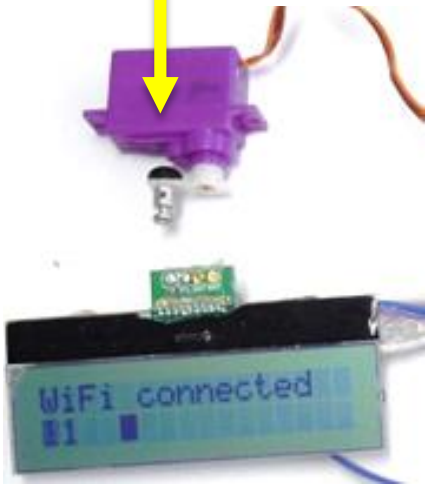
WebSocket を用いたスマホによる LEDのPWM調光およびServo制御

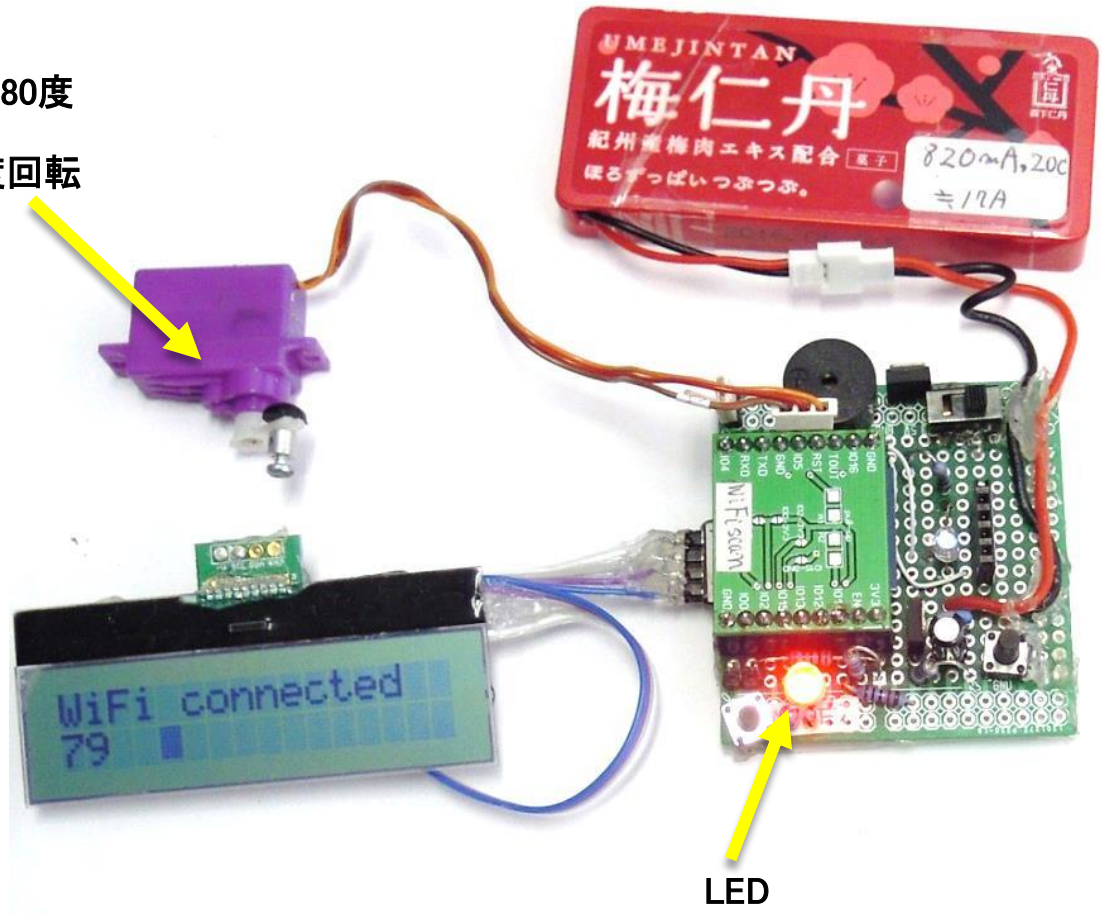
2017.3.16

Servo Motor
回転範囲：約0～180度

約130度回転

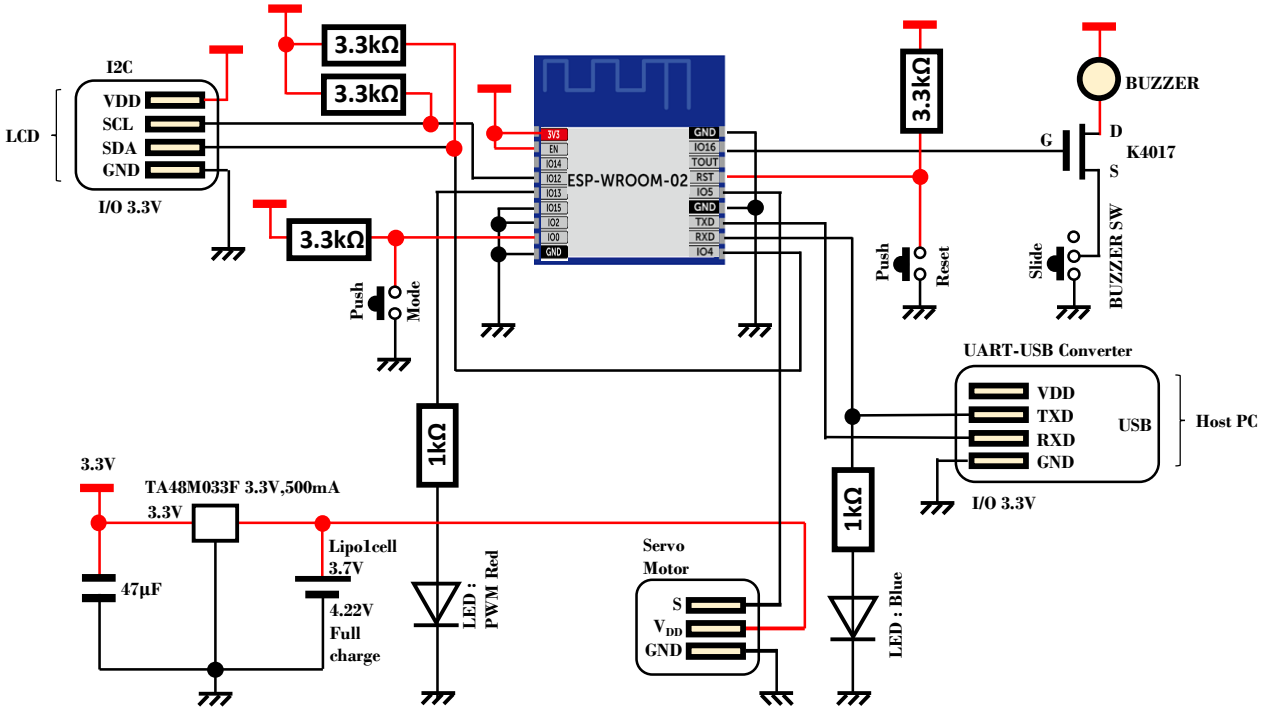
約10度回転





LED
PWM調光

参考にさせて頂いた他人様のページ　https://www.mgo-tec.com/blog-entry-websocket-wroom03.html



ESP8266他人様の元図　http://qiita.com/umi_kappa/items/ac3d37db44a2dcfe71fd

応答速度を改良する必要があります

```
/*  2017.3.15 matkapii
  WebSocket
  LED Modulation from Smapho...
  LCD AQM1602 Akizuki
  WiFi module WROOM-02(ESP8266) aitendo
  web ref.
  https://www.mgo-tec.com/blog-entry-websocket-wroom03.html   */

#include <Servo.h>
#include <Arduino.h>
#include <Hash.h>
#include <Wire.h>
#include "ESP8266WiFi.h"

#define ADDR   0x3e
#define C_Low  0x70
#define C_High 0x56   // 0x5c?

#define AQM1602
#ifdef  AQM1602
#define Cont     5   // Contrast AQM1602 Akizuki
#else
#define Cont    10   // Contrast 1602D1 Strawberry
#endif

//#define Cont     9   // Contrast

/*  ========= Trans Smapho http server =========  */
#define BUZZER 16
//#define WIFI_SSID1 "planexuser"
//#define WIFI_PSK1  "1223334444" // Univ.

//#define WIFI_SSID3 "W01_D4612E0BD29C"    // 0 ==> O (Oh---)
//#define WIFI_PSK3  "an863fdm12tg472"     // Private
//const char* LocalIPaddress = "192.168.100.100";

#define WIFI_SSID3 "IODATA-79d8cd-2G"    //
#define WIFI_PSK3  "9994683780233"      // Home
const char* LocalIPaddress = "192.168.0.5/";

#define DEST_HOST "api.fixer.io"
#define DEST_PORT 80
#define DEST_URL  "/latest?base=USD&symbols=JPY"

const char* ssid     ;
const char* password ;
boolean Ini_html_on = false;
Servo servo;//サーボのインスタンス

//ブラウザからの初回HTTPレスポンス完了したかどうかのフラグ
boolean WS_on = false;//WebSocket設定が済んだかどうかのフラグ
WiFiServer server(80);
char Android_or_iPad; //スマホが Android か iPad かを判定するフラグ
//WiFiClient client;
//LED点灯用ピンアサイン GPIO 13
#define ledPin 13
//通信トラフィックをオーバーフローを起こさせないようにする変数。
//ミリセコンド単位でスマホからのスライダー値送信を間引く
byte s_rate = 10;

uint8_t cmd_cr[] = {0xc0};  // C/R
uint8_t cmd_cl[] = {0x01};  // CLear Display
int ID0 = 0;       // = ID0

void setup() {

  uint8_t cmd_init1[] = {0x38, 0x39, 0x14};
  uint8_t cmd_init2[] = {C_Low | (Cont & 0x0f), C_High | (Cont >> 4 & 3), 0x6c};
  uint8_t cmd_init3[] = {0x38, 0x0d, 0x01};   // 0x0d => 0x0c ?

  //uint8_t mess_setup_done[]  = "Setup done";
  //uint8_t mess_select_APInten[]   = "ON=AP, OFF=Inten";
  uint8_t mess_start_server[]   = "start HTT Server";
  uint8_t mess_WiFi_connected[] = "WiFi connected";
  uint8_t mess_Web_Socket_Test[] = "Web Socket Test";
  uint8_t message[]            = "              ";
  uint8_t mess_point[]         = {0x2e};

  servo.attach(5);
  pinMode(BUZZER, OUTPUT);
  digitalWrite(BUZZER, LOW);

  pinMode(ID0, INPUT);
  Serial.begin(115200);
  delay(10);
  //Wire.begin(4, 14);
  Wire.begin(4, 12);
  delay(40);

  // LCD Initialize
  command(cmd_init1, sizeof(cmd_init1));
  command(cmd_init2, sizeof(cmd_init2));
  delay(300);
  command(cmd_init3, sizeof(cmd_init3));

  delay(3000);
  clear_LCD();
  write(mess_Web_Socket_Test, sizeof(mess_Web_Socket_Test));
  CR_LF();
```

```
  ssid    = WIFI_SSID3;
  password = WIFI_PSK3;

  WiFi.begin(ssid, password);
  Serial.println();
  Serial.println("Web socket '17314");
  while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.print(".");
   write(mess_point, sizeof(mess_point));
  }
  Serial.println("");
  Serial.println("WiFi connected");
  clear_LCD();
  write(mess_WiFi_connected, sizeof(mess_WiFi_connected));
  server.begin();
  Serial.println("Websocket started ! ");
  Serial.println(WiFi.localIP());
  delay(1000);
  //HTTP_server();
}

void loop() {
  CR_LF();
  if (Ini_html_on == false) {

    Ini_HTTP_Response();
  }
  if (Ini_html_on == true && WS_on == false) {
    WS_HTTP_Responce();
  }
  delay(1);//これは重要かも。これがないと動作かも。
}


//*****初回ブラウザからのGET要求によるJavaScript吐き出しHTTPレスポンス*******
void Ini_HTTP_Response()
{
  WiFiClient client = server.available();//クライアント生成は各関数内でしか実行できないので注意
  String req;

  while (client) {
    req = client.readStringUntil('¥n');
    Serial.println(req);
    if (req.indexOf("GET / HTTP") != -1) { //ブラウザからリクエストを受信したらこの文字列を検知する
      Serial.println("-----from Browser FirstTime HTTP Request---------");
      Serial.println(req);
      //ブラウザからのリクエストで空行（¥r¥nが先頭になる）まで読み込む
      while (req.indexOf("¥r") != 0) {
        req = client.readStringUntil('¥n');//¥nまで読み込むが¥n自身は文字列に含まれず、捨てられる
        //ここでブラウザがChromeかSafariかをリクエスト文字列から判定
        if (req.indexOf("Android") != -1) {
          Android_or_iPad = 'A';
  } else if (req.indexOf("iPad") != -1) {
    Android_or_iPad = 'i';
  }
  Serial.println(req);
}
req = "";
delay(10);//10ms待ってレスポンスをブラウザに送信

//メモリ節約のため、Fマクロで文字列を囲う
//普通のHTTPレスポンスヘッダ
client.print(F("HTTP/1.1 200 OK¥r¥n"));
client.print(F("Content-Type:text/html¥r¥n"));
client.print(F("Connection:close¥r¥n¥r¥n"));//１行空行が必要
//ここからブラウザ表示のためのHTML JavaScript吐き出し
client.println(F("<!DOCTYPE html>"));
client.println(F("<html><head>"));
client.println(F("<meta charset='utf-8'>"));
client.println(F("<meta name='viewport' content='initial-scale=1.1'>"));
client.println(F("<title>WebSocket Test</title>"));
client.println(F("<script language='javascript' type='text/javascript'>"));
client.print(F("var wsUri = 'ws://"));
client.print(LocalIPaddress); //ローカルIPアドレス
client.println(F("';"));
client.println(F("var output;"));
client.println(F("var websocket = null;"));
client.println(F("var ms;"));
client.println(F("function init()"));
client.println(F("{ms = new Date();"));
client.println(F("output = document.getElementById('output');"));
client.println(F("testWebSocket();}"));
client.println(F("function testWebSocket()"));
client.println(F("{"));
client.println(F("if(websocket == null){"));
client.println(F("websocket = new WebSocket(wsUri);"));//WebSocketオブジェクト生成
client.println(F("websocket.onopen = function(evt) { onOpen(evt) };"));
client.println(F("websocket.onclose = function(evt) { onClose(evt) };"));
client.println(F("websocket.onmessage = function(evt) { onMessage(evt) };"));
client.println(F("websocket.onerror = function(evt) { onError(evt) };}"));
client.println(F("}"));
client.println(F("function onOpen(evt)"));
client.println(F("{writeToScreen('CONNECTED'); doSend('WebSocket rocks');}"));
client.println(F("function onClose(evt)"));
client.println(F("{"));
client.println(F("writeToScreen('WS.Close.DisConnected');"));
client.println(F("websocket.close();"));
client.println(F("}"));
client.println(F("function onMessage(evt)"));
client.println(F("{var ms1 = document.getElementById('WROOM_DATA');"));
client.println(F("ms1.innerHTML = evt.data;}"));
client.println(F("function onError(evt)"));
```

```
  client.println(F("{writeToScreen(¥"<span style='color: red;'>ERROR:</span> ¥" + evt.data);}"));
  client.println(F("function doSend(data)"));
  client.println(F("{var mms = new Date();"));
  client.println(F("if(mms-ms>"));
  client.print(s_rate);
  client.println(F("){websocket.send(data);"));
  client.println(F("ms = new Date();"));
  client.println(F("}}"));
  client.println(F("function WS_close()"));
  client.println(F("{websocket.close();}"));
  client.println(F("function writeToScreen(message)"));
  client.println(F("{var msg = document.getElementById('msg');"));
  client.println(F("msg.innerHTML = message;}"));
  client.println(F("window.onload = function(){"));
  client.println(F("setTimeout('init()', 3000);}"));
  client.println(F("</script></head>"));
  client.println(F("<body>"));
  client.println(F("<h2 style='color:#5555FF'><center>ESP-WROOM-02(ESP8266)<br>"));
  client.println(F("WebSocket Test</center></h2>"));
  client.println(F("from WROOM DATA = "));
  client.println(F("<font size=4>"));
  client.println(F("<span id='WROOM_DATA' style='font-size:45px; color:#FF0000;'></span>"));//改行しない場合は<span>を使う
  client.println(F("<br>JS-innerHTML="));
  client.println(F("<input type='number' name='v_box' id='v_box' style='width:30px'>"));
  client.println(F("<br><br><center>LED dimming "));
  client.println(F("<input type='range' name='slider' ontouchmove=¥"doSend(this.value); document.getElementById('v_box').value=this.value;¥">"));
  client.println(F("</center><br><br>"));
  client.println(F("<div id='msg' style='font-size:25px; color:#FF0000;'></div><br>"));
  client.println(F("<input type='button' id='WS_close' value='WS.CLOSE' style='width:150px; height:40px; font-size:17px;' onclick='WS_close()'>"));
  client.println(F("<br>"));
  client.println(F("</body></html>¥r¥n"));

  delay(1);//これが重要！これが無いと切断できないかもしれない。
  client.stop();//一旦ブラウザとコネクション切断する。
  delay(1);
  Serial.println("¥nGET HTTP client stop----------");
  req = "";

  //スマホがiPadならばループを抜け出す
  if (Android_or_iPad == 'i') {
   Ini_html_on = true; //初回HTTPレスポンス終わったらtrueにする。
   break;
  }

 } else if (req.indexOf("favicon") != -1) {
  //ChromeはGetリクエストの直ぐ後のfaviconを投げかけてくるところの対処
  Serial.println();
  Serial.println("******GET favicon Request**********");
  Serial.print(req);
    while (client.available()) {
     //ブラウザからデータが送られている間読み込む
     Serial.write(client.read());
    }
    delay(1);
    client.stop();  //GET/faviconでも一旦ブラウザとコネクション切断する必要あり。
    delay(1);
    Serial.println();
    Serial.println("Client Stop----------");
    Ini_html_on = true; //HTTPレスポンス終わったらtrueにする。
    break;
   }
  }
}

//***********HTTPレスポンスとデータ送受信関数*************************
void WS_HTTP_Responce()
{
 WiFiClient client = server.available();//クライアント生成は各関数内でしか実行できないので注意
 String req;
 String hash_req_key;

 while (client) {
  req = client.readStringUntil('¥n');
  Serial.println(req);
  if (req.indexOf("websocket") != -1) { //ブラウザからリクエストを受信したらこの文字列を検知する
   Serial.println("-----from Browser HTTP WebSocket Request---------");
   Serial.println(req);
   //ブラウザからのリクエストで空行（¥r¥nが先頭になる）まで読み込む
   while (req.indexOf("¥r") != 0) {
    req = client.readStringUntil('¥n');//¥nまで読み込むが¥n自身は文字列に含まれず、捨てられる
    Serial.println(req);
    if (req.indexOf("Sec-WebSocket-Key") >= 0) {
     hash_req_key = req.substring(req.indexOf(':') + 2, req.indexOf('¥r'));
     Serial.println();
     Serial.print("hash_req_key =");
     Serial.println(hash_req_key);
    }
   }

   delay(10);
   req = "";

   char h_resp_key[28];
   //ハッシュ値、BASE64エンコード関数
   Hash_Key(hash_req_key, h_resp_key);

   Serial.print("h_resp_key = ");
   Serial.println(h_resp_key);
   String str;
```

```
 //-------ここからHTTPレスポンスのHTMLとJavaScriptコード
 str = "HTTP/1.1 101 Switching Protocols¥r¥n";
 str += "Upgrade: websocket¥r¥n";
 str += "Connection: Upgrade¥r¥n";
 str += "Sec-WebSocket-Accept: ";
 for (byte i = 0; i < 28; i++) {
  str += h_resp_key[i];
 }
 //"Sec-WebSocket-Protocol: chat¥r¥n";これは不要。これを入れるとコネクションできない。
 str += "¥r¥n¥r¥n";//空行は必須

 Serial.println("-----HTTP Respons start-------");
 Serial.println(str);
 client.print(str);
 str = "";

 WS_on = true;//WebSocket 設定終了フラグ

} else if (req.indexOf("favicon") != -1) {
 //Chromeでfaviconを2回連続で投げてきた時の対処
 Serial.println();
 Serial.println("******GET favicon Request**********");
 Serial.print(req);
 while (client.available()) {
  //ブラウザからデータが送られている間読み込む
  Serial.write(client.read());
 }
 delay(1);
 client.stop();  //GET/faviconでも一旦ブラウザとコネクション切断する必要あり。
 delay(1);
 Serial.println();
 Serial.println("Client Stop----------");
 Ini_html_on = true; //HTTPレスポンス終わったらtrueにする。
 break;
}
delay(10);

//ここからWebSocketデータ送受信。
if (WS_on == true) {
 byte b = 0;
 byte data_len;
 byte mask[4];
 byte data_b;
 byte i;
 byte cnt = 0;

 long PingLastTime = millis();
 long PongLastTime = millis();
 long CountTestTime = millis();

 uint8_t LED_Inten[] = "   ";

while (client) {
 //ブラウザがping受信して1秒後までにPongを受信しない場合、コネクション切断する。
 if (millis() - PongLastTime > 4000) break;
 //データ受信が無い時に3sec毎にping送信----------
 if (millis() - PingLastTime > 3000) {
  client.write(B10001001);
  client.write(4);
  client.print("Ping");
  //ブラウザにPing送信すると送信した文字そのものが返って来る。
  Serial.println("Ping Send----------");
  PingLastTime = millis();
 }
 //WROOMのカウンター数値を300ms毎にブラウザに送信
 //あまり秒数が短いとエラーになりクローズするので注意
 if (millis() - CountTestTime > 300) {
  client.write(B10000001);//データ送信ヘッダ
  client.write(1);//送信文字数
  if (cnt > 9) {
   cnt = 0;
  }
  client.print(cnt);
  cnt++;
  CountTestTime = millis();
 }

 if (client.available()) {
  b = client.read();
  if (b == B10000001 || b == B10001010) {
   //B10001010はPongデータ受信
   switch (b) {
    case B10000001:
     //ブラウザからデータ受信している時はPing送信しないようにする。
     PingLastTime = millis();
     PongLastTime = millis();
     break;
    case B10001010:
     PongLastTime = millis();
     Serial.println("Pong Receive**********");
     break;
   }

   b = client.read();
   //マスクビットを削除
   data_len = b - B10000000;

   //マスクキーを読み込む
   for (i = 0; i < 4; i++) {
    mask[i] = client.read();
   }
```

```
          byte m_data[data_len];
        char data_c[data_len];

        Serial.print("Receive Data = ");
        for (i = 0; i < data_len; i++) {
         //マスクされたデータを読み込む
         m_data[i] = client.read();
         //マスクキーとマスクデータをXOR演算すると実テキストデータが得られる
         data_c[i] = mask[i % 4] ^ m_data[i];
         Serial.print(data_c[i]);
        }
        Serial.println();

        //テキストデータを数値に変換
        switch (data_len) {
         case 1:
           data_b = data_c[0] - 0x30; //Char型を数値に変換
           break;
         case 2:
           data_b = ((data_c[0] - 0x30) * 10) + (data_c[1] - 0x30);
           break;
         case 3:
           data_b = ((data_c[0] - 0x30) * 100) + ((data_c[1] - 0x30) * 10) + (data_c[2] - 0x30);
           break;
        }

// ********** mtakapii 2017.3.15 **********
        CR_LF();
        for (int cntr = 0; cntr < 16; cntr++) {
         LCD_space();
        }
        CR_LF();
        for (int cntr = 0; cntr < 2; cntr++) {
         LED_Inten[cntr] = data_c[cntr];
        }
        write(LED_Inten, sizeof(LED_Inten) - 1);
        LED_analog(data_b);    //LED analog 点灯関数
        servo.write(data_b*1.8);      //0~180まで
// ****************************************


       } else if (b == B10001000) {
        delay(1);
        client.write(B10001000);
        delay(1);
        Serial.println("Close Send----------");
        Serial.println(b, BIN);
        break;
       }
      }
     }
```

```
    delay(1);
    client.stop();
    delay(1);
    Serial.println();
    Serial.println("Client.STOP----------");
    WS_on = false;
    Ini_html_on = false;
    break;
   }
  }
 }

//**********ハッシュ値、BASE64エンコード関数**********
void Hash_Key(String h_req_key, char* h_resp_key)
{
 char Base64[65] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
                     'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
                     'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
                     'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/',
                     '='
                    };
 byte hash_six[27];
 byte dummy_h1, dummy_h2;
 byte bb;
 byte i, j;
 i = 0;
 j = 0;

 String GUID_str = "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";
 String merge_str;

 merge_str = h_req_key + GUID_str;
 Serial.println();
 Serial.print("merge_str =");
 Serial.println(merge_str);
 Serial.print("SHA1:");
 Serial.println(sha1(merge_str));

 byte hash[20];
 sha1(merge_str, &hash[0]);

 Serial.print("SHA1:");
 for (uint16_t i = 0; i < 20; i++) {
  Serial.printf("%02x", hash[i]);
  Serial.print("-");
 }
 Serial.println();
 Serial.print("SHA1:");
 for (uint16_t i = 0; i < 20; i++) {
  Serial.print(hash[i], BIN);
  Serial.print("-");
```

```
  }
  Serial.println();

  for ( i = 0; i < 20; i++) {
   hash_six[j] = hash[i] >> 2;
   hash_six[j + 1] = hash[i + 1] >> 4;
   bitWrite(hash_six[j + 1], 4, bitRead(hash[i], 0));
   bitWrite(hash_six[j + 1], 5, bitRead(hash[i], 1));

   if (j + 2 < 26) {
    hash_six[j + 2] = hash[i + 2] >> 6;
    bitWrite(hash_six[j + 2], 2, bitRead(hash[i + 1], 0));
    bitWrite(hash_six[j + 2], 3, bitRead(hash[i + 1], 1));
    bitWrite(hash_six[j + 2], 4, bitRead(hash[i + 1], 2));
    bitWrite(hash_six[j + 2], 5, bitRead(hash[i + 1], 3));
   } else if (j + 2 == 26) {
    dummy_h1 = 0;
    dummy_h2 = 0;
    dummy_h2 = hash[i + 1] << 4;
    dummy_h2 = dummy_h2 >> 2;
    hash_six[j + 2] = dummy_h1 | dummy_h2;
   }

   if ( j + 3 < 27 ) {
    hash_six[j + 3] = hash[i + 2];
    bitWrite(hash_six[j + 3], 6, 0);
    bitWrite(hash_six[j + 3], 7, 0);
   } else if (j + 3 == 27) {
    hash_six[j + 3] = '=';
   }

   h_resp_key[j] = Base64[hash_six[j]];
   h_resp_key[j + 1] = Base64[hash_six[j + 1]];
   h_resp_key[j + 2] = Base64[hash_six[j + 2]];

   if (j + 3 == 27) {
    h_resp_key[j + 3] = Base64[64];
    break;
   } else {
    h_resp_key[j + 3] = Base64[hash_six[j + 3]];
   }

   i = i + 2;
   j = j + 4;
  }

  Serial.print("hash_six = ");
  for (i = 0; i < 28; i++) {
   Serial.print(hash_six[i], BIN);
   Serial.print('_');
  }
  Serial.println();
}
```

```
//************ LED_PWM 出力関数 **************************
void LED_analog(byte data_b)
{
 //analogWriteは 0-255 の値。とりあえずスライダー値の2倍にした。
 //analogWrite(ledPin, data_b * 2.5);
 analogWrite(ledPin, data_b * 10 - 50);    // R=820 ohm
}

//************LCD制御関数**************************
void  LCD_space() {
 uint8_t LED_space[] = {0x20};
 write(LED_space, sizeof(LED_space));
}

void CR_LF() {
 command(cmd_cr, sizeof(cmd_cr));
 delay(10);
}

void clear_LCD() {
 command(cmd_cl, sizeof(cmd_cl));
 delay(10);
}

void command(uint8_t  *cmd, size_t len) {
 size_t i;
 for (i = 0; i < len; i++) {
  Wire.beginTransmission(ADDR);
  delayMicroseconds(30);
  Wire.write(0x00);
  Wire.write(cmd[i]);
  delayMicroseconds(30);
  Wire.endTransmission();
  delayMicroseconds(30);    // 26.3us
 }
}

void write(uint8_t *cmd, size_t len) {
 size_t i;
 for (i = 0; i < len; i++) {
  Wire.beginTransmission(ADDR);
  delayMicroseconds(30);
  Wire.write(0x40);
  delayMicroseconds(30);
  Wire.write(cmd[i]);
  delayMicroseconds(30);
  Wire.endTransmission();
  delayMicroseconds(30);    // 26.3us
 }
 delay(1);
}
```