Arduino Systems Overview

Arroyo
Spring 2016

What is Arduino?

- The word "Arduino" can mean 3 things

- A physical piece of hardware

- A programming environment

- A community & philosophy

What is Arduino?

- Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer.
- It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.
- Arduino can be used to develop interactive objects, taking inputs from a variety of switches or sensors, and controlling a variety of lights, motors, and other physical outputs.

What is Arduino?

- Arduino projects can be stand-alone, or they can be communicate with software running on your computer (e.g. Flash, Processing, MaxMSP.)
- The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.
- The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

Why Arduino?

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost about $50-$60
- Cross-platform - The Arduino software runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.

Why Arduino?

- Simple, clear programming environment - The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with the look and feel of Arduino

Why Arduino?

- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.

Why Arduino?

- Open source and extensible hardware - The Arduino is based on Atmel's ATMEGA8 and ATMEGA168 microcontrollers. The plans for the modules are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Why Not Arduino?

- There is a tendency to rely exclusively on existing libraries and no one thinks they may have to develop their own modules until they are needed at the end of the semester. Also the examples on the community are all nearly simplistic and do not prepare the IMDL student for system integration. Without a clear understanding of how Arduino's handle software (inside the hood) this is quite an un-surmountable task for many students (especially those who have not had a course in Microprocessors, which is nearly every graduate student enrolled in IMDL)

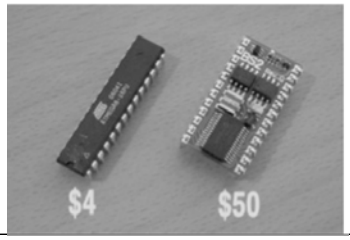
Arduino Philosophy & Community

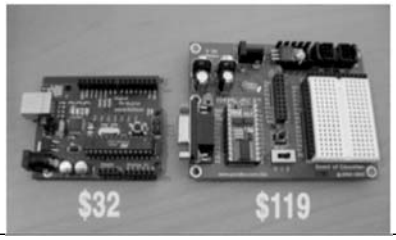
- Open Source Physical Computing Platform
  - "open source hardware"
  - open source: free to inspect & modify
  - physical computing(er), what? ubiquitous computing, pervasive computing, ambient intelligence, calm computing, everyware, spimes, blogjects, smart objects…
- Community-built
  - Examples wiki (the "playground") editable by anyone
  - Forums with lots of helpful people
- Simplistic
  - Simple examples editable by anyone but lacking integration (many sensor systems acting simultaneously)

Arduino Hardware

- Similar to Basic Stamp (if you know of it)
  - but cheaper, faster, & open
- Uses AVR ATmega168 microcontroller chip
  - AVR chips were designed to be used with C language

{ The designers of the AVR purposefully arranged its registers and instruction set so that C programs would compile efficiently on it. This is a big deal, compared to previous microcontrollers where C programs were almost always less efficient than a hand-coded assembly language variant. }





Arduino
Blink External LEDs

IMDL_External_LED

```
#include <LiquidCrystal.h>

/*
  EEL-5666 Example 1: Use of Digital Output Pins. Turn on/off external
  LEDs connected to Pins 11 & 12 alternatively & the internal LED
  on Pin 13 on/off. Add a delay in each state of 1 sec.
  Arroyo January 11 2016
 */

int extLED1 = 11;   // External LED connected to digital pin 11
int extLED2 = 12;   // External LED connected to digital pin 12
int intLED  = 13;   // Internal LED connected to digital pin 13

void setup( ) {
  pinMode(extLED1, OUTPUT);    // sets digital pin 12 as output
  pinMode(extLED2, OUTPUT);    // sets digital pin 12 as output
  pinMode(intLED,  OUTPUT);    // sets digital pin 13 as output
}

  int count=0;
void loop( ) {
  if (count < 1) {
    for (int j=0; j<8; j++){
      digitalWrite(13, HIGH);   // set the internal LED on
      digitalWrite(12, LOW);    // set the external LED1 off
      digitalWrite(11, HIGH);   // set the external LED2 on
      delay(1000);               // wait for a 1/2 second
      digitalWrite(13, LOW);    // set the internal LED off
      digitalWrite(12, HIGH);   // set the external LED on
      digitalWrite(11, LOW);    // set the external LED off
      delay(1000);               // wait for a 1/2 second
    }
  digitalWrite(13, HIGH);   // set the internal LED  on
  digitalWrite(12, HIGH);   // set the external LED1 on
  digitalWrite(11, HIGH);   // set the external LED2 on
  }
  count=1;
}
```