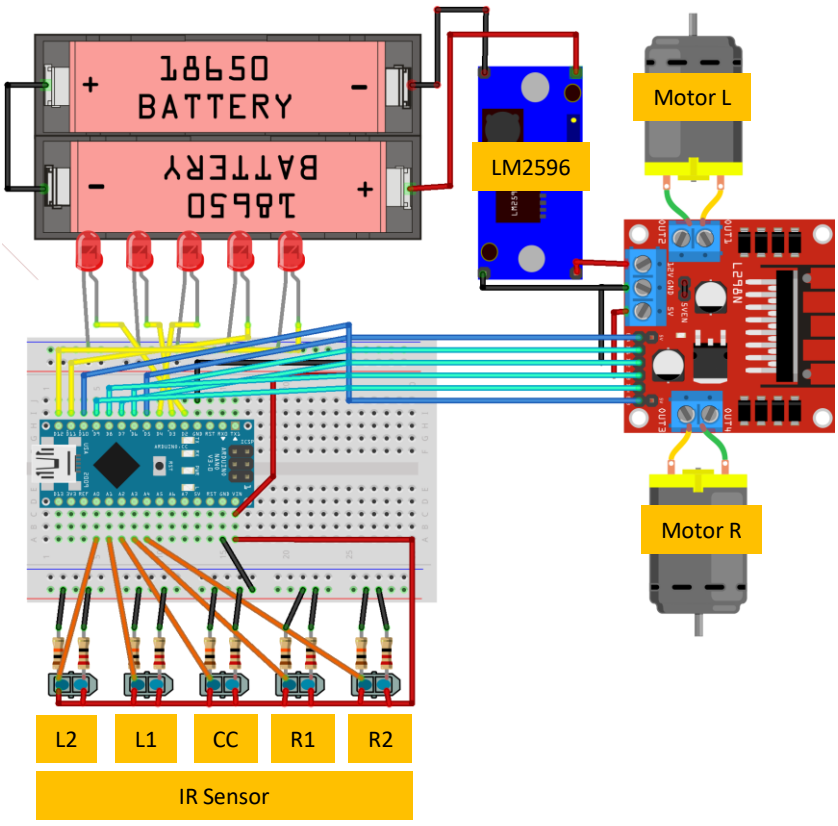
# Introduction to Computer Engineering

## Mini Project ##: Robot Car

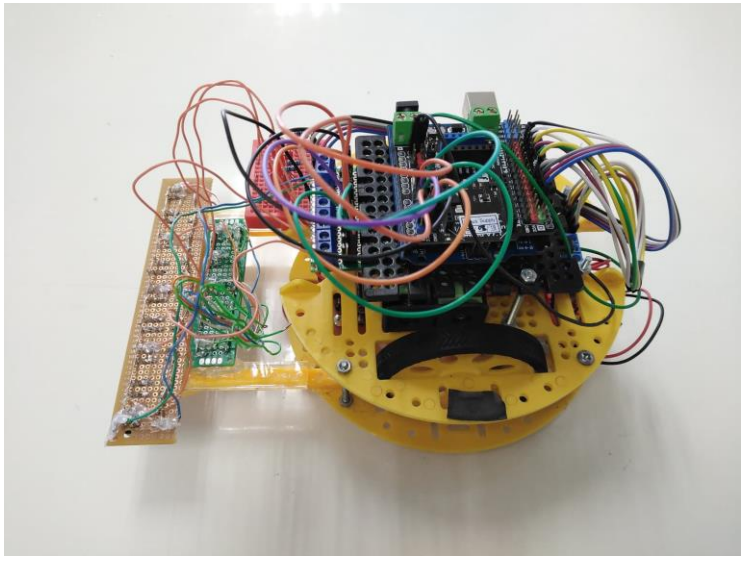
### รูปแบบการเชื่อมต่อ




### PIN MAP

| Device | Arduino |
|---|---|
| L2_LED | 2 |
| L1_LED | 3 |
| CC_LED | 4 |
| R1_LED | 11 |
| R2_LED | 12 |
| enA | 5 |
| in1 | 6 |
| in2 | 7 |
| enB | 10 |
| in3 | 8 |
| in4 | 9 |
| Sensor L2 | A4 |
| Sensor L1 | A3 |
| Sensor CC | A2 |
| Sensor R1 | A1 |
| Sensor R2 | A0 |

### รูปถ่ายชิ้นงาน



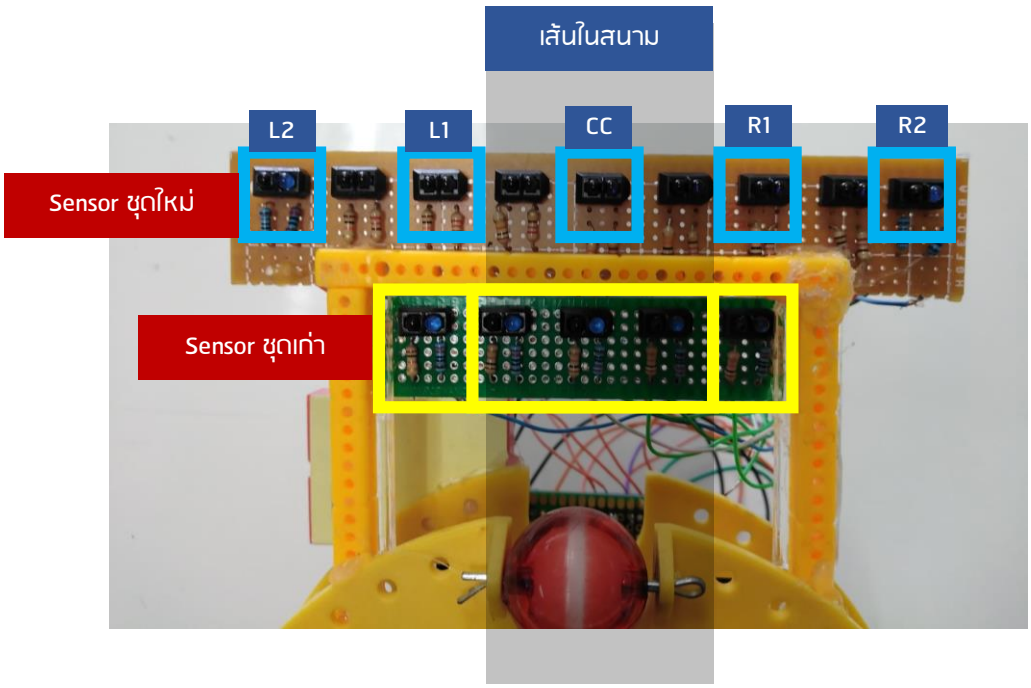
ทางกลุ่มของเรานั้นเลือกใช้ ชุดประกอบหุ่นยนต์สำเร็จรูป 2 ล้อ จาก inex ซึ่งประกอบไปด้วยแผ่นฐาน จำนวน 2 แผ่น , ชุดล้อพร้อมมอเตอร์แบบทดเกียร์ จำนวน 2 ชุด , น็อตพร้อมอุปกรณ์เสริม และ Arduino NANO Rev.3 จำนวน 1 บอร์ด มาใช้งานและมีส่วนต่อขยายเพิ่มเติมคือ หลอด LED (สีแดง) จำนวน 5 หลอด, เซ็นเซอร์แสงชนิดอินฟราเรด TCRT5000 จำนวน 5 ตัว โดยได้ทำการเชื่อมต่อกันในลักษณะดังภาพด้านข้าง

# Introduction to Computer Engineering
## Mini Project ##: Robot Car

### แนวคิดการออกแบบ (Conceptual Design)

Robot Car สำหรับทางกลุ่มของข้าพเจ้านั้นได้ทำการศึกษาและได้ข้อสรุปเกี่ยวกับการเคลื่อนที่ของหุ่นยนต์ให้บรรลุตามวัตถุประสงค์ของโจทย์ที่ให้มา โดยได้ทดลองทำการศึกษาการเคลื่อนที่แบบต่างๆ หรือการทำงานแบบ multitasking เทียมบน FreeRTOS ที่ทางกลุ่มได้ลองนำมาใช้ร่วมกันกับ PID Control แต่ผลปรากฏไม่เป็นที่พึงพอใจนักเลยลองให้หุ่นยนต์ได้ทำการเคลื่อนที่แบบ PID แบบ task เดียวใน Loop แต่ผลลัพธ์ที่ได้ก็ไม่เป็นไปตามที่คาดหวังไว้สักเท่าไหร่ เหตุเพราะในสนามการแข่งขันจะมีบางจุดหรือบางมุมที่มีการโค้งมากจนเกินไปทำให้ต้องเพิ่มค่า Kp จึงทำให้หลุดโค้งได้ง่าย อีกทั้งเงื่อนไขของ PID ในบางจุด เช่น จุดตัด ส่วนเส้นเกิน จะแยกค่อนข้างยาก โดย PID Control นั้นส่วนใหญ่นิยมมาเดินในเส้นทางตรงหรือ Single Line กันมากกว่า แต่สนามนี้มีความท้าทายพอสมควรทางกลุ่มจึงไม่เลือกการเคลื่อนที่แบบ PID Control โดยทางกลุ่มมีแนวคิดในการติดตั้งชุด Sensor ดังนี้

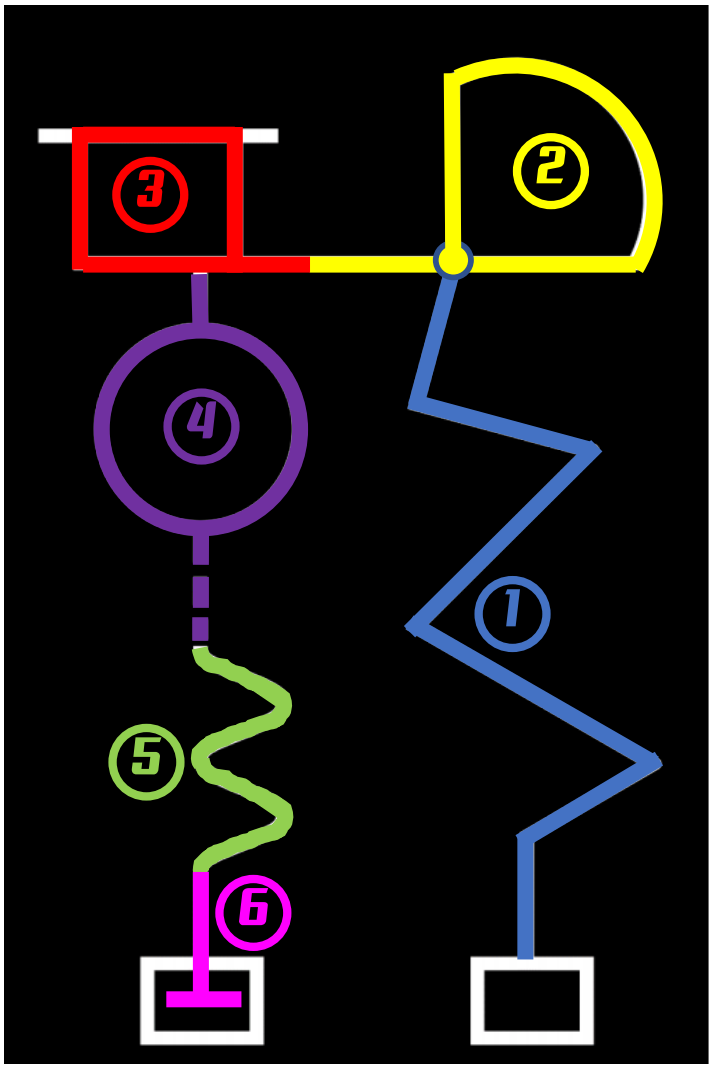
#### การติดตั้ง Sensor




ทางกลุ่มของข้าพเจ้าได้ทำการทดลองโดยเริ่มแรกได้ทำการติดตั้ง Sensor ชุด แรก (Sensor ชุดเก่า) เข้ากับตัวหุ่นยนต์ และลองใช้รูปแบบการเดิน PID ดู แต่ผลลัพธ์ที่ออกมาไม่ดีเพราะติด Sensor ใกล้กันเกินไปทำให้เวลาไปคิด Error Case ใน PID มีน้อยจึงหลุดไปในบ้าง case และทางกลุ่มจึงคิดเห็นว่าควรทำรูปแบบการเดินขึ้นมาเองโดยอ้างอิงจาก Sensor เลยได้ทำการติดตั้งชุด sensor เข้าไปอีกชุดนึง (Sensor ชุดใหม่) จำนวน 9 ตัว แต่ใช้งานจริงเพียง 5 ตัวเท่านั้นลงไป โดยกำหนดให้มี sensor 1 ตัว อยู่บริเวณกลางเส้น อีก 2 ตัว อยู่นอกเส้นทั้งซ้ายและขวาสำหรับการเดินตามเส้น และอีก 2 ตัวอยู่กรอบนอกสุดสำหรับการเดินและตรวจเส้นตัด

# Introduction to Computer Engineering
## Mini Project ##: Robot Car

### อัลกอริทึมในการเดินใน Map




Robot Car ของทางกลุ่มข้าพเจ้านั้นไม่ได้ใช้ PID Control ในการเคลื่อนที่แต่อย่างใด ใช้เพียงการเดินตามเส้นปกติที่ใช้ sensor เพียง 3 ตัว (L1,CC,R1) และเวลาในการกำหนด โดยหลักการคือแบ่งเส้นทางและการทำงานของชุดโปรแกรมออกเป็น 6 ส่วนใหญ่ ตามแต่ละอุปสรรค ดังรูปด้านข้างนี้

**ส่วนที่ 1** ตั้งแต่จุดเริ่มต้นผ่านเดินตามเส้นผ่านมุมต่างๆ และถึง 4 แยก

**ส่วนที่ 2** ตั้งแต่ 4 แยก เดินตรงไปผ่านเส้นโค้งครึ่งวงกลมและตรงไปสักระยะหนึ่ง

**ส่วนที่ 3** จะเป็นการเดินวนรอบเป็น 4 เหลี่ยมเพื่อเก็บเวลาและจุด Check Point

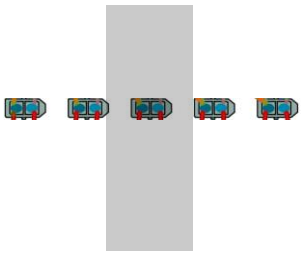
**ส่วนที่ 4** จะเป็นการเคลื่อนที่ออกจากจุด Check point เข้าไปในวงกลมจนออกจากวงกลมผ่านเส้นประ

**ส่วนที่ 5** จะเป็นการเคลื่อนที่ไปตามทางโค้งรูป sine wave

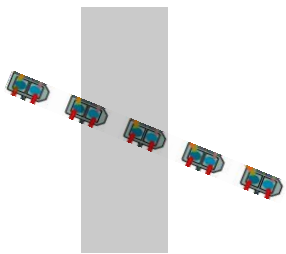
**ส่วนที่ 6** เป็นส่วนสุดท้ายคือหลุดจากเส้น sine wave เป็นทางตรงเข้าสู่จุด Stop

โดยทางกลุ่มจะแบ่งโปรแกรมเป็น 2 ส่วนใหญ่ คือ
1. โปรแกรมจากจุด Start ไปจุด Check Point
2. โปรแกรมจากจุด Check Point ไปยังจุด Stop

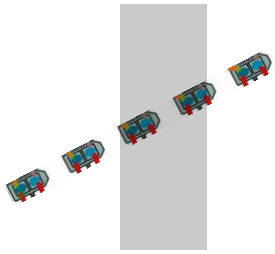
โดยส่วนใหญ่ก็ใช้ Map นี้จะเป็นการเดินตามเส้นแบบปกติไม่ได้ใช้อะไรควบคุมมากมายโดยรูปแบบการเคลื่อนที่ของหุ่นยนต์เป็นตามนี้ คือ



เดินตรงเมื่อ sensor CC เจอเส้นดำและนอกนั้นขาวหมด



เลี้ยวซ้ายเมื่อ sensor L1 หรือ L2 เจอเส้นดำและนอกนั้นขาวหมด

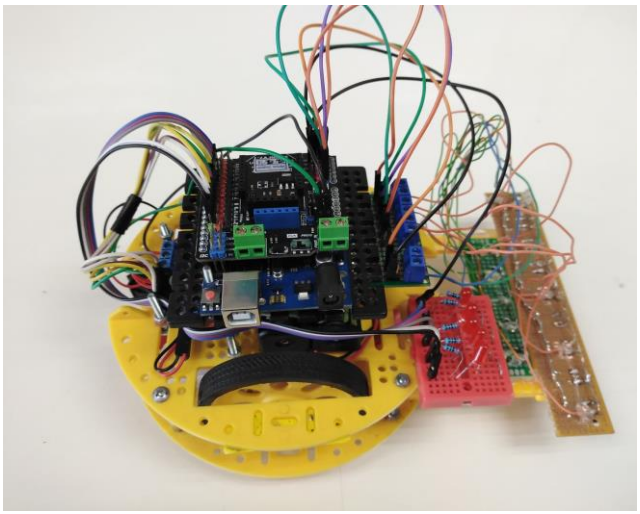


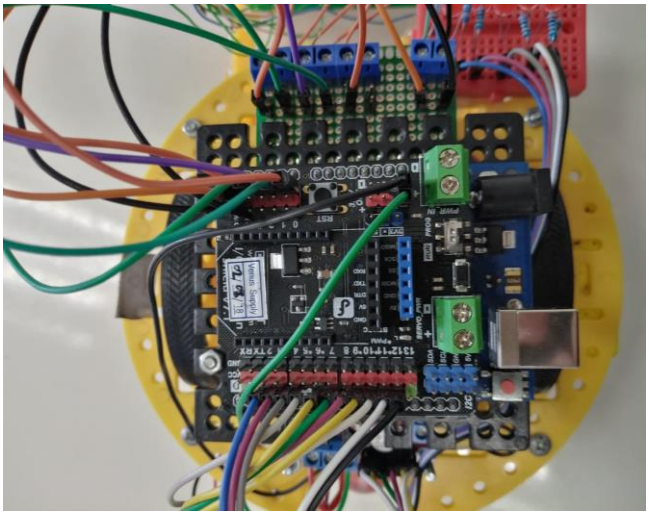
เลี้ยวขวาเมื่อ sensor R1 หรือ R2 เจอเส้นดำและนอกนั้นขาวหมด

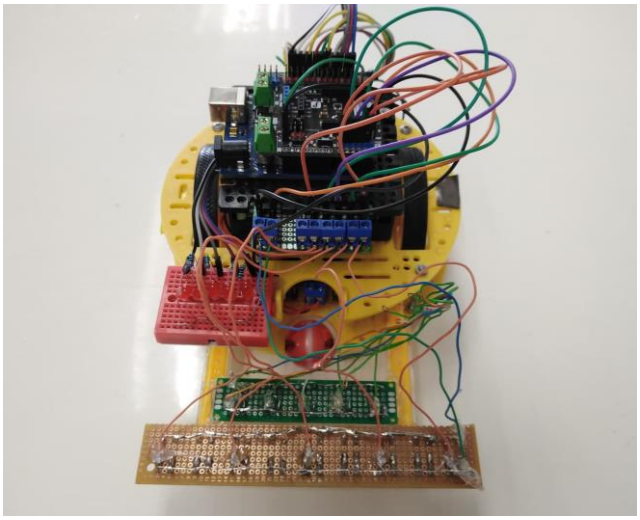
หยุดการเคลื่อนที่เมื่อ sensor ทุกตัวเจอเส้นดำและ

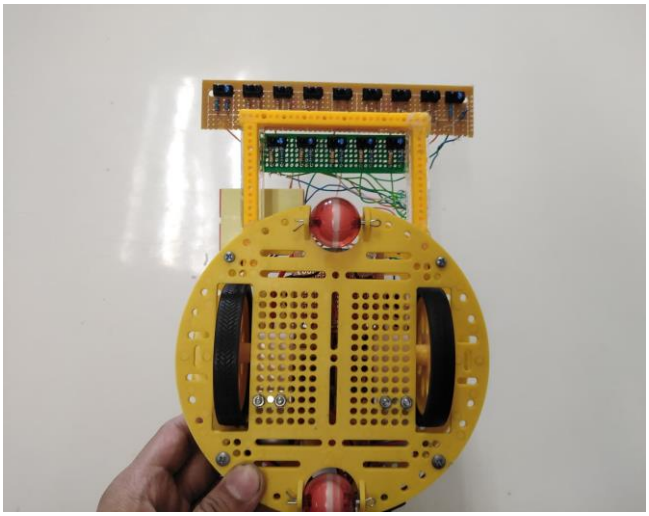
# Introduction to Computer Engineering
## Mini Project ##: Robot Car

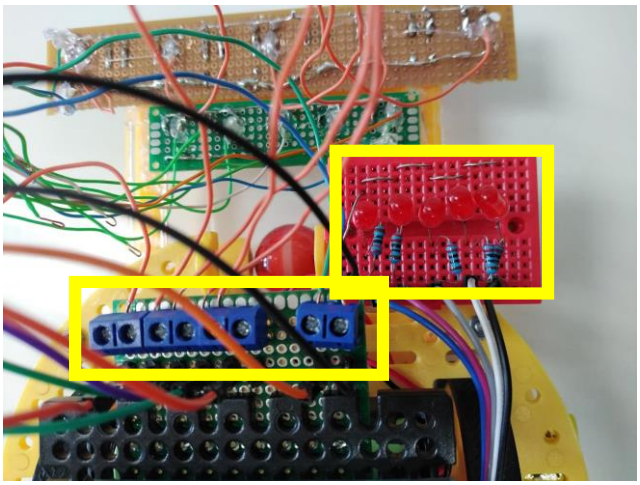
รูปถ่ายและความสร้างสรรค์











1. ใช้บอร์ดเพื่อแปลงชุด Sensor ให้ถอดแบบ Modular ทำให้ง่ายและสะดวกในการเปลี่ยนหรือปรับแต่งชุด Sensor
2. ในโปรแกรมในส่วนของการเคลื่อนที่ไม่ได้ใช้คำสั่ง delay(); เลยทำให้หุ่นยนต์เคลื่อนที่ด้วยความเร็วเกือบสูงสุด
3. ใช้ Regulator step down เพื่อรักษาแรงดันให้คงที่ไม่ให้กระชากหรือตกเวลามี load
4. ออกแบบหุ่นยนต์โดยมีจุด CG ที่กึ่งกลางตัวหุ่นและน้ำหนักถ่วง (แบตเตอร์รี่ 2 ก้อน) ที่จุดศูนย์กลางทำให้เวลาเลี้ยวและเคลื่อนที่มีความสมดุลขึ้น
5. มีไฟ LED แสดงผลเวลา sensor ตรวจจับเจอสีดำ

# Introduction to Computer Engineering
## Mini Project ##: Robot Car

### โปรแกรมและคำอธิบาย

โปรแกรมที่ใช้มี 2 ไฟล์ คือ Go และ Go2 โดย

Go คือ [STATE-1/2]: Start -> Check Point

Go2 คือ [STATE-2/2]: Check Point -> Stop

```
/*=========================================
    [Final Project]- Autonomous Robot car
            by Raweeroj & Nithi
     Introduction to Computer Engineering
              Date: 25/11/62
      [STATE-1/2]: Start -> Check Point
  =========================================*/
/*================= Define ================
     L2  L1 CC R1  R2
    [0] [1][2][3] [4]
     |___|__|__|___|
    _/             \_
   |6| In1      In3 |8|
   |7| In2      In4 |9|
   |_|  5       10  |_|
     \_____________/

  LED[2]  [3][4][11]  [12]
  =========================================*/
```

ส่วนนี้เป็นการ Comment และ Define รายละเอียดภายในโปรแกรมโดยระบุถึง pin ต่างๆ ที่ใช้ในหุ่นยนต์

```
/*Left Wheel*/
#define enA 5
#define in1 6
#define in2 7

/*Right Wheel*/
#define enB 10
#define in3 8
#define in4 9

/*Right Wheel*/
#define L2_LED 2
#define L1_LED 3
#define CC_LED 4
#define R1_LED 11
#define R2_LED 12

```

```
/*============== Config-Value ================*/
int Mid_L2 = 834;
int Mid_L1 = 305;
int Mid_CC = 309;
int Mid_R1 = 325;
int Mid_R2 = 334;

int B[5], W[5];

/*====== Speed Control =====*/
float init_motor_speed = 200;
/*==========================================*/
/*================= State ==================*/
bool L2 = false;
bool L1 = false;
bool CC = false;
bool R1 = false;
bool R2 = false;

bool isForward = false;
bool isRight = false;
bool isLeft = false;
int state = 0;

/*==========================================*/
```

ส่วนนี้เป็นการกำหนดค่าตัวแปรเริ่มต้นและสถานะได้แก่ ค่ากลางของ sensor ระหว่างสีดำกับสีขาว , ความเร็วเริ่มต้นของมอเตอร์ เป็นต้น

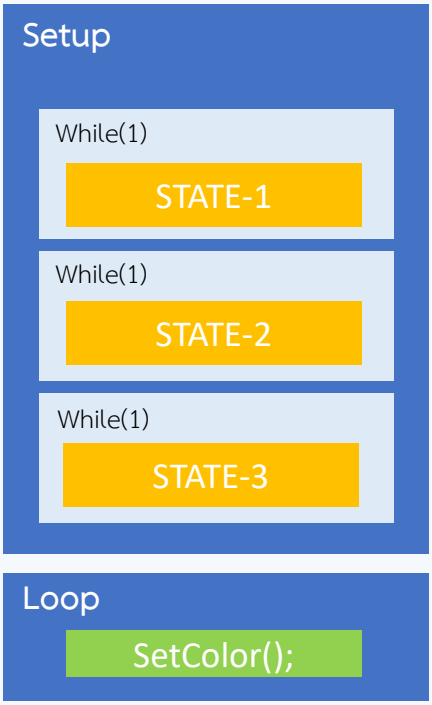
# Introduction to Computer Engineering

## Mini Project ##: Robot Car

โปรแกรมและคำอธิบาย

หมายเหตุ * Comment เพิ่มเติมในไฟล์ source code

## GO



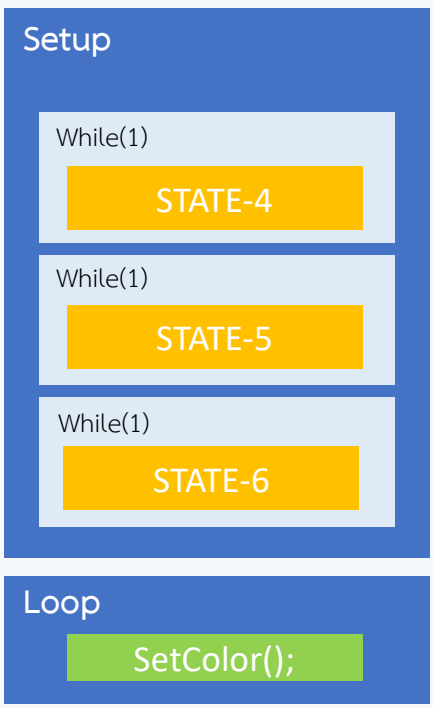

โดยในแต่ละ STATE จะเป็นโปรแกรมเดินตามเส้นแต่จะมีการกำหนด Time Counter และ Priority ของการเลี้ยวและความเร็วในการเคลื่อนที่แตกต่างกันไป กล่าวคือ บรรทัดบนจะทำก่อนซึ่งเป็นปกติของการเขียนโปรแกรมแบบ sequent เช่น เรากำหนดให้เจอสีดำของ sensor ขวาก่อน sensor ซ้าย หุ่นยนต์ก็จะเลี้ยวขวาก่อนซ้ายเป็นต้น และในขณะทำ STATE ใดอยู่ STATE นั้นจะสิ้นสุดก็ต่อเมื่อ เวลาหมดลง ก็จะ break(); ออกจาก while loop และเข้าไปยัง while loop ของ STATE ถัดไป

STATE-1 State 1 --> ทางโค้งเป็นมุมไปจนถึง 4 แยก (ซ้ายสำคัญ)

STATE-2 State 2 --> จาก 4 แยกผ่านโค้งครึ่งวงกลมไปทางตรงครึ่งนึง (ขวาสำคัญ)

STATE-3 State 3 --> จากกึ่งกลางเส้นตรงไปวนรอบสี่เหลี่ยมและหยุดที่ Check Point (รู้จักแค่ขวา : sensor ซ้ายไม่มีผล)

## GO2




STATE-4 State 4 --> จากจุด Check Point ไปจนถึงเส้นประ (ซ้ายสำคัญ)

STATE-5 State 5 --> จากเส้นประผ่านโค้ง sine wave (ขวาสำคัญ)

STATE-6 State 6 --> เดินตรงๆ แล้วไปหยุดที่จุก Stop


**จัดทำโดย**

1. นายนิธิ น้อมประวัติ 62010497
2. นายรวีโรจน์ ทองดี 62010763

**กลุ่ม ...(3D@s)**