

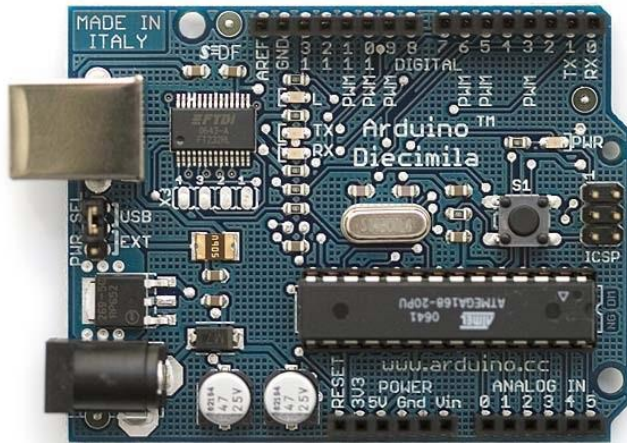
# IMPLEMENTING IOE

Week 2

Assist. Prof. Rassim Suliyeu - SDU 2017

# What is Arduino?

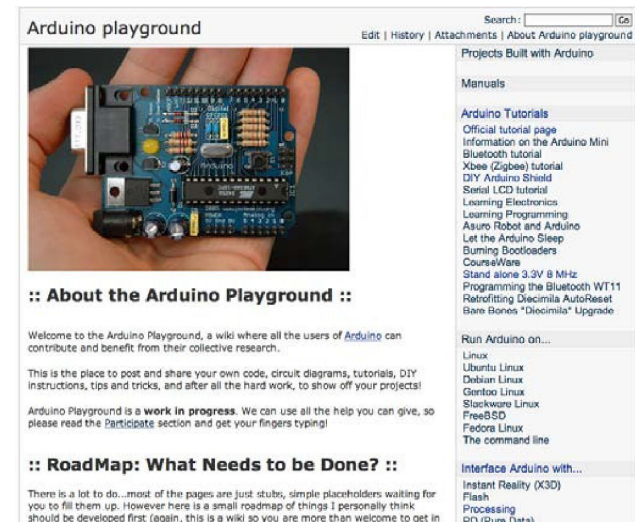
## Physical Device



## IDE

```
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  
}
```

## Community



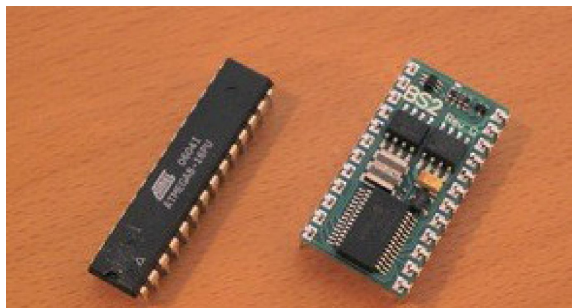
<http://www.arduino.cc>

# Arduino Philosophy and Community

- Open Source Physical Computing Platform
  - “open source hardware”
  - open source: free to inspect & modify
  - physical computing
    - ubiquitous computing
    - pervasive computing
    - ambient intelligence
    - calm computing
    - Spimes
    - Blogjects
    - smart objects
- Community-built
  - Examples wiki (the “playground”) editable by anyone
  - Forums with lots of helpful people

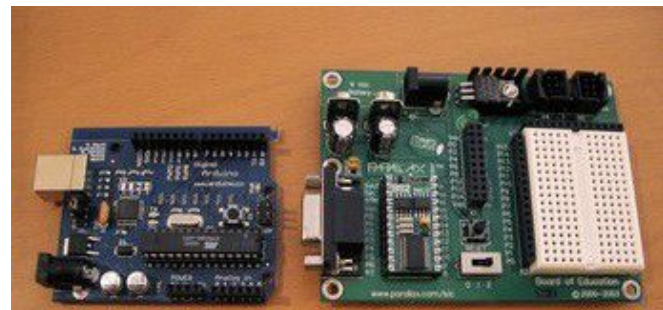
# Arduino Hardware

- Similar to Basic Stamp (if you know of it)
  - but cheaper, faster, & open
- Uses AVR ATmega328 microcontroller chip
  - chip was designed to be used with C language
- The designer of the AVR purposefully arranged its registers and instruction set so that C programs would compile efficiently on it. This is a big deal, compared to previous microcontrollers where C programs were almost always less efficient than a hand-coded assembly language variant.



\$2

\$50



\$20

\$70

# Arduino Hardware Variety

- Openness has its advantages, many different varieties.
- Anyone can build an Arduino work-alike in any form-factor they want



ARDUINO/GENUINO UNO



ARDUINO NANO

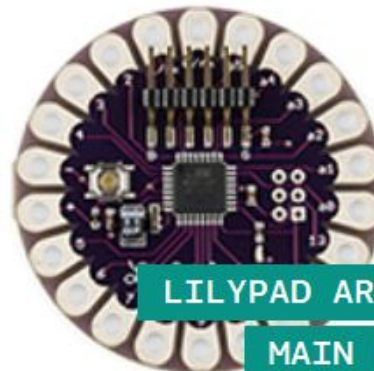


ARDUINO/GENUINO

MEGA 2560



ARDUINO YÚN



LILYPAD ARDUINO

MAIN BOARD



ARDUINO/GENUINO

STARTER KIT

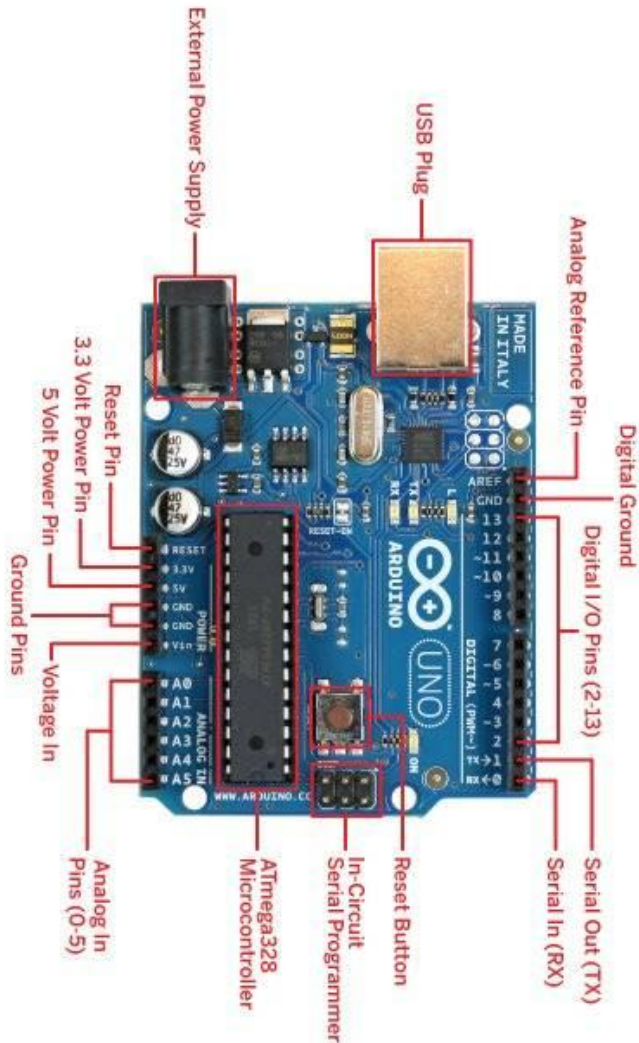
# Arduino Capabilities

- 16 kBytes of Flash program memory
  - 1 kByte of RAM
  - 16 MHz (Apple II: 1 MHz)
  - Inputs and Outputs
    - 14 digital input/output pins
    - 6 analog input pins
    - 6 analog output pins (pseudo-analog, uses PWM , which we'll talk about later)
  - Completely stand-alone: doesn't need a computer once programmed
- \* Don't worry if the above doesn't make sense, you don't really need to know it.

# Arduino Types Comparison

Name	Processor	Operating / Input Voltage	CPU Speed	Analog In/Out	Digital IO / PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	UART
Ethernet	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/4	1	2	32	-
Leonardo	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	1
LilyPad	ATmega328P	2.7-5.5 V	8 MHz	6/0	14/6	0.512	1	16	-
Mega ADK	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	4
Micro	ATmega32U4	5 V / 7-12 V	16 MHz	12/0	20/7	1	2.5	32	1
Mini	ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	1	2	32	-
Nano	ATmega328P	5 V / 7-9 V	16 MHz	8/0	14/6	1	2	32	1
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	1
Yun	AR9331 Linux	5 V	400MHz	12/0	20/7	1	16MB	64 MB	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	2

# Arduino Uno



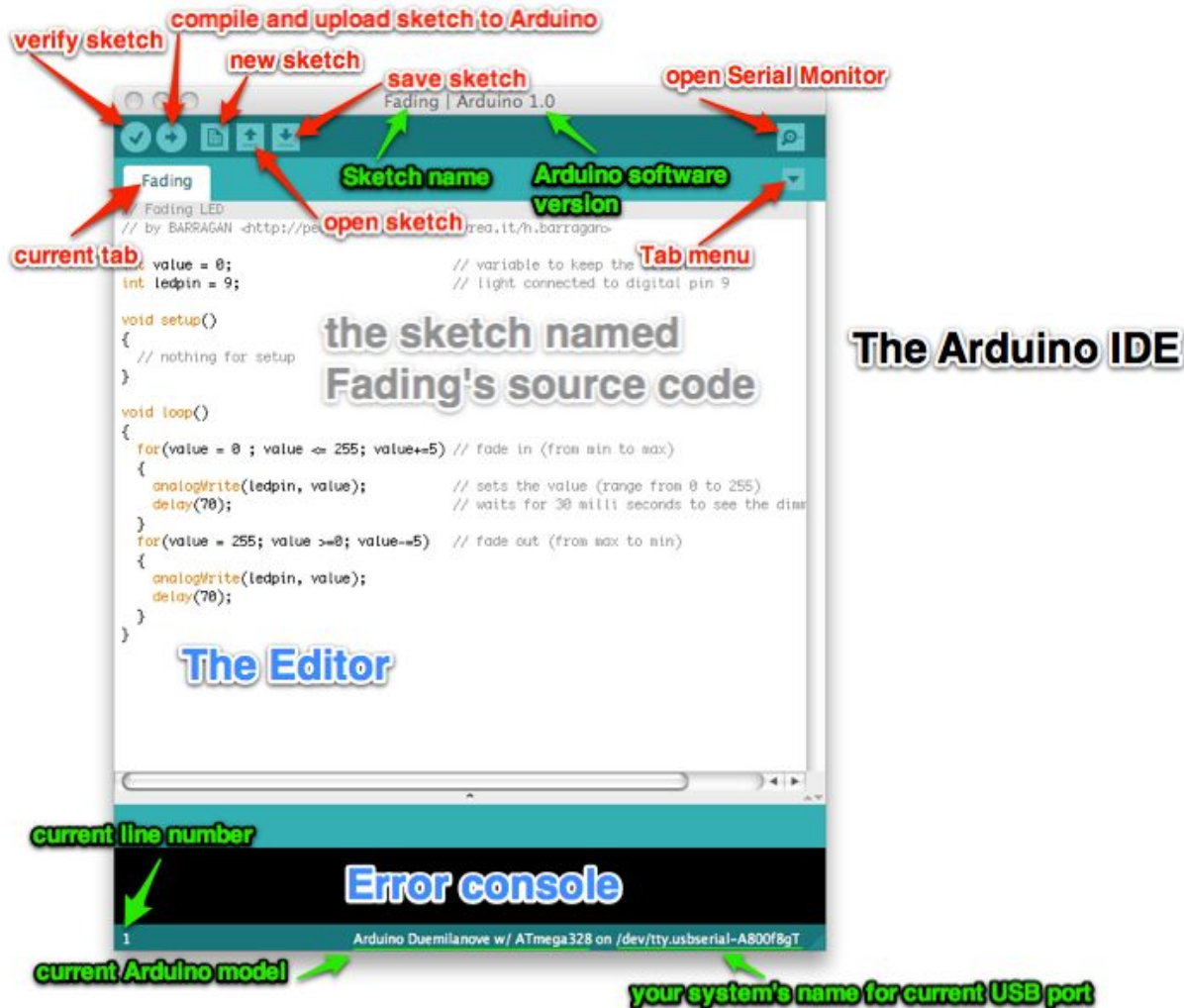
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recom)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (6 PWM)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g



# 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, input from a knob.
- “digital” – value is either HIGH or LOW. (aka on/off, one/zero) e.g. switch state
- “analog” – value ranges, usually from 0-255. e.g. LED brightness, motor speed, etc.

# Arduino Software



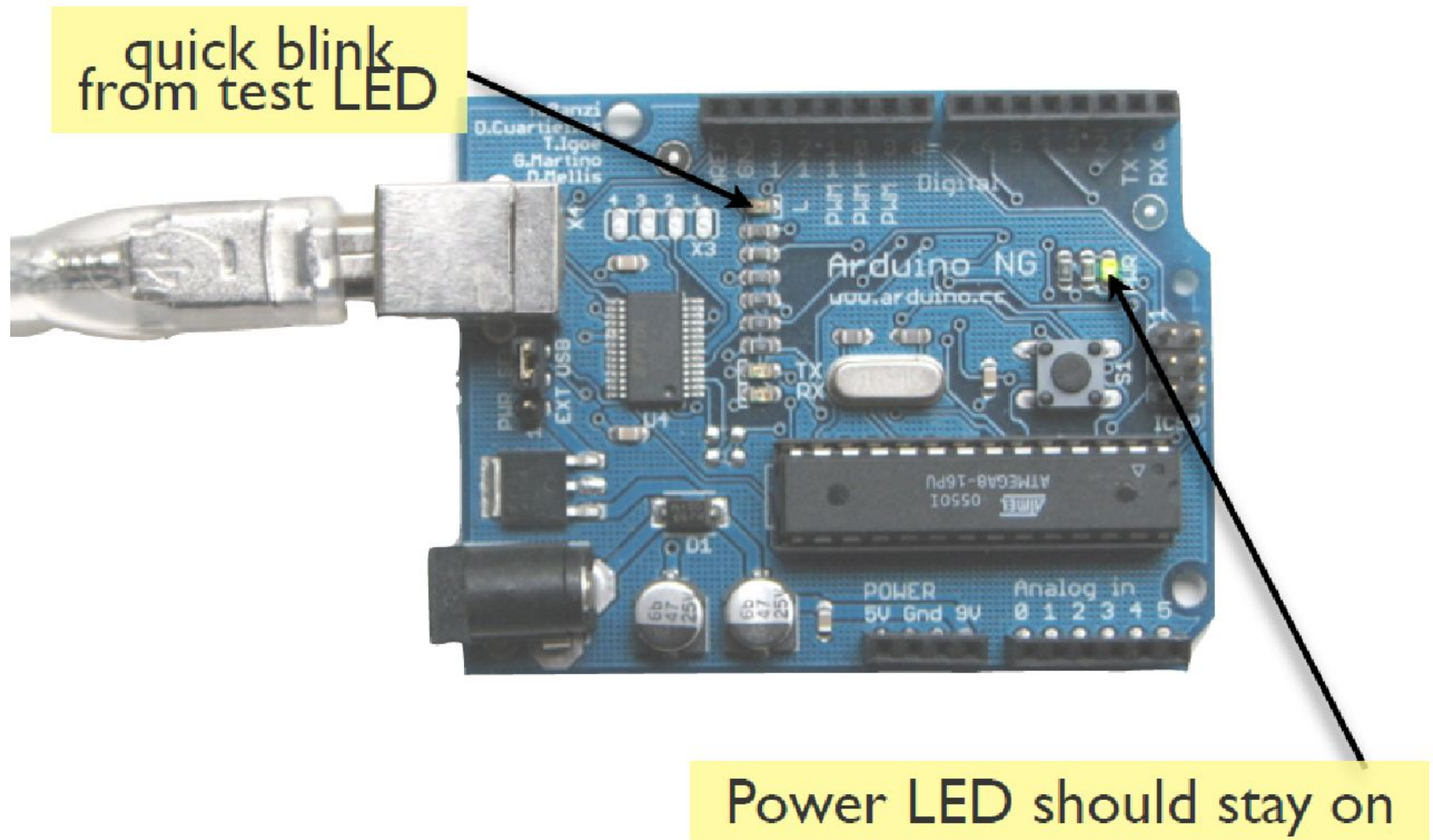
The Arduino IDE

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

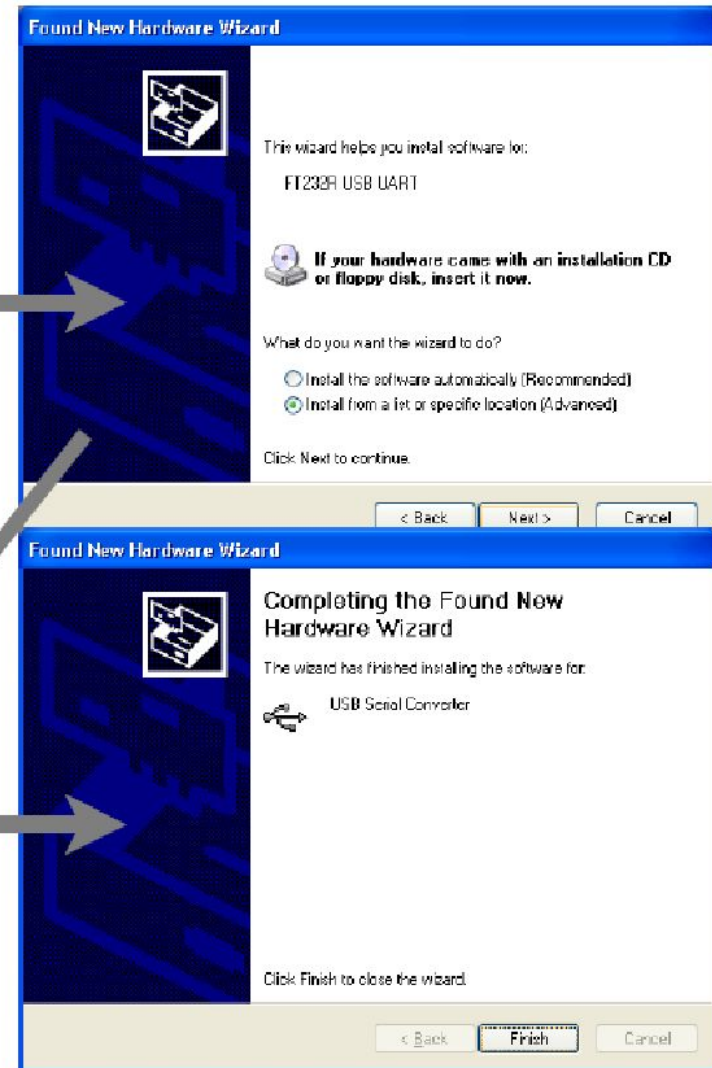
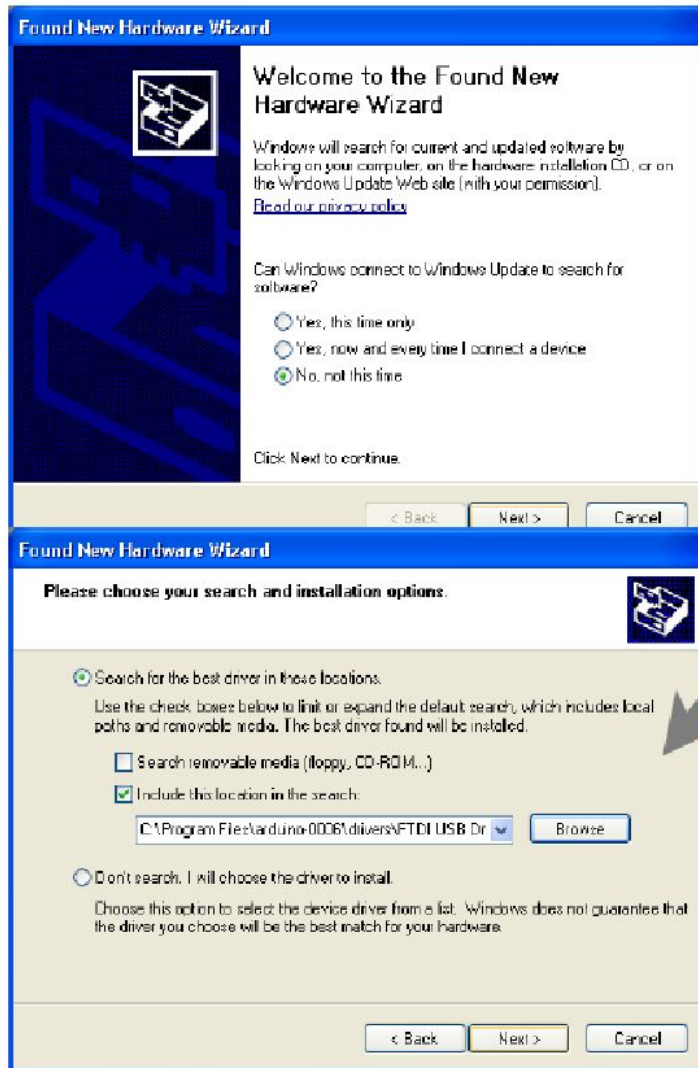
# Installing Arduino

1. Get the Arduino software & unzip it
2. Plug in Arduino board
3. Install the driver
4. Reboot
5. Run the Arduino program
6. Tell Arduino (program) about Arduino (board)

# Plug in Arduino board

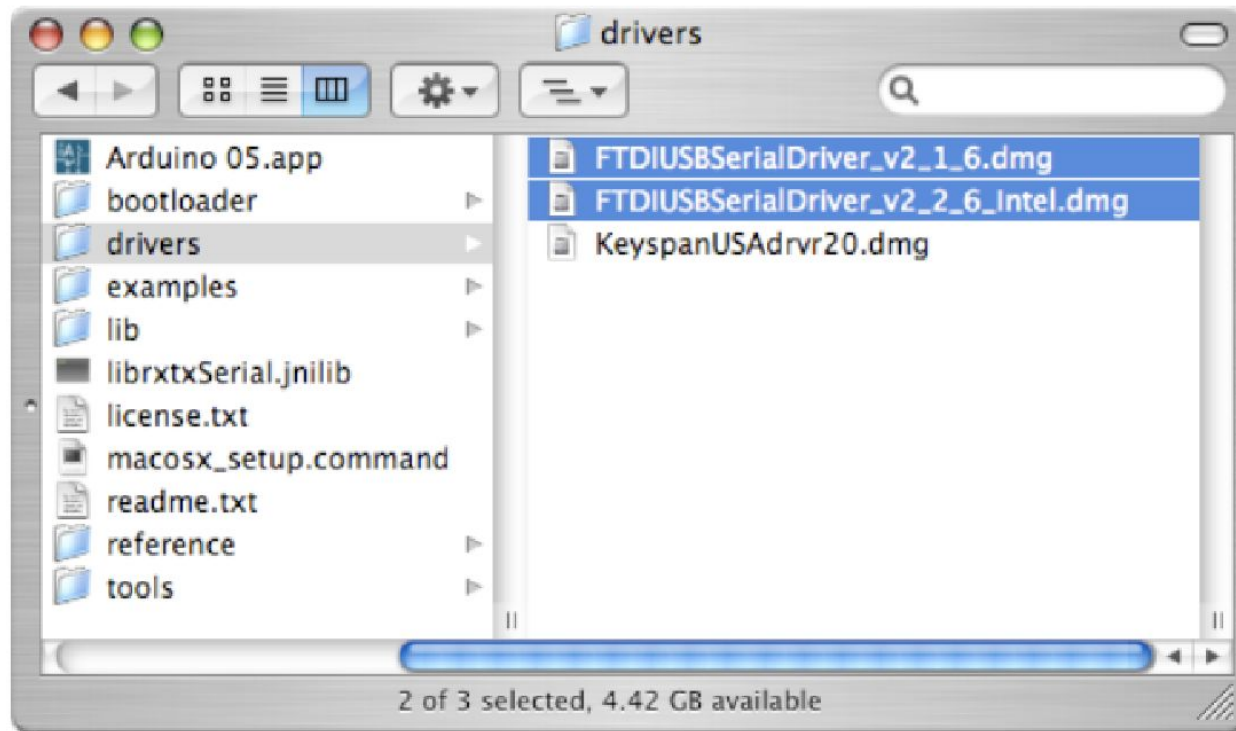


# Windows Driver Install



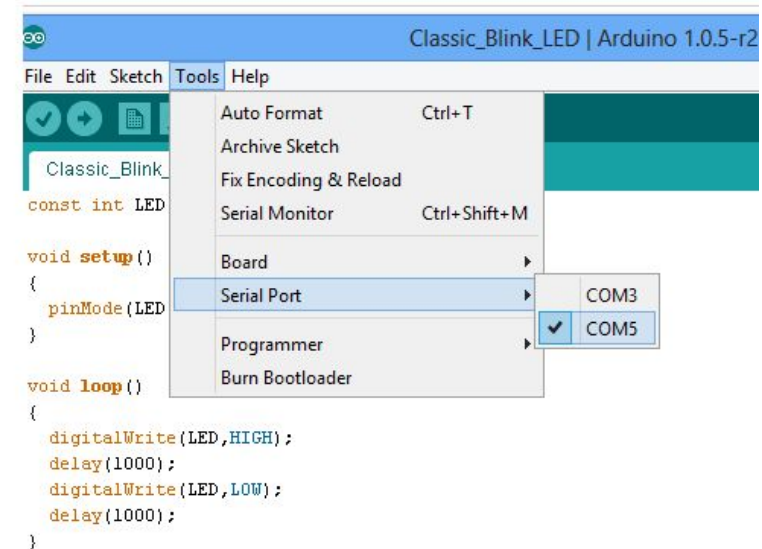
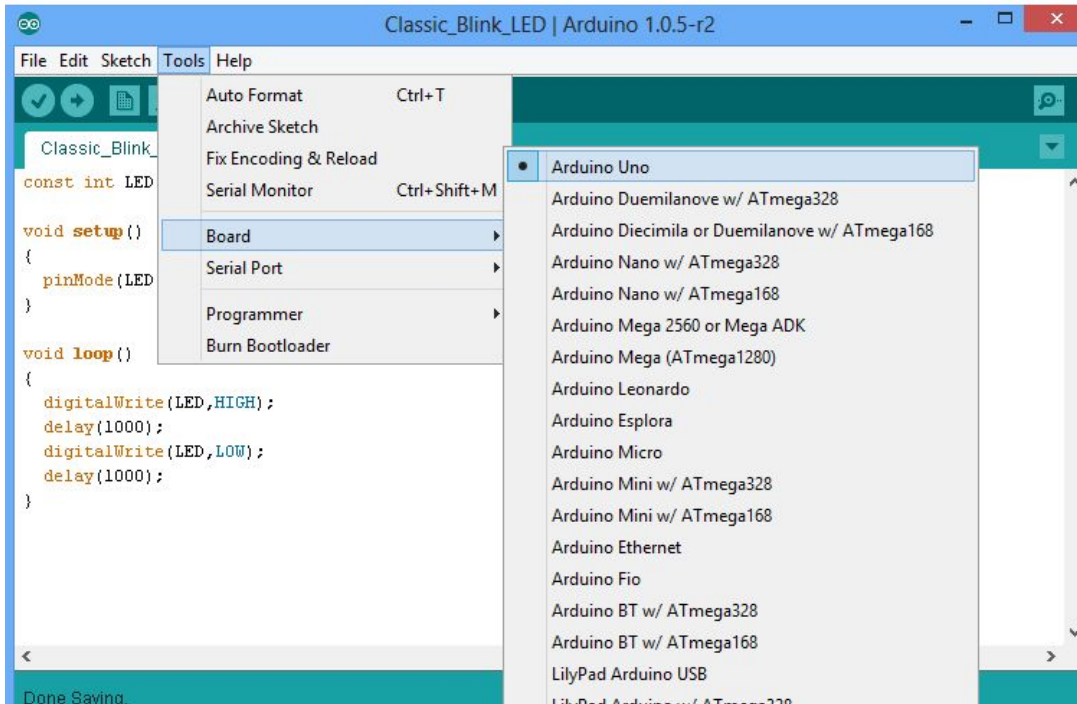
# Mac Driver Install

Double-click on .dmg Installer

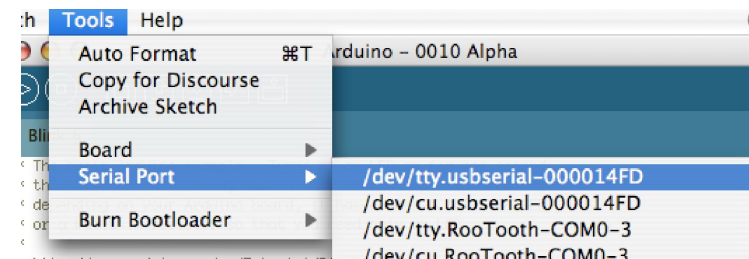
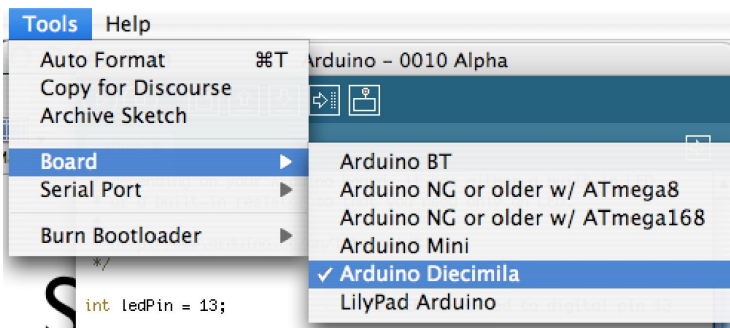


- v2\_1\_6 for PPC Macs
- v2\_2\_6 for Intel Macs

# Selecting Location & Type

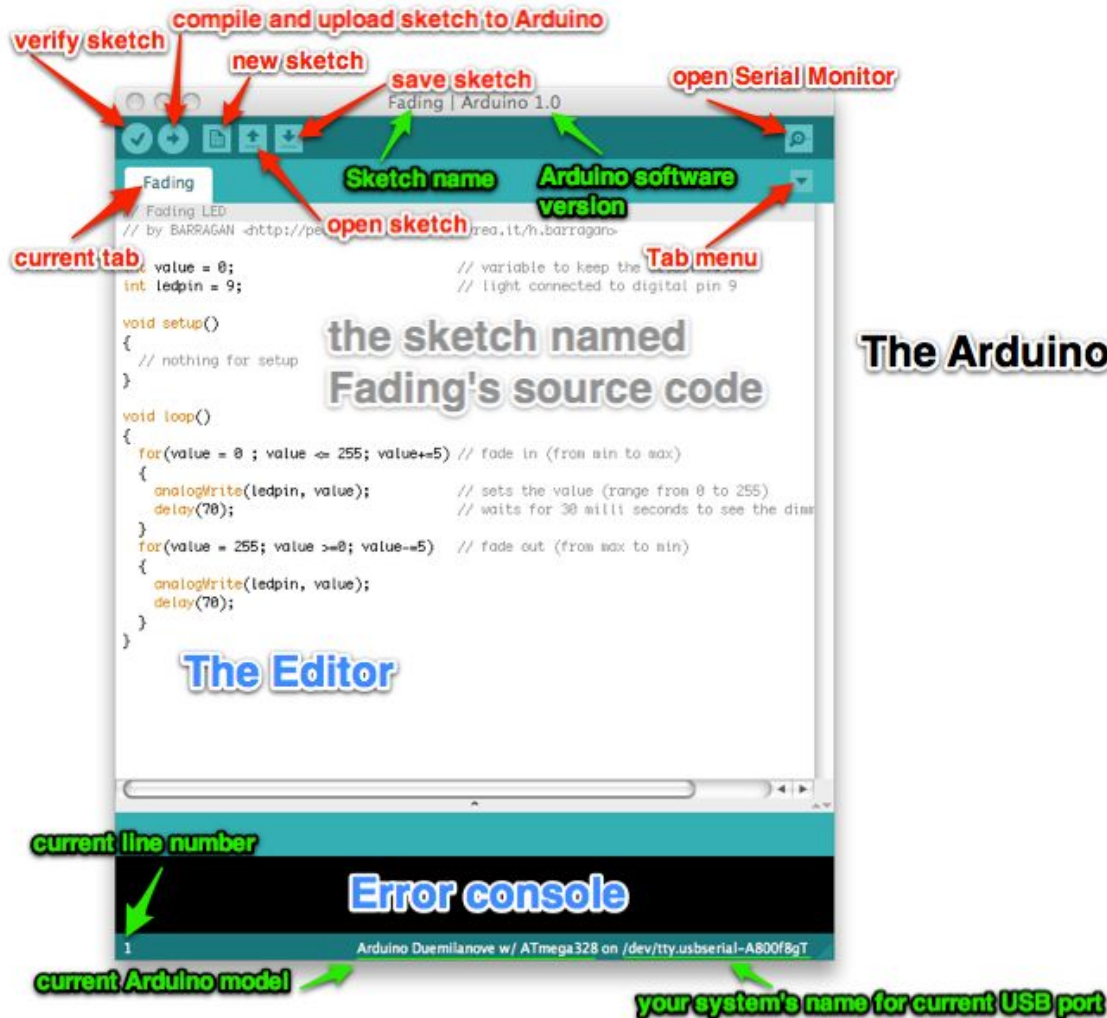


usually highest numbered port



starts with *tty.usbserial*

# Arduino IDE

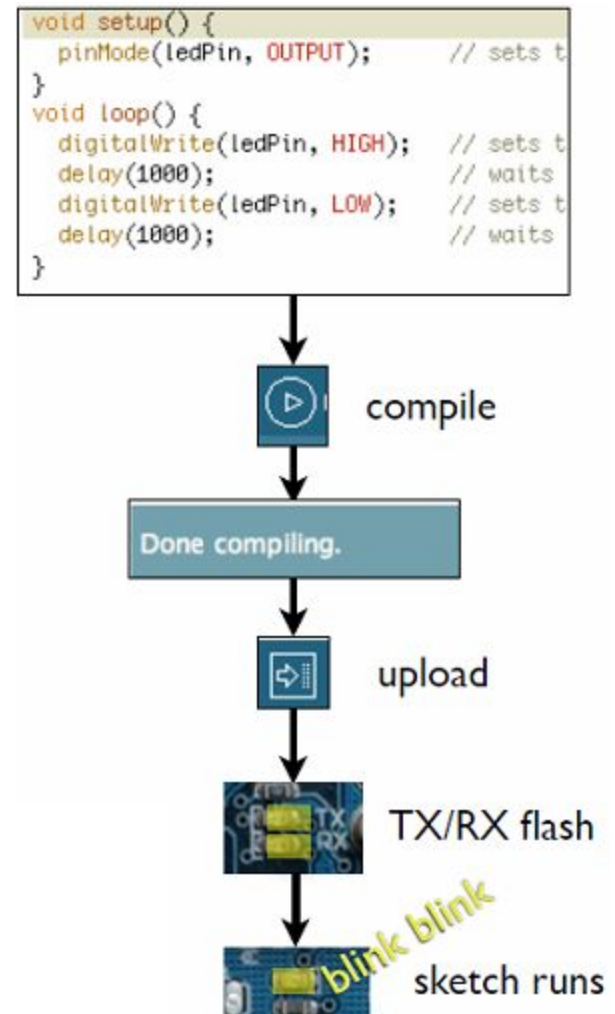


The Arduino IDE



# Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch
- Try it out with the “Blink” sketch!
- Load “File/Examples/Basics/Blink”



# Status Messages

## Uploading worked

```
Done uploading.  
Binary sketch size: 1110 bytes (of a 14336 byte maximum)
```

Size depends on  
complexity of your sketch

## Wrong serial port selected

```
Serial port '/dev/tty.usbserial-A4001qa8' not found. Did you select the  
java.awt.EventQueue$PumpEvents(EventDispatchThread.java:170  
)  
at  
java.awt.EventQueue.run(EventDispatchThread.java:110)
```

## Wrong board selected

```
Wrong microcontroller found. Did you select the right board from the T  
Binary sketch size: 000 bytes (of a 7100 byte maximum)  
avrdude: Expected signature for ATMEGA8 is 1E 93 07  
Double check chip, or use -F to override this check.
```

nerdy cryptic error messages

# Troubleshooting

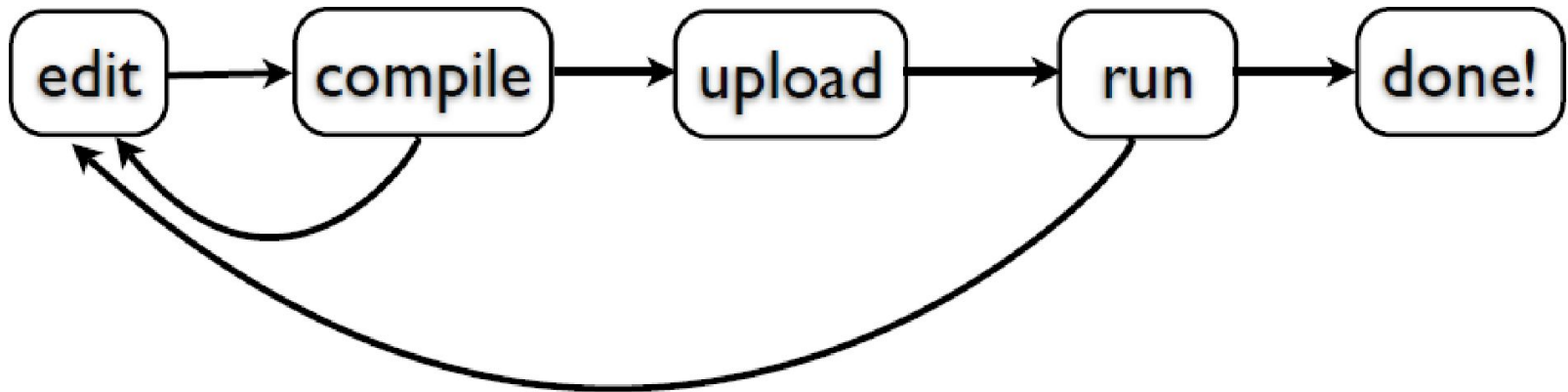
- Most common problem is incorrect serial port setting
- If you ever have any “weird” errors from the Arduino environment, just try again.
- The red text at the bottom is debugging output in case there may be a problem
- Status area shows summary of what’s wrong

# I made an LED blink, so what?

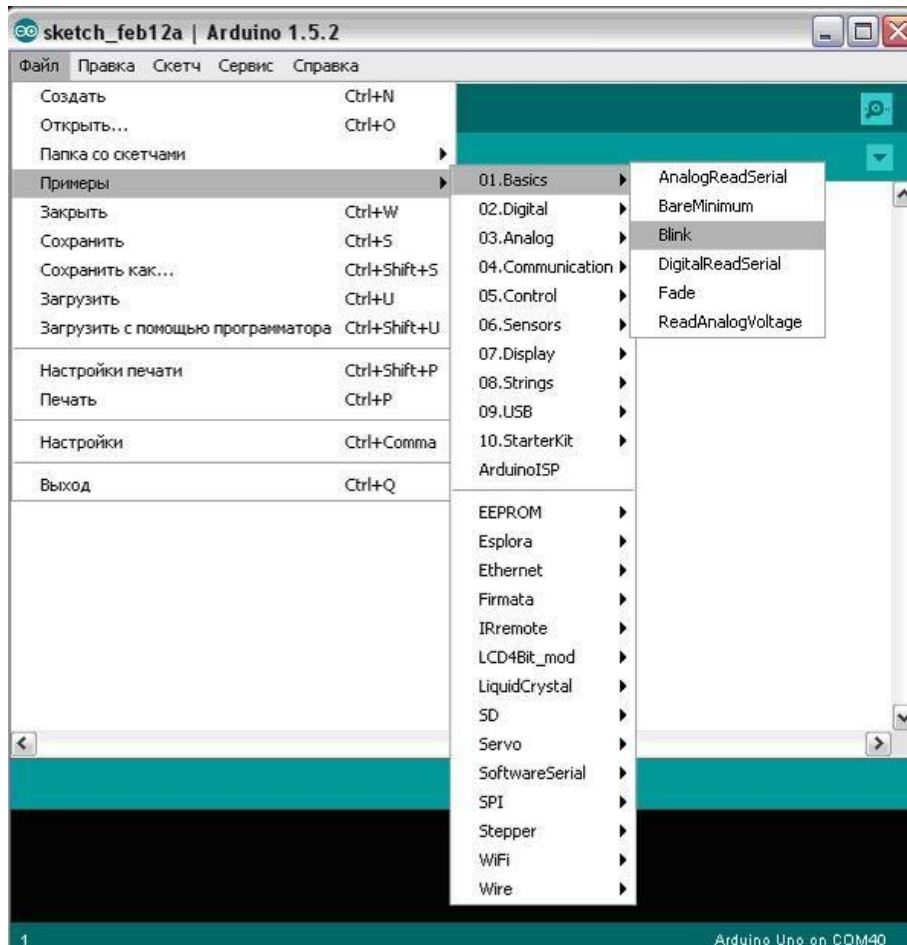
- Most actuators are switched on and off with a digital output
- The `digitalWrite()` command is the software portion of being able to control just about anything
- LEDs are easy, motors come in a bit
- Arduino has up to 13 digital outputs, and you easily can add more with helper chips

# Development Cycle

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



# Lots of Built-in Examples



And more here:  
<http://www.arduino.cc/en/Tutorial/HomePage>

And all over the Net. Search for “Arduino tutorial” or “Arduino notes” or whatever you’re interested in and “Arduino” and likely you’ll find some neat pages.

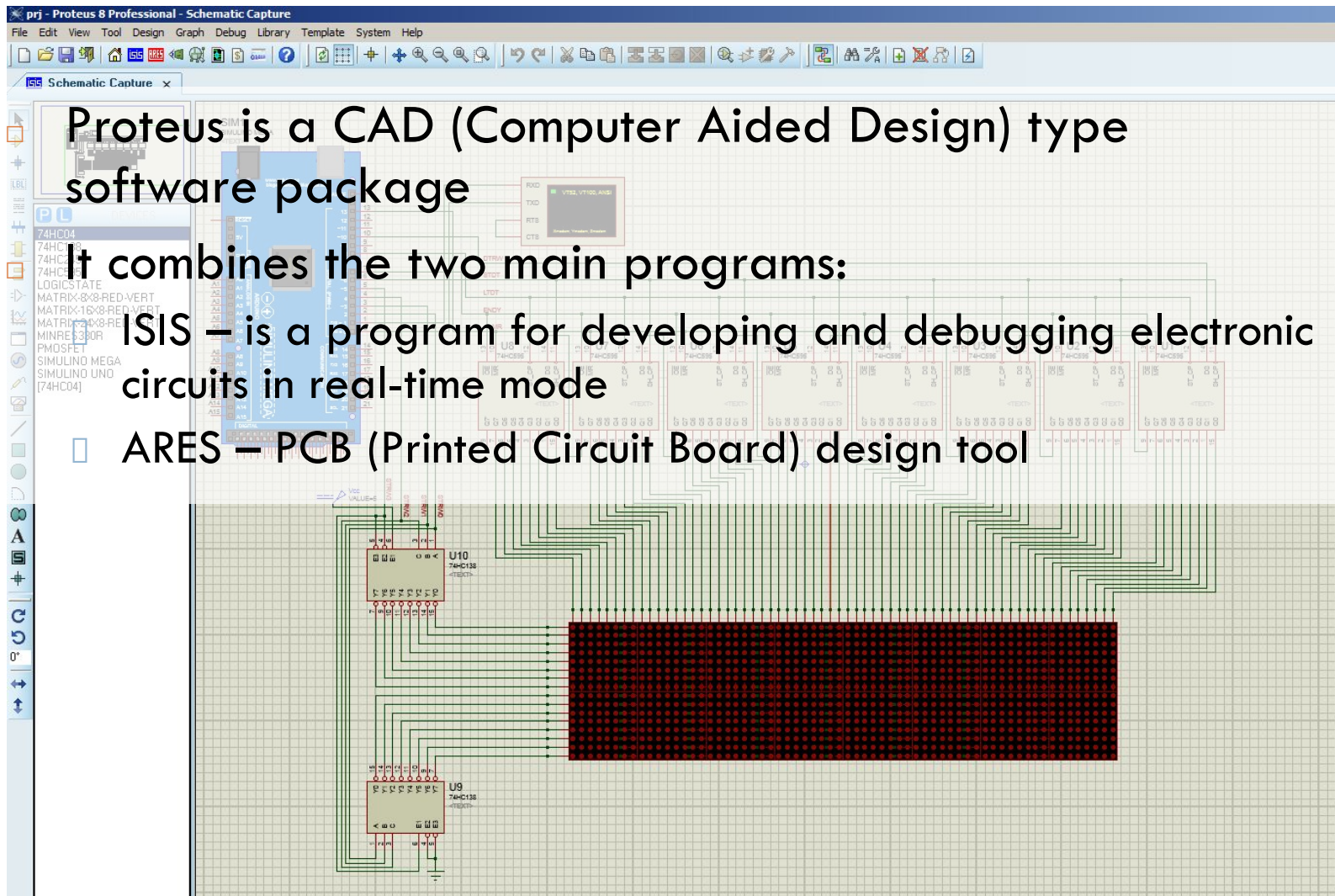
# Proteus ISIS Simulation System

Proteus is a CAD (Computer Aided Design) type software package

It combines the two main programs:

ISIS – is a program for developing and debugging electronic circuits in real-time mode

ARES – PCB (Printed Circuit Board) design tool



# Proteus Menu and Navigation

The screenshot shows the Proteus software interface with several red arrows pointing to specific features, each labeled with a Russian text annotation:

- Верхнее меню команд и инструментов** (Top menu of commands and instruments) points to the menu bar and toolbar.
- Предпросмотр** (Preview) points to the small schematic view in the top-left corner.
- Режимы селектора** (Selector modes) points to the selection tool icons on the left toolbar.
- Библиотека** (Library) points to the DEVICES list on the left.
- Селектор объектов** (Object selector) points to the DEVICES list.
- Ориентация элементов** (Element orientation) points to the rotation tool icon on the left toolbar.
- Управление симуляцией** (Simulation control) points to the simulation control buttons at the bottom.
- Лог событий** (Event log) points to the log window at the bottom right.
- Координаты курсора** (Cursor coordinates) points to the coordinate display at the bottom right.

The main workspace, labeled **Рабочее окно** (Working window), displays a circuit diagram with an ATMEGA1280 microcontroller, two 7-segment displays, and a switch. The component list on the right includes:

U1	RESIST	PF60C04	20
J1	XTAL1	PF60C02	20
J2	XTAL2	PF60C02	20
P1	PF60C04	PF60C04	20
P2	PF60C04	PF60C04	20
P3	PF60C04	PF60C04	20
P4	PF60C04	PF60C04	20
P5	PF60C04	PF60C04	20
P6	PF60C04	PF60C04	20
P7	PF60C04	PF60C04	20
P8	PF60C04	PF60C04	20
P9	PF60C04	PF60C04	20
P10	PF60C04	PF60C04	20
P11	PF60C04	PF60C04	20
P12	PF60C04	PF60C04	20
P13	PF60C04	PF60C04	20
P14	PF60C04	PF60C04	20
P15	PF60C04	PF60C04	20
P16	PF60C04	PF60C04	20
P17	PF60C04	PF60C04	20
P18	PF60C04	PF60C04	20
P19	PF60C04	PF60C04	20
P20	PF60C04	PF60C04	20
P21	PF60C04	PF60C04	20
P22	PF60C04	PF60C04	20
P23	PF60C04	PF60C04	20
P24	PF60C04	PF60C04	20
P25	PF60C04	PF60C04	20
P26	PF60C04	PF60C04	20
P27	PF60C04	PF60C04	20
P28	PF60C04	PF60C04	20
P29	PF60C04	PF60C04	20
P30	PF60C04	PF60C04	20
P31	PF60C04	PF60C04	20
P32	PF60C04	PF60C04	20
P33	PF60C04	PF60C04	20
P34	PF60C04	PF60C04	20
P35	PF60C04	PF60C04	20
P36	PF60C04	PF60C04	20
P37	PF60C04	PF60C04	20
P38	PF60C04	PF60C04	20
P39	PF60C04	PF60C04	20
P40	PF60C04	PF60C04	20
P41	PF60C04	PF60C04	20
P42	PF60C04	PF60C04	20
P43	PF60C04	PF60C04	20
P44	PF60C04	PF60C04	20
P45	PF60C04	PF60C04	20
P46	PF60C04	PF60C04	20
P47	PF60C04	PF60C04	20
P48	PF60C04	PF60C04	20
P49	PF60C04	PF60C04	20
P50	PF60C04	PF60C04	20
P51	PF60C04	PF60C04	20
P52	PF60C04	PF60C04	20
P53	PF60C04	PF60C04	20
P54	PF60C04	PF60C04	20
P55	PF60C04	PF60C04	20
P56	PF60C04	PF60C04	20
P57	PF60C04	PF60C04	20
P58	PF60C04	PF60C04	20
P59	PF60C04	PF60C04	20
P60	PF60C04	PF60C04	20
P61	PF60C04	PF60C04	20
P62	PF60C04	PF60C04	20
P63	PF60C04	PF60C04	20
P64	PF60C04	PF60C04	20
P65	PF60C04	PF60C04	20
P66	PF60C04	PF60C04	20
P67	PF60C04	PF60C04	20
P68	PF60C04	PF60C04	20
P69	PF60C04	PF60C04	20
P70	PF60C04	PF60C04	20
P71	PF60C04	PF60C04	20
P72	PF60C04	PF60C04	20
P73	PF60C04	PF60C04	20
P74	PF60C04	PF60C04	20
P75	PF60C04	PF60C04	20
P76	PF60C04	PF60C04	20
P77	PF60C04	PF60C04	20
P78	PF60C04	PF60C04	20
P79	PF60C04	PF60C04	20
P80	PF60C04	PF60C04	20
P81	PF60C04	PF60C04	20
P82	PF60C04	PF60C04	20
P83	PF60C04	PF60C04	20
P84	PF60C04	PF60C04	20
P85	PF60C04	PF60C04	20
P86	PF60C04	PF60C04	20
P87	PF60C04	PF60C04	20
P88	PF60C04	PF60C04	20
P89	PF60C04	PF60C04	20
P90	PF60C04	PF60C04	20
P91	PF60C04	PF60C04	20
P92	PF60C04	PF60C04	20
P93	PF60C04	PF60C04	20
P94	PF60C04	PF60C04	20
P95	PF60C04	PF60C04	20
P96	PF60C04	PF60C04	20
P97	PF60C04	PF60C04	20
P98	PF60C04	PF60C04	20
P99	PF60C04	PF60C04	20
P100	PF60C04	PF60C04	20

The status bar at the bottom shows: ANIMATING: 00:04:08.900000 (CPU load 14%) Таймер симуляции и окно загрузки ЦП +3600.0 -400.0 th



# Creating a Circuit on Proteus

The screenshot displays the Proteus ISIS Professional interface. The main workspace shows a circuit diagram with a +5V power source, a switch labeled SW1 (SW-SPDT-MOM), and a lamp labeled L1 (12V). The left sidebar contains a 'DEVICES' list with 'LAMP', 'OMI-SH-T12L', and 'SW-SPDT-MOM'. The bottom status bar shows 'Root sheet 1' and a zoom level of 100%.

**1** Создаём новый документ, если он ещё не создан, если создан переходим к пункту 2

**2** Заходим в библиотеку(нажимаем на кнопку "P")

**3** В библиотеке проекта выделяем добавленный элемент, и два раза кликаем по пустому пространству, в котором нам нужно этот элемент разместить.

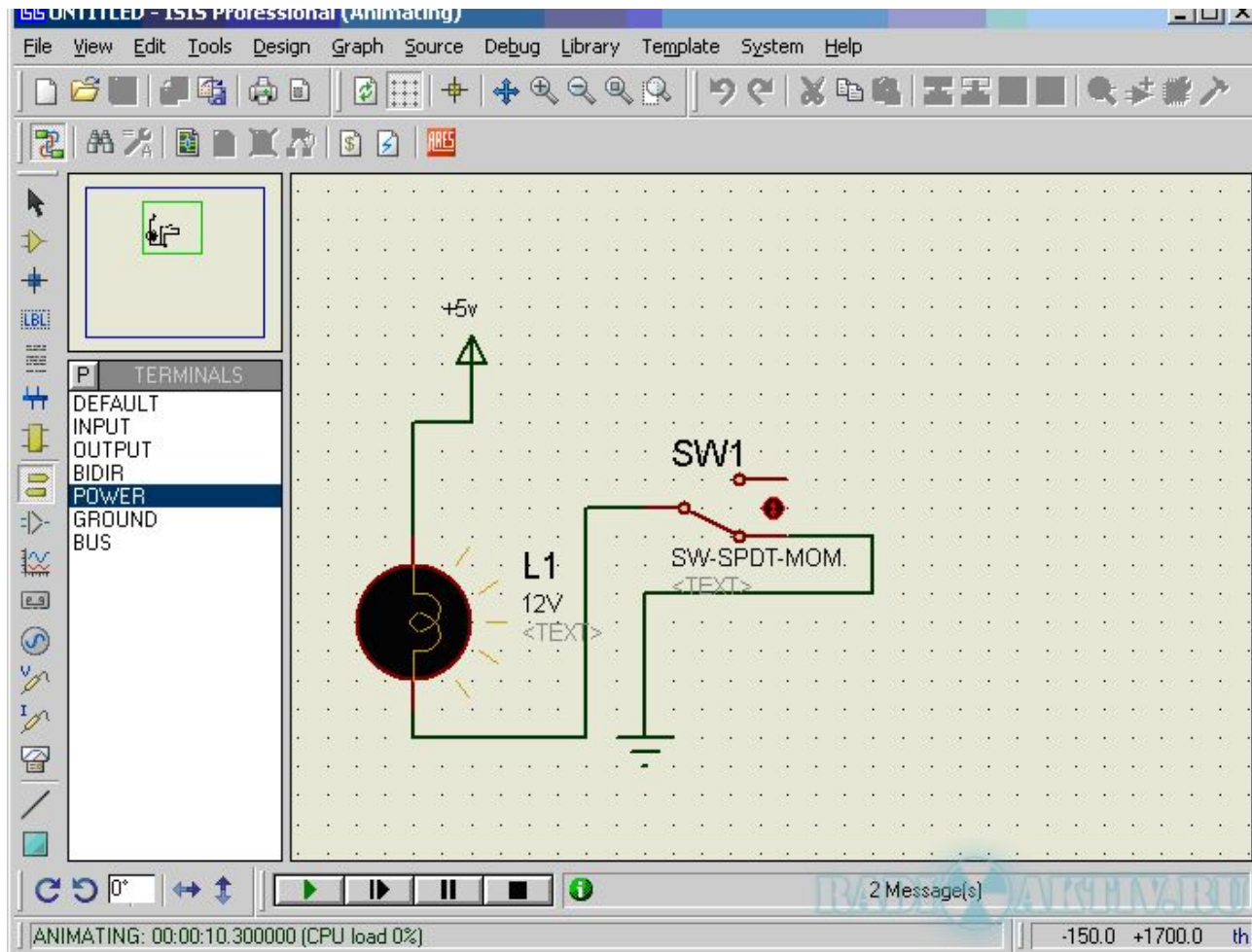
**4** Для подключения питания к нашей схеме нажмём на кнопку Terminals (терминалы)

# Connecting Elements

The screenshot displays the ISIS Professional software interface. The main workspace shows a circuit diagram on a grid. A red wire is being connected from a component labeled 'L1 12V <Text>' to another component labeled 'SW1 SW-SPDT-MOM <Text>'. A blue cursor is positioned at the end of the red wire. A red arrow points from the cursor to the SW1 component. A red text box with a white background contains the following text: **Подводим курсор к концу проводника, нажимаем один раз и ведём проводник к элементу с которым нам необходимо его соединить.**

The interface includes a menu bar (File, View, Edit, Tools, Design, Graph, Source, Debug, Library, Template, System, Help), a toolbar with various icons, and a component list on the left side. The component list is titled 'TERMINALS' and includes: DEFAULT, INPUT, OUTPUT, BIDIR, POWER (highlighted), GROUND, and BUS. The status bar at the bottom shows 'COMPONENT L1, Name=1, Number=<NONE>, Type=PS' and coordinates '-1400.0 +1600.0 th'.

# Simulation



# Installing Arduino Library for Proteus

- For Windows XP

**Copy file BLOGEMBARCADO.LIB into:**

C:\Program Files\Labcenter Electronics\Proteus 8 Professional\Data\LIBRARY

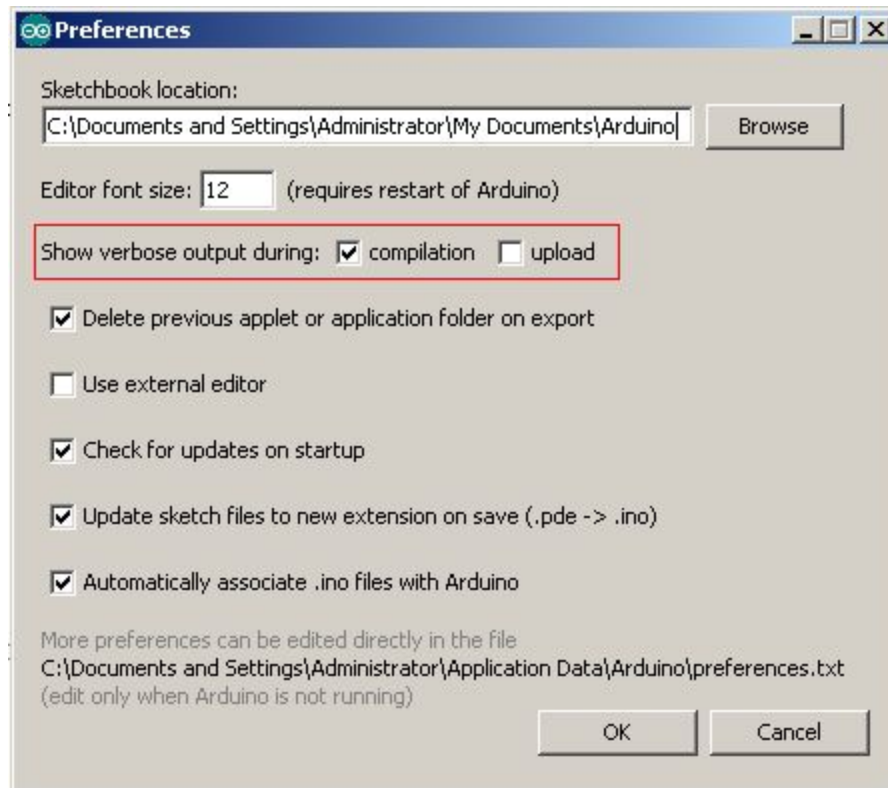
- For Windows 7 and later

**Copy file BLOGEMBARCADO.LIB into:**

C:\ProgramData\Labcenter Electronics\Proteus 8 Professional\LIBRARY

# Loading the compiled file to Proteus

- File → Preferences → Show verbose output during **compilation**



# Loading the compiled file to Proteus

Select and copy the location of .hex file

```
C:\Users\zain\Desktop\arduino-1.6.5-r5\hardware\tools\avr\bin\avr-ar rcs C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp\WString.cpp.o
C:\Users\zain\Desktop\arduino-1.6.5-r5\hardware\tools\avr\bin\avr-gcc -w -Os -Wl,--gc-sections C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp/Blink.cpp.elf C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp\core.a -LC:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp -o C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp/Blink.cpp.eep
C:\Users\zain\Desktop\arduino-1.6.5-r5\hardware\tools\avr\bin\avr-objcopy -O ihex -j .eeprom --change-section-lma .eeprom=0 C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp/Blink.cpp.eep C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp/Blink.cpp.hex
C:\Users\zain\Desktop\arduino-1.6.5-r5\hardware\tools\avr\bin\avr-objcopy -O ihex -R .eeprom C:\Users\zain\AppData\Local\Temp\build7243111610766241365.tmp/Blink.cpp.hex

Sketch uses 1,030 bytes (3%) of program storage space. Maximum is 32,256 bytes.

Global variables use 9 bytes (0%) of dynamic memory, leaving 2,039 bytes for local variables.

4 Arduino/Genuino Uno on COM5
```

# Loading the compiled file to Proteus

Double click

ISIS Edit Component

Part Reference: SIM1

Part Value: SIMULINO MEGA

Element: [New]

PROGRAM FILE: ..\..\AppData\Local\Temp\bui [Browse]

Clock Frequency: 16MHz

NAME: SIMULINO MEGA

URL: blogembarcado.blogspot.com

VERSION: 4.1

Advanced Properties:

Disassemble Binary Code: No

Other Properties:

Exclude from Simulation

Exclude from PCB Layout

Exclude from Bill of Materials

Attach hierarchy module

Hide common pins

Edit all properties as text

Paste the location of .hex file here

# Useful Links

- <http://arduino.cc/>

Official homepage. Also check out the Playground & forums

- <http://arduino.ru/>

Lots of useful information about Arduino and programming language on Russian language

- <http://arduino-project.net/videouroki-arduino-arduino4life/>

Arduino video tutorials

- <http://adafruit.com/>

Arduino starter kits, Boarduino Arduino clone, lots of cool kits

- <http://sparkfun.com/>

Sells Arduino boards and lots of neat sensors & stuff

- Books:

- “Arduino cookbook”, Michael Margolis
- “Arduino programming notebook”, Brian W. Evans
- “Getting started with Arduino”, Massimo Banzi



# Some Common Commands

- **Serial.println(value);**  
Prints the value to the Serial Monitor on your computer
- **pinMode(pin, mode);**  
Configures a digital pin to read (input) or write (output) a digital value
- **digitalRead(pin);**  
Reads a digital value (HIGH or LOW) on a pin set for input
- **digitalWrite(pin, value);**  
Writes the digital value (HIGH or LOW) to a pin set for output
- **delay(value)**  
Stops the program execution for amount of milliseconds given by value

# Hidden Treasure

```
int main(void)
{
    init(); // initializes the Arduino hardware
    setup();
    for (;;)
        loop();
    return 0;
}
```

# Tasks

- Blinking LED on 12<sup>th</sup> pin
- 3 LEDs blink by order (interval - 1s)
- Traffic lights (Rd-5s, Yl-1s, Gr-5s, Yl-1s ...)
- 3 LEDs binary counter (0-7)
- 4 LED ripple

# 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.

# Flow control

## 1. if

```
if(expression){ //if expression is true
  doSomething;
}
```

## 2. if... else

```
if(inputPin == HIGH){
  doThingA;
} else{
  doThingB;
}
```

## 3. for

```
for (initialization; condition; expression){
  doSomething;
}
```

## 4. while

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

## 5. do... while

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

```
if (inputPin < 500){
  doThingA;
}
else if (inputPin >= 1000){
  doThingB;
}
else{
  doThingC;
}
```

```
for(j=0; j < 4; j++ ){
  Serial.println(j);
}
```

```
while (someVariable < 200){ //if less than 200
  doSomething; // executes enclosed statements
  someVariable++; // increments variable by 1
}
do{ // assign readSensors value to x
  x = readSensors();
  delay (50); // pauses 50 milliseconds
} while (x < 100); // loops if x is less than 100
```

# Using Floating-Point Numbers

```
float value = 1.1;
void setup(){
  Serial.begin(9600);
}
void loop(){
  value = value - 0.1;
  //reduce value by 0.1 each time through the loop
  if( value == 0)
    Serial.println("The value is exactly zero");
  else if(fabs(value) < .0001)
    //function to take the absolute value of a float
    Serial.println("The value is close enough to zero");
  else
    Serial.println(value);
  delay(100);
}
```

## OUTPUT:

```
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
The value is close enough
to zero
-0.10
-0.20
```

This is because the only memory-efficient way that floating-point numbers can contain the huge range in values they can represent is by storing an approximation of the number. The solution to this is to check if a variable is close to the desired value.

# Arrays

- Arrays are zero indexed, with the first value in the array beginning at index number 0. An array needs to be declared and optionally assigned values before they can be used.

```
int myArray[] = {value0, value1, value2...}
```

- Likewise it is possible to declare an array by declaring the array type and size and later assign values to an index position

```
int myArray[5]; // declares integer array with 5 positions  
myArray[3] = 10; // assigns the 3rd index the value 10
```

- To retrieve a value from an array, assign a variable to the array and index position:

```
first = myArray[0]; // this is the first element  
last = myArray[4]; // this is the last element
```

# Tasks with arrays and loops

- 3 LEDs blink by order (interval - 1 s)
- Traffic lights (Rd-5s, Yl-1s, Gr-5s, Yl-1s ...)
- 3 LEDs binary counter (0-7)
- 4 LED ripple