โครงงานเรื่องหุ่นยนต์ตรวจวัดคุณภาพน้ำชุมชน

จัดทำโดย 1. นายนันทิวัฒน์ จิตต์พิริยะการ
2. นางสาวชุติกาญจน์ ธรารักษ์
3. นายธราเทพ พิมพิศาล
4. นายศุภวิชญ์ ภูโอบ
5. นายวชิยากรณ์ ไสวอมร

อาจารย์ที่ปรึกษาโครงงาน 1. นายชุมพล ชารีแสน
2. นางสาวเนตรชนก ศรีโท

สถานศึกษา โรงเรียนกาฬสินธุ์พิทยาสรรพ์ อำเภอเมือง จังหวัดกาฬสินธุ์
เบอร์โทรศัพท์ 043-811278 , 098-584-7880
อีเมล https://kalasinpit.ac.th/

บทคัดย่อ

1. ที่มาและความสำคัญ

น้ำเป็นทรัพยากรธรรมชาติที่มีความสำคัญต่อการดำรงชีวิตและการพัฒนาเศรษฐกิจ แหล่งน้ำตามธรรมชาติ ได้แก่ น้ำในบรรยากาศ (ฝน) น้ำผิวดินและน้ำบาดาล นับเป็นผลิตผลจากธรรมชาติที่มนุษย์ไม่สามารถผลิตขึ้นหรือลดปริมาณที่มีอยู่ในธรรมชาติได้เองตามต้องการ (ปราโมทย์ ไม้กลัด. 2557) จังหวัดกาฬสินธุ์มีแหล่งน้ำตามธรรมชาติจำนวนมาก แต่ก็ยังประสบกับปัญหาเกี่ยวกับน้ำ อาทิ การขาดแคลนน้ำและภัยแล้ง ปัญหาอุทกภัย และปัญหาน้ำเสียในแหล่งน้ำธรรมชาติจากผู้ใช้น้ำ การถูกละเลยและถูกบุกรุกไปใช้ประโยชน์ส่วนตน การระบายน้ำเสียลงสู่แม่น้ำลำคลองโดยมิได้ทำการบำบัด (กรมชลประทาน. 2561)

คณะผู้จัดทำได้เล็งเห็นปัญหาและความสำคัญดังกล่าว จึงมีแนวคิดที่จะนำความรู้ด้านวิทยาศาสตร์และเทคโนโลยี และระบบอัตโนมัติ มาประยุกต์สร้างเป็นหุ่นยนต์ตรวจวัดคุณภาพน้ำชุมชน ควบคุมการทำงานด้วยรีโมทคอนโทรลโดยอาศัยคลื่นวิทยุ เพื่อสำรวจแหล่งน้ำระยะไกลหรือในพื้นที่ที่เข้าถึงได้ยาก โดยตรวจวัดค่าต่าง ๆ ของน้ำ ได้แก่ วัดปริมาณของแข็ง (TDS) ค่าความเป็นกรด (pH) และปริมาณออกซิเจนที่ละลายน้ำ (DO) ซึ่งค่าเหล่านี้จะช่วยบอกถึงคุณภาพของน้ำเบื้องต้น โดยเก็บข้อมูลผ่าน google sheet แล้วนำข้อมูลไปวิเคราะห์ผ่าน app.geckoboard.com ที่สะดวกมากขึ้น ซึ่งเหมาะกับการเฝ้าระวังการสะสมมลพิษที่จะก่อให้เกิดน้ำเน่าเสียได้

2. ลักษณะของซอฟต์แวร์

หุ่นยนต์ตรวจวัดคุณภาพน้ำชุมชน เป็นชุดอุปกรณ์ประกอบด้วยชุด Hardware ที่พัฒนาด้วยบอร์ด Arduino Uno R3 Wifi ESP8266 และ Software ที่พัฒนาด้วยภาษา C++ ที่สั่งการให้เซ็นเซอร์และอุปกรณ์ต่าง ๆ ภายในระบบทำงานอย่างถูกต้องแม่นยำ

3. ทฤษฎี หลักการ เทคนิค และเทคโนโลยีที่ใช้

คณะผู้จัดทำได้พัฒนาโปรแกรมตรวจวัดคุณภาพน้ำด้วยภาษา C++ เพื่อใช้ในระบบการแจ้งเตือนคุณภาพน้ำจากแหล่งน้ำชุมชน ด้วยระบบ Arduino Uno R3 Wifi ESP8266 โดยเขียนโค้ดการทำงานสำหรับการตรวจวัดคุณภาพน้ำ หากคุณภาพน้ำต่ำกว่าเกณฑ์มาตรฐานระบบพร้อมทั้งเก็บข้อมูลผ่าน google sheet **แล้วนำข้อมูลไปวิเคราะห์ผ่าน app.geckoboard.com ที่สะดวกมากขึ้น**

ปริมาณออกซิเจนที่ละลายน้ำ (DO) (กรมควบคุมมลพิษ : 2555.)

1) น้ำในธรรมชาติทั่วไปปกติจะมีค่าดีโอ Dissolved Oxygen ประมาณ 5-7 มิลลิกรัมต่อลิตร (mg/L)

2) มาตรฐานน้ำที่มีคุณภาพดี จะมีค่า DO ประมาณ 5 – 8 mg/L

3) น้ำเสีย จะมีค่า DO ต่ำกว่า 3 mg/L

ค่าความเป็นกรด (pH)

1) pH มีค่า 7 หมายความว่ามีความเป็นกลาง (natural pH)

2) pH มีค่าต่ำกว่า 7 แสดงความเป็นกรด (acidic pH)

3) pH มีค่าสูงกว่า 7 แสดงความเป็นเบสหรือด่าง (alkaline pH)

การจำแนกน้ำตามค่า TDS

1) ระหว่าง 50-150 เหมาะสำหรับดื่ม

2) 150-250 ดี

3) 250-300 ปานกลาง

4) 300-500 แย่

5) สูงกว่า 1200 ไม่สามารถยอมรับได้

น้ำดื่มโดยทั่วไปมี TDS ต่ำกว่า 500 ppm น้ำจืด TDS ที่สูงกว่าสามารถดื่มได้ แต่รสชาติอาจไม่เหมาะสม

ซอฟต์แวร์ที่ใช้ในการพัฒนา (Software)

โปรแกรม Arduino IDE

อุปกรณ์ที่ใช้ในการพัฒนา (Hardware)

- อุปกรณ์คอมพิวเตอร์ที่ติดตั้งโปรแกรม Arduino IDE
- ระบบปฏิบัติการ Microsoft Windows
- บอร์ด Arduino Uno IDE

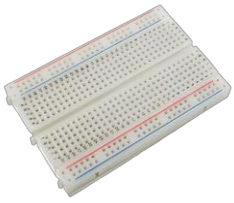
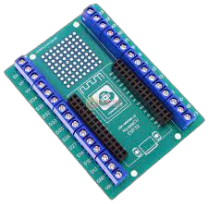
4. กลุ่มผู้ใช้งาน

คณะผู้จัดทำได้ออกแบบและมุ่งเน้นพัฒนาโปรแกรมชุดนี้เพื่อตรวจสอบคุณภาพน้ำจากแหล่งน้ำในชุมชน เพื่อความปลอดภัยของสิ่งมีชีวิตและสิ่งแวดล้อมในชุมชน

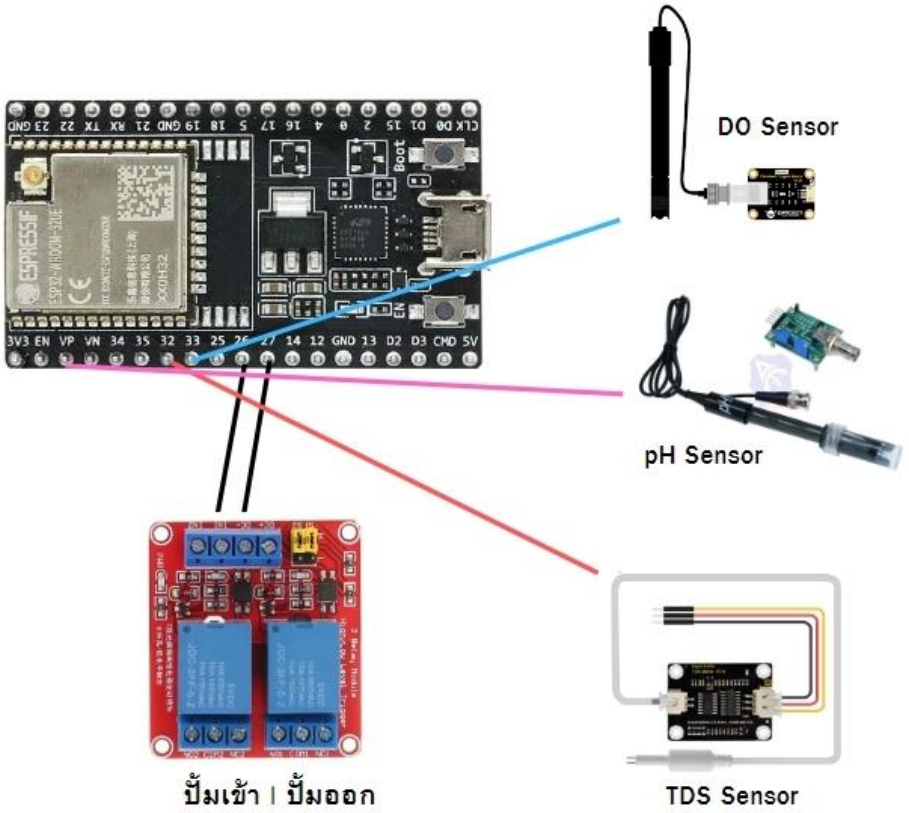
5. ผลที่คาดว่าจะได้รับ

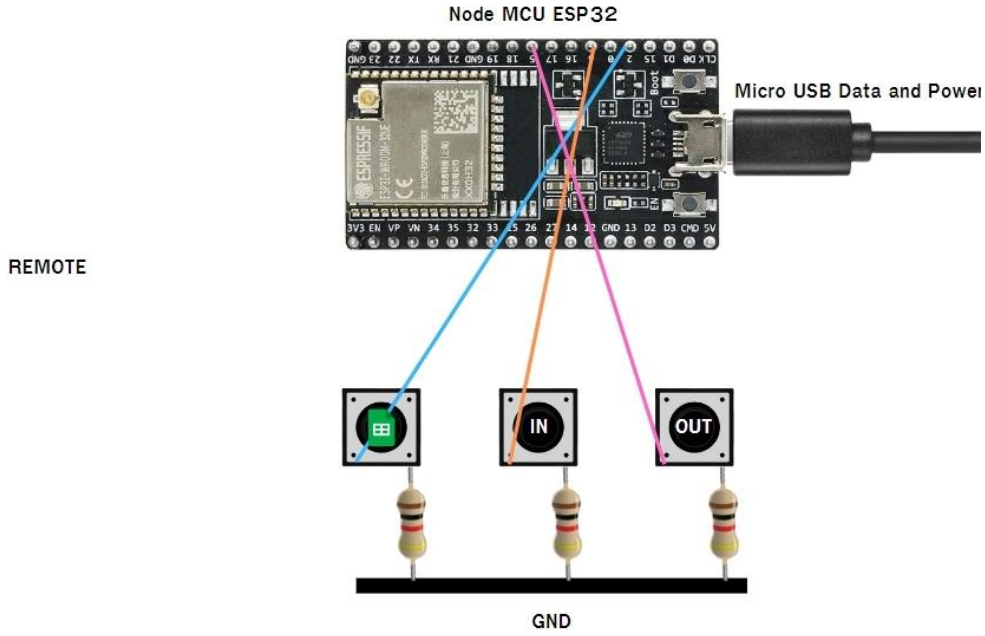
1. ได้หุ่นยนต์ตรวจวัดคุณภาพน้ำ
2. ได้เรียนรู้การประยุกต์ใช้วงจรวิทยุในการควบคุมการเคลื่อนที่ของเรือ
3. ได้นำความรู้ที่ได้จากชั้นเรียนมาสู่การปฏิบัติงานจริง นำไปสู่องค์ความรู้ที่ได้จากการลงมือปฏิบัติ และการรู้จักแก้ไขปัญหาที่ประสบขณะทำการวิจัย

6. วัสดุและอุปกรณ์

	
บอร์ด NodeMCU esp32	สายจัมเปอร์ผู้-ผู้และเมีย-เมีย
	
sensor tds Arduino	sensor ph arduino
	
Breadboard	sensor Do arduino
	
nodemcu shield	เรือบังคับ

7. การต่ออุปกรณ์





8. หลักการทำงาน

Receiver 2nd ESP32

1. โค้ดเริ่มต้นด้วยการเรียกใช้ไลบรารีต่างๆเช่น esp_now, WiFi, WiFiClientSecure เพื่อใช้ในการเชื่อมต่อ Wi-Fi, สื่อสารแบบ ESP-NOW, และการส่งข้อมูลผ่าน HTTPS ไปยัง Google Spreadsheet
2. กำหนดค่าตัวแปรต่างๆ เช่น พินของเซ็นเซอร์ต่างๆ, เวลาการทำงานของปั๊ม, และค่าสำหรับการสองรอบในการประมวลผลสำหรับเซ็นเซอร์ TDS
3. มีฟังก์ชันสำหรับการส่งข้อมูลไปยัง Google Spreadsheet โดยใช้วิธีการส่งคำร้องเควส GET และ HTTPS
4. มีฟังก์ชันสำหรับการอ่านค่าเซ็นเซอร์ต่างๆ เช่น pH, DO (Dissolved Oxygen), TDS (Total Dissolved Solids)
5. มีการเชื่อมต่อ Wi-Fi และการตั้งค่าเพื่อให้ ESP8266 สามารถเชื่อมต่อไปยัง Google Spreadsheet ผ่าน HTTPS
6. ใช้ ESP-NOW เพื่อรับข้อมูลจากอุปกรณ์อื่นและดำเนินการตามคำสั่งที่รับมา เช่น เปิดปิดปั๊ม

โปรแกรมนี้ใช้งานได้ดังนี้:

1.เริ่มต้นโปรแกรมและเชื่อมต่อ Wi-Fi กับเราเตอร์ที่กำหนด

2.อ่านค่าเซ็นเซอร์ต่าง ๆ เช่น pH, DO, TDS

3.รอรับคำสั่งจากอุปกรณ์อื่นผ่าน ESP-NOW และดำเนินการตามคำสั่ง เช่น เปิดปิด.

4.หากได้รับคำสั่งเปิดปิดผ่าน ESP-NOW โปรแกรมจะส่งสัญญาณเปิดหรือปิดไปยังปั๊มตามคำสั่งที่ได้รับ

5.โปรแกรมจะวนซ้ำและอ่านค่าเซ็นเซอร์ต่าง ๆ เพื่อส่งข้อมูลไปยัง Google Spreadsheet ทุก ๆ ช่วงเวลาที่กำหนด

6.หากมีการเปิดหรือปิดปั๊มโดยใช้ ESP-NOW โปรแกรมจะบันทึกสถานะปั๊มล่าสุดลงใน Google Spreadsheet

7.การอ่านค่าเซ็นเซอร์ต่าง ๆ และการส่งข้อมูลไปยัง Google Spreadsheet จะเกิดขึ้นเป็นรอบ ๆ ตามเวลาที่กำหนด และสามารถปรับเวลาการอัปเดตได้ตามต้องการ32

Remote(1st ESP32)

1. ไลบรารี:โค้ดนี้ใช้ไลบรารีสองตัวคือ esp_now.h และ WiFi.h ไลบรารีเหล่านี้มีฟังก์ชันและคำนิยามที่จำเป็นสำหรับการใช้งาน ESP-NOW และฟังก์ชันที่เกี่ยวข้องกับฟังก์ชัน WiFi
2. ที่อยู่ MAC:ตัวแปร broadcastAddress เป็นอาร์เรย์ที่แทนที่ที่อยู่ MAC ของตัวรับ ESP32 โค้ดได้กำหนดค่าเริ่มต้นไว้แต่คุณจะต้องแทนที่ด้วยที่อยู่ MAC ของ ESP32 ตัวรับจริงของคุณ
3. Debounce Button:มีฟังก์ชันที่ชื่อว่า debounce ซึ่งช่วยในการกันการกระพริบของปุ่ม ฟังก์ชันนี้รับหมายเลขขาของปุ่มเป็นอาร์กิวเมนต์และคืนค่าสถานะของปุ่มหลังจากถูก debounce
4. ตัวแปร : ตัวแปร send_rnd_val_1, send_rnd_val_2, และ coolDown ใช้สำหรับเก็บและจัดการข้อมูลที่จะถูกส่ง
5. ฟังก์ชัน Callback: ฟังก์ชัน OnDataSent เป็นฟังก์ชัน Callback ที่ทำงานเมื่อข้อมูลถูกส่งโดยใช้ ESP-NOW ฟังก์ชันนี้พิมพ์สถานะการส่งแพ็คเก็ต
6. ฟังก์ชัน Setup : ฟังก์ชัน setup จัดการการเริ่มต้นการสื่อสารด้วยฟังก์ชันทางบริเวณล่าง

7. ฟังก์ชัน Loop : ฟังก์ชัน loop ตรวจสอบสถานะปุ่มและส่งข้อมูลตามการกดปุ่ม ปุ่ม 1, ปุ่ม 2, และ ปุ่ม 3 เมื่อกดปุ่ม ค่าจะถูกกำหนดให้กับ send_rnd_val_1 และค่าที่สอดคล้องจะถูกตั้งค่าใน send_Data ข้อมูลจะถูกส่งโดยใช้ ESP-NOW ด้วย esp_now_send หลังจากส่งข้อมูลแล้วจะตั้งค่าระยะเวลาการรอเพื่อป้องกันการกดปุ่มต่อเนื่องอย่างรวดเร็ว

8. การจัดการสถานะแป้นพิมพ์ : อาร์เรย์ key ใช้สำหรับติดตามสถานะของแต่ละปุ่ม (กดหรือปล่อย) เพื่อป้องกันการดำเนินการส่งหลายครั้งสำหรับการกดปุ่มเดียวกันเมื่อกดปุ่ม ค่าของแป้นพิมพ์ที่สอดคล้องจะถูกตั้งค่าเป็น 1 เพื่อแสดงสถานะการกด และเมื่อปุ่มถูกปล่อย ค่าของแป้นพิมพ์จะถูกตั้งค่าเป็น 0

การทดสอบระบบ

- กด ปุ่ม1 เพื่อปั๊มน้ำเข้ากล่องเก็บน้ำตัวอย่าง
- กดปุ่ม 2 เพื่อให้เซนเซอร์ตรวจวัดค่าคุณภาพน้ำ
- โปรดรอให้การทำงานแต่ละปุ่มเสร็จก่อน ค่อยกดปุ่มใหม่ ไม่ควรกดพร้อมกัน
- กด 3 ปั๊มน้ำปล่อยน้ำออก

การประมวลผลด้วย app.geckoboard.com

-ขั้นที่ 1 ดึงข้อมูลจาก Google Sheet มาลงใน app.geckoboard.com

-ขั้นที่ 2 นำมาทำเป็นกราฟของค่า pH ,DO , TDS ณ ช่วงเวลาที่ตรวจวัด

-ขั้นที่ 3 ดูกราฟผ่านลิ้ง

:https://share.geckoboard.com/dashboards/5HY5F3TKW2EHS3QP

โค้ดคำสั่ง

```
//---------------------------------------Load libraries
#include <esp_now.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>


// WiFi connect timeout per AP. Increase when connecting takes longer.
const uint32_t connectTimeoutMs = 10000;
//////////////// TDS Sensor ////////////////
#define TdsSensorPin 35
#define VREF 1 // analog reference voltage(Volt) of the ADC
#define SCOUNT 30 // sum of sample point
int analogBuffer[SCOUNT]; // store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0, copyIndex = 0;
float averageVoltage = 0, tdsValue = 0, temperature = 25;
//---------------------------------------
#define INTERVAL_TIME_PUMP_1 10000 // 10วินาที
#define INTERVAL_TIME_PUMP_2 1500 // 1.5วินาที
#define Pump_1 26
#define Pump_2 27
//---------------------------------------Define variables to store incoming readings
int receive_Button;
int receive_SendSpeadsheet;
int Memory_data_1;
unsigned long time_1 = 0;
unsigned long time_2 = 0;
#define DO_PIN 33
void sendData();

#define VREF 3300    //VREF (mv)
#define ADC_RES 4096 //ADC Resolution

//Single-point calibration Mode=0
//Two-point calibration Mode=1
#define TWO_POINT_CALIBRATION 0

#define pHPin 36
float analog_pH_value = 0;
float ph_Value;

#define READ_TEMP (25) //Current water temperature ℃, Or temperature sensor function

//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V (1600) //mv
#define CAL1_T (25)   //℃
//Two-point calibration needs to be filled CAL2_V and CAL2_T
//CAL1 High temperature point, CAL2 Low temperature point
#define CAL2_V (1300) //mv
#define CAL2_T (15)   //℃

const uint16_t DO_Table[41] = {
    14460, 14220, 13820, 13440, 13090, 12740, 12420, 12110, 11810, 11530,
    11260, 11010, 10770, 10530, 10300, 10080, 9860, 9660, 9460, 9270,
    9080, 8900, 8730, 8570, 8410, 8250, 8110, 7960, 7820, 7690,
    7560, 7430, 7300, 7180, 7070, 6950, 6840, 6730, 6630, 6530, 6410};

uint8_t Temperaturet;
uint16_t ADC_Raw;
uint16_t ADC_Voltage;
uint16_t DO;
String DO_Value;

int16_t readDO(uint32_t voltage_mv, uint8_t temperature_c) {
  #if TWO_POINT_CALIBRATION == 0
    uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * temperature_c - (uint32_t)CAL1_T * 35;
    int16_t doValue = (voltage_mv * DO_Table[temperature_c] / V_saturation);
    return map(doValue, 0, 16000, 0, 9000);
```

```
  #if TWO_POINT_CALIBRATION == 0
    uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * temperature_c - (uint32_t)CAL1_T * 35;
    int16_t doValue = (voltage_mv * DO_Table[temperature_c] / V_saturation);
    return map(doValue, 0, 16000, 0, 9000);
  #else
    uint16_t V_saturation = (int16_t)((int8_t)temperature_c - CAL2_T) * ((uint16_t)CAL1_V - CAL2_V) / ((uint8_t)CAL1_T - CAL2_T) + CAL2_V;
    int16_t doValue = (voltage_mv * DO_Table[temperature_c] / V_saturation);
    return map(doValue, 0, 16000, 0, 9000);
  #endif
}

  String t;
#define ON_Board_LED 2  //--> Defining an On Board LED, used for indicators when the process of connecting to a wifi router
//
//----------------------------------------SSID dan Password wifi mu gan.
const char* ssid = "CE-ESL"; //--> Nama Wifi / SSID.
const char* password = "ceeslonly"; //-->  Password wifi .
//----------------------------------------

//----------------------------------------Host & httpsPort
const char* host = "script.google.com";
const int httpsPort = 443;
//----------------------------------------
// Initialize DHT sensor.
WiFiClientSecure client; //--> Create a WiFiClientSecure object.

// Timers auxiliar variables
long now = millis();
long lastMeasure = 0;

String GAS_ID = "AKfycbx4B70amh_4hWVnHGY-LieehsfvYmIUm_WxEe7_RuWS3FiJzEaq6Qg56mEegpOQ78fRLg"; //--> spreadsheet script ID

//----------------------------------------

//----------------------------------------Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    int rnd_1;
} struct_message;

struct_message receive_Data; //--> Create a struct_message to receive data.
//----------------------------------------

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Callback when data is received
void sendData(String value,float value2, float value3) {
  Serial.println("==========");
  Serial.print("connecting to ");
  Serial.println(host);

  //----------------------------------------Connect to Google host
  if (!client.connect(host, httpsPort)) {
    Serial.println("connection failed");
    return;
  }
  //----------------------------------------

  //----------------------------------------Proses dan kirim data

  String string_DO = value;
  float string_ph = value2;
  float string_tds = value3;
  String url = "/macros/s/" + GAS_ID + "/exec?DO_Value=" + string_DO + "&ph_Value="+string_ph+ "&tds_Value="+string_tds; //  2 variables
  Serial.print("requesting URL: ");
  Serial.println(url);

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
         "Host: " + host + "\r\n" +
         "User-Agent: BuildFailureDetectorESP8266\r\n" +
         "Connection: close\r\n\r\n");
```

```
    //---------------------------------------
    digitalWrite(ON_Board_LED, HIGH); //--> Turn off the On Board LED when it is connected to the wifi router.
    //---------------------------------------If successfully connected to the wifi router, the IP Address that will be visited is displayed in the serial monitor
    Serial.println("");
    Serial.print("Successfully connected to : ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    //---------------------------------------

    client.setInsecure();    WiFi.begin(ssid, password); //--> Connect to your WiFi router
        Serial.println("");

    pinMode(ON_Board_LED,OUTPUT); //--> On Board LED port Direction output
    digitalWrite(ON_Board_LED, HIGH); //-->

    //---------------------------------------Wait for connection
    Serial.print("Connecting");
    while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      //---------------------------------------Make the On Board Flashing LED on the process of connecting to the wifi router.
      digitalWrite(ON_Board_LED, LOW);
      delay(250);
      digitalWrite(ON_Board_LED, HIGH);
      delay(250);
      //---------------------------------------
    }
    //---------------------------------------
    digitalWrite(ON_Board_LED, HIGH); //--> Turn off the On Board LED when it is connected to the wifi router.
    //---------------------------------------If successfully connected to the wifi router, the IP Address that will be visited is displayed in the serial monitor
    Serial.println("");
    Serial.print("Successfully connected to : ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.println();
    //---------------------------------------

    client.setInsecure();
}

void pH_Sensor() {
  analog_pH_value = analogRead(pHPin);
  float voltage = analog_pH_value * (3.3 / 4095.0);
  ph_Value = (3.3 * voltage);
}

void setup() {
  Serial.begin(115200);
  pinMode(Pump_1,OUTPUT);
  pinMode(Pump_2,OUTPUT);
  pinMode(TdsSensorPin, INPUT);
  pinMode(pHPin,INPUT);

  // Connect to WiFi in STA mode
  WiFi.mode(WIFI_STA);
  //---------------------------------------Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //---------------------------------------

  esp_now_register_recv_cb(OnDataRecv); //--> Register for a callback
}

void DO_Sensor(){
  Temperaturet = (uint8_t)READ_TEMP;
  ADC_Raw = analogRead(DO_PIN);
```

```
  DO_Value = String(readDO(ADC_Voltage, Temperaturet));

}

int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++)
    bTab[i] = bArray[i];
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
    for (i = 0; i < iFilterLen - j - 1; i++)
    {
      if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

void loop() {
  static unsigned long analogSampleTimepoint = millis();
  if (millis() - analogSampleTimepoint > 40U) //every 40 milliseconds,read the analog value from the ADC
  {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and store into the buffer
    analogBufferIndex++;
    if (analogBufferIndex == SCOUNT)
      analogBufferIndex = 0;
  }
  static unsigned long printTimepoint = millis();
  if (millis() - printTimepoint > 800U)
  {
    printTimepoint = millis();
    for (copyIndex = 0; copyIndex < SCOUNT; copyIndex++)
      analogBufferTemp[copyIndex] = analogBuffer[copyIndex];
    averageVoltage = getMedianNum(analogBufferTemp, SCOUNT) * (float)VREF / 1024.0; // read the analog value more stable by the median filtering algorithm, and convert to voltage value
    float compensationCoefficient = 1.0 + 0.02 * (temperature - 25.0); //temperature compensation formula: fFinalResult(25^C) = fFinalResult(current)/(1.0+0.02*(fTP-25.0));
    float compensationVolatge = averageVoltage / compensationCoefficient; //temperature compensation
       tdsValue = (133.42 * compensationVolatge * compensationVolatge * compensationVolatge - 255.86 * compensationVolatge * compensationVolatge + 857.39 * compensationVolatge) * 0.5 * 0.00000001;

  }
  DO_Sensor();
  pH_Sensor();

  if(receive_Button == 1){
      delay(INTERVAL_TIME_PUMP_1);
      digitalWrite(Pump_1,LOW);
  }
  else if(receive_Button == 2){
      delay(INTERVAL_TIME_PUMP_2);
      digitalWrite(Pump_2,LOW);
    }
  else if(receive_Button == 3){
    Wificlientsetup();
    float tds_Value = tdsValue;
    sendData(DO_Value,ph_Value,tds_Value);
    ESP.restart();
  }
}
```

```
    client.setInsecure();
}

void pH_Sensor() {
  analog_pH_value = analogRead(pHPin);
  float voltage = analog_pH_value * (3.3 / 4095.0);
  ph_Value = (3.3 * voltage);
}

void setup() {
  Serial.begin(115200);
  pinMode(Pump_1,OUTPUT);
  pinMode(Pump_2,OUTPUT);
  pinMode(TdsSensorPin, INPUT);
  pinMode(pHPin,INPUT);

  // Connect to WiFi in STA mode
  WiFi.mode(WIFI_STA);
  //----------------------------------------Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //-----------------------------------------

  esp_now_register_recv_cb(OnDataRecv); //--> Register for a callback
}

void DO_Sensor(){
  Temperaturet = (uint8_t)READ_TEMP;
  ADC_Raw = analogRead(DO_PIN);
  ADC_Voltage = uint32_t(VREF) * ADC_Raw / ADC_RES;

  DO_Value = String(readDO(ADC_Voltage, Temperaturet));

}

int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i < iFilterLen; i++)
    bTab[i] = bArray[i];
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
    for (i = 0; i < iFilterLen - j - 1; i++)
    {
      if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
    bTemp = bTab[(iFilterLen - 1) / 2];
  else
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  return bTemp;
}

void loop() {
  static unsigned long analogSampleTimepoint = millis();
  if (millis() - analogSampleTimepoint > 40U) //every 40 milliseconds,read the analog value from the ADC
  {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and store into the buffer
```

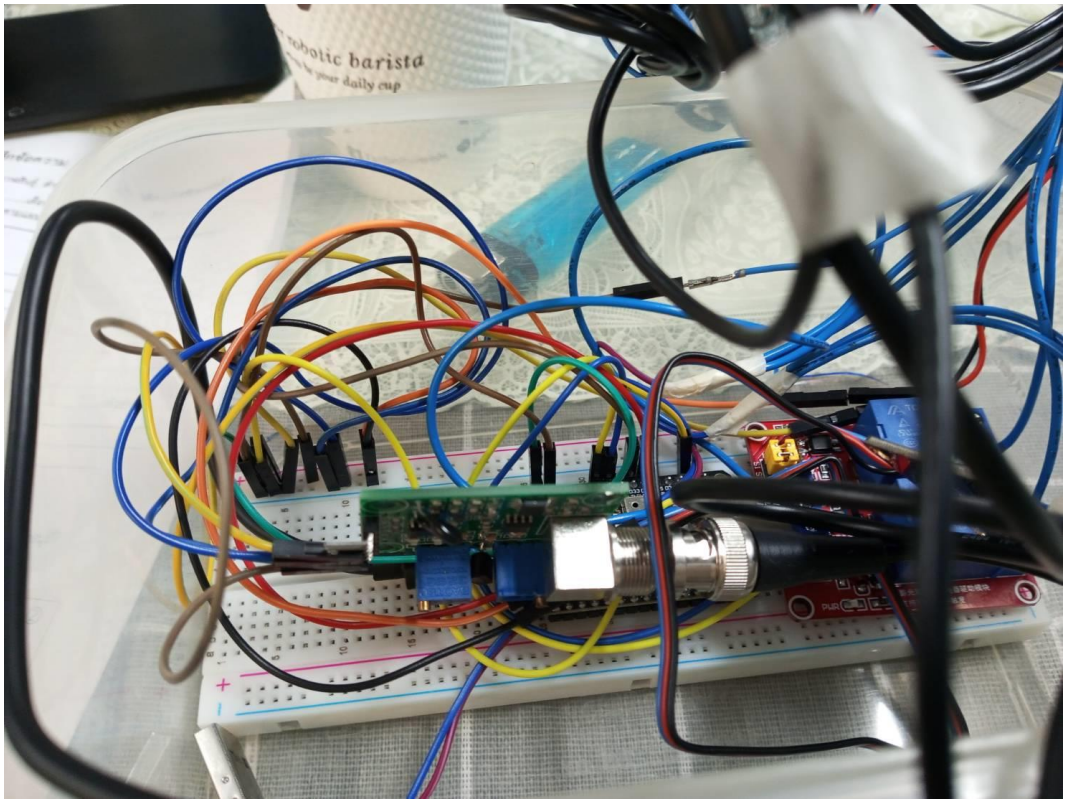
การทำงาน remote control

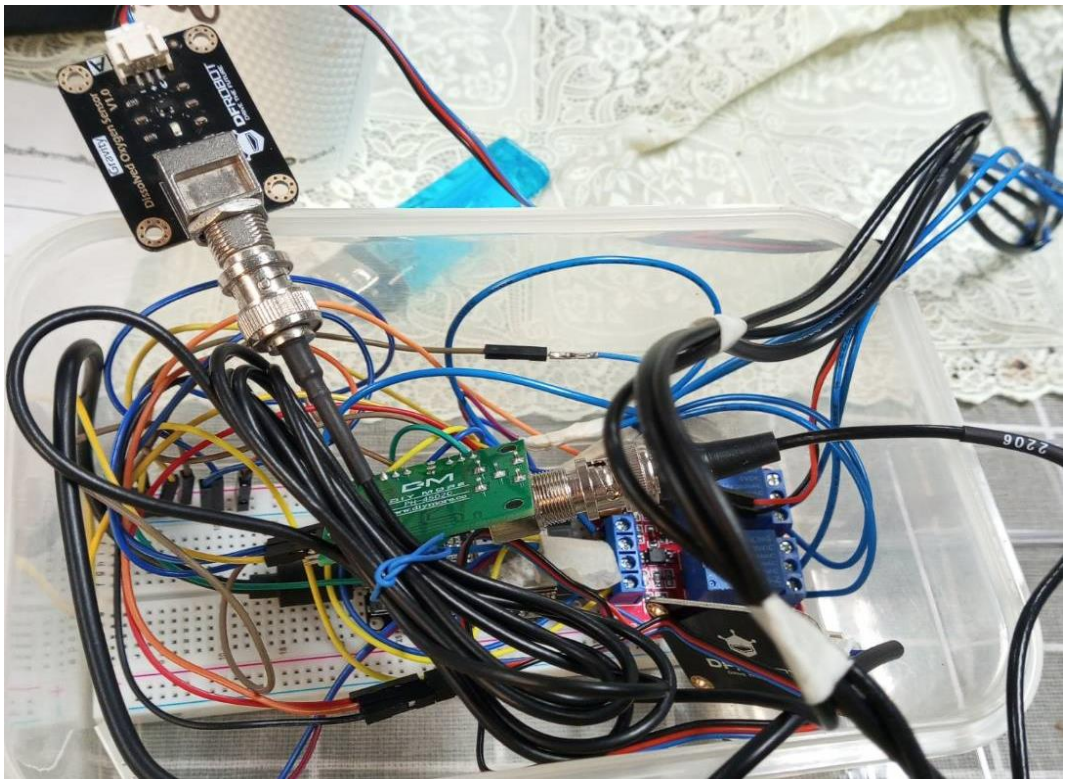
```
#include <esp_now.h>
#include <WiFi.h>

uint8_t broadcastAddress[] = {0xC0, 0x49, 0xEF, 0xD0, 0x30, 0x30}; //--> REPLACE WITH THE MAC Address of your receiver / ESP32 Receiver.

//----------------------------------------Variables to accommodate the data to be sent.
int send_rnd_val_1;
int send_rnd_val_2;
int coolDown;
////////////////////////////////////// Debounce button /////////////////
//int runn = 0;
int Button[4] = {2,4,5,18};
//bool reading[4];
//bool buttonstate[4] = {LOW,LOW,LOW,LOW};
//bool lastbuttonstate[4] = {LOW,LOW,LOW,LOW};
//unsigned long long int lastDebouncetime[4];
//////initfunctions

int debounce(int pin) {
  const int delay_ms = 50; // debounce time in milliseconds
  static int last_readings[4] = {0, 0, 0, 0}; // last 4 readings of each button
  int reading = digitalRead(Button[pin]); // read the current state of the button
  if (reading != last_readings[pin]) { // if the current state is different from the last 4 readings
    delay(delay_ms); // wait for the debounce time
    reading = digitalRead(Button[pin]); // read the current state again
    if (reading != last_readings[pin]) { // if it's still different
      last_readings[pin] = reading; // update the last readings
      return reading; // return the current state
    }
  }
  return -1; // return -1 if the state hasn't changed or has been debounced
}


//---------------------------------------

String success;

//----------------------------------------Structure Pump Relay_1 to send data
// Must match the receiver structure
typedef struct struct_Pump {
    int rnd_1;
} struct_Pump;

struct_Pump send_Data;

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
  if (status ==0){
    success = "Switch Pump Success :)";
  }
  else{
    success = "Switch Pump Fail :(";
  }
  Serial.println(">>>>>");
}

void setup() {
  Serial.begin(115200);
  for(int i=0;i<=3;i++){
    pinMode(Button[i],INPUT);
  }
  WiFi.mode(WIFI_STA); //--> Set device as a Wi-Fi Station.

  //----------------------------------------Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
```

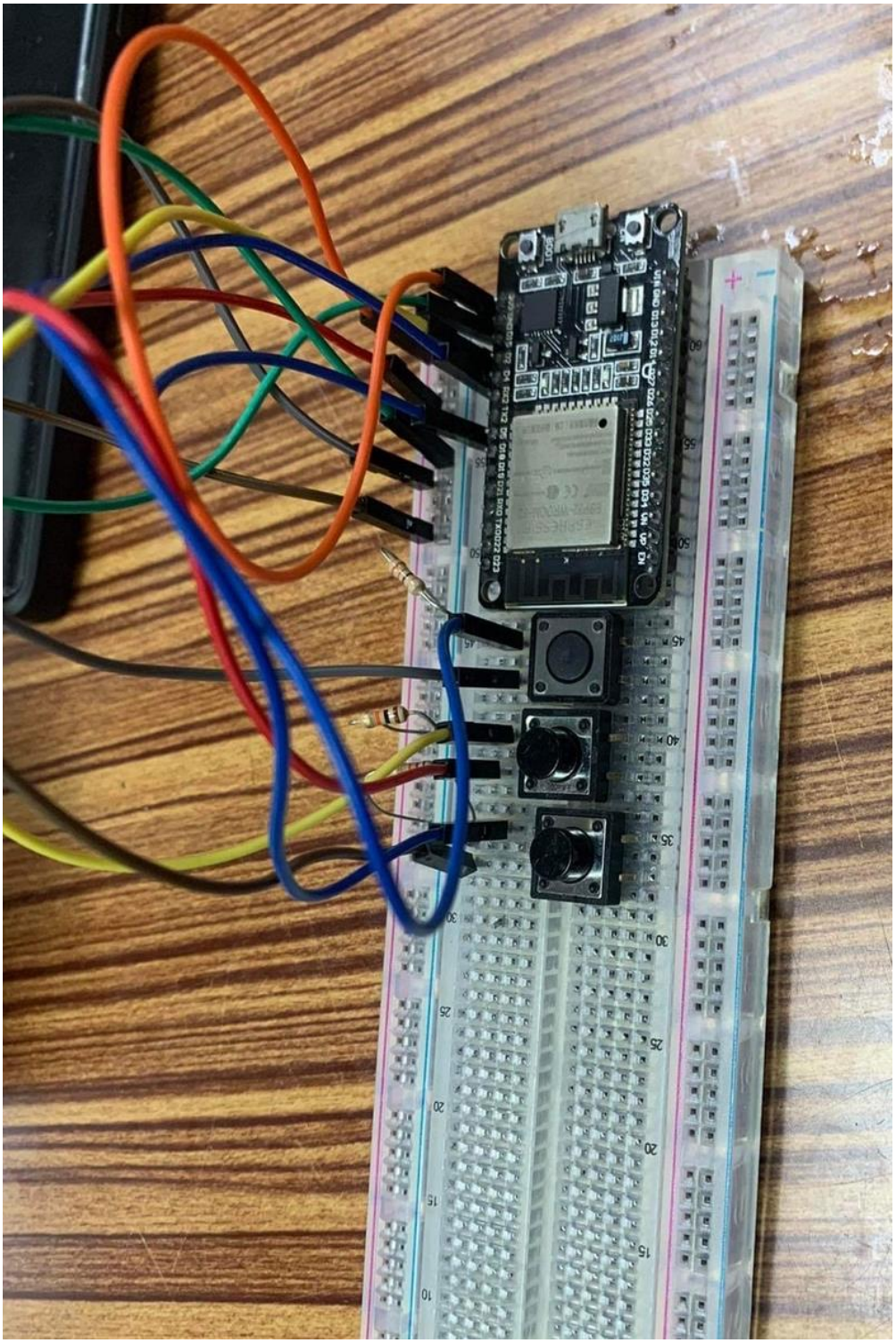
```
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  //----------------------------------------

  //----------------------------------------Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);
  //----------------------------------------

  //----------------------------------------Register peer
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  //----------------------------------------

  //----------------------------------------Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
  //----------------------------------------
}

int key[4]={0,0,0,0};
void loop() {
  debounce(0);
  debounce(1);
  debounce(2);
  debounce(3);
////////////////// Button 1 - Pump_1 ////////////////////////
  if(digitalRead(Button[0]) == HIGH && key[0]==0){
    if(millis() - coolDown > 80){
    //----------------------------------------Set values to send Pump-1
    send_rnd_val_1 = 1;
    send_Data.rnd_1 = send_rnd_val_1;
    //----------------------------------------
    Serial.println();
    Serial.print(">>>>> ");
    Serial.println("Send BUTTON_1");
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &send_Data, sizeof(send_Data));
      if (result == ESP_OK) {
        Serial.println("Sent with success");
      }
      else {
        Serial.println("Error sending the data");
      }
    coolDown = millis();
    key[0]=1;
   }
  }
////////////////// Button 2 - Pump_2 ////////////////////////
  if(digitalRead(Button[1]) == HIGH && key[1]==0){
    if(millis() - coolDown > 80){
    //----------------------------------------Set values to send Pump-1
    send_rnd_val_1 = 2;
    send_Data.rnd_1 = send_rnd_val_1;
    //----------------------------------------
    Serial.println();
    Serial.print(">>>>> ");
    Serial.println("Send BUTTON_2");
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &send_Data, sizeof(send_Data));
      if (result == ESP_OK) {
        Serial.println("Sent with success");
      }
      else {
        Serial.println("Error sending the data");
      }
   key[1]=1;
```

```
      }
      else {
        Serial.println("Error sending the data");
      }
    coolDown = millis();
    key[0]=1;
   }
  }
////////////////// Button 2 - Pump_2 ////////////////////////
  if(digitalRead(Button[1]) == HIGH && key[1]==0){
    if(millis() - coolDown > 80){
    //---------------------------------------Set values to send Pump-1
    send_rnd_val_1 = 2;
    send_Data.rnd_1 = send_rnd_val_1;
    //---------------------------------------
    Serial.println();
    Serial.print(">>>>> ");
    Serial.println("Send BUTTON_2");
    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &send_Data, sizeof(send_Data));
      if (result == ESP_OK) {
        Serial.println("Sent with success");
      }
      else {
        Serial.println("Error sending the data");
      }
   key[1]=1;
   coolDown = millis();
   }
  }
////////////////// Button 3 - Google Sheet ////////////////////////
  if(digitalRead(Button[2]) == HIGH && key[2]==0){
    if(millis() - coolDown > 70){
    //---------------------------------------Set values to send Pump-1
    send_rnd_val_1 = 3;
    send_Data.rnd_1 = send_rnd_val_1;
    //---------------------------------------
      Serial.println("Button 3 : Google Sheets");
      esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &send_Data, sizeof(send_Data));
      key[2]=1;
    }
   coolDown = millis();
  }
////////////////// Button 4 - Line Notify ////////////////////////
  if(digitalRead(Button[3]) == HIGH && key[3]==0){
    if(millis() - coolDown > 70){
    //---------------------------------------Set values to send Pump-1
    send_rnd_val_1 = 4;
    send_Data.rnd_1 = send_rnd_val_1;
    //---------------------------------------
      Serial.println("Button 4 : LINE Notify");
      esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &send_Data, sizeof(send_Data));
      key[3]=1;
    }
   coolDown = millis();
  }
////////-------------- Release key Button --------------////////////
   if(digitalRead(Button[0]) == LOW && key[0]==1){
    key[0]=0;
   }
   if(digitalRead(Button[1]) == LOW && key[1]==1){
    key[1]=0;
   }
   if(digitalRead(Button[2]) == LOW && key[2]==1){
    key[2]=0;
   }
   if(digitalRead(Button[3]) == LOW && key[3]==1){
    key[3]=0;
   }
}
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

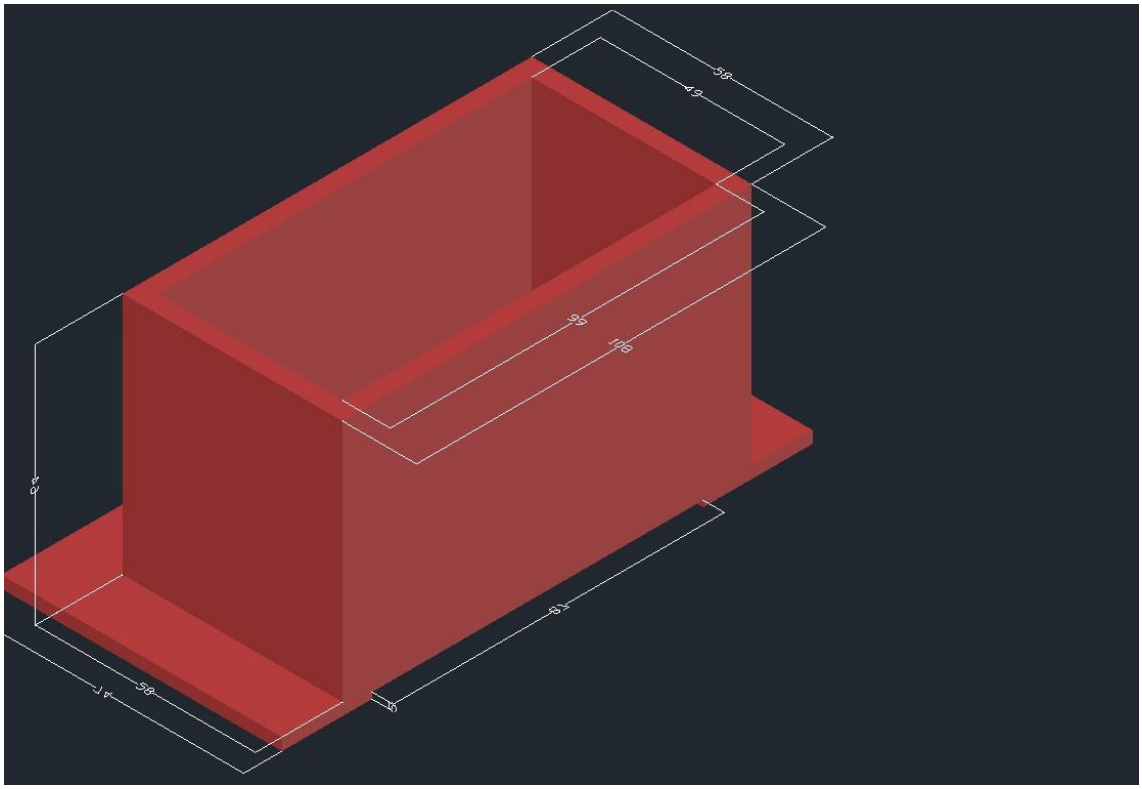
ภาพผลงาน / สิ่งประดิษฐ์

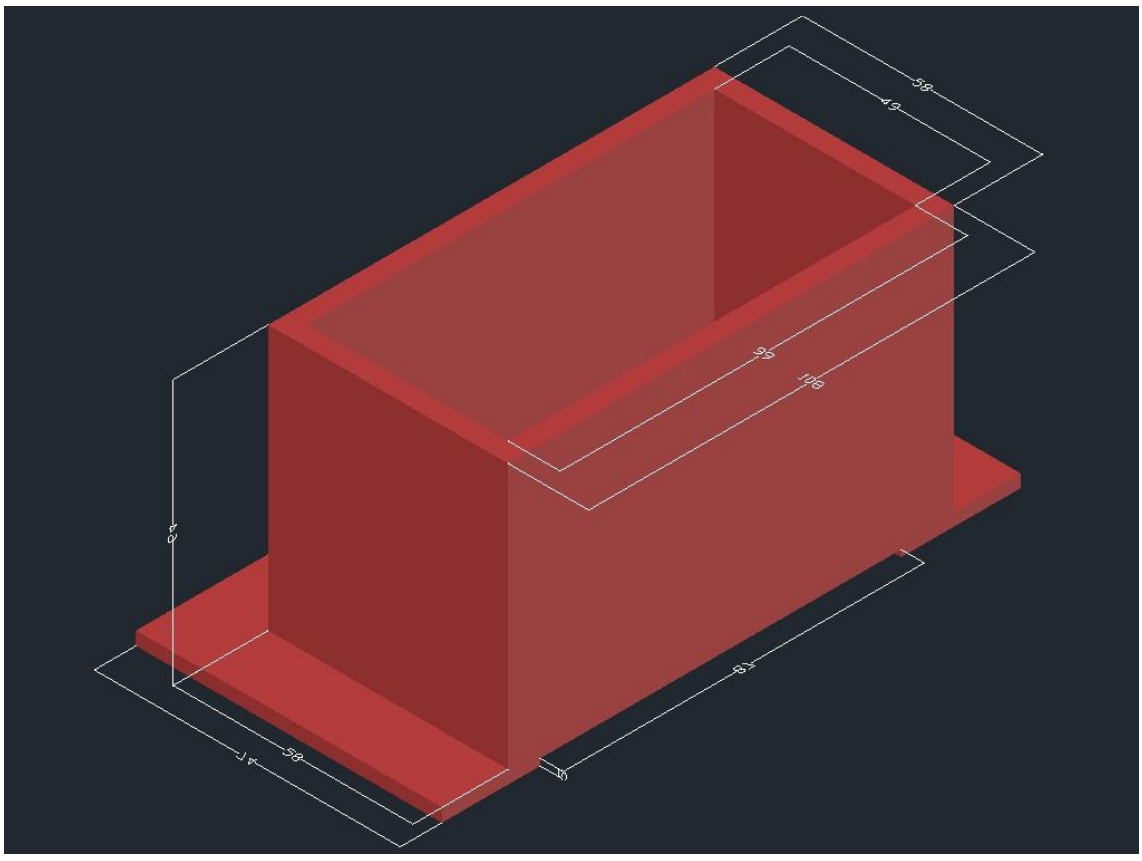


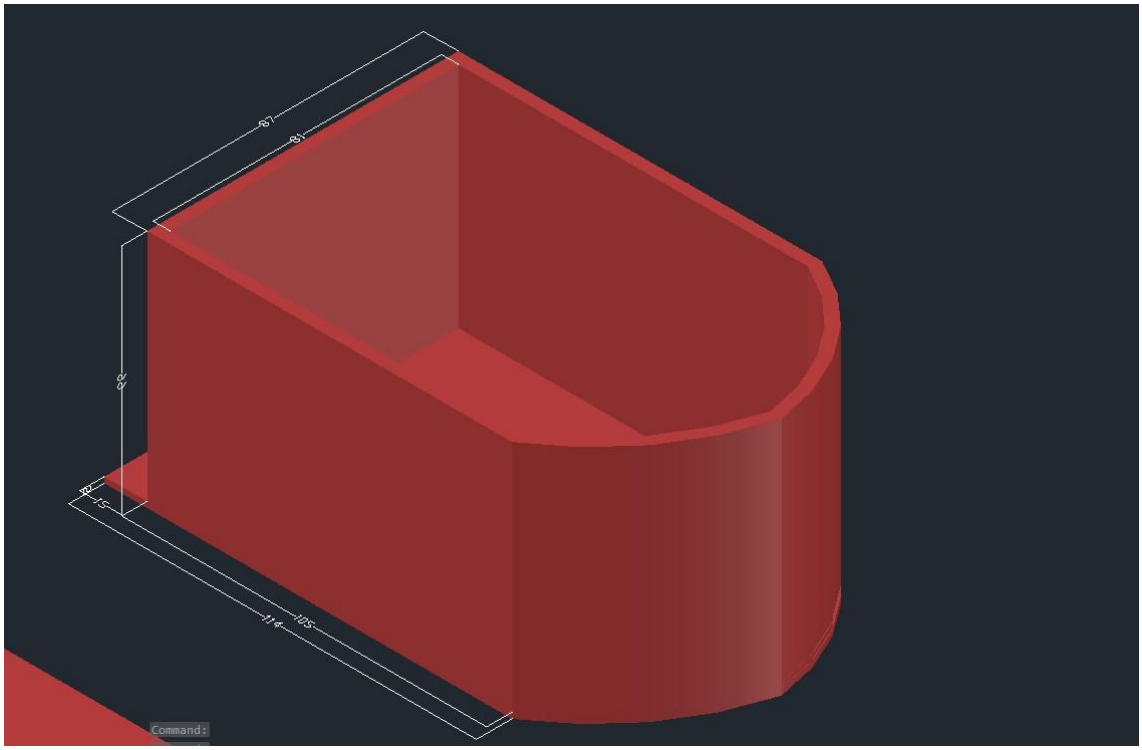


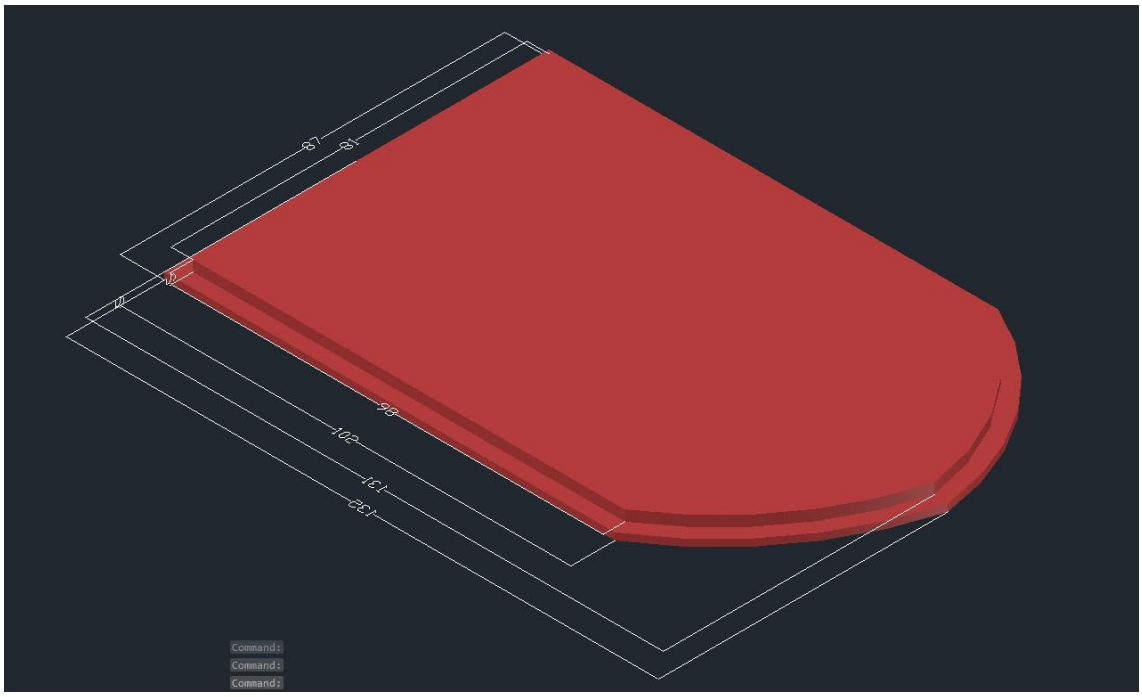


ออกแบบส่วนตัวเรือด้วยโปรแกรม AutoCAD



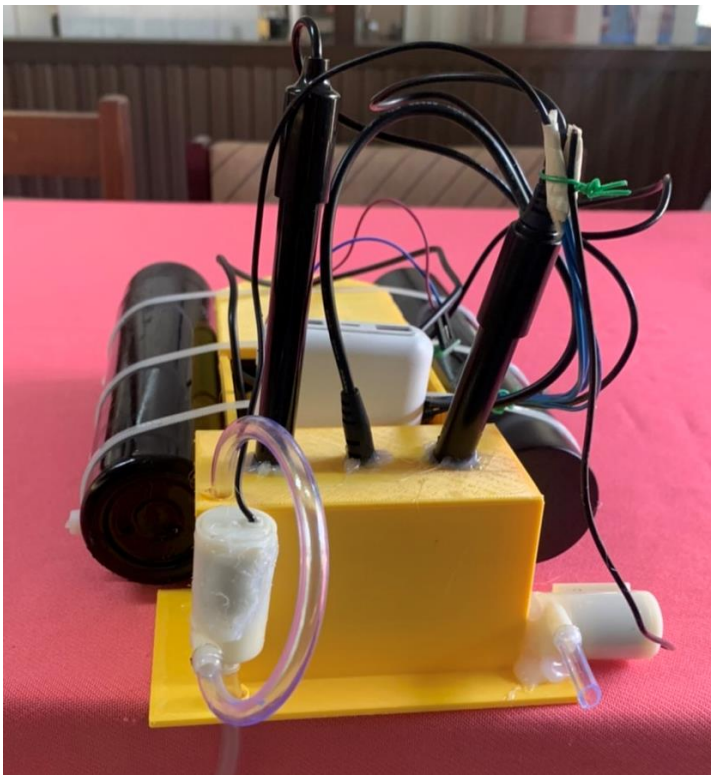



67
81
82
15
114
105
Command:

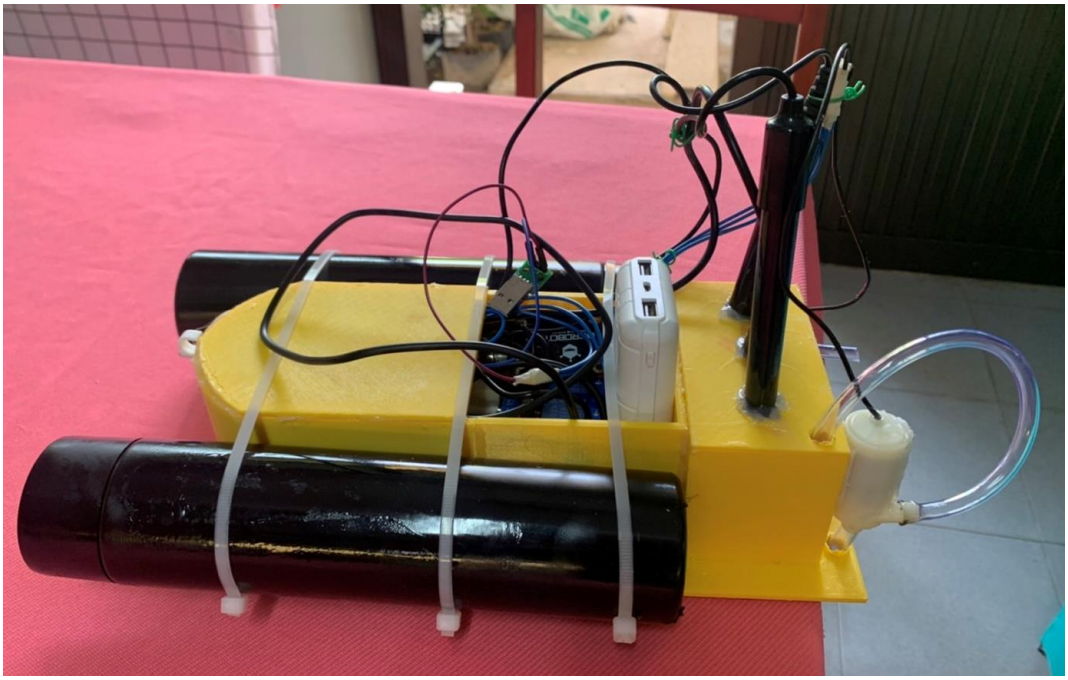

67
81
12
96
102
101
Command:
Command:
Command:

หุ่นยนต์ต้นแบบ

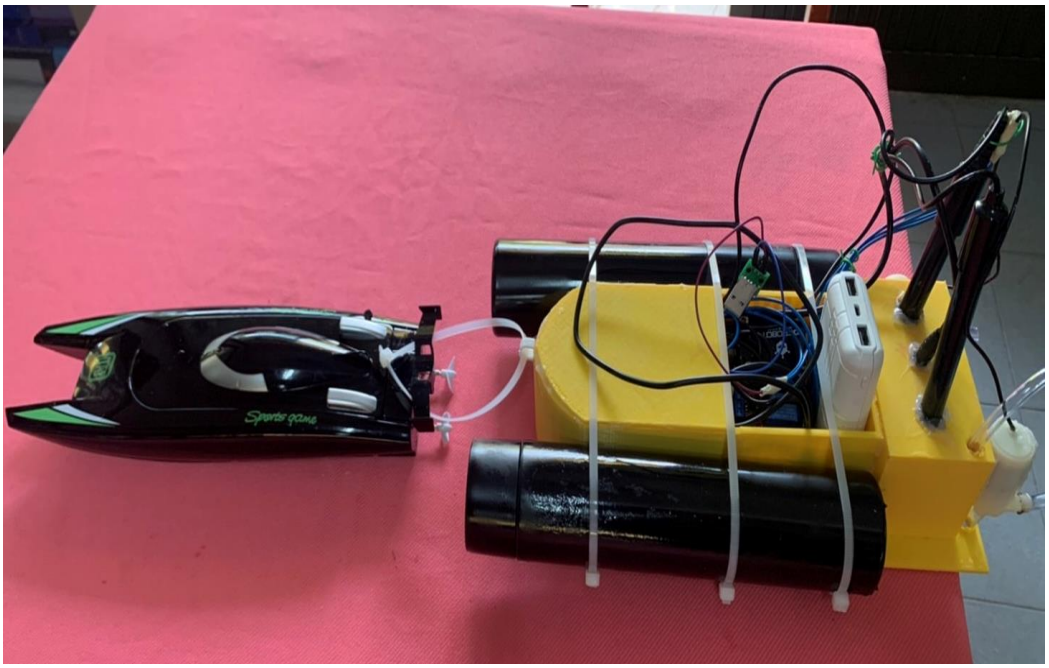
ด้านหน้าส่วนของหุ่นยนต์



ด้านหลังเป็นกล่องเก็บและตรวจวัดคุณภาพน้ำ



ด้านข้างของหุ่นยนต์

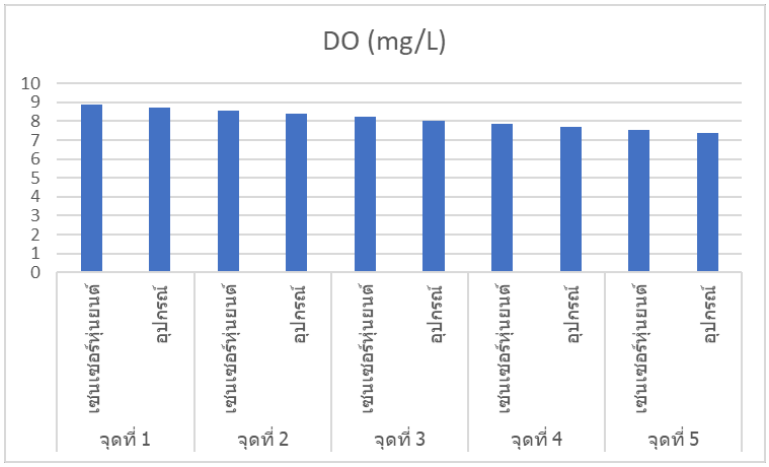


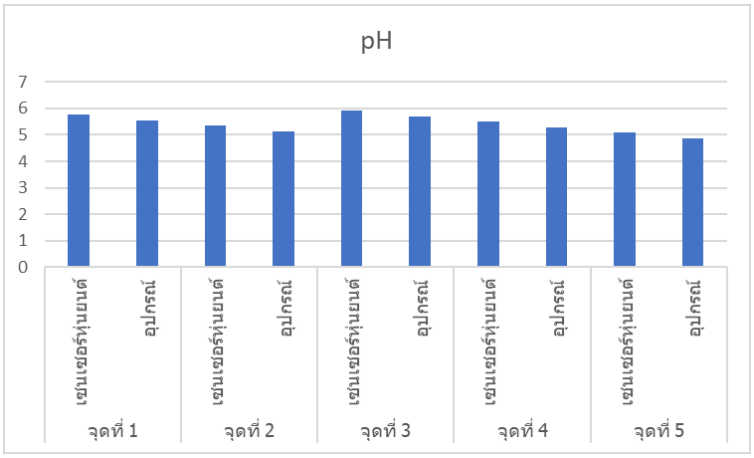
ส่วนประกอบของหุ่นยนต์ตรวจวัดคุณภาพน้ำ

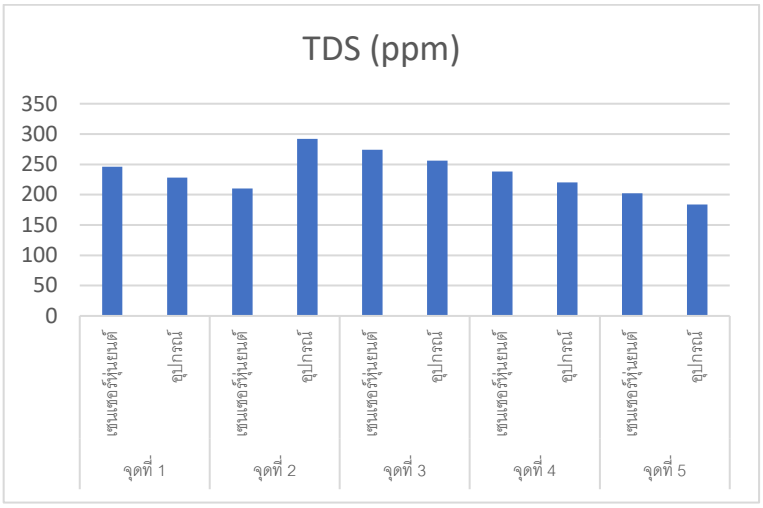
ส่วนต่าง ๆ ของหุ่นยนต์และรีโมท

การทดสอบประสิทธิภาพหุ่นยนต์ตรวจวัดคุณภาพน้ำ

โดยการสุ่มตรวจวัดคุณภาพจำนวน 5 จุด ภายในโรงเรียน โดยใช้หุ่นยนต์ตรวจวัดคุณภาพน้ำ กับ การเก็บน้ำตัวอย่างขึ้นมาตรวจวัดด้วยเครื่องวัดค่า DO miter (Vernier DOV 2-001) และ เครื่องตรวจวัดคุณภาพน้ำดิจิตอล Yieryi Professional (pH และ TDS) เพื่อเปรียบเทียบคุณภาพน้ำ ได้ผลดังแผนภูมิ

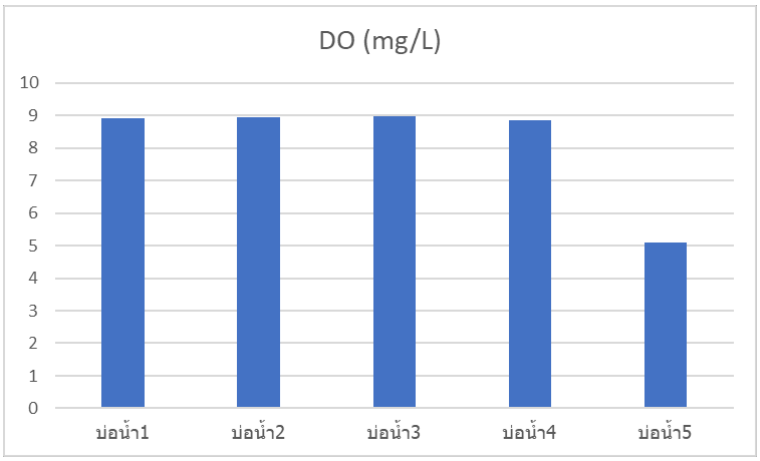


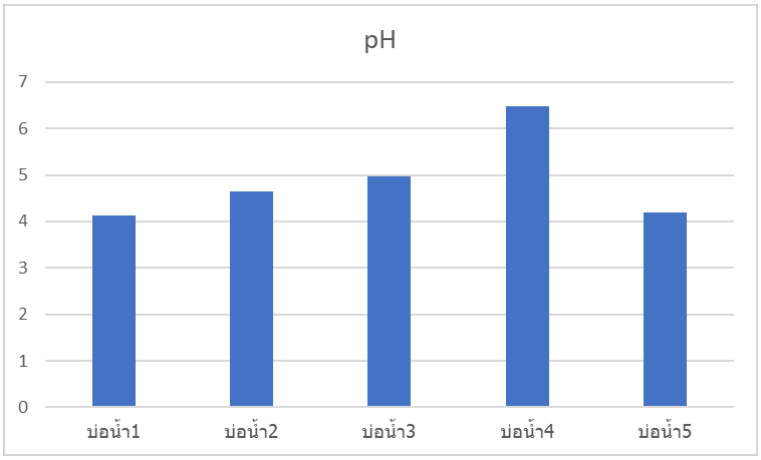


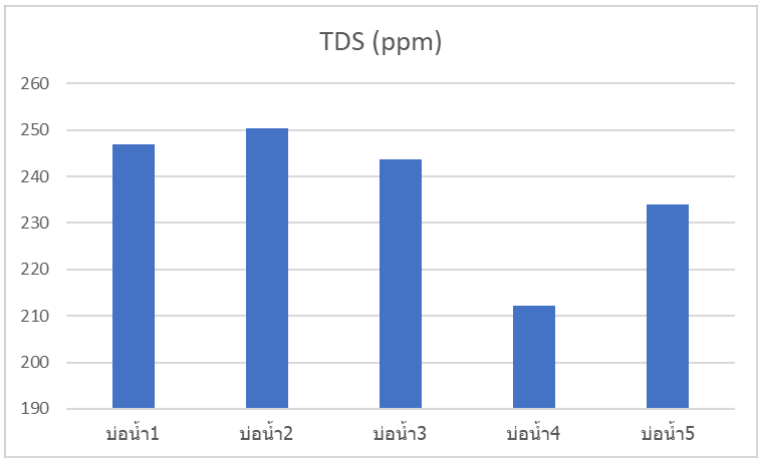


จากแผนภูมิพบว่า เมื่อเปรียบเทียบค่า DO pH และ TDS ที่ตรวจวัดโดยใช้โดยใช้หุ่นยนต์ตรวจวัดคุณภาพน้ำ กับ การเก็บน้ำตัวอย่างขึ้นมาตรวจวัดด้วยเครื่องวัดค่า DO miter (Vernier DOV 2-001) และ เครื่องตรวจวัดคุณภาพน้ำดิจิตอล Yieryi Professional (pH และ TDS) ให้ค่าที่ใกล้เคียงกัน โดยพบว่า หุ่นยนต์ตรวจวัดคุณภาพน้ำวัดค่า DO อยู่ระหว่าง 7.54 – 8.90 มิลลิกรัมต่อลิตร วัดค่า pH อยู่ระหว่าง 5.08 – 5.76 และ ตรวจวัดค่า TDS อยู่ระหว่าง 202 – 246 ppm เมื่อตรวจด้วยเครื่องวัดค่า DO miter (Vernier DOV 2-001) และ เครื่องตรวจวัดคุณภาพน้ำดิจิตอล Yieryi Professional (pH และ TDS) วัดค่า DO อยู่ระหว่าง 7.37 – 8.73 มิลลิกรัมต่อลิตร ตรวจวัดค่า pH อยู่ระหว่าง 4.87 -5.55 และค่า TDS อยู่ระหว่าง 184 – 228 ppm ซึ่งใกล้เคียงกันทุกจุด

ผลการทดสอบในแหล่งน้ำชุมชน







หุ่นยนต์ตรวจวัดคุณภาพน้ำวัดค่า DO อยู่ระหว่าง 5.10 – 8.90 มิลลิกรัมต่อลิตร วัดค่า pH อยู่ระหว่าง 4.08 – 6.96 และ ตรวจวัดค่า TDS อยู่ระหว่าง 212 – 250 ppm ตรวจวัดห่างจากฝั่งประมาณ 2 เมตร

บรรณานุกรม

กรมควบคุมมลพิษ. 2555. **คู่มือปฏิบัติการเก็บตัวอย่างนและตัวอย่างดินเพื่อวิเคราะห์หาสารเคมีป้องกันและกำจัดศัตรูพืช.** กรุงเทพฯ: กรมควบคุมมลพิษ กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม.

ปราโมทย์ ไม้กลัด. (2557). **ทางออกการบริหารจัดการน้ำของไทย.** [Online]. Available: https://tdri.or.th/. [2566, เมษายน 10].

กรมชลประทาน. (2561). **ฝายหวยผึ้งชำรุด.** [Online]. Available: http://www.rid.go.th [2566, เมษายน 10].