



YBR-Tawan

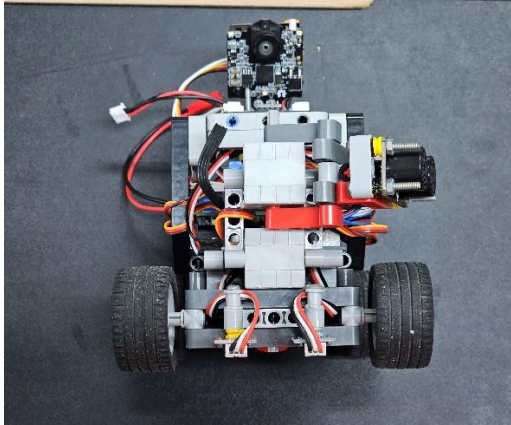
WRO Future Engineer
Documentation
(Thai-version)

เอกสารนี้แบ่งออกเป็น

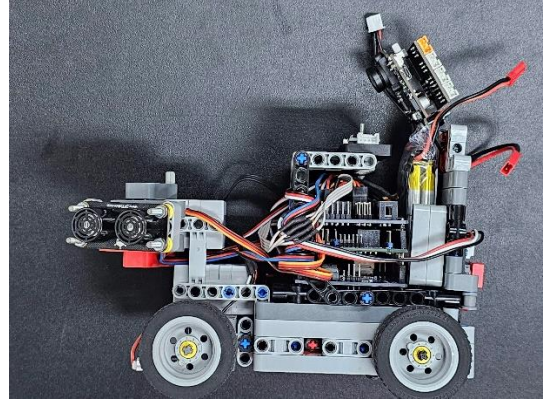
1. รูปภาพทีมงานและรถ
2. ข้อมูลทางวิศวกรรม
 - 2.1 การจัดการการเคลื่อนไหว
 - 2.2 การจัดการพลังงานและการตรวจเช็ค
 - 2.3 การจัดการอุปสรรค
 - 2.4 ปัจจัยทางวิศวกรรม
3. Youtube Link
4. Github link

ຮູບຖານຖິ້ມ

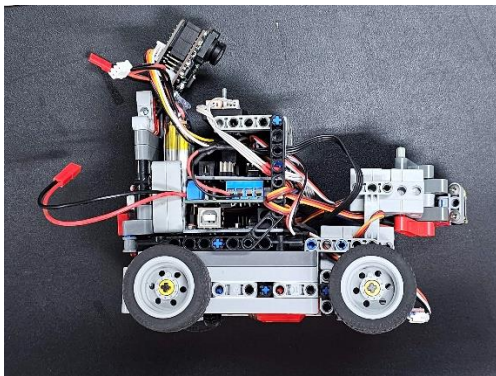
ຫນ້າ



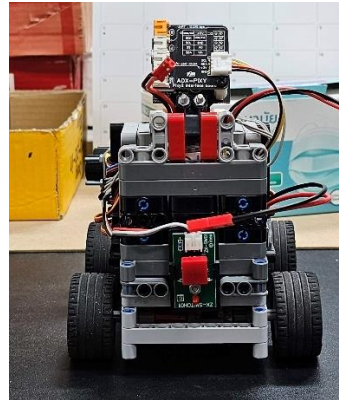
ຮ້າຍ



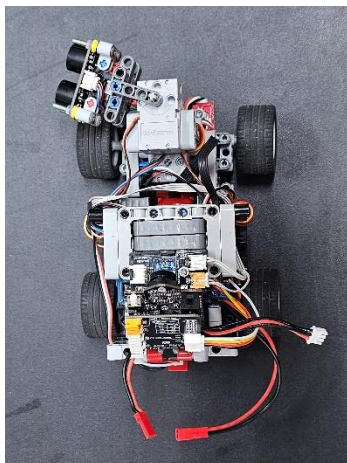
ບວກ



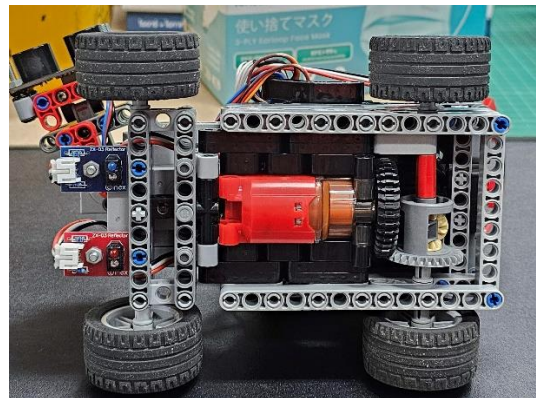
ຫລັງ



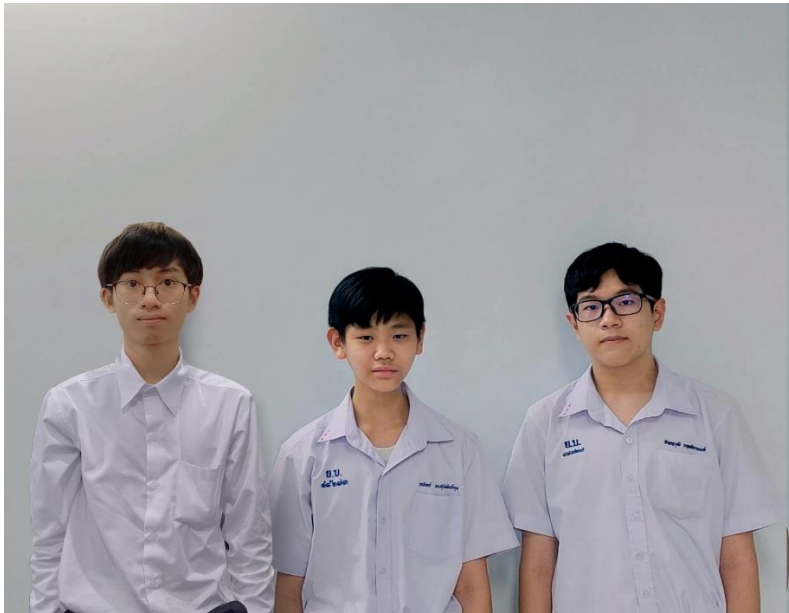
ບນ



ລ່າງ



ຮູບຖາວັນ



2. ข้อมูลทางวิศวกรรม

2.1 การจัดการการเคลื่อนไหว

เราใช้ Power Functions L-Motor ใน
การขับเคลื่อนของหุ่น

เป็นมอเตอร์ที่ง่ายและ เราเลือกมอเตอร์นี้เนื่องจาก
ความง่ายต่อการเชื่อมต่อมอเตอร์เข้ากับหุ่นยนต์ของเรา และมอเตอร์นี้ก็มีราคาที่ถูกรอีกด้วย

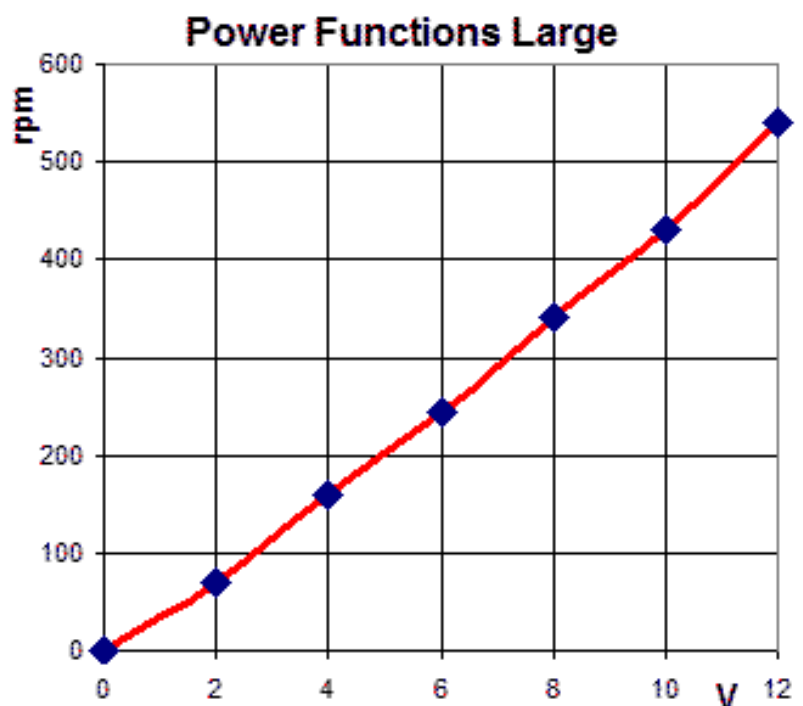
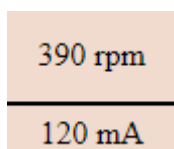


ข้อมูลทางเทคนิค

น้ำหนัก:



ความเร็วสูงสุด และ ปริมาณ
กระแสไฟฟ้า



ความเร็ว เทียบกับ volt

เราใช้ Geekservo 2kg 360 Degrees ในการเลี้ยวของหุ่น และใช้หมุน Ultrasonic Sensor

เป็น servo ที่ใช้ได้กับ LEGO ใช้งานและสะดวกต่อการ
สร้างหุ่น



ข้อมูลทางเทคนิค

- ความเร็วในการหมุนแกน 60 องศา/0.12 วินาที
- แรงบิด 2 กิโลกรัม.เซนติเมตร ที่ไฟเลี้ยง 4.8V
- กระแสไฟฟ้าขณะไม่มีโหลด 20 ถึง 50mA
- กระแสไฟฟ้าขณะบังคับแกนหมุน (Stall) 600 ถึง 700mA
- กระแสไฟฟ้าสงบขณะหยุดทำงาน 7mA
- ความกว้างของสัญญาณพัลส์ที่ต้องการ 0.6 ถึง 2.4 มิลลิวินาที
- หมุนได้ 0 ถึง 360 องศา

Code ควบคุม servo

```
void steering_servo(int degree) {  
    servo1.write((90 + max(min(degree, 45), -45)) / 2);  
}
```

2.2 การจัดการพลังงานและการตรวจเช็ค

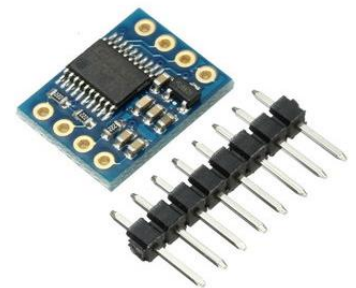
เราใช้ ZX-03 Reflector ในการตรวจค่าสีบนสนาม 2 ตัว (แดง และ น้ำเงิน)

Light Reflector นี้เป็น sensor ที่ใช้วัดค่าแสงสะท้อน และในสนามนี้ เราใช้ sensor นี้เพื่อที่จะ check เส้นตอนที่หุ่นเลี้ยว



เราใช้ sensor GY-25 เพื่อกำหนดทิศทางของหุ่นยนต์ของเราบนสนาม

เนื่องจากหุ่นจำเป็นต้องเดินเป็นเส้นตรงเป็นส่วนใหญ่ เราจึงเลือกใช้ sensor gyro เพื่อให้หุ่นรู้ทิศทางตัวมันเอง



Dataset: http://mkpochtoi.ru/GY25_MANUAL_EN.pdf

Library: <https://github.com/ElectronicCats/mpu6050>

เราใช้ Pixy 2.1 เพื่อตรวจจับสิ่งกีดขวางที่อยู่บนสนาม

Module กล้องตัวนี้มาพร้อม library และ function ที่ผู้พัฒนาของ pixy camera ได้ทำไว้แล้ว จึงง่ายต่อการใช้งาน



Documentaion:

<https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:start>

Software and Library: <https://pixycam.com/downloads-pixy2/>

เราใช้ Ultrasonic Sensor (SEN0307) เพื่อวัดระยะห่าง
ระหว่างหุ่นยนต์กับผนัง

Sensor ตัวนี้ใช้เอาต์พุตแรงดันไฟฟ้าแบบอะนาล็อก
และให้การวัดระยะทางที่แม่นยำภายใน 2-500 ซม. ด้วย
ความละเอียด 1 ซม. และความแม่นยำ $\pm 1\%$ ซ้ำเหมาะกับการแข่งขันนี้มาก



เราใช้ Arduino UNO สมองสำหรับ
หุ่นยนต์ของเรา



เราใช้ แบตเตอรี่ลิโพอ Helicox
2200mah (7.4V) เพื่อให้
พลังงานหุ่นของเรา



การเดินสาย

// Motor

ENB -> Arduino UNO Pin 11

INB -> Arduino UNO Pin 13

// Servos

STEER_SRV -> Arduino UNO Pin 9

ULTRA_SRV -> Arduino UNO Pin 8

// Ultrasonic Sensor

ULTRA_DATA_PIN -> Arduino UNO Pin 2

// Light Sensors

RED_DATA_SEN -> Arduino UNO Pin 0

BLUE_DATA_SEN -> Arduino UNO Pin 1

// Button

BUTTON_DATA -> Arduino UNO Pin 3

// Pixy Camera

PIXY_SDA -> Arduino UNO Pin 4

PIXY_SCL -> Arduino UNO Pin 5

// Gryo

TX -> Arduino UNO RX

RX -> Arduino UNO TX

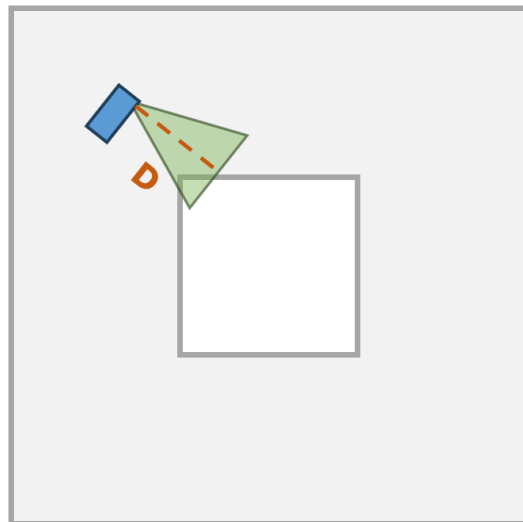
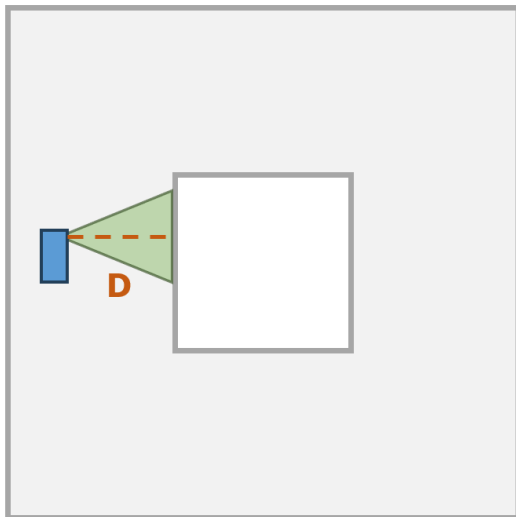
// Battery

BATTERY -> PWR_IN of Arduino UNO

2.3 การจัดการอุปสรรค

2.3.1 รอบคัดเลือก

ในรอบคัดเลือกหุ่นยนต์จะใช้ Ultrasonic sensor เพื่อคำนวณระยะทางระหว่างกำแพงของสนามกับหุ่น



หุ่นจะนำระยะทางของหุ่นจากกำแพงและ องศาของ sensor gyro มาคำนวณ เป็น steering degree (องศาการเลี้ยว) เพื่อให้หุ่นคงระยะห่างระหว่างกำแพง ได้ด้วยสูตร **Proportional Integral Derivative (PID)**

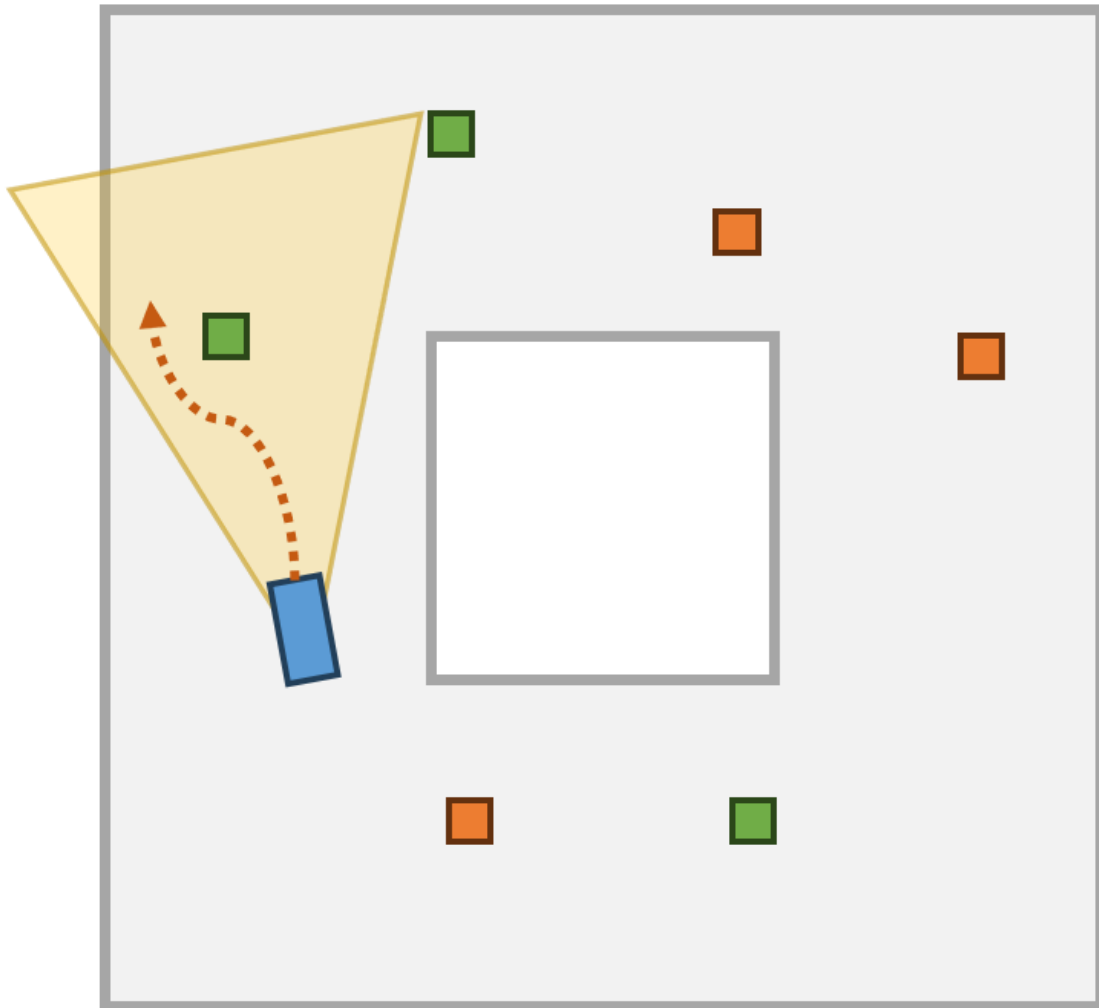
$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de}{dt}$$

สูตรการเดิน

```
getIMU();  
line_detection();  
ultra_servo(pvYaw, TURN);  
int wall_distance = getDistance();  
motor_and_steer(-1 * compassPID.Run(pvYaw + ((wall_distance - 25) * 1) *  
((float(TURN == 'R') - 0.5) * 2)));
```

2.3.2 รอบชิง

ในรอบคัดเลือกหุ่นยนต์จะใช้ Pixy camera Ultrasonic sensor และ Gyro เพื่อนำมาคำนวณ steering degree (องศาการเลี้ยว)



หุ่นยังคงใช้ PID เหมือนกับรอบคัดเลือก แต่ตัวแปรที่เพิ่มขึ้นมาคือ avoidance degree (องศาในการเลี้ยวหลบ) โดย code จะเป็นดังต่อไปนี้

สูตรคำนวณองศาในการเลี้ยวหลบ

```
float calculate_avoidance() {
    int blocks = pixy.ccc.getBlocks();

    found_block = false;

    if (blocks) {
        int signature = -1;        // Signature of the object you want to detect
        int targetHeight = 10;    // Height of the object in centimeters
        float focallength = 2.3;  // Focal length of the camera in centimeters
        float cameraFOV = 80.0;   // Field of view of the camera in degrees

        int largestBlockIndex = -1;
        int largestBlockArea = 0;

        for (int i = 0; i < blocks; i++) {
            if (pixy.ccc.blocks[i].m_height > 1.33 *
float(pixy.ccc.blocks[i].m_width)) {
                int objectArea = pixy.ccc.blocks[i].m_width;
                // * pixy.ccc.blocks[i].m_height;
                found_block = true;
                if (objectArea > largestBlockArea) {
                    largestBlockIndex = i;
                    largestBlockArea = objectArea;
                    signature = pixy.ccc.blocks[i].m_signature;
                }
            }
        }

        if (signature != -1) {
            int objectHeight = pixy.ccc.blocks[largestBlockIndex].m_height;
            float distance = (targetHeight * focallength * 100) / objectHeight;

            float blockCenterX = pixy.ccc.blocks[largestBlockIndex].m_x;
            float blockCenterY = pixy.ccc.blocks[largestBlockIndex].m_y;

            float deltaX = blockCenterX - pixy.frameWidth / 2;
            float deltaY = blockCenterY - pixy.frameHeight / 2;

            float detected_degree = deltaX * 40 / pixy.frameWidth;

            float blockPositionX = distance *
sin(degreesToRadians(detected_degree));
            float blockPositionY = distance * cos(degreesToRadians(detected_degree))
- 17;

            if (signature == 1) {
```

```

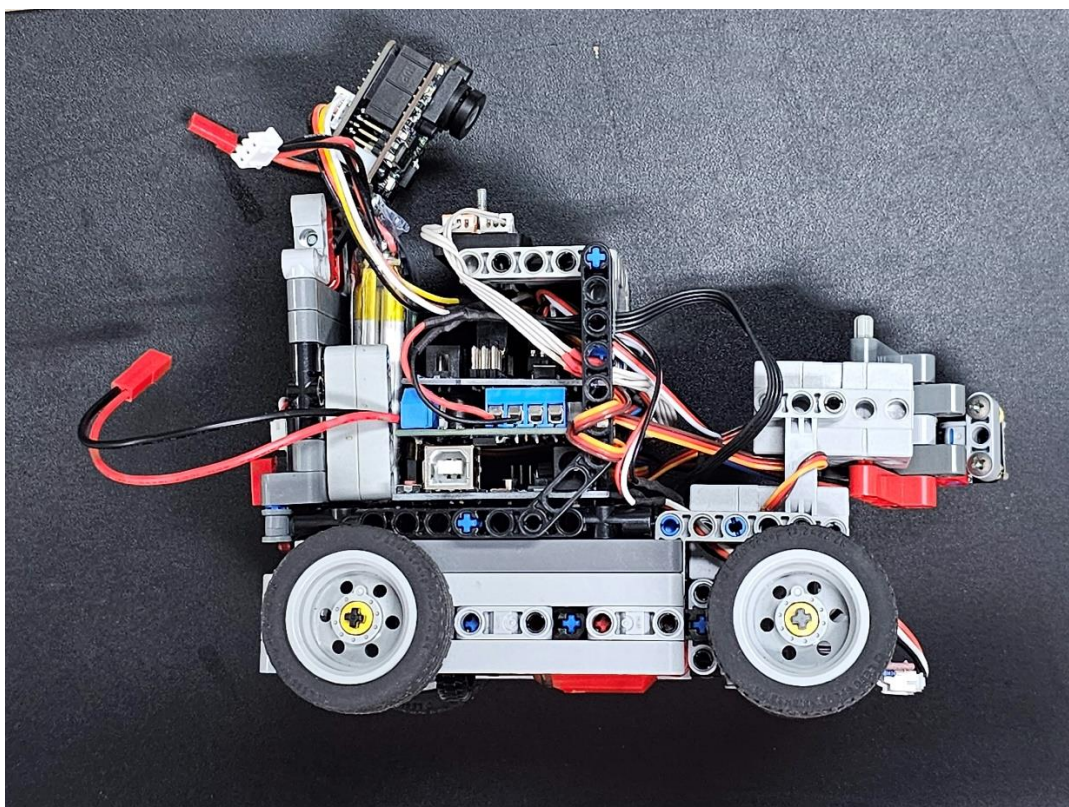
        avoidance_degree = max(radiansToDegree(atan2(blockPositionX + 9,
blockPositionY)), 5);
        Blocks_TURN = 'R';
    } else {
        avoidance_degree = min(radiansToDegree(atan2(blockPositionX - 9,
blockPositionY)), -5);
        Blocks_TURN = 'L';
    }
}
}

return avoidance_degree;
}

```

2.4 ปัจจัยทางวิศวกรรม

หุ่นที่เราสร้างขึ้นมาเป็นหุ่นที่ออกแบบมาเพื่อใช้สำหรับงานนี้โดยเฉพาะ โดยใช้ชิ้นส่วน LEGO และน็อตต่างๆ ในการยึดชิ้นส่วนของหุ่นเข้าด้วยกัน



ปัญหาของหุ่นที่เราพบ

1. แรงบิดของมอเตอร์

มอเตอร์ตัวนี้มีแรงบิดอยู่ที่ 18 N.cm ซึ่งเราพบว่าหาก รับมอเตอร์โดยใช้ แบตเตอรี่ลิโ 7.4V ที่ power **ประมาณ 20-30%** จะสามารถทำให้ หุ่นยนต์เริ่มเคลื่อนที่ได้

2. แบตเตอรี่ลิโ

เนื่องจาก Arduino UNO ไม่มี indicator ที่ใช้เตือนว่าแบตเตอรี่หมด เราจึงต้องวัด แบตเตอรี่อยู่ตลอด แบตเตอรี่ **ไม่ควรมีประจุต่ำกว่า 7.7V** เพราะค่าของ Program อาจจะผิดพลาดได้

3. Gyro Drift

หมายถึงปรากฏการณ์ที่เอาต์พุตของไจโรสโคปค่อยๆ เบี่ยงเบนไปจาก ค่าที่คาดไว้เมื่อเวลาผ่านไป เพราะฉะนั้นตอนเปิดหุ่นของเรา **จำเป็นต้อง วางหุ่นไว้ที่พื้นสนามก่อน และจึงจะค่อยๆ เสียบแบตเตอรี่ได้**

4. Loose Power Connection

บางครั้งหุ่นจะ reset ตัวเอง เหตุการณ์นี้เกิดจากสายที่เสียบเข้ากับ Arduino UNO อาจจะหลวม หรือไม่ได้บัดกรีสายก่อน ทำให้ไม่สามารถส่ง ไฟฟ้า ไปที่ Pixy camera, Ultrasonic sensor, Light sensor 2 ตัว, Gyro sensor, servo และ motor ได้อย่างทั่วถึง

3. YouTube Video

Qualification Round (ทำภารกิจสำเร็จประมาณ 37 วินาที)

<https://youtu.be/zxIOy2REIk4>

Final Round (ทำภารกิจสำเร็จประมาณ 45 วินาที)

<https://youtu.be/RNagUxmCluk>

4. GitHub link

<https://github.com/natapol2547/YBR-Tawan>

YB Robot Club **[YBR-Tawan]**