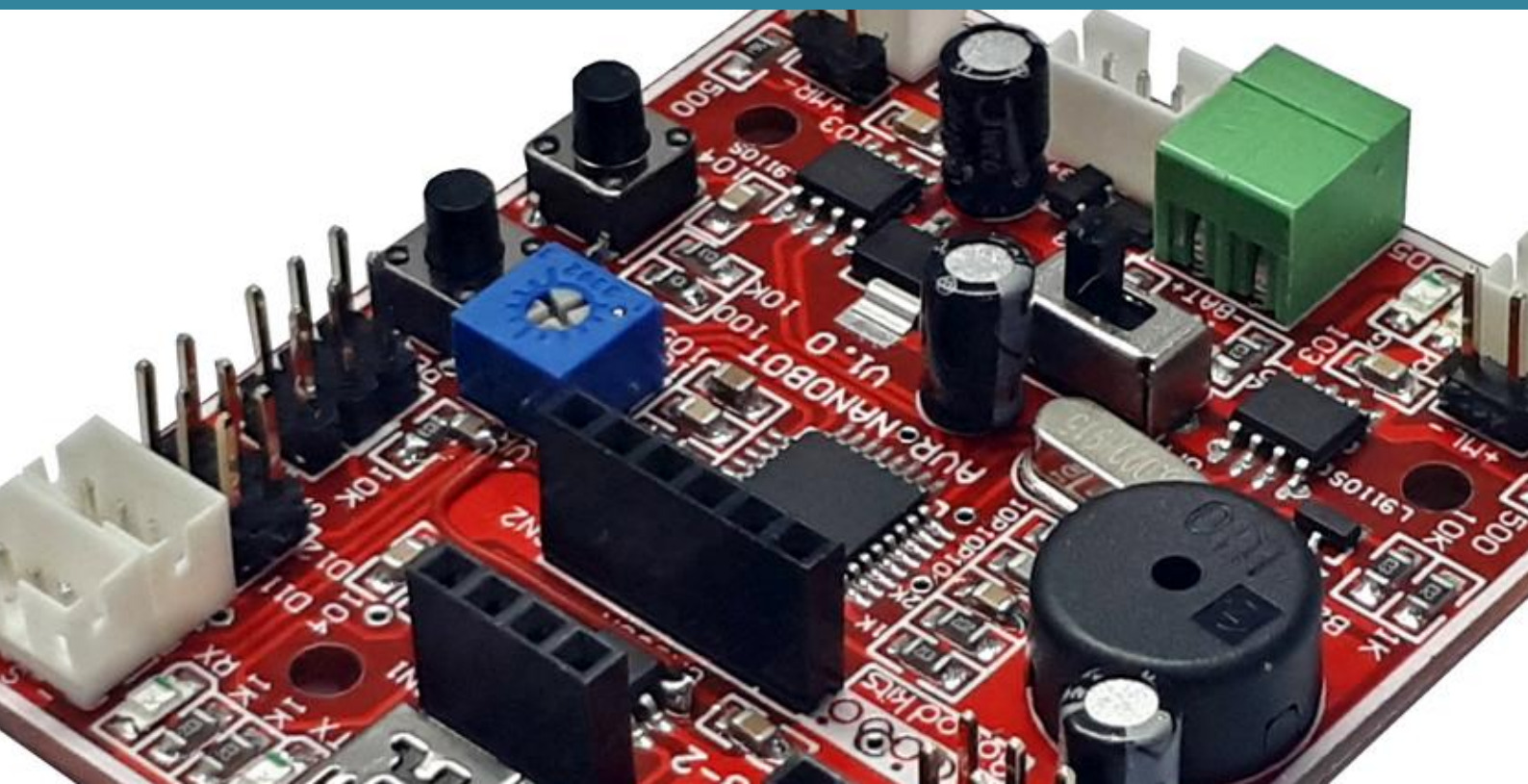
Robot kits

เหมาะสำหรับนักเรียน นักศึกษา และผู้ที่สนใจทั่วไป

AVR NANOBOT

รายละเอียด	
ไมโครคอนโทรลเลอร์	ATmega328
แหล่งจ่ายไฟ	AA 1.5Vx4 มาตรฐาน
Connector Power 2 Pin	6 – 9 Vdc
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16Mhz
Motor Drive on board	2 ตัว
Sensor Infrared	2 ตัว เพิ่มได้อีก 2
รองรับการต่อ Servo	2 ตัว ไม่มีในชุด
Buzzer on board	1 ตัว
A to D test on board	1 ตัว
Switch Test on board	1 ตัว
Switch Reset on board	1 ตัว
รองรับการต่อ Ultrasonic	1 ตัว
รองรับการต่อ Bluetooth	1 ตัว
รองรับการต่อ I2C LCD Display	1 ตัว ไม่มีในชุด



คำนำ

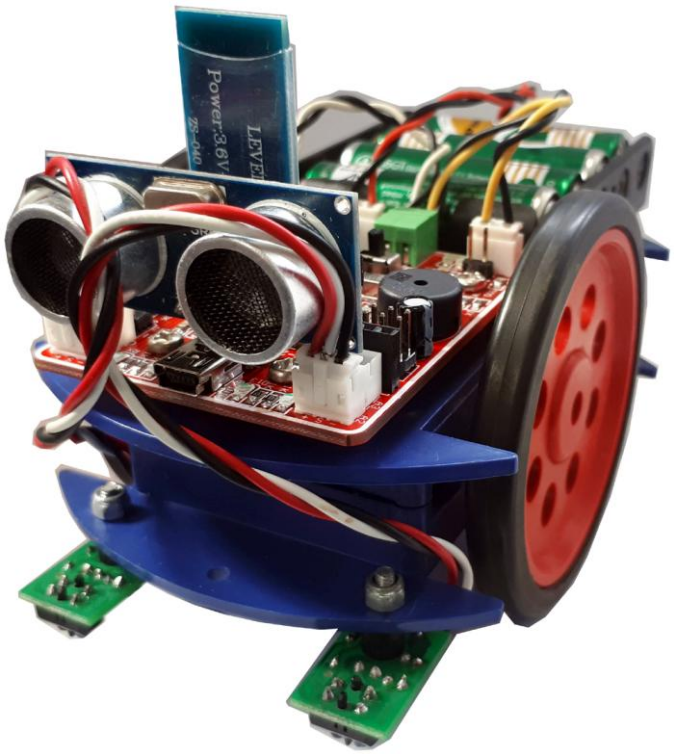
จากประสบการที่ผ่านมาทางทีมงาน www.robodkit.com เห็นว่า การใช้งาน MCU ได้แพร่หลายมากขึ้น เดิมทีจะมีระดับที่ใช้งานอยู่ที่ Engenia หรือระดับผู้ออกแบบ และระดับนักศึกษา วิชาชีพ ซึ่งในสมัยก่อนการใช้งาน MCU ค่อนข้างยาก และมีตัวเลือกน้อยแถมราคาก็แพงมาก อีกทั้งการข้นหาข้อมูลก็ยากมาก สำหรับผู้ที่สนใจก็จะหาหนังสือมาอ่านศึกษาข้อมูลกันเอง รู้บ้าง ไม่รู้บ้าง เขียนถูก เขียนผิด จะไปสอบถามใครก็ยาก ไม่ใช่ว่าไม่อยากถามหรือไม่มีคนให้ถาม แต่เราจะตั้งคำถามอย่างไรให้เขาเข้าใจและเราก็ต้องเข้าใจด้วย การเริ่มต้นในสมัยนั้นก็เลยดูมีอุปสรรคอยู่ไม่ใช้น้อย

แต่มาปัจจุบัน MCU มีให้เลือกใช้งานมากมาย และราคาก็ถูกลงมาก การศึกษาหาความรู้ก็มีให้ค้นหามากมาย มีโค้ดตัวอย่างให้ทดลอง สามารถหาได้จาก อากู๋ หรือ Google ง่ายแสนง่าย มีทั้ง YouTube ให้เห็นทั้งภาพและเสียงจึงทำให้ง่ายต่อการศึกษาใช้งาน

MCU เป็นอุปกรณ์อิเลคทรอนิคส์ที่ถูกผลิตขึ้นมา โดยมีโครงสร้างที่สลับซับซ้อน รวมไว้ในตัว MCU เพียงตัวเดียว การเขียนคำสั่งควบคุมก็ไม่ยากเหมือนก่อน แต่เดิมต้องเขียนด้วยคำสั่ง ภาษาแอสแซมบลี้ (Assembly) ซึ่งเป็นคำสั่งภาษาของ MCU ที่ผู้ผลิตได้กำหนดขึ้นมา แต่ตอนนี้ผู้พัฒนาได้ทำให้การเขียนง่ายขึ้นโดยใช้ ภาษาซี (C) ซึ่งเป็นภาษาคอมพิวเตอร์ ถึงแม้ว่าภาษา C มีความสามารถในระดับ Low-level แต่มันยังถูกออกแบบเพื่อช่วยให้สามารถเขียนโปรแกรมแบบ Cross-platform โค้ดของโปรแกรมที่เขียนขึ้นจากมาตรฐานของภาษา C นั้นสามารถนำไปคอมไพล์ได้ในคอมพิวเตอร์ในแพลตฟอร์มและระบบปฏิบัติการที่หลากหลายโดยเพียงแค่เปลี่ยนแปลงโค้ดเพียงเล็กน้อย ภาษา C นั้นสามารถใช้ได้อย่างกว้างขวางในแพลตฟอร์มขนาดต่างๆ ตั้งแต่ Embedded microcontrollers ไปจนถึง Supercomputer

Arduino เป็นโครงการพัฒนาระบบ MCU ตระกูล AVR ด้วยภาษา C++ แบบ Open Source ที่ได้รับการเผยแพร่มาตั้งแต่ปี ค.ศ. 2005 และเป็นที่รู้จักกันทั่วโลกและแพร่หลายมากขึ้นและรวดเร็วดูได้จากจำนวนสมาชิกผู้ใช้ที่หลากหลายมีตั้งแต่ระดับ ผู้เริ่มต้น ผู้ที่สนใจทั่วไป นักศึกษา จนถึงระดับนักพัฒนา จึงทำให้เกิดความสนใจขึ้นจากหลากหลายวงการ และช่วยกันพัฒนาและปรับปรุงซอร์ฟแวร์ และ ฮาร์ดแวร์ ไปจนถึงอุปกรณ์เชื่อมต่ออีกมากมาย เพื่อนำมาใช้งานร่วมกันกับโปรแกรม Arduino รวมถึงโค้ดตัวอย่างในการทดลองและการประยุกต์การใช้งานในรูปแบบต่างๆ อีกมากมาย ดังนั้นจึงเป็นที่สนใจของบุคลต่างๆที่อยากจะศึกษาระบบ MCU ด้วย Arduino กันอย่างแพร่หลายมากขึ้น

ทั้งนี้ทางทีมงานจึงได้ออกแบบบอร์ด AVR NANOBOT ขึ้นมาเพื่อให้ตอบโจทย์ผู้ที่เริ่มต้นศึกษาเขียนโปรแกรมภาษา C ให้ทำงานได้ง่ายขึ้นโดยบนบอร์ดจะรวมการทดลองขั้นพื้นฐานของอุปกรณ์ต่างๆไว้ให้ทดลองใช้งาน และมีจุดต่อเสริมสำหรับการทดลองอื่นๆอีก เช่น บอร์ดอุลตร้าโซนิกส์ บอร์ดดิสเพล LCD มอเตอร์เซอร์โว หรือ Bluetooth เป็นต้น สำหรับผู้ที่ยังไม่มีพื้นฐาน ยังไม่รู้ว่าจะต้องใช้บอร์ดอะไร แบบไหน มาทดลอง เพื่อเรียนรู้การเขียนโปรแกรม จึงขอแนะนำ AVR NANOBOT ให้เป็นทางเลือกสำหรับผู้ที่สนใจไว้พิจารณา



ทีมงาน

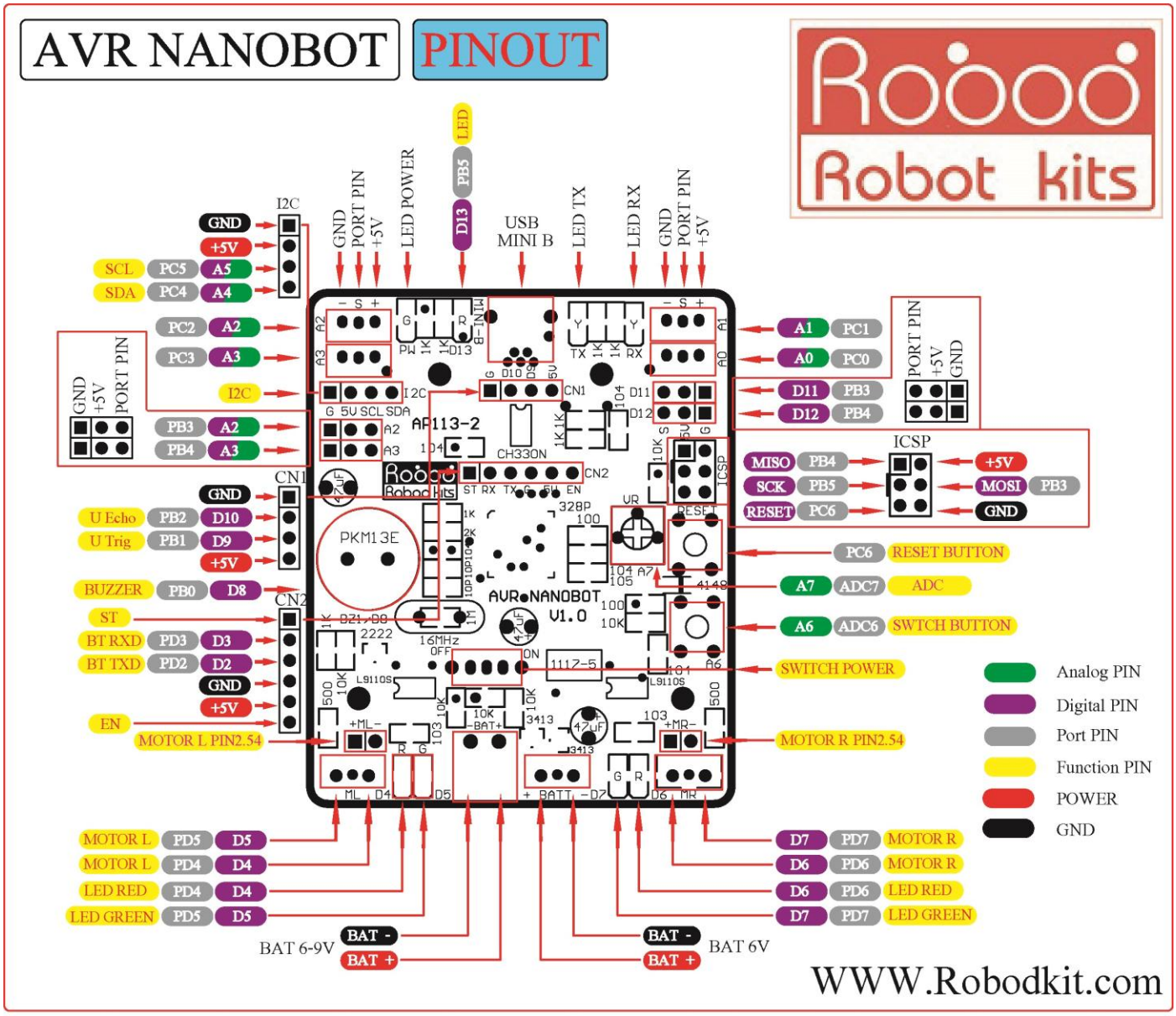
www.robodkit.com

สารบัญ

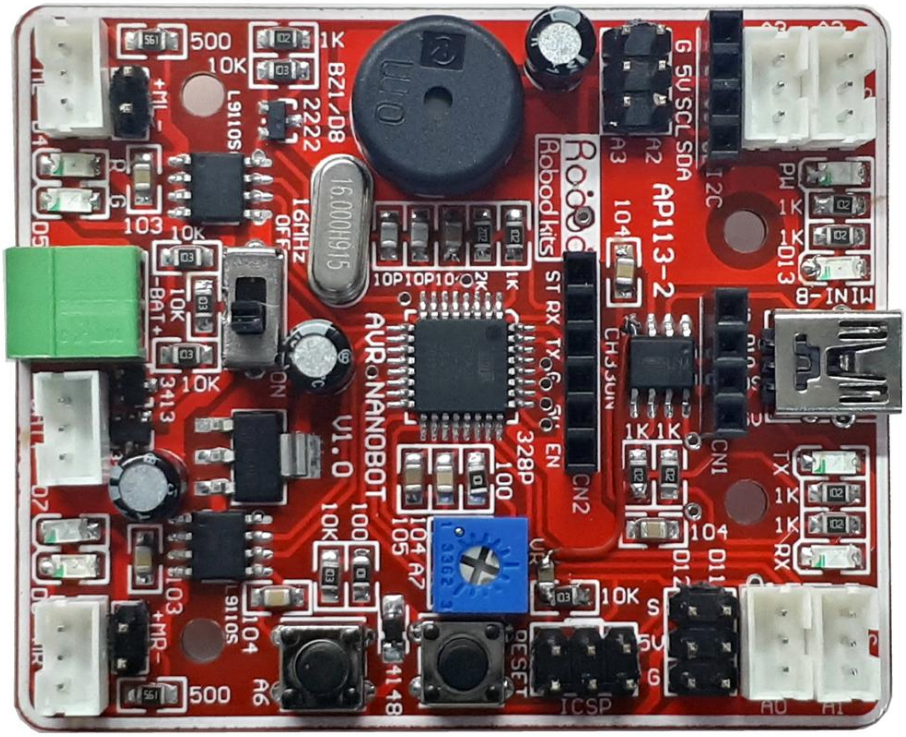
ขั้นตอนการเขียน Program และ การทดสอบหุ่นยนต์

แนะนำบอร์ด AVR NANOBOT

AVR NANOBOT เป็นบอร์ดไมโครคอนโทรเลอร์ในตระกูล AVRได้ถูกพัฒนาต่อจาก AVR BuilIn Programmer ตัวก่อน รุ่นนี้จะเปลี่ยนจากชิพ Atmega8 มาเป็น Atemga328P ที่มีหน่วยความจำมากขึ้นขนาด 32KBytes 1KBytes EEPROM และ 2KBytes Internal SRAM ซึ่งเหมาะสำหรับการเขียนโปรแกรมที่มากขึ้น และตัวบอร์ดยังออกแบบจุดต่อให้สามารถต่อกับอุปกรณ์ได้หลากหลาย เช่น เซ็นเซอร์แบบอินฟราเรด บอร์ดอัลตร้าโซนิค บอร์ดบลูทูธ หรือจะต่อกับ เซอร์โวได้หลายตัวพร้อมกันเพื่อทำมือจับสิ่งของ และยังมีจุดต่อแบบ I2C ที่จะต่อได้กับ จอ LCD บนบอร์ดยังต่อวงจรขับมอเตอร์มาให้สองตัวเหมือนบอร์ดเดิมแต่วงจรขับมาใช้ IC L9110 จึงทำให้ควบคุมมอเตอร์ได้ดีกว่า และจ่ายกระแสได้สูงกว่าบอร์ดเดิม

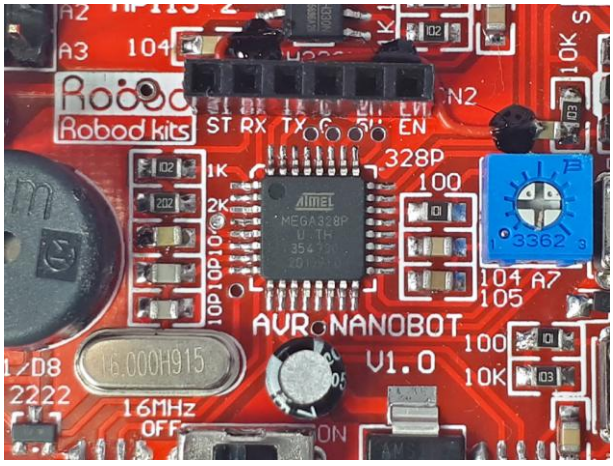


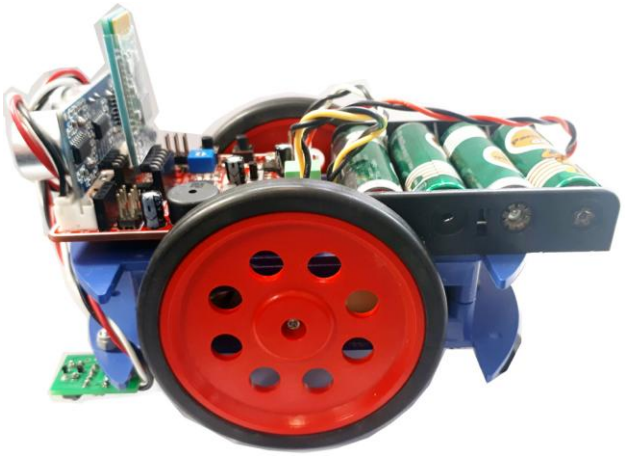
รูปแสดง ตำแหน่งขาใช้งานของ AVR NANOBOT



รูปแสดงอุปกรณ์ต่างๆบนบอร์ด AVR NANOBOT

AVR NANOBOT ใหม่นี้จะเขียนโปรแกรมบน Arduino IDE และอ้างอิงบอร์ด Arduino NANO โดยใช้ bootloader ในรุ่น NANO และอุปกรณ์ต่างๆที่นำมาติดตั้งก็สามารถใช้โค้ดตัวอย่างที่มีอยู่บน Arduino IDE ได้เลยเพียงบางตัวอาจจะเปลี่ยนขาสัญญาณให้ ตรงกับตัวบอร์ดเท่านั้น หรือจะทดลองโค้ดตามที่ทีมงานได้เขียนไว้ให้ก็ได้เพราะตัวโค้ดได้อธิบายการเขียนไว้ให้เพื่อที่จะเข้าใจได้ง่ายขึ้น ก่อนที่เราจะเขียนโค้ดเราต้องเตรียมอะไรบ้างมาดูกัน

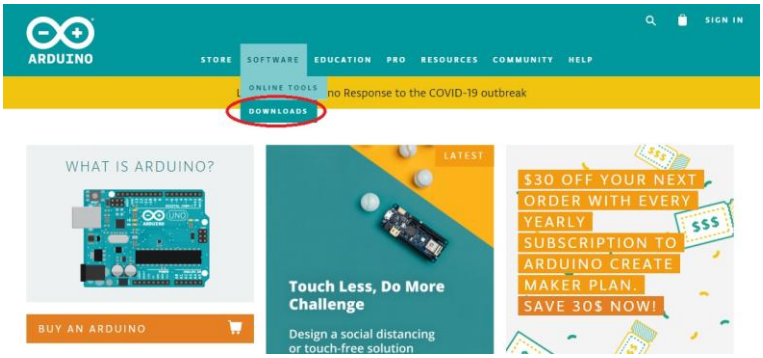




รูปแสดง ชิพไอซีเบอร์ Atmega328P และ ตัวหุ่นยนต์ที่ประกอบแล้ว

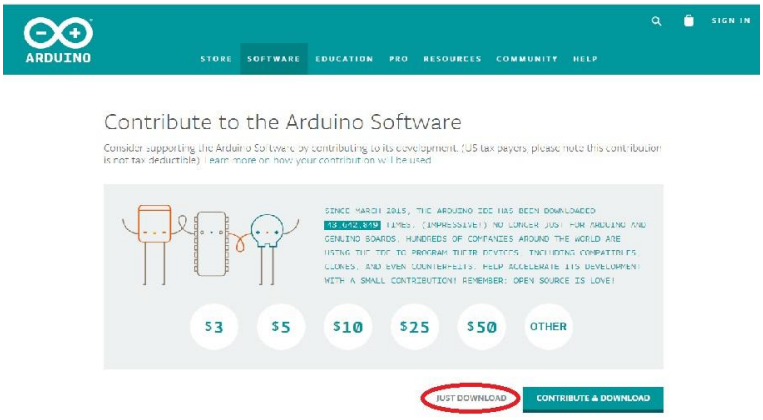
ติดตั้ง Arduino IDE

ในการใช้งานตัวบอร์ดนั้นก็ไม่ยุ่งยากเพียงแค่ติดตั้งโปรแกรม Arduino IDE ไปที่หน้าเว็บตามลิงค์นี้ https://www.arduino.cc ไปที่เมนูด้านบน SOFTWARE และเลือกที่ Download



เมื่อเข้าไปที่หน้าเว็บให้เลือก ระบบปฏิบัติการ windows ของเรา

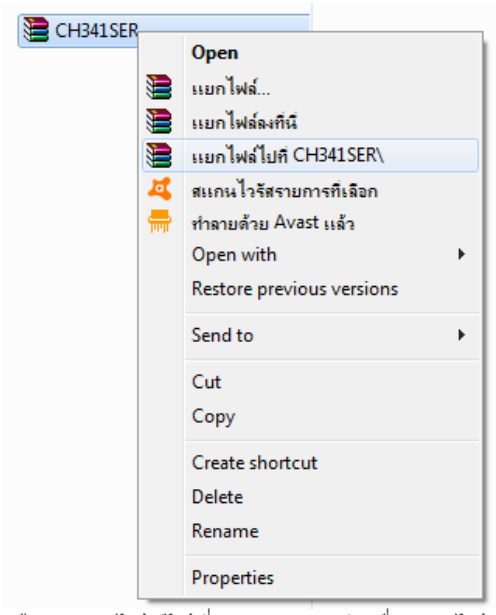
จากนั้นที่หน้าต่าง Contribute to the Arduino Software ให้เลือก JUST DOWNLOAD หรือในกรณีที่เราต้องการบริจาคเพื่อช่วยการพัฒนา Arduino Software สามารถกดเลือกจำนวนเงินเป็นเหรียญตามที่ต้องการแล้วกด CONTRIBUTE & DOWNLOAD เพื่อร่วมบริจาคได้



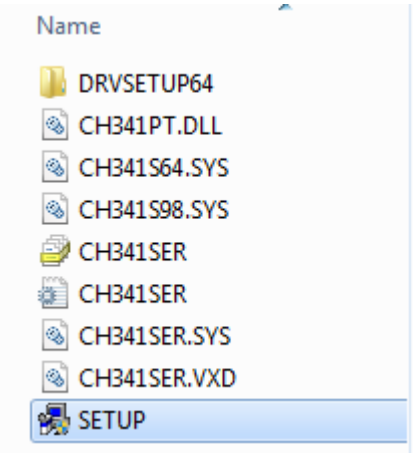
หรือจะเข้าไปดูวิธีการติดตั้งได้ที่ https://www.robodkit.com ที่เมนูด้านบนขวามือเลือก Arduino เลือก บทความ Arduino เลือก แนะนำการติดตั้ง Arduino IDE

ติดตั้ง Driver USB

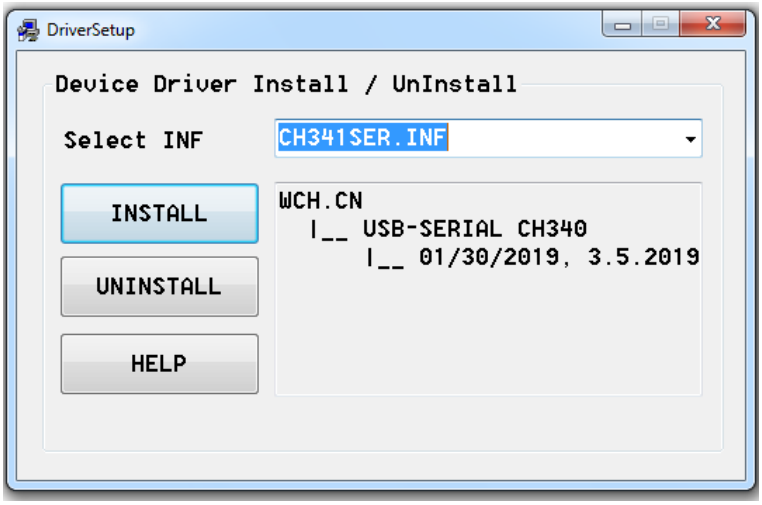
ให้ติดตั้งไดรเวอร์ USB โดยเข้าไปที่ไดร์ฟ CD แตกไฟล์ที่ชื่อ CH341SER.ZIP หรือถ้าไฟล์แตกมาแล้วให้ข้ามไปขั้นตอนต่อไป



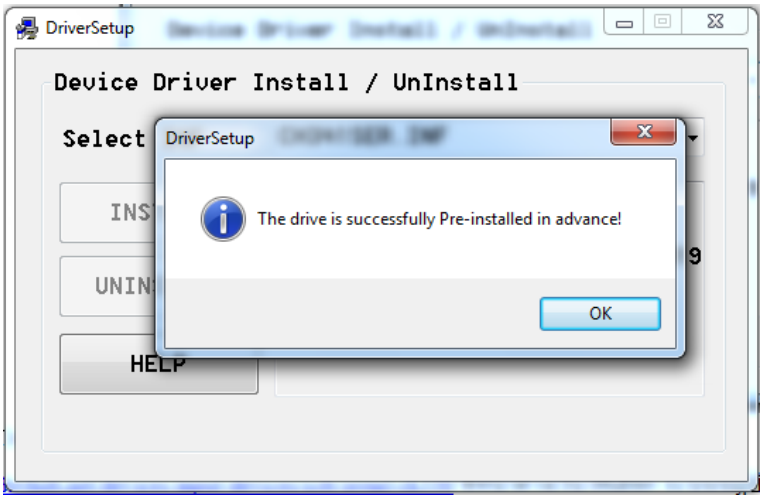
ให้คลิกขวาที่ไฟล์ CH341SER จากนั้นเลือก แยกไฟล์ไปที่ CH341SER\ เมื่อแตกไฟล์เสร็จก็จะได้โฟลเดอร์ CH341SER ขึ้นมา ให้ดับเบิลคลิกที่โฟลเดอร์



ดับเบิลคลิกอีกครั้งที่ไฟล์ SETUP จะขึ้นหน้าต่าง User Account Control ให้ตอบ Yes จากนั้นหน้าต่าง DriverSetup จะแสดงขึ้นมา



ให้คลิกที่ INSTALL รอการติดตั้งโปรแกรมจนเสร็จ และจะมีหน้าต่างใหม่ขึ้นมาแจ้งว่า The drive is successfully Pre-installed in advance! ให้คลิก OK เป็นอันเสร็จสินการติดตั้ง และให้ปิดโปรแกรมติดตั้งได้เลย



ทำความรู้จักกับ Arduino

Arduino เป็น บริษัท ฮาร์ดแวร์และซอฟต์แวร์ โอเพ่นซอร์ส โครงการและชุมชนผู้ใช้ที่ออกแบบและผลิตไมโครคอนโทรลเลอร์บอร์ดเดี่ยวและชุดไมโครคอนโทรลเลอร์สำหรับสร้างอุปกรณ์ดิจิทัล ผลิตภัณฑ์ฮาร์ดแวร์ได้รับอนุญาตภายใต้ใบอนุญาต CC-BY-SA ในขณะที่ซอฟต์แวร์ได้รับอนุญาตภายใต้ GNU Lesser General Public License (LGPL) หรือ GNU General Public License (GPL), [1] อนุญาตให้ผลิตบอร์ด Arduino และการแจกจ่ายซอฟต์แวร์ โดยใครก็ได้ บอร์ด Arduino มีวางจำหน่ายทั่วไปจากเว็บไซต์ทางการหรือผ่านตัวแทนจำหน่ายที่ได้รับอนุญาต

การออกแบบบอร์ด Arduino ใช้ไมโครโปรเซสเซอร์และคอนโทรลเลอร์ที่หลากหลาย บอร์ดมี ชุด พิน อินพุต / เอาท์พุต (I / O) แบบดิจิตอลและอนาล็อกที่อาจเชื่อมต่อกับบอร์ดส่วนขยายต่างๆ หรือบอร์ดทดลอง และวงจรอื่น ๆ บอร์ดมีอินเทอร์เฟซการสื่อสารแบบอนุกรมซึ่งรวมถึง Universal Serial Bus (USB) ในบางรุ่นซึ่งใช้สำหรับโหลดโปรแกรมจากคอมพิวเตอร์ส่วนบุคคลด้วย ไมโครคอนโทรลเลอร์สามารถตั้งโปรแกรมได้โดยใช้ภาษาโปรแกรม C และ C ++ โดยใช้ API มาตรฐานซึ่งเรียกอีกอย่างว่า "ภาษา Arduino" นอกเหนือจากการใช้เป็นเครื่องมือคอมไพเลอร์แบบเดิมให้กับบอร์ดต่างแล้วโครงการ Arduino ยังมีสภาพแวดล้อมการพัฒนาแบบบูรณาการ (IDE) และเครื่องมือบรรทัดคำสั่ง (arduino-cli)

โครงการ Arduino เริ่มต้นในปี 2548 เพื่อเป็นเครื่องมือสำหรับนักเรียนที่ Interaction Design Institute Ivrea ใน Ivrea ประเทศอิตาลี [2] โดยมีเป้าหมายเพื่อจัดหาวิธีที่ง่ายและต้นทุนต่ำสำหรับมือใหม่และมืออาชีพในการสร้างอุปกรณ์ที่โต้ตอบกับสภาพแวดล้อมโดยใช้เซ็นเซอร์และ ตัวกระตุ้น ตัวอย่างทั่วไปของอุปกรณ์ดังกล่าวสำหรับมือสมัครเล่นระดับเริ่มต้น ได้แก่ หุ่นยนต์ เทอร์โมสตัทและเครื่องตรวจจับการเคลื่อนไหว

บทความนี้หาอ่านได้ตามลิงค์ https://en.wikipedia.org/wiki/Arduino

แนะนำการใช้งาน Arduino IDE

การเขียนโปรแกรม Arduino จะจัดรูปแบบโคลงสร้างที่สำคัญอยู่สองส่วนด้วยกันคือ ฟังก์ชั่น setup() และ ฟังก์ชั่น loop()

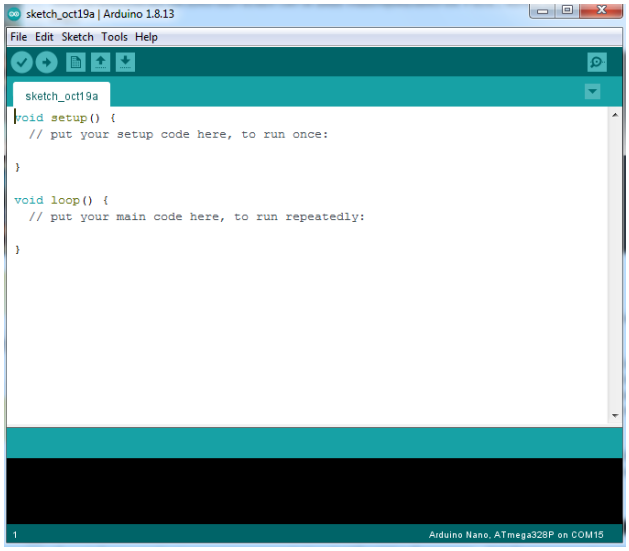
```
setup(){
        ชุดคำสั่ง
        }
```

ฟังก์ชั่น setup() ทำหน้าที่ในการกำหนดการทำงานของระบบ หรือ กำหนดคุณสมบัติการทำงานให้กับอุปกรณ์ต่างๆซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชั่นของ Setup() นี้ จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอนเริ่มต้นการทำงานของโปรแกรม โดย

คำสั่งที่นิยมบรรจุไว้ในฟังก์ชั่นส่วนนี้ ได้แก่ คำสั่งสำหรับกำหนดโหมดการทำงานของ Digital Pin หรือ คำสั่งสำหรับ กำหนดคุณสมบัติของพอร์ตสื่อสารอนุกรม เป็นต้น

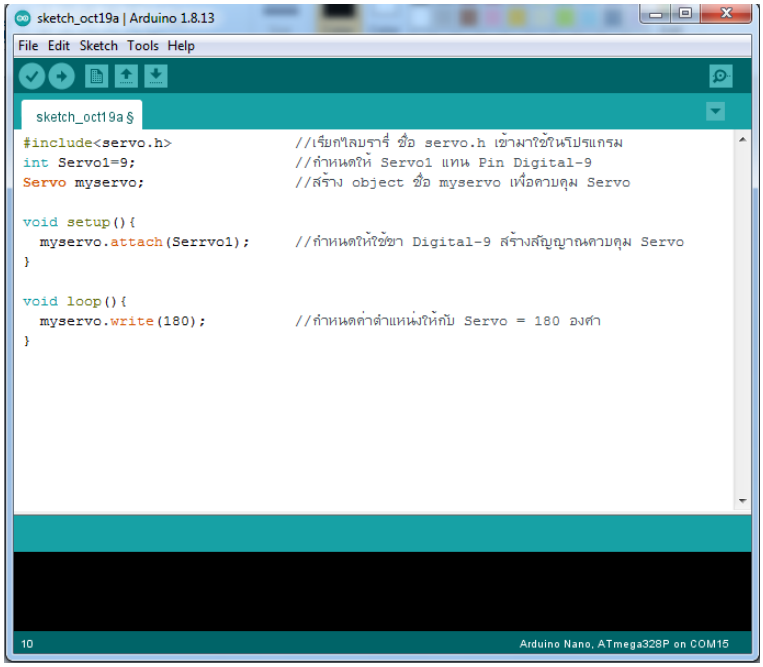
```
Loop(){
        ชุดโปรแกรม
        }
```

ฟังก์ชั่น loop() ใน Arduino ใช้ทำหน้าที่เป็นส่วนของโปรแกรมหลัก สำหรับใช้บรรจุคำสั่งควบคุมการทำงานต่างๆของโปรแกรม โดยคำสั่งที่บรรจุไว้ในฟังก์ชั่นนี้จะถูกเรียกขึ้นมาทำงานซ้ำๆกันตามลำดับและเงื่อนไขที่กำหนดไว้



รูปแสดงหน้าต่างของโปรแกรม

และยังมีอีกส่วนคือส่วนของ Header ซึ่งส่วนนี้จะมีหรือไม่มีก็ได้ ถ้ามีต้องกำหนดไว้ในส่วนเริ่มต้นของโปรแกรม ซึ่งส่วนของ Header ได้แก่ ส่วนที่เป็น Compiler Directive ต่างๆรวมไปถึงส่วนของการประกาศตัวแปร และค่าคงที่ต่างๆที่จะใช้ในโปรแกรม



รูปแสดงการเขียนโปรแกรม

จากรูปข้างบนจะเห็นว่ามีส่วนของโปรแกรมต่างมาให้จนครบเช่น

#include<servo.h> หมายถึงส่วนที่เป็น Header ซึ่งในส่วนนี้จะเป็นการดึงเอาค่าที่กำหนดเป็น ไลบรารี่ของ servo มาใช้งาน

int servo1 = 9; หมายถึง การกำหนดชื่อของตัวแปลในตำแหน่งที่ 9 หรือขา Digital D9 นั้นใช้ชื่อว่า servo1

Servo myservo; หมายถึงการสร้าง object ที่ชื่อ myservo เพื่อควบคุม servo

ในฟังก์ชั่น setup() กำหนดให้ myservo.attach(Servo); หมายถึง กำหนดให้ขาตำแหน่ง Digital D9 ทำหน้าที่ในการควบคุม servo

ในฟังก์ชั่น loop() กำหนดให้ myservo.write(180); หมายถึง สั่งให้ servo หมุนไปที่ 180 องศา และคำสั่งก็จะวนอยู่ใน loop() นี้ไม่มีวันสิ้นสุด เป็นการจบโปรแกรม

หน้าที่ของปุ่มต่างๆบน Arduino

 หมายถึงการ Verify หรือการ compiler ให้ภาษาที่เราเขียนเปลี่ยนไปเป็นภาษาเครื่องเพื่อใช้ในการ upload ลงบนบอร์ด Arduino

 หมายถึงการ Verify พร้อมกับ upload โปรแกรมลงบนบอร์ด Arduino

 หมายถึงการเปิดหน้าต่างใหม่ในการเขียนโปรแกรม

 หมายถึงการเปิดโปรแกรมที่เราเขียนเอาไว้หรือโปรแกรมตัวอย่างอื่นๆของ Arduino

 หมายถึงการบันทึกงานที่เราเขียน

เมื่อเข้าใจในการเขียนโปรแกรมแบบง่ายๆ เรามาทำความรู้จักกกับภาษา๙กันก่อน ส่วนใครที่อยากศึกษาลายละเอียดอื่นๆ สามารถหาอ่านได้จาก Google ซึ่งมีการจัดทำไว้มากมายหรือจะลองหาอ่านได้ที่ https://www.arduino.cc ได้โดยตรง

เบื่องต้นกับ ภาษา C

ความหมายของเครื่องหมายในภาษา C ตามมาตรฐาน สถาบันมาตรฐานแห่งชาติอเมริกัน American National Standard Institute (ANSI) หรือข้อกำหนดมาตรฐานของภาษาซี โดยเรียกกันว่า "ANSI-C" เพื่อใช้เป็นข้อบังคับและคงภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม

Slash star (/*) และสิ้นสุดที่ Star slash (*/) หมายถึงการใส่คำอธิบายไว้ภายใน /* และเมื่อจบคำอธิบายต้องปิดด้วย */ เสมอ และจะไม่มีผลต่อการคอมไพเลอร์ การใช้เครื่องหมายแบบนี้จะนิยมใช้กับการใส่คำอธิบายแบบยาวๆหลายบรรทัด

Double slash (//) หมายถึงการใส่คำอธิบายไว้ด้านหลังเครื่องหมาย // เหมาะสำหรับการอธิบายแบบสั้นๆ บรรทัดเดียว

Semicolon (;) หมายถึงเมื่อเราเขียนคำสั่งในหนึ่งชุดคำสั่งจะต้องต่อท้ายด้วย ;

บล็อก { ..... } คือสิ่งที่กำหนดขอบเขตและควบคุมการทำงานของโปรแกรม ซึ่งจะใช้เครื่องหมาย { และสิ้นสุดด้วย } ในภาษา C บล็อกนั้นมีหลายรูปแบบ เช่น บล็อกของฟังก์ชัน บล็อกของคำสั่งควบคุม หรือบล็อกย่อยในโปรแกรม และนอกจากนี้บล็อกยังสามารถซ้อนกันได้ นี่เป็นตัวอย่างของบล็อกในภาษา C

Whitespace คือตัวอักษรหรือเครื่องหมายที่ใช้แบ่งแยกคำสั่งและ Token ในโค้ดของโปรแกรม ในภาษา C นั้น Whitespace จะประกอบไปด้วย การเว้นวรรค Tab และการขึ้นบรรทัดใหม่ Whitespace ที่เรียงต่อกันเป็นจำนวนมากนั้นไม่มีผลต่อการทำงานของโปรแกรมและคอมไพเลอร์ แต่มันช่วยให้โปรแกรมเมอร์สามารถทำโค้ดให้เป็นระเบียบและสามารถอ่านเข้าใจได้ง่ายขึ้นโดยคนอื่นๆ แต่ Whitespace ยังคงต้องใช้ในบางที่ เช่น ระหว่างคำสั่งของภาษา C และชื่อของตัวแปร เป็นต้น

Keyword เป็นกลุ่มคำที่ถูกสงวนไว้โดยเราไม่สามารถใช้คำเหล่านี้ในการประกาศเป็นชื่อตัวแปร ฟังก์ชัน ซึ่งในโปรแกรมทุกภาษาต่างก็มี Keyword

นี่เป็น Keyword มาตรฐานในภาษา C

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Keyword 32 คำเหล่านี้มีหน้าที่การทำงานที่แน่นอนซึ่งขึ้นกับวัตถุประสงค์ของมัน int, short, float, double ใช้เพื่อประกาศตัวแปร ในขณะที่ if for while เป็นคำสั่งในการควบคุมการทำงานของโปรแกรม

ตัวแปรและประเภทข้อมูล

ตารางข้างล่างนี้แสดงประเภทข้อมูลทั้งหมดในภาษา C

Data type	Size bit	Value	**ความหมาย**
char	8	-128 to 127	เก็บค่าตัวอักษร
unsigned char	8	0 to 255	เก็บค่าตัวเลข
int	16	-32,768 to 32,767	เก็บค่าตัวเลข
unsigned int	16	0 to 65,535	เก็บค่าตัวเลข
long	32	-2,147,483,648 to 2,147,483,647	เก็บค่าตัวเลข
unsigned long	32	0 to 4,294,967,295	เก็บค่าตัวเลข
float	32	3.4x10E-38 to 3.4x10E+38	เก็บค่าตัวเลขที่เป็นทศนิยม
double	64	1.7x10E-308 to 1.7x10E+308	เก็บค่าตัวเลขที่เป็นทศนิยม

long double	80	3.4x10E-4932 to 3.4x10E+4932	เก็บค่าตัวเลขที่เป็นทศนิยม
boolean	1	0 to 1	เก็บค่าตัวเลข

ตัวแปรในภาษา C เป็นสถานที่สำหรับเก็บข้อมูลในหน่วยความจำ โดยมีชื่อของตัวแปรเพื่อใช้อ้างถึงข้อมูล (Identifier) ตัวแปรถูกใช้เพื่อเก็บค่า เราสามารถสร้างตัวแปรได้เป็นจำนวนมากโดยมีชื่อที่แตกต่างกัน รูปแบบของการประกาศตัวแปรในภาษา C คือ:

```
type identifier;
type identifier = value;
```

type เป็นประเภทของข้อมูลที่จะเก็บในตัวแปร identifier เป็นที่รู้จักกันในในชื่อของตัวแปร เราใช้ชื่อนี้เพื่ออ้างถึงค่าที่ตัวแปรนั้นเก็บอยู่ values เป็นทางเลือกที่คุณสามารถกำหนดค่าให้กับตัวแปรเมื่อมันถูกสร้าง หรือกำหนดในภายหลังได้ มาดูตัวอย่าง

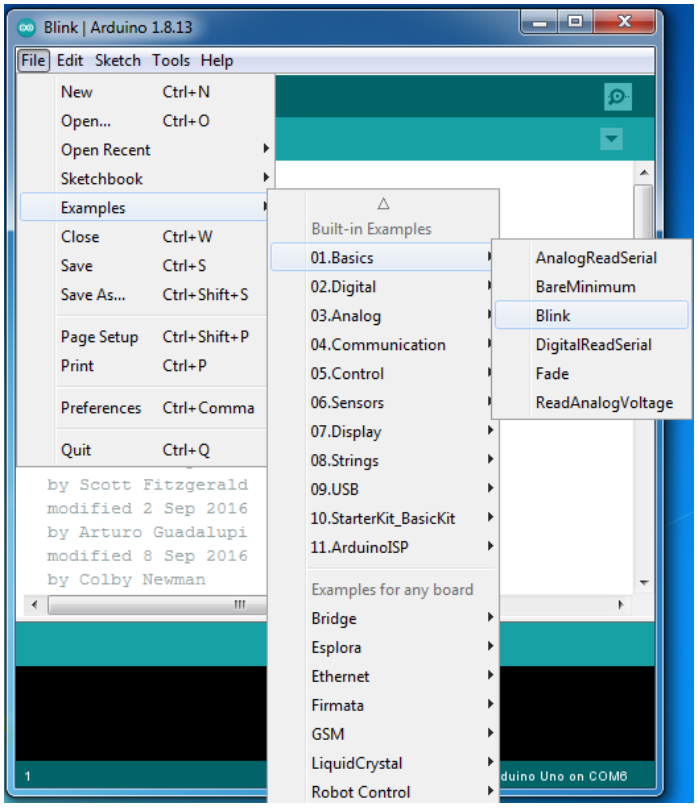
```
int a;
float b;
char c = 'A';
```

ในตัวอย่างเรามีตัวแปร 3 ตัว ตัวแรกประเภทของมันคือ int และมีชื่อว่า a มันใช้เพื่อเก็บค่าเลขจำนวนเต็ม (Integer) และเรายังไม่ได้กำหนดค่าใดๆ ให้มัน ดังนั้นค่าเริ่มต้นของตัวแปรเมื่อถูกสร้างขึ้นจะเป็น NULL ตัวแปรที่สองมีประเภทเป็น float ตัวแปรนี้จะถูกใช้เพื่อเก็บค่าของจำนวนจริง และตัวแปรที่สามมีประเภทเป็น char มันถูกใช้เพื่อเก็บสัญลักษณ์หนึ่งตัวใน ASCII code และเรากำหนด 'A' เป็นค่าเริ่มต้นให้กับมัน ในที่นี้ก็จะขอกล่าวเพียงย่อๆไว้เท่านี้ถ้าต้องการข้อมูลเพิ่มเติมสามารถหาอ่านได้ที่ Google พิมพ์หาคำว่า ภาษาซีเบื้องต้น ได้เลยครับ หรือที่ http://marcuscode.com/lang/c , https://sites.google.com/a/phukhieo.ac.th/dd/history_c , http://marcuscode.com/lang/c/operators

หลังจากได้ทำความเข้าใจภาษาซีและหน้าตาของมันพอสังเขปแล้วต่อมาเรามารองทดสอบบอร์ดและทดลองเขียนโปรแกรมกัน

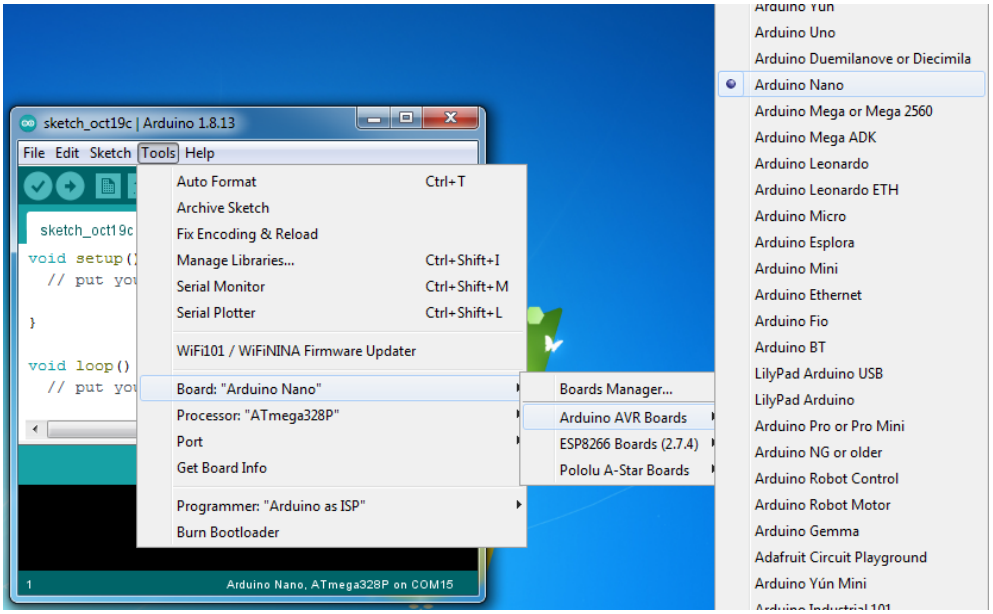
ทดลองเขียนโปรแกรมกัน และต่อหุ่นยนต์เข้ากับ Arduino IDE

หลังจากที่ลงโปรแกรม และติดตั้ง ไดรเวอร์แล้ว ให้เราเลียบบอร์ด AVR NANOBOT เข้ากับคอมพิวเตอร์ แล้วเปิดโปรแกรม Arduino IDE ขึ้นมา จากนั้นให้เปิดโปรแกรมตัวอย่างที่ชื่อ blink ที่หน้าต่าง Arduino IDE ไปที่เมนู File เลือก Examples>Basics>Blink



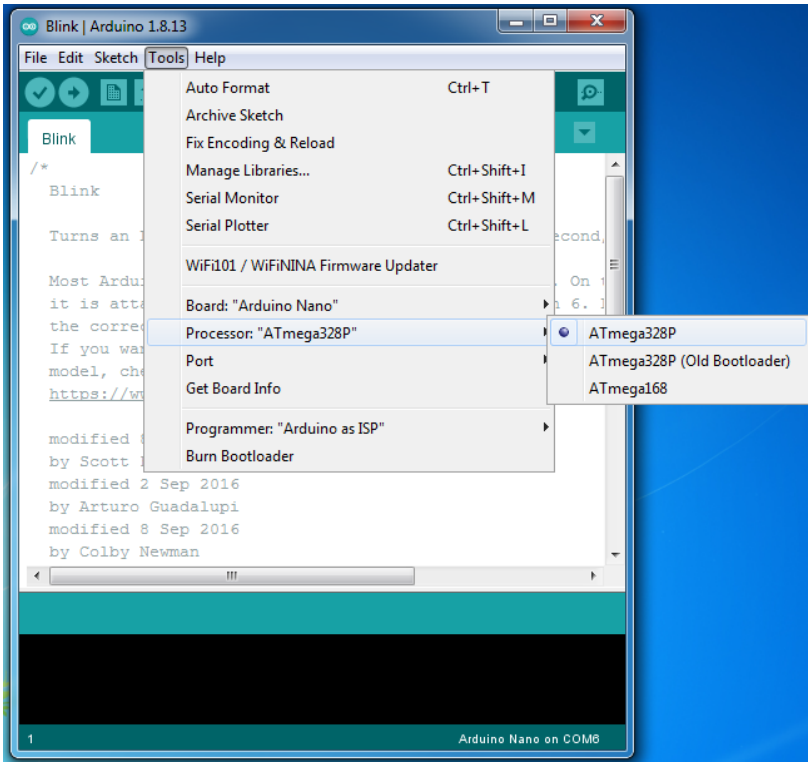
รูปแสดงการเปิดไฟล์ตัวอย่างที่ชื่อ Blink

เมื่อได้ไฟล์ตัวอย่างแล้วต่อไปเราจะเลือกบอร์ดที่จะใช้งาน ในที่นี้เราจะใช้บอร์ดที่ชื่อ Arduino Nano ให้ไปที่เมนู Tools เลือก Board>Arduino AVR Board>Arduino Nano ตามภาพ ยังไม่เสร็จครับ เราจะต้องเลือก Processor ด้วยว่าเป็นเบอร์อะไร และอยู่ Com Port ไหน

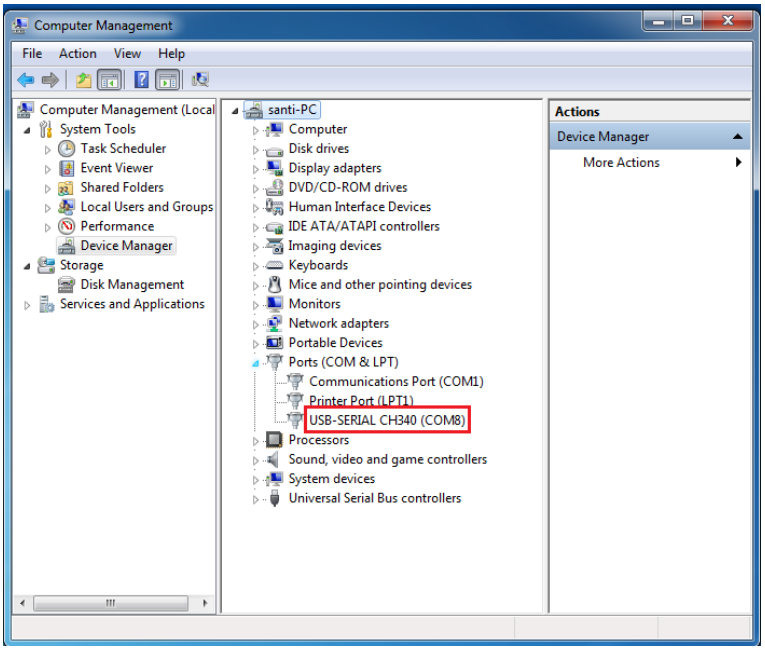


รูปแสดงการเลือกบอร์ดที่จะนำมาเขียนโปรแกรม

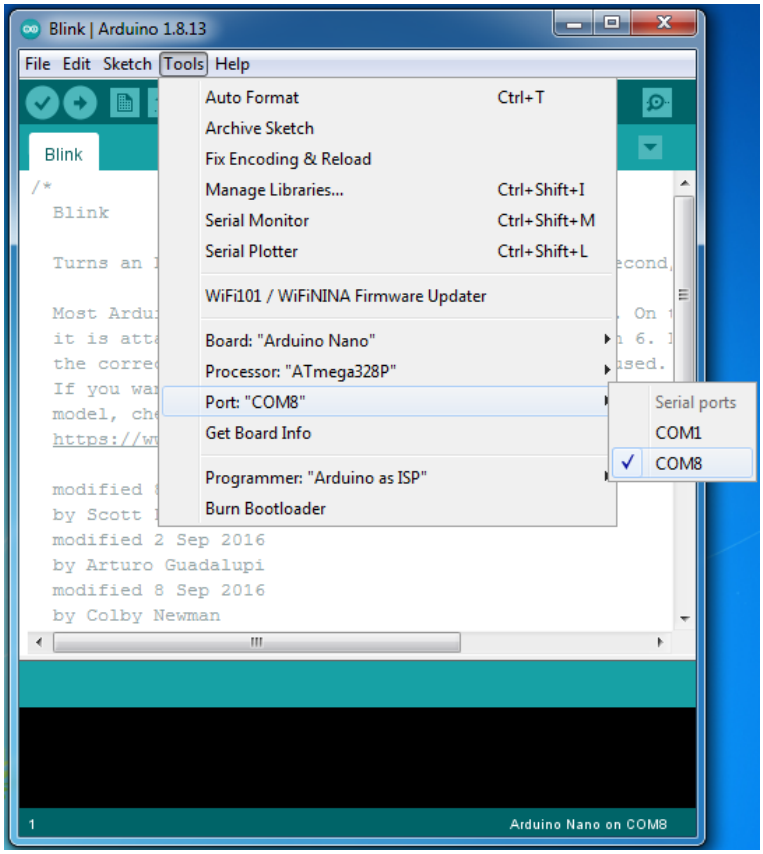
ให้ไปที่เมนู Tool อีกครั้ง เลือกไปที่ Processor>ATmega328p ตามรูปด้านล่าง



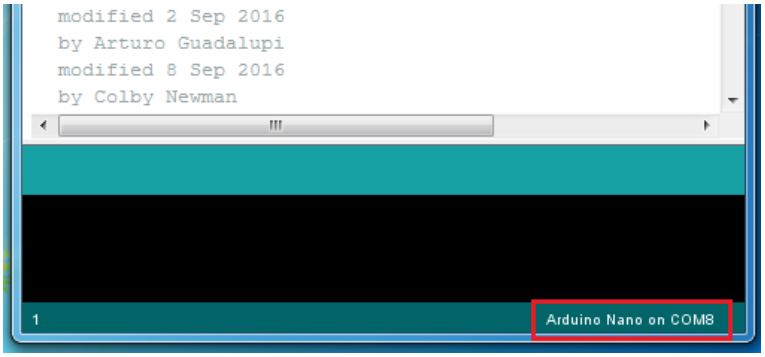
และไปที่เมนู Tools อีกครั้งเพื่อที่จะเลือกว่าอยู่ Com Port ไหน วิธีดูว่าบอร์ด AVR NANOBOT อยู่ COM Port ที่เท่าไร ให้คลิกที่ Start เมนูของวินโดว์แล้วคลิก ขวา ที่ computer เลือก manage จากนั้นจะขึ้นหน้าต่าง Computer Management ให้เลือกที่ Device Manager จะมีลายการแสดงทางด้านขวามือดูที่ Port จะเห็นไดร์เวอร์ USB ที่ถูกติดตั้งไว้ และบอกตำแหน่งของ COM ที่เท่าไร ดังรูปด้านล่าง



ตามรูปจะแสดงตำแหนง COM8 ดังนั้นให้เราเลือกที่ Arduino ให้เป็น COM8 เช่นกัน โดยไปที่ Tools เลือก Port>COM8 ตามรูปด้านล่าง



เมื่อมาถึงขั้นตอนนี้แล้วให้เราจะสังเกตุที่หน้าต่าง Aruino มุมล่างขวา จะแสดง ชื่อบอร์ดและตำแหน่งของ Port USB ที่ใช้งานดังรูปด้านล่าง



ต่อไปเรามาลอง Burn Program ตัวอย่างลงไปบนบอร์ดกัน โดยคลิกที่  และสังเกตที่ไฟ LED RX และ TX จะติดกระพริบ นั้นแสดงว่า บอร์ดกับโปรแกรมสามารถติดต่อกันได้และกำลัง Downs load ไฟล์ที่ชื่อ Blink ลงไนชิพไอซีแล้ว เมื่อโปรแกรมเสร็จ จะเห็น LED D13 ติดกระพริบเป็นจังหวะห่างกัน 1 วินาที ให้เราลองแก้ไขโค้ด ให้การติดดับเร็วขึ้น ตามตัวอย่างด้านล่าง

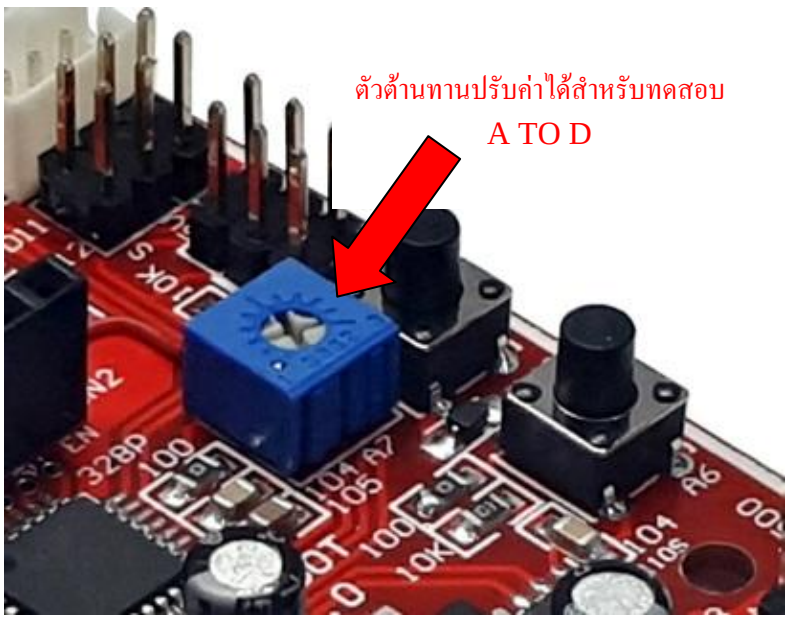
```
void loop() {
                digitalWrite(LED_BUILTIN, HIGH);
                delay(200);
                digitalWrite(LED_BUILTIN, LOW);
                delay(200);
}
```

ให้แก้ที่ ฟังชั่น loop() ในบรรทัด delay(1000); เปลี่ยนเป็น delay(200); ทั้งสองบรรทัด จากนั้นก็คลิกที่  สั้งเกตุ LED D13 จะกระพริบติดดับเร็วขึ้น หรือจะลองแก้เป็นค่าอื่นก็ได้แล้วสังเกตการทำงานดู

มาถึงขั้นตอนนี้เราก็สามารถใช้บอร์ดทำงานต่างๆตามที่เราเขียนได้แล้วครับ ต่อไปจะมีตัวอย่างการใช้งานในแบบต่างๆให้ทดลองกัน ไปติดตามกันต่อ

ใน Arduino มีโค้ดตัวอย่างให้ทดลองมากมายแต่บางโค้ดเราต้องแก้ตำแหน่งของขาให้ตรงกับบอร์ด AVR NANOBOT เราไปดูกันเลยดีกว่าจากตัวอย่างที่จะให้ทดลองดังต่อไปนี้

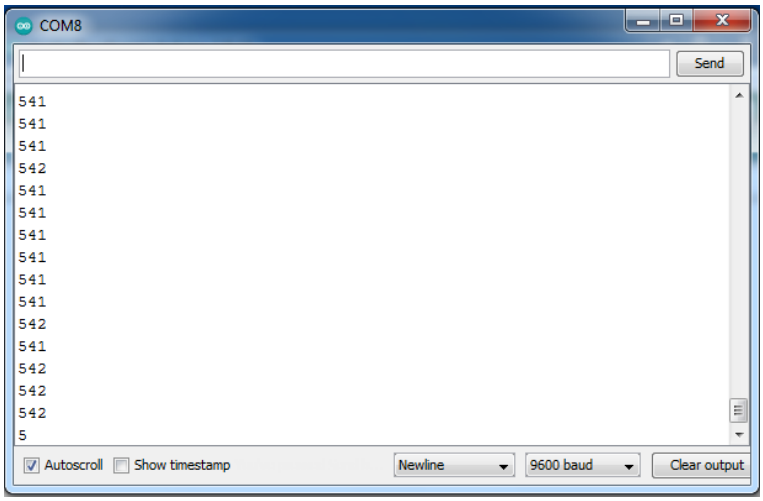
ตัวอย่าง A TO D



EX1_AtoD การทดลองอ่านค่า A to D

```
int adc_A7;                          // ประกาศตัวแปรชื่อ adc A7
void setup()
{
  Serial.begin(9600);                //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
}

void loop()
{
  adc_A7 = analogRead(A7);     //อ่านค่าอนาล็อกตำแหน่งที่ A7 มาเก็บไว้ที่ adc_A7
  Serial.println(adc_A7);      //แสดงค่า adcValue ออกทางหน้าจอ
  delay(500);                        //หน่วงเวลา 500ms
        }
```

ให้พิมพ์โค้ดตามตัวอย่างที่ EX1 แล้ว คลิก Upload เมื่อเสร็จให้เปิดหน้าต่าง Serial Monitor โดยไปที่ มุมขวาบนของหน้าต่าง Arduino คลิกที่  จะปรากฏหน้าต่างดังรูปด้านล่าง



ให้เราปรับตัวต้านทานแบบปรับค่าได้ ต่ำสุด และ จนสูงสุด จะได้ค่าอยู่ที่ 0 –1023 เมื่อได้ตามนี้ต่อไปเราจะลองเพิ่มโค้ด โดยเมื่อเราปรับค่า มากกว่า 550 ให้ LED D13 ติด และเมื่อเราปรับให้น้อยกว่า 500 ให้ LED D13 ดับ มาดูโค้ดกัน ตามตัวอย่างที่ EX2.AtoD

EX2_AtoD ทดสอบการติดดับตามการปรับค่าความต้านทาน

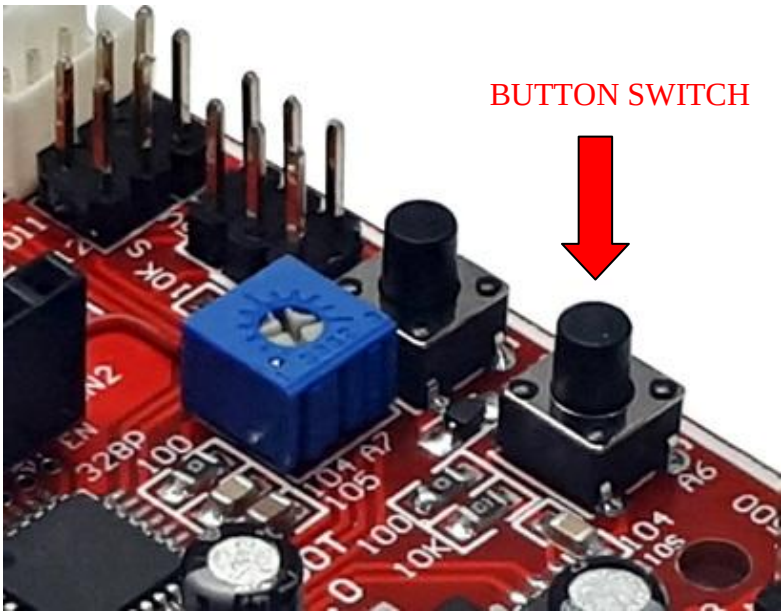
```
int adc_A7;          // ประกาศตัวแปรชื่อ adc A7
int ledPin = 13;     // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
void setup()
{
  Serial.begin(9600);  //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
}

void loop()
{
  adc_A7 = analogRead(A7);     //อ่านค่าอนาล็อกตำแหน่งที่ A7 มาเก็บไว้ที่ adc_A7
  Serial.println(adc_A7);      //แสดงค่า adcValue ออกทางหน้าจอ
   if(adc_A7 > 550)                //เปลียบเทียบค่า adc_A7 มากกว่า 550 หรือไม่
       digitalWrite(ledPin, HIGH);        //มากกว่าให้ ledPin ติด
   else if(adc_A7 < 550)                   //เปลียบเทียบค่า adc_A7 น้อยกว่า 500 หรือไม่
       digitalWrite(ledPin, LOW);         //มากกว่าให้ ledPin ดับ

  delay(500);                        //หน่วงเวลา 500ms
}
```

ให้เพิ่มโค้ดในตำแหน่งก่อนฟังชั่น setup() เพื่อประกาศตัวแปลชื่อ ledPin ส่วนในฟังชั่น setup() ให้กำหนดตัวแปล ledPin เป็น OUTPUT และในส่วน loop() ที่ใต้บรรทัด Serial.println(adc_A7); ให้เพิ่มคำสั่งอีกสองบรรทัดและ upload โค้ด จากนั้นให้เราลองปรับ ตัวต้านทาน ให้มากกว่า 550 จะเห็น LED ติด และเมื่อปรับให้น้อยกว่า 500 LED จะดับ มาถึงตอนนี้เราก็สามารถควบคุม LED ให้ติด ดับ ตามการปรับที่ ตัวต้านทาน ได้แล้ว โค้ดต่อไปจะเป็นอะไรไปดูกัน

Ex3.AtoD ให้ LED ติดดับ เร็วขึ้นหรือช้าลง ด้วยการปรับ ค่าตัวต้านทาน

```
int adc_A7;           // ประกาศตัวแปรชื่อ adc A7
int ledPin = 13;      // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
void setup()
{
  Serial.begin(9600);                 //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
  pinMode(ledPin,HIGH);          //กำหนดให้ ledPin เป็น output
}
void loop()
{
  adc_A7 = analogRead(A7);              //อ่านค่าอนาล็อกตำแหน่งที่ A7 มาเก็บไว้ที่ adc_A7
  Serial.println(adc_A7);               //แสดงค่า adcValue ออกทางหน้าจอ

  digitalWrite(ledPin, HIGH);          //มากกว่าให้ ledPin ติด
  delay(adc_A7);                        //นำค่า adc_A7 มาหน่วงเวลา

  digitalWrite(ledPin, LOW);           //มากกว่าให้ ledPin ติด
  delay(adc_A7);                        //นำค่า adc_A7 มาหน่วงเวลา
}
```

ตัวอย่าง BUTTON SWITCH



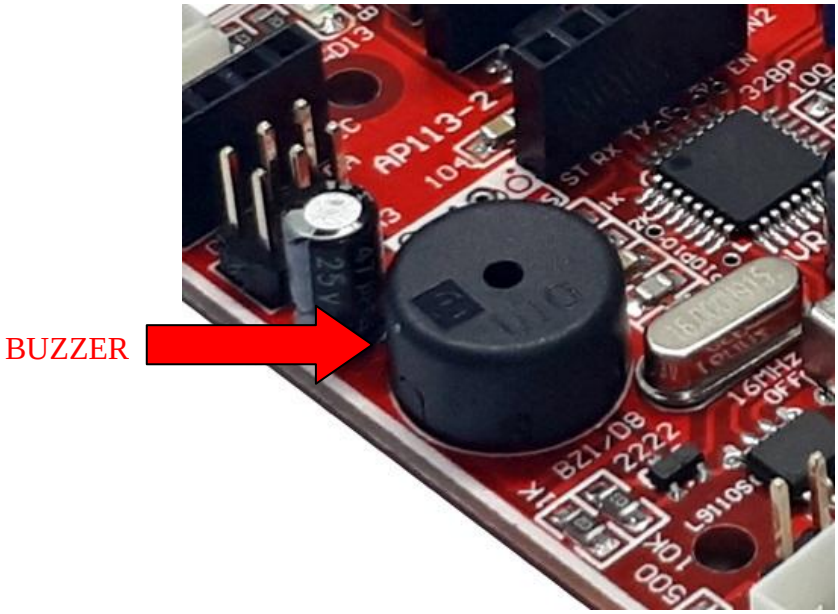
EX4.buttonSwitch ทดสอบการทำงานของ switch ที่ตำแหน่ง A6 แบบ กดติด ปล่อยดับ

```
int switchValue;      // ประกาศตัวแปรชื่อ switchValue
int ledPin = 13;      // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
void setup()
{
  Serial.begin(9600);  //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
}

void loop()
 {
  switchValue = analogRead(A6); //อ่านค่าอนาล็อกตำแหน่งที่ A7 มาเก็บไว้ที่ switchValue
  Serial.println(switchValue);  //แสดงค่า swithValue ออกทางหน้าจอ

  if(switchValue > 500)         //เปลียบเทียบค่าของ switchValue มากกว่า 500 หรือไม่
        digitalWrite(ledPin, HIGH);   //น้อยกว่าให้ ledPin ติด
  else if(switchValue < 500)          //เปลียบเทียบค่าของ switchValue น้อยกว่า 500 หรือไม่
        digitalWrite(ledPin, LOW);    //มากกว่าให้ ledPin ดับ
  delay(100);                         //หน่วงเวลา 100ms
}
```

EX5.buttonSwitch เมื่อกดสวิตช์ A6 แล้วให้ LED D13 ติด และเมื่อกดอีกครั้งให้ LED D13 ดับ

```
int switchValue;      // ประกาศตัวแปรชื่อ switchValue
int ledPin = 13;      // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
boolean toggle;       // ประกาศตัวแปรชื่อ toogle
void setup()
{
  Serial.begin(9600);  //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
}

void loop()
{
  switchValue = analogRead(A6); //อ่านค่าอนาล็อกตำแหน่งที่ A6 มาเก็บไว้ที่ switchValue
  Serial.println(switchValue);  //แสดงค่า switchValue ออกทางหน้าจอ

  if(switchValue > 500)    //เปลียบเทียบค่าของ switchValue มากกว่า 500 หรือไม่
        toggle = !toggle;  //น้อยกว่า เปลี่ยนค่า toggle จาก 0 ให้เป็น 1 หรือ จาก 1 เป็น 0

  if(toggle == 1 )        //ค่าของ toogle = 1 หรือไม่
  {
    digitalWrite(ledPin, HIGH);   //ใช่ให้ ledPin ติด
  }

  else
  {
    digitalWrite(ledPin, LOW);              //ไม่ใช่ ให้ LED ดับ
  }

  while(switchValue > 500) //ต่าที่อ่านได้มากว่า 500 ให้วนในลูบ while
  {
      switchValue = analogRead(A6);//อ่านค่าอนาล็อกตำแหน่งที่ A6 มาเก็บไว้ที่ switchValue
  }//ถ้าค่าที่อ่านได้มากกว่า 500 ให้ออกจาก ลูบ while

  delay(20);              //หน่วงเวลาเพื่อป้องกันการเกิดสัญญาณกวนของการกดสวิตช์
}
```

ตัวอย่างต่อไปจะว่าด้วยเรื่องของ BUZZER



EX6. BuzSwitch ตัวอย่างนี้เมื่อกดสวิตช์ A6 จะทำให้มีเสียง

```
int switchValue;      // ประกาศตัวแปรชื่อ switchValue
int ledPin = 13;      // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
int BuzPin = 8;       // ประกาศตัวแปร BuzPin ใช้ชื่อแทน Digital Pin 8

int noteDuration;     // ประกาศตัวแปรชื่อ noteDuration
int pauseBetweenNotes;// ประกาศตัวแปรชื่อ pauseBetweenNotes

// ฟังชั่น sound พร้อมตัวแปรกำหนดความยาวเสียง และ ความถี่เสียง
void sound(int noteDT , int melody)
{
  noteDuration = 1000 / noteDT; // นำค่าความยาวเสียงแบบ 4 หรือ 8 ตัวโนด หารในช่วงเวลา 1 วินาที และเก็บไว้ที่ตัวแปร
noteDuration
  tone(BuzPin, melody, noteDuration);// นำค่าที่ได้ใส่ในตัวแปรตามลำดับ และส่งไปที่ฟังก์ชั่น tone
}

void setup() {
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
  pinMode(BuzPin,HIGH);//กำหนดให้ BuzPin เป็น output
}

void loop() {
  switchValue = analogRead(A6); //อ่านค่าอนาล็อกตำแหน่งที่ A6 มาเก็บไว้ที่ switchValue
  if(switchValue > 500)    //เปลียบเทียบค่าของ switchValue มากว่า 500 หรือไม่
  {
    digitalWrite(ledPin, HIGH);   //ให้ ledPin ติด
    sound(4, 1000);              //กำหนดค่าความยาวเสียง 4 ความถี่เสียง 1000 เฮิรตซ์
    digitalWrite(ledPin, LOW);   //ให้ ledPin ดับ
  }
}
```

เมื่ออัพโหลดโค้ดแล้ว ให้กดสวิตช์ A6 จะได้ยินเสียงตามการกดสวิตช์

EX7.melody

```
#include "pitches.h" //เรียกใช้ไลบรารี่ pitches.h มาใช้งาน ในไลบารี่จะกำหนดความถี่ของตัวโนตไว้ให้แล้ว
int switchValue;     // ประกาศตัวแปรชื่อ switchValue
int ledPin = 13;     // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
int BuzPin = 8;      // ประกาศตัวแปร BuzPin ใช้ชื่อแทน Digital Pin 8

int noteDuration;    // ประกาศตัวแปรชื่อ noteDuration
int pauseBetweenNotes;// ประกาศตัวแปรชื่อ pauseBetweenNotes

// กำหนดการเรียกใช้ ค่าความถี่โนตในไลบรารี่ pitches.h
int melody[] = {
  NOTE_C5, NOTE_G4, NOTE_G4, NOTE_A4,
  NOTE_G4,       0, NOTE_B4, NOTE_C5
};

// กำหนดค่าความยาวเสียง 4 = quarter note, 8 = eighth note
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void sound(int noteDT) // ฟังชั่น sound พร้อมตัวแปรกำหนดความยาวเสียง และ ความถี่เสียง
{
  // กำหนดความยาวเสียงแบบ 4 หรือ 8 ตัวโนต ในช่วงเวลา 1 วินาที
  noteDuration = 1000 / noteDurations[noteDT]; //นำ 1000 หาร ด้วยค่าความยาวเสียง และนำไปเก็บไว้ที่ตัวแปล
noteDuration
  tone(BuzPin, melody[noteDT], noteDuration); // นำค่าที่ได้ใส่ในตัวแปรตามลำดับ และส่งไปที่ฟังก์ชั่น tone
  pauseBetweenNotes = noteDuration * 1.30; //นำค่าในตัวแปล noteDuration คูณ กับ 1.30 แล้วเก็บที่
pauseBetweenNotes
  delay(pauseBetweenNotes); //นำต่า pauseBetweenNotes ส่งไปที่ฟังก์ชั้น delay เพื่อกำหนดควมห่างของการอยุด
ของเสียง
}
void setup() {
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
}
void loop() {
  switchValue = analogRead(A6); //อ่านค่าอนาล็อกตำแหน่งที่ A6 มาเก็บไว้ที่ switchValue
if(switchValue > 500)    //เปรียบเทียบค่าของ switchValue มากว่า 500 หรือไม่
  {
    for (int thisNote = 0; thisNote < 8; thisNote++)
    {
      digitalWrite(ledPin, HIGH); //ใช่ให้ ledPin ติด
      sound(thisNote);
      digitalWrite(ledPin, LOW);  //ใช่ให้ ledPin ดับ
      noTone(BuzPin);             //หยุดการเล่นเสียง
    }
  }
}
```

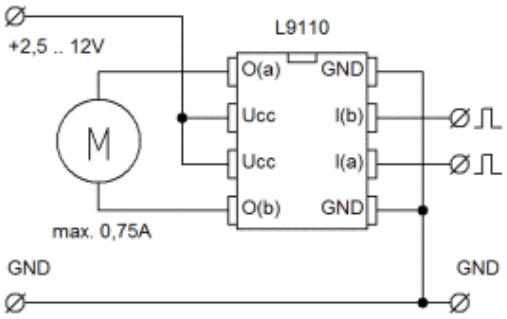
เมื่ออัพโหลดโค้ดแล้ว ให้กดสวิตช์ A6 จะได้ยินเสียงเพลงตามโค้ดที่เราเขียน

EX8.melodyJingleBells ตัวอย่างเพลง Jingle bells

```
#include "pitches.h" //เรียกใช้ไลบรารี่ pitches.h มาใช้งาน ในไลบารี่จะกำหนดความถี่ของตัวโนตไว้ให้แล้ว
int switchValue;      // ประกาศตัวแปรชื่อ switchValue
int ledPin = 13;      // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
int BuzPin = 8;       // ประกาศตัวแปร BuzPin ใช้ชื่อแทน Digital Pin 8

int noteDuration;    // ประกาศตัวแปรชื่อ noteDuration
int pauseBetweenNotes;// ประกาศตัวแปรชื่อ pauseBetweenNotes

// เพลง Jingle Bells
int jingle_melody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5,
  NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5
};
// จังหวะเพลง Jingle bells
int jingle_tempo[] = {
  6, 6, 3,
  6, 6, 3,
6, 6, 6, 6,
  2,
  6, 6, 6, 6,
  6, 6, 6, 10, 10,
  6, 6, 6, 6,
  2, 2,
};

void sound(int noteDT) // ฟังชั่น sound พร้อมตัวแปรกำหนดความยาวเสียง และ ความถี่เสียง
{
  // กำหนดความยาวเสียงแบบ 4 หรือ 8 ตัวโนต ในช่วงเวลา 1 วินาที
  noteDuration = 1000 / jingle_tempo[noteDT]; //นำ 1000 หาร ด้วยค่าความยาวเสียง และนำไปเก็บไว้ที่ตัวแปล noteDuration
  tone(BuzPin, jingle_melody[noteDT], noteDuration); // นำค่าที่ได้ใส่ในตัวแปรตามลำดับ และส่งไปที่ฟังก์ชั่น tone
  pauseBetweenNotes = noteDuration * 1.30; //นำค่าในตัวแปล noteDuration คูณ กับ 1.30 แล้วเก็บที่ pauseBetweenNotes
  delay(pauseBetweenNotes); //นำค่า pauseBetweenNotes ส่งไปที่ฟังก์ชั่น delay เพื่อกำหนดคาวมห่างของการอยุดของเสียง
}
void setup() {
  pinMode(ledPin,HIGH);//กำหนดให้ ledPin เป็น output
}

void loop() {
  switchValue = analogRead(A6); //อ่านค่าอนาล็อกตำแหน่งที่ A6 มาเก็บไว้ที่ switchValue
  if(switchValue < 500)    //เปรียบเทียบค่าของ switchValue น้อยกว่า 500 หรือไม่
  {
for (int thisNote = 0; thisNote < 25; thisNote++) //กำหนดให้อ่านค่าตัวโน้ดไม่เกิน 8 ตัว
    {
      digitalWrite(ledPin, HIGH); //ใช่ให้ ledPin ติด
      sound(thisNote);
      digitalWrite(ledPin, LOW);  //ใช่ให้ ledPin ดับ
      noTone(BuzPin);             //หยุดการเล่นเสียง
    }}}
```

มอเตอร์ DC

IC L9110



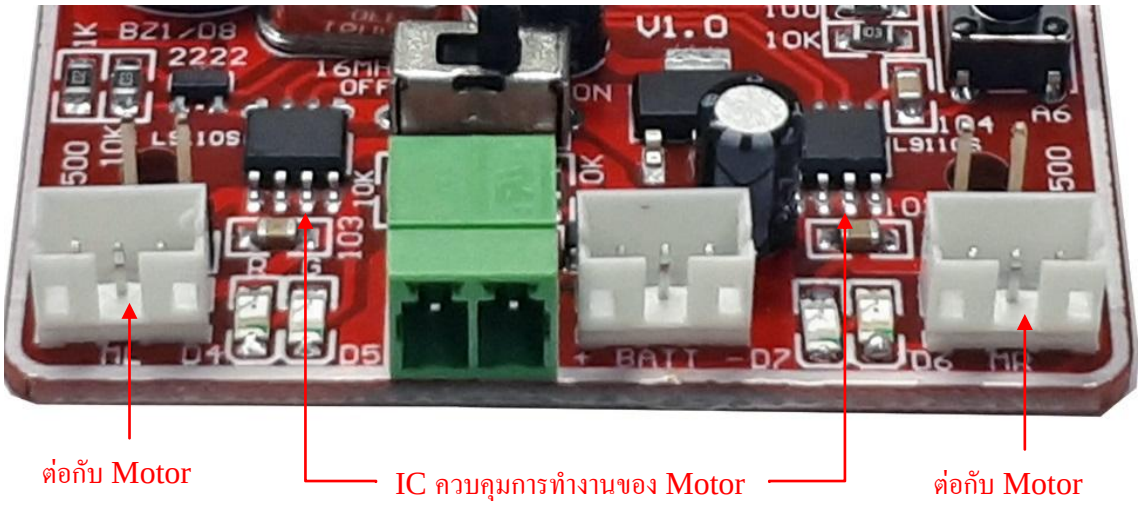
MOTOR DC

ตารางแสดงการทำงานของ IC L9110S ควบคุมมอเตอร์ DC

IA	IB	OA	OB
D5,D7	D4,D6		
L	L	L	L
H	L	H	L
L	H	L	H
H	H	H	H

ตารางแสดงทิศทางการทำงานของมอเตอร์

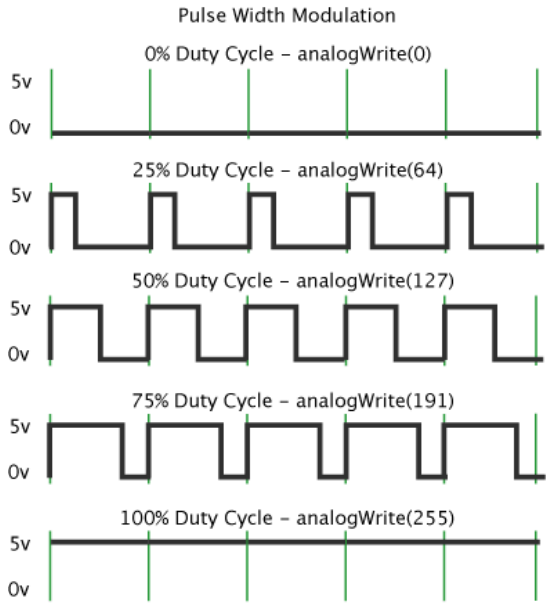
MOTOR L		MOTOR R		การทำงานของมอเตอร์
D4/LIB	D5/LIA/PWM	D6/RIB/PWM	D7/RIA	
0	0	0	0	STOP OUTPUT LOW
1	1	1	1	STOP OUTPUT HIGH
0	1	0*	1	เดินหน้า
1	0*	1	0	ถอยหลัง
0	1	1	0	เลี้ยวขวา
1	0	0	1	เลี้ยวซ้าย
หมายเหตุ (*) คำสั่ง PWM ต้องกลับค่าการทำงาน				



EX9.motor ตัวอย่างการทดสอบการทำงานของมอเตอร์

```
int LIA = 5;  //ประกาศตัวแปรชื่อ LIA ใช้ชื่อแทน Digital Pin 5
int LIB = 4;  //ประกาศตัวแปรชื่อ LIB ใช้ชื่อแทน Digital Pin 4
int RIA = 7;  //ประกาศตัวแปรชื่อ RIA ใช้ชื่อแทน Digital Pin 7
int RIB = 6;  //ประกาศตัวแปรชื่อ RIB ใช้ชื่อแทน Digital Pin 6
void setup()
{
  pinMode(LIA,OUTPUT);  //กำหนดให้ LIA เป็น output
pinMode(LIB,OUTPUT);  //กำหนดให้ LIB เป็น output
  pinMode(RIA,OUTPUT);  //กำหนดให้ RIA เป็น output
  pinMode(RIB,OUTPUT);  //กำหนดให้ RIB เป็น output
}
void loop()
{
  //เดินหน้า
  digitalWrite(LIA,HIGH);  //มอเตอร์ซ้ายเดินหน้า
  digitalWrite(LIB,LOW);
  digitalWrite(RIA,HIGH);  //มอเตอร์ขวาเดินหน้า
  digitalWrite(RIB,LOW);
  delay(2000);
    //ถอยหลัง
  digitalWrite(LIA,LOW);  //มอเตอร์ซ้ายถอยหลัง
  digitalWrite(LIB,HIGH);
  digitalWrite(RIA,LOW);  //มอเตอร์ขวาถอยหลัง
  digitalWrite(RIB,HIGH);
  delay(2000);

  //เลี้ยวซ้าย
  digitalWrite(RIA,HIGH);  //มอเตอร์ขวาเดินหน้า
  digitalWrite(RIB,LOW);
  digitalWrite(LIA,LOW);  //มอเตอร์ซ้ายถอยหลัง
  digitalWrite(LIB,HIGH);
  delay(500);
  //เลี้ยวขวา
  digitalWrite(LIA,HIGH);  //มอเตอร์ซ้ายเดินหน้า
  digitalWrite(LIB,LOW);
  digitalWrite(RIA,LOW);  //มอเตอร์ขวาถอยหลัง
  digitalWrite(RIB,HIGH);
    delay(500);
 }
```

การทดสอบควบคุมความเร็วของมอเตอร์ด้วย PWM



รูปแสดงสัญญาณควบคุมมอเตอร์แบบ PWM (Pulse Width Modulation)

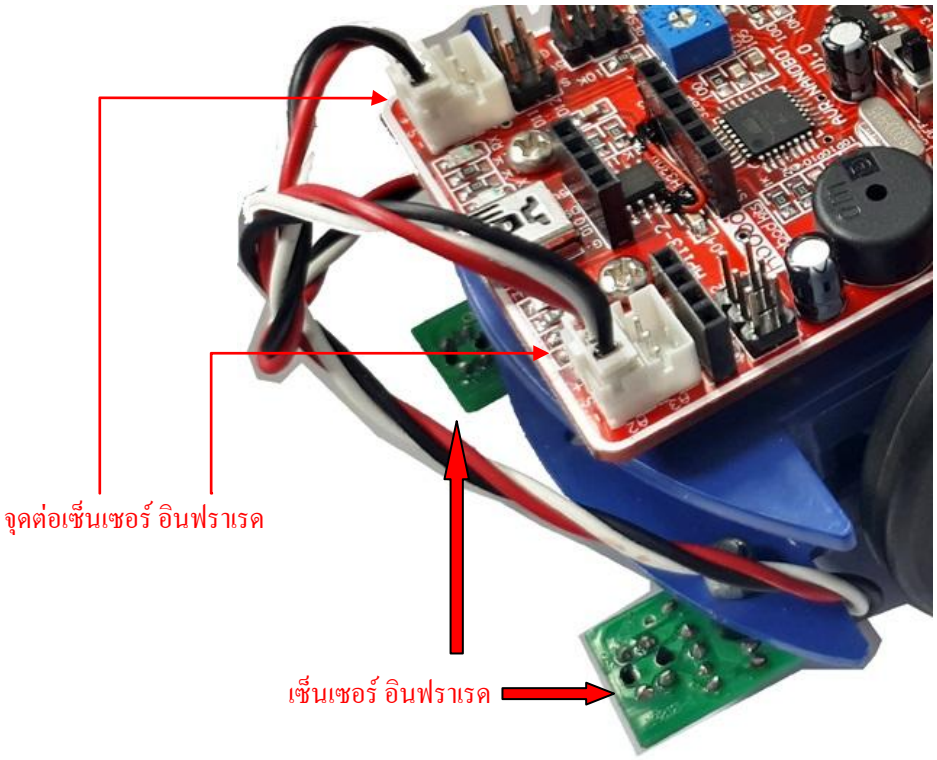
คือมอเตอร์จะทำงานตามความกว้างของพัลส์ หรือ Duty Cycle ถ้ามีค่าเปอร์เซ็นน้อยก็จะหมุนช้า และ ถ้ามีค่าเปอร์เซ็นมากก็จะหมุนเร็ว เราลองมาดูโค้ดโปรแกรมกัน

EX10.motorPWM

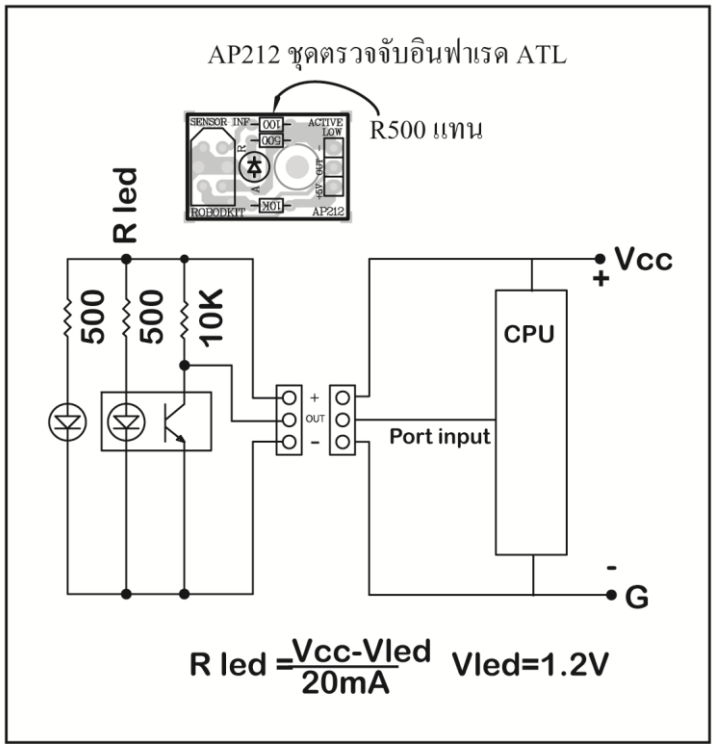
```
int LIA = 5;  //ประกาศตัวแปรชื่อ LIA ใช้ชื่อแทน Digital Pin 5 PWM
int LIB = 4;  //ประกาศตัวแปรชื่อ LIB ใช้ชื่อแทน Digital Pin 4
int RIA = 7;  //ประกาศตัวแปรชื่อ RIA ใช้ชื่อแทน Digital Pin 7
int RIB = 6;  //ประกาศตัวแปรชื่อ RIB ใช้ชื่อแทน Digital Pin 6 PWM

int XIA;      //ประกาศตัวแปรชื่อ XIA
int YIB;      //ประกาศตัวแปรชื่อ YIB

void motor(char Mot_N, char _move, int adj_speed) //ฟังก์ชั่นการควบคุมการทำงานของ motor
{
 switch(Mot_N) //ตรวจสอบตัวแปร Mot_N เป็นการเลือกการทำงานของมอเตอร์ ซ้าย หรือ ขวา
 {
   case 'L':     //รับค่า L คือเลือกมอเตอร์ ซ้าย
     XIA = LIA;  //ให้ตัวแปรชื่อ XIA แทนขา Pin 5 PWM
     YIB = LIB;  //ให้ตัวแปรชื่อ YIB แทนขา Pin 4
    if(_move == 'B') adj_speed = 255-((255*adj_speed)/100);  //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255 และ กลับค่า
     else    adj_speed = (255*adj_speed)/100;                //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255
     break;      //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
   case 'R':     //รับค่า R คือเลือกมอเตอร์ ขวา
     XIA = RIA;  //ให้ตัวแปรชื่อ XIA แทนขา Pin 7
     YIB = RIB;  //ให้ตัวแปรชื่อ YIB แทนขา Pin 6 PWM

     if(_move == 'F') adj_speed = 255-((255*adj_speed)/100);  //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255 และ กลับค่า
     else    adj_speed = ((255*adj_speed)/100);              //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255
   break;       //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
 }

 switch(_move)   //ตรวจสอบตัวแปร _move เป็นการเคลื่อนที่ของมอเตอร์แบบ เดินหน้า ถอยหลัง หรือ หยุด
 {
   case 'F':                    //ตัวแปร _move คือ F สั่งให้มอเตอร์ เดินหน้า
     if(Mot_N == 'L')              //ตรวจสอบเป็นมอเตอร์ ซ้าย ใช่ ให้ทำงานใน {....} ไม่ใช้ไปที่ else
     {
       analogWrite(XIA, adj_speed);  //เขียนคำสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
       digitalWrite(YIB, LOW);       //เขียนคำสั่ง LOW ออกไปที่ Pin ที่เลือก
     }
     else                         //การทำงานของมอเตอร์ ขวา
     {
       analogWrite(YIB, adj_speed);  //เขียนคำสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
       digitalWrite(XIA, HIGH);      //เขียนคำสั่ง LOW ออกไปที่ Pin ที่เลือก
     }
   break;                        //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch

   case 'B':                    //ตัวแปร _move คือ B สั่งให้มอเตอร์ ถอยหลัง
     if(Mot_N == 'L')              //ตรวจสอบเป็นมอเตอร์ ซ้าย ใช่ ให้ทำงานใน {....} ไม่ใช้ไปที่ else
     {
       analogWrite(XIA, adj_speed);  //เขียนคำสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
       digitalWrite(YIB, HIGH);      //เขียนคำสั่ง HIGH ออกไปที่ Pin ที่เลือก
     }
     else                         //การทำงานของมอเตอร์ ขวา
     {
analogWrite(YIB, adj_speed);  //เขียนคำสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
       digitalWrite(XIA, LOW);       //เขียนคำสั่ง LOW ออกไปที่ Pin ที่เลือก
     }
   break;                        //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch

   case 'S':                    //ตัวแปร _move คือ S สั่งให้มอเตอร์ หยุด
     digitalWrite(XIA, LOW);       //เขียนคำสั่ง LOW ออกไปที่ Pin ที่เลือก
     digitalWrite(YIB, LOW);       //เขียนคำสั่ง LOW ออกไปที่ Pin ที่เลือก
   break;                        //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
 }
}
void forward(int spd)     //ฟังก์ชั่น มอเตอร์ เดินหน้า และตัวแปร spd กำหนดค่าความเร็ว
{
  motor('L', 'F', spd);   //มอเตอร์ ซ้าย เดินหน้า และค่า ความเร็ว
  motor('R', 'F', spd);   //มอเตอร์ ขวา เดินหน้า และค่า ความเร็ว
}
```

```
void backward(int spd)    //ฟังก์ชั่น มอเตอร์ เดินหน้า และตัวแปร spd กำหนดค่าความเร็ว
{
  motor('L', 'B', spd);   //มอเตอร์ ซ้าย ถอยหลัง และค่า ความเร็ว
  motor('R', 'B', spd);   //มอเตอร์ ขวา ถอยหลัง และค่า ความเร็ว
}

void right(int spd)       //ฟังก์ชั่น มอเตอร์ เลี้ยวขวา และตัวแปร spd กำหนดค่าความเร็ว
{
  motor('L', 'F', spd);   //มอเตอร์ ซ้าย เดินหน้า และค่า ความเร็ว
  motor('R', 'B', spd);   //มอเตอร์ ขวา ถอยหลัง และค่า ความเร็ว
}
void left(int spd)        //ฟังก์ชั่น มอเตอร์ เลี้ยงซ้าย และตัวแปร spd กำหนดค่าความเร็ว
{
  motor('L', 'B', spd);   //มอเตอร์ ซ้าย ถอยหลัง และค่า ความเร็ว
  motor('R', 'F', spd);   //มอเตอร์ ขวา เดินหน้า และค่า ความเร็ว
}
void STOP()               //ฟังก์ชั่น มอเตอร์ หยุด
{
  motor('L', 'S', 0);     //มอเตอร์ ซ้าย ส่งค่า 0 เพื่อหยุดการทำงานมอเตอร์
  motor('R', 'S', 0);     //มอเตอร์ ขวา ส่งค่า 0 เพื่อหยุดการทำงานมอเตอร์
}
void setup()
{
  pinMode(LIA,OUTPUT);  //กำหนดให้ LIA เป็น output
  pinMode(LIB,OUTPUT);  //กำหนดให้ LIB เป็น output
  pinMode(RIA,OUTPUT);  //กำหนดให้ RIA เป็น output
  pinMode(RIB,OUTPUT);  //กำหนดให้ RIB เป็น output
}

void loop()
{
  forward(50);        //คำสังเดินหน้าด้วยความเร็ว 50%
  delay(500);         //เป็นเวลา 0.5 วินาที
  STOP();             //คำสังหยุด
  delay(1000);        //หยุดเป็นเวลา 1 วินาที

  backward(50);       //คำสังถอยหลังด้วยความเร็ว 50%
  delay(500);         //เป็นเวลา 0.5 วินาที
  STOP();             //คำสั่งหยุด
  delay(1000);        //หยุดเป็นเวลา 1 วินาที

  right(80);          //คำสัง เลี้ยวขวา ด้วยความเร็ว 80%
  delay(500);         //เป็นเวลา 0.5 วินาที
  STOP();             //คำสังหยุด
  delay(1000);        //หยุดเป็นเวลา 1 วินาที

  left(80);           //คำสัง เลี้ยวซ้าย ด้วยความเร็ว 80%
  delay(500);         //เป็นเวลา 0.5 วินาที
  STOP();             //คำสังหยุด
  delay(1000);        //หยุดเป็นเวลา 1 วินาที
}
```

การใช้งานเซ็นเซอร์แบบอินฟราเรด

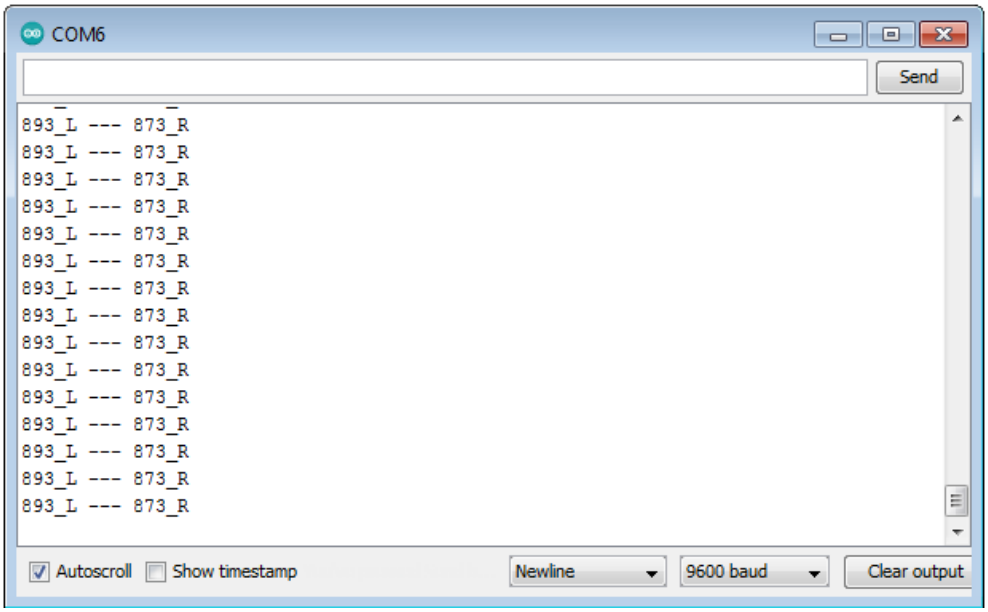


เซ็นเซอร์อินฟราเรดจะมีการจัดวงจรให้ทำงานด้วยกันอยู่สองแบบคือ เมื่อทำงานหรือมีการสะท้อนของแสงอินฟราเรดกลับมาที่ตัวรับจะให้สัญญาณเป็นแบบ LOW คือ ศูนย์โวลต์ (Active Low or ATL) หรือ ให้สัญญาณเป็น HIGH หรือ 5โวลต์ (Active High or ATH)

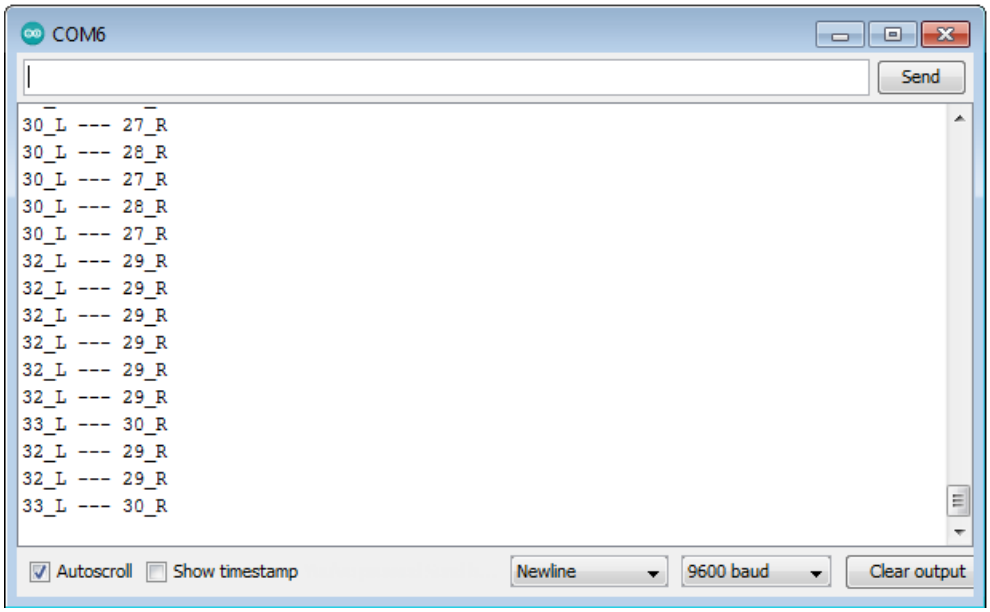


EX11.sensorInf ทดสอบการอ่านค่าสัญญาณที่ขา A1และA2 และแสดงผลออกทาง Serial Monitor

```
int sensor_L;       // ประกาศตัวแปรชื่อ sensor_L
int sensor_R;       // ประกาศตัวแปรชื่อ sensor_R
int ledPin = 13;    // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13

void setup()
{
  Serial.begin(9600);     //เปิดใช้งานพอร์ตอนุกรม baud rate 9600
  pinMode(ledPin,OUTPUT); //กำหนดให้ ledPin เป็น output
}

void loop()
{
  sensor_L = analogRead(A2);//อ่านค่าอนาล็อกตำแหน่งที่ A2 มาเก็บไว้ที่ sensor_L
  sensor_R = analogRead(A1);//อ่านค่าอนาล็อกตำแหน่งที่ A1 มาเก็บไว้ที่ sensor_R
  Serial.print(sensor_L); //แสดงค่า sensor_L ออกทางหน้าจอ
  Serial.print("_L --- "); //พิมพ์อักษร L ต่อจากค่า sensor_L ออกทางหน้าจอ
  Serial.print(sensor_R); //แสดงค่า sensor_R ออกทางหน้าจอ
  Serial.println("_R"); //พิมพ์อักษร R ต่อจากค่า sensor_R ออกทางหน้าจอ
  delay(300);
}
```



รูปแสดงค่าเซ็นเซอร์ อินฟราเรดด้าน ซ้าย และ ขวา เมื่ออยู่บนพื้นดำ ค่าจะมาก



รูปแสดงค่าเซ็นเซอร์ อินฟราเรดด้าน ซ้าย และ ขวา เมื่ออยู่บนพื้นขาว ค่าจะน้อย

สังเกตการณ์แสดงผล ถ้าเซ็นเซอร์อยู่บนพื้นดำหรือยกให้สูงขึ้นค่าที่อ่านได้จะมาก และถ้าอยู่บนพื้นขาว ค่าที่อ่านได้จะน้อย

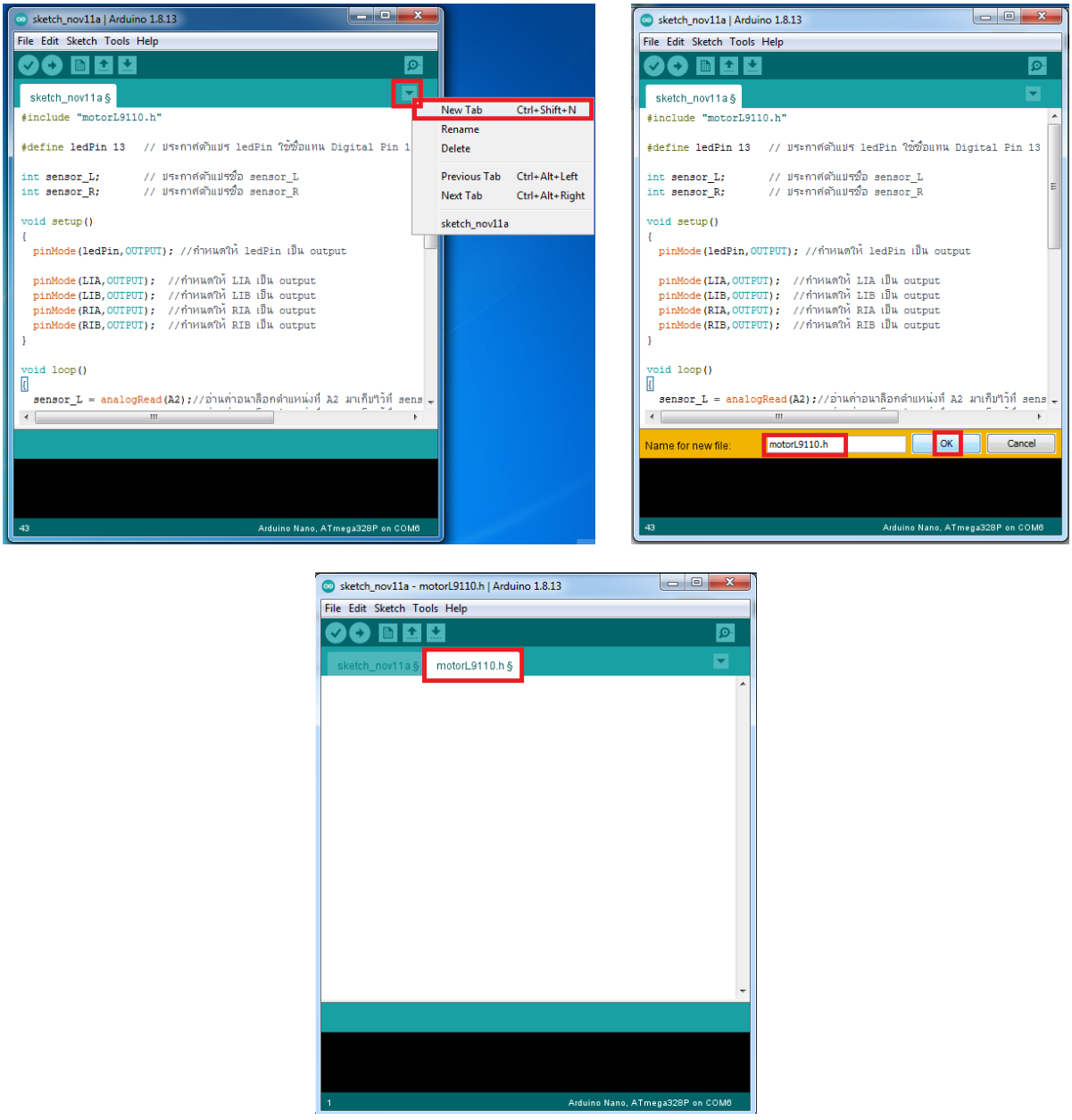
EX12. LineTracking ทดสอบการทำงานของหุ่นยนต์เดินตามเส้น

การทดสอบนี้เราจะทำการเขียนโค้ดเซ็นเซอร์อินฟราเรดให้ทำงานพร้อมกับการควบคุมมอเตอร์ให้หุ่นยนต์เดินไปตามเส้นสีดำ หลักการก็คือ เมื่อเซ็นเซอร์ด้านใดเข้าไปที่เส้นดำเราจะทำให้มอเตอร์ด้านนั้นหยุดทำงาน มาลองดูโค้ดกัน

```
#include "motorL9110.h" // เรียกใช้ ไลบรารี่ motorL9110.h

#define ledPin 13   // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13

int sensor_L;       // ประกาศตัวแปรชื่อ sensor_L
int sensor_R;       // ประกาศตัวแปรชื่อ sensor_R

void setup()
{
  pinMode(ledPin,OUTPUT); //กำหนดให้ ledPin เป็น output

  pinMode(LIA,OUTPUT);  //กำหนดให้ LIA เป็น output
  pinMode(LIB,OUTPUT);  //กำหนดให้ LIB เป็น output
  pinMode(RIA,OUTPUT);  //กำหนดให้ RIA เป็น output
  pinMode(RIB,OUTPUT);  //กำหนดให้ RIB เป็น output
}

void loop()
{
  sensor_L = analogRead(A2);//อ่านค่าอนาล็อกตำแหน่งที่ A2 มาเก็บไว้ที่ sensor_L
  sensor_R = analogRead(A1);//อ่านค่าอนาล็อกตำแหน่งที่ A1 มาเก็บไว้ที่ sensor_R

  if((sensor_L<100)&&(sensor_R<100)) //ตรวจสอบเซ็นเซอร์ทั้งสองอยู่บนพื้นขาวหรือไม่ ใช้ให้หุ่นยนต์เดินหน้า
  {
    digitalWrite(ledPin,HIGH);  // ให้ LED D13 ติด
    forward(100); //คำสั่งเดินหน้าด้วยความเร็ว 50%
  }
else if((sensor_L>100)&&(sensor_R<100)) //ตรวจสอบเซ็นเซอร์ซ้ายอยู่บนพื้นขาวและขวาอยู่บนเส้นดำถ้าใช้ให้หุ่นยนต์เลี้ยงขวา
  {
    digitalWrite(ledPin,LOW);   // ให้ LED D13 ดับ
    left(80); //คำสั่ง เลี้ยวซ้าย ด้วยความเร็ว 80%
  }
else if((sensor_L<100)&&(sensor_R>100)) //ตรวจสอบเซ็นเซอร์ขวาอยู่บนพื้นขาวและซ้ายอยู่บนเส้นดำถ้าใช้ให้หุ่นยนต์เลี้ยงซ้าย
  {
    digitalWrite(ledPin,LOW);   // ให้ LED D13 ดับ
    right(80); //คำสั่ง เลี้ยวขวา ด้วยความเร็ว 80%
  }
```

```
else //เซ็นเซอร์ทั้งสองอยู่บนพื้นดำ ให้หยุดการทำงานของมอเตอร์
  {
    digitalWrite(ledPin,LOW);   // ให้ LED D13 ดับ
    STOP(); //หยุดการทำงานของมอเตอร์
  }
}
```

ในโค้ดจะเห็นว่าการเรียกใช้ library ของ motor ที่ชื่อ motorL9110.h จริงแล้วมันก็คือโปรแกรมการทำงานของมอเตอร์ในตัวอย่างก่อนหน้านี้นั่นเอง เพื่อที่เราจะได้ไม่ต้องเขียนคำสั่งเดิมๆในโปรแกรมนี้อีกที่ใช้ควบคุมมอเตอร์เราจึงมาสร้างให้เป็น library เพื่อที่เราจะได้ลดการเขียนคำสั่งมอเตอร์ที่ให้ยุ่งยากและอาจเกิดการผิดพลาดได้ มาดูขั้นตอนกัน เมื่อเราเขียนโปรแกรมในตัวอย่างที่ EX12 เสร็จแล้ว ให้คลิกที่มุมขวาของหน้าโปรแกรมตามรูป เลือกที่ New Tap จะมีช่องข้อความแถบสีเหลืองให้เราตั้งชื่ออะไรก็ได้ และตามด้วย (.h) แต่ในที่นี้จะตั้งว่า motorL9110.h เมื่อเสร็จให้กด ok ก็จะปรากฏแถบหน้าต่างใหม่ขึ้นมา พอได้หน้าต่างใหม่มา ให้เอาโค้ดคำสั่งการทำงานของมอเตอร์มาใส่ เท่านี้เราก็ได้ library มาใช้งานแล้วครับ



รูปแสดงไว้การเปิดใช้งานการสร้าง library

โค้ด motorL9110.h

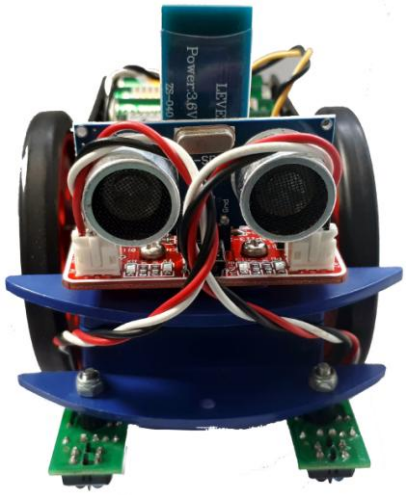
```
/*************************************************************************
* motor L9001.h
 *************************************************************************/
//int LIA = 5;  //ประกาศตัวแปรชื่อ LIA ใช้ชื่อแทน Digital Pin 5 PWM
//int LIB = 4;  //ประกาศตัวแปรชื่อ LIB ใช้ชื่อแทน Digital Pin 4
//int RIA = 7;  //ประกาศตัวแปรชื่อ RIA ใช้ชื่อแทน Digital Pin 7
//int RIB = 6;  //ประกาศตัวแปรชื่อ RIB ใช้ชื่อแทน Digital Pin 6 PWM

#define LIA 5
#define LIB 4
#define RIA 7
#define RIB 6

int XIA;      //ประกาศตัวแปรชื่อ XIA
int YIB;      //ประกาศตัวแปรชื่อ YIB

void motor(char Mot_N, char _move, int adj_speed) //ฟังก์ชั่นการควบคุมการทำงานชอง motor
{
  switch(Mot_N) //ตรวจสอบตัวแปร Mot_N เป็นการเลือกการทำงานของมอเตอร์ ซ้าย หรือ ขวา
  {
    case 'L':     //รับค่า L คือเลือกมอเตอร์ ซ้าย
      XIA = LIA;  //ให้ตัวแปรชื่อ XIA แทนขา Pin 5 PWM
      YIB = LIB;  //ให้ตัวแปรชื่อ YIB แทนขา Pin 4

      if(_move == 'B') adj_speed = 255-((255*adj_speed)/100);  //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255 และ กลับค่า
      else    adj_speed = (255*adj_speed)/100;                 //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255

      break;      //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
    case 'R':     //รับค่า R คือเลือกมอเตอร์ ขวา
      XIA = RIA;  //ให้ตัวแปรชื่อ XIA แทนขา Pin 7
      YIB = RIB;  //ให้ตัวแปรชื่อ YIB แทนขา Pin 6 PWM

      if(_move == 'F') adj_speed = 255-((255*adj_speed)/100);  //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255 และ กลับค่า
      else    adj_speed = ((255*adj_speed)/100);               //แปลงค่า เปอร์เซ็น ให้เป็นค่า 0-255
    break;        //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
  }

  switch(_move)   //ตรวจสอบตัวแปร _move เป็นการเคลื่อนที่ของมอเตอร์แบบ เดินหน้า ถอยหลัง หรือ หยุด
  {
    case 'F':                     //ตัวแปร _move คือ F สั่งให้มอเตอร์ เดินหน้า
      if(Mot_N == 'L')                //ตรวจสอบเป็นมอเตอร์ ซ้าย ใช่ ให้ทำงานใน {....} ไม่ใช้ไปที่ else
      {
        analogWrite(XIA, adj_speed);  //เขียนค่าสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
        digitalWrite(YIB, LOW);       //เขียนค่าสั่ง LOW ออกไปที่ Pin ที่เลือก
      }
      else                          //การทำงานของมอเตอร์ ขวา
      {
        analogWrite(YIB, adj_speed);  //เขียนค่าสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
        digitalWrite(XIA, HIGH);      //เขียนค่าสั่ง LOW ออกไปที่ Pin ที่เลือก
      }
    break;                          //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch

    case 'B':                     //ตัวแปร _move คือ B สั่งให้มอเตอร์ ถอยหลัง
      if(Mot_N == 'L')                //ตรวจสอบเป็นมอเตอร์ ซ้าย ใช่ ให้ทำงานใน {....} ไม่ใช้ไปที่ else
      {
        analogWrite(XIA, adj_speed);  //เขียนค่าสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
        digitalWrite(YIB, HIGH);      //เขียนค่าสั่ง HIGH ออกไปที่ Pin ที่เลือก
      }
      else                          //การทำงานของมอเตอร์ ขวา
      {
        analogWrite(YIB, adj_speed);  //เขียนค่าสั่ง PWM ตามความเร็วที่อยู่ในตัวแปร adj_speed ออกไปที่ Pin ที่เลือก
        digitalWrite(XIA, LOW);       //เขียนค่าสั่ง LOW ออกไปที่ Pin ที่เลือก
      }
    break;                          //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch

    case 'S':                     //ตัวแปร _move คือ S สั่งให้มอเตอร์ หยุด
      digitalWrite(XIA, LOW);       //เขียนค่าสั่ง LOW ออกไปที่ Pin ที่เลือก
      digitalWrite(YIB, LOW);       //เขียนค่าสั่ง LOW ออกไปที่ Pin ที่เลือก
    break;                          //หยุดการตรวจสอบเงื่อนไข ออกจากคำสั่ง switch
  }
}
```

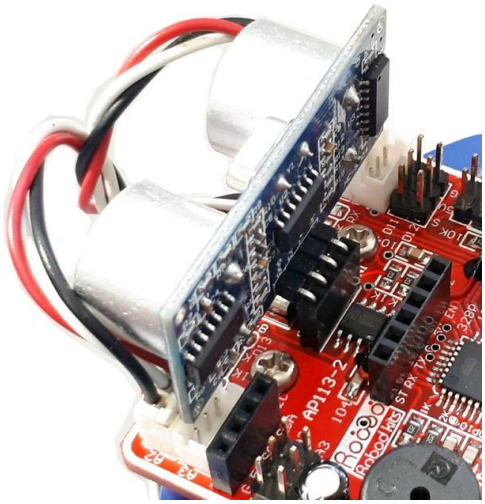
```
void forward(int spd)     //ฟังก์ชั่น มอเตอร์ เดินหน้า
{
  motor('L', 'F', spd);
  motor('R', 'F', spd);
}

void backward(int spd)
{
  motor('L', 'B', spd);
  motor('R', 'B', spd);
}

void right(int spd)
{
  motor('L', 'F', spd);
  motor('R', 'B', spd);
}
void left(int spd)
{
  motor('L', 'B', spd);
  motor('R', 'F', spd);
}
void STOP()
{
  motor('L', 'S', 0);
  motor('R', 'S', 0);
}
```

Ultrasonic HC-SR04 Module กับการใช้งานเพื่อการวัดระยะทาง เป็น Sensor ใช้วัดระยะทางด้วยคลื่นเสียงย่านอัลตร้าโซนิค โดนการส่งคลื่นอัลตร้าโซนิคสะท้อนไปที่วัตถุ และส่งกลับมาที่ตัวรับ จากนั้นจะคำนวณหาระยะเวลาที่คลื่นอัลตร้าโซนิคได้กระทบวัตถุแล้วกลับมายัง Sensor ก็จะได้ค่า ระยะทางของวัตถุนั้น

การต่อใช้งาน ดังนี้

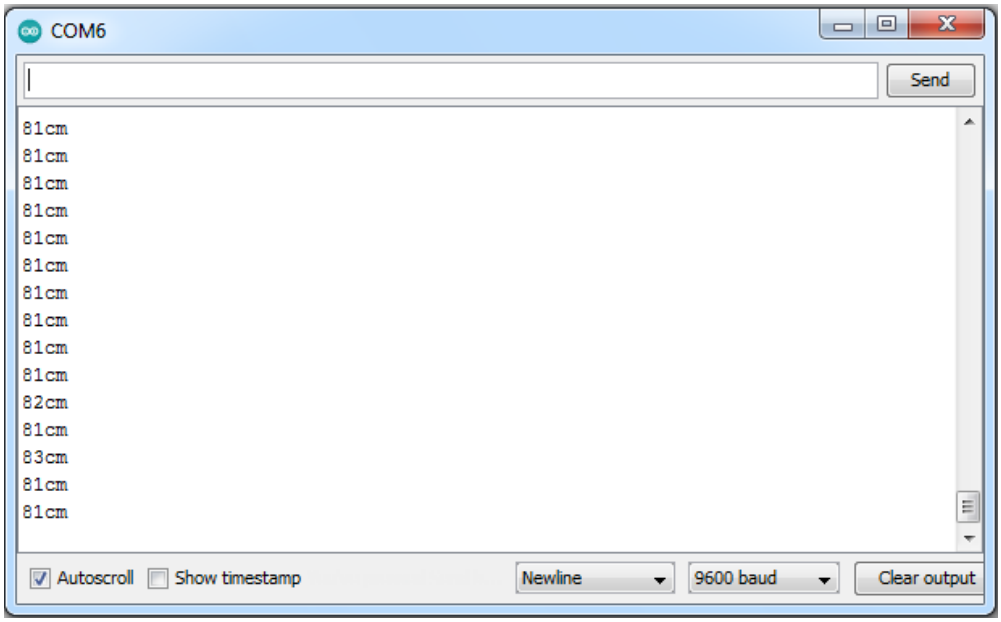
AVR NANOBOT	**HC-SR04**
5V	VCC
9	trig
10	echo
Gnd	Gnd





EX13.Ultrasonic ทดสอบวัดระยะทางและแสดงผลที่หน้าจอ Serial monitor

```
// Vcc - 5v
// Gnd - Gnd
// Trig - 9
// Echo - 10

#define ledPin 13    // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
#define trigPin 9   // ประกาศตัวแปร trigPin ใช้ชื่อแทน Digital Pin 9
#define echoPin 10   // ประกาศตัวแปร echoPin ใช้ชื่อแทน Digital Pin 10
void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);    //กำหนดให้ ledPin เป็น output
  pinMode(trigPin, OUTPUT);   //กำหนดให้ trigPin เป็น output
  pinMode(echoPin, INPUT);    //กำหนดให้ echoPin เป็น input
}
void loop()
{
  long duration, cm;  //ประกาศตัวแปรชือ duration และ cm สำหรับคำนวนค่าระยะทาง

  digitalWrite(trigPin, LOW);   //เขียนคำสั่ง LOW ออกไปที่ trigPin
  delayMicroseconds(2);         //หน่วงเวลาไว้ 2 ไมโครวินาที
  digitalWrite(trigPin, HIGH);   //เขียนคำสั่ง HIGH ออกไปที่ trigPin
  delayMicroseconds(5);         //หน่วงเวลาไว้ 5 ไมโครวินาที
  digitalWrite(trigPin, LOW);   //เขียนคำสั่ง LOW ออกไปที่ trigPin
  duration = pulseIn(echoPin, HIGH);   //รับค่าสัญญาณที่ขา echoPin มาเก็บไว้ที่ตัวแปร duration

  cm = microsecondsToCentimeters(duration); //นำค่าที่ได้มาคำนวนแล้วเก็บไว้ที่ตัวแปร cm

  // แสดงค่าที่อ่านได้ออกทางหน้าจอ Serial Monitor
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);

  // ตรวจสอบถ้าระยะทางนอยกว่า 20 เซ็นติเมตร ให้ ledPin ติด และถ้ามากกว่า ให้ ledPin ดับ
  if(cm<20) digitalWrite(ledPin, HIGH);
  else digitalWrite(ledPin, LOW);
}
long microsecondsToCentimeters(long microseconds)
{
  //ความเร็วของเสียง 340 m / s หรือ 29 microseconds ต่อ cm
  // ปิงเดินทางออกไปและกลับดังนั้นเพื่อหาระยะทางของ
  // วัตถุเราใช้เวลาครึ่งหนึ่งของระยะทางที่เดินทาง
  return microseconds / 29 / 2;
}
```

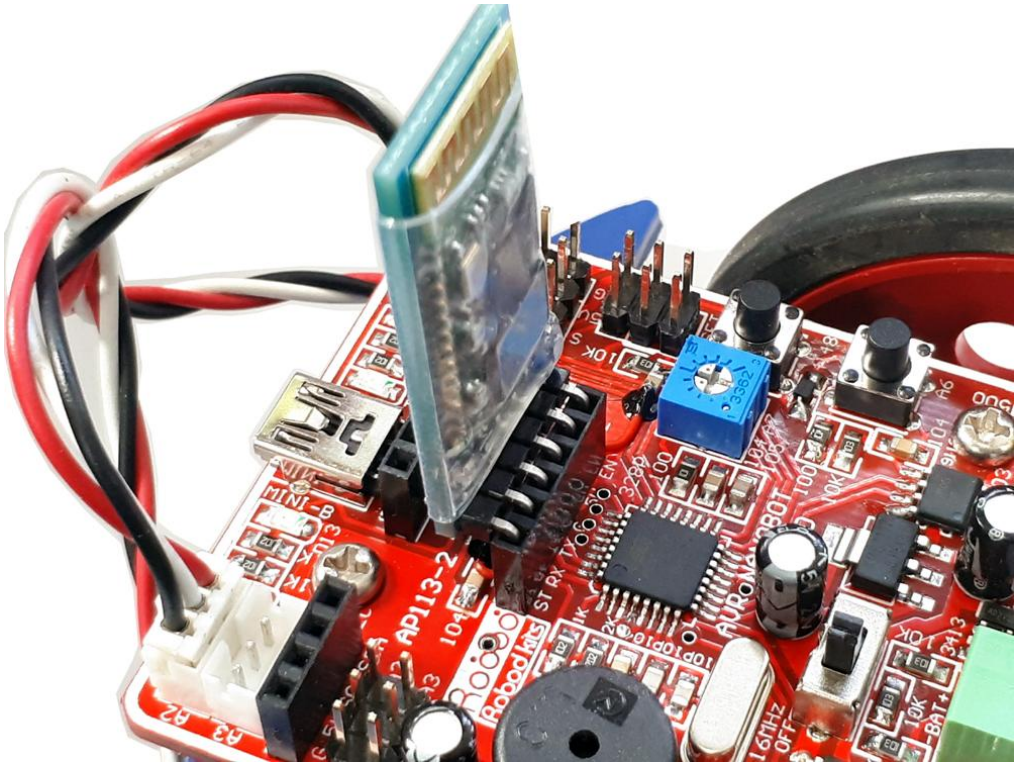


รูปแสดงค่าการวัดค่าระยะทางของอัลตร้าโซนิค

EX14.UltrasonicZumo การทดสอบการทำงานแบบเจอวัตถุในระยะให้หุ่นยนต์วิ่งชน

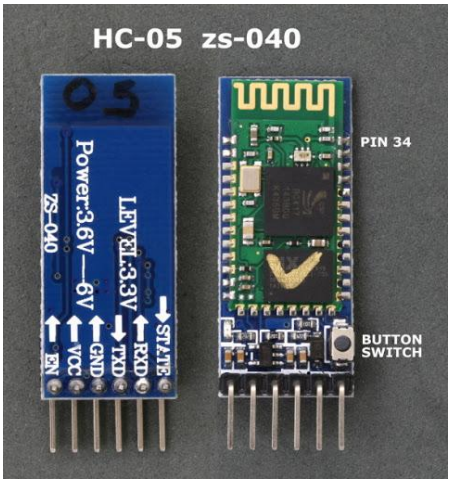
```
// Vcc - 5v
// Gnd - Gnd
// Trig - 9
// Echo - 10
#include "motorL9110.h"

#define ledPin 13    // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13
#define trigPin 9   // ประกาศตัวแปร trigPin ใช้ชื่อแทน Digital Pin 9
#define echoPin 10   // ประกาศตัวแปร echoPin ใช้ชื่อแทน Digital Pin 10

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);    //กำหนดให้ ledPin เป็น output
  pinMode(trigPin, OUTPUT);   //กำหนดให้ trigPin เป็น output
  pinMode(echoPin, INPUT);    //กำหนดให้ echoPin เป็น input
}

void loop()
{
  long duration, cm;  //ประกาศตัวแปรชือ duration และ cm สำหรับคำนวนค่าระยะทาง
digitalWrite(trigPin, LOW);   //เขียนค่าสั่ง LOW ออกไปที่ trigPin
  delayMicroseconds(2);         //หน่วงเวลาไว้ 2 ไมโครวินาที
  digitalWrite(trigPin, HIGH);   //เขียนค่าสั่ง HIGH ออกไปที่ trigPin
  delayMicroseconds(5);         //หน่วงเวลาไว้ 5 ไมโครวินาที
  digitalWrite(trigPin, LOW);   //เขียนค่าสั่ง LOW ออกไปที่ trigPin
  duration = pulseIn(echoPin, HIGH);   //รับค่าสัญญาณที่ขา echoPin มาเก็บไว้ที่ตัวแปร duration
 cm = microsecondsToCentimeters(duration); //นำค่าที่ได้มาคำนวนแล้วเก็บไว้ที่ตัวแปร cm

  // แสดงค่าที่อ่านได้ออกทางหน้าจอ Serial Monitor
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);

  //ตรวจสอบถ้าระยะทางนอยกว่า 20 เซ็นติเมตร ให้ ledPin ติด พร้อมกับให้หุ่นยนต์เดินหน้า
  //และถ้ามากกว่า ให้ ledPin ดับ และหุ่นยนต์หยุดเดิน
  if(cm<20)
  {
    digitalWrite(ledPin, HIGH);
    forward(100); //คำสังเดินหน้าด้วยความเร็ว 50%
  }

  else
  {
    digitalWrite(ledPin, LOW);
    STOP(); //หยุดการทำงานของมอเตอร์
  }
}

long microsecondsToCentimeters(long microseconds)
{
  //ความเร็วของเสียง 340 m / s หรือ 29 microseconds ต่อ cm
  // ปิงเดินทางออกไปและกลับดังนั้นเพื่อหาระยะทางของ
  // วัตถุเราใช้เวลาครึ่งหนึ่งของระยะทางที่เดินทาง
  return microseconds / 29 / 2;
}
```

Bluetooth Module HC-05



HC-05 เป็นโมดูล Bluetooth ที่ใช้งานในการเชื่อมต่อกับสมาร์ทดีไวซ์ต่างๆ ให้สมาร์ทดีไวซ์สามารถสื่อสารกับ ไมโครคอนโทรเลอร์ Arduino AVR ได้ ผ่านทาง Serial port โมดูลรุ่น HC-05 สามารถตั้งให้ใช้งานเป็นได้ทั้งโหมด Master (ให้อุปกรณ์อื่นมาเชื่อมต่อ) และโหมด Slave (เชื่อมต่อกับอุปกรณ์อื่น) การตั้งค่าต่างๆ เช่น ชื่ออุปกรณ์ รหัสผ่าน ทำได้ผ่าน AT Command ซึ่งจะต้องมีการต่อขาพิเศษเพื่อให้โมดูลเข้าโหมดการตั้งค่า หรือกดปุ่มบนโมดูลค้างไว้ ในบทความนี้ จะสอนใช้ Bluetooth Module HC-05 เชื่อมต่อกับ Application ในมือถือ โดยการรับคำสั่งการกดปุ่มต่างๆในมือถือ ส่งเข้า AVR NANOBOT

AVR NANOBOT	HC-05
EN	EN
Vcc	Vcc
GND	GND
D2	RXD
D3	TXD
ST	ST

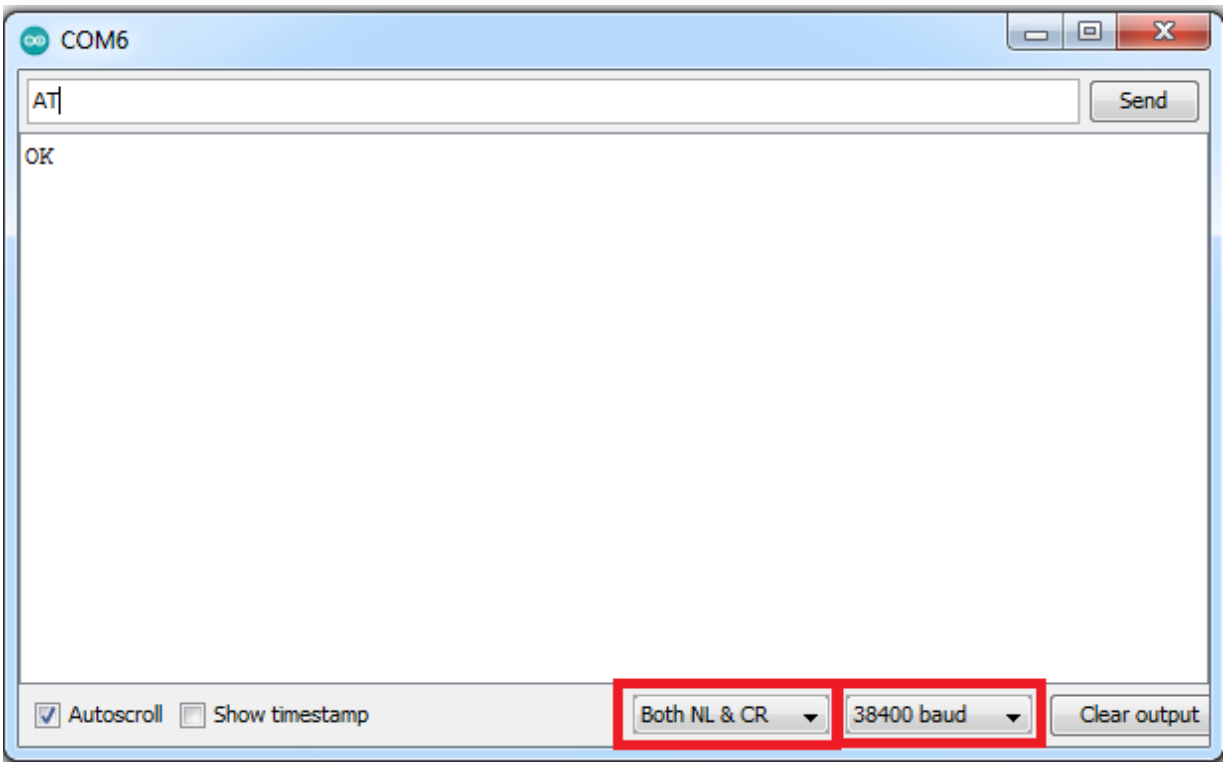


ขั้นตอนการต่อใช้งาน ให้ปิดสวิตช์ POWER และดึงสาย USB ออกก่อน จากนั้นให้เราต่อ Blutooth เข้าที่ตำแหน่งของ CN2 เสียบขาให้ตรงกับตำแหน่งที่บอกไว้บนบอร์ด ต่อมาให้กด สวิตช์ที่บอร์ด Bluetooth ค้างไว้ และเปิดสวิช์ POWER จะสังเกตเห็นไฟ LED บนบอร์ด Bluetooth ติดกระพริบ แบบ ช้าๆ แต่ถ้าติดกระพริบเร็ว ให้ปิดสวิตช์ และทำตามขั้นตอนข้างต้นใหม่ เมื่อได้แล้วต่อไปให้เสียบสาย USB เข้าที่บอร์ด และทำการโหลดโปรแกรมตัวอย่างที่ 15 ได้เลย

EX15.Bluetooth การทดลองการตั้งค่า AT Command

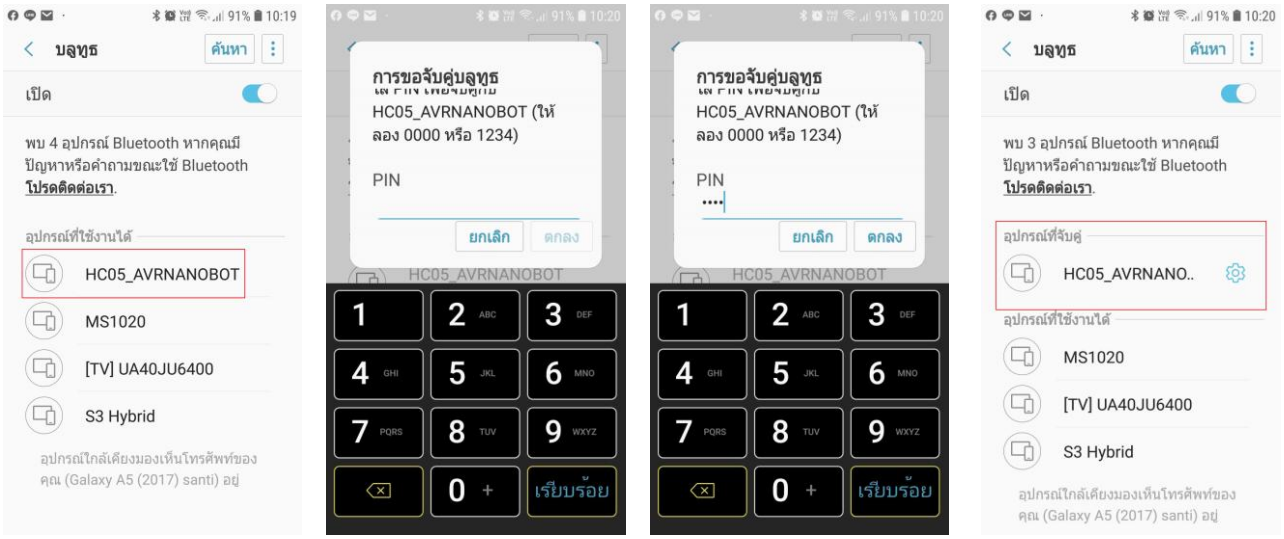
```
/*
 * RX is digital pin 2 (connect to TX of other device)
 * TX is digital pin 3 (connect to RX of other device)
 */

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX
void setup()
{
Serial.begin(38400);
while (!Serial) ;
mySerial.begin(38400);
}
void loop()
{
if (mySerial.available())
Serial.write(mySerial.read());
if (Serial.available())
mySerial.write(Serial.read());
}
```

เมื่อโหลดโปรแกรมเสร็จแล้วให้เปิด Serial monitor ที่โปรแกรม arduino ขึ้นมา ตั้ง Both NL & CR และ 38400 baud ตามรูป



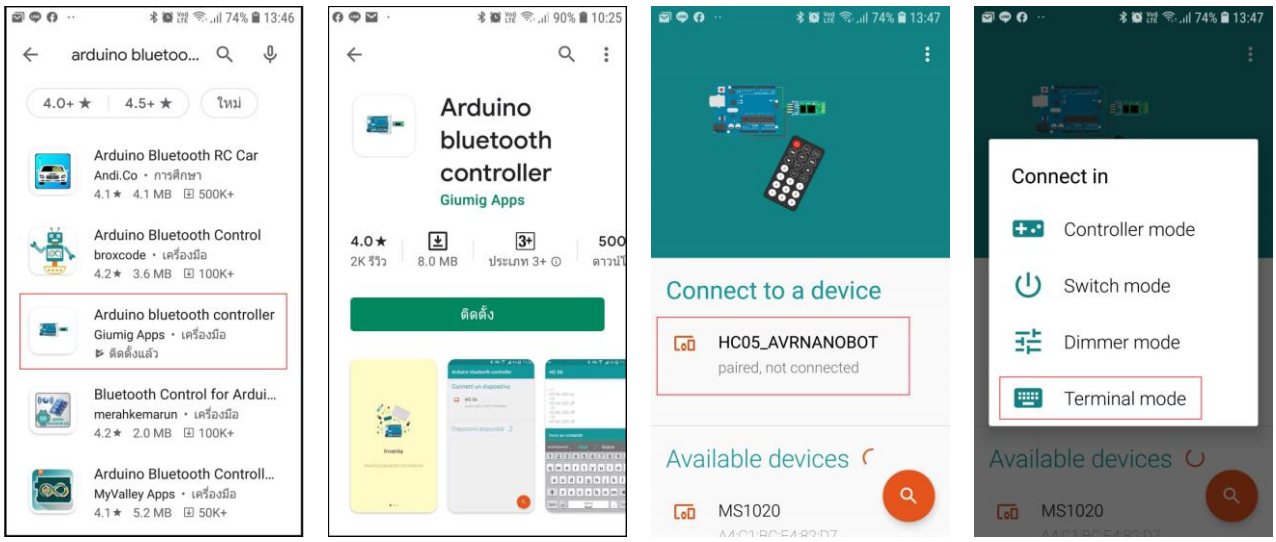
เมื่อเซตค่าที่หน้าต่าง Serial monitor เสร็จแล้วให้ลองพิมพ์ AT ในช่องด้านบนและกด ENTER จะได้รับข้อความตอบกลับว่า OK ต่อไปจะมาลองเปลี่ยนชื่อกันให้พิมพ์ AT+NAME=ตามด้วยชื่อ เช่น AT+NAME=HC05_AVRNANOBOT และกด ENTER หากสำเร็จจะตอบกลับมาว่า OK คำสั่ง AT Command ยังมีอีกหลายคำสั่งสามารถดูได้จาก https://www.estudioelectronica.com/wp-content/uploads/2018/09/istd016A.pdf

เมื่อทำการตั้งชื่อแล้วต่อไปให้ ดึงสาย USB ออก และปิดสวิตช์ POWER จากนั้นให้เปิดสวิตช์ POWER และเสียบสาย USB อีกครั้ง โดยที่ไม่ต้องกดสวิตช์ที่ Bluetooth จะเห็นไฟ LED ที่ Bluetooth ติดกระพริบเร็ว ต่อมาเปิด Bluetooth ที่มือถือและค้นหาดู จะเจอ Bluetooth ที่เราตั้งชื่อไว้ให้เรากดเชื่อมต่อและใส่รหัสผ่านเป็น 0000 หรือ 1234 ก็จะทำการเชื่อต่อให้ตามรูปด้านล่าง

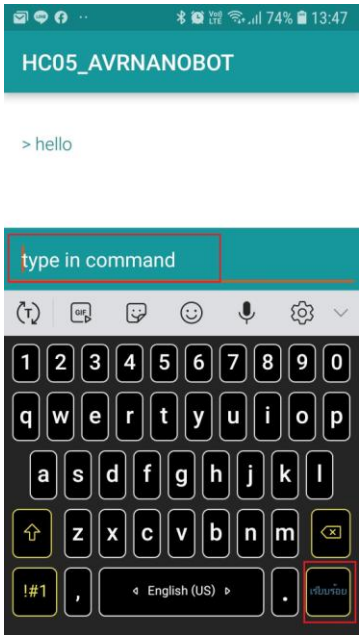


ต่อไปนี้เราจะมารองรับส่งข้อความจากมือถือมาที่ Serial monitor ของ Arduino กัน ให้เราแก้โค้ดของตัวอย่างที่ EX15 ในฟังชั่น setup() ในบรรทัดของ Serial.begin(38400); และ mySerial.begin(38400); เดิม baud rate อยู่ที่ 38400 ให้เปลี่ยนเป็น 9600 ทั้งหมด และ โหลดโปรแกรมลงบอร์ด

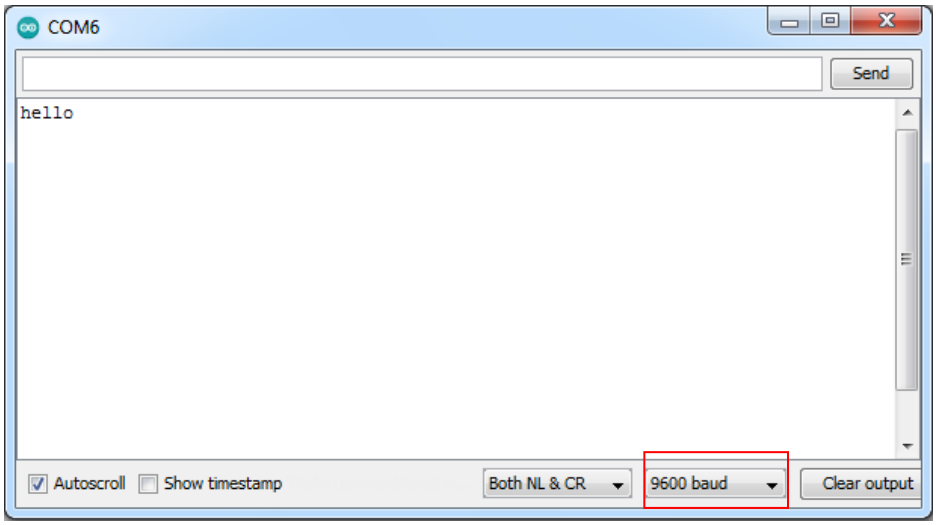
ที่มือถือให้โหลด Application ที่ชื่อ Arduino Bluetooth control เมื่อโหลดเสร็จให้เปิดโปรแกรมและทำการเชื่อต่อกับ Bluetooth ของเรา



จากรูปด้านบน ในหน้าค้นหา ตามมาด้วยหน้าการติดตั้ง ถัดมาคือการเปิดแอบ และเลือก Bluetooth ที่เราตั้งชื่อไว้ และสุดท้ายเลือกการใช้งานเป็น Terminal mode จะได้หน้าต่างตามรูปด้านล่าง



ในรูปซ้ายมือ ให้พิมพ์ข้อความอะไรก็ได้ในช่อง Type in command ในที่นี้จะพิมพ์คำว่า hello และกดที่ปุ่มด้านล่างขวา ขอความจะไปปรากฏที่หน้าต่างด้านบนว่า >hello กลับมาที่หน้าต่าง Serial monitor ของ Arduino เซ็ทค่าตามรูปด้านล่าง



จะสังเกตเห็นว่าที่หน้าต่าง Serial monitor จะแสดงขอความ hello เช่นกัน ให้เราลองพิมพ์อะไรก็ได้ที่ Serial monitor และกด Enter ข้อความก็จะไปปรากฏที่ แอบบนมือถือเช่นกัน

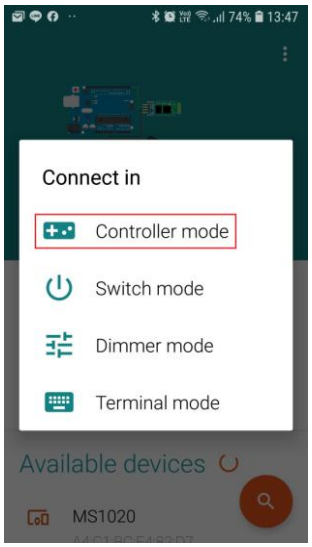
ตัวอย่างต่อไปเราจะลองควบคุมหุ่นยนต์ด้วยแอบบนมือถือกัน

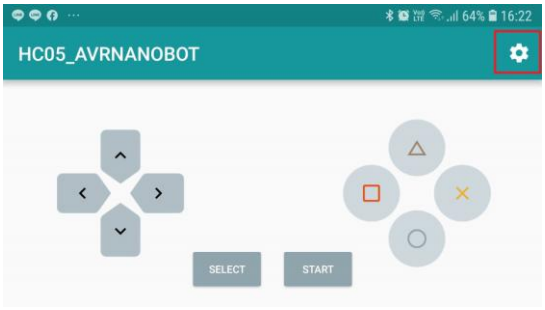
EX16. RemoteCarControl จะเป็นการทดลองการควบคุมหุ่นยนต์ด้วยมือถือผ่าน Bluetooth

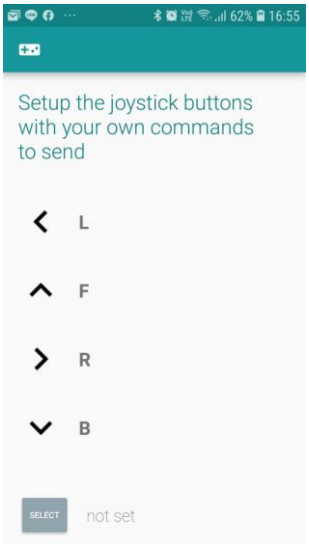
```
#include "motorL9110.h"

#define ledPin 13   // ประกาศตัวแปร ledPin ใช้ชื่อแทน Digital Pin 13

#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX

char data = 0;

void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600);

  pinMode(ledPin,OUTPUT); //กำหนดให้ ledPin เป็น output

  pinMode(LIA,OUTPUT);  //กำหนดให้ LIA เป็น output
  pinMode(LIB,OUTPUT);  //กำหนดให้ LIB เป็น output
   pinMode(RIA,OUTPUT);  //กำหนดให้ RIA เป็น output
  pinMode(RIB,OUTPUT);  //กำหนดให้ RIB เป็น output
}
void loop()
{
  if (mySerial.available() > 0)
  {
    data = mySerial.read();    //Read the incoming data and store it into variable data
    Serial.print(data);        //Print Value of data in Serial monitor
    Serial.print("\n");        //New line


    if (data == 'S')
    {
      digitalWrite(ledPin,LOW);  // ให้ LED D13 ดับ
      STOP(); //หยุดการทำงานของมอเตอร์
    }
```

```
    else if (data == 'F')
    {
      digitalWrite(ledPin,HIGH);  // ให้ LED D13 ติด
      forward(60); //คำสั่งเดินหน้าด้วยความเร็ว 50%
    }

    else if (data == 'B')
    {
      digitalWrite(ledPin,HIGH);   // ให้ LED D13 ติด
      backward(60); //คำสั่ง เลี้ยวขวา ด้วยความเร็ว 80%
    }

    else if (data == 'L')
    {
      digitalWrite(ledPin,HIGH);   // ให้ LED D13 ติด
      left(30); //คำสั่ง เลี้ยวซ้าย ด้วยความเร็ว 80%
    }

    else if (data == 'R')
    {
      digitalWrite(ledPin,HIGH);   // ให้ LED D13 ติด
      right(30); //คำสั่ง เลี้ยวขวา ด้วยความเร็ว 80%
    }
  }
}
```

มาที่มือถือให้ออกจากโหมด Terminal mode แล้วเข้าที่ Controller mode จากนั้นเข้าไปตั้งค่าที่มุมขวา ใส่ตัวอักษร F ที่หัวลูกศรขึ้นบน ให้เดินหน้า ใส่ L ที่หัวลูกศรทางซ้าย ให้เลี้ยวซ้าย ใส่ R ที่หัวลูกศรทางขวา ให้เลี้ยวขวา ใส่ B ที่หัวลูกศรลงล่าง ให้ถอยหลังและใส่ S ที่ปุ่ม START เพื่อหยุดมอเตอร์







เมื่อเราปรับปุ่มกดตามที่บอกแล้ว ให้ออกจากการตั้งค่าโดยกดที่รูปจอยที่มุมบนซ้ายมือ ก็จะกลับมาที่หน้าควบคุม ให้ลองกดที่ปุ่ม ขึ้นบน หุ่นยนต์จะเดินไปด้านหน้า กดปุ่มลงล่างหุ่นยนต์จะเดินถอยหลัง กดซ้ายเดินซ้าย กดขวาเดินขวา กด SYART หุ่นจะหยุด

มาถึงตอนนี้แสดงว่าเราได้เข้าใจการเขียนโปรแกรมไม่มากก็น้อยละครับ การเขียนโปรแกรมยังมีอีกมากมายที่จะให้หุ่นทำอะไร อยู่ที่ผู้เขียนอยากให้หุ่นยนต์ของเราทำอะไร

มีปัญหาปรึกษาทีมงาน www.robodkit.com

Circuit AVR NANOBOT