

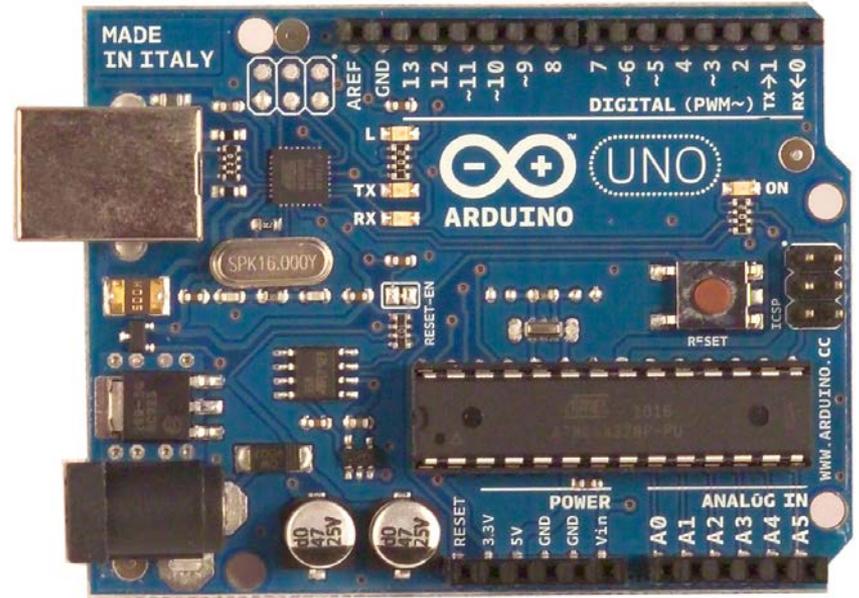
Arduino

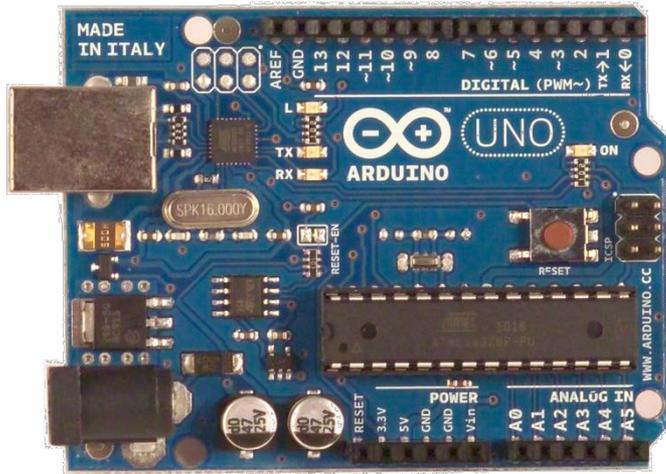
Arduino Workshop

- Short intro Arduino
- Writing a program
- Connecting hardware

Actions:

- Making a LED glow
- Read a button
- State machine

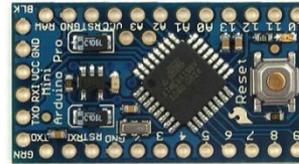




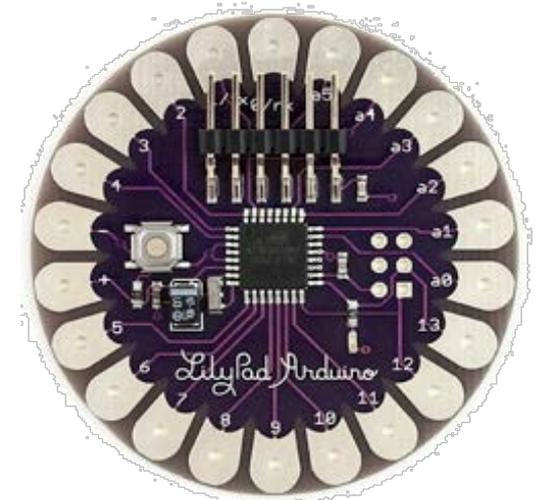
Arduino Uno



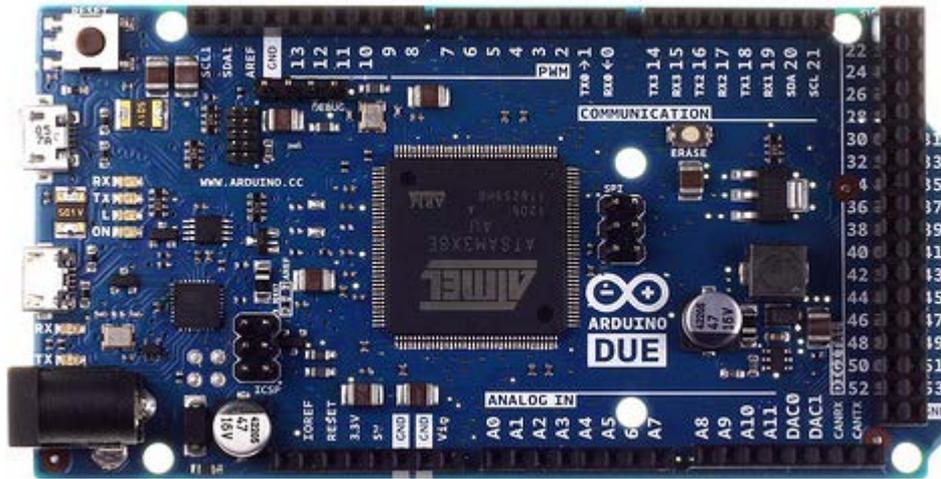
Arduino Nano



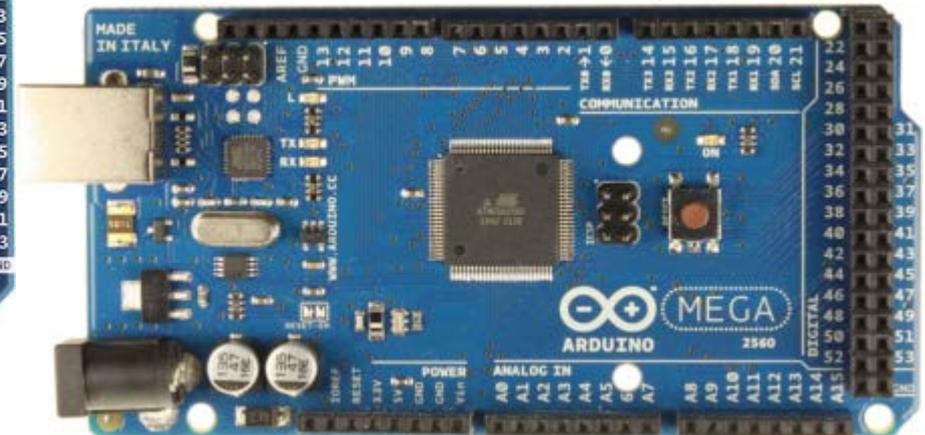
Arduino Pro Mini



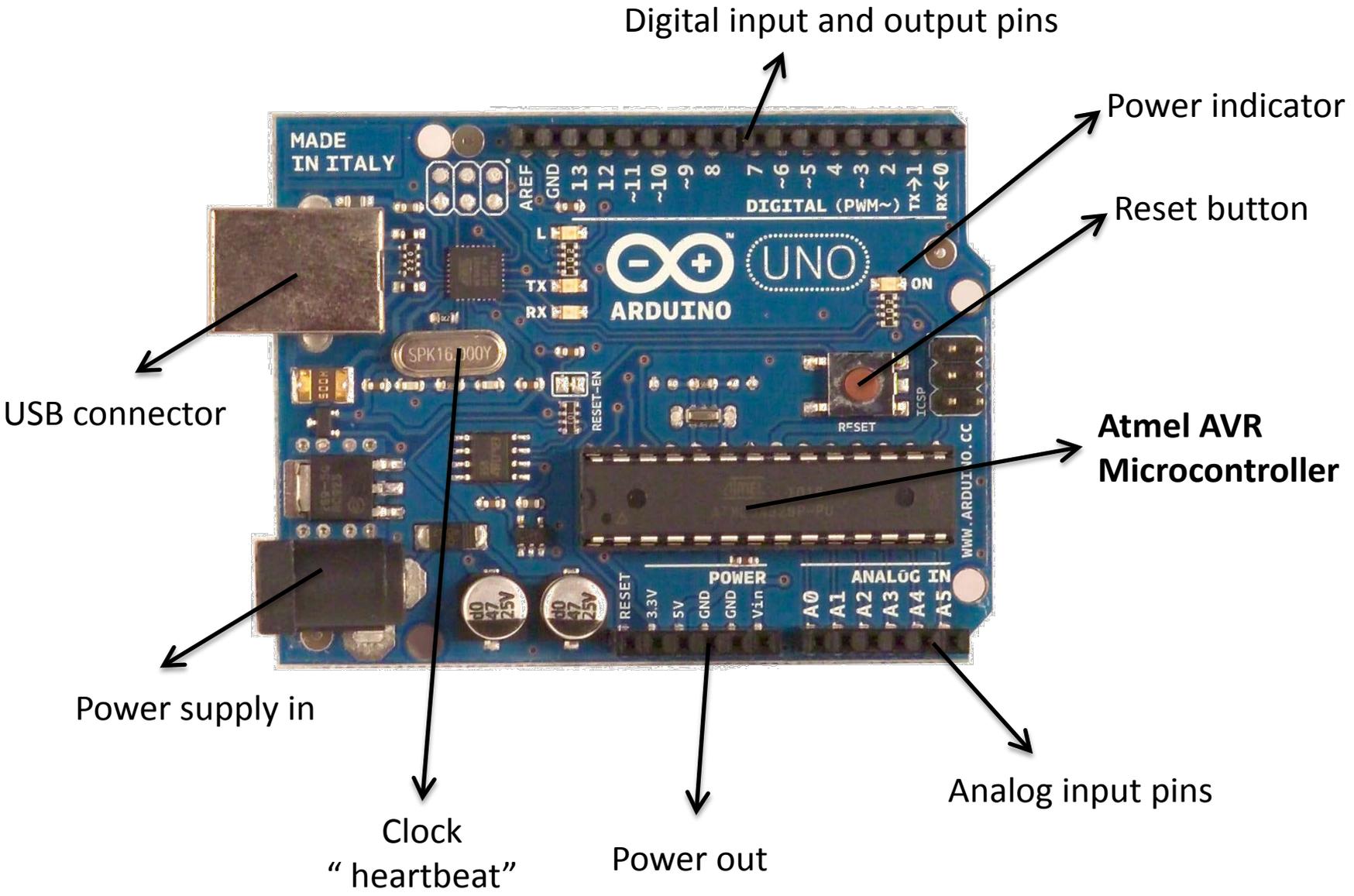
Arduino LilyPad



Arduino DUE (ARM based)



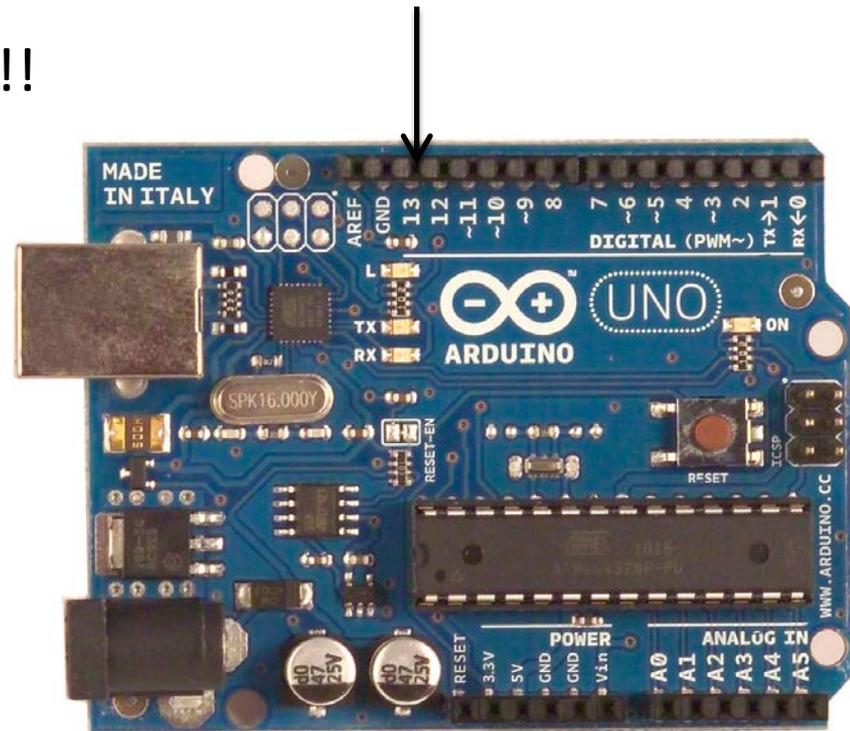
Arduino MEGA



Electronics: Digital Inputs

Digital input:

- 0 Volt: Low
- 5 Volt: High
- 2.5 Volt or floating?: Undefined !!



Electronics: Digital Inputs

External pull up

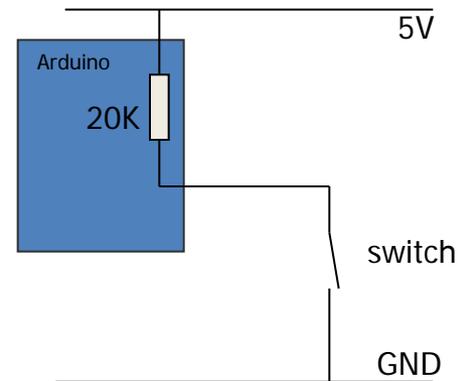
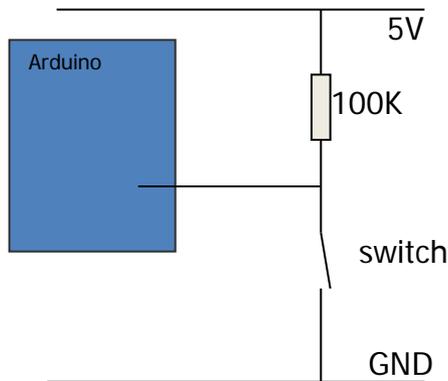
```
pinMode(12, INPUT);  
digitalWrite(12, LOW);  
int a = digitalRead(12);
```

Internal pull up

```
pinMode(12, INPUT);  
digitalWrite(12, HIGH);  
int a = digitalRead(12);
```

Internal pull up (new)

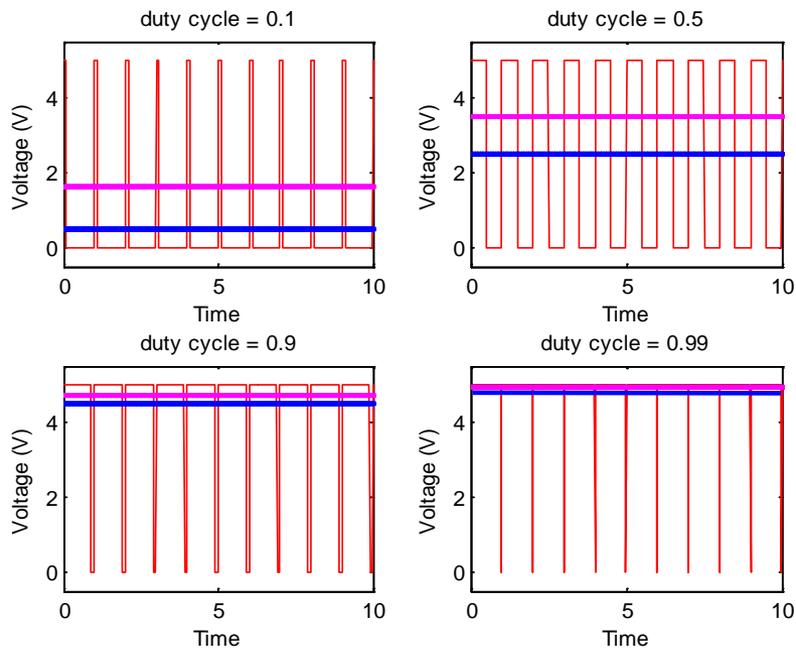
```
pinMode(12, INPUT_PULLUP);  
int a = digitalRead(12);
```



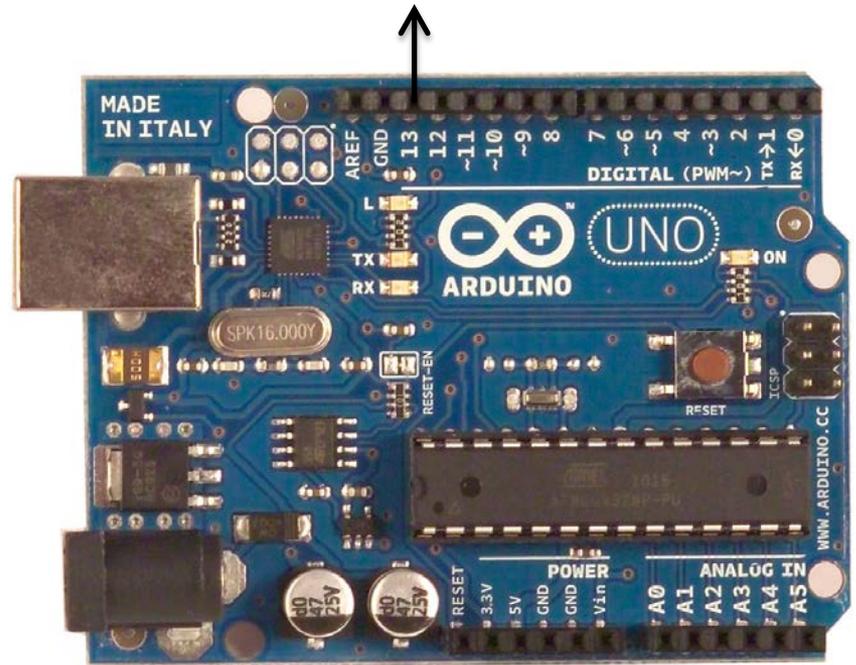
Electronics: Digital Outputs

Digital output:

- 0 Volt: Low
- 5 Volt: High
- 2.5 Volt? “PWM” on specific pins (3, 5, 6, 9, 10, 11)



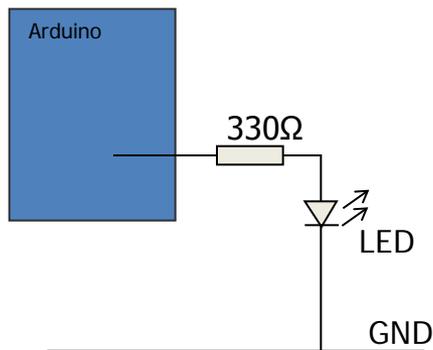
PWM and average voltage for a number of duty cycles



Electronics: Digital Outputs

```
pinMode(10, OUTPUT);  
digitalWrite(10, LOW);  
digitalWrite(10, HIGH);
```

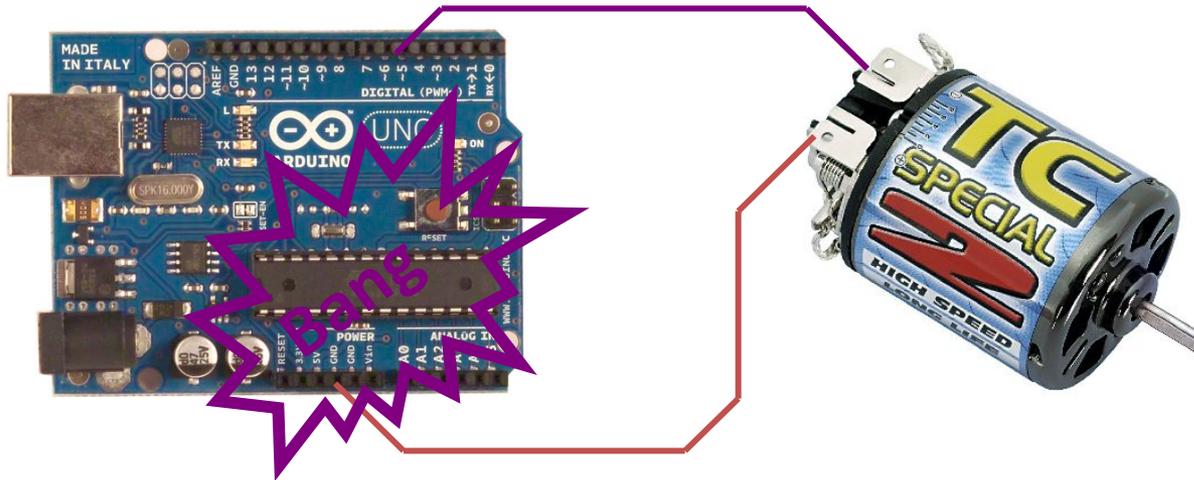
```
pinMode(10, OUTPUT);  
analogWrite(10, 104);
```



Electronics: More Digital Outputs

Digital output:

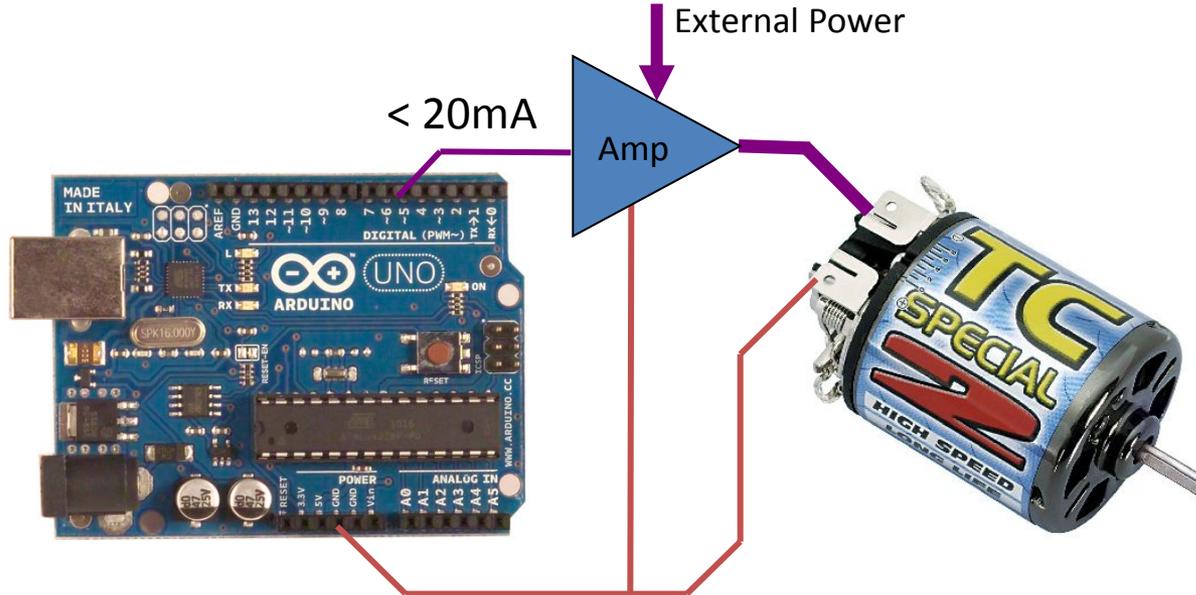
- 0 Volt: Low
- 5 Volt: High



Electronics: More Digital Outputs

Digital output:

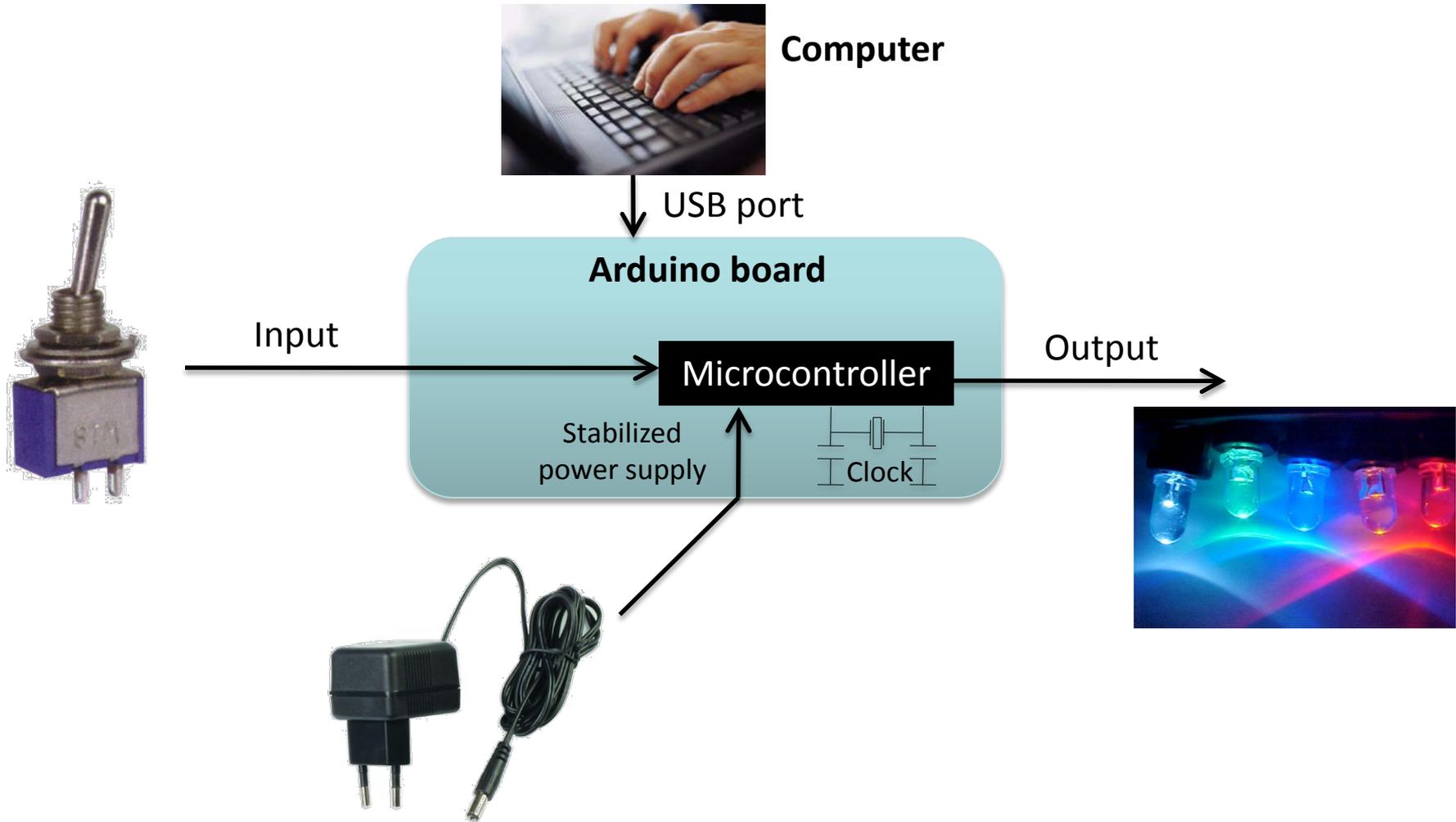
- 0 Volt: Low
- 5 Volt: High



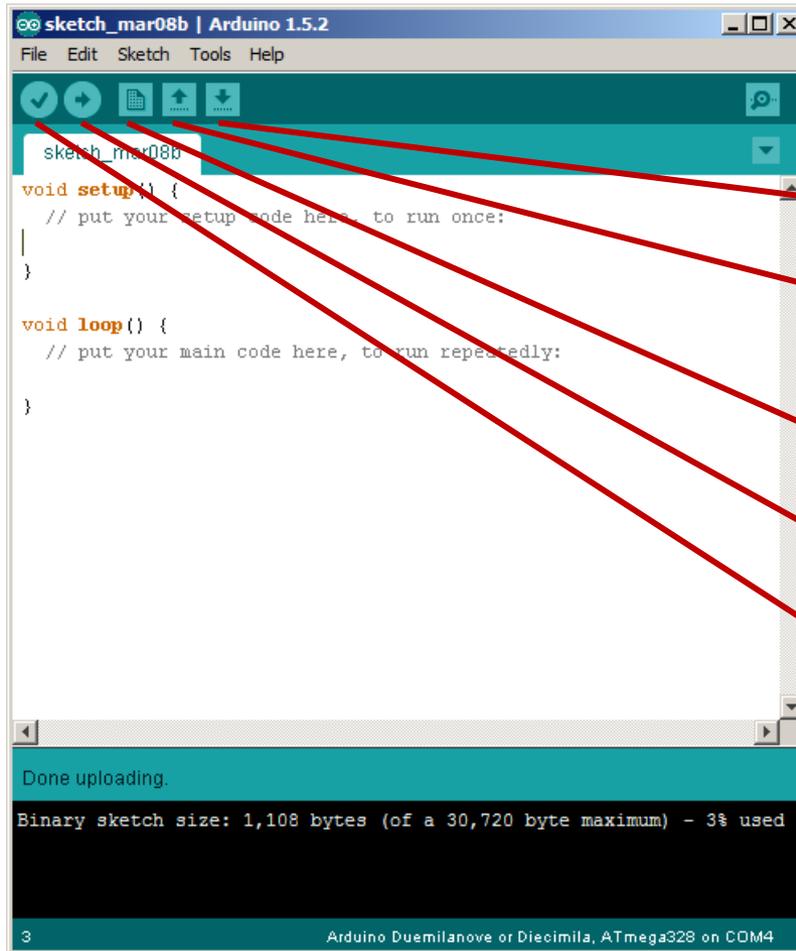
Some Hardware Limitations (Arduino Uno)

- The Uno can be powered via **USB** or **external power supply**; power source is selected automatically. Usually a PC **USB port is power limited to 100 mA max.**
- External (non-USB) power can come from an **AC-to-DC adapter** or **battery**. The adapter can be connected into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
- Operating Voltage 5 V
- Input Voltage (**recommended**) **7-12 V**, (limits) 6-20 V. If the Uno is supplied with *less than 7 V*, the 5 V pin may supply less than five volts and the board may be *unstable*. If using more than 12 V, the voltage regulator may overheat and damage the board
- DC Current per **I/O pin max. 40 mA**
- DC Current for **3.3 V pin max. 50 mA**
- Depending on supply voltage and current usage (including the current for the processor) **the voltage regulator may overheat** and stop working until it has cooled down. Generally driving one servo or a few normal LEDs is the maximum.

The development system



The Arduino Programming Environment



Save Sketch

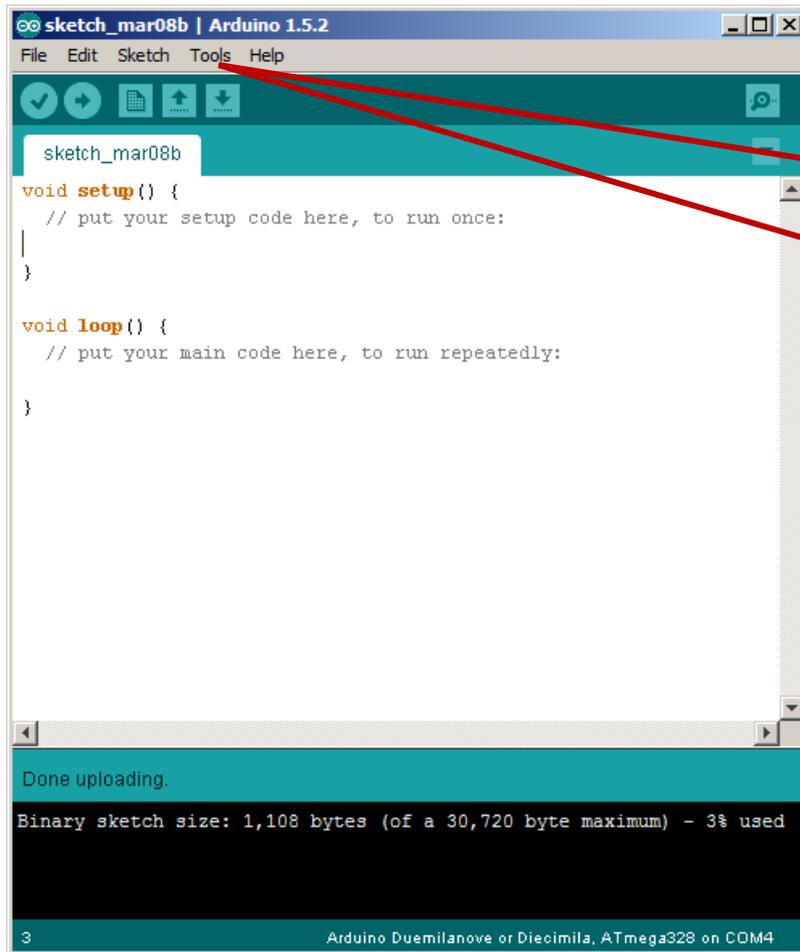
Open Sketch
(old: *.pde, new: *.ino)

New Sketch

Compile and Upload

Verify (Compile)

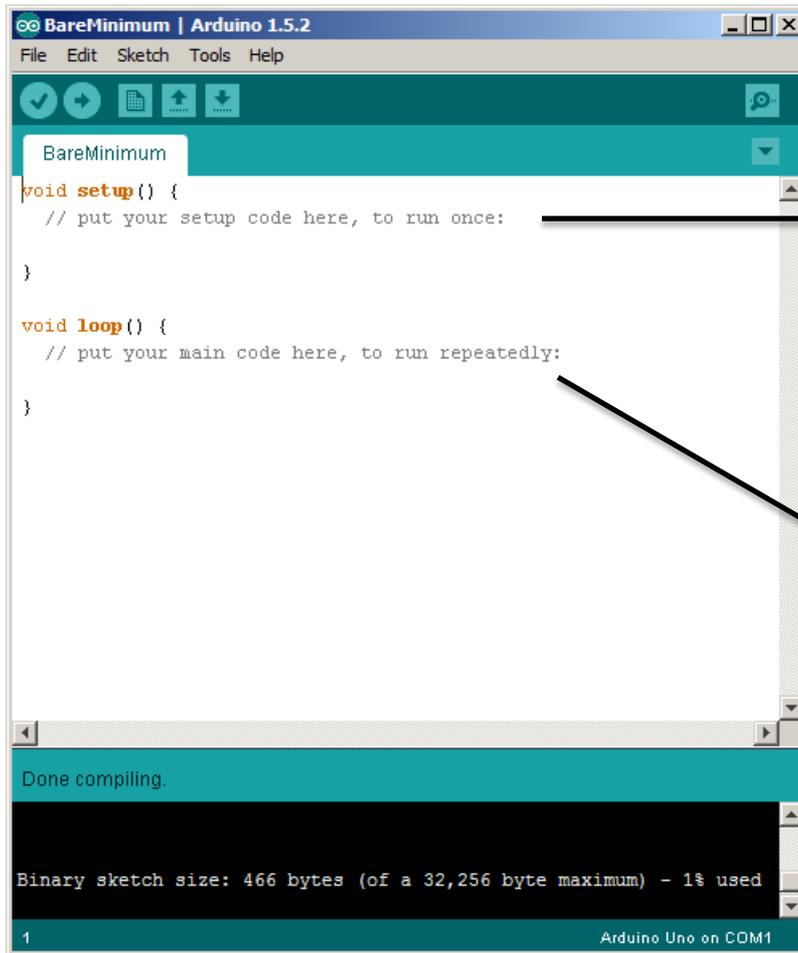
Settings



Board

Serial port

The program



```
BareMinimum | Arduino 1.5.2
File Edit Sketch Tools Help

BareMinimum
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}

Done compiling.

Binary sketch size: 466 bytes (of a 32,256 byte maximum) - 1% used

1 Arduino Uno on COM1
```

First, tell Arduino what it is:

- Declare pins as input or output
 - Set global variables
- This is done only once

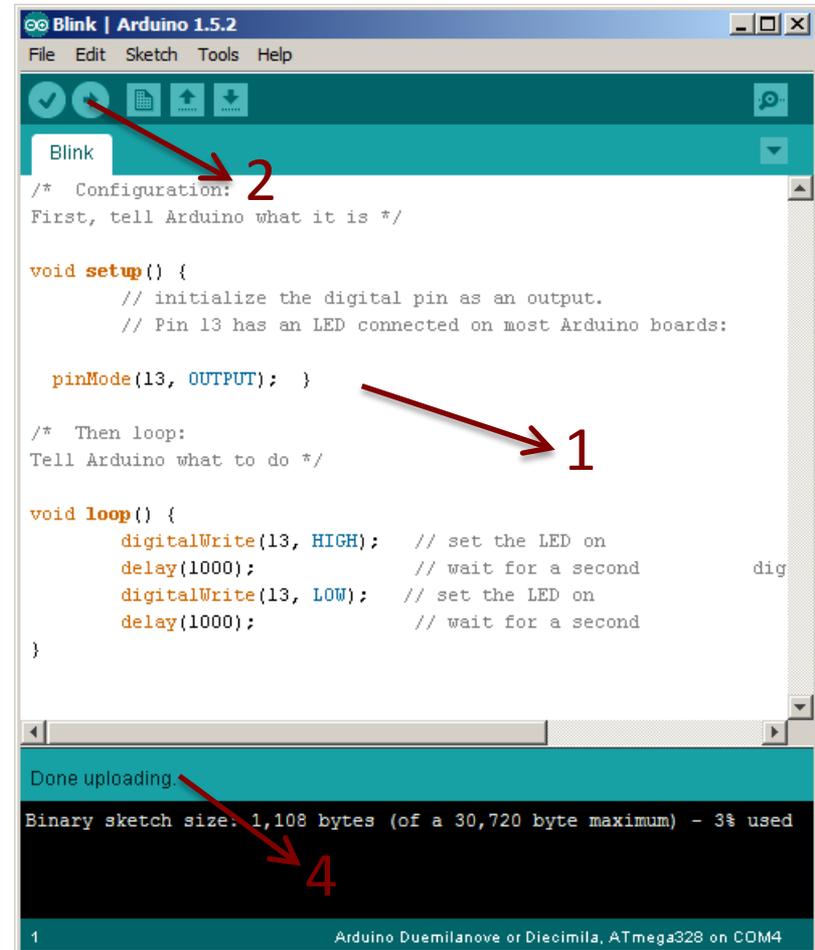
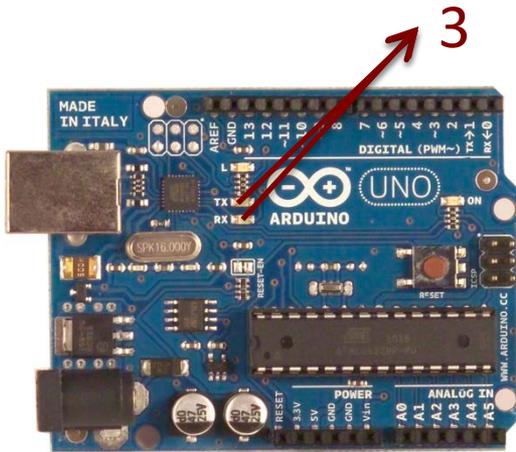
Then tell it what to do:

- Read inputs
- Do calculations
- Set outputs

This is done continuously,
in a loop

Compiling and uploading code

1. Type code in text window
2. Push 'upload' button
3. Check if TX and RX Leds are blinking rapidly
4. If the 'Done uploading' message displays, the Arduino is ready and the program starts.



```
Arduino 1.5.2
File Edit Sketch Tools Help

Blink
/* Configuration:
First, tell Arduino what it is */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:

  pinMode(13, OUTPUT); }

/* Then loop:
Tell Arduino what to do */

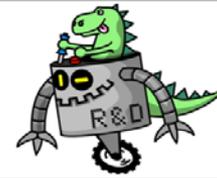
void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED on
  delay(1000);           // wait for a second
}

Done uploading
Binary sketch size: 1,108 bytes (of a 30,720 byte maximum) - 3% used

1 Arduino Duemilanove or Diecimila, ATmega328 on COM4
```

ARDUINO CHEAT SHEET V.02c

Mostly taken from the extended reference:
<http://arduino.cc/en/Reference/Extended>
 Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace



Structure
 void setup() void loop()

Control Structures
 if (x<5) { } else { }
 switch (myvar) {
 case 1:
 break;
 case 2:
 break;
 default:
 }
 for (int i=0; i <= 255; i++) { }
 while (x<5) { }
 do { } while (x<5);
 continue; //Go to next in do/for/while loop
 return x; // Or "return;" for voids.
 goto // considered harmful :-)

Further Syntax
 // (single line comment)
 /* (multi-line comment) */
 #define DOZEN 12 //Not baker's!
 #include <avr/pgmspace.h>

General Operators
 = (assignment operator)
 + (addition) - (subtraction)
 * (multiplication) / (division)
 % (modulo)
 == (equal to) != (not equal to)
 < (less than) > (greater than)
 <= (less than or equal to)
 >= (greater than or equal to)
 && (and) || (or) !(not)

Pointer Access
 & reference operator
 * dereference operator

Bitwise Operators
 & (bitwise and) | (bitwise or)
 ^ (bitwise xor) ~ (bitwise not)
 << (bitshift left) >> (bitshift right)

Compound Operators
 ++ (increment) -- (decrement)
 += (compound addition)
 -= (compound subtraction)
 *= (compound multiplication)
 /= (compound division)
 &= (compound bitwise and)
 |= (compound bitwise or)

Constants
 HIGH | LOW
 INPUT | OUTPUT
 true | false
 143 // Decimal number
 0173 // Octal number
 0b11011111 // Binary
 0x7B // Hex number
 7U // Force unsigned
 10L // Force long
 15UL // Force long unsigned
 10.0 // Forces floating point
 2.4e5 // 240000

Data Types
 void
 boolean (0, 1, false, true)
 char (e.g. 'a' -128 to 127)
 unsigned char (0 to 255)
 byte (0 to 255)
 int (-32,768 to 32,767)
 unsigned int (0 to 65535)
 word (0 to 65535)
 long (-2,147,483,648 to 2,147,483,647)
 unsigned long (0 to 4,294,967,295)
 float (-3.4028235E+38 to 3.4028235E+38)
 double (currently same as float)
 sizeof(myint) // returns 2 bytes

Strings
 char S1[15];
 char S2[8]={'a','r','d','u','i','n','o'};
 char S3[8]={'a','r','d','u','i','n','o','\0'};
 //Included \0 null termination
 char S4[] = "arduino";
 char S5[8] = "arduino";
 char S6[15] = "arduino";

Arrays
 int myInts[6];
 int myPins[] = {2, 4, 8, 3, 6};
 int mySensVals[6] = {2, 4, -8, 3, 2};

Conversion
 char() byte()
 int() word()
 long() float()

Qualifiers
 static // persists between calls
 volatile // use RAM (nice for ISR)
 const // make read-only
 PROGMEM // use flash

Digital I/O
 pinMode(pin, [INPUT,OUTPUT])
 digitalWrite(pin, value)
 int digitalRead(pin)
 //Write High to inputs to use pull-up res

Analog I/O
 analogReference([DEFAULT,INTERNAL,EXTERNAL])
 int analogRead(pin) //Call twice if switching pins from high Z source.
 analogWrite(pin, value) // PWM

Advanced I/O
 tone(pin, freqhz)
 tone(pin, freqhz, duration_ms)
 noTone(pin)
 shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
 unsigned long pulseIn(pin, [HIGH,LOW])

Time
 unsigned long millis() // 50 days overflow.
 unsigned long micros() // 70 min overflow
 delay(ms)
 delayMicroseconds(us)

Math
 min(x, y) max(x, y) abs(x)
 constrain(x, minval, maxval)
 map(val, fromL, fromH, toL, toH)
 pow(base, exponent) sqrt(x)
 sin(rad) cos(rad) tan(rad)

Random Numbers
 randomSeed(seed) // Long or int
 long random(max)
 long random(min, max)

Bits and Bytes
 lowByte() highByte()
 bitRead(x,bitn) bitWrite(x,bitn,bit)
 bitSet(x,bitn) bitClear(x,bitn)
 bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
 attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
 detachInterrupt(interrupt)
 interrupts()
 noInterrupts()

Libraries:

Serial.
 begin([300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200])
 end()
 int available()
 int read()
 flush()
 print()
 println()
 write()

EEPROM (#include <EEPROM.h>)
 byte read(intAddr)
 write(intAddr,myByte)

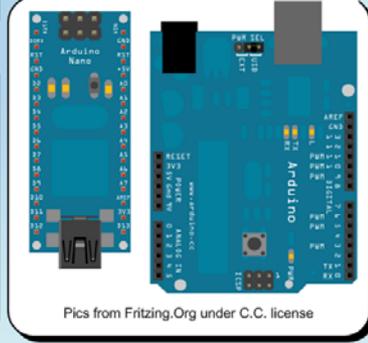
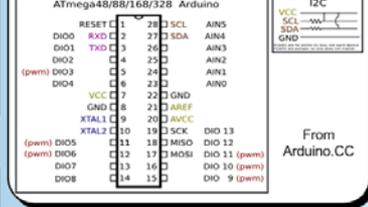
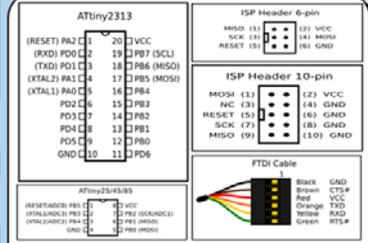
Servo (#include <Servo.h>)
 attach(pin , [min_uS, max_uS])
 write(angle) // 0-180
 writeMicroseconds(uS) //1000-2000,
 1500 is midpoint
 read() // 0-180
 attached() //Returns boolean
 detach()

SoftwareSerial(RxPin,TxPin)
 // #include<SoftwareSerial.h>
 begin(longSpeed) // up to 9600
 char read() // blocks till data
 print(myData) or println(myData)

Wire (#include <Wire.h>) // For I2C
 begin() // Join as master
 begin(addr) // Join as slave @ addr
 requestFrom(address, count)
 beginTransmission(addr) // Step 1
 send(mybyte) // Step 2
 send(char * mystring)
 send(byte * data, size)
 endTransmission() // Step 3
 byte available() // Num of bytes
 byte receive() //Return next byte
 onReceive(handler)
 onRequest(handler)

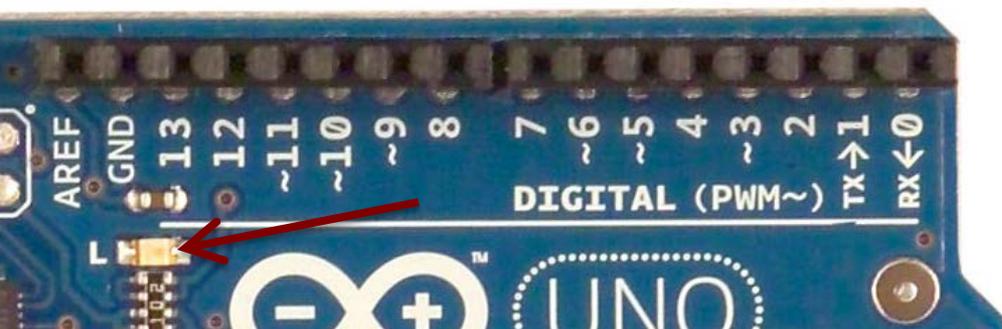
	ATmega168	ATmega328	ATmega1280
Flash (2k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilaneve/ Nano/Pro/Mini	Mega
# of IO	14 + 6 analog (Nano has 14+8)	54 + 18 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 18 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,18,18 (IRQ0- IRQ5)
PWM pins	5, 8 - Timer 0 9, 10 - Timer 1 3, 11 - Timer 2	0-13 53 - SS 51 - MCSI 50 - MISO 52 - SCK 20 - SDA 21 - SCL
I2C	Analog5 - SDA Analog6 - SCK	21 - SCL



Blink

```
/* comment */  
  
void setup() {  
    // initialize the digital pin as an output.  
    // Pin 13 has a LED connected on most Arduino boards:  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);    // switch LED on  
    delay(1000);              // wait 1000 ms  
    digitalWrite(13, LOW);   // switch LED off  
    delay(1000);              // wait 1000 ms  
}
```



Blink with button

- The period of “Blink” is $2 \times 1000 \text{ ms} = 2 \text{ s}$
- Now we want: if we push a button, the period should become 0.5 s

- To do:
 - Connect a button
 - Read a button
 - Change the “1000” in “delay” depending on the button

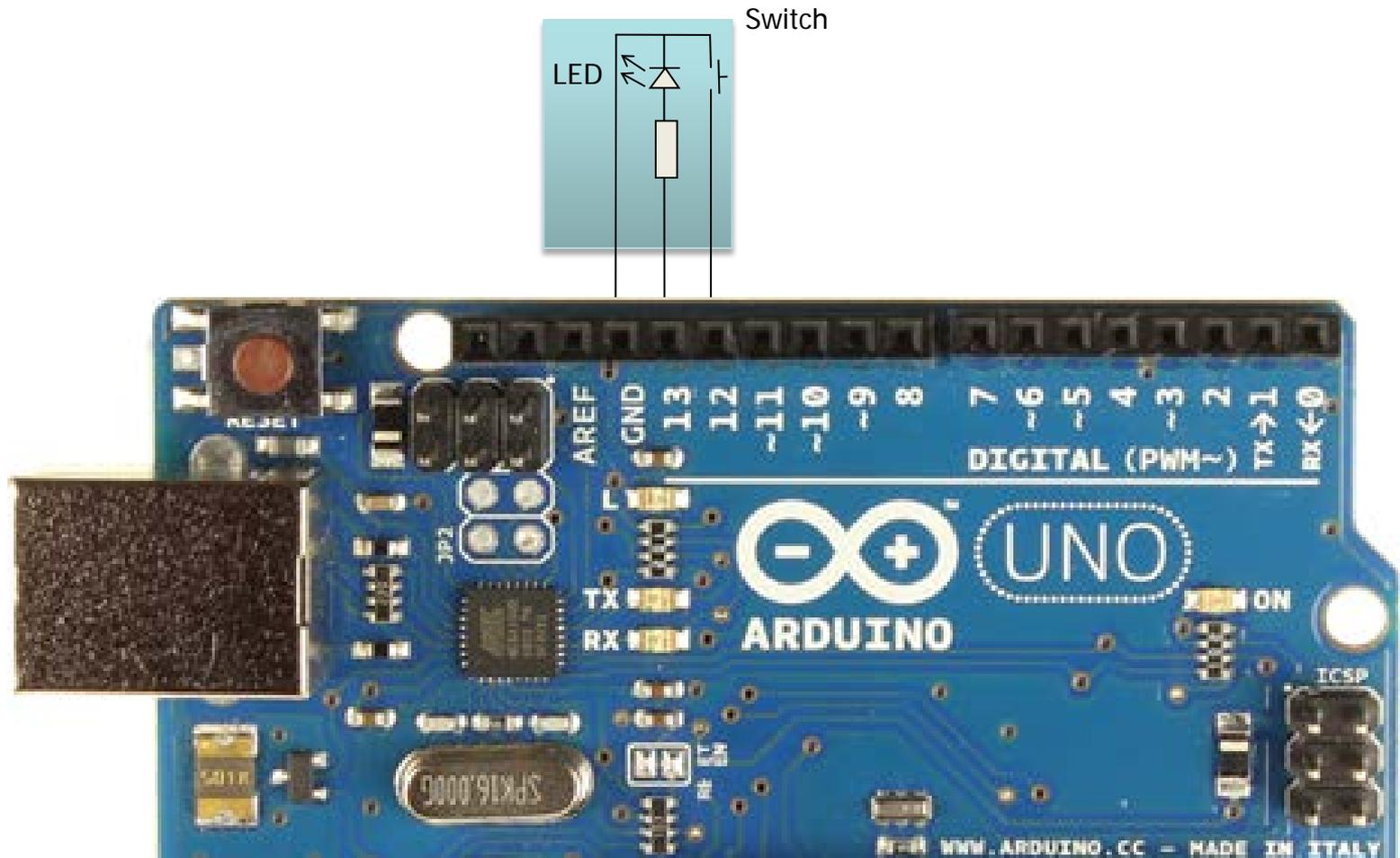
Blink and read button

```
/* comment */
```

```
void setup() {  
    // initialize digital pin 13 as an output.  
    // Pin 13 has a LED connected on most Arduino boards:  
    pinMode(13, OUTPUT);  
    pinMode(12, INPUT);           // initialize digital pin 12 as input.  
    digitalWrite(12, HIGH);      // activate pull-up resistor  
}
```

```
void loop() {  
    digitalWrite(13, HIGH);      // switch LED on  
    delay(1000);                 // wait one second  
    digitalWrite(13, LOW);       // switch LED off  
    delay(1000);                 // wait one second  
}
```

Hardware for Blink with button



```
void setup() {  
    // initialize the digital pin 13 as an output.  
    // Pin 13 has a LED connected on most Arduino boards:  
    pinMode(13,OUTPUT);  
    pinMode(12,INPUT);           // initialize digital pin 12 as input.  
    digitalWrite(12,HIGH);      // activate pull-up resistor  
}
```

```
void loop() {  
    if (digitalRead(12) == HIGH) {  
        // Do something  
    }  
    else {  
        // Do something else  
    }  
    digitalWrite(13,HIGH);      // switch LED on  
    delay(1000);                // wait one second  
    digitalWrite(13,LOW);       // switch LED off  
    delay(1000);                // wait one second  
}
```

```
int wait = 1000;
```

```
void setup() {  
    ...  
}
```

```
void loop() {  
    if (digitalRead(12) == HIGH) {  
        // Do something  
    }  
    else {  
        // Do something else  
    }  
    digitalWrite(13,HIGH); // switch LED on  
    delay(wait);           // wait for wait msec  
    digitalWrite(13,LOW); // switch LED off  
    delay(wait);           // wait for wait msec  
}
```

```

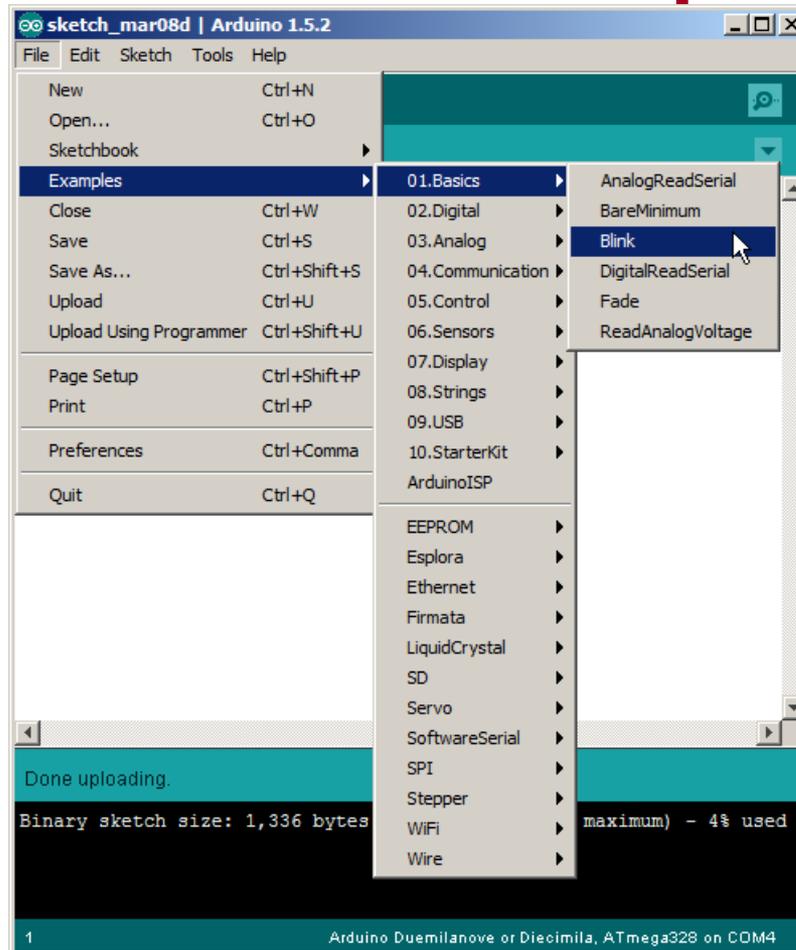
int wait = 1000;

void setup() {
    ...
}

void loop() {
    if (digitalRead(12) == HIGH) {
        wait = 1000; // Wait 1 sec
    }
    else {
        wait = 250; // Wait 0.25 sec
    }
    digitalWrite(13, HIGH); // switch LED on
    delay(wait); // wait for wait msec
    digitalWrite(13, LOW); // switch LED off
    delay(wait); // wait for wait msec
}

```

Available examples



For descriptions see <http://arduino.cc/en/Tutorial/HomePage>

A word on code and schematics



Code:

- Use comments `/* like this */` or `// this`
- Use meaningful variable names
- Use functions with meaningful names

Schematics:

- Draw the complete circuit (correctly!!)

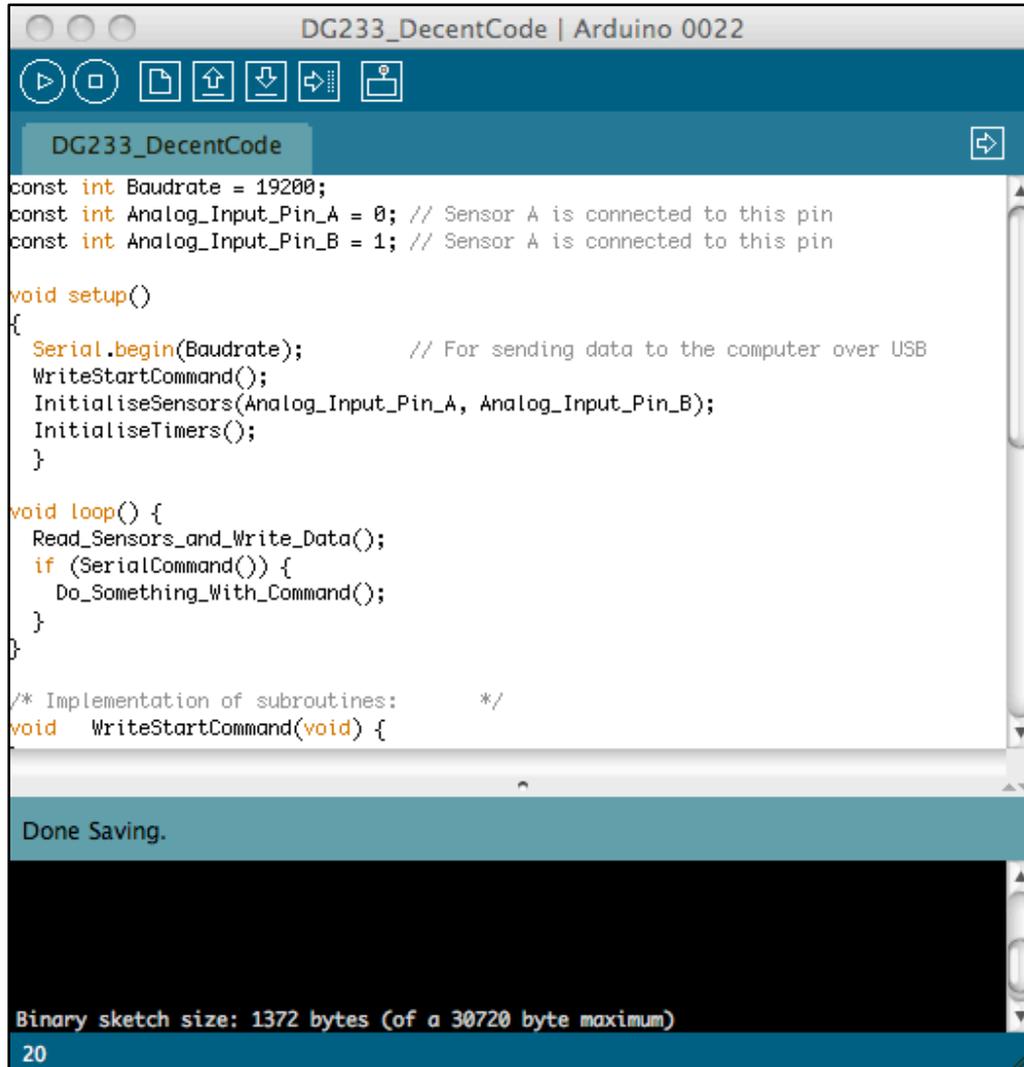
Circuit (Soldering or breadboard)

- Should be structured
- Use color coding in wires
- Think about component placement

Otherwise:

- Debugging becomes impossible
- You have no evidence for your report

Code



```
const int Baudrate = 19200;
const int Analog_Input_Pin_A = 0; // Sensor A is connected to this pin
const int Analog_Input_Pin_B = 1; // Sensor A is connected to this pin

void setup()
{
  Serial.begin(Baudrate);          // For sending data to the computer over USB
  WriteStartCommand();
  InitialiseSensors(Analog_Input_Pin_A, Analog_Input_Pin_B);
  InitialiseTimers();
}

void loop() {
  Read_Sensors_and_Write_Data();
  if (SerialCommand()) {
    Do_Something_With_Command();
  }
}

/* Implementation of subroutines:      */
void WriteStartCommand(void) {
```

Done Saving.

Binary sketch size: 1372 bytes (of a 30720 byte maximum)

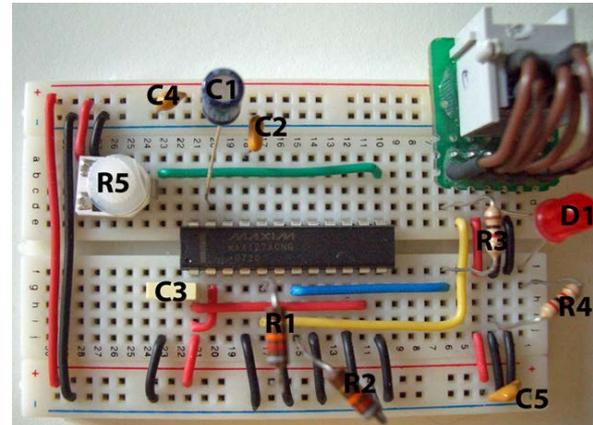
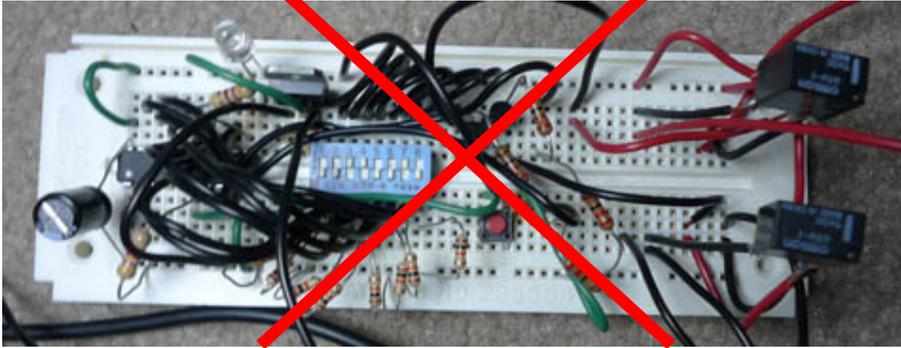
20

Code:

- Clear variable names
- Proper use of local and global variables
- Functions `setup()` and `loop()` are self-explaining
- Proper function names
- Comments

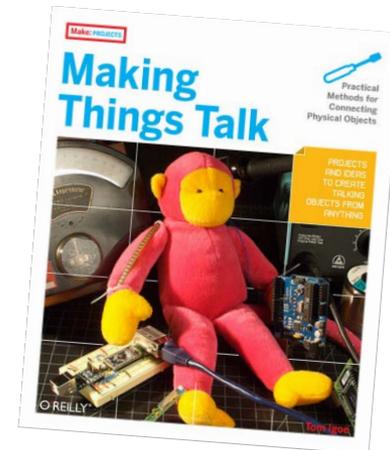
- Examples
 - [Bad](#)
 - [Good](#)

Circuits



How to proceed?

- Examples
- Find new functions in the reference (<http://arduino.cc/en/Reference/HomePage>)
- Arduino playground Manuals and Curriculum (also downloads) (<http://playground.arduino.cc/Main/ManualsAndCurriculum>)
- Libraries
- “Making things talk”: more on connecting to the internet and less on interfacing with electronics



Installation 1: The Environment

Download

Arduino 1.0.3 ([release notes](#)), hosted by [Google Code](#):

- + [Windows](#)
- + [Mac OS X](#)
- + [Linux: 32 bit, 64 bit](#)
- + [source](#)

Next steps

[Getting Started](#)

[Reference](#)

[Environment](#)

[Examples](#)

[Foundations](#)

[FAQ](#)

Download Arduino 1.5 BETA (with support for Arduino Due Board)

If you have the new Due Board you must download the 1.5.2 version. Once you get the software follow [this instruction](#) to get started with the Arduino Due.

WARNING: This software is a beta version, you may encounter bugs or unexpected behaviours. Please discuss any issues in the [Due forum](#)

Arduino 1.5.2 ([release notes](#)):

- + [Windows](#)
- + [Mac OS X](#)
- + [Linux: 32 bit, 64 bit](#)
- + [source](#)

Installation 1: The Environment

Download

Arduino 1.0.3 ([release notes](#)), hosted by [Google Code](#):

- + [Windows](#)
- + [Mac OS X](#)
- + [Linux: 32 bit, 64 bit](#)

Next steps

- [Getting Started](#)
- [Reference](#)
- [Environment](#)
- [Examples](#)
- [Foundations](#)



Getting Started w/ Arduino on Windows

This document explains how to connect your Arduino board to the computer and upload your first sketch.

On this page... (hide)

- + [1 | Get an Arduino board and USB cable](#)
- + [2 | Download the Arduino environment](#)
- + [3 | Connect the board](#)
- + [4 | Install the drivers](#)
- + [5 | Launch the Arduino application](#)
- + [6 | Open the blink example](#)
- + [7 | Select your board](#)
- + [8 | Select your serial port](#)
- + [9 | Upload the program](#)

Installation 2: The COM port

A “Virtual COM port” is needed to communicate with the board.

For the Arduino UNO: plug in the Arduino

- On Mac OS-X, the VCP driver is installed automatically
- On Windows, you have to point to the .inf file in the “drivers” directory of the Arduino software

For the Arduino Duemilanove, you have to point to the FTDI install directory which is in the “drivers” directory as well

<http://arduino.cc/en/Guide/Windows>

<http://arduino.cc/en/Guide/MacOSX>