PRACTICAL - I

AIM: Understanding Arduino and using Arduino and the normal bread board to switch on and off the LED.

Introduction to Arduino

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. It is like the brain of a project.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

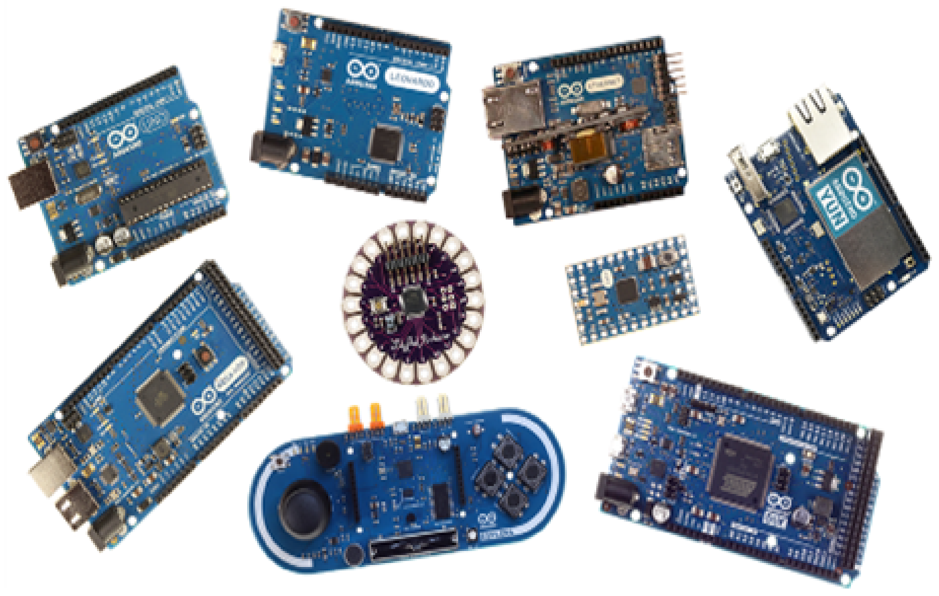


Fig 1.1 - Different Arduino Boards

The 'Arduino UNO'

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output(I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated Development Environment) via a type B USB cable.

It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo.

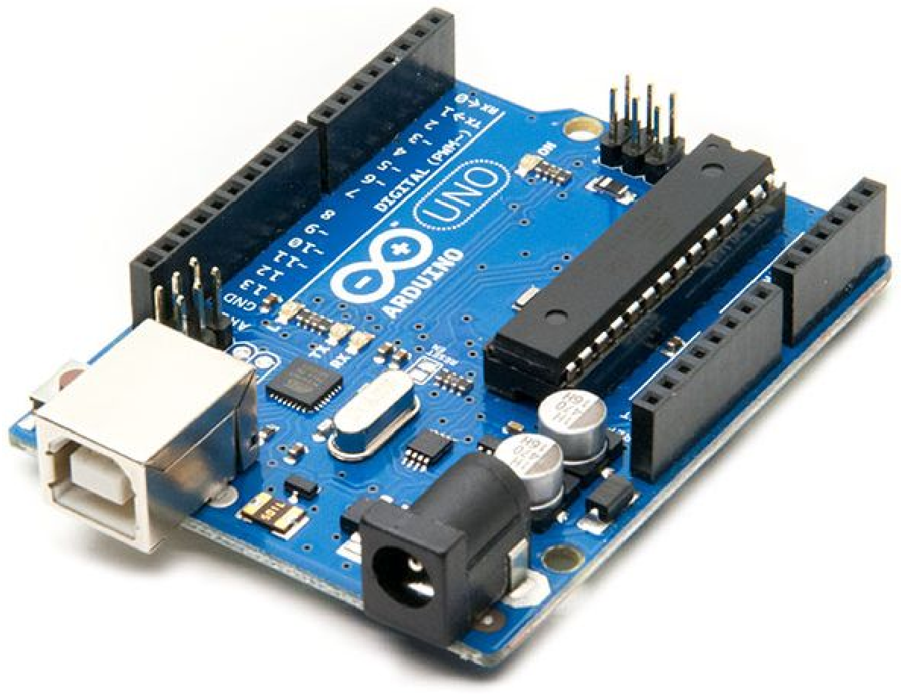


Fig 1.2 - Arduino Uno Controller

Here are some prominent features of the Arduino Uno

- The operating voltage is 5V and the input voltage ranges from 6v to 20V
- Digital I/O pins are 14 and Analog I/O pins are 6
- DC Current for each I/O pin is 40 mA and for 3.3V Pin is 50 mA
- Flash Memory is 32 KB, SRAM is 2 KB, EEPROM is 1 KB and the processor is clocked at a CLK Speed of 16 MHz

The 'Arduino UNO' Pin Structure

Power Supply: The Arduino Uno power supply can be done with the help of a USB cable or an external power supply. The external power supplies mainly include AC to DC adapter otherwise a battery. The adapter can be connected to the Arduino Uno by plugging into the power jack of the Arduino board. Similarly, the battery leads can be connected to the Vin pin and GND pin of the POWER connector. The suggested voltage range will be 7 volts to 12 volts.

Input & Output: The 14 digital pins on the Arduino Uno can be used as input & output with the help of the functions like pinMode(), digitalWrite(), & Digital Read().

Pin1 (TX) & Pin0 (RX) (Serial): This pin is used to transmit & receive TTL serial data, and these are connected to the ATmega8U2 USB to TTL Serial chip equivalent pins.

Pin2 & Pin3 (External Interrupts): External pins can be connected to activate an interrupt over a low value, change in value.

Pins 3, 5, 6, 9, 10, & 11 (PWM): This pin gives 8-bit PWM o/p by the function of analogWrite(). SPI Pins (Pin-10 (SS), Pin-11 (MOSI), Pin-12 (MISO), Pin-13 (SCK): These pins maintain SPI-communication, even though offered by the fundamental hardware, is not presently included within the Arduino language.

Pin-13(LED): The inbuilt LED can be connected to pin-13 (digital pin). As the HIGH-value pin, the light emitting diode is activated, whenever the pin is LOW.

Pin-4 (SDA) & Pin-5 (SCL) (I2C): It supports TWI-communication with the help of the Wire library. AREF (Reference Voltage): The reference voltage is for the analog i/ps with analogReference().

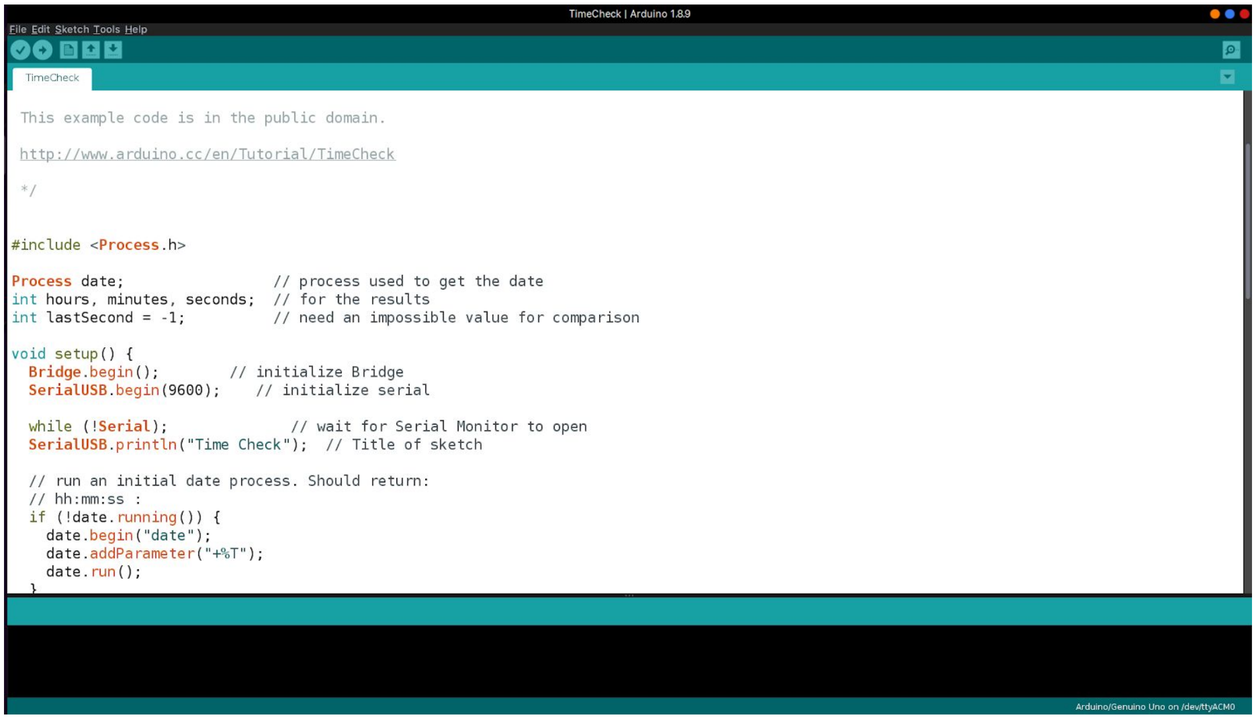
Reset Pin: This pin is used for reset (RST) the microcontroller.

The 'Arduino IDE'

Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.

Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded to the controller on the board.

The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.


TimeCheck | Arduino 1.8.9
File Edit Sketch Tools Help
TimeCheck
Arduino/Genuino Uno on /dev/ttyACM0

```
 This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/TimeCheck

 */


#include <Process.h>

Process date;                 // process used to get the date
int hours, minutes, seconds;  // for the results
int lastSecond = -1;          // need an impossible value for comparison

void setup() {
  Bridge.begin();        // initialize Bridge
  SerialUSB.begin(9600);    // initialize serial

  while (!Serial);              // wait for Serial Monitor to open
  SerialUSB.println("Time Check");  // Title of sketch

  // run an initial date process. Should return:
  // hh:mm:ss :
  if (!date.running()) {
    date.begin("date");
    date.addParameter("+%T");
    date.run();
  }
```

Fig 1.2 - Arduino IDE Interface and Code Editor

Blinking an LED using the Arduino UNO

Requirements

- 1 x Arduino UNO board and USB cable
- 1 x 5mm/3mm LED
- 1 x Breadboard

Pin Diagram

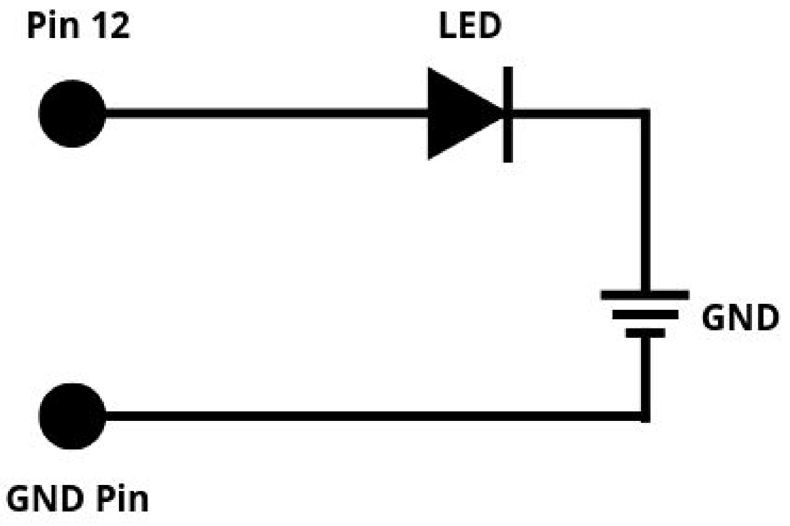


Fig 1.3 - Pin Diagram for Blinking an LED with Arduino

Sketch/Code

```
int PIN = 12;
void setup() {
  pinMode(PIN, OUTPUT);
}
void loop() {
  digitalWrite(PIN, HIGH);
  delay(256);
  digitalWrite(PIN, LOW);
  delay(256);
}
```

OUTPUT

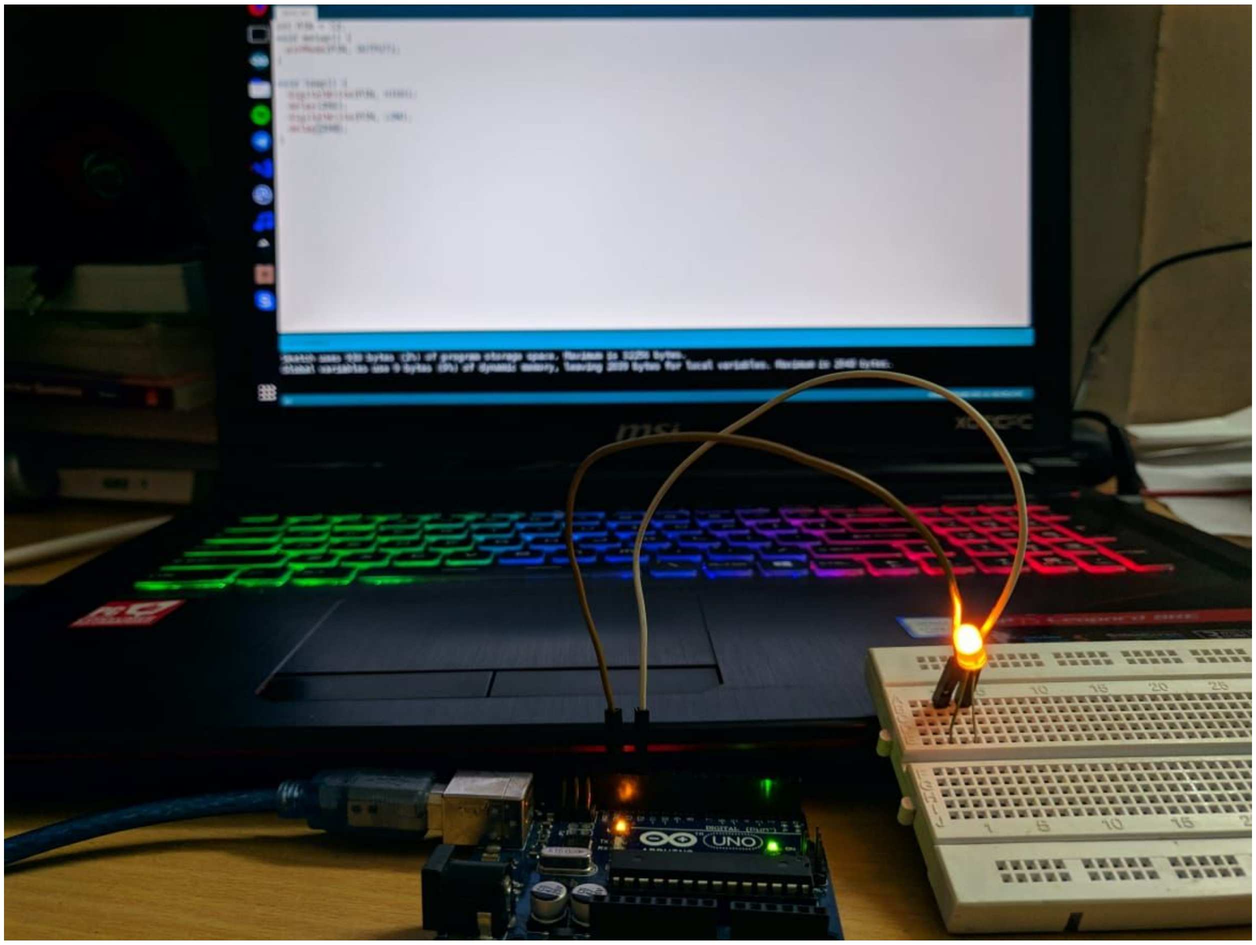


Fig 1.4 - Blinking an LED with Arduino