Introduction to Arduino

Instructor: Plamena Milusheva

May 15, 2012
Set up: Drivers and IDE

Download the Arduino Software

The open-source Arduino environment makes it easy to write code and upload it to the microcontroller. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, JenO, and other open-source software.

The Arduino Software is provided to you “as is” and we make no express or implied warranties whatsoever with respect to its functionality, operability, or use, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or infringement. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility or likelihood of such damage.

By downloading the software from this page, you agree to the specified terms.

Getting Started with Arduino

Introduction: What Arduino is and why you’d want to use it.

Installation: Step-by-step instructions for setting up the Arduino software and connecting it to an Arduino Uno, Mega 2560, Due, and others.

- Windows
- Mac OS X
- Linux (on the playground wiki)

Environment: Description of the Arduino development environment.

Troubleshooting: Advice on what to do if things don’t work.
Some Arduino applications:

...and so much more!
Microcontroller: The brain of your Arduino

ATmega328p

Miniature computer
- processor
- long term memory (program storage)
- short term memory (calculations)

Input and Output
- digital (binary: High, Low)
- analog (data sequence)
What makes your Arduino Duemilanove board work:

- Runs at 16 MHz
- 32 KB Flash memory
- 2 KB RAM (for calculations)
- 1 KB EEPROM (re-writable read-only memory)
- 14 Digital Input/Output pins
- 6 Analog Output pins
The essentials of the Arduino board
The Arduino IDE

```
I've
Blink

/**

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
```
Basic button controls

VERIFY  NEW  SAVE

UPLOAD  OPEN  SERIAL MONITOR

Arduino IDE interface with buttons for Verify, New, Save, Upload, Open, and Serial Monitor.
Library of Arduino code

Language Reference

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

Structure

+ setup()
+ loop()

Control Structures

+ if
+ if...else
+ for
+ switch case
+ while
+ do...while
+ break
+ continue
+ return
+ goto

Further Syntax

+ ; (semicolon)
+ { (curly braces)

Variables

Constants

+ HIGH | LOW
+ INPUT | OUTPUT
+ true | false
+ integer constants
+ floating point constants

Data Types

+ void
+ boolean
+ char
+ unsigned char
+ byte
+ int
+ unsigned int
+ word
+ long
+ unsigned long

Functions

Digital I/O

+ pinMode()
+ digitalWrite()
+ digitalRead()

Analog I/O

+ analogReference()
+ analogRead()
+ analogWrite() - PWM

Advanced I/O

+ tone()
+ noTone()
+ shiftOut()
+ shiftIn()
+ pulseIn()

Time

+ millis()
Getting your Arduino to do things...

```cpp
void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.println("success!");
}
```

Setup () function:
Runs once at beginning of program execution

Loop () function:
Loops the dynamic elements of the program

*You may need to change the rate of the serial monitor to 57600 in order to get a read.*
Breadboards
Digital Output: Make an LED blink
Digital Output: Code

```cpp
void setup(){
    pinMode(13, OUTPUT);
}

void loop(){
    digitalWrite(13, HIGH);
    delay(500);
    digitalWrite(13, LOW);
    delay(500);
}
```

Play around with the delay length
Create a variable blinking sequence
Add more LEDs
Digital Input: Controlling your LED with a button
Digital Output: Code

```c
void setup(){
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
}

void loop(){
  if (digitalRead(2) == HIGH)
  {
    digitalWrite(13, HIGH);
  }else
  {
    digitalWrite(13, LOW);
  }
}
```

How IF statements work

Switch the result produced by pressing the button

== vs. =
Analog Input: Reading a variable resistor using Serial
Analog Input serial read: Code

```java
void setup() {
    //no setup required for analog inputs
    //need to set up Serial
    Serial.begin(9600);
}

void loop() {
    Serial.println(analogRead(0));
    delay(30);
}
```
Analog Input: Controlling LED with a potentiometer
Analog Input: Code

```c
const int analogInPin=A0;  //analog pin to which the potentiometer is attached
const int analogOutPin=9;  //analog output pin to which the LED is attached

int sensorValue=0;  //value read from the potentiometer
int outputValue=0;  //value output to the analog out pin

void setup(){
}

void loop(){
    //read the analog value
    sensorValue = analogRead(analogInPin);
    //map it to the range of the analog out
    outputValue = map(sensorValue, 0, 1023, 0, 255);
    //change the analog out value
    analogWrite(analogOutPin, outputValue);
    delay(10);
}
```
Expanding the capabilities of the Arduino board

Using external power supplies
- USB vs. DC adapter
- Vin pin vs. 5V pin

Using shields: modular pre-made circuit boards for various functions (touch screen, ethernet, blue tooth, and much more)