

Number Display

Frank Li

July 2021

1 Introduction

In Figure 4, it is a small electronic device that will display a number of your choosing. In order to display a number, the user has to first log into the ESP8266 local server, and from there, user can input a number to the electronic device, increments or decrements that number, or display the last digit of bitcoin at that moment. This device has both hardware and software component, and the details about each respective design will be given below.

2 Motivation

This project is a way for me to implement what I have learned in electrical engineering and computer science, and put them together to achieve an embedded system. Since the schools went remote all an entire year, there has been little or no in-person lab components of my engineering classes. I believe it is essential for me, as a future engineer, to have as many hands-on experiences as I can. And this project is a perfect way for me to practice my circuit design, soldering, and many more.

3 First Design

In Figure 1, it is the first iteration of the project. In this iteration, it can only display the last digit of bitcoin. The functions and design process will be explained below.

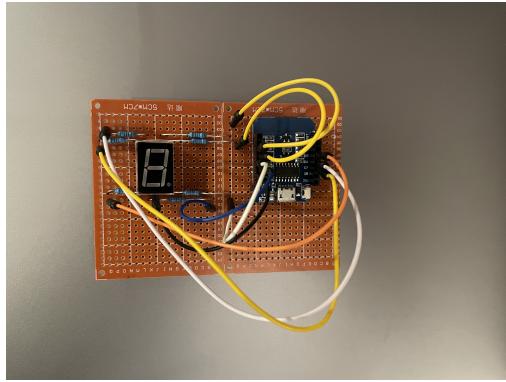


Figure 1: First Design

3.1 Hardware

The hardware part of the Number Display is fairly straightforward, it concerned about two simple concepts in electrical engineering: voltage divider and Ohms Law. Let's start with Ohms Law:

$$V = I * R$$

Ohms Law describes the relationship between current, resistance and voltage. If we have two elements known, then we can find the third one by using this equation. As for voltage divider, it simply states that when there are two resistors in series, the voltage of each component is proportional to each resistance.

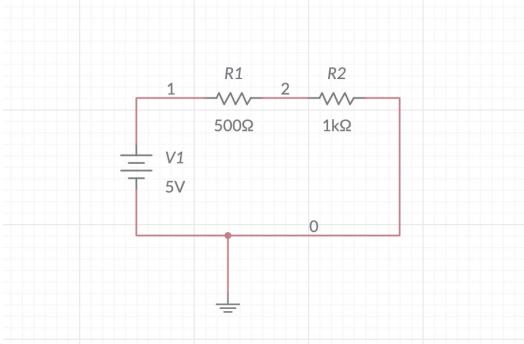


Figure 2: Voltage Division

As shown in Figure 2, the input voltage is 5v, and since the resistors are in series, we can use Ohms Law and Voltage Division to calculate the voltage across each resistor. If we want to designate 1k resistor as Output Voltage, R2, and 500 resistor as R1, then it can be written as:

$$V_{out} = \frac{V_{in}}{R1 + R2} * R2$$

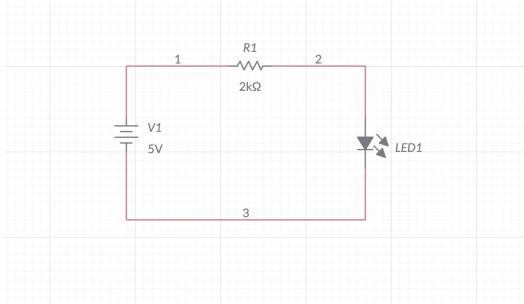


Figure 3: Simple LED Circuit

These two concepts are important in designing the circuits because if there is no resistor between the input voltage and the LED, then the LED will be burned. However, in order to figure out how much voltage should be in LED, we need to look into the Data sheet for the typical voltage in the LED and then choose a good resistance for the voltage division. And in my case, either 1k or 2k resistor will do the work. In Figure 3, a 2k resistor will do the work.

3.1.1 Electronics

The electronics involved in this projects are resistors, a seven segment display, and a ESP8266. As shown in Figure 3, resistor is used for voltage division. Since seven segment display is consists of many LED, Figure 3 is an illustration of the connection to one of the seven LED. The seven segment display I am using is Common Cathode, therefore COM pin should be connected to the ground. As for ESP8266, it is the WiFi module that will be able to connect to a local WiFi network. In my case, it also serves as a micro controller to manage the outputs to the seven segment display.

3.1.2 Assembly

The assembly process is basically soldering, however, it would be extremely hard to describe the soldering process in this paper. Therefore, I can only describe a few key things during the assembly. The first being

that soldering has safety concerns. In all times, the user should wear a safety goggles, and be absolutely careful to not have any contact with the iron for it is burning hot. Secondly, For the first design, I have made the mistake by soldering the jumper wires onto the perfboard. Being this is the first time, I have doing anything like this, I think it is understandable why I would make such mistake. And finally, be sure to not creating short circuit during the soldering process(e.g. do not put solder into places that will mess up the original circuit).

3.2 Code

Here is the software part of the project, I am thankful for Brian Lough[1] for putting up a sample code of using ESP8266 to make API request. This code, written in arduino C, is first going to connect to the local WIFI network, then it is going to the targeted website, cryptonator.com, to fetch the bitcoin data. Then with the get request, it will use the arduino Json library to parse the get request and display the bitcoin last digit onto the seven segment display.

```

1 #include <ESP8266WiFi.h>
2 #include <WiFiClientSecure.h>
3
4 // -----
5 // Additional Libraries - each one of these will need to be installed.
6 // -----
7
8 #include <ArduinoJson.h>
9 // Library used for parsing Json from the API responses
10
11 // Search for "Arduino Json" in the Arduino Library manager
12 // https://github.com/bblanchon/ArduinoJson
13
14 //----- Replace the following! -----
15 char ssid[] = "*****";           // your network SSID (name)
16 char password[] = "*****";      // your network key
17
18 // For Non-HTTPS requests
19 // WiFiClient client;
20
21 // For HTTPS requests
22 WiFiClientSecure client;
23
24
25 // Just the base of the URL you want to connect to
26 #define TEST_HOST "api.cryptonator.com"
27
28 // OPTIONAL - The finferprint of the site you want to connect to.
29 #define TEST_HOST_FINGERPRINT "10 76 19 6B E9 E5 87 5A 26 12 15 DE 9F 7D 3B 92 9A 7F 30 13"
30 // The finger print will change every few months.
31
32
33 void setup() {
34
35   pinMode(5, OUTPUT);
36   pinMode(4, OUTPUT);
37   pinMode(0, OUTPUT);
38   pinMode(2, OUTPUT);
39   pinMode(14, OUTPUT);
40   pinMode(12, OUTPUT);
41   pinMode(13, OUTPUT);
42   Serial.begin(115200);
43
44   // Connect to the WiFi
45   WiFi.mode(WIFI_STA);
46   WiFi.disconnect();
47   delay(100);
48
49   // Attempt to connect to Wifi network:
50   Serial.print("Connecting Wifi: ");

```

```

51 Serial.println(ssid);
52 WiFi.begin(ssid, password);
53 while (WiFi.status() != WL_CONNECTED) {
54     Serial.print(".");
55     delay(500);
56 }
57 Serial.println("");
58 Serial.println("WiFi connected");
59 Serial.println("IP address: ");
60 IPAddress ip = WiFi.localIP();
61 Serial.println(ip);
62 //-----
63
64 // If you don't need to check the fingerprint
65 // client.setInsecure();
66
67 // If you want to check the fingerprint
68 client.setFingerprint(TEST_HOST_FINGERPRINT);
69
70
71 }
72
73 void makeHTTPRequest() {
74
75     // Opening connection to server (Use 80 as port if HTTP)
76     if (!client.connect(TEST_HOST, 443))
77     {
78         Serial.println(F("Connection failed"));
79         return;
80     }
81
82     // give the esp a breather
83     yield();
84
85
86     // Send HTTP request
87     client.print(F("GET "));
88     // This is the second half of a request (everything that comes after the base URL)
89     client.print("/api/ticker/btc-usd"); // %2C == ,
90     client.println(F(" HTTP/1.1"));
91
92     //Headers
93     client.print(F("Host: "));
94     client.println(TEST_HOST);
95
96     client.println(F("Cache-Control: no-cache"));
97
98     if (client.println() == 0)
99     {
100         Serial.println(F("Failed to send request"));
101         return;
102     }
103     //delay(100);
104     // Check HTTP status
105     char status[32] = {0};
106     client.readBytesUntil('\r', status, sizeof(status));
107     if (strcmp(status, "HTTP/1.1 200 OK") != 0)
108     {
109         Serial.print(F("Unexpected response: "));
110         Serial.println(status);
111         return;
112     }
113
114     // Skip HTTP headers
115     char endOfHeaders[] = "\r\n\r\n";
116     if (!client.find(endOfHeaders))
117     {
118         Serial.println(F("Invalid response"));

```

```

119     return;
120 }
121
122 // This is probably not needed for most, but I had issues
123 // with the Tindie api where sometimes there were random
124 // characters coming back before the body of the response.
125 // This will cause no hard to leave it in
126 // peek() will look at the character, but not take it off the queue
127 while (client.available() && client.peek() != '{')
128 {
129     char c = 0;
130     client.readBytes(&c, 1);
131     Serial.print(c);
132     Serial.println("BAD");
133 }
134
135 // // While the client is still available read each
136 // // byte and print to the serial monitor
137 // while (client.available()) {
138 //     char c = 0;
139 //     client.readBytes(&c, 1);
140 //     Serial.print(c);
141 // }
142
143 int valid = bitCoinPrice();
144 if (valid < 0){
145     return;
146 }else{
147     Serial.print("Bitcoin most significant digit: ");
148     Serial.println(valid);
149     //5 to A, 4 to B and 0 to C 2 to D 14 to E 12 to F 13 to G
150     switch(valid){
151         case 0:
152             digitalWrite(5, HIGH);
153             digitalWrite(4, HIGH);
154             digitalWrite(0, HIGH);
155             digitalWrite(2, HIGH);
156             digitalWrite(14, HIGH);
157             digitalWrite(12, HIGH);
158             break;
159         case 1:
160             digitalWrite(4, HIGH);
161             digitalWrite(0, HIGH);
162             break;
163         case 2:
164             digitalWrite(5, HIGH);
165             digitalWrite(4, HIGH);
166             digitalWrite(2, HIGH);
167             digitalWrite(14, HIGH);
168             digitalWrite(13, HIGH);
169             break;
170         case 3:
171             digitalWrite(5, HIGH);
172             digitalWrite(4, HIGH);
173             digitalWrite(0, HIGH);
174             digitalWrite(2, HIGH);
175             digitalWrite(13, HIGH);
176             Serial.println("Good");
177             break;
178         case 4:
179             digitalWrite(4, HIGH);
180             digitalWrite(0, HIGH);
181             digitalWrite(12, HIGH);
182             digitalWrite(13, HIGH);
183             break;
184         case 5:
185             digitalWrite(5, HIGH);
186             digitalWrite(0, HIGH);

```

```

187     digitalWrite(2, HIGH);
188     digitalWrite(12, HIGH);
189     digitalWrite(13, HIGH);
190     break;
191 case 6:
192     digitalWrite(5, HIGH);
193     digitalWrite(13, HIGH);
194     digitalWrite(0, HIGH);
195     digitalWrite(2, HIGH);
196     digitalWrite(14, HIGH);
197     digitalWrite(12, HIGH);
198     break;
199 case 7:
200     digitalWrite(5, HIGH);
201     digitalWrite(4, HIGH);
202     digitalWrite(0, HIGH);
203     break;
204 case 8:
205     digitalWrite(5, HIGH);
206     digitalWrite(4, HIGH);
207     digitalWrite(0, HIGH);
208     digitalWrite(2, HIGH);
209     digitalWrite(14, HIGH);
210     digitalWrite(12, HIGH);
211     digitalWrite(13, HIGH);
212     break;
213 case 9:
214     digitalWrite(5, HIGH);
215     digitalWrite(4, HIGH);
216     digitalWrite(0, HIGH);
217     digitalWrite(12, HIGH);
218     digitalWrite(13, HIGH);
219     break;
220 }
221 }
222 }
223 }
224
225 int bitCoinPrice() {
226 // Stream& input;
227
228 DynamicJsonDocument doc(384);
229
230 DeserializationError error = deserializeJson(doc, client);
231
232 if (error) {
233 Serial.print(F("deserializeJson() failed: "));
234 Serial.println(error.f_str());
235 return -1;
236 }
237
238 JsonObject ticker = doc["ticker"];
239 const char* price = ticker["price"]; // "33908.95216874"
240 char temp = price[strlen(price) - 10];
241 int bitPrice = temp - '0';
242 return bitPrice;
243 }
244 void loop() {
245 // put your main code here, to run repeatedly:
246 makeHttpRequest();
247 delay(2000);
248 }
```

4 Second Design

As shown in Figure 1, the second iteration is cleaner and I did not solder any jumper wire this time. The hardware aspects are the same, but there are more functions in the software part.

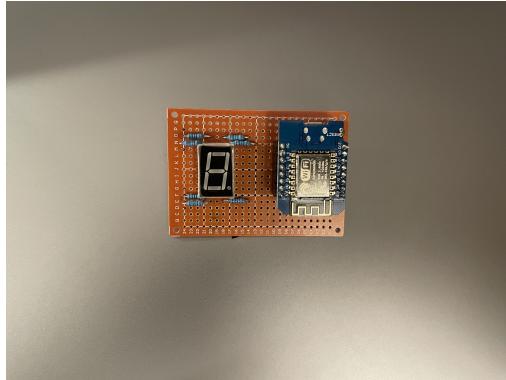


Figure 4: Second Design

4.1 Hardware

The hardware part is exactly the same as the first design. The two key concepts in the second iteration are still Ohms Law and Voltage Divider.

4.1.1 Electronics

The electronic part is exactly the same as the first design. The components used are still resistors, seven segment display and ESP8266.

4.1.2 Assembly

The assembly process in the second iteration is very similar to the first design except I did not use jumper wires this time. All the connections are on the backside. As shown in Figure 5, I used 22 gauge wire to make all the connections. One thing to be careful is that, do not let wire touch each other because it will create a short circuit and mess up the whole system.

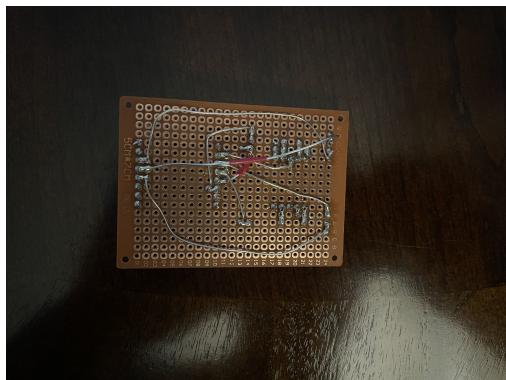


Figure 5: Second Design Soldering

4.2 Code

The second iteration adds a few features compare to the first design. First, the ESP8266 is a server, and user can log into this server and interact with the system. Second, the ESP8266 will not only show the last digit of bitcoin, it can also allow user to input a number in its local website, and user can increments or decrements that number. I would like to thank circuit4you.com [2] for providing me a sample of handling HTML forms and input information on ESP8266.

```
1 #include <ESP8266WiFi.h>
2 #include <ESP8266WebServer.h>
3 #include <WiFiClientSecure.h>
4
5 // -----
6 // Additional Libraries - each one of these will need to be installed.
7 // -----
8
9 #include <ArduinoJson.h>
10 // Library used for parsing Json from the API responses
11
12 // Search for "Arduino Json" in the Arduino Library manager
13 // https://github.com/bblanchon/ArduinoJson
14
15 //----- Replace the following! -----
16 char ssid[] = "*****";           // your network SSID (name)
17 char password[] = "*****";       // your network key
18 // For Non-HTTPS requests
19 // WiFiClient client;
20
21 // For HTTPS requests
22 WiFiClientSecure client;
23
24
25 // Just the base of the URL you want to connect to
26 #define TEST_HOST "api.cryptonator.com"
27
28 // OPTIONAL - The finferprint of the site you want to connect to.
29 #define TEST_HOST_FINGERPRINT "10 76 19 6B E9 E5 87 5A 26 12 15 DE 9F 7D 3B 92 9A 7F 30 13"
30 // The finger print will change every few months.
31
32 const char MAIN_page[] PROGMEM = R"=====(
33 <!DOCTYPE html>
34 <html>
35 <title>Number Display</title>
36 <style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}
37 body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto 30px;} h3 {color: #444444;
38   margin-bottom: 50px;}
39 .button {display: block; width: 80px; background-color: #1abc9c; border: none; color: white;
40   padding: 13px 30px; text-decoration: none; font-size: 25px; margin: 0px auto 35px; cursor: pointer; border-radius: 4px;}
41 .button:active {background-color: #16a085;}
42 p {font-size: 14px; color: #888; margin-bottom: 10px;}
43 </style>
44 <body>
45 <h2>A Number Display</h2>
46 <h3> By Frank Li</h3>
47 <form action="/input_page">
48   <label for="number">Enter a Number between 0-9:</label>
49   <input type="number" id="number" name="number" min="0" max="9">
50   <input type="submit" value="Submit">
51 </form>
52
53 <p>Increment Button</p><a class="button" href="#increment">Increment</a>
54 <p>Decrement Button</p><a class="button" href="#decrement">Decrement</a>
55 )=====
```

```

56 <p>Display Bitcoin Last Digit</p><a class="button" href="bitcoin">Bitcoin</a>
57 </body>
58 </html>
59 )=====";
60
61
62 ESP8266WebServer server(80);
63 int globalNum;
64
65 void handleRoot() {
66     String s = MAIN_page;
67     server.send(200, "text/html", s);
68 }
69
70 void handleForm() {
71     int number = getNumber();
72     globalNum = number;
73     String s = MAIN_page;
74     server.send(200, "text/html", s);
75     chooseNumber(number);
76 }
77
78 void handleIncrement() {
79     String s = MAIN_page;
80     int input = globalNum;
81     Serial.println(input);
82     int number = (input + 1)%10;
83     globalNum = number;
84     server.send(200, "text/html", s);
85     chooseNumber(number);
86 }
87
88 void handleDecrement() {
89     String s = MAIN_page;
90     int input = globalNum;
91     Serial.println(input);
92     int number = ((input - 1)%10);
93     if(number < 0) number=number+10;
94     globalNum = number;
95     server.send(200, "text/html", s);
96     chooseNumber(number);
97 }
98
99 void handleBitcoin() {
100    String s = MAIN_page;
101    makeHTTPRequest();
102    server.send(200, "text/html", s);
103 }
104 void setup() {
105
106     pinMode(5, OUTPUT);
107     pinMode(4, OUTPUT);
108     pinMode(0, OUTPUT);
109     pinMode(2, OUTPUT);
110     pinMode(14, OUTPUT);
111     pinMode(12, OUTPUT);
112     pinMode(13, OUTPUT);
113     Serial.begin(115200);
114
115     // Connect to the WiFi
116     WiFi.mode(WIFI_STA);
117     WiFi.disconnect();
118     delay(100);
119
120     // Attempt to connect to Wifi network:
121     Serial.print("Connecting Wifi: ");
122     Serial.println(ssid);
123     WiFi.begin(ssid, password);

```

```

124  while (WiFi.status() != WL_CONNECTED) {
125      Serial.print(".");
126      delay(500);
127  }
128  Serial.println("");
129  Serial.println("WiFi connected");
130  Serial.println("IP address: ");
131  IPAddress ip = WiFi.localIP();
132  Serial.println(ip);
133
134 //-----
135
136 // If you don't need to check the fingerprint
137 // client.setInsecure();
138
139 // If you want to check the fingerprint
140 client.setFingerprint(TEST_HOST_FINGERPRINT);
141 server.on("/", handleRoot);
142 server.on("/input_page", handleForm);
143 server.on("/increment", handleIncrement);
144 server.on("/decrement", handleDecrement);
145 server.on("/bitcoin", handleBitcoin);
146 server.begin();
147 Serial.println("HTTP started");
148 }
149
150 void makeHTTPRequest() {
151
152 // Opening connection to server (Use 80 as port if HTTP)
153 if (!client.connect(TEST_HOST, 443))
154 {
155     Serial.println(F("Connection failed"));
156     return;
157 }
158
159 // give the esp a breather
160 yield();
161
162 // Send HTTP request
163 client.print(F("GET "));
164 // This is the second half of a request (everything that comes after the base URL)
165 client.print("/api/ticker/btc-usd"); // %2C == ,
166 client.println(F(" HTTP/1.1"));
167
168 //Headers
169 client.print(F("Host: "));
170 client.println(TEST_HOST);
171
172 client.println(F("Cache-Control: no-cache"));
173
174 if (client.println() == 0)
175 {
176     Serial.println(F("Failed to send request"));
177     return;
178 }
179 //delay(100);
180 // Check HTTP status
181 char status[32] = {0};
182 client.readBytesUntil('\r', status, sizeof(status));
183 if (strcmp(status, "HTTP/1.1 200 OK") != 0)
184 {
185     Serial.print(F("Unexpected response: "));
186     Serial.println(status);
187     return;
188 }
189
190 // Skip HTTP headers
191 char endOfHeaders[] = "\r\n\r\n";

```

```

192 if (!client.find(endOfHeaders))
193 {
194     Serial.println(F("Invalid response"));
195     return;
196 }
197
198 // This is probably not needed for most, but I had issues
199 // with the Tindie api where sometimes there were random
200 // characters coming back before the body of the response.
201 // This will cause no hard to leave it in
202 // peek() will look at the character, but not take it off the queue
203 while (client.available() && client.peek() != '{')
204 {
205     char c = 0;
206     client.readBytes(&c, 1);
207     Serial.print(c);
208     Serial.println("BAD");
209 }
210
211 // // While the client is still available read each
212 // // byte and print to the serial monitor
213 // while (client.available()) {
214 //     char c = 0;
215 //     client.readBytes(&c, 1);
216 //     Serial.print(c);
217 // }
218
219 int valid = bitcoinPrice();
220 if (valid < 0){
221     return;
222 }else{
223     Serial.print("Bitcoin most significant digit: ");
224     Serial.println(valid);
225     //5 to A, 4 to B and 0 to C 2 to D 14 to E 12 to F 13 to G
226     chooseNumber(valid);
227 }
228 }
229 void clear() {
230     digitalWrite(5, LOW);
231     digitalWrite(4, LOW);
232     digitalWrite(0, LOW);
233     digitalWrite(2, LOW);
234     digitalWrite(12, LOW);
235     digitalWrite(13, LOW);
236     digitalWrite(14, LOW);
237 }
238 void chooseNumber(int input) {
239     Serial.println(input);
240     clear();
241     switch(input){
242         case 0:
243             digitalWrite(5, HIGH);
244             digitalWrite(4, HIGH);
245             digitalWrite(0, HIGH);
246             digitalWrite(2, HIGH);
247             digitalWrite(14, HIGH);
248             digitalWrite(12, HIGH);
249             //Serial.print(0);
250             break;
251         case 1:
252             digitalWrite(4, HIGH);
253             digitalWrite(0, HIGH);
254             //Serial.print(1);
255             break;
256         case 2:
257             digitalWrite(5, HIGH);
258             digitalWrite(4, HIGH);
259             digitalWrite(2, HIGH);

```

```

260     digitalWrite(14, HIGH);
261     digitalWrite(13, HIGH);
262     //Serial.print(2);
263     break;
264 case 3:
265     digitalWrite(5, HIGH);
266     digitalWrite(4, HIGH);
267     digitalWrite(0, HIGH);
268     digitalWrite(2, HIGH);
269     digitalWrite(13, HIGH);
270     //Serial.print(3);
271     break;
272 case 4:
273     digitalWrite(4, HIGH);
274     digitalWrite(0, HIGH);
275     digitalWrite(12, HIGH);
276     digitalWrite(13, HIGH);
277     //Serial.print(4);
278     break;
279 case 5:
280     digitalWrite(5, HIGH);
281     digitalWrite(0, HIGH);
282     digitalWrite(2, HIGH);
283     digitalWrite(12, HIGH);
284     digitalWrite(13, HIGH);
285     //Serial.print(5);
286     break;
287 case 6:
288     digitalWrite(5, HIGH);
289     digitalWrite(13, HIGH);
290     digitalWrite(0, HIGH);
291     digitalWrite(2, HIGH);
292     digitalWrite(14, HIGH);
293     digitalWrite(12, HIGH);
294     //Serial.print(6);
295     break;
296 case 7:
297     digitalWrite(5, HIGH);
298     digitalWrite(4, HIGH);
299     digitalWrite(0, HIGH);
300     //Serial.print(7);
301     break;
302 case 8:
303     digitalWrite(5, HIGH);
304     digitalWrite(4, HIGH);
305     digitalWrite(0, HIGH);
306     digitalWrite(2, HIGH);
307     digitalWrite(14, HIGH);
308     digitalWrite(12, HIGH);
309     digitalWrite(13, HIGH);
310     //Serial.print(8);
311     break;
312 case 9:
313     digitalWrite(5, HIGH);
314     digitalWrite(4, HIGH);
315     digitalWrite(0, HIGH);
316     digitalWrite(2, HIGH);
317     digitalWrite(12, HIGH);
318     digitalWrite(13, HIGH);
319     //Serial.print(9);
320     break;
321 }
322 }
323
324 int getNumber() {
325     String inputNumber = server.arg("number");
326     char temp = inputNumber[0];
327     int number = temp - '0';

```

```

328     return number;
329 }
330 int bitcoinPrice() {
331     // Stream& input;
332
333     DynamicJsonDocument doc(384);
334
335     DeserializationError error = deserializeJson(doc, client);
336
337     if (error) {
338         Serial.print(F("deserializeJson() failed: "));
339         Serial.println(error.f_str());
340         return -1;
341     }
342
343     JsonObject ticker = doc["ticker"];
344     const char* price = ticker["price"]; // "33908.95216874"
345     char temp = price[strlen(price) - 10];
346     int bitPrice = temp - '0';
347     return bitPrice;
348 }
349 void loop() {
350     // put your main code here, to run repeatedly:
351     //makeHTTPRequest();
352     server.handleClient();
353 }
```

5 Conclusion

This project is a great way to put together what I have learned in computer science and electrical engineering and create an embedded system. Throughout the project, I became better at soldering, writing code and writing out circuit diagrams. I also have the chance to learn about ESP8266 and its many great libraries. In the end, it might be the first project that I have implemented both software and hardware, but it is stepping a stone for something larger later on.

6 References

- [1] Brian Lough, May 24, 2021, arduino-sample-api-request,
<https://github.com/witnessmenow/arduino-sample-api-request>
- [2] Circuit4you.com, March 20, 2019,
<https://circuits4you.com/2019/03/20/esp8266-receive-post-get-request-data-from-website/>