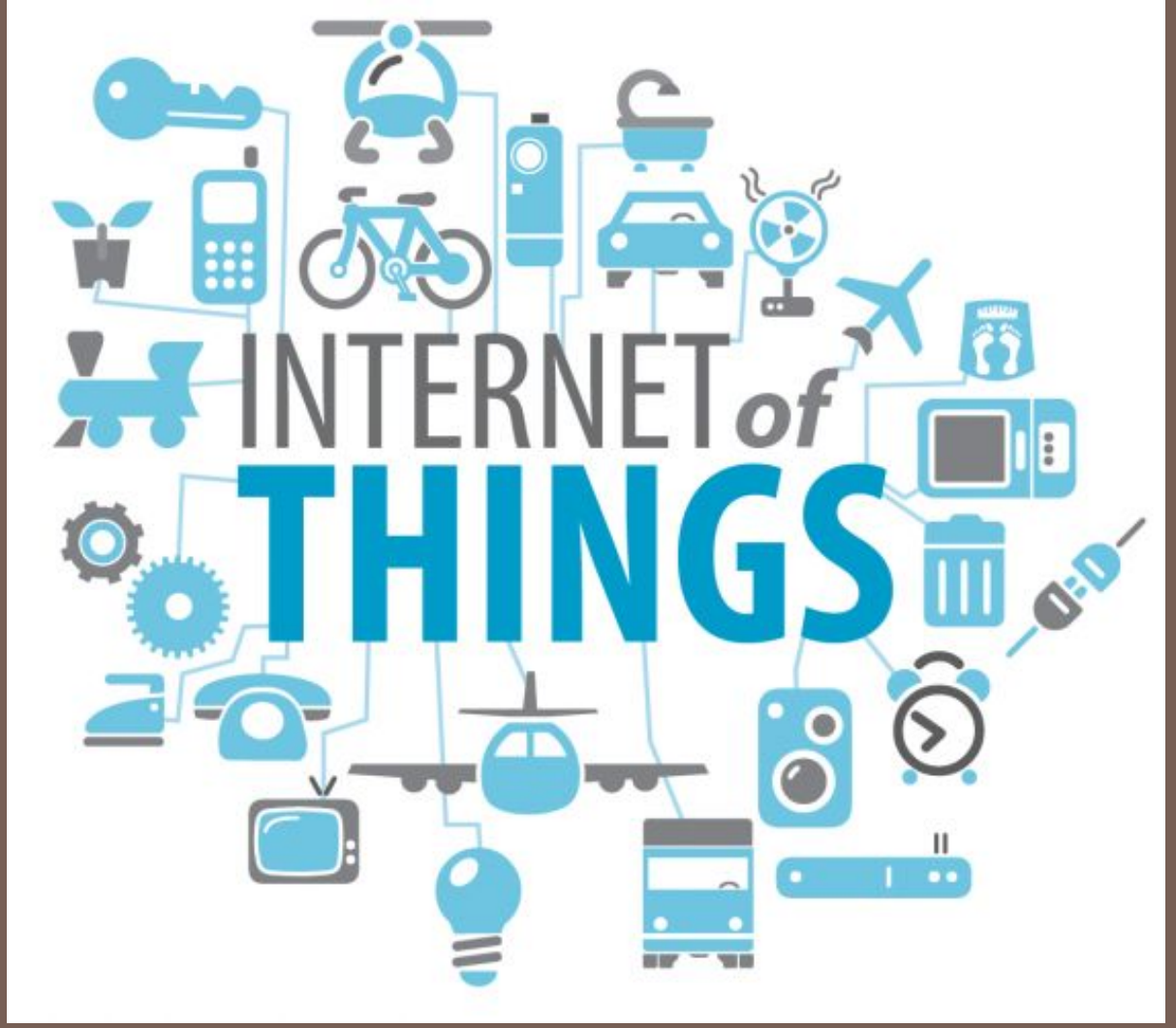


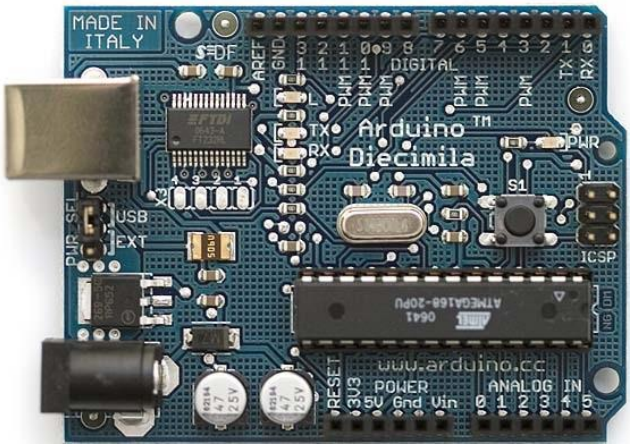
IMPLEMENTING IOE

Week 2

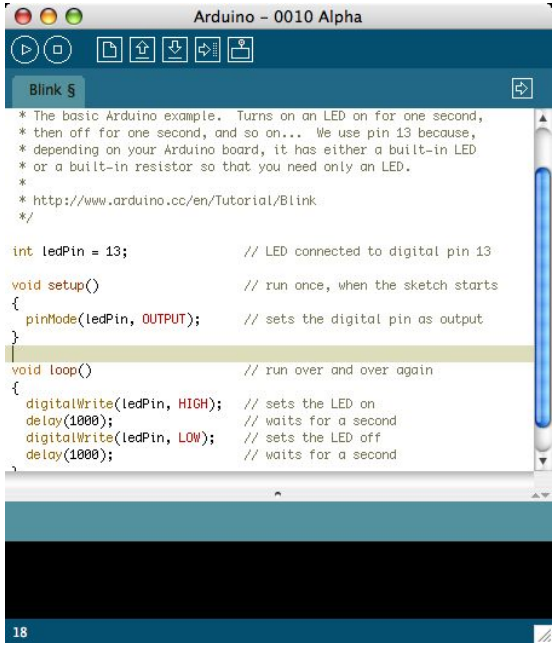
Assist. Prof. Rassim Suliyev - SDU 2017

What is Arduino?

Physical Device



IDE



Community

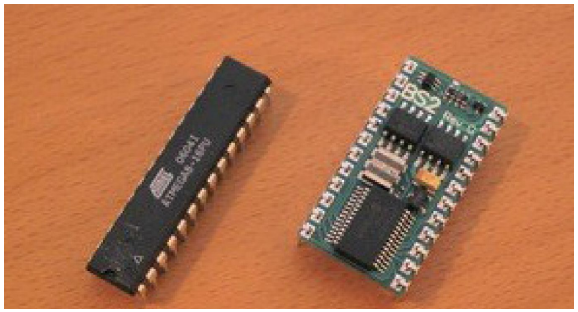
http://www.arduino.cc

Arduino Philosophy and Community

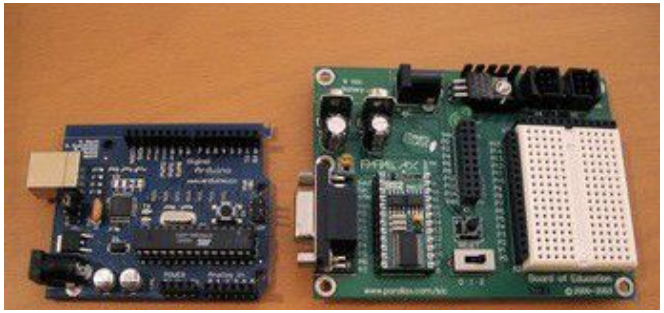
- Open Source Physical Computing Platform
  - "open source hardware"
  - open source: free to inspect & modify
  - physical computing
    - ubiquitous computing
    - pervasive computing
    - ambient intelligence
    - calm computing
    - Spimes
    - Blogjects
    - smart objects
- Community-built
  - 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.



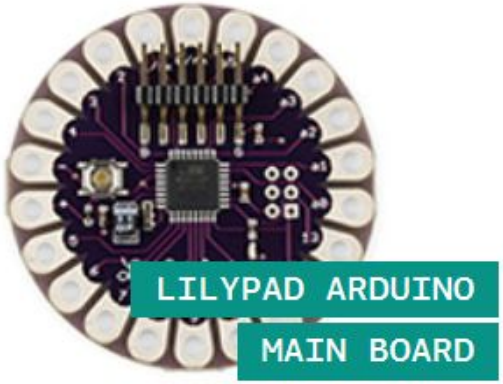
$2 $50



$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 Capabilities

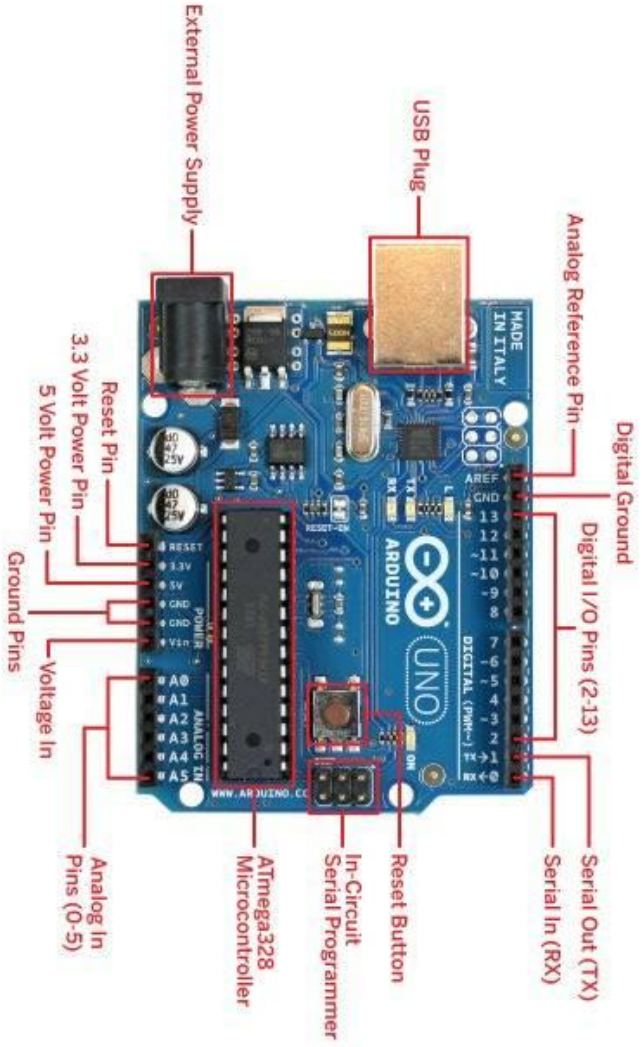
- 16 kBytes of Flash program memory
- 1 kByte of RAM
- 16 MHz (Apple II: 1 MHz)
- Inputs and Outputs
  - 14 digital input/output pins
  - 6 analog input pins
  - 6 analog output pins (pseudo-analog, uses PWM , which we'll talk about later)
- Completely stand-alone: doesn't need a computer once programmed

* 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

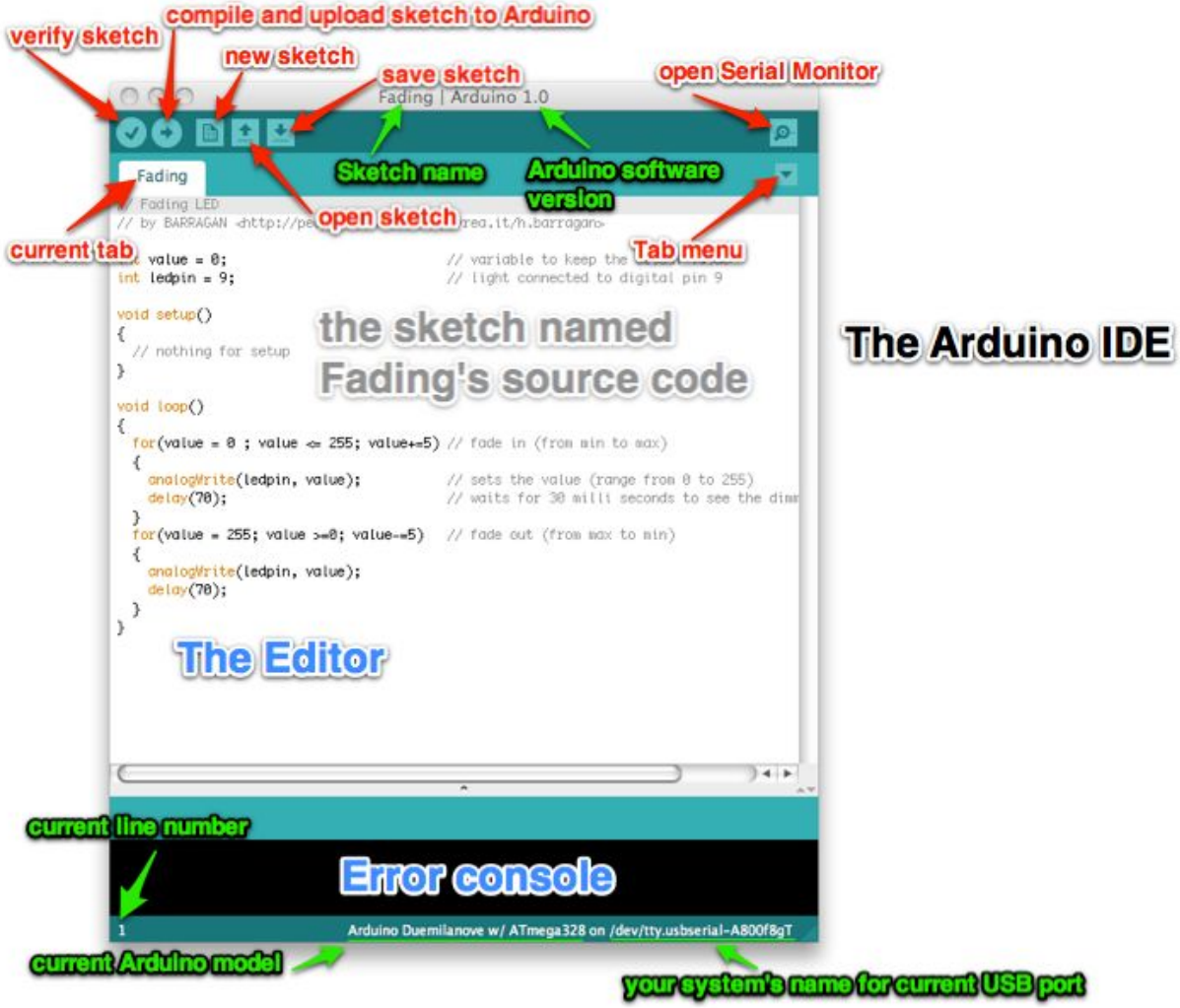


Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recom)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (6 PWM)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (0.5 KB bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Length	68.6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

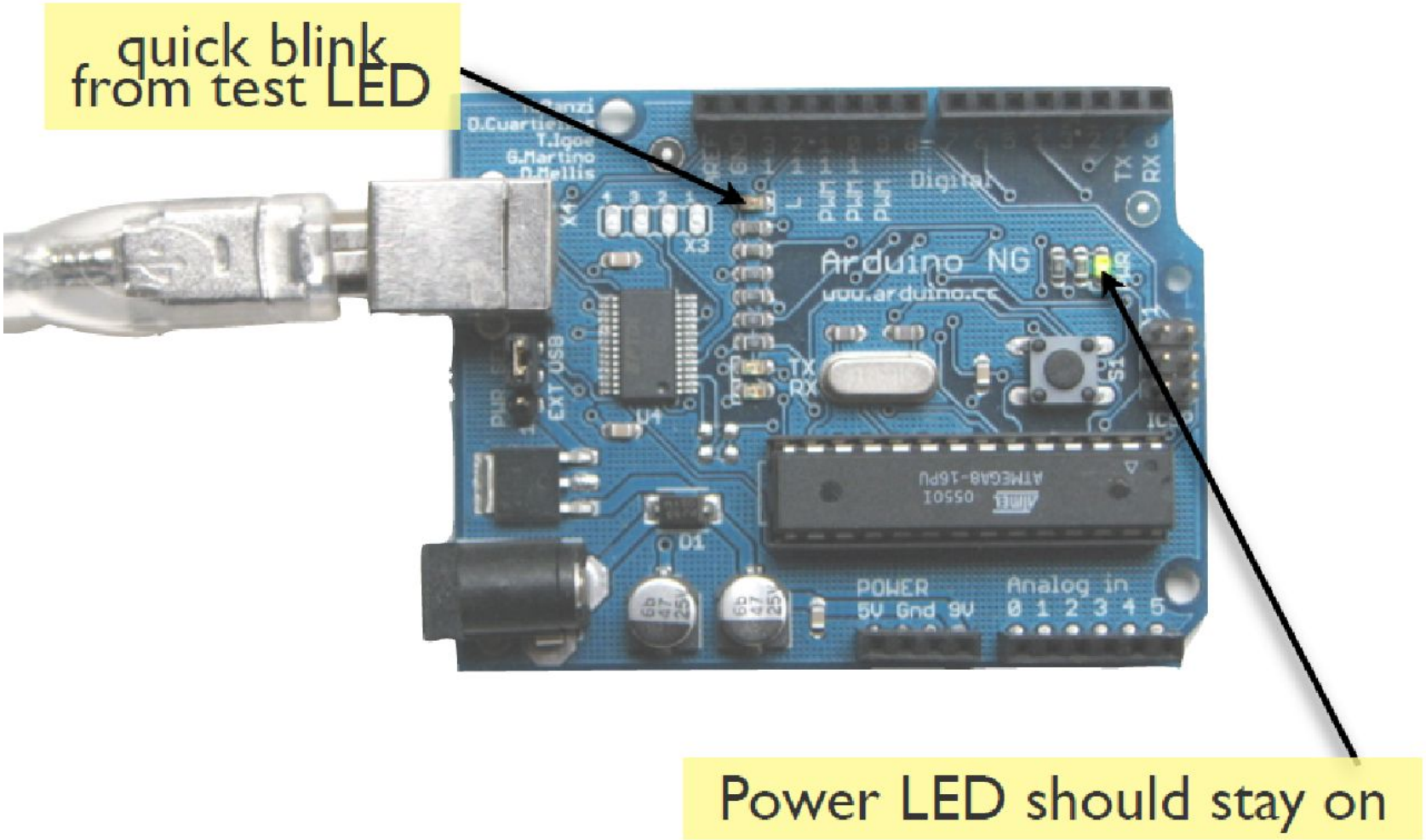


- Like a text editor
- View/write/edit sketches
- But then you program them into hardware

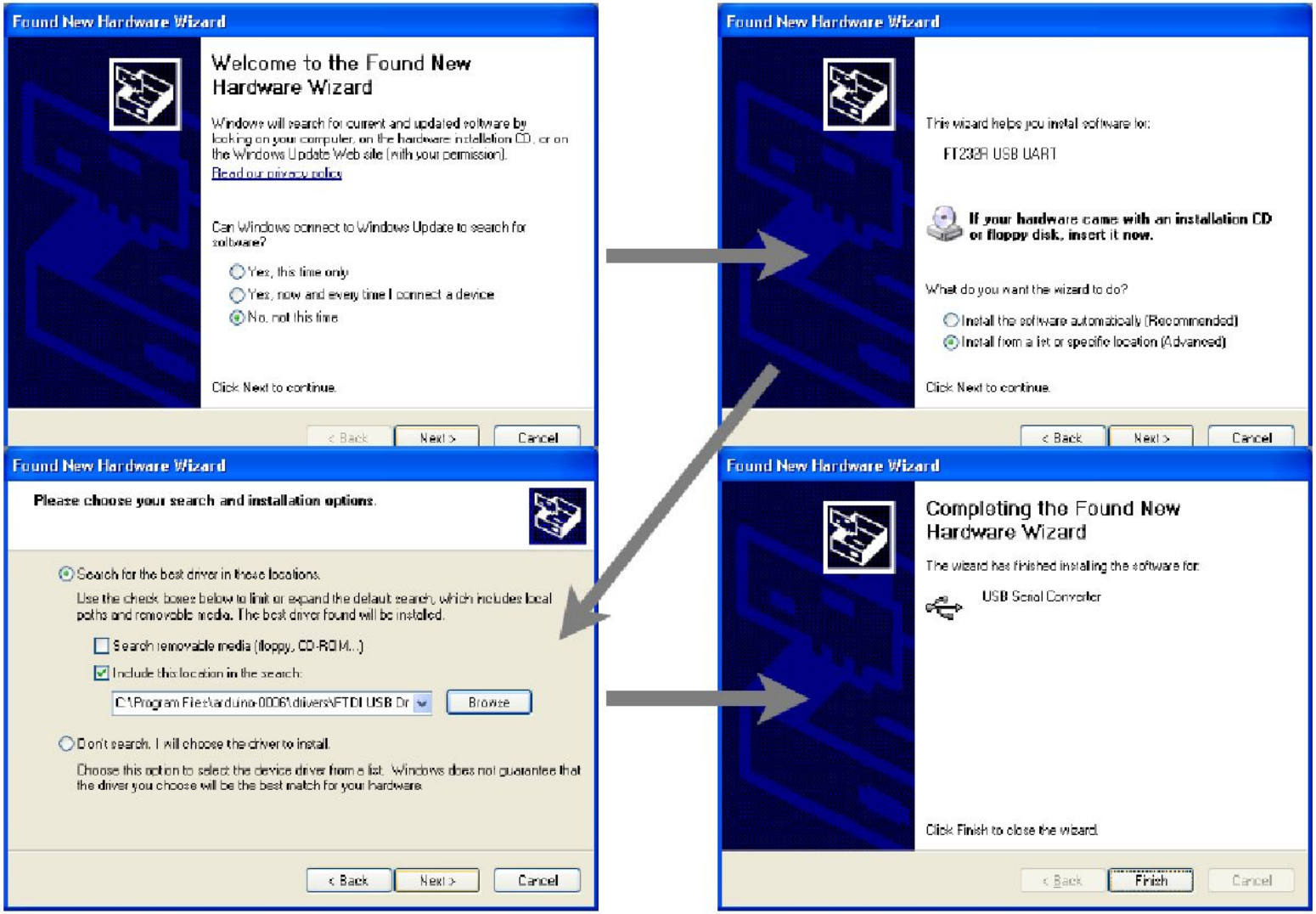
Installing Arduino

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

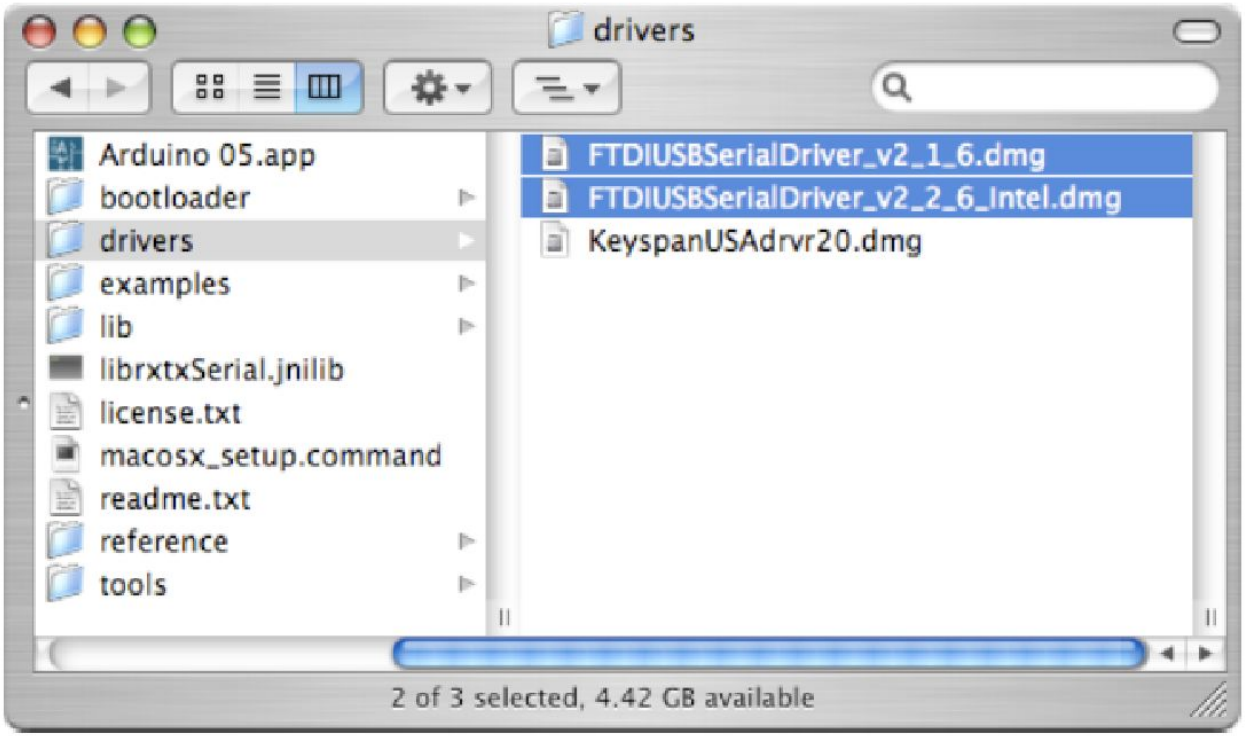


Windows Driver Install



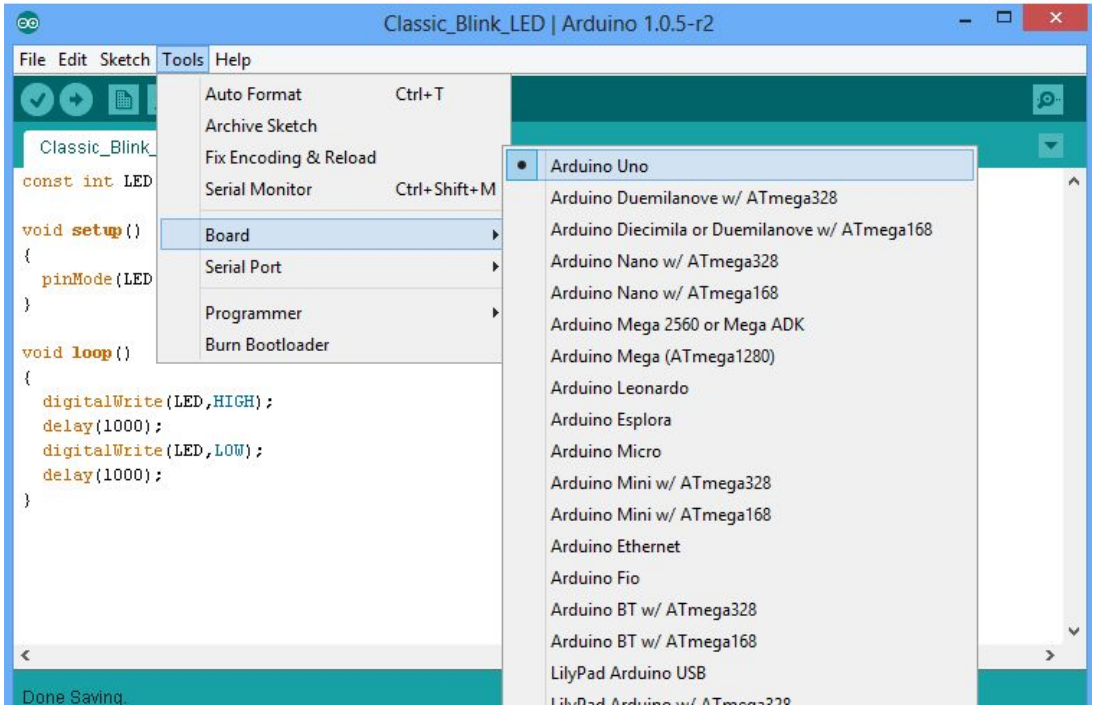
Mac Driver Install

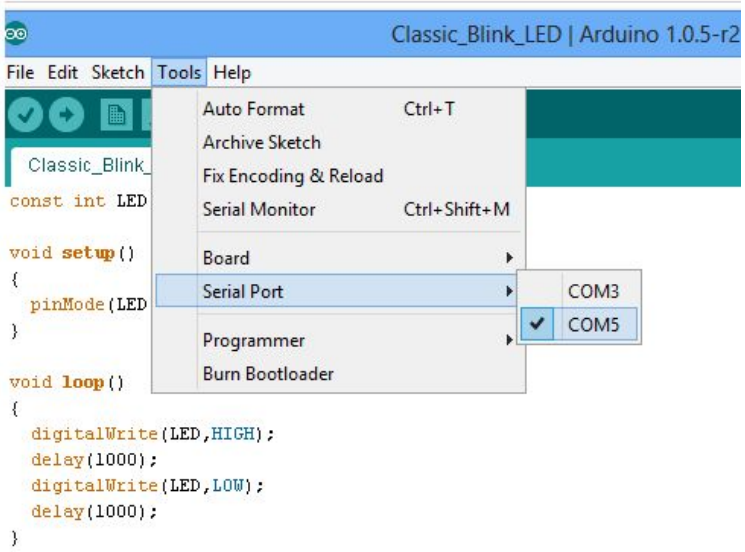
Double-click on .dmg Installer



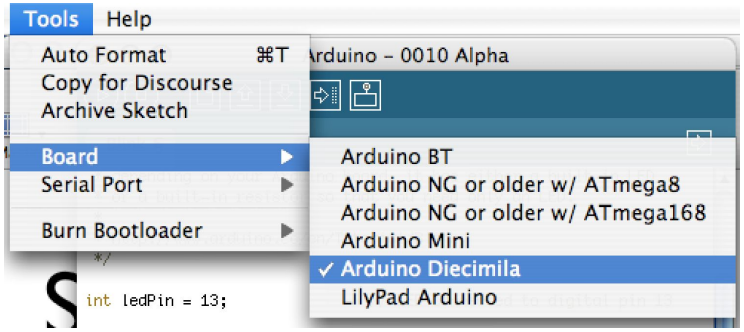
- v2_1_6 for PPC Macs
- v2_2_6 for Intel Macs

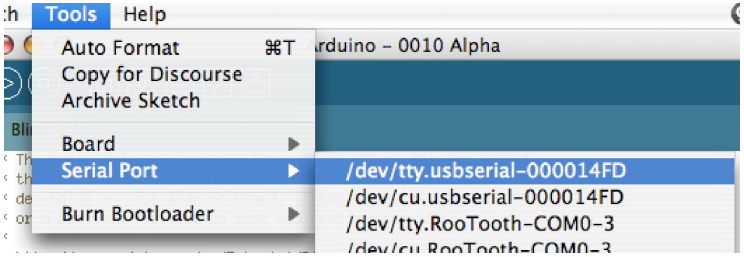
Selecting Location & Type





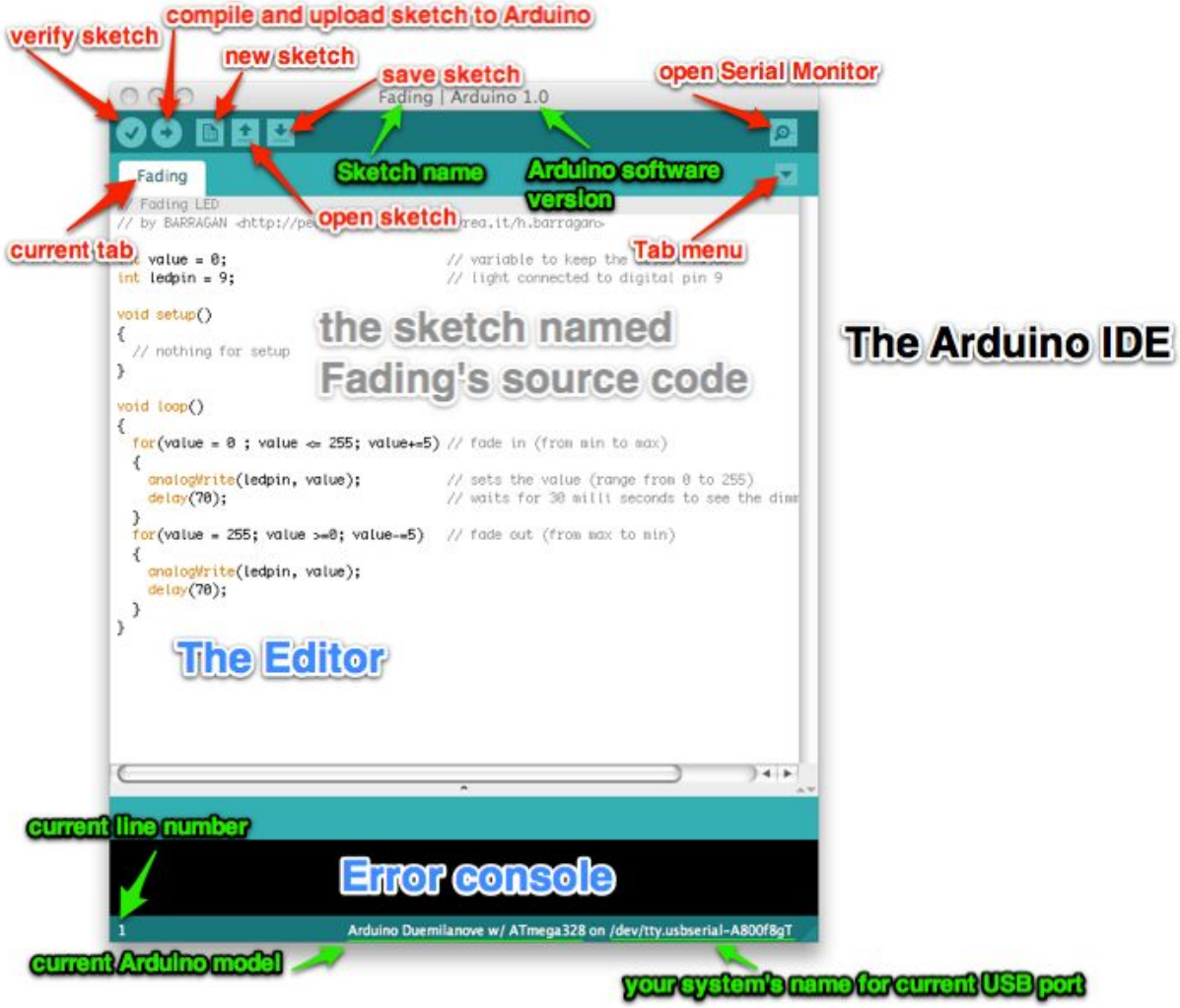
usually highest numbered port





starts with *tty.usbserial*

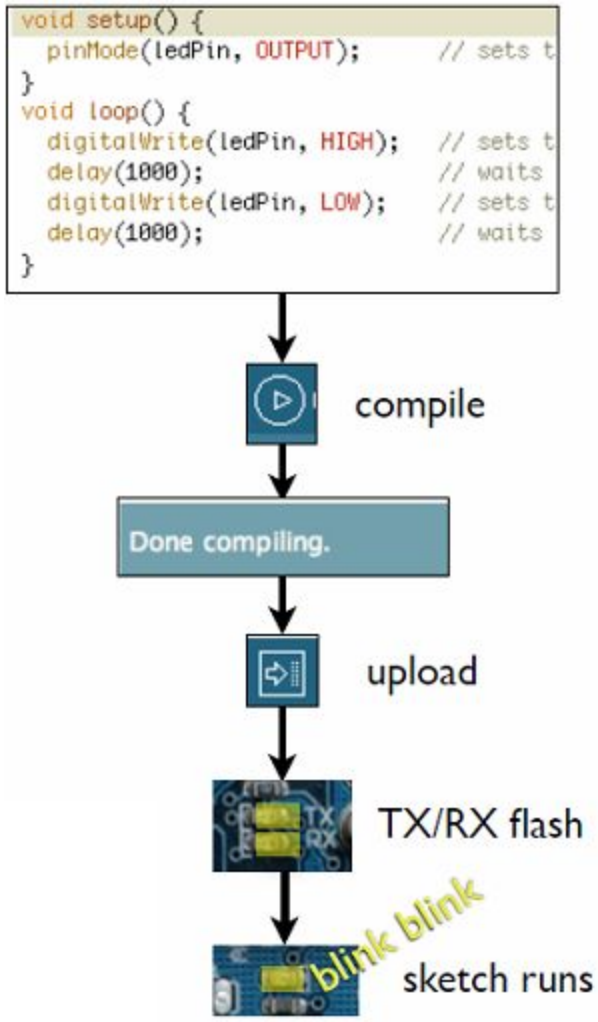
Arduino IDE


verify sketch
compile and upload sketch to Arduino
new sketch
save sketch
open Serial Monitor
Fading | Arduino 1.0
Sketch name
Arduino software version
Fading
current tab
open sketch
Tab menu
// Fading LED
// by BARRAGAN <http://pe
rea.it/h.barragan>
value = 0; // variable to keep the
int ledpin = 9; // light connected to digital pin 9
void setup()
{
// nothing for setup
}
void loop()
{
for(value = 0 ; value <= 255; value+=5) // fade in (from min to max)
{
analogWrite(ledpin, value); // sets the value (range from 0 to 255)
delay(70); // waits for 30 milli seconds to see the dim
}
for(value = 255; value >=0; value-=5) // fade out (from max to min)
{
analogWrite(ledpin, value);
delay(70);
}
}
the sketch named
Fading's source code
The Editor
The Arduino IDE
current line number
Error console
1
Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A800f8gT
current Arduino model
your system's name for current USB port

Using Arduino

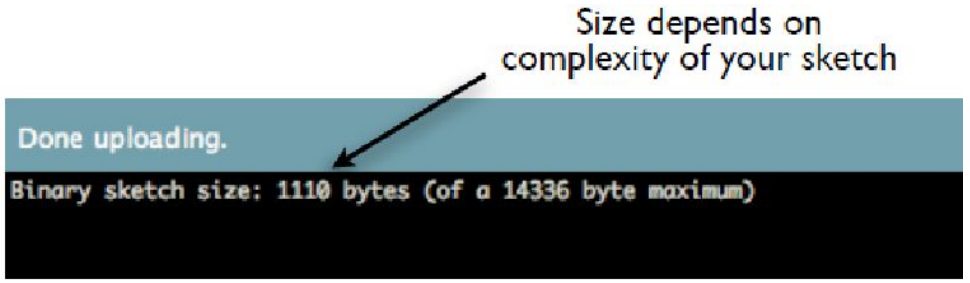
- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

- Try it out with the “Blink” sketch!
- Load “File/Examples/Basics/Blink”



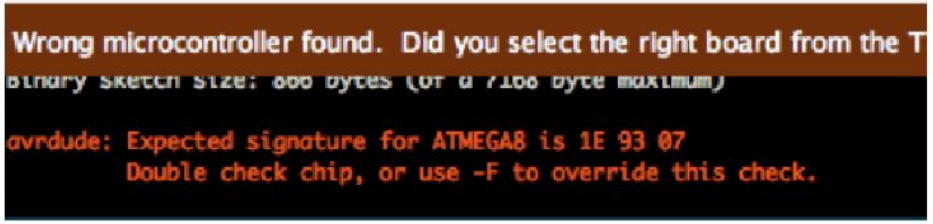
Status Messages

Uploading worked



Wrong serial port selected

Wrong board selected



nerdy cryptic error messages

Troubleshooting

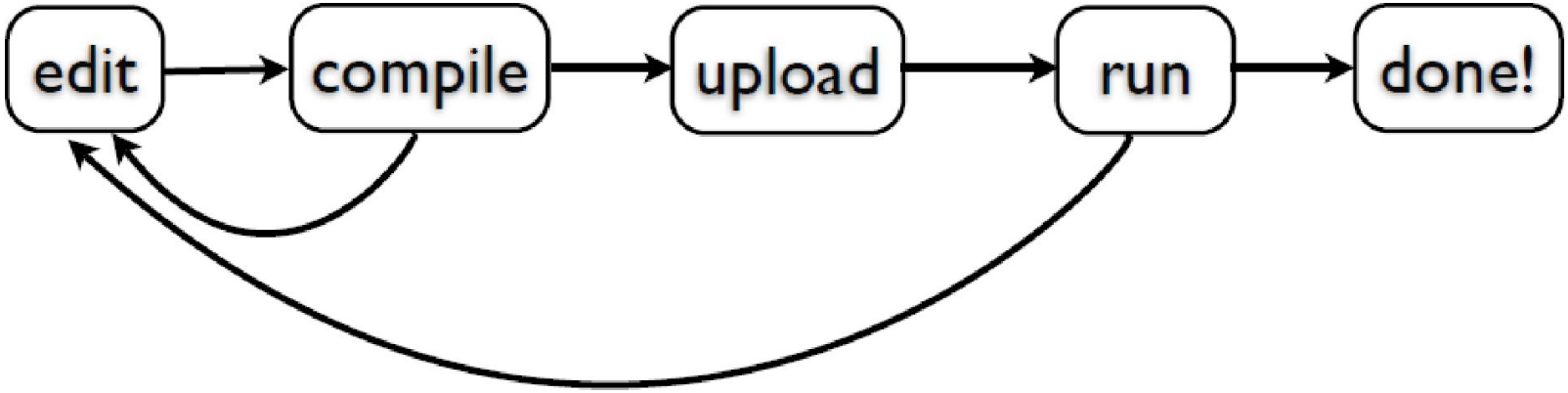
- Most common problem is incorrect serial port setting
- If you ever have any “weird” errors from the Arduino environment, just try again.
- The red text at the bottom is debugging output in case there may be a problem
- Status area shows summary of what’s wrong

I made an LED blink, so what?

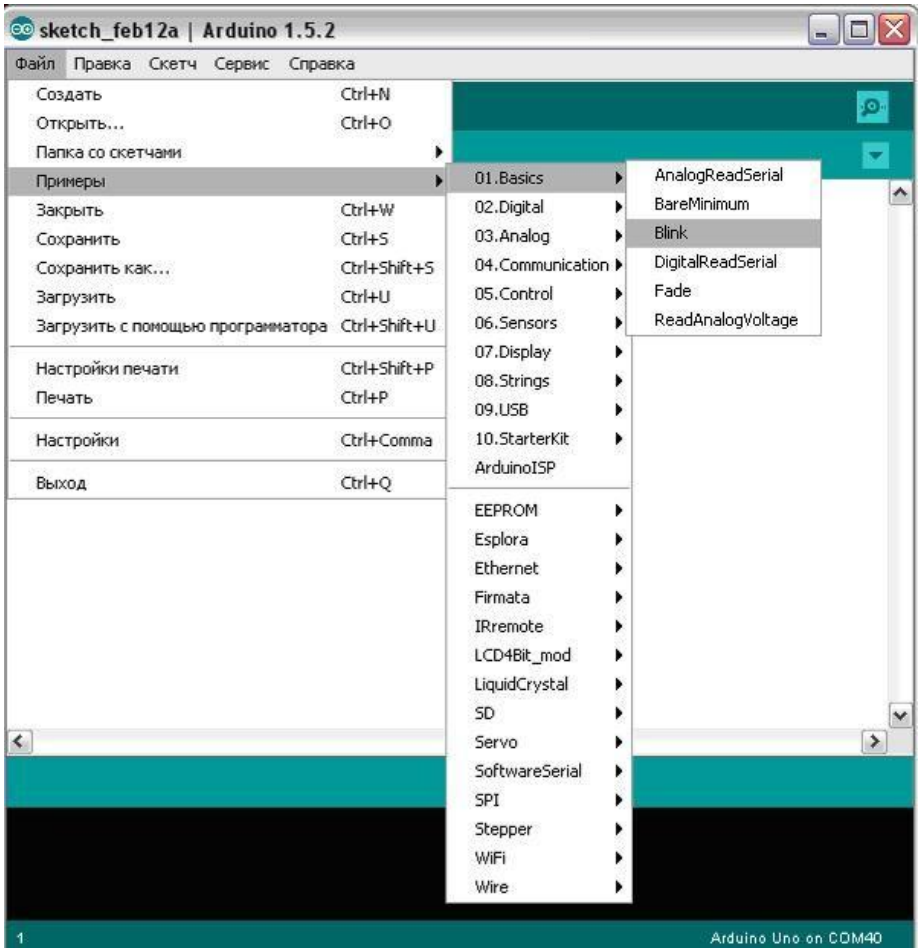
- Most actuators are switched on and off with a digital output
- The digitalWrite() command is the software portion of being able to control just about anything
- LEDs are easy, motors come in a bit
- Arduino has up to 13 digital outputs, and you easily can add more with helper chips

Development Cycle

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



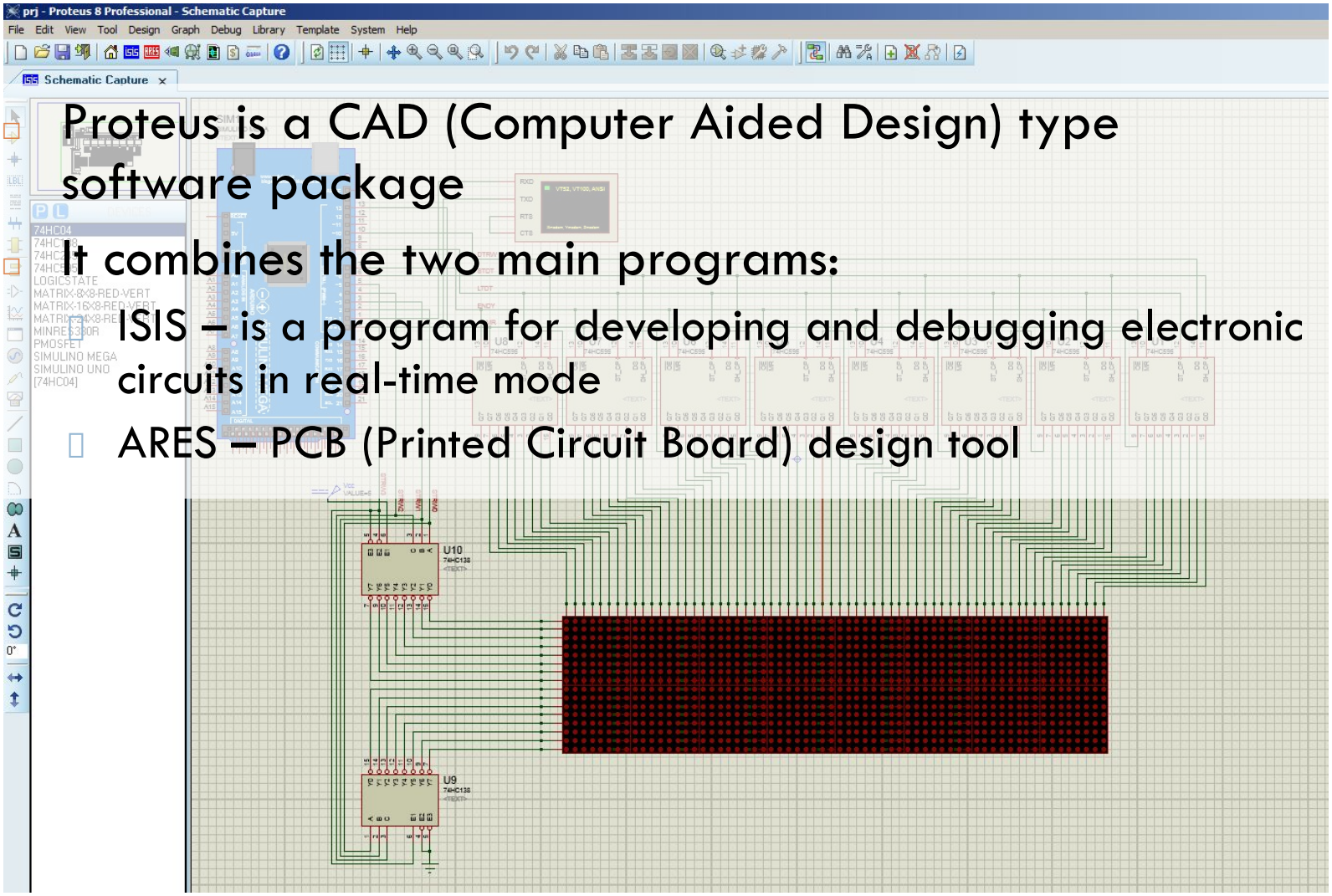
Lots of Built-in Examples



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

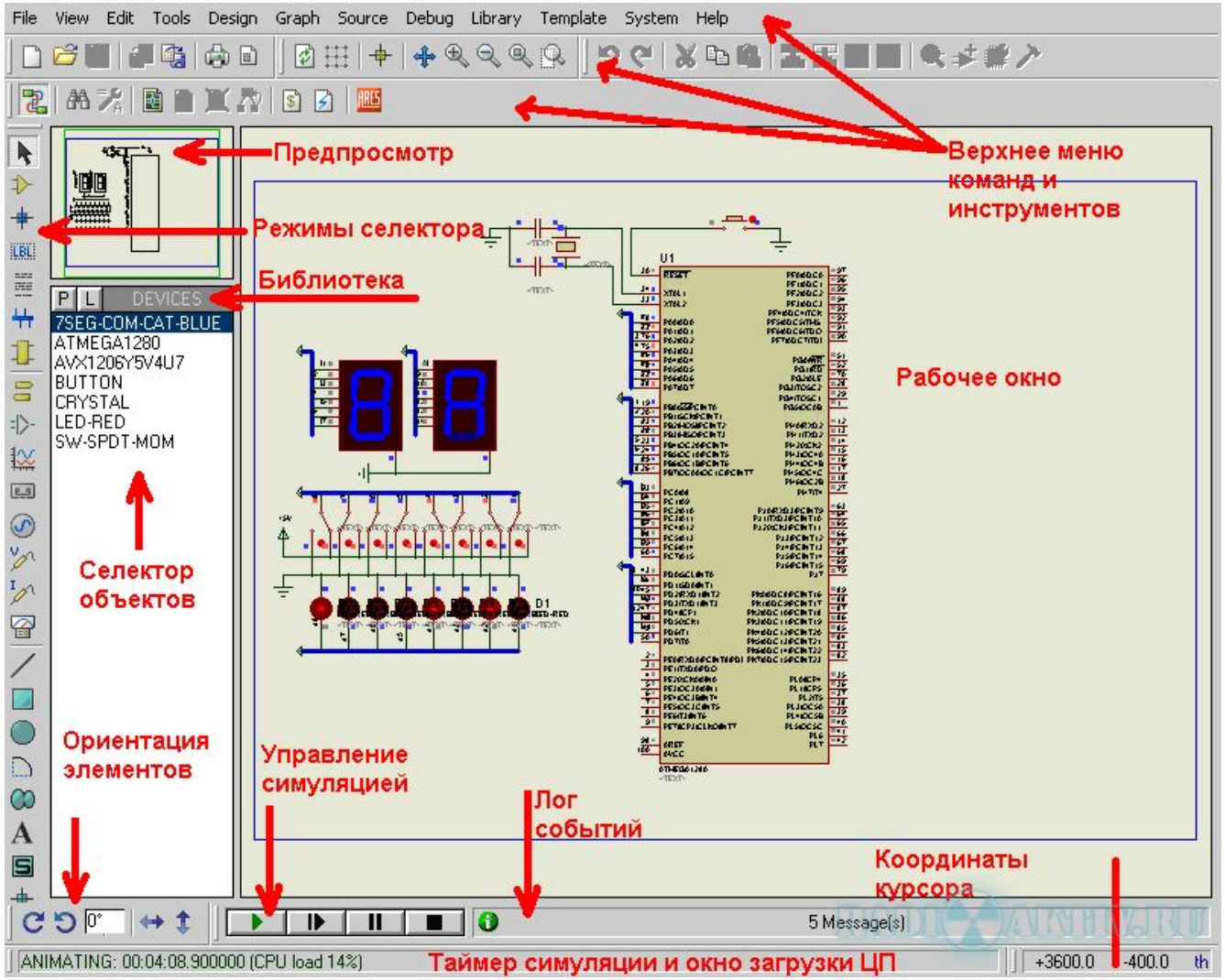
And all over the Net. Search for “Arduino tutorial” or “Arduino notes” or whatever you’re interested in and “Arduino” and likely you’ll find some neat pages.

Proteus ISIS Simulation System

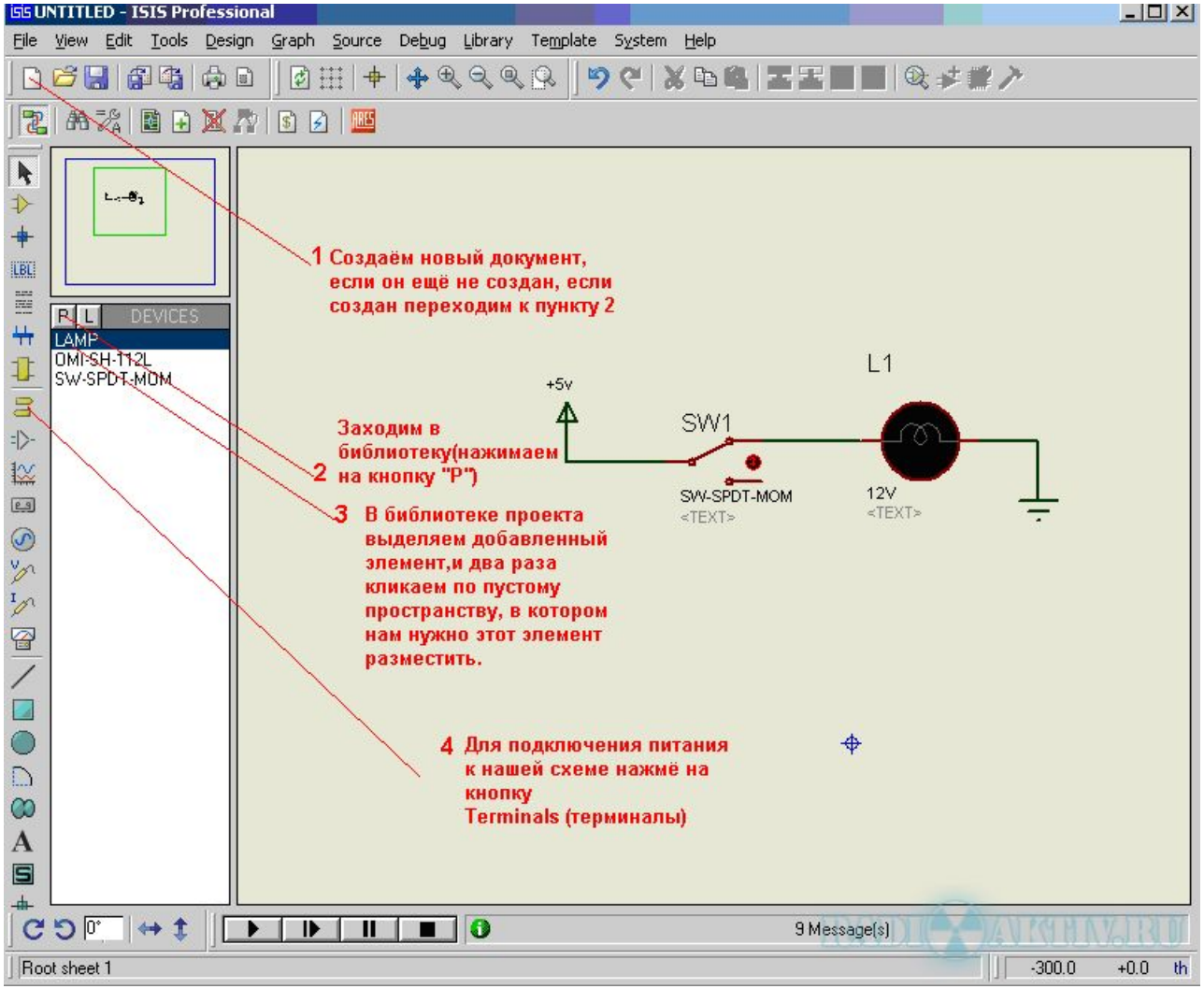


- Proteus is a CAD (Computer Aided Design) type software package
- It combines the two main programs:
  - ISIS – is a program for developing and debugging electronic circuits in real-time mode
  - ARES – PCB (Printed Circuit Board) design tool

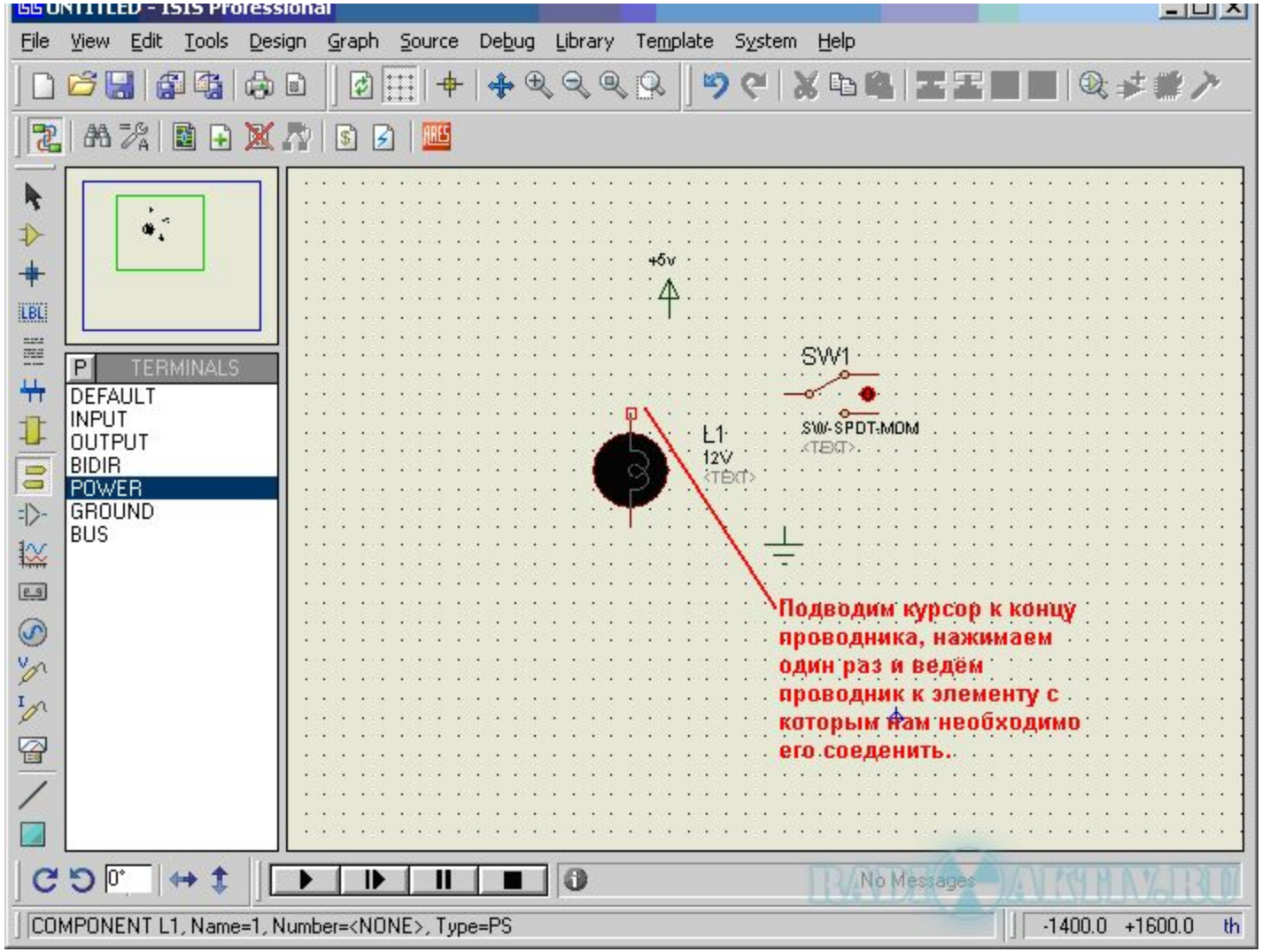
Proteus Menu and Navigation



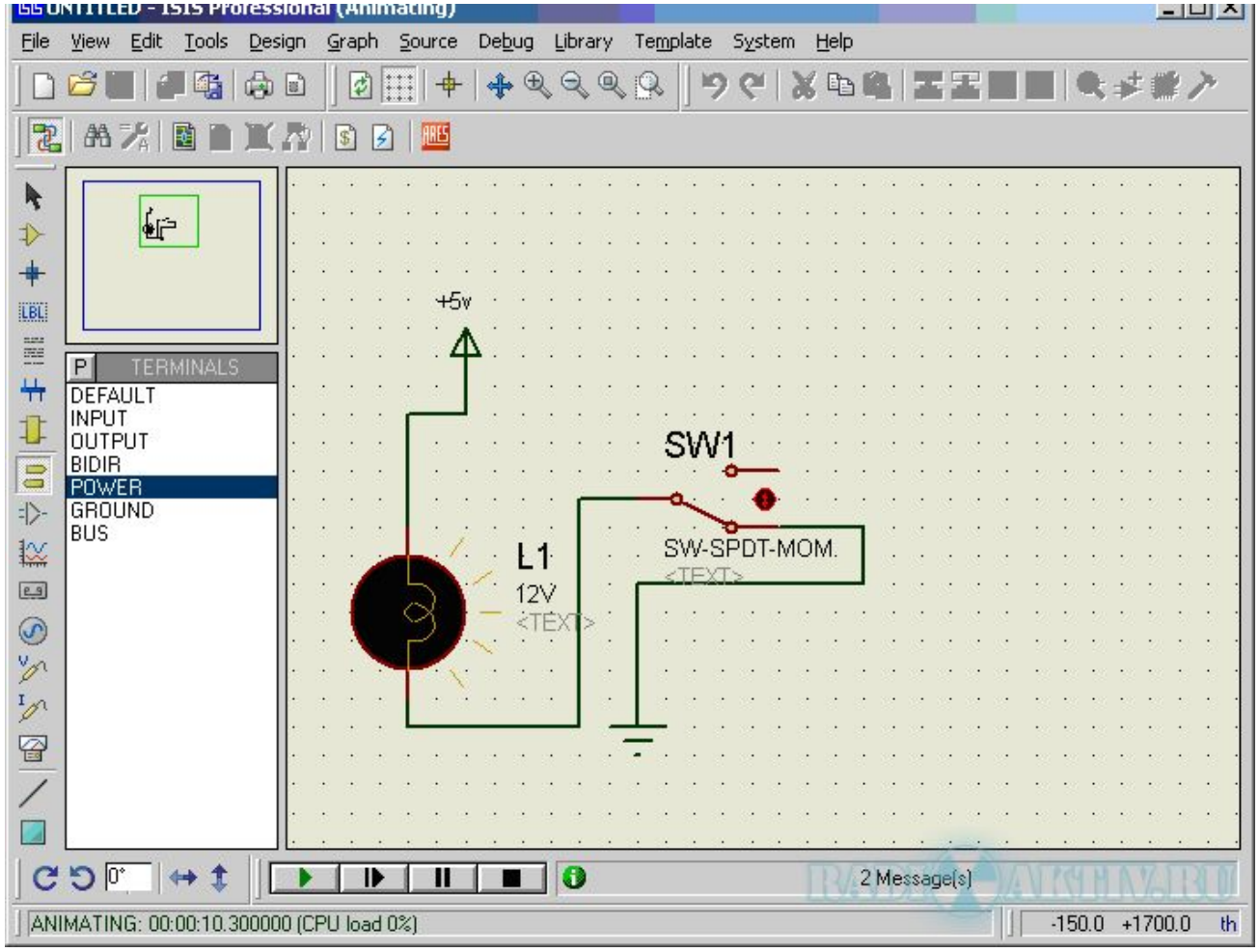
Creating a Circuit on Proteus



Connecting Elements



Simulation


File View Edit Tools Design Graph Source Debug Library Template System Help
TERMINALS
DEFAULT
INPUT
OUTPUT
BIDIR
POWER
GROUND
BUS
+5v
SW1
L1
12V
<TEXT>
SW-SPDT-MOM
<TEXT>
2 Message(s)
ANIMATING: 00:00:10.300000 (CPU load 0%)
-150.0 +1700.0

Installing Arduino Library for Proteus

- For Windows XP

Copy file BLOGEMBARCADO.LIB into:

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

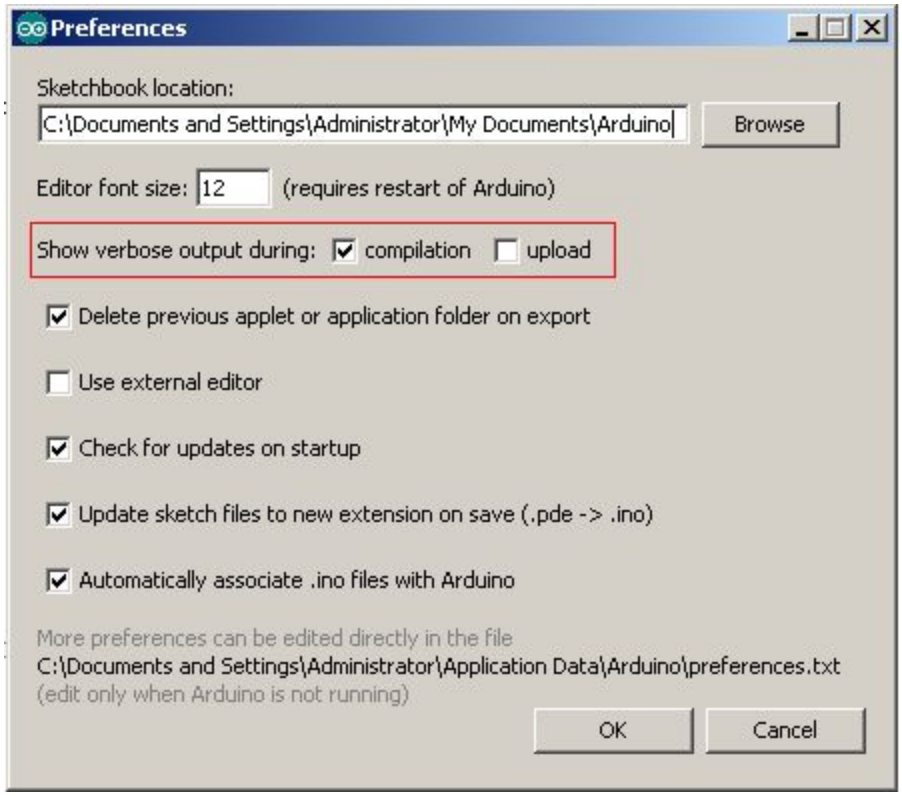
- For Windows 7 and later

Copy file BLOGEMBARCADO.LIB into:

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

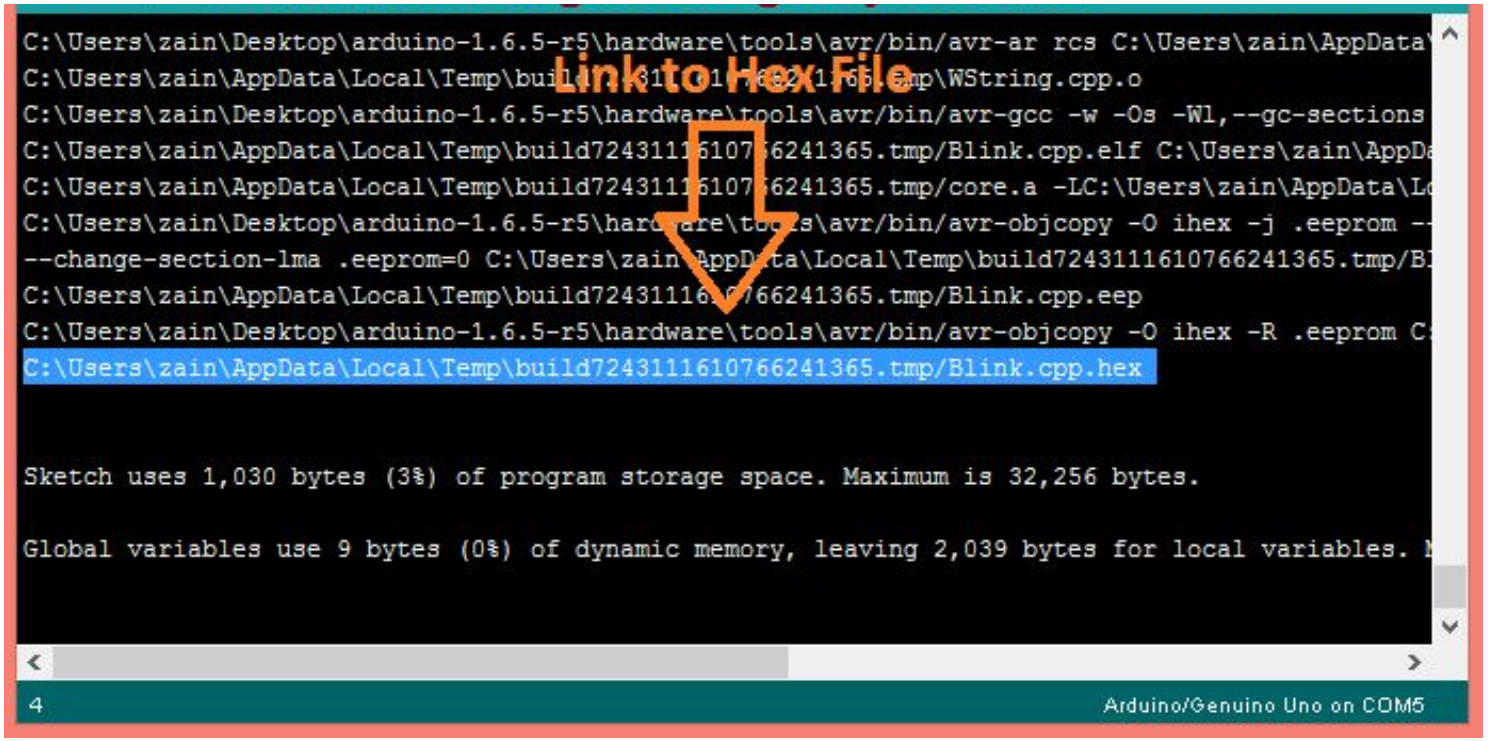
Loading the compiled file to Proteus

- File –> Preferences -> Show verbose output during **compilation**

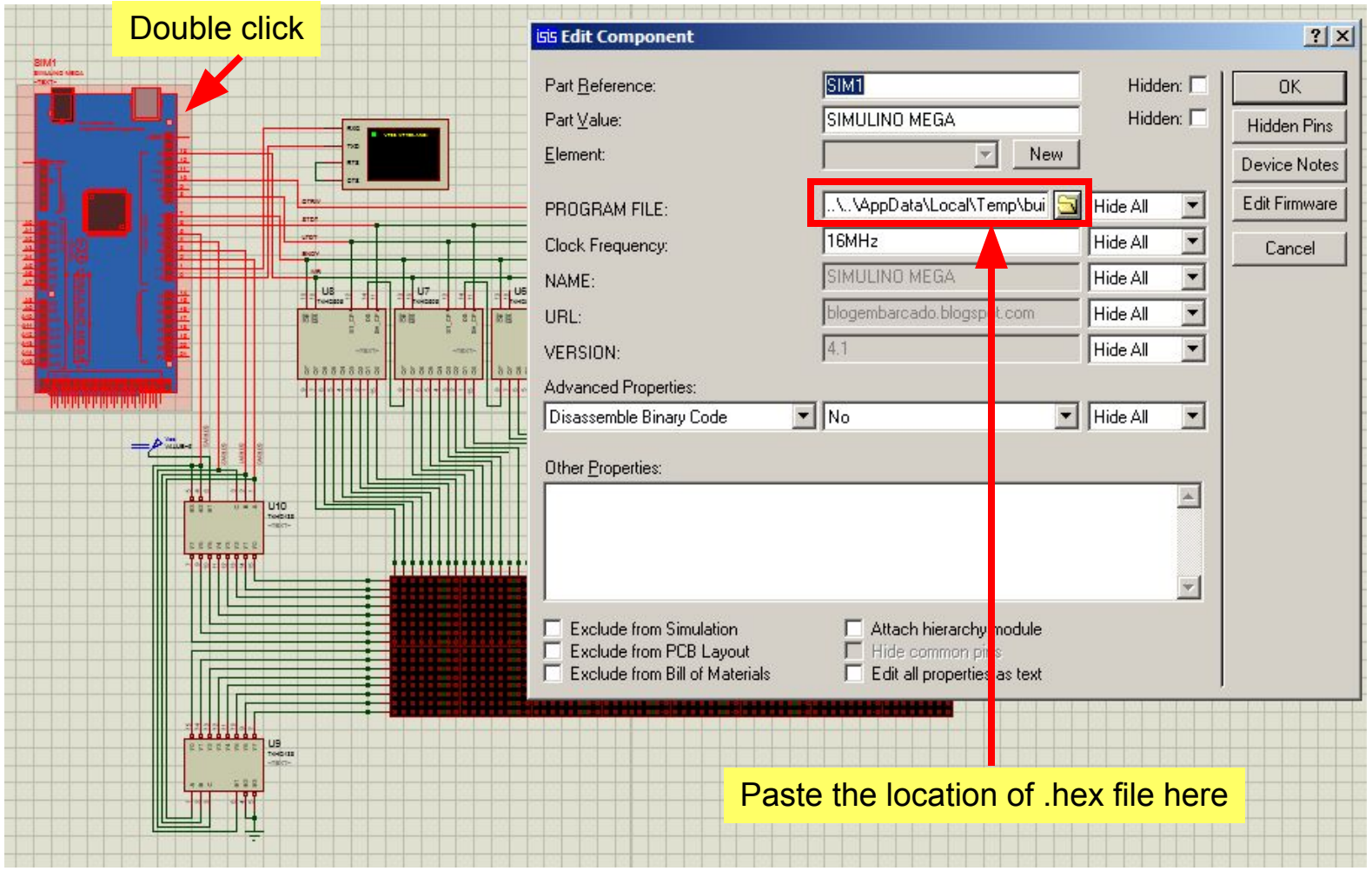


Loading the compiled file to Proteus

Select and copy the location of .hex file



Loading the compiled file to Proteus



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 12th 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 - 1s)
- Traffic lights (Rd-5s, Yl-1s, Gr-5s, Yl-1s …)
- 3 LEDs binary counter (0-7)
- 4 LED ripple