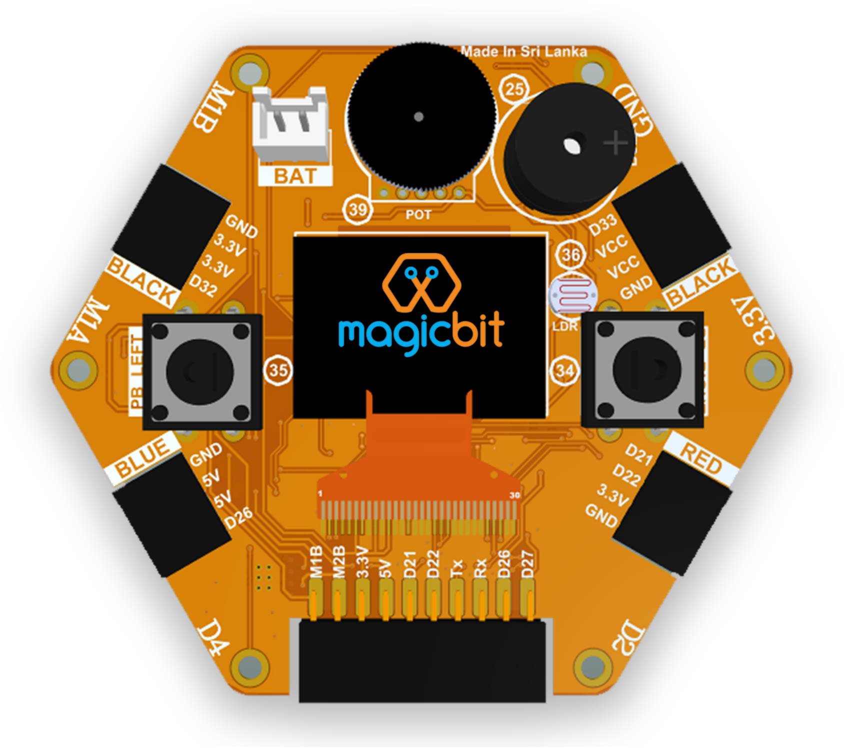
Magicbit-Arduino Documentation

Release latest

Oct 03, 2020

Contents



This device is an Integrated development platform for learning and solution designing of electronics, robotics, Internet of Things and coding. The device can interact with a host of sensors such as light sensors, fire sensors, motion detectors etc. and output devices such as LEDs, switches, buzzers, speakers, motors etc. The hardware can be programmed using industry standard integrated development environment (IDE) that runs on PCs, Tablets, Mobile devices, or web platforms. This device supports a large number of programming platforms such as C++, python, scratch, magicblocks, mblock and codda, Hence the learning curve to learn to operate and utilize this device is shorter. Ardunio & MicroPython are for users with prior programming experience. Magicbit provides extension for mBlock3 for kids without programming knowledge. Codda is a visual programming language which same time can experience the true coding. Magicblocks is a NodeRed based platform for IoT solution design for any user group.

This device also has the following special features:

- In-built battery charger, WiFi & Bluetooth connectivity;
- Integrated sensors and actuators to enable users to test and design projects without additional components;
- An internal OLED display;
- Plug & play feature to easily connect accessories;
- An enclosure for productization of designs

Brain of the Magicbit is ESP32, which is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. Therefore any project or document available on internet which supports ESP32 is supported for Magicbit as well.

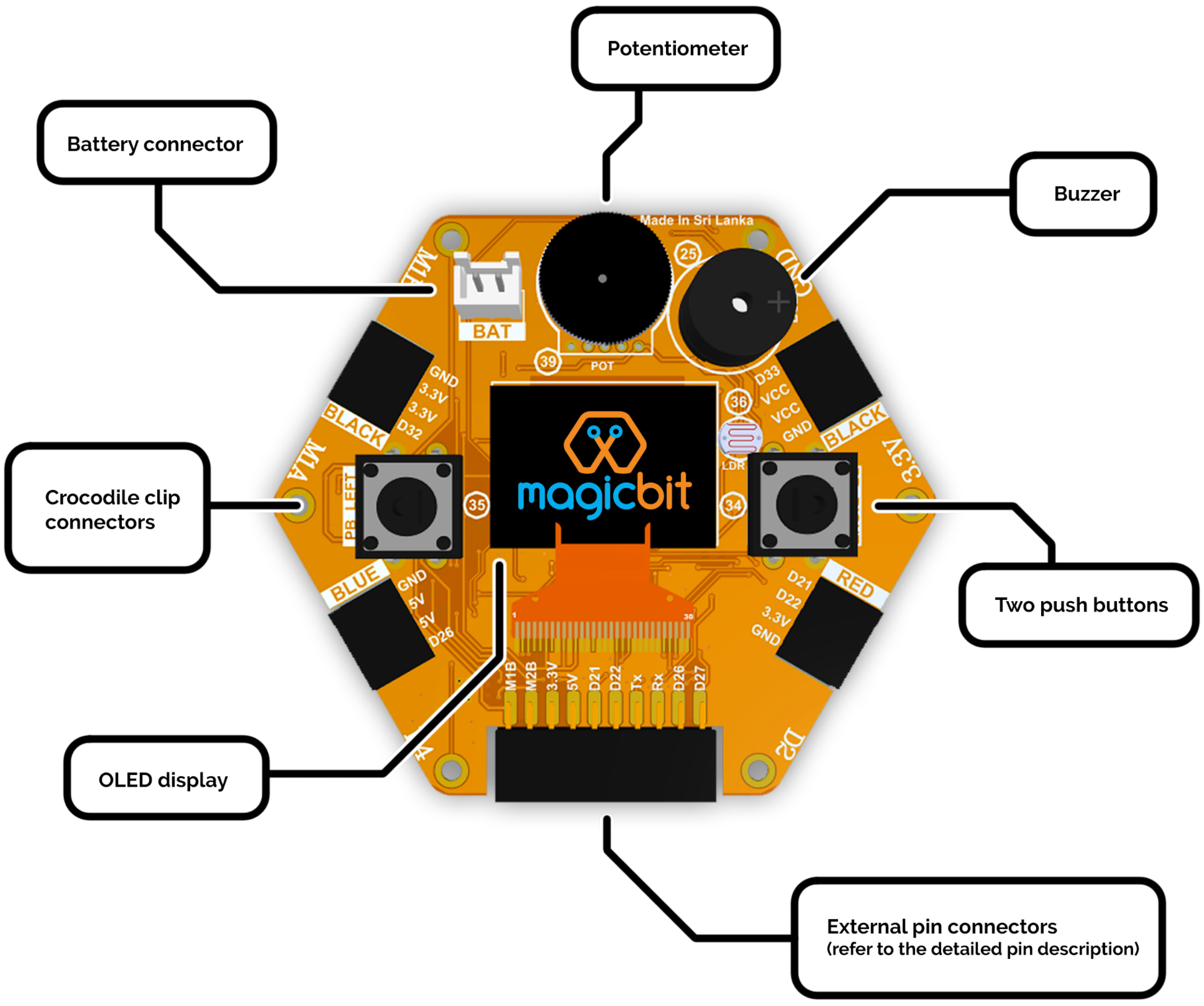
CHAPTER 1

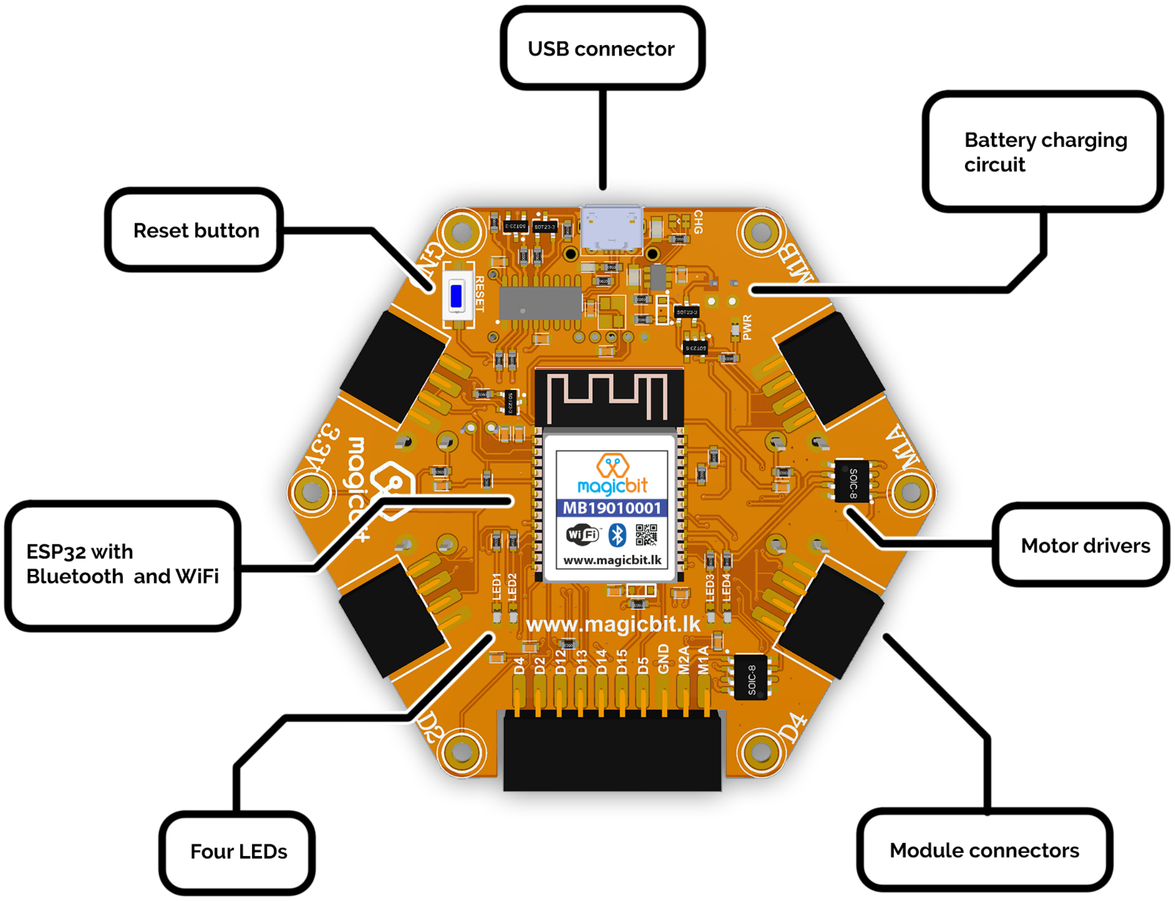
Hardware

1.1 Specifications

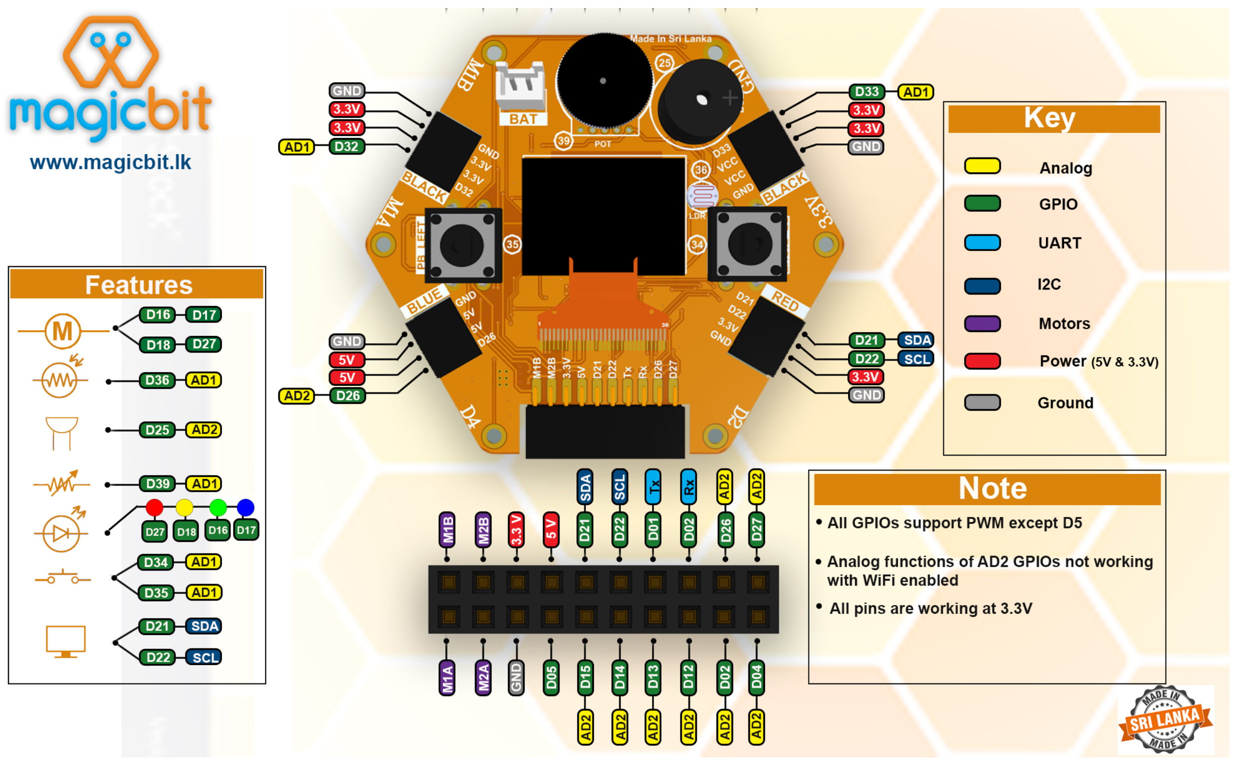
- **Processor** - Xtensa dual-core
- **Speed**- Up to 240Mhz
- **Flash Memory**-4MB
- **Ram**-520KB
- **Inputs**-Pushbutton, LDR, Potentiometer
- **Outputs**-LEDs, OLED Display, Buzzer
- **Other**- Dual Motor Driver, Li-Ion Charger
- **Connectivity**- USB, WiFi, Bluetooth

1.2 Layout


Potentiometer
Battery connector
Buzzer
Crocodile clip connectors
Two push buttons
OLED display
External pin connectors
(refer to the detailed pin description)
magicbit
Made In Sri Lanka
BAT
POT
LDR
M1B
M2B
3.3V
5V
D21
D22
Tx
Rx
D26
D27


USB connector
Battery charging circuit
Reset button
ESP32 with Bluetooth and WiFi
Motor drivers
Four LEDs
Module connectors

1.3 Pinmap



1.4 Features

1.4.1 LED

There are four leds on backside of the Magicbit with color red, yellow, green & blue. A LED(light-emitting diode) is a semiconductor light source that emits light when current flows through it. Blinking a LED is the hello world to the microcontroller programming world.

1.4.2 BUTTON

There are two buttons on the front of the Magicbit. The push-button is a component that connects two points in a circuit when you press it. The example turns on an LED when you press the button.

1.4.3 LDR

There is a LDR on the front of the Magicbit. LDR(Light Dependent Resistor) is a light-controlled variable resistor. The resistance of a photo-resistor decreases with increasing incident light intensity. You can measure light intensity using LDR as a analog output.

1.4.4 POTENTIOMETER

The potentiometer is a component with rotating contact that forms an adjustable voltage divider. A potentiometer is a simple knob that provides a variable resistance, which we can read into the Magicbit board as an analog value.

1.4.5 DISPLAY

OLED (Organic Light Emitting Diodes) is a flat light emitting technology. OLED display has a film of organic compound that emits light in response to an electric current.You can display varoius graphics and text on the display.

1.4.6 BUZZER

There is a buzzer on the front of the Magicbit. Buzzer is an electronic device commonly used to produce sound.

1.4.7 BATTERY

There is a Battery connector on the front of the Magicbit.Single cell rechargeable li-ion battery (3.7V) can be plugged in to a battery connector to puwer the Magicbit. Battery can be recharged by providing USb power to the Magicbit.

1.4.8 MODULES

There are four module connectors on the edge connector of the Magicbit, which we refer to as ports. Which can connect various accessories to Magicbit board and program to work with Magicbit. Matching accessory pin connector color marked on the Magicbit. As an example module with blue pin connector should plug in to blue port of the Magicbit.

1.4.9 USB

There is a micro USB port on the back of the Magicbit.Connect the micro USB port to a mobile phone charger or computer through a cable and it will draw power required for the board to function and it also used program magibit and data transferring with a computer.

1.4.10 WiFi

WiFi is a technology that uses radio waves to provide network connectivity. Magicbit consists with wifi module. WiFi technology has widely spread lately and you can get connected almost anywhere; at home, at work, in libraries, schools, airports, hotels and even in some restaurants enabling IOT connectivity capabilities.

1.4.11 BLUETOOTH

Bluetooth is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using short-wavelength UHF radio waves. Magicbit consists with wifi module which enables IOT connectivity capabilities

1.4.12 EXPANSION HEADER

Magicbit can connect various electronic sensors, electronically controlled actuators,etc to Magicbit via these external connectors

1.4.13 CROCODILE CLIP

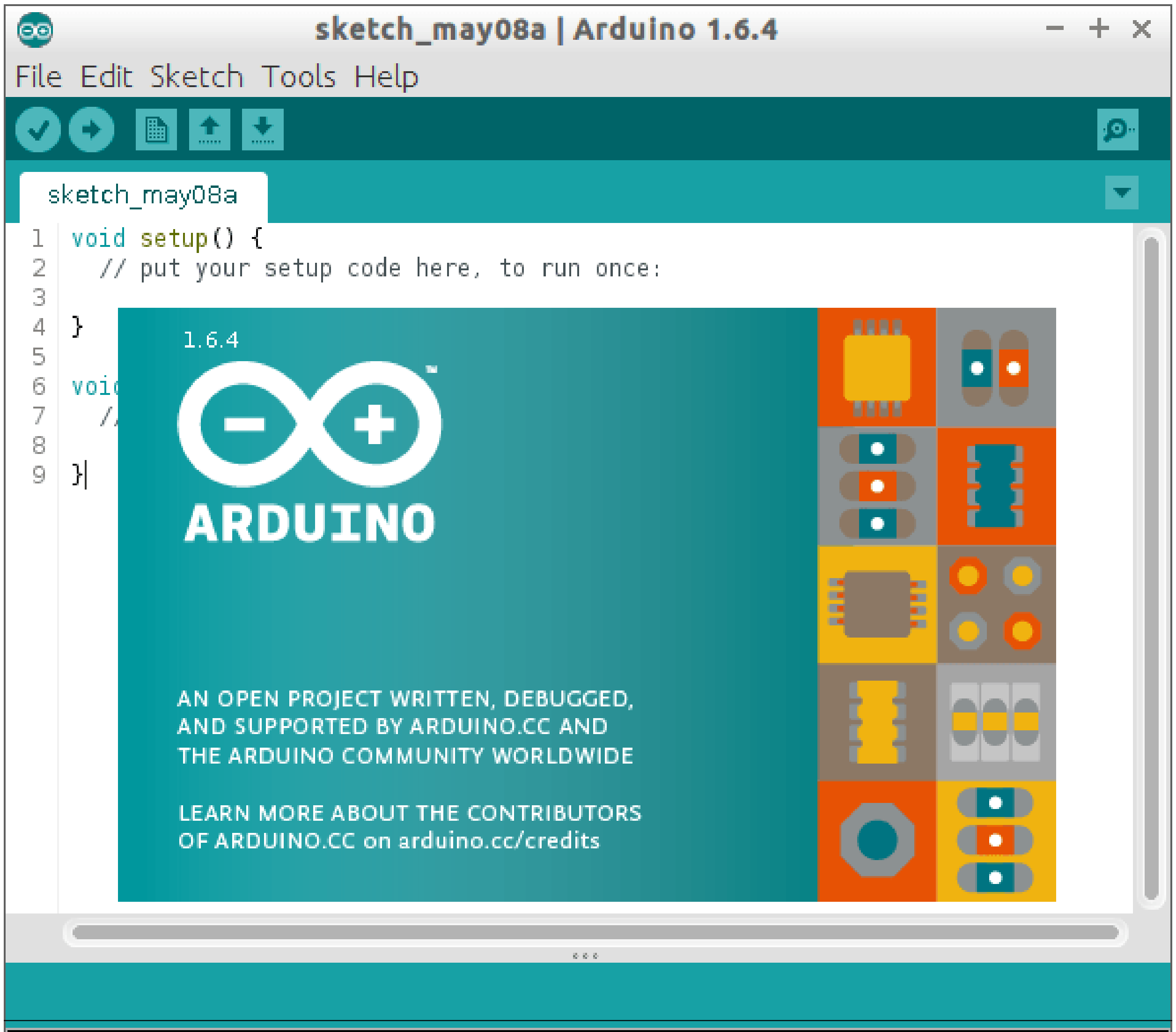
Magicbit crocodile clip connectors used to connect an electrical cable to a battery or some other component. Functioning much like a spring-loaded clothespin, the clip's tapered, serrated jaws are forced together by a spring to grip an object

1.4.14 RESET BUTTON

In electronics and technology, a reset button is a button that can reset a device. On Magicbit, the reset button restarts the Magicbit's programme

CHAPTER 2

Getting Started



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

}

voi
  /

}
```

sketch_may08a | Arduino 1.6.4
File Edit Sketch Tools Help
sketch_may08a
1.6.4
ARDUINO
AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE
LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.Arduino IDE is not only used for programme Arduino boards but also other development boards like esp32, magicbita dn so on. Let's get some general idea about Arduino IDE preview. Below picture shows the startup preview of the Arduino IDE. In this at the middle you have your text editor and at the top left side there is a menu bar which have several options

Let's look at options in the Arduino menu bar.

- File- As the name suggests you can do the documentary things with this option. You can open new sketch, close, save that sketch, open existing sketch or example sketch and change system preferences.
- Edit- With this you can change your font size, indentations and some writing options
- Sketch- This is important as compilation, uploading and library options are available in this.
- Tools- Hardware setup is done in this option. You can change the board type, programmer type and connecting port name with this and configure other hardware settings.
- Help- This option is used for connecting with Arduino references and getting help.

Below the menu bar you have some shortcut keys (command keys). First one is verify key button. By using this you can know your code is grammarly correct or not according to programming language rules. Using the upload button you can do both the compile and upload process at same time. The right top corner have serial monitor icon. This is very important option. This is used to communicate between Arduino IDE and your development board. You can send data to and watch the data which is sent by the development board from this. Under the serial monitor icon we can see some small icon options. This option is used to set the configurations of the new tabs and existing tabs. At the bottom side you can see the debugging console. This shows the result of the compilation and uploading and some other messages which are very useful for debugging.

In the text editor space we can write any Arduino program. This is our playground. We can write any related code in that space. Here are two sections. One is void setup and the other one is void loop. In the void setup we write which we want to execute at the start and one time. So we write pin configurations and other library configurations in the void setup. Our main algorithm is in the void loop. As its name suggests the processor execute this loop section repeatedly at every time. So we write the things which we want to execute at every time in this section. After writing the code go to tools->select your board type and select your port name correctly. Then click the upload button to upload the code.

Learn more about Arduino <https://www.arduino.cc/en/main/software>

Magicbit is based on ESP32 and Arduino core for the Magicbit forked from the espressif/arduino-esp32

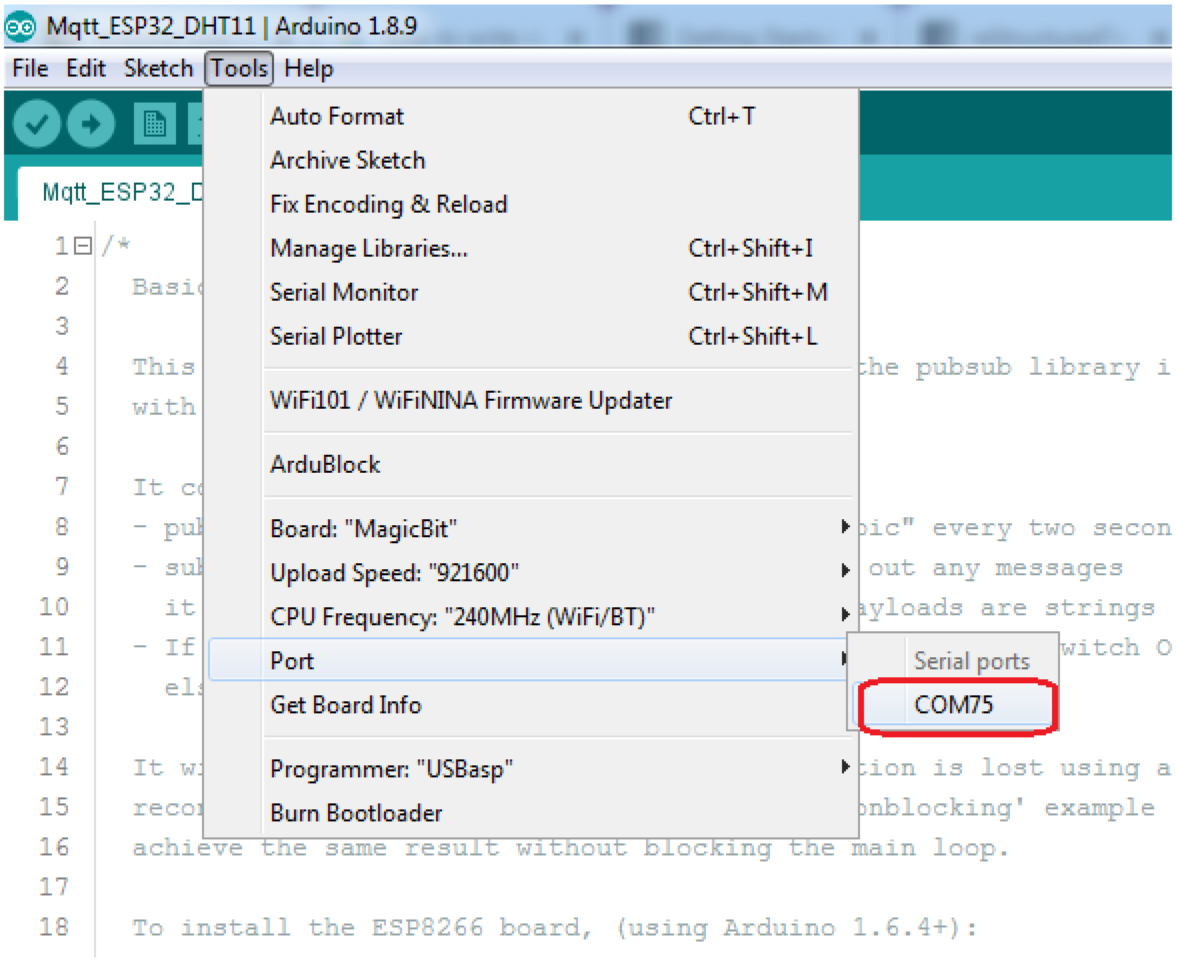
2.1 Installation Instructions

- Relase Link -https://github.com/magicbitlk/arduino-esp32/releases/download/Magicbit/package_magicbit_index.json
- Install the current upstream Arduino IDE at the 1.8 level or later. The current version is at the [Arduino website](http://www.arduino.cc/en/main/software).
- Start Arduino and open Preferences window.
- Enter one of the release links above into Additional Board Manager URLs field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install Magicbit platform (and don't forget to - - select Magicbit from Tools > Board menu after installation).

2.2 Powering Up

Magicbit can be powerup by either connecting USB cable or connecting battery. For programming USB cable must be connected to the computer. For the first time powering up Magicbit self test program will be running on the Magicbit and you can see the features available and functional tests on Magicbit display.

To check whether drivers are correctly installed open the Ardunio IDE and go the Tools menu. There should be a port (Eg:COM1) shown when plugging Magicbit to the computer as shown below. If not please follow Installation drivers section.


Mqtt_ESP32_DHT11 | Arduino 1.8.9
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ArduBlock
Board: "MagicBit"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Port
Get Board Info
Programmer: "USBasp"
Burn Bootloader
Serial ports
COM75

2.3 Installation Drivers (Optional)

Magicbit has CH340 chip as USB-Serial converter which driver already packaged with Ardunio IDE. If port not shown in the Arduino as shown below please install driver

2.4 First Project

- Open Ardunio IDE if not opened already.
- Select Magicbit from **Tools->Boards**

- Select port **Tools->Ports**
- Open Blink Example **File->Examples->Basic->Blink**
- Upload the code to the Magicbit using upload button on Arduino IDE
- If Green Led on backside of the Magicbit is blinking your have just begun the magic with Magicbit

Warning: To use analogWrite, Tone and Servo funtions,Include ESP32Servo **Sketch->Include Library->ESP32Servo** or put **#include <ESP32Servo.h>** on top of arduino sketch.

CHAPTER 3

Examples

3.1 Example 1: Blinking an LED

3.1.1 Introduction

In this example you are learning how to turn on and off a LED or any other actuator which can be controlled by a digital output such as relay, bulb, motor.

3.1.2 Learning Outcomes

From this example, you'll get an understanding about,

- Pin Mode
- Digital Write
- Delay Functions

3.1.3 Components

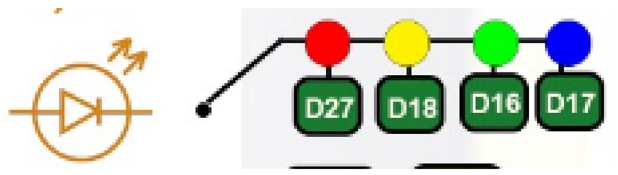
- Magicbit

3.1.4 Theory

A digital output allows you to control a voltage with an electronic device. If the device instructs the output to be high, the output will produce a voltage (generally about 5 or 3.3 volts). If the device instructs the output to be low, it is connected to ground and produces no voltage.Here Magicbit is the device and output voltage is either 3.3V for HIGH and 0V for LOW.

3.1.5 Methodology

Magicbit equipped with four onboard leds in Magicbit development board, Lets select yellow LED (which is wired to D18)



By setting output state to high of LED pin will turn on the led and by setting output state to LOW will turn of LED.

3.1.6 Coding

```
void setup(){
   pinMode(18,OUTPUT);
}
void loop(){
   digitalWrite(18,HIGH);
   delay(1000);
   digitalWrite(18,LOW);
   delay(1000);
}
```

3.1.7 Explanation

pinMode(pin, Mode): Configures the specified pin to behave either as an input or an output. Here we use pin as an output

digitalWrite(pin No, State): Write a HIGH or a LOW value to a digital pin.Pin mode must be setup for the same pin in Setup to work this function properly.

delay(ms): Pauses the program for the amount of time (in milliseconds) specified as parameter.(note 1000 milliseconds equals to one second)

Note: Write code for a knight rider pattern using on board leds of Magicbit

3.2 Example 2: Reading the state of a push button

3.2.1 Introduction

In this example you are learning how read a digital input from something like a button & use it to turn on and off a LED or any other digital device.

3.2.2 Learning Outcomes

From this example, you'll get an understanding about,

- Digital Read
- IF-ELSE conditions

- Variables

3.2.3 Components

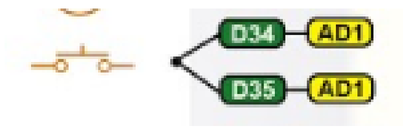
- Magicbit

3.2.4 Theory

A digital input allows you to read digital signals. Microcontroller recognizes the signal as 1(HIGH) when the signal is close to 3.3v (or 5v depending on the microcontroller) and recognizes as 0(LOW) when the signal is close to 0v. This reading can be used in the program to do various things.

3.2.5 Methodology

Magicbit equipped with two onboard push buttons in Magicbit development board, Lets select the push button which is wired to D34. Buttons on the board are in pulled up internally (to learn more about pullups/pulldowns follow this link), which means when button is not pressed the status of the button is 1(HIGH), & when the button is pressed the status of the button is 0(LOW).



Also like in previous example we need to select an LED to indicate the change, lets select RED LED which is wired to pin D27.

First we set the input output configurations of the Button and the LED using pinMode, in this case button is an INPUT, LED is an OUTPUT. Then in the loop section we check the state of the button & store it in an int type variable called buttonState (follow this link to learn more about data types in arduino).

Then we can use the variable as the condition of the if block, and if the button is pressed, the bulb should turn on, and the button is not pressed the light should turn off.

3.2.6 Coding

```
void Setup(){
   pinMode(27,OUTPUT);
   pinMode(34,INPUT);
}
void loop(){

 int buttonState = digitalRead(34);
 if(buttonState == LOW){
   digitalWrite(27, HIGH);
 }else{
   digitalWrite(27, LOW);
 }
}
```

3.2.7 Explanation

digitalRead(pin No): Reads the condition of the given pin and returns a digital value HIGH or LOW.

IF/ELSE: Used to evaluate a digital condition, we can put a digital logic condition in then parenthesis. If the condition is true, it executes the code block in the immediate curly bracket section, if the condition is false it executes the code block in the else curly bracket.

if(condition){ //Do if condition is true

}else{ //Do if condition is false}

Note: Write a code to toggle an LED in the button press. LED turns on when button pressed & released, LED turns off when button is pressed & released again. (Hint: Make use of variables to 'remember' the state of the button press).

3.3 Example 3: Working with Analog Write

3.3.1 Introduction

In this example you are learning how to turn on and off a LED or any other actuator which can be controlled by a digital output such as relay, bulb, motor.

3.3.2 Learning Outcomes

From this example, you'll get an understanding about,

- Pulse Width Modulation
- Analog Write

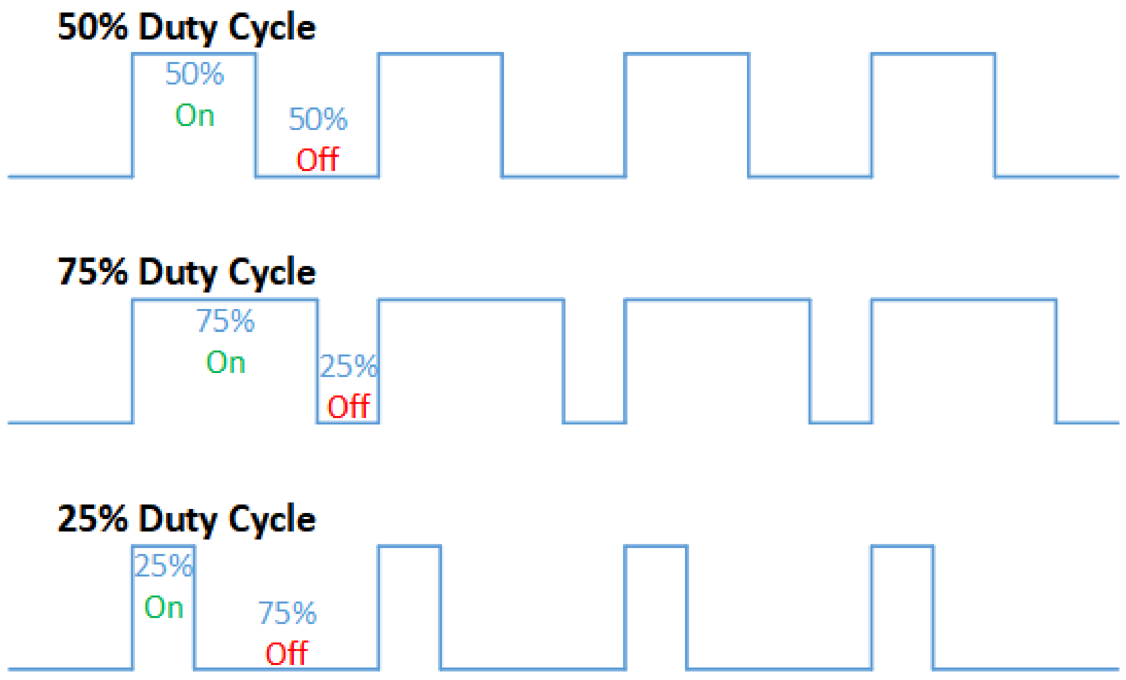
3.3.3 Components

- Magicbit

3.3.4 Theory

To change the brightness of a LED we could change the voltage the LED is supplied with, but in a microcontroller, ability to change the voltage (converting a digital number to an analog voltage) is limited, so a method called PWM (Pulse Width Modulation) is used. What this does is pulsing on and off the pin in a high frequency. The length of the pulses creates the perception of brightness.

Duty cycle is a term used to describe the ratio between on and off times.



In this example higher Duty cycle gives higher brightness & lower duty cycle gives lower brightness.

3.3.5 Methodology

Lets select green LED (which is wired to D16). We will use a for loop to generate the duty cycle (0 - 0% duty, 255-100% duty). And also to generate 255 cycles.

3.3.6 Coding

```
#include <ESP32Servo.h>

void setup(){
      pinMode(16,OUTPUT);
}
void loop(){
      for(int i = 0; i < 256; i++){
      analogWrite(16, i);
      delay(10);
}
```

3.3.7 Explanation

for(int i=0; i<256; i++): There are 3 parameters in a for loop, first parameter we are defining a variable to store the value generated by the for loop. Second parameter specifies the condition that needs to be true to run the for loop(else it breaks out from the loop), third parameter specifies the change happens to the variable in each cycle, in this case 1, added to i.

analogWrite(pin number, pwm value): You can input the pin number you need to do pwm and then the pwm value you need to give to that pin. This assigns the corresponding duty cycle to the pin.

Note This example we have coded to increase the brightness, write a code to do the opposite of that, to fade the brightness of the led, & put both effects together to create a beautiful fade & light up effect.

3.4 Example 4: Using Serial Protocol

3.4.1 Introduction

In this example you are learning to use serial communication function.

3.4.2 Learning Outcomes

From this example, you'll get an understanding about,

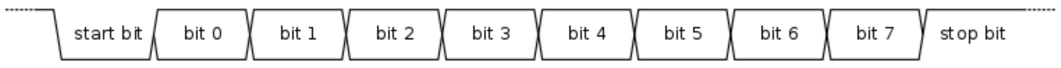
- Serial Protocol usage between Magicbit & the PC

3.4.3 Components

- Magicbit
- Computer with arduino installed

3.4.4 Theory

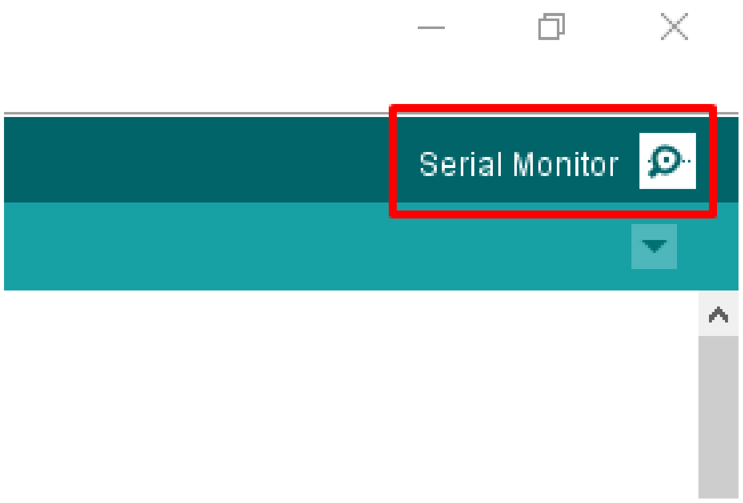
In microcontroller programming, communication between devices is essential. There are hundreds of protocols available, but most common & easy to use is Serial Protocol. Commonly used to communicate information between a microcontroller and a computer.



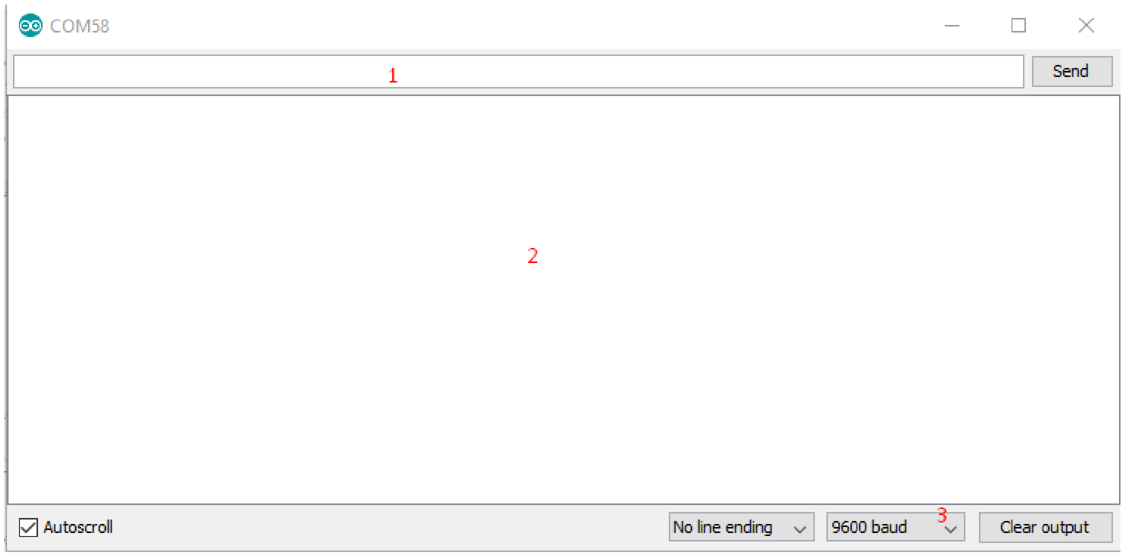
3.4.5 Methodology

We configure a button as the 2nd example (D34 is used). Then we initialize serial communication between the computer and Magicbit. After that in the loop section if condition check if the button is pressed. If pressed, it prints "Button Pressed" on the serial console.

You could use the serial monitor window of arduino IDE to view the serial output



Then the serial console appears (you have to select the serial port number correctly, follow this link to learn how to).



1: You can type in stuff here & hit enter to send data to Magicbit

2: This area shows the data coming from Magicbit

3: From this menu you have to select a common baudrate between the computer and the magic bit.

3.4.6 Coding

```
void setup(){
  pinMode(34,INPUT);
  Serial.begin(9600);
}
void loop(){
  if(digitalRead(34) == LOW){
    Serial.println("Button is Pressed");
  }
}
```

3.4.7 Explanation

Serial.begin(baudrate): Initializes a serial connection, baudrate specifies the speed of data transfer (bits per second). Standard values are 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000 and 25600

Serial.print(stuff to print): Using this function, serial data can be sent, stuff to print can be any type of arduino variable, or even a static string.

Serial.println(stuff to print): Using this function, serial data can be sent, stuff to print can be any type of arduino variable, or even a static string, this is different than Serial.print() is this always prints the content in a new line, rather than printing all in one line.

3.4.8 Activity

Note: do the same example using Serial.print(), observe the difference. Create a button press counter, which displays the button press count on the serial console of arduino IDE.

3.5 Example 5: Reading an Analog Signal

3.5.1 Introduction

In this example you are learning to read an analog sensor & print it on the serial console.

3.5.2 Learning Outcomes

From this example, you'll get an understanding about,

- Analog Read function

3.5.3 Components

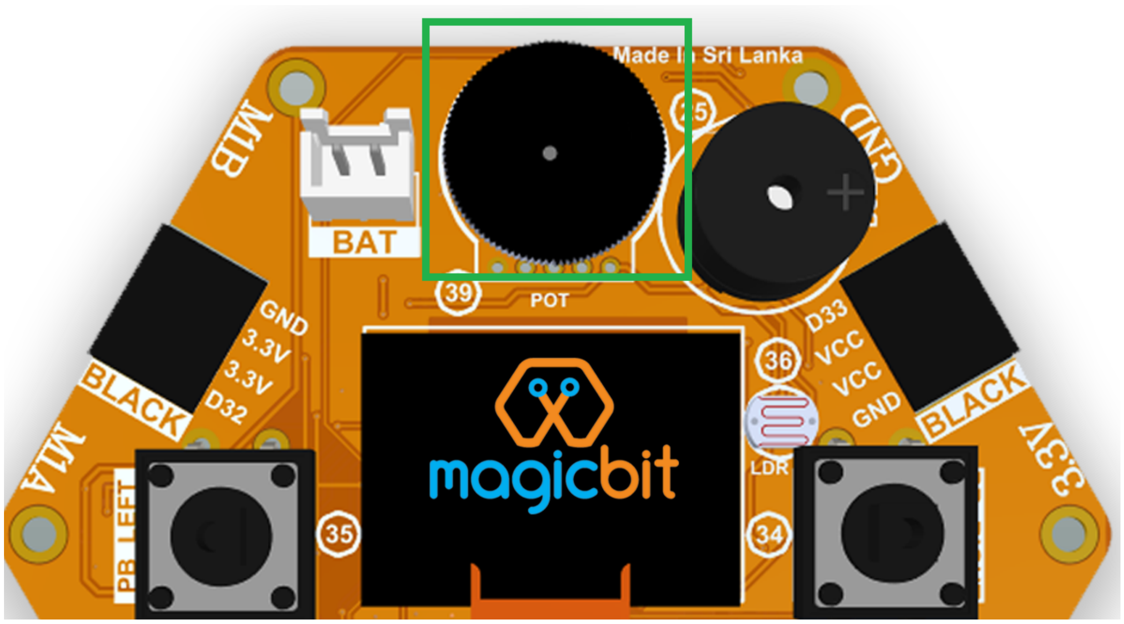
- Magicbit

3.5.4 Theory

In real world most of the signals we encounter are analog signals (temperature, air pressure, velocity), they are continuous. But computers work on digital domain, to interact between the worlds, representing an analog signal in the digital domain is important. (to read more about analog to digital conversation, follow this link)

3.5.5 Methodology

For this example we use the potentiometer on the Magicbit board, which is connected to pin, D39. It generates a voltage between 0 and 3.3V according to the angle of the potentiometer.



We read the analog signal and storing it in an int type variable(0v= 0 analog value, 3.3v = 1024 analog value), sensorValue, later, we use this value to print on the serial window of arduino IDE as well as light up the red LED(D27) if the analog value exceeds than 512.

3.5.6 Coding

```
void setup(){
     pinMode(39,INPUT);
     pinMode(27,OUTPUT)
     Serial.begin(9600);
}
void loop(){
     int sensorValue = analogRead(39);
     Serial.println(sensorValue);

     if(sensorValue > 512){
        digitalWrite(27,HIGH);
     }else{
        digitalWrite(27,LOW);
     }
 }
```

3.5.7 Explanation

analogRead(pin No): this reads and assigns the corresponding analog value to the left.

3.5.8 Activity

Note: Do the same example using the LDR on the board (D36)

3.6 Example 6: Generating Tones

3.6.1 Introduction

In this example you are learning to generate a tone using the onboard buzzer on the Magicbit.

3.6.2 Learning Outcomes

From this example, you'll get an understanding about,

- Tone Function

3.6.3 Components

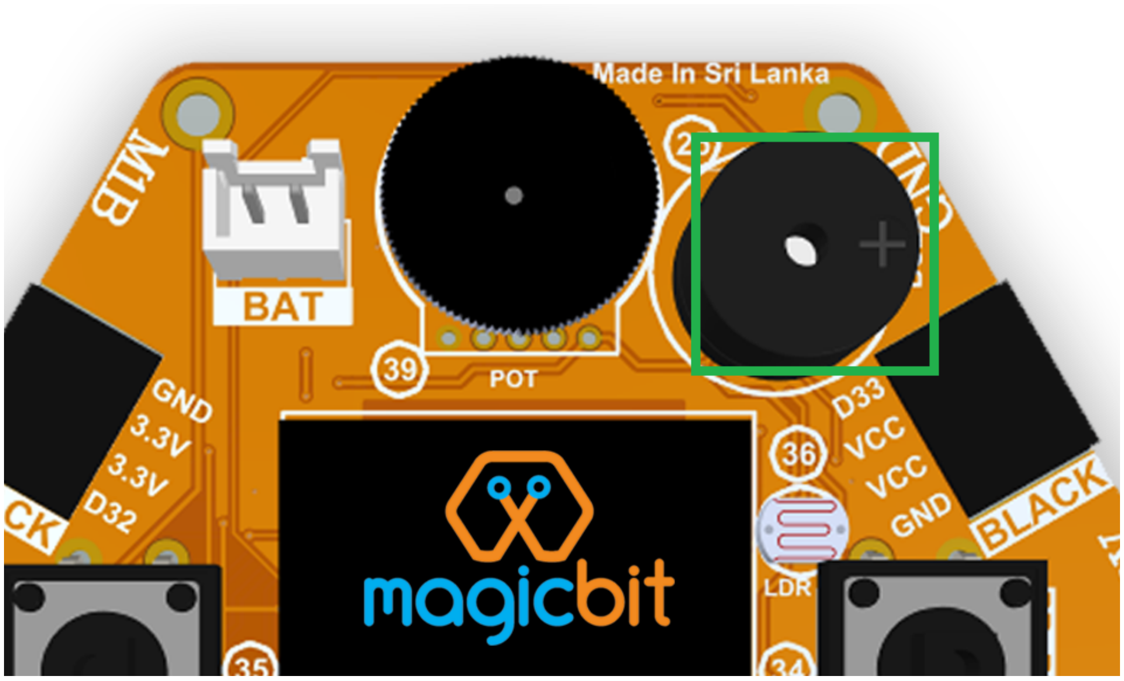
- Magicbit

3.6.4 Theory

Piezo buzzers are commonly used in embedded systems to give audible tones. Combined with ESPservo library Magicbit can generate various tones. (Follow this link to know how to install ESPservo library)

3.6.5 Methodology

For this example we use the piezo buzzer wired to pin 25 of the Magicbit.



ESP32Servo.h library is used to generate pwm signals needed to generate tones. We could specify the frequency & duration of the tone.

3.6.6 Coding

```
#include <ESP32Servo.h>
void Setup(){
   pinMode(25,OUTPUT);
}
void loop(){
   tone( 25 ,4186,500); //C Note
   delay(1000);
   tone( 25,5274,500); //E Note
   delay(1000);
}
```

3.6.7 Explanation

tone(pin No, frequency, duration): generates pwm to corresponding to the given parameters

3.6.8 Activity

Note:: Create a program that plays one frequency when one push button on the board pressed, and another frequency when the other push button when pressed.

3.7 Example 7: Using the onboard OLED Screen

3.7.1 Introduction

Color OLED screen on Magicbit can display text as well as simple logos & images.

3.7.2 Learning Outcomes

From this example, you'll get an understanding about,

- Using Adafruit OLED library

3.7.3 Components

- Magicbit

3.7.4 Theory

Magicbit has a 0.96" OLED Screen which can be communicated with from I2C protocol. The display has the address, **0x3c**.

3.7.5 Methodology

Adafruit OLED library(Adafruit_SSD1306 & Adafruit_GFX) is used to handle the LCD, its important to install those libraries beforehand. First we create the content we need to print onto the screen and then use display.display command to update the screen.

3.7.6 Coding

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET 4

Adafruit_SSD1306 display(128,64);

void setup(){
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  delay(3000);
}


void loop(){
  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(10, 0);
  display.println("Hello");

  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 25);
  display.println("Welcome to");
  display.println();
  display.println("Magic");
  display.println("Bit");
  display.display();
  display.clearDisplay();

  delay(1000);

  }
```

3.7.7 Explanation

display.clearDisplay(): Clears the OLED display.

display.setTextSize(2): Set the font size of the text.

display.setCursor(0, 25): Sets the cursor(determines where the next text will appear).

display.println(stuff to print): print the data given on a new line, similar effect like Serial.println.

display.setTextColor(WHITE): Sets the color of the text.

display.display(): Updates the changes to the screen.

Note:: Make a program to display the ADC value of the potentiometer on the OLED display.

3.8 Magicbit Sensors

3.9 01. Proximity Sensor

3.9.1 1.1 Introduction

In this example, you are learning how to use proximity sensor. This sensor (TCRT5000) uses reflection on a surface theory to working. From a white and polished surface reflects large percentage of light and from a black and rough surface absorbs (not reflect) large percentage of light that incidence on the surface.

Learning outcomes:

- Reflection theory using with Infrared radiations.
- Turning physical parameter to an analog electric signal.

3.9.2 1.2 Components

- Magicbit
- Magicbit proximity sensor

3.9.3 1.3 Theory

A proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact. A proximity sensor often emits an electromagnetic field or a beam of electromagnetic radiation (infrared, for instance), and looks for changes in the field or return signal.

Features:

- Supply voltage 3.3V ~ 5V
- Detect distance 1mm ~ 8mm

3.9.4 1.4 Methodology

1. As the first step, you should connect a Magicbit proximity sensor to Magicbit core board. For this you can use one side connector from four side connectors of the Magicbit core board or if you want to extend the length the connection you can use jumper wires.
2. For this example, the proximity sensor was connected to the upper left connector of the Magicbit core board.
3. Then connect the Magicbit to your pc and upload the code. You can get outputs using serial monitor.

3.9.5 1.5 Coding

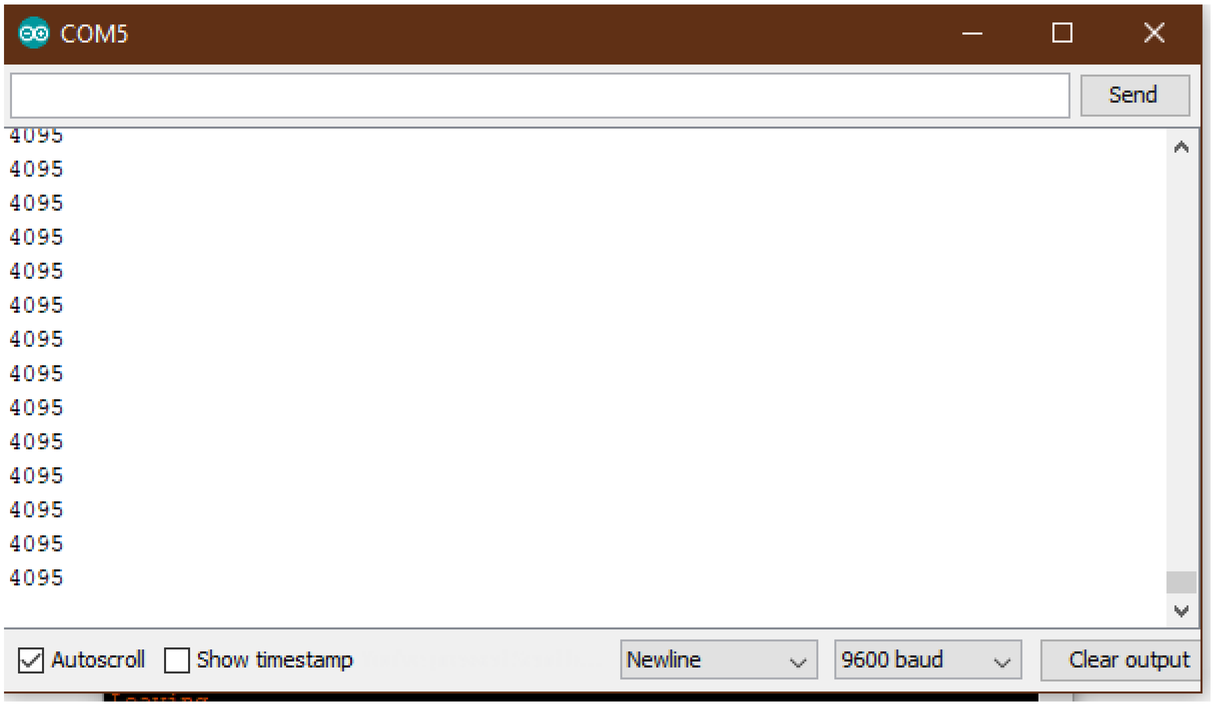
```
const int IRpin = 32;

void setup() {
  Serial.begin(9600);
  pinMode (IRpin, INPUT);
}
void loop() {
```

(continues on next page)

(continued from previous page)

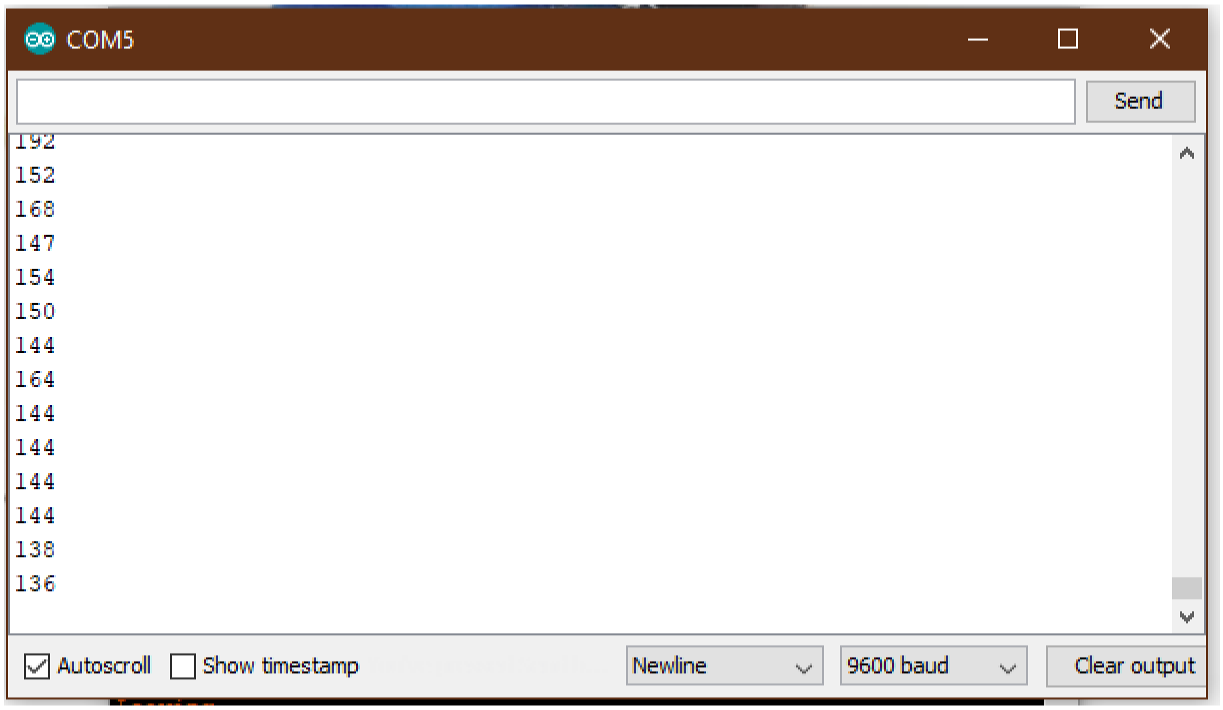
```
  Serial.println(analogRead(IRpin));
  delay(100);
}
```

Outputs: Serial monitor



```
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
4095
```

Figure 1: Serial output when faced a black surface



```
192
152
168
147
154
150
144
164
144
144
144
144
138
136
```

Figure 2: Serial output when faced a white surface

3.9.6 1.6 Explanation

Const int: Defining input pin.

Serial.begin(9600): Setting baud rate.

analogRead: Read the data input of configured data pin.

3.10 02. Tilt Sensor

3.10.1 2.1 Introduction

In this example, you are learning how to use Tilt sensor. Tilt sensor produces digital outputs such as high and low. Therefore, tilt sensor works as a switch that gives on and off states.

Learning outcomes

- Digital reads and print them on serial monitor
- Working principle of the tilt sensor

3.10.2 2.2 Components

- Magicbit
- Magicbit Tilt Sensor

3.10.3 2.3 Theory

When the device gets power and is in its upright position, then the rolling ball settle at the bottom of the sensor to form an electrical connection between the two end terminals of the sensor.

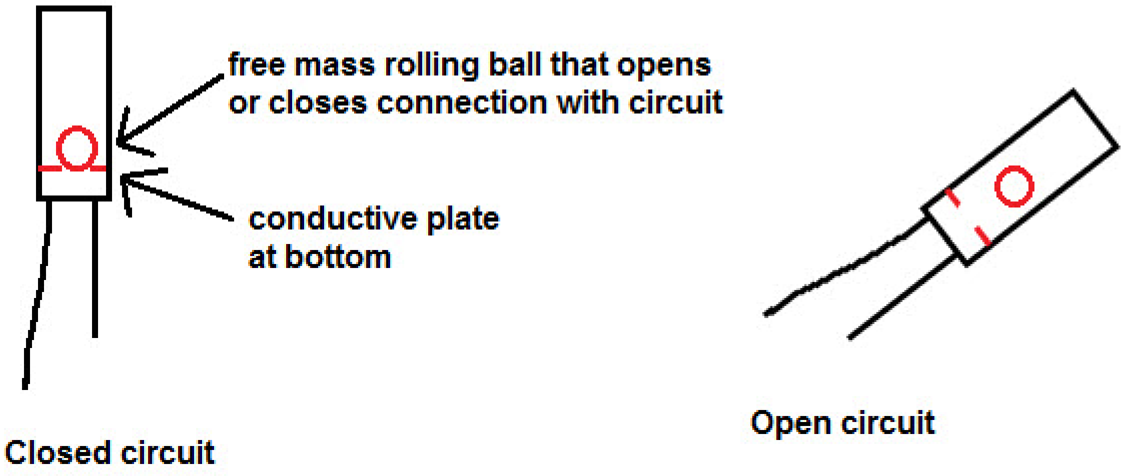

free mass rolling ball that opens or closes connection with circuit
conductive plate at bottom
Closed circuit
Open circuit

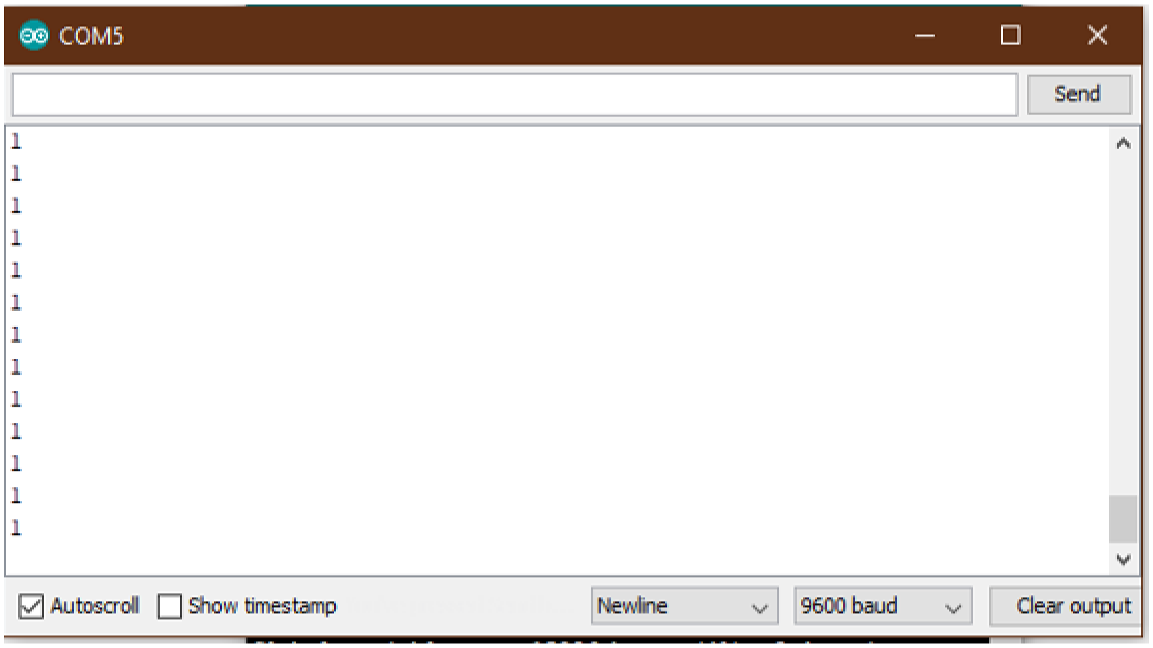
Figure 3: Working priciple of tilt sensor [1]

3.10.4 2.4 Methodology

As the first step, you should connect a Magicbit tilt sensor to Magicbit core board. For this, you can use one side connector from four side connectors of the Magicbit core board or if you want to extend the length the connection, you can use jumper wires. For this example, the tilt sensor was connected to the upper left connector of the Magicbit core board. Then connect the Magicbit to your pc and upload the code. You can get outputs using serial monitor.

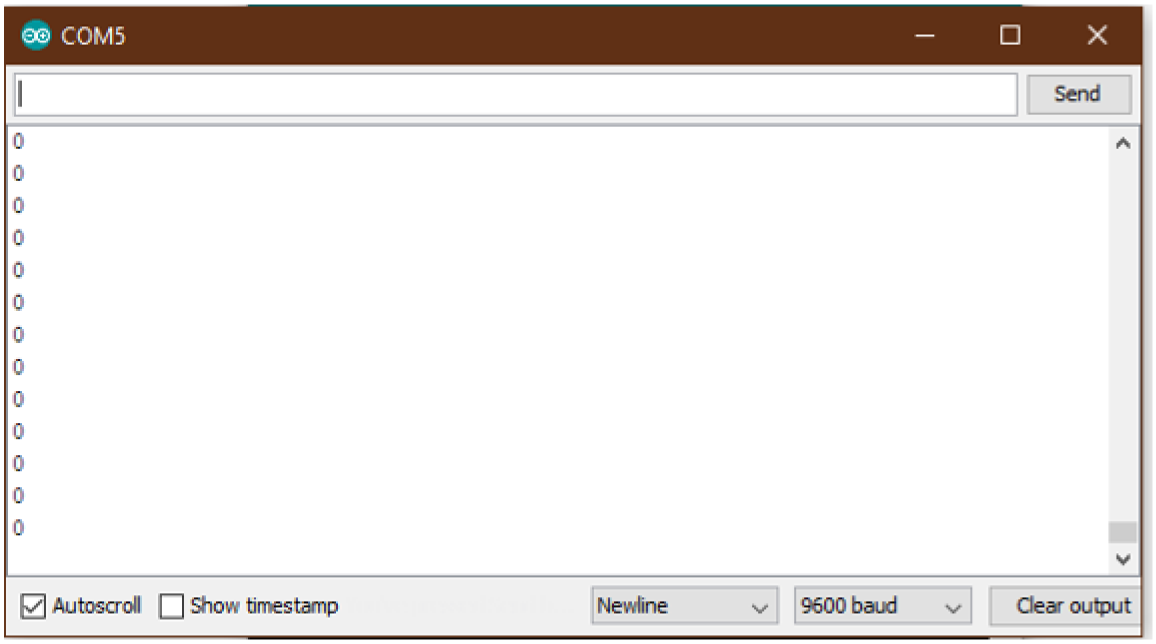
3.10.5 2.5 Coding

```
const int TILTpin = 32;

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

void loop() {
  Serial.println(digitalRead(TILTpin));
  delay(100);
}
```

Outputs: Serial monitor 

```
1
1
1
1
1
1
1
1
1
1
1
1
1
```

Figure 4: High state of the tilt sensor



```
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Figure 5: Low state of the tilt sensor

3.10.6 2.6 Explanation

Const int TILTpin: Defining input pin **digitalRead:** Read the data input of configured data pin.

3.11 03. Flame Sensor

3.11.1 3.1 Introduction

A flame sensor module that consists of a flame sensor (IR receiver), resistor, capacitor, potentiometer, and comparator LM393 in an integrated circuit. It can detect infrared light with a wavelength ranging from 700nm to 1000nm.

Learning outcomes:

- Using flame sensor for identify infrareds/heat bodies

3.11.2 3.2 Components

- Magicbit
- Magicbit Flame Sensor

3.11.3 3.3 Theory

IR receiver mainly use with a IR Transmitter, not only for identify heat bodies. IR light is emitted by the sun, light bulbs, and anything else that produces heat. That means there is a lot of IR light noise all around us. To prevent this noise from interfering with the IR signal, a signal modulation technique is used. In IR signal modulation, an encoder on the IR remote converts a binary signal into a modulated electrical signal. This electrical signal is sent to the transmitting LED. The transmitting LED converts the modulated electrical signal into a modulated IR light signal.

The IR receiver then demodulates the IR light signal and converts it back to binary before passing on the information to a microcontroller.In here we use this sensor for identify flames.

3.11.4 3.4 Methodology

As the first step, you should connect a Magicbit flame sensor to Magicbit core board. For this, you can use one side connector from four side connectors of the Magicbit core board or if you want to extend the length the connection, you can use jumper wires. For this example, the flame sensor was connected to the upper left connector of the Magicbit core board. Then connect the Magicbit to your pc and upload the code. You can get outputs using serial monitor.

3.11.5 3.5 Coding

```
const int FLAMEpin = 32;

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

void loop() {
  Serial.println(analogRead(FLAMEpin));
  delay(100);
}
```

3.11.6 3.6 Explanation

Here we give an **analogRead**. That because we have to measure a range to take a decision that is there a flame or not.

3.12 04. DOOR Sensor

3.12.1 4.1 Introduction

A magnetic contact switch is a reed switch encased in a plastic shell so that you can easily apply them in a door, a window or a drawer to detect if the door is open or closed.

Learning outcomes:

- Using magnetic door sensor.

3.12.2 4.2 Components

- Magicbit
- Magicbit Magnetic door sensor

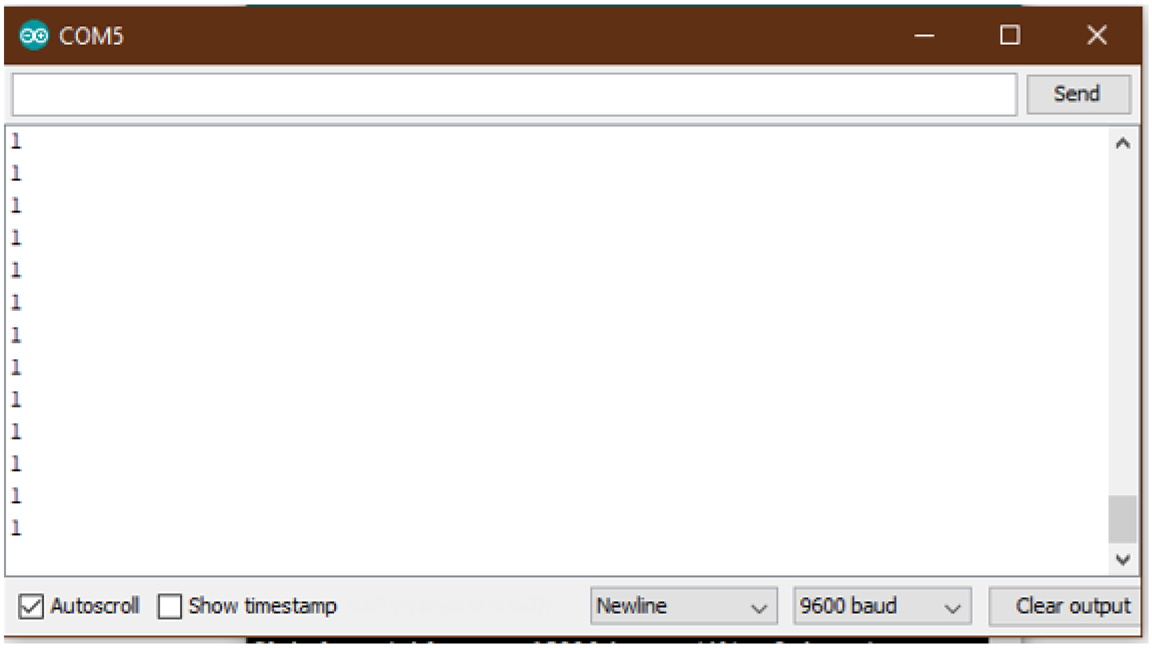
3.12.3 4.3 Theory

Almost all door and window sensors use a "reed switch" to determine when a protected area has been breached. A reed switch consists of a set of electrical connectors placed slightly apart. When a magnetic field is placed parallel to the electrical connectors, it pulls them together, closing the circuit. Door sensors have one reed switch and one magnet, creating a closed circuit. If someone opens an armed door or window, the magnet is pulled away from the switch, which breaks the circuit and triggers an event. [2]

Figure 6: Working principal of magnetic door sensor

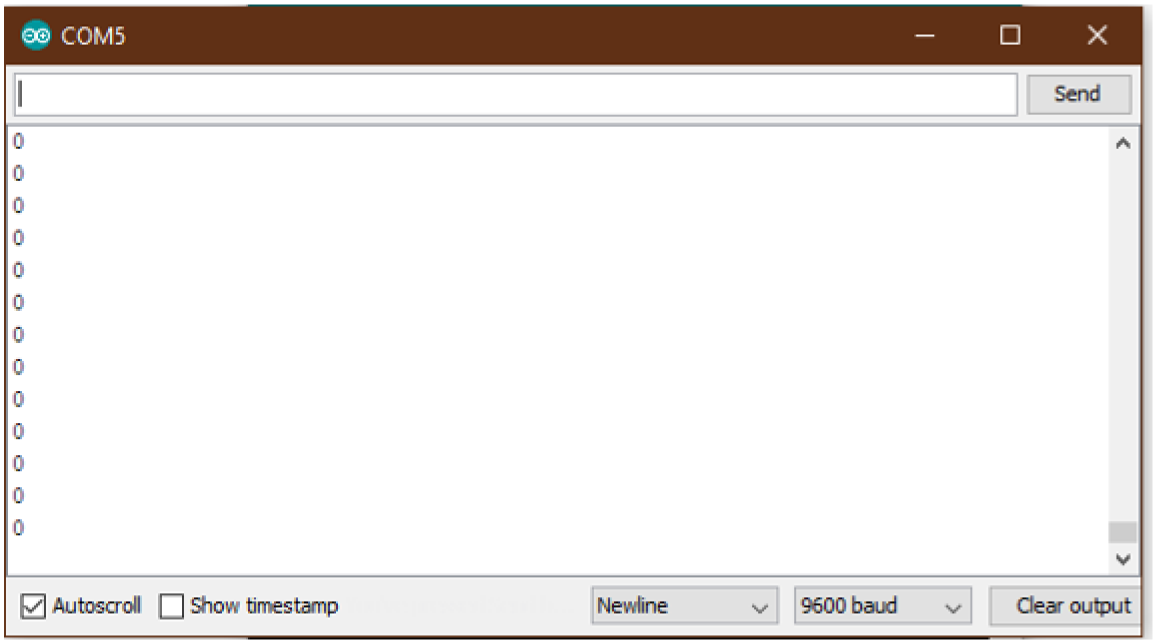
3.12.4 4.4 Methodology

First, take the Magicbit door sensor and connect it with the Magicbit core. In this example, we use 32th pin of Magicbit for implement this. After connect the door sensor upload following code for your Magicbit. Then open your serial monitor in your Arduino IDE and see outputs while changing the door sensor module.



```
1
1
1
1
1
1
1
1
1
1
1
1
1
```

Figure 7: Door open state



```
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Figure 8: Door closed state

3.12.5 4.5 Coding

```
const int DOORpin = 32;

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

void loop() {
  Serial.println(digitalRead(DOORpin));
  delay(100);
}
```

3.12.6 4.6 Explanation

DOORpin: Defined input pin for door sensor

3.13 05. Magicbit Servo

3.13.1 5.1 Introduction

A servomotor is an electrical device, which can push or rotate an object with great precision. If you want to rotate and object at some specific angles or distance, then you use servomotor. It is just made up of simple motor, which run through servomechanism.

Leaning outcome:

- Using servo motor with Magicbit

3.13.2 5.2 Components

- Magicbit
- Servomotor

3.13.3 5.3 Theory

Servo motor works on the PWM (Pulse Width Modulation) principle, which means its angle of rotation, is controlled by the duration of pulse applied to its control PIN. Servomotor is made up of DC motor, which is controlled by a variable resistor (potentiometer), and some gears. Servomotors control position and speed very precisely. Now a potentiometer can sense the mechanical position of the shaft. Hence, it couples with the motor shaft through gears. The current position of the shaft is converted into electrical signal by potentiometer, and is compared with the command input signal. In modern servomotors, electronic encoders or sensors sense the position of the shaft. A pulse of 1ms will move the shaft anticlockwise at -90 degree, a pulse of 1.5ms will move the shaft at the neutral position that is 0 degree and a pulse of 2ms will move shaft clockwise at +90 degree. [3]

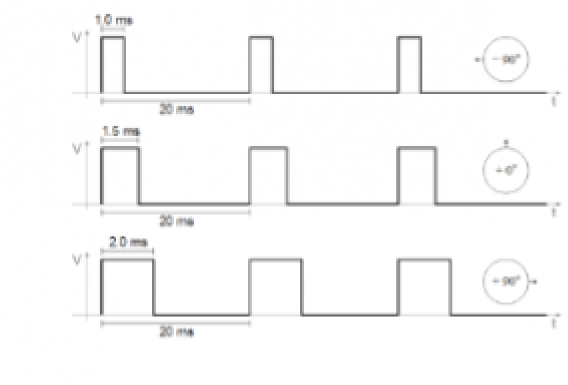
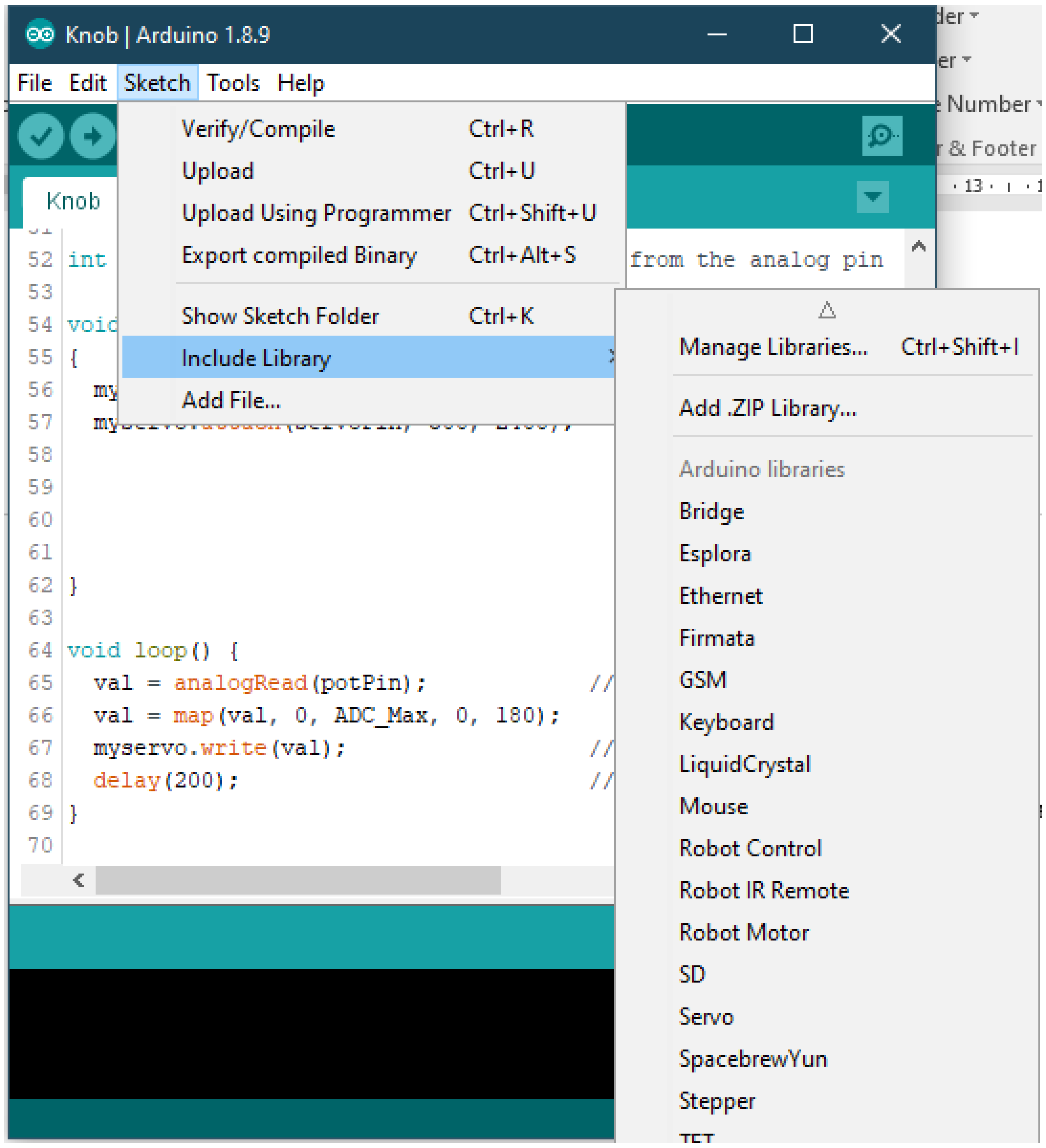


Figure 9:PWM Signals for various angles

3.13.4 5.4 Methodology

For implement this project ESP32Servo library should be installed. Click here to download ESP32Servo library. Then install the library for Arduino IDE. Follow these steps to install ESP32Servo library.



```
int                                  from the analog pin

void
{
  my
  my

}

void loop() {
  val = analogRead(potPin);            //
  val = map(val, 0, ADC_Max, 0, 180);
  myservo.write(val);                  //
  delay(200);                          //
}
```

Figure 10: Iclude library -> Add.ZIP library

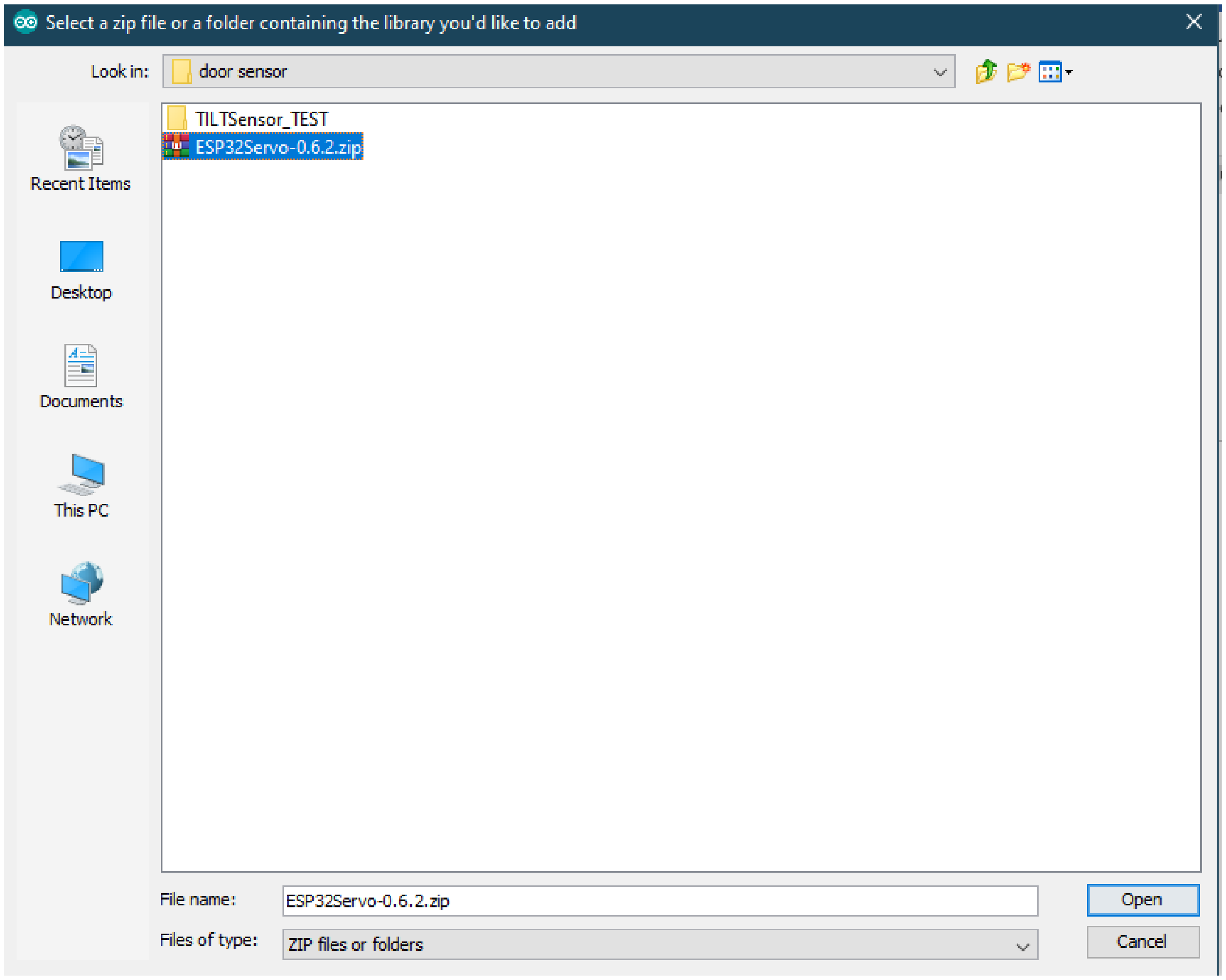


Figure 11: Select ZIP file

Then connect the magic servo motor t magic bit. After completed those steps, upload following code for your Magicbit.

3.13.5 5.5 Coding

```
#include <ESP32Servo.h>
Servo MagicServo;

void setup() {
MagicServo.attach(32);
 }

void loop( ) {
 for(int i=0; i<=180; i++){
  MagicServo.write(i);
  delay(10);
  }
}
```

3.13.6 5.6 Explanation

Servo MagicServo: We should create an object in program for define the servomotor **MagicServo.attach:** 'attach' means define which pin of the Magicbit connects to the servomotor. **For loop:** In here, we use for loop to incrementing loop action. Because of this the servomotor increments its angle 0 to 180 and after complete this action reset to the start position. This action is continued repeatedly inside the 'for loop'.

3.14 06. Motion Sensor

3.14.1 6.1 Introduction

A motion sensor (or motion detector) is an electronic device that is designed to detect and measure movement. Motion sensors are used primarily in home and business security systems. PIR Sensor is short for passive infrared sensor, which applies for projects that need to detect human or particle movement in a certain range, and it can be referred as PIR (motion) sensor, or IR sensor. [4]

Learning outcomes:

- Using motion sensor
- Theoretical background of using Infrared waves in motion sensor

3.14.2 6.2 Components

- Magicbit
- Magicbit Motion sensor

3.14.3 6.3 Theory

When a human or animal body will get in the range of the sensor, it will detect a movement because the human or animal body emits heat energy in a form of infrared radiation. That is where the name of the sensor comes from, a Passive Infra-Red sensor. In addition, the term "passive" means that sensor is not using any energy for detecting purposes; it just works by detecting the energy given off by the other objects.

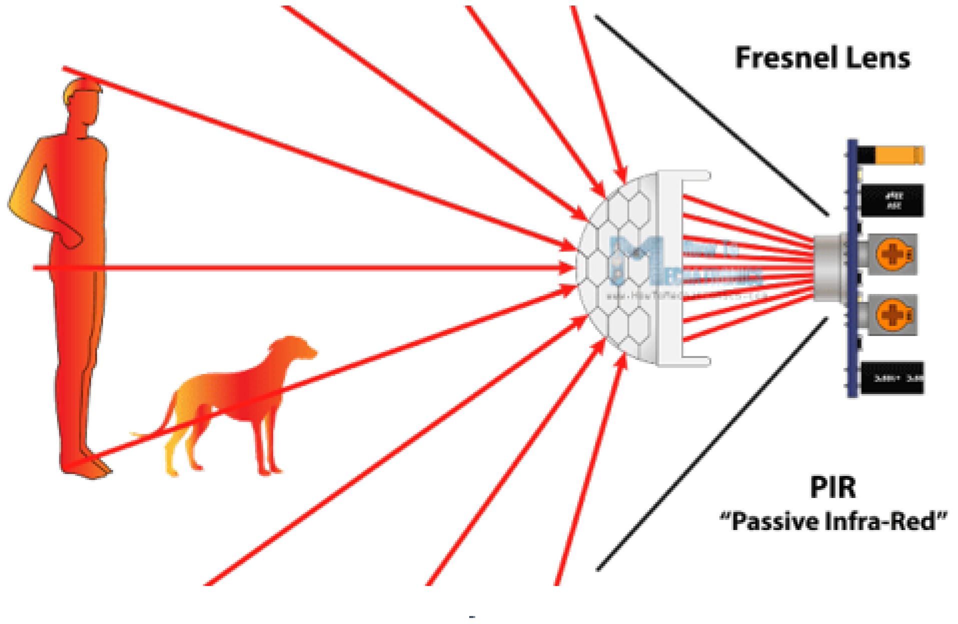


Figure 12: PIR Sensor –Howtomechatronics.com

3.14.4 6.4 Methodology

First, connect the motion sensor to your Magicbit and upload the following code to your Magicbit. In this demonstration like other demonstrations, we use D32 as the data pin.

3.14.5 6.5 Coding

```
int MOTIONsensor =32;
void setup() {
  pinMode(MOTIONsensor, INPUT);
  Serial.begin(9600);
}
void loop() {
  Serial.println(digitalRead(MOTIONsensor));
}
```

3.14.6 6.6 Explanation

When some human being detected by the motion sensor, which is in the range of the sensor, the output of the serial monitor, will be displayed '1'. If not there will be displayed '0'.

3.15 07. RGB Module

3.15.1 7.1 Introduction

An RGB LED has 4 pins, one for each color (Red, Green, Blue) and a common cathode. It has three different color-emitting diodes that can be combined to create all sorts of color. R- Red G- Green B- Blue

Learning outcomes:

- Using a RGB led and changing its color as the required

3.15.2 7.2 Components

- Magicbit
- RGB module

3.15.3 7.3 Theory

The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue. The main purpose of the RGB color model is for the sensing, representation, and display of images in electronic systems, such as televisions and computers, though it has also been used in conventional photography. Before the electronic age, the RGB color model already had a solid theory behind it, based in human perception of colors [5].

3.15.4 7.4 Methodology

For this demonstration, you have to install Adafruit NeoPixel library. For more details Click here. As usually connect the RGB module to your Magicbit, for this, we take data pin as pin 32. After connect the RGB module to the Magicbit, connect it to your pc and upload following code.

3.15.5 7.5 Coding

```
#include <Adafruit_NeoPixel.h>
#define LED_PIN  32
#define LED_COUNT 1
Adafruit_NeoPixel LED(1,32, NEO_RGB + NEO_KHZ800);

void setup() {
  LED.begin();
  LED.show();
}

void loop() {
  LED.setPixelColor(0, 255, 0, 255); // you can change these arguments and make your␣
↪own designs using                  those commands. Follow the link in our␣
↪documentary for more details.
  LED.show();
}
```

3.15.6 7.6 Explanation

Adafruit NeoPixel library is for LED strips. However, it can be used for single RGB LED as your requirement (like this example). 'LED.begin & LED.show' are functions of Adafruit NeoPixel library for display the color on RGB led. 'LED.setPixelColor' is use to color led brightness values. (Eg:- 255 – maximum brightness & 0 – lowest brightness)

3.16 08. Magnetic Sensor

3.16.1 8.1 Introduction

> Magnetic sensors are able to detect magnetic fields and process this information. The outcome on the position, angle and strength (Hall Effect) or the direction (Magneto Resistive) of an applied magnetic field can be converted into specific output signals.

Learning outcomes:

- Using Hall Effect sensor and detect magnetic fields.
- Applications of Hall Effect Sensor

3.16.2 8.2 Components

- Magicbit
- Soil Moisture Sensor

3.16.3 8.3 Theory

There are actually, two different types of Hall sensors one is Digital Hall sensor and the other is Analog Hall sensor. The digital Hall sensor can only detect if a magnet is present or not (0 or 1) but an analog hall sensor's output varies based on the magnetic field around the magnet that is it can detect how strong or how far the magnet is. In this project will aim only at the digital Hall sensors for they are the most commonly used ones. [6]

In a Hall Effect sensor, a thin strip of metal has a current applied along it. In the presence of a magnetic field, the electrons in the metal strip are deflected toward one edge, producing a voltage gradient across the short side of the strip (perpendicular to the feed current).

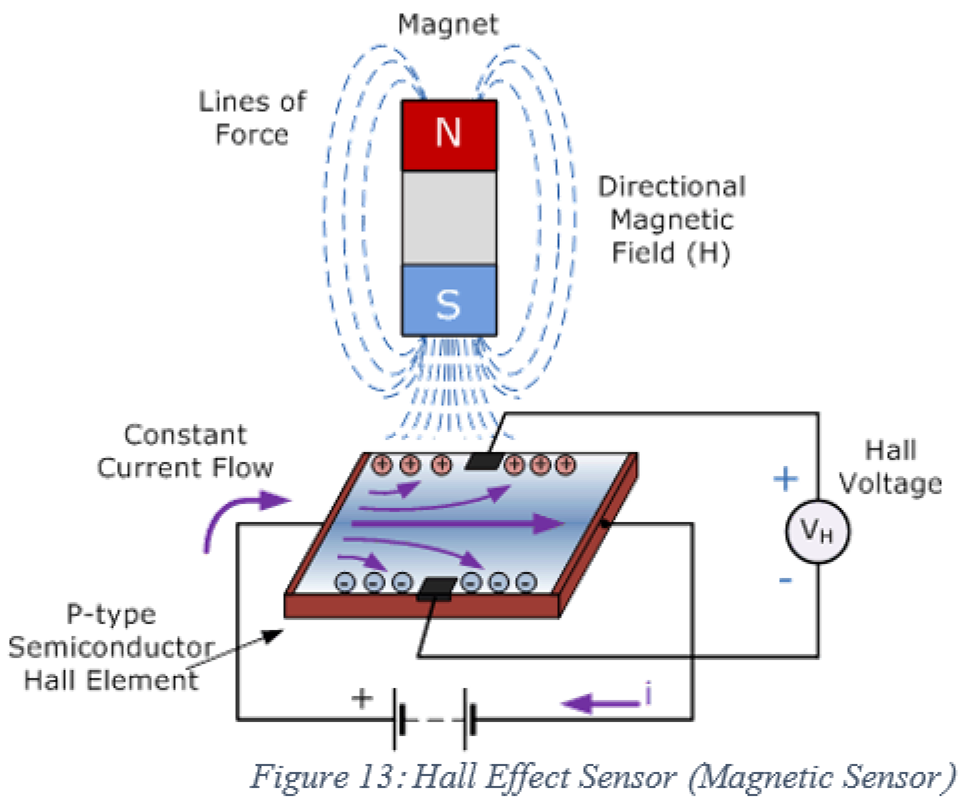

Magnet
Lines of Force
N
S
Directional Magnetic Field (H)
Constant Current Flow
Hall Voltage
+
V_H
-
P-type Semiconductor Hall Element
+
i
Figure 13: Hall Effect Sensor (Magnetic Sensor)

Figure 13: Hall Effect Sensor (Magnetic Sensor)

3.16.4 8.4 Methodology

Connect the magnetic sensor to the Magicbit. For this demonstration, we connect the magnetic sensor to D32 pin of the Magicbit. After connect the magnetic sensor to the Magicbit connect it to your pc and upload the code below.

3.16.5 8.5 Coding

```
#define MAGNETICsensor 32
void setup() {
  Serial.begin(9600);
  pinMode(32, INPUT);
}
void loop() {
Serial.println(digitalRead(MAGNETICsensor));
delay(100);
}
```

3.16.6 8.6 Explanation

This Magnetic sensor gives digital outputs. Therefor you can open the serial monitor and see the outputs. ‘1’ for occurred a magnetic field near to the sensor ‘0’ for there is no any considerable magnetic field near by the sensor

3.17 9.0 Soil Moisture Sensor

3.17.1 9.1 Introduction

Soil moisture sensors typically refer to sensors that estimate volumetric water content. Another class of sensors measure another property of moisture in soils called water potential; these sensors are usually referred to as soil water potential sensors and include tensiometers and gypsum blocks.

Learning outcomes:

- Using Soil moisture sensor and implement its applications
- Working principal of soil moisture sensor

3.17.2 9.2 Components

- Magicbit
- Soil Moisture Sensor

3.17.3 9.3 Theory

Soil Moisture Sensor. Soil moisture is basically the content of water present in the soil. This can be measured using a soil moisture sensor which consists of two conducting probes that act as a probe. It can measure the moisture content in the soil based on the change in resistance between the two conducting plates.

3.17.4 9.4 Methodology

Connect the soil moisture sensor to the Magicbit. As usually in here also we connect sensor module to the upper left (D32) connector on the Magicbit. After connect the sensor module put it in to a wet soil mixture for get results. Then connect the Magicbit to your pc and upload the code below.

3.17.5 9.5 Coding

```
int SENSOR = 32;
int output_value ;

void setup() {
   pinMode(32,INPUT);
   Serial.begin(9600);
   Serial.println("Reading From the Sensor ...");
   delay(2000);

}

void loop() {

   output_value= analogRead(SENSOR);
   output_value = map(output_value,550,0,0,100);
   Serial.print("Mositure : ");
   Serial.print(output_value);
   Serial.println("%");
```

(continues on next page)

(continued from previous page)

```
    delay(1000);
}
```

3.17.6 9.6 Explanation

'output_value = map(output_value, 550,0, 0,100)' - output_value is an user defined variable. For display a moisture percentage we should map the analog output value of the sensor given according to the sample (the wet soil mixture). From serial monitor we can get our outputs.

3.18 10. Temperature and Humidity Sensor

3.18.1 10.1 Introduction

A humidity sensor (or hygrometer) senses, measures and reports both moisture and air temperature. The ratio of moisture in the air to the highest amount of moisture at a particular air temperature is called relative humidity. Relative humidity becomes an important factor when looking for comfort.

Learning outcomes:

- Using DHT11 sensor and getting outputs of temperature and humidity
- Apply Temperature & Humidity sensor in projects

3.18.2 10.2 Components

- Magicbit
- Temperature and Humidity Sensor

3.18.3 10.3 Theory

The DHT11 detects water vapor by measuring the electrical resistance between two electrodes. The humidity-sensing component is a moisture holding substrate with electrodes applied to the surface. When water vapor is absorbed by the substrate, ions are released by the substrate, which increases the conductivity between the electrodes. The change in resistance between the two electrodes is proportional to the relative humidity. Higher relative humidity decreases the resistance between the electrodes, while lower relative humidity increases the resistance between the electrodes. The DHT11 measures temperature with a surface mounted NTC temperature sensor (thermistor) built into the unit. [7]

3.18.4 10.4 Methodology

First, you have to download and install library for DHT11. For more details and download the library Click here Connect the Temperature & Humidity sensor to the Magicbit via left upper connector (D32). Then connect the Magicbit to your pc and upload the following code.

3.18.5 10.5 Coding

```
#include "DHT.h"
#define DHTPIN 32
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
 Serial.begin(9600);
 Serial.println(F("DHTxx test!"));
 dht.begin();
}
void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
  Serial.println(F("Failed to read from DHT sensor!"));
  return;
}
  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.print(f);
  Serial.print(F("°F  Heat index: "));
  Serial.print(hic);
  Serial.print(F("°C "));
  Serial.print(hif);
  Serial.println(F("°F"));
}
```

3.19 11. Ultrasonic Sensor

3.19.1 11.1 Introduction

Ultrasonic sensor is used to measure the distance to objects in front of the sensor by using ultrasonic waves.The human body doesn't sensitive for this signal. Therefor we can't hear any sound when it is working.

Learning outcomes:

- Using HC-SR04 ultarsonic sensor and getting outputs of distances
- Apply Ultrasonic sensor in projects

3.19.2 11.2 Components

- Magicbit
- Ultrasonic sensor

3.19.3 11.3 Theory

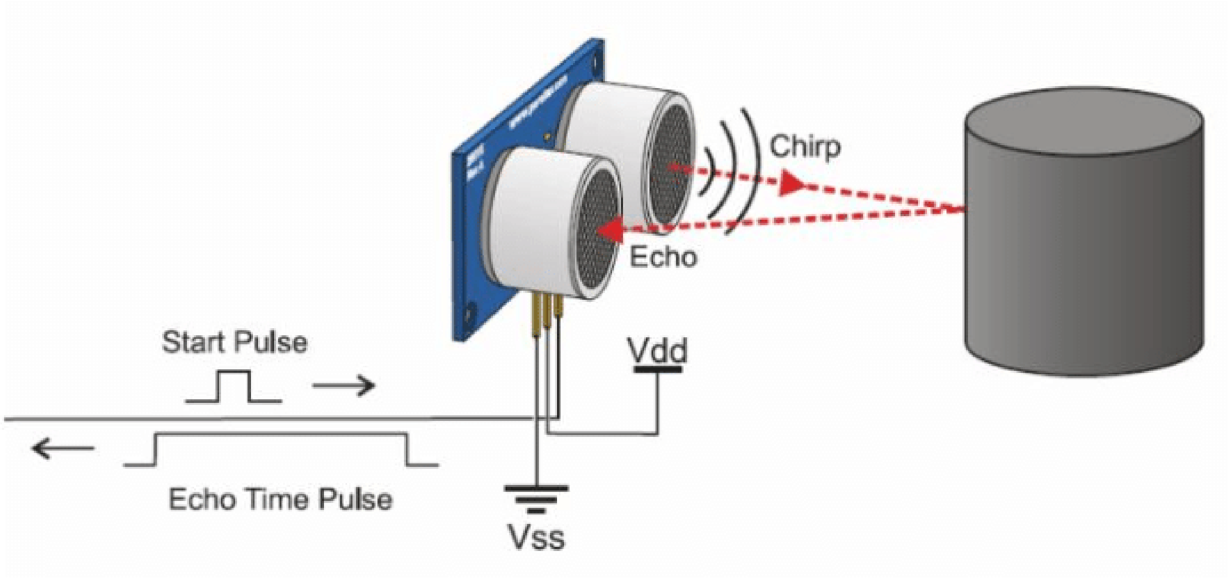
Any kind of ultrasonic sensor works on same way. For measuring distance to object it uses ultrasonic waveform. The sensor have two parts. One is wave transmitter part and other one is receiving part. The transmitter part emits an ultrasonic wave and receives the reflected waveform back from the emitter. The time duraion between transmit and receive is used to measure the distance. If the time duration is low then object is near. If the time duration is high the object is too far. Distance and the time duration is directly propotional parameters. Distance between object and the sensor can be determined by following equation.

> Distance=(speed of ultrasound wave in air)*(time duration)/2

Speed of ultrasound wave in air is 340 meters per second. To measure the distance,we have to trigger the transmitter in certain time duration. If this time duration is very small, then it cant be measured. If this is too high it can cause to noices. Therefor it emites ultrasonic waves in small certain time duraions. Then checks the reciever part(echo pin) until it detects.

3.19.4 11.4 Methodology

Connect the ultrasonic sensor module to magicbit using connector wire.in here we connect sensor module to the upper left (D32) connector on the Magicbit. Then connect the Magicbit to your pc and upload the following code. Now open serial monitor. For good results keep the sensor vertcally and keep the object surface parrallel to the senosr sensor face.



3.19.5 11.5 Coding

```
#include <NewPing.h>
#define TRIGGER_PIN  32
#define ECHO_PIN     32
#define MAX_DISTANCE 200

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup() {
Serial.begin(115200);
```

(continues on next page)

(continued from previous page)

```
}

void loop() {
delay(50);
Serial.print("Ping: ");
Serial.print(sonar.ping_cm());
Serial.println("cm");
}
```

3.20 12. IR LED

3.20.1 12.1 Introduction

IR or Infrared is commonly use communication technology in many cases. Becasue it is inexpensive and easy. Becuase of this technology is depend on light, this is widely used for short distance communication. The Ir LED module emits Infrared light in slightly higher freaquencies. So it use to transmit data and using IR receiver, we can take that data from another place.

Learning outcomes:

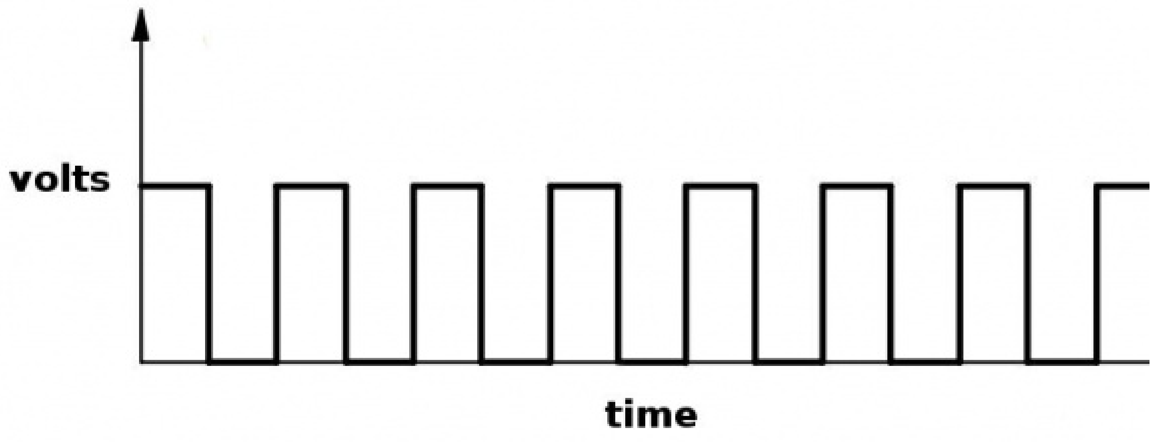
- Using IR LED and getting transmit data
- Apply IR LED in projects

3.20.2 12.2 Components

- Magicbit
- IR LED module

3.20.3 12.3 Theory

IR light have slightly high wave length than visible light. So it can't see for us. Also it have high frequency range. Therfor using IR light we can trnsmit data from high frequency waveform. This is called modulated signal transmites. The sun and every light source emmite IR light. IR light is always arround us with many higher frequencies. Therefor in the communication we used some rare natural IR freaquencies. In many cases 38KHz is used. In this frequency the IR LED on and OFF 38000 times in a second.the encoded modulated data(binary data) transmitted by changing on and off pattern. Then this wave is receiving and demodulate by using some IR sensor. After using some microcontroller we can decode and know what is the the transmitter side is sent.



3.20.4 12.4 Methodology

Connect the IR LED module to magicbit. As usually we connect this module to upper right connector(D33) of the magicbit. Download and install IRremote library from here.Then connect the Magicbit to your pc and upload the following code.This code is used for transmit some data to some other device like TV,A/C or etc, When Right push button is pressed. According to your purpose change the binary code to change data what you want to transmit.

3.20.5 12.5 Coding

```
#include<IRremote.h>

#define PB_Right 34;
#define IR_SEND_PIN 33;
boolean buttonState;
long irKeyCode= 16582903;
IRsend irsend;

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

void loop(){
buttonState=digitalRead(34);
if(buttonState==LOW){
        irsend.sendSony(irKeyCode, 32);
        Serial.println("Sending");
        delay(40);
        }
}
```

3.21 13. Inbuilt motor controller

3.21.1 13.1 Introduction

There are many projects where we have to use motors for many purposes. All processors work under 5V or 3.3V. So their outputs are not enough to supply larger current and voltages to control motors. In this case we use an additional module to control motors. That is a motor driver. As the name suggests, every motor driver is doing the same thing. That is , controlling motors using external power sources based on microcontroller signals. These controlling signals are not constant voltage values. They are PWM(Pulse width modulation) signals. These signals are digital signals. Lot of motor drivers use the H-Bridge mechanism to control the motors. If you use an Arduino board you have to use an external motor driver to control motors. But in the Magicbit you don't want to buy any external motor controllers. Because it already has an inbuilt H-bridge motor driver. So you can directly connect motors to the Magicbit and you can play with them.

Learning outcomes:

- Using inbuilt motor driver to control motors
- Apply motor controlling process for projects

3.21.2 13.2 Components

- Magicbit
- brushed DC motors(3V or 6V)

3.21.3 13.3 Theory

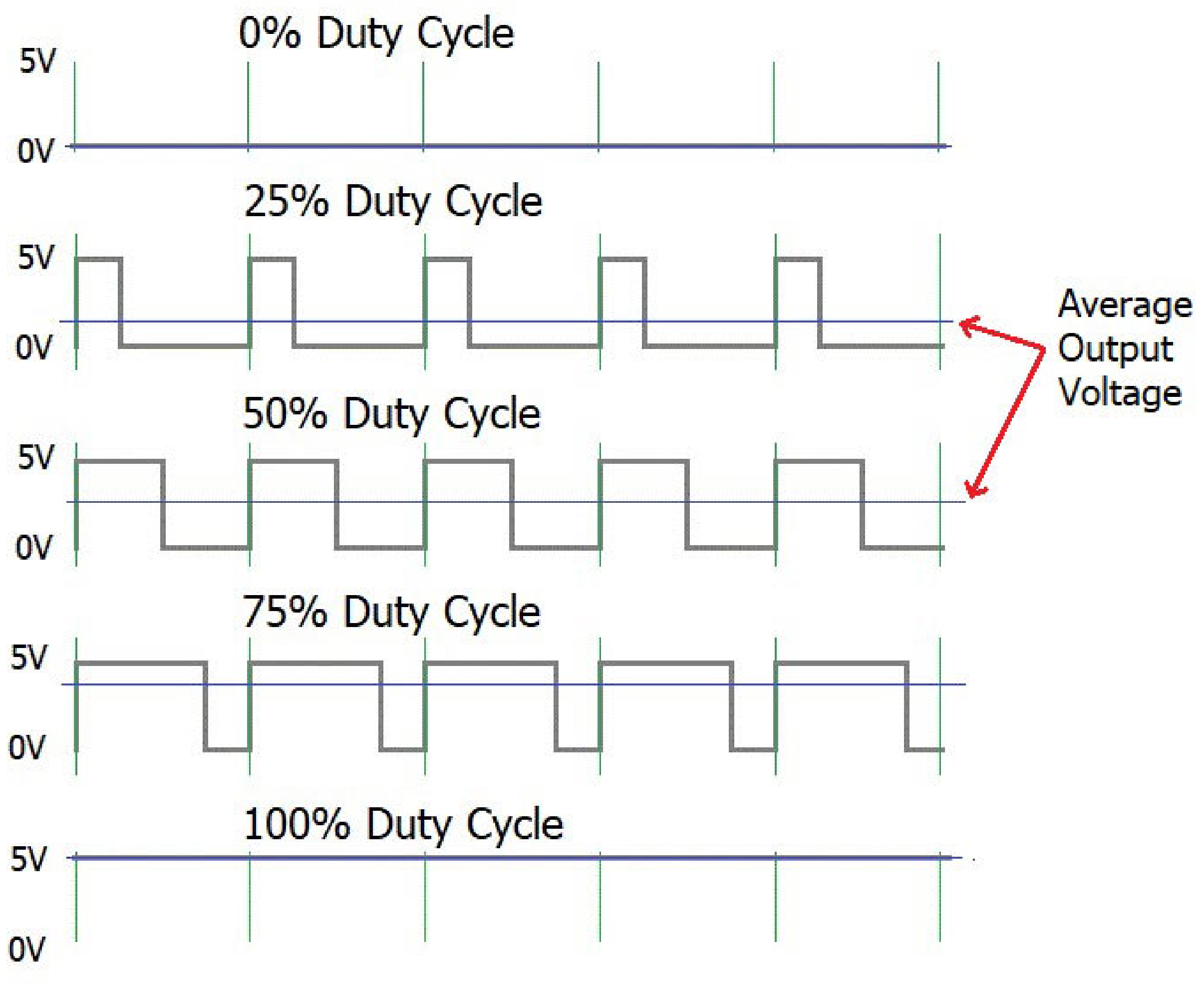
First let's look at how this whole process is happening. We all know about that every motor needs power source to work. So if you bought 3v motor you have to supply 3V for proper working. The passing current through the motor is depend on torque of the motor. If motor axel is working freely then it gets low amount of current. If motors are in difficult condition to rotate there axel then it gets higher amount of current. To control motors we used voltage sources. Voltage sources are sources which supply any amount of current under constant voltage. So the speed of motors will depend on the voltages. If we supply high voltage then motor will work at higher RPM and vice versa. Therefore now you can understand that we can control the speed by controlling the supply voltage level. But this is an old way and it's not efficient and accurate. Modernly we use PWM pulses. PWM means pulse width modulation. In this case we generate a square wave with some constant frequency to control the motors. So the lower state of this wave is nearly zero volt (0V) and the higher state of the wave is nearly supply voltage. Therefore we are able to use our full supply voltage to control the motors. But how can we control the motor speeds using this theorem. That is very easy. If we consider one cycle (duty cycle) of the wave that includes two parts. One is the High stage part and the other one is Low stage part. Let's say High stage time duration is T1, Low stage Time duration is T2 and one cycle time duration is T. So we can simply write the equation below.

$T=T1+T2$

If T2=0 and T=T1,then there isn't any lower state part in every cycle. In this case motor is working with full speed. Because we always give source voltage to motor. But if T1=0 and T=T2, there is no supply voltage and current to the motor. In this case motor is fully turned off. So the speed will be zero. Let's look at another situation. Lets say T1=T2 ,then T1=T2=T/2. So at this time both time periods of high and low states are equal to every cycle. Now the average value of the wave is half of the source power. Therefore the supply voltage to the motor is half of the main supply voltage(we don't know the variation between the supply voltage and the motor speed. because it depends on your motor.therefore we can't say the motor speed will be half of the maximum speed under half of supply voltage). In this way we can get every voltage between 0 and source voltage from average voltage by changing the ratio between

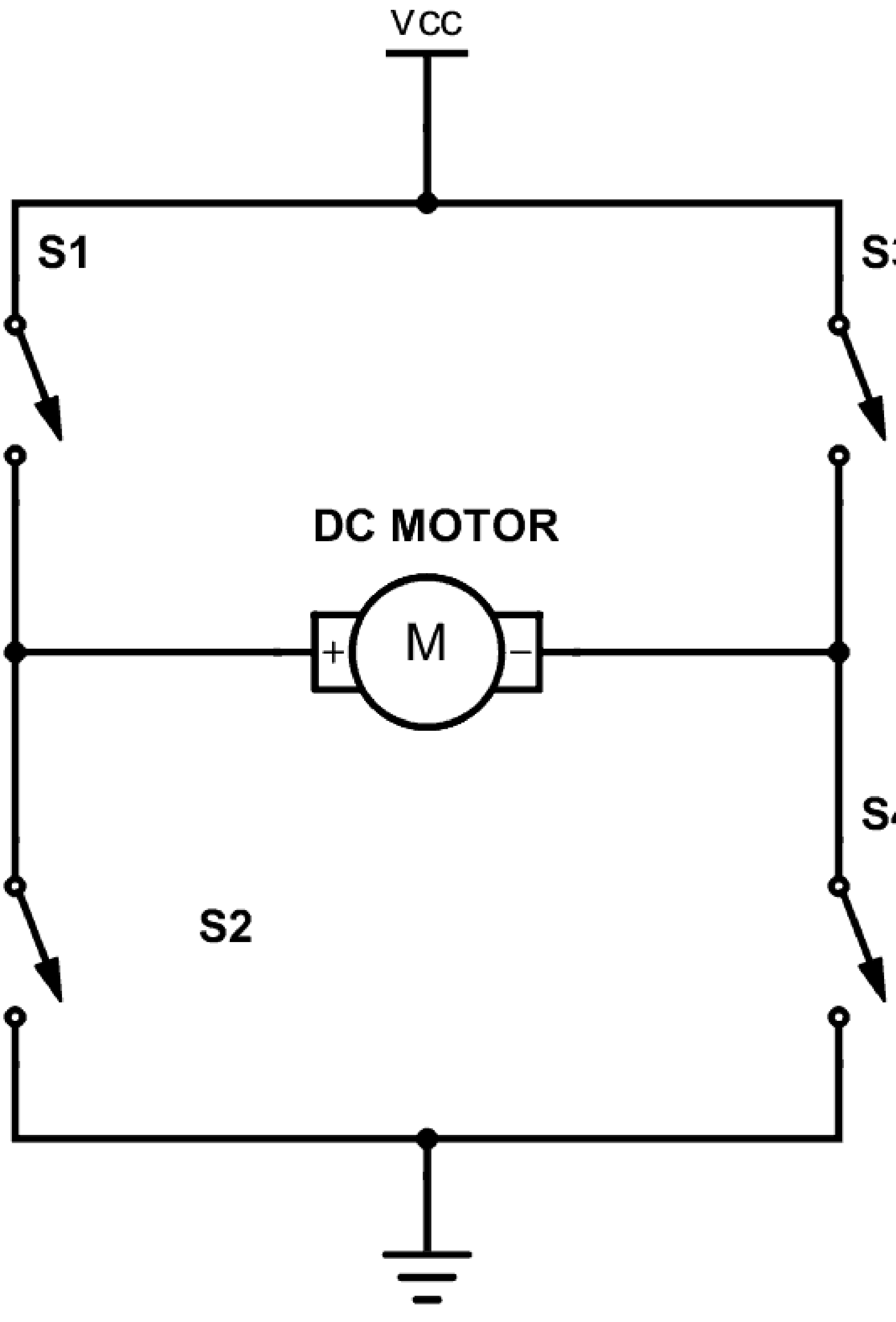
T1 and T2. To measure the PWM signal average, we use some factor value called duty cycle value. This value is a percentage value of the ratio of T1 and T.

Duty cycle=(T1/T)x100%

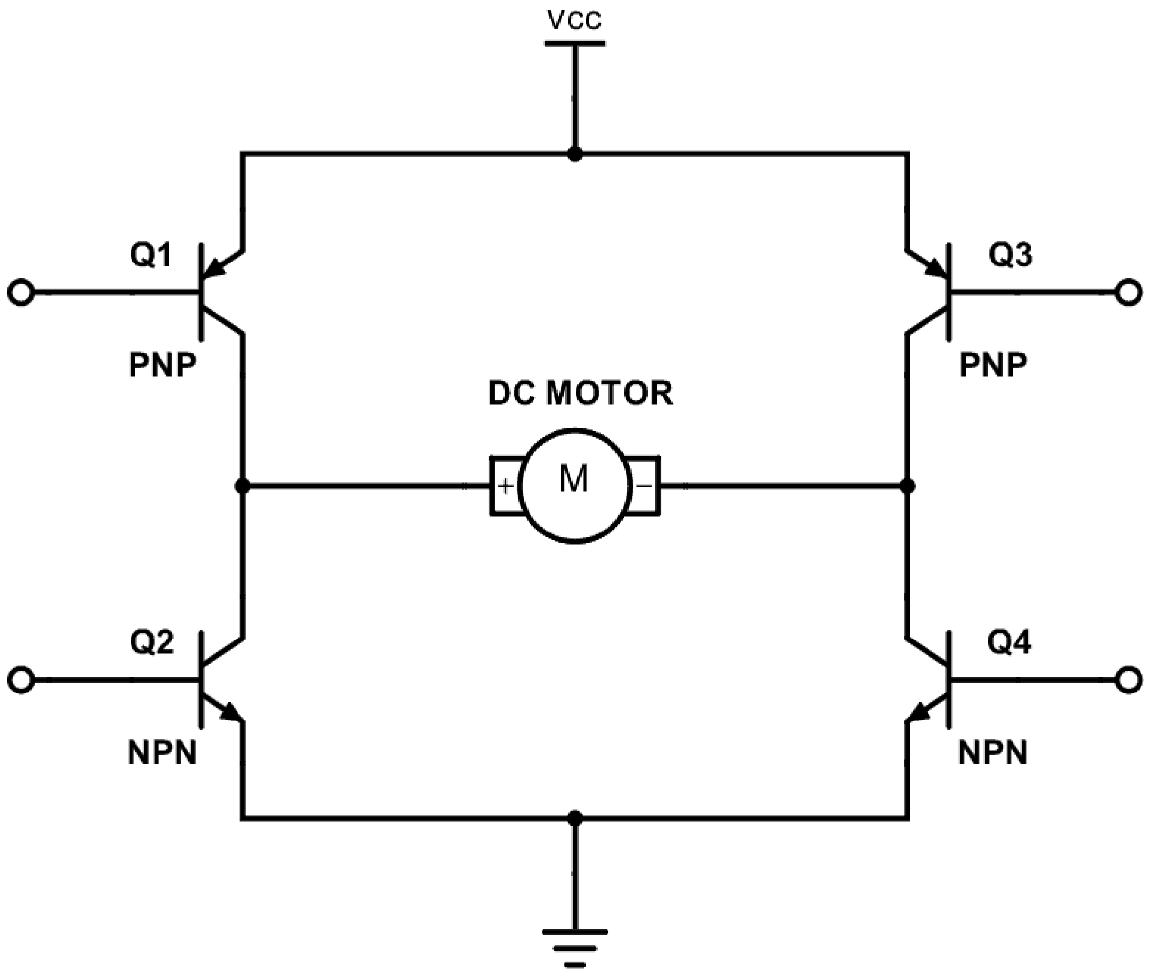


In the microcontrollers we represent this duty cycle value from bit value. If we use 8 bits then we can get 0-255 range to represent duty cycle. In that case 255 means 100% duty cycle and 0 means 0% duty cycle and so on. Now you have some general idea about motor control signals. These PWM signals are not limit to motor controlling applications. These signals are used for many purposes. Next look at how can we use this theorem to motor controlling process. To control motors we use a microcontroller to generate PWM signals. As the introduction describes, these voltage ranges and current of the PWM signals are not sufficient to control the motors. So we used motor driver for that purpose. All we know is that motors can rotate in two opposite directions with various speeds. Because of that reason we get two outputs from the microcontroller to control motor. If we want to rotate the motor in one direction then we use one output to generate a PWM signal while the other one is in low state. If we want that motor rotates in the opposite direction, in that case we use a second output pin to generate a PWM signal while setting the low state to the first pin. Because of the lack of current and voltage of this pin outputs, we will use motor controller unit. This unit includes an H-bridge switching mechanism. Let’s look at how it works. Before moving to that part, take a look at what transistor is. Transistor is a semiconductor device which is used to control signals. There are a lot of transistor types. But every transistor works in the same principle. A transistor has three pins. One pin is used to supply the signal. These signals can be voltage or current signals. The source current is going through other pins. According to the input signal this flowing current is changing. If input signal is larger than some defined value then the passing current will be maximized and if input signal is lower than some amount, then the passing current will be nearly neglectable. So these 2 situations are known as cutoff and saturation regions of the transistor. At these stages the transistor works as a switch. So if we connect microcontroller output into the transistor input signal, then at the high digital signal transistor will be on and at the low digital signal transistor will be off. Now you have a basic idea about transistor mechanisms.

So now let's look at the H-bridge mechanism.


VCC
S1
S3
DC MOTOR
M
+
–
S4
S2

As you can see there are four switches s1, s2, s3 and s4. These switches are actually transistors or some switching component. Let's analyse this diagram. If s1 and s4 are ON and others are OFF then, the motor will work in one direction. By changing the ON and OFF time of the S1 and S4 with some constant frequencies, we can rotate that motor with various speeds. If s3 and s2 are ON and others are OFF then the motor will go in the other direction. Also we can change the motor speed by changing the ON and OFF time of S2 and S3 switches. If all switches are OFF or all are ON then the motor will stop.



So now you can see how switches are replaced by transistors. Q1, Q2, Q3 and Q4 represents S1, S2, S3 and S4 switches. In this diagram the base pin is the input signal pin of the transistor. If we supply a high state signal to that pin the transistor will saturate and that transistor acts as a closed switch (ON). Otherwise it will act as an open (OFF) switch. In this diagram there are four inputs to control for transistors. But we combine these four inputs to two inputs which satisfy the above switching conditions.

In the Magicbit it includes L110 motor driver IC which has the ability to control two motors. So it is a two channel motor driver IC. It internally connected to the esp32 processor of the Magicbit from four. M1A, M1B, M2A and M2B are the pins of the lower port of the Magicbit which are output pins of the L9110 IC.

3.21.4 13.4 Methodology

Connect the motor to the M1A and M1B or M2A and M2B pins or connect two motors to the port on the left corner in the lower side of the Magicbit. Connect the Magicbit to your PC and upload the following code.

3.21.5 13.5 Coding

```
#include <ESP32Servo.h>
int M1A = 26; //motor drive input pins
int M1B = 2;
int M2A = 27;
int M2B = 4;
void setup() {
  pinMode(M1A, OUTPUT); //configure as inputs
  pinMode(M1B, OUTPUT);
  pinMode(M2A, OUTPUT);
  pinMode(M2B, OUTPUT);
}
void loop() {
  for (int i = 0; i <= 255; i++) { //rotate both motors to direction
    analogWrite(M1A, i);//pwm signal
    analogWrite(M1B, 0);
    analogWrite(M2A, i);
    analogWrite(M2B, 0);
    delay(100);
  }
  analogWrite(M1A, 255);//stop for 1 second
  analogWrite(M1B, 255);
  analogWrite(M2A, 255);
  analogWrite(M2B, 255);
  delay(1000);
  for (int i = 0; i <= 255; i++) { //rotate both motors to opposite direction
    analogWrite(M1A, 0);
    analogWrite(M1B, i);
    analogWrite(M2A, 0);
    analogWrite(M2B, i);
    delay(100);
  }
  analogWrite(M1A, 255);//stop for 1 second
  analogWrite(M1B, 255);
  analogWrite(M2A, 255);
  analogWrite(M2B, 255);
  delay(1000);
}
```