Principles and Applications of Microcontrollers

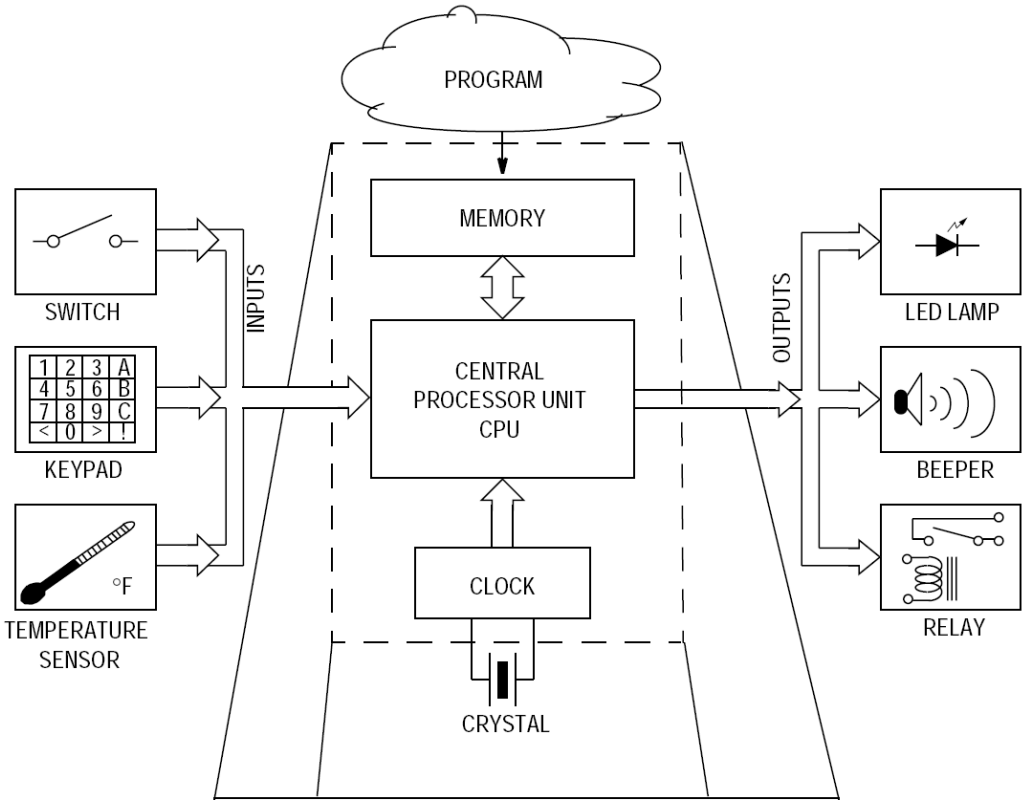
Yan-Fu Kuo
Dept. of Bio-industrial Mechatronics Engineering
National Taiwan University

Today:

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

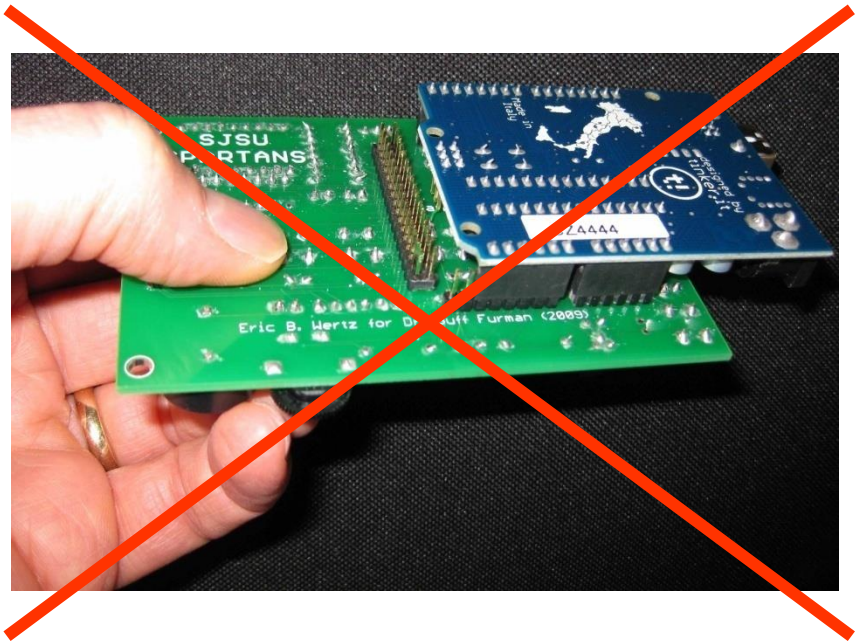
What is A Microcontroller?

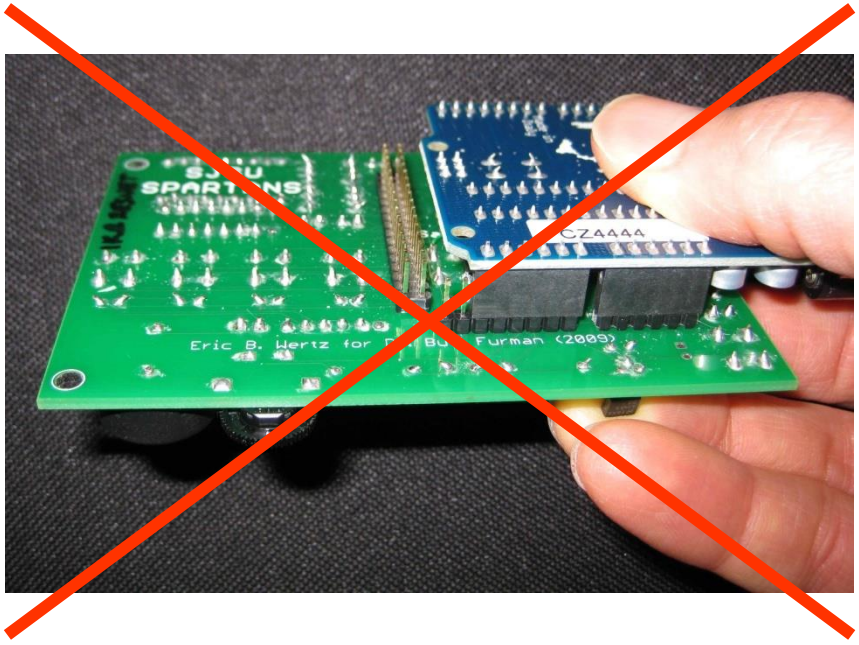
- A small computer on a single integrated circuit
- Containing a processor, memory, and programmable input/output peripherals



Handling Arduino – Do *NOT* Do This!

Improper Handling - ***NEVER***!!!



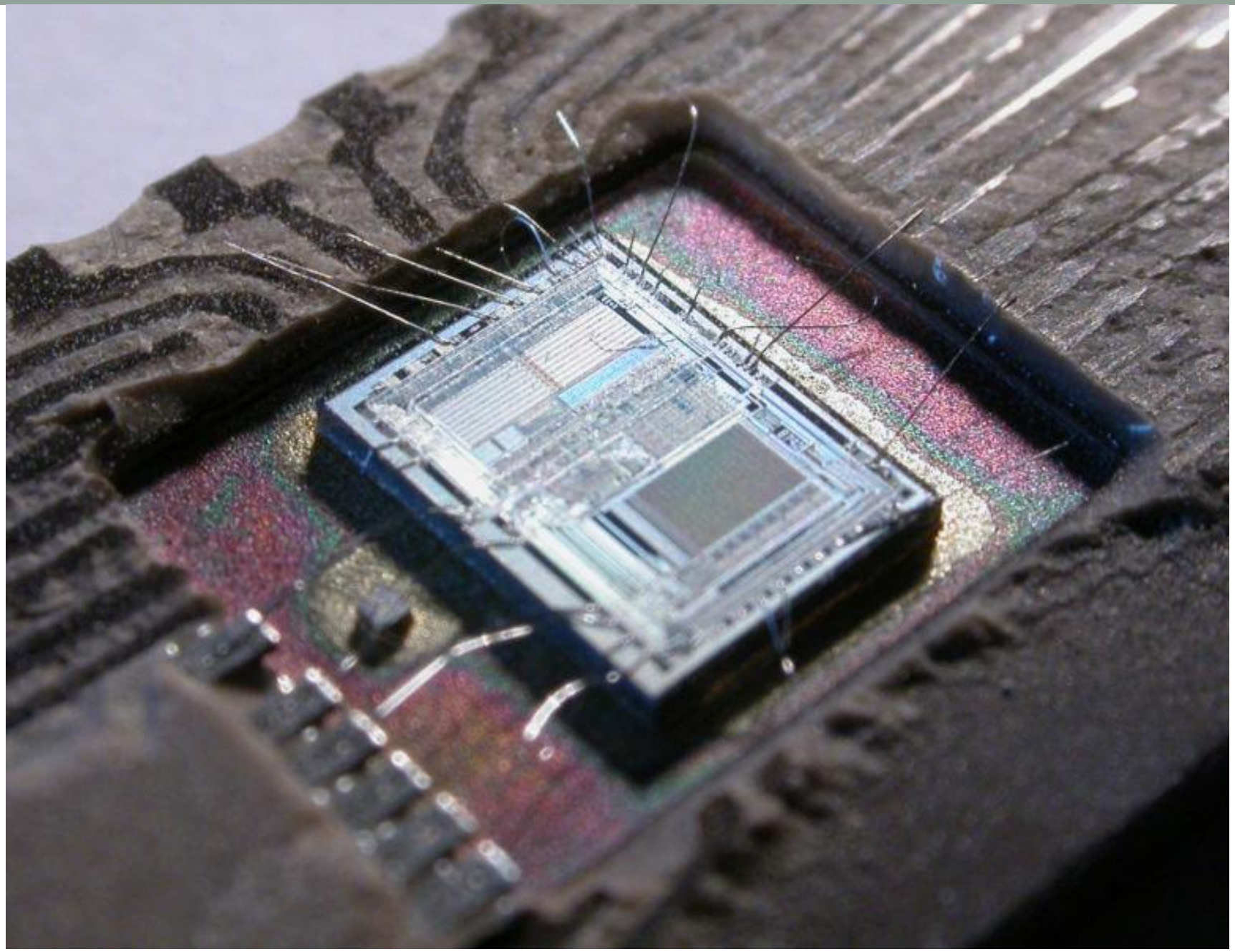


Arduino Uno

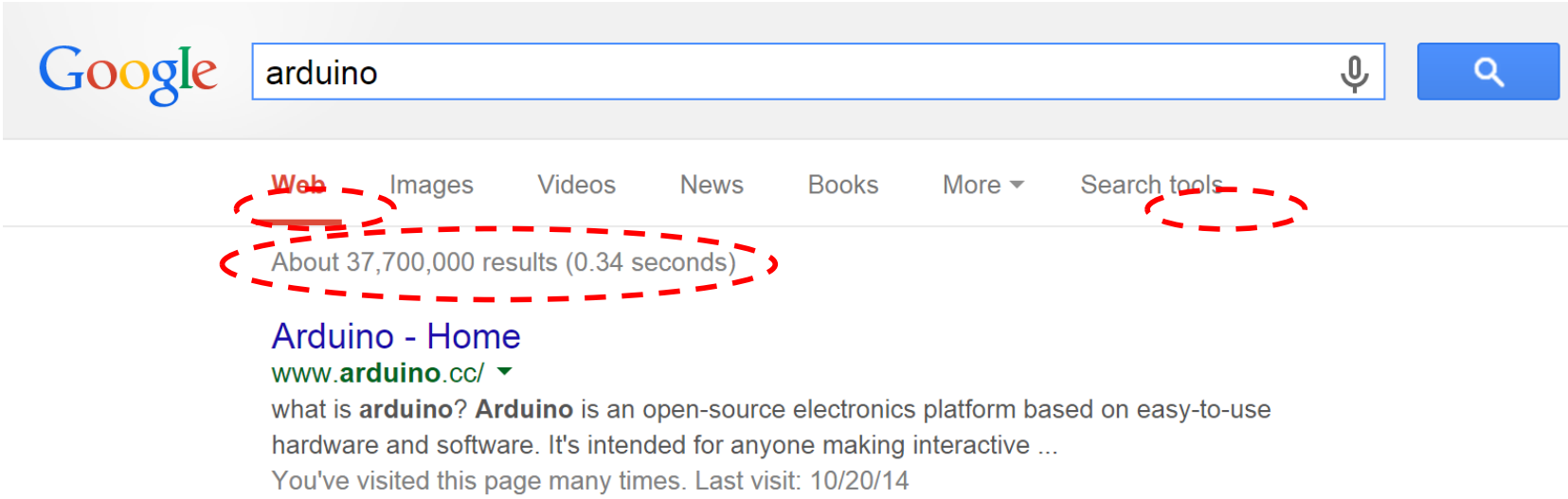
- An integrated microcontroller
- Atmel ATmega328p

Intel 80286

=



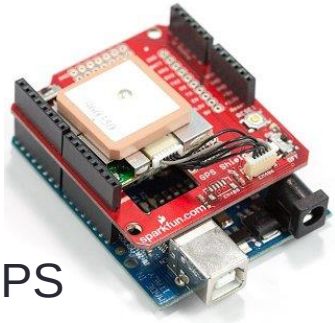
Why Arduino?



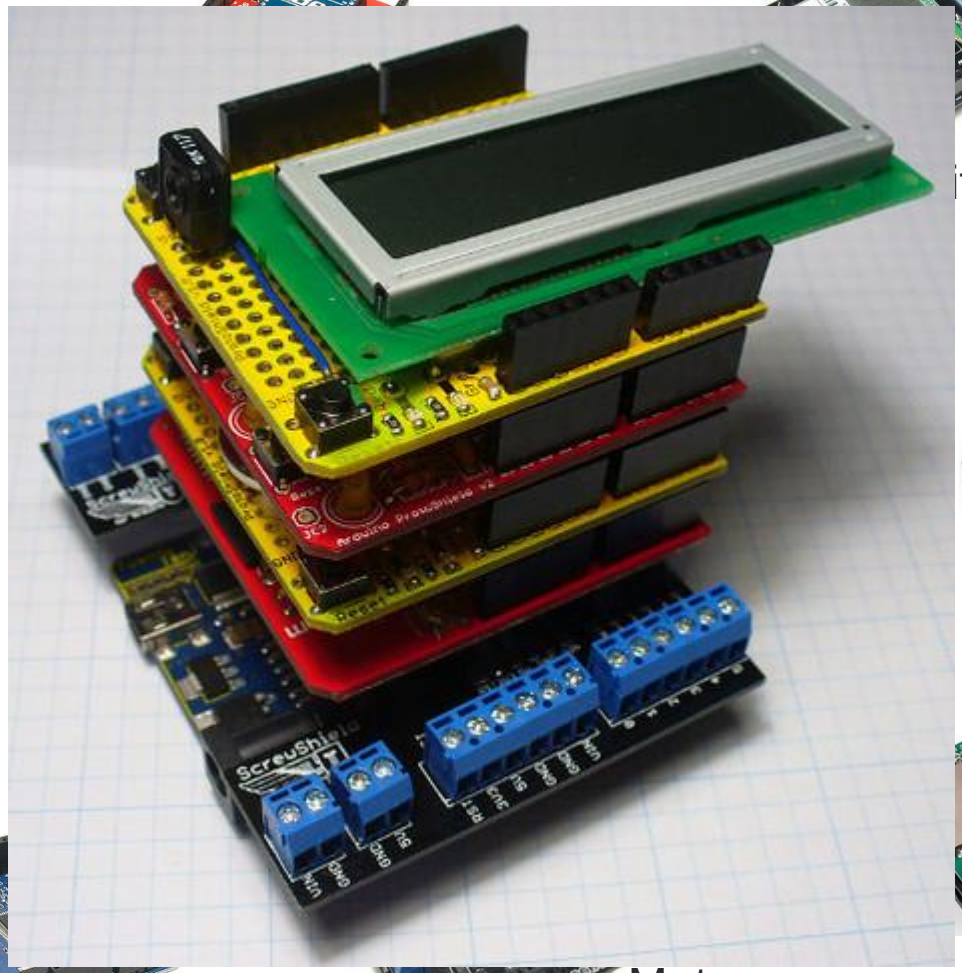
- It has dominated the market
- Open source
- Low-cost – less than $30 USD
- Easy to use – less hardware logic
- Friendly programming environment

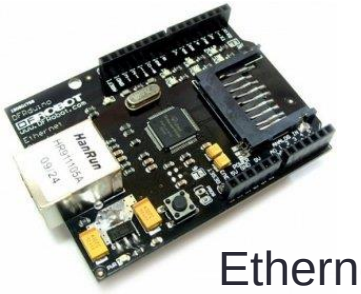
ARDUINO
OPEN-SOURCE
COMMUNITY

Arduino Shields



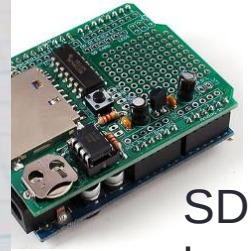
GPS





Ethernet

Joystick



SD card logger

VGA

Motor controller

SPECIAL BOARD FIELD GUIDE

EVOLUTION OF ARDUINO

2005 2006 2007 2008

In 2005, a group at Italy's **Interaction Design Institute Ivrea** 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.

Old-style RS-232 serial port rather than USB.

Designed to be built on a home-etched PCB.

Adds female pin headers, data transfer LEDs.

First board to bear the Arduino name.

Most advanced etch-it-yourself PCB design.

Atmel's 8-bit megaAVR microcontroller family is an Arduino signature.

ATmega8

First to ship with ATmega168.

The ATmega168 doubles on-board memory to 16KB, but is otherwise nearly identical to the ATmega8.

ATmega168

Large 28-pin Plastic Dual In-line Package (PDIP-28) for through-hole soldering.

Thin Quad Flat Package (TQFP) designed for surface mount soldering.

Very thin Quad Flat No-lead (VQFN) package replaces leads with underside pads.

Sew-through contact pads for connecting conductive thread.

Underside pins for breadboard connection.

Designed for battery-powered wireless projects.

First board to use surface-mount processor.

Auto-selects power supply. First to ship with ATmega328.

On-chip memory doubles again to 32KB.

ATmega328

Designed for semipermanent installation.

Mini form-factor compatible.

Official reference model for Arduino platform.

Replaces sew-through contacts with button snaps.

The Mega took Arduino to a new level, quadrupling on-chip memory to 128KB and more than tripling the total number of I/O and input pins in a significantly larger form factor.

ATmega1280

Emulates mouse and keyboard over USB.

Integrates USB controller into processor chip.

ATmega32u4

With the Mega2560, memory doubled again to 256KB. Though larger, the new form factor remains pin-compatible with the standard Arduino shield footprint.

ATmega2560

Mega update for use with Android Accessory Development Kit (ADK).

First dual-core model, combining ATmega32u4 with MIPS embedded Linux machine.

The Due marks Arduino's first departure from the AVR architecture. The ATSAM3X8E is an ARM Cortex-M3 processor with twice the memory and four times the clock speed of the ATmega2560.

ATSAM3X8E

First Arduino to mount 32-bit processor. Runs at 3.3V rather than 5V.

2009 2010 2011 2012 2013

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

The Arduino Uno

TI LaunchPad

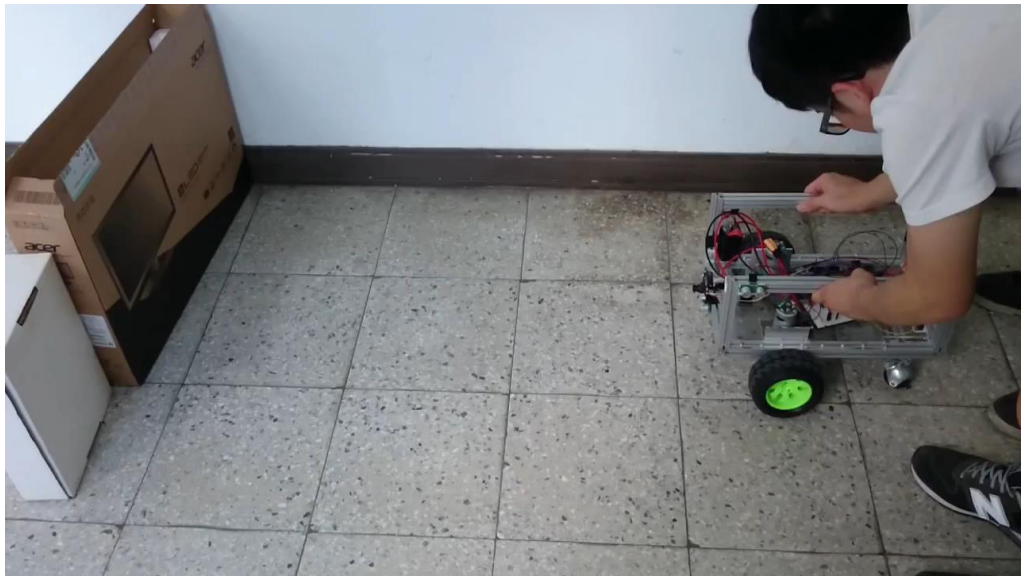
Ever-Shrinking Derivatives

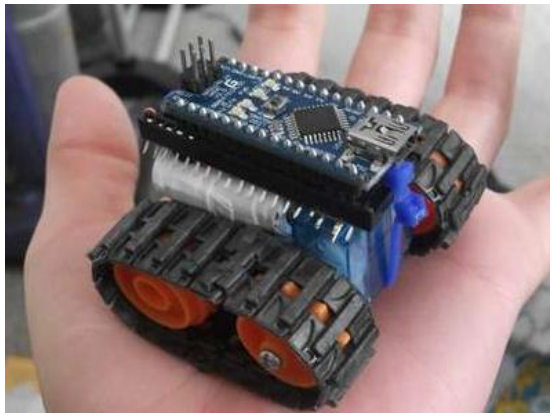
As discussed, the success of the Arduino has led to numerous copies and compatible boards arriving on the market. The crowdfunding site Kickstarter is littered with them; some amazingly successful, some not so much. It'd be impossible to list them all, but there are some that stand out, chiefly because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

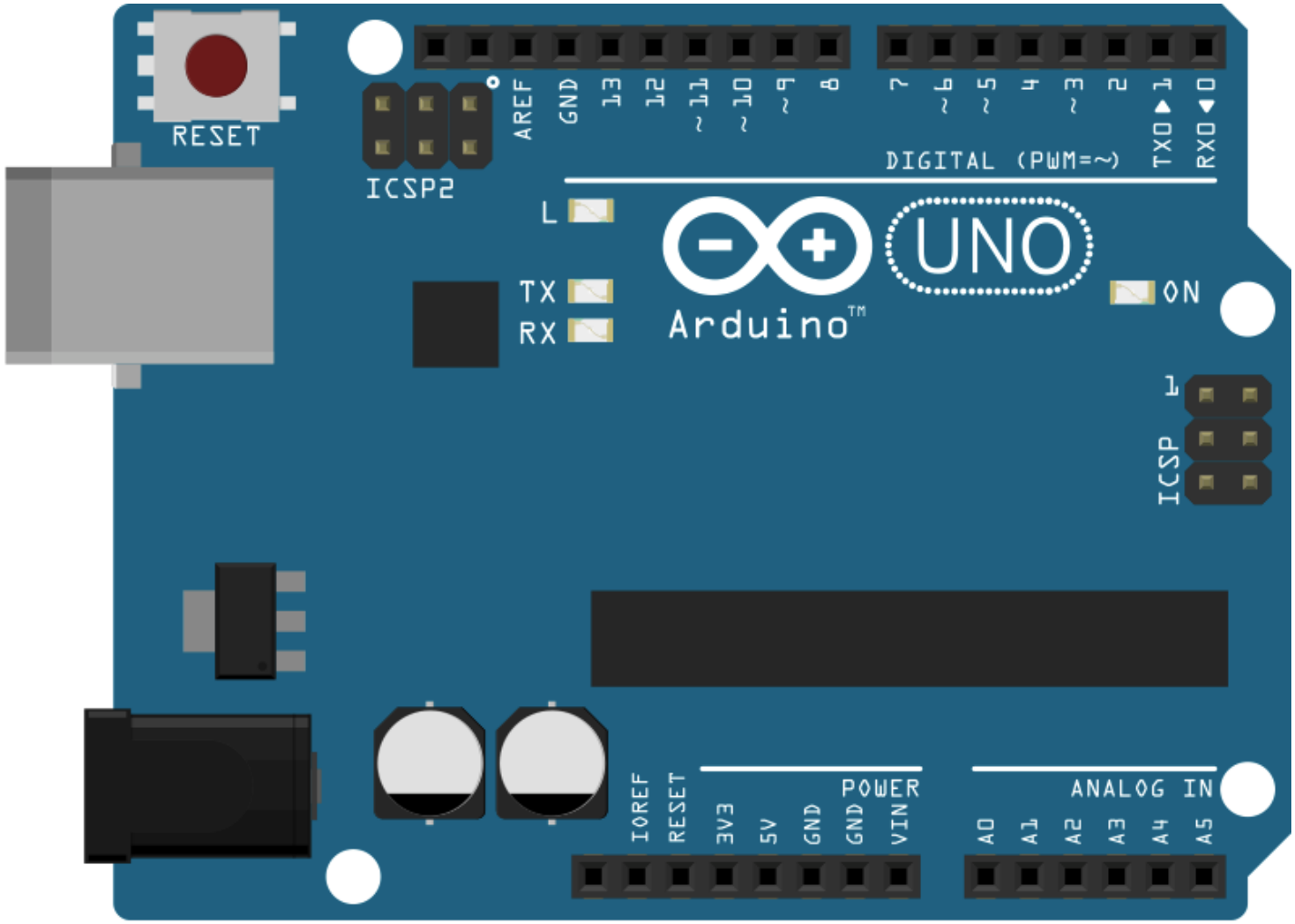
The DigiSpark is another tiny Arduino-compatible board (**Figure B**). It is built around the ATtiny85 microcontroller, making it much less powerful than the TinyDuino. It only has 6 I/O pins but, on the other hand, it costs just $9. Like the TinyDuino, it has a variety of interesting shield kits allowing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

Application of Arduino



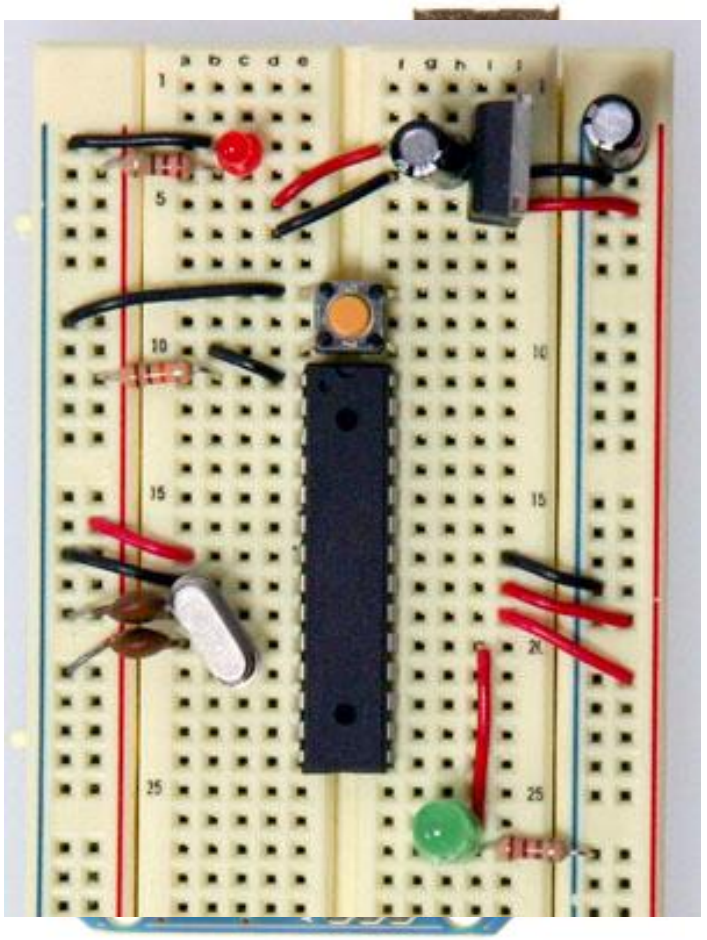


Look into Arduino



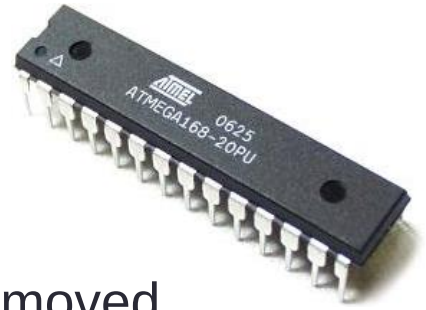
Components of Arduino Uno

- ATMega328P
- USB <-> serial
- LEDs
- Resistors
- 16MHz crystal
- Pinout sockets
- Power regulators



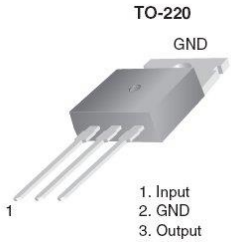
Major Components

- ATMega168/328
- The ‘brains’ of the Arduino
- Program is loaded onto the chip
- Runs main loop until power is removed



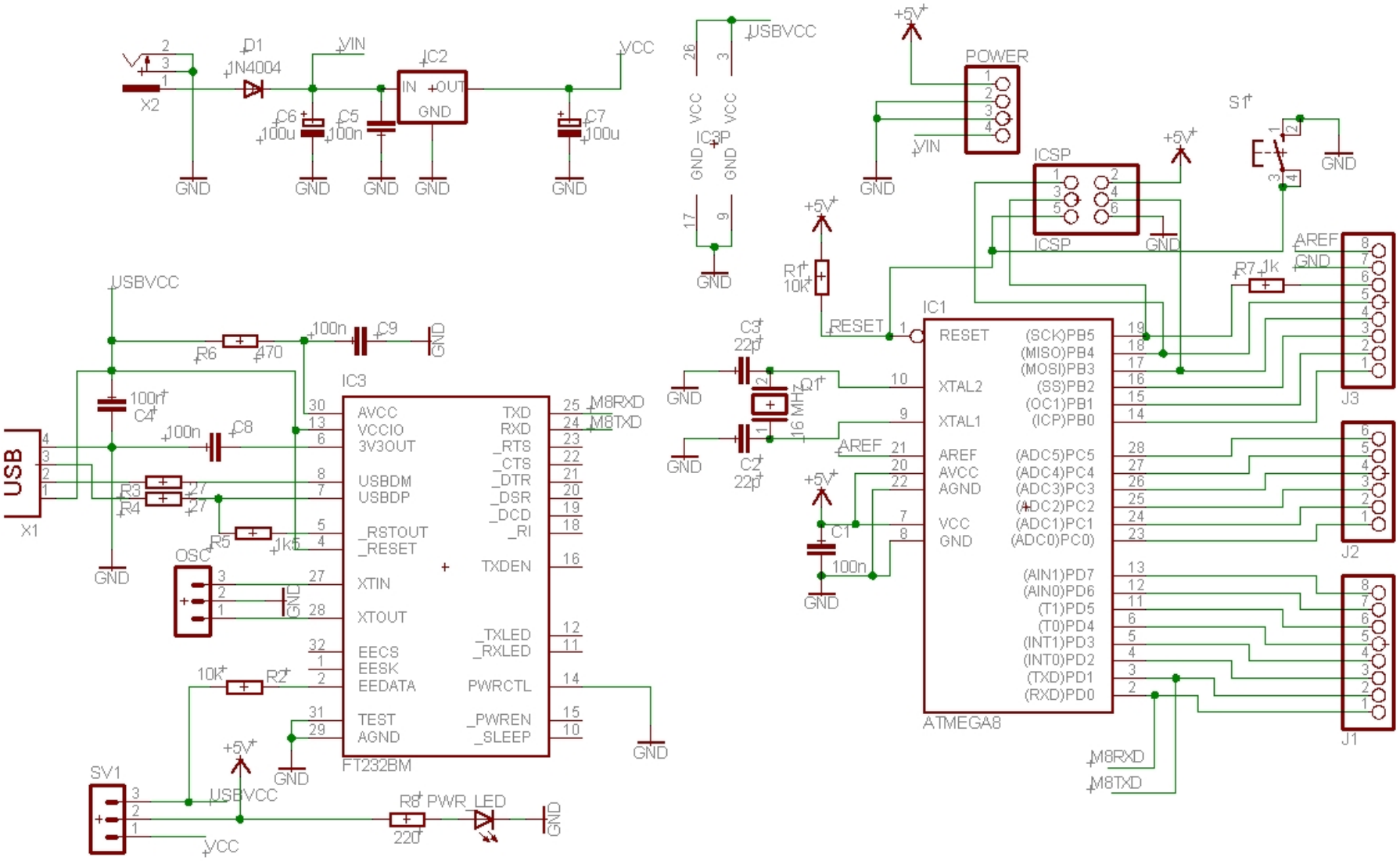
- 16Mhz Crystal – the ‘heartbeat’ of the AVR chip

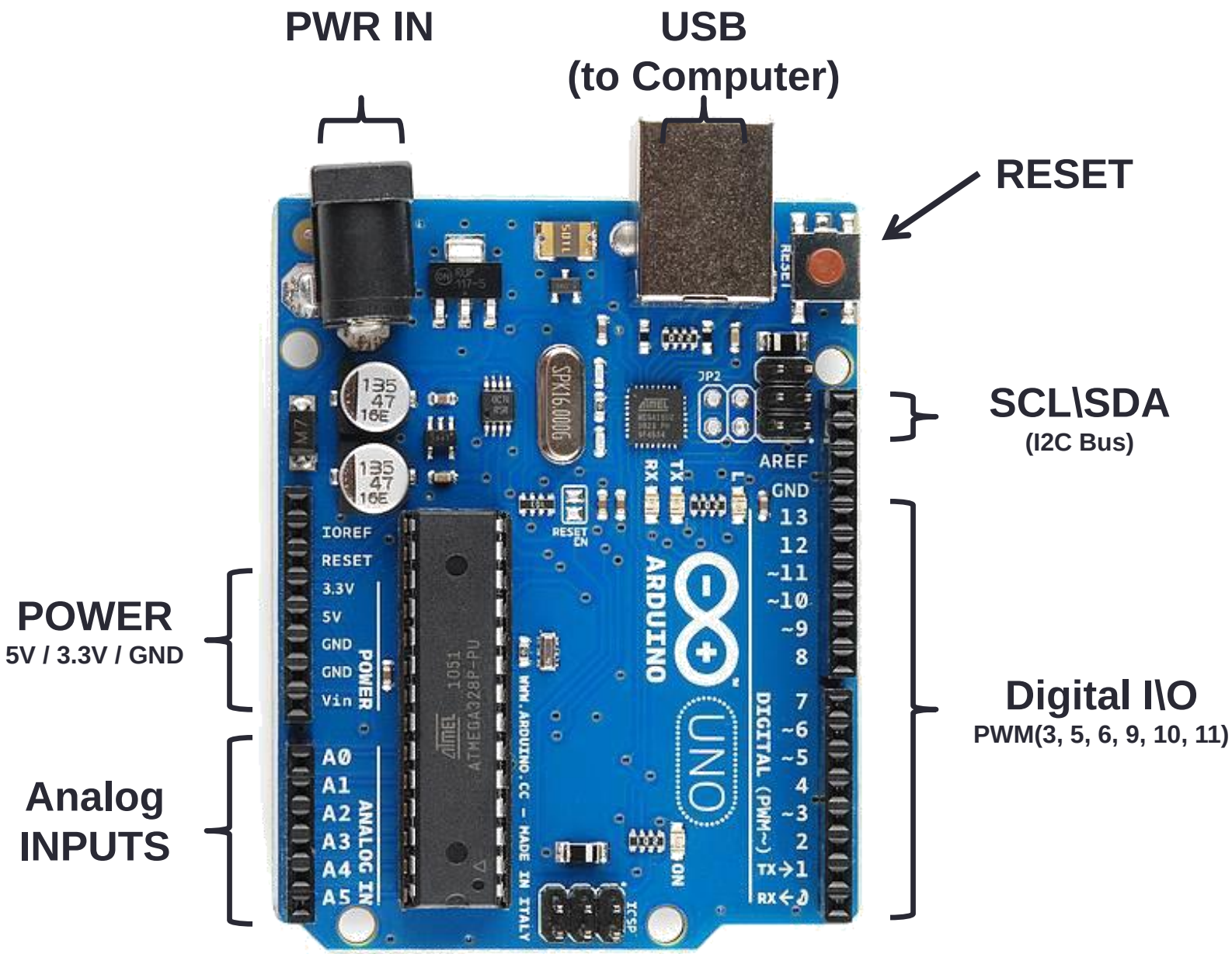
- 5 Volt and 3.3 Volt Regulators



- USB to TTL chip (ATmega16U2) that allows your Arduino to communicate with your computer over a simple USB link

Schematic


X2
D1
1N4004
VIN
IC2
IN
OUT
GND
VCC
C6
100u
C5
100n
C7
100u
USBVCC
IC3P
VCC
GND
POWER
VIN
ICSP
S1
R1
10k
RESET
IC1
C3
22p
C2
22p
Q1
16 MHz
C1
100n
AREF
R7
1k
J3
J2
J1
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
(SCK)PB5
(MISO)PB4
(MOSI)PB3
(SS)PB2
(OC1)PB1
(ICP)PB0
(ADC5)PC5
(ADC4)PC4
(ADC3)PC3
(ADC2)PC2
(ADC1)PC1
(ADC0)PC0)
(AIN1)PD7
(AIN0)PD6
(T1)PD5
(T0)PD4
(INT1)PD3
(INT0)PD2
(TXD)PD1
(RXD)PD0
ATMEGA8
M8RXD
M8TXD
USB
X1
R6
470
100n
C9
C4
100n
C8
R3
27
R4
27
R5
1k5
OSC
R2
10k
SV1
R8
220
PWR LED
IC3
AVCC
VCCIO
3V3OUT
USBDM
USBDP
_RSTOUT
_RESET
XTIN
XTOUT
EECS
EESK
EEDATA
TEST
AGND
TXD
RXD
_RTS
_CTS
_DTR
_DSR
_DCD
_RI
TXDEN
_TXLED
_RXLED
PWRCTL
_PWREN
_SLEEP
FT232BM

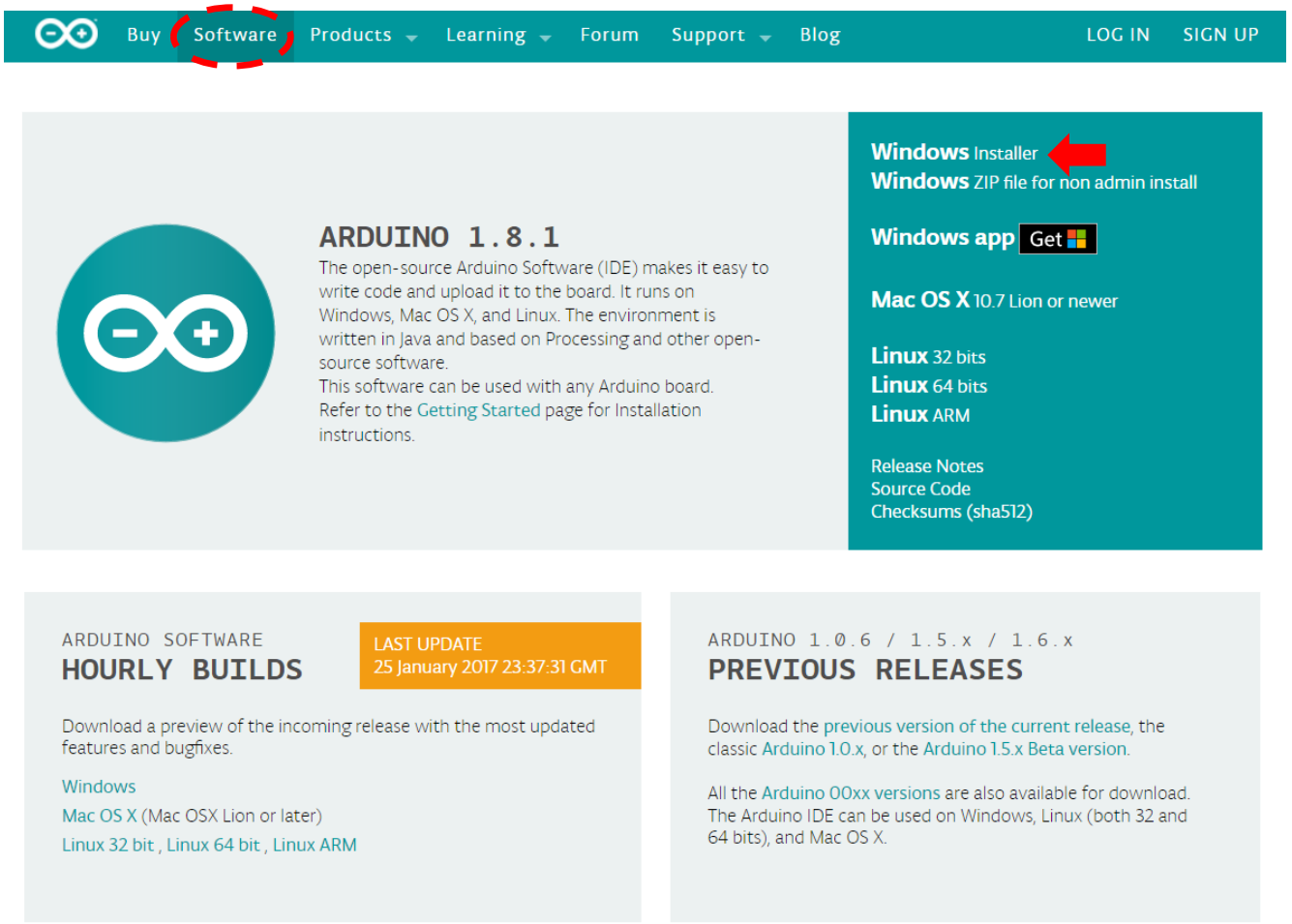

PWR IN
USB
(to Computer)
RESET
SCL\SDA
(I2C Bus)
POWER
5V / 3.3V / GND
Analog
INPUTS
Digital I\O
PWM(3, 5, 6, 9, 10, 11)
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
ARDUINO
UNO
ATMEGA328P-PU
WWW.ARDUINO.CC – MADE IN ITALY
ICSP
ON
JP2
SPX16.000G

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

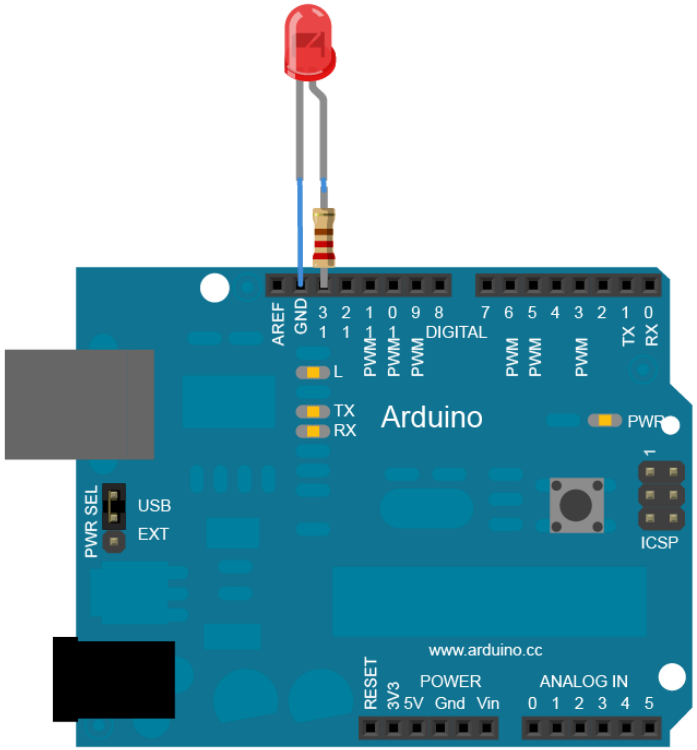
Program Download

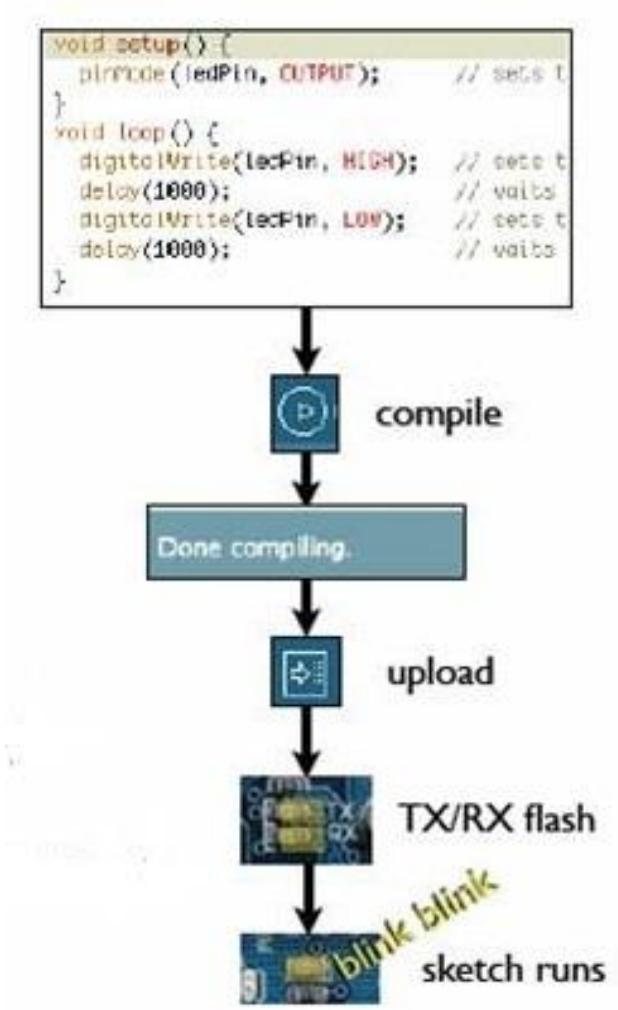
- http://arduino.cc/



First Example – Blink

- First program of Arduino – blinking LED:





Link

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

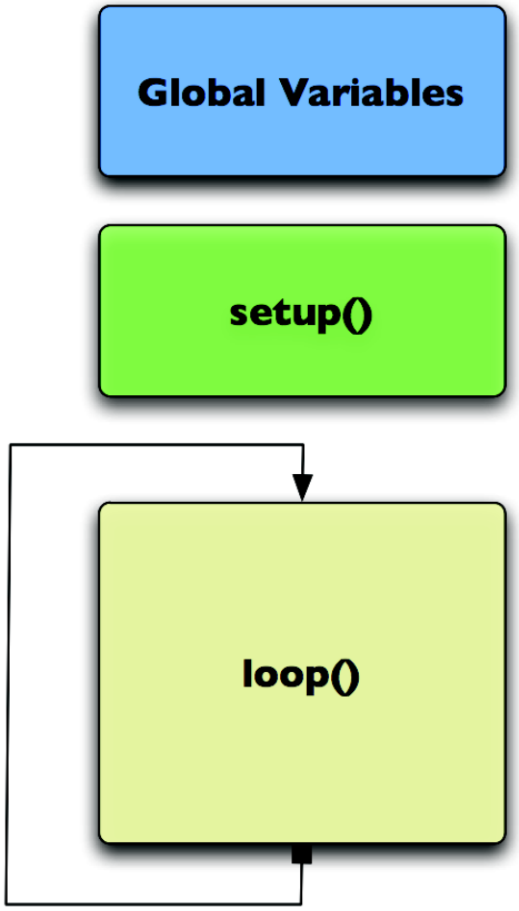
1. Open "\Basic\Blink\Blink.ino"
2. To compile your sketch, click the checkmark
3. Make sure your Arduino is plugged into a USB port
4. Install driver for Arduino
5. Click the arrow to download the program to Arduino

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"

Global Variables

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

Setup

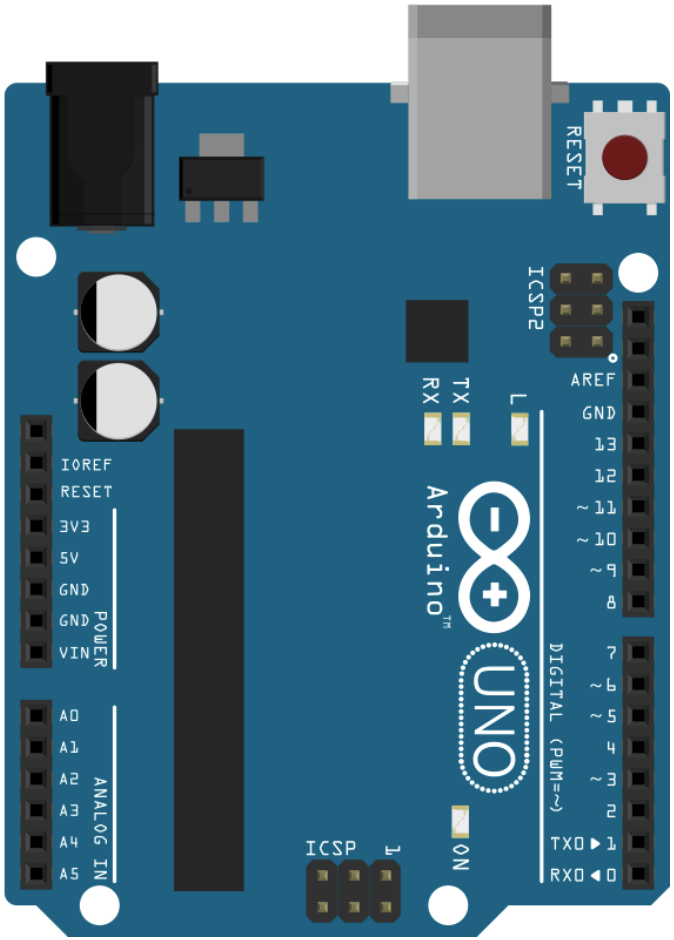
```
// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);    // initialize the digital pin as an output
}
```

Loop

```
// 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
}
```

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 32,768
- **long*** – a variable which stores an integer number in 4 bytes and has a range of -2,147,483,648 and 2,147,483,648
- **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

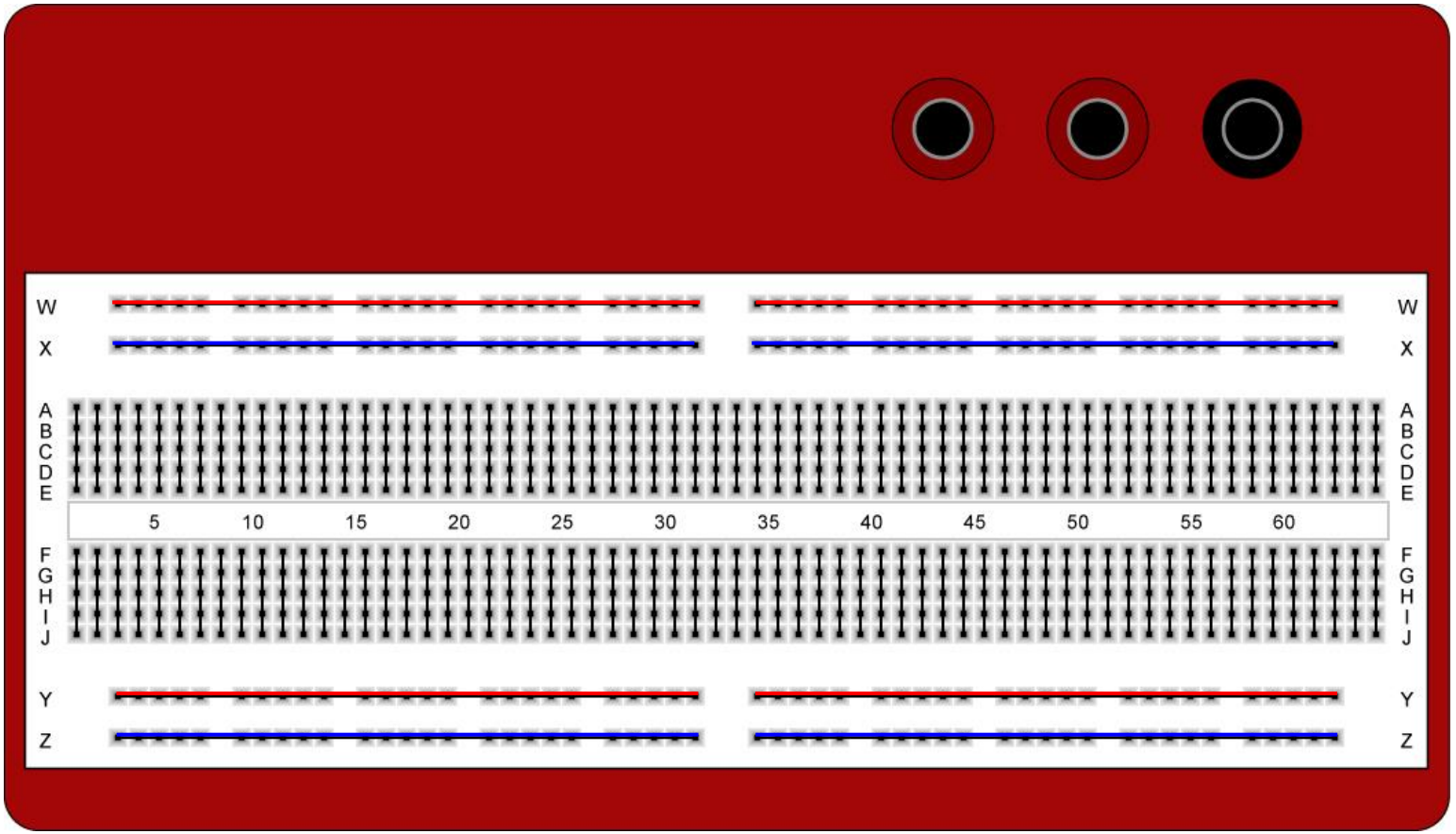
- **=** assignment
- **+** addition
- **-** subtraction
- ***** multiplication
- **/** division
- **%** remainder
- **==** equal to
- **!=** not equal to
- **<** less than
- **<=** less than or equal to
- **>** greater than
- **>=** greater than or equal to

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

Control Structures and Further

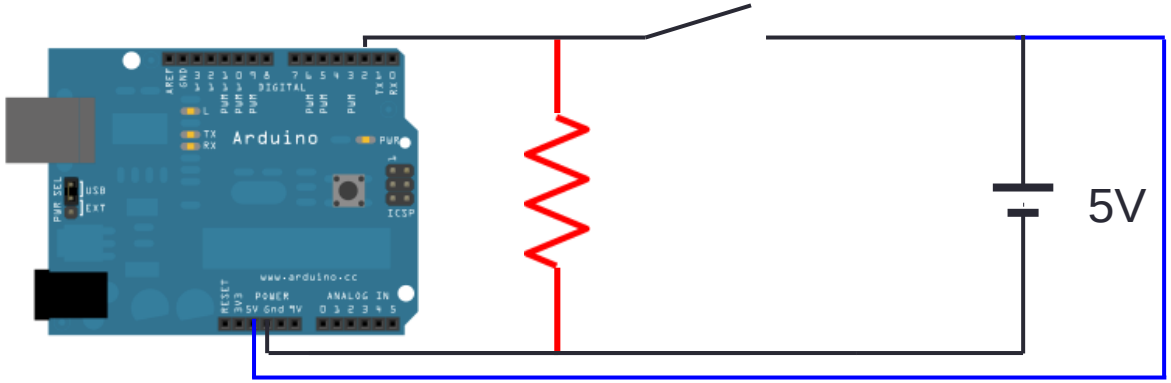
- Structure:
  - if (condition) { } else { }
  - for (int i=0; i < #repeats; i++) { }
  - while (condition) { }
  - switch (variable) { case X: case Y: default: }
- Others:
  - **;** the end of a line of code
  - ***//*** single line comment
  - ***/* */*** multiline comment

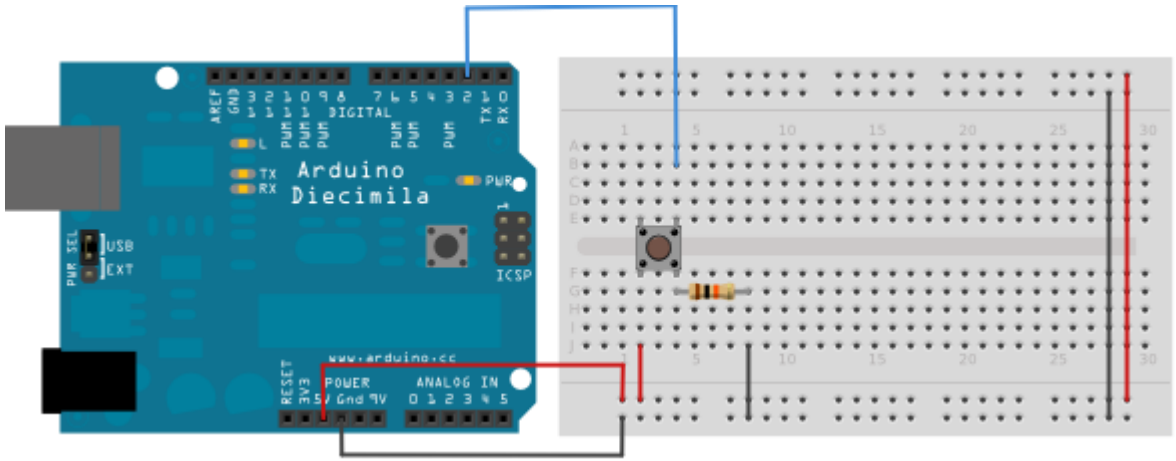
Typical Solderless Breadboard



Example – LED Control Using Button

- Voltage measurement – 0 or 5 V?





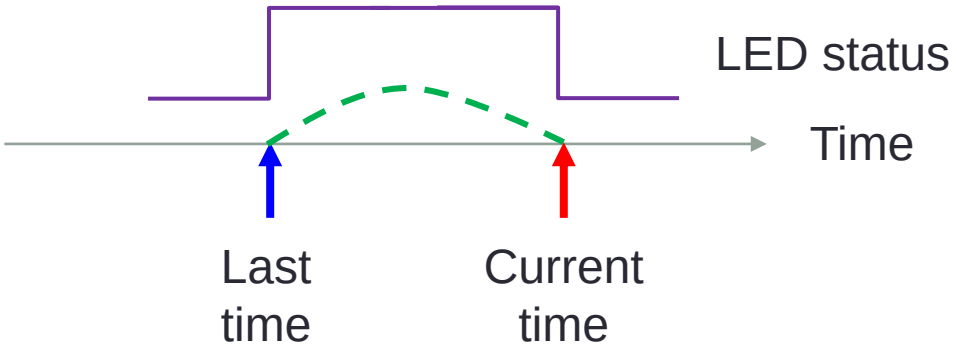
Link

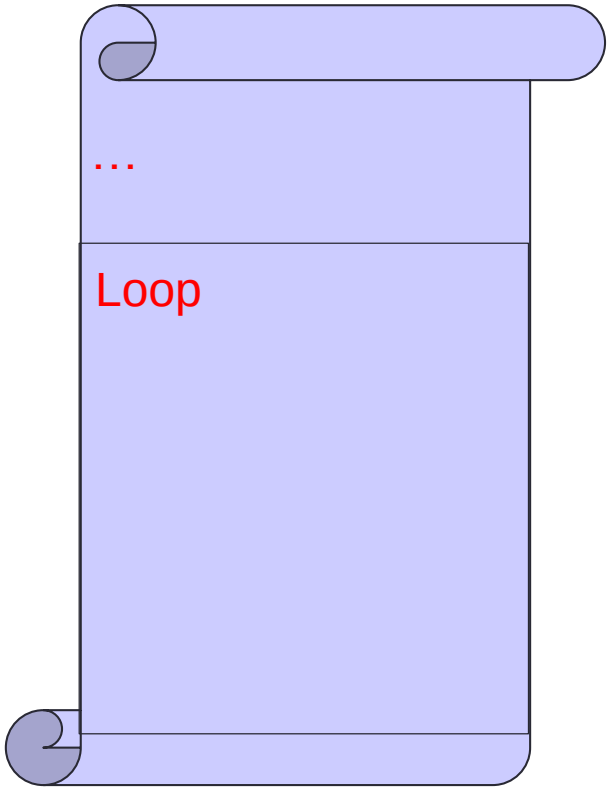
Sketch Code – Button

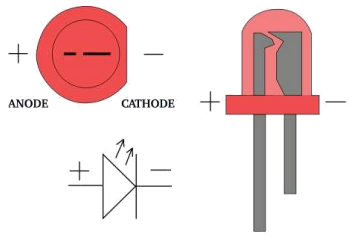
```
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin
int buttonState = 0;         // variable for reading the pushbutton status

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?





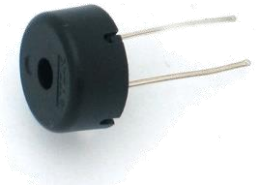


Link

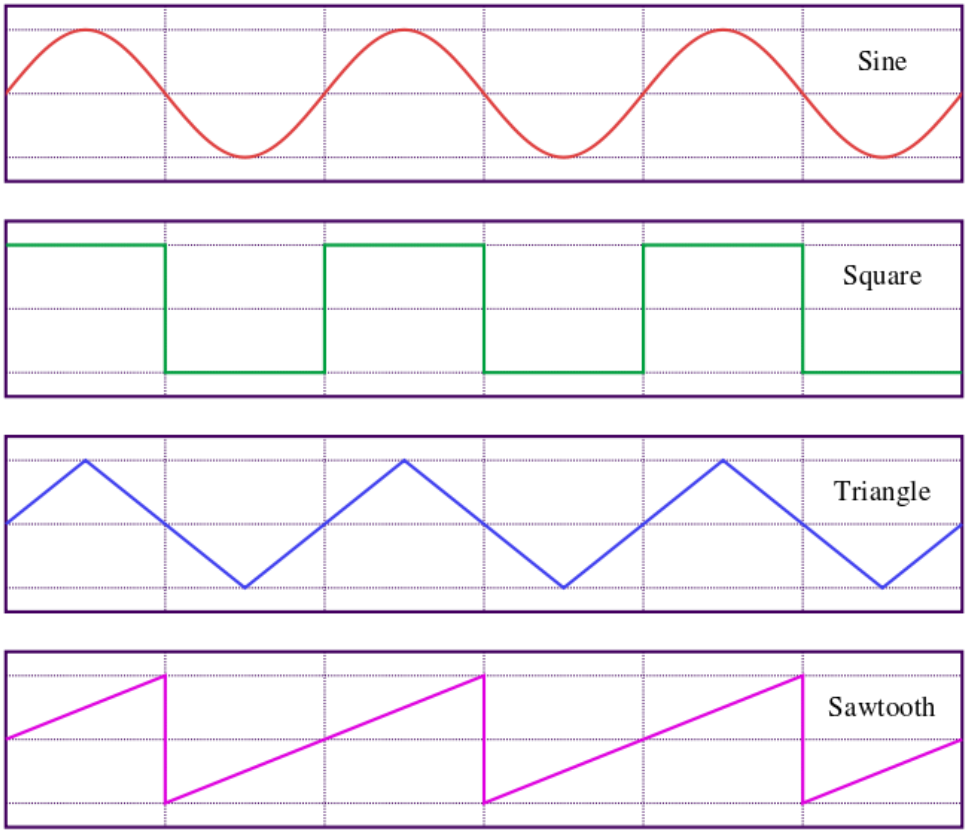
Sketch Code – Blink Without Delay

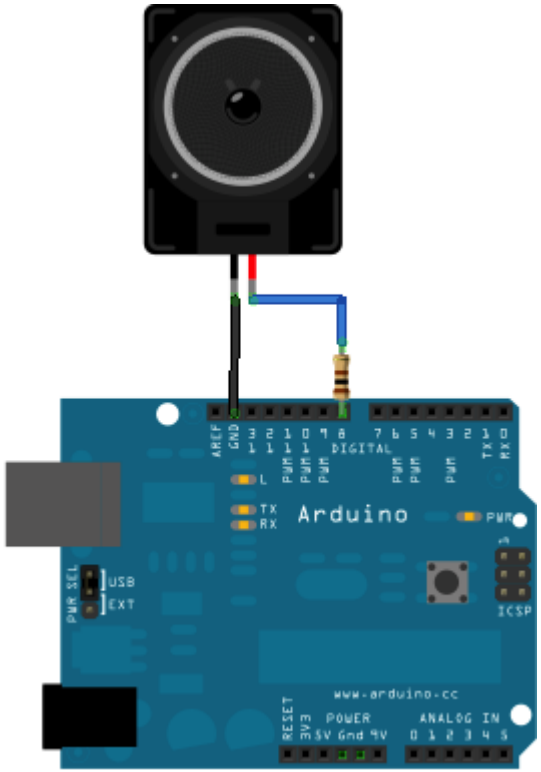
```
const int ledPin =  13;          // the number of the LED pin
int ledState = LOW;              // ledState used to set the LED
long previousMillis = 0;         // will store last time LED was updated
long interval = 1000;            // interval at which to blink (milliseconds)

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?





Link

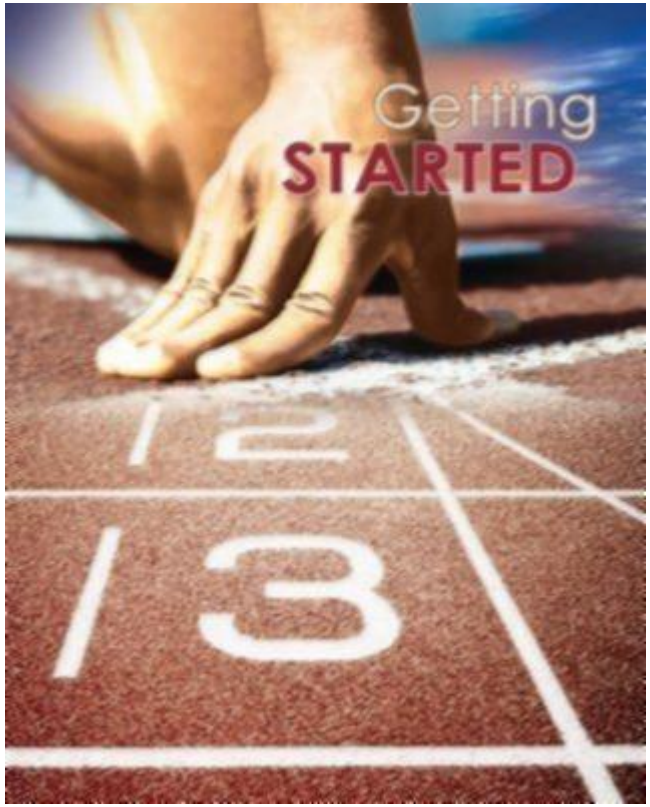
Sketch Code – ToneMelody

```
#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?

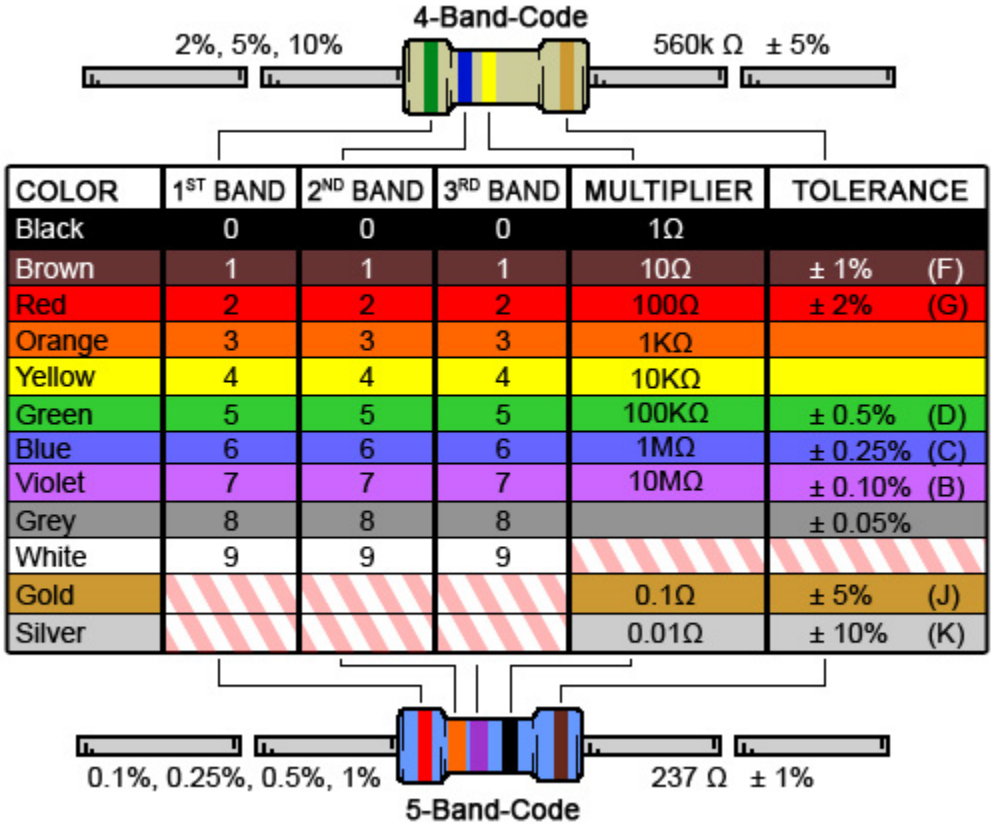
- There are pins of the microcontroller that can function as input or output – I/O
- There is a component in the microcontroller that counts the time – timer/counter
- There are libraries of Arduino that we can use to control buzzer

Getting Started



Reference

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



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)