



01076001

วิศวกรรมคอมพิวเตอร์เบื้องต้น

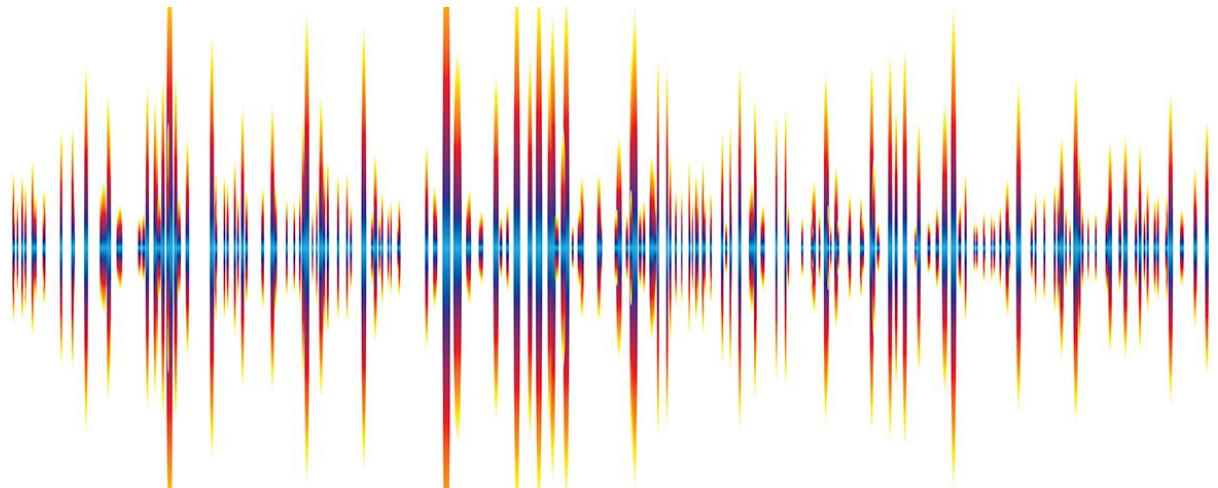
Introduction to Computer Engineering

Arduino #3

Sound, PWM, LED Dot Matrix



# Speaker and Sound

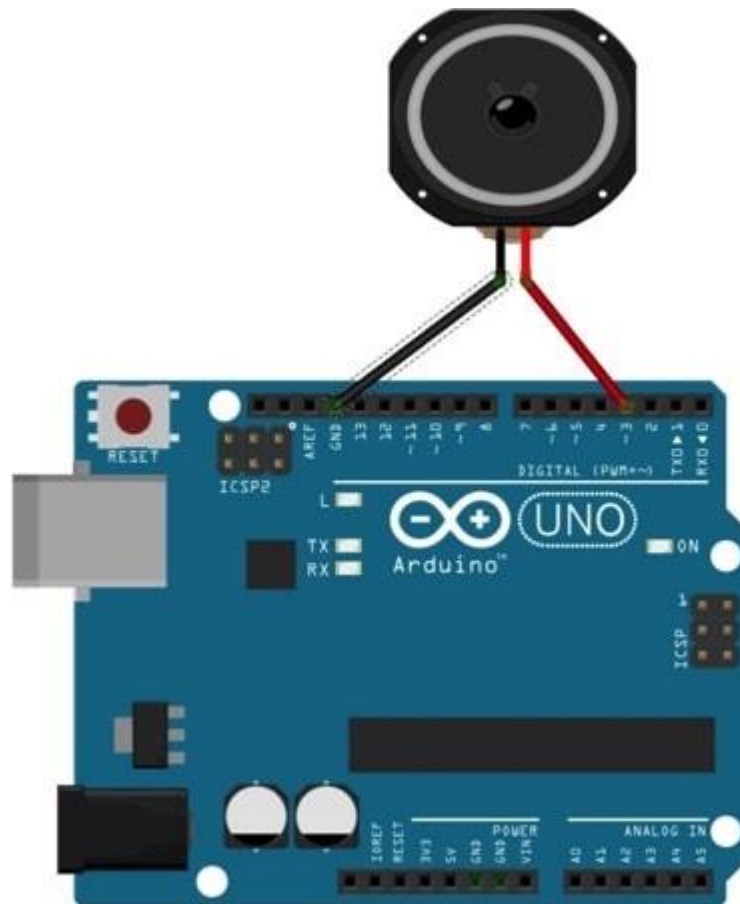




# Making Sound

- 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



# 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)



# Exercise : 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

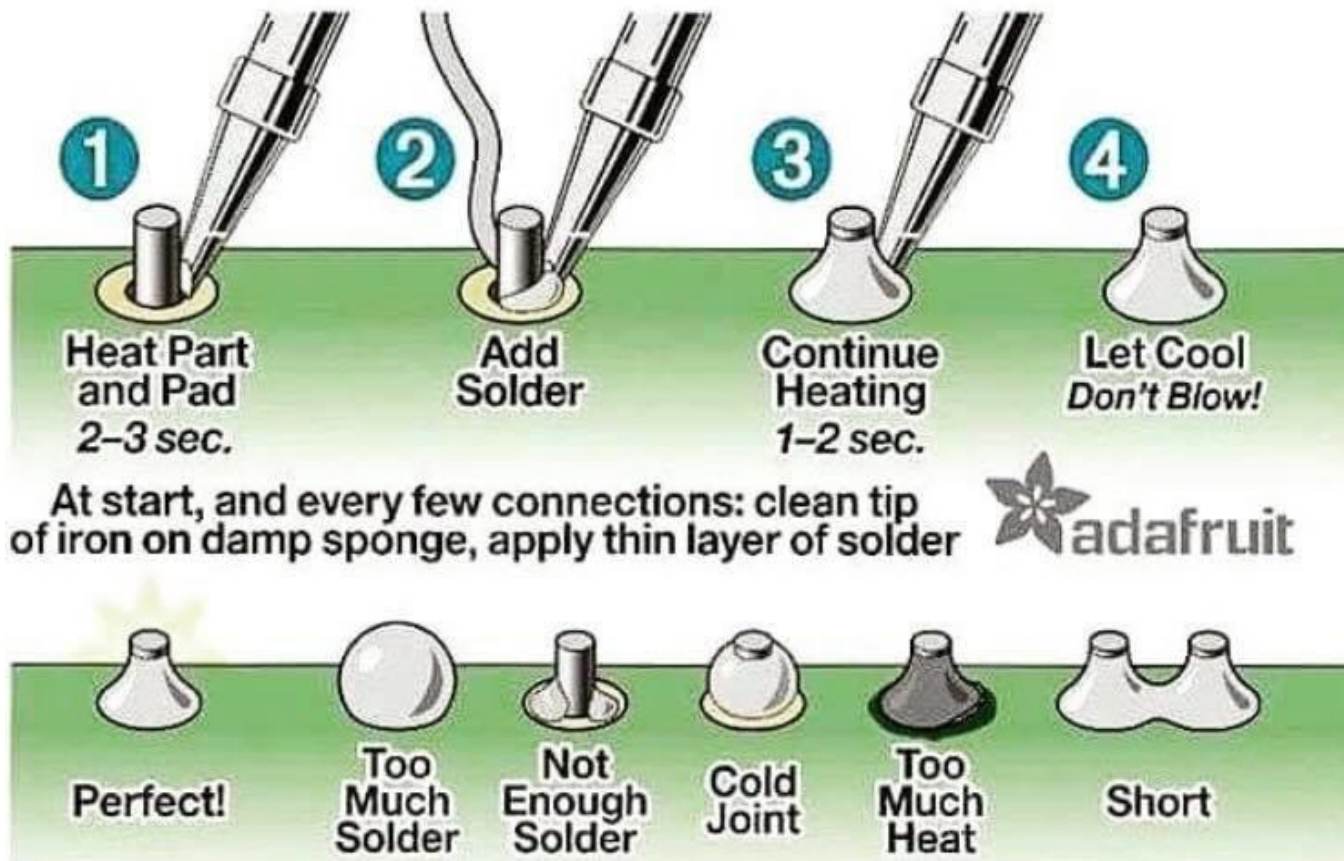


- ให้ต่อลำโพงตามรูป โดยต่อที่ขา 3
- โหลดไฟล์จาก Examples->Digital->toneMelody แล้วลองเล่นเพลง
- จากนั้นโหลดไฟล์จาก FB แล้วลองเล่นเพลง





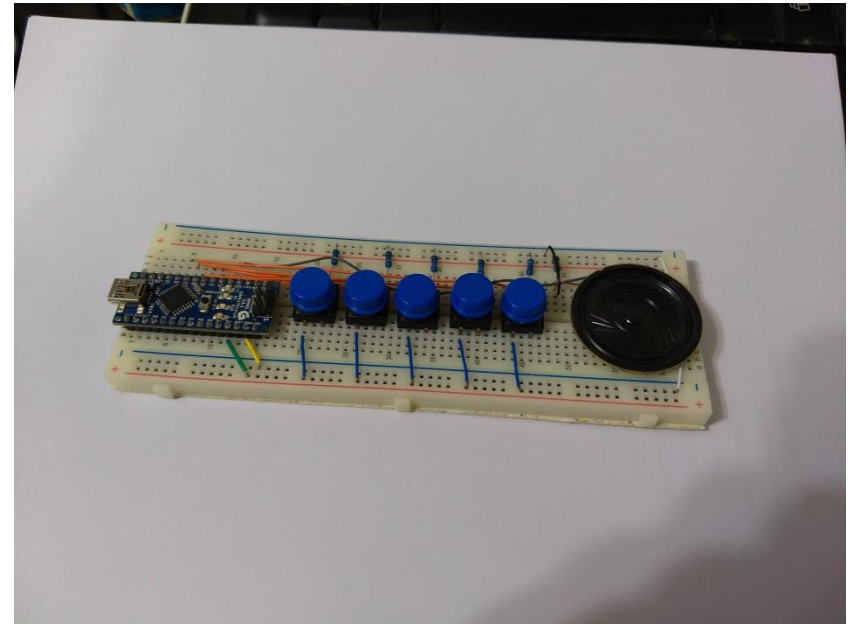
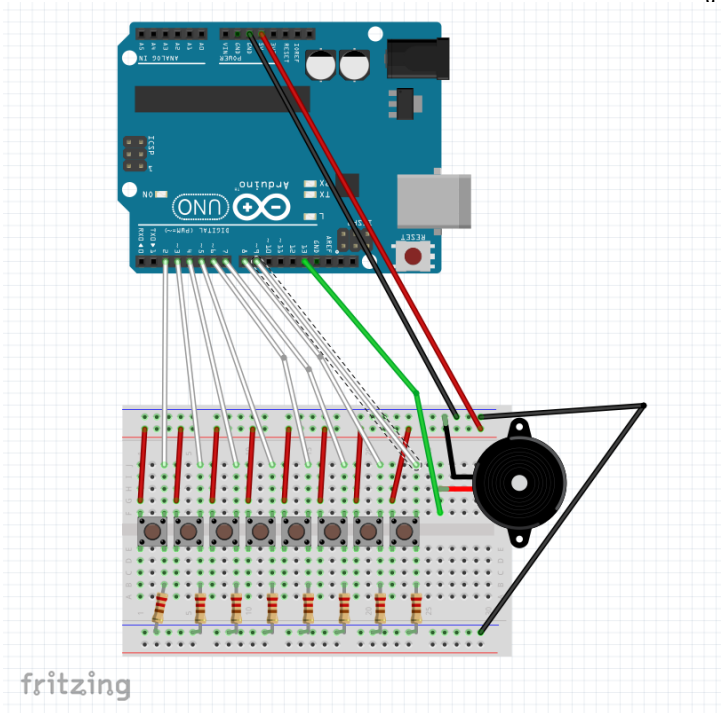
## Soldering Procedures



# Assignment #3 mini Piano



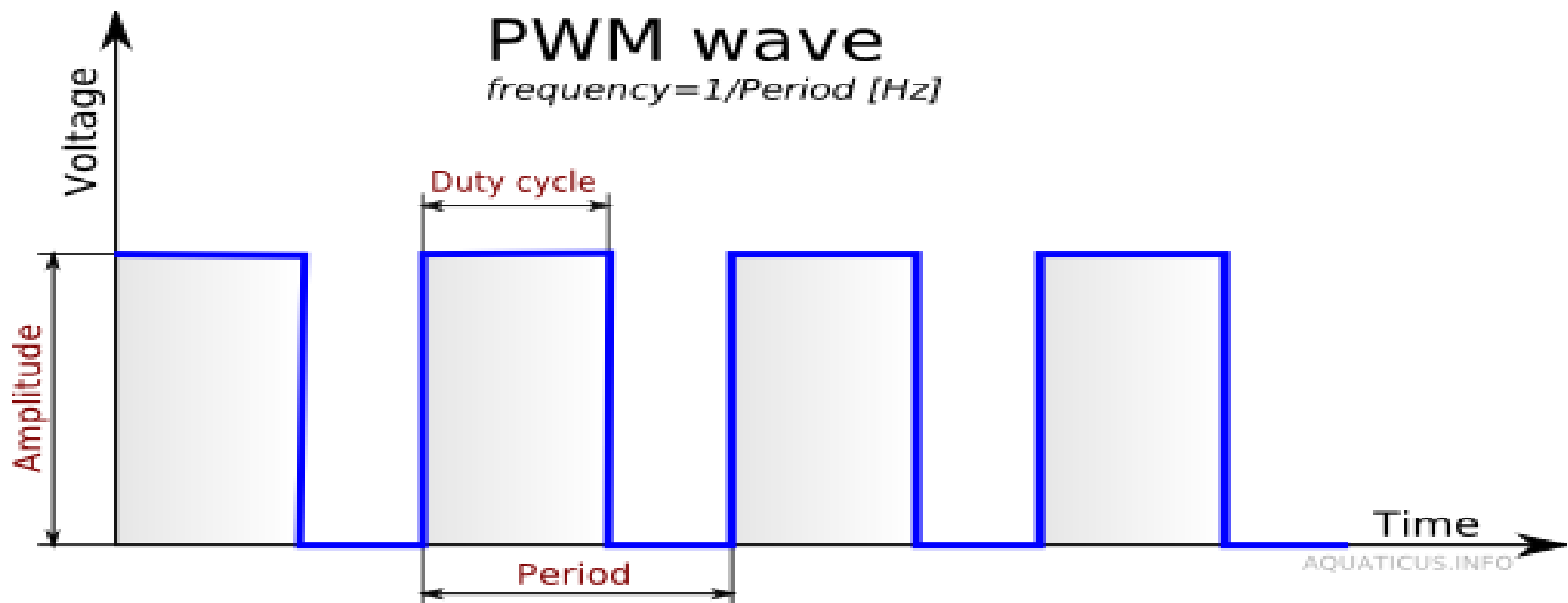
- ให้ต่อ Arduino กับ Switch > 5 ตัว และลำโพง
- ให้เขียนโปรแกรมเพื่อสร้าง mini Piano โดยสามารถเล่นเป็น piano และสามารถบันทึกการเล่น และสามารถเล่นกลับได้ (การส่งต้องเล่นเพลง 1 เพลง)
- หากลำโพงเสียงดังเกินไปให้ต่อ R อนุกรม





# Pulse Width Modulation (PWM)

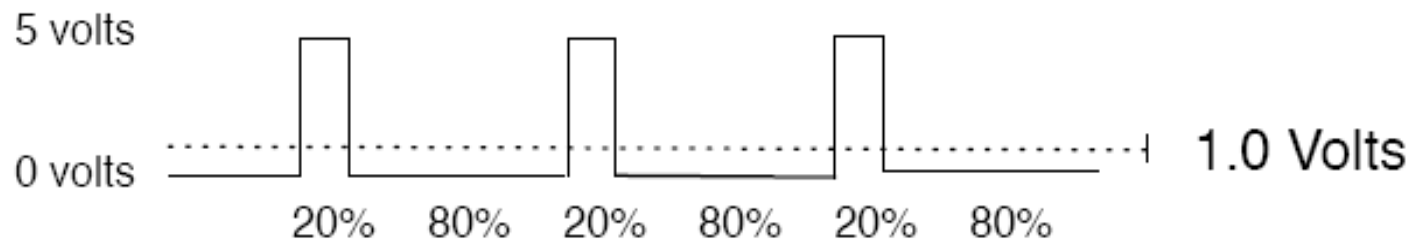
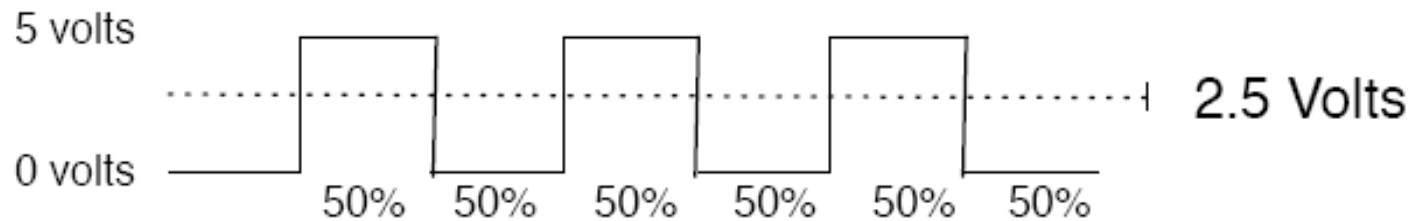
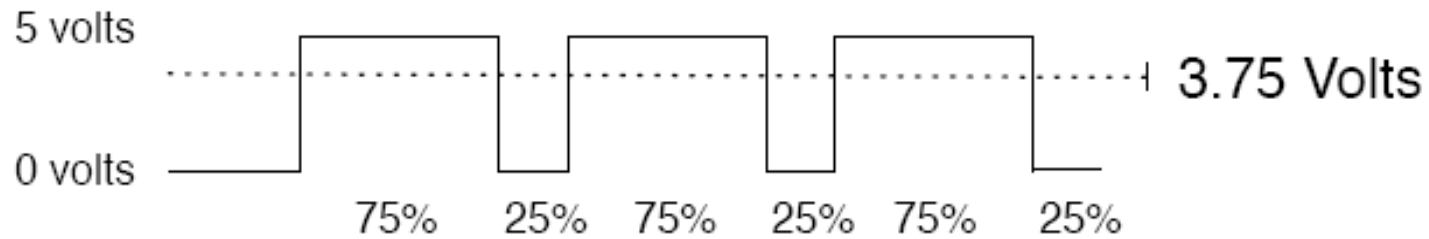
- Pulse Width Modulation หรือ PWM เป็นรูปแบบสัญญาณแบบหนึ่ง ที่ใช้ความกว้างของ Pulse เป็นส่วนสำคัญ



- Duty Cycle: on-time / period



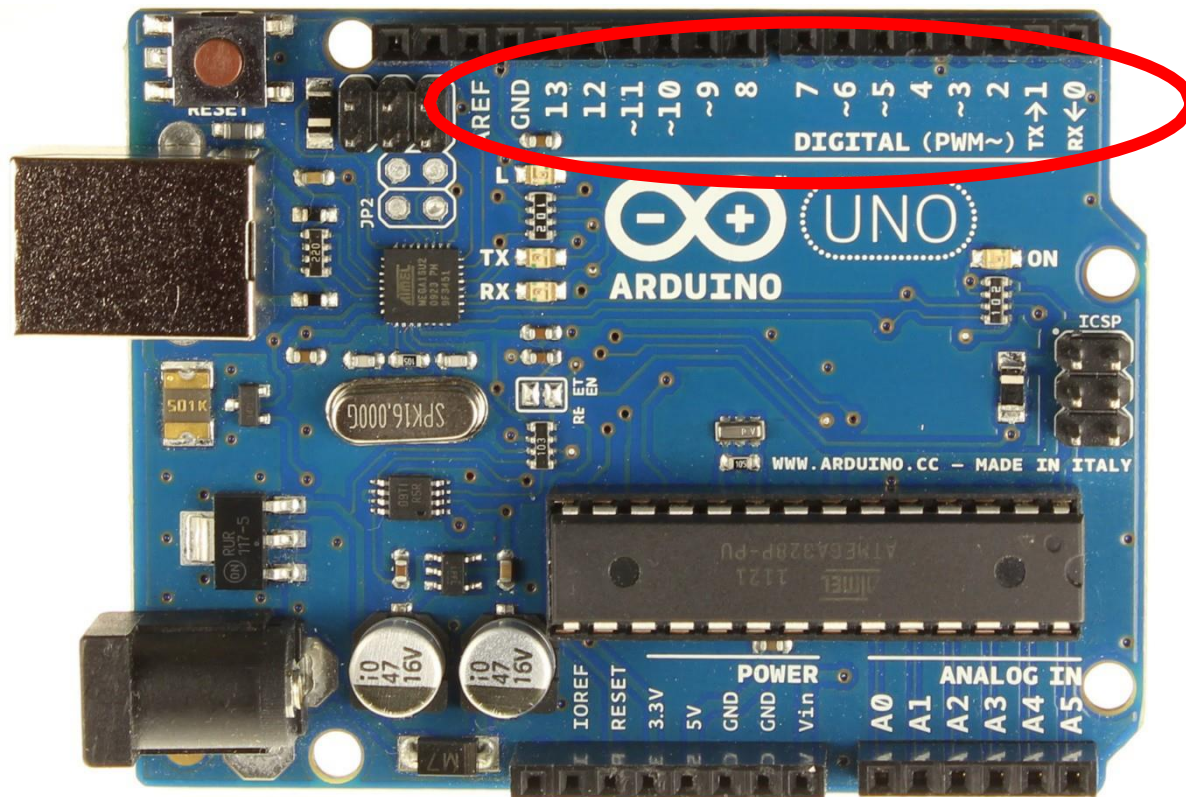
# PWM : Adjust duty cycle



# 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 เปอร์เซนต์ โดยให้เว้น 3 วินาที ระหว่างแต่ละ duty cycle ให้สังเกตความสว่างของ LED



# Joystick Shield



- Shield หมายถึงบอร์ดที่สร้างขึ้นมาเพื่อเสียบบนบอร์ด Arduino เพื่อทำงานด้านใดด้านหนึ่งโดยเฉพาะ
- Joystick Shield สร้างขึ้นมาเพื่อให้สามารถนำ Arduino ไปเล่นเกมได้สะดวกมากขึ้น
- Joystick Shield จะต่อกับ Pin ต่างๆ ดังนี้
- A (บน) ขา 2, C (ล่าง) ขา 4
- B (ขวา) ขา 3, D (ซ้าย) ขา 5
- E,F ขา 6,7
- ส่วนแท่ง Joystick นั้นประกอบด้วย 2 ส่วน  
ถ้ากดจะเป็น Switch ต่อกับขา 8  
ถ้าโยกจะต่อกับขา Analog 0 และ 1



# Exercise : Joystick Shield



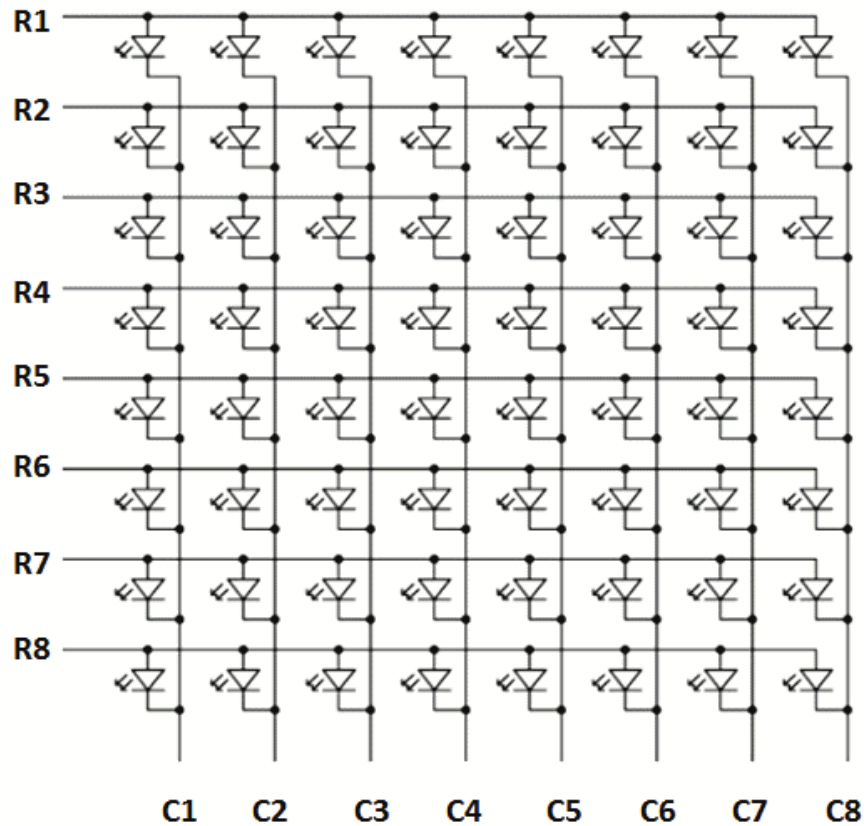
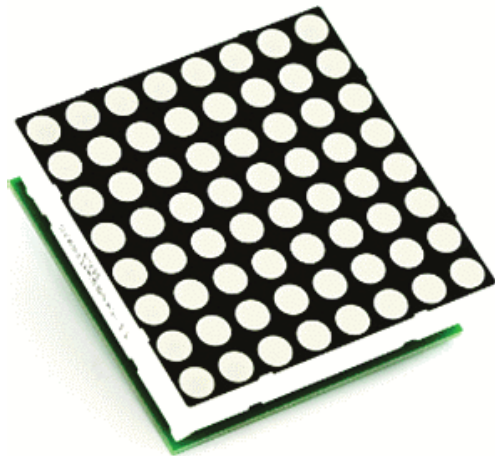
- โหลดโปรแกรมจาก FB แล้วทดลอง Run



# LED Dot Matrix



- เป็นอุปกรณ์ที่นำเอา LED จำนวนมากมาไว้ในชิ้นเดียวกัน มีวงจรตามรูป

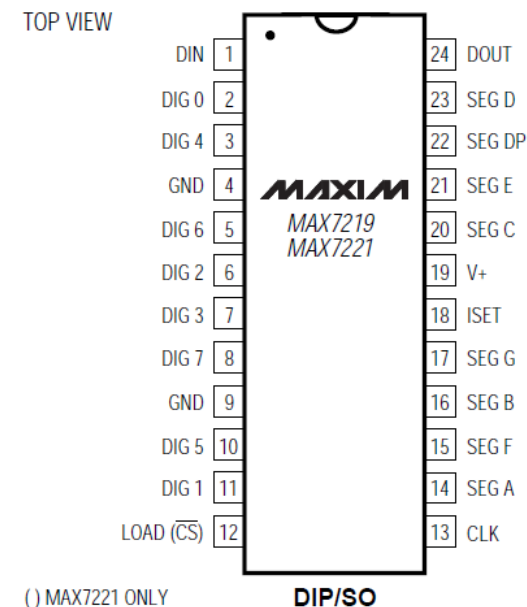


-

# การเชื่อมต่อ Arduino กับ LED Dot Matrix



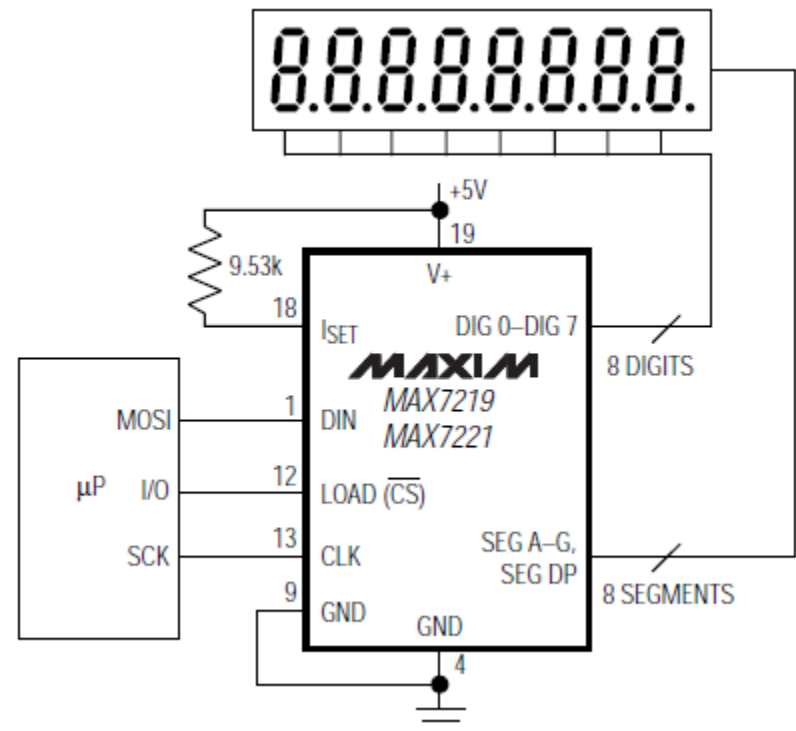
- ปัญหาข้างต้นเกิดขึ้นกับการต่อใช้งาน LED 7 Segment ที่มีหลายหลักเช่นกัน
- จึงได้มีผู้สร้างชิป IC สำหรับใช้ต่อกับ 7 Segment หลายหลัก โดยมีชื่อว่า MAX7219
- MAX7219 สามารถควบคุม LED 7 Segment ได้ 8 หลัก หรือใช้กับ LED Dot Matrix ขนาด 8x8 ได้



# การเชื่อมต่อ Arduino กับ LED Dot Matrix



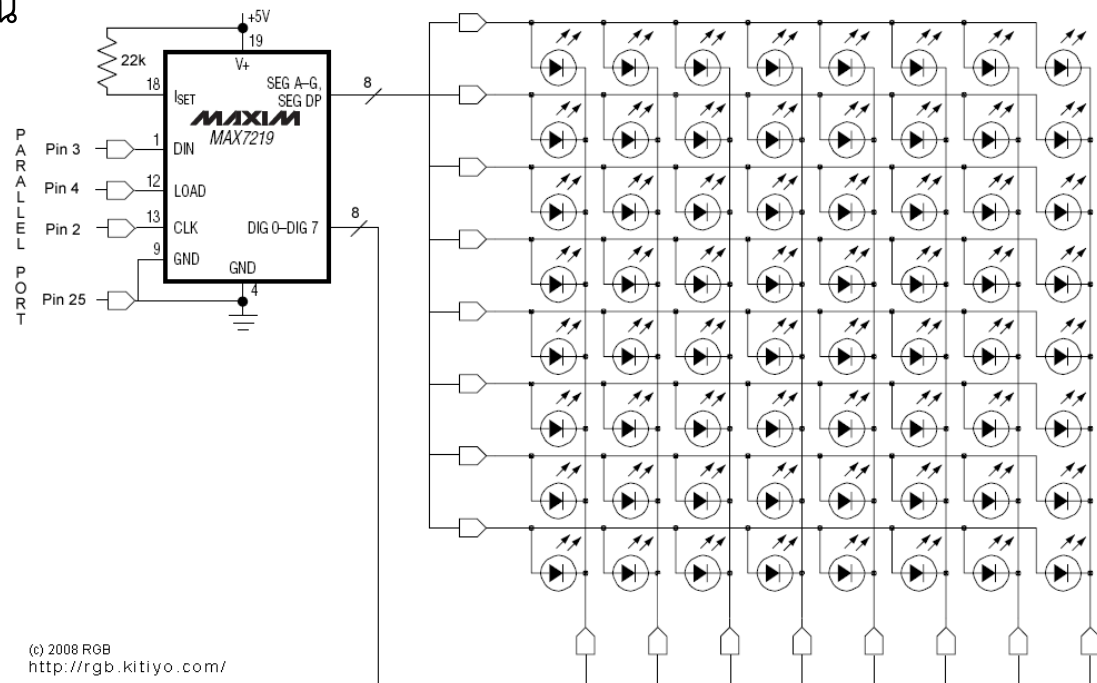
- ในการต่อ MAX7219 เข้ากับ LED 7 Segment จะต่อขา SEG A-G, DP เข้ากับขา a-g ของแต่ละหลัก และต่อ DIG 0-7 เข้ากับแต่ละหลัก (ตามรูป)
- ในการทำงาน MAX7219 จะส่งข้อมูลของแต่ละหลักไปที่ละครั้ง เช่น ส่งข้อมูลของหลักที่ 1 ในขณะที่ส่ง DIG 0 ออกไป ซึ่งจะทำให้หลักที่ 1 ติด จากนั้นทิ้งไว้ระยะหนึ่งแล้วจึงส่งหลักที่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



# 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)

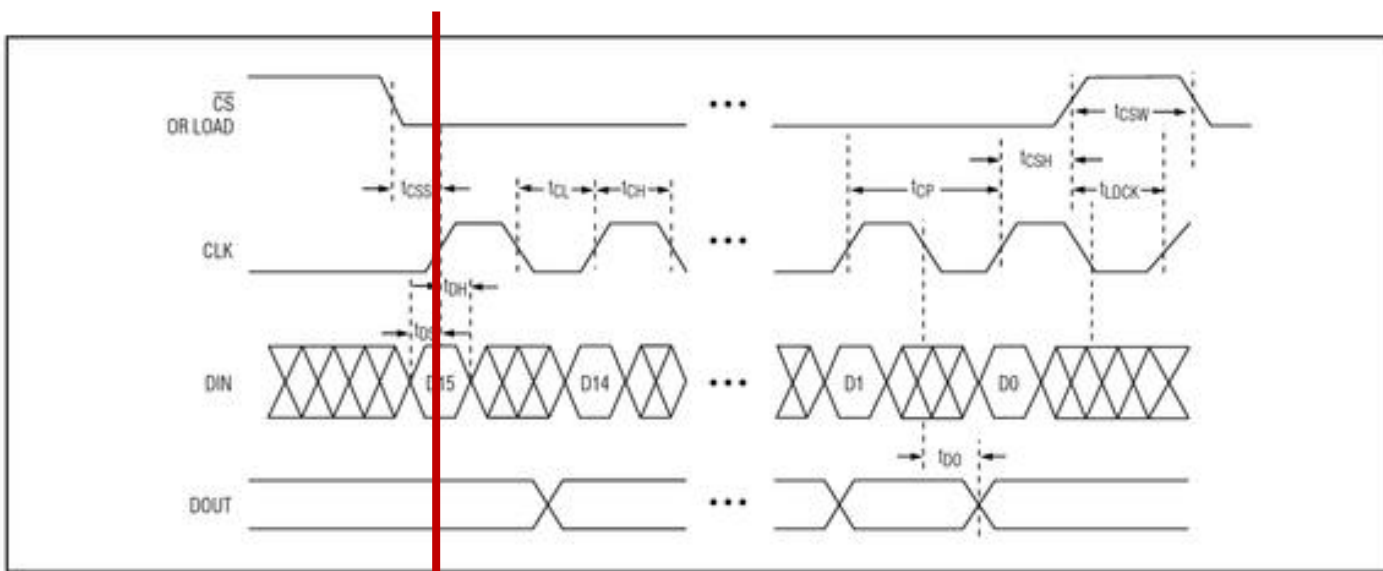


Figure 1. Timing Diagram

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    (0xA)
#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

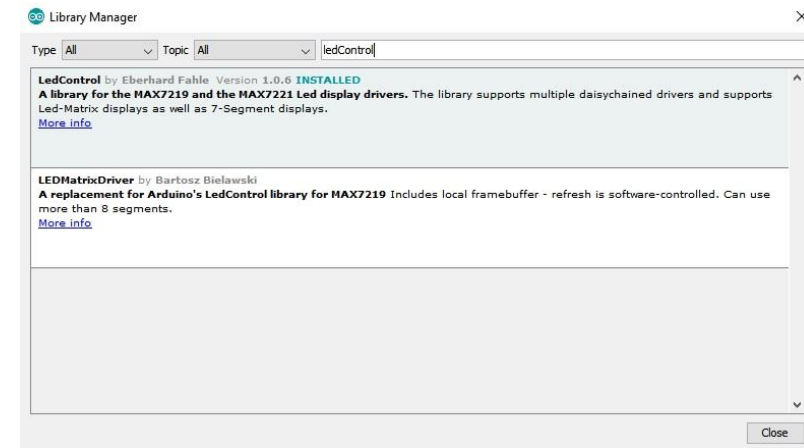


```
void loop() {  
  for (uint8_t i=0; i < 3; i++) {  
    flashing();  
  }  
  MAX7219_write_reg( REG_SHUTDOWN, 0x01 );    // normal operation  
  while (1) {  
    show_ce();  
  }  
}
```

# 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	⌘	⊥	⌞	β	±
x2	☹	‡	"	2	B	R	b	r	é	ŕ	ó	⌘	⌞	π	Γ	≥
x3	♥	!!	#	3	C	S	c	s	ä	ö	ū		⌞	μ	π	≤
x4	♦	¶	\$	4	D	T	d	t	ä	ö	ñ	⌞	—	⌞	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	s	ü	°	⌞	⌞	⌞	γ	≈
x8	☐	↑	(	8	H	X	h	x	ë	ÿ	¿	⌞	⌞	⌞	⌞	°
x9	○	↓	)	9	I	Y	i	y	ë	ÿ	¿	⌞	⌞	⌞	⌞	.
xA	☒	→	*	:	J	Z	j	z	ē	ū	⌞	⌞	⌞	⌞	⌞	.
xB	♂	←	+	;	K	[	k	{	ī	¼	⌞	⌞	⌞	⌞	⌞	√
xC	♀	⌞	,	<	L	\	l	!	ŕ	¼	⌞	⌞	⌞	⌞	⌞	⌞
xD	♂	⌞	-	=	M	]	m	}	ī	¼	⌞	⌞	⌞	⌞	⌞	⌞
xE	♂	♂	.	>	N	^	n	~	ŕ	¼	⌞	⌞	⌞	⌞	⌞	⌞
xF	⌘	♂	/	?	O	_	o	Δ	ā	f	>>	⌞	⌞	⌞	⌞	⌞

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



## Assignment #4 : game

- ให้สร้างโปรแกรมเกม space invader บน LED 8x8 หรือ เกมอื่น
- เริ่มต้นโดยการสร้างฐานปืนที่ด้านล่าง ควบคุมการเลื่อนซ้ายขวาโดย Joy Stick
- เมื่อกดปุ่มยิง ให้ยิงกระสุนออกจากฐานปืน โดยให้กระสุนวิ่งไปเรื่อยๆ
- สร้างยานอวกาศ โดยให้สุ่มแสดงผล และสุ่มการปล่อยระเบิด
- ถ้ายิงโดนระเบิดให้ระเบิดหายไป กำหนดจำนวนระเบิด ถ้ายิงได้ครบ ชนะ
- ถ้ามีระเบิดมาถึงด้านล่าง ก็ตาย (อาจกำหนดชีวิตก็ได้)
- ให้เขียนเอง ถ้าพบว่าลอกมาจากที่อื่น จะได้ 0
- การส่ง ให้ส่งไฟล์ source code และภาพถ่ายเกม



***For your attention***