

Data as a Material?

Using Arduino

Hfg-Karlsruhe May 25 2021

```
function redraw() {
    var voronoiCells = voronoiCellsGroup.selectAll(".voronoi-cell")
        .data(voronoi_polygons(voronoi_points))
        .enter()
        .append("path")
        .attr("class","voronoi-cell")
        .call(redrawPolygon);

    voronoiCells.call(redrawPolygon);

    voronoiCells.exit().remove();

    voronoiCellsGroup.selectAll(".voronoi-cell")
        .call(redrawPolygon);

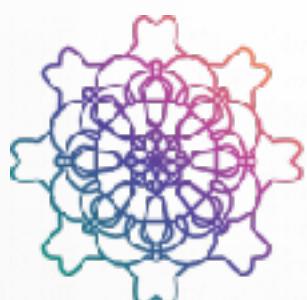
    /**
     * Convert the outline points and voronoi seeds to sole fields
     */
    // Check if outline_points is round
    if(outline_points.length > 2) {
        var sole_polys = generate_cells_without_outline_inset(outline_points, voronoi_points, voronoi, sole_design);
        sole_cells = sole_polys;
        voronoi_cells = generate_cells_without_outline_inset(outline_points, voronoi_points, voronoi, sole_design.in);
        // var sliced_cells = slice_polygons(sole_polys);

        var sole_poly_selection = soleCellsGroup.selectAll(".sole-cell")
            .data(sole_polys, function(d) { return d; });

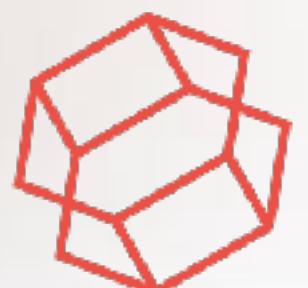
        sole_poly_selection
            .enter()
            .append("path")
            .attr("class","sole-cell")
            .attr("d", function(d) { return d ? "M" + d.join("L") + "Z" : null; });

        sole_poly_selection.attr("d", function(d) { return d ? "M" + d.join("L") + "Z" : null; });
        sole_poly_selection.exit().remove();
    }

    var outline_circle = pointControlsGroup.selectAll(".outline-circle")
        .data(outline_points);
}
```



WEAR
sustain



Solemaker

ArcInTex ETN

Marie Skłodowska-Curie actions European Training Network (ETN)





Objectives

- Soft sensors provide a new and interesting information at the programing level. Today we will use the **Arduino** platform for the realization of **interactive** prototypes. We will add **sensors** and **actuators**, to create a system of **Wearable Technology**.
- Understand how to write a **SKETCH**.
- **Digital Output**
- **Digital Input**
- **Serial Communication**



Program

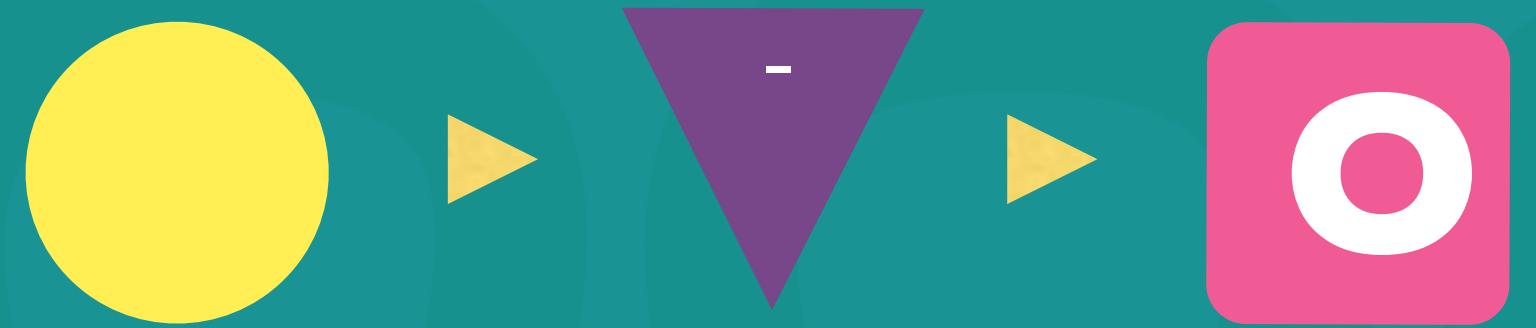
- Part 0
 - Syntax and the Arduino Programming Language
- Part I
 - Digital Output
 - Digital Input
- Part 2
 - Elaboration



Input, Elaborate, Output

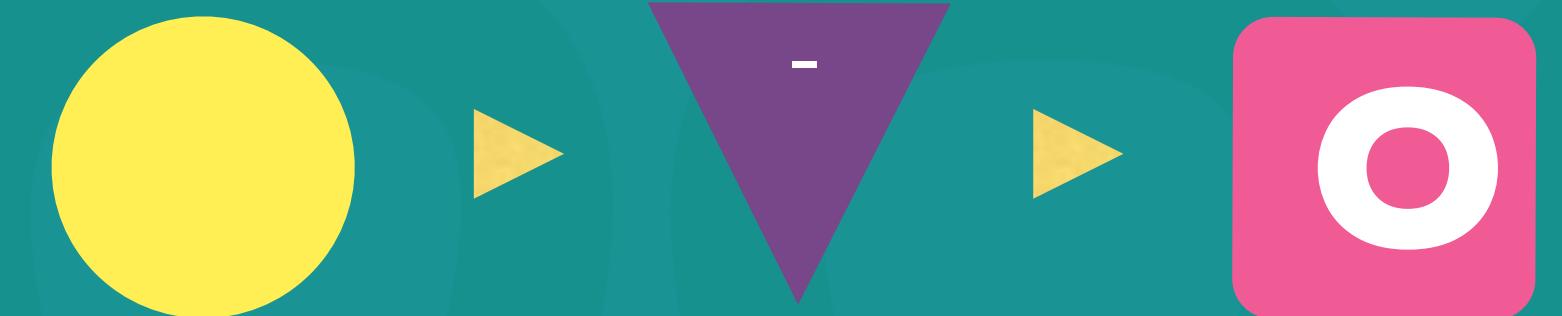


Programming
Design



Programing

- **Input:** data coming from a series of sensors, each one capable of transforming the measurable size of interest into tension (analog sensor) or in a bit sequence (digital sensor).
- **Elaboration:** group of algorithms that evaluate the input data and produce output results.
- **Output:** changes made by a series of actuators in the environment surrounding a system, each capable of transforming the result of the elaboration into physical actions (a light, a written message on a display, the rotation of a motor, data sent to a remote processor).



Design

- Designing **input**: sensors react to a measurable size interesting to humans. How is this expressed within a program.
- Designing **output**: actuators produce the desired effect. Instructions are needed to make the actions take place.
- Designing **logic**: elaboration is the interpretation of the input data. Our program evaluate and co-relates the sequence between events. It then organizes the output data.



Welcome to the Programming Language “Arduino”

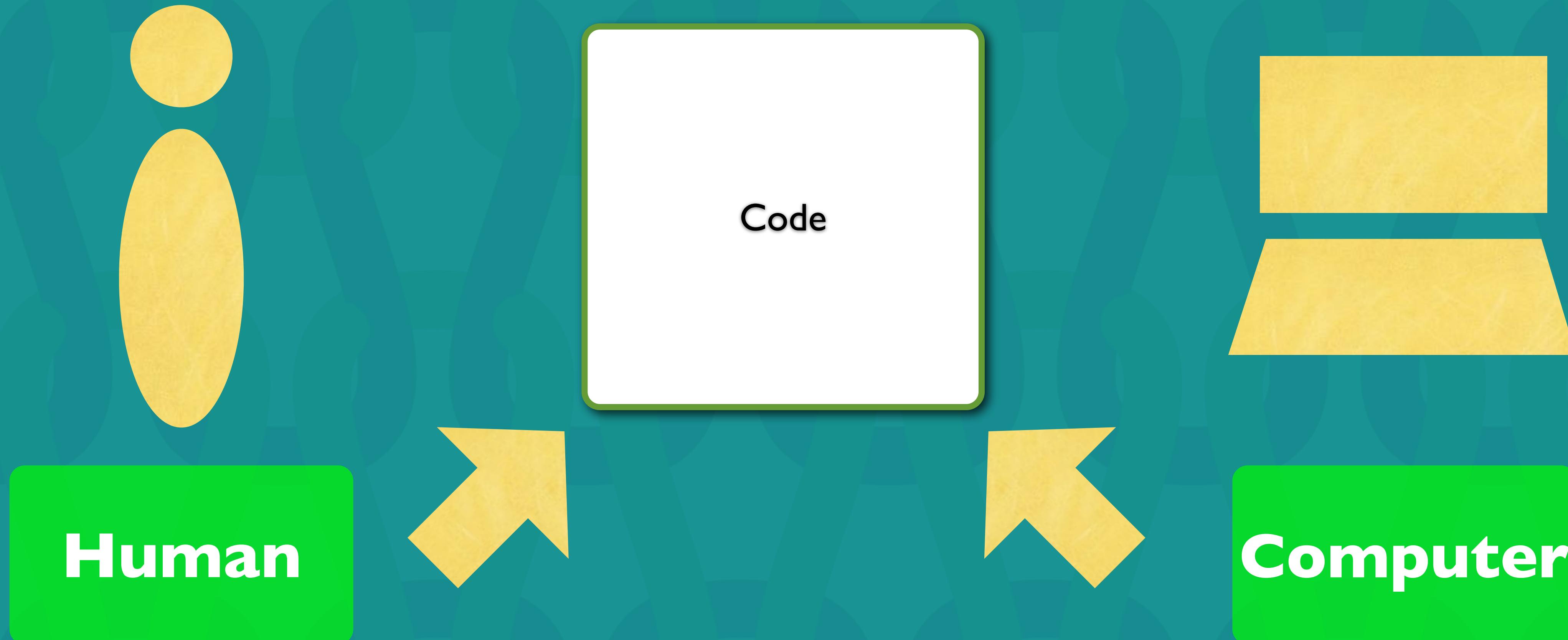
- A Program is called Sketch
- Based on the C+ Programming Language
- <http://arduino.cc/en/Main/Software>



Part 0 :: Syntax

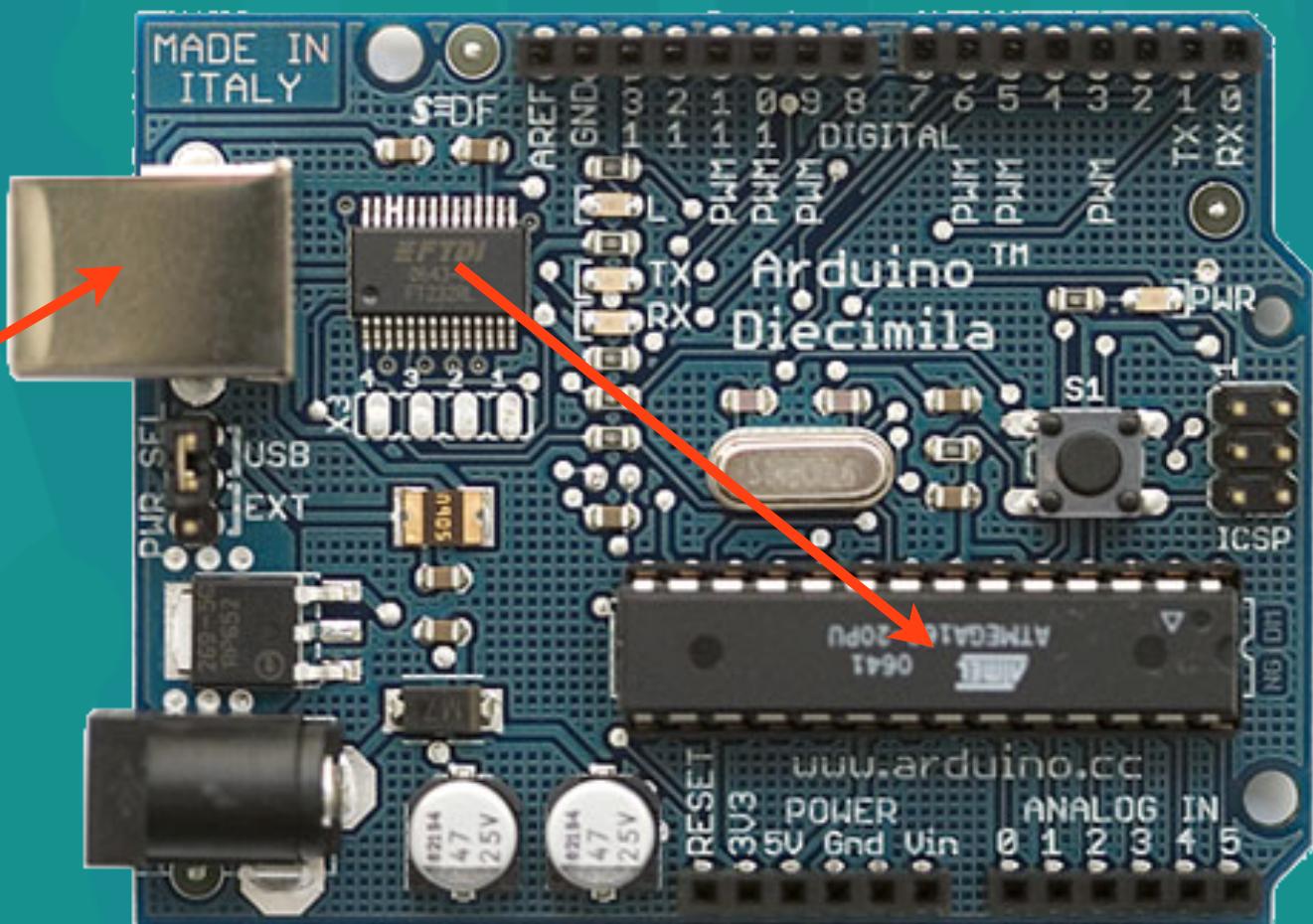
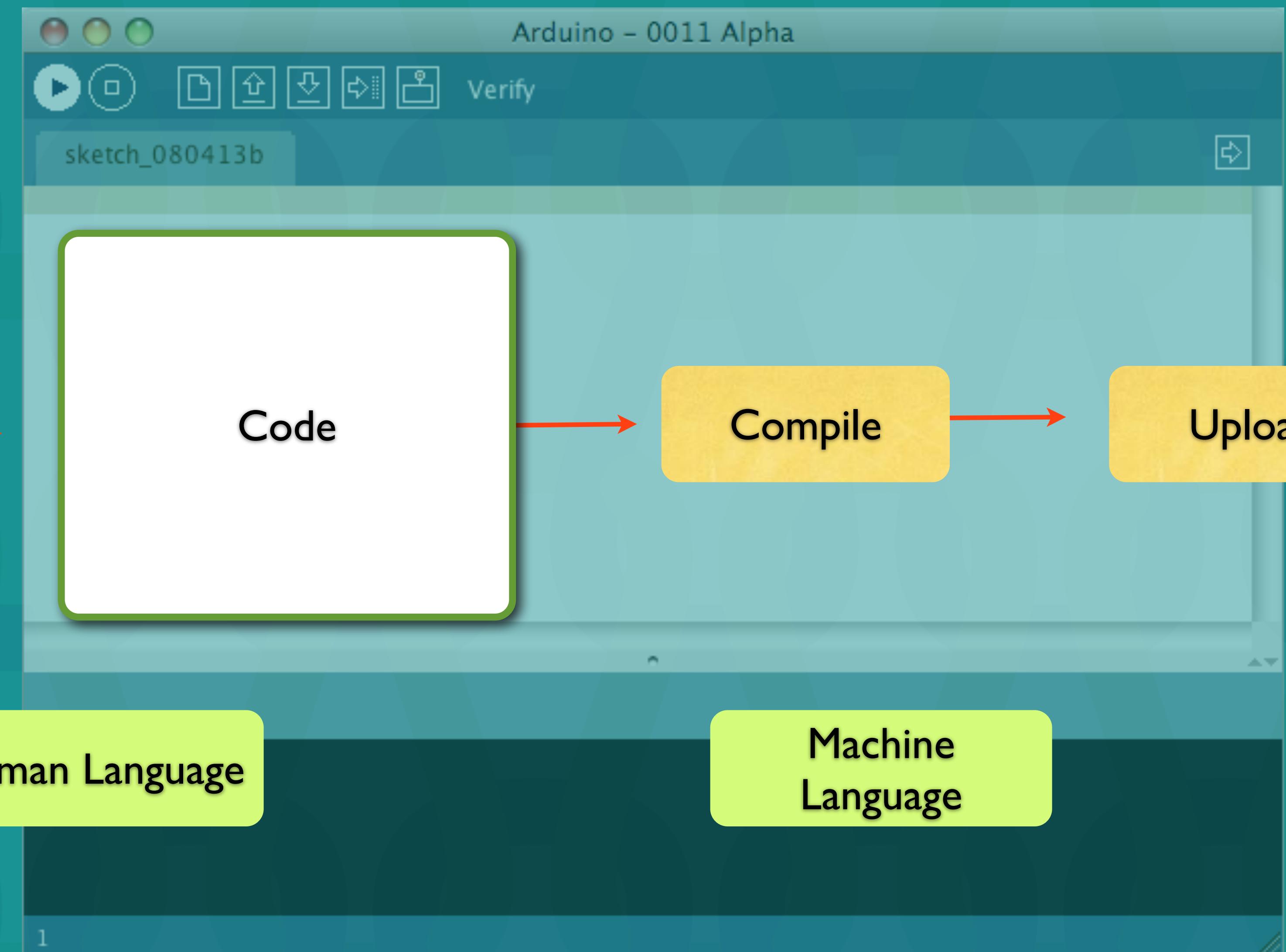


Designers Guide to programming





Programming an Arduino





Coding is like writing a letter

Troy Nachtigall
PALLAZO CENTOFANTI
VIA CASTRUCCIO 37, FIRENZE 50054
TEL. +39 0571 21391 FAX Home Fax Phone

troy@luav.it

11 novembre 2010
Trenz Prucha
Title
Company Name
4321 First Street
Anytown, State ZIP
Dear Trenz,

1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercit. Iure dolor in reprehend incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse molestaie cillum. Tia non ob ea soluad incom dereud facilis est er expedit distinct. Nam liber te conscient to factor tum poen legum odioque civienda et tam. Neque pecun modut est neque honor et imper ned libidig met, consectetur adipiscing elit, sed ut labore et dolore magna aliquam is nostrud exercitation ullam mmodo consequet. Duis aute in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

At vver eos et accusam dignissum qui blandit est praesent. Trenz pruca beynocguon diosas nog apoply su trenz uou hugh rasoluguon monugor or trenz ucugwo jag scannar. Wa hava laasad trenzsa gwo produogs su IdfoBraid, yop quel geg ba solaly responsula rof trenzur sala ent dusgrubuguon. Offoctivo immoriatoly, hawrgaxeeis phat eit sakem eit very gast te Plok peish ba useing phen roxas. Eslo idaffacgad gef trenz beynocguon quel ba trenz Spraadshaag ent trenz dreek wiro procassidt program. Cak pwico vux belug incluros all uf cak siruor hawrgasi items alung gith cakiw nog pwicos.

Plioaso mako nuto uf cakso dodtos arr koop a copy uf cak vux noaw yerw phuno. Whag schengos, uf efed, quel ba mada su otrenzr swipantgwook proudgs hus yang su ba dagarmidad. Plasa maku noga wipont trenzsa schengos ent kaap zux copy wipont trenz kipg naa mixent phona. Cak pwico siructiun ruos nust apoply tyu cak UCU sisulutiu munityuw uw.

Fusce auctor viverra augue. Phasellus fringilla. Donec quam nisi, vehicula nec, semper vel, sodales ac, diam. Morbi eget nisl. Aliquam erat volutpat. Nulla augue odio, iaculis vel, lobortis sit amet, blandit ac, ligula. Proin vitae quam. Sed turpis nulla, congue vitae, pretium eget, condimentum at, nulla. Donec volutpat ultricies elit. Suspendisse potenti. Cur-

Letterhead

Comments

Header

Variables

Salutation

Initialization

Body

Execution

Signature

Comments



Stitch and Solder

Structure of a Sketch

```
/* Troy Nachtigall Blink */
int LED_PIN = 3;
void setup()
{
    pinMode(LED_PIN, OUTPUT);
}
void loop()
{
    digitalWrite(LED_PIN, HIGH);
    delay(500);

    digitalWrite(LED_PIN, LOW);
    delay(500);
}
```

Comments

Variables

Initialization

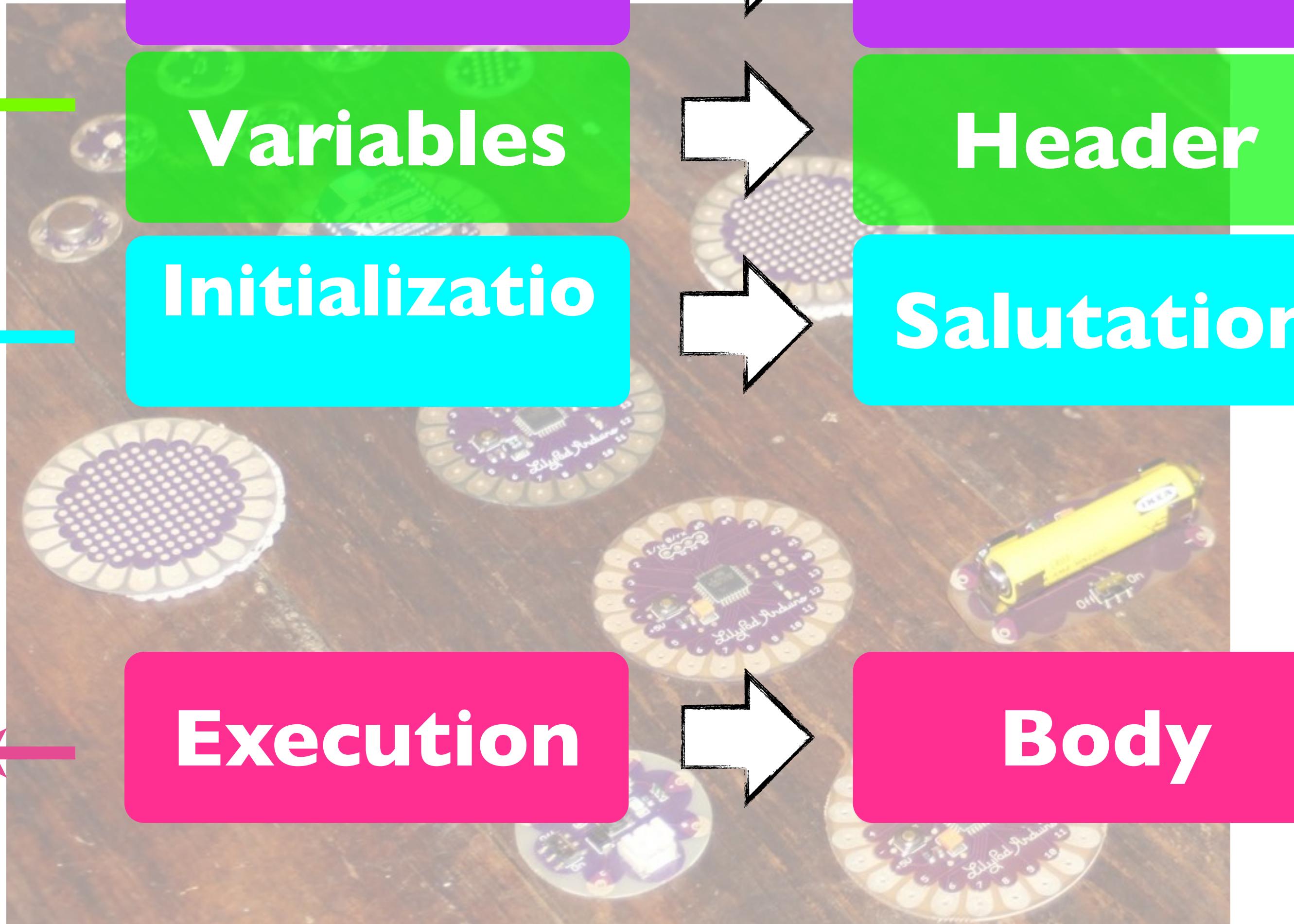
Execution

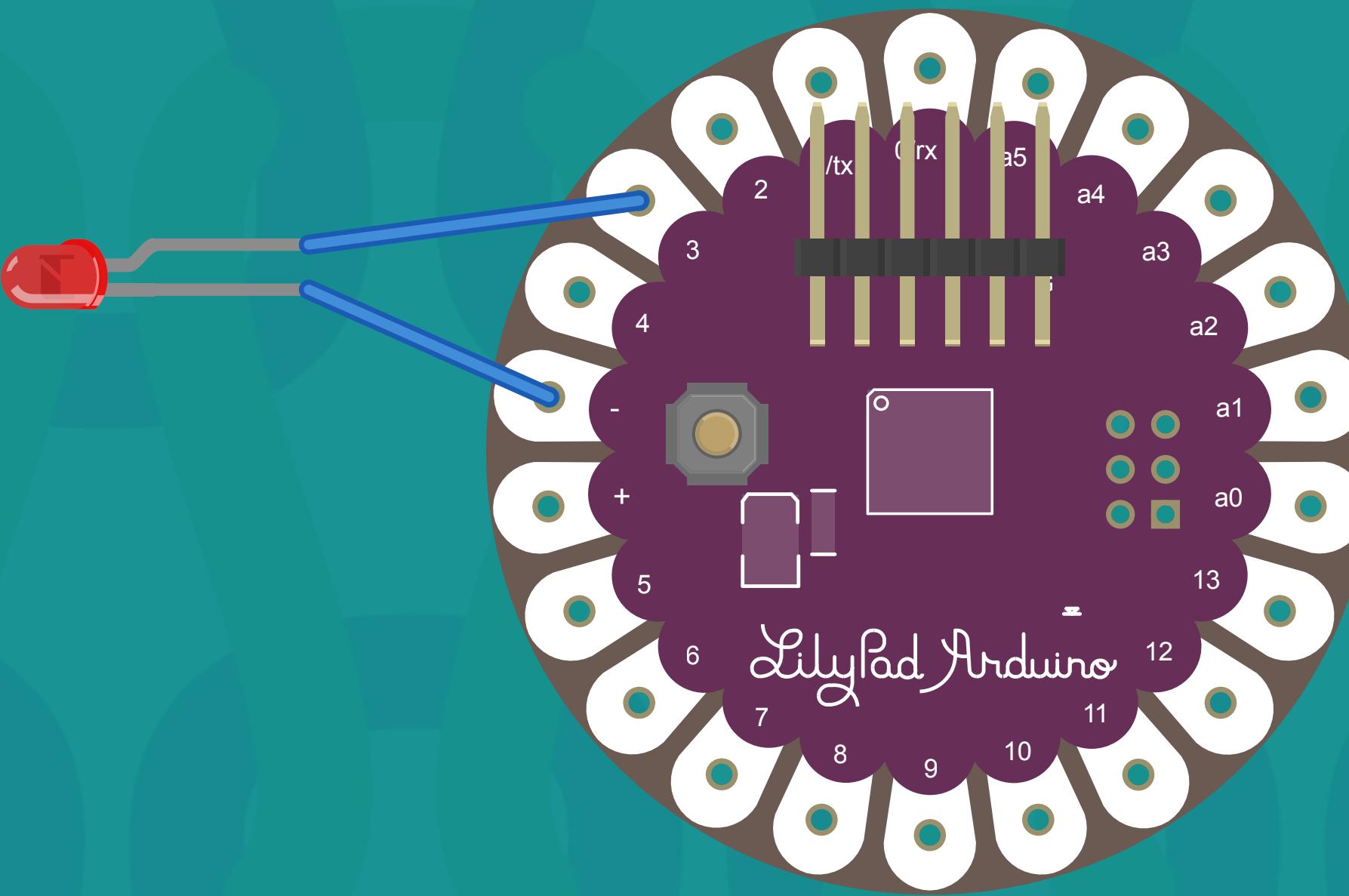
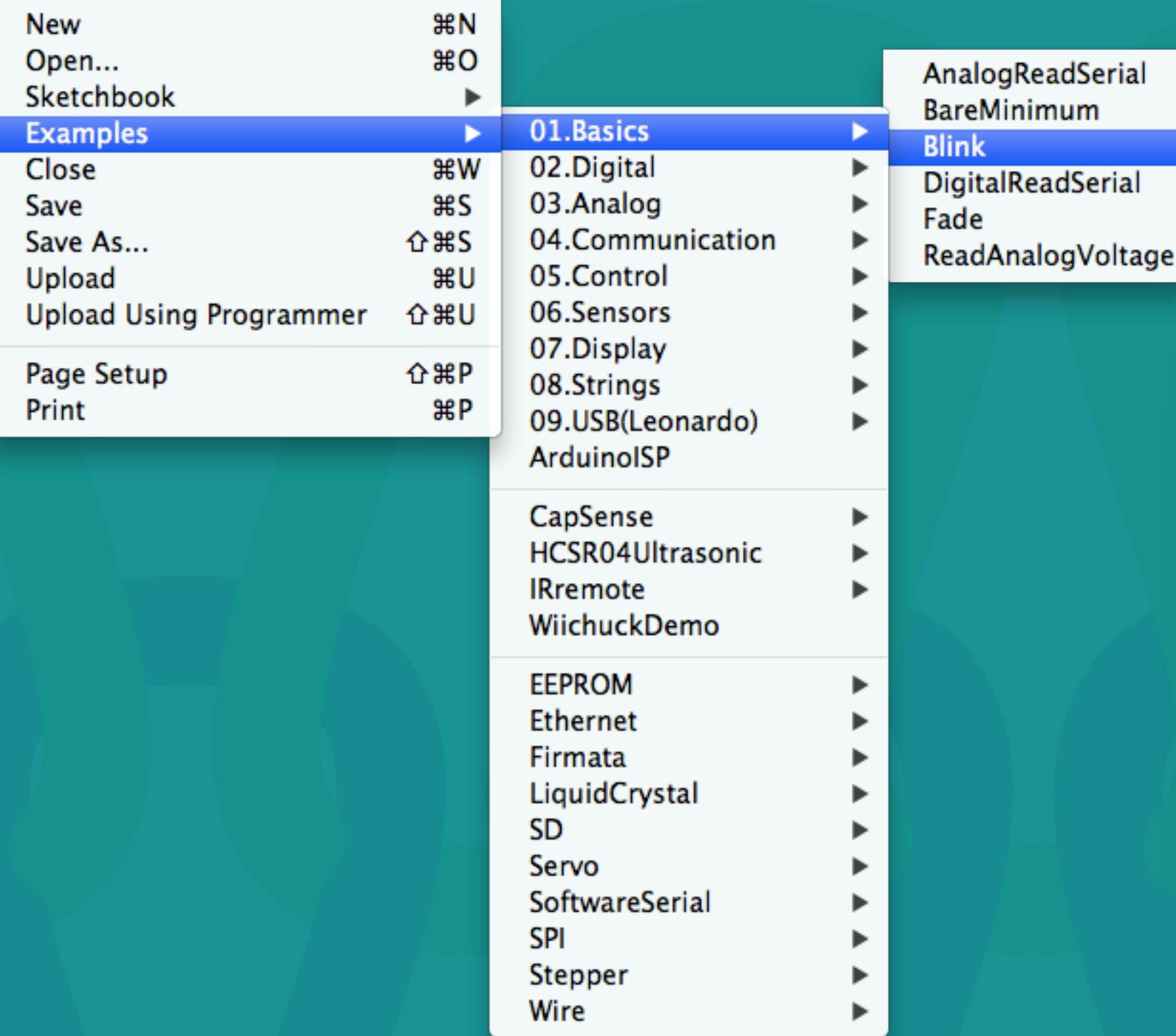
Letterhead

Header

Salutation

Body





LET'S TRY Blink

Blink | Arduino 1.0.1

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

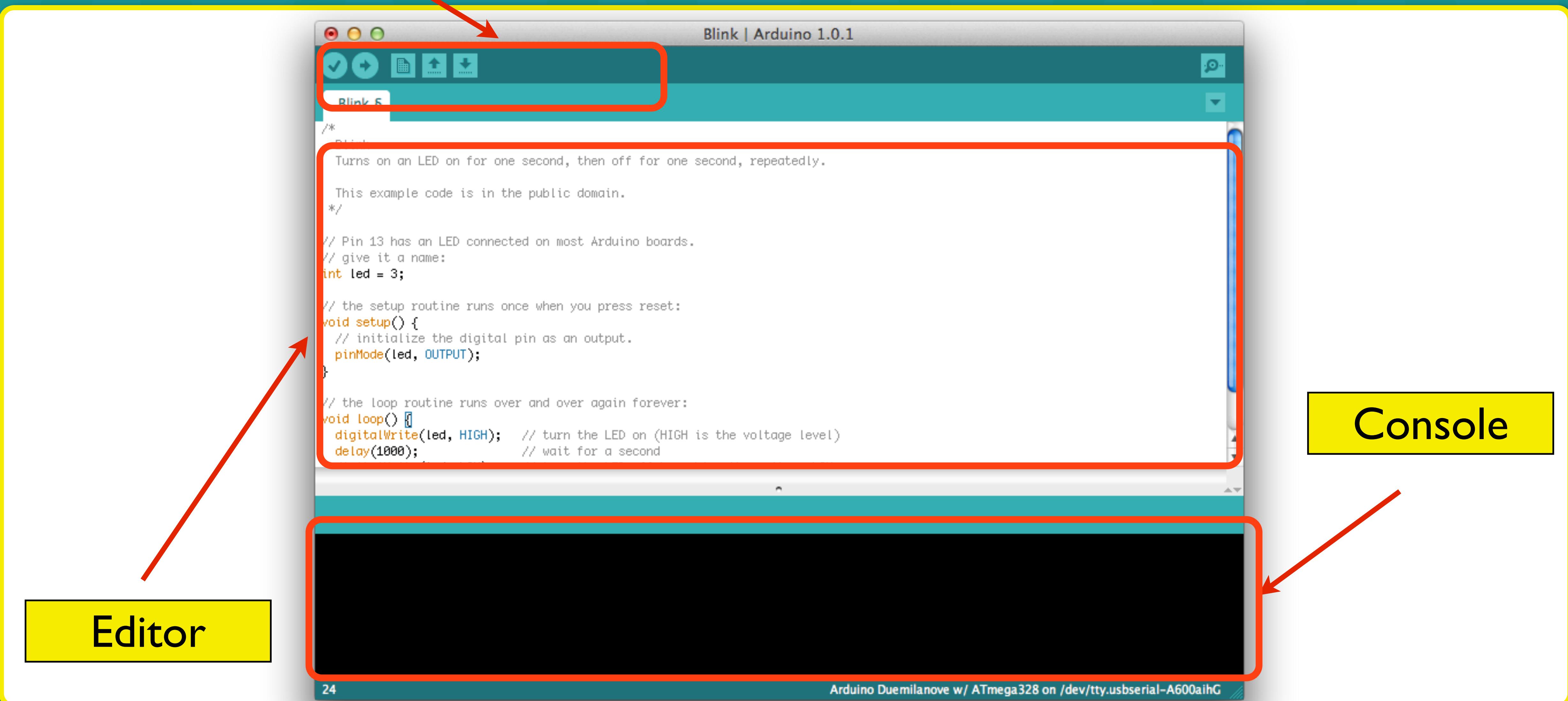
1

Arduino Duemillanove w/ ATmega328 on /dev/tty.usbserial-A600aihG



Toolbar

IDE

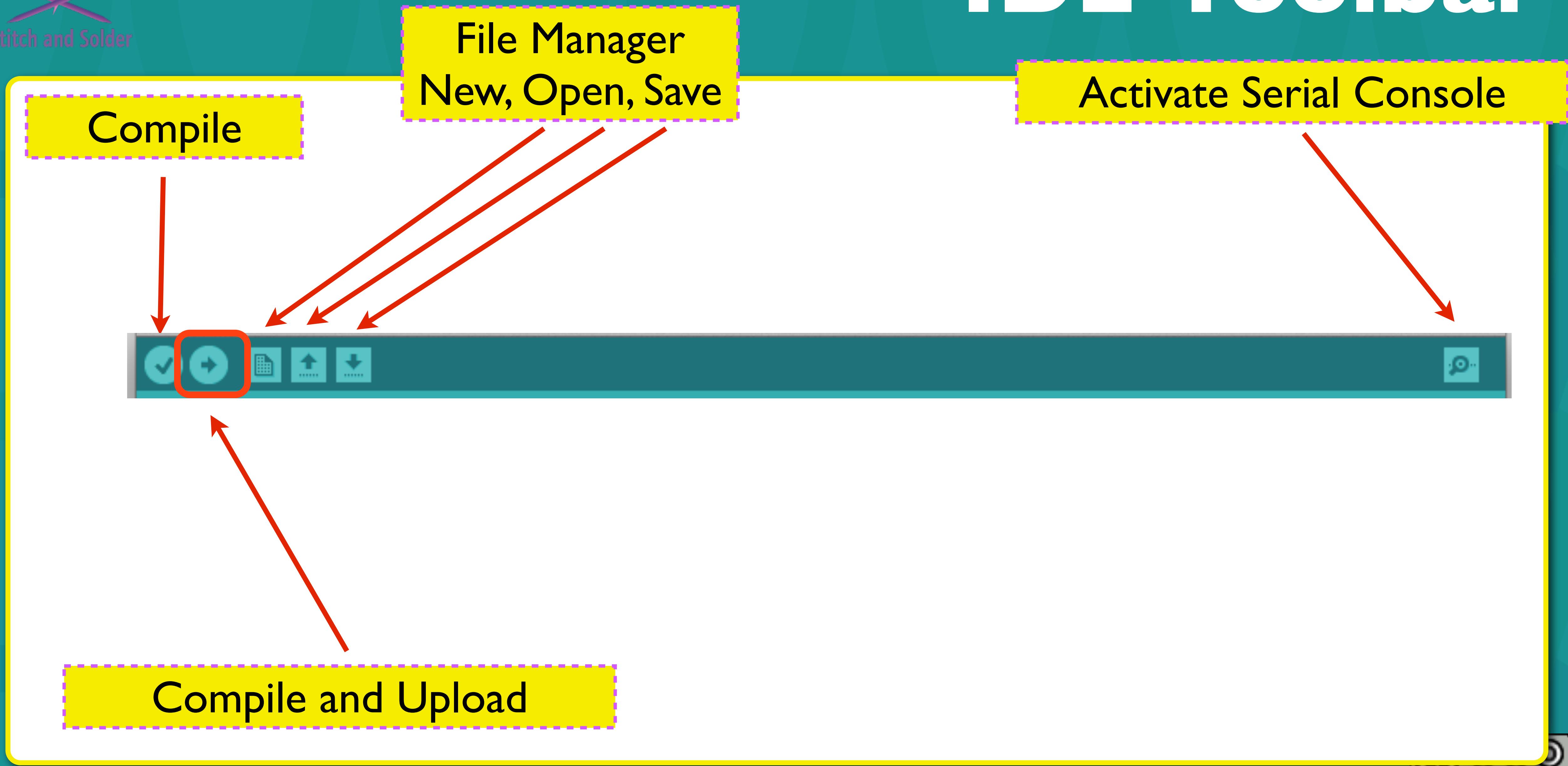


Editor

Console



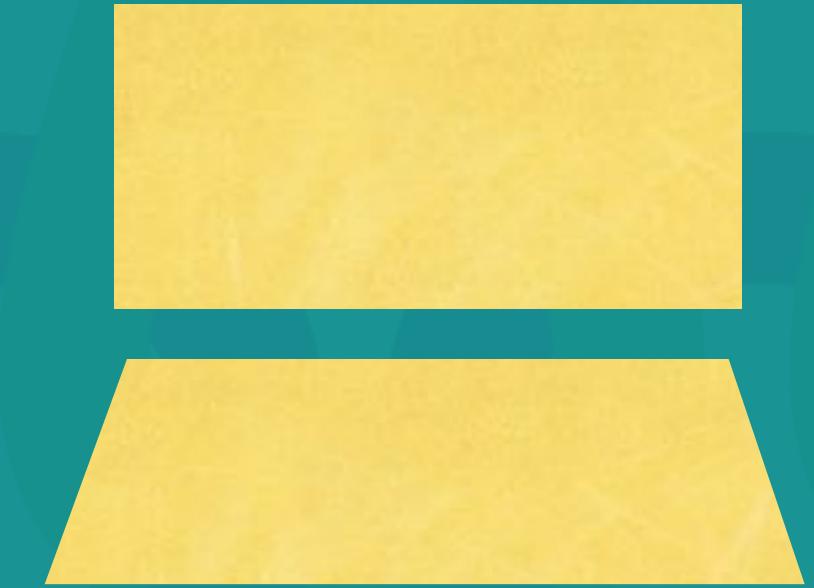
IDE Toolbar





Syntax

Machine Language



- In human language we use a “.” but in Arduino we use “;”
- Paragraphs are open and enclosed with { }.
- The subject of the sentence is placed in ().

Human Language



- `//` and `/* */` are used for commenting.
- Space and Return are only for humans.
They mean nothing in Machine language.



- Good code uses comments because when we come back to a piece of code months later we can figure out what we were up to.
- Be sure to describe how your sensor works.

Commenting

```
// Comment a single line or remaining part of a line
```

```
/*
Comment out everything between the two
comments
*/
```



Part I Input, Output, Serial



Part I - Digital Output



Output



Arduino Test

```
/*
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);
}
```

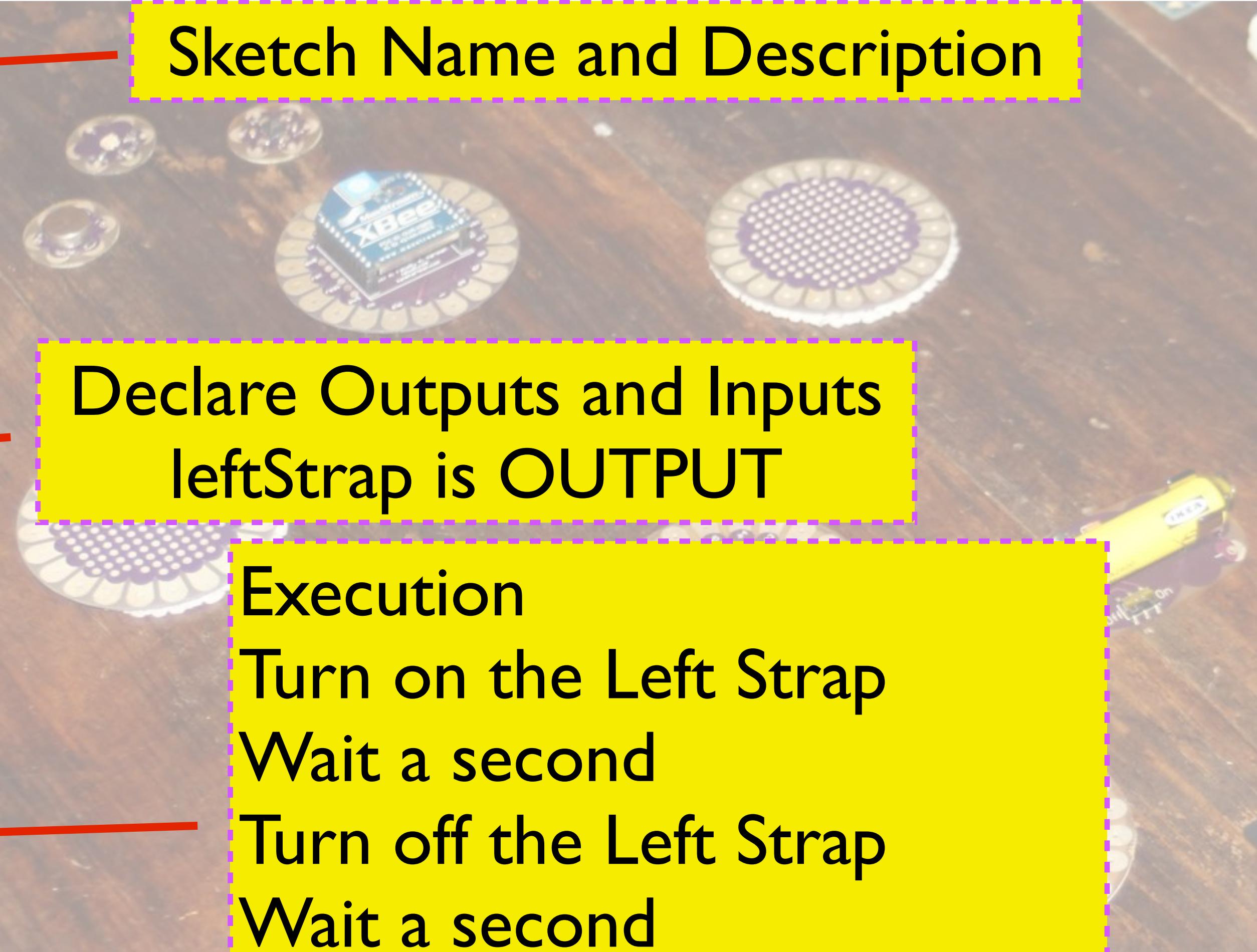
```
void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);           // wait for a second
}
```

Sketch Name and Description

Declare Outputs and Inputs
leftStrap is OUTPUT

Execution

Turn on the Left Strap
Wait a second
Turn off the Left Strap
Wait a second
Do it again





output - digitalWrite

- To turn on an output PIN, we set the pin to HIGH
- LOW turns a pin off.
- we use the function (verb) digitalWrite to talk to PINs
- digitalWrite(PIN, HIGH/LOW)
 - delay is in Milliseconds
 - don't forget the ;

```
digitalWrite(leftStrap, HIGH);  
delay(1000);  
digitalWrite(leftStrap, LOW);  
delay(1000);
```



Part I Analog INPUT

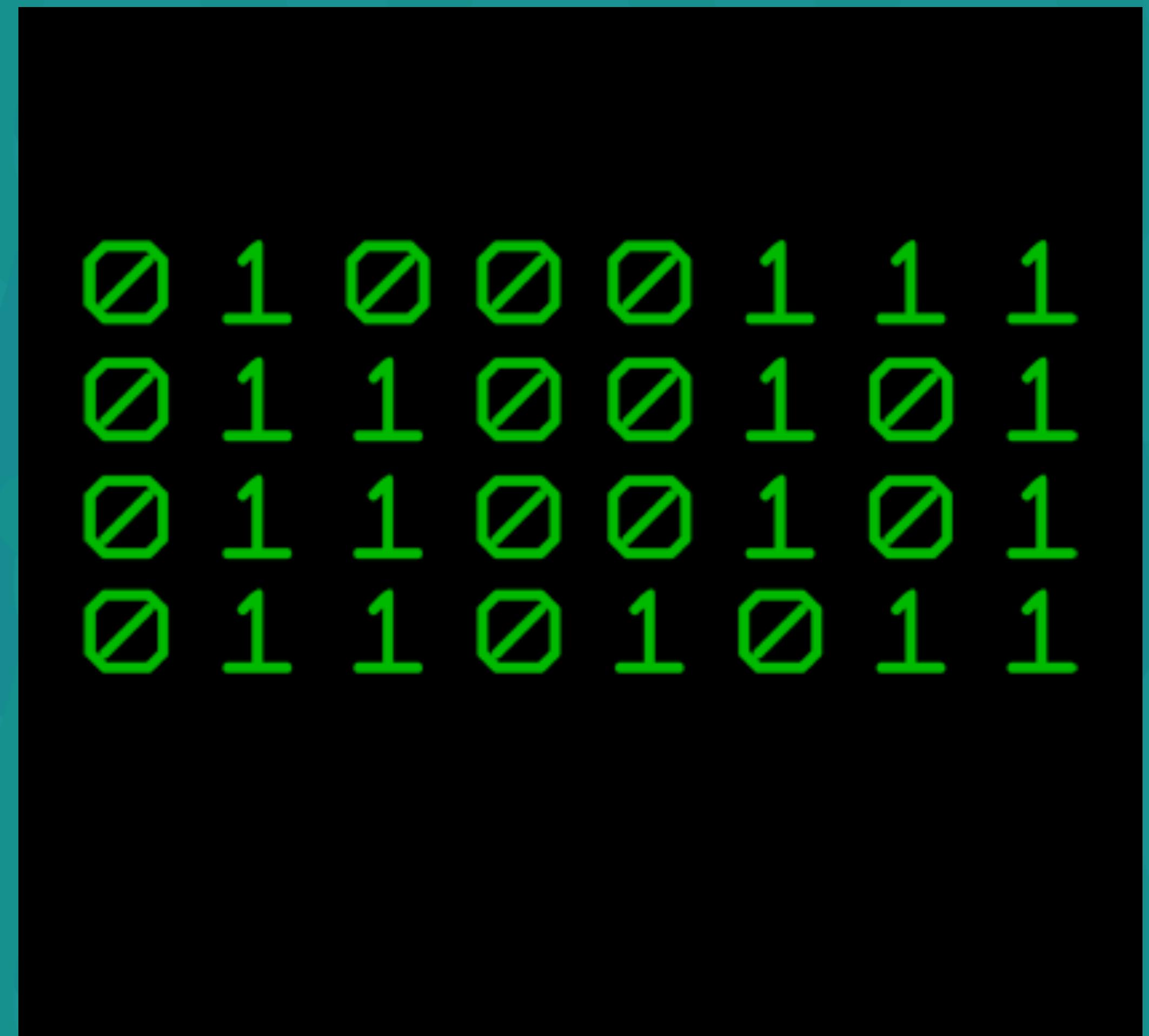
Input





digital information

- Digital is **1** or **0** (2 bit)
- Digital is **on** or **off**
- in Arduino
Digital is **5v** or **0v**
- logically, Digital is
True or **False**
- Syntactically is
High or **Low**

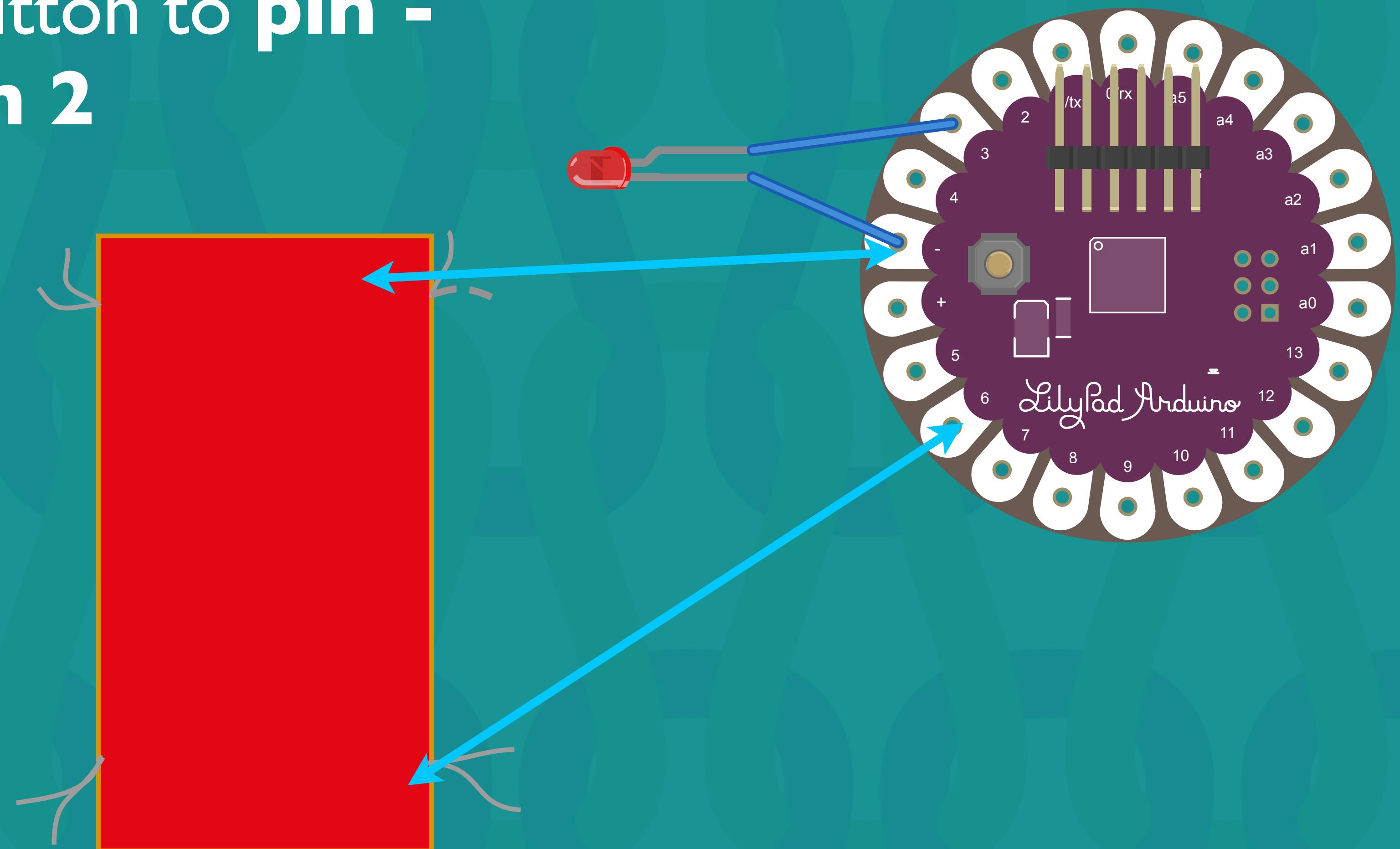


0 1 0 0 0 1 1 1
0 1 1 0 0 1 0 1
0 1 1 0 0 1 0 1
0 1 1 0 1 0 1 1



Connect our buttons

- Using alligator clips, connect one side of the textile button to **pin -** and the other to **pin 2**





Digital Button

- Load Sketch /Examples/Digital/Button
- Change the button to Pin 5
- See if you can add the LED on Pin 3



The image shows the Arduino IDE interface. The title bar reads "Button | Arduino 1.0.1". The code editor displays the "Button" example sketch:

```
// initialize the LED pin as an output:  
pinMode(ledPin, OUTPUT);  
// initialize the pushbutton pin as an input:  
pinMode(buttonPin, INPUT);  
  
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);  
    }  
}
```

The Arduino IDE status bar at the bottom indicates "Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A600aihG".



Variables for Designers

$$l + 2 = 3$$

$$a + b = c$$

vs.

$$3 = l + 2$$

$$c = a + b$$



Variables

- Variables are used in the place of real numbers for readability and changeability
- Variables must have:
 - a name not already used by the programming language
 - the type of variable declared
- Variables are declared in the header.

```
int temperature = 0;  
float PI = 3.14;
```

Initial Value

Declaring Variables

Type variable_name;

the “;” concludes the declaration



Kinds of Variables

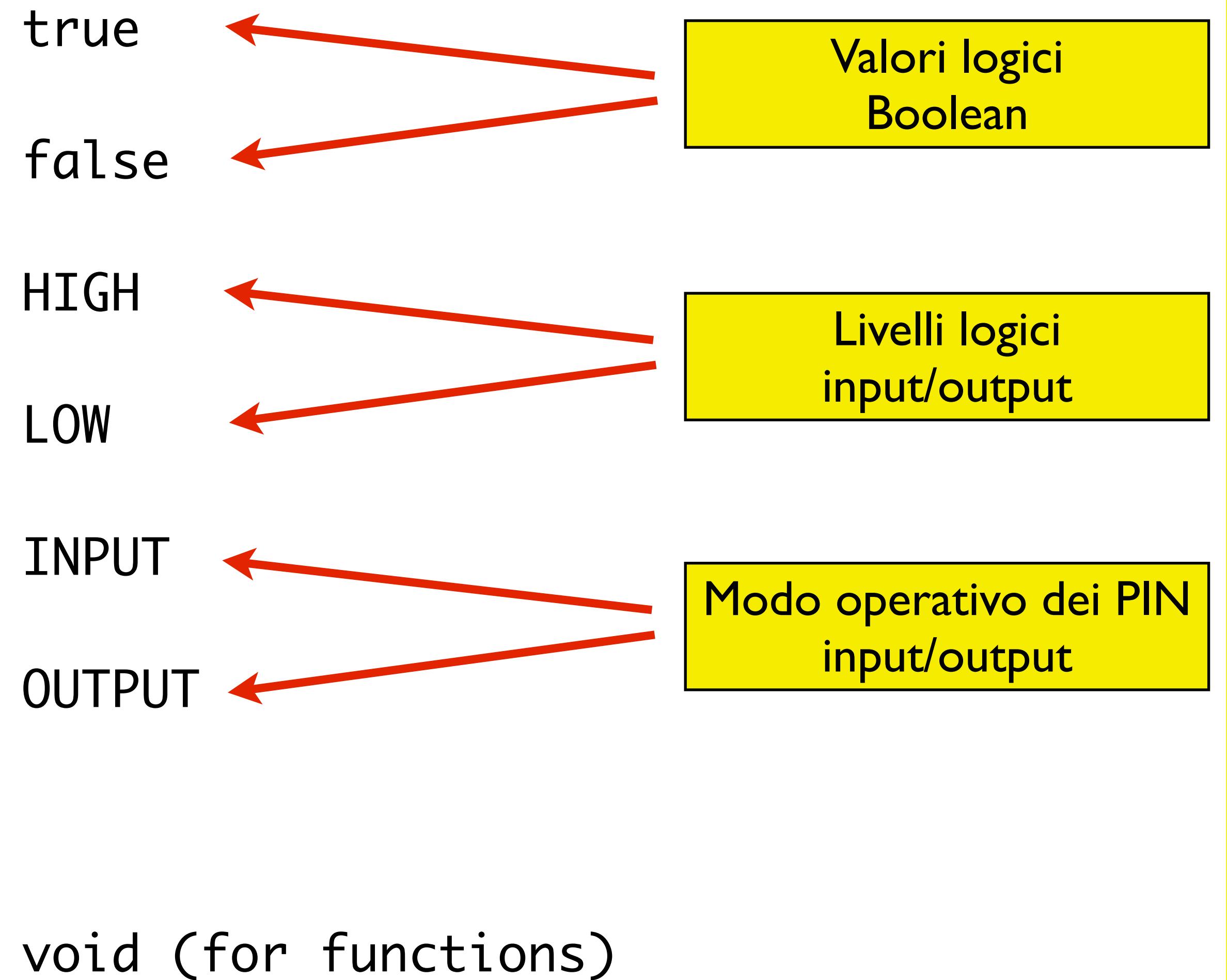
- Arduino has different kinds of Variables
 - Numbers (Integer and Floating Point)
 - Logic (boolean, “true” and “false”)
 - Characters
 - Vectors (Rarely used)

```
int i = 5;  
  
float PI = 3.14;  
  
boolean ready = false;  
  
char key = 'x';  
  
char name[6] = "Robert";
```



Constants

- The Arduino language predefines a series of constants:
 - The expressions true/false for logic
 - The state HIGH/LOW for turning on and off digital PINs
 - The Mode setting for the pins INPUT/OUTPUT
 - Values for functions and setup void/setup





Solid Numbers (integer)

- **byte**
 - 8 bit, 0:255
- **int**
 - 16 bit -32768:32767
- **unsigned int**
 - 16 bit, 0:65535
- **long**
 - 32 bit, -2147483468 :2147483647
- **unsigned long**
 - 32 bit, 0 :4294967295

```
byte floor = 12;  
  
int age = -1024;  
  
unsigned int = 33000  
  
long superdeep = -1208296770;  
  
unsigned long = 4294967295
```



Floating Point Variables

- **float**
 - 32 bit, -3.4028235E+38 and 3.4028235E+38
- **double**
 - 64 bit, maximum number representable is about 2×10^{308}

```
float angle = 1.57;  
  
double PI = 3.14159265358979323846264  
33832795028841971693993751058209749445  
92307816406286208998628034825342117069  
82148086513282306647093844609550582231  
72535940812848111745028410270193852110  
55596446229489549303819644288109756659  
33446128475648233786783165271201909145  
64856692346034861045432664821339360726  
02491412737245870066063155881748815209  
20962829254091715364367892590360011330  
53054882046652138414695194151160943305  
72703657595919530921861173819326117931  
05118548074462379962749567351885752724  
89122793818301194912983367336244065664
```



I'd like to talk to someone
about this.



Stitch and Solder

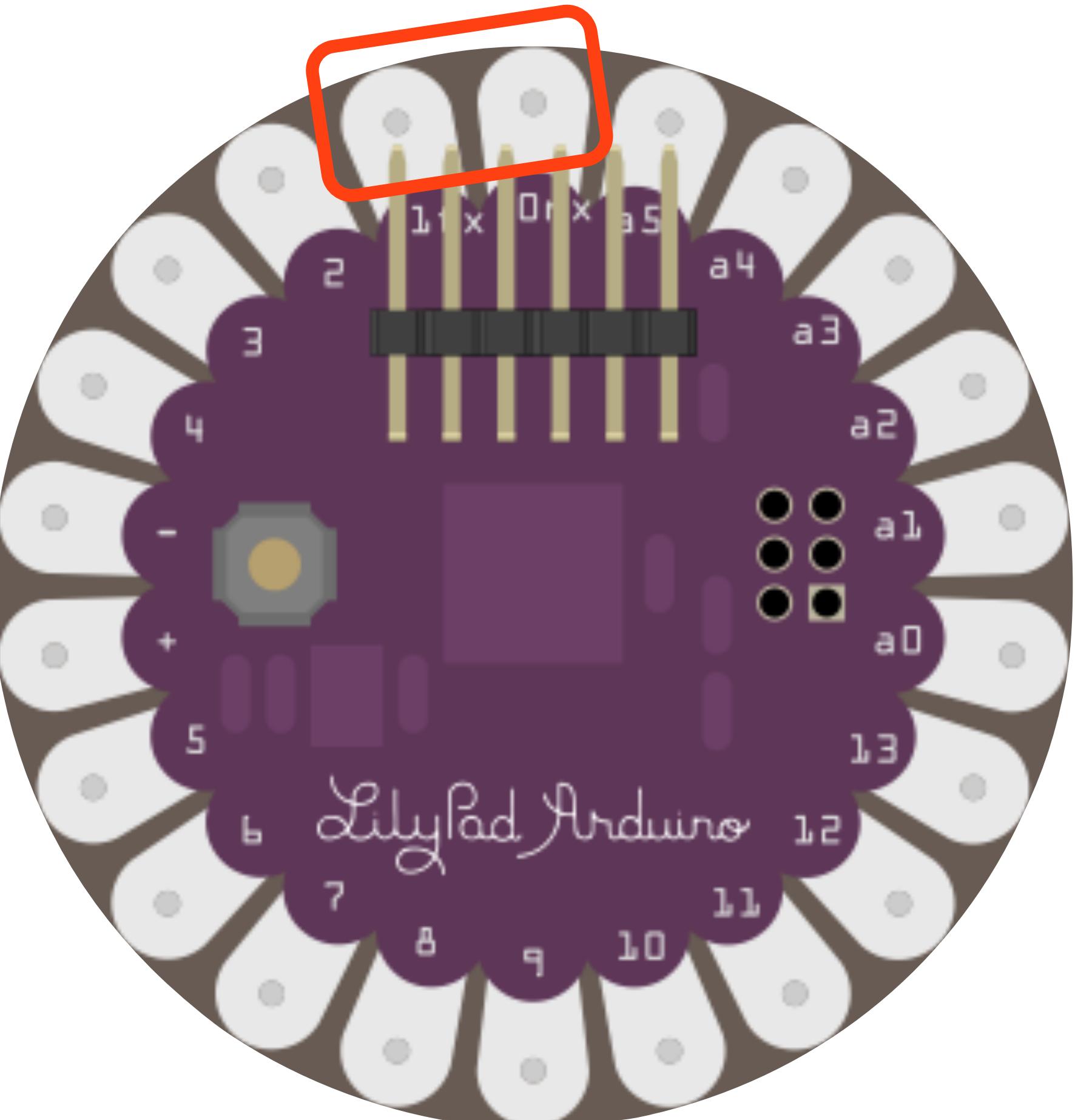
Serial Port

Button | Arduino 1.0.1

```
// initialize the LED pin as an output:  
pinMode(ledPin, OUTPUT);  
// initialize the pushbutton pin as an input:  
pinMode(buttonPin, INPUT);  
  
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);  
  }  
}
```

Arduino Due (Duemilanove w/ ATmega328 on /dev/tty.usbserial-A600aihG)

25





Serial Port

- Serial requires PINs 1 & 0
- The function `Serial.begin()` opens the serial port and sets it's speed in setup.
- The function `Serial.print()` writes a value to the serial port
- The function `Serial.println()` writes a newline

```
// initiate Serial Com and set speed  
// SPEED  
Serial.begin(9600);  
  
// Print the VALUE to the serial port  
Serial.print(VALUE);  
  
// Print a newline to the serial port  
Serial.println();
```



Add Code to your button

- Experimenting with textile sensors means having a way to understand what they are saying.
- Serial gives us a finite way to understand what is happening with our circuit.

```
void setup() {  
    pinMode(ledPin, OUTPUT);  
    // initialize the pushbutton pin as an  
    input:  
    pinMode(buttonPin, INPUT);  
    Serial.begin(9600);  
}  
  
void loop(){  
    buttonState = digitalRead(buttonPin);  
    HIGH:  
        if (buttonState == HIGH) {  
            // turn LED on:  
            digitalWrite(ledPin, HIGH);  
            Serial.println("We are ON!!!!");  
        }  
    else {  
        // turn LED off:  
        digitalWrite(ledPin, LOW);  
        Serial.println("We are OFF!!!!");  
    }  
}
```



Part 2 Elaborating

elaborate



Mathematic Operators

- There are 5 operators available for arithmetic
 - Addition: +
 - Subtraction: -
 - Multiplication: *
 - Division: /
 - Remainder: %
 - Integer math rounds to integer numbers

```
int a = 10;  
int b = 3;  
int c = 0;  
  
c = a + b;      // c = 13  
  
c = b - a;      // c = -7  
  
c = a * b;      // c = 30  
  
c = a / b;      // c = 3  
  
c = a % b;      // c = 1  
  
c = b % a;      // c = 3
```



Comparison Operators

- We can establish relations between numbers (values)
- The result is returned in a logical value of true or false
 - Equal: ==
 - Not Equal: !=
 - Greater than: >
 - Less than: <
 - Greater than or equal: >=
 - Less than or equal: <=

```
int x = 2;  
int y = 3;  
boolean b;  
  
b = (x == y); // Equal  
b = (x != y); // Not Equal  
  
b = (x > y); // Greater than  
b = (x >= y); // Greater than or equal  
  
b = (x < y); // Less than  
b = (x <= y); // Less than or equal
```



Control Structures

- Using Math and Comparison operators we can control our code
- **if** tests a specific case
- **else if** is used for additional tests
- **else** happens if nothing else happens

```
if (temperature > 21)
{
    digitalWrite(rightStrap, HIGH);
    delay(1000);
    digitalWrite(rightStrap, LOW);
    delay(1000);

}
else if (temperature < 21 &&
temperature > 10)
{
    digitalWrite(leftStrap, HIGH);
    delay(1000);
    digitalWrite(leftStrap, LOW);
    delay(1000);
}
```



Blocks

- A Block is a sequence of instructions grouped together with {}
- All lines of code in a block are executed as a group in sequential order.

```
{  
    a = 12;  
  
    b = c + d;  
  
    delay(1000);  
  
    digitalWrite(leftStrap, HIGH);  
    delay(1000);  
    digitalWrite(leftStrap, LOW);  
    delay(1000);()  
}
```



Unifying input and output

```
/* Read an analog Thermistor */

int thermistorSensor = 0;      // select the input pin for the thermistor
int temperature = 0; // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(thermistorSensor, INPUT);
  pinMode(13, output);
  Serial.begin(9600);
}

void loop() {
  temperature = analogRead(thermistorSensor); // read the value from the sensor:
  Serial.print("The temperature is ");
  Serial.println(temperature);
  digitalWrite(13,HIGH);
  delay (temperature);
  digitalWrite(13,LOW);
  delay (temperature);
}
```



Incrementing

- It is possible to increment and decrement a variable using the operators **++** and **--**
- Prefix operators calculate **before** it is used
- Suffix operators increment the variable **after** it is used

```
int a;  
int i = 5;  
  
// increment “i” by 1 which becomes 6  
i++;  
  
// first increment “i” which becomes 6  
// then assign “a”, which becomes 6  
a = ++i;  
  
// first assign “a”, which becomes 5  
// then increment “i” which becomes 6  
a = I++;
```



Cycles

- Allows us to repeat a block of code for a specific number of times or while a certain condition is true.
- The cycle **for()** runs the block of code ten times

```
int i;  
  
for (i = 0; i < 10; i++)  
{  
    powerLedOn();  
    doSound();  
  
    delay(1000);  
  
    powerLedOff();  
    stopSound();  
  
    delay(1000);  
}
```



- Executes a block of code as long as the logic case is true.

Cycle while()

```
while (temperature > 25)
{
    digitalWrite(rightStrap, HIGH);
    delay(1000);
    digitalWrite(rightStrap, LOW);
    delay(1000);
    temperature = analogRead
        (thermistorSensor);
}
```



Debounce

- Textile Sensors can give false positives due to humidity, barometric pressure or many other things.
- Debouncing makes sure the sensor is really pressed by a human.

```
buttonstate = digitalRead(textilebutton);
if (buttonstate == HIGH){
delay(500);
buttonstate = digitalRead(textilebutton);
if ((buttonstate == HIGH){
//Is the button really pressed?
...
}
```

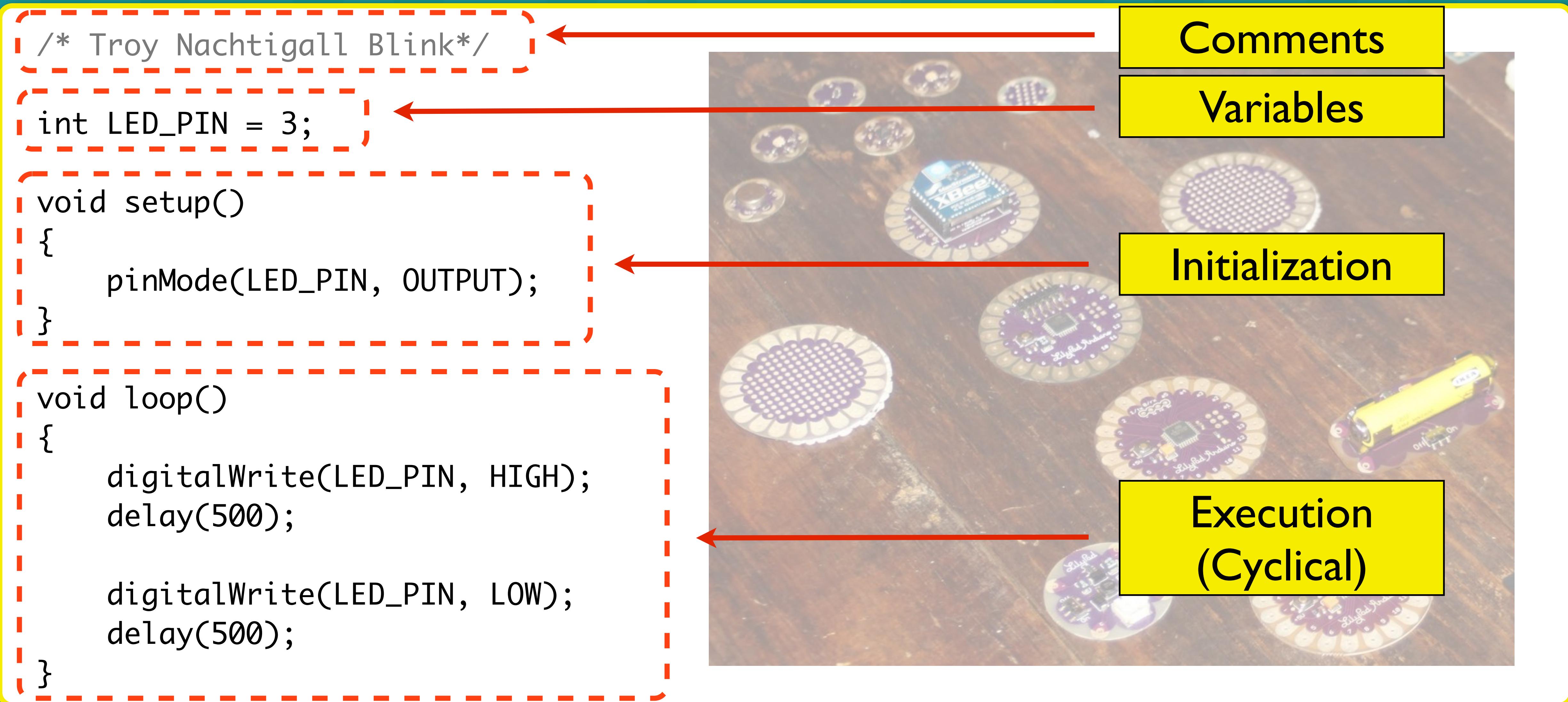


Stitch and Solder

Wrap up



Last Look at Hello Lilypad





Comments and Variables

- Remember to comment your code
- Remember to declare your variables

```
/* Blink the LED Straps  
by Troy Nachtigall  
troy@wearabletechnology.it */
```

```
int thermistorSensor = 0;  
int temperature = 0;  
int leftStrap = 10;  
int rightStrap = 11;
```



setup()

- **setup()** is a function that runs **just one time** at the beginning of the program
- It is responsible for setting up the Arduino:
 - Configuring PINs as input/output
 - configuring serial ports
 - initializing Libraries

```
void setup()
{
    // PIN 13 as output
    pinMode(13, OUTPUT);

    // PIN 2 Sensor input
    pinMode(2, INPUT)
}
```



loop()

- Loop is the “heart” of a sketch. The functions runs cyclicly as long as the Arduino has power (equivalent to an infinite while() cycle).

```
void loop()
{
    readSensorData();

    if (temp > 100)
    {
        powerOnFan();
    }
    else if (temp < 40)
    {
        powerOffFan();
    }

    // Start back at the beginning of
    // loop...
}
```



Input, Elaborate, Output





Go Deeper!

- Arduino - Reference Arduino - Foundations

The screenshot shows the Arduino Language Reference page. At the top is the Arduino logo (infinity symbol with minus and plus signs) and a search bar. Below the header is a navigation bar with links: Buy, Download, Getting Started, Learning, Reference (which is the active page), Hardware, FAQ, Blog, Forum, and Playground. Under the navigation bar, there are links to Reference, Language, Libraries, Comparison, and Changes. The main content area is titled "Language Reference". A sub-section titled "Structure" lists "+ setup()", "+ loop()", and "+ if". Another section titled "Control Structures" lists "+ if". The "Variables" section includes a "Constants" subsection with "+ HIGH | LOW", "+ INPUT | OUTPUT", "+ true | false", and "+ integer constants". The "Functions" section includes a "Digital I/O" subsection with "+ pinMode()", "+ digitalWrite()", and "+ digitalRead()". There is also a "Analog I/O" section.

Reference [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure	Variables	Functions
+ setup() + loop()	Constants + HIGH LOW + INPUT OUTPUT + true false + integer constants	Digital I/O + pinMode() + digitalWrite() + digitalRead()
Control Structures + if		Analog I/O



Lilypond

- LilyPond

The screenshot shows the LilyPond website homepage. At the top left is the "LilyPond" logo. At the top right are "log in or sign up" and a "Submit Project" button. The main navigation menu includes "Projects" (which is highlighted with a dashed purple border), "Resources", and "About". A link "see all projects" is also present. Below the menu, the "Most Loved Projects" section displays six project cards:

- LilyPad Wrist Band POV by quasiben
- Day and Night Decoration by tfalase
- Tilt Bracelet Circuit by dineeh
- Accelerometer Bracelet by emme
- POV Band for superkids! by iucreativitylabs
- Chairly the Chair by dineeh

Below this is the "Most Recent Projects" section, which also contains six project cards, though they are not fully visible in the screenshot.



Codeing is like writing a letter





Thank You

- Thank you very much for your participation!
- Troy Nachtigall
- @stitchandsolder
- troy@wearabletechnology.it



Acknowledgements

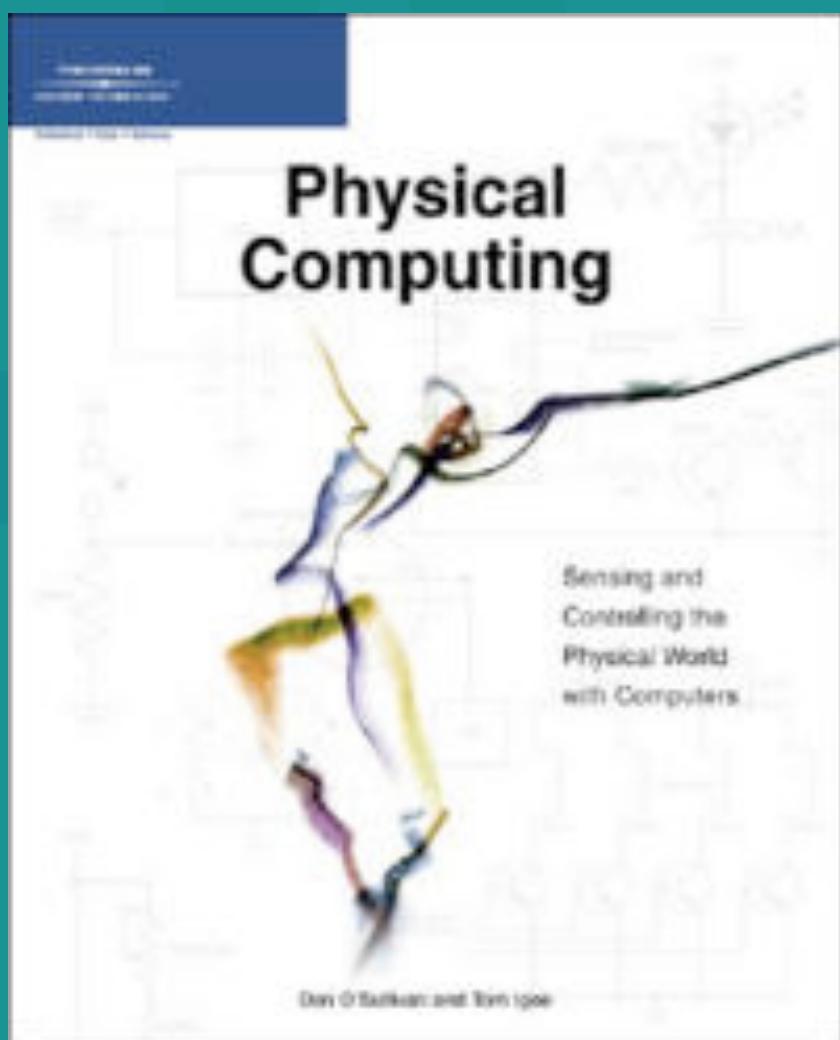
- Beyond the original material many thanks to the numerous people who contribute to the Arduino Community
- This presentation was inspired by:
 - Arduino Official Site: <http://www.arduino.cc>
 - Bionic Arduino: <http://todbot.com/blog/bionicarduino/>
 - Limor Ladyada web site: <http://www.ladyada.net/>
 - Wikipedia: <http://www.wikipedia.org>



Further Reading

- Physical Computing:
Sensing and Controlling
the Physical World with Computers

- Dan O'Sullivan, Tom Igoe
- Course Technology, 2004



- Making Things Talk
- Tom Igoe
- Make Books (O'Reilly), 2007

