



01076101

วิศวกรรมคอมพิวเตอร์เบื้องต้น

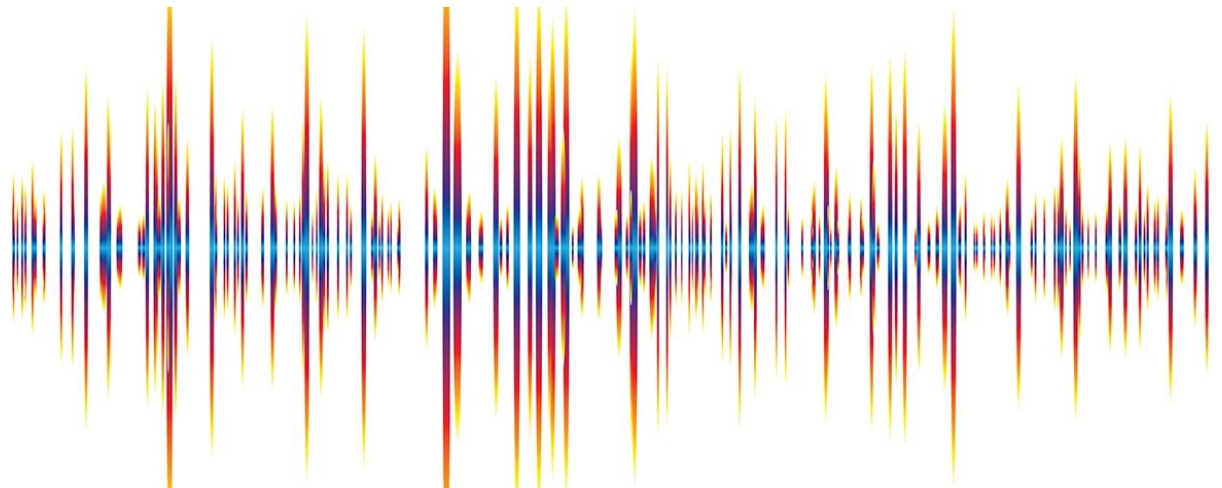
Introduction to Computer Engineering

Arduino #4

Sound, PWM, LED Dot Matrix, OLED



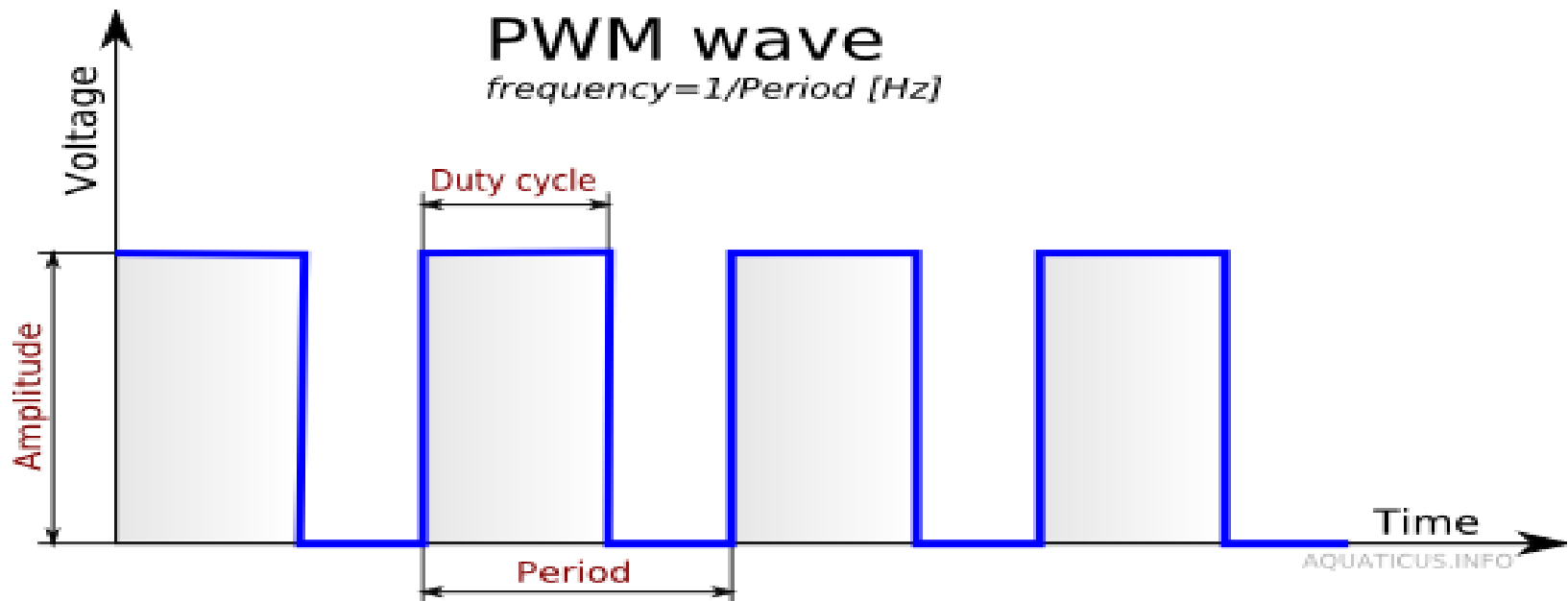
Speaker and Sound





Pulse Width Modulation (PWM)

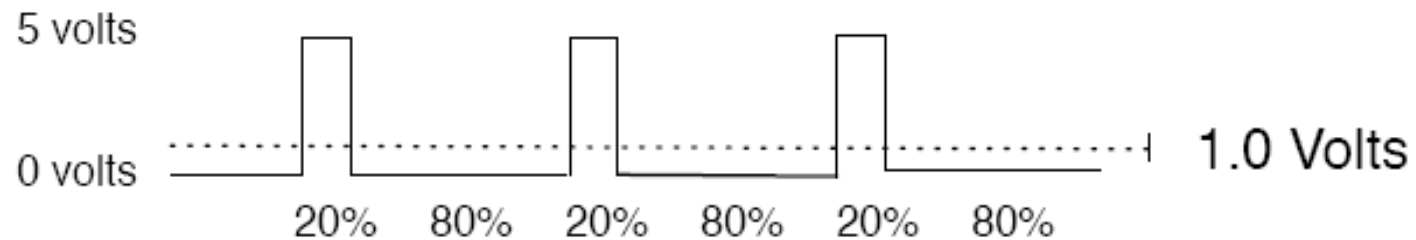
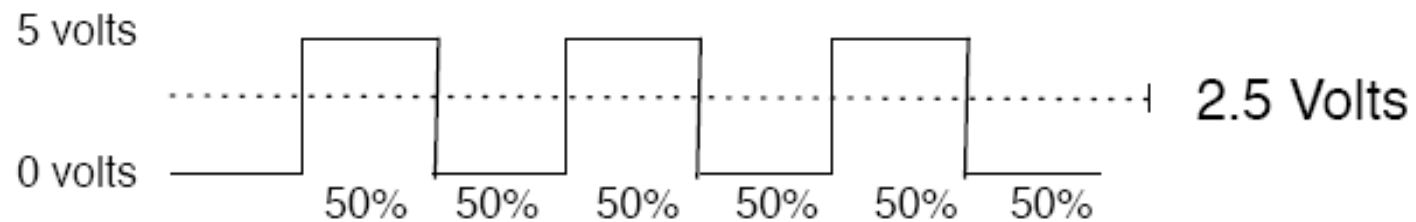
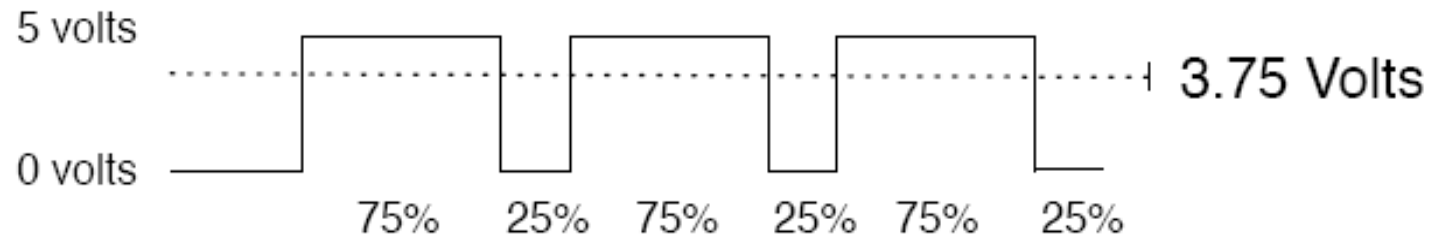
- Pulse Width Modulation หรือ PWM เป็นรูปแบบสัญญาณแบบหนึ่ง ที่ใช้ความกว้างของ Pulse เป็นส่วนสำคัญ



- Duty Cycle: on-time / period



PWM : Adjust duty cycle





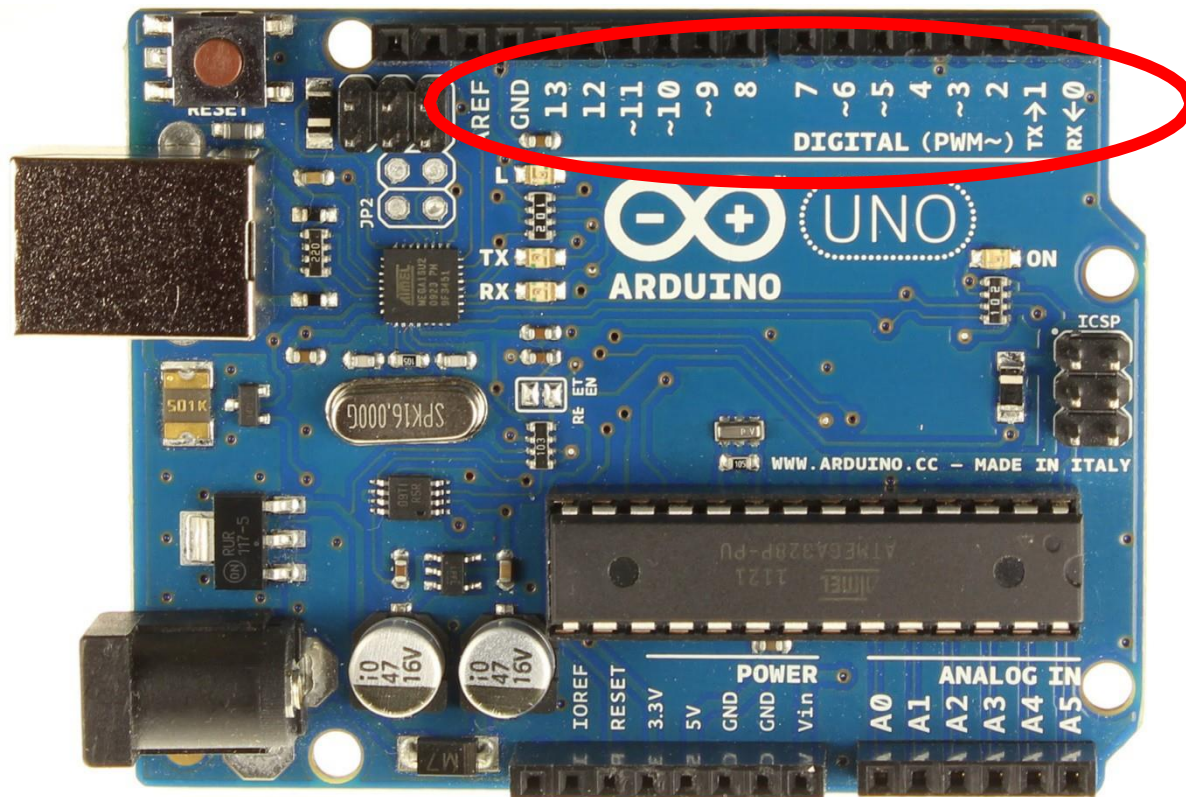
Activity: Fade

- นำ LED ต่อที่ขา ~3, ~5, ~6, ~9, ~10 หรือ ~11 ของ Arduino Board
- ให้โหลดโปรแกรม Examples -> 01.Basic -> Fade
- โหลดลงที่บอร์ด สั่งให้ทำงาน แล้วสังเกตการณ์ทำงาน



Pulse Width Modulation (PWM)

Which pin should be used?



Look at '~' pins 11,10,9,6,5,3



PWM : Analog Write

Syntax:

`analogWrite(pin, duty)`

Parameter:

pin: the pin to write to.

duty: duty cycle in 8 bits(0-255).

Note: The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz.



PWM : Analog Write

```
#define PWM_Pin 3 // Look at '~' pins
11,10,9,6,5,3
#define PWM_off LOW

int duty = 200
void setup()
{
    pinMode(PWM_Pin,OUTPUT);
    digitalWrite(PWM_Pin,PWM_off);
}

void loop()
{
    analogWrite(PWM_Pin,duty);
}
```




Activity : ทดสอบ AnalogWrite

- ให้อ่าน LED 1 ดวง กับขา 11,10,9,6,5,3 (ต่อขาใดขาหนึ่ง)
- ใช้คำสั่ง `analogWrite` ให้ทดลองเปลี่ยน duty cycle 20, 40, 60, 80,100 เปอร์เซ็นต์ โดยให้เว้น 3 วินาที ระหว่างแต่ละ duty cycle ให้สังเกตความสว่างของ LED

Making Sound

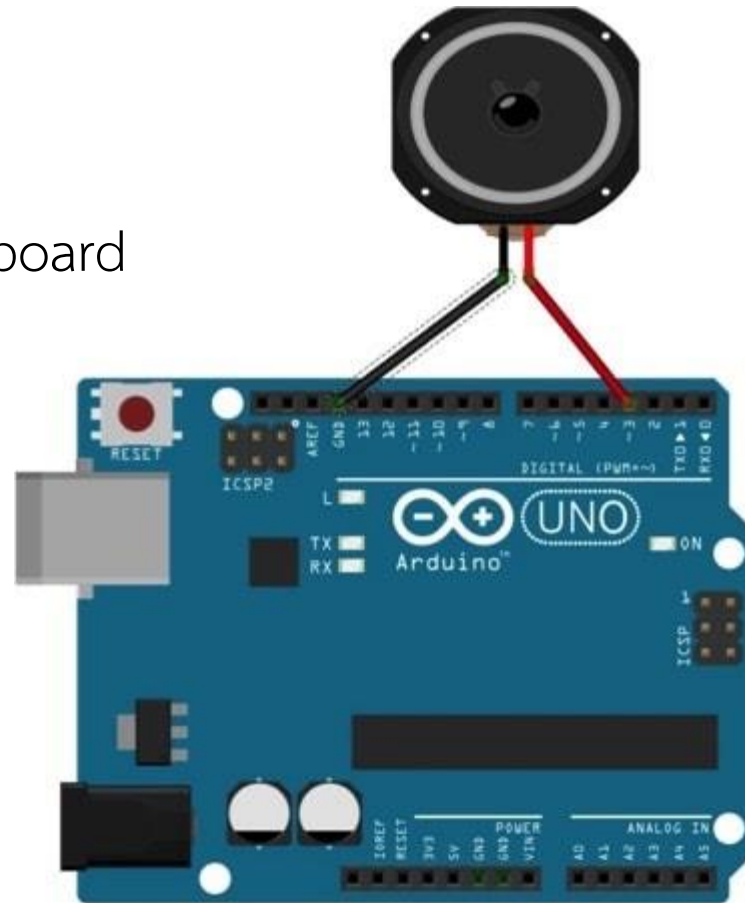


- เสียงเป็นคลื่นความถี่ ลำโพงมีหน้าที่กำเนิดคลื่นความถี่เสียง
- หากเราส่งสัญญาณ High และ Low ไปที่ลำโพงในความถี่ที่เหมาะสม ก็จะเกิดเสียงได้ หากต้องการให้ลำโพงส่งเสียง 440 Hz (ความถี่มาตรฐานของ Note A (ลา))
 - เริ่มจากกำหนดให้ Output ที่ Pin ที่ต่อกับลำโพงเป็น High
 - จากนั้น delay เท่ากับ 1136 microseconds
 - แล้วกำหนดให้ Output ที่ Pin ที่ต่อกับลำโพงเป็น Low
 - จากนั้น delay เท่ากับ 1136 microseconds เช่นกัน
 - เราก็จะได้สัญญาณ 4 เหลี่ยมเท่ากับ 440 Hz



How to Connect

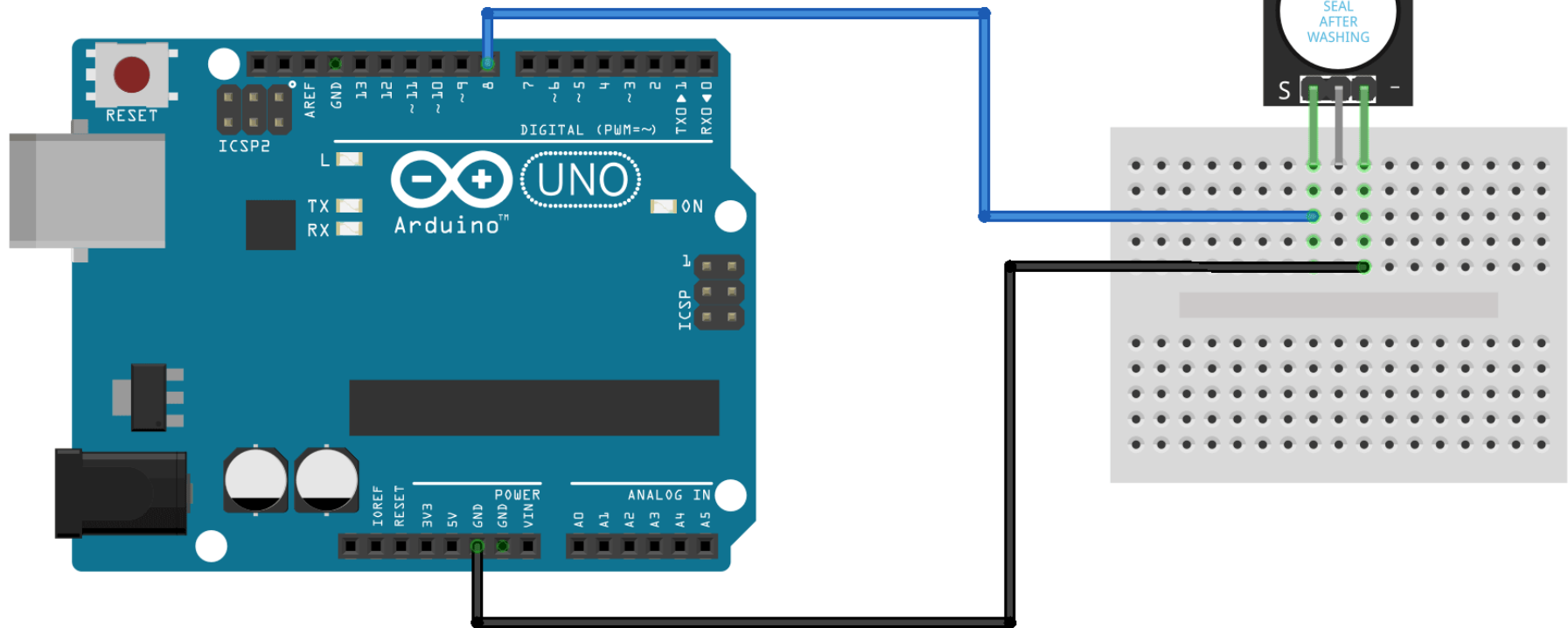
- เอาลำโพงต่อที่ขา 3 หรือขาอื่นๆ
- ควรระวังสายไฟฝั่งลำโพง
อาจจะหักได้ ควรยึดไว้กับ Protoboard
และเชื่อมกับบอร์ด



How to Connect



- Passive Buzzer

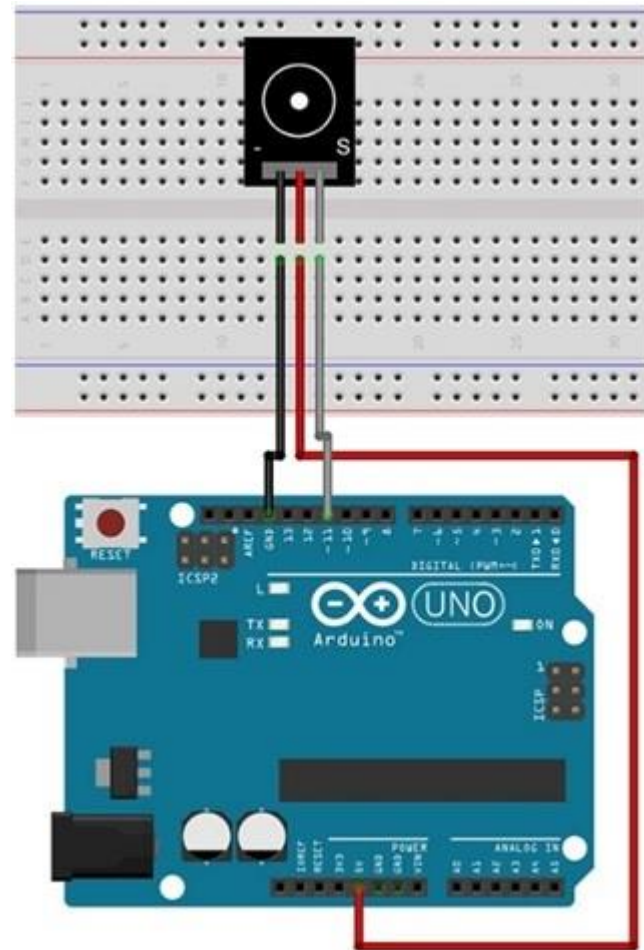


fritzing

How to Connect



- Active Buzzer



Speaker and Sound



- การสร้างเสียงใช้ฟังก์ชัน tone โดยมีรูปแบบดังนี้ (Pin ต้องเป็นขา 3 หรือ 11)

Syntax:

tone(pin, frequency)

tone(pin, frequency, duration)

Parameter:

pin: the number of the pin

- การหยุดเสียงใช้ฟังก์ชัน noTone(pin)



Activity : Play Note

```
int speakerPin = 3;
int numTones = 10;
int tones[] = {261,277,294,311,330,349,370,392,415,440};
// mid C  C#   D    D#   E    F    F#   G    G#   A

void setup()
{
    for (int i=0; i < numTones; i++)
    {
        tone(speakerPin, tones[i]);
        delay(500);
    }
    noTone(speakerPin);
}

void loop()
{
}
```

Play Song



- ในกรณีที่เรากำลังต้องการอ้างอิงความถี่เสียงของ Note ใน Arduino ได้เตรียมไฟล์ pitches.h ให้เราใช้งาน โดยมีค่าอ้างอิงดังนี้

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117

#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262 // เสียงโด (กลาง)
#define NOTE_CS4 277
#define NOTE_D4 294 // เสียงเร (กลาง)
#define NOTE_DS4 311
#define NOTE_E4 330 // เสียงมี (กลาง)
#define NOTE_F4 349 // เสียงฟา (กลาง)
#define NOTE_FS4 370
#define NOTE_G4 392 // เสียงซอล (กลาง)
#define NOTE_GS4 415
#define NOTE_A4 440 // เสียงลา (กลาง)
#define NOTE_AS4 466

#define NOTE_B4 494 // เสียงที (กลาง)
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865

#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
```


Activity Play Song



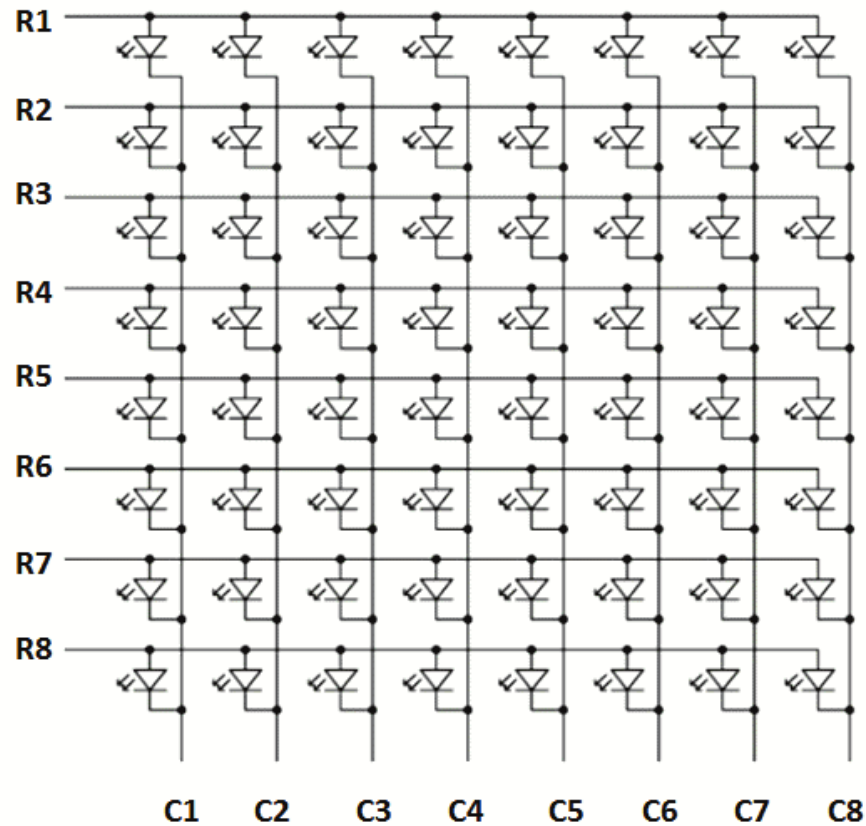
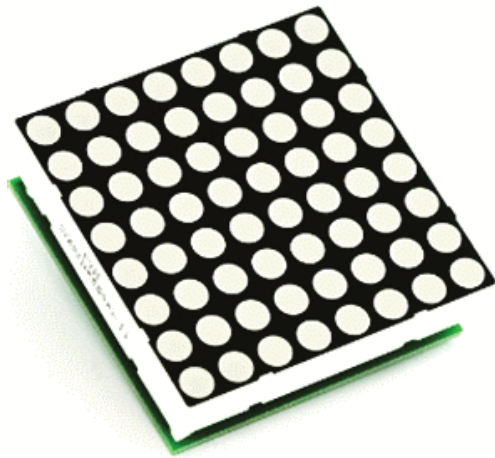
- ให้ต่อลำโพงตามรูป โดยต่อที่ขา 3
- โหลดไฟล์จาก Examples->Digital->toneMelody แล้วลองเล่นเพลง
- จากนั้นโหลดไฟล์จาก FB แล้วลองเล่นเพลง



LED Dot Matrix



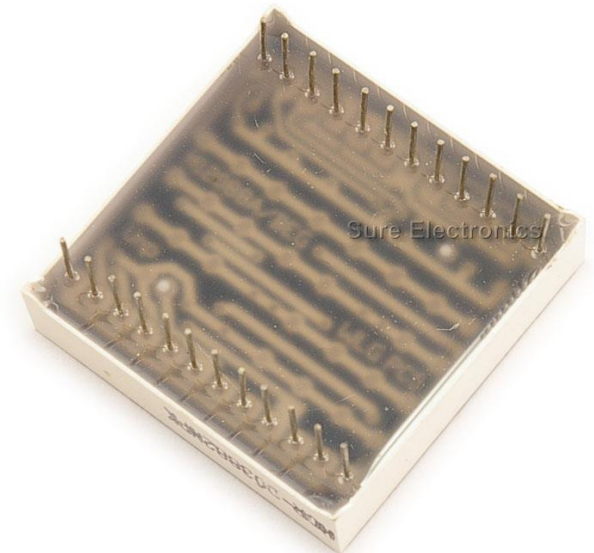
- เป็นอุปกรณ์ที่นำเอา LED จำนวนมากมายไว้ในชิ้นเดียวกัน มีวงจรตามรูป



LED Dot Matrix



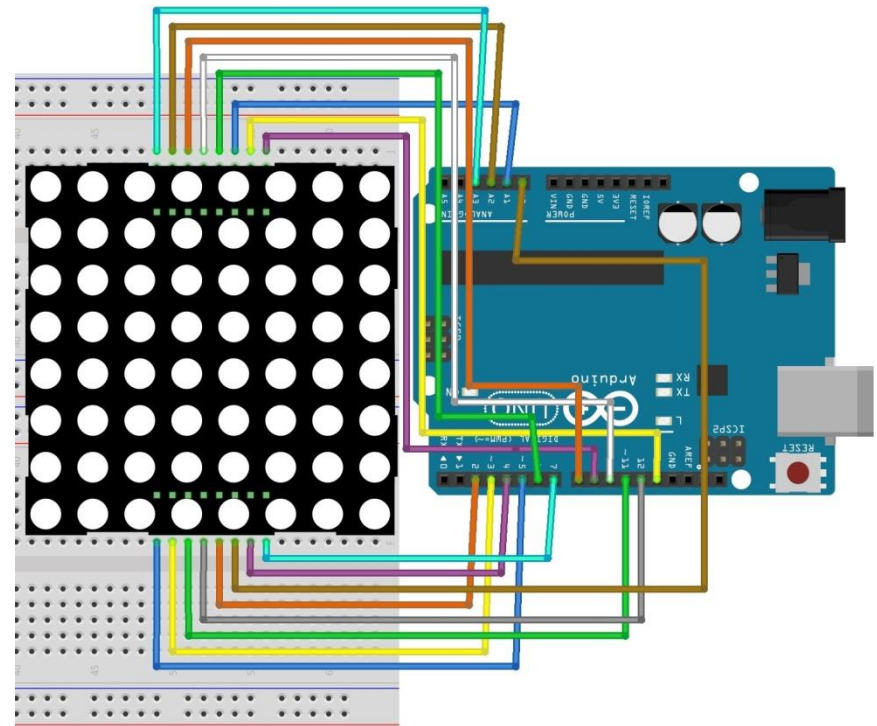
- ตามรูปจะเห็นว่า มีขา 2 แถว
- โดยแถวหนึ่งจะเป็น Row อีกแถวจะเป็น Column



การเชื่อมต่อ Arduino กับ LED Dot Matrix



- เราสามารถเชื่อมต่อ LED Dot Matrix กับ Arduino ได้ ตามวงจรตัวอย่างในรูปแบบ
- จะเห็นว่ามีการใช้สายไฟจำนวนมาก
- ทำให้ไม่สะดวก
- ในรูปไม่ได้ต่อ R เพื่อให้ดูง่าย

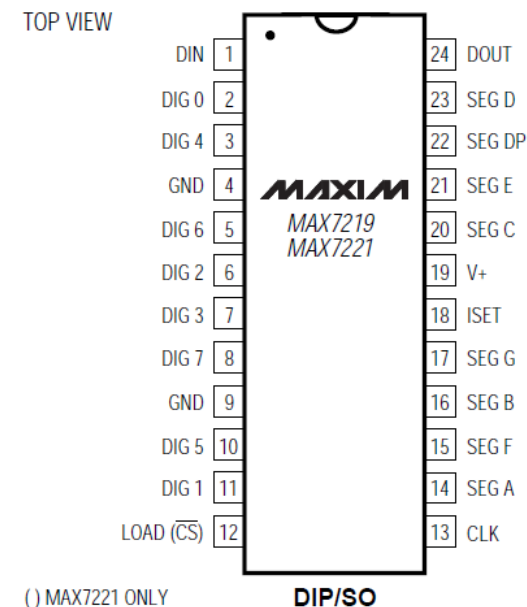


fritzing

การเชื่อมต่อ Arduino กับ LED Dot Matrix



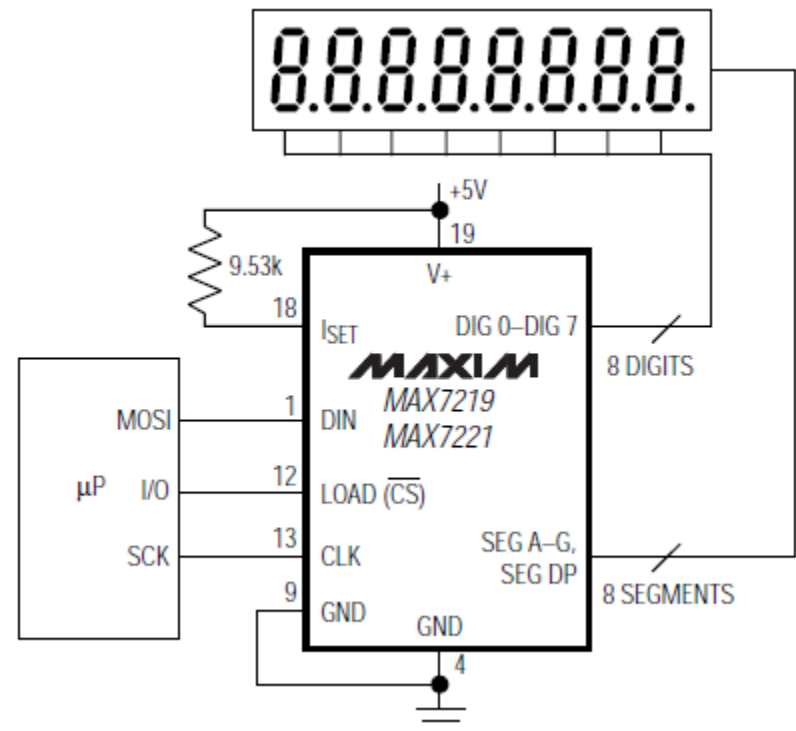
- ปัญหาข้างต้นเกิดขึ้นกับการต่อใช้งาน LED 7 Segment ที่มีหลายหลักเช่นกัน
- จึงได้มีผู้สร้างชิป IC สำหรับใช้ต่อกับ 7 Segment หลายหลัก โดยมีชื่อว่า MAX7219
- MAX7219 สามารถควบคุม LED 7 Segment ได้ 8 หลัก หรือใช้กับ LED Dot Matrix ขนาด 8x8 ได้



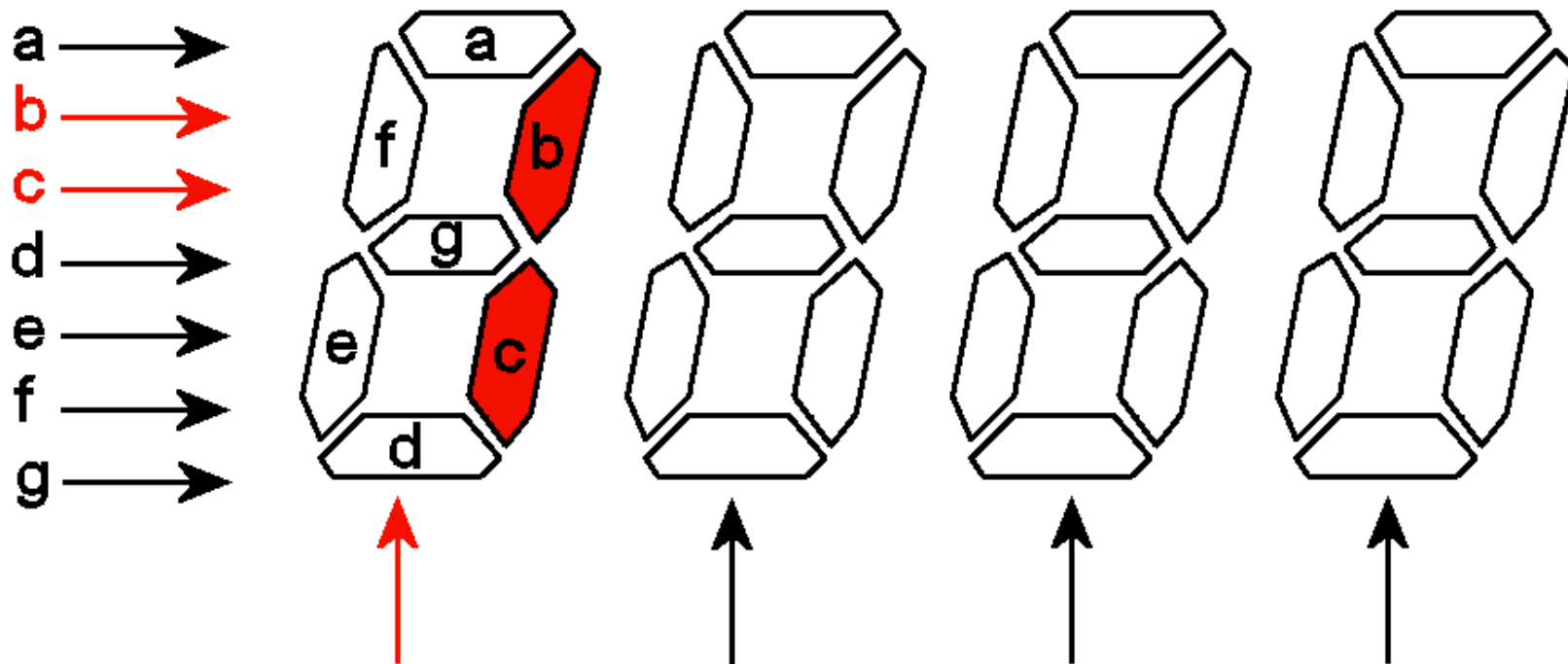
การเชื่อมต่อ Arduino กับ LED Dot Matrix



- ในการต่อ MAX7219 เข้ากับ LED 7 Segment จะต่อขา SEG A-G, DP เข้ากับขา a-g ของแต่ละหลัก และต่อ DIG 0-7 เข้ากับแต่ละหลัก (ตามรูป)
- ในการทำงาน MAX7219 จะส่งข้อมูลของแต่ละหลักไปที่ละครั้ง เช่น ส่งข้อมูลของหลักที่ 1 ในขณะที่ส่ง DIG 0 ออกไป ซึ่งจะทำให้หลักที่ 1 ติด จากนั้นทิ้งไว้ระยะหนึ่งแล้วจึงส่งหลักที่ 2-8 ในทำนองเดียวกัน
- โดยใช้ความเร็วที่เหมาะสม ตาของมนุษย์จะเห็นทุกหลักติดหมด วิธีนี้เรียกว่าการ **Scan**



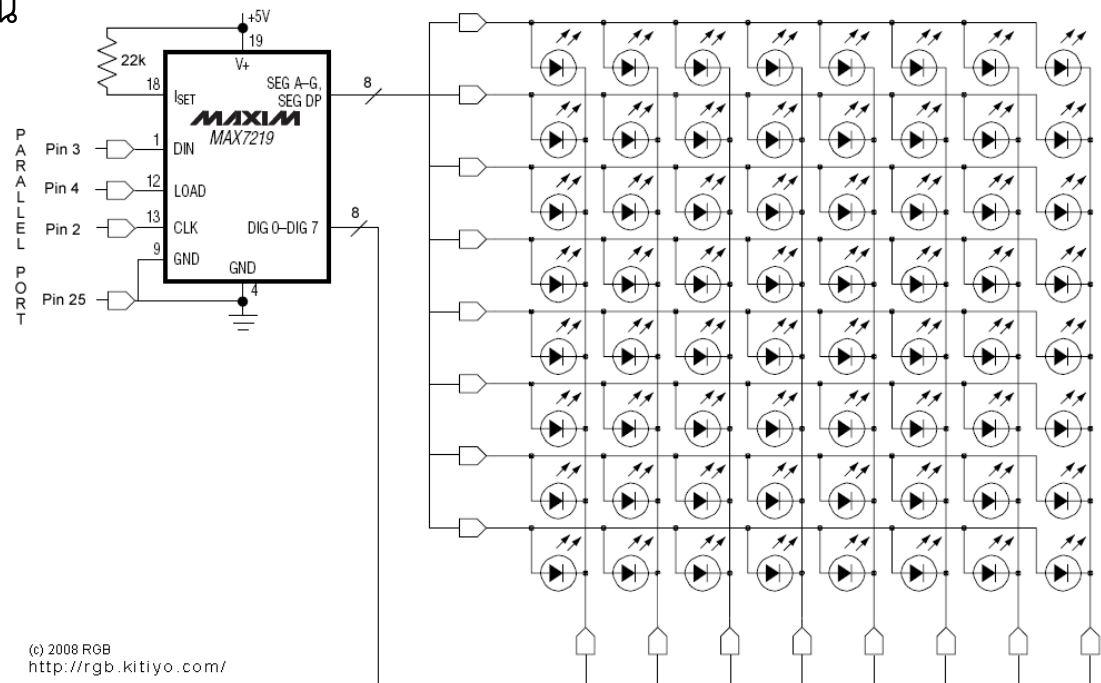
7 Segment Scanning



การเชื่อมต่อ Arduino กับ LED Dot Matrix



- สำหรับการเชื่อมต่อกับ Arduino จะใช้ต่อผ่านขาจำนวน 3 ขา คือ
 - Din เป็นขาสำหรับข้อมูล
 - CLK เป็นขาคlockสำหรับ Sync ข้อมูล โดยคlock 1 สัญญาณ จะหมายถึงข้อมูล 1 บิต
 - LOAD/CS จะใช้บอกว่า จะโหลดข้อมูลแล้ว



(c) 2008 RGB
<http://rgb.kitiyo.com/>



การเชื่อมต่อ Arduino กับ LED Dot Matrix

- การส่งข้อมูลจะส่งเป็นชุด ชุดละ 16 บิต ขั้นตอนจะเริ่มจาก 1) CS เป็น LOW 2) ส่งสัญญาณ CLK 3) ส่งข้อมูลทุกครั้งที่ชอบขาขึ้น จาก D15-> D0 (เส้นแดง)
- รูปแบบการส่งข้อมูลเรียกว่า Protocol : Synchronous Serial Communication

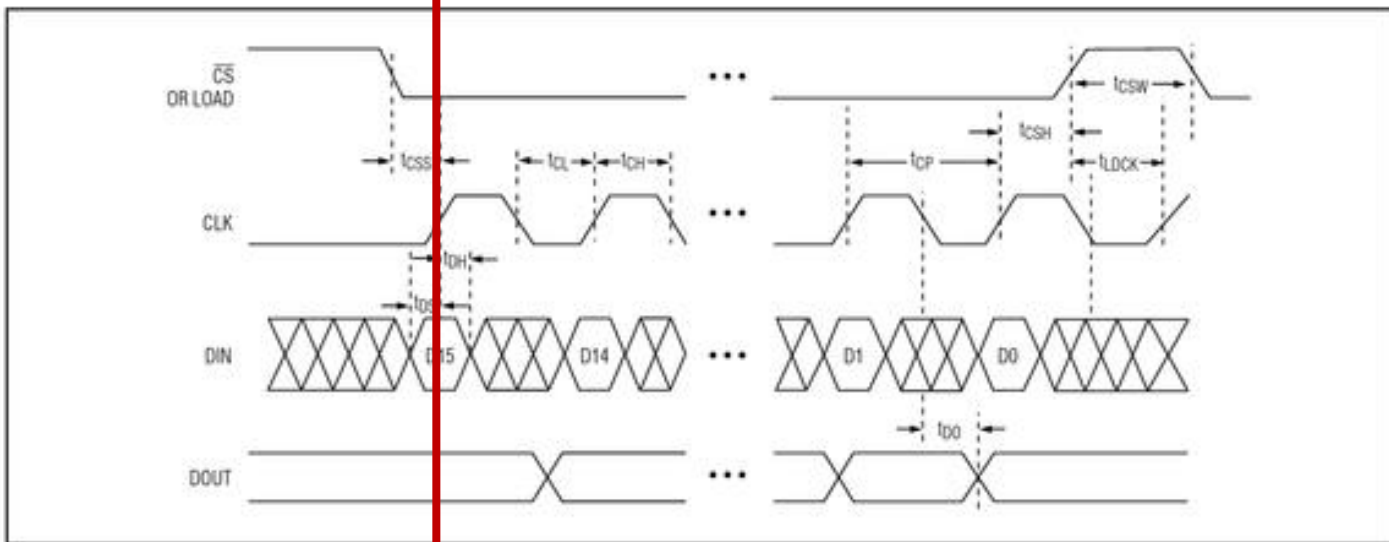


Figure 1. Timing Diagram

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

การเชื่อมต่อ Arduino กับ LED Dot Matrix



- ในชิป MAX7219 จะมีที่เก็บข้อมูลที่เรียกว่า รีจิสเตอร์ (Register) ขนาด 8 บิตอยู่ 14 ตัว
- รีจิสเตอร์แต่ละตัวจะเก็บข้อมูลต่างกัน บางตัวเก็บค่าข้อมูลที่จะแสดง บางตัวเก็บความสว่าง เป็นต้น ชิปจะใช้ข้อมูลจากรีจิสเตอร์เหล่านี้ไปทำงาน หรืออาจจะบอกว่าสามารถสั่งงานชิปผ่านรีจิสเตอร์ก็ได้
- รูปแบบข้อมูลที่ส่งจะเป็นไปตามรูปด้านล่าง โดย D15-D12 จะเป็นอะไรก็ได้ (ไม่สนใจ แต่โดยทั่วไปจะกำหนดเป็น 0)
- ในการระบุว่าการส่งข้อมูลแต่ละครั้ง จะส่งเข้าไปที่ใด จะระบุใน Address (D11-D8) และตามด้วยค่าข้อมูล 8 บิต

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

การเชื่อมต่อ Arduino กับ LED Dot Matrix



- Reg. 1-8 เก็บข้อมูลที่จะแสดง
- Reg. 9 บอกว่าแต่ละหลักจะใช้ decode แบบ BCD หรือไม่
- Reg. A ใช้กำหนดความสว่างโดยมี 15 ระดับ (00h-0Fh)
- Reg. B ใช้กำหนดว่าแสดงหลัก ไตบ้าง (00h-07h)
- Reg. C ค่า 1=ทำงานปกติ 0=หยุดทำงาน

REGISTER	ADDRESS					HEX CODE
	D15-D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	X0
Digit 0	X	0	0	0	1	X1
Digit 1	X	0	0	1	0	X2
Digit 2	X	0	0	1	1	X3
Digit 3	X	0	1	0	0	X4
Digit 4	X	0	1	0	1	X5
Digit 5	X	0	1	1	0	X6
Digit 6	X	0	1	1	1	X7
Digit 7	X	1	0	0	0	X8
Decode Mode	X	1	0	0	1	X9
Intensity	X	1	0	1	0	XA
Scan Limit	X	1	0	1	1	XB
Shutdown	X	1	1	0	0	XC
Display Test	X	1	1	1	1	XF

การเชื่อมต่อ Arduino กับ LED Dot Matrix



```
#include <SPI.h>

const int CS_PIN    = 10;  // SPI /SS
const int CLK_PIN   = 13;  // SPI SCK
const int DIN_PIN    = 11;  // SPI MOSI

void MAX7219_write_reg( uint8_t addr, uint8_t data ) {
    digitalWrite( CS_PIN, LOW );
    SPI.transfer( addr );
    SPI.transfer( data );
    digitalWrite( CS_PIN, HIGH );
}

#define REG_DIGIT(x)      (0x1+(x))
#define REG_DECODE_MODE  (0x9)
#define REG_INTENSITY    (0x7)
#define REG_SCAN_LIMIT   (0xB)
#define REG_SHUTDOWN      (0xC)
#define REG_DISPLAY_TEST  (0xF)

void MAX7219_init(void) {
    MAX7219_write_reg( REG_DECODE_MODE, 0x00 ); // decode mode: no decode for digits 0-7
    MAX7219_write_reg( REG_INTENSITY, 0x07 );   // set intensity: 0x07=15/32
    MAX7219_write_reg( REG_SCAN_LIMIT, 0x07 );  // scan limit: display digits 0-7
    MAX7219_write_reg( REG_SHUTDOWN, 0x01 );    // shutdown: normal operation
    MAX7219_write_reg( REG_DISPLAY_TEST, 0x00 ); // display test: no display test
}
```

การเชื่อมต่อ Arduino กับ LED Dot Matrix



```
void setup() {  
    SPI.begin();  
    SPI.setBitOrder( MSBFIRST );  
    SPI.setClockDivider( SPI_CLOCK_DIV16 ); // 16MHz/16 -> 1MHz SCK frequency  
    SPI.setDataMode( SPI_MODE0 ); // use SPI mode 0  
    pinMode( CS_PIN, OUTPUT );  
    digitalWrite( CS_PIN, HIGH );  
    MAX7219_init();  
}  
  
void flashing() {  
    MAX7219_write_reg( REG_SHUTDOWN, 0x01 ); // normal operation  
    MAX7219_write_reg( REG_DISPLAY_TEST, 0x01 ); // enter display test mode  
    delay(100);  
    MAX7219_write_reg( REG_DISPLAY_TEST, 0x00 ); // exit display test mode  
    MAX7219_write_reg( REG_SHUTDOWN, 0x00 ); // shutdown operation  
    delay(900);  
}
```

การเชื่อมต่อ Arduino กับ LED Dot Matrix



```
const byte char_data[][8] = { // 'C', 'E'
  { B00000000,
    B10000001,
    B10000001,
    B10000001,
    B10000001,
    B10000001,
    B10000001,
    B11111111,
    B00000000 },
  { B00000000,
    B10000001,
    B10000001,
    B10010001,
    B10010001,
    B10010001,
    B11111111,
    B00000000 }
};

void show_ce() {
  static uint8_t ch=0;
  for (uint8_t i=0; i < 8; i++) {
    MAX7219_write_reg( REG_DIGIT(i), char_data[ch][i] );
  }
  delay(500);
  ch = (ch+1) % 2;
}
```

การเชื่อมต่อ Arduino กับ LED Dot Matrix

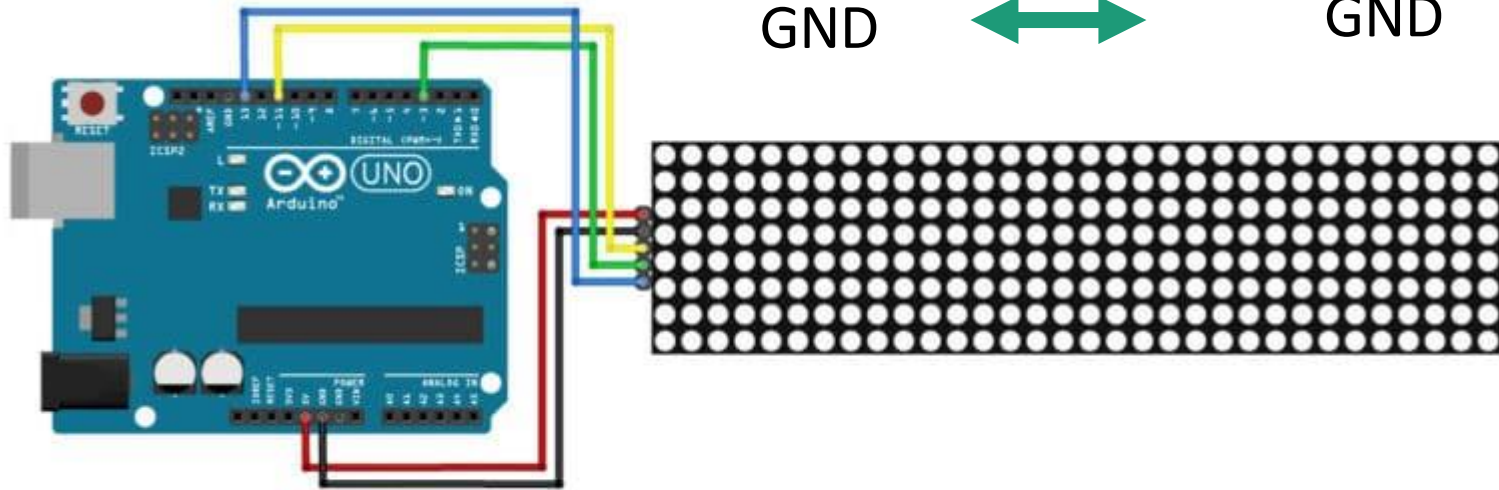


```
void loop() {  
    for (uint8_t i=0; i < 3; i++) {  
        flashing();  
    }  
    MAX7219_write_reg( REG_SHUTDOWN, 0x01 );    // normal operation  
    while (1) {  
        show_ce();  
    }  
}
```



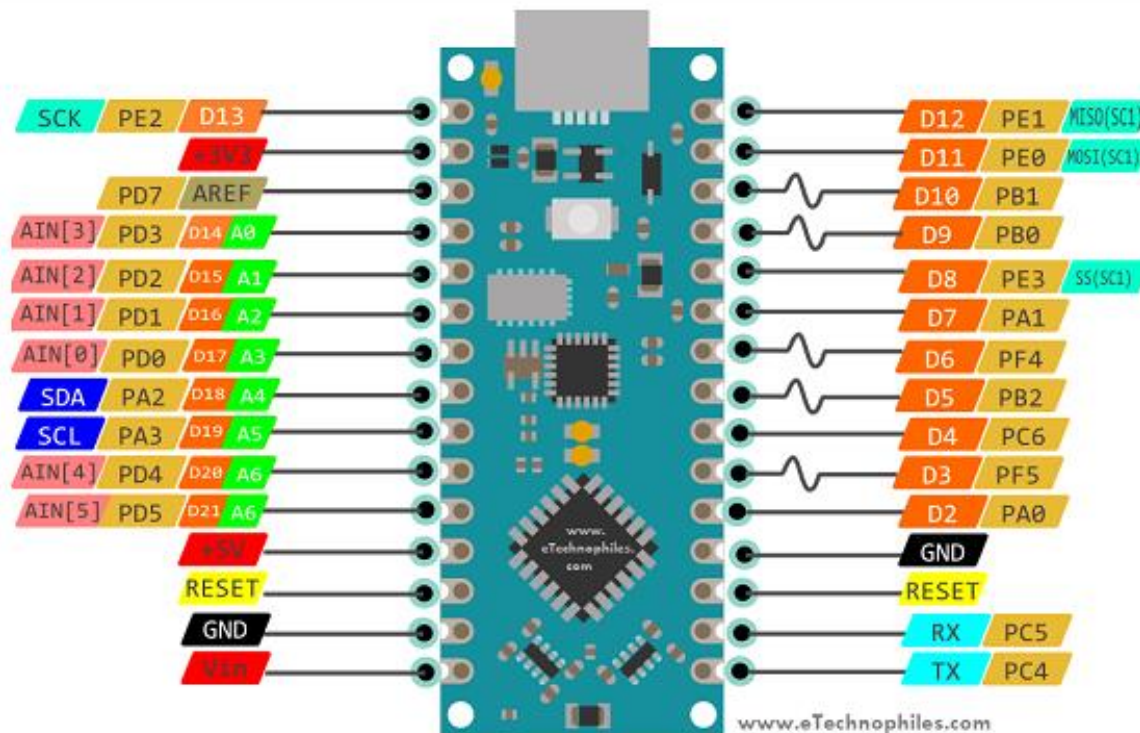
การต่อ Arduino กับ Dot Matrix LED

LED		Arduino
VCC	↔	5V
DIN	↔	11
CLK	↔	13
CS	↔	10
GND	↔	GND















fritzing

Arduino Nano Pinout

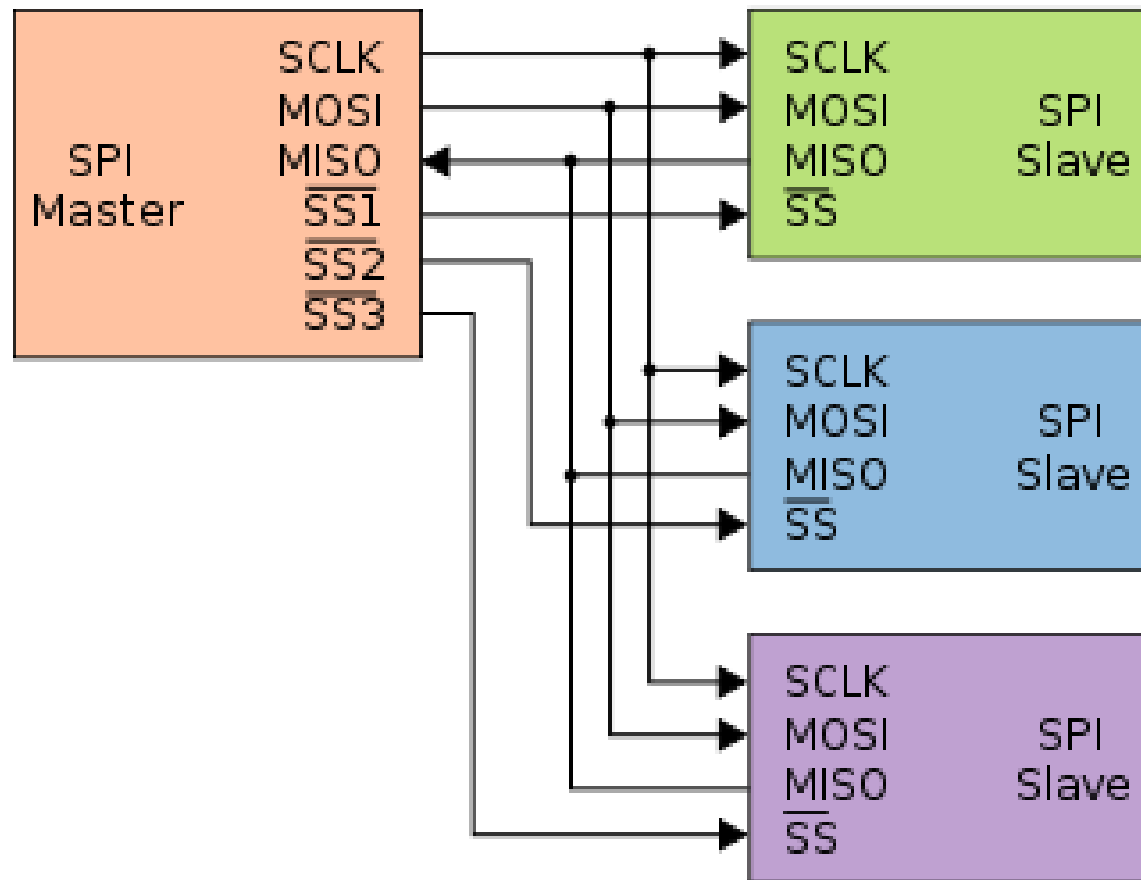


**Arduino
Nano
Every
Pinout**

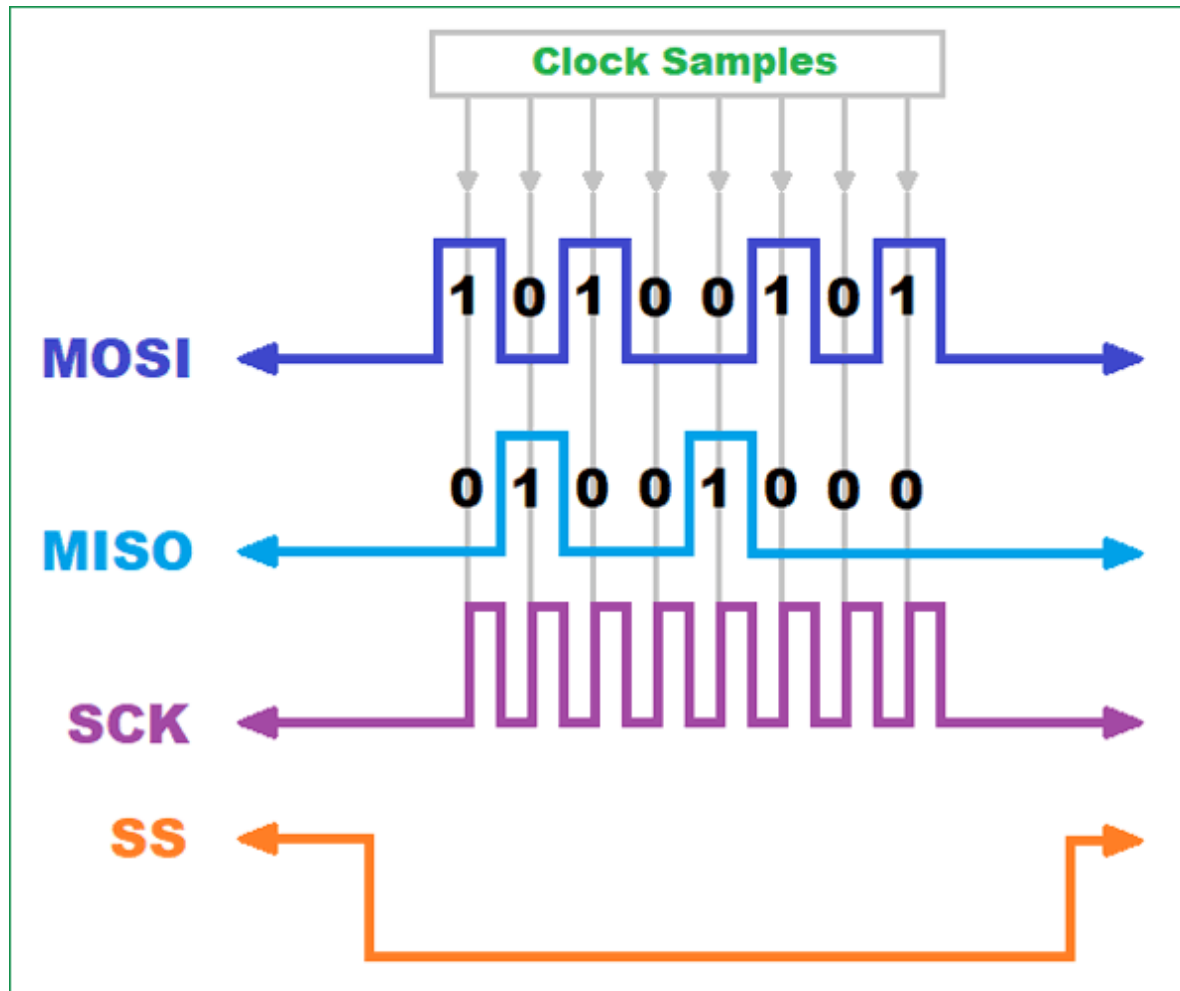
	Power		Ground		Analog Input		Default
	I2C Pins		AREF		Microcontroller Port		Digital Pins
	UART		RESET		PWM		SPI pins

[illegible]

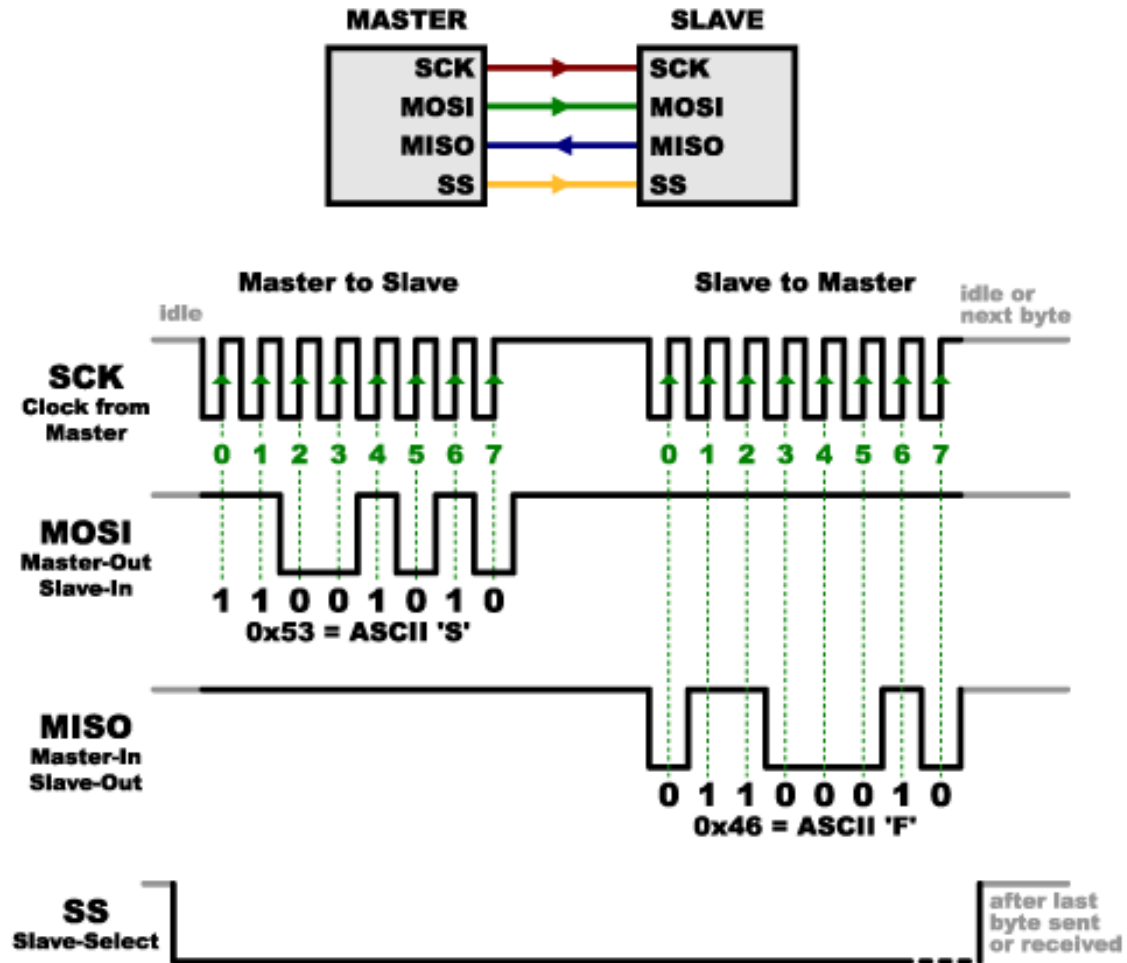
SPI Protocol



SPI Protocol



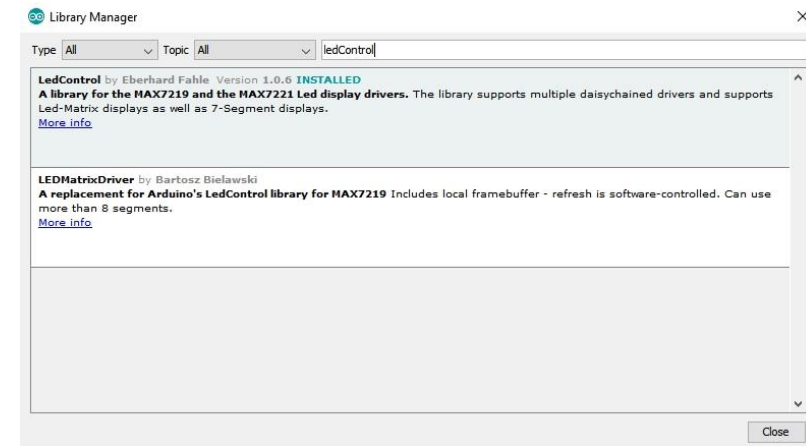
SPI Protocol



การเชื่อมต่อ Arduino กับ LED Dot Matrix



- จากวิธีการที่กล่าวมาข้างต้น แม้จะแสดงผลบนจอ LED Dot Matrix ได้ แต่การเอาไปใช้ก็ยุ่งยากอยู่
- จึงได้มีผู้สร้าง Library ขึ้นมาใช้งาน ซึ่งก็มีจำนวนมาก Library ตัวหนึ่งที่นิยมใช้กันมีชื่อว่า LedControl
- การติดตั้ง
 - ไปที่ Sketch -> Include Library -> Manage Libraries
 - จะปรากฏหน้าต่าง Library Manager ให้ป้อน LedControl แล้วเลือกติดตั้ง



การเชื่อมต่อ Arduino กับ LED Dot Matrix



```
#include "LedControl.h"

LedControl lc=LedControl(11,13,10,4); // CLK,DIN,CS,Number of LED Module

unsigned long delaytime=500; // time to updates of the display

void setup() {
    int devices=lc.getDeviceCount(); // find no of LED modules
    //we have to init all devices in a loop
    for(int address=0;address<devices;address++) { // set up each device
        lc.shutdown(address,false);
        lc.setIntensity(address,8);
        lc.clearDisplay(address);
    }
}

void loop() {
    int devices=lc.getDeviceCount(); // find no of LED modules

    for(int row=0;row<8;row++) {
        for(int col=0;col<8;col++) {
            for(int address=0;address<devices;address++) {
                delay(delaytime);
                lc.setLed(address,row,col,true);
                delay(delaytime);
                lc.setLed(address,row,col,false);
            }
        }
    }
}
```

การเชื่อมต่อ Arduino กับ LED Dot Matrix



- ฟังก์ชัน **setLed(addr, row, col, T/F)** จะรับข้อมูลประกอบด้วย addr คือ โมดูล row,col ค่าแถวและคอลัมน์ และ T/F คือ ให้ติดหรือดับ
- ฟังก์ชัน **setRow(addr, row, value)** จะรับข้อมูลประกอบด้วย addr คือ โมดูล row คือ แถวที่จะแสดง และ value คือค่า 8 บิตที่จะให้แสดง
- ฟังก์ชัน **setColumn(int addr, int col, byte value)** ใช้งานเช่นเดียวกับ setRow

การเชื่อมต่อ Arduino กับ LED Dot Matrix



- สำหรับการแสดงผลตัวอักษร มีผู้ที่ทำข้อมูล 8x8 เอาไว้

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx	
x0		▶		0	P	'	p	9	ē	ā	⋮	L	μ	α	≡		
x1	☺	◀	!	1	A	Q	a	q	ü	æ	ī	☒	⊥	τ	β	±	
x2	☹	‡	"	2	B	R	b	r	é	ŕ	ó	☒	τ	π	Γ	≥	
x3	♥	!!	#	3	C	S	c	s	ä	ö	ū		†	μ	π	≤	
x4	♦	¶	\$	4	D	T	d	t	ä	ö	ñ	†	—	ε	Σ	Γ	
x5	♣	§	%	5	E	U	e	u	ä	ö	ñ	†	+	F	σ	J	
x6	♠	=	&	6	F	V	f	v	ä	ö	ñ	†	†	π	μ	÷	
x7	•	♣	'	7	G	W	g	w	ä	ö	ñ	†	†	π	μ	÷	
x8	☐	↑	(8	H	X	h	x	ä	ö	ñ	†	†	π	μ	÷	
x9	○	↓)	9	I	Y	i	y	ä	ö	ñ	†	†	π	μ	÷	
xA	☒	→	*	:	J	Z	j	z	ä	ö	ñ	†	†	π	μ	÷	
xB	♂	←	+	;	K	L	k	l	ä	ö	ñ	†	†	π	μ	÷	
xC	♀	⊥	,	<	L	\	l	!	ä	ö	ñ	†	†	π	μ	÷	
xD	♂	←	+	=	M	I	m	}	i	ä	ö	ñ	†	†	π	μ	÷
xE	♂	←	+	>	N	^	n	~	ä	ö	ñ	†	†	π	μ	÷	
xF	☒	▼	/	?	O	_	o	Δ	Δ	f	>>	†	†	π	μ	÷	

<http://www.gammon.com.au/forum/?id=11516>

Activity LED Dot Matrix



- ให้โหลดโปรแกรมจาก FB แล้วทดสอบการทำงาน
- ลองเปลี่ยน ข้อความ ให้เป็นไปตามที่ต้องการ



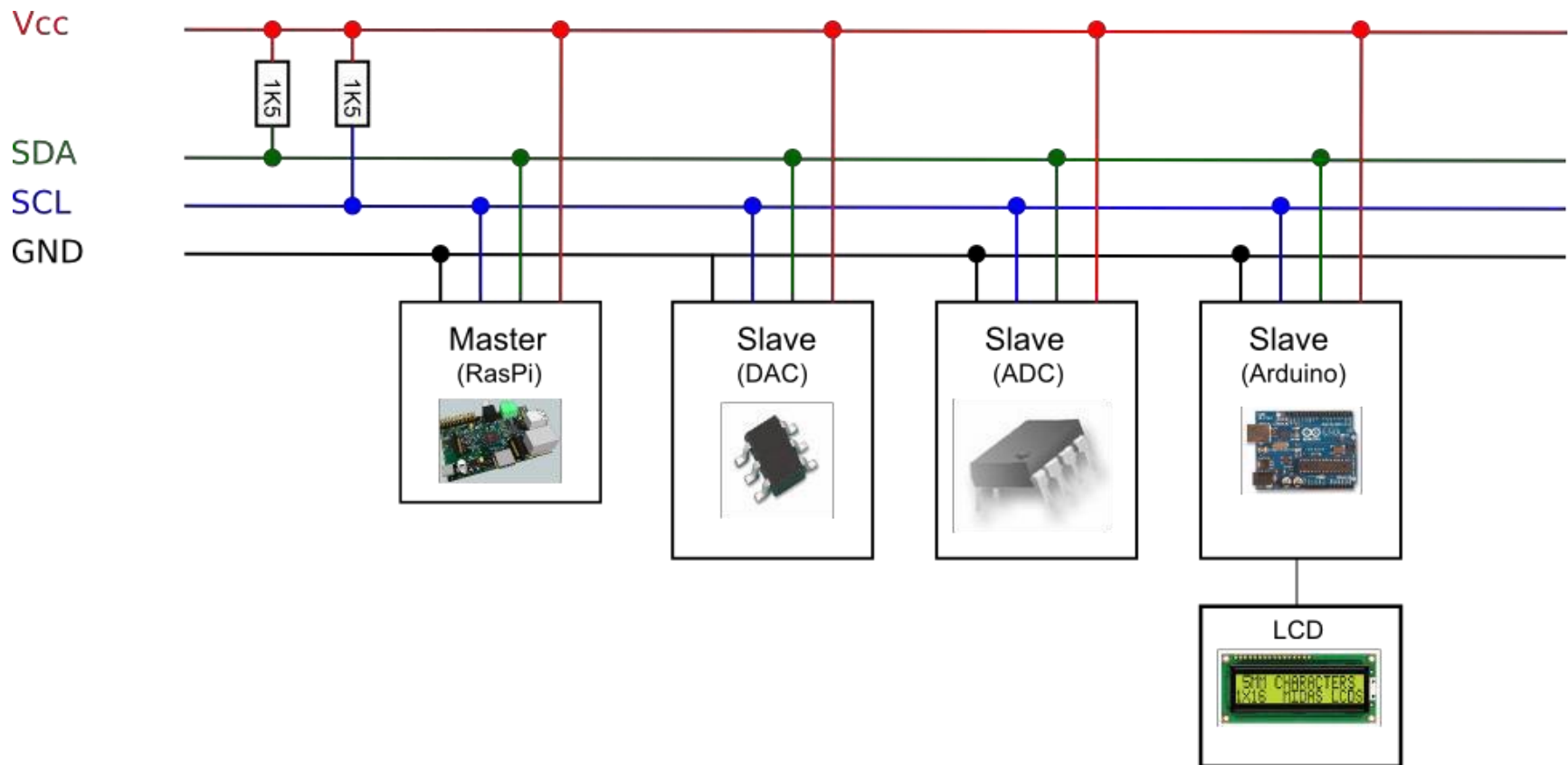
I2C : Inter-Integrated Circuit Bus

- I²C ย่อมาจาก Inter IC Communication
- พัฒนาขึ้นโดยบริษัท Philips
- สามารถควบคุม-ติดต่อ-สั่งงานระหว่างชิป เพียงสายสัญญาณ 2 เส้น
- สามารถต่ออุปกรณ์ร่วมบนบัสได้หลายตัวและแต่ละตัวไม่จำเป็นต้องมีไฟเลี้ยงวงจรที่เท่ากัน
- มีสายสัญญาณ 2 เส้น คือ SDA และ SCL เป็น bi-directional line
- มีความเร็วในการส่งข้อมูล 100k (bits/sec) ที่โหมดปกติ และที่ fast mode 400k (bits/sec) ที่ high speed mode 3.4M (bits/sec)

I²C : Inter-Integrated Circuit Bus



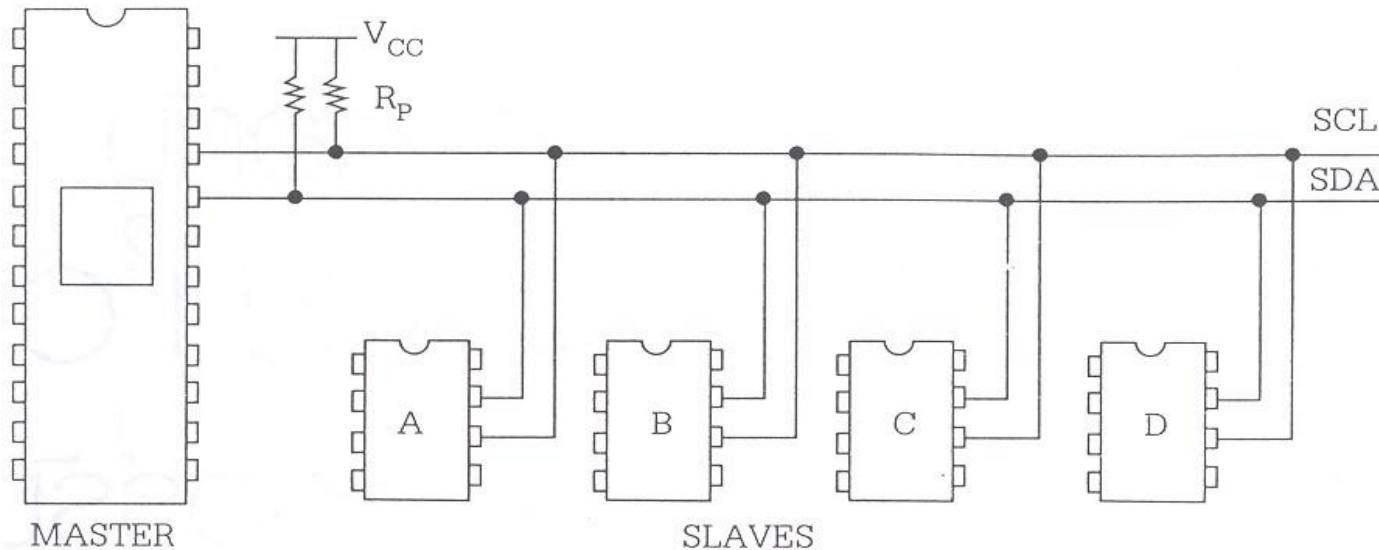
- I²C ย่อมาจาก Inter IC Communication หมายถึงการติดต่อสื่อสารระหว่างไอซี



I²C : Inter-Integrated Circuit Bus



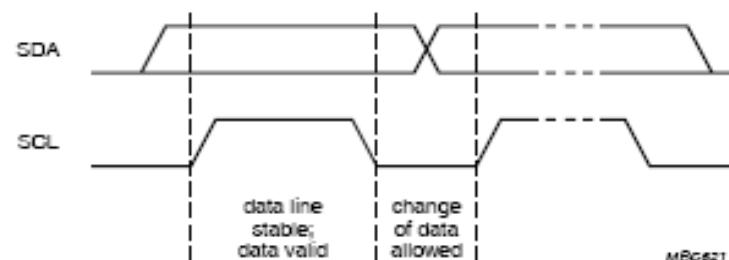
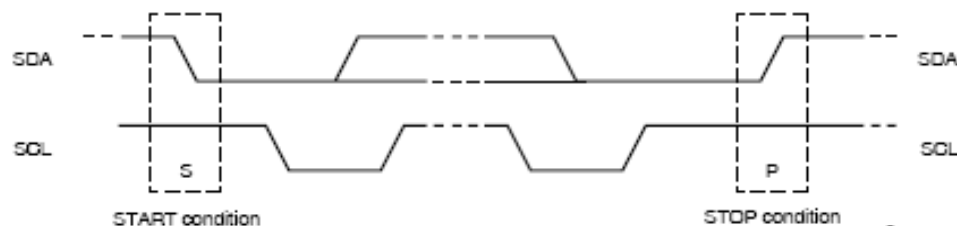
- ใช้สองสายได้แก่ Serial Data Line (SDA) ในการรับส่ง ข้อมูลแบบอนุกรมและ Serial Clock Line (SCL) เป็น สายสัญญาณ Clock สามารถรับส่งข้อมูลด้วยความเร็ว 100 kbits/s



I²C : Inter-Integrated Circuit Bus



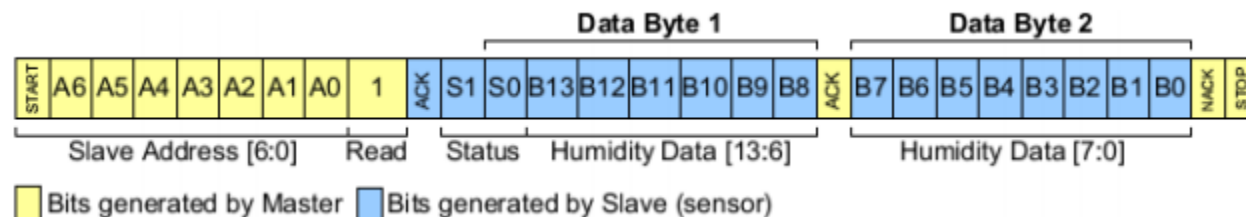
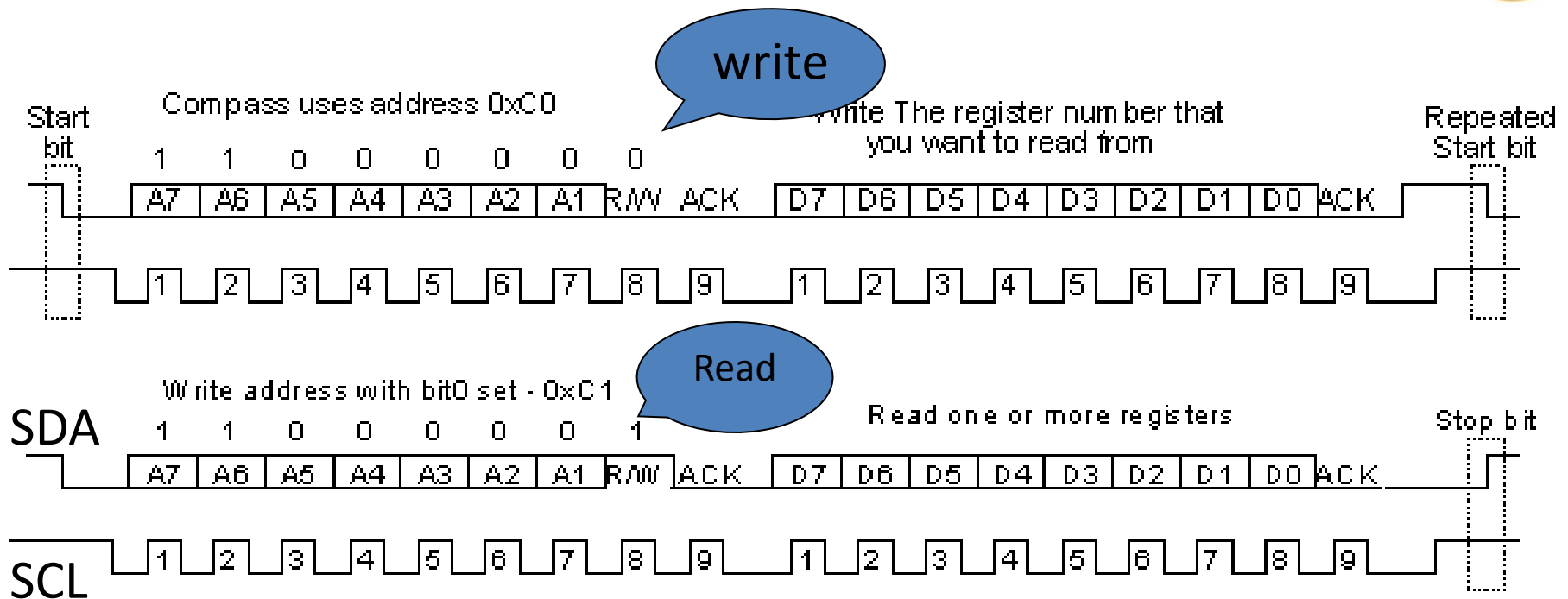
สถานะบนระบบบัส	SDA	SCL
1. Bus not busy	Hi	Hi
2. Start data transfer		Hi
3. Stop		Hi
4. Data valid การรับส่งข้อมูล 1 บิต ใช้ clock 1 ลูก ขณะรับ/ส่ง ข้อมูลบน SDA ต้องคงที่ และ SCL Hi ข้อมูลบน SDA เปลี่ยนแปลงได้เมื่อ SCL Lo	X Change	Hi Lo
5. Acknowledge ตัวรับจะส่งสัญญาณ Ack เมื่อรับข้อมูลครบ 1 byte แล้ว	Lo	Hi







ตัวอย่าง Bit sequence





การประยุกต์ใช้งาน I²C

- ใช้กับไมโครคอนโทรลเลอร์ร่วมกับ ไอซีเพื่อขยายพอร์ต ไอซีแปลงสัญญาณ DAC/ADC ไอซี RTC ไอซีหน่วยความจำ
- ใช้กับไมโครคอนโทรลเลอร์ร่วมกับอุปกรณ์ เช่น LCD, OLED



Speed of various connectivity methods (bits/sec)

CAN (1 Wire)	33 kHz (typ)
I ² C ('Industrial', and SMBus)	100 kHz
SPI	110 kHz (original speed)
CAN (fault tolerant)	125 kHz
I ² C	400 kHz
CAN (high speed)	1 MHz
I ² C 'High Speed mode'	3.4 MHz
USB (1.1)	1.5 MHz or 12 MHz
SCSI (parallel bus)	40 MHz
Fast SCSI	8-80 MHz
Ultra SCSI-3	18-160 MHz
Firewire / IEEE1394	400 MHz
Hi-Speed USB (2.0)	480 MHz

ตารางสรุปเปรียบเทียบข้อดีข้อเสียของระบบ Serial Bus



Serial Bus Comparison Summary

Pros and Cons of the different buses

UART	CAN	USB	SPI	I ² C
<ul style="list-style-type: none">• Well Known• Cost effective• Simple	<ul style="list-style-type: none">• Secure• Fast	<ul style="list-style-type: none">• Fast• Plug&Play HW• Simple• Low cost	<ul style="list-style-type: none">• Fast• Universally accepted• Low cost• Large Portfolio	<ul style="list-style-type: none">• Simple• Well known• Universally accepted• Plug&Play• Large portfolio• Cost effective
<ul style="list-style-type: none">• Limited functionality• Point to Point	<ul style="list-style-type: none">• Complex• Automotive oriented• Limited portfolio• Expensive firmware	<ul style="list-style-type: none">• Powerful master required• No Plug&Play SW - Specific drivers required	<ul style="list-style-type: none">• No Plug&Play HW• No "fixed" standard	<ul style="list-style-type: none">• Limited speed



- | | | | | | | | |
|--------------------|---|---|---|----------------------|----|----|-------------------|
| x | x | x | x | A2 | A1 | A0 | R/ \overline{W} |
| └────────────────┘ | | | | └────────────────┘ | | | |
| บิตแอดเดรสคงที่ | | | | บิตแอดเดรสโปรแกรมได้ | | | |



โปรแกรมหา Address ของ I2C

```
#include <Wire.h>
void setup()
{
    Wire.begin();
    Serial.begin(9600);
    while (!Serial);           // wait for serial monitor
    Serial.println("\nI2C Scanner");
}
void loop()
{
    byte error, address;
    int nDevices;

    Serial.println("Scanning...");
```



โปรแกรมหา Address ของ I²C

```
nDevices = 0;
for(address = 1; address < 127; address++ )
{
    // The i2c_scanner uses the return value of the Write.endTransmission
    // to see if a device did acknowledge to the address.
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
        Serial.print("I2C device found at address 0x");
        if (address<16)
            Serial.print("0");
        Serial.print(address, HEX);
        Serial.println("  !");

        nDevices++;
    }
}
```

โปรแกรมหา Address ของ I²C



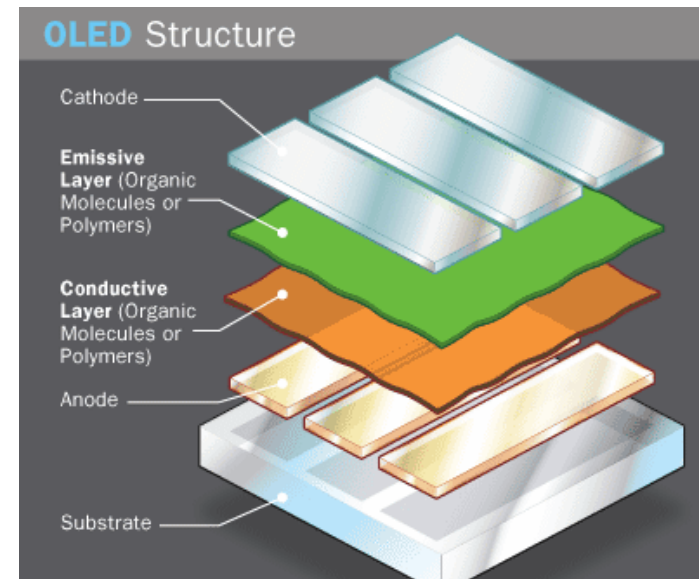
```
else if (error==4)
{
    Serial.print("Unknown error at address 0x");
    if (address<16)
        Serial.print("0");
    Serial.println(address,HEX);
}
}
if (nDevices == 0)
    Serial.println("No I2C devices found\n");
else
    Serial.println("done\n");

delay(5000);           // wait 5 seconds for next scan
}
```

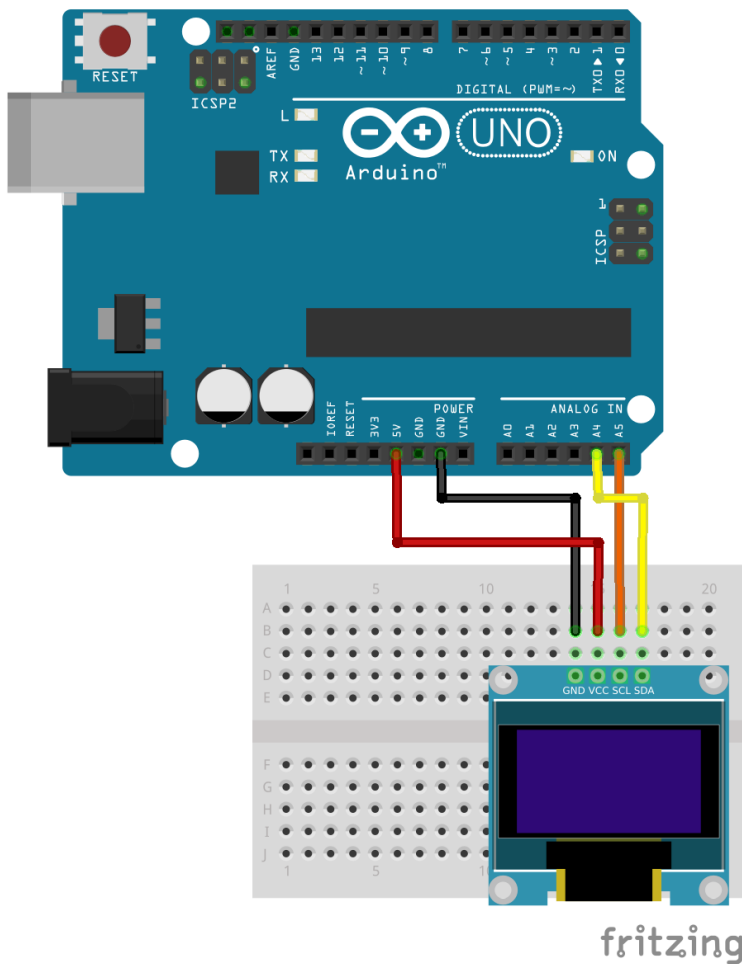
OLED



- OLED (Organic Light Emitting Diodes) คือจอภาพที่มีลักษณะคล้ายแผ่นฟิล์ม
- มีส่วนประกอบเป็นสารอินทรีย์ที่สามารถเปล่งแสงเองได้เมื่อได้รับพลังงานไฟฟ้า เรียกว่ากระบวนการอิเล็กโตรลูมิเนสเซนส์ (Electroluminescence)
- ไม่ต้องพึ่งพาแสง Backlight และจะไม่มี การเปล่งแสงในบริเวณที่เป็นภาพสีดำ ส่งผลให้สีดำนั้นดำสนิท
- มีความสว่างมากกว่า



การต่อ OLED กับ Arduino



OLED

VCC

SCL

SDA

GND



Arduino

3.3V,5V

A5

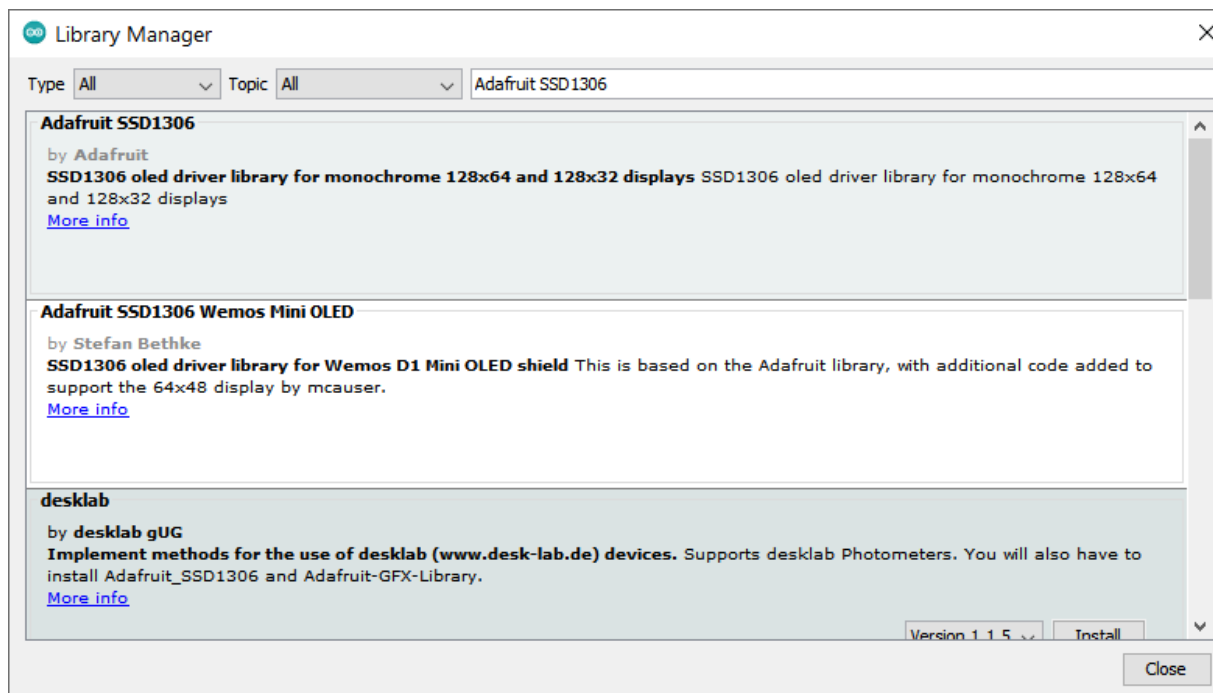
A4

GND

OLED



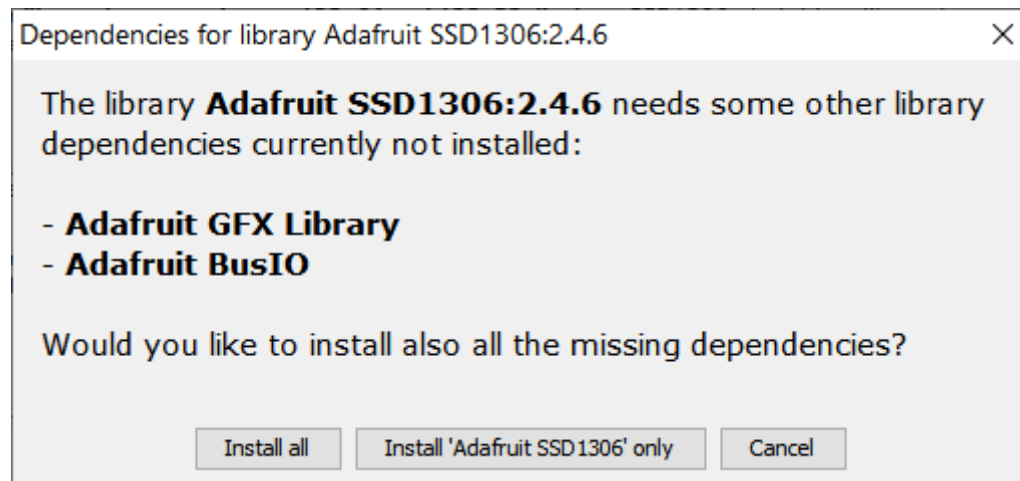
- โมดูล OLED ที่ใช้มีขนาด 0.91 นิ้ว จำนวนจุดภาพ 128x32
- ต้องติดตั้ง Library
 - ไปที่ Sketch -> Include Library -> Manage Libraries
 - จะปรากฏหน้าต่าง Library Manager ให้ป้อน Adafruit SSD1306 แล้วเลือกติดตั้ง



OLED



- ระบบจะแสดงหน้าต่าง ให้เลือก Install All



Activity



- ให้ต่อวงจร OLED กับ Arduino
- หา Address ของ OLED



การใช้งาน Class OLED

- `OLED.begin(SSD1306_SWITCHCAPVCC, 0x3C);`
 - กำหนดโมเดลของ OLED และ Address ของ OLED
- `OLED.clearDisplay();` -> ล้างหน้าจอ
- `OLED.setTextColor(WHITE);` ->
 - กำหนดสีของตัวอักษร
- `OLED.setCursor(15,0);`
- `OLED.setTextSize(2);`
- `OLED.println("Hello");`
- `OLED.display();`



ตัวอย่างโปรแกรม

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET -1

Adafruit_SSD1306 OLED(OLED_RESET);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

void setup()
{
  Serial.begin(115200);
  OLED.begin(SSD1306_SWITCHCAPVCC, 0x3C);
}
```

```
void loop()
{
  OLED.clearDisplay();

  OLED.setTextColor(WHITE);
  OLED.setCursor(10,0);
  OLED.setTextSize(2);
  OLED.println("Hello");
  OLED.setCursor(10,15);
  OLED.println("World!");
  OLED.display();
}
```

ฟังก์ชันอื่นๆ ใน Class OLED



- `OLED.setRotation(rotation); //can be 0, 1, 2 or 3.`
- `OLED.dim(dim) // Dim = true, false`
- `OLED.setTextColor(color,background)`
- `OLED.startscrolldiagleft(start,stop)`
- `OLED.startscrolldiagright(start,stop)`
- `OLED.startscrollleft(start,stop)`
- `OLED.startscrollright(start,stop)`
- `OLED.stopscroll()`

Activity

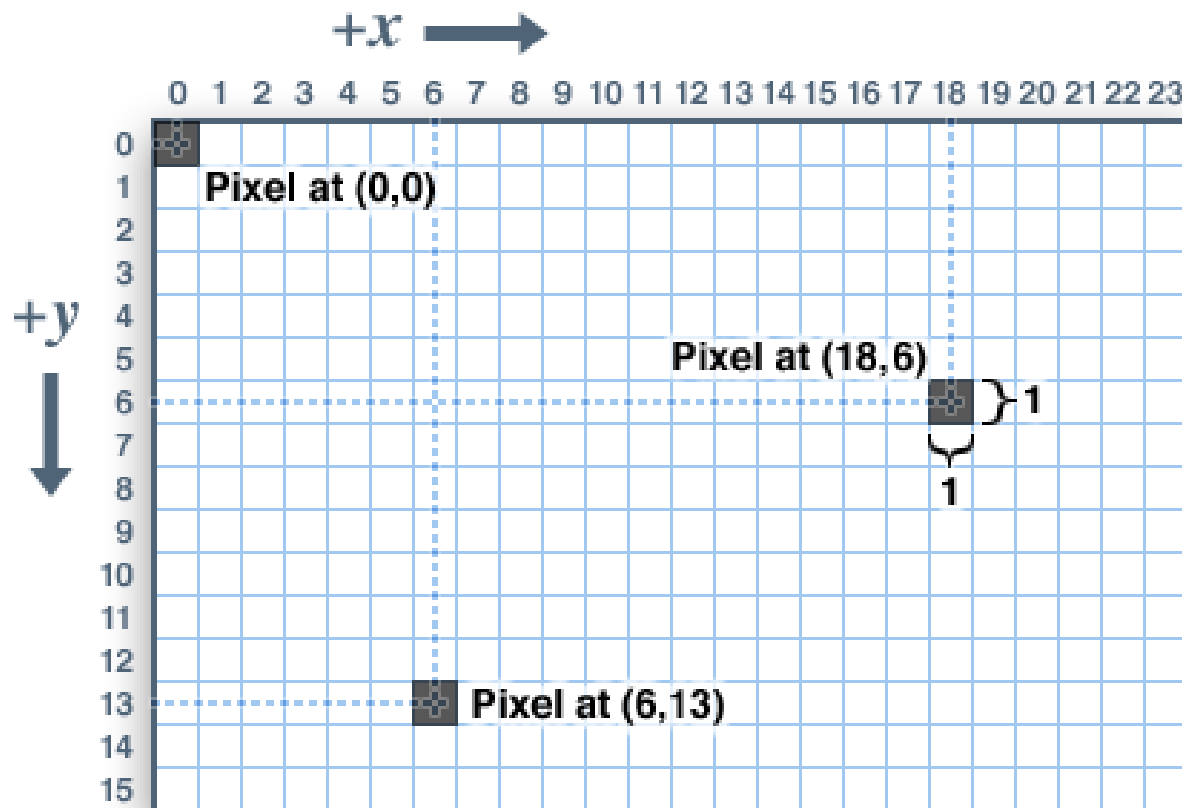


- ให้ทดลองใช้ Function ต่างๆ ของ OLED
- ให้โหลดไฟล์ File -> Examples -> Adafruit SSD1306->ssd1306_128x32_i2c
- ทดลองโหลดให้ทำงาน

การใช้งานกราฟิกใน OLED



- การนับจุด





การใช้งานกราฟิกใน OLED

- คำสั่ง Plot จุด
 - `OLED.drawPixel(x, y,color);`
- คำสั่งลากเส้น
 - `OLED.drawLine(x0,y0,x1,y1,color);`
- คำสั่งวาดรูปสี่เหลี่ยม
 - `OLED.drawRect(x0,y0, w,h,color);`
 - `OLED.fillRect(x0,y0, w,h,color);`
- คำสั่งวาดรูปวงกลม
 - `OLED.drawCircle(x0,y0,radius,color);`
 - `OLED.fillCircle(x0,y0,radius,color);`
- คำสั่งวาดรูปสี่เหลี่ยมหัวมน
 - `OLED.drawRoundRect(x0,y0,w,h,radius,color);`
 - `OLED. fillRoundRect(x0,y0,w,h,radius,color);`
- คำสั่งวาดรูปสามเหลี่ยม
 - `OLED.drawTriangle(x0,y0,x1,y1,x2,y2,color);`

Activity

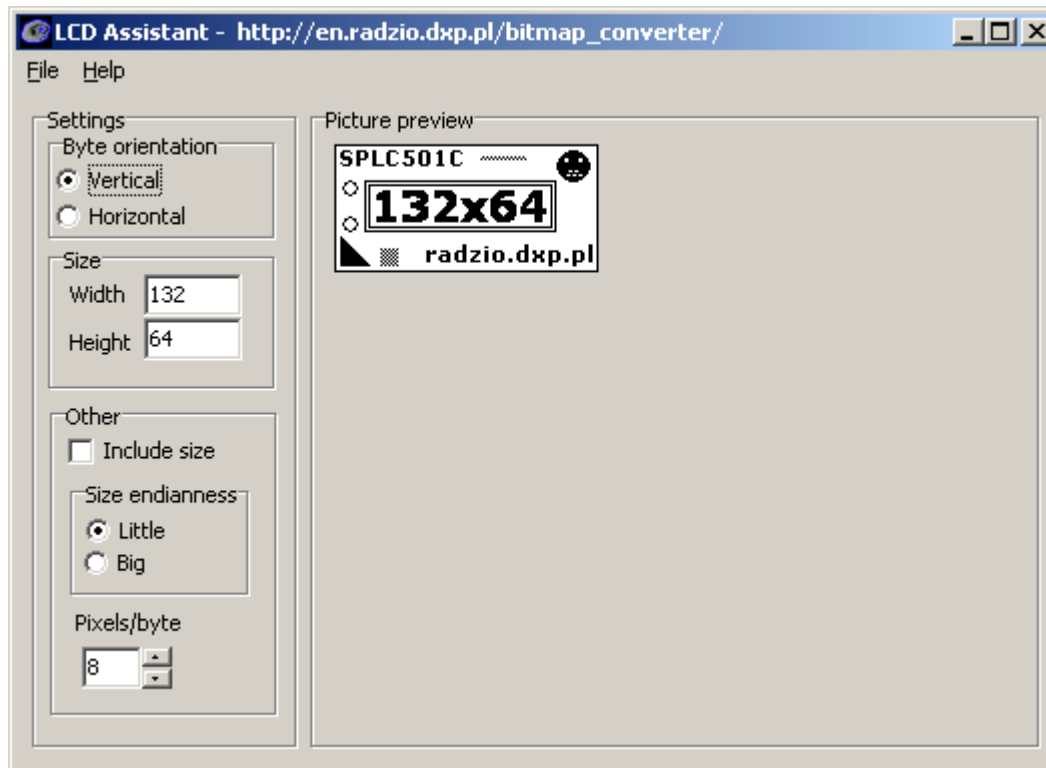


- ให้เขียนโปรแกรมแสดงกราฟิกดังนี้
 - วาดรูปสี่เหลี่ยมใหญ่สุด จากนั้นห้วงเวลา .25 วินาที และลดขนาดลงมาด้านละเท่าๆ กัน แล้ววาดใหม่จำนวน 5 รูป จะได้รูปสี่เหลี่ยมซ้อนกัน 5 รูป
 - วาดรูปสี่เหลี่ยมใหญ่สุด จากนั้นห้วงเวลา .25 วินาที และลดขนาดลงมาด้านละเท่าๆ กัน ลบหน้าจอแล้ววาดใหม่จำนวน 5 รูป จะได้รูปสี่เหลี่ยมลดขนาดลง 5 รูป
 - ทำตามขั้นตอนข้างต้น แต่ให้เปลี่ยนเป็นวงกลม

การ Upload ภาพกราฟิกเพื่อแสดงใน OLED



- ไปที่ http://en.radzio.dxp.pl/bitmap_converter/
- ดาวน์โหลดไฟล์ LCD_Assistance.zip



การ Upload ภาพกราฟิกเพื่อแสดงใน OLED



- หารูปที่มีขนาดไม่เกิน 128x32 โดยเป็นแบบ BW
- เลือก File -> Load Image แล้วโหลดไฟล์เข้ามา
- เลือก File -> Save Output แล้วตั้งชื่อไฟล์อะไรก็ได้
- จากนั้นให้เปิดไฟล์ที่ Save จะได้ Array ของรูปภาพ
- ให้ copy array มาไว้ในโปรแกรมตามตัวอย่าง
- `OLED.drawBitmap(40, 10, images, 48, 48, WHITE);`
 - พารามิเตอร์ 1,2 คือ ตำแหน่งเริ่มต้น
 - พารามิเตอร์ 4,5 คือ ขนาดกว้างและสูง โดยความกว้างจะต้องเป็นตัวเลขที่หารด้วย 8 ลงตัว
 - โดยหากจุดภาพทางขวาเป็นช่องว่างที่เกิน mod 8 โปรแกรม LCD_Assistance จะตัดจุดภาพออก แต่หากไม่เป็นช่องว่าง จะเติม 0 เข้าไป

โปรแกรมแสดงภาพ



```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define OLED_RESET -1 Adafruit_SSD1306 OLED(OLED_RESET);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

static const unsigned char PROGMEM images[] =
{
};

void setup()
{
  Serial.begin(115200);
  OLED.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C (for the 128x64)
  OLED.clearDisplay();

  // miniature bitmap display
  OLED.drawBitmap(40, 10, images, 48, 48, WHITE);
  OLED.display();
}

void loop()
{
}
```



Assignment #4 : game

- ให้สร้างเกม Pong บน LED Dot Matrix หรือ OLED
- ขนาด 16x8 (Dot) หรือ 64x32 (OLED) หรือ ตามความเหมาะสม
- ควบคุมโดยปุ่มสวิตช์ หรือ R ปรับค่าได้
- จะทำแบบ Bar 1 ด้าน หรือ 2 ด้านก็ได้
- ให้มีเสียงประกอบด้วย
- กำหนดส่งหลังสอบกลางภาค
- คะแนน 10 คะแนน (ชิ้นงานหลัก)
- หากพบว่าลอกมาจากที่อื่น จะได้คะแนน 0



For your attention