PROJECT CODE

```
/** LED_BLINK_0 START */
#define DEF_LED_BLINK_GPIO 2
/** LED_BLINK_0 END */
/** DHT_MODULE_0 START */
#define DHT_PIN 4
/** DHT_MODULE_0 END */


/** ULTRASONIC_MODULEA_0 START */
/** ULTRASONIC_MODULEA_0 END */
/** ULTRASONIC_MODULEB_0 START */
/** ULTRASONIC_MODULEB_0 END */
/** SERVO_MODULEA_1 START */
/** SERVO_MODULEA_1 END */
/** SERVO_MODULEB_1 START */
/** SERVO_MODULEB_1 END */
/** ENTRY_CODE_0 START */
/** ENTRY_CODE_0 END */
/** EXIT_CODE_0 START */
/** EXIT_CODE_0 END */
/** FINGER_SENS_0 START */
/** FINGER_SENS_0 END */
/** USER_PINS_1 START */
#define PH_PIN     35
#define SUN_PIN    34
#define PUMP_PIN   18
#define FAN_PIN    19
#define LIGHT_PIN  15
#define LEVEL1_PIN 32
#define LEVEL2_PIN 33
#define LEVEL3_PIN 25
/** USER_PINS_1 END */

/** HEADER_FILES_WRBSERVER_1 START */
#include <WiFi.h>
#include <NetworkClient.h>
#include <WebServer.h>
#include <ESPmDNS.h>

/** HEADER_FILES_WRBSERVER_1 END */
/** DHT_MODULE_1 START */
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
/** DHT_MODULE_1 END */
/** LCD_MODULE_1 START */
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
/** LCD_MODULE_1 END */
/** SERVO_MODULEA_1 START */
/** SERVO_MODULEA_1 END */
struct{
        String message;
        String message_class = "hide";

  /** DATA_STRUCT_1 START */
        String clock;
        String clock_class = "success";

        float temperature;
        String temperature_class = "success";

        float humidity;
        String humidity_class = "success";

        float ph;
        String ph_class = "success";

        String pump;
        String pump_class = "success";

        String level;
        String level_class = "success";

        String light;
        String light_class = "success";

        String air;
        String air_class = "success";

        String pump_control = "OFF";
        String pump_control_class = "success";
        String pump_control_cmd = "";

        String btn_in_out = "IN";
        String btn_in_out_class = "success";
        String btn_in_out_cmd = "";

  /** DATA_STRUCT_1 END */
} data;
struct{
          const int BTN_NONE = -1;
    /** BTN_CONSTS_1  START */
          const int PUMP_CONTROL = 1000;
          const int BTN_IN_OUT = 1001;
```

```
    /** BTN_CONSTS_1 END */
} btnAction;
int userBtnAction = btnAction.BTN_NONE;

/** TEMPLATE_HEADERS_1 START */
/** TEMPLATE_HEADERS_1 END */
/** DHT_MODULE_2 START */
#define DHTTYPE    DHT11
DHT_Unified dht(DHT_PIN, DHTTYPE);

uint32_t delayMSDHT11;
uint32_t lastDHTRead;

struct{
  double temp = 0.0;
  double humidity = 0.0;
  int error = 0;
} DHT11Data;


void setUpDHT11(){
dht.begin();
  Serial.println(F("DHTxx Unified Sensor Example"));
  // Print temperature sensor details.
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println(F("------------------------------------"));
  Serial.println(F("Temperature Sensor"));
  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:  ")); Serial.println(sensor.version);
  Serial.print (F("Unique ID:  ")); Serial.println(sensor.sensor_id);
  Serial.print (F("Max Value:  ")); Serial.print(sensor.max_value); Serial.println(F(" C"));
  Serial.print  (F("Min Value:  ")); Serial.print(sensor.min_value); Serial.println(F(" C"));
  Serial.print  (F("Resolution:  ")); Serial.print(sensor.resolution); Serial.println(F(" C"));
  Serial.println(F("------------------------------------"));
  // Print humidity sensor details.
  dht.humidity().getSensor(&sensor);
  Serial.println(F("Humidity Sensor"));
  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);
  Serial.print  (F("Driver Ver:  ")); Serial.println(sensor.version);
  Serial.print (F("Unique ID:  ")); Serial.println(sensor.sensor_id);
  Serial.print (F("Max Value:  ")); Serial.print(sensor.max_value); Serial.println(F("%"));
  Serial.print  (F("Min Value:  ")); Serial.print(sensor.min_value); Serial.println(F("%"));
  Serial.print  (F("Resolution:  ")); Serial.print(sensor.resolution); Serial.println(F("%"));
  Serial.println(F("------------------------------------"));
  // Set delay between sensor readings based on sensor details.
  delayMSDHT11 = (sensor.min_delay / 1000)+200;
  lastDHTRead = millis();
}
```

```
boolean isDHTReady(){
  if( ((lastDHTRead + delayMSDHT11) < millis()) && ((lastDHTRead + 1000) < millis())){
    return true;
  }
  else{
    return false;
  }
}

void readDHT11(boolean printdebug){
  DHT11Data.error = 0;
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (isnan(event.temperature)) {
  if(printdebug){
      Serial.println(F("Error reading temperature!"));
    }
    DHT11Data.error = 1;
  }
  else {
    if(printdebug){
      Serial.print(F("Temperature: "));
      Serial.print(event.temperature);
      Serial.println(F(" C"));
    }
    DHT11Data.temp =  event.temperature;//*1.8 + 32;
    // Serial.println(sizeof(event.temperature));
  }
  // Get humidity event and print its value.
  dht.humidity().getEvent(&event);
  if (isnan(event.relative_humidity)) {
    if(printdebug){
      Serial.println(F("Error reading humidity!"));
    }
    DHT11Data.error = 1;
  }
  else {
    if(printdebug){
      Serial.print(F("Humidity: "));
      Serial.print(event.relative_humidity);
      Serial.println(F("%"));
    }
    DHT11Data.humidity =  event.relative_humidity;
  }
  lastDHTRead = millis();
}
/** DHT_MODULE_2 END */
```

```
/** LCD_MODULE_2 START */
void setUpLcd(){
  Wire.begin();
  lcd.begin();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("power up");
  lcd.setCursor(0,1);
  lcd.print("booting..");
}

/** LCD_MODULE_2 END */
/** ULTRASONIC_MODULEA_1 START */
/** ULTRASONIC_MODULEA_1 END */
/** ULTRASONIC_MODULEB_1 START */
/** ULTRASONIC_MODULEB_1 END */

/** LEVEL_FUNC_1 START */
/** LEVEL_FUNC_1 END */

/** USER_MODULE_10 START */
/** USER_MODULE_10 END */

/** SERVO_MODULEA_2 START */
/** SERVO_MODULEA_2 END */

/** SERVO_MODULEB_2 START */
/** SERVO_MODULEB_2 END */

/** ENTRY_CODE_1 START */
/** ENTRY_CODE_1 END */

/** INTRRUPT_HANDLERA_1 START */
  /** ENTRY_CODE_2 START */
  /**ENTRY_CODE_2 END */
/** INTRRUPT_HANDLERA_1 END */

/** EXIT_CODE_1 START */
/** EXIT_CODE_1 END */

/** INTRRUPT_HANDLERB_1 START */
  /** ENTRY_CODE_2 START */
  /** ENTRY_CODE_2 END */
/** INTRRUPT_HANDLERB_1 END */

/** ENTRYEXIT_FUNCTION_1 START */
/** ENTRYEXIT_FUNCTION_1 END */

String getMainJson(void){
```

```
/** JSON_RESPONSE_MAIN_1 START */
        return
"{\"message\":\""+data.message+"\",\"message_class\":\""+data.message_class+"\", "
                                "\"clock\":\""+data.clock+"\",
\"clock_class\":\""+data.clock_class+"\",  "
                                "\"temperature\":\""+String(data.temperature)+"\",
\"temperature_class\":\""+data.temperature_class+"\",  "
                                "\"humidity\":\""+String(data.humidity)+"\",
\"humidity_class\":\""+data.humidity_class+"\",  "
                                "\"ph\":\""+String(data.ph)+"\",
\"ph_class\":\""+data.ph_class+"\",  "
                                "\"pump\":\""+data.pump+"\",
\"pump_class\":\""+data.pump_class+"\",  "
                                "\"level\":\""+data.level+"\",
\"level_class\":\""+data.level_class+"\",  "
                                "\"light\":\""+data.light+"\",
\"light_class\":\""+data.light_class+"\",  "
                                "\"air\":\""+data.air+"\", \"air_class\":\""+data.air_class+"\", "
                                "\"pump_control\":\""+data.pump_control+"\",
\"pump_control_class\":\""+data.pump_control_class+"\",  "
                                "\"btn_in_out\":\""+data.btn_in_out+"\",
\"btn_in_out_class\":\""+data.btn_in_out_class+"\"}";
/** JSON_RESPONSE_MAIN_1 END */
}


String getDataJson(void){
/** JSON_RESPONSE_DATA_1 START */
        return "{\"message\":\""+data.message+"\",  "
                                "\"clock\":\""+data.clock+"\", "
                                "\"temperature\":\""+String(data.temperature)+"\", "
                                "\"humidity\":\""+String(data.humidity)+"\", "
                                "\"ph\":\""+String(data.ph)+"\", "
                                "\"pump\":\""+data.pump+"\",  "
                                "\"level\":\""+data.level+"\",  "
                                "\"light\":\""+data.light+"\",  "
                                "\"air\":\""+data.air+"\", "
                                "\"pump_control\":\""+data.pump_control+"\", "
                                "\"btn_in_out\":\""+data.btn_in_out+"\" }";
/** JSON_RESPONSE_DATA_1 END */
}
/** USER_HEADERS_1 START */

#include <Wire.h>
#include <DS3231-RTC.h>

DS3231 myRTC;

byte year;
byte month;
```

```
byte date;
byte dow;
byte hour;
byte minute;
byte second;

bool century = false;
bool h12Flag;
bool pmFlag;

/**************************************************************************
***************************
 * Setup
 *  - Open Serial and Wire connection
 *  - Explain to the user how to use the program

**************************************************************************
***************************/
void setupRTC() {
  // Start the serial port

  // Start the I2C interface
  // Wire.begin();
  // Request the time correction on the Serial
  delay(1000);
  Serial.println("Format YYMMDDwhhmmssx");
  Serial.println("Where YY = Year (ex. 20 for 2020)");
  Serial.println("      MM = Month (ex. 04 for April)");
  Serial.println("      DD = Day of month (ex. 09 for 9th)");
  Serial.println("      w  = Day of week from 1 to 7, 1 = Sunday (ex. 5 for Thursday)");
  Serial.println("      hh = hours in 24h format (ex. 09 for 9AM or 21 for 9PM)");
  Serial.println("      mm = minutes (ex. 02)");
  Serial.println("      ss = seconds (ex. 42)");
  Serial.println("Example for input : 2004095090242x");
  Serial.println("-------------------------------------------------------------------------------");
  Serial.println("Please enter the current time to set on DS3231 ended by 'x':");
}

/** USER_HEADERS_1 END */
/** WIFI_CONFIG_1 START */
const char *ssid = "iota0314-water";
const char *password = "iota0314";
WebServer server(80);
/** WIFI_CONFIG_1 END */
/** HOTSPOT_CODE_1 START */
IPAddress   local_ip(192,168,1,1);
IPAddress   gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
/** HOTSPOT_CODE_1 END */
```

```
/** CLIENT_CODE_1 START */
/** CLIENT_CODE_1 END */
/** LED_BLINK_1 START */

uint32_t def_led_blink_timestamp = 0;

void setUpLedBlink(void){
   pinMode(DEF_LED_BLINK_GPIO,OUTPUT);
   def_led_blink_timestamp = millis();
}

void blink_led(int delay){
   if(def_led_blink_timestamp+delay > millis()){
      return;
   }
   if(digitalRead(DEF_LED_BLINK_GPIO)){
      digitalWrite(DEF_LED_BLINK_GPIO,LOW);
   }
   else{
      digitalWrite(DEF_LED_BLINK_GPIO,HIGH);
   }
   def_led_blink_timestamp = millis();
}
/** LED_BLINK_1 END */

/** SERVER_HANDLERS_1 START */
void forwardTo(String location){
  server.sendHeader("Location", location, true);
  server.send( 302, "text/plain", "");
}

void handle_Home() {
  server.send( 200, "text/html; charset=UTF-8", html_page);
}

void handle_MainJS() {
  server.send( 200, "application/javascript; charset=UTF-8", js_client);
}

void handle_MainJson(){
  server.send( 200, "text/json", getMainJson());
}

void handle_DataJson(){
  server.send( 200, "text/json", getDataJson());
}

void handle_MainCSS(){
  server.send( 200, "text/css; charset=UTF-8", main_css);
}
```

```
void handle_NotFound(){
  server.sendHeader("Location", "/", true);
  server.send( 302, "text/plain", "");
}


void handel_UserAction(){
          for (uint8_t i = 0; i < server.args(); i++) {
    /** BTN_ACTIONS_1 START */
                    if(server.argName(i) == "pump_control"){
                                        userBtnAction = btnAction.PUMP_CONTROL;
                                        data.pump_control_cmd = server.arg(i);
                    }
                    else if(server.argName(i) == "btn_in_out"){
                                        userBtnAction = btnAction.BTN_IN_OUT;
                                        data.btn_in_out_cmd = server.arg(i);
                    }

    /** BTN_ACTIONS_1 END */
           }
          server.send(200, "text/json", getMainJson());
}

/** SERVER_HANDLERS_1 END */


void setUpServer(){
    delay(500);

     /** HOTSPOT_CODE_2 START */
    WiFi.softAP(ssid, password);
    WiFi.softAPConfig(local_ip, gateway, subnet);
    delay(100);
    /** LCD_MODULE_6 START */
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("IP=192.168.1.1");
    lcd.setCursor(0,1);
    lcd.print("PAS=");
    lcd.print(password);
    delay(5000);
    /** LCD_MODULE_6 END */
    /** HOTSPOT_CODE_2 END */
    /** CLIENT_CODE_2 START */
    /** LCD_MODULE_5 START */
    /** LCD_MODULE_5 END */
       /** LCD_MODULE_6 START */
       /** LCD_MODULE_6 END */
```

```
    /** LCD_MODULE_6 START */
    /** LCD_MODULE_6 END */
    /** CLIENT_CODE_2 END */

    if (MDNS.begin("esp32")) {
        Serial.println("MDNS responder started");
    }

    server.on("/", handle_Home);
    server.on("/main.js", handle_MainJS);
    server.on("/act", handel_UserAction);
    server.on("/main.json", handle_MainJson);
    server.on("/data.json", handle_DataJson);
    server.on("/main.css", handle_MainCSS);
    server.onNotFound(handle_NotFound);
    server.begin();
    delay(300);
    Serial.println("HTTP server started");

}

void setUpGPIO(){
/** FINGER_SENS_0 START */
/** FINGER_SENS_0 END */

/** USER_PINS_2 START */
  pinMode(SUN_PIN, INPUT);
  pinMode(LEVEL1_PIN, INPUT);
  pinMode(LEVEL2_PIN, INPUT);
  pinMode(LEVEL3_PIN, INPUT);

  pinMode(PUMP_PIN,  OUTPUT);
  pinMode(LIGHT_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);

/** USER_PINS_2 END */
}

/** FINGER_SENS_0 START */
/** FINGER_SENS_0 END */

/** HEART_RATE_0 START */
/** HEART_RATE_0 END */

/** OXY_GEN_0 START */
/** OXY_GEN_0 END */


  /** USER_GLOBALS_2 START */
  /** USER_GLOBALS_2 END */
```

```
void setup(){
  Serial.begin(115200);
  Serial.println("powerup.");
  setUpGPIO();

  /** ULTRASONIC_MODULEA_2 START */
  /** ULTRASONIC_MODULEA_2 END */

  /** ULTRASONIC_MODULEB_2 START */
  /** ULTRASONIC_MODULEB_2 END */

  /** USER_MODULE_11 START */
  /** USER_MODULE_11 END */

  /** LCD_MODULE_3 START */
  setUpLcd();
  /** LCD_MODULE_3 END */

  /** SERVO_MODULEA_2 START */
    /** USER_SERVO_1 START */
    /** USER_SERVO_1 END */
  /** SERVO_MODULEA_2 END */

  /** ENTRY_CODE_3 START */
  /** ENTRY_CODE_3 END */

  /** EXIT_CODE_3 START */
  /** EXIT_CODE_3 END */

  setUpServer();

  /** USER_GLOBALSINIT_1 START */
  setupRTC();
  /** USER_GLOBALSINIT_1 END */

  /** LED_BLINK_2 START */
  setUpLedBlink();
  /** LED_BLINK_2 END */

  /** USER_INIT_1 START */
  /** USER_INIT_1 END */

}

/** USER_GLOBALS_3 START */
void inputDateFromSerial() {
        // Call this if you notice something coming in on
        // the serial port. The stuff coming in should be in
        // the order YYMMDDwHHMMSS, with an 'x' at the end.
```

```
boolean isStrComplete = false;
char inputChar;
byte temp1, temp2;
char inputStr[20];

uint8_t currentPos = 0;
while (!isStrComplete) {
        if (Serial.available()) {
                inputChar = Serial.read();
                inputStr[currentPos] = inputChar;
                currentPos += 1;

    // Check if string complete (end with "x")
                        if (inputChar == 'x') {
                                isStrComplete = true;
                        }
                }
        }
        Serial.println(inputStr);

 // Find the end of char "x"
 int posX = -1;
 for(uint8_t i = 0; i < 20; i++) {
  if(inputStr[i] == 'x') {
    posX = i;
    break;
  }
 }

 // Consider 0 character in ASCII
 uint8_t zeroAscii = '0';

        // Read Year first
        temp1 = (byte)inputStr[posX - 13] - zeroAscii;
        temp2 = (byte)inputStr[posX - 12] - zeroAscii;
        year = temp1 * 10 + temp2;

        // now month
        temp1 = (byte)inputStr[posX - 11] - zeroAscii;
        temp2 = (byte)inputStr[posX - 10] - zeroAscii;
        month = temp1 * 10 + temp2;

        // now date
        temp1 = (byte)inputStr[posX - 9] - zeroAscii;
        temp2 = (byte)inputStr[posX - 8] - zeroAscii;
        date = temp1 * 10 + temp2;

        // now Day of Week
        dow = (byte)inputStr[posX - 7] - zeroAscii;
```

```
        // now Hour
        temp1 = (byte)inputStr[posX - 6] - zeroAscii;
        temp2 = (byte)inputStr[posX - 5] - zeroAscii;
        hour = temp1 * 10 + temp2;

        // now Minute
        temp1 = (byte)inputStr[posX - 4] - zeroAscii;
        temp2 = (byte)inputStr[posX - 3] - zeroAscii;
        minute = temp1 * 10 + temp2;

        // now Second
        temp1 = (byte)inputStr[posX - 2] - zeroAscii;
        temp2 = (byte)inputStr[posX - 1] - zeroAscii;
        second = temp1 * 10 + temp2;
}


float calibration_value = 28.34;
int phval = 0;
unsigned long avgval;
int buffer_arr[10],temp;


/** USER_GLOBALS_3 END */

uint32_t data_update_timestamp = 0;
void loop(){
  server.handleClient();

  /** LED_BLINK_3 START */
  blink_led(500);
  /** LED_BLINK_3 END */

  /** USER_LOOPVARS_1 START */
  /** USER_LOOPVARS_1 END */
  if(userBtnAction != btnAction.BTN_NONE){
  /** BUTTON_ACTION_2 START */
                    if(userBtnAction == btnAction.PUMP_CONTROL ){
                            //data.pump_control = data.pump_control_cmd;
 /** USER_CONDITION_1 START */
    data.pump_control = data.pump_control_cmd;
    if(data.pump_control == "ON"){
    data.pump_control_class = "danger";
    }
    else if(data.pump_control == "OFF"){
      data.pump_control_class = "success";
    }
    else if(data.pump_control == "AUTO"){
      data.pump_control_class = "primary";
```

```
   }

/** USER_CONDITION_1 END */
                    }
                    if(userBtnAction == btnAction.BTN_IN_OUT ){
                            //data.btn_in_out = data.btn_in_out_cmd;
/** USER_CONDITION_2 START */
    data.btn_in_out = data.btn_in_out_cmd;
    if(data.btn_in_out == "OUT"){
    digitalWrite(FAN_PIN,LOW);
    data.btn_in_out_class = "danger";
    }
    else{
      data.btn_in_out_class = "success";
      digitalWrite(FAN_PIN,HIGH);
    }
/** USER_CONDITION_2 END */
                    }

 /** BUTTON_ACTION_2 END */
  userBtnAction = btnAction.BTN_NONE;
  Serial.println("Button clicked");
 }


 if(data_update_timestamp+500 < millis()){
   data_update_timestamp = millis();


  /** FINGER_SENS_0 START */
    /** HEART_RATE_0 START */
    /** HEART_RATE_0 END */
    /** OXY_GEN_0 START */
    /** OXY_GEN_0 END */
    /** HEART_RATE_0 START */
    /** HEART_RATE_0 END */
    /** OXY_GEN_0 START */
    /** OXY_GEN_0 END */
  /** FINGER_SENS_0 END */

 /** USER_LOOP_1 START */
 // Serial.print(myRTC.getYear(), DEC);
 // Serial.print("-");
 // Serial.print(myRTC.getMonth(century), DEC);
 // Serial.print("-");
 // Serial.print(myRTC.getDate(), DEC);
 // Serial.print(" ");
 // Serial.print(myRTC.getHour(h12Flag, pmFlag), DEC); //24-hr
 // Serial.print(":");
 // Serial.print(myRTC.getMinute(), DEC);
```

```
  // Serial.print(":");
  // Serial.println(myRTC.getSecond(), DEC);

  data.clock = String( myRTC.getYear() ) +"-"+ String( myRTC.getMonth(century) ) +"-"+
String( myRTC.getDate() ) + "<br>" + String( myRTC.getHour(h12Flag, pmFlag) )
+":"+String( myRTC.getMinute() )+":"+ String(myRTC.getSecond());
  Serial.println(data.clock);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("T:");
  lcd.print(data.temperature);
  lcd.print((char)223);
  lcd.print("C ");
  lcd.print("H:");
  lcd.print(data.humidity);
  lcd.print("%");
  lcd.setCursor(0,1);
  lcd.print(data.clock.substring(data.clock.indexOf(">")+1,  data.clock.length()));



  int sun = digitalRead(SUN_PIN) == 1 ? 0 : 1;

  if(sun){
    data.light = "DAY";
    data.light_class = "warning";
    digitalWrite(LIGHT_PIN, LOW);
  }
  else{
    data.light = "NIGHT";
    data.light_class = "secondary";
    digitalWrite(LIGHT_PIN, HIGH);
  }


  int level = digitalRead(LEVEL1_PIN) == 0 ? 0x01 : 0x00;
  // level |= digitalRead(LEVEL2_PIN) == 0 ? 0x02 : 0x00;
  // level |= digitalRead(LEVEL3_PIN) == 0 ? 0x04 : 0x00;

  if(level){
    data.level = "FULL";
    data.level_class = "success";
  }
  else{
    data.level = "LOW";
    data.level_class = "warning";
  }
  // if(level&0x01){
  //    data.level = "FULL";
  //    data.level_class = "success";
```

```
// }
// else if(level&0x02){
//    data.level = "MEDIUM";
//    data.level_class = "success";
// }
// else if(level&0x04){
//    data.level = "LOW";
//    data.level_class = "warning";
// }
// else{
//   data.level = "EMPTY";
//   data.level_class = "danger";
// }

if(data.pump_control == "ON"){
  data.pump = "ON";
  data.pump_class = "danger";
  digitalWrite(PUMP_PIN,HIGH);
}
else if(data.pump_control == "OFF"){
  data.pump = "OFF";
  data.pump_class = "success";
  digitalWrite(PUMP_PIN,LOW);
}
else if(data.pump_control == "AUTO"){

  if(data.humidity < 40){
     data.pump = "AUTO ON";
     data.pump_class = "danger";
     digitalWrite(PUMP_PIN,HIGH);
  }
  else{
     data.pump = "AUTO OFF";
     data.pump_class = "success";
     digitalWrite(PUMP_PIN,LOW);
  }
}


  for(int i=0;i<10;i++)
  {
    buffer_arr[i]=analogRead(PH_PIN);
    delay(2);
  }
  for(int i=0;i<9;i++)
  {
    for(int j=i+1;j<10;j++)
    {
      if(buffer_arr[i]>buffer_arr[j]){
        temp=buffer_arr[i];
```

```
buffer_arr[i]=buffer_arr[j];
buffer_arr[j]=temp;
        }
      }
    }
    avgval=0;
    for(int i=2;i<8;i++)
    avgval+=buffer_arr[i];
    float volt=(float)avgval*3.3/4095/6;
    float ph_act = -5.70 * volt + calibration_value;
    Serial.print("volt:");
    Serial.println(volt);
    Serial.print("ph_act:");
    Serial.println(ph_act);
    data.ph = ph_act;
  /** USER_LOOP_1 END */

  /** LCD_MODULE_4 START */

    // lcd.clear();
    // lcd.setCursor(0,0);
    // lcd.print("Hello 1");
    // lcd.setCursor(0,1);
    // lcd.print("Hello 2");
  /** LCD_MODULE_4 END */

  }

  /** USER_LOOP_2 START */

  /** USER_LOOP_2 END */

  /** USER_LOOP_3 START */
   if (Serial.available()) {
   inputDateFromSerial();

    myRTC.setClockMode(false);  // set to 24h

    myRTC.setYear(year);
    myRTC.setMonth(month);
    myRTC.setDate(date);
    myRTC.setDoW(dow);
    myRTC.setHour(hour);
    myRTC.setMinute(minute);
    myRTC.setSecond(second);

    // Give time at next five seconds
    // Notify that we are ready for the next input
```

```
 /** USER_LOOP_3 END */


  /** DHT_MODULE_3 START */
  zif(isDHTReady()){
  readDHT11(false /*debug*/);
  if(DHT11Data.error == 1){
     Serial.println("DHT READ ERROR!");
   }else{
       Serial.print("Temp:"); Serial.println(DHT11Data.temp);
       /** TEMPRATURE_VAR_0 START */
       data.temperature  = DHT11Data.temp;
       /** TEMPRATURE_VAR_0 END */
       Serial.print("Hum:");
       Serial.println(DHT11Data.humidity);

       /** HUMIDITY_VAR_1 START */
       data.humidity  = DHT11Data.humidity;
       /** HUMIDITY_VAR_1 END */
   }
  }
  /** DHT_MODULE_3 END */

  delay(10);
}
```