *itself* *a* *static* *IP* *IPAddress*

*then* *listen* *send* *a* *UDP* *packet* *with* *a* *delay* *interval* *in* *between*

*Sabrina* *Flemming*

*November* *2017* *\*/*

*/\*To* *upload* *onto* *specific* *port,* *type* *the* *following* *into* *terminal:*

*platformio* *run* *--target* *upload* *--upload-port* *COM8*

*-* *Example* *for* *port* *8\*/*

*//Don't* *forget* *to* *choose* *upload* *port* *under* *platformio.ini* *or* *in* *the* *terminal*

*/\*This* *code* *utilizes* *the* *code* *from* *ESP8266\_Connect\_to\_network* *and* *adds* *Code*

*based* *on* *the* *sketch:*

*http://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/udp-examples.html*

*\*/*

*//assign* *static* *IP* *address*

IPAddress ip(192, 168, 42, 3); *//raspberry* *pi* *IP* *address* *is* *192.168.42.1*

IPAddress gateway(192, 168, 42, 21);

IPAddress subnet(255, 255, 255, 0);

IPAddress broadcastIP(192, 168, 42, 255);

*//* *WiFi* *parameters* *to* *be* *configured*

const char\* ssid = "PI\_ME310";

const char\* password = "ME310\_criticalprototype";

*//WiFiUDP* *object* *called* *Udp* *(used* *to* *handle* *programming* *UDP* *over* *wifi)*

WiFiUDP Udp;

unsigned int localUdpPort = 4210; *//port* *4210* *is* *unregistered*

*//https://en.wikipedia.org/wiki/List\_of\_TCP\_and\_UDP\_port\_numbers*

*//sending* *and* *receiving* *port* *must* *be* *the* *same*

unsigned int loop\_count = 0;

unsigned int message\_count = 0;

char\* receive(const char replyPacket[]);

void setup() {

Serial.begin(115200);

delay(10);

*//* *Set* *static* *IP* *address*

WiFi.config(ip, gateway, subnet);

*//* *Connect* *to* *WiFi*

WiFi.begin(ssid, password);

*//* *while* *wifi* *not* *connected* *yet,* *print* *'.'*

*//* *then* *after* *it* *connected,* *get* *out* *of* *the* *loop*

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.println(WiFi.status());

*/\*Serial.println("\n");*

*Serial.println(WL\_CONNECTED);* *//3*

*Serial.println(WL\_CONNECT\_FAILED);* *//4*

*Serial.println(WL\_CONNECTION\_LOST);* *//5*

*Serial.println(WL\_NO\_SSID\_AVAIL);* *//1*

*Serial.println(WL\_SCAN\_COMPLETED);* *//2*

*Serial.println(WL\_DISCONNECTED);* *//6*

*Serial.println(WL\_NO\_SHIELD);* *//255*

*Serial.println(WL\_IDLE\_STATUS);* *//0* *\*/*

}

*//print* *a* *new* *line,* *then* *print* *WiFi* *connected* *and* *the* *IP* *address*

Serial.println("");

Serial.println("WiFi connected");

*//* *Print* *the* *IP* *address*

Serial.println(WiFi.localIP());

*//start* *listening* *to* *incoming* *UDP* *packets*

Udp.begin(localUdpPort);

}

void loop() {

loop\_count++;

char\* incoming = receive("Message Received.");

}

char\* receive(const char replyPacket[]) {

char incomingPacket[255]; *//character* *array* *of* *size* *255*

*//this* *needs* *to* *be* *reinitialized* *every* *time* *the* *receive* *function* *is* *called*

*//Udp.begin(localUdpPort);* *//here* *or* *in* *setup...*

*/\*Check* *for* *the* *presence* *of* *a* *UDP* *packet,* *and* *report* *the*

*size.* *parsePacket()* *must* *be* *called* *before* *reading* *the* *buffer* *with*

*UDP.read()* *\*/*

int packetSize = Udp.parsePacket(); *//*

*//if* *the* *packet* *has* *a* *non-zero* *size* *(i.e.* *evaluates* *to* *TRUE)*

if (packetSize){

Serial.printf("Reciever: Received %d message in loop %d\n", message\_count,

loop\_count);

Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());

*//* *We* *want* *to* *send* *out* *a* *reply* *immediately* *to* *minimize* *time* *delays*

Udp.write(replyPacket);

Udp.endPacket();

message\_count++;

}

*//return* *(const* *char)* *\*Udp.remoteIP().toString().c\_str();*

return incomingPacket;

}