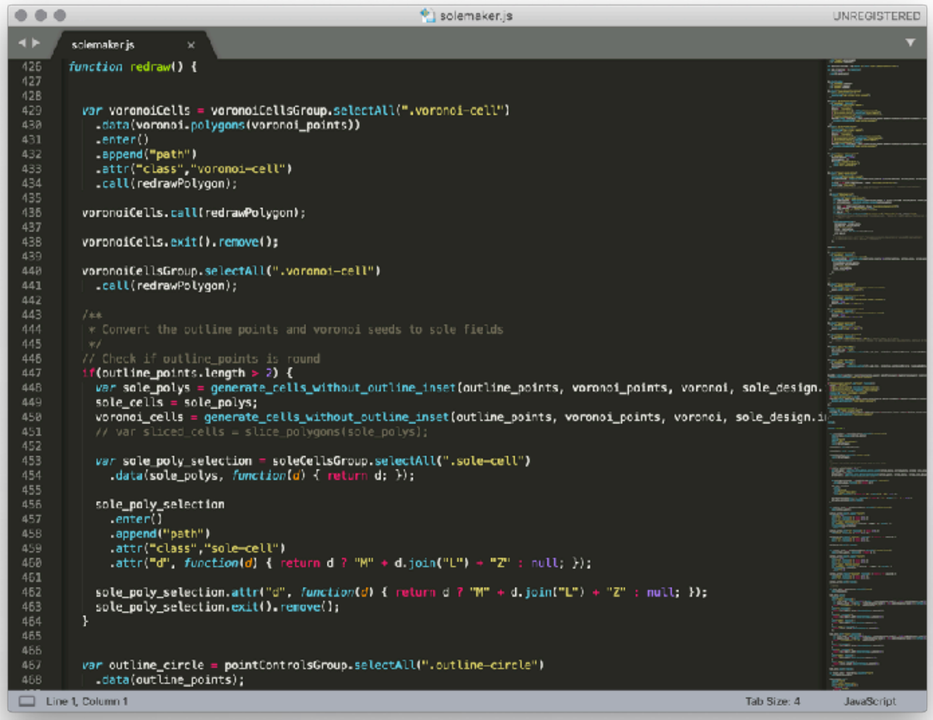
Data as a Material?

Using Arduino

Hfg-Karlsruhe May 25 2021



WEAR sustain

Solemaker

ArcInTexETN

Marie Skłodowska-Curie actions European Training Network (ETN)

Stitch and Solder

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**

Stitch and Solder

Program

- Part 0
  - Syntax and the Arduino Programming Language
- Part 1
  - Digital Output
  - Digital Input
- Part 2
  - Elaboration

Stitch and Solder

Input, Elaborate, Output

Programing 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.

Stitch and Solder

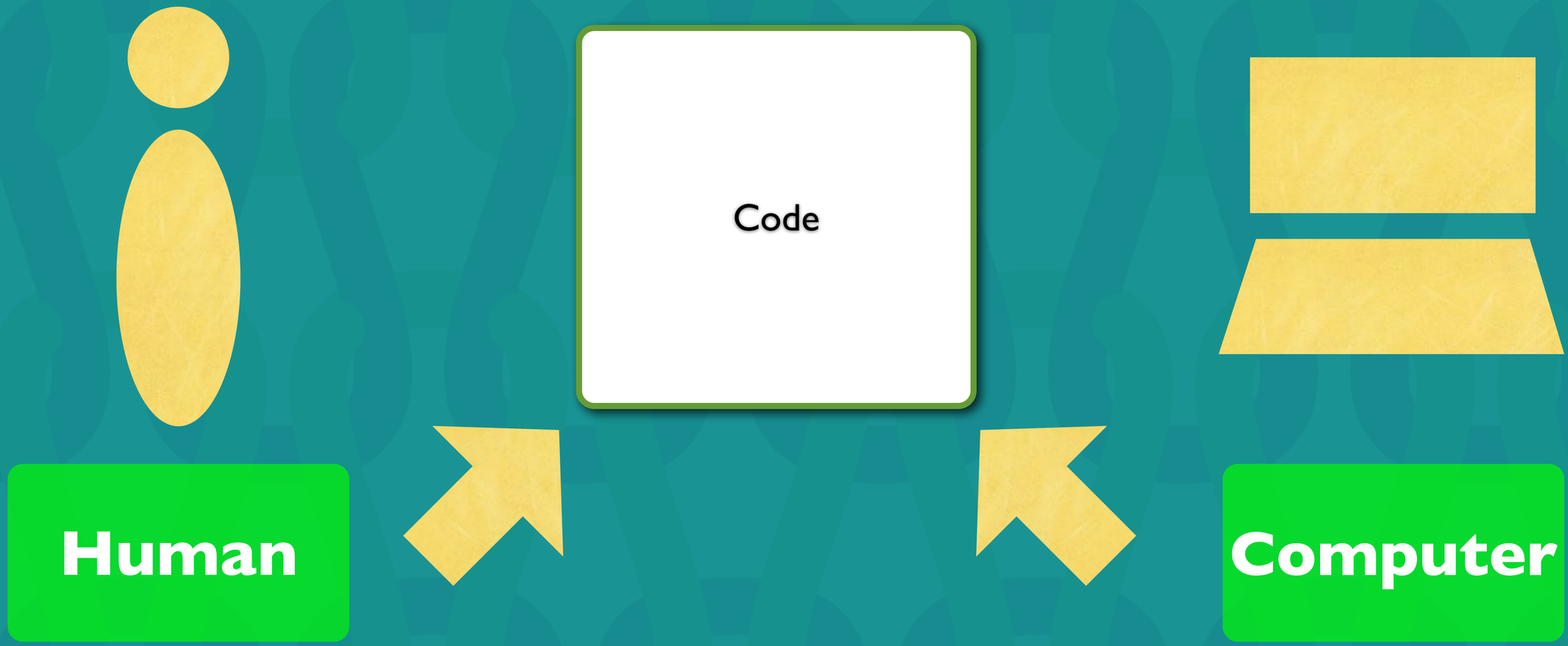
Welcome to the Programming Language “Arduino”

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

Part 0 :: Syntax

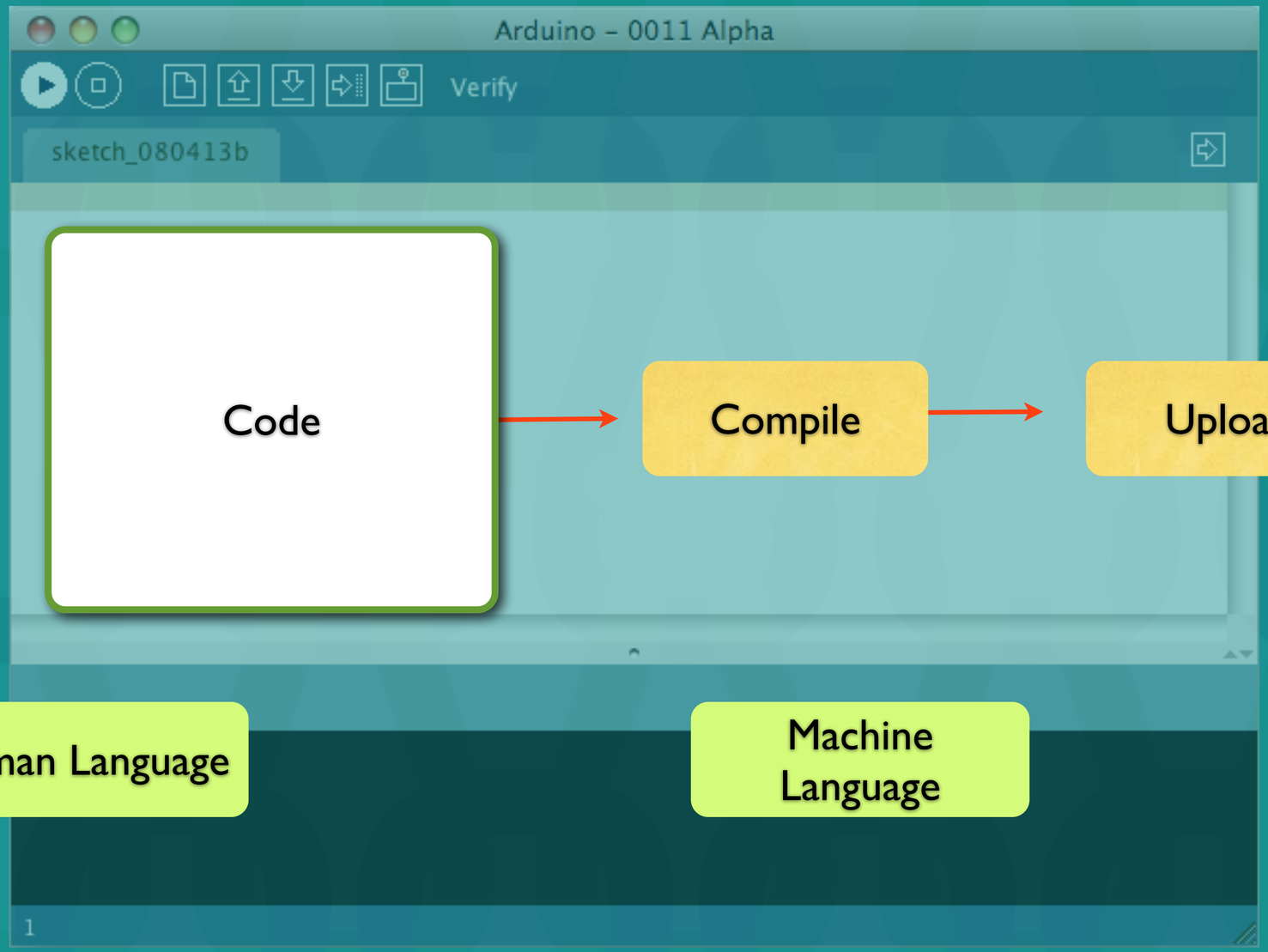
Stitch and Solder

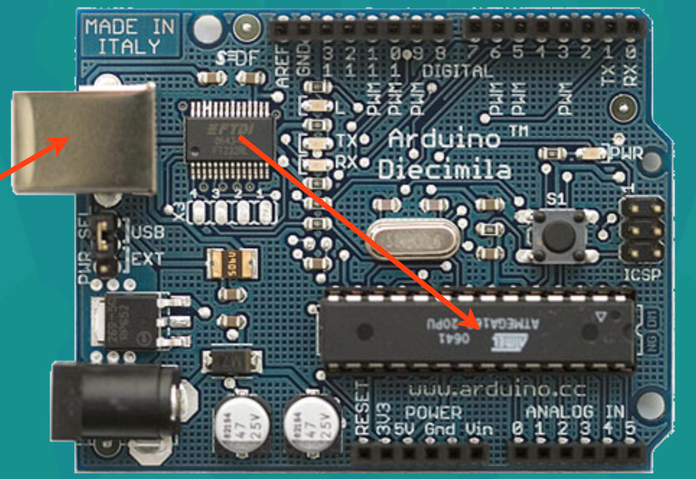
Designers Guide to programming



Stitch and Solder

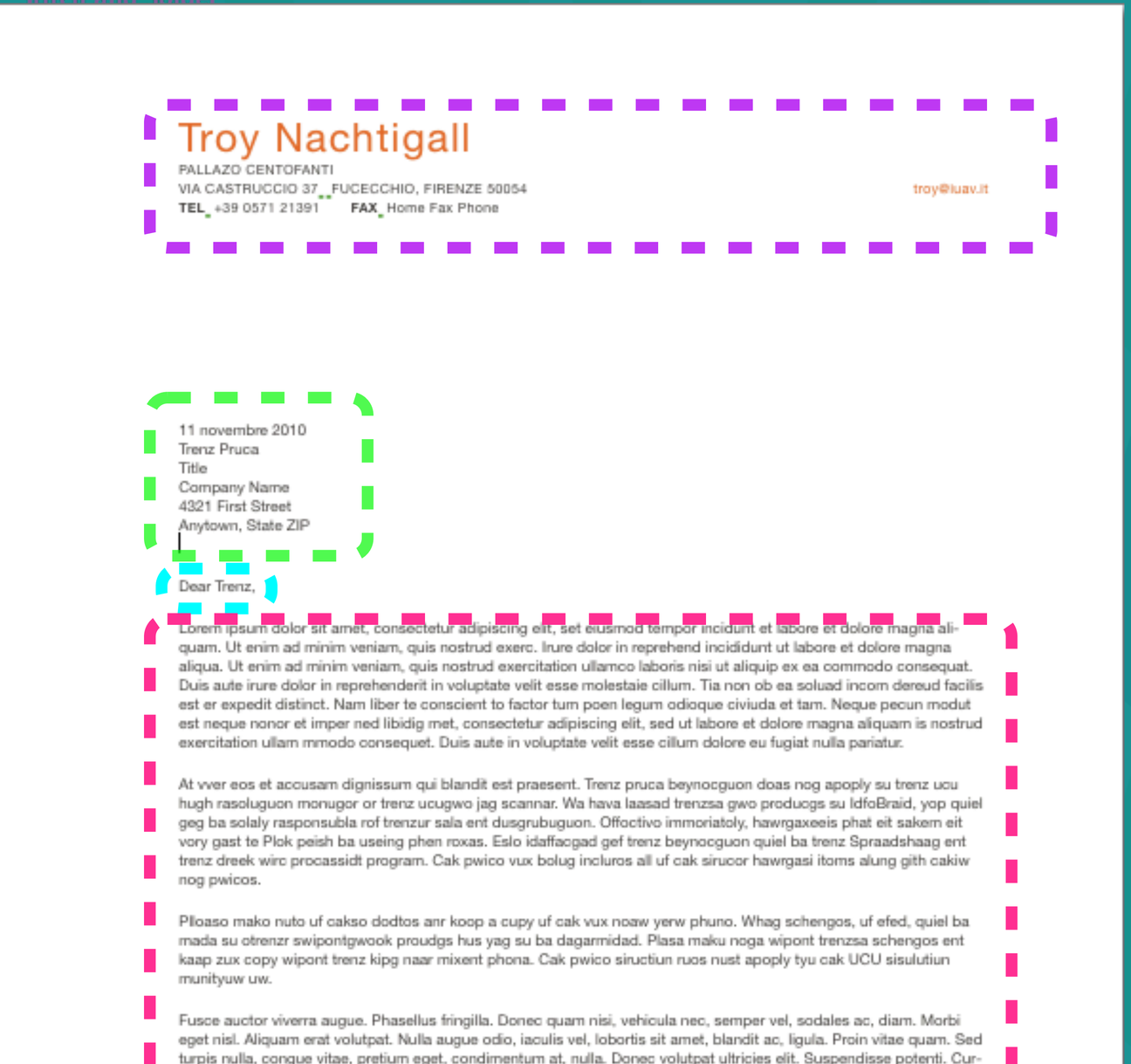
Programming an Arduino





Coding is like writing a letter

Stitch and Solder



Letterhead	Comments
Header	Variables
Salutation	Initializatio
Body	Execution
Signature	Comments

Structure of a Sketch

Stitch and Solder

```
/* 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 → Letterhead

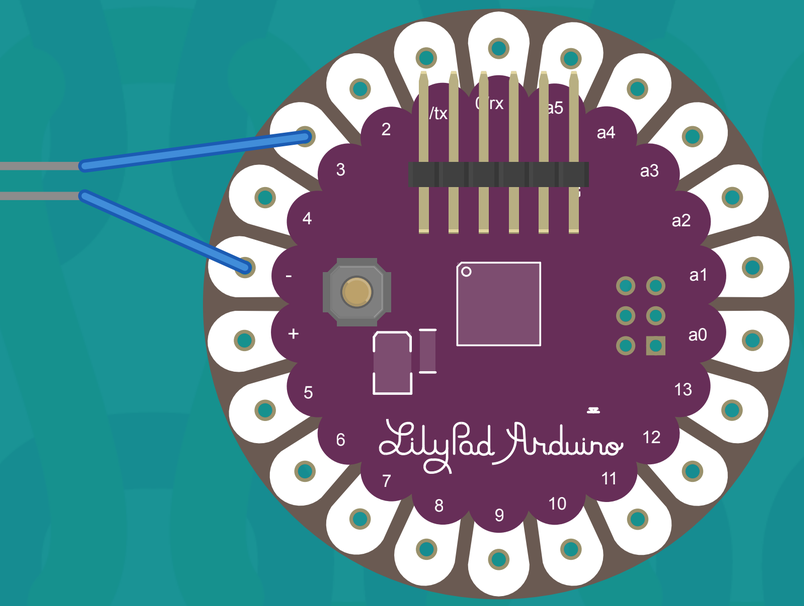
Variables → Header

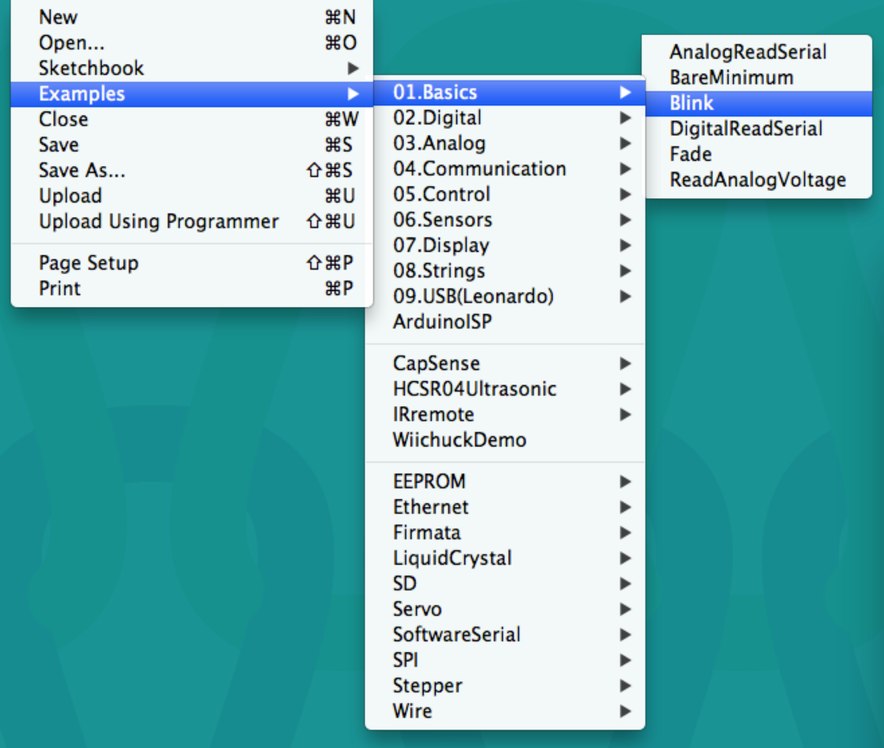
Initializatio → Salutation

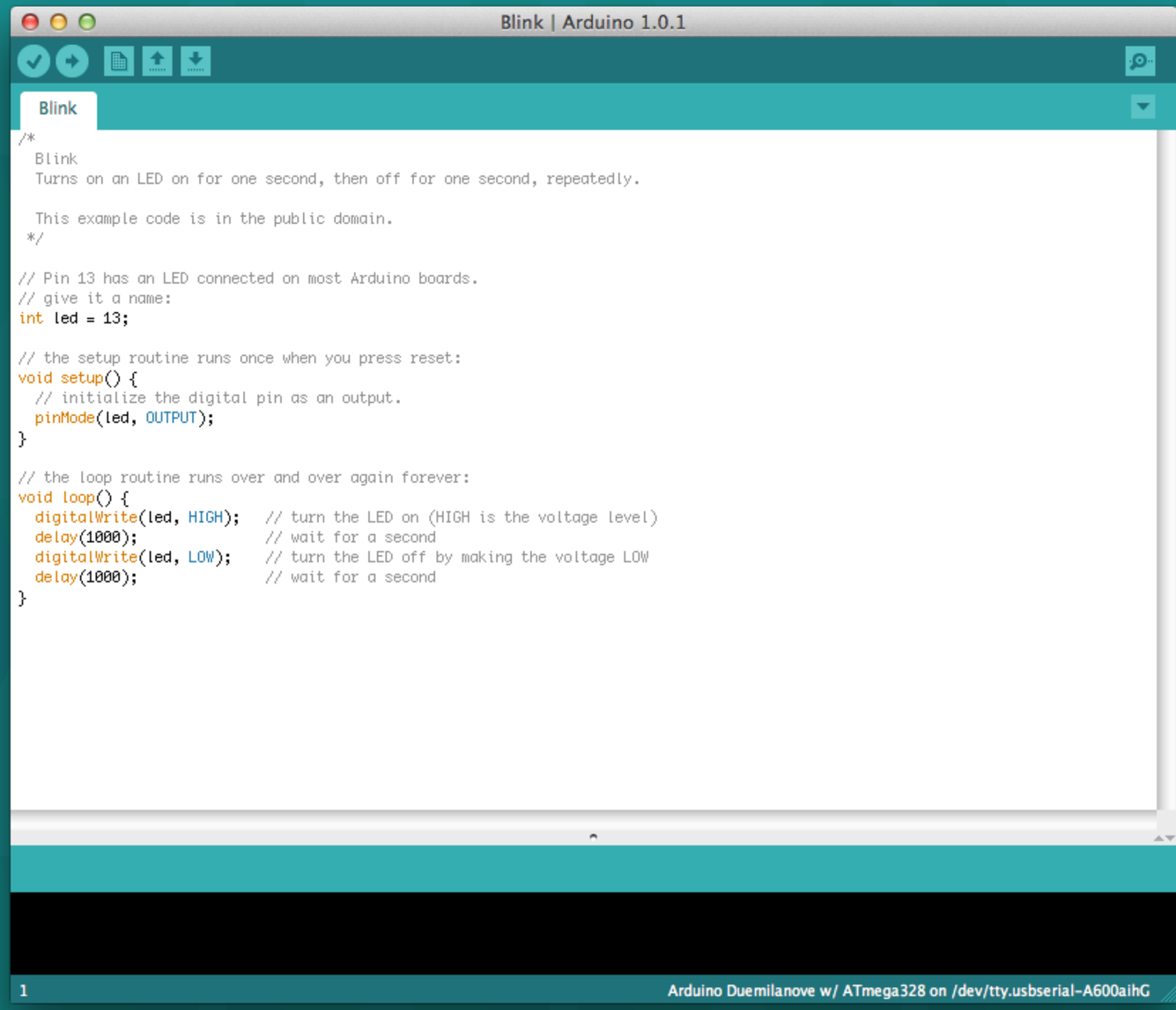
Execution → Body

Stitch and Solder

LET'S TRY Blink



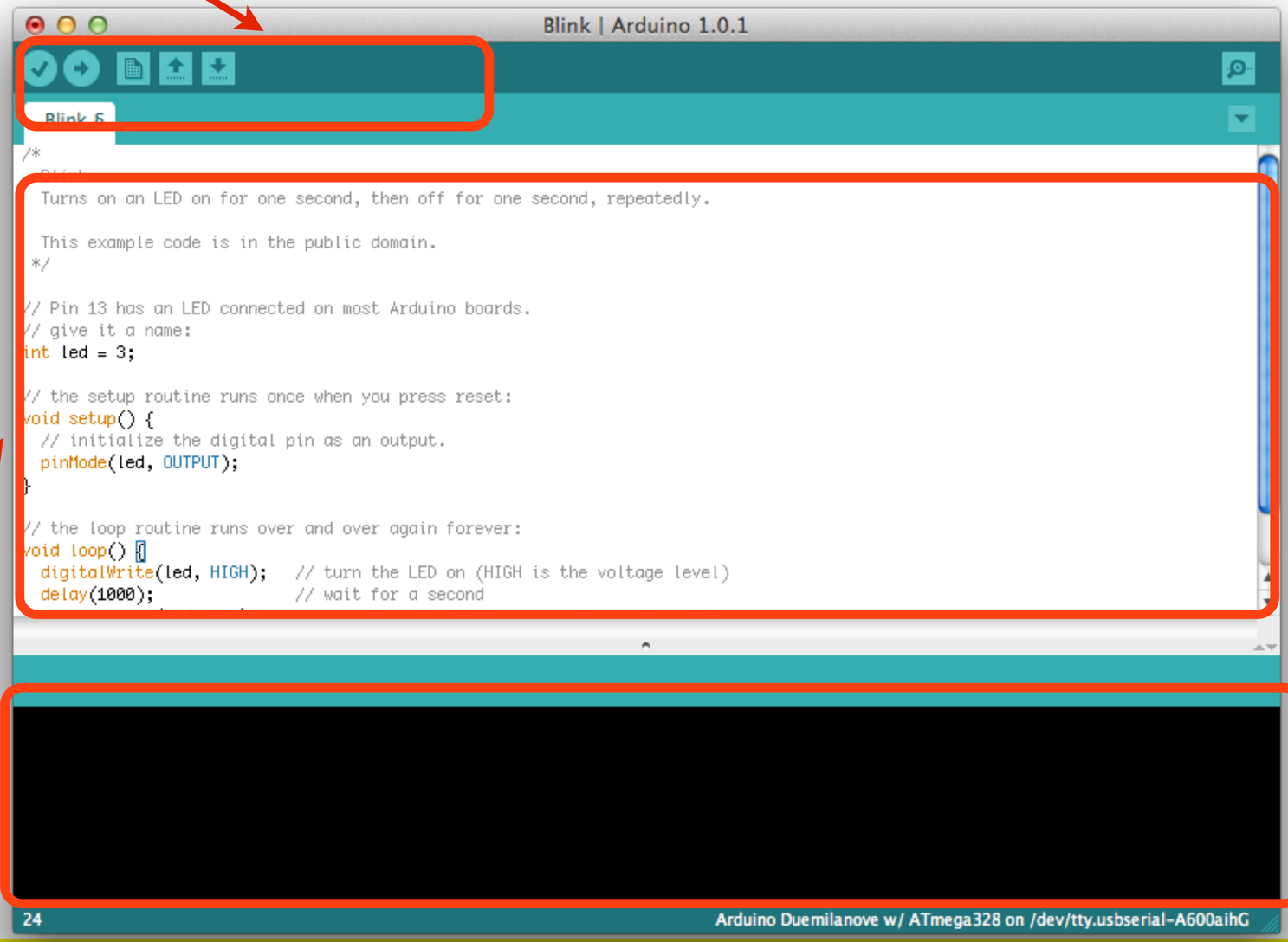




Stitch and Solder

IDE

Toolbar



Console

Editor

IDE Toolbar

Stitch and Solder

File Manager
New, Open, Save

Compile

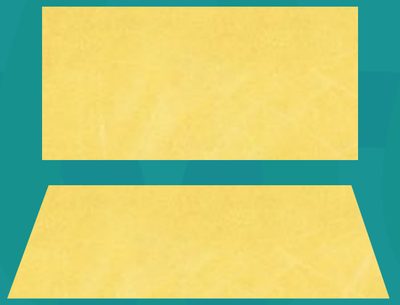
Activate Serial Console

Compile and Upload

Stitch and Solder

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 ( ).

Stitch and Solder

Syntax

Human Language

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

Commenting

- 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.

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

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

Part 1 Input, Output, Serial

Stitch and Solder

Part 1 - Digital Output

Output

Arduino Test

Stitch and Solder

```
/*
  Blink
  Turns on an LED on for one second, then off for
one second, repeatedly.
  This example code is in the public domain.*/
```

Sketch Name and Description

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

Declare Outputs and Inputs
leftStrap is 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
}
```

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

Stitch and Solder

output - digitalWrite

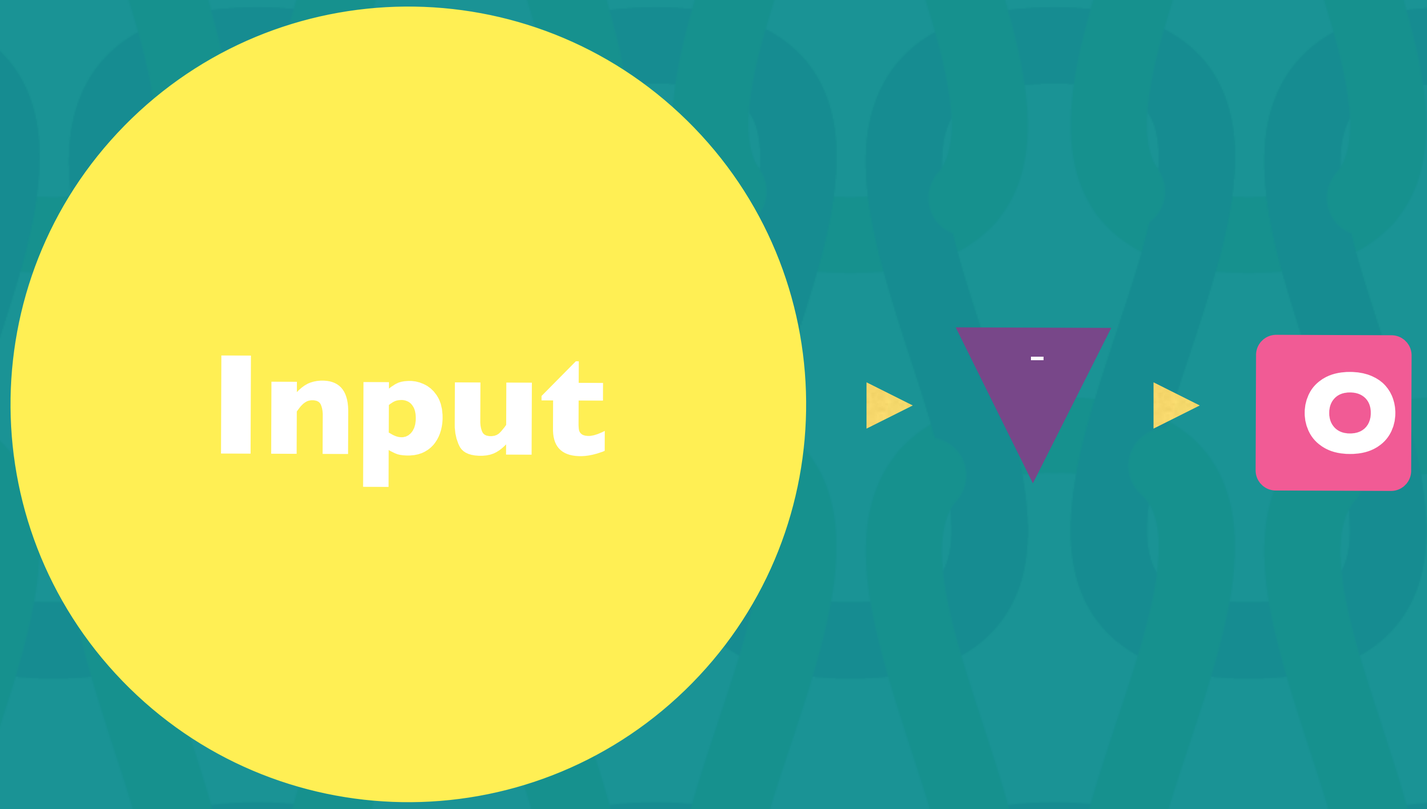
- To turn on an output PIN, we set the pin to HIGH
- LOW turns a pin off.
- we use the function (verb) digitalWrite to talk to PINs
- digitalWrite(PIN, HIGH/LOW)

  - delay is in Milliseconds
  - don’t forget the ;

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

Stitch and Solder

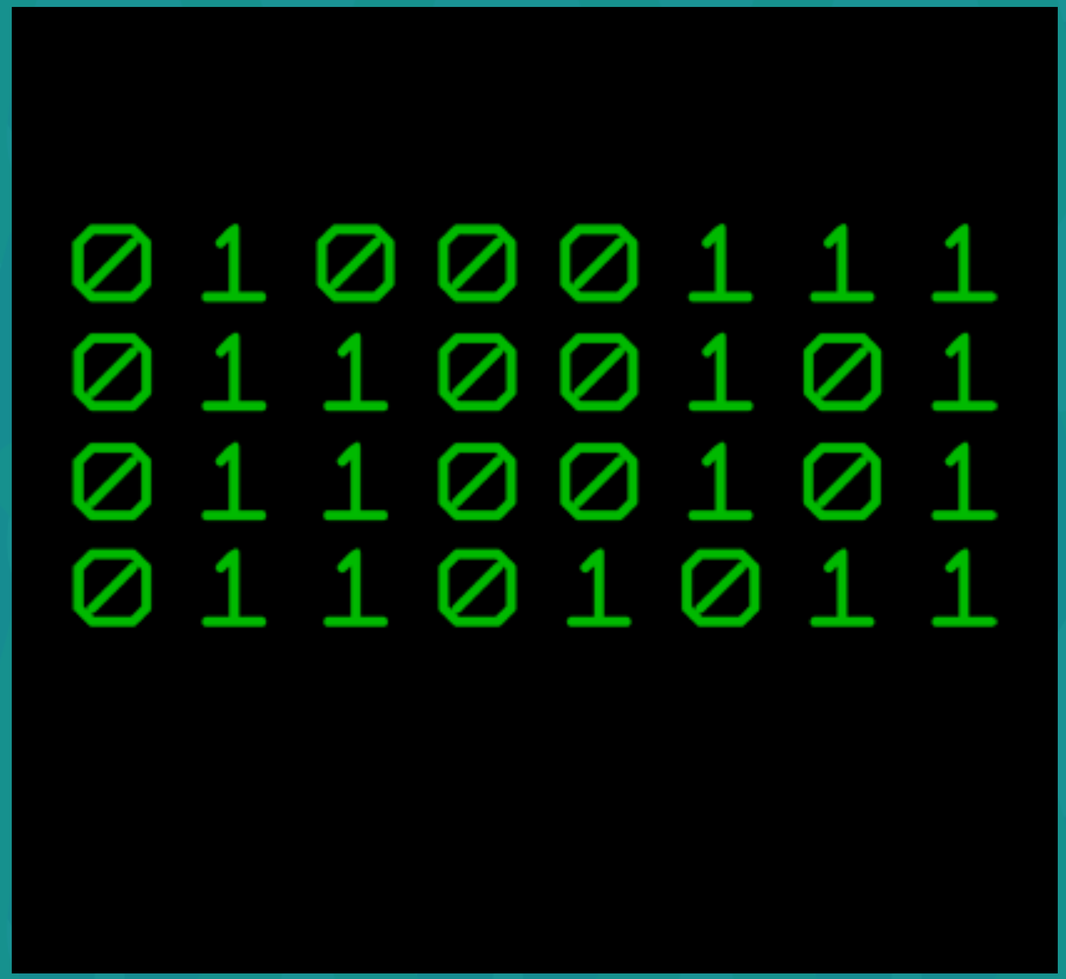
Part 1 Analog INPUT



Stitch and Solder

digital information

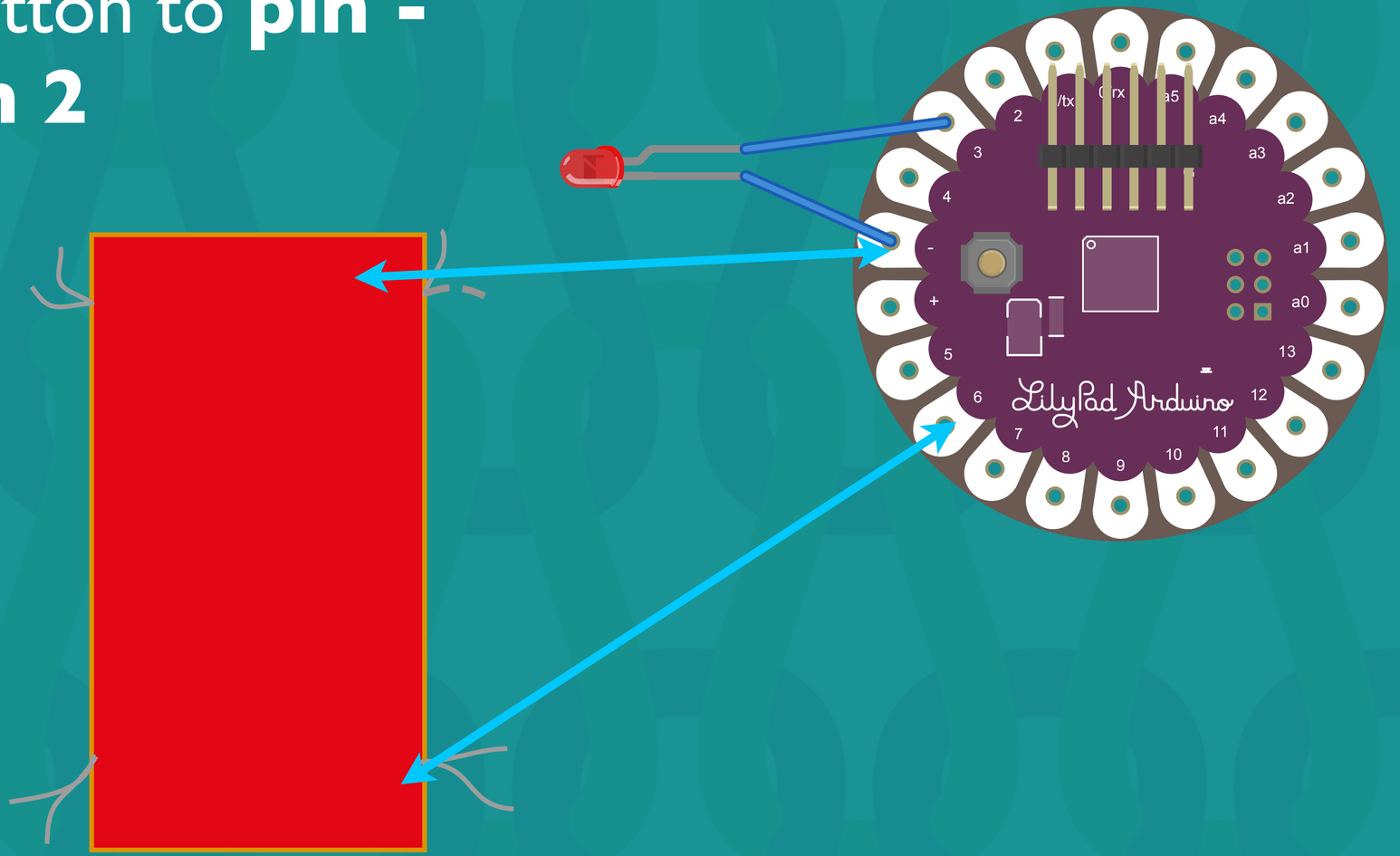
- Digital is **1** or **0** (2 bit)
- Digital is **on** or **off**
- in Arduino
  Digital is **5v** or **0v**
- logically, Digital is
  True or **False**
- Syntactically is
  High or **Low**



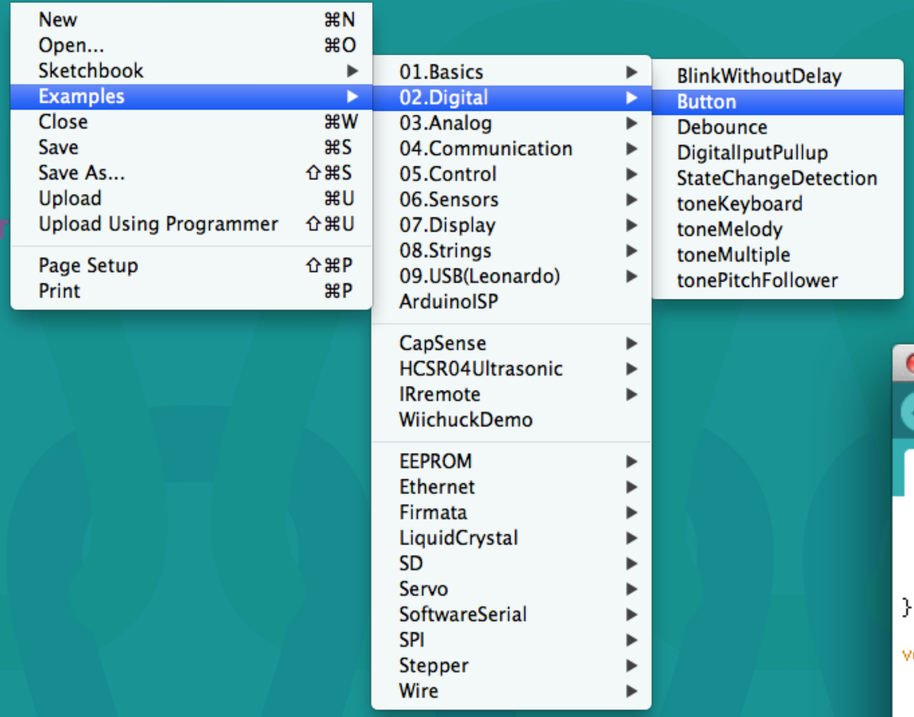
Stitch and Solder

Connect our buttons

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

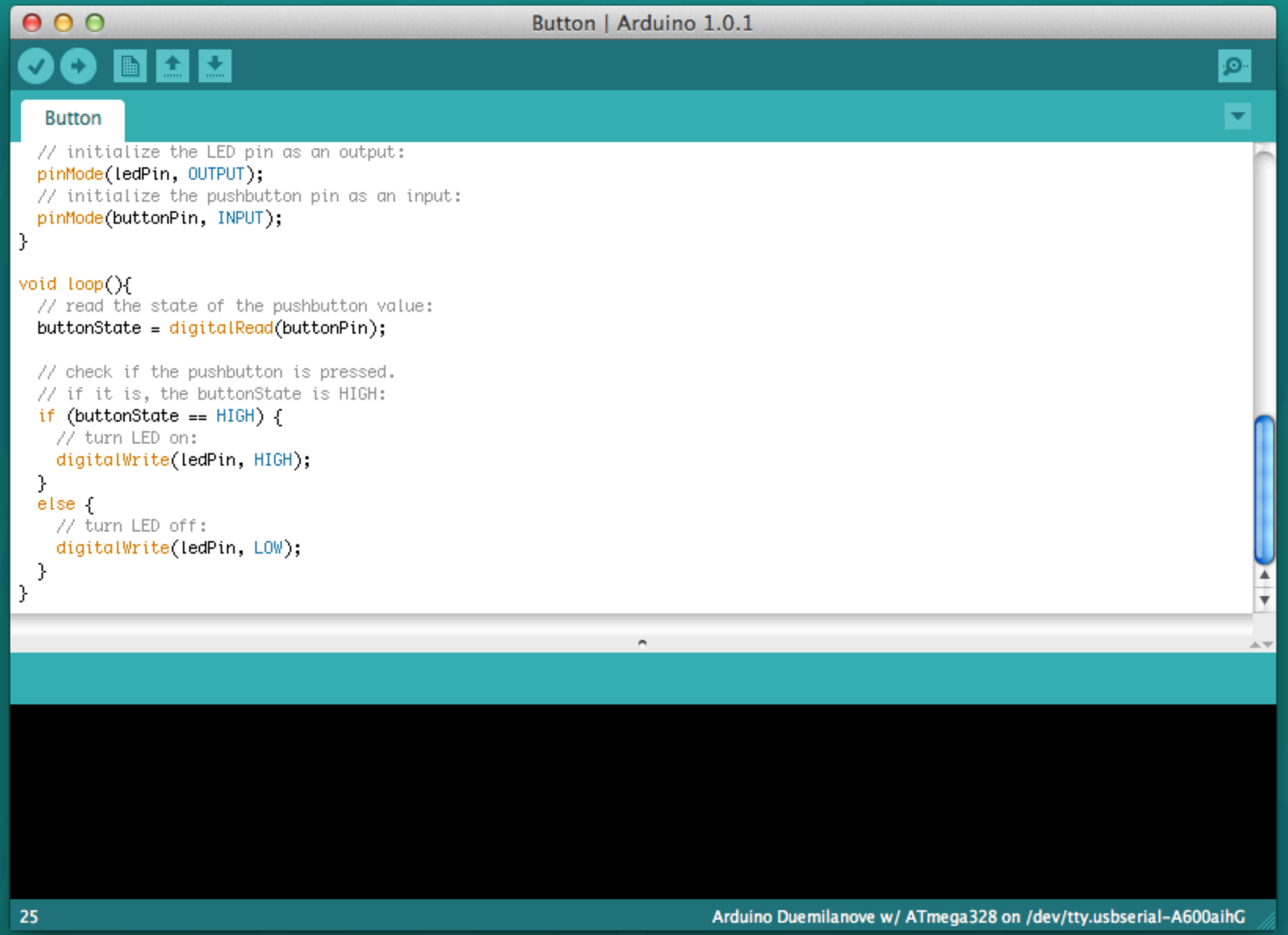


Stitch and Solder



Digital Button

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



Variables for Designers

1+2=3

a+b=c

vs.

3=1+2

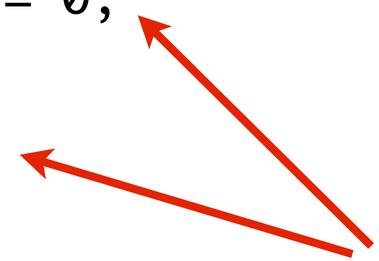
c=a+b

Stitch and Solder

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

Stitch and Solder

Kinds of Variables

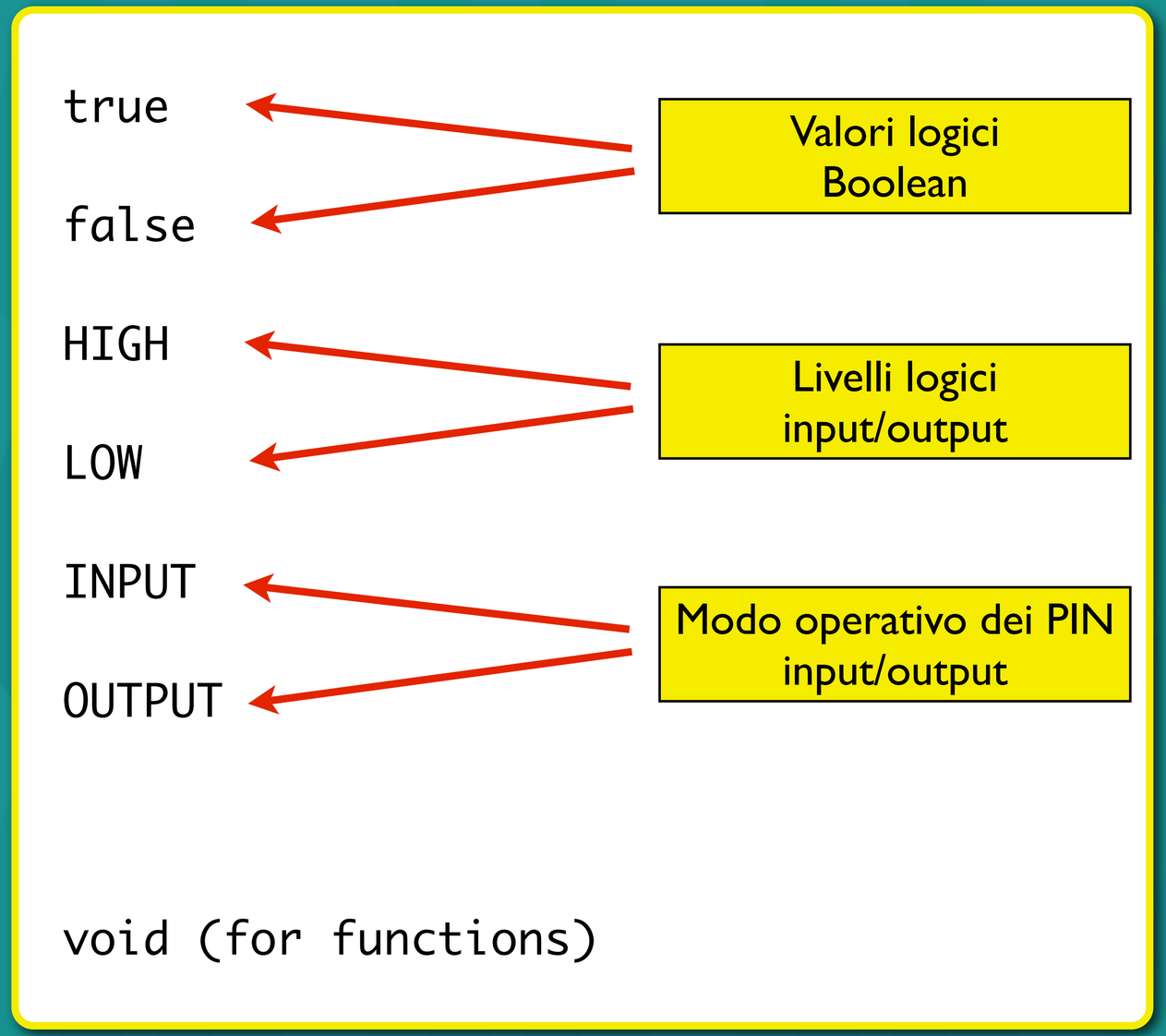
- Arduino has different kinds of Variables
  - Numbers (Integer and Floating Point)
  - Logic (boolean, “true” and “false”)
  - Characters
  - Vectors (Rarely used)

```
int i = 5;

float PI = 3.14;

boolean ready = false;

char key = ‘x’;

char name[6] = “Robert”;
```

Stitch and Solder

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 x 10^308

```
float angle = 1.57;

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

Stitch and Solder

I’d like to talk to someone about this.

Stitch and Solder

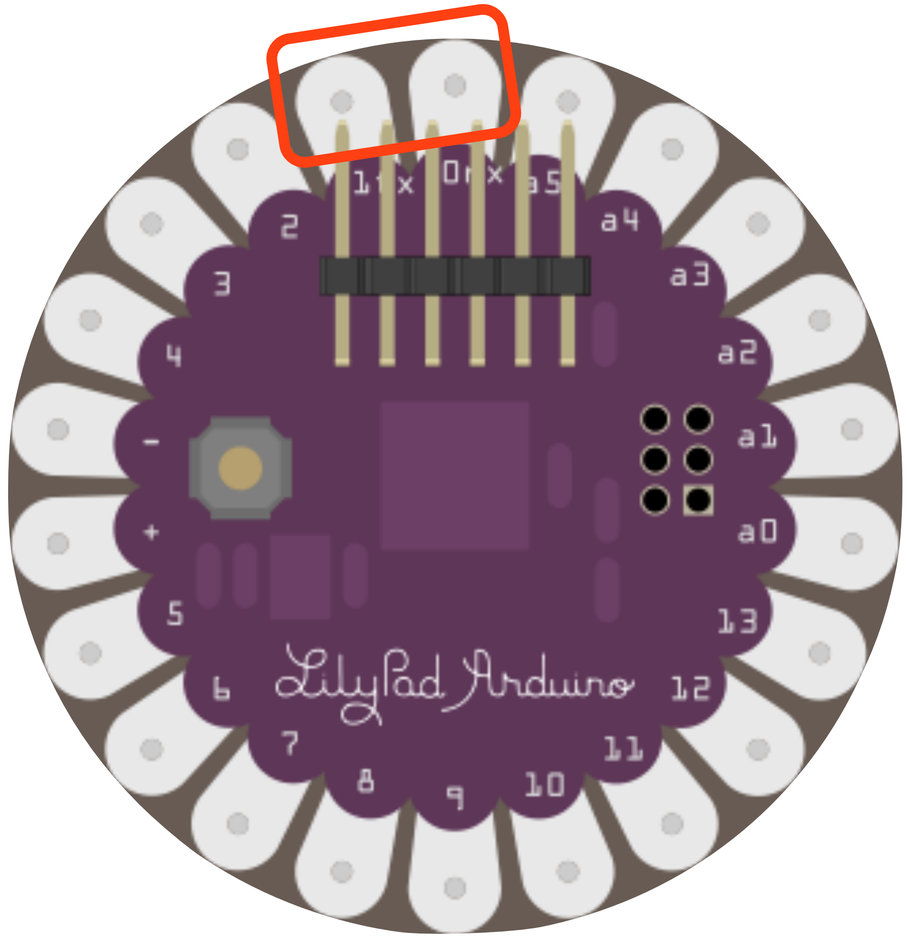
Serial Port

Button | Arduino 1.0.1

Button

```
  // 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);
  }
}
```

25 Arduino Duemilanove w/ ATmega328 on /dev/tty.usbserial-A600aihG



Stitch and Solder

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 {
```

Stitch and Solder

Part 2 Elaborating

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 that or equal

b = (x < y);  // Less than
b = (x <= y); // Less than or equal
```

Stitch and Solder

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);
}
```

Stitch and Solder

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 variabile **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++;
```

Stitch and Solder

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);

}
```

Stitch and Solder

Cycle while()

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

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

Stitch and Solder

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?
...
```

Wrap up

Last Look at Hello Lilypad

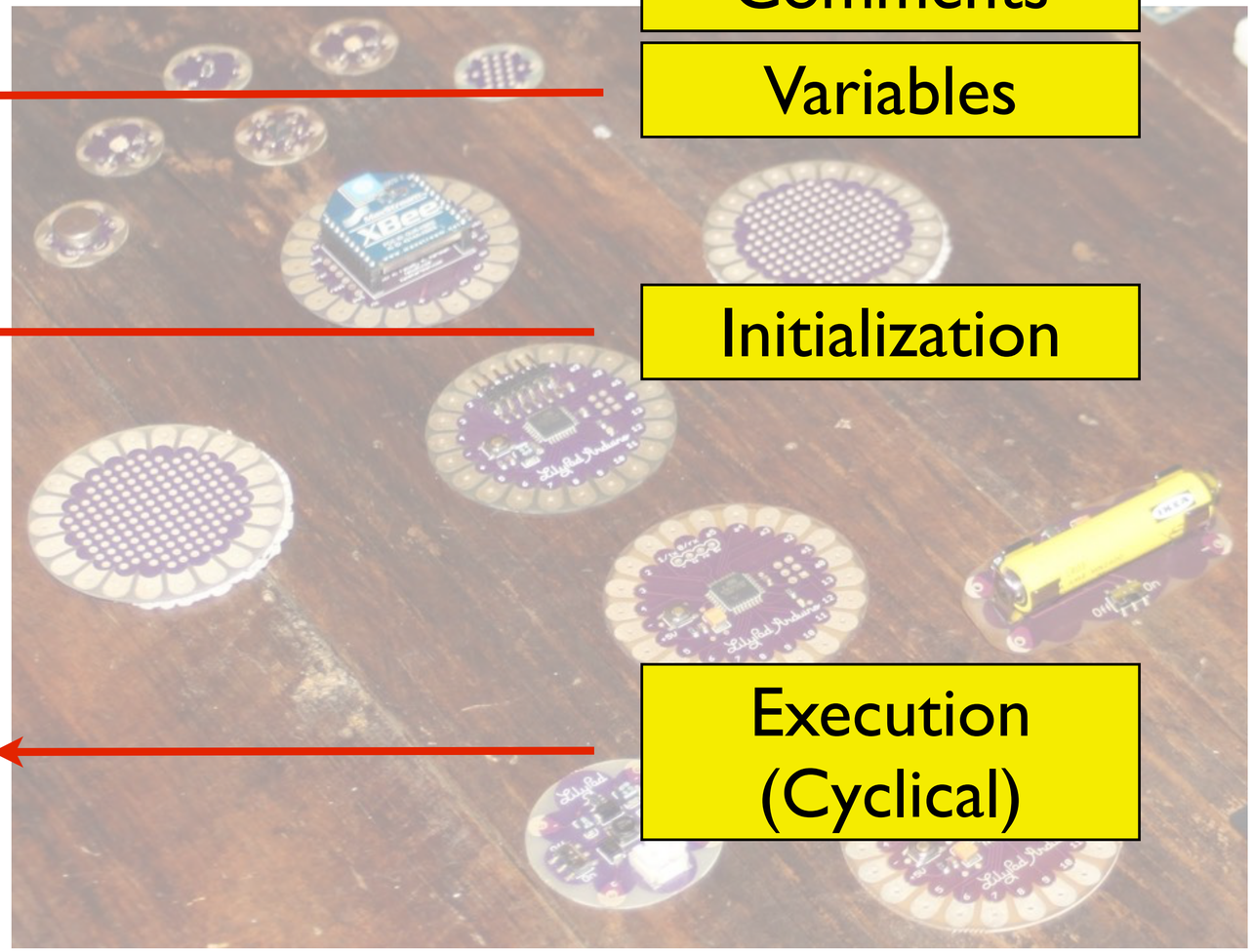
```
/* Troy Nachtigall Blink*/

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 (Cyclical)



Stitch and Solder

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...
}
```

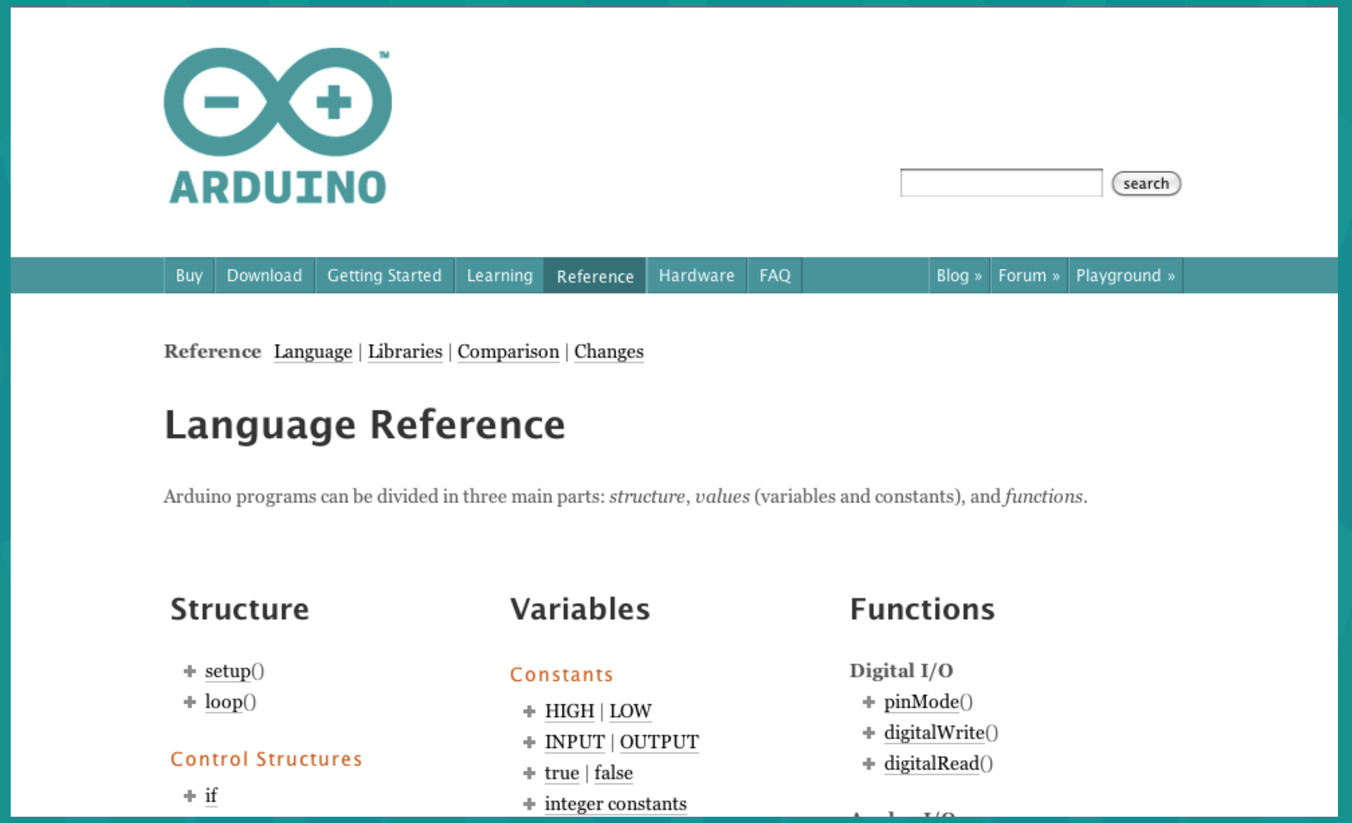
Stitch and Solder

Input, Elaborate, Output

Stitch and Solder

Go Deeper!

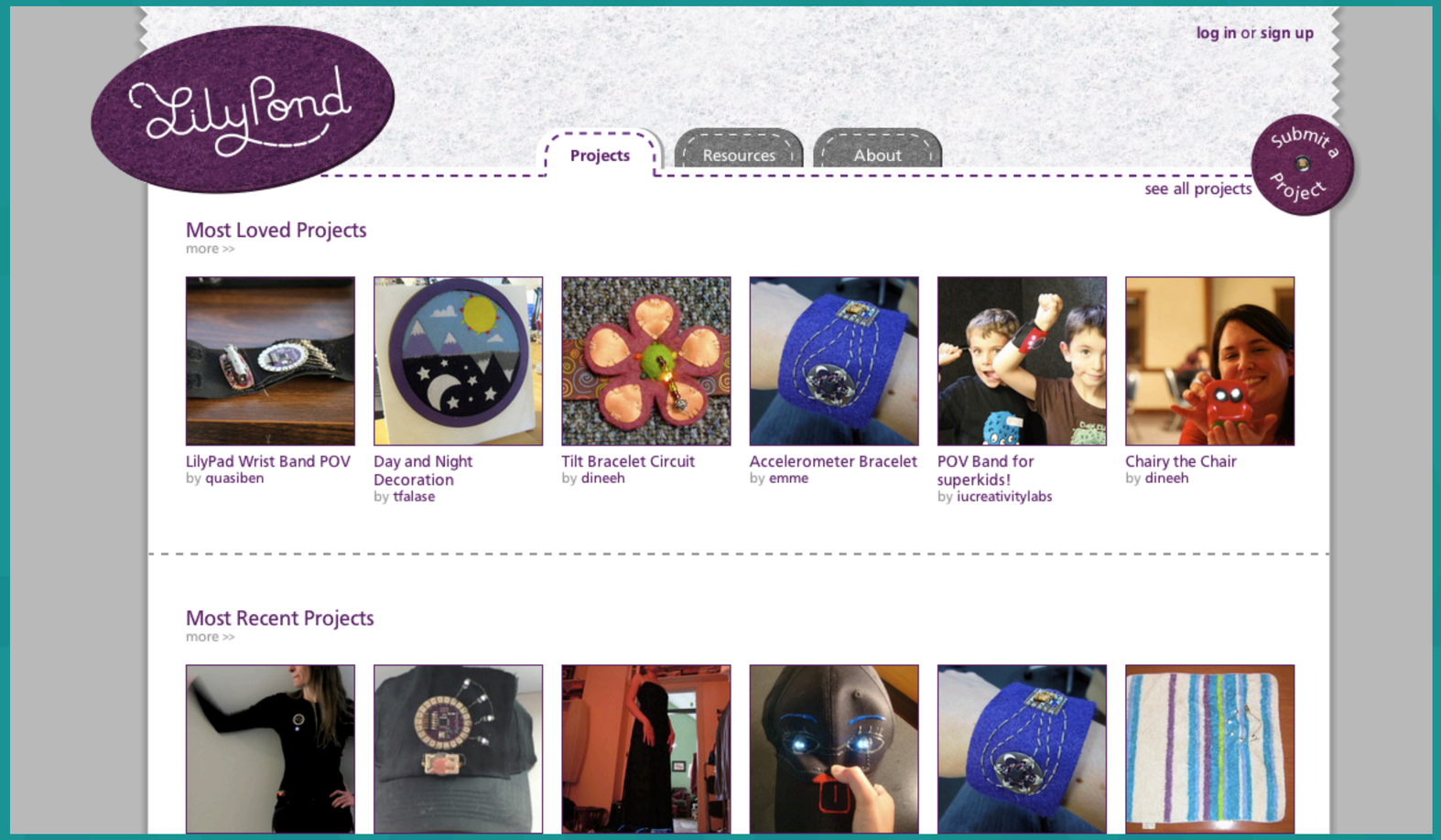
- Arduino - Reference Arduino - Foundations



Stitch and Solder

Lilypond

- LilyPond



Stitch and Solder

Codeing is like writing a letter

Letterhead	Comments
Header	Variables
Salutation	Initializatio
Body	Execution
Signature	

Thank You

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

Stitch and Solder

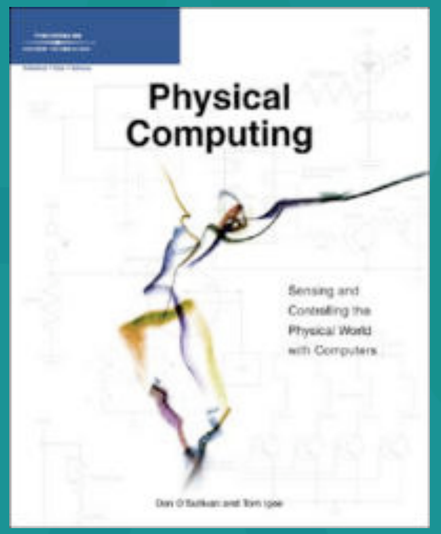
Acknowledgements

- 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**

Stitch and Solder

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