VACANT PARKING DETECTOR 2.0

Senior Design Appendix

ECE 445 Team #21

Spring 2018

Qingtao Hu, Jiahe Liu, Zeyu Zhang

TA: Zhen Qin

1 Board Layouts

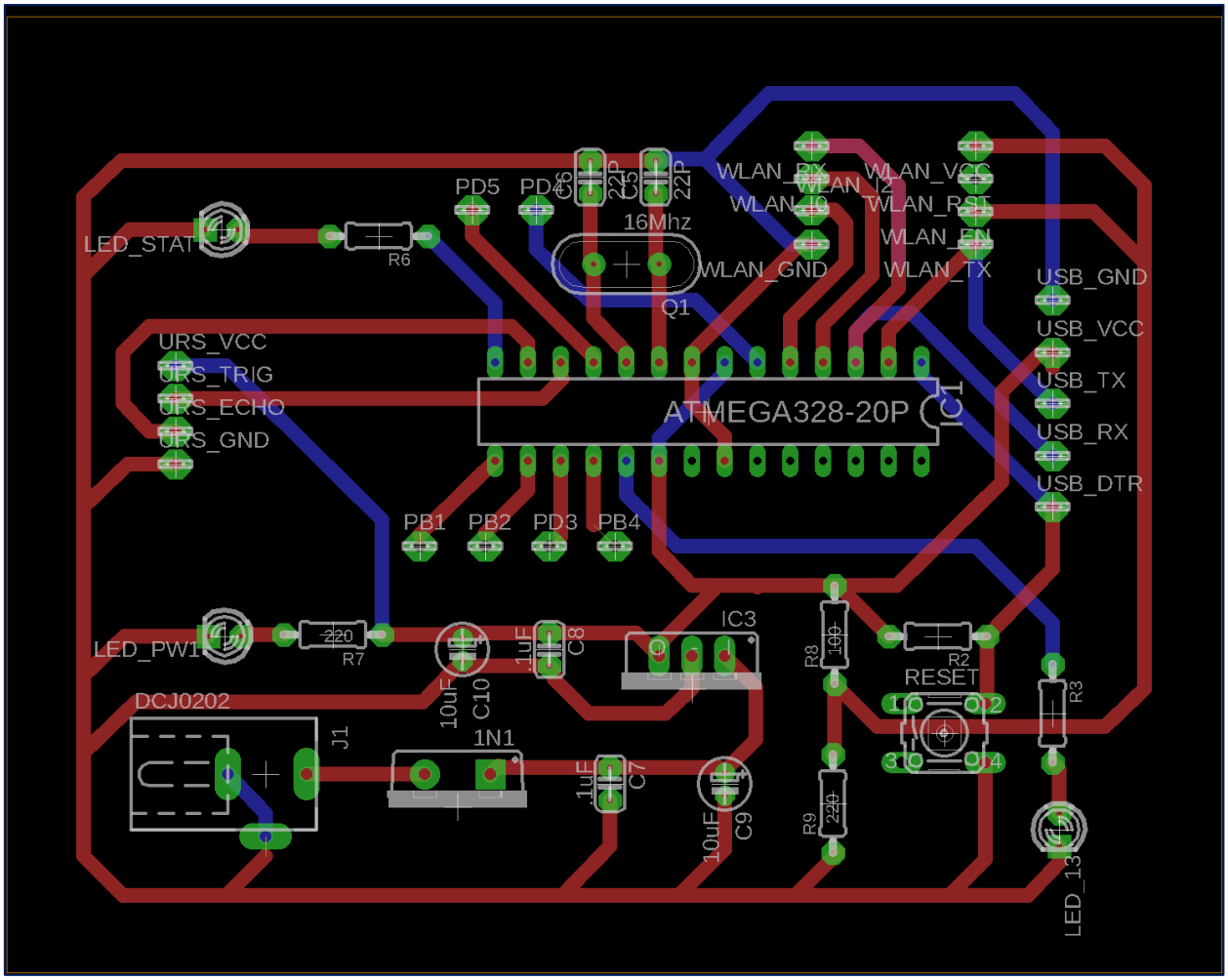

Figure 1: Sensor Module Board Layout

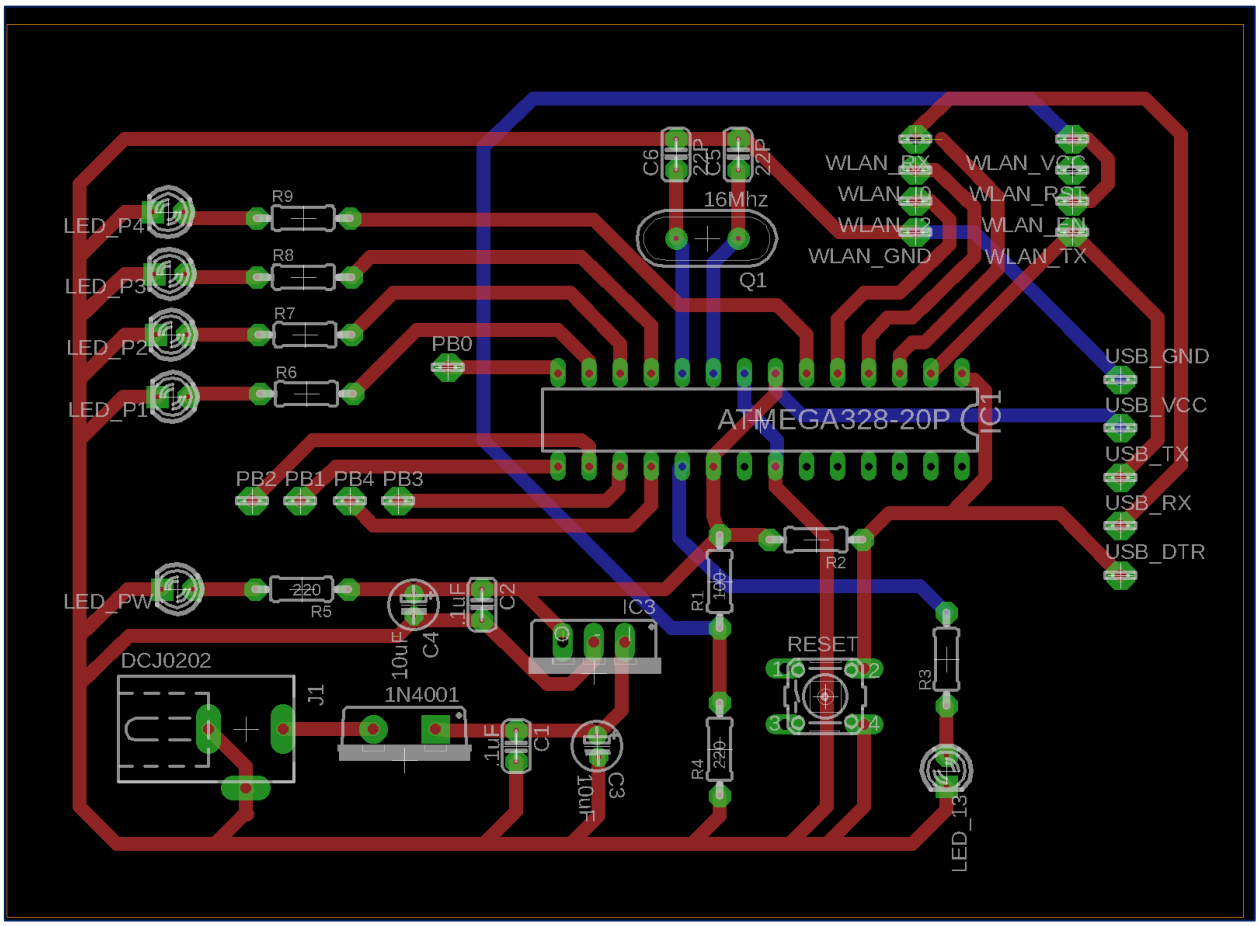

Figure 2: Control Module Board Layout

2 Arduino Sketches

2.1 Sensor Module ATMEGA328P:

```
/*
 * Sensor Module Sketch
 * Parts present:
 * HC-SR04 Ultrasonic Ranging Sensor (TIRG, ECHO)
 * ESP-01S WLAN IC (RX, TX, IO0, IO2)
 * ATMEGA328P Microcontroller ()
 */
// sensor related pin assignment
#define ursTrig 6
#define ursEcho 7
#define statLed 8

void setup() {
  // set up serial monitor frequency
  Serial.begin(115200);
  // initialize pin assignment
  pinMode(ursTrig, OUTPUT);
  pinMode(ursEcho, INPUT);
  pinMode(statLed, OUTPUT);
}

void loop() {
  // initialize variables
  long duration, distance;
  bool parked = false;
  // start detection sequence
  digitalWrite(ursTrig, LOW);
  delayMicroseconds(2);
  digitalWrite(ursTrig, HIGH);
  delayMicroseconds(10);
  digitalWrite(ursTrig, LOW);
  // calculate object distance based on duration
  duration = pulseIn(ursEcho, HIGH);
  distance = (duration*0.034)/2;
  // identify parked car if within 2m
  if(distance < 200) {
    digitalWrite(statLed, HIGH);
    parked = true;
  }
  else{
    digitalWrite(statLed, LOW);
    parked = false;
  }
  // print out detection in serial monitor
  // and send it to server
  Serial.print("Client{1} Status{");
  Serial.print(parked);
```

```
  Serial.print("} Distance{");
  Serial.print(distance);
  Serial.println("}");
  // repeat process every second
  delay(2000);
}
```

2.2 Sensor Module ESP-01S

```
/*
 * Connect client to the server w/ fixed IP address,
 * and transmit detection result to control module.
 */
#include <SPI.h>
#include <ESP8266WiFi.h>
// predefined variables
byte ledPin = 2;
char ssid[] = "Jiahe";
char pass[] = "zhangzeyu";
// predefined objects
IPAddress server(172,20,10,10);
WiFiClient client;

void setup() {
  Serial.begin(115200);
  // connect to router
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  pinMode(ledPin, OUTPUT);
}

void loop () {
  // connect to server via port 23
  if (client.connect(server, 23)) {
    digitalWrite(ledPin, HIGH);
    // fetch message from serial
    String stat = Serial.readString();
    // send message to server
    client.println(stat+'\r');
    client.flush();
    digitalWrite(ledPin, LOW);
    // set refreshing rate to 0.5Hz
    delay(2000);
  }
  else {
    digitalWrite(ledPin, HIGH);
  }
}
```

2.3 Control Module ESP-01S

```
/*  Connects to the home WiFi network
 *  Asks some network parameters
 *  Starts WiFi server with fix IP and listens
 *  Receives and sends messages to the client
 *  Communicates: wifi_client_01.ino
```

```
 */
#include <SPI.h>
#include <ESP8266WiFi.h>

byte ledPin = 2;
char ssid[] = "Jiahe";            // SSID of your home WiFi
char pass[] = "zhangzeyu";        // password of your home WiFi
// esp server
WiFiServer server(23);
// web server
WiFiServer server_(80);

IPAddress ip(172,20,10,10);         // IP address of the server
IPAddress gateway(172,20,10,1);   // gateway of your network
IPAddress subnet(255,255,255,0);// subnet mask of your network

String stat_1 = "vacant";
String stat_2 = "vacant";
String stat_3 = "vacant";
// Variable to store the HTTP request
String header;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(115200);                    // only for debug
  WiFi.config(ip, gateway, subnet);       // forces to use the fix IP
  WiFi.begin(ssid, pass);                  // connects to the WiFi router
  while (WiFi.status() != WL_CONNECTED) {
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
  }
  // esp server begins
  server.begin();
  Serial.println("esp server begins");
  server_.begin();
  Serial.println("web server begins");
}

void loop () {
  // handle web client
  WiFiClient client_ = server_.available();   // Listen for incoming client_s
  if (client_) {                             // If a new client_ connects,
    Serial.println("New Web Client.");          // print a message out in the
serial port
    String currentLine = "";                // make a String to hold incoming
data from the client_
    while (client_.connected()) {            // loop while the client_'s
connected
      if (client_.available()) {             // if there's bytes to read from
the client_,
        char c = client_.read();             // read a byte, then
        Serial.write(c);                    // print it out the serial
monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline
character
```

```
          // if the current line is blank, you got two newline characters in
a row.
          // that's the end of the client_ HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1
200 OK)
            // and a content-type so the client_ knows what's coming, then a
blank line:
            client_.println("HTTP/1.1 200 OK");
            client_.println("Content-type:text/html");
            client_.println("Connection: close");
            client_.println();

            // Display the HTML web page
            client_.println("<!DOCTYPE html><html>");
            client_.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client_.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size
attributes to fit your preferences
            client_.println("<style>html { font-family: Helvetica; display:
inline-block; margin: 0px auto; text-align: center;}");
            client_.println(".button { background-color: #FF0000; border:
none; color: white; padding: 16px 40px;");
            client_.println("text-decoration: none; font-size: 30px; margin:
2px; cursor: pointer;}");
            client_.println(".button2 {background-color:
#00FF00;}</style></head>");

            client_.println("<body><h1>Parking Monitoring System 2.0</h1>");
            client_.println("<p>Lot #1 " + stat_1 + "</p>");
            if (stat_1 == "occupied") {
              client_.println("<p><button
class=\"button\">OCP</button></p>");
            } else {
              client_.println("<p><button class=\"button
button2\">VCT</button></p>");
            }
            client_.println("<p>Lot #2 " + stat_2 + "</p>");
            if (stat_2 == "occupied") {
              client_.println("<p><button
class=\"button\">OCP</button></p>");
            } else {
              client_.println("<p><button class=\"button
button2\">VCT</button></p>");
            }
            client_.println("</body></html>");

            // The HTTP response ends with another blank line
            client_.println();
            // Break out of the while loop
            break;
          } else { // if you got a newline, then clear currentLine
            currentLine = "";
          }
```

```
        } else if (c != '\r') {  // if you got anything else but a carriage
return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client_.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
  // handle esp client
  WiFiClient client = server.available();
  int loc = 0;
  int stat = 0;
  if (client) {
    Serial.println("New ESP Client.");
    if (client.connected()) {
      digitalWrite(ledPin, LOW);  // to show the communication only (inverted
logic)
      String request = client.readStringUntil('\r');    // receives the
message from the client
      Serial.println(request);
      if (request != "") {
        loc = int(request[7]-'0');
        stat = int(request[17]-'0');
        if(loc == 1){
          if(stat == 1) {stat_1 = "occupied";}
          else {stat_1 = "vacant";}
        }
        else if(loc == 2){
          if(stat == 1) {stat_2 = "occupied";}
          else {stat_2 = "vacant";}
        }
      }
      client.flush();
      digitalWrite(ledPin, HIGH);
    }
    client.stop();                // tarminates the connection with the
client
  }
}
```