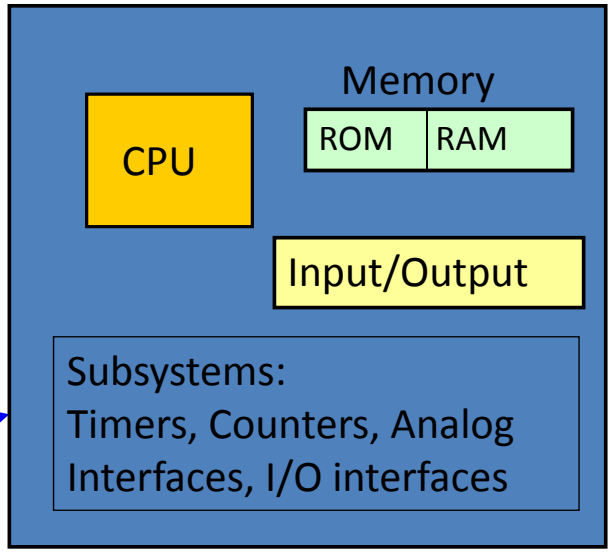
1.1 Basic Components of computer

- **องค์ประกอบของคอมพิวเตอร์**
  - หน่วย input/output
    - จอ monitor
    - Mouse, keyboard
    - Port usb, serial, parallel
  - หน่วยประมวลผล
    - Central Processor Unit
  - หน่วยความจำ
    - Harddisk
    - RAM

Microcontrollers



A single chip

- Microcontroller มาจากคำว่า “Micro” ที่แปลว่าเล็ก ๆ รวมกับคำว่า “Controller” ซึ่งหมายถึง ตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้น Microcontroller หมายถึง อุปกรณ์ควบคุมที่มีขนาดเล็ก ซึ่งตัว microcontroller เปรียบเสมือนกับเครื่องคอมพิวเตอร์ขนาดย่อมๆ เพราะว่า ประกอบด้วย CPU, memory และ Port ต่างๆ ที่รวมกันใน IC ตัวเดียว

โครงสร้างโดยทั่วไปของ Microcontroller

1. CPU (หน่วยประมวลผลกลาง) : ทำหน้าที่ประมวลผลทุกอย่างเหมือนกับ CPU ในคอมพิวเตอร์

2. Memory (หน่วยความจำ): มีหน้าที่ในการเก็บข้อมูลต่างๆ แบ่งออกเป็นสองส่วน คือ Program Memory และ Data Memory โดยที่ Program Memory นี้จะเก็บข้อมูลหลักของโปรแกรมเอาไว้ เสมือน Harddisk ของเครื่องคอมพิวเตอร์ แต่ Data Memory หรือ RAM เป็นที่พักข้อมูลชั่วคราว ถ้าไม่ได้จ่ายไฟเลี้ยง ข้อมูลก็จะหายไป

โครงสร้างโดยทั่วไปของ Microcontroller

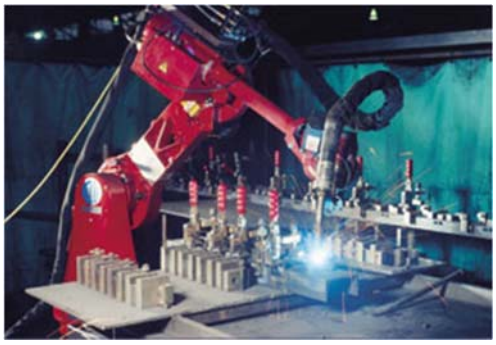
3. Port (ส่วนติดต่อภายนอก) : มี 2 แบบ คือ Input กับ Output

– Input ทำหน้าที่เพื่อ รับข้อมูลเข้ามา เช่นรับข้อมูลจาก Switch ,Sensor

– Output ทำหน้าที่เพื่อ ส่งข้อมูลออกไป เช่นส่งข้อมูลไปยัง หลอดไฟ LED หน้าจอ LCD หรือส่งไปยังคอมพิวเตอร์

4. Bus (ช่องทางเดินของสัญญาณ) : คือเส้นทางที่ใช้ในการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง CPU, Memory และ Port ซึ่งแบ่งเป็น Data Bus , Address Bus และ Control Bus

Where we can find microcontroller ?



1.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์

- MCS-51 (8-bit) ออกแบบโดย intel
- MCS-96 (16-bit) ออกแบบโดย intel
- PIC ออกแบบโดย Microchip Technology
- AVR ออกแบบโดย Atmel
- ARM ออกแบบโดย ARM Holdings
- 68HC11 ออกแบบโดย Motorola
- Rabbit 2000 ออกแบบโดย Rabbit Semiconductor

1.3 สถาปัตยกรรมของ AVR

- แบ่งออกเป็น 2 ตระกูลคือ 8-bit AVR และ 32-bit AVR
- ในที่นี้จะกล่าวถึงสถาปัตยกรรมของ AVR ขนาด 8 บิตเท่านั้น
- สถาปัตยกรรม AVR ออกแบบโดย ATMEL เมื่อปี 1996 เป็นซีพียูแบบ RISC (Reduced Instruction Set Computer)
- มีสถาปัตยกรรมการต่อหน่วยความจำแบบ Harvard ซึ่งแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ออกจากกันโดยเด็ดขาด
- ใช้หน่วยความจำแบบ Flash สำหรับเป็นหน่วยความจำโปรแกรม
- ใช้หน่วยความจำแบบ SRAM สำหรับหน่วยความจำข้อมูล
- นอกจากนี้ยังมีหน่วยความจำแบบ EEPROM ซึ่งสามารถเก็บข้อมูลเอาไว้ได้โดยไม่จำเป็นต้องมีไฟเลี้ยง

1.4 ประเภทการใช้งาน AVR ขนาด 8 บิต

- **tinyAVR** เป็นซีพียูในรุ่นเล็ก ซึ่งต้องการความเล็กกะทัดรัดของวงจร โดยเหมาะกับระบบควบคุมขนาดเล็กๆ ที่ต้องการหน่วยความจำและวงจรสนับสนุนไม่มากนัก ซีพียูในรุ่นนี้จะมีราคาถูกกว่ากลุ่มอื่น

- **megaAVR** จะมีชื่ออีกอย่างว่า ATmega โดยมีวงจรสนับสนุนภายในเพิ่มเติมตลอดจนเพิ่มขนาดของหน่วยความจำให้ใช้งานมากกว่าตระกูล Tiny เหมาะกับงานควบคุมทั่วๆ ไป

- **XMEGA** เพิ่มความละเอียดของวงจร A/D จากปกติมีความละเอียด10 บิตในรุ่นเล็กกว่าเป็น 12 บิต และวงจร DMA controller ซึ่งช่วยลดภาระของซีพียูในการควบคุมการรับส่งข้อมูลระหว่าง อุปกรณ์ I/O กับหน่วยความจำ

- **FPSLIC (AVR core with FPGA)** สำหรับงานที่ต้องการควบคุมที่ต้องการความยืดหยุ่นในขั้นตอนการออกแบบและพัฒนา โดยผู้ออกแบบสามารถออกแบบวงจรในระดับฮาร์ดแวร์เพิ่มเติมด้วยภาษาบรรยายฮาร์ดแวร์ (HDL: Hardware Description Language) เช่น ภาษา VHDLหรือภาษา Verilog และให้วงจรที่ออกแบบทำงานร่วมกับซีพียู AVR core

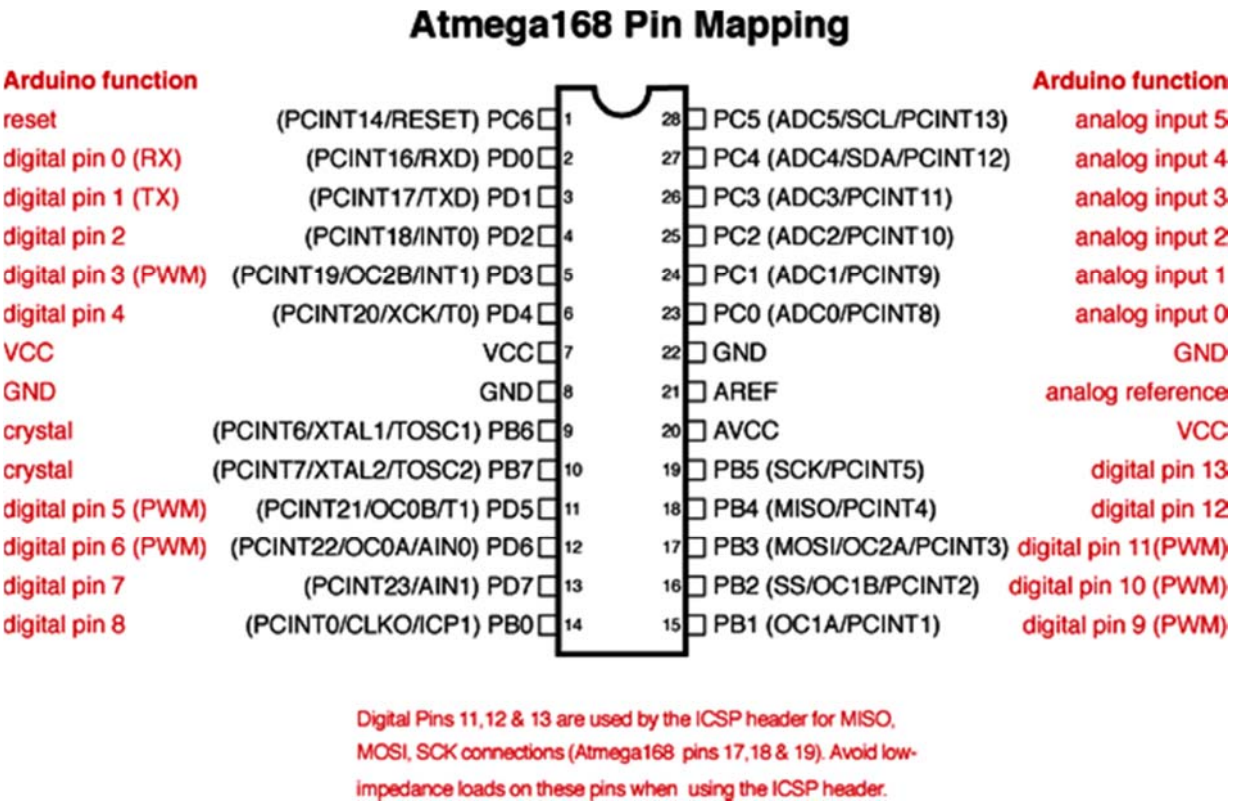
- **Application Specific AVR** เป็นซีพียูที่ออกแบบมาโดยเพิ่มวงจรควบคุมเฉพาะด้านเข้าไปซึ่งไม่ พบในซีพียูกลุ่มอื่นๆ เช่นวงจร USB controller หรือวงจร CAN bus เป็นต้น

AVR มีให้เลือกใช้งานหลายเบอร์ แต่ละเบอร์จะมีขนาด ราคา ความสามารถ และขนาด หน่วยความจำตลอดจนถึงวงจรสนับสนุนภายในที่แตกต่างกันออกไป

1.5.1 ATmega168

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 16 Kbyte
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 1 Kbyte
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด 512 byte
- สนับสนุนการเชื่อมต่อแบบ I^2C bus
- พอร์ตอินพุตเอาต์พุตจำนวน 20 ports
- วงจรสื่อสารอนุกรม
- วงจรนับ/จับเวลาขนาด 8 บิต จำนวน 2 ตัว และขนาด 16 บิตจำนวน
- สนับสนุนช่องสัญญาณสำหรับสร้าง PWM จำนวน 6 ช่องสัญญาณ
- วงจรแปลงอนาลอกเป็นดิจิตอลขนาด 10 บิตในตัว จำนวน 6 ช่อง
- ทำงานได้ตั้งแต่ย่านแรงดัน 1.8-5.5 Volts
- ความถี่ใช้งานสูงสุด 20 MHz

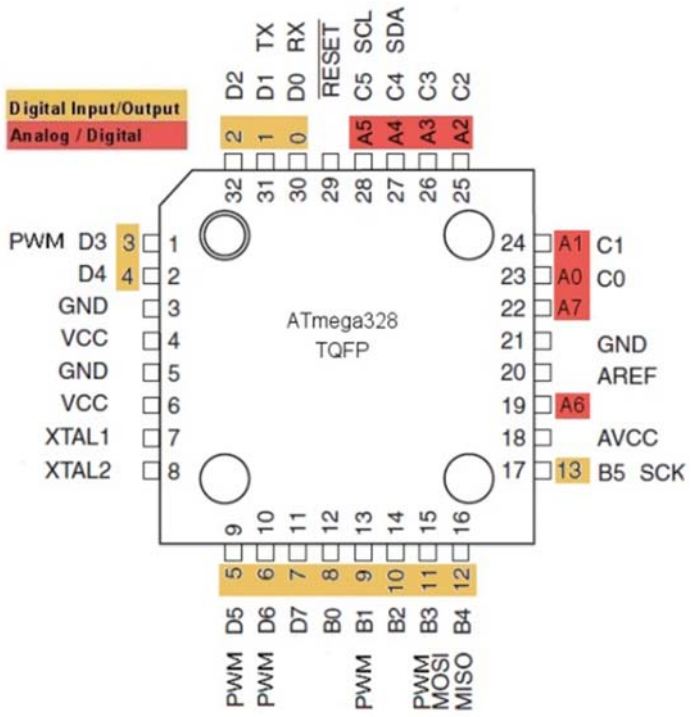
รายละเอียด ขา ต่าง ๆ ของ ATmega168



1.5.2 ATmega328

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 32 Kbyte
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 2 Kbyte
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด 1 Kbyte
- สนับสนุนการเชื่อมต่อแบบ I^2C bus
- พอร์ตอินพุตเอาต์พุตจำนวน 22 ports
- วงจรสื่อสารอนุกรม
- วงจรนับ/จับเวลาขนาด 8 บิต จำนวน 2 ตัว และขนาด 16 บิตจำนวน 1 ตัว
- สนับสนุนช่องสัญญาณสำหรับสร้าง PWM จำนวน 6 ช่องสัญญาณ
- วงจรแปลงอนาลอกเป็นดิจิตอลขนาด 10 บิตในตัว จำนวน 8 ช่อง
- ทำงานได้ตั้งแต่ย่านแรงดัน 1.8-5.5 Volts
- ความถี่ใช้งานสูงสุด 20 MHz

รายละเอียด ขา ต่าง ๆ ของ ATmega328



1.6 Platform

Platform หมายถึง การทำงานร่วมกันของ Hardware และ Software สำหรับการทำงานของ Application

Hardware

- หมายถึง สถาปัตยกรรมคอมพิวเตอร์ หรือ สถาปัตยกรรมหน่วยประมวลผล ตัวอย่างเช่น CPU ที่ใช้สถาปัตยกรรม x86 หรือ x86-64 เป็นต้น

Software

- หมายถึง ระบบปฏิบัติการ หรือสภาพแวดล้อมทางโปรแกรมมิ่ง

  ตัวอย่างเช่น Microsoft Windows, Mac OS X x86 , Mac OS X PowerPC, Linux x86 , Java Platform

1.7 What is Arduino ? [1]

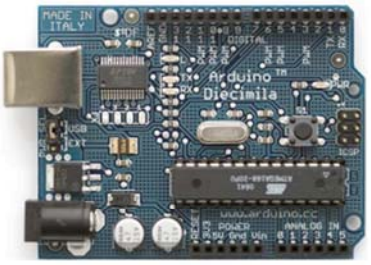
- Arduino เป็นโครงการพัฒนาไมโครคอนโทรลเลอร์ของตระกูล AVR แบบ Open Source โดยเป็นชื่อเรียกของ platform ซึ่งประกอบไปด้วย
  - Development Board ที่ใช้ microcontroller ตระกูล AVR
  - IDE เพื่อใช้ในการพัฒนาโปรแกรม โดย IDE พัฒนาโดยภาษา Java ส่งผลให้สามารถทำงานได้ทุก OS เช่น Linux, MAC OS, Windows

    และ IDE มี library มาตรฐาน ในการอ้างอิงกับบอร์ด Arduino จำนวนมาก ทำให้สะดวกในการพัฒนาโปรแกรม
- โดย AVR ที่เอามาใช้นั้น จะต้องมีการติดตั้ง Firmware ไว้แล้ว โดยหน้าที่หลักของ Firmware คือ การรับโปรแกรมที่ Compile แล้ว จาก IDE มาเขียนไว้ที่ตัวมันเองเพื่อทำงานตามโปรแกรมที่เขียน

What is Arduino ? [2]

The word "Arduino" can mean 3 things

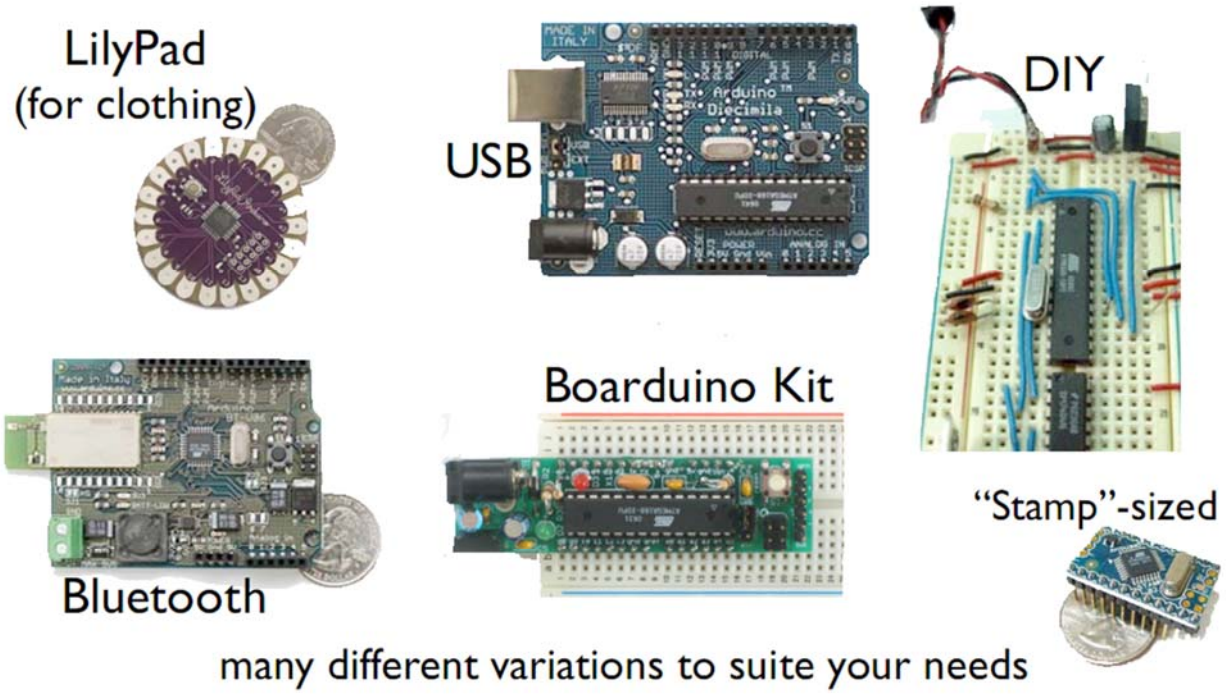
A physical piece of hardware



A programming environment

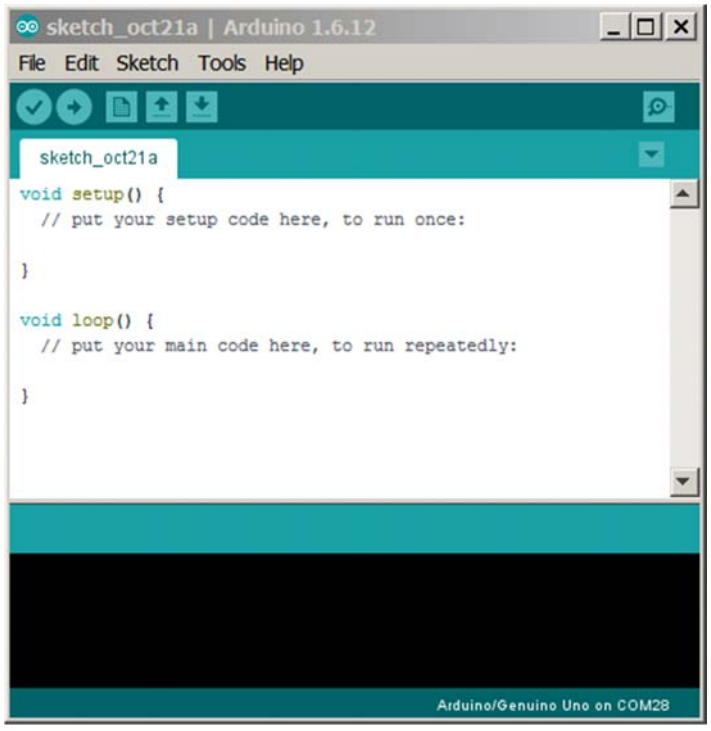
A community & philosophy

1.7.1 Arduino Hardware Variety



Openness has its advantages, many different varieties.
Anyone can build an Arduino work-alike in any form-factor they want.
Product images from Sparkfun.com and Adafruit.com

1.7.2. Arduino Software



- Like a text editor
- View/write/edit sketches
- But then you program them into hardware

1.7.3 ข้อดี-ข้อเสีย Arduino

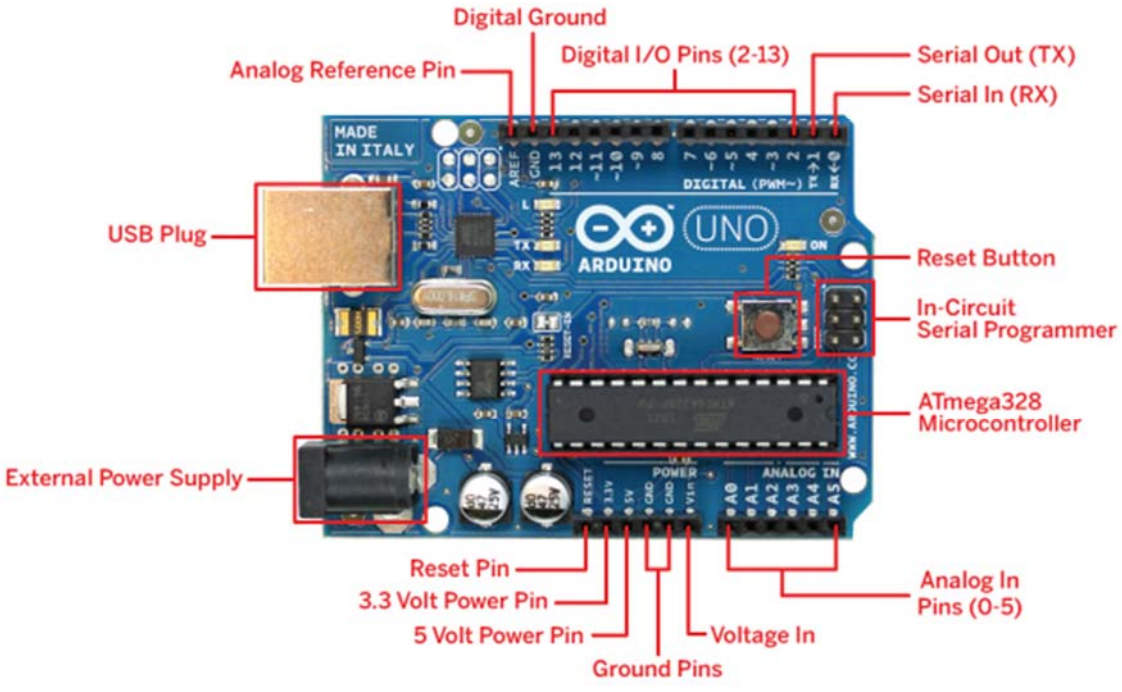
ข้อดี

- พัฒนาโปรแกรมด้วยภาษา c++ จึงสามารถใช้คุณสมบัติ OOP ได้
- ไม่ต้องใช้อุปกรณ์ ISP ในการโหลดโปรแกรมไปยัง microcontroller
- โครงสร้างการเขียนเข้าใจง่าย ไม่ซับซ้อน
- ต้นทุนมีราคาถูก
- เป็นโครงการ opensource จึงทำให้มีผู้ให้ความสนใจมาก และมี Library ให้ใช้งานเป็นจำนวนมาก

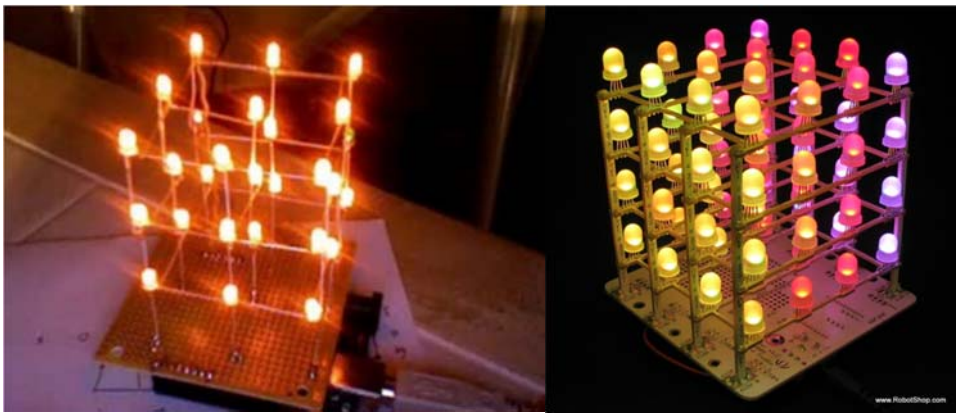
ข้อเสีย

- เนื่องจากพัฒนาโปรแกรมด้วยภาษา c++ จึงใช้ทรัพยากรมากพอสมควร

Arduino Board



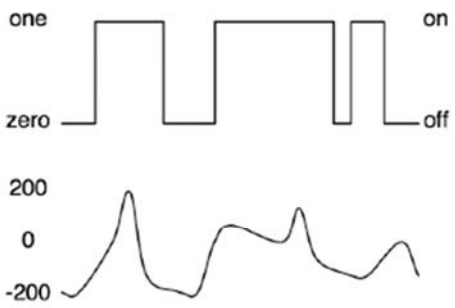
2. การส่งข้อมูลออกพอร์ต



arduino cube

Digital? Analog?

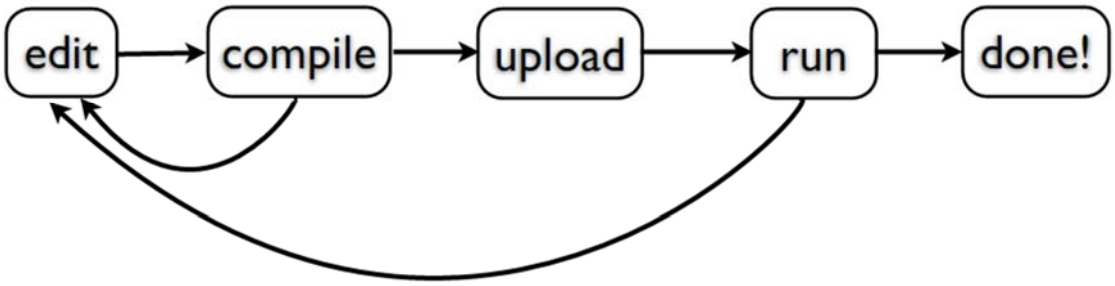
- Digital – only has two values: on/off
- Analog – has many (infinite) values



- Computers don't really do analog
- So they fake it, with *quantization*

2.1 ขั้นตอนการพัฒนาโปรแกรมด้วย Arduino (Development Circle)

- Make as many changes as you want
- Not like most web programming: edit → run
- Edit → compile → upload → run



2.2 Arduino Language

Language is standard C (but made easy)

core function

Digital I/O
- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O
- analogReference()
- analogRead()
- analogWrite() - *PWM*

Communication
- Serial

Advanced I/O
- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Time
- millis()
- micros()
- delay()
- delayMicroseconds()

Math
- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

Trigonometry
- sin()
- cos()
- tan()

Random Numbers
- randomSeed()
- random()

2.3 Some useful function

- **`pinMode()`** – set a pin as input or output
- **`digitalWrite()`** – set a digital pin high/low
- **`digitalRead()`** – read a digital pin's state
- **`analogRead()`** – read an analog pin
- **`analogWrite()`** – write an "analog" value
- **`delay()`** – wait an amount of time
- **`millis()`** – get the current time

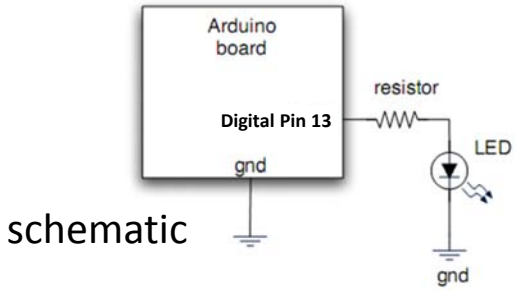
2.4 การเขียนโปรแกรมบน Arduino

การเขียนโปรแกรมบน Arduino จะคล้ายกับการเขียนโปรแกรม C/C++ ไฟล์โปรแกรม ที่ทำการเขียนจะเรียกว่า sketch files มีส่วนประกอบสามส่วน คือ

- Declare variables at top
- Initialize
  - `setup()` – run once at beginning, set pins
- Running
  - `loop()` – run repeatedly, after `setup()`

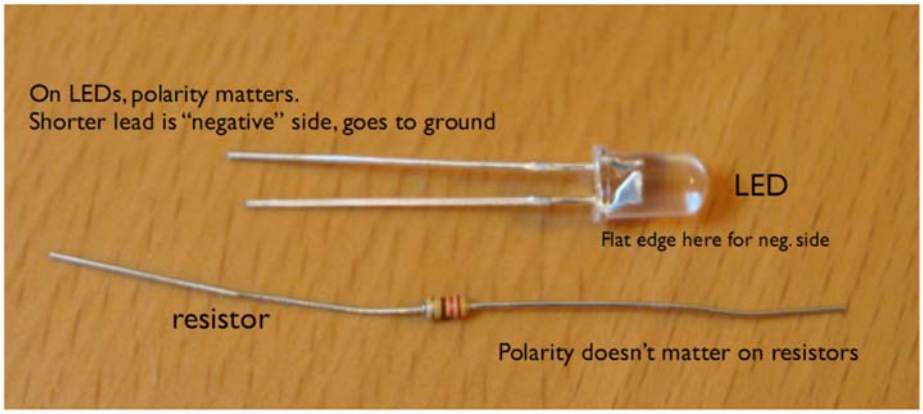
2.5 การส่งค่าออกพอร์ต แบบดิจิตอล

การทดลองที่ 2.1. LED Blink : Hello world of Microcontroller



schematic

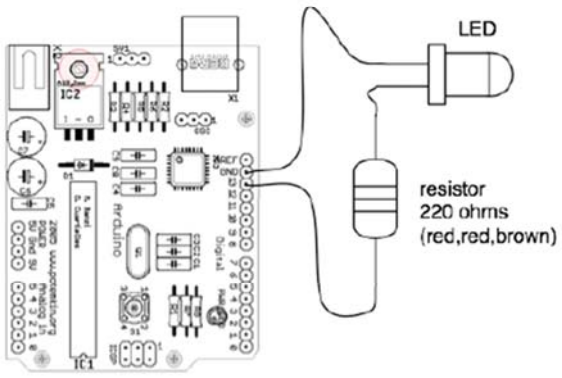
To turn on LED use
`digitalWrite(13,HIGH)`



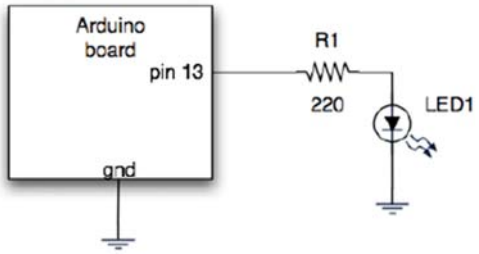
รายละเอียดขา อยู่หน้า 19

Blinky LED circuit

"hello world" of microcontrollers



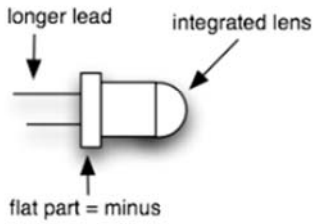
wiring diagram



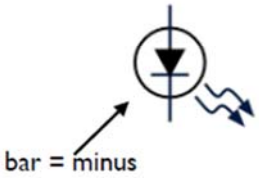
schematic

LEDs

- LED = Light-Emitting Diode
  - electricity only flows one way in a diode
- Needs a "current limiting" resistor, or burns out

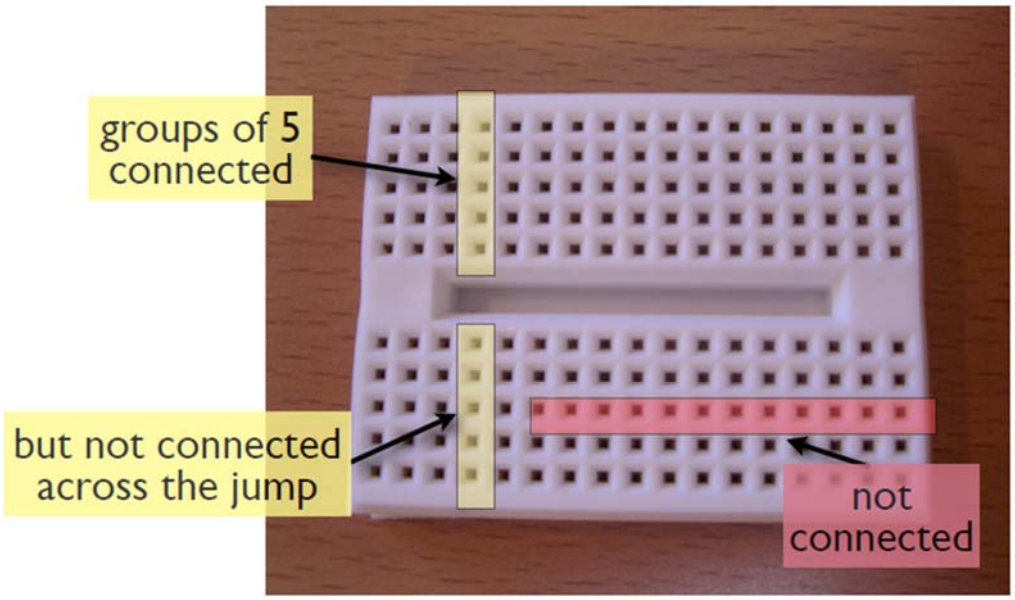


physical characteristics



schematic symbol

Solderless Breadboards



โปรแกรม LED Blink แบบที่ 1

lab01

```
void setup()                    // Setup function
{
  pinMode(13,OUTPUT);           // Digital Pin13 = Output pin
}

void loop()                     // Main function
{
  digitalWrite(13,HIGH);        // Digital Pin13 = High
  delay(250);                   // wait 250 ms
  digitalWrite(13,LOW);         // Digital Pin13 = Low
  delay(250);                   // wait 250 ms
}
```

Done compiling.

โปรแกรม LED Blink แบบที่ 2

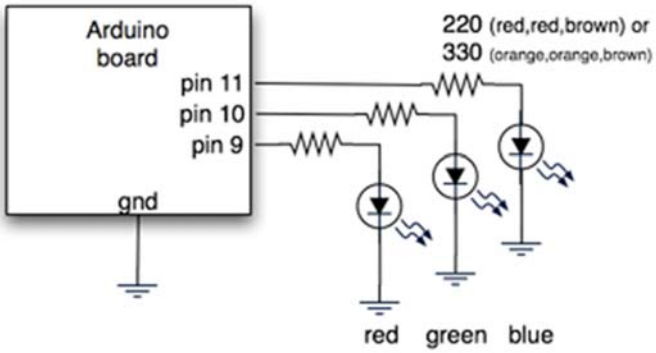
lab02

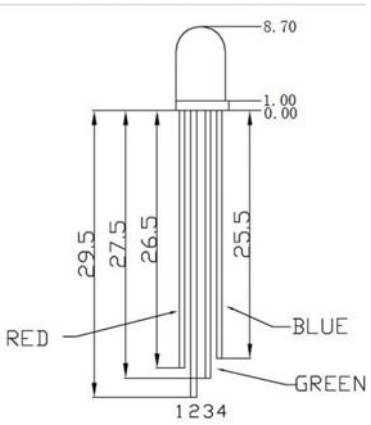
```
int  ledPIN = 13;               // ledPIN = DigitalPin 13
void setup()                    // Setup function
{
  pinMode(ledPIN,OUTPUT);       // ledPIN = Output pin
}

void loop()                     // Main function
{
  digitalWrite(ledPIN,HIGH);    // ledPIN = High
  delay(250);                   // wait 250 ms
  digitalWrite(ledPIN,LOW);     // ledPIN = Low
  delay(250);                   // wait 250 ms
}
```

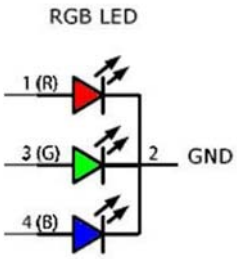
Done compiling.

แบบฝึกหัดที่ 2.1 โปรแกรมควบคุม Color LED

จงเขียนโปรแกรมแสดงสี 8 สี โดยเว้นช่วงสีละ 1 วินาที
กำหนดสีเอง ตามใจชอบ

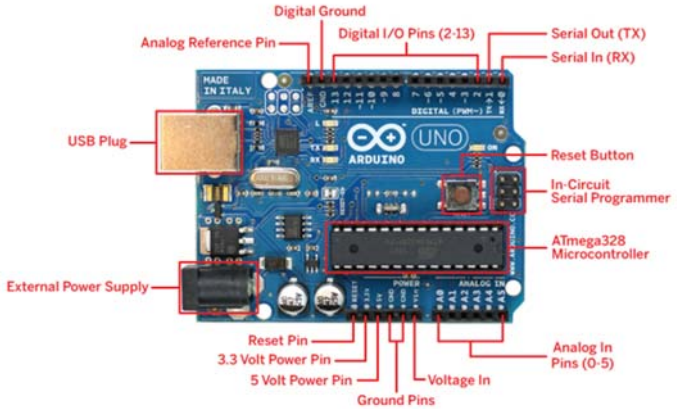




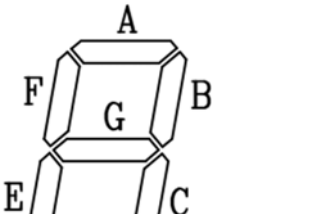


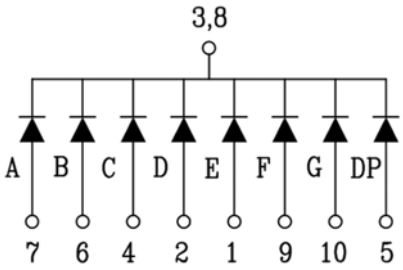
แบบฝึกหัดที่ 2.2 โปรแกรมควบคุม 7-segment

- ดัดแปลงวงจร จากการทดลองที่ 1
  โดยแทน **LED** ด้วย **7-segment**



- จงเขียนโปรแกรมแสดง 0-9 โดยเว้น
  ช่วงละ 1 วินาที





การจัดขาของ 7-segment

More details of 7-segments

http://commandronestore.com/learning/7segment.php