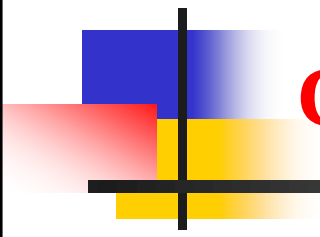


Chapter 7

ARDUINO PROGRAMMING

Introduction

- The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with single identifiers and the capability to automatically transfer and the capability to automatically transfer data more to a network without requiring human-to-human or human-to-computer communication
- **Arduino Board:**
  - An Arduino is actually a microcontroller based kit.
  - It is basically used in communications and in controlling or operating many devices.
  - Arduino UNO board is the most popular board in the Arduino board family.
  - In addition, it is the best board to get started with electronics and coding.
  - Some boards look a bit different from the one given below, but most Arduino's have majority of these components in common.
  - It consists of two memories- Program memory and the data memory.
  - The code is stored in the flash program memory, whereas the data is stored in the data memory.
  - Arduino Uno consists of 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button

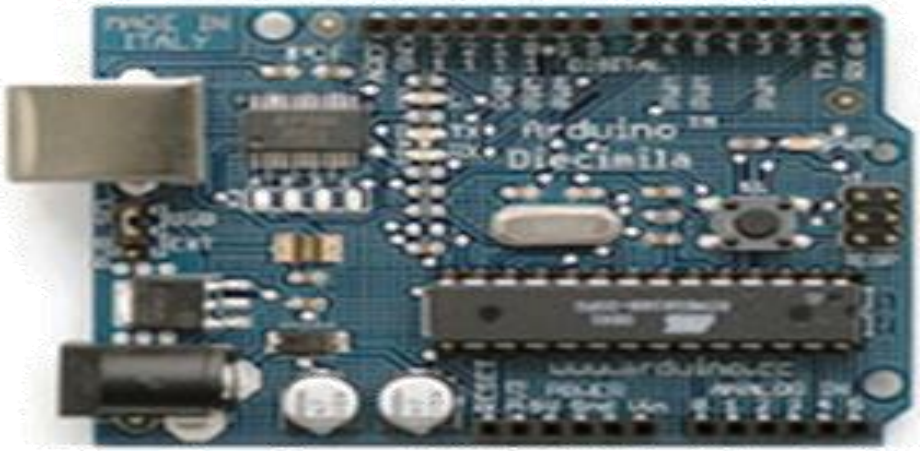
WHAT IS ARDUINO?

Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer.

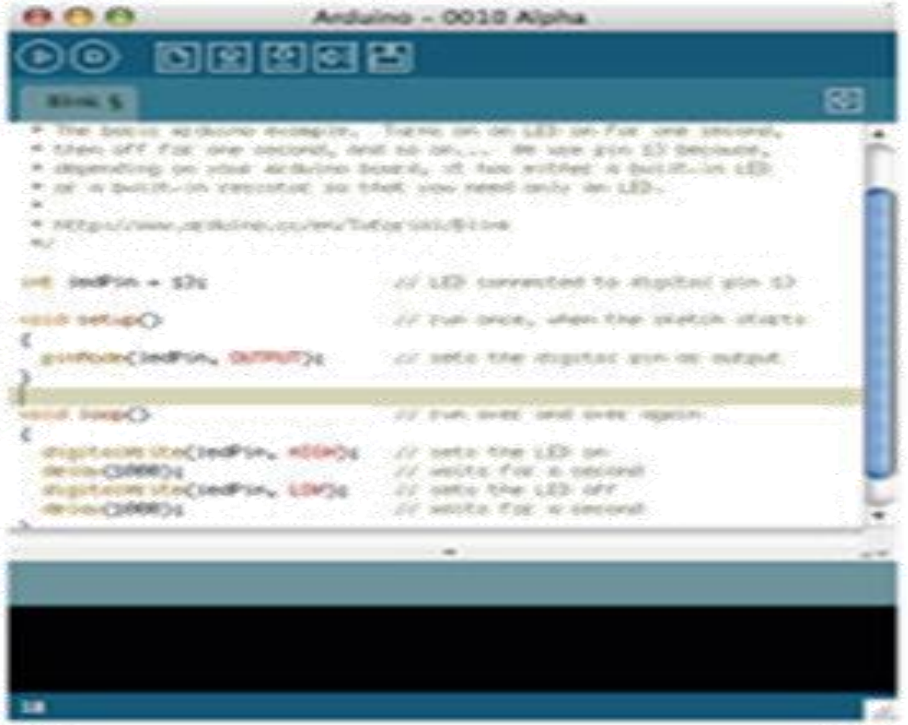
It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board

The word "Arduino" can mean 2 things

A physical piece of hardware



A programming environment



PROJECT HUB™
FOR INNOVATIVE BHARAT

WHY ARDUINO?

Inexpensive

Arduino boards are relatively inexpensive compared to other microcontroller platforms

Cross-platform

The Arduino software runs on Windows, and Linux operating systems.

Open source and extensible software

The Arduino software is published as open source tools, available for extension by experienced programmers

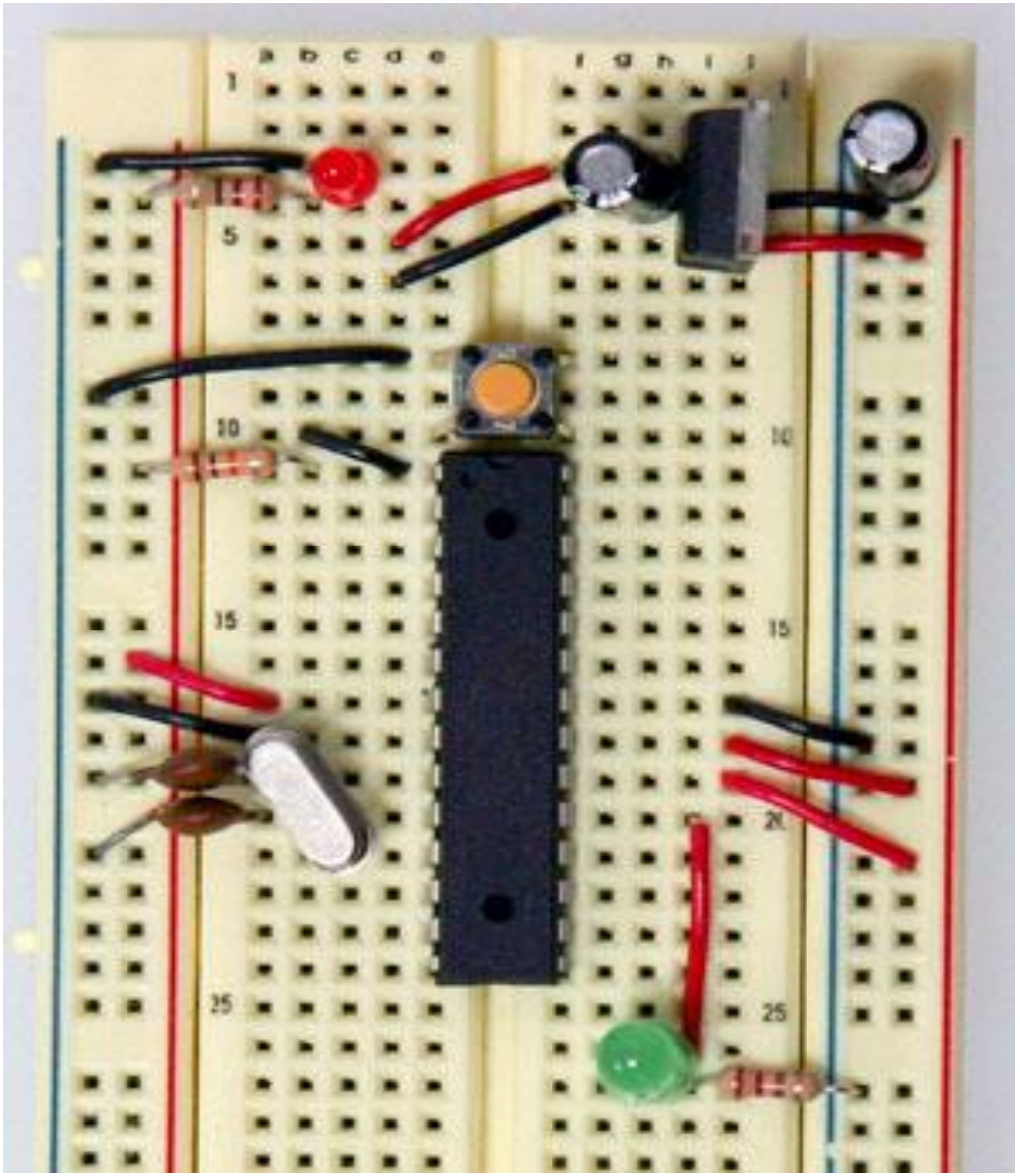
Open source and extensible hardware

Simple, clear programming environment

PROJECT HUB™
FOR INNOVATIVE BHARAT

COMPONENTS OF THE ARDUINO

- ATMega168/328
- 16MHz crystal/filtering capacitors
- Onboard power regulators
- FTDI USB <-> Serial Chip
- Hardware



PROJECT HUB
FOR INNOVATIVE BHARAT

PROJECT HUB™
FOR INNOVATIVE BHARAT

NECESSARY PARTS FOR ARDUINO

MICROCONTROLLER UNIT

- ATMega168/328
- The ‘brains’ of the Arduino
- Program is loaded onto the chip
- Runs main loop until power is removed

PROJECT HUB
FOR INNOVATIVE BHARAT

16MHZ CRYSTAL

- 16Mhz Crystal
- The ‘heartbeat’ of the ATMega chip
- Speed of crystal determines chip speed
- Possible to over/underclock depending on application
- ATMega series has onboard oscillator; less precise

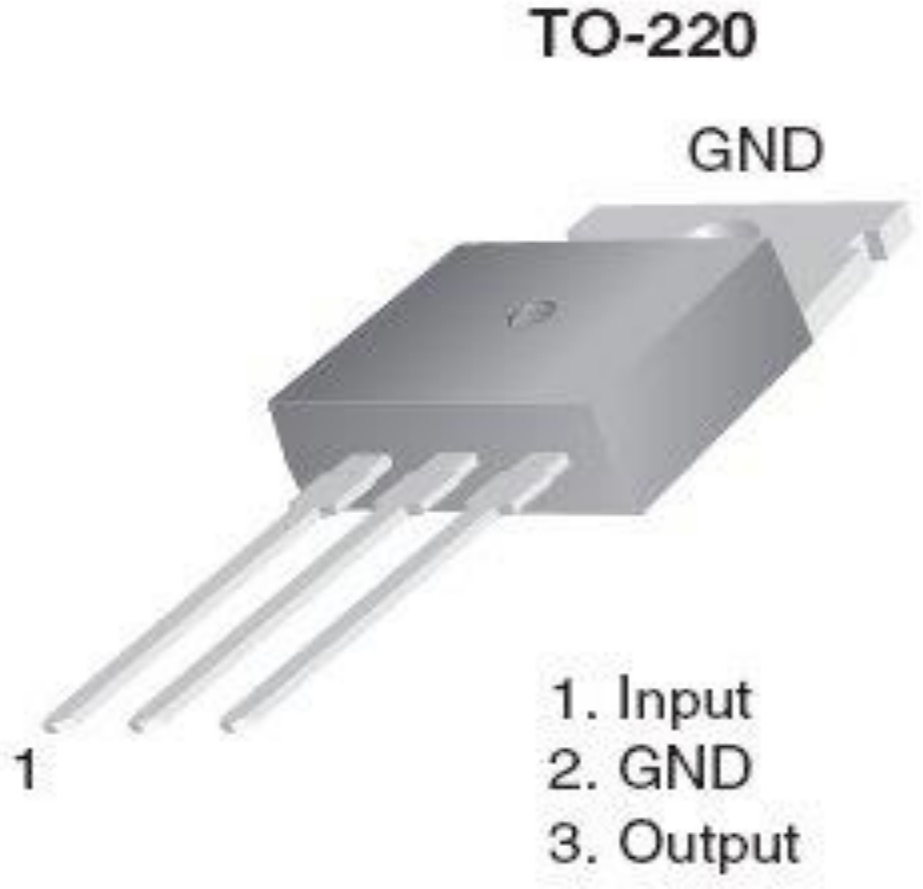
PROJECT HUB™
FOR INNOVATIVE BHARAT

FTDI USB CHIP

- Allows your Arduino to communicate with your computer over a simple USB link
- Only necessary for communicating with USB

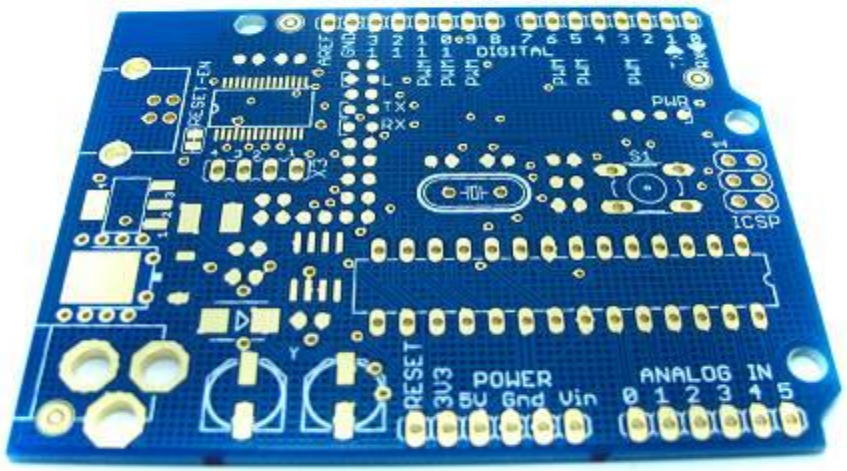
POWER SUPPLY

- 5 Volt and 3.3 Volt Regulators
- Filtering capacitors
- Automatic switching between external and USB Power
- Leave it out if you have a filtered 5 Volt power supply



HARDWARE

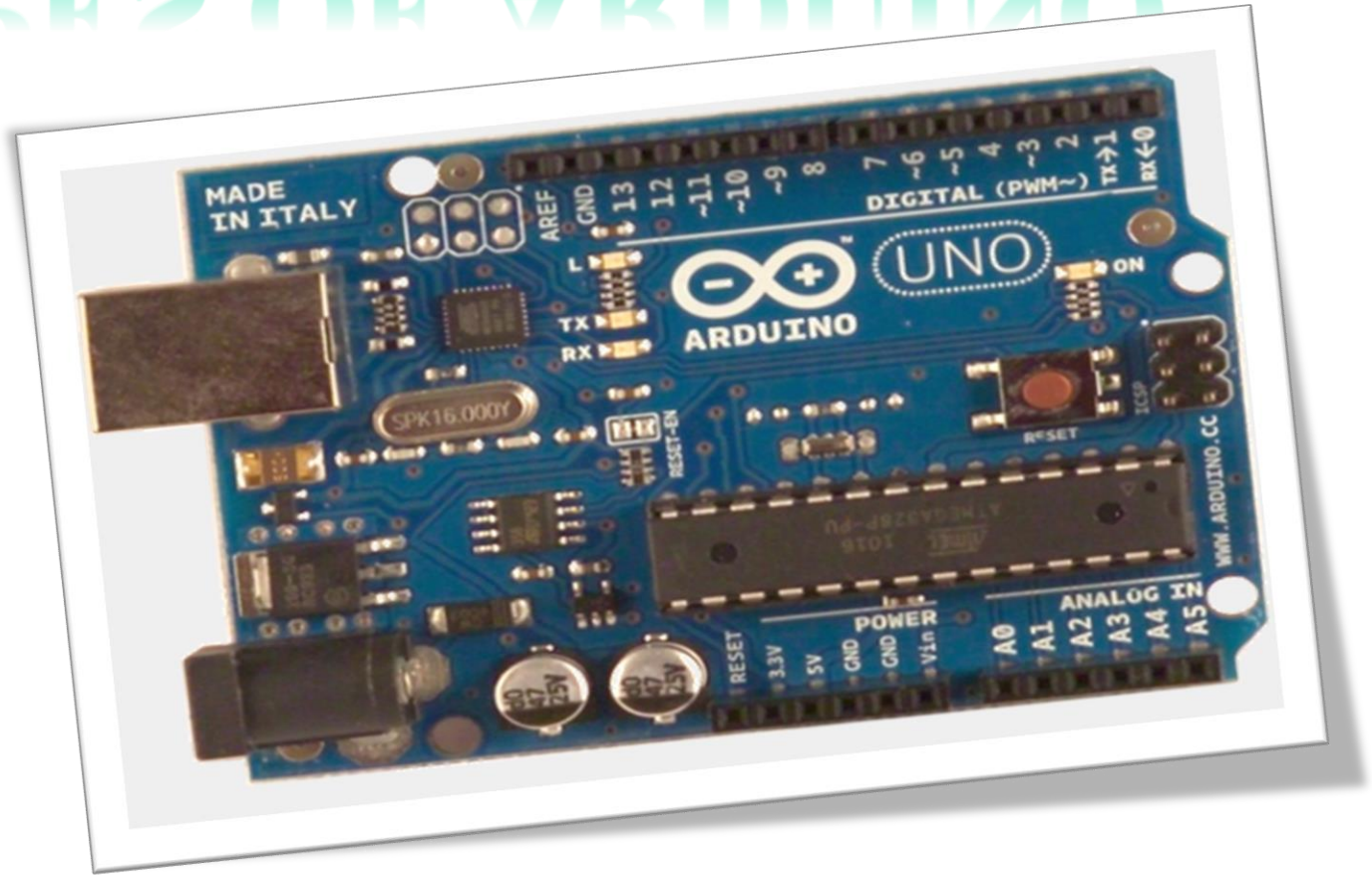
- Circuit Board
- Headers
- USB port
- Sockets



PROJECT HUB™
FOR INNOVATIVE BHARAT

PROJECT HUB
FOR INNOVATIVE BHARAT

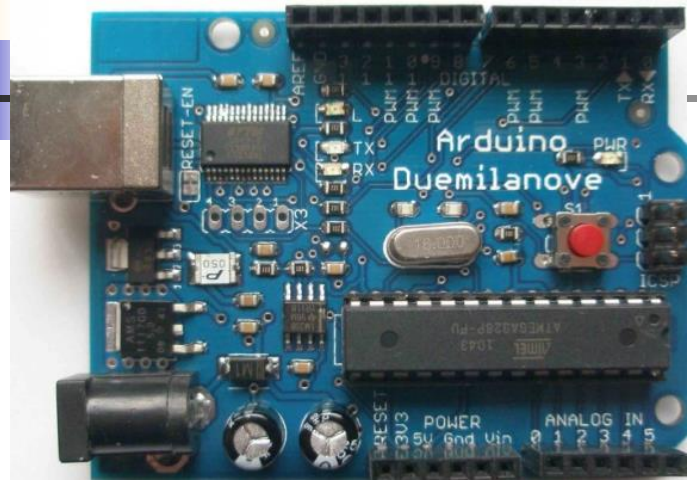
TYPES OF ARDUINO

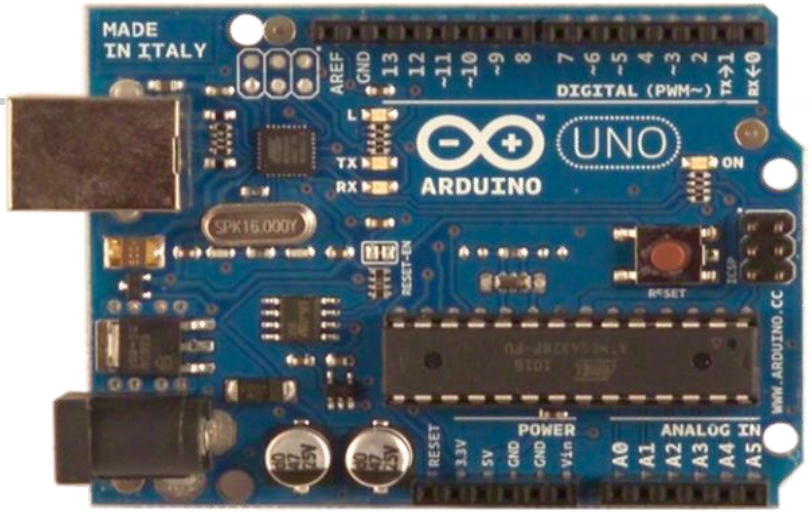


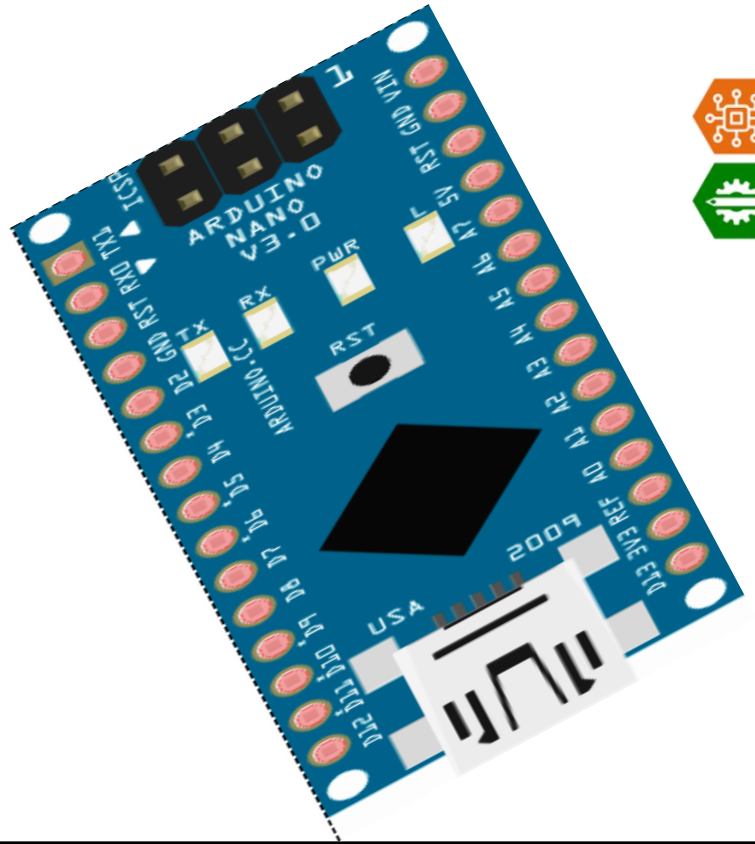
TYPES OF ARDUINO

There are many different Arduino variations on the market, from small boards like the Arduino mini to large boards like the Arduino MEGA. All have certain features in common:

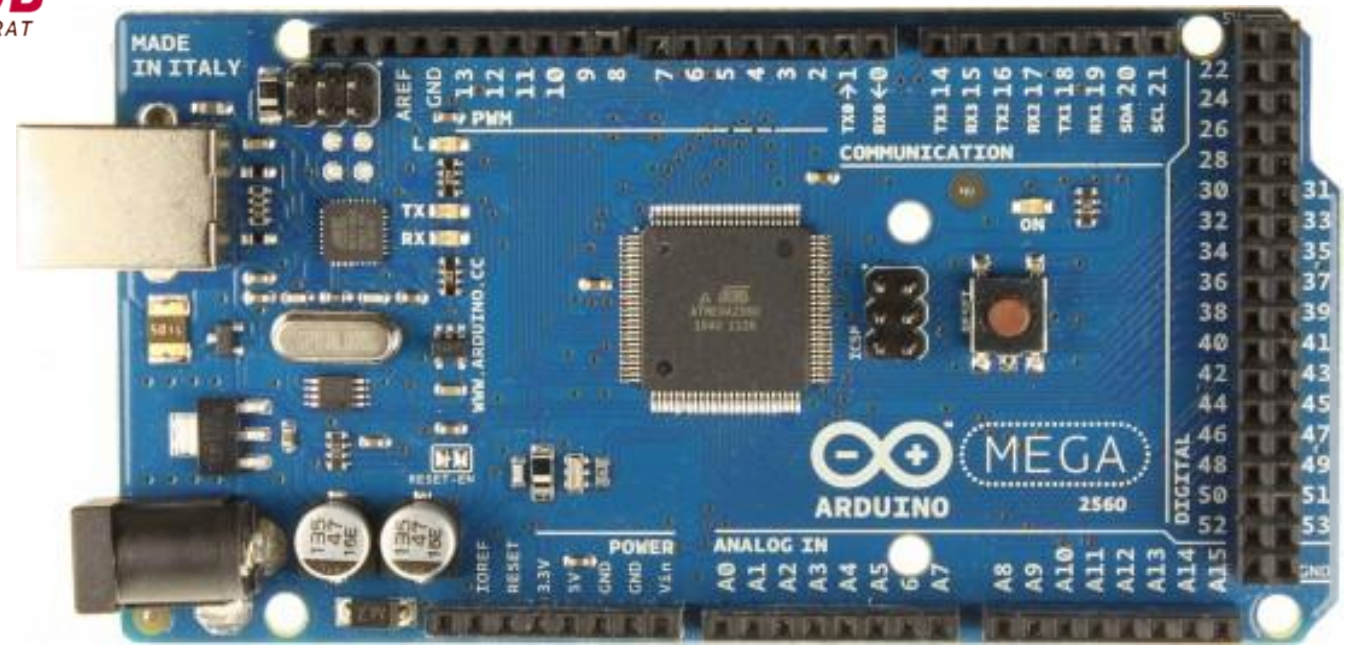
- Digital input/output pins (some double as PWM pins)
- Analog input/output pins
- Serial communication pins
- In-system programming pins (ISP)
- Compatibility with Arduino software



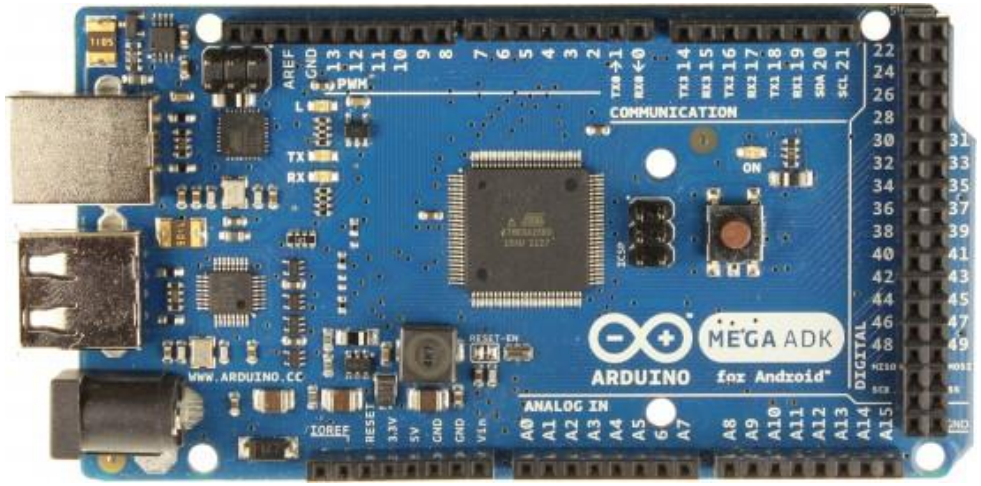


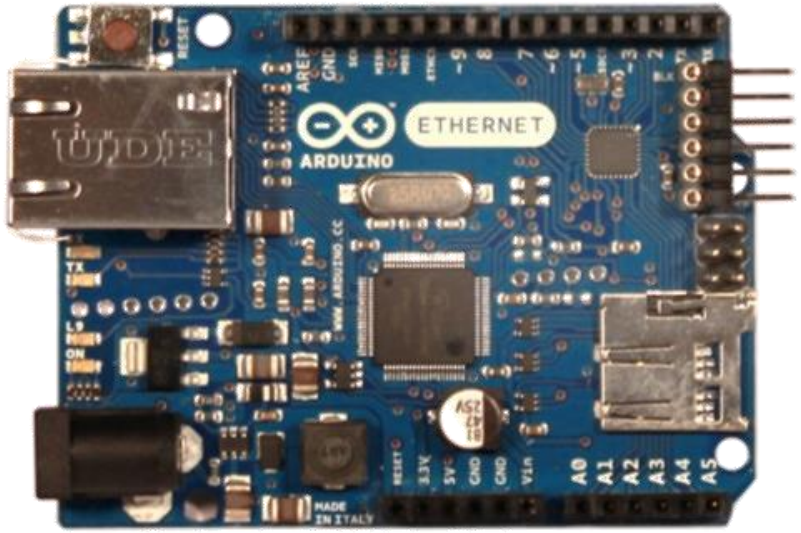


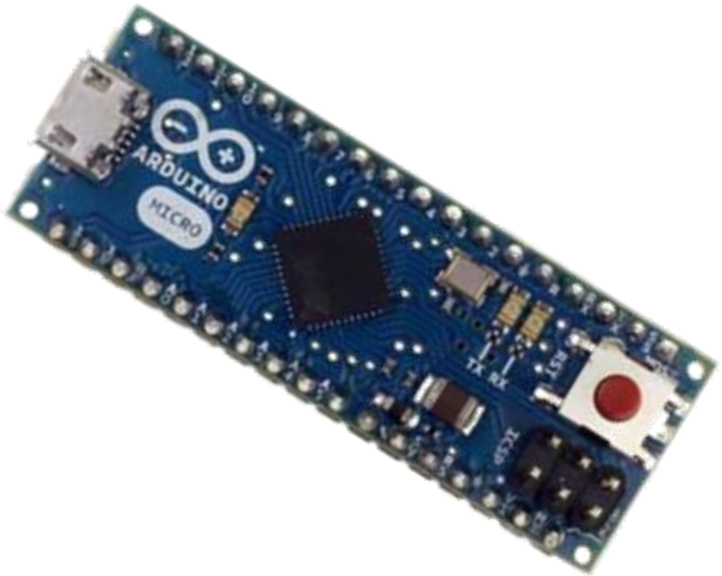
PROJECT HUB™
FOR INNOVATIVE BHARAT



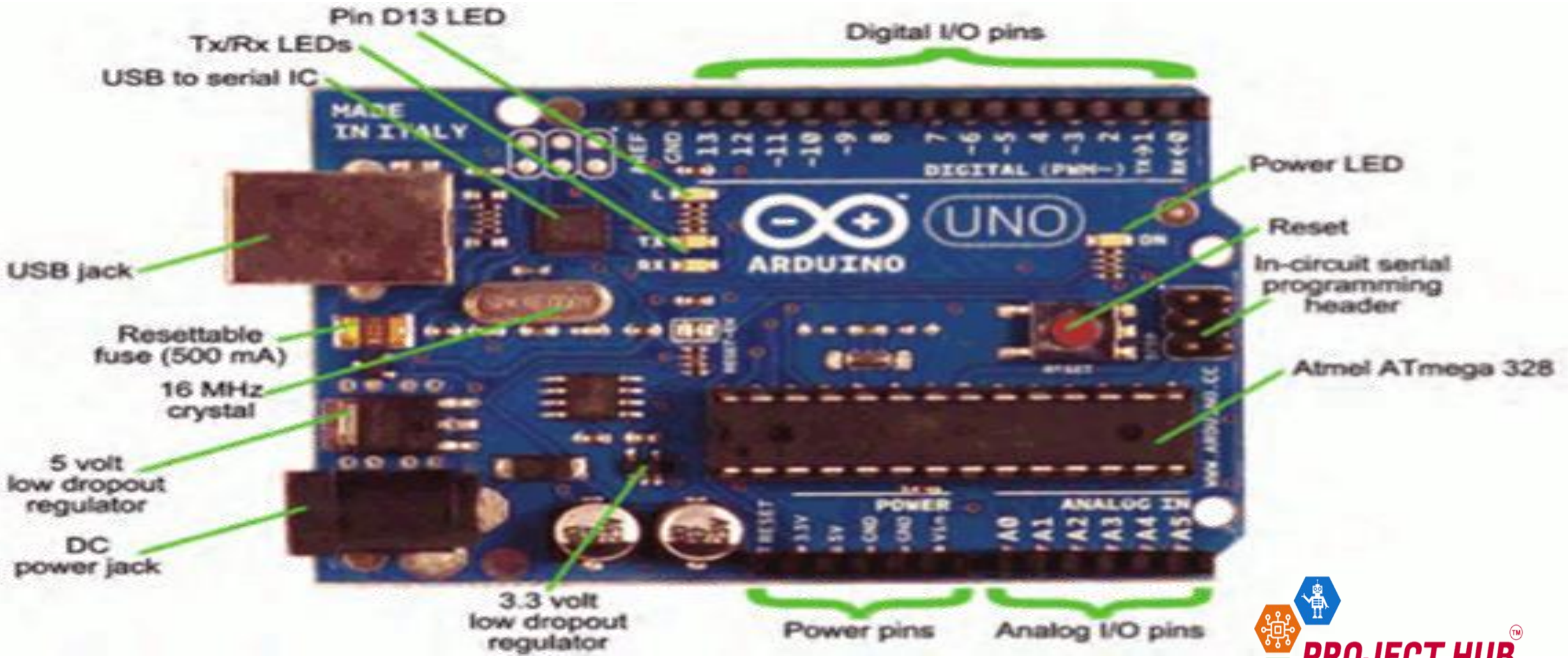
PROJECT HUB
FOR INNOVATIVE BHARAT





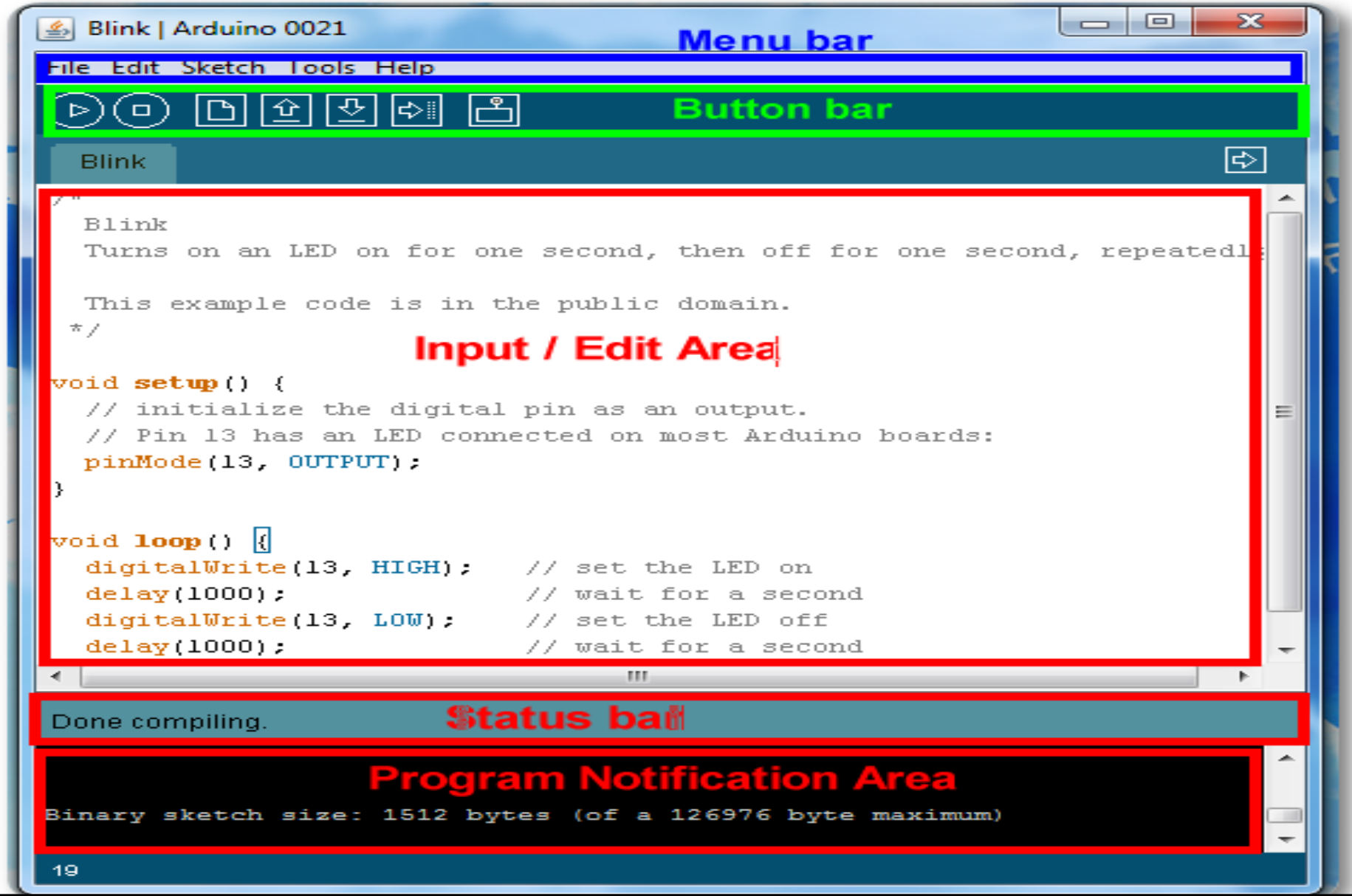


ARDUINO DEVELOPMENT BOARD



PROJECT HUB
FOR INNOVATIVE BHARAT

ARDUINO SOFTWARE



PROJECT HUB
FOR INNOVATIVE BHARAT

ARDUINO LANGUAGE

C like syntax, but simplified

Abstracts the pin naming to numbers

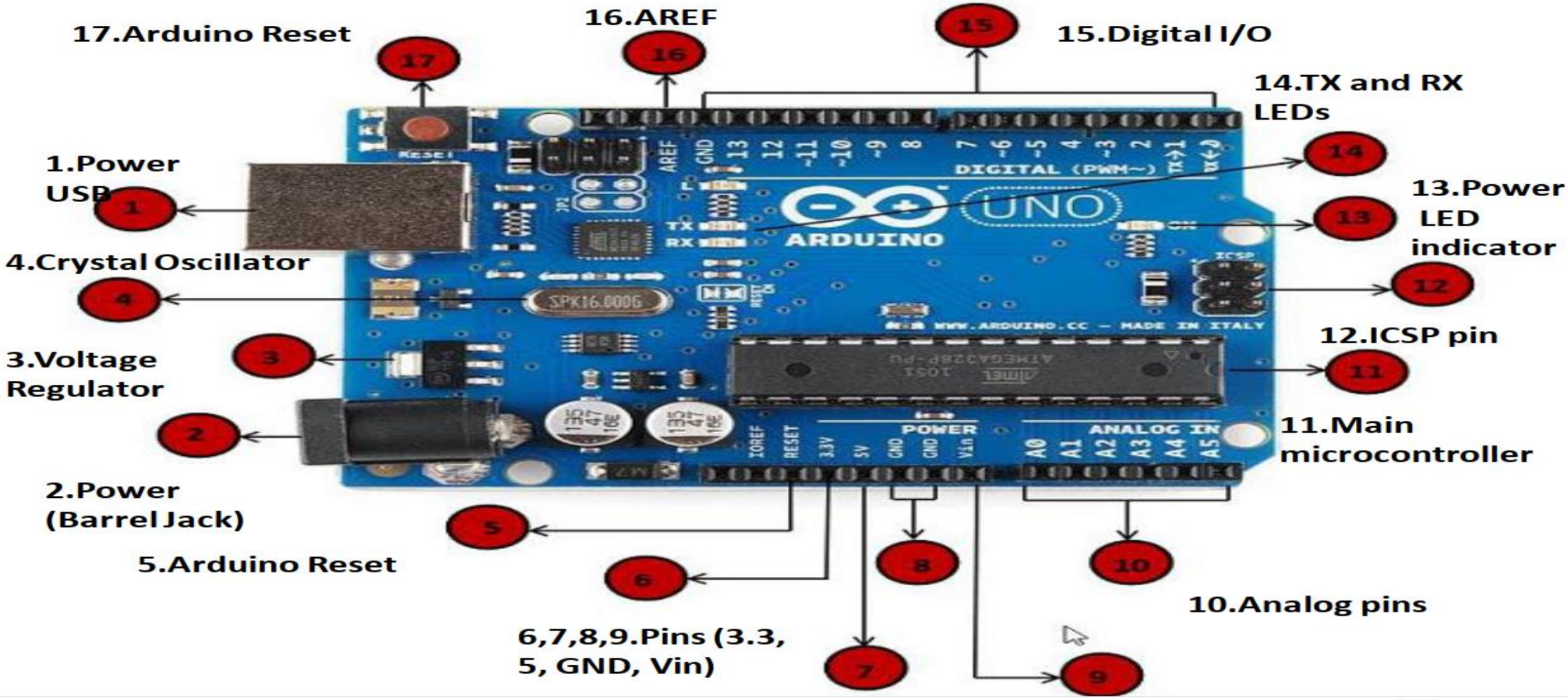
Easy to learn, yet powerful

Lots of example code

Easy to reuse C-code from other projects

Libraries can be written in C++

Lots of libraries available


17.Arduino Reset
16.AREF
15
15.Digital I/O
14.TX and RX LEDs
1.Power USB
13.Power LED indicator
4.Crystal Oscillator
12.ICSP pin
3.Voltage Regulator
11.Main microcontroller
2.Power (Barrel Jack)
5.Arduino Reset
10.Analog pins
6,7,8,9.Pins (3.3, 5, GND, Vin)
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
WWW.ARDUINO.CC – MADE IN ITALY


Transistor
SOT-23
Resistor Array
0805 LED
VIA
5V-Regulator
SOT-223
0603 Resistor
3V-Regulator
SOT-753
SMB Diode
0603 Capacitor
ARDUINO
UNO

- **1.Power USB**
  Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).
  2.Power (Barrel Jack)
  Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).
  3.Voltage Regulator
  The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.
  4.Crystal Oscillator
  The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.
  5,17.Arduino Reset
  You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

- **6,7,8,9.Pins (3.3, 5, GND, Vin)**
  • 3.3V (6) - Supply 3.3 output volt
  • 5V (7) - Supply 5 output volt
  • Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
  • GND (8)(Ground) - There are several GND pins on the Arduino, any of which can be used to ground your circuit.
  • Vin (9) - This pin also can be used to power the Arduino board from an external power source, like AC mains power supply

  10.Analog pins

  The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor

- **11.Main microcontroller**
  Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet.
- **12.ICSP pin**
  Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

- **13.Power LED indicator**

  This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection.

  14.TX and RX LEDs

  On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

- **15.Digital I/O**

  • The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM.

  16.AREF

  • AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Program an Arduino

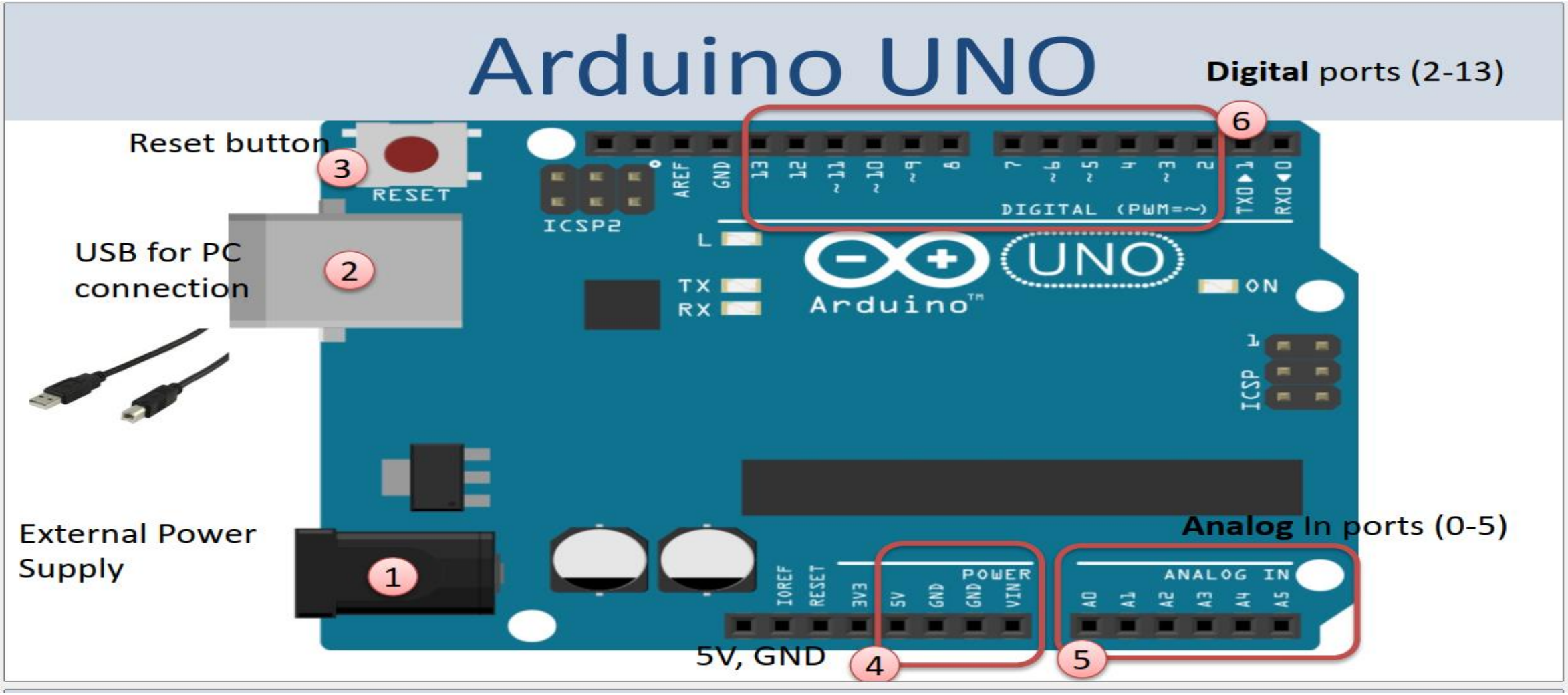
- ➢ The most important advantage with Arduino is the programs can be directly loaded to the device without requiring any hardware programmer to burn the program.
- ➢ This is done because of the presence of the 0.5KB of Bootloader which allows the program to be burned into the circuit.
- ➢ All we have to do is to download the Arduino software and writing the code.
- ➢ The Arduino tool window consists of the toolbar with the buttons like verify, upload, new, open, save, serial monitor.
- ➢ It also consists of a text editor to write the code, a message area which displays the feedback like showing the errors, the text console which displays the output and a series of menus like the File, Edit, Tools menu.

Steps to program an Arduino

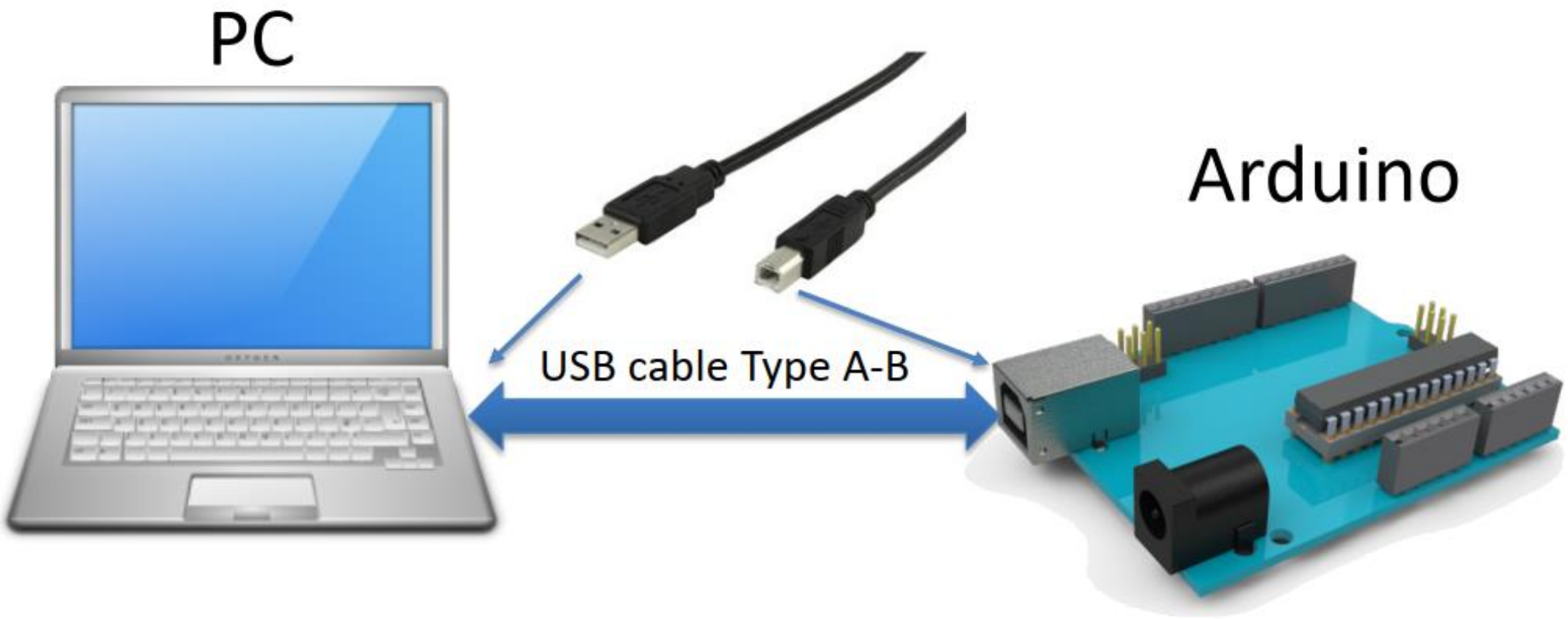
- ➢ Programs written in Arduino are known as sketches. A basic sketch consists of 3 parts
  1. Declaration of Variables
  2. Initialization: It is written in the setup () function.
  3. Control code: It is written in the loop () function.
- ➢ The sketch is saved with .ino extension. Any operations like verifying, opening a sketch, saving a sketch can be done using the buttons on the toolbar or using the tool menu.
- ➢ The sketch should be stored in the sketchbook directory.
- ➢ Chose the proper board from the tools menu and the serial port numbers.
- ➢ Click on the upload button or chose upload from the tools menu. Thus the code is uploaded by the bootloader onto the microcontroller

Basic Adruino functions are:

- **digitalRead**(pin): Reads the digital value at the given pin.
- **digitalWrite**(pin, value): Writes the digital value to the given pin.
- **pinMode**(pin, mode): Sets the pin to input or output mode.
- **analogRead**(pin): Reads and returns the value.
- **analogWrite**(pin, value): Writes the value to that pin
- **serial.begin**(baud rate): Sets the beginning of serial communication by setting the bit rate.


Arduino UNO
Digital ports (2-13)
Reset button
3
RESET
USB for PC connection
2
ICSP2
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TXD 1
RXD 0
DIGITAL (PWM=~)
6
L
TX
RX
UNO
Arduino™
ON
ICSP
External Power Supply
1
Analog In ports (0-5)
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
5V, GND
4
5

Connect Arduino to your PC



Arduino Software

Upload Code to Arduino Board

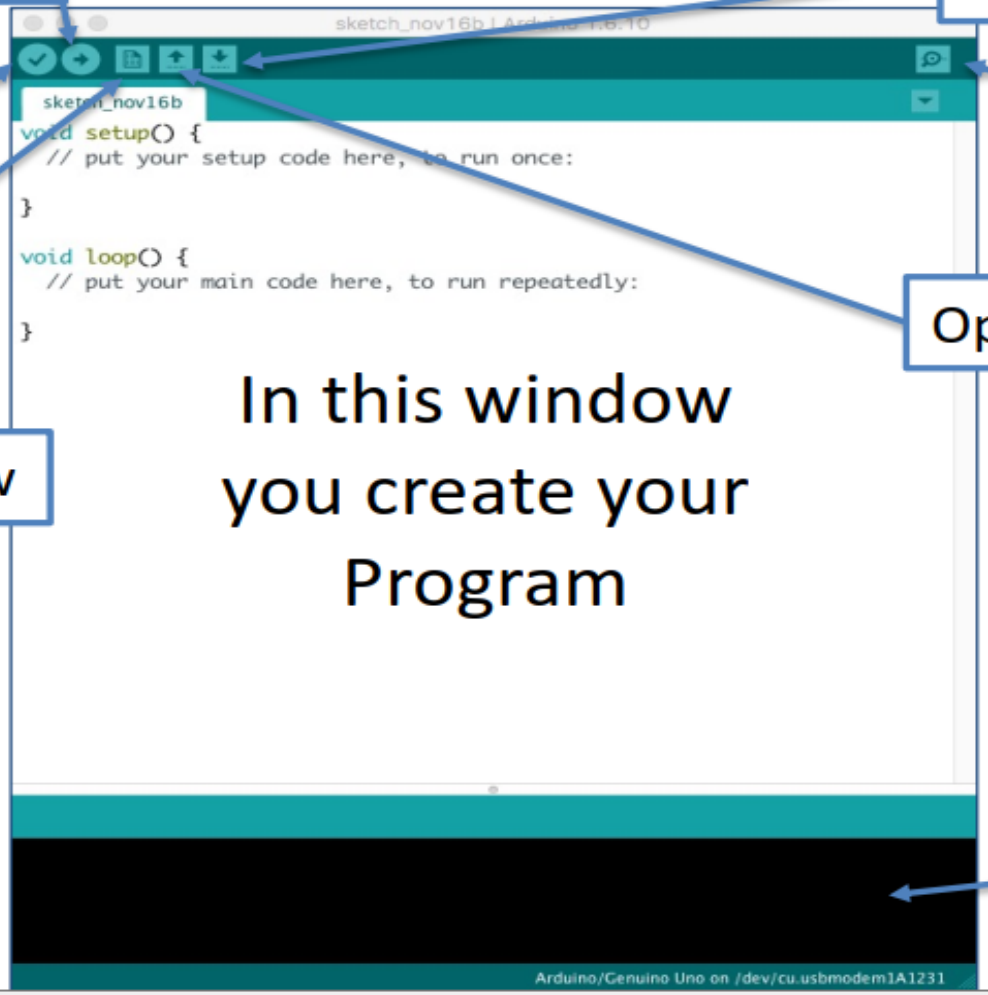
Save

Open Serial Monitor

Compile and Check if Code is OK

Open existing Code

Creates a New Code Window



In this window you create your Program

The software can be downloaded for free:

www.arduino.cc

Error Messages can be seen here

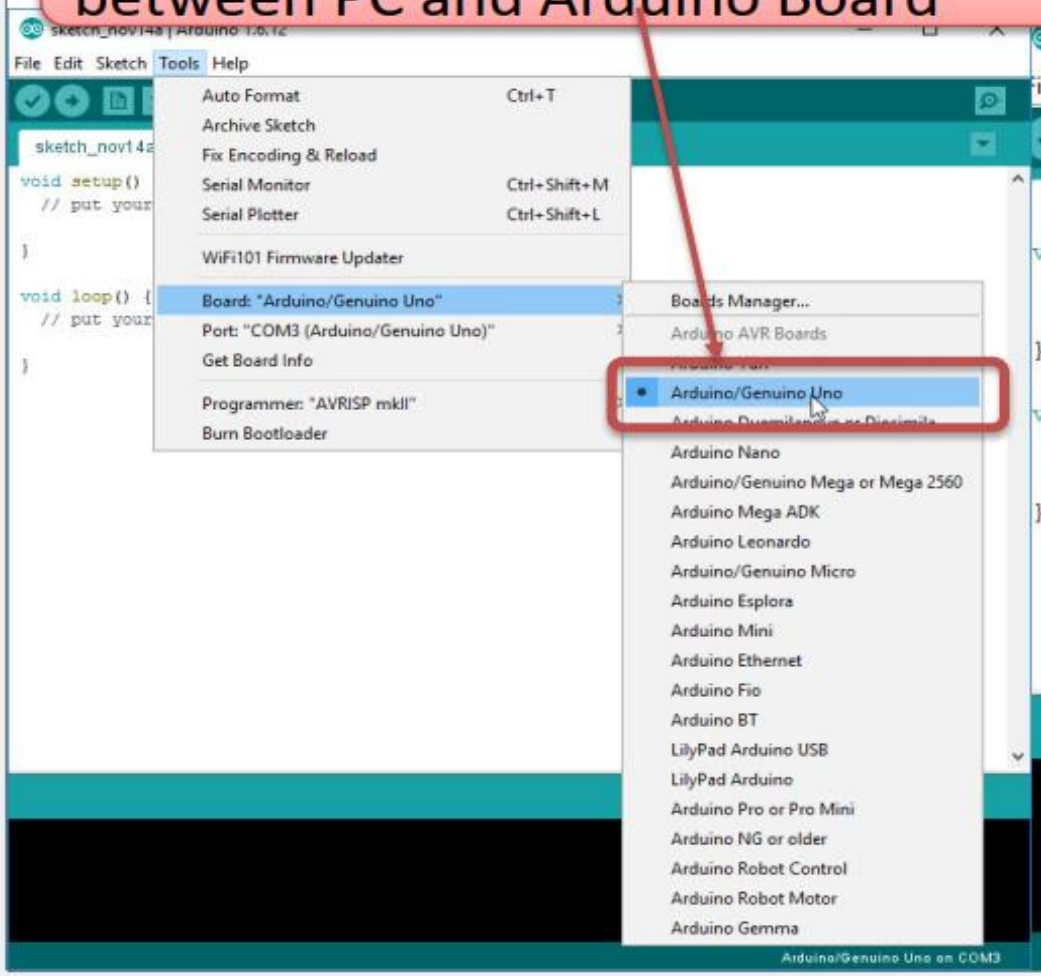
Arduino Programs

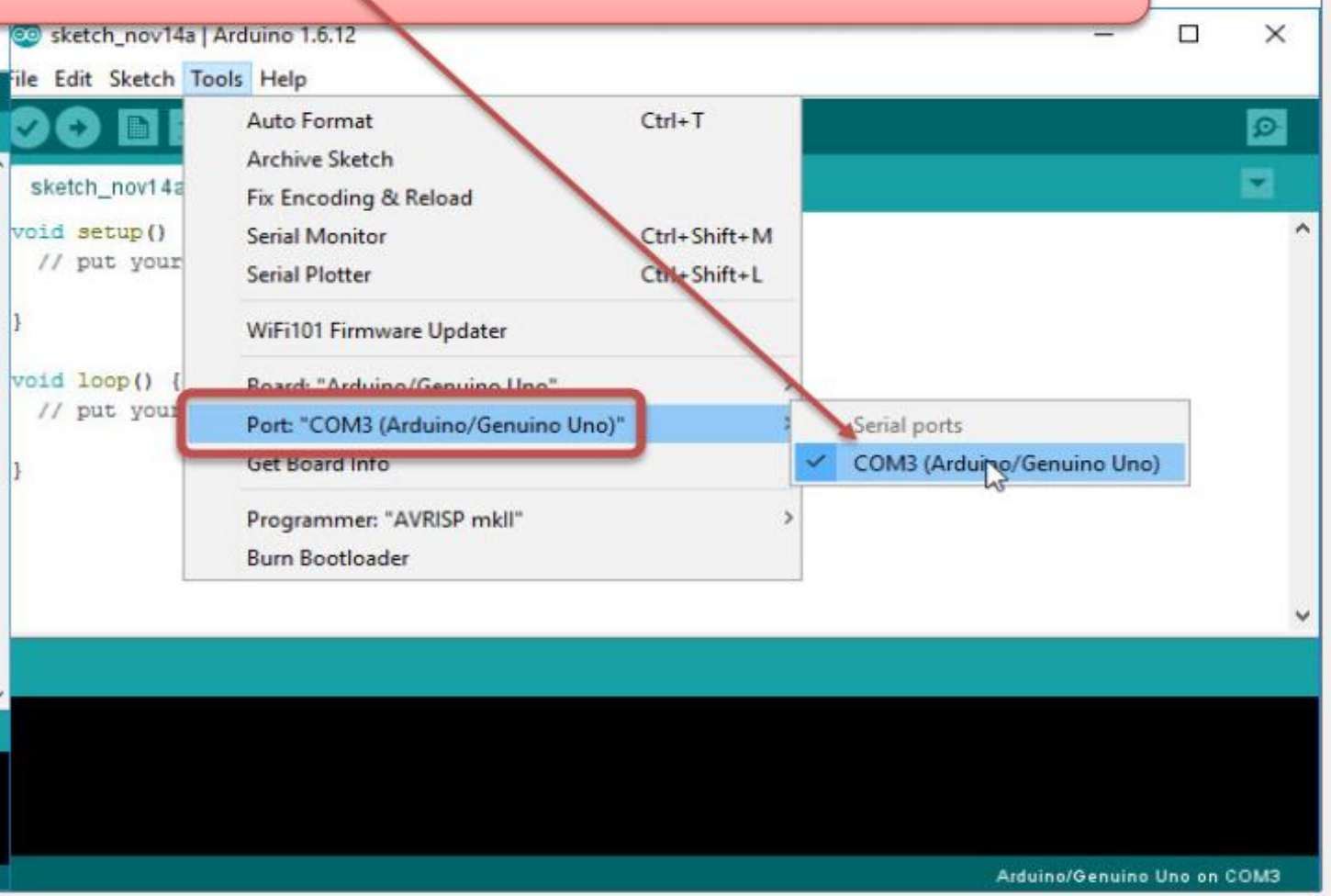
All Arduino programs must follow the following main structure:

```
// Initialization, define variables, etc.

void setup()
{
      // Initialization
      ...
}

void loop()
{
      //Main Program
      ...
}
```

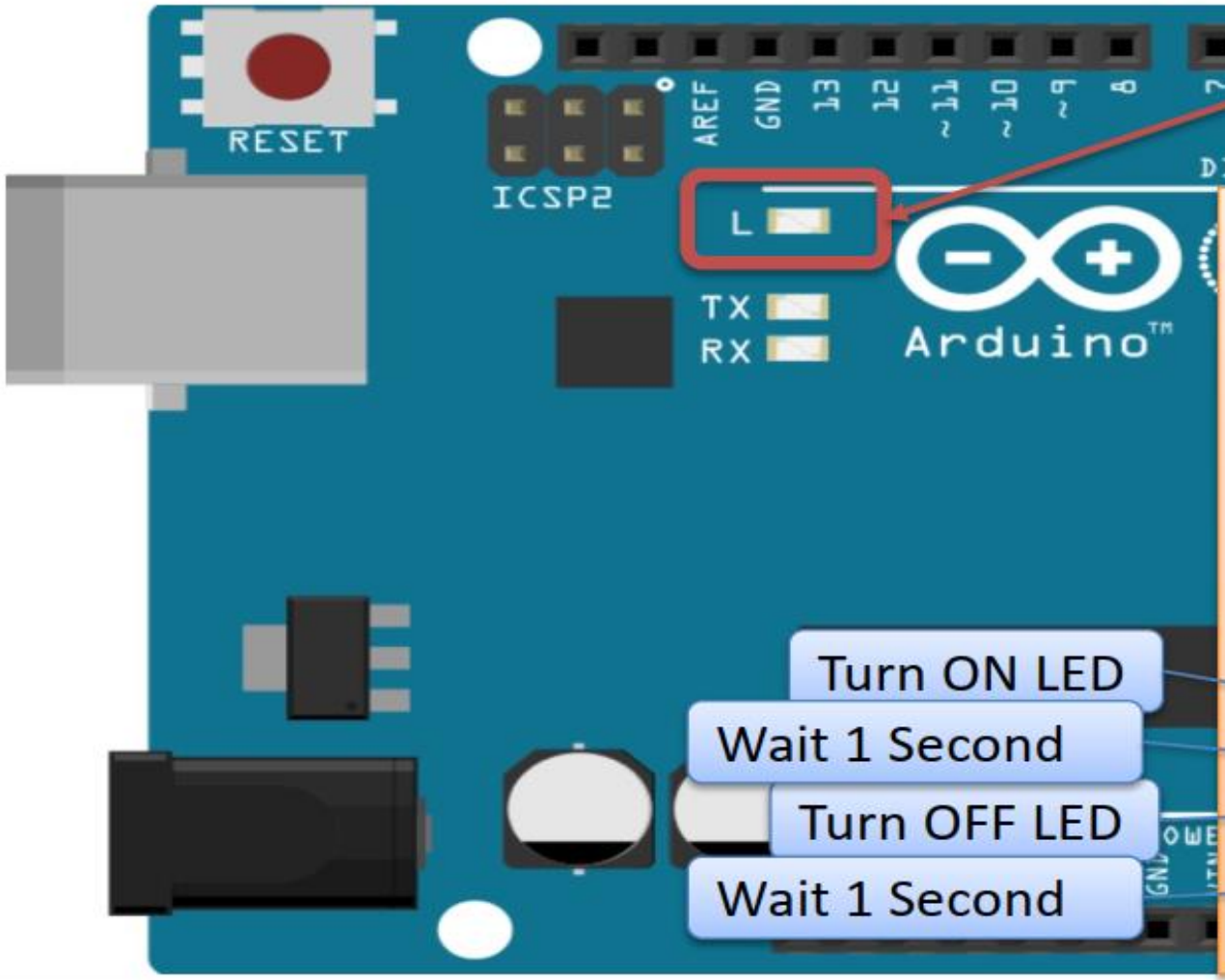
Do you get an Error?

Choose correct Board (Arduino UNO) and the correct Port for Communication between PC and Arduino Board





Blinking LED Example



Arduino UNO has a built-in LED that we can control

```
int ledPin = LED_BUILTIN;
int timerWait = 100;

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

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(timerWait);
  digitalWrite(ledPin, LOW);
  delay(timerWait);
}
```

Turn ON LED

Wait 1 Second

Turn OFF LED

Wait 1 Second

Code Comments

```
void setup()
{
        pinMode(13, OUTPUT);
}

void loop()
{
    digitalWrite(13, HIGH);
    delay(1000);
    digitalWrite(13, LOW);
    delay(1000);
}
```

Which Pin (0, 1, 3, ...) are you using?

```
pinMode(pin, mode);
```

A Digital Pin can either be an INPUT or an OUTPUT. Since we shall use it to turn-on a LED, we set it to OUTPUT.

```
digitalWrite(pin, value);
```

Turn-on LED

Turn-off LED

A Digital Pin can have 2 values, either **HIGH** or **LOW**

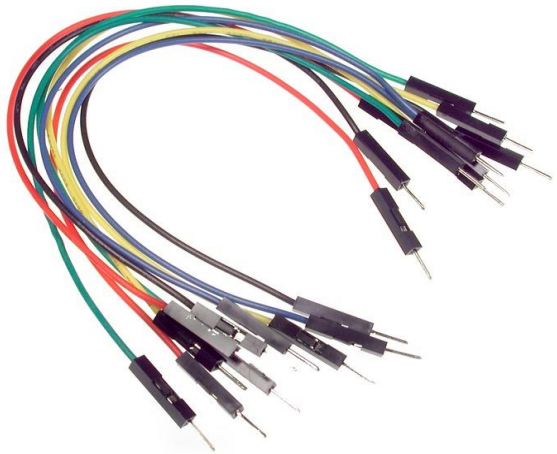
```
delay(ms);
```

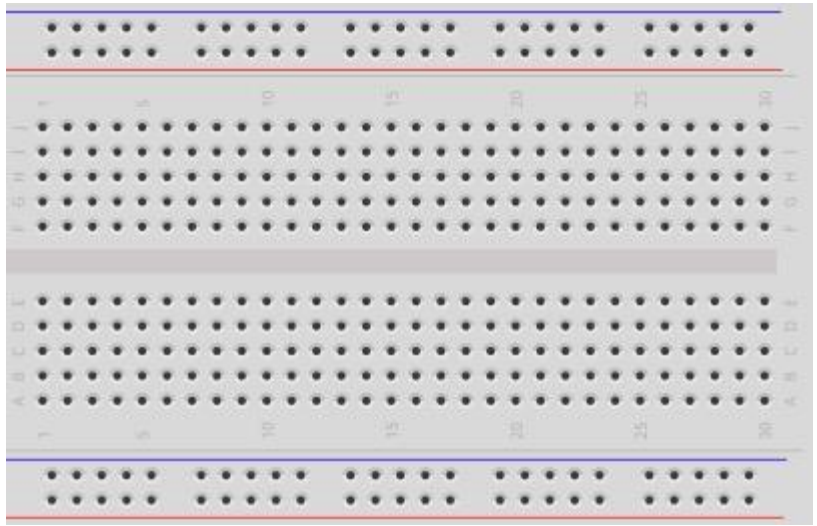
The delay() function makes a small pause in milliseconds (ms), e.g., delay(1000) pause the program for 1 second

External LED Example

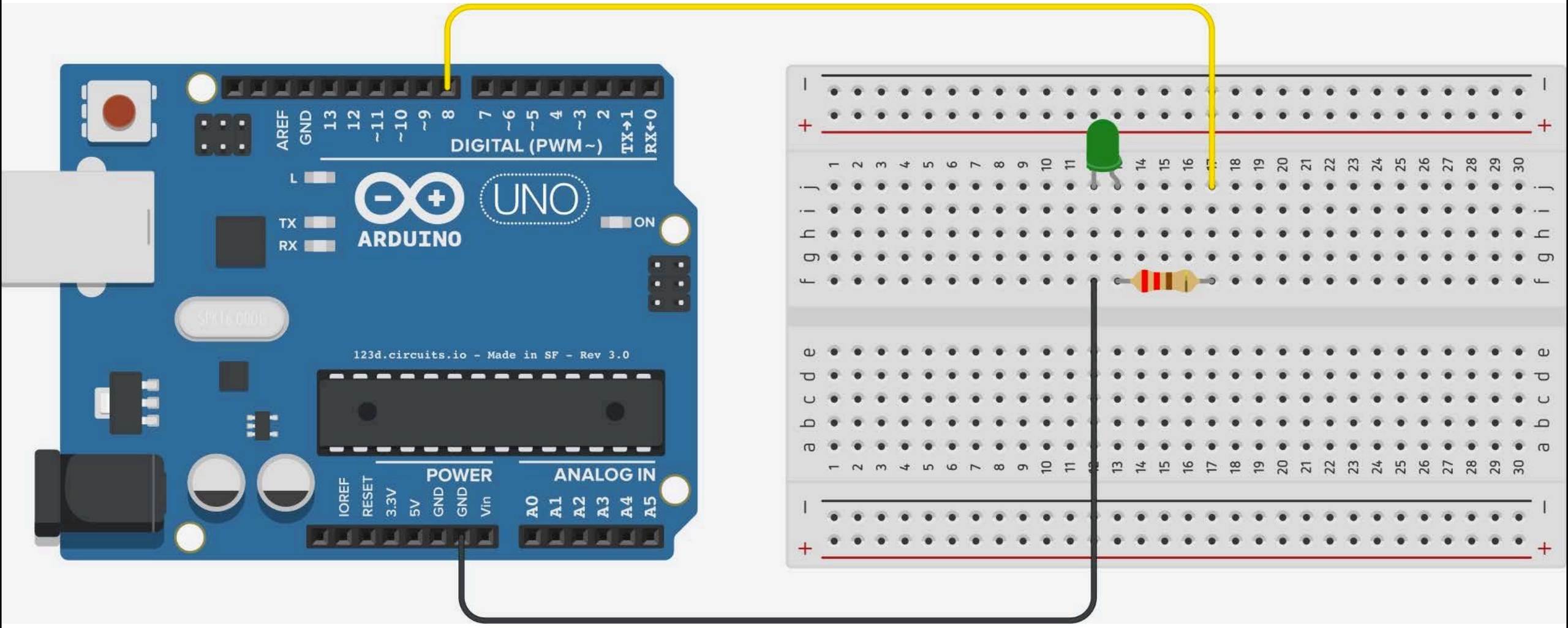
- So far, we have just used the Arduino itself
- Typically, we want to connect external components like LEDS, Temperature Sensors, etc.

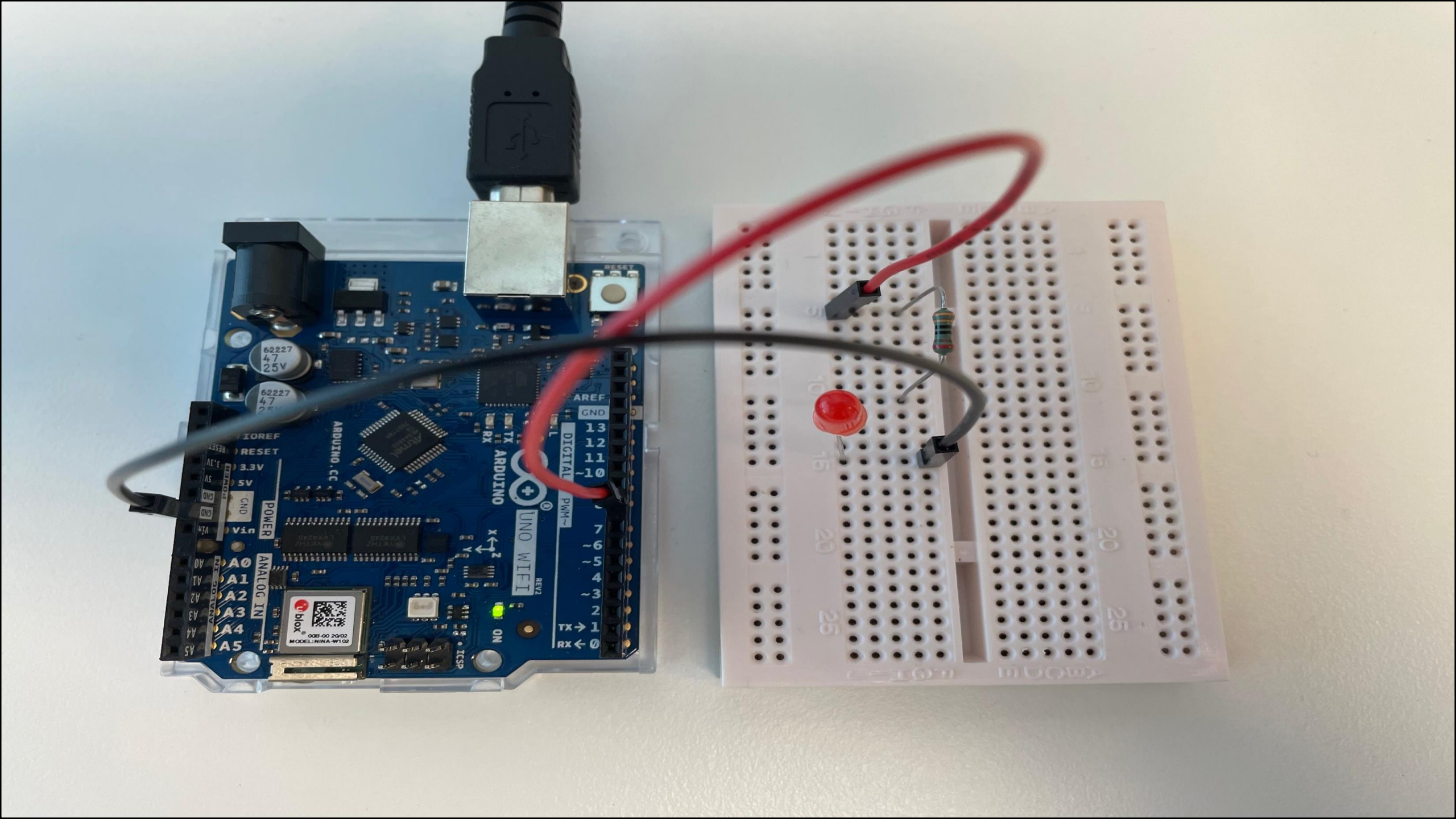
- Let's start by using and program an external LED
- External LED Example
  - What do we need?
  - Breadboard
  - LED
  - Wires
  - Resistor (e.g., $R = 270\Omega$)





- Wiring

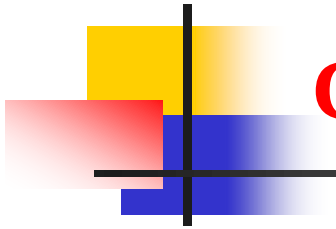

AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
123d.circuits.io - Made in SF - Rev 3.0
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


RESET
AREF
GND
13
12
11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL
PWM~
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO.CC
ARDUINO
UNO WIFI
REV2
ON
ICSP
TX
RX
u-blox
MODEL:NINA-W102

Code

```
int ledPin = 8;
int timerWait = 1000;

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

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(timerWait);
  digitalWrite(ledPin, LOW);
  delay(timerWait);
}
```

The code is the same as for the interna LED, you just need to change the pin number



Create Functions

- So far, we have used built-in functions like digitalWrite(), delay(), etc.
- Like other Programming Languages it is also possible to create and use your own Functions
- Let's “improve“ the LED example by creating some Functions

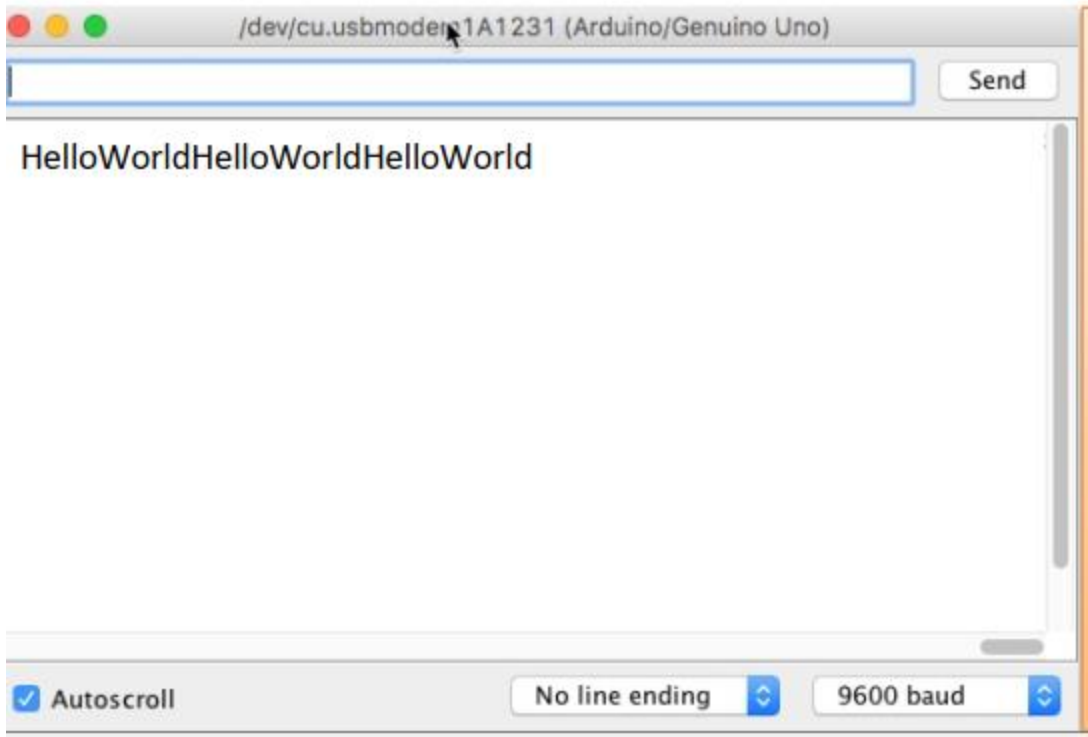
Code

```
int ledPin = 8;
int timerWait = 1000;

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

void loop() {
  ledon();
  delay(timerWait);
  ledoff();
  delay(timerWait);
}

void ledon(){
  digitalWrite(ledPin, HIGH);
}

void ledoff(){
  digitalWrite(ledPin, LOW);
}
```

Self-made Functions

Serial Monitor

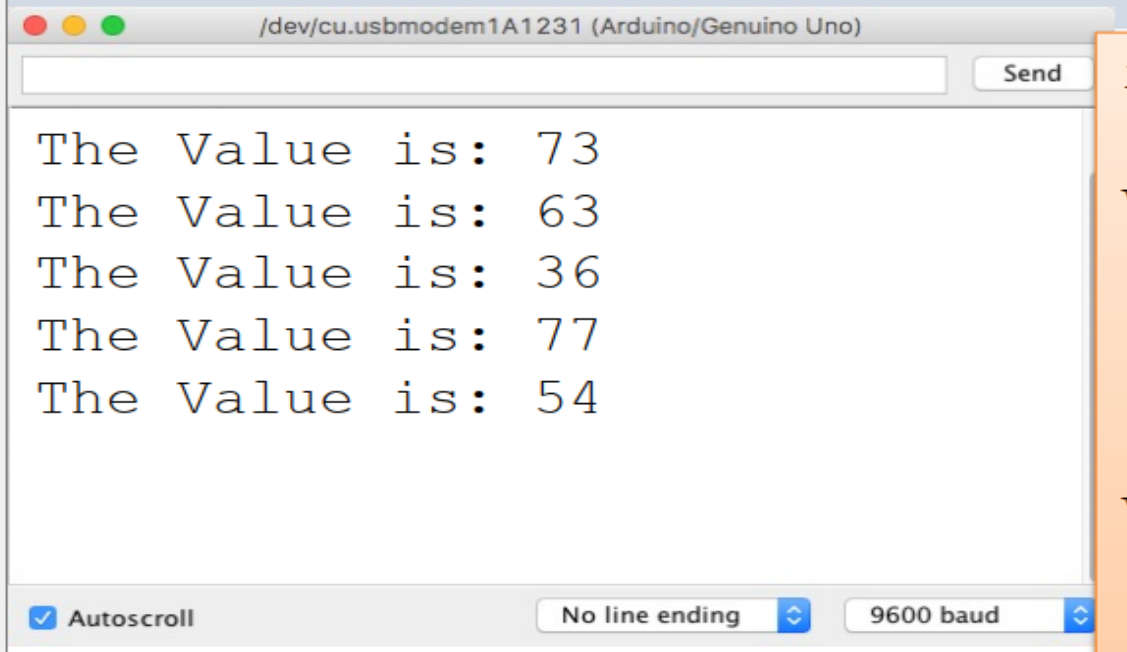
- The Arduino works like an embedded system where you download the code to the device, and then it runs independently of your Computer
- You can remove the USB cable and only connect a Power Supply (or using a 9V Battery)
- This means an Arduino application has no Graphical User Interface and you cannot use a Mouse or a keyboard to communicate with the program
- You use the Serial Monitor when **Debugging** Arduino programs or when you want to **show data or values from your program**. You need to have Arduino connected to your PC (using the USB cable) in order to use the Serial Monitor.



Serial.print()
Serial.println()

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

void loop()
{
      Serial.print("Hello World");
      delay(1000);
}
```

Serial Monitor Example



Here you see how we can write a value to the Serial Monitor. This can be a value from a sensor, e.g., a temperature sensor.

```
int myValue = 0;

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

void loop()
{
    myValue = random(100);
    Serial.print("The Value is: ");
    Serial.println(myValue);
    delay(1000);
}
```

Example

This Example uses both built-in functions in addition to self-made functions. The results are written to the Serial Monitor

```
int z;int a;int b;
void setup()
{
        Serial.begin(9600);
}
void loop()
{
        a = random(100);
        b = random(100);
        z = calculate(a,b); //Adding 2 Numbers

        //Write Values to Serial Monitor
        Serial.print(a);
        Serial.print(" + ");
        Serial.print(b);
        Serial.print(" = ");
        Serial.println(z);

        delay(1000);
}
float calculate(int x, int y)
{
        return (x + y);
}
```

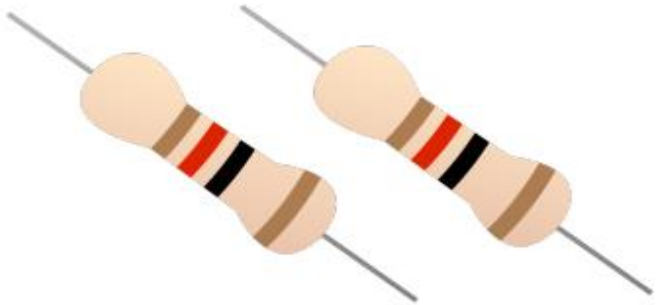
Sensors and Actuators

Temperature sensor

Switch

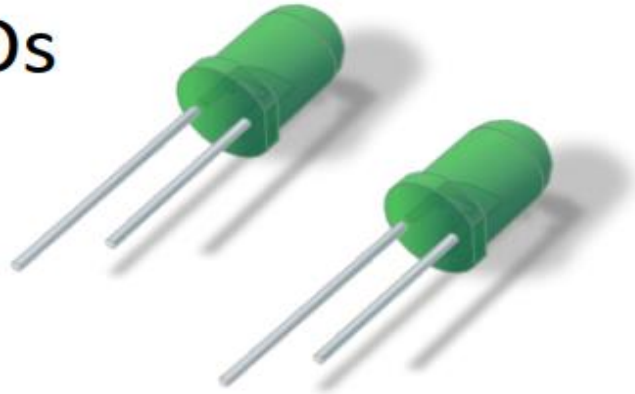
Potentiometer

Light sensor



Thermistor

LEDs



Sensors and Actuators

Temperature Sensor

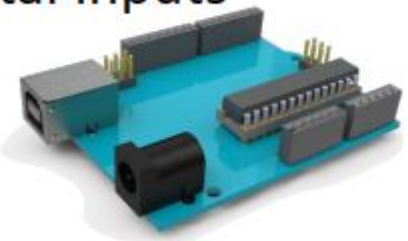
Push Button

Light Sensor

Potentiometer

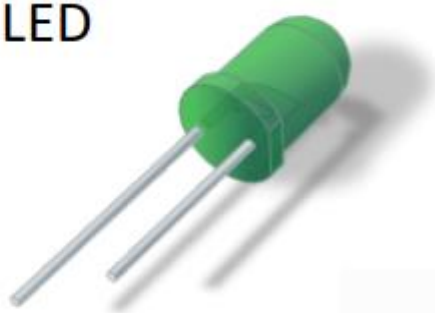
Analog/Digital Inputs

Sensors



Signals from the surroundings

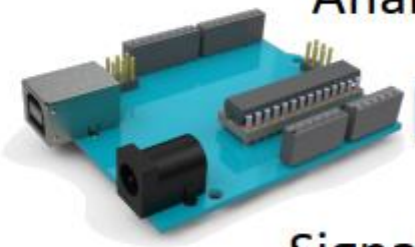
LED



Motor

Buzzer

Analog/Digital Outputs

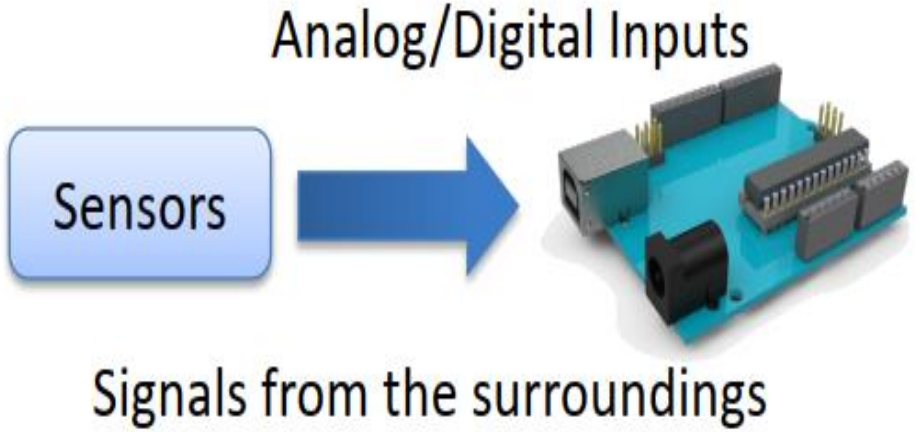


Actuators

Signals to the surroundings

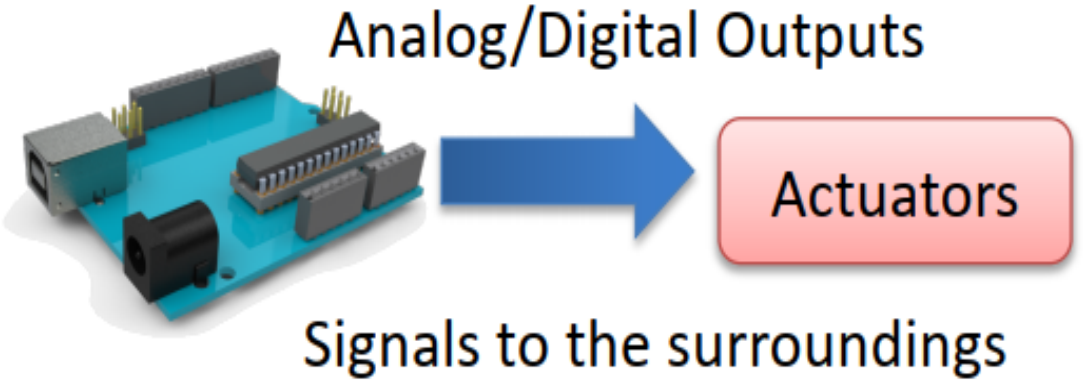
Sensors

- A Sensor is a converter that measures a physical size and converts it to a signal that can be read by an instrument, data acquisition device, or an Arduino in our case



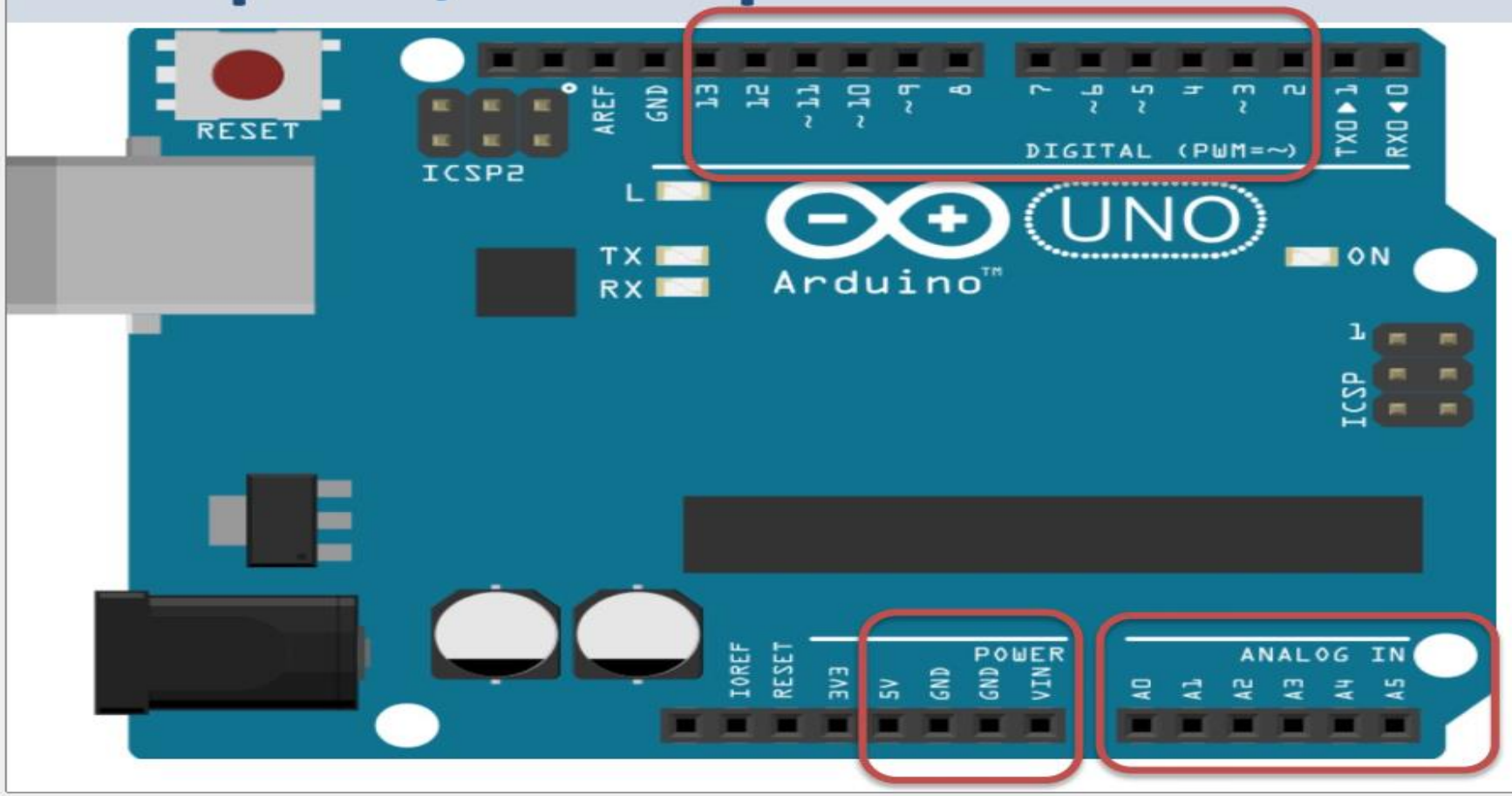
- Examples: Temperature sensor, Pressure sensor, etc.
- We use **Analog In** pins and **Digital In** pins for reading data from Sensors into the Arduino

Actuators



- An Actuator is a kind of motor that moves or controls a mechanism or system.
- It is powered by an energy source, typical electric current, hydraulic fluid pressure, or air pressure, and converts this energy into motion.
- Examples: Engine, Pump, Valve, etc**.** We use **Digital Out** pins for controlling the Actuators from the Arduino. Note! Arduino UNO has no Analog Out pins

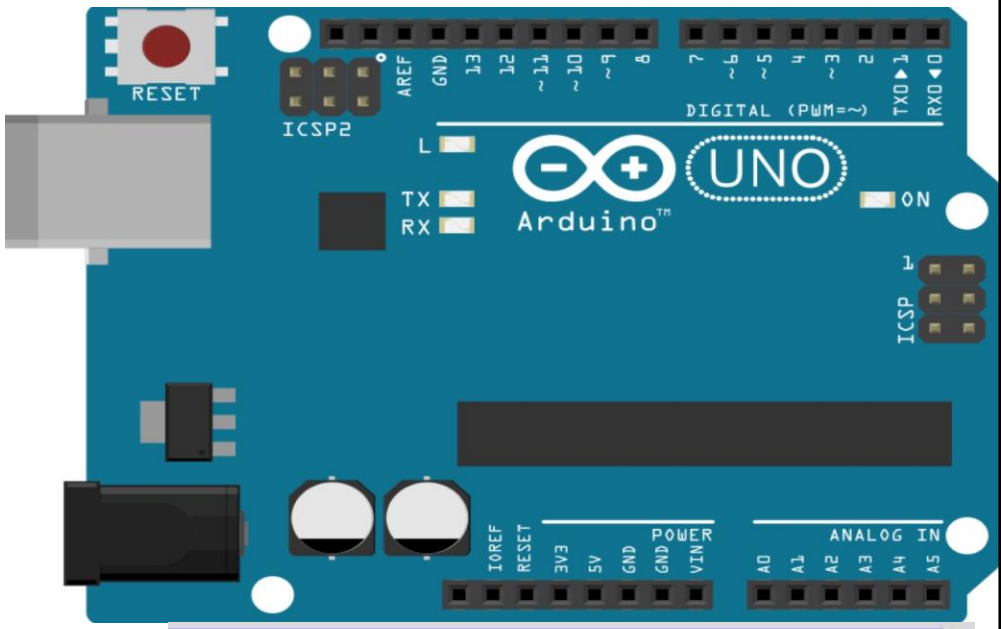
Input/Output Pins on Arduino

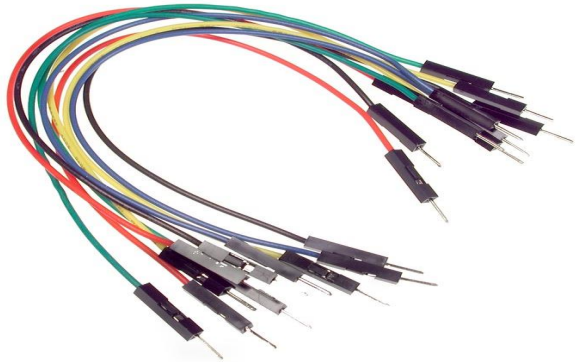


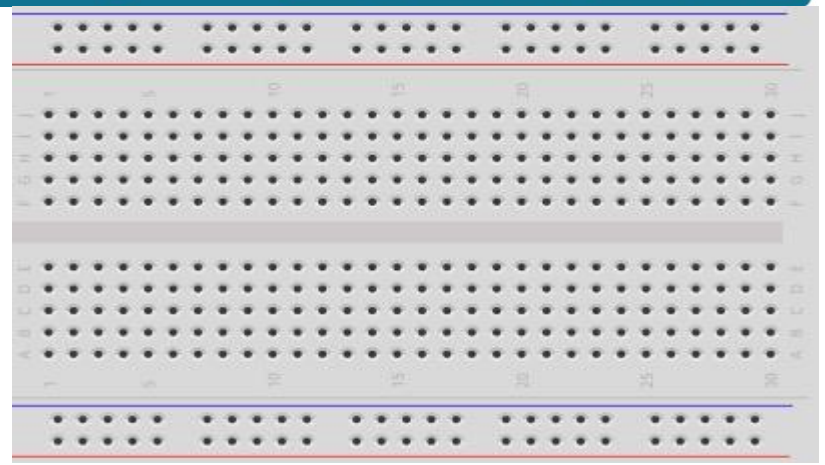
The Digital pins can be set to be either Digital In or Digital Out. This is done in your Arduino program

Temperature Sensor Example

- In this example we will use a small temperature sensor to read the temperature in the room.
- The Temperature Sensor is called “TMP36”
  • In this example we will use one of the "Analog In" ports on the Arduino board
- Nesessary Equipment
  - Arduino
  - Breadboard
  - TMP36
  - Wires (Jumper Wires)

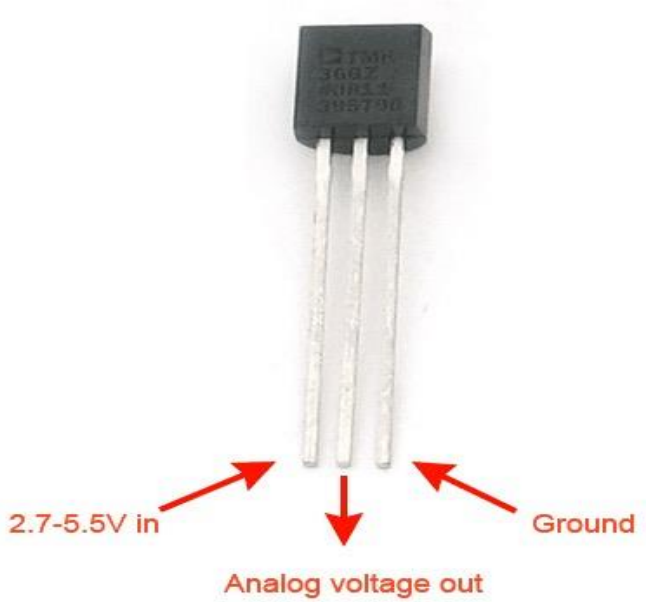


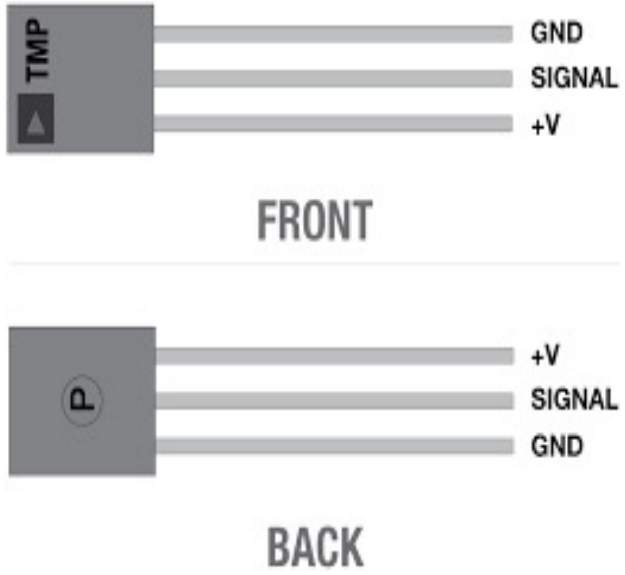




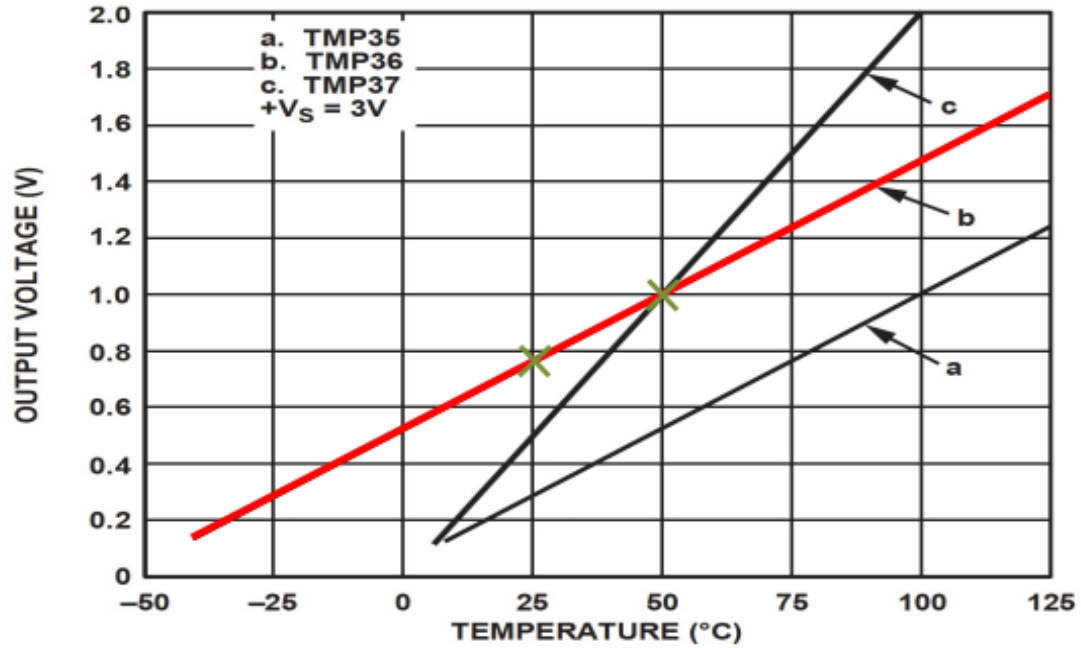
TMP36

- TMP is a small, low-cost temperature sensor and cost about $1 (you can buy it “everywhere”)





Linear Scaling



This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75}(x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

Convert form Voltage (V) to degrees Celsius
From the Datasheet we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

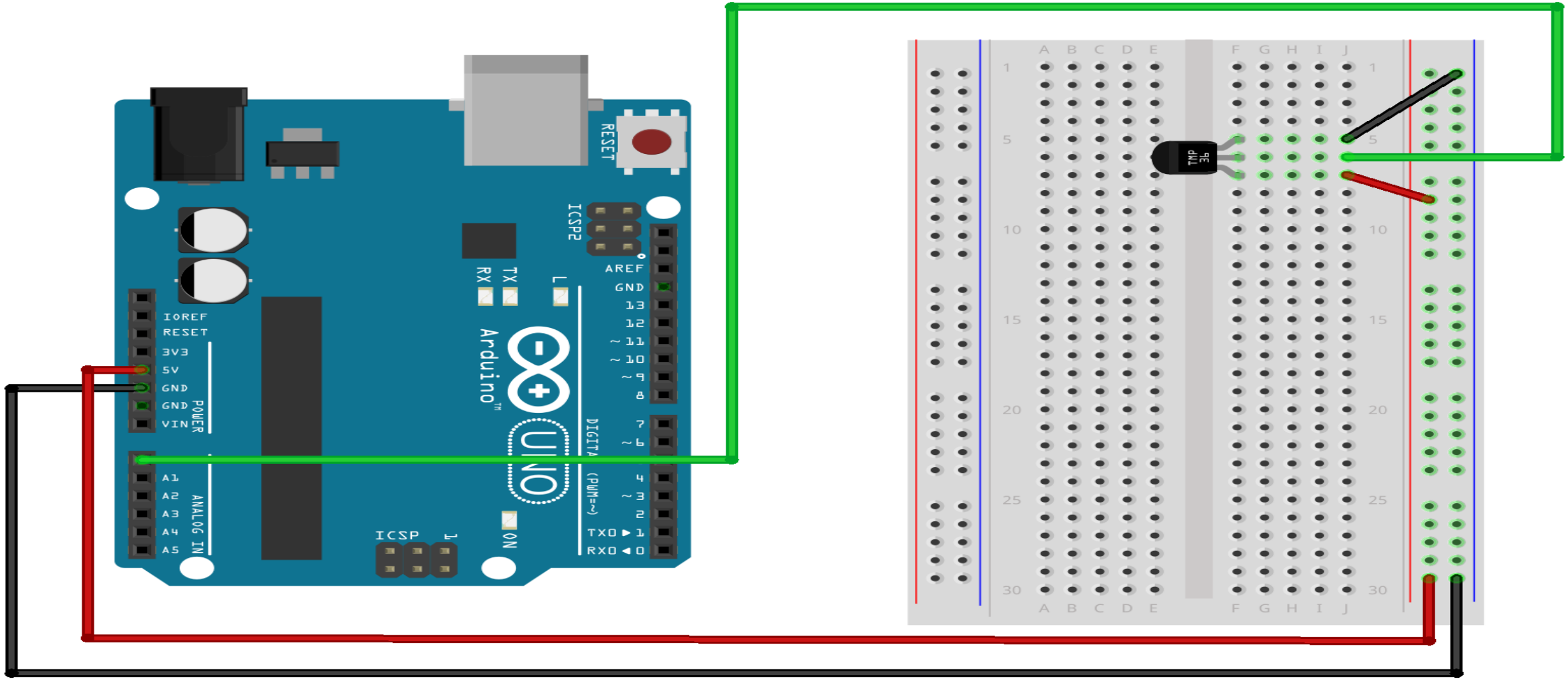
There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

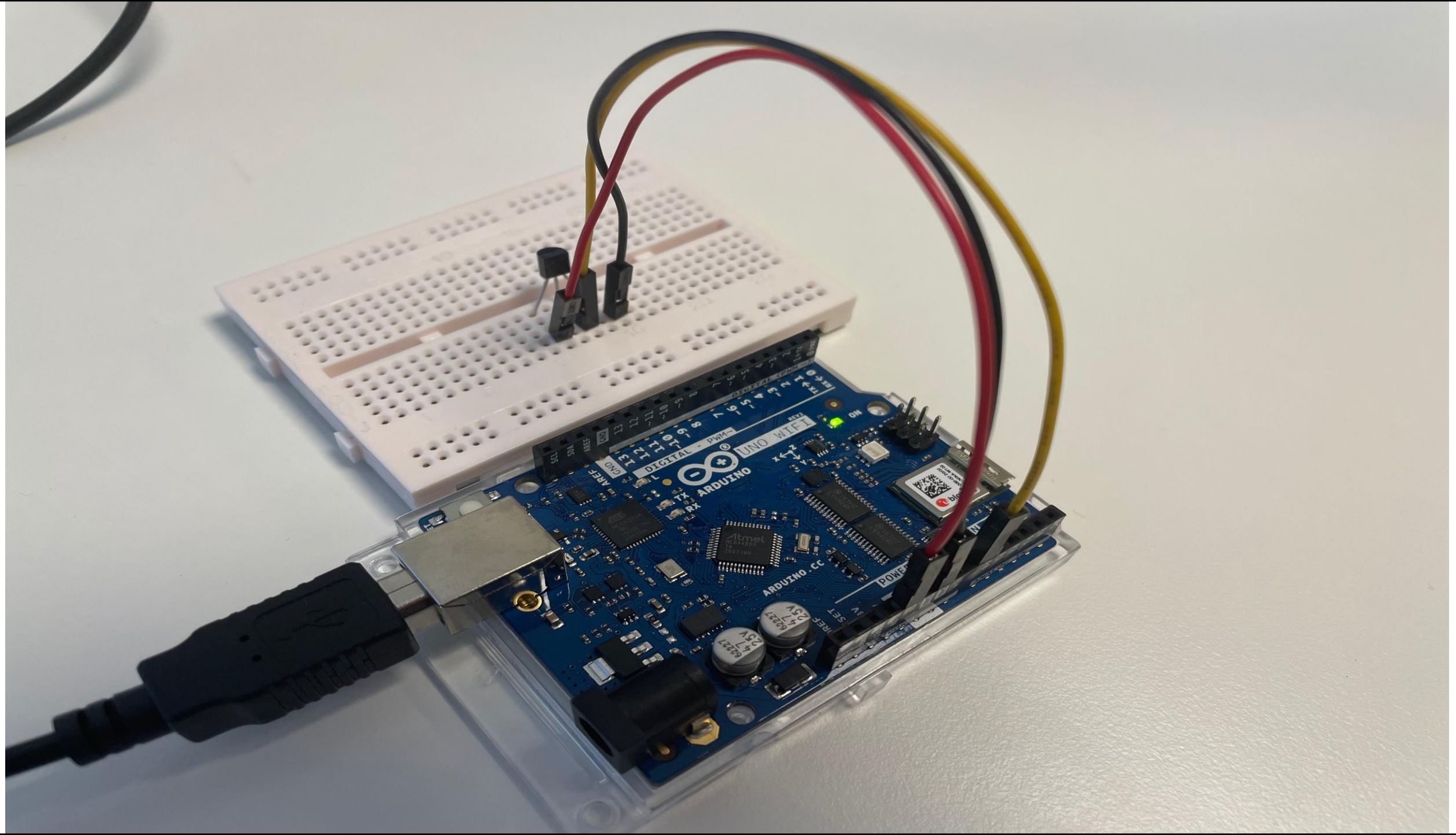
We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Wiring



fritzing


DIGITAL - PWM~
UNO WIFI
ARDUINO
ARDUINO.CC
POWER

Temperature Conversion

- We want to present the value from the sensor in degrees Celsius:
  - 1. The function analogRead() gives a value between 0 and 1023 (Arduino UNO has a built-in 10-bit ADC, 2^10=1024)
  - 2. Then we convert this value to 0-5V.
  - 3. Finally, we convert to degrees Celsius using information from the Datasheet presented on the previous page ($y = 100x - 50$)
  - 4. The we can, e.g., show the Temperature value in the Serial Monitor

Code

```
const int temperaturePin = 0;

float adcValue;
float voltage;
float degreesC;

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

void loop()
{

  adcValue = analogRead(temperaturePin);

  voltage = (adcValue*5)/1023;

  degreesC = 100*voltage - 50;

  Serial.print("ADC Value: ");
  Serial.print(adcValue);

  Serial.print("  voltage: ");
  Serial.print(voltage);

  Serial.print("  deg C: ");
  Serial.println(degreesC);

  delay(1000);
}
```

Arduino Programming

- We have already created and used Variables
- We have also created and used Functions
- Basically, Arduino Programming is very similar to other Programming Languages, so we can also use For Loops, While Loops, create and use Arrays, If..Else, etc

For Loop

In this program we use a For Loop to find the Sum of 100 Random Numbers.

Then we find the Average.

The Sum and Average are written to the Serial Monitor.

```
int x; int sum = 0; float average = 0;
void setup()
{
        Serial.begin(9600);
}
void loop()
{
    sum = 0;
    for (int i = 0; i<100; i++)
    {
        x = random(100);
        sum = sum + x;
    }

    average = sum / 100;
    Serial.print(" Sum = ");
    Serial.print(sum);
    Serial.print(" , Average = ");
    Serial.println(average);
    delay(1000);
}
```

While Loop

In this program we use a While Loop to find the Sum of 100 Random Numbers.

Then we find the Average.

The Sum and Average are then written to the Serial Monitor.

```
int x; int sum = 0; float average = 0; int i;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  sum = 0;
  i=0;
  while (i<100)
  {
    x = random(100);
    sum = sum + x;
    i++;
  }

  average = sum / 100;
  Serial.print(" Sum = ");
  Serial.print(sum);
  Serial.print(" , Average = ");
  Serial.println(average);
  delay(1000);
}
```

Arrays

```
const int arraysize = 100;
int x;
int sum = 0;
float average = 0;
int myarray[arraysize];

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

void loop()
{
    sum = 0;
    for (int i = 0; i < arraysize; i++)
    {
        x = random(200);
        myarray[i] = x;
    }
    sum = calculateSum(myarray);
    average = sum / 100;
    Serial.print(" Sum = ");
    Serial.print(sum);
    Serial.print(" , Average = ");
    Serial.println(average);
    delay(1000);
}

int calculateSum (int sumarray[])
{
    for (int i = 0; i < arraysize; i++)
    {
        sum = sum + sumarray[i];
    }
    return sum;
}
```

If .. Else

```
int number;

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

void loop()
{
  number = random(0,100);

  if (number < 50)
  {
     Serial.println("Small Number");
  }
  else
  {
     Serial.println("Large Number");
  }
  delay(1000);
}
```