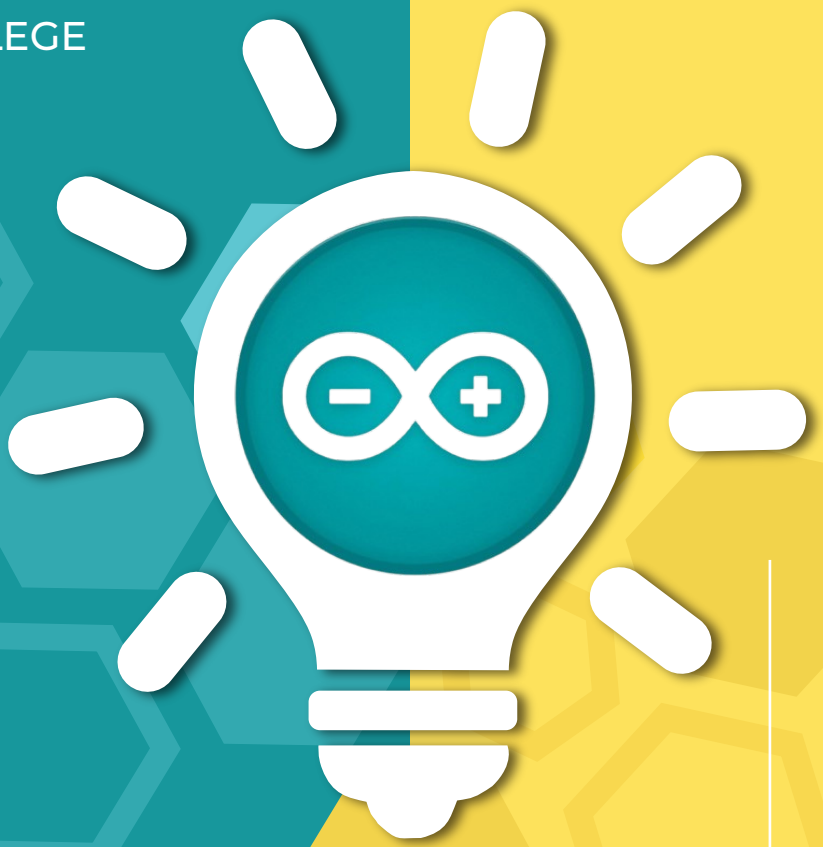
Follows TIY
(Teach it Yourself)
Approach

COMPREHENSIVE GUIDE TO ARDUINO

SECONDARY SCHOOL & COLLEGE



In Line with CBSE & in Tune with
Child's Aspirations
& their Technological Future

CHAPTER

01 Introduction to Arduino

1 Lesson Overview

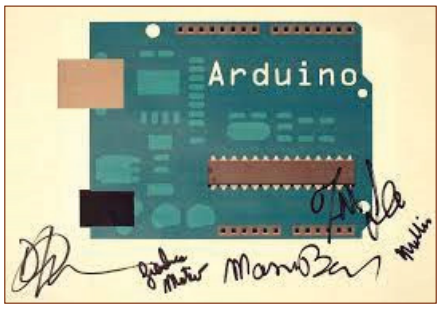
By the end of this lesson students will have a good idea of what is Arduino, why it was developed, difference in the approach adopted for its development to that of a computer, what you can do with it, its open-source advantages, ease of its learning, & its hardware range.

2 Necessity is the Mother of Invention

Nothing can prove this more than the development of Arduino. In 2006, the founding members of Arduino found a **pressing market need for a device for automation. It had to be simple, easy to connect to various things** (such as relays, motors, and sensors), and easy to program. Above all, **it needed to be inexpensive** for mass utilisation. They could not find one, so they decided to **try make one themselves.**

The development started in a small house at Interaction Design Institute, Ivrea, Italy. Its goal was to create an **affordable & Straight forward tools for use by non engineers to work on digital projects.** The core Arduino team consisted of Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino & David Mellis.

Platform chosen for development of Arduino was the ATMega168 microcontroller. This was part of AVR family of Atmel and constituted its brain. It used an IDE based on Processing. Later the development was shifted to a cheaper ATMega8 controller. Attached is the image of the first board signed by the core team members.



The project was named **Arduino** for two reasons.
Firstly, Arduino means best friends.
Secondly, it was also the name of a hang out bar in which these best friends met to develp the chipset.

3 Approach to Development of Arduino Chip Set

Arduino has all the key ingrediants that a Computer board has. However, there exists a fundamental difference in the approach adopted in development of a computer chip set, & to that adopted in development of Arduino. In layman's words, computer is designed for computing. Its main component is a chip set called CPU. To perform the desired functions, it works using several external input & output devices, operating System like windows, & an application SW like power point. In simple words, **it's Hardware & SW are separate entities.**

On the other hand, Arduino is designed for i**nteraction with surroundings using sensors & actuators.** While a computer could have well done this job, it would have been an overkill for the proposed applications (automation). The approach in Arduino revolved around selecting the basic controller, designing a **self-contained circuit board** with easy-to-use connections around it, writing the **bootloader firmware** for the microcontroller, and **packaging** it all into a simple integrated development environment **(IDE)** that used programs called **"sketches".** In totally non-technical words, we can say that **Arduino chip set is a stripped-down computer with its own IDE or SW.** In simple words, **it's hardware & SW are a combined entity.**

Since then, **Arduino has grown** in several different directions, with some versions getting smaller than the original, and some getting larger. Each has a **specific intended niche** to fill. A typical **ex of niche is the washable Arduino chip for wearables.** Common element among all of them is the Arduino **runtime AVR-GCC library** that is supplied with the Arduino IDE, and the on-board bootloader firmware that comes preloaded on the microcontroller of every Arduino board.

4 What You Can do with Arduino

- **Small-scale automation**
  - Automated greenhouse
  - Automated aquarium
  - Laboratory sample shuttle robot
  - Home & workplace automation
  - Automated electronic test system
- **Small-scale control**
  - Small robots
  - Model rockets
  - Model aircraft
  - Quadrotor UAVs
  - Simple CNC for machine tools
- **Real-world monitoring**
  - Automated weather station
  - Automated responses
  - Sun tracking for solar panels
  - Environment monitoring
  - Automatic wildlife detector
  - Home or business security system
- **Performance art**
  - Dynamic lighting control
  - Dynamic sound control
  - Dynamic effects & illusions
  - Audience-responsive artwork

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

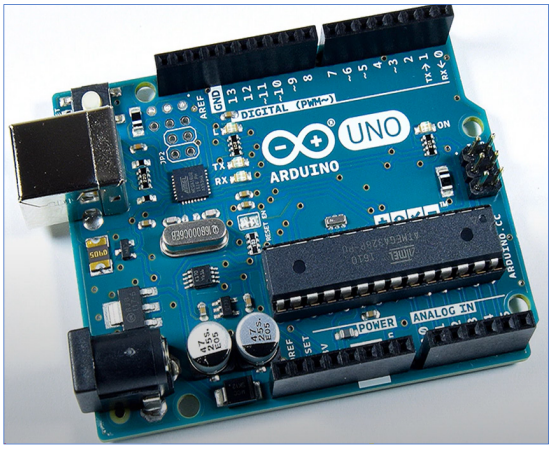
CHAPTER

02 Structure of Arduino UNO

1 Lesson Overview

By the end of this lesson, students will have a good idea of the layout & functions of the key segments & components on an Arduino Uno board. This understanding will form the basis of them understanding other Arduino boards that follow a similar layout structure

2 Salient Features of Arduino UNO Board



- It is an electronic **Prototyping** platform.
- It takes an input like a reading from a sensor or pressing of a button, & gives an electrical output like starting a motor, or sending data to the web.
- It is great for controlling LED's, motors, relays & more. It connects to hundreds of external sensors. It is a **Microcontroller** (MCU) or a mini computer built into a little IC. It can be programmed in C or Assembly languages. The program resides in its memory. UNO has a Atmega 328 MCU (8-bit AVR with 32 KB flash memory).
- Arduino programs are called **Sketches.** They are written in a mix of C & C++. Sketch is written ina free desk top app called IDE. It supports windows, apple, & UNIX. Once you have written your sketch, IDE will check it for mistakes, & then upload it in the Arduino. It connects to your PC via USB cable. It draws power from this, or through a DC adapter.

3 Closer look at the UNO Board Components

Corelate text & images here with main UNO board above.

a) **Atmega 328 MCU.** This is the brain of UNO. It is a male MCU.

b) **Two banks of GP I/O Pins.** Used to connect external components

- **Upper bank** supports Digital pins 0 to 13 & GND.
  - 1 & 2 are Transmit (Tx) & Receive (Rx) pins for **UART** communications. .
  - Pine 3, 5, 6, 9, 10 & 11 are the six **PWM** pins.

Note symbol below the nums. These are conventions that must be learnt.

- **Lower bank** supports, Analogue GP I/O pins A0 to A5. It has 3.3 & 5 V Power & gnd pins to power ext devices & circuits.

c) Rest Button.

d) DC Power Supply Control Circuit. This has three parts:

- **Barrel Connector port.** Power is supplied by a DC adapter.
- **Two Coupling Capacitors.** Located on right of power port. Provides spike suppression. Left for output & right for input.
- **Voltage Regulator.** Above the barrel connector.

e) 16 MHz Crystal Controller. It is a programable watch dog timer with an Internal oscillator. It controls all time related operations.

f) USB Control Circuit. It is located above voltage regulator. It has three parts:

- **USB Input Port.** Interface for all communications.

- **USB Controller.** Square chip to right of USB port. It is a 32-pin thin quad flat pack (TQFP-32 Atmega 328 P (programable) controller. It controls all communications.
- **ICSP Header.** Four pins above USB controller. Used for refreshing of USB firmware.

g) ICSP Pins for refreshing of ATmega firmware. On left top of analogue pins.

h) LED Circuit. It has three LED's:
Circuit mounted LED connected to pin 13. When pin voltage is high, it lights up. Goes down as voltage falls.
Tx, Rx LED. This is below above LED. They flash when data is being passed.

i) Power LED. Above ATmega ICSP header. Lights up when board is powered up.

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

03 Structure of Arduino UNO

1 Lesson Overview

By the end of this lesson, students will have a good idea of what are different types of Arduino boards & their key ingredients. They have been also provided with a link to help them decide the right board to use.

2 Learning of Arduino

Arduino marks a **shift from general purpose 8-bit board** philosophy, to a philosophy of **simple & cheap stripped-down boards,** designed for mass use to meet specific needs. These specific needs include automation, IoT applications, 3D printing, wearable, and embedded systems.

Over the years different types of Arduino boards have been used to build thousands of projects, from daily gadgets to compound scientific instruments. As students, it is not enough to know how to use these boards but learn how to tailor them to the needs of the project being attempted. **Being open source will help them do so.**

Experimenting with Arduino is assited by an **international community** of designers, programers, artists, hobbyists, & students who have come together. Their contribution has added an unbelievable amount of knowledge that can be of immense use to beginners and specialists alike. The working mantra is -**'If you have an idea, google it'.** Chances are that someone has worked on it & you will find the help you are looking for.

3 Key Ingredients of Arduino Bboards

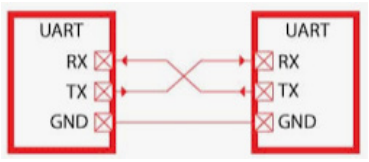
Every board has five key ingredients. These are:

1) **Microcontroller:** It is Arduino's "brain" handling all processing tasks and providing &) controlling access to input/output pins (I/O).

2) **Digital & Analog I/O Pins:** These are general-purpose digital input/output pins that read signals from sensors or buttons, while analogs enable connecting complex elements like distance sensors or motor controllers.

3) **Power Supply:** External power source like a battery or USB port is needed. Some models offer alternates, like solar or AC adapters.

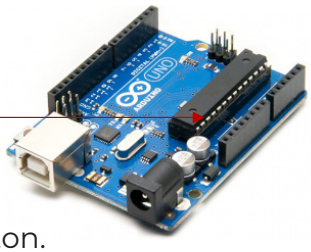
4) **USB Interface:** Serial communication protocol is UART over a mini-USB port. Note, UART (Universal Asynchronous Receive Transmit) is most used device-to-device communication protocol.



6) Clock Speed & Memory Capacity: Higher clock speeds result in faster performance. Larger memory capacities enable more complex projects. **These are important considerations for board selection.**

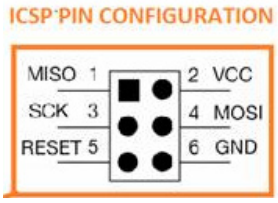
Note: UART and digital ports on Arduino enable it to communicate in both - **serial way and parallel way .**

4 Board 1 - Arduino Uno (R3)



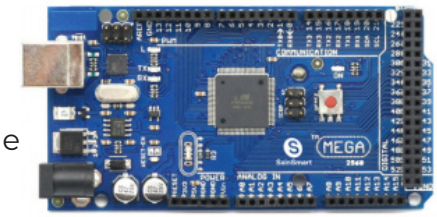
Using **ATmega328P microcontroller,** Uno is simplest & most used option for beginners. It is placed on a UNO board here. It consists of 14-digital I/O pins where: 6-pins can be used as PWM (pulse width mod generates variable-width pulses to represent amplitude of an analog input signal). 6-analog inputs. A reset button. A power jack. USB connection. In-Circuit Serial Programming header (ICSP) & more. R3 is **third revision** of original Arduino UNO. It is a great choice for DIY electronics & programming. For types of Atmega visit – **https://www.futurlec.com/ICAtmel_ATMega_Comparison.shtml**

Tip: Kindly start making your own vocabulory of abbreviations.

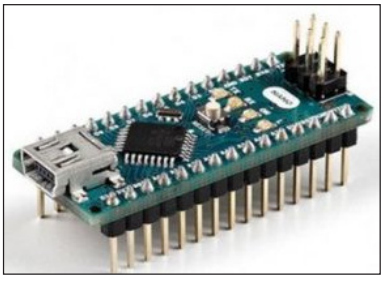


5 Board 2 - Arduino Mega R3

It is the big brother of UNO designed for solutions that require many I/O or peripherals. It boasts an impressive 54 I/O pins (15 of them can be used as PWM outputs), 16 analog inputs, & 4 UARTs. In addition It has more flash and SRAM. it is popular among open-source CNC makers and 3D printer builders. It can be easily implemented into various open-source Programmable Logic Controller projects. Huge number of pins make it a Choice for **uno projects that need more digital i/ps or o/ps.**



6 Board 3 - Arduino Nano



It is popular among developers due to its **small size & capabilities it shares with UNO.** It is a board of choice for **tight spaces projects.** Based on ATmega328P it includes analog pins-8, digital pins-14 power pins-6 & RST (reset) pins-2. Mini-USB & Arduino IDE are required to build projects. For details see **https://www.elprocus.com/an-overview-of-arduino-nano-board/**

It is a good site to ref for details of other boards as well.

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

04 Arduino Learning Tools

1 Lesson Overview

By the end of this lesson, students will have a good idea of the three main Arduino Boards, the two learning methods, & the tools required for methos no 1 (use of breadboards). Method number 2 using Snap-on modules will be seen once we get our hands on experience on method 1.

2 Learning Methods

We have two methods:

- Method 1 - Using Breadboards.
- Method 2 - Using Snap-on Modules.

Both have their advantages & disadvantages. We will start with breadboards using LED's & some very basic sensors. Thereafter, we shall move to Snap-on Modules doing more advance projects. This will help you focus on coding. Once you have finished this, you will be in a very good position to experiment more with breadboards.

3 Learning Tools for Method 1

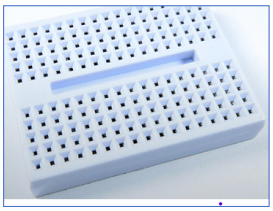
These items have to be individual procured. These have been covered one by one.

4 Breadboards

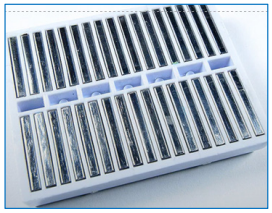
- Are used to make circuits without soldering.

They are primarily of three types:

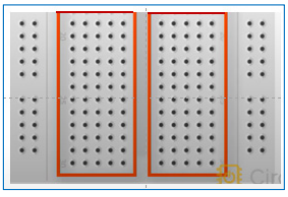
- Full size breadboards.
- Half size breadboards.
- Mini size breadboards. This is the best to start.



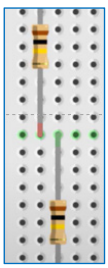
Note the presence of pins in rows.



- Reverse of this has metal strips connecting the rows like a wire.
- The rows can be divided into two rows as shown.

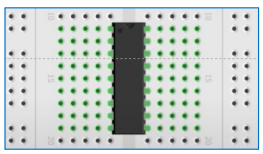


- Components are inserted in these rows as shown.



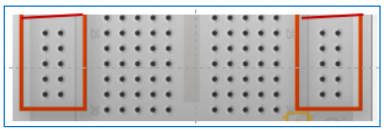
Components connected in same row are in series.

- Gap between rows has been sized for IC's



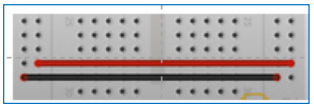
5 Power Rails

- The two rows on the side act as the two Power Rails.



Components get their power from them.

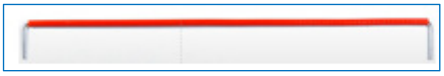
- The power rails can be interconnected with jumper wires.



6 Jumper Wires

These are of Four Types:

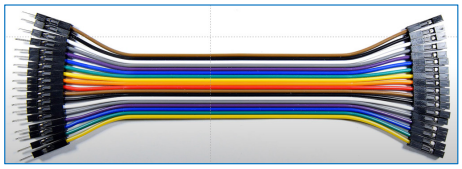
- Solid core jumpers. End insulation of these is peeled off.



- Jumpers with Male-to-Male pins.

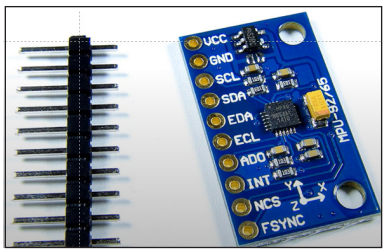
- Jumpers with Male & Female pins.

- Ribbon Jumper.



7 Pin Headers

These can be required sometimes.
These will have to be soldered by you.



8 Set of Components

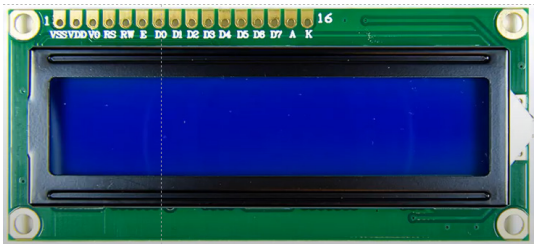
We are talking of a set containing:

- Resistances from 100 ohm to 100 K ohm.
- Capacitors from 450 Pico farad to around 2000 micro farad.
- Mixed LED's.

9 Liquid Crystal Display (LCD)

You could go in for either:

- 16 x 2 LCD.
- 20 x 4 LCD.



10 Tactile Push Buttons

You could go in for common switches & push buttons for DC operation & control LED's, motors & relays.

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

05 Installing & Using the IDE

1 Lesson Overview

By the end of this lesson students will have a good idea of installing the IDE, & the procedure of making a Sketch & running it.

2 Arduino IDE

To learn keep trying on your PC as you read.

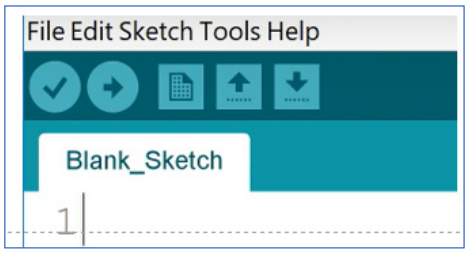
This is a free software app for programming Arduino Board. **It is used as a text editor** to create, open, edit, and validate codes or programs which in Arduino are called **"sketch".**

Beside text editor, IDE app includes :

- **Compiler.** This makes sure, the sketch is correct.
- **Uploader.** This uploads the sketch to the Arduino boards.

To download:

- click on https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE
- Select the option for windows, mac or linux.
- Click the appropriate link & follow the instructions.
- Pin the IDE in the task bar.
- Window showing the cmd prompt in line 1 appears.



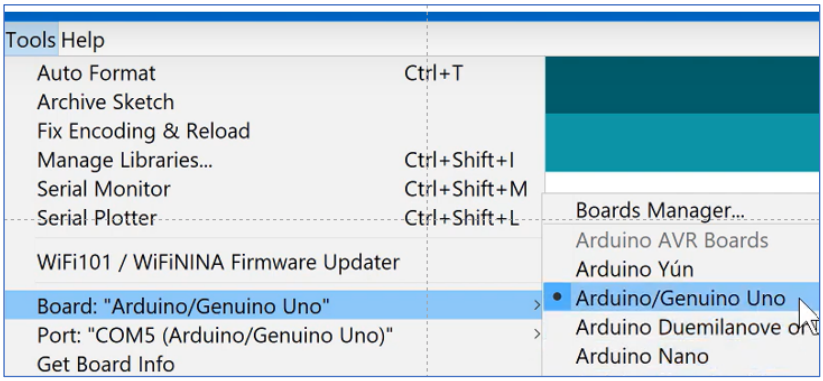
3 Selecting the Board Being Used

Boards are of diff types having diff MCU. If the MCU is different, the sketch will be different.

We will be using UNO. Thus, the first thing to do is to set the board type being used.

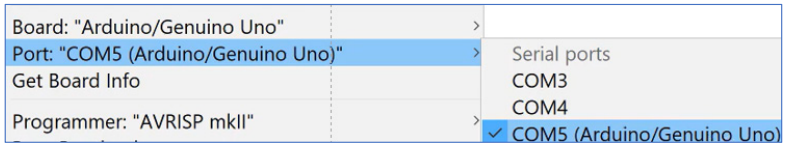
For this, click on **Tools.** In dropdown, select **Boards.**

In dropdown select **UNO.**



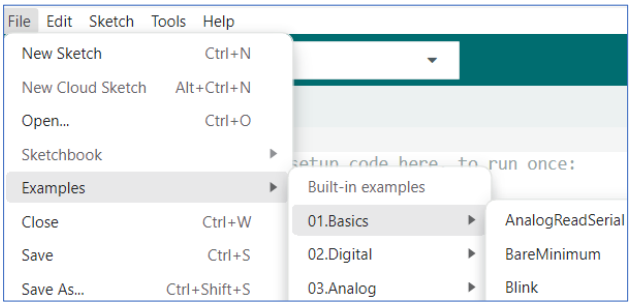
4 Deciding Corn Port Being Used

To do this, connect the board to the PC vis USB cable. In **Tool** select **Port** (below board).
In dropdown select **Com 5** for UNO.
You are now all set to Code.



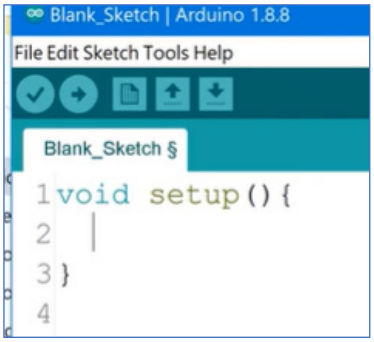
5 Seeing Example of Sketches

Before you start coding, have a look at a few built-in examples of sketches. For this under **File** click Examples. In its dropdown select **Basic.** In its dropdown select example of basic – say **Blink.**



6 Seeing Example of Sketches

- Every sketch needs two things to work properly - Set up func & loop Func. Set up func is executed only once when the sketch starts. This is used to put the code that does not change throughout the program. This is where we put for setting pins for inputs, outputs, for initializing serial monitor & for initializing the sensors.
- To create set up funs, type line 1 as shown. Leave 2 blank for entering code. In line 3, close the Curley braces. Do remember that the set-up code goes in separate lines between the two Curley Braces. It can run into any number of lines.



Remember, function is a block of code, that can be stored with a name & used again & again.

7 Loop Function

This is the second func we need to make. This is created in a similar way.It is the main part of the sketch. All the imp parameters go here. Code in this is executed over and over till stopped or you exit. It executes from top to bottom line by line.
The blank sketch looks like this.

```
1 void setup(){
2   CODE GOES HERE
3 }
4
5 void loop(){
6   CODE GOES HERE
7 }
8
```

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

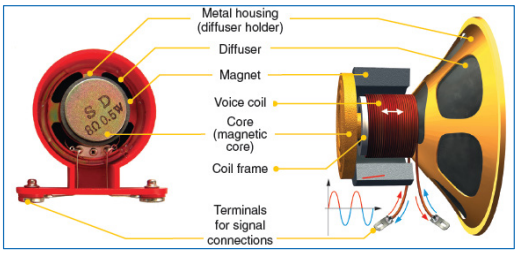
06 Snap-on Functional Arduino Modules

1 Lesson Overview

By the end of this lesson, students will have a basic idea of role, functioning & use of loudspeaker, microphone, bulb, LED, RGB, transistor, electric motor, electric motor drive, gearmotor, electromagnetic relay, power amplifier, ultrasonic ranging, thermal sensor, optical effects, universal display, & the smart traffic light module.

2 Snap-on Loudspeaker Module

It converts **electrical** oscillations into sound vibrations. A loudspeaker is also called a speaker. Its key components are shown in the figure.
These are used in headphones, TVs, computers, sound systems, telephones.
On schematic diagram it is as shown:

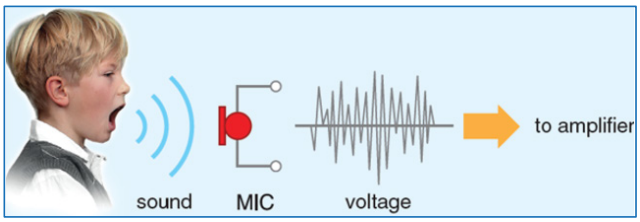


3 Snap-on Microphone Module

It converts audio signal into an electrical signal. Sound is produced by oscillations in the air. Stronger these oscillations, stronger the electric Current produced by the microphone. Being small it needs to be amplified. Schematic is: —◯—

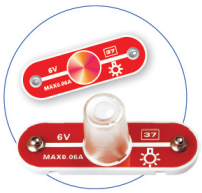
Types of microphones. These are electrodynamic, Electromagnetic, piezoelectric, capacitive, or electret. This set uses an electret microphone. It uses a permanently charged membrane to convert sound waves into electrical signals: Its main **characteristics is its** sensitivity, frequency range, directional selectivity & resistance.
Application: smartphones, computers, cameras, video cameras, record studios, movies and television, and wherever you want to record, transmit or amplify sound.

Pressure of sound air vibration acts on mics thin membrane, which in turn transmits these vibrations either to an electromagnetic coil, or a capacitor plate or a piezoelectric element, which converts these into electricity. Mic characteristics depend on type of conversion & membrane used



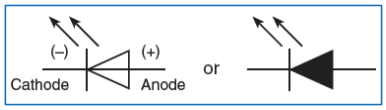
4 Snap-on Bulb Module

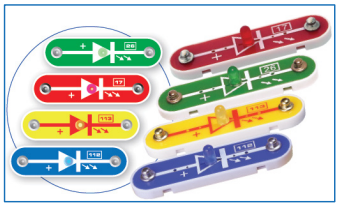
When electric current passes through a tungsten filament, it heats up & begins to glow. The filament is inside a sealed glass bulb. Our light-bulb is designed to work at a voltage of no higher than 6 volts and a maximum current of 0.3 amperes. Its schematic representation is:



5 Snap-on LED Module

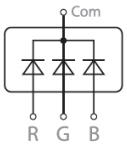
It is a semiconductor device with a **P-N junction,** which starts to glow when electric current passes through it. LEDs are **uni-directional.** They allow current to flow only from +ive to –ive inside it.
Use of LED, requires a current limiting resistor in the circuit. In our LEDs, a resistor is built in the housing. It can be connected directly to the power supply. Its schematic dia is:

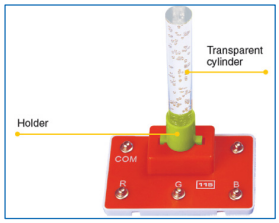




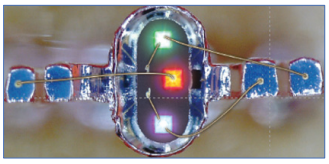
6 Snap-on RGB Module

Unlike conventional LED, a RGB module has three LEDs inside the plastic lens – a **red** one, a **green** one and a **blue** one. The transparent cylinder attaches to module **115** via a holder. Our RGB LED has a standard cathode circuit. In this schematic protective resistors are not shown.



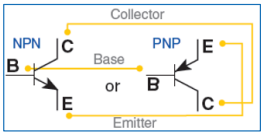


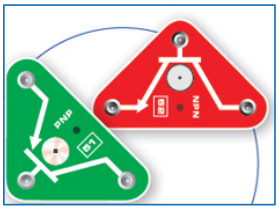
This is how an RGB LED looks under a microscope when the LEDs are dim. You can see the square semiconductor crystals and the wiring which is several times thinner than a human hair.



7 Snap-on Transistor Module

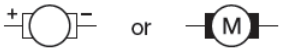
It is a semiconductor device. Depending on the switching circuit (PNP etc) used, it amplifiers, switches, generates or converts electrical signals. It Has three terminals – collector **C,** base **B** & emitter **E.** This kit uses silicon bipolar transistors with NPN and PNP conductivity. It is covered with a cap. Without the cap, it looks like this:





8 Snap-on Electric Motor Module

This converts electrical energy into mechanical energy. In our case, the electrical voltage is produced by the rotating shaft. Higher the applied voltage, higher the rotation speed of the motor. But if there's too much voltage, this can cause a breakdown. There are AC &DC motors. Electric motor **38** is a DC motor. Its schematic representation is;



To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

07 Our First Code – LED Control

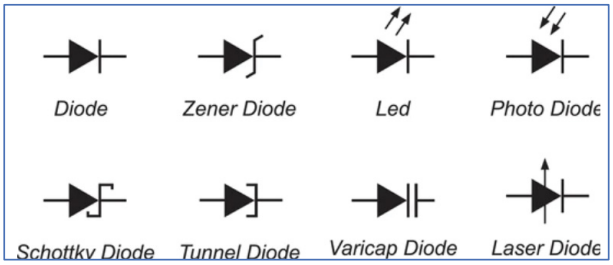
1 Lesson Overview

By the end of this lesson, students will have a functional idea of;

- What are Diodes & LED.
- How do LED function & what controls its brightness.
- Ingredients of a sketch, & the method of making the set up & loop functions.
- Changing delay timings, adding a second LED, & real life use in creating elaborate light shows.
- Introduction to the concept of variables while making Arduino sketches.

2 Diode & LED

Diode is a semi conductor device that allows current to flow in one direction. Diodes are of different types.

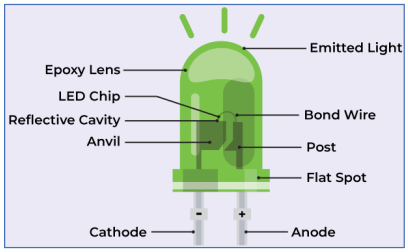


Most popular amongst them is the LED (light emitting diode).
LED is a type of diode, that emits light when current is passed through it.

Since current can flow in one direction only, they have polarity.
In an LED, one leg is longer than the other as shown in this diagram.
The longer leg is called Anode & is the positive terminal. Shorter is the Cathode & the negative terminal. Cathode connects to the ground.
Note: Bottom of LED has a rim. The part of the rim that is flat is the cathode.

3 Functioning of LED

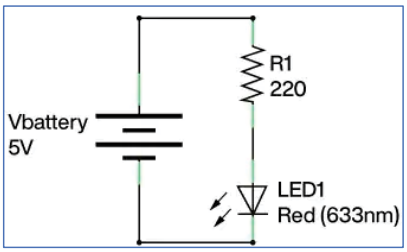
- A semiconductor LED chip (see diagram) is linked to external terminals known as anode (+) and the cathode (-). Anode in turn is linked to P-region of the chip, and cathode to N-region, forming a junction there.
- Flow of charged carriers (**electrons & holes**) across this **P-N junc** drives the activities of an LED.

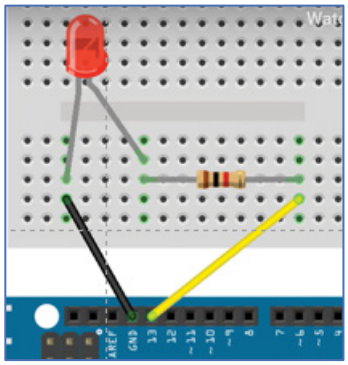


- When a forward voltage (**anode positive** in comparison to the **cathode**) is applied to the LED chip, **electrons & holes recombine** at the junc, releasing energy in the form of **photons (light).** Thus, its name **Light Emitting Diode**.

4 Brightness of LED

Brightness of a LED is determined by the amount of current allowed to flow through it. Higher the current, brighter the LED. Lower the current, dimmer the LED gets.





There is a limit to the amount of current that can be passed through an LED. To control current passing through an LED, we make use of Current Limiting Resistances in Series with it. The value of this is between 220 ohms & 1 K ohms. This is the pin diagram of connecting an LED along with limiting resistance to an Arduino UNO Board using a breadboard. Cathode connects to GND & anode to digital Pin 13.

5 Skeleton Sketch

- When you open Arduino the skeleton of this self-explanatory sketch appears.
- Set up func runs only at the start. Loop func keeps repeating.
- The two need to be populated as per the needs of the code.
- Let us see how?

Note: Comments have been given in lines 2/3 & 8/9.

```
1   void setup() {
2     // put your setup code
3     here,to run once:
4
5   }
6
7   void loop() {
8     // put your main code
9     to run repeatedly:
10
11  }
12
```

6 Populating the Set Up Func

```
1   void setup() {
2     pinMode (13, OUTPUT);
3   }
4
```

- Every project starts with information for setting up the Arduino board.
- In this, we need to tell Arduino two things – the pin num & whether it will be an input or output. This information is required whenever Arduino starts up, & since it will never change, it has been put in the set up.
- To do so, we make a Func named pinMode. It takes two comma separated args. First arg is pin num, & second is input or output mode. Since pin 13 has to send current, it will be an output (If we were using the pin to detect a sig, it would have been an input. Note: It has to be in upper case. At the end, put a semicolon.

To view more, please contact:

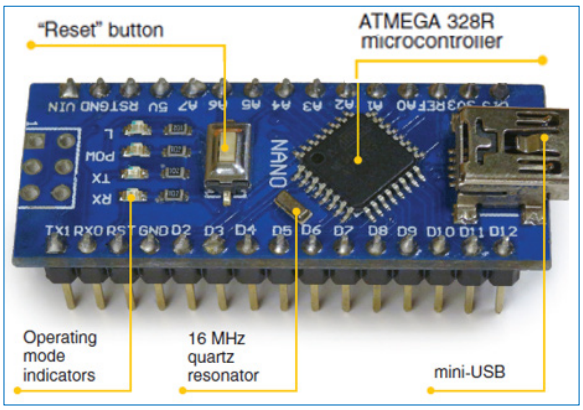
Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

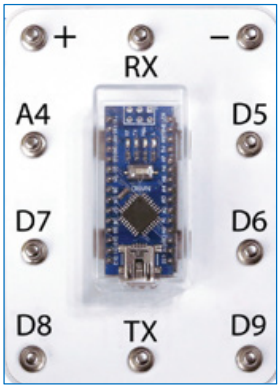
www.bdseducation.in

family. AVR STANDS for **Alf & Vegard's RISC processor** in honour of its developers.
It is an Advanced Virtual RISC (Reduced Instruction Set Computer) based microcontroller architecture.

Pin out of Arduino Module 111. As you can see, module **111** doesn't use all outputs of **Arduino Nano.** This is because the kit has been designed for doing projects focused on grasping of essentials using the most used keys of the board. Having learnt the essentials doing projects using other pins becomes a breeze.



Outputs of module 111.



"+"	"Plus" power supply
"-"	"Minus" power supply
TX	Transmission of asynchronous output data
RX	Transmission of asynchronous input data
A4	Analog input. In fact, this input is to an analog-to-digital converter (ADC), which converts input voltage into codes.
D5	Digital input/output with PWM function
D6	Digital input/output with PWM function
D7	Digital input/output
D8	Digital input/output
D9	Digital input/output with PWM function
mini-USB	Device for connecting to a computer

The microcontroller in this set can:

- **Provide information** to control external devices, such as LEDs, lamps, electric motors. Play sounds, indicate the time.
- **Receive & process information** from the environment & external devices for automation & to build various measuring instruments.
- **Analyse information,** & take decisions. AI, IoT, Automations & smart devices industry including robots take advantage of these capabilities.

Pulse-Width Modulation (PWM) is an important techique used by Arduino for controlling power applied to a load. This is done by changing the pulse width at a constant frequency.

CHAPTER

08 Snap-on Basic Arduino Modules

❶ Lesson Overview

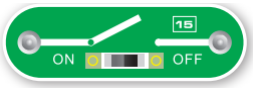
By the end of this lesson, students will have a good idea of the concept of Snap-on Kits & their advantages in learning Arduino. They will also have a good idea of its ten basic modules. However, **Do Remember** Arduino can be learnt using breadboards, if snap on kits are not available. The sketch of both is similar.

❷ Introduction to Basic Arduino Kit

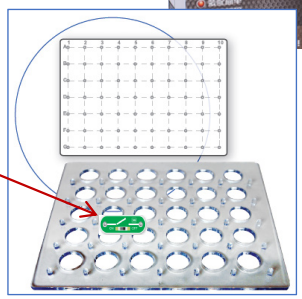
ARDUINO BASIC is **an Arduino Tinkering lab in a box.**

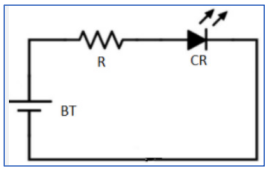
Based on Snap-on, Snap-off technology it is designed for use by enthusiasts in schools, colleges & homes.

All components in the kit including wires come as snap-on modules.



To experiment, modules are **snapped on** to a plastic mounting plate as per the circuit dia & snapped off after the experiment.

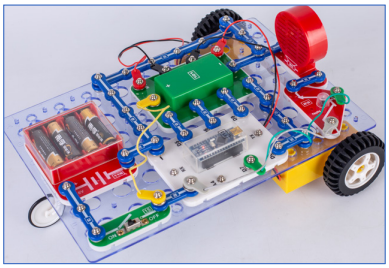




This helps children understand a circuit diagram & corelate it to the experiment he is doing. It also helps them focus time on learning coding.

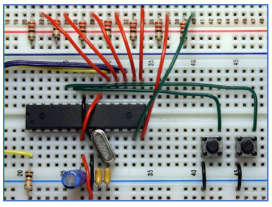
Using these children find learning of Arduino and its associated electronics fun.

Component layout of a typical experiment will look like:



It looks like a 3D circuit diagram.

It is devoid of confusing wires found using breadboards.



Children focus on sharpening of coding skills & learn **electronics** required to make daily life projects.

Using breadboards, component resources of the lab **get tied down** for days in addition to other disadvantages.

❸ Snap-on Arduino Module 111

Modules have nums & attractive colours for easy identification. Its **Arduino module** has num111.

The module uses Arduino Nano, which, in turn, uses ATMEGA 328P microcontroller from AVR

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

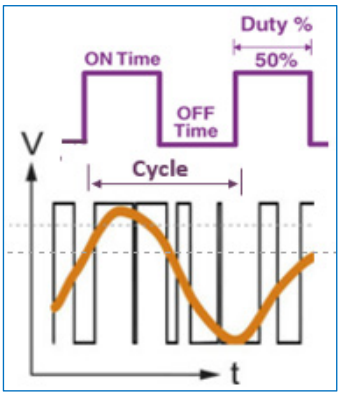
09 Controlling Brightness of LED

1 Lesson Overview

To control brightness of an LED, we must have control over its voltage. In the last lesson we saw that Arduino can have brightness value of an LED corresponding to 0 or 5 V only. This means the LED can be either fully on, or fully off. To control its brightness in-between these, we need to give it values that are between 0 & 5 V.
To do so, we use a technique called **PWM (pulse width modulation).** To use PWM, we need to understand how it works, what is a duty cycle, & how to use Analogue write func. We also need to learn how this increase & decrease of voltage, can be done by the code.

2 What is PWM

Pulse Width Modulation (PWM) is a technique used to control power delivered to electrical devices by varying the width of the pulses in a pulse train. The basic principle of PWM involves switching the power supply on and off rapidly, with the ratio of the on-time to the off-time determining the amount of power delivered to the load. This ratio is known as the **Duty Cycle.** For instance, a 50% duty cycle means the power is on for half the time and off for the other half, delivering average power to the device.



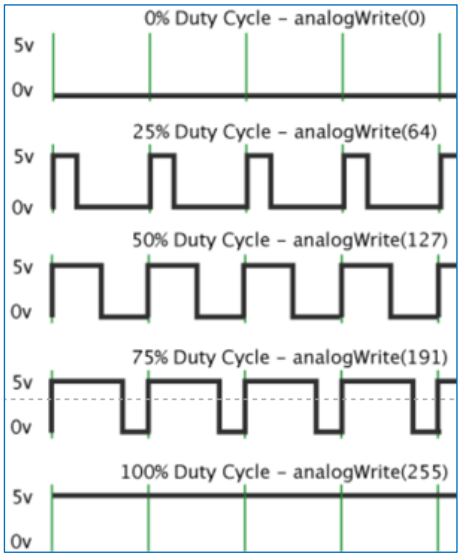
PWM is used in various applications where efficient control of power is required. The advantage of PWM is that it allows for efficient power control without generating significant heat as the switching devices (LED) operate either fully on or fully off minimizing energy loss.

3 PWM Signal

PWM signal looks like a **square wave** with voltage switching from 0 to 5 very quickly. On the Arduino these are short high freq pulses of current with a frequency around 500 Hz. Thus, there are 500 of these pulse cycles per sec. Duration of the cycle is around 2 milli sec. This is so fast that we do not notice it. Duration of the high part of the pulse is called Pulse width. Duration of the cycle is duration of high & low part of the pulse. A duty cycle is expressed as a percentage of upper to lower signal. It is expressed as:

$$Duty\ Cycle = \frac{Pulse\ Width\ Duration}{Cycle\ Duration} \times 100\%$$

4 Implications of Duty Cycle



To control the brightness of LED we have to make Arduino mimic an analogue signal (yellow **sine curve**). This is done writing **analogueWrite** functions instead of digital The equivalence of the two is shown in this dia.

- With a duty cycle of 100 %, the voltage is on for the full duration & output is 5 v (Bottom Graph). This corresponds to analogueWrite of 255.
- With a duty cycle of 75, the pulse width is on for 75 % of the cycle and voltage is 75 % of 5 v or 3.75 V & analogueWrite if 191, & so on.
- Thus, by changing the duty cycle we can change the voltage value to any value between 0 & 5 V.

To find the analogue value that will provide a specific voltage, use the formula

$$analogWrite\ Value = 255 \times \frac{Desired\ Voltage}{5\ Volts}$$

where 5 is the max voltage & the step in which the voltage will decrease with each iteration of the **for loop.**

5 PWM Compatible Pins

PWM does not work with every pin. It works with pins that have squeaky line next to them. For UNO these are digital pins 3, 5, 6, 9, 10 & 11.

6 Making the Set-up Function

For this, we need to first define the variables. We have made three var:

- First is **ledpin** = the pin the LED is connected to (line 1).
- We will iterate full range of analogue values from 0 to 255. So, we need another var to store that value. This is **brightness** & has been set to 0 (line 2).
- WE also want to control how long LED's stays on or off. One way is to see how many steps we take to go from 0 to 255. So, I have taken a var **fade amount** & set it to 5 (line 3).
- In setup func we have to set the pinmode with two args - **ledpin & OUTPUT** (line 6).

```
1 int ledPin = 6;
2 int brightness = 0;
3 int fadeAmount = 5;
4
5 void setup() {
6   pinMode(ledPin, OUTPUT);
7 }
8
```

7 Making the Loop Function

The loop has to control the three vars.

- Line 10 def the values for args **ledpin & brightness.**

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER 10

Coding a Push Button Switch

1 Lesson Overview

In this lesson we shall learn what are switches, the use of push buttons in any Arduino project, what are floating pins, what problems they face, & how to prevent them. We will also earn the use of pull-up & pull-down resistors, how to use the digitalRead func, & how to use Arduinos internal pull-up resistor. Try & follow the logic. Learning logic will come handy with other types of switches & sensors when used.

2 What are Switches

Electric switch is a basic control device that manages the flow of electricity in a circuit.

- They allow you to turn electrical devices on or off by either opening or closing the circuit.
- When the switch is in the "on" position, the circuit is complete, allowing electricity to flow and power the connected device. On the other hand, when the switch is in the "off" position, the circuit is broken, stopping the flow of electricity.
- Knowing what the purpose of using an electric switch is essential for proper coding.

There are two essential components of a switch. First is the num of circuits a switch can control. This is defined as its **poles.** The second is the number of positions a switch can adopt. This is defined as **throws.**

In a single-throw switch, there is only one pair of contacts, such as close or open. **Push button** is a single-throw switch.

3 Push Buttons

Push buttons are the most used control device for things like LED's relays, motors etc. With potentiometers we learnt the measuring of analog input voltages. With push button, we shall learn the measuring of digital input voltages.

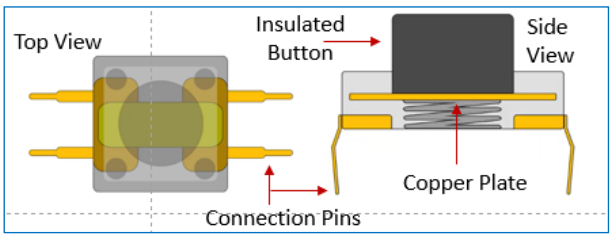
They are also known as Tactile push buttons or Momentary push buttons.

For our learning we shall be using a commonly available push button.

4 Construction of a Push Button

Top & side views are shown. Note the:

- Insulated button on top.
- Copper plate that makes contact, & the spring below to break it.
- Connecting pins.



We will connect this button to the Arduino. In its sketch, we will ask it to **detect when the button is pressed, & then send a high signal to the LED, which will then go to on.**

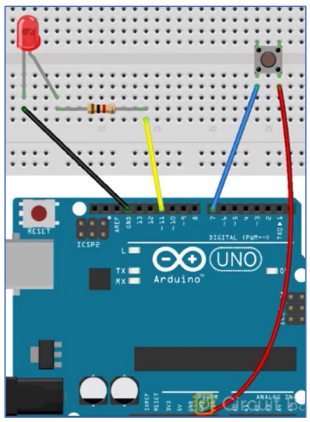
5 digitalRead() Func

This is similar to digitalWrite we learnt earlier. Instead of sending a signal to a GPIO pin, this will read the signal coming from a GPIO pin.
digitalRead takes only one arg & that is which pin we want to read.
This returns value 1 or 0 depending on the voltage at the pin. It returns value 1 for high voltage, & value 0 for low or no voltage.

6 Making the Connections

The circuit for this is as shown.
Using jumper wires:

- Connect Cathode of LED to GND.
- Anode to pin 11 via current limiting resistance of 200 ohm to 1 K ohm.
- Right pins of push button connect to 5 V.
- Left pins to pin 7.



Let us look at its working:

- When we press the button, 5 V shows up at pin 7, & causes the pin to go High. When we write the sketch, this will help us detect when pin 7 goes high.
- Then use digitalWrite func to set pin 11 to high. This will put on the LED.

7 Making the Sketch

Lines 1 & 2 declare the var for the two GPIO pins 7 & 11.
In set up func we set the two args, button pin as an input, & led pin as an output.
In loop func we set up a var equal to a func Line 10. This var will Keep a track of when push button is pressed.

```
1 int buttonPin = 7;
2 int ledPin = 11;
3
4 void setup() {
5   pinMode(buttonPin, INPUT);
6   pinMode(ledPin, OUTPUT);
7 }
8
9 void loop() {
10   int buttonState = digitalRead(buttonPin);
11   digitalWrite(ledPin, buttonState);
```

8 Logic of the sketch

Track is kept by arg button pin (7) in line 10. When button is not pressed voltage of pin will be low, so digitalread func which is listening to pin 7, returns low & the value will be stored in the button pin var of line 10. When it is pressed vol at the pin will be high & will again get stored in the button pin var.

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER 11

Data Types, Variables & Constants

1 Lesson Overview

In this lesson we shall learn the basic fundaments of Data Types, Variables & Constants. These will help us in making sketches ahead.

2 Important Resource

So far, we have seen the making of a few sketches. This has given us an idea of some of the terms involved. To make more sketches, it is important for us to learn the key elements of the theory. More will be learnt as we experiment with sensors & boards. Arduino has provided a **very useful resource** to help us learn on our own.
The same is – https://www.Arduino.cc/reference/en
Kindly go through it following its menu & instructions.

3 What are Data Types

Data types refer to a system used for declaring variables or functions of different types. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

1) Data Type Void

In Arduino this specifies an empty set of values, and only used to declare functions. Every Arduino sketch includes **void setup()** and **void loop().** Without them, your program won't run. The code that we put inside void setup() will only run once, at the beginning of our program. Example, when we want to turn our robot on — this does not happen multiple times. Code that we put inside Void loop() performs the action. This is where we can manipulate inputs and outputs, such as writing to a pin or changing a sensor value.

2) Data type Int

These store whole numbers like 8, -901 etc. They do not have any fractional part. Uses 2 bytes of memory. Can hold 2 to power of 16 = 65,536 nums, split from -32768 to 32767.

3) Data Type Float

Stores positive & negative fractional nums. These nums must be written with a decimal point (5.03). Use 4 byte of memory. Can store nums from 3.4028235 X 10 to power of 38 to -3.4028235 X 10 to power of 38.

4) Data Type Boolean

Stores boolean values. These return TRUE or FALSE, HIGH or LOW, & 1 to 0.

5) Data Type Char (character)

Stores single letters, nums, or characters – a, A, 4, @, %. Can be any ASCII char, or ASCII numeric value from -128 to 128.

6) Data type Byte

Stores integers from 0 to 155. Uses 1 byte of memory.

7) Data Type Const Int

Uses **const** to make variable constant. This is used for pin numbers & constants. Does not use any SRAM (static-random access memory).

8) Data Type Unsigned Int

Unsigned ref to only positive nums. These are nums from 0 to 65,536. Use 2 bytes of memory.

9) Data Type Long

Used to store long integers. Can hold 2 to the power 32, or 4,294,967,296 different nums, split between – 2,147,483,648 to 2,147,483,647. Use 4 byte of memory.

10) Data type Unsigned Long

Used to store long integers. Can hold 2 to the power 32, or 4,294,967,296 different nums from 0 to 2,147,483,647. No negatives. Use 4 byte of memory.

4 Variables

1) What are Variables

They are defined as **named containers,** used to store data. Look at Variables (**Var**) as a box containing something. If you put jewellery inside, it will become a jewellery box. In the same way, if you put an arithmetic value like 6 inside it will become a **numeric var.**

2) Types of Variables

Var type is determined by **what it holds**. Variables can hold Nums (integers & float), **Text** (strings), **Mix of nums & text, Booleans, Funcs, & other vars.**

3) Types of Variables

- Once made, vars are stored in Arduino's memory. For its subsequent retrieval, it must have a **unique name** (example ledPin).
- It becomes a var, only once it is given a **value** (example ledPin = 10).
  Process of giving a value for first time is called **Initializing a var.**

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

CHAPTER

12 Operators & Functions

1 Lesson Overview

In this lesson we shall learn the basic fundaments of Basic Operators, & Functions.

2 Operators

An operator is a **symbol** that **tells** the compiler to **perform** specific mathematical, logical or conditional operation on a var or a value. They are used to manipulate data as per the needs of the action to be performed.

1) Operand

An operand refer to a value or expression on which some actions are performed. It is manipulated by an operator by sitting on both its side. In the ex 5 + 6 = 11; 5 & 6 are two operands that sit on either sides of the plus operator to give the result 11. The two operands could also be written as X + Y, where X has value 5 & Y has 6

2) Types of Operators

C language is rich in built-in operators. It provides five type of operators. These include Arithmetic Operators, Comparison Operators, Boolean Operators, Bitwise Operators & Compound Operators.

3) Arithmetic Operators

These execute basic arithmetic functions in your sketch. Arduino, offers six of them. Assume A has value 10, & B has 20, then:

Name	Symbol	Description	Example
assignment operator	=	Stores the value to the right of the equal sign in the variable to the left of the equal sign.	A = B
addition	+	Adds two operands	A + B will give 30
subtraction	-	Subtracts second operand from the first	A - B will give -10
multiplication	*	Multiply both operands	A * B will give 200
division	/	Divide numerator by denominator	B / A will give 2
modulo	%	Modulus Operator and remainder of after an integer division	B % A will give 0

Sketch using Arithmetic operator.

- Line 1 & 2, declare var num1 & num2.
- Line 3, create var sum.
- Line 5, initialise func begin.
- Line 6, declare var sum.
- Line 7, initialise func print.
- Line 8, initialise func println.
- Connect Arduino & serial monitor.
- Run.

Try others operators in similar way.

```
1   int num1 = 28;
2   int num2 = 10;
3   int sum;
4   void setup() {
5   Serial.begin(9600);
6   sum = num1 + num2;
7   Serial.print("Addition of num1 and num2 is");
8   Serial.println(sum);
9   }
10
11  void loop() {
12  }
```

4) Comparison Operators

These compare values on the either side of operand. Arduino offers six of them. Assume A has value 10, & B has 20, then:

Name	Symbol	Description	Example
equal to	==	Checks if the value of two operands is equal or not, if yes then condition becomes true.	(A == B) is not true
not equal to	!=	Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	(A != B) is true
less than	<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true
greater than	>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true
less than or equal to	<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true
greater than or equal to	>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true

Sketch using greater than comparison operator

- Line 1 & 2, declare var.
- Line 4, initialise func begin.
- Line 6 & 7, declare the two vars.
- Line 7, initialise func print for result1.
- Line 8, initialise func println for reult1.
- Line 9 & 10 repeat for Result2.
- Connect Arduino & serial monitor.
- Run.

Try more yourself in similar way.

```
1   int Result1;
2   int Result2;
3   void setup() {
4   Serial.begin(9600);
5   Result1 = (8>5);
6   Result2 = (5>9);
7   Serial.print("is 8 greater than 5: " );
8   Serial.println(Result1);
9   Serial.print("is 5 greater than 9: " );
10  Serial.println(Result2);
11  }
12  void loop() {
13  }
```

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

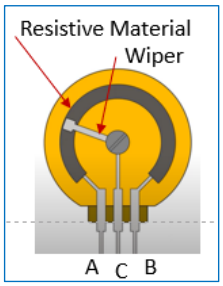
CHAPTER

13 Coding a Potentiometer

1 Lesson Overview

In this lesson we shall learn what are potentiometers, how they work, how to control analogue voltages, & how to use serial monitors.

2 What are Potentiometers

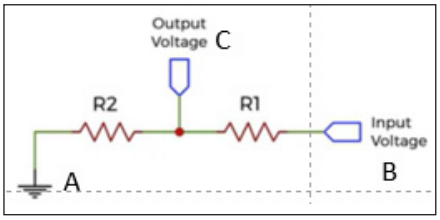


A potentiometer is a variable resistor that can be used to measure voltage, regulate power, or introduce resistance levels:

- It has three terminals – A B & C.
- It consists of a U-shaped layer of resistive material called **resistive element** connected to a positive terminal A & a negative B.
- It has a moving contact called a **wiper**. The wiper is connected to common terminal C. This pin is also called the **tap** that provides the variable voltage. It moves from one end of the resistive element to the other & the resistance it thus produces Is proportional to the wiper's position.
- **Turning the knob** attached to the wiper changes the resistance in the circuit. More the resistive material it covers, more the resistance. At point B it is 0 & at point A it is max.
- They can be used to adjust the vol of an amplifier, control motors or adjust the brightness & contrast of an LCD display.

3 How does a Potentiometer Work

Think of potentiometer as a voltage divider used to reduce or increase the voltage in a circuit. It consists of resistances R1 & R2 in series. Terminals at which input & output voltages will be available are B & C.



In case of Arduino the voltage varies from 0 to 5V. If R1 = R2, then output voltage will be 2.5V. If R2 has a very high resistance, output will be very close to 5V, & if R1 has a very high resistance, the output will be close to 0V.

Thus, the voltage available at pins B & C can be adjusted manually, or automatically using Arduino. Let us see how?

4 Reading Output Voltage from a Potentiometer

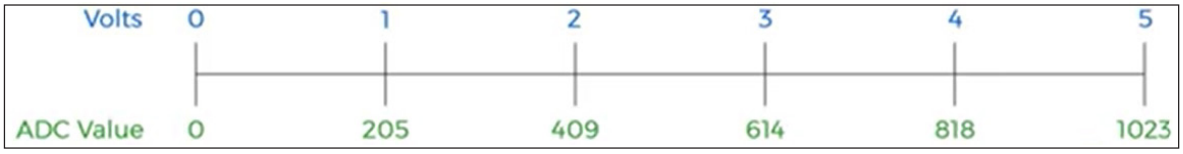
To measure the output voltage, we need to connect the output pin of the potentiometer to one of Arduinos analogue pins. It has six analogue pins from A0 to A5. Any of these can measure the output voltage using **analogRead ()** func. It needs only one arg – The **analog pin num** at which we want to read the voltage – **analogRead(pin);**

The voltage is measured by Arduinos **analog to digital converter** & varies from 0 to 5V.

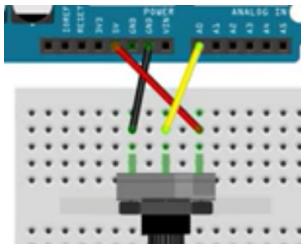
5 A to D Converter - ADC

An ADC (A to D Converter), converts a continuous-time & continuous-amplitude analog signal to a discrete-time & discrete-amplitude digital signal. Arduinos ADC:

- Converts analog voltages to digital values.
- Outputs a number between 0 & 1023 to represent these values. Thus, 0 V has ADC value of 0 & 5 V has ADC value of 1023. Other voltages fall in between as show here.



6 How does this Work



Analog pin func returns the ADC value of the analog pin it is measuring. To see how, populate the breadboard as shown. Right pin of potentiometer connects to 5 V, left to GND, & center to A0.

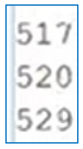
The sketch for this will look like:

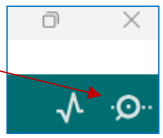
- This will display the values returned by analogRead on Arduinos serial monitor. Serial monitor displays data on PC monitor.
- Line 2 initializes serial monitor. It takes one arg (Baud rate). This is selected in the dropdown of serial monitor.
- Line 6 creates a var rawvalue. This takes one arg (A0).
- Line 7 is print func. It takes one arg & that is a num, or string, or var you want to print. Line 8 gives delay of 100 milli sec.
- Upload to Arduino.

```
1   void setup() {
2    Serial.begin(9600);
3   }
4
5   void loop() {
6    int rawvalue = analogeRead(A0);
7    Serial.println(rawvalue);
8    delay(100);
9   }
10
```

7 Printing to Serial Monitor

Once uploaded, open serial monitor using the magnifying glass. You will see a bunch of nums being printed.





- These nums are raw ADC values. Moving potentiometer all the way to left drops ADC values to 0, & moving all the way to right increases them to 1023.
- We now need to **convert raw ADC values to voltages.** For this we make changes in loop func.

To view more, please contact:

Diwaker :- +91 93122 64502

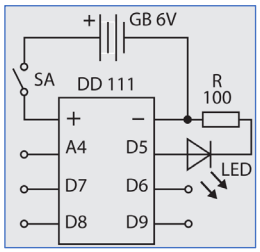
Pankaj Kabir Panthi :- +91 88514 60895

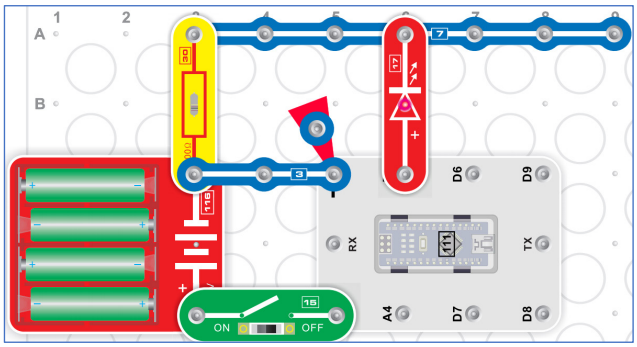
www.bdseducation.in

CHAPTER

14 Control Of Devices

1 Blinking of LED





To control blinking of LED, assemble the circuit as shown. Its code is as below.

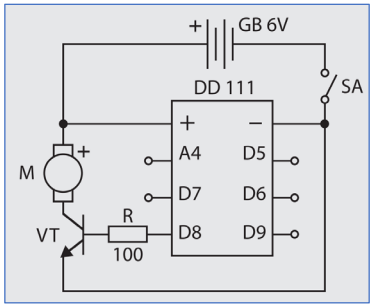
- In set up func, **pin number** 5 is set to **digital output pin**
- In loop function pin number 5 is made **HIGH** or given **5 volts for 100 milliseconds** and then it is made **LOW** or given **0 volt** for **900 milliseconds.**
- These two actions are carried out for forever and the LED turns **ON** for less time and **OFF** for more time representing blinking action of the LED.

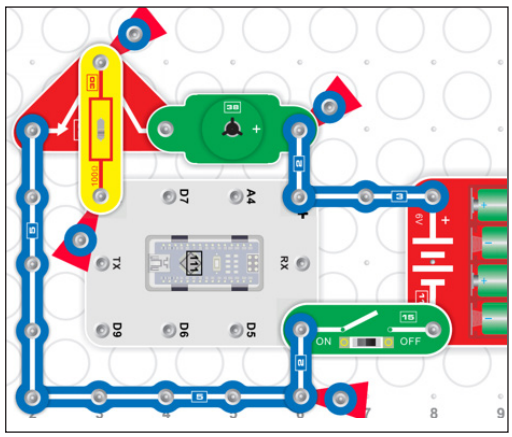
Code of Project 1.

```
void setup() // Setup function - all commands are executed only once
{
  pinMode(5, OUTPUT); // Setup digital pin D5 to OUTPUT
}

void loop() // Main loop - commands executed before shutdown
{
  digitalWrite(5, HIGH); // Set "HIGH" logic level on D5 - switch on LED
  delay(100); // Do nothing 100 ms
  digitalWrite(5, LOW); // Set "LOW" logic level on D5 - switch off LED
  delay(900); // Do nothing 900 ms
}
```

2 Motor Control

Assemble circuit shown here.





Method & Code:

Assemble circuit shown below. **Pin 8** is set to **digital output Pin 8.**

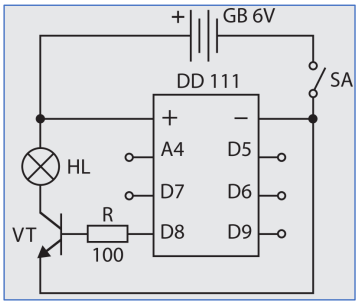
- In the setup func motor is OFF. Then a **for loop** is run where pin 8 is made **HIGH** or given **5 volts for 2 seconds** and then it is made **LOW** or given **0 volt** for **2 seconds.**
- These two actions are carried out for **5 Times.** The motor turns **ON** for 2 secs, & **OFF** for 2 secs.

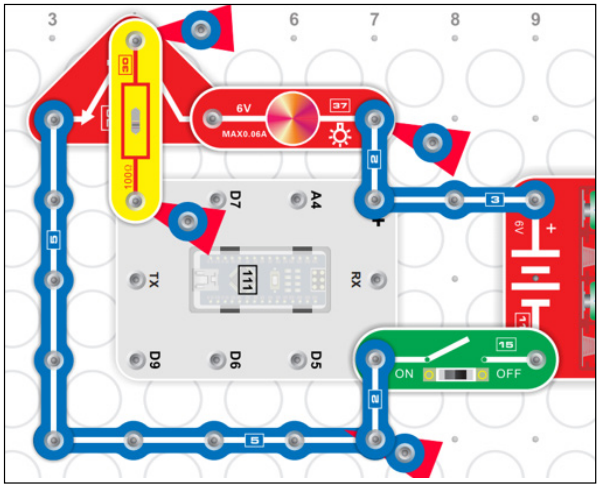
Note- A motor requires larger current as compared to a LED, hence, transistor is used in the circuit.

```
#define MOTOR_PIN 8

void setup()
{
  pinMode(MOTOR_PIN, OUTPUT);

  // Switch off the motor at the beginning
  digitalWrite(MOTOR_PIN, LOW);


  int switchDelay = 2000;
  for (int i = 0; i < 5; i++)
  {
    digitalWrite(MOTOR_PIN, HIGH);
    delay(switchDelay);
    digitalWrite(MOTOR_PIN, LOW);
    delay(switchDelay);
  }
}
```

3 Lamp Control

Assemble the circuit as shown.





Note- A lamp requires larger current as compared to a LED, hence, transistor is used.

- As shown, following is the code for the **functioning of a lamp**. Here the **pin number 8** is set to the **digital output pin** for a lamp to function in the setup function.
- In the **loop function** the pin number 8 is made HIGH or given **5 volts** for **2 seconds** and then it is made **LOW** or given **0 volt** for **2 seconds.**
- These two actions are carried out for forever and the lamp turns **ON** for 2 secs and **OFF** for 2 secs representing the blinking of a lamp.

Note-The timings (switch delay) can be changed to see and observe different results.

```
#define LAMP_PIN 8

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

void loop()
{
  int switch_delay = 2000;

  digitalWrite(LAMP_PIN, HIGH);
  delay(switch_delay);
  digitalWrite(LAMP_PIN, LOW);
  delay(switch_delay);

}
```

To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

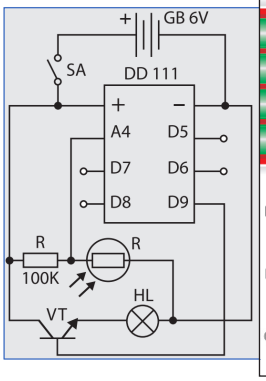
www.bdseducation.in

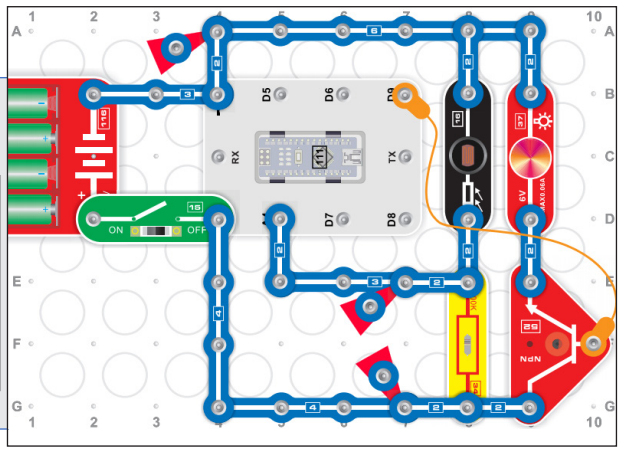
CHAPTER

15 Feedback Control & My First Robot

1 Smart Light System

Assemble the circuit as shown.





Project Code.

This is code for **a light Activation in the night.** Here **pin 9** is set to **digital output pin** for a lamp to operate and **pin A4** is set to **analog input pin** to operate a **photoresistor** in the setup function.

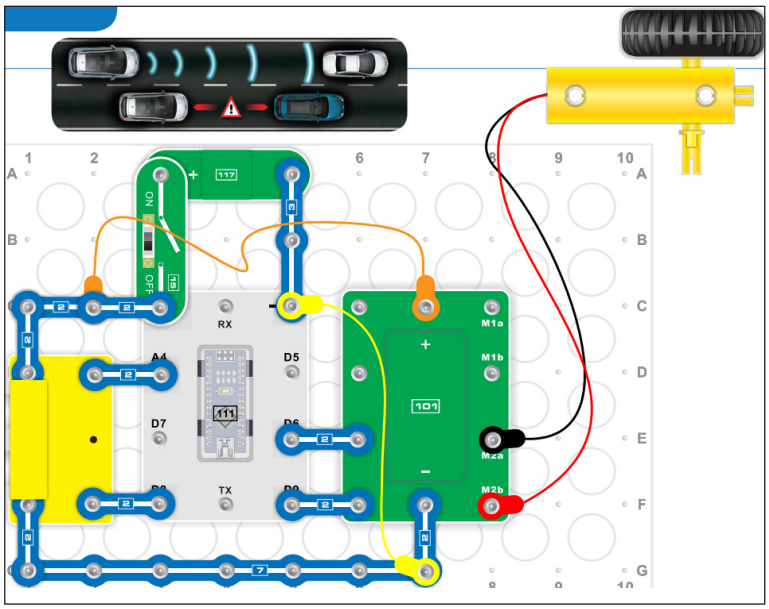
```
#define LAMP_PIN 9

void setup() {
  pinMode(LAMP_PIN, OUTPUT);
  pinMode(A4, INPUT);
}
```

```
void loop()
{
  int minBrightness = 0;
  int maxBrightness = 255;
  int threshold = 800;

  int brightness = analogRead(A4);

  if (brightness > threshold)
  {
    analogWrite(LAMP_PIN, map(brightness, threshold, 1023, minBrightness, maxBrightness));
  } else
  {
    digitalWrite(LAMP_PIN, LOW);
```

In **loop function** condition is checked if **pin A4** reads the value **greater than a threshold value** & **photoresistor** catches the **light.** When the condition is **true** then **pin 9** is made **HIGH according to the brightness of light**. When light intensity is **more**, the lamp is **OFF** & when light is **less or no light** then the lamp is **ON**.

1 Tailgating System

Tailgating means following a car very closely. Assemble circuit as shown. Its setup func is:



```
#define D1 6
#define M1 9

void setup()
{
  pinMode(D1, OUTPUT);
  pinMode(M1, OUTPUT);
}

Ultrasonic ultrasonic(8, A4);
int cur_time = 0;
```

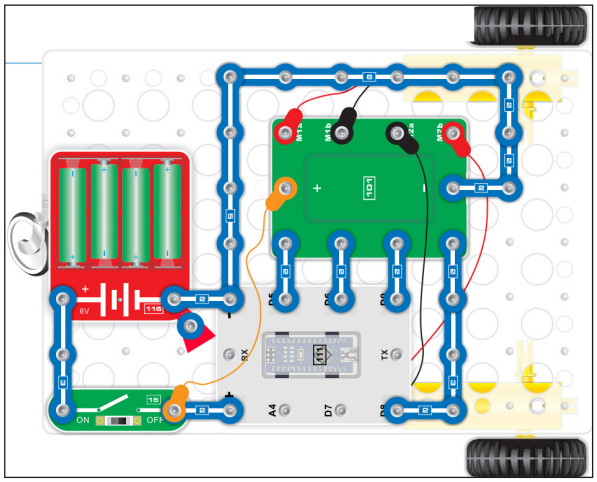
Here, **Pin 6 & 9** are set to **digital outputs** in the **setup function** to operate the motor driver. The **Ultrasonic function** is called to read the Trigger pin(**input at pin A4**) & **Echo** pin(**output at pin 8**) & an instance is created.

In **loop function** ultrasonic sensor reads the **distance** between sensor & obstacle.
The condition is checked, if the distance is **greater than 32 cms** then motor runs in **forward** direction, else if distance is **less than 29 cms then** motor runs in **backward** direction.
If **none** of the conditions is satisfied then motor **stops** running.

```
void loop() {

  int dist = ultrasonic.distanceRead();

  if (dist > 32)
  {
    digitalWrite(D1, HIGH);
    digitalWrite(M1, LOW);
  } else if (dist < 29)
  {
    digitalWrite(D1, LOW);
    digitalWrite(M1, HIGH);
  } else
  {
    digitalWrite(D1, LOW);
    digitalWrite(M1, LOW);
  }
  delay(100);
```

1 Robo moves in Square Path in Clockwise direction

Assemble circuit shown here.



To view more, please contact:

Diwaker :- +91 93122 64502

Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

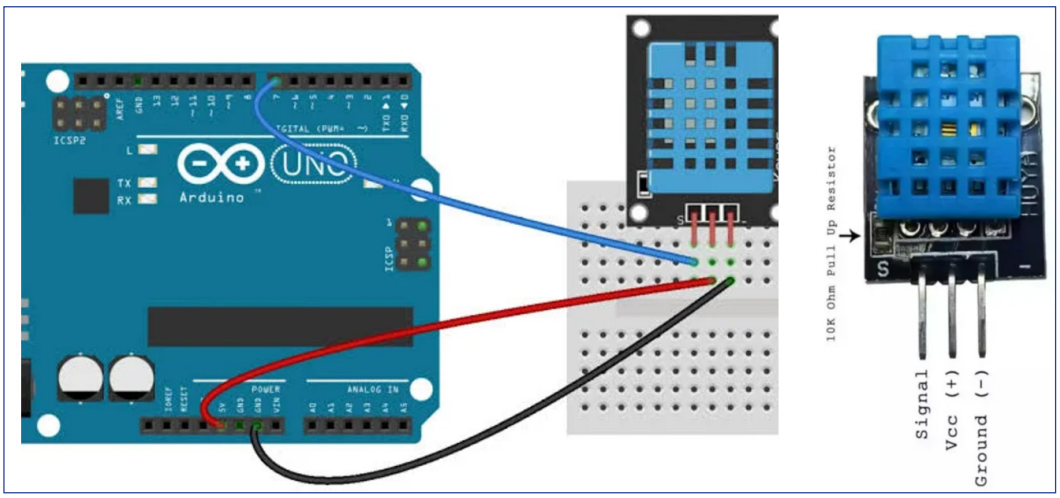
CHAPTER

16 Arduino Projects Using Breadboard

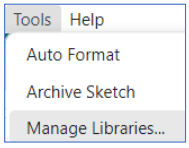
Project 1. Temperature & Humidity Detection

1) Preparation.

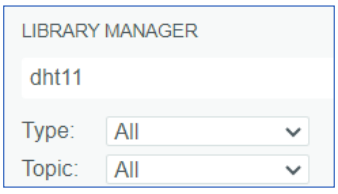
Get an **Arduino Uno board, a breadboard** and a **DHT11 Temperature & Humidity Sensor.** Assemble the circuit as shown.



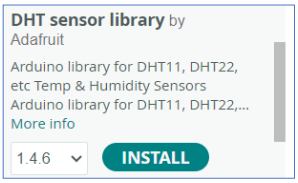
1) Installation DHT11 Temperature & Humidity Sensor library.



- Go to **Tools tab** & click & select **Manage Libraries.**
- The **Library Manager** opens on the **left side** of the screen. Type **dht11** in the search box and press Enter.

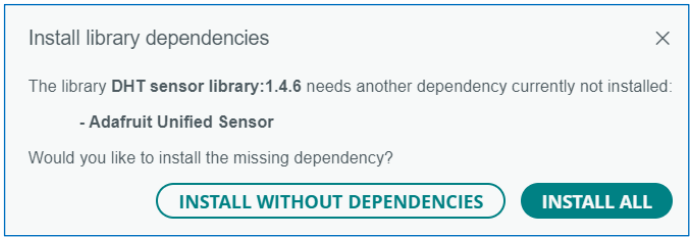


A number of sensors versions are available as scroll down. Select version of your choice. Here, the version selected is shown in the figure.



- Click on **Install** button to install the library.

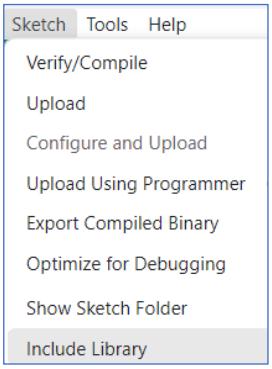
- A dialog box appears for the installation, click on **Install All.**

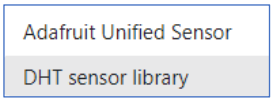


```
Output
Downloading Adafruit Unified Sensor@1.1.14
Adafruit Unified Sensor@1.1.14
Installing Adafruit Unified Sensor@1.1.14
Installed Adafruit Unified Sensor@1.1.14
Downloading DHT sensor library@1.4.6
DHT sensor library@1.4.6
Installing DHT sensor library@1.4.6
Installed DHT sensor library@1.4.6
```

- Observe the installation of library & it's file down below in the **Output** window.
- Once installation is done, it prints **Installed DHT sensor.**

- Now, go to **Sketch tab** on the top & select **Include Library** from the options.
- Click on the **DHT sensor library** and **Adafruit Unified Sensor** to include the related **header file** in the code in script area.





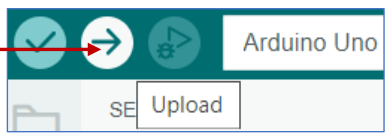

```
#include <Adafruit_Sensor.h>

#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 7
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {

  Serial.begin(9600);
  dht.begin();

}
```

Header files related to the sensor are included in the code.
In the code, **Pin 7** is set for the **digital input** & an **instance dht** has been created of **type DHT.**
Also, type of **DHT sensor** has also been defined as **DHT11.**
In the **setup function, dht sensor is initiated** and **serial monitor** of Arduino IDE is **initiated.**

In the **loop function** two variables are created.
One for **temperature** & one for **humidity.**
Both are assigned to the functions being called by **dht instance** respectively.

```
void loop() {

  float h = dht.readHumidity();
  float t = dht.readTemperature();
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  delay(2000);

}
```

Now upload the code by clicking on the **upload** button on top left as show in the figure.



To view more, please contact:

Diwaker :- +91 93122 64502

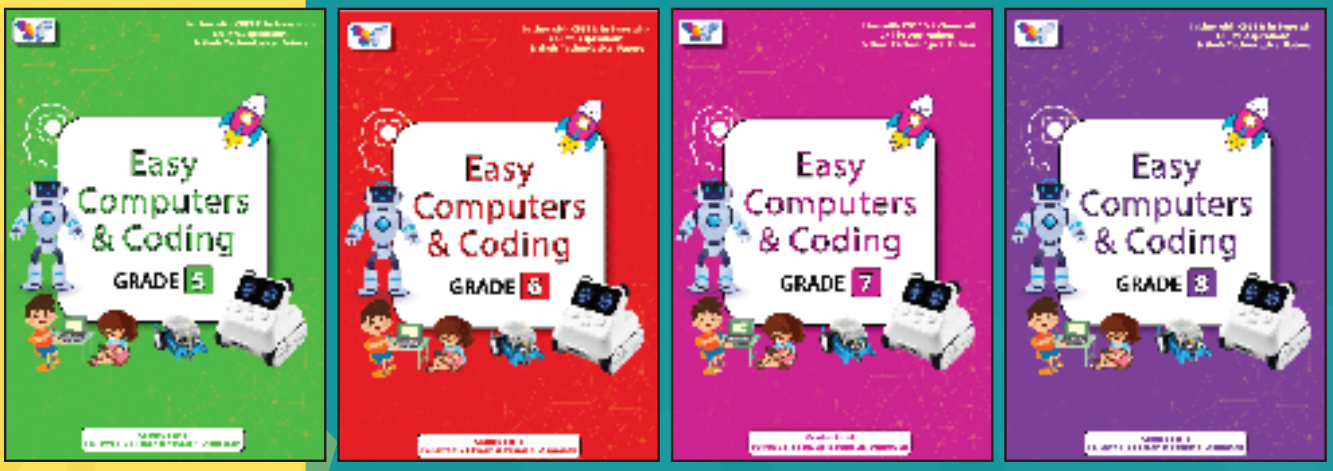
Pankaj Kabir Panthi :- +91 88514 60895

www.bdseducation.in

OUR BOOK TITLES

Level-1

Level-2



Level-3

BDS EDUCATION