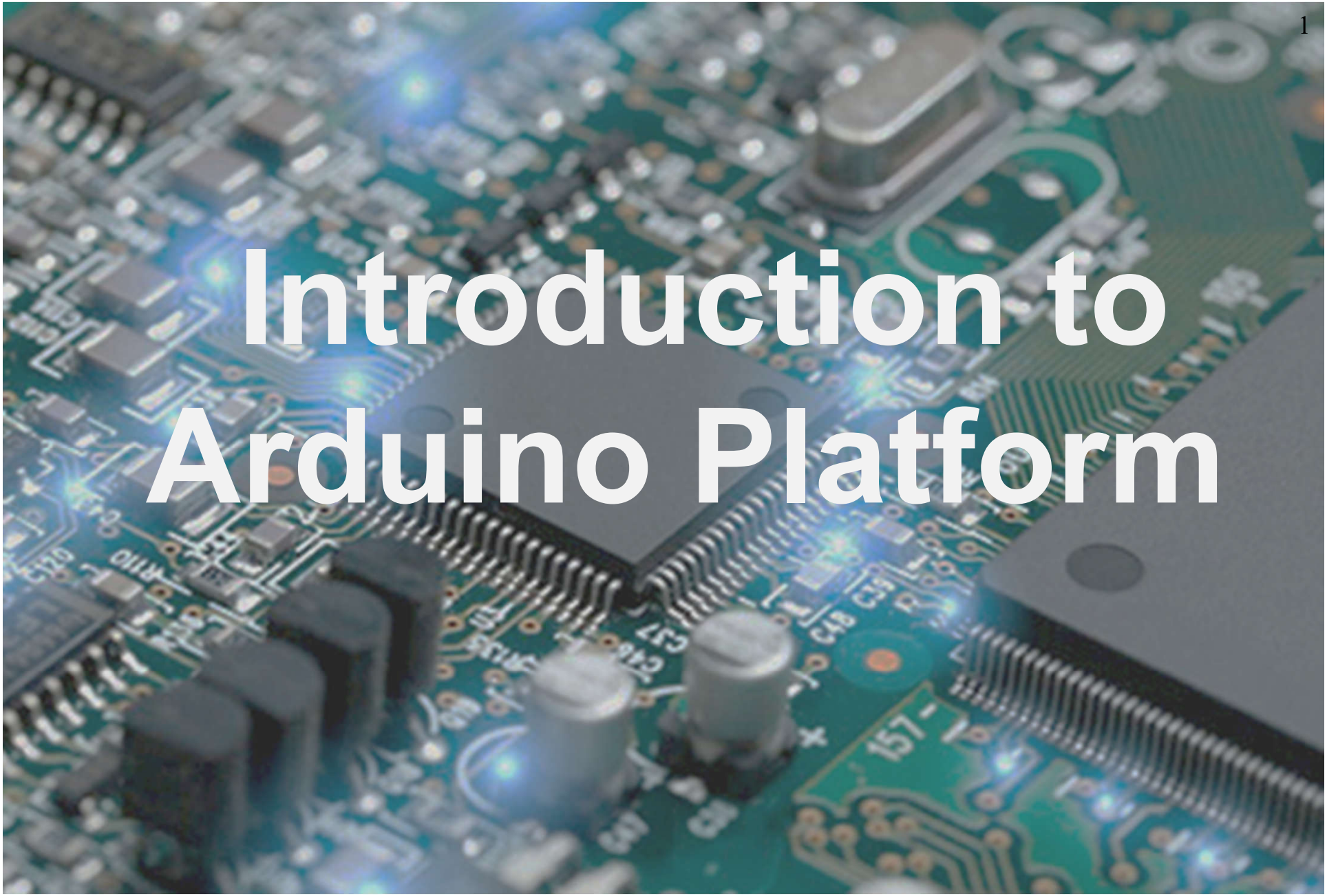


Introduction to Arduino Platform

Introduction to Arduino Platform

Contents

1.1 Objectives

- In this tutorial, you will learn:
  - To understand Arduino basic concepts.
  - To know about the Importance of an Arduino.
  - To become familiar with the types of Arduino Boards.
  - To understand the advantages and disadvantages of an Arduino.
  - To know how to start Arduino programming
  - To appreciate the history of an Arduino.

1.2 Introduction

- Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis had an idea for an easy-to-use microcontroller development board for Interaction Design Institute Ivrea in Ivrea, Italy way back **2005**. They need a cheap programmable device for students and artists. They selected the ATmega8 microcontroller, an AVR family of 8-bit microcontroller from Atmel, wrote bootloader firmware called Optiboot, and incorporated with Integrated Development Environment to write programs called "sketches." The project is called Arduino.

1.3 The History of an Arduino

- The Arduino project was started at the Interaction Design Institute Ivrea (IDII) in Ivrea, Italy.At that time, the students used a BASIC Stamp microcontroller at a cost of $50, a considerable expense for many students. Standardization
- In 2003 Hernando Barragán created the development platform Wiring as a Master's thesis project at IDII, under the supervision of Massimo Banzi and Casey Reas. Casey Reas is known for co-creating, with Ben Fry, the Processing development platform. The project goal was to create simple, low cost tools for creating digital projects by non-engineers. The Wiring platform consisted of a printed circuit board (PCB) with an ATmega168 microcontroller, an IDE based on Processing and library functions to easily program the microcontroller.
- In 2005, Massimo Banzi, with David Mellis, another IDII student, and David Cuartielles, extended Wiring by adding support for the cheaper ATmega8 microcontroller. The new project, forked from Wiring, was called Arduino.
- It was estimated in mid-2011 that over 300,000 official Arduinos had been commercially produced, and in 2013 that 700,000 official boards were in users' hands.

1.4 What is an Arduino?

Arduino is an open-source embedded system development platform using C/C++ programming language with Arduino Libraries created by the Arduino Company and very popular in embedded community/users.

2 Essential parts of Arduino:

1. Arduino Development Board-the hardware part of the ecosystem
2. Arduino Integrated Development Environment (IDE)-the software part where you wrote codes called sketches to compile and upload to the hardware.

1.5 Why Is It Arduino Important?

- Easy-to-use programming environment
- Flexible enough for advanced users.
- Cross platform IDE, runs on Mac, Windows, and Linux.
- Low cost development board
- Free compiler
- Open source both hardware and software.
- Large Community Users

1.6 Types of Arduino Boards

1.7 Advantages and Disadvantages of an Arduino

Advantages of Arduino Platform compared to others:

1. Multiplatform environment and it can run on Windows, Mac and Linux.
2. IDE is based on Processing which is easy-to- use for both artists and engineers.
3. Uploading program via USB.PCs and laptops today has no serial ports.
4. It is open source hardware and software- you can copy and modify hardware and software for your own application as long as you published as General Public Licensed in the internet.
5. Huge community of users-there are so many Arduino libraries can be found at internet and you can add easily on IDE.
6. The learning curve is not that difficult-you can start immediately after reading the Arduino official website.

1.7 Advantages and Disadvantages of an Arduino

Disadvantages of Arduino Platform:

1. We don't get to get to the know about the microcontroller inside the **Arduino** deeply.

2. Also if we use **Arduino** IDE we get limited library. If we use Atmel studio we can know the microcontroller deeply.

3. Many people just copy the code from internet without any knowledge about it.

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

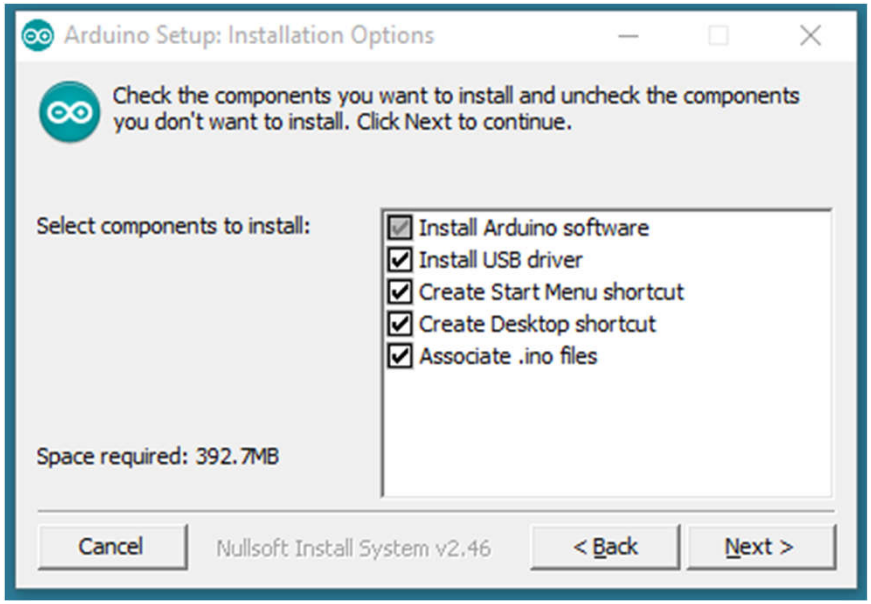
These are the steps how to connect Arduino Uno to your computer and upload your code called sketch. To upload the sketch to your Arduino Uno, you need to install the Arduino Software called Arduino Ingrated Development Environment (IDE). Download the installer using site below.

https://www.arduino.cc/download_handler.php?f=/arduino-1.8.12-windows.exe

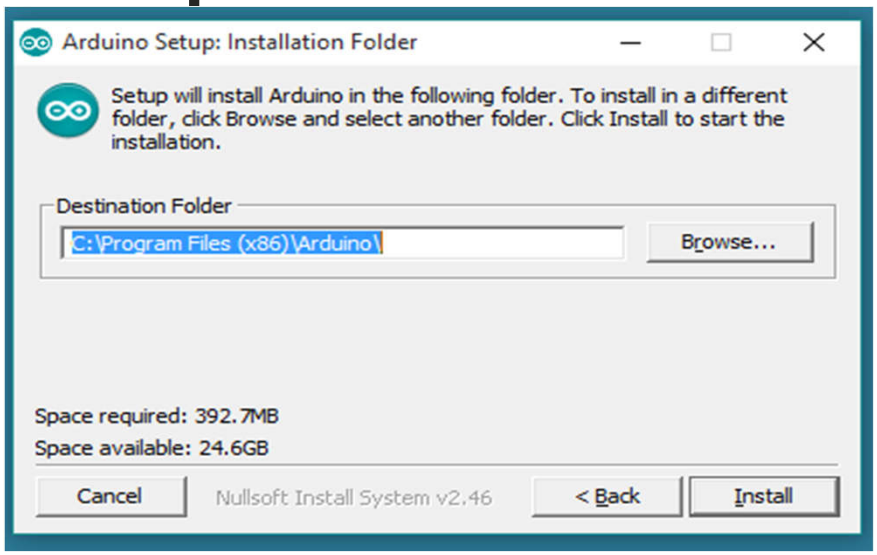
Or if you want standalone Arduino IDE you can also download in this link

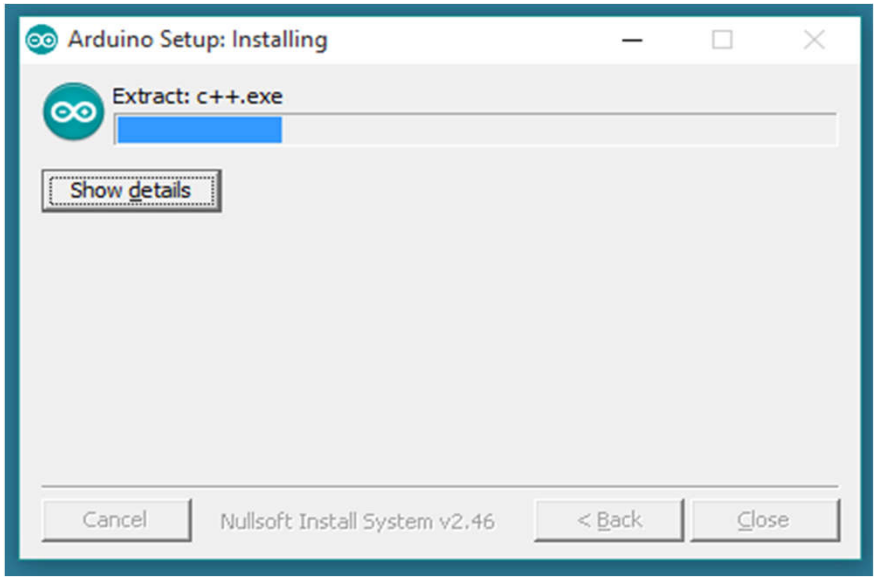
https://www.arduino.cc/download_handler.php?f=/arduino-1.8.12-windows.zip

During installaltion click all the components then select installation directory, wait until the Arduino IDE finished installed.

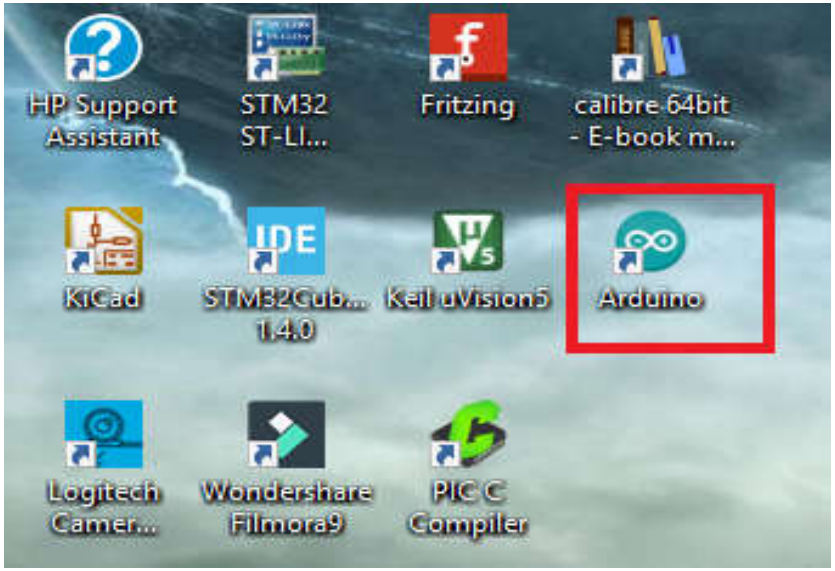


1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board





1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board



1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Installing the Arduino Drivers

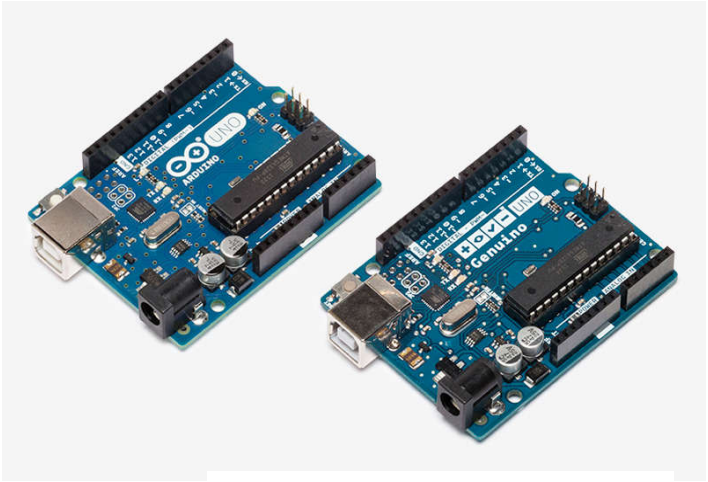
By using USB cable, connect the Arduino Uno board to the PC or laptop

Arduino Uno Drivers

FTDI

https://learn.sparkfun.com/tutorials/how-to-install-ftdi-drivers/all

https://ftdichip.com/drivers/



CH340

https://sparks.gogo.co.nz/ch340.html

CP210x Silabs

https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

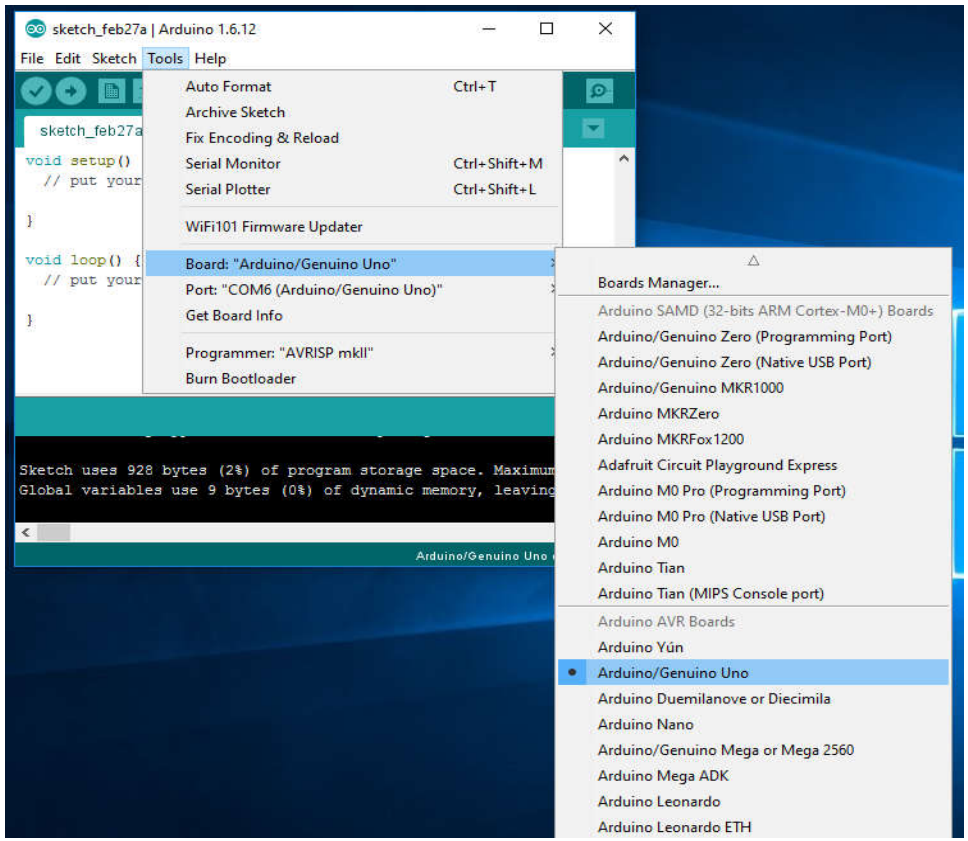
Prolific

http://www.prolific.com.tw/US/ShowProduct.aspx?p_id=225&pcid=41

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

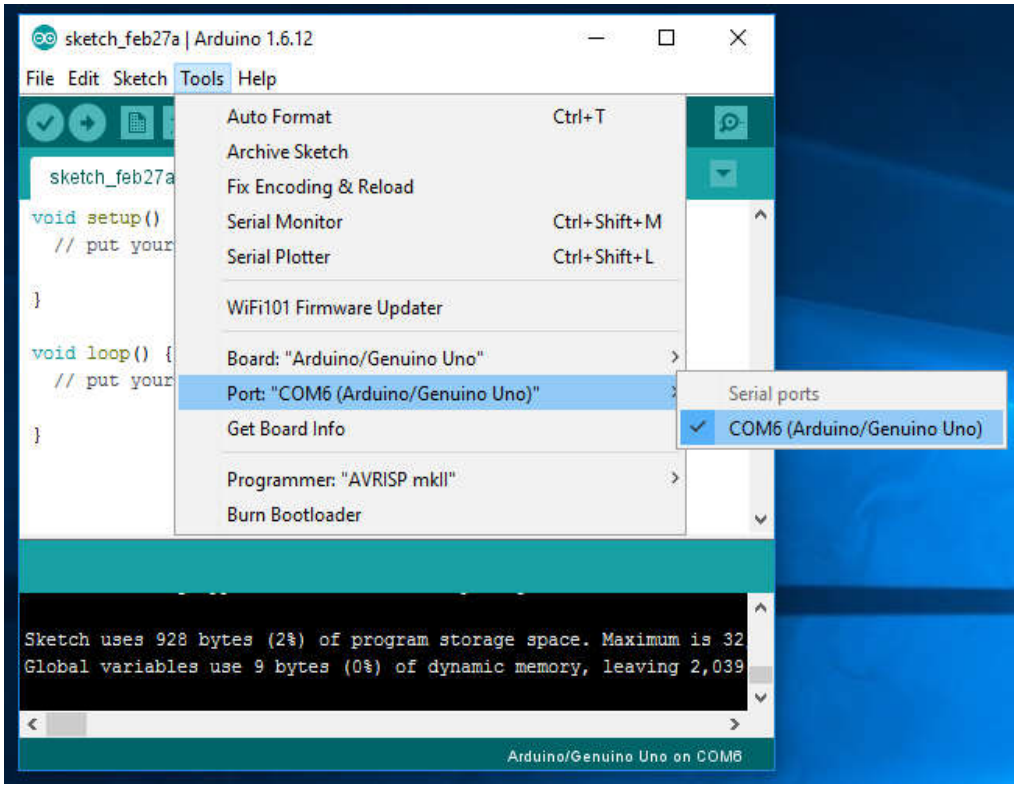
Testing your first program the Hello World of Embedded System: Blinking LED

1. Open the Arduino IDE and connect the Arduino Uno to the computer.
2. Select Tools->Board then choose Arduino/Genuino Uno.



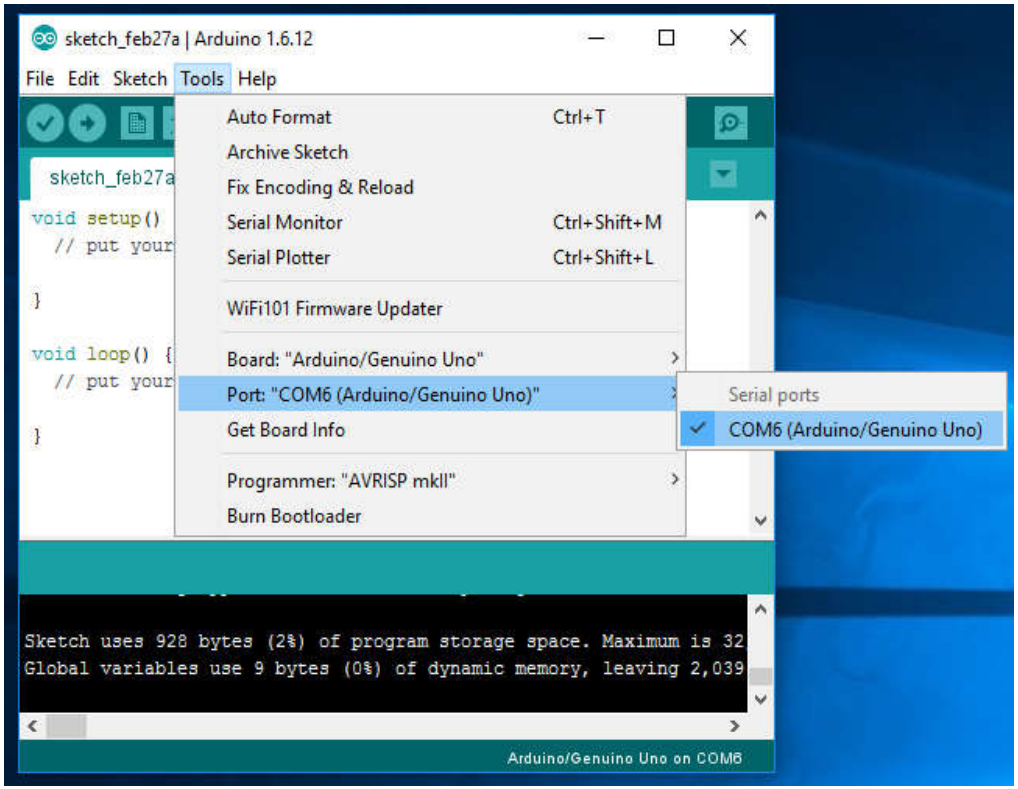
1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

3. Click Tools->Port the select COM port of Arduino/Genuino Uno.



1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

4. Next is go to File->Examples->Basics then Blink. The IDE will display another window and the sketch Blink is already encoded in the IDE. See the code below.

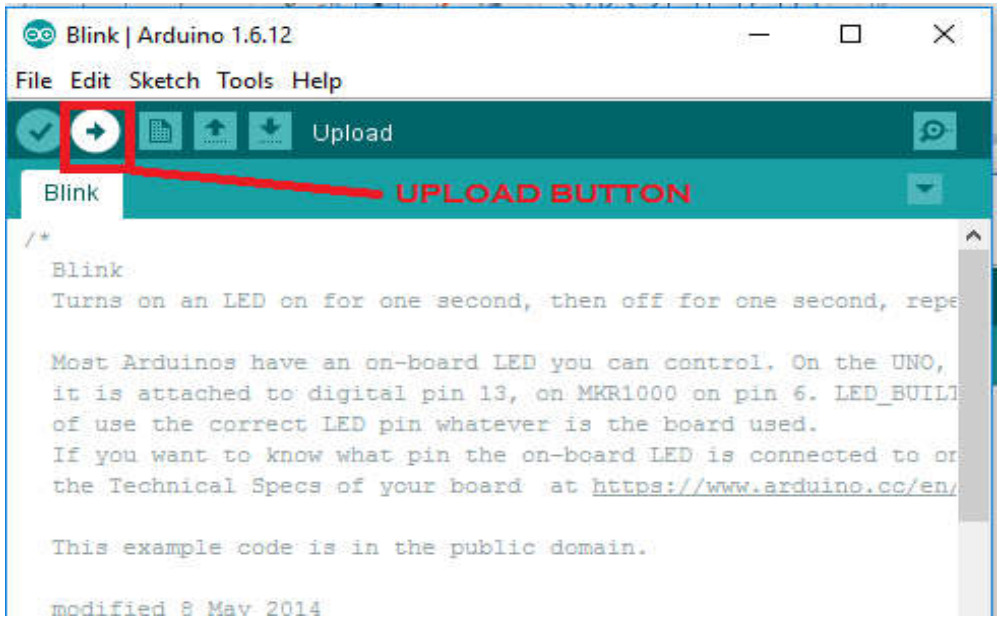


1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

```
/*
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, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN takes care
of use the correct LED pin whatever is the board used.
If you want to know what pin the on-board LED is connected to on your Arduino model,
check
*/
// the setup function runs once when you press reset or power the board
void setup()
{
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
}

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

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

5. Click the Upload button in the IDE and wait after the code or sketch uploaded into the Arduino Uno.



6. After programming the Arduino Uno, the pin 13 with LED is blinking ON and OFF for a second.

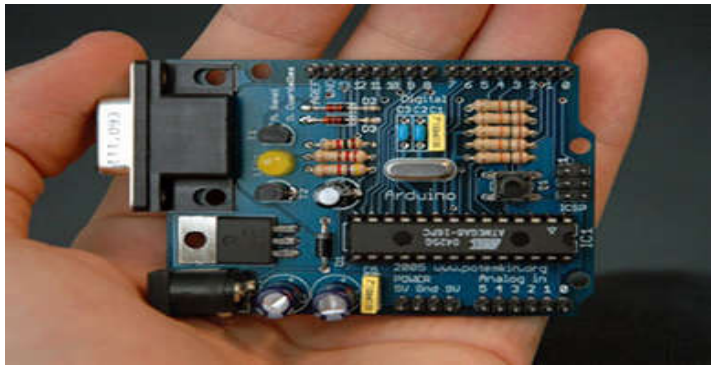
1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

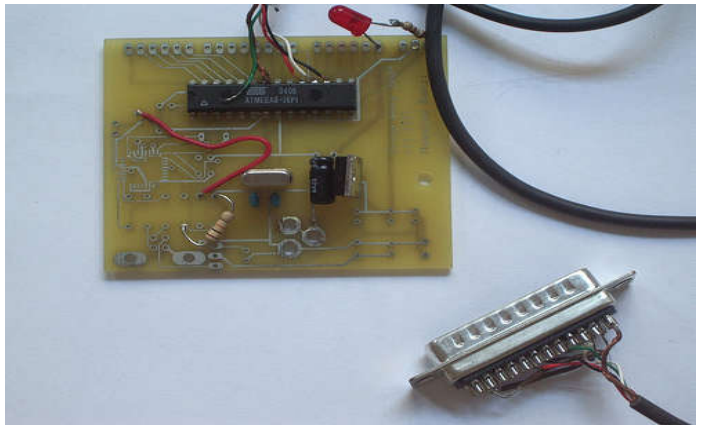
Technical specs of an Arduino Uno

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P)
of which 0.5 KB used by bootloader	
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

- A project which began in Ivrea (Italy) in 2005
- Massimo Banzi
- David Cuartielles
- device for controlling student-built interaction design
- less expensive
- more modern than what is available
- Basic Stamp killer





1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

What is a Arduino?

- named the project after a local bar
- italian masculine first name
- "strong friend"
- English equivalent is "Hardwin"

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup and Loop

```
void setup()
{
   //enter code here
}

void loop()
{
   //enter code here
 }
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup()

called when a sketch starts running

use it to initialize

variables,

pin modes,

- start using libraries
- will only run once, after each power up or reset of the Arduino board

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Setup()

```
void setup()
    {
        LEDPin = 13;
        pinMode(LEDPin, OUTPUT);
        digitalWrite(LEDPin, LOW);
    }

    void loop()
    {
    }
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Loop()

-Executed repeatedly

-Allows the program to change and respond.

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Programming

- Loop()

```
void setup()
{
    LEDPin = 13;
    pinMode(LEDPin, OUTPUT);
    digitalWrite(LEDPin, LOW);
}

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

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- Predefined functions to easily configure Arduino
- Pin Operation
- Peripheral Operation

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- pinMode(pin, mode)
- Configures a digital pin to behave either as an input or an output
- pin: the pin number whose mode you wish to set
- mode: either INPUT or OUTPUT

- Example

```
pinMode(13, OUTPUT);
```

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- digitalWrite(pin, value)
- Write a HIGH or a LOW value to a digital pin
- pin: the pin number whose value you wish to set
- value: either HIGH or LOW

- Example
  digitalWrite(13, HIGH);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- digitalRead(pin)
- Reads the value from a specified digital pin
- pin: the digital pin number you want to read
- returns either HIGH or LOW

- Example
- result = digitalRead(13);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- delay(value)
- Pauses the program for the amount of time
- value: the number of milliseconds to pause

- Example
- delay(1000);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

- analogRead(pin)
- Reads the value from a specified analog pin
- pin: the analog pin number you want to read
- returns an integer (0 to 1023) corresponding to pin voltage

- Example
- result = analogRead(13);

1.8 Getting Started Using Arduino Uno- The Most Important Arduino Board

Arduino Built-in Functions

```
Blinking LED Example

int LEDPin;
// setup initializes the LED pin as output
// and initially turned off
void setup()
{
        LEDPin = 13;
        pinMode(LEDPin, OUTPUT);
        digitalWrite(LEDPin, LOW);
}
// loop checks the button pin each time,
void loop()
{
        delay(500);
        digitalWrite(LEDPin, HIGH);
        delay(500);
        digitalWrite(LEDPin, LOW);
}
```

1.9 Arduino Language Reference

Digital I/O

digitalWrite()

Description
Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(),
its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.
It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.

If you do not set the pinMode() to OUTPUT, and connect an LED to a pin, when calling digitalWrite(HIGH), the LED may appear dim.
Without explicitly setting pinMode(), digitalWrite() will have enabled the internal pull-up resistor,
which acts like a large current-limiting resistor.

Syntax:
digitalWrite(pin, value)

Parameters:
pin: the Arduino pin number.
value: HIGH or LOW.

Returns:
Nothing

1.9 Arduino Language Reference

Example Code:
The code makes the digital pin 13 an OUTPUT and toggles it by alternating between HIGH and LOW at one second pace.

```
void setup()
{
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

1.9 Arduino Language Reference

digitalRead()

Reads the value from a specified digital pin, either HIGH or LOW.

Syntax
digitalRead(pin)

Returns
HIGH or LOW

Example Code
Sets pin 13 to the same value as pin 7, declared as an input.

```
int ledPin = 13;  // LED connected to digital pin 13
int inPin = 7;    // pushbutton connected to digital pin 7
int val = 0;      // variable to store the read value

void setup()
{
  pinMode(ledPin, OUTPUT);  // sets the digital pin 13 as output
  pinMode(inPin, INPUT);    // sets the digital pin 7 as input
}

void loop()
{
  val = digitalRead(inPin);   // read the input pin
  digitalWrite(ledPin, val);  // sets the LED to the button's value
}
```

1.9 Arduino Language Reference

pinMode()

Description
Configures the specified pin to behave either as an input or an output.

Syntax
pinMode(pin, mode)

Parameters
pin: the Arduino pin number to set the mode of.
mode: INPUT, OUTPUT, or INPUT_PULLUP.

Returns
Nothing

Example Code
The code makes the digital pin 13 OUTPUT and Toggles it HIGH and LOW

```
void setup()
{
  pinMode(13, OUTPUT);    // sets the digital pin 13 as output
}

void loop()
{
  digitalWrite(13, HIGH); // sets the digital pin 13 on
  delay(1000);            // waits for a second
  digitalWrite(13, LOW);  // sets the digital pin 13 off
  delay(1000);            // waits for a second
}
```

1.9 Arduino Language Reference

Analog I/O

Description
Reads the value from the specified analog pin. Arduino boards contain a multichannel, 10-bit analog to digital converter. This means that it will map input voltages between 0 and the operating voltage(5V or 3.3V) into integer values between 0 and 1023.
On an Arduino UNO, for example,
this yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit.

The input range can be changed using analogReference(), while the resolution can be changed (only for Zero, Due and MKR boards) using analogReadResolution().

On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

1.9 Arduino Language Reference

Syntax
analogRead(pin)

Parameters
pin: the name of the analog input pin to read from (A0 to A5 on most boards, A0 to A6 on MKR boards, A0 to A7 on the Mini and Nano, A0 to A15 on the Mega).

Returns
The analog reading on the pin. Although it is limited to the resolution of the analog to digital converter (0-1023 for 10 bits or 0-4095 for 12 bits).
Data type: int.

Example Code
The code reads the voltage on analogPin and displays it.

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0;  // variable to store the value read

void setup()
{
  Serial.begin(9600);           //  setup serial
}

void loop()
{
  val = analogRead(analogPin);  // read the input pin
  Serial.println(val);          // debug value
}
```

1.9 Arduino Language Reference

analogReference()

Description
Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The options are:

Arduino AVR Boards (Uno, Mega, Leonardo, etc.)

DEFAULT: the default analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards)

INTERNAL: a built-in reference, equal to 1.1 volts on the ATmega168 or ATmega328P and 2.56 volts on the ATmega32U4 and ATmega8 (not available on the Arduino Mega)

INTERNAL1V1: a built-in 1.1V reference (Arduino Mega only)

INTERNAL2V56: a built-in 2.56V reference (Arduino Mega only)

EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference.

1.9 Arduino Language Reference

Syntax
analogReference(type)

Parameters
type: which type of reference to use (see list of options in the description).

Returns
Nothing

Notes and Warnings
After changing the analog reference, the first few readings from analogRead() may not be accurate.

Don't use anything less than 0V or more than 5V for external reference voltage on the AREF pin! If you're using an external reference on the AREF pin, you must set the analog reference to EXTERNAL before calling analogRead(). Otherwise, you will short together the active reference voltage (internally generated) and the AREF pin, possibly damaging the microcontroller on your Arduino board.

Alternatively, you can connect the external reference voltage to the AREF pin through a 5K resistor, allowing you to switch between external and internal reference voltages. Note that the resistor will alter the voltage that gets used as the reference because there is an internal 32K resistor on the AREF pin. The two act as a voltage divider, so, for example, 2.5V applied through the resistor will yield 2.5 * 32 / (32 + 5) = ~2.2V at the AREF pin.

1.9 Arduino Language Reference

Random Numbers

random()

Description
The random function generates pseudo-random numbers.

Syntax
random(max)
random(min, max)

Parameters
min: lower bound of the random value, inclusive (optional).
max: upper bound of the random value, exclusive.

Returns
A random number between min and max-1.
Data type: long.

1.9 Arduino Language Reference

Example Code
The code generates random numbers and displays them.

```
long randNumber;

void setup() {
  Serial.begin(9600);

  // if analog input pin 0 is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
  randomSeed(analogRead(0));
}

void loop() {
  // print a random number from 0 to 299
  randNumber = random(300);
  Serial.println(randNumber);

  // print a random number from 10 to 19
  randNumber = random(10, 20);
  Serial.println(randNumber);

  delay(50);
}
```

1.9 Arduino Language Reference

randomSeed()

Description
randomSeed() initializes the pseudo-random number generator, causing it to start at an arbitrary point in its random sequence. This sequence, while very long, and random, is always the same.

If it is important for a sequence of values generated by random() to differ, on subsequent executions of a sketch, use randomSeed() to initialize the random number generator with a fairly random input, such as analogRead() on an unconnected pin.

Conversely, it can occasionally be useful to use pseudo-random sequences that repeat exactly. This can be accomplished by calling randomSeed() with a fixed number, before starting the random sequence.

Syntax
randomSeed(seed)

Parameters
seed: number to initialize the pseudo-random sequence. Allowed data types: unsigned long.

Returns
Nothing

1.9 Arduino Language Reference

Example Code
The code generates a pseudo-random number and sends the generated number to the serial port.

```
long randNumber;

void setup() {
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop() {
  randNumber = random(300);
  Serial.println(randNumber);
  delay(50);
}
```

1.9 Arduino Language Reference

Timers

delay()

Description
Pauses the program for the amount of time (in milliseconds) specified as parameter.
(There are 1000 milliseconds in a second.)

Syntax
delay(ms)

Parameters
ms: the number of milliseconds to pause. Allowed data types: unsigned long.

Returns
Nothing

Example Code
The code pauses the program for one second before toggling the output pin.

```
int ledPin = 13;                // LED connected to digital pin 13

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

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                  // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);                  // waits for a second
}
```

1.9 Arduino Language Reference

delayMicroseconds()

Description
Pauses the program for the amount of time (in microseconds) specified by the parameter. There are a thousand microseconds in a millisecond and a million microseconds in a second.

Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay() instead.

Syntax
delayMicroseconds(us)

Parameters
us: the number of microseconds to pause. Allowed data types: unsigned int.

Returns
Nothing

Example Code
The code configures pin number 8 to work as an output pin. It sends a train of pulses of approximately 100 microseconds period. The approximation is due to execution of the other instructions in the code.

```
int outPin = 8;                 // digital pin 8

void setup() {
  pinMode(outPin, OUTPUT);      // sets the digital pin as output
}

void loop() {
  digitalWrite(outPin, HIGH); // sets the pin on
  delayMicroseconds(50);      // pauses for 50 microseconds
  digitalWrite(outPin, LOW);  // sets the pin off
  delayMicroseconds(50);      // pauses for 50 microseconds
}
```

1.9 Arduino Language Reference

millis()

Description
Returns the number of milliseconds passed since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Syntax
time = millis()

Parameters
None

Returns
Number of milliseconds passed since the program started. Data type: unsigned long.

Example Code
This example code prints on the serial port the number of milliseconds passed since the Arduino board started running the code itself.

```
unsigned long myTime;

void setup() {
  Serial.begin(9600);
}
void loop() {
  Serial.print("Time: ");
  myTime = millis();

  Serial.println(myTime); //prints time since program started
  delay(1000);          // wait a second so as not to send massive amounts of data
}
```

1.9 Arduino Language Reference

Advanced I/O

noTone()

Description
Stops the generation of a square wave triggered by tone(). Has no effect if no tone is being generated.

Syntax
noTone(pin)

Parameters
pin: the Arduino pin on which to stop generating the tone

Returns
Nothing

Note:
If you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

1.9 Arduino Language Reference

tone()

Description
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin. A duration can be specified, otherwise the wave continues until a call to noTone(). The pin can be connected to a piezo buzzer or other speaker to play tones.

Only one tone can be generated at a time. If a tone is already playing on a different pin, the call to tone() will have no effect. If the tone is playing on the same pin, the call will set its frequency.

Use of the tone() function will interfere with PWM output on pins 3 and 11 (on boards other than the Mega).

It is not possible to generate tones lower than 31Hz. For technical details, see Brett Hagman's notes.

Syntax
tone(pin, frequency)
tone(pin, frequency, duration)

Parameters
pin: the Arduino pin on which to generate the tone.
frequency: the frequency of the tone in hertz. Allowed data types: unsigned int.
duration: the duration of the tone in milliseconds (optional). Allowed data types: unsigned long.

Returns
Nothing

Notes
If you want to play different pitches on multiple pins, you need to call noTone() on one pin before calling tone() on the next pin.

1.9 Arduino Language Reference

pulseIn()

Description
Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

The timing of this function has been determined empirically and will probably show errors in longer pulses. Works on pulses from 10 microseconds to 3 minutes in length.

Syntax
pulseIn(pin, value)
pulseIn(pin, value, timeout)

Parameters
pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

Returns
The length of the pulse (in microseconds) or 0 if no pulse started before the timeout. Data type: unsigned long.

1.9 Arduino Language Reference

Example Code
The example prints the time duration of a pulse on pin 7.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseIn(pin, HIGH);
  Serial.println(duration);
}
```

1.9 Arduino Language Reference

pulseInLong()

Description
pulseInLong() is an alternative to pulseIn() which is better at handling long pulse and interrupt affected scenarios.

Reads a pulse (either HIGH or LOW) on a pin. For example, if value is HIGH, pulseInLong() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout.

The timing of this function has been determined empirically and will probably show errors in shorter pulses. Works on pulses from 10 microseconds to 3 minutes in length. This routine can be used only if interrupts are activated. Furthermore the highest resolution is obtained with large intervals.

Syntax
pulseInLong(pin, value)
pulseInLong(pin, value, timeout)

Parameters
pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second. Allowed data types: unsigned long.

Returns
The length of the pulse (in microseconds) or 0 if no pulse started before the timeout. Data type: unsigned long.

1.9 Arduino Language Reference

Example Code
The example prints the time duration of a pulse on pin 7.

```
int pin = 7;
unsigned long duration;

void setup() {
  Serial.begin(9600);
  pinMode(pin, INPUT);
}

void loop() {
  duration = pulseInLong(pin, HIGH);
  Serial.println(duration);
}
```

Notes
This function relies on micros() so cannot be used in noInterrupts() context.

1.9 Arduino Language Reference

shiftIn()

Description
Shifts in a byte of data one bit at a time. Starts from either the most (i.e. the leftmost) or least (rightmost) significant bit. For each bit, the clock pin is pulled high, the next bit is read from the data line, and then the clock pin is taken low.

If you're interfacing with a device that's clocked by rising edges, you'll need to make sure that the clock pin is low before the first call to shiftIn(), e.g. with a call to digitalWrite(clockPin, LOW).

Note: this is a software implementation; Arduino also provides an SPI library that uses the hardware implementation, which is faster but only works on specific pins.

Syntax
byte incoming = shiftIn(dataPin, clockPin, bitOrder)

Parameters
dataPin: the pin on which to input each bit. Allowed data types: int.
clockPin: the pin to toggle to signal a read from dataPin.
bitOrder: which order to shift in the bits; either MSBFIRST or LSBFIRST. (Most Significant Bit First, or, Least Significant Bit First).

Returns
The value read. Data type: byte.

1.9 Arduino Language Reference

shiftOut()

Description
Shifts out a byte of data one bit at a time. Starts from either the most (i.e. the leftmost) or least (rightmost) significant bit. Each bit is written in turn to a data pin, after which a clock pin is pulsed (taken high, then low) to indicate that the bit is available.

Note- if you're interfacing with a device that's clocked by rising edges, you'll need to make sure that the clock pin is low before the call to shiftOut(), e.g. with a call to digitalWrite(clockPin, LOW).

This is a software implementation; see also the SPI library, which provides a hardware implementation that is faster but works only on specific pins.

Syntax
shiftOut(dataPin, clockPin, bitOrder, value)

Parameters
dataPin: the pin on which to output each bit. Allowed data types: int.
clockPin: the pin to toggle once the dataPin has been set to the correct value. Allowed data types: int.
bitOrder: which order to shift out the bits; either MSBFIRST or LSBFIRST. (Most Significant Bit First, or, Least Significant Bit First).
value: the data to shift out. Allowed data types: byte.

Returns
Nothing

1.9 Arduino Language Reference

Example Code
For accompanying circuit, see the tutorial on controlling a 74HC595 shift register.

```
//  Name    : shiftOutCode, Hello World                   //
//  Author  : Carlyn Maw,Tom Igoe                         //
//  Date    : 25 Oct, 2006                                //
//  Version : 1.0                                         //
//  Notes   : Code for using a 74HC595 Shift Register     //
//          : to count from 0 to 255                      //
//Pin connected to ST_CP of 74HC595
int latchPin = 8;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;
////Pin connected to DS of 74HC595
int dataPin = 11;

void setup() {
  //set pins to output because they are addressed in the main loop
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  //count up routine
  for (int j = 0; j < 256; j++) {
    //ground latchPin and hold low for as long as you are transmitting
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, j);
    //return the latch pin high to signal chip that it
    //no longer needs to listen for information
    digitalWrite(latchPin, HIGH);
    delay(1000);
  }
}
```

1.9 Arduino Language Reference

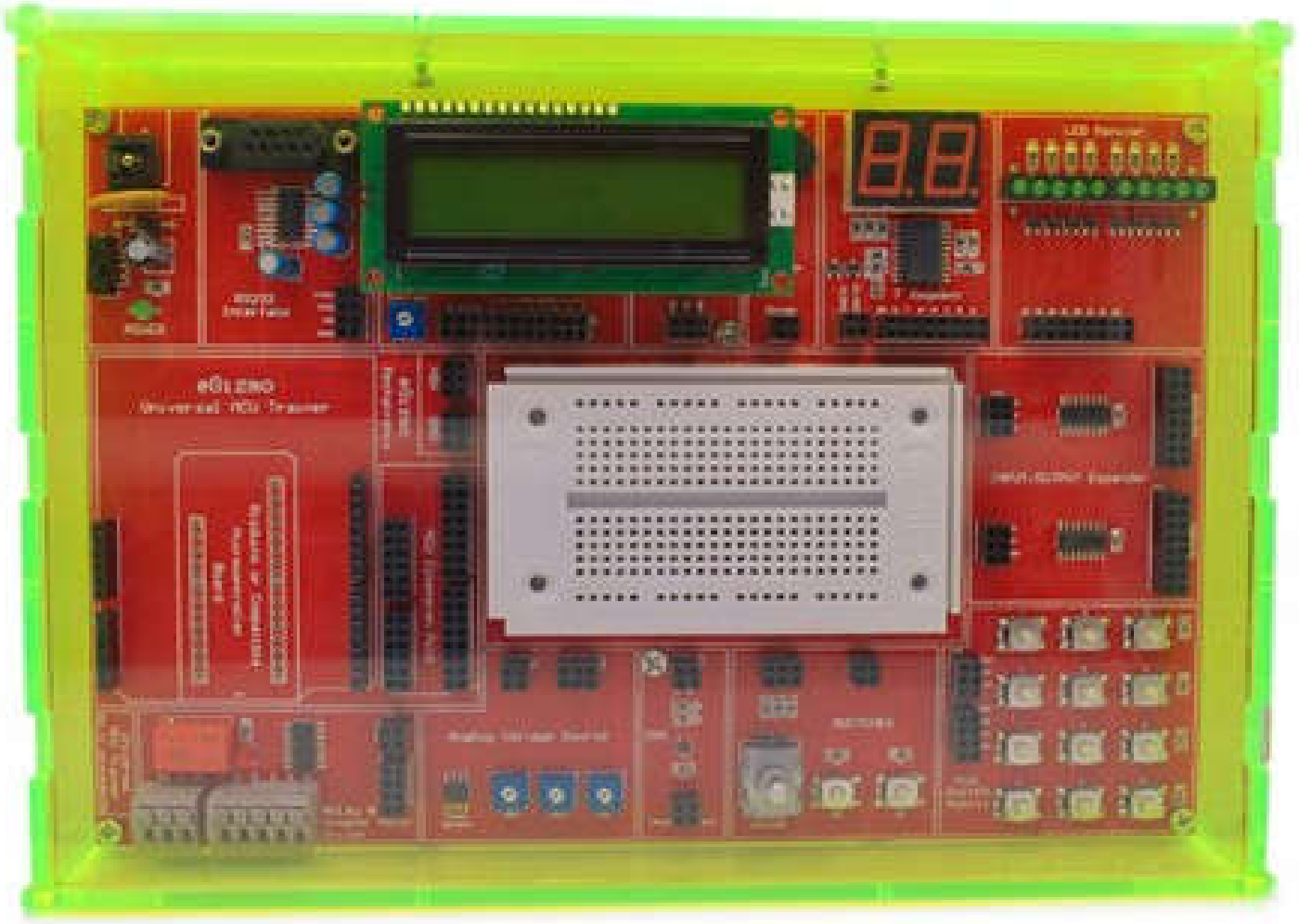
Notes
The dataPin and clockPin must already be configured as outputs by a call to pinMode().

shiftOut is currently written to output 1 byte (8 bits) so it requires a two step operation to output values larger than 255.

```
// Do this for MSBFIRST serial
int data = 500;
// shift out highbyte
shiftOut(dataPin, clock, MSBFIRST, (data >> 8));
// shift out lowbyte
shiftOut(dataPin, clock, MSBFIRST, data);

// Or do this for LSBFIRST serial
data = 500;
// shift out lowbyte
shiftOut(dataPin, clock, LSBFIRST, data);
// shift out highbyte
shiftOut(dataPin, clock, LSBFIRST, (data >> 8));
```

1.10 Arduino Preferred Kits

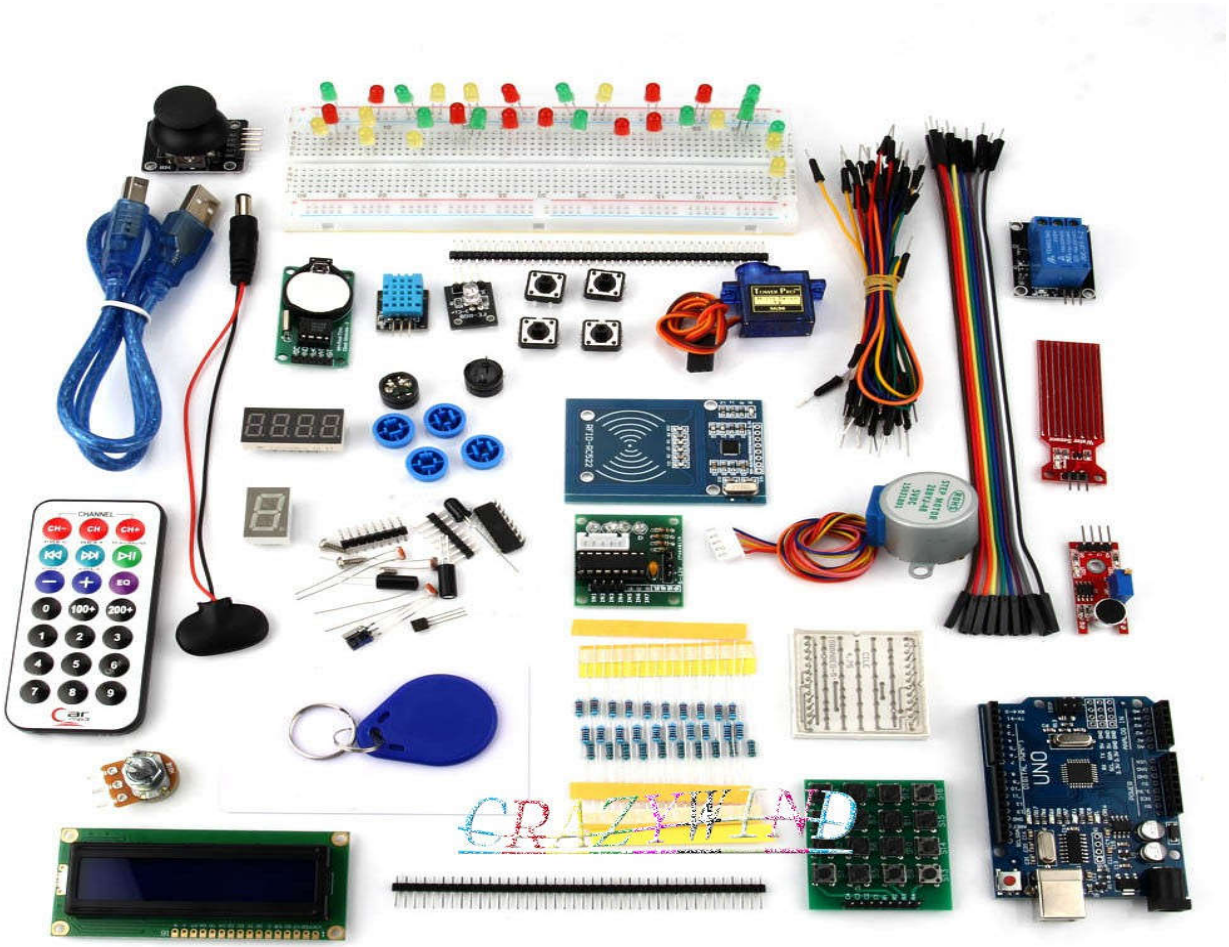
Universal MCU Digital Trainer



https://www.e-gizmo.net/oc/index.php?route=product/product&product_id=481

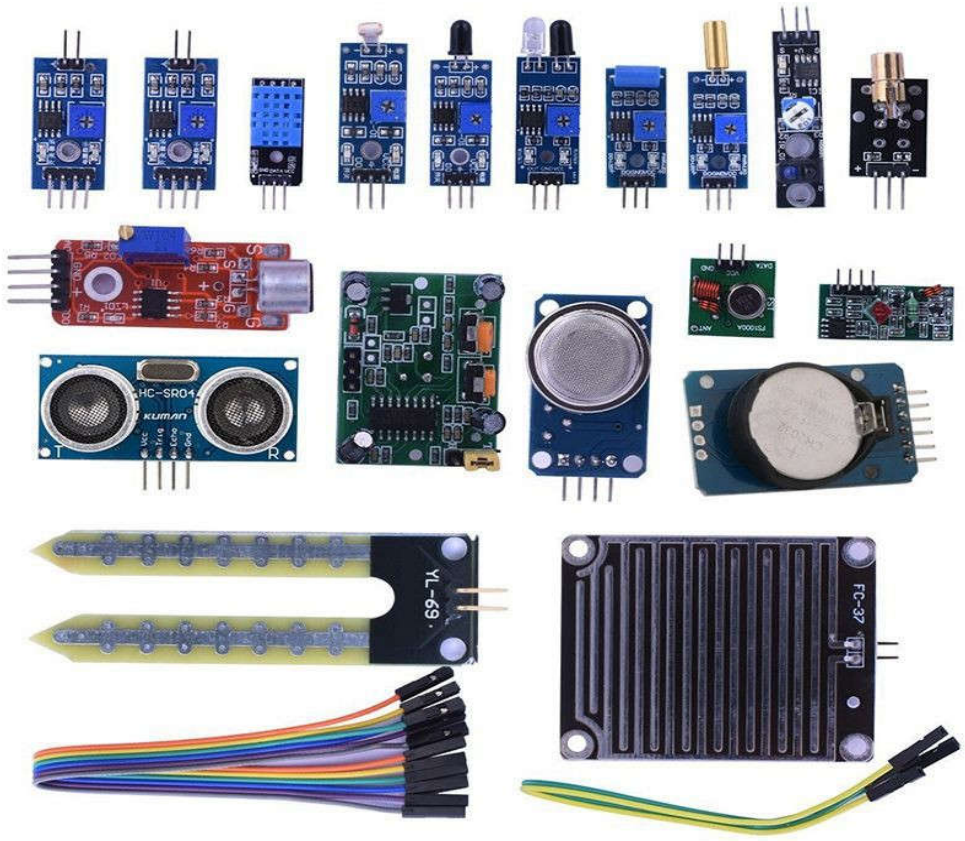
1.10 Preferred Kits for Arduino

RFID Learning Starter Kit for Arduino R3 Upgraded Version



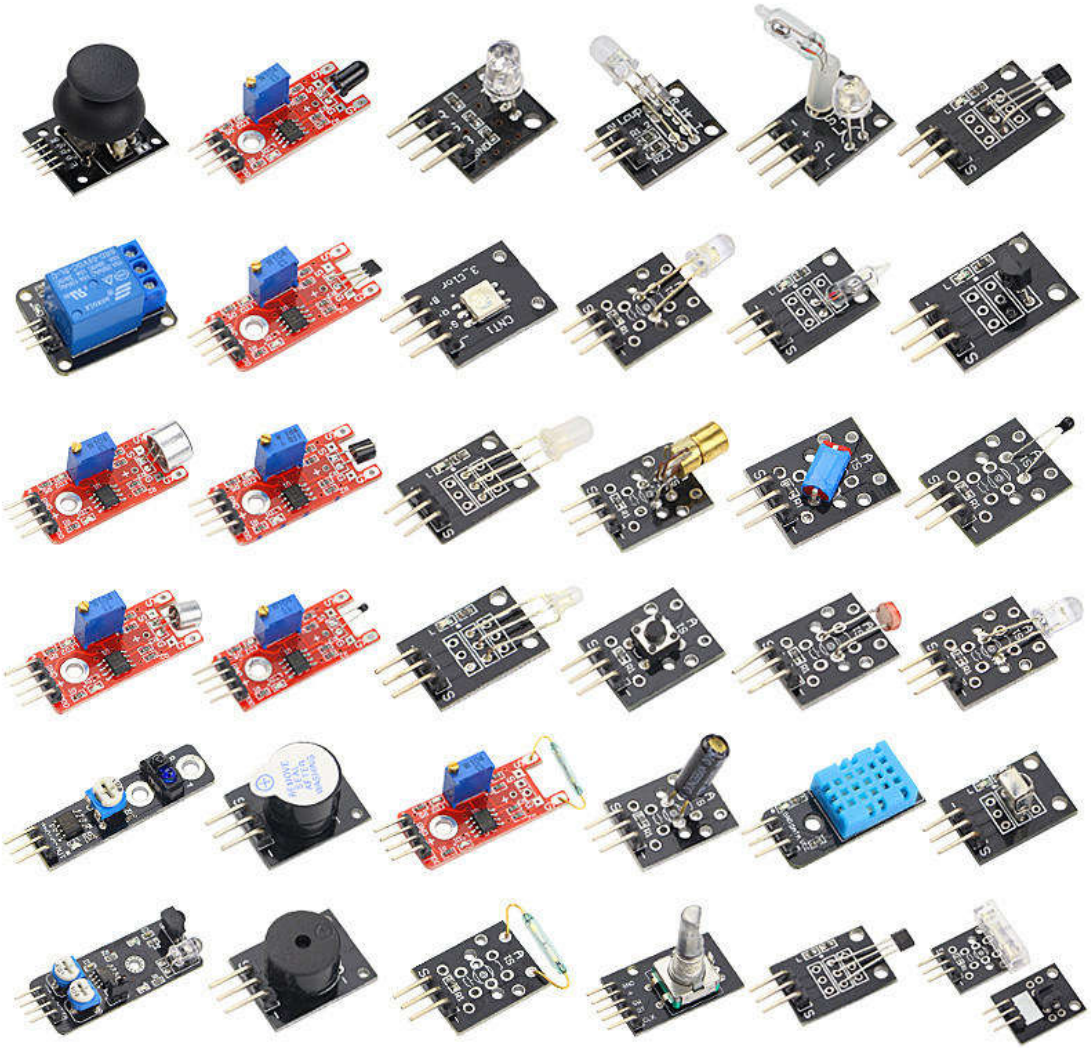
1.10 Preferred Kits for Arduino

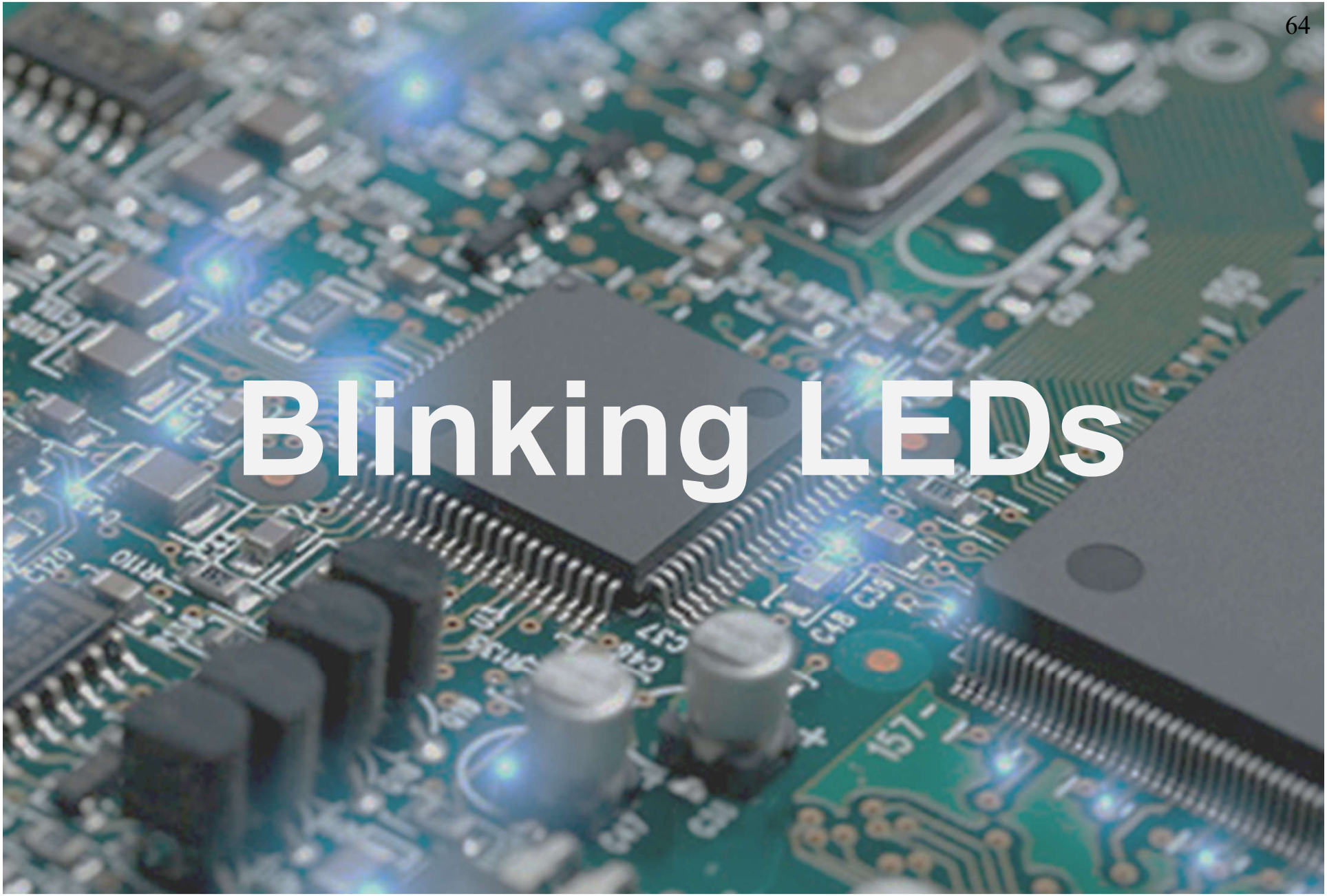
16 in 1 Sensor Kit



1.10 Preferred Kits for Arduino

37 in 1 Sensor Kit





Blinking LEDs

Blinking LEDs

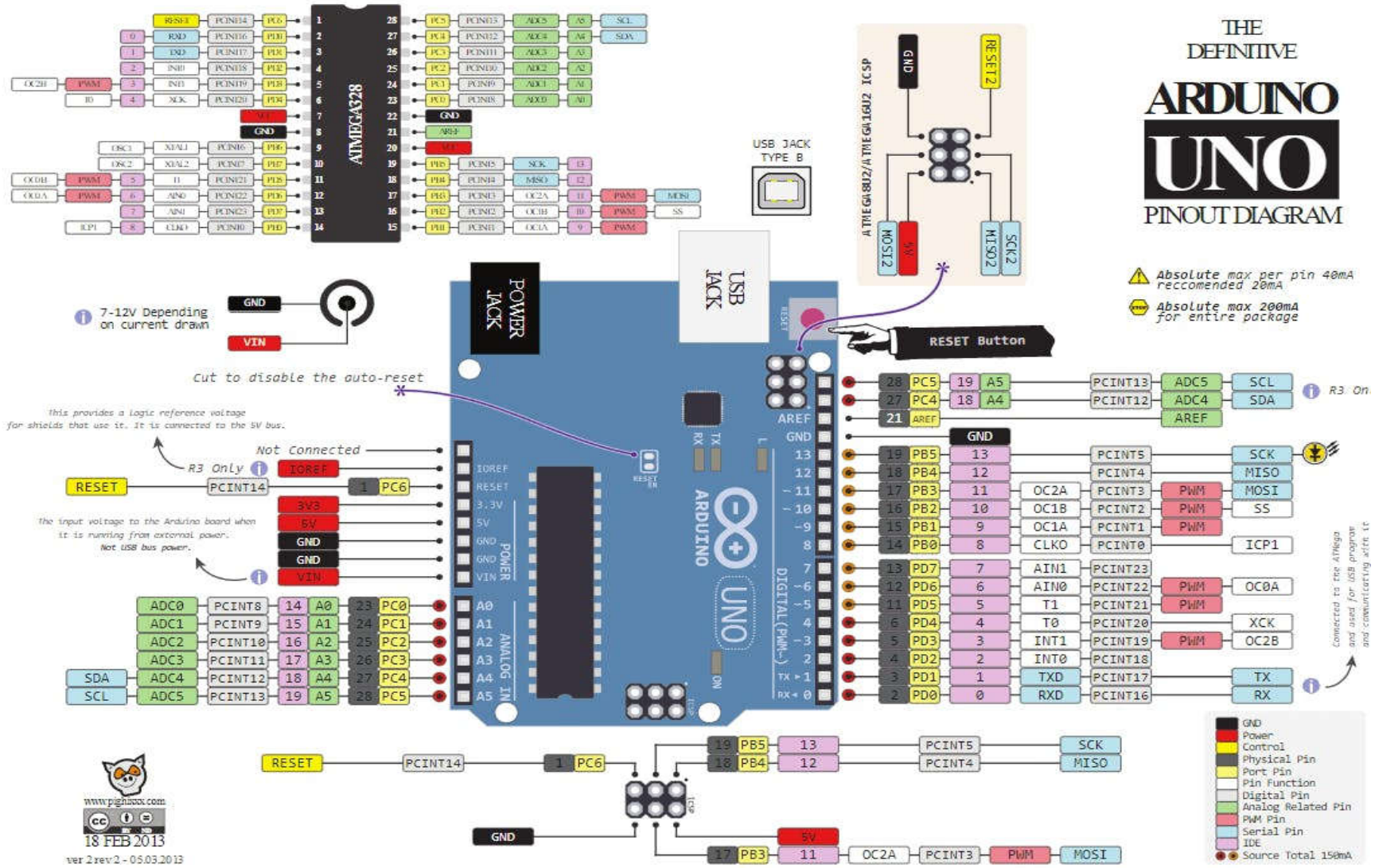
Contents

2.0 Objectives

- In this tutorial, you will learn:
  - To understand Arduino basic Inputs and Outputs (I/O's).
  - To know about the Importance of an Light Emitting Diodes (LEDs).
  - To become familiar with the types of Resistors.
  - To understand the difference between Pull-up and Pull-down in Buttons.
  - To know how to use Breadboard and Jumper Wires
  - To be able to start Arduino Programming using Blinking LED Project: "Hello World" of embedded systems.
  - To appreciate how the Blinking LED code works

2.1 Arduino I/O's

- **Arduino Pinouts Guide**



2.1 Arduino I/O's

- **Arduino I/O's**

Inputs:

Arduino pins are configured as INPUT by default.

Pins are configured in High-Impedance state typically >100Mohm impedance.

Outputs:

Pin configured as output is in low impedance state to drive as an output to other circuit. Atmega pins can be **source** (positive current source) or **sink** (negative current source) up to 40mA max.

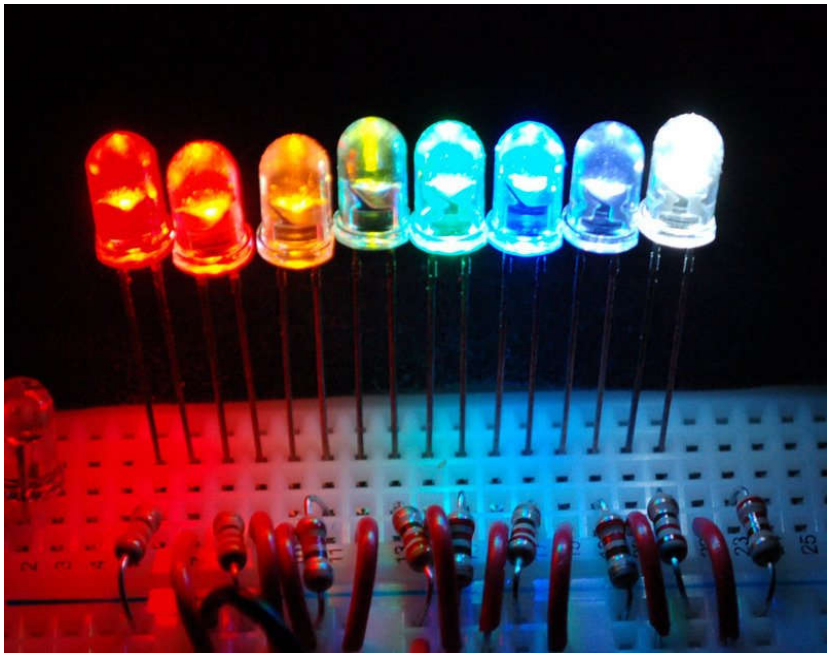
2.1 Arduino I/O's

Technical Specifications:

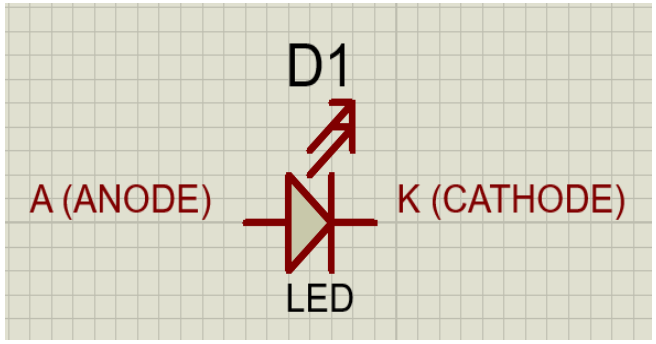
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

2.2 Light Emitting Diodes

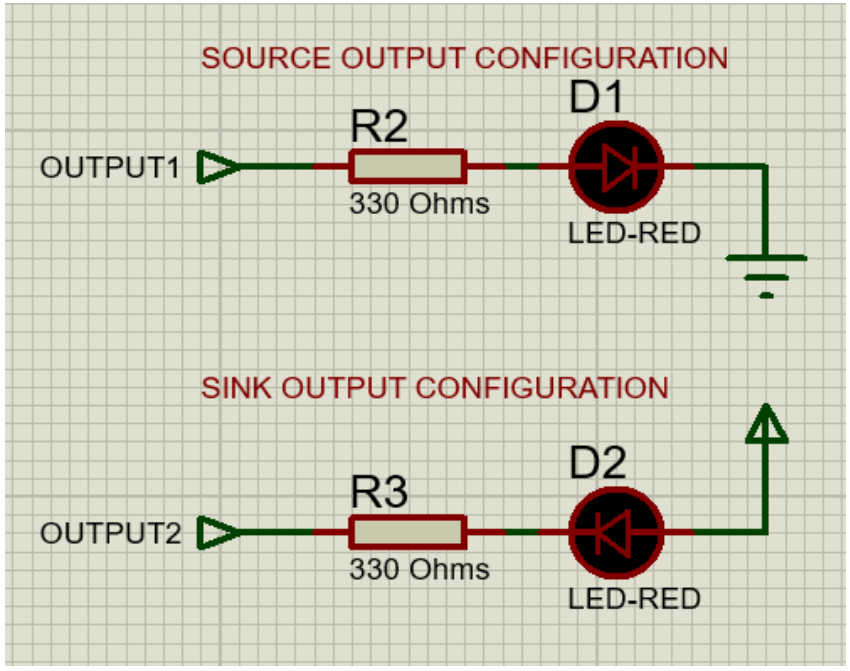
- **Light-emitting diode (LED)** is a semiconductor light source that emits light when current flows through it.
- In 1962, **Nick Holonyak Jr.** is an American engineer who invented the first LED that emitted visible red light while working at General Electric laboratory in New York.



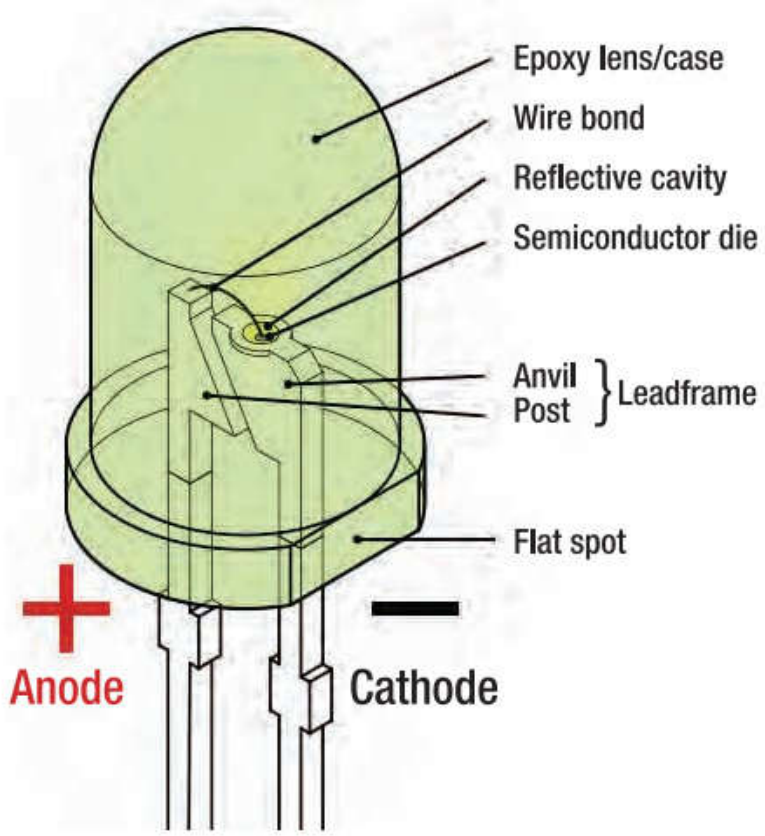
2.2 Light Emitting Diodes



LED SYMBOL



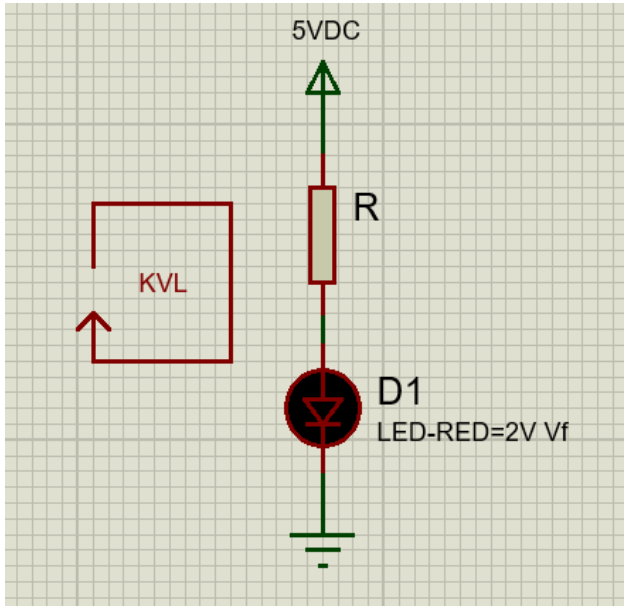
SOURCE/SINK OUTPUT TYPE



2.2 Light Emitting Diodes

Typical LED has a maximum forward current $\mathbf{I_f}$ has 25-30mA, but for simplicity, we use 10mA for nominal forward current and 2V for nominal forward voltage $\mathbf{V_f}$, so from KVL, 5-I*R-2=0; R = (5-2/10mA) thus:

R= 300 Ohms or 330 Ohms for common usage



Solving the Value of R

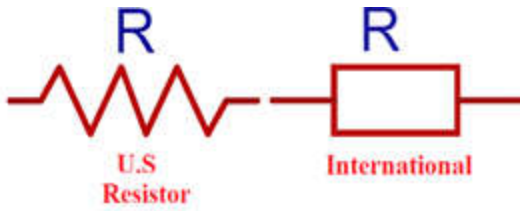
Colour	Approx. Forward Voltage Vf (V)
Red	1.7
HE Red*	2.0
Bright Red	2.3
Orange	2.0
Yellow	2.1
Green	2.2
Blue	3.2
White	3.2

*HE - High Efficiency

Typical LED Forward Voltage

2.3 Resistors

- **Resistor** is a passive two-terminal electrical component that implements electrical resistance to reduce the rate of flow of electrons (electrical current). The unit of resistance is Ohms and has a symbol of omega Ω.



- **Applications of Resistor**

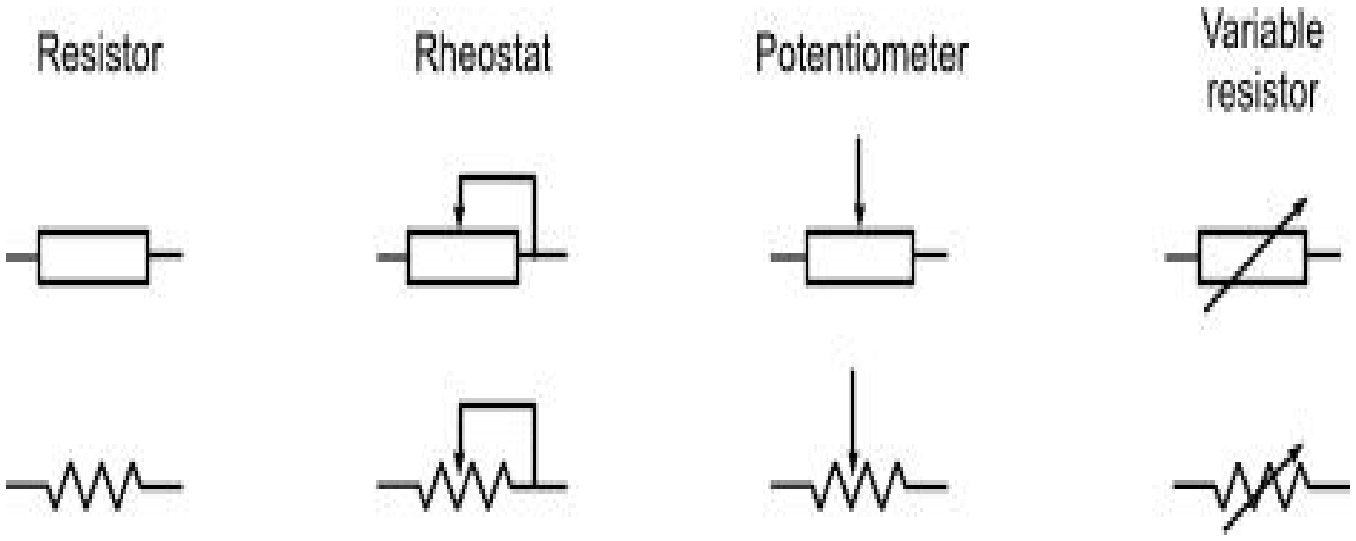
  Reduce current flow
  Adjust signal levels
  Divide voltages
  Bias active elements
  Terminate transmission lines
  Act as a passive load

2.3 Resistors

2 Types of Resistor

1. Fixed-resistance that only change slightly with temperature, time or operating voltage.

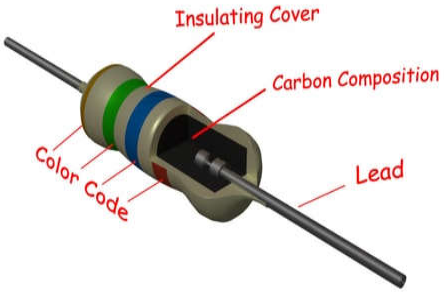
2. Variable-used to adjust circuit elements or as sensing devices for heat, light, humidity, force, or chemical activity.



2.3 Resistors

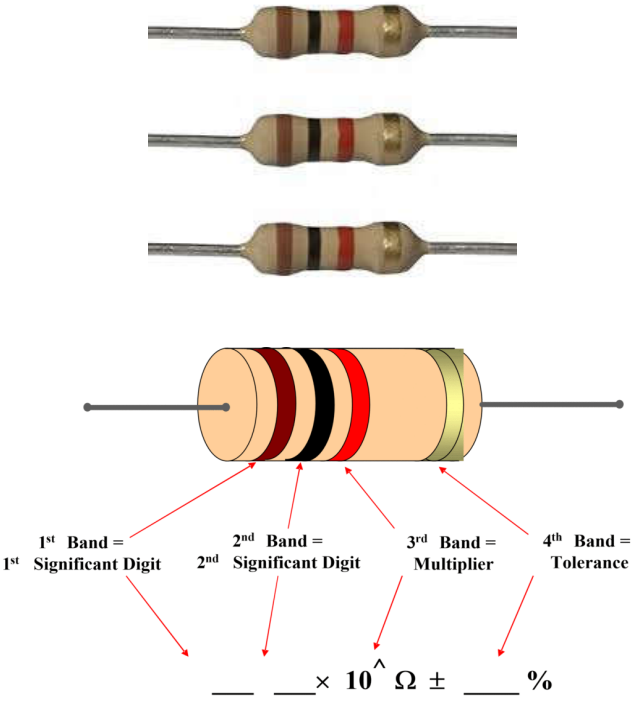
Types of Fixed Resistor

1. Carbon Composition



Color	Value	Multiplier	Tolerance
Black	0	x 10^0	±20%
Brown	1	x 10^1	± 1%
Red	2	x 10^2	± 2%
Orange	3	x 10^3	± 3%
Yellow	4	x 10^4	-0,+100%
Green	5	x 10^5	± 0.5%
Blue	6	x 10^6	± 0.25%
Violet	7	x 10^7	± 0.10%
Gray	8	x 10^8	±0.05%
White	9	x 10^9	± 10%
Gold	–	x 10^{-1}	± 5%
Silver	–	x 10^{-2}	± 10%
None	–	–	±20%

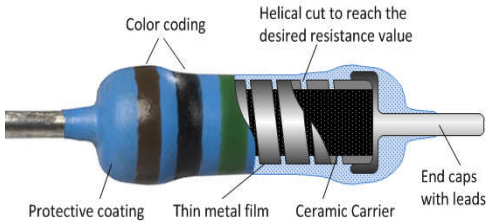
Color Code of Carbon Resistor



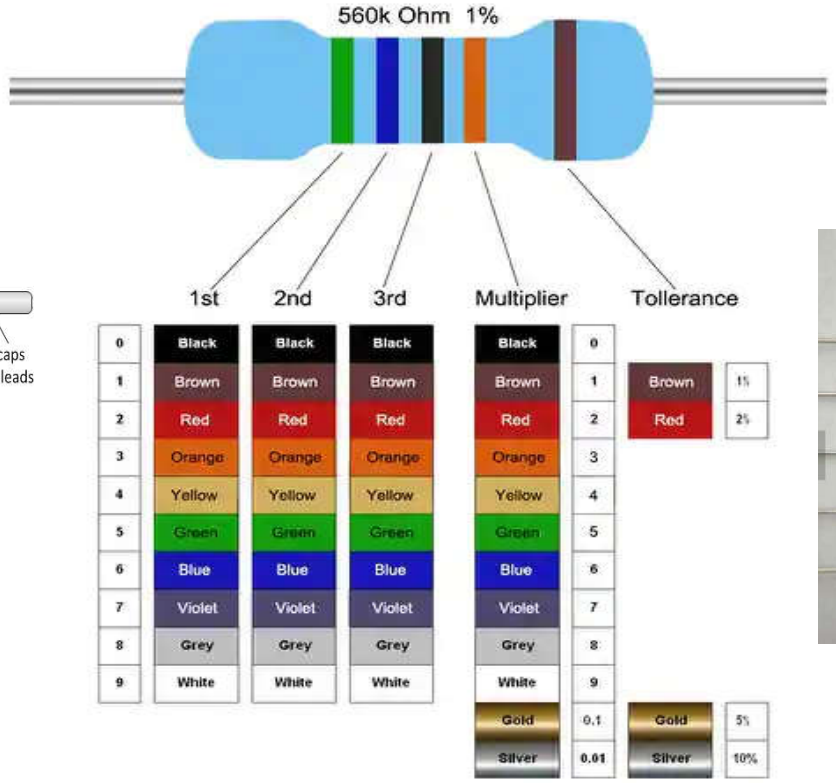
2.3 Resistors

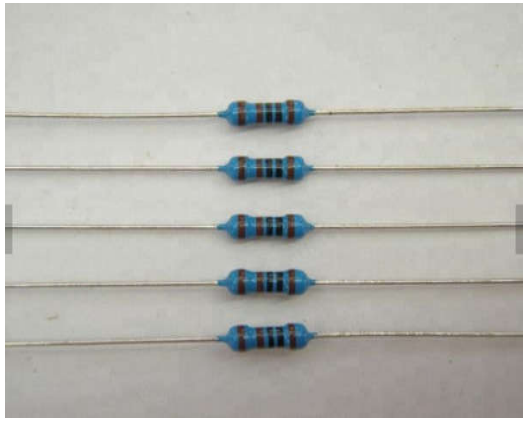
Types of Fixed Resistor

2. Metal Film



Find Tollerance Band (Usually Separated) and work from other side

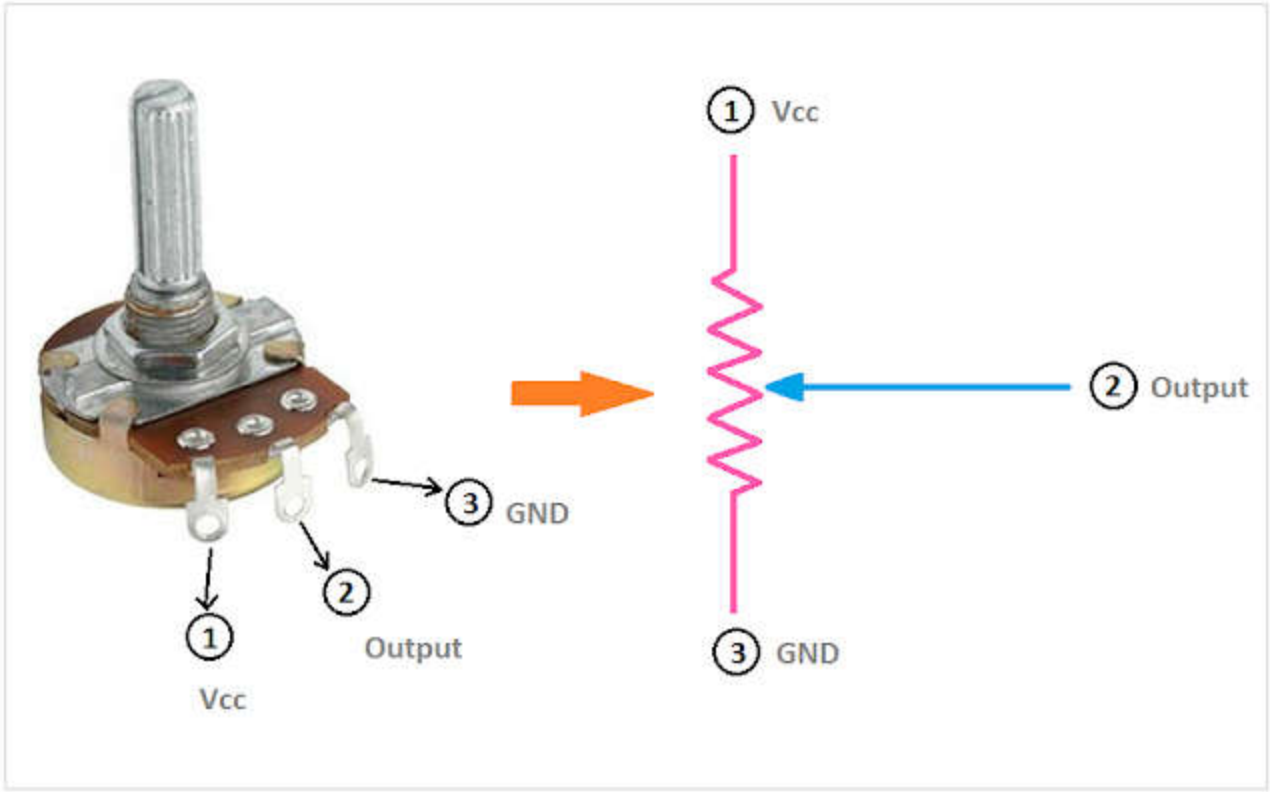




2.3 Resistors

Types of Variable Resistor

1. Potentiometer

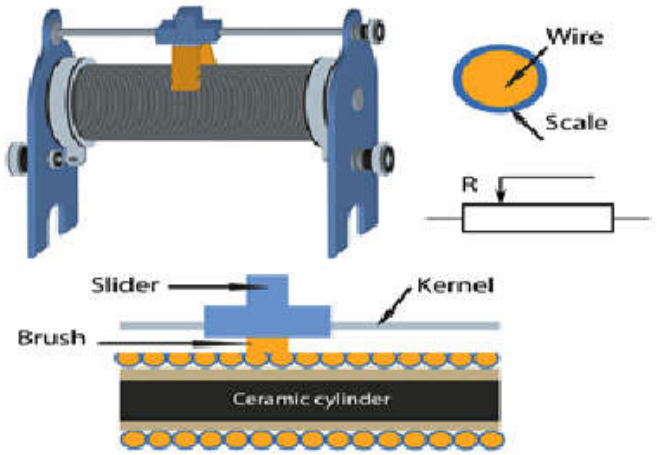


2.3 Resistors

Types of Variable Resistor

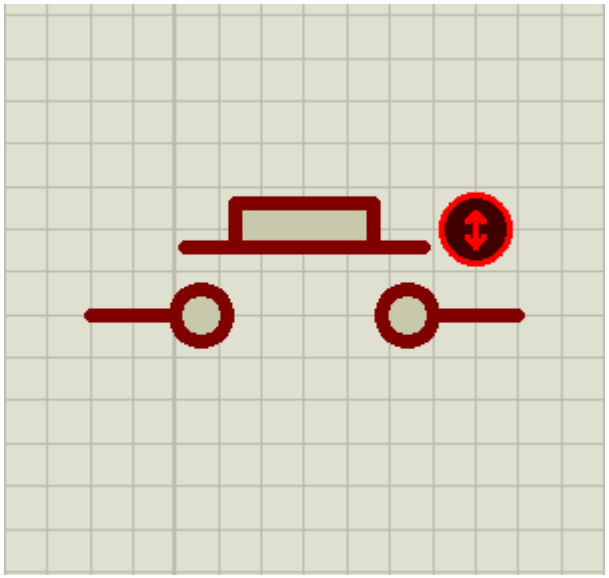
2. Rheostat

Practical Rheostats



2.4 Buttons: Pull-up and Pull-down

- Push Buttons

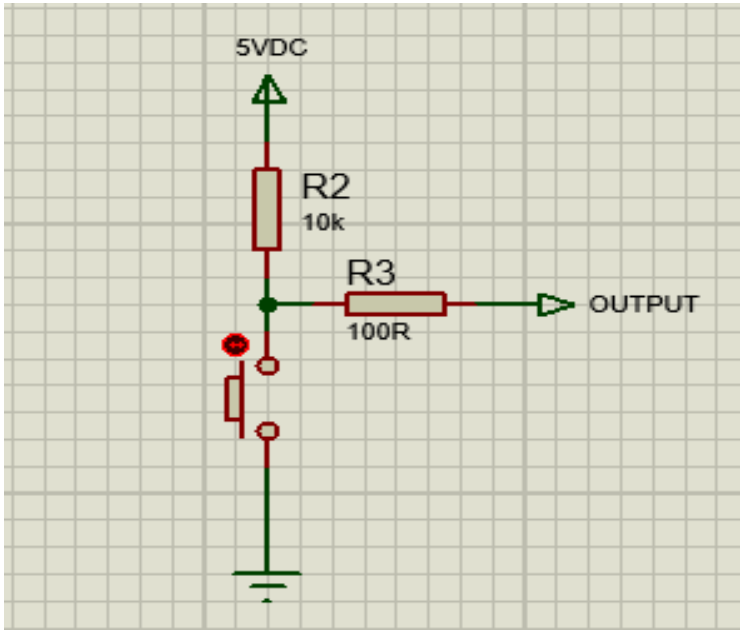


- Contact Bounce

Figure 1

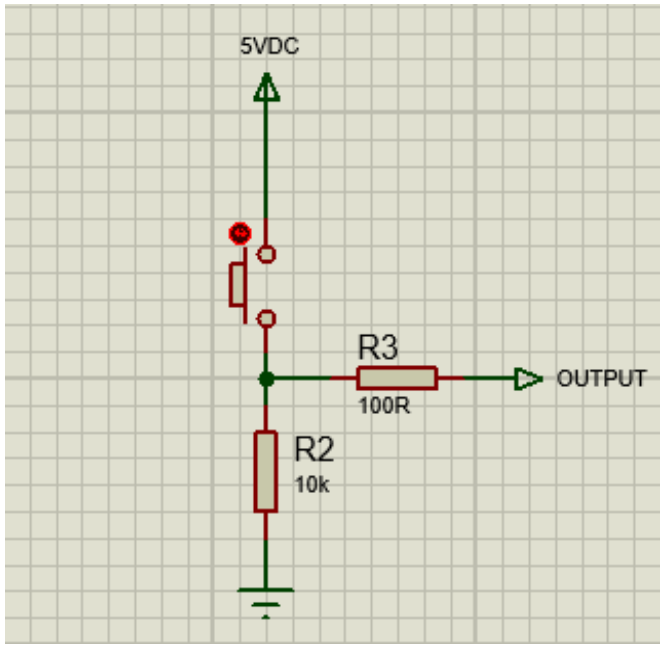
2.4 Buttons: Pull-up and Pull-down

- Pull-up
  - Active LOW Configuration



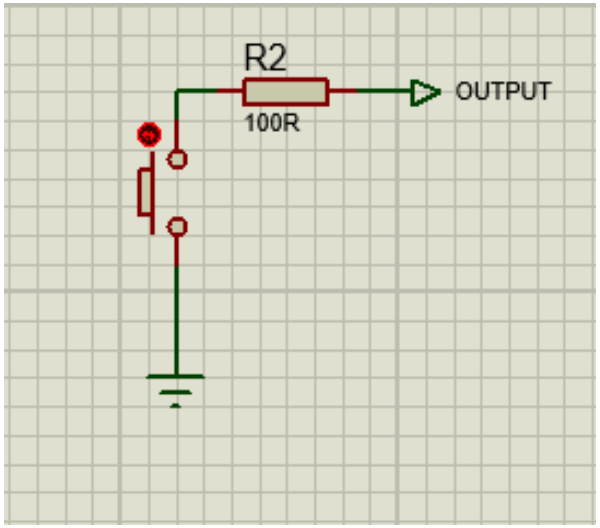
2.4 Buttons: Pull-up and Pull-down

- Pull-down
  - Active HIGH Configuration

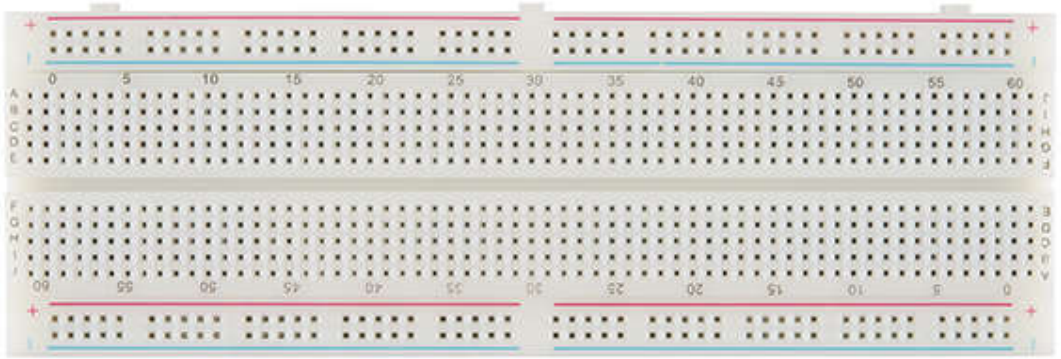


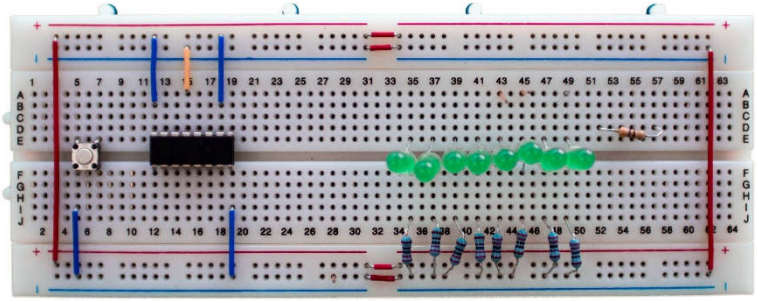
2.4 Buttons: Pull-up and Pull-down

- Weak Pull-up
  - Active HIGH Configuration



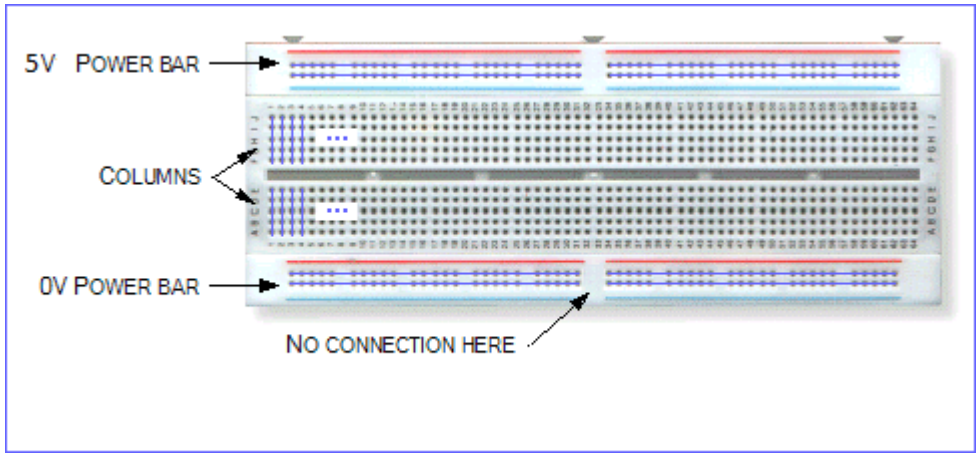
2.5 Breadboard and Jumper Wires

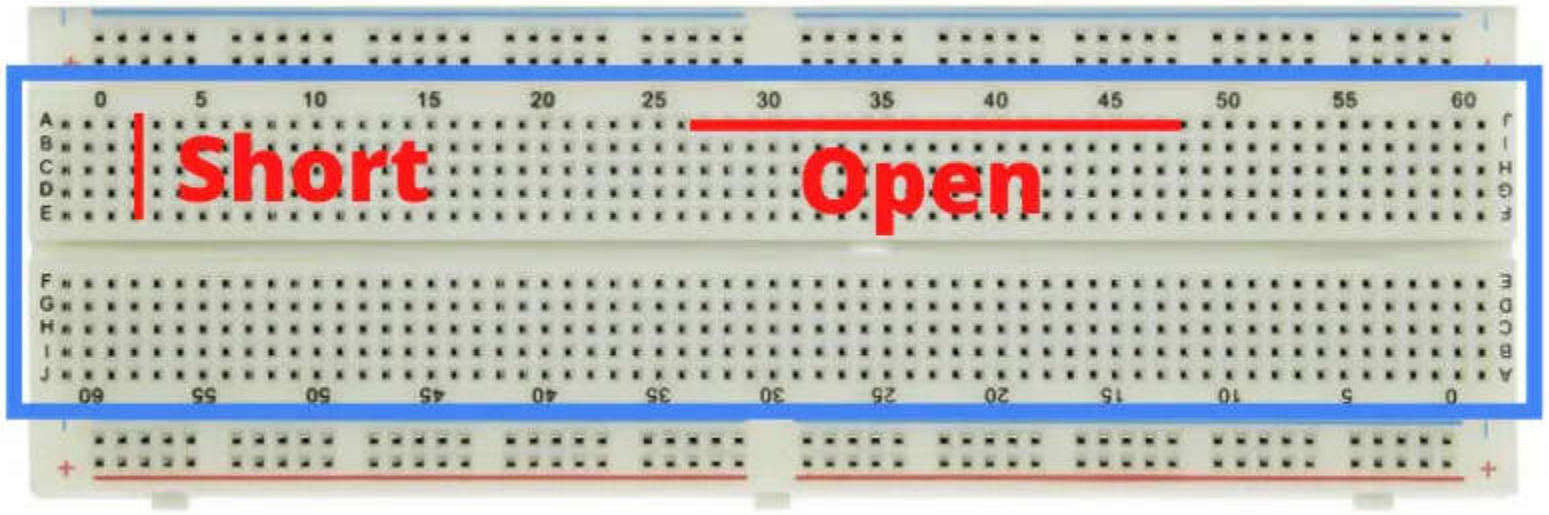




2.5 Breadboard and Jumper Wires

Breadboard Connection





2.6 Blinking LED Project: "Hello World" of Embedded Systems

- **Blinking LED- Hello World of Embedded Systems**

```
//Example 2.6 - BlinkingLED.ino
//Author: Rally Uminga
//Description: Blinking LED in pin 13 of Arduino Uno

int LED13 = 13;              //D13 has built-in LED in Arduino Uno

void setup()                 //All initialization put here
{

pinMode(LED13, OUTPUT);      //Initialize pin 13 as output

}

void loop()                  //The main program that is repeated all over again
{

digitalWrite(LED13, HIGH);   //Writing the pin 13 into output high
delay(1000);                 //Delay 1000 millisecond of Arduino delay function
digitalWrite(LED13, LOW);    //Writing the pin 13 into output low
delay(1000);                 //Delay 1000 millisecond of Arduino delay function

}
```

2.7 Code Explanation

- **Comment**

C++ Style comment- double slash for single line comment

```
//Example 1: BlinkingLED.ino
//Author: Rally Uminga
//Description: Blinking LED in pin 13 of Arduino Uno
```

or

C Style comment-multiple statements

```
/*Example 1: BlinkingLED.ino
Author: Rally Uminga
Description: Blinking LED in pin 13 of Arduino Uno*/
```

2.7 Code Explanation

- **Variables**

```
int LED13 = 13;//D13 has built-in LED in Arduino Uno
```

int is a signed 16 bit integer data type from -32768 to 32767 values.

LED13 is a variable name replacing the pinout 13 of the Arduino Uno

2.7 Code Explanation

- **Arduino Data Types**

Data type	RAM	Number Range
void keyword	N/A	N/A
boolean	1 byte	0 to 1 (True or False)
byte	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
word	2 byte	0 to 65,535
long	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long	4 byte	0 to 4,294,967,295
float	4 byte	-3.4028235E+38 to 3.4028235E+38
double	4 byte	-3.4028235E+38 to 3.4028235E+38
string	1 byte + x	Arrays of chars
array	1 byte + x	Collection of variables

2.7 Code Explanation

- **Setup function prototype**

```
void setup()              //All initialization put here
{

pinMode(LED13, OUTPUT);//Initialize pin 13 as output

}
```

All initialization will be made here at void setup(). The code made here will only run once in entire program cycle.

2.7 Code Explanation

- **Loop function prototype**

```
void loop()    //The main program that is repeated all over
again
{

digitalWrite(LED13, HIGH);              //Writing the pin 13
into output high
delay(1000);   //Delay 1000 millisecond of Arduino delay
function
digitalWrite(LED13, LOW);                //Writing the pin 13
into output low
delay(1000);   //Delay 1000 millisecond of Arduino delay
function

}
```

The code will be executed repeatedly forever as part of a Arduino code or sketch to blink the LED on and off for 1 second.

LED Projects using ESP32

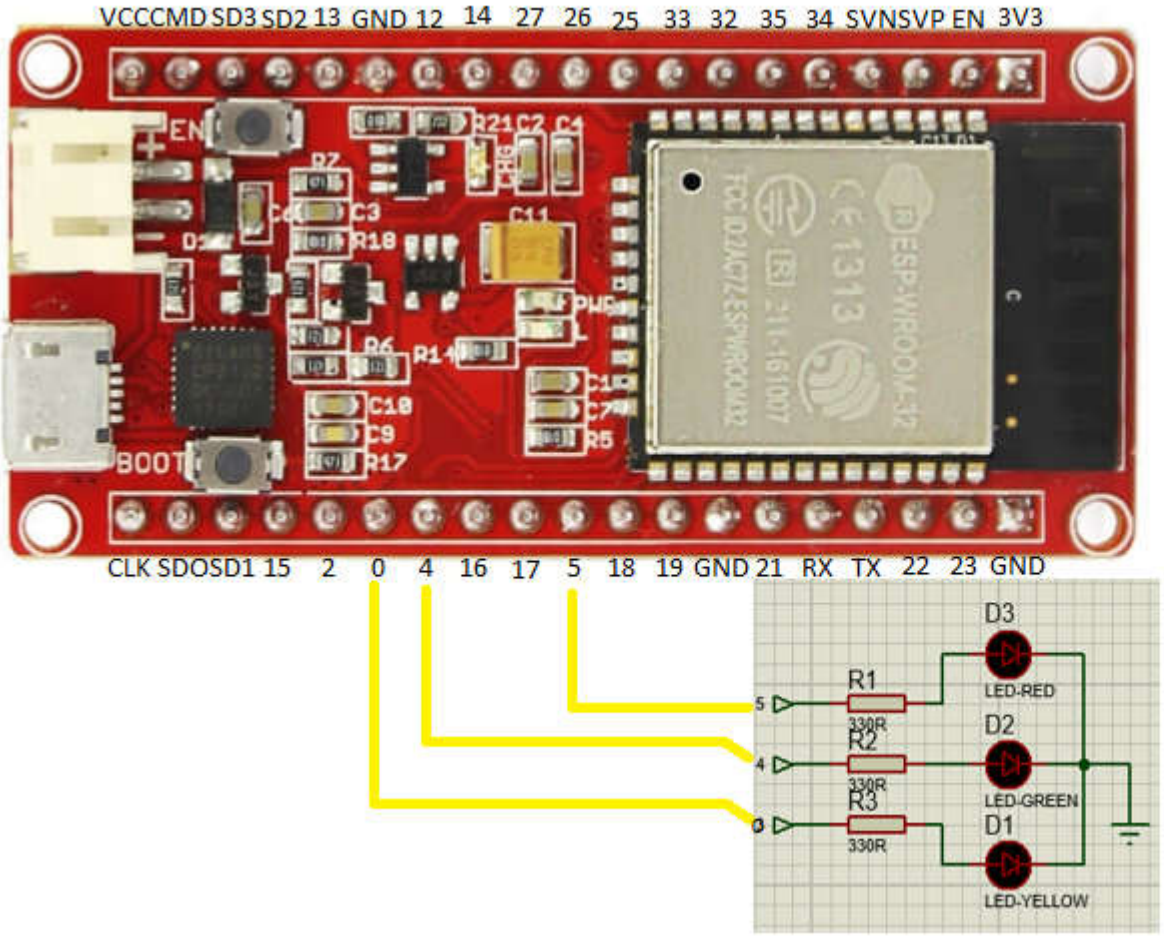
LED Projects

Contents

3.0 Objectives

- In this tutorial, you will learn:
  - To understand how to implement Traffic LEDs simulation in Arduino.
  - To know how to make Scrolling LEDs and potentiometer.
  - To become familiar using Pulsating LEDs with PWM output.
  - To understand how to make Mood Lamp using PWM output.
  - To appreciate how use LED Effects.

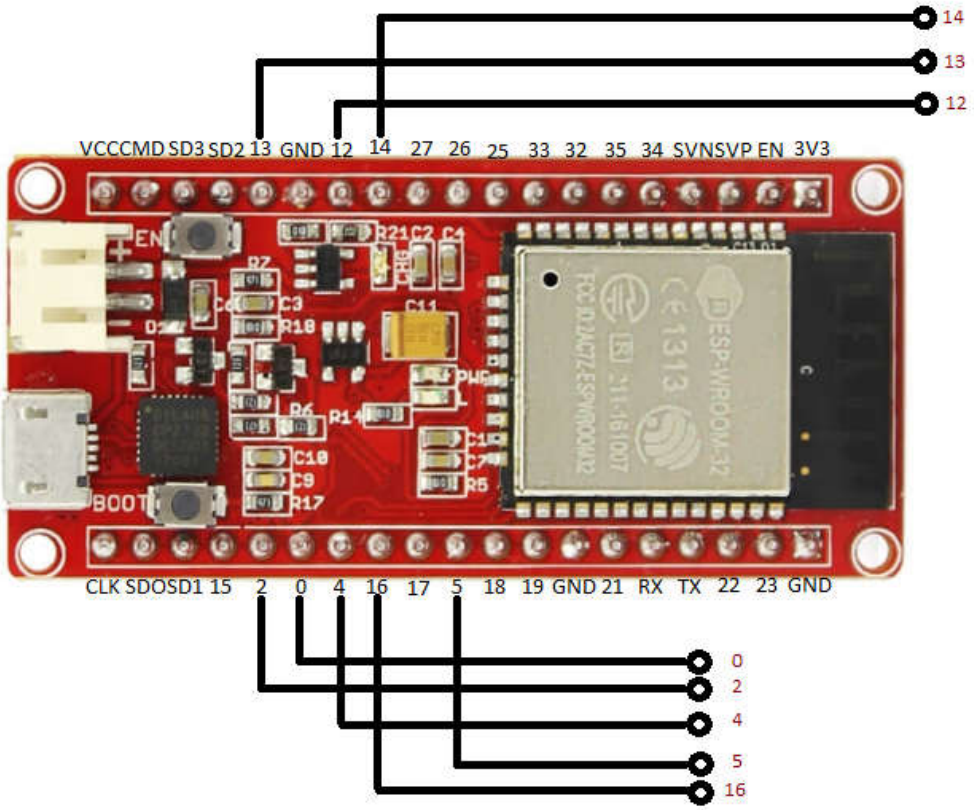
3.1 Traffic Light LEDs

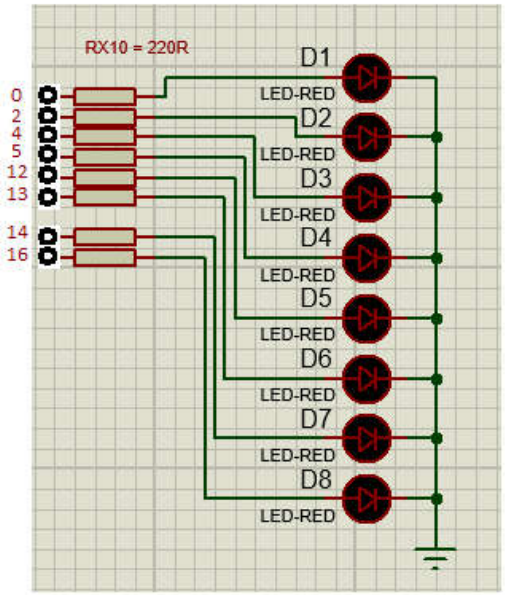


3.1 Traffic Light LEDs

```
// Example 3.1 – TrafficLightsLED_ESP32.ino

int LED_DELAY = 10000;              //Traffic Lights Delay
int RED_LED = 5;                    //Set RED_LED to pin 5
int YELLOW_LED = 4;                 //Set RED_LED to pin 4
int GREEN_LED = 0;                  //Set RED_LED to pin 0

void setup()
{

pinMode(RED_LED, OUTPUT);           //Initilalize RED_LED to output
pinMode(YELLOW_LED, OUTPUT);        //Initialize YELLOW_LED to output
pinMode(GREEN_LED, OUTPUT);         //Initialize GREEN_LED to output

}

void loop()
{

digitalWrite(RED_LED, HIGH);         // RED_LED ON
delay(LED_DELAY);                    // 10 seconds delay
digitalWrite(YELLOW_LED, HIGH);      // YELLOW_LED ON
delay(2000);                         // 2 seconds delay
digitalWrite(GREEN_LED, HIGH);       // GREEN_LED ON
digitalWrite(RED_LED, LOW);          // RED_LED OFF
digitalWrite(YELLOW_LED, LOW);       // YELLOW_LED OFF
delay(LED_DELAY);                    // 10 seconds delay
digitalWrite(YELLOW_LED, HIGH);      // YELLOW_LED ON
digitalWrite(GREEN_LED, LOW);        // GREEN_LED OFF
delay(2000);                         // 2 seconds delay
digitalWrite(YELLOW_LED, LOW);       // YELLOW_LED OFF
                                     // loop forever
}
```

3.2 Scrolling LEDs





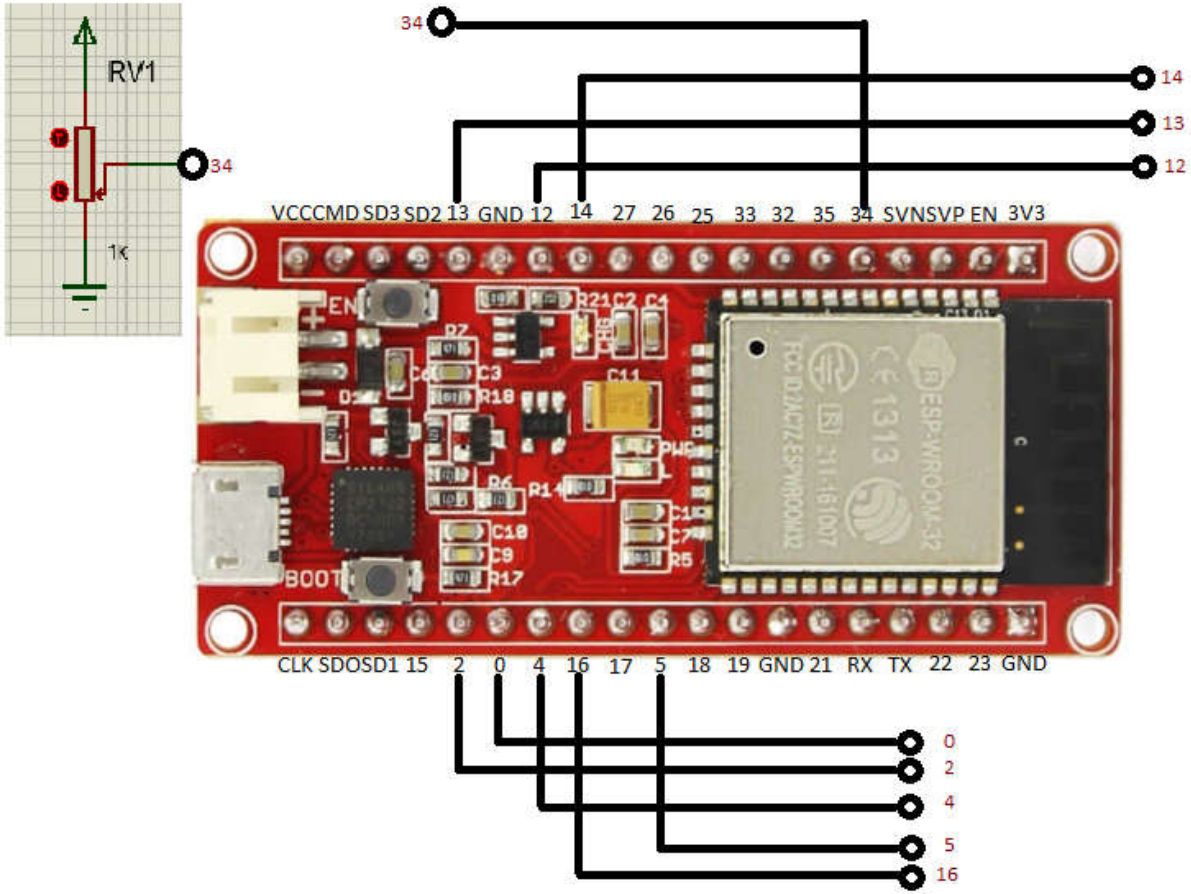
3.2 Scrolling LEDs

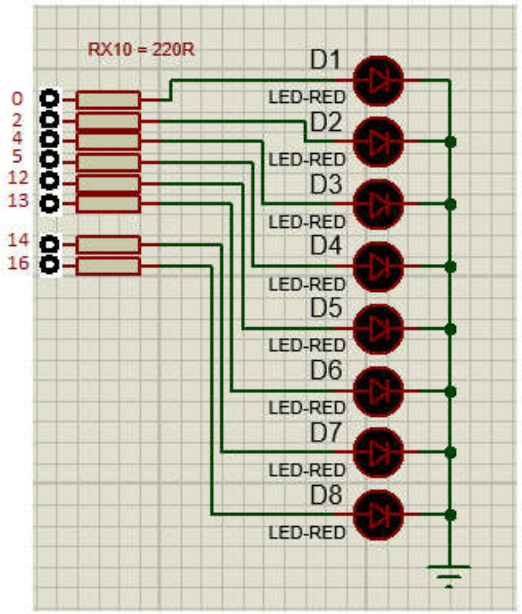
```
// Example 3.2 – ScrollingLED_32.ino
byte LED_Pins[] = {0, 2, 4, 5, 12, 13, 14, 16};          // LED pins array
int LED_Delay(65);                                       // Delay between LED changes
int LED_direction = 1;                                   // Direction flag
int current_LED = 0;                                     // Initialize LED direction
unsigned long elapsed_Time;                              // Elapsed time value for millis

void setup()
{
    for (int x=0; x<8; x++)
    {                                                    // Loop to set all pins to output
        pinMode(LED_Pins[x], OUTPUT);
    }
        elapsed_Time = millis();                         // Equate elapsedTime to millis()
}

void loop()
{
    if ((millis() - elapsed_Time) > LED_Delay)
    {                                                    // LED_Delay ms since last change
      change_LED();
      elapsed_Time = millis();
    }
}
```

3.2 Scrolling LEDs

```
void change_LED() {
    for (int x=0; x<10; x++) {                          // Turn OFF all LED's
       digitalWrite(LED_Pins[x], LOW);
    }
    digitalWrite(LED_Pins[current_LED], HIGH);          // Turn ON the current LED
    current_LED += LED_direction;                       // Increment by the direction value
                                                        // Change direction if we reach the end
    if (current_LED == 7) {LED_direction = -1;}
    if (current_LED == 0) {LED_direction = 1;}
}
```

3.3 Scrolling LEDs with Potentiometer





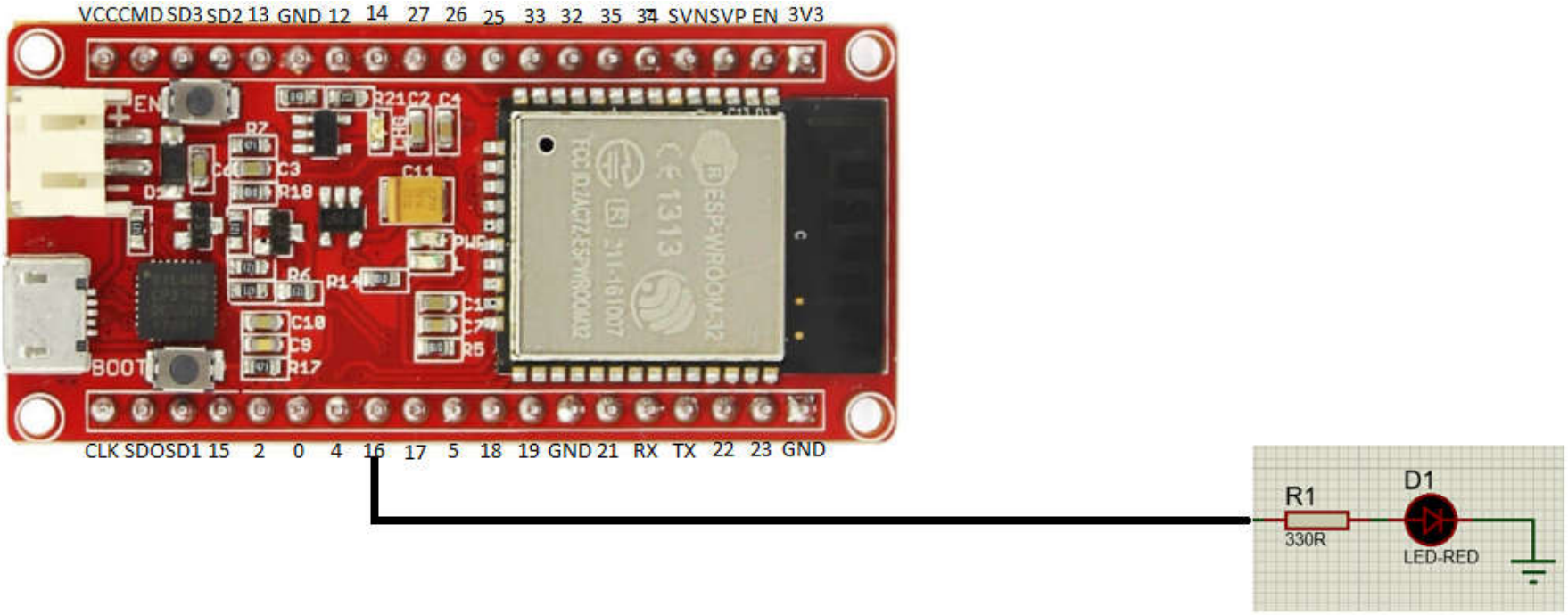
3.3 Scrolling LEDs with Potentiometer

```
// Example 3.3 - ScrollingLED_POT_32.ino
byte LED_Pins[] = {0, 2, 4, 5, 12, 13, 14, 16};   // LED pins array
int LED_Delay;                                      // Delay between LED changes
int LED_direction = 1;                              // Direction flag
int current_LED = 0;                                // Initialize LED direction
int pot_Pin = 34;                                   // A1 is input pin for the potentiometer
unsigned long elapsed_Time;                         // Elapsed time value for millis

void setup()
{
    for (int x=0; x<8; x++)
    {                                               // Loop to set all pins to output
        pinMode(LED_Pins[x], OUTPUT);
    }
        elapsed_Time = millis();                    // Equate elapsedTime to millis()
}

void loop()
{
    LED_Delay = analogRead(pot_Pin);                 // Read the value from the pot in A1

    if ((millis() - elapsed_Time) > LED_Delay)
    {                                               // LED_Delay ms since last change
      change_LED();
      elapsed_Time = millis();
    }
}
```

3.3 Scrolling LEDs with Potentiometer

```
void change_LED()
{


    for (int x=0; x<8; x++) {                          // Turn OFF all LED's
       digitalWrite(LED_Pins[x], LOW);
    }
    digitalWrite(LED_Pins[current_LED], HIGH);          // Turn ON the current LED
    current_LED += LED_direction;                       // Increment by the direction value

    if (current_LED == 7) {LED_direction = -1;}         // Change direction if we reach the end
    if (current_LED == 0) {LED_direction = 1;}
}
```

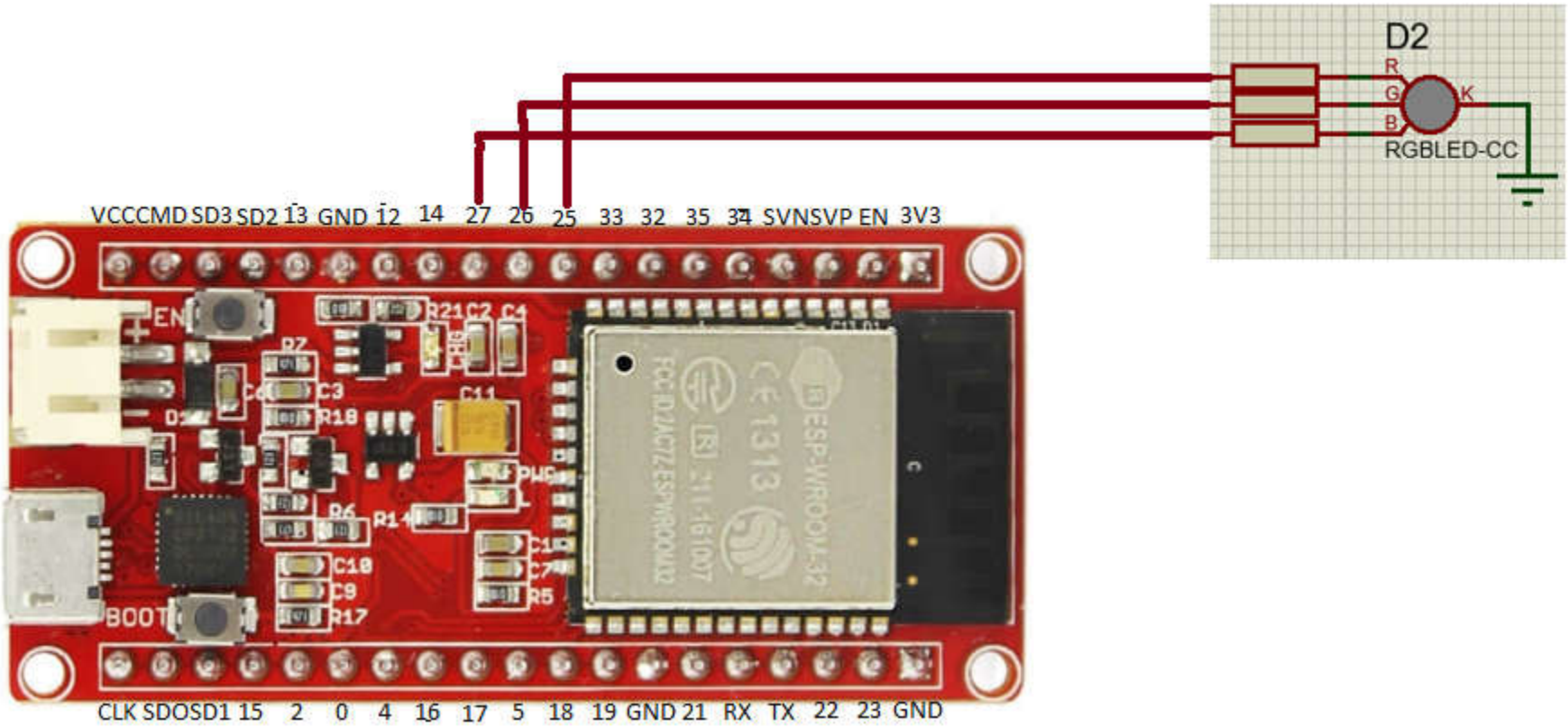
3.4 Pulsating LED



3.4 Pulsating LED

```
// Example 3.4 - PulsatingLED_32.ino
const int ledPin = 16;                          // Pin 16 PWM output
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;


void setup()
{

    //pinMode(ledPin, OUTPUT);           // Initialize LED_Pin to output
    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);

    // attach the channel to the GPIO to be controlled
    ledcAttachPin(ledPin, ledChannel);

}

void loop()
{
    // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }

  // decrease the LED brightness
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

3.5 Mood Lamp


D2
R
G
B
K
RGBLED-CC
VCCCMD SD3 SD2 13 GND 12 14 27 26 25 33 32 35 34 SVNSVP EN 3V3
CLK SDOSD1 15 2 0 4 16 17 5 18 19 GND 21 RX TX 22 23 GND
EN
BOOT
ESP-WROOM-32

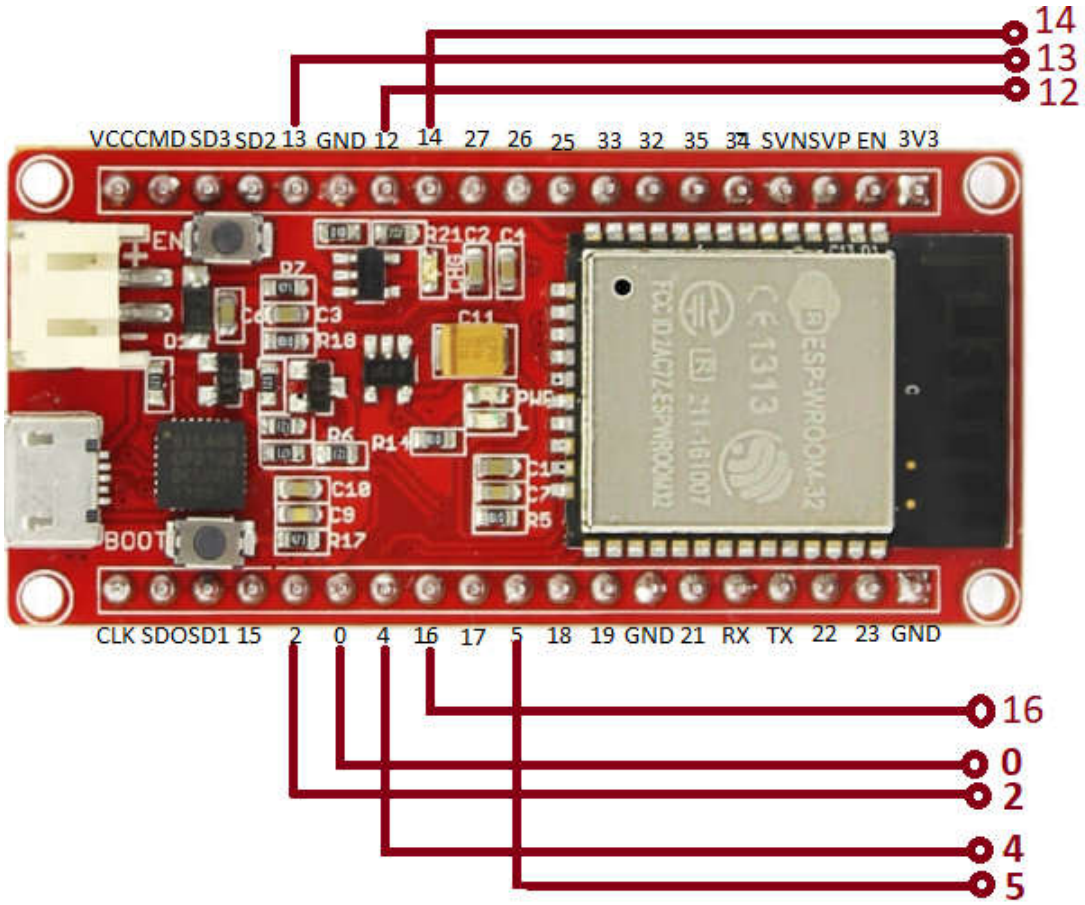
3.5 Mood Lamp

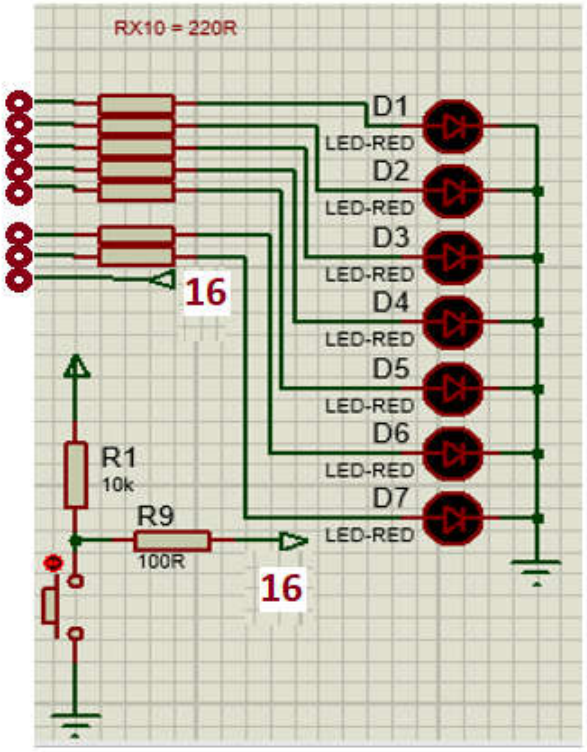
```
// Example 3.5 - MoodLampLED_32.ino
#define LEDR 25 // RED pin of rgb led is connected to 25 gpio pin
 #define LEDG 26 // green pin is connected to 26 gpio
 #define LEDB 27 //
 #define R_channel 0
 #define G_channel 1
 #define B_channel 2
 #define pwm_Frequency 5000 // pwm frequency
 #define pwm_resolution 8 // 8 bit resolution
 void setup() {
  ledcAttachPin(LEDR, R_channel);
  ledcAttachPin(LEDG, G_channel);
  ledcAttachPin(LEDB, B_channel);
  ledcSetup(R_channel, pwm_Frequency, pwm_resolution);
  ledcSetup(G_channel, pwm_Frequency, pwm_resolution);
  ledcSetup(B_channel, pwm_Frequency, pwm_resolution);
 }
 void loop() {
  RGB_Color(255,0,0); // RED ccolor
  delay(500);
  RGB_Color(0,255,0); // green color
  delay(500);
  RGB_Color(0,0,255); // blue color
  delay(500);
  RGB_Color(255,255,0); // yellow color
  delay(500);
  RGB_Color(0,255,255); // cyan color
   delay(500);
  RGB_Color(255,0,255); // magenta color
   delay(500);
  RGB_Color(255,20,147); // deep pink color
 }
```

3.5 Mood Lamp

```
void RGB_Color(int i, int j, int k)
 {
  ledcWrite(R_channel, i);
  ledcWrite(G_channel, j);
  ledcWrite(B_channel, k);
 }
```

3.6 Button and LED Effects





3.6 Button and LED Effects

```
// Example 3.8 - LEDEffectsButton_32.ino
int LED_12 = 14;
int LED_11 = 13;
int LED_10 = 12;
int LED_9 = 5;
int LED_8 = 4;
int LED_7 = 2;
int LED_6 = 0;

int buttonPin_5 = 16;  //the number of the pushbutton pin

int delay_time = 100;                // time delay

int p = 0;                           // variable for pattern

bool buttonState = 1;                // variable for reading the pushbutton status

void setup()
{
  pinMode(LED_12, OUTPUT);
  pinMode(LED_11, OUTPUT);
  pinMode(LED_10, OUTPUT);
  pinMode(LED_9, OUTPUT);
  pinMode(LED_8, OUTPUT);
  pinMode(LED_7, OUTPUT);
  pinMode(LED_6, OUTPUT);

  pinMode(buttonPin_5, INPUT);
}
```

3.6 Button and LED Effects

```
void loop()
{

 buttonState = digitalRead(buttonPin_5);

 if (buttonState == LOW)

    {
      p++;
      delay(50);
    }

  if(p==1)
  {
 digitalWrite(LED_12,1);
 digitalWrite(LED_11,0);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //1
 delay(delay_time);

 digitalWrite(LED_12,0);
 digitalWrite(LED_11,1);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //2
 delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //7
delay(delay_time);
 }
 if(p==2)
 {
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //7
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);


digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);

digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
```

3.6 Button and LED Effects

```
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //1
delay(delay_time);

 }
   if(p==3)
 {
digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //1
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //6
delay(delay_time);

 digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //5
delay(delay_time);

 digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //2
delay(delay_time);
 }
   if(p==4)
 {
digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //1,7
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //2,6
delay(delay_time);
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3,5
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

 }

     if(p==5)
 {

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,1);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //4
delay(delay_time);

digitalWrite(LED_12,0);
digitalWrite(LED_11,0);
digitalWrite(LED_10,1);
digitalWrite(LED_9,0);
digitalWrite(LED_8,1);
digitalWrite(LED_7,0);
digitalWrite(LED_6,0);  //3,5
delay(delay_time);
```

3.6 Button and LED Effects

```
digitalWrite(LED_12,0);
digitalWrite(LED_11,1);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,1);
digitalWrite(LED_6,0);  //2,6
delay(delay_time);

digitalWrite(LED_12,1);
digitalWrite(LED_11,0);
digitalWrite(LED_10,0);
digitalWrite(LED_9,0);
digitalWrite(LED_8,0);
digitalWrite(LED_7,0);
digitalWrite(LED_6,1);  //1,7
delay(delay_time);
 }
```

3.6 Button and LED Effects

```
 if(p==6)
{
 digitalWrite(LED_12,1);
 delay(delay_time);
 digitalWrite(LED_11,1);
 delay(delay_time);
 digitalWrite(LED_10,1);
 delay(delay_time);
 digitalWrite(LED_9,1);
 delay(delay_time);
 digitalWrite(LED_8,1);
 delay(delay_time);
 digitalWrite(LED_7,1);
 delay(delay_time);
 digitalWrite(LED_6,1);  //1,7
 delay(delay_time);
 digitalWrite(LED_6,0);  //1,7
 delay(delay_time);
 digitalWrite(LED_7,0);
 delay(delay_time);
 digitalWrite(LED_8,0);
 delay(delay_time);
 digitalWrite(LED_9,0);
 delay(delay_time);
 digitalWrite(LED_10,0);
 delay(delay_time);
 digitalWrite(LED_11,0);
 delay(delay_time);
 digitalWrite(LED_12,0);
 delay(delay_time);

}
```

3.6 Button and LED Effects

```
 if(p==7)
{
 digitalWrite(LED_12,0);
 digitalWrite(LED_11,0);
 digitalWrite(LED_10,0);
 digitalWrite(LED_9,0);
 digitalWrite(LED_8,0);
 digitalWrite(LED_7,0);
 digitalWrite(LED_6,0);  //1,7
  p=0;
}

}
```

Button and LED Projects using ESP8266

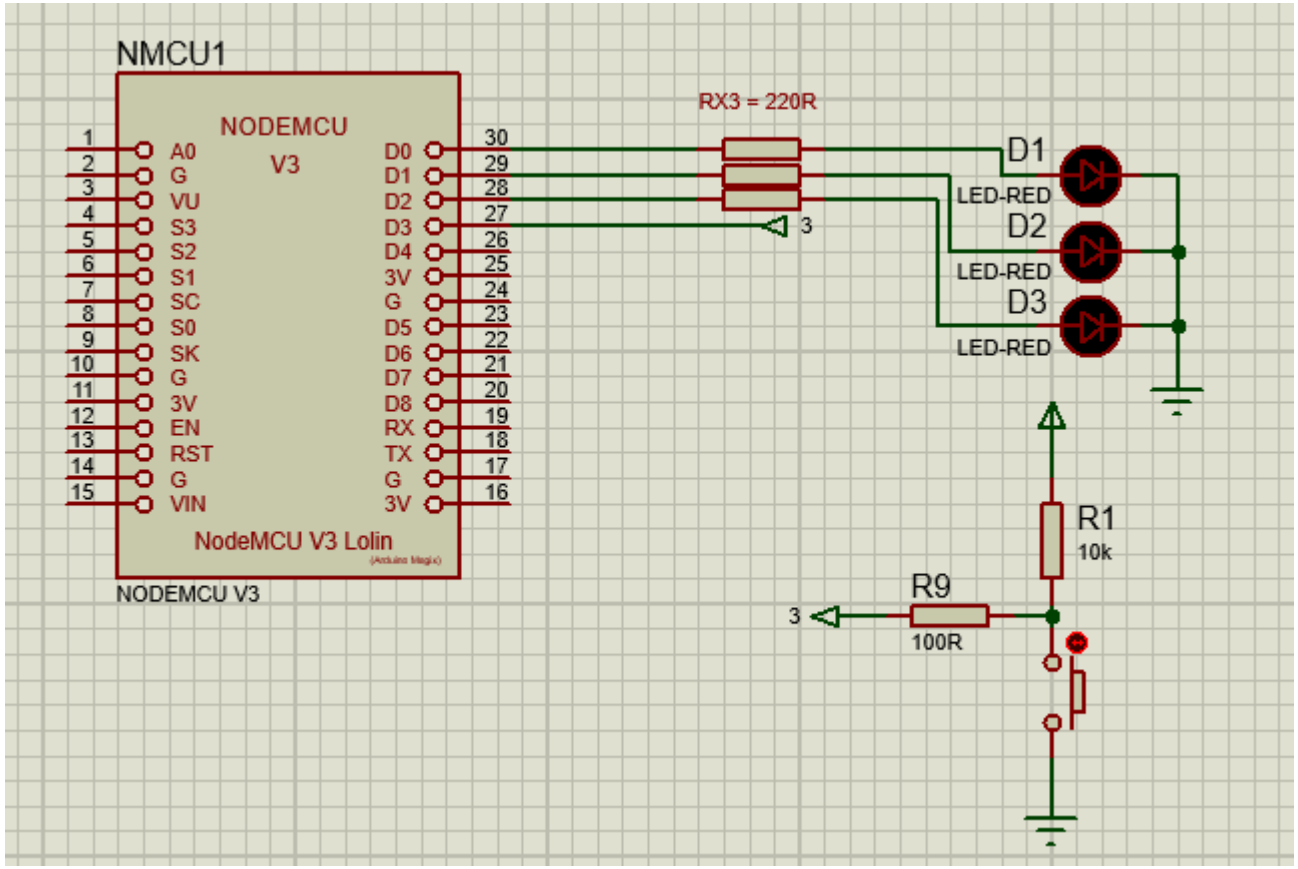
Button and LED Projects

Contents

4.0 Objectives

- In this tutorial, you will learn:
  - To understand how to implement LED_Button1.ino sketch.
  - To know how to make LED_Button2.ino sketch.
  - To become familiar using LED_Button3.ino sketch.
  - To understand how to make LED_Button4.ino sketch.
  - To know how to use LED_Button5.ino sketch.
  - To be able to start LED_Button6.ino sketch.
  - To appreciate how use LED_Button7.ino sketch.
  - To appreciate how use LED_Button8.ino sketch.

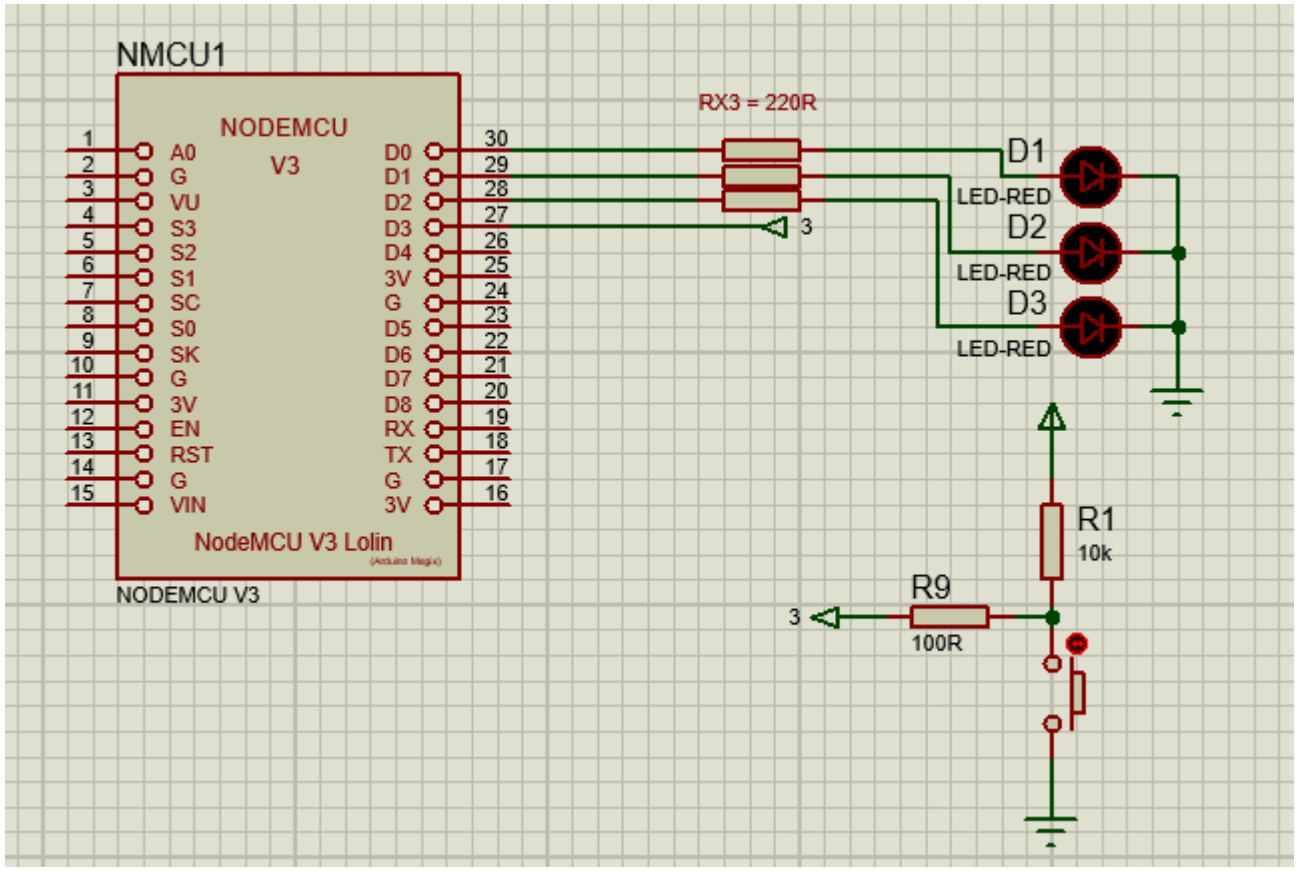
4.1 LED Button 1



4.1 LED Button 1

```
// Example 4.1 LED_Button1_8266.ino
// ESP8266 Pins to be used
int pin_LED0 = D0;                        // LED in pin D0
int pin_Button3 = D3;                     // Push Button in D3

void setup()
{

    pinMode(pin_LED0, OUTPUT);           // Initialize pin D0 to output
    digitalWrite(pin_LED0,LOW);          // Initialize pin D0 LOW during start
    pinMode(pin_Button3, INPUT);         // Initialize pin D3 as digital input
}

void loop()
{
    if ( digitalRead(pin_Button3) == LOW)// If push button was pressed
    {
       digitalWrite(pin_LED0, HIGH);     // LED pin D0 ON
    }
    else
    {
       digitalWrite(pin_LED0, LOW);      // If not pressed, pin D0 OFF
    }
}
```

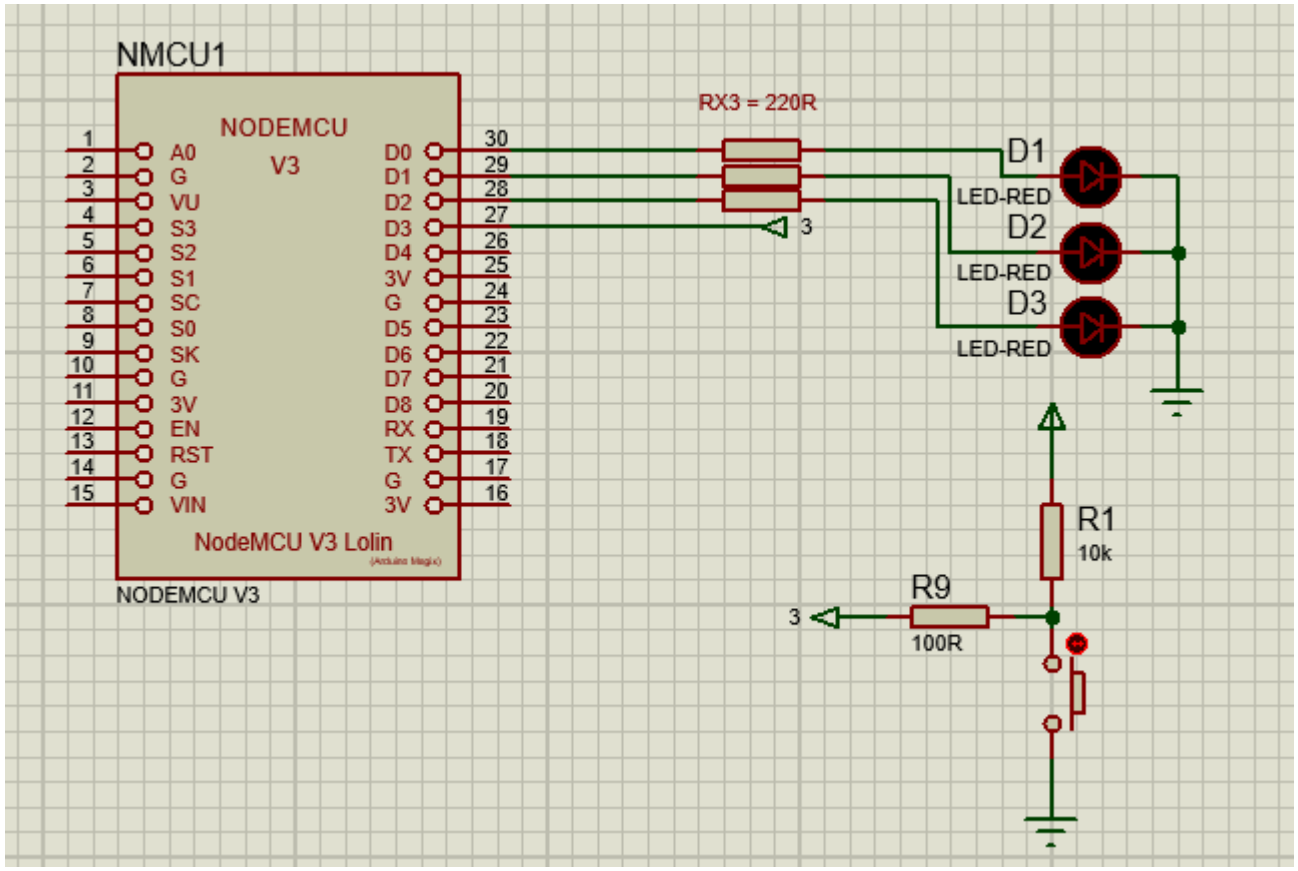
4.2 LED Button 2



4.2 LED Button 2

```
//  Example 4.2 - LED_Button2_8266.ino
//  Turning an LED in pin 8 on and off using a button switch in pin 2

// Arduino pins being used
int pin_LED0 = D0;
int pin_Button3 = D3;

// Variables new and old button states
boolean oldButton3State = HIGH;
boolean newButton3State = HIGH;

void setup()
{
    pinMode(pin_LED0, OUTPUT);
    digitalWrite(pin_LED0,LOW);
    pinMode(pin_Button3, INPUT);
}

void loop()
{
    newButton3State = digitalRead(pin_Button3);

    if ( newButton3State != oldButton3State )
    {
       if ( newButton3State == LOW )
       {
        digitalWrite(pin_LED0, HIGH);
       }
       else
       {
        digitalWrite(pin_LED0, LOW);
       }

       oldButton3State = newButton3State;
    }
}
```

4.3 LED Button 3



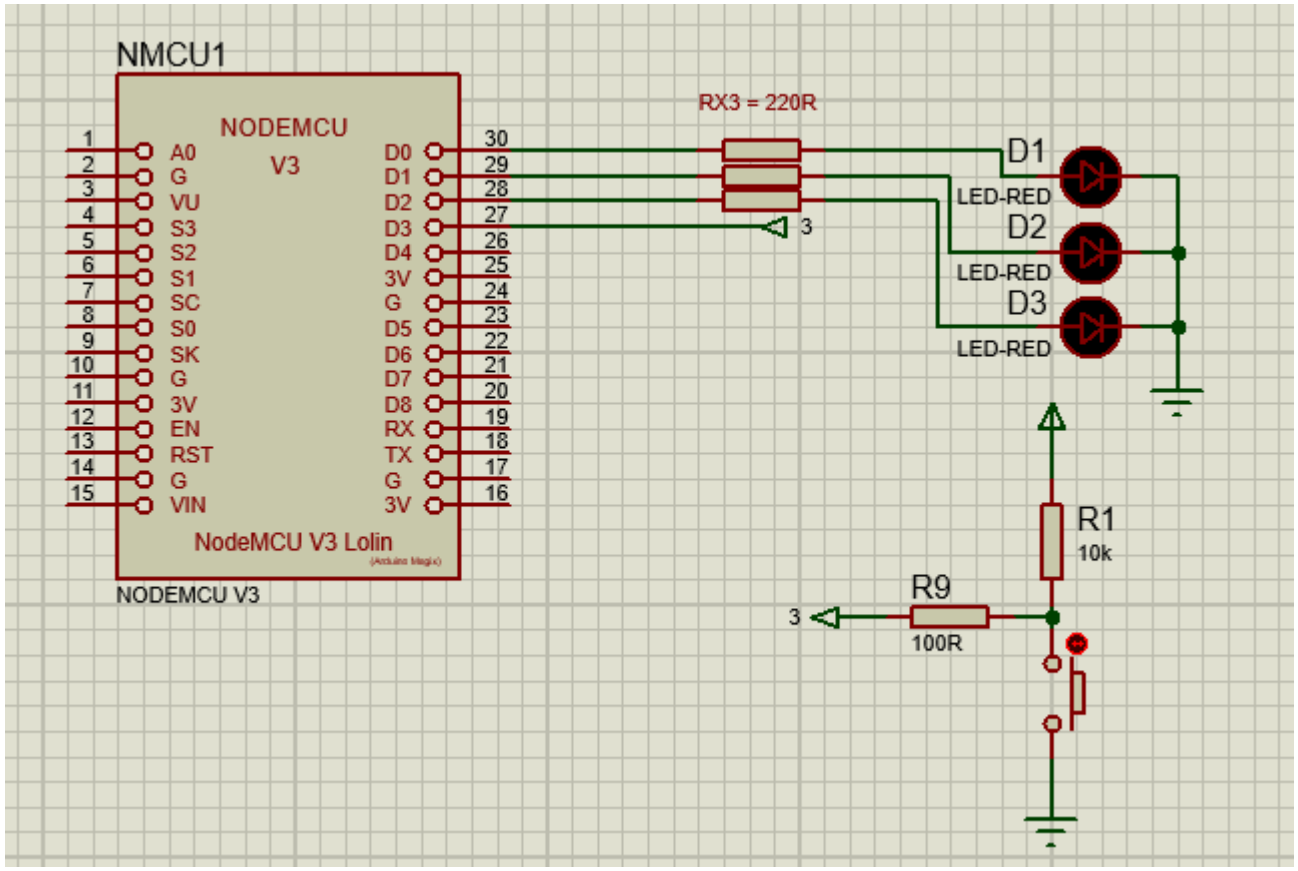
4.3 LED Button 3

```
//  Example 4.3 - LED_Button3_8266.ino
//  Button switch as a toggle switch to turn an LED on or off


// Define the pins being used
int pin_LED0 = D0;
int pin_Button3 = D3;

// Variables of new and old button states
boolean oldButton3State = HIGH;
boolean newButton3State = HIGH;

boolean LED0state = LOW;

void setup()
{
    pinMode(pin_LED0, OUTPUT);
    digitalWrite(pin_LED0,LOW);
    pinMode(pin_Button3, INPUT);
}
```

4.3 LED Button 3

```
void loop()
{
    newButton3State = digitalRead(pin_Button3);

    if ( newButton3State != oldButton3State )
    {
       // if pin_Button3 pressed
       if ( newButton3State == LOW )
       {
           if ( LED0state == LOW )
           { digitalWrite(pin_LED0, HIGH);
           LED0state = HIGH;
           }
           else
           { digitalWrite(pin_LED0, LOW);
           LED0state = LOW;
           }
       }
       oldButton3State = newButton3State;
    }
}
```

4.4 LED Button 4



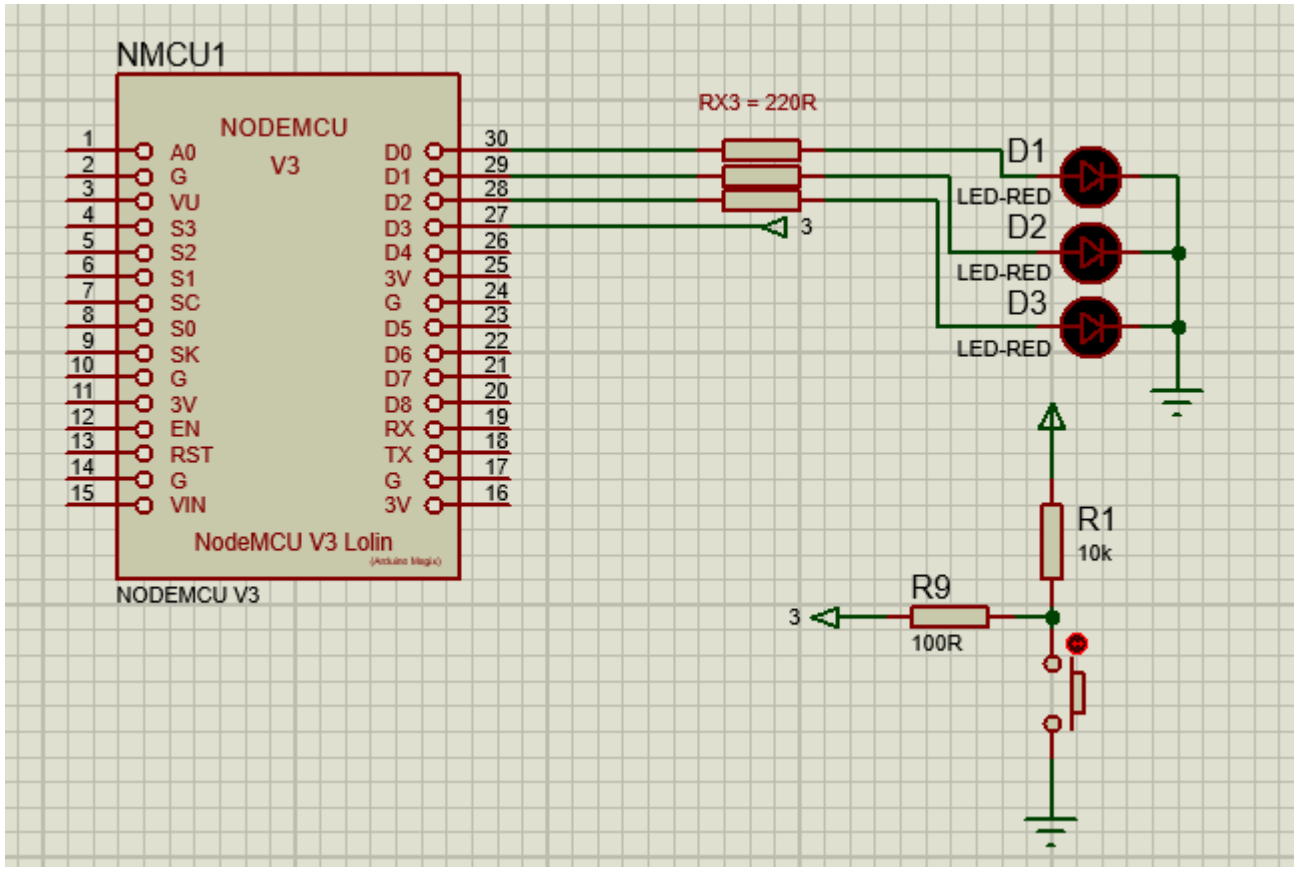
4.4 LED Button 4

```
//  Example 4.4 - Led_Button4_8266.ino
//  Using a button switch as a toggle switch to turn an LED on or off
//  with a simple debounce

// Define the pins being used
int pin_LED0 = D0;
int pin_Button3 = D3;

// new and old switch states
boolean oldButton3State = HIGH;
boolean newButton3State1 = HIGH;
boolean newButton3State2 = HIGH;
boolean newButton3State3 = HIGH;

// LED0 state
boolean LED0state = LOW;

void setup()
{
    pinMode(pin_LED0, OUTPUT);
    digitalWrite(pin_LED0,LOW);
    pinMode(pin_Button3, INPUT);
}
```

4.4 LED Button 4

```
void loop()
{
    newButton3State1 = digitalRead(pin_Button3);
    delay(1);
    newButton3State2 = digitalRead(pin_Button3);
    delay(1);
    newButton3State3 = digitalRead(pin_Button3);

    // if all 3 values are the same we can continue
    if (  (newButton3State1==newButton3State2) && (newButton3State1==newButton3State3) )
    {

        if ( newButton3State1 != oldButton3State )
        {

           // has the button switch been closed?
           if ( newButton3State1 == LOW )
           {
               if ( LED0state == LOW )
               {
                  digitalWrite(pin_LED0, HIGH);
                  LED0state = HIGH;
               }
               else
               {
                  digitalWrite(pin_LED0, LOW);
                  LED0state = LOW;
               }
           }
           oldButton3State = newButton3State1;
        }
    }
}
```

4.5 LED Button 5



4.5 LED Button 5

```
//  Example 4.5 - LED_Button5_8266.ino
//
//  Example of using a single button switch to set multiple states or conditions
//
//  D0-red LED
//  D1-green LED
//  D2-yellow LED
//  D3-push button
//
//  state holds the current status.
//  0 = all off.
//  1 = green LED
//  2 = yellow LED
//  3 = red LED

// Define the pins being used
int pin_LED1green =   D1;
int pin_LED2yellow =  D2;
int pin_LED0red =     D0;

int pin_Button3 = D3;

// new and old switch states
boolean oldButtonState = HIGH;
boolean newButtonState1 = HIGH;
boolean newButtonState2 = HIGH;
boolean newButtonState3 = HIGH;

byte state = 0;
```

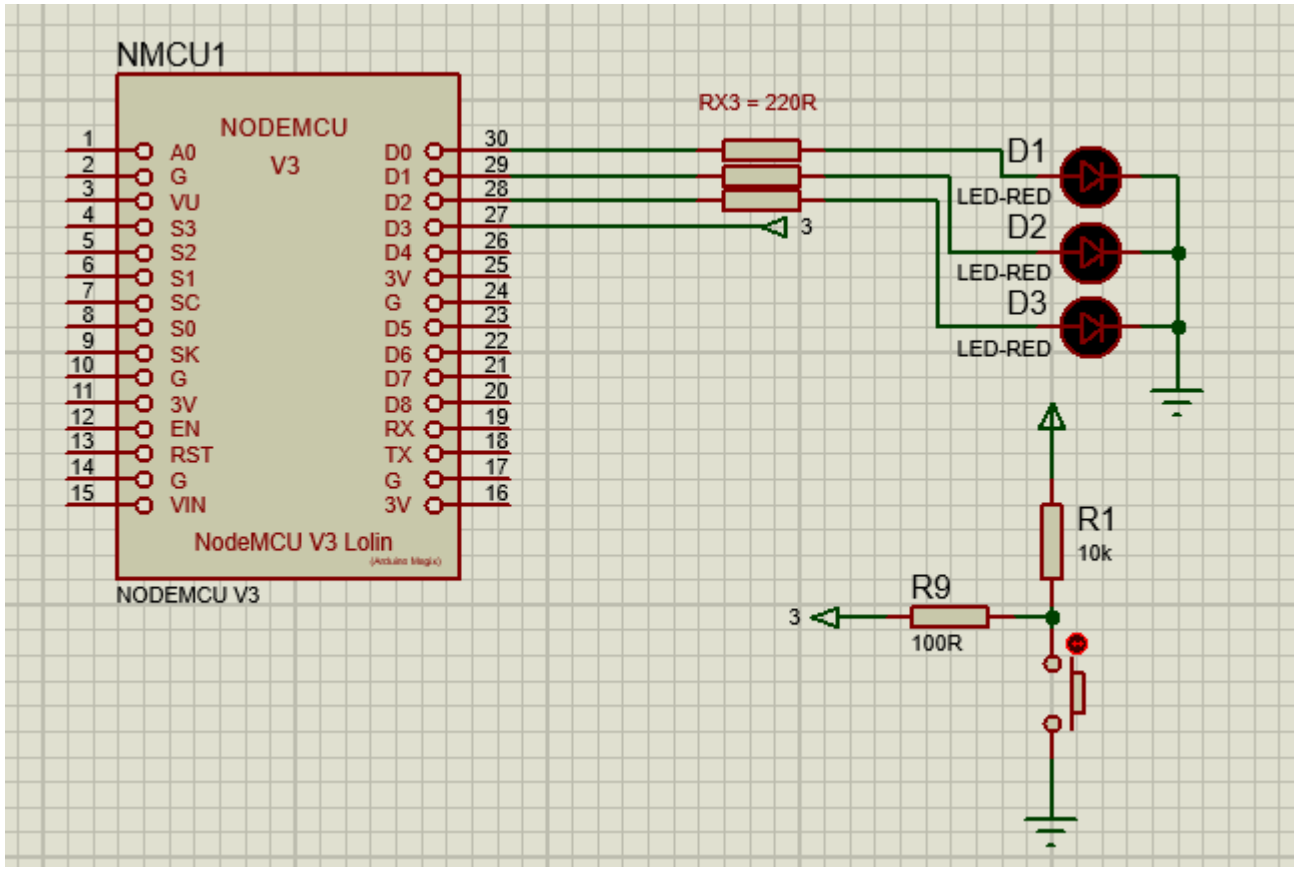
4.5 LED Button 5

```
void setup()
{

    pinMode(pin_LED1green, OUTPUT);    digitalWrite(pin_LED1green,LOW);
    pinMode(pin_LED2yellow, OUTPUT);   digitalWrite(pin_LED2yellow,LOW);
    pinMode(pin_LED0red, OUTPUT);      digitalWrite(pin_LED0red,LOW);
    pinMode(pin_Button3, INPUT);
}

void loop()
{
    newButtonState1 = digitalRead(pin_Button3);
    delay(1);
    newButtonState2 = digitalRead(pin_Button3);
    delay(1);
    newButtonState3 = digitalRead(pin_Button3);

    // if all 3 values are the same we can continue
    if (  (newButtonState1==newButtonState2) && (newButtonState1==newButtonState3) )
    {

        if ( newButtonState1 != oldButtonState )
        {

           // has the button switch been closed?
           if ( newButtonState1 == LOW )
           {
                // increase the value of state
                state++;
                if (state > 3) { state = 0; }
```

4.5 LED Button 5

```
// turn all LEDs off. Doing it this way means we do not need to care about the individual LEDs
                // simply turn them all off and then turn on the correct one.
                digitalWrite(pin_LED1green, LOW);
                digitalWrite(pin_LED2yellow, LOW);
                digitalWrite(pin_LED0red, LOW);

                // Turn on the next LED
                // Because the value of state does not change while we are testing it we don't
need to use else if
                if (state==1) { digitalWrite(pin_LED1green, HIGH);  }
                if (state==2) { digitalWrite(pin_LED2yellow, HIGH);  }
                if (state==3) { digitalWrite(pin_LED0red, HIGH);  }

            }
            oldButtonState = newButtonState1;
        }
    }
}
```

4.6 LED Button 6



4.6 LED Button 6

```
//  Example 4.6 - LED_Button6_8266.ino
//  Example of using a single button switch to set multiple states or conditions
//
//  Pins
//  D1-green LED
//  D2-yellow LED
//  D0-red LED
//  D3-push button
//
//  state holds the current status.
//  0 = all off.
//  1 = green LED
//  2 = yellow LED
//  3 = red LED

// Define the pins being used fro the LEDs green/yellow/red
char LED_Pins_array[] = { 5, 4, 16};

// Array to hold the LED sequence. green, yellow, red, green.
// position 0 is not used (considered not good practice but keeps the code easy to understand)
char LED_Sequence_array[] = { 5, 4, 16, 5};
byte sequenceLength = 4;

int pin_Button3 = 0;

// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;
boolean newSwitchState1 = HIGH;
boolean newSwitchState2 = HIGH;
boolean newSwitchState3 = HIGH;

byte state = -1;
```

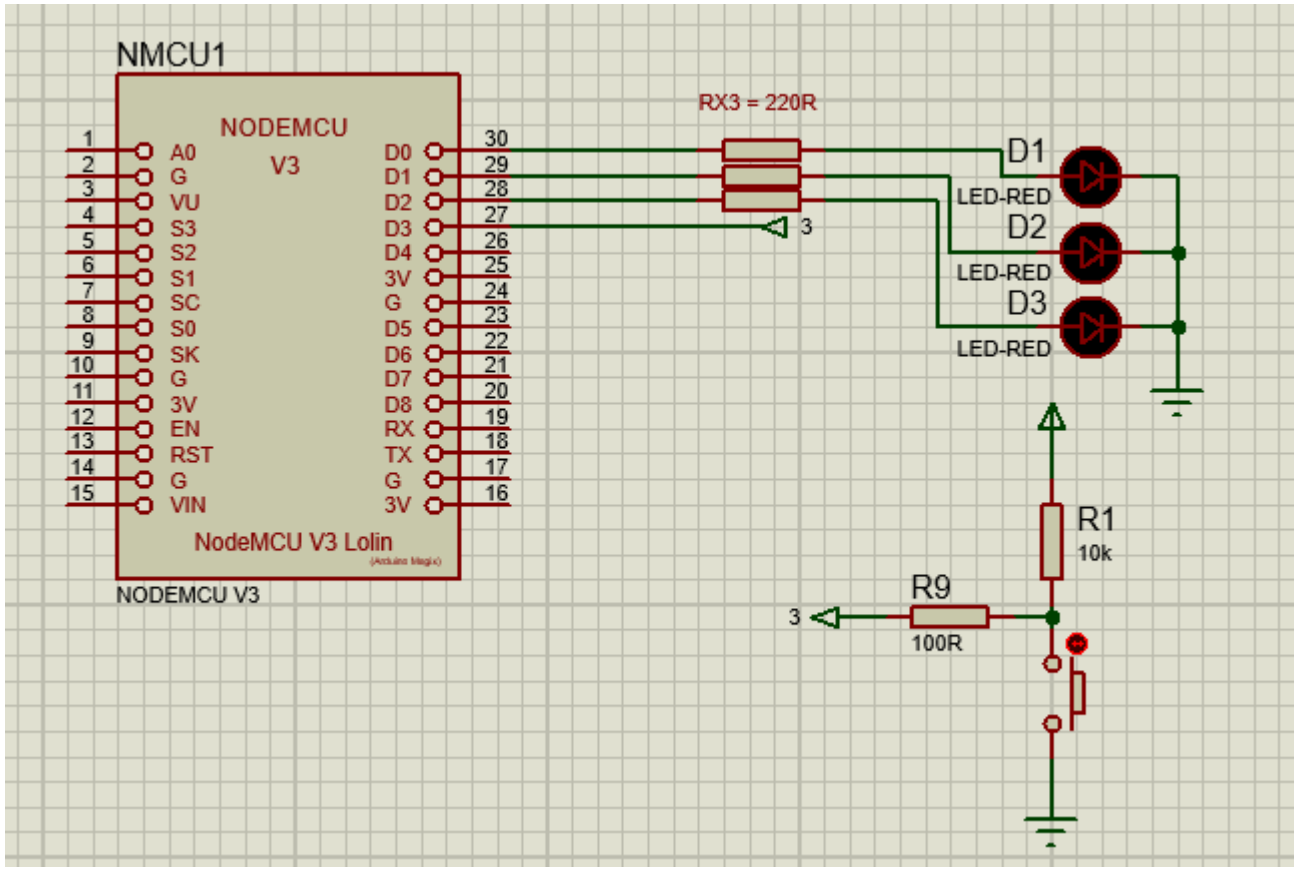
4.6 LED Button 6

```
void setup()
{

    for (byte i=0; i< 3; i++)
    {
         pinMode(LED_Pins_array[i], OUTPUT);
         digitalWrite(LED_Pins_array[i],LOW);
    }
         pinMode(pin_Button3, INPUT);
}
```

4.6 LED Button 6

```
void loop()
{
    newSwitchState1 = digitalRead(pin_Button3);
    delay(1);
    newSwitchState2 = digitalRead(pin_Button3);
    delay(1);
    newSwitchState3 = digitalRead(pin_Button3);

    // if all 3 values are the same we can continue
    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {

           // has the button switch been closed?
           if ( newSwitchState1 == LOW )
           {
                state++;
                if (state > (sequenceLength -1) ) { state = 0; }

                // turn all LEDs off. Doing it this way means we do not need to care about the
individual LEDs
                // simply turn them all off and then turn on the correct one.
                for (byte i=0; i< 3; i++)
                {
                     digitalWrite(LED_Pins_array[i],LOW);
                }

                // Turn on the next LED
                digitalWrite(LED_Sequence_array[state],HIGH);
           }
           oldSwitchState = newSwitchState1;
        }
    }
}
```

4.7 LED Button 7



4.7 LED Button 7

```
//  Example 4.7 - LED_Button7_8266.ino
//
//  Example of using a button switch as a toggle switch to turn a blinking LED on or off
//
//  Pins
//  D0 - LED
//  D3 - Push Button
//

// Define the pins being used
int pin_LED0 = D0;
int pin_Button3 = D3;


// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;
boolean newSwitchState1 = HIGH;
boolean newSwitchState2 = HIGH;
boolean newSwitchState3 = HIGH;

// variables to hold the times
unsigned long timeNow = 0;
unsigned long timePrev = 0;
unsigned int timeWait = 100;

// variables used to control the LED
boolean flashingLEDisON = false;
boolean LEDstatus = LOW;
boolean keyPressed = false;
```

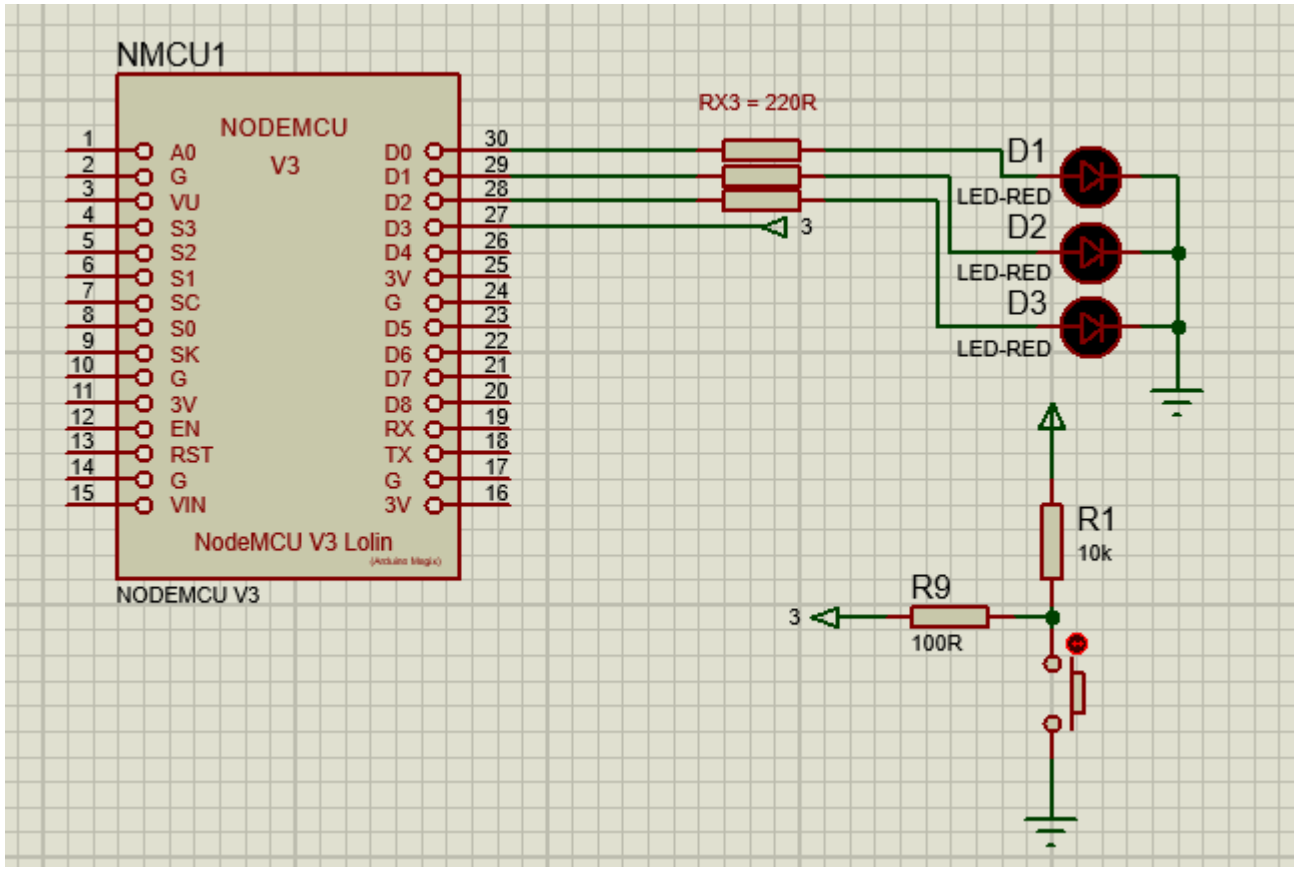
4.7 LED Button 7

```
void setup()
{


    pinMode(pin_LED0, OUTPUT);
    digitalWrite(pin_LED0,LOW);
    pinMode(pin_Button3, INPUT);
}

void loop()
{
    newSwitchState1 = digitalRead(pin_Button3);      delay(1);
    newSwitchState2 = digitalRead(pin_Button3);      delay(1);
    newSwitchState3 = digitalRead(pin_Button3);

    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {
            if ( newSwitchState1 == LOW ) { keyPressed = true; } else { keyPressed =
false; }
            oldSwitchState = newSwitchState1;
        }
    }
```

4.7 LED Button 7

```
if ( keyPressed )
    {
         // turn on or turn off the blinking LED
         if ( flashingLEDisON == false)
         {
             flashingLEDisON = true;
         }
         else
         {
              flashingLEDisON = false;
              // the LED may be on so to be safe we turn it off. If you wished you could check
LEDstatus
              LEDstatus = LOW;  digitalWrite(pin_LED0, LEDstatus);
         }
         keyPressed = false;
    }


    // if the blinking LED is on. See if it is time to blink it
    if ( flashingLEDisON == true )
    {
        timeNow = millis();
        if (timeNow-timePrev >= timeWait )
        {
              timePrev = timeNow;
              if (LEDstatus == LOW) { LEDstatus = HIGH; } else { LEDstatus = LOW; }
              digitalWrite(pin_LED0, LEDstatus);
        }
    }
}
```

4.8 LED Button 8



4.8 Button LED 8

```
//  Example 4.8 - LED_Button8_8266.ino
//
//  Example of using a button switch as a toggle switch to turn a blinking LED on or off
//  now using functions
//
//  Pins
//  D0-LED
//  D3-push button
//

// Define the pins being used
int pin_LED0 = D0;
int pin_Button3 = D3;


// variables to hold the new and old switch states
boolean oldSwitchState = HIGH;

// variables to hold the times
unsigned long timeNow = 0;
unsigned long timePrev = 0;
unsigned int timeWait = 100;

// variables used to control the LED
boolean flashingLEDisON = false;
boolean LEDstatus = LOW;
boolean keyPressed = false;
```

4.8 Button LED 8

```
void setup()
{


    pinMode(pin_LED0, OUTPUT);
    digitalWrite(pin_LED0,LOW);

    pinMode(pin_Button3, INPUT);
}

void loop()
{
    keyPressed = checkButtonSwitch();
    if ( keyPressed )
    {
        keyPressed = false;
        startAndStop();
    }
    if ( flashingLEDisON == true )  { blinkTheLED();   }
}
```

4.8 Button LED 8

```
boolean checkButtonSwitch()
{
    boolean key = false;

    boolean newSwitchState1 = digitalRead(pin_Button3);      delay(1);
    boolean newSwitchState2 = digitalRead(pin_Button3);      delay(1);
    boolean newSwitchState3 = digitalRead(pin_Button3);

    if (  (newSwitchState1==newSwitchState2) && (newSwitchState1==newSwitchState3) )
    {
        if ( newSwitchState1 != oldSwitchState )
        {
            if ( newSwitchState1 == LOW ) { key = true; } else { key =  false; }
            oldSwitchState = newSwitchState1;
        }
    }
    return key;
}
```

4.8 Button LED 8

```
void startAndStop( )
{
     // turn on or turn off the blinking LED
     if ( flashingLEDisON == false)
     {
         flashingLEDisON = true;
     }
     else
     {
         flashingLEDisON = false;
         // the LED may be on so we turn it off just is case
         LEDstatus = LOW;
         digitalWrite(pin_LED0, LEDstatus);
     }
}


void blinkTheLED()
{
    timeNow = millis();
    if (timeNow-timePrev >= timeWait )
    {
          timePrev = timeNow;
          if (LEDstatus == LOW) { LEDstatus = HIGH; } else { LEDstatus = LOW; }
          digitalWrite(pin_LED0, LEDstatus);
    }
}
```


THANK
You!