Arduino

open-source hardware and integrated development environment

Goals of this Presentation

I. **Introduce Arduino**

II. **History behind Arduino**

   I. *Timeline*

   II. *Influence*

   III. *Basic UI and Coding*

III. **Hardware and Comparisons**

   I. *Basic Wiring and Arduino Hardware*

   II. *Comparison between the two Ecosystems*

IV. **Talk about the Software and the IDE**

   I. *Cover a standard Arduino program*

   II. *Build a basic program with I/O*

   III. *Demo libraries*

V. **Recap the Presentation**

VI. **Offer Resources**

   I. *Show some flexibility and additional Hardware*

What is Arduino

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments."
- arduino.cc

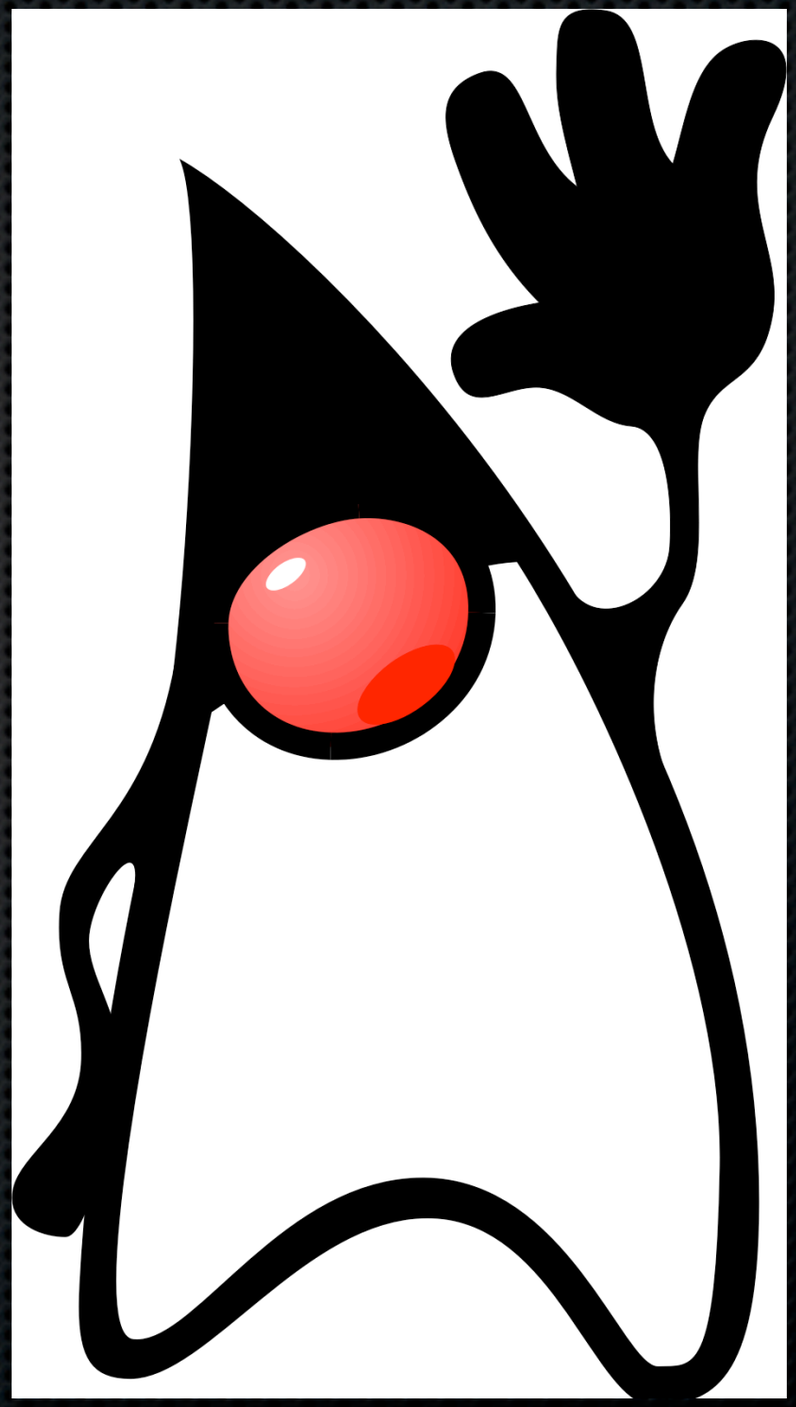
Those words, taken from the arduino.cc/en web page, embody the goal that many scientists, artists, designers and some computer scientists have had for many years.

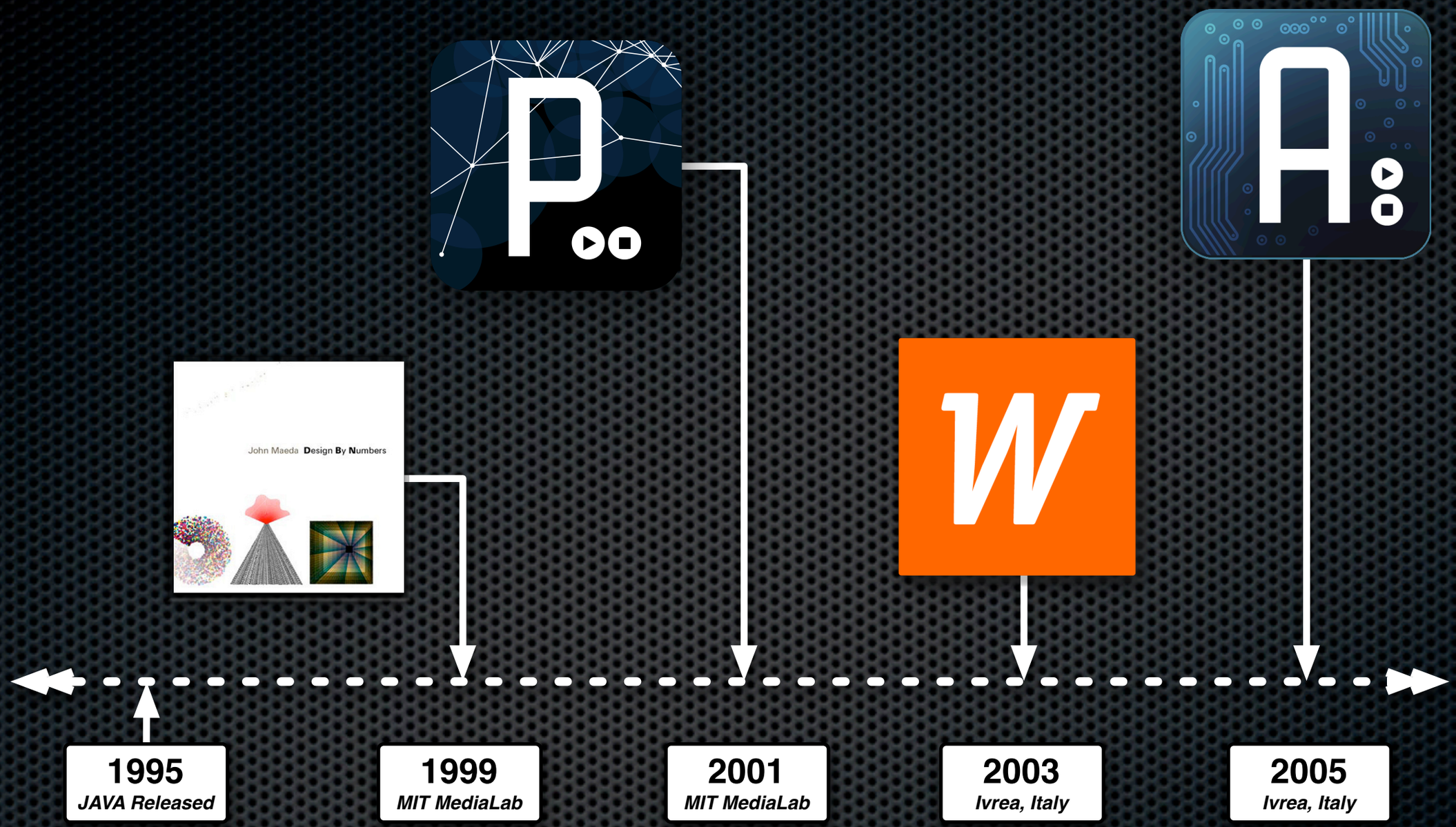
Arduino comes out of many years of tangential development. Scientist at MIT and Designers at Ivrea Institute in Italy have been working hard towards the same goal. That goal is a simple IDE, programming language and in some cases hardware environment with which to construct art and prototype ideas.

Arduino is the most recent installment of that dream. In this presentation I hope to enumerate the success' of the past and describe the features and benefits of the Arduino Environment.

Background

Where Arduino Came from



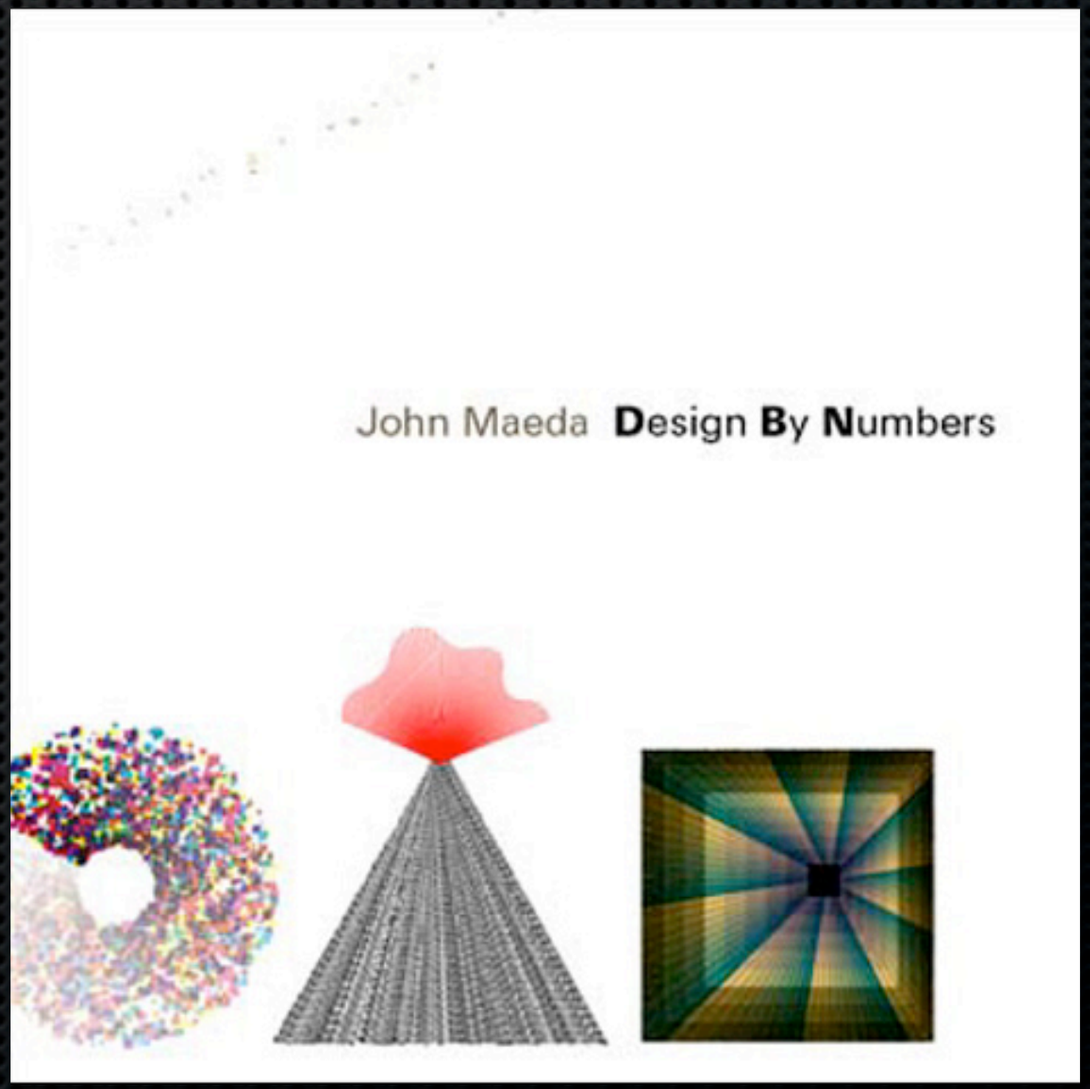


History of Arduino

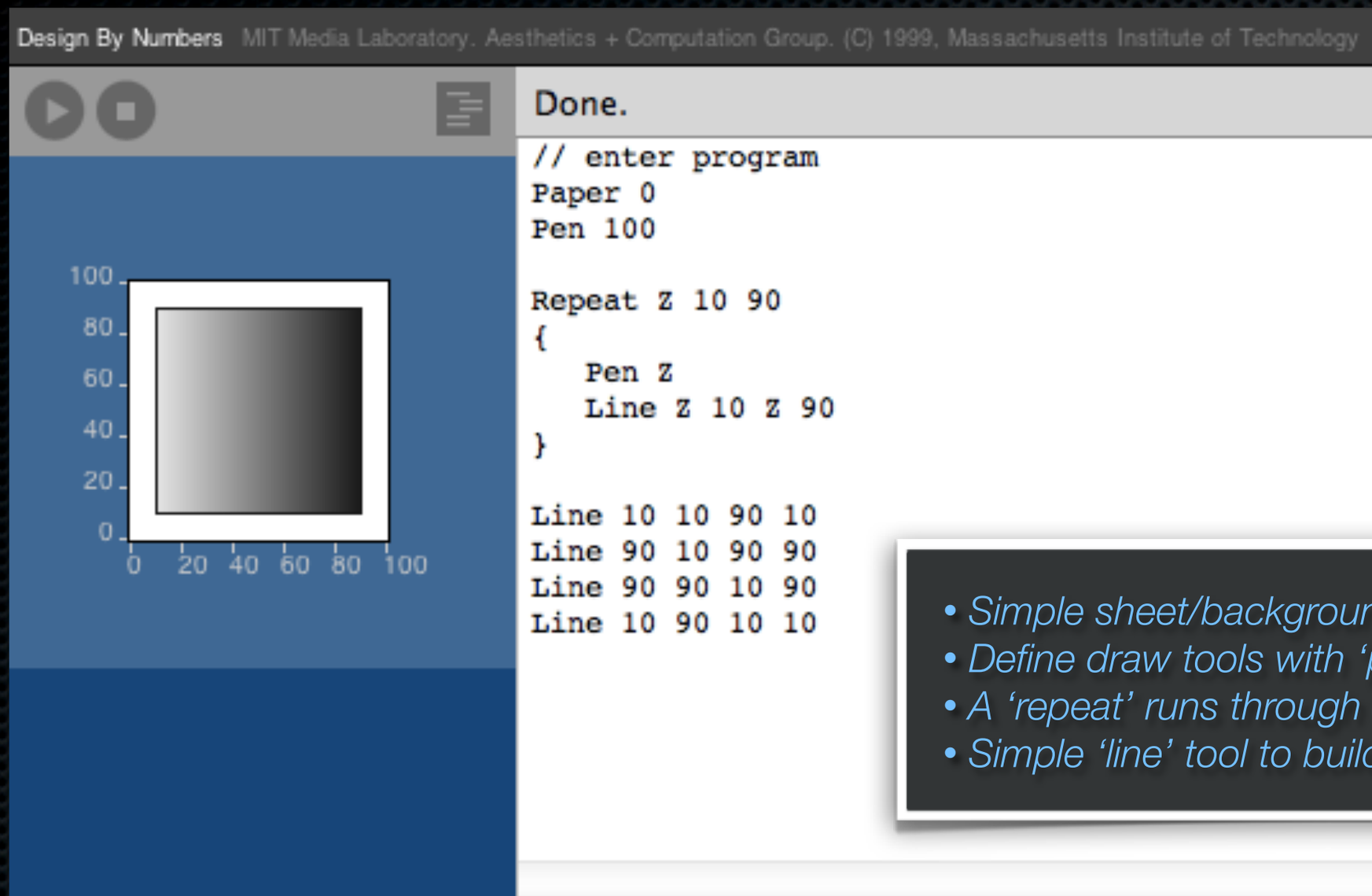
a timeline of development

Design by Numbers

- **Developed** in 1999
- **Designers**:
  Jon Maede and Students
  (MIT Media Lab)
- **Composed of**:
  - Programming Language
    (based primarily upon JAVA, but influenced by LISP, LOGO, C, and BASIC)
  - In Browser IDE
- **Intent**:
  Enable users to get right into computer programming



> *"[John Maede]* views the computer not as a substitute for brush and paint but as an artistic medium in its own right."
> *- MIT Press*



- *Simple sheet/background definition with 'paper'*
- *Define draw tools with 'pen' (followed by shade)*
- *A 'repeat' runs through drawing each line*
- *Simple 'line' tool to build shapes*

Design by Numbers IDE

Simplistic and straightforward interface

Processing

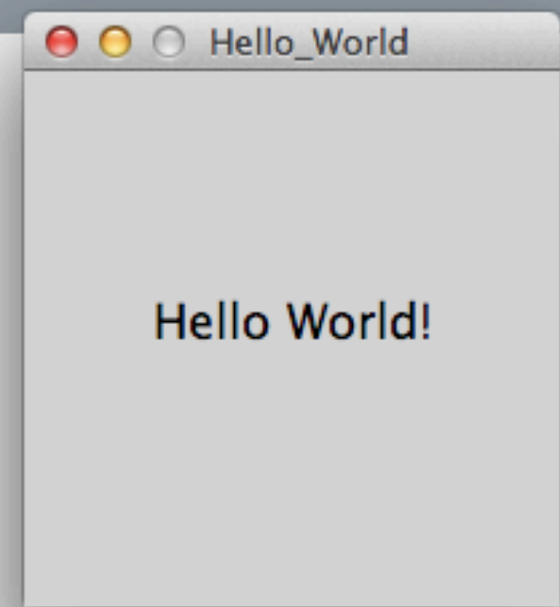
"Processing is an open source programming language and environment for people who want to create images, animations, and interactions."
- processing.org

- **Developed** in 2001
- **Designers**:
  Casey Reas and Benjamin Fry
  (formerly of MIT Media Lab, under Maede)
- **Composed of**:
  - Programming Language
    (based on JAVA)
  - IDE
- **Influences:**
  Design By Numbers, Java, OpenGL, PostScript, C
- **Intent**:
  Teach the basics of computer programming

STANDARD

Hello_World

```
// Hello World in Processing

void setup() {
 // define the window size & enable anti-aliasing
 size(200 , 200);
 smooth();
 // Set "ink" color, font, and alignment for rendering text.
 fill(1);  // Black
 // setup the font (system default sans serif)
 textFont(createFont("SansSerif",18));
 textAlign(CENTER);
 noLoop();  // draw() executes only once
}

void draw() {
 // Draw text to screen using the previously set font.
 text("Hello World!", width/2, height/2);
}
```



- *setup() defines the initial state and initial properties*
- *size() draws the window and defines the active region*
- *text() is where you enter text to be displayed and general location*

Processing IDE

Simple interface reminiscent of scripting languages and a bit of C

Wiring

- **Developed** in 2003
- **Designer**:
  Hernando Barragán
  (formerly of Interactive Design Institute Ivrea)
- **Composed of**:
  - Programming Language
    (C++ Libraries)
  - Microcontroller
  - IDE
- **Influences**:
  Based upon the *Processing* IDE and Programming Language
- **Intent**:
  Allow designers and artists to rapidly construct hardware centered projects

Arduino

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments."
- arduino.cc

- **Developed** in 2005
- **Designed by**:

  Massimo Banzi, David Cuartielles, Dave Mellis, Gianluca Martino with Nicholas Zambetti

  (Developed for an interactive design class overseen by Banzi, Barragáns' thesis advisor)
- **Composed of**:
  - Programming Language

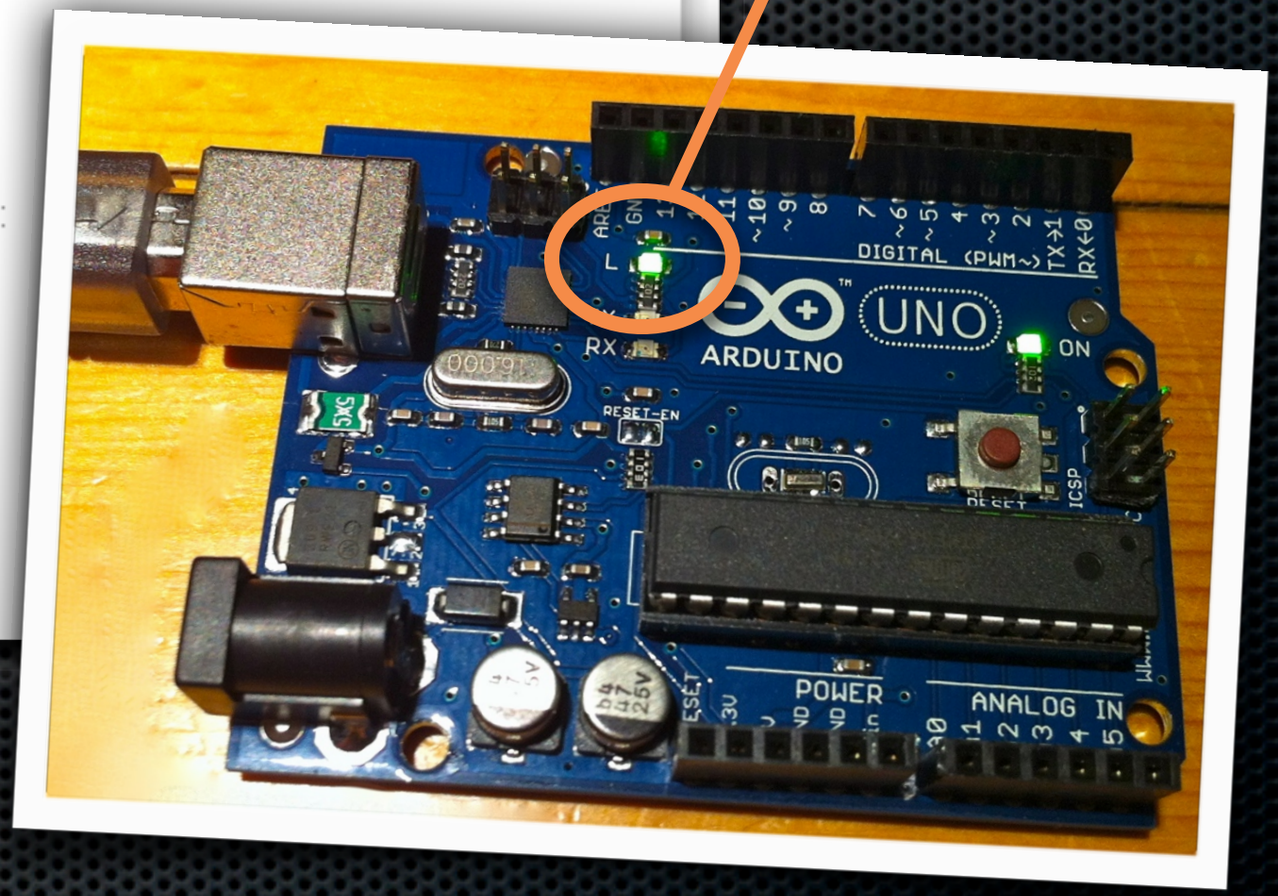
    (Wiring, C++ Libraries and feal)
  - Microcontroller
  - IDE

    (Processing, Java)
- **Influences**:

  Processing Development Environment and *Wiring* Programming Language

blink_short

```
/*
  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(100);               // wait for 0.1 second
  digitalWrite(13, LOW);    // set the LED off
  delay(100);               // wait for 0.1 second
}
```

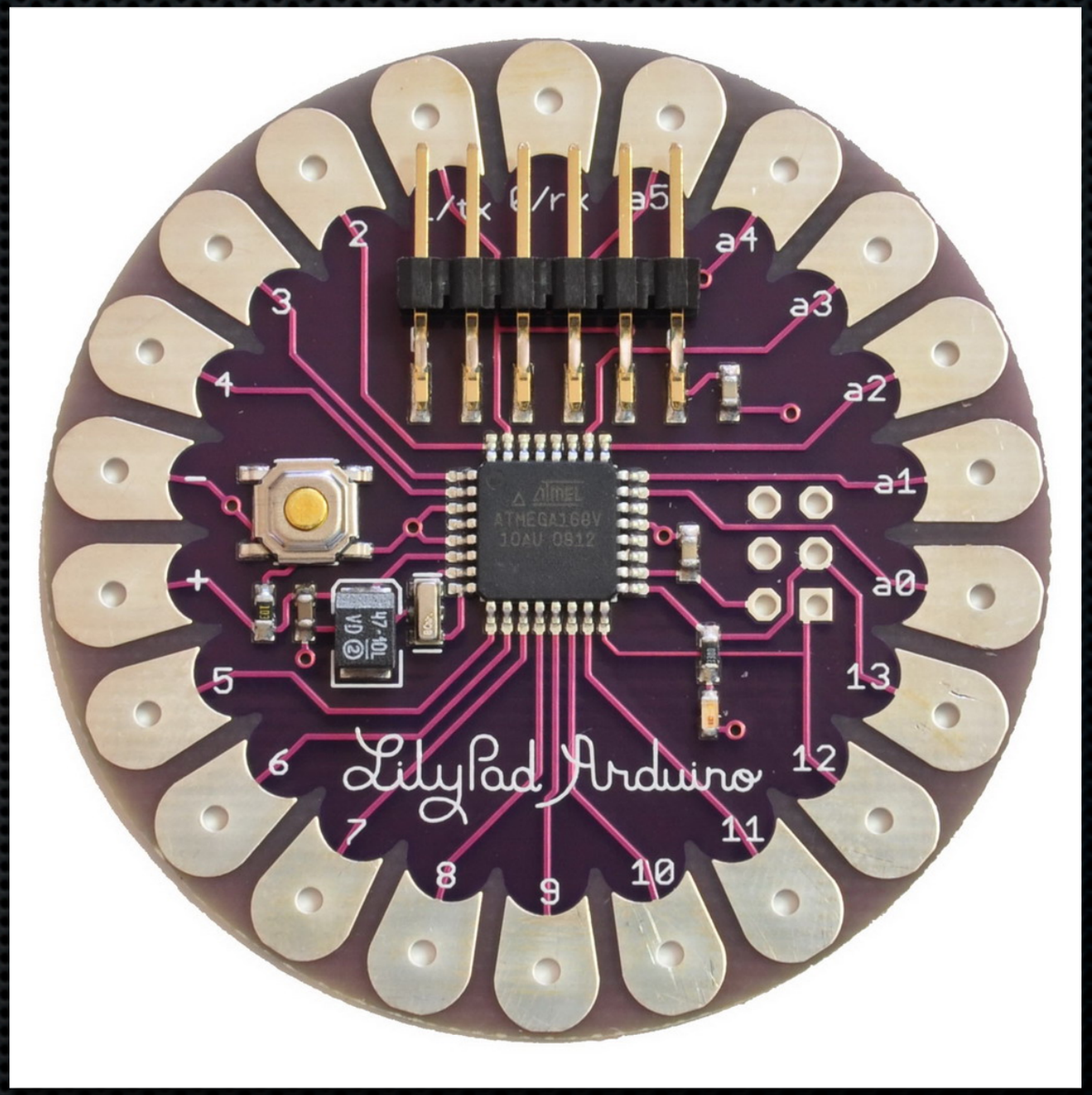


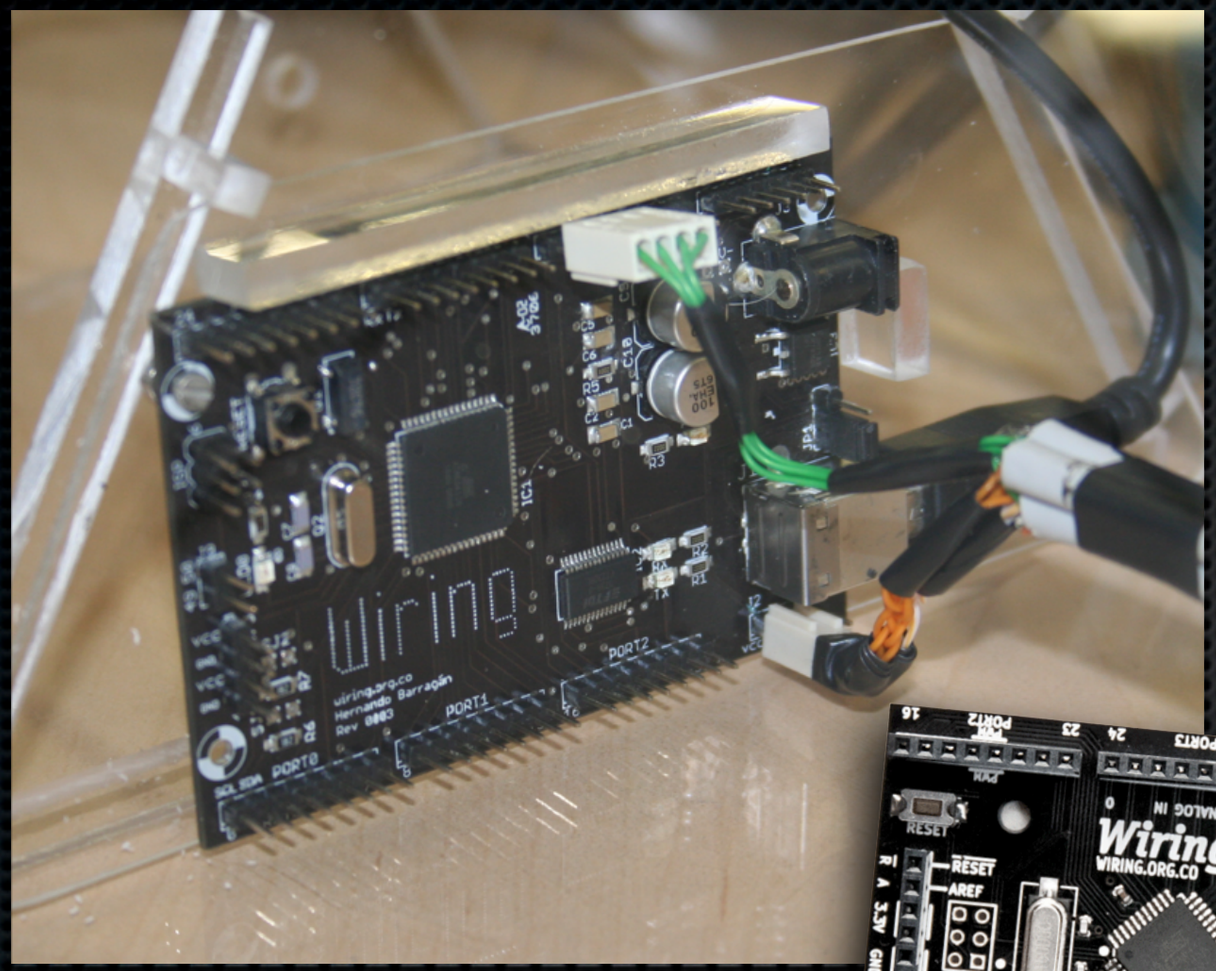
Arduino IDE

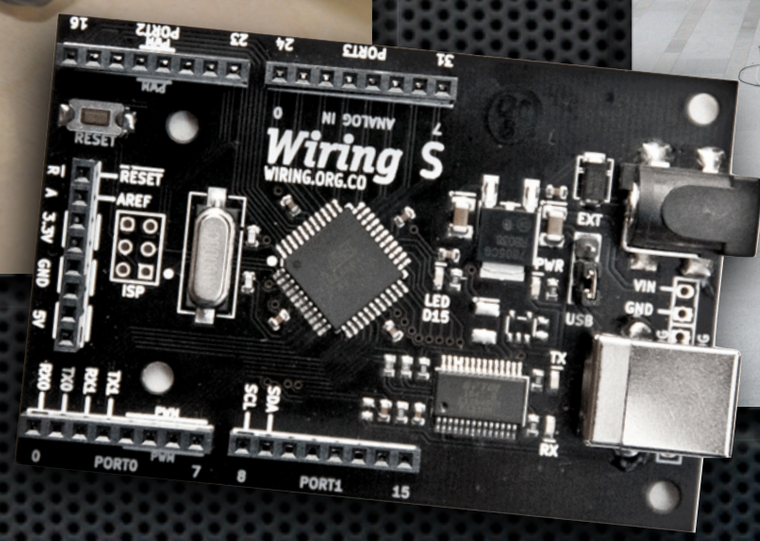
Pretty simple! Initialize pin 13 as output. Set pin 13 HIGH then wait. Set pin 13 LOW then wait.

Hardware

Why Arduino?

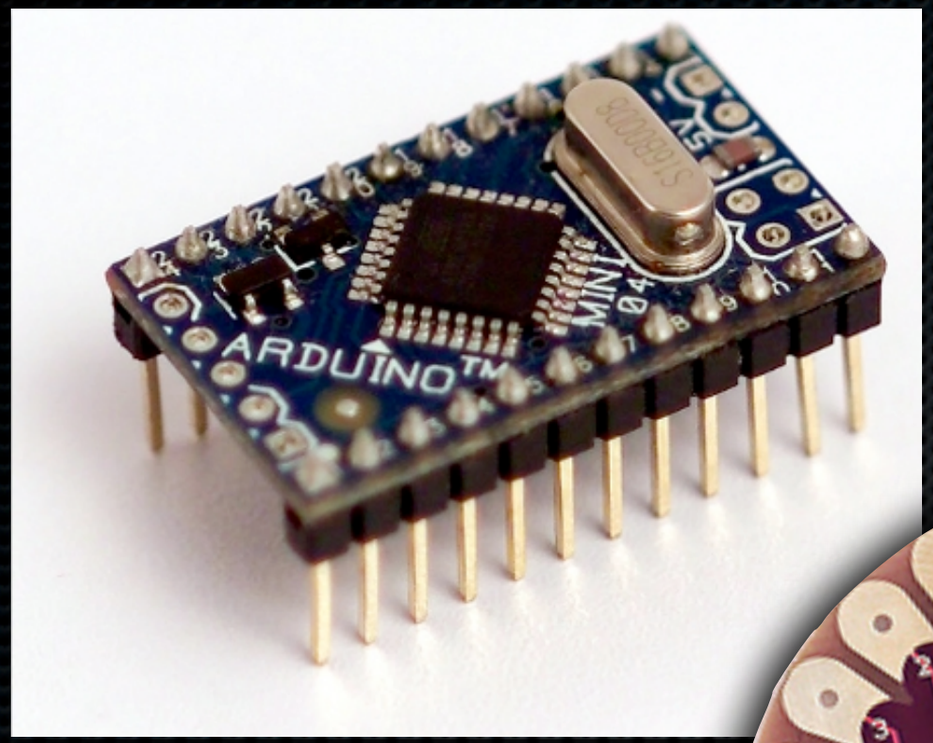




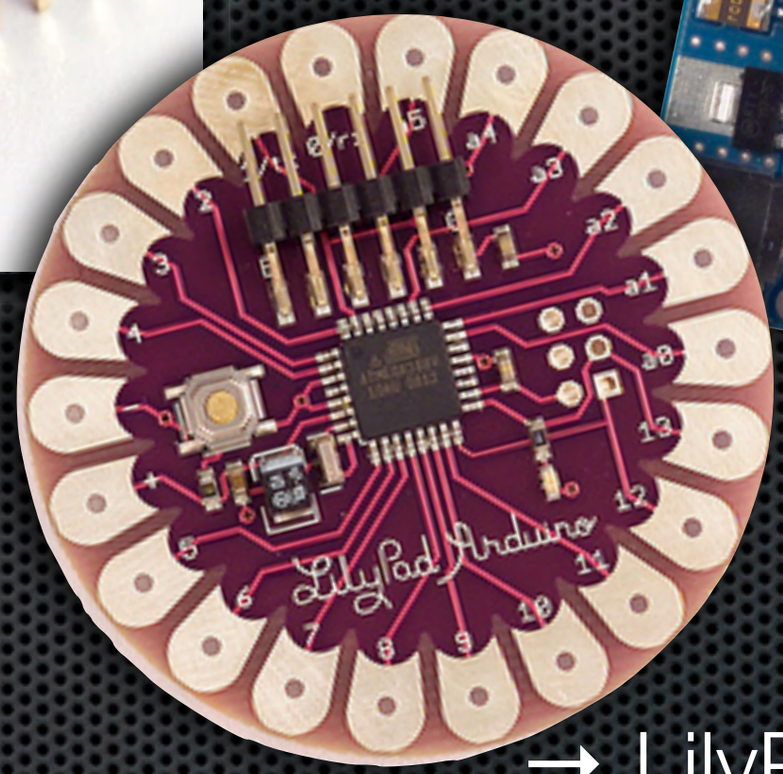


Wiring Hardware

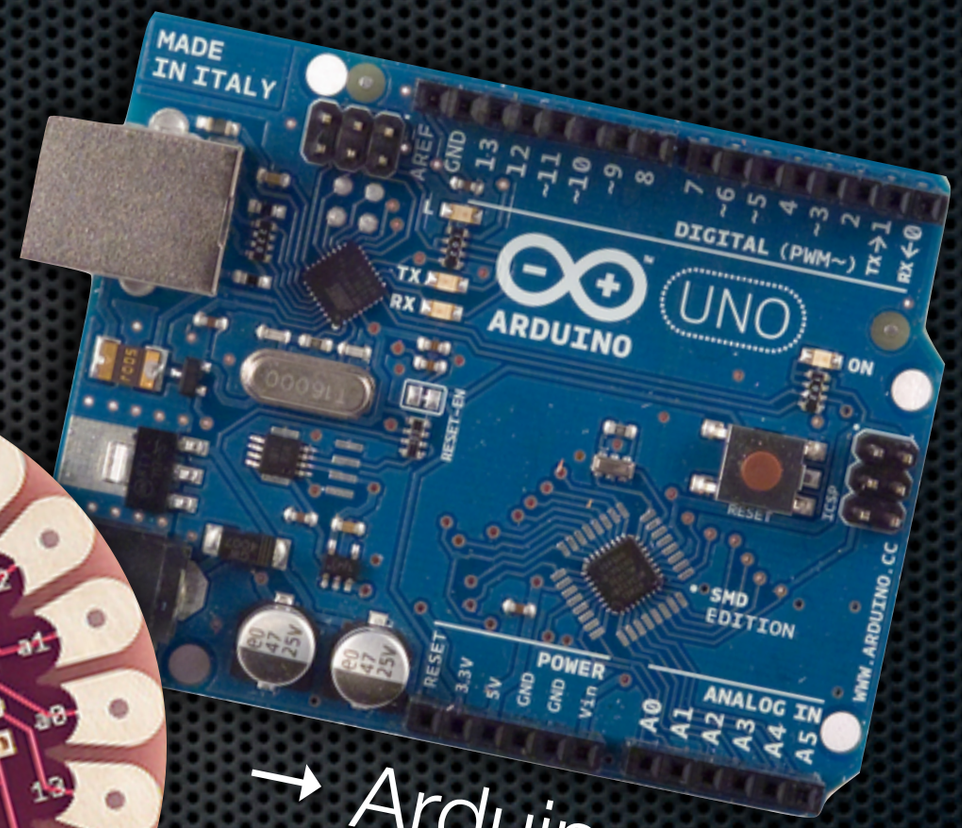
Front and center is the 'Wiring S' board. Behind to the left and right is a design piece that visualizes ambient noise in reactive way.



→ Arduino Mini



→ LilyPad Arduino



→ Arduino UNO

Arduino Hardware

Arduino has many physical forms and there are new ones all the time. On top of that, companies and individuals often construct Arduino specific add-on hardware.

	Wiring hardware V1: atmega128 / Wiring hardware V1.1 Sparkfun: atmega1281 / atmega2561	Wiring S: atmega644p	BasicX	BasicStamp	PIC16F876	atmega8 / atmega168 / atmega328 (Arduino)
Digital I/O Pins	54	32	16	15	22	11
Memory	128K / 128K / 256K	64K	32K	2K	14K	8K / 16K / 32K
Analog inputs	8	8	8	n/a	n/a	6
External Interrupts	8	3	n/a	n/a	1	2
Hardware serial ports	2	2	1	1	1	1
USB	yes	yes	no	no	no	yes
Power	External 7-12V generic adapter or through the USB when connected to a computer	External 7-12V generic adapter or through the USB when connected to a computer	Requires power regulator circuit + adapter	Requires power regulator circuit + adapter	Requires power regulator circuit + adapter	External 7-12V generic adapter or through the USB when connected to a computer
PWM (analog) outputs	6	6	n/a	n/a	2	3
Programming language	C++ with Wiring Framework	C++ with Wiring Framework	basic	basic	basic	C++ with Wiring Framework

Hardware Comparison

Arduino is based upon the Wiring programming language, the main differences appear in hardware

Software

A more in-depth look at the Arduino methods

An Arduino Program

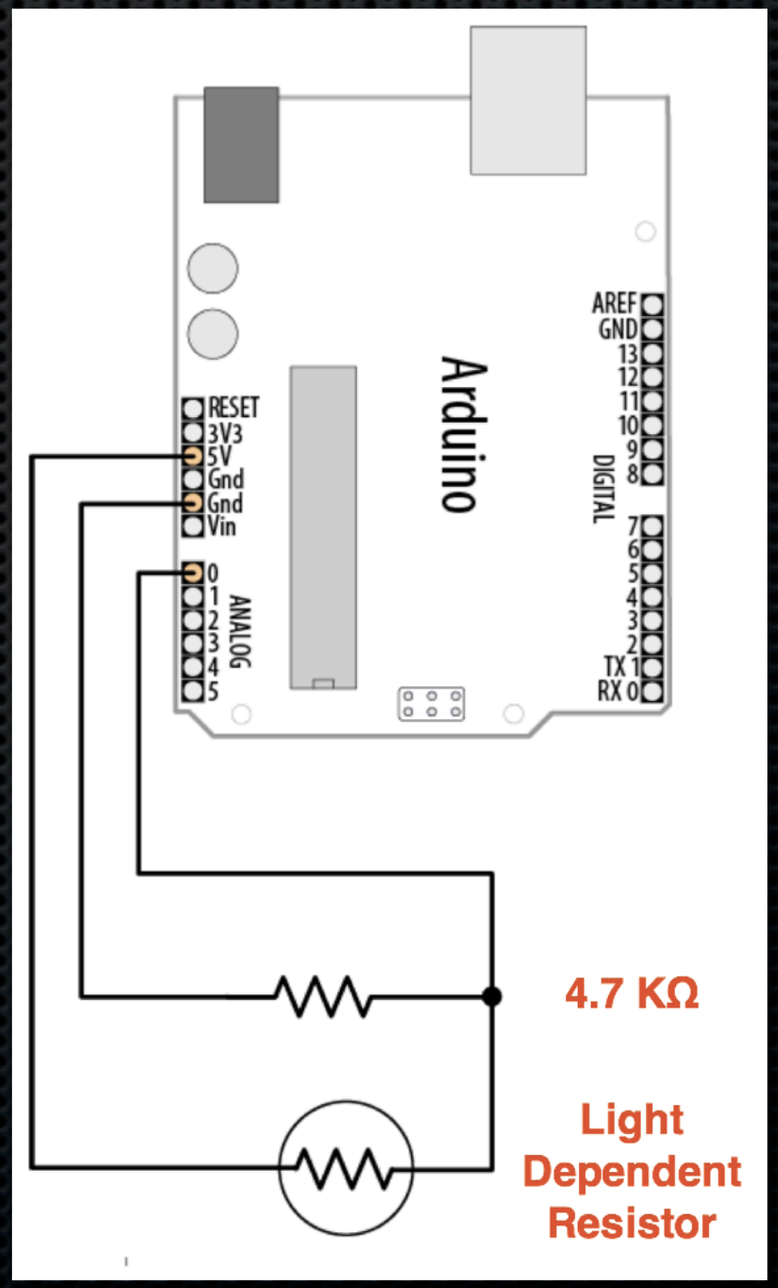
- Just as in *Processing*, **setup()** initializes involved entities.
  - Unlike *Processing*, these entities are now physical objects, such as:
    - input or output pins
    - initial state.
  - We'll see an example of initial state in **setup()** later.
- **loop()** is the next main part of any program. Anything that will update or persist will be declared in **loop()**.

blink_short §

```
/*
 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(100);               // wait for 0.1 second
  digitalWrite(13, LOW);    // set the LED off
  delay(100);               // wait for 0.1 second
}
```

Lets Build a Program - 1

- A blinking light's great, but lets add some physical interface.
- We're going to add a light dependent resistor (LDR) and change the speed with which the LED blinks based upon the LDR.



Lets Build a Program - 2

- First begin by defining your inputs and outputs
- Next we set our constants, which in this case will be our max/min spacing between blinks

```
/*
 Blink Light Sense
 Turns the LED on and off based upon the light level (faster)
 */

const int ledPin = 13;                    // LED connected to digital pin 13
const int sensorPin = 0;                  // connected sensor to analog input 0

// the next two lines set the min and max delay between blinks
const int minDuration = 100;              // minimum wait between blinks
const int maxDuration = 1000;             // maximum wait between blinks
```

Lets Build a Program - 3

```
void setup()
{
  pinMode(ledPin, OUTPUT);          // enable output on the led pin
}
```

- As mentioned before, **setup()** defines initial or persisting state. In this case **ledPin** will be an output throughout the duration of the program. So we use pinMode to define it as an **OUTPUT**
- Note: all caps in *Arduino* signifies a built in method or constant (such as **HIGH** or **LOW**)

Lets Build a Program - 4

```
void loop()
{
  int rate = analogRead(sensorPin);  // read the analog input

  // the next line scales the blink rate between the min and max values
  rate = map(rate,200,800,minDuration,maxDuration); // convert to blink rate

  if (rate <minDuration)
      rate = minDuration;              // constrain the value
```

- We set **rate** equal the **sensorPin** from before, using the built in function **analogRead()**
- **rate** is now being redefined partly by itself as a previous value, as well as by the constraints we set and conversion factors.
- The **if** statement just bounds the blinking rate

Lets Build a Program - 5

```
  digitalWrite(ledPin, HIGH);          // set the LED on
  delay(rate);                         // wait duration dependant on light level
  digitalWrite(ledPin, LOW);           // set the LED off
  delay(rate);
}
```

- This part should be familiar from the blinking example from earlier.
- All we're doing now is setting the output pin, **ledPin** to **HIGH** OR **LOW**, and **delay**ing by the **rate**

Lets Build a Program - 6

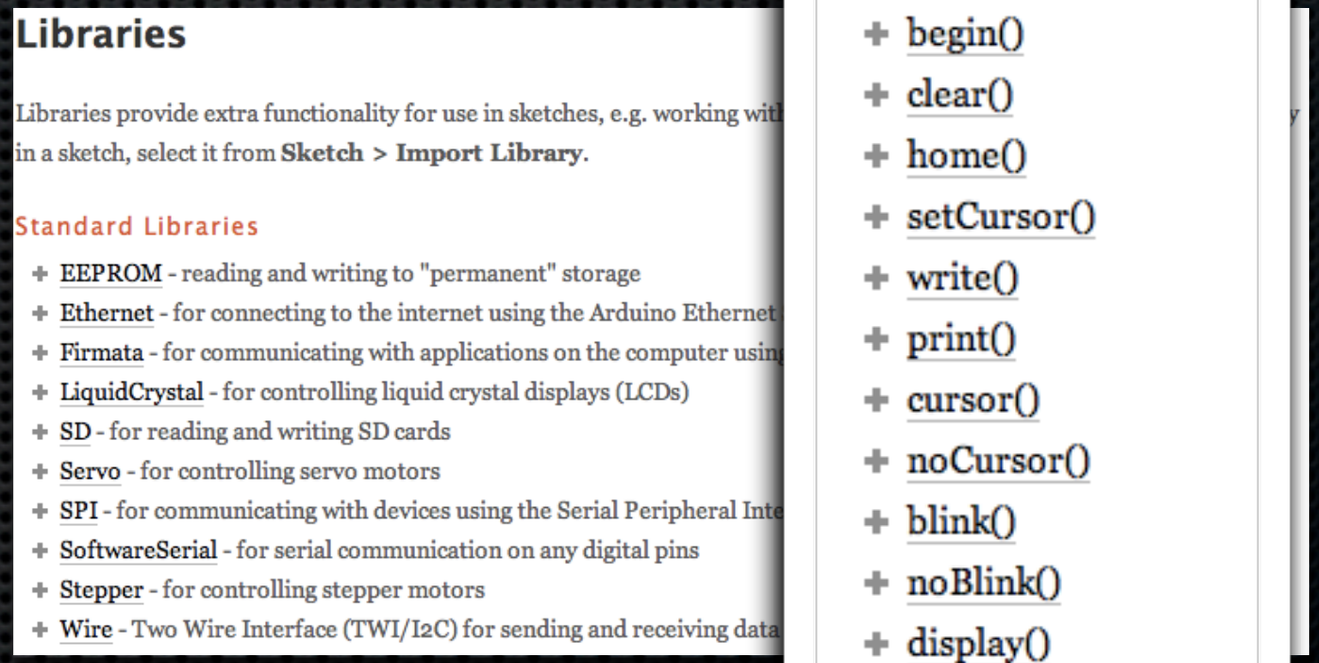
- And now the program is complete
- All that's left to do is to check by hitting the Verify button
- Then, when you've made sure all is well, hit the upload button, and done!

blink_light_sense_faster §

```
/*
 Blink Light Sense
 Turns the LED on and off based upon the light level (faster)
 */

const int ledPin = 13;                  // LED connected to digital pin 13
const int sensorPin = 0;                // connected sensor to analog input 0

// the next two lines set the min and max delay between blinks
const int minDuration = 100;            // minimum wait between blinks
const int maxDuration = 1000;           // maximum wait between blinks

void setup()
{
  pinMode(ledPin, OUTPUT);              // enable output on the led pin
}

void loop()
{
  int rate = analogRead(sensorPin);  // read the analog input

  // the next line scales the blink rate between the min and max values
  rate = map(rate,200,800,minDuration,maxDuration); // convert to blink rate

  if (rate <minDuration)
      rate = minDuration;               // constrain the value

  digitalWrite(ledPin, HIGH);           // set the LED on
  delay(rate);                          // wait duration dependant on light level
  digitalWrite(ledPin, LOW);            // set the LED off
  delay(rate);
}
```

Libraries

- Libraries in *Arduino* consist of two things:
  - C++ Source file
  - C++ Header Source file
- When you implement an object of an included library or a custom library, you:
  - create an instance of that object in the Arduino IDE
  - manipulate the given object with the public functions in the objects' class.



Libraries

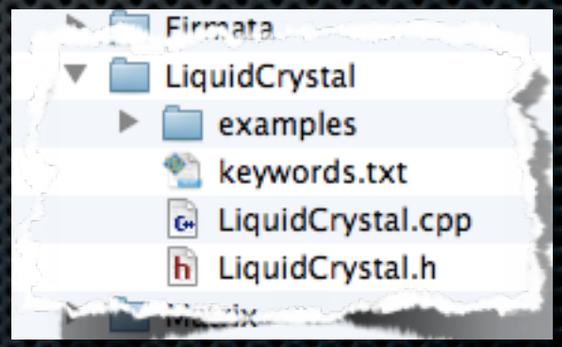
Libraries provide extra functionality for use in sketches, e.g. working wit
in a sketch, select it from **Sketch > Import Library**.

Standard Libraries

- EEPROM - reading and writing to "permanent" storage
- Ethernet - for connecting to the internet using the Arduino Ethernet
- Firmata - for communicating with applications on the computer usin
- LiquidCrystal - for controlling liquid crystal displays (LCDs)
- SD - for reading and writing SD cards
- Servo - for controlling servo motors
- SPI - for communicating with devices using the Serial Peripheral Inte
- SoftwareSerial - for serial communication on any digital pins
- Stepper - for controlling stepper motors
- Wire - Two Wire Interface (TWI/I2C) for sending and receiving data

Function

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()
- createChar()



Library Writing Tutorial

liquid_crystal_Hello_World

```
/*
 LiquidCrystal Library - Hello World

 Library originally added 18 Apr 2008
 by David A. Mellis
 modified 22 Nov 2010
 by Tom Igoe

 This example code is in the public domain.
 http://www.arduino.cc/en/Tutorial/LiquidCrystal
 */

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis()/1000);
}
```

hello world with LCD Library

Note: "hello, world!" on an external LCD only requires 5 lines of code!

LCD Library tie in w/example

- Here are the two functions used from the LCD Library
- **begin(16,2)** initializes a screen that is 16 columns wide and 2 rows tall. And defaults the cursor to space 0 in row 0.
- **setCursor(0,1)** this sets your cursor to begin printing at space one of the second row (row 1).

↓ **begin()**

```
void LiquidCrystal::begin(uint8_t cols, uint8_t lines,
  if (lines > 1) {
    _displayfunction |= LCD_2LINE;
  }
  _numlines = lines;
  _currline = 0;
```

↓ **setCursor()**

```
void LiquidCrystal::setCursor(uint8_t col, uint8_t row)
{
  int row_offsets[] = { 0x00, 0x40, 0x14, 0x54 };
  if ( row > _numlines ) {
    row = _numlines-1;    // we count rows starting w/0
  }

  command(LCD_SETDDRAMADDR | (col + row_offsets[row]));
}
```

Recap

Summing up the Presentation

Arduino Software

- The programing language is identical to that of Wiring, which is in turn based upon C++
- Libraries are written in C++, thus making them easily constructible and readily accessible
- Objects from libraries are instantiated within the Arduino IDE
- The Arduino IDE is identical to that of Processing, which is in turn based upon Java

Arduino Hardware

- Both Wiring and Arduino employ the same language, and the same Atmel architecture
- There's a huge array of hardware and sensors that interface with Arduino either using built in libraries or user built and submitted libraries
- Many form factors and spec ranges of Arduino exist to cater to nearly any project.
- If you need a special form factor, build your own!

Resources

Tutorials, References & where to get Hardware

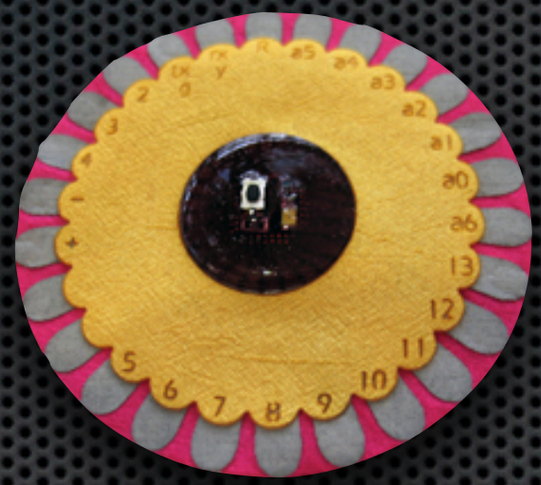
Novel Arduino Hardware

And Arduino Compatible Hardware

(click images for weblinks)

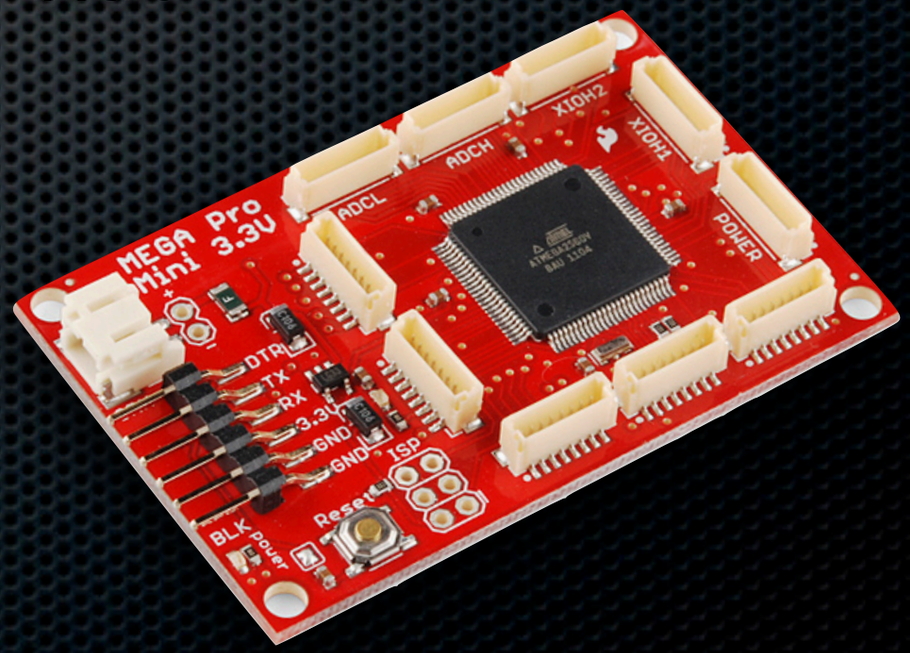
→ LilyPad Arduino

When Leah first built the LilyPad, they were fabric and glue, no PCB. She's recently put out a kit

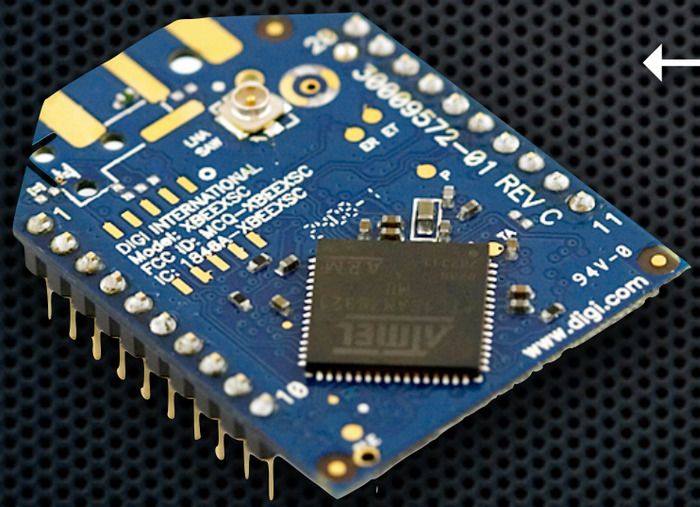


↓ Mega Pro Mini

This board was just released by SparkFun on Nov 3rd. It's Arduino compatible (not official Arduino) and quite a robust little board.



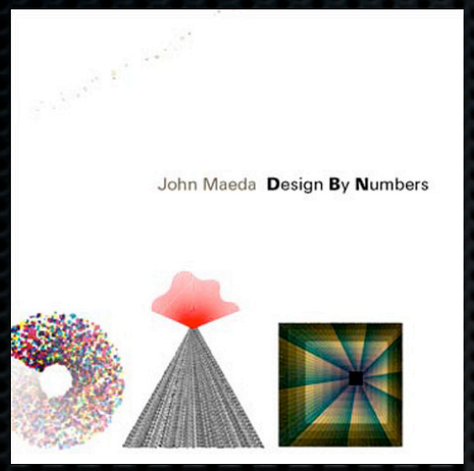
← XBee



This line of wireless (and wired) network adapters has it's own Arduino Library and can range as far as 15 miles

Resources

Logos link to homepages, any link will end show & jump to web



Download

Tutorials

Online IDE

Download

Tutorials

Download

Tutorials

Hardware

Buy

Download

Tutorials

Hardware

Buy