Assignment Report

Robot Car Contest

กรณีศึกษาการทำหุ่นยนต์สำหรับเดินตามเส้นโดยใช้ Arduino Uno (Robot Car Contest)

Contributer

นายพลัฏฐ์ภาณุ เพ็ชรสมาน รหัสนักศึกษา 66010542

นายพิพรรธน์พงศ์ พันธุ์พฤกษ์ รหัสนักศึกษา 66010574

Introduction to Computer Engineering (ICE)

Semester 1 Year 1 | Bachelor of Computer Engineering

King Mongkut's Institute of Technology Ladkrabang, KMITL

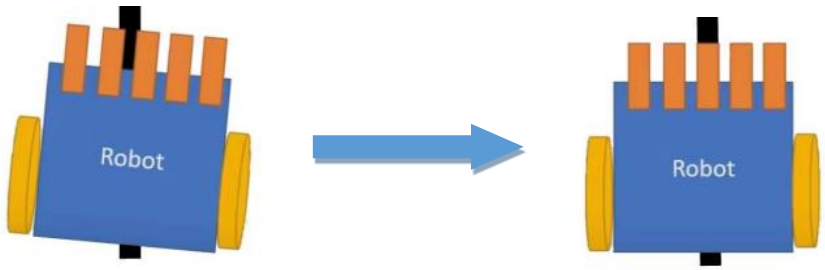
Table of Content

ROBOT CAR CONTEST – REPORT

การทำงานของ Robot โดยสังเขป

การทำงานของ Robot โดยสังเขป คือ โดยทั่วไป ตัว Robot จะพยายามเดินตามเส้นสีดำบนสนาม (ทั้งเส้นทึบและเส้นประ) โดยหากตัว Robot มีการเดินออกนอกทิศทาง (เกิด Error) ก็จะพยามยามหาวิธีกลับเข้าสู่เส้นทางด้วยตนเอง โดยจะเริ่มเดินที่จุดเริ่มต้น จนไปถึงจุดสิ้นสุด (แสดงดังภาพ)

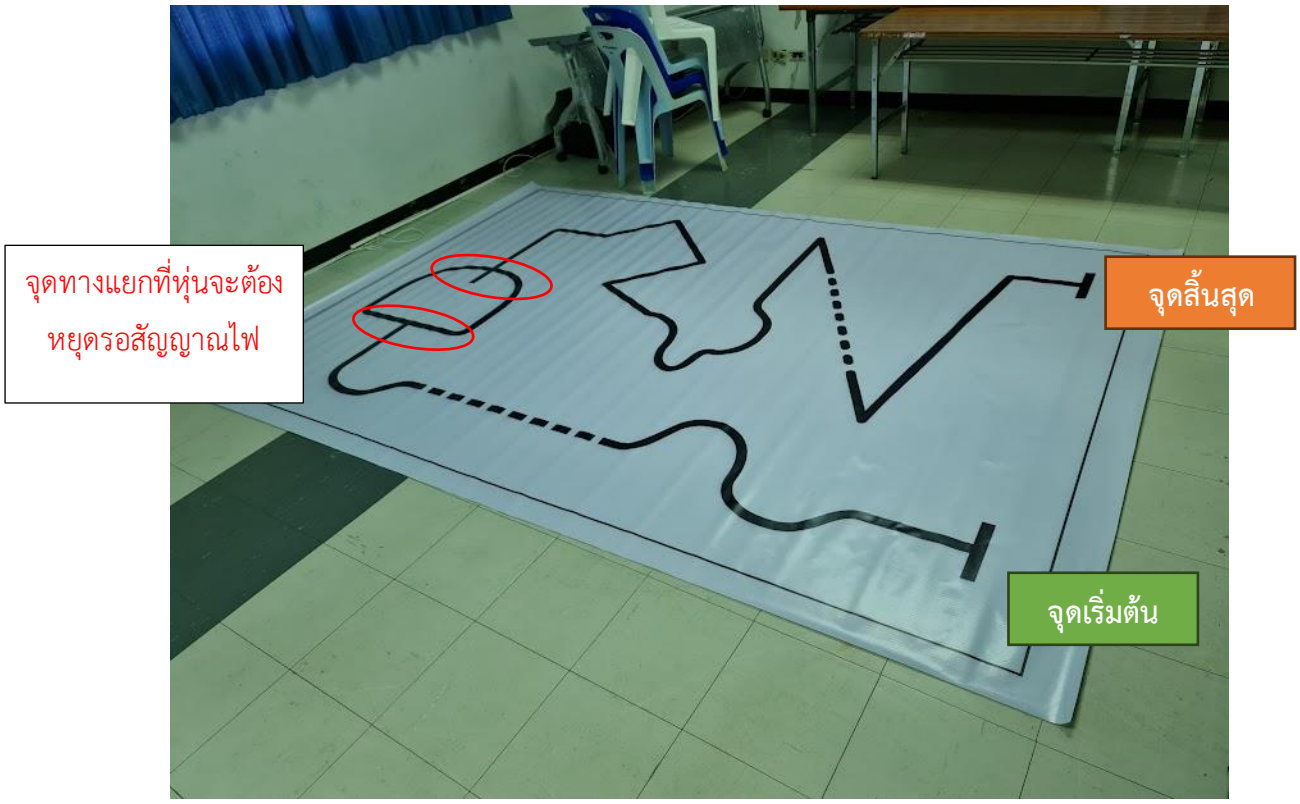
อีกหนึ่งฟีเจอร์ คือ เมื่อหุ่นเผชิญหน้ากับทางแยก จะทำการหยุดรอสัญญาณไฟ โดยเมื่อไฟกระพริบเป็นจำนวน 1 ครั้ง จะเลี้ยวไปทางซ้าย และเมื่อไฟกระพริบจำนวน 2 ครั้ง ก็จะเลี้ยวไปทางขวา (แสดงดังภาพ)



หุ่นมีอาการ Error คือ ทิศทางเบี้ยว ไปทางขวาของเส้น

โปรแกรมทำการแก้ไข ให้หุ่นกลับมาเดินตรงเส้นอีกครั้ง

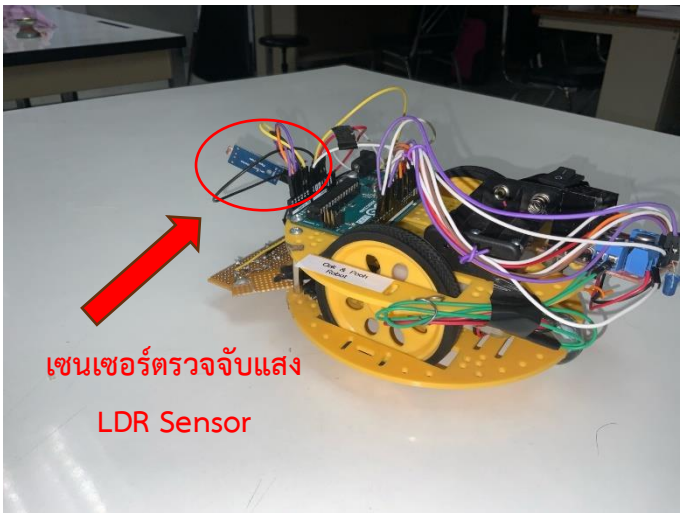
หมายเหตุ การทำงานโดยละเอียดของหุ่นโดยละเอียด จะอธิบายแยกแต่ละประเด็นต่อไป

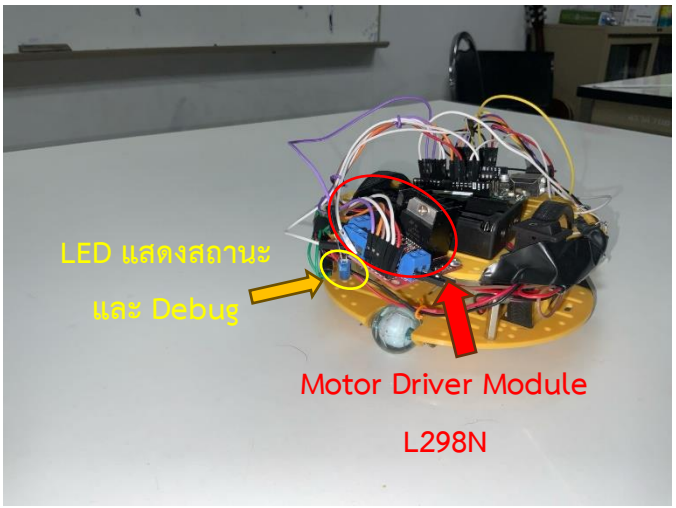


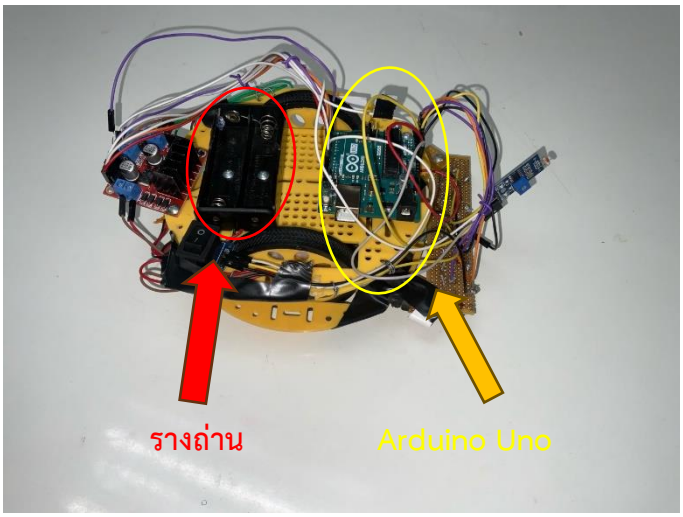
สนามที่ใช้ในการแข่งขัน

การออกแบบ Robot Car

ในการออกแบบตัว Robot ได้คำนึงถึงการจัดวางองค์ประกอบต่าง ๆ ของตัว Robot เช่น รางถ่าน (ประกอบมาให้อยู่แล้ว ไม่ได้แก้ไขเปลี่ยนแปลง), บอร์ด Arduino Uno, แผง IR Sensor, Motor Driver Module (L298N) ทั้งนี้ เพื่อให้เกิดความสมดุลของน้ำหนักของตัว Robot และเพื่อเพิ่ม Aero Dynamic ที่ดีของตัว Robot

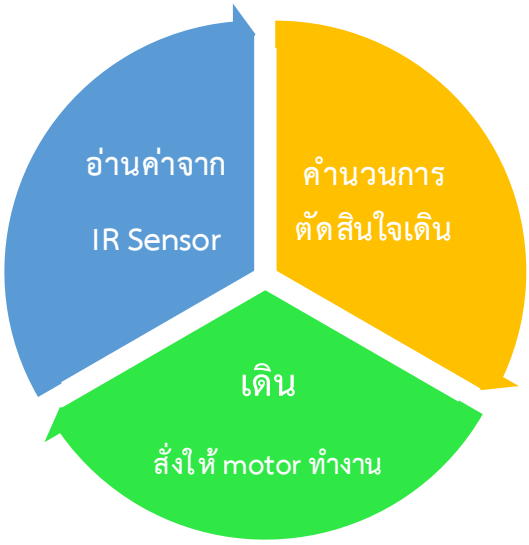






หุ่นยนต์ Robot Car

หลักการทำงานของ Robot Car

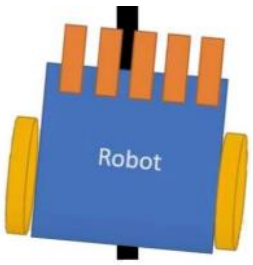


Loop การทำงานของ Robot Car

- กลไกการเดินตามเส้น

ใช้ IR Sensor ทั้งหมด 5 ตัว ในการอ่านข้อมูลเส้นบนสนาม ณ ขณะเวลา หนึ่ง ๆ นำมาเก็บใน Array เพื่อนำไปประมวลผลการตัดสินใจเดินต่อไป

โดย 1 หมายถึงสีขาว และ 0 หมายถึง สีดำ



ตัวอย่าง Robot Car ข้างต้น IR Sensor จะอ่านค่าได้ {1, 0, 0, 1, 1}

จากการคำนวณข้างต้น จะสังเกตได้ว่าหาก Robot Car อยู่ในสถานะปกติ (เดินตรง, อยู่กลางเส้น) จะอ่านค่า IR Sensor ได้เป็น {1, 1, 0, 1, 1} คือ อ่านเจอค่าสีดำที่ IR Sensor กลางเพียงตัวเดียว ในทางกลับกัน (เช่นในตัวอย่าง) หาก Robot Car มีการเอียงไปด้านขวา ก็จะอ่านค่าจาก IR Sensor ได้เป็น {1, 0, 0, 1, 1} โดยหากสรุปผลจากการอ่านค่า IR Sensor โดยทั่วไปที่เป็นไปได้ จะได้ดังตาราง

ค่าของ IR Sensor	พฤติกรรมของหุ่น
1 1 0 1 1	เดินตรง
1 0 0 1 1	ค่อนข้างเอียงซ้าย
0 0 1 1 1	เอียงซ้าย
1 1 0 0 1	ค่อนข้างเอียงขวา
1 1 1 0 0	เอียงขวา
1 1 1 1 1	หลุดออกจากเส้นทาง
0 0 0 0 0	เจอทางแยก

โดยที่พฤติกรรมของหุ่นที่แตกต่างกันในแต่ละกรณี จะนำไปสู่การตัดสินใจเดินที่แตกต่างกัน เช่น หากหุ่นอยู่ในสถานะเดินตรงตามเส้น ก็จะพยายามเดินตรงตามเส้นต่อไป แต่หากในกรณีที่หุ่นมีการเอียงออกจากเส้น ก็จะพยายามเดินกลับเข้าสู่เส้น โดยมี algorithm ในการเดินที่แตกต่างกัน ดังจะได้แสดงในลำดับถัดไป

- การควบคุมทิศทางของ Robot Car

ในการควบคุม Robot Car จะใช้ DC motor จำนวนสองตัว (ซ้ายและขวา) โดยสั่งงานผ่าน Motor Driver Module (L298N) โดยหากต้องการควบคุมให้ Robot Car เคลื่อนที่ไปในทิศทางที่ต่างกัน อาจมีหลักการในการสั่งงานดังนี้

ทิศทาง	**DC motor ซ้าย**	**DC motor ขวา**	**หมายเหตุ**
เดินหน้า	✓	✓	
ถอยหลัง	✓	✓	Motor หมุนถอยหลัง
เลี้ยวซ้าย	✗	✓	
เลี้ยวขวา	✓	✗	
หยุดนิ่ง	✗	✗	

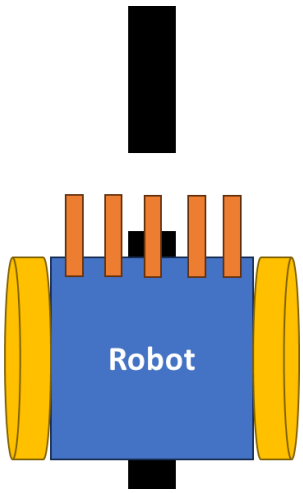
โดยในการสั่งงาน ได้สร้างฟังก์ชัน motor(int leftMotor, int rightMotor); เพื่อสั่งงาน motor ซ้ายและขวาตามลำดับ โดยความแรงของมอเตอร์แทนด้วยตัวแปร leftMotor, rightMotor ซึ่งจะได้ลงรายละเอียดต่อไปในการอธิบาย Source Code

หากใช้หลักการควบคุมทิศทางของ Robot Car ข้างต้น ก็จะสามารถควบคุมทิศทางพื้นฐานของ Robot Car ได้ โดยเมื่อนำมารวมกับการอ่านค่าของ IR Sensor ก็จะทำให้ Robot Car เดินได้ถูกต้องอย่างที่ควรจะเป็น ดังตาราง

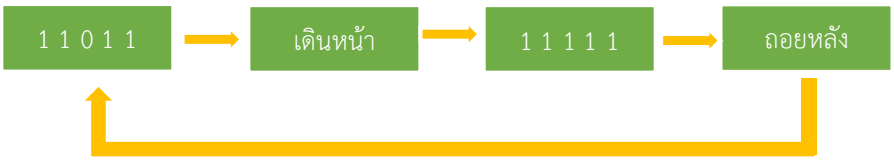
ค่าของ IR Sensor	**พฤติกรรมของหุ่น**	**การแก้ไข**	**DC motor ซ้าย**	**DC motor ขวา**	**หมายเหตุ**
1 1 0 1 1	เดินตรง	เดินหน้า	✓	✓	
1 0 0 1 1	ค่อนข้างเอียงซ้าย	เลี้ยวขวา	✓	✗	
0 0 1 1 1	เอียงซ้าย	เลี้ยวขวา	✓	✗	
1 1 0 0 1	ค่อนข้างเอียงขวา	เลี้ยวซ้าย	✗	✓	
1 1 1 0 0	เอียงขวา	เลี้ยวซ้าย	✗	✓	
1 1 1 1 1	หลุดออกจากเส้นทาง	ถอยหลัง	✓	✓	Motor หมุนถอยหลัง
0 0 0 0 0	เจอทางแยก	หยุดนิ่ง	✗	✗	

- Algorithm ในการจัดการกับเส้นประ

เนื่องจากในสนามแข่งมีเส้นที่เป็นลักษณะของเส้นประ ทำให้เมื่อตัว Robot Car เดินทางถึงตำแหน่งเส้นประดังกล่าว (ตามภาพ) แล้ว IR Sensor จะอ่านค่าได้เป็น {1, 1, 1, 1, 1}

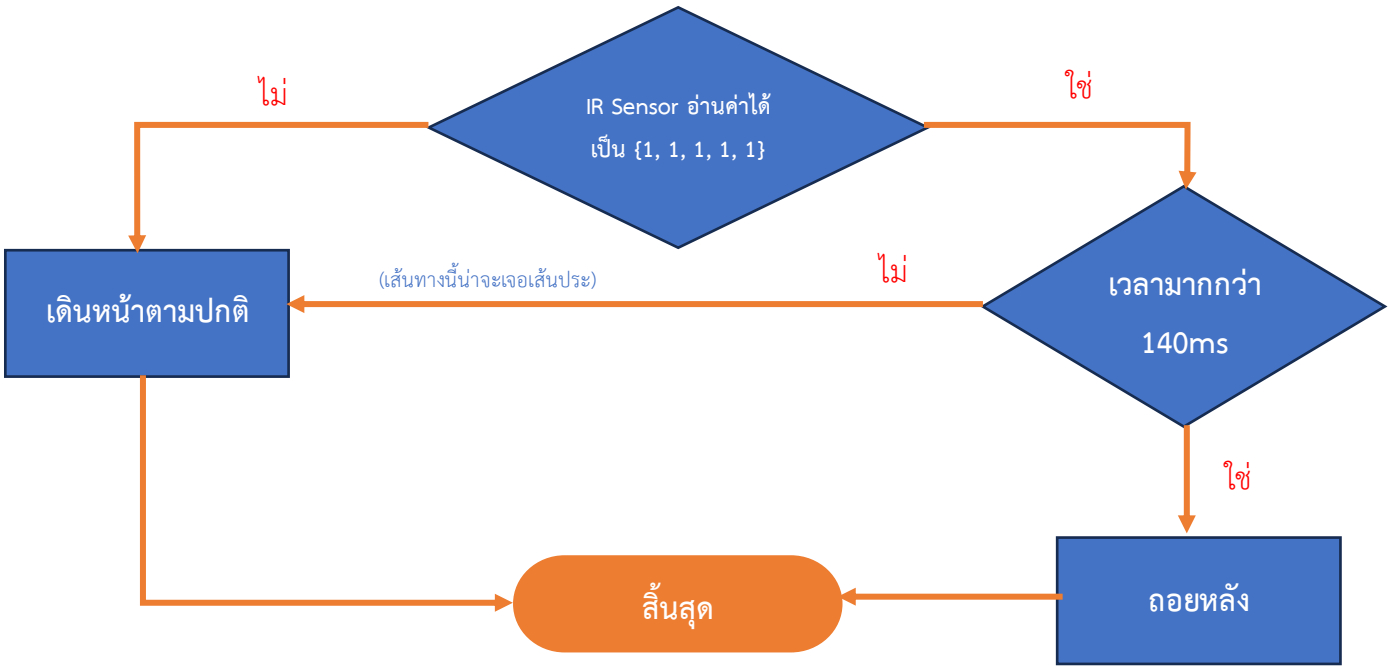


นั่นเอง ทำให้เมื่อ Robot Car เจอกับเส้นประ ตัว Robot Car จะมองว่าหลุดออกนอกเส้นทาง จึงทำการถอยหลังกลับจนกลับมาเจอเส้นสีดำ แล้วก็จะเดินหน้าจนเจอสีขาวทั้งหมดอีก แล้วก็จะถอยหลังกลับมาเจอสีดำอีก วนแบบนี้ไปเรื่อย ๆ ทำให้ Robot Car ไม่สามารถผ่านเส้นประได้

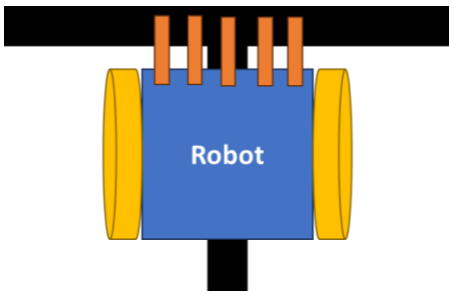


Robot Car เมื่อเจอกับเส้นประ
IR Sensor จะอ่านค่าได้เป็นสีขาวทั้งหมด

ในการแก้ไขปัญหานี้ ทำได้โดยการสร้างตัวแปรเวลา (time) ขึ้นมา โดยจะจับเวลาว่าหุ่นออกนอกเส้นทางไปนานเท่าใดแล้ว โดยหากหุ่นออกนอกเส้นทางเกินระยะเวลา Threshold ที่กำหนด จึงจะถือว่า Robot Car ออกนอกเส้นทางแล้วจริง ๆ จึงจะเริ่มถอยหลังกลับ ในทางกลับกัน หาก Robot Car ออกนอกเส้นทางไม่ถึงระยะเวลา Threshold แต่เจอเส้นสีดำ (กลับเข้าสู่เส้นทางได้) มีความเป็นไปได้ว่าหุ่นยนต์จะเดินผ่านเส้นประ ก็ไม่ต้องสั่งให้ Robot Car ถอยหลังกลับ โดยสามารถแสดงเป็นแผนผังการทำงานได้ดังนี้



- Algorithm ในการจัดการกับสัญญาณไฟ



เมื่อ Robot Car อ่านค่าจาก IR Sensor ได้เป็นสีดำทั้งหมด หรือ {0, 0, 0, 0, 0} (ตามภาพ) Robot Car จะ detect ได้ว่าตอนนี้น่าจะต้องรอรับสัญญาณไฟ เมื่อสัญญาณไฟกระพริบ 1 และ 2 ครั้ง Robot Car จะเลี้ยวไปทางซ้ายและขวาตามลำดับ

การตรวจสอบว่าสัญญาณไฟกระพริบ 1 หรือ 2 ครั้ง จะใช้ LDR Sensor ซึ่ง Output เป็น Digital คือ 0 หมายถึง มีแสงไฟ และ 1 หมายถึง ไม่มีแสงไฟ โดยมี Algorithm ในการจัดการกับสัญญาณไฟ ดังต่อไปนี้

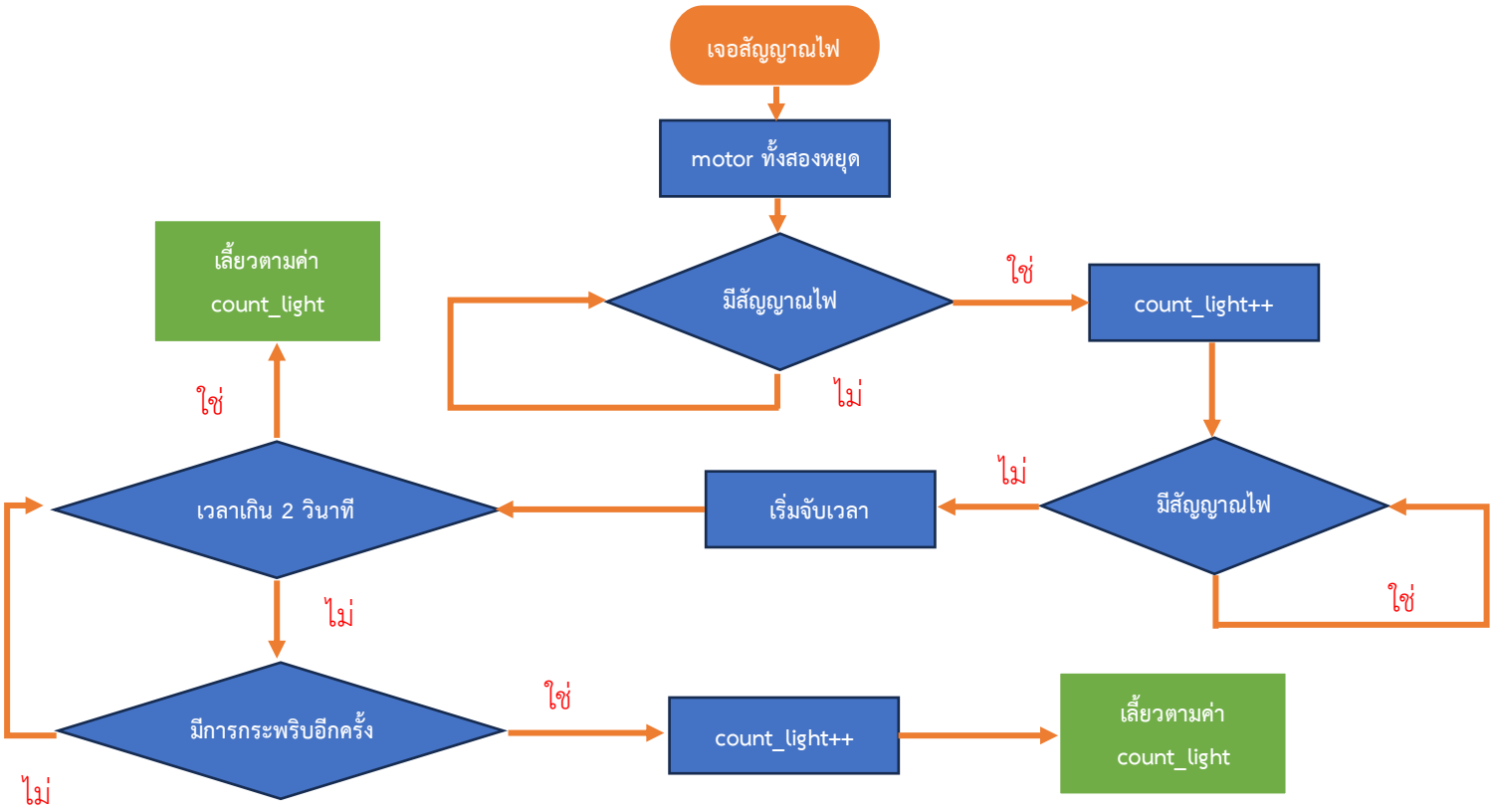
1. เมื่อ IR Sensor ตรวจสอบเจอทางแยก ให้ motor ทั้งสองหยุดการทำงาน เพื่อเตรียมรับสัญญาณไฟ

2. ให้ LDR Sensor รอรับแสง โดยหากมีแสง ให้เพิ่มค่าตัวแปร count_light จาก 0 -> 1 (หรือ 1 -> 2 หากเป็นการกระพริบครั้งที่ 2) และรอให้แสงเดิมดับ

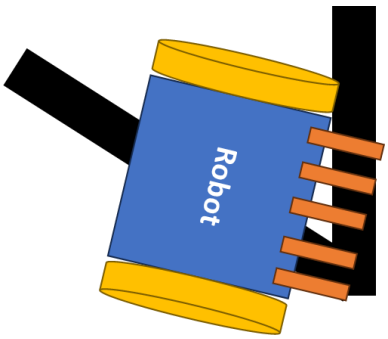
3. เมื่อแสงกระพริบครั้งแรกดับลง จะเริ่มจับเวลา เมื่อไม่มีแสงกระพริบอีกครั้งภายในเวลา 2 วินาที จะถือว่าเป็นการกระพริบ 1 ครั้ง และจะเลี้ยวซ้าย ในทางกลับกัน หากมีการกระพริบของแสงอีกครั้งภายในระยะเวลา 2 วินาที จะถือว่าเป็นการกระพริบ 2 ครั้ง แล้วจะเลี้ยวขวา (จำนวนครั้งในการกระพริบ เก็บไว้ในตัวแปร count_light)

4. เมื่อการเลี้ยวแต่ละครั้งสำเร็จแล้ว ให้ reset ค่าตัวแปร count_light กลับไปเป็น 0

Algorithm ในการจัดการกับสัญญาณไฟทั้งหมด แสดงดังแผนภาพ



- ปัญหาที่เกิดขึ้นเนื่องจาก Algorithm สัญญาณไฟ และการหยุดเมื่อถึงเส้นชัย



จาก Algorithm ในการจัดการกับสัญญาณไฟที่ได้กล่าวไปข้างต้น คือ ตรวจสอบว่า IR Sensor อ่านค่าได้เป็น {0, 0, 0, 0, 0} หรือไม่ จะทำให้มีโอกาสเล็กน้อยที่ Robot Car จะทำงานตามเงื่อนไขนี้ ทั้ง ๆ ที่ไม่ได้เจอแยกที่ต้องรอสัญญาณไฟ ตัวอย่างสถานการณ์ดังกล่าว เช่น เมื่อ Robot Car เผชิญหน้ากับทางเลี้ยวหักศอก (ตามภาพ)

ในการแก้ปัญหาดังกล่าว อาจทำได้ด้วยการตรวจสอบว่า Robot Car ผ่านแยกสัญญาณไฟมาครบสองครั้งแล้วหรือยัง หากครบแล้ว ก็ไม่ต้องให้ Robot Car หยุดรอสัญญาณไฟอีก (เนื่องจากตามสนามที่ใช้ในการแข่งขัน มีจุดหยุดรอสัญญาณไฟเพียงสองจุด และก่อนหน้าสัญญาณไฟจุดแรกไม่มีทางโค้งหักศอกที่ทำให้ Robot Car เกิดปัญหา) แต่หากใช้วิธีนี้ในการแก้ปัญหา จะพบว่า ตัว Robot Car จะไม่หยุดที่จุดเส้นชัย เนื่องจากได้ผ่านสัญญาณไฟมาครบสองครั้งแล้วนั่นเอง (แต่สามารถแก้ปัญหาที่ Robot Car หยุดเมื่อเจอโค้งหักศอกได้)

จากปัญหาข้างต้น จึงได้ทำการพัฒนา Algorithm ที่ใช้ในการแก้ปัญหา คือ เมื่อ Robot Car ผ่านจุดสัญญาณไฟทั้งสองจุดแล้ว จะไม่ให้ Robot Car หยุดเมื่อ IR Sensor อ่านค่าได้เป็น {0, 0, 0, 0, 0} อีก จนกว่าจะถึงเวลาที่กำหนด คือ เวลาที่มั่นใจแล้วว่า Robot Car ผ่านโค้งหักศอกทุกโค้งไปแล้ว (ในกรณีที่ทดสอบมาคือ ประมาณ 20 – 25 วินาที) จึงจะให้กลับมาหยุดอีกครั้งเมื่อ IR Sensor อ่านค่าได้สีดำทั้งหมด ทำให้ Robot Car หยุดเมื่อถึงจุดเส้นชัย แต่ไม่หยุดเมื่อเจอโค้งหักศอก

```
else if ((irSensor[0] == 0 && irSensor[1] == 0 && irSensor[2] == 0 &&
          irSensor[3] == 0 && irSensor[4] == 0) && millis() - stopClock >= 26500){
     while (1){
        digitalWrite(3, HIGH);
        motor(0, 0);
     }
}
```

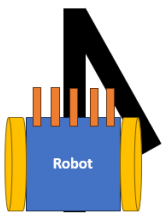
ตัวอย่าง Code ที่ใช้ในการแก้ปัญหา

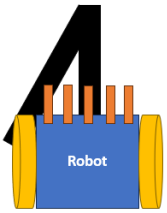
- **Algorithm การเลี้ยวเมื่อถึงโค้งหักศอกที่มุมแคบมาก ๆ (Hard-turn Algorithm)**

เมื่อเผชิญหน้ากับโค้งหักศอกที่มุมแคบมาก ๆ (โค้งมรณะ) บ่อยครั้งที่ Robot Car จะเกิดปัญหาในการเลี้ยว คือ ไม่สามารถเลี้ยวได้ หรือบางครั้งอาจออกนอกเส้นทางโดยไม่ทราบสาเหตุ จึงได้คิดค้น Algorithm ในการจัดการกับการเลี้ยวในมุมแคบพิเศษขึ้น เรียกว่า Hard-turn Algorithm

พฤติกรรมของโค้งหักศอกที่มีมุมแคบพิเศษ คือ ก่อนที่จะถึงจุดยอดของโค้งนั้น IR Sensor จะอ่านค่าได้เป็น {1, 1, 0, 1, 0} ในกรณีเลี้ยวขวา หรือ {0, 1, 0, 1, 1} ในกรณีเลี้ยวซ้าย (ตามภาพ)

กรณีเจอเลี้ยวขวามุมแคบ
IR Sensor จะอ่านค่าได้เป็น
{1, 1, 0, 1, 0}





กรณีเจอเลี้ยวซ้ายมุมแคบ
IR Sensor จะอ่านค่าได้เป็น
{1, 0, 1, 0, 0}

เพื่อจัดการกับโค้งโค้งมรณะที่กล่าวมาข้างต้น เมื่อ IR Sensor ตรวจสอบ Pattern ของเส้นได้ตามที่ได้กล่าวแล้ว จะใช้ Hard-turn Algorithm ในการจัดการกับโค้งดังกล่าว คือ

1. จะถอยหลังกลับออกมา เป็นระยะเวลา 200ms
2. จะโค้งไปในทิศทางที่ต้องการ เป็นเวลา 900ms (เวลาที่ใช้เป็นค่าคงตัว เกิดจากการ Calibration)
3. จะกลับเข้าสู่การควบคุมปกติ

จาก Hard-turn Algorithm ดังกล่าว ทำให้ Robot Car สามารถจัดการกับโค้งมรณะได้ โดยมีอัตราการสำเร็จมากกว่า 80% และหากการ Hard-turn ไม่สำเร็จ ก็มีโอกาสที่ Robot Car จะถอยหลังจนกลับมาเจอเส้นสีดำ และลอง Hard-turn อีกครั้งต่อไป

ปัญหาด้าน Hardware

ปัญหาที่เกิดขึ้นในการพัฒนา Robot Car ส่วนมาก ไม่ได้มาจากการเขียน Software หากแต่เป็น Hardware ที่ไม่ค่อยมีประสิทธิภาพและคุณภาพค่อนข้างต่ำ ในหัวข้อนี้ จะอธิบายปัญหาที่เกิดขึ้นเนื่องจาก Hardware และแนวทางในการแก้ไขปัญหา

- DC motor ที่มีกำลังที่ไม่เท่ากัน

จากการทดสอบ พบว่า เมื่อลองสั่งให้ DC motor ทำงานด้วยกำลังที่เท่ากันใน Software เช่น ทดลองใช้คำสั่ง

```
motor(150, 150);
```

ซึ่งควรทำให้ Robot Car เดินตรงไปด้านหน้า เนื่องจากสั่งให้ DC motor ซ้ายและขวาด้วยกำลังที่เท่ากัน แต่ในทางปฏิบัติพบว่า Robot Car มีการเคลื่อนที่เบี้ยวไปทางซ้าย เนื่องจาก DC motor ด้านขวามีกำลังที่มากกว่า (ทั้งที่สั่งใน Software เท่ากัน) จึงต้องแก้ปัญหาด้วยการเพิ่มกำลังของ DC motor ซ้าย เพื่อให้ DC motor ทั้งสองทำงานด้วยกำลังที่เท่ากัน โค้ดที่ใช้ในการแก้ปัญหา เป็นดังนี้

```
int errorShiftLeft = 40, errorShiftRight = 0;

if (irSensor[0] == 1 && irSensor[1] == 1 && irSensor[2] == 0 &&
    irSensor[3] == 1 && irSensor[4] == 0) {
        setForward();
        rightSPD = 100 + errorShiftRight;
        leftSPD = 100 + errorShiftLeft;
}
```

จากตัวอย่าง ได้กำหนดให้ errorShiftLeft = 40 คือ ในกรณีที่จะเดินตรง (หรือถอยหลังตรง ๆ) ให้ motor ด้านซ้ายแรงกว่าที่สั่ง 40 ในขณะที่ motor ด้านขวา แรงด้วยความเร็วเท่ากับที่สั่ง ก็จะทำให้หุ่นสามารถเดินเป็นเส้นตรงได้ตามปกติ

ทั้งนี้ การหาค่า errorShift ที่เหมาะสม ไม่ใช่เรื่องที่สามารถทำได้โดยง่าย เพราะค่า error ขึ้นอยู่กับหลายปัจจัย เช่น พื้นผิวที่ใช้ในการทดสอบมีแรงเสียดทานที่ไม่เท่ากัน หรือค่าความต่างศักย์ของถ่านที่ใช้ ณ ขณะที่ทำการทดสอบ (ซึ่งเป็นอีกหนึ่งปัญหาที่จะได้อธิบายต่อไป)

- ปัญหาเรื่องถ่าน รางถ่าน และเครื่องชาร์จถ่าน

ถ่าน รางถ่าน และเครื่องชาร์จถ่าน เป็นปัญหาหลักที่สำคัญในการพัฒนา Robot Car เนื่องจากความต่างศักย์ของถ่าน (ต่อจากนี้จะใช้คำว่า แบตเตอรี่) ในเวลาที่แตกต่างกันจะไม่เท่ากัน กล่าวคือ เมื่อทำการทดสอบ Robot Car ไปเรื่อย ๆ จะทำให้ความต่างศักย์ของแบตเตอรี่ลดลง ส่งผลให้ขับ DC motor ได้เบาลง เมื่อเทียบกับขณะที่แบตเตอรี่มีความต่างศักย์มากกว่า ปัญหานี้เอง ทำให้การ Calibrate ค่าตัวแปร errorShift ทำได้ยากขึ้น เนื่องจากค่า errorShift ในขณะที่แบตเตอรี่มีความต่างศักย์แตกต่างกันจะไม่เท่ากัน ซึ่งปัญหานี้สามารถแก้ได้ด้วยการนำแบตเตอรี่ไปชาร์จให้เต็มก่อนนำมาทำการ Calibrate และทดสอบ

อนึ่ง ถ่านที่ได้รับมาก้อนหนึ่ง มีอาการไม่สามารถอัดประจุได้ (ถ่านเสื่อม) จึงทำให้การชาร์จและ Calibrate เป็นไปโดยยาก โดยแก้ปัญหาด้วยการยืมถ่านเพื่อนเพื่อทำหารทดสอบในขณะที่เพื่อนกลุ่มอื่น ๆ ไม่ใช้

ซึ่งนำมาสู่ปัญหาต่อมา คือ เครื่องชาร์จถ่านมีอาการชาร์จไม่ได้ หรือบางครั้งชาร์จได้เพียงครั้งละ 1 ก้อน จึงทำให้ต้องผลัดกันชาร์จ ทำให้ความต่างศักย์ของถ่ายทั้งสองหมดช้า-เร็ว ไม่เท่ากัน ทำให้มีความลำบากในการทดสอบ Robot Car ภายหลัง เมื่อใช้งานที่ชาร์จไปเรื่อย ๆ พบว่า ที่ชาร์จไม่สามารถใช้งานได้เลย ทำให้สามารถชาร์จถ่านได้ แก้ปัญหาโดยการไปเบิกอุปกรณ์ใหม่ที่ห้อง 602

- ปัญหาเรื่องการอ่านค่าของ IR Sensor

จากกการทดลองต่อ IR Sensor เข้ากับ Analog Pin มีโอกาสเล็กน้อยที่ IR Sensor จะอ่านค่าผิดพลาด โดยบางครั้งจะอ่านค่าว่ากำลังเจอกับสีขาวอยู่ทั้ง ๆ ที่จริง ๆ แล้วอาจจะกำลังเจอกับสีดำ เป็นต้น แต่ไม่นานก็จะกลับมาใช้งานได้ตามปกติเช่นเดิม ปัญหานี้ หากเกิดขึ้นในจังหวะเวลาที่เหมาะสมก็อาจทำให้การทำงานของหุ่นผิดพลาดได้

อีกกรณีหนึ่ง คือปัญหา Hardware ที่เกิดจากการบัดกรี IR Sensor คือบัดกรีแล้วตะกั่วไม่เชื่อมกันสนิทดี ทำให้วงจรไม่สมบูรณ์ ส่งผลให้ IR Sensor ไม่สามารถอ่านค่าได้อย่างที่ควรจะเป็น

สรุป ปัญหาด้าน Hardware เป็นปัญหาสำคัญที่เกิดขึ้นในการทำ Robot Car ครั้งนี้ ถึงแม้ว่าจะมีประเด็นปัญหาน้อยกว่า Software แต่การที่ Hardware ไม่สามารถทำงานได้ นำไปสู่การไม่สามารถทดสอบ Hardware ได้นั่นเอง

Source Code

Soruce code

```
////////////////////////////////////////////////////
//                !!! Motor Pin !!!                //
//  Right Speed Control = 11 | Direction = 6 & 7  //
//  Left Speed Control = 5   | Direction = 9 & 10 //
// ---------------------------------------------   //
//       Direction 1 | Direction 2 | Robot        //
//           LOW          HIGH      FORWARD       //
//           HIGH         LOW       BACKWARD      //
////////////////////////////////////////////////////

const int leftMotorPin = 5, leftMotorDirection1 = 10, leftMotorDirection2 = 9;
const int rightMotorPin = 11, rightMotorDirection1 = 7, rightMotorDirection2 = 6;

const int onoffButton = 13;
const int LDR = 2 ;


// ---------------------------------------
```

code ส่วนนี้คือการ set pin ของ Hardware เข้ากับ Arduino

```
////////////////////////////
// IR pin = analog 0 - 4  //
////////////////////////////
const int irAmout = 5;
const int irPin[irAmout] = {};
int irSensorRead[irAmout] = {};
int irSensor[irAmout] = {};
const float irTreshold = 500;

bool isEndClockTriggered = false;

int hardturn = 0 ;
int count_hardturn = 0 ;

unsigned long time = 0 , time_stop = 0  , end_stop = 0 ;
unsigned long time_intersec = 0 ;
unsigned long delay_end = 0 ;

long stopClock = 0;

bool istime = false ;
bool islight = false ;
int count_light = 0 ;
int count_stop = 0 ;
const int baseSpeed = 150;
int errorShiftLeft = 40, errorShiftRight = 0;
int leftSPD, rightSPD;
```

ประกาศตัวแปรที่ใช้ในการควบคุมหุ่นยนต์

```
// Forward Declare
void motor(int leftMotorSPD, int rightMotorSPD);
void readSensor();
void calculateSpeed();
void setForward();
void setBackward();
void intersections();
```

ประกาศ Function เพื่อให้ง่ายต่อการใช้งาน

- function สั่งการทำงาน motor
- function อ่านค่า sensor
- function คำนวณ และปรับความเร็วของ motor
- function ตั้งค่าทิศทางของล้อ(เดินหน้า / ถอยหลัง)
- function ที่ต้องใช้เมื่อเจอทางแยกที่มรสัญญาณไฟ

```
void setup() {
  ////////////////////////////////////////////
  // Turn Serial of when not in use naja eiei //
  ////////////////////////////////////////////
  //Serial.begin(9600); // <--- Comment this line

  //////////////////////
  // MOTOR PIN SETUP //
  //////////////////////
  pinMode(rightMotorPin, OUTPUT);
  pinMode(leftMotorPin, OUTPUT);
  pinMode(rightMotorDirection1, OUTPUT);
  pinMode(rightMotorDirection2, OUTPUT);
  pinMode(leftMotorDirection1, OUTPUT);
  pinMode(leftMotorDirection2, OUTPUT);

  pinMode(onoffButton, INPUT);
  pinMode(LDR , INPUT);

  pinMode(3, OUTPUT);
  digitalWrite(3, LOW);
```

setup pin ต่าง ๆ ที่ต้องใช้

```
//*********************************************************************************************
void loop() {
  if (!digitalRead(onoffButton)) {
    motor(0, 0);
    delay(1000);
  }
  else if (digitalRead(onoffButton)) {
    readSensor();
    calculateSpeed();
    motor(rightSPD, leftSPD); // r l
  }
}
//*********************************************************************************************
```

ใช้ loop สั่งการโดยเมื่อกด switch ให้เริ่มทำงาน โดยเป็นลำดับดังนี้
1) อ่านค่าจาก sersor
2) นำค่าที่ได้จาก sensor แล้วนำมาคำนวณว่าควรเคลื่อนที่อย่างไร ทิศทางใด
3) สั่งงาน motor ให้ทำงานตามที่ได้คำนวณไว้

```
void setForward() {
  digitalWrite(rightMotorDirection1, LOW);
  digitalWrite(rightMotorDirection2, HIGH);
  digitalWrite(leftMotorDirection1, LOW);
  digitalWrite(leftMotorDirection2, HIGH);
}

void setBackward() {
  digitalWrite(rightMotorDirection1, HIGH);
  digitalWrite(rightMotorDirection2, LOW);
  digitalWrite(leftMotorDirection1, HIGH);
  digitalWrite(leftMotorDirection2, LOW);
}
```

function ในการตั้งค่า ทิศทางของล้อ ให้เดินหน้า หรือถอยหลัง

```
void readSensor() {
  irSensorRead[0] = analogRead(A0);
  irSensorRead[1] = analogRead(A1);
  irSensorRead[2] = analogRead(A2);
  irSensorRead[3] = analogRead(A3);
  irSensorRead[4] = analogRead(A4);

  for (int i = 0; i < irAmout; i++) {
    if (irSensorRead[i] > irTreshold) irSensor[i] = 1  ;
    else if (irSensorRead[i] <= irTreshold) irSensor[i] = 0;
  }
}
```

funtion ในการอ่านค่า sersor โดยอ่านค่าเป็น Analog และกำหนด Treshold ไว้ ถ้ามากกว่า Treshold ให้มีค่าเท่ากับ 1 == สีขาว และ ถ้าน้อยกว่า Treshold ให้มีค่าเท่ากับ 0 == สีดำ

```
void motor(int rightMotorSPD, int leftMotorSPD) {
  if (rightMotorSPD < 0) rightMotorSPD = 0;
  if (leftMotorSPD < 0) leftMotorSPD = 0;

  analogWrite(rightMotorPin, rightMotorSPD);
  analogWrite(leftMotorPin, leftMotorSPD);
}
```

Function ควบคุม motor ให้ทำงานตามที่เราคำนวณ

Function ในการคำนวณ และตัดสินใจว่าควรเคลื่อนที่อย่างไร

```
void calculateSpeed() {
    if(count_stop == 2 && !isEndClockTriggered) {
      stopClock = millis();
      isEndClockTriggered = true;
    }

  if (irSensor[0] == 0 || irSensor[1] == 0 || irSensor[2] == 0 || irSensor[3] == 0 || irSensor[4] == 0) time = millis();

  // Full Forward W-W-B-W-W
  if (irSensor[0] == 1 && irSensor[1] == 1 && irSensor[2] == 0 && irSensor[3] == 1 && irSensor[4] == 1) {
    setForward();
    rightSPD = 100  + errorShiftRight;
    leftSPD = 100 + errorShiftLeft;
  }

  // Encounter Intersection B-B-B-B-B
  else if (irSensor[0] == 0 && irSensor[1] == 0 && irSensor[2] == 0 && irSensor[3] == 0 && irSensor[4] == 0 && count_stop < 2 ) {//11011
    intersections();
    count_stop++;
  }
  else if ((irSensor[0] == 0 && irSensor[1] == 0 && irSensor[2] == 0 && irSensor[3] == 0 && irSensor[4] == 0) && millis() - stopClock >=
26500) {
    while (1) {
      digitalWrite(3, HIGH);
      motor(0, 0);
    }
  }

 // Force Backward W-W-W-W-W in case the robot is out of track more than 140ms
 else if ((irSensor[0] == 1 && irSensor[1] == 1 && irSensor[2] == 1 && irSensor[3] == 1 && irSensor[4] == 1) && millis() - time >= 140) {
   // New Bacward
   while (!(irSensor[0] == 0 || irSensor[1] == 0 || irSensor[2] == 0 || irSensor[3] == 0 || irSensor[4] == 0)) {
     readSensor();
     setBackward();
     rightSPD = 130;
     leftSPD = 130;
     motor(rightSPD, leftSPD);
   }
 }

 // Hard turning
 else if((irSensor[0] == 1 && irSensor[1] == 1 && irSensor[2] == 0 && irSensor[3] == 1 && irSensor[4] == 0) && count_stop >= 2){ // hard
turn right
 digitalWrite(3, HIGH);
   //Old hardturn
   setBackward();
   motor(130 + errorShiftRight, 130 + errorShiftLeft);
   delay(200);

   setForward();
   motor(0, 110);
   delay(950);
   digitalWrite(3, LOW);
 }
 else if((irSensor[0] == 0 && irSensor[1] == 1 && irSensor[2] == 0 && irSensor[3] == 1 && irSensor[4] == 1) && count_stop >= 2){ // hard
turn left
   digitalWrite(3, HIGH);

   setBackward();
   motor(130 + + errorShiftRight, 130 + errorShiftLeft);
   delay(200);

   setForward();
   motor(110 , 0);
   delay(900);
   digitalWrite(3, LOW);
 }

 // Normal turning
 else if (irSensor[0] == 0 || irSensor[1] == 0) {
   // Left Turn
     setForward();
     rightSPD = 90;
     leftSPD = 0;
 }
 else if (irSensor[3] == 0 || irSensor[4] == 0) {
   // Right Turn
     setForward();
     rightSPD = 0;
     leftSPD = 90 + errorShiftLeft;
 }

 // Default Case
 else  {
   setForward();
   rightSPD = 110;
   leftSPD = 110;
 }
}
```

ถ้า sersor ตรวจว่ามีสีดำอยู่ จะเริ่มต้นการจับเวลาใหม่ เพื่อนำไปใช้ในกรณี เจอเส้นประ

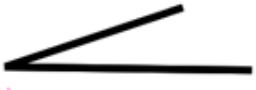
ถ้า sersor ตรวจเจอสีดำอยู่ตรงกลางเพียงอย่างเดียว (ทางตรง) จะสั่งให้ปรับความเร็วให้มากขึ้น

ถ้า sersor ตรวจเจอ สีดำทุกจุด แสดงว่าเจอทางแยก ดังนั้นจึงสั่งให้หยุด แล้วเข้าไปทำงานใน function intersections

เพื่อป้องกันไม่ให้ sensor ตรวจเจอสีดำทั้งหมดแล้วหยุด เลยได้ตั้งว่าเมื่อ เจอทางแยกครบ 2 ครั้งแล้วให้จับเวลา 26.5 วินาที ถ้าหลังจากนี้เจอสีดำทั้งหมดให้ หุ่นยนต์หยุด แล้วแสดงสัญญาณ LED ว่าจบแล้ว

เมื่อ sensor ตรวจเจอสีขาวทั้งหมด ให้จับเวลา ถ้าเวลาเกิน 140 ms ให้ถอยหลังมาจนกว่า sensor จะตรวจเจอสีดำ

เมื่อ sensor ตรวจเจอตรงกลาง เป็นสีดำ และทางขวาสุด เป็นสีดำ ซึ่งเป็นโค้งที่ค่อนข้างแคบ จึงทำให้ตอนเลี้ยวปกติไม่สามารถผ่านได้ จึงต้องให้ hard turn right ซึ่งคือการที่ต้อง ถอยหลัง 200 ms และให้ robot หันขวาไป ระยะเวลา 950 ms จนสามารถเข้าโค้งได้ปกติ



เมื่อ sensor ตรวจเจอตรงกลาง เป็นสีดำ และทางซ้ายสุด เป็นสีดำ ซึ่งเป็นโค้งที่ค่อนข้างแคบ จึงทำให้ตอนเลี้ยวปกติไม่สามารถผ่านได้ จึงต้องให้ hard turn left ซึ่งคือการที่ต้อง ถอยหลัง 200 ms และให้ robot หันซ้ายไป ระยะเวลา 900 ms จนสามารถเข้าโค้งได้ปกติ

เมื่อ sensor ทางโซนด้านซ้ายตรวจเจอสีดำ จะสั่งให้robot เลี้ยวซ้าย

เมื่อ sensor ทางโซนด้านขวาตรวจเจอสีดำ จะสั่งให้robot เลี้ยวขวา

ถ้าไม่เข้า case ไหนเลยให้เดินตรงไปข้างหน้า

Function ในการตัดสินใจเมื่อเกิดทางแยก ที่มีสัญญาณไฟ

```
void intersections(){
  motor(0,0);
  while(1){
    if(digitalRead(LDR) == 0){
      count_light++;
      while(digitalRead(LDR) == 0) {
        Serial.println(count_light);
        Serial.println("STOP");
        time_intersec = millis();
      }
    }
    else if(count_light > 0 ){
      if(count_light >= 2 && millis() - time_intersec >= 1000){
              // turn right
        setForward();
        motor(0, 130);
        delay(450);
        count_light = 0;
        break ;
      }
      else if(count_light == 1 && millis() - time_intersec >= 2000 ){
        //turn left
        setForward();
        motor(130, 0);
        delay(450);
        count_light = 0;
        break ;
      }
    }
  }
}
```

เมื่อเจอทางแยก สั่งให้ motor หยุดทำงาน

เมื่อเจอไฟ ตัวแปร count_light เพิ่ม 1 แล้วรอจนกว่าสัญญาณไฟจะหายไป

รอ 2 วิถ้ามีไฟ ให้ count_light เพิ่มอีก 1 แล้ว ถ้าไม่มีไฟมาให้เช็คเงื่อนไขต่อไปนี้

ถ้า count_light เท่ากับ 2 ให้เลี้ยวขวา

ถ้า count_light เท่ากับ 1 ให้เลี้ยวซ้าย