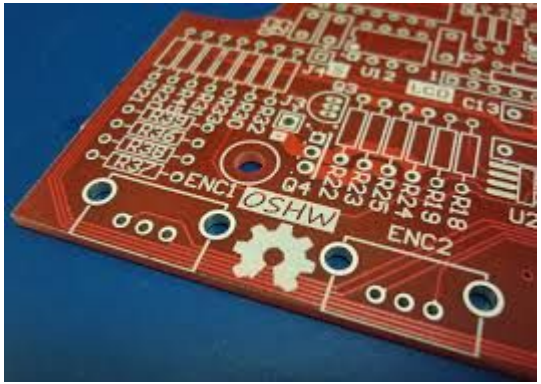
Open Source Hardware

Robert Barron
Jorel Lalicki

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.

OSHWA
OPEN SOURCE HARDWARE ASSOCIATION



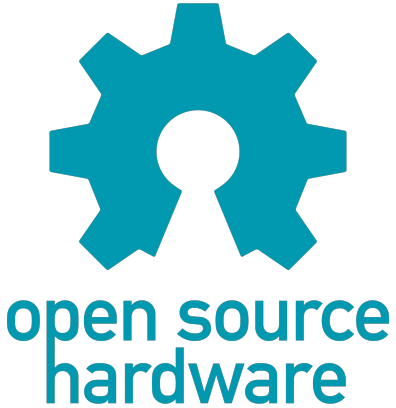
Open Hardware?

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

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

Open Hardware?

The 'OSHW Definition 1.0' lists 12 key attributes a hardware project must have to be considered Open Source.



Open Source Hardware Association - oshwa.org

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

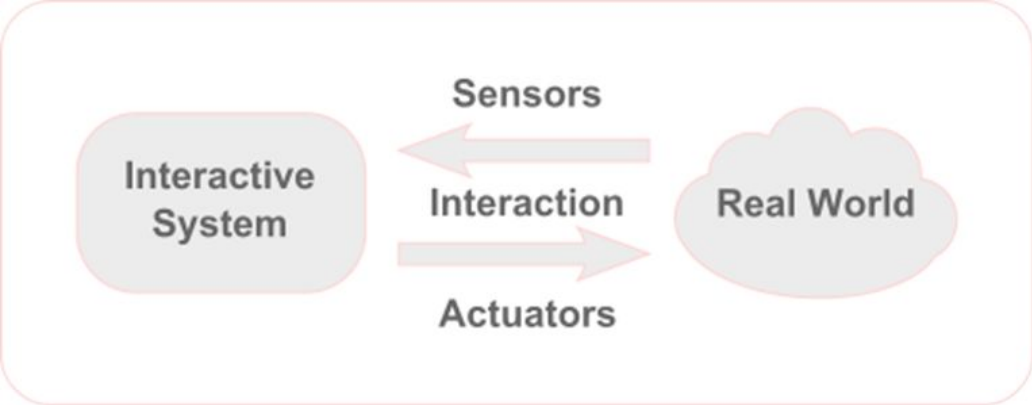
While OSS licenses control distribution of source code/documents (copyright law), OSHW licenses focus on manufacturing and use (patent law).

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

OpenCores (largest OSHW community) uses LGPL.

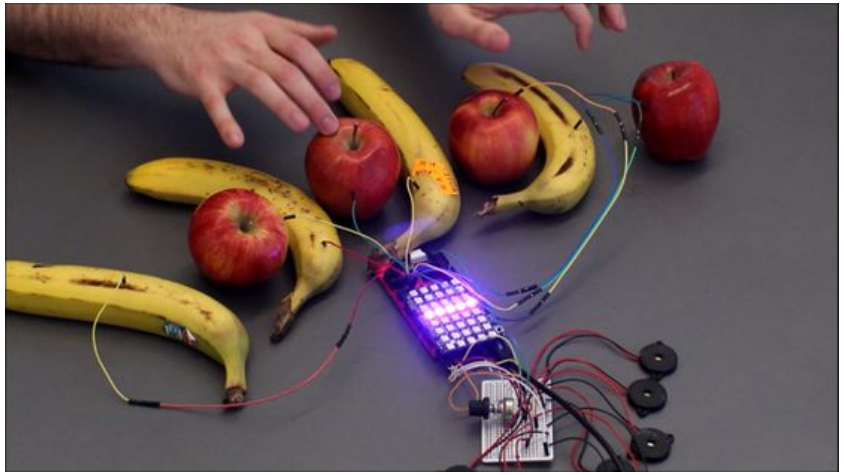
What is Physical Computing

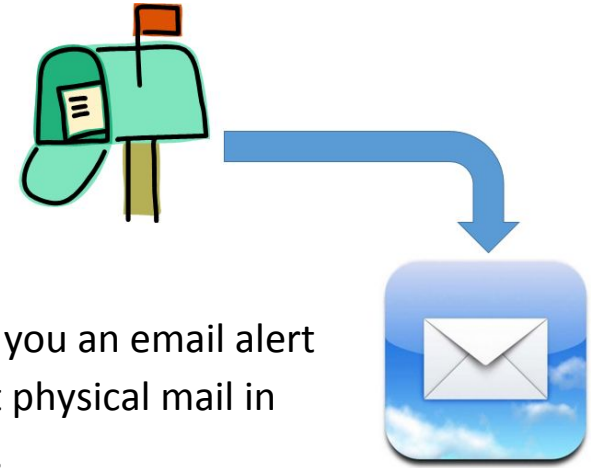
- The process of integrating existing electronics into tangible, real systems.
  - It is a way for a person to interact with what they create, and for creations to interact with it's creator.



Physical Computing

Physical Computing lets you play a keyboard made of fruit



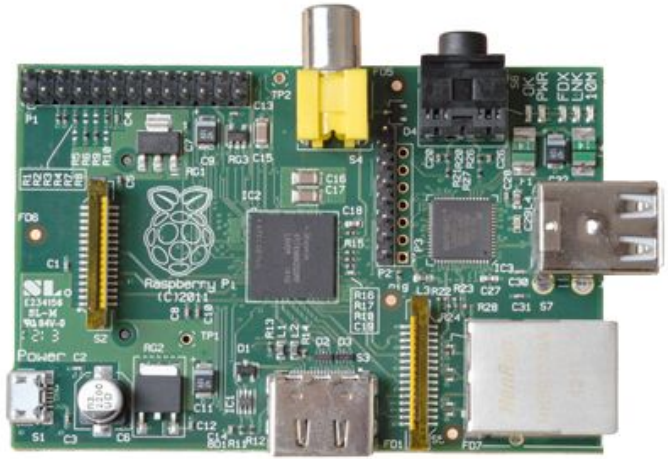


Or it can give you an email alert when you get physical mail in your mailbox.

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

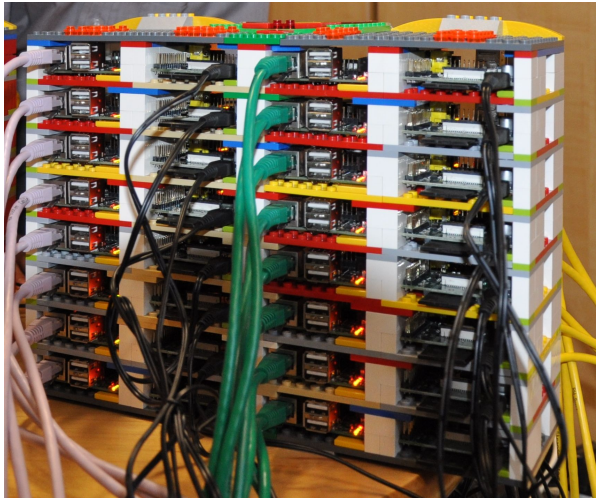
Raspberry Pi

- 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

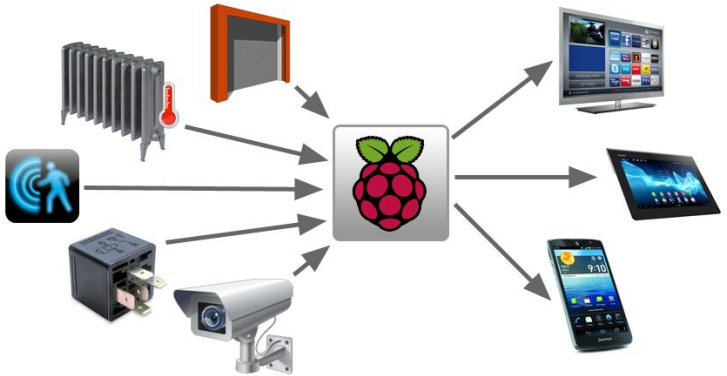
Super Computer



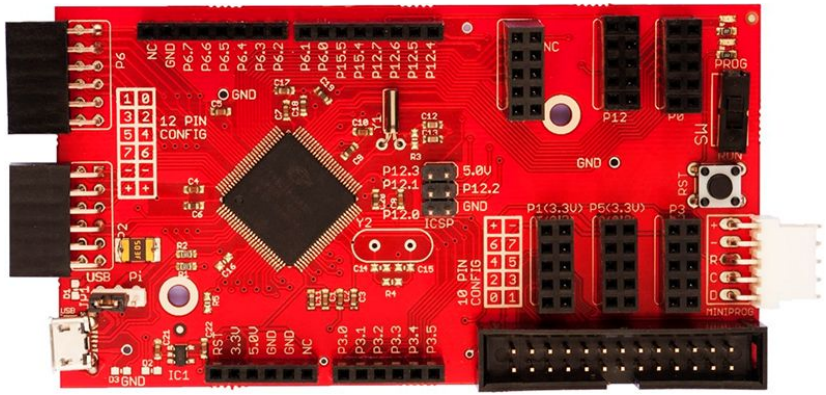
Arcade Machine

PiSoC - Face Tracker

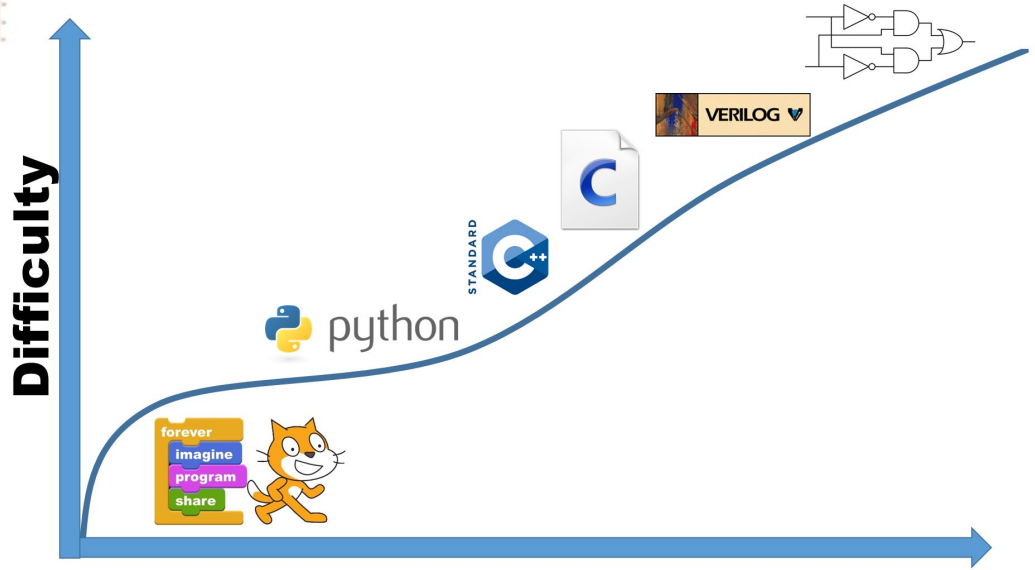
Home Automation



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



Arduino

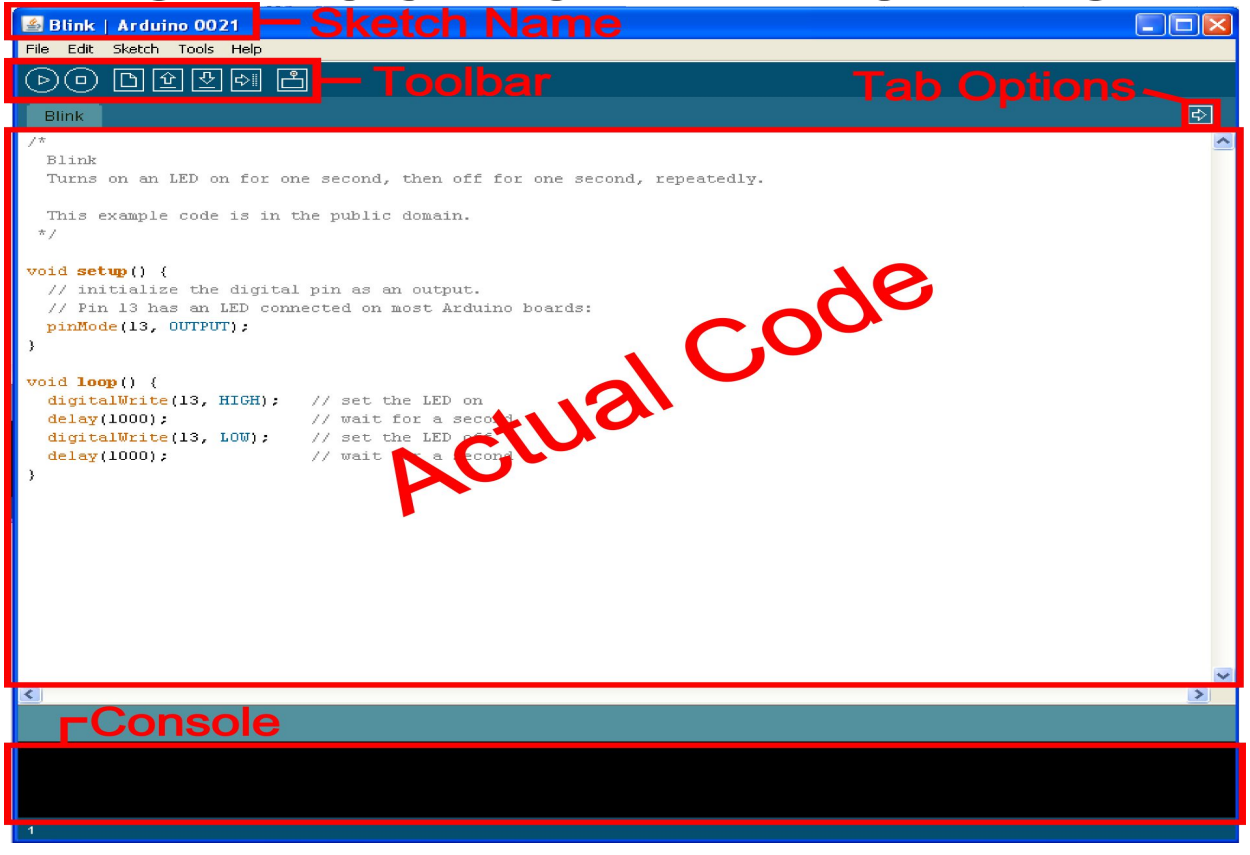

TM
ARDUINO

What?

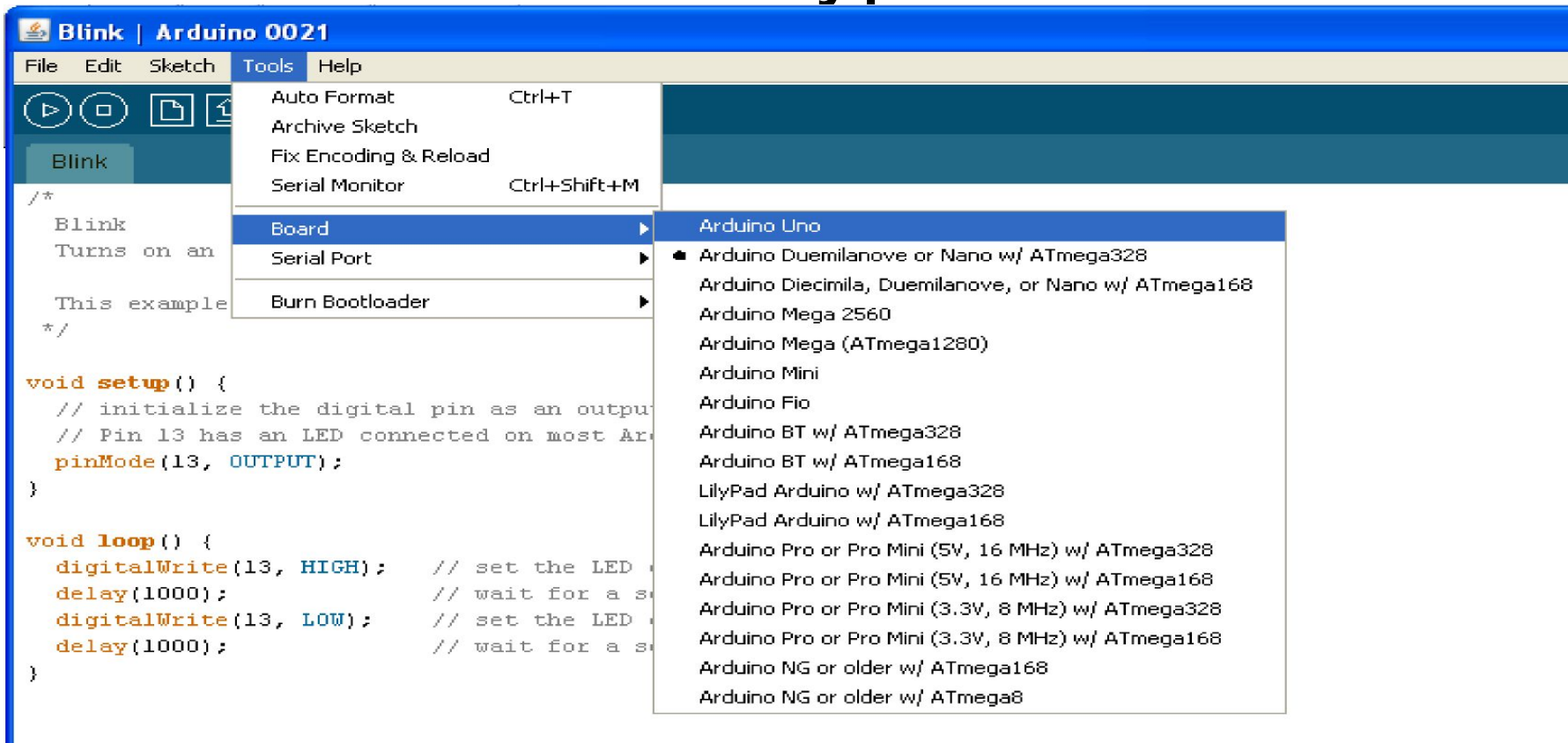
- Physical computing platform
- Open source
- “Hardware Abstracted” Wiring Language
- USB programmable
- Large community
- Multi platform Win/Mac/Linux
- Inexpensive

- 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

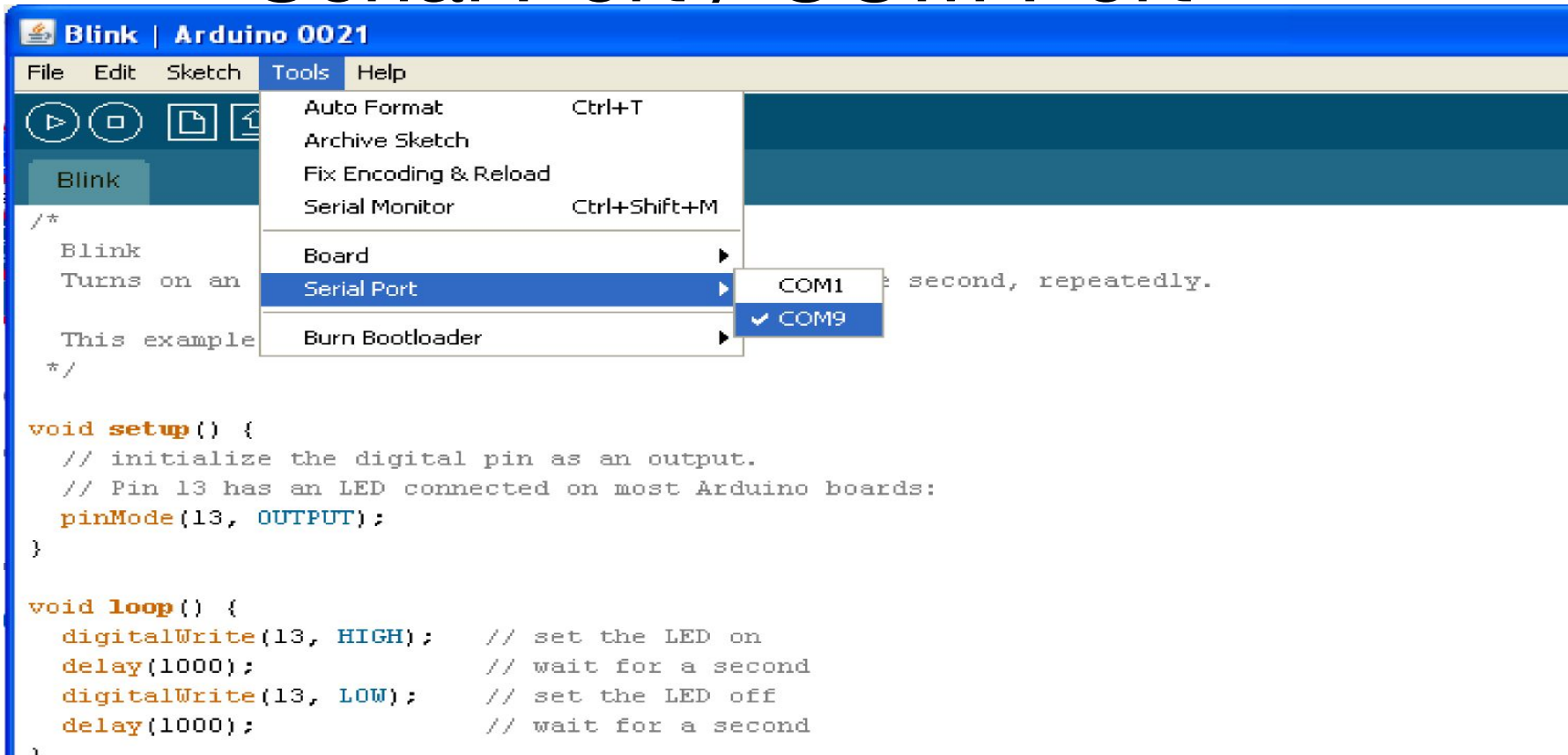
The Arduino Environment



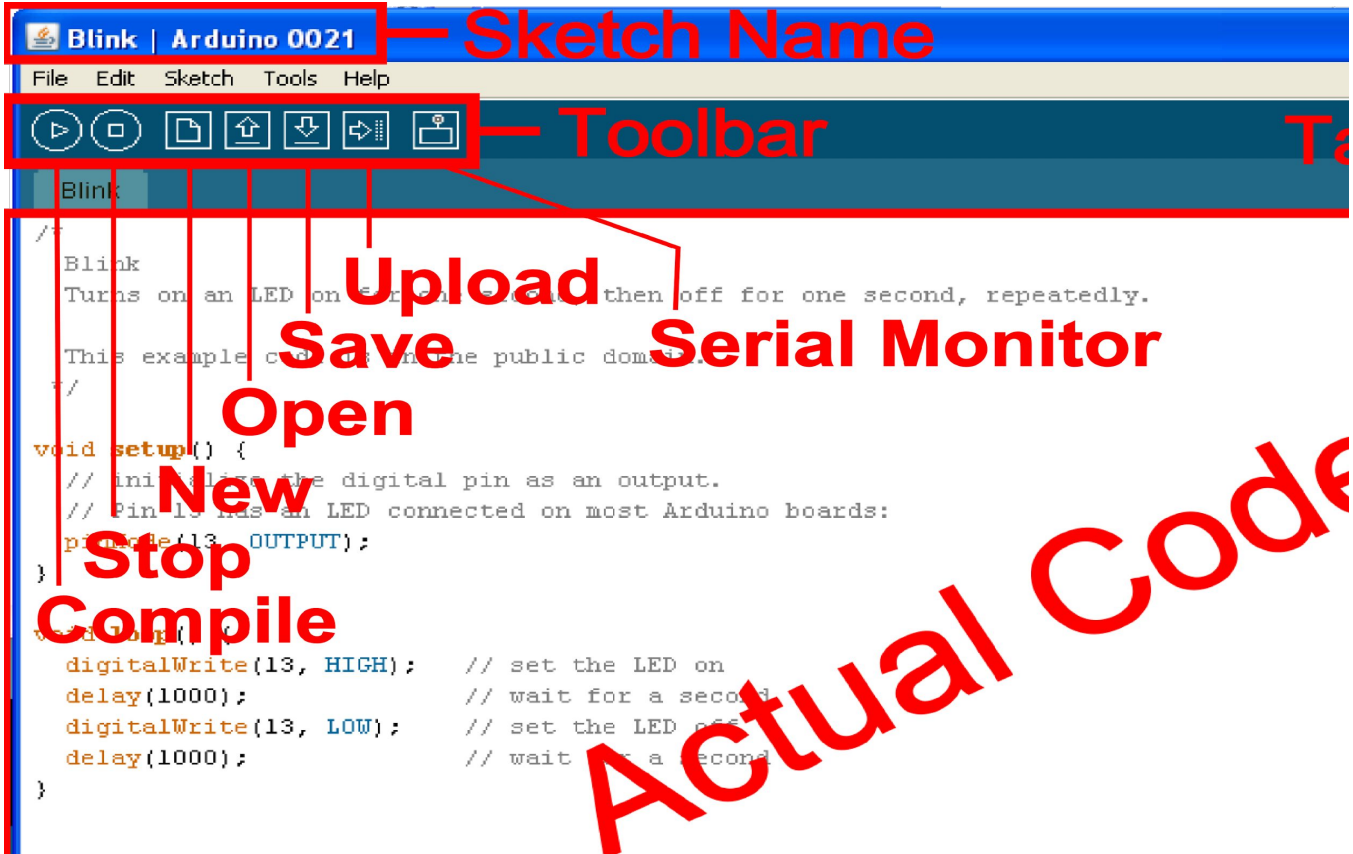
Board Type


Blink | Arduino 0021
File Edit Sketch Tools Help
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
Blink
/*
Blink
Turns on an
This example
*/
void setup() {
// initialize the digital pin as an output
// Pin 13 has an LED connected on most Ar
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH); // set the LED
delay(1000); // wait for a s
digitalWrite(13, LOW); // set the LED
delay(1000); // wait for a s
}

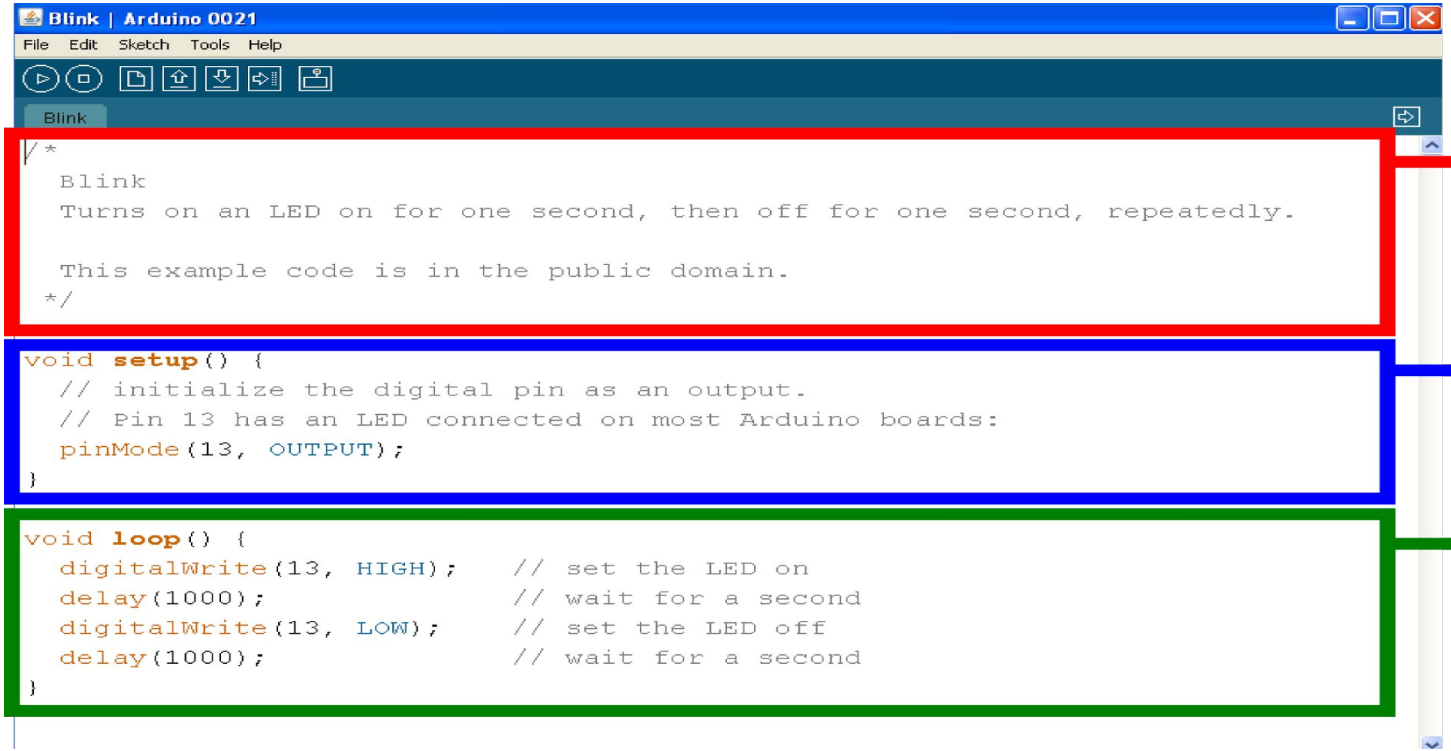
Serial Port / COM Port



The Environment



Parts of the Sketch



Comments / Explaining the game

Setup / Stretching or tying shoes

Loop / Playing the game

Comments

- Comments can be anywhere
- Comments created with // or /* and */
- Comments do not affect code
- You may not need comments, but think about the community!

Operators

The equals sign

= is used to assign a value

== is used to compare values

Operators

And & Or

&& is “and”

|| is “or”

Variables

Basic variable types:

Boolean

Integer

Character

Declaring Variables

Boolean: ***boolean variableName;***

Declaring Variables

Boolean: ***boolean variableName;***

Integer: ***int variableName;***

Declaring Variables

Boolean: ***boolean variableName;***

Integer: ***int variableName;***

Character: ***char variableName;***

Declaring Variables

Boolean: ***boolean variableName;***

Integer: ***int variableName;***

Character: ***char variableName;***

String: ***stringName [ ];***

Assigning Variables

Boolean: ***variableName = true;***

or ***variableName = false;***

Assigning Variables

Boolean: ***variableName = true;***

or ***variableName = false;***

Integer: ***variableName = 32767;***

or ***variableName = -32768;***

Assigning Variables

Boolean: ***variableName = true;***

or ***variableName = false;***

Integer: ***variableName = 32767;***

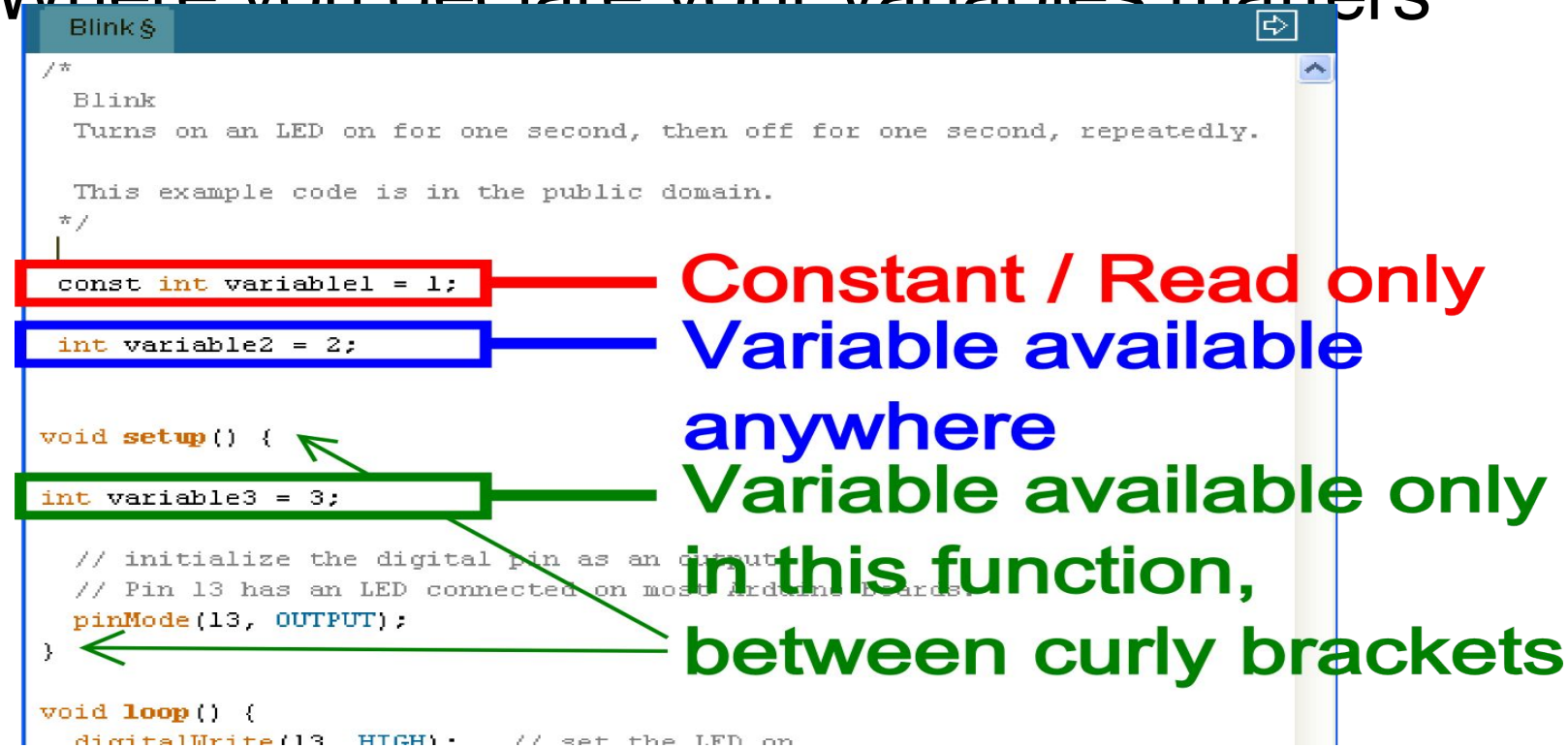
or ***variableName = -32768;***

Character: ***variableName = ‘A’;***

or ***stringName = “SparkFun”;***

Variable Scope

Where you declare your variables matters



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 ( ) { }

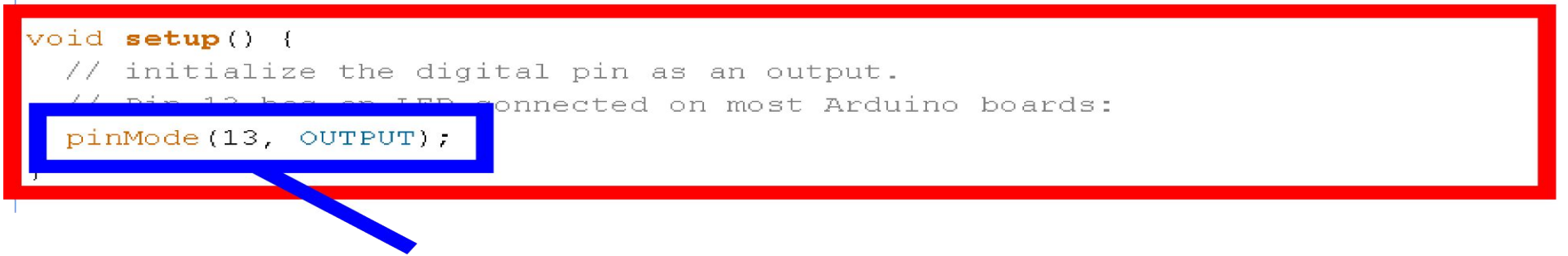
```
void setup() {
  // Initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}
```

The setup header will never change, everything else that occurs in setup happens inside the curly brackets

Setup

void setup ( ) {

pinMode (13, OUTPUT); }



Outputs are declare 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:
  Serial.begin(9600);
}
```

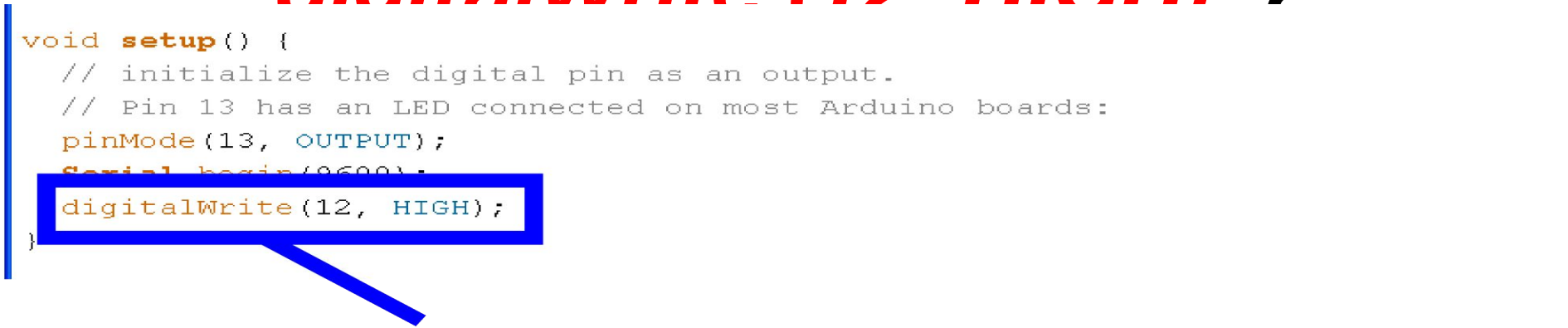
Serial communication also begins in setup

This particular example declares Serial communication

Setup, Internal Pullup Resistors

void setup ( ) {

digitalWrite (12, HIGH); }



You can also create internal pullup resistors in setup, to do so digitalWrite the pin HIGH

This takes the place of the pullup resistors currently on

Setup, Interrupts

void setup ( ) {

attachInterrupt (interrupt, function, mode) }

You can designate an interrupt function to Arduino pins # 2 and 3

This is a way around the linear processing of Arduino

Setup, Interrupts

void setup ( ) {

attachInterrupt (interrupt, function, mode) }

Interrupt: the number of the interrupt, 0 or 1, corresponding to Arduino pins # 2 and 3 respectively

Function: the function to call when the interrupt occurs

Mode: defines when the interrupt should

Setup, Interrupts

void setup ( ) {

attachInterrupt (interrupt, function, mode) }

- ***LOW*** whenever pin state is low
- ***CHANGE*** whenever pin changes value
- ***RISING*** whenever pin goes from low to high
- ***FALLING*** whenever pin goes from low to high

If Statements

if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

If Statement

If

if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

IF

Conditional

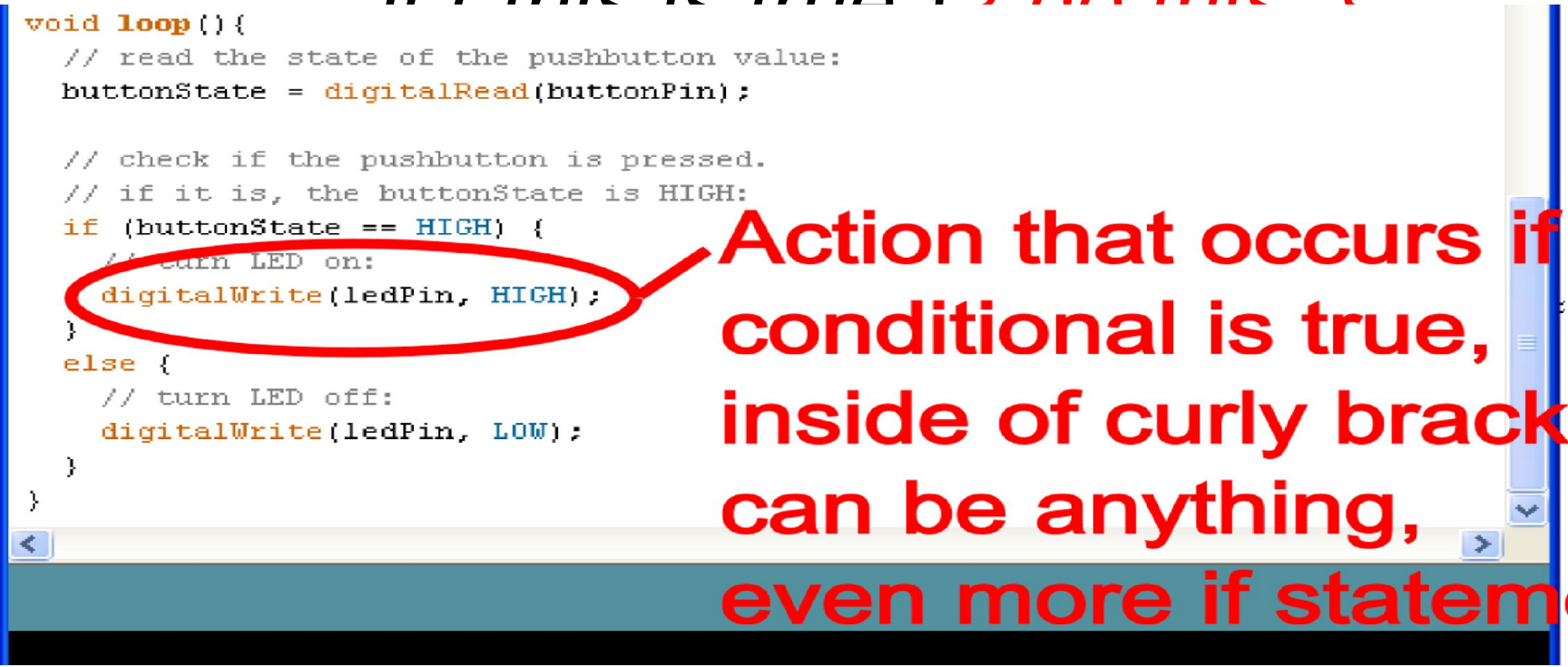
if (this is true) { do this; }

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Conditional inside parenthesis, uses ==, <=, >= or ! you can also nest using && or ||

Action

if (this is true) { do this; }



Else

else { do this; }

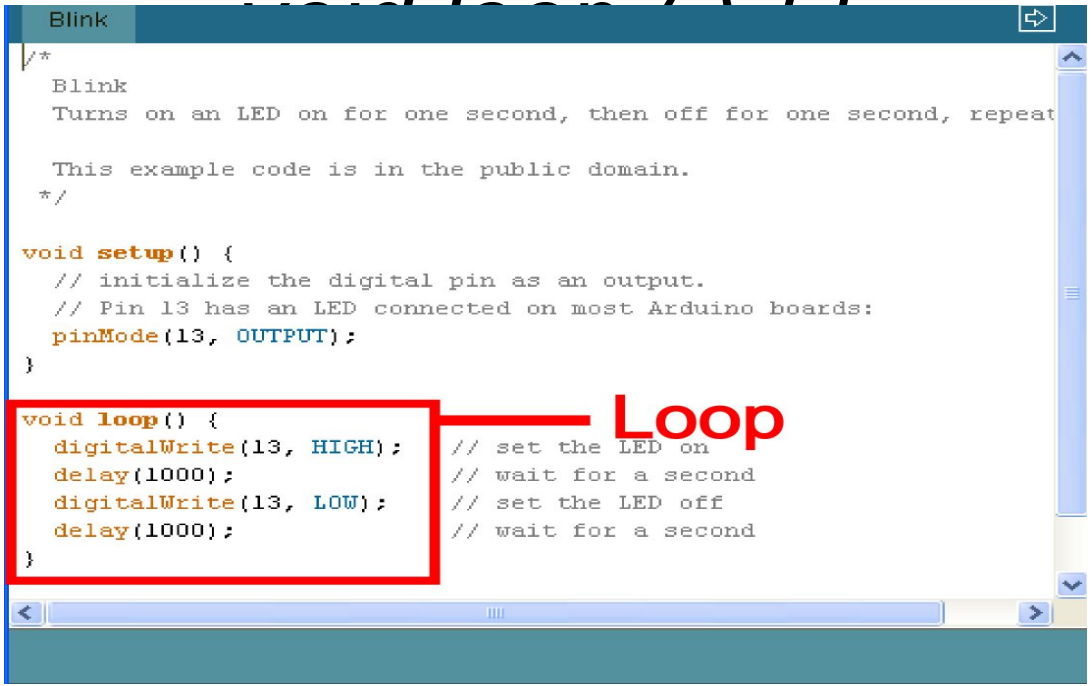
```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Else, optional

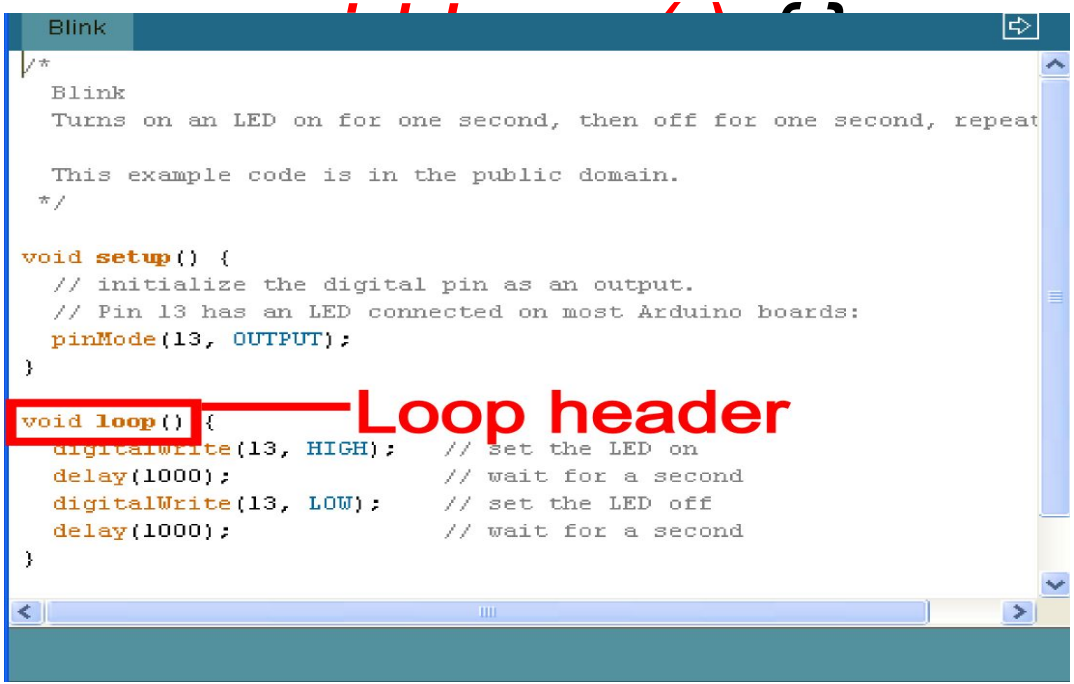
Basic Repetition

- loop
- For
- while

Basic Repetition



Basic Repetition



Basic Repetition

void loop ( ) { }

The “void” in the header is what the function will return (or spit out) when it happens, in this case it returns nothing so it is void

Basic Repetition

void loop ( ) { }

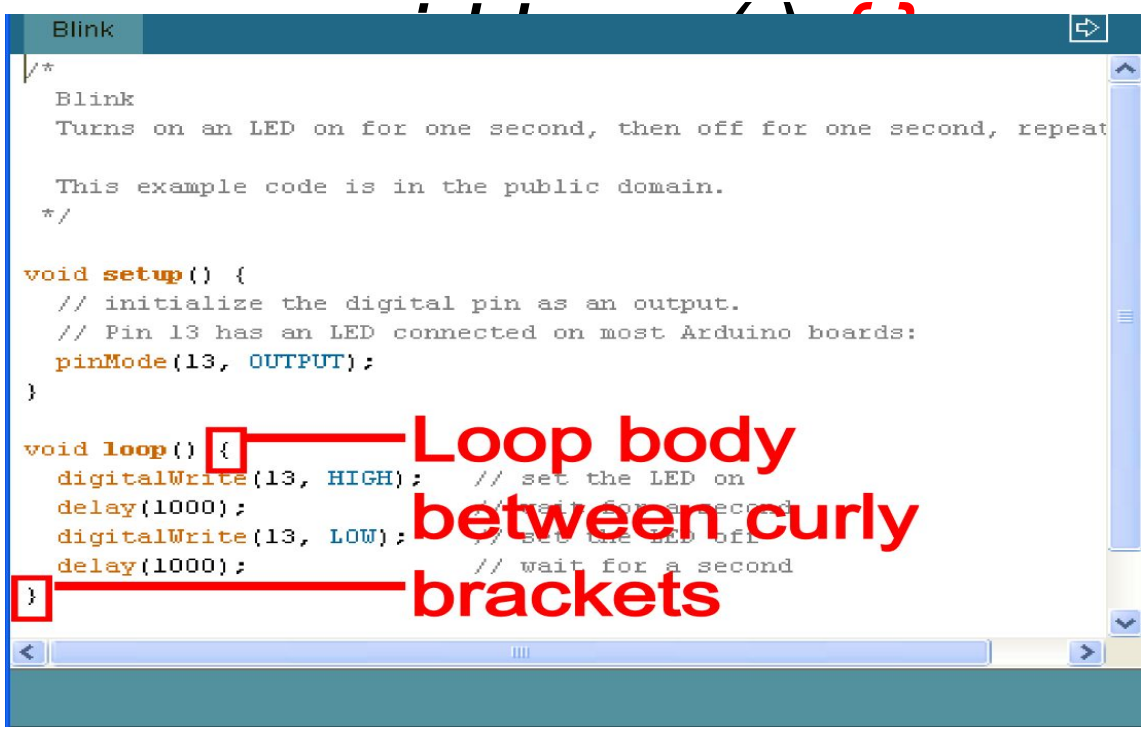
The “loop” in the header is what the function is called, sometimes you make the name up, sometimes (like loop) the function already has a name

Basic Repetition

void loop ( ) { }

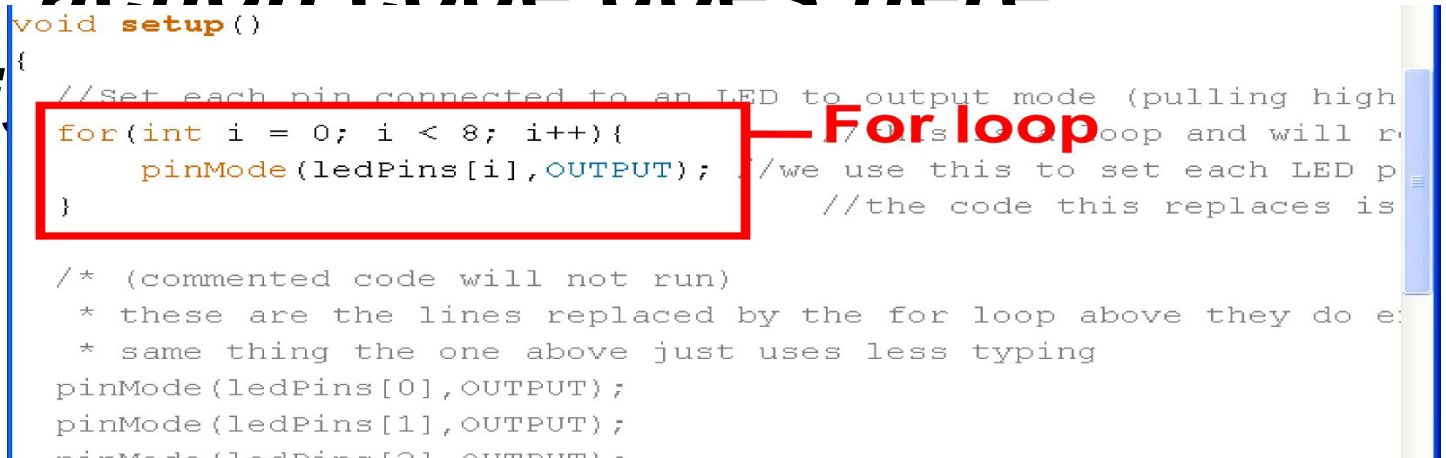
The “( )” in the header is where you declare any variables that you are “passing” (or sending) the function, the loop function is never “passed” any variables

Basic Repetition



Basic Repetition

for (int count = 0; count<10; count++)
{
//for action code goes here
//thi
}



Basic Repetition

for (int count = 0; count<10; count++)

{

//for

}

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                               //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

For header

Basic Repetition

for (int count = 0; count<10; count++)
{
//for
}

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 0; i++){                    //this is a loop and will r
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }                                              //the code this replaces is

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

For

Basic Repetition

for (int count = 0; count<10; count++)
{
//for
}

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }

  /* (commented code will not run)
   * these are the lines replaced by the for loop above they do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Declare a variable and assign it a value

Basic Repetition

for (int count = 0; count<10; count++)
{
//fo
}

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++){
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }

  /* (commented code will not run)
   * these are the lines replaced by the for loop above the do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

If this conditional is true do the code inside the curly brackets, if it's false the computer exits the for loop

Basic Repetition

for (int count = 0; count<10; count++)
{
//for
}

```
void setup()
{
  //Set each pin connected to an LED to output mode (pulling high
  for(int i = 0; i < 8; i++) {
      pinMode(ledPins[i],OUTPUT); //we use this to set each LED p
  }

  /* (commented code will not run)
   * these are the lines replaced by the for loop above. do e
   * same thing the one above just uses less typing
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
```

Change variable so the computer isn't stuck inside for loop forever

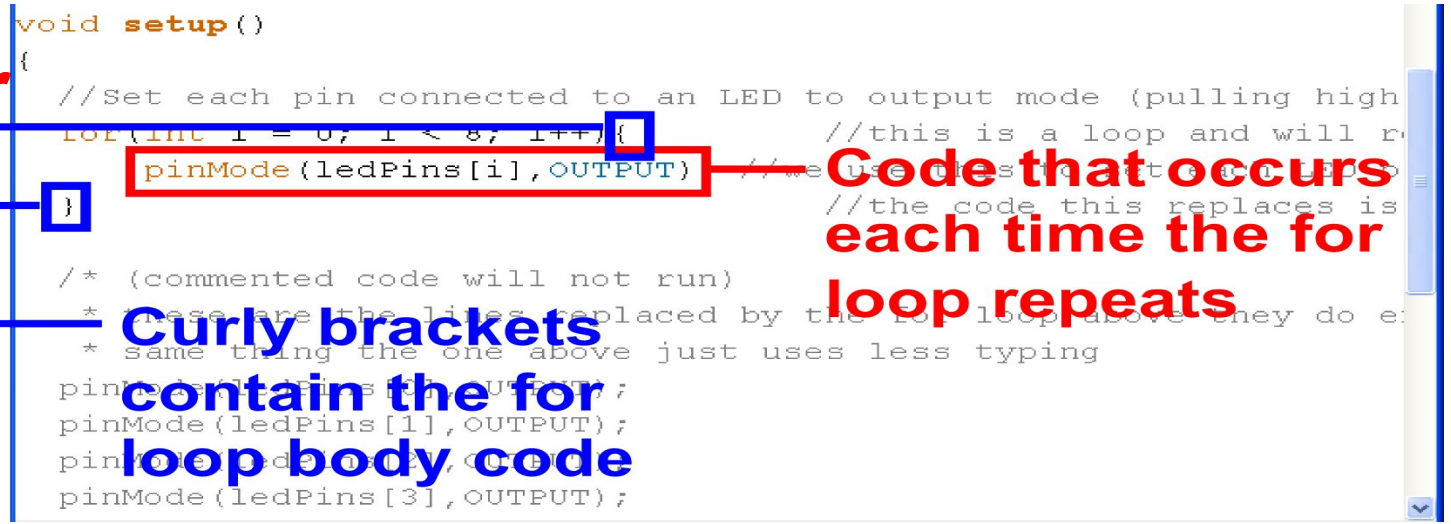
Basic Repetition

for (int count = 0; count<10; count++)

{

//for

}



Basic Repetition

```
while ( count<10 )
{
//while action code goes here
}
```

Basic Repetition

```
while ( count<10 )
{
//while action code goes here
//should include a way to change count
//variable so the computer is not stuck
//inside the while loop forever
}
```

Basic Repetition

```
while ( count<10 )
{
//looks basically like a “for” loop
//except the variable is declared before
//and incremented inside the while
//loop
}
```

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
```