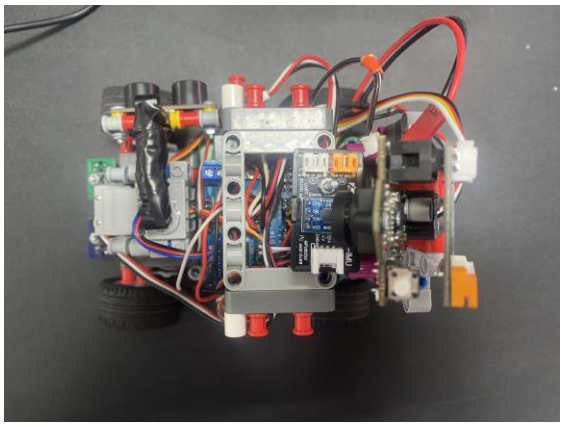
WRO Future Engineer 2023

By Yothinburana School Robot Club

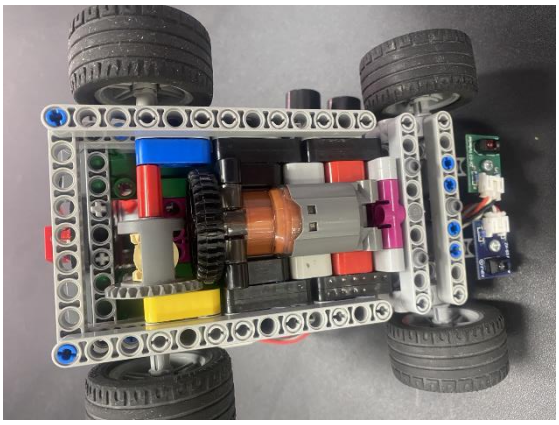
สารบัญ

รูปภาพหุ่นยนต์

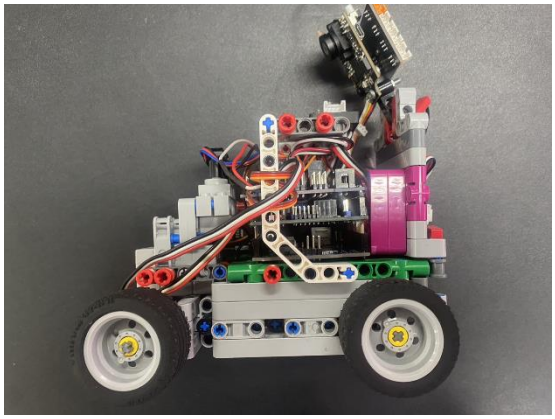
บน



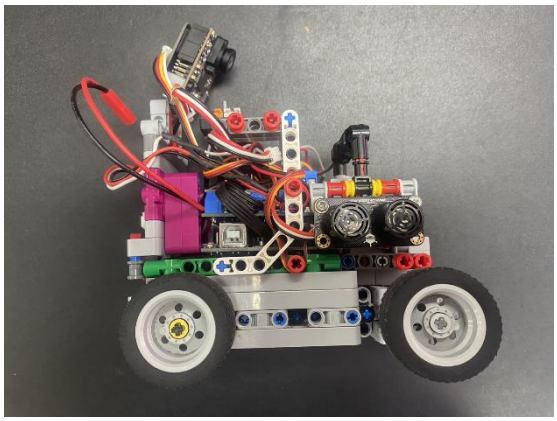
ล่าง



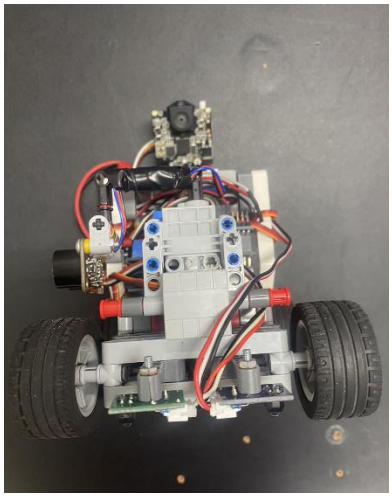
ซ้าย



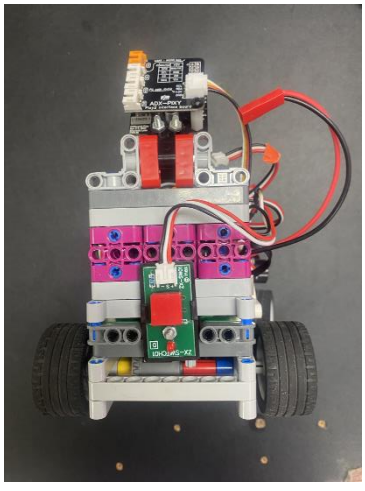
ขวา



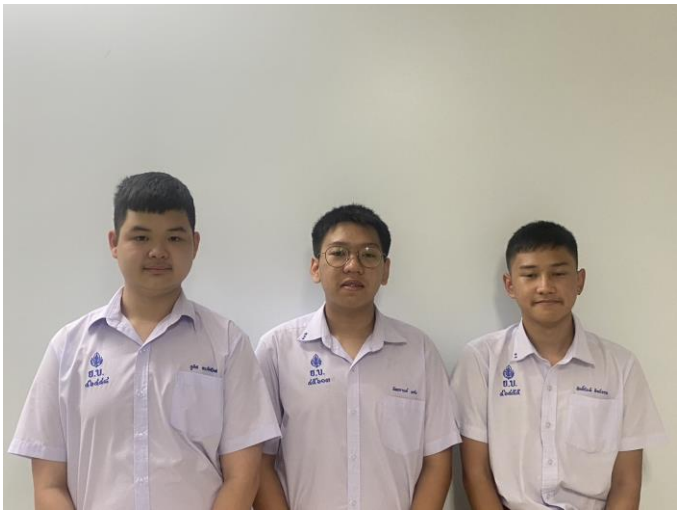
หน้า



หลัง



รูปภาพทีม



ข้อมูลทางวิศวกรรม

2.1 การจัดการการเคลื่อนไหว

Power Function L-Motor

ในการขับเคลื่อนของหุ่นเป็นมอเตอร์ที่ใช้ง่าย

และสามารถเอามาดัดแปลงเพื่อเชื่อมต่อกับหุ่นยนต์ได้ง่าย

ซึ่งมอเตอร์ยังมีความเร็วที่สูงและมีประสิทธิภาพมาก

ลักษณะของอุปกรณ์

น้ำหนัก:

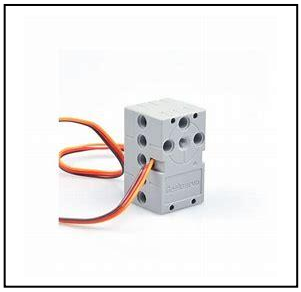
PF Large
42g

ความเร็วสูงสุดและ

ปริมาณกระแสไฟฟ้า:

390rpm
120mA

Geekservo 2kg 360 Degrees



ในการเลี้ยวของหุ่นยนต์และใช้หมุน Ultrasonic Sensor

เป็น Servo ที่ใช้กับ LEGO ใช้ง่ายและสะดวกต่อการสร้างหุ่นยนต์

ลักษณะของอุปกรณ์

- ความเร็วในการหมุนแกน 60องศา/0.12วินาที
- แรงบิด 2 กิโลกรัมเซนติเมตร ที่ไฟเลี้ยง 4.8 V
- กระแสไฟฟ้าขณะบังคับแกนหมุน (Stall) 600ถึง 700 mA
- กระแสไฟฟ้าสงบขณะหยุดการทำงาน 7mA
- ความกว้างของสัญญาณพัลส์ที่ต้องการ 0.6 ถึง 2.4 มิลลิวินาที
- หมุนได้ 0 ถึง 360 องศา

Code การหมุน servo เพื่อให้ไปในทิศทางที่ในต้องการ

```
void steering_servo(int degree) {
  myservo2.write((90 + max(min(degree, 45), -45)) / 2);
}
```

2.2 การจัดการพลังงานและการตรวจเช็ค

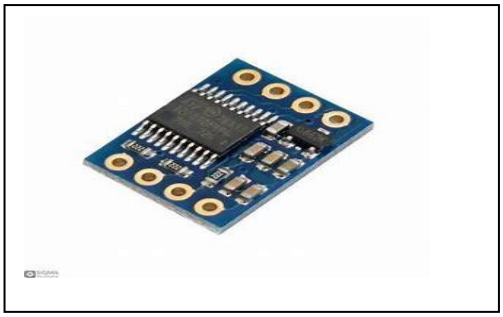
ZX-03 Reflector ในการตรวจค่าสีบนสนาม 2 ตัว

Light Reflector เป็น sensor

ที่ใช้วัดค่าแสงเงาสะท้อนในสนาม

เพื่อตรวจสอบเส้นในสนามสำหรับหุ่นยนต์เวลาเลี้ยว

sensor GY-25 (Sensor GYRO)



ในการกำหนดทิศทางของหุ่นยนต์เพื่อให้หุ่นยนต์เดินเป็นเส้นตรงและสามารถเลี้ยวไปในทิศทางที่กำหนด

Dataset : http://mkpochtoi.ru/GY25_Manual_EN.pdf

Library: http://github.com/ElectronicCats/mpu6050

Pixy2.1

ใช้ในการตรวจจับสิ่งกีดขวางและสีของสิ่งกีดขวางบนสนามเพื่อให้สามารถหลบไปในทิศทางที่ถูกต้องได้

Module เป็นกล้องที่มาพร้อมกับ **Library** และ **Function**

Documentation :

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

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

Ultrasonic Sensor (SEN0307)

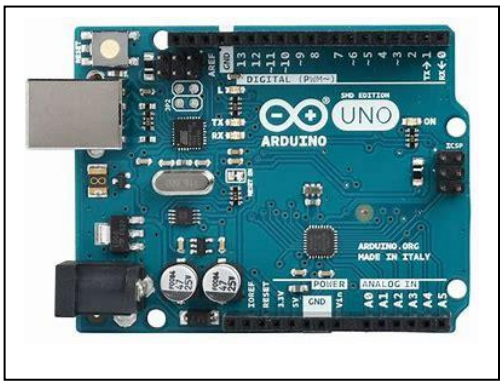
เพื่อวัดระยะห่างของหุ่นยนต์กับผนังทำให้สามารถ

ควบคุมหุ่นยนต์ให้อยู่ในจุดของสนามที่เราต้องการได้

Sensor ตัวนี้มีการวัดระยะทางที่แม่นยำอยู่ ที่ **+-1%**

โดยมีความละเอียดเท่ากับ **1** ซม**.**

Arduino UNO



เป็นเสมือนแกนกลางสำคัญสำหรับหุ่นยนต์

ที่สามารถเอาไว้เชื่อมต่ออุปกรณ์ต่าง ๆเอาไว้ด้วยกัน

แบตเตอรี่ลิโพ Helicox 2200mah(7.4V)

เป็นเสมือนตัวให้พลังงานหุ่นยนต์เพื่อให้สามารถ

ทำงานได้

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_PIN -> Arduino UNO Pin 2

// Light Sensors

RED _SEN -> Arduino UNO Pin 5

BLUE _SEN -> Arduino UNO Pin 1

// Button

BUTTON -> 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

หุ่นยนต์จะใช้ Ultrasonic sensor เพื่อคำนวณหาระยะทางระหว่างกำแพงกับหุ่นเพื่อที่จะนำระยะทางของหุ่นจากกำแพงและ องศาของ sensor gyro มาหลังจากนั้นนำมาคำนวณเป็น steering degree (องศาการเลี้ยว) เพื่อให้หุ่นคงระยะห่างระหว่างกำแพงได้ด้วยสูตร **Proportional Integral Derivative (PID)**

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

สูตรการเดิน

```
    getIMU();
    line_detection();
    ultra_servo(pvYaw, TURN);
    int wall_distance = getDistance();
    motor_and_steer(-1 * compassPID.Run(pvYaw + ((wall_distance - 35 * 1) *
((float(TURN == 'R') - 0.5) * 2)));
```

2.3.2 รอบชิง

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

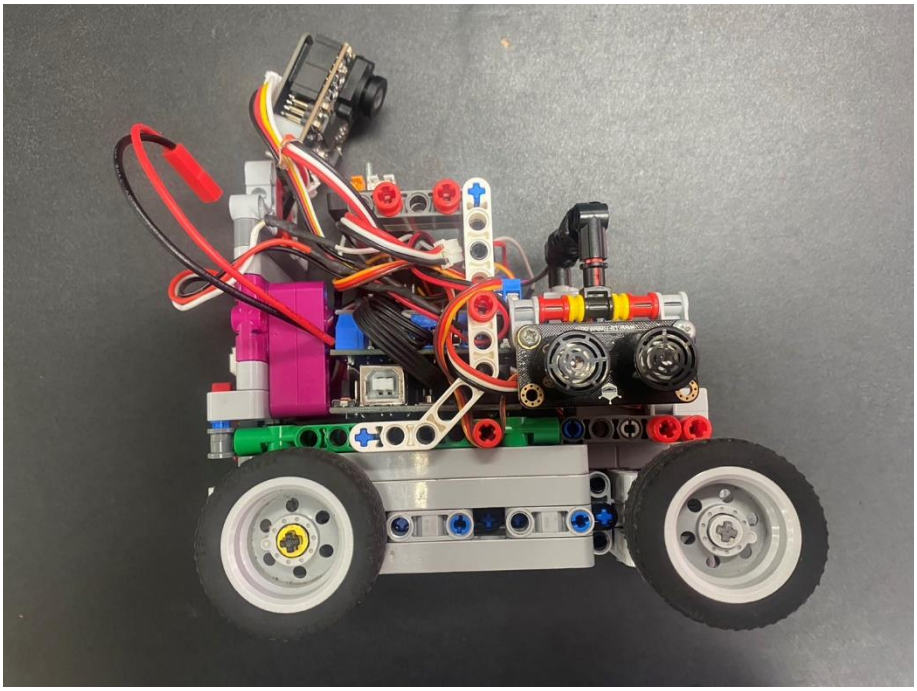
หุ่นยนต์ยังคงใช้ PID เหมือนกับรอบคัดเลือก แต่จะมีตัวแปลที่เพิ่มขึ้นมาคือ avoidance degree (องศาในการเลี้ยวหลบ) โดย code จะเป็นดังต่อไปนี้

สูตรคำนวนองศาในการเลี้ยวหลบ

```
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
   มอเตอร์ตัวนี้มีแรงบิดอยู่ที่ประมาณ 18 N.cm ซึ่งทำให้เราพบว่า หากเรารันมอเตอร์โดยใช้ แบตเตอรี่ลิโพ 7.4V ที่ power **ประมาณ 20-30%** จะสามารถทำให้หุ่นยนต์เริ่มเคลื่อนที่ได้

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

3. Gyro Drift
   ปรากฏการณ์ที่เอาต์พุตของไจโรสโคปค่อยๆ เบี่ยงเบนไปจากค่าที่คาดไว้เมื่อเวลาผ่านไป เพราะฉะนั้นตอนเปิดหุ่นของเรา **จำเป็นต้องวางหุ่นไว้ที่พื้นสนามก่อน รอให้หุ่นนิ่งอยู่กับที่ก่อนและจึงจะค่อยๆ เสียบแบตเตอรี่ได้**

4. Loose Power Connection
   บางครั้งหุ่นจะ reset ตัวเอง เกิดจากสายที่เสียบเข้าที่ Arduino UNO อาจจะหลวม หรือบัดกรีสายไม่ดี ทำให้ม่สามารถแจกจ่ายไฟฟ้าอย่างเพียงพอสำหรับ Pixy camera, Ultrasonic sensor, Light sensor 2 ตัว, Gyro sensor, servo และ motor ได้อย่างทั่วถึง

3. YouTube Video

Qualification Round (ทำภารกิจสำเร็จประมาณ 31 วินาที)

https://youtu.be/7kWcytFyiAU

4. GitHub link

https://github.com/PattakanA/YBR-FMS
