

ROBO-SOCCER



TPA JUNIOR ROBOT 2025

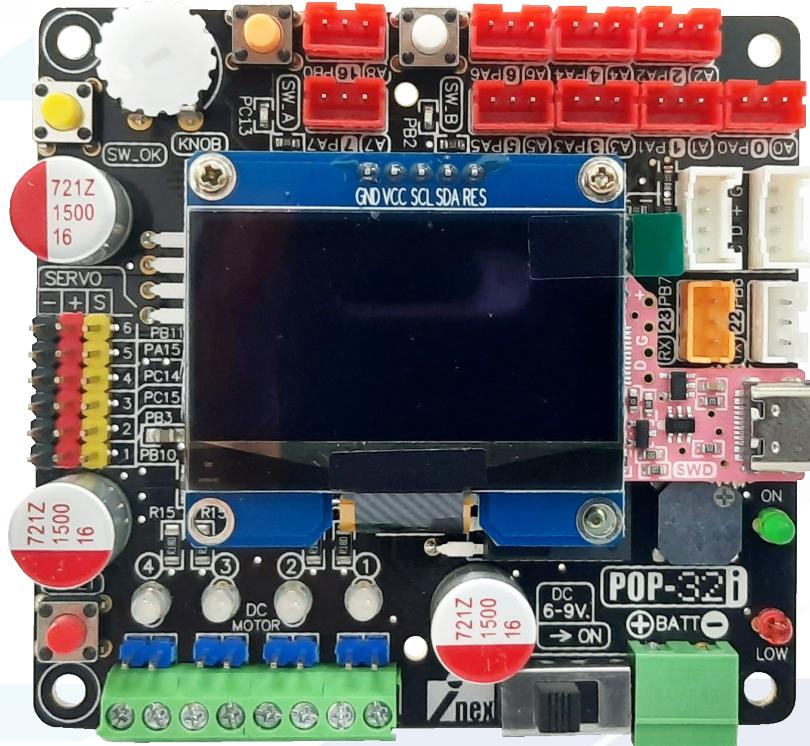




เริ่มแรกรู้จักบอร์ด POP-32i

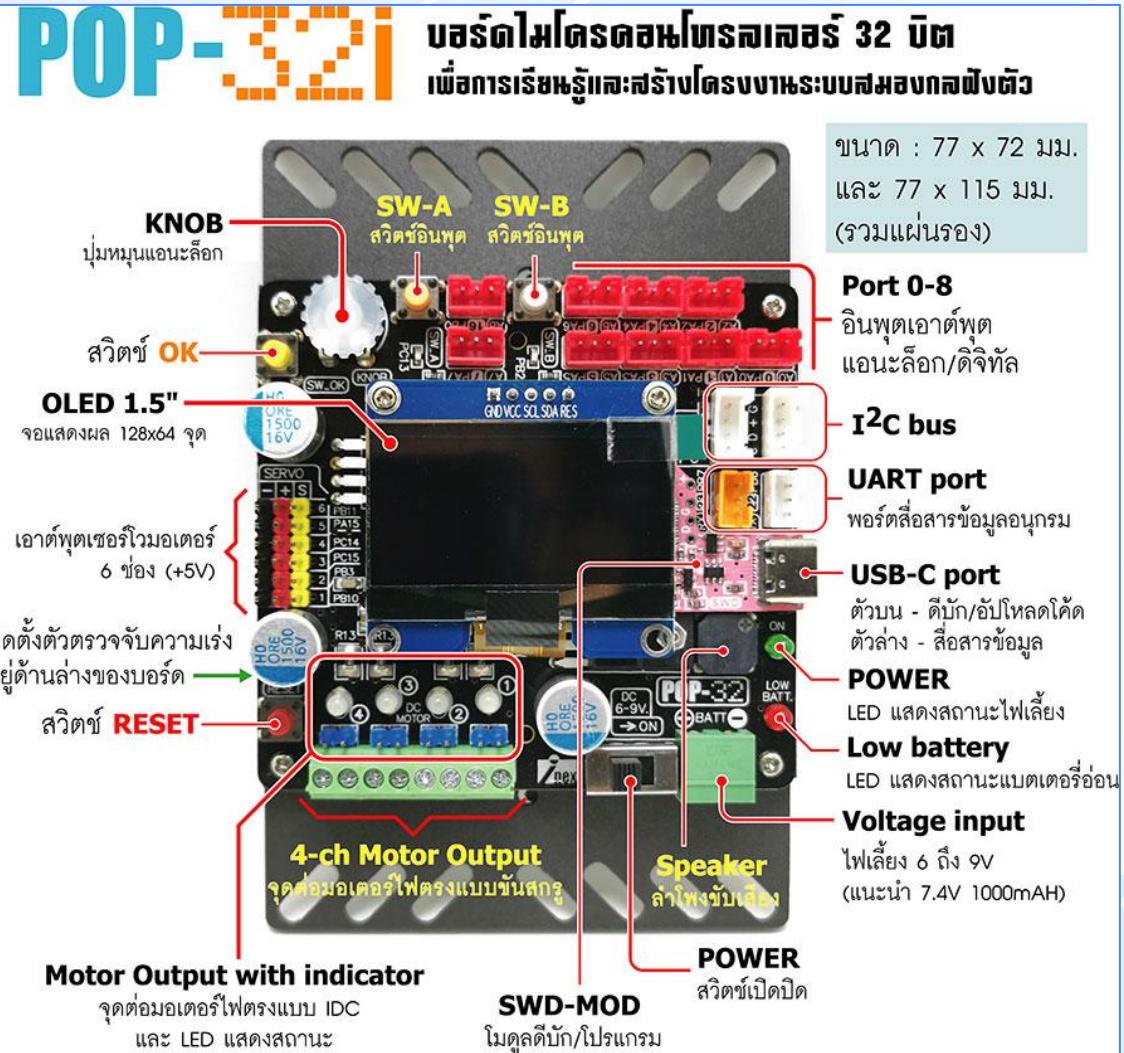
คุณสมบัติของบอร์ด

- ใช้ไมโครคอนโทรลเลอร์ 32 บิต เบอร์ STM32F103CBT6 มี



ส่วนประกอบของบอร์ด

POP-32i





แหล่งจ่ายไฟสำหรับหุ่นยนต์ Lithium Polymer Battery



สายไฟสีแดง เป็นข้าวบาก

สายไฟสีดำ เป็นข้าวลับ

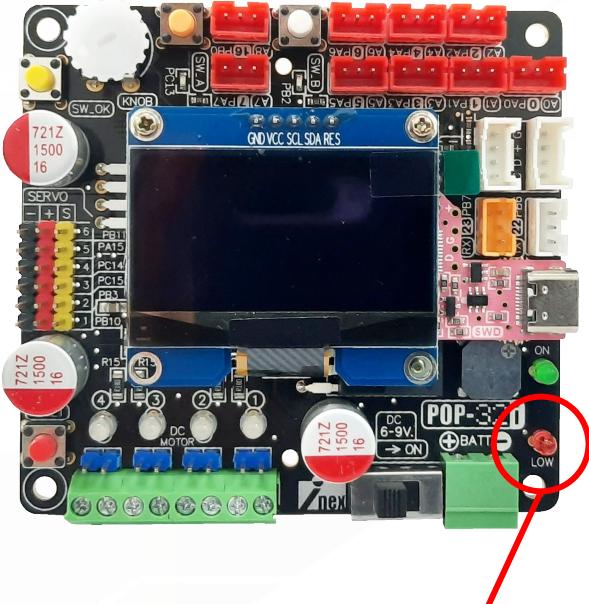


คุณสมบัติของแบตเตอรี่

- ใช้แบตเตอรี่ 2 เซล 7.4V (1 เซล = 3.7V)
- จ่ายกระแสได้ต่อเนื่อง 1,100mA ต่อชั่วโมง
- จ่ายกระแสชั่วขณะได้ 30 เท่า
- ชาร์จได้ 5 เท่า (5,500mA) ใน 20 นาที

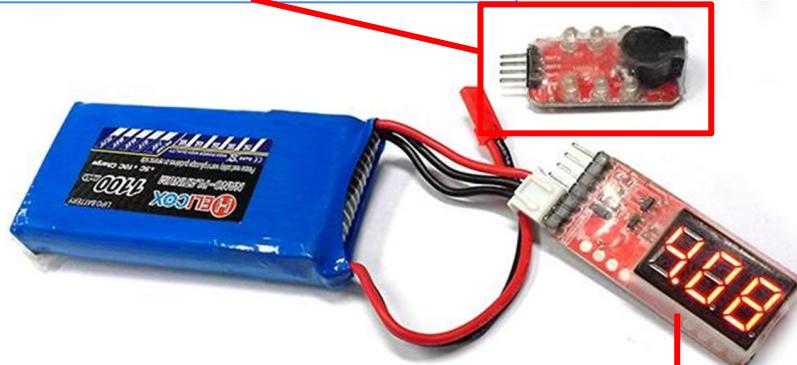


วิธีตรวจสอบแรงดันแบตเตอรี่ต่อ

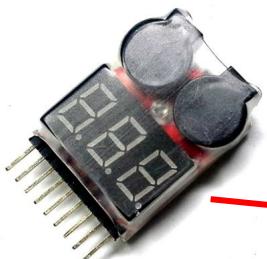


วิธีที่ 1 LED LOW กระพริบ

วิธีที่ 2 ใช้โมดูลแสดงไฟและเตือน



วิธีที่ 3 ใช้โมดูลวัดแรงดัน



วิธีที่ 4 ใช้โมดูลวัดแรงดันและเสียงเตือน



การชาร์จแบตเตอรี่

เครื่องชาร์จแบบที่ 1

เครื่องชาร์จแบบประจำที่

เครื่องชาร์จชนิดนี้จะสามารถชาร์จแบตเตอรี่ได้หลายชนิด เช่น ถ่าน Ni-MH, แบตเตอรี่ Li-Po, แบตเตอรี่รยอนต์ เป็นต้น นอกจากนั้นยังสามารถตั้งกระแสในการชาร์จช้าหรือเร็วได้อีกด้วย





เครื่องชาร์จแบบที่ 2

เครื่องชาร์จแบบ USB

เครื่องชาร์จชนิดนี้จะใช้ไฟ

จากอะแดปเตอร์แบบ USB เป็นตัวจ่ายไฟ โดย
การชาร์จจะใช้เฉพาะกับแบตเตอรี่ Li-Po
ขนาด 7.4V 1,000mA



การติดตั้งโปรแกรม Arduino IDE



- เข้าไปที่เว็บไซต์ www.arduino.cc จากนั้นคลิกที่คำว่า “SOFTWARE”



www.arduino.cc

The screenshot shows the official Arduino website at https://www.arduino.cc. The top navigation bar includes links for PROFESSIONAL, EDUCATION, STORE, HARDWARE, SOFTWARE (which is highlighted with a black box), CLOUD, DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. Below the navigation is a search bar labeled "Search on Arduino.cc". The main content area features several sections: "WHAT IS ARDUINO?" with an image of an Arduino Uno board; "ARDUINO PLUG AND MAKE KIT" with two people holding boards; "Arduino is the ONE!" with various tools and components; and sections for "BUY AN ARDUINO", "LEARN ARDUINO", "DONATE", "ARDUINO IN THE CLOUD", and "CAREERS". A "Discover Now" button is also visible.



2. เลือกดาวน์โหลดโปรแกรมให้ตรงกับระบบปฏิบัติการที่ใช้อยู่

(ในที่นี้เป็นระบบปฏิบัติการ Windows)

Downloads



Arduino IDE 2.3.4

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

[SOURCE CODE](#)

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits เลือกที่นี่

Windows MCL installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)



3. กด JUST DOWNLOAD เพื่อเข้าสู่ขั้นตอนต่อไป

Download Arduino IDE & support its progress

Since the 1.x release in March 2015, the Arduino IDE has been downloaded **91,688,277** times — impressive! Help its development with a donation.

\$3

\$5

\$10

\$25

\$50

Other

CONTRIBUTE AND DOWNLOAD

or

JUST DOWNLOAD

เลือกที่นี่



4. กด JUST DOWNLOAD อีกครั้ง เพื่อทำการดาวน์โหลดโปรแกรม

Stay in the Loop: Join Our Newsletter!

As a beginner or advanced user, you can find inspiring projects and learn about cutting-edge Arduino products through our **weekly newsletter!**

email *

I confirm to have read the [Privacy Policy](#) and to accept the [Terms of Service](#) *

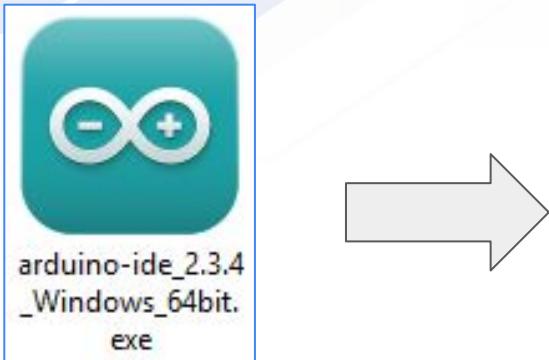
I would like to receive emails about special deals and commercial offers from Arduino.

SUBSCRIBE & DOWNLOAD

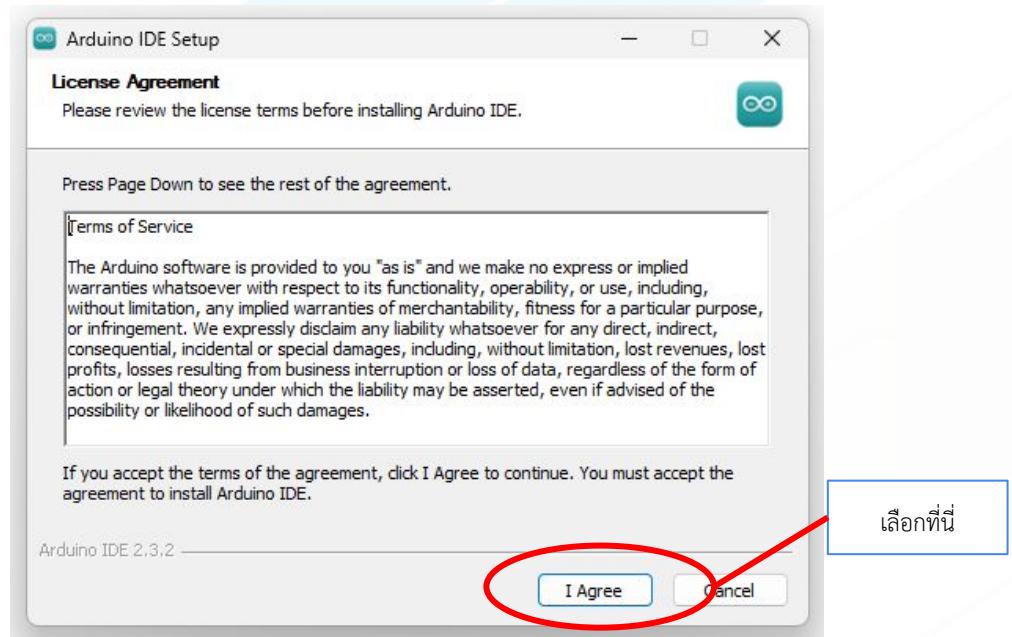
— or —

JUST DOWNLOAD

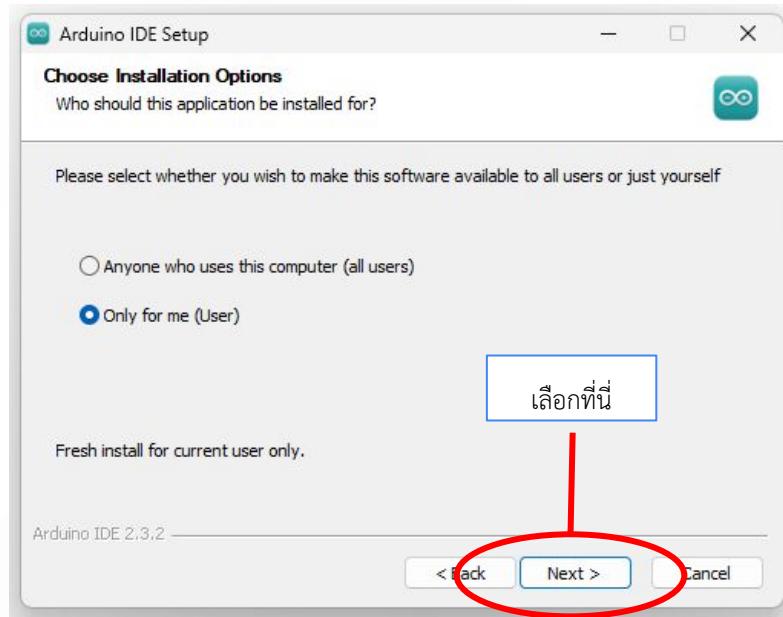
เลือกที่นี่



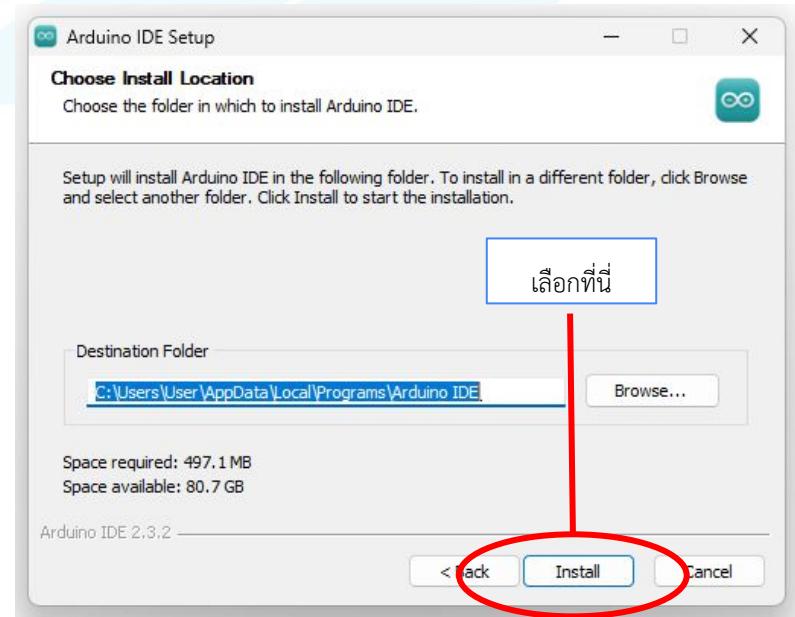
5. เมื่อดาวน์โหลดเสร็จ จะได้ไฟล์ตามรูป ให้ทำการดับเบิลคลิก เพื่อเริ่มทำการติดตั้งโปรแกรม



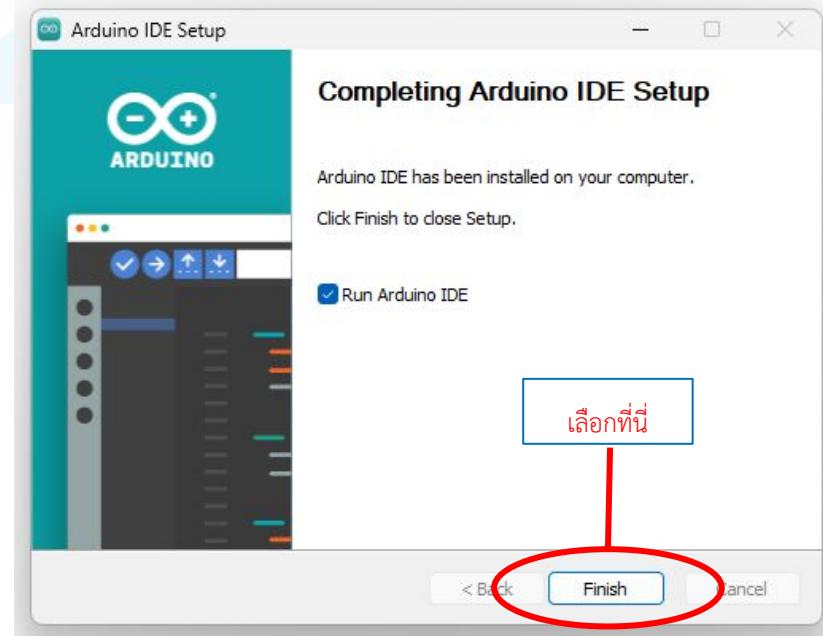
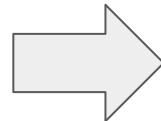
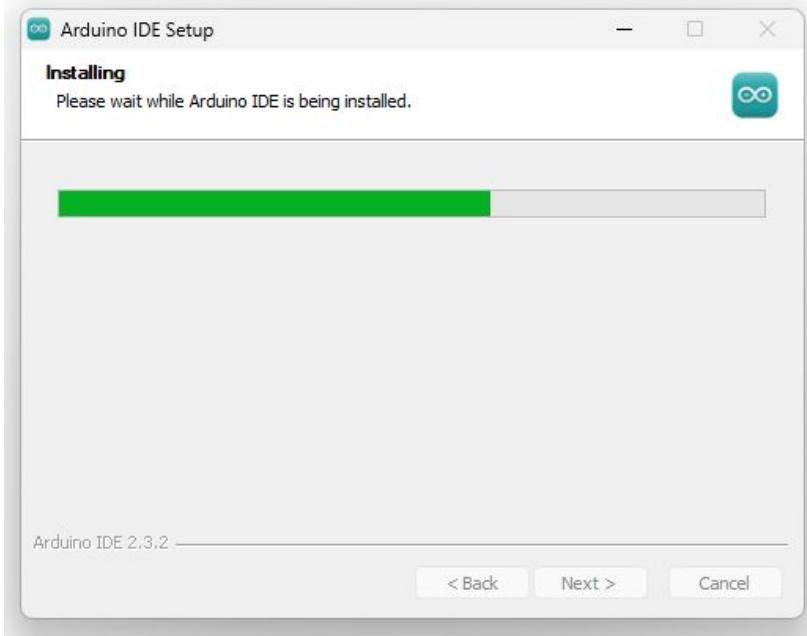
6. กดปุ่ม I Agree เพื่อยอมรับข้อตกลง



7. กดปุ่ม Next



8. กดปุ่ม I Agree



9. ขณะนี้โปรแกรมกำลังทำการติดตั้ง

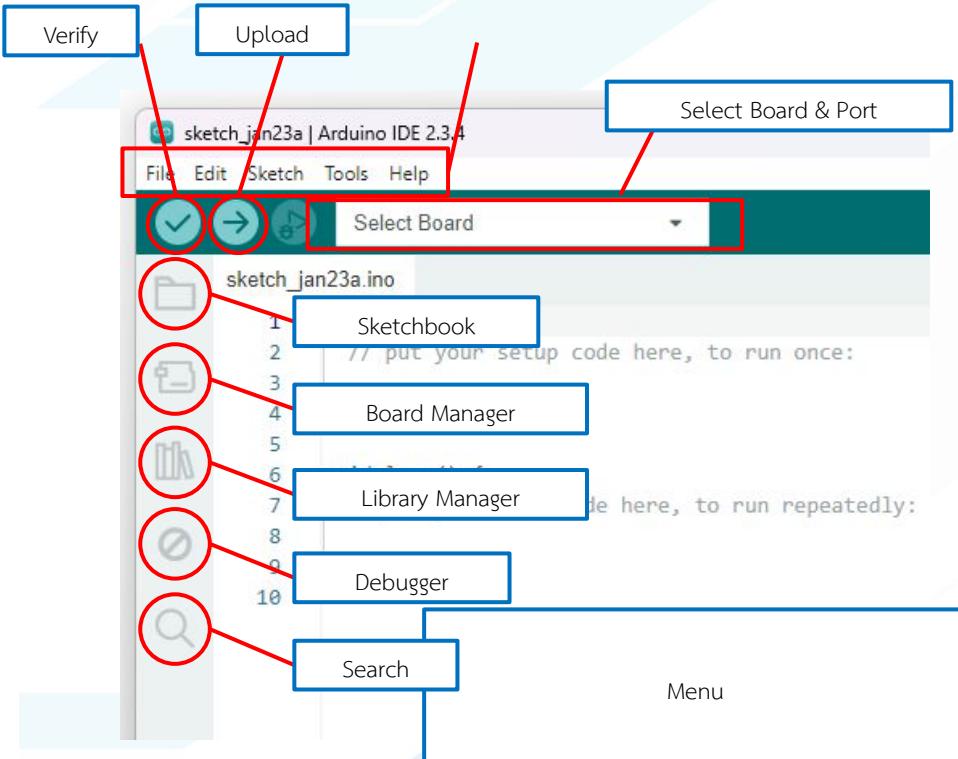
10. กดปุ่ม Finish เพื่อสิ้นสุดการติดตั้ง



รู้จักกับโปรแกรม Arduino IDE

ส่วนต่างๆ ของโปรแกรม

- Menu ใช้เลือกและกำหนดค่าสั่งต่างๆ ของโปรแกรม
- Verify ใช้ตรวจสอบการเขียนคำสั่งในโค้ด
- Upload ใช้อัปโหลดโค้ดไปยังบอร์ด
- Select Board & Port เลือกบอร์ดและพอร์ตที่ใช้เชื่อมต่อ
- Sketchbook ใช้ดูไฟล์สเก็ตซ์ที่จัดเก็บอยู่ในคอมพิวเตอร์
- Boards Manager ค้นหาและติดตั้งแพคเกจชาร์ดแวร์ของบอร์ดที่ใช้งาน
- Library Manager ค้นหาและติดตั้งไลบรารีของบอร์ดที่ใช้งาน
- Debugger ใช้ทดสอบและดีบักโค้ดตามเวลาจริง
- Search ค้นหาคำสำคัญของโค้ดในสเก็ตซ์



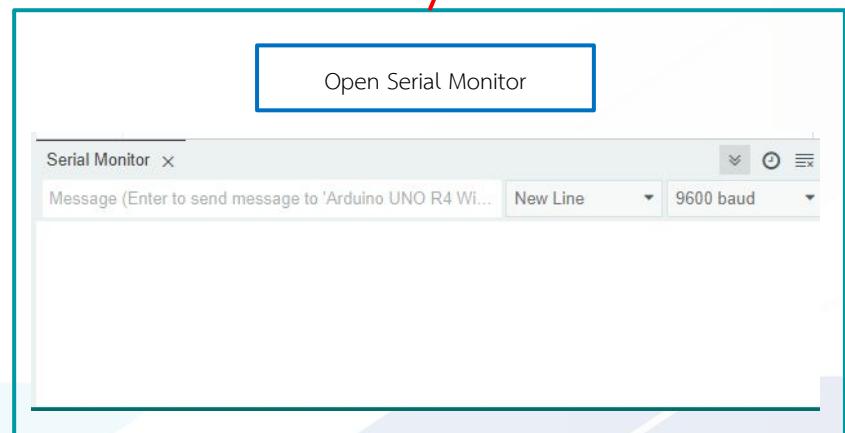
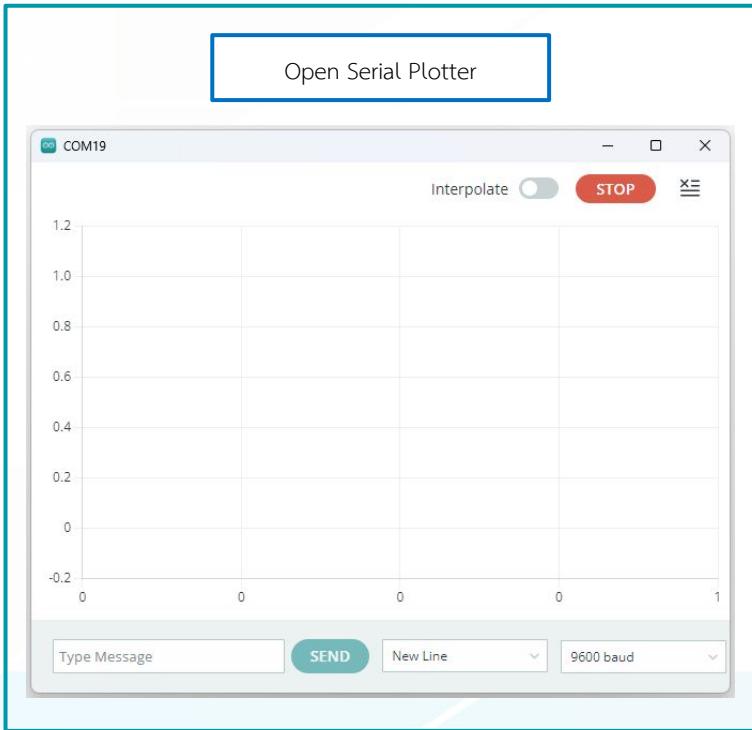


- Open Serial Monitor เปิดหน้าต่างสื่อสารและแสดงข้อมูลอนุกรม

ข้อมูลอนุกรม

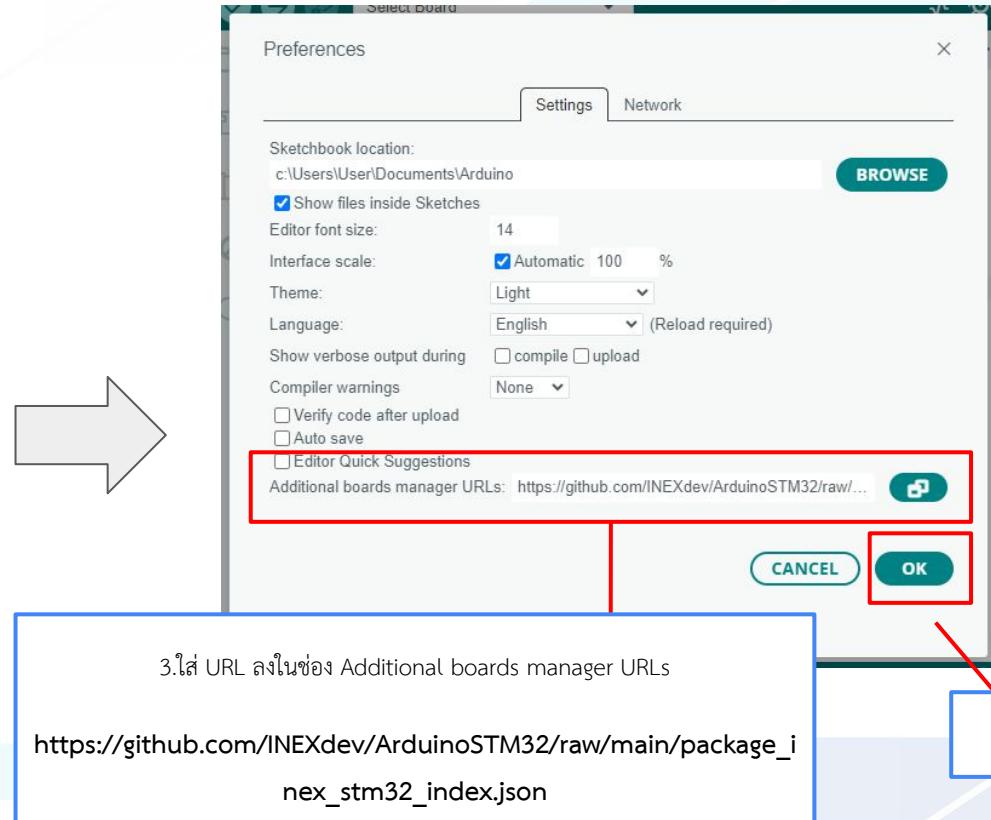
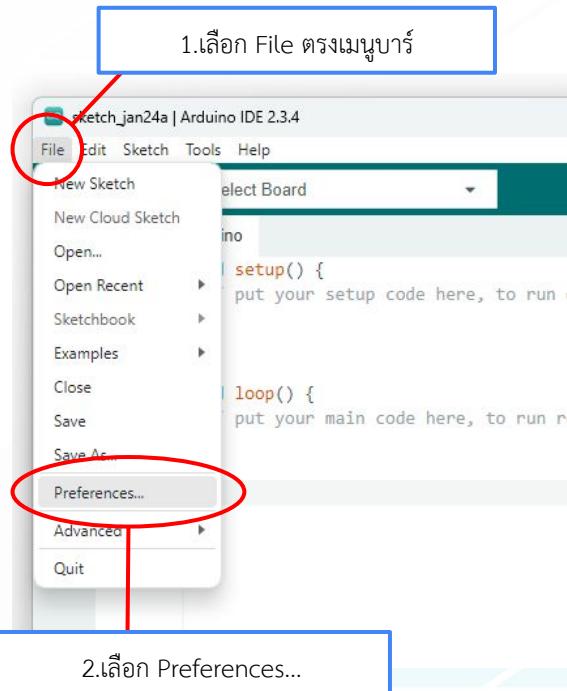
- Open Serial Plotter เปิดหน้าต่างแสดงข้อมูลในแบบสร้างเส้นกราฟ

สร้างเส้นกราฟ



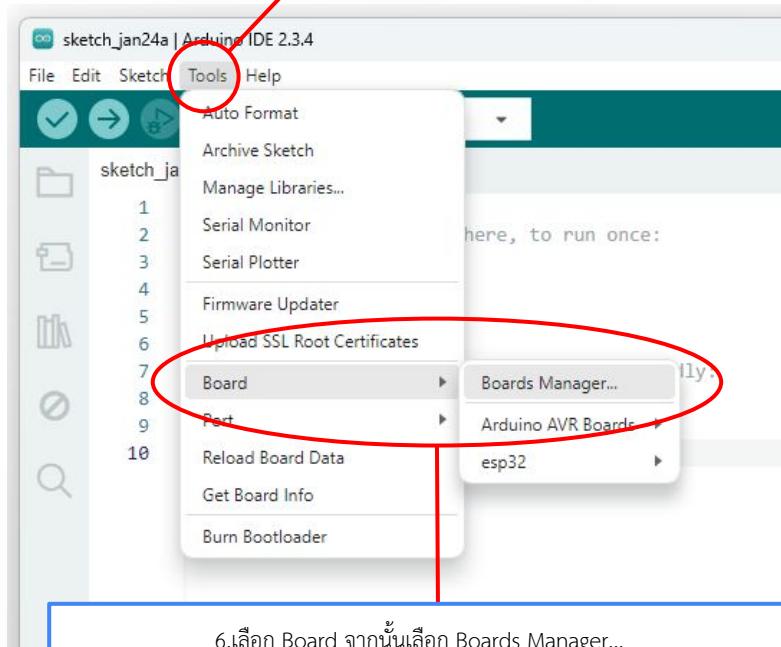


การตั้งค่าโปรแกรม Arduino IDE เพื่อเชื่อมต่อบอร์ด POP-32i



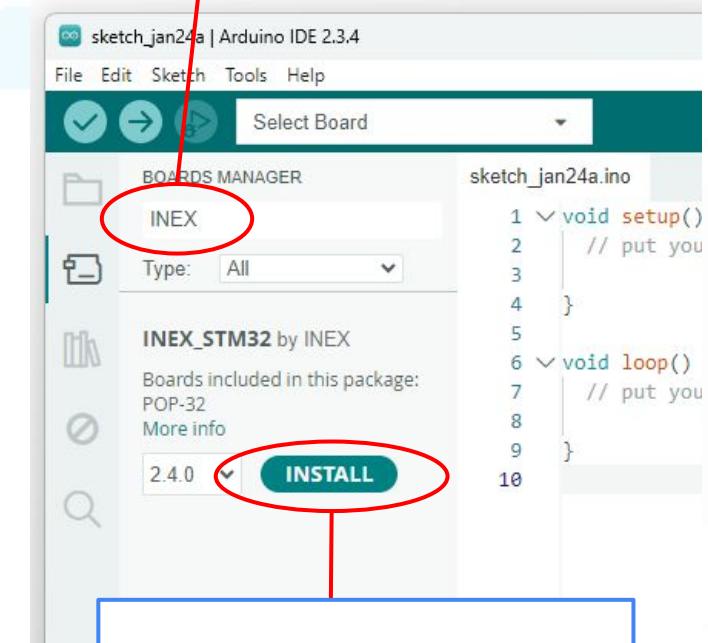


5.เลือก Tools ตรงเมนูบาร์



6.เลือก Board จากนั้นเลือก Boards Manager...

7.พิมพ์คำว่า “INEX” ลงในช่องค้นหา



8.กดปุ่ม INSTALL เพื่อทำการติดตั้ง INEX_STM32



9.รูปของโปรแกรมกำลังติดตั้ง INEX_STM32

sketch_jan24a | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Select Board

BOARDS MANAGER

INEX

Type: All

INEX_STM32 by INEX

Boards included in this package:
POP-32
More info

2.4.0

sketch_jan24a.ino

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8  
9 }  
10
```

Output

Downloading packages
INEX_STM32:xpack-arm-none-eabi-gcc@10.3.1-2.3

Processing INEX_STM32:2.4.0: INEX_STM32:xpack-arm-none-eabi-gcc@10.3.1-2.3

Ln 10, Col 1 X No board selected 1

sketch_jan24a | Arduino IDE 2.3.4

File Edit Sketch Tools Help

Select Board

BOARDS MANAGER

INEX

Type: All

INEX_STM32 by INEX

2.4.0 installed

Boards included in this package:
POP-32
More info

2.4.0

sketch_jan24a.ino

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8  
9 }  
10
```

Output

INEX_STM32:STM32Tools@2.1.1 installed
Installing INEX_STM32:CMSIS@5.7.0
Configuring tool.
INEX_STM32:CMSIS@5.7.0 installed
Installing platform INEX_STM32:stm32@2.4.0
Configuring platform.
Platform INEX_STM32:stm32@2.4.0 installed

Ln 10, Col 1 X No board selected 1

10.รูปโปรแกรมเมื่อติดตั้ง
INEX_STM32 เรียบร้อยแล้ว ปุ่ม
INSTALL จะเปลี่ยนเป็น REMOVE



การติดตั้งโปรแกรม STM32 Cube Programmer

The screenshot shows the official STM32CubeProg software page on the ST website. The URL in the browser address bar is highlighted with a red box: <https://www.st.com/en/development-tools/stm32cubeprog.html>. A red arrow points from the top right of the page towards the user profile icon in the top right corner of the browser window. The user profile icon is also highlighted with a red box.

1. เข้าไปที่เว็บไซต์ <https://www.st.com/en/development-tools/stm32cubeprog.html>

2. กดเลือกไอคอนรูปคน เพื่อทำการเข้าสู่ระบบหรือสมัครสมาชิก ก่อนทำการดาวน์โหลด



3. หลังจาก Log in เรียบร้อยแล้ว ให้ก็ลับมาที่เว็บไซต์ <https://www.st.com/en/development-tools/stm32cubeprog.html> อีกครั้ง แล้วกดที่ [Get Software](#)

The screenshot shows the STM32CubeProg product page on the ST website. At the top, there's a navigation bar with links for Careers, Sample & buy, Support & community, and language options (日本語, 中文, English). Below the navigation is a main menu with categories like Products, Tools & software, Applications, Solutions, ST Developer Zone, About us, and a shopping cart/icon. The breadcrumb navigation shows the path: Development tools > Software development tools > STM32 software development tools > STM32 programmers > STM32CubeProg. The main content area features the title "STM32CubeProg" (ACTIVE) and "STM32CubeProgrammer software for all STM32". Below the title are two buttons: "Get Software" (highlighted with a red box) and "Download databrief". Further down, there are tabs for Overview, Documentation, and Tools & Software. The "Overview" tab is selected. A "Product overview" section follows, containing a "Description" heading and a paragraph about the tool. At the bottom right of the page, there's a yellow "Feedback" button.



Get Software

Part Number	General Description	Supplier	Download	All versions
+ STM32CubePrg-Lin	STM32CubeProgrammer software for Linux	ST	Get latest	Select version
+ STM32CubePrg-Mac	STM32CubeProgrammer software for Mac	ST	Get latest	Select version
+ STM32CubePrg-W32	STM32CubeProgrammer software for Win32	ST	Get latest	Select version
+ STM32CubePrg-W64	STM32CubeProgrammer software for Win64	ST	Get latest	Select version

4. กดปุ่ม Get latest ให้ตรงกับระบบปฏิบัติการที่เราใช้ (ในที่นี้เป็นระบบปฏิบัติการ Windows)



License Agreement

[Download as .pdf](#)

Please indicate your acceptance or NON-acceptance by selecting "I ACCEPT" or "I DO NOT ACCEPT" as indicated below in the media.

BY INSTALLING COPYING, DOWNLOADING, ACCESSING OR OTHERWISE USING THIS SOFTWARE PACKAGE OR ANY PART THEREOF (AND THE RELATED DOCUMENTATION) FROM STMICROELECTRONICS INTERNATIONAL N.V., SWISS BRANCH AND/OR ITS AFFILIATED COMPANIES (STMICROELECTRONICS), THE RECIPIENT, ON BEHALF OF HIMSELF OR HERSELF, OR ON BEHALF OF ANY ENTITY BY WHICH SUCH RECIPIENT IS EMPLOYED AND/OR ENGAGED AGREES TO BE BOUND BY THIS SOFTWARE PACKAGE LICENSE AGREEMENT.

Under STMicroelectronics' intellectual property rights and subject to applicable licensing terms for any third-party software incorporated in this software package and applicable Open Source Terms (as defined here below), the redistribution, reproduction and use in source and binary forms of the software package or any part thereof, with or without modification, are permitted provided that the following conditions are met:

1. Redistribution of source code (modified or not) must retain any copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form, except as embedded into microcontroller or microprocessor device manufactured by or for STMicroelectronics or a software update for such device, must reproduce the above copyright notice, this list of

[Additional License Terms for STM32CubeProgrammer 2.18.0](#)

Decline

Accept

Downloads

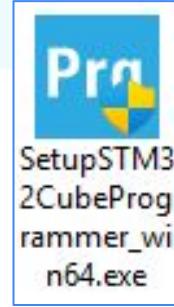


en.stm32cubeprg-win64-v2-18-0.zip

1,011 KB/s - 171 MB

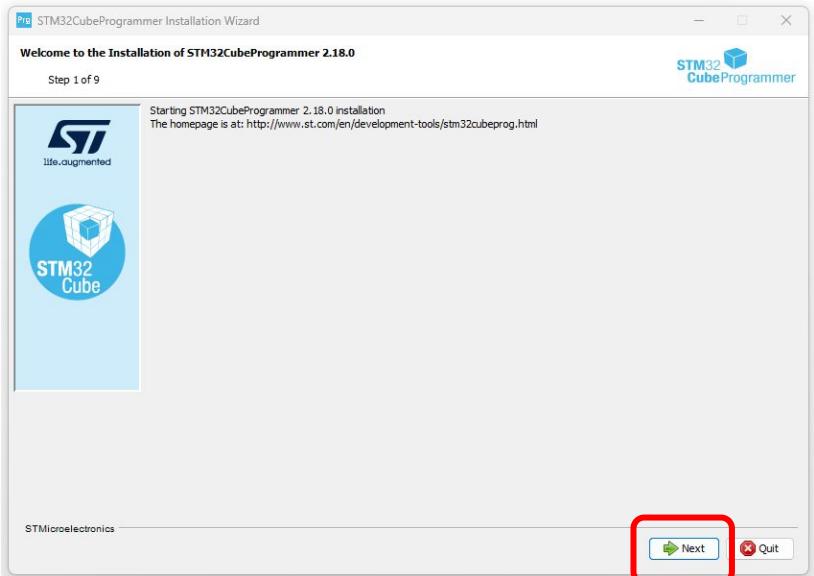
5. กดปุ่ม Accept เพื่อยอมรับข้อตกลง



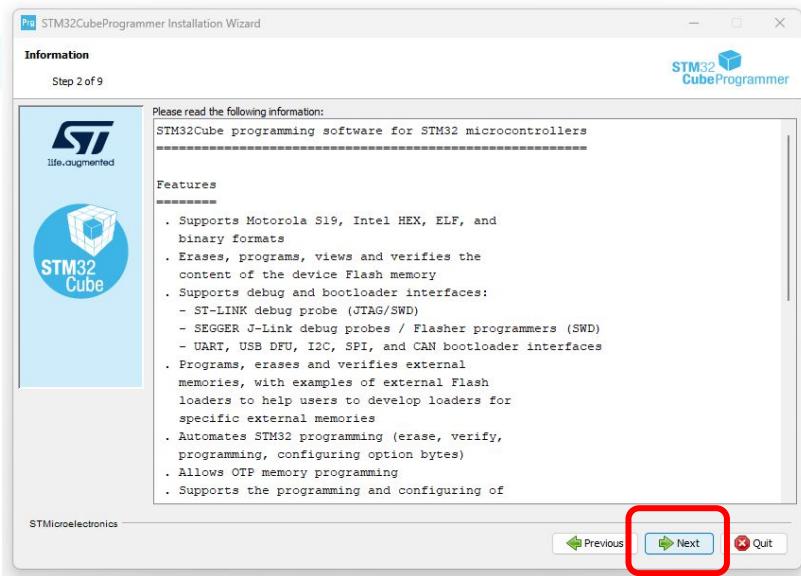


7.ไฟล์ที่ดาวน์โหลดเสร็จเรียบร้อยแล้ว ทำการ ดับเบิลคลิกไฟล์ดังกล่าว เพื่อนำไฟล์ที่อยู่ภายในออกมานะ

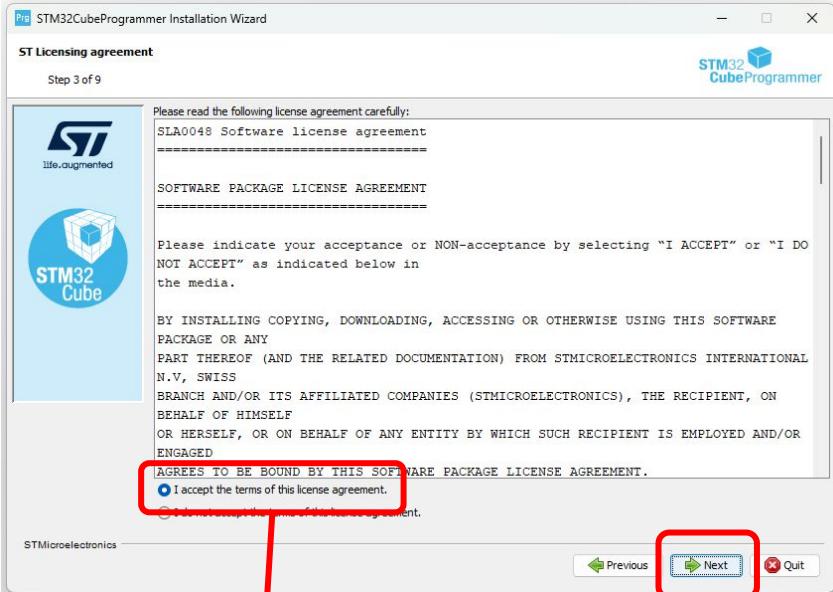
8.ทำการดับเบิลคลิกไฟล์ เพื่อเข้าสู่การติดตั้งโปรแกรม STM32 Cube Programmer



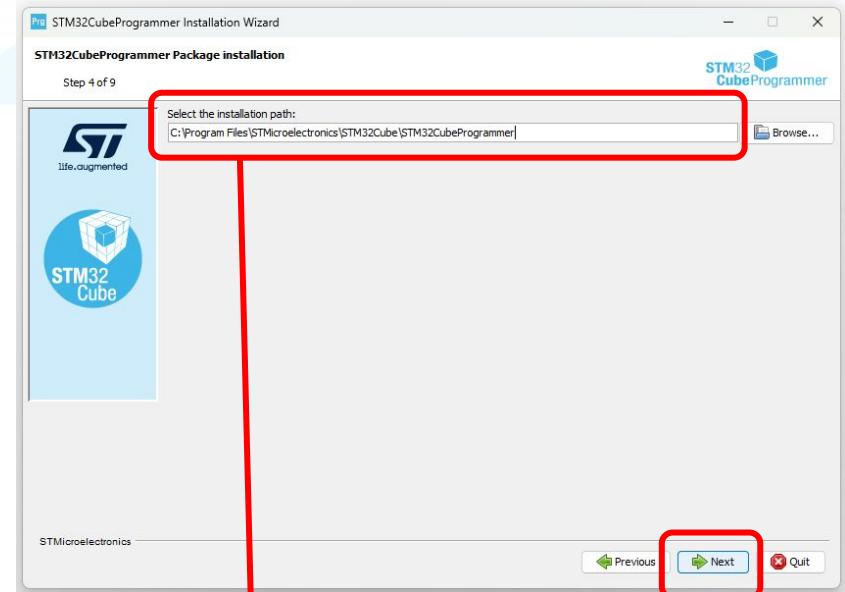
9.กดปุ่ม Next



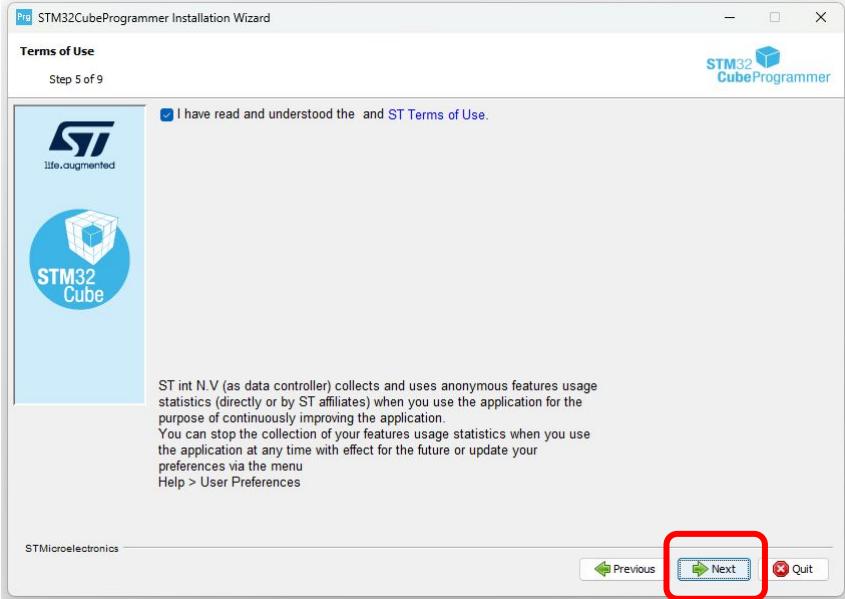
10.กดปุ่ม Next



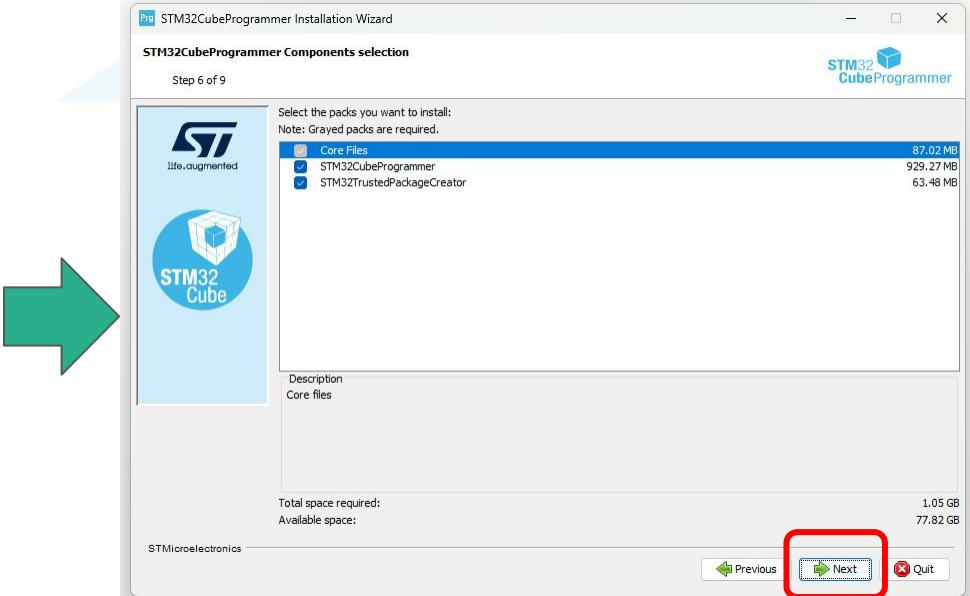
11.ติ๊กที่ I accept... เพื่อยอมรับข้อตกลง และกดปุ่ม Next



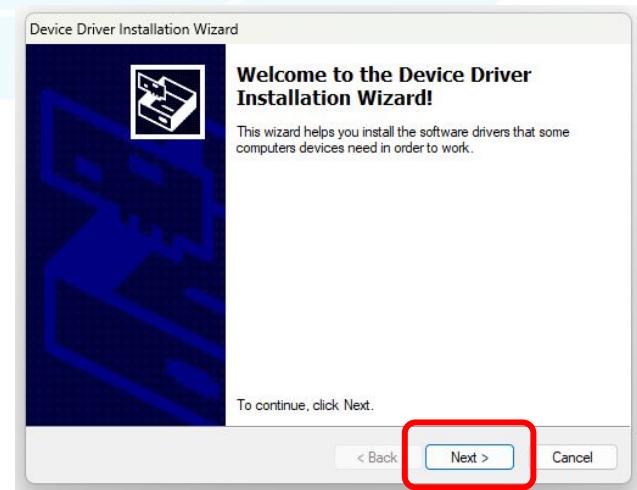
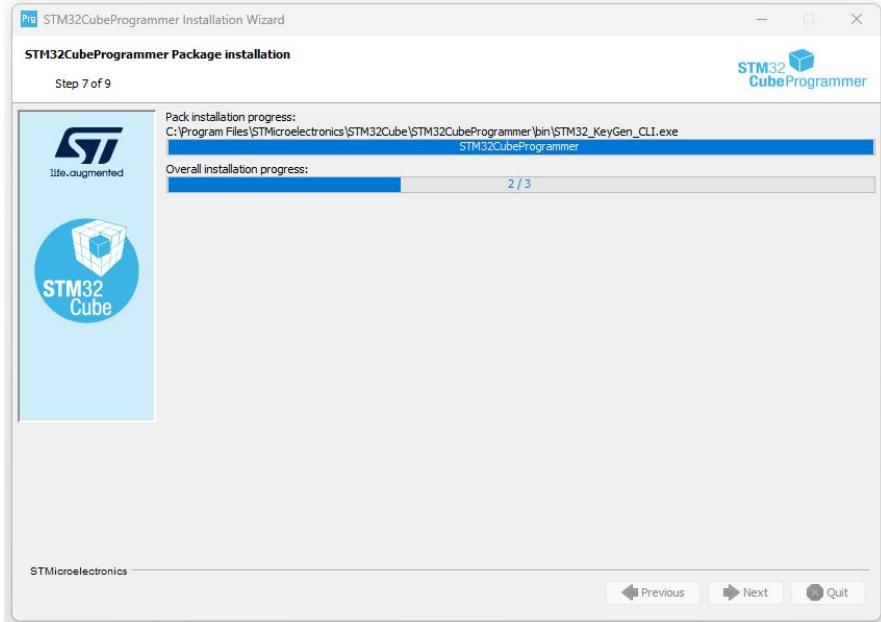
12.เลือกพื้นที่จัดเก็บและกดปุ่ม Next



13. กดปุ่ม Next



14. กดปุ่ม Next

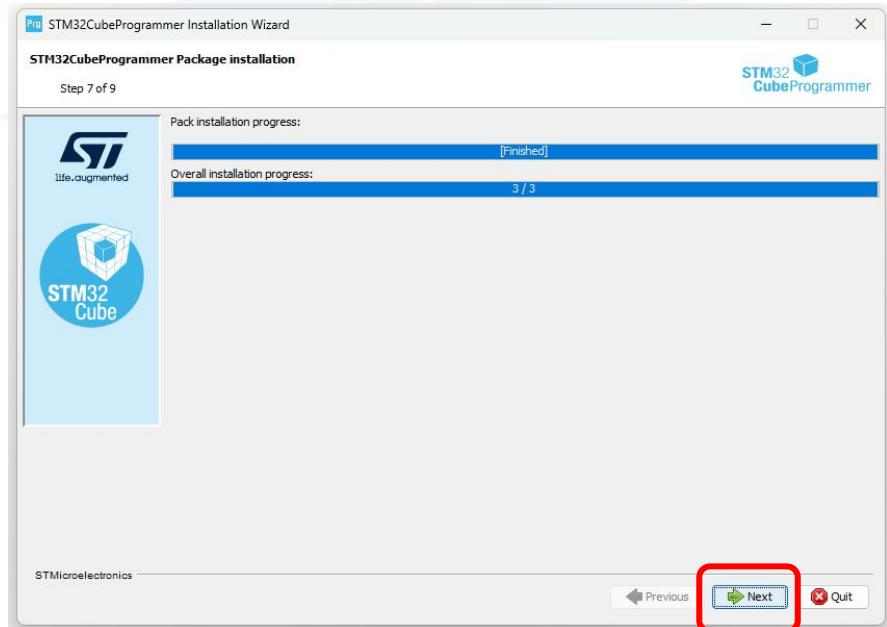


16.ขณะที่โปรแกรมกำลังทำการติดตั้ง จะมีการติดตั้ง Driver
พร้อมกันไปด้วย กดปุ่ม Next

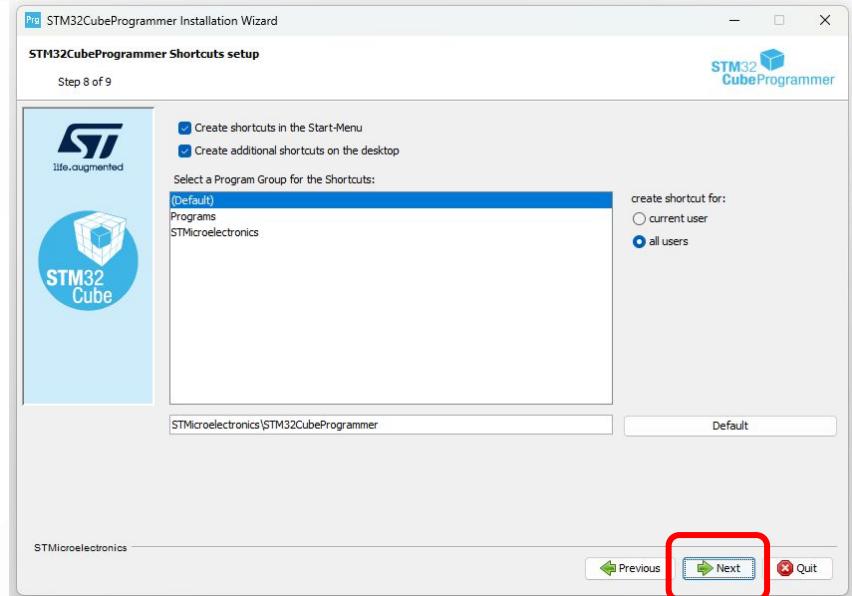
15.โปรแกรมกำลังทำการติดตั้ง



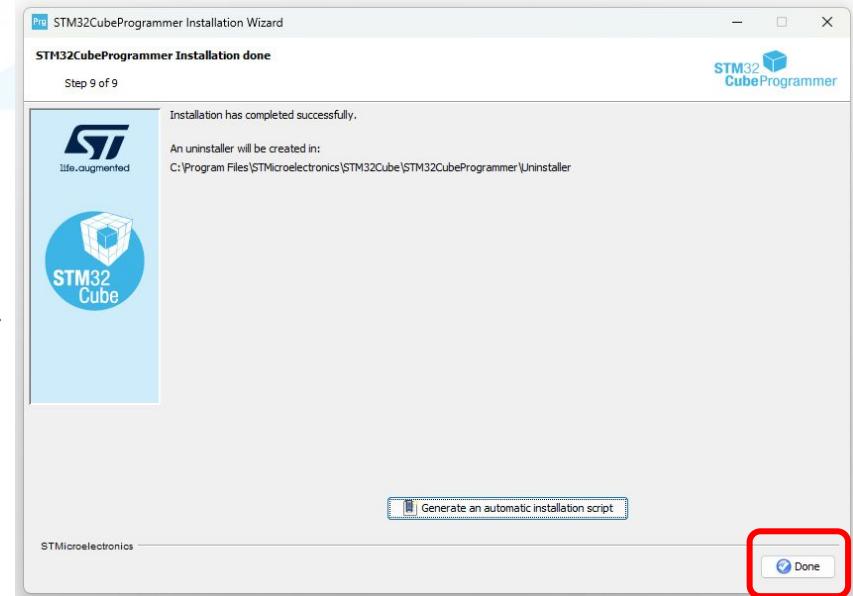
17.รูปเมื่อทำการติดตั้ง Driver เสร็จเรียบร้อยแล้ว กดปุ่ม Finish



18.กดปุ่ม Next



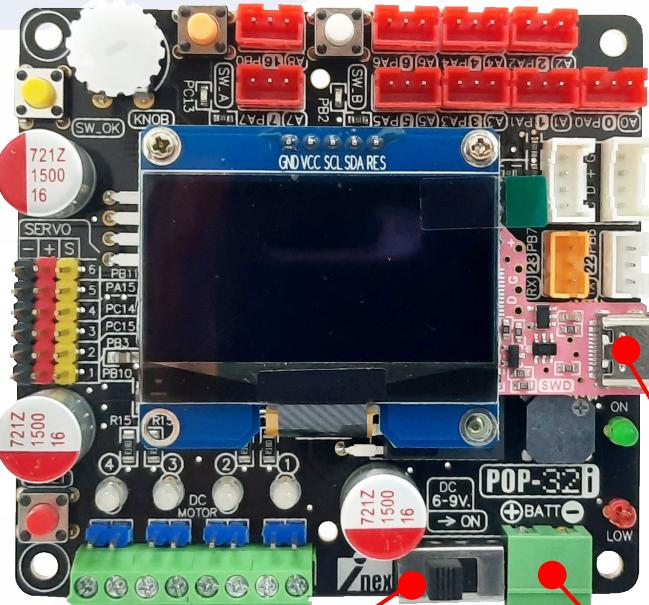
19.กดปุ่ม Next



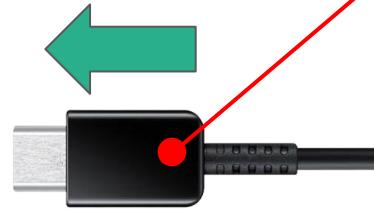
20.หลังจากติดตั้งโปรแกรมเสร็จเรียบร้อย กดปุ่ม
Done



การเชื่อมต่อบอร์ด POP-32i กับคอมพิวเตอร์



1. ต่อสาย USB ที่มาจากเครื่องคอมพิวเตอร์เข้ากับบอร์ด POP-32i



4. เลื่อนสวิตซ์จ่ายไฟไปที่
ตำแหน่ง ON

3. ต่อ LiPo Battery ขนาด 7.4V
ที่จุดนี้

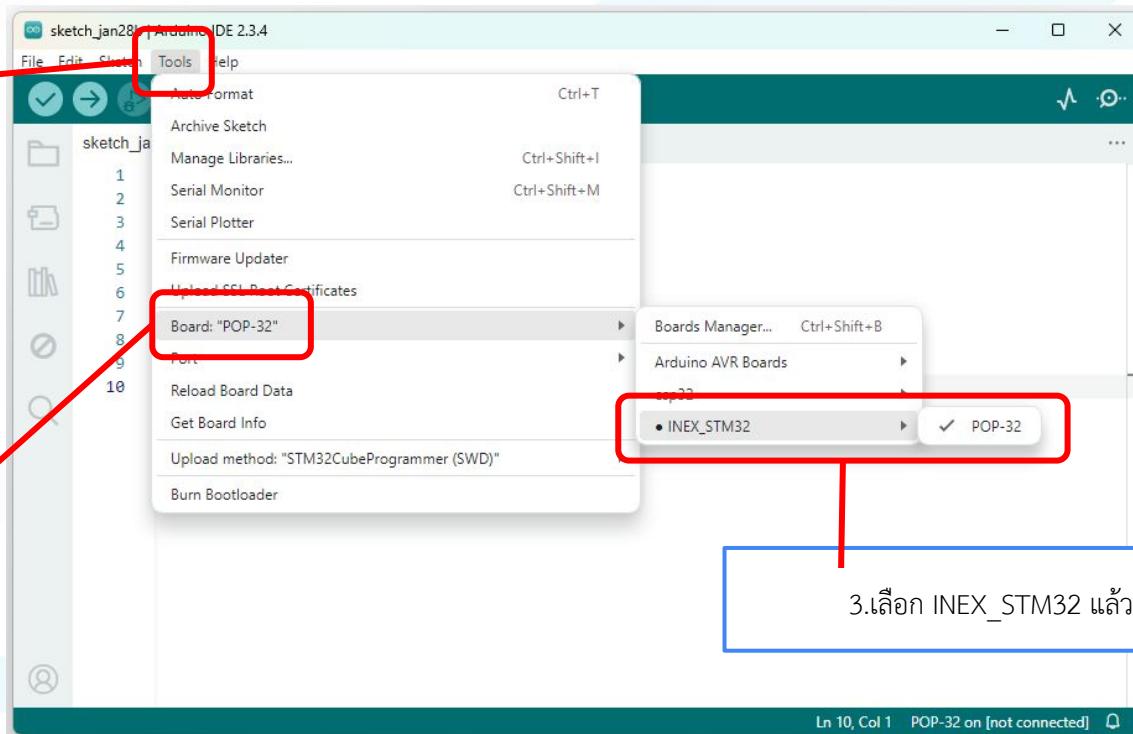
2. ต่อสาย USB เข้าที่ช่อง USB ด้านบนสุด





การเลือกบอร์ดและ Upload method ใน โปรแกรม Arduino

1.เลือกบอร์ดให้ตรงกับที่เราใช้งาน โดยเลือก Tools ตรงเมนูบาร์



2.เลือก Board ที่เมนู

3.เลือก INEX_STM32 แล้วเลือก POP-32



sketch_jan28b | Arduino IDE 2.3.4

File Edit Sketch Tools Help

POP-32

sketch_jan28b.ino

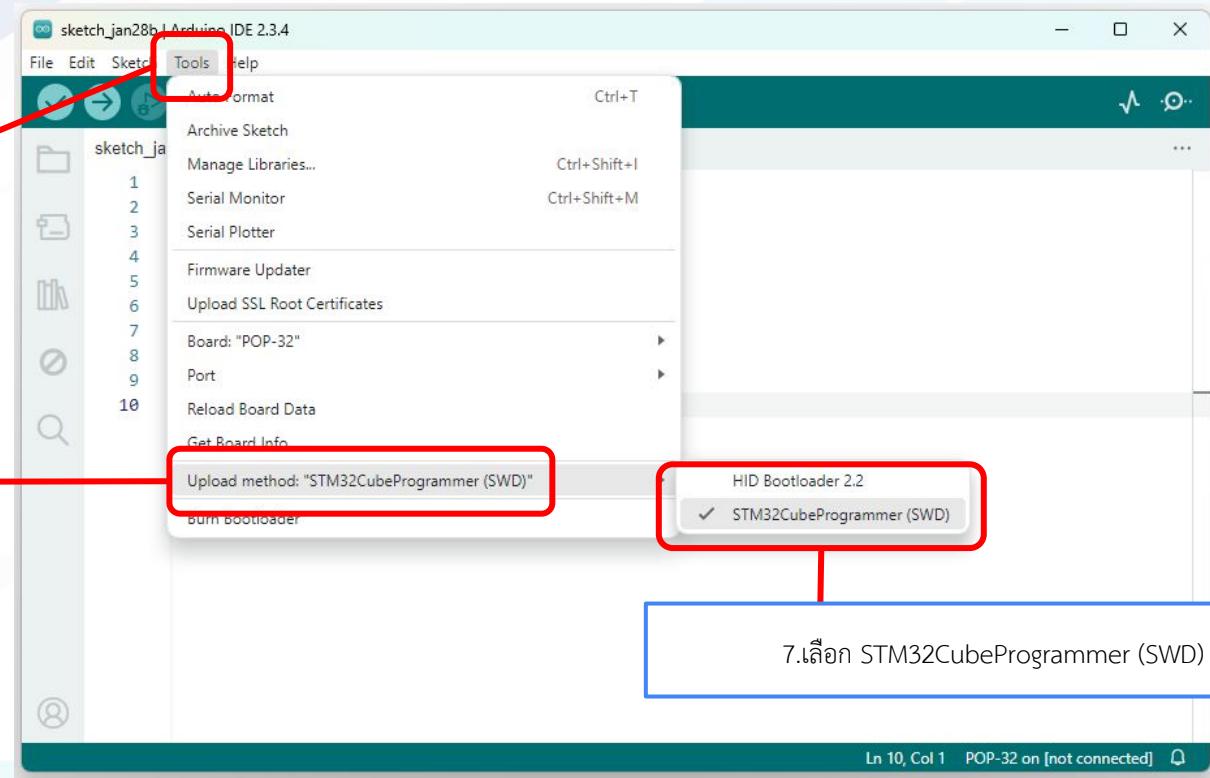
```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8 }  
9  
10
```

4.สั่งเกตที่ตำแหน่งตั้งกล่าวจะแสดง POP-32

Ln 10, Col 1 POP-32 on [not connected]



5. กำหนดวิธีการอัปโหลดโปรแกรม
โดยเลือก Tools ตรงเมนูบาร์



6. เลือก Upload method ที่เมนู

7. เลือก STM32CubeProgrammer (SWD)



ทดสอบอัปโหลดโปรแกรมลงบอร์ด POP-32i

2. กดปุ่มอัปโหลดโปรแกรม

1. พิมพ์โค้ดคำสั่งลงในช่องนี้

```
#include <POP32.h>
void setup()
{
    oled.text(0,0,"Hello POP-32");
    oled.show();
}
void loop()
{ }
```

POP32i_test | Arduino IDE 2.3.4

File Edit Sketch Tools Help

POP-32

POP32i test.ino

```
1 #include <POP32.h>
2 void setup()
3 {
4     oled.text(0,0,"Hello POP-32");
5     oled.show();
6 }
7 void loop()
8 { }
```

Output

Compiling sketch...

CANCEL

In 1, Col 1 POP-32 on [not connected] 4 1

3. หน้าต่างแสดงสถานะกำลังอัปโหลด



POP32i_test | Arduino IDE 2.3.4

File Edit Sketch Tools Help

POP-32

POP32i_test.ino

```
1 #include <POP32.h>
2 void setup()
3 {
4 oled.text(0,0,"Hello POP-32");
5 oled.show();
6 }
7 void loop()
8 {}
```

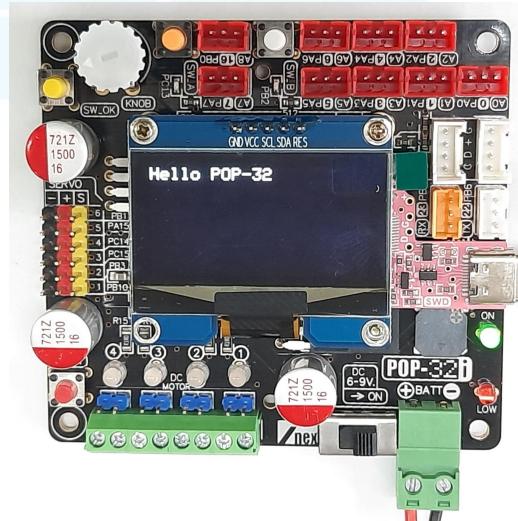
Output

```
File download complete
Time elapsed during download operation: 00:00:03.153

RUNNING Program ...
| Address: : 0x8000000
Application is running, Please Hold on...
Start operation achieved successfully
```

Ln 9, Col 1 POP-32 on [not connected] 2

4. หน้าต่างแสดงสถานะ เมื่ออัปโหลดเสร็จเรียบร้อยแล้ว



5. รูปแสดงการทำงาน เมื่ออัปโหลดเสร็จเรียบร้อยแล้ว บอร์ด POP-32i จะแสดง
ข้อความ Hello POP-32 ที่จอแสดงผล OLED



รูปแบบการทำงานของโค้ดควบคุม (Sketch) บนโปรแกรม Arduino

sketch_jan28a.ino

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8 }  
9  
10
```

การเขียนโค้ดควบคุม สามารถแบ่งออกเป็น 2 ส่วน คือ

1. **void setup()** จะเริ่มทำงานเพียงครั้งแรกครั้งเดียว เพื่อใช้ในการกำหนดค่าเริ่มต้นการทำงาน เช่น กำหนดการทำงานของขาต่างๆ

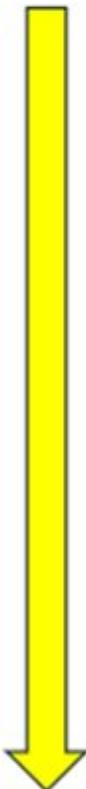
2. **void loop()** เป็นส่วนกำหนดการทำงาน ซึ่งจะทำงานวนซ้ำไปตลอดเวลา เช่น การอ่านค่าอินพุต ประมวลผล สั่งงานเอาต์พุต



รูปแสดงลำดับการทำงานของโค้ดควบคุม

```
void setup()
{
    command1;
    command2;
    command3;
}

void loop()
{
    command4;
    command5;
    command6;
}
```



การประมวลผลของโค้ดควบคุม จะทำงานดังนี้

command1;
command2;
command3;

รอบเดียวสำหรับ setup

command4;
command5;
command6;

รอบที่ 1 สำหรับ loop

command7;
command8;
command9;

รอบที่ 2 สำหรับ loop

:
:

รอบที่ n สำหรับ loop



ค่าตัวแปรต่างๆ ที่ใช้งานบ่อย

byte 0 ถึง 255 (unsigned char)

bool 0 และ 1 (true และ false)

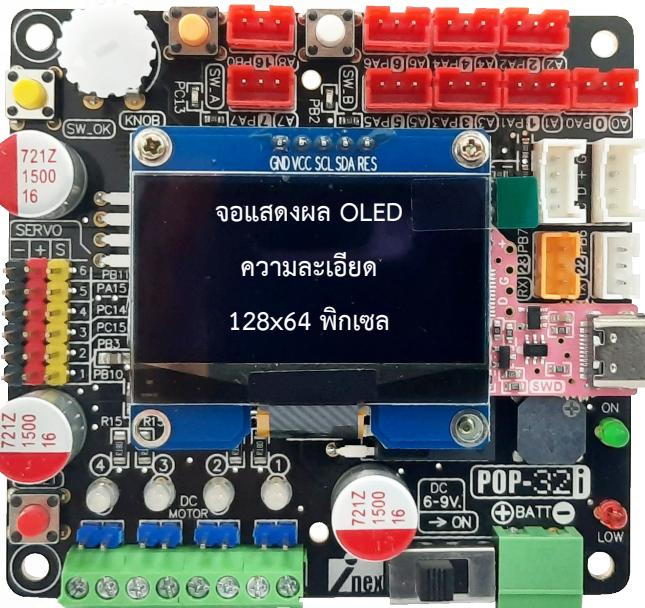
int -2147483648 ถึง 2147483647

char -128 ถึง 127

float -3.4×10^{38} ถึง 3.4×10^{38}



คำสั่งแสดงผลบนหน้าจอ OLED



ชุดคำสั่งสำหรับควบคุมการแสดงผล

คลาส OLED_I2C_SSD1309 เป็นคลาสที่รวมรวมเมธอดต่างๆ สำหรับควบคุมการแสดงผลกราฟฟิกแบบ OLED ความละเอียด 128x64 พิกเซลอย่างไรก็ได เพื่อความสะดวกในการควบคุมการแสดงผลกราฟฟิกภายในคลาส OLED_I2C_SSD1309 ได้ทำการประกาศสร้างออบเจกต์ที่ทำการสืบทอดมาจากคลาส OLED_I2C_SSD1309 โดยตั้งชื่อเป็น oled เป็นที่เรียบร้อยแล้ว



คำสั่ง oled.text

oled.text (x, y, *p, ...);

ทำหน้าที่แสดงข้อความที่จะแสดงผล OLED ซึ่งแสดงได้ 21 ตัว 8 บรรทัด (textSize = 1)

พารามิเตอร์

x คือ ตำแหน่งบรรทัด มีค่าตั้งแต่ 0-7

y คือ ตำแหน่งตัวอักษร มีค่าตั้งแต่ 0-20

*p คือ ข้อความที่ต้องการนำมาแสดง

ค่าพิเศษ

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

%h แสดงตัวเลขฐานสิบหก

%b แสดงตัวเลขฐานสอง

%f แสดงผลตัวเลขจำนวนจริง (แสดงทศนิยม 3 หลัก)



คำสั่งสำหรับควบคุมการแสดงผล

คำสั่ง oled.show

oled.show();

ทำหน้าที่อัพเดทการแสดงผลของหน้าจอแสดงผล OLED

พารามิเตอร์

ไม่มี

การคืนค่า

ไม่มี



คำสั่งอื่นๆ สำหรับใช้งานอ็อกเจ็กต์ oled

คำสั่งเกี่ยวกับการแสดงผลตัวอักษร

- **text**
- **textSize**
- **mode**
- **textColor**
- **textBackgroundColor**
- **clear**
- **fillScreen**

คำสั่งเกี่ยวกับการแสดงกราฟิก

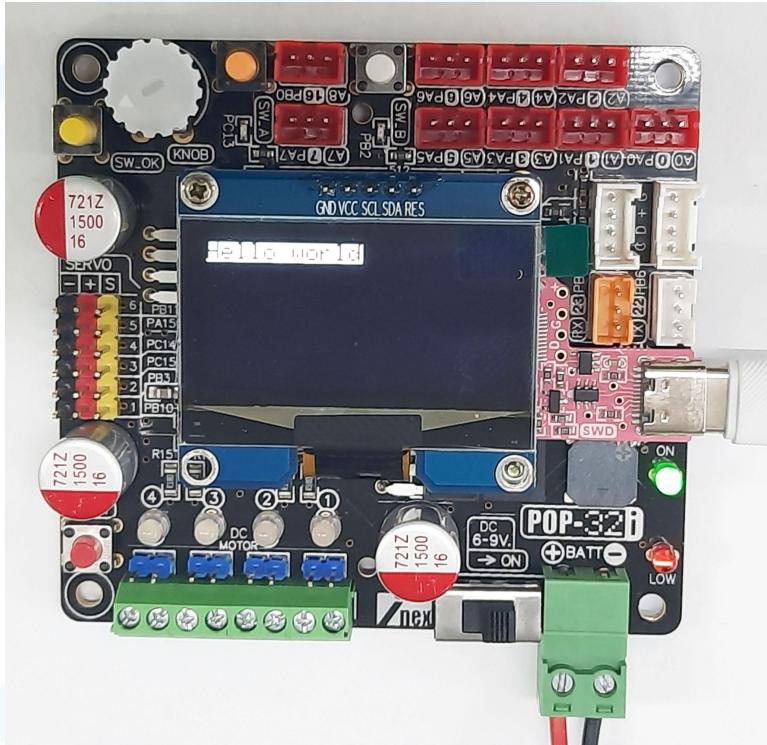
- **drawPixel**
- **drawRect**
- **fillRect**
- **drawLine**
- **drawCircle**
- **fillCircle**



ตัวอย่างการกำหนดสีตัวอักษร

```
#include <POP32.h>
void setup()
{
    oled.setTextColor(BLACK,WHITE);
    oled.text(0,0,"Hello world");
    oled.show();
}
void loop()
{}
```

ตัวอักษรสีดำขนาด 1 เท่า บนพื้นหลังสีขาว

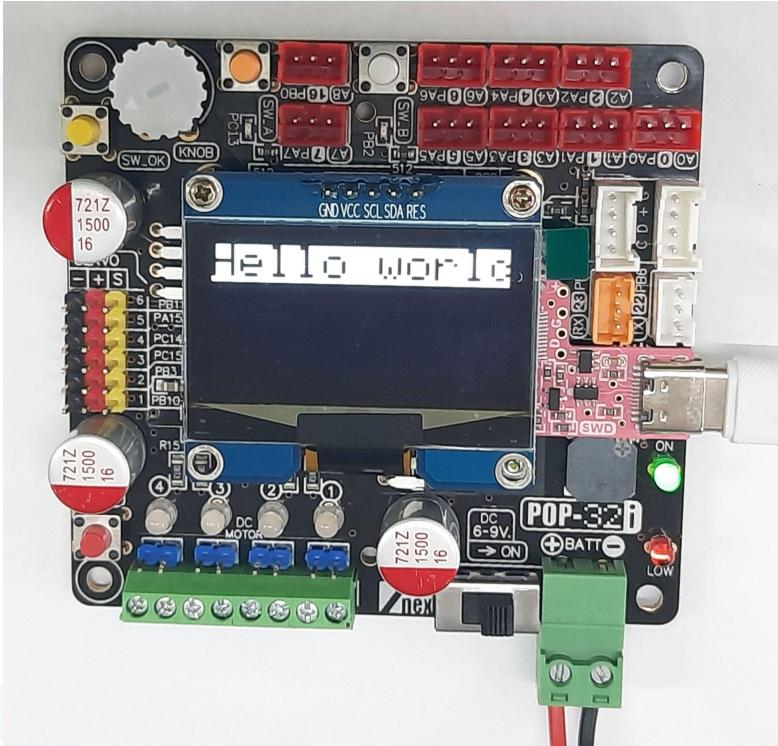




ตัวอย่างการกำหนดขนาดตัวอักษร

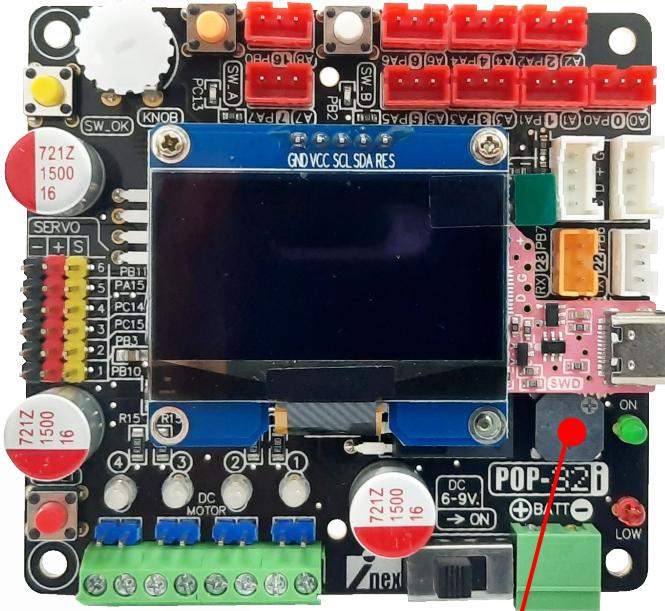
```
#include <POP32.h>
void setup()
{
    oled.setTextSize(2);
    oled.setTextColor(BLACK,WHITE);
    oled.text(0,0,"Hello world");
    oled.show();
}
void loop()
{}
```

ตัวอักษรsizeกำหนด 2 เท่า บนพื้นหลังสีขาว





คำสั่งสร้างสัญญาณเสียงออกลำโพงเปียโซ



ลำโพงเปียโซ

คำสั่ง beep

`beep();`

ทำหน้าที่สร้างสัญญาณเสียงบีฟออกทางลำโพง

คำสั่ง sound

`sound(freq, time);`

ทำหน้าที่สร้างสัญญาณเสียงตามความถี่ที่กำหนดออกทางลำโพง

พารามิเตอร์

freq คือ ความถี่เสียง กำหนดได้ตั้งแต่ 300Hz ถึง 3000Hz

time คือ ค่าเวลา มีหน่วยเป็นมิลลิวินาที



คำสั่ง delay

delay (DelayTime);

ทำหน้าที่หน่วงเวลาการทำงานของคำสั่งก่อนหน้านี้ โดยมีหน่วยเป็นมิลลิวินาที

พารามิเตอร์

DelayTime คือ ค่าเวลาที่หน่วง ในหน่วยวินาที

```
 เช่น delay(500); // หน่วงเวลา 500 มิลลิวินาที (0.5 วินาที)
```

```
 delay(2000); // หน่วงเวลา 2000 มิลลิวินาที (2 วินาที)
```



ตัวอย่างโค้ดควบคุมการสร้างเสียงออกลำโพง

ตัวอย่างการใช้งานคำสั่ง **beep**

```
#include <POP32.h>
void setup()
{}
void loop()
{
    beep();
    delay(500);
}
```

สร้างเสียง 500Hz นาน 0.5 วินาที ต่อเนื่อง

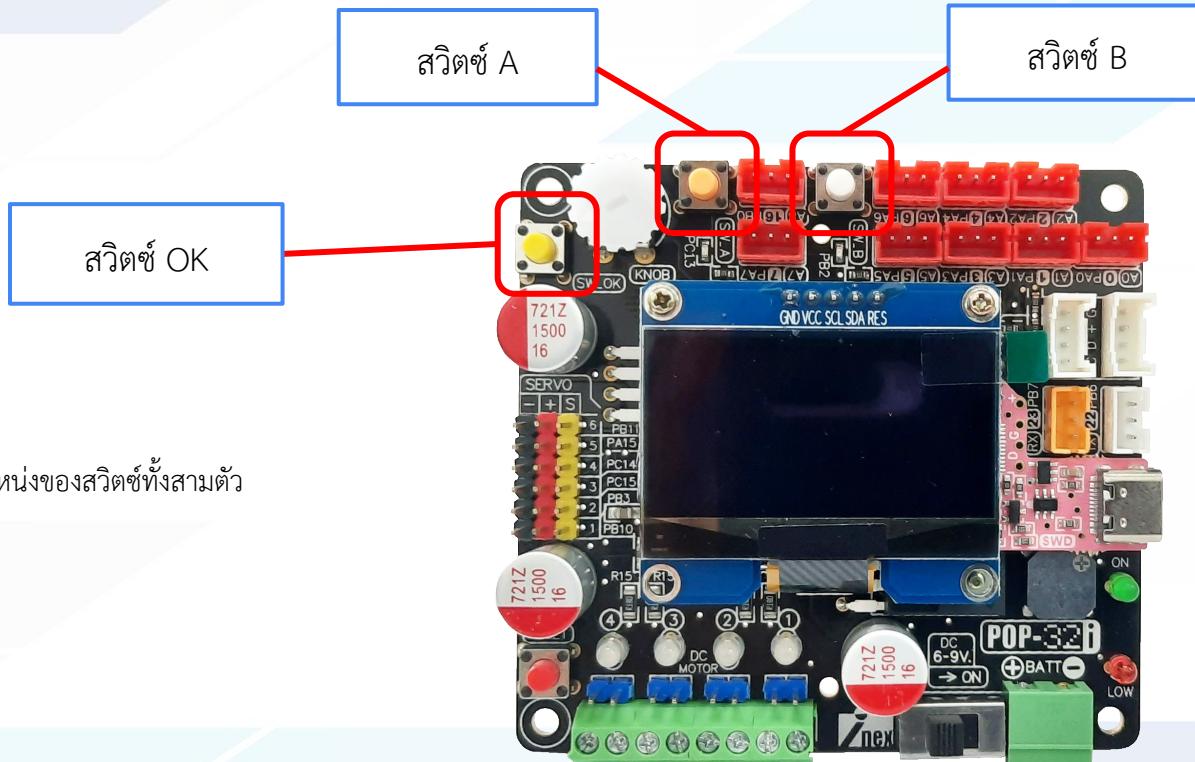
ตัวอย่างการใช้งานคำสั่ง **sound**

```
#include <POP32.h>
void setup()
{}
void loop()
{
    sound(1000,100);
    delay(500);
}
```

สร้างเสียง 1000Hz นาน 0.1 วินาที ต่อเนื่อง



คำสั่งอ่านค่าสถานะสวิตซ์ A, B และ OK



รูปแสดงตำแหน่งของสวิตซ์ทั้งสามตัว



คำสั่งที่เกี่ยวข้องกับสวิตซ์

คำสั่งอ่านค่าสถานะของสวิตซ์

SW_A();

SW_B();

SW_OK();

ทำหน้าที่อ่านค่าสถานะการกดสวิตซ์ตัวนั้นๆ

การคืนค่า

- คืนค่าเป็น true(1) ในขณะที่สวิตซ์ถูกกด
- คืนค่าเป็น false(0) ในขณะที่สวิตซ์ไม่ถูกกด

คำสั่งรอการกดสวิตซ์

waitSW_A();

waitSW_B();

waitSW_OK();

waitAnykey();

ทำหน้าที่รอการกดสวิตซ์ที่กำหนด โดย LED ประจำตัวสวิตซ์จะกระพริบต่อเนื่อง เพื่อแสดงการรอการกดสวิตซ์

การคืนค่า

ไม่มี



คำสั่งตรวจสอบเงื่อนไข if-else

```
if (เงื่อนไข) {  
    คำสั่ง 1 ทำเมื่อเงื่อนไขเป็นจริง;  
    ...  
}  
  
else {  
    คำสั่ง 2 ทำเมื่อเงื่อนไขเป็นเท็จ;  
    ...  
}
```

คำสั่ง **if** เป็นคำสั่งพื้นฐานสำหรับควบคุมการทำงานของโปรแกรม เพื่อให้ทำงานตามเงื่อนไขที่กำหนดเอาไว้ หรือเป็นการสร้างทางเลือกในการทำงานให้กับโปรแกรม

โดยถ้าดูจากรูปแบบของเงื่อนไข การทำงานจะเริ่มจากการตรวจสอบเงื่อนไขที่เรากำหนดเอาไว้ ถ้าเงื่อนไขเป็นจริงหรือถูกต้องตามเงื่อนไข ก็จะทำงานในคำสั่งที่ 1 แต่ถ้าเงื่อนไขไม่เป็นจริงหรือไม่ถูกต้อง ก็จะกระโดดไปทำงานในคำสั่งที่ 2



ตัวอย่างโค้ดควบคุมการกดสวิตช์ทีละตัว

ตรวจจับการกดสวิตช์ A

```
#include <POP32.h>
void setup()
{}
void loop()
{
    if(SW_A())
    {
        beep();
        delay(200);
    }else{
        sound(2000,200);
        delay(200);
    }
}
```

ถ้ามีการกดสวิตช์ A จะทำการส่งเสียง 500Hz ต่อเนื่อง

แต่ถ้าไม่มีการกดสวิตช์ A จะทำการส่งเสียง 2000Hz เป็นเวลา 0.2 วินาที ต่อเนื่อง

ตรวจจับการกดสวิตช์ A และ B

```
#include <POP32.h>
void setup()
{}
void loop()
{
    if(SW_A())
    {
        beep();
        delay(200);
    }
    if(SW_B())
    {
        sound(2000,200);
        delay(200);
    }
}
```

ถ้ามีการกดสวิตช์ A จะทำการส่งเสียง 500Hz ต่อเนื่อง

แต่ถ้ากดสวิตช์ B จะทำการส่งเสียง 2000Hz เป็นเวลา 0.2 วินาที ต่อเนื่อง



ตัวอย่างโค้ดควบคุมการกดสวิตซ์ร่วมกัน

ตัวอย่างการใช้เงื่อนไข && (AND)

```
#include <POP32.h>
void setup()
{}
void loop()
{
    if(SW_A() && SW_B())
    {
        beep();
        delay(200);
    }else{
        sound(2000,200);
        delay(200);
    }
}
```

ถ้ามีการกดสวิตซ์ A และ B
พร้อมกัน จะทำการส่งเสียง
500Hz ต่อเนื่อง

แต่ถ้าไม่มีการกดสวิตซ์ A และสวิตซ์ B พร้อมกัน จะทำการส่งเสียง
2000Hz เป็นเวลา 0.2 วินาที ต่อเนื่อง

ตัวอย่างการใช้เงื่อนไข || (OR)

```
#include <POP32.h>
void setup()
{}
void loop()
{
    if(SW_A() || SW_B())
    {
        beep();
        delay(200);
    }else{
        sound(2000,200);
        delay(200);
    }
}
```

ถ้ามีการกดสวิตซ์ A หรือ B จะทำการส่งเสียง 500Hz ต่อเนื่อง

แต่ถ้าไม่มีการกดสวิตซ์ A หรือสวิตซ์ B จะทำการส่งเสียง 2000Hz เป็นเวลา 0.2 วินาที ต่อเนื่อง



ตัวอย่างโค้ดควบคุมรอการกดสวิตช์

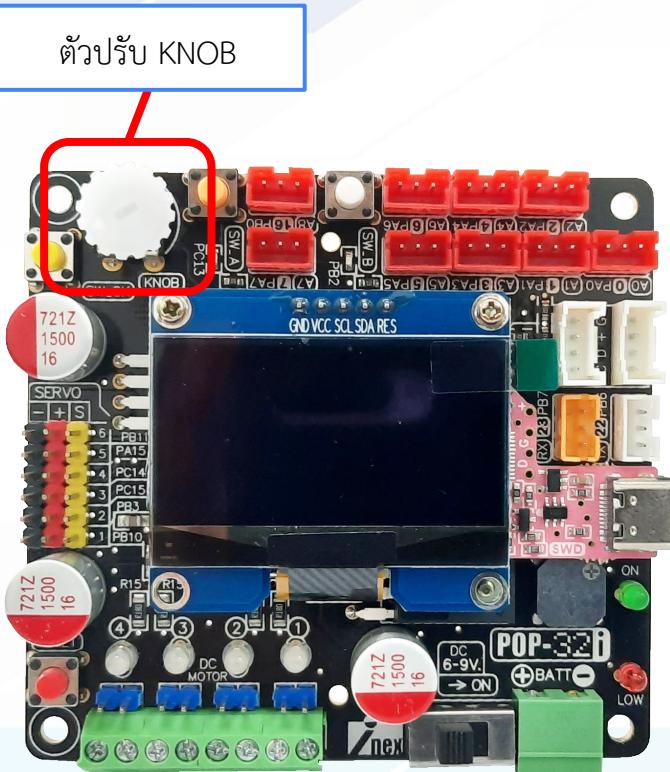
```
#include <POP32.h>
void setup()
{
    oled.text(0,0,"Press SW_OK");
    oled.show();
    waitSW_OK();
}
void loop()
{
    beep();
    delay(1000);
}
```

เมื่อเริ่มโปรแกรมหน้าจอ OLED จะแสดง Press SW_OK ที่
ตำแหน่ง X=0 และ Y=0

จากนั้นจะรอการกดสวิตช์ OK เมื่อมีการกดสวิตช์ OK เมื่อไร ก็จะ
ทำการส่งเสียง 500Hz ต่อเนื่อง



คำสั่งอ่านค่า KNOB



คำสั่ง knob

knob();

ทำหน้าที่อ่านค่าอ่อนนาลอกจากตัวปรับ knob โดยจะมีค่าตั้งแต่ 0 ถึง 4000

การคืนค่า

คืนค่าตั้งแต่ 0 ถึง 4000 จากการปรับค่าจากตัวปรับ knob



คำสั่ง knob(x)

knob(x);

ทำหน้าที่อ่านค่าอ่อนโยนจากตัวปรับ knob โดยจะมีค่าตั้งแต่ 0 ถึง x ที่กำหนด

พารามิเตอร์

x คือ ค่าสูงสุดที่สามารถปรับได้

การคืนค่า

คืนค่าตั้งแต่ 0 ถึง x จากการปรับค่าจากตัวปรับ knob

คำสั่ง knob(x,y)

knob(x,y);

ทำหน้าที่อ่านค่าอ่อนโยนจากตัวปรับ knob โดยจะปรับในช่วง x ถึง y ที่กำหนด

พารามิเตอร์

x คือ ค่าต่ำสุดที่สามารถปรับได้

y คือ ค่าสูงสุดที่สามารถปรับได้

การคืนค่า

คืนค่าตั้งแต่ 0 ถึง x จากการปรับค่าจากตัวปรับ knob



ตัวอย่างโค้ดควบคุมอ่านค่าจากตัวปรับ knob

```
#include <POP32.h>
void setup()
{}
void loop()
{
    oled.text(0,0,"knob=%d ",knob());
    oled.text();
}
```

แสดงค่าที่อ่านได้จากตัวปรับ knob ออกทางหน้าจอ OLED
ทดลองปรับตัวปรับ knob ตัวเลขจะเปลี่ยนแปลงตามการ
ปรับของเรา



การสร้างฟังก์ชันในโค้ดควบคุม

```
void Fb(int spd=100)
{
    // ชุดคำสั่ง
    speed = 100*(spd/100);
    ...
}
```

ชื่อฟังก์ชัน

ตัวแปรที่ส่งไปยัง
ฟังก์ชัน

รูปแบบการใช้งานฟังก์ชัน

Fb(100);

```
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

รูปแบบการใช้งานฟังก์ชัน

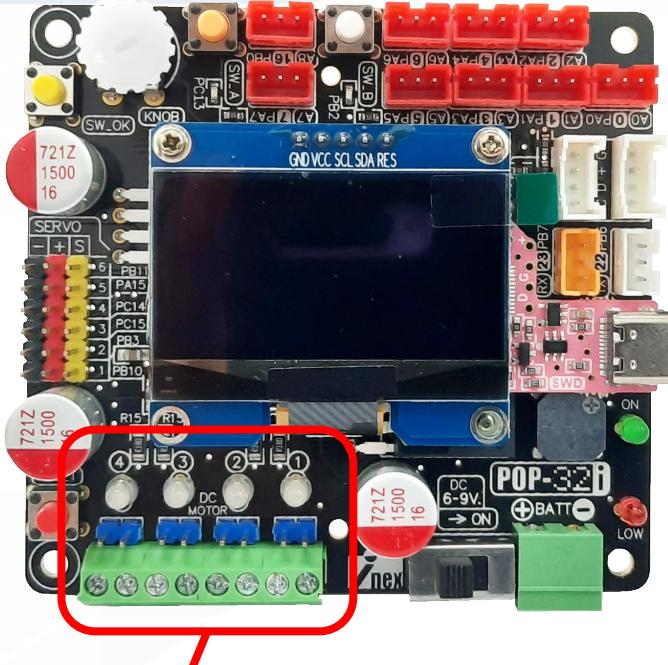
Fb(100);

ฟังก์ชัน เป็นส่วนของโค้ดที่ถูกเขียนขึ้น เพื่อทำงานเฉพาะในสิ่งที่เราต้องการ โดยฟังก์ชันนี้จะมีที่มา กับตัวโปรแกรมและฟังก์ชันที่ผู้เขียนเขียนขึ้นมาเอง

แนวคิดของฟังก์ชันก็คือ การเรียกใช้โค้ดที่ต้องการใช้บ่อยๆ โดยที่ไม่ต้องนำโค้ดทั้งหมดไปวาง ซึ่งทำให้โค้ดควบคุมยาวจนเกินไป จนเกิดความไม่สะดวกในการเขียนโค้ด



คำสั่งโค้ดควบคุมขับเคลื่อนมอเตอร์ไฟต์รัง



จุดต่อมอเตอร์ 1-4

คำสั่ง motor(ch,pow)

motor(ch, pow);

ทำหน้าที่ขับเคลื่อนมอเตอร์ซองที่กำหนด

พารามิเตอร์

Ch คือ ช่องขับมอเตอร์ซองที่ 1 ถึง 4

pow คือ ค่ากำลังขับในช่วง -100 ถึง 100 กรณีเป็นค่าบวก ขับทิศทางไปข้างหน้า และกรณีเป็นค่าลบ ขับทิศทางโดยหลัง



หลักการเคลื่อนที่ของหุ่นยนต์

โดยการใช้ล้อ Omni แบบติดตั้ง 3 ล้อ





ลักษณะของล้อ Omni

ล้อ Omni เป็นล้อที่มีลักษณะพิเศษ โดยจะมีล้อที่มีขนาดเล็กๆ วางขวางซ้อนอยู่

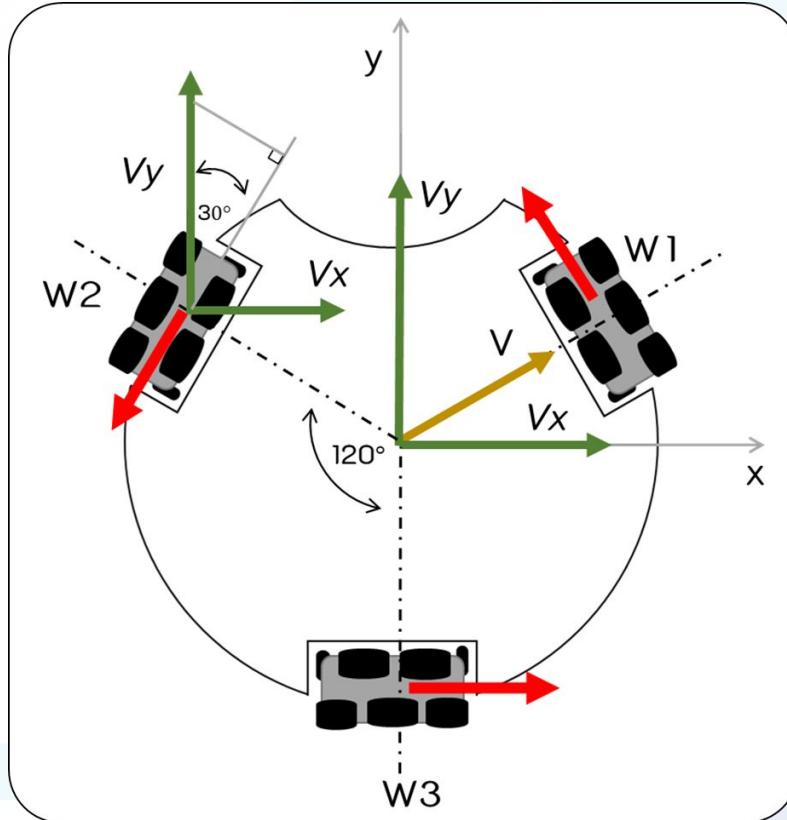
ในล้อหลักทำมุ่งตั้งฉากกับทิศทางแรงดูดของล้อ ทำให้สามารถเคลื่อนที่ในแนวตั้ง และแนวอนันได้อย่างอิสระ มีความคล่องตัวสูง





การวางแผนของล้อ Omni

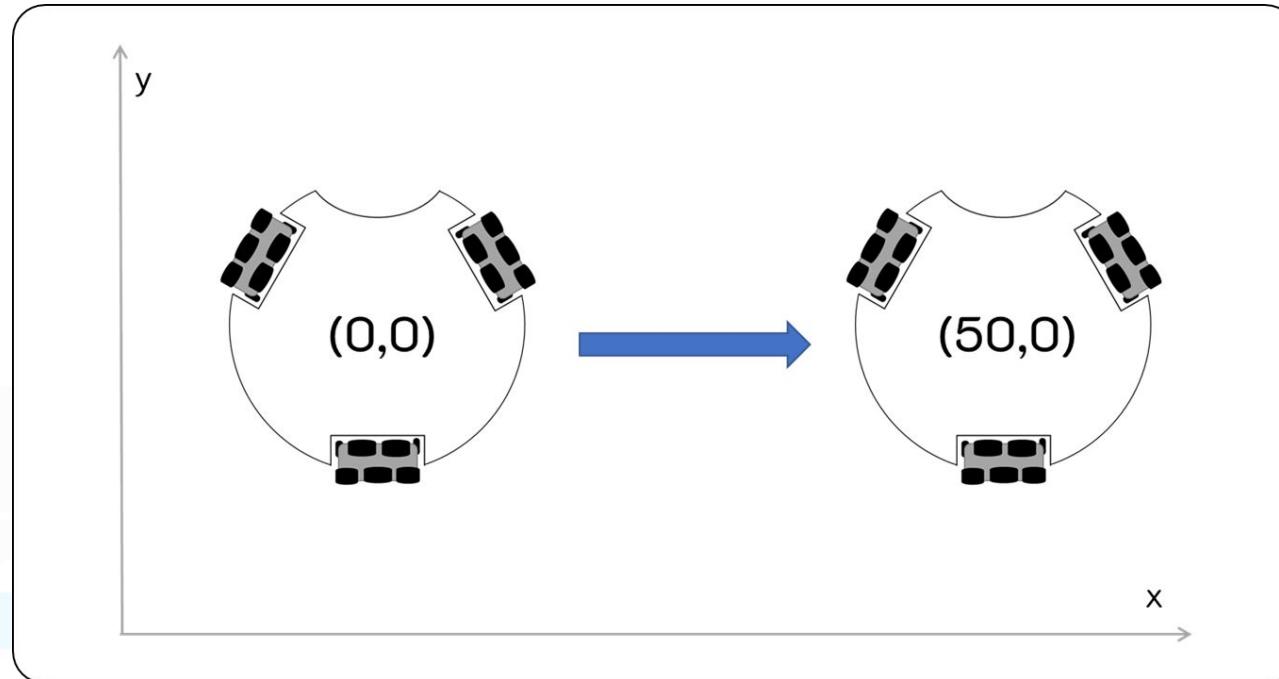
แกนหมุนในแต่ละล้อ
จะทำมุม 120° เท่าๆ กัน





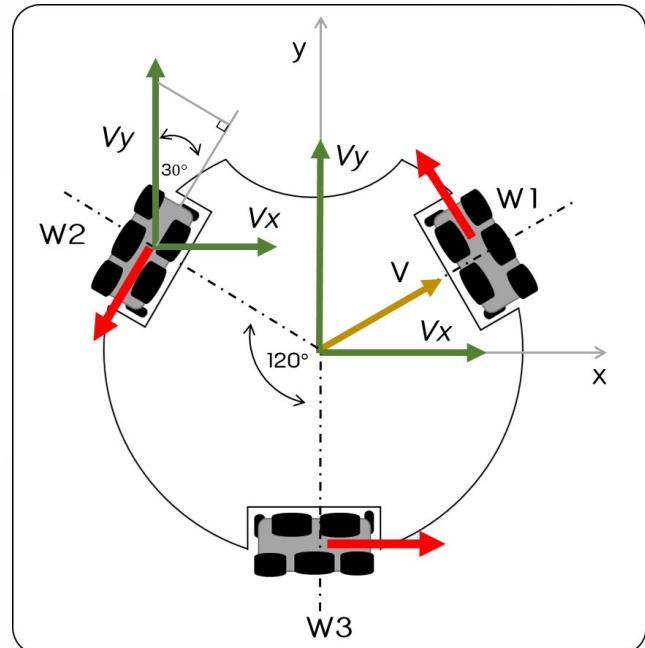
สมการการเคลื่อนที่แบบเชิงเส้น

การเคลื่อนที่แบบเชิงเส้นด้วยวิธีการนี้ คือ หุ่นยนต์จะเคลื่อนที่ไปยังจุดปลายทางโดยไม่ต้องหมุนตัว เพียงแต่เป็นการเคลื่อนที่แบบเลื่อนสไลด์ไปยังจุดปลายทางหรือทิศทางได้ตามต้องการ





สมการการเคลื่อนที่ ล้อที่ 1 (W1)



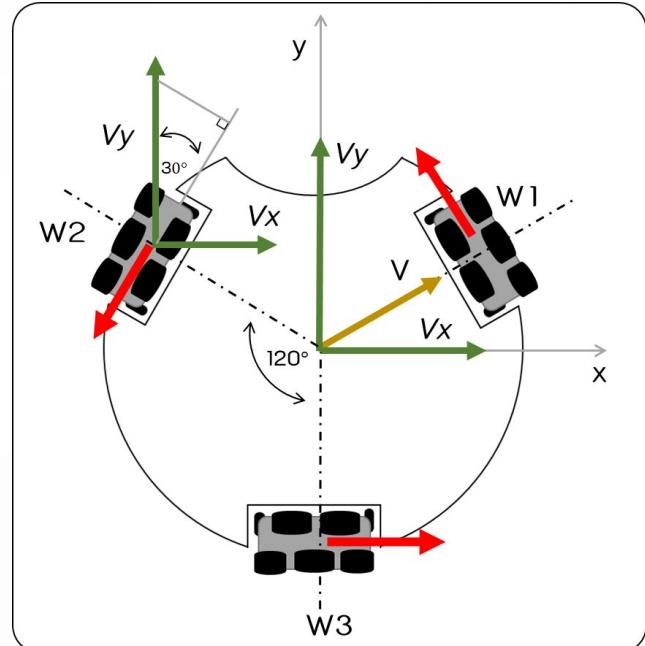
กำหนดทิศทางบวกของมอเตอร์ โดยแทนด้วย

ลูกศรสีแดง สังเกตได้ว่า ชี้ไปทางแกน x ผิ่งลบและชี้ไปทางแกน y ผิ่งบวก โดยล้อทำมุมกับแนวแกน y 30 องศา จึงได้รูปแบบสมการดังนี้

$$v_1 = v_y \cos(30^\circ) - v_x \sin(30^\circ)$$



สมการการเคลื่อนที่ ล้อที่ 2 (W2)

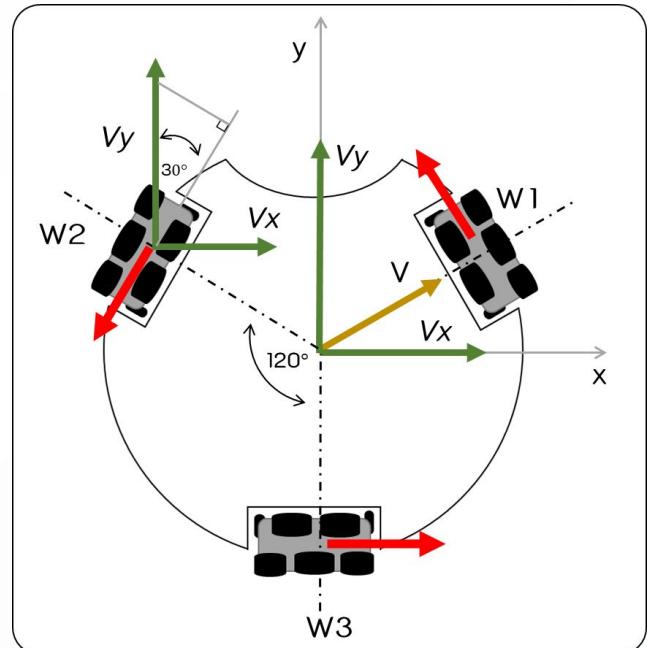


กำหนดทิศทางบวกของมอเตอร์ โดยแทนด้วยลูกศรสีแดง สังเกตได้ว่าชี้ไปทางแกน x ฝั่งลับและชี้ไปทางแกน y ฝั่งลับ โดยล้อทำมุนกับแนวแกน y 30 องศา จึงได้รูปแบบสมการดังนี้

$$v_2 = -v_y \cos(30^\circ) - v_x \sin(30^\circ)$$



สมการการเคลื่อนที่ ล้อที่ 3 (W3)



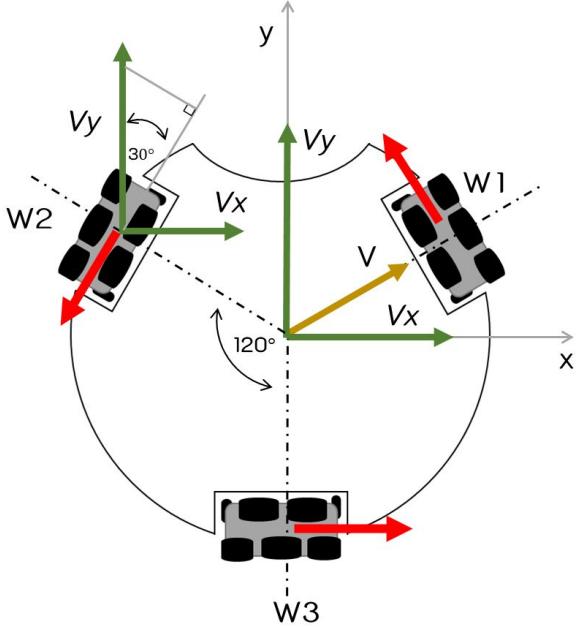
กำหนดทิศทางบวกของมอเตอร์ โดยแทนด้วยลูกศรสีแดง สังเกตได้ว่าซี่ไปทาง
แกน x ผิดปกติและไม่ได้มีแรงใดๆ กระทำในแนวแกน y โดยล้อทำมุกับแนว
แกน y 90 องศา จึงได้รูปแบบสมการดังนี้

$$v_3 = v_y \cos(90^\circ) + v_x \sin(90^\circ)$$

$$v_3 = v_x$$



สรุปสมการการเคลื่อนที่แบบเชิงเส้น



การกำหนดทิศทางหมุนของล้อ

จากรูป ลูกศรสีแดงกำหนดทิศทางบวกของมอเตอร์

ทิศทางบวก (+) หมุนตามเข็มนาฬิกา

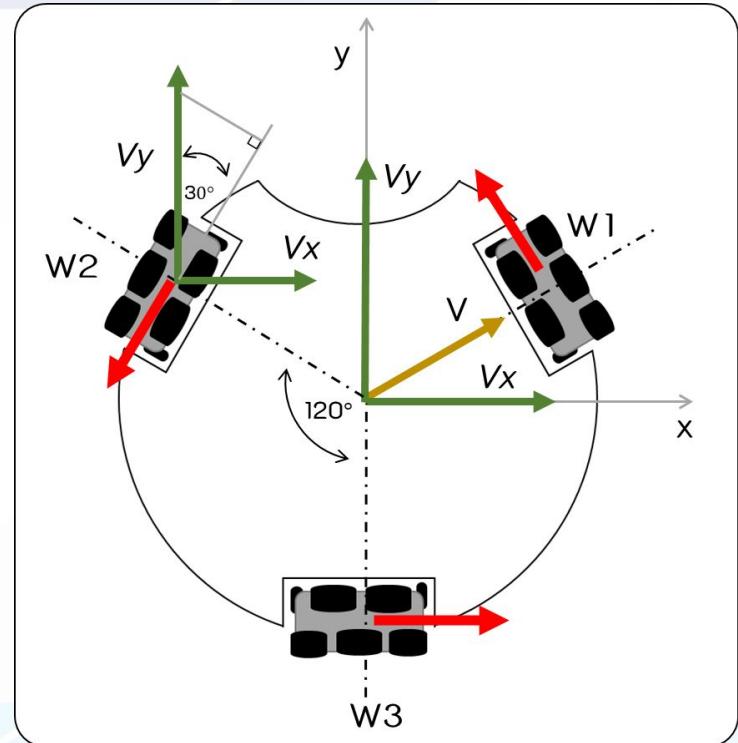
ทิศทางลบ (-) หมุนวนเข็มนาฬิกา

แสดงการหาทิศทางความเร็วของแต่ละล้อ

$$\text{ล้อที่ } 1 (\text{W1}) : v_1 = v_y \cos(30^\circ) - v_x \sin(30^\circ)$$

$$\text{ล้อที่ } 2 (\text{W2}) : v_2 = -v_y \cos(30^\circ) - v_x \sin(30^\circ)$$

$$\text{ล้อที่ } 3 (\text{W3}) : v_3 = v_x$$



ต้องการเคลื่อนที่ไปที่จุด $V_x=50$ และ $V_y=0$

วิธีทำ

ทิศทางล้อที่ 1

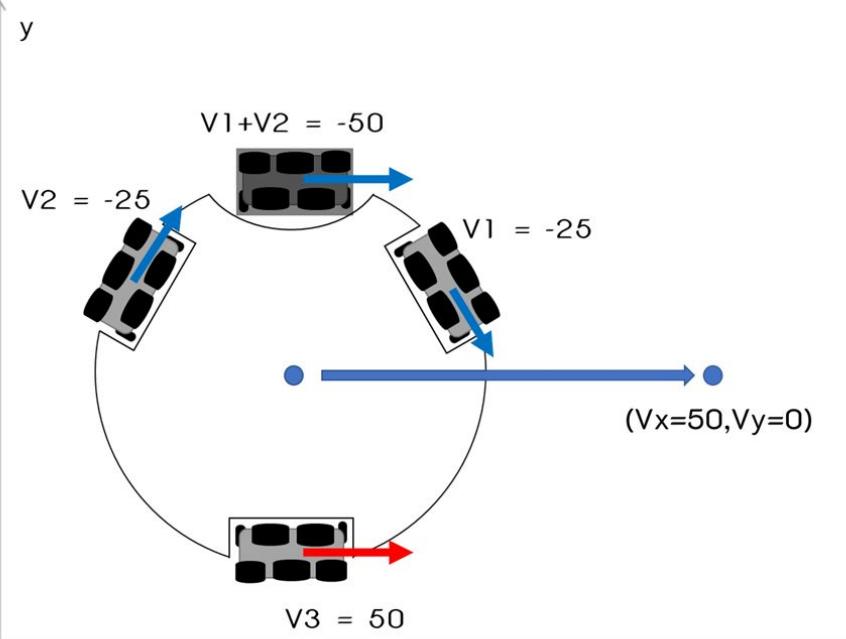
$$V1 = -50 * \cos(30) \Rightarrow -25$$

ทิศทางล้อที่ 2

$$V2 = -50 * \cos(30) \Rightarrow -25$$

ทิศทางล้อที่ 3

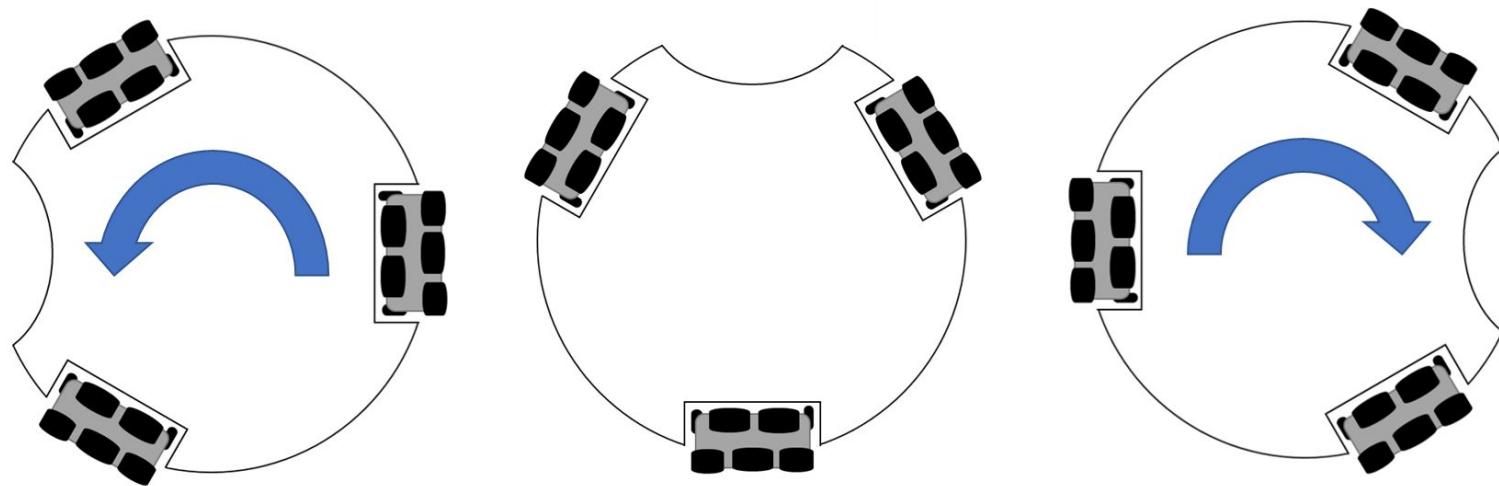
$$V3 = 50 \Rightarrow 50$$



จากผลลัพธ์จะเห็นว่า เมื่อนำ $V1$ และ $V2$ มารวมกัน จะมีค่าเท่ากับ -50 โดยจะแสดงเป็นล้อเสนียงที่ถูกติดตั้งไว้ออกผ่างของล้อที่ 3 ที่มีแรงกระทำ ค่าเท่ากับ 50 เมื่อมอเตอร์เริ่มทำงาน หุ่นยนต์จะเคลื่อนที่ไปทางด้านขวาทันที

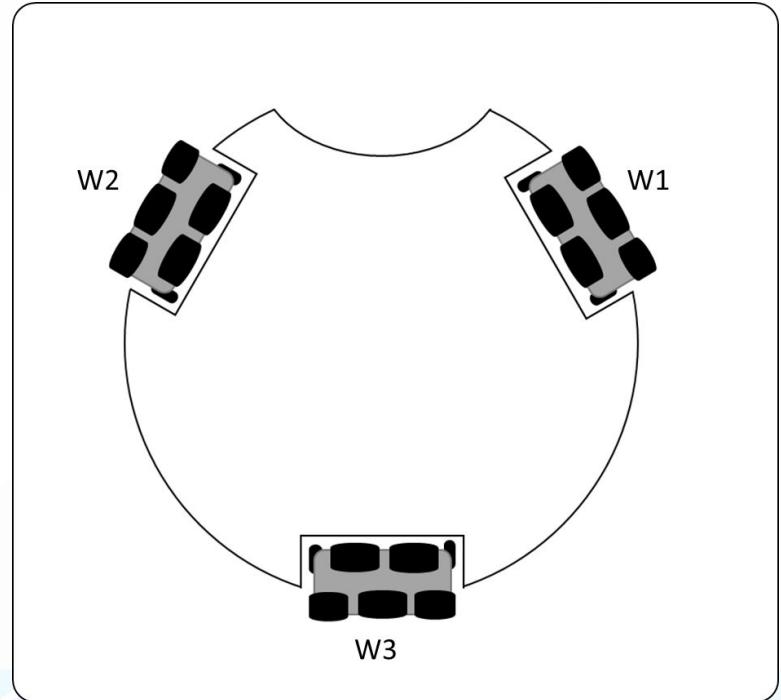


สมการการเคลื่อนที่แบบความเร็วเชิงมุ่ง





สมการความเร็วเชิงมุม



ความเร็วเชิงมุม (ω : อเมกา) จะนำมาใช้ในการหมุนตัวของหุ่นยนต์ โดยทั้ง 3 ล้อ จะถูกกำหนดความเร็วและทิศทางการหมุนเท่าๆ กัน

สมการกำหนดความเร็วเชิงมุม

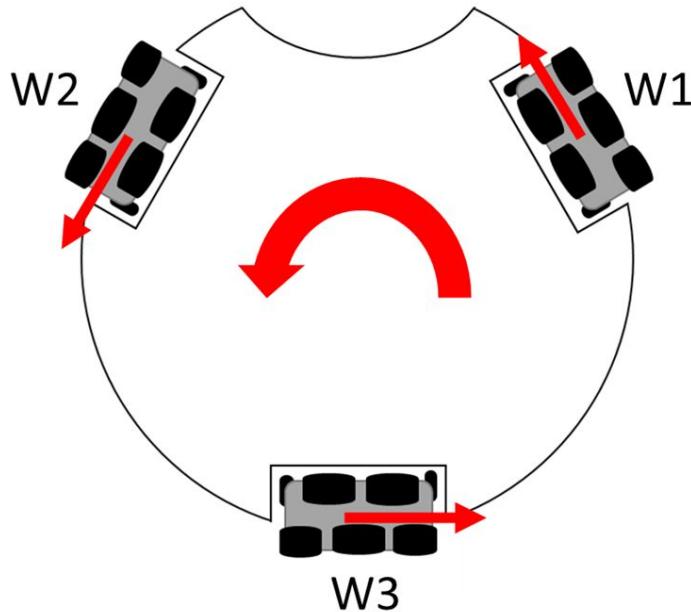
$$\text{ล้อที่ } 1 (W1) : V_1 = \omega$$

$$\text{ล้อที่ } 2 (W2) : V_2 = \omega$$

$$\text{ล้อที่ } 3 (W3) : V_3 = \omega$$



ตัวอย่าง



ต้องการหมุนตัวหุ่นยนต์ด้วยความเร็ว 50

ในทิศทางทวนเข็มนาฬิกา

สมการกำหนดความเร็วเชิงมุม

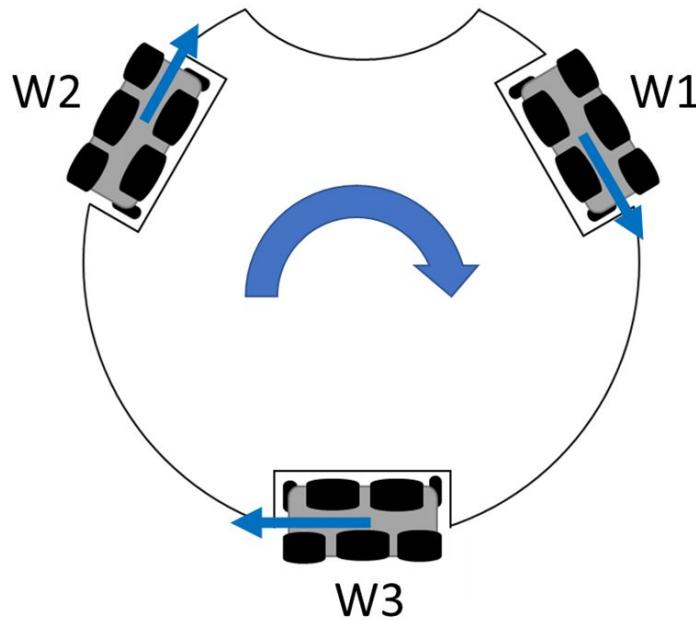
$$\text{ล้อที่ } 1 (\text{W1}) : V_1 = 50$$

$$\text{ล้อที่ } 2 (\text{W2}) : V_2 = 50$$

$$\text{ล้อที่ } 3 (\text{W3}) : V_3 = 50$$



ตัวอย่าง



ต้องการหมุนตัวหุ่นยนต์ด้วยความเร็ว 50
ในทิศทางตามเข็มนาฬิกา

สมการกำหนดความเร็วเชิงมุม

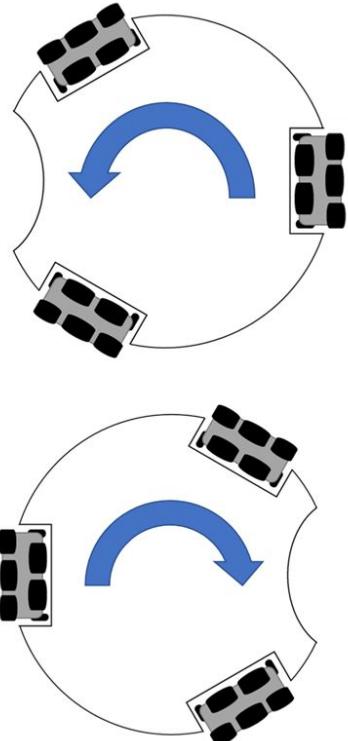
$$\text{ล้อที่ } 1 \text{ (W1)} : V_1 = -50$$

$$\text{ล้อที่ } 2 \text{ (W2)} : V_2 = -50$$

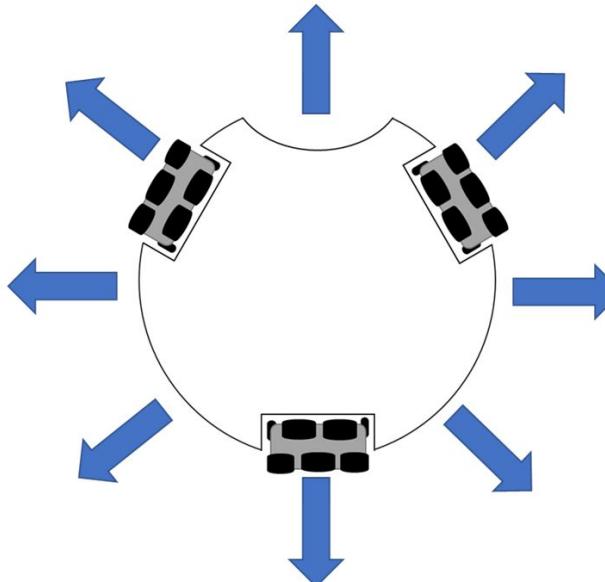
$$\text{ล้อที่ } 3 \text{ (W3)} : V_3 = -50$$



สรุปการเคลื่อนที่ทั้ง 10 รูปแบบ

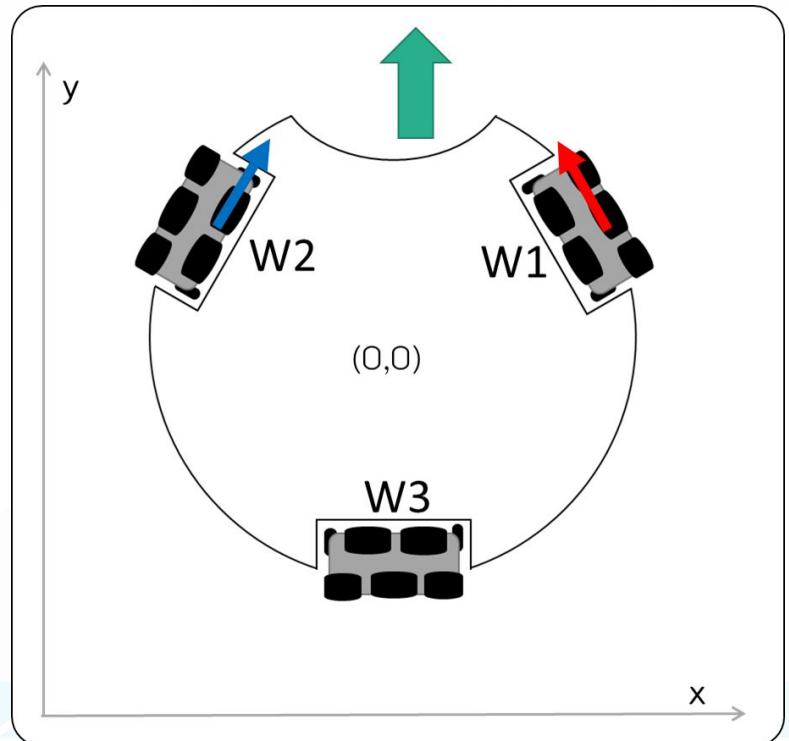


จากตัวอย่างจะใช้ค่าแรงเท่ากับ 50 มาคำนวณและหากการเคลื่อนที่เดียว ในแนวทะแยง จะใช้ค่า Vx และ Vy เท่ากัน โดยต่างกันเพียงแค่ทิศทาง





ขับเคลื่อนหุ่นยนต์ไปข้างหน้า



ต้องการเคลื่อนที่ไปที่จุด $V_x=0$ และ $V_y=50$

วิธีทำ

ทิศทางล้อที่ 1 : $V_1 = 50 * \cos(30) \Rightarrow 43.30$

ทิศทางล้อที่ 2 : $V_2 = -50 * \cos(30) \Rightarrow -43.30$

ทิศทางล้อที่ 3 : $V_3 = 0$

ตัวอย่างโปรแกรม

```
motor(1, 43);
```

```
motor(2, -43);
```

```
motor(3, 0);
```



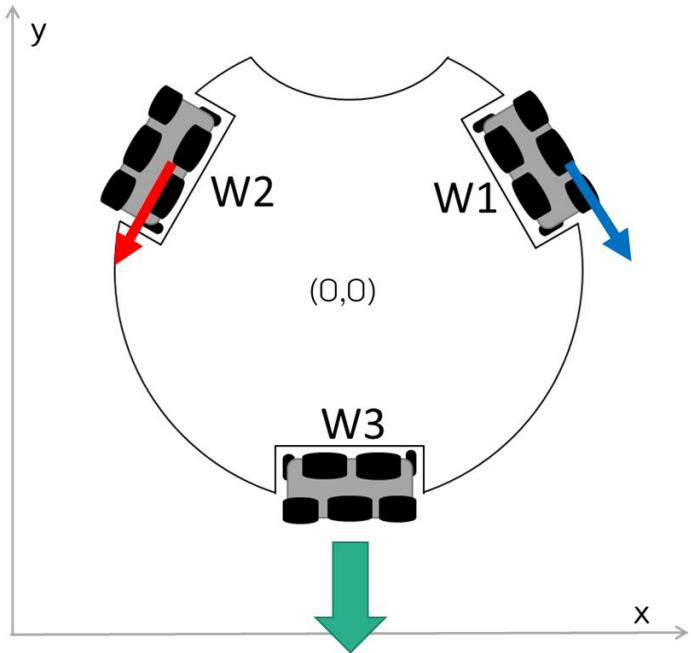
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ไปข้างหน้า

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(43, -43, 0);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code ->
Test_movement.ino



ขั้นเคลื่อนที่นยนต์ถอยหลัง



ต้องการเคลื่อนที่ไปที่จุด $Vx=0$ และ $Vy=-50$

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = -50 * \cos(30) \Rightarrow -43.30$

ทิศทางล้อที่ 2 : $V2 = 50 * \cos(30) \Rightarrow 43.30$

ทิศทางล้อที่ 3 : $V3 = 0$

ตัวอย่างโปรแกรม

```
motor(1, -43);
```

```
motor(2, 43);
```

```
motor(3, 0);
```



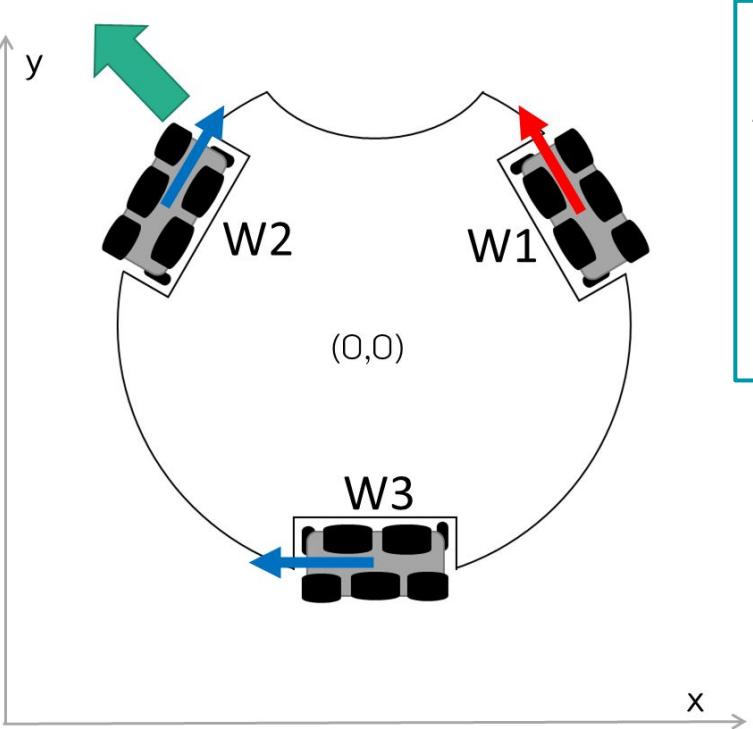
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ถอยหลัง

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(-43,43,0);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นเคลื่อนที่นุยนต์ทะแยงซ้ายไปข้างหน้า



ต้องการเคลื่อนที่ไปที่จุด $Vx=-50$ และ $Vy=50$

วิธีทำ

$$\text{ทิศทางล้อที่ 1 : } V1 = 50 * \cos(30) - (-50) * \sin(30) \Rightarrow 68.30$$

$$\text{ทิศทางล้อที่ 2 : } V2 = -(50) * \cos(30) - (-50) * \sin(30) \Rightarrow -18.30$$

$$\text{ทิศทางล้อที่ 3 : } V3 = (-50)$$

ตัวอย่างโปรแกรม

```
motor(1, 68);
```

```
motor(2, -18);
```

```
motor(3, -50);
```



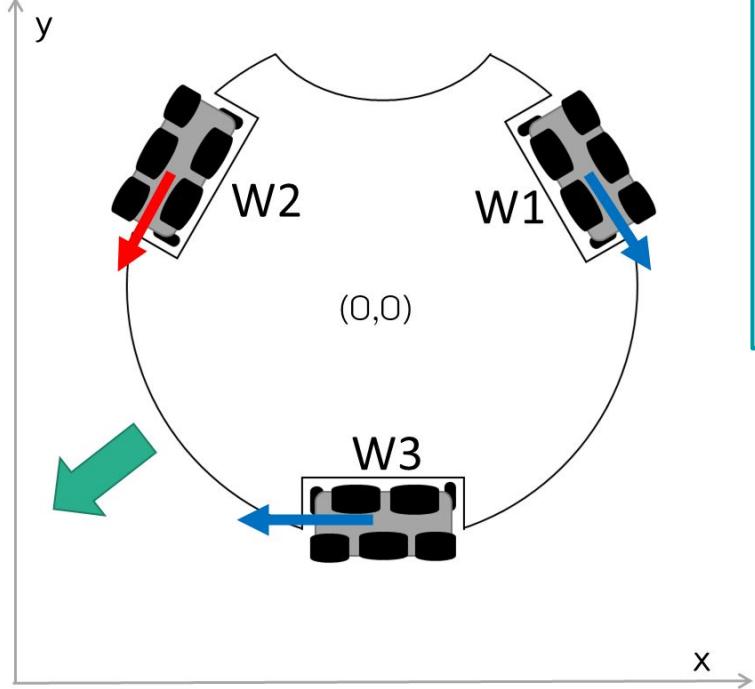
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ทะแยงซ้ายไปข้างหน้า

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(68, -18, -50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นเคลื่อนที่นุยนต์ถอยหลังทะแยงซ้าย



ต้องการเคลื่อนที่ไปที่จุด $Vx=-50$ และ $Vy=-50$

วิธีทำ

$$\text{ทิศทางล้อที่ 1 : } V1 = (-50)\cos(30) - (-50)\sin(30) \Rightarrow -18.30$$

$$\text{ทิศทางล้อที่ 2 : } V2 = -(-50)\cos(30) - (-50)\sin(30) \Rightarrow 68.30$$

$$\text{ทิศทางล้อที่ 3 : } V3 = (-50)$$

ตัวอย่างโปรแกรม

```
motor(1, -18);
```

```
motor(2, 68);
```

```
motor(3, -50);
```



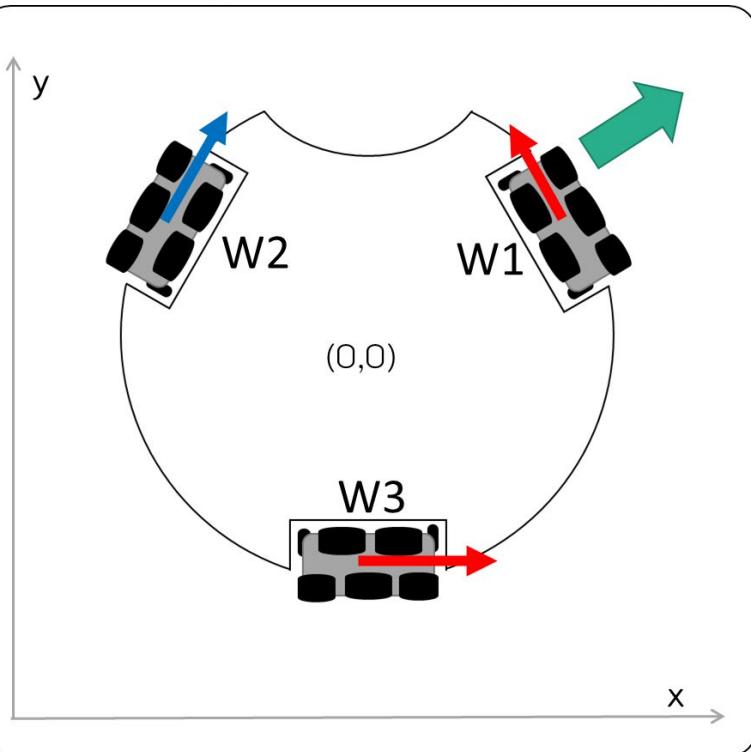
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ถอยหลังทะเบี่ยงซ้าย

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(-18,68,-50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขับเคลื่อนหุ่นยนต์ทะแยงขวาไปข้างหน้า



ต้องการเคลื่อนที่ไปที่จุด $Vx=50$ และ $Vy=50$

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = (50)\cos(30) - (50)\sin(30) \Rightarrow 18.30$

ทิศทางล้อที่ 2 : $V2 = -(50)\cos(30) - (50)\sin(30) \Rightarrow -68.30$

ทิศทางล้อที่ 3 : $V3 = 50$

ตัวอย่างโปรแกรม

```
motor(1, 18);
```

```
motor(2, -68);
```

```
motor(3, 50);
```



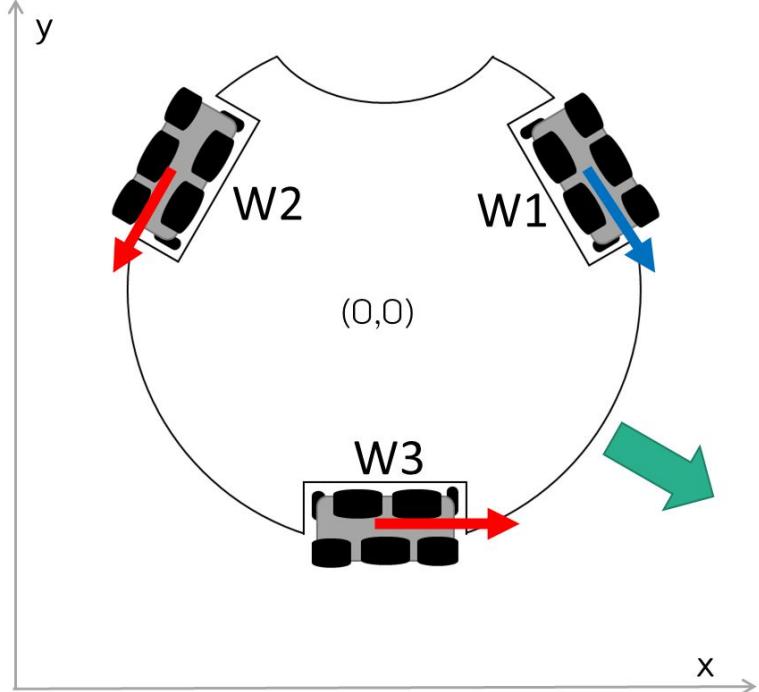
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ทะแยงขวาไปข้างหน้า

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(18, -68, 50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นเคลื่อนที่นุยนต์ถอยหลังทางขวา



ต้องการเคลื่อนที่ไปที่จุด $Vx=-50$ และ $Vy=-50$

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = (-50)\cos(30) - (50)\sin(30) \Rightarrow -68.30$

ทิศทางล้อที่ 2 : $V2 = -(-50)\cos(30) - (50)\sin(30) \Rightarrow 18.30$

ทิศทางล้อที่ 3 : $V3 = 50$

ตัวอย่างโปรแกรม

```
motor(1, -68);  
motor(2, 18);  
motor(3, 50);
```



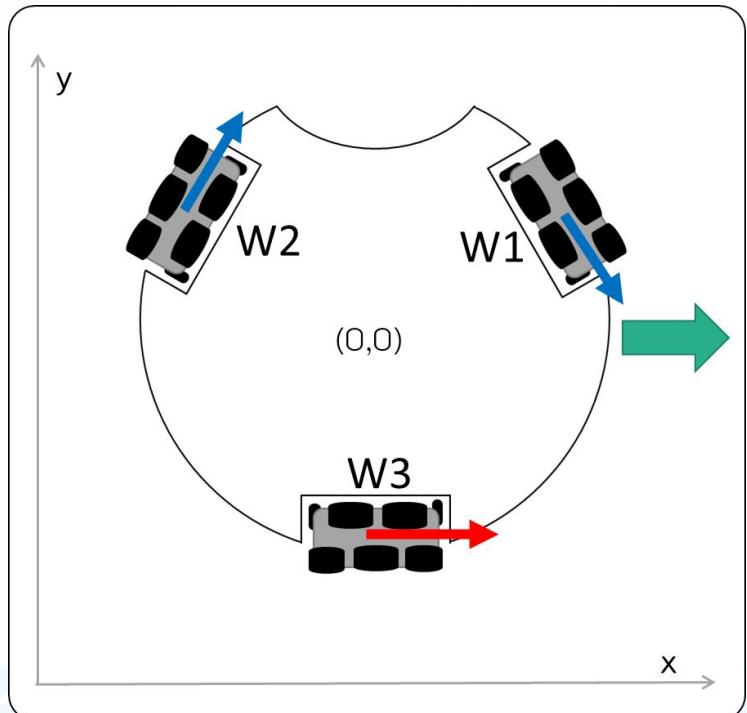
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ถอยหลังทะแยงขวา

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(-68,18,50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขับเคลื่อนหุ่นยนต์ไปทางด้านขวา



ต้องการเคลื่อนที่ไปที่จุด $Vx=50$ และ $Vy=0$

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = -50 * \sin(30) \Rightarrow -25$

ทิศทางล้อที่ 2 : $V2 = -50 * \sin(30) \Rightarrow -25$

ทิศทางล้อที่ 3 : $V3 = 50$

ตัวอย่างโปรแกรม

```
motor(1, -25);
```

```
motor(2, -25);
```

```
motor(3, 50);
```



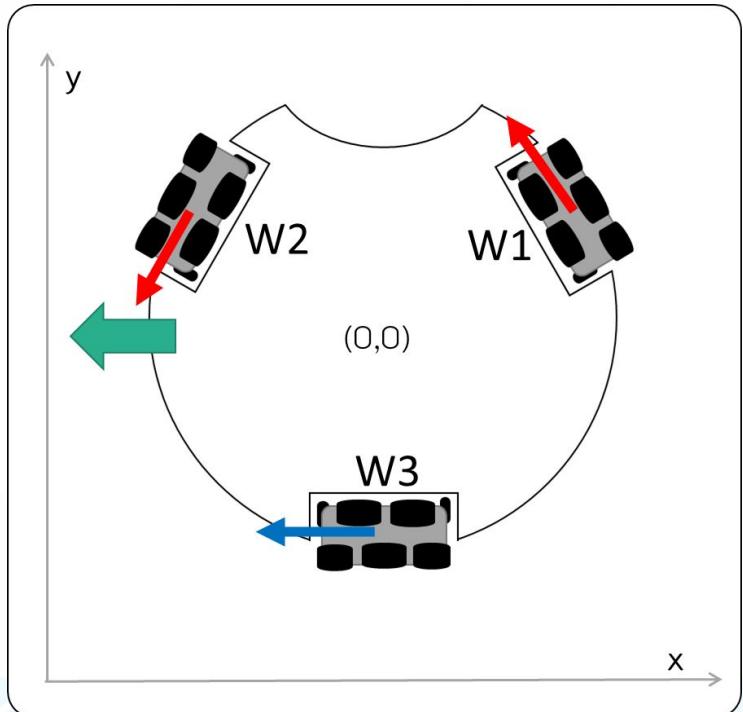
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ไปด้านขวา

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(-25,-25,50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นเคลื่อนที่นุยนต์ไปทางด้านซ้าย



ต้องการเคลื่อนที่ไปที่จุด $Vx=-50$ และ $Vy=0$

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = -(-50) * \sin(30) \Rightarrow 25$

ทิศทางล้อที่ 2 : $V2 = -(-50) * \sin(30) \Rightarrow 25$

ทิศทางล้อที่ 3 : $V3 = -50$

ตัวอย่างโปรแกรม

```
motor(1, 25);
```

```
motor(2, 25);
```

```
motor(3, -50);
```



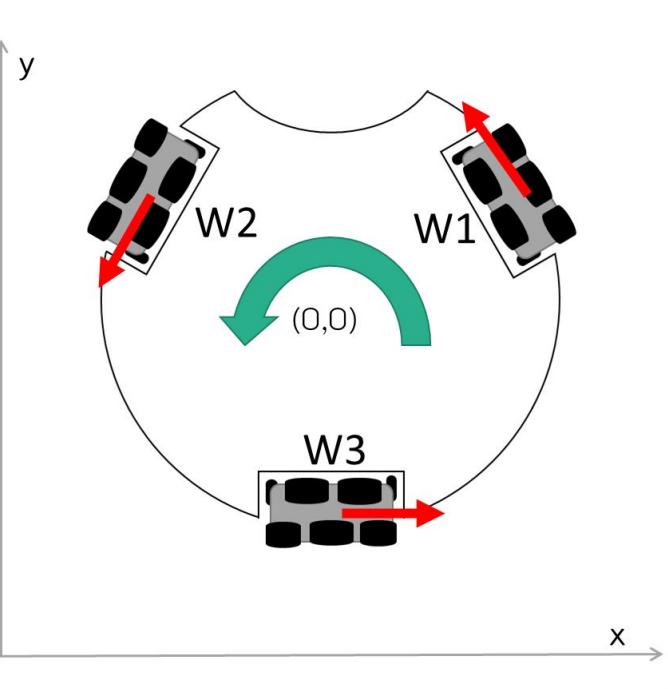
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์ไปด้านซ้าย

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(25,25,-50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นตอนที่ 4 หมุนซ้าย



ต้องการเคลื่อนที่หมุนซ้ายด้วยความเร็วเท่ากับ 50

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = 50$

ทิศทางล้อที่ 2 : $V2 = 50$

ทิศทางล้อที่ 3 : $V3 = 50$

ตัวอย่างโปรแกรม

```
motor(1, 50);
```

```
motor(2, 50);
```

```
motor(3, 50);
```



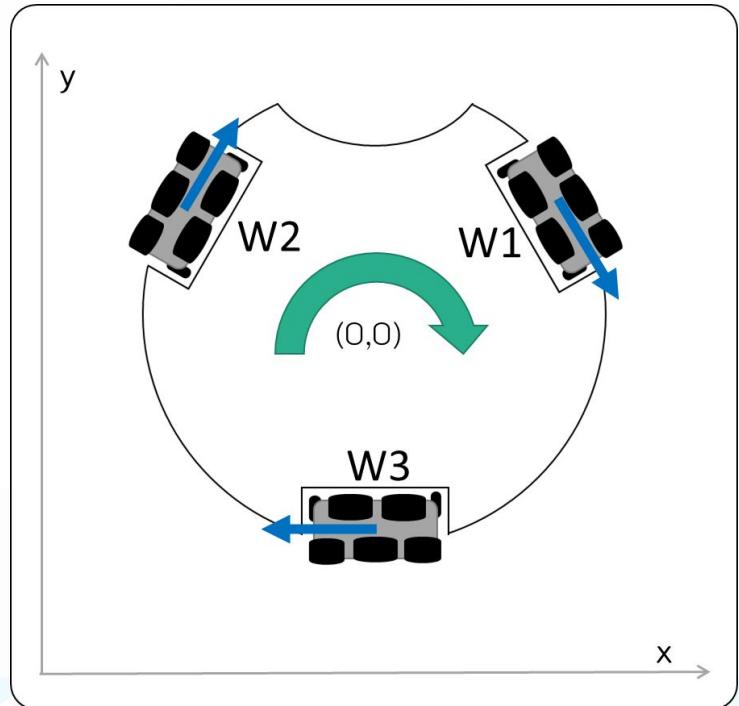
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์หมุนซ้าย

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(50,50,50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



ขั้นเคลื่อนที่นุยนต์หมุนขวา



ต้องการเคลื่อนที่หมุนขวาด้วยความเร็วเท่ากับ 50

วิธีทำ

ทิศทางล้อที่ 1 : $V1 = -50$

ทิศทางล้อที่ 2 : $V2 = -50$

ทิศทางล้อที่ 3 : $V3 = -50$

ตัวอย่างโปรแกรม

```
motor(1, -50);
```

```
motor(2, -50);
```

```
motor(3, -50);
```



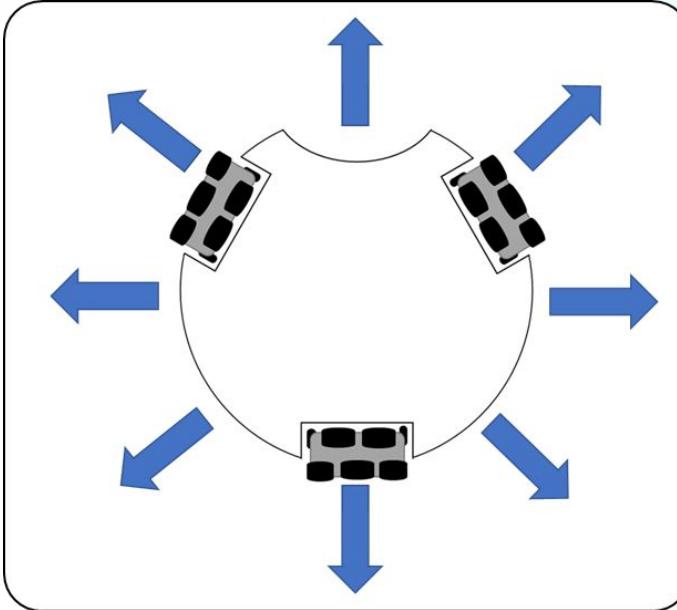
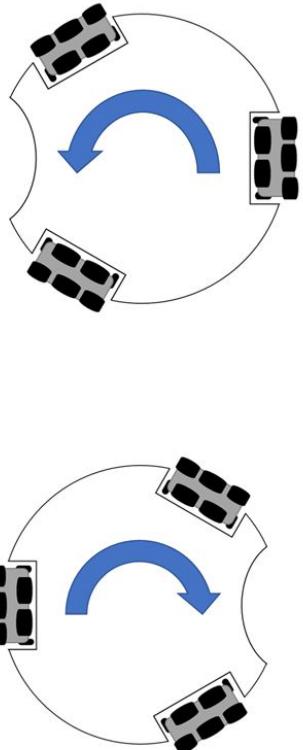
สร้างโปรแกรม ขับเคลื่อนหุ่นยนต์หมุนขวา

```
#include <POP32.h>
void setup()
{
}
void loop()
{
    waitSW_A_bmp();
    wheel(-50,-50,-50);
    delay(3000);
    ao();
}
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
```

ปรับแต่งในตัวอย่าง code -> Test_movement.ino



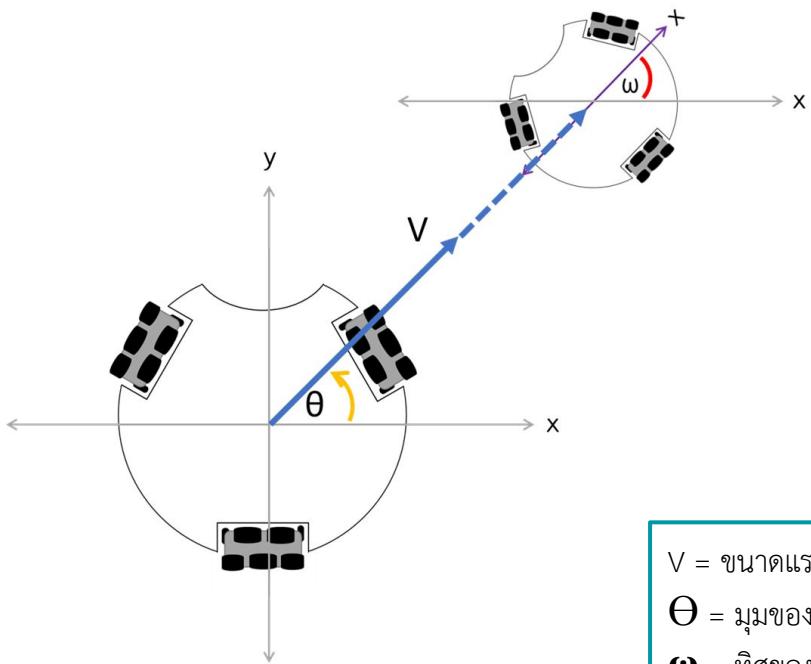
การเคลื่อนที่แบบโฮโลโนมิก (Holonomic)



การเคลื่อนไหวแบบโฮโลโนมิก จะเกิดขึ้นได้หากกระดับความอิสระที่ควบคุมได้เท่ากับทั้งหมด ยกตัวอย่างเช่น ล้อเลื่อนบนรถเข็นซื้อปี๊ง สามารถควบคุมการเคลื่อนที่ได้ในทุกของศาสของความอิสระเท่าที่จะเป็นไปได้ ตรงกันข้ามกับล้อของจักรยาน



ลักษณะการเคลื่อนที่



คำสั่ง $\cos(\text{radians})$

$\sin(\text{radians})$

1 degrees = 0.0174532925 radians

ตัวอย่าง

double x = $\cos(1.566)$

double x = $\cos(90 * 0.0174)$

x จะมีค่าเท่ากับ 1

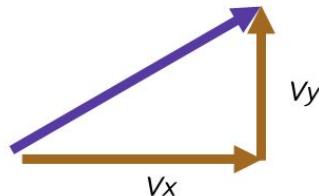
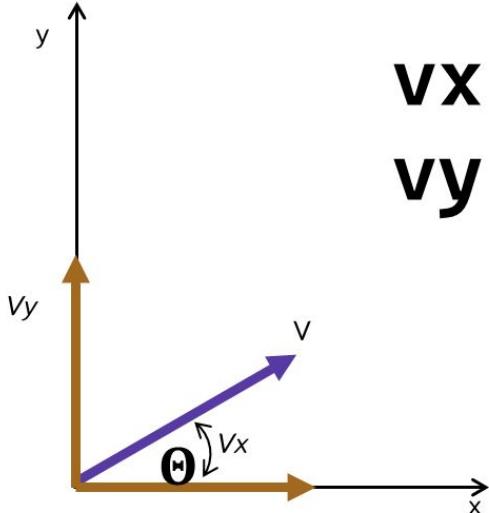
V = ขนาดแรงลัพธ์หรือความเร็ว

Θ = มุมของแรงที่ต้องการเคลื่อนที่ไป

ω = ทิศของหุนยนต์ปลายทางที่ต้องการเมื่อเทียบกับต้นทาง



$$v_x = V * \cos(\theta)$$
$$v_y = V * \sin(\theta)$$





การสร้างโปรแกรม

การแตกเรง

```
thetaRad = theta * degToRad;  
vx = spd * cos(thetaRad);  
vy = spd * sin(thetaRad);
```

กำหนดเรงในแต่ละข้อ

```
wheel(spd1, spd2, spd3);
```

คำนวนเรงในแต่ละล้อ

```
spd1 = vy * cos30 - vx * sin30 + omega;  
spd2 = -vy * cos30 - vx * sin30 + omega;  
spd3 = vx + omega;
```



สร้างโปรแกรมแบบโอลีมิก

ปรับแต่งในตัวอย่าง code ->Test_movement_Holonomic.ino

```
#define degToRad 0.0174f
#define sin30 sin(30.f * degToRad)
#define cos30 cos(30.f * degToRad)
float thetaRad, vx, vy, spd1, spd2, spd3;
void holonomic(float spd, float theta, float omega)
{
    thetaRad = theta * degToRad;
    vx = spd * cos(thetaRad);
    vy = spd * sin(thetaRad);
    spd1 = vy * cos30 - vx * sin30 + omega;
    spd2 = vy * cos30 - vx * sin30 + omega;
    spd3 = vx + omega;
    wheel(spd1, spd2, spd3);
}
```

เรียกใช้งานฟังก์ชัน

holonomic(20,0,0)



สร้างโปรแกรมขับเคลื่อนหุ่นยนต์หมุนขวา

```
#define degToRad 0.0174f
#define sin30 sin(30.f * degToRad)
#define cos30 cos(30.f * degToRad)
float thetaRad, vx, vy, spd1, spd2, spd3;
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
void holonomic(float spd, float theta, float omega)
{
    thetaRad = theta * degToRad;
    vx = spd * cos(thetaRad);
    vy = spd * sin(thetaRad);
    spd1 = vy * cos30 - vx * sin30 + omega;
    spd2 = vy * cos30 - vx * sin30 + omega;
    spd3 = vx + omega;
    wheel(spd1, spd2, spd3);
}
```

```
#include <POP32.h>
void setup()
{}
void loop() {
    waitSW_A_bmp();
    holonomic(30, 0, 0);
    delay(3000);
    ao();
}
```



สร้างโปรแกรมขับเคลื่อนหุ่นยนต์แบบความโค้ง Curve

```
#define degToRad 0.0174f
#define sin30 sin(30.f * degToRad)
#define cos30 cos(30.f * degToRad)
float thetaRad, vx, vy, spd1, spd2, spd3;
void wheel(int s1, int s2, int s3)
{
    motor(1, s1);
    motor(2, s2);
    motor(3, s3);
}
void holonomic(float spd, float theta, float omega)
{
    thetaRad = theta * degToRad;
    vx = spd * cos(thetaRad);
    vy = spd * sin(thetaRad);
    spd1 = vy * cos30 - vx * sin30 + omega;
    spd2 = vy * cos30 - vx * sin30 + omega;
    spd3 = vx + omega;
    wheel(spd1, spd2, spd3);
}
```

```
#include <POP32.h>
void setup() {
}
void loop() {
    waitSW_A_bmp();
    holonomic(40, 180, -15);
    delay(2000);
    ao();
}
```

หมุนตามเข็มนาฬิกา

holonomic(40, 180, -15);

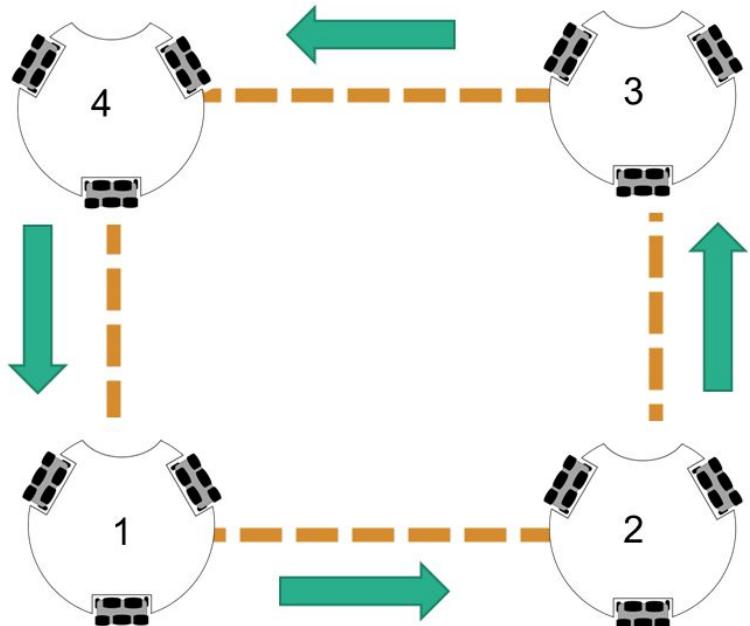
หมุนทวนเข็มนาฬิกา

holonomic(40, 0, 15);



สร้างโปรแกรมขับเคลื่อนหุ่นยนต์เป็นรูปสี่เหลี่ยม

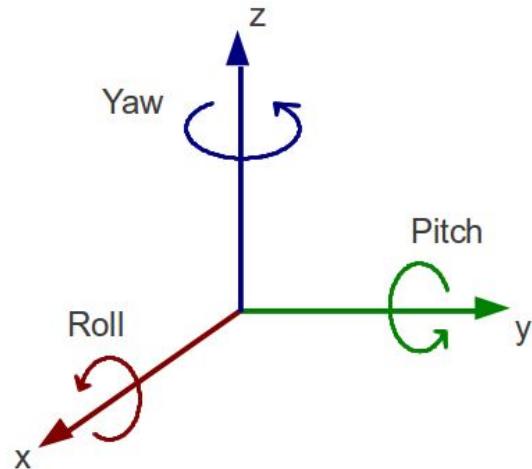
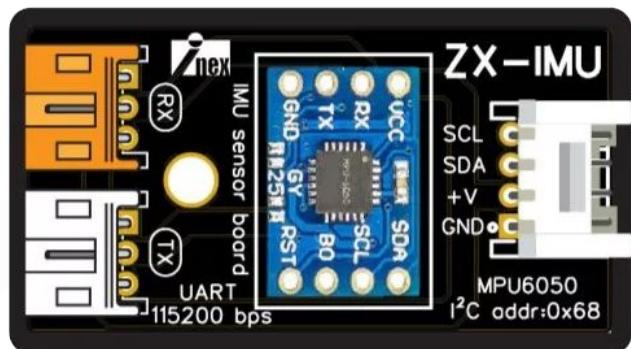
ปรับแต่งในตัวอย่าง code ->Test_movement_Holonomic.ino



```
#include <POP32.h>
void setup()
{}
void loop() {
    waitSW_A_bmp();
    holonomic(30, 0 , 0); delay(3000);
    holonomic(30, 90 , 0); delay(3000);
    holonomic(30, 180 , 0); delay(3000);
    holonomic(30, 270 , 0); delay(3000);
    ao();
}
```

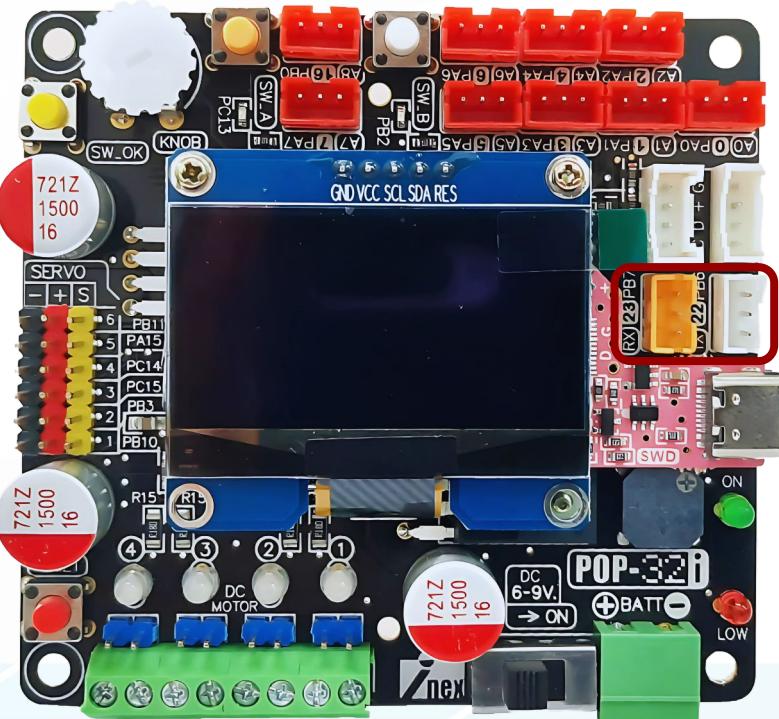


สามารถต่อใช้งานได้ทั้ง I2C(SDA,SCL) และ Serial(Rx,Tx) โดยมีโมดูลประจุอิเล็กทรอนิกส์ MPU6050 เป็นเซ็นเซอร์ Gyro และ Accelerometer โดยมีไมโครคอนโทรลเลอร์ ใช้เป็นตัวกลางเพื่ออ่านค่าจาก MPU6050 และส่งค่าออก Serial(Rx,Tx)

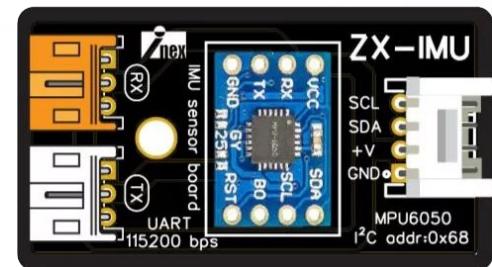


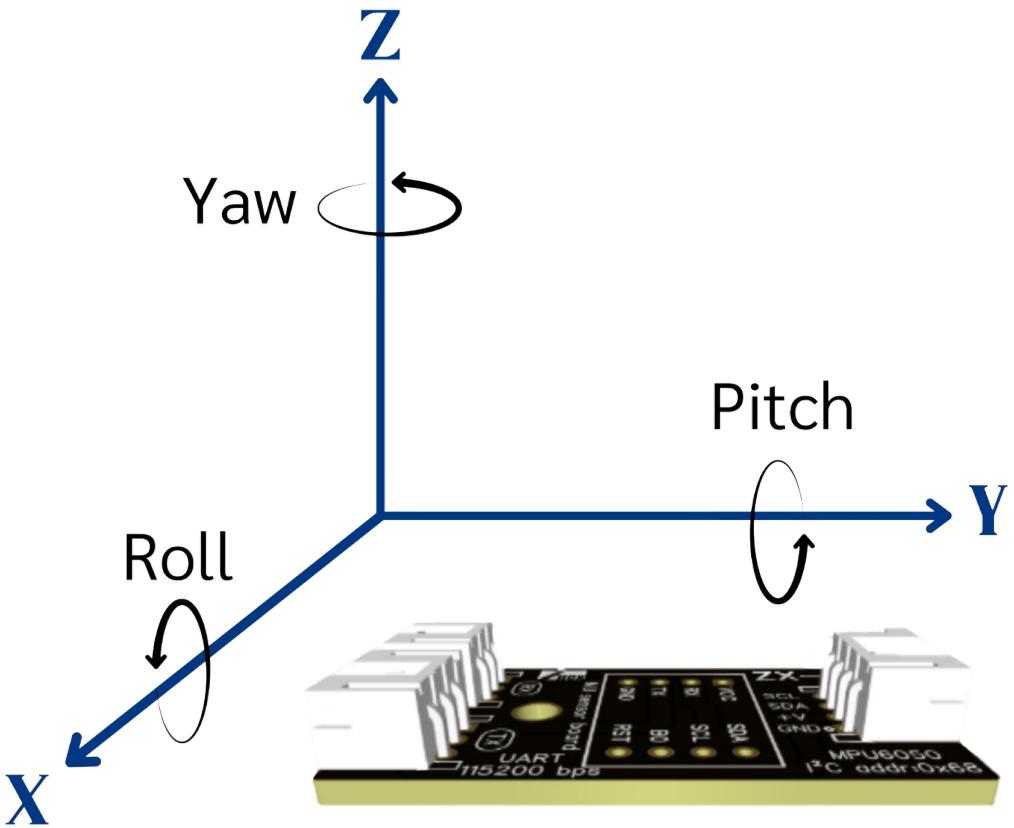


การต่อใช้งาน



ใช้แบบ UART





Yaw

นำมานี้เป็นเข็มทิศอ้างอิงให้หุ่นยนต์



Serial.begin(baud rate)

▶ กำหนดอัตราการสื่อสารข้อมูล เช่น 115200 bps

Serial.write(data)

▶ ส่งข้อมูลออกไป ได้เพียงแค่ทีละ 1 byte

Serial.available()

▶ จำนวนไบต์ที่สามารถอ่านได้

Serial.read()

▶ อ่านข้อมูลอนุกรมที่เข้ามา

ฟังก์ชันอ่านค่า Yaw

ปรับแต่งในตัวอย่าง code ->Test_ZX-IMU.ino



```
float pvYaw;  
uint8_t rxCnt = 0, rxBuf[8];  
bool getIMU() {  
    while (Serial1.available()) {  
        rxBuf[rxCnt] = Serial1.read();  
        if (rxCnt == 0 && rxBuf[0] != 0xAA) return;  
        rxCnt++;  
        if (rxCnt == 8) {  
            rxCnt = 0;  
            if (rxBuf[0] == 0xAA && rxBuf[7] == 0x55) {  
                pvYaw = (int16_t)(rxBuf[1] << 8 | rxBuf[2]) / 100.f;  
                return true;  
            }  
        }  
    }  
    return false;  
}
```

เมื่อมีข้อมูลส่งเข้ามา

อ่านค่าเก็บไว้ในตัวแปร

ตรวจสอบความถูกต้อง

ตรวจสอบความถูกต้อง

อ่านและคำนวณค่ามุม Yaw



ฟังก์ชันรีเซตมุม Yaw

ขณะรีเซตหุ่นยนต์ต้องอยู่ในจังหวะ อย่างน้อย 3 วินาที ก่อนใช้งาน

```
void zeroYaw() {
```

```
    Serial1.begin(115200); delay(100);
```

เริ่มต้นใช้งาน Serial ช่องหมายเลข 1

```
    Serial1.write(0XA5); Serial1.write(0X54); delay(100);
```

ปรับปรุงค่ามุม pitch และ roll ให้เป็น 0

```
    Serial1.write(0XA5); Serial1.write(0X55); delay(100);
```

ปรับปรุงค่ามุม Yaw ให้เป็น 0

```
    Serial1.write(0XA5); Serial1.write(0X52); delay(100);
```

เลือกโหมดอัตโนมัติ

```
}
```



ฟังก์ชันอ่านค่า Auto_zero

```
void Auto_zero(){  
    zeroYaw();getIMU();  
  
    int timer = millis();  
  
    oled.clear(); oled.text(1, 2,"Setting zero");  
  
    while (abs(pvYaw) > 0.05){  
        if(getIMU()){  
            oled.text(3, 6,"Yaw: %f " ,pvYaw);oled.show();  
            beep();  
            if (millis() - timer > 5000){  
                zeroYaw();  
                timer = millis();  
            }  
        }  
    }  
  
    oled.clear();oled.show();  
}
```

วนทำซ้ำจนกว่าค่า Yaw น้อยกว่า 0.05 องศา

ถ้าเวลามากกว่า 5 วินาทีแล้ว
ให้ เช็ตใหม่อีกครั้ง



สร้างโปรแกรมอ่านค่า Yaw

```
void setup() {  
    Auto_zero();  
}  
  
void loop() {  
    if (SW_A()) {  
        Auto_zero();  
    }  
    getIMU();  
    oled.text(0, 0, "Yaw=%f      ",pvYaw);  
    oled.show();  
}
```

ปรับแต่งในตัวอย่าง code ->Test_ZX-IMU.ino

กดปุ่ม A เพื่อรีเซ็ตเข็มทิศ
จากนั้นหมุนหุ่นยนต์เพื่อทดสอบ



การเคลื่อนที่เพื่อรักษาทิศของหุ่นยนต์ให้คงที่

การเคลื่อนที่เพื่อรักษาทิศอ้างอิงของหุ่นยนต์ให้คงที่จะต้องอาศัยการควบคุม

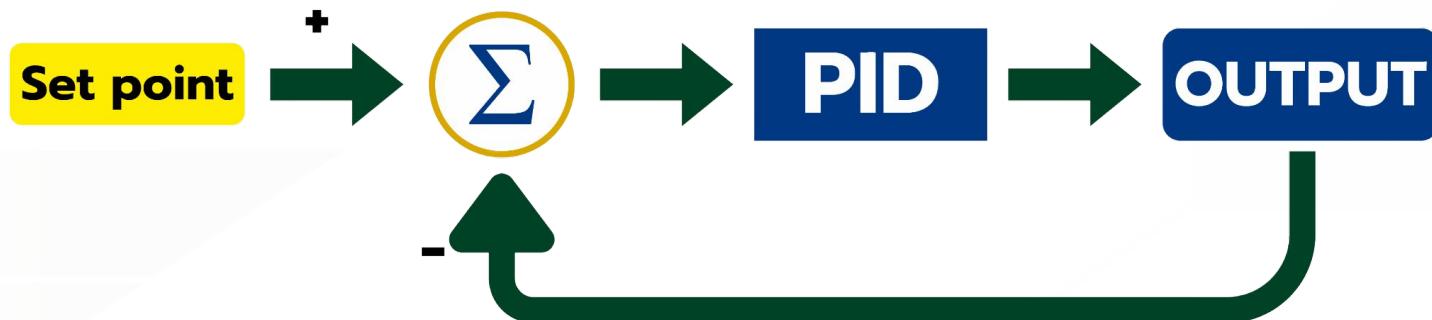
ความเร็ว
ความเร็ว
ความเร็ว
ความเร็ว





การควบคุมแบบ PID : Proportional Integral Derivative

เป็นการควบคุมแบบป้อนกลับ โดยใช้วิธีการนำค่าความผิดพลาดที่ได้จากผลต่างระหว่าง ค่าที่ต้องการ (Set point) กับค่าป้อนกลับมาคำนวณ เพื่อพยายามลดค่าความผิดพลาดให้เหลือน้อยที่สุด หรือให้ค่าความผิดพลาดเข้าใกล้ 0 ให้ได้มากที่สุด





การควบคุมแบบ PID : Proportional Integral Derivative

P : สัดส่วนความผิดพลาด

$$\text{Error} = \text{Set_point} - \text{feedback}$$

I : สัดส่วนความผิดพลาดสะสม

$$\text{SumError} = \text{SumError} + \text{Error}$$

D : สัดส่วนความผิดพลาดปัจจุบันต่ออดีต

$$\text{dError} = \text{Error} - \text{PrevError}$$

สมการ PID จะนำค่า kp,ki,kd มาเพิ่มอัตราการขยายในแต่ละเทอม

$$\text{Output} = (\text{Error} * \text{kp}) + (\text{SumError} * \text{ki}) + (\text{dError} * \text{kd})$$



หน้าที่ของแต่ละเทอม

Kp : ช่วยทำหน้าที่ควบคุมทิศทางหุ่นยนต์ให้เข้าใกล้สู่จุดที่ค่าผิดพลาดเท่ากับ 0

Ki : ช่วยให้หุ่นยนต์เข้าใกล้สู่จุดที่ค่าผิดพลาดเท่ากับ 0 มากขึ้น

Kd : ช่วยลดการสะบัด ให้หุ่นยนต์เข้าสู่จุดที่ค่าผิดพลาดเท่ากับ 0 ได้เร็ว

การปรับจูนค่า Kp,Ki,Kd จะขึ้นอยู่กับความเร็วของมอเตอร์, การตอบสนองของค่า Error กล่าวคือ ความเร็วของตัวประมวลผลหรือการตอบสนองของเซนเซอร์ที่ใช้งานนั้นเอง ในโปรแกรมตัวอย่างจะเห็นว่าอาจปรับแค่ค่า kp ก็เพียงพอที่จะทำให้หุ่นยนต์ปรับทิศได้อย่างมีประสิทธิภาพแล้ว



```
float pvYaw;  
#define head_Kp 0.1f  
#define head_Ki 0.0f  
#define head_Kd 0.0f  
float head_error, head_pError,head_w ,head_d,head_i;
```

ประกาศตัวแปรและกำหนดค่า

```
void setup(){  
    zeroYaw();  
    waitSW_A_bmp();  
}
```

เริ่มต้นใช้งานรีเซ็ตค่า Yaw และ รอกดปุ่ม



คำสั่ง

constrain(value, min, max)

เป็นคำสั่งที่ใช้กำหนดค่าให้อยู่ในช่วงที่เราต้องการ

ตัวอย่าง

```
int y = constrain(300, -180, 180)
```

y จะมีค่าเท่ากับ 180



P : สัดส่วนความผิดพลาด

$$\text{Error} = \text{Set_point} - \text{feedback}$$

$$\text{head_error} = \text{spYaw} - \text{pvYaw}$$

หาค่าความผิดพลาด

$$\text{head_w} = (\text{head_error} * \text{head_Kp})$$

เพิ่มอัตราการขยาย



I : สัดส่วนความผิดพลาดสะสม

$$\text{SumError} = \text{SumError} + \text{Error}$$

```
head_i = head_i + head_error
```

สะสมค่าความผิดพลาด

```
head_i = constrain(head_i, -180, 180)
```

จำกัดกรอบค่าความผิดพลาด ไม่ให้สูงเกินค่าความเร็วสูงสุด

```
head_w = (head_i * head_Ki)
```

เพิ่มอัตราการขยาย



D : สัดส่วนความผิดพลาดปัจจุบันต่ออดีต

$$dError = Error - PrevError$$

```
head_d = head_error - head_pError  
head_pError = head_error
```

ผลต่างค่าความผิดพลาดปัจจุบันต่ออดีต

```
head_w = (head_d * head_Kd)
```

เพิ่มอัตราการขยาย



รวมทุกเทอมเข้าด้วยกัน

```
head_w = (head_error * head_Kp) + (head_i * head_Ki) + (head_d * head_Kd)
```

รวมทุกเทอมเข้าด้วยกัน

```
head_w = constrain(head_w , -100, 100);
```

คำสั่งปรับความเร็วมอเตอร์

จำกัดค่าไม่ให้เกินความเร็วมอเตอร์

```
holonomic(0, 0, head_w)
```



```
#define head_Kp 0.1f
#define head_Ki 0.0f
#define head_Kd 0.0f
float head_error, head_pError,head_w ,head_d,head_i;
void heading(float spd, float theta, float spYaw){
    head_error = spYaw - pvYaw;
    head_i = head_i + head_error;
    head_i = constrain(head_i ,-180,180);
    head_d = head_error - head_pError;
    head_w = (head_error * head_Kp) + (head_i * head_Ki) +
    (head_d * head_Kd);    head_w = constrain(head_w ,-100,100);
    holonomic(spd, theta, head_w);
    head_pError = head_error;
}
```



สร้างโปรแกรมเคลื่อนที่รักษาทิศของหุ่นยนต์

```
void setup() {  
    Auto_zero();  
    waitSW_A_bmp();  
}  
  
void loop() {  
    if (SW_A()) {  
        wheel(0,0,0);  
        Auto_zero();  
    }  
    getIMU();  
    heading(0, 0, 0);  
}
```

ปรับแต่งในตัวอย่าง code ->Test_heading.ino

กดปุ่ม A เพื่อเลือกทิศใหม่

heading(**float** spd, **float** theta, **float** spYaw)

Spd = ความเร็ว

theta = มุมที่ต้องการเคลื่อนที่ไป

spYaw = ทิศที่ต้องการรักษาไว้



เคลื่อนที่ไปข้างหน้าโดยการรักษาทิศไว้ให้คงที่

```
void setup() {  
    Auto_zero();  
    waitSW_A_bmp();  
}  
  
void loop() {  
    if (SW_A()) {  
        wheel(0,0,0);  
        Auto_zero();  
    }  
    getIMU();  
    heading(20, 90, 0);  
}
```

heading(**float** spd, **float** theta, **float** spYaw)

Spd = 20

theta = 90

spYaw = 0

ความแตกต่างแบบระหว่าง



ใช้ ZX-IMU

```
void setup() {
    Auto_zero();
    waitSW_A_bmp();
}

void loop() {
    if(SW_A()){
        Auto_zero();
    }
    getIMU();
    heading(20, 90, 0);
}
```

ปรับหมุนตัวให้ด้านหน้าเข้าที่ทิศอ้างอิงเสมอ

ไม่ใช้ ZX-IMU

```
void setup() {
    Auto_zero();
    waitSW_A_bmp();
}

void loop() {
    if(SW_A()){
        Auto_zero();
    }
    getIMU();
    holonomic(20, 90, 0);
}
```

ไม่ปรับหมุนตัวคืนมาที่ทิศเดิม



คำสั่งตรวจสอบเงื่อนไข while

รูปแบบ

```
while(เงื่อนไข) {  
    ทำการสั่งจนเงื่อนไขเป็นเท็จ;  
}
```

ออกจาก while แบบทันทีได้ด้วย

คำสั่ง break;

```
#include <POP32.h>  
int i=0;  
void setup() {}  
void loop() {  
    while(SW_A()) {  
        beep();  
    }  
}
```



คำสั่งวนทำซ้ำแบบมีเงื่อนไข for

รูปแบบ

```
for(ค่าเริ่มต้น;เงื่อนไข;เพิ่ม/ลดค่า)
{
    //statement(s);
}
```

ออกจาก while แบบทันทีได้ด้วย

คำสั่ง **break;**

```
#include <POP32.h>
void setup() {
    waitSW_OK_bmp(); }
void loop() {
    for (int i = 0; i < 5; i++)
    {
        oled.text(0, 0, "%d ", i);
        oled.show();
        delay(100);
    }
}
```



คำสั่ง

millis()

เป็นคำสั่งที่ใช้อ่านค่าเวลาที่กำลังเดิมมีหน่วยเป็น มิลลิวินาที

ตัวอย่าง

```
unsigned long loopTime = millis()
```

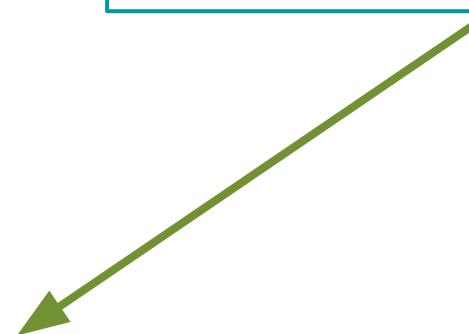
เป็นคำสั่งที่ใช้อ่านค่าเวลาที่กำลังเดิมมีหน่วยเป็น มิลลิวินาที



สร้างโปรแกรมออกจาก while โดยใช้เวลาในการหยุด

```
loopTimer = millis();
while (1) {
    command1;
    command2;
    (คำสั่งต่างๆที่เกี่ยวข้องในโปรแกรม)
    command3;
    command4;
    if (millis() - loopTimer >= 2000) break;
}
```

จำนวนอยู่ใน while นาน 2 วินาที



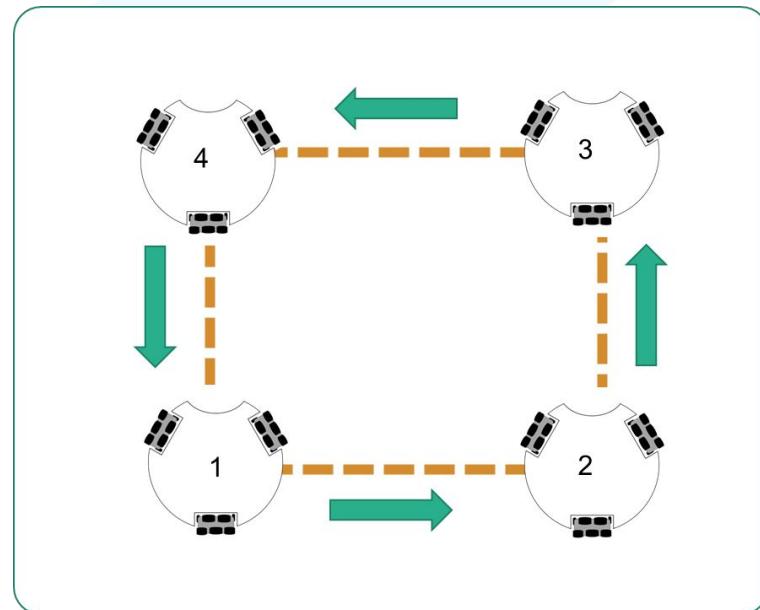
```

unsigned long loopTimer = millis();
void loop() {
    ao();
    waitSW_A_bmp();
    zeroYaw();
    loopTimer = millis();
    while(1){
        getIMU();heading(30, 0, 0);
        if (millis()-loopTimer >= 2000)break;
    }
    loopTimer = millis();
    while(1){
        getIMU();heading(30, 90, 0);
        if (millis()-loopTimer >= 2000)break;
    }
    loopTimer = millis();
    while(1){
        getIMU();heading(30, 180, 0);
        if (millis()-loopTimer >= 2000)break;
    }
    loopTimer = millis();
    while(1){
        getIMU();heading(30, 270, 0);
        if (millis()-loopTimer >= 2000)break;
    }
}

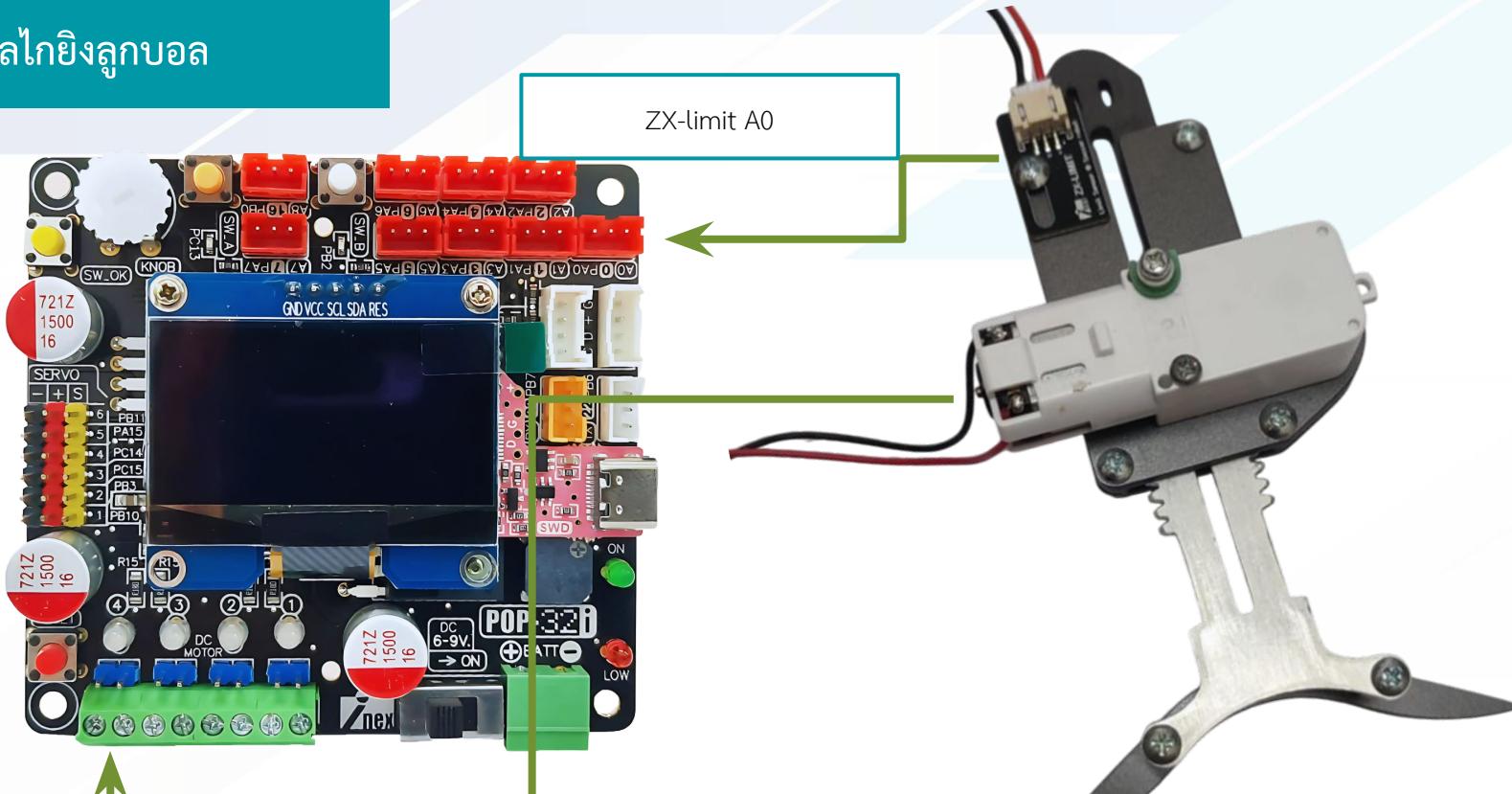
```

ขับเคลื่อนหุ่นยนต์เป็นรูปสี่เหลี่ยม

ปรับแต่งในตัวอย่าง code ->Test_holonomic_heading.ino



กลไกยิงลูกบอล



มอเตอร์ซี่องที่ 4



ตัวอย่างโค้ดควบคุมอ่านค่าจากตัว ZX-limit

```
#include <POP32.h>
void setup() {}
void loop() {
    int val = analog(0);
    oled.text(0, 0, "Adc: %d      ", val);
    oled.show();
}
```

แสดงค่าที่อ่านได้จากตัว ZX-limit ออกทางหน้าจอ OLED
ทดลองเลื่อนก้านยิ่งเข้า-ออกจนสุดแล้วดูการเปลี่ยนแปลง



ปรับแต่งในตัวอย่าง code ->Test_shoot_V3.ino

```
void shoot() {
    motor(4, reloadSpd);
    delay(150);
    motor(4, 0);
    delay(50);
}
```

```
#include <POP32.h>
#define limPin PA0
#define reloadSpd 60
int timer = 0;
void setup() {
void loop() {
    if (SW_A()) { reload();}
    if (SW_B()) { shoot(); }
    if (SW_OK()) {shoot();reload();}
}
```

```
void reload() {
motor(4, reloadSpd);
timer = 0;
for (int i = 0; i < 2000; i++) {
    timer++;
    if (analog(limPin)>1000) break;
    delay(1);
}
if (timer == 2000) {
    motor(4, -reloadSpd);
    delay(500);
    motor(4, reloadSpd);
    timer = 0;
    for (int i = 0; i < 2000; i++) {
        timer++;
        if (analog(limPin)>1000) break;
        delay(1);
    }
    motor(4, 0);
}
```





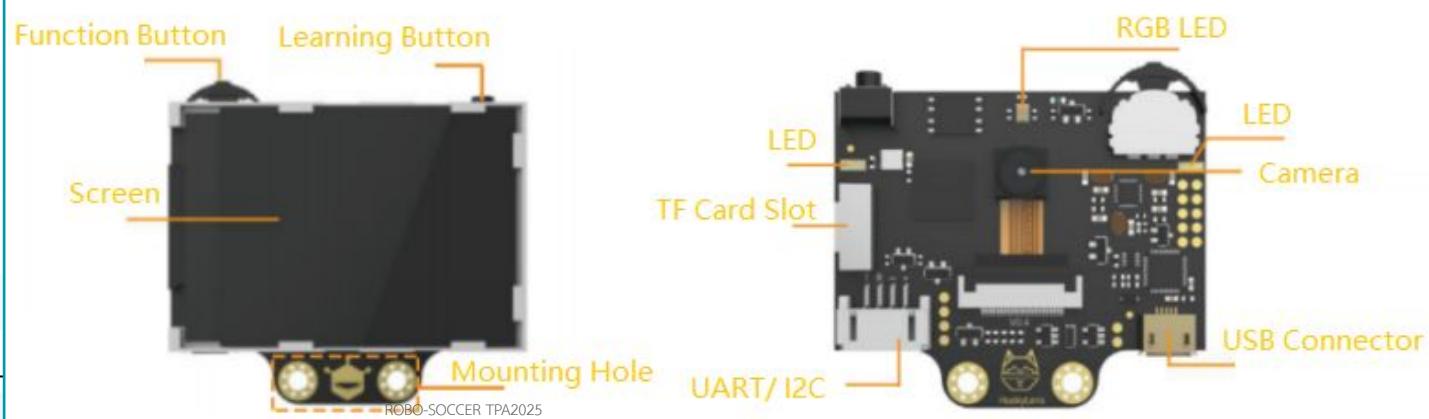
- face recognition
- object tracking
- object recognition
- line tracking
- color recognition
- tag recognition
- object classification.



https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336



- Processor: Kendryte K210
- Image Sensor:
- SEN0305 Huskylens: OV2640 (2.0Megapixel Camera)
- SEN0336 Huskylens PRO: OV5640 (5.0MegaPixel Camera)
- Supply Voltage: 3.3~5.0V
- Communication Port: UART; I2C
- Display: 2.0-inch IPS screen with 320*240 resolution



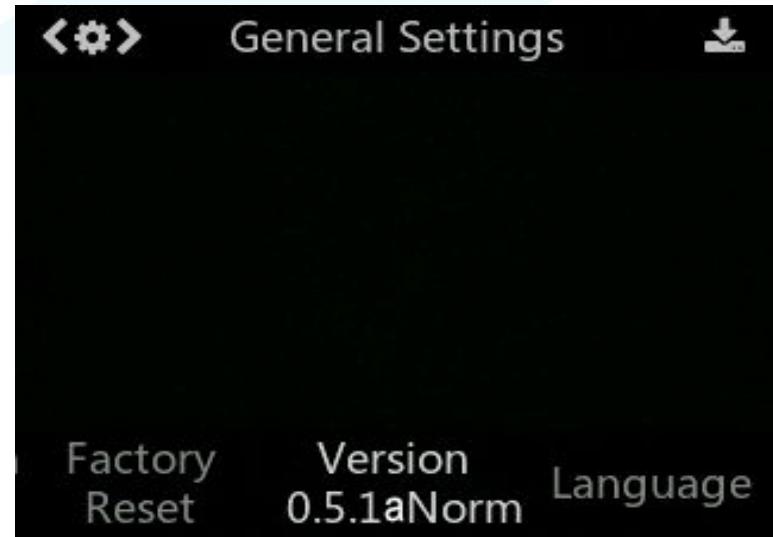
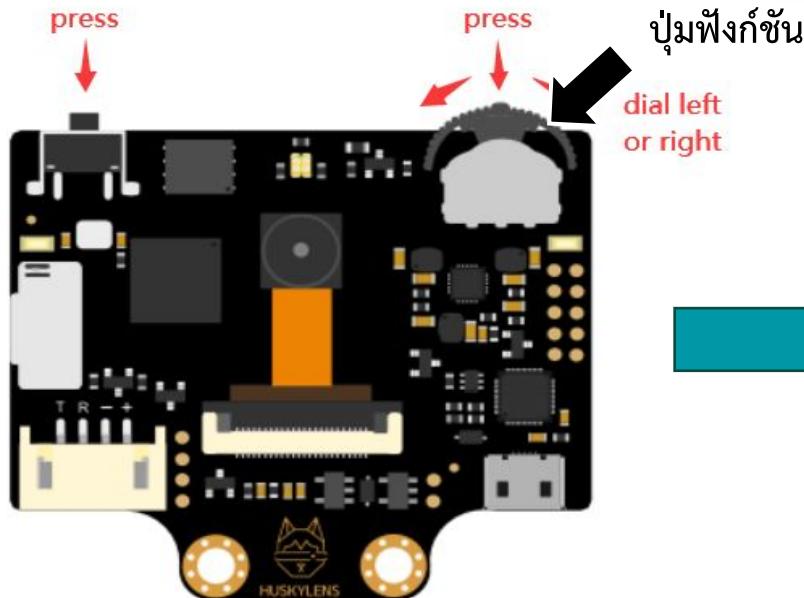


การอัพเกรดเฟิร์มแวร์





วิธีการตรวจสอบเวอร์ชัน



หมุน “ปุ่มพังก์ชัน” ไปทางขวาจนถึงตัวเลือกสุดท้าย “General Settings” กด “ปุ่มพังก์ชัน” สัก ๆ เพื่อเข้าสู่เมนูรอง



ดาวน์โหลดโปรแกรม อัพเกรดเฟิร์มแวร์

HuskyLens / HUSKYLENSUploader Public

Code Issues 3 Pull requests Actions Projects Wiki Security Insights

master Go to file Code ▾ 1

Clone HTTPS GitHub CLI

<https://github.com/HuskyLens/HUSKYLENSUploader>

This is the firmware uploader on windows

Readme 23 stars 10 watching 12 forks

Releases

No releases published

jimaobian update 0.5.3 ...
driver Init update
HUSKYLENS Uploader... add new
HUSKYLENSV0.4.7Sta... add bin
HUSKYLENSV0.4.7bSt... update C
HUSKYLENSV0.4.8Cla... add bin
HUSKYLENSV0.4.9Cla... update t
HUSKYLENSWithMod... Init update

2 years ago

<https://github.com/HuskyLens/HUSKYLENSUploader>



อัปเกรดเฟิร์มแวร์

ทำการแตกไฟล์ HUSKYLENSUploader-master.ZIP ภายในจะประกอบไปด้วยไฟล์เฟิร์มแวร์เวอร์ชันต่างๆ และโปรแกรมที่ใช้อัปเกรดเฟิร์มแวร์ดังรูป

driver	firwareresult.bin	HUSKYLENS Uploader-V2.1.zip	HUSKYLENSV0.4.7bStable.bin	HUSKYLENSV0.4.7Stable.bin	HUSKYLENSV0.4.8Class.bin	HUSKYLENSV0.4.9Class.bin	HUSKYLENSWi thModelV0.3.3Stable.kfpkg
HUSKYLENSWi thModelV0.3.7Stable.kfpkg	HUSKYLENSWi thModelV0.4.5Stable.kfpkg	HUSKYLENSWi thModelV0.4.6Stable.kfpkg	HUSKYLENSWi thModelV0.4.7Stable.kfpkg	HUSKYLENSWi thModelV0.4.8Class.kfpkg	HUSKYLENSWi thModelV0.4.9Class.kfpkg	HUSKYLENSWi thModelV0.5.1aNorm.kfpkg	HUSKYLENSWi thModelV0.5.1Norm.kfpkg
HUSKYLENSWi thModelV0.5.3	K-Flash.exe	kflash.py		README.md			



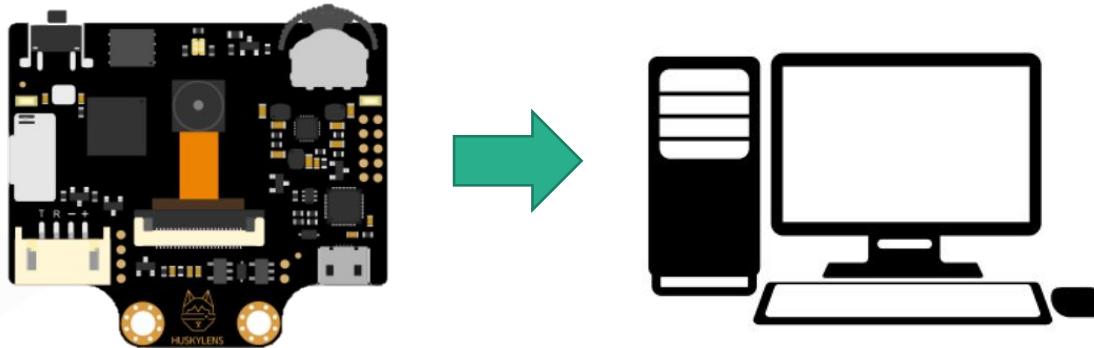
สำหรับ Windows



HUSKYLENS
Uploader-V2.1.exe

ทำการแตกไฟล์ที่ชื่อว่า HUSKYLENS Uploader-V2.1.zip จากนั้นจะได้ไฟล์ที่ชื่อว่า HUSKYLENS

Uploader-V2.1.exe โปรแกรมที่ทำหน้าที่อัพเกรดเฟิร์มแวร์



เชื่อมต่อ HuskyLens เข้ากับคอมพิวเตอร์ด้วยสาย microUSB



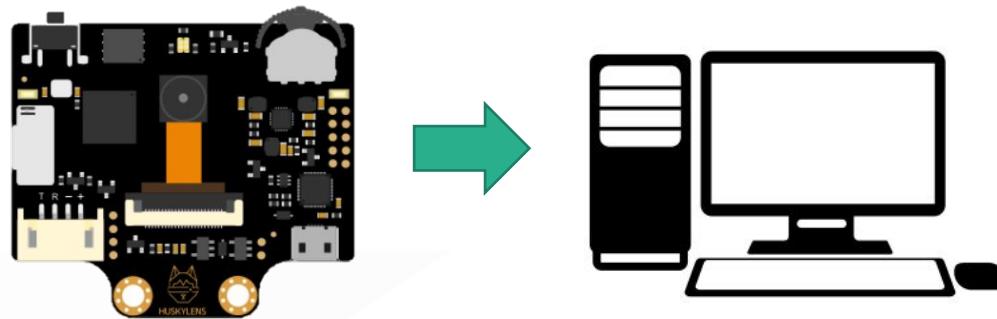
สำหรับ Windows



HUSKYLENS
Uploader-V2.1.exe

ทำการแตกไฟล์ที่ชื่อว่า HUSKYLENS Uploader-V2.1.zip จากนั้นจะได้ไฟล์ที่ชื่อว่า HUSKYLENS

Uploader-V2.1.exe โปรแกรมที่ทำหน้าที่อัปเกรดเฟิร์มแวร์



ตรวจสอบหมายเลขพอร์ต

▼ PORT (COM & LPT)

Silicon Labs CP210x USB to UART Bridge (COM6)

เชื่อมต่อ HuskyLens เข้ากับคอมพิวเตอร์ด้วยสาย microUSB

สำหรับ Windows



คลิกปุ่ม "Select File" เพื่อโหลดเฟิร์มแวร์ โดยเลือกไฟล์ที่ชื่อว่า

"HUSKYLENSWithModelV0.5.1aNorm.kfpkg"

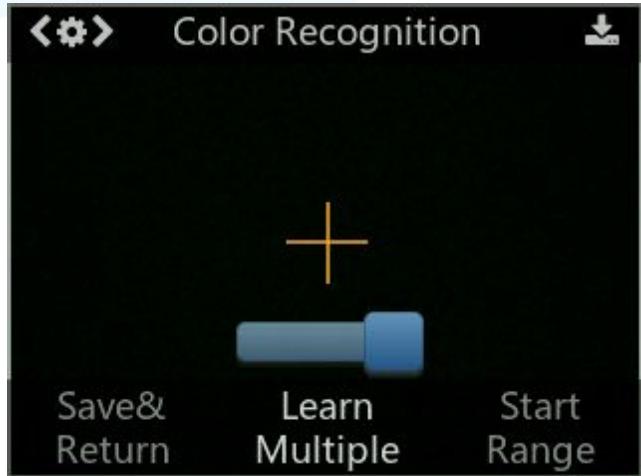
คลิกปุ่ม "Upload" เพื่อยืนยัน และรอประมาณ 5 นาทีจนกว่าจะเสร็จ





การตั้งค่า

1. เลือกโมด "Color Recognition"
2. กดปุ่มฟังก์ชันค้างไว้เพื่อเข้าสู่การตั้งค่าของฟังก์ชันการรู้จำสี
3. เปิดใช้งาน "Learn Multiple"
4. บันทึกการตั้งค่า Save&Return





การเรียนรู้และการตรวจจับ

การตั้งค่า

1. เลือกโหมด "Color Recognition"
2. กดปุ่มฟังก์ชันค้างไว้เพื่อเข้าสู่การตั้งค่าของฟังก์ชันการรู้จำสี
3. เปิดใช้งาน "Learn Multiple"
4. บันทึกการตั้งค่า Save&Return





การจดจำสี Color Recognition

พิงก์ชันนี่สามารถเรียนรู้

จดจำ และติดตามสีที่ระบุได้

ข้อแนะนำ

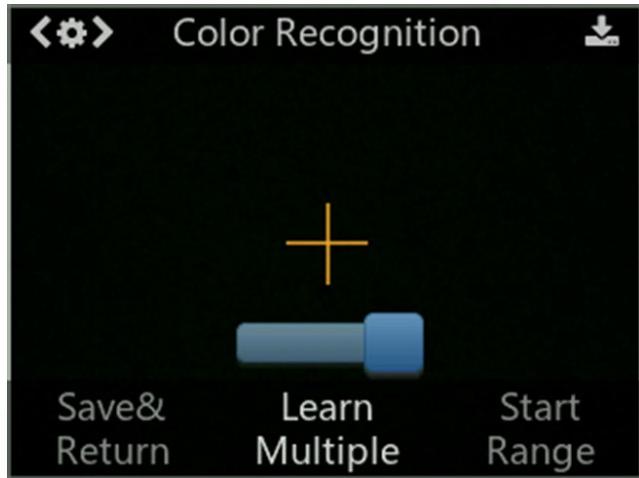
การรู้จำสีได้รับผลกระทบอย่างมากจากแสงโดยรอบ บางครั้ง HuskyLens อาจระบุสีที่คล้ายกันผิด โปรดพยายามให้แสงโดยรอบไม่เปลี่ยนแปลง





การจดจำสี Color Recognition

การตั้งค่า



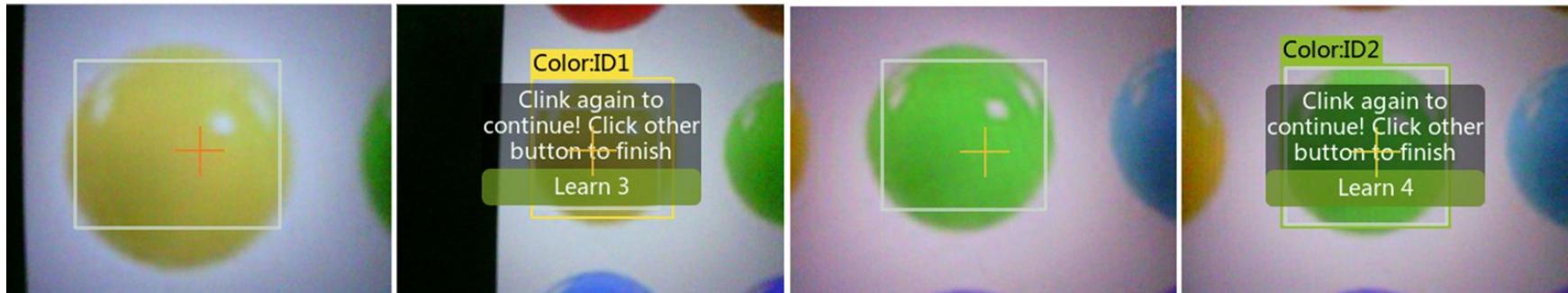
1. เลือกโหมด “Color Recognition”
2. กดปุ่มฟังก์ชันค้างเอาไว้ เพื่อเข้าสู่การตั้งค่าของฟังก์ชันการรู้จำสี
3. เปิดใช้งาน “Learn Multiple”
4. บันทึกการตั้งค่า Save&Return



การเรียนรู้และการตรวจจับ

การเรียนรู้หลายรายการ

กด “ปุ่มการเรียนรู้” สั้นๆ ก่อนการนับถอยหลังจะจบลง



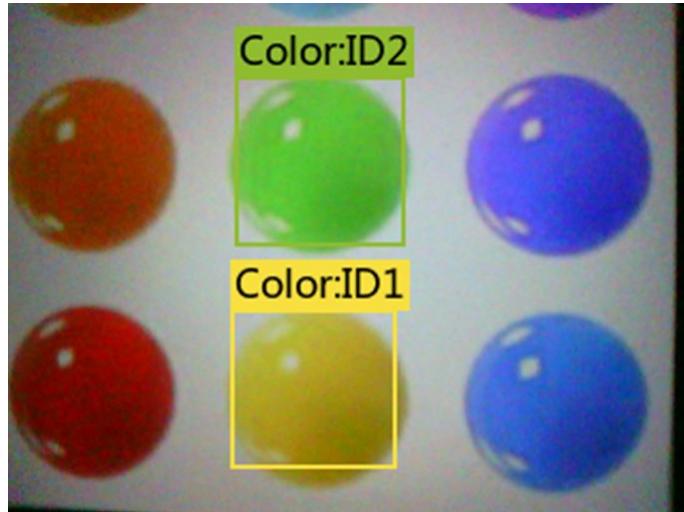
การลบ

เมื่อกดปุ่มเรียนรู้เสร็จแล้ว รอจนกว่าจะหมดเวลา จากนั้นกดปุ่มเรียนรู้อีกครั้งและกดยืนยันเพื่อลบข้อมูล



การเรียนรู้และการตรวจจับ

เมื่อพบรักษ์ล็อกสีที่เหมือนกันหรือคล้ายกัน กรอบสีบางกรอบที่มี ID จะแสดงบนหน้าจอโดยอัตโนมัติ





การติดตั้งไลบรารีสำหรับบอร์ด POP-32

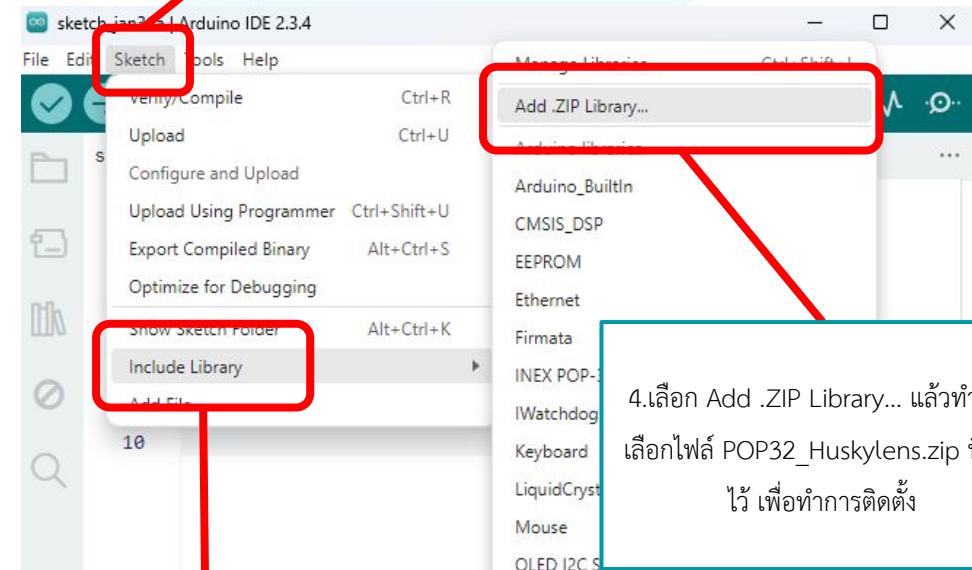


POP32_Huskeylens.zip



1.ดาวน์โหลดไลบรารีได้ที่
[https://inex.co.th/store/software/
POP32_Huskeylens.zip](https://inex.co.th/store/software/POP32_Huskeylens.zip)

2.เลือก Sketch ตรงเมนูบาร์

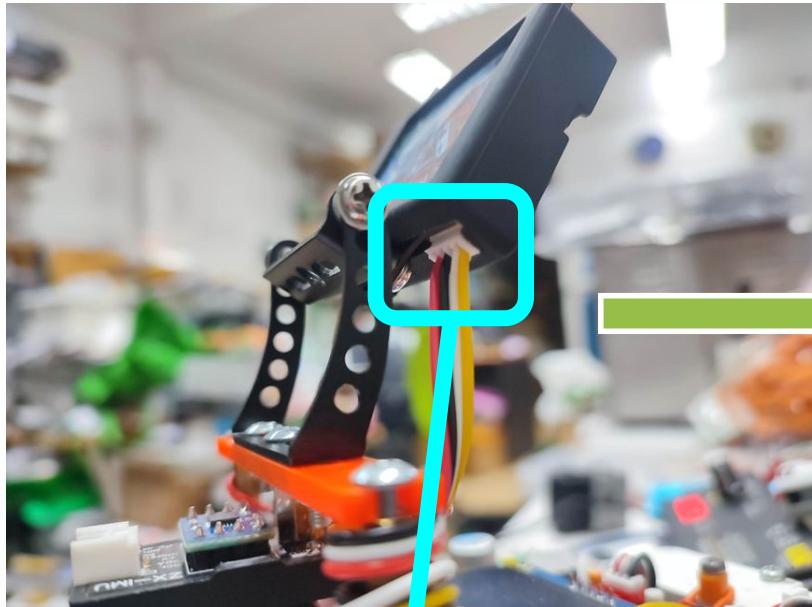


3.เลือก Include Library

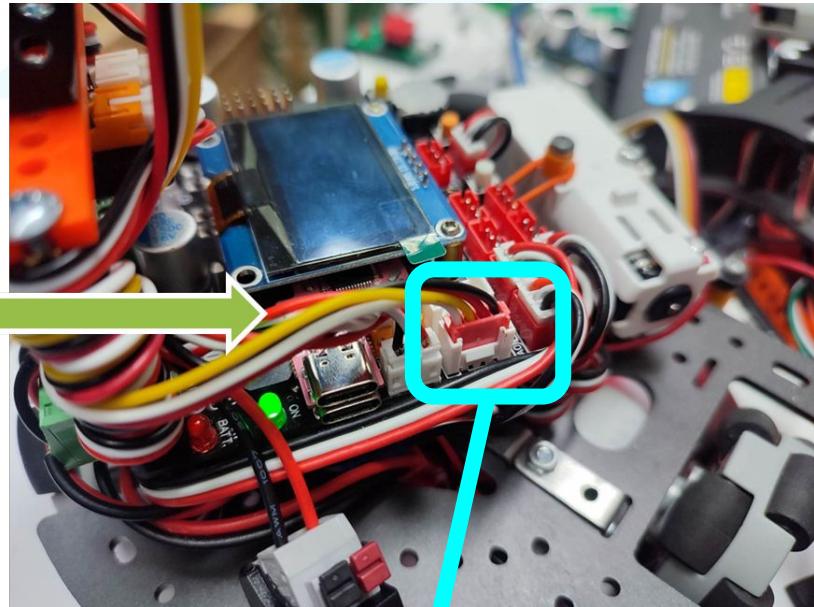
4.เลือก Add .ZIP Library... แล้วทำการ
เลือกไฟล์ POP32_Huskeylens.zip ที่เก็บ
ไว้ เพื่อทำการติดตั้ง



การเชื่อมต่อกล้อง Huskylens กับบอร์ด POP-32



1. ดำเนินการเชื่อมต่อสายที่กล้อง Huskylens



2. ต่อสายที่มาจากการกล้อง Huskylens เข้าที่ดำเนินการ I²C



การ Settings รูปแบบการเชื่อมต่อในกล้อง Huskylens



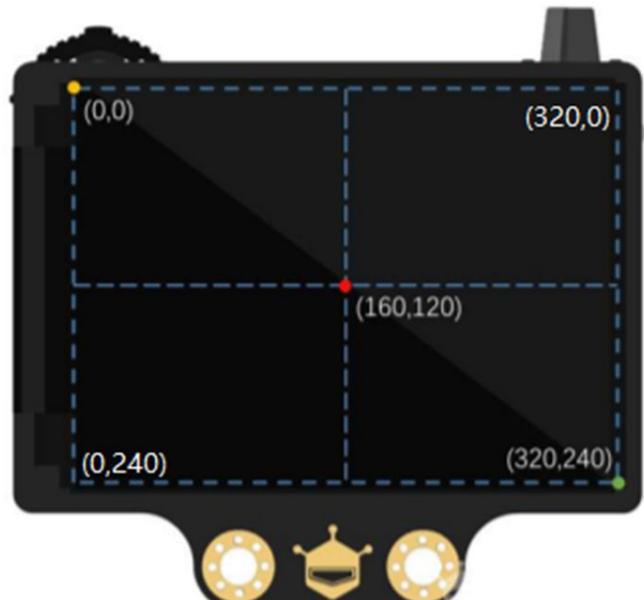


การทำงานของกล้อง Huskylens

Huskylens จะมีผลลัพธ์ 2 รูปแบบ คือ

1.แบบล็อก Block ผลลัพธ์ที่ได้จะให้ค่าตำแหน่งจุดศูนย์กลาง, ความกว้างและความสูง และเลขระบุของวัตถุนั้นๆ โดยจะได้ผลลัพธ์ในแบบนี้เมื่อใช้ mode face recognition, object tracking, object recognition, color recognition, tag recognition และ object classification

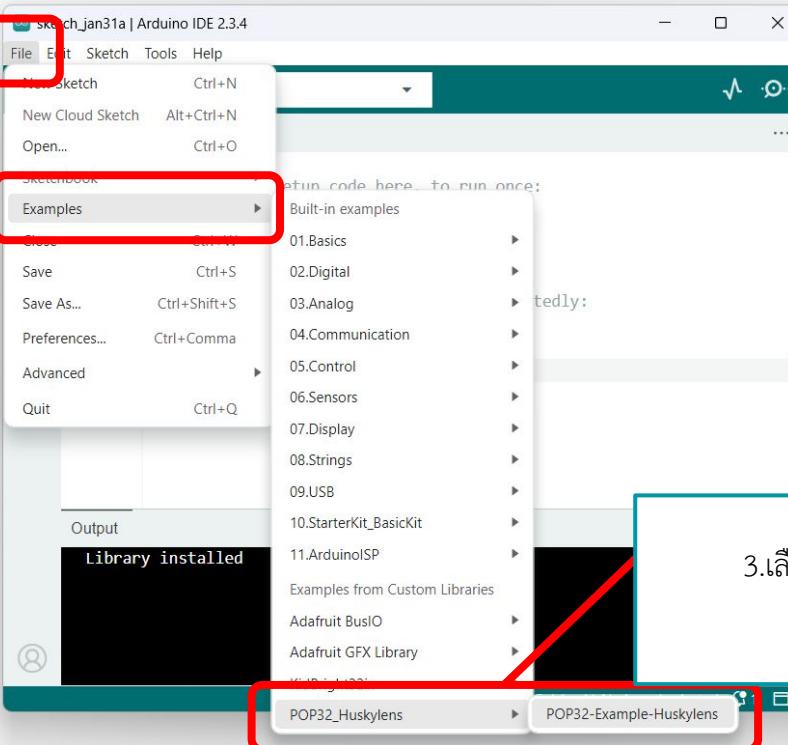
2.แบบลูกศร Arrow ผลลัพธ์ที่ได้จะให้ค่าตำแหน่งจุดเริ่มต้น, จุดสิ้นสุดและเลขระบุประจำลูกศร โดยจะได้ผลลัพธ์ในแบบนี้เมื่อใช้ mode line tracking





ทดสอบไลบรารีสำหรับบอร์ด POP-32

1.เลือก File ตรงเมนูบาร์



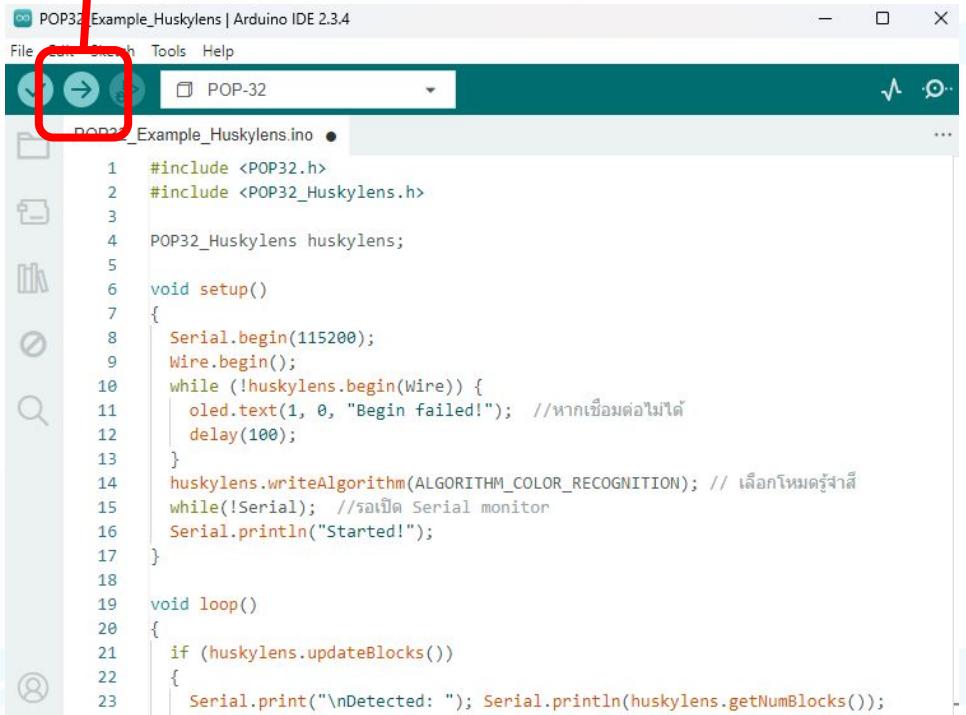
2.เลือก Examples ที่เมนู

3.เลือก POP32_Huskylens แล้วเลือก
POP32-Example-Huskylens



ทดสอบไลบรารีสำหรับบอร์ด POP-32 (ต่อ)

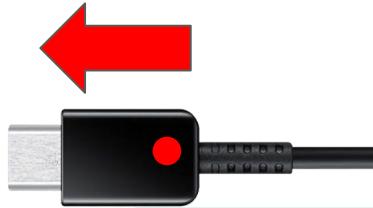
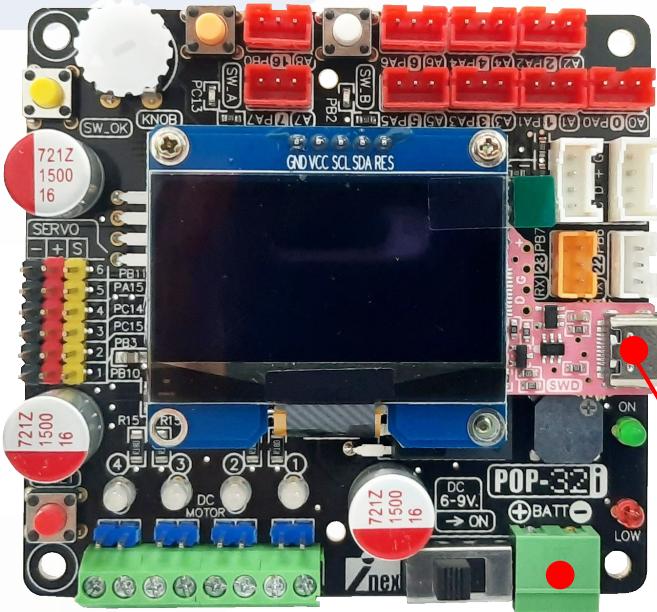
1.Upload โค้ด เพื่อทำการทดสอบ



```
POP32_Husylens | Arduino IDE 2.3.4
File Edit Sketch Tools Help
POP32_Husylens.ino
1 #include <POP32.h>
2 #include <POP32_Husylens.h>
3
4 POP32_Husylens husylens;
5
6 void setup()
7 {
8     Serial.begin(115200);
9     Wire.begin();
10    while (!husylens.begin(Wire)) {
11        oled.text(1, 0, "Begin failed!"); // หากเข้ามายังต่อไม่ได้
12        delay(100);
13    }
14    husylens.writeAlgorithm(ALGORITHM_COLOR_RECOGNITION); // เลือกโหมดรู้จำสี
15    while(!Serial); // รอเปิด Serial monitor
16    Serial.println("Started!");
17}
18
19 void loop()
20{
21    if (husylens.updateBlocks())
22    {
23        Serial.print("\nDetected: "); Serial.println(husylens.getNumBlocks());
```



การเชื่อมต่อบอร์ด POP-32i กับคอมพิวเตอร์



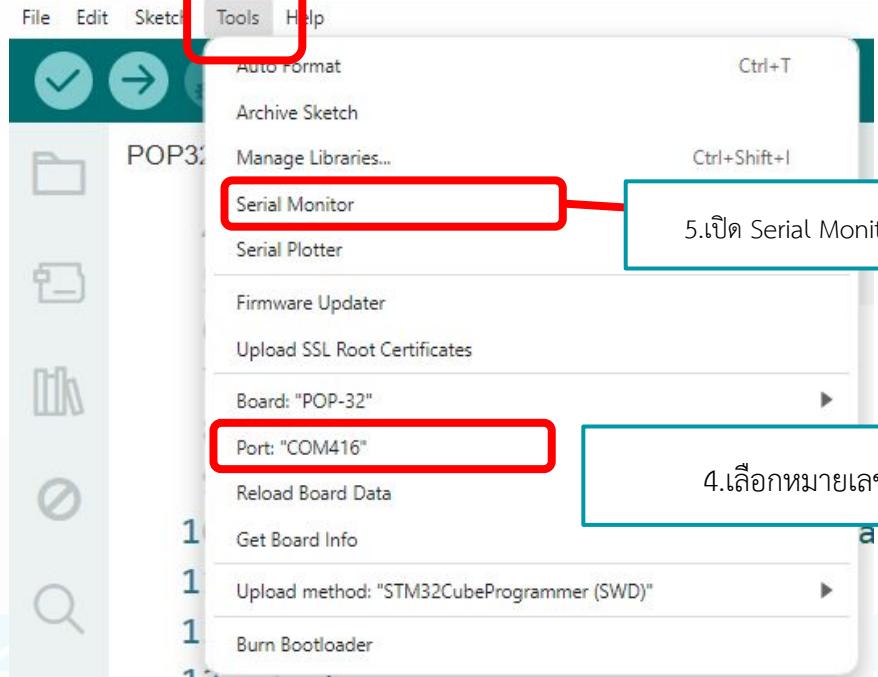
2. ต่อสาย USB เข้าที่ช่อง USB ด้านล่างสุด





ทดสอบไฟบารีสำหรับบอร์ด POP-32 (ต่อ)

3.เลือกเมนู Tools



5.เปิด Serial Monitor เพื่อการตรวจสอบสิ่งที่บันทึกไว้

Not detected
Not detected

ไม่พบสิ่งที่บันทึกไว้

Detected: 1
x y width height
-> Blocks: 1
300 127 37 34

พบสิ่งที่บันทึกไว้

Detected: 1
x y width height
-> Blocks: 1
300 127 37 34

4.เลือกหมายเลข Comport



อธิบายคำสั่งไลบรารีของ Huskylens

```
#include <POP32_Huskylens.h>
```

ผนวกไฟล์ไลบรารี

```
POP32_Huskylens huskylens
```

กำหนดตัวแปร huskylens เพื่อรับคำสั่ง

```
huskylens.begin(Wire)
```

เริ่มต้นใช้งานกล้อง Huskylens

เปิดการจดจำสี

```
huskylens.writeAlgorithm(ALGORITHM_COLOR_RECOGNITION)
```

```
huskylens.updateBlocks()
```

ตรวจสอบบล็อกทั้งหมด

```
huskylens.getNumBlocks()
```

จำนวนบล็อกทั้งหมด



อธิบายคำสั่งไลบรารีของ Huskylens

`huskylens.blockSize[i]`

นับจำนวนบล็อกทั้งหมดของแต่ละคุณสมบัติ

`huskylens.blockInfo[i][j].x`

ตำแหน่ง x ตรงกลางบล็อก

`huskylens.blockInfo[i][j].y`

ตำแหน่ง y ตรงกลางบล็อก

`huskylens.blockInfo[i][j].width`)

ความกว้างของบล็อก

`huskylens.blockInfo[i][j].height)`

ความสูงของบล็อก

i = หมายเลข ID ของบล็อก

j = ลำดับบล็อกในคุณลักษณะเดียวกัน



การสร้างโปรแกรมเพื่อเรียกใช้ฟังก์ชันการเรียนรู้

คำสั่งที่เกี่ยวข้อง

`huskylens.writeLearn(ID)`

กำหนดเลข ID ที่ต้องการจดจำ

`huskylens.writeForget();`

รีเซ็ตการจดจำ



การสร้างโปรแกรมเพื่อเรียกใช้ฟังก์ชันการเรียนรู้ (ต่อ)

```
#include <POP32.h>
#include <POP32_Huskylens.h>

Huskylens huskylens;

void setup()
{
    Serial.begin(115200);
    Wire.begin();
    while (!huskylens.begin(Wire)) {
        oled.text(1, 0, "Begin failed!"); // หากเชื่อมต่อไม่ได้
        delay(100);
    }
    huskylens.writeAlgorithm(ALGORITHM_COLOR_RECOGNITION); // เลือกโหมดรู้จำสี
    while(!Serial); // รอเปิด Serial monitor
    Serial.println("Started!");
}
```

ปรับแต่งในด้านอย่าง code -> Make_Learning.ino

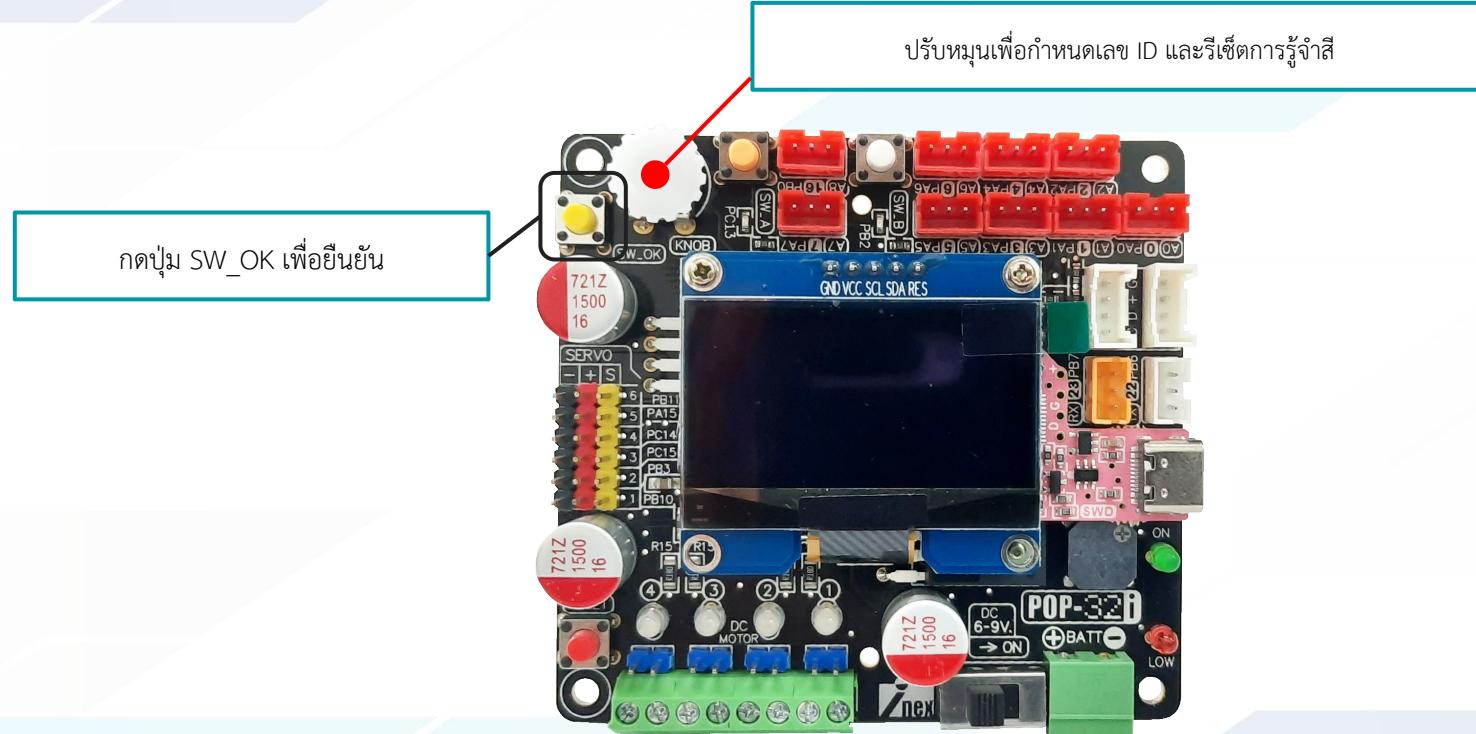


การสร้างโปรแกรมเพื่อเรียกใช้ฟังก์ชันการเรียนรู้ (ต่อ)

```
void loop()
{
    if (huskylens.updateBlocks())
    {
        Serial.print("\nDetected: "); Serial.println(huskylens.getNumBlocks());
        Serial.println("x\ty\twidth\theight");
        for (int i=1; i<=7; i++) // for loop Blocks
        {
            if(huskylens.blockSize[i]) // Blocks not empty
            {
                Serial.print("-> Blocks: "); Serial.println(i);
                for (int j=0; j<huskylens.blockSize[i]; j++) // for loop index of Blocks
                {
                    Serial.print(huskylens.blockInfo[i][j].x);      Serial.print("\t");
                    Serial.print(huskylens.blockInfo[i][j].y);      Serial.print("\t");
                    Serial.print(huskylens.blockInfo[i][j].width);  Serial.print("\t");
                    Serial.println(huskylens.blockInfo[i][j].height);
                }
            }
        }
    }
    else
    {
        Serial.println("Not detected");
    }
}
```



การใช้งาน โปรแกรมรุ๊จASIC





การเคลื่อนที่ติดตามลูกบอล

การเคลื่อนที่ติดตามลูกบอลจะต้องอาศัยการควบคุมความเร็วมอเตอร์ โดยที่

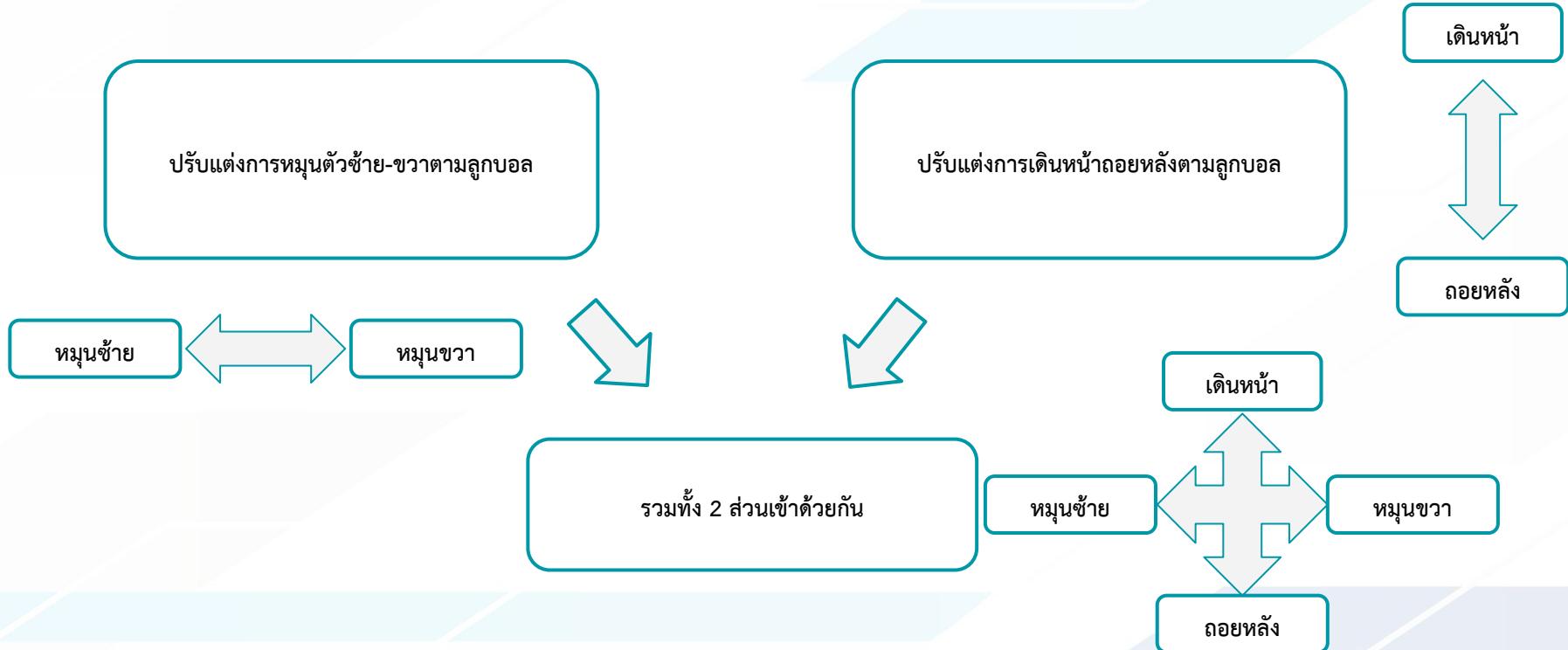
ความเร็วมอเตอร์จะแปรผันตามระยะทางของลูกบอลอย่างรวดเร็ว

มิฉะนั้นจะทำให้ลูกบอลหลุดจากมุ่งมองกล้องได้



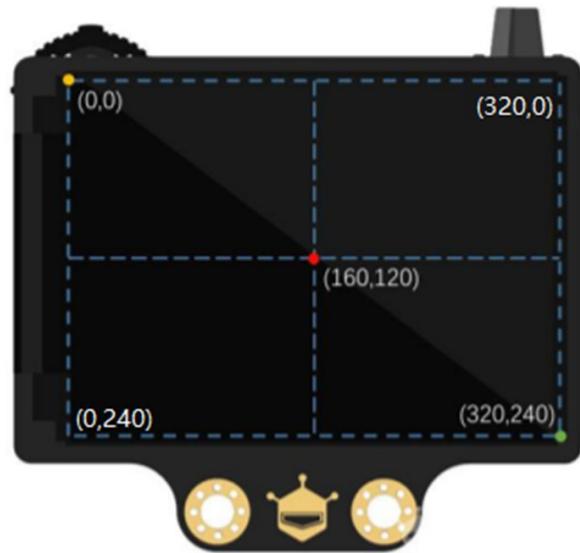


ขั้นตอนการสร้างโปรแกรมเคลื่อนที่ติดตามลูกบอล

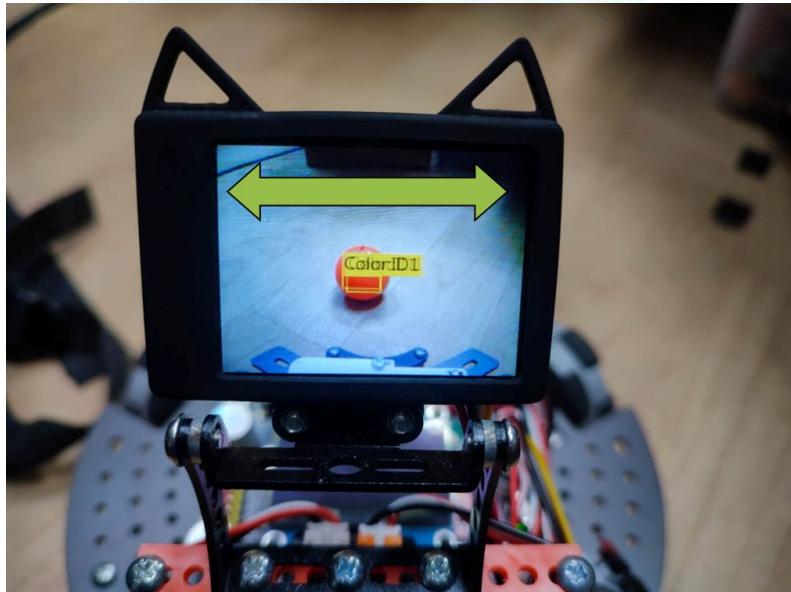




ปรับแต่งการหมุนตัวซ้าย-ขวา ตามลูกบอล



ใช้แกน x มาพิจารณาการปรับทิศ





```
#include <POP32.h>
#include <POP32_Huskylens.h>
POP32_Huskylens huskylens;
#define degToRad 0.0174f
#define sin30 sin(30.f * degToRad)
#define cos30 cos(30.f * degToRad)

// rotate controller
#define rot_Kp 0.6
#define rot_Ki 0.0
#define rot_Kd 0.1
#define rotErrorGap 15 // ค่า Error ที่ยอมให้หุ่นยนต์ทำงาน
#define sp_rot 160 // ค่า setpoint ที่ลูกบอลอยู่ตรงกลางกล้องแกน x 320/2 = 160
float rot_error, rot_pError, rot_i, rot_d, rot_w;
int ballPosX;

void setup() {
    while (!huskylens.begin(Wire)) {
        oled.text(1, 0, "Begin failed!");
        oled.show();
        delay(100);
    }
    waitSW_A_bmp();
```

ปรับแต่งในตัวอย่าง code ->[Test_rotate_controller.ino](#)

เรียกใช้ไลบรารี POP32 และ Huskylens ประกาศตัวแปรและกำหนดค่า

เริ่มต้นใช้งาน Huskylens และรอการกด
ปุ่ม

สร้างโปรแกรมเคลื่อนที่ติดตามลูกบอลปรับทิศหุ่น
ยนต์ หมุน ซ้าย-ขวา



สร้างโปรแกรมเคลื่อนที่ติดตามลูกบอลปรับทิศทุนยนต์ หมุน ซ้าย-ขวา (ต่อ)

```
void loop() {
    if (huskylens.updateBlocks() && huskylens.blockSize[1]) {
        ballPosX = huskylens.blockInfo[1][0].x; // เลือกใช้แกน x ball ID 1 บล็อกแรก
        rot_error = sp_rot - ballPosX;
        rot_i = rot_i + rot_error;
        rot_i = constrain(rot_i, -100, 100);
        rot_d = rot_error - rot_pError;
        rot_w = rot_error * rot_Kp + rot_i * rot_Ki + rot_d * rot_Kd;
        rot_w = constrain(rot_w, -100, 100);
        holonomic(0, 0, rot_w);
        rot_pError = rot_error;
        if (abs(rot_error) < rotErrorGap) {
            holonomic(0, 0, 0);
            beep();
        }
    }
}
```



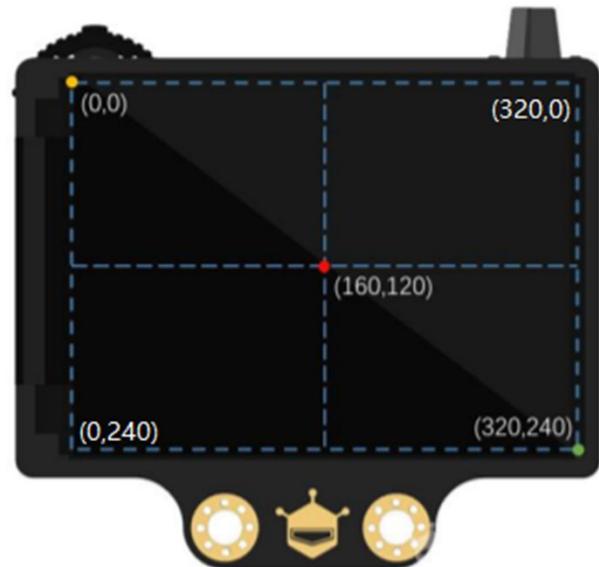
ปรับแต่งค่า เพื่อให้หุ่นยนต์หมุนตัวซ้าย-ขวาไปตามลูกบอล

```
#define rot_Kp 0.1  
#define rot_Ki 0.0  
#define rot_Kd 0.0
```

เริ่มจากเพิ่มค่า Kp ก่อน เพื่อให้เข้าสู่จุด Set point ลูกบอลอยู่ตรงกลางหุ่นยนต์พอดี



ปรับแต่งค่าการเดินหน้า-ถอยหลังตามลูกบอล



ใช้แกน y มาพิจารณา





```
// fling controller
#define fli_Kp 1.5
#define fli_Ki 0.0
#define fli_Kd 0.0
#define flingErrorGap 15 // ค่า Error ที่ยอมให้หุ่นยนต์ทำงาน
float spFli = 120; // ค่า setpoint ที่ยอมให้ลูกบอลอยู่ใกล้หุ่นมากที่สุด อาจเริ่มที่จุดกลางจอ แก้ไขได้
float fli_error, fli_pError, fli_i, fli_d, fli_spd;
int ballPosY;

void setup() {
    while (!huskylens.begin(Wire)) {
        oled.text(1, 0, "Begin failed!");
        oled.show();
        delay(100);
    }
    waitSW_A_bmp();
}
```

ปรับแต่งในตัวอย่าง code ->Test_fling_controller.ino

เรียกใช้ไลบรารี POP32 และ Huskylens ประกาศตัวแปรและกำหนดค่า

สร้างโปรแกรมเคลื่อนที่ติดตามลูกบอลรับระยะ
หุ่นยนต์เข้าใกล้ลูกบอล

เริ่มต้นใช้งาน Huskylens และรอการกดปุ่ม



สร้างโปรแกรมเคลื่อนที่ติดตามลูกบอลปรับระยะหุ้นยนต์เข้าใกล้ลูกบอล

```
void loop() {
    if (huskylens.updateBlocks() && huskylens.blockSize[1]) {
        ballPosY = huskylens.blockInfo[1][0].y; // เลือกใช้แกน y ball ID 1 บล็อกแรก
        fli_error = spFli - ballPosY;
        fli_i = fli_i + fli_error;
        fli_i = constrain(fli_i, -100, 100);
        fli_d = fli_error - fli_pError;
        fli_spd = fli_error * fli_Kp + fli_i * fli_Ki + fli_d * fli_Kd; //รวมเทอม PID
        fli_spd = constrain(fli_spd, -100, 100);
        fli_pError = fli_error;
        holonomic(fli_spd, 90, 0);
        if (abs(fli_error) < flingErrorGap) {
            holonomic(0, 0, 0);
            beep();
        }
    }
}
```



```
#define fli_Kp 1.5  
#define fli_Ki 0.0  
#define fli_Kd 0.0
```

เริ่มจากเพิ่มค่า Kp ก่อน เพื่อให้เข้าสู่จุด Set point



รวมการเคลื่อนที่ในแต่ละแกนเข้าด้วยกัน

ปรับแต่งในตัวอย่าง code ->Test_follow-Ball.ino

```
holonomic(fli_spd, 90, rot_w);
```

โดยการนำค่า fli_spd และ rot_w มาใส่ในฟังก์ชัน holonomic และกำหนดมุมของการเคลื่อนที่ไป 90

องศา เหนืออนกัน



รวมการเคลื่อนที่ในแต่ละแกนเข้าด้วยกัน

```
if((abs(rot_error) < rotErrorGap) && (abs(fli_error) < flingErrorGap)) {  
    wheel(0, 0, 0);  
    beep();  
}
```

เมื่อถึงจุดที่ต้องการ อาจจะหยุดหุ่นยนต์และส่งเสียงดังออกมานะ



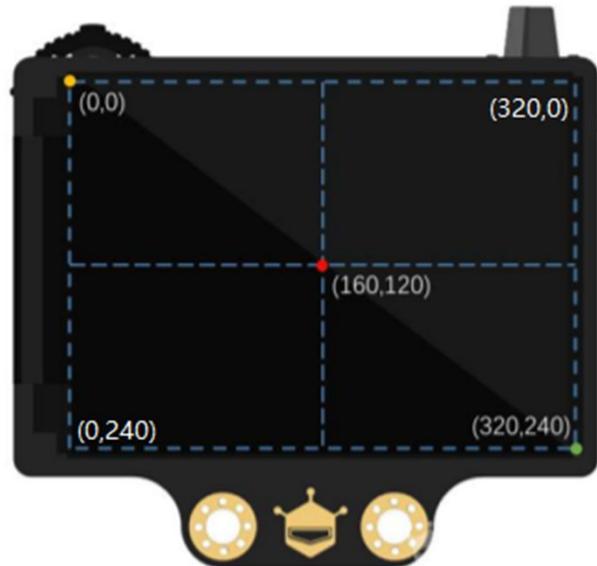
การเคลื่อนที่ปรับทิศหุ่นยนต์ให้ตรงกับทิศอ้างอิง โดยมีลูกบอลเป็นจุดศูนย์กลาง



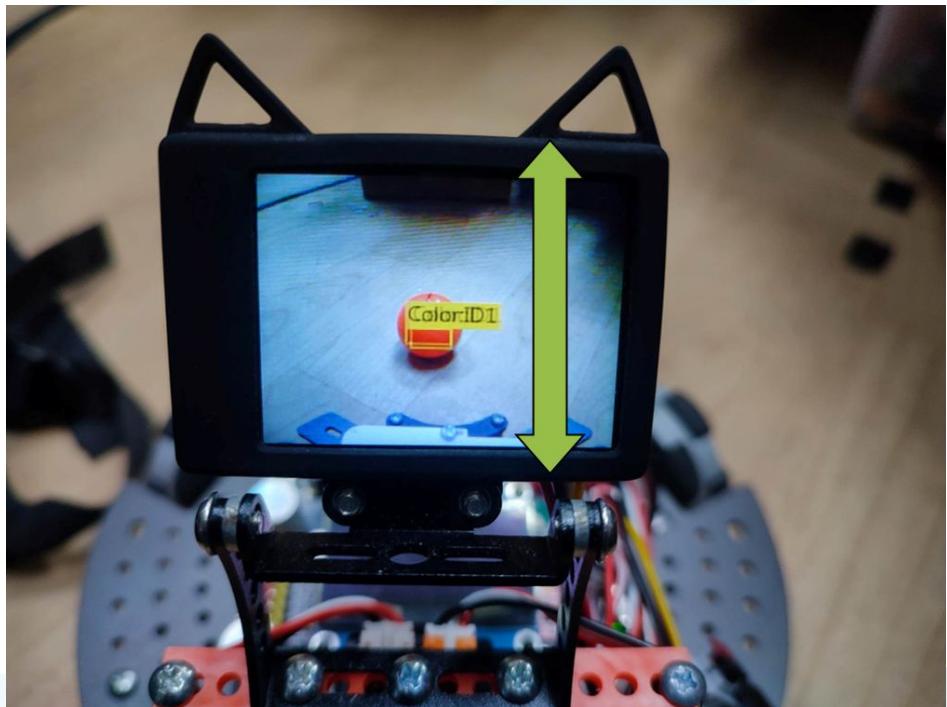
หุ่นยนต์จะเคลื่อนที่มุนรอบลูกบอลให้ตรงกับทิศอ้างอิง เพื่อให้เตรียมความพร้อมในการยิง



ปรับแต่งการหมุนตัวรอบลูกบอล



ใช้แกน y มาพิจารณา





ปรับแต่งในตัวอย่าง code ->Test_align_pre_shoot.ino

```
// align controller
#define ali_Kp 2.75
#define ali_Kd 0.0
#define alignErrorGap 4
float spFli = 130;
float ali_error, ali_pError, ali_d, ali_vec, lastYaw, vecCurve, radCurve;
// overall
int ballPosY, discoveState;
```

```
void setup() {
    while (!huskylens.begin(Wire)) {
        oled.text(1, 0, "Begin failed!");oled.show();
    }
    zeroYaw(); // รีเซ็ต zero
    oled.text(4, 0, "SW_B => RUN");
    while (!SW_B()) {
        getIMU();
        oled.text(0, 0, "Yaw=%f      ", pvYaw);
        oled.show();
    }
}
```

เรียกใช้ไลบรารี POP32 และ Huskylens ประการตัวแปรและกำหนดค่า

การเคลื่อนที่ปรับทิศหุ่นยนต์ให้ตรงกับทิศอ้างอิง
โดยมีลูกบอลเป็นจุดศูนย์กลาง

เริ่มต้นใช้งาน Huskylens และรอการกดปุ่ม B เริ่มทำงาน



```
void loop() {
    getIMU();
    if (huskylens.updateBlocks() && huskylens.blockSize[1]) {
        ballPosY = huskylens.blockInfo[1][0].y;
        ballPosX = huskylens.blockInfo[1][0].x;
        ali_error = ballPosY - spFli;
        ali_d = ali_error - ali_pError;
        ali_vec = ali_error * ali_Kp + ali_d * ali_Kd;
        ali_pError = ali_error;
        // หุ่นเลือกทิศทางที่ใกล้ที่สุด ที่จะปรับท้ายหุ่นหากลับbol
        if (lastYaw > 0) {
            vecCurve = 180;
            radCurve = -15;
        } else {
            vecCurve = 0;
            radCurve = 15;
        }
        holonomic(40, vecCurve, radCurve);
        if (abs(pvYaw) < alignErrorGap) { //เมื่อทิศอยู่ในค่าที่รับได้
            holonomic(0, 0, 0); //หยุดทำงาน
            beep();
        }
    }
    lastYaw = pvYaw;
}
```

การเคลื่อนที่ปรับทิศหุ่นยนต์ให้ตรงกับทิศอ้างอิง
โดยมีลูกบอลเป็นจุดศูนย์กลาง



รวมการเคลื่อนที่ ติดตามลูกบอลและการปรับทิศเข้าด้วยกัน

```
if (huskylens.updateBlocks() && huskylens.blockSize[1]) {  
    if(discoveState) {  
        .....กลุ่มค่าสั่งเคลื่อนที่ติดตามลูกบอล.....  
        if(setpoint) {  
            discoveState = 0;  
        }  
    } else {  
        .....กลุ่มค่าสั่งเคลื่อนที่ปรับทิศหุนยนต์.....  
        if(setpoint) {  
            discoveState = 1;  
        }  
    }  
} else {  
    .....กลุ่มค่าสั่งเคลื่อนที่หาลูกบอล.....  
}
```

ปรับแต่งในตัวอย่าง code ->Test_align_fling_controller.ino

discoveState จะเป็นตัวกำหนดที่เลือกการเคลื่อนที่



ชุดคำสั่งเคลื่อนที่หาลูกบอล

```
int sideRot = rot_error;      // หากติดทางล่างสุดที่กล้องตรวจพบ  
holonomic(0, 0, sideRot/abs(sideRot) * idleSpd);    // ความเร็วที่ต้องการหมุนรอบตัว
```

ใช้การหมุนรอบตัว เพื่อหาลูกบอล



```
if (huskylens.updateBlocks() && huskylens.blockSize[1]) {  
    if(discoveState){  
        .....กลุ่มค่าสั้งเคลื่อนที่ติดตามลูกบอล.....  
        if(setpoint){  
            discoveState=0;  
        }  
    } else {  
        .....กลุ่มค่าสั้งเคลื่อนที่ปรับทิศทุนยนต์.....  
        if(setpoint){  
            discoveState=1;  
            .....กลุ่มค่าสั้งชุดค่าสั้งตรวจสอบก่อนยิง.....  
        }  
    }  
} else {  
    .....กลุ่มค่าสั้งเคลื่อนที่หาลูกบอล.....  
}
```

ชุดคำสั่งตรวจสอบก่อนยิง



ชุดคำสั่งตรวจสอบก่อนยิง

เมื่อเคลื่อนที่ปรับทิศให้ตรงเรียบร้อยแล้ว จากนั้นจะเป็นขั้นตอนการเตรียมยิงลูกบอล

ตรวจสอบค่า Error ทั้งระยะห่างและทิศทางของลูกบอล

ค่ากึ่งกลางในแนวแกน X

```
if ((abs(150 - ballPosX) < rotErrorGap) && (abs(spFli - ballPosY) < flingErrorGap)) {  
    beep();  
    unsigned long loopTimer = millis();  
    while(1){      // เดินหน้าตรง เต็มความเร็ว นาน 0.25 วินาที  
        getIMU();heading(100, 90, 0);  
        if (millis()-loopTimer >= 250)break;  
    }  
    shoot();    // ทำการยิงและเก็บ  
    reload();  
}
```



ชุดคำสั่งฟังก์ชันทั้งหมดที่สร้างขึ้นมาใช้งาน

```
void zeroYaw()  
void getIMU()
```

เกี่ยวกับปรับทิศใช้ร่วมกับ ZX-IMU

```
void shoot()  
void reload()
```

เกี่ยวกับการยิงและเก็บก้านยิง

```
void wheel(int s1, int s2, int s3)  
void holonomic(float spd, float theta, float omega)  
void heading(float spd, float theta, float spYaw)
```

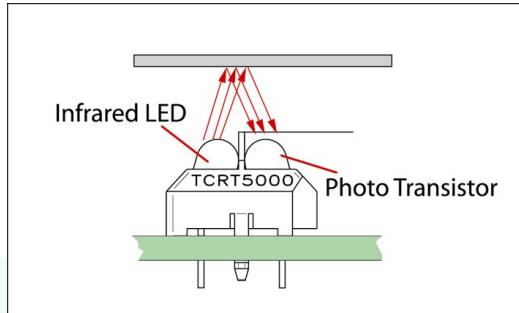
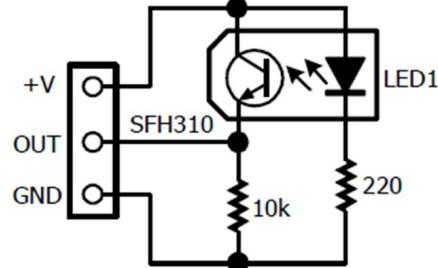
เกี่ยวกับการเคลื่อนที่

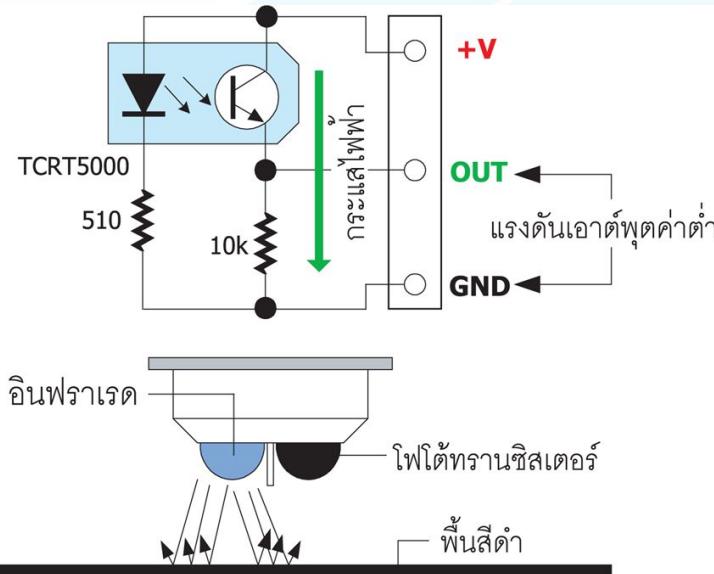
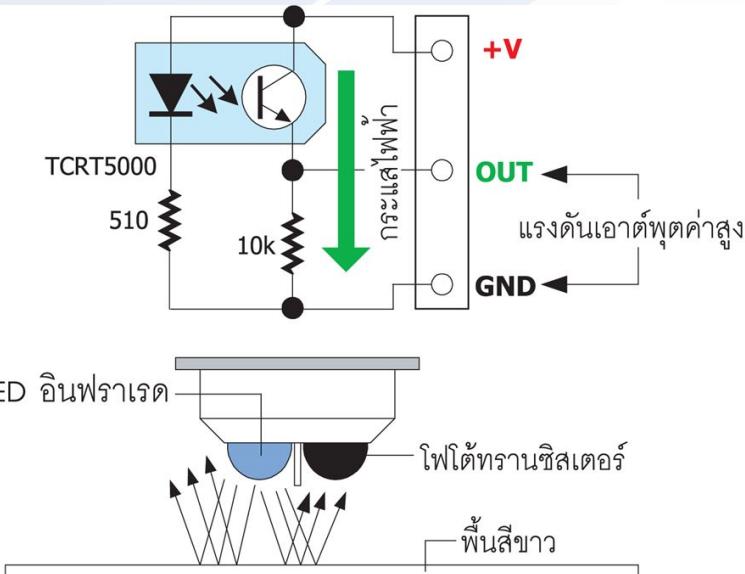


มินิบอร์ดวงจรตรวจจับแสงสะท้อนอินฟารेड

วงจรตรวจจับแสงสะท้อนอินฟารेडจะประกอบไปด้วย LED อินฟารेडเป็นตัวส่งแสงอินฟารेडและมีโพโต้ทรานซิสเตอร์เป็นตัวรับแสงอินฟารेडที่สะท้อนกลับมา

ปริมาณของแสงที่สะท้อนกลับมาขึ้นอยู่กับวัตถุที่มา กด ขวาง ว่าสามารถสะท้อนแสงอินฟารेडได้ดีขนาดไหน (พื้นสีขาวสะท้อนแสงดีที่สุด พื้นสีดำสะท้อนแสงได้น้อยที่สุด)
ถ้าไฟโพโต้ทรานซิสเตอร์ได้รับแสงมากจะทำให้ที่จุดเอาร์พุตจะมีแรงดันเพิ่มขึ้นไปด้วย สำหรับค่าที่อ่านได้จะอยู่ในช่วง 0-4000





รูปแสดงการสะท้อนแสงของแสงอินฟราเรด





ตัวอย่างโค้ดควบคุมการอ่านค่าแสงของ ZX-03

```
#include <POP32.h>

void setup() {

}

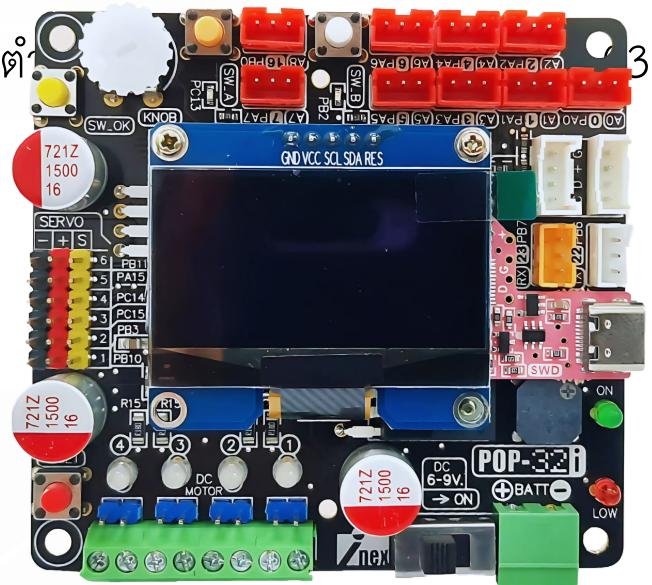
void loop() {
    oled.text(0, 0, "C=%d ", analog(1));
    oled.text(1, 0, "R=%d ", analog(2));
    oled.text(2, 0, "L=%d ", analog(3));
    oled.show();
}
```



แสดงค่าแสงของ ZX-03 ตัวที่ 1 ที่บรรทัดที่ 1

แสดงค่าแสงของ ZX-03 ตัวที่ 2 ที่บรรทัดที่ 2

แสดงค่าแสงของ ZX-03 ตัวที่ 3 ที่บรรทัดที่ 3



พอร์ต A0

พอร์ต A1

พอร์ต A2



ZX-03 ตัวที่ 1



ZX-03 ตัวที่ 2



ZX-03 ตัวที่ 3



ตัวอย่างการตรวจสอบการสะท้อนแสงของ ZX-03

สีพื้น	ZX-03 ตัวที่ 1	ZX-03 ตัวที่ 2	ZX-03 ตัวที่ 3
ยกจากพื้น	0	0	0
สีขาว	3924	3527	3854
สีดำ	671	429	565
ค่าเฉลี่ย (สีขาว+สีดำ)/2	2298	1798	2210

หมายเหตุ : นำค่าเฉลี่ยที่ได้ไปใช้ในการเขียนโค้ดควบคุมเพื่อเปรียบเทียบพื้นสนามกับเส้นขอบสนาม