**APPENDIX C**

**TRAFFIC CONTROL PROGRAM**

#include <UIPEthernet.h>

EthernetUDP udp;

int led = 2;

unsigned long next;

char a;

int c;

String val;

char rec;

String lane1, lane2;

int latchPin1 = 2; //pin 12 on the 595

int dataPin1 = 3; //pin 14 on the 595

int clockPin1 = 4; //pin 11 on the 595

int g =6;

int y = 7;

int r = 8;

int number[10] = {63, 6, 91, 79, 102, 109, 125, 7, 127, 111};

void setup() {

Serial.begin(9600);

uint8\_t mac[6] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05};

Ethernet.begin(mac);

Serial.print("localIP: ");

Serial.println(Ethernet.localIP());

Serial.print("subnetMask: ");

Serial.println(Ethernet.subnetMask());

Serial.print("gatewayIP: ");

Serial.println(Ethernet.gatewayIP());

Serial.print("dnsServerIP: ");

Serial.println(Ethernet.dnsServerIP());

next = 0;

pinMode(latchPin1, OUTPUT);

pinMode(dataPin1, OUTPUT);

pinMode(clockPin1, OUTPUT);

pinMode(g,OUTPUT);

pinMode(y,OUTPUT);

pinMode(r,OUTPUT);

next = millis() + 5000;

digitalWrite(led, LOW);

}

void loop() {

int success;

int len = 0;

val = "";

rec = NULL;

a = NULL;

delay(1000);

//a = (char)Serial.read(); // read the incoming data as string

a = '2';

Serial.println(a);

resend:

do

{

success = udp.beginPacket(IPAddress(192, 168, 0, 101), 5000);

Serial.print("beginPacket: ");

Serial.println(success ? "success" : "failed");

}

while (!success < 0);

if (!success )

goto stop;

success = udp.write(a);

Serial.print("bytes written: ");

Serial.println(success);

success = udp.endPacket();

Serial.print("endPacket: ");

Serial.println(success ? "success" : "failed");

do

{

//check for new udp-packet:

success = udp.parsePacket();

}

while (!success < 0);

if (!success )

goto resend;

Serial.print("received: '");

do

{

c = udp.read();

Serial.write(c);

//Serial.print(c);

rec = c;

val += rec;

len++;

}

while ((success = udp.available()) > 0);

Serial.print("', ");

Serial.print(len);

Serial.println(" bytes");

lane1 = getValue(val, ':', 0);

lane2 = getValue(val, ':', 1);

Serial.print("fnum");

Serial.println(lane1);

Serial.print("Snum");

Serial.println(lane2);

ForDelay(latchPin1, dataPin1, clockPin1, number, lane1.toInt(), lane2.toInt());

digitalWrite(y,LOW);

digitalWrite(r,HIGH);

val = "";

rec = NULL;

udp.flush();

stop:

udp.stop();

Serial.println("Stop");

}

String getValue(String data, char separator, int index)

{

int found = 0;

int strIndex[] = { 0, -1 };

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

if (data.charAt(i) == separator || i == maxIndex) {

found++;

strIndex[0] = strIndex[1] + 1;

strIndex[1] = (i == maxIndex) ? i + 1 : i;

}

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

}

void ForDelay(int latchPin, int dataPin, int clockPin, int numb[], int fnum, int snum) {

bool check = false;

int upper = 0;

int lower = 0;

if (fnum > snum) {

ledoff();

digitalWrite(r,HIGH);

upper = fnum;

lower = snum;

} else {

ledoff();

digitalWrite(g,HIGH);

upper = snum;

lower = fnum;

check = true;

}

int diff\_num = abs(fnum - snum) - 1;

for (int k = upper; k >= 0; k--) {

int n1 = fnum / 10;

int n2 = fnum % 10;

if (fnum >= 0) {

writeTo595(latchPin, dataPin, clockPin, n1, n2);

Serial.println(snum);

Serial.println(fnum);

} else if (fnum <= 0 && check) {

digitalWrite(g,LOW);

digitalWrite(y,HIGH);

writeTo595(latchPin, dataPin, clockPin, 0, diff\_num--);

//Serial.println("ku");

}

fnum--;

snum--;

}

Serial.println("finish");

}

void writeTo595(int latchPin, int dataPin, int clockPin, int num1, int num2) {

digitalWrite(latchPin, LOW);

shiftOut(dataPin, clockPin, MSBFIRST, number[num1]);

shiftOut(dataPin, clockPin, MSBFIRST, number[num2]);

digitalWrite(latchPin, HIGH);

delay(1000);

}

void ledoff(){

digitalWrite(r,LOW);

digitalWrite(y,LOW);

digitalWrite(g,LOW);

}