

Open Source Hardware

Robert Barron

Jorel Lalicki





vitalvio
SMARTER LIGHTING FOR SAFER HOSPITALS

Our Vision

To be at the forefront of the lighting disinfection industry and establish Vital Vio as the disruptive scientific and engineering thought leader in the field of disinfection technology and products

The Solution: Vital Vio Lighting Disinfection System

Safe, efficient overhead lighting & disinfection for sensitive environments

Consistent disinfection to reduce pathogenic microorganisms

Safe for continuous use around people

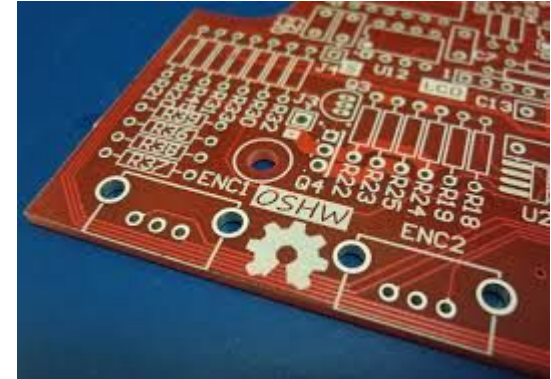
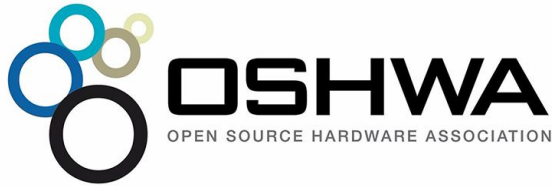
Retrofit **LED** unit for existing lighting



Daily reductions in bacteria, spores, fungus, yeast of up to 99% on exposed surfaces

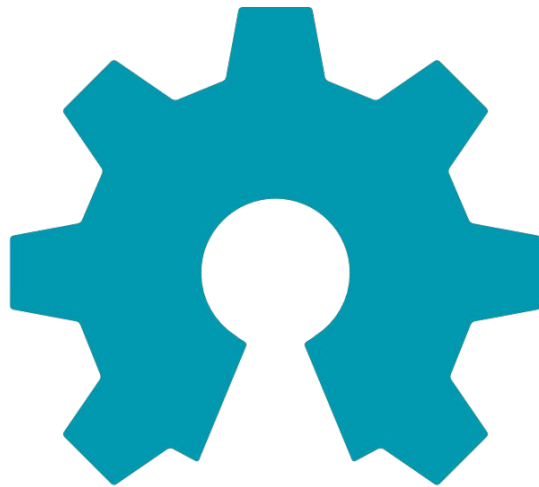
Open Source Hardware

- Open source hardware is hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design
 - The hardware's source, the design from which it is made, is available in the preferred format for making modifications to it.



Open Hardware?

You all (hopefully) know what
Open Source Software means.



open source
hardware

Open Source Hardware (OSHW) supports the same ideals as OSS, although the defining characteristics of an OSHW project are somewhat different.



OSHWA
OPEN SOURCE HARDWARE ASSOCIATION

Open Source Hardware Association

- Maintains a Community Definition of Open Source Hardware
- Educates the public about Open Source Hardware through events, conferences, etc
- Organizes the Open Source Hardware movement around shared values/principles
- Facilitates STEM education through the use of Open Source Hardware
- Collects, compiles, and publishes data on Open Source Hardware
- Administers the Open Source Hardware Certification Program (more on this later)

Open Source Hardware Community Definition

Version 1.0

1: Documentation

The hardware must be released with documentation including raw design files (in the preferred format for editing), and must allow modification and distribution of the design files.

This means that the PCB layout as a PDF or image is not enough - the source files used to generate the layout must also be available.

2: Scope

It must be clearly specified what portions of the design are released under the license.

3: Necessary Software

If the design requires software (embedded or otherwise), one of the following must be met:

- The interfaces are well documented, and it would be 'straightforward' to produce an open source version.
- The software is released under an OSI-approved license.

4: Derived Works

Modifications and derived works must be allowed under the same Open Source license, as well as manufacture, sale, distribution, and use of products created from the design.

5: Free Redistribution

Anyone can sell or give away the project documentation or derived works, without needing to pay a royalty fee.

6: Attribution

Attribution to the licensors can be required in derived documents and copyright notices, but you cannot specify the format of display.

7: No Discrimination Against People or Groups

This one should be pretty self-explanatory...

8: No Discrimination Against Fields of Endeavor

Anyone can use the work in any field - such as commercially, military, education, etc...

9: Distribution of License

The license must still apply when the work is redistributed, without the need for any additional licenses.

10: License Must Not Be Specific to a Product

The license must apply to any individual component, without regard to any particular product. This allows parts of a design to be used in new (different) products.

11: License Must Not Restrict Other Hardware or Software

The hardware must be allowed to be distributed with closed source components or software.

12: License Must Be Technology-Neutral

The license cannot require a particular technology, part, component, material, or style of interface.

OSHW Licenses

While OSS licenses control distribution of source code/documents (copyright law), OSHW licenses focus on manufacturing and use (patent law).

It is common to use adapted forms of OSI licenses such as GPL, LGPL, or BSD.

OpenCores (largest OSHW community) uses LGPL.

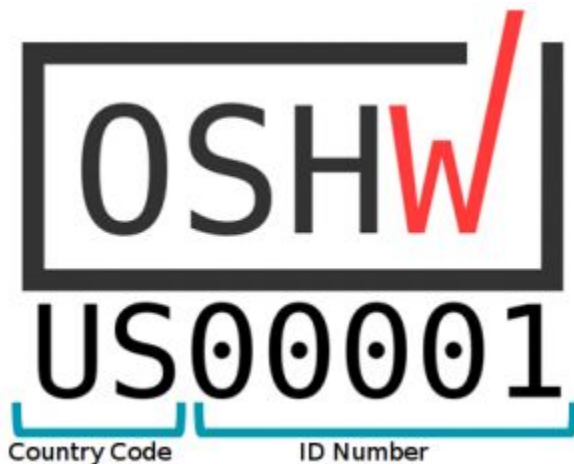
OSHW Now has an official certification process! (as of Oct 7, 2016)

- Complements the existing open gear logo
- New (proprietary) logo
- ID number to allow identification of a specific OSHW project
- Free

In order to use the certification logo, a hardware creator must make a **legally binding** promise that their hardware complies with the community definition of open source hardware.

Open Hardware?

The Open Source Hardware Certification program allows hardware that meets the standards of the community definition of open source hardware to display a certified logo and ID number.



Open Source Hardware Association - oshwa.org



Open Source Hardware



MUST

Allow anyone to study, modify, distribute, make, and sell the hardware.

Provide publicly accessible design files and documentation (the source).

Clearly specify what portion of the design, if not all, is being released under the license.

Not imply that derivatives are manufactured, sold, warranted, or otherwise sanctioned by the original designer.

Not use the trademarks of other companies without permission.

Not be released as non-commercial or no derivatives.

MAY

Require attribution be given.

Use the open source hardware logo to signify their hardware follows the open source hardware definition.

Require derived works to carry a different name or version number from the original design.

Be copied directly or have derivatives created from it.

Require a viral license.



Created by the Open Source Hardware Association
Learn more at oshwa.org



MUST

Allow anyone to study, modify, distribute, make, and sell the hardware.

Provide publicly accessible design files and documentation (the source).

Clearly specify what portion of the design, if not all, is being released under the license.

Not imply that derivatives are manufactured, sold, warranted, or otherwise sanctioned by the original designer.

Not use the trademarks of other companies without permission.

Not be released as non-commercial or no derivatives.

MAY

Require attribution be given.

Use the open source hardware logo to signify their hardware follows the open source hardware definition.

Require derived works to carry a different name or version number from the original design.

Be copied directly or have derivatives created from it.

Require a viral license.

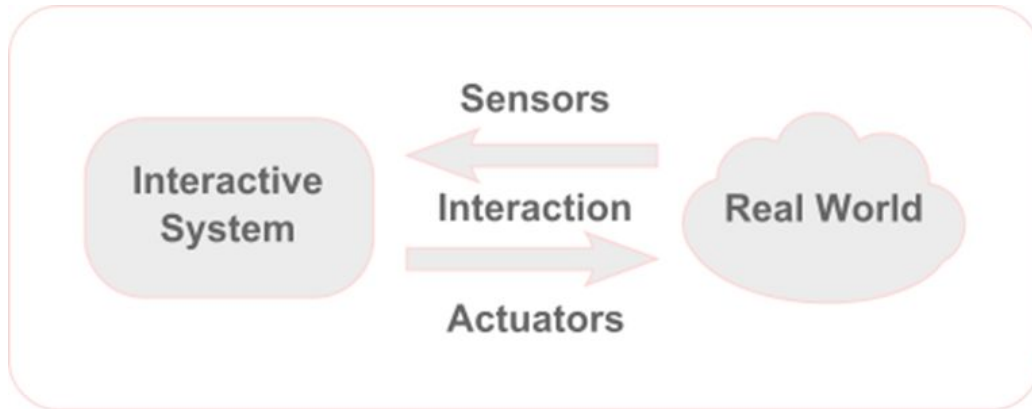
— Copyleft, like the GPL or CC-BY-SA

Some of the companies will certify at least one OSHW product in 2016



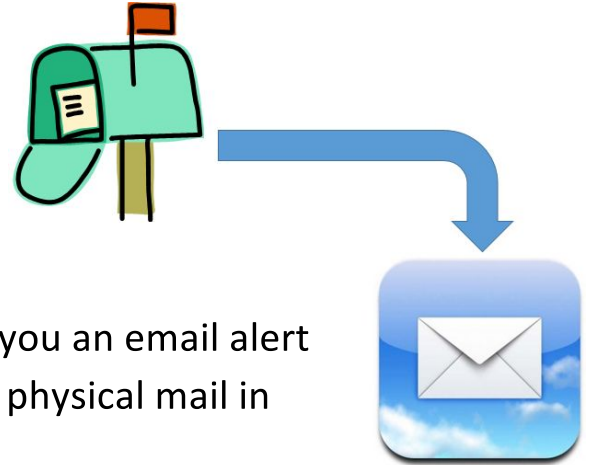
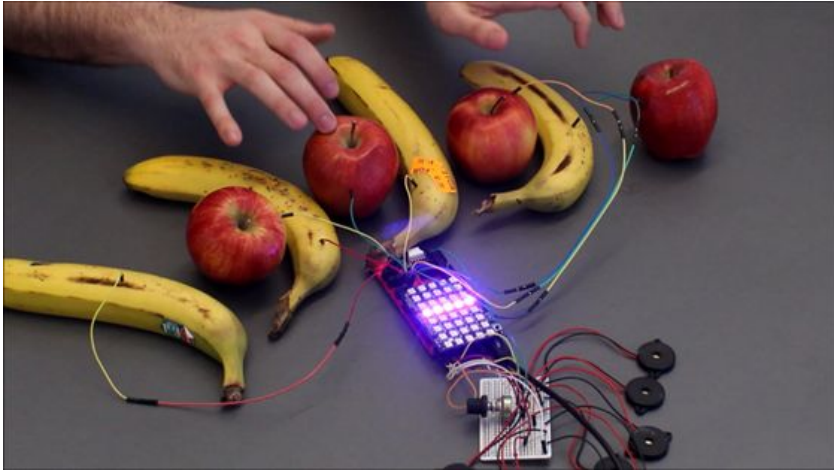
What is Physical Computing

- The process of integrating existing electronics into tangible, real systems.
 - It is a way for a person to interact with what they create, and for creations to interact with it's creator.



Physical Computing

Physical Computing lets you play a keyboard made of fruit



Or it can give you an email alert when you get physical mail in your mailbox.

- Possibilities are endless!
- Real connections can be made with technology

Raspberry Pi

- A Credit Card sized PC
- Can be plugged into a monitor or used remotely (SSH, VNC)
 - Use with keyboard and mouse like a normal computer
- Various Linux Distros available (Raspbian, Ubuntu, Windows 10 IOT, XBMC and more)
- Quad Core ARM CPU and 1GB of RAM for \$35
- Designed for Education, but used for much more
 - Scratch, Python, Mathematica and more



Raspberry Pi Projects

Arcade Machine



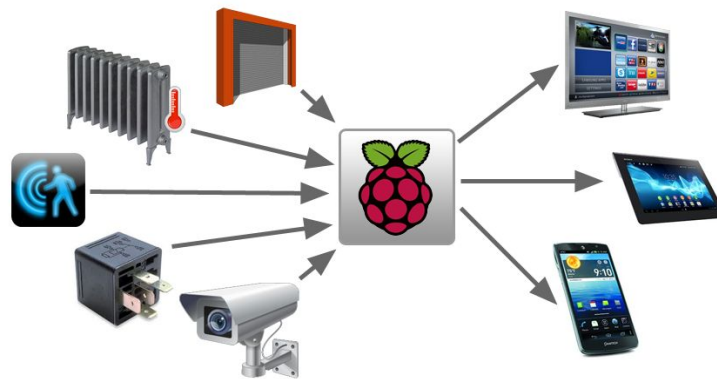
PiSoC - [Face Tracker](#)



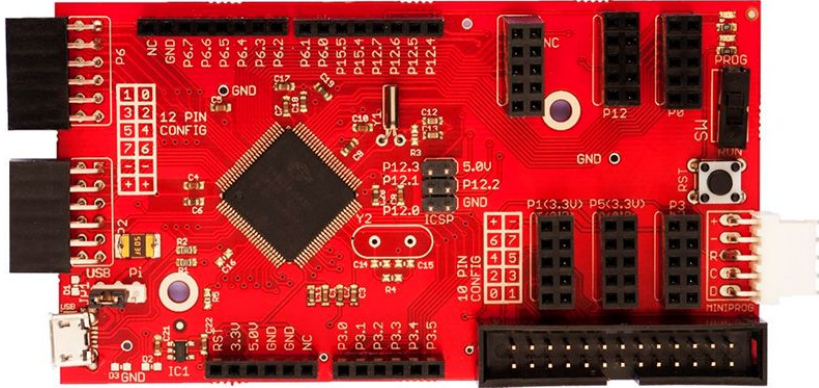
“Super” Computer



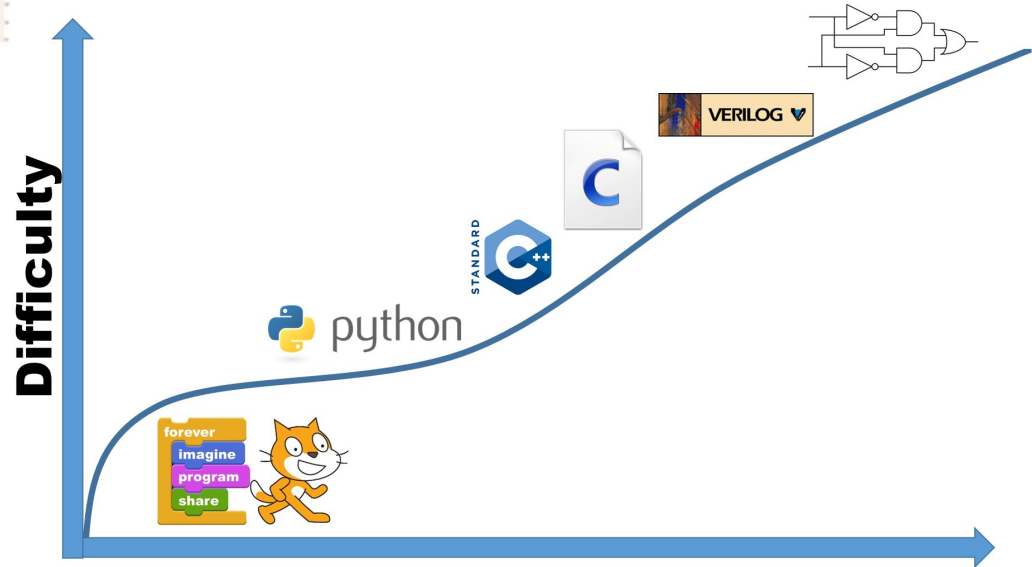
Home Automation

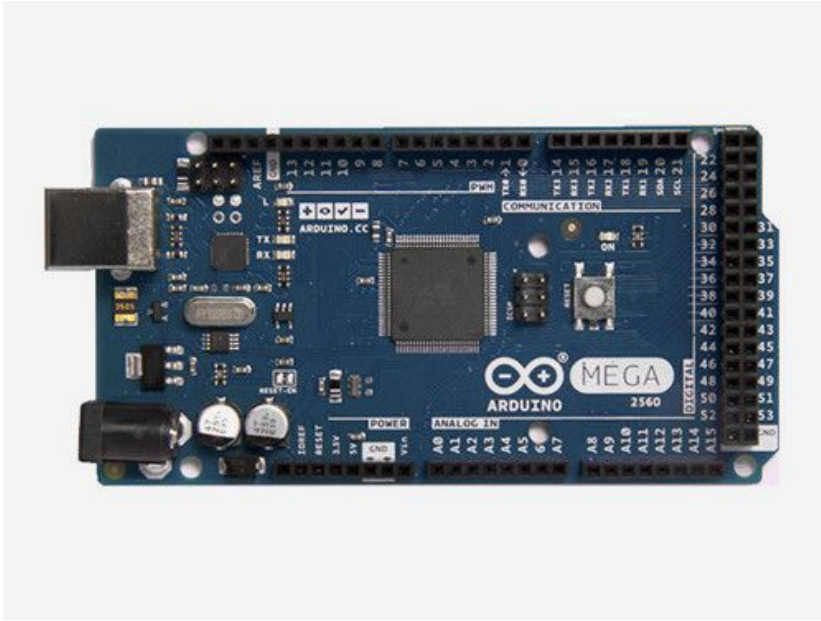


PiSoC - Physical Computing Made Easy



We have extended the Scratch Language, and created Python libraries in order to introduce Physical computing to a new generation





Website - [Arduino.cc](https://www.arduino.cc)

Please download the IDE on the website before
Fridays's Lab!

What?

- Physical computing platform
- Open source
- “Hardware Abstracted” Wiring Language
- USB programmable
- Large community
- Multi platform Win/Mac/Linux
- Inexpensive

- Based on ATmega328 – 8 BIT

- USB interface

- Voltage regulator

- Specs

 - RISC @ 16 Mhz, 20 MIPS

 - 32 K Memory

 - 6 Ch 10 Bit A/D

 - PWM, I2C, SPI

- The “power” is in: Standard board design, Wiring language, Open Source**



Blink

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  This example code is in the public domain.  
  */  
  
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  digitalWrite(13, HIGH); // set the LED on  
  delay(1000);            // wait for a second  
  digitalWrite(13, LOW);  // set the LED off  
  delay(1000);            // wait for a second  
}
```

Actual Code

Console



Blink

```
/*  
  Blink  
  Turns on an LED  
  on for one second, then  
  off for one second.  
  This example is  
  commented out to  
  prevent the Arduino IDE  
  from compiling it  
  as a sketch.  
*/
```

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

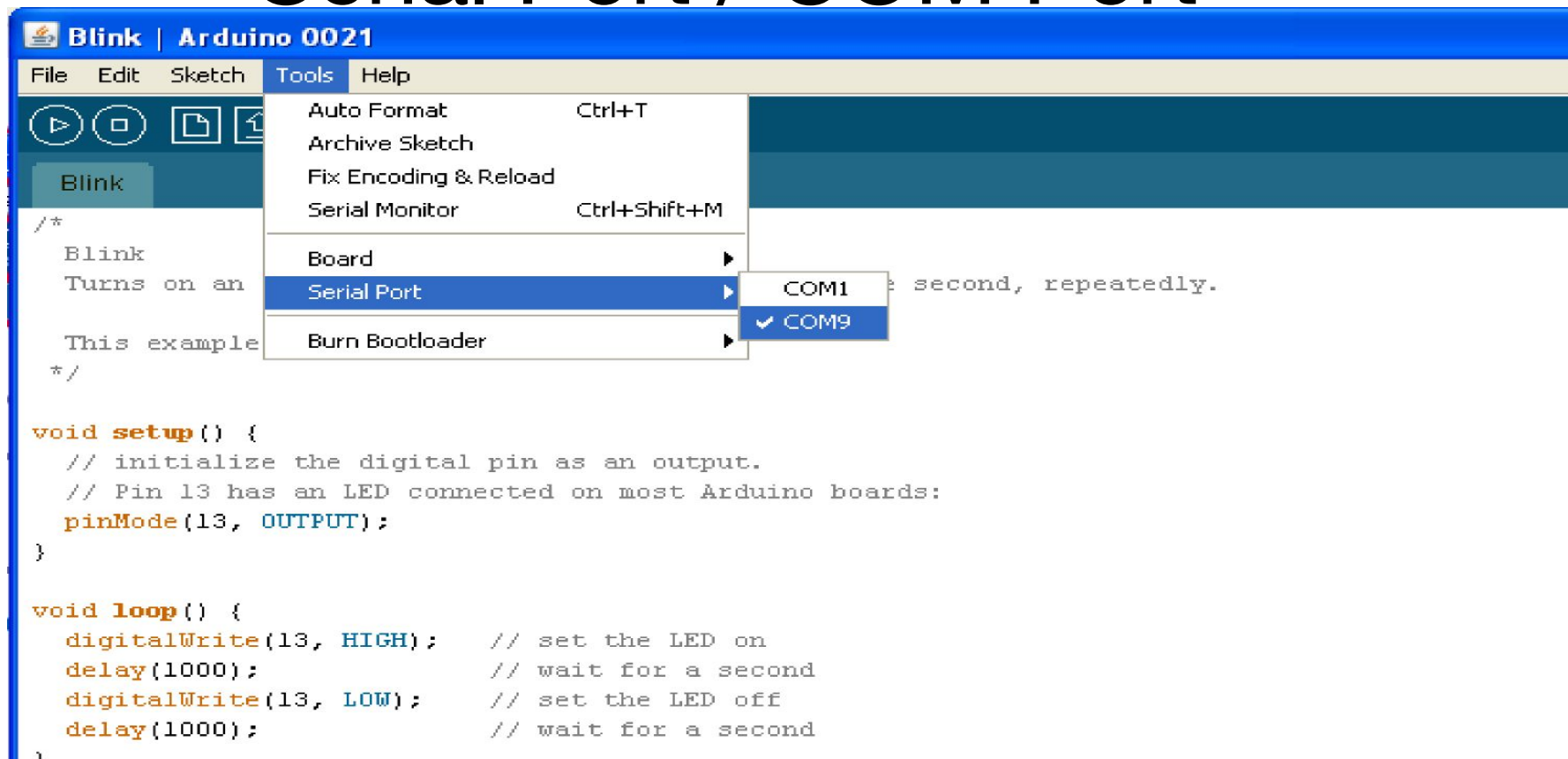
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M

Board
Serial Port
Burn Bootloader

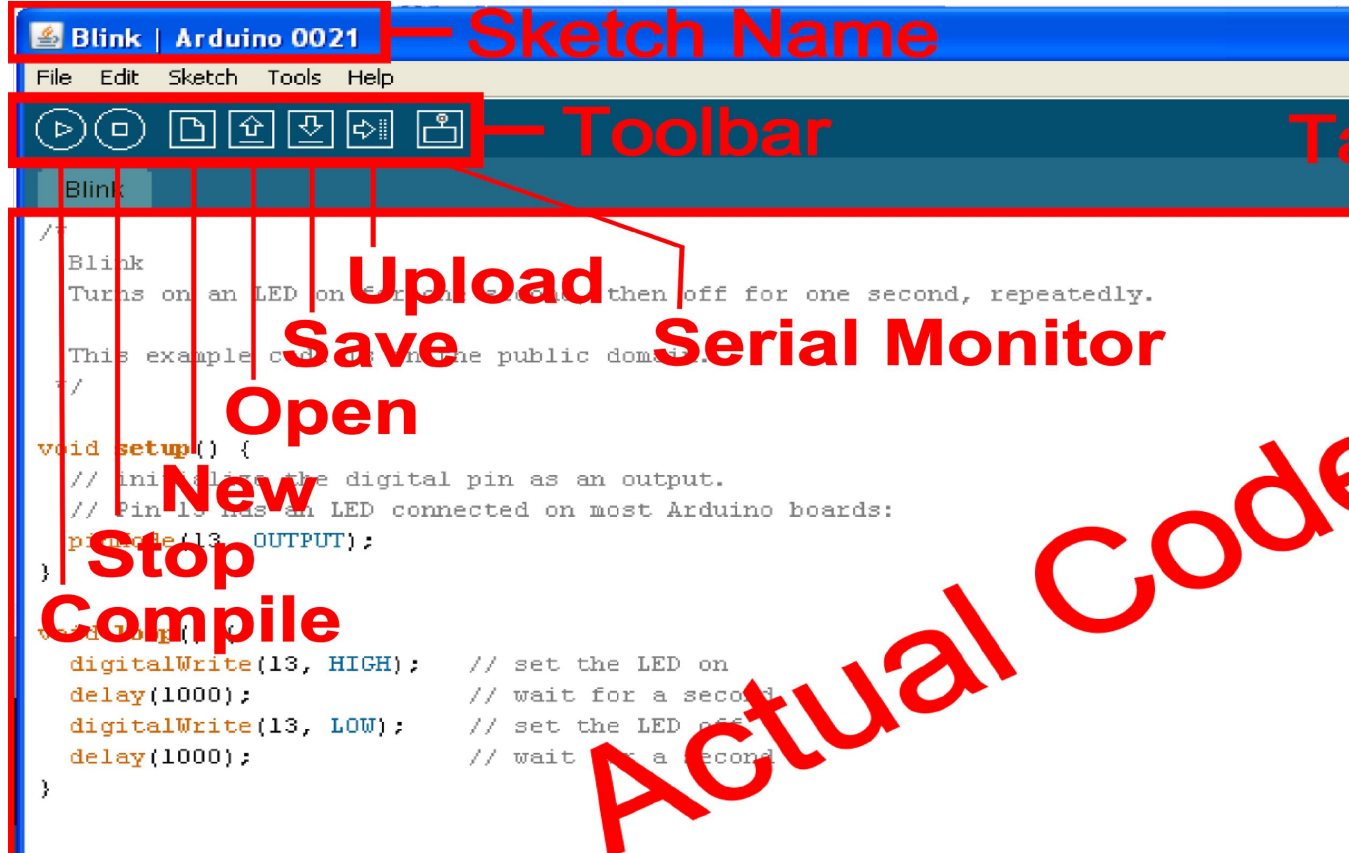
Arduino Uno

- Arduino Duemilanove or Nano w/ ATmega328
- Arduino Diecimila, Duemilanove, or Nano w/ ATmega168
- Arduino Mega 2560
- Arduino Mega (ATmega1280)
- Arduino Mini
- Arduino Fio
- Arduino BT w/ ATmega328
- Arduino BT w/ ATmega168
- LilyPad Arduino w/ ATmega328
- LilyPad Arduino w/ ATmega168
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
- Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
- Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
- Arduino NG or older w/ ATmega168
- Arduino NG or older w/ ATmega8

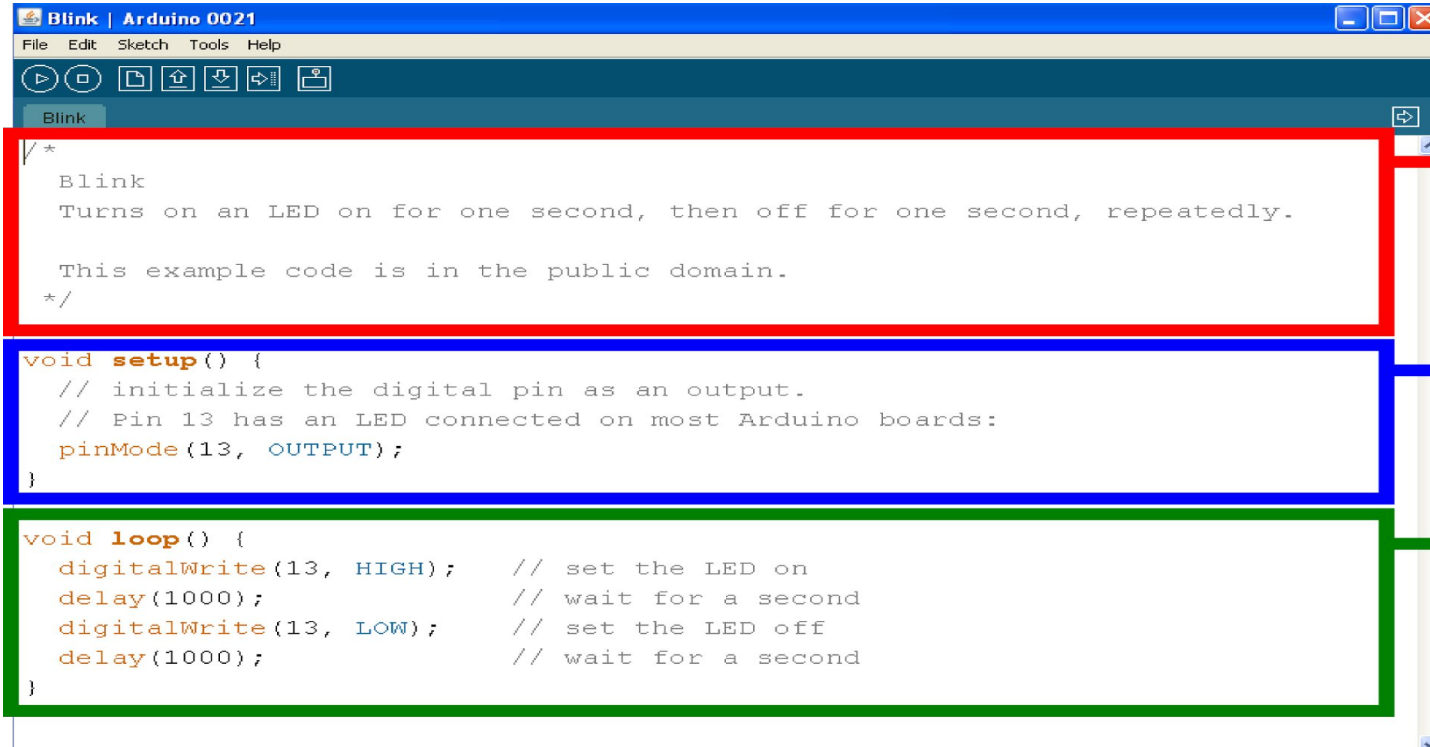
Serial Port / COM Port



The Environment



Parts of the Sketch



**Comments /
Explaining
the game**

**Setup /
Stretching or
tying shoes**

**Loop /
Playing the
game**

Setup

void setup () { }

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

The setup function comes before the loop function and is necessary for all Arduino sketches

Setup

***void setup () {
pinMode (13, OUTPUT); }***

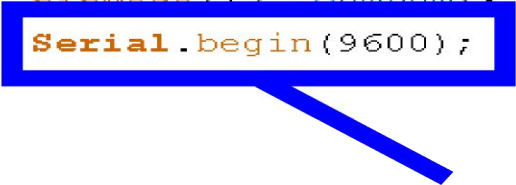
```
void setup() {  
  // initialize the digital pin as an output.  
  // pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
}
```

Outputs are declared in setup, this is done by using the pinMode function

Setup

void setup () { **Serial.begin;** }

```
void setup() {  
  // initialize the digital pin as an output.  
  // Pin 13 has an LED connected on most Arduino boards:  
  pinMode(13, OUTPUT);  
  Serial.begin(9600);  
}
```



Serial communication also begins in setup

This particular example declares Serial communication at a baud rate of 9600. More on Serial later...

Basic Repetition

Or maybe:

```
while ( digitalRead(buttonPin)==1 )  
{  
  //instead of changing a variable  
  //you just read a pin so the computer  
  //exits when you press a button  
  //or a sensor is tripped  
}
```