

# Principles and Applications of Microcontrollers

---

Yan-Fu Kuo

Dept. of Biomechatronics Engineering

National Taiwan University

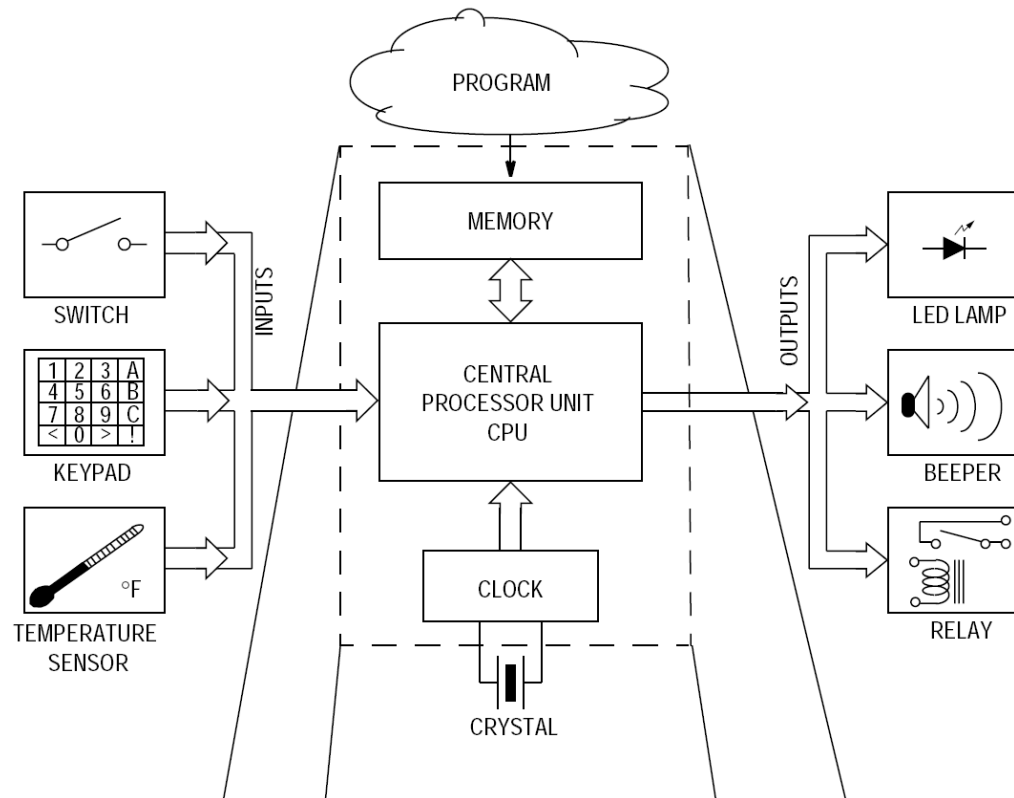
Today:

- Introduction of Arduino
- Arduino programming
- Arduino digital I/O



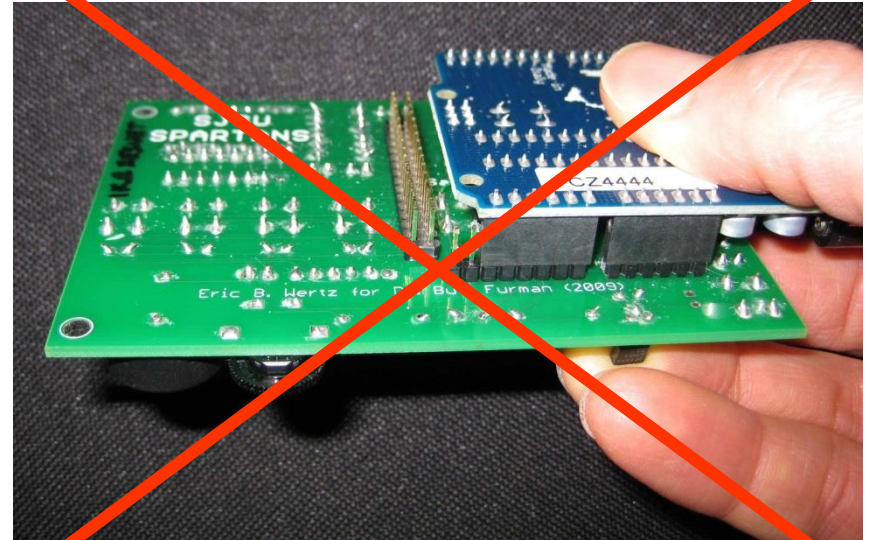
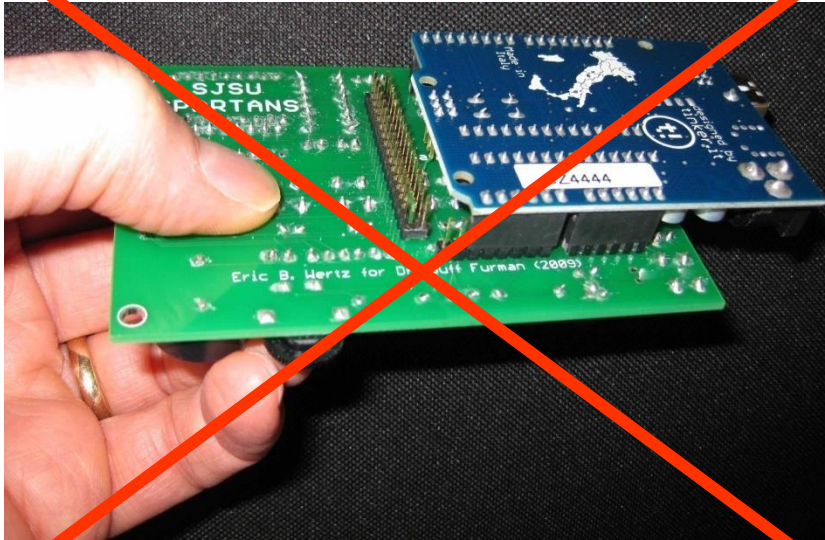
# What is A Microcontroller?

- A small computer on a single integrated circuit
- Containing a processor, memory, and programmable input/output peripherals



# Handling Arduino – Do NOT Do This!

Improper Handling - **NEVER!!!**



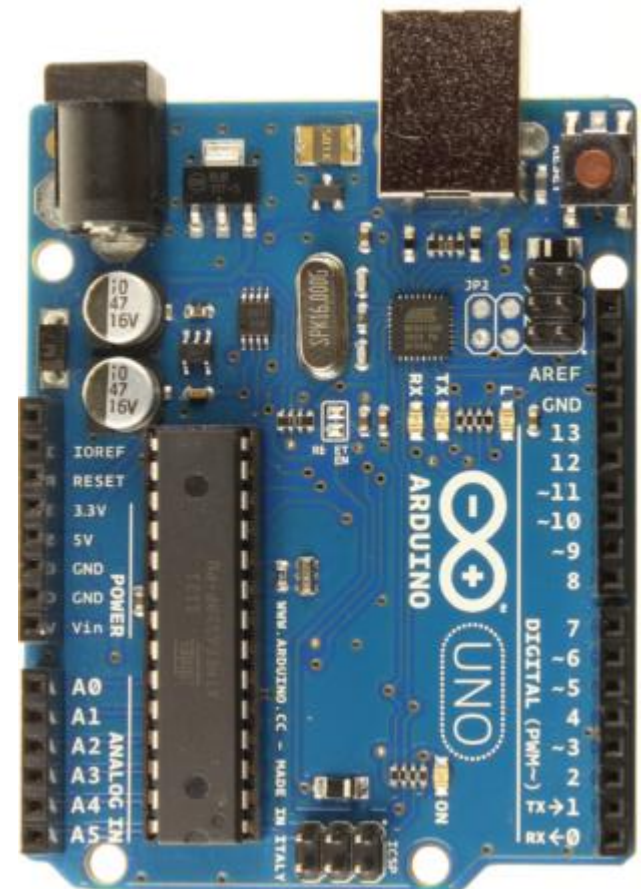
# Arduino Uno

- An integrated microcontroller
- Atmel ATmega328p

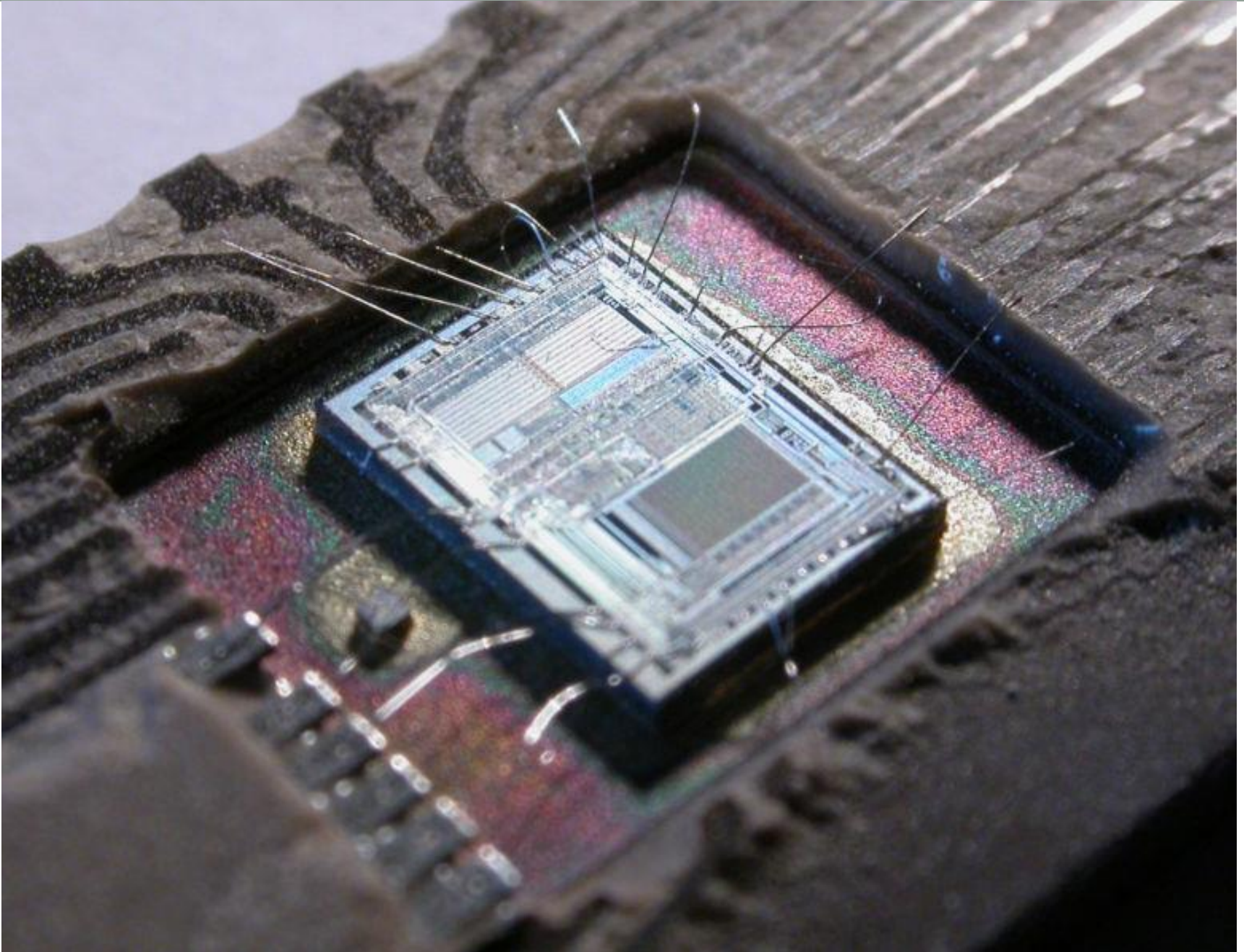


Intel 80286

=







# Why Arduino?

[All](#)[Images](#)[Videos](#)[News](#)[Maps](#)[More](#)[Settings](#)[Tools](#)

About 135,000,000 results (0.47 seconds)

台灣正式代理Arduino 原廠開發板 | Arduino 美國原廠最新原裝進口

**Ad** [www.taiwansensor.com.tw/](http://www.taiwansensor.com.tw/) ▼ 06 222 1898

台灣智能感測科技提供**Arduino**原廠零件現貨供應，輕鬆申請產品報價。不限最小訂購量。

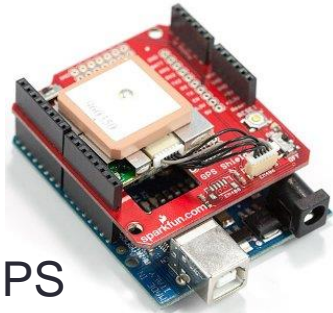
[Arduino 原裝開發板](#) · [Arduino Shield 功能擴展板](#)

- It has dominated the market
- Open source
- Low-cost – less than \$30 USD
- Easy to use – less hardware logic
- Friendly programming environment





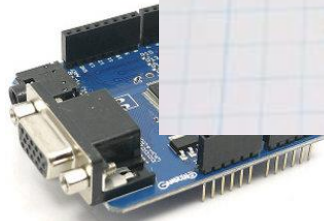
# Arduino Shields



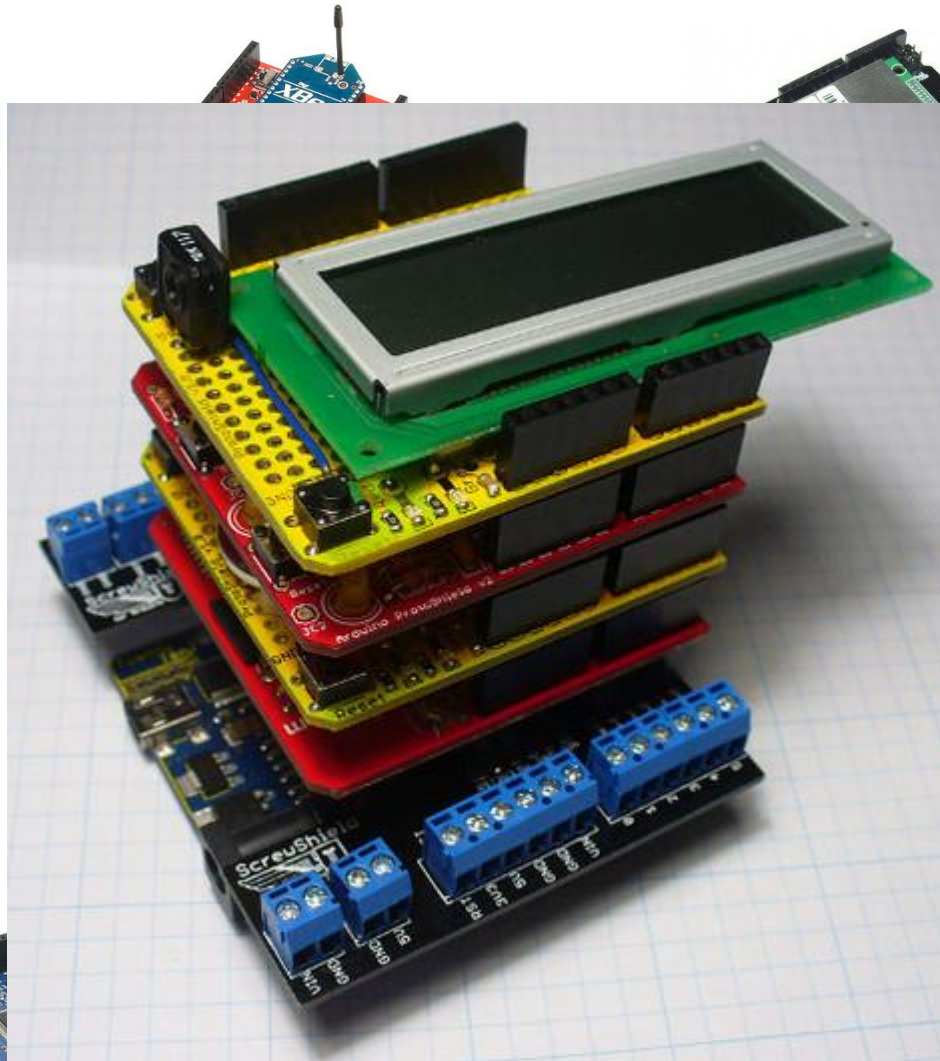
GPS



Ethernet



VGA



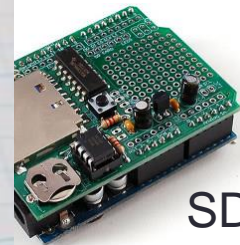
Motor  
controller



Wi-Fi



Joystick



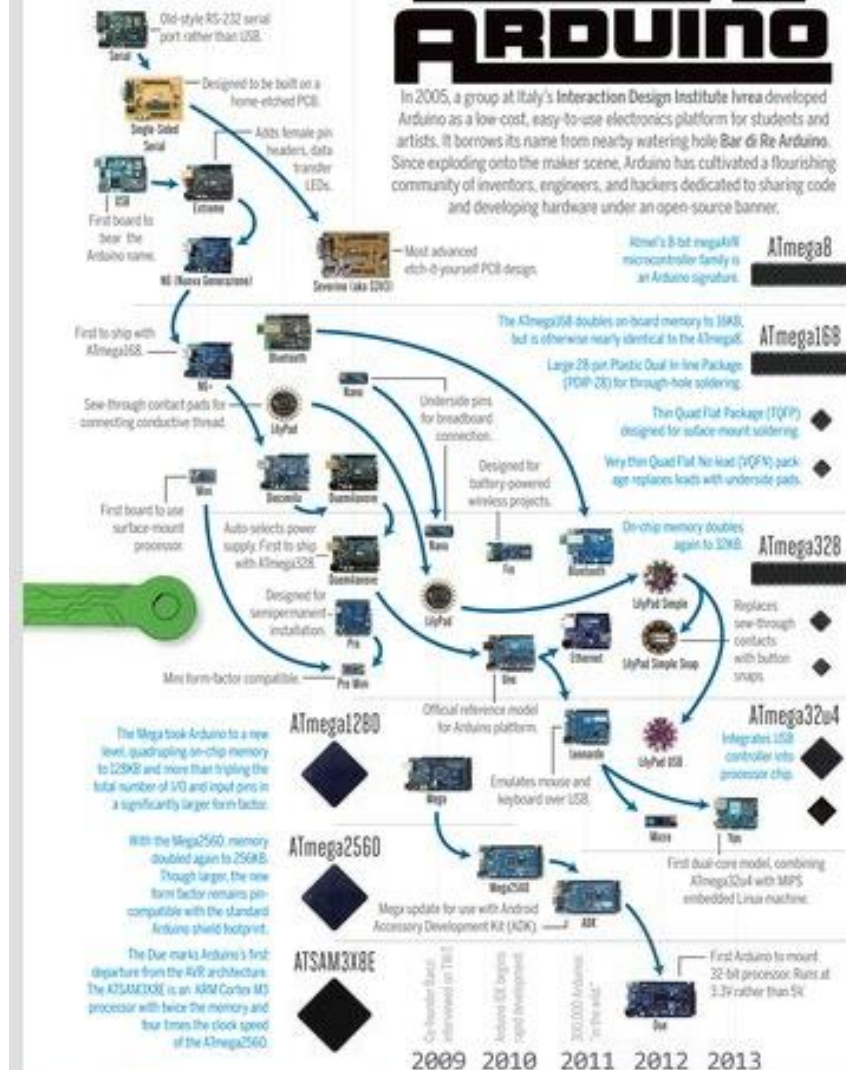
SD card  
logger

## SPECIAL BOARD FIELD GUIDE

2005 2006 2007 2008

## EVOLUTION OF ARDUINO

In 2005, a group at Italy's Interaction Design Institute I'veea developed Arduino as a low-cost, easy-to-use electronics platform for students and artists. It borrows its name from nearby watering hole Bar di Re Arduino. Since exploding onto the maker scene, Arduino has cultivated a flourishing community of inventors, engineers, and hackers dedicated to sharing code and developing hardware under an open-source banner.



The latest board in the series, the Arduino Leonardo, differs from its predecessors in that, in addition to the virtual serial port necessary to transfer code from the IDE to the board, it can also appear to a connected computer as a USB mouse and keyboard.

## Alternatives to Arduino

The Arduino-and-derivatives phenomenon has driven interesting innovation, and convergence, in the microcontroller marketplace.

## The LaunchPad MSP430

The Texas Instruments MSP430 is very similar to the Atmel ATmega microcontroller chip. Notable differences include a very low price point, as well as some interesting refinements for low power consumption. It's also readily available in the through-hole DIP form factor, while dual-inline-packaged ATmega chips often seem to be in short supply. If through-hole mounting is important to you, take a look at the MSP430. The easiest way to get acquainted is to pick up a TI LaunchPad developer board.

The major difference between LaunchPad and Arduino is cost. While a new Uno will run you \$30, and a Leonardo \$25, the LaunchPad MSP430 rings up at just \$10 directly from



## Ever-Shrinking Derivatives

As discussed, the success of the Arduino has led to numerous copies and compatible boards arriving on the market. The crowdfunding site Kickstarter is littered with them, some amazingly successful, some not so much. It'd be impossible to list them all, but there are some that stand out, chiefly because of their size (or lack thereof).

The TinyDuino, for example, is an Arduino-compatible microcontroller using the same processor as the Arduino Uno, but at the size of a U.S. quarter (Figure A). The main processor board includes the microcontroller and supporting circuitry, while the USB and DC power regulators (among other things) have been offloaded to shields. If you don't need them for your project, you don't have to install them. However, despite its size, or more probably because of it, the TinyDuino costs \$20 for the main processor board, plus another \$18 for the USB/ICP programmer shield you're likely to need. Miniaturization doesn't come cheaply.

The DigSpark is another tiny Arduino-compatible board (Figure B). It is built around the ATtiny85 microcontroller, making it much less powerful than the TinyDuino. It only has 6 I/O pins but, on the other hand, it costs just \$9. Like the TinyDuino, it has a variety of interesting shield kits allowing you to easily extend its capabilities.



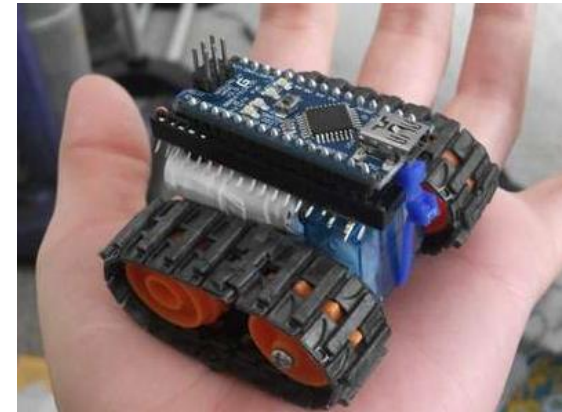
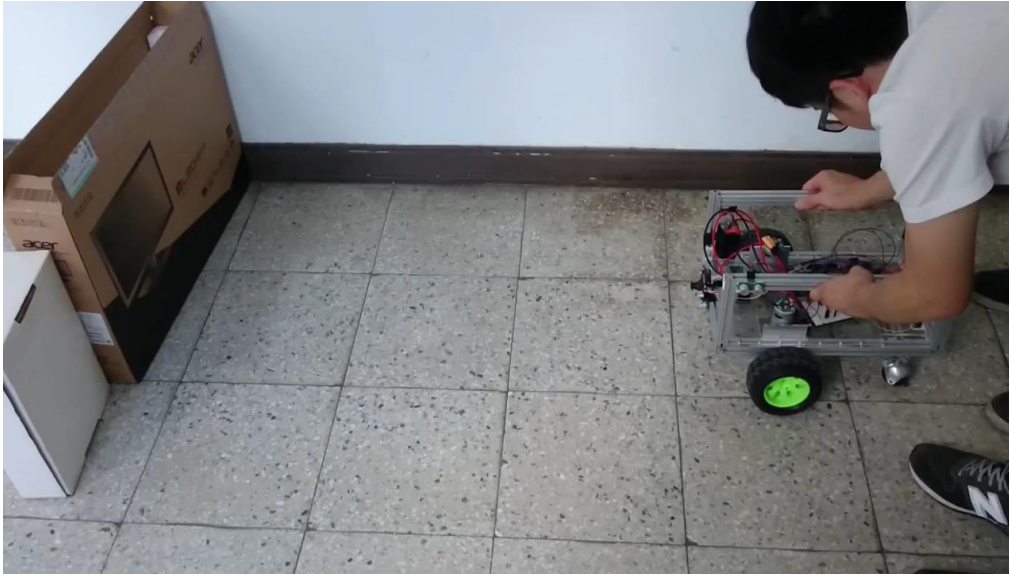
The Arduino Uno



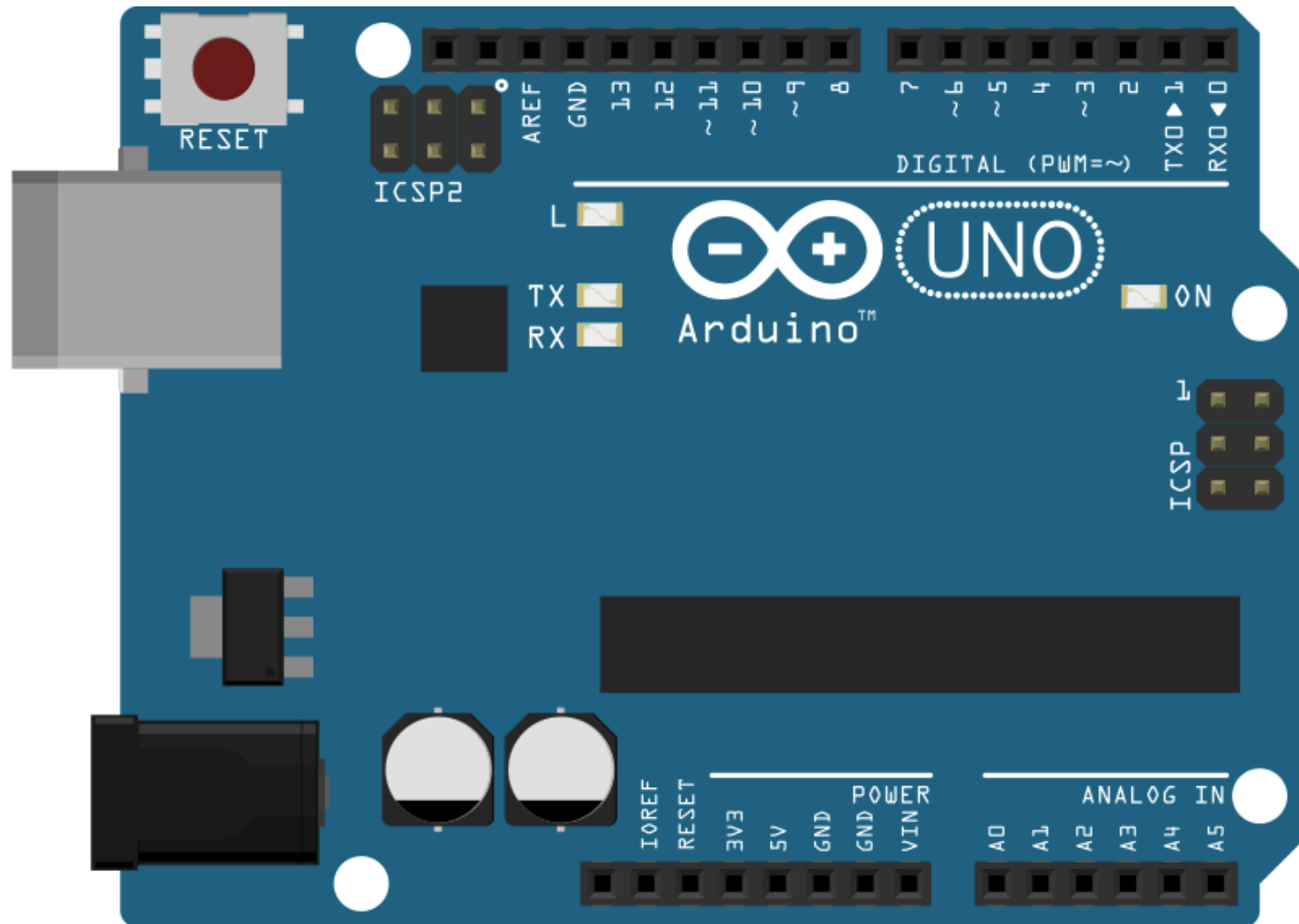
TI LaunchPad



# Application of Arduino

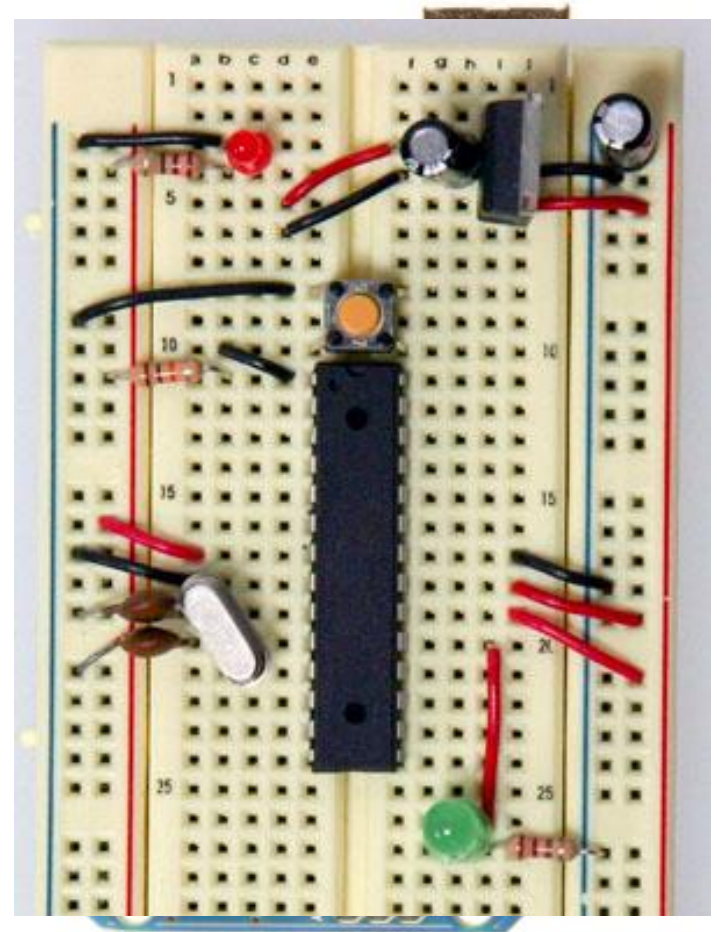


# Look into Arduino



# Components of Arduino Uno

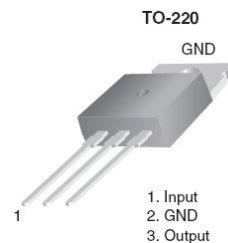
- ATmega328P
- USB <-> serial
- LEDs
- Resistors
- 16MHz crystal
- Pinout sockets
- Power regulators



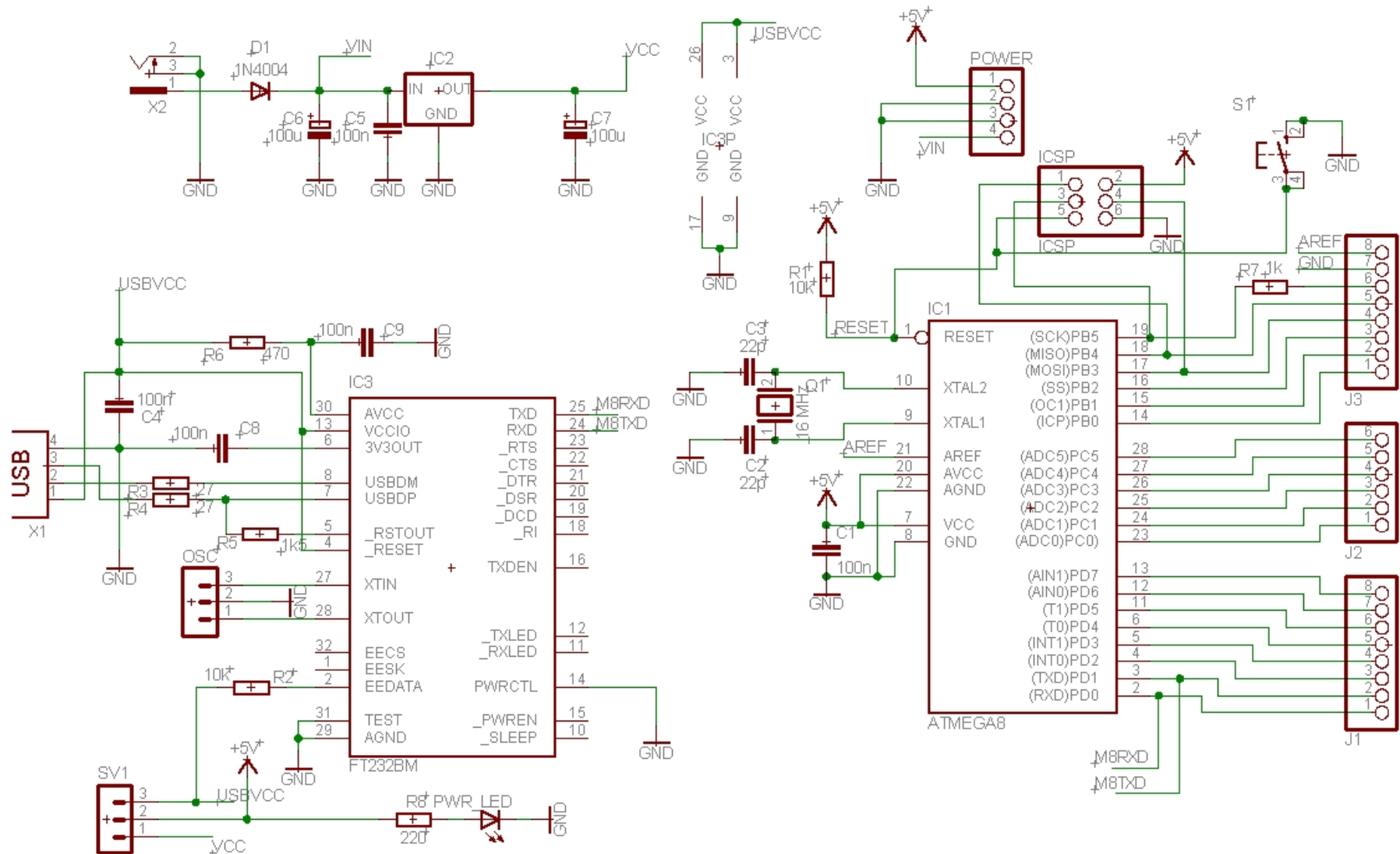


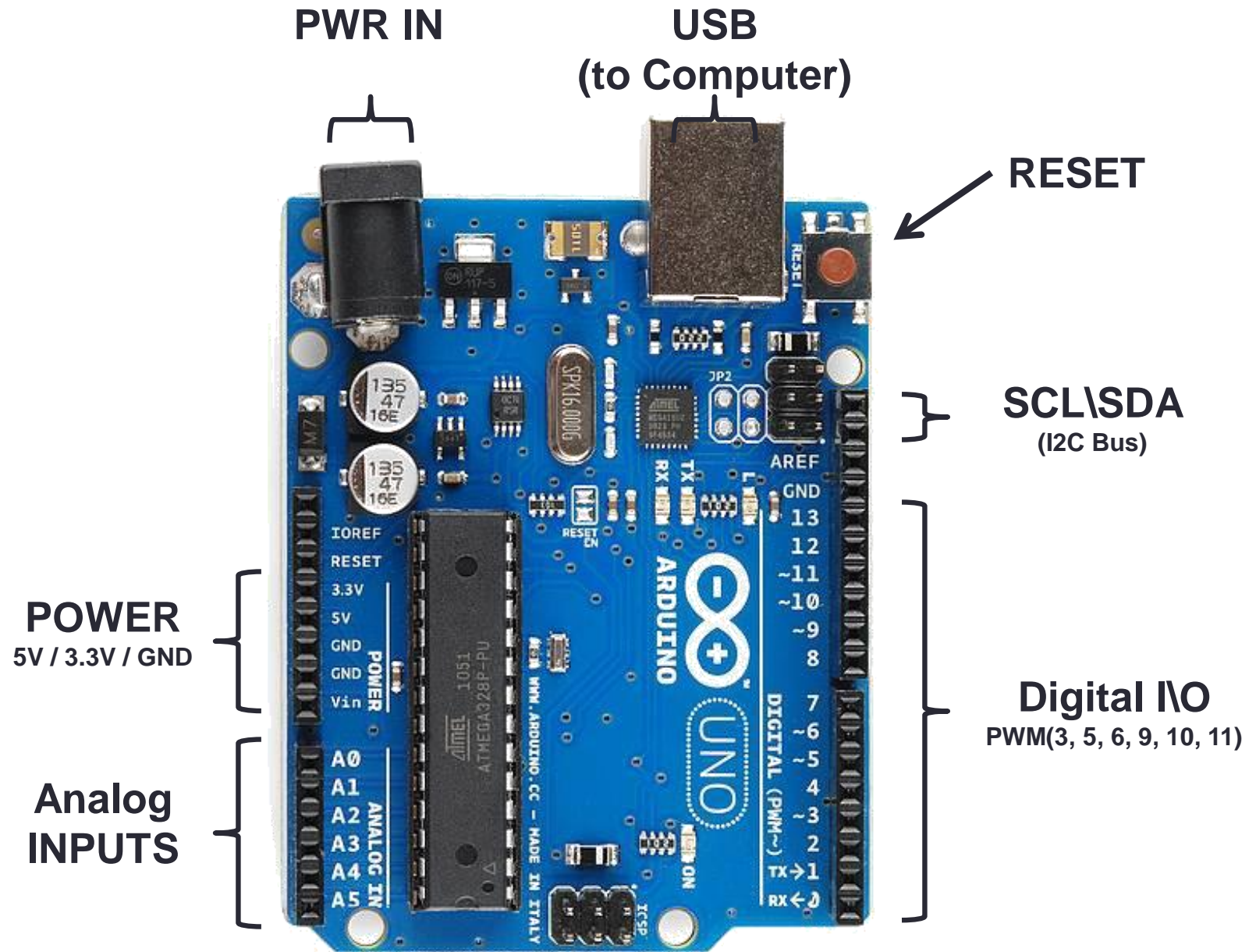
# Major Components

- ATmega168/328
- The 'brains' of the Arduino
- Program is loaded onto the chip
- Runs main loop until power is removed
- 16Mhz Crystal – the 'heartbeat' of the AVR chip
- 5 Volt and 3.3 Volt Regulators
- USB to TTL chip (ATmega16U2) that allows your Arduino to communicate with your computer over a simple USB link



# Schematic







# Key Specification of Arduino Uno

Arduino Uno	
Microcontroller	ATmega328 – Atmel 8-bit MCU RISC – 135 instructions 2 8-bit timers 1 16-bit timer
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
Clock Speed	16 MHz


# Program Download


- <http://arduino.cc/>

The screenshot shows the Arduino.cc website. The top navigation bar is teal with the Arduino logo, 'Buy', 'Software' (highlighted with a red dashed box), 'Products', 'Learning', 'Forum', 'Support', 'Blog', 'LOG IN', and 'SIGN UP'. The main content area has a light gray background. On the left is the Arduino logo. To its right, the text reads 'ARDUINO 1.8.1' followed by a description of the IDE. On the right side, a teal sidebar lists download options: 'Windows Installer' (with a red arrow pointing to it), 'Windows ZIP file for non admin install', 'Windows app' (with a 'Get' button), 'Mac OS X 10.7 Lion or newer', 'Linux 32 bits', 'Linux 64 bits', and 'Linux ARM'. Below these are links for 'Release Notes', 'Source Code', and 'Checksums (sha512)'. At the bottom, there are two sections: 'ARDUINO SOFTWARE HOURLY BUILDS' with a 'LAST UPDATE' box showing '25 January 2017 23:37:31 GMT' and a list of OSes; and 'ARDUINO 1.0.6 / 1.5.x / 1.6.x PREVIOUS RELEASES' with a description of previous versions.

**ARDUINO 1.8.1**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows Installer**   
**Windows ZIP file for non admin install**

**Windows app** [Get](#) 

**Mac OS X 10.7 Lion or newer**

**Linux 32 bits**  
**Linux 64 bits**  
**Linux ARM**

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

**ARDUINO SOFTWARE HOURLY BUILDS**

**LAST UPDATE**  
25 January 2017 23:37:31 GMT

Download a preview of the incoming release with the most updated features and bugfixes.

[Windows](#)  
[Mac OS X](#) (Mac OSX Lion or later)  
[Linux 32 bit](#), [Linux 64 bit](#), [Linux ARM](#)

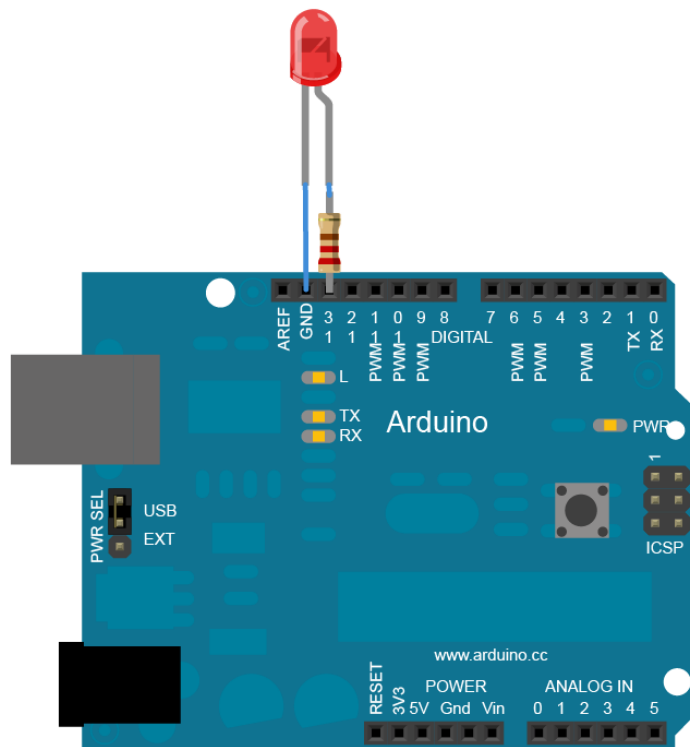
**ARDUINO 1.0.6 / 1.5.x / 1.6.x PREVIOUS RELEASES**

Download the [previous version of the current release](#), the classic [Arduino 1.0.x](#), or the [Arduino 1.5.x Beta version](#).

All the [Arduino 00xx versions](#) are also available for download. The Arduino IDE can be used on Windows, Linux (both 32 and 64 bits), and Mac OS X.

# First Example – Blink

- First program of Arduino – blinking LED:



```
void setup() {  
  pinMode(ledPin, OUTPUT); // sets t  
}  
void loop() {  
  digitalWrite(ledPin, HIGH); // sets t  
  delay(1000); // waits  
  digitalWrite(ledPin, LOW); // sets t  
  delay(1000); // waits  
}
```



compile

Done compiling.



upload



TX/RX flash



sketch runs



# Sketch Code – Blink

```
int led = 13; // Pin 13 has an LED connected on most Arduino boards

// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT); // initialize the digital pin as an output
}

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

# Program Download

1. Open  
“\Basic\Blink\Blink.ino”
2. To compile your sketch,  
click the checkmark
3. Make sure your Arduino is  
plugged into a USB port
4. Install driver for Arduino
5. Click the arrow to  
download the program to  
Arduino



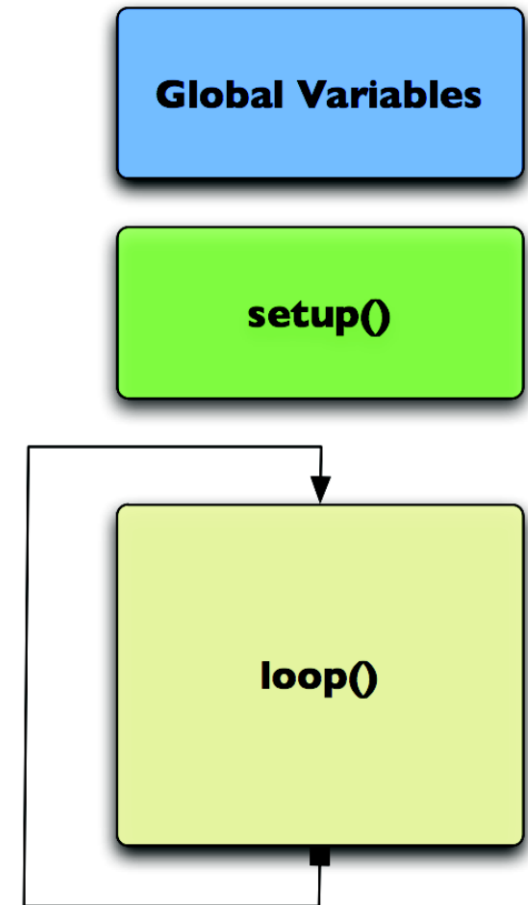
# Terminology

- “*sketch*” – a program you write to run on an Arduino board
- “*pin*” – an input or output connected to something, e.g., output to an LED, or input from a distance sensor
- “*digital*” – value is either HIGH or LOW (aka on/off; one/zero)
- “*analog*” – value ranges, usually from 0-255, e.g., LED brightness, motor speed, and etc.



# Arduino Sketch

- The language used to write sketch is very similar to C/C++
- Two required segments:
  1. **void setup ( ) { }** – all of the code within the curly braces will be run **ONCE** when the program first runs
  2. **void loop ( ) { }** – this function is run **AFTER** setup has finished, and all of the code within the curly braces will be run again and again until the power is removed



# Review of the Sketch “Blink”

```
int led = 13; // Pin 13 has an LED connected on most Arduino boards
```

**Global  
Variables**

```
// the setup routine runs once when you press reset:
```

**Setup**

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

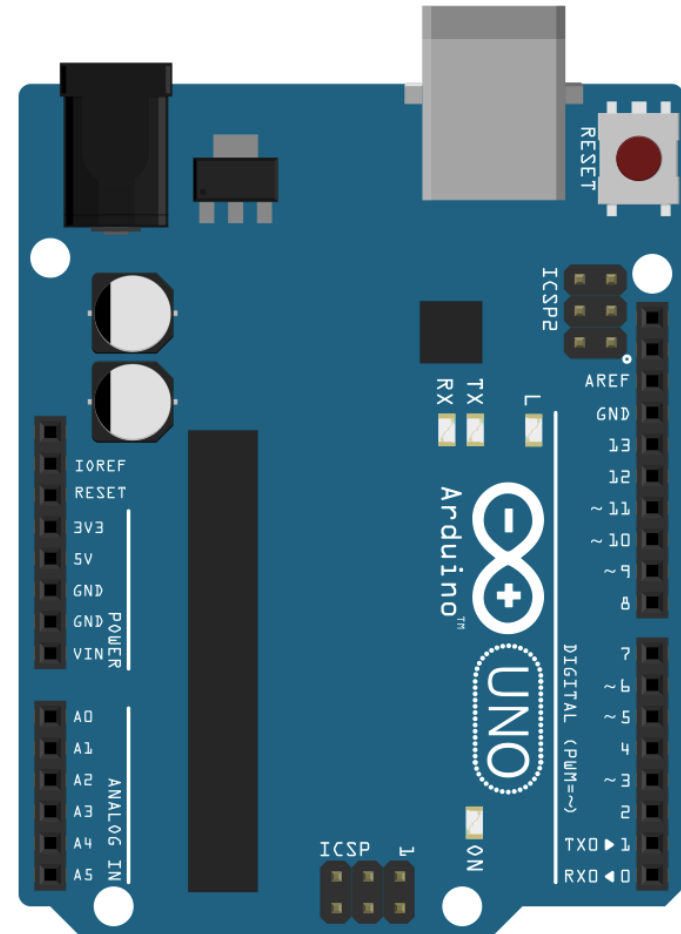
```
// the loop routine runs over and over again forever:
```

**Loop**

```
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

# Programming – Digital I/O

- `pinMode(pin, mode);`
  - `pin`: (0 to 19) on the Arduino board
  - `mode`: (`INPUT` or `OUTPUT`)
- `digitalWrite(pin, value);`
  - `value`: (`HIGH` or `LOW`)
- `value = digitalRead(pin);`



# Review of the Sketch “Blink”

```
int led = 13; // Pin 13 has an LED connected on most Arduino boards
```

```
// the setup routine runs once when you press reset:
```

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

```
// the loop routine runs over and over again forever:
```

```
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```



# Variables

- **boolean** – a simple true and false variable which takes 1 bit of RAM
- **char** – a character variable that stores ASCII code (“A” = 65) and uses 1 byte of RAM
- **int**\* – an variable which stores an integer number in 2 bytes and has a range of -32,768 and 32,767
- **long**\* – a variable which stores an integer number in 4 bytes and has a range of -2,147,483,648 and 2,147,483,647
- **float** – floating decimals which takes 4 bytes of RAM and has a range of -3.4028235E+38 and 3.4028235E+38

Note: \* represents “unsigned” option

# Operators – Arithmetic and Relational

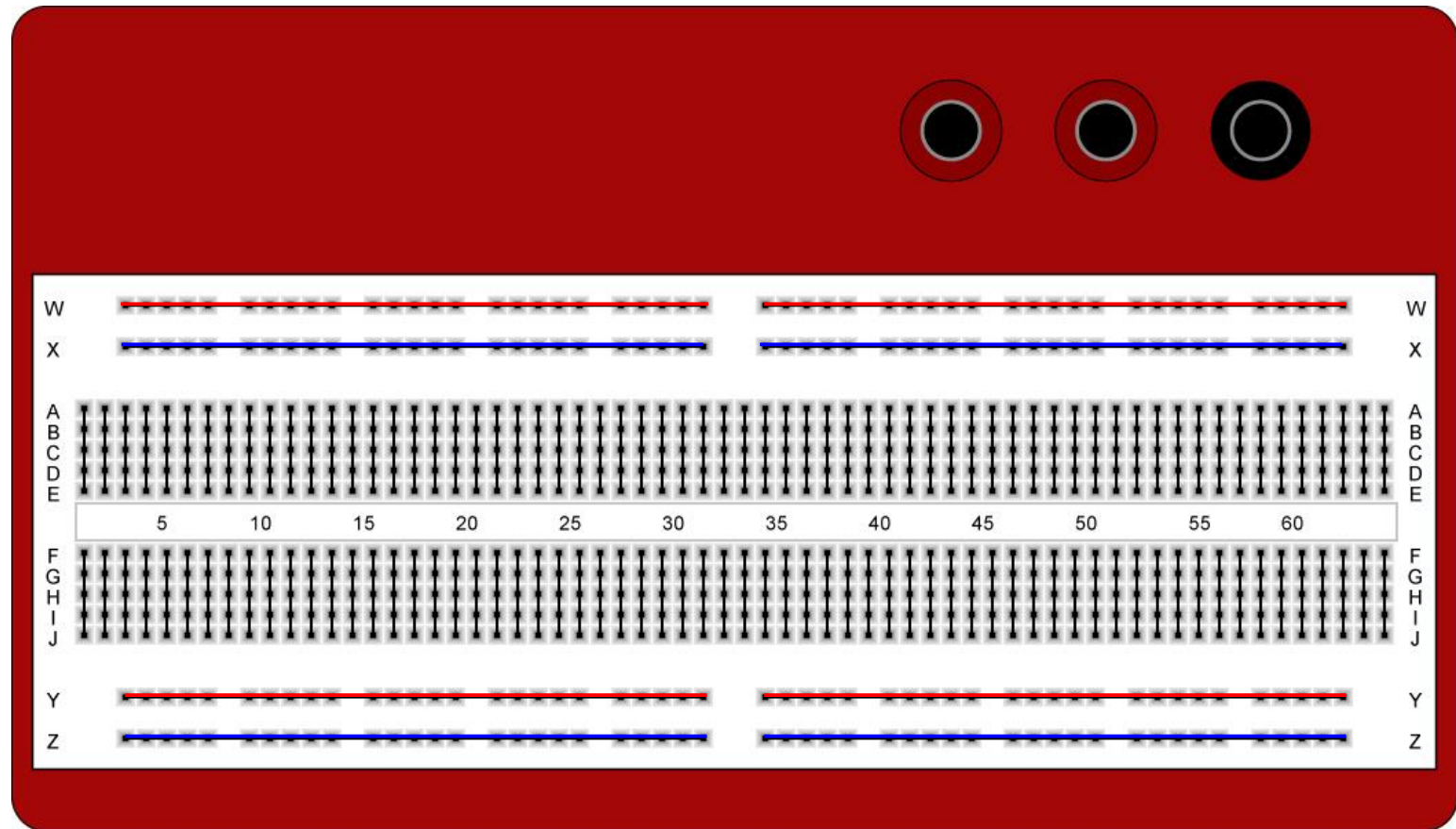
- **=** assignment
- **+** addition
- **-** subtraction
- **\*** multiplication
- **/** division
- **%** remainder
- **==** equal to
- **!=** not equal to
- **<** less than
- **<=** less than or equal to
- **>** greater than
- **>=** greater than or equal to

- Complete list of operator  
<http://arduino.cc/en/Reference/HomePage>

# Control Structures and Further

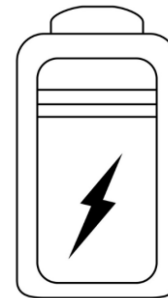
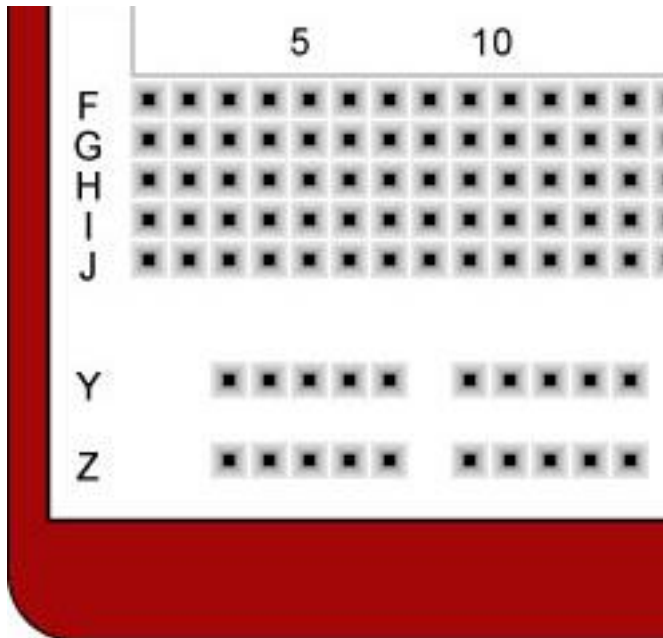
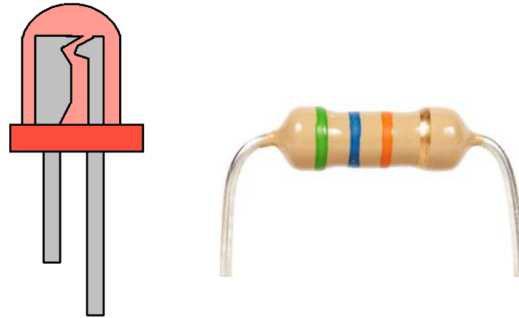
- Structure:
  - `if (condition) { } else { }`
  - `for (int i=0; i < #repeats; i++) { }`
  - `while (condition) { }`
  - `switch (variable) { case X: case Y: default: }`
- Others:
  - `;` the end of a line of code
  - `//` single line comment
  - `/* */` multiline comment

# Typical Solderless Breadboard



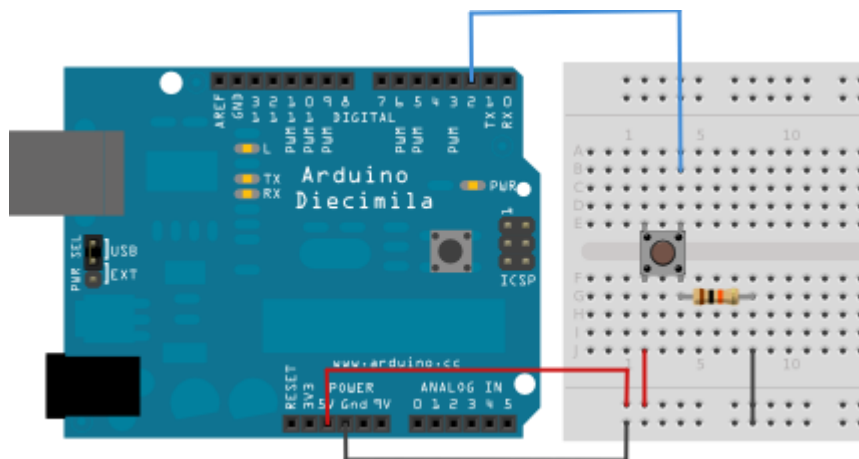
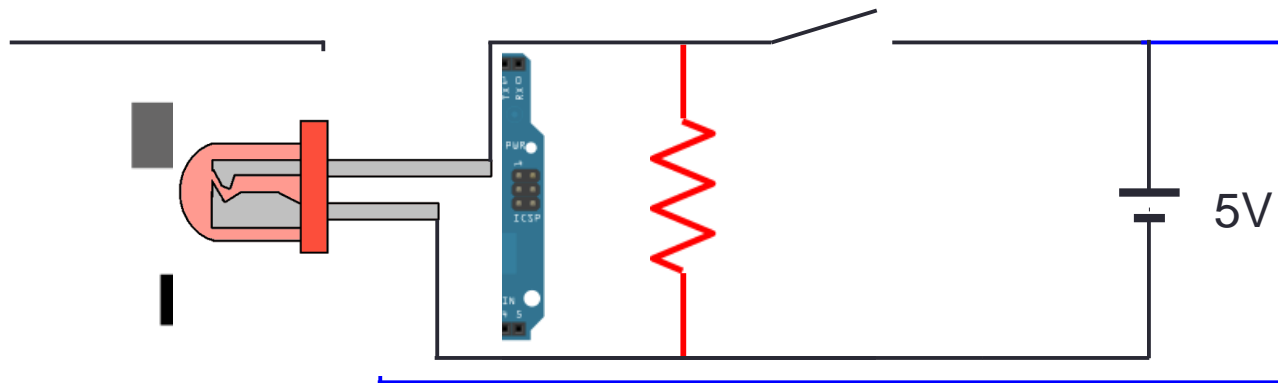


# Using Solderless Breadboard



# Example – LED Control Using Button

- Voltage measurement – 0 or 5 V?



# Sketch Code – Button

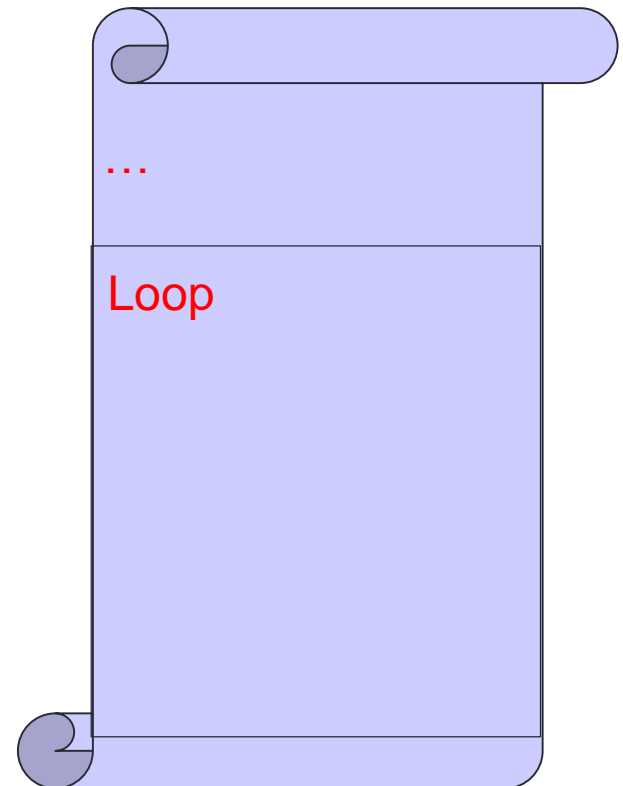
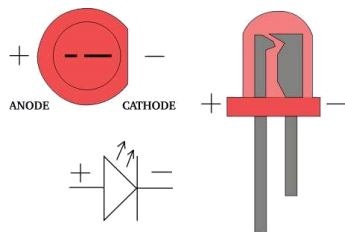
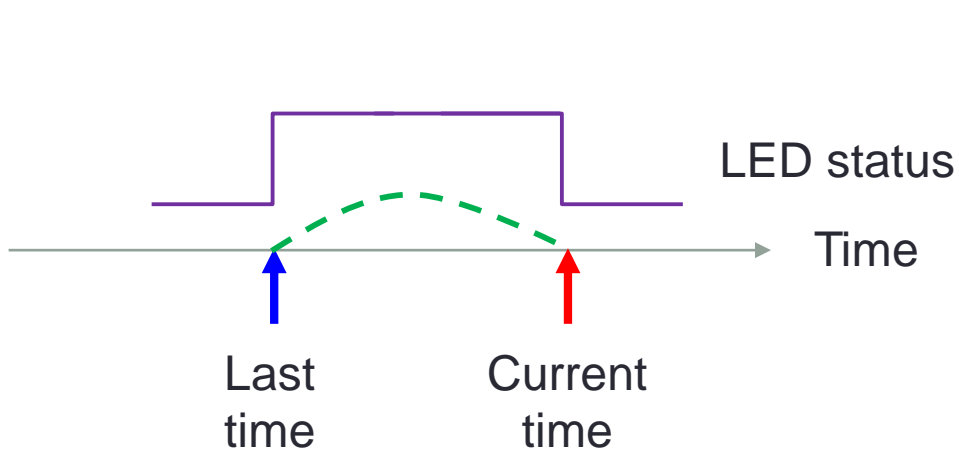
```
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  pinMode(ledPin, OUTPUT);   // initialize the LED pin as an output
  pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input
}

void loop(){
  buttonState = digitalRead(buttonPin); // read the state of the pushbutton value
  if (buttonState == HIGH) {           // if pushbutton is pressed
    digitalWrite(ledPin, HIGH);        // turn LED on
  }
  else {
    digitalWrite(ledPin, LOW);         // turn LED off
  }
}
```

# Example – Blink Without Using Delay

- Delay makes the microcontroller doing nothing but wait
- What if you want do something while blinking an LED?





# Sketch Code – Blink Without Delay

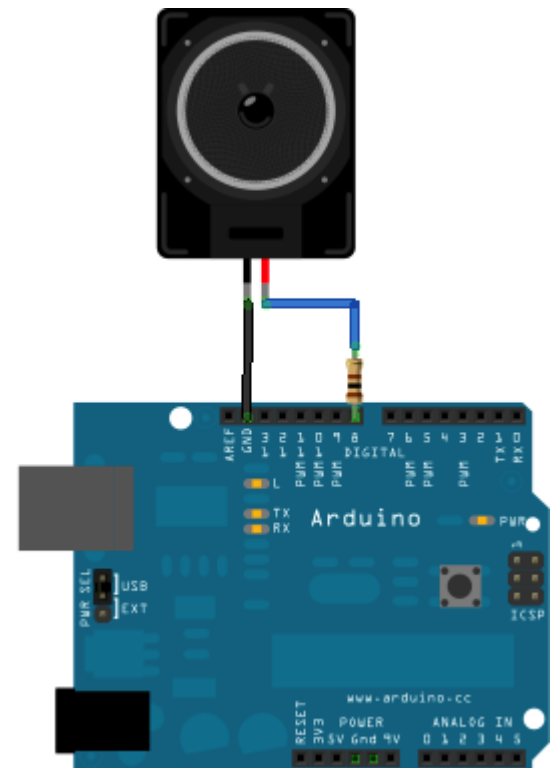
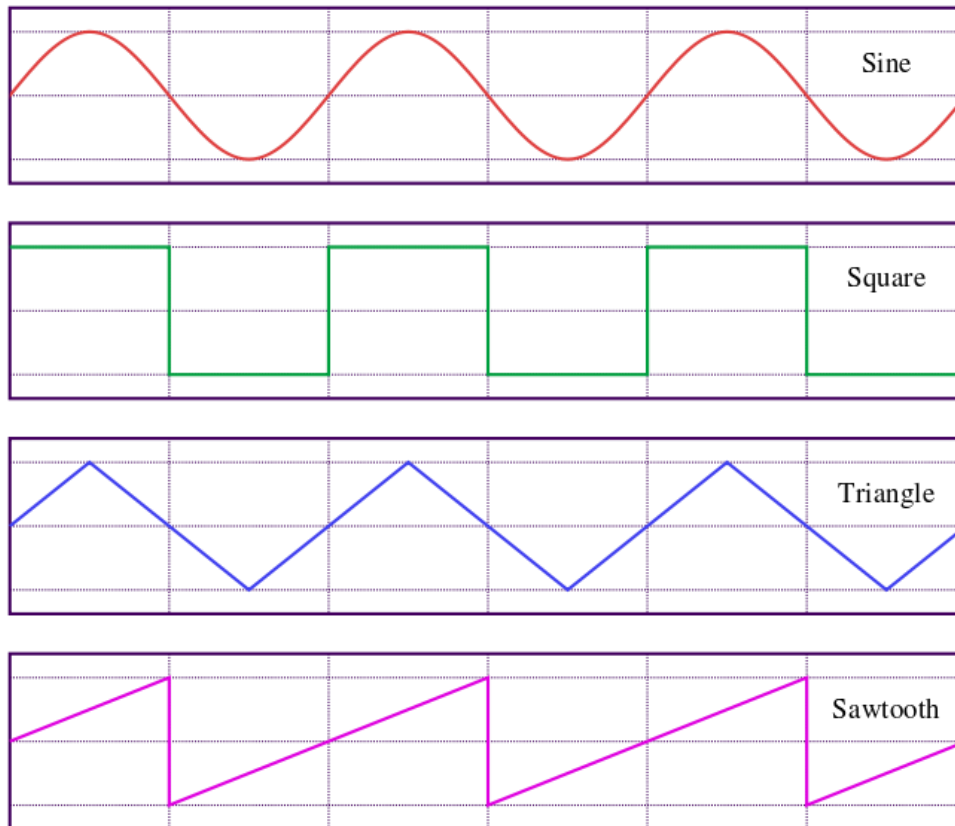
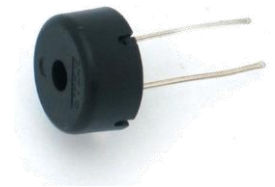
```
const int ledPin = 13;      // the number of the LED pin
int ledState = LOW;         // ledState used to set the LED
long previousMillis = 0;    // will store last time LED was updated
long interval = 1000;       // interval at which to blink (milliseconds)

void setup() {
  pinMode(ledPin, OUTPUT);  // set the digital pin as output:
}

void loop() {
  unsigned long currentMillis = millis();
  if(currentMillis - previousMillis > interval) {
    previousMillis = currentMillis; // save the last time you blinked the LED
    if (ledState == LOW)           // if the LED is off turn it on and vice-versa
      ledState = HIGH;
    else
      ledState = LOW;
    digitalWrite(ledPin, ledState);
  }
}
```

# Example – Tone

- Use the `tone()` command to generate notes
- What kind of wave can an Arduino generate?



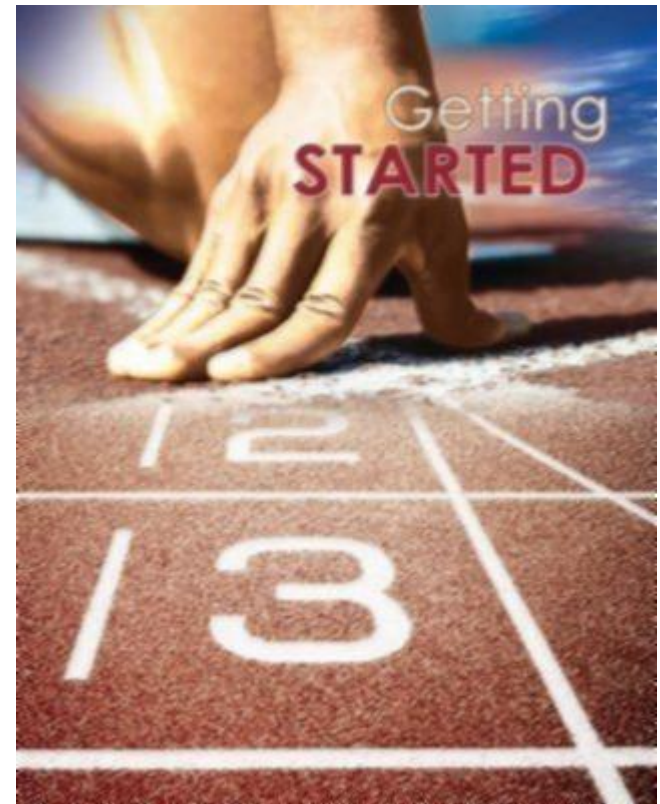
# Sketch Code – ToneMelody

```
#include "pitches.h" ← How to include "pitches.h"!?
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4};
// note durations: 4 = quarter note, 8 = eighth note, etc.
int noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4 };
void setup() {
  for (int thisNote = 0; thisNote < 8; thisNote++) {
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    noTone(8);  // stop the tone playing:
  }
}
void loop() {
}
```

# What Have We Learned So Far?

- There are pins of the microcontroller that can function as input or output – I/O
- There is a component in the microcontroller that counts the time – timer/counter
- There are libraries of Arduino that we can use to control buzzer

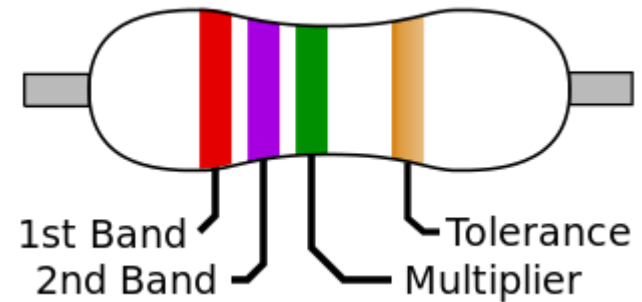
# Getting Started





# Reference

- <http://www.arduino.cc/>
- ATmega328P data sheet



**4-Band-Code**

2%, 5%, 10%      560k  $\Omega$   $\pm$  5%

COLOR	1 <sup>ST</sup> BAND	2 <sup>ND</sup> BAND	3 <sup>RD</sup> BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1 $\Omega$	
Brown	1	1	1	10 $\Omega$	$\pm$ 1% (F)
Red	2	2	2	100 $\Omega$	$\pm$ 2% (G)
Orange	3	3	3	1K $\Omega$	
Yellow	4	4	4	10K $\Omega$	
Green	5	5	5	100K $\Omega$	$\pm$ 0.5% (D)
Blue	6	6	6	1M $\Omega$	$\pm$ 0.25% (C)
Violet	7	7	7	10M $\Omega$	$\pm$ 0.10% (B)
Grey	8	8	8		$\pm$ 0.05%
White	9	9	9		
Gold				0.1 $\Omega$	$\pm$ 5% (J)
Silver				0.01 $\Omega$	$\pm$ 10% (K)

**5-Band-Code**

0.1%, 0.25%, 0.5%, 1%      237  $\Omega$   $\pm$  1%