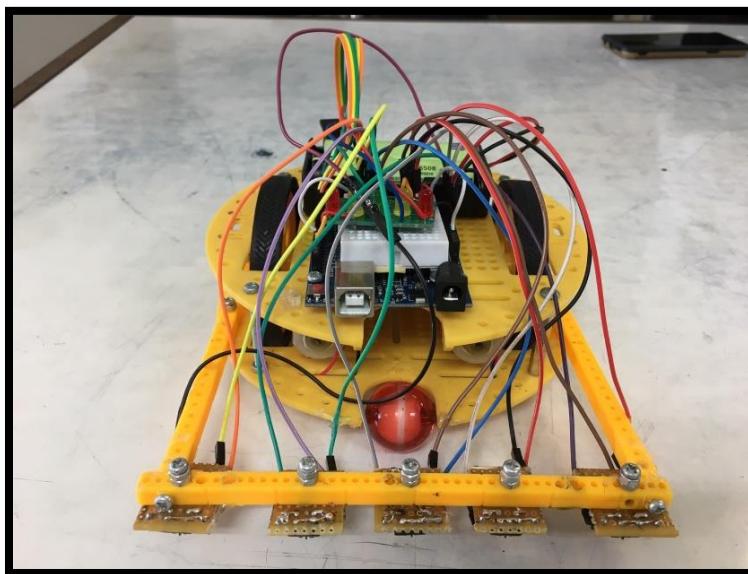
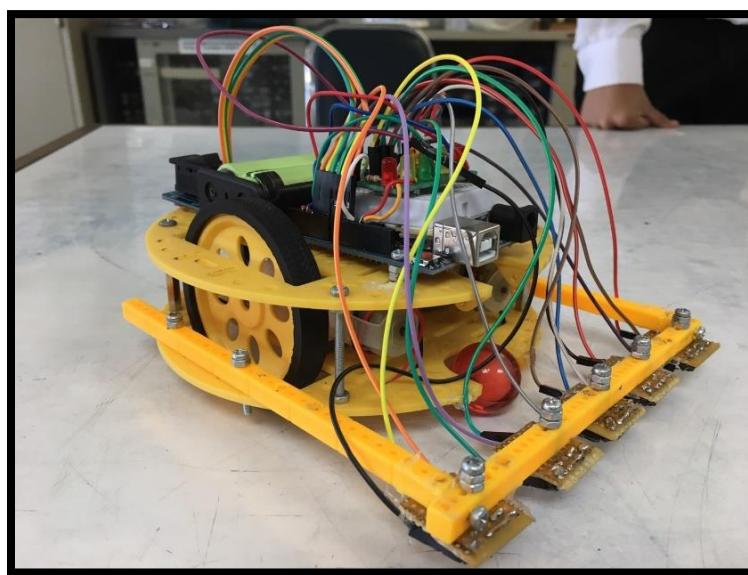


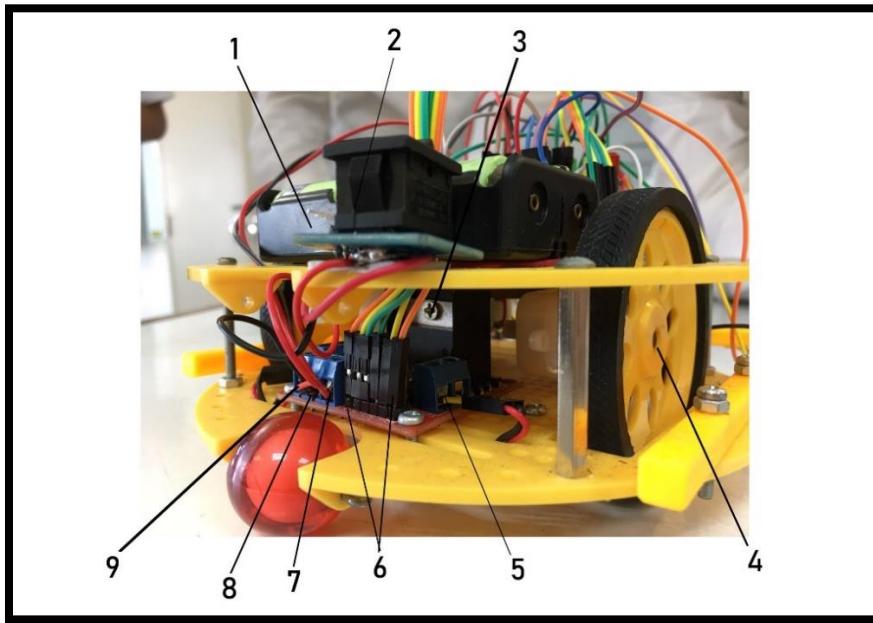
PROJECT ROBOT CAR

TRACKING LINE

กลุ่มขายตรงแบบ 300 %



โครงสร้าง ROBOT CAR



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

2. Switch – ใช้เปิดปิดการจ่ายไฟ Battery – Arduino

3. L298N Module – ชุดมอเตอร์ชินิค H-Bridge ช่วยในการคุ้มความเร็วและทิศทางของตัวรถ

4. Wheel - ช่วยในการเคลื่อนที่ของรถ

5. Port DC Motor +, - มีหน้าที่ทำให้ motor ทำงาน

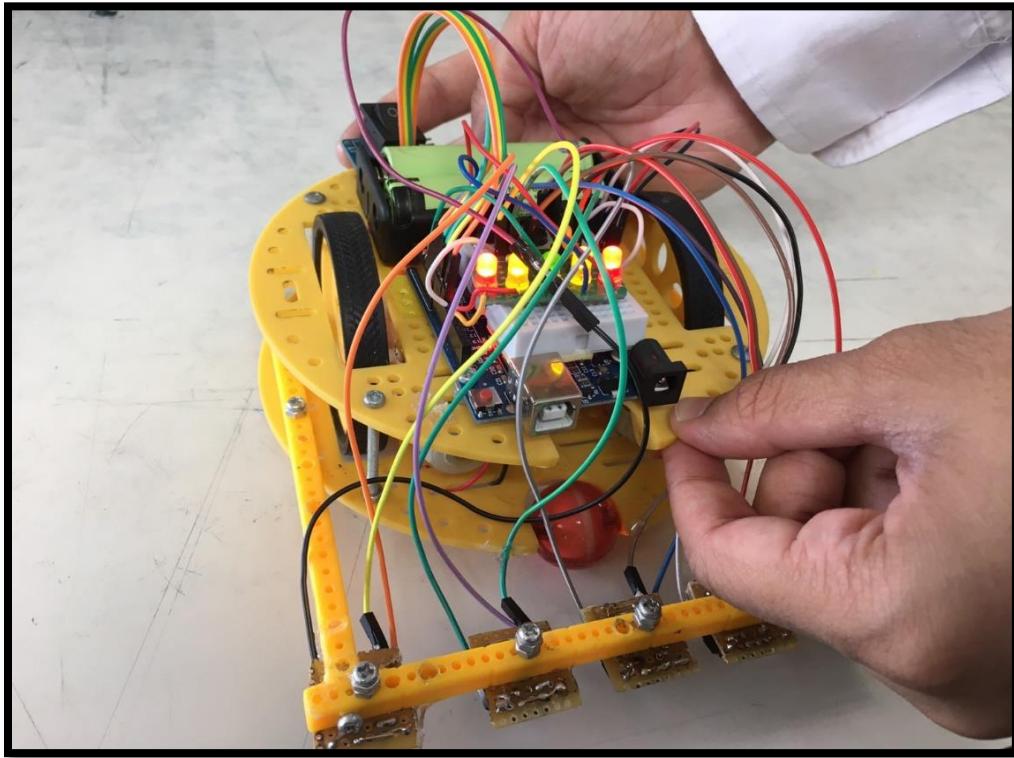
6. Port Control speed - เสียบกับขา PWM ของ Arduino เพื่อควบคุมความเร็วในการเคลื่อนที่ (0-255)

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 โดยเรามีก้านพลาสติก 3 ก้านด้วยกัน ก้าน 2 ก้านวาง

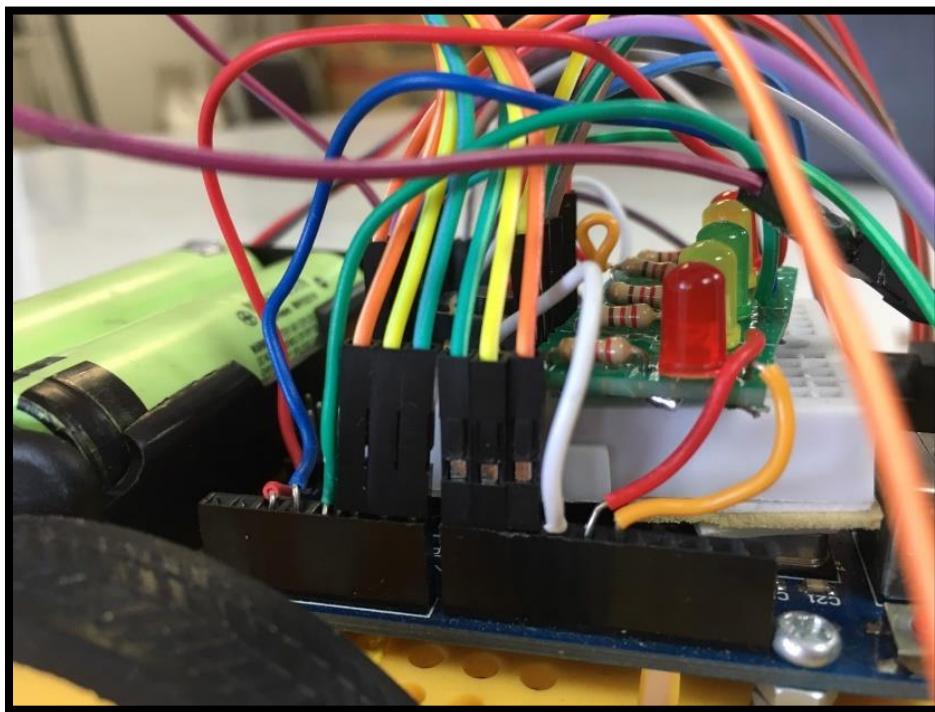
- ด้านข้างและมีก้านด้านหน้าเพื่อติดเซนเซอร์ โดยเราจะใช้ตะปุกับกรร้อนในการเชื่อมเข้าด้วยกัน ทำให้เซนเซอร์จับค่าได้ง่ายขึ้นแล้ว **calibrate** ค่า ได้ง่ายขึ้นอีก

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

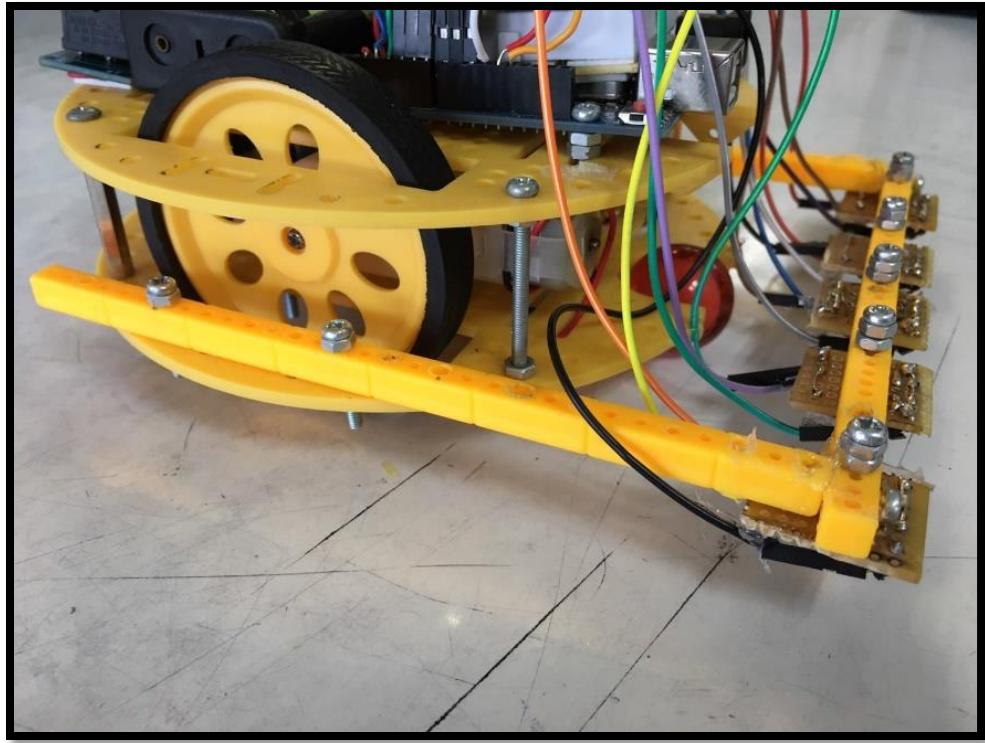
ก็คือมีหลอดไฟ L2 L1 M R1 R2 จะแทนที่เซนเซอร์คือ L2 L1 M R1 R2 เป็นลำดับ รถของเรามีเซนเซอร์จับค่าที่เป็นช่วงของสีขาวได้ ก็จะให้หลอดไฟสว่าง และเมื่อเซนเซอร์จับค่าที่เป็นช่วงของสีดำได้ ก็จะให้หลอดไฟดับ

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

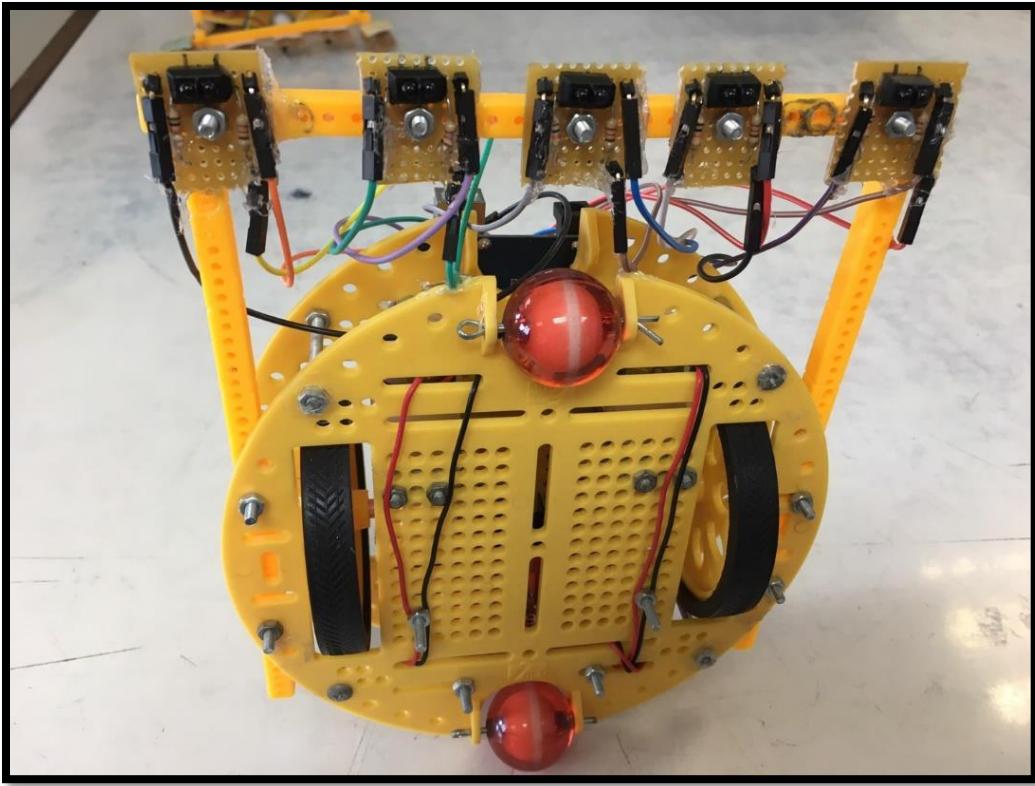
ปัญหาที่พบเจอ เกี่ยวกับ ROBOT CAR



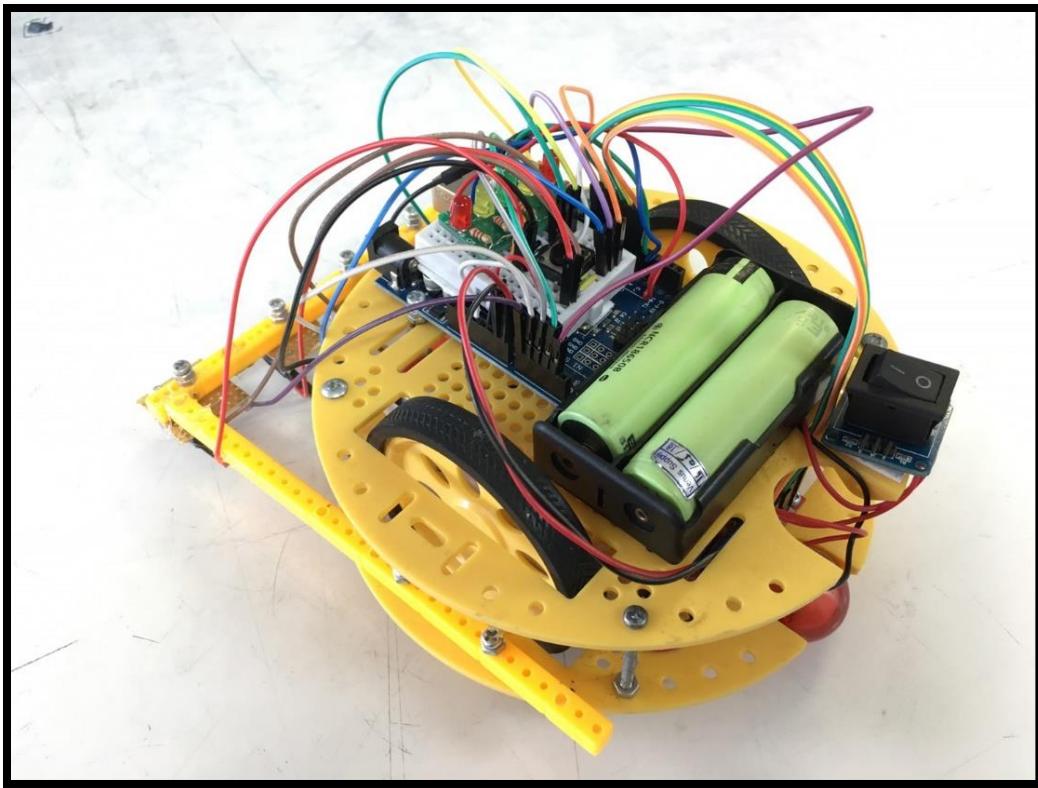
1.ปัญหาระบบที่พน – **สายไฟขาดภายใน** ทำให้การอ่านค่า เซนเซอร์ (TCRT5000) ผิดเพี้ยน และยกต่อการคำนวณค่า ทำให้เสียเวลาการทดลอง



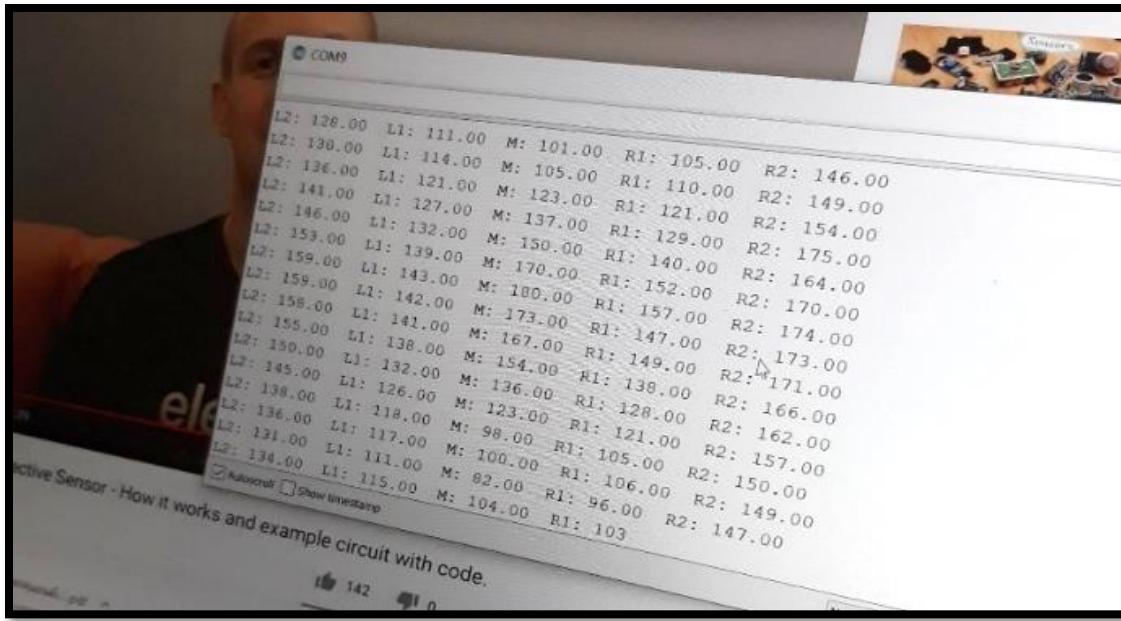
2. ปัญหาที่สอง - การติดตั้งเซนเซอร์ ช่วงแรกพบว่าติดค่าเซนเซอร์ไว้ประมาณ 1.7 mm แต่ค่าที่วัดได้ในช่วงสีดำและช่วงสีขาวผิดเพี้ยน จึงทำให้ยากต่อการcaribeث ช่วงหลังเราก็เลยติดเซนเซอร์ไว้ประมาณ 3.5 mm ค่าที่วัดได้ในช่วงสีดำและช่วงสีขาวก็ผิดเพี้ยนเช่นกัน เราจึงพยายามหาค่าช่วงหนึ่งที่พิเศษที่สุด(หรือช่วงที่วัดค่าได้แม่นยำมากที่สุด) โดยการเริ่มจากค่าอย่างเอาเซนเซอร์วัดระยะห่างระหว่างสนาม และอ่านค่าเซนเซอร์ว่า จุดไหนที่ค่าสีดำ และสีขาวห่างกันมากที่สุด เท่าที่จะเป็นไปได้ จากการทดลองได้ว่า ช่วงที่酵ะที่สุดคือ ห่างกันอยู่ประมาณ 100-300 คือสีขาววัดได้ประมาณ 600 ในขณะที่สีดำ วัดได้ประมาณ 300 ได้ระยะห่างจากพื้นประมาณ 2.6 mm จึงทำให้วัดค่าเซนเซอร์ได้ค่าผิดเพี้ยนที่น้อยลงมาก และง่ายต่อการcaribeทมากขึ้น แต่กว่าจะหาจุดเจอก็เสียเวลา การทดลองไปหลายวัน



3.ปัญหาที่สาม - สายไฟหลุมและไฟไม่ค่อยเข้า จึงทำให้ค่าเซนเซอร์เพี้ยนและระหว่างการแข่งขัน สายไฟหลุดออกจากกัน เราจึงใช้การร้อนมาช่วยในการติดตั้ง เนื่องจากภาวะร้อนสามารถช่วยทำให้สายไฟกับตัวก้างปลาสามารถเชื่อมกันได้ง่าย ทำให้สายไฟไม่หลุดออกจากกันระหว่างการแข่งขัน และอีกวิธีหนึ่งคือ การบัดกรีเข้ากับสายจ้มโดยตรง



4.ปัญหาที่สี่ – ค่าไฟเปลี่ยนแปลงอยู่เสมอ ปัญหานี้เกิดมาจากการแบตเตอรี่ในการใช้งานที่ลดลงไปเรื่อยๆ ตามการใช้งานของเรา ทำให้บางที การcaribeทค่าของเซนเซอร์อาจเปลี่ยนแปลงได้เล็กน้อย สามารถมีผลต่อระบบCodingหลักได้ ทำให้ต้องคอยชาร์จแบตอย่างสม่ำเสมอ ไม่ปล่อยให้แบตหมดน้อยเกินไป เพื่อรักษาค่าความเร็ว(speed) ไว้ให้เท่าๆกัน และค่าของเซนเซอร์ ที่caribeไว้ ตั้งแต่แรก ให้เท่าๆกัน ตลอดการทดลอง



ในรูปด้านบนเกิดจากปัญหาค่าไม่ค่อนข้างกัน ทำให้วัดช่วงยาก และการibeทไปก็ไม่ก่อประโยชน์ใดๆ จึงแก้ไขด้วยวิธีการcaribeทตามstanard จริงๆ

5. ปัญหาที่ห้า – การcaribeท(Callibration) ปัญหานี้กว่าจะแก้ได้

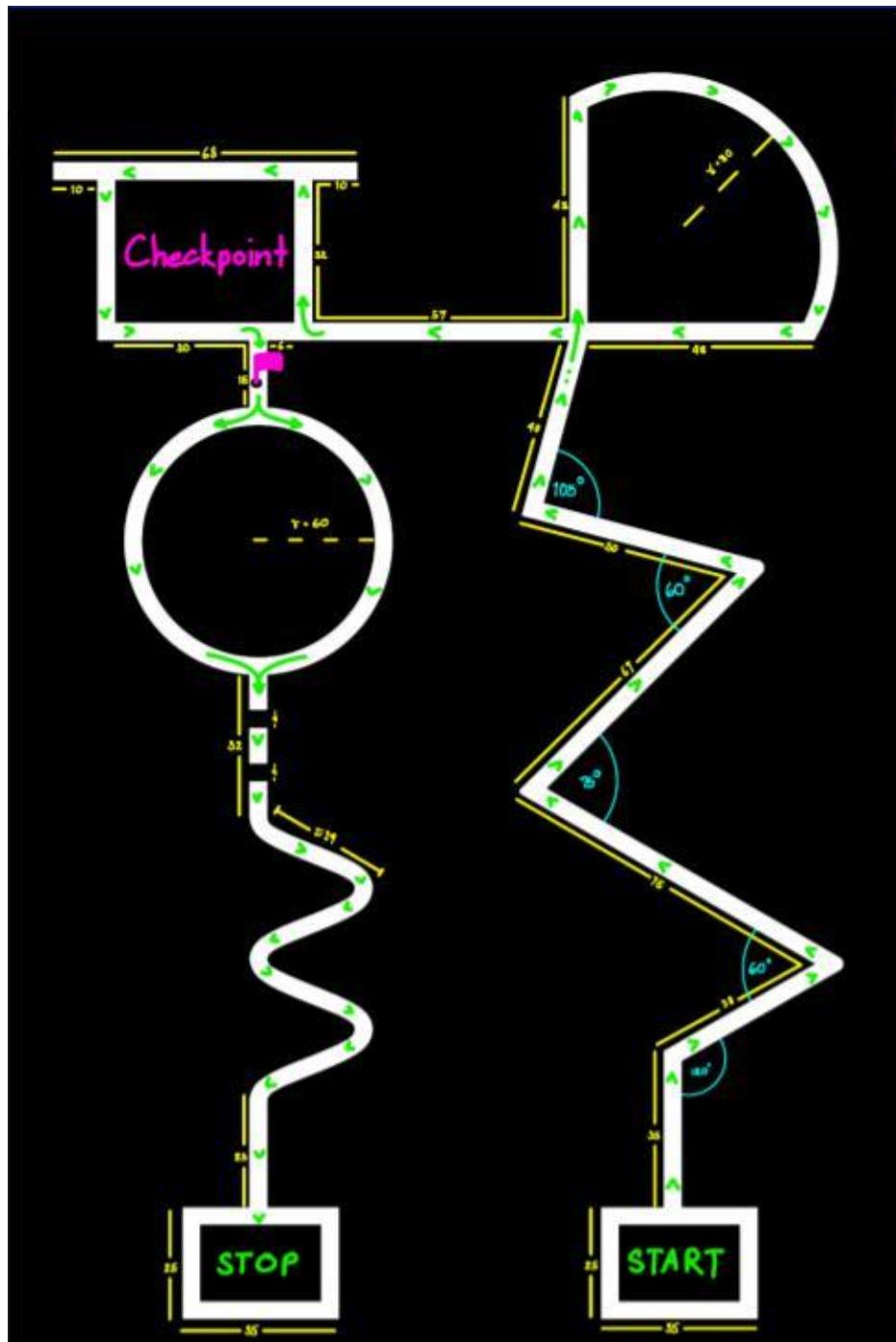
นานอยู่พอกว่า เพราะเวลา caribeทค่าจริงในคอมพิวเตอร์ กับเวลาลงสนาม จริงๆ เมื่อถอดสาย USB ออกจะพบได้ว่า ค่าcaribeทเพียงไป ตอนแรกคิดว่า เป็นที่เซนเซอร์เสีย หรือติดตั้งตำแหน่งผิด แต่มีจุด ๆ หนึ่งที่สังเกตได้ชัดเจนเลย ว่า เมื่อเสียบสาย USB ค่าที่caribeทตรงกับค่าความจริงทุกอย่าง พอกถอดออก ค่ากับผิดเพียงไป เราจึงเริ่มรู้แล้วว่า ปัญหาน่าจะเกิดมาจากการไฟของถ่านและไฟ จากคอมพิวเตอร์ที่มีผลทำให้การอ่านค่าเพียงไปหมด รวมถึงการต่อเซนเซอร์แบบ 3V หรือ 5V ด้วย ช่วงแรก มีปัญหามาก เพราะต่อ 5V ตาม datasheet ที่อาจารย์แจกให้ ทำให้เวลาจะcaribeท ถ้าอยากให้ตรงค่ามากที่สุดต้องเปิดไฟถ่านไว้ แต่เมื่อถอดออก ก็ไม่ตรงอยู่ดี เลยไม่มีประโยชน์อะไร เราจึงลองมาเปลี่ยนมาใช้ 3V เลี้ยงเซนเซอร์ทุกตัว ปรากฏว่า เวลา caribeท ถ้าใช้ 3V ไม่

จำเป็นต้องเปิดสวิตซ์ถ่าน ทำให้ค่ามีความตรงกับความเป็นจริงมากขึ้น แต่ก็ยังไม่ดีพอก เพราะเมื่อถอดสาย **USB** ค่าก็เปลี่ยนอีก ไม่ได้ช่วยอะไร

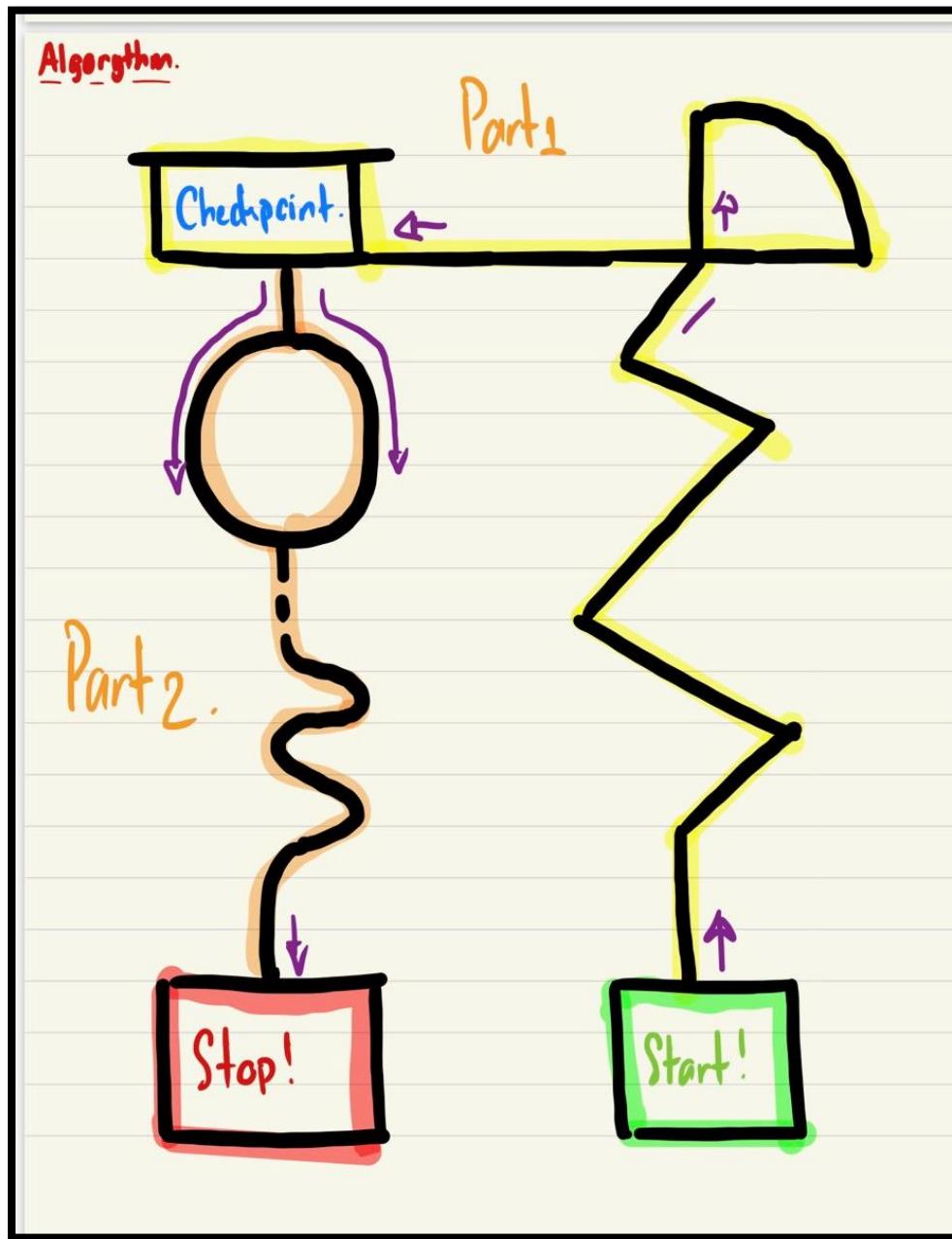
การแกะปัญหาของกลุ่มผู้ใช้ เป็นการที่ ยึดการจ่ายไฟแบบ **3V** ที่ใกล้เคียงที่สุด และอ่านค่าจาก **USB** และทำการคาริเบทค่าที่อ่านได้ภายในคอมไปก่อน เมื่อถอดสาย และทดลองจริง ให้สังเกตหลอดไฟที่กลุ่มพวงผู้ใช้ได้ติดตั้งไว้ ถ้าปกติ ก็ถือว่าดี แต่ถ้ามีอันไหนผิดปกติ ก็ให้ปรับแก้ตามความเป็นจริง

โดยการคาริเบทเบื้องต้นของผู้ใช้ ถ้าเกิดเกินค่า เท่านี้ จะเป็นสีขาวไฟจะติด ถ้าต่ำกว่านี้ ก็จะเป็นสีดำ หลอดไฟ LED ก็จะไม่ติด จึงสามารถคาริเบทปรับค่าได้จากการทดลองจริง เพิ่มขึ้น หรือลดลงเล็กน้อย ส่วนใหญ่จะอยู่ +- ไม่เกิน **100-200** ต่อเซนเซอร์ และเซนเซอร์ทุกตัว ค่าก็ไม่เท่ากันอีก ก็ต้องคงมานั่งคาริเบทแต่ละตัว แต่เมื่อทำสำเร็จแล้ว ใช้ได้ผลมาก ๆ กับการแทรคเส้นหลอดไฟ LED ติดถูกต้องตามความเป็นจริงอย่างที่ควรจะเป็นทุกดวง

แผนที่สนาม ROBOT CAR



ขั้นตอนการคิด ALGORITHM LOGIC



แมพส่วนนี้ กลุ่มผู้ใช้แบ่งออกเป็น 2 ส่วนหลัก ๆ ก็คือ

1. ส่วนแรก - การเดินทางจาก Start -> Checkpoint

2. ส่วนที่สอง - การเดินทางจาก Checkpoint -> Stop

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

PROCESS PART I Start -> Checkpoint

PART 1.1

```
RobotCar §
1 // motor one
2 #define lt_L2 2
3 #define lt_L1 3
4 #define lt_M 4
5 #define lt_R1 12
6 #define lt_R2 13
7 #define button 11
8
9 #define ss_L2 A0
10 #define ss_L1 A1
11 #define ss_M A2
12 #define ss_R1 A3
13 #define ss_R2 A4
14
15 #define enA 5
16 #define in1 6
17 #define in2 7
18
19 #define enB 10
20 #define in3 9    //Swap port in Program because hardware motor are swap
21 #define in4 8
22
23 float sensor_L2 = 0.f, sensor_L1 = 0.f, sensor_M = 0.f, sensor_R1 = 0.f, sensor_R2 = 0.f;
24 bool light_L2 = LOW, light_L1 = LOW, light_M = LOW, light_R1 = LOW, light_R2 = LOW;
25
26 int speed_L = 90;
27 int speed_R = 70;
28
29 int state = 1;
30 int stateLeft = 0;
```

1. โค้ดส่วนแรกจะเป็นส่วนของการต่อสายจ้มเข้า PIN ต่างๆ โดยจะมี PIN LED 5 ดวง PIN Sensor 5 ดวง สายจ้มสำหรับ motor ของตัวรถ กำหนด speed Left wheel = 90 หน่วย และ speed Right Wheel = 70 หน่วย ที่ไม่เท่ากัน เพราะเกิดจากการทดลอง เมื่อทำให้ความเร็วเท่ากัน พบร่วงทั้งสองข้างหมุนเร็วช้าต่างกัน เราจึงพยายามหาค่าความเร็วที่ทำให้รถสามารถแล่นได้ตรงที่สุด ก็คือล้อซ้ายต้องซ้ายกว่าล้อขวาอยู่ 20 หน่วย

PART 1.2

```
32 void setup(){
33
34     Serial.begin(9600);
35
36     // set arduino pins output to motor pins
37     pinMode(enA, OUTPUT);
38     pinMode(in1, OUTPUT);
39     pinMode(in2, OUTPUT);
40
41     pinMode(enB, OUTPUT);
42     pinMode(in3, OUTPUT);
43     pinMode(in4, OUTPUT);
44
45     pinMode(lt_L2, OUTPUT);
46     pinMode(lt_L1, OUTPUT);
47     pinMode(lt_M, OUTPUT);
48     pinMode(lt_R1, OUTPUT);
49     pinMode(lt_R2, OUTPUT);
50
51     pinMode(ss_L2, INPUT);
52     pinMode(ss_L1, INPUT);
53     pinMode(ss_M, INPUT);
54     pinMode(ss_R1, INPUT);
55     pinMode(ss_R2, INPUT);
56
57     pinMode(button, INPUT_PULLUP);
58
59 }
```

2. โค้ดส่วน setup/output/input

โดยเราจะให้สายของ motor LED เป็น **OUTPUT** ส่วนสายของ sensor(TCRT5000) เป็น **INPUT**

```

67 ////////////////////////////////////////////////////////////////// Check State //////////////////////////////////////////////////////////////////
68 void CheckState()
69 {
70     if(state == 1)
71     {
72         if(light_L2 == HIGH && light_L1 == HIGH && light_M == HIGH && light_R1 == HIGH && light_R2 == HIGH)
73         {
74             Forward(speed_L, speed_R);
75             delay(250);
76         }
77         else if((light_L1 == HIGH && light_L2 == HIGH && light_M == HIGH && light_R1 == LOW && light_R2 == LOW)
78         {
79             TurnLeft(0, speed_R+55);
80             stateLeft++;
81             delay(650);
82         }
83         else if(light_L2 == LOW && light_R2 == HIGH)
84         {
85             TurnRight(speed_L+40, 0);
86         }
87         else if(light_L2 == HIGH && light_R2 == LOW)
88         {
89             TurnLeft(speed_L-40, speed_R+40);
90         }
91         else if(light_L1 == LOW && light_R1 == HIGH)
92         {
93             TurnRight(speed_L+40, speed_R-40);
94
95         }
96         else if(light_L1 == HIGH && light_R1 == LOW)
97         {
98             TurnLeft(speed_L-40, speed_R+40);
99         }
100        else if(light_R2 == HIGH && light_R1 == HIGH)
101        {
102            TurnRight(speed_L+60, speed_R-40);
103        }
104        else if(light_L2 == LOW && light_L1 == HIGH && light_M == HIGH && light_R1 == HIGH && light_R2 == HIGH)
105        {
106            TurnLeft(speed_L-40, speed_R+60);
107        }
108        else if (light_M == HIGH)
109        {
110            Forward(speed_L, speed_R);
111        }
112        else
113        {
114            TurnRight(speed_L+40, 0);
115        }
116    }
117 }

```

3. โค้ดส่วน tracking line

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

```

142 ////////////////////////////////////////////////////////////////// Light Debugging Output //////////////////////////////////////////////////////////////////
143 void LightOutput()
144 {
145 //***** L2 *****/*
146     if(sensor_L2 > (500)) { light_L2 = HIGH; } else { light_L2 = LOW; }
147
148 //***** L1 ****/*
149     if(sensor_L1 > (300+450)/2.f) { light_L1 = HIGH; } else { light_L1 = LOW; }
150
151 //***** M ****/*
152     if(sensor_M > (390)) { light_M = HIGH; } else { light_M = LOW; }
153
154 //***** R1 ****/*
155     if(sensor_R1 > (300+500)/2.f) { light_R1 = HIGH; } else { light_R1 = LOW; }
156
157 //***** R2 ****/*
158     if(sensor_R2 > (450)) { light_R2 = HIGH; } else { light_R2 = LOW; }
159
160     digitalWrite(lt_L2,light_L2);
161     digitalWrite(lt_L1,light_L1);
162     digitalWrite(lt_M,light_M);
163     digitalWrite(lt_R1,light_R1);
164     digitalWrite(lt_R2,light_R2);
165 }

```

4.โค้ดส่วน Calibrate

เราจะทำการวัดช่วงและกำหนดค่าให้กับช่วงของสีดำและสีขาว โดยปกติทั่วไปแล้ว สีดำจะมีค่าน้อยกว่าสีขาว เราจึงใช้สูตรคำนวน

$$((\text{ช่วงสีดำมากที่สุด}) + (\text{ช่วงสีขาวน้อยที่สุด})) / 2$$

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

$$\text{DarkMax} = 270 , \text{BrightMin} = 640$$

$$\text{ช่วงกึ่งกลางคือ } (270+640) / 2 = 455$$

สรุป : **Dark < 455 < Light**

แต่ทั้งหมดนี้ คือในทางทฤษฎี ในทางปฏิบัติแล้ว ใช้ได้ยาก เพราะ...

1. เช่นเซอร์ ไม่ได้รับค่าที่ถูกต้องจริง ๆ อาจเกิดจากความคลาดเคลื่อนของหั้งตัวเซนเซอร์เอง หรือสภาพแวดล้อมอื่น ๆ เช่น สนาม หรือไฟ
2. ความคลาดเคลื่อนของคนวัด การวัดของแต่ละคน ก็จะไม่เท่ากันมาก เมื่อเอาไปเทียบกับสนามอีกที่ ค่าก็มีความต่าง น้อยไม่เท่ากัน สุดท้าย ก็ต้องการibeทใหม่ ซึ่งไปซื้ามาเรื่อยๆ
3. ค่าไฟของถ่านมีผลอย่างมาก เนื่องจากการอ่านค่าของเซนเซอร์ขึ้นอยู่กับค่าไฟของถ่านส่วนนึงด้วย ดังนั้น การที่เราการibeทดูค่าสด ๆ จากคอมพิวเตอร์ด้วยการเสียบสาย **usb** เป็นการทำให้ไฟไม่เท่ากับเวลาปล่อยรถแล่นจริง ๆ ทำไง เกิดค่าเพียง

วิธีแก้ก็คือ

ใช้ไฟให้เซนเซอร์หั้งหมวดเป็น **3.3V** ขณะการibe ห้ามเปิดแบต เสียบได้แค่สาย **USB** จะได้ค่าการibeทที่ค่อนข้างตรงกับถอดใช้จริงที่สุด แต่ก็ยังมีความคลาดเคลื่อน เราจึงใช้การวัดค่าตรง ๆ จาก **USB** ก่อน และจึงค่อย ๆ ปรับตามความเหมาะสมของใช้จริง เช่น ถ้าเกิดค่าในคอมเมื่อเสียบ **USB** ได้ค่า **300** (มากกว่านี้ให้ไฟติด) แต่ถ้าไปวางสีขาว และไฟยังดับอยู่ ให้ตั้งค่าให้ลดลงกว่านี้ (เพื่อให้ไฟติดในพื้นที่สีขาว) การทำแบบนี้จะเป็นการทำให้เซนเซอร์สามารถอ่านค่าได้ถูกต้องตามสภาพแวดล้อมจริง ทำให้สามารถแทรคเส้นได้อย่างไม่มีปัญหาความคลาดเคลื่อนมากนัก

```

167 ////////////////////////////////////////////////////////////////// Movement Functions //////////////////////////////////////////////////////////////////
168 //***** FORWARD *****
169 void Forward(int L , int R){
170     go_L();
171     go_R();
172     analogWrite(enA, L);
173     analogWrite(enB, R);
174 }
175
176 //***** TURN RIGHT *****
177 void TurnRight(int L , int R){
178     go_L();
179     go_R();
180     analogWrite(enA, L);
181     analogWrite(enB, R);
182 }
183
184 //***** TURN LEFT *****
185 void TurnLeft(int L , int R){
186     go_R();
187     go_L();
188     analogWrite(enA, L);
189     analogWrite(enB, R);
190 }

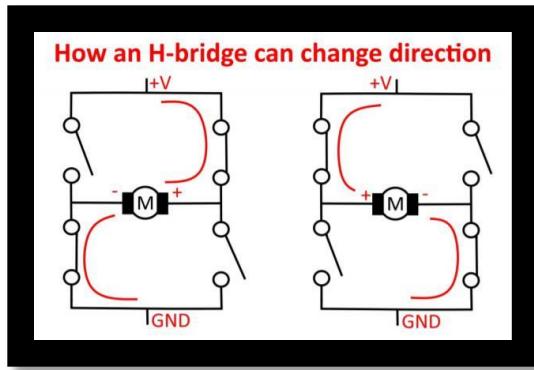
192 ////////////////////////////////////////////////////////////////// Wheel Elements //////////////////////////////////////////////////////////////////
193 void go_L(){
194     digitalWrite(in1, HIGH);      //Don't make it High both
195     digitalWrite(in2, LOW);
196 }
197 void go_R(){
198     digitalWrite(in3, HIGH);
199     digitalWrite(in4, LOW);
200 }

```

5.โค้ดส่วน DC MOTOR

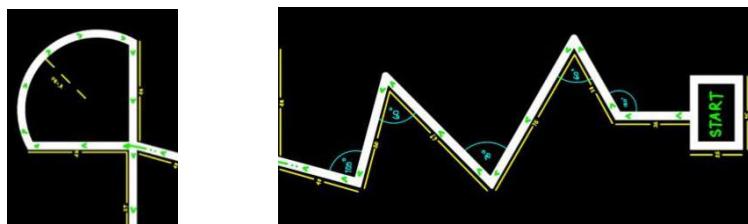
การทำงานของล้อรถมีอยู่ 2 พังก์ชั่น คือ **go_L()** และ **go_R()** **go_L()** จะทำงานแค่ล้อขวา(เพราเลี้ยวซ้าย) **go_R()** จะทำงานแค่ล้อซ้าย(เพราเลี้ยวขวา)

การที่เขียนพังก์ชั่นแบบนี้ ทำให้ง่ายต่อการเขียนโปรแกรม และความเข้าใจของคนที่อ่าน และกันปัญหาเรื่องการจ่ายไฟให้มอเตอร์ทั้งสองพร้อมกัน (กันระเบิดนั้นเอง) นี่เป็นสาเหตุที่เราเขียนพังก์ชั่นจากการทำงานเล็กๆ เช่น **go_R()**, **go_L()** และระบบพังก์ชั่นเล็กๆเป็นพังก์ชั่นใหญ่ๆอีกที เช่น **Forward()**, **Backward()** เป็นต้น



PROCESS PART II Checkpoint -> Stop

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



โค้ดส่วนสุดท้ายจะคล้าย ๆ กับส่วนแรก แต่เพิ่มแค่ **1** เคส นั่นก็คือ เมื่อ **L1 L2 M R1 R2** (หรือทุกเซนเซอร์) เป็น **LOW** จะให้รถทำพังก์ชัน **Forward();** (เดินหน้า) เพื่อที่จะผ่านตรงรออยู่ในสนามได้

