Introduction to Arduino

Mobile Computing, aa. 2016/2017
May 12, 2017

Daniele Ronzani - Ph.D student in Computer Science
dronzani@math.unipd.it
What are Microcontrollers

- Very small and simple computers
- Cheap and useful
- Present in every smart system
- **Atmel** is one of the family of microcontrollers
What is the DIY paradigm

- Create something without professional knowledge
- Many fields
  - Programming
  - Education
  - Critical theory
  - Business
  - Electronics
  - ...
The Arduino project

**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.

- An Italian prototyping tool project
- Ready-made electronic products at low costs
- Open Source
- Boards with different performances
- Shields that extend the functions
  - Wi-Fi, GPSR, Ethernet, microSD reader, ...

**Arduino UNO**

*Arduino.cc*

**Shields and sensors**
Some History of Arduino

- Arduino was born in 2005 in Ivrea, Italy
- **IDEA:** less expensive device for controlling interactive electronic projects
  - Rapid prototyping
- Name "Arduino" in honor of Bar di Re Arduino
- Start to produce boards in a small factory
- Arduino was built around the **Wiring** project of Hernando Barragan.
  - Hernando's thesis project at the Interaction Design Institute Ivrea.
  - Remove the deep knowledge barriers

*Massimo Banzi*
The First Prototype
Arduino Today


- Arduino community is mushrooming across the globe
- 80% from Europe and USA
- More than 200 distributors
Some Competitors - 1

- **Lego Mindstorm**
  - Intelligent brick computer
  - Programmed in Brick Logo
  - Strong community
  - Too much expensive (cost: ~ $349)

- **i-cubeX**
  - Objects respond to human actions and environmental parameters
  - Human Interface Devices
  - Easy but expensive

- **Phidgets**
  - Easy-to-assemble sensors and controllers
  - Not open source (proprietary)
  - Requires less hardware and software knowledge
  - ~ 3.5 x the cost of Arduino board
Some Competitors - 2

- **Make Controller Kit**
  - More powerful CPU than Arduino
  - Has quite a lot of "bang for the buck"

- **Raspberry**
  - It is closer to a computer
  - Includes CPU, USB ports, Ethernet, HDMI, ...
  - Includes an own OS (Linux)

- **Teensy**
  - Same Arduino firmware
  - Compatible with Arduino
  - Very small size

- **UDOO**
  - Merges Arduino and Raspberry
  - Powerful prototyping platform
  - Linux or Android OSs
The three key concepts of Arduino Project

**Hardware**

Can sense the environment by sensors, and affects it by controlling lights, motors, and other actuators.

**Environment**

Allows to write code in the Arduino programming language and using the Arduino development environment.

**Community**

It is made up of everyone from and to hobbyists, students, designers and engineers all across the world.
Why should we use Arduino?

- **Simple, academic purposes**

- **Open source**
  - **Hardware & Software**: permits to manufacture the boards and software distribution by anyone
    - Arduino compatible: Canaduino, Freeduino, Linduino, SainSmart, ...
  - GNU Lesser General Public License (LGPL)

- **Opportunistic prototyping**

- **Community**
  - Wiki
  - Forum
  - Tutorials

- **Could be used as an IoT starting point**
  - Physical computing objects
2005: New Generation (NG)
- ATmega168
- First Mini Arduino

2007: Portable and communication
- i-Bluetooth
- Sew-through contacts pads

2009: Nano and LilyPad
- New Arduino Mega with ATmega1280

2010: Rapid development
- Official Arduino Uno

2012: Miniaturization and CPU power enhance
- LilyPad Simple/Snap and Micro
- Arduino Due (32 bit)
Evolution of Arduino

- **2013**: 700,000 official boards were sold
- **2016**: 17 versions of the Arduino board have been commercially produced.
Some Current Arduino Boards

- **UNO**
  - Current official reference of Arduino Boards
  - Most used and documented board

- **Mega**
  - Designed for more complex projects
  - 54 digital I/O pins, 16 analog inputs
  - ATmega2560

- **LilyPad**
  - Designed for e-textiles and wearables projects
  - Can be sewn to fabric and to power supplies

- **Nano**
  - Compact board similar to the UNO
Physical Computing involves the design of interactive objects that can communicate with humans using sensors and actuators controlled by a behaviour implemented as software running inside a microcontroller.
Smart Objects in the Internet Of Things

IoT is a new emerging technology where the things are connected altogether and interact

- Arduino provides the tools
  - Remote processing of data
  - Connection of physical objects to the Internet

- IoT Manifest
  - **Open SW & HW:**
    - Possibility to share works
    - More innovation
  - **Sustainable**
    - Devices should be easily upgraded
  - **Fair**
    - You should have control of your devices
    - Security
Arduino Cloud

- A new platform to make building IoT
- Creation of tools that allow connection and control of device on/by the Internet
- MQTT broker makes connection between each object
Home Control and Automation

Arduino & Android Home remote control
Arduino Usage Examples

Beat Bearing
Arduino Usage Examples

Twitter Lamp
TeleBall (Breakout game)
Other Examples
Arduino Architecture and Components
General Architecture

Main components

- AVR Microcontroller
- Analog and digital I/O pins
- Flash memory
  - Integrated in the microcontroller
- USB port for serial communication
Components: mandatory

Wires

Prototyping board (breadboard)

USB connector

Resistors and leds
Components: I/O devices
Components: communication devices

Wired and Wireless comm. devices

- WiFi
- Bluetooth
- Ethernet
- Infrared
- Zig-Bee
Characteristics - 1

Arduino UNO (ATmega 328)

Technical characteristics

- Clock speed: 16 MHz (Intel 286: 12.5 MHz) - 8-bit
- Flash program memory: 32 KBytes (0.5 used by bootloader)
- SRAM: 2 KBytes
- Input / Output
  - 14 digital input/output pins
  - 6 analog input pins
  - 6 analog output pins (PWM)
Layout of Arduino UNO

1. Analog Reference pin
2. Digital Ground
3. Digital pins 2-13
4. Digital pins 0-1/Seral In/Out
5. Reset Button
6. In-circuit Serial Programmer
7. Analog In pins 0-5
8. Power and Ground pins
9. External Power Supply In
10. USB port
Digital and Analog pins

- **Digital pins have only two values on/off (0/1)**
  - Arduino UNO has 14 digital I/O pins
  - To connect devices that read/produce digital values (switch, LEDs, ...)

- **Analog pins use many states using quantization**
  - Arduino UNO has 6 input analog pins
  - To read analog sensors (temperature, pressure, ...)
  - An A/D converter of 10 bit returns integers from 0 to 1023
Digital and Analog pins

- Analog output pins use PWM technique
  - Arduino UNO has 6 digital pins used as PWM (~)
  - The signal is affected on the input pulse duration

![Pulse Width Modulation Diagram](image)
Programming in Arduino
Life Cycle of a program

- **power ON**
  - **reset**
  - **Global Variables**
    - Declare variables
    - Initialize variables
  - **setup()**
    - Run once at beginning
    - Set pins
  - **loop()**
    - Run the main program
    - Repeatedly, after the setup
Arduino IDE software

- Friendly Multi Platform application written in Java
- Allows you to write programs and upload them to your Arduino board
- Derived by "Processing" IDE
  - Initially developed in 2003 to simplifies electronic projects
- Very simple for novice programmers
  - Simple programming language
  - Bootloader

- Source code of IDE and library are distributed under the free GNU GPLv2 license
Arduino IDE software

- **Text editor** containing syntax highlighting and automatic indentation
- **Toolbar**
- **Text Console**
- **Compiler**
- **Serial monitor** to debug
  - Allows you to read the data that Arduino communicates through the COM serial port

Sketch filled with the two basic Arduino functions: the `setup()` and `loop()`.
Online IDE - Web Editor

```cpp
#include <Servo.h>
Servo Servo;
int tempo[6];
int contatore = 0;
int meta = 0;
int LedAscolto = 7;
int LedSuono = 8;
int metaOld = 0;
bool su = false;

void setup(){
  Serial.begin(9600);
  battezia.attach(6);
  pinMode(LedAscolto, OUTPUT);
  pinMode(LedSuono, OUTPUT);
  meta = millis();
  /* Comment */
}

void loop(){
  digitalWrite(LedAscolto, HIGH);
  digitalWrite(LedSuono, LOW);
  while(contatore<6){
    int toc = analogRead(A0);
    int to = 0;
    Serial.print("Suono numero ");
    Serial.println(contatore);
    Serial.print(" ");
    Serial.println(toc);
    if(toc>60){
      if(contatore==0){
        tempo[contatore] = 0;
      }
    else{
        tempo[contatore-1] = millis() - meta;
        Serial.print(" Intervalle: ");
    }
```
Wiring-based Language

- Open Source computer programming language
- Derived by C/C++ language
  - With some slight simplifications and modifications
  - Includes classical libraries and functions
    - Data types (Integer, float, long, character, ...)
    - Operators (Mathematical, logical, comparison, ...)
    - Control statements (If, switch/case, while, for, ...)
- Offer to the programmer simple access to I/O devices
- Wiring programs are called *sketch*
Some functions

- `pinMode(pin, Input|Output)`  - set pin `ledPin` as an input or output
- `Serial.begin(9600)`  - talk to the computer at 9600 baud rate
  - Some values: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200
- `Serial.print(" ... ")`  - write text on Serial Monitor
- `digitalWrite(pin, HIGH|LOW)`  - set a digital pin high/low
- `digitalRead(pin)`  - read a digital pin's state
- `analogRead(pin)`  - read an analog pin
- `analogWrite(pin, intValue)`  - write an "analog" PWM value
- `delay(milliseconds)`  - wait an amount of time
Arduino's Hello World: LED blinking

/*
Blink
Turn on and off a LED every one second
*/

int ledPin = 13; // LED connected to digital pin 13

void setup()
{
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

// the loop routine runs over and over again forever:
void loop()
{
    digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
Arduino's Hello World: LED blinking

/*
   Blink
   Turn on and off a LED every one second
 */

int ledPin = 13; // LED connected to digital pin 13

void setup()
{
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

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

int ledPin = 13; // LED connected to digital pin 13

void setup()
{
    Serial.begin(9600);
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

// the loop routine runs over and over again forever:
void loop()
{
    digitalWrite(ledPin, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(ledPin, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
Arduino's Hello World: LED blinking

/*
  Blink
  Turn on and off a LED every one second
*/

int ledPin = 13; // LED connected to digital pin 13

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

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

Turn the LED on and off continuously in the loop
LED and Servo control via Android Application
Thank you for the attention

Daniele Ronzani
dronzani@math.unipd.it