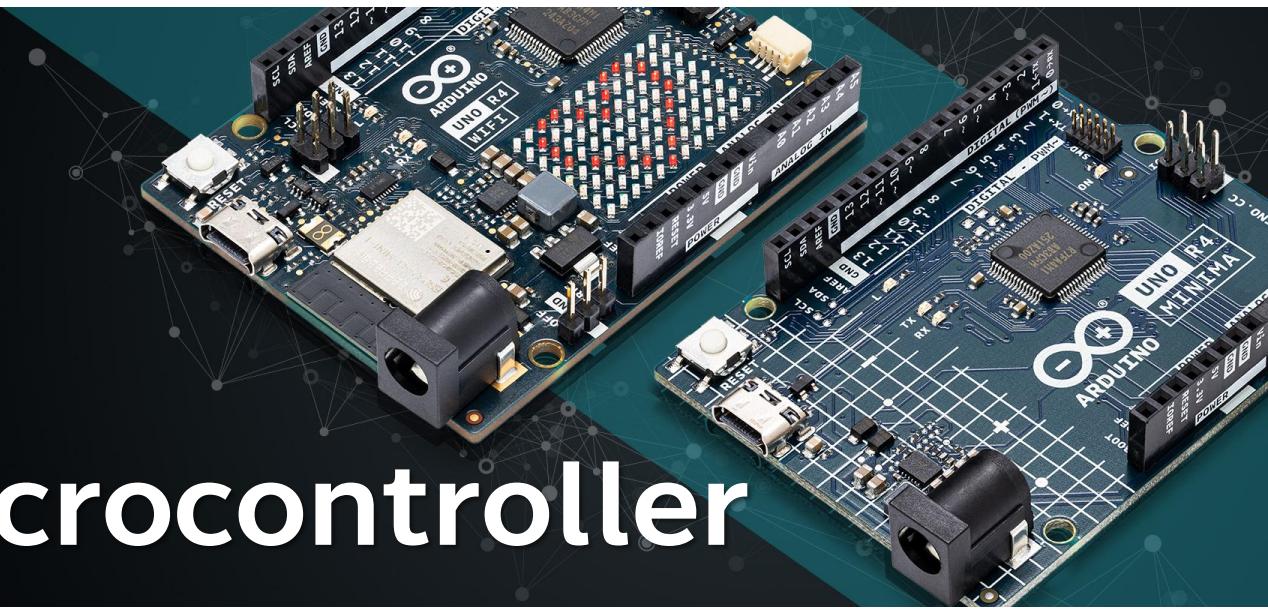




Physical Computing



ครั้งที่ 13. Microcontroller



1. Introduction



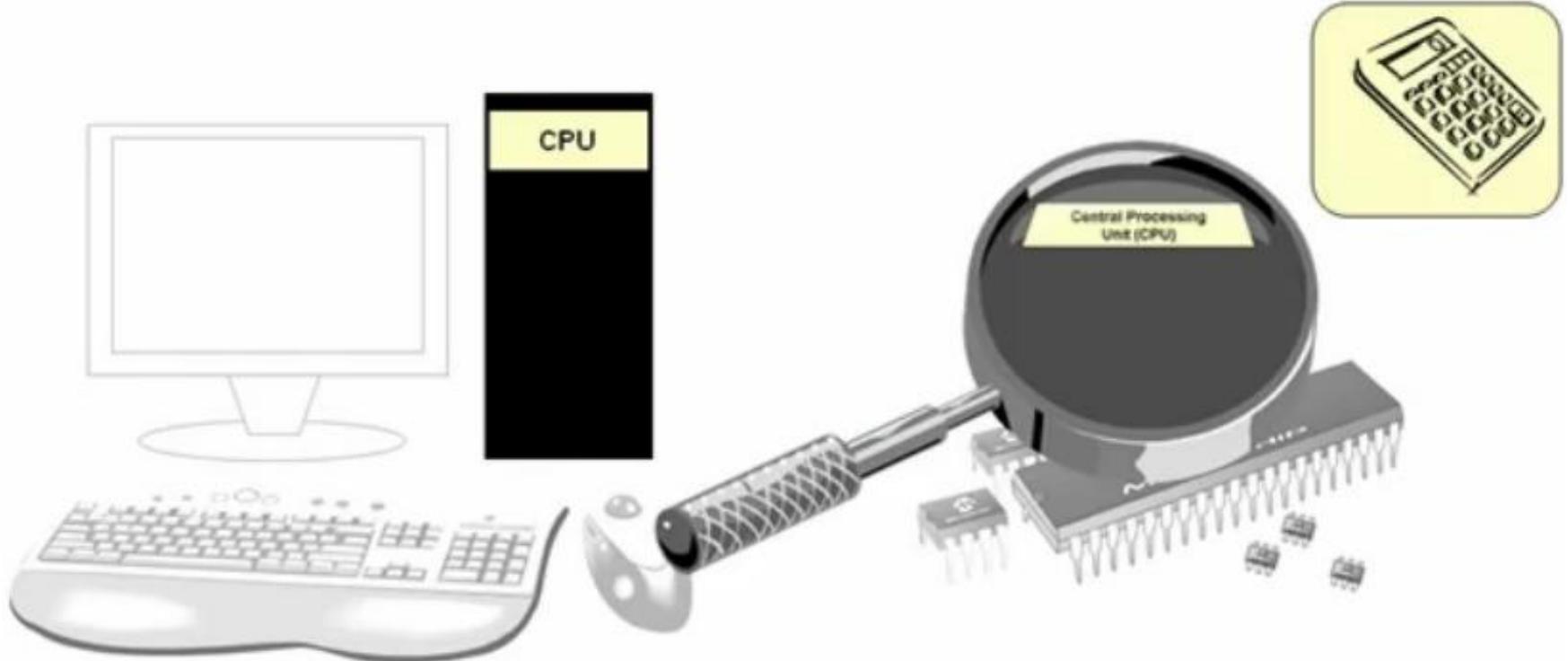
Best Microcontrollers
for All Uses

1. What is a Microcontroller?



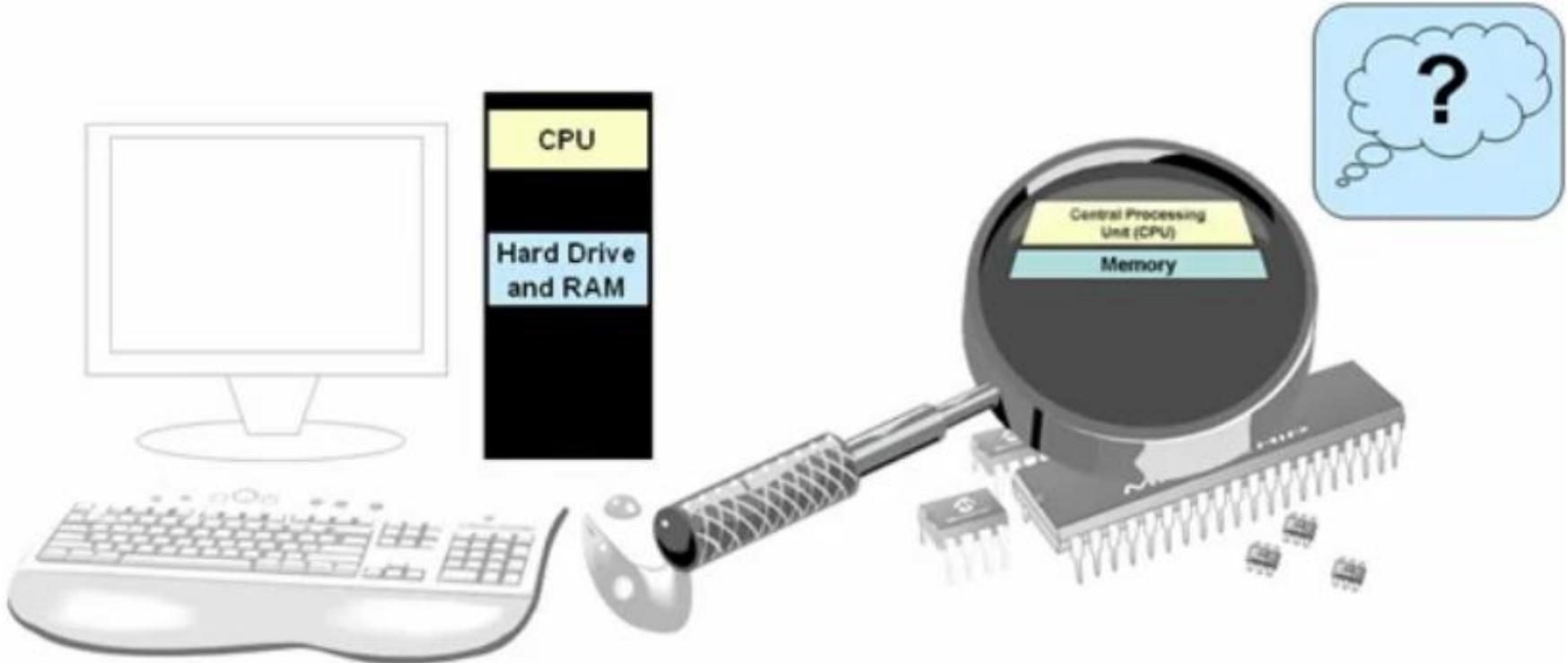
- A microcontroller (**μC**, **uC** or **MCU**) is a single integrated circuit that contains the four major parts of every computer system
- A device optimized for control applications

What is in a Microcontroller?



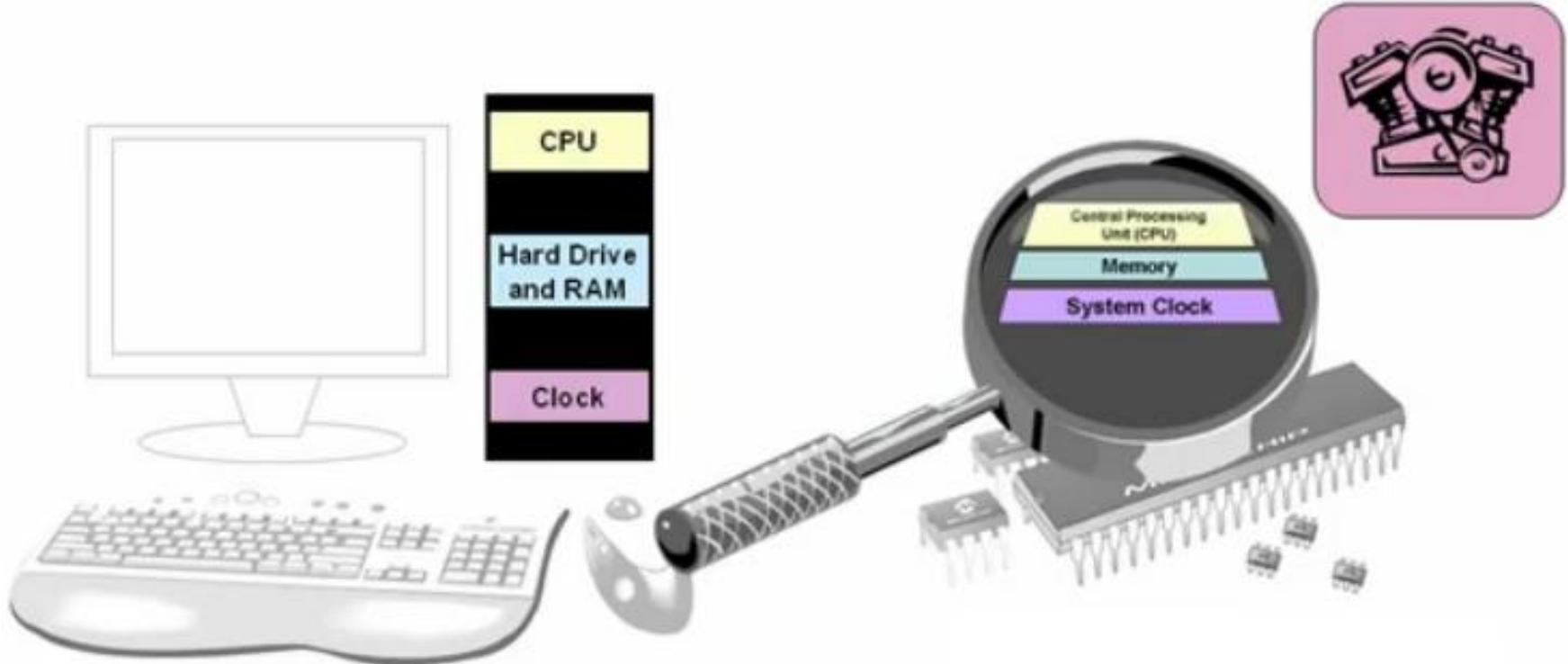
- All computers have a CPU (central processing unit) that executes programs

What is in a Microcontroller?



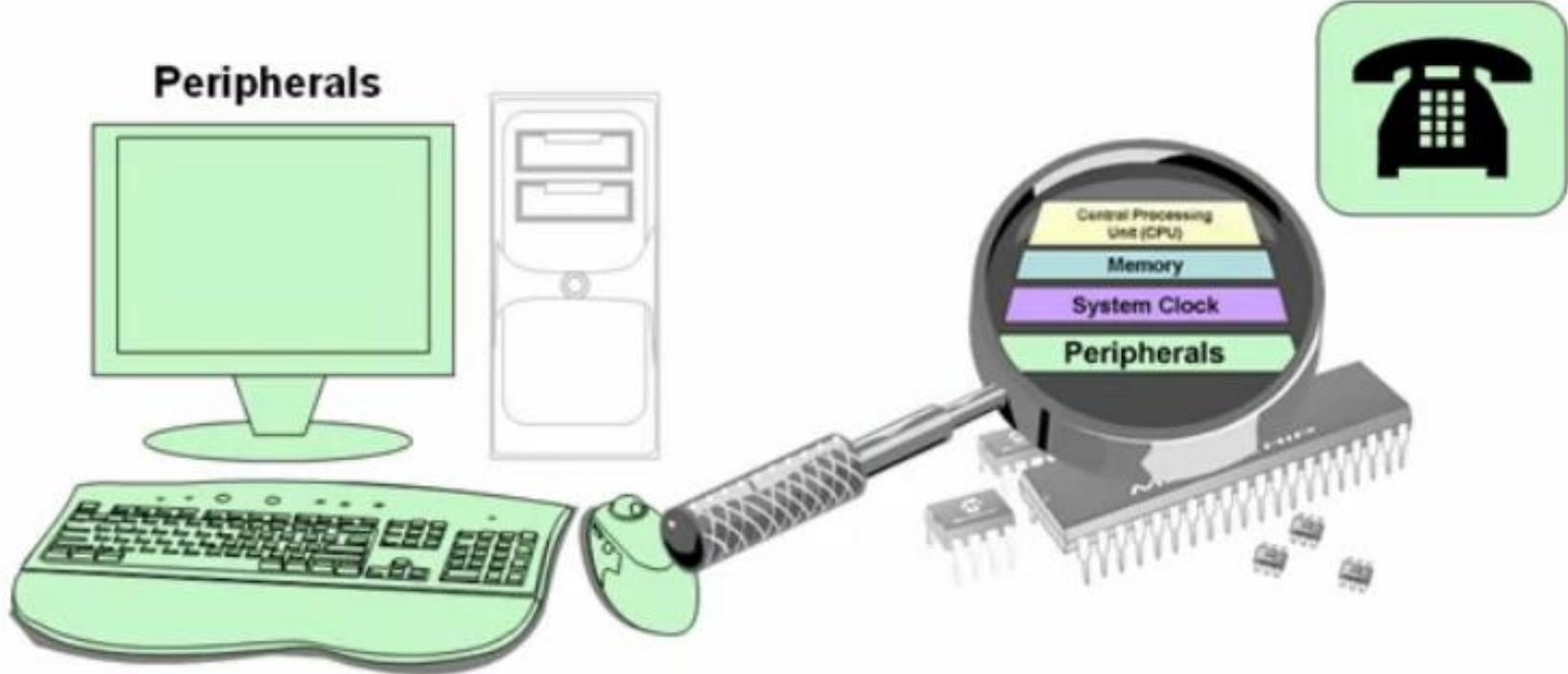
- A computer has some type of memory where it can store variables and instructions or steps
- Memory may be RAM, ROM, (E)EPROM, Flash

What is in a Microcontroller?



- Computers have a clock or oscillator that determine the speed of program execution

What is in a Microcontroller?



- Computers have various input and output (I/O) capabilities to support various peripherals.

What is in a Microcontroller?



Central Processing Unit



Memory



System Clock (Oscillator)



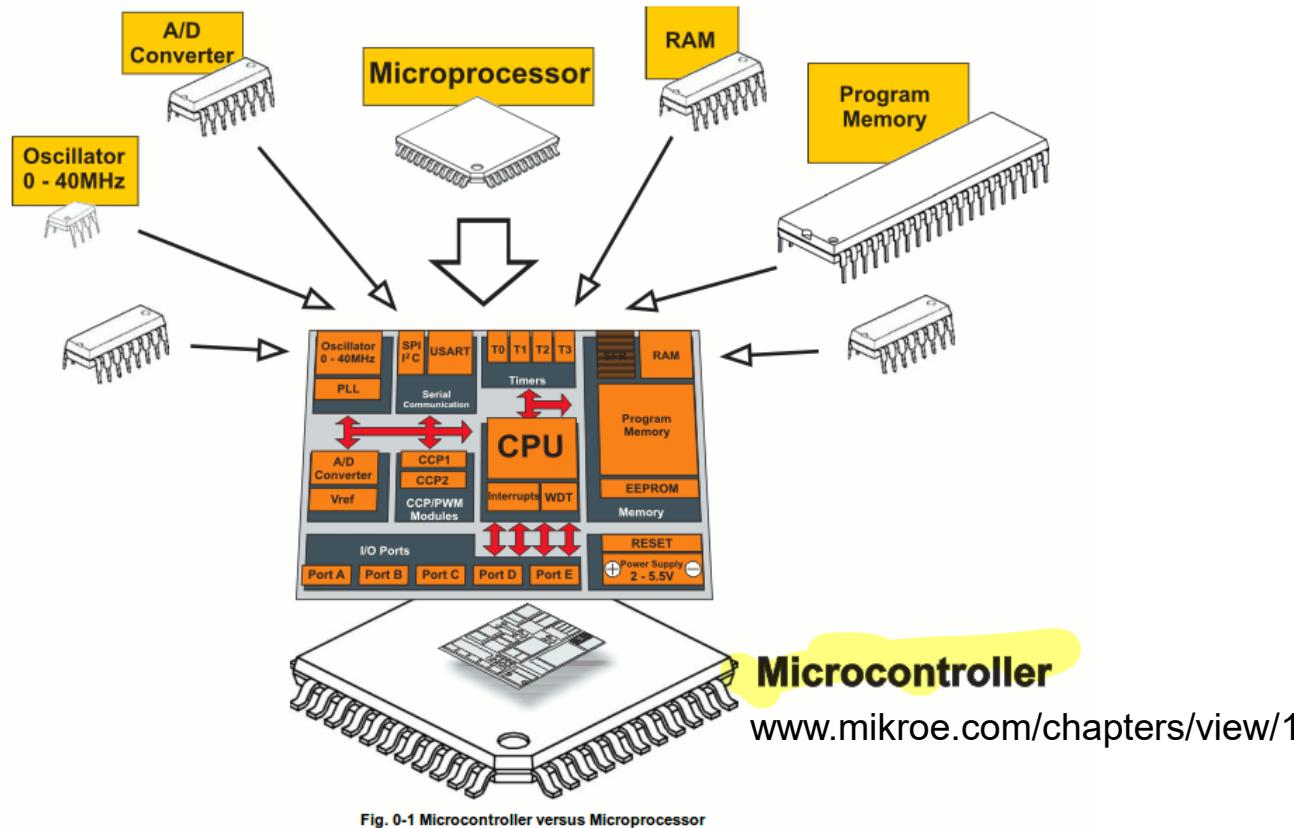
Peripherals



Microcontroller (MCU)

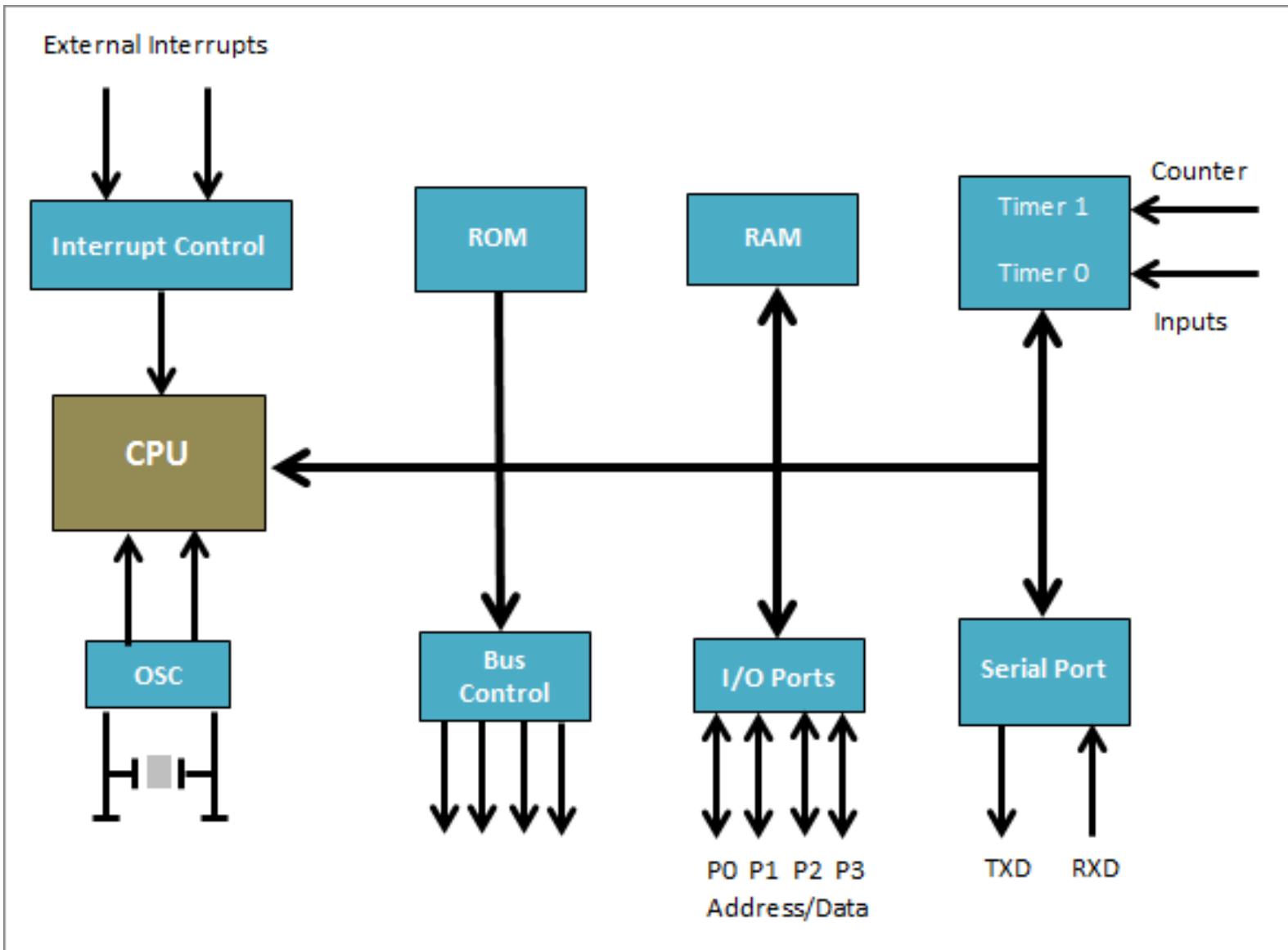
- Microcontroller range from 8 – 40+ pin packages
- Extra pins provide analog and digital I/O options
- 8, 16, 32 & 64 bit instructions & data controllers

What is a Microcontroller



- A small computer on a single chip
 - containing a processor, memory, and input/output
- Typically "**embedded**" inside some device that they control
- A microcontroller is often small and low cost

Block diagram of a microcontroller



Where To Find Microcontrollers?



- Microcontroller are found everywhere!
- uC's are considered embedded systems

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



- ในอดีต ไมโครคอนโทรลเลอร์ที่นิยมใช้งาน คือตระกูล **MCS-51** แต่ในปัจจุบันความต้องการ ความสามารถในการประมวลผลมีเพิ่มขึ้น ทำให้มีสถาปัตยกรรมอื่นๆ ที่มีประสิทธิภาพสูงกว่าเริ่มได้รับความนิยมในการใช้งานขึ้นมาแทน
- ในวิชานี้จะสอนให้นักศึกษาได้เรียนรู้สถาปัตยกรรม **AVR** เนื่องจากมีประสิทธิภาพสูง และมีราคาถูกกว่าสถาปัตยกรรมอื่นๆ ในราคาก็ใกล้เคียงกัน นอกจากนี้ยังมีข้อดีคือ เป็น **Open Source** ทำให้มีข้อมูลต่างๆ และ **Library** ต่างๆ เป็นจำนวนมาก และสามารถพัฒนาระบบได้ทั้งภาษาระดับต่ำ คือภาษา **Assembly** และ ภาษาระดับสูง เช่น ภาษา **C** เป็นต้น

3. สถาปัตยกรรมของ AVR

- แบ่งออกเป็น 2 ระดับคือ 8-bit AVR และ 32-bit AVR
- ในวิชานี้จะกล่าวถึงสถาปัตยกรรมของ AVR ขนาด 8 บิตเท่านั้น
- สถาปัตยกรรม AVR ออกแบบโดย ATMEL เมื่อปี 1996 เป็นชิปปุ๊บ
แบบ **RISC (Reduced Instruction Set Computer)**
- มีสถาปัตยกรรมการต่อหน่วยความจำแบบ **Harvard** ซึ่งแยก
หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ออกจากกันโดยเด็ดขาด
- ใช้หน่วยความจำแบบ **Flash** สำหรับเป็นหน่วยความจำโปรแกรม
- ใช้หน่วยความจำแบบ **SRAM** สำหรับหน่วยความจำข้อมูล
- นอกจากนี้ยังมีหน่วยความจำแบบ **EEPROM** ซึ่งสามารถเก็บข้อมูล
เอาไว้ได้โดยไม่จำเป็นต้องมีไฟเลี้ยง

Difference between RISC and CISC Embedded Architecture



Two Specific Computing in the CPU

- Computer architecture of the CPU has two categories.

- Reduced Instruction Set Computer (RISC)

- Focus on **small** instruction set and running in a clock cycle. (*Simpler instructions, longer code*)
 - A long process build from a sequence of small instruction set.
 - Easy to pipeline
 - CPU: PowerPC, Atmel's AVR, Microchip PIC, ARM processors,

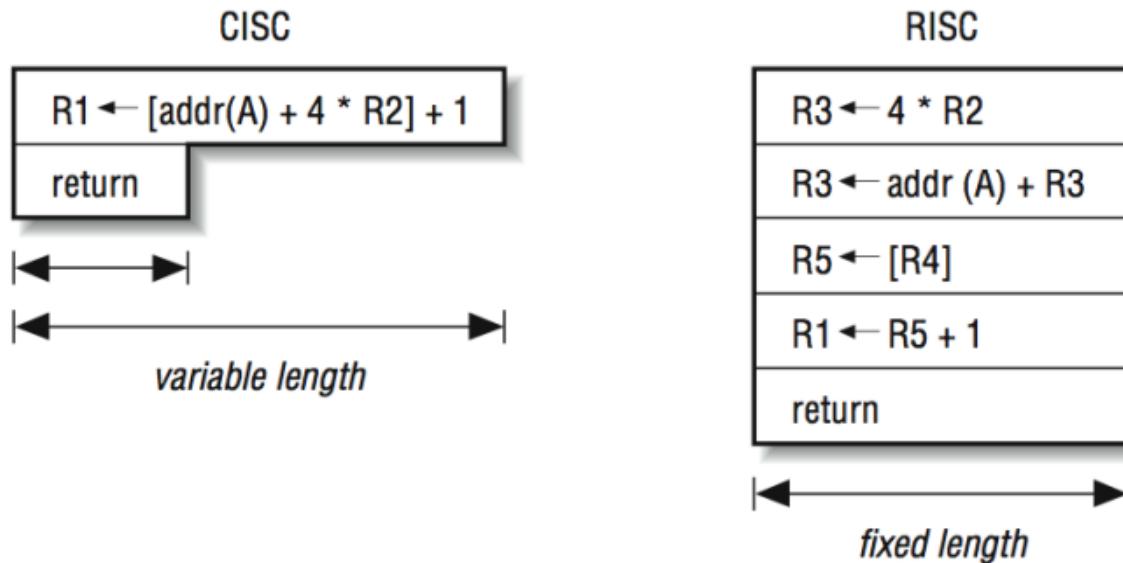
- Complex Instruction Set Computer (CISC)

- Focus on a work completed in an instruction set by using a long clock cycle. (*Longer instructions, shorter code*)
 - Hard to pipeline
 - CPU: Intel x86 architectures

- Hybrid RISC/CISC

- AMD processors

Compare ADD instruction



- Suppose the code below comparing

CISC

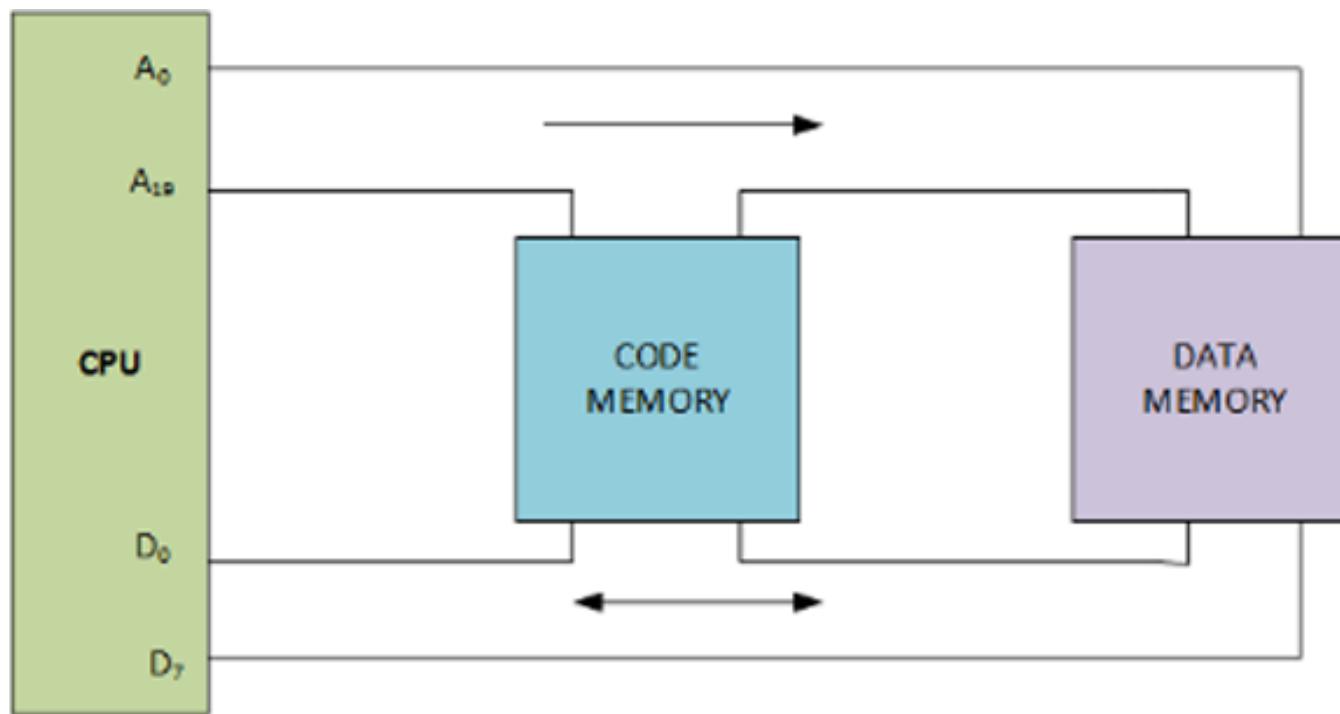
```
mov ax, 10    ← 1  
mov bx, 5     ← 1  
mul bx, ax   ← 30
```

RISC

```
mov ax, 0      ← 1  
mov bx, 10    ← 1  
mov cx, 5      ← 1  
begin: add ax, bx ← 1  
loop cx, begin ← 1
```

Princeton Architecture & Harvard architecture

Every CPU has a memory associated with it to store the program and data. Earlier a single bus was used for getting access to the program and data. A single bus for getting the code and data means, they come to get in each other's way and **slow down the processing speed of the CPU** because each had to wait for the other to finish the fetching.

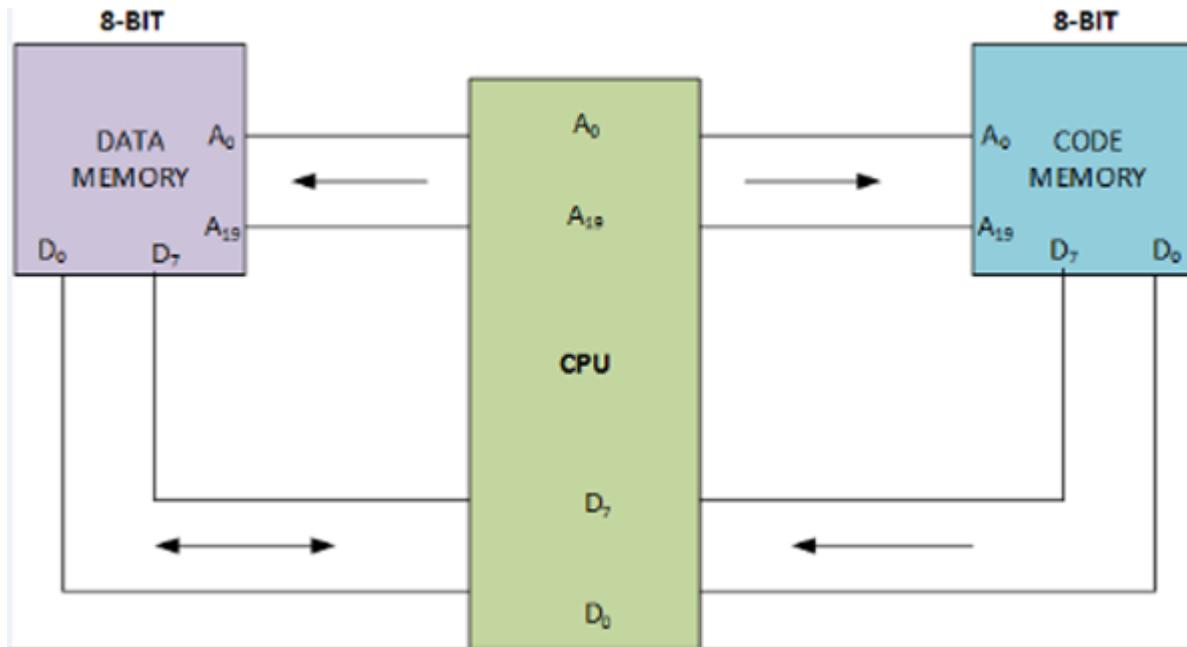


Princeton architecture

Princeton Architecture & Harvard architecture

To speed up the process **Harvard Architecture** was proposed. In this architecture a separate data buses for data and program are present. So it means this architecture proposed the use of four buses

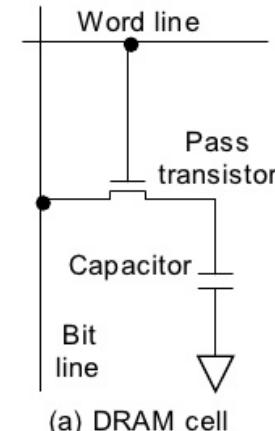
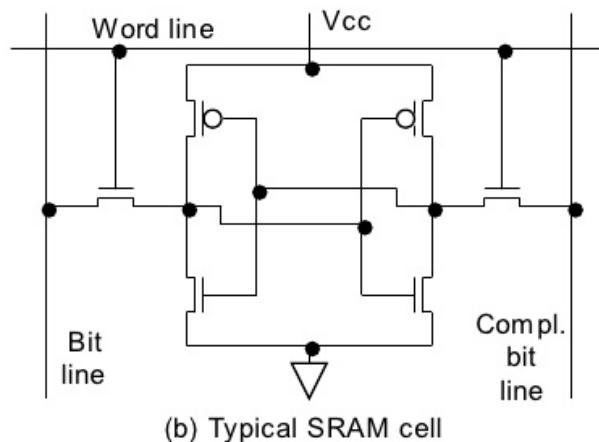
1. A set of data bus carrying the data into and out of the CPU.
2. A set of address bus for accessing the data.
3. A set of data bus for carrying code into the CPU.
4. An address bus for accessing the code.



Harvard Architecture

SRAM & DRAM

Static RAM	Dynamic RAM
➤ SRAM uses transistor to store a single bit of data	➤ DRAM uses a separate capacitor to store each bit of data
➤ SRAM does not need periodic refreshment to maintain data	➤ DRAM needs periodic refreshment to maintain the charge in the capacitors for data
➤ SRAM's structure is complex than DRAM	➤ DRAM's structure is simplex than SRAM
➤ SRAM are expensive as compared to DRAM	➤ DRAM's are less expensive as compared to SRAM
➤ SRAM are faster than DRAM	➤ DRAM's are slower than SRAM
➤ SRAM are used in Cache memory	➤ DRAM are used in Main memory

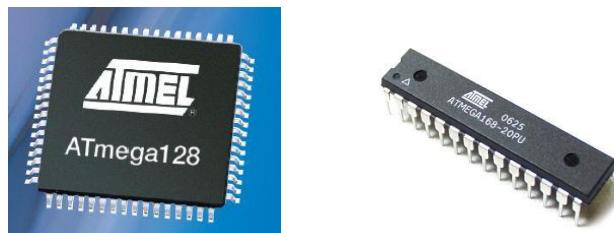


4. ประเภทการใช้งาน AVR ขนาด 8 บิต

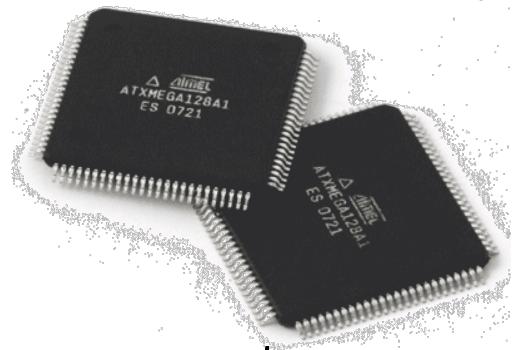
- **tinyAVR** เป็นชิปียูในรุ่นเล็ก ซึ่งต้องการความลึกมากทั้งรัดของวงจร โดยหมายเหตุระบบควบคุมขนาดเล็กๆ ที่ต้องการหน่วยความจำและวงจรสนับสนุนไม่มากนัก ชิปียูในรุ่นนี้จะมีราคาถูกกว่ากลุ่มนี้



- **megaAVR** จะมีชื่ออีกอย่างว่า **ATmega** โดยมีวงจรสนับสนุนภายในเพิ่มเติมตลอดจนเพิ่มขนาดของหน่วยความจำให้ใช้งานมากกว่า ตระกูล **Tiny** เหมาะกับงานควบคุมทั่วๆ ไป



- **XMEGA** เพิ่มความละเอียดของวงจร A/D จากปกติมีความละเอียด 10 บิตในรุ่นเล็กกว่าเป็น 12 บิต และ增加 DMA controller ซึ่งช่วยลดภาระของซีพียูในการควบคุมการรับส่งข้อมูลระหว่าง อุปกรณ์ I/O กับหน่วยความจำ



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

- **Application Specific AVR** เป็นชิปยูที่ออกแบบมาโดยเพิ่ม
วงจรควบคุมเฉพาะด้านเข้าไปซึ่งไม่พบในชิปยูกลุ่มนี้น่า เช่น วงจร USB
controller หรือ วงจร CAN bus เป็นต้น
-

AVR มีให้เลือกใช้งานหลายเบอร์ แต่ละเบอร์จะมีขนาด ราคา
ความสามารถ และขนาด หน่วยความจำตลอดจนถึงวงจรสนับสนุน
ภายในที่แตกต่างกันออกไป

5.1 ATmega328

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 32 Kbyte
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 2 Kbyte
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด 1 Kbyte
- สนับสนุนการเชื่อมต่อแบบ I²C bus , SPI , UART
- พอร์ตอินพุตเอาต์พุตจำนวน 20 ports
- Analog input (ADC) 6 Port ขนาด 10 bit
- วงจรนับ/จับเวลาขนาด 8 บิต จำนวน 2 ตัว และขนาด 16 บิตจำนวน 1 ตัว
- สนับสนุนช่องสัญญาณสำหรับสร้าง PWM จำนวน 6 ช่องสัญญาณ
- ทำงานได้ตั้งแต่ปานกลางดัน 1.8-5.5 Volts
- ความถี่ใช้งานสูงสุด 20 MHz

รายละเอียด ขา ต่างๆ ของ ATmega328

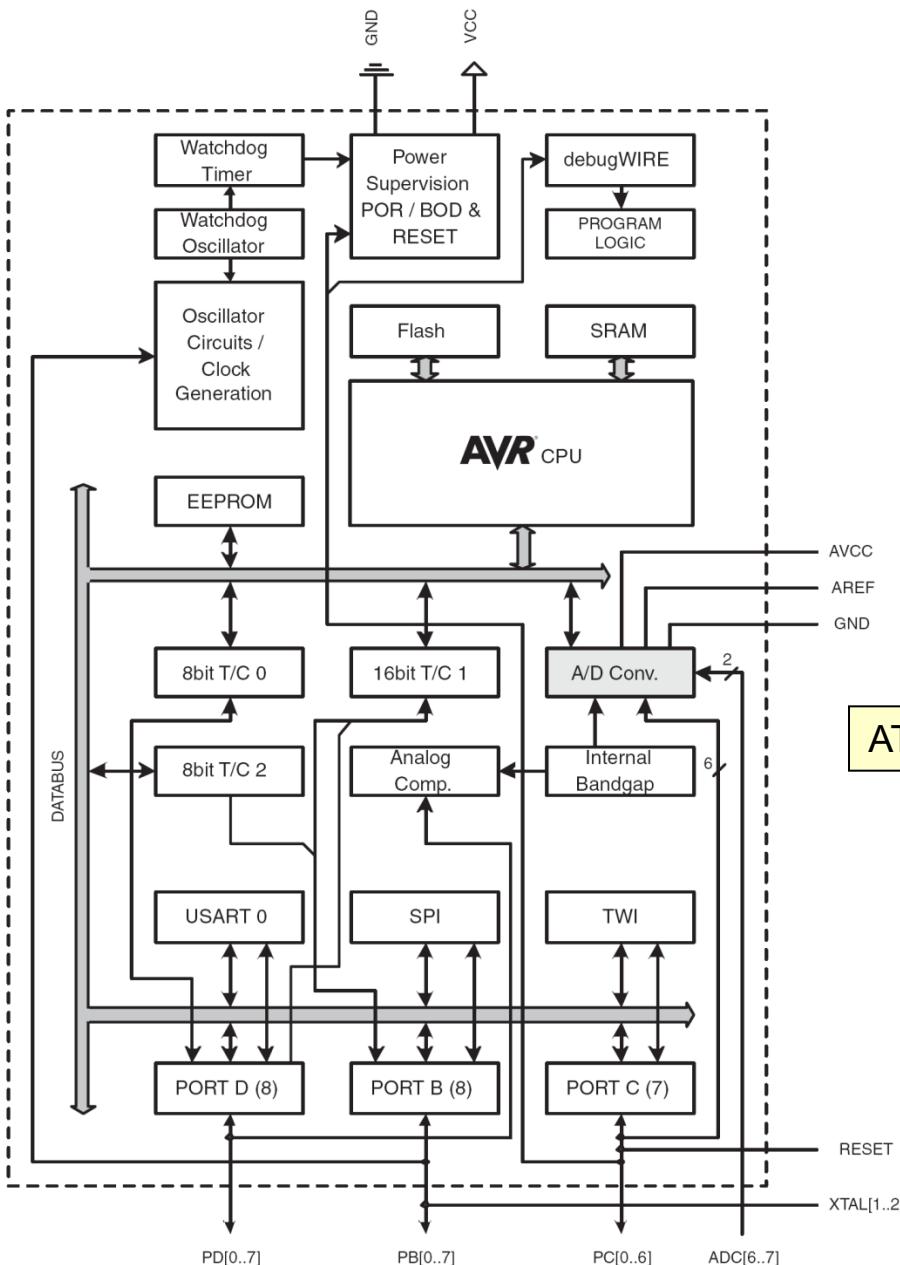
ATMega328P and Arduino Uno Pin Mapping

Arduino function

reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLK0/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

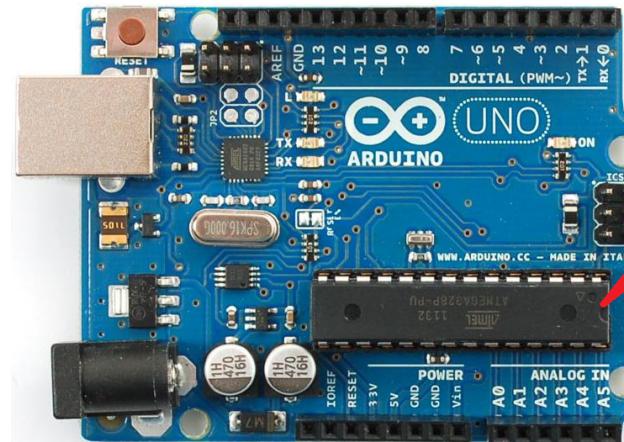
Digital Pins 11,12 & 13 are used by the ICSP header for MOSI,
MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-
impedance loads on these pins when using the ICSP header.

ATmega328 Internal Architecture



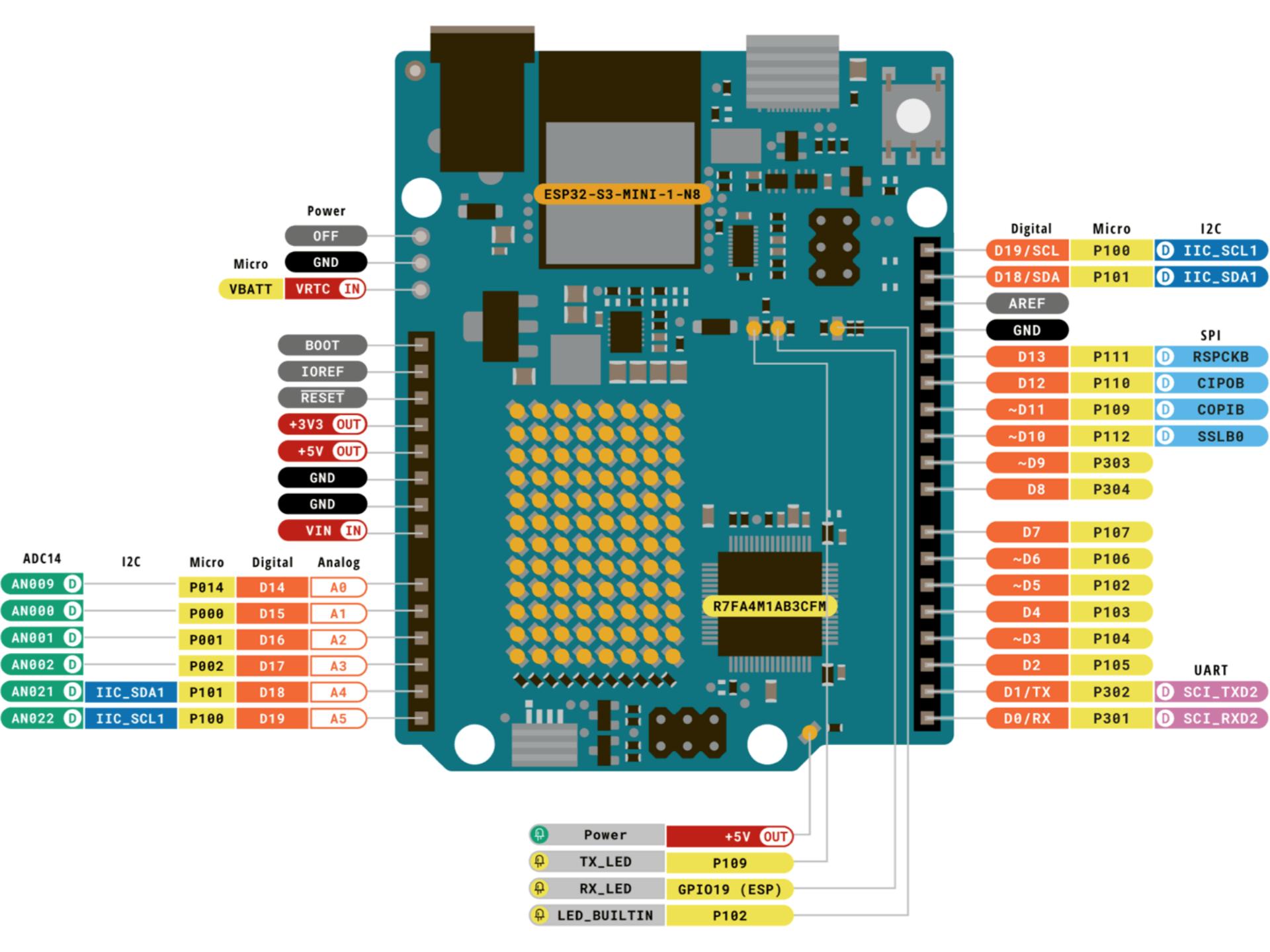
(PCINT14/RESET)	PC6	1	28	PC5 (ADC5/SCL/PCINT13)
(PCINT16/RXD)	PD0	2	27	PC4 (ADC4/SDA/PCINT12)
(PCINT17/TXD)	PD1	3	26	PC3 (ADC3/PCINT11)
(PCINT18/INT0)	PD2	4	25	PC2 (ADC2/PCINT10)
(PCINT19/OC2B/INT1)	PD3	5	24	PC1 (ADC1/PCINT9)
(PCINT20/XCK/T0)	PD4	6	23	PC0 (ADC0/PCINT8)
VCC		7	22	GND
GND		8	21	AREF
PCINT6/XTAL1/TOSC1)	PB6	9	20	AVCC
PCINT7/XTAL2/TOSC2)	PB7	10	19	PB5 (SCK/PCINT5)
(PCINT21/OC0B/T1)	PD5	11	18	PB4 (MISO/PCINT4)
(PCINT22/OC0A/AIN0)	PD6	12	17	PB3 (MOSI/OC2A/PCINT3)
(PCINT23/AIN1)	PD7	13	16	PB2 (SS/OC1B/PCINT2)
(PCINT0/CLKO/ICP1)	PB0	14	15	PB1 (OC1A/PCINT1)

ATmega328 data sheet pp. 2, 5



5.2 Renesas RA4M1 (Arm Cortex-M4F) ຕັດກໍເຈັກ

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด **256 Kbyte**
- หน่วยความจำข้อมูลแบบ SRAM ขนาด **32 Kbyte**
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด **8 Kbyte**
- สนับสนุนการเชื่อมต่อแบบ I²C bus , SPI, UART, CAN
- พอร์ตอินพຸຕເອົາຕຸພຸຕຈຳນວນ 20 ports
- Analog input (ADC) 6 port ขนาด **14 bit**, DAC 1 port (**12 bit**)
- วงศ์รันນັບ/ຈຳປເວລາขนาด 8 ບີຕ ຈຳນວນ 2 ຕັ້ງ ແລະ ขนาด 16 ບີຕຈຳນວນ 1 ຕັ້ງ
- สนับสนູນຊ່ອງສົງຄູານສໍາຮັບສ້າງ PWM ຈຳນວນ 6 ຊ່ອງສົງຄູານ
- ทำงานທີ່ແຮງດັນ **5 Volts**
- ຄວາມຄືໃຊ້ງານສູງສຸດ **48 MHz**
- WiFi 802.11 b/g/n 2.4GHz band , Bluetooth 5.0



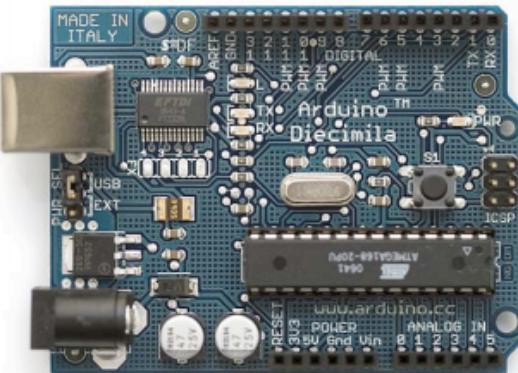
6. What is Arduino ? [1]

- Arduino เป็นโครงการพัฒนาไมโครคอนโทรลเลอร์ของตระกูล AVR แบบ Open Source
- **Arduino** เป็นชื่อเรียกของ **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** มาตรฐานในการอ้างอิงกับบอร์ด Arduino จำนวนมาก ทำให้สะดวกในการพัฒนาโปรแกรม
- โดย **microcontroller** ตระกูล AVR ที่นำมาใช้นั้น จะต้องมีการติดตั้ง Firmware ไว้แล้ว โดยหน้าที่หลักของ Firmware คือ การรับโปรแกรมที่ **Compile** แล้ว จาก IDE มาเขียนไว้ที่ตัวมันเอง เพื่อทำงานตามโปรแกรมที่เขียน

What is Arduino ? [2]

The word “Arduino” can mean 3 things

A physical piece
of hardware



A programming
environment

```
Arduino - 0010 Alpha

Blink §

* The basic Arduino example. Turns on an LED on for one second,
* then off for one second, and so on... We use pin 13 because,
* depending on your Arduino board, it has either a built-in LED
* or a built-in resistor so that you need only an LED.

* http://www.arduino.cc/en/Tutorial/Blink

int ledPin = 13; // LED connected to digital pin 13

void setup() // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() // run over and over again
{
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for a second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000); // waits for a second
}
```

A community
& philosophy

Arduino playground

Welcome to the Arduino Playground, a wiki where all the users of Arduino can contribute and benefit from their collective research.

This is the place to post and share your own code, circuit diagrams, tutorials, DIY instructions, tips and tricks, and after all the hard work, to show off your project!

Arduino Playground is a *work in progress*. We can use all the help you can give, so please read the [Participate](#) section and get your fingers typing!

:: About the Arduino Playground ::

There is a lot to do... most of the pages are just stubs, simple placeholders waiting for you to fill them up. However here is a small roadmap of things I personally think should be developed first: (again, this is a wiki so you are more than welcome to get in

:: RoadMap: What Needs to be Done? ::

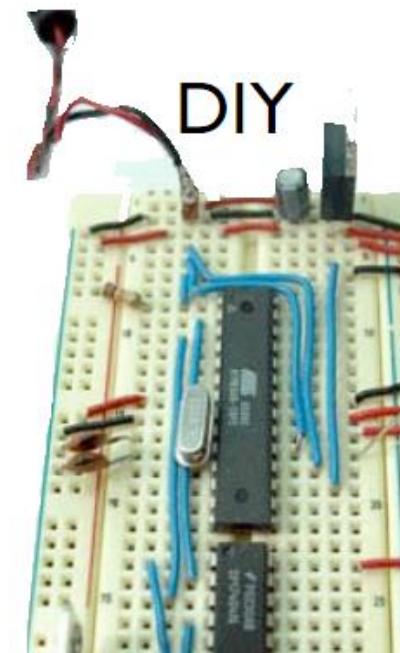
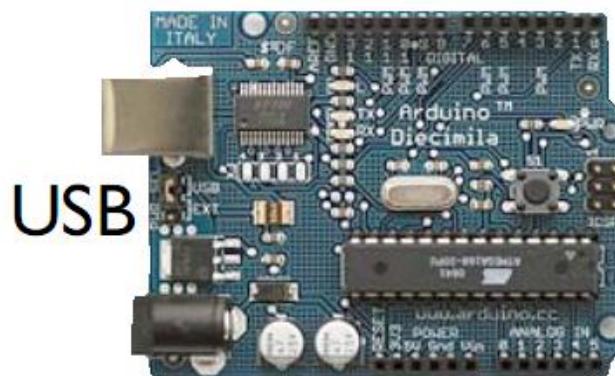
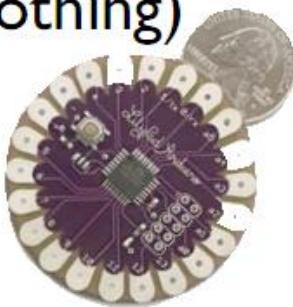
Arduino Tutorials
Official tutorial page
Information on the Arduino Mini
Bluetooth
Xbee (Zigbee) tutorial
DIY Arduino Shield
Serial LCD tutorial
LCD library
Learning Programming
Asus Robot and Arduino
Let the Arduino Sleep
Building Broadcasters
Courses
Stand alone 3.3V 8 MHz
Programming the Bluetooth WTI1
Retrofitting Diecimila AutoReset
Beam Boxes "Diecimila" Upgrade

Run Arduino on...
Linux
Ubuntu Linux
Debian Linux
gentoo Linux
BlackBerry Linux
FreeBSD
Fedora Linux
The command line

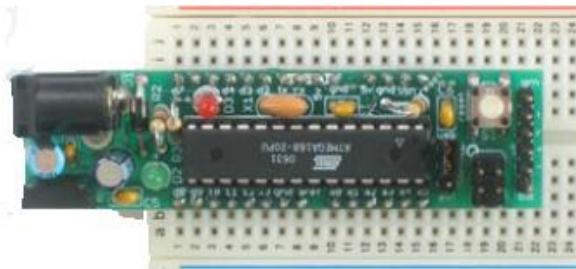
Interface Arduino with...
Instant Reality (X3D)
TurtleBot
Processing
PD (Pure Data)

6.1 Arduino Hardware Variety

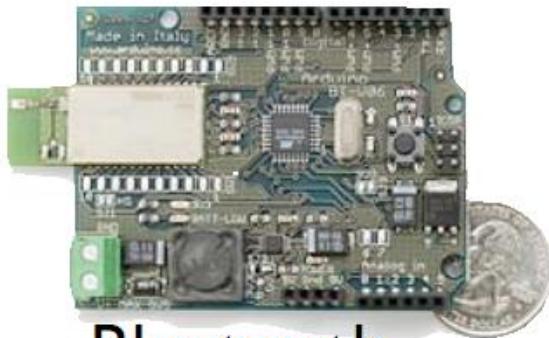
LilyPad
(for clothing)



Boarduino Kit



Bluetooth



“Stamp”-sized



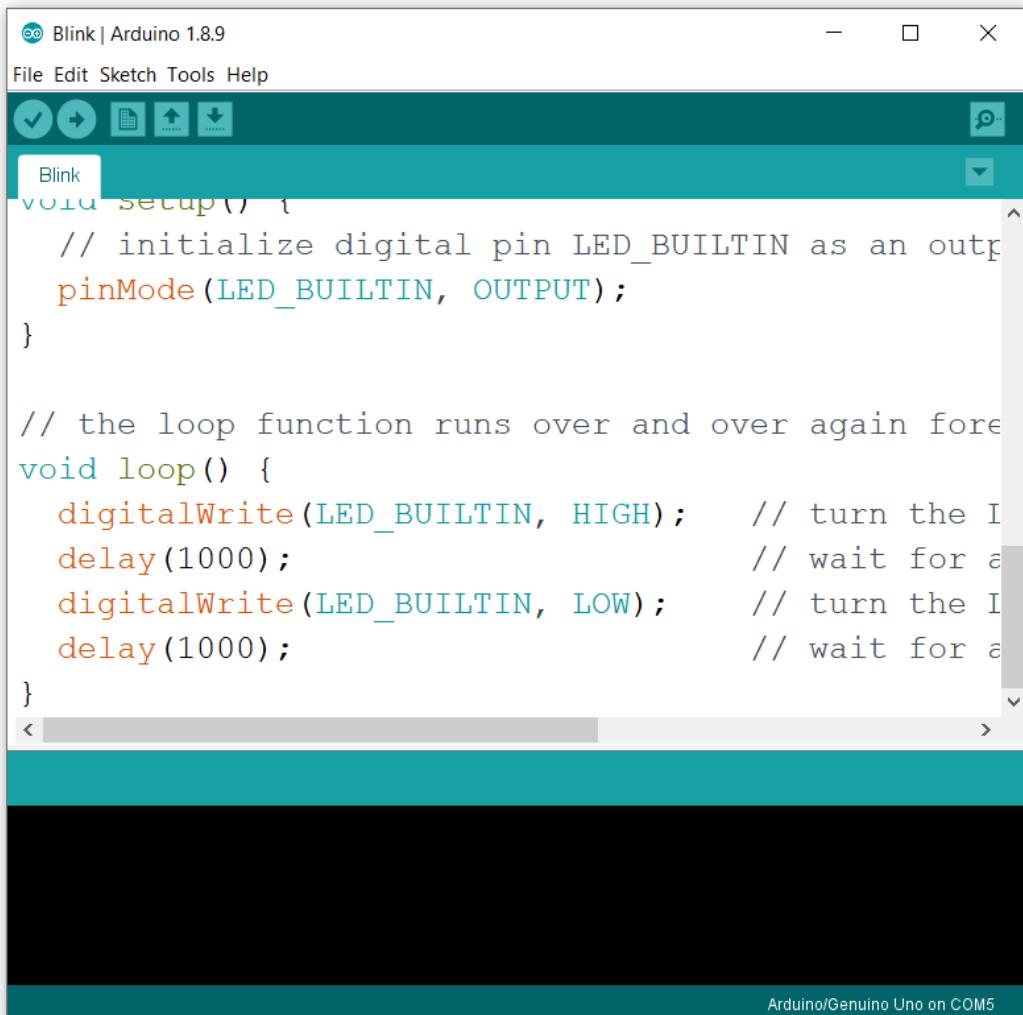
many different variations to suite your needs

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

6.2. Arduino IDE



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.9". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Print. A tab labeled "Blink" is active. The main code editor window displays the following sketch:

```
void setup() {
    // initialize digital pin LED_BUILTIN as an output
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on
    delay(1000);                      // wait for a second
    digitalWrite(LED_BUILTIN, LOW);     // turn the LED off
    delay(1000);                      // wait for a second
}
```

At the bottom of the IDE, a status bar indicates "Arduino/Genuino Uno on COM5".

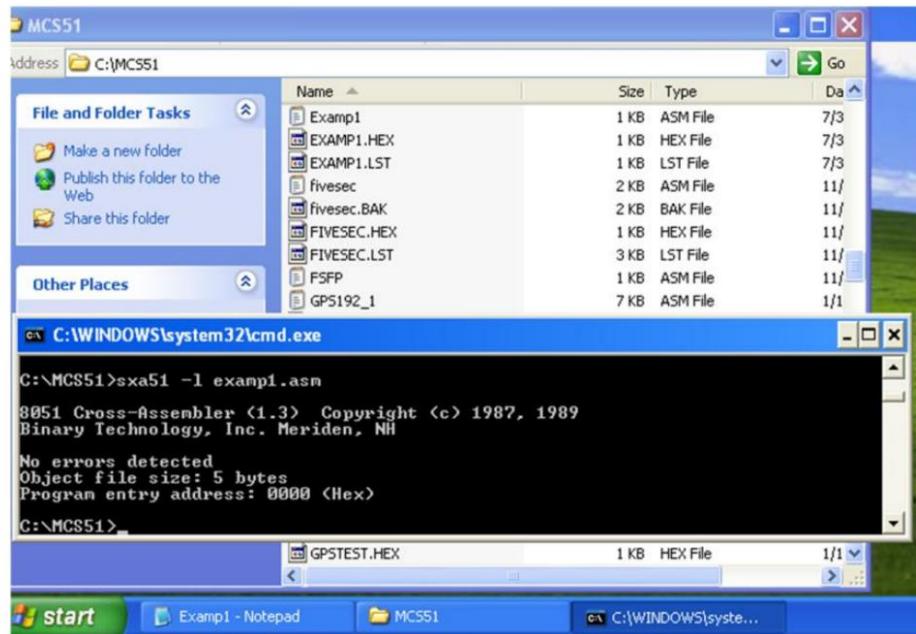
- Like a text editor
- View/write/edit sketches
- But then you program them into hardware

IDE (Integrated Development Environment)

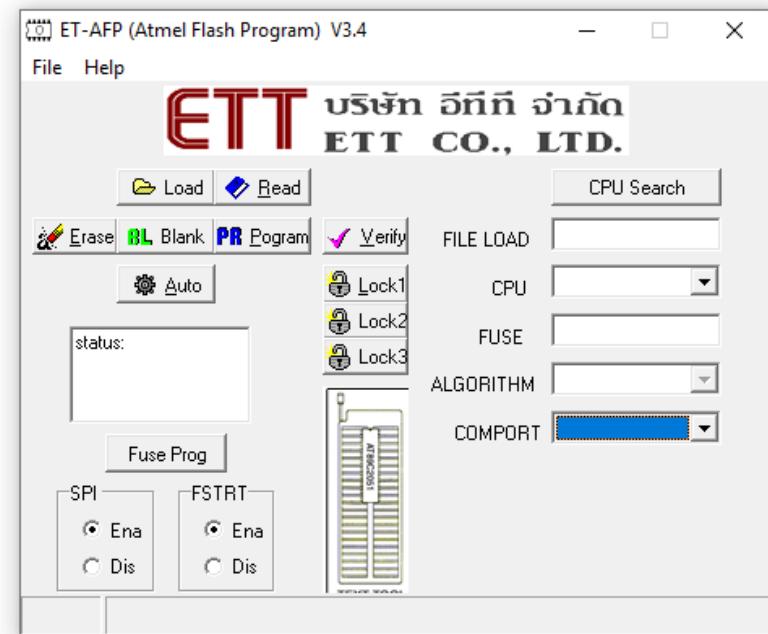
- ในอดีต การพัฒนาโปรแกรมขึ้นมา ต้องใช้โปรแกรมอื่นๆ หลายตัวในการสร้างโปรแกรม
- เช่น ต้องการเขียนโปรแกรมลงบอร์ด MCS51 มีขั้นตอน
 - เขียนโปรแกรม โดยใช้โปรแกรม **text editor** เช่น notepad
 - Compile program โดยใช้โปรแกรม **SXA51**
 - Upload โปรแกรม โดยใช้โปรแกรม **ET-AFP**
 - ดูค่าที่ได้จากโปรแกรม โดยใช้โปรแกรม **hyperterminal**

จึงเป็นที่มาของการพัฒนา IDE ขึ้นมา โดยรวมโปรแกรมย่อยต่างๆ ให้เป็นโปรแกรมเดียว ที่มีสภาพแวดล้อมที่อำนวยต่อการพัฒนาโปรแกรมให้ง่ายขึ้น

IDE (Integrated Development Environment)

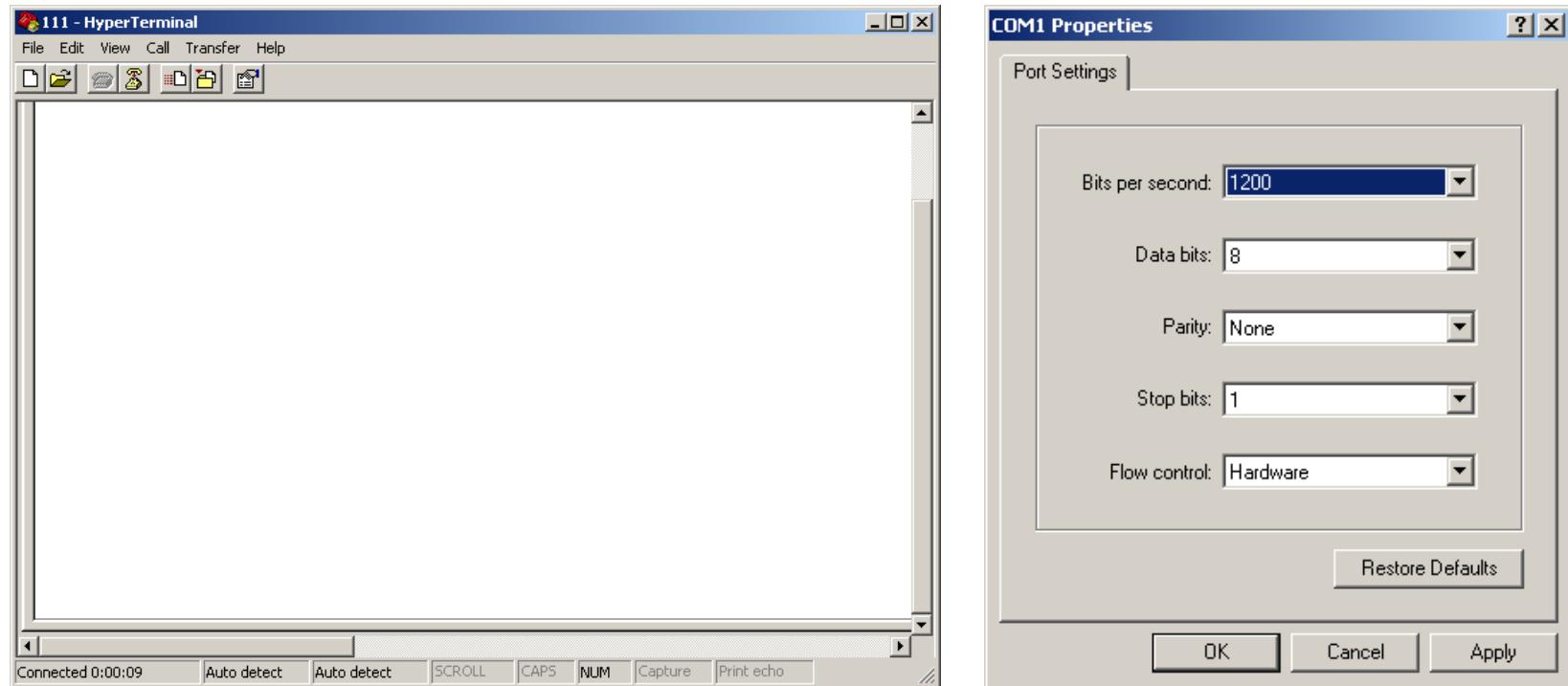


Compile **file.asm** to **file.hex**



Upload **file.hex** to board

IDE (Integrated Development Environment)



หน้าจอโปรแกรม hyperterminal

6.3 ข้อดี-ข้อเสีย Arduino

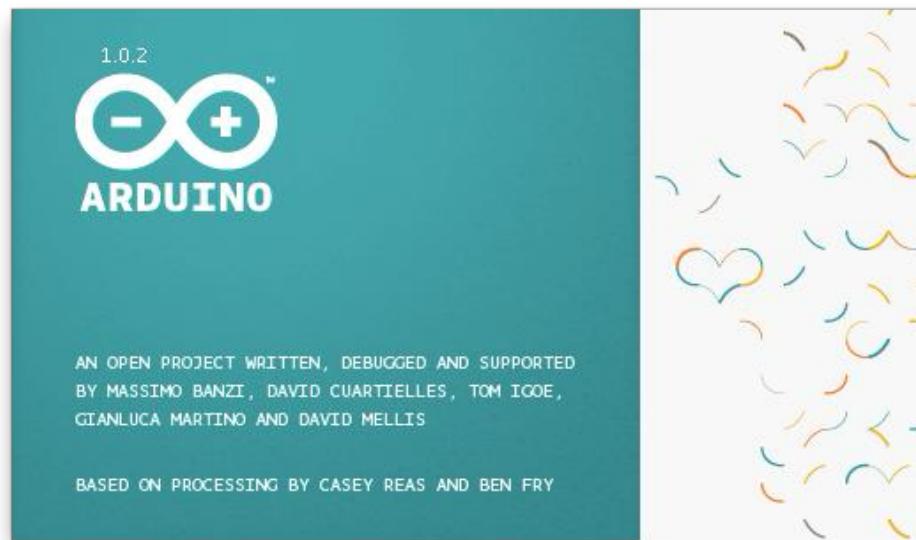
ข้อดี

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

ข้อเสีย

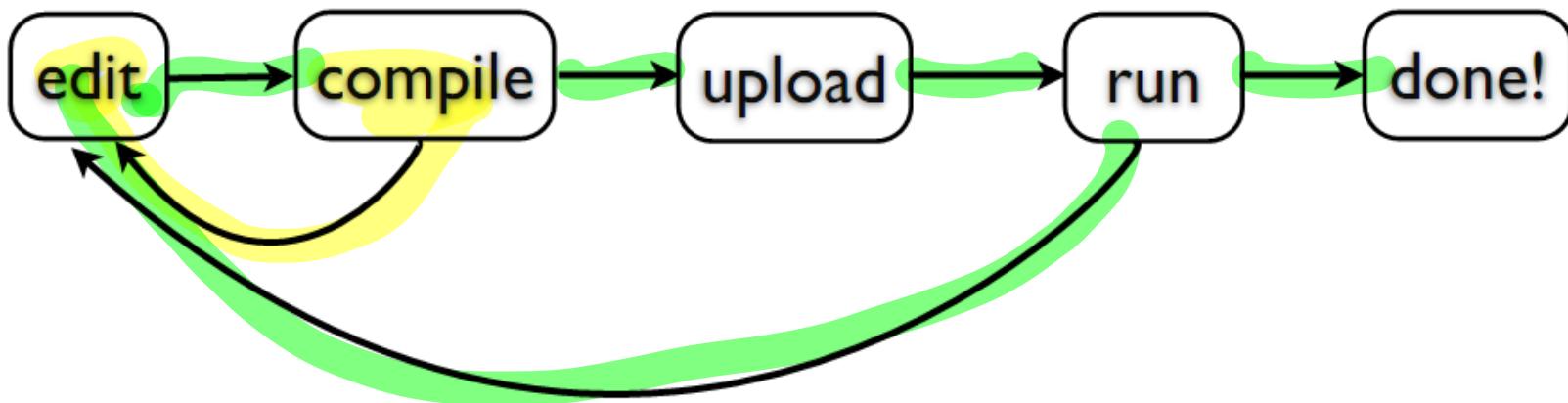
- เนื่องจากพัฒนาโปรแกรมด้วยภาษา C++ จึงใช้ทรัพยากรามากพอสมควร

2. Arduino Software and Instruction set



1. ขั้นตอนการพัฒนาโปรแกรมด้วย Arduino (Development Circle)

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

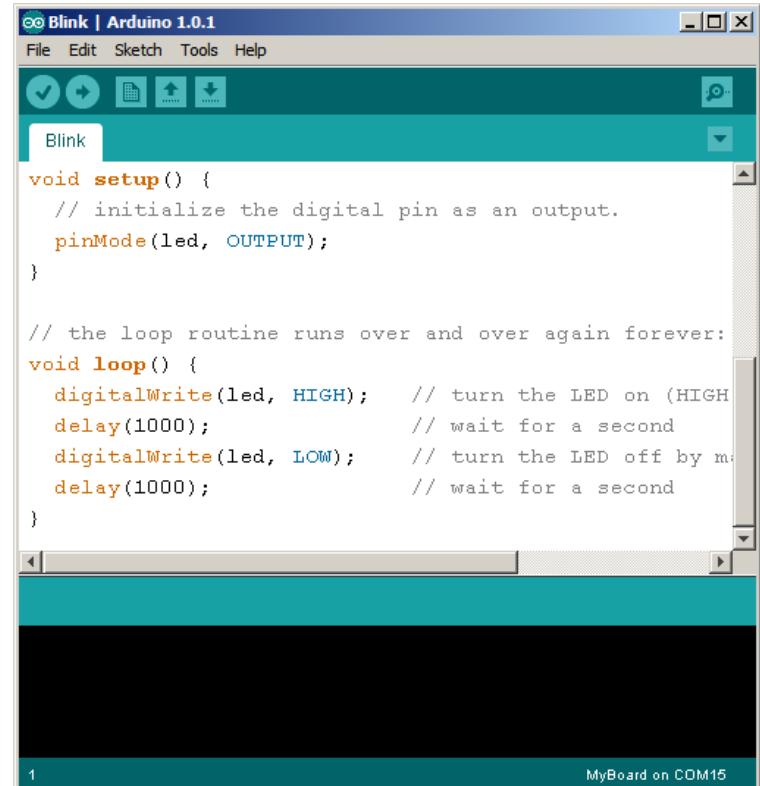


2. Arduino IDE

The Arduino programming platform was designed in JAVA to help newcomers become familiar with programming. The language used to write code is C/C++ .

Each Arduino program is called a **SKETCH** and has two required functions, called ROUTINES.

void setup (){} , void loop (){}



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.0.1". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for Open, Save, Upload, and Download. A status bar at the bottom right shows "MyBoard on COM15". The main area displays the following C/C++ code:

```
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);      // turn the LED on (HIGH)
  delay(1000);                // wait for a second
  digitalWrite(led, LOW);       // turn the LED off by making
  delay(1000);                // wait for a second
}
```

3. Arduino Program (Sketch) Structure

Declare variables at top

- Initialize
 - **setup()** – run ONCE at beginning, set pins
- Running
 - **loop()** – run repeatedly, after **setup()**

All of the code within the curly braces will be run again, and again, until the power is removed.

Example of a bare minimum program:

```
void setup()
{
}

void loop()
{}
```

4. Programming - Syntax

Similar to C, the formatting requirement is the same.

//

- Single line comment

/* */

- Multiline comment

{ }

- used to define a block of code that starts and ends.

;

- used to define the end of a line of code.

5. Programming Concepts: Variables

ProtosnapProMiniExample2 \$

```
// Comments go here  
// Written by: Joesephine Jones  
// Date: April 12, 2013  
  
int sensorValue;  
int ledPin;  
  
void setup()  
{  
    // put your setup code here, to run once:  
    int setupVariable;  
  
}  
  
void loop()  
{  
    // put your main code here, to run repeatedly:  
    int loopScopeVariable  
}
```

Variable Scope

• *Global*

• ---

• *Function-level*

Variable Types

- **Variable Types:**



8 bits

byte
char



16 bits

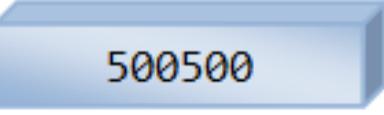
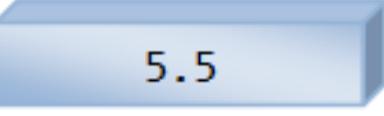
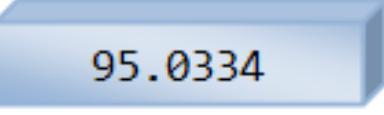
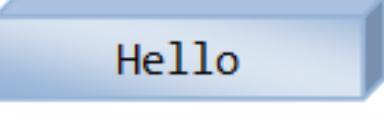
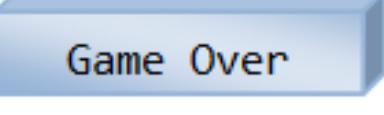
int
unsigned int



32 bits

long
unsigned long
float

Variables

TYPE	NAME	VALUE	
int	number	→ 	Stored only Integer
int	sum	→ 	Stored only Integer
double	radius	→ 	Stored only floating-point number
double	area	→ 	Stored only floating-point number
String	greeting	→ 	Stored only texts
String	statusMsg	→ 	Stored only texts

A *variable* has a **name**, stores a **value** of the declared **type**.

Format of variables

All variables have to be ***declared*** before they are used.

Declaring a variable means ***defining its type***, and optionally, setting an initial value (initializing the variable).

For example,

```
int inputVariable = 0;
```

declares that variable **inputVariable** is of type **int**, and that its initial value is zero.

Format of variables

1. **char** – a data type that takes up **1 byte** of memory that stores a character value. Character literals are written in single quotes, like this: 'A'.

2. **byte** – a byte stores an **8-bit** unsigned number, from **0 to 255**.

3. **int** – integers are your primary data type for number storage, and store a **2 byte** value. This yields a range of **-32,768 to 32,767**.

4. **unsigned int** – unsigned integers are the same as integers in that they store a **2 byte** value. Instead of storing negative numbers however they only store positive values, yielding a useful range of **0 to 65,535**.

Format of variables

5. **long** – long variables are extended size variables for number storage, and store 32 bits (**4 bytes**), from **-2,147,483,648** to **2,147,483,647**.

6. **unsigned long** – unsigned long variables are extended size variables for number storage, and store 32 bits (**4 bytes**). Unlike standard longs unsigned longs won't store negative numbers, making their range from **0 to 4,294,967,295**

Format of variables

7. **Float** – datatype for floating-point numbers, a number that has a decimal point. They are often used to approximate analog and continuous values because they have greater resolution than integers. Floating-point numbers can be as large as **3.4028235E+38** and as low as **-3.4028235E+38**. They are stored as 32 bits (**4 bytes**) of information.
8. **double** – double precision floating point number. Occupies **4 bytes**. The double implementation on the Arduino is currently exactly the same as the float, with no gain in precision.

Format of variables

Example of variables:

```
char myChar = 'A';  
char myChar = 65; // both are equivalent  
  
byte b = B10010; // "B" is the binary  
                  formatter (B10010 = 18  
                  decimal)  
  
int ledPin = 13;  
  
unsigned int ledPin = 13;
```

6. Arduino data types

Numeric types	Bytes	Range	Use
int	2	-32768 to 32767	Represents positive and negative integer values.
unsigned int	2	0 to 65535	Represents only positive values; otherwise, similar to int.
long	4	-2147483648 to 2147483647	Represents a very large range of positive and negative values.
unsigned long	4	4294967295	Represents a very large range of positive values.
Numeric types	Bytes	Range	Use
float	4	3.4028235E+38 to -3.4028235E+38	Represents numbers with fractions; use to approximate real-world measurements.
double	4	Same as float	In Arduino, double is just another name for float.
boolean	1	false (0) or true (1)	Represents true and false values.
char	1	-128 to 127	Represents a single character. Can also represent a signed value between -128 and 127.
byte	1	0 to 255	Similar to char, but for unsigned values.
Other types			
string			Represents arrays of chars (characters) typically used to contain text.
void			Used only in function declarations where no value is returned.

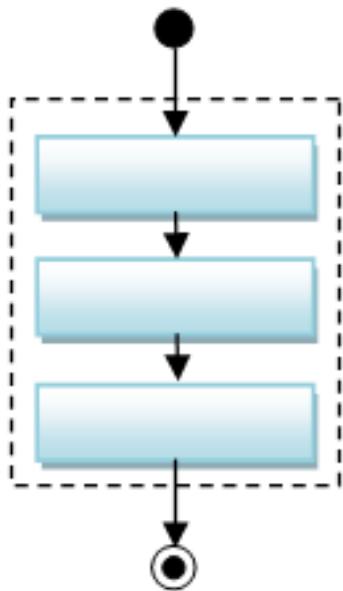
7. Arithmetic Operators

Math	Description
=	Assignment : makes something equal to something
%	Modulo : this gives the remainder when one number is divided by another
+	Addition
-	Subtraction
*	Multiplication
/	Division

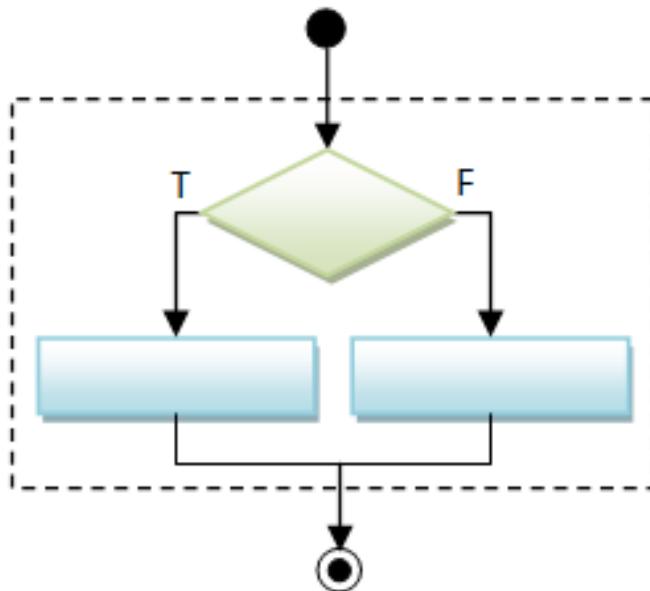
8. Comparison Operators

Comparison	Description
() == ()	is equal?
() != ()	is not equal?
() > ()	greater than
() >= ()	greater than or equal
() < ()	less than
() <= ()	less than or equal

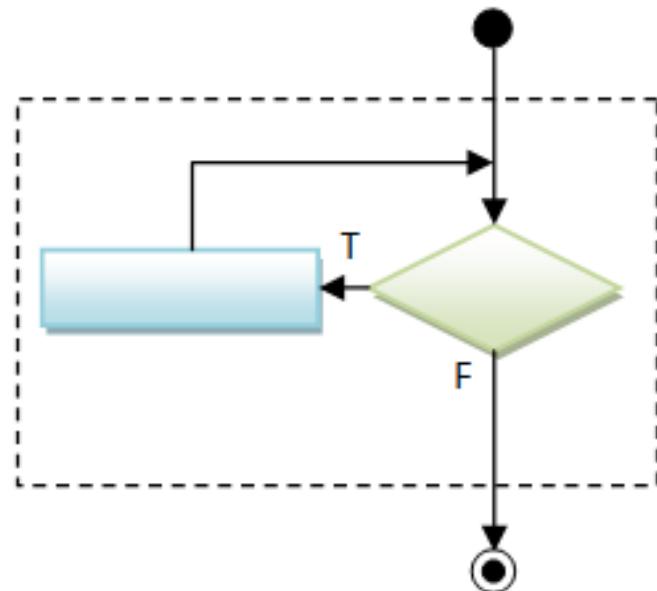
9. Flow Control



Sequential



Conditional (Decision)

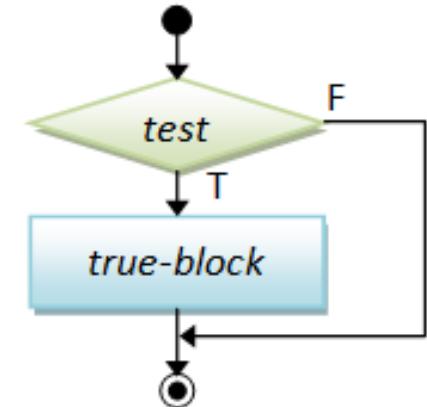


Loop (Iteration)

Condition Flow Control

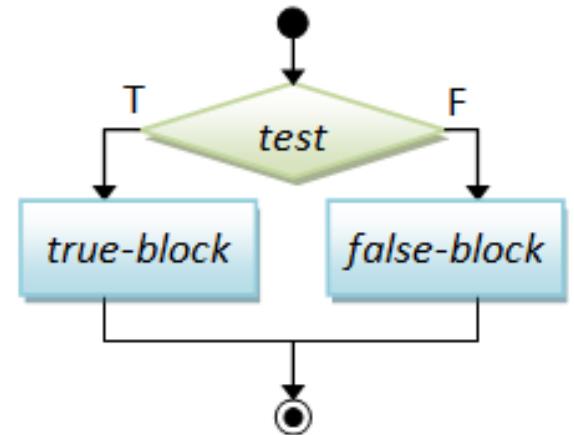
1. if- then

```
if(expression)
{
    True Block;
}
```



2. if- then-else

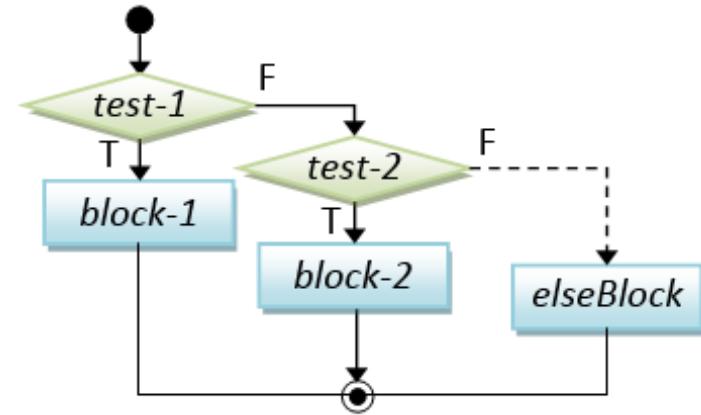
```
if(expression)
{
    True Block;
} else
{
    False Block;
}
```



10. Condition Flow Control

3. nested-if

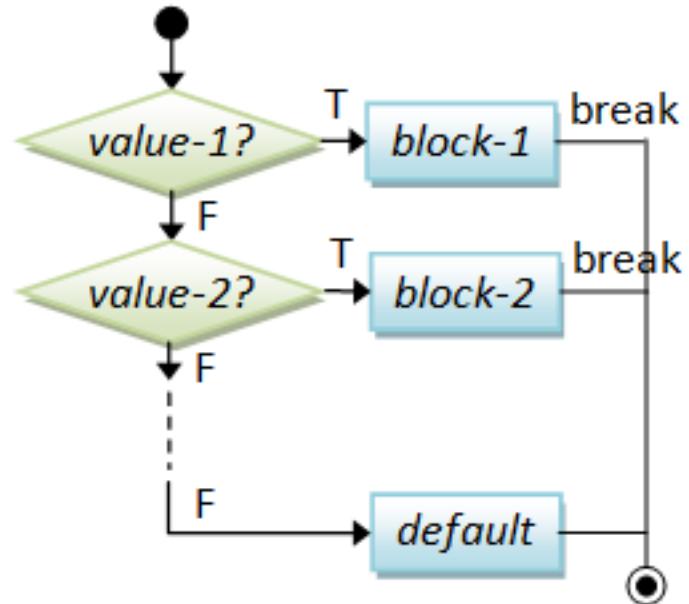
```
if(expression 1)
{
    True Block-1;
} else if(expression 2)
{
    True Block-2;
} else if(expression 3)
{
    ...
} else
{
    else Block
}
```



Condition Flow Control

4. switch-case

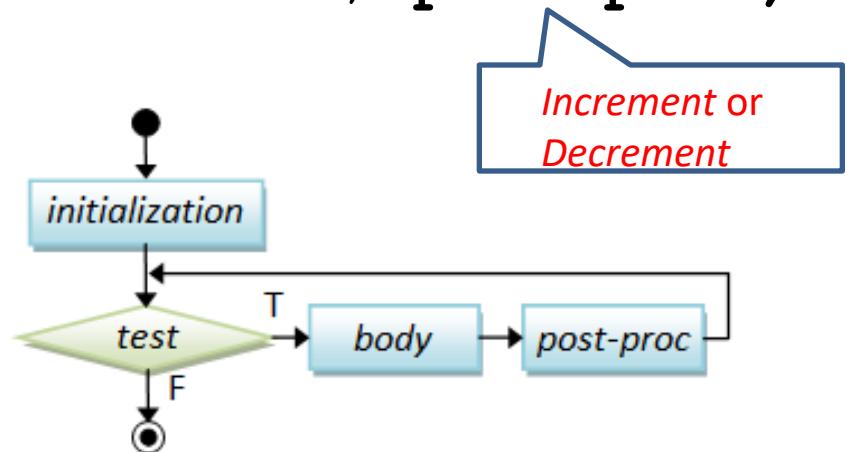
```
switch(selector) {  
    case value-1:  
        Block-1;  
        break;  
    case value-2:  
        Block-2;  
        break;  
    case value-3:  
        Block-3;  
        break;  
    ....  
    default  
        Default Block  
        break;  
}
```



11. Loop Flow Control

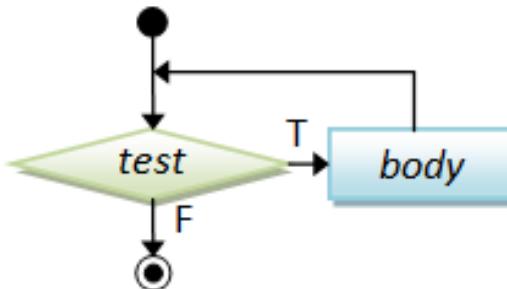
1. for

```
for (initial; test-condition; post-proc)
{
    body;
}
```



2. while

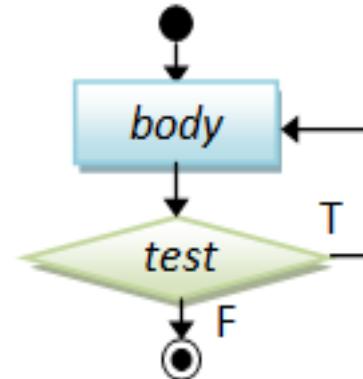
```
while (expression)
{
    body;
}
```



Loop Flow Control

3. do.. while

```
do  
{  
    Body;  
}  
while (expression);
```



12. Arduino Function

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\(\)](#)

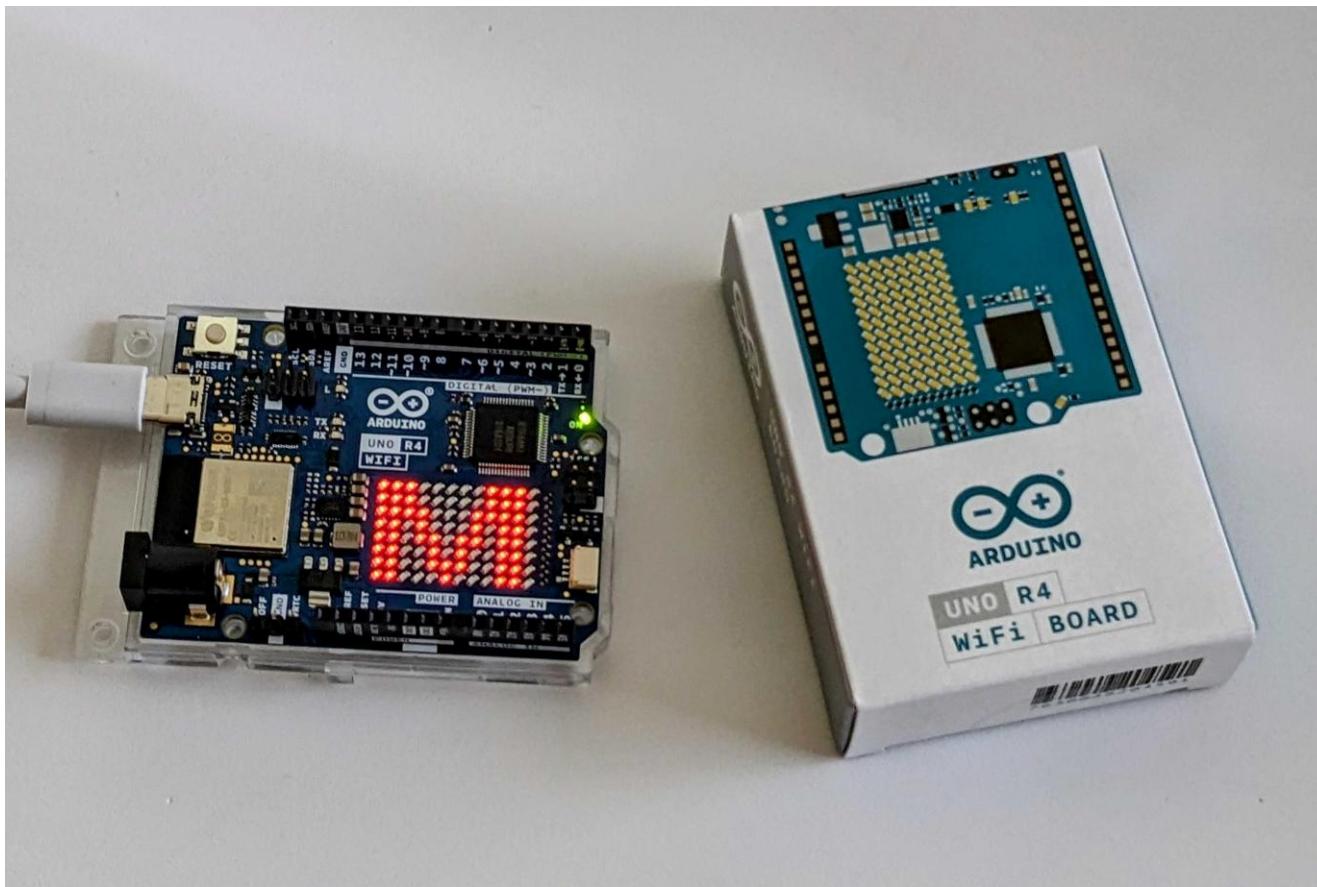
Trigonometry

- [sin\(\)](#)
- [cos\(\)](#)
- [tan\(\)](#)

Random Numbers

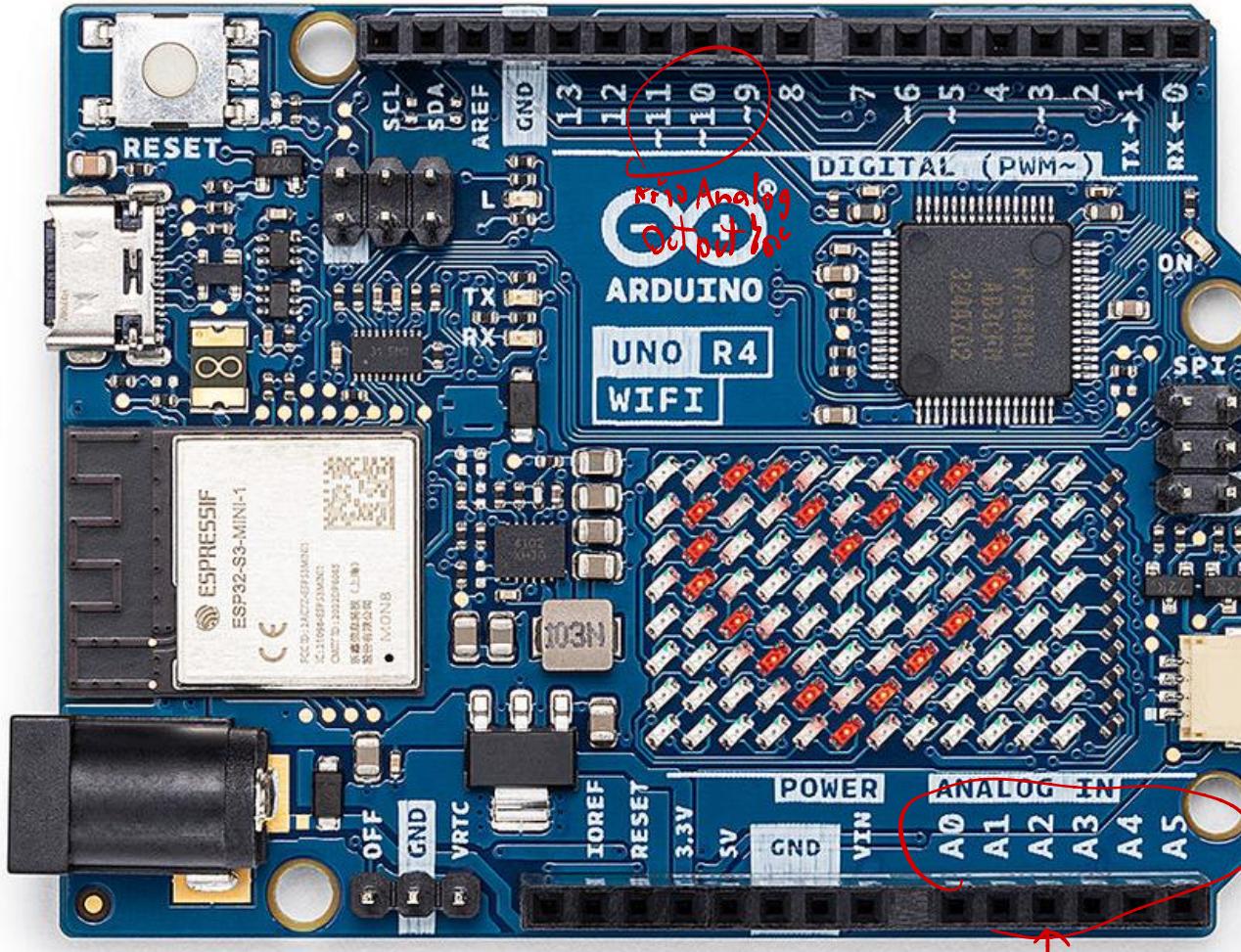
- [randomSeed\(\)](#)
- [random\(\)](#)

3. Laboratory Device



Arduino Uno Board

Digital Input / Output D0-D13

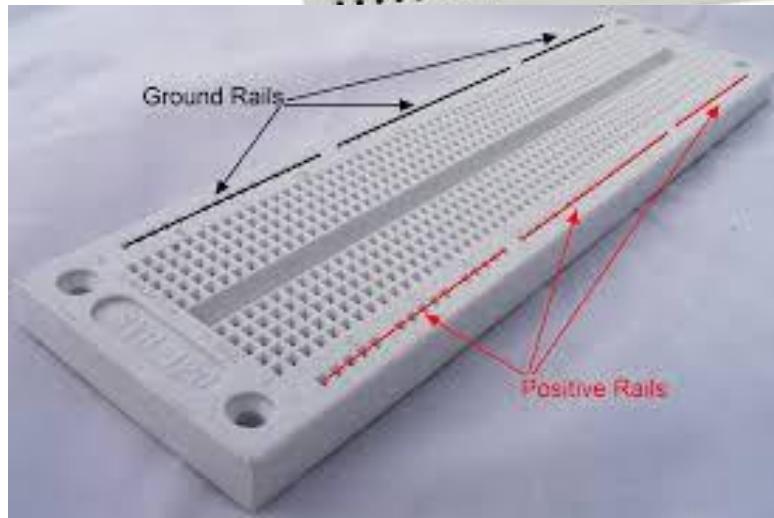
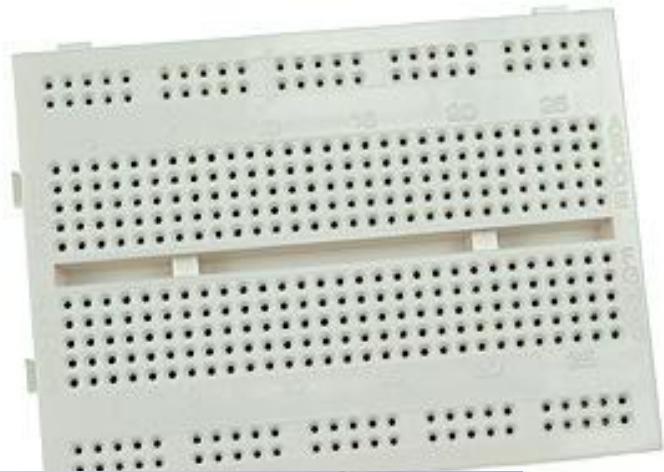
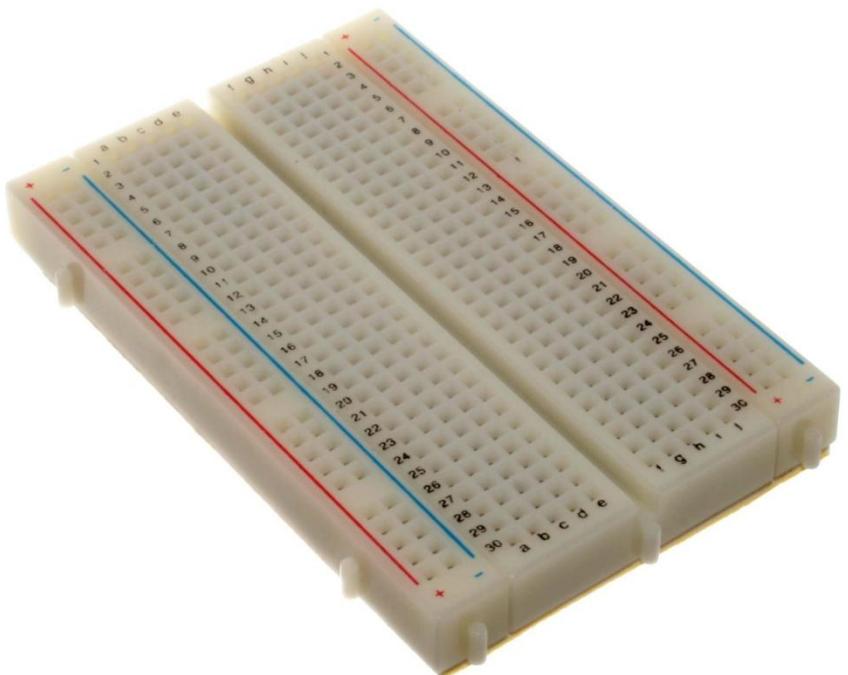


USB Type C

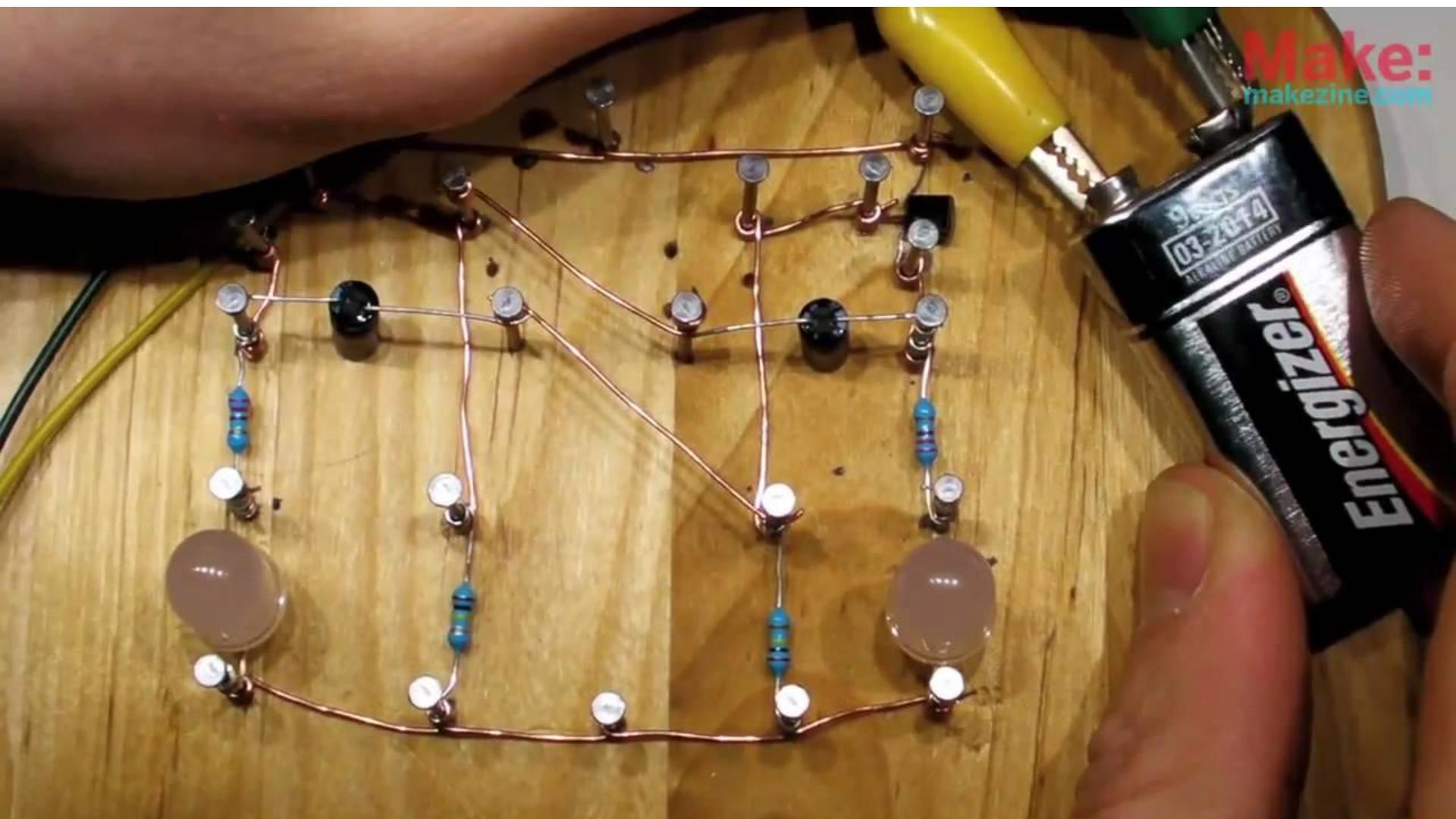
12 x 8
LED Matrix

Analog Input
(ADC)

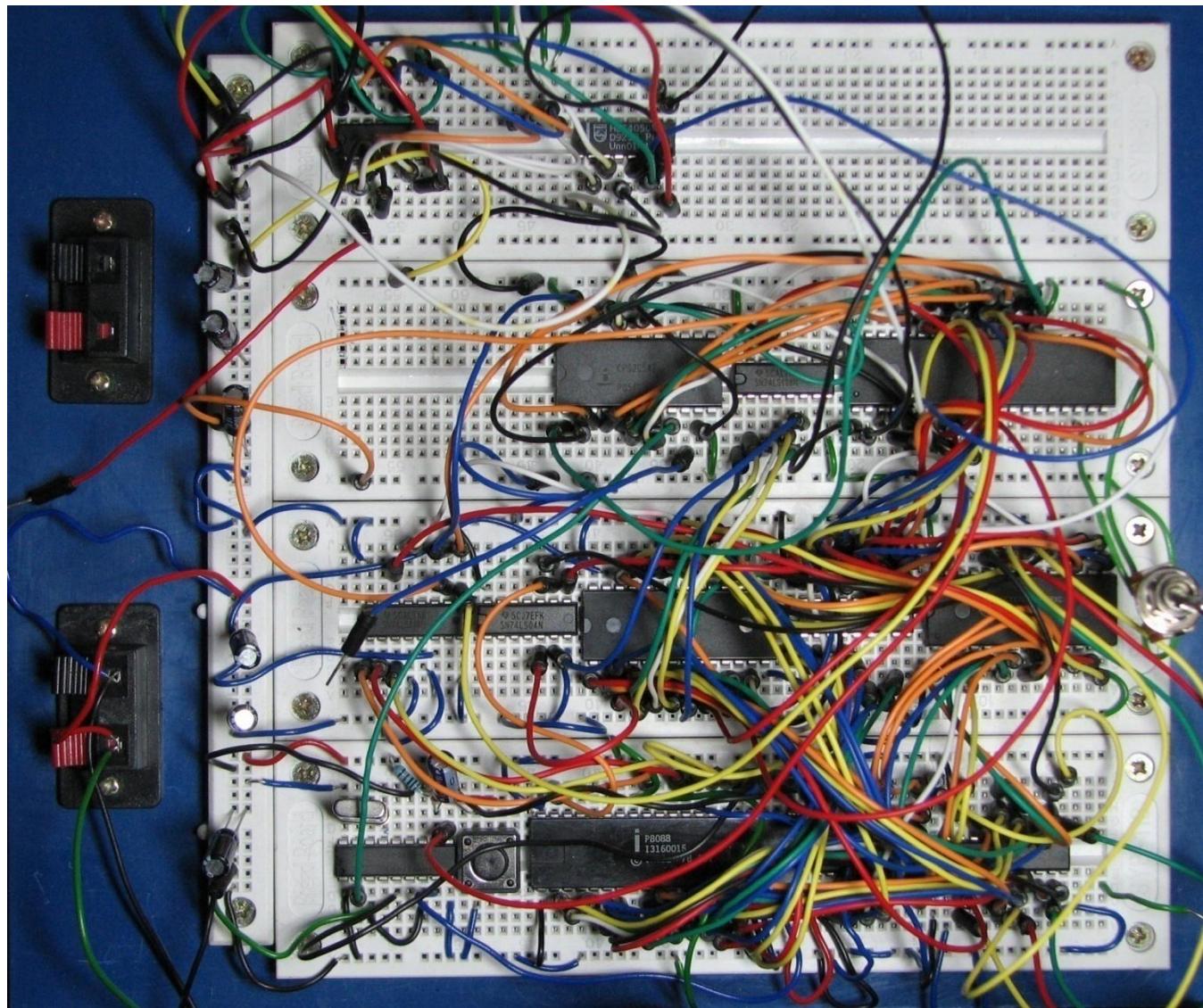
What is a Breadboard?



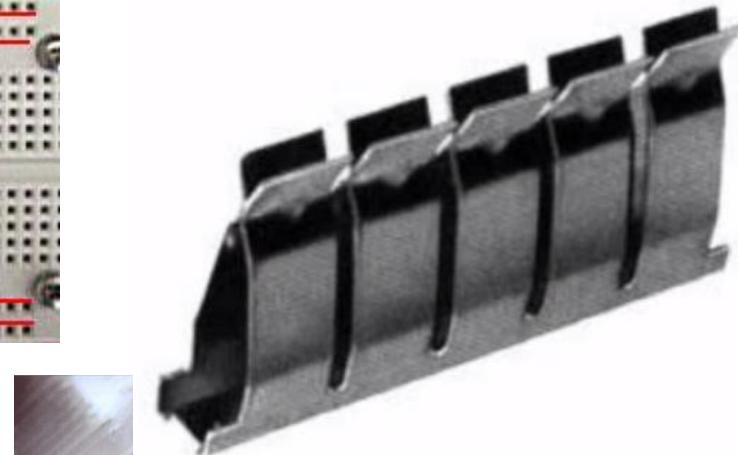
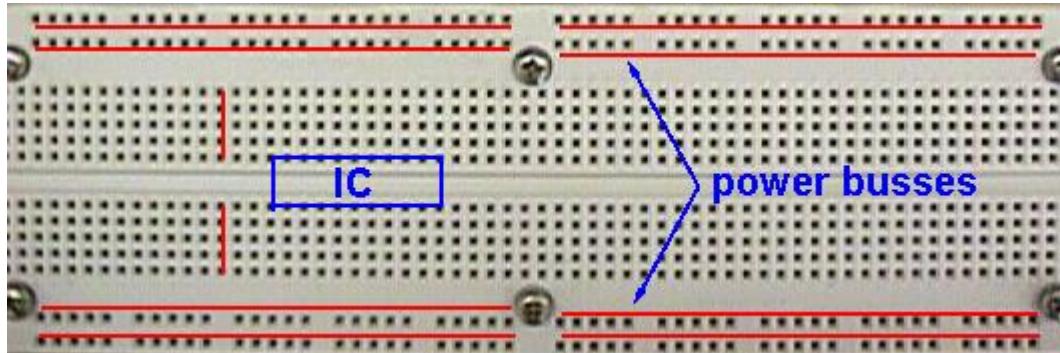
Make:
makezine.com



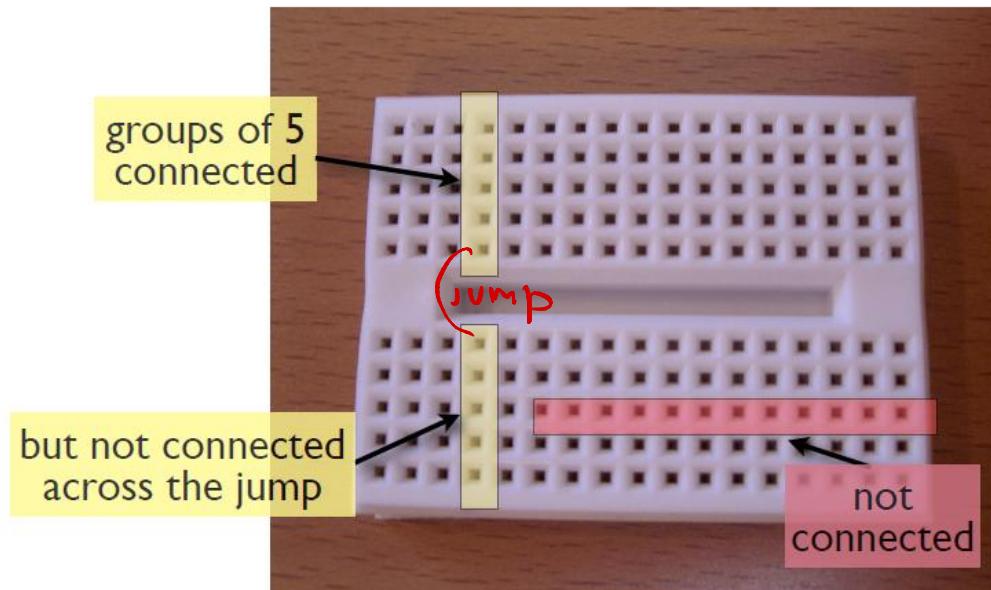
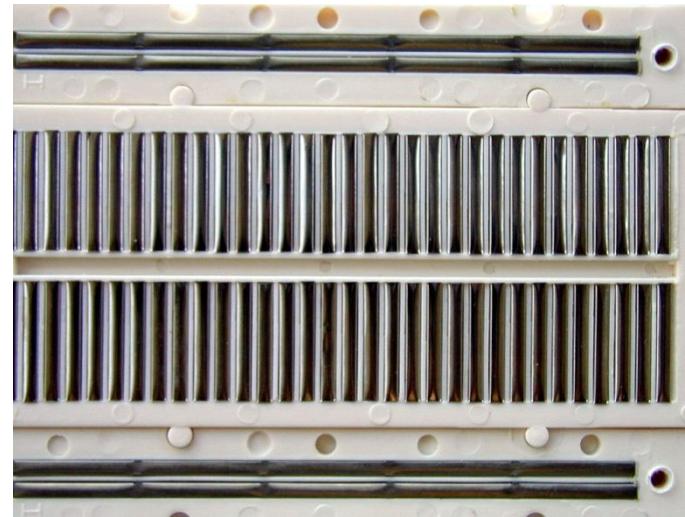
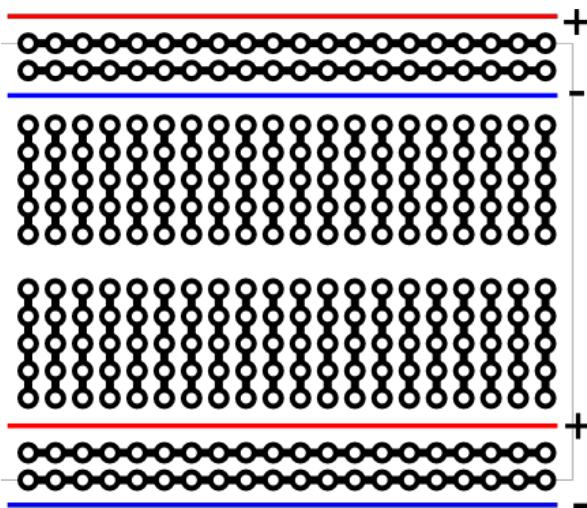
What circuit are we going to breadboard today?



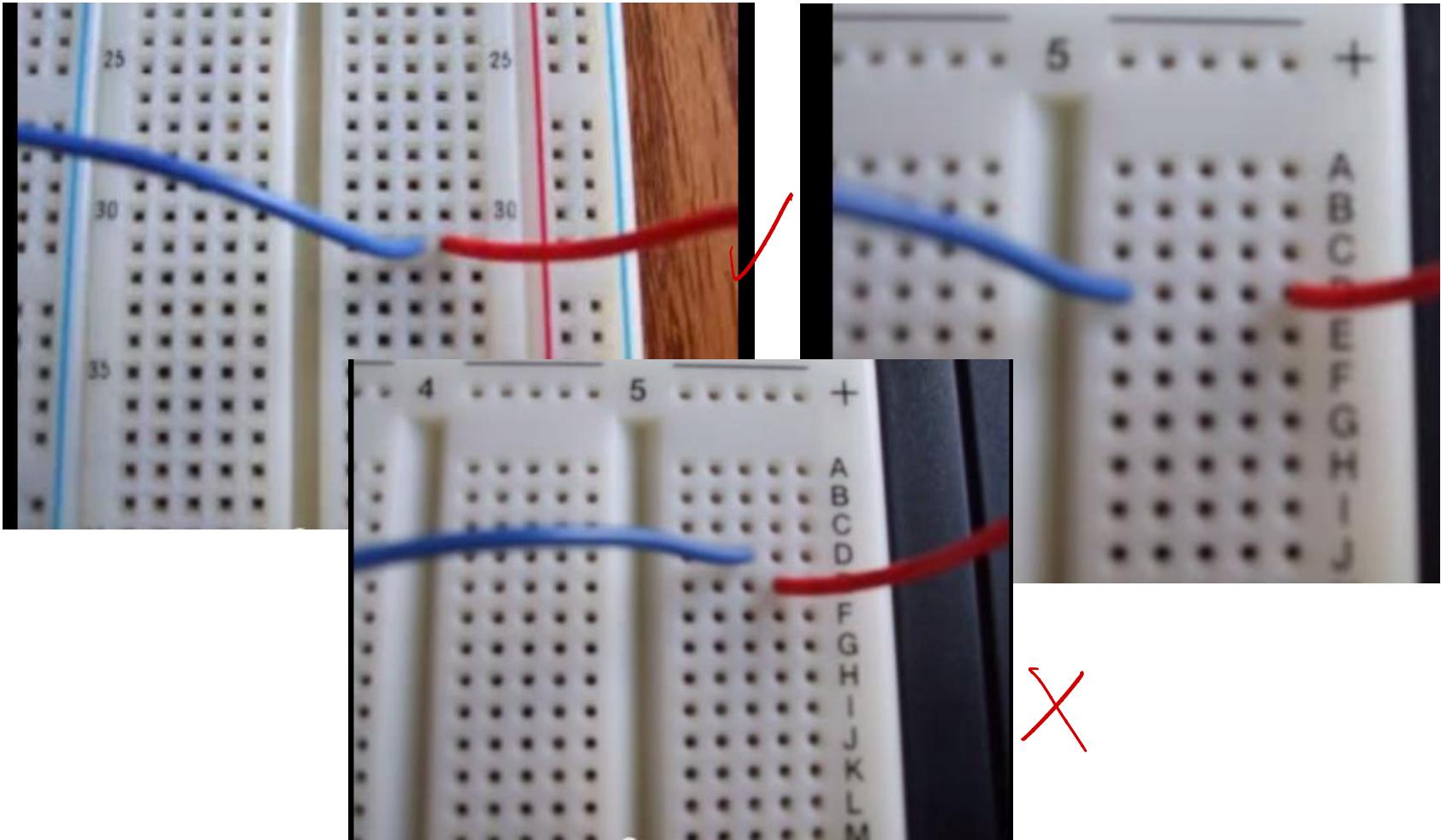
How does a breadboard work?



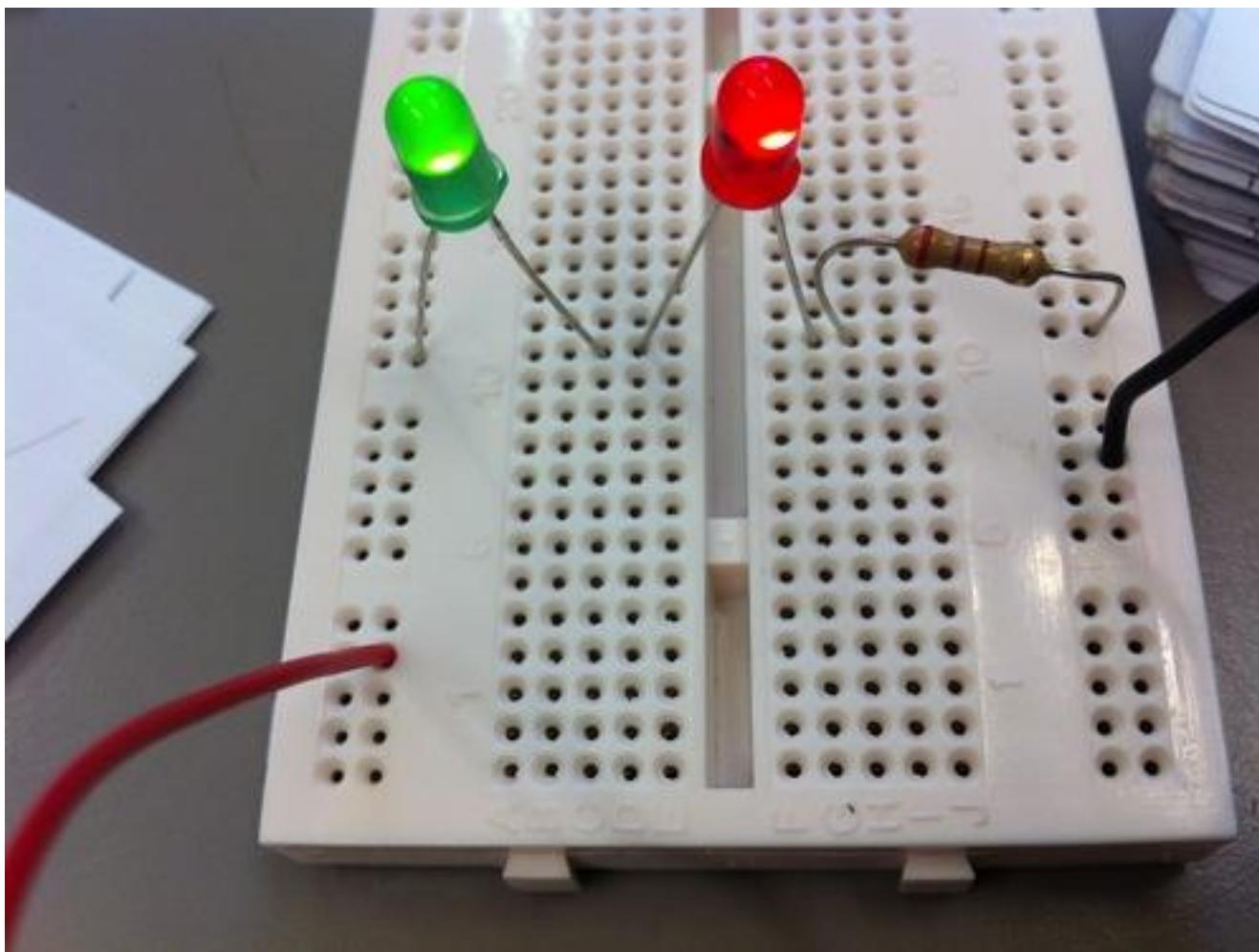
How does a breadboard work?



Are these wires connected?

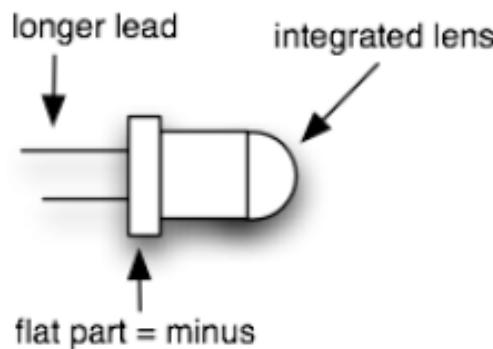


How to use a breadboard

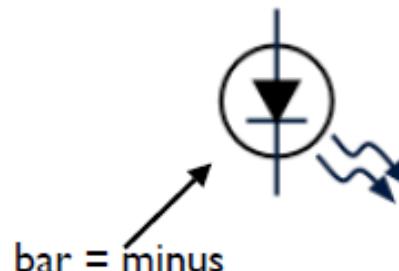


LEDs

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

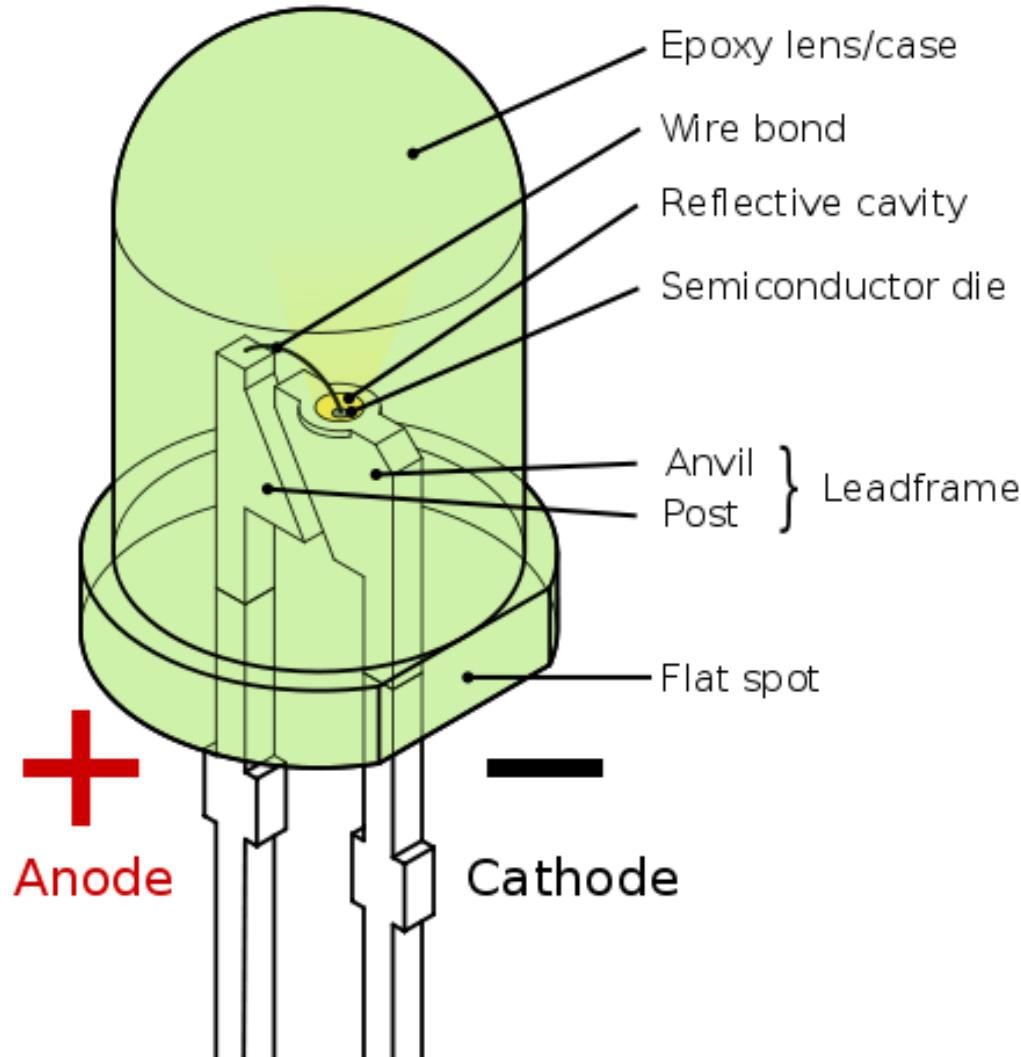


physical characteristics



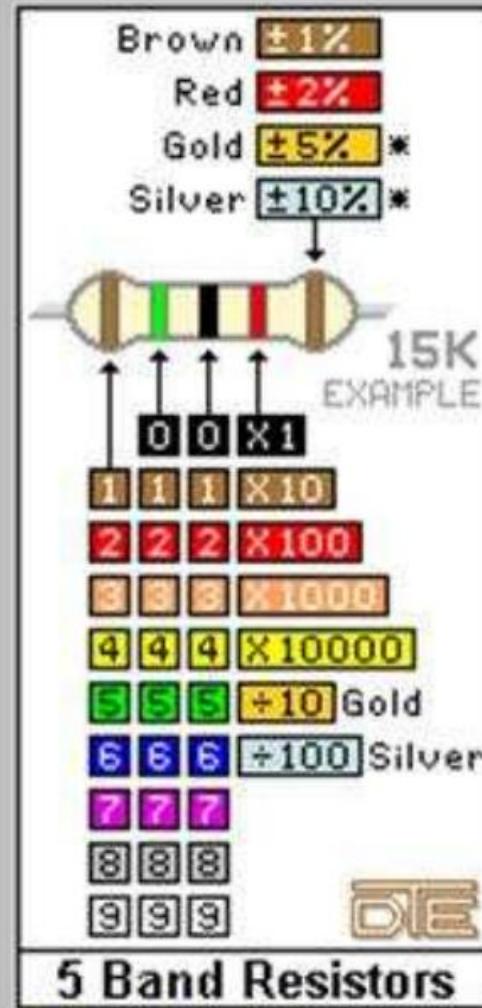
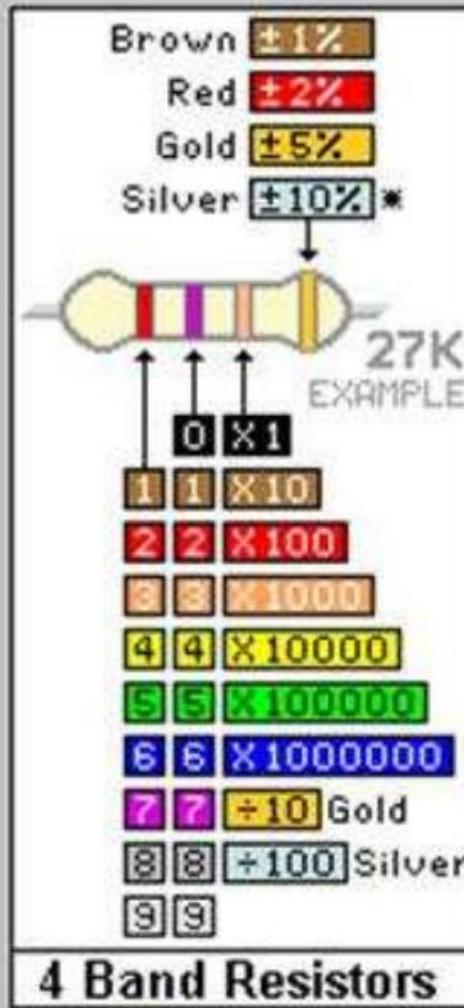
schematic symbol

What are LEDs?



การอ่านค่า ตัวต้านทาน

0 1 2 3 4 5 6 7 8 9
0 Black
1 Brown
2 Red
3 Orange
4 Yellow
5 Green
6 Blue
7 Purple
8 Grey
9 White
$\pm 5\%$ Gold
$\pm 10\%$ Silver
Colour Codes



These colour codes and their uses are explained in the text

Laboratory

การทดลองที่ 1 Getting Start with Arduino

1. จัดเตรียม Hardware และ Software เพื่อการพัฒนาโปรแกรมโดยแบ่งเป็น 3 ขั้นตอนดังนี้
 - 1.1 ติดตั้งโปรแกรม Arduino IDE
 - 1.2 ต่อบอร์ด Arduino Uno เข้ากับ Computer ด้วยสาย Mini USB
2. ทดสอบ บอร์ด Arduino Uno และโปรแกรม ว่า สามารถใช้งานได้จริง

1.1 Installing Arduino IDE

- **Download the Arduino environment**

<http://arduino.cc/en/Main/Software>

โดยเลือก Version ล่าสุด ในที่นี่คือ Arduino 2.3.6

และเลือกตามระบบปฏิบัติการ ที่เราใช้ (Mac, Windows,...)

- **Install the Software**

Windows

- Unzip โปรแกรม ไปที่ folder ที่ต้องการติดตั้ง

MAC OS

- Copy the Arduino application into the Applications folder

Downloads



Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits

Windows MSI installer

Windows ZIP file

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

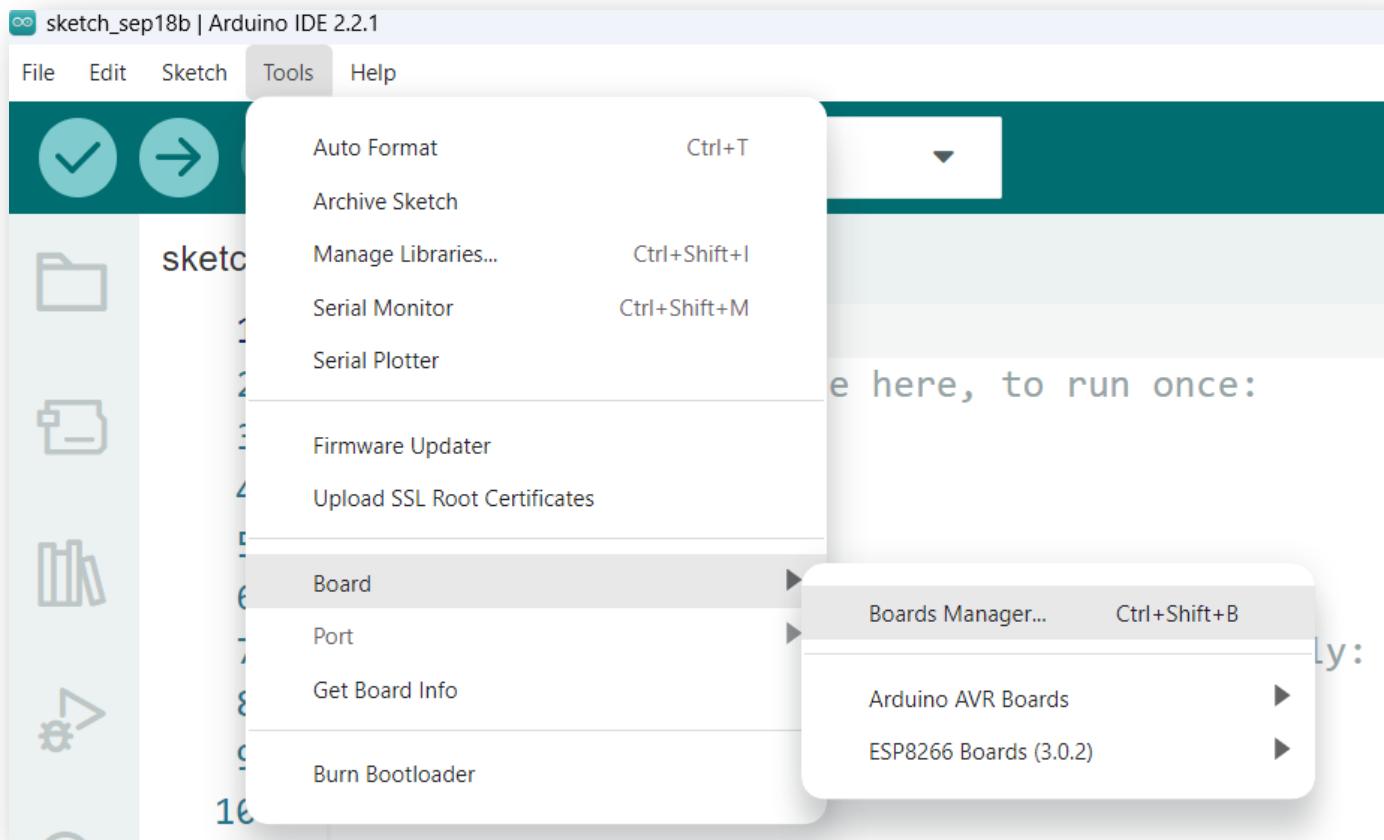
macOS Intel, 10.15: "Catalina" or newer, 64 bits

macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

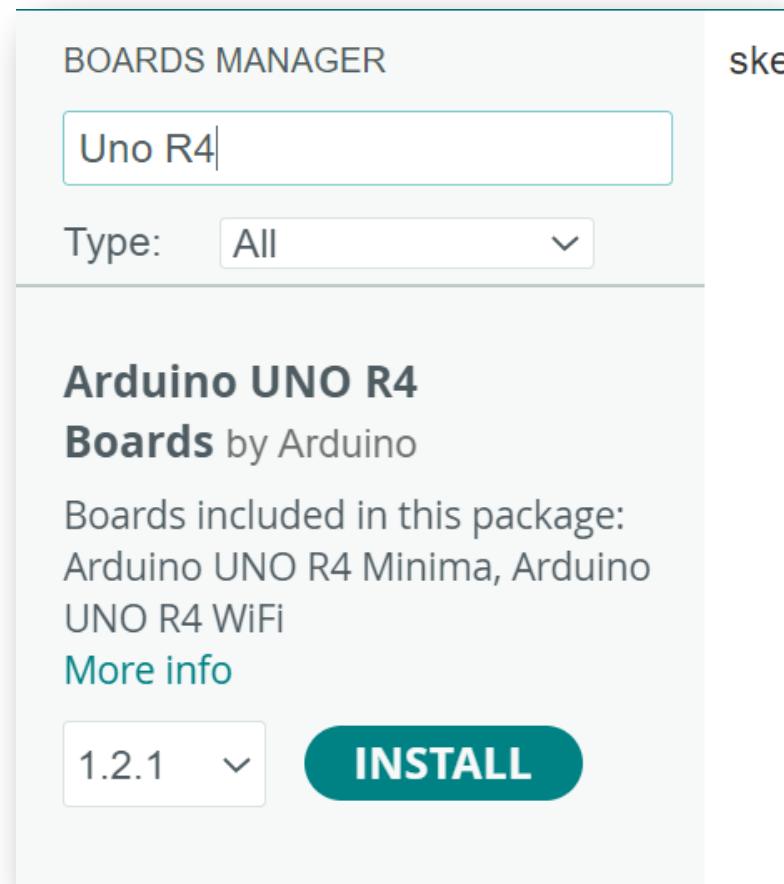
ติดตั้ง Arduino IDE ให้รู้จัก Arduino Uno R4 WiFi

ไปที่เมนู Tools และไปที่ Board Manager



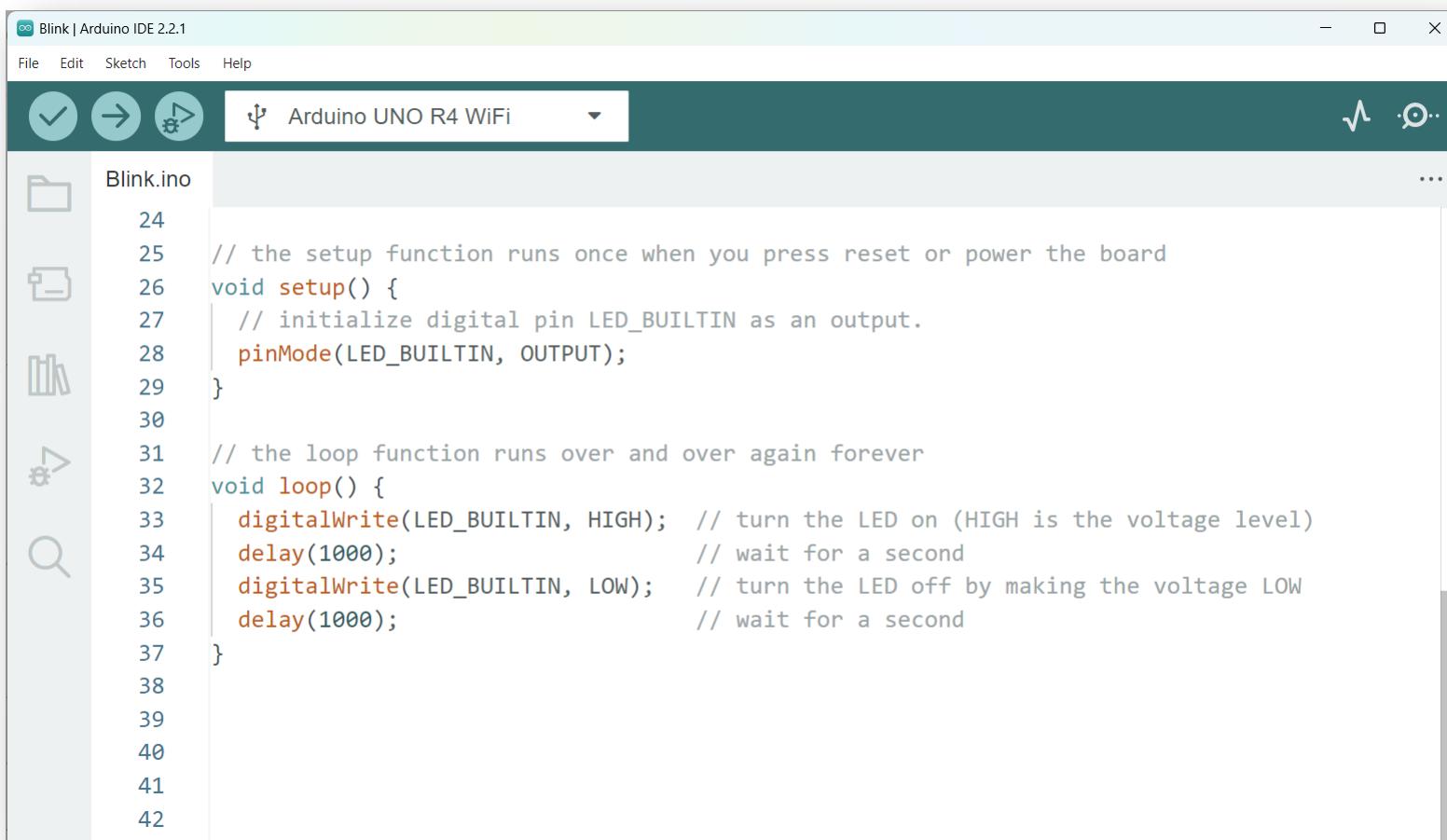
ติดตั้ง Arduino IDE ให้รู้จัก Arduino Uno R4 WiFi

ค้นหา UNO R4 จากนั้นกด **INSTALL**



1.2 Test the Arduino board

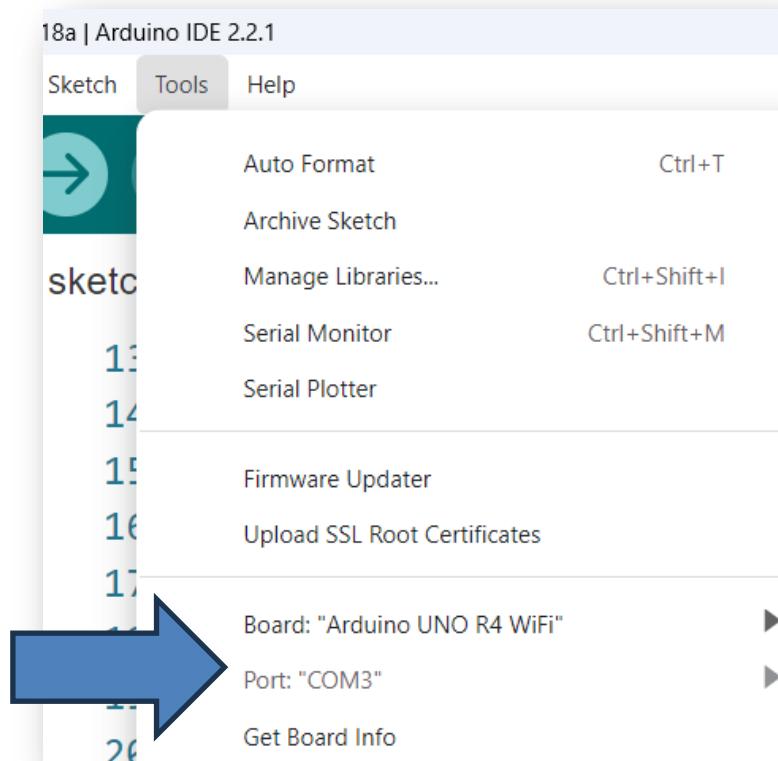
- เปิดโปรแกรมตัวอย่าง File > Examples > 01.Basics > Blink



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino IDE 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar features icons for save, build, upload, and refresh. The board selector dropdown shows "Arduino UNO R4 WiFi". The code editor displays the "Blink.ino" sketch:

```
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
34     delay(1000);                      // wait for a second
35     digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
36     delay(1000);                      // wait for a second
37 }
38
39
40
41
42
```

- **Select your board** : เลือกเป็น **Arduino UNO R4 WiFi**
- **Select your Serial port** : เลือกเป็นพอร์ตที่ติดตั้งไว้ ปกติจะเป็น **COM3**



- **Compile your program** โดยการกดปุ่ม **verify**

**compile
(verify)**

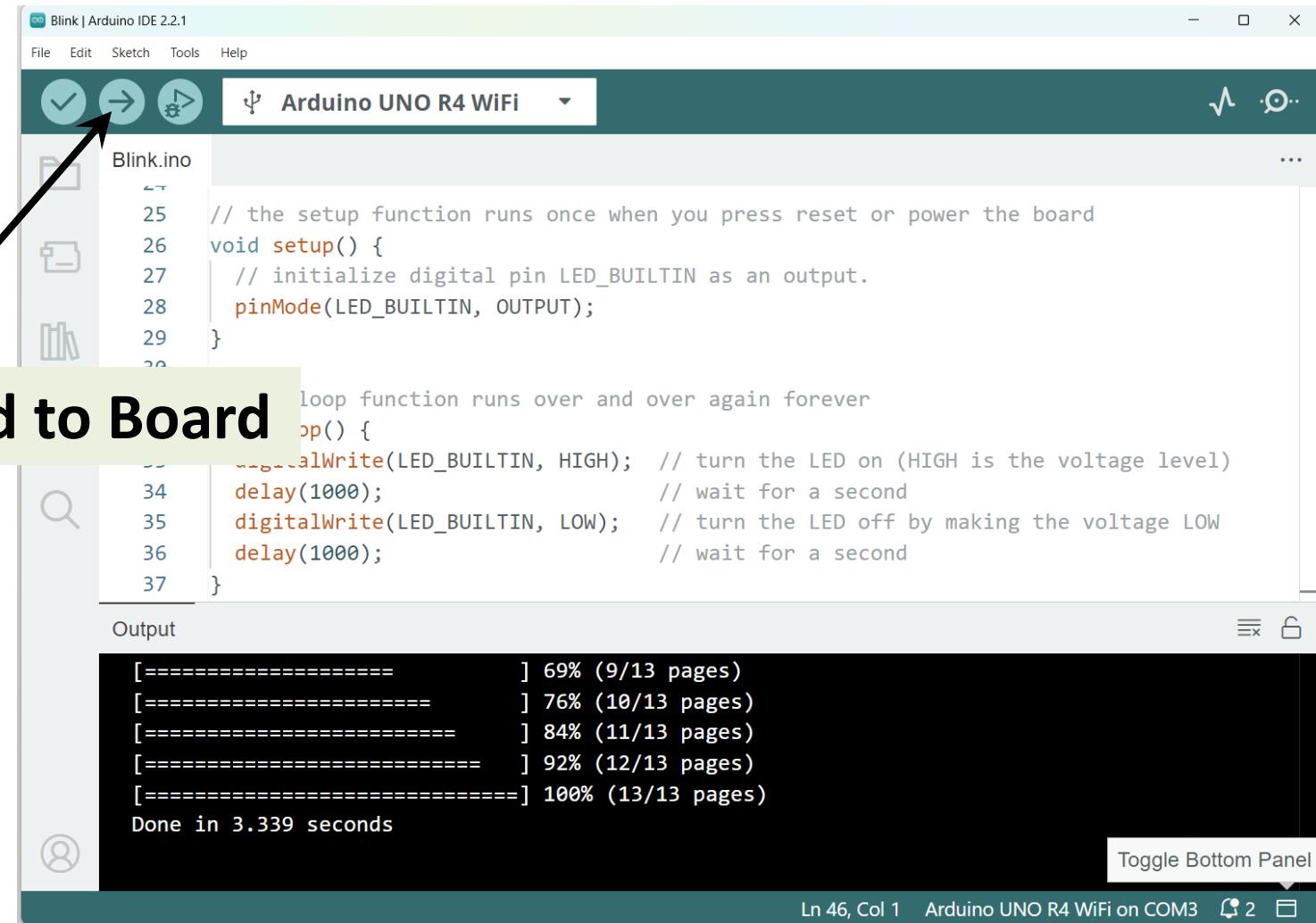
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Blink | Arduino IDE 2.2.1
- Sketch Selection:** Arduino UNO R4 WiFi
- Code Editor:** Displays the `Blink.ino` sketch:

```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
34     delay(1000);                      // wait for a second
35     digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
36     delay(1000);                      // wait for a second
37 }
```
- Status Area:** A white box with the text "Status Area" overlaid on the bottom right of the code editor.
- Output Window:** Shows the compilation results:

```
Sketch uses 52256 bytes (19%) of program storage space. Maximum is 262144 bytes.
Global variables use 6744 bytes (20%) of dynamic memory, leaving 26024 bytes for local variables.
```
- Toolbar:** Shows icons for Open, Save, Verify, and Run.

- Upload the program
 - เมื่อสถานะเป็น Done ไฟ LED ที่บอร์ด UNO R4 จะกระพริบติด-ดับ

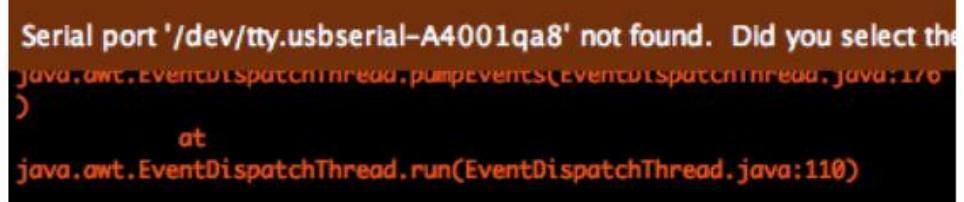


1.3 Status Message

Uploading worked

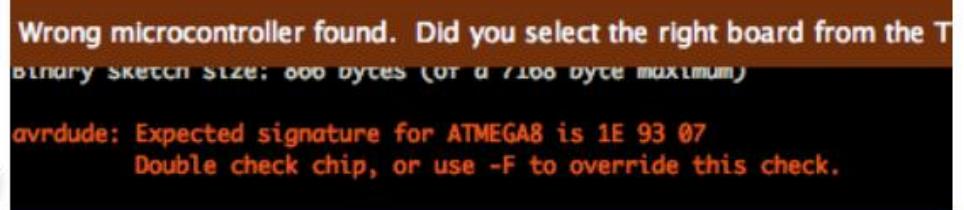


Wrong serial port selected



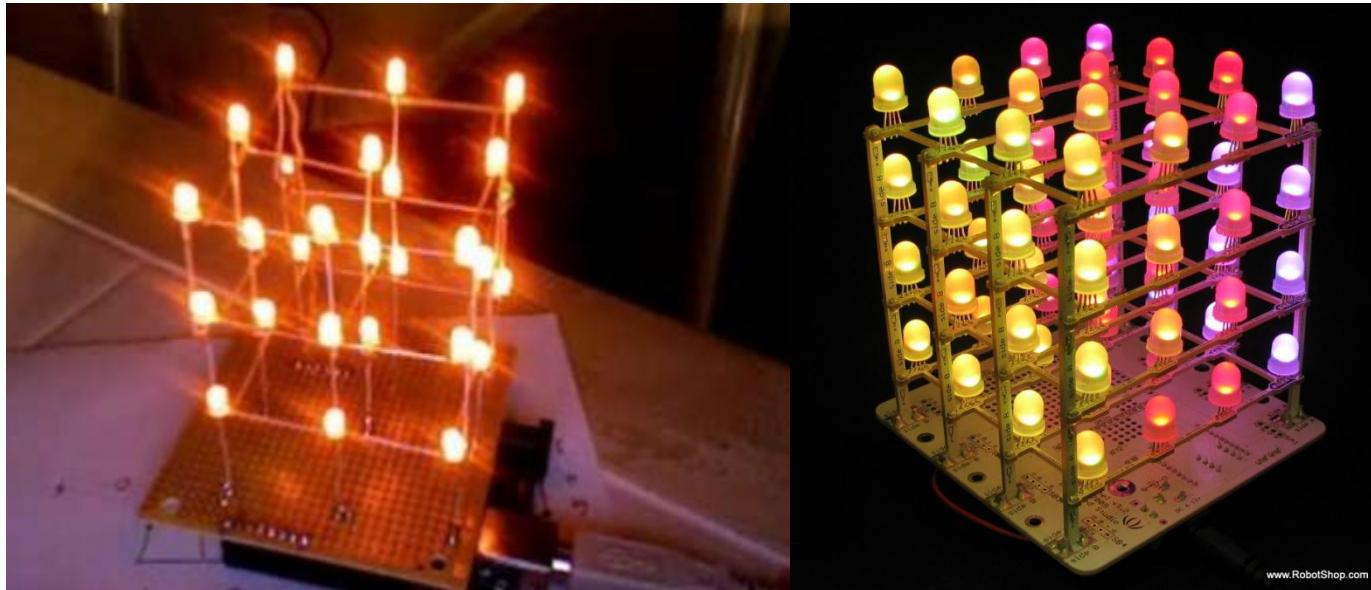
Wrong board selected

nerdy cryptic error messages



4. การส่งข้อมูลออกพอร์ต

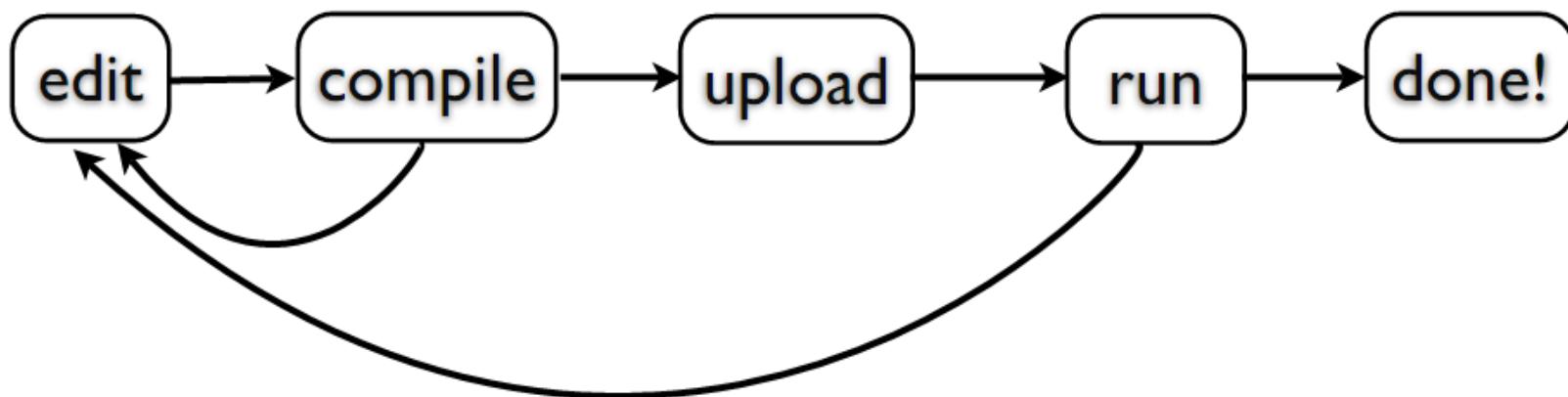
Digital output



arduino cube

1. ขั้นตอนการพัฒนาโปรแกรมด้วย Arduino (Development Circle)

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



2. 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
- **analogWrite ()**
 - write an “analog” value
- **analogRead ()**
 - read an analog pin
- **delay ()**
 - wait an amount of time
- **millis ()**
 - get the current time

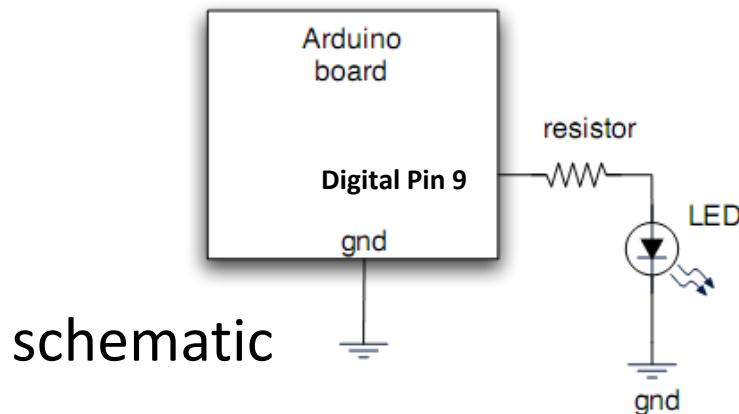
3. การเขียนโปรแกรมบน Arduino

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

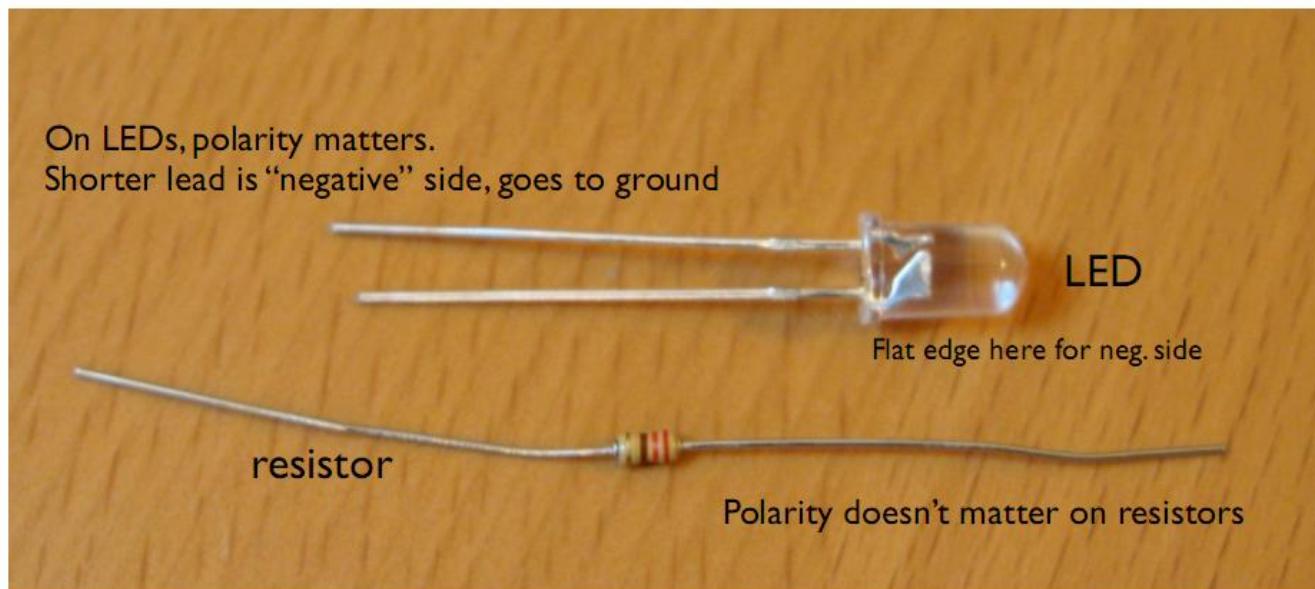
- Declare variables at top
- Initialize
 - `setup()` – run once at beginning, set pins
- Running
 - `loop()` – run repeatedly, after `setup()`

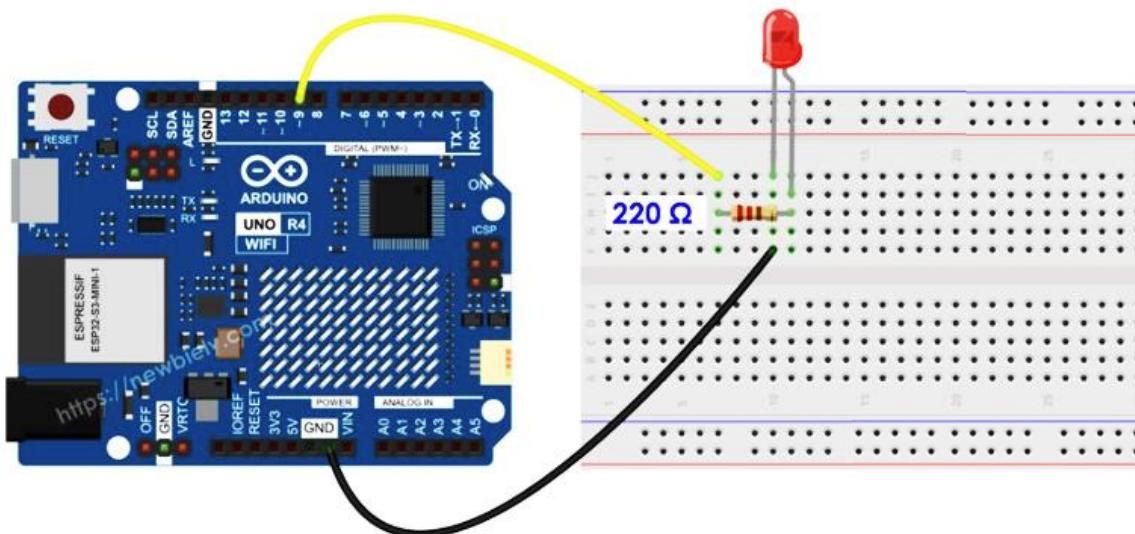
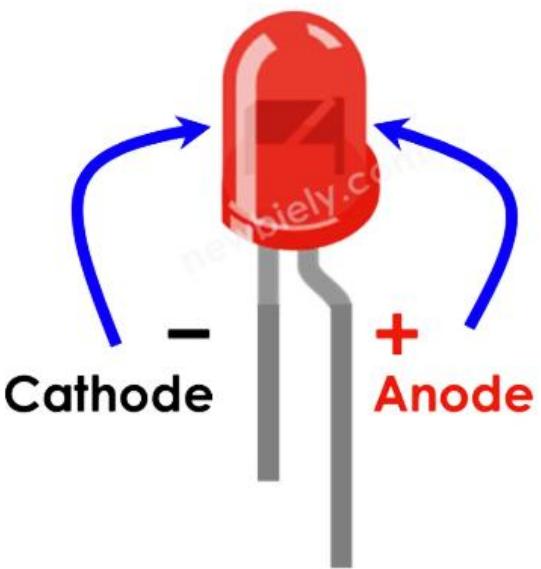
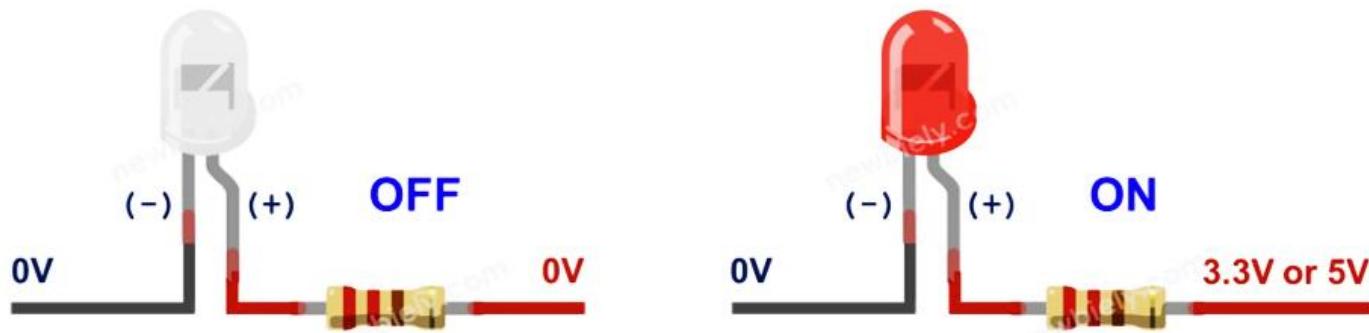
4. การส่งค่าอุปกรณ์แบบดิจิตอล

การทดลองที่ 1. LED Blink : Hello world of Microcontroller



To turn on LED use
`digitalWrite(9, HIGH)`





บุณยรัตน์ กุลวิชล

โปรแกรม LED Blink

```
void setup() {  
    // initialize digital pin LED_BUILTIN as an output.  
  
    pinMode(9, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
  
void loop() {  
    digitalWrite(9, HIGH);    // turn the LED on (HIGH is the voltage level)  
    delay(1000);             // wait for a second  
    digitalWrite(9, LOW);     // turn the LED off by making the voltage LOW  
    delay(1000);             // wait for a second  
}
```

แบบฝึกหัดที่ 1.1 โปรแกรมควบคุม Color LED

จงเขียนโปรแกรมแสดงสี 7 สี โดยเว้นช่วงสีละ 1 วินาที
กำหนดสีเอง ตามใจชอบ

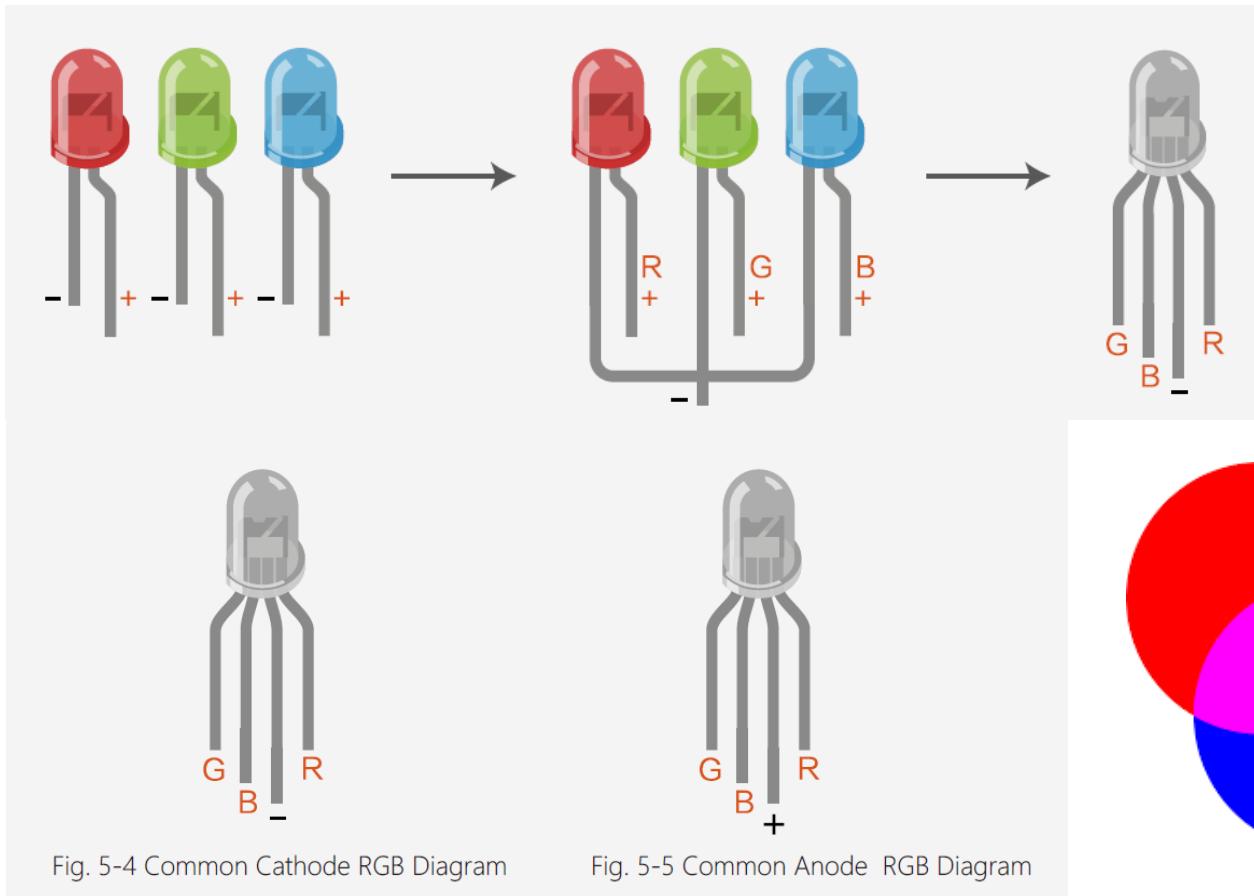
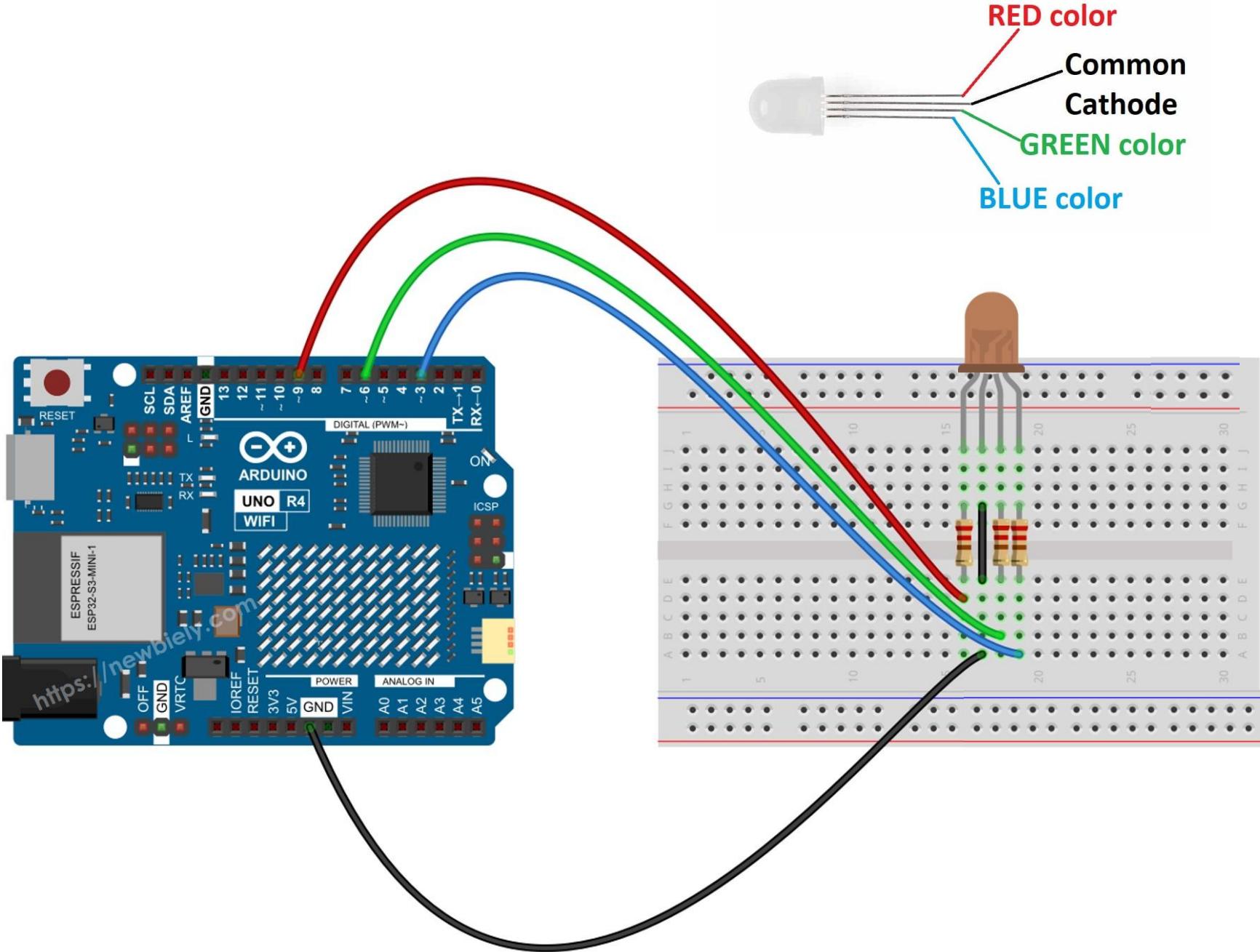


Fig. 5-4 Common Cathode RGB Diagram

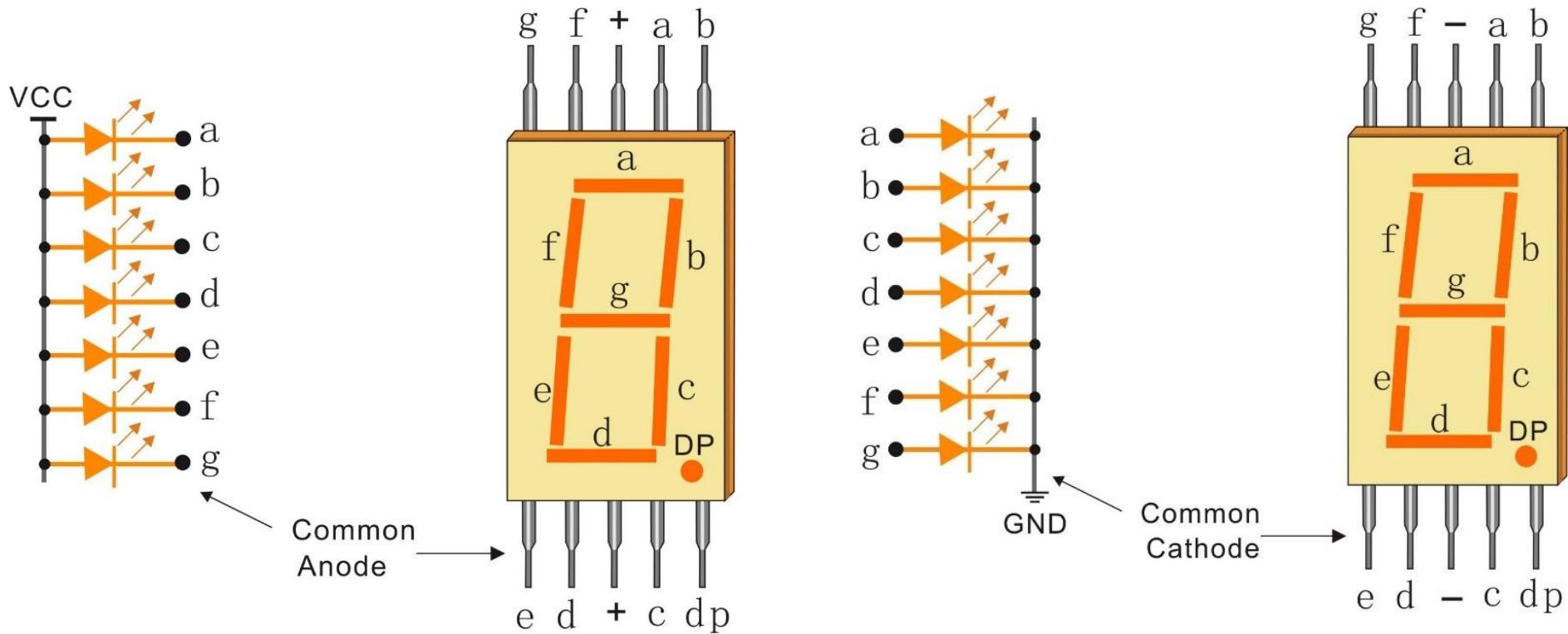
Fig. 5-5 Common Anode RGB Diagram

Common Cathode

Common Anode



ประเภทของ 7-segment



แบบฝึกหัดที่ 1.2 โปรแกรมควบคุม 7-segment

- ดัดแปลงวงจรไฟวิ่งในการทดลองที่ 2

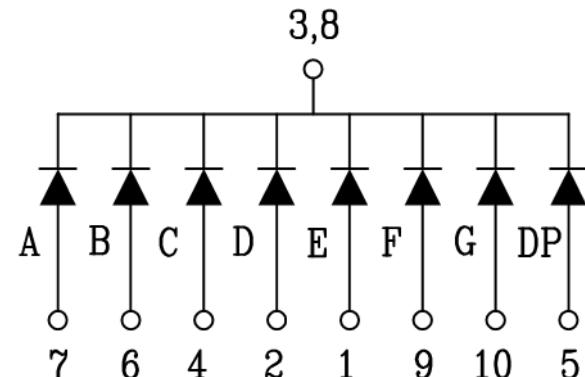
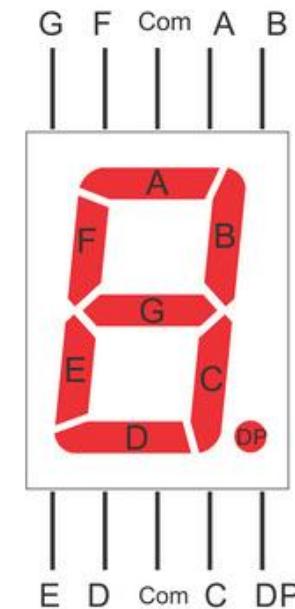
โดยแทน LED ด้วย 7-segment

ATMega328P and Arduino Uno Pin Mapping

Arduino function		Arduino function
reset	(PCINT14/RESET) PC6	1
digital pin 0 (RX)	(PCINT16/RXD) PD0	2
digital pin 1 (TX)	(PCINT17/TXD) PD1	3
digital pin 2	(PCINT18/INT0) PD2	4
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5
digital pin 4	(PCINT20/XCK/T0) PD4	6
VCC	VCC	7
GND	GND	8
crystal	(PCINT6/XTAL1/TOSC1) PB6	9
crystal	(PCINT7/XTAL2/TOSC2) PB7	10
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12
digital pin 7	(PCINT23/AIN1) PD7	13
digital pin 8	(PCINT0/CLKO/CP1) PB0	14
		28
	PC5 (ADC5/SCL/PCINT13)	PC analog input 5
	27	PC4 (ADC4/SDA/PCINT12)
	26	PC3 (ADC3/PCINT11)
	25	PC2 (ADC2/PCINT10)
	24	PC1 (ADC1/PCINT9)
	23	PC0 (ADC0/PCINT8)
	22	GND
	21	AREF
	20	AVCC
	19	PB5 (SCK/PCINT5)
	18	PB4 (MISO/PCINT4)
	17	PB3 (MOSI/OC2A/PCINT3)
	16	Digital pin 11(PWM)
	15	PB2 (SS/OC1B/PCINT2)
		Digital pin 10 (PWM)
		Digital pin 9 (PWM)

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI.
MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-
impedance loads on these pins when using the ICSP header.

- จงเขียนโปรแกรมแสดง 0-9 โดยเว้น
ช่วงละ 1 วินาที

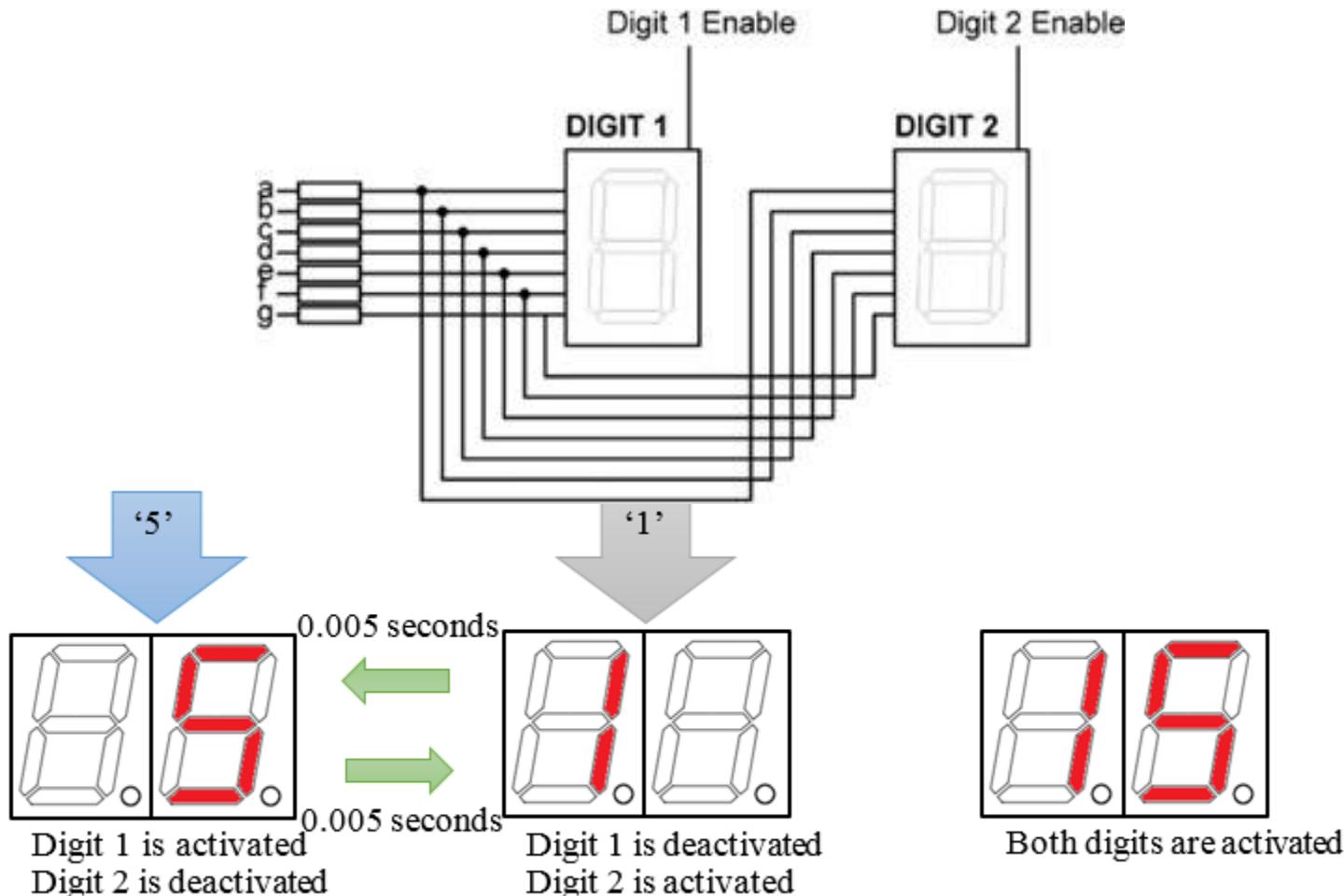


การจัดขาของ 7-segment

แบบฝึกหัดที่ 1.3 โปรแกรมควบคุม 7-segment

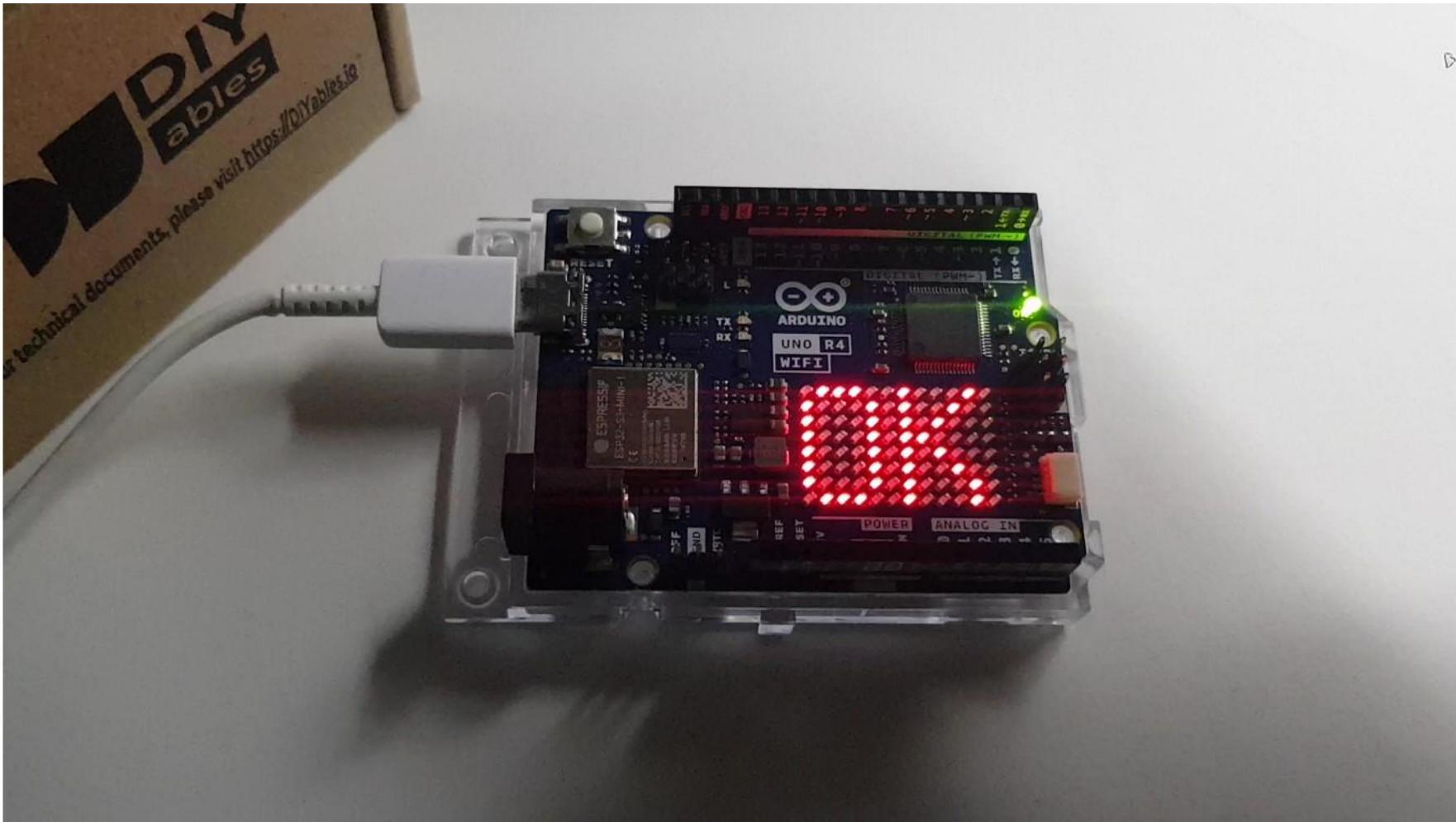
จงเขียนโปรแกรมแสดงเลขทีละ 2 หลักท้ายของรหัสนักศึกษา

เช่นรหัส 67070012 ก็แสดง 67 07 00 12 ช่วงละ 1 วินาที



แบบฝึกหัดที่ 1.4 โปรแกรมควบคุม LED Matrix

- จงเขียนโปรแกรมแสดงเลขทีละ 2 หลักท้ายของรหัสนักศึกษา



The END

Next week

Digital Input, Analog Input, Analog Output

