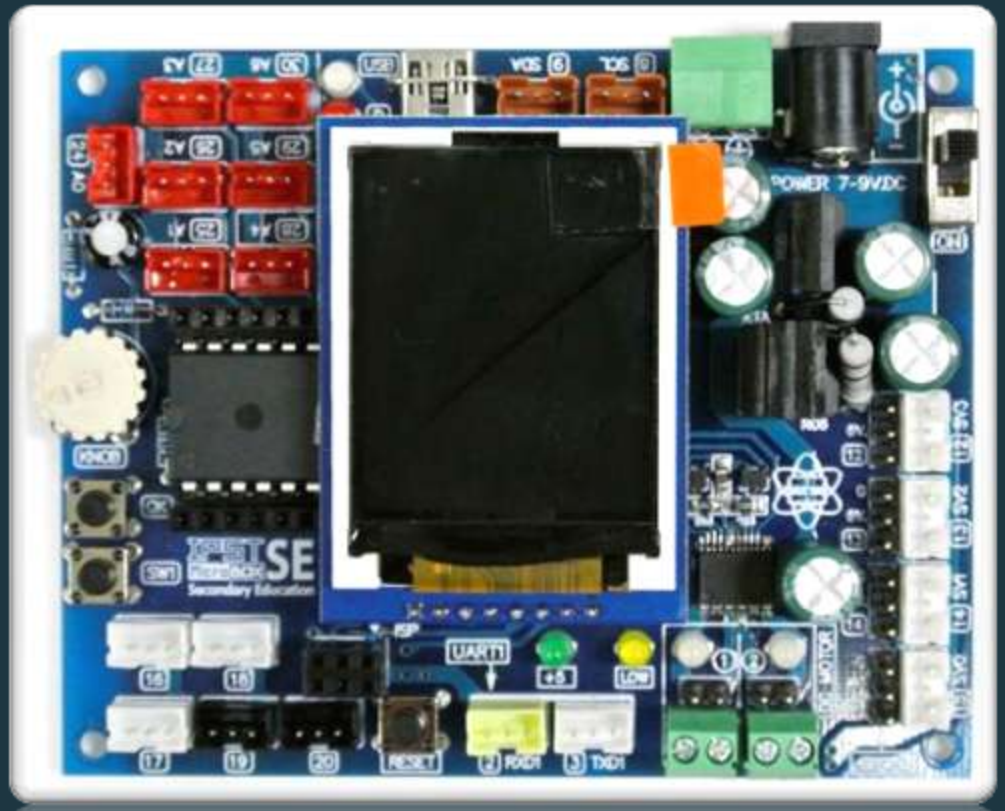
การใช้งาน

กล่องสมองกล



IPST MicroBox SE

สู่โลกไมโครคอนโทรลเลอร์

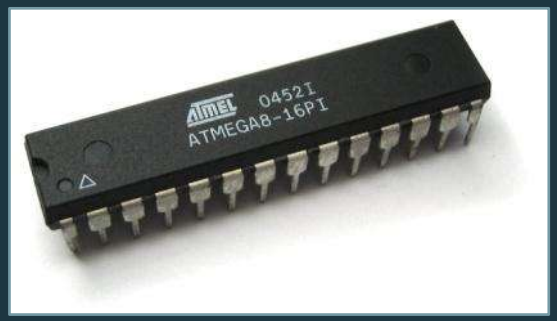
ไมโครโปรเซสเซอร์

ไมโครคอนโทรลเลอร์ที่มีใช้งานทั่ว ๆ ไป

ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ PIC

ไมโครคอนโทรลเลอร์ BASIC Stamp



ไมโครคอนโทรลเลอร์ AVR

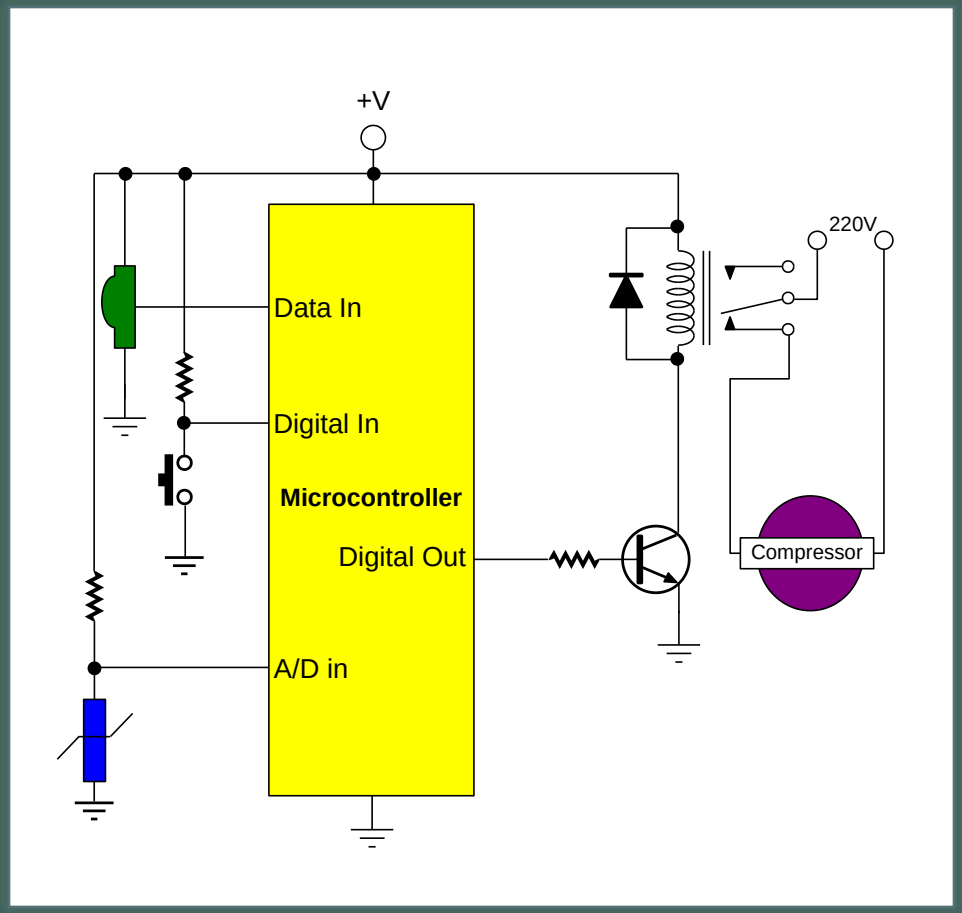
สู่โลกไมโครคอนโทรลเลอร์

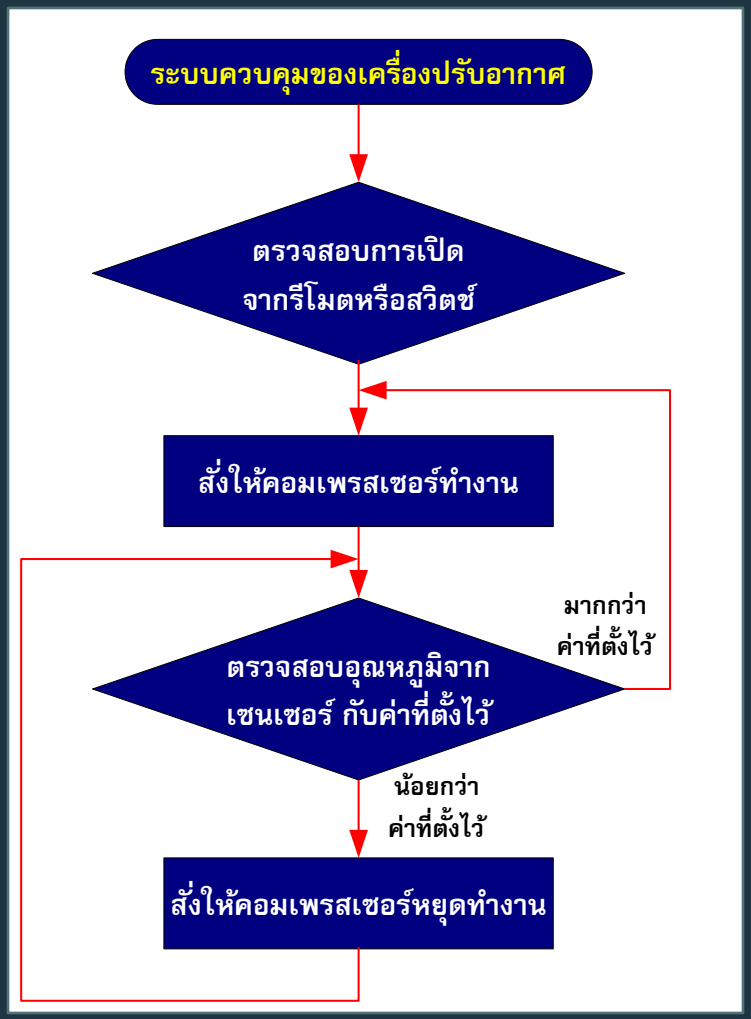
ไมโครคอนโทรลเลอร์อยู่รอบ ๆ ตัวเรา

หลักการของระบบควบคุม

ตัวอย่างระบบควบคุมอย่างง่าย ๆ

ตัวอย่างการใช้งานไมโครคอนโทรลเลอร์ในเครื่องปรับอากาศ





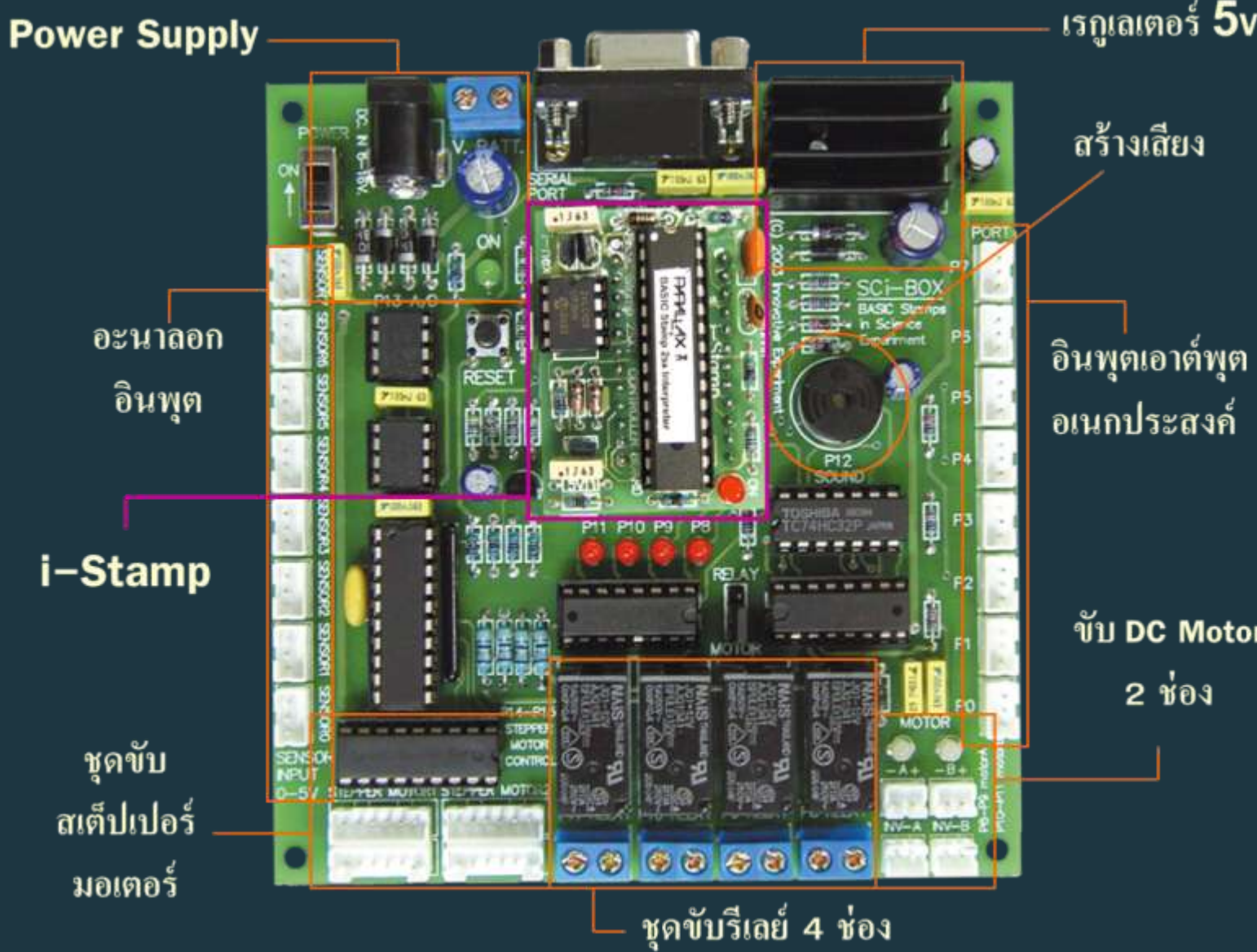
กว่าจะมาเป็น IPST MicroBOX SE

ชุดเรียนรู้การทดลองวิทยาศาสตร์กับกล่องสมองกล

SCI-BOX
Microcontroller in science experiment

กว่าจะมาเป็น IPST MicroBOX SE

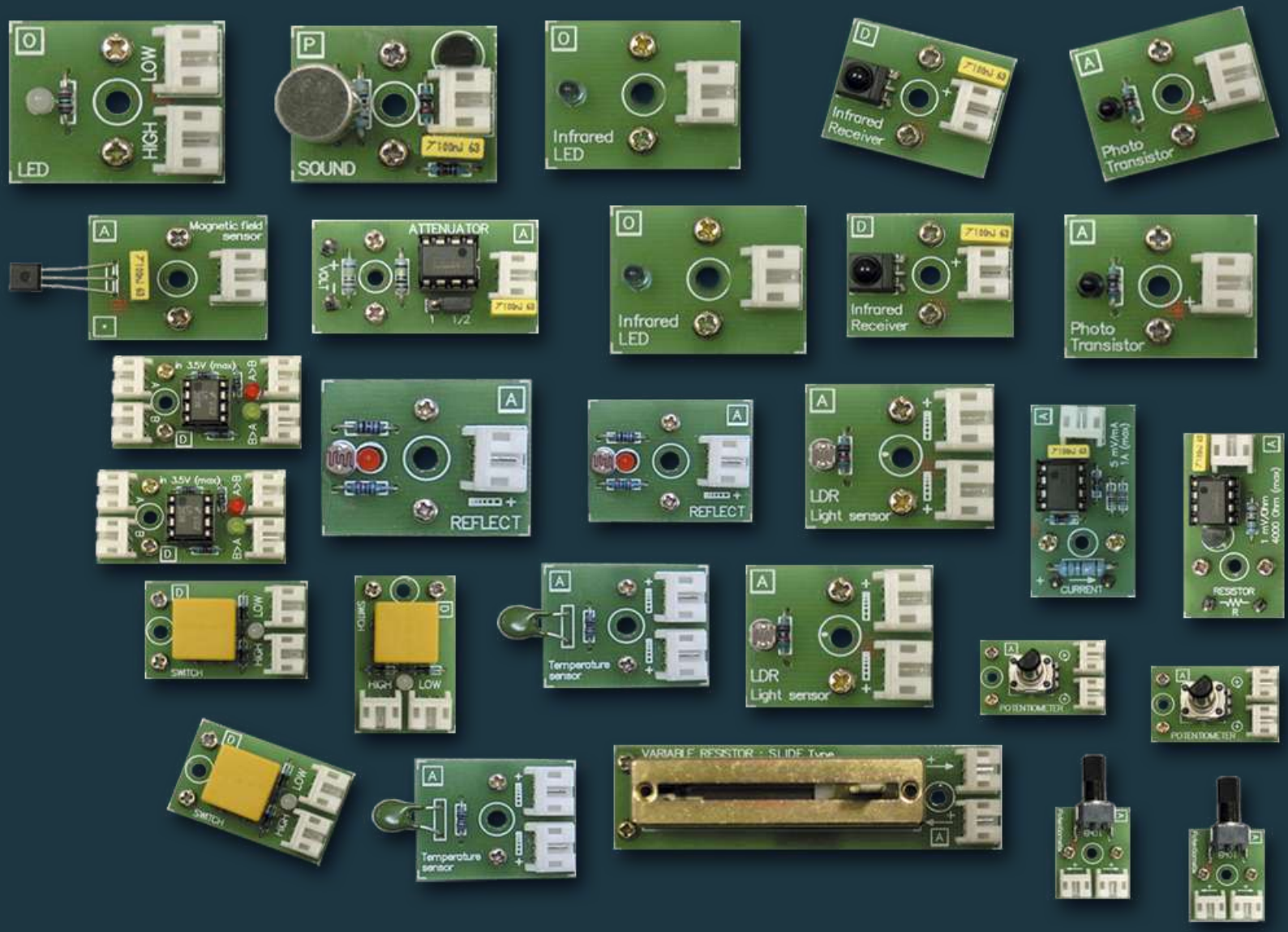
บอร์ดควบคุมหลัก



กว่าจะมาเป็น

IPST MicroBOX SE

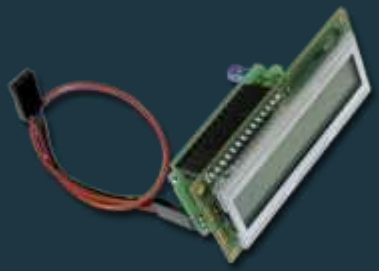
Sensor ในชุด SCI-BOX



กว่าจะมาเป็น IPST MicroBOX SE

อุปกรณ์ย่อยสำหรับการทดลอง

ปากคีบ



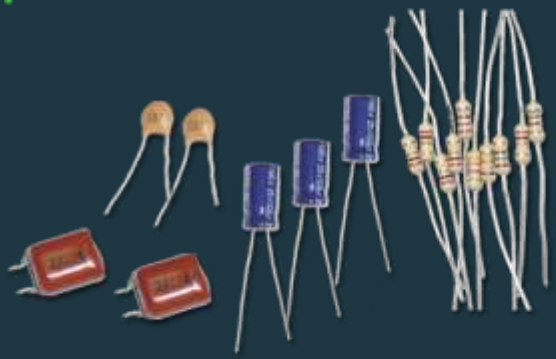
SERIAL LCD MODULE

สายเชื่อมต่อ Sensor

Adapter

Stepper Motor

DC Motor



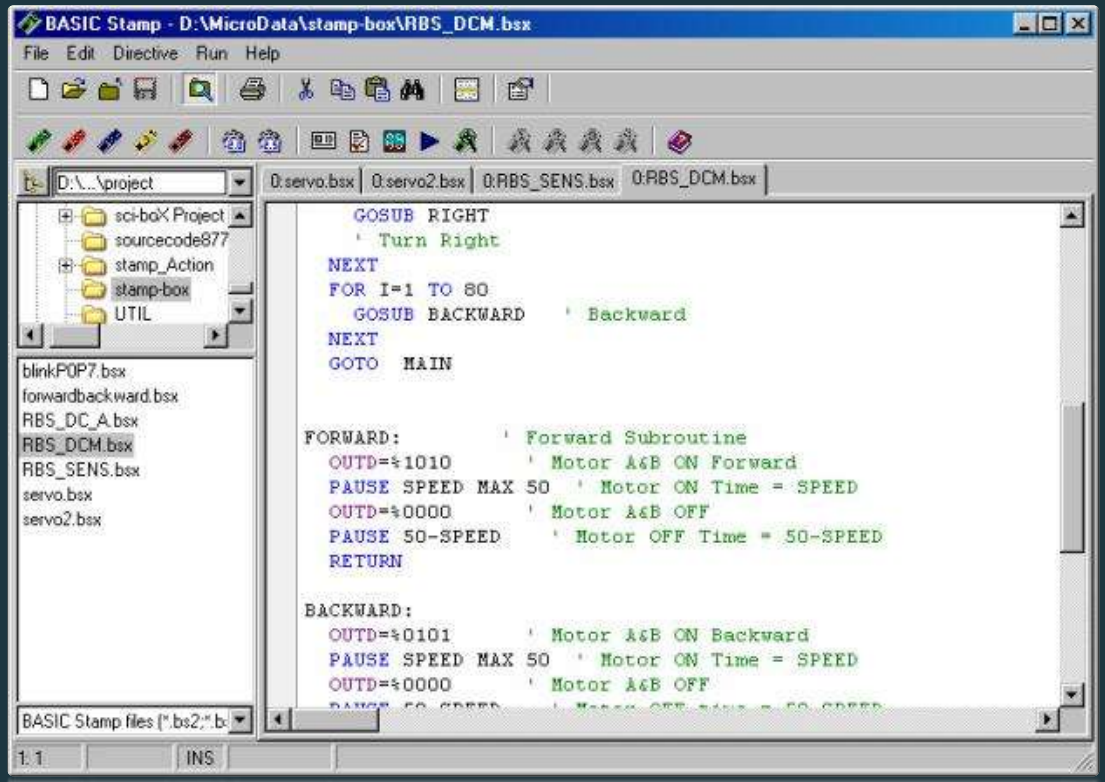
Electronics Component

Serial Cable

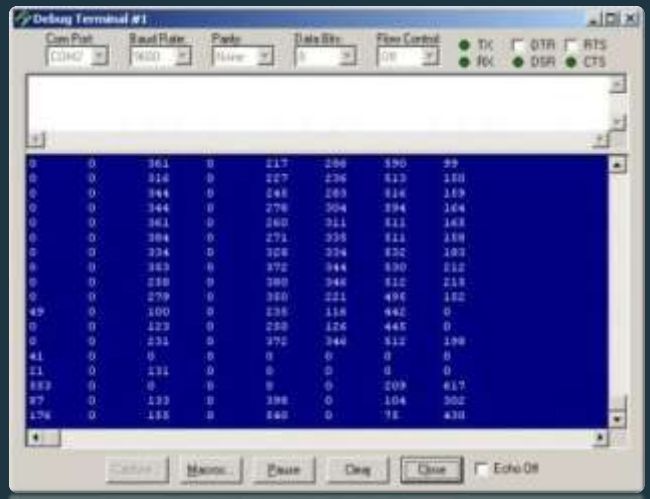
กว่าจะมาเป็น

IPST MicroBOX SE

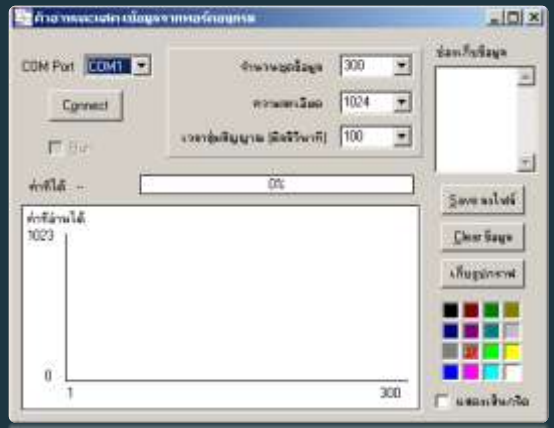
เขียนโปรแกรมด้วยภาษา BASIC



Basic Stamp Editor



Debug Terminal



Dual Data

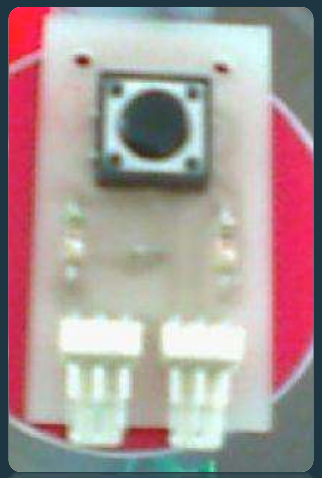
ข้อด้อยของ Sci-BOX

- การเขียนโปรแกรมด้วยภาษาเบสิก
- โมดูลไมโครคอนโทรลเลอร์ราคาสูง (i-Stamp)
- สร้างบอร์ดและอุปกรณ์ต่อพ่วงเองได้ยาก

กว่าจะมาเป็น IPST MicroBOX SE

ออกแบบฮาร์ดแวร์ใหม่จาก สสวท.

ต้นแบบรุ่นแรกจาก สสวท.

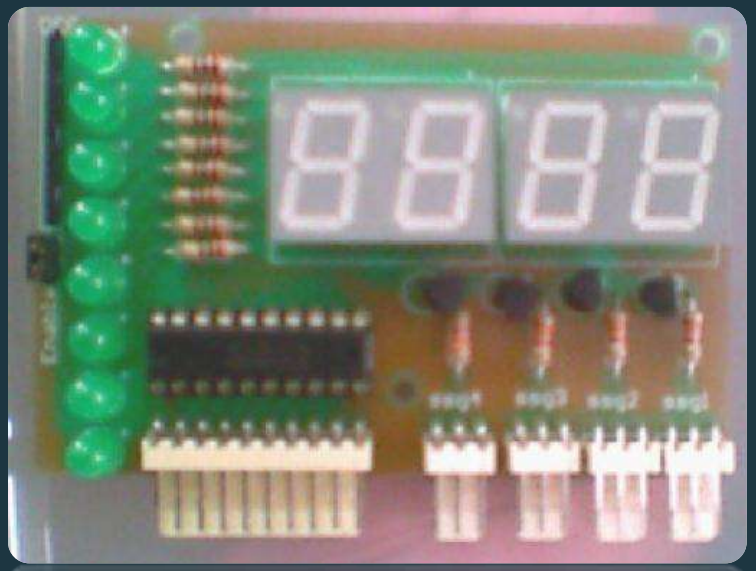


สวิตช์

LDR

ลำโพง

ตัวต้านทานปรับค่าได้



LED + 7 Segment

รีเลย์

ต้นแบบรุ่นแรกจาก สสวท.

กว่าจะมาเป็น IPST MicroBOX SE

พัฒนาต่อโดย inex

สร้างเป็น IPST-MicroBOX

แผงวงจร IPST-MicroBOX

แปลง USB ==> Serial

เครื่องโปรแกรม PX-400

กว่าจะมาเป็น IPST MicroBOX SE

กลุ่มแผงวงจรขับเอาต์พุต

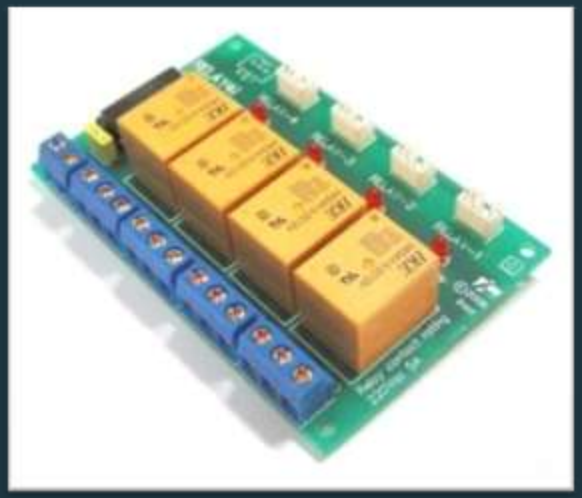
แผงวงจรขับ LED สองสี

แผงวงจรขับ LED อินฟราเรด

แผงวงจรขับลำโพงเปียโซ

แผงวงจรขับ ตัวเลข 7 ส่วน

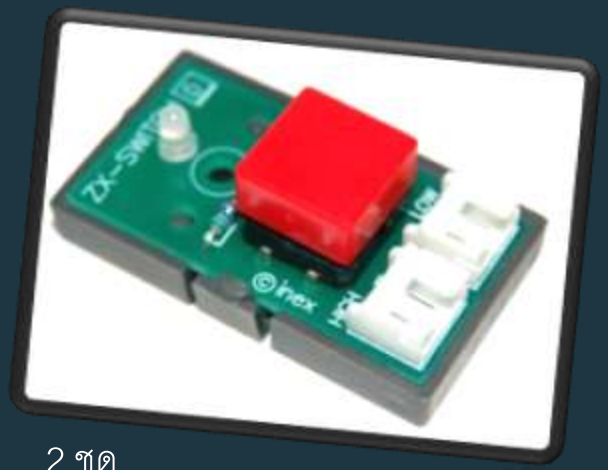
แผงวงจรขับมอเตอร์



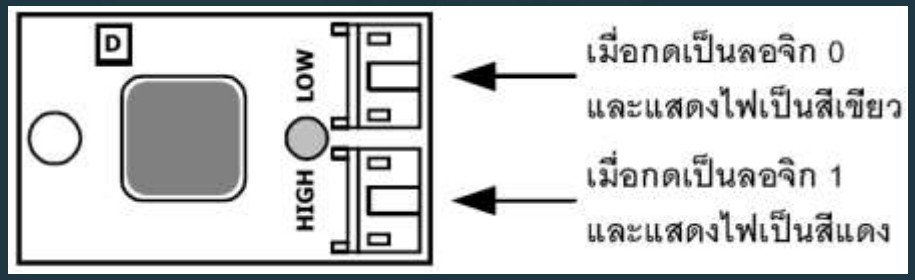
แผงวงจรขับรีเลย์

แผงวงจรแสดงผลและพอร์ตเอนกประสงค์

กลุ่มแผงวงจรขับเอาต์พุต



2 ชุด



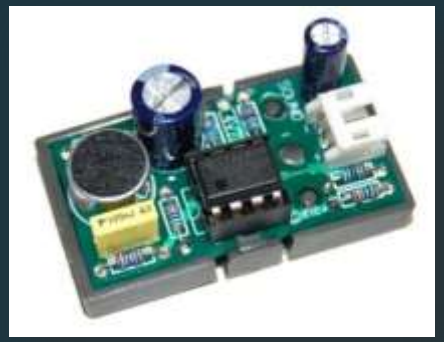
- เป็นอุปกรณ์รับข้อมูลดิจิทัล
- ให้โลจิก ‘0’ ถ้าตรวจจับคลื่นอินฟราเรดย่านความถี่ 38kHz ได้

กว่าจะมาเป็น IPST MicroBOX SE

กลุ่มแผงวงจรตรวจจับแบบอะนาลอก

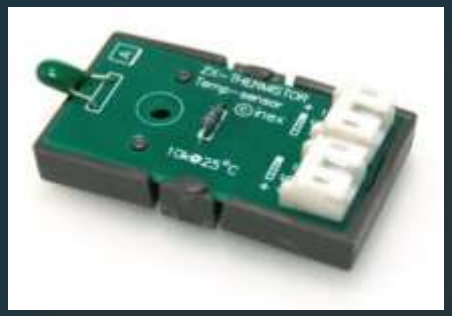
แผงวงจรตรวจจับแสง

แผงวงจรตรวจจับสนามแม่เหล็ก



โมดูลตรวจจับเสียง

แผงวงจรตรวจจับแสง
อินฟราเรด



แผงวงจรตรวจจับอุณหภูมิ

โมดูลตรวจจับและวัดระยะทางด้วย
แสงอินฟราเรด

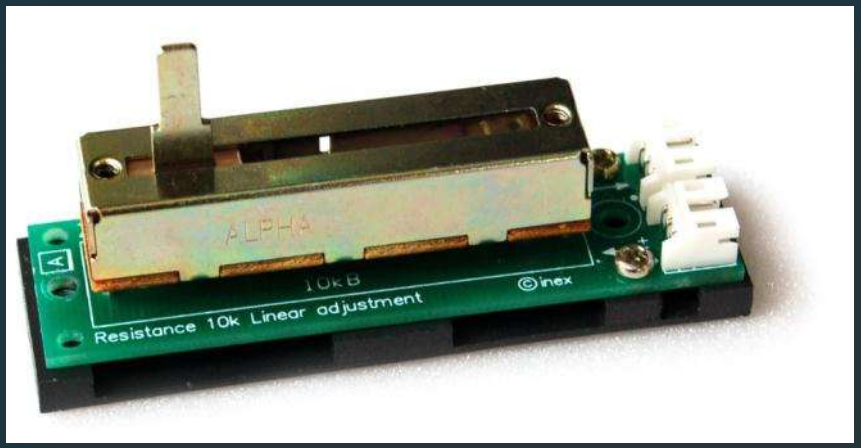
แผงวงจรตรวจจับการสะท้อน

แผงวงจรตรวจวัดค่าความต้านทาน

กว่าจะมาเป็น IPST MicroBOX SE

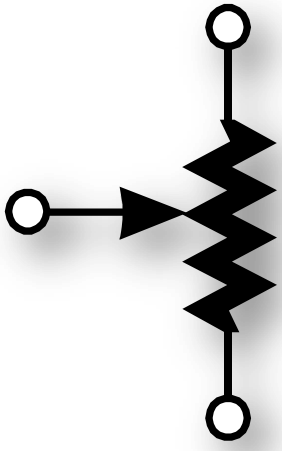
กลุ่มแผงวงจรตัวต้านทานปรับค่าได้

แผงวงจรตัวต้านทานปรับค่าได้ ตั้วตั้ง



แผงวงจรตัวต้านทานปรับค่าได้ แบบเลื่อน

แผงวงจรตัวต้านทานปรับค่าได้ ตั้วนอน

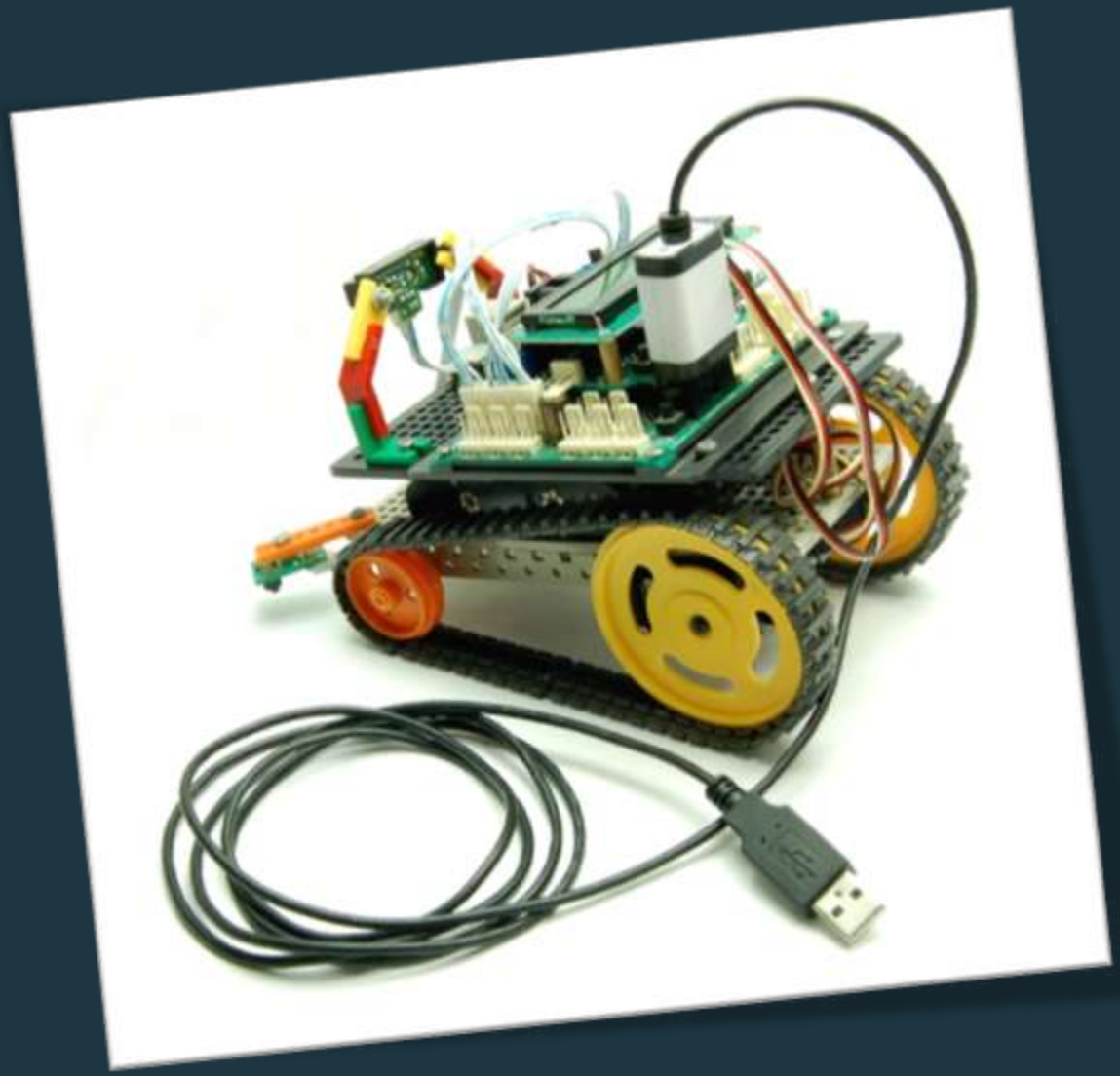


สัญลักษณ์ของ
ตัวต้านทานปรับค่าได้

กว่าจะมาเป็น

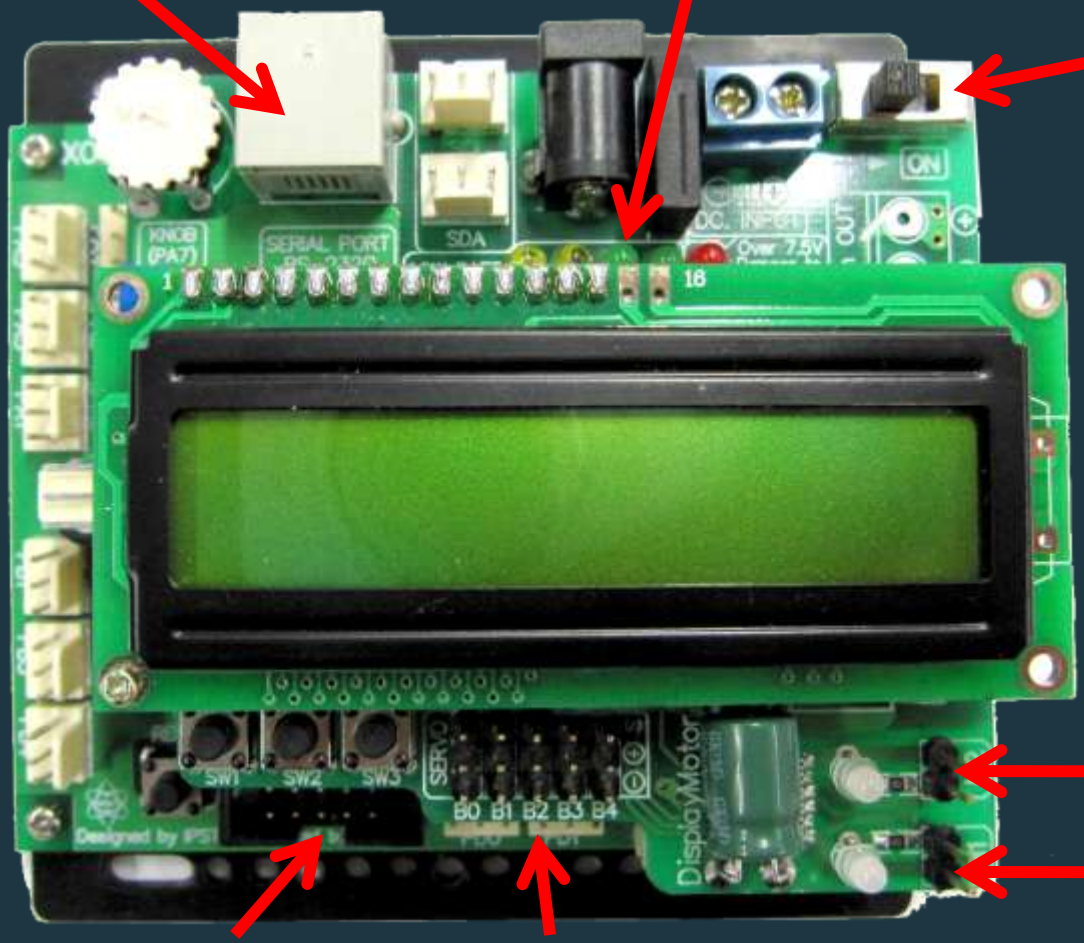
IPST MicroBOX SE

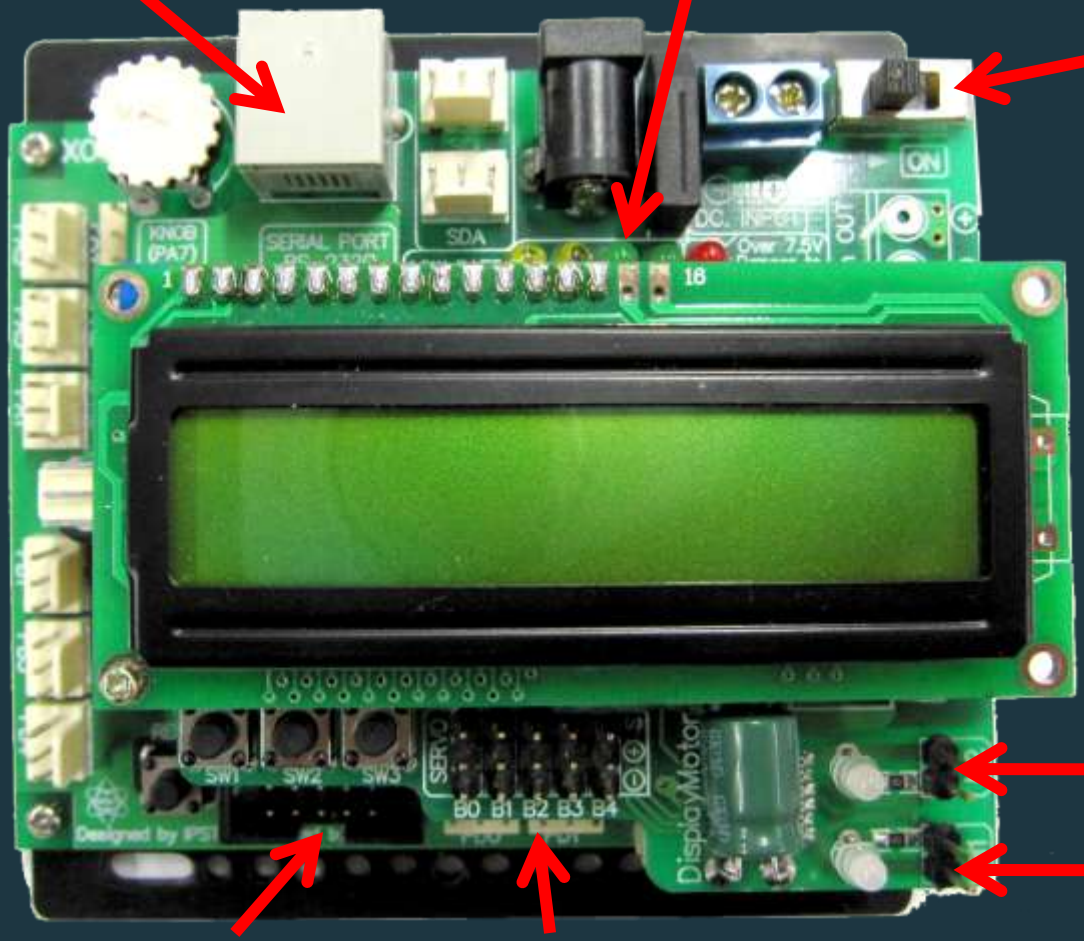
IPST-BOT



กว่าจะมาเป็น IPST MicroBOX SE

เสียบโมดูลเสริมเพื่อใช้ควบคุมมอเตอร์

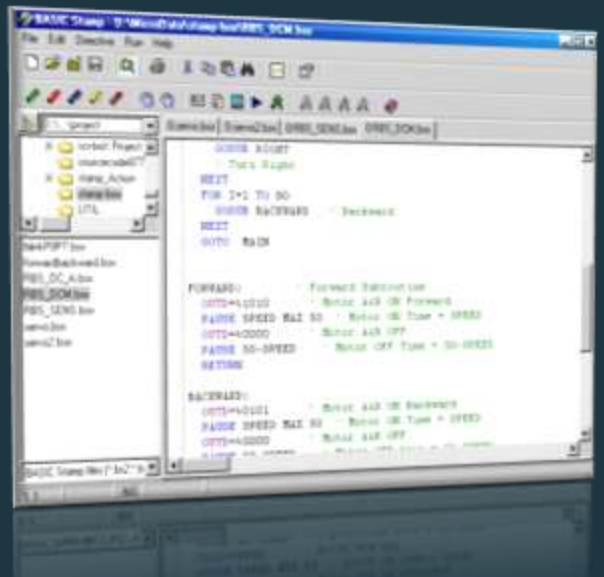



สื่อสาร
ไฟแสดงระดับแบตเตอรี่
สวิตช์เปิด/ปิด
Motor 2
Motor 1
ดาวน์โหลด
SERVO MOTOR

รูปแบบการพัฒนาไมโครคอนโทรลเลอร์ในปัจจุบัน

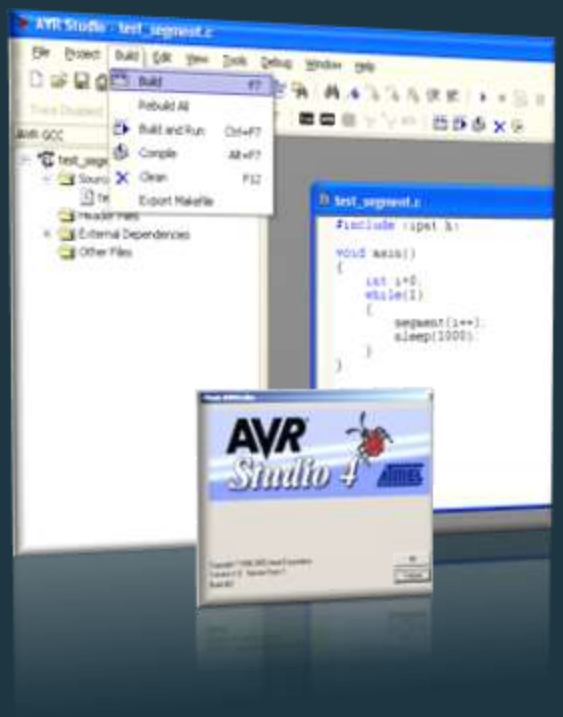
รูปแบบที่ 1 ใช้ไมโครคอนโทรลเลอร์มีตัวแปลภาษา

- โมดูลสำเร็จรูป ไม่ต้องพึ่งอุปกรณ์ภายนอก
- มี อินเตอร์พรีตเตอร์ (ตัวแปลภาษาในตัว)
- ซอฟต์แวร์ภาษาเบสิกเขียนง่าย มีให้ใช้งานฟรี
- ใช้เวลาในการเรียนรู้น้อย พัฒนางานได้เร็ว



ราคาค่อนข้างสูงเมื่อเทียบกับไมโครคอนโทรลเลอร์ตระกูลอื่น

รูปแบบการพัฒนาไมโครคอนโทรลเลอร์ในปัจจุบัน



รูปแบบที่ 2 ใช้ไมโครคอนโทรลเลอร์ผ่านคอมไพเลอร์

- ต้องใช้เครื่องโปรแกรมภายนอกในการโปรแกรม
- คอมไพเลอร์ภาษา C แจกฟรี
- ทำงานด้วยความเร็วสูง
- ราคา(ไมโครคอนโทรลเลอร์) ไม่แพง

สำหรับผู้เริ่มต้น ใช้ระยะเวลาในการเรียนรู้นานกว่ารูปแบบที่ 1

รูปแบบการพัฒนาไมโครคอนโทรลเลอร์ในปัจจุบัน

รูปแบบที่ 3 ใช้ไมโครคอนโทรลเลอร์ผ่านคอมไพเลอร์แบบซอร์สเปิด

- ไม่ต้องใช้เครื่องโปรแกรมภายนอก
- คอมไพเลอร์ภาษา C แจกฟรี
- ทำงานด้วยความเร็วสูง
- ราคา(ไมโครคอนโทรลเลอร์) ไม่แพง
- สร้างไลบรารี่ได้เอง และมีไลบรารี่สำหรับอุปกรณ์ต่อพ่วง

รูปแบบการพัฒนาไมโครคอนโทรลเลอร์ในปัจจุบัน

รูปแบบที่ 3 ใช้ไมโครคอนโทรลเลอร์ผ่านคอมไพเลอร์แบบซอร์สเปิด

- ไม่ต้องใช้เครื่องโปรแกรมภายนอก
- คอมไพเลอร์ภาษา C แจกฟรี
- ทำงานด้วยความเร็วสูง
- ราคา(ไมโครคอนโทรลเลอร์) ไม่แพง
- สร้างไลบรารี่ได้เอง และมีไลบรารี่สำหรับอุปกรณ์ต่อพ่วง

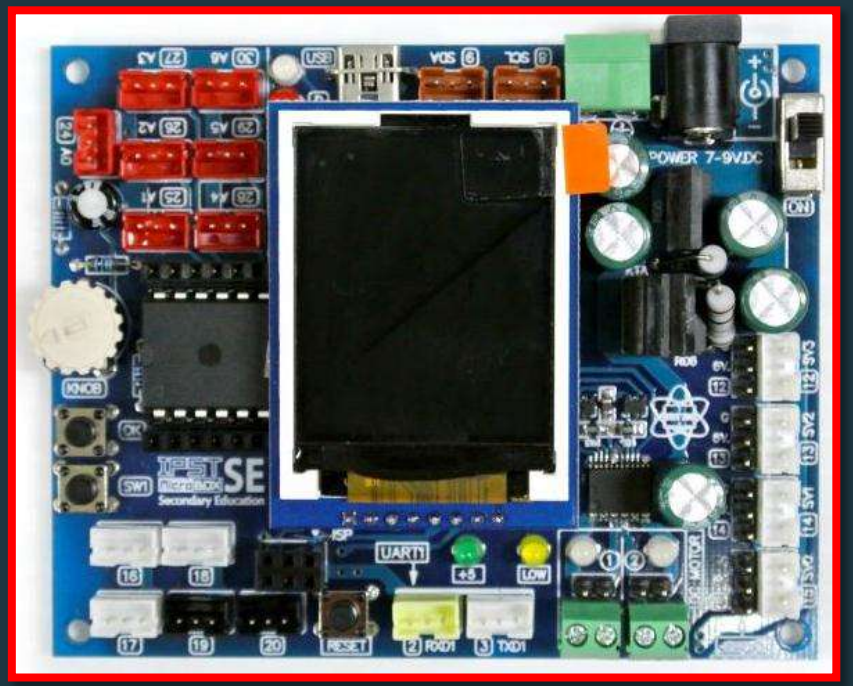

wiring.org.co
Google
Wiring
Search wiring.org.co:
Cover \ Exhibition \ Reference \ Learning \ Hardware \ Download \ About
»Feed »Forum »Wiki »Code
What will YOU do with the W ?
Wiring
get on board.
» Download Wiring
» Getting the Wiring hardware
» Browse Tutorials
» Play with Examples

ต้นกำเนิด IPST-SE

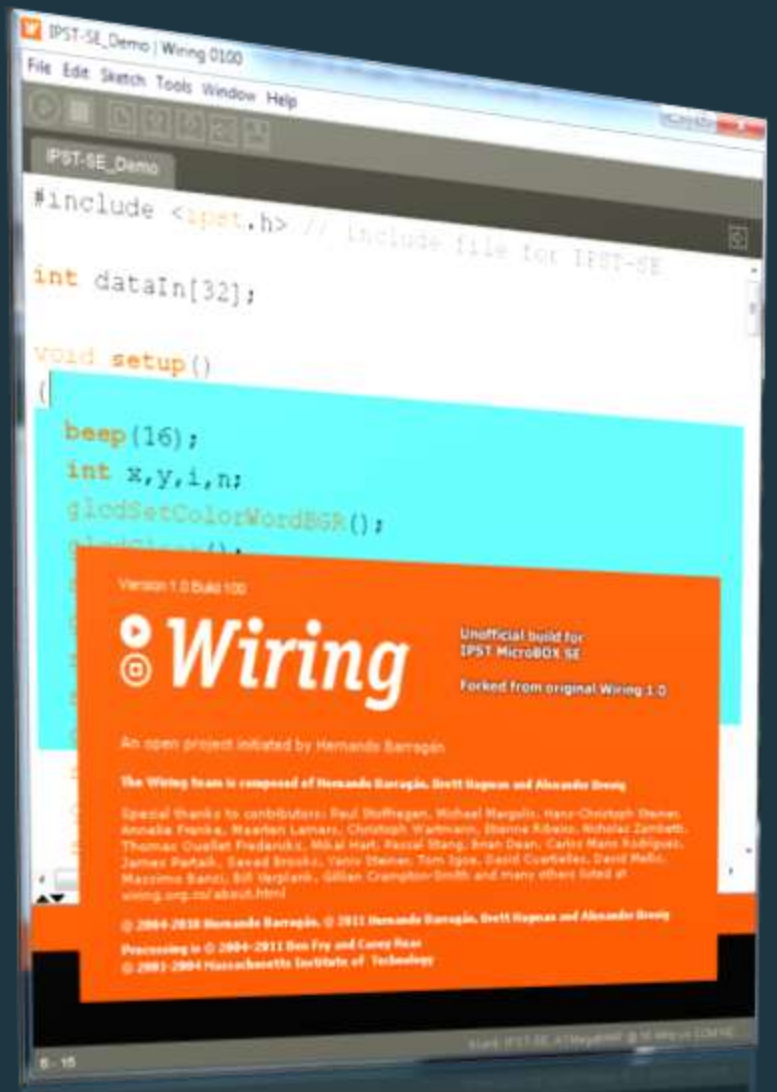
โครงการ Opensource ของ Wiring

HARDWARE

IPST-SE



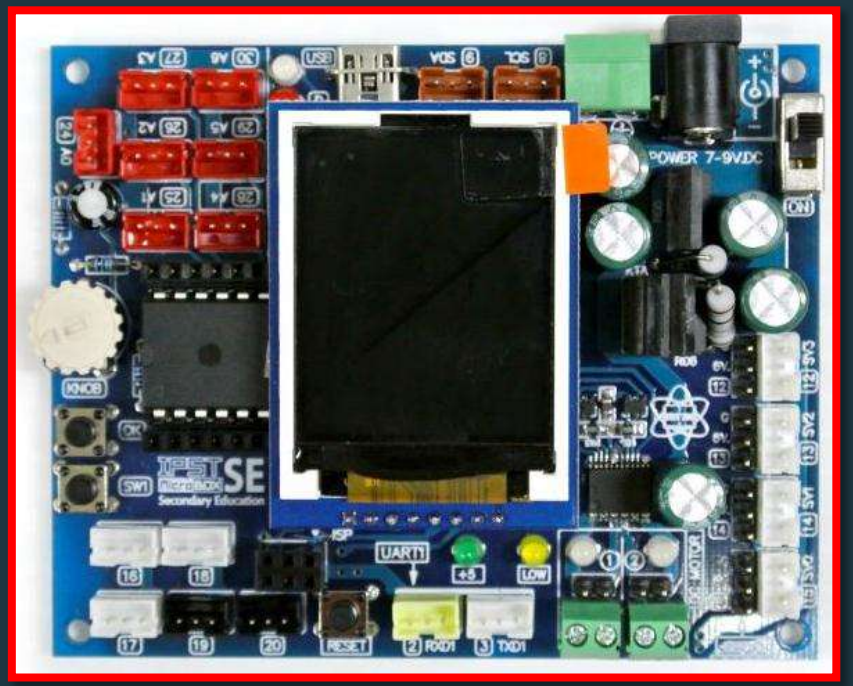
คอมไพเลอร์ Opensource



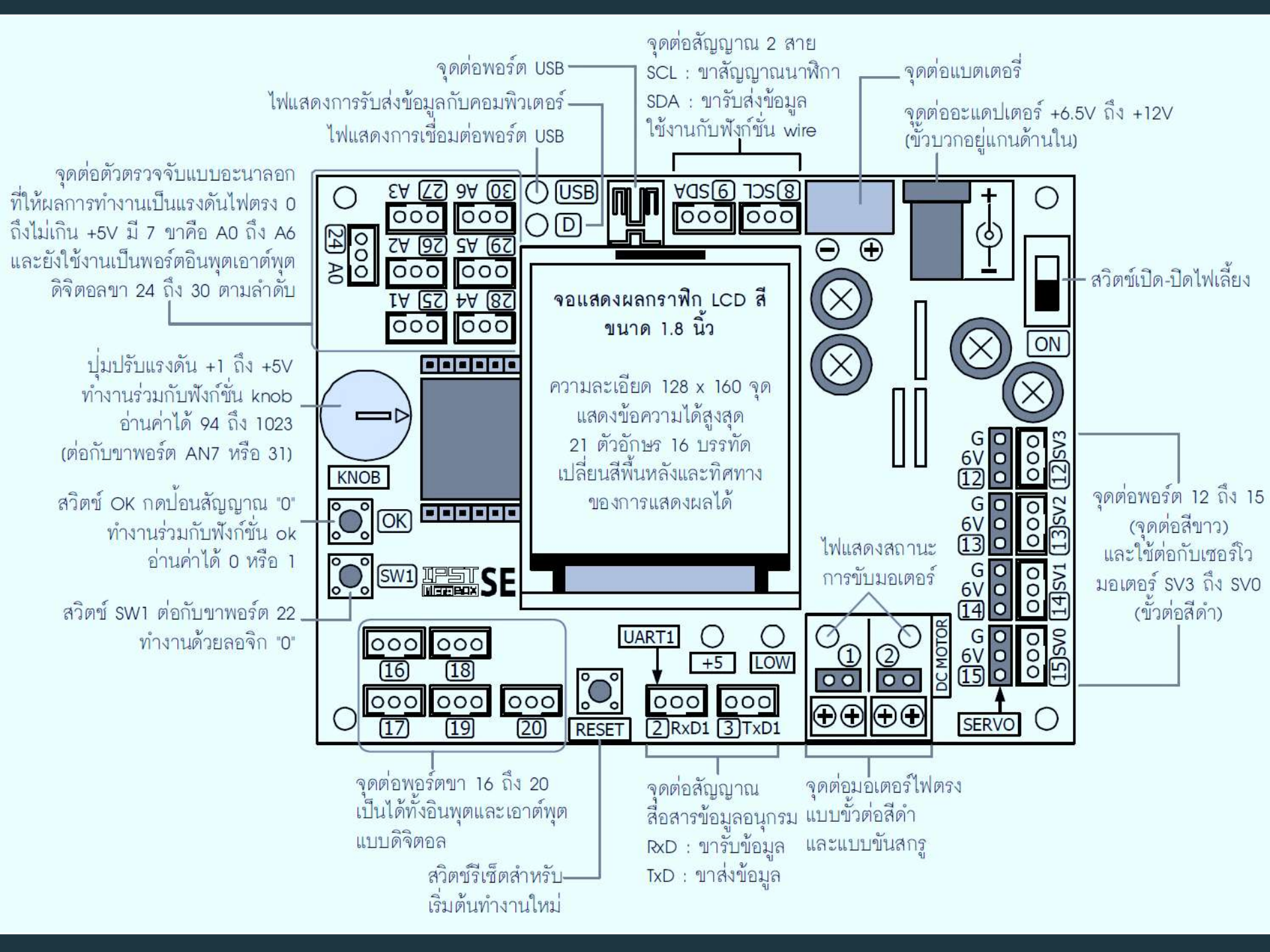
Edit+Compire+Download

Software

IPST-SE

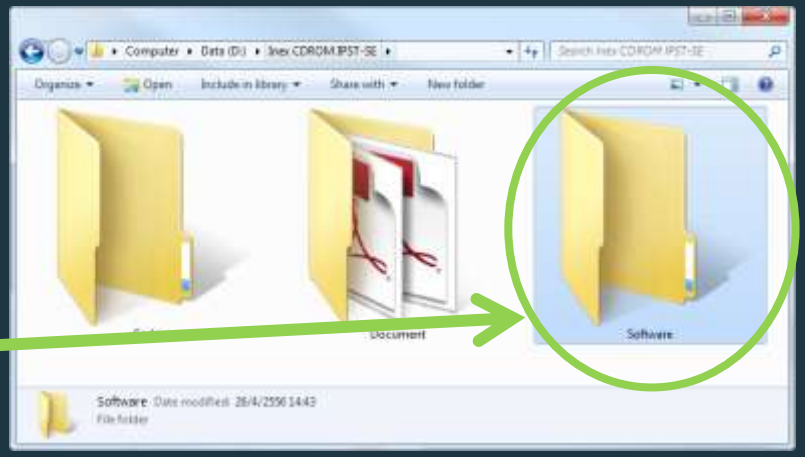



LED8
ZX-POTH
POTENTIOMETER
SPEAKER
THE CONTAINER


จุดต่อพอร์ต USB
ไฟแสดงการรับส่งข้อมูลกับคอมพิวเตอร์
ไฟแสดงการเชื่อมต่อพอร์ต USB
จุดต่อสัญญาณ 2 สาย
SCL : ขาสัญญาณนาฬิกา
SDA : ขารับส่งข้อมูล
ใช้งานกับฟังก์ชั่น wire
จุดต่อแบตเตอรี่
จุดต่ออะแดปเตอร์ +6.5V ถึง +12V
(ขั้วบวกอยู่แกนด้านใน)
จุดต่อตัวตรวจจับแบบอะนาลอก
ที่ให้ผลการทำงานเป็นแรงดันไฟตรง 0
ถึงไม่เกิน +5V มี 7 ขาคือ A0 ถึง A6
และยังใช้งานเป็นพอร์ตอินพุตเอาต์พุต
ดิจิตอลขา 24 ถึง 30 ตามลำดับ
สวิตช์เปิด-ปิดไฟเลี้ยง
ปุ่มปรับแรงดัน +1 ถึง +5V
ทำงานร่วมกับฟังก์ชั่น knob
อ่านค่าได้ 94 ถึง 1023
(ต่อกับขาพอร์ต AN7 หรือ 31)
สวิตช์ OK กดป้อนสัญญาณ "0"
ทำงานร่วมกับฟังก์ชั่น ok
อ่านค่าได้ 0 หรือ 1
สวิตช์ SW1 ต่อกับขาพอร์ต 22
ทำงานด้วยลอจิก "0"
จอแสดงผลกราฟิก LCD สี
ขนาด 1.8 นิ้ว
ความละเอียด 128 x 160 จุด
แสดงข้อความได้สูงสุด
21 ตัวอักษร 16 บรรทัด
เปลี่ยนสีพื้นหลังและทิศทาง
ของการแสดงผลได้
จุดต่อพอร์ต 12 ถึง 15
(จุดต่อสีขาว)
และใช้ต่อกับเซอร์โว
มอเตอร์ SV3 ถึง SV0
(ขั้วต่อสีดำ)
ไฟแสดงสถานะ
การขับมอเตอร์
จุดต่อพอร์ตขา 16 ถึง 20
เป็นได้ทั้งอินพุตและเอาต์พุต
แบบดิจิตอล
สวิตช์รีเซ็ตสำหรับ
เริ่มต้นทำงานใหม่
จุดต่อสัญญาณ
สื่อสารข้อมูลอนุกรม
RxD : ขารับข้อมูล
TxD : ขาส่งข้อมูล
จุดต่อมอเตอร์ไฟตรง
แบบขั้วต่อสีดำ
และแบบขันสกรู
USB
D
SCL
SDA
A0
A1
A2
A3
A4
A5
A6
ON
KNOB
OK
SW1
IPST
Micro Box
SE
UART1
+5
LOW
RESET
2 RxD1
3 TxD1
DC MOTOR
SERVO
SV0
SV1
SV2
SV3
G
6V

ขั้นตอนติดตั้งโปรแกรม

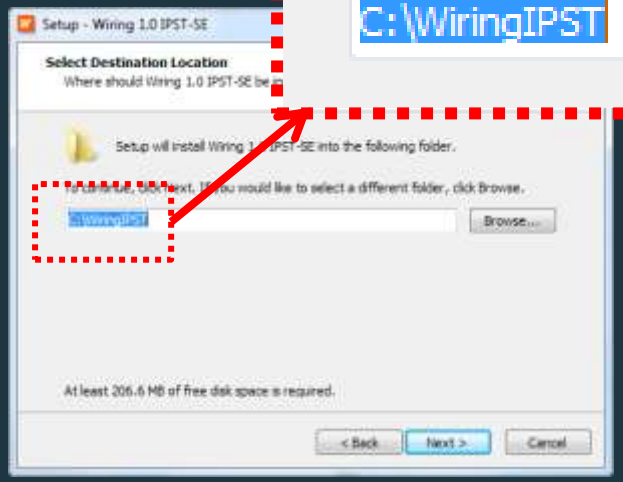
เปิดโฟลเดอร์ติดตั้งโปรแกรม

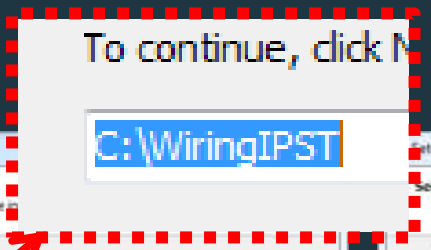


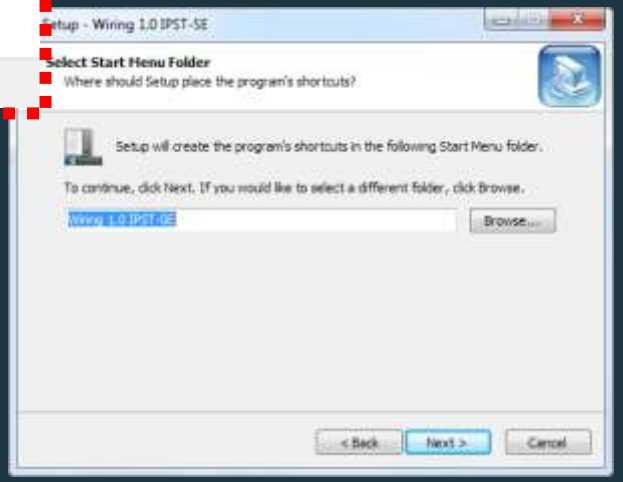
เปิดโฟลเดอร์สำหรับ Windows

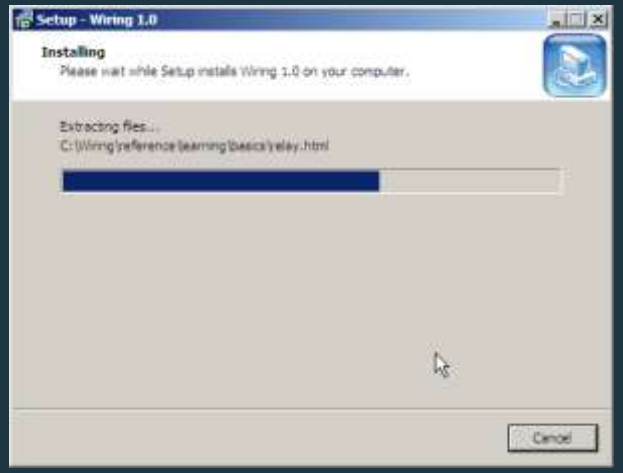
ดับเบิ้ลคลิก ติดตั้ง

ขั้นตอนติดตั้งโปรแกรม









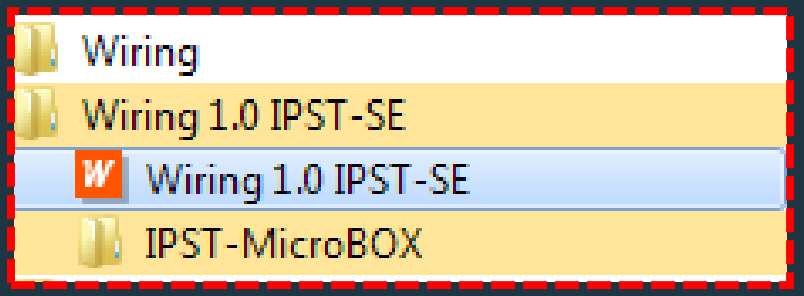
เมื่อจบขั้นตอนนี้จะมี
หน้าต่าง ติดตั้งไดรเวอร์
ห้ามกด Cancel

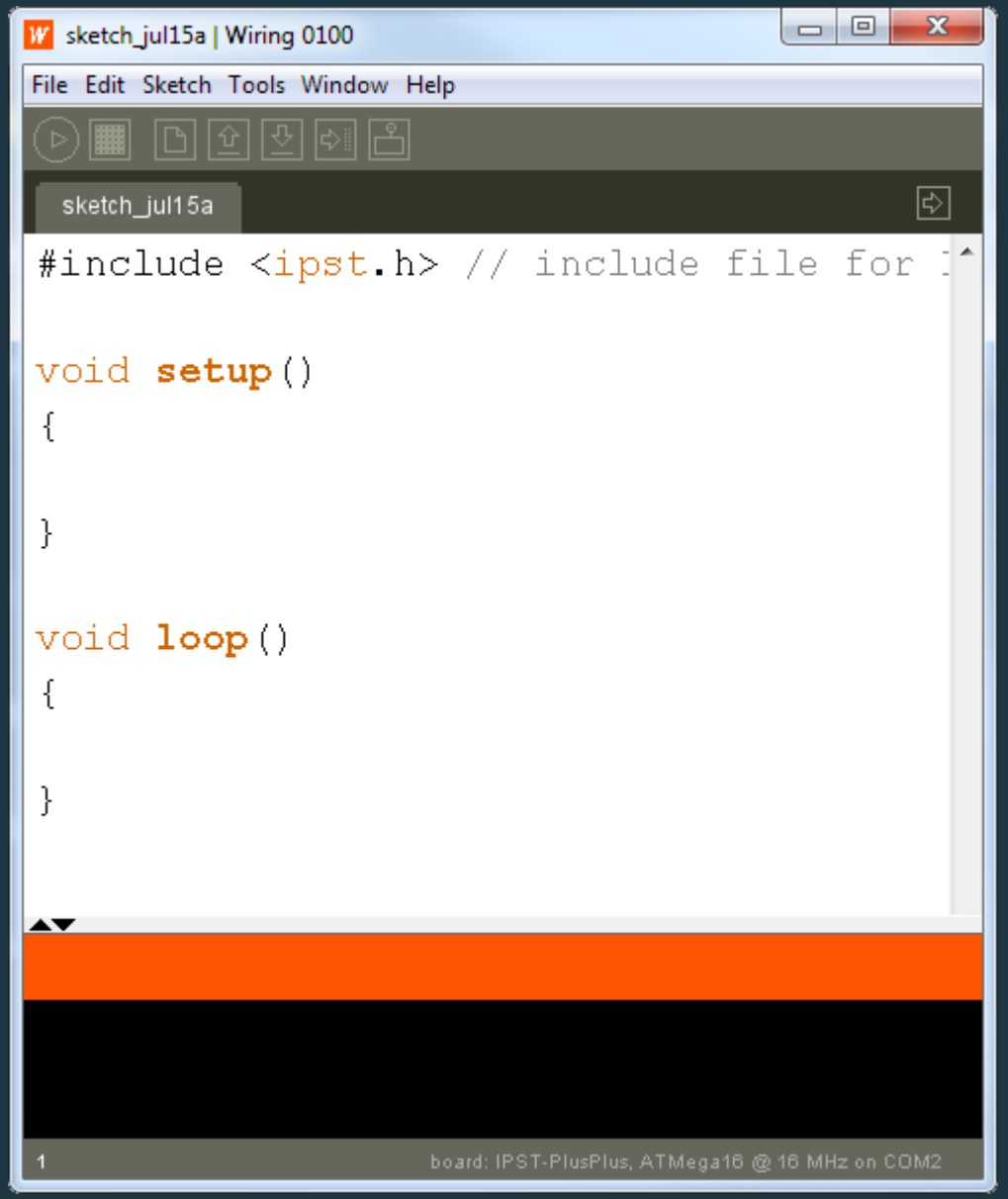
ขั้นตอนติดตั้งไดรเวอร์

เมื่อจบขั้นตอนนี้จะมีหน้าต่าง ติดตั้งไดรเวอร์

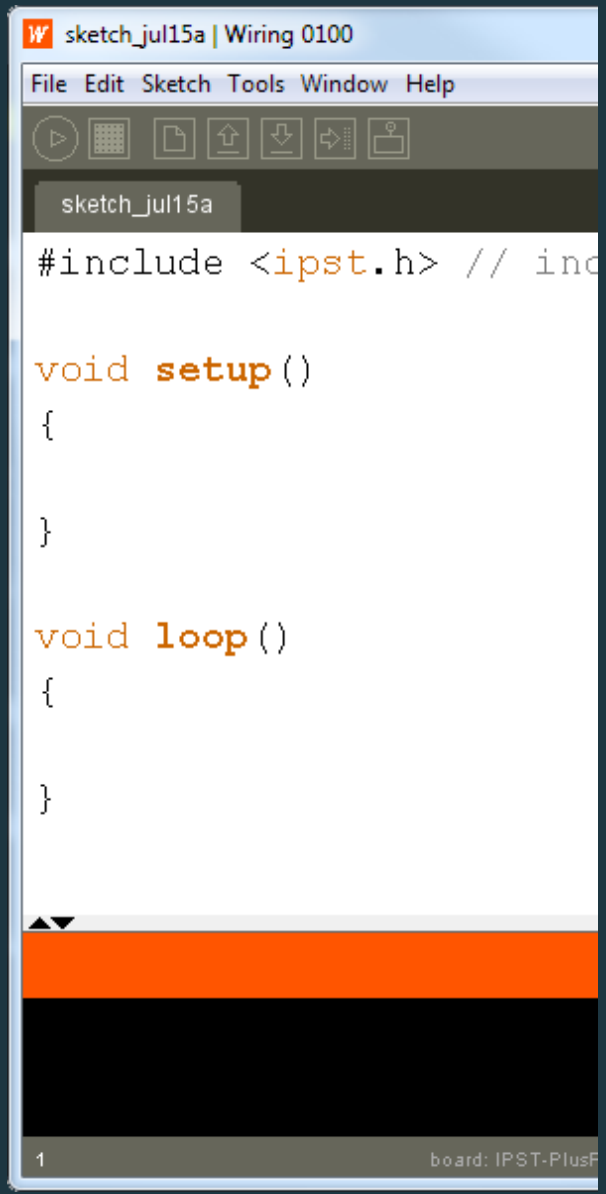
ห้ามกด Cancel

โปรแกรม wiring





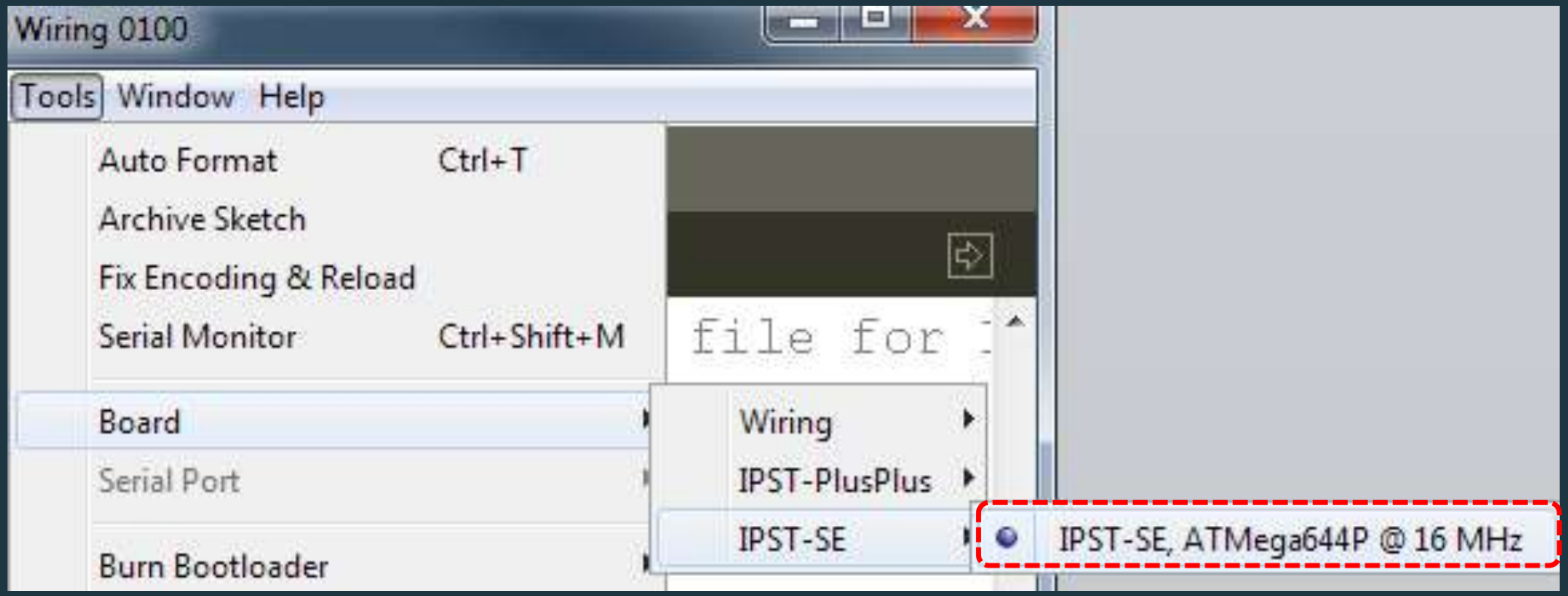
รูปแบบการทำงานโปรแกรม wiring




```
void setup()
{
  สำหรับกำหนดค่า เกิดขึ้นครั้งเดียว
}

void loop()
{
  โปรแกรมหลักทำงานต่อเนื่อง
}
```

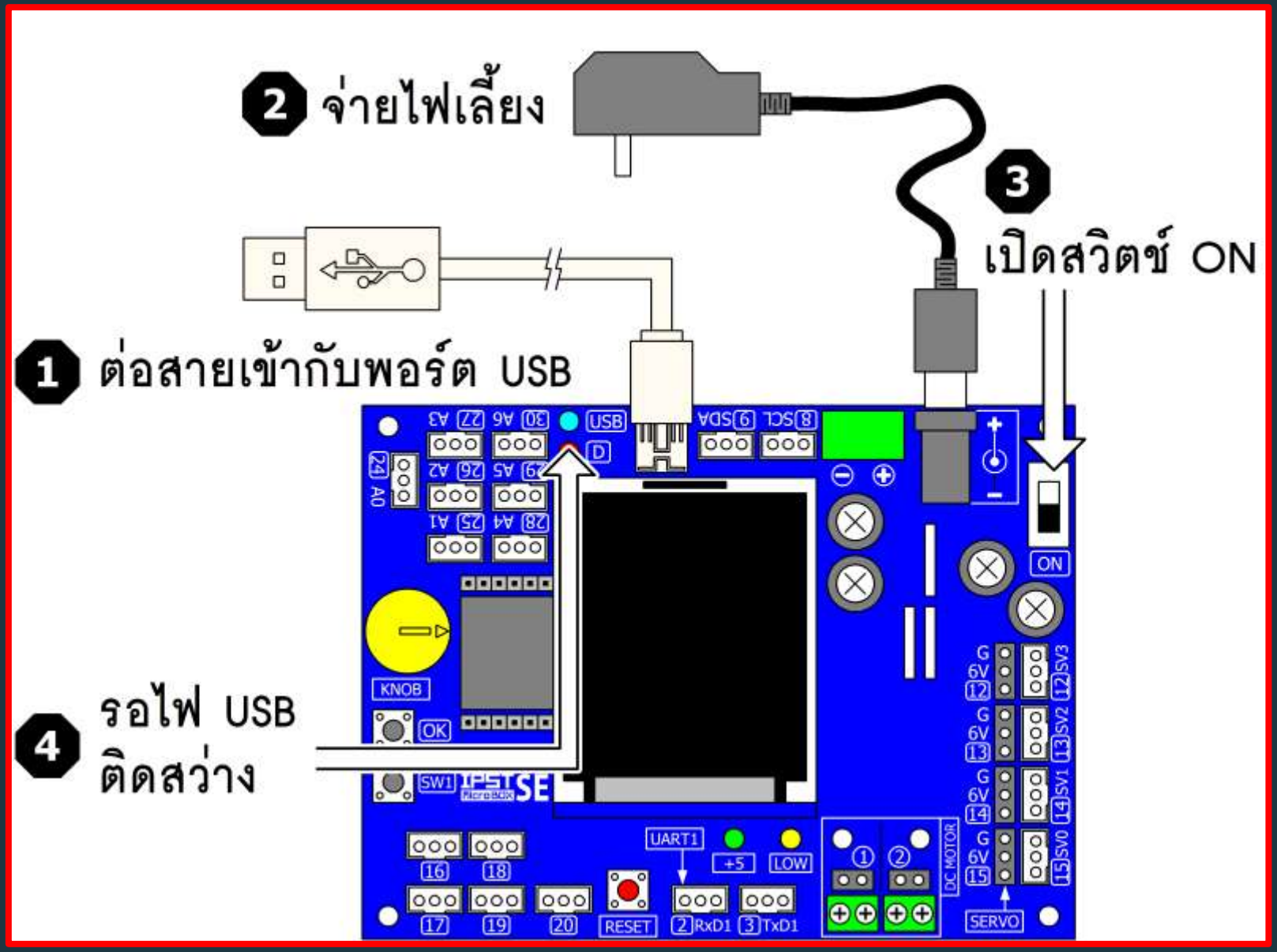
รูปแบบการทำงานโปรแกรม wiring

เลือกบอร์ดที่ใช้งาน

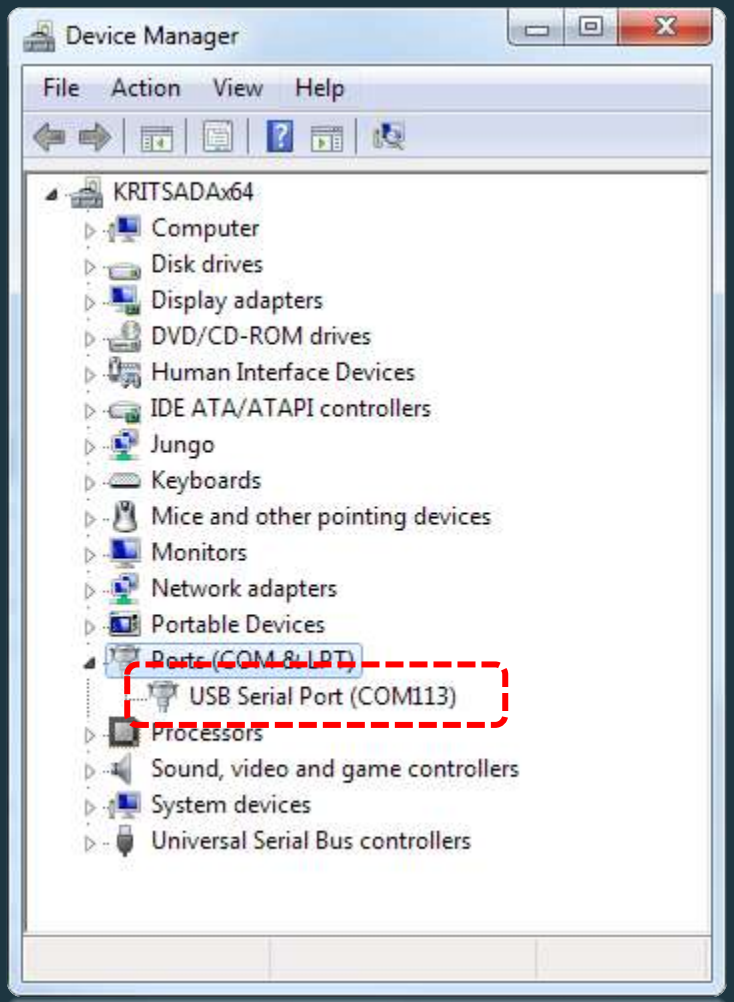


เลือกบอร์ดเป็น IPST-SE

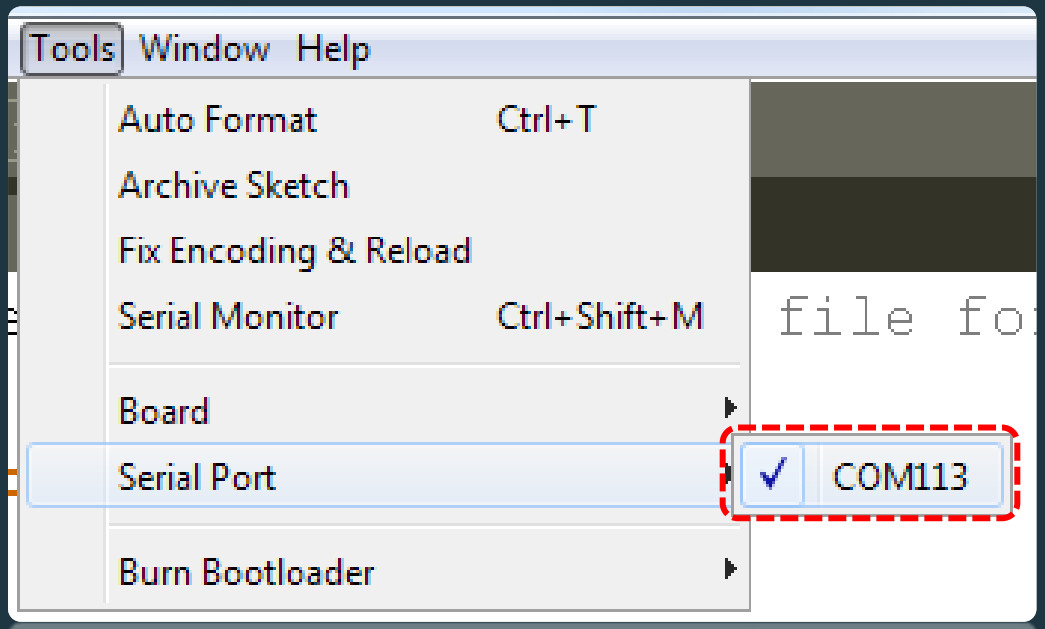
เชื่อมต่อกับคอมพิวเตอร์



เลือกพอร์ตอนุกรม



เลือก Serial Port ให้ตรงตำแหน่ง



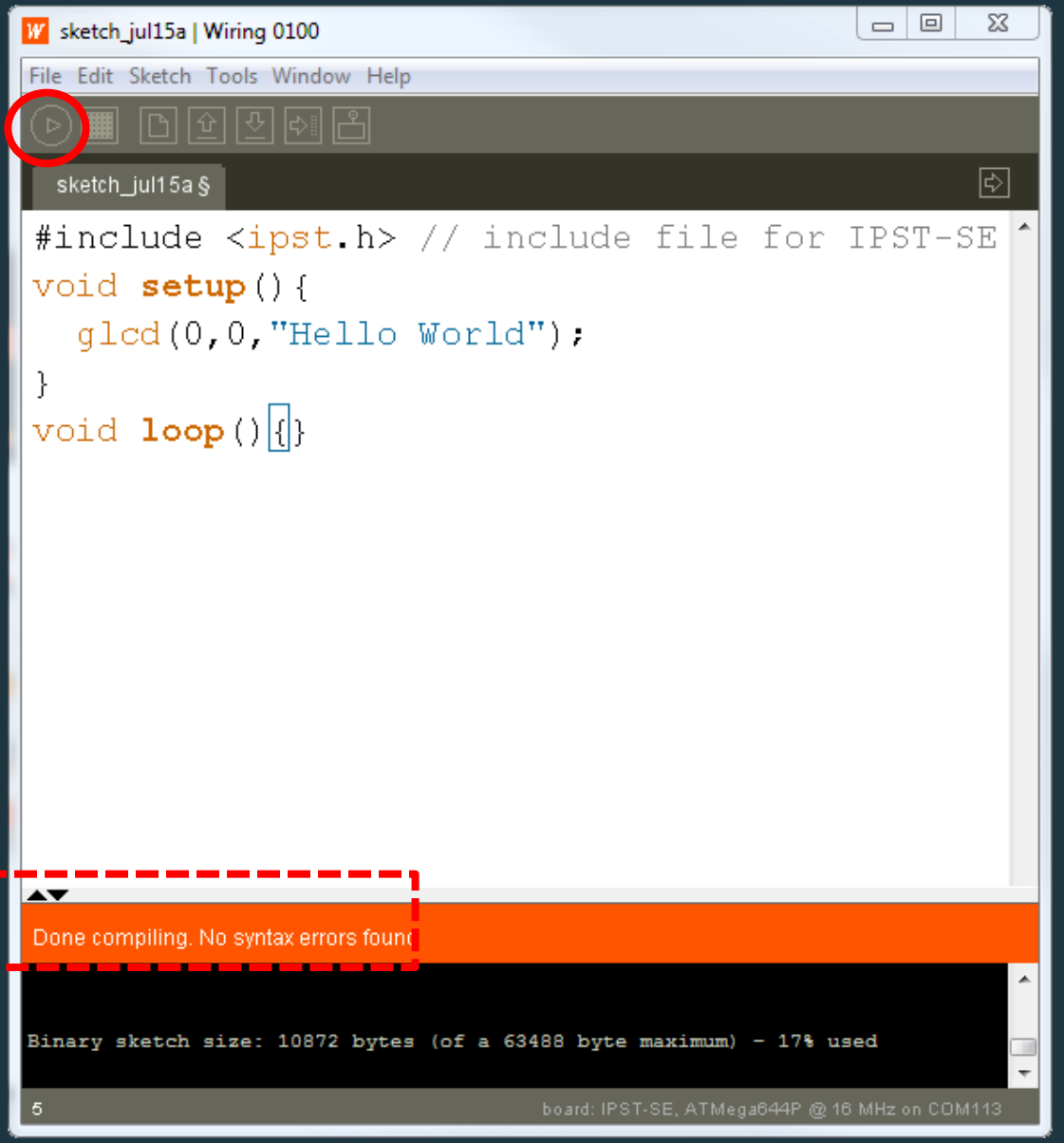
โปรแกรมแรก

```
#include <ipst.h>
void setup()
{
   glcd(0,0,"Hello World");
}

void loop()
{
}
```

ตรวจสอบไวยกรณ์

คอมไพล์

แจ้งผลว่าคอมไพล์ผ่าน

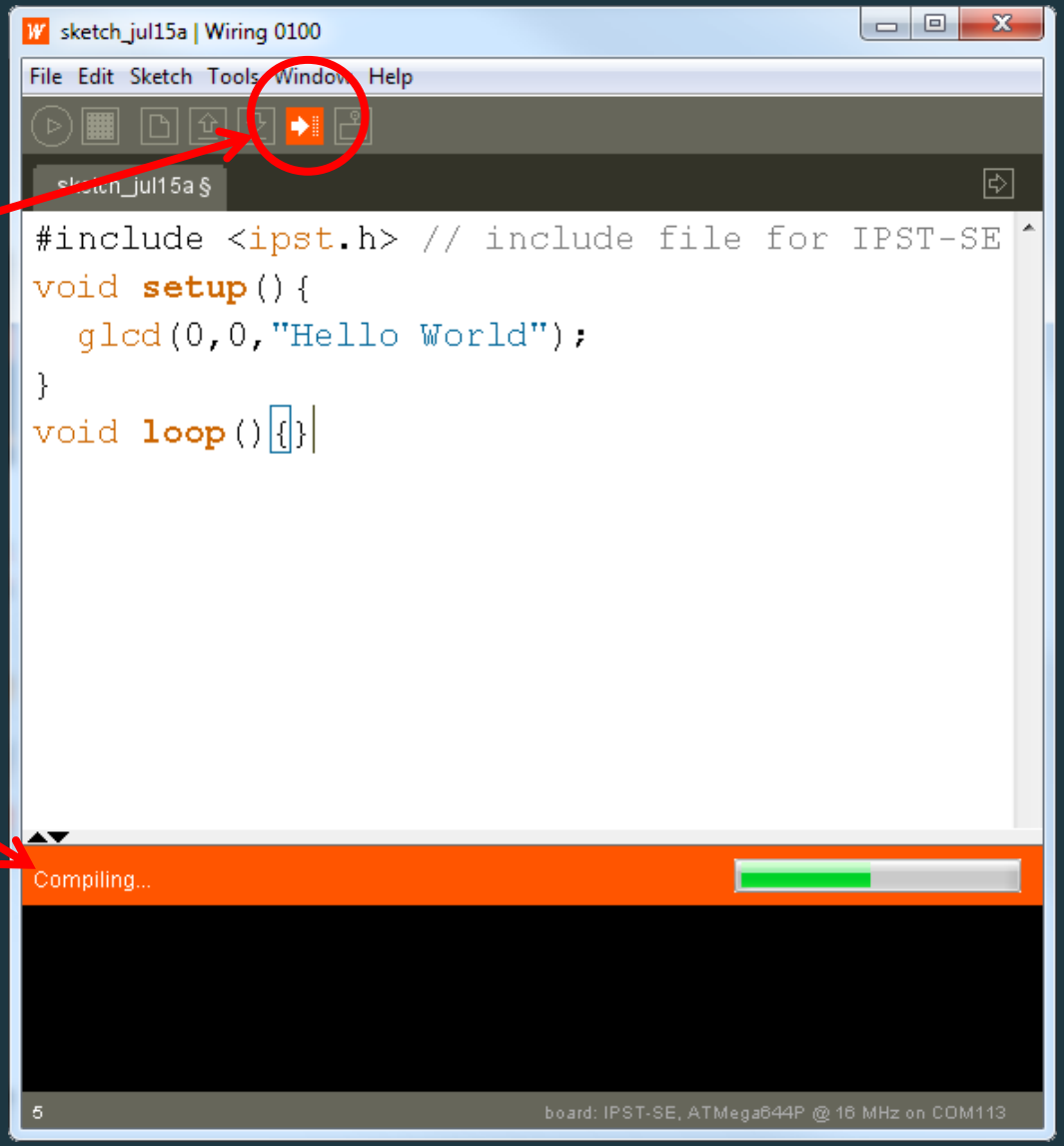


โหลดโปรแกรมไปยัง IPST-SE

อัพโหลด

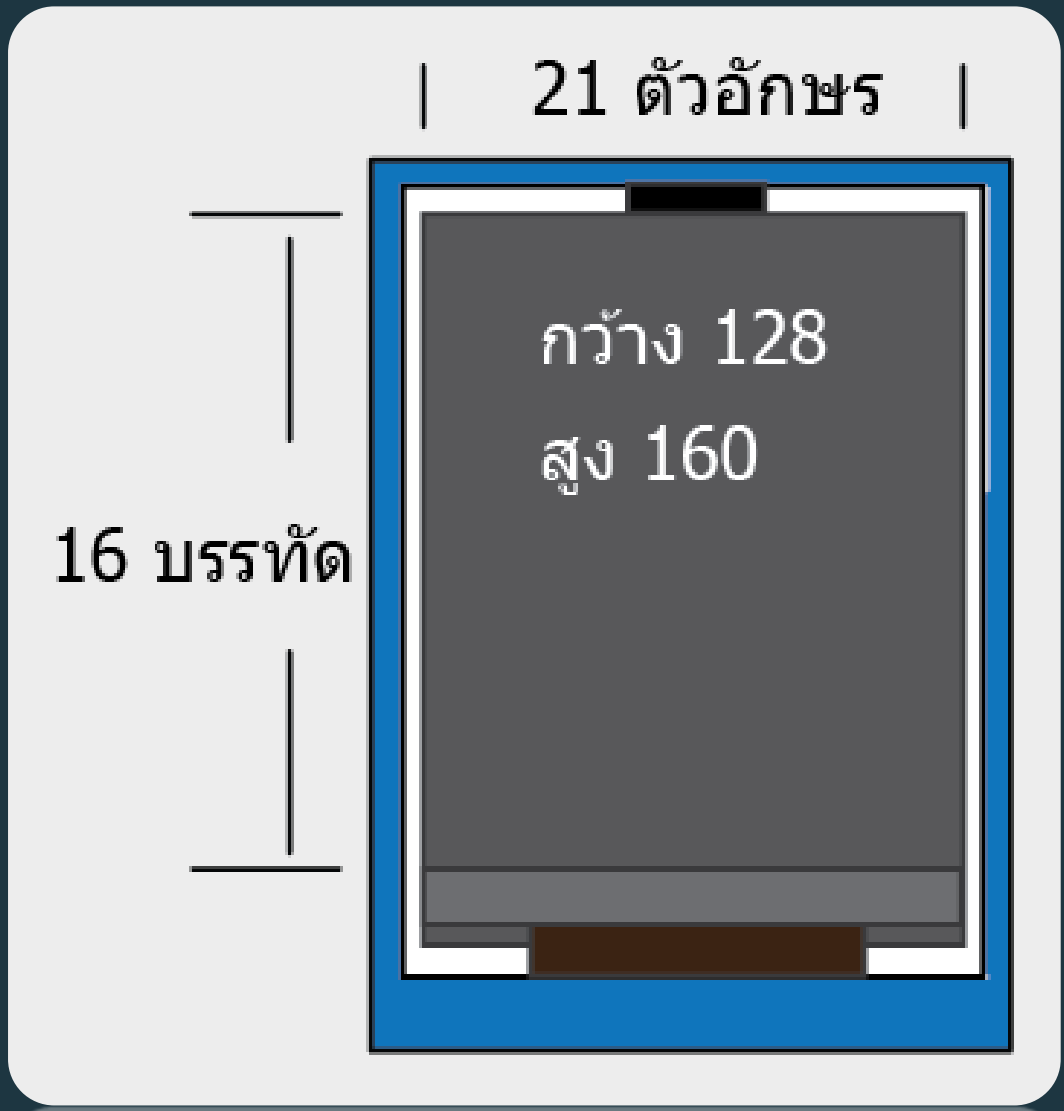
คีย์ลัด Ctrl+U

คอมไพล์อีกครั้ง



ผลลัพธ์โปรแกรมที่ 1

คุณสมบัติของจอภาพ



คำสั่ง GLCD

glcd แสดงข้อความที่จอ GLCD ได้ 21 ตัว 16 บรรทัด (size 1)

รูปแบบ

```
void glcd(x,y,*p,...)
```

พารามิเตอร์

x คือตำแหน่งบรรทัดมีค่าตั้งแต่ 0-15

y คือตำแหน่งตัวอักษรมีค่าตั้งแต่ 0-24

*p คือข้อความที่ต้องการนำมาแสดง

ค่าพิเศษ

%d แสดงตัวเลขจำนวนเต็มในช่วง -32,768 ถึง 32,767

%h แสดงตัวเลขฐานสิบหก

%b แสดงตัวเลขฐานสอง

%l แสดงตัวเลขจำนวนเต็มในช่วง -2,147,483,648 ถึง 2,147,483,647

%f แสดงผลตัวเลขจำนวนจริง (แสดงทศนิยม 3 หลัก)

ไลบรารี่ ของ GLCD

glcd
setTextColor
setTextBackgroundColor
glcdClear
glcdFillScreen
glcdMode
setTextSize

glcdPixel
glcdRect
glcdFillRect
glcdLine
glcdCircle
glcdFillCircle
glcdArc

setTextColor(COLOR)

ค่าสีตัวอักษร

```
unsigned   color[]={
  GLCD_RED,
  GLCD_GREEN,
  GLCD_BLUE,
  GLCD_YELLOW,
  GLCD_BLACK,
  GLCD_WHITE,
  GLCD_SKY,
  GLCD_MAGENTA
};
```

ตัวอย่าง

```
#include <ipst.h>
void setup(){
   setTextColor(GLCD_WHITE);
   glcd(0,0,"Hello");
   setTextColor(GLCD_GREEN);
   glcd(1,0,"World");
}
void loop(){}
```

setTextBackgroundColor(COLOR)

ค่าสีพื้นหลังตัวอักษร

ตัวอย่าง

```
#include <ipst.h>
void setup(){
setTextBackgroundColor(GLCD_RED);
setTextColor(GLCD_YELLOW);
glcd(0,0,"Hello World");
}
void loop(){}
```

```
unsigned   color[]={
  GLCD_RED,
  GLCD_GREEN,
  GLCD_BLUE,
  GLCD_YELLOW,
  GLCD_BLACK,
  GLCD_WHITE,
  GLCD_SKY,
  GLCD_MAGENTA
};
```

`glcdClear()`

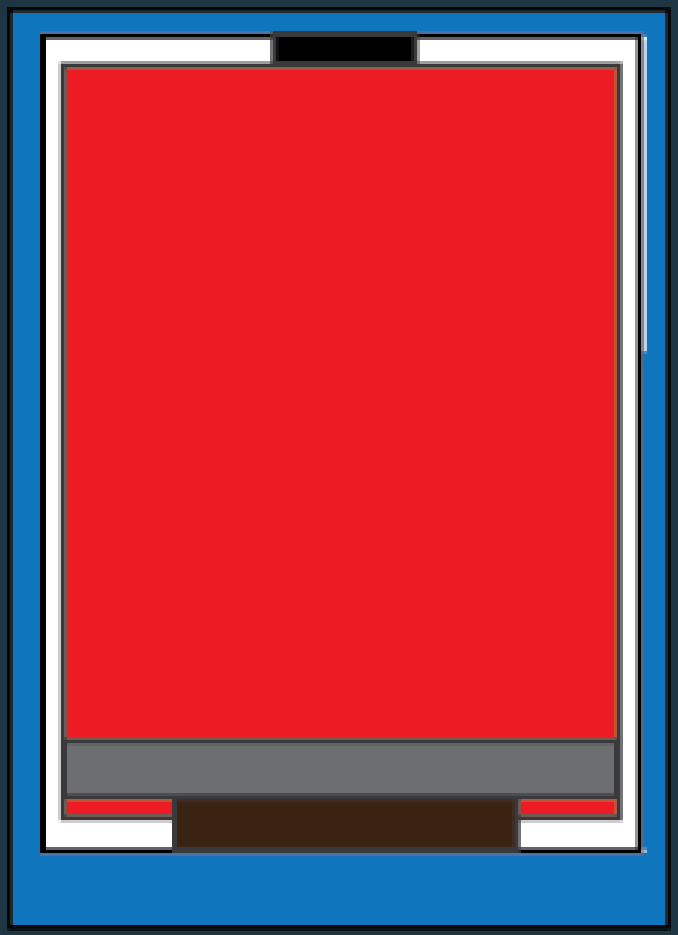
เคลียร์หน้าจอ

`glcdFillScreen(COLOR)`

เทสีลงบนหน้าจอทั้งหน้า

ตัวอย่าง

```
#include <ipst.h>
void setup(){}
void loop(){
 glcdClear();
 sleep(500);
 glcdFillScreen(color[0]);
 sleep(500);
 glcdFillScreen(color[1]);
 sleep(500);
 glcdFillScreen(color[2]);
 sleep(500);
}
```



glcdMode หมุนหน้าจอ ปกติเป็น Mode 0

ตัวอย่าง

```
#include <ipst.h>
void setup(){}
void loop(){
 glcdMode(0);
 glcd(0,0,"Mode 0");
 sleep(1000);
 glcdMode(1);
 glcd(0,0,"Mode 1");
 sleep(1000);
 glcdMode(2);
 glcd(0,0,"Mode 2");
 sleep(1000);
 glcdMode(3);
 glcd(0,0,"Mode 3");
 sleep(1000);
}
```

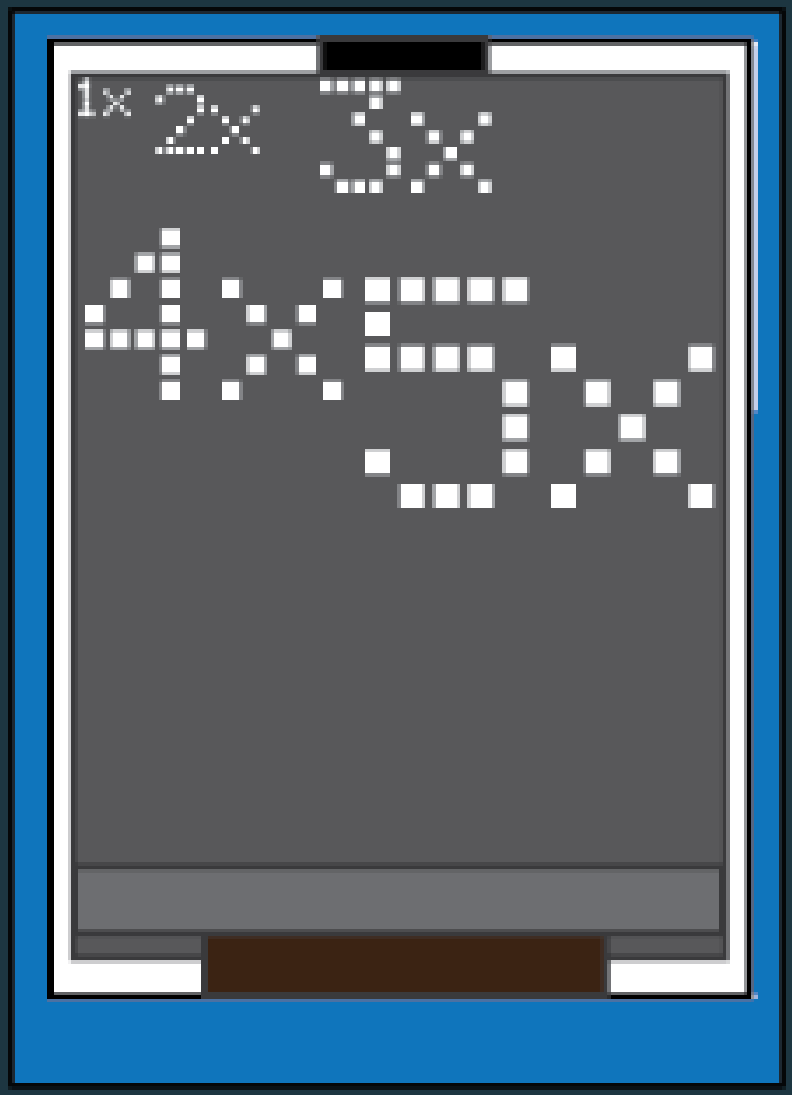
setTextSize

ปรับขนาดตัวอักษร เป็นเท่าตัว

ถ้าไม่กำหนดขนาดเป็น 1 เท่า

ตัวอย่าง

```
#include <ipst.h>
void setup(){}
void loop(){
 setTextSize(1);
 glcd(0,0,"1x");
 setTextSize(2);
 glcd(0,2,"2x");
 setTextSize(3);
 glcd(0,3,"3x");
 setTextSize(4);
 glcd(1,0,"4x");
 setTextSize(5);
 glcd(1,2,"5x");
}
```



การแสดงผลกราฟิก

```
glcdRect(x,y,width,height,color)
glcdFillRect(x,y,width,height,color)
glcdCircle(x,y,radius,color)
glcdFillCircle(x,y,radius,color)
glcdLine(x1,y1,x2,y2,color)
```

x ตำแหน่งแนวนอน

y ตำแหน่งแนวตั้ง

width ความกว้าง

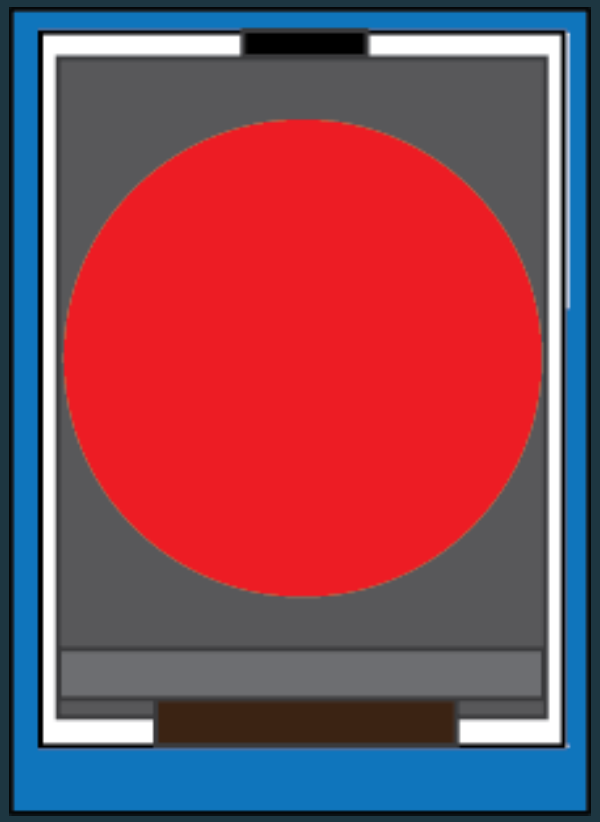
height ความสูง

redius รัศมี

จอกว้าง 128 pixel

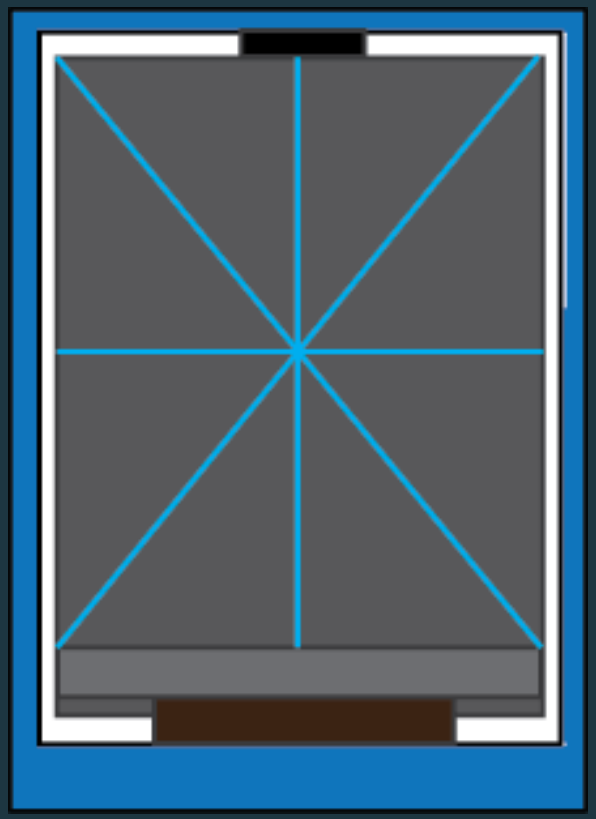
จอสูง 160 pixel

โจทย์



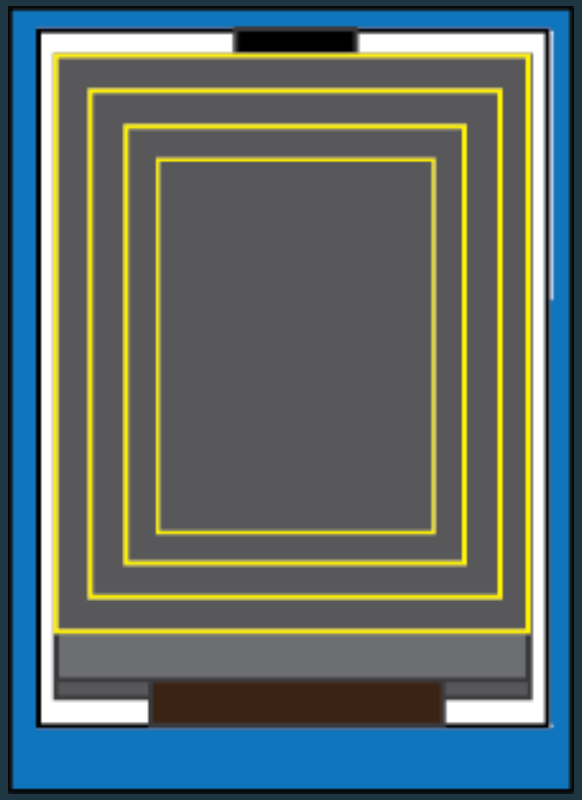
สร้างวงกลมสีแดงอยู่กึ่งกลางจอภาพ>รัศมีเต็มจอพอดี

โจทย์



ลากเส้น 4 เส้นโดยมีจุดตัดอยู่กลางจอภาพพอดี

โจทย์



สร้างสี่เหลี่ยมซ้อนกันดังรูป

คำสั่ง GLCD

glcd แสดงข้อความที่จอ GLCD ได้ 21 ตัว 16 บรรทัด (size 1)

รูปแบบ

```
void glcd(x,y,*p,...)
```

พารามิเตอร์

x คือตำแหน่งบรรทัดมีค่าตั้งแต่ 0-15

y คือตำแหน่งตัวอักษรมีค่าตั้งแต่ 0-24

*p คือข้อความที่ต้องการนำมาแสดง

ค่าพิเศษ

%d แสดงตัวเลขจำนวนเต็มในช่วง -32,768 ถึง 32,767

%h แสดงตัวเลขฐานสิบหก

%b แสดงตัวเลขฐานสอง

%l แสดงตัวเลขจำนวนเต็มในช่วง -2,147,483,648 ถึง 2,147,483,647

%f แสดงผลตัวเลขจำนวนจริง (แสดงทศนิยม 3 หลัก)

การแสดงผลตัวเลข

```
glcd(0,0,"%d",100);
```

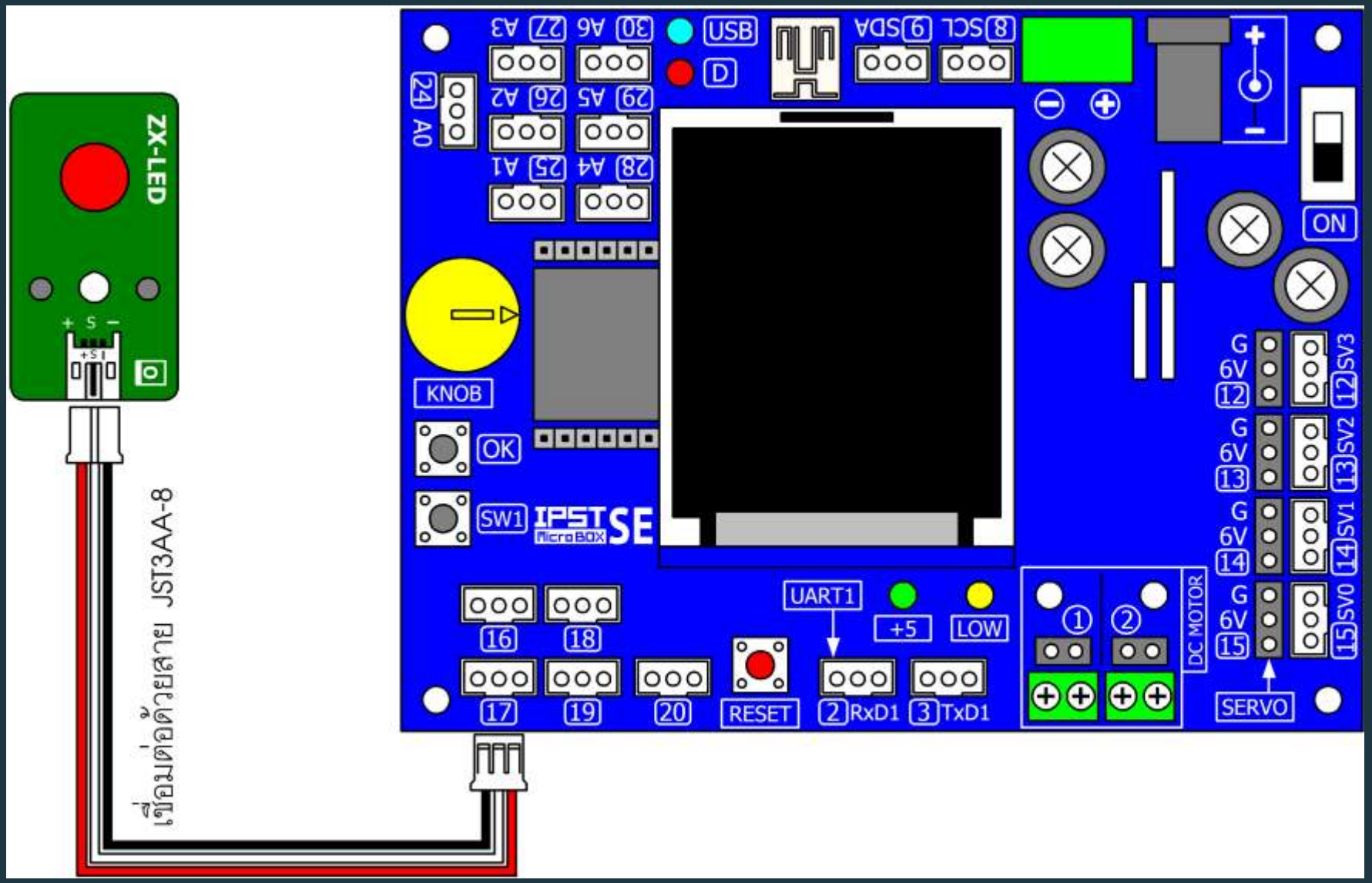
ตัวแปรใน Wiring

```
byte    0-255 (unsigned char)
word    0-65535 (unsigned int)
boolean 0-1  True  False
int     -32768>ถึง 32767
char    -128 ถึง 127
float   -3.4 x 10^38 ถึง 3.4 x 10^38
```

หาข้อมูลเพิ่มเติมจาก reference

หลอด LED

เอาต์พุตดิจิตอลอย่างง่าย



คำสั่งส่งค่าออกเอาต์พุตดิจิตอล

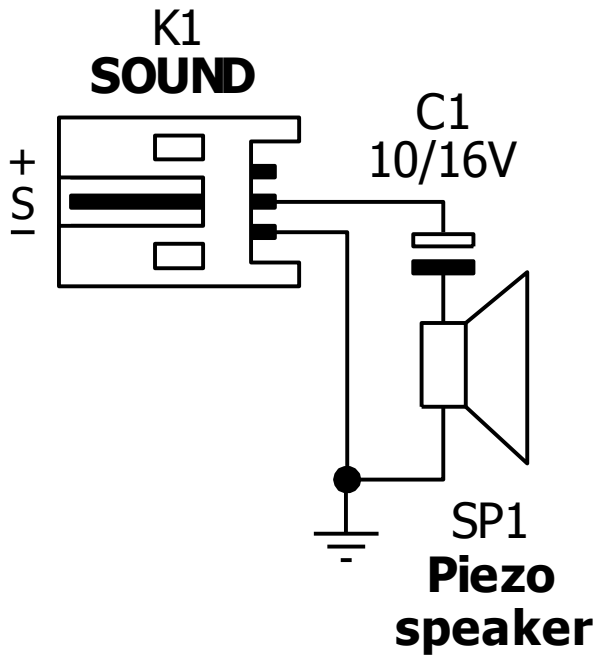
```
out(ch,state);
```

ส่งค่าสถานะ(state) 0 หรือ 1

ออกไปยังตำแหน่งขา (ch)ที่ระบุ

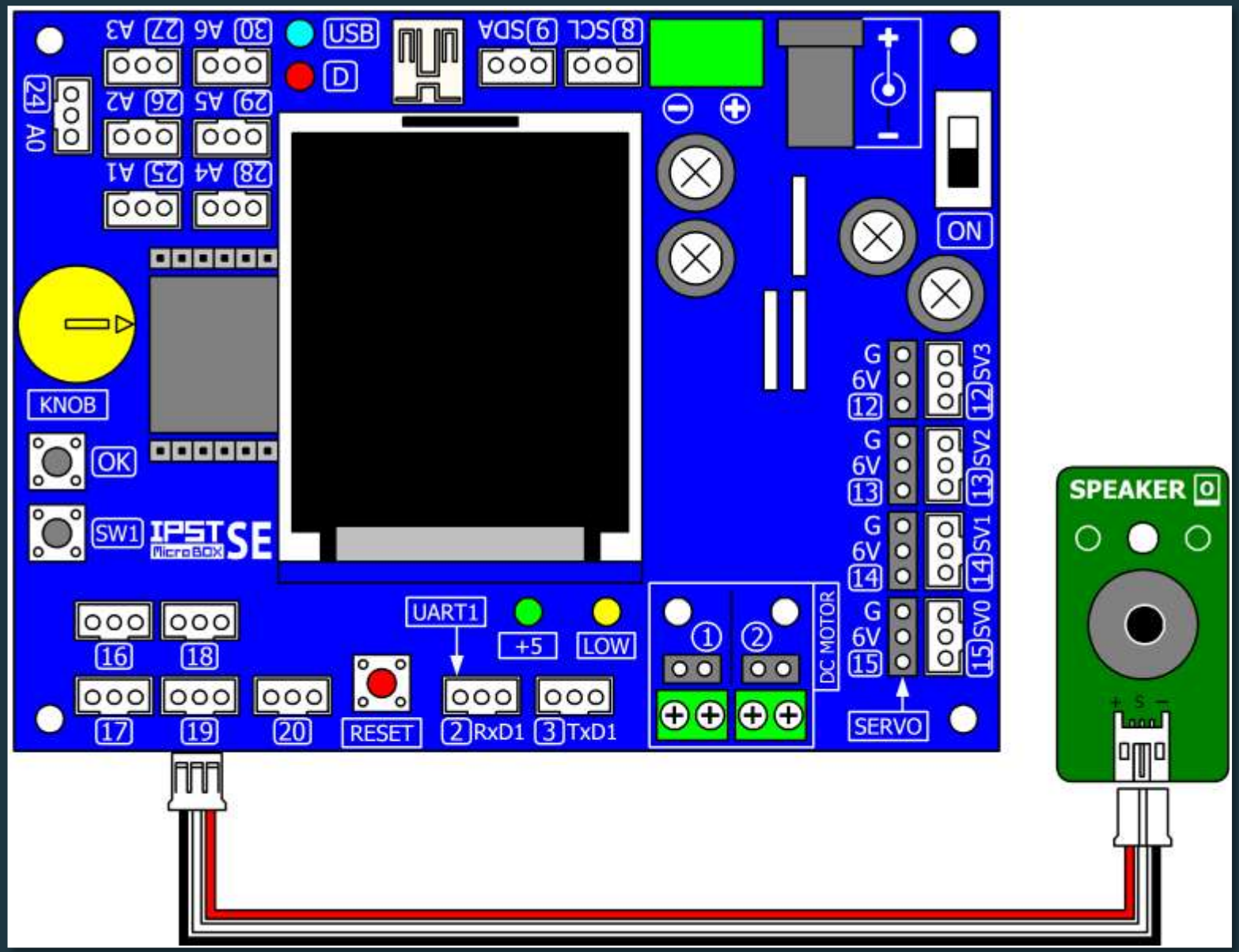
เช่น out(17,1);

แผงวงจรลำโพงเปียโซ : SPEAKER



- ใช้ลำโพงเปียโซ มีอิมพีแดนซ์ 32W
- มีค่าความถี่ย่าน 300Hz ถึง 3000 Hz

สร้างเสียงออกลำโพง



คำสั่งสร้างเสียง

ฟังก์ชั่นกำเนิดเสียงอย่างง่าย `beep` : ทำหน้าที่กำเนิดเสียงความถี่ 500 Hz นาน 100 มิลลิวินาที

```
beep(ch);
```

ฟังก์ชั่นกำเนิดเสียงความถี่ใด ๆ `sound` : ทำหน้าที่กำเนิดเสียงความถี่ ตามช่วงเวลาที่กำหนด

```
sound(ch,freq,time);
```

พารามิเตอร์ `freq` ใช้กำหนดค่าความถี่ค่าสัญญาณเสียง

`time` ใช้กำหนดช่วงเวลาในการกำเนิดสัญญาณเสียงในหน่วยมิลลิวินาที

สร้างสัญญาณเสียงติ๊ดทุก ๆ 1 วินาที (ความถี่เสียง 500 Hz ดังนาน 0.1 วินาที)

```
#include <ipst.h>
void setup(){

}
void loop(){
  beep(19);
  sleep(1000);
}
```

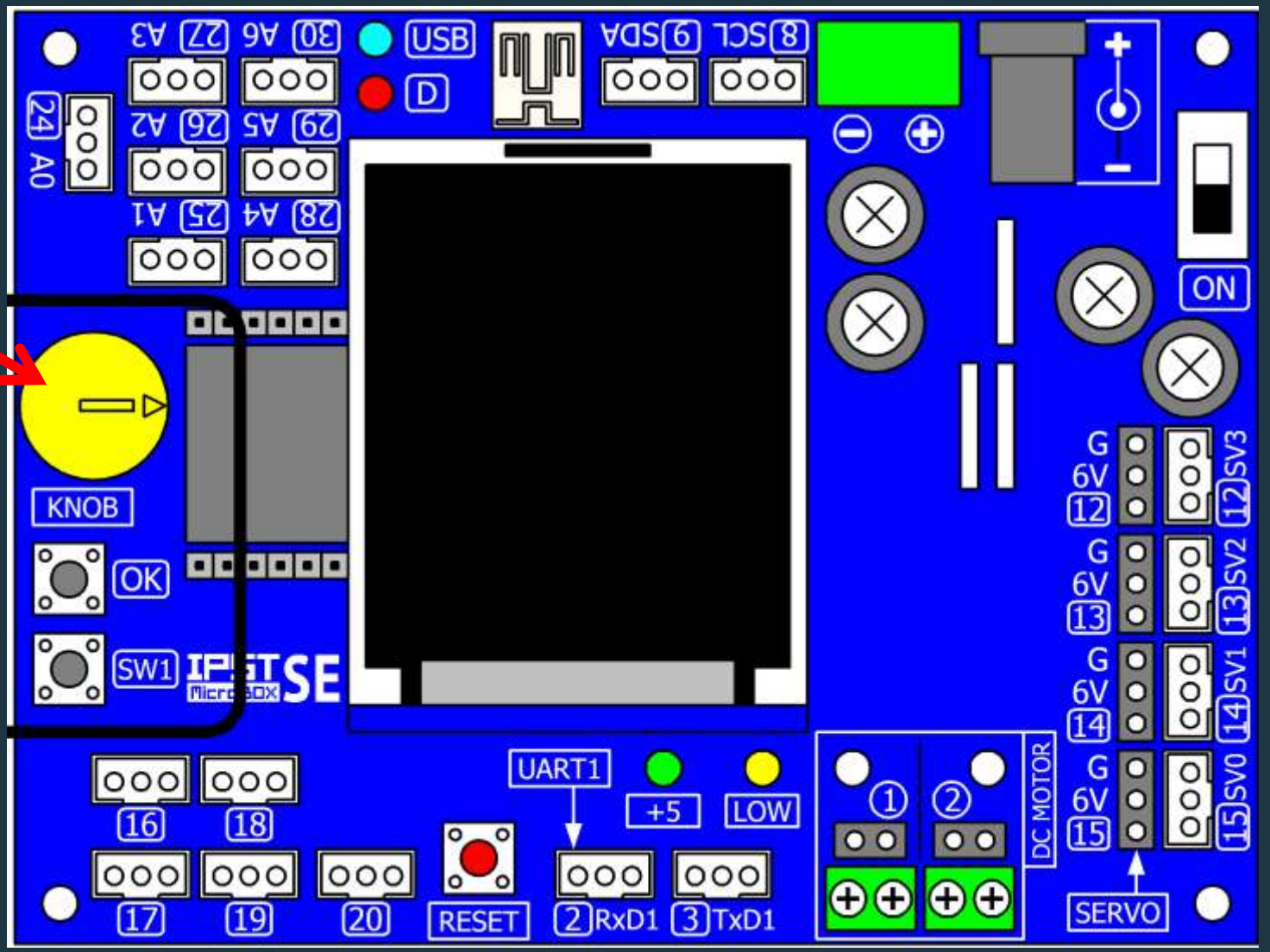
สร้างสัญญาณเสียงความถี่ 1200 Hz ดังนาน 0.5 วินาที โดยเว้นห้วงทุก ๆ 1 วินาที

```
#include <ipst.h>
void setup(){

}
void loop(){
  sound(19,1200,500);
  sleep(1000);
}
```

knob()

ปรับค่าได้ในช่วง 80-1023

knob



knob()

knob เป็นฟังก์ชั่นอ่านค่าตัวต้านทานปรับค่าได้บน IPST-SE เหมือนคำสั่ง analog(8) ค่าอยู่ในช่วง 80-1023

รูปแบบ

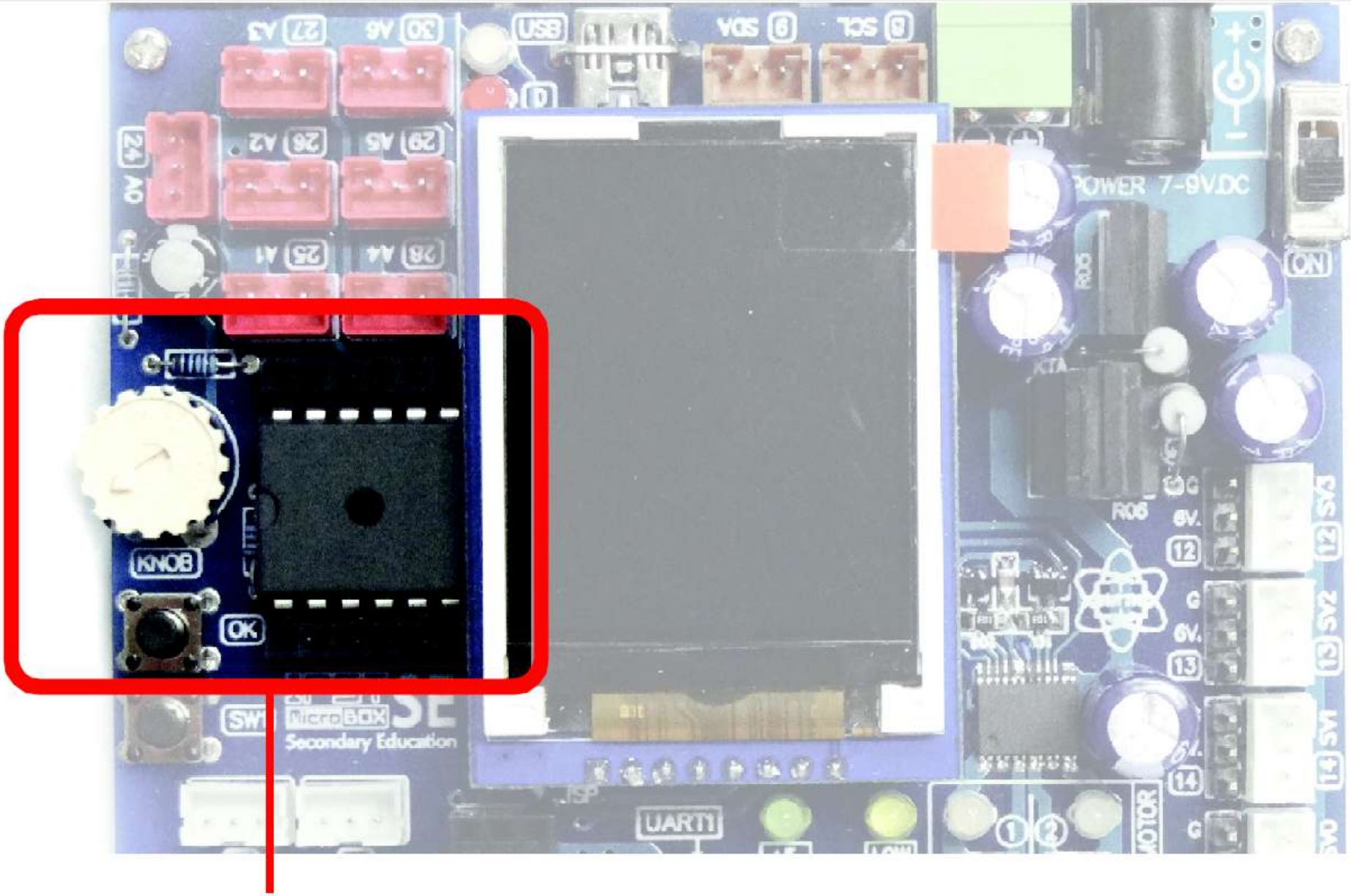
```
knob();
```

การคืนค่า

ค่าที่อ่านได้จาก knob มีค่าระหว่าง 80-1023

ตัวอย่าง

```
int val=0;       // กำหนดค่าตัวแปรสำหรับเก็บค่าอะนาลอก
val=knob();      // อ่านค่าจาก knob เก็บค่าในตัวแปร val
glcd(1,2,"%d",val); // นำค่าแสดงที่ GLCD
glcdClear();
```



ปุ่ม KNOB และสวิตช์ OK บนแผงวงจร IPST-SE ที่อ่านค่าด้วยฟังก์ชั่น knob() และ sw_OK() กับ sw_OK_press()

สวิตช์ OK บนบอร์ด

sw_OK() ตรวจสอบสวิตช์ OK บน IPST-SE ให้สถานะ True เมื่อกดสวิตช์และ False เมื่อไม่กดสวิตช์

รูปแบบ

sw_OK()

การคืนค่า

1 (True) เมื่อกดสวิตช์

0 (False) เมื่อไม่กดสวิตช์

หมายเหตุ การกดสวิตช์ทำให้ค่าที่อ่านได้จาก Knob มีค่าเป็น 0

ตัวอย่าง

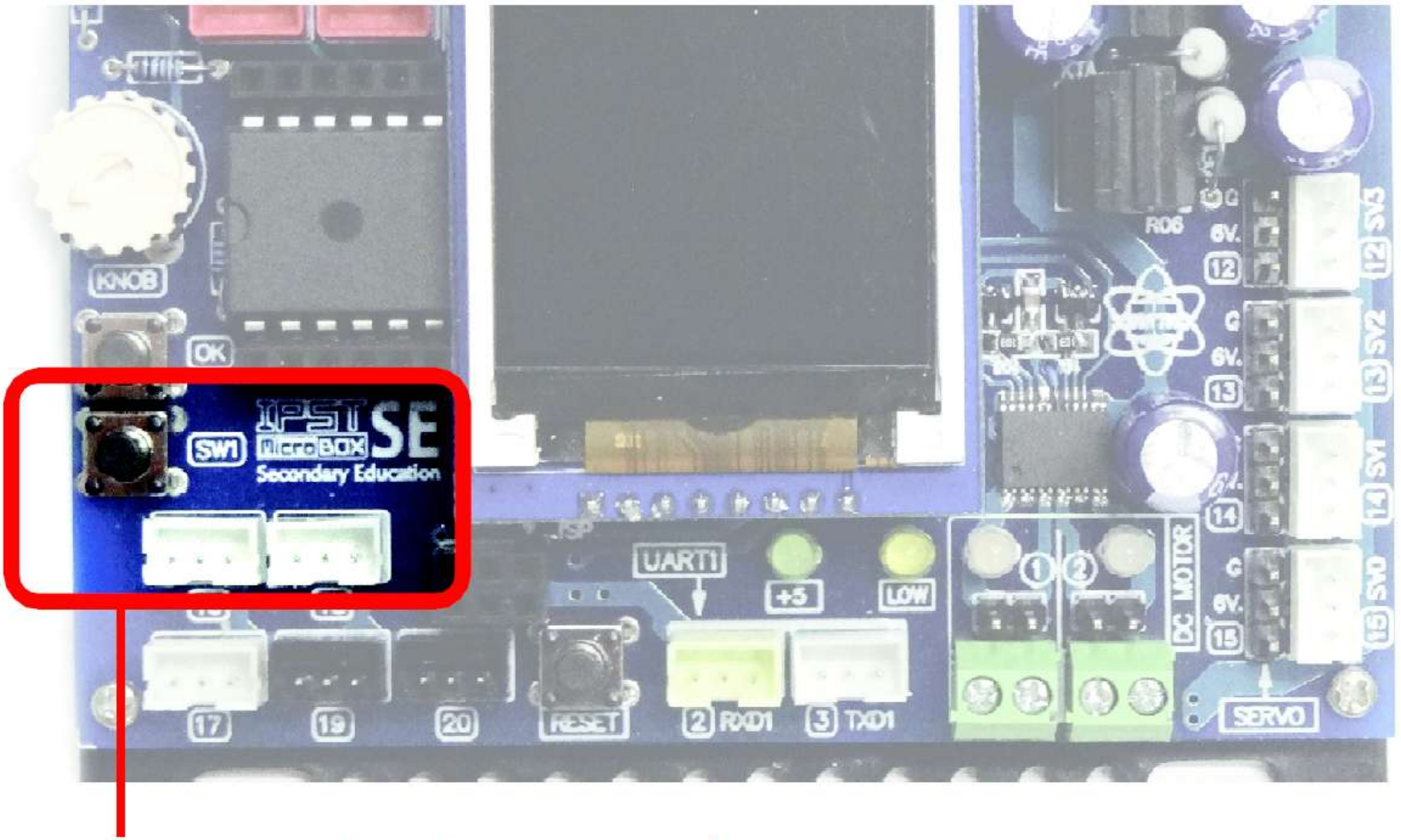
```
if(sw_OK()==1)
{
    beep(19);
}
```

สวิตช์ sw_ok_press()

เป็นฟังก์ชั่นวนตรวจสอบการกดสวิตช์ OKหรือSW1 บนบอร์ด IPST-SE ต้องรอจนกระทั่ง OKหรือSW1 ถูกปล่อยหลังจากมีการกดสวิตช์ จึงจะผ่านฟังก์ชั่นนี้ไปทำงานคำสั่งอื่น ๆ

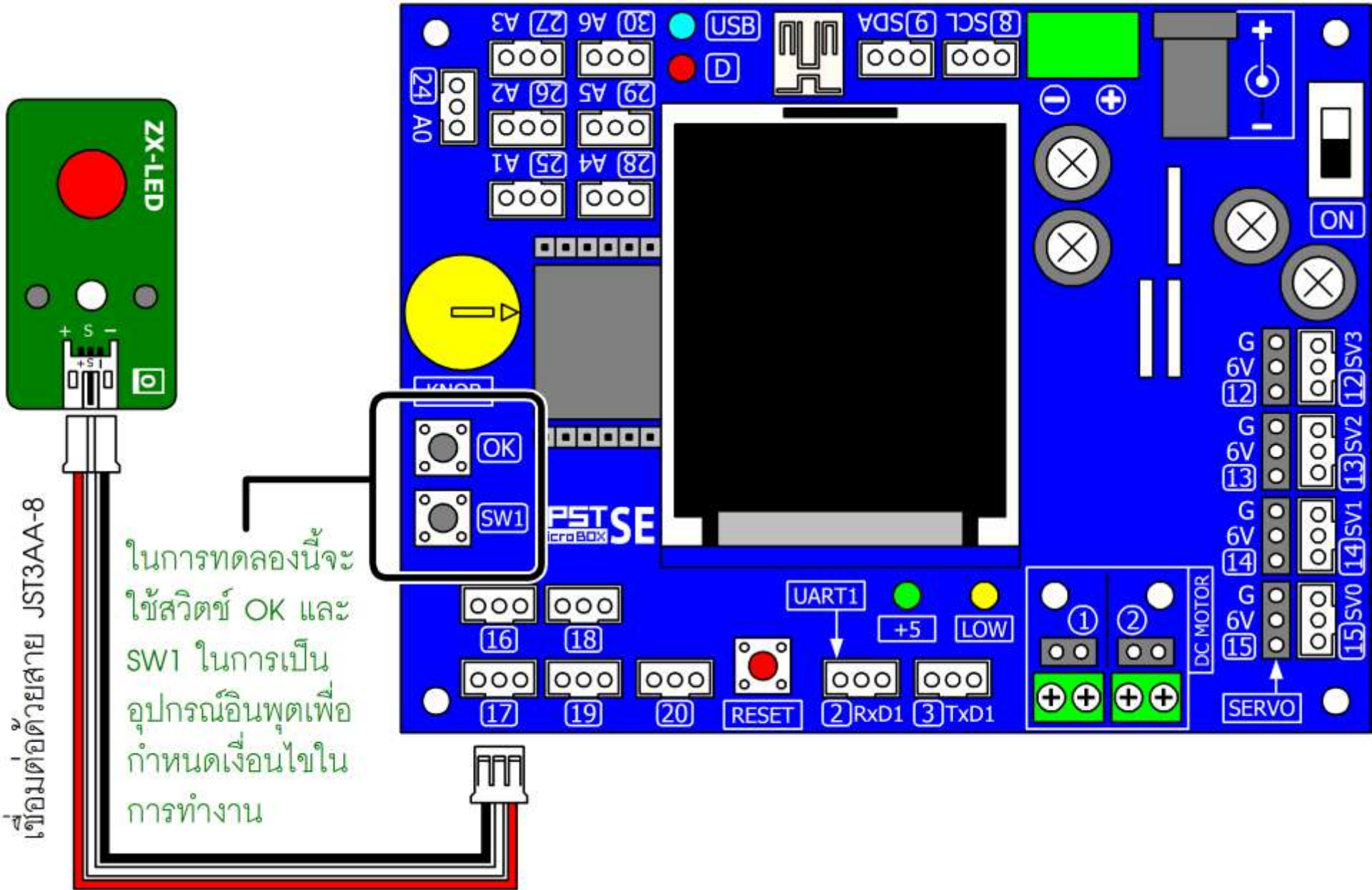
ตัวอย่าง

```
      ......
sw_OK_press();   //  รอจนกระทั่งกดสวิตช์ OK
sw1_press(); //  รอจนกระทั่งกดสวิตช์ SW1
      ......
```

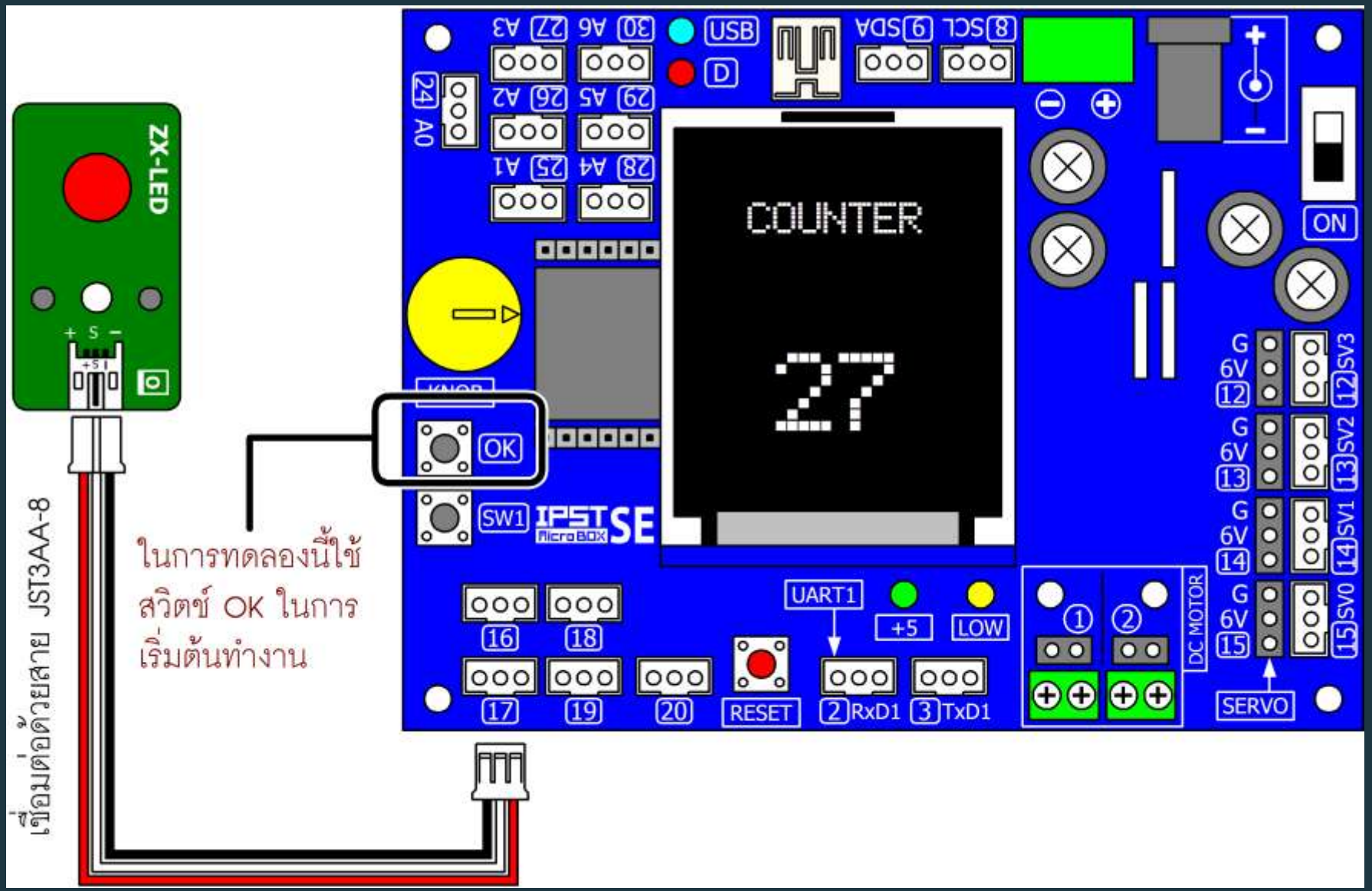


สวิตช์ SW1 อ่านค่าด้วยฟังก์ชั่น sw1() และ sw1_press()

สวิตช์สองตัวควบคุมการเปิดปิดหลอดไฟ

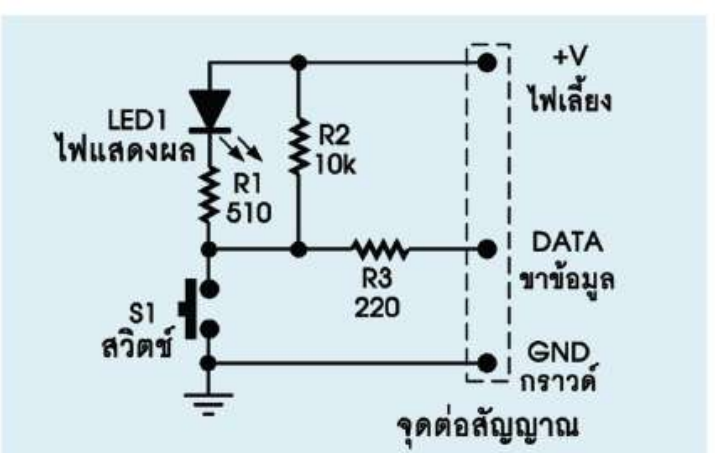


เขียนโปรแกรมใช้สวิตช์ OK กับ LED



แผงวงจรสวิตช์: ZX–SWITCH

ลักษณะของแผงวงจรสวิตช์



วงจรของแผงวงจรสวิตช์

คุณสมบัติทางเทคนิค

- ถ้าสวิตช์ถูกกดจะอ่านค่าข้อมูลได้เป็นลอจิก “0” พร้อมกับไฟแสดงสถานะติดสว่าง
- ถ้าสวิตช์ไม่ถูกกดจะอ่านค่าข้อมูลได้เป็นลอจิก “1”

ฟังก์ชั่น in

สำหรับอ่านค่าสัญญาณแบบดิจิตอลจากขาพอร์ตใด ๆของบอร์ดควบคุมหลัก

รูปแบบ

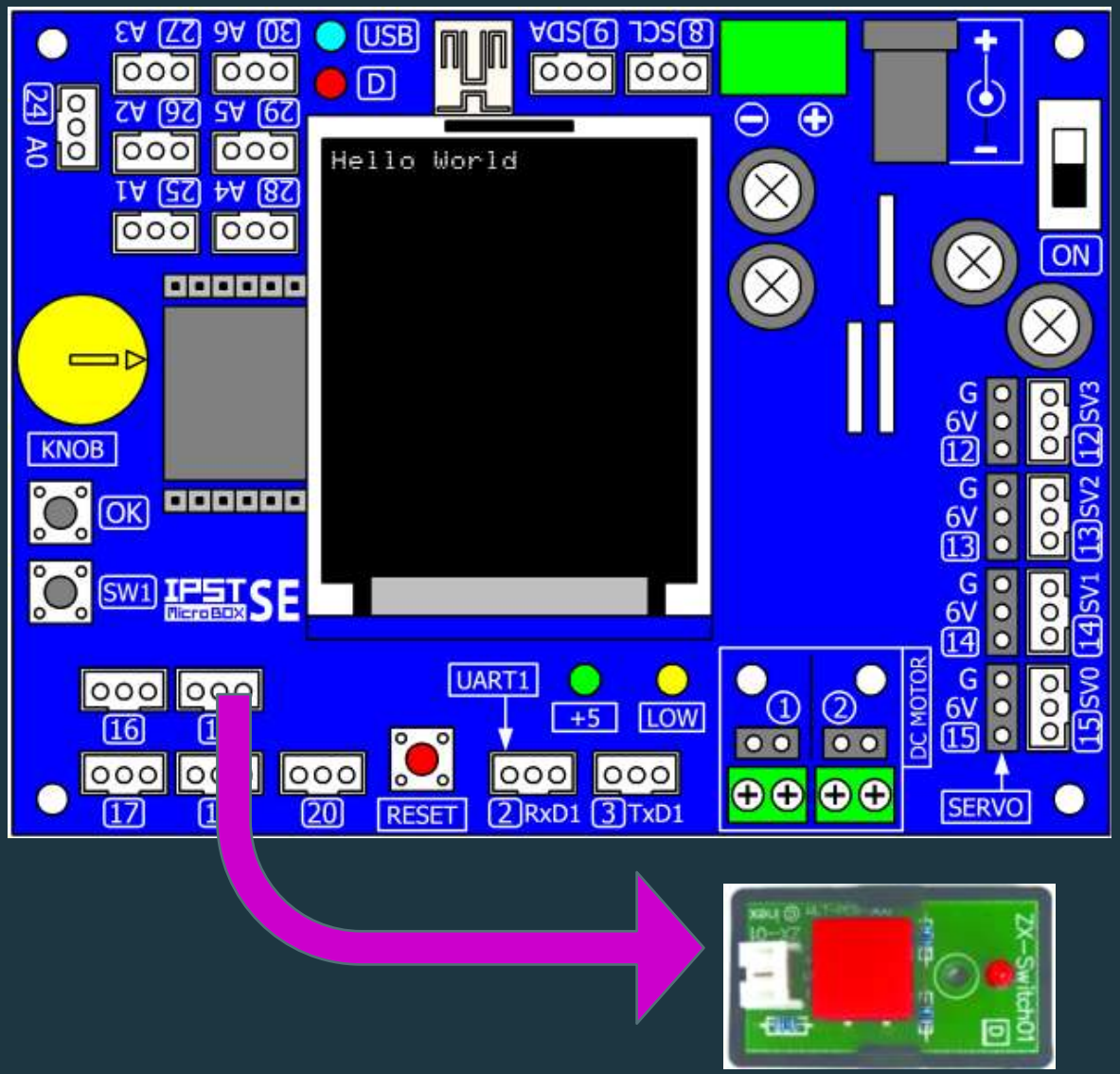
```
char in(char _bit);
```

พารามิเตอร์ `_bit` ใช้กำหนดตำแหน่งหมายเลขพอร์ตที่ต้องการติดต่อ

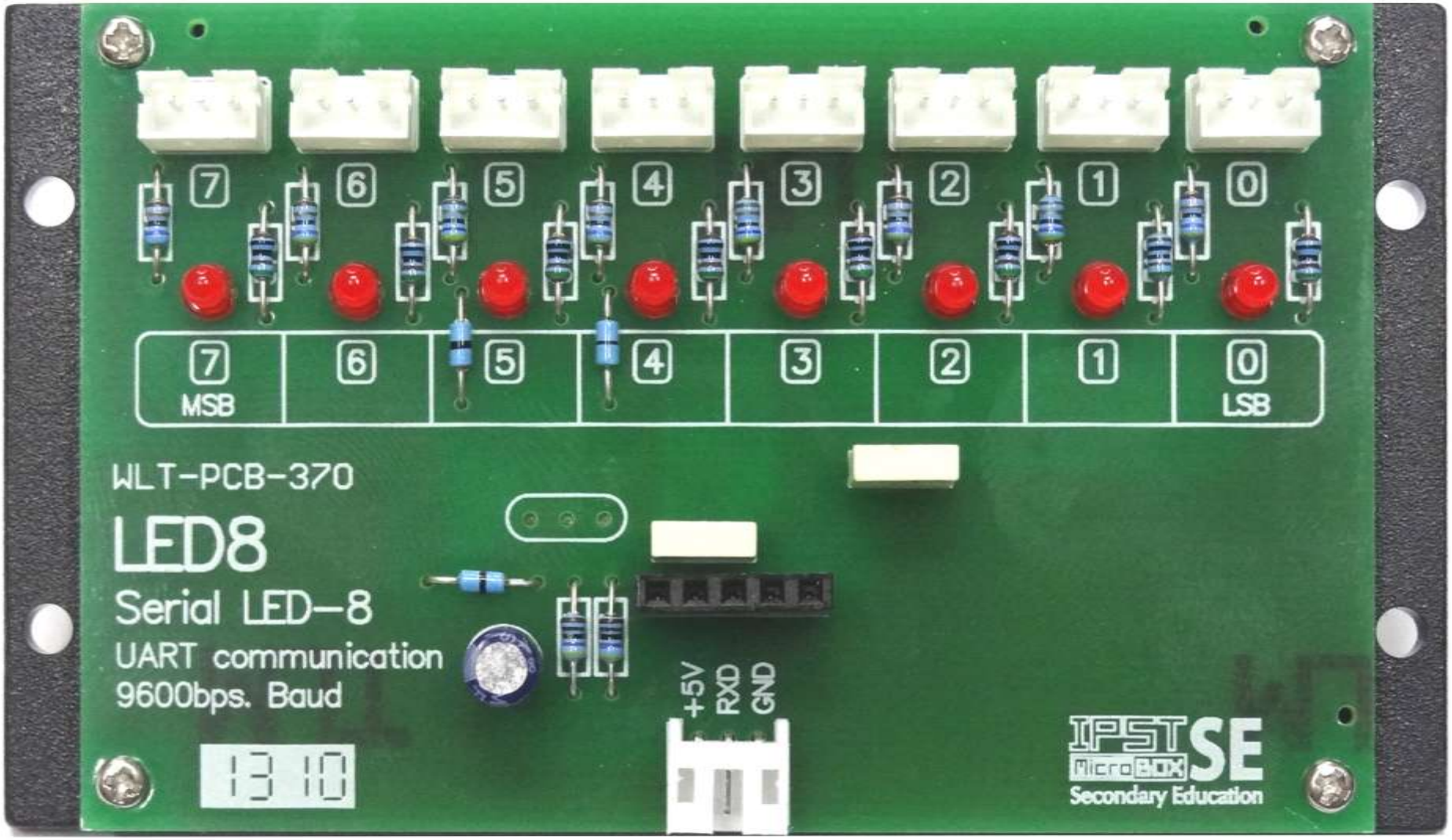
การคืนค่า ฟังก์ชั่นจะทำการคืนค่าสัญญาณดิจิตอลของตำแหน่งขาพอร์ตที่อ่านซึ่งอาจมีค่าเป็น 0 หรือ 1 เท่านั้น

```
sw_OK_press();
Sw1_press();
in(14);
analog(0);
knob();
```

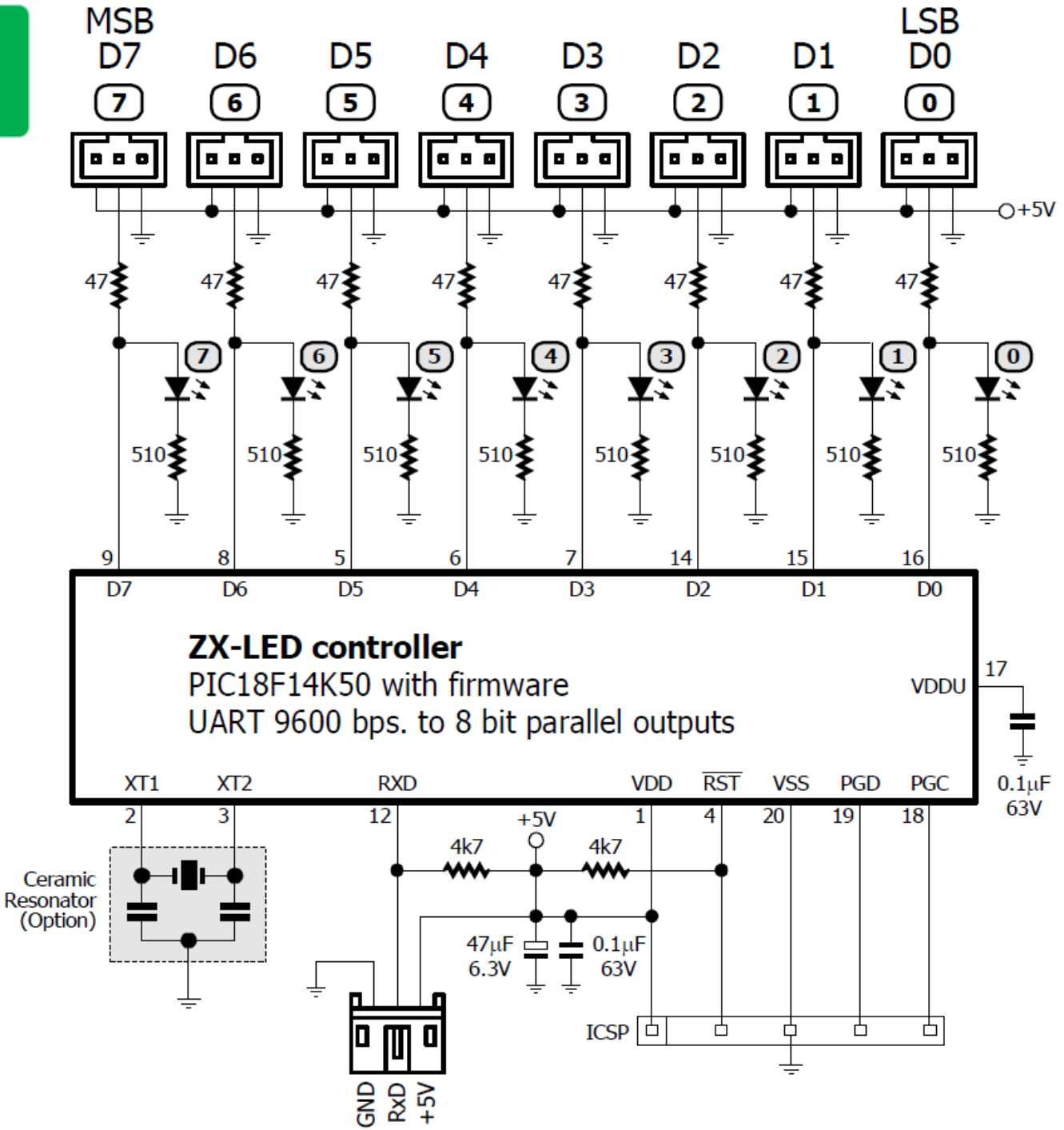
ทดสอบเขียนโปรแกรมกับสวิตช์

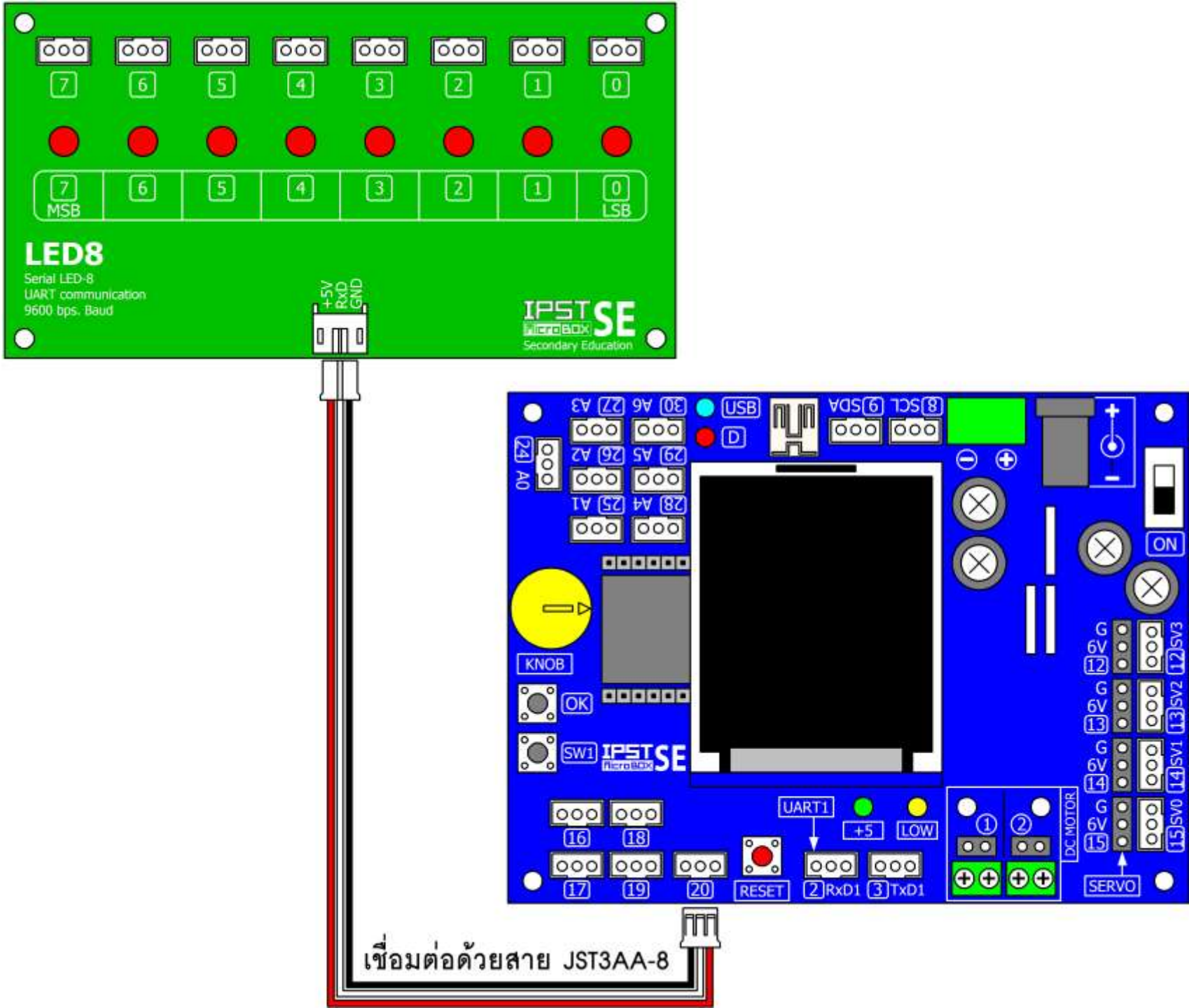


LED8


7
6
5
4
3
2
1
0
7
MSB
6
5
4
3
2
1
0
LSB
WLT-PCB-370
LED8
Serial LED-8
UART communication
9600bps. Baud
+5V
RXD
GND
IPST
MicroBOX
SE
Secondary Education

วงจร LED8


MSB
D7
D6
D5
D4
D3
D2
D1
LSB
D0
7
6
5
4
3
2
1
0
+5V
47
510
9
8
5
6
7
14
15
16
ZX-LED controller
PIC18F14K50 with firmware
UART 9600 bps. to 8 bit parallel outputs
VDDU
17
0.1µF
63V
XT1
XT2
RXD
VDD
RST
VSS
PGD
PGC
2
3
12
1
4
20
19
18
Ceramic
Resonator
(Option)
4k7
47µF
6.3V
0.1µF
63V
ICSP
GND
RxD
+5V


7 6 5 4 3 2 1 0
7 MSB
6
5
4
3
2
1
0 LSB
LED8
Serial LED-8
UART communication
9600 bps. Baud
+5V
RxD
GND
IPST MicroBOX SE
Secondary Education
USB
D
SCL 9 SDA 8
A3 27 A6 30
A2 26 A5 29
A1 25 A4 28
24 A0
KNOB
OK
SW1
IPST MicroBOX SE
ON
G 6V 12 SV3
G 6V 13 SV2
G 6V 14 SV1
G 6V 15 SV0
SERVO
DC MOTOR
16 18
17 19 20
RESET
UART1
+5
LOW
2 RxD1 3 TxD1
เชื่อมต่อด้วยสาย JST3AA-8

ฟังก์ชั่น LED8()

สำหรับส่งค่าออกไปยังบอร์ด LED8 ในรูปแบบข้อมูล 1 ไบต์

รูปแบบ

```
LED8(pin,dat);
```

พารามิเตอร์ **pin** ใช้กำหนดตำแหน่งหมายเลขพอร์ตที่ต้องการติดต่อ

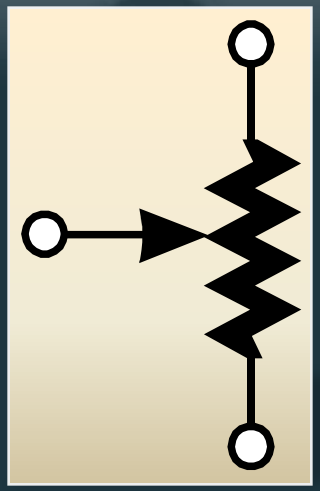
dat ข้อมูลขนาด 1 ไบต์ที่จะส่งไปยัง LED 8 ดวง ค่า 0 LED ดับหมด ค่า 255 LED ติดทั้งหมด

แผงวงจรตัวต้านทานปรับค่าได้

ตัวต้านทานปรับค่าได้แบบตัวนอน

ตัวต้านทานปรับค่าได้แบบตัวตั้ง

ตัวต้านทานปรับค่าได้แบบเลื่อน



สัญลักษณ์

ZX-POTV
A
ZX-POTH
A
USB
D
SCL 8
SDA 9
30 A6
29 A5
28 A4
25 A1
24 A0
Analog
583
Volts
2.847
KNOB
OK
SW1
IPST
MicroBOX
SE
ON
G
6V
12
SV3
13
SV2
14
SV1
15
SV0
SERVO
DC MOTOR
UART1
+5
LOW
16
18
17
19
20
RESET
2 RxD1
3 TxD1

ฟังก์ชั่น analog()

อ่านค่าสัญญาณอะนาลอก จากตำแหน่งพอร์ตที่ระบุ (A0-A6)

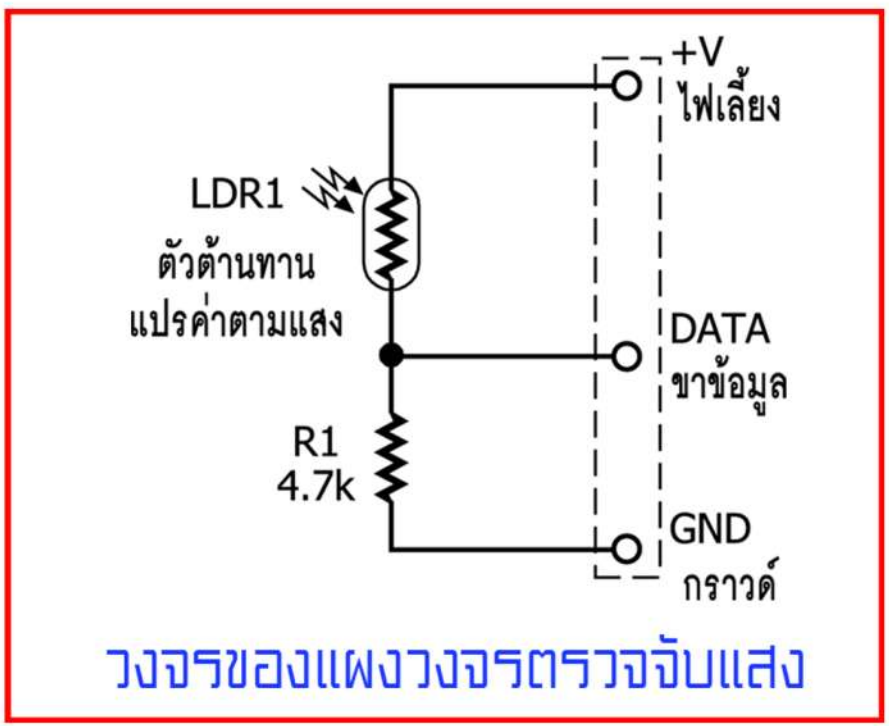
รูปแบบ

```
analog(ch);
```

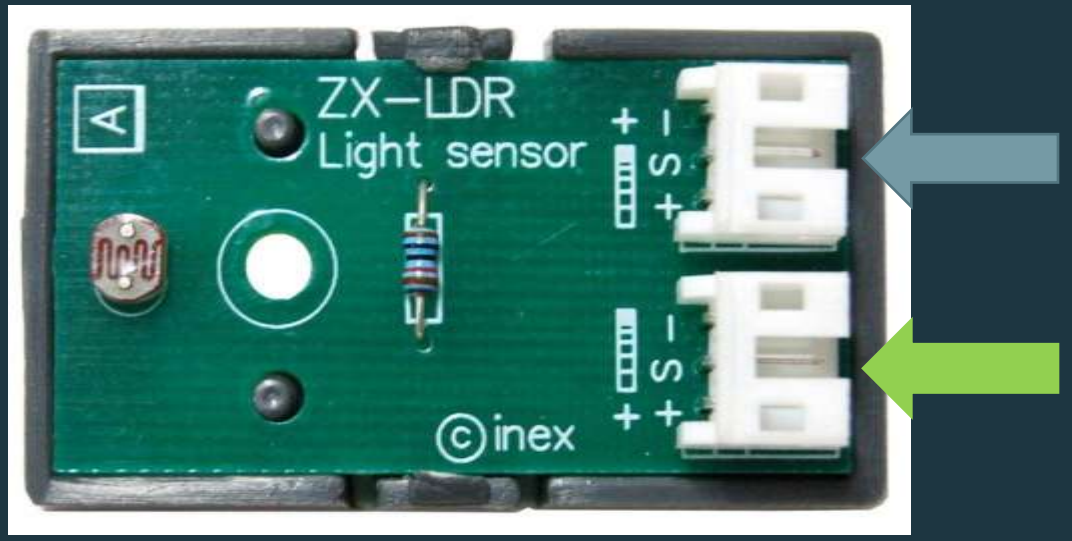
พารามิเตอร์ **`ch`** คือขาพอร์ตอะนาลอกในตำแหน่งที่ต้องการอ่านค่า

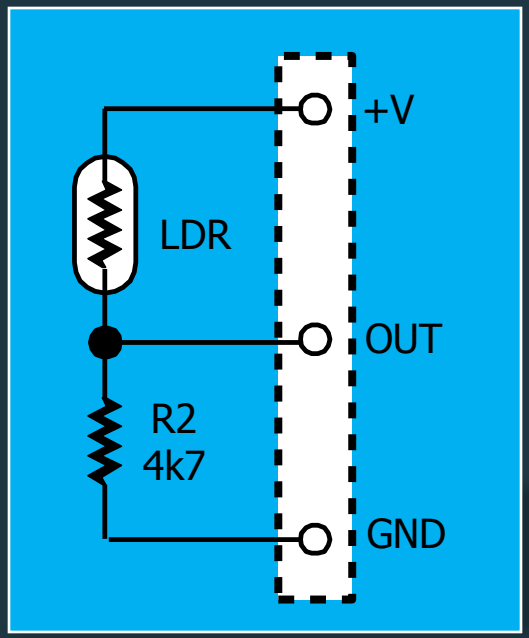
การคืนค่า ค่า 0-1023 (10 บิต) จากตำแหน่งขาพอร์ตที่ต้องการ

แผงวงจรตรวจจับแสง LDR



แผงวงจรตรวจจับแสง LDR

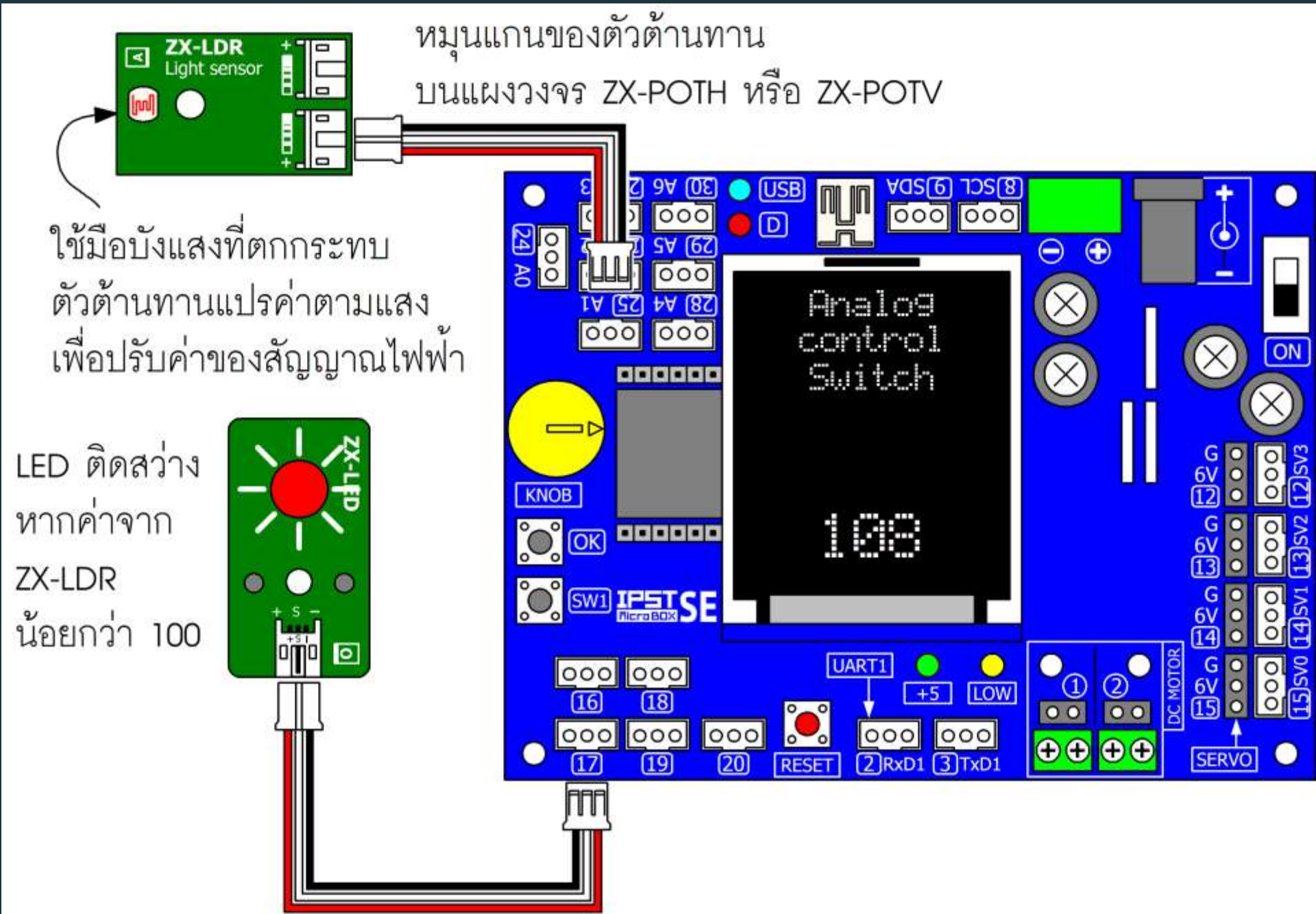




ใช้ตรวจจับแสงสว่าง เลือกเอาต์พุตได้ 2 แบบคือ

แรงดันเอาต์พุตเพิ่ม เมื่อแสงตกกระทบมากขึ้น

แรงดันเอาต์พุตลดลง เมื่อแสงตกกระทบมากขึ้น

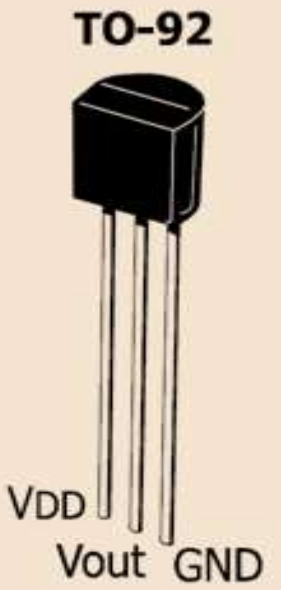

ZX-LDR
Light sensor
หมุนแกนของตัวต้านทาน
บนแผงวงจร ZX-POTH หรือ ZX-POTV
ใช้มือบังแสงที่ตกกระทบ
ตัวต้านทานแปรค่าตามแสง
เพื่อปรับค่าของสัญญาณไฟฟ้า
LED ติดสว่าง
หากค่าจาก
ZX-LDR
น้อยกว่า 100
ZX-LED
Analog
control
Switch
108
KNOB
OK
SW1
IPST
MicroBOX
SE
USB
D
UART1
+5
LOW
RESET
2 RxD1
3 TxD1
DC MOTOR
SERVO
ON

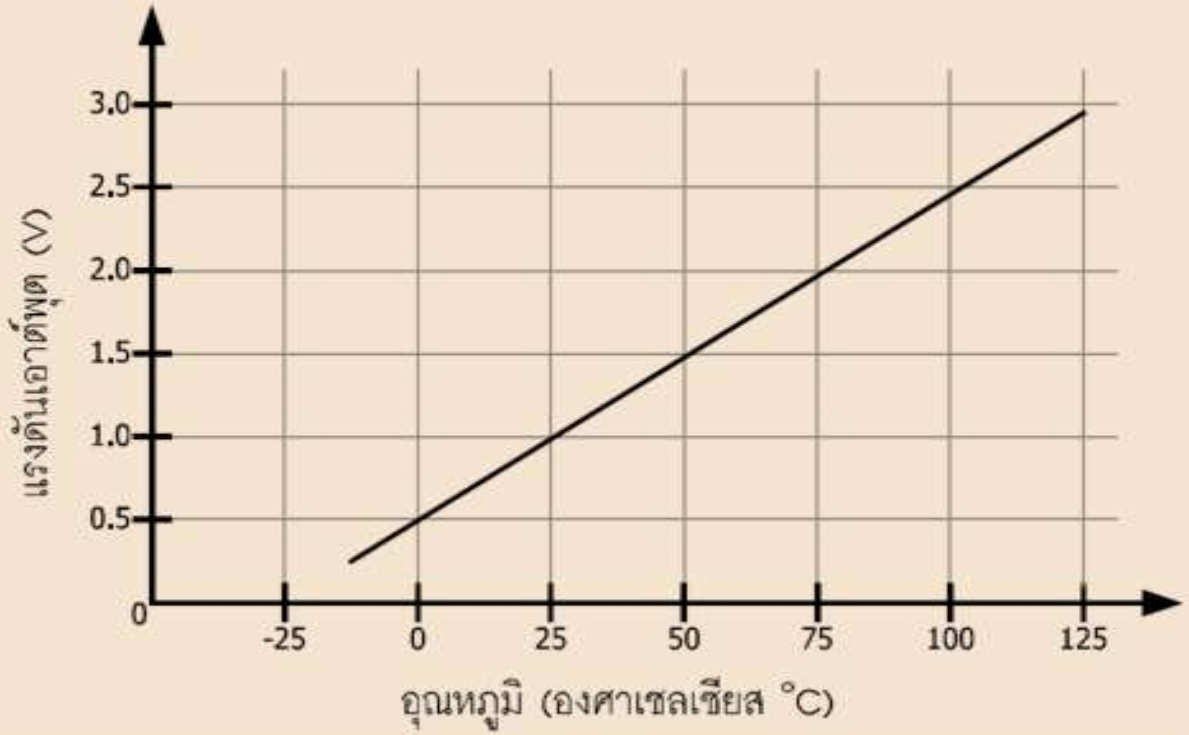
ไอซีวัดอุณหภูมิ MCP9701

แรงดันเอาต์พุตเปลี่ยนแปลง 19.5mV/องศา

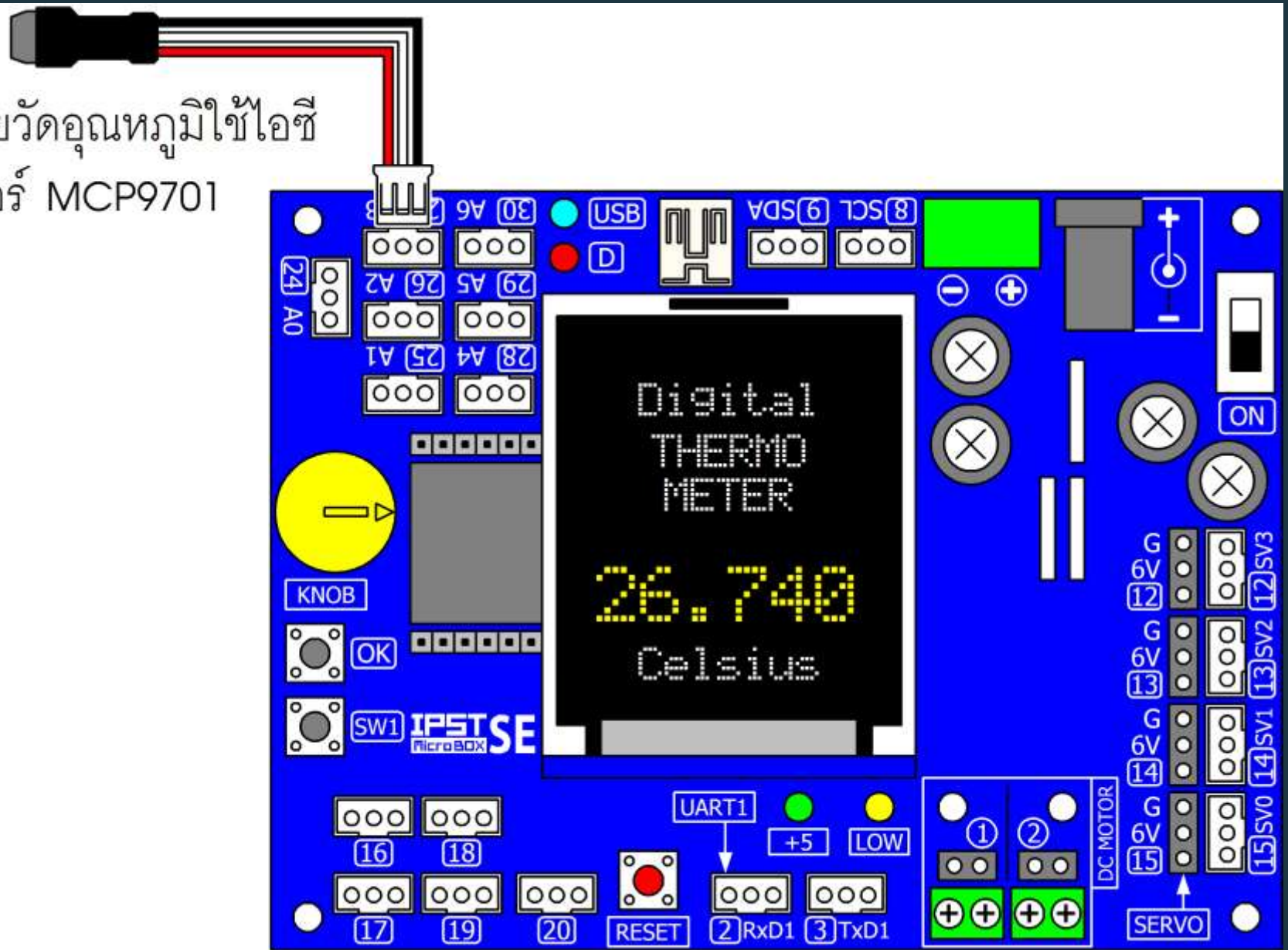
สามารถคำนวณค่าจากค่าอะนาลอกที่อ่านได้จากสูตร

Temp = (val x 0.25) - 20.51



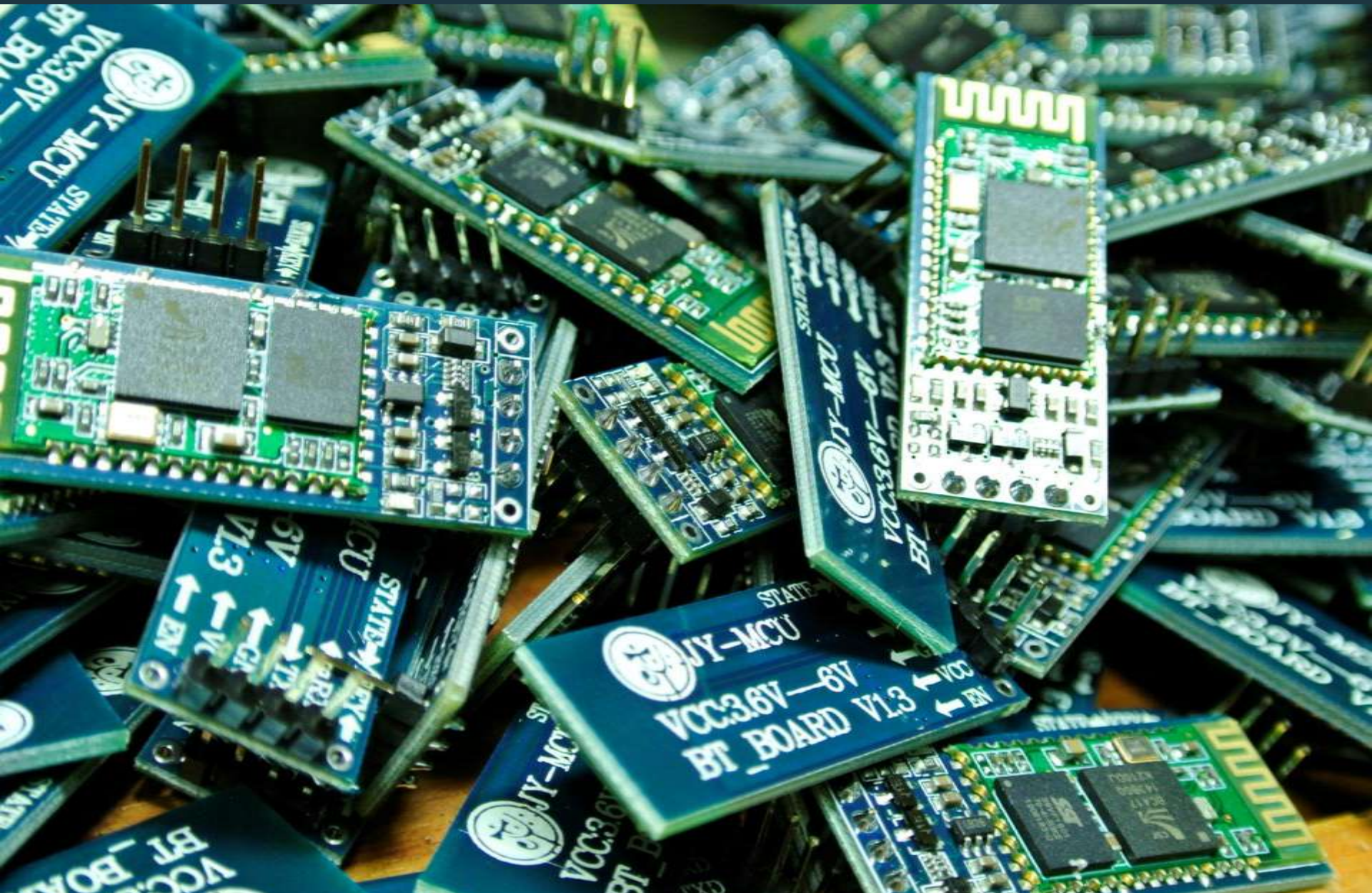


MCP9701

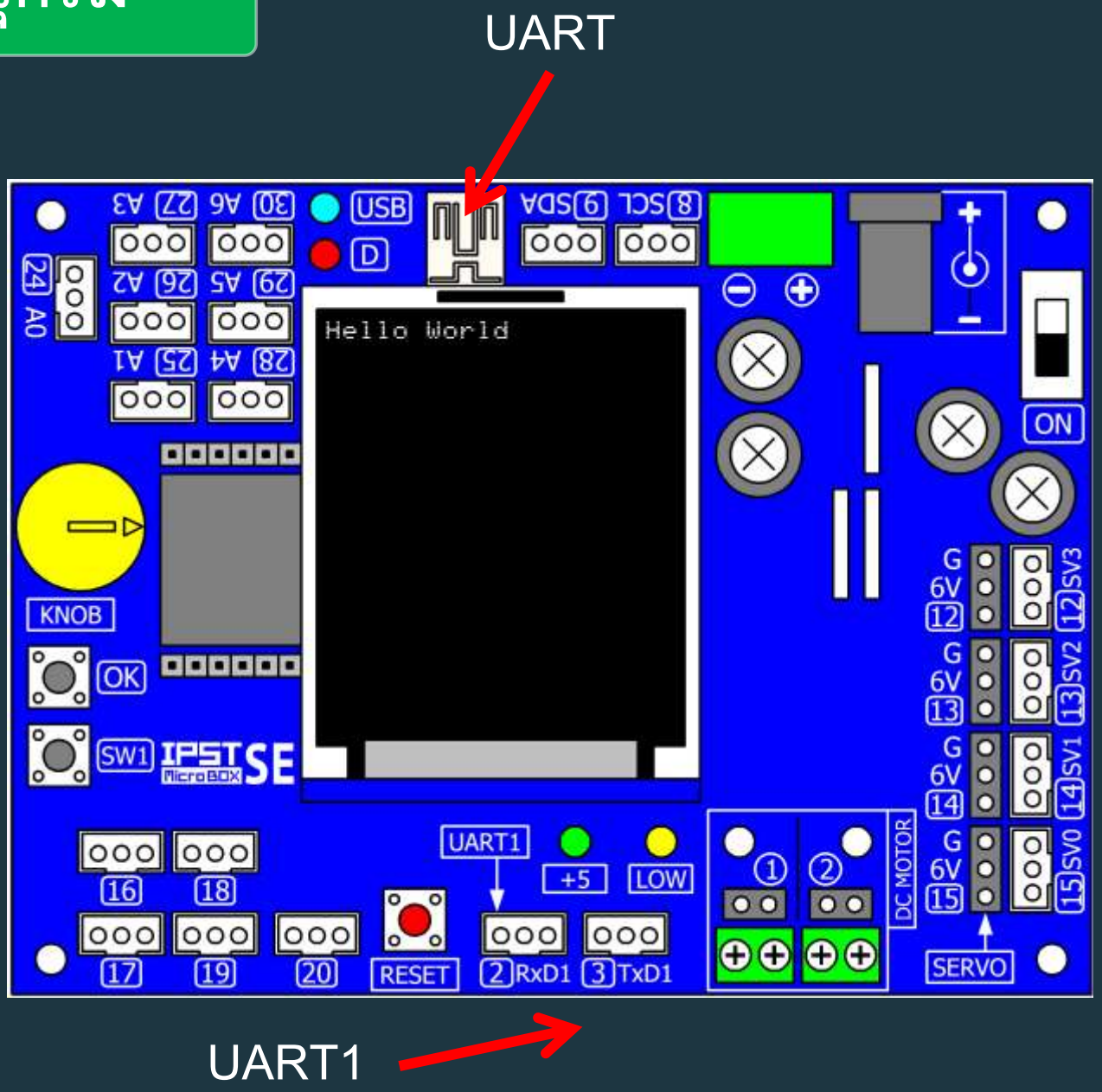

สายวัดอุณหภูมิใช้ไอซี
เบอร์ MCP9701
USB
D
Digital
THERMO
METER
26.740
Celsius
KNOB
OK
SW1
IPST
MicroBOX
SE
UART1
+5
LOW
RESET
2 RxD1
3 TxD1
DC MOTOR
SERVO
ON
A0
A1
A2
A4
A5
A6
SCL
SDA
G
6V
SV0
SV1
SV2
SV3

```
#include <ipst.h>
int val,i;
float Temp;
void setup(){
  glcdClear();
  setTextSize(2);
}
void loop(){
  glcd(1,2,"Digital");
  glcd(2,2,"THERMO");
  glcd(3,3,"METER");
  val=0;
  for (i=0;i<20;i++)  {val = val+analog(3);}
  val = val/20;
  Temp = (float(val)*0.25) - 20.51 ;
  setTextSize(3);
  setTextColor(GLCD_YELLOW);
  glcd(3,1,"%f",Temp);
  setTextColor(GLCD_WHITE);
  setTextSize(2);
  glcd(6,2,"Celsius");
  delay(500);
}
```

การสื่อสารอนุกรม



การสื่อสารอนุกรม



การสื่อสารอนุกรม

ฟังก์ชั่น

uart_available() ถ้ามีข้อมูลถูกป้อนเข้ามาเงื่อนไขเป็นจริง

uart_getkey() *รับค่าข้อมูล 1 ไบต์*

uart *ส่งข้อมูลหลายๆ ไบต์ออกไป*

uart1_available() *ถ้ามีข้อมูลถูกป้อนเข้ามาเงื่อนไขเป็นจริง*

uart1_getkey() *รับค่าข้อมูล 1 ไบต์*

uart1 *ส่งข้อมูลหลายๆ ไบต์ออกไป*

ไฟล์ไลบรารี ipst.h

จอแสดงผลกราฟิก LCD สี

ipst_glcd.h
- glcd
- setTextColor
- setTextBackgroundColor
- glcdClear
- glcdFillScreen
- glcdMode
- setTextSize
- getTextColor
- getTextBackgroundColor
- getTextSize
- glcdGetMode
- glcdPixel
- glcdRect
- glcdFillRect
- glcdLine
- glcdCircle
- glcdFillCircle
- glcdArc

ipst_sound.h
- beep
- sound

แผงวงจรขับลำโพง

ipst_led8.h
- pinLED8
- LED8

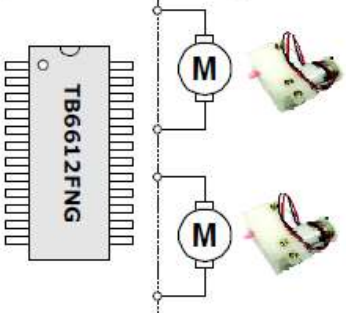
แผงวงจร LED8

ipst_servoMotor.h
- servo

เซอร์โวมอเตอร์

ipst_motor.h
- motor
- ao
- motor_stop
- fd
- fd2
- bk
- bk2
- tl
- tr
- sl
- sr

วงจรขับมอเตอร์ไฟตรง



รับส่งข้อมูลอนุกรมผ่านหน้าต่าง Serial Monitor ของ Wiring IDE

ipst_serial.h
- uart
- uart_putc
- uart_puts
- uart_set_baud
- uart_get_baud
- uart_available
- uart_getkey

- uart1
- uart1_putc
- uart1_puts
- uart1_set_baud
- uart1_get_baud
- uart1_available
- uart1_getkey

แผงวงจรขับเซอร์โวมอเตอร์ 16 ช่อง

โมดูลบลูทูธ

โมดูล XBEE

ipst_in_out.h
- in
- out
- sw1

แผงวงจรขับ LED ZX-LED

แผงวงจรสวิตช์ ZX-SWITCH01

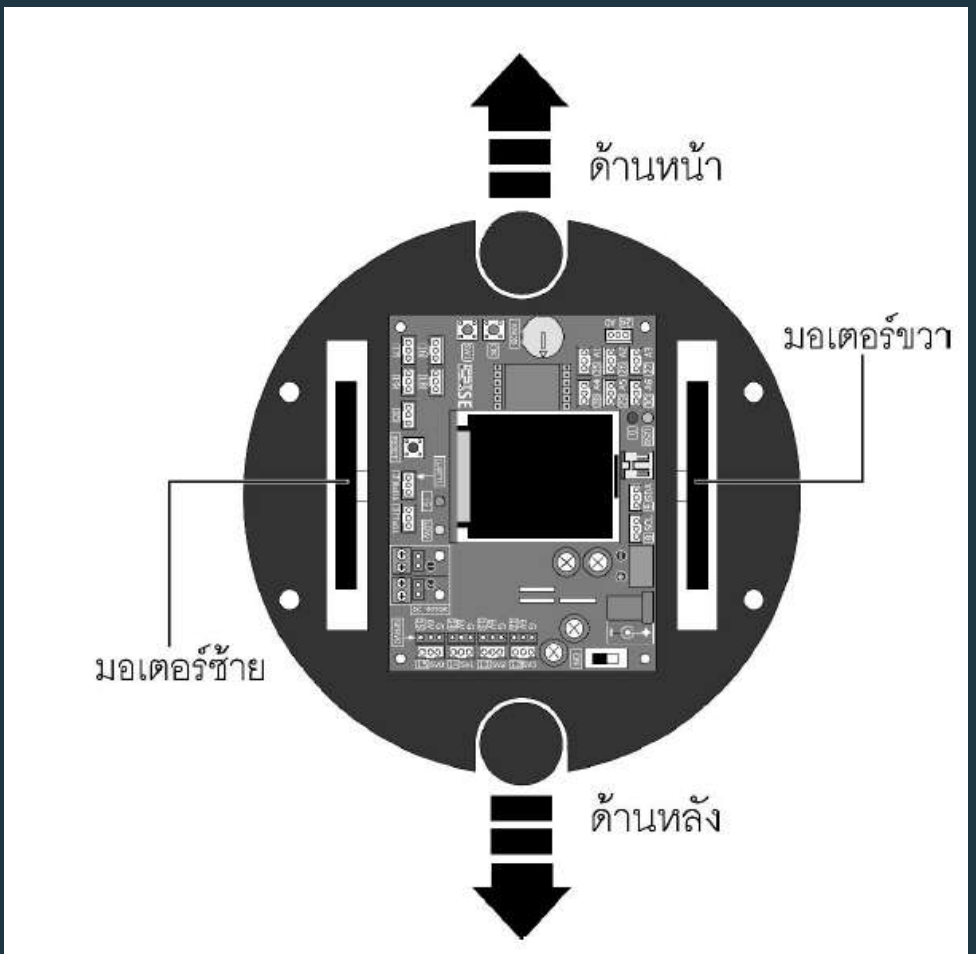
ipst_analog.h
- analog
- knob
- sw_ok
- sw_ok_press

ตัวตรวจจับแบบต่าง ๆ ที่ให้ผลการทำงานเป็นแรงดันไฟฟ้า

ipst_sleep.h
- sleep
- delay
- delay_us

ขับเคลื่อนมอเตอร์เบื้องต้น

ใช้การสั่งงานมอเตอร์ 2 ตัว คือ DC Motor 1(ซ้าย) และ DC Motor 2(ขวา)



คำสั่งเดินหน้า

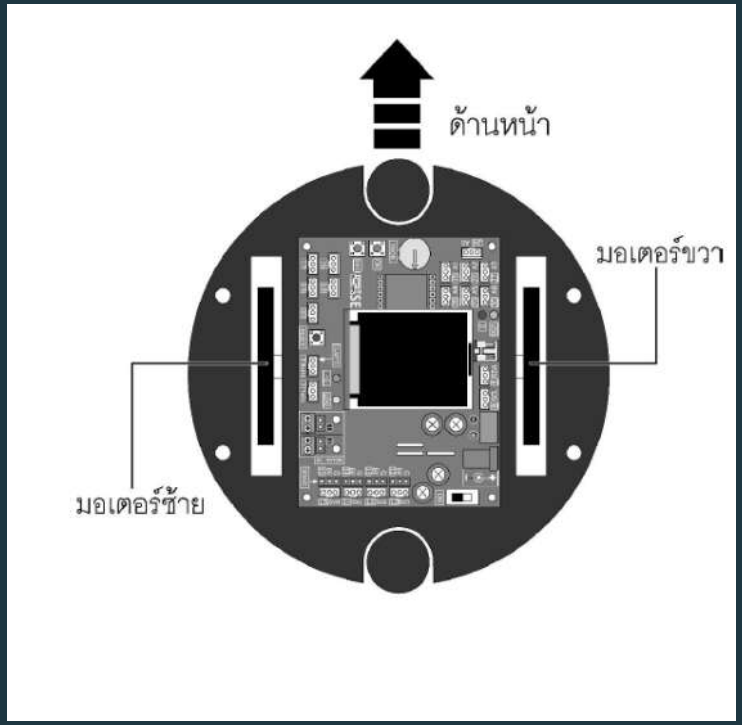
```
motor(1,Speed);//ซ้าย
motor(2,Speed);//ขวา
```

คำสั่งถอยหลัง

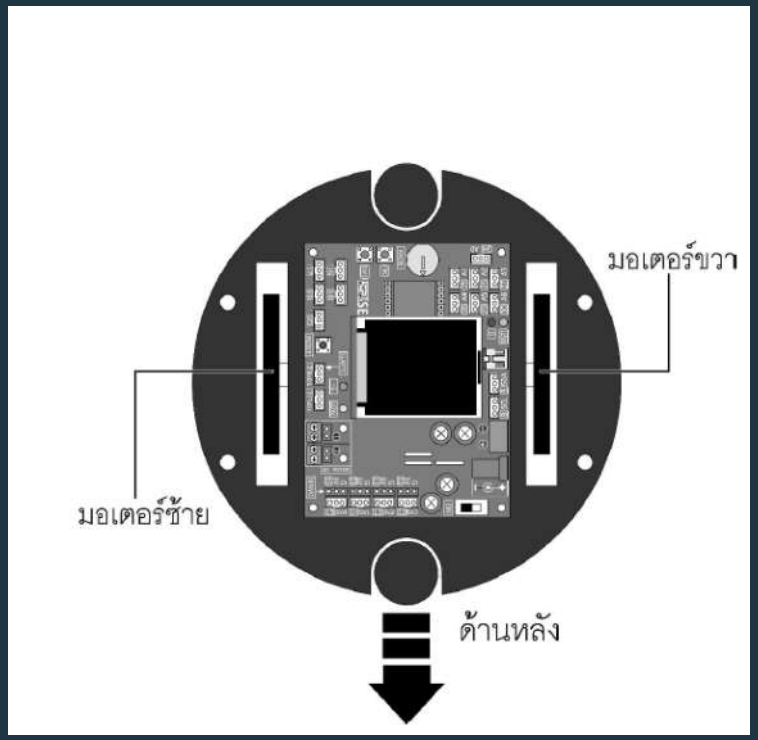
```
motor(1,-Speed);

motor(2,-Speed);
```

ขับเคลื่อนหุ่นยนต์ไปด้านหน้า

```
#include <ipst.h>
void setup()
{
  sw_OK_press();
}
void loop()
{
    motor(1,40);//ซ้าย
    motor(2,40);//ขวา
   sleep(1000);
}
```

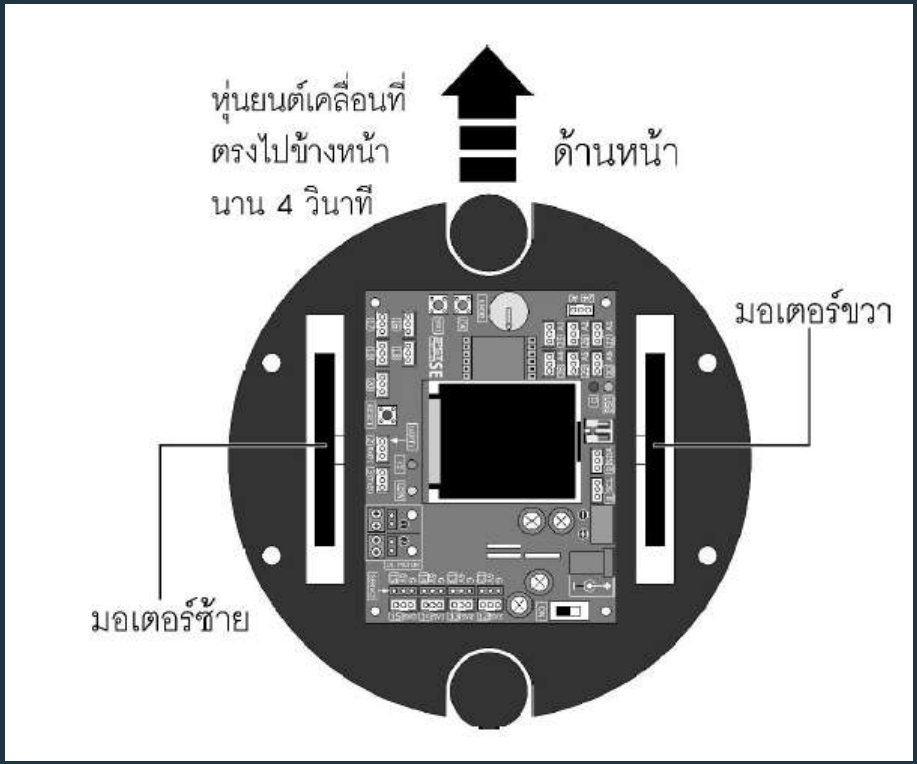


ขับเคลื่อนหุ่นยนต์ไปด้านถอยหลัง



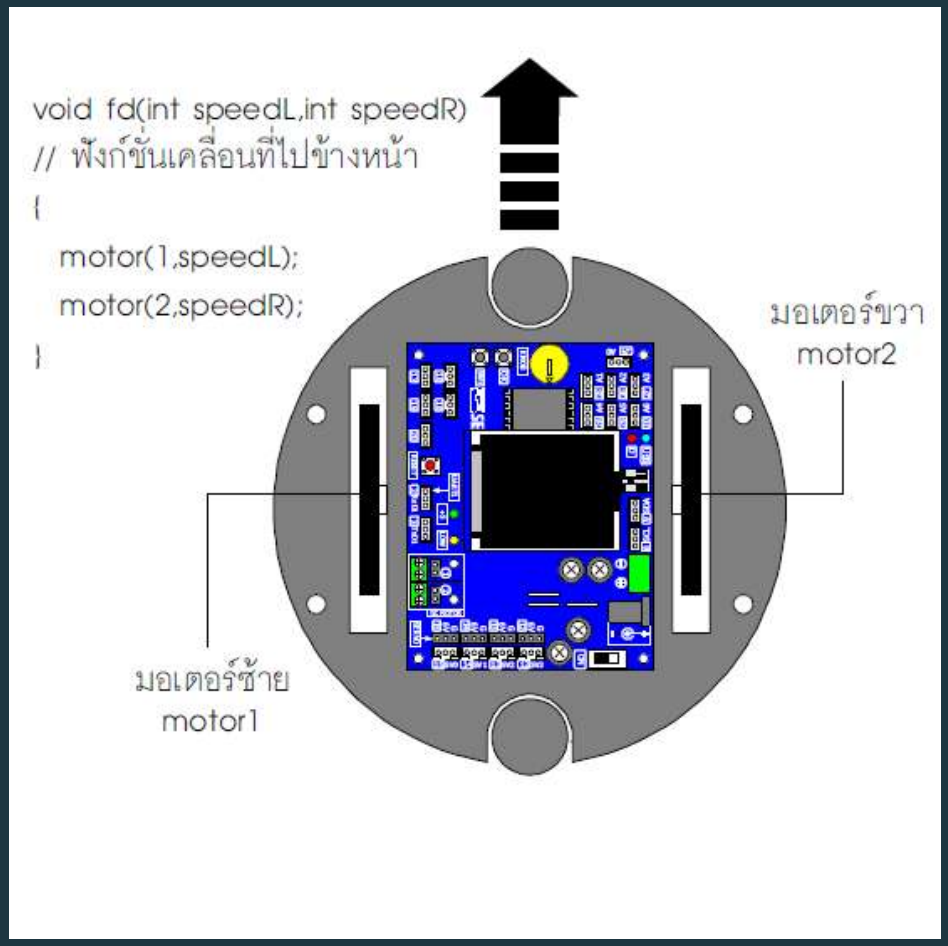

```
#include <ipst.h>
void setup()
{
   sw_OK_press();
}
void loop()
{
     motor(1,-40);//ซ้าย
     motor(2,-40);//ขวา
   sleep(1000);
}
```

หุ่นยนต์ iBOT เคลื่อนที่ตามเวลา




```
#include <ipst.h>
void setup()
{
glcd(1,1,"Press OK");
sw_OK_press();
motor(1,40); //ซ้าย
motor(2,40);// ขวา
sleep(4000);//ทำงานคำสั่งข้างบน 4 วินาที
ao();
}
void loop()
{}
```

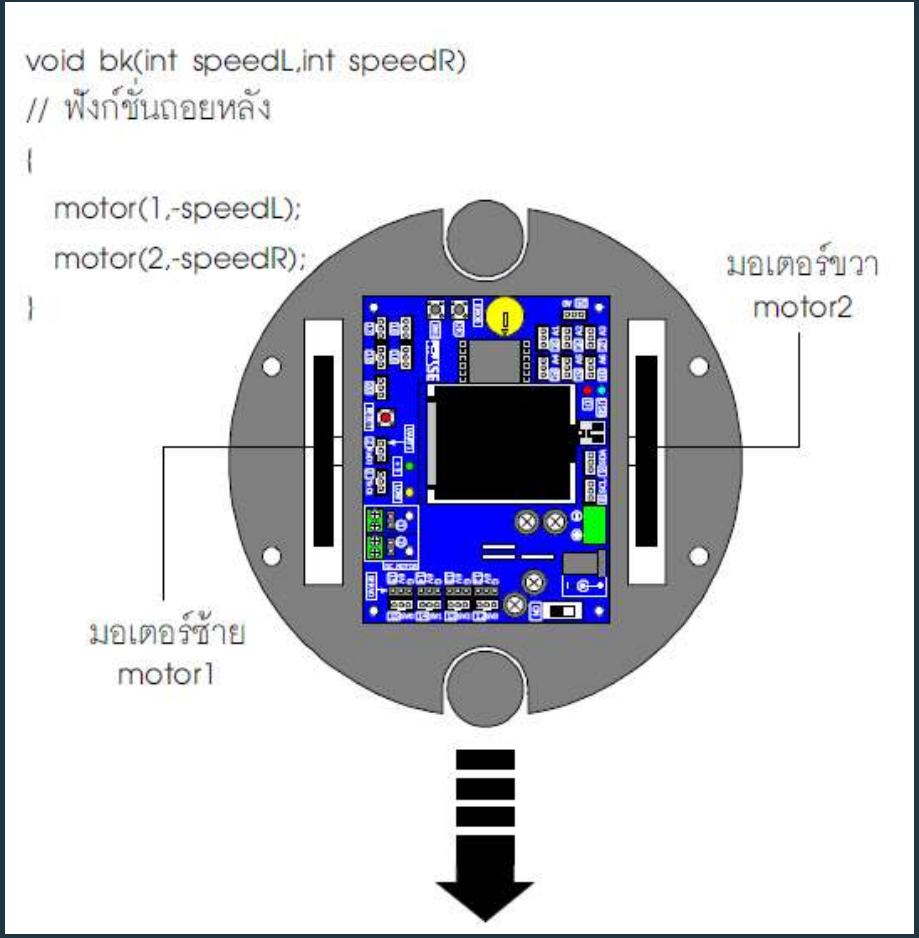
ฟังก์ชั่น
การเคลื่อนที่หุ่นยนต์ iBOT



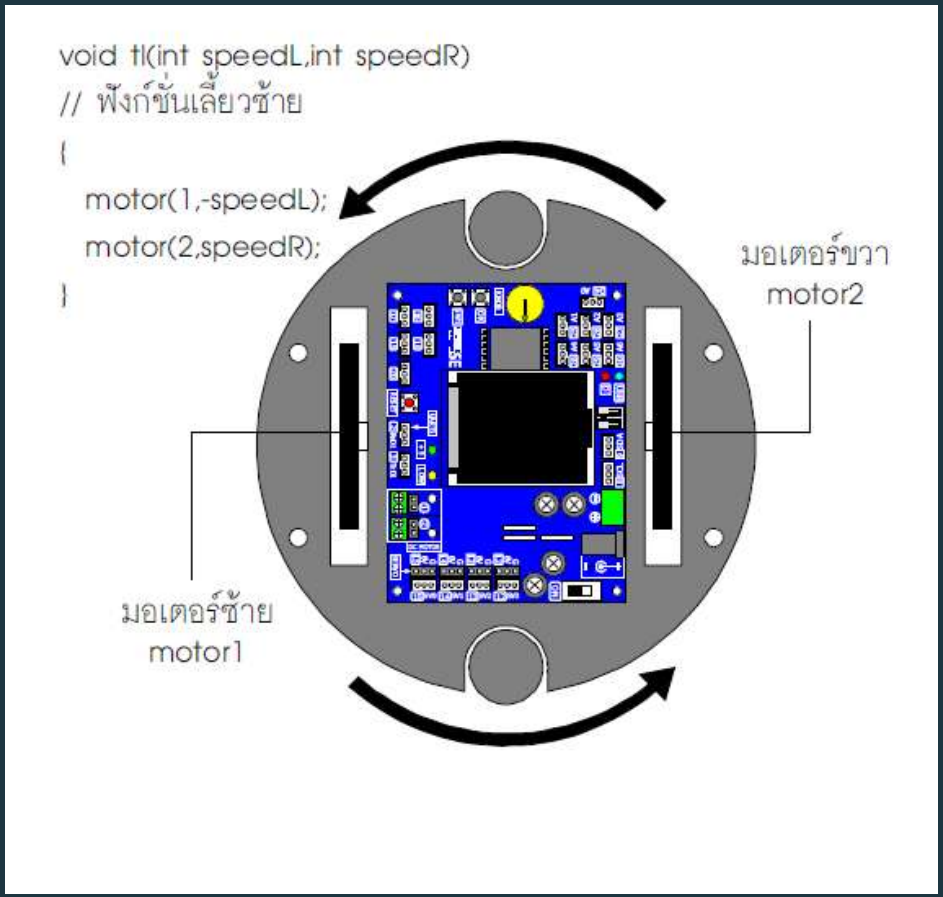

```
void fd(int speedL,int
speedR)
{
 motor(1,speedL);//ซ้าย
 motor(2,speedR);//ขวา
}
```

ฟังก์ชั่น
การเคลื่อนที่หุ่นยนต์ iBOT

```
void bk(int speedL,int
speedR)
{
 motor(1,-speedL);//ซ้าย
 motor(2,-speedR);//ขวา
}
```



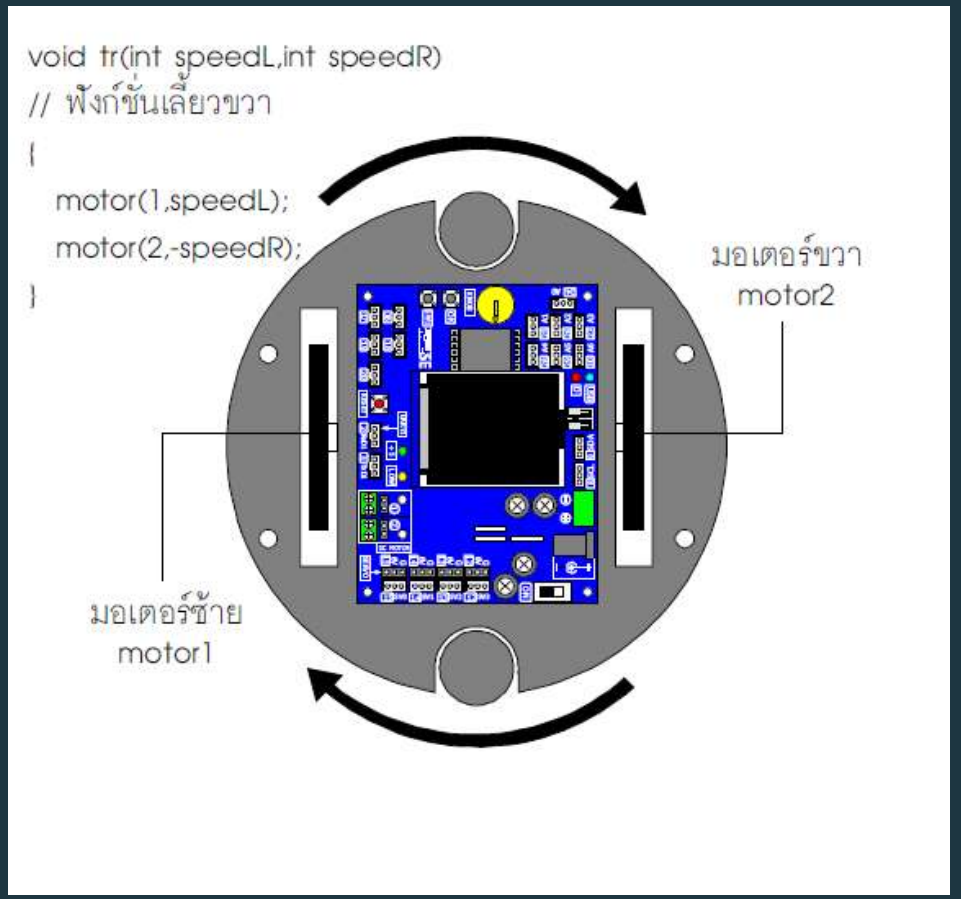
ฟังก์ชั่น
การเคลื่อนที่หุ่นยนต์ iBOT




```
void tl(int speedL,int
speedR)
{
 motor(1,-speedL);//ซ้าย
 motor(2,speedR);//ขวา
}
```

ฟังก์ชั่น
การเคลื่อนที่หุ่นยนต์ iBOT

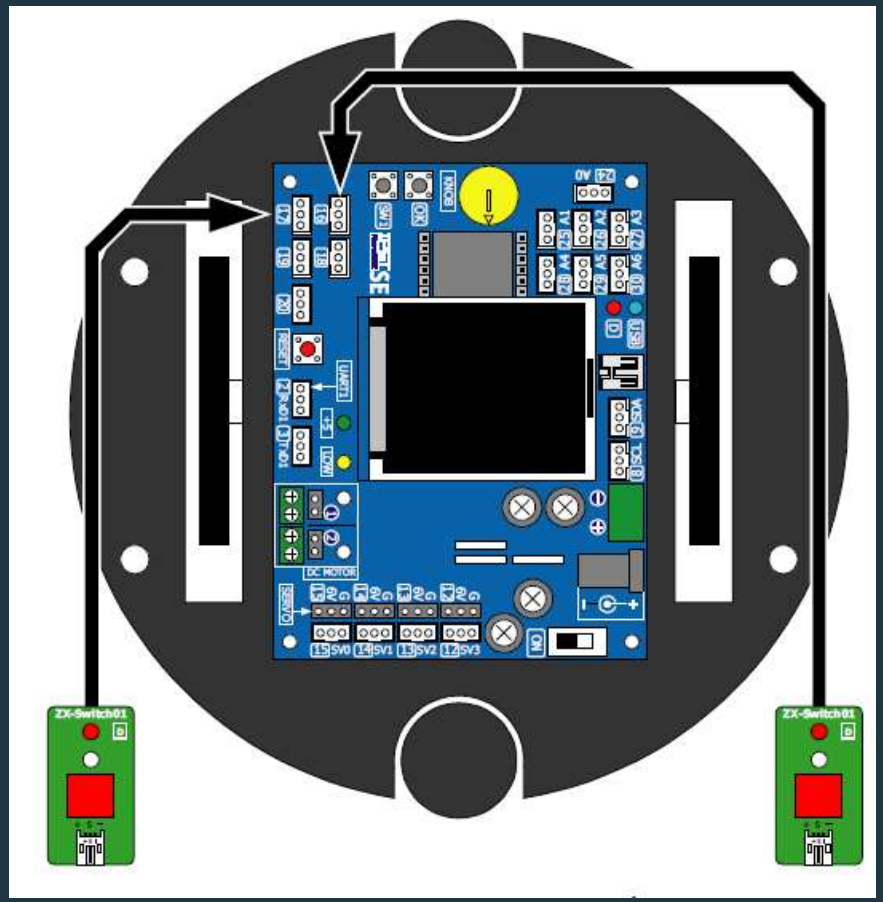
```
void tr(int speedL,int
speedR)
{
 motor(1,speedL);//ซ้าย
 motor(2,-speedR);//ขวา
}
```



การเรียกใช้ฟังก์ชั่น
การเคลื่อนที่หุ่นยนต์ iBOT

```
void setup() {
    sw_OK_press();
}
void loop() {
    fd(40,40);//เรียกใช้ฟังก์ชั่นเดินตรง
    sleep(1000);//ทำงานคำสั่งข้างบนจนครบ 1 วินาที
    tl(40,40);//เรียกใช้ฟังก์ชั่นเลี้ยวซ้าย
    sleep(250);//ทำงานคำสั่งก่อนหน้า  0.25 วินาที
    ao();//สั่งให้มอเตอร์ทุกตัวหยุดการทำงาน
}
```

สวิตช์ควบคุมหุ่นยนต์ iBOT

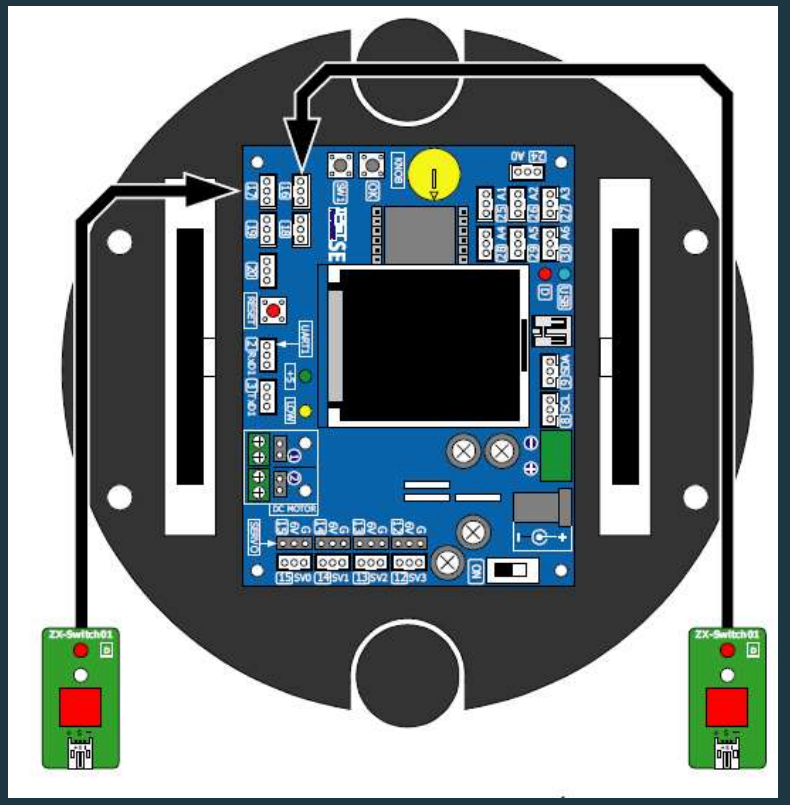


ต่อ สวิตช์ เข้ากับ Port Digital

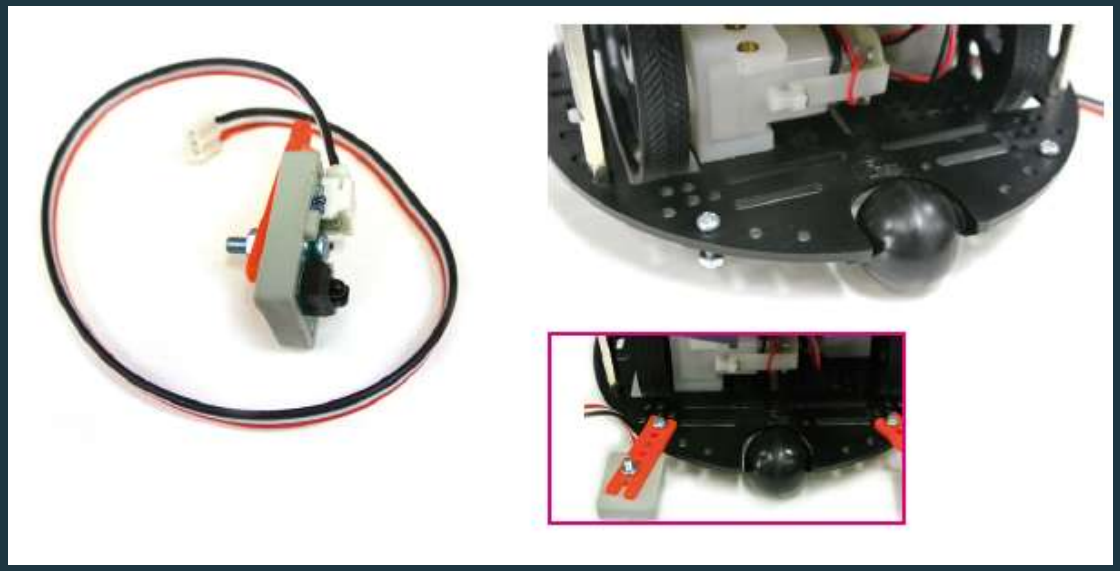
```
PORT 17,in(17) =>>ซ้าย

PORT 16,in(16) = ขวา
```

สร้างรถบังคับด้วยหุ่นยนต์ iBOT

```
void loop()
{
    if (in(17)==0&&in(16)==0)
        {fd(40,40);}
    else if(in(17)==0)
          {tl(40,40);}
    else if(in(16)==0)
          {tr(40,40);}
    else
          {ao(); }
}
```



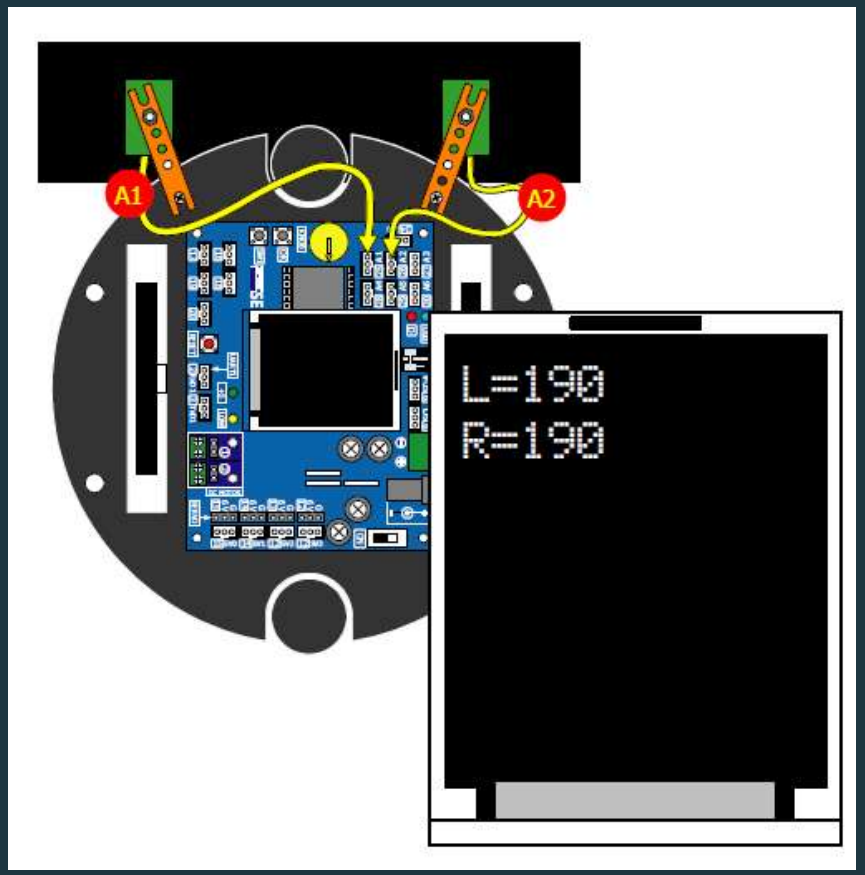
การอ่านค่าตัวตรวจจับของหุ่นยนต์ iBOT



ต่อ สวิตช์ เข้ากับ Port Analog

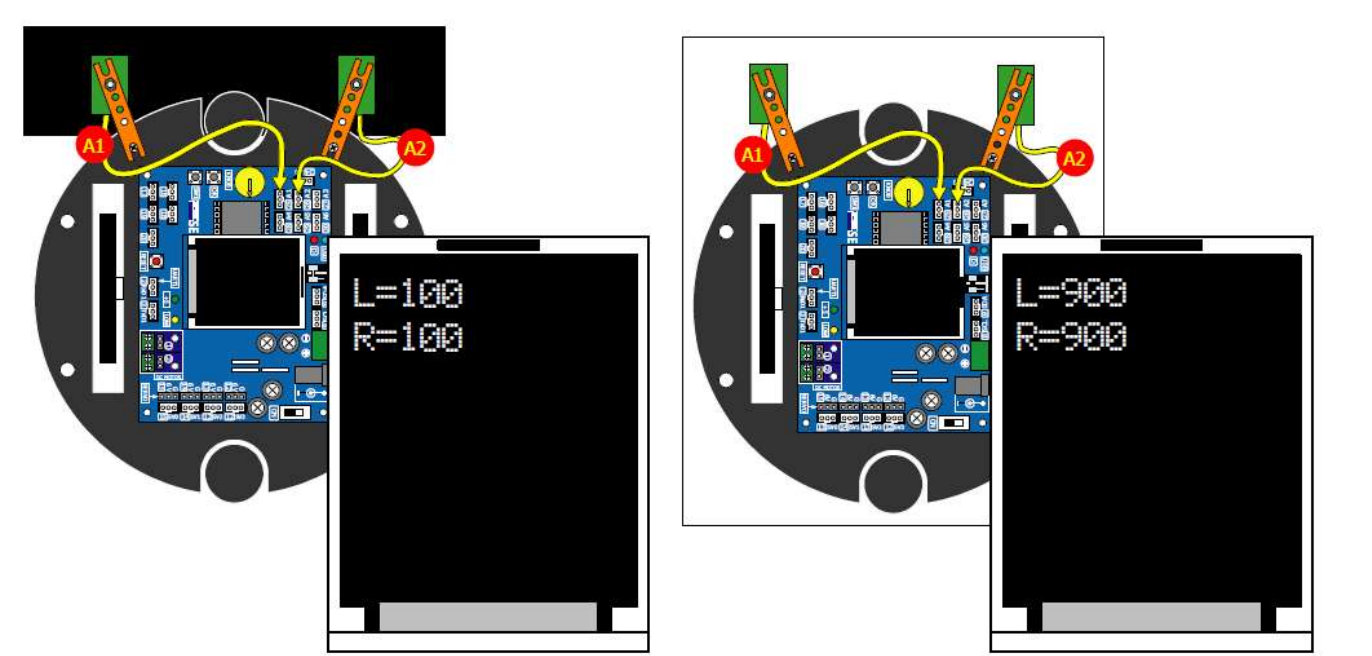
PORT A1,analog(1) =>>ซ้าย ,PORT A2,analog(2) = ขวา

การอ่านค่าตัวตรวจจับของหุ่นยนต์ iBOT




```
#include <ipst.h>
void setup()
{
      setTextSize(2);
}
void loop()
{
   glcd(0,0,"L=%d
   ",analog(1));
   glcd(1,0,"R=%d
   ",analog(2));
}
```

การอ่านค่าตัวตรวจจับของหุ่นยนต์ iBOT

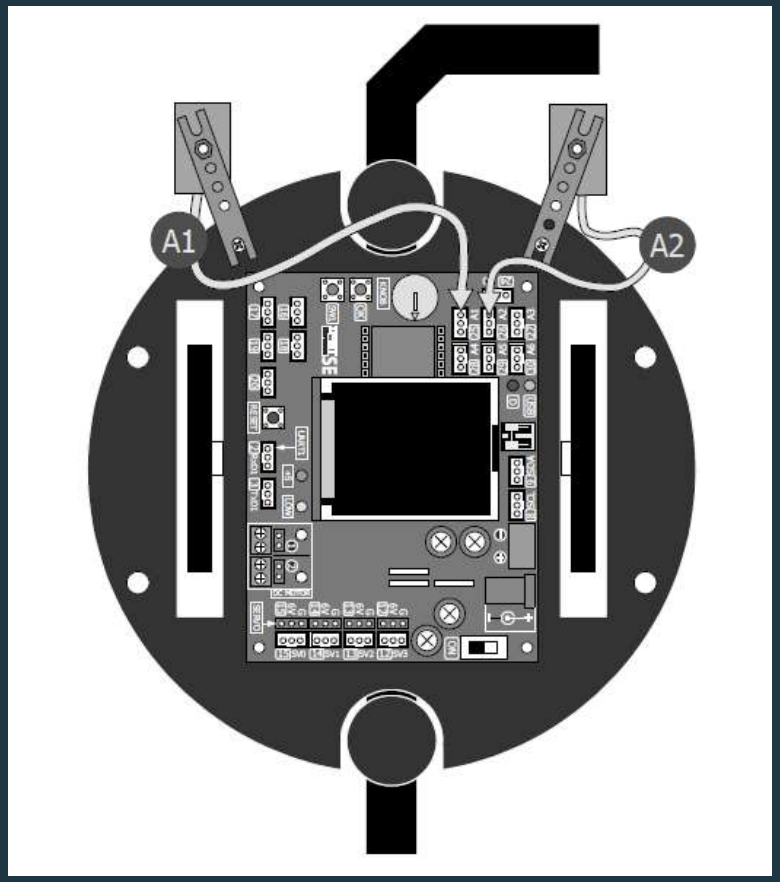


การหาค่ากลาง = (ค่าสีขาว +>ค่าสีดำ)/2

= (900+100)/2

= 500

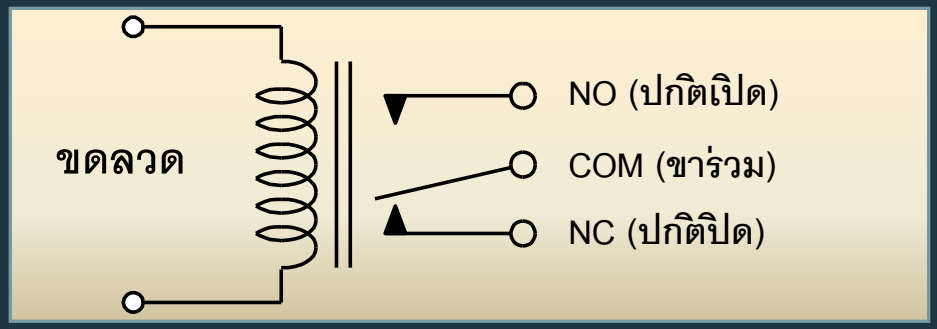
หุ่นยนต์ iBOT เคลื่อนที่ตามเส้น

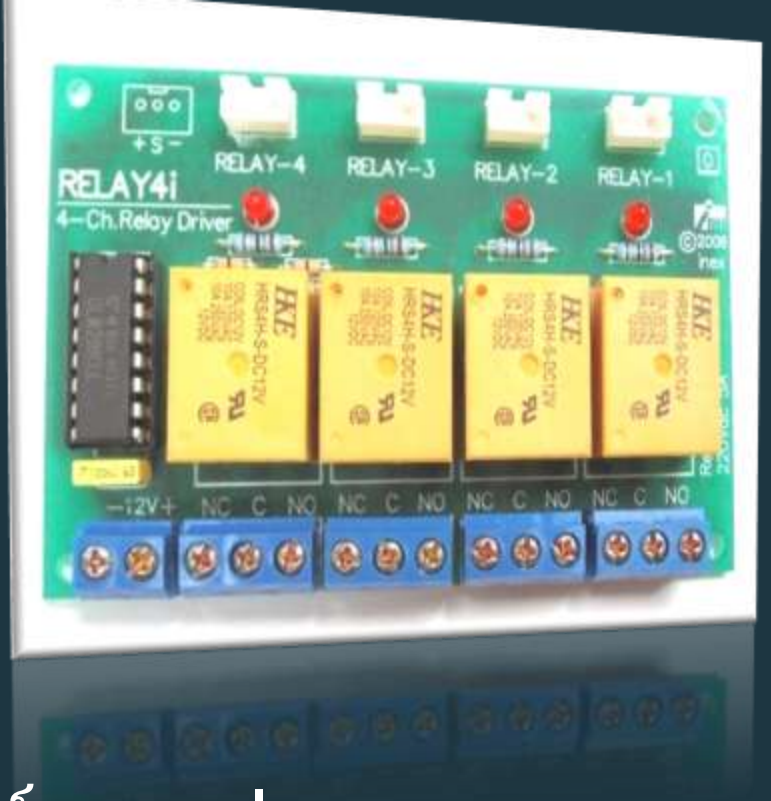



```
void loop()
{
    L=analog(1);R=analog(2);
    if(L>500&&R>500)
       {fd(60);}
    else if(L<500&&R>500)
       {tl(60);}
    else if(L>500&&R<500)
       {tr(60);}
}
```

```
#include <ipst.h>
void setup(){
    glcd(1,1,"Press OK");
    sw_OK_press();
    motor(1,40); motor(2,40);
    sleep(4000); ao();
}
void loop(){
}
```

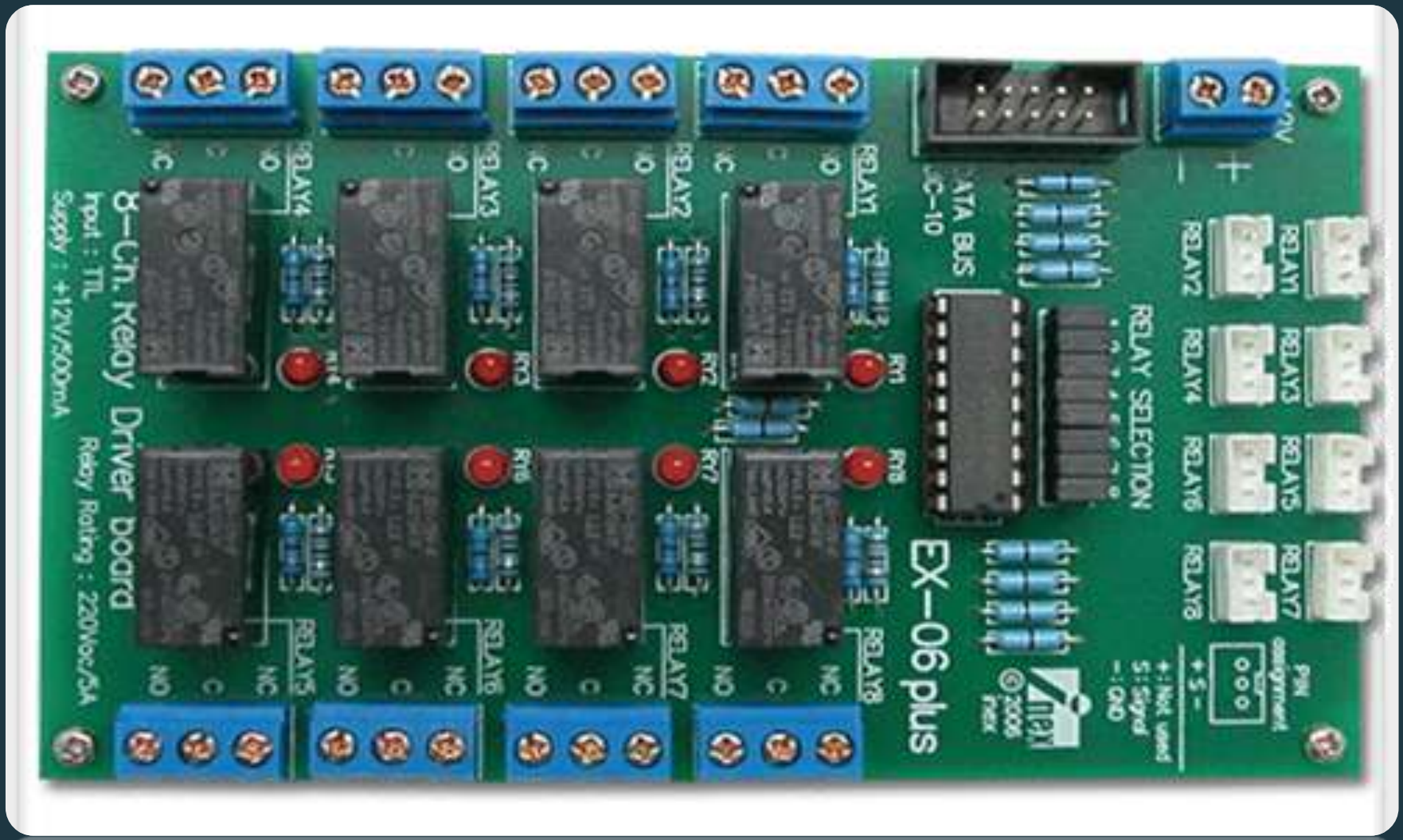
บอร์ดขับรีเลย์ 4 ช่อง



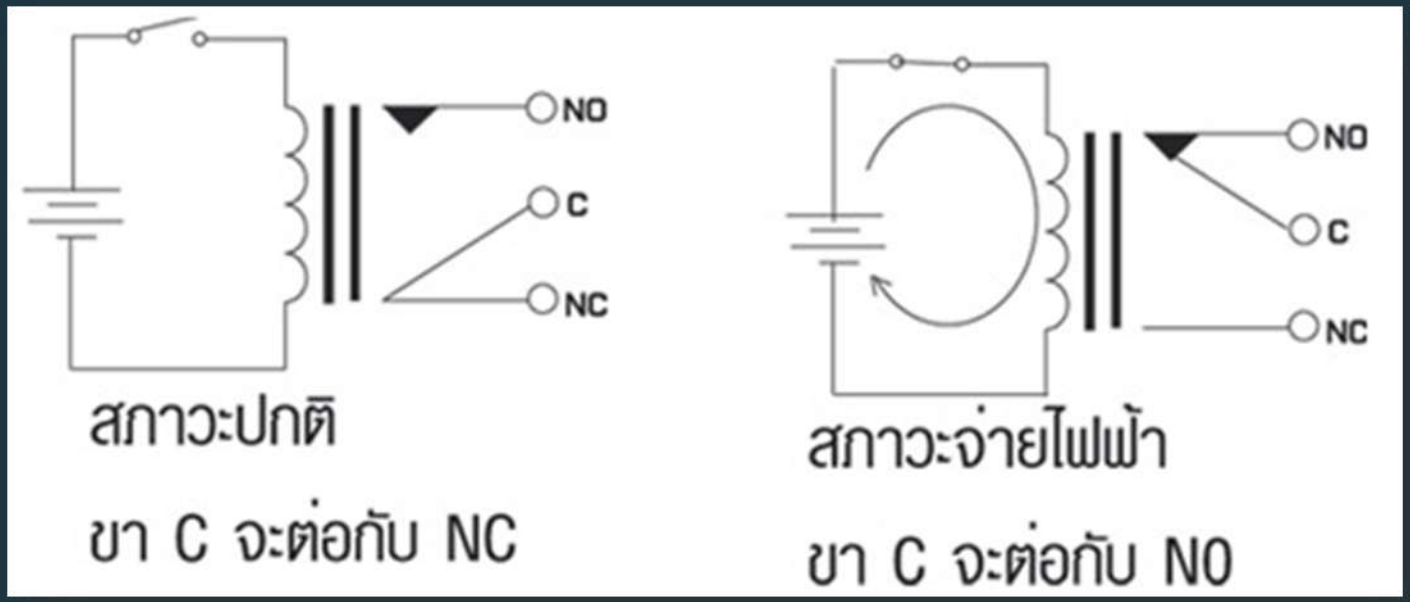


- ใช้ไอซีเบอร์ ULN2003 ขับรีเลย์ 12V 4 ช่อง
- ใช้ไฟเลี้ยง +12V ต่อแยก
- รับลอจิก “1” ให้รีเลย์ทำงาน
- มี LED แสดงการทำงาน
- ขับ 220VAC 5A ขับโหลดได้ 600W

บอร์ดขับรีเลย์ 8 ช่อง



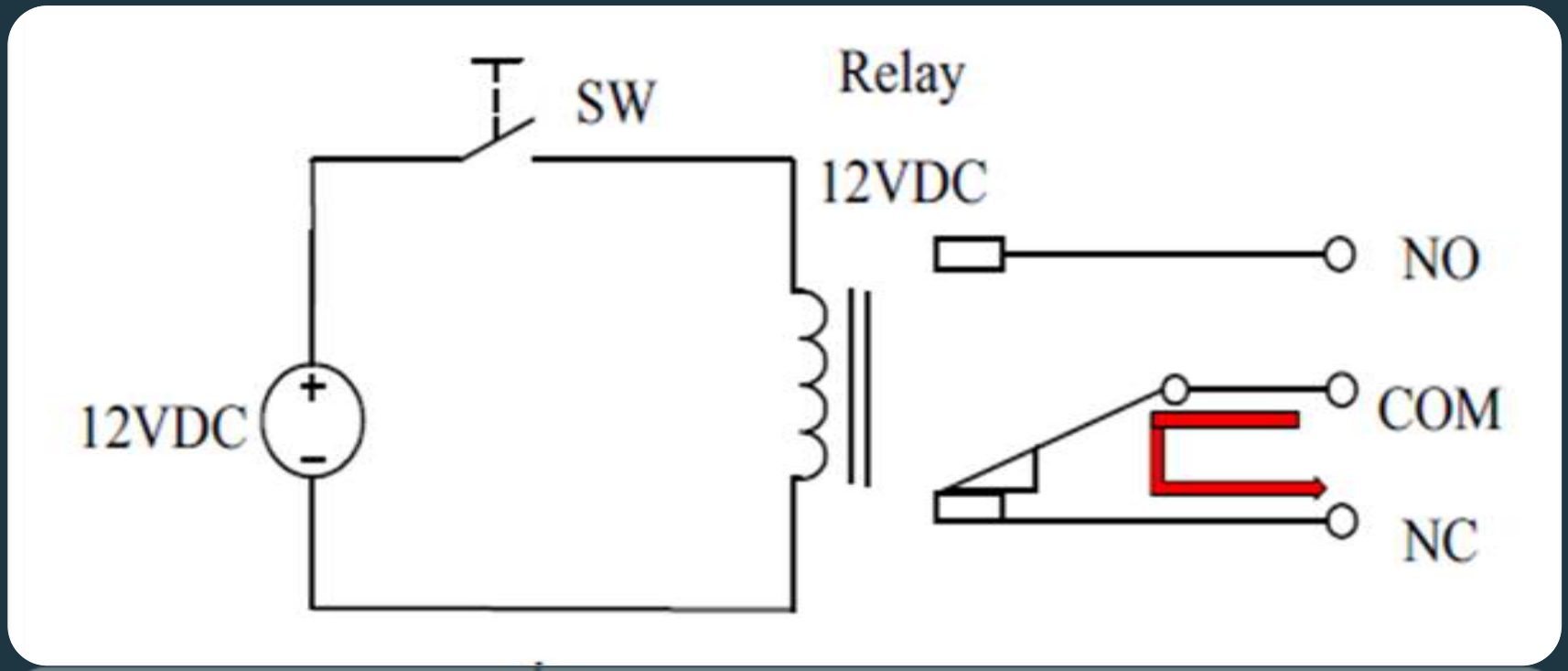
รีเลย์ (RELAY)



รีเลย์ (Relay) คือ สวิตซ์ที่ถูกสั่งให้ทำงาน โดยการป้อนสัญญาณเพื่อใช้ควบคุมการทำงาน โดยใช้หลักการเหนี่ยวนำของขดลวด เกิดเป็นสนามแม่เหล็กไฟฟ้าบริเวณหน้าสัมผัส การที่จะทำให้รีเลย์ทำงานต้องจ่ายไฟให้ตามที่กำหนด ก็จะทำให้หน้าสัมผัสติดกันดังภาพ

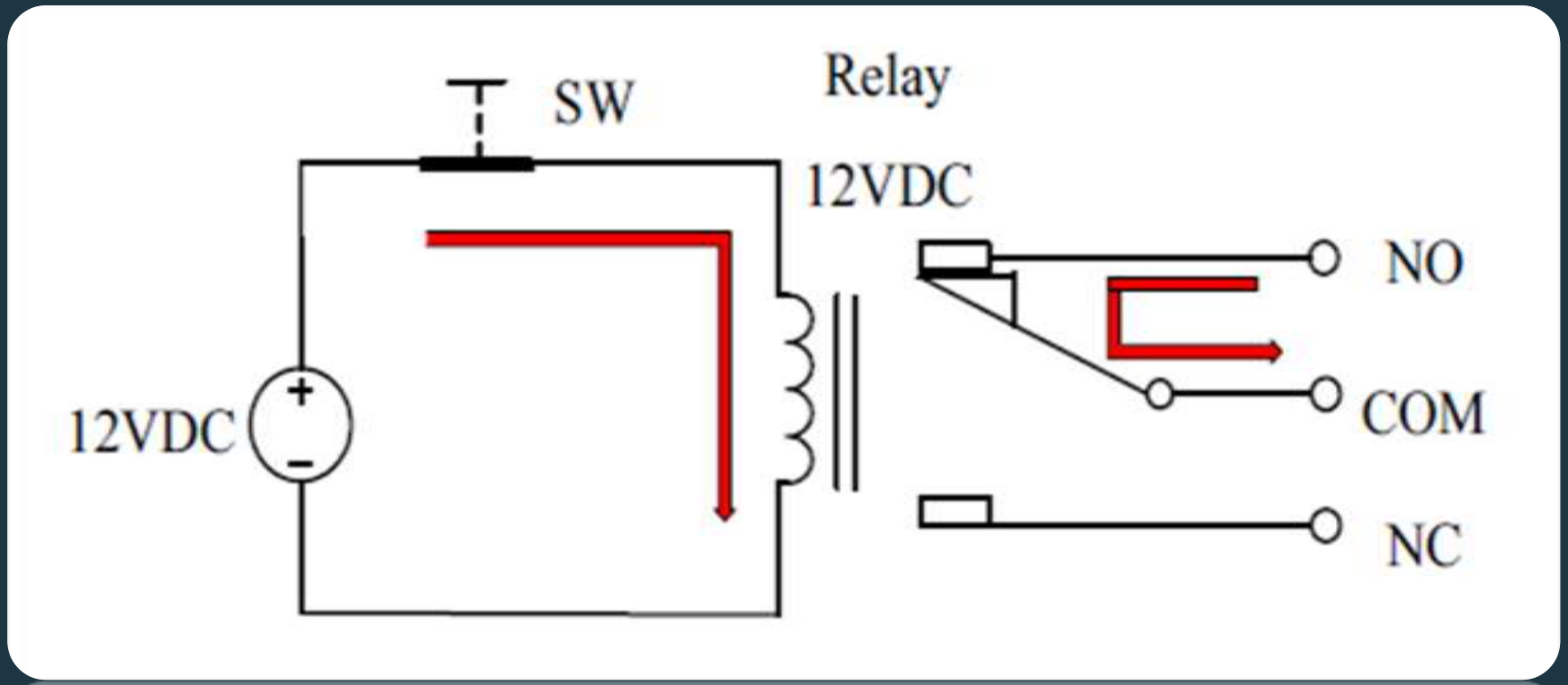
หลักการทำงานของรีเลย์

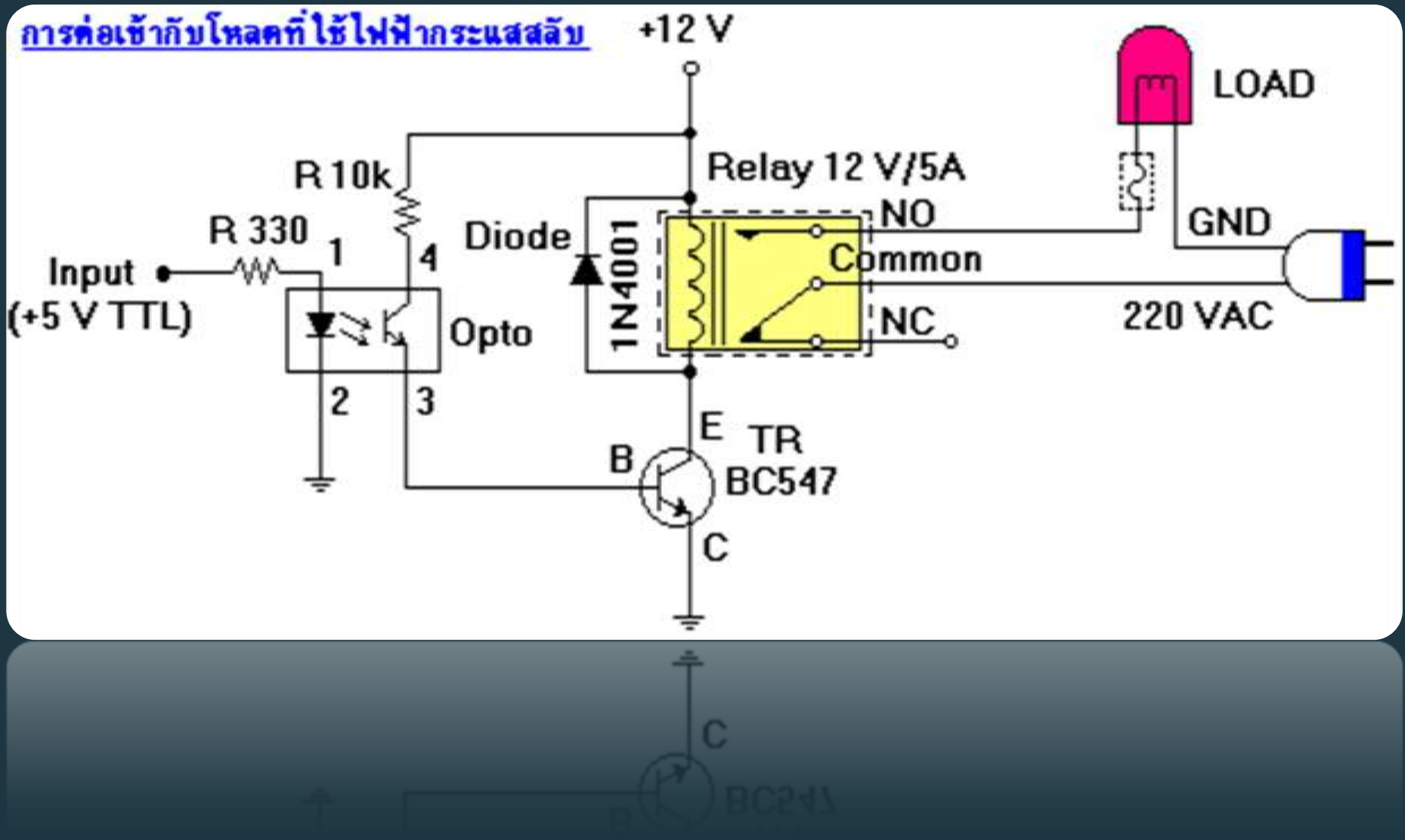
เมื่อไม่กดสวิตซ์ รีเลย์จะไม่ทำงานขา COM เชื่อมต่อกับขา NC

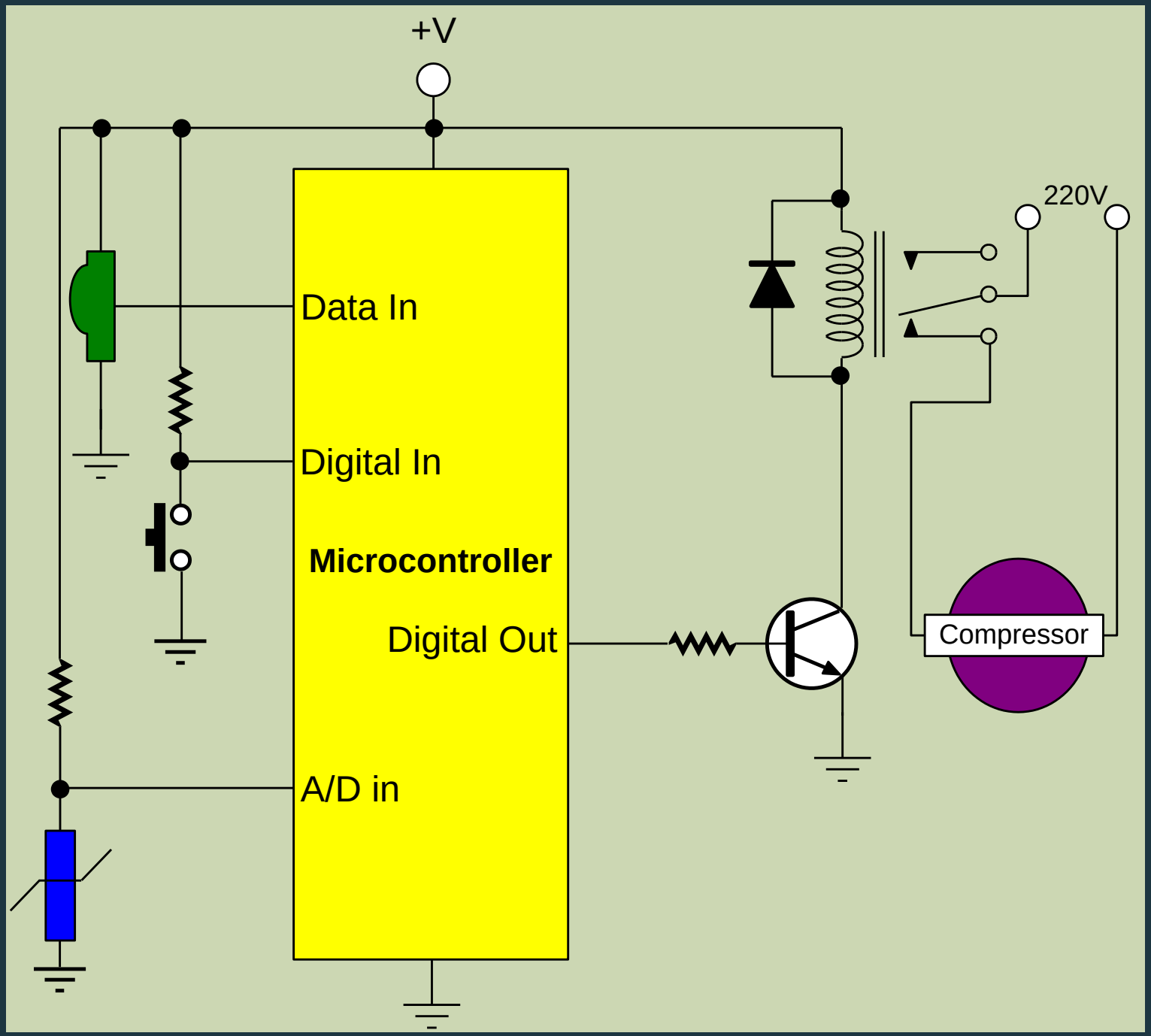


หลักการทำงานของรีเลย์

เมื่อกดสวิตซ์ รีเลย์จะทำงาน ขา COM จะเชื่อมต่อกับขา NO



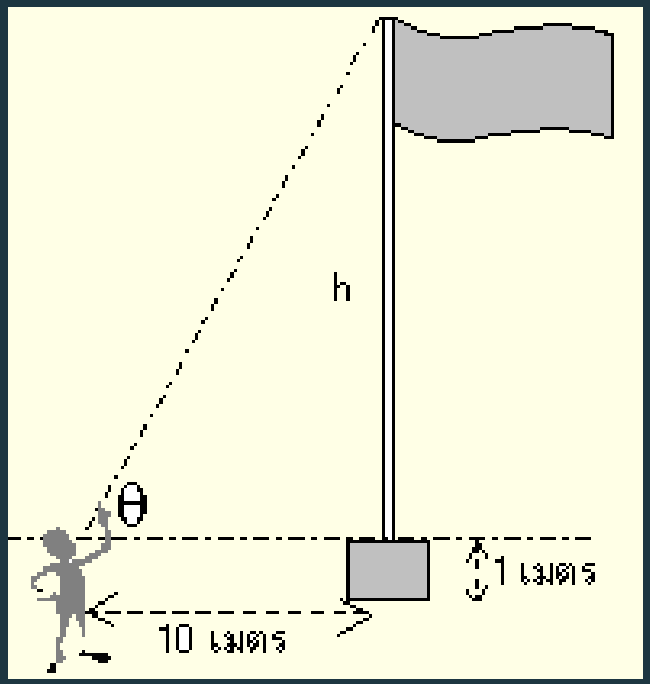

การต่อเข้ากับโหลดที่ใช้ไฟฟ้ากระแสสลับ
+12 V
LOAD
R 10k
Relay 12 V/5A
NO
GND
R 330
1
4
Diode
Common
Input
(+5 V TTL)
1N4001
Opto
NC
220 VAC
2
3
E
TR
B
BC547
C

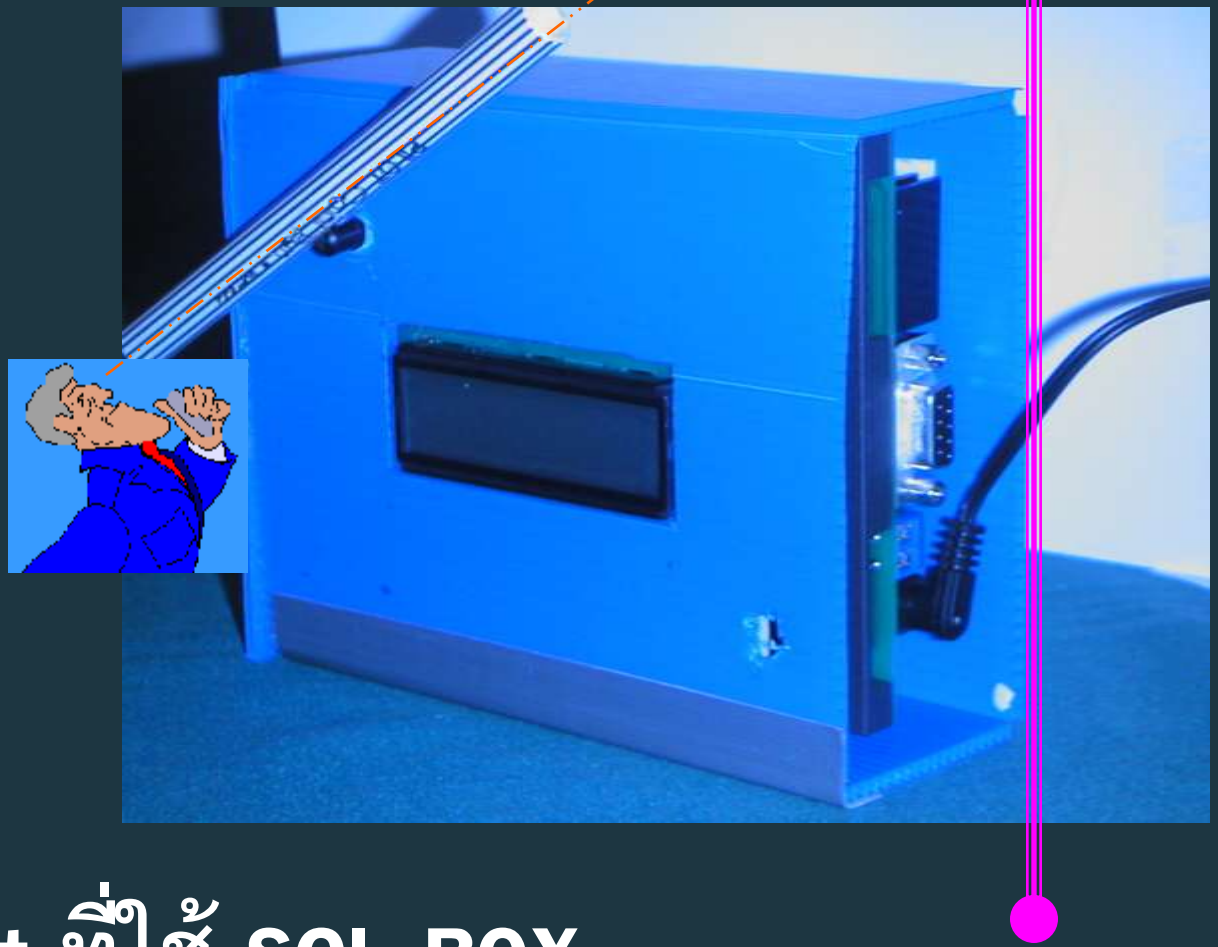

+V
Data In
Digital In
Microcontroller
Digital Out
A/D in
220V
Compressor

Solid State Relay

ตัวอย่างโครงงาน

กล่องหาความสูงอัตโนมัติ



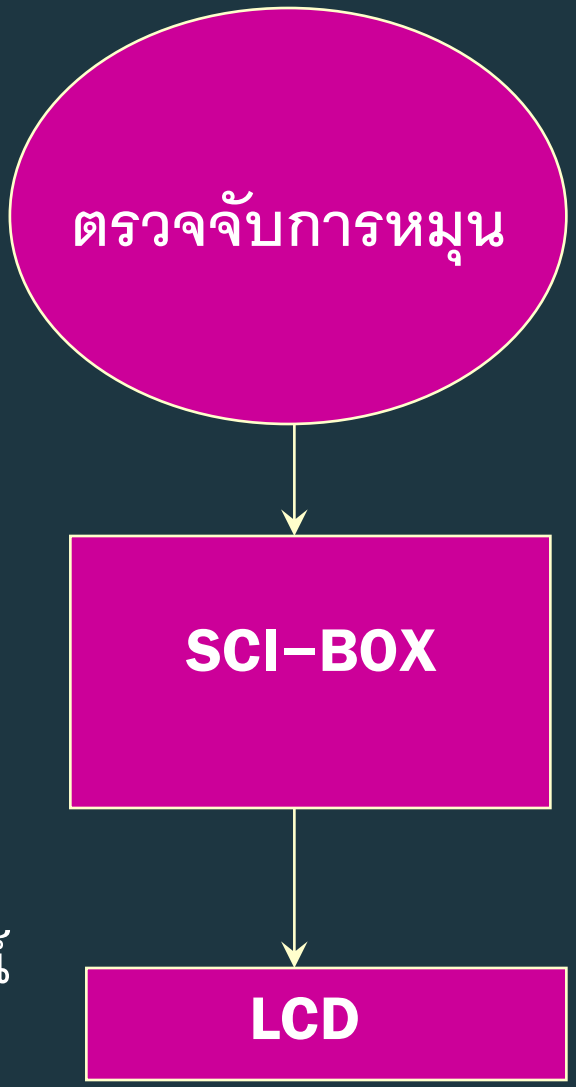


ตัวอย่าง Project ที่ใช้ SCI-BOX

กล่องหาความสูงอัตโนมัติ

- วัตถุประสงค์ เพื่อสร้างกล่องหาความสูงอัตโนมัติ
- แนวคิด ในชีวิตประจำวันทั่วไปหลายครั้งที่ต้องการทราบความสูงของวัตถุแต่ไม่สามารถใช้อุปกรณ์หรือเครื่องวัดโดยตรงได้ เช่น ความสูงของเสาธง ความสูงของต้นไม้ ความสูงของหน้าผา เป็นต้น แต่เราสามารถคำนวณหาความสูงสิ่งเหล่านี้ได้โดยใช้กระบวนการทางคณิตศาสตร์ ในเรื่องของสามเหลี่ยมคล้าย ใช้ทฤษฎีพิธากอรัส หรือใช้ความรู้ทางตรีโกณมิติ แต่จะต้องเสียเวลาในการคำนวณหรือคำนวณไม่ถูกต้อง แนวทางหนึ่งที่สามารถหาความสูงของวัตถุดังกล่าวได้ง่ายและ รวดเร็วก็คือ การสร้างกล่องหาความสูงอัตโนมัติ

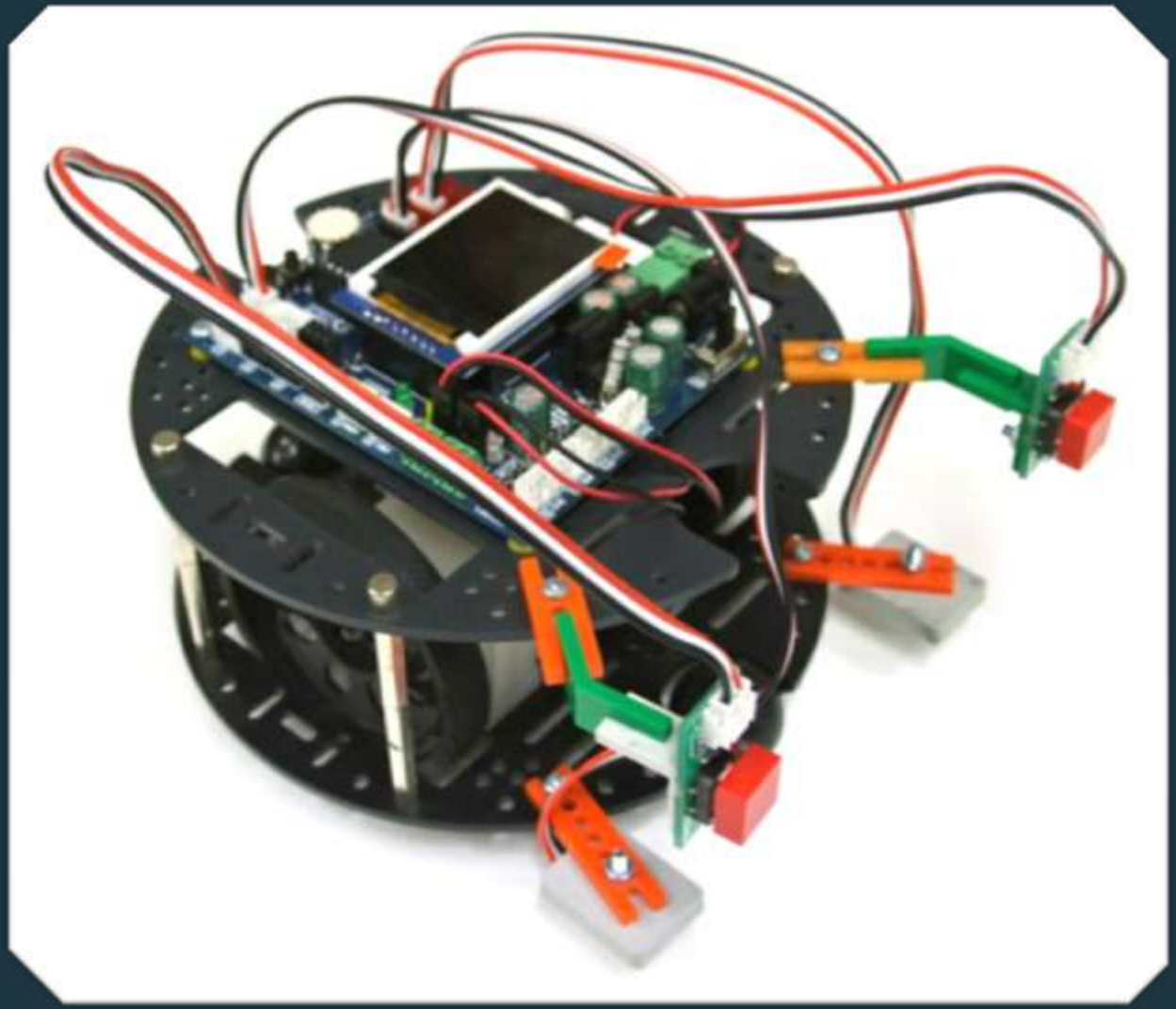
วงล้ออัศวิน



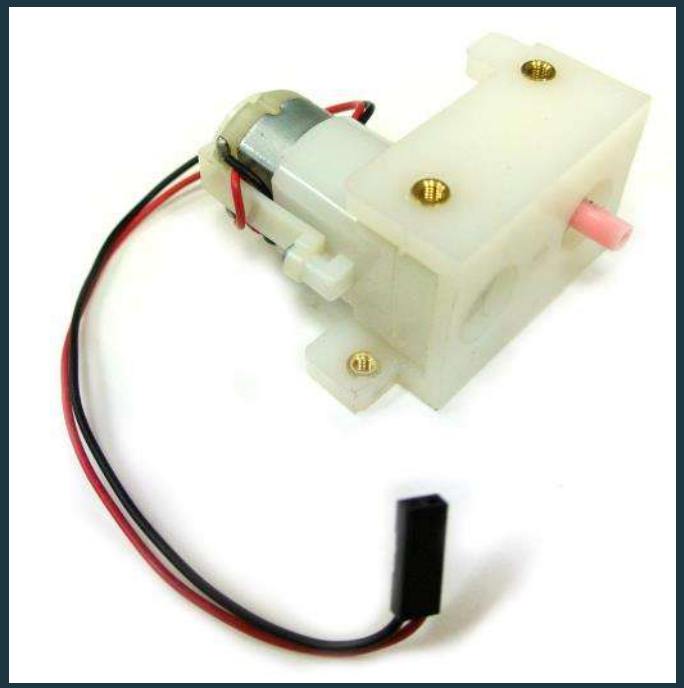
หลักการ

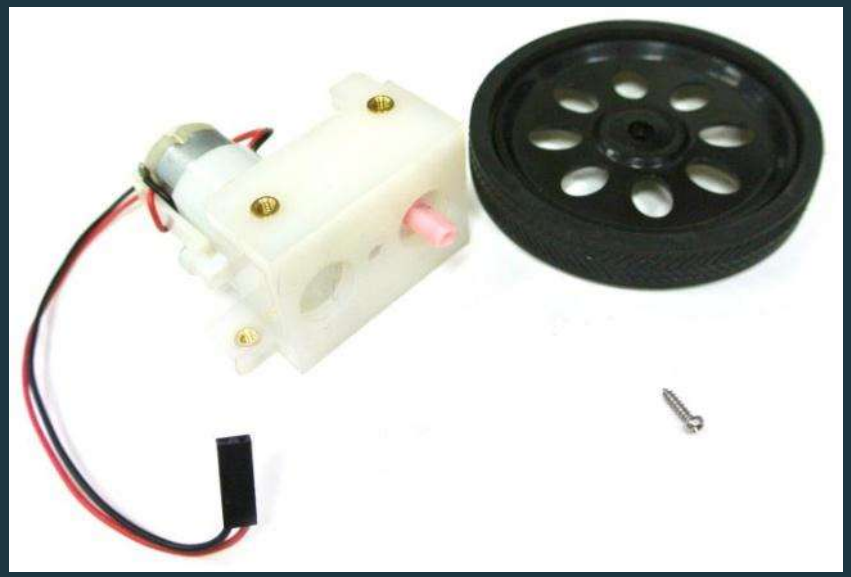
ใช้วงล้อหมุนเพื่อวัดระยะทางด้วยการนับจำนวนรอบ นำมาคูณกับความยาวของเส้นรอบวง โดยการใช้อุปกรณ์ตรวจจับนับจำนวนรอบและคำนวณระยะทางด้วยกล่องสมองกลและแสดงผลผ่านทางจอ LCD

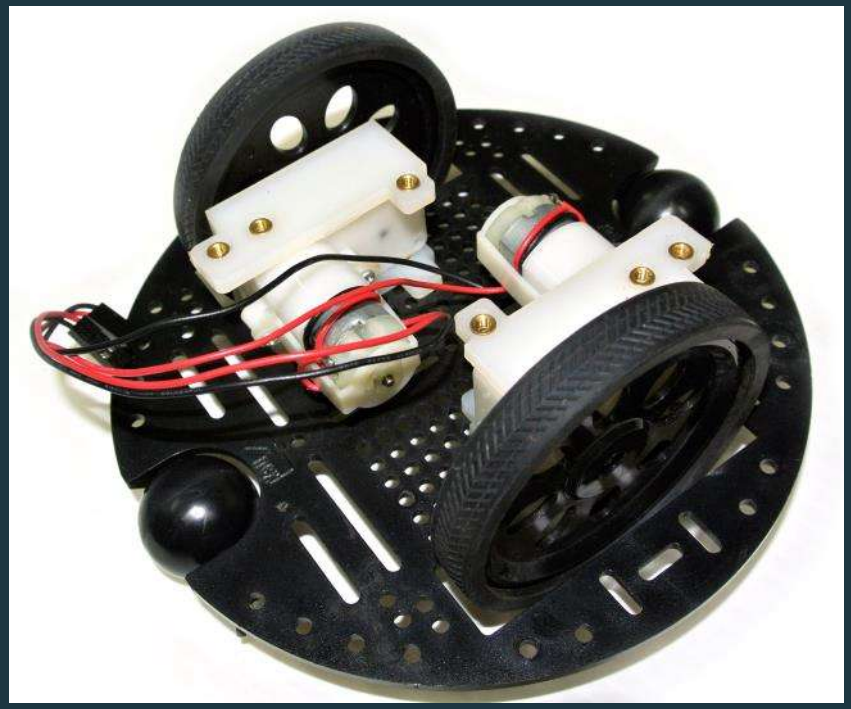
IPST MicroBOX SE → ROBOT

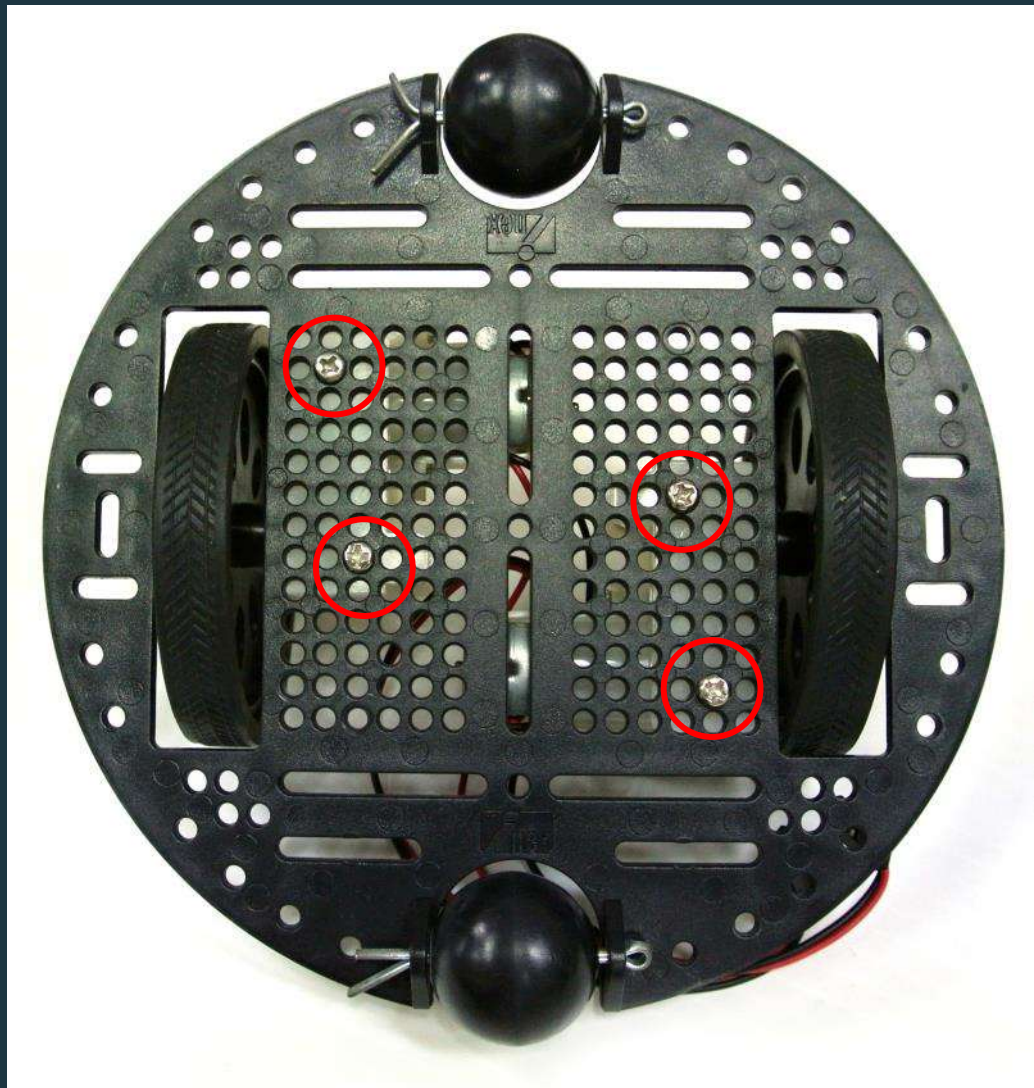


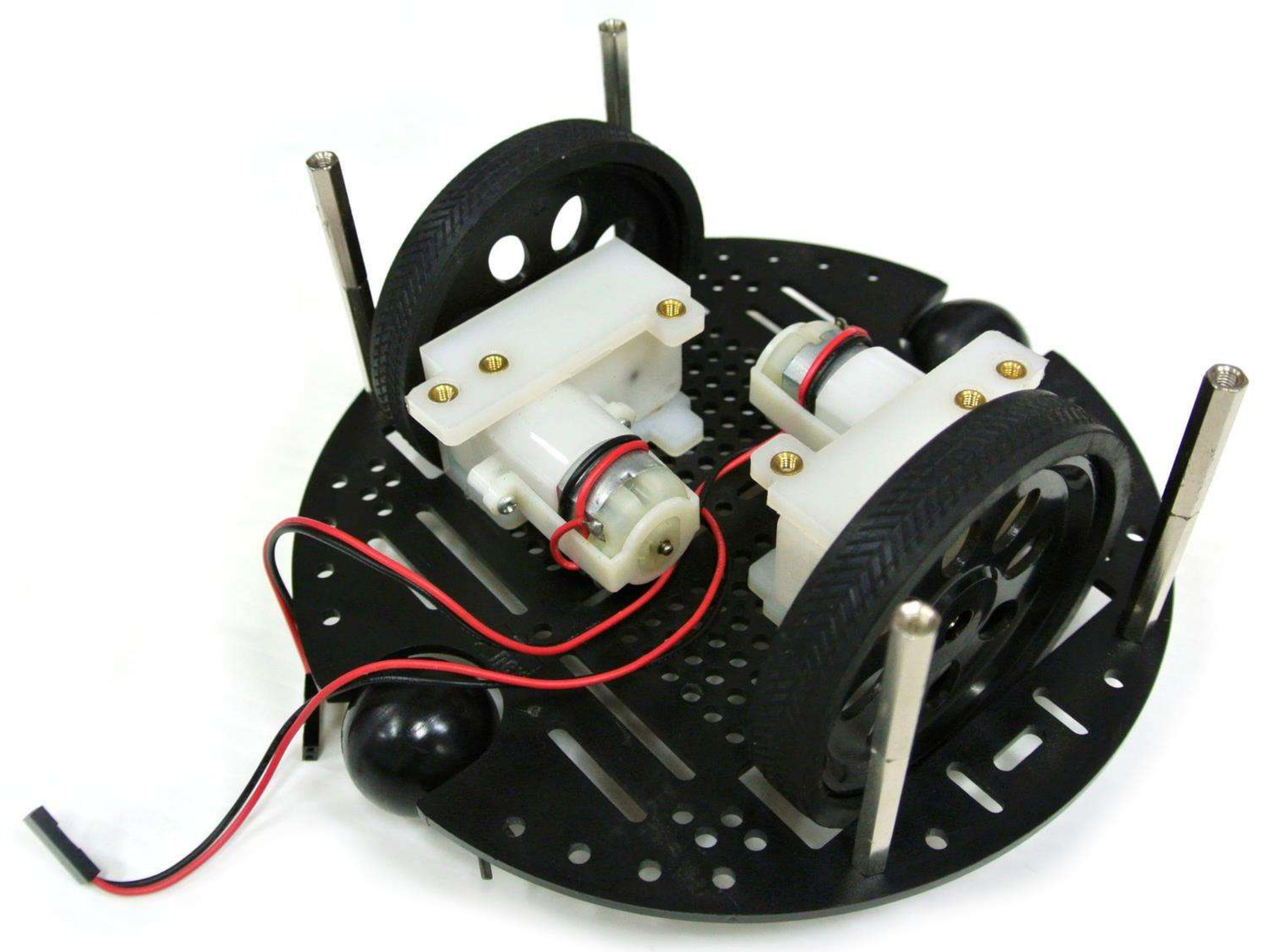
การสร้างหุ่นยนต์

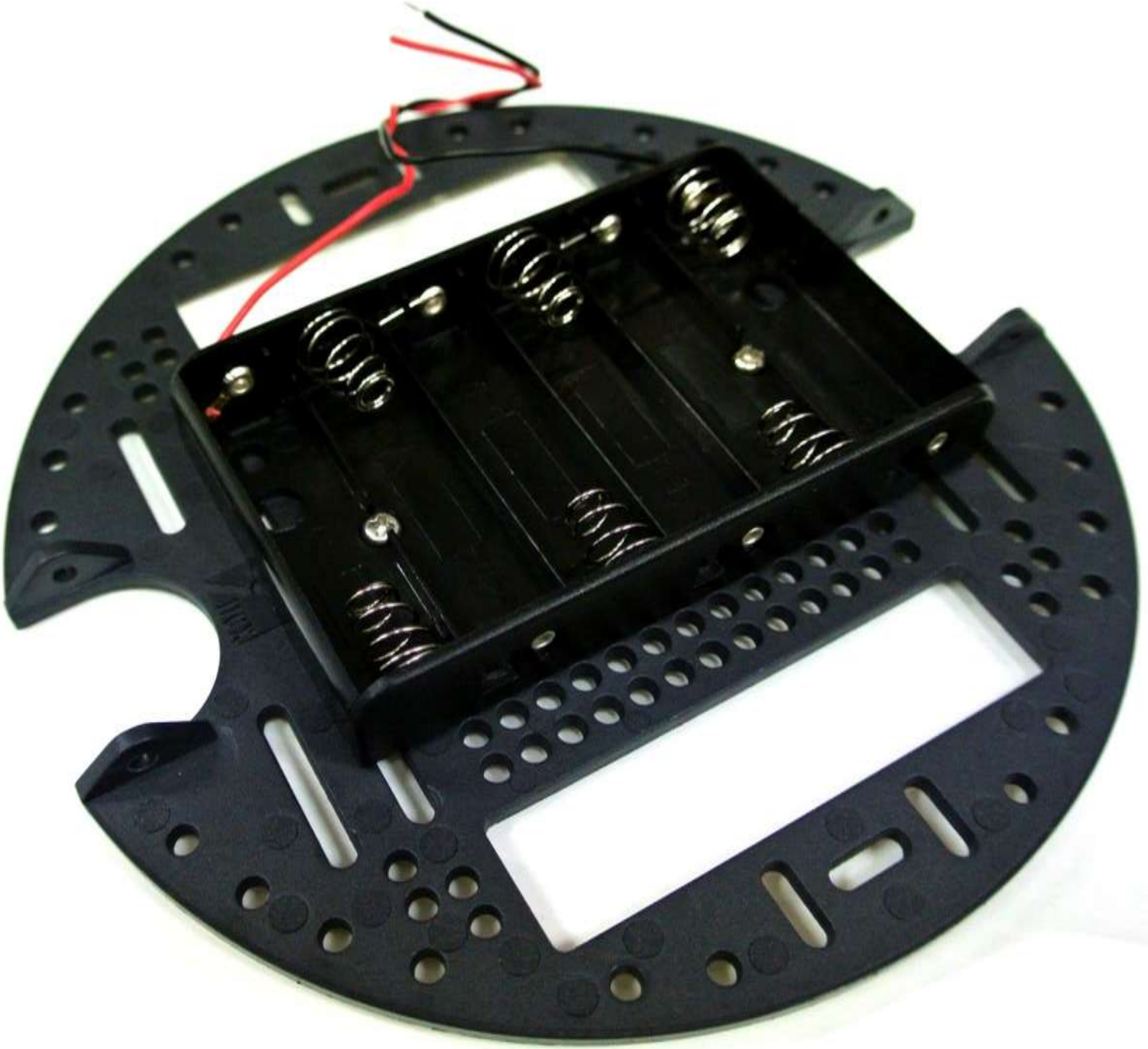


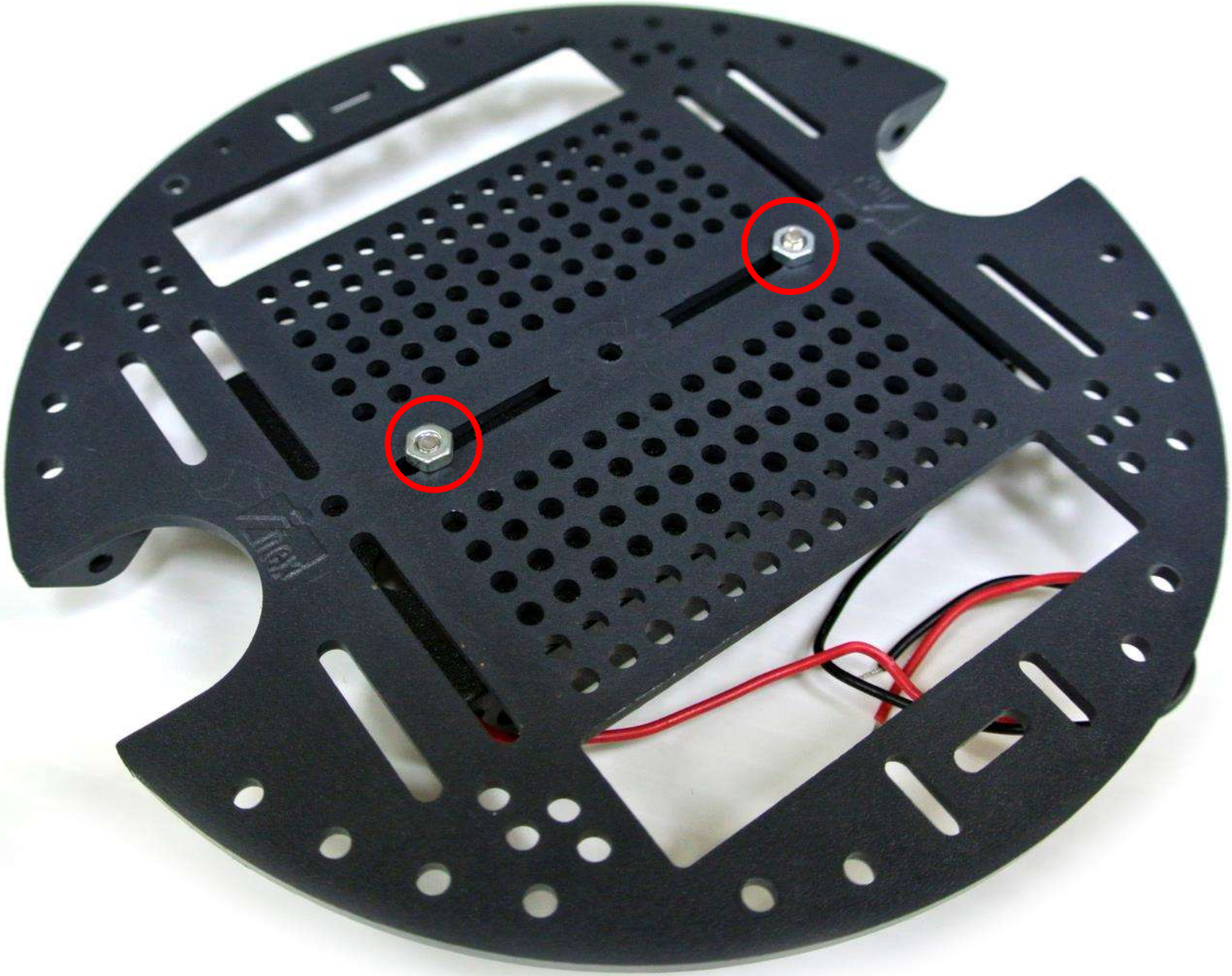


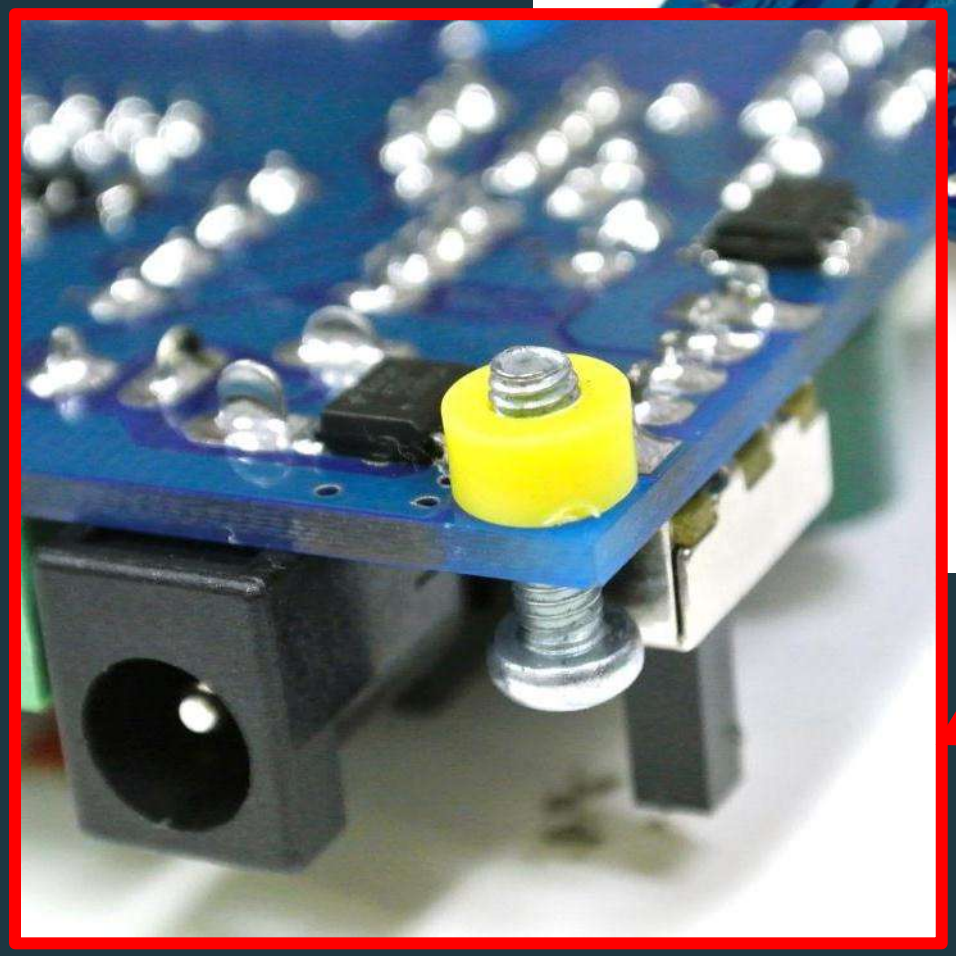


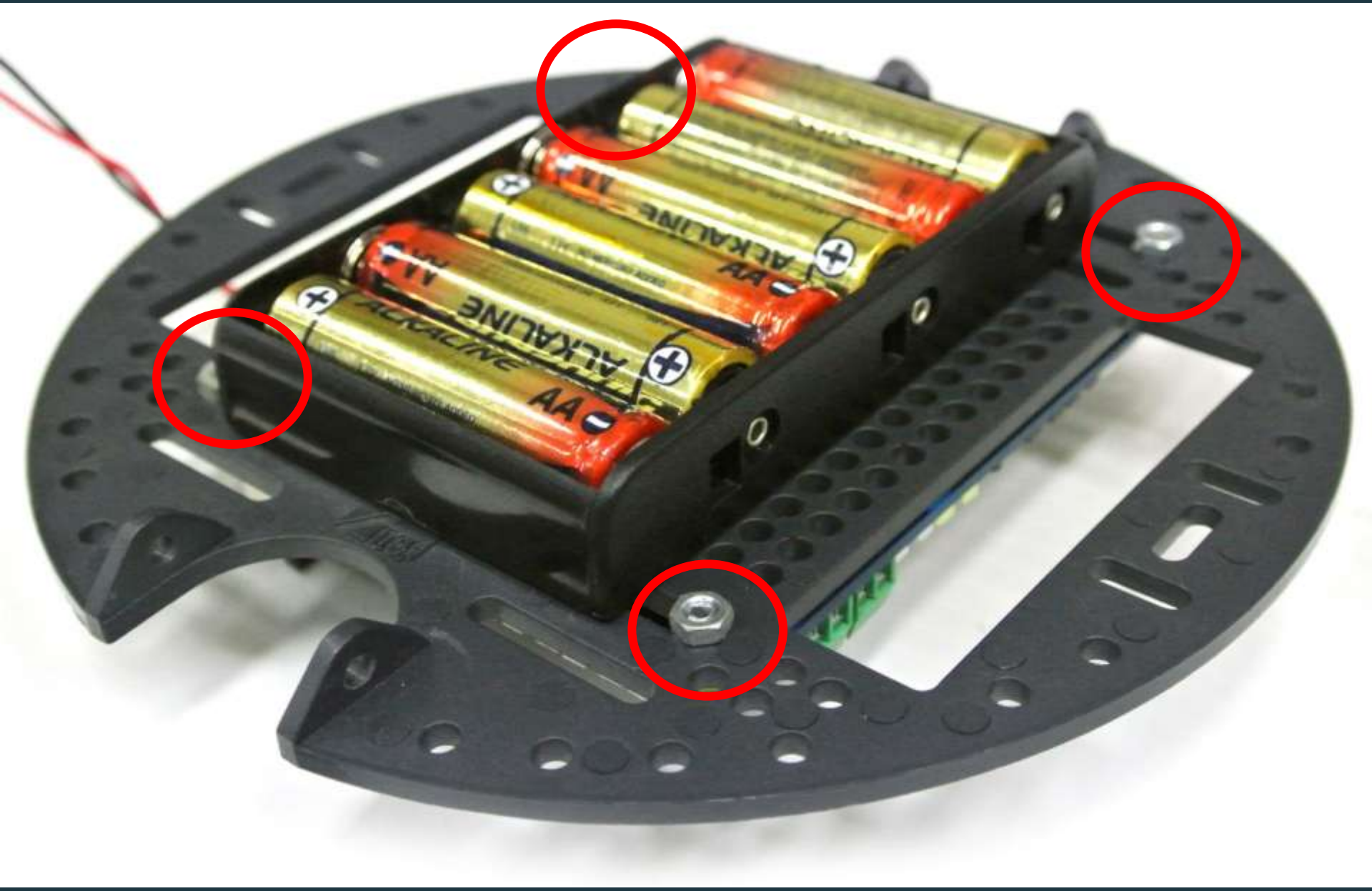


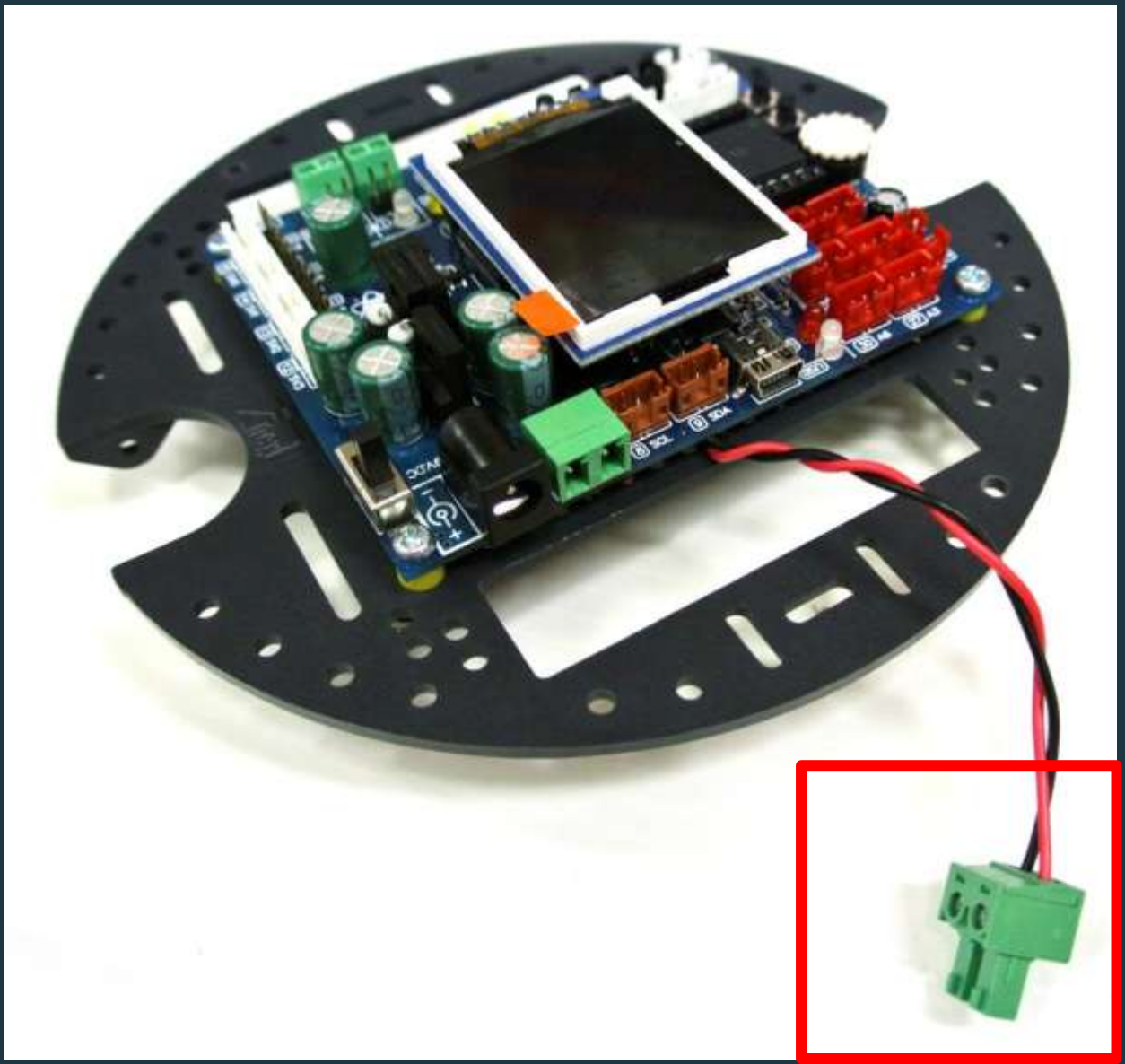


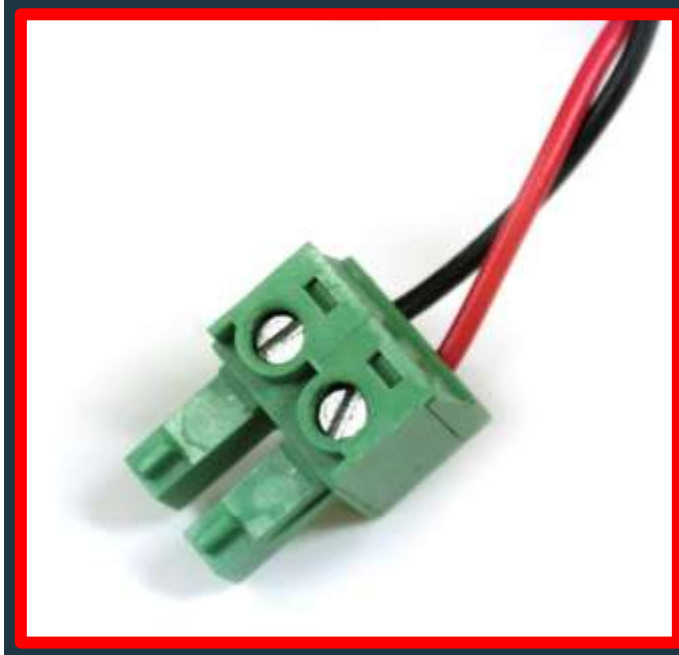


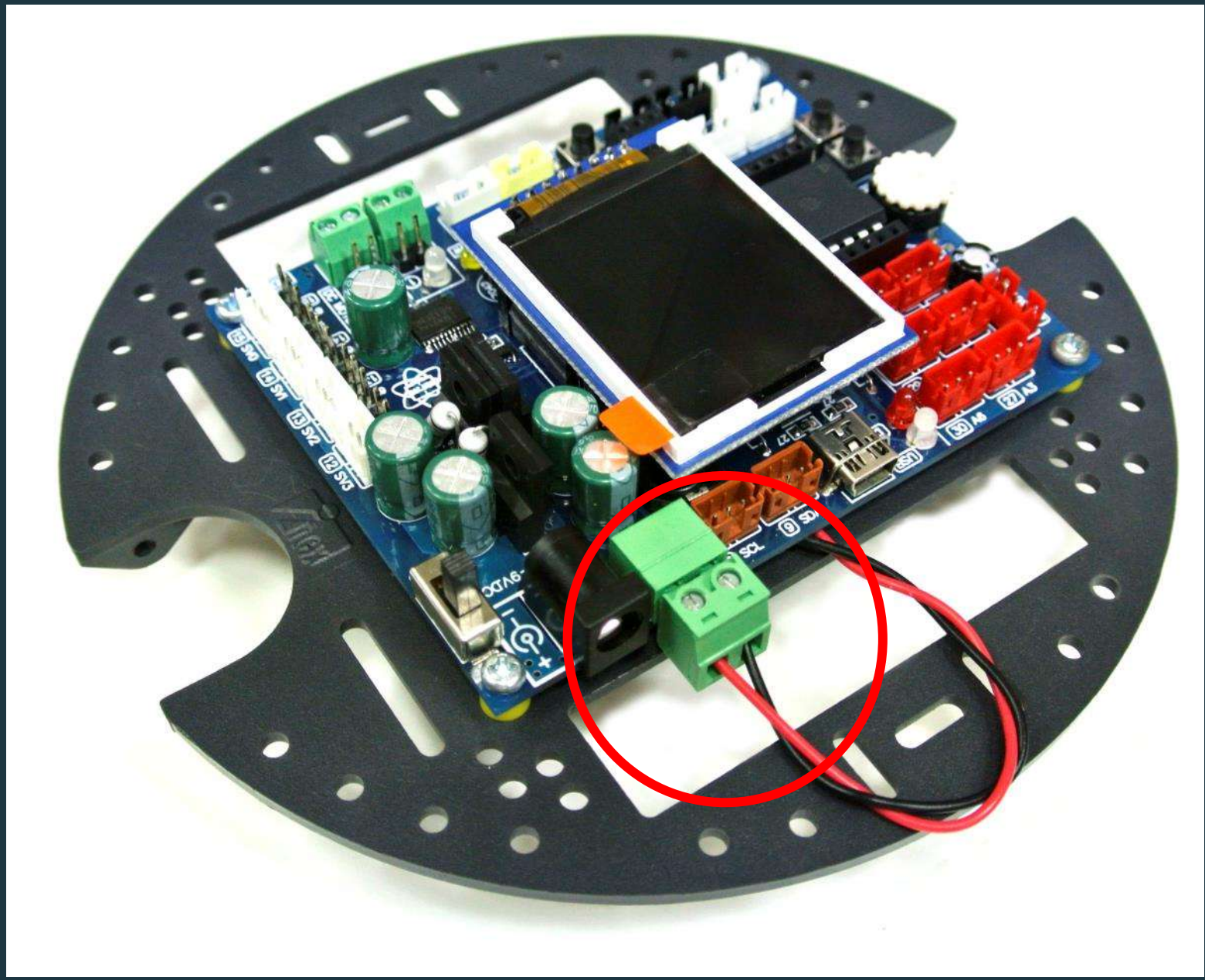


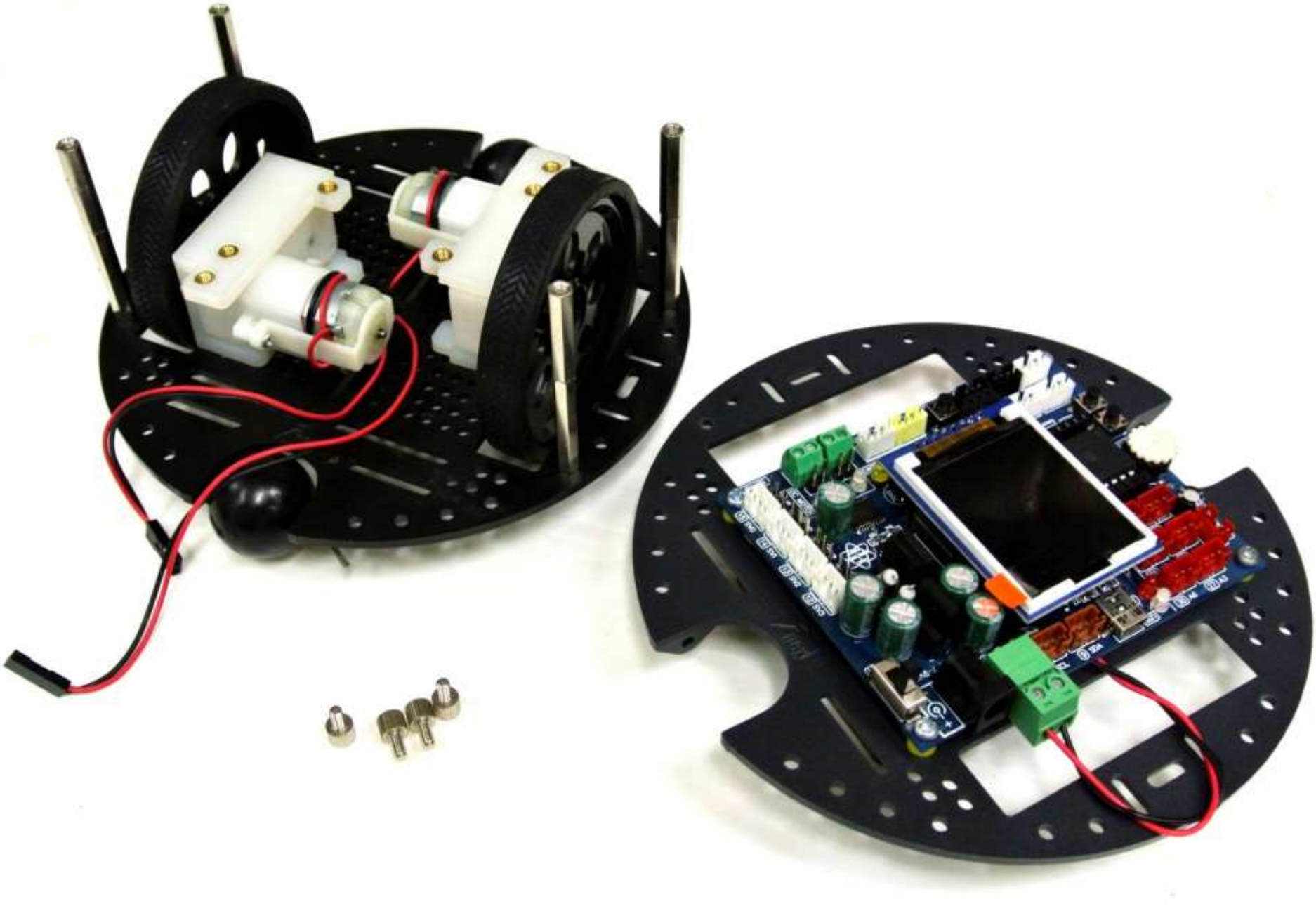


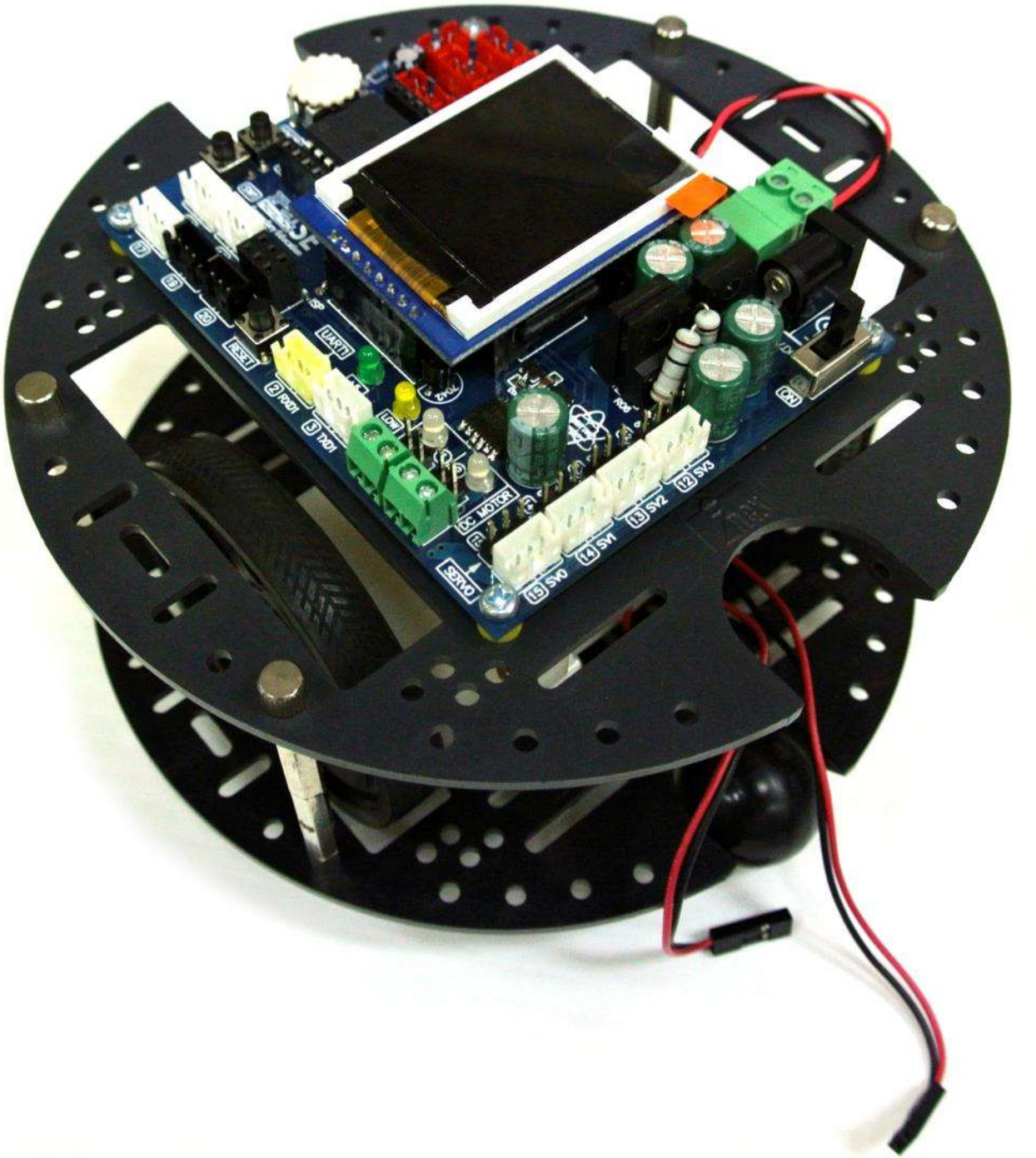

ALKALINE
AA
ALKALINE
AA
ALKALINE
AA

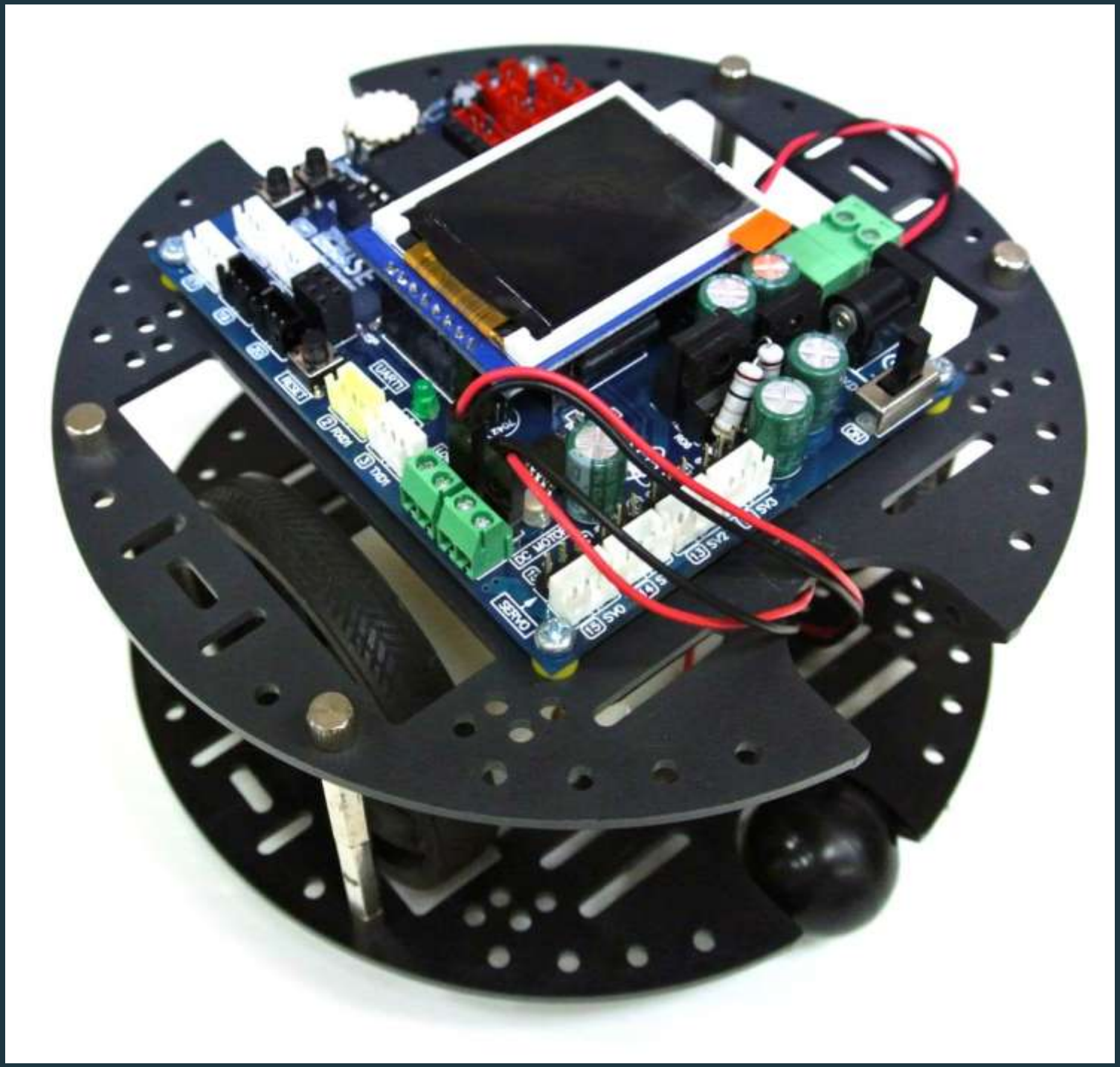

SCL
SDA

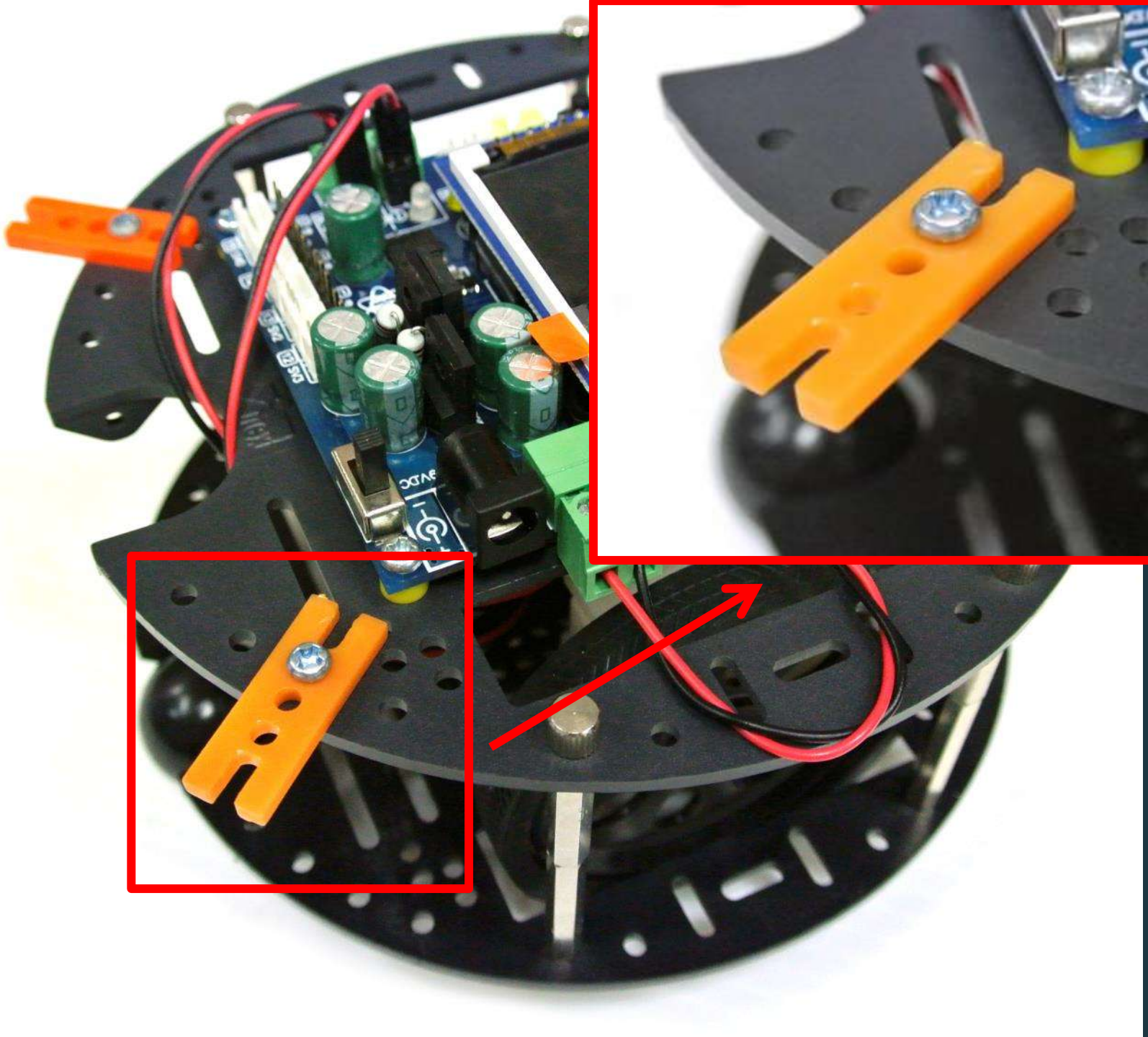


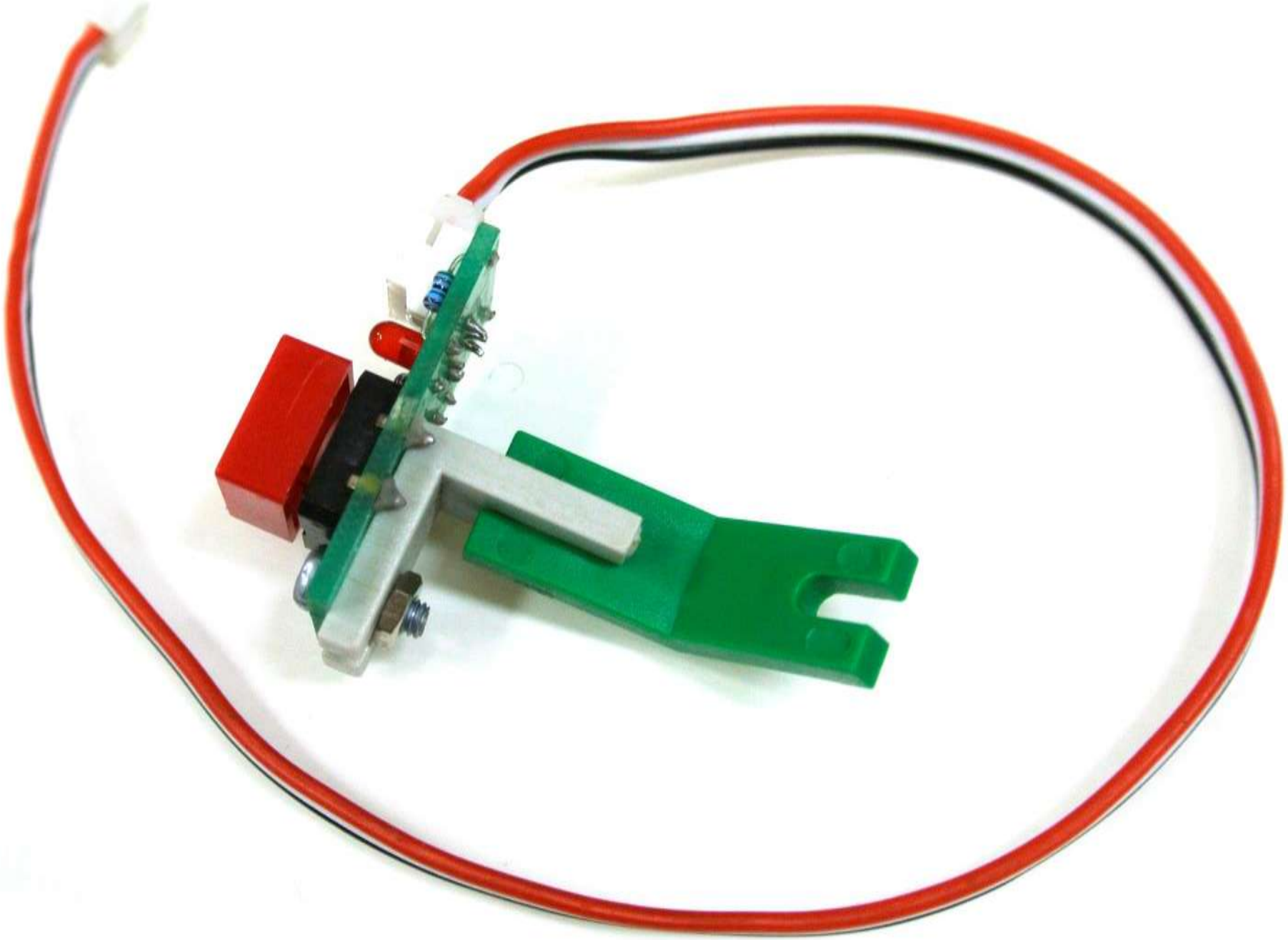


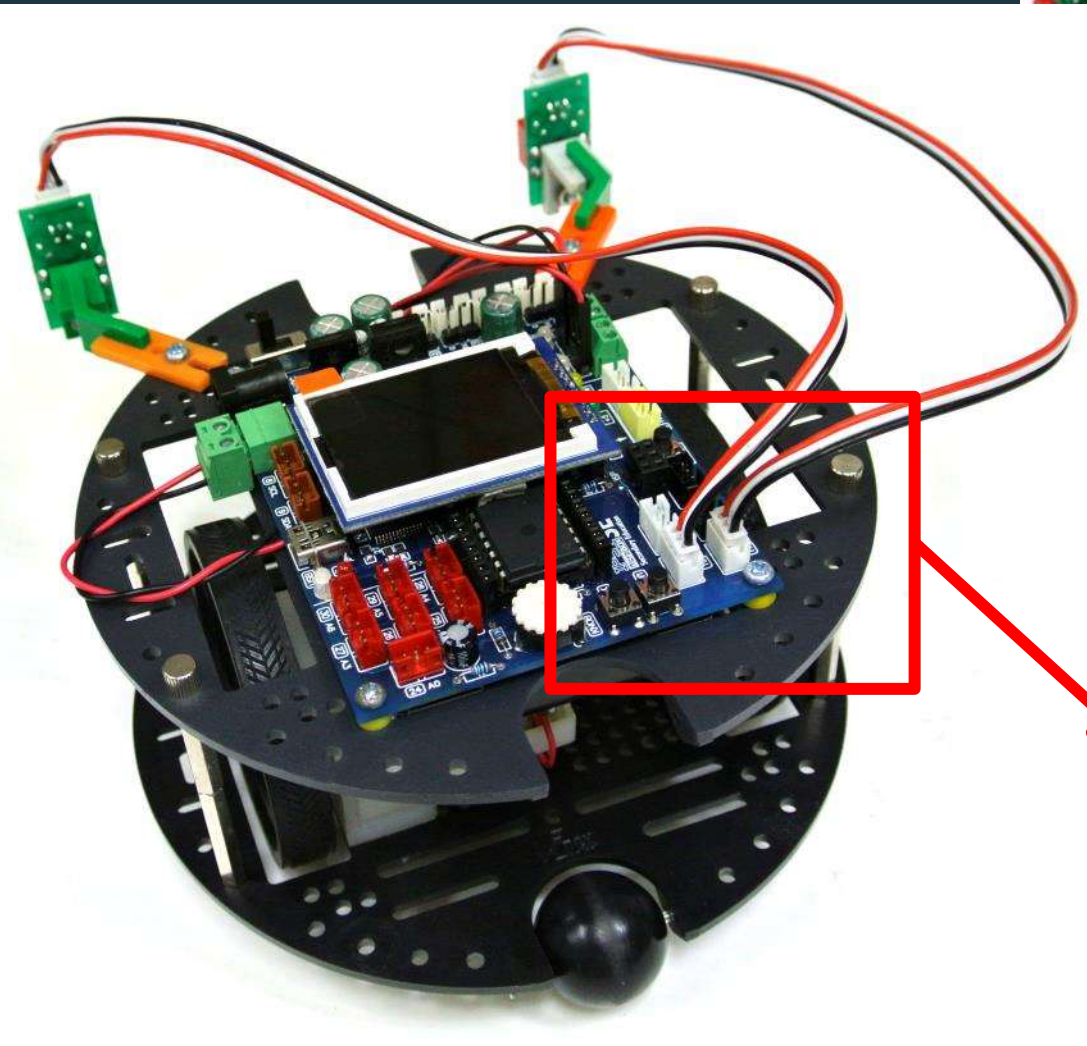


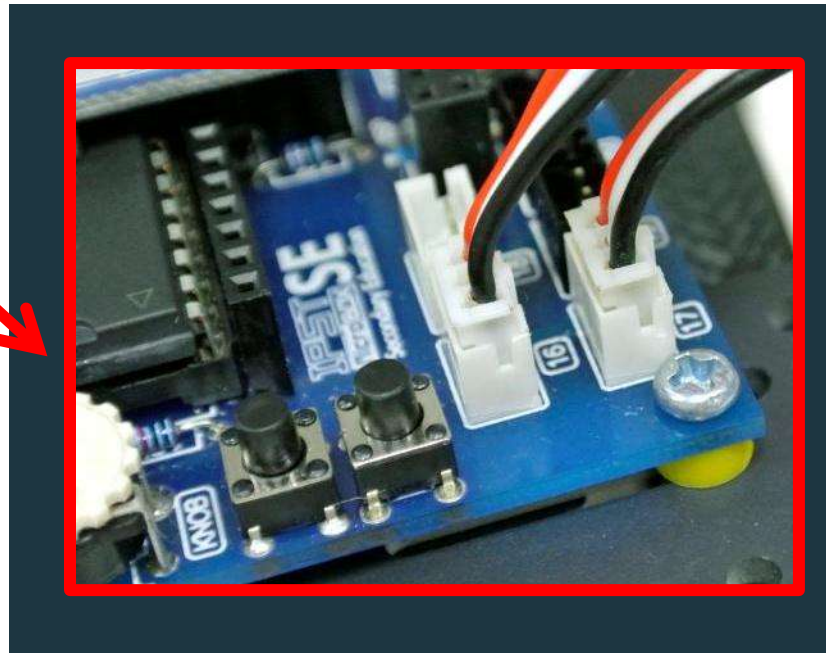

UART
DC MOTOR
SERVO
SV0
SV1
SV2
SV3

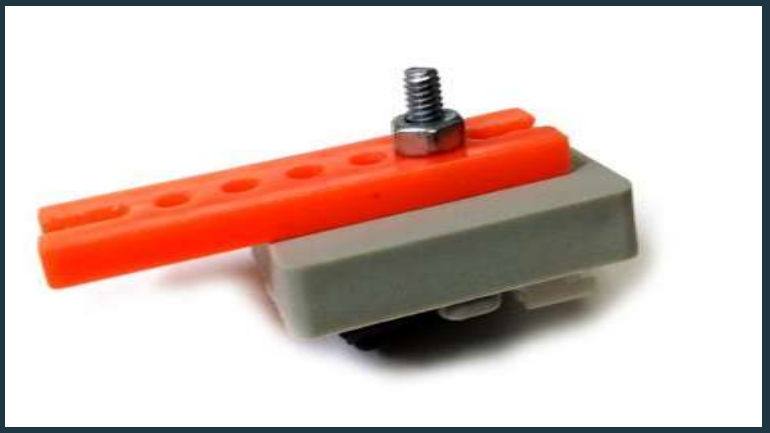


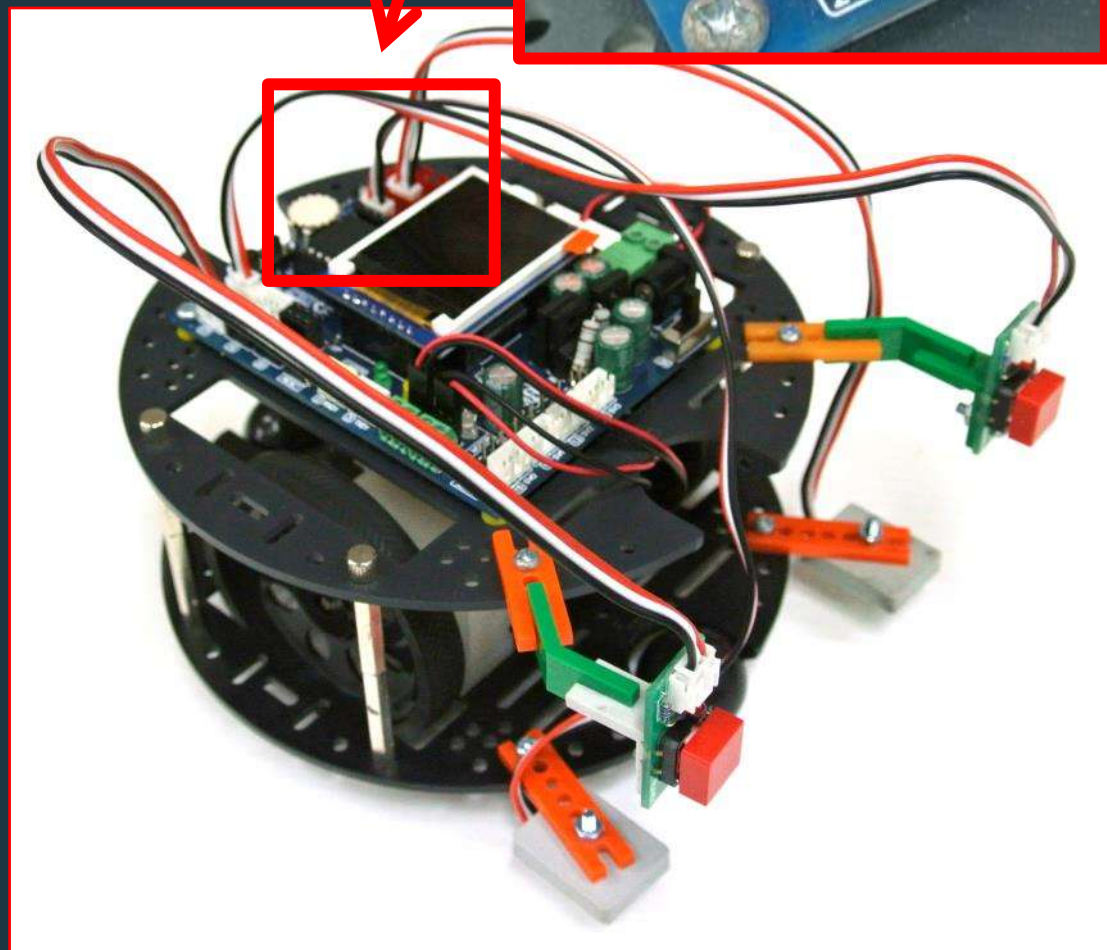











A5
27
A3
24
A0