**PROJECT ROBOT CAR**

TRACKING LINE

**กลุ่มขายตรงแบบ 300 %**

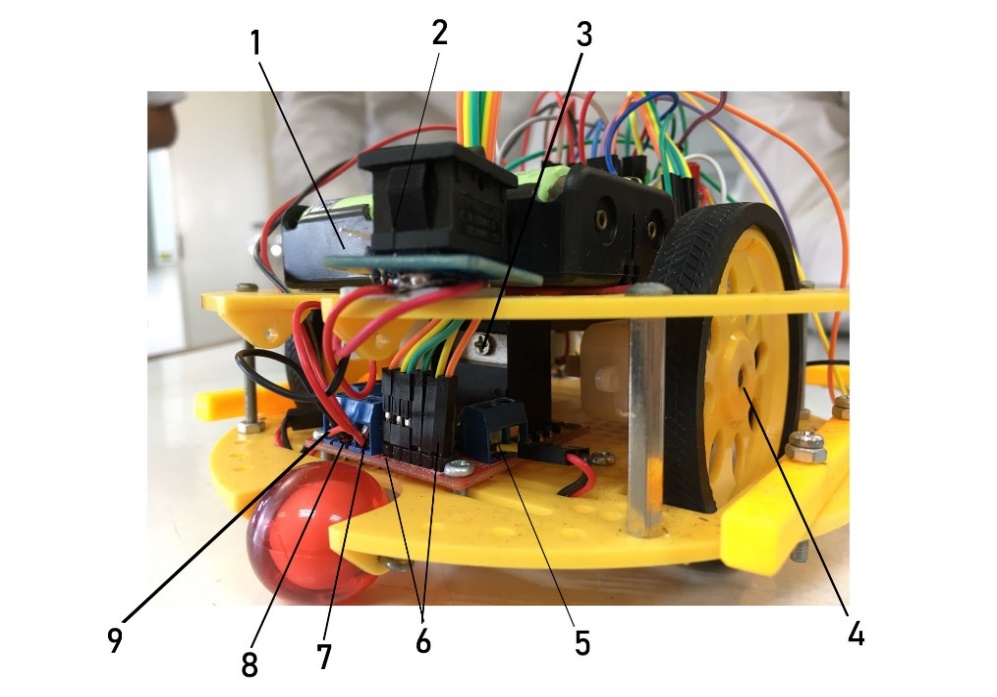
**A circuit board

Description automatically generated**

**A close up of a device

Description automatically generated**

**โครงสร้าง ROBOT CAR**

****

**1.Battery – ใช้ในการจ่ายไฟให้กับ Arduino (3.3v,5v)**

**2.Switch – ใช้เปิดปิดการจ่ายไฟ Battery – Arduino**

**3.L298N Module – ชุดมอเตอร์ชนิด H-Bridge ช่วยในการคุมความเร็วและทิศทางของตัวรถ**

**4.Wheel - ช่วยในการเคลื่อนที่ของรถ**

**5.Port DC Motor +, - มีหน้าที่ทำให้ motor ทำงาน**

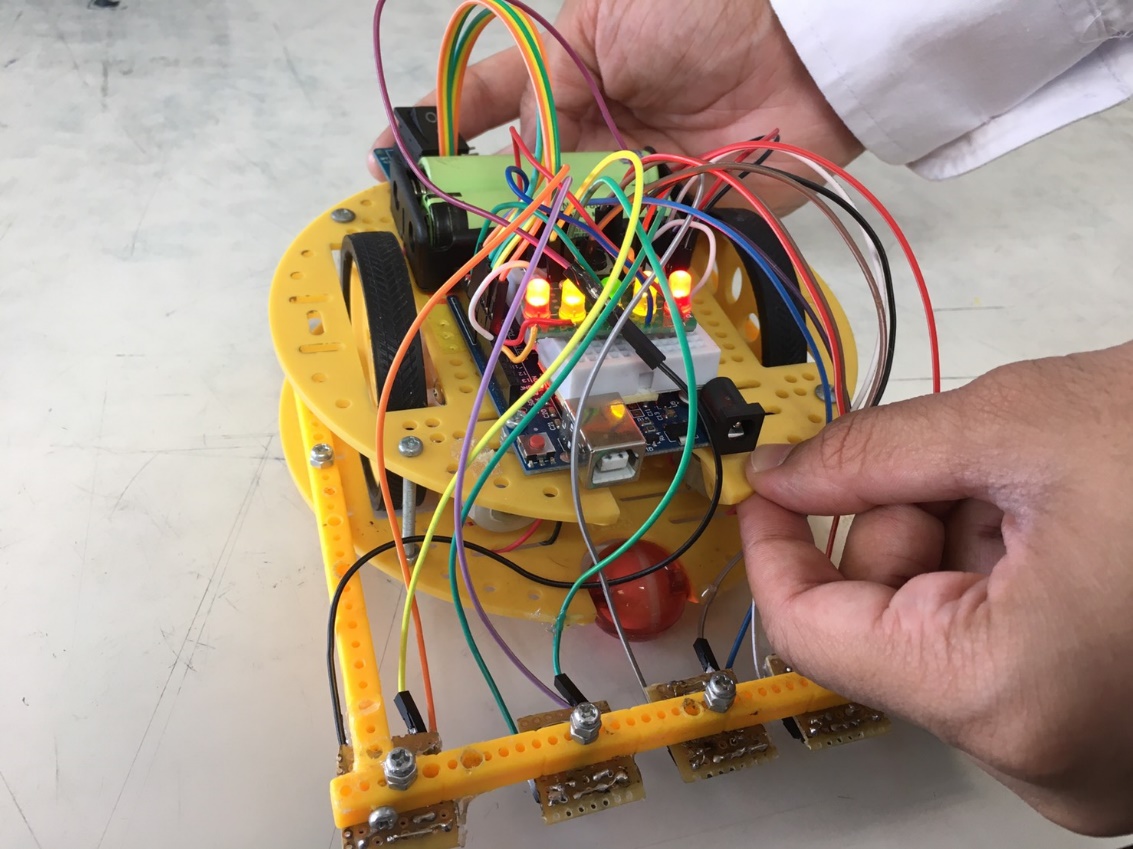
**6.Port Control speed - เสียบกับขา PWM ของ Arduino เพื่อควบคุมความเร็วในการเคลื่อนที่ (0-255)**

**7.Port Voltage - จุดจ่ายไฟ 5V นำไปต่อไฟเลี้ยง Arduino ผ่านขา 5V (12V jumper ต้องต่ออยู่)**

**8.Port GND – ต่อสายกราวด์**

**9.Port Voltage – จุดต่อไฟเข้าบอร์ด (ไม่เกิน 35V)**

**รายละเอียด ROBOT CAR**

****

**- ตัวรถของเราจะใช้เซนเซอร์ TCRT5000 เป็นจำนวน 5 ตัวด้วยกัน เพื่อใช้ในการ tracking ตามเส้นให้ง่ายขึ้นและแม่นยำขึ้น โดยเราจะมีก้านพลาสติก 3 ก้านด้วยกัน ก้าน 2 ก้านวาง**

**- ด้านข้างและมีก้านด้านหน้าเพื่อติดเซนเซอร์ โดยเราจะใช้ตะปูกับกาวร้อนในการเชื่อมเข้าด้วยกัน ทำให้เซนเซอร์จับค่าได้ง่ายขึ้นแล้ว calibrate ค่าได้ง่ายขึ้นอีก**

**- ด้านบนของรถจะมีหลอดไฟ LED ติดด้วยกันอยู่ 5 ดวง หลอดไฟ 5 ดวงนี้จะเปรียบเหมือนเป็นค่าเซนเซอร์ที่อ่านได้ว่าเป็นช่วงไหน**

**ก็คือมีหลอดไฟ L2 L1 M R1 R2 จะแทนที่เซนเซอร์คือ L2 L1 M R1 R2 เป็นลำดับ รถของเราเมื่อเซนเซอร์จับค่าที่เป็นช่วงของสีขาวได้ ก็จะให้หลอดไฟสว่าง และเมื่อเซนเซอร์จับค่าที่เป็นช่วงของสีดำได้ ก็จะให้หลอดไฟดับ**

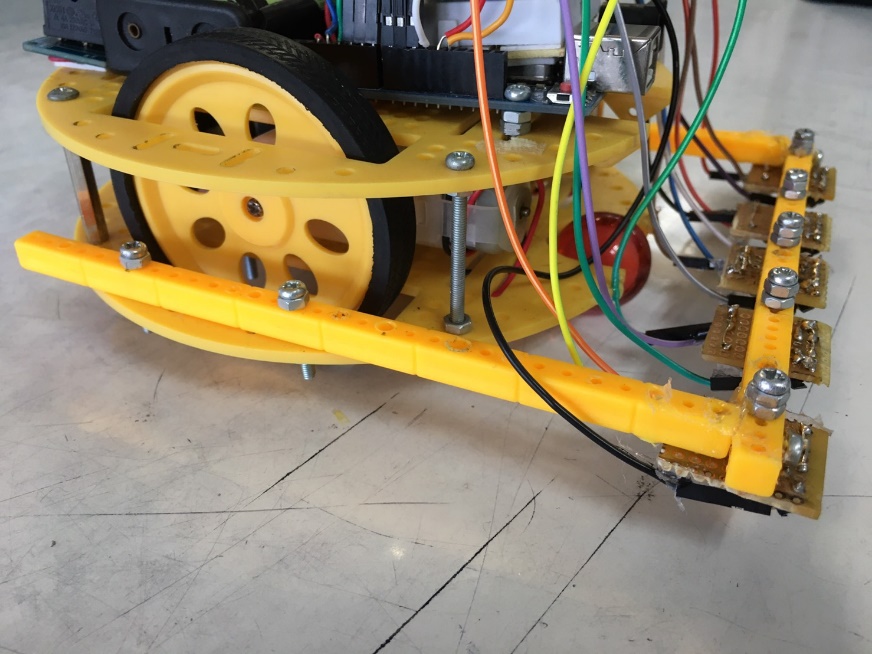
**\*\*\* โดยลายละเอียดการ calibrate นั้นจะอธิบายพร้อมกับส่วนของ code/program ในส่วนถัดไปครับ \*\*\***

**ปัญหาที่พบเจอ เกี่ยวกับ ROBOT CAR**

**A circuit board

Description automatically generated**

**1.ปัญหาแรกที่พบ – สายไฟขาดภายใน ทำให้การอ่านค่า เซนเซอร์ (TCRT5000) ผิดเพี้ยน และยากต่อการคาริเบทค่า ทำให้เสียเวลาการทดลอง**

****

**2.ปัญหาที่สอง - การติดตั้งเซนเซอร์ ช่วงแรกพวกเราติดค่าเซนเซอร์ไว้ประมาณ 1.7 mm แต่ค่าที่วัดได้ในช่วงสีดำและช่วงสีขาวผิดเพี้ยน จึงทำให้ยากต่อการคาริเบท ช่วงหลังเราก็เลยติดเซนเซอร์ไว้ประมาณ 3.5 mm ค่าที่วัดได้ในช่วงสีดำและช่วงสีขาวก็ผิดเพี้ยนเช่นกัน เราจึงพยายามหาค่าช่วงหนึ่งที่พีคที่สุด(หรือช่วงที่วัดค่าได้แม่นยำมากที่สุด) โดยการเริ่มจากค่อยเอาเซนเซอร์วัดระยะห่างระหว่างสนาม และอ่านค่าเซนเซอร์ว่า จุดไหนที่ค่าสีดำ และสีขาวห่างกันมากที่สุด เท่าที่จะเป็นไปได้ จากการทดลองได้ว่า ช่วงที่เยอะที่สุดคือ ห่างกันอยู่ประมาณ 100-300 คือสีขาววัดได้ประมาณ 600 ในขณะที่สีดำ วัดได้ประมาณ 300 ได้ระยะห่างจากพื้นประมาณ 2.6 mm จึงทำให้วัดค่าเซนเซอร์ได้ค่าผิดเพี้ยนที่น้อยลงมาก และง่ายต่อการคาริเบทมากขึ้น แต่กว่าจะหาจุดเจอ ก็เสียเวลา การทดลองไปหลายวัน**

**A picture containing yellow

Description automatically generated**

**3.ปัญหาที่สาม - สายไฟหลวมและไฟไม่ค่อยเข้า จึงทำให้ค่าเซนเซอร์เพี้ยนและระหว่างการแข่งขัน สายไฟหลุดออกจากกัน เราจึงใช้กาวร้อนมาช่วยในการติดตั้ง เนื่องจากกาวร้อนสามารถช่วยทำให้สายไฟกับตัวก้างปลามาเชื่อมกันได้ง่าย ทำให้สายไฟไม่หลุดออกจากกันระหว่างการแข่งขัน และอีกวิธีหนึ่งคือการบัดกรีเข้ากับสายจั้มโดยตรง**

**A close up of a device

Description automatically generated**

**4.ปัญหาที่สี่ – ค่าไฟเปลี่ยนแปลงอยู่เสมอ ปัญหานี้เกิดมาจากแบตเตอรี่ในการใช้งานที่ลดลงไปเรื่อยๆตามการใช้งานของเรา ทำให้บางที การคาริเบทค่า ของเซนเซอร์อาจเปลี่ยนแปลงได้เล็กน้อย สามารถมีผลต่อระบบcodingหลักได้ ทำให้ต้องคอยชาร์จแบตอย่างสม่ำเสมอ ไม่ปล่อยให้แบตน้อยเกินไป เพื่อรักษาค่าความเร็ว(speed) ไว้ให้เท่าๆกัน และค่าของเซนเซอร์ ที่คาริเบทไว้ตั้งแต่แรก ให้เท่าๆกัน ตลอดการทดลอง**

**A picture containing indoor

Description automatically generated**

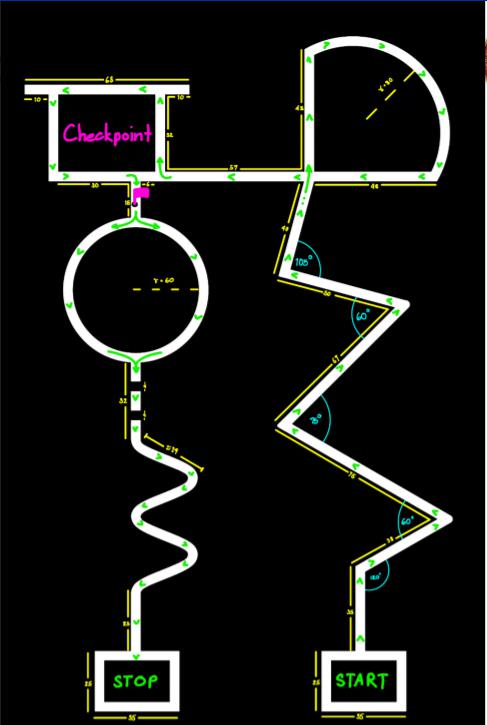
**ในรูปด้านบนเกิดจากปัญหาค่าไม่ค่อยห่างกัน ทำให้วัดช่วงยาก และคาริเบทไปก็ไม่ก่อประโยชน์ใดๆ จึงแก้ไขด้วยวิธีการคาริเบทตามสนามจริงๆ**

**5.ปัญหาที่ห้า – การคาริเบท(Callibration) ปัญหานี้กว่าจะแก้ได้ นานอยู่พอสมควร เพราะเวลาคาริเบทค่าจริงในคอมพิวเตอร์ กับเวลาลงสนามจริงๆ เมื่อถอดสาย USB ออกจะพบได้ว่า ค่าคาริเบทเพี้ยนไป ตอนแรกคิดว่า เป็นที่เซนเซอร์เสีย หรือติดตั้งตำแหน่งผิด แต่มีจุดๆหนึ่งที่สังเกตได้ชัดเจนเลยว่า เมื่อเสียบสาย USB ค่าที่คาริเบทตรงกับค่าความจริงทุกอย่าง พอถอดออก ค่ากับผิดเพี้ยนไป เราจึงเริ่มรู้แล้วว่า ปัญหาน่าจะเกิดมาจาก ไฟของถ่านและไฟจากคอมพิวเตอร์ที่มีผลทำให้การอ่านค่าเพี้ยนไปหมด รวมถึงการต่อเซนเซอร์แบบ 3V หรือ 5V ด้วย ช่วงแรก มีปัญหามาก เพราะต่อ 5V ตาม datasheet ที่อาจารย์แจกให้ ทำให้เวลาจะคาริเบท ถ้าอยากให้ตรงค่ามากที่สุดต้องเปิดไฟถ่านไว้ แต่เมื่อถอดออก ก็ไม่ตรงอยู่ดี เลยไม่มีประโยชน์อะไร เราจึงลองมาเปลี่ยนมาใช้ 3V เลี้ยงเซนเซอร์ทุกตัว ปรากฏว่า เวลาคาริเบท ถ้าใช้ 3V ไม่จำเป็นต้องเปิดสวิตซ์ถ่าน ทำให้ค่ามีความตรงกับความเป็นจริงมากขึ้น แต่ก็ยังไม่ดีพอ เพราะเมื่อถอดสาย USB ค่าก็เปลี่ยนอีก ไม่ได้ช่วยอะไร**

**การแก้ปัญหาของกลุ่มผม เลยเป็นการที่ ยึดการจ่ายไฟแบบ 3V ที่ใกล้เคียงที่สุด และอ่านค่าจาก USB และทำการคาริเบทค่าที่อ่านได้ภายในคอมไปก่อน เมื่อถอดสาย และทดลองจริง ให้สังเกตหลอดไฟที่กลุ่มพวกผมได้ติดตั้งไว้ ถ้าปกติ ก็ถือว่าดี แต่ถ้ามีอันไหนผิดปกติ ก็ให้ปรับแก้ตามความเป็นจริง**

**โค้ดการคาริเบทเบื้องต้นของผมคือ ถ้าเกิดเกินค่า เท่านี้ จะเป็นสีขาว ไฟจะติด ถ้าต่ำกว่านี้ ก็จะเป็นสีดำ หลอดไฟ LED ก็จะไม่ติด จึงสามารถคาริเบทปรับค่าได้จากการทดลองจริง เพิ่มขึ้น หรือลดลงเล็กน้อย ส่วนใหญ่จะอยู่ +- ไม่เกิน 100-200 ต่อเซนเซอร์ และเซนเซอร์ทุกตัว ค่าก็ไม่เท่ากันอีก ก็ต้องคอยมานั่งคาริเบทแต่ละตัว แต่เมื่อทำสำเร็จแล้ว ใช้ได้ผลมากๆกับการแทรคเส้น หลอดไฟ LED ติดถูกต้องตามความเป็นจริงอย่างที่ควรจะเป็นทุกดวง**

**แผนที่สนาม ROBOT CAR**

****

**ขั้นตอนการคิด ALGORYTHM LOGIC**

**A picture containing screenshot

Description automatically generated**

**แมพส่วนนี้ กลุ่มผมจะแบ่งออกเป็น 2 ส่วนหลักๆ ก็คือ**

**1.ส่วนแรก - การเดินทางจาก Start -> Checkpoint**

**2.ส่วนที่สอง - การเดินทางจาก Checkpoint -> Stop**

**เหตุผลในการแบ่งก็คือ จะได้คิดอัลกอริทึมได้ง่ายขึ้น และเขียนโค้ดได้ง่ายขึ้น สามารถถึงเส้นชัยได้ และช่วยลดความผิดพลาดของ hardware ระหว่างทาง เช่น เซนเซอร์วัดค่าผิดพลาด หรือหลุดเข้าไปใน state ที่ผิด เป็นต้น**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**PROCESS PART I Start -> Checkpoint**

**PART 1.1**

A screenshot of a social media post

Description automatically generated

**1.โค้ดส่วนแรกจะเป็นส่วนของการต่อสายจั้มเข้า PIN ต่างๆ โดยจะมี PIN LED 5 ดวง PIN Sensor 5 ดวง สายจั้มสำหรับmotor ของตัวรถ กำหนด speed Left wheel = 90 หน่วย และ speed Right Wheel = 70 หน่วย ที่ไม่เท่ากันเพราะเกิดจากการทดลอง เมื่อทำให้ความเร็วเท่ากัน พบว่า ทั้งสองข้างหมุนเร็วช้าต่างกัน เราจึงพยายามหาค่าความเร็วที่ทำให้รถสามารถแล่นได้ตรงที่สุด ก็คือล้อซ้ายต้องช้ากว่าล้อขวาอยู่ 20 หน่วย**

**PART 1.2**

A close up of text on a white background

Description automatically generated

**2.โค้ดส่วน setup/output/input**

**โดยเราจะให้สายของ motor LED เป็น OUTPUT ส่วน สายของ sensor(TCRT5000) เป็น INPUT**

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

**3.โค้ดส่วน tracking line**

**โดยเราจะให้ state ขึ้นอยู่กับสถานะของไฟ สมมุติในโค้ดเขียนว่า L1 && L2 && M && R1 && R2 เป็น HIGH ทั้งหมด จะให้รถเดินไปข้างหน้า แต่ใน state นี้เราสามารถกำหนดความเร็วให้กับรถได้อีกด้วย**

A screenshot of a social media post

Description automatically generated

**4.โค้ดส่วน Callibrate**

**เราจะทำการวัดช่วงและกำหนดค่าให้กับช่วงของสีดำและสีขาว โดยปกติทั่วไปแล้ว สีดำจะมีค่าน้อยกว่าสีขาว เราจึงใช้สูตรคำนวณ**

**((ช่วงสีดำมากที่สุด)+(ช่วงสีขาวน้อยที่สุด)) / 2**

**EX1: ค่าสีดำที่วัดได้ ได้แก่ 230 240 250 260 270 ค่าสีขาวที่วัดได้ ได้แก่ 700 690 670 650 640 เราจะวัดช่วงได้ดั้งนี้**

**DarkMax = 270 , BrightMin = 640**

**ช่วงกึ่งกลางคือ (270+640) / 2 = 455**

**สรุป : Dark < 455 < Light**

**แต่ทั้งหมดนี้ คือในทางทฤษฎี ในทางปฏิบัติแล้ว ใช้ได้ยากเพราะ…**

1. **เซนเซอร์ ไม่ได้รับค่าที่ถูกต้องจริงๆ อาจเกิดจากความคลาดเคลื่อนของทั้งตัวเซนเซอร์เอง หรือสภาพแวดล้อมอื่นๆเช่น สนาม หรือไฟ**
2. **ความคลาดเคลื่อนของคนวัด การวัดของแต่ละคน ก็จะไม่เท่ากันมาก เมื่อเอาไปเทียบกับสนามอีกที่ ค่าก็มีความต่าง น้อยไม่เท่ากัน สุดท้าย ก็ต้องคาริเบทใหม่ ซ้ำไปซ้ำมาเรื่อยๆ**
3. **ค่าไฟของถ่านมีผลอย่างมาก เนื่องจากการอ่านค่าของเซนเซอร์ขึ้นอยู่กับค่าไฟของถ่านส่วนนึงด้วย ดังนั้น การที่เราคาริเบทดูค่าสดๆจากคอมพิวเตอร์ ด้วยการเสียบสาย usb เป็นการทำให้ไฟไม่เท่ากับเวลาปล่อยรถแล่นจริงๆ ทำไง เกิดค่าเพี้ยน**

**วิธีแก้ก็คือ**

**ใช้ไฟให้เซนเซอร์ทั้งหมดเป็น 3.3V ขณะคาริเบท ห้ามเปิดแบต เสียบได้แค่สาย USB จะได้ค่าคาริเบทที่ค่อนข้างตรงกับถอดใช้จริงที่สุด แต่ก็ยังมีความคลาดเคลื่อน เราจึงใช้การวัดค่าตรงๆจาก USB ก่อน แล้วจึงค่อยๆปรับตามความเหมาะสมขณะใช้จริง เช่น ถ้าเกิดค่าในคอมเมื่อเสียบ USB ได้ค่า 300 (มากกว่านี้ให้ไฟติด) แต่ถ้าไปวางสีขาว แล้วไฟยังดับอยู่ ให้ตั้งค่าให้ลดลงกว่านี้ (เพื่อให้ไฟติดในพื้นที่สีขาว) การทำแบบนี้จะเป็นการทำให้เซนเซอร์สามารถอ่านค่าได้ถูกต้องตามสภาพแวดล้อมจริง ทำให้สามารถแทรคเส้นได้อย่างไม่มีปัญหาความคลาดเคลื่อนมากนัก**

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

**5.โค้ดส่วน DC MOTOR**

**การทำงานของล้อรถมีอยู่ 2 ฟังก์ชั่น คือ go\_L() และ go\_R() go\_L() จะทำงานแค่ล้อขวา(เพราะเลี้ยวซ้าย) go\_R() จะทำงานแค่ล้อซ้าย(เพราะเลี้ยวขวา)**

**การที่เขียนฟังก์ชั่นแบบนี้ ทำให้ง่ายต่อการเขียนโปรแกรม และความเข้าใจของคนที่อ่าน และกันปัญหาเรื่องการจ่ายไฟให้มอเตอร์ทั้งสองพร้อมกัน(กันระเบิดนั้นเอง) นี่เป็นสาเหตุที่เราเขียนฟังก์ชั่นจากการทำงานเล็กๆ เช่นgo\_R(), go\_L() แล้วรวบฟังก์ชั่นเล็กๆเป็นฟั่งก์ชั่นใหญ่ๆอีกที เช่น Forward(), Backward() เป็นต้น**

**A close up of a map

Description automatically generated**

**PROCESS PART II Checkpoint -> Stop**

**โค้ดในส่วนที่สองจะต่างจากส่วนแรกตรงที่ stateLeft เนื่องจาก การแทรกส่วนแรก เราจะใช้ stateLeft เข้ามาช่วยเพื่อให้ตัวรถสามารถหันไปทางซ้ายได้สองครั้ง และเมื่อเข้า stateLeft สองครั้งแล้วเราจะทำการดักเคสเพิ่มเพื่อที่จะให้รถขับผ่านจุด 4 แยกได้ โดยที่ไม่เลี้ยวซ้ายก่อน**

**A picture containing object

Description automatically generated A close up of a computer

Description automatically generated**

**โค้ดส่วนสุดท้ายจะคล้ายๆกับส่วนแรก แต่เพิ่มเคสมา 1 เคส นั่นก็คือ เมื่อ L1 L2 M R1 R2 (หรือทุกเซนเซอร์) เป็น LOW จะให้รถทำฟังก์ชัน Forward(); (เดินหน้า) เพื่อที่จะผ่านตรงรอยประในสนามได้**

**A close up of a sign

Description automatically generated**