Project in Arduino and the Internet of Things (236504/604)

Dr. Nir Levy

Software and Systems Development Laboratory
Department of Computer Science, The Technion
Spring 2016

Lecture 1 – Arduino
Course Info – Safety

• For the lectures:
  • All project must be reviewed and approved by the course staff
  • Only approved projects can be connected to a power source
    • That includes the computer’s USB

• Do not connect ANYTHING to high voltage, i.e. 220V
  • Except the relevant teams, and only with the staff’s permission

• No soldering! yet...
  • Safety procedures for soldering will be given at later stage
Course Info – Equipment

• Equipment must be checked-out using the spreadsheet
  • https://docs.google.com/spreadsheets/d/1cc8wE289f3KLLa8MdreufJs42SGStVm4mOljiCw_s8nA/edit?usp=sharing
  • Transferring items between students – must also be reflected in the spreadsheet

• Equipment that needs to be checked-out:
  • Tools (soldering irons, multi-meters, etc.)
  • Arduino boards
  • Modules (Ethernet, WiFi, Bluetooth, Accelerometer, Rangefinder, etc.)
  • Discreet components (ICs, motors, etc.)

• Equipment should be ordered through:
  • https://docs.google.com/spreadsheets/d/1C-RtnTK8Z3C2i3i0lifyCPjwX9l_CE0gL5sbPvd3Y/edit?usp=sharing
Overview of Lecture 1 – Arduino

• Getting started:
  • Introduction and materials
• Programming, circuit building and Fritzing:
  • Digital Output
  • Digital Input
  • High Current & Analog I/O
  • Serial Communication
Getting Started
Dev Boards
Dev Boars Common Features

Computing
- It is mostly the differences between chips that distinguish one board from another

Indicating
- Most boards will have at least a power LED and a software-controlled LED

Resetting
- Raspberry Pi 2 does not have one...

Powering
- Basic boards typically run at 5V or 3.3V and power should be managed carefully

Interacting
- Most boards have at least simple inputs and outputs (I/O) for interfacing with a wealth of signals in the real world. Almost every board can handle basic digital voltages and signals. Many boards can also handle analog voltages, which can be anything from zero volts up to the chip’s power supply voltage
- Advanced boards are basically computers on a chip, they tend to also have a more desktop computer-like, built-in peripheral set that can include HDMI or other video, audio in and out, external memory, USB host, Ethernet, etc.

Communicating
- RS232, USB, Wi-Fi, Bluetooth, Ethernet, etc.

Mounting
The original Raspberry Pi went on sale in 2012 and sold more than 4M.

New Raspberry Pi Zero micro computer is smaller, faster, and more capable — and it costs $5.
Open Source Hardware

Startups start here

- Created in Ivrea, Italy in 2005 by Massimo Banzi & David Cuartielles

- Consists of an open-source hardware board
- 8-bit Atmel AVR microcontroller, or a 32-bit Atmel ARM, or ...
  - USB interface
  - 6 analog input pins
  - 14 digital I/O pins

- Arduino programs are written in C or C++
  - Users need define 2 functions to make a runnable cyclic-executive program:
    - `setup()` : a function run once at the start of a program that can initialize settings
    - `loop()` : a function called repeatedly until the board powers off
Arduino

- *Once there was a King:*
  Arduino of Ivrea, King of Italy (1002-1014)

- *And a local pub...*
Useful Sources of Information

- The official Arduino web site http://arduino.cc/
- SparkFun Electronics https://www.sparkfun.com/
- Adafruit https://www.adafruit.com/
- Instructables http://www.instructables.com/
- YouTube and the Internet – endless number of designs are shared
Board Layout

Arduino Shields

PCB

Built Shield

Inserted Shield
Discrete Components

<table>
<thead>
<tr>
<th>Color</th>
<th>Significant Figures</th>
<th>Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0</td>
<td>±5%</td>
</tr>
<tr>
<td>Orange</td>
<td>2</td>
<td>±1%</td>
</tr>
<tr>
<td>Yellow</td>
<td>4</td>
<td>±1%</td>
</tr>
<tr>
<td>Green</td>
<td>5</td>
<td>±1%</td>
</tr>
<tr>
<td>Blue</td>
<td>6</td>
<td>±0.25%</td>
</tr>
<tr>
<td>Violet</td>
<td>7</td>
<td>±1%</td>
</tr>
<tr>
<td>Grey</td>
<td>8</td>
<td>±1%</td>
</tr>
<tr>
<td>White</td>
<td>9</td>
<td>±1%</td>
</tr>
<tr>
<td>Gold</td>
<td>10</td>
<td>±1%</td>
</tr>
<tr>
<td>Silver</td>
<td>11</td>
<td>±1%</td>
</tr>
<tr>
<td>None</td>
<td>12</td>
<td>±10%</td>
</tr>
</tbody>
</table>

- **Jumper Wire**
  - Various Colors
  - x30

- **LED (5mm)**
  - Light Emitting Diode
  - x10
  - x10
  - x1

- **330Ω Resistor**
  - x25
  - *Actual Size*

- **10KΩ Resistor**
  - x25
  - *Actual Size*

- **Potentiometer**
  - x1

- **Diode (1N4148)**
  - x2
  - *Actual Size*
Discrete Components

- **Photo Resistor**: x1
- **Piezo Element**: x1
- **Temp. Sensor (TMP36)**: x1
- **Transistor (P2N2222AG)**: x2
- **DC Motor**: x1
- **Push Button**: x2
Solderless Breadboard

- Each row (horiz.) of 5 holes are connected.
- Vertical columns – called power bus are connected vertically
Circuit Building and Programming

Exercise 1 – Digital Output
Using the Breadboard to build a simple circuit

- Use the breadboard to wire up a single LED with a 330 Ohm Resistor (Orange-Orange-Brown).

- Note: the longer leg on the LED is the positive leg and the shorter leg is the negative
Fritzing View of Breadboard Circuit

https://github.com/sparkfun/Fritzing_Parts
Our Tools

SW

• Arduino IDE (http://arduino.cc/en/Main/Software)
• Visual Studio Community 2015 (http://www.visualstudio.com/)
  • See https://msdn.microsoft.com/en-us/library/60k1461a.aspx for C++
• Fritzing (http://fritzing.org/home/)

HW

• SparkFun SIK (https://www.sparkfun.com/products/12060)
The Sketch Project

- In Visual Studio, click File > New > Sketch Project
- Enter a name for your project and click OK

Two required functions:

```c
void setup()
{
    // runs once
}

void loop()
{
    // repeats
}
```

Make sure your environment is configured correctly.
Coding

- **Comments**
  /* Blink ...
  */

- **Variables**
  ```
  const int ledPin = 13; // LED connected to digital pin 13
  ```

- **Functions**
  ```
  void setup()
  {
    pinMode(ledPin, OUTPUT); // sets the digital pin as output
  }

  void loop()
  {
    digitalWrite(ledPin, HIGH); // outputs a value on pin 13
    delay(1000); // the Arduino waits for the specified number of milliseconds
    ...
  }
  ```

Blinking a LED

```c
// Exercise 1 - Blinking LED

int led = 13; // LED in pin 13

void setup()
{
    pinMode(led, OUTPUT);
}

void loop()
{
    digitalWrite(led, HIGH);
    delay(1000);
    digitalWrite(led, LOW);
    delay(1000);
}
```
Programming and Circuit Building

Exercise 2 – Digital Input
Digital Sensor (a.k.a Switch) and an Indicator LED
void setup()
{
    ...

    pinMode(ledPin, OUTPUT); // sets the digital pin as output
    pinMode(buttonPin, INPUT); // sets the digital pin as input

    ...
}

void loop()
{
    ...

    buttonState = digitalRead(buttonPin); // read the state of the pushbutton

    ...
}

Exercises:
1. Build and run the project
2. Change the circuit to normally-closed by SW and by HW
Switch and an Indicator LED

```c
/*
 * Button
 * Turns on and off a light emitting diode (LED) connected to digital
 * pin 13, when pressing a pushbutton attached to pin 7.
 */

// constants won't change. They're used here to set pin numbers
const int buttonPin = 7;  // pushbutton pin
cost int ledPin = 13;     // LED pin

// variables will change:
int buttonState = 0;       // variable for reading the pushbutton status

void setup()
{
    // initialize the LED pin as an output:
    pinMode(ledPin, OUTPUT);
    // initialize the pushbutton pin as an input:
    pinMode(buttonPin, INPUT);
}

void loop()
{
    // read the state of the pushbutton value:
    buttonState = digitalRead(buttonPin);
    // check if the pushbutton is pressed. If it is, the buttonState is HIGH
    if (buttonState == HIGH)
    {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else
    {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
}
```

Output: Compiling "Ex2" for "Arduino Uno"
Programming and Circuit Building

Exercise 3 – High Current & Analog I/O
Spinning a Motor

Flyback diode

5 volt

Diode (1N4148)

DC Motor

RedBoard

PIN 9

Resistor (330 ohm)
(Orange-Orange-Brown)

Transistor (P2N2222AG)

GND
(Ground)
Analog Input and Output

• **analogRead(pin)**
  - Reads the value from the specified analog pin (A0-A5)
  - The Arduino board contains a 6 channel A/D converter, mapping input voltages between 0 and 5 volts into integer values between 0 and 1023

• **analogWrite(pin, value)**
  - Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means
  - A call to analogWrite() is on a scale of 0 - 255, e.g.: analogWrite(pin, 255) requests a 100% duty cycle (always on), and analogWrite(pin, 127) is a 50% duty cycle (on half the time)
  - pin – refers to the OUTPUT pin (limited to pins 3, 5, 6, 9, 10, 11.) – denoted by a ~ symbol

• Exercise: Spin the DC motor to half speed and to full, in loop
Speed Control

Voltage Divider

\[ V_{out} = \frac{R_2}{R_1 + R_2} \cdot V_{in} \]

PWM motor speed control using Arduino

- Exercise: build a speed controlled motor project using analog input and output
Programming and Circuit Building

Exercise 4 – Serial Communication
Serial

- Used for communication between the Arduino board and a computer or other devices

- **Serial.begin(speed)**
  - Sets the data rate in bits per second for serial data transmission
  - For communicating with the computer, use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200

- **Serial.print(val, format)**
  - Prints data to the serial port as ASCII text
  - format is optional, can be set to: BIN, OCT, DEC or HEX

- **Serial.println(val, format)**
  - Prints data to the serial port as ASCII text followed by a carriage return character
Serial Communication

- Opens up a serial terminal window
- Initializes the serial communication
- Sets the terminal's baud rate

Exercise: print the motor speed relative to the maximum speed (assume linearity)
Exercise

• Build a project that sweeps the shaft of a servo motor back and forth across 180 degrees, using the Arduino servo library (see http://arduino.cc/en/Reference/Servo)
  • Hint #1: look for it...
  • Hint #2
Servo Exercise


// Sweep - Sweeps the shaft of a servo motor back and forth across 180 degrees.

#include <Servo/Servo.h> // use the Servo library

Servo myservo; // create servo object to control a servo

int pos = 0; // variable to store the servo position

void setup()
{
    myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop()
{
    for (pos = 0; pos < 180; pos += 1) // goes from 0 degrees to 180 degrees
    {
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
    for (pos = 180; pos >= 0; pos -= 1) // goes from 180 degrees to 0 degrees
    {
        myservo.write(pos); // tell servo to go to position in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the position
    }
}