



ระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

Automatic pet feeding system

ผู้จัดทำ

นายสุรเชษฐ์	อินทมงคล	รหัส 64366690
นายเอกธนา	ทองเมือง	รหัส 64367345
นางสาวบุญลิตา	เมฆแจ้ง	รหัส 64363651
นางสาวประกายกานต์	ฤทธิพิศ	รหัส 64363880
นางสาวนางสาวภัทรภร	วิธีกกลาง	รหัส 64364658
นางสาวสุพิชชา	พรหมอยู่	รหัส 64366621

เสนอ

ดร.เศรษฐา

ตั้งคำวานิช

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 305394 Special Topic in Embedded Systems

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2567

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 305394 Special Topic in Embedded Systems ทางผู้จัดทำได้พัฒนาระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ เนื่องจากเล็งเห็นถึงปัญหาในปัจจุบันเกี่ยวกับการเลี้ยงสัตว์เลี้ยงที่การดูแลเพื่อให้ได้รับสารอาหารที่เพียงพอและตรงตามเวลาอาจเป็นปัญหาสำหรับผู้เลี้ยงสัตว์ที่มีกิจกรรมอื่นนอกบ้านที่ต้องทำ การจัดเตรียมอาหารที่เหมาะสมและสม่ำเสมอจึงเป็นปัญหาสำหรับเจ้าของสัตว์เลี้ยง

ด้วยเหตุนี้ ระบบให้อาหารสัตว์เลี้ยงอัตโนมัติจึงถูกพัฒนาขึ้น เพื่อตอบโจทย์ความต้องการดังกล่าว โดยระบบนี้จะช่วยให้การให้อาหารสัตว์เลี้ยงได้สะดวกและตรงเวลา อีกทั้งยังสามารถควบคุมการให้อาหารสัตว์เลี้ยงได้ผ่านระยะไกล

รายงานฉบับนี้จะนำเสนอถึงแนวคิด หลักการทำงาน การออกแบบ และขั้นตอนการพัฒนาระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ ซึ่งจะเป็นประโยชน์ในการแก้ปัญหาด้านการจัดการอาหารของสัตว์เลี้ยง และช่วยเพิ่มคุณภาพชีวิตของสัตว์เลี้ยงอีกด้วย

คณะผู้จัดทำ

สารบัญ

1. ที่มาและความสำคัญ	1
2. การออกแบบระบบที่ต้องพัฒนาจริง	1
3. การออกแบบระบบที่จำลองจากระบบที่ต้องพัฒนาจริง	2
4. การเชื่อมต่ออุปกรณ์และวงจรในระบบ	2
5. โฟลว์ชาร์ตการทำงานของระบบเพื่อนำไปพัฒนาเฟิร์มแวร์	6
6. การออกแบบ Topic และตัวแปรที่เกี่ยวข้องกับ MQTT ที่ใช้	7
7. ซอร์สโค้ดโปรแกรมของเฟิร์มแวร์และหน้า UI ของ Dash board ที่พัฒนา.....	8
7.1 ซอร์สโค้ดของระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ.....	8
7.2 หน้า UI ของ Dash board ที่พัฒนา.....	16
8. รูปผลลัพธ์ของระบบที่พัฒนา.....	18
9. ผลการทดลอง.....	19
10. สรุปผลการทดลอง	19
11. รายละเอียดการแบ่งงานในกลุ่ม.....	20

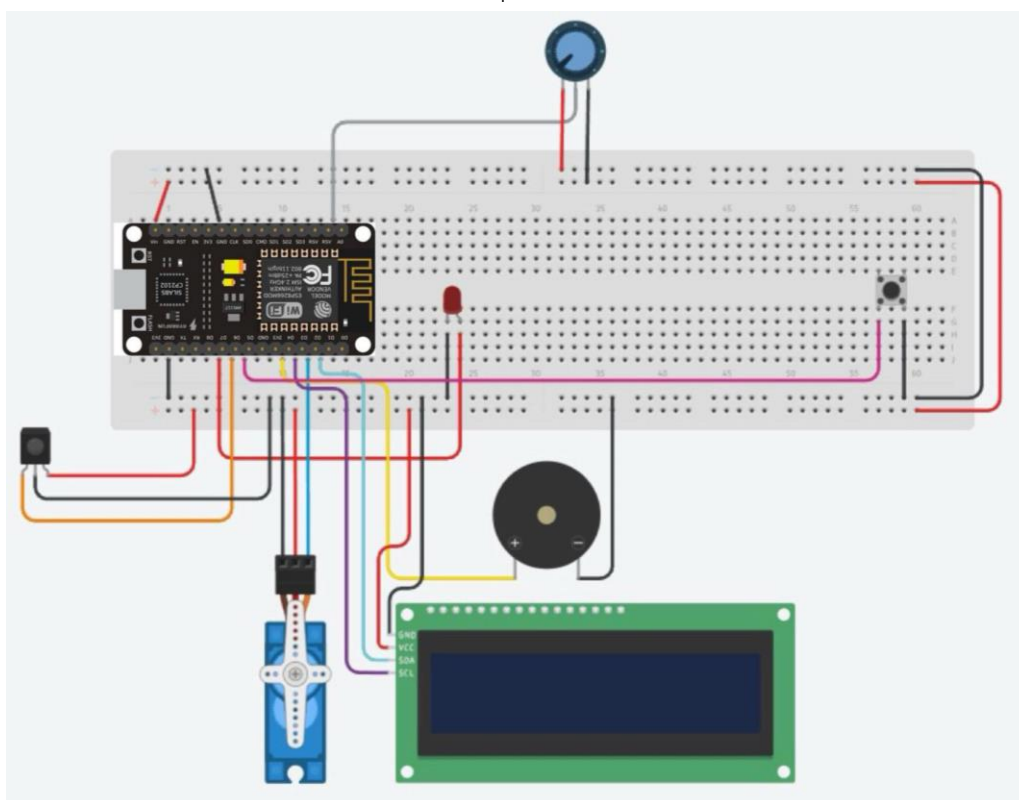
1. ที่มาและความสำคัญในการออกแบบพัฒนาระบบ

ในปัจจุบันการดูแลสัตว์เลี้ยงอย่างสม่ำเสมอและมีประสิทธิภาพเป็นเรื่องที่สำคัญมาก โดยเฉพาะการให้อาหาร ซึ่งเป็นปัจจัยพื้นฐานที่มีผลต่อสุขภาพและการเจริญเติบโตของสัตว์ อย่างไรก็ตาม การให้อาหารด้วยตนเองในเวลาและปริมาณที่เหมาะสมตลอดเวลาอาจเป็นปัญหาสำหรับผู้เลี้ยงที่มีเวลาจำกัดหรือมีกิจกรรมอื่นที่ต้องทำ เช่น การเลี้ยงสัตว์เลี้ยงในหอพักแต่ต้องออกไปทำงานตลอดทั้งวัน นอกจากนี้ อาจเกิดความผิดพลาดในการให้อาหาร เช่น การให้อาหารมากเกินไปหรือน้อยเกินไป ซึ่งอาจส่งผลเสียต่อสุขภาพของสัตว์ได้ ดังนั้น การพัฒนาระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ จึงเป็นแนวทางในการแก้ปัญหาเหล่านี้ ช่วยให้การให้อาหารเป็นไปอย่างมีประสิทธิภาพและสม่ำเสมอ

ความสำคัญของการออกแบบและพัฒนาระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ คือเพื่อเพิ่มประสิทธิภาพในการดูแลสัตว์เลี้ยง การใช้ระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ จะช่วยให้การให้อาหารสัตว์เลี้ยงไม่เป็นปัญหาของคนที่ต้องการเลี้ยงสัตว์แต่ไม่มีเวลาให้อาหารสัตว์ตลอดเวลาเนื่องจากมีสัตว์หลายชนิดทานอาหารไม่เป็นเวลา และระบบให้อาหารสัตว์เลี้ยงอัตโนมัติจะช่วยในการกำหนดปริมาณอาหารที่สัตว์เลี้ยงทานเข้าไป จะได้ไม่มากเกินไปหรือน้อยเกินไปจนทำให้สัตว์เลี้ยงมีสุขภาพไม่ดี

2. การออกแบบระบบที่ต้องพัฒนาจริง

ระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ จะใช้การตั้งเวลาเพื่อให้อาหารสัตว์ตามรอบที่เวลากำหนด และหากอาหารหมดแล้วแต่เวลายังไม่ครบรอบ ระบบจะส่งข้อความไปหาแบบจำลองหน้า NETPIE แล้วผู้ใช้ที่เป็นคนเลี้ยงเห็นว่าควรให้อาหารสัตว์เพิ่มก็จะสามารถกดปุ่มที่หน้า NETPIE เพื่อให้อาหารสัตว์เลี้ยงได้



รูปที่ 1 รูปการจำลองของระบบ

3. การออกแบบระบบที่จำลองจากระบบพัฒนาจริง

3.1 ใช้ Infrared Module (INPUT Digital) ในการตรวจเช็คว่ามีอาหารอยู่ในถาดหรือไม่

- ถ้ามี return HIGH แสดงว่าอาหารหมด

3.2 ใช้ Switch (INPUT Digital) เพื่อยืนยันการตั้งเวลาตามที่ต้องการ

3.3 ใช้ Variable Resistor (INPUT Analog) ในการตั้งเวลาในการให้อาหารสัตว์เลี้ยง เมื่อครบเวลาให้ Servo Motor ทำงาน ไฟ LED ติดและ Buzzer ดังเพื่อจำลองการจ่ายอาหาร

3.4 ใช้ Servo Motor (OUTPUT) เพื่อจำลองสถานการณ์การปล่อยอาหารใส่ถาด

3.5 ใช้ LED (OUTPUT) เพื่อจำลองสถานะการมีอยู่ของอาหารในถาด

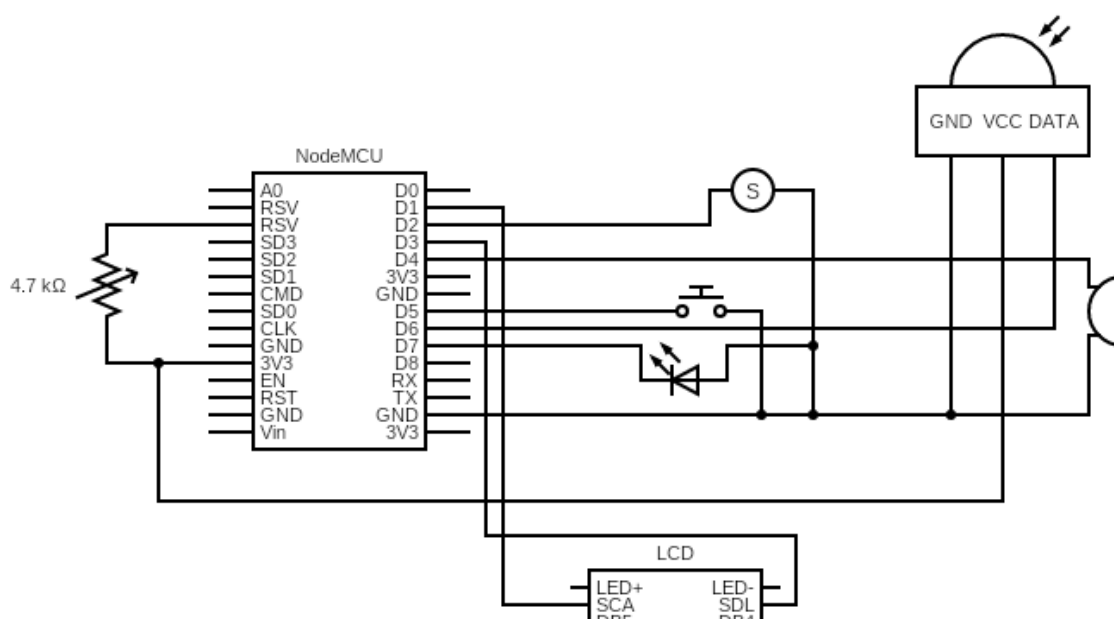
3.6 ใช้ Buzzer (OUTPUT) เพื่อจำลองเสียงเรียกสัตว์เลี้ยง

3.7 ใช้ LCD (OUTPUT)


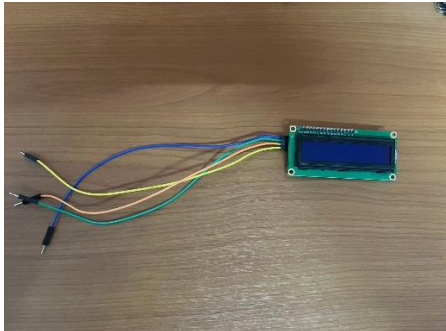
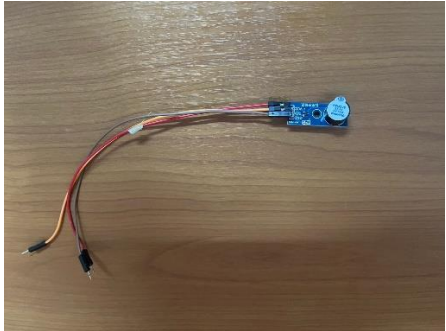

- ใช้จอ LCD แสดงการนับเวลา และสถานะของการตรวจสอบอาหาร





หากใช้ Infrared Module ในการตรวจเช็คว่ามีอาหารอยู่หรือไม่ แล้ว return HIGH แสดงว่าอาหารหมด แต่การใช้ Variable Resistor นับเวลายังไม่ถึงรอบ จะส่งข้อความไป NETPIE เพื่อให้ผู้เลี้ยงสามารถให้อาหารสัตว์เลี้ยงโดยกดปุ่มที่หน้า Dash Board เพื่อสั่งให้ระบบให้อาหารสัตว์เลี้ยงได้


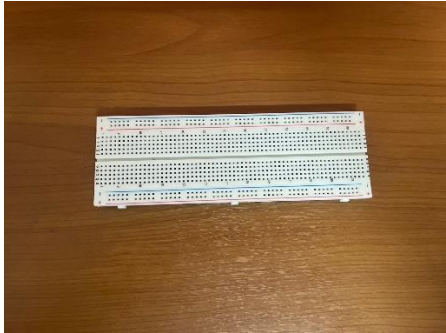

4. การเชื่อมต่ออุปกรณ์และวงจรในระบบ



รูปที่ 2 Schematic วงจรสำหรับระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

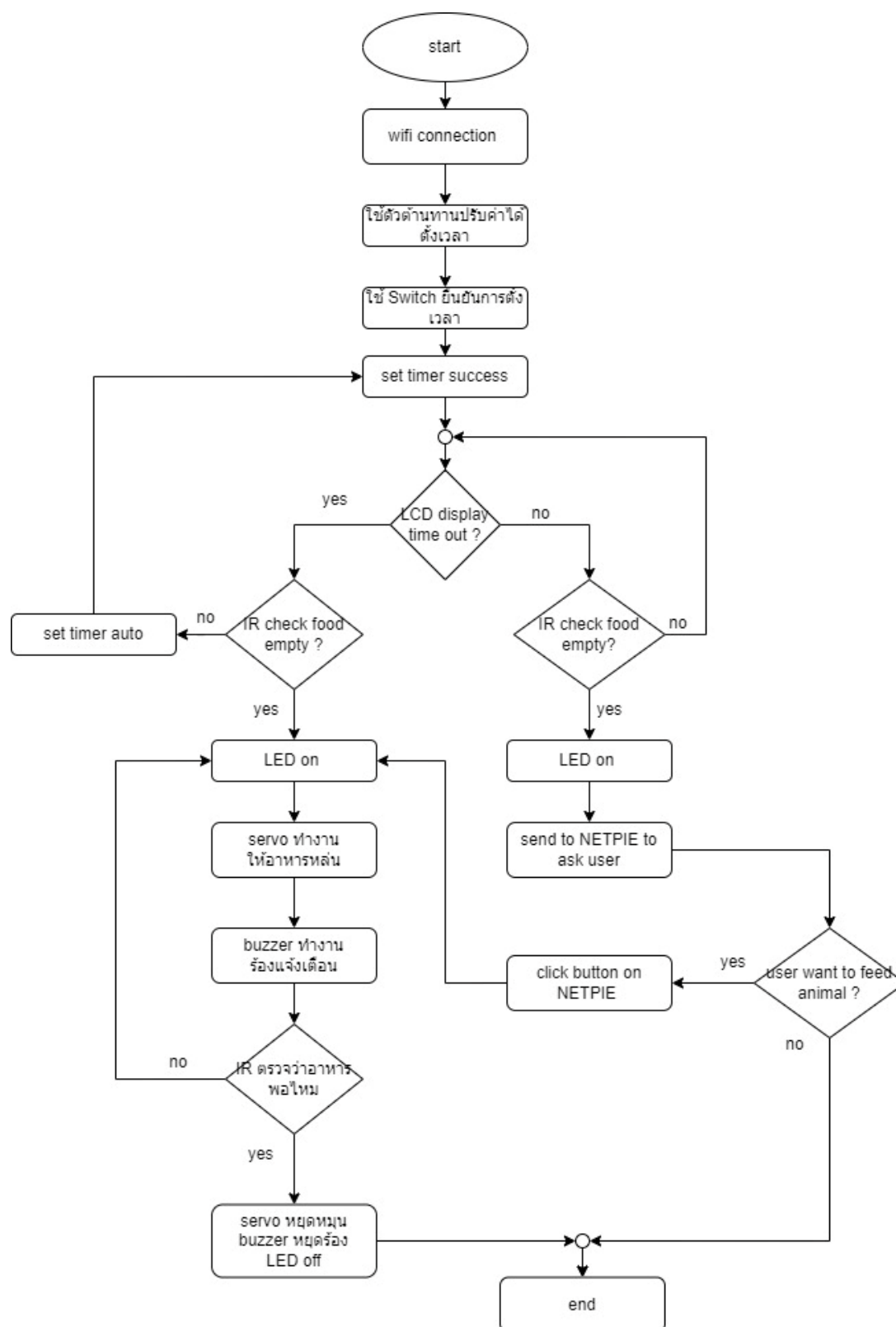
รูปอุปกรณ์	ชื่ออุปกรณ์
	NodeMCU esp8266
	LCD
	Buzzer/Speaker
	Infrared Module

	Variable Resistor
	Servo Motor
	ตัวต้านทาน
	Switch

	สายไฟ
	breadboard
	LED

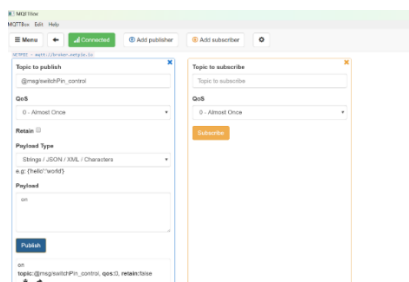
ตารางอุปกรณ์ที่ใช้ในการพัฒนาระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

5. โฟลว์ชาร์ตการทำงานของระบบเพื่อนำไปพัฒนาเฟิร์มแวร์



รูปที่ 3 Flowchart การทำงานของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

6. การออกแบบ Topic และตัวแปรที่เกี่ยวข้องของ MQTT ที่ใช้



รูปที่ 4 การใช้งานโปรโตคอล MQTT

ระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติมีการใช้งานโปรโตคอล MQTT เพื่อส่งและรับข้อมูลจากเซิร์ฟเวอร์ MQTT โดยใช้ topic และตัวแปรที่เกี่ยวข้องกับการควบคุมอุปกรณ์ดังนี้

6.1 MQTT Topics ที่ใช้

- **@msg/ledPin_status** : ใช้เพื่อรับข้อมูลสถานะของ LED pin ผ่าน MQTT โดย topic นี้จะใช้เพื่อเช็คสถานะของ LED จะเปิดหรือปิดจากคำสั่งระยะไกล (Remote)
- **@msg/switchPin_control** : ใช้เพื่อรับคำสั่งในการควบคุมสวิตช์ (switchPin) ผ่าน MQTT ซึ่งสามารถควบคุมการทำงานของระบบให้อาหารสัตว์เลี้ยงผ่านสวิตช์ระยะไกล
- **@shadow/data/update** : ใช้เพื่อส่งข้อมูลอัปเดตสถานะของระบบไปยังเซิร์ฟเวอร์ MQTT หรือ NETPIE เช่น สถานะของ LED (ledPin_status), สถานะของอาหาร (status)

6.2 ตัวแปรที่เกี่ยวข้องกับ MQTT

- **mqtt_client** : เป็นตัวระบุ client ID สำหรับเชื่อมต่อกับ MQTT broker ซึ่งใช้ในการเชื่อมต่อ MQTT ของโปรแกรม
- **mqtt_username** เป็นชื่อผู้ใช้ที่ใช้ในการเชื่อมต่อกับ MQTT broker
- **mqtt_password** เป็นรหัสผ่านที่ใช้เชื่อมต่อกับ MQTT broker
- **client** เป็นตัวแปรที่ใช้ในการสื่อสารกับ MQTT broker (โดยใช้ PubSubClient) ซึ่งรับผิดชอบการ publish และ subscribe ข้อความ

6.3 ฟังก์ชันที่เกี่ยวข้องกับ MQTT

- **reconnect()** เป็นฟังก์ชันที่ใช้ในการเชื่อมต่อกับ MQTT broker ใหม่เมื่อการเชื่อมต่อหลุดและ subscribe topic @msg/ledPin_status และ @msg/switchPin_control หลังจากเชื่อมต่อสำเร็จ
- **callback()** เป็นฟังก์ชันที่จะถูกเรียกเมื่อได้รับข้อความจาก topic ที่ได้ทำการ subscribe เอาไว้ โดยจะตรวจสอบข้อมูลที่รับจาก topic @msg/switchPin_control เพื่อกดปุ่มสวิตช์ว่าเปิดหรือปิด

7. ซอร์สโค้ดและหน้า UI ของ dash board

7.1 ซอร์สโค้ดของระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ

```
#include <dummy.h>
#include <Wire.h>
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// พินสำหรับเซ็นเซอร์และมอเตอร์
#define buzzerPin D4    // Buzzer ต่อกับ D4
#define motorPin D2    // Servo motor ต่อกับ D2
#define irSensorPin D6 // IR sensor ต่อกับ D6
#define switchPin D5   // ขาสำหรับสวิตช์ (เชื่อมต่อกับ D5)
#define potPin A0      // กำหนดขาที่เชื่อมต่อกับ Potentiometer
#define ledPin D7      // กำหนดพินสำหรับ LED (เชื่อมต่อกับ D7)
#define SDA_PIN D1     // กำหนด SDA
#define SCL_PIN D3     // กำหนด SCL

const char* ssid = "Karaket";
const char* password = "ket0614790515";
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_client = "cf4b04f8-1054-4276-ae37-c6aab834acd";
const char* mqtt_username = "TgdEgYbfTiy5eqzsQuXPQZY8cRoLThYb";
const char* mqtt_password = "588dnawbLi4QWuxcPGHYSAmPjaF5oTjm";

int ledPin_Status = 0;
int switchPin_Control = 1;
char msg[100];
WiFiClient espClient;
PubSubClient client(espClient);
```

```

void reconnect()
{
  while (!client.connected())
  {
    Serial.print("Attempting MQTT connection...");
    if (client.connect(mqtt_client, mqtt_username, mqtt_password))
    {
      Serial.println("Connected");

      client.subscribe("@msg/ledPin_status");
      client.subscribe("@msg/switchPin_control");
    }
    else
    {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println("Try again in 5 seconds...");
      delay(5000);
    }
  }
}

Servo myServo;
int potValue = 0;          // ค่าจาก Potentiometer
int lastSwitchState = HIGH; // เก็บสถานะล่าสุดของสวิตช์
int countdownTime = 0;     // เวลารับถอยหลัง (วินาที)
bool countingDown = false; // สถานะการนับถอยหลัง
bool switchPressed = false; // เก็บสถานะว่ามีการกดปุ่มแล้วหรือไม่

LiquidCrystal_I2C lcd(0x27, 16, 2); // กำหนดที่อยู่ I2C ของ LCD

```

```

// ฟังก์ชันการตรวจสอบว่าอาหารหมดหรือไม่
bool isFoodEmpty() {
    int irValue = digitalRead(irSensorPin);
    return (irValue == HIGH); // ถ้าค่า IR เป็น HIGH แสดงว่าอาหารหมด
}

// ฟังก์ชันการให้อาหาร
void feedAnimal() {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Feeding...");
    digitalWrite(buzzerPin, LOW);
    digitalWrite(ledPin, HIGH); // เปิด LED เมื่อเริ่มจ่ายอาหาร
    client.publish("@shadow/data/update", "{\"data\": {\"ledPin_status\": \"on\"}}");

    myServo.write(180); // หมุนเซอร์โวไปที่ 180 องศาเมื่อเริ่มให้อาหาร
    Serial.println("Feeding in progress...");

    // รอจนกว่าอาหารจะเต็ม (IR sensor เปลี่ยนเป็น LOW)
    while (digitalRead(irSensorPin) == HIGH) {
        lcd.setCursor(0, 1);
        lcd.print("Food empty!");
        delay(500); // รอ 500ms เพื่อลดภาระ CPU
    }

    // เมื่ออาหารเต็ม
    Serial.println("Food is full, stopping feeding.");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Feeding done.");
    lcd.setCursor(0, 1);
    lcd.print("Food full");
    digitalWrite(buzzerPin, HIGH);

```

```

digitalWrite(ledPin, LOW); // ปิด LED เมื่อจ่ายอาหารเสร็จ
client.publish("@shadow/data/update", "{\"data\": {\"ledPin_status\": \"off\"}}");
stopFeeding(); // เรียกฟังก์ชันหยุดการให้อาหาร
}

// ฟังก์ชันหยุดการให้อาหาร
void stopFeeding() {
    myServo.write(0);          // หมุนเซอร์โวกลับไป 0 องศาเมื่อหยุดให้อาหาร
    digitalWrite(buzzerPin, HIGH); // ปิดเสียง Buzzer
    Serial.println("Feeding stopped.");
}

// ฟังก์ชันการรอคอยจนกว่าจะสามารถจ่ายอาหารได้
void waitUntilCanFeed() {
    while (lisFoodEmpty()) { // วนลูปตราบใดที่อาหารยังเต็มอยู่
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Food full, wait");
        Serial.println("Food is already full. Waiting...");
        countdownTime += 5; // เพิ่มเวลานับถอยหลังอีก 5 วินาที
        lcd.setCursor(0, 1);
        lcd.print("Next try in 5s");
        delay(5000);        // รอ 5 วินาทีก่อนตรวจสอบอีกครั้ง
    }
    // เมื่ออาหารไม่เต็มแล้ว สามารถเริ่มให้อาหารได้
    Serial.println("Food is now empty. Starting feeding.");
    feedAnimal();           // เริ่มให้อาหาร
}

int switchState = digitalRead(switchPin);
bool remotePressed = false; // ตัวแปรเพื่อบันทึกว่ามีการกดจาก NETPIE

```

```

void callback(char* topic, byte* payload, unsigned int length) {
    String msg;
    for (int i = 0; i < length; i++) {
        msg += (char)payload[i];
    }

    if (String(topic) == "@msg/switchPin_control") {
        if (msg == "on") {
            Serial.println("Turning switch ON");
            remotePressed = true; // บันทึกว่ามีการกดจาก NETPIE
            countingDown = true; // เปิดสวิตช์
        }
        else if (msg == "off") {
            Serial.println("Turning switch OFF");
            remotePressed = false;
        }
    }
}

void notifyFoodEmpty() {
    String message = "{\"data\": {\"status\": \"อาหารหมด\"}}"; // ข้อความที่ต้องการส่ง
    client.publish("@shadow/data/update", message.c_str()); // ส่งข้อความไปยัง shadow
    Serial.println("Sent message: " + message); // พิมพ์ข้อความที่ส่งใน Serial Monitor
}

void notifyFoodfull() {
    String message = "{\"data\": {\"status\": \"อาหารเต็มแล้ว\"}}"; // ข้อความที่ต้องการส่ง
    client.publish("@shadow/data/update", message.c_str()); // ส่งข้อความไปยัง shadow
    Serial.println("Sent message: " + message); // พิมพ์ข้อความที่ส่งใน Serial Monitor
}

```

```

void setup() {
  Wire.begin(SDA_PIN, SCL_PIN); // เริ่มต้น I2C ที่กำหนดพอร์ต SDA และ SCL
  lcd.begin(16, 2); // เริ่มต้น LCD โดยระบุจำนวนคอลัมน์และแถว
  lcd.backlight(); // เปิดไฟแบคไลท์
  lcd.setCursor(0, 0); // ตั้งค่าตำแหน่งของเคอร์เซอร์ที่ (0, 0)
  lcd.print("System ready");
  delay(2000);
  lcd.clear();

  Serial.begin(115200);
  myServo.attach(motorPin); // เชื่อมต่อ Servo motor เข้ากับ D2
  myServo.write(0);        // กำหนดให้เซอร์โวหมุนไปที่มุม 0 องศาเริ่มต้น
  Serial.println("Servo motor initialized.");

  pinMode(switchPin, INPUT); // ตั้งค่าเป็น Input สำหรับสวิตช์
  pinMode(buzzerPin, OUTPUT); // ตั้งค่าเป็น Output สำหรับ Buzzer
  pinMode(irSensorPin, INPUT); // ตั้งค่าเป็น Input สำหรับ IR Sensor
  pinMode(ledPin, OUTPUT);    // ตั้งค่าเป็น Output สำหรับ LED
  digitalWrite(buzzerPin, HIGH); // ปิดเสียงแจ้งเตือนเริ่มต้น
  digitalWrite(ledPin, LOW);    // ปิด LED เริ่มต้น
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
  client.subscribe("@msg/switchPin_control");
}

```



```

void loop() {

    if (!client.connected()) {
        reconnect();
    }
    client.loop(); // ตรวจสอบและประมวลผล MQTT ทุกครั้งใน loop

    // ตรวจสอบสถานะอาหารตลอดเวลา
    lcd.setCursor(0, 0);
    if (isFoodEmpty()) {
        lcd.print("Food empty   "); // เคลียร์ข้อความเมื่ออาหารหมด
        notifyFoodEmpty();
    } else {
        lcd.print("Food full   "); // เคลียร์ข้อความเมื่ออาหารเต็ม
        notifyFoodfull();
    }

    // อ่านค่า potentiometer เพื่อตั้งเวลา
    potValue = analogRead(potPin);
    countdownTime = map(potValue, 0, 1023, 10, 600); // แปลงค่าเป็นเวลานับถอยหลัง (10 ถึง 600 วินาที)

    lcd.setCursor(0, 1);
    lcd.print("Time: ");
    lcd.print(countdownTime);
    lcd.print(" sec");

    Serial.print("Countdown time set to: ");
    Serial.print(countdownTime);
    Serial.println(" seconds.");

    int switchState = digitalRead(switchPin);

    // พิมพ์สถานะของสวิตช์ออกมาใน Serial Monitor
    Serial.print("Switch state: ");
    Serial.println(switchState);
}

```

```
// ตรวจสอบการเปลี่ยนแปลงสถานะของสวิตช์เมื่อยังไม่เริ่มนับถอยหลัง
if (switchState == LOW && lastSwitchState == HIGH && !countingDown &&
!switchPressed) {
    Serial.println("Switch pressed! Starting countdown...");
    countingDown = true; // เริ่มนับถอยหลังเมื่อกดสวิตช์
    switchPressed = true; // บันทึกว่าปุ่มถูกกดแล้ว
}

// รีเซ็ตสถานะ switchPressed เมื่อปล่อยปุ่ม
if (switchState == HIGH && lastSwitchState == LOW) {
    switchPressed = false;
}

lastSwitchState = switchState;

// ถ้ากำลังนับถอยหลัง
if (countingDown) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Counting down...");
    Serial.println("Counting down...");
    while (countdownTime > 0) {
        lcd.setCursor(0, 1);
        lcd.print("Time left: ");
        lcd.print(countdownTime);
        delay(1000); // รอ 1 วินาที
        countdownTime--;
        client.loop();
    }
    // เมื่อเวลานับถอยหลังถึงศูนย์
    Serial.println("Time's up! Checking food status...");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Time's up!");
}
```

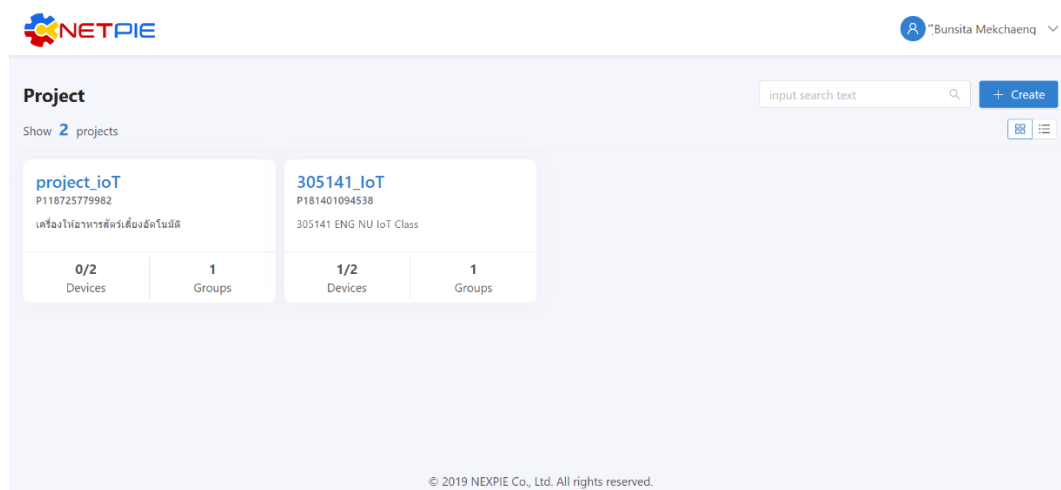
```
// ถ้าอาหารเต็ม จะเรียกฟังก์ชันใหม่
if (!isFoodEmpty()) {
    waitUntilCanFeed(); // เริ่มฟังก์ชันรอคอยจนสามารถจ่ายอาหารได้
} else {
    // ถ้าอาหารไม่เต็ม ให้จ่ายอาหารได้ทันที
    feedAnimal();
}

countingDown = false; // รีเซ็ตสถานะการนับถอยหลังหลังจากให้อาหารเสร็จสิ้น
delay(3000); // รอให้แสดงผลครบ 3 วินาที
lcd.clear();
}

delay(100); // หน่วงเวลาในแต่ละรอบของ loop
}
```

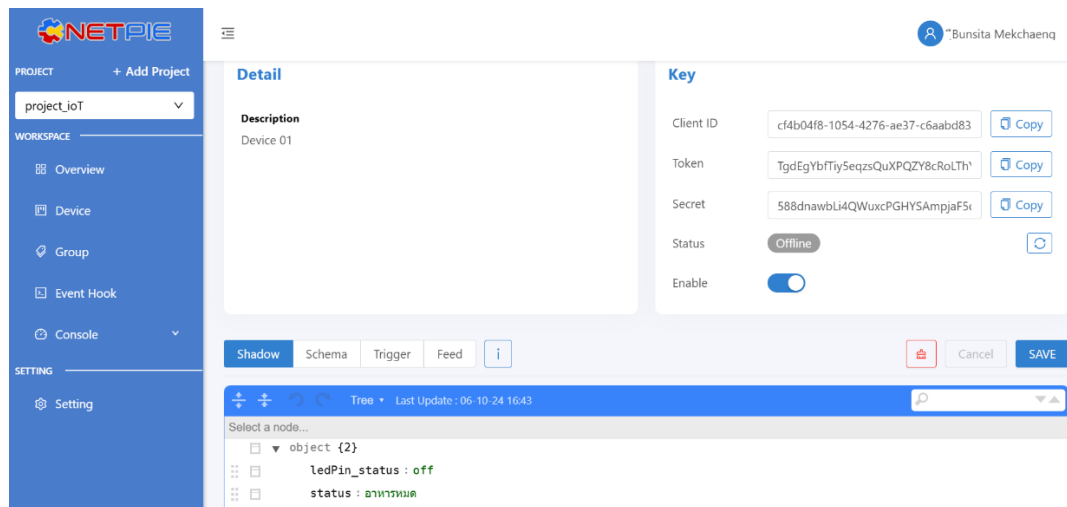
7.2 หน้า UI ของ Dash board ระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

ทำการสร้างโปรเจกขึ้นมาใหม่เพื่อใช้สำหรับการทำงานของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติที่จะพัฒนาขึ้น



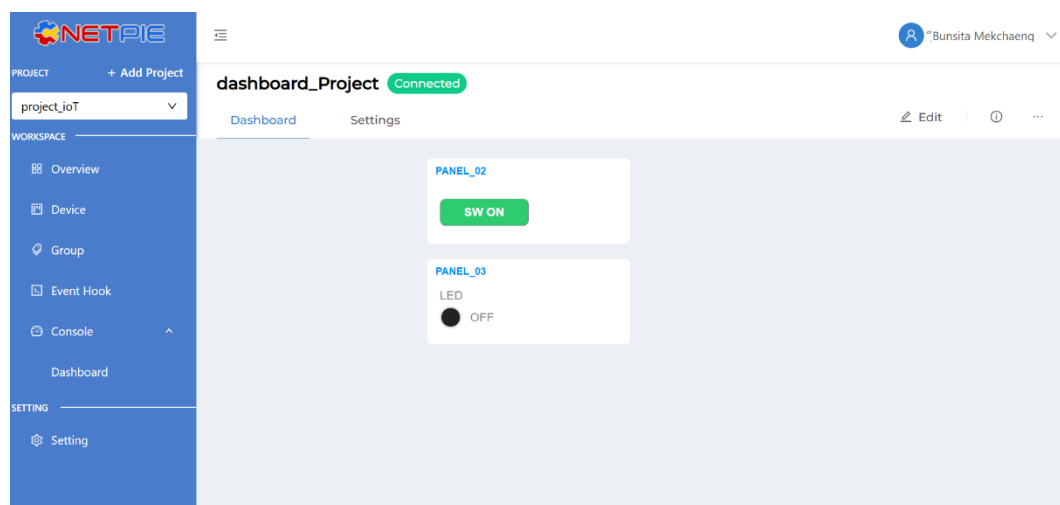
รูปที่ 5 สร้างโปรเจกใหม่สำหรับระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

สร้าง Device01 ขึ้นมาเพื่อใช้สำหรับการพัฒนาระบบ โดยที่แถบ Shadow จะแสดงมี status ใช้สำหรับแสดงสถานะของอาหารที่ถาด และแสดงสถานะการเปิด-ปิดไฟของ LED บนบอร์ด



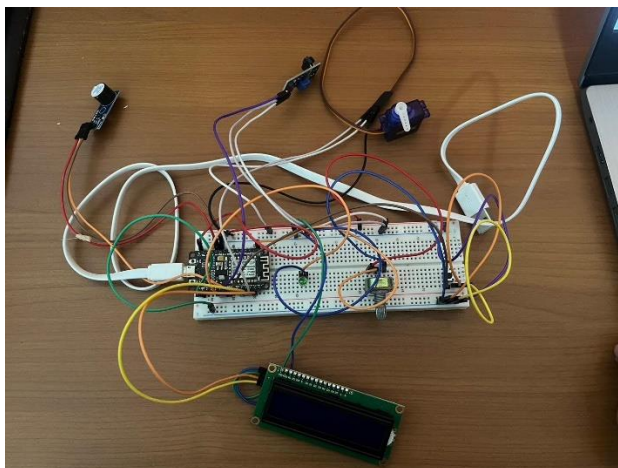
รูปที่ 6 ค่า Client ID, Token, Secret เชื่อมต่อกับบอร์ดและ MQTT

หน้า UI ที่ Dash board ของ NETPIE จะมี **PANEL_02** เป็น button widget สำหรับให้ผู้ใช้กดปุ่มให้อาหารสัตว์เลี้ยง ถ้ากดปุ่ม ON ที่ **PANEL_02** จะส่งผลให้ Servo motor ที่วงจรม้วน และมี **PANEL_03** เป็น Light Indicator เพื่อแสดงการเปิด-ปิดของไฟ LED บนบอร์ดวงจร

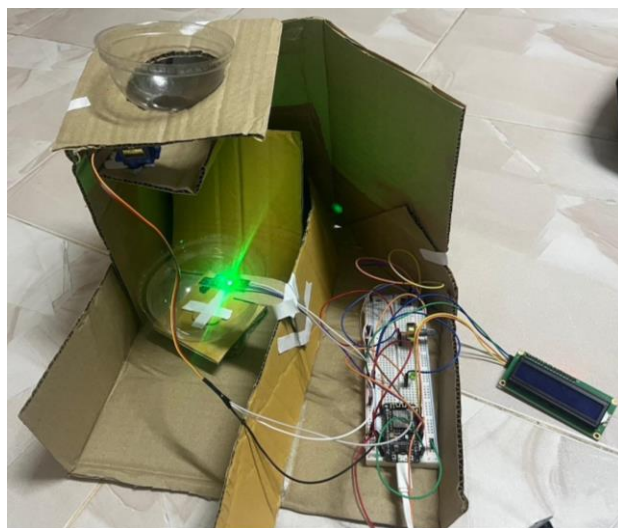


รูปที่ 7 หน้า Dash board ของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

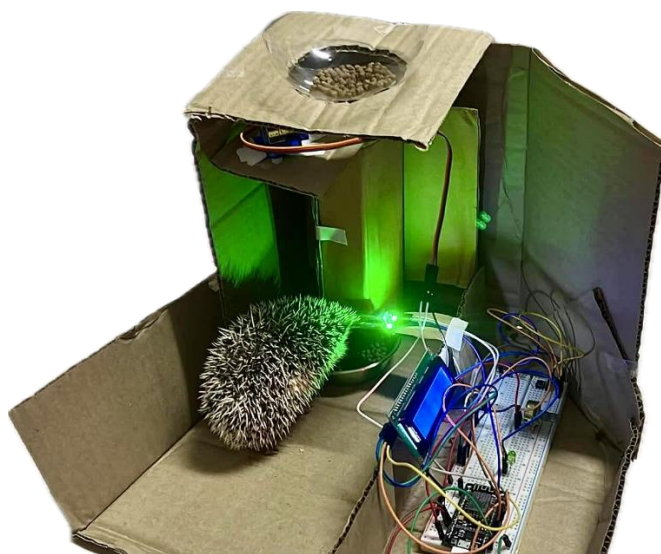
8. รูปผลลัพธ์ของระบบที่พัฒนา



รูปที่ 8 วงจรของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ



รูปที่ 9 Model จำลองของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ



รูปที่ 10 จำลองการทำงานของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ

9. ผลการทดลอง

จากการออกแบบและการจำลองของระบบให้อาหารสัตว์เลี้ยงโดยอัตโนมัติ โดยการประยุกต์ใช้บอร์ดไมโครคอนโทรลเลอร์ esp8266 Node MCU มีผลการทดลองคือ เมื่อเริ่มการทำงานของระบบให้อาหารสัตว์เลี้ยงอัตโนมัติจะเชื่อมต่อกับ WIFI โดยอัตโนมัติ จากนั้นจะต้องตั้งเวลาในการให้อาหารสัตว์เลี้ยง เมื่อตั้งเวลาสำเร็จจะแสดงผลออกทาง LCD และเวลาจะนับถอยหลัง จะแบ่งออกเป็น 3 กรณี คือ

1. การตั้งเวลาโดยใช้ตัวต้านทานปรับค่าได้หมุนเพื่อเลือกเวลาที่ต้องการ แล้วกดปุ่มสวิตช์เพื่อยืนยันเวลาที่ต้องการ
2. เวลาจะนับถอยหลังจนหมดเวลาที่ตั้งไว้ จะมี Infrared module (IR sensor) คอยตรวจเช็คว่ามีอาหารที่ถาดหมดหรือยัง โดยถ้าอาหารที่ถาดยังไม่หมด จะทำการตั้งเวลาต่อไปโดยอัตโนมัติ แต่ถ้าหากอาหารที่ถาดหมดแล้ว ไฟ LED จะติดเพื่อจำลองว่าอาหารหมด ซึ่ง Servo motor จะหมุนจำลองว่าปล่อยอาหารใส่ถาดและ Buzzer จะร้องแจ้งเตือนจำลองการเรียกสัตว์เลี้ยง เมื่ออาหารถูกปล่อยมาจน Infrared module (IR sensor) ตรวจจับอาหารได้ ไฟ LED จะดับเพื่อจำลองถึงมีอาหารในถาดเพียงพอแล้ว ซึ่ง Servo motor จะหยุดหมุนจำลองถึงการหยุดปล่อยอาหารใส่ถาดและ Buzzer จะหยุดร้องแจ้งเตือน
3. เวลาจะนับถอยหลังแต่ยังไม่หมดเวลาที่ตั้งไว้ แต่ Infrared module (IR sensor) ตรวจเช็คว่ามีอาหารที่ถาดหมด ไฟ LED จะติดจำลองว่าอาหารหมด และส่งข้อความแจ้งเตือนหาผู้ใช้งานโปรโตคอล MQTT ไปยัง NETPIE เพื่อบอกคำสั่ง ถ้าหากผู้ใช้ต้องการที่จะให้อาหารสัตว์เลี้ยงในขณะที่เวลาที่ตั้งไว้ยังไม่หมด จะคลิกปุ่ม ON ที่ Dash board ของ NETPIE เพื่อสั่งการให้ Servo motor หมุนจำลองปล่อยอาหารใส่ถาดและ Buzzer ร้องแจ้งเตือน จนกว่า Infrared module (IR sensor) ตรวจจับอาหารได้ จึงหยุด และที่ Dash board ของ NETPIE จะมี LED จำลองสถานะการปล่อยอาหาร ถ้าไฟติดจำลองว่าอาหารยังไม่เต็มถาด ถ้าไฟดับจำลองว่าอาหารเต็มถาด

ระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ มีระบบการแจ้งเตือนและแสดงผลผ่านโปรโตคอล MQTT เมื่ออาหารหมดหรือเต็ม ไปยัง NETPIE โดยผู้ใช้สามารถกดปุ่มบนหน้า Dash board ของ NETPIE เพื่อสั่งการปล่อยอาหารได้ ตามผลการทดลองข้างต้น

10. สรุปผลการทดลอง

จากทดลองของระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ โดยการประยุกต์ใช้บอร์ดไมโครคอนโทรลเลอร์ esp8266 Node MCU ร่วมกับการสื่อสารระหว่างโปรโตคอล MQTT ที่ NETPIE และโมดูลต่างๆ ในการเชื่อมต่อรวมถึงการเขียนโปรแกรมด้วยภาษา C/C++ เพื่อจำลองการทำงานของระบบแล้วได้ผลการทดลองตามที่ออกแบบไว้ ได้ฟังก์ชันการทำงานหลัก 2 ฟังก์ชัน ดังนี้

ฟังก์ชันที่ 1 ฟังก์ชันการตั้งเวลาให้อาหารอัตโนมัติ แบ่งออกเป็น 3 ฟังก์ชันย่อย ได้แก่

1. สามารถตั้งเวลาในการให้อาหารได้ ถ้าหากหมดเวลาและอาหารยังมีอยู่จะเพิ่มเวลาถัดไปอัตโนมัติ
2. ถ้าเวลาที่ตั้งไว้นับถอยหลังจนหมดเวลา อาหารจะหล่นออกมาอัตโนมัติ
3. มีเสียงแจ้งเตือนขณะที่อาหารกำลังปล่อย

ฟังก์ชันที่ 2 ฟังก์ชันการให้อาหารโดยผู้ใช้ จากระยะไกลผ่าน NETPIE ในกรณีอาหารหมดก่อนเวลาที่ตั้งไว้ แบ่งออกเป็น 2 ฟังก์ชันย่อย ได้แก่

- 1.) สามารถกดปุ่มที่ Dash board ของ NETPIE เพื่อสั่งการให้อาหารจะหล่นออกมา
- 2.) สามารถรับการแจ้งเตือนของสถานะอาหารว่าหมดหรือไม่หมดผ่านแถบ Shadow

11. รายละเอียดการแบ่งงานในกลุ่ม

รหัสสนិត	ชื่อ-สกุล	หน้าที่ที่ได้รับมอบ
64366690	นายสุรเชษฐ์ อินทมงคล	เขียนโค้ดส่วนของบอร์ด esp8266 Node MCU และต่อวงจร
64367345	นายเอกธนา ทองเมือง	เขียนโค้ดส่วนของบอร์ด esp8266 Node MCU และต่อวงจร
64363651	นางสาวบุญสิตา เมฆแจ้ง	ออกแบบการทำงานของระบบ ทำรายงานและไฟล์นำเสนอ
64363880	นางสาวประกายกานต์ ฤทธิพิศ	เขียนโค้ด MQTT ส่วนของ NETPIE และ dashboard
64364658	นางสาวภัทรกร วิจิกลาง	ตัดต่อวิดีโอนำเสนอ ทำโมเดลจำลองระบบ
64366621	นางสาวสุพิชชา พรหมอยู่	ออกแบบการทำงานของระบบ ทำรายงานและไฟล์นำเสนอ

วิดีโอการนำเสนอ



https://youtu.be/2KYJCz_IPLE?si=0KTxwyG-EWU4H2xS