A Primer on the Arduino Microcontroller

This document is a brief explanation of the microcontrollers used in the Mechatronics labs. It briefly describes the main components of the microcontroller, the board on which it is mounted and its programming environment. A short note illustrates the use of breadboards.

1. Introduction to Microcontrollers

A microcontroller is a dedicated, single-chip, special-purpose digital computer which typically takes on the control functions in mechatronic systems.

A microcontroller will typically have a *CPU*, called a *microprocessor*, for processing digital information, *memory* for storing digital information, and *input/output* circuitry for interacting and communicating with other devices. All these components are typically packaged into a single Integrated Circuit – IC – chip. Today, microcontrollers have become an integral part of most mechatronic devices and systems.

In the Mechatronics labs, we use the Arduino family of microcontrollers for a better understanding of the role and operation of microcontrollers. Although developed for mass market electronics, these microcontrollers are similar in concept, function and operation to any industrial microcontroller. We therefore focus this document on the Arduino microcontroller.

2. Arduino Platform

Arduino is an open source hardware and software platform meant for creating accessible electronics projects and components.

2.1. Hardware

On the hardware side, this platform consists of a microcontroller on a circuit board. There is a variety of Arduino boards which differ mostly on the microcontroller chip and the number of input/output pins. The basic functionality is very similar among all of them. The Arduino UNO board which is used in the labs is shown in Figure 1. It incorporates the ATmega 328P microcontroller made by ATMEL.
As an open source platform, there are many third-party Arduino platform type boards available on the market.

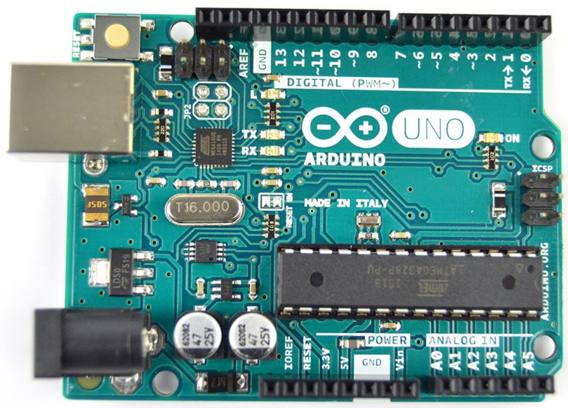
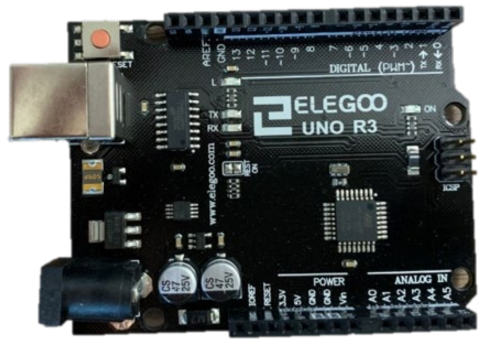


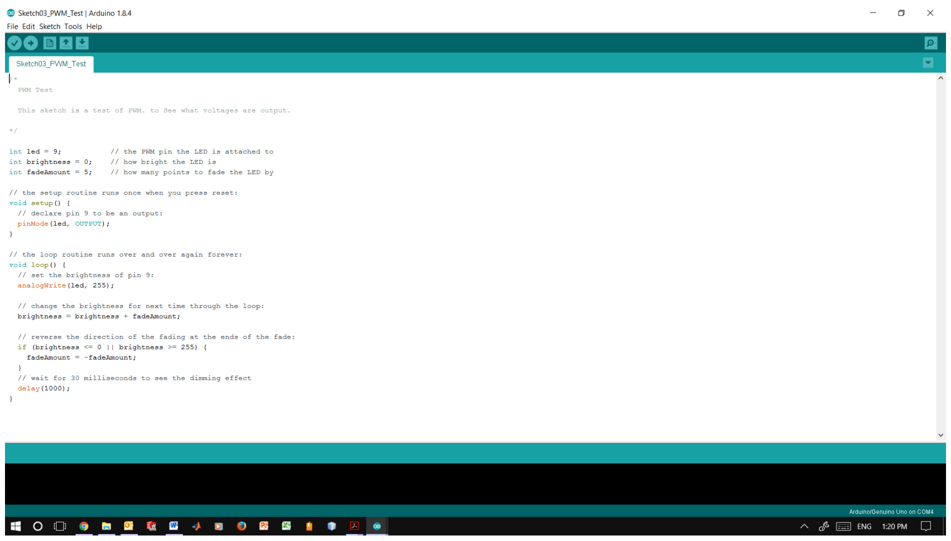
Figure 1- Arduino UNO microcontroller board

Note) The Elegoo Uno R3 we use in this laboratory is the same as Arduino UNO.



2.2. Software

On the software side, the Arduino platform consists of an integrated development environment – IDE – that runs on the computer and is used to upload programmed instructions (called sketches) to the Arduino microcontroller board. The Arduino IDE environment is shown in Figure 2.


Sketch03_PWM_Test | Arduino 1.8.4
File Edit Sketch Tools Help
Sketch03_PWM_Test
Arduino/Genuino Uno on COM4

```
/*
  PWM Test

  This sketch is a test of PWM. to See what voltages are output.

*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, 255);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(1000);
}
```

Figure 2- The Arduino integrated development environment – IDE

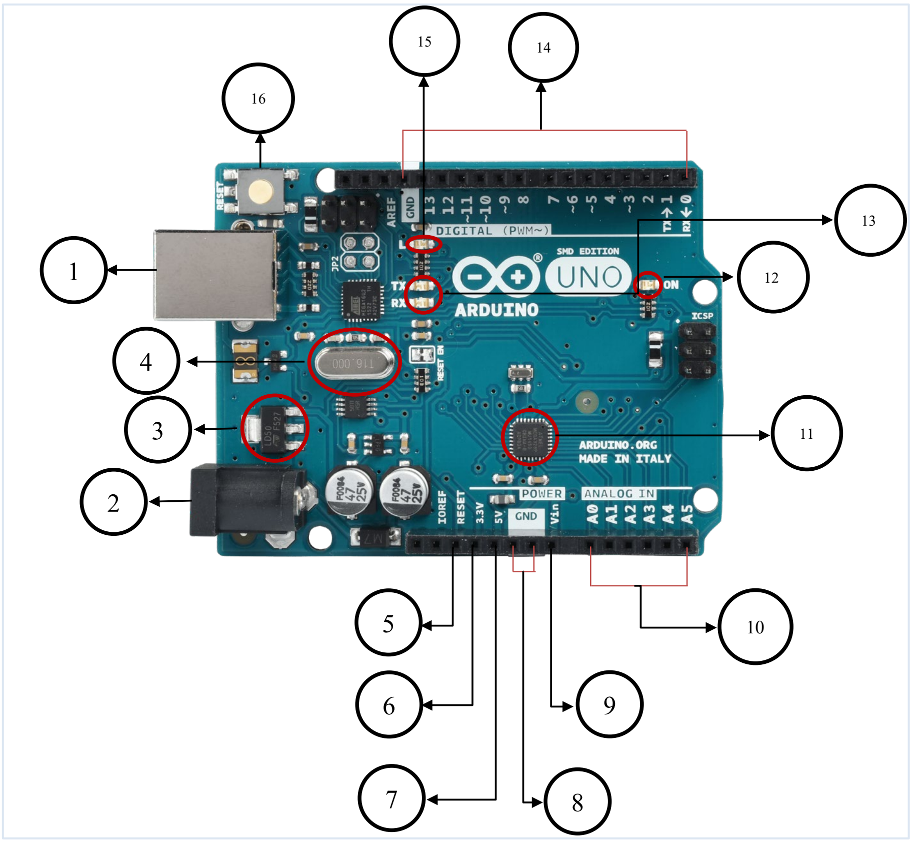


Figure 3- The Arduino UNO board components

3. Arduino UNO Board Description

This section describes the different components on the typical Arduino board.

Power USB (1)

The Arduino board can be powered using a USB cable connection from the computer. This USB connection is also used for serial communication between the computer and the Arduino to upload programs (called sketches, see below) that run on the Arduino.

Power Barrel Jack (2)

The Arduino board can also be powered directly from the AC mains power supply by connecting it to the Barrel Jack.

Voltage Regulator (3)

An electronic (semi-conductor-based) voltage regulator which stabilizes the DC voltage used by the processor and other board elements.

Oscillator Clock (Crystal) (4)

The crystal clock oscillator is an electronic circuit developed around the oscillations of a piezoelectric crystal; hence the abbreviated name crystal. This generates a square wave with a frequency of 16MHz which can become the basis of time calculation on the Arduino.

Reset and Power Pins (5-9)

Arduino board has several “pins”; these are the places on the board to which wires are connected in order to construct circuits and read and write signals to devices. There are several types of pins for different functions.

- Reset Pin (5)
  This pin can be used to reset the Arduino board, i.e., re-start the program (sketch) running on the microcontroller from the beginning. An external push button can be connected to this pin to reset the program. The same function can be performed with the included push button (16).
- 3.3V (6)
  When the board is powered via the USB connection, this pin supplies 3.3 volts of regulated electric power to devices and circuits connected to it.
- 5V (7)
  When the board is powered via the USB connection, this pin supplies 5 volts of regulated electric power to devices and circuits connected to it. Most of the components built and sold for use with Arduino operate at these voltages.
- GND (8)
  These pins can be used to ground the circuits built with the above voltage sources. The GND pin on top of the board performs the same function.
- Vin (9)
  This is the input voltage pin to the Arduino when it is powered from an external power source. Voltage is supplied through this pin, or, if using the power jack, voltage is accessed through this pin. The recommended external input voltage to the UNO is 7-12 Volts and should not exceed 20 Volts.

Analog In Pins (10)

Pins labeled A0 to A5 on the board are the analog inputs to the Arduino. These pins read signals from analog sensors (potentiometer, photoresistor in lab 1) and convert it (analog to digital conversion – ADC) to a digital signal that the microcontroller can use.

Microcontroller (11)

The microcontroller is the processing brain of the board. The Arduino UNO uses the ATmega328P microcontroller (the microcontroller on the boards in the lab look different from the picture, but it is the same microcontroller).

Power LED indicator (12)

This tiny LED lights up when the board is powered.

TX and RX LEDs (13)

TX (transmit) and RX (receive) LEDs flash when respectively sending and receiving serial data from the computer. The symbols RX and TX also appear next to the Digital pins 0 and 1. These two pins are powered when data transmission occurs.

Digital Input and Output (14)

Pins 0 through 13 can be configured as digital inputs and/or outputs. Digital input may represent the pressing of a push button, for instance. As an example, digital output may be used to power an LED. These pins read or are assigned digital values (0 or 1, or in Arduino terms LOW or HIGH). Six of the digital pins are labeled with a "~", indicating that these pins can be used as Pulse Width Modulation (PWM) output. PWM is a technique to approximate analog signals from a digital source by manipulating the frequency (rate) at which the digital signal is turned on and off.

LED (15)

Another tiny LED is connected to PIN 13 of the digital pins. This LED lights up when pin 13 is powered.

Reset Push Button (16)

This button resets the board, re-starting the program (sketch) that is being run on the Arduino.

4. Processing and Memory

The ATmega 328 has an 8-bit CPU. This means that the CPU can handle arithmetic and logic that is 8 bits wide in one instruction. In decimal terms, this translates into numbers between 0 and 255.

There are three types of memory on the ATmega 328.

- Flash memory is where programs are stored. There are 32kilobytes of flash memory on the Arduino.
- SRAM (static random access memory) is where an Arduino program creates variables it needs when running. There are 2kbytes of SRAM.
- EEPROM (Electrically erasable programmable read only memory) is permanent or long term storage space that can be modified occasionally. A life cycle of read/write operations is specified. There are 1kbytes of EEPROM on the ATmega 328.
- SRAM is volatile. It is erased when power is disconnected from the board. Flash memory and EEPROM are non-volatile. They remain even after power off. This means that when Arduino is powered on, it runs the program that was on it last.

5. Arduino Integrated Development Environment (IDE)

This section is a short description of the Arduino software and the basic structure of the programs. The Arduino software is open-source, free to use and expanded upon by libraries. The programs are based on an intuitive simplified version of the C programming language. Programs are called sketches. These are written in a text editor and are saved with the file extension .ino.

5.1. Segments of IDE

The different segments of the IDE are shown in Figure 4. The toolbar on the top contains File, Edit, Sketch, Tools and Help drop down menus. File and Edit menus are similar to most applications. Saved sketches are organized in a sketchbook. This is accessed from the File drop down menu.

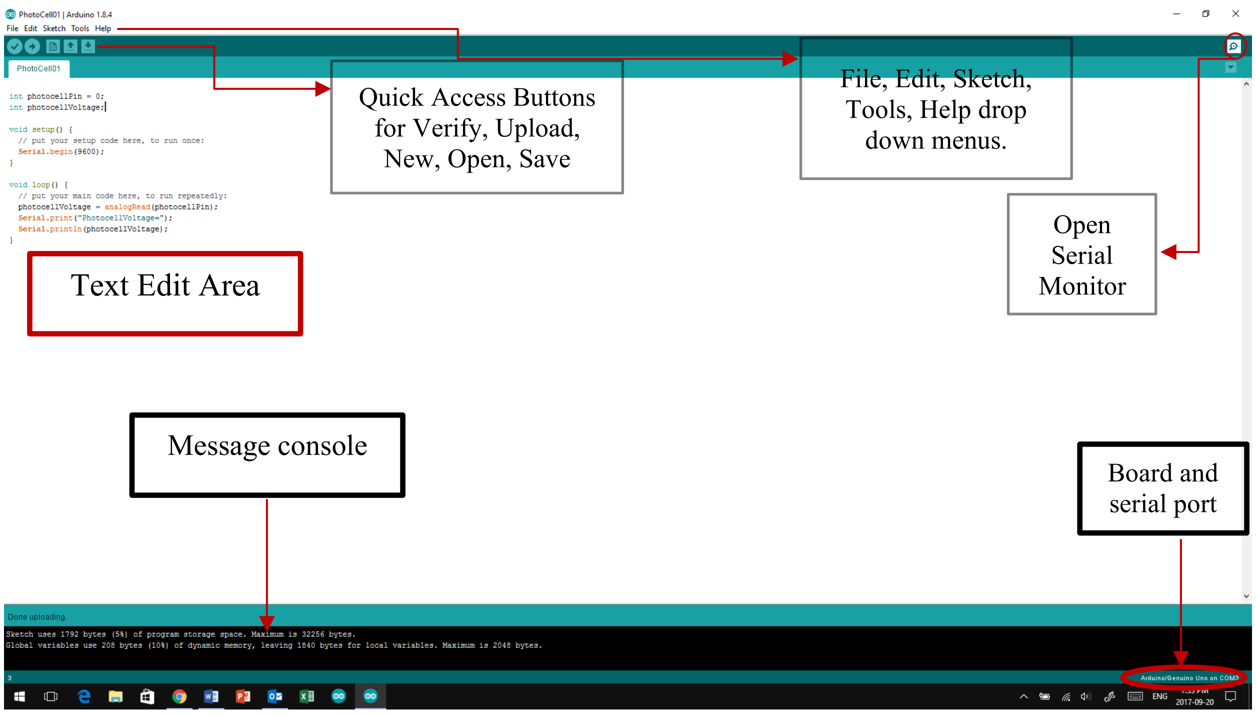
The Sketch drop down menu allows users to Verify/Compile programs, Upload programs, Export a .hex compiled version of the program, Open the current Sketch folder where sketches are stored, etc.

The Tools drop down menu allows users to Auto Format, Archive the sketch, open the Serial Monitor for serial communication, set Board type, select Port, etc.

The Quick Access Buttons Area provides quick access to Verify, Upload, New, Open and Save functions on the left and the Serial Monitor on the right.

- Verify: Checks a sketch for errors. Displays memory use for the program and the variables it uses. Also generates a .hex file which is saved in a temporary location to be uploaded to Arduino.
- Upload: Loads the complied file onto the board through the configured port.
- Serial Monitor: Opens the “Serial Monitor” for exchange of data between computer and the board

The message area provides feedback while saving and exporting and also displays errors. The bottom righthand corner of the window displays the configured board and serial port.


PhotoCell01 | Arduino 1.8.4
File Edit Sketch Tools Help
PhotoCell01
Quick Access Buttons for Verify, Upload, New, Open, Save
File, Edit, Sketch, Tools, Help drop down menus.
Open Serial Monitor
Text Edit Area
Message console
Board and serial port
Done uploading.
Arduino/Genuino Uno on COM3

```
int photocellPin = 0;
int photocellVoltage;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  photocellVoltage = analogRead(photocellPin);
  Serial.print("PhotocellVoltage=");
  Serial.println(photocellVoltage);
}
```

```
Sketch uses 1792 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 208 bytes (10%) of dynamic memory, leaving 1840 bytes for local variables. Maximum is 2048 bytes.
```

Figure 4- The sections of the Arduino IDE

5.2. Program Structure

The Arduino IDE programming language is a simplified version of the C programming language. The basic Arduino sketch contains two main functions.

- void setup()

  This is where the set up commands are written. We use this section to set pin modes, initialize variables, start serial communication, start using libraries, etc. Commands are terminated by a semi-colon and are placed within the two curly brackets.
  The setup() function is a void function, meaning that it does not return any values to the program.
  Commands in the setup() area are run only once upon upload.

- void loop()

  This is where the main commands of the sketch are written. These commands are run over and over again when a sketch is uploaded to the board.

  Commands are terminated by a semi-colon and are placed within the two curly brackets. The loop() function is a void function, meaning that it does not return any values to the program.

- Libraries

Libraries are pieces of computer program that extend the functionality of the Arduino environment. Several libraries come with the Arduino IDE. There are many more libraries created by users which can be downloaded and used. Libraries can be accessed through Sketch→Include Library. Once included, the command #include<"NameofLibrary".h > will be automatically inserted at the beginning of the program.

5.3. Uploading

Before a sketch can be uploaded to the Arduino for the first time, the Arduino board and the computer port to which it is connected must be selected from Tools→Board and Tools→Port. Selected once, these setting should remain unchanged in future attempts.

Once the serial port and board are chosen and the program verified, hitting Upload will load the program onto the Arduino. The RX and TX LEDs will blink during the upload process and a status bar appears indicting the upload progress.

Arduino uses a bootloader, a small computer program (firmware) that has been preloaded on the microcontroller's flash memory and allows uploading code without the use of additional hardware.

6. Breadboard

Breadboards are bases for creating electronic circuits. The solderless breadboards available in the labs can be used repeatedly for different projects.

Breadboards have two outer vertical columns on the sides. These are called power rails and marked by red(+) and blue(-) vertical lines and are used to connect the voltage sources. All the holes on the same column are electrically connected. It is common and good practice to connect the power source to the red(+) rails and the ground to blue(-) rails.

The two midsections are called the terminal strips and are used to connect the electronics of the circuit. On each of the two mid-sections, the holes in a row are electrically connected. The rows are not connected across the middle ridge. These connections are shown in Figure 5.

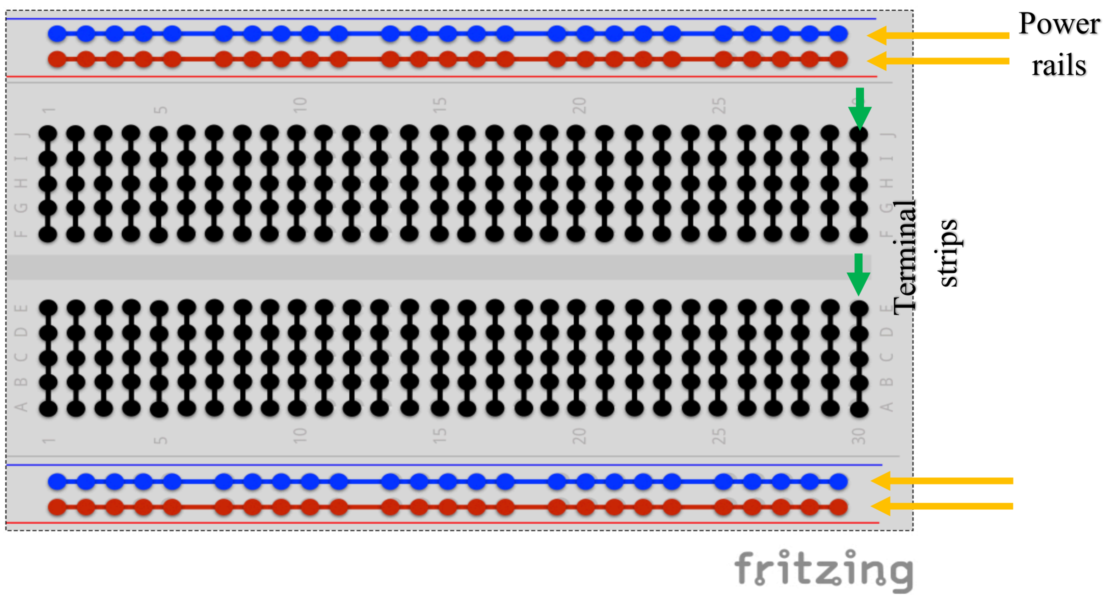


Figure 5- Breadboard and its connections

7. Resources references and further reading

Arduino has become a very popular open source platform. There are references and resources available on the Internet for almost any project or any function. Some of these are used in this document and referenced here. A simple Internet search will yield many more.

8. References

Adafruit. (2017, September). Retrieved from Adafruit: https://learn.adafruit.com/

Arduino References. (2017). Retrieved from Arduino: https://www.arduino.cc/en/Reference/HomePage

How to Mechatronics. (2017, September). Retrieved from http://howtomechatronics.com/category/tutorials/

Newbie Hack. (2017). Retrieved from Newbie Hack: https://www.newbiehack.com/

What is An Arduino? (2017). Retrieved 2017, from Spark Fun Electronics: https://learn.sparkfun.com/tutorials/what-is-an-arduino?_ga=2.28877820.1967696047.1505866652-1501311264.1505866652