

คำนำ

ในช่วงหลายปีที่ผ่านมาเราจะเห็นได้ว่า ไมโครคอนโทรลเลอร์ (MCU) มีการพัฒนาไปอย่างรวดเร็วเป็นอย่างมาก โดยเฉพาะในปัจจุบันนี้ การพัฒนาจะเป็นไปอย่างก้าวกระโดด จะเห็นได้จากที่ผู้ผลิตชิพมีการเพิ่มเติมฮาร์ดแวร์แบบต่างๆมากมายรวมไว้ใน MCU เพียงตัวเดียว ทำให้โครงสร้างของ MCU เริ่มมีความซับซ้อนมากขึ้นเป็นอย่างมาก ทำให้เริ่มเป็นอุปสรรคสำหรับผู้ที่จะเริ่มต้นศึกษาเป็นอย่างมาก เพราะกว่าจะเริ่มต้นศึกษาทำความเข้าใจ MCU สักเบอร์หนึ่ง จนสามารถนำไปใช้งานได้นั้นต้องใช้เวลานานมาก เครื่องมือที่ช่วยลดระยะเวลาในการเขียนโปรแกรมใช้งาน MCU ที่มีความซับซ้อนมากๆได้อย่างรวดเร็ว คือ การใช้โปรแกรมแปลภาษาซี (C Compiler) ที่มี Library คำสั่ง สำหรับใช้สั่งงานและเข้าถึงทรัพยากรของ MCU ได้โดยง่าย แต่การเริ่มต้นด้วยภาษาซีสำหรับผู้เริ่มต้น ก็ไม่ใช่เรื่องง่ายนัก เพราะภาษาซี ที่มีความสามารถสูงๆและใช้งานง่ายๆ มักมีราคาแพงมาก ส่วนพวกภาษาซีที่แจกจ่ายให้ใช้งานกันฟรี ก็มักมีข้อจำกัด และ ใช้งานยาก แต่ภาษา C++ ของ Arduino ถือว่าเป็นคำตอบที่น่าสนใจไม่น้อยสำหรับผู้เริ่มต้นที่คิดจะเรียนรู้ MCU เพราะมีการใช้งานที่ง่ายไม่ซับซ้อน และที่สำคัญ คือ แจกจ่ายให้ใช้กันฟรีๆ

Arduino เป็นโครงการพัฒนาระบบ MCU ตระกูล AVR ด้วยภาษา C++ แบบ Open Source ที่ได้รับการเผยแพร่ออกมาในช่วงปลายปี ค.ศ.2005 นี้เองแต่กลับเป็นที่รู้จักจากผู้คนทั่วโลกอย่างรวดเร็วภายในระยะเวลาสั้นๆและปัจจุบันนี้ ก็กำลังได้รับความสนใจและยอมรับจากผู้ใช้งานทั่วโลกอย่างรวดเร็วและแพร่หลายมากขึ้นไปอีกจะดูได้จากการที่มีจำนวนสมาชิกเป็นจำนวนมากจากผู้คนในหลากหลายวงการมีตั้งแต่ระดับ ผู้เริ่มต้น ผู้สนใจทั่วไป นิสิตนักศึกษา ไปจนถึง ระดับที่เป็นนักพัฒนาที่มีความเชี่ยวชาญจากทั่วโลก ซึ่งมีจุดมุ่งหมายเดียวกันคือ มีความสนใจอยากเรียนรู้และศึกษาระบบ MCU เพื่อนำไปประยุกต์ใช้งานในรูปแบบต่างๆกัน ซึ่งจากการรวมตัวกันของผู้คน จากหลากหลายวงการนี้เอง ที่ส่งผลให้เกิดความร่วมมือกัน และมีการช่วยเหลือกันพัฒนาและปรับปรุงทั้งระบบ ฮาร์ดแวร์ และ ฮาร์ดแวร์ ไปจนถึง อุปกรณ์เชื่อมต่ออื่นๆ มากมายเพื่อนำมาใช้งานร่วมกับโปรแกรมพัฒนาของ Arduino ซึ่งทำให้มีการแลกเปลี่ยนความรู้ มีตัวอย่างการทดลองและการประยุกต์ใช้งานแบบต่างๆ จำนวนมากมาย เกิดขึ้นภายในเวลาอันสั้นจึงเป็นแรงจูงใจให้ผู้สนใจทั่วไปอยากเริ่มต้นศึกษาเรียนรู้ระบบ MCU ด้วย Arduino กันมากยิ่งขึ้น

แต่อย่างไรก็ตามข้อมูลต่างๆยังกระจุกกระจายอยู่มากซึ่งเป็นอุปสรรคสำหรับผู้เริ่มต้นใหม่ๆ ที่คิดจะศึกษาด้วยตนเองอยู่พอสมควร ทางผู้เขียนเองได้ตระหนักดีถึงปัญหานี้จึงได้พยายามรวบรวมข้อมูลที่กระจุกกระจายอยู่จากแหล่งต่างๆ นำมาผนวกเข้ากับประสบการณ์ของผู้เขียนเอง แล้วเรียบเรียงเป็นหนังสือเล่มนี้ขึ้นมา เพื่อหวังให้หนังสือเล่มนี้เป็นผู้ช่วยนำทางให้ผู้อ่านสามารถก้าวเข้าสู่โลกของ MCU ได้โดยง่าย โดยในหนังสือเล่มนี้จะมีเนื้อหา ที่เป็นข้อมูลความรู้ที่เกี่ยวข้อง และ ตัวอย่างโปรแกรมต่างๆ เพื่อให้ ผู้อ่าน สามารถใช้ทดลองปฏิบัติตามในช่วงเริ่มต้นก่อน แล้วจึงพัฒนาปรับปรุงด้วยตนเองในลำดับต่อไปได้

เอกชัย มะการ

มกราคม ๒๕๕๒

สารบัญ

บทที่1 : รู้จักกับ Arduino	1
โครงสร้างการเขียนโปรแกรมภาษาซีของ Arduino	6
ตัวแปรใน Arduino	9
ค่าสเกลต่างๆของ Arduino	10
ชนิดและประเภทของตัวแปร	11
คุณสมบัติเฉพาะของตัวแปร	12
การกำหนดชนิดของตัวแปรขึ้นมาใช้งานเอง	14
การแปลงค่าตัวแปรโดยการ cast	16
ตัวแปรแบบ array	17
ตัวแปรแบบ string	21
ตัวแปรแบบ pointer	23
ขอบเขตของตัวแปร	25
ค่าคงที่ใน Arduino	29
นิพจน์และตัวดำเนินการของ Arduino	30
การตรวจสอบเงื่อนไขในภาษาซีของ Arduino	33
คำสั่ง if	34
คำสั่ง if...else แบบ 2 ทางเลือก	35
คำสั่ง if...else แบบหลายเงื่อนไข	36
คำสั่ง for	39
คำสั่ง switch/case	40
คำสั่ง while	43
คำสั่ง do...while	44
คำสั่ง break	46
คำสั่ง continue	47
คำสั่ง goto	48
คำสั่ง return	50
โปรแกรมย่อย(function)	51

คำสั่งหรือฟังก์ชันของ Arduino	57
กลุ่มคำสั่งเกี่ยวกับ Digital I/O	58
กลุ่มคำสั่งเกี่ยวกับ Analog I/O	62
กลุ่มคำสั่งเพิ่มเติมเกี่ยวกับ I/O	66
กลุ่มคำสั่งเกี่ยวกับการหน่วงเวลา	71
กลุ่มคำสั่งทางคณิตศาสตร์และตรีโกณมิติ	75
กลุ่มคำสั่งเกี่ยวกับการสื่อสารอนุกรม	89
บทที่2 : เริ่มต้นปฏิบัติการกับ Arduino	98
แนะนำบอร์ด ET-EASY168 STAMP	100
แนะนำบอร์ด ET-BASE AVR EASY88	110
การติดตั้งโปรแกรม Arduino	120
การติดตั้ง Library ของ อีทีที ให้กับ Arduino	127
ทดสอบใช้งานโปรแกรม Arduino	128
ทดสอบใช้งานโปรแกรม Serial Monitor ของ Arduino	136
บทที่3 : Arduino กับ Software LAB	140
การใช้โปรแกรม Hyper Terminal ของ Windows	142
การทดลองที่ 1 ทดสอบการรับส่งข้อมูลด้วย RS232	144
การทดลองที่ 2 ทำความเข้าใจรูปแบบข้อมูลของ RS232	146
การทดลองที่ 3 การใช้คำสั่ง if	148
การทดลองที่ 4 การใช้คำสั่ง if...else หลายเงื่อนไข	150
การทดลองที่ 5 การใช้คำสั่งเงื่อนไข switch...case	152
การทดลองที่ 6 การใช้คำสั่ง for	154
การทดลองที่ 7 การใช้คำสั่ง while	156
การทดลองที่ 8 การใช้คำสั่ง do...while	158
การทดลองที่ 9 การใช้คำสั่ง break	160
การทดลองที่ 10 การใช้คำสั่ง continue	162
การทดลองที่ 11 การใช้คำสั่ง goto	164
การทดลองที่ 12 คุณสมบัติของตัวแปรแบบ Local	166

การทดลองที่ 13 คุณสมบัติของตัวแปรแบบ Global	168
การทดลองที่ 14 คุณสมบัติของตัวแปรแบบ Volatile	170
การทดลองที่ 15 คุณสมบัติของตัวแปรแบบ Static	172
การทดลองที่ 16 คุณสมบัติของ Function แบบ void	174
การทดลองที่ 17 คุณสมบัติของ Function แบบ return ค่า	176
การทดลองที่ 18 การผ่านค่าให้ Function ด้วย ตัวแปร	178
การทดลองที่ 19 การผ่านค่าให้ Function ด้วย pointer	180
การทดลองที่ 20 การผ่านค่าให้ Function ด้วย Array	182
การทดลองที่ 21 การกระทำกับข้อมูลระดับบิต	184
การทดลองที่ 22 การใช้คำสั่งทางคณิตศาสตร์และตรีโกณมิติ	186
การทดลองที่ 23 การตรวจสอบ string ด้วย isalnum	189
การทดลองที่ 24 การตรวจสอบ string ด้วย isalpha	191
การทดลองที่ 25 การตรวจสอบ string ด้วย isspace	193
การทดลองที่ 26 การตรวจสอบ string ด้วย islower	195
การทดลองที่ 27 การตรวจสอบ string ด้วย isupper	197
การทดลองที่ 28 การแปลงค่า string ด้วย tolower	199
การทดลองที่ 29 การแปลงค่า string ด้วย toupper	201

บทที่4 : Arduino กับ Hardware LAB 203

การทดลองตอนที่1 : Arduino กับ Digital Input/Output 204

การทดลองที่ 1.1 โปรแกรมไฟกระพริบด้วย LED 1 ดวง แบบที่ 1	206
การทดลองที่ 1.2 โปรแกรมไฟกระพริบด้วย LED 1 ดวง แบบที่ 2	207
การทดลองที่ 1.3 โปรแกรมไฟกระพริบด้วย LED 1 ดวง แบบที่ 3	208
การทดลองที่ 1.4 โปรแกรมไฟกระพริบด้วย LED 1 ดวง แบบที่ 4	209
การทดลองที่ 1.5 โปรแกรมไฟกระพริบด้วย LED 2 ดวง	211
การทดลองที่ 1.6 โปรแกรมไฟวิ่งแบบใช้ LED หลายดวง	214
การทดลองที่ 1.7 โปรแกรม Switch ควบคุม LED ON/OFF	217
การทดลองที่ 1.8 โปรแกรม Switch ควบคุม LED Toggle	218

การทดลองตอนที่2 : Arduino กับ Analog Input/Output	220
การทดลองที่ 2.1 โปรแกรม อ่านค่า ADC ปรับความสว่าง LED	222
การทดลองที่ 2.2 โปรแกรม อ่านค่า ADC แสดงผลด้วย RS232	224
การทดลองที่ 2.3 โปรแกรม อ่านค่า ADC แสดงผลด้วย LED Bargraph จาก Arduino I/O	226
การทดลองที่ 2.4 โปรแกรม อ่านค่า ADC แสดงผลด้วย LED Bargraph จาก 74HC595	229
การทดลองตอนที่ 3 : Arduino กับ จอแสดงผล LCD	230
คำสั่ง LCD ของ ET_LCD_4BIT Library	245
การทดลองที่ 3.1 โปรแกรมเชื่อมต่อ LCD 4บิต แบบใช้สัญญาณ 6 เส้น	257
การทดลองที่ 3.2 โปรแกรมอ่านข้อมูลการแสดงผลจากจอแสดงผล LCD	260
การทดลองที่ 3.3 โปรแกรมแสดงผลตัวอักษรที่สร้างขึ้นเองให้ LCD	262
คำสั่ง LCD ของ ET_LCDHC595 Library	266
การทดลองที่ 3.4 โปรแกรมเชื่อมต่อ LCD 4บิต แบบใช้ 74HC595	276
การทดลองตอนที่4 : Arduino กับ Sound Speaker	277
การทดลองที่ 4.1 โปรแกรมสร้างเสียง Beep	279
การทดลองที่ 4.2 โปรแกรมสร้างเสียงความถี่ตัวโน้ตดนตรี	280
การทดลองที่ 4.3 โปรแกรมสร้างเสียงเพลงด้วย Melody	282
การทดลองที่5 : Arduino กับ Stepping Motor	285
การทดลองที่ 5.1 โปรแกรมควบคุม Stepping Motor แบบ 1 เฟส (Wave Drive)	295
การทดลองที่ 5.2 โปรแกรมควบคุม Stepping Motor แบบ 2 เฟส (Full Step)	297
การทดลองที่ 5.3 โปรแกรมควบคุม Stepping Motor แบบ Half Step	299
การทดลองตอนที่6 : Arduino กับ DC Motor	302
การทดลองที่ 6.1 โปรแกรมควบคุมความเร็ว DC Motor แบบทิศทางคงที่	309
การทดลองที่ 6.2 โปรแกรมควบคุมความเร็ว DC Motor แบบเลือกทิศทางได้	311

การทดลองตอนที่7 : Arduino กับ อุปกรณ์ 1 Wire Bus	313
คำสั่ง 1-Wire ของ Arduino(OneWire Library)	318
แนะนำ ไอซีรหัสกุญแจอิเล็กทรอนิกส์ แบบ 1-Wire เบอร์ DS1990A	331
การทดลองที่ 7.1 โปรแกรมอ่านรหัส Serial Number จาก DS1990A	335
แนะนำ ไอซีวัดอุณหภูมิแบบ 1-Wire เบอร์ DS1820/DS18S20	337
การทดลองที่ 7.2 โปรแกรมอ่านค่าอุณหภูมิจาก DS18S20 แบบ Single Device	348
การทดลองที่ 7.3 โปรแกรมอ่านค่าอุณหภูมิจาก DS18S20 แบบ Multi Drop Device	353

การทดลองตอนที่8 : Arduino กับ อุปกรณ์ I2C Bus	356
คำสั่ง I2C Bus ของ Arduino(Wire Library)	362
แนะนำ I2C Real Time Clock เบอร์ DS1307	373
การทดลองที่ 8.1 โปรแกรมอ่านค่า RTC จาก DS1307 แบบ RTC Polling	385
การทดลองที่ 8.2 โปรแกรมอ่านค่า RTC จาก DS1307 แบบ RTC Interrupt	390
แนะนำ I2C Real Time Clock เบอร์ PCF8583	395
การทดลองที่ 8.3 โปรแกรมอ่านค่า RTC เบอร์ PCF8583 แบบ RTC Polling	423
การทดลองที่ 8.4 โปรแกรมอ่านค่า RTC เบอร์ PCF8583 แบบ RTC Interrupt	428
การทดลองที่ 8.5 โปรแกรมอ่านค่า RTC เบอร์ PCF8583 แบบ Alarm Interrupt	432
แนะนำ I2C EEPROM เบอร์ 24LC32	436
การทดลองที่ 8.6 โปรแกรมอ่าน เขียน EEPROM เบอร์ 24LC32	447
แนะนำ I2C Input/Output เบอร์ PCF8574/A	451
การทดลองที่ 8.7 โปรแกรมทดสอบการใช้ PCF8584 เป็น Output	461
การทดลองที่ 8.8 โปรแกรมทดสอบการใช้ PCF8574 เป็น Input	463

การทดลองตอนที่9 : Arduino กับ อุปกรณ์ SPI	465
แนะนำ SPI Output เบอร์ 74HC595	466
การทดลองที่ 9.1 โปรแกรมทดสอบการใช้ Output ของ 74HC595	472
แนะนำ SPI DAC เบอร์ MCP4922	473
การทดลองที่ 9.2 โปรแกรมการสร้างแรงดัน Output ด้วย DAC(MCP4922)	479
การทดลองที่ 9.3 โปรแกรมการสร้างสัญญาณ Sine Wave ด้วย DAC(MCP4922)	481

การทดลองตอนที่10 : Arduino กับ RC Servo Motor	484
คำสั่ง Servo ของ ET_SERVO Library	495
การทดลองที่ 10.1 โปรแกรมควบคุม Servo ด้วย RS232	509
การทดลองที่ 10.2 โปรแกรมอ่านค่า ADC ปรับตำแหน่งการเคลื่อนที่ของ Servo	513
การทดลองที่ 10.3 โปรแกรมควบคุม Modify Servo ด้วย RS232	515
การทดลองตอนที่11 : Arduino กับ Internal Memory	519
การทดลองที่ 11.1 โปรแกรม อ่าน เขียน Internal EEPROM	520
การทดลองที่ 11.2 โปรแกรมตัวอย่างการอ่าน Flash Memory	523

บทที่1 : รู้จักกับ Arduino

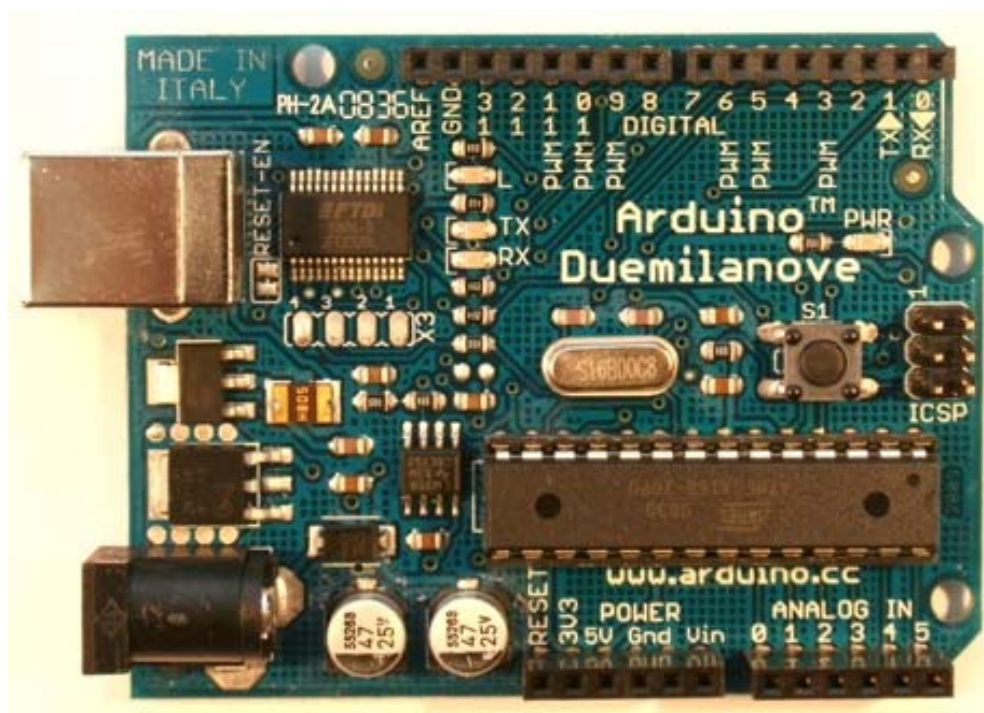
Arduino เป็นภาษา อิตาลี ซึ่งใช้เป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงการพัฒนา Open Source ของ AVR อีกโครงการหนึ่งที่ชื่อว่า **“Wiring”** แต่เนื่องจากโครงการของ “Wiring” เลือกใช้ AVR เบอร์ ATmega128 ซึ่งเป็นไมโครคอนโทรลเลอร์ที่มีจำนวนของหน่วยความจำ และ I/O ค่อนข้างมาก และที่สำคัญ ATmega128 เป็นชิปที่มีตัวถังแบบ SMD จึงทำให้เป็นอุปสรรคสำหรับผู้เริ่มต้นในการสร้างบอร์ดและต่อวงจรขึ้นมาใช้งานกันเอง และบอร์ดจะมีขนาดค่อนข้างใหญ่ ซึ่งอาจดูว่าเกินความจำเป็นสำหรับผู้เริ่มต้น จึงไม่ค่อยได้รับความนิยมเท่าที่ควร แต่หลังจากที่ทางทีมงาน Arduino นำ Source Code ของ “Wiring” มาพัฒนาปรับปรุงใหม่ โดยให้สามารถใช้งานกับไมโครคอนโทรลเลอร์ AVR ขนาดเล็ก อย่าง Mega8 และ Mega168 ได้ จึงทำให้ระบบวงจรของบอร์ดมีขนาดเล็กลงกว่า “Wiring” มากและยังใช้อุปกรณ์น้อยขึ้น ทำให้ง่ายต่อการต่อวงจรใช้งานกันเอง และยังประหยัดต้นทุนในการสร้างบอร์ดไปได้มาก ด้วยเหตุนี้เองที่ทำให้ “Arduino” ได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมาก ในระยะเวลาอันรวดเร็ว

เนื่องจาก Arduino เป็นภาษา อิตาลี ซึ่งมีสำเนียงการอ่านออกเสียงที่เป็นรูปแบบเฉพาะ และยังไม่มีการกำหนดเป็นคำภาษาไทยขึ้นมาอย่างเป็นทางการ ถึงแม้ว่า Arduino จะเป็นชื่อที่รู้จักของคนไทยมาระยะเวลาหนึ่งแล้วก็ตาม แต่ก็ยังไม่มีคำอ่านที่เป็นภาษาไทย อย่างเป็นทางการว่า คำๆนี้ ควรอ่านออกเสียงเป็นไทยว่าอย่างไร บางคนอ่านว่า อาร์-ดู-วี-โน บางคนอ่านว่า อา-เดีย-โน บางคนอ่าน เอ-อา-ดู-ไอ-โน และอื่นๆอีกมากมาย ซึ่งผู้เขียนเองก็ไม่มีความรู้เรื่องภาษา อิตาลี เลย จึงไม่ทราบเหมือนกันว่า จะเรียกชื่อของ Arduino เป็นภาษาไทยว่าอย่างไรจึงจะถูกต้อง ดังนั้นเพื่อไม่ให้เกิดความสับสน จึงขอใช้การทับศัพท์ตามชื่อเรียกที่เขียนเป็นภาษาอังกฤษเป็น Arduino ไปเลย

Arduino มีจุดเด่นในเรื่องของ ความง่ายในการเรียนรู้และใช้งาน เนื่องจากมีการออกแบบคำสั่งต่างๆขึ้นมาสนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งถึงแม้ว่า Arduino เองจะมีรูปแบบการใช้งาน คล้ายๆกับกันไมโครคอนโทรลเลอร์อย่าง Basic Stamp ของ Parallax, BX-24 ของ Netmedias และ Handy Board ของ MIT แต่ก็มีจุดเด่นกว่าของรายการอื่นๆหลายอย่าง เป็นต้นว่า

- ราคาไม่แพง เนื่องจากมี Source Code และวงจร แจกให้ฟรี สามารถต่อวงจรขึ้นมาใช้งานได้เอง
- โปรแกรมที่ใช้พัฒนาของ Arduino รองรับการทำงานทั้ง Windows, Linux และ Macintosh OSX
- มีรูปแบบคำสั่งที่ง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆที่มีความซับซ้อนมากๆได้ และยัง สามารถสร้างคำสั่งและ Library ใหม่ๆ ขึ้นมาใช้งานได้ เมื่อมีความชำนาญมากขึ้นแล้ว
- มีการเปิดเผยวงจรและ Source Code ทั้งหมดทำให้สามารถนำไปพัฒนาต่อยอดเพิ่มเติมได้ตามความต้องการทั้ง Hardware และ Software

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์ โดยใช้ AVR ขนาดเล็กเป็นตัวประมวลผลและสั่งงาน เหมาะสำหรับนำไปใช้ในการศึกษาเรียนรู้ระบบไมโครคอนโทรลเลอร์ และ นำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input / Output ต่างๆ ได้มากมาย ทั้งในแบบที่เป็นการทำงานตัวเดียวอิสระ หรือ เชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่นๆ เช่น คอมพิวเตอร์ PC ทั้งนี้ก็เนื่องมาจากว่า Arduino สนับสนุนการเชื่อมต่อกับอุปกรณ์ Input / Output ต่างๆ ได้มากมาย ทั้งแบบ Digital และ Analog เช่น การรับค่าจากสวิตช์ หรือ อุปกรณ์ตรวจจับ (Sensor) แบบต่างๆ รวมไปถึง การควบคุมอุปกรณ์ Output ต่างๆ ตั้งแต่ LED, หลอดไฟ, มอเตอร์, รีเลย์ ฯลฯ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เอง ในกรณีที่ผู้ใช้พอมีความรู้ด้านอิเล็กทรอนิกส์อยู่บ้าง หรือ สามารถซื้อแผงวงจรสำเร็จรูปที่มีการผลิตออกจำหน่ายกันในราคาที่ไม่แพง สำหรับเรื่องของโปรแกรมที่จะใช้เป็นเครื่องมือในการพัฒนานั้น สามารถ Download มาใช้งานกันได้ฟรีโดยไม่เสียค่าใช้จ่ายใดๆ โดย Arduino มีจุดเด่น ในเรื่องของความง่ายต่อการพัฒนาโปรแกรม และมีเอกสารข้อมูลรวมทั้งตัวอย่างต่างๆ ให้ใช้เป็นแนวทางในการศึกษาเรียนรู้เป็นจำนวนมาก เนื่องจาก Arduino เป็นระบบการพัฒนาไมโครคอนโทรลเลอร์แบบ Open Source ซึ่งมีการตีพิมพ์เอกสารต่างๆ ที่เกี่ยวข้องออกมาเผยแพร่ให้ได้รับรู้เป็นระยะๆ รวมทั้งการเปิดเผย Source Code และตัวอย่างต่างๆ ให้ผู้ใช้นำไปใช้งาน หรือ พัฒนาดัดแปลงต่อยอดได้โดยไม่เสียค่าใช้จ่าย ด้วยเหตุนี้จึงมีผู้คนทั่วไปให้ความสนใจและนำไปศึกษาทดลองใช้งานกันมากมาย มีการนำไปดัดแปลงและสร้างเป็นโครงงาน แบบต่างๆ กันเป็นจำนวนมาก จึงเป็นประโยชน์อย่างยิ่งสำหรับผู้เริ่มต้นที่สามารถนำเอาตัวอย่างและโครงงานต่างๆ ที่คนอื่นทำไว้แล้ว มาใช้อ้างอิงเป็นแนวทางในการศึกษาเรียนรู้ได้โดยง่ายและที่สำคัญคือ ฟรี ไม่เสียค่าใช้จ่าย



รูปบอร์ด Arduino ซึ่งเป็นผู้ออกแบบและพัฒนา Arduino ออกมาเผยแพร่

เปรียบเทียบ ภาษาซี กับ Arduino

โปรแกรมภาษาของ Arduino จะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์ แบบหนึ่ง ที่มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับ ภาษาซีมาตรฐาน (ANSI-C) อื่นๆ เพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C เล็กน้อย เพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้ง่ายและสะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรง

เมื่อกล่าวถึงภาษาซีนั้น คิดว่าหลายคนคงเคยได้ยินชื่อของภาษาซีกันมาบ้างแล้ว และบางคนอาจเคยเขียนโปรแกรมด้วยภาษาซีกันมาแล้ว บางคนอาจกำลังศึกษาและใช้งานกันอยู่ สำหรับภาษาซีที่มีการใช้งานกันอยู่ในแวดวงของ ไมโครคอนโทรลเลอร์ ปัจจุบันนั้นจะมีอยู่มากมายหลายยี่ห้อ ทั้งแบบที่แจกจ่ายให้ใช้งานกันฟรีๆ และแบบเสียเงินซื้อด้วยระดับราคาต่างๆกันไป อาจกล่าวได้ว่ามันมีมากจนนับกันไม่ถ้วนเลยทีเดียว มักมีคำถามประจำว่า ถ้าคิดจะเริ่มเขียนภาษาซี จะเลือกใช้งานภาษาซีตัวไหนดี ภาษาซีแต่ละตัวมันแตกต่างกันอย่างไร คำถามเหล่านี้ มักเกิดขึ้นเสมอๆกับผู้ที่กำลังคิดจะเริ่มต้น หรือแม้แต่คนที่เคยเรียนโปรแกรมภาษาซีมาบ้างแล้วแต่อยู่บนโลกของคอมพิวเตอร์ เมื่อคิดจะเปลี่ยนหรือย้ายมาอยู่ในโลกของไมโครคอนโทรลเลอร์บ้าง กลับเริ่มต้นไม่ถูก และไม่สามารถเขียนโปรแกรมใช้งานได้ทันที ทั้งๆที่การเขียนโปรแกรมก็ใช้ภาษา ที่เรียกว่า ภาษาซีเหมือนกัน ดังนั้น ในที่นี้ผู้เขียนจึงจะขอกล่าวนำให้รู้จัก ภาษาซี กันก่อนซักเล็กน้อย ก่อนที่จะเริ่มต้นเข้าสู่เรื่องราวที่เป็นภาษาซีของ Arduino กันโดยตรง

โดยประวัติความเป็นมาของภาษาซีนั้น ถูกบันทึกไว้ว่า ภาษาซี (C Programming Language) นั้นถูกพัฒนาขึ้นมาในปี ค.ศ. 1970 โดย Dennis Ritchie แห่ง Bell Laboratories ซึ่งในช่วงแรกๆ ภาษาซีถูกใช้งานเฉพาะแต่ในห้องปฏิบัติการของ Bell จนกระทั่งปี 1978 Brian Kernighan กับ Dennis Ritchie จึงได้ออกหนังสือ กำหนดมาตรฐานของภาษาซี ออกมาเผยแพร่ ชื่อกำหนดนี้ ถูกเรียกว่า K&R C ทำให้ภาษาซีเริ่มเป็นที่รู้จักของคนทั่วไปกันมากขึ้น จนกระทั่งปี 1980 ภาษาซี ก็ได้รับความนิยมมากขึ้นเป็นลำดับ จนได้มีการพัฒนาโปรแกรมที่ใช้ในการแปลภาษาซี (C-Compiler) ออกมาใช้งานกันอย่างแพร่หลาย ซึ่งสิ่งที่ทำให้ภาษาซีได้รับความนิยมในเวลาอันรวดเร็วก็เนื่องจากว่า ภาษาซีมีความได้เปรียบและมีความอ่อนตัวในการใช้งานที่เหนือกว่าภาษาอื่นๆ คือ ภาษาซี สามารถนำไปใช้งานบนระบบฮาร์ดแวร์ที่มีความแตกต่างกันได้หลากหลาย โดยผู้ใช้เพียงแต่เลือกใช้ตัวแปลคำสั่ง (C-Compiler) ให้ตรงกับระบบฮาร์ดแวร์ที่ใช้งานอยู่ ส่วนเรื่องรูปแบบในการเขียนโปรแกรมจะเป็นมาตรฐานอันเดียวกัน จนในบางครั้ง อาจสามารถย้ายโปรแกรมจากระบบฮาร์ดแวร์หนึ่งไปใช้งานกับอีกระบบหนึ่งได้ ทำให้ไม่ต้องเสียเวลาในการศึกษาโปรแกรมใหม่ๆให้เสียเวลา เนื่องจากว่าภาษาซี เป็นภาษาที่มีรูปแบบการใช้งานที่ง่าย คือ มีแต่ข้อกำหนดในการใช้งาน หรือ Syntax แต่ไม่มีฟังก์ชันสำเร็จรูป (Built-in Function) ใดๆรวมอยู่ในตัวของภาษาด้วย โดยส่วนที่เป็นฟังก์ชันการใช้งานต่างๆ เช่น การดำเนินการเกี่ยวกับ Input / Output การจองหน่วยความจำ (Memory Allocation)

เป็นหน้าที่ของผู้ใช้ที่จะต้องสร้างขึ้นใช้งานเอง หรือ ในบางครั้งก็อาจใช้วิธีการเรียกใช้ฟังก์ชันที่ผู้ผลิตตัวแปลคำสั่ง (C-Compiler) สร้างเตรียมไว้ให้ใช้งานในรูปแบบของ Library Function ซึ่งคำสั่งหรือฟังก์ชันส่วนที่เป็น Library Function นี้เอง ที่เป็นสาเหตุทำให้ภาษาซี มีความแตกต่างกัน เพียงแต่ว่าในบางครั้งผู้ใช้จะแบ่งแยกไม่ออกว่า อันไหนเป็นคำสั่งส่วนมาตรฐานของภาษาซี อันไหนเป็นส่วนที่สร้างขึ้นมาใหม่

ด้วยเหตุที่ภาษาซี เป็นที่นิยมใช้งานกันอย่างแพร่หลายมากขึ้น จึงทำให้มีผู้ผลิต Compiler ของภาษาซี เกิดขึ้นมามากมาย ทั้งแบบที่แจกจ่ายให้ใช้งานกันฟรีๆ และแบบที่จำหน่ายเพื่อการค้า สาเหตุของการแข่งขันกันนี้เองที่ทำให้เริ่มมีการเพิ่มเติมความสามารถและลูกเล่นต่างๆ ให้กับภาษาซีมากขึ้น เพื่อหวังดึงดูดใจผู้ซื้อและเป็นจุดขาย ทำให้ภาษาซีเริ่มมีความแตกต่างกันในแต่ละผู้ผลิต จนทำให้เหล่าผู้ใช้เกิดความสับสนกันเป็นอย่างมาก ดังนั้นทาง American National Standard Institute (ANSI) จึงได้ทำการตั้งข้อกำหนดมาตรฐานของภาษาซีขึ้น โดยเรียกกันว่า “ANSI-C” เพื่อใช้เป็นข้อบังคับและคงมาตรฐานของภาษาซีไว้ไม่ให้เปลี่ยนแปลงไปจากเดิม โดยทุกผู้ผลิตจะต้องสร้างตัวแปลภาษาให้มีคุณสมบัติการใช้งานขั้นพื้นฐานที่เป็นสิ่งเดียวกัน ตามข้อกำหนดของ ANSI สำหรับส่วนของคำสั่งที่จะมีการสร้างเพิ่มเติมขึ้นมาใช้งานใหม่นั้น ให้เป็นสิทธิของผู้ผลิตแต่ละรายที่จะสร้างขึ้นใหม่ แต่จะไม่ถือเอาส่วนคำสั่งที่มีการสร้างขึ้นมาเพิ่มเติมนั้นว่าเป็นส่วนของ ANSI-C ด้วย ดังนั้นเมื่อจะใช้งานภาษาซีตัวใด ก็ขอให้ผู้อ่านศึกษารายละเอียดทางด้านคุณสมบัติของตัวภาษานั้นๆ ด้วย ว่ามีคุณสมบัติการใช้งานที่รองรับคำสั่งของ ANSI-C หรือไม่ มีข้อยกเว้นอย่างไรบ้าง และมีการสร้างคำสั่งหรือเพิ่มเติมความสามารถมากกว่า ANSI-C อย่างไรบ้าง ซึ่งถ้าโปรแกรมที่เขียนขึ้นนั้น เลือกใช้เฉพาะคำสั่งที่เป็น ANSI-C ทั้งหมด จะทำให้สามารถย้าย Code โปรแกรมจากระบบฮาร์ดแวร์แบบหนึ่ง เพื่อไปใช้งานกับอีกระบบฮาร์ดแวร์แบบหนึ่งได้ทันที แต่ถ้าใน Code โปรแกรมนั้นมีการใช้คำสั่งที่นอกเหนือไปจากคำสั่งของ ANSI-C แล้ว ก็จะไม่สามารถใช้ได้ทันที ต้องมีการปรับแก้ Code กันใหม่เป็นบางส่วน หรือต้องสร้างคำสั่งบางส่วนขึ้นมาทดแทน จึงจะสามารถใช้งานได้ ซึ่งส่วนของคำสั่งที่เป็นภาษาซีตามมาตรฐานของ ANSI-C จะมี 32 คำสั่ง คือ

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				

คำสั่งทั้ง 32 คำสั่งนี้ ไม่ว่าจะอยู่บน C-Compiler ตัวใดก็จะต้องมีข้อกำหนดและรูปแบบการใช้งานของคำสั่งที่เหมือนกันทั้งหมด ยกเว้นว่า C-Compiler ตัวนั้น ไม่รองรับคำสั่งของ ANSI-C ทั้งหมด ซึ่งส่วนนี้ต้องดูจากคุณสมบัติของภาษาซีที่จะใช้ว่าเป็นอย่างไร

มาถึงตรงนี้ก็คงพอจะทำให้ผู้อ่านได้รู้จักภาษาซีกันดีขึ้นบ้างพอสมควรแล้ว ดังนั้น ในการศึกษาภาษาซีนั้น ขอให้ศึกษาโครงสร้าง ข้อกำหนด และรูปแบบการใช้งาน ของคำสั่งที่เป็น ANSI-C ทั้ง 32 คำสั่งนี้ไว้ก่อนเป็นอันดับแรก ซึ่งจะเป็นพื้นฐานและเป็นประโยชน์อย่างมาก และยังสามารถที่จะนำความรู้ที่ได้ไปใช้งานได้กับ C-Compiler ทุกยี่ห้อ ที่รองรับมาตรฐานของ ANSI-C ได้ทั้งหมด แล้วค่อยไปศึกษาเพิ่มเติมในเรื่องของลูกเล่น คำสั่งเพิ่มเติม และ ความความสามารถอื่นๆในส่วนที่แตกต่างกันของ C-Compiler

สำหรับการเขียนโปรแกรมของ Arduino นั้นจะใช้ภาษา C++ ซึ่งเป็นรูปแบบภาษาซีประยุกต์แบบหนึ่ง ที่มีโครงสร้างการทำงานของตัวเอง คล้ายกับ ภาษาซีมาตรฐาน (ANSI-C) ทั่วไป เพียงแต่ได้มีการปรับปรุงเพื่อลดความยุ่งยากในการใช้งานลง เพื่อให้ผู้ใช้สามารถใช้งานและเขียนโปรแกรมได้ง่าย และสะดวกมากกว่าเขียนภาษาซีแบบมาตรฐาน โดยตรง ซึ่งในความเป็นจริงแล้วในการเขียนโปรแกรมของ Arduino เราสามารถใช้คำสั่งต่างๆที่เป็นคำสั่งตามมาตรฐานของ ANSI-C เข้ามาใช้ในการเขียนโปรแกรมได้ทันที โดยรูปแบบการเขียนโปรแกรม และการใช้งานคำสั่งต่างๆนั้น สามารถอ้างอิงจากหนังสือ ตำรา ของ ภาษาซี มาตรฐาน ANSI-C ได้โดยตรง

ซึ่งหนังสือเล่มนี้จะเขียนขึ้นโดยอ้างอิงข้อมูลบางส่วนมาจากข้อกำหนดต่างๆของภาษาซีตามข้อกำหนดของ ANSI-C ผสมรวมเข้ากับข้อกำหนดเฉพาะและรายละเอียดเพิ่มเติมในส่วนที่เป็น C++ ของ Arduino เนื่องจาก Arduino ได้ทำการปรับปรุงและดัดแปลงรูปแบบการเขียนโปรแกรมที่ผุดผื่นไปจากภาษาซีมาตรฐาน ซึ่งในบางส่วนก็เป็นส่วนที่ถูกออกแบบและสร้างขึ้นใหม่ โดยเนื้อหาบางส่วนก็ได้ ตัดต่อเรียบเรียงและแปลมาจากเค้าโครงของคู่มืออ้างอิงของ Arduino ที่เป็นภาษาอังกฤษ โดยได้มีการสอดแทรกเนื้อหาอื่นๆและตัวอย่างโปรแกรมต่างๆเพิ่มเติมเข้าไปตามความเหมาะสม

ซึ่งจากการที่ผู้เขียนได้ทำการศึกษาค้นคว้าทดลองและใช้งานภาษาซี ของ Arduino มาในระยะเวลาหนึ่ง พบว่าในความเป็นจริงแล้ว Arduino นั้นไม่ใช่ C-Compiler โดยตรง แต่ Arduino จะมีลักษณะการทำงานเช่นเดียวกันกับ Text Editor ของภาษา C++ ตัวหนึ่ง โดยจะทำงานร่วมกับ Utility บางส่วนที่ Arduino สร้างขึ้นมารองรับ โดย Arduino จะใช้รูปแบบการทำงานของ Text Editor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆนั้น Arduino จะไปเรียกใช้ตัวแปลภาษาซี และ Utility อื่นๆที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของ ไมโครคอนโทรลเลอร์ ตระกูล AVR อีกทีหนึ่ง โดย Arduino จะเลือกใช้ C-Compiler ของ “GNU AVR-GCC Toolchain” ร่วมกับ Library Function ของ “avr-libc” ส่วน Utility ที่ใช้ในการ Upload Code ให้กับ AVR นั้นก็จะใช้ของ “AVRDude” ดังนั้นผู้ที่เขียนภาษาซีของ AVR เป็นอยู่แล้ว และต้องการประยุกต์ใช้งาน Arduino ให้ได้ประสิทธิภาพการทำงานมากยิ่งขึ้นไปอีก ก็สามารถศึกษาข้อกำหนด และ หน้าที่ในการใช้งาน Library และคำสั่งอื่นๆที่บรรจุไว้ใน Library ต่างๆทั้งจากของ “GNU AVR-GCC Toolchain” และ “avr-libc” เพิ่มเติมอีก เพื่อใช้เป็นแนวทางในการปรับปรุงและประยุกต์ใช้งาน Arduino ในรูปแบบที่สลับซับซ้อนมากขึ้นไปได้อีก

โครงสร้างการเขียนโปรแกรม ภาษาซี ของ Arduino

ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมออกเป็นส่วนย่อยๆ หลายๆ ส่วน โดยเรียกแต่ละส่วนว่า ฟังก์ชัน และ เมื่อนำฟังก์ชัน มารวมเข้าด้วยกัน ก็จะเรียกว่าโปรแกรม โดยโครงสร้างการเขียนโปรแกรมของ Arduino นั้น ทุกๆโปรแกรมจะต้องประกอบไปด้วยฟังก์ชันจำนวนเท่าใดก็ได้ แต่อย่างน้อยที่สุดต้องมีฟังก์ชัน จำนวน 2 ฟังก์ชัน คือ setup() และ loop() ดังตัวอย่าง

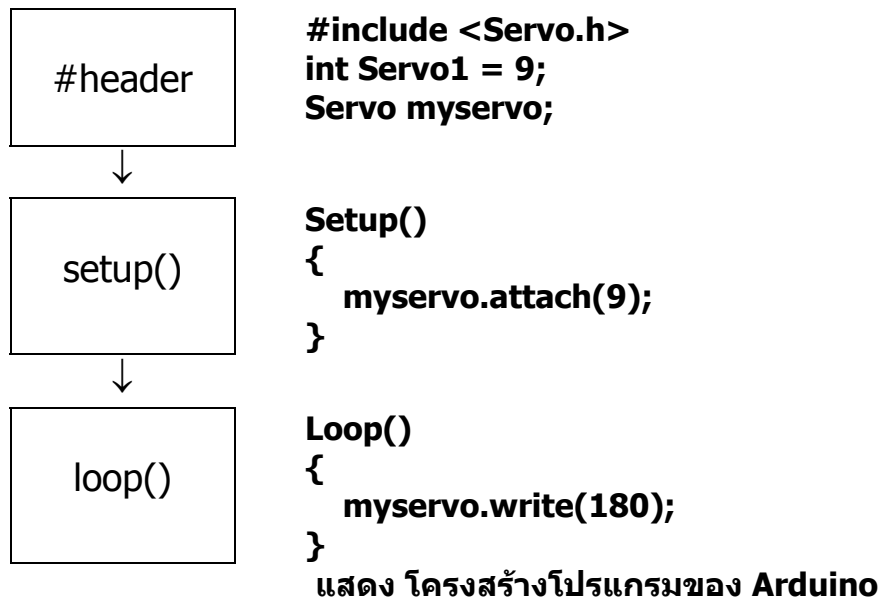
```
#include <Servo.h>          // สั่งผนวกไฟล์ ชื่อ Servo.h เข้ามาใช้ในโปรแกรม
int Servo1 = 9;             // กำหนดให้ Servo1 แทน Pin Digital-9
Servo myservo;              // สร้าง object ชื่อ myservo เพื่อควบคุม Servo

void setup()
{
  myservo.attach(Servo1);    // กำหนดให้ใช้ขา Digital-9 สร้างสัญญาณควบคุม Servo
}

void loop()
{
  myservo.write(180);        // กำหนดค่าตำแหน่งให้กับ Servo = 180 องศา
}
```

จะเห็นได้ว่าโครงสร้างพื้นฐานของภาษาซีที่ใช้กับ Arduino นั้น จะประกอบไปด้วย 3 ส่วนใหญ่ๆด้วยกัน คือ

- **header** ในส่วนนี้จะมีหรือไม่มีก็ได้ ถ้ามีต้องกำหนดไว้ในส่วนเริ่มต้นของโปรแกรม ซึ่งส่วนของ Header ได้แก่ ส่วนที่เป็น Compiler Directive ต่างๆ รวมไปถึงส่วนของการประกาศตัวแปร และค่าคงที่ต่างๆที่จะใช้ในโปรแกรม
- **setup()** ในส่วนนี้เป็นฟังก์ชันบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรม ถึงแม้ว่าในบางโปรแกรมจะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้ด้วยเสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้ในระหว่างวงเล็บปีกกา { } ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชัน โดยฟังก์ชันนี้จะใช้สำหรับบรรจุคำสั่งในส่วนที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียวตอนเริ่มต้นทำงานของโปรแกรมครั้งแรกเท่านั้น ซึ่งได้แก่คำสั่งเกี่ยวกับการ Setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่การใช้งานของ PinMode และการกำหนดค่า Baudrate สำหรับใช้งานพอร์ตสื่อสารอนุกรม เป็นต้น
- **loop()** เป็นส่วนฟังก์ชันบังคับที่ต้องกำหนดให้มีในทุกๆโปรแกรมเช่นเดียวกันกับฟังก์ชัน setup() โดยฟังก์ชัน loop() นี้จะบรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเป็นวงรอบซ้ำๆกันไปไม่รู้จบ ซึ่งถ้าเปรียบเทียบกับรูปแบบของ ANSI-C ส่วนนี้ก็คือ ฟังก์ชัน main() นั่นเอง



จะเห็นได้ส่วนแรกซึ่งถือเป็นส่วนเริ่มต้นของโปรแกรม ซึ่งจะเรียกว่า Header โดยประกอบด้วยคำสั่ง `#include` ซึ่งเป็นคำสั่งพิเศษที่เรียกว่า Compiler Directive ซึ่งมัน ไม่ใช่คำสั่ง สำหรับสั่งงานในโปรแกรม ดังนั้นคำสั่งนี้จึงไม่ต้องมีเครื่องหมายเซมิโคลอนปิดท้ายคำสั่งเหมือนคำสั่งอื่นๆ โดย Compiler Directive จะใช้ทำหน้าที่สำหรับบอกให้ Compiler รับรู้เงื่อนไขในการแปลคำสั่งเท่านั้น ซึ่งในกรณีคำสั่ง `#include` จะใช้สำหรับบอกให้ Compiler รับรู้ว่าการแปลคำสั่งของโปรแกรมนี้นี้ มีไฟล์ภายนอกใดบ้างที่จำเป็นต้องใช้ ร่วมในการแปลคำสั่งให้กับโปรแกรมนี้อย่างไร โดย จากตัวอย่าง ข้างต้นจะเป็นการบอกให้ Compiler ทำการผนวก ไฟล์ ชื่อ “Servo.h” เข้ามา ใช้เพื่อเรียกใช้คำสั่งต่างๆที่บรรจุไว้ เข้ามาใช้งานในโปรแกรม โดยใช้รูปแบบ

```
#include <header.h>
```

โดยเมื่อพบคำสั่ง `#include` ตัวแปลภาษาของ Arduino จะไปค้นหาไฟล์ที่ระบุไว้ในเครื่องหมาย <> หลังคำสั่ง `#include` จากตำแหน่ง Directory ที่เก็บรวบรวม Library ของโปรแกรม Arduino ไว้ ซึ่งก็คือ “..\\arduino-0012\\hardware\\libraries\\” เช่น เมื่อทำการติดตั้งโปรแกรมของ Arduino ไว้ที่ Directory ที่ชื่อ ว่า “c:\\arduino-0012” ไฟล์ภายนอกที่เป็น Library Function และ Header ต่างๆ จะถูกรวบรวมเก็บไว้ที่ “c:\\arduino-0012\\hardware\\libraries\\” เมื่อโปรแกรมพบคำสั่ง `#include` โปรแกรมของ Arduino จะไป ค้นหาไฟล์ต่างๆจากตำแหน่งของ Directory ที่ชื่อ “c:\\arduino-0012\\hardware\\libraries\\” นั่นเอง

โดยส่วนของ Header จะนับรวมไปถึง คำสั่งส่วนที่ใช้ประกาศสร้าง ตัวแปร (Variable Declaration) และค่าคงที่ (Constant Declaration) รวมทั้ง ฟังก์ชันต่างๆ (Function Declaration) ด้วย ซึ่งจากตัวอย่าง ได้แก่ส่วนที่เป็นคำสั่ง

```
int Servo1 = 9;
Servo myservo;
```

สำหรับส่วนที่มีความสำคัญและจำเป็นที่สุดของโปรแกรม Arduino ที่จำเป็นต้องมีและจะขาดไม่ได้ในการเขียนโปรแกรมของ Arduino คือ ฟังก์ชัน `setup()` และ ฟังก์ชัน `loop()` ซึ่งฟังก์ชัน ทั้ง 2 ส่วนนี้มีรูปแบบโครงสร้างที่เหมือนกัน แต่ถูกกำหนดด้วยชื่อของฟังก์ชันเป็นการเฉพาะ คือ `setup()` และ `loop()` โดย `setup()` จะเขียนไว้ก่อน `loop()` ซึ่งทั้ง 2 ฟังก์ชันนี้ มีขอบเขต เริ่มต้นและสิ้นสุด อยู่ภายใต้เครื่องหมาย `{ }`

void setup()

{

· คำสั่งต่างๆ ที่ต้องการเขียนไว้ภายใต้ฟังก์ชัน **setup()**

·
}

หน้าที่ของฟังก์ชัน `setup()` ใน Arduino คือ ใช้ทำหน้าที่เป็นส่วนของโปรแกรมน้อยๆ สำหรับใช้บรรจุคำสั่งต่างๆ ที่ใช้สำหรับกำหนดการทำงานของระบบ หรือ กำหนดคุณสมบัติการทำงานให้กับอุปกรณ์ต่างๆ ซึ่งคำสั่งทั้งหมดที่บรรจุไว้ภายใต้ฟังก์ชันของ `setup()` นี้ จะถูกเรียกขึ้นมาทำงานเพียงรอบเดียวคือตอนเริ่มต้นการทำงานของโปรแกรม (หลังการรีเซ็ตให้ MCU เริ่มต้นทำงาน) เท่านั้น โดยคำสั่งที่นิยมบรรจุไว้ในฟังก์ชันส่วนนี้ ได้แก่ คำสั่งสำหรับกำหนดโหมดการทำงานของ Digital Pin หรือ คำสั่งสำหรับ กำหนดคุณสมบัติของพอร์ตสื่อสารอนุกรม เป็นต้น

void loop()

{

· คำสั่งต่างๆ ที่ต้องการให้ทำงานภายใต้ฟังก์ชัน **loop()**.

·
}

หน้าที่ของฟังก์ชัน `loop()` ใน Arduino คือ ใช้ทำหน้าที่เป็นส่วนหลักของโปรแกรม สำหรับใช้บรรจุคำสั่งควบคุมการทำงานต่างๆ ของโปรแกรม ที่ต้องการให้โปรแกรมทำงาน โดยคำสั่งที่บรรจุไว้ในฟังก์ชันนี้ จะถูกเรียกขึ้นมาทำงานซ้ำๆ กันตามลำดับและเงื่อนไขที่กำหนดไว้

ข้อสังเกต จะเห็นได้ว่าโปรแกรมนี้อาจจะประกอบไปด้วย คำสั่งที่เขียนขึ้นเองส่วนหนึ่ง และ อีกส่วนหนึ่งเป็น คำสั่งจากภายนอก ที่มีการสร้างและเก็บรวบรวมเป็นไฟล์ ในรูปแบบของ Library Function เก็บอยู่ภายนอกโปรแกรม เมื่อต้องการใช้งานก็สั่งผนวกไฟล์นั้นเข้ามาใช้งานในโปรแกรม จากนั้นก็สามารถเรียกใช้งานคำสั่งต่างๆ ที่สร้างเก็บไว้ในไฟล์นั้นได้ตามต้องการ

ตัวแปรใน Arduino

ตัวแปร หมายถึง กลุ่มของ ตัวอักษร ตัวเลข และ เครื่องหมายใดๆ ที่รวมกันเป็นชื่อ เพื่อใช้กำหนดเป็นตัวแทนของค่าข้อมูลที่เราต้องการจะอ้างถึงในโปรแกรม ทั้งนี้ก็เนื่องจกว่าในการทำงานของโปรแกรมจริงๆ นั้นจะใช้ค่าตัวเลขที่ผู้กำหนดให้ มาทำการประมวลผล ซึ่งในการเขียนโปรแกรมถ้าเราต้องเขียนโปรแกรมโดยกำหนดเป็นค่าตัวเลขให้กับโปรแกรมจริงๆ เลย ก็จะทำให้โปรแกรมที่เราเขียนขึ้นเต็มไปด้วยค่าตัวเลขต่างๆ มากมาย ซึ่งยากต่อการอ่าน ยากต่อการทำความเข้าใจ และยากต่อการตรวจสอบความถูกต้อง และอาจทำให้เกิดความผิดพลาดได้ง่ายด้วย ดังนั้นทุกภาษา จึงยอมให้มีการกำหนดชื่อ ขึ้นมาใช้แทนค่าตัวเลข เพื่อให้เขียนโปรแกรมได้สะดวกและง่ายต่อการอ่าน ทำความเข้าใจ ได้มากยิ่งขึ้น

ซึ่งลักษณะของข้อมูล อาจมีทั้งแบบที่เป็นค่าซึ่งสามารถเปลี่ยนแปลงได้ (variable) หรือ อาจเป็นแบบที่มีค่าคงที่ไม่สามารถเปลี่ยนแปลงได้ (constant) ในการประกาศใช้งานตัวแปร จำเป็นต้องประกาศชนิดของตัวแปร หรือบางครั้งอาจมีการกำหนดค่าเริ่มต้นให้กับตัวแปรด้วยก็ได้

การตั้งชื่อตัวแปรของ Arduino จะยึดหลักของ ANSI-C ทุกประการนั่นก็คือ

- ชื่อของตัวแปรต้องประกอบไปด้วยตัวอักษร ตัวเลข และ ยอมให้ใช้เครื่องหมายพิเศษอีก 2 ตัว คือ เครื่องหมาย Under Line (_) และ Dollar Sign (\$) มาใช้ในการตั้งชื่อได้ โดยชื่อต้องเรียงติดทั้งหมดห้ามมีการเว้นวรรค และ มีความยาวไม่เกิน 32 ตัวอักษร
- ชื่อของตัวแปรที่กำหนด จะใช้เฉพาะตัวอักษรทั้งหมด หรือ ตัวอักษรผสมกับตัวเลข หรือ อาจผสมกับเครื่องหมาย _ และ \$ ด้วยก็ได้ แต่ต้องเริ่มต้นชื่อด้วยตัวอักษรเป็นลำดับแรกเสมอ ห้ามตั้งชื่อโดยเริ่มต้นด้วยตัวเลข หรือ เครื่องหมาย
- ชื่อของตัวแปร ต้องกำหนดด้วยตัวอักษรที่เป็นตัวอักษรพิมพ์เล็กเท่านั้น และในภาษาซีจะถือว่าตัวอักษรที่เป็นตัวอักษรพิมพ์ใหญ่ และ ตัวอักษรพิมพ์เล็ก มีความหมายต่างกัน เช่น “LED” และ “led” และ “Led” ภาษาซี จะถือว่าเป็นคนละชื่อกัน
- ในการตั้งชื่อตัวแปร ห้ามนำคำสงวน (Reserve Word) มาใช้ตั้งชื่อ ซึ่งคำสงวน ได้แก่ ชื่อคำสั่ง และ ชื่อ Internal Function ต่างๆ ที่สร้างไว้แล้วในตัว ภาษา

*****ข้อแนะนำ***** ในการตั้งชื่อตัวแปรที่จะใช้ในภาษาซี ควรหลีกเลี่ยงและไม่ควรตั้งชื่อตัวแปร ด้วยชื่อที่เป็นตัวอักษร ที่ใช้ ตัวพิมพ์เล็ก ทั้งหมด เพราะมีโอกาสไปซ้ำกับคำสั่งของภาษาซีได้ง่าย

คำสั่งงานต่างๆของ Arduino

ในโปรแกรมภาษาซีของ Arduino นั้นจะมี ประโยค หรือ คำสั่ง ที่กำหนดขึ้นมาใช้งานไว้แล้วจำนวนหนึ่ง เมื่อผู้ใช้ต้องการจะประกาศ สร้างตัวแปร หรือ สร้างคำสั่ง หรือ ฟังก์ชันต่างๆ ขึ้นมาใช้งานเอง ก็ต้องไม่ตั้งชื่อให้ซ้ำกับชื่อของคำสั่งที่มีอยู่แล้วของ Arduino อีก ซึ่งคำสั่งที่ Arduino กำหนดขึ้นเพื่อใช้งานเป็นการเฉพาะไว้แล้วนั้นจะถือเป็น คำสงวน ที่ห้ามไม่ให้ผู้ใช้ทำการประกาศขึ้นมาใช้งานใหม่อีก เพราะจะไปซ้ำกับคำสั่ง เดิมที่มีอยู่แล้วนั่นเอง ซึ่งคำสั่งสงวนของ Arduino ได้แก่

คำสั่งงานที่เป็น Constant ของ Arduino

HIGH	LOW	INPUT	OUTPUT	SERIAL	DISPLAY	PI	HALF_PI
TWO_PI	LSBFIRST	MSBFIRST	CHANGE	FALLING	RISING	false	true
null							

คำสั่งงานที่เป็น Port Variables & Constants ของ Arduino

DDRB	PINB	PORTB	PB0	PB1	PB2	PB3	PB4
PB5	PB6	PB7	DDRC	PINC	PORTC	PC0	PC1
PC2	PC3	PC4	PC5	PC6	PC7	DDRD	PIND
PORTD	PD0	PD1	PD2	PD3	PD4	PD5	PD6
PD7							

คำสั่งงานที่เป็น Datatypes ของ Arduino

boolean	byte	char	class	default	do	double	int
long	private	protected	public	return	short	signed	static
switch	throw	try	unsigned	void			

คำสั่งงานที่เป็นคำสั่ง ของ Arduino

abs	acos	+=	+	[]	asin	=	atan
atan2	&		boolean	byte	case	ceil	char
class	,	//	?:	constrain	cos	{}	--
default	delay	delayMicroseconds	/	/**	.	else	
==	exp	false	float	floor	for	<	<=
if	++	!=	int	<<	<	<=	log
&&	!		loop	max	millis	min	-
%	/*	*	new	null	()	PI	return
>>	;	Serial	Setup	sin	sq	sqrt	-=
switch	tan	this	true	void	while	begin	read
write	print	println	available	digitalWrite		digitalRead	
analogRead		analogWrite		attachInterrupts		detachInterrupts	
beginSerial		serialWrite		serialRead		serialAvailable	
printString		printInteger		printByte		printHex	
printOctal		printBinary		println		pulseIn	shiftOut
pinMode							

ชนิดและประเภทของตัวแปร

ถ้าหากว่าเราจะเปรียบเทียบว่า ตัวแปร คือ ภาษาสำหรับบรรจุสิ่งของ และ ข้อมูล คือ สิ่งของที่เราต้องการจะเก็บ จะเห็นได้ว่า สิ่งของต่างๆรอบๆตัวเรานั้น จะมีคุณสมบัติที่แตกต่างกันไป ดังนั้นในการเลือกภาษาสำหรับใช้บรรจุสิ่งของ เราก็จำเป็นต้องเลือกชนิดของภาษาให้มีความเหมาะสมที่จะใช้เก็บสิ่งของด้วย ซึ่งสิ่งแรกที่ต้องพิจารณาคือ เราจะต้องรู้จักคุณสมบัติของสิ่งของที่ต้องการจะจัดเก็บ และ จุดประสงค์การใช้งาน ก่อน จากนั้นจึงจัดหาภาษาที่มี ขนาด และ รูปทรงของภาษา เหมาะสมที่จะใช้เก็บสิ่งของ เพื่อให้สามารถจัดเก็บ และ นำสิ่งของออกมาใช้งาน ได้อย่างง่ายประหยัดมากที่สุด

ในภาษาซีนั้น มีการกำหนด และ จำแนก ชนิดของตัวแปร ไว้เป็น 5 ชนิดด้วยกัน โดยแต่ละชนิดจะมีคุณสมบัติการใช้งานที่ต่างกัน เพื่อใช้ในการเก็บข้อมูลที่มีรูปแบบแตกต่างกัน คือ

- **char** ใช้เก็บข้อมูลที่เป็นตัวอักษร (character) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 256 ค่า
- **int** ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็ม (integer) ใช้เก็บข้อมูลที่เป็นเลขจำนวนเต็มได้ 65536 ค่า
- **float** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Single Precision
- **double** ใช้เก็บข้อมูลที่เป็นเลขทศนิยมแบบ Double Precision ซึ่ง สามารถเก็บค่าตัวเลขทศนิยมที่มีความละเอียดและถูกต้องของทศนิยมมากกว่าแบบ float ถึง 2 เท่า
- **void** ใช้เก็บตัวแปรที่ไม่มีค่า

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
char	8	-128 ถึง +127
int	16	-32768 ถึง +32767
float	32	3.4E-38 ถึง 3.4E+38
double	64	1.7E-308 ถึง 1.7E+308
void	0	ไม่มีค่า

ตารางแสดงคุณสมบัติของตัวแปรของภาษาซี

โดยในการประกาศสร้างตัวแปรขึ้นมาใช้งานของภาษาซีนั้นจะใช้รูปแบบดังนี้

type name

- **type** หมายถึง ชนิดของตัวแปร
- **name** หมายถึง ชื่อของตัวแปร

คุณสมบัติเฉพาะของตัวแปร

สำหรับตัวแปรชนิดที่ใช้เก็บค่าเลขจำนวนเต็ม (**char** และ **int**) นั้น ในภาษาซี ไม่ได้มีการจำแนก ชนิดของตัวแปรเพื่อใช้เก็บค่าตัวเลขที่เป็น ค่าบวก หรือ ค่าลบ เป็นการเฉพาะ แต่ภาษาซี จะใช้วิธีการเพิ่ม คำสั่งสำหรับกำหนดคุณสมบัติเฉพาะให้กับตัวแปรไว้อีก 4 คำสั่ง สำหรับใช้กำหนดคุณสมบัติของตัวแปรแบบนี้ให้มีคุณสมบัติที่เฉพาะเจาะจงลงไปอีกเป็นต้นว่า จะใช้ตัวแปรในการเก็บค่าตัวเลขที่เป็นค่าบวกอย่างเดียว หรือต้องการเก็บค่าตัวเลขแบบคิดเครื่องหมายด้วย เพื่อให้ผู้ใช้สามารถปรับแก้คุณสมบัติในการใช้งานของตัวแปรให้มีคุณสมบัติใกล้เคียงกับความต้องการใช้งานมากขึ้นไปอีก และเพื่อจำกัดขอบเขตการใช้งานของตัวแปรให้ตรงกับจุดประสงค์มากยิ่งขึ้น และยังเป็นการช่วยให้ประหยัดจำนวนของหน่วยความจำที่ใช้สร้างตัวแปรด้วยและทำให้โปรแกรมทำงานได้เร็วขึ้นกว่าเดิมอีกด้วย โดยในภาษาซีจะมีคำสั่ง ที่ใช้สำหรับระบุคุณสมบัติเฉพาะของตัวแปรที่ใช้เก็บค่าเลขจำนวนเต็ม 4 คำสั่ง คือ

- **unsigned** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรเฉพาะค่าที่เป็นบวกเท่านั้น
- **signed** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรทั้งค่า บวก และ ลบ
- **short** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่าน้อยกว่า int
- **long** ใช้ระบุให้เก็บค่าเลขจำนวนเต็มในตัวแปรที่มีค่ามากกว่า int เป็น 2 เท่า

โดยคำสั่งทั้ง 4 นี้จะใช้นำหน้าชนิดของตัวแปร เพื่อบอกให้ Compiler รับรู้ว่าตัวแปรที่เราประกาศสร้างขึ้นมานั้น มีคุณสมบัติเป็นอย่างไรเช่น

char ch1;	//กำหนดให้ ch1 เก็บค่า -128 ถึง +127
sign char ch2;	//กำหนดให้ ch2 เก็บค่า -128 ถึง +127
unsigned char ch3;	//กำหนดให้ ch3 เก็บค่า 0 ถึง 255
int num1;	//กำหนดให้ num1 เก็บค่าได้ -32768 ถึง +32767
signed int num2;	//กำหนดให้ num2 เก็บค่าได้ -32768 ถึง +32767
unsigned int num3;	//กำหนดให้ num3 เก็บค่าได้ 0 ถึง 65535
short int num4;	//กำหนดให้ num4 เก็บค่าได้ -32768 ถึง +32767
sign short int num5;	//กำหนดให้ num5 เก็บค่าได้ -32768 ถึง +32767
unsigned short int num6;	//กำหนดให้ num6 เก็บค่าได้ 0 ถึง 65535
long int num7;	//กำหนดให้ num7 เก็บค่าได้ -2147483648 ถึง +2147483647
signed long int num8;	//กำหนดให้ num8 เก็บค่าได้ -2147483648 ถึง +2147483647
unsigned long int num9;	//กำหนดให้ num9 เก็บค่าได้ 0 ถึง +4294967295

ชนิดตัวแปร	จำนวนบิต	ค่าข้อมูลที่เก็บได้
char	8	-128 ถึง +127
signed char	8	-128 ถึง +127
unsigned char	8	0 ถึง +255
int	16	-32768 ถึง +32767
signed int	16	-32768 ถึง +32767
unsigned int	16	0 ถึง 65535
short int	16	-32768 ถึง +32767
signed short int	16	-32768 ถึง +32767
unsigned short int	16	0 ถึง 65535
long int	32	-2147483648 ถึง +2147483647
signed long int	32	-2147483648 ถึง +2147483647
unsigned long int	32	0 ถึง +4294967295
float	32	3.4E-38 ถึง 3.4E+38
double	64	1.7E-308 ถึง 1.7E+308
long double	80	3.4E-4932 ถึง 3.4E+4932

ตารางสรุป คุณสมบัติของตัวแปรแบบต่างๆตามมาตรฐานของ ANSI-C

หมายเหตุ ตามปกติแล้ว การประกาศตัวแปรในภาษาซีนั้น ถ้าใช้รูปแบบการประกาศตัวแปรโดยไม่กำหนด คุณสมบัติเฉพาะ ของตัวแปรด้วย ตัว C-Compiler จะถือว่าตัวแปรนั้นเป็นตัวแปรแบบ sign โดยอัตโนมัติ อยู่แล้ว ตัวอย่างเช่น การใช้รูปแบบ int หรือ sign int จึงถือว่ามีความหมายแบบเดียวกัน ซึ่งส่วนมากแล้วจะนิยมใช้ int แทน signed int เพราะเขียนได้สั้นกว่าแต่มีความหมายเหมือนกัน แต่เมื่อต้องการให้คิดเครื่องหมายจึงใช้ unsigned นำหน้าชนิดของตัวแปรที่ประกาศไว้แทน

การกำหนดชนิดตัวแปรขึ้นมาใช้งานเอง

นอกจากตัวแปรตามมาตรฐานของ ANSI-C ทั้ง 5 ชนิดที่กล่าวไปแล้วนั้น ภาษาซี ยังยอมให้ผู้ใช้งานสามารถกำหนดชนิดตัวแปร ขึ้นมาใช้งานได้เองด้วย โดยใช้คำสั่ง **typedef** โดยมีรูปแบบการใช้งานดังนี้

typedef type name

- **type** คือ ชื่อของชนิดตัวแปรที่ต้องการสร้างขึ้นใหม่
- **name** คือ ชื่อที่ใช้เรียกชนิดตัวแปรจริง(ตัวแปรมาตรฐาน)

โดยวิธีการนี้จะช่วยให้เราสามารถกำหนด ชื่อชนิดของตัวแปร แบบใหม่ๆ ขึ้นมาใช้งานได้เอง ในแบบที่เราต้องการ ซึ่งในกรณี C-Compiler ของ AVR และ Arduino เอง ก็มักใช้วิธีการนี้ เพื่อประกาศสร้างชื่อชนิดของตัวแปร แบบใหม่ขึ้นมาใช้งาน ซึ่งจุดประสงค์หลักก็เพื่อ ลดความยาวในการพิมพ์คำสั่ง ในการเขียนโปรแกรมให้สั้นลงเท่านั้นเอง ตัวอย่างของ ชื่อชนิดของตัวแปร แบบใหม่ๆ ที่พบเห็นในภาษาซีของ AVR และ Arduino ที่จะพบเห็นได้บ่อยๆ ได้แก่ `int8_t`, `uint8_t`, `int16_t`, `uint16_t`, `int32_t`, `uint32_t`,....

โดยถ้าต้องการจะทราบว่า ชื่อชนิดของตัวแปร แบบใหม่ๆ ที่พบเห็นนี้ เมื่อเทียบกับ ชื่อของชนิดตัวแปรของ ภาษาซี ตามมาตรฐานของ ANSI-C แล้วมันคือ อะไร ในกรณีของ Arduino สามารถตรวจสอบได้จากไฟล์ “`stdint.h`” ที่อยู่ใน “`c:\arduino-0012\hardware\tools\avr\avr\include`” ซึ่งเมื่อตรวจสอบดูในไฟล์ของ “`stdint.h`” ก็จะพบ บรรทัด คำสั่งที่ใช้ประกาศสร้าง ชื่อของชนิดตัวแปร ใหม่ๆ ขึ้นมาใช้งาน เช่น

```
typedef signed char int8_t;
typedef unsigned char uint8_t;
typedef signed int int16_t;
typedef unsigned int uint16_t;
typedef signed long int int32_t;
typedef unsigned long int uint32_t;
```

จากตัวอย่างข้างต้นจะเห็นได้ว่า ในความเป็นจริงแล้ว ชื่อชนิดของตัวแปร แบบใหม่ๆ ที่มีการประกาศสร้างขึ้นมาจาก Arduino นั้น ในความเป็นจริงแล้ว มันก็เป็นแค่การกำหนดชื่อใหม่ๆ ขึ้นมาเท่านั้น แต่คุณสมบัติการใช้งานก็จะยังคงเหมือนกับตัวแปรที่ใช้ใน ภาษาซีมาตรฐานตามแบบของ ANSI-C อยู่เหมือนเดิม ซึ่งจุดประสงค์จริงๆ ไม่ได้ต้องการสร้างชนิดของตัวแปรแบบใหม่ขึ้นมาใช้งาน แต่ใช้เพื่อลดความยาวของชื่อ ให้สั้นลง เพื่อที่เวลาเขียนโปรแกรมจะใช้พิมพ์คำสั่งได้สั้นกระชับขึ้น คือ แทนที่จะต้องพิมพ์คำสั่งยาวๆ อย่าง **`unsigned long int`** ก็ใช้ชื่อ **`int32_t`** แทน ซึ่งมีความหมายเหมือนกันทุกประการ แต่เขียนในรูปแบบที่สั้นและกระชับกว่ากันเท่านั้นเอง

สรุปชนิดของตัวแปรใน Arduino ที่ใช้บ่อยๆ

- **boolean** ใช้เก็บค่าข้อมูล เพียง 2 จำนวน คือ TRUE (จริง) และ FALSE (เท็จ)
- **char** ใช้เก็บค่าข้อมูลขนาด 8 บิต ใช้สำหรับเก็บค่ารหัสของตัวอักษร ซึ่งสามารถกำหนดเป็นค่าหรือเขียนตัวอักษรไว้ภายใต้เครื่องหมาย ฟันเดี่ยวก็ได้ เช่น 'A' หรือ 0x41 หรือ 65
- **byte** ใช้เก็บค่าข้อมูลขนาด 8 บิตที่เป็นค่าจำนวนเต็มแบบไม่คิดเครื่องหมาย(เหมือนกันกับ unsigned char ในภาษาซี) ซึ่งสามารถเก็บค่าข้อมูลได้ 256 ค่า คือ 0-255
- **int** หรือ Integer ใช้เก็บค่าข้อมูลขนาด 16บิต ที่เป็นค่าจำนวนเต็ม แบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ -32768 ถึง +32767
- **unsigned int** ใช้เก็บค่าข้อมูลขนาด 16บิต ที่เป็นค่าจำนวนเต็ม แบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 65536 ค่า คือ 0-65535
- **long** ใช้เก็บค่าข้อมูลขนาด 32บิต ที่เป็นค่าเลขจำนวนเต็มแบบคิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่า คือ -2,147,483,648 ถึง 2,147,483,647
- **unsigned long** ใช้เก็บค่าข้อมูลขนาด 32บิต ที่เป็นค่าเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย โดยสามารถใช้เก็บข้อมูลได้ 4294967296 ค่า คือ 0 ถึง 4,294,967,295
- **float** ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมแบบคิดเครื่องหมายขนาด 32 บิต โดยสามารถเก็บค่าได้ระหว่าง $3.4\text{E}-38$ ถึง $3.4\text{E}+38$ ($-3.4028235\text{E}+38$ ถึง $3.4028235\text{E}+38$)
- **double** ใช้เก็บค่าข้อมูลที่เป็นเลขทศนิยมเช่นเดียวกับ float แต่มีค่าความละเอียดดีกว่า float ถึง 2 เท่า สามารถเก็บค่าได้มากถึง $1.7\text{E}+308$
- **void** เป็นตัวแปรแบบที่ไม่มีการเก็บค่าใดๆ คือ ไม่มีค่านั่นเอง
- **arrays** เป็นตัวแปรที่ใช้เก็บข้อมูลหลายๆค่าไว้ในตัวแปรตัวเพียงชื่อเดียว แต่มีตัวเลขสำหรับชี้ตำแหน่งการเก็บข้อมูลต่างกัน โดยตัวเลขที่ใช้ทำหน้าที่เป็นตัวชี้ตำแหน่งของข้อมูล เรียกว่า Index Number โดยค่าลำดับของข้อมูลในตัวแปร array ตำแหน่งแรกจะมีค่าเป็น ศูนย์เสมอ
- **string** เป็นตัวแปรใช้เก็บข้อความ หรือ ตัวอักษรหลายๆตัว ซึ่ง string ก็คือ array ของตัวแปรแบบ char นั่นเอง
- **pointer** เป็นตัวแปรที่ไม่ได้ใช้เก็บข้อมูล แต่ใช้เก็บค่าตำแหน่งแอดเดรสของหน่วยความจำที่ใช้สร้างเป็นตัวแปรสำหรับเก็บข้อมูล ซึ่งตัวแปรแบบนี้จะทำให้ทำหน้าที่เป็นตัวชี้ไปยังตำแหน่งแอดเดรสของตัวแปรอื่นๆอีกทีหนึ่ง