Unit-4:Introduction of Arduino and Sensors :

Arduino board (pin configuration and description),

Basic principle of ultrasonic sensor, Temperature, IR sensor.

Introduction and Programming Arduino

Agenda

- Introduction to Arduino Boards
- Getting started with Arduino IDE
- Arduino Programming and Proteus designs
- Interfacing Output (LED) & Input (Key) devices
- Working with IR Sensor, LDR and its Interfacing
- Serial Communication feature of Arduino
- Working with DHT11/22 and its interfacing
- Ultrasonic Sensor Interfacing

What is Arduino?

- A microcontroller board, contains on-board power supply, USB port to communicate with PC, and an Atmel microcontroller chip.
- It simplify the process of creating any control system by providing the standard board that can be programmed and connected to the system without the need to any sophisticated PCB design and implementation.
- It is an open source hardware, any one can get the details of its design & modify it or make his own one himself.

What can it do?

- Sensors (to sense stuff)
  - Push buttons, touch pads, tilt switches.
  - Variable resistors (eg. volume knob / sliders)
  - Photoresistors (sensing light levels)
  - Thermistors (temperature)
  - Ultrasound (proximity range finder)
- Actuators (to do stuff)
  - Lights, LED’s
  - Motors
  - Speakers
  - Displays (LCD)

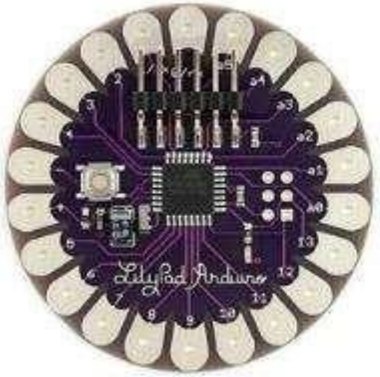
Why Arduino?

- It is Open Source, both in terms of Hardware and Software.
- It is cheap(1300₹), the hardware can be built from components or a prefab board can be purchased for approx. 900₹.
- It can communicate with a computer via serial connection over USB.
- It can be powered from USB or standalone DC power.
- It can run standalone from a computer (chip is programmable) and it has memory (a small amount).
- It can work with both Digital and Analog electronic signals. Sensors and Actuators.
- You can make cool stuff! Some people are even making simple robots.

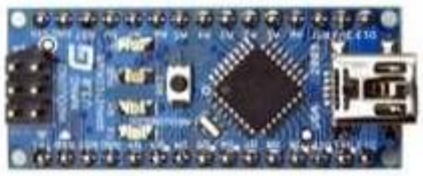
Different types of Arduino boards:

UNO

Mega

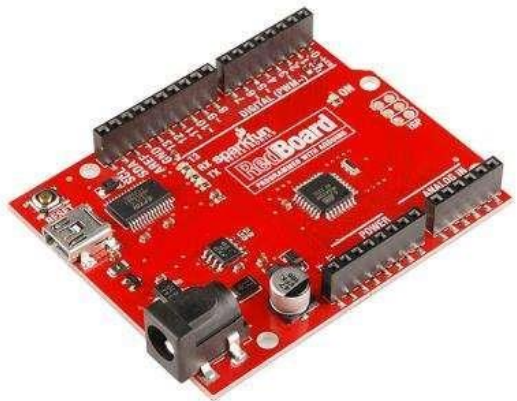

LilyPad

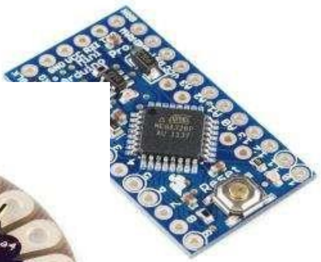
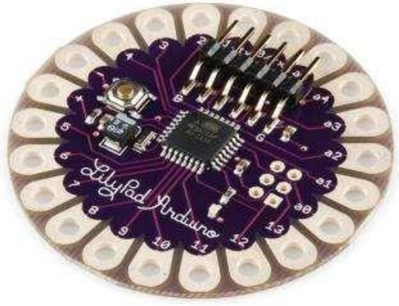
Arduino BT


Arduino
Nano

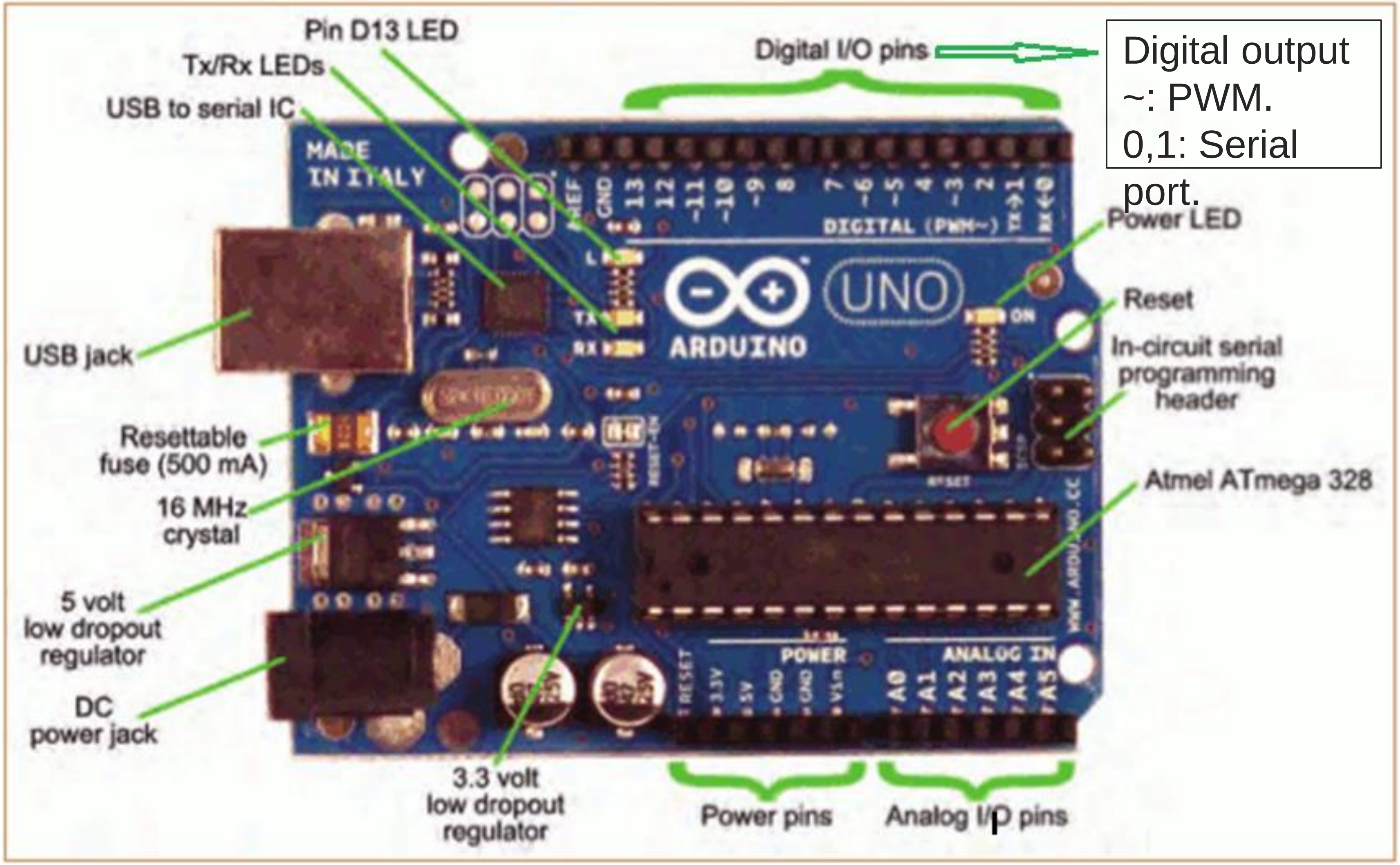
Arduino
Mini

Arduino & Arduino compatible boards:



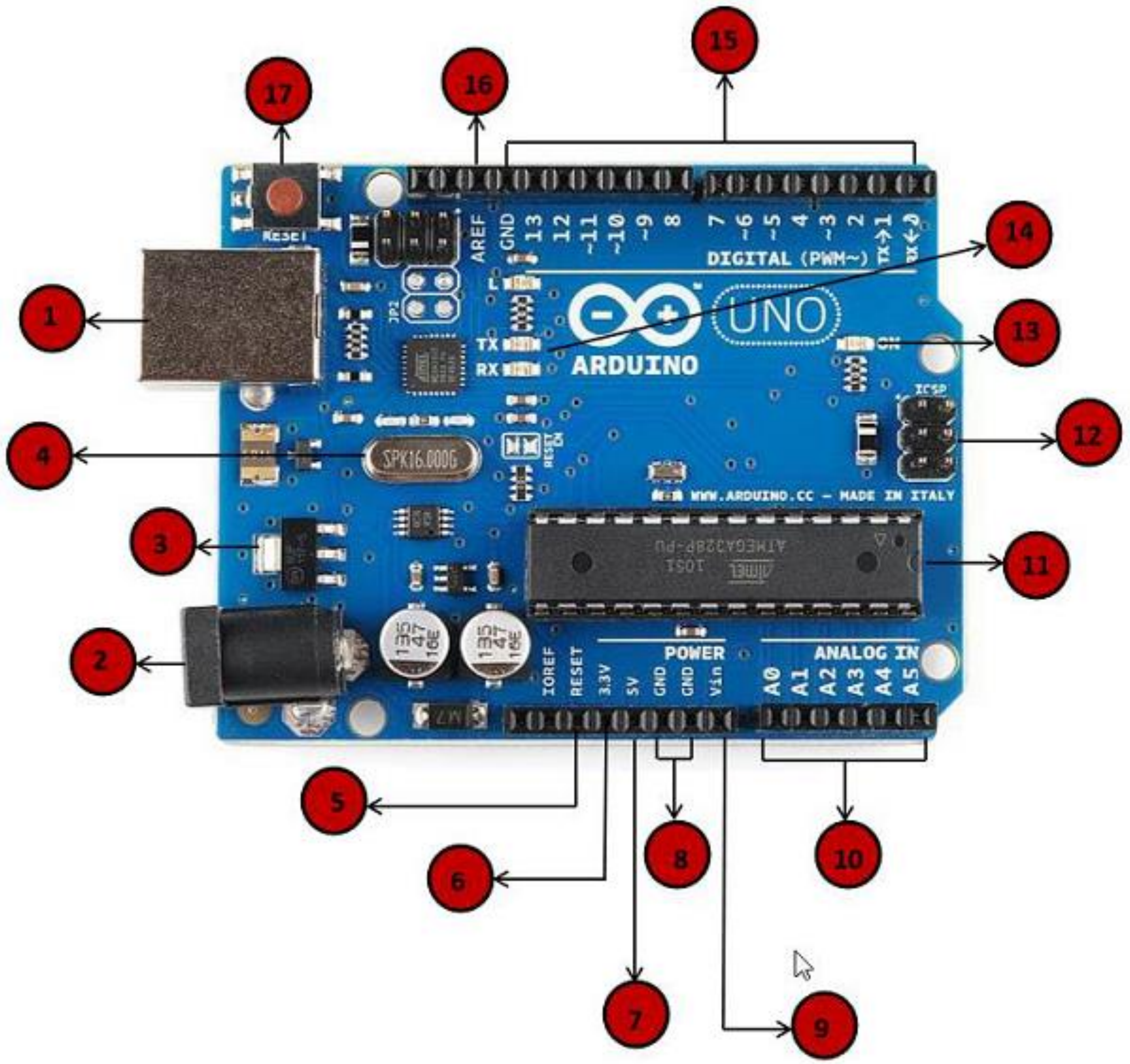
Arduino Uno Board



Digital output
~: PWM.
0,1: Serial port.

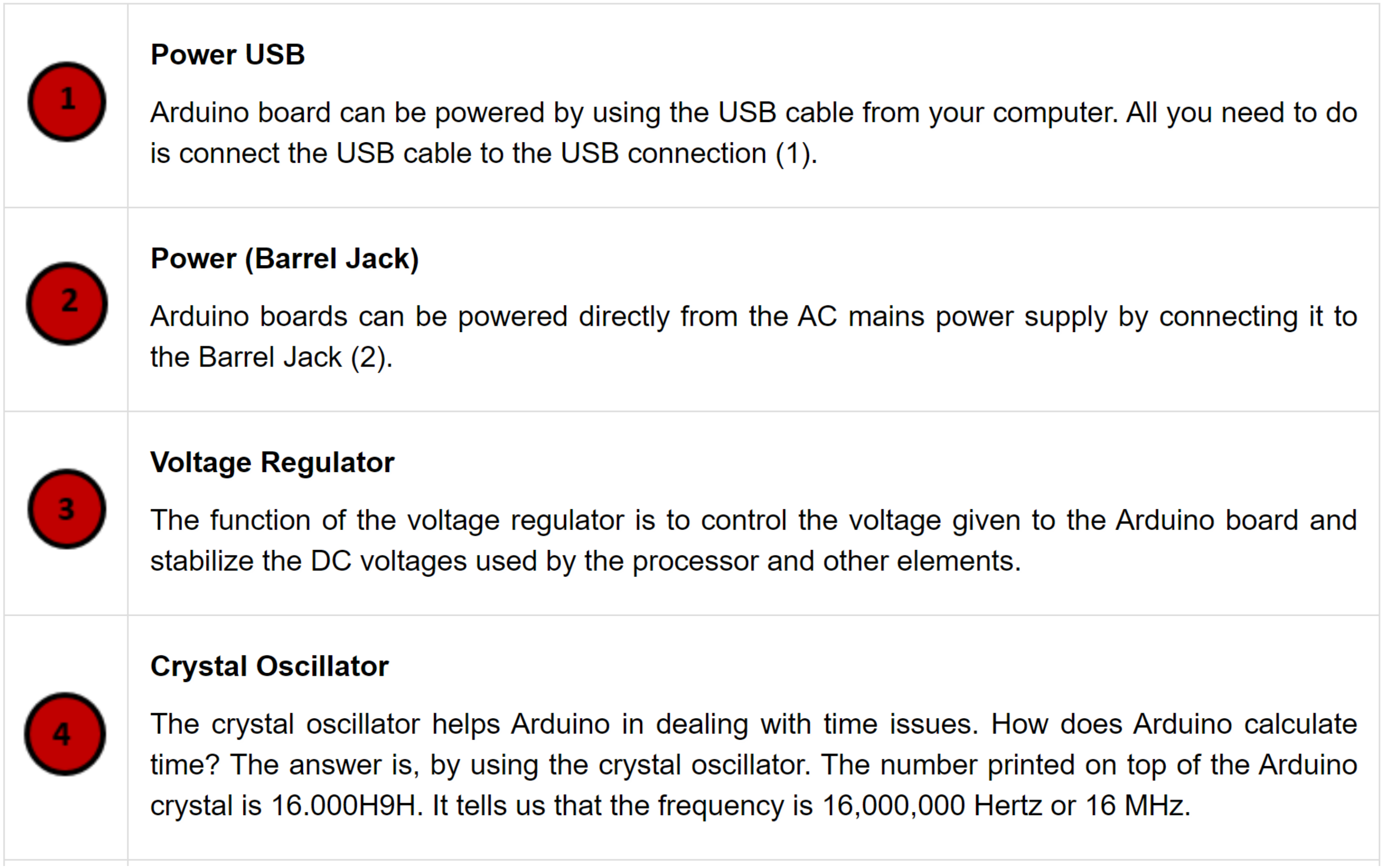
Connectors/Cables to work with Uno:

Arduino Uno Board Description



1- Power USB
2- Power (Barrel Jack)
3- Voltage Regulator
4- Crystal Oscillator
5,17- Arduino Reset
6,7,8,9- Pins (3.3, 5, GND, Vin)
10- Analog pins
11- Main microcontroller
12- ICSP pin
13- Power LED indicator
14- TX and RX LEDs
15- Digital I / O
16- Analog Reference

Arduino Uno Board Description (Cont..)



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.

Arduino Uno Board Description (Cont..)

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.

Arduino Uno Board Description (Cont..)

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.

Arduino Uno Board Description (Cont..)

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.

INPUT v/s OUTPUT

Referenced from the perspective of the microcontroller (electrical board).

Inputs is a signal / information going into the Arduino board.

Examples: Buttons Switches, Light Sensors, IR Sensors, Humidity Sensors, Temperature Sensors...

Output is any signal exiting the Arduino board.

Examples: LEDs, DC motor, servo motor, a piezo buzzer, relay, an RGB LED

https://getintopc.com/softwares/3d-cad/proteus-professional-2020-free-download/

Getting Started (Installing Arduino IDE and Setting up Arduino Board)

Check out: http://arduino.cc/en/Reference/HomePage , http://arduino.cc/en/Guide/HomePage .

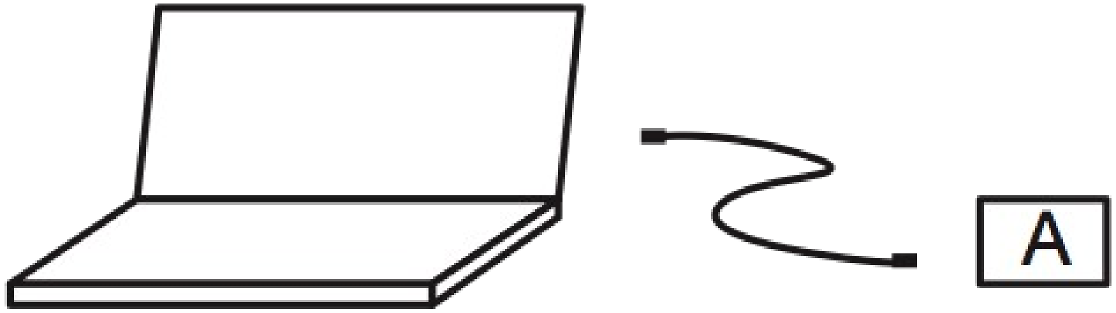
1. Download & install the Arduino environment (IDE) https://www.arduino.cc/en/Main/Software
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. Open the blink example
8. Upload the program

Basic Procedure

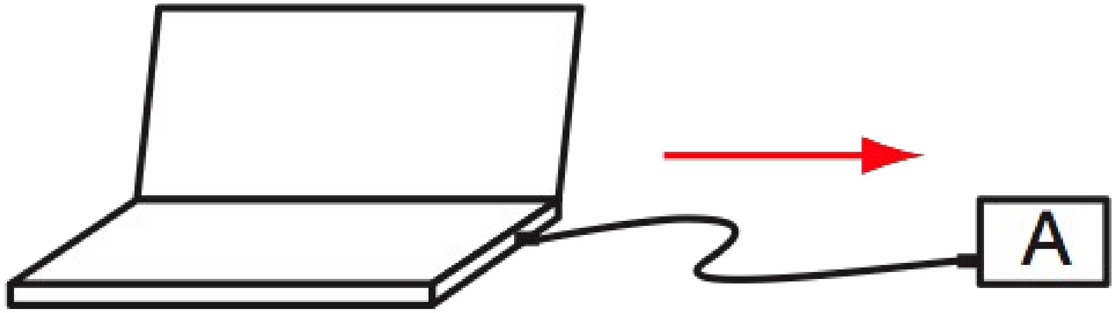
- Design the circuit:
  - What are electrical requirements of the sensors or actuators?
  - Identify inputs (analog inputs)
  - Identify digital outputs
- Write the code:
  - Build incrementally
    - Get the simplest piece to work first
    - Add complexity and test at each stage
    - Save and Backup frequently
  - Use variables, not constants
  - Comment liberally

Writing and Uploading Code into Arduino

Write sketch on PC

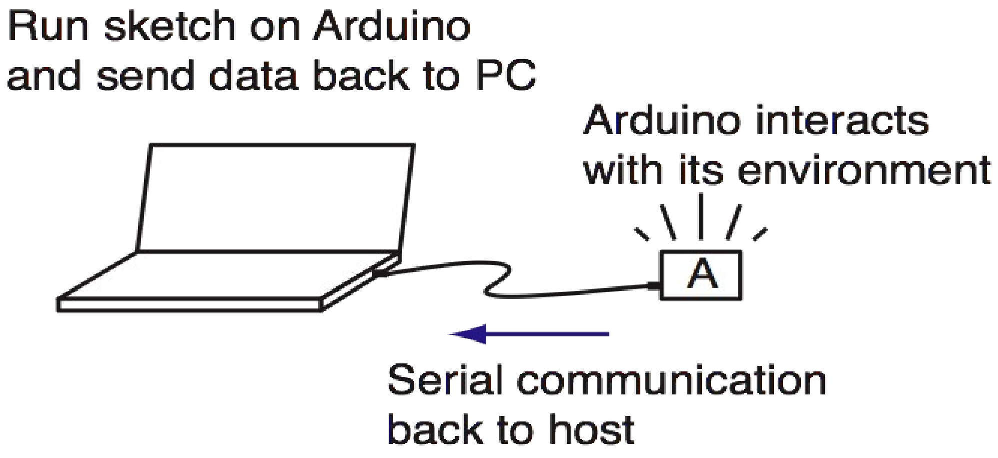


Download sketch to Arduino

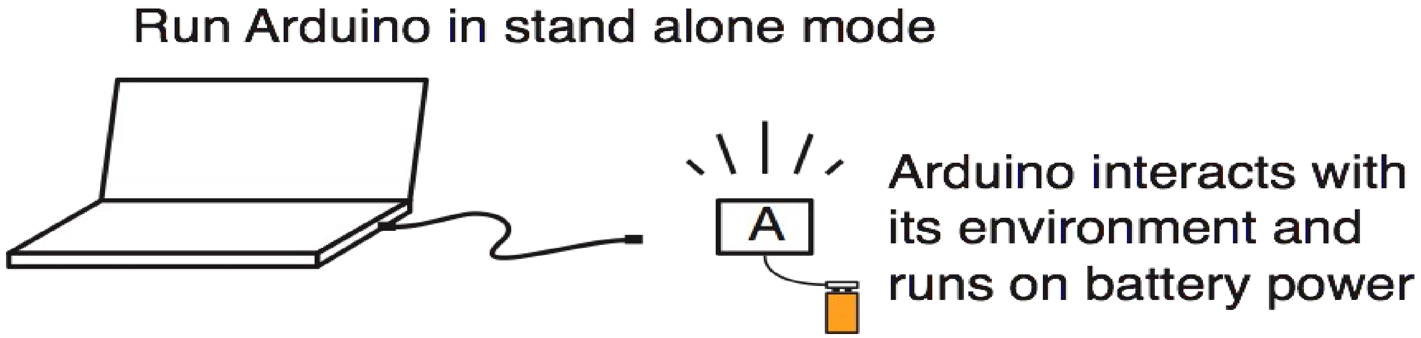


Running Code

- Running Code While Tethered


Run sketch on Arduino and send data back to PC
Arduino interacts with its environment
A
Serial communication back to host

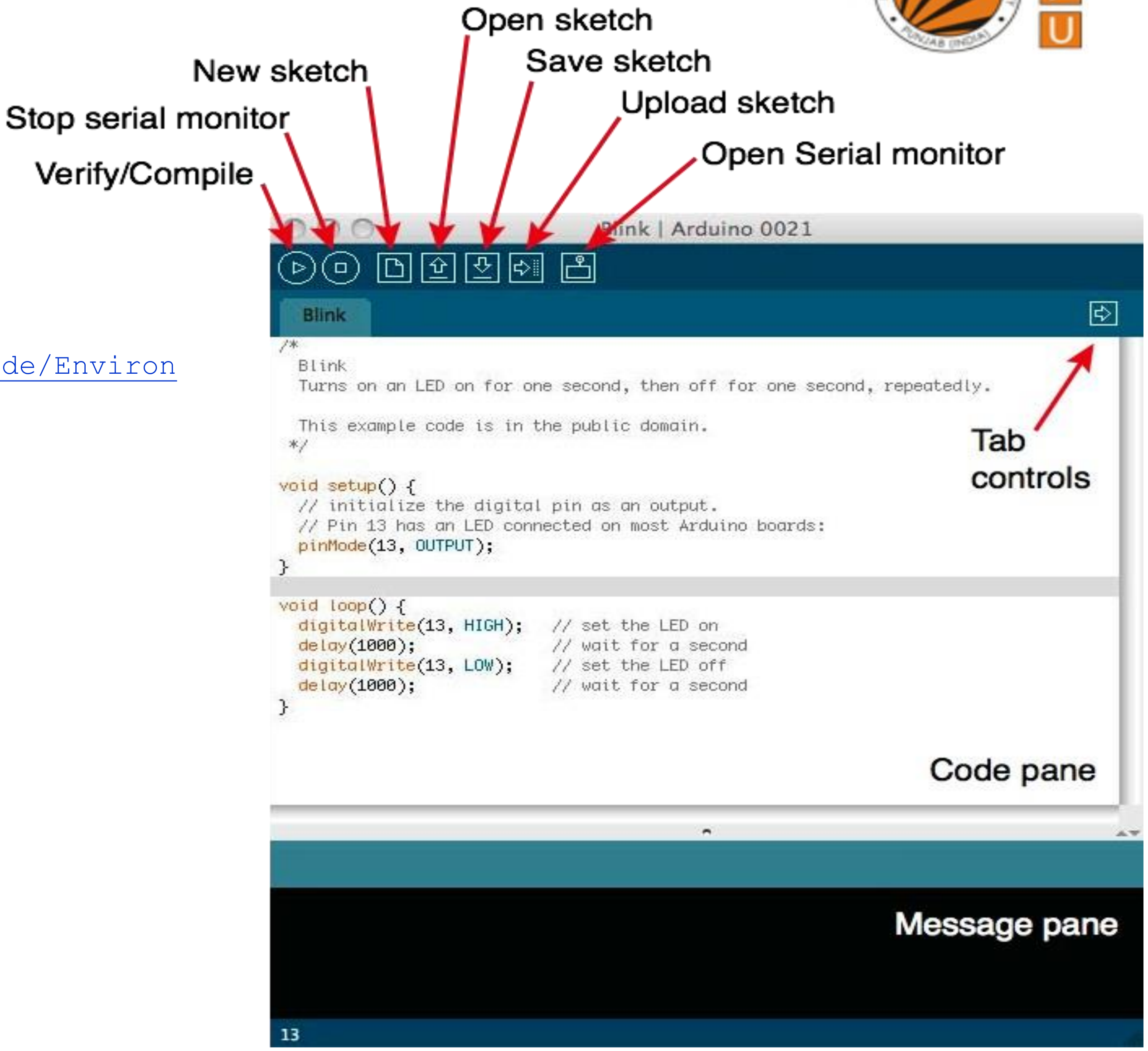
- Running Code Stand-Alone


Run Arduino in stand alone mode
A
Arduino interacts with its environment and runs on battery power

Arduino IDE

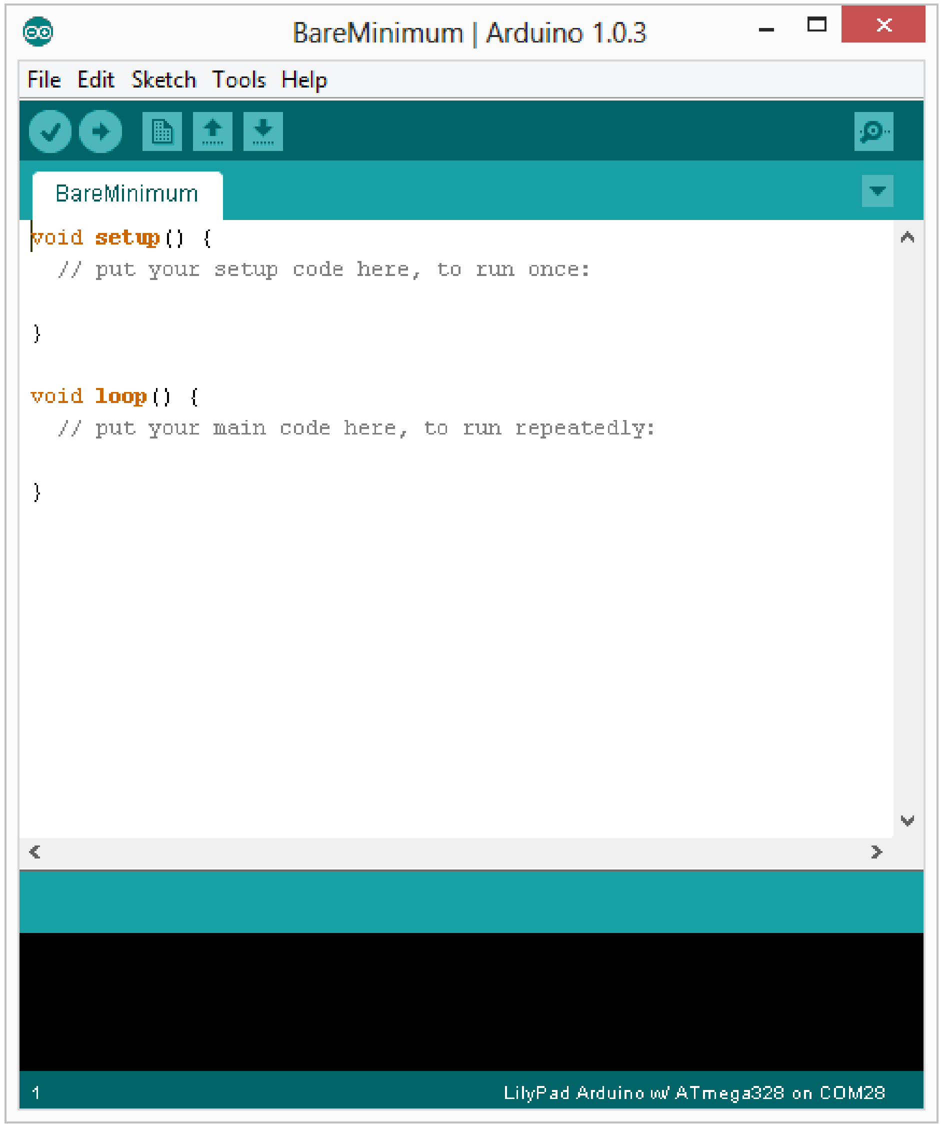
IDE =
Integrated Development Environment

http://www.arduino.cc/en/Guide/Environment



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);   // set the LED on
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // set the LED off
  delay(1000);              // wait for a second
}
```

Arduino IDE (Cont..)

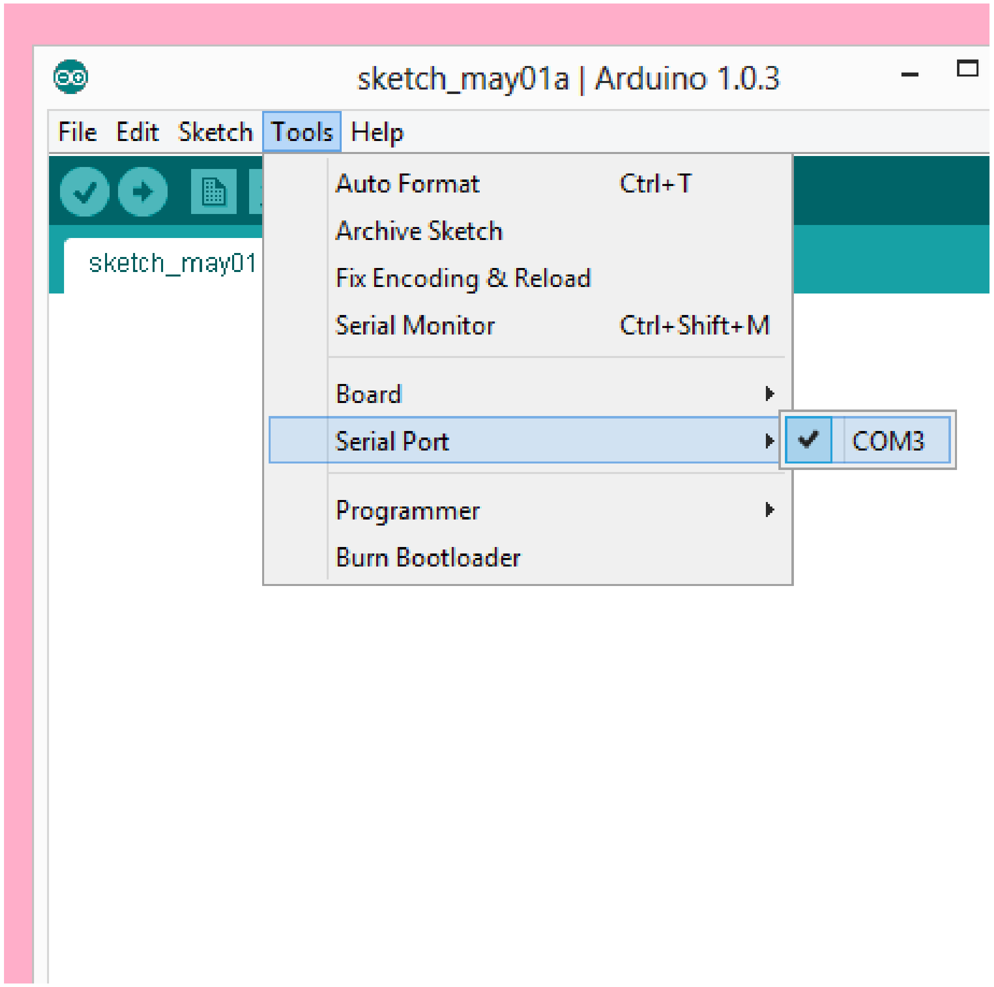

BareMinimum | Arduino 1.0.3
File Edit Sketch Tools Help
BareMinimum
LilyPad Arduino w/ ATmega328 on COM28

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Two required functions / methods / routines:

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

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

Arduino IDE (Cont..)

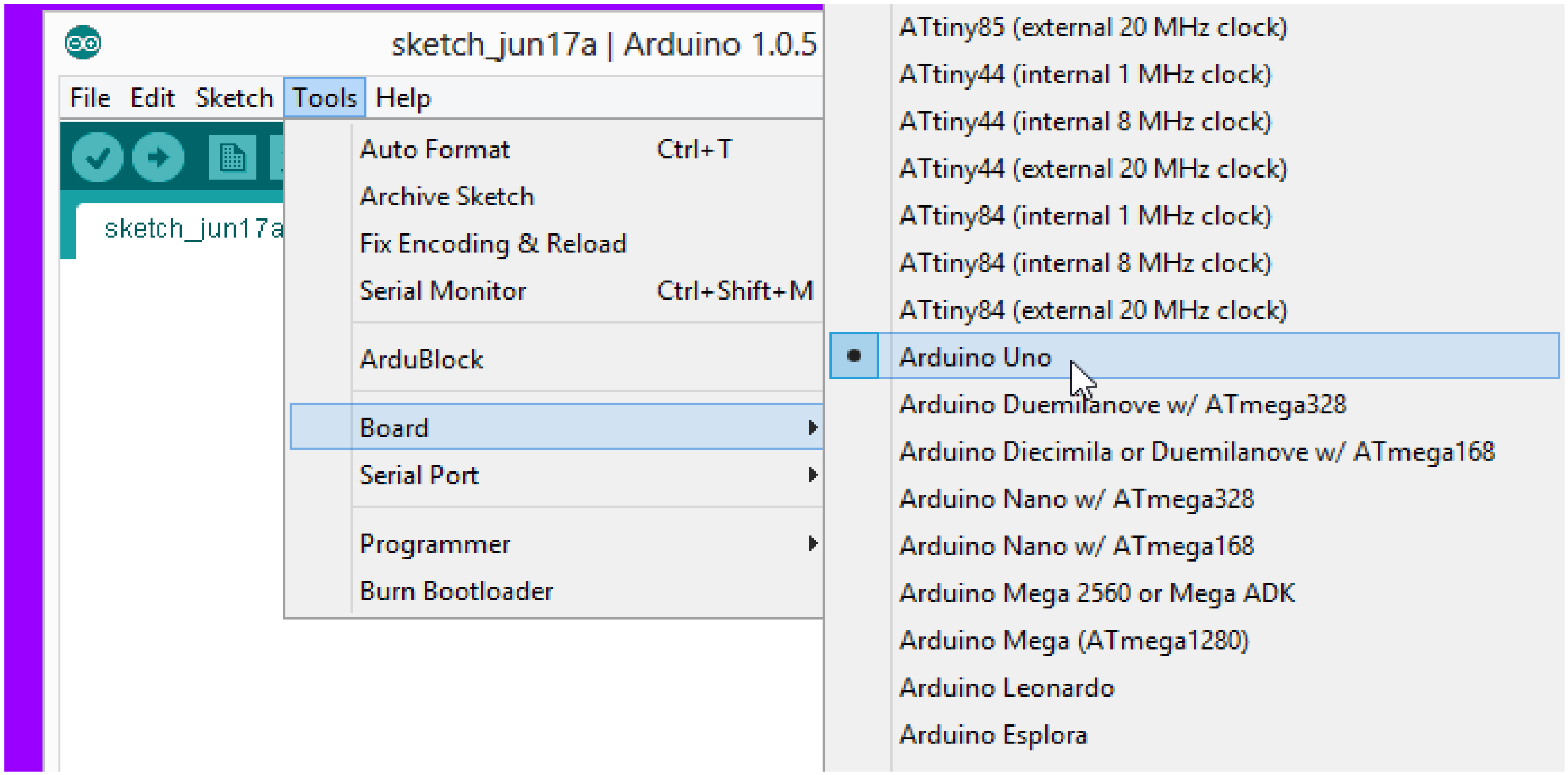


Settings: Tools → Serial Port

Your computer communicates to the Arduino microcontroller via a serial port → through a USB-Serial adapter.

Check to make sure that the drivers are properly installed.

Arduino IDE (Cont..)


sketch_jun17a | Arduino 1.0.5

File Edit Sketch Tools Help

sketch_jun17a

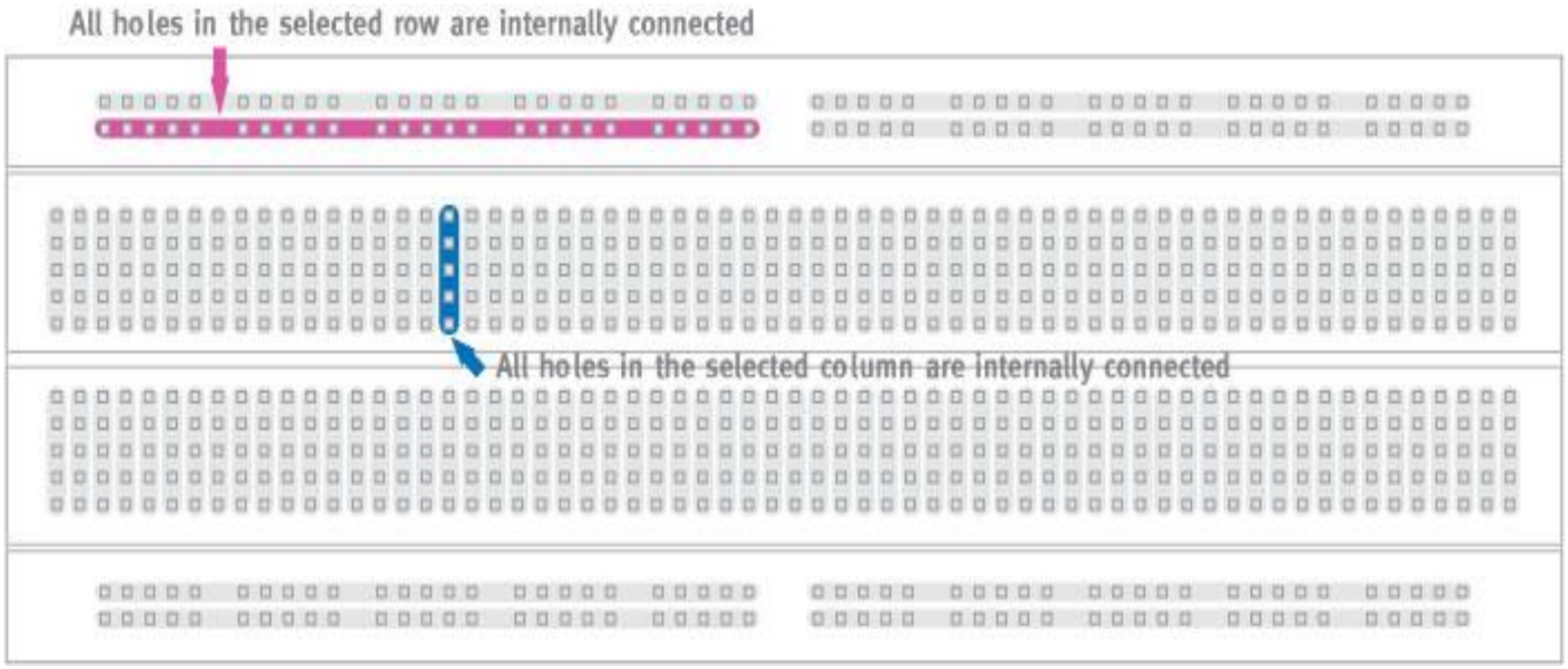
Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Serial Monitor	Ctrl+Shift+M
ArduBlock	
Board	▸
Serial Port	▸
Programmer	▸
Burn Bootloader	

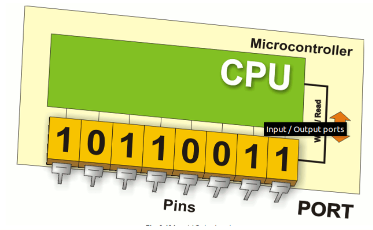
- ATtiny85 (external 20 MHz clock)
- ATtiny44 (internal 1 MHz clock)
- ATtiny44 (internal 8 MHz clock)
- ATtiny44 (external 20 MHz clock)
- ATtiny84 (internal 1 MHz clock)
- ATtiny84 (internal 8 MHz clock)
- ATtiny84 (external 20 MHz clock)
- ● Arduino Uno
- Arduino Duemilanove w/ ATmega328
- Arduino Diecimila or Duemilanove w/ ATmega168
- Arduino Nano w/ ATmega328
- Arduino Nano w/ ATmega168
- Arduino Mega 2560 or Mega ADK
- Arduino Mega (ATmega1280)
- Arduino Leonardo
- Arduino Esplora

Settings: Tools → Board

Next, double-check that the proper board is selected under the Tools→Board menu.

Breadboard Layout:


All holes in the selected row are internally connected
All holes in the selected column are internally connected



Arduino Digital I/0

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

Sets pin to either `INPUT` or `OUTPUT`

```
Eg1. pinMode(13, OUTPUT);
```

`digitalRead(`*`pin`*`);`

Reads `HIGH` or `LOW` from a pin

```
Eg3. digitalRead(2);
```

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

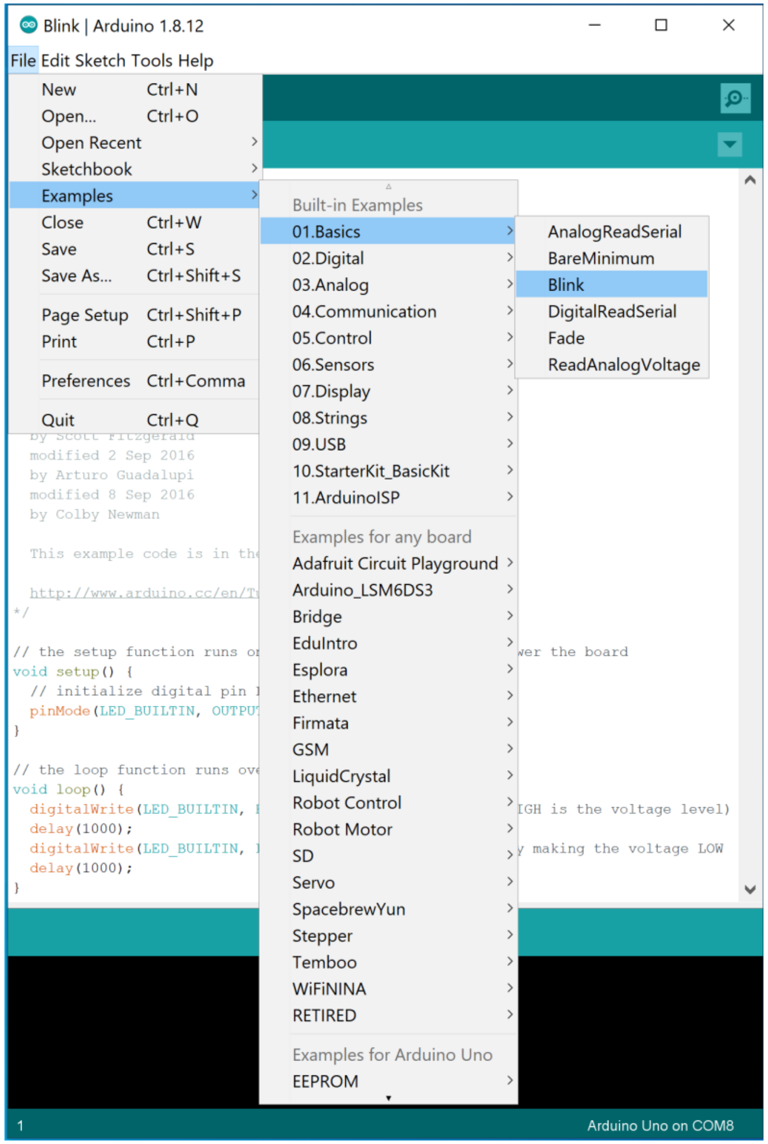
Writes `HIGH` or `LOW` to a pin

```
Eg2. digitalWrite(13, HIGH);
```

Our first Arduino Sketch/Program

```
/*
 *  Arduinos ketch to toggle the LED connected to pin-13 with a rate/delay of 1sec
 */
void setup()
{
  // put your setup code here, to run once:                      -->I*
  pinMode(13, OUTPUT);  //pin-13 configures as o/p               -->II
}
void loop()
{
  // put your main code here, to run repeatedly:                                -->1*
  digitalWrite(13, HIGH);       //HIGH Value or Bunary-1 send to pin-13     -->2
  //delay(x);                   //x-ms second(s) delay                          -->3*
  //delayMicroseconds(y);       //y-us second(s) delay                          -->4*
  delay(1000);                  //1000-milliseconds=1second delay      -->5
  digitalWrite(13, LOW);        //LOW Value or Bunary-1 send to pin-13      -->6
  delay(1000);                  //1000-milliseconds=1second delay      -->7
  //Toggling rate of led connected to pin-13 is of 1second                      -->8*
}
```

Uploading and Running the blink sketch



```
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the

  http://www.arduino.cc/en/T
*/

// the setup function runs o                          wer the board
void setup() {
  // initialize digital pin l
  pinMode(LED_BUILTIN, OUTPU
}

// the loop function runs ove
void loop() {
  digitalWrite(LED_BUILTIN, H                         IGH is the voltage level)
  delay(1000);
  digitalWrite(LED_BUILTIN, L                         y making the voltage LOW
  delay(1000);
}
```

In Arduino, open up:

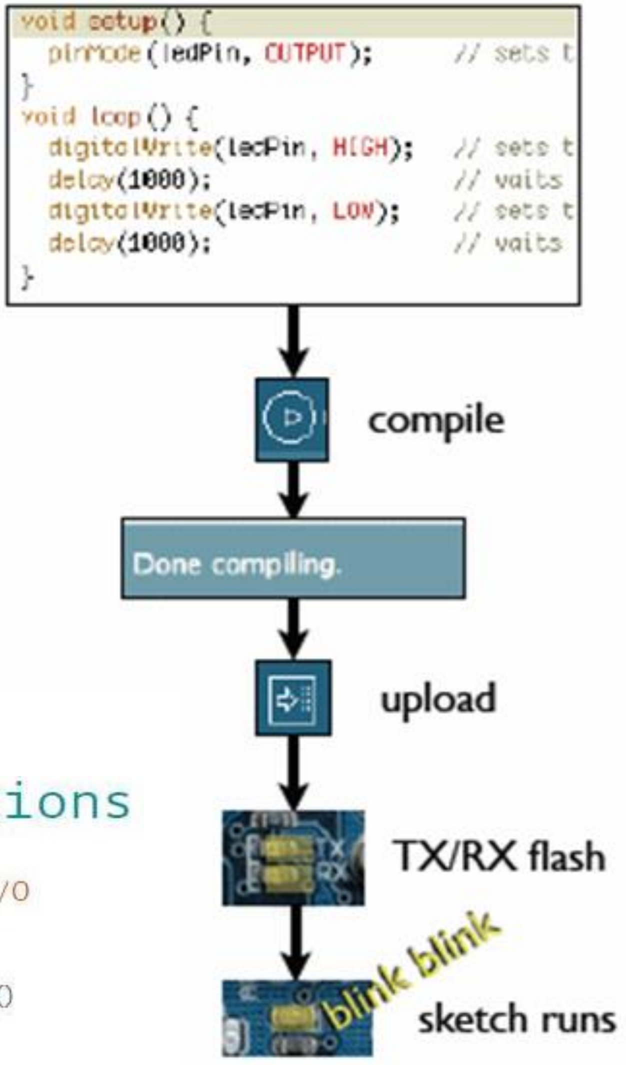
File → Examples → 01.Basics → Blink

```
const int ledPin = 13; // LED connected
to digital pin 13
void setup()
{
pinMode(ledPin, OUTPUT);
}
void loop()
{
digitalWrite(ledPin, HIGH); // set the LED
on
delay(2000); // wait for two seconds
digitalWrite(ledPin, LOW); // set the LED
off
delay(2000); // wait for two seconds
}
```

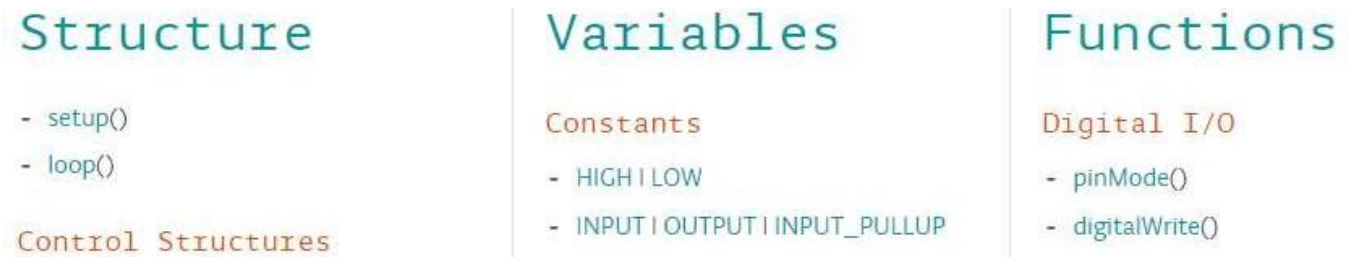
Uploading and Running the blink sketch (Cont..)

- Write your sketch/program
- Press compile button
  (to check for errors)
- Press upload button to program
  Arduino board with your sketch
- Try it out with the "Blink" sketch!

Arduino programs can be divided in three main parts: Values(Variables & Constants), Structure, and functions.

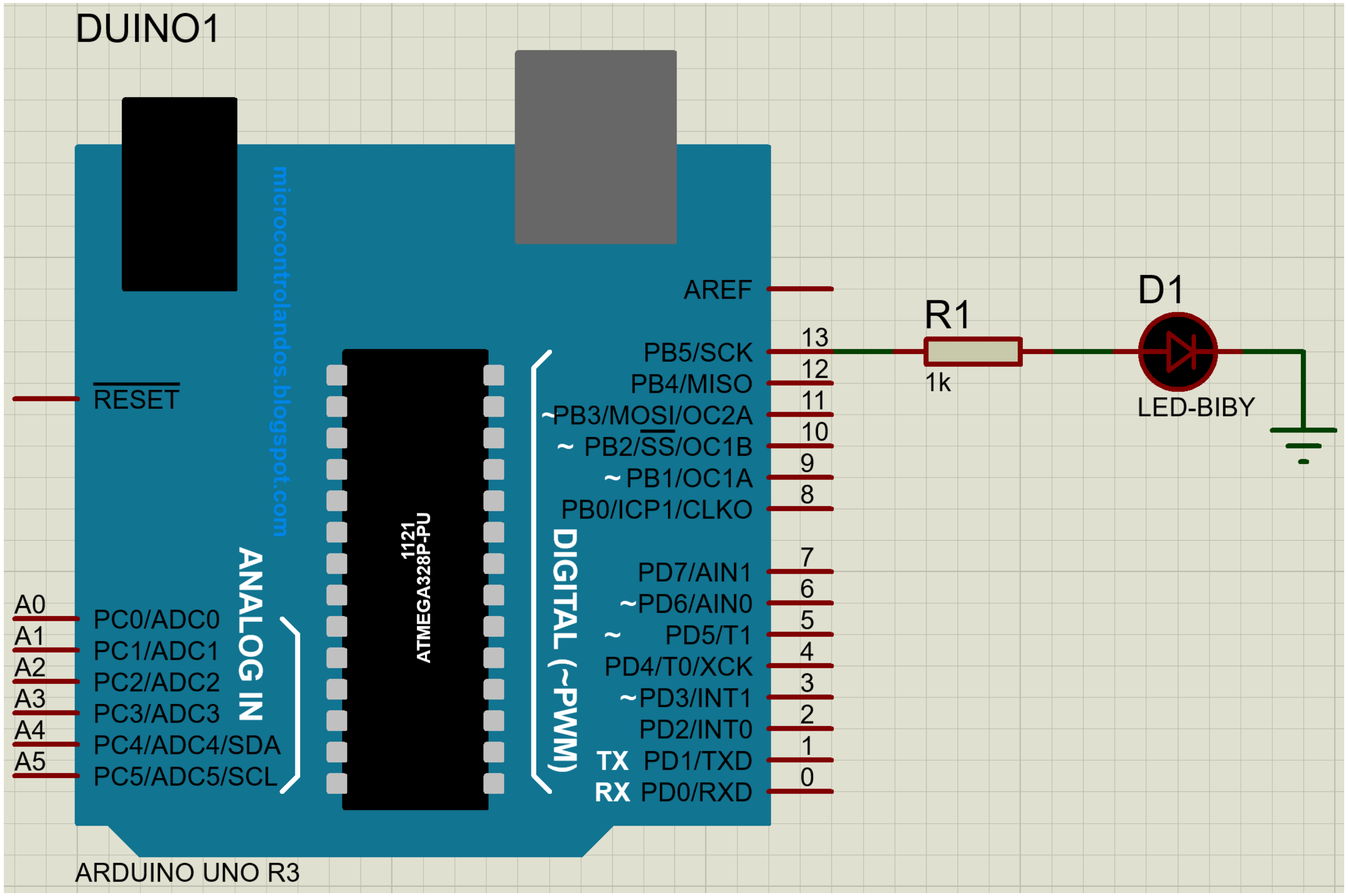


```
void setup() {
  pinMode(ledPin, OUTPUT);          // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```



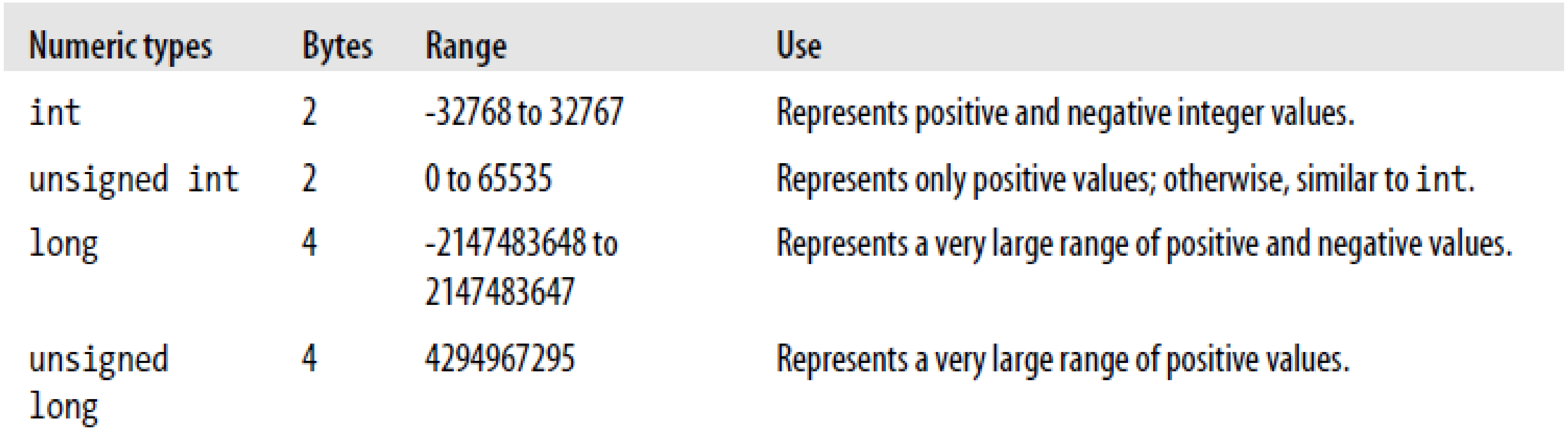
Structure	Variables	Functions
- setup()	Constants	Digital I/O
- loop()	- HIGH I LOW	- pinMode()
Control Structures	- INPUT I OUTPUT I INPUT_PULLUP	- digitalWrite()

Proteus design



Arduino Data Types

- Except in situations where maximum performance or memory efficiency is required, variables declared using int will be suitable for numeric values if the values do not exceed the range (shown in the first row in below Table) and if you don’t need to work with fractional values.
- Most of the official Arduino example code declares numeric variables as int.
- But sometimes you do need to choose a type that specifically suits your application.

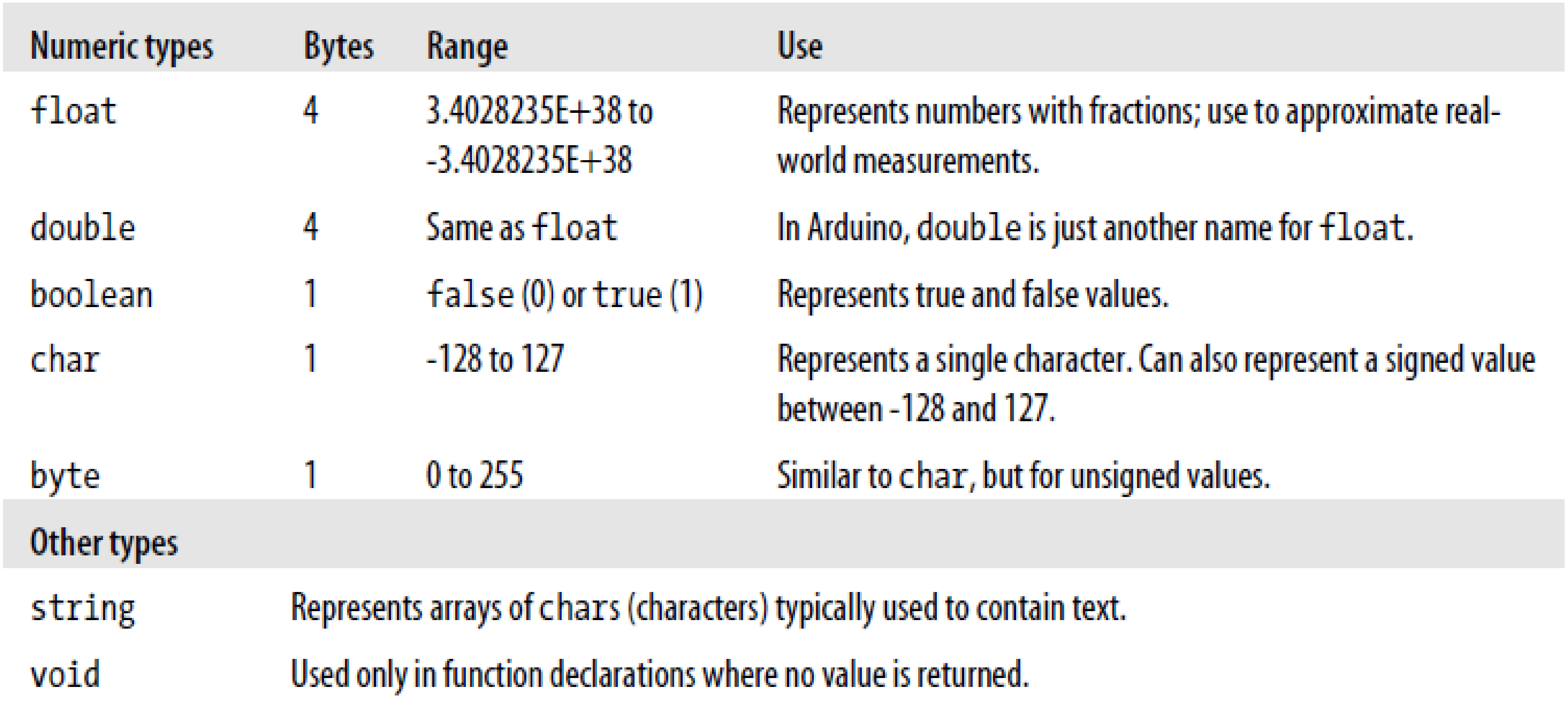


Numeric types	Bytes	Range	Use
`int`	2	-32768 to 32767	Represents positive and negative integer values.
`unsigned int`	2	0 to 65535	Represents only positive values; otherwise, similar to `int`.
`long`	4	-2147483648 to 2147483647	Represents a very large range of positive and negative values.
`unsigned long`	4	4294967295	Represents a very large range of positive values.

Arduino Data Types (Cont..)

- Sometimes you need negative numbers and sometimes you don’t, so numeric types come in two varieties: signed and unsigned.
- unsigned values are always positive. Variables without the keyword unsigned in front are signed so that they can represent negative and positive values.
- One reason to use unsigned values is when the range of signed values will not fit the range of the variable (an unsigned variable has twice the capacity of a signed variable).
- Another reason programmers choose to use unsigned types is to clearly indicate to people reading the code that the value expected will never be a negative number.

Arduino Data Types (Cont..)



Numeric types	Bytes	Range	Use
float	4	3.4028235E+38 to -3.4028235E+38	Represents numbers with fractions; use to approximate real-world measurements.
double	4	Same as float	In Arduino, double is just another name for float.
boolean	1	false (0) or true (1)	Represents true and false values.
char	1	-128 to 127	Represents a single character. Can also represent a signed value between -128 and 127.
byte	1	0 to 255	Similar to char, but for unsigned values.
Other types			
string	Represents arrays of chars (characters) typically used to contain text.		
void	Used only in function declarations where no value is returned.		

Arduino Data Types (Cont..)

- Boolean types have two possible values: true or false. They are commonly used for things like checking the state of a switch (if it’s pressed or not).

- You can also use HIGH and LOW as equivalents to true and false where this makes more sense;
  - digitalWrite(pin, HIGH) is a more expressive way to turn on an LED than digitalWrite(pin, true) or digitalWrite(pin,1), although all of these are treated identically when the sketch actually runs.

Data Types and Operators

Integer: used with integer variables with value between 2147483647 and -2147483647.

Eg: int x=1200;

Character: used with single character, represent value from - 127 to 128.

Eg. char c='r';

Long: Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from -2,147,483,648 to 2,147,483,647.

Eg. long u=199203;

Floating-point numbers can be as large as 3.4028235E+38 and as low as -3.4028235E+38. They are stored as 32 bits (4 bytes) of information.

Eg. float num=1.291; [The same as **double** type]

Statements and operators:

Statement represents a command, it ends with **;**

Eg: **int x; x=13;**

Operators are symbols that used to indicate a specific function:

Math operators: [**+,-,*,/,%,^**]

Logic operators: [**==, !=, &&, ||**]

Comparison/Boolean operators: [**==, >, <, !=, <=, >=**]

Syntax:

; Semicolon, **{}** curly braces,

// single line comment,

/* *Multi-line comments* ****/***

Boolean Operators:

== (is equal?)
!= (is not equal?)
> (greater than)
>= (greater than or equal)
< (less than)
<= (less than or equal)

Compound Operators:

++ (increment)
-- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)

Control statements:

If Conditioning:

```
if(condition)
{
statements-1;
…
Statement-N;
}
else if(condition2)
{
Statements;
}
else
{
statements;
}
```

Switch case:

```
switch (var)
{
        case 1:
//do something when var equals 1
break;
case 2:
//do something when var equals 2
break;
default:
// if nothing else matches, do the default
// default is optional
}
```

Loop statements:

Do….while:

```
do
{
Statements;
}
  while(condition)    // the statements are run at least once.
```

While:

```
  While(condition
  )
  { statements; }
```

```
for (int i=0; i <= val; i++)
{
statements;
}
```

Use *break* statement to stop the loop whenever needed.

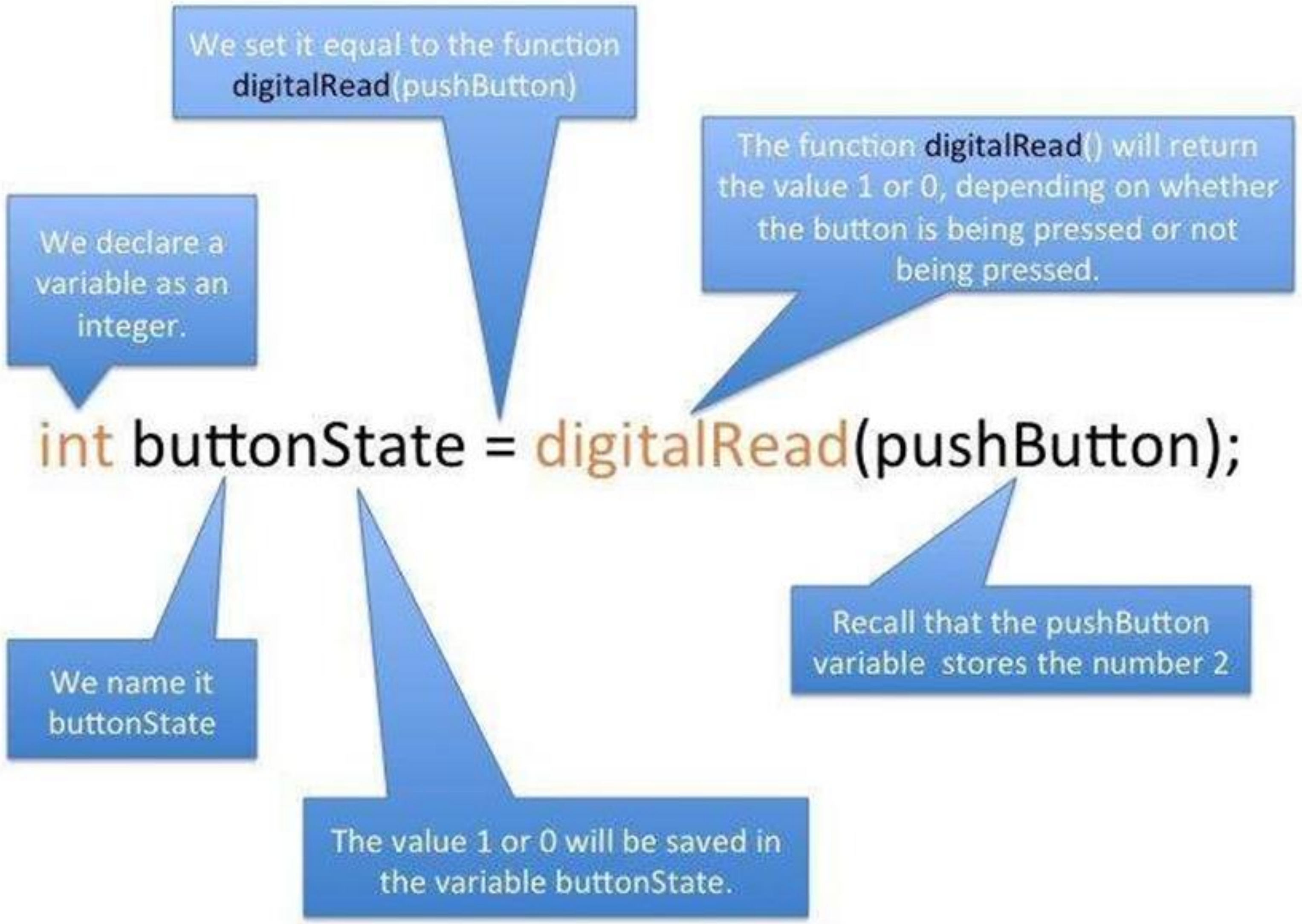
Digital Input

- Connect digital input to your Arduino using Pins # 0 – 13 (Although pins # 0 & 1 are also used for programming)
- Digital Input needs a `pinMode` command: **`pinMode (pinNumber, INPUT);`** *Make sure to use ALL CAPS for **INPUT***
- To get a digital reading:

```
int buttonState = digitalRead (pinNumber);
```

- Digital Input values are only **HIGH** (On) or **LOW** (Off)

Digital Input (Cont..)



We set it equal to the function digitalRead(pushButton)

The function digitalRead() will return the value 1 or 0, depending on whether the button is being pressed or not being pressed.

We declare a variable as an integer.

```
int buttonState = digitalRead(pushButton);
```

We name it buttonState

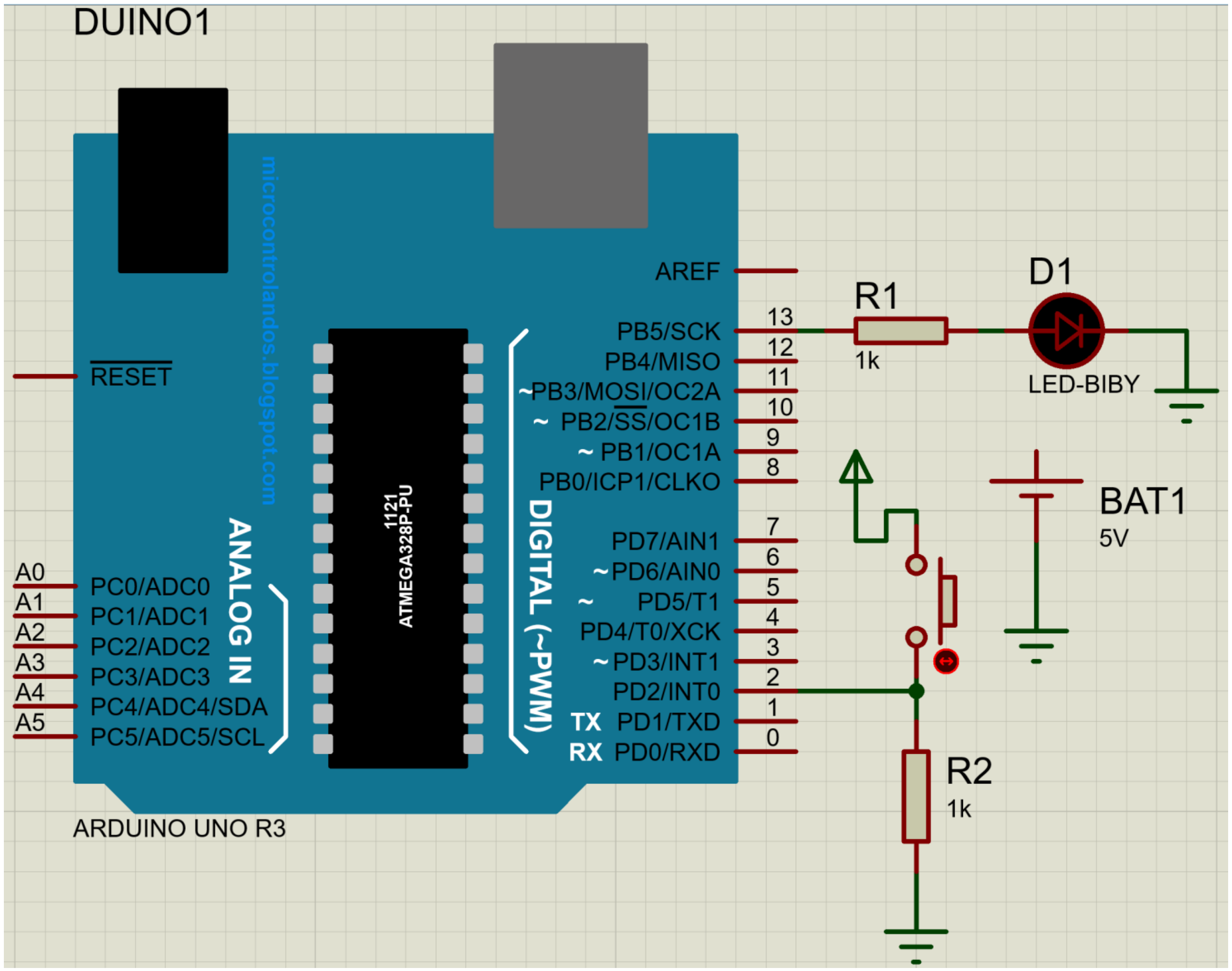
The value 1 or 0 will be saved in the variable buttonState.

Recall that the pushButton variable stores the number 2

// Pushbutton sketch: a switch is connected to pin 2 lights the LED on pin 13 // (or) Control an LED connected to pin-13 w.r.to. a switch connected to pin-2, with the help of external resister

```
const int ledPin = 13; // choose the pin for the LED
const int inputPin = 2; // choose the input pin (for a pushbutton)
void setup()
{
pinMode(ledPin, OUTPUT); // declare LED as output
pinMode(inputPin, INPUT); // declare pushbutton as input
}
void loop()
{
int val = digitalRead(inputPin); // read input value
        if (val == HIGH) // check if the input is HIGH
        {
        digitalWrite(ledPin, HIGH); // turn LED on if switch is pressed
        }
        else
        {
        digitalWrite(ledPin, LOW); // turn LED off
        }
}
```

Proteus design (with external resister)



Using a key/push button with external resistor

- The digitalRead function monitors the voltage on the input pin (inputPin), and it returns a value of HIGH if the voltage is high (5 volts) and LOW if the voltage is low (0 volts).

- Actually, any voltage that is greater than 2.5 volts (half of the voltage powering the chip) is considered HIGH and less than this is treated as LOW.
- If the pin is left unconnected (known as *floating*) the value returned from digitalRead is indeterminate (it may be HIGH or LOW, and it cannot be reliably used).
- The resistor shown in Figure shown in previous slide ensures that the voltage on the pin will be low when the switch is not pressed, because the resistor “pulls down” the voltage to ground.
- When the switch is pushed, a connection is made between the pin and +5 volts, so the value on the pin interpreted by digital Read changes from LOW to HIGH.

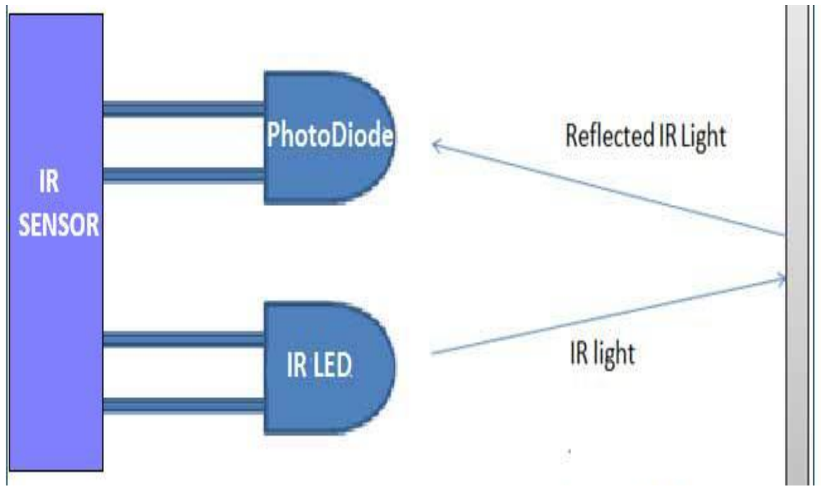
InfraRed (IR) Sensor

- IR sensor consists of two parts, emitter circuit and receiver circuit. This is collectively known as a photo-coupler or an opto coupler.

- The emitter is IR LED and detector is IR photodiode.
- The IR photodiode is sensitive to the IR light emitted by an IR LED.

Principle:

The photodiode’s resistance and output voltage change in proportion to the IR light received. This is the working principle of the IR sensor.



There are several types of IR sensors, each with different characteristics and applications. Some common types include:

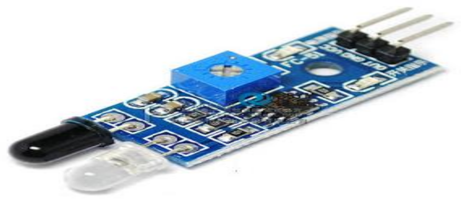
Passive Infrared (PIR) Sensor: PIR sensors are used to detect motion by sensing changes in infrared radiation. They are commonly used in security systems, lighting control, and automatic doors.

IR proximity Sensor: IR proximity sensors are used to detect the presence of an object without making physical contact. They are commonly used in mobile devices, robotics, and automation systems.

IR Imaging sensor: These sensors use infrared radiation to create images, they are used in thermal imaging, night vision, and surveillance cameras.

IR temperature Sensor: These can measure the temperature of an object by detecting the infrared radiation emitted by it. They are used in industrial, HVAC, and medical applications.

InfraRed (IR) Sensor



Features-

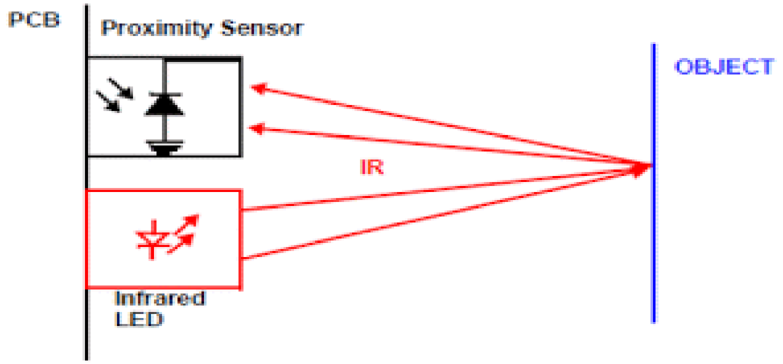
- Can be used for obstacle sensing, fire detection, line sensing, etc
- Input Voltage: 5V DC
- Comes with an easy to use digital output
- Can be used for wireless communication and sensing IR remote signals

IR Sensor have three Pins

1. VCC = +5V DC

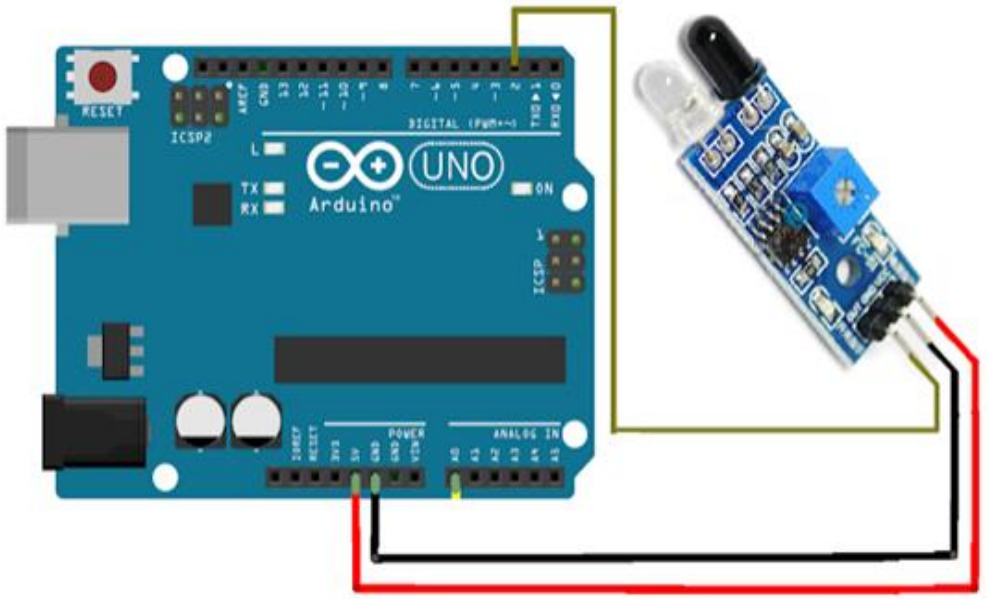
2. GND

3. D0 or OUT (Digital Output)



Arduino sketch & Circuit diagram

```
const int ProxSensor=2;
void setup()
{
pinMode(13, OUTPUT);
pinMode(ProxSensor, INPUT);
}
void loop()
{
if(digitalRead(ProxSensor)==HIGH)
            {
            digitalWrite(13, HIGH);
            }
else
            {
            digitalWrite(13, LOW);
            }
      delay(100);
}
```

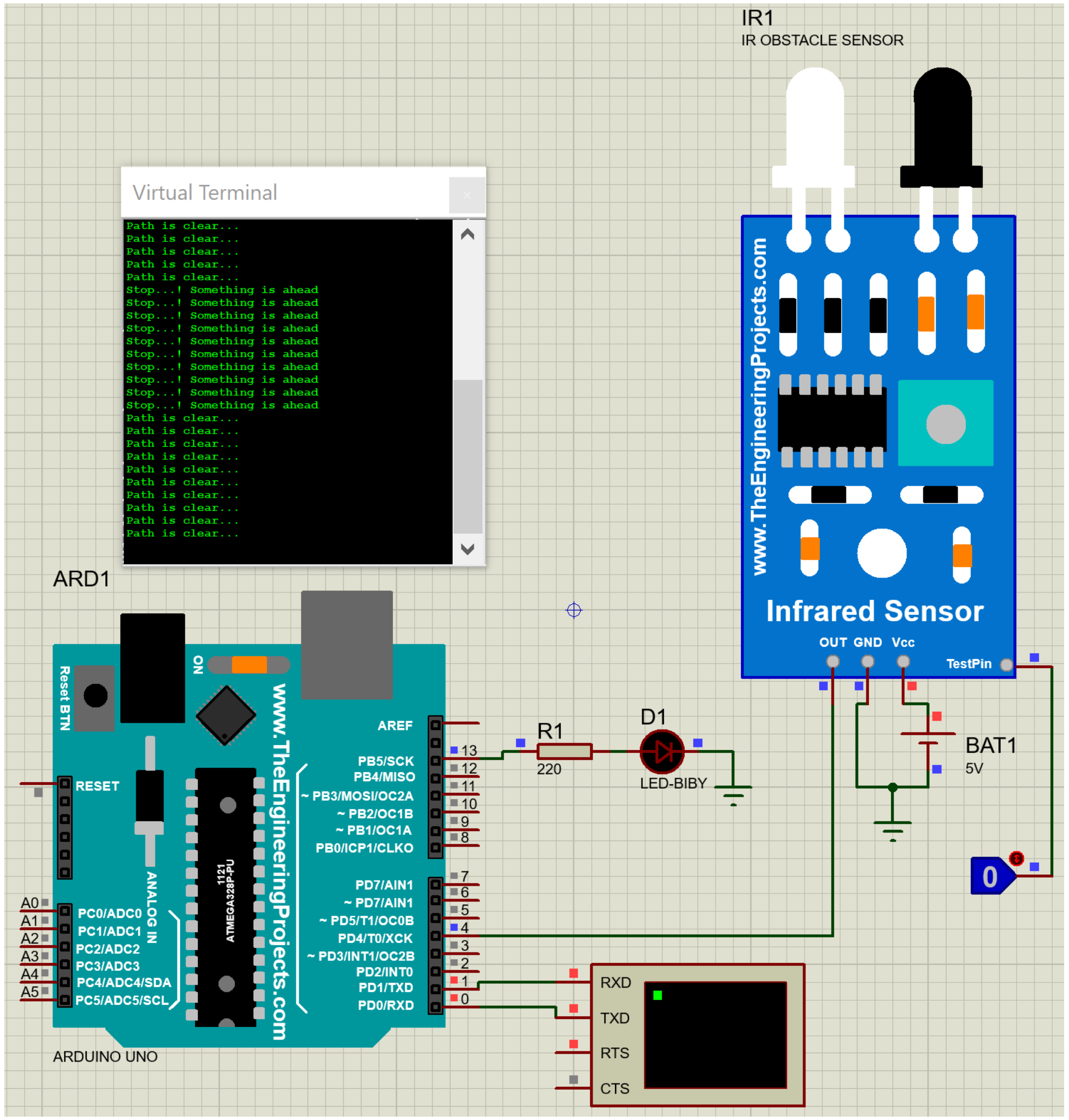


IR Sensor Interfacing, display the o/p on Serial Monitor

```
/*
IR Proximity Sensor interface code
Turns on an LED on when obstacle is
detected, else off.
*/

const int ProxSensor=2;

void setup()
{
// initialize the digital Serial port.

Serial.begin(9600);
// initialize the digital pin as an output.
pinMode(13, OUTPUT);
pinMode(ProxSensor,INPUT);
}
```

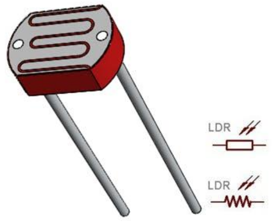
```
void loop()
{
  if(digitalRead(ProxSensor)==LOW)
//Check the sensor output
  {
  digitalWrite(13, HIGH);   // set the LED on
  Serial.println("Stop something is ahead!! ");
//Message on Serial Monitor
  }
  else
  {
  digitalWrite(13, LOW);    // set the LED off
  Serial.println("Path is clear");
//Message on Serial Monitor
  }
delay(1000);              // wait for a second
}
```

Proteus design:



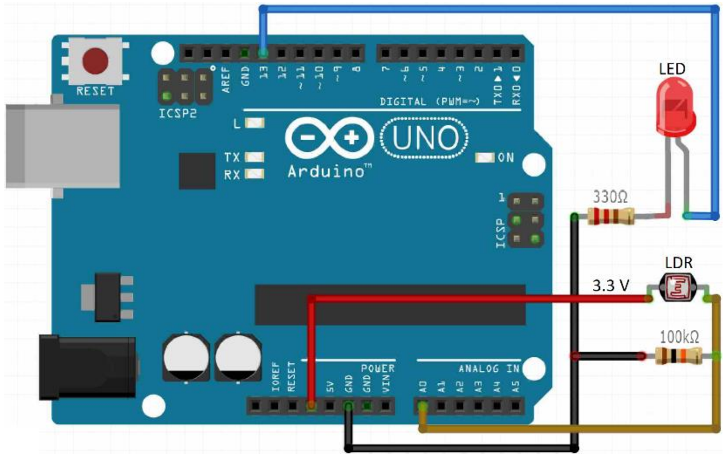
```
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Stop...! Something is ahead
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
Path is clear...
```

Light Dependent Resistor



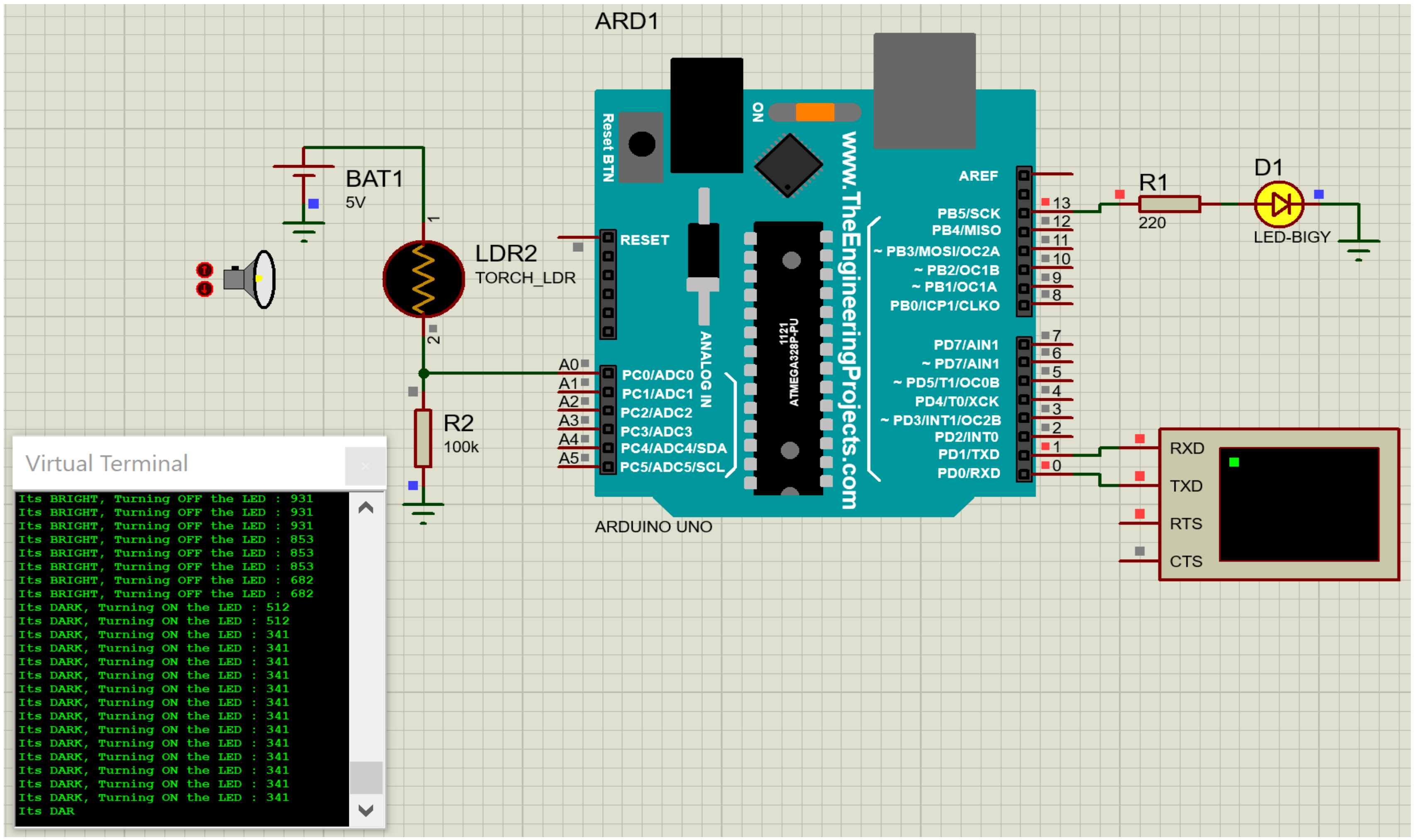
- LDR (light dependent resistor) also called as photoresistor is responsive to light.
- Photoresistors are used to indicate the light intensity (or) the presence or absence of light.
- When there is darkness then the resistance of photoresistor increases, and when there is sufficient light it’s resistance decreases.

LDR with Arduino: sketch & Circuit



```
const int ledPin = 13; //pin at which LED is connected
const int ldrPin = A0; //pin at which LDR is connected
int threshold = 600;
void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT); //Make LED pin as output
  pinMode(ldrPin, INPUT); //Make LDR pin as input
}
void loop()
{
  int ldrStatus = analogRead(ldrPin); //saving the analog values received from LDR
  if (ldrStatus <= threshold) //set the threshold value below at which the LED will turn on
  {                //you can check in the serial monior to get approprite value for your LDR
    digitalWrite(ledPin, HIGH);  //Turing LED ON
    Serial.print("Its DARK, Turn on the LED : ");
    Serial.println(ldrStatus);
  }
  else
  {
    digitalWrite(ledPin, LOW); //Turing OFF the LED
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);
  }
}
```

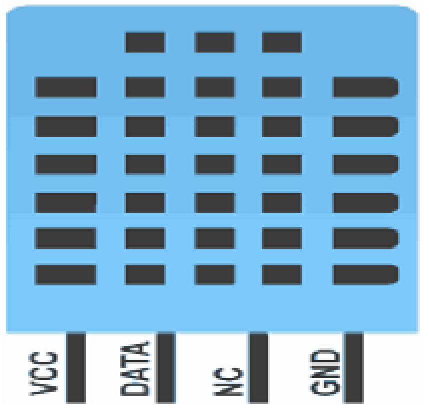
Proteus design:



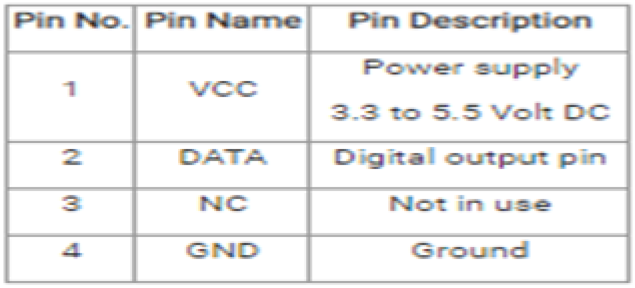
```
Its BRIGHT, Turning OFF the LED : 931
Its BRIGHT, Turning OFF the LED : 931
Its BRIGHT, Turning OFF the LED : 931
Its BRIGHT, Turning OFF the LED : 853
Its BRIGHT, Turning OFF the LED : 853
Its BRIGHT, Turning OFF the LED : 853
Its BRIGHT, Turning OFF the LED : 682
Its BRIGHT, Turning OFF the LED : 682
Its DARK, Turning ON the LED : 512
Its DARK, Turning ON the LED : 512
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DARK, Turning ON the LED : 341
Its DAR
```

DHT11 Sensor Interfacing with Arduino UNO:

- DHT11 sensor measures and provides humidity and temperature values serially over a single wire.
- It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50˚C.
- It has 4 pins; one of which is used for data communication in serial form.
- DHT11 sensor uses resistive humidity measurement component, and NTC temperature measurement component.

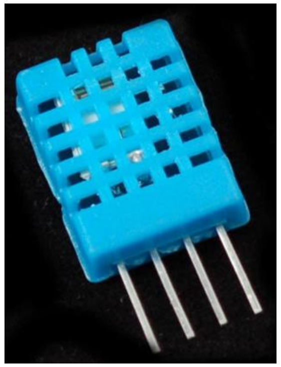


Pin diagram of DHT11



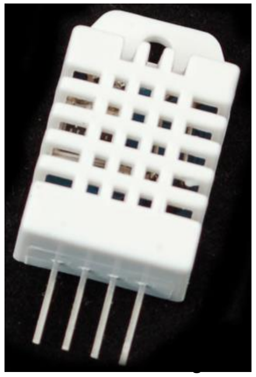
Pin No.	Pin Name	Pin Description
1	VCC	Power supply 3.3 to 5.5 Volt DC
2	DATA	Digital output pin
3	NC	Not in use
4	GND	Ground

DHT11 Specifications



- Ultra-low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 20-80% humidity readings with 5% accuracy
- Good for 0-50˚C temperature readings $\pm$2˚C accuracy

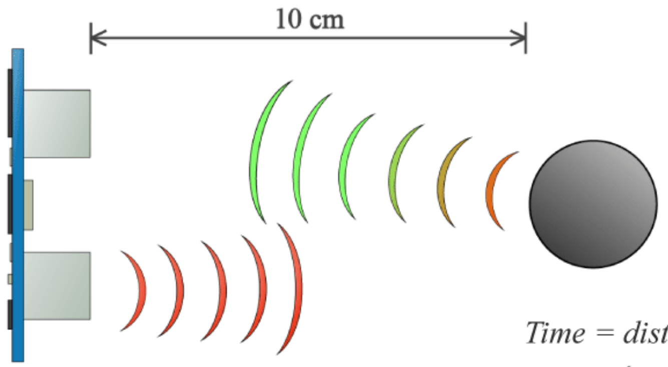
DHT22 Specifications



- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy
- Good for -40 to 125˚C temperature readings ±0.5˚C accuracy

Ultrasonic Sensor

- An Ultrasonic sensor is a device that can measure the distance to an object by using sound waves.
- It measures distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back.
- By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, it is possible to calculate the distance between the sonar sensor and the object.



speed of sound:

$v = 340\ m/s$

$v = 0{,}034\ cm/\mu s$

Time = distance / speed:

$t = s / v = 10 / 0{,}034 = 294\ \mu s$

Distance:

$s = t \cdot 0{,}034 / 2$

$$distance = \frac{speed\ of\ sound \times time\ taken}{2}$$

Ultrasonic Sensor (Cont..)

- Since it is known that sound travels through air at about 344 m/s (1129 ft/s), you can take the time for the sound wave to return and multiply it by 344 meters (or 1129 feet) to find the total round-trip distance of the sound wave.

- Round-trip means that the sound wave traveled 2 times the distance to the object before it was detected by the sensor; it includes the 'trip' from the sonar sensor to the object AND the 'trip' from the object to the Ultrasonic sensor (after the sound wave bounced off the object).

- To find the distance to the object, simply divide the round-trip distance in half.

$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

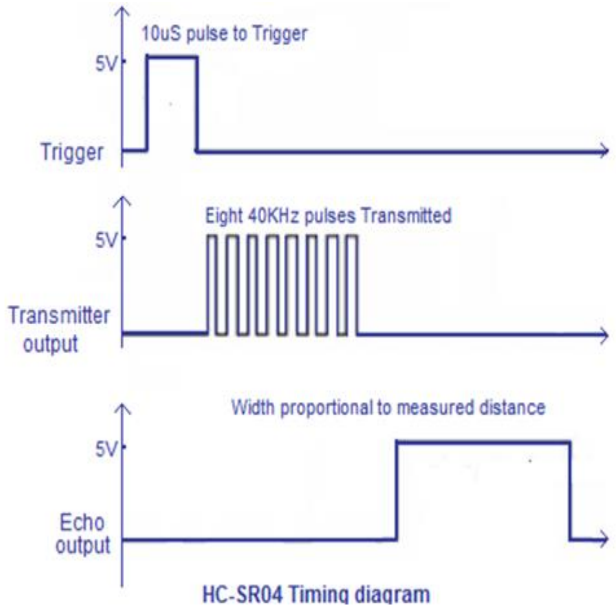
Working principle

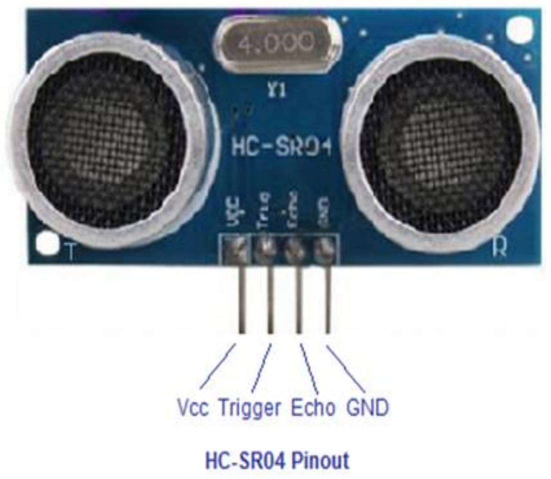
The Trig pin will be used to send the signal and
the Echo pin will be used to listen for returning signal

(1) Using IO trigger for at least 10us high level signal, Trig -> Pin-9 (o/p) of Arduino

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.





$$distance = \frac{speed\ of\ sound\ \times time\ taken}{2}$$

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Links to download the software's:

Link to download Proteus 8.1:

- https://getintopc.com/softwares/3d-cad/proteus-professional-2020-free-download/

Link to download Proteus Libraries:

- https://github.com/officialdanielamani/proteus-library-collection

Link to download Arduino IDE 1.8.19:

- https://www.arduino.cc/en/software

WOKWI simulations

Following are the links of WOKWI simulations:

1. Simulation link for LED TOGGLING experiment:
https://wokwi.com/projects/378978156175779841

2. Simulation link for LED control using SWITCH/PUSH BUTTON experiment:
https://wokwi.com/projects/378978765317787649

3. Simulation link for LDR (Light Dependent Resistor) experiment:
https://wokwi.com/projects/378755041857942529

4. Simulation link for Ultrasonic Sensor (HCSR_04) experiment for distance measurement:
https://wokwi.com/projects/378752418231320577

Link to download Proteus 8.13:

https://ettron.com/how-to-download-and-install-proteus-8-13-latest-version/