Lab-1: Introduction to Arduino

Ahmed Okasha
okasha1st@gmail.com
LAB(1) Objective:

- Understand what is meant by Microcontroller, Microprocessor, Embedded System, Integrated Development Environment (IDE), and Arduino.
- Understand the layout of Arduino UNO board.
- Understand Arduino IDE, and how to write, compile, and upload a code to Arduino.
- Deal with LED using Arduino digital output pins.
- Understand the difference between Pull-up and Pull-down resistors configurations.
- Use push buttons
- Understand switch bouncing problem and how to overcome it by simple coding solution.
What is a Microcontroller?

Basically, any product or device that interacts with its user has a microcontroller buried inside.
What is a Microcontroller?

A microcontroller is a computer. All computers have several things in common:

1. All computers have a CPU (central processing unit) that executes programs.
2. The computer has some RAM (random-access memory) where it can store "variables."
3. And the computer has some input and output devices so it can talk to people. On your desktop machine, the keyboard and mouse are input devices and the monitor and printer are output devices. A hard disk is an I/O device -- it handles both input and output.
What is a Microcontroller?

- But, Microcontrollers are "special purpose computers." that differs from the PC on:
  1. Microcontrollers are "**embedded**" inside some other device (often a consumer product) so that they can control the features or actions of the product.
  2. Microcontrollers are **dedicated** to one task and run one specific program. The program is stored in **ROM** (read-only memory) and generally does not change.
  3. Microcontrollers are often **low-power devices**. A desktop computer is almost always plugged into a wall socket and might consume 50 watts of electricity. A battery-operated microcontroller might consume 50 milliwatts.
What is a Microcontroller?

- A small computer on a single chip
- Containing a processor, memory, and input/output
- Typically "embedded" inside some device that they control
- A microcontroller is often small and low cost
Common Microcontroller brands

- Intel
- Atmel
- Motorola
- Microchip (PIC)

Plus many others
What is Arduino?

- An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components that facilitate programming and incorporation into other circuits.
- It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board.
What is Arduino?

The word “Arduino” can mean 3 things

A physical piece of hardware

A programming environment

A community & philosophy

todbot.com/blog/bionicarduino
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.

- 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.

- Inexpensive

- Cross-platform

- Simple, clear programming environment
A library is a big collection of procedures, where all the procedures are related! If you, say, want to control a motor, you may want to find a Motor Control Library: a collection of procedures that have already been written for you that you can use without having to do the dirty work of learning the nuances of motors.
Examples of Arduino Projects
Arduino UNO board

- Digital Ground
- Digital I/O Pins (2-13)
- Serial Out (TX)
- Serial In (RX)
- Analog Reference Pin
- USB Plug
- Reset Button
- In-Circuit Serial Programmer
- ATmega328 Microcontroller
- External Power Supply
- Reset Pin
- 3.3 Volt Power Pin
- 5 Volt Power Pin
- Analog In Pins (0-5)
- Voltage In
- Ground Pins
## UNO specs

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>5V</td>
</tr>
<tr>
<td>Input Voltage (recommended)</td>
<td>7-12V</td>
</tr>
<tr>
<td>Input Voltage (limits)</td>
<td>6-20V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>14 (of which 6 provide PWM output)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>6</td>
</tr>
<tr>
<td>DC Current per I/O Pin</td>
<td>40 mA</td>
</tr>
<tr>
<td>DC Current for 3.3V Pin</td>
<td>50 mA</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>32 KB (ATmega328) of which 0.5 KB used by bootloader</td>
</tr>
<tr>
<td>SRAM</td>
<td>2 KB (ATmega328)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>1 KB (ATmega328)</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>16 MHz</td>
</tr>
<tr>
<td>Length</td>
<td>68.6 mm</td>
</tr>
<tr>
<td>Width</td>
<td>53.4 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>25 g</td>
</tr>
</tbody>
</table>
Memory Types

Memory types for smart card microcontrollers

- Volatile memory
  - RAM

- Non-volatile memory
  - ROM
  - PROM
  - EPROM
  - EEPROM
  - Flash EEPROM
  - FRAM
Arduino has:

- **Flash memory**: It's a rewritable non-volatile memory. This means that its content will still be there if you turn off the power. It's a bit like the hard disk on the Arduino board. Your program is stored here. Code for writing and retrieving any data structure to EEPROM easily.

- **RAM**: It's like the RAM in your computer. Its content disappears when you turn off the power, but it can be read and written really fast. Every normal variable in your sketch is held in RAM while your sketch runs.

- **EEPROM**: It's an older technology to implement rewritable non-volatile memory. It's normally used to store settings and other parameters between resets.
The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
Arduino Shields

- Shields are boards that can be plugged on top of the Arduino PCB extending its capabilities. The different shields follow the same philosophy as the original toolkit: they are easy to mount, and cheap to produce.
Examples Arduino Shields

- **Arduino Wi-Fi Shield** - This is the Arduino Ethernet Shield sans wires. This shield can get your Arduino connected to a WiFi router, so it can host webpages and scour the Internet.

- **Cellular Shield w/ SM5100B** - Turn your Arduino into a cellular phone! Send SMS text messages, or hook up a microphone and speaker and use it to replace your iPhone.

- **GPS Shield** - GPS isn’t as complicated as you might think. With a GPS Shield, your Arduino will always know where it is.
Getting Started!

- Download & install the Arduino environment (IDE) for Windows, Mac, or Linux. (Latest version: 1.6)
- Extract the ZIP file. (The extracted folder will contain both the Arduino program itself and also the drivers that allow the Arduino to be connected to your computer by a USB cable.
- Connect the board to your computer via the USB cable.
- The power light on the LED will light up and you may get a 'Found New Hardware' message from Windows.
- Ignore this message and cancel any attempts that Windows makes to try and install drivers automatically for you.
Getting Started (cont.)

- Open Device Manager
- Under the section “Other devices” you should see an icon for “unknown device”, write click on it and press update driver software.
- Select the option: “Browse my computer for driver software”.
- Navigate to:
  - arduino-1.0.2-windows\arduino1.0.2\drivers, in the extracted folder.
- You should be done by successfully installing the Arduino driver.
Getting Started (cont.)

- Launch the Arduino IDE
- Select your board (Tools>>board>>UNO)
Arduino IDE

```
// 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
}
```
Status messages

Uploading worked

Wrong serial port selected

Wrong board selected

nerdy cryptic error messages
Using Arduino

- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the "Blink" sketch!

Load "File/Sketchbook/Examples/Digital/Blink"
Our First Program ! Blink

- Now, you will learn how to make the built-in LED blink.
Our First Program ! Blink

- You might notice that your Arduino board's built-in LED already blinks when you connect it to a USB.
- This is because Arduino boards are generally shipped with the 'Blink' sketch preinstalled.
- We will do a simple variation to the program by changing the rate of the blink.
Our First Program! Blink

- In the IDE, select: File >> Examples >> Basics >> Blink
- This is a read-only version, save it as with any other name.

```cpp
/**
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.

 Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure what pin the on-board LED is connected to on your Arduino model, check the documentation at http://arduino.cc

 This example code is in the public domain.

 modified 8 May 2014
 by Scott Fitzgerald
 */

// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}
```
Understand the code!

```cpp
MyBlink

/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

Most Arduinos have an on-board LED you can control. On the Uno and Leonardo, it is attached to digital pin 13. If you're unsure what pin the on-board LED is connected to on your Arduino model, check the documentation at http://arduino.cc

This example code is in the public domain.

modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(13, HIGH);  // turn the LED on (HIGH is the voltage level)
    delay(1000);              // wait for a second
    digitalWrite(13, LOW);   // turn the LED off by making the voltage LOW
    delay(1000);              // wait for a second
}```
Run the code

- Make sure that Arduino is connected to your PC.
- Click Upload and wait until the status is done.
Change the code

- Change the delay, and upload the new version to your Arduino.

- What is the highest rate (minimum delay) that human being can observe?

```c
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);  // turn the LED on (HIGH is the voltage level)
delay(1000);              // wait for a second
digitalWrite(13, LOW);     // turn the LED off by making the voltage LOW
delay(1000);              // wait for a second
}
```
Ex1-Blink using External LED

- Connect a LED to any digital pin on your Arduino, and put a resistance (270 ohm) in series to limit the current passing through the LED.
- Make the necessary code changes.
- Change the resistance value: 470 ohm, 1K, 10K, What is your observation?

Observation: The LED illumination decreases the as larger resistance is used.
Ex1-Blink using External LED
Ex2-LEDs Traffic:

- Design a system to demonstrate a street traffic light system.

  **First state**
  Duration: 6 sec.

  **Second state**
  Duration: 4 sec.

  **Third state**
  Duration: 2 sec.
Ex2-LEDs Traffic:

- Your hardware might look like as:
Digital Input

- Connecting Switch to Arduino

This is the simplest form of input with only two possible states: on or off. This example reads a simple switch or pushbutton connected to pin2. When the switch is closed the input pin will read HIGH and turn on an LED.

Pull-Up Configuration
Pull-Up vs. Pull down:

Note: 5v and Gnd are taken from Arduino

http://www.devreyapimi.com/
Why the need for Pull-up and Pull-down?

- Pull-up and Pull-down resistors are used in electronic logic circuits to ensure that inputs to the Arduino settle at expected logic levels if external devices are disconnected or high-impedance.

- When you have nothing at all connected to an input pin doesn't mean it is a logical zero.
Ex3-Control Led with push button

- Design a system that let the user when PRESS and HOLD a button, a LED illuminate, and when the button released, the LED switch off. (The default of the LED status is off).
Ex4-Change LED status with push button press

- Modify the previous program to let user change the status of the LED when the button is pressed once. (i.e: when the LED is ON, it should go off when button is pressed)

What is the problem you have observed?
Switch Bouncing

- When a mechanical contact, such as pushbutton, switch, user-interface button, limit switch, relay, or contactor contact, is opened or closed, the contact seldom demonstrates a clean transition from one state to another. When a contact is closed or opened, it will close and open (technically speaking "make" and "break"), many times before finally settling in a stable state.

![Graph showing contact bouncing](http://www.edn.com/)
Software de-bouncing solution

if (reading == LOW) {  //Pull up configuration
delay(200);  // 200 ms delay, increase if the output flickrs
if (reading == LOW)
{
    YOUR CODE
}
}
Helpful Tools

- [http://fritzing.org/home/](http://fritzing.org/home/)
  - Creates virtual circuits (Schematic for your real circuits)
  - Has a PCB view to put components on a PCB with autoroute feature.
  - Share your design (if you want)
  - Contain a collection of projects for beginners and advanced users
Helpful Tools

- [http://123d.circuits.io/](http://123d.circuits.io/)
  - Simulates your Arduino code
Home Work!

- Review Arduino Language:

- Read the PDF of reviewing the most important functions and instructions.
References

- http://arduino.cc/
- http://www.ladyada.net/learn/arduino/index.html
- https://learn.adafruit.com/