

Department of Electronic and Telecommunication Engineering

Internet of Things
EN3250

Currency Converter Service Project Report



University of Moratuwa

170258L : R.H.R. Jayarathne
170543G : M.K.T. Sampath
170698J : L.T.A. Wijayaratne
170375R : D.R. Marasinghe

This report is submitted in partial fulfilment of the requirements for the module
EN3250 - Internet of Things

Contents

1	Introduction	2
1.1	Overview	2
1.2	Objectives	2
1.3	Scope	2
1.4	Architecture	2
2	Methodology	3
2.1	Exchange API	3
2.2	Integrating Firebase	3
2.3	Creating the Node-Red dashboard	3
2.4	Technical analysis charts	3
2.5	Problems and proposed solutions	3
2.6	Configuring ESP8266	3
3	Conclusion	3
4	Annexes	4
4.1	ESP8266 code	4
4.2	JavaScript codes used in Node-Red	14

1 Introduction

1.1 Overview

Foreign exchange market is a global market, where people around the world buy and sell different foreign currencies everyday for various purposes. One purpose of the traders in the market is engaging in day-trading activities. Day trading is performed by normal people who intend to obtain profits and earn out of small price mismatches that are available for a short amount of time. Day traders' activity is influenced by information generated from past foreign currency price information. This information is generated by performing various kinds of mathematical transformations on time series price charts. These people find a lot of value in this information generated and they usually buy required market information from their trusted services.

This project mainly focuses on providing foreign exchange market day-traders, a selected set of useful information. This is achieved by using a currency API that provides live currency prices of various currencies around the world. Usually, hourly past currency price data are not available easily, because past currency data is usually provided on a daily basis. Due to this reason, storing the live currency data in the database is useful for making predictions. Users will also find the notification system that warns when the prices are beginning to exceed their expected boundaries, valuable.

ESP8266 is used for controlling the display as well as connecting to the user's mobile phone and Node-Red through WiFi technology. Node-Red to Node-MCU communication happens through the MQTT communication protocol. Node-MCU's ability to enter into sleep mode is also utilized. Node-MCU acts as both Wifi access point and http server. The user may sign-in using a username and a password, when the Node-MCU is functioning as a server. The access point mode is required when it is fetching and submitting data.

1.2 Objectives

- To provide several technical analysis charts such as Simple Moving Average
- to the day-trader so that they can make useful predictions
- To provide the useful information in a user friendly manner through the Node-Red dashboard as well as mobile interface.
- To notify and alarm the day-trader clients when their set upper and lower boundaries are exceeded so that they can take quick actions.
- To provide the user flexibility to choose the interested currencies for observation

1.3 Scope

The scope of the project is to build an innovative IOT application using IOT concepts, tools and standards available. The basic architecture should include a realtime database, an open-source API, Node-Red, Node-MCU and a client mobile phone. Use of a communication protocol such as CoAP or MQTT is also encouraged.

1.4 Architecture

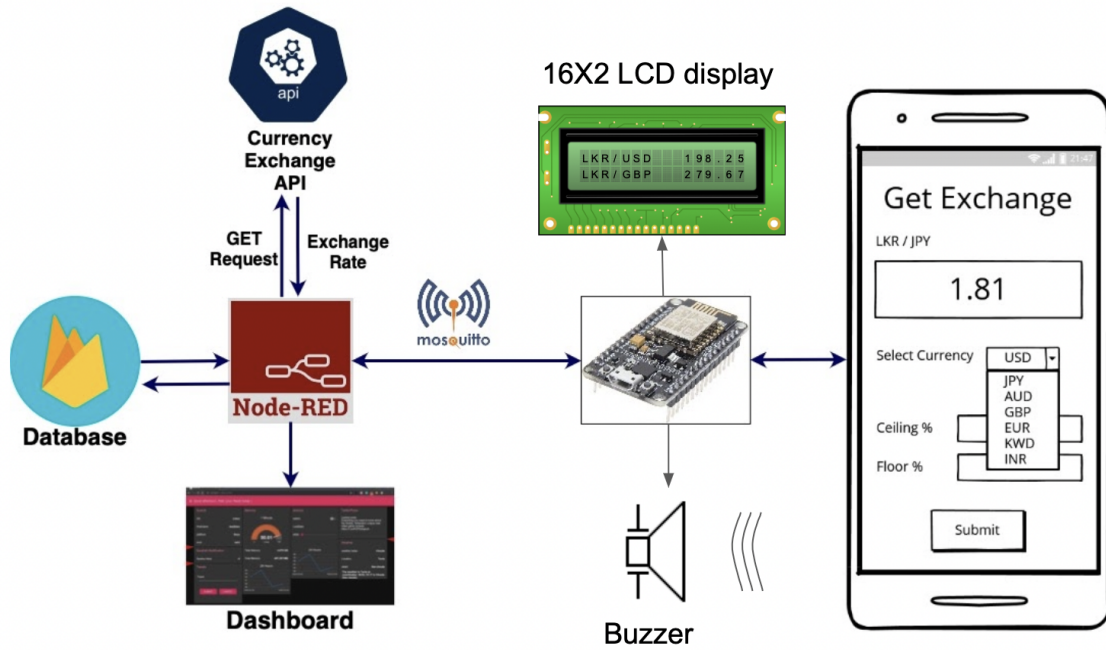


Figure 1: Architecture of Currency Converter Service

2 Methodology

2.1 Exchange API

2.2 Integrating Firebase

2.3 Creating the Node-Red dashboard

2.4 Technical analysis charts

2.5 Problems and proposed solutions

2.6 Configuring ESP8266

3 Conclusion

4 Annexes

4.1 ESP8266 code

```
1 {
2 #if defined(ESP8266)
3 #include <ESP8266WiFi.h>
4 #else
5 #include <WiFi.h>
6 #endif
7
8 #if defined(ESP8266)
9 #include <ESP8266WebServer.h>
10 #else
11 #include <WebServer.h>
12 #endif
13
14 #include <WiFiManager.h>
15 #include <DNSServer.h>
16 #include <PubSubClient.h>
17
18 #include <WiFiUdp.h>
19 #include <NTPClient.h>
20 #include <TimeLib.h>
21 #include <LiquidCrystal.h>
22 LiquidCrystal lcd(D6, D5, D1, D2, D3, D4);
23
24 ESP8266WebServer server(80);
25
26 WiFiClient wifiClient;
27 PubSubClient client(wifiClient);
28 const char* mqttServer = "test.mosquitto.org";
29
30 // Setup NTP for time and date
31 WiFiUDP ntpUDP;
32 NTPClient timeClient(ntpUDP, "asia.pool.ntp.org", 19800, 60000);
33 char Time[ ] = "TIME:00:00:00";
34 char Date[ ] = "DATE:00/00/2000";
35 byte last_second, second_, minute_, hour_, day_, month_;
36 int year_;
37 int counter = 0;
38 unsigned long unix_epoch;
39 char ascii;
40
41 // Node MCU expects from Node-red per 15 seconds
42 //1) All six currencies values and up/down status ( in the order USD, KWD, AUD,
    EUR, JPY, GBP)
43 //2) Selected currency of the user
44 //3) floor or ceiling exceeded (true/false)
45
46
47 // Node Red expects from Node MCU
48 //1) Clients selected Currency type
49 //2) Ceil and Floor of the currency type as percentages (eg-:5,4)
50 String payloadstr;
51 unsigned long timestamp;
52
53 float USD = 198.25; // up - true, down - false
54 float GBP = 198.25; // up - true, down - false
55 float JPY = 198.25; // up - true, down - false
56 float AUD = 198.25; // up - true, down - false
57 float KWD = 198.25; // up - true, down - false
```

```

58 float EUR = 198.25; // up - true, down - false
59
60 bool usd_up = false;
61 bool gbp_up = false;
62 bool jpy_up = false;
63 bool aud_up = false;
64 bool kwd_up = false;
65 bool eur_up = false;
66 char * binary;
67
68 String current_user;
69 String current_currency;
70 bool ceil_crossed = false;
71 bool floor_crossed = false;
72
73 //Variables required for buzzer sound
74 int speakerPin = 13;
75 int len = 15; // the number of notes
76 char notes[] = " C C C C C C C C C "; // a space represents a rest
77 int beats[] = { 1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
78 int tempo = 300;
79
80
81 void setup_wifi() {
82     // Connecting to a WiFi network
83     delay(5000);
84     WiFiManager wifiManager;
85     wifiManager.autoConnect("IoT6B_G05","12345678");
86 }
87
88 void setupMQTT() {
89     client.setServer(mqttServer,1883);
90     client.setCallback(callback);
91 }
92
93 void reconnect() {
94     // Loop until we're reconnected
95     while (!client.connected()) {
96         Serial.print("Attempting MQTT connection...");
97         // Create a random client ID
98         String clientId = "ESP32Client-";
99         clientId += String(random(0xffff), HEX);
100        // Attempt to connect
101        if (client.connect(clientId.c_str())) {
102            Serial.println("connected");
103            // Once connected, publish an announcement...
104            client.publish("IOT_6B/G05/start", "Hello World");
105            // ... and resubscribe
106            client.subscribe("IOT_6B/G05/BuzzerNotification");
107            client.subscribe("IOT_6B/G05/CommonData");
108        } else {
109            Serial.print("failed, rc=");
110            Serial.print(client.state());
111            Serial.println(" try again in 500 milli seconds");
112            // Wait 5 seconds before retrying
113            delay(500);
114        }
115    }
116 }
117
118 void setup() {
119     // put your setup code here, to run once:
120

```

```

121 lcd.begin(16, 2); // Initialize 16x2 LCD Display
122 lcd.clear();
123 lcd.setCursor(0, 0);
124 lcd.print(Time);
125 lcd.setCursor(0, 1);
126 lcd.print(Date);
127 timeClient.begin();
128
129 pinMode(speakerPin, OUTPUT); // Output pin for buzzer
130
131 pinMode(BUILTIN_LED, OUTPUT); // Initialize the BUILTIN_LED pin as an
    output
132 WiFi.mode(WIFI_AP_STA);
133 Serial.begin(115200);
134 setup_wifi();
135 setupMQTT();
136
137 //client.subscribe("IOT_6B/G05/Response");
138 //client.subscribe("IOT_6B/G05/ceil");
139
140 server.on("/", handlerequest);
141 server.onNotFound(handle_NotFound);
142
143 server.begin();
144 Serial.println("HTTP server started");
145
146 }
147
148 void loop() {
149     server.handleClient();
150     if (!client.connected()) {
151         reconnect();
152     }
153     client.loop();
154
155     timeClient.update();
156     unix_epoch = timeClient.getEpochTime(); // Get Unix epoch time from the
        NTP server
157     //Serial.println(unix_epoch);
158     second_ = second(unix_epoch);
159     if (last_second != second_) {
160
161
162         minute_ = minute(unix_epoch);
163         hour_ = hour(unix_epoch);
164         day_ = day(unix_epoch);
165         month_ = month(unix_epoch);
166         year_ = year(unix_epoch);
167
168         Time[12] = second_ % 10 + 48;
169         Time[11] = second_ / 10 + 48;
170         Time[9] = minute_ % 10 + 48;
171         Time[8] = minute_ / 10 + 48;
172         Time[6] = hour_ % 10 + 48;
173         Time[5] = hour_ / 10 + 48;
174
175         Date[5] = day_ / 10 + 48;
176         Date[6] = day_ % 10 + 48;
177         Date[8] = month_ / 10 + 48;
178         Date[9] = month_ % 10 + 48;
179         Date[13] = (year_ / 10) % 10 + 48;
180         Date[14] = year_ % 10 % 10 + 48;
181

```

```

182 //Serial.println(Time);
183 //Serial.println(Date);
184
185 lcd.setCursor(0, 0);
186 lcd.print(Time);
187 lcd.setCursor(0, 1);
188 lcd.print(Date);
189 last_second = second_;
190
191 }
192 delay(500);
193
194 if (counter == 8){
195     counter = 0;
196     lcd.setCursor(0, 0);
197     lcd.print("LKR/USD "+ String(USD));
198     lcd.setCursor(0, 1);
199     lcd.print("LKR/GBP "+ String(GBP));
200
201     if (usd_up){
202         ascii = 0x5e;
203         lcd.setCursor(15 , 0);
204         lcd.print(ascii);
205     } else {
206         ascii = 0x76;
207         lcd.setCursor(15 , 0);
208         lcd.print(ascii);
209     }
210
211     if (gbp_up){
212         ascii = 0x5e;
213         lcd.setCursor(15 , 1);
214         lcd.print(ascii);
215     } else {
216         ascii = 0x76;
217         lcd.setCursor(15 , 1);
218         lcd.print(ascii);
219     }
220     delay(2000);
221
222     server.handleClient();
223     client.loop();
224
225
226     lcd.clear();
227     lcd.setCursor(0, 0);
228     lcd.print("LKR/JPY "+ String(JPY));
229     lcd.setCursor(0, 1);
230     lcd.print("LKR/AUD "+ String(AUD));
231
232     if (jpy_up){
233         ascii = 0x5e;
234         lcd.setCursor(15 , 0);
235         lcd.print(ascii);
236     } else {
237         ascii = 0x76;
238         lcd.setCursor(15 , 0);
239         lcd.print(ascii);
240     }
241
242     if (aud_up){
243         ascii = 0x5e;
244         lcd.setCursor(15 , 1);

```



```

245     lcd.print(ascii);
246 } else {
247     ascii = 0x76;
248     lcd.setCursor(15 , 1);
249     lcd.print(ascii);
250 }
251 delay(2000);
252
253 server.handleClient();
254 client.loop();
255 lcd.clear();
256 lcd.setCursor(0, 0);
257 lcd.print("LKR/KWD " + String(KWD));
258 lcd.setCursor(0, 1);
259 lcd.print("LKR/EUR " + String(EUR));
260
261 if (kwd_up){
262     ascii = 0x5e;
263     lcd.setCursor(15 , 0);
264     lcd.print(ascii);
265 } else {
266     ascii = 0x76;
267     lcd.setCursor(15 , 0);
268     lcd.print(ascii);
269 }
270
271 if (eur_up){
272     ascii = 0x5e;
273     lcd.setCursor(15 , 1);
274     lcd.print(ascii);
275 } else {
276     ascii = 0x76;
277     lcd.setCursor(15 , 1);
278     lcd.print(ascii);
279 }
280
281 delay(2000);
282 server.handleClient();
283 client.loop();
284 lcd.clear();
285 } else {
286     counter += 1;
287 }
288
289 }
290
291 void handlerequest(){
292 // if (server.hasArg("plain")== false){ //Check if body received
293 //     server.send(200, "text/plain", "Body not received");
294 //     return;
295 // }
296 String UserNeeds;
297 current_currency = server.arg("currency");
298 String Ceil = server.arg("ceil");
299 String Floor = server.arg("floor");
300 unsigned long timenow = unix_epoch - 19800;
301
302 UserNeeds  = timenow + "$" + current_currency + "$" + Ceil + "$" + Floor;
303
304
305
306 int currency_len = current_currency.length() ;
307 int ceil_len = Ceil.length();

```

```

308     int floor_len = Floor.length();
309
310     int UserNeeds_len = currency_len + ceil_len + floor_len + 3;
311
312     char UserNeeds_array[UserNeeds_len];
313     UserNeeds.toCharArray(UserNeeds_array, UserNeeds_len);
314     client.publish("IOT_6B/G05/UserNeeds", UserNeeds_array );
315
316
317     server.send(200, "text/html", SendHTML(current_currency));
318 }
319
320 void handle_NotFound(){
321     server.send(404, "text/plain", "Not found");
322 }
323
324 String SendHTML(String Currency){
325     String ptr = "<!DOCTYPE html> <html lang=\"en\">\n";
326     ptr+= "<head>\n";
327     ptr+= "<link rel=\"stylesheet\" href=\"https://cdnjs.cloudflare.com/ajax/libs\n";
328           /font-awesome/4.7.0/css/font-awesome.min.css\">\n";
329     ptr+= "<meta charset=\"UTF-8\">\n";
330     ptr+= "<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">\n";
331     ptr+= "<meta name=\"viewport\" content=\"width=device-width, initial-scale\n";
332           =1.0\">\n";
333     ptr+= "<title>GROUP 5</title>\n";
334     ptr+= "</head>\n";
335     ptr+= "<body style=\"text-align:center;display:grid;place-content: center;\n";
336           background-color: rgb(23, 196, 196);\">\n";
337     ptr+= "<h1 style=\"font-size: 200px;\" >Get Exchange</h1>\n";
338
339     if (Currency == "USD"){
340         ptr+="<h2>LKR/USD</h2>\n";
341     }
342     else if (Currency == "JPY"){
343         ptr+="<h2>LKR/JPY</h2>\n";
344     }
345     else if (Currency == "GBP"){
346         ptr+="<h2>LKR/GBP</h2>\n";
347     }
348     else if (Currency == "EUR"){
349         ptr+="<h2>LKR/EUR</h2>\n";
350     }
351     else if (Currency == "KWD"){
352         ptr+="<h2>LKR/KWD</h2>\n";
353     }
354     else if (Currency == "INR"){
355         ptr+="<h2>LKR/INR</h2>\n";
356     }
357     else{
358         ptr+="<h2>LKR/NON</h2>\n";
359     }
360
361     ptr+= "<h1 style=\" color :rgb(62, 128, 0)\">1.81 <i class=\"fa fa-arrow-up\n";
362           \"></i></h1>\n";
363     ptr+= "<h1 style=\" color :red\">1.81 <i class=\"fa fa-arrow-down\"></i></h1\n";
364           >\n";
365     ptr+= "<form name=\"dropdown\" method=\"get\" style=\" font-size: xx-large;\">\n";
366           >\n";
367     ptr+= "<label for=\"currency_label\">Select Currency :</label><br>\n";
368     ptr+= "<select name=\"currency\" id=\"currency\">\n";
369     ptr+= "<option value=\"USD\">USD</option>\n";
370     ptr+= "<option value=\"JPY\">JPY</option>\n";

```

```

365 ptr+= "<option value=\"GBP\">GBP</option>\n";
366 ptr+= "<option value=\"EUR\">EUR</option>\n";
367 ptr+= "<option value=\"KWD\">KWD</option>\n";
368 ptr+= "<option value=\"INR\">INR</option>\n";
369 ptr+= "</select>\n";
370 ptr+= "<br><br>\n";
371 ptr+= "<label for=\"ceil\">Ceil% :</label><br>\n";
372 ptr+= "<input type=\"number\" id=\"ceil\" name=\"ceil\" value=5><br><br>\n";
373 ptr+= "<label for=\"floor\">Floor% :</label><br>\n";
374 ptr+= "<input type=\"number\" id=\"floor\" name=\"floor\" value=5><br><br>\n"
    ;
375 ptr+= "<input type=\"submit\" value=\"Submit\">\n";
376 ptr+= "</form>\n";
377 ptr+= "</body>\n";
378 ptr+= "</html>\n";
379 return ptr;
380 }
381
382 void callback(char* topic, byte* payload, unsigned int length) {
383
384     if (String(topic) == "IOT_6B/G05/BuzzerNotification") {
385         process_notification(payload, length, 50, 5);
386     }
387
388     if (String(topic) == "IOT_6B/G05/CommonData") {
389         process_data(payload, length, 70, 8);
390     }
391
392     if (ceil_crossed || floor_crossed) {
393         buzzerinit();
394         ceil_crossed = false;
395         floor_crossed = false;
396     }
397 }
398
399 void process_notification(byte* payload, unsigned int length, int charlen, int
    numitem) {
400
401     int digit;
402     payloadstr = "";
403     Serial.println();
404     for (int i = 0; i < length; i++) {
405         payloadstr += (char)payload[i];
406     }
407
408     char payloadstr_array[charlen];
409     payloadstr.toCharArray(payloadstr_array, charlen);
410
411     char * token = strtok(payloadstr_array, "$");
412
413     for (int i = 1; i < numitem+1; i++) {
414         switch (i) {
415             case 1:
416                 timestamp = atol(token);
417                 Serial.print(timestamp);
418                 Serial.println();
419                 break;
420             case 2:
421                 if (timestamp > unix_epoch - 19820) {
422                     current_user = String(token);
423                     Serial.print(current_user);
424                     Serial.println();
425                 }

```

```

426         break;
427     case 3:
428         if (timestamp > unix_epoch - 19820) {
429             current_currency = String(token);
430             Serial.print(current_currency);
431             Serial.println();
432         }
433         break;
434     case 4:
435         if (timestamp > unix_epoch - 19820) {
436             digit = String(token).toInt();
437             if (digit == 1) {
438                 ceil_crossed = true;
439             } else {
440                 ceil_crossed = false;
441             }
442             Serial.print(ceil_crossed);
443             Serial.println();
444         }
445         break;
446     case 5:
447         if (timestamp > unix_epoch - 19820) {
448             digit = String(token).toInt();
449             if (digit == 1) {
450                 floor_crossed = true;
451             } else {
452                 floor_crossed = false;
453             }
454             Serial.print(ceil_crossed);
455             Serial.println();
456         }
457         break;
458     }
459     token = strtok(NULL, "$");
460 }
461 }
462
463 void process_data(byte* payload, unsigned int length, int charlen, int numitem)
464 {
465     payloadstr = "";
466     Serial.println();
467     for (int i = 0; i < length; i++) {
468         payloadstr += (char)payload[i];
469     }
470
471     char payloadstr_array[charlen];
472     payloadstr.toCharArray(payloadstr_array, charlen);
473
474     char * token = strtok(payloadstr_array, "$");
475
476     for (int i = 1; i < numitem+1; i++) {
477         switch (i) {
478             case 1:
479                 timestamp = atol(token);
480                 Serial.print(timestamp);
481                 Serial.println();
482                 break;
483             case 2:
484
485                 USD = String(token).toFloat();
486                 Serial.print(USD);
487                 Serial.println();

```

```

488         break;
489     case 3:
490
491         GBP = String(token).toFloat();
492         Serial.print(GBP);
493         Serial.println();
494         break;
495     case 4:
496
497         JPY = String(token).toFloat();
498         Serial.print(JPY);
499         Serial.println();
500         break;
501     case 5:
502
503         AUD = String(token).toFloat();
504         Serial.print(AUD);
505         Serial.println();
506         break;
507     case 6:
508         //do something when var equals 1
509         KWD = String(token).toFloat();
510         Serial.print(KWD);
511         Serial.println();
512         break;
513     case 7:
514
515         EUR = String(token).toFloat();
516         Serial.print(EUR);
517         Serial.println();
518         break;
519     case 8:
520         set_updown(token);
521         break;
522     }
523     token = strtok(NULL, "$");
524 }
525
526 }
527
528 void set_updown(char * binary) {
529     int digit;
530     for (int i = 1; i < 7; i++) {
531         switch (i) {
532             case 1:
533                 digit = String((char)binary[i-1]).toInt();
534                 if (digit == 1) {
535                     usd_up = true;
536                 } else {
537                     usd_up = false;
538                 }
539                 Serial.print(usd_up);
540                 Serial.println();
541                 break;
542
543             case 2:
544                 digit = String((char)binary[i-1]).toInt();
545                 if (digit == 1) {
546                     gbp_up = true;
547                 } else {
548                     gbp_up = false;
549                 }
550                 Serial.print(gbp_up);

```

```

551     Serial.println();
552     break;
553
554     case 3:
555     digit = String((char)binary[i-1]).toInt();
556     if (digit == 1) {
557         jpy_up = true;
558     } else {
559         jpy_up = false;
560     }
561     Serial.print(jpy_up);
562     Serial.println();
563     break;
564
565     case 4:
566     digit = String((char)binary[i-1]).toInt();
567     if (digit == 1) {
568         aud_up = true;
569     } else {
570         aud_up = false;
571     }
572     Serial.print(aud_up);
573     Serial.println();
574     break;
575
576     case 5:
577     digit = String((char)binary[i-1]).toInt();
578     if (digit == 1) {
579         kwd_up = true;
580     } else {
581         kwd_up = false;
582     }
583     Serial.print(kwd_up);
584     Serial.println();
585     break;
586
587     case 6:
588     digit = String((char)binary[i-1]).toInt();
589     if (digit == 1) {
590         eur_up = true;
591     } else {
592         eur_up = false;
593     }
594     Serial.print(eur_up);
595     Serial.println();
596     break;
597 }
598 }
599 }
600
601 void playTone(int tone, int duration) {
602     for (long i = 0; i < duration * 1000L; i += tone * 2) {
603         digitalWrite(speakerPin, HIGH);
604         delayMicroseconds(tone);
605         digitalWrite(speakerPin, LOW);
606         delayMicroseconds(tone);
607     }
608 }
609
610 void playNote(char note, int duration) {
611     char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
612     int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
613

```

```

614 // play the tone corresponding to the note name
615 for (int i = 0; i < 8; i++) {
616     if (names[i] == note) {
617         playTone(tones[i], duration);
618     }
619 }
620 }
621
622 void buzzerinit() {
623     for (int i = 0; i < len; i++) {
624         if (notes[i] == ',') {
625             delay(beats[i] * tempo); // rest
626         } else {
627             playNote(notes[i], beats[i] * tempo);
628         }
629
630         // pause between notes
631         delay(tempo / 2);
632     }
633 }
634 }

```

4.2 JavaScript codes used in Node-Red