LEARNING ARDUINO IN 2 MONTHS

CODING MADE EASY

ARYAN KURKURE

Ithought of writing this book as I had one main motive that coding should be known by everyone as coding is going to start to enter each and every occupation in 21rst century this is why I wanted too show how to code arduino ,so that instead of working with robots in future you will master them. This is why read the book with full enthuiasam so that you understand what's the logic and C++ language all about .

I was stimulated to write this book after seeing my cousins' and my friend's keen interest in arduino coding ,this is why this book is dedicated to my dear cousins Ojas, Adruth and my friend Shrihan.

Contents

Foreword

I started by buying a arduino board and some important components with it and my first experience was really very interesting and joyful, I tried to understand that how the code works makes the arduino work, I simply searched for arduino coding app on google and discovered that I was not the only person who was coding arduino but there was a whole community of arduino coders, it was really one of beautiful thing which I discovered I felt really comfortable and relaxed after I came to know that there are a lot of people who know and can help me to code arduino ,so if you are a little bit tensed coding arduino for first time please don't feel tensed as this book is going to be your companion for your life in learning arduino.

Preface

The bold text/pictures of this book is licensed under Apache 2.0 license. Arduino is a registered trademark of of arduino LLC, to learn more visit,

http://www.arduino.cc/en/Main/FAQ

CHAPTER ONE

What exactly is arduino ?

The first Arduino board was introduced in 2005 to help design students — who had no previous experience in electronics or microcontroller programming — to create working prototypes connecting the physical world to the digital world. Further according to needs of people having interest in arduino etc. many arduino boards were created such as arduino UNO, NANO, MEGA etc. however over time all these boards became older and had many limitations due to which they were not used because of this arduino company decided to make modifications in arduino board, for instance arduino Uno R3 board is much more advanced then the previous arduino boards this is why this is mostly sold.

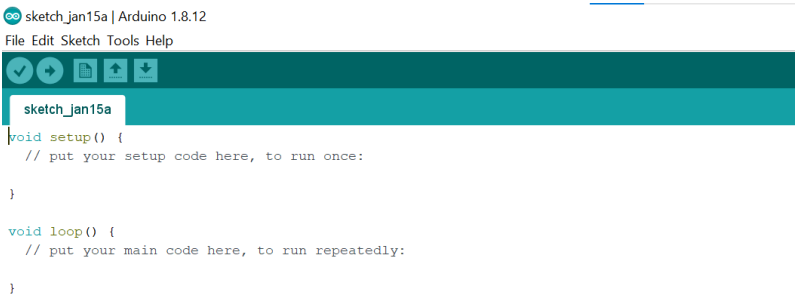
If we take a look at arduino closely ,its easy to notice some group of pins , power pins, digital pins & analog pins. All these pins have different use for example power pins on arduino board can supply 5v ,9v,3.3v of electricity without programming however the case is much different with digital and analog pins as if you want to use one of these pins

you need to know how to code in C++ in arduino IDE app which you can download from google for free. Before going to the coding part lets first discuss about digital and analog pins, digital and analog have one major difference that digital means communication in 0,1 (binary format) whereas analog means communication in a range of 0,1023 which increases the reading capacity of sensor in a efficient manner, for example digital output from sensor will either be 0 /1 which can't tell us the value of sensor properly on the other hand if we use analog pins we can understand the reading more efficiently as it will give a good range and variability. Other thing about arduino Uno board is that it has 32 Kb storage in Atmega328P chip and processor of 2Kb which is pretty fast as arduino needs to run text code which hardly occupies any space in memory.

CHAPTER TWO

Starting to code in Arduino IDE

Before we start to make some important and smart projects we will simply practise some important functions before we start to use sensors, motors and all other thimgs. In this chapter we will be learning how to introduce new connections ,classifying them as input or outputs and switching them on and off, I wish all the readers to please read this chapter with full concentration as the concepts which we are going to learn today are very important as if you don't learn these you won't be able to learn other chapters.



This is how the arduino IDE app looks like.

Introducing new connections is always done in the starting of the code if you see in the image some what above the void setup function, you might think why so, if you introduce the connection anywhere else it will show an error, so you must introduce the connection using simple function called-

```
const int ledPin (any connection) = 13 (any digital pin) ;
```

The connection name & digital pin number can be any according to you .

As you saw in the image simply write the above function to introduce connection to any pin, its your wish to decide what should be the connection's name and connected to which digital pin, note no connections needed to be done with negative

terminal of any component, simply connect it with Gnd pin on arduino board. Once you introduce a connection you also have to tell the arduino board that it is a output or input connection, mostly sensor will be classified as input and motors, leds, buzzers will be classified as output, the reason is that arduino board gets input of data from sensor and the data is represented on output devices previously mentioned .To classify the pin as output or input we use a function -

```
void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin (pin introduced before setup), OUTPUT (if output device) / INPUT (if a sensor));

}
```

Write pinMode as shown and in brackets , the pin name you have introduced and " , " and then write whether it is output or input in capital letters.

As shown the information that whether the pin is input or output should be written in void setup as once introduced it will be understood by arduino and code written in setup is run only once. Moving further now we will switch on and off and keep some delays between the code so that the code written after that will run after that given time. we use the following function -

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin (pin name) , HIGH (to switch on));
  delay(1000 (the delay period is to be written in millisecond ) );
  digitalWrite(ledPin (pin name) , LOW (to switch off));
  delay(2000 (the delay period is to be written in millisecond ) );

}
```

Use digitalWrite and delay function in following format -

So this is how we can simply introduce connection, classify as output or input, switch on and off and keep some delays between code in milliseconds.

NOTE : Some important things to note that you can introduce connections using any name as shown in image and then use that name to classify ,switch on and off.

To switch on a pin and switch it off write- digitalWrite((pin name) , HIGH (on) / LOW(off));

Always write a semi colon after each line's ending (;) .

The pin number which you write in the code ,when uploaded on arduino board should be connected to that pin, eg- if I write digital pin 13 and introduce it in connection ,I will have to connect led / buzzer to that

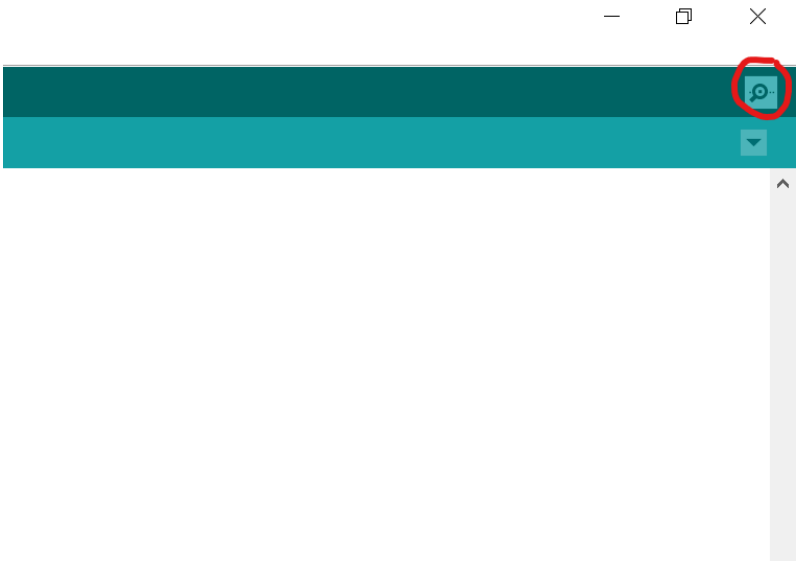
pin .

Try attaching a led to your arduino board and code according to where it is connected and switch it on and off after each second.

CHAPTER THREE

Using the serial monitor as output -

Have you wandered a bit on arduino IDE app if you see the top-right corner you will find a button with image of magnifying image, its called serial monitor which can be used for recieving output from arduino or giving input to arduino using this. In this chapter we will be learning how to use it as output and in the next chapter we will use it as input.



The button is marked by making a circle around it, by clicking on it opens serial monitor.

Whenever we are using a serial monitor as output it's somewhat the same as lcd monitor, so be ready to rock this topic. Lets start by introducing the serial monitor in our code.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

}

void loop() {
```

Start the serial monitor by writing [Serial.begin(9600);]

Once the serial monitor is started its so easy that even a new born child can code the remaining part, you can simply write [Serial.println("what you want to print should be double quotes");], I am also giving you a image as well -

```
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Project tutorial by me !");

}
```

Write- [Serial.println("whatever you want to print should be in double quotes");]

This is all for this chapter, I know that you may be suprised by knowing that this chapter is sooooo small !, but that's the truth.

Note :-

1. *Whenever you want to introduce a serial monitor you will have to do it in setup by writing [Serial.begin(9600);] .*

2.

To print something on serial monitor simply write [Serial.println("whatever you want to print should be double quotes");]

Try using the ultrasonic sensor with the serial monitor and display the distance on serial monitor.

CHAPTER FOUR

How to use a servo motor

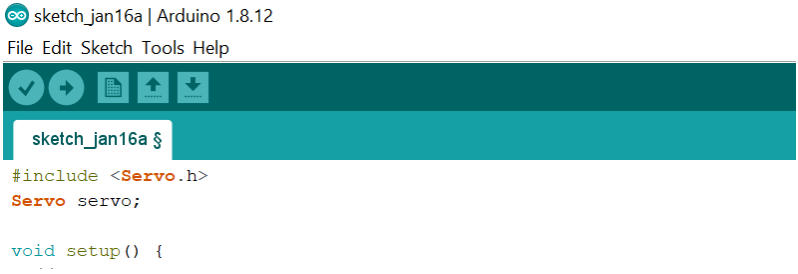
Servo motor is used in many projects and has many advantages such as high torque and acts like arm, this is why most of the coders for robotics do use this for many projects ,this is why we will learn how to learn operating servo motor.

Servo motor looks just like the above image.

The above image shows how the servo motor looks like, it has servo controller along with a brushless motor inside it ,it can rotate accurately upto 180 degrees which make it ideal for arms of robots, knob for indicators etc. Whenever we use a servo its coding is bit different from previous chapters, there are some simple steps to follow -

1. ***Include library from 'sketch > include library > servo' (we have to simply click on servo) ;***

2. ***when you click on servo in include library the library will be automatically added, after that line create a servo object by-***



Simply write library name leave some space and give a name for controlling servo ,which name to give is your wish but name given has to be used to control the respective servo.

3. We will write a function to attach our servo to a digital pin so that we can instruct it to rotate at a angle using it, we write a attach function for this -

```
void setup() {
  // put your setup code here, to run once:
  servo (name of servo you had introduced) .attach(9 [any PWM pin] );

}
```

3. `void loop() {`

Write your servo name then a dot and then attach and in bracket pwm pin.(whichever pwm pin you give you have to attach arduino to it)

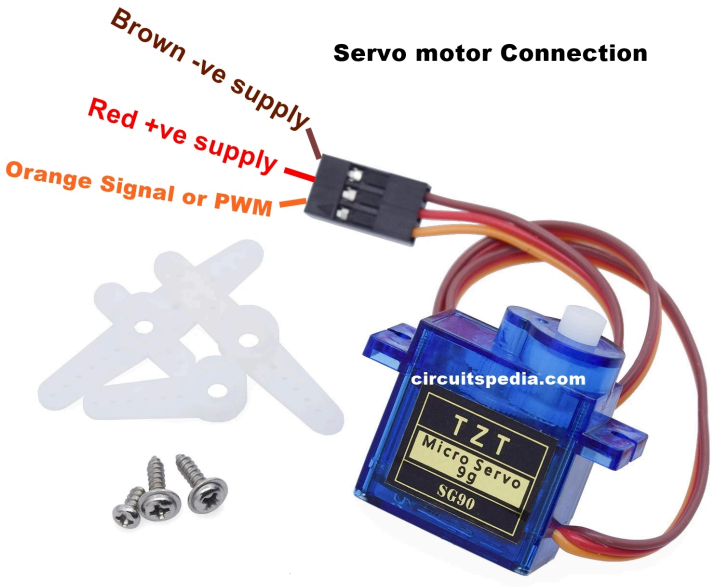
4. the only thing remaining is to tell the angle of rotation to servo motor, we can simply do that by writing "servo's name" .write('angle') -

```
void loop() {
  // put your main code here, to run repeatedly:
  servo.write(180);
  delay(500);

  servo.write(90);
  delay(500);

  servo.write(0);
  delay(500);

}
```

Write the servo's name and then .write(angle), you may add some delays in between and then change the angle in loop function.

Connections of servo motor to arduino -

The coding is done we introduced a servo object, attached it to a PWM pin & moved it at certain angles but what are the connections for this ? the connections are quiet simple -



Servo connection diagram

As per the code the servo should be connected to PWM pin to which servo is attached.

Note:-

1. ***Insert the library by going to "sketch > add libraries" and then click on servo to add servo library before setup.***

2.

Create a servo object by "Servo (name of servo object)" whichever name you give like servo, myservo etc. should be used to attach , write angle, should be written before setup.

3. ***Once you create a servo object attach it to a PWM pin present in digital pin group in setup using ['servo name'.attach('PWM pin number')] .***

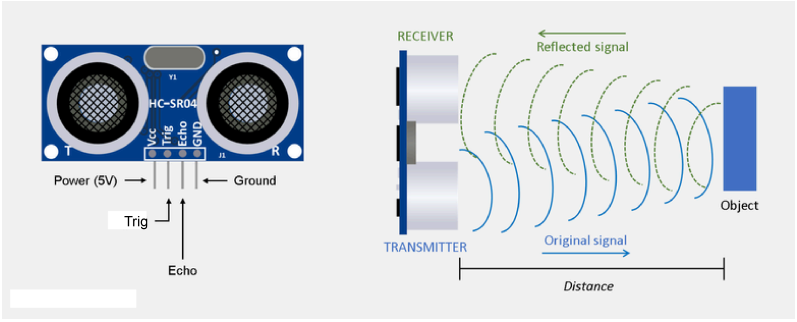
4. ***Turn it to different angles using ["servo name".write('angle')] .***

Try to attach servo motor to pwm pin and then write the code using the given rules and rotate it at different angles.

CHAPTER FIVE

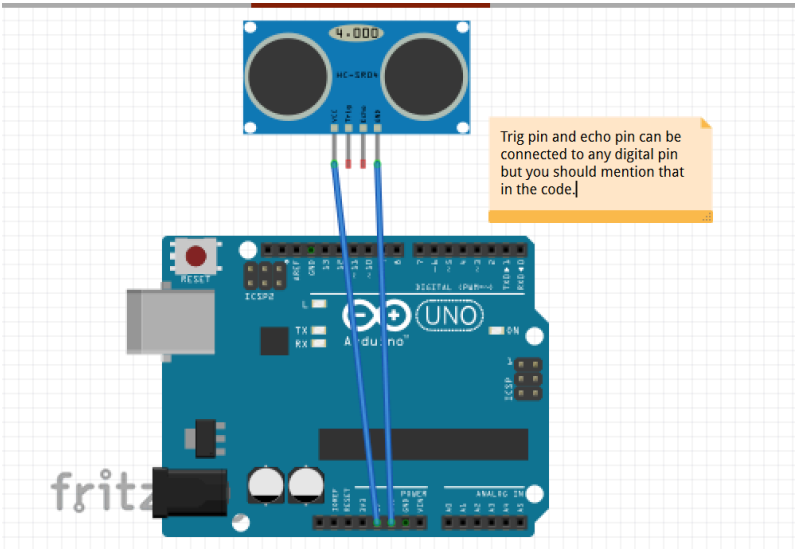
The unique ultrasonic sensor –

Ultrasonic sensor was really a very great discovery of history, do you know why the ultrasonic transmitters are used by doctors, ultrasonic devices are used to find fishes in water by fishermen etc. ,the ultrasonic sensorwhich we use is indeed small but it has great uses for example it can detect a obstacle within a range of 2-400 cm (0.02 - 4m), it can also accurately tell us the distance between it and the obstacle as well. So we are going to learn how to operate ultrasonic sensor, before that we will understand its working it is device which calculates distance or detects obstacle using help of ultrasonic sound, you may be thinking that you have heard this word before, yes you are right ultrasonic sounds are nothing but sounds having frequency more than 20,000 Hz ,this is why we can't hear its sound. But how does it create such high frequency sounds it has 2 drum like things called transmitters and recievers which transmit these ultrasonic sound and are reflected if a object is present and the reciever recieves these sounds. I think that a image would be helpful -



The working of ultrasonic sensor and its pins.

So, working towards programming it we will program for only 2 pins as other 2 pins will be connected to 5v and Gnd pin on arduino board, You have to attach the Vcc pin of ultrasonic sensor to 5v (remember as most of the sensors have these pins) and the Gnd pin of ultrasonic sensor to Gnd pin of arduino board.Then comes the chance of trig pin and echo pin these are basically digita pins, the pin to which trig pin is connected is the pin which will controll the transmitting of ultrasonic waves and pin connected to Echo pin will be used to see whether a object is present or not.



the connection of ultrasonic sensor and arduino should be according to image.

Lets start to code ! ,as to steps we will make connections of trig pin and echo pin and some variables for storing the distance and time taken by ultrasonic waves to be recieved by reciever.

Connection name for trigPin and echoPin is 13, 12 digital pin respectively.

As we have introduced connection for trigPin, echoPin we will also classify them as output / input, this is going to be a bit tricky as we are using a sensor but we will have to write one pin as output and one as input. The trig pin will be output as we are going to be using it for transmitting waves which is a output function and echo pin will be input as we are going to get data from it. We have also introduced a ledPin as we are going to switch it on if we detect a obstacle.

```
void setup() {
  // put your setup code here, to run once:
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
  pinMode(ledPin,OUTPUT)

}

void loop() {
```

The trigPin and ledPin are outputs and echopin is input.

Lets give it some logic to detect a obstacle we have a simple idea for this we will switch on the trigPin for sending some ultrasonic waves and then switch it off then we will count the duration took for recieving waves and from that and sound's speed we will come to know the distance.

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(trigPin,HIGH);
  delay(500);
  digitalWrite(trigPin,LOW);

  duration=pulseIn(echoPin,HIGH);
  distance=duration*0.034/2;

}
```

Logic behind ultrasonic sensor's working.

First of all I know that you are a bit confused why I wrote the above written lines of code, so please read carefully for detecting a obstacle we need transmission and recieving of waves so, I used digitalWrite function for switching on and off the trigPin and then I calculated the time using taken to come back ,by pulseIn which means time taken between the previous function and the present ,then by sound's speed I calculated the formula for distance as [time taken * (speed of sound divided by 10K) /2] you may have understood why I wrote time multiplied by speed but you may be confused with '/2' as this was written as the time taken will be for twice the distance as the waves go and then are recieved after completing the round this was why it was necessary to divide it by 2.

What if we want to switch the led on when obstacle detected and switch it off when not ? For that we will use a if condition it is basically a simple condition , if the information in the curved bracket is correct it will run the code in flower brackets .

```
  if(distance < 30){
    digitalWrite(ledPin,HIGH);
  }
  else{
    digitalWrite(ledPin,LOW);
  }

}
```

I wrote this condition and I wrote the condition as if the distance is lessar than 30cm it will switch on the led and else part so that if the if condition is false the led will switch off.

I used the if condition to switch on the led when the distance was less than 30cm, I wrote this in curved brackets and for switching on I wrote code in flower brackets. For switching off the led when distance is not less than 30 I wrote else part where if the if condition is wrong the code in the else part will be executed.

Note:-

1. ***Connect the vcc or vln pin to 5v on arduino and Gnd pin of ultrasonic sensor to Gnd pin on arduino .***

2. ***The pins you introduce in code for trigPin and echoPin should be used for connecting the ultrasonic sensor to that pins on arduino board, eg- if I gave trigPin = 10 and echoPin = 7 , I will have to connect trig of ultrasonic sensor to 10 and echo of ultrasonic sensor to 7.***

3.

Classify trig as output in setup as it transmits waves which is output and echo as input in setup as gives info. to arduino.

4. ***Switch trig on for sending waves and switch it off after a delay.***

5. ***Use pulseIn function as shown in image in this chapter for calculating the time taken for recieving waves .***

6. ***Use the simple formula of [time taken * (speed of sound /10K) /2]***

7. ***Use the distance found to switch on a led if there is a object in range of particular distance and switch it off if not.***

8. ***Remember the if and else condition if the condition given in curved bracket is true than the code in flower bracket runs, and if you write else and then flower brackets and code in it ,then if the condition in "if condition" is wrong than the code in else condition runs.***

Try making a blindstick by using ultrasonic sensor and buzzer if ultrasonic sensor detects a obstacle then the buzzer creates sound alerting the blind person.

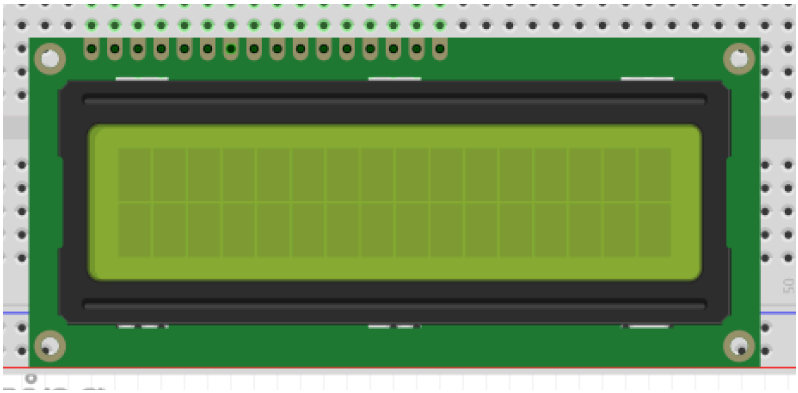
CHAPTER SIX

Using a alphanumeric LCD screen –

Everyone we learnt in the previous chapters about using a servo motor and a ultrasonic sensor ,and that was really amazing to learn to use all these components. We also learnt to use simple switch on /off functions and glowing a led at variable brightness but today we are going to learn to use something different this is called as alphanumeric LCD screen ,this means that you can display alphabets and numbers. There are other lcd screens also available called as graphical lcd screen and TFT lcd screen ,graphical lcd screen means nothing but we can draw shapes on this screen along with alphabets and numbers & TFT lcd screen is best among all as it provides very high resolution and we can draw anything on it however mostly alphanumeric LCD screen is used in most of the projects ,so we are going to learn how to use this lcd screen.

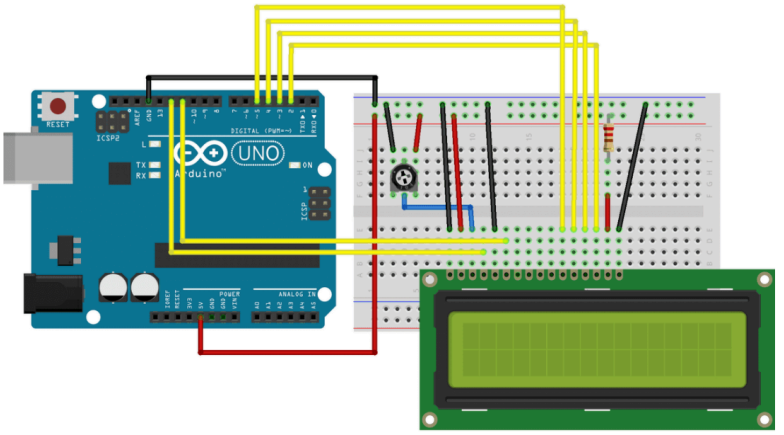
Alphanumeric lcd screen is available in many sizes such as 16*2 , 20*4 & 128*64 characters

(characters are nothing but the number of alphabets / numbers to be shown in one line,row/ column). If you see 20*4 and 128*64 are very expensive and are less popular than 16*2 lcd screen. So after a long process of desciption we will be learning how to use a alphanumeric lcd screen of size 16*2 characters, it looks like the below shown image-



A typical 16*2 alphanumeric lcd screen .

As we know how a lcd screen looks lets connect it to arduino and then we will learn how to code, please note that connections to a lcd screen are most difficult than any other arduino compatible component, so its your wish whether you want to remember the connections or else you can take help of this book-



Here is a fritzing diagram about connecting a lcd screen.

If you aren't familiar with fritzing diagrams then you can refer to the connections given below-

LCD RS pin to digital pin 12
LCD Enable pin (E) to digital pin 11
LCD D4 pin to digital pin 5
LCD D5 pin to digital pin 4
LCD D6 pin to digital pin 3
LCD D7 pin to digital pin 2
LCD R/W pin to GND
LCD VSS pin to GND
LCD VDD pin to 5V
LCD A pin to 5V through a 220 ohm resistor
LCD K pin to GND
LCD VO pin to potentionmeter's middle pin (potentionmeter's right pin to 5v and left pin to Gnd)

As we know the connections now we will start to code, but please remember this important point that the digital pins which we have attached to lcd screen, can't be changed, you can't attach any other digital pin in place of the above given.

Coding steps-

1. *Include the library in arduino IDE by going to "sketch > include library > liquid crystal", for selecting the library you have to simply click on 'liquid crystal'*

2. *As we had created a servo object in previous chapters, here we will have to create a lcd object which will be used to control the lcd screen connected to arduino. (We will have to create a object to control if we are using a library), after adding a object by (any name) we will have to add connections to it to control lcd screen, do you remember the connections I had told you in this chapter ? , the 6 digital pins we had connected will be written here (12,11,5,4,3,2) .*

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

void setup() {
```

Make a object by writing [LiquidCrystal "name"(digital pin numbers)]

3. *After this we will start the lcd screen, have you seen a bike starting we will have to start bike before we drive it right this is why we will have to write a command to start the lcd screen, we will also be writing the size (characters) of lcd screen beside the command to start.*

```
void setup() {
  // put your setup code here, to run once:
  lcd.begin(16,2);

}

void loop() {
```

As we are using 16*2 lcd screen write ["name" .begin(characterSize)]

4. *Are you eagar to print something on lcd screen so lets print ,this is to be done with help of a super easy function called ..... , but before that we will be learning 2 functions ,one to print on lcd and to set cursor. You may be thinking that what do I mean when I say 'set cursor', it is nothing but to adjust where you want to print on the first line or second ? (there are two rows, by default whatever you print will be printed in first row).*

```
void loop() {
  // put your main code here, to run repeatedly:
  lcd.print("project tutorial");

}
```

I have given this image to tell you that if you want to print anything in first row, simply write ["name" .print(whatever you want to write should be in double quotes);]

```
void loop() {
  // put your main code here, to run repea
  lcd.setCursor(0,1);
  lcd.print("project tutorial");

}
```

This is other image in which if you want to write in second row, write ["name".setCursor(0,1)] before ["name".print();]

Wasn't it easy to use a lcd screen, now lets revise our concepts -

Note :-

1. *Connect the lcd to arduino using the image and connections given in this chapter .*

2. *Add library using the simple steps (sketch>add library>LiquidCrystal)*

3. *Create a lcd object the name of object can be anything and connections to it using the following format [LiquidCrystal "name of object"(12,11,5,4,3,2);]*

4. *Start the lcd screen by giving it a command - ["name of object".begin(16,2)]*

5. *Printing in first row needs just a simple function ["name of object".print("whatever you want to write should be in double quotes")]*

6. *Printing in second row needs two functions*

["name of object".setCursor(0,1)];

["name of object".print("whatever you want to print on lcd screen should be double quotes")]

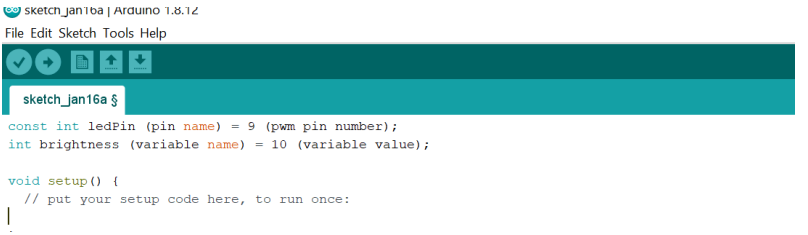
Try using a ultrasonic sensor and lcd screen and display the distance on lcd screen.

CHAPTER SEVEN

What if we want to add some variables ?

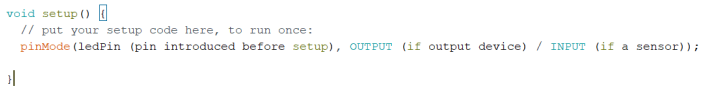
In the previous chapter we learnt how to introduce new connections, classify them as input/ouputs and how to switch them on & off but what if you want to switch on the led or buzzer at medium voltage or other, this is when we use variables, this is the introduction to variables and this would be helpful for you for further concepts. For practising we will use some leds / buzzers for being experienced how to use variables.

As we are starting for instance I would like if you revise your last chapter's concepts of introducing connections, in the similar way we will introduce some variables before setup and switch on a led at varying brightness. To introduce variables we write 'int' simply removing 'const' from 'const int', this makes it easy to remember how to introduce.



We can give any name to variable & connection name.

Once we introduce a connection and a variable with name of your wish and note that as we are going to send a controlled varying voltage to led, we introduced a variable (brightness) with its value given as 10. After some steps we will learn to send varying volts to led using our variable but step by step. Now we will classify it as output.



Use pinMode and in bracket write the connection name and then write as output in capital words.

Now in loop function we will write to switch on the led at varying voltage but remember we are using a special pin called PWM(pulse width modulation) pin as this pin can send varying voltage, you can find this pin in digital pins' section with a pwm sign so whenever you want to send a varying voltage you have to introduce connection with that pin this

is why I used pin 9 as it is a PWM pin. Please note down this function as this is a very important function-

```
void loop() {
  brightness=10;
  analogWrite(ledPin,brightness);
  delay(1000);

  brightness=5;
  analogWrite(ledPin,brightness);
  delay(1000);

  brightness=0;
  analogWrite(ledPin,brightness);
  delay(1000);

}
```

We use a new function called analogWrite instead of digitalWrite as analogWrite function can send variable voltage, we also change the value of brightness as this helps us to indirectly change voltage sent to led .

I hope you understood how to introduce variables for sending variable voltage using analogWrite.

Note:

1. ***Introduce a variable before setup like connection ,use 'int' with a variable name as your wish and a value minimum of 10 ;***

2. ***Use analogWrite function for this purpose and in brackets write connection name and variable ;***

3. ***Remember to use a PWM pin as we are sending variable volts to led, you can find these pins in digital pins group.***

4. ***If you change the variable value the brightness will also change, so you can simply change the variable's value to less value when you want to dim the led and increase value of variable to higher value when you want to increase the brightness of led.***

Try making a fading led project using concepts like 'int', 'analogWrite' etc. at home using arduino.

CHAPTER EIGHT

Using the serial monitor as input -

In the last chapter we learnt how to use serial monitor as output using just two steps, starting the serial monitor and then printing any values or words in double quotes. Here we would learn how we can use serial monitor as input ,wherein we can send signals or messages to arduino.

When we open the serial monitor we will find a input section on top and a send button beside this is what we are going to use today, we will be typing some input and then sending it to arduino board and we will program the board according to that. Before starting to code I want to tell that we will be using some leds and according to input given through serial monitor the leds will switch on. You can experiment using a servo and then giving some angles to it using serial monitor.

Again for sending input to arduino we will have to start the serial monitor as done in previous chapter,

we will also have to introduce some new connections with leds and classify them as output -

```
const int led1=13;
const int led2=12;
const int led3=11;
const int led4=10;

void setup() {
  // put your setup code here, to run once:
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  Serial.begin(9600);

}
```

We introduced connections and classifed them as output, we also started the serial monitor.

As we have introduced some connections ,classified them and started the serial monitor, we will be learning how to type code for arduino for reading the data we have sent and that's quite easy -

```
void loop() {
  // put your main code here, to run repeatedly:
  int data=Serial.read();

}
```

We simply introduced a variable data using ‘int’ and stored serial data - [Serial.read();] .

Once we have made a variable for reading the serial data we will also add some ‘if conditions’ that if the input we give is this than it will it will do this and so on, do you remember the ‘if condition’, Ok no problem I will recall what we had learnt in 5th chapter. A ‘if condition’ is nothing but a simple function, in which if the condition written curved brackets is true / correct than it runs the code in flower brackets.

```
if(  /*condition you want to give*/  ){

  /*  whatever you write in flower brackets will
  run if the condition given in curved
  brackets is true / correct */

}
```

The format of writing a if condition.

I hope you are clear about the if condition, so now I am going to tell you how arduino will switch on led according to input you give.

```
if(data==='1'){
  digitalWrite(led1,HIGH);
  digitalWrite(led2,LOW);
  digitalWrite(led3,LOW);
  digitalWrite(led4,LOW);

}

if(data==='2'){
  digitalWrite(led1,LOW);
  digitalWrite(led2,HIGH);
  digitalWrite(led3,LOW);
  digitalWrite(led4,LOW);

}
```

I have given 4 if conditions like if data = 1 than 1rst led will switch on and remaining will switch off , if data = 2 than 2nd led will switch on and remaining will switch off , and so on till 4th led.

I hope you are clear about using a serial monitor as input and giving arduino some data and programming it to work according to data.

Note :-

1. *For using a serial monitor as input the first step is to start it using - [Serial.begin(9600);] .*

2. *To read data from serial monitor introduce a variable - [int data = Serial.read();] .*

3. *Use the data to make arduino work using if condition, write the if condition and inside that write the variable for [Serial.read();] , for instance - if the data === '1' then the first led will switch on and rest will switch off etc.*

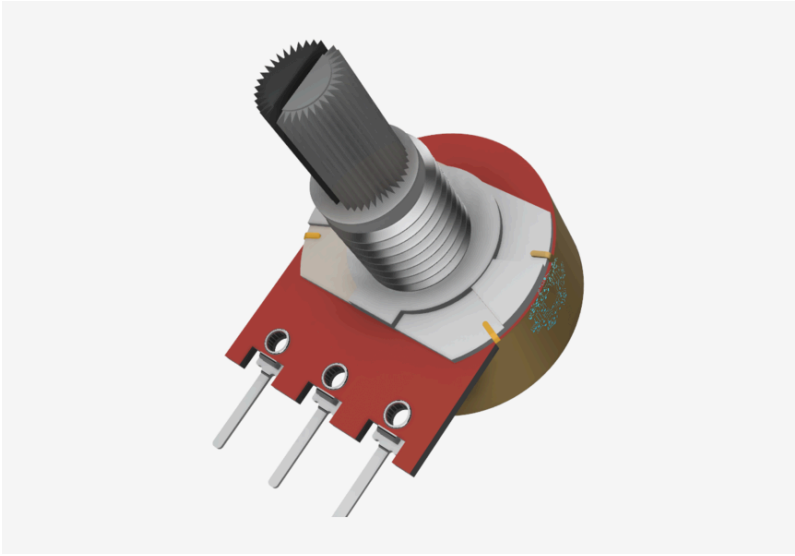
Try using serial monitor as input and rotate the servo according to serial data collected .

CHAPTER NINE

What exactly is potentiometer ?

Potentiometer is nothing but a device which can provide variable resistance according to its knob turned. You can also say it as a group of resistors or 'analogWrite' function as well. It can be used to turn on a led at varying brightness/ buzzer at varying shrillness etc. just like the analogWrite function which we had learnt in chapter 3. The advantage here is that we can turn on led, buzzer etc. at variable brightness without coding so, this is going to be a very easy lesson for us today.

Lets understand the structure of potentiometer first, it has three pins and it looks just like our leg if we see it from a certain angle.



This is how the potentiometer looks like.

I gave you a image for reference how a potentiometer looks like, please remember that

- *Right pin from front side is to be connected to 5v ;*
- *Middle pin from front side can be connected to the device to which you want to send varying volt ;*
- *Left pin from front side is to be connected to Gnd pin .*

Once you have remembered the connections for potentiometer, lets connect it to a buzzer and led to experiment this, its quite simple, you have to simply attach the potentiometer to arduino as told above and then you can attach the middle pin of potentiometer to led / buzzer and then connect the negative terminal of arduino to Gnd.

What we have learnt about the potentiometer is for toddlers so, lets try to make something more interesting, how about controlling a servo's angle using potentiometer, its interesting isn't it so lets build it, but before that do you remember how to use a servo motor ?, I am sure you may have forgotten some important points , so go to chapter 4 for revising the concepts tought in that chapter . Once using a servo motor is clear lets add the logic part as we can't simply connect the potentiometer's middle pin to servo motor's pwm pin. According to coding rules we will include library and classify a servo object.

```
#include <Servo.h>
Servo servo;
```

Add the servo library and a servo object.

Once the servo object is created ,lets attach it to a pwm pin (it's present in the digital pins group) .

```
void setup() {
  // put your setup code here, to run once:
  servo.attach(11);

}
```

Attach the servo to a pwm pin.

Once the servo's pwm pin is attached to a pwm pin lets turn it according to potentiometer, for this we will be using analogRead function and we will be connecting the potentiometers middle pin to analog pin 0.

```
void loop() {
  // put your main code here, to run repeatedly:
  int value=analogRead(A0);
  int ang=(value/5.683);
  servo.write(ang);


}
```

Logic behind turning the servo according to potentiometer.

I think that the logic isn't quite clear, so don't worry I will tell you the logic behind this, first we introduce a variable using ' int ' to store the value from the A0 pin, then we use the value got to change

it to a angle between 0 - 180 degrees for this we will simply introduce another variable to convert the output to angle from 0 - 180 degrees and then we divide the ‘ value / 5.683 ’ (we get a maximum value of 1023 from a analog pin so we will have to divide it by 5.683 to make it 180 degrees which is maximum angle at which a servo can be turned) . I hope the doubts are clear or you can also refer to the image as well.

Note : -

1. *There are three connections of potentiometer -*

- Right pin from front side to 5v ;
- Middle pin to device where you want to send variable voltage ;
- Left pin from front side to Gnd .

2. We can control a servo using a series of functions -

- ‘ int val = analogRead(" analog pin number "); ’ , this is for reading the analog pin 0 which is connected to potentiometers middle pin and storing the value in a variable .
- ‘int ang = val/5.683 ’ , we will change the ‘val’ recieved by arduino from potentiometer to angle .
- ‘ servo.write(ang); ’ the angle got from changing the ‘val’ is used for turning the servo’s knob .

CHAPTER TEN

Using a DHT11 sensor -

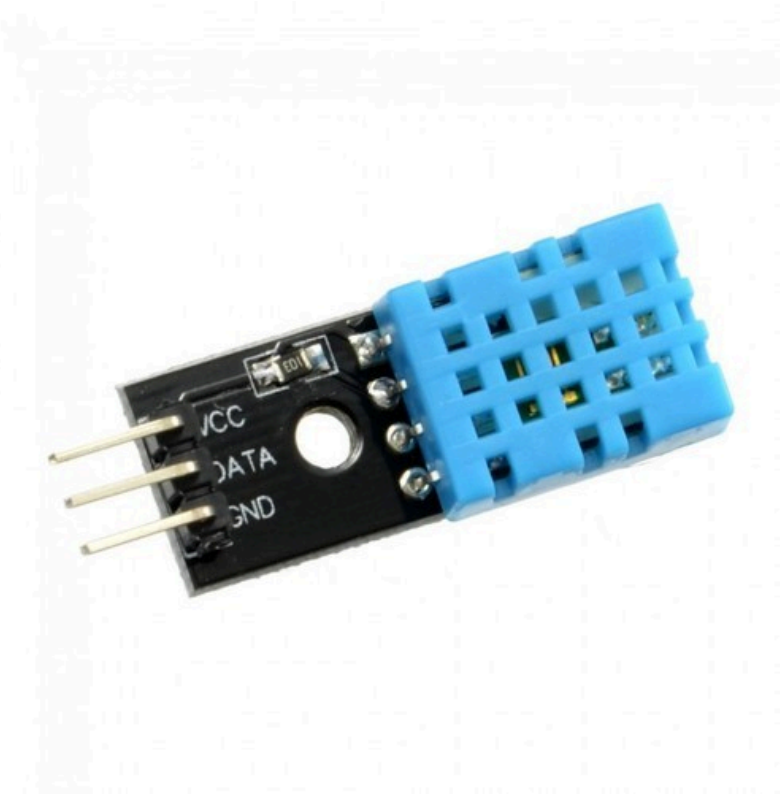
In this chapter we will be using a DHT11 temperature and humidity sensor, there are many arduino compatible temperature sensors in the market such as DS18B20 temperature sensor, digital temperature sensor etc. however after some experimentation I saw that DHT11 sensor is much more accurate and compact sensor than other, and another thing like a pro about this sensor is that it not only shows the temperature but also shows the humidity as well.

Using a DHT11 sensor is quite easy to use, code and has wide range too ! , lets discuss the physical properties or making of DHT11 sensor (please note that we are using 3 pin DHT11 sensor as its more efficient than the 4 pin DHT11 sensor) , The 3 pin DHT11 sensor has 3 pins -

- *DHT Gnd is to be connected to Gnd pin ;*
-

DHT data is to be connected to a digital pin which you assign in the code ;

- *DHT Vcc is to be connected to 5v pin .*

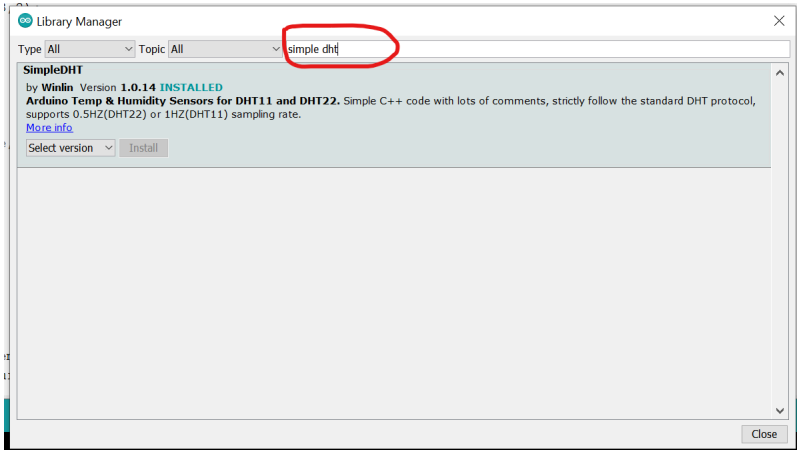


DHT11 sensor's pins and body.

I have given image for reference of dht11 sensor, I would also briefly explain what's inside the blue capsule like thing . The blue capsule like thing is

also the container of a thermistor and capacitive humidity resistor, its other role is to protect the inner componenets from dust, damage etc . , whenever we use a DHT11 sensor it we use its library to read the temperature and humidity of environment . Its coding is bit tricky and difficult but if you read the book thorougly then you can easily program DHT11 sensor . So , lets start coding .

As I told you we are going to add library but this library will not be present in the arduino IDE app , you will have to download it from library manager , you will have to go to ' sketch > include library > manage library ' , in manage library you will have to search for simple dht in search option -



Search for simple dht and then install it to use it.

2nd step is to add the library in code and create a dht11 object -

```
#include <SimpleDHT.h>
SimpleDHT11 dht11;
```

Add the library after downloading library by going to sketch > include library > simpledht

Create a variable for storing the values from sensor and create a connection to which data pin of dht11 sensor will be attached -

```
const int DHT11pin=7;
int tmp;
```

a connection for attaching the dht11 sensor and a variable for storing the values .

Once this is done we will start to write functions in loop to find temperature and humidity -

```
void loop() {
  // put your main code here, to run repeatedly:
  byte temperature;
  byte humidity;

  tmp=dht11.read(dht11Pin,&temperature,&humidity,NULL);

  Serial.println((int)temperature);
  Serial.println((int)humidity);
  delay(2000);

}
```

Writing this code in arduino IDE will enable reading the output

.

I would tell you why we have written the above written lines of code , first of all we created the memory storing units using (byte temperature , byte humidity) for storing temperature and humidity , once we introduce a storage unit we will also add a function as shown above to read the dht11 object and pin attached to it .

I hope you understood everything as I tried my level best to tell you about sensor .

Note : -

1. Add the library by " sketch > include library > SimpleDHT "

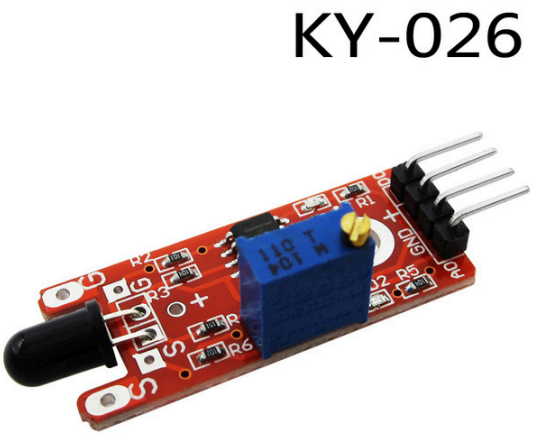
2. Create a dht11 object - [SimpleDHT11 'object name']

3. Understand the simple logic behind the working and type it in the code .

CHAPTER ELEVEN

Using a flame sensor with arduino -

Have you ever seen movies in which there are some compact devices atttached at the ceiling of room which detects if there is fire or not , today we will be building a similar kind of compact device which can detect flame / fire nearby , for this purpose we will be using a arduino compatible flame sensor , I am giving you a image for reference for you , you can use this image to understand the sensor better .



Flame sensor also called KY-026

As we know how a flame sensor looks like you can buy it or if you already have it then its ok . A flame sensor has 4 pins outof which we can use 3 pins , I will also tell you which pins are necessary to be connected to arduino -

- *+ pin to 5v pin on arduino ;*
- *Gnd pin to Gnd pin on arduino ;*
-

Analog output (AO) pin to A0 pin on arduino .

We know which pin is to be connected to which pin on arduino so lets start coding but before that I would tell you how does this flame sensor detects flame ? , this flame sensor has a phototransistor attached on it which is used for detecting light coming from flame , you may be thinking that it won't be useful as it will detect light and if its kept in dark place it won't work but its to true . The phototransistor attached on sensor has specific range for detecting only light coming only from flame sensor and this is done as it can detect light only of 760nm of wavelength which is of light produced by flames .

As we know the connections and working lets code arduino for this purpose , its going to be very easy for doing this as we know these concepts are learnt in previous concepts as well , so don't be too tensed .

1. ***Create a connection for sensor's analog pin to read the output from sensor please remember that we will be using a anaolg pin so while introducing connection write any analog pin name and also create a variable for storing the values from sensor . This is quite easy use ' const int ' and ' int ' ;***

```
const int sensorPin = A0 ;
int sensorValue;

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

Create a connection and a variable for storing sensor values .

2. ***Once connection is created we will classify it as input as we are going to connect it to sensor and then we will start the the serial monitor for printing whether fire is detected or not ;***

```
void setup() {
  // put your setup code here, to run once:
  pinMode(sensorPin,INPUT);
  Serial.begin(9600);

}

void loop() {
```

Classify connections and start serial monitor .

3. ***Here we will be using ‘ analogRead ’ function for reading sensor’s values accaording that we will print on serial monitor whether fire is there or not . Here we will use trial and error technic as we don’t know what is the analog output if fire is detected or if not . First we will read the output from sensor from when fire is not present then we will note that in notebook or paper , then we will bring it near flame not too much and then we will read the output and note it , according to change***

we will write a if condition to print in the serial monitor that ‘ fire is detected ’ when value of the sensor is near the output we had noted when we kept flame near sensor .

```
void loop() {
  // put your main code here, to run repeatedly:
  sensorValue = analogRead(sensorPin);

  if(sensorValue<100){
    Serial.println("Fire is detected, please contact officials");

  }else{
    Serial.println("No fire detected, detecting further .... ");
  }

}
```

Final logic .....

I hope you understood the third step , you will require to try sometimes to find a correct if condition as we need to find the output difference when flame was near the sensor and when not .

Note : -

- *The majorly used connection with sensor are with Vcc - 5v , Gnd - Gnd and AO - Analog pin 0 ;*
-

We will have to create a connection for sensor and then variable for storing the sensor's output ;

- *We will classify the sensor pin as input and start the serial monitor for printing whether fire is detected or not ;*

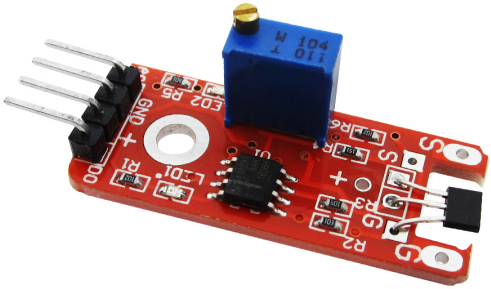
- *We add the final logic of code , we first use the variable made by us to store the sensor value using ' analogRead ' function , then we will be using trial error method and write a if condition , its quite simple to do so , we will read the output from sensor and print it on serial monitor when the flame is near the sensor and when not . Seeing the output we can find the difference using that we will write a if condition .*

CHAPTER TWELVE

How to detect magnets -

Magnets either natural (lodestone) or man made , but have you ever wondered of making something with arduino which can detect magnets ? , you may of thought of it , so lets bring it reality . Today we will be using linear hall magnet sensor which is pretty good at detecting magnets and we will be using this sensor and this will be pretty simple as we have learned how to read analog outputs from a sensor , today's topic is similar to last chapter's topic just we will have to conduct some trial - error situation and find correct difference . Lets first see how the sensor looks like -

KY-024



This is how a linear hall magnet sensor looks like .

Lets also discuss how will we connect the sensor to arduino , as we already know that we are using the sensor's analog pin for reading the sensor values so we will be connecting -

- *sensor's + pin to 5v ;*
- *sensor's Gnd to Gnd pin ;*
- *sensor's AO pin to any analog pin on arduino but whichever pin you are connection should be same in*

code .

We already know the steps we will introduce some connections write whether it is input or output , create a variable for storing the sensor's analog output and then after some trial - error conditions we will write if conditions. So lets start coding -

1. *Lets first create a connection and variable for storing sensor values -*

```
const int sensor = A3;
int value;

void setup() {
```

Introducion of connection and variable .

2. *Classifying the sensor as input and starting the serial monitor is to be done in this step -*

```
void setup() {
  // put your setup code here, to run once:
  pinMode(sensor,INPUT);
  Serial.begin(9600);

}

void loop() {
```

Classification .

3. *In the loop function we will be first reading the sensor pin and storing the value in variable and once done we will print it on serial monitor using some simple functions , we will do a trial error situation here for knowing the sensor's output when the magnet is near the sensor and when not , then we will see the difference when magnet is near the sensor and when its not . On coming to know the difference we will create a if condition .*

```
void loop() {
  // put your main code here, to run repeatedly:
  value=analogRead(sensor);

  if(value>250){
    Serial.println("magnet found");
  }

  if(value<250){
    Serial.println("no magnets nearby");
  }

}
```

Function in void loop .

I hope the trial - error situation is clear , it may be bit confusing but you must master it as is used in many sensors , you may note this in your note book you will have to read the output from sensor and then note when there is a magnet and when its not , there will be difference when magnet is present and when not so according to the difference we will make a if condition .

For instance , when a magnet is present the value from sensor might be 270 - 275 and when there is no magnet nearby the value may be 200 - 210 so according to this we can write a if condition that if sensor value is greater than 250 then it will be printed on serial monitor that ‘ magnet detected ’ and if sensor value is lessar than 250 then it will be printed on serial monitor that ‘ no magntes nearby ’ .

I hope its crystal clear how to use a linear effect magnet sensor .

Note : -

- *While using linear hall efffect sensor connections are said in the chapter and we use analog output of sensor , so we use the AO (analog output) of sensor .*

- *We create connection and make a variable for storing sensor's value ;*

- *We classify the connection as input and start the serial monitor ;*

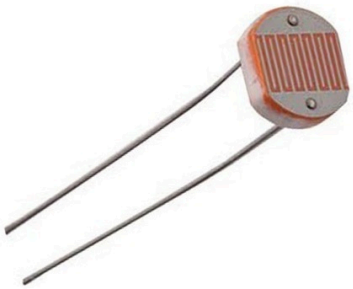
- *We add final logic after a trial and error situation when we read the output of sensor when magnet is near and when not .*

You may make a magnet detector using the sensor and a buzzer , switch on the buzzer when the magnet detects magnets .

CHAPTER THIRTEEN

Detection of light , the medium of sight -

In chapter 12 we had learnt how to use a flame sensor , I also told you about its working , what it comprises of ? and more ... I wrote that it has a phototransistor for detecting flame's light but what if we want detect normal light , you may be thinking what's its need but what about the blind people and not only that we can also make smart lights using this . Connections are pretty simple , It has two pins longer pin is to be connected to 5v and shorter pin should be connected to any analog pin for reading the intensity of light .



A photoresist

Now we are knowing the connections as well so lets make smart lights which will switch on when its dark and switch off when the sun's up ! , its going to be pretty simple we will be using a led , photoresistor . We will be connecting the photoresistor's longer pin to 5v and other to Analog pin for reading the lights intensity , and the led to a digital pin . So , lets start coding ,

```
const int led=13;
const int sensor=A1;

int sensorValue;

void setup() {
```

Introduction of connection for led, sensor and a variable for storing the sensor values .

As each time in our code we will introduce a connection for our photoresistor and led , we will also introduce a variable for storing sensor's values , please remember that we are using a analog pin for sensor so we will write ' A1 ' (analog pin 1) . Lets also classify our pins .

```
void setup() {
  // put your setup code here, to run once:
  pinMode(led,OUTPUT);
  pinMode(sensor,INPUT);

}

void loop() {
```

Classify the led connection pin as output and sensor pin as input .

We classify the sensor pin as input as we are receiving values from sensor and we will write led as output as led emits light which is output .

Lets also write our final code in loop function , don't be tensed as you may be thinking that we are again using trial - error situation but not in this chapter as we know one thing that more the brightness the sensor's value will be more and when

less brightness is present then the sensor's value will be less , so we will simply use a average value found by me which is ' 200 ' .

```
void loop() {
  // put your main code here, to run repeatedly:
  sensorValue=analogRead(sensor);

  if(sensorValue<200){
    digitalWrite(led,HIGH);

  }
  if(sensorValue>200){
    digitalWrite(led,LOW);

  }

}
```

Write a if condition for switching on led when ' sensorValue ' is lessar than 200 and switch led off when it is more than 200 .

I hope you understood the logic , the brightness is what increases or decreases the sensor's value so according to that we will be switching on and off the led . Please note that more the brightness more will be the ' sensorValue ' so the value is directly dependent on brightness . We want to switch on the led when the brightness is less , so we will write that if brightness is less than a small value such as ' 100 - 200 ' than the led will switch on .

Note : -

- *Connections of photoresistor with arduino are very simple we will have to connect the longer terminal of photoresistor to 5v and shorter one to Analog pin for reading the brightness .*

- *Remember that brightness is what decides what will be the sensor's value , ie , sensor's value is directly dependent on brightness .*

- *As we want to switch on led when brightness is low we will have to write that if the brightness is less than a low sensor value than the led will switch on .*

CHAPTER FOURTEEN

Using a Joystick module :-

In chapter 9 we had learnt how to use a potentiometer to control the brightness of led and how to control a servo's angle but using it was bit difficult lets use a joystick module which alone is capable of controlling two servos which is really a very big thing as it makes a project more compact if you want to control two servo's .

We can simply call it as a two in one potentiometer as it has two pins from where we can know how much the joystick module's knob is turned either left - or - right / up - or - down , we will be learning how to use it for controlling our robotic arm . Isn't that interesting to make our own robotic arm . So , lets first understand the connections of joystick module to our arduino board .

This is how a joystick module looks like its the same which is present in remotes of rc cars .

Lets also discuss the connections done with joystick module , you might have guessed that two of the conection will be to 5v , Gnd & that's right ! .

- *Joystick Gnd to arduino Gnd pin ;*
-

Joystick 5v to arduino 5v ;

- *Joystick VRX to arduino Analog pin 0 ;*
- *Joystick VRY to arduino Analog pin 1 .*

Its easy to memorize the these connections as 2 connections are from - to same pin and then the remaining two pins , VRX and VRY are to give us data about how much the joystick module's knob is turned , so they are connected to analog pins to read data from it . Lets start to code joystick module , its pretty simple as if we are programming two potentiometers .

In this chapter we will be controlling two servo's with joystick module so we will be first creating two servo objects , two connections with analog pins to connect joystick's VRx and VRy pins , we will also create some variables for storing the values read from these analog pins , but we will also have to create some more variables for changing the values from sensor to angles .

```
#include <Servo.h>
Servo servo1;
Servo servo2;

const int Vrx = A0;
int VrxValue;
int ang1;

const int Vry = A1;
int VryValue;
int ang2;

void setup() {
```

Add the servo library & create two servo objects . Create two analog connections so that we can connect our joystick module's Vrx , Vry pin to it .

In setup we will also have to attach our servo motors to pwm digital pins and classify the analog pin connections as input so that we can read the values from sensor .

```
void setup() {
  // put your setup code here, to run once:
  pinMode(Vrx,INPUT);
  pinMode(Vry,INPUT);

  servo1.attach(11);
  servo2.attach(9);

}

void loop() {
```

Classify the analog connections as input and attach our two servo's to pwm digital pins .

In loop function we will be reading output from the sensor and storing it in variables we had created before setup , these values will be changed to angle for turning the knob of servo motor , we know already that any analog input from pin can be max - 1023 , min - 0 , so we will be simply dividing the analog valus in two angles for our two servo's . We will have to create another two variables for doing this like - ' ang1 ' , ' ang2 ' .

```
void loop() {
  // put your main code here, to run repeatedly:
  VrxValue=analogRead(Vrx);
  ang1=VrxValue/5.683;
  servo1.write(ang1);

  VryValue=analogRead(Vry);
  ang2=VryValue/5.683;
  servo2.write(ang2);

}
```

The final logic behind our joystick controlled servo motors .

I know that the connections , logic etc . isn't that clear so I will simply elobarate why we write the above written sketch .

- *First of all we will create two servo connections as we are going to control two servo's using joystick , then we will be creating two connections from analog pins to Vrx and Vry pins of joystick , we will also have to create two variables for reading*

the values from both the pins , however we will also have to create two more variables for changing the sensor's values into angles according to which our servo's knob can be turned .

- *In setup its pretty simple to write code as we will have to simply classify the connections to analog pins as input and attach the servos to two pwm digital pins .*

- *In the loop function its pretty simple to write code as we know the steps , we will be first reading sensor's values from both the pin and storing the sensor's value in the variables created before setup and then we will be changing these values into angles by dividing them by ' 5.683 ' and storing the angle in variables created ' ang1 ' & ' ang2 ' . Finally we will be turning the servo using the variables in which we had stored the angle .*

Try making you own robotic arm using the concepts learnt today and some cardboard parts as well . You may also use potentiometers in place of joystick module .

CHAPTER FIFTEEN

Controlling geared motors

–

Using geared motors which is also popularly known as bo motors . We can't simply use normal motors as they can't hold that much of weight and their speed will also not be constant . The other thing is that the motor will be over heated . This is why we do use bo motors instead of normal motors .

BO motors also have their own problems such as their rpm is quiet lessar than normal motors like -

normal motor's rpm = 5,000 +

vs

bo motor's rpm = 60 - 300

they also need more voltage than normal motor's -

normal motor's required volt. = 3v

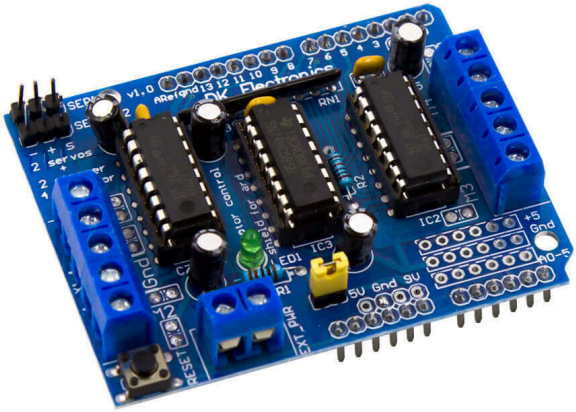
vs

bo motor's required volt. = 9v

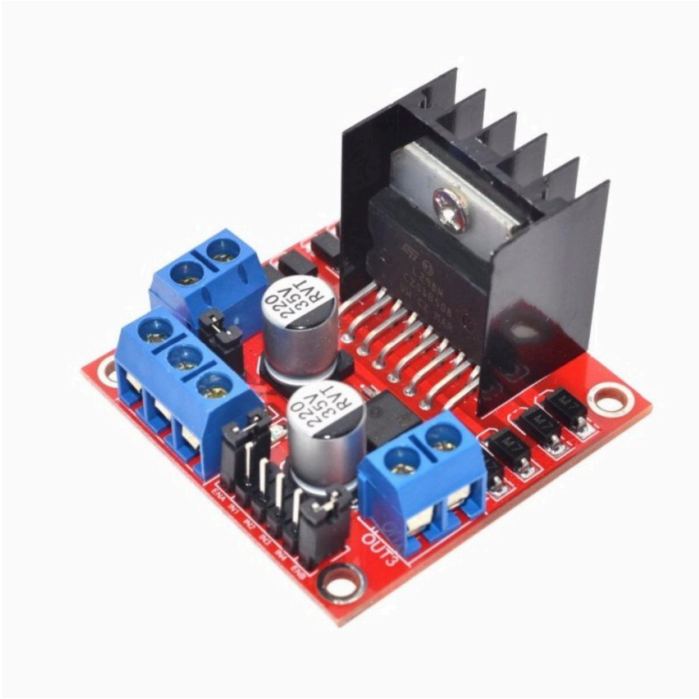
(minimum voltage)

To overcome these problems we need to use a motor driver as it can easily supply 9v to our bo motors , we can't connect arduino directly to them as arduino can supply only 5v at max which is lessar than voltage required by bo motors .

I am going to show how do these motor driver and motor driver shield looks like -



L293D motor driver shield .



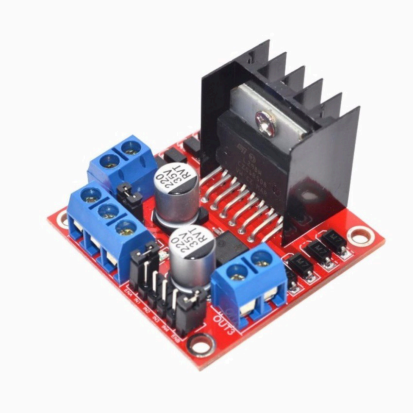
L298N motor driver .

I have simply given you brief idea how do these motor driver and motor shield looks like . In the coming chapter we will be mastering them for controling our robotic car .

CHAPTER SIXTEEN

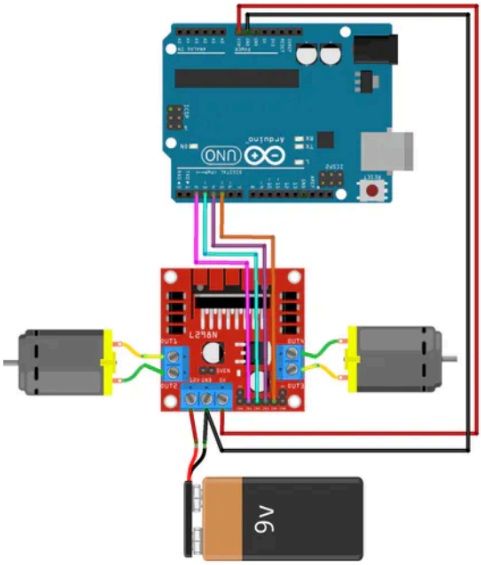
Lets drive motors with L298N motor driver !

As introduced in previous chapter we had learnt about geared motors and we found that we need motor driver or motor driver shield for this purpose , in this chapter we will be learning how to use L298N motor driver which is quite small and efficient at its work .



This is how L298N motor driver looks like , its as small as an eraser !

We will first understand how we connect arduino to it its quite simple , indeed . There are 4 input pins (in1 , in2 , in3 , in4) which are used to control 4 outputs (out1 , out2 , out3 , out4) respectively , Its pretty simple if you connect 4 input pins to digital pins then making a pin high will send electricity to that output pin . One more thing to note is that we will also have to connect a 9v battery to motor driver , the following images show the above things -



Connections -

As shown in above picture we have connected -

1. 12v pin on l298n to + pin of 9v battery .

2. Gnd pin on l298n to - pin of 9v battery .

3. in1 , in2 , in3 , in4 to digital pins on arduino board .

4. out1 , out2 , out3 , out4 to 2 motor's pins .

We will be trying to understand in detail how can we control out1 , out2 , out3 , out4 using in1 , in2 .... ;

First of all whenever we make any of the input pin high then the respective output pin will supply 9v to gearbox and when we make any of the output pin low then the respective pin will become gnd pin or - negative pin .

Some rules we must remember while coding for l298n are that we can't simply set both pins of motor to 9v , one should be 9v and other should be gnd , if in1 is high then in2 should be low and if in2 is high then in1 should be low , same for in3 and in4 . When trying to make robotic car make sure that both wheel move in same direction .

Lets start coding for our motor driver -

As usual we will start by introducing connections for in1 , in2 , in3 , in4 for controlling out1 , out2 , out3 , out4 -

```
const int in1=7; //I am connecting inputs to digital
const int in2=6; //pins 7,6,5,4 , you can use any pins
const int in3=5; //but the pins you give here should
const int in4=4; //be used to connect motors to them .

void setup() {
```

Please read instructions , I have made 4 connections to input pins which we will be using to control motor . You can digital pins as per your mind but note to connect input pins to the pins in code .

We will classify them as output as we are going to turn them high and low to change the directions of motors -

```
void setup() {
  // put your setup code here, to run once:
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);
  pinMode(in4,OUTPUT);

}

void loop() {
```

Classify them as output .

Here comes the final code in loop function , it as simple as switching on and off a led so don't bother about it :)

```
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);

  delay(1500);

  digitalWrite(in2,HIGH);
  digitalWrite(in1,LOW);
  digitalWrite(in4,HIGH);
  digitalWrite(in3,LOW);

  delay(1500);

}
```

We will turn the motor at a direction and then reverse .

I hope you understood why we write the above lines of code , if not I will explain but before that please read this book , especially this chapter properly in full concentration and enthusiasm as this chapter needs practice and logic to understand why we write the following lines of code . Lets now discuss why we write the following lines of code , first of all we will make in1 , in3 high and in2 , in4 low to make the motor rotate , I hope you remember why we write in1 high and in2 low , in3 high and in4 low so that one of output will supply 9v and another which is low will become gnd (- negative) . Then we will keep a delay of 1.5 seconds and then we will rotate motors in opposite direction by making in2 high and in1 low , in4 high and in3 low , then again we will keep delay of 1.5 seconds after which the code will again run from

start .

You might be bit confused but I have tried my level best so , you can practice this for 2 - 3 times which will make you clear how to use L298N motor driver . There are just some points which you must remember while coding for L298N -

1. The input pins (in1 , in2 , in3 , in4) are used to control output pins (out1 , out2 , out3 , out4) , for example ; Attaching in1 , in2 , in3 , in4 to digital pins (13 , 12 , 11 , 10) respectively will control the outputs as switching on a digital pin will affect as - eg , in1 is connected to digital pin 6 then turning pin 6 high will result to making out1 supply 9v to the motor .

2. Whenever a motor is connected to driver then one of the outputs out of two should be high and one should be low for eg , in1 is high then in2 should be low , in1 is low then in2 should be high ; in3 is high then in4 should be low , in3 is low then in4 should be high . This is because ' any electric component to work it must should be connected to one + pin (3.3 volt , 5 volt , 9 volts etc .) and other to - pin (gnd) .

CHAPTER SEVENTEEN

Drive motors with L293D

We had seen in the previous chapter that how are bo motors drived by l298n today we will learning to drive the motors using l293d motor driver shield .

l293d motor driver shield is quite good to use in some projects in which we are not bothered about compactness of project , when we compare l298n and l293d we find many differences , some of the most imposrtant differences you must know are that -

- *l298n is very small compared to l293d , l298n is as small as an eraser and l293d is as big as arduino board , so it takes lots of space .*
-

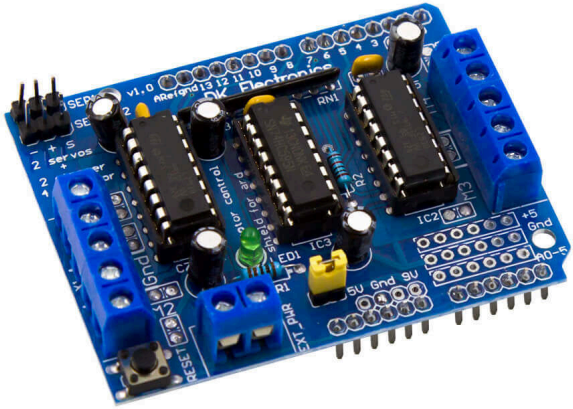
l298n can control just 2 motors , however l293d can control 4 bo motors , 2 servos , 2 stepper motors which makes it worth it size .

- *l298n heats up much faster than l293d , its really cool !*

- *coding for l298n is very difficult then l293d , it also requires more logic to program l298n .*

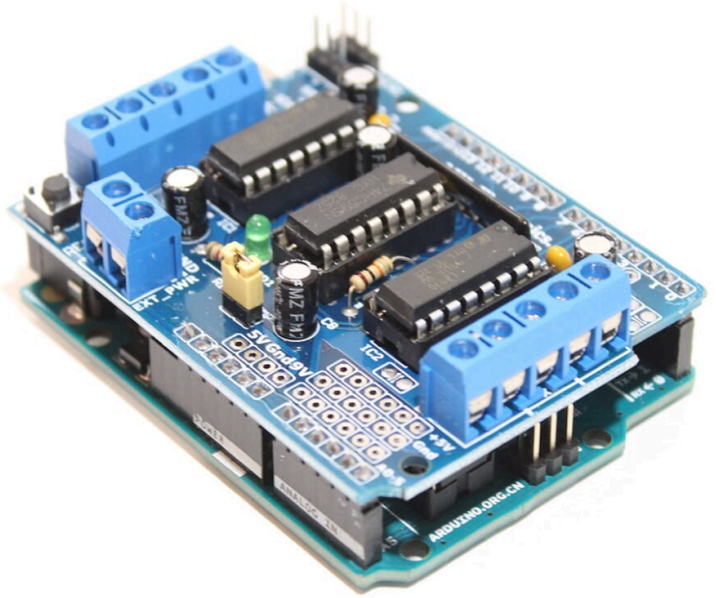
According to above comparison its clear that l293d is much more better to use than l298n , however you can use l298n as well .

Lets see and understand how to use l293d motor driver shield , lets first understand the physical features and connections for our master driver l293d motor driver shield -



L293D motor driver shield .

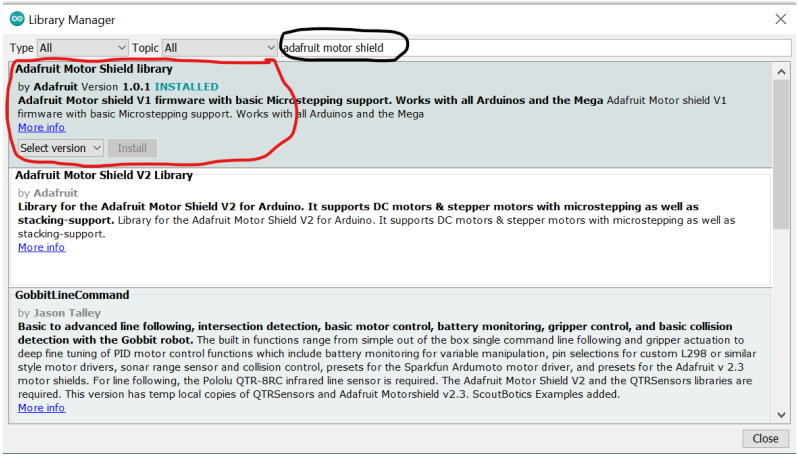
As we see l293d is a motor driver shield we clearly understand the word shield means that it is attached over the programming board which is arduino . I hope its clear that l293d is motor driver shield which is attached over arduino board , the following image makes the connections clear to you -



As you can see it is attached over arduino in following manner .

As the connections are clear lets see how its coded but before that lets understand how to connect our geared motors to it , there are some sections on l293d which are used to connect to our bo motors . There are 4 such sections to which bo motors can be connected they are called - m1 , m2 , m3 , m4 , while writing code we will have to classify which port are we using m1 , m2 , m3 or m4 . We have to simply connect both of our motor's pins to the section we are using .

Lets code it up step by step , today's coding will be bit different as we are not going to write code in void setup . We will be writing code in only before setup for introduction of connection and in void loop . We will also have to add library in our code , I am sharing how to add the library -



Library manager .

Simply go to sketch > include library > manage libraries > search for what is marked in black in above image > download the library marked in red , once you have done this the library will be added to your list as ' Adafruit motor shield library ' .

The second step is to go to sketch > include library > click on ' Adafruit motor shield library ' to add it in your code , once added we will create a motor object and connect it to one of the sections on l293d (m1

, m2 , m3 , m4) , we will write 1 to connect it to m1 slot , 2 to connect it to m2 slot and so on ...

```
#include <AFMotor.h>
AF_DCMotor motor(1);

void setup() {
```

Include the library , create the motor object and connect it to any of one slot present on l293d (I have connected to slot 1)

I will explain why we wrote the above code - firstly we add the library using the simple steps which I had told to you (sketch > include library > click on ' Adafruit motor shield control ') and then we create a motor object usinf the following format - ' AF_DCMotor motor (1) ' you can keep the object name accaording to you and whichever slot you write eg : - 1 (m1) , 2 (m2) , 3 (m3) , 4 (m4) .

Lets now add code in setup , as we are using a library it will be very simple for us to code it up ! in setup we are only going to set the speed of motor from 0 - 255 -

```
void setup() {
  // put your setup code here, to run once:
  motor.setSpeed(200);

}
```

I have set the speed as 200 you can set it anywhere from 0 - 255 .

Once the speed is set we will run the motor forward , backward and stop .

```
void loop() {
  // put your main code here, to run repeatedly:
  motor.run(FORWARD);
  delay(2500);
  motor.run(BACKWARD);
  delay(2500);
  motor.run(RELEASE);
  delay(2500);

}
```

We use the name we had used to create the motor object , I had used ‘ motor ’ then we write ‘. run ’ and in brackets how should motor run .

As in the above image its crystal clear how to run the motor , use the motor object name and write ‘ .run ’ and in brackets write whether it should run forward , backward or should stop (release) .

Note :-

•

To connect arduino to l293d you should simply attach l293d over arduino .

- *l293d has 4 slots for controlling our bo motors (m1 , m2 , m3 , m4) , the port we select in the code should be used while we connect our motor with l293d , eg ; I select 2 in the code then I will have to connect bo motor to m2 slot , if I select 3 in the code then I will have to connect bo motor to m3 slot etc.*

- *We first download the library as told in the chapter and then we add the library in code as ‘ #include<AFMotor.h> ’*

- *We create a motor object , whichever name we chose should be used further in code to control our motor . We also classify to which slot is it connected to l293d by writing in brackets besides the motor object as (1) to connect out bo motor to m1 so on till m4 .*

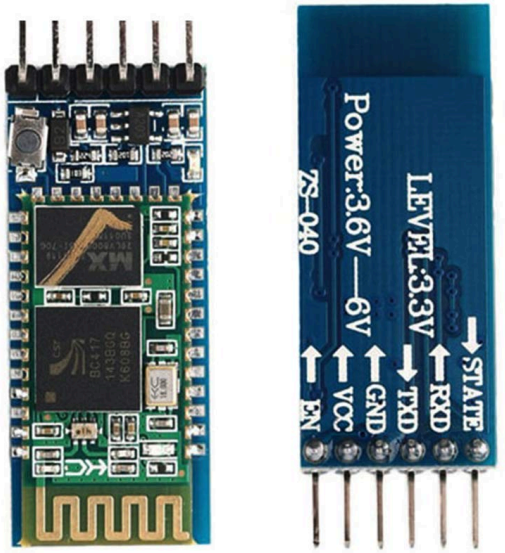
- *Setting the speed of motor is done in setup using motor object’s name and ‘ .setSpeed ’ and in brackets the speed from 0 to 255 [‘ motor object name ’ .setSpeed (0-255)]*

- *In loop we write to move the motor forward , backward or stoppin it as told in the chapter .*

CHAPTER EIGHTEEN

Making something smart using Bluetooth !

This is going to be very exciting chapter as we are going to going to make wirelessly controlled robot which can be controlled using smartphone , isn't that cool . So lets start our chapter with some simple introduction to bluetooth module and how it communicates with our smartphone -



HC-05 bluetooth module .

In the above image I have shown how a bluetooth module looks like , in the image I have shown the front and rear view of the sensor to understand it better , in this chapter I have given instructions to use a HC-05 bluetooth module clearly and I am also going to share the app for our smartphone which will be used by us to control our today's project , but before that lets se how bluetooth module HC-05 sensor works , starting with connections there are 6 pins on bluetooth module out of which only 4 are used to transmit and recieve signals , the center 4

pins of bluetooth module are connected to arduino as follows -

- *Bluetooth module pin Gnd to Gnd pin on arduino ;*
- *Bluetooth module pin Vcc to 5v pin on arduino ;*
- *Bluetooth module pin TXD to RX pin on arduino ;*
- *Bluetooth module pin RXD to TX pin on arduino .*

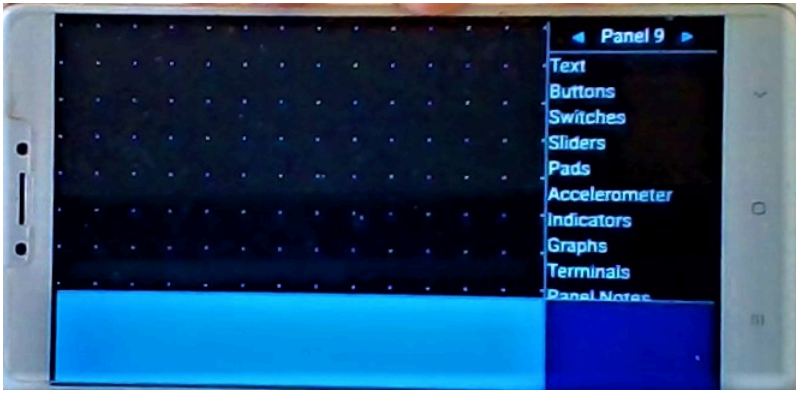
I hope the connections are clear they are pretty simple and these connection are not needed to introduce in our code , we have to upload code to arduino and simply connect bluetooth module to arduino . When we connect Bluetooth module to arduino , a serial communication starts between our smartphone and bluetooth module , which is used for transferring and recieving data from smartphone to bluetooth module , so what's " serial communication " ? , this question might be buzzing in your mind , but there's nothing much to know about it . It's simply a type of communiction in which the data is transferred from our smartphone to arduino through serial monitor (that's not the exact definition but to make clear) .

This is why while writing code we will be reading serial values or values sent from smartphone through serial monitor , according to that we will be writing conditions . In this chapter for brief introduction we will be learning how to switch on and off led lamp just from a tap on your smart phone , so lets start !

We will be first downloading the app for sending data to bluetooth module , I have used " Bluetooth Electronics " app for sending data to bluetooth module , you can download this app from google or playstore . Once downloaded open the app but please switch on your bluetooth as we are going to require it for establishing connection , then there are multiple panels , you have to simply open a empty panel -

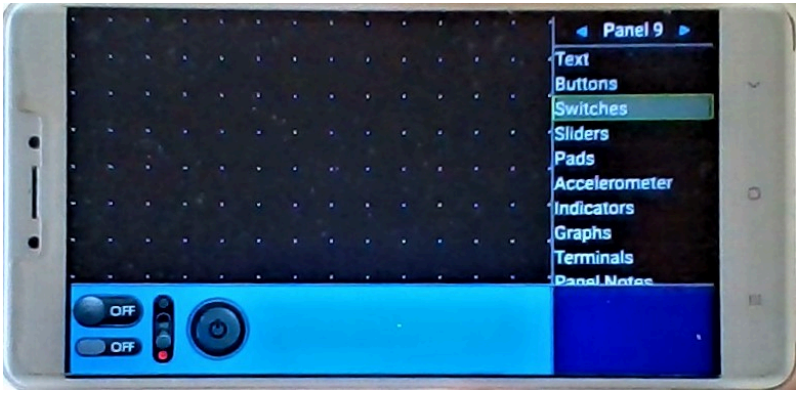
The above image shows how the app looks like after opening app .

After opening the app open a empty panel -



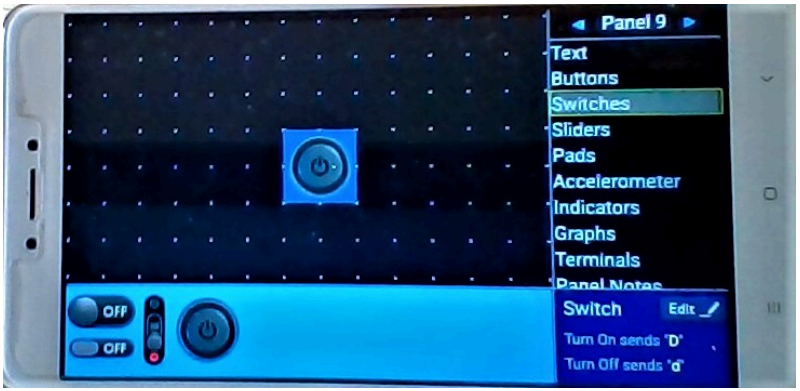
An empty panel , you can open ‘ panel 9 ’ as its mostly empty .

Once opened go to switches as in the top right corner , click on it to open it and the following options will appear on bottom of app -



Click on the switches option to get the above options .

You can select any one of the switch from below and drag it to panel (dotted area) to enable it and use it . I am bringing the button type switch as given below -



In the image above I have shown which switch I am using .

As in the image I am using a button type switch . If you take a look at the bottom right corner on pressing the switch I have selected it shows me its features and working , it displays that whenever I run the app and press the switch to make it on it will send " D " and when I press it again to switch off it will send " d " . This is what matters as these are the values which our bluetooth module will be recieving on switching on the switch and when switching off . According to this we will be writing code for our

arduino board . Lets begin what are we waiting for ?

First of all like each time we will be introducing connection for our led before setup .

```
const int ledPin=10;

void setup() {
```

Introduce the connection for our led .

Once introducing connection is over we will also classify it as output and start the serial monitor , which we will be using for creating a serial communication with help of bluetooth module .

```
void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin,OUTPUT);
  Serial.begin(9600);

}

void loop() {
```

Our piece of sketch in setup .

Now its chance for writing code in loop / final / main code for our project , I will be first showing what we are writing and then I explain the logic to you .

```
void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()>0){
    int data=Serial.read();

    switch(data){
      case 'D':digitalWrite(ledPin,HIGH);
      break;
      case 'd':digitalWrite(ledPin,LOW);
      break;

    }

  }

}
```

Enter Caption

As I told let me tell you why we write the above shown code , first of all we check whether our bluetooth module connected to any device or not using a if condition and ‘ Serial . available() ’ . Then we introduce a storage unit for storing values from our smartphone , using variable int , ‘ data ’ and [Serial . read();] , serial.read is nothing but values read from serial communication . And now comes the new and toughest part which is the " switch case " , we haven’t used it before so it might be bit

confusing so , let me make it clear to you , whenever using switch case its very much like if condition and some things are similar too . like you have to give the variables name which you want to use in curved brackets and then give the flower brackets and the conditions inside it , like above given image and if the variables value is equal to the value in the hyphen then the code after the colon will run . If you look with bit concentration you will find the values 'D' and 'd' which are going to be sent by our smartphone when switch on and off .

As we have completed writing code lets test our first project with bluetooth module , for that we will have to go to the app click on the connect button > bluetooth classic > discover till we see HC-05 > select HC-05 and pair .

Once paired go back to main page of app and there you have to select the panel in which we had added button and then click run . Once you click on run the switch will appear and then simply click on the switch button to switch on the led click again on it again to switch off led .

Thank You Readers !

It was my dream of having a book with my own name on it , its everyone's dream and the day I published this it was accomplished , thank you readers for buying this book and giving me immense satisfaction and joy . I hope this book was helpful to you , if you find need for any improvements or want some more books please write in comments . You can also go to the following youtube channel for more and amazing arduino projects -

https://www.youtube.com/channel/
UCvzUMpZFM8wyFQIDD_b3TPQ