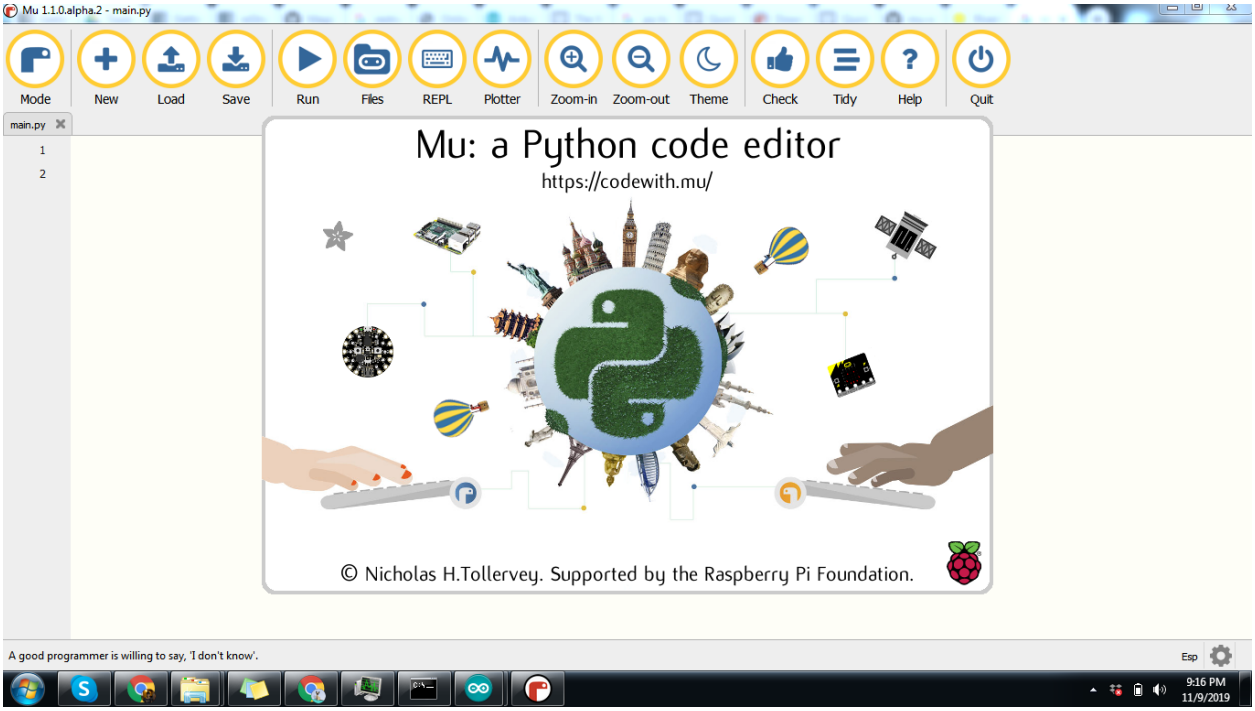
Magicbit-MicroPython Documentation

Release latest

Oct 14, 2020

Contents



MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimised to run on microcontrollers and in constrained environments. Learn more about MicroPython

Magicbit is based on ESP32 which in supported by MicroPython and Thonny Editor

- Download and install http://thonny.org/
- Open the Magicbit Utility and connect the Magicbit Board to the PC
- Choose the port connected to the device and choose Micropython as the platform
- Click 'Update Firmware'
- Open Thonny IDE
- Click Run > Select Interpreter
- Choose Micropython(ESP32) from the first dropdown box and leave the other dropdown box as it is.

CHAPTER 1

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.

CHAPTER 2

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

CHAPTER 3

First Project

- Open Thonny IDE
- Paste following code on editor and click save as

```
from machine import Pin
import time
led = Pin(16, Pin.OUT)
for i in range(10):
     led.on()
     time.sleep_ms(500)
     led.off()
     time.sleep_ms(500)
```

- Click 'Micropython Device' and choose file name as 'boot.py'

-Click Run (Green Arrow) on the IDE - If Green Led on backside of the Magicbit is blinking your have just begun the magic with Magicbit

3.1 Hardware

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 4

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

4.1 Example 2 : Reading the state of a push button

4.1.1 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.

Learning Outcomes

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

- IF-ELSE conditions
- Variables

Components

Magicbit

Theory

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.

Methodology

Magicbit is equipped with two onboard push buttons. Let us select the push button which is wired to D34. Buttons on the board are pulled up internally (to learn more about pullups/pulldowns follow this link), which means when the 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 the previous example we need to select a 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 Pin(), in this case button is an INPUT, LED is an OUTPUT.

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.

Coding

```
from machine import Pin

led = Pin(16, Pin.OUT)
btn = Pin(34, Pin.IN)

while True:
     if btn() == 0:
          led.on()
     else:
          led.off()
```

Explanation

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 immediately below it, if the condition is false it executes the code block in the else block.

```
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).

4.1.2 Example 3: Working with Analog Write

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.

Learning Outcomes

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

- Pulse Width Modulation

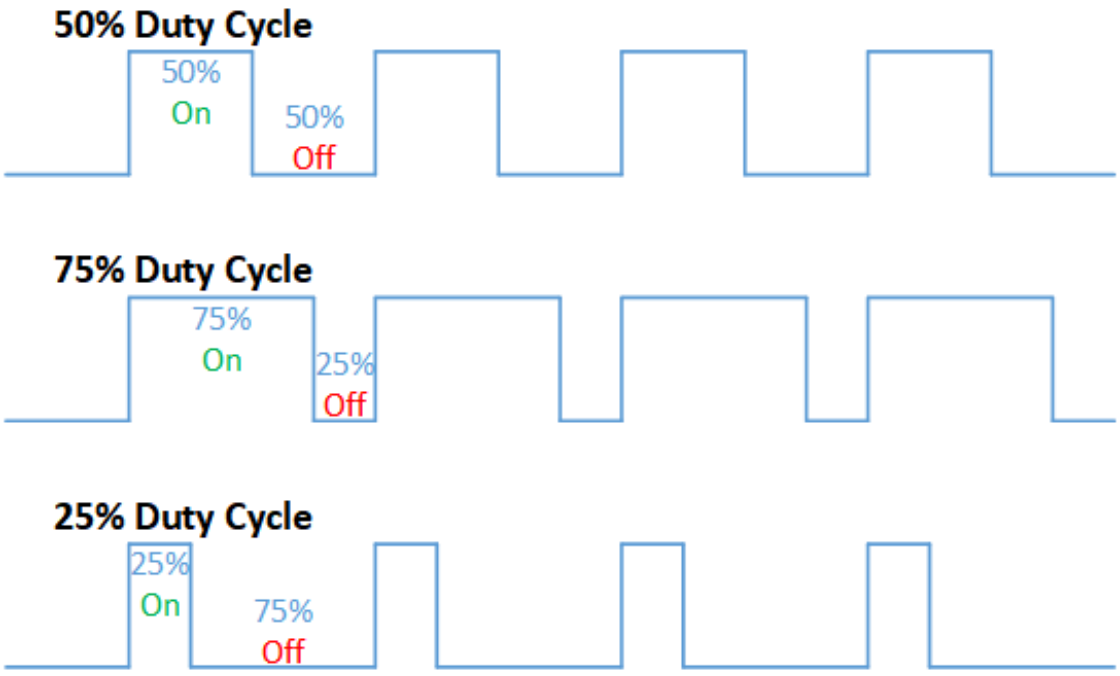
Components

- Magicbit

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.

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.

Coding

```
from machine import Pin,PWM
import time
LED=Pin(16, Pin.OUT)
pwm = PWM(LED)
pwm.freq(50)
while True:
  for i in range (0,256,1):
          pwm.duty(i)
          time.sleep_ms(500)
```

Explanation

for i in range (0,256,1): There are 3 parameters in a for loop, first parameter we are defining the starting value for the loop. Second parameter specifies the ending value for the loop, third parameter specifies the change happens to the variable in each cycle, in this i is incrementd by one.

pwm.duty(i) 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.

4.1.3 Example 4: Reading an Analog Signal

Introduction

In this example you are learning to read an analog sensor.

Learning Outcomes

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

- Analog Read

Components

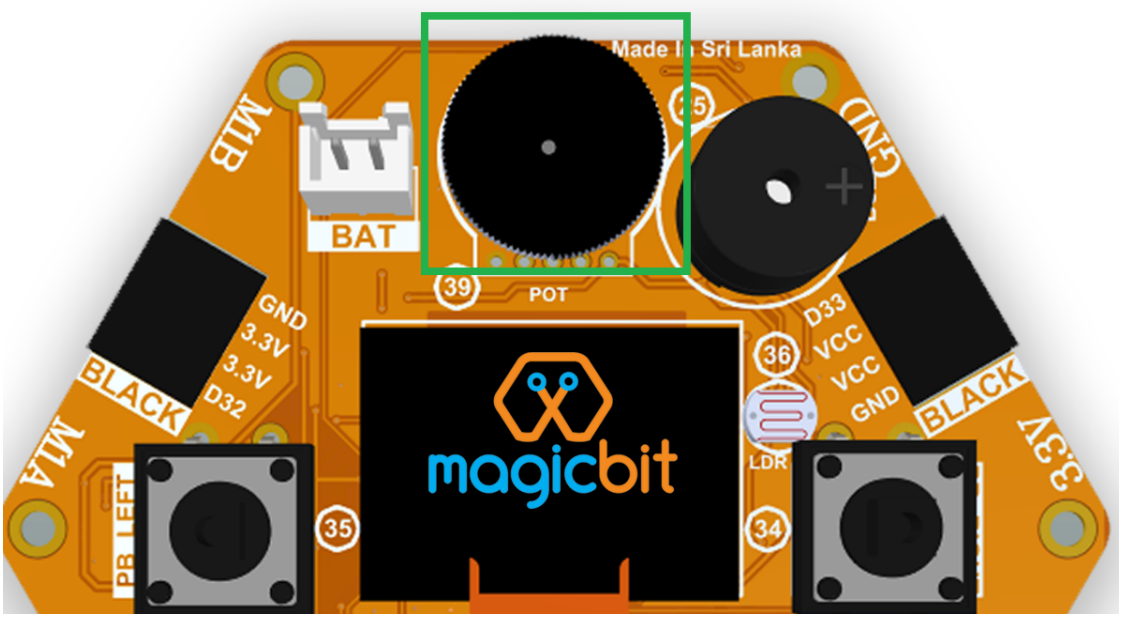
- Magicbit

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)

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 store it in an int type variable(0v= 0 analog value, 3.3v = 4096 analog value), sensorValue, later, we use this value to light up the red LED(D27) if the analog value exceeds than 2048.

Coding

```
from machine import Pin,ADC
LED=Pin(16,Pin.OUT)
adc=ADC(Pin(39))
while True:
    sensorValue=adc.read()
    if sensorValue>2048:
        LED.off()
    else:
        LED.on()
```

Explanation

adc.read(): this reads and assigns the corresponding analog value to the left.

Activity

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

4.1.4 Example 6: Generating Tones

Introduction

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

Learning Outcomes

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

- genearting square waves

Components

- Magicbit

Theory

Piezo buzzers are commonly used in embedded systems to give audible tones. It works well with a square wave as input.

Methodology

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

Coding

```
from machine import Pin,PWM
import time
BUZZER=Pin(25, Pin.OUT)
```

(continues on next page)

(continued from previous page)

```
while True:
      BUZZER.on()
      time.sleep_ms(300)
      BUZZER.off()
      time.sleep_ms(300)
```

Explanation

There is no function for micropython to generate a square wave to activate the buzzer. Therefore we should create a square wave to replicate the tone function used in arduino.

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.

4.1.5 Example 7: Using the onboard OLED Screen

Introduction

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

Learning Outcomes

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

- Using Adafruit OLED library

Components

- Magicbit

Theory

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

Methodology

Adafruit OLED library(Adafruit_SSD1306) 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.

Coding

```
from machine import Pin, I2C
import ssd1306
from time import sleep

# Magicbit Pin assignment
i2c = I2C(-1, scl=Pin(22), sda=Pin(21))


oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

oled.text('Hello, World 1!', 0, 0)
oled.text('Hello, World 2!', 0, 10)
oled.text('Hello, World 3!', 0, 20)

oled.show()
```

Explanation

oled_width , oled_height : Define the OLED width and height on the following variables:

oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c) : create an SSD1306_I2C object called oled. This object accepts the OLED width, height, and the I2C pins you've defined earlier.S1e2n3

oled.text(): The text() method accepts the following arguments (in this order):

- Message: must be of type String
- X position: where the text starts
- Y position: where the text is displayed vertically
- **Text color: it can be either black or white. The default color is white and this parameter is optional.**
  - 0 = black
  - 1 = white

oled.show(): To print on the screen.

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

4.1.6 01. Proximity Sensor

1.1 Introduction

In this example, you are learning how to use proximity sensor. This sensor (TCRT5000) uses reflection on a surface. It is known that a white and polished surface reflects large percentage of light and 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.

1.2 Components

- Magicbit
- Magicbit proximity sensor

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

1.4 Methodology

1. As the first step, you should connect a Magicbit proximity sensor to Magicbit core board.
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.

1.5 Coding

```
from machine import Pin, ADC
from time import sleep

prox = ADC(Pin(34))
prox.atten(ADC.ATTN_11DB)       #Full range: 3.3v

while True:
   prox_value = prox.read()
   print(prox_value)
   sleep(0.1)
```

4.1.7 02. Tilt Sensor

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 read and print output
- Working principle of the tilt sensor

2.2 Components

- Magicbit
- Magicbit Tilt Sensor

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.

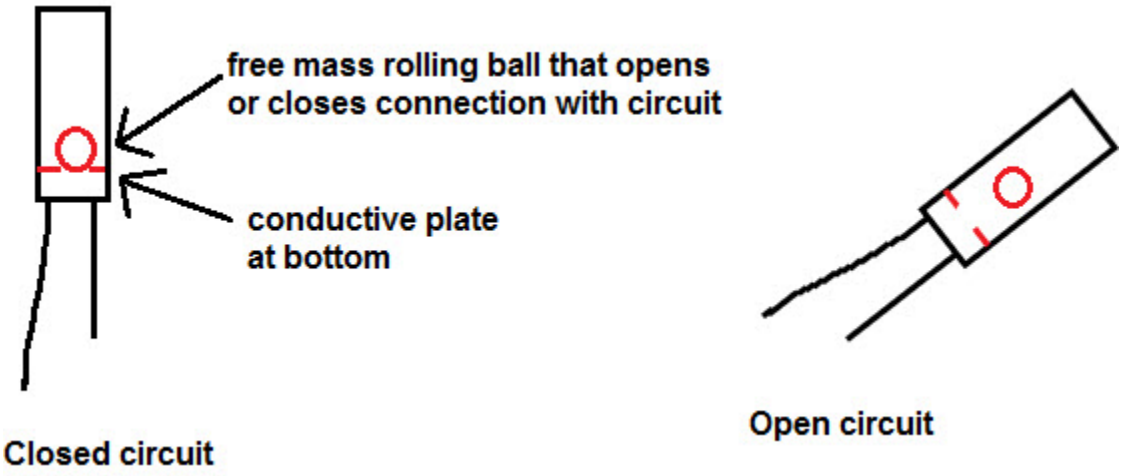


Figure 3: Working priciple of tilt sensor [1]

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.

2.5 Coding

```
from machine import Pin
import time

Tilt = Pin(32, Pin.IN)

while True:
   print(Tilt.value())
   time.sleep_ms(1000)
```

2.6 Explanation

Tilt = Pin(32, Pin.IN): Defining input pin **Tilt.value()** Read the data input of configured data pin.

4.1.8 03. Flame Sensor

3.1 Introduction

A flame sensor module 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 to identify infrareds/heat bodies

3.2 Components

- Magicbit
- Magicbit Flame Sensor

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.

3.4 Methodology

As the first step, you should connect a Magicbit flame sensor to Magicbit core board. For this, you can use one of the four side cnnectors 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.

3.5 Coding

```
from machine import Pin
import time

flame = ADC(Pin(32))
flame.atten(ADC.ATTN_11DB)       #Full range: 3.3v

while True:
  flame_value = flame.read()
  print(flame_value)
  sleep(0.1)
```

3.6 Explanation

Here we have used **print** , because we have to measure a range make the decision that there is a flame or not.

4.1.9 04. DOOR Sensor

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.

4.2 Components

- Magicbit
- Magicbit Magnetic door sensor

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

4.4 Coding

```
from machine import Pin
import time

door_sens = Pin(32, Pin.IN)

while True:
   print(door_sens.value())
   time.sleep_ms(1000)
```

4.5 Explanation

door_sens: Defined input pin for door sensor

4.1.10 05. Magicbit Servo

5.1 Introduction

A servomotor is an electrical device, which can push or rotate an object with great precision. If you want to rotate an 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

5.2 Components

- Magicbit
- Servomotor

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]

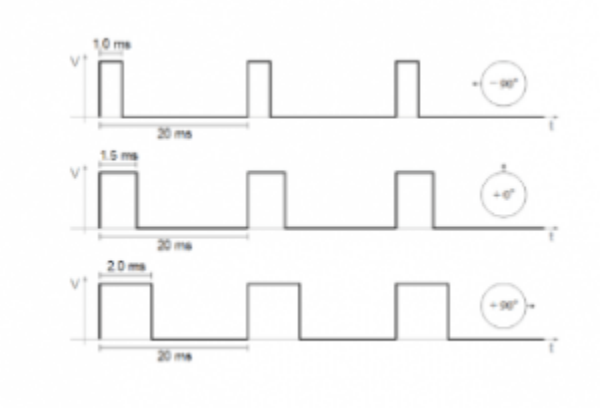


Figure 9:PWM Signals for various angles

5.4 Coding

```
import pyb

s1 = pyb.Servo(1)    # create a servo object on position P7
s2 = pyb.Servo(2)    # create a servo object on position P8

s1.angle(45)         # move servo 1 to 45 degrees
s2.angle(0)          # move servo 2 to 0 degrees

# move servo1 and servo2 synchronously, taking 1500ms
s1.angle(-60, 1500)
s2.angle(30, 1500)
```

4.1.11 06. Motion Sensor

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

6.2 Components

- Magicbit
- Magicbit Motion sensor

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.

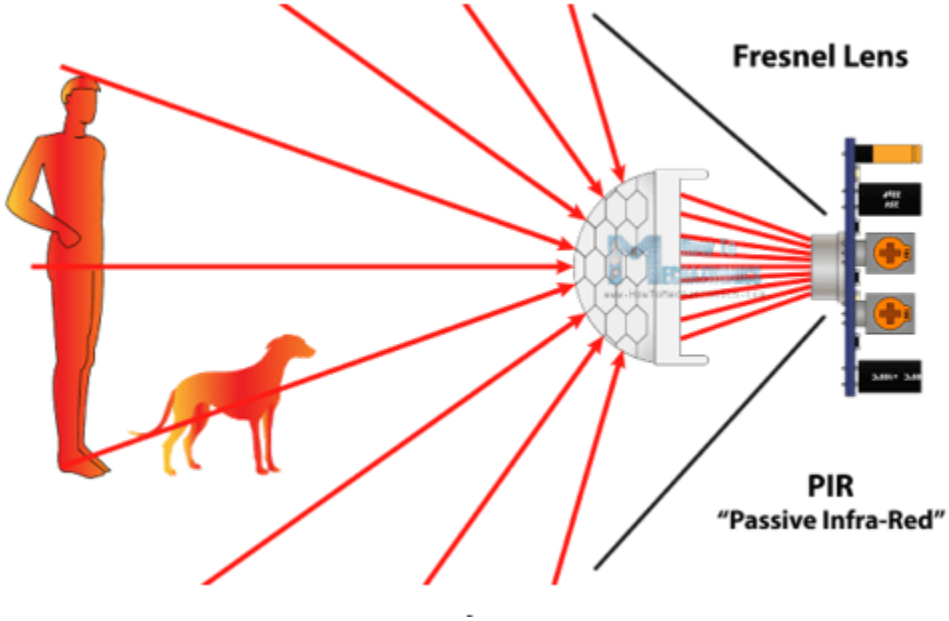


Figure 12: PIR Sensor –Howtomechatronics.com

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.

6.5 Coding

```
from machine import Pin
import time

motion_sens = Pin(32, Pin.IN)

while True:
   print(motion_sens.value())
   time.sleep_ms(1000)
```

6.6 Explanation

When some human being is detected by the motion sensor the output willdisplay '1' otherwise '0'.

4.1.12 07. RGB Module

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

7.2 Components

- Magicbit
- RGB module

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].

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.

7.5 Coding

```
from microbit import *
from random import randint

button_a=Pin(34, Pin.IN)
button_b=Pin(35, Pin.IN)
redVal = randint(0, 255)
greenVal = 0
blueVal = 0

while True:
if button_a.is_pressed():
        redVal = 0
        blueVal = 0
        greenVal = randint(0, 255)
        pin0.write_analog(redVal)
        pin1.write_analog(greenVal)
        pin2.write_analog(blueVal)
elif button_b.is_pressed():
        redVal = 0
        blueVal = randint(0, 255)
        greenVal = 0
        pin0.write_analog(redVal)
        pin1.write_analog(greenVal)
        pin2.write_analog(blueVal)
else:
        pin0.write_analog(redVal)
        pin1.write_analog(greenVal)
        pin2.write_analog(blueVal)
```

7.6 Explanation

You should see your LED turn on red. If you press the A button on the micro:bit, the color will change to green, and if you press the B button, the color will change to blue.

4.1.13 08. Magnetic Sensor

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

8.2 Components

- Magicbit
- Soil Moisture Sensor

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).

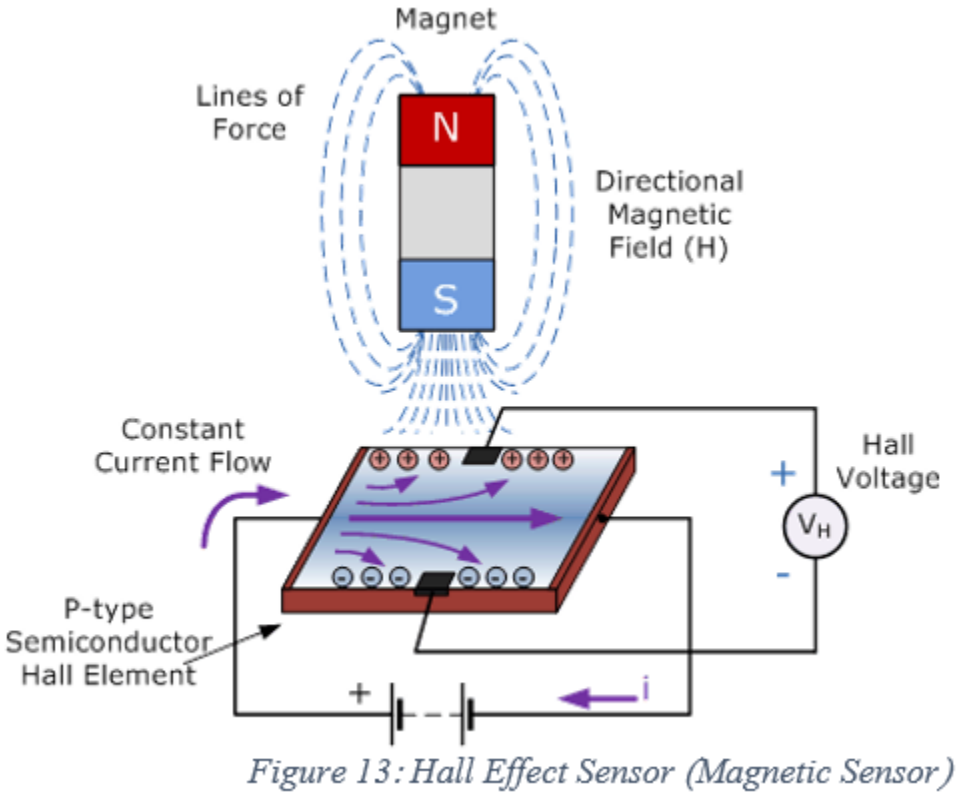


Figure 13: Hall Effect Sensor (Magnetic Sensor)

Figure 13: Hall Effect Sensor (Magnetic Sensor)

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.

8.5 Coding

```
from machine import Pin
import time

hall_sens = Pin(32, Pin.IN)

while True:
```

(continues on next page)

(continued from previous page)

```
    print(hall_sens.value())
    time.sleep_ms(1000)
```

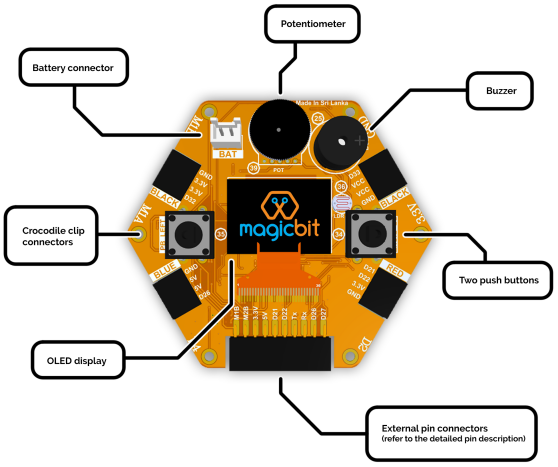
8.6 Explanation

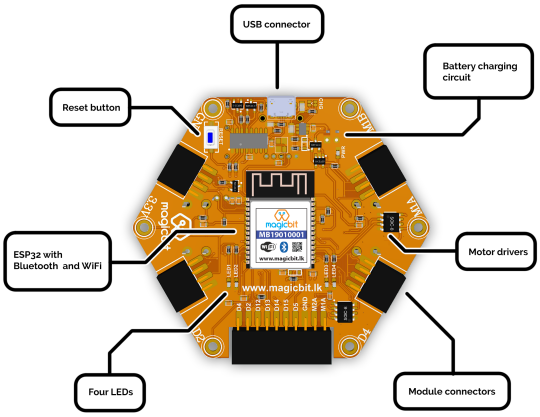
This Magnetic sensor gives digital outputs. Therefor you can open the serial monitor and see the outputs. '1' for occurred a magnetic field near to the sensor '0' for there is no any considerable magnetic field near by the sensor

4.1.14 Learning Outcomes

CHAPTER 5

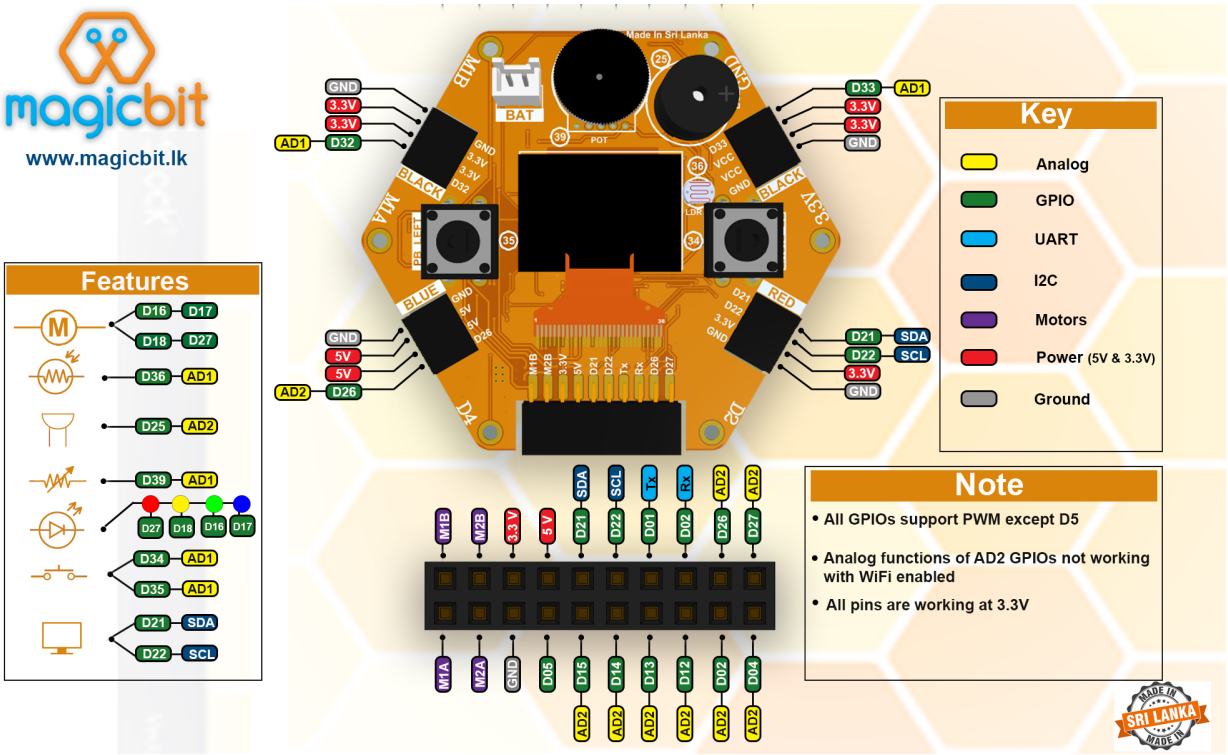
Features





CHAPTER 6

Pinmap



CHAPTER 7

Accessories

7.1 MicroPython Documentation

Detailed documentation about functions and usage can be found in original MicroPython documentation