YBR-Tawan

WRO Future Engineer

Documentation

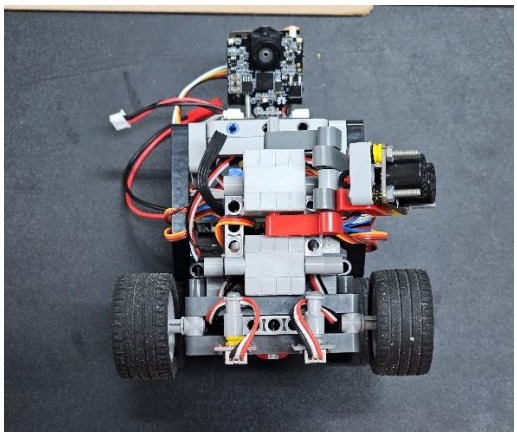
(Thai-version)

เอกสารนี้แบ่งออกเป็น

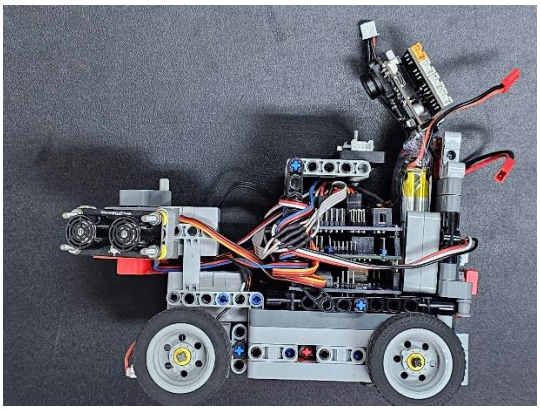
1. รูปภาพทีมงานและรถ
2. ข้อมูลทางวิศวกรรม
    2.1 การจัดการการเคลื่อนไหว
    2.2 การจัดการพลังงานและการตรวจเช็ค
    2.3 การจัดการอุปสรรค
    2.4 ปัจจัยทางวิศวกรรม
3. Youtube Link

รูปภาพหุ่น

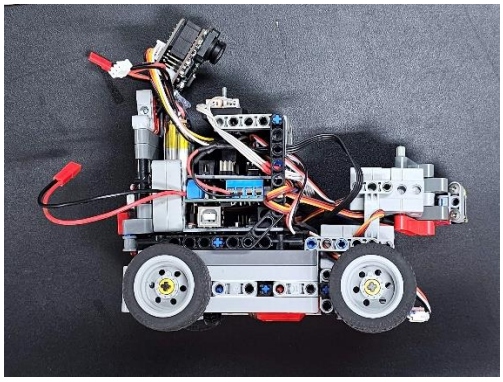
หน้า



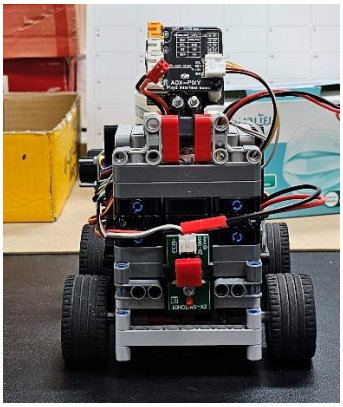
ซ้าย



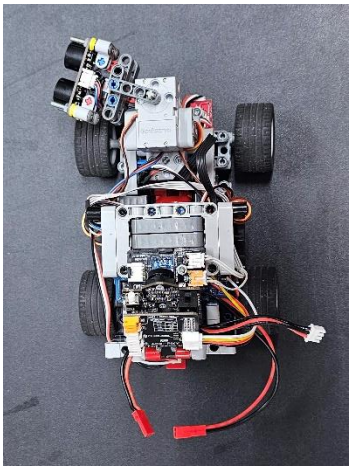
ขวา



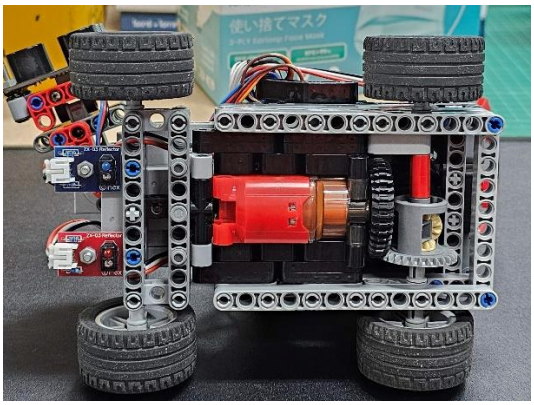
หลัง



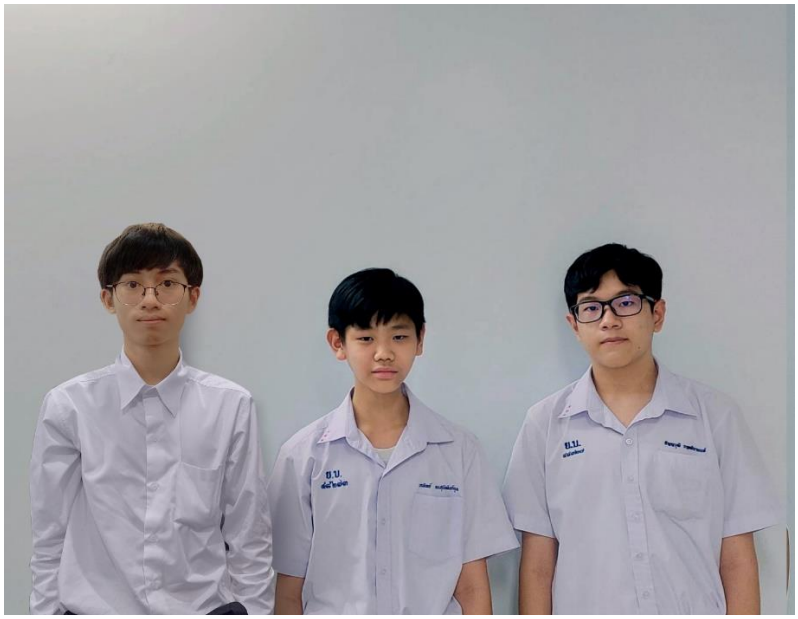
บน

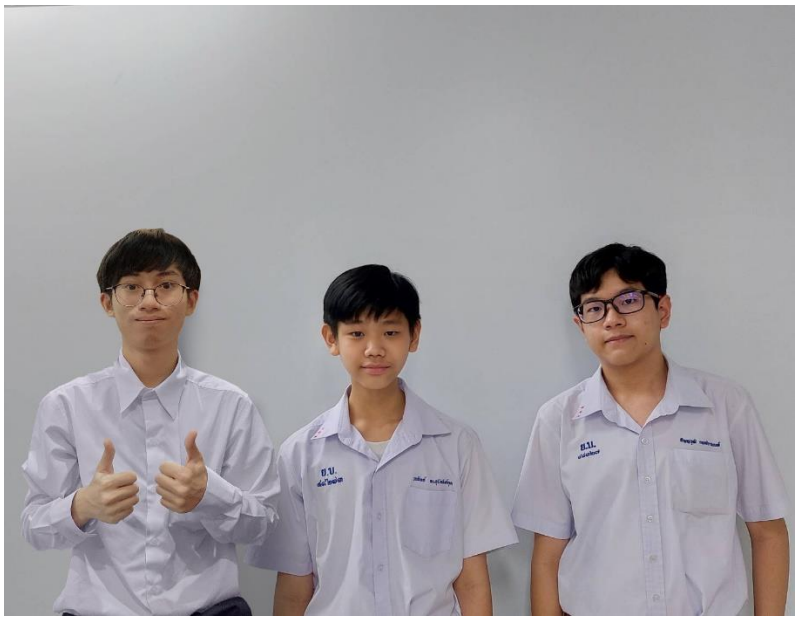


ล่าง



รูปภาพทีมงาน





2. ข้อมูลทางวิศวกรรม

2.1 การจัดการการเคลื่อนไหว

เราใช้ Power Functions L-Motor ในการขับเคลื่อนของหุ่น

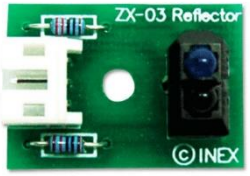
เป็นมอเตอร์ที่ง่ายและ เราเลือกมอเตอร์นี้เนื่องจากความง่ายต่อการเชื่อมต่อมอเตอร์เข้ากับหุ่นยนต์ของเรา และมอเตอร์นี้ก็มีราคาที่ถูกอีกด้วย

เราใช้ Geekservo 2kg 360 Degrees ในการเลี้ยวของหุ่น และใช้หมุน Ultrasonic Sensor

เป็น servo ที่ใช้ได้กับ LEGO ใช้ง่ายและสะดวกต่อการสร้างหุ่น

2.2 การจัดการพลังงานและการตรวจเช็ค

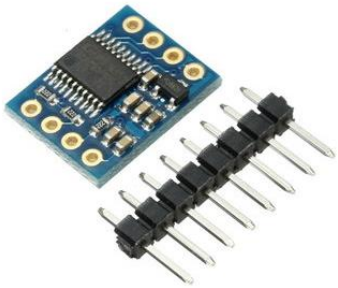
เราใช้ ZX-03 Reflector ในการตรวจค่าสีบนสนาม



Light Reflector นี้เป็น sensor ที่ใช้วัดค่าแสงสะท้อน และในสนามนี้ เราใช้ sensor นี้เพื่อที่จะ check เส้นตอนที่หุ่นเลี้ยว

เราใช้ sensor GY-25 เพื่อกำหนดทิศทางของหุ่นยนต์ของเราบนสนาม

เนื่องจากหุ่นจำเป็นต้องเดินเป็นเส้นตรงเป็นส่วนใหญ่ เราจึงเลือกใช้ sensor gyro เพื่อให้หุ่นรู้ทิศทางตัวมันเอง



เราใช้ Pixy 2.1 เพื่อตรวจจับสิ่งกีดขวางที่อยู่บนสนาม

Module กล้องตัวนี้มาพร้อม library และ function ที่ผู้พัฒนาของ pixy camera ได้ทำไว้แล้ว จึงง่ายต่อการใช้มาก

เราใช้ Ultrasonic Sensor (SEN0307) เพื่อวัดระยะห่างระหว่างหุ่นยนต์กับผนัง

Sensor ตัวนี้ใช้เอาต์พุตแรงดันไฟฟ้าแบบอะนาล็อก และให้การวัดระยะทางที่แม่นยำภายใน 2-500 ซม. ด้วยความละเอียด 1 ซม. และความแม่นยำ ±1% ซึ่มเหมาะกันการแข่งขันนี้มาก

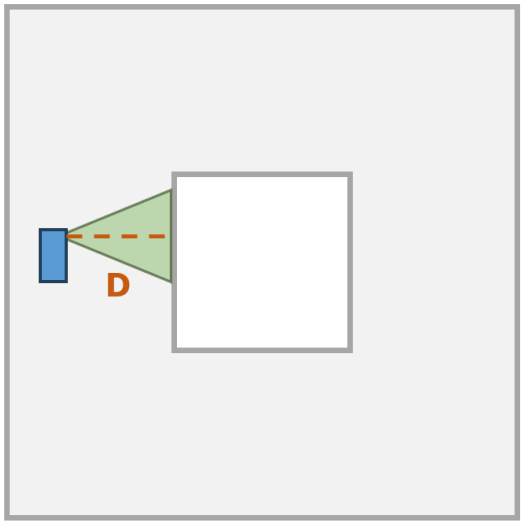
เราใช้ Arduino UNO สมองสำหรับหุ่นยนต์ของเรา

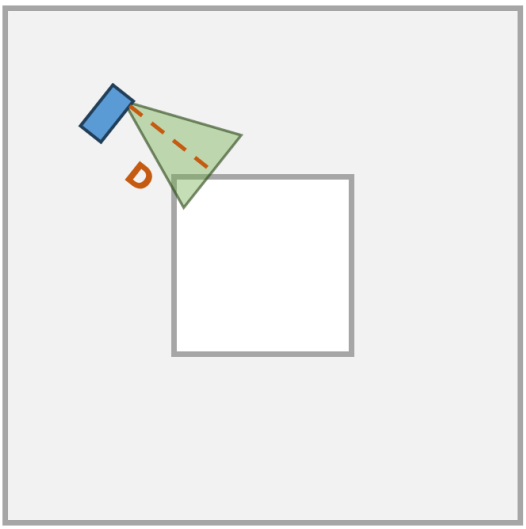
เราใช้ แบตเตอรี่ลิโพ Helicox 2200mah (7.4V) เพื่อให้พลังงานหุ่นของเรา

2.3 การจัดการอุปสรรค

2.3.1 รอบคัดเลือก

ในรอบคัดเลือกหุ่นยนต์จะใช้ Ultrasonic sensor เพื่อคำนวณระยะทางระหว่างกำแพงของสนามกับหุ่น





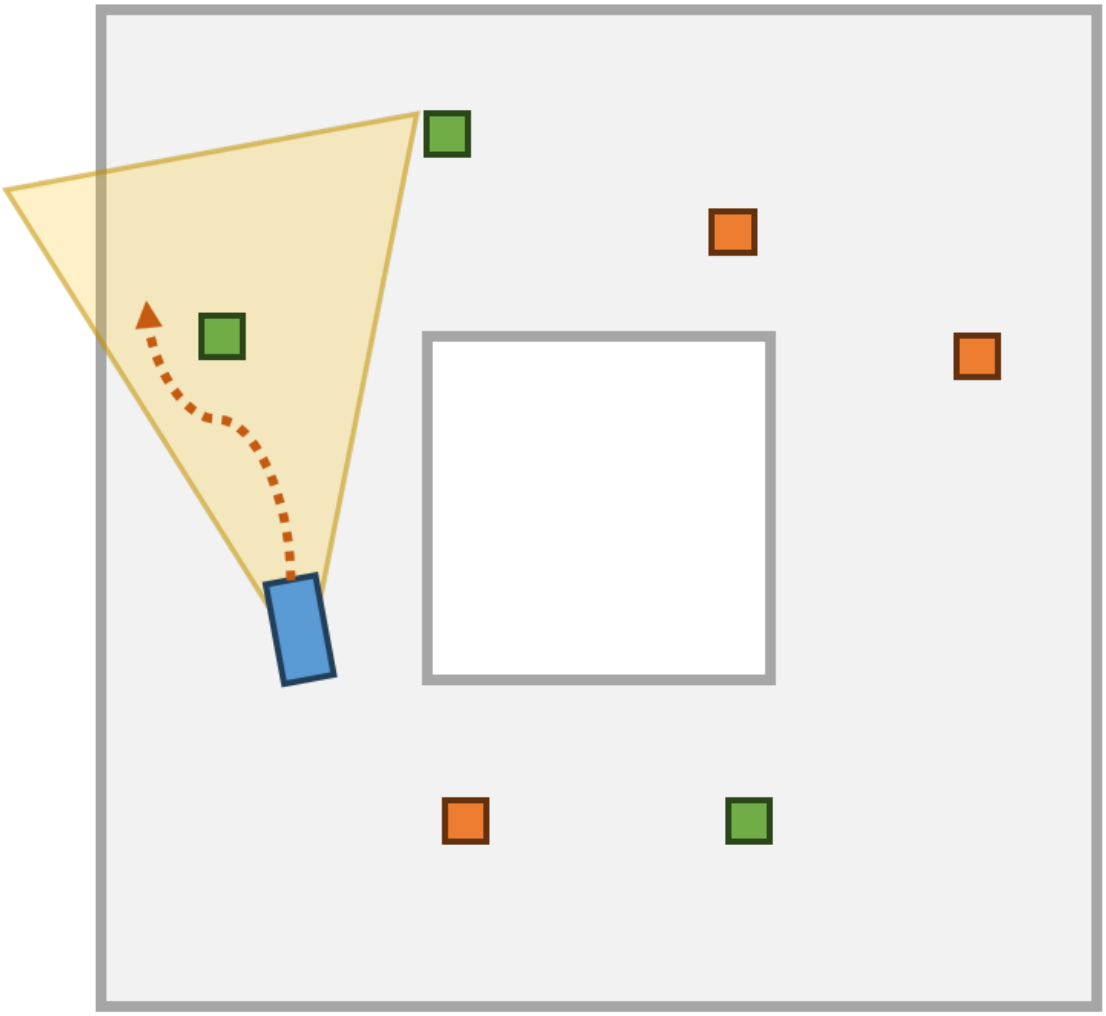
หุ่นจะนำระยะทางของหุ่นจากกำแพงและ องศาของ sensor gyro มาคำนวณเป็น steering degree (องศาการเลี้ยว) เพื่อให้หุ่นคงระยะห่างระหว่างกำแพงได้ด้วยสูตร Proportional Integral Derivative (PID)

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

```
    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 แต่ตัวแปลที่เพิ่มขึ้นมาคือ avoidance degree (องศาในการเลี้ยวหลบ) โดย code จะเป็นดังต่อไปนี้

```
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

เราต้องคำนึงถึงหลากหลายปัจจัยด้วยกัน

1. แรงบิดของมอเตอร์
   มอเตอร์ตัวนี้มีแรงบิดอยู่ที่ 18 N.cm ซึ่งเราพบว่าหาก รันมอเตอร์โดยใช้แบตเตอรี่ลิโพ 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
   หมายถึงปรากฏการณ์ที่เอาต์พุตของไจโรสโคปค่อยๆ เบี่ยงเบนไปจากค่าที่คาดไว้เมื่อเวลาผ่านไป เพราะฉะนั้นตอนเปิดหุ่นของเรา **จำเป็นต้องวางหุ่นไว้ที่พื้นสนามก่อน และจึงจะค่อยๆ เสียบแบตเตอรี่ได้**

4. Power Overload
   บางครั้งหุ่นจะ reset ตัวเองหารมีการใช้ power ที่มากไป เหตุการณ์นี้เกิดจากสายที่เสียบเข้าที่ 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/zxlOy2REIk4

Final Round (ทำภารกิจสำเร็จประมาณ 45 วินาที)

https://youtu.be/RNagUxmCluk
