

# Mini Project

## Reading Data from Sensors

**Prof. Sudip Misra**

Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Email: [smisra@sit.iitkgp.ernet.in](mailto:smisra@sit.iitkgp.ernet.in)

Website: <http://cse.iitkgp.ac.in/~smisra/>

Research Lab: [cse.iitkgp.ac.in/~smisra/swan/](http://cse.iitkgp.ac.in/~smisra/swan/)





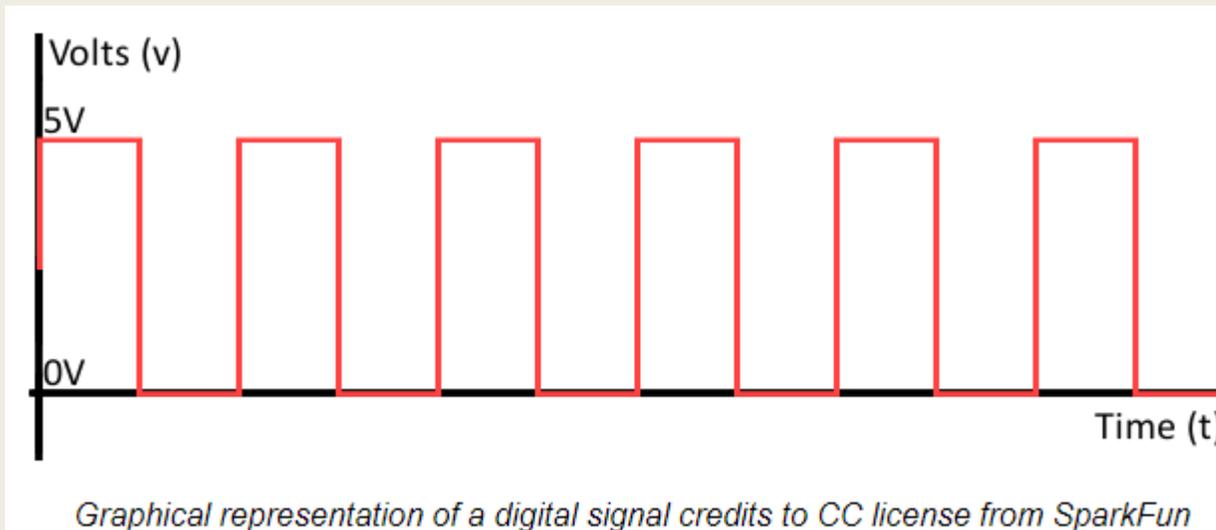
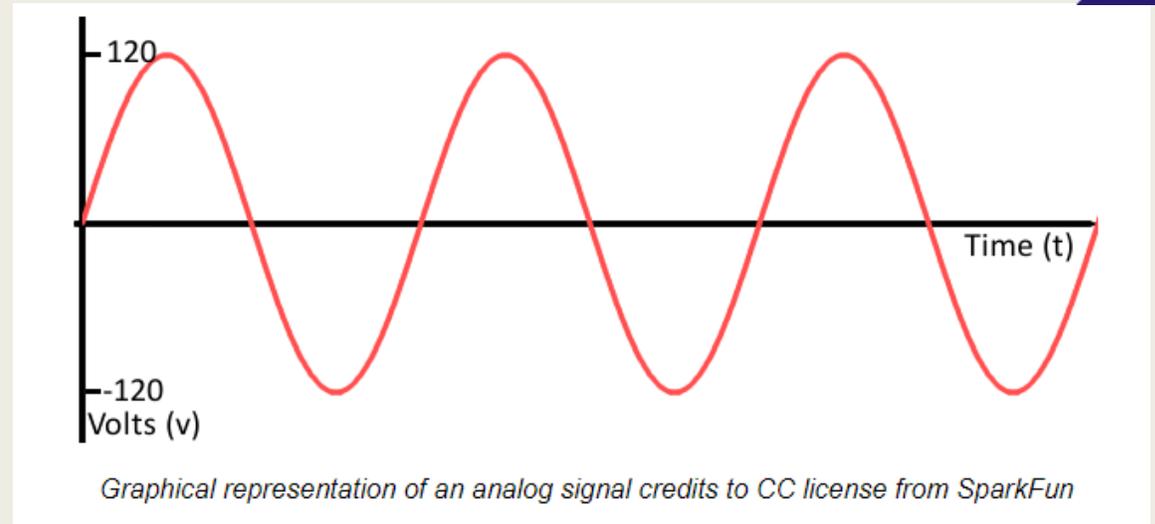
# Sensors - Analog and Digital

- A sensor is a device that **generates an output signal to detect a physical event**. In the broadest sense, a sensor is a **device, machine, module, or subsystem that senses changes or events in its surroundings and transmits the data to other electronics**, most commonly a micro-controller or a micro-processor.
- Sensors that generate a **continuous analog output signal** are termed as **Analog Sensors**.
- Sensors that generate a **discrete digital output signal** are termed as **Digital Sensors**.



# Signals - Analog and Digital

- **Analog signals** are continuous signals, meaning they are time-varying signals that reflect a quantity (e.g., current, voltage, or power) that changes over time.
- When we look at an analog signal graph, we will see a continuous plot with a defined value at each time point.



- A **digital signal** represents data as a sequence of discrete values; it can only take on one of a finite number of values at any given time. Digital signals are discrete signals with only two possible values: high and low (1 and 0).
- When we look at a digital signal graph, we observe a square wave that varies between two points and doesn't take any other value. A digital signal is a binary signal that can only take one of two values: 1 or 0.



# Sensors - Analog vs Digital

Analog Sensors	Digital Sensors
Produces continuous output signal or voltage which is proportional to quantity to be measured.	Produces discrete digital output signal or voltage which is digital representation of quantity to be measured.
Very susceptible to signal quality deterioration due to noise.	Immune to signal quality deterioration due to noise.
Lower Bandwidth	Higher Bandwidth
Consumes more power.	Consumes less power.
One signal line can transmit data from only one sensor.	One signal line can transmit data from multiple sensors.
Needs an ADC to be interfaced with a computer.	Can be directly interfaced with any computer (after agreeing upon a language).
No clock synchronisation is required.	Need to synchronize clocks if using multi bit data.





# Analog Sensor - LDR

- An **LDR** or a **Light Dependent Resistor** is a sensor that is used to sense the intensity of light.
- As the name suggests, an LDR changes its resistance based on the intensity of the light falling on it.
- LDRs are usually made of photosensitive semiconductor materials like Cadmium Sulphide (CdS), Lead Sulfide (PbS), Lead Selenide (PbSe), Indium Antimonide (InSb), or Cadmium Selenide (CdSe).
- The resistance of the LDR changes along with the intensity of light falling on it thereby changing the **voltage drop** across the LDR.



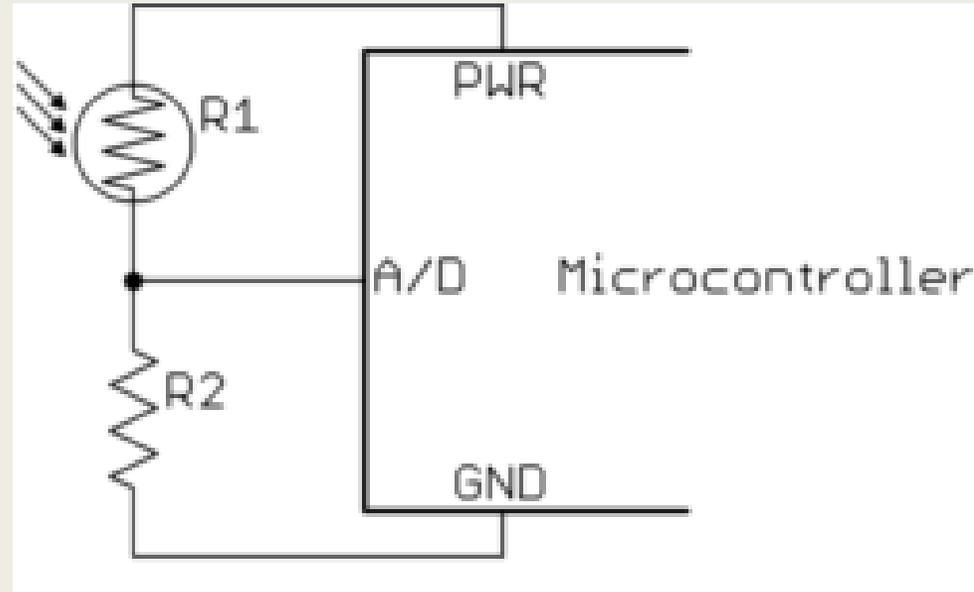
# Analog Sensor - LDR

- An **LDR** or a **Light Dependent Resistor** is a sensor that is used to sense the intensity of light.
- As the name suggests, an **LDR changes its resistance based on the intensity of the light** falling on it.
- LDRs are usually made of photosensitive semiconductor materials like Cadmium Sulphide (CdS), Lead Sulfide (PbS), Lead Selenide (PbSe), Indium Antimonide (InSb), or Cadmium Selenide (CdSe).
- The resistance of the LDR changes along with the intensity of light falling on it thereby changing the **current passing** through the LDR.



# Analog Sensor - LDR

- As a microcontroller can not measure current or resistance directly, we measure the **voltage drop** across a known resistor using the circuit below.



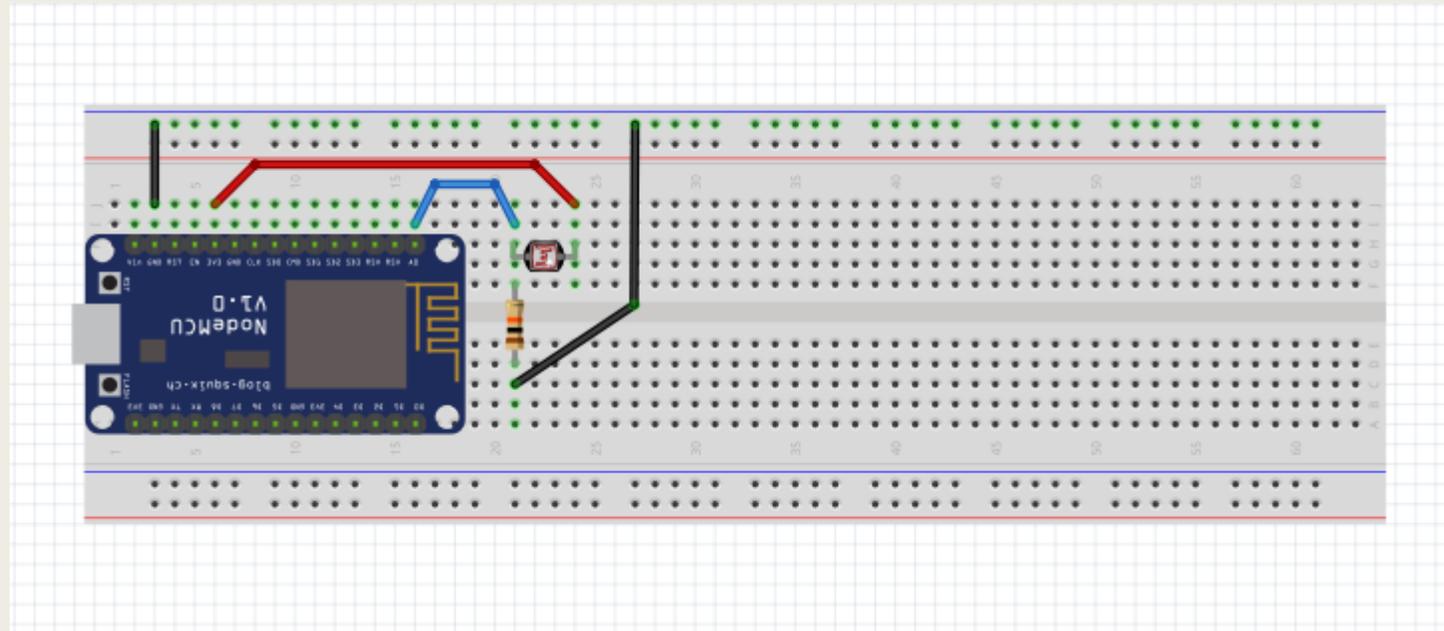
## **Further reading**

<https://electronics.stackexchange.com/questions/70009/why-use-a-pull-down-resistor-with-a-ldr-and-microcontroller>

# Building the circuit

1. Connect the components like the diagram below.

**Attention:** Please disconnect the USB Cable from your NodeMCU before making the connections on the breadboard.



# Code to read Analog Data

1. *First we import the required modules.*

```
Thonny - - <untitled> @ 4:24
File Edit View Run Tools Help
[Icons]
main1.py * x <untitled> * x <untitled> * x
1 from machine import ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5
6 while True:
7
8     ldr_value = ldr_pin.read()
9     print("LDR Value:", ldr_value)
10    sleep(1)
```

*More information on Micropython libraries can be found in the link below:*  
<https://docs.micropython.org/en/latest/library/index.html>

# Code to read Analog Data

2. Then we create an instance of the **ADC** class with pin number **“0”** (referring the **“A0”** pin of NodeMCU) set as an output pin. It is then assigned to the variable **ldr\_pin**.

```
Thonny - <untitled> @ 4:24
File Edit View Run Tools Help
main1.py * <untitled> * <untitled> *
1 from machine import ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5
6 while True:
7
8     ldr_value = ldr_pin.read()
9     print("LDR Value:", ldr_value)
10    sleep(1)
```

# Code to read Analog Data

3. Then we create an infinite loop and inside it, we read the LDR value and then we print the instantaneous LDR value with a delay of 1 second.

```
Thonny - <untitled> @ 4:24
File Edit View Run Tools Help
main1.py * x <untitled> * x <untitled> * x
1 from machine import ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5
6 while True:
7
8     ldr_value = ldr_pin.read()
9     print("LDR Value:", ldr_value)
10    sleep(1)
```

# Uploading the Code

1. Click on the **FILE** menu as shown.

```
Thonny - <untitled> @ 9:28
File Edit View Run Tools Help
<untitled> *x <untitled> *x
1 from machine import Pin, ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5 led_pin = Pin(5, Pin.OUT)
6
7 while True:
8
9     ldr_value = ldr_pin.read()
10    print("LDR Value:", ldr_value)
11
12    if ldr_value <= 500:
13        led_pin(1) # Turn on the LED
14    else:
15        led_pin(0)
16
17    sleep(1)
```

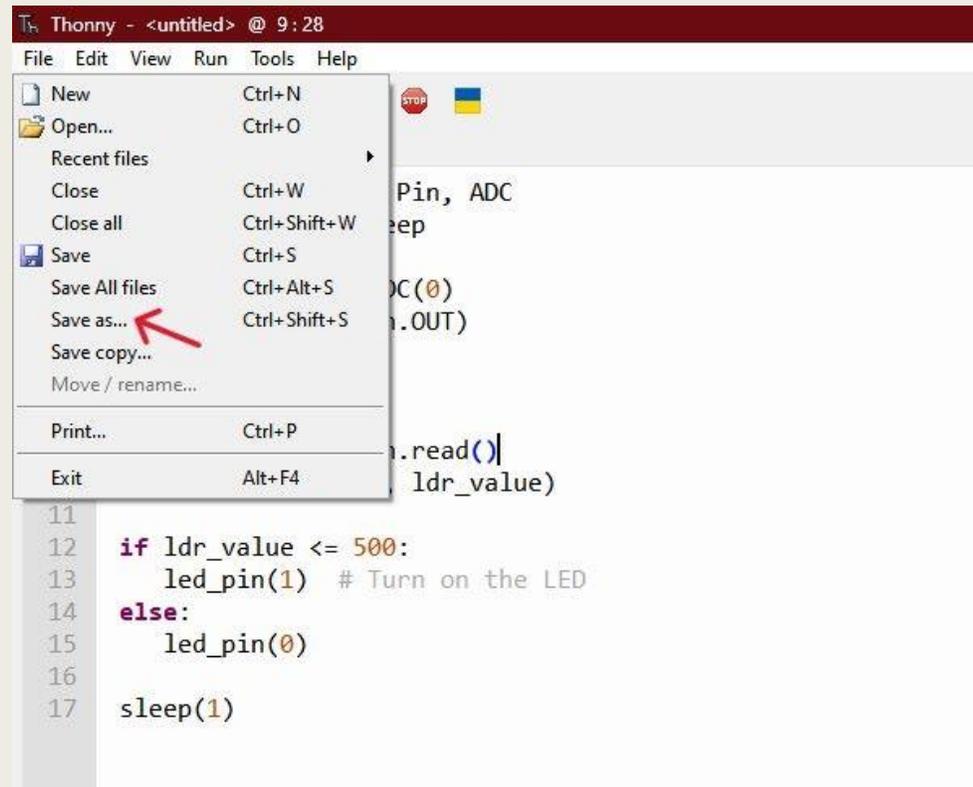
# Uploading the Code

2. Click on **“Save as”** from a list of options in the file menu.

```
Thonny - <untitled> @ 9:28
File Edit View Run Tools Help
<untitled> * x <untitled> * x
1 from machine import Pin, ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5 led_pin = Pin(5, Pin.OUT)
6
7 while True:
8
9     ldr_value = ldr_pin.read()
10    print("LDR Value:", ldr_value)
11
12    if ldr_value <= 500:
13        led_pin(1) # Turn on the LED
14    else:
15        led_pin(0)
16
17    sleep(1)
```

# Uploading the Code

2. Click on **“Save as”** from the list of options in the file menu.



The screenshot shows the Thonny IDE interface. The 'File' menu is open, and the 'Save as...' option is highlighted with a red arrow. The menu items and their keyboard shortcuts are as follows:

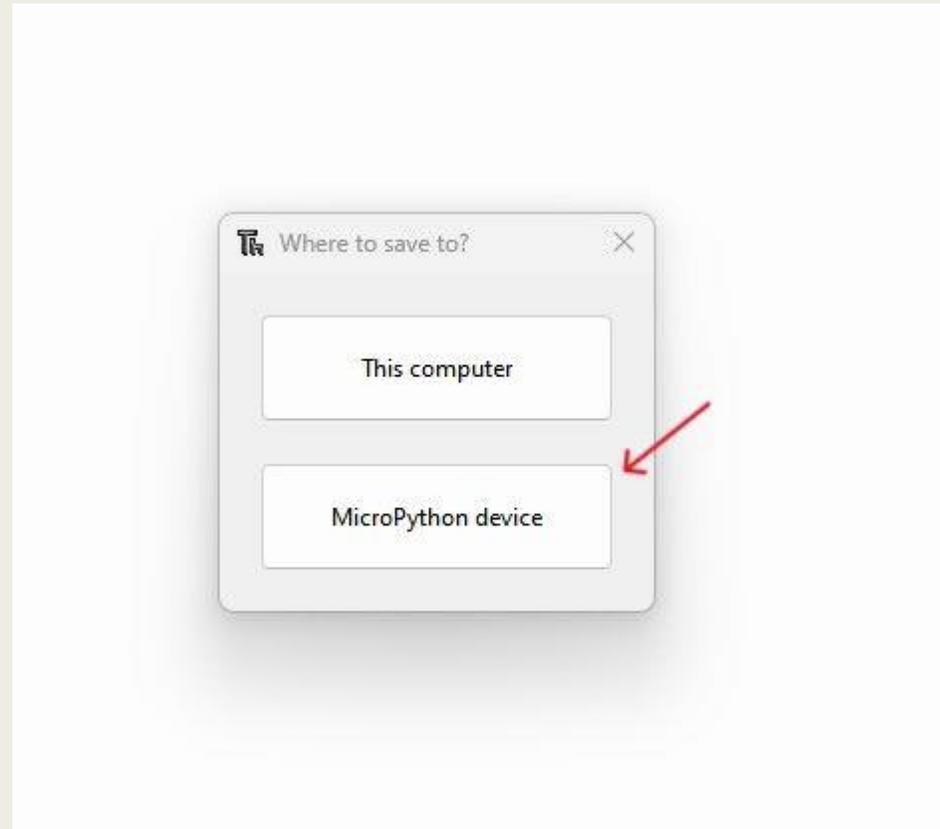
File Menu Item	Keyboard Shortcut
New	Ctrl+N
Open...	Ctrl+O
Recent files	
Close	Ctrl+W
Close all	Ctrl+Shift+W
Save	Ctrl+S
Save All files	Ctrl+Alt+S
Save as...	Ctrl+Shift+S
Save copy...	
Move / rename...	
Print...	Ctrl+P
Exit	Alt+F4

The background code in the editor is as follows:

```
11  
12 if ldr_value <= 500:  
13     led_pin(1) # Turn on the LED  
14 else:  
15     led_pin(0)  
16  
17 sleep(1)
```

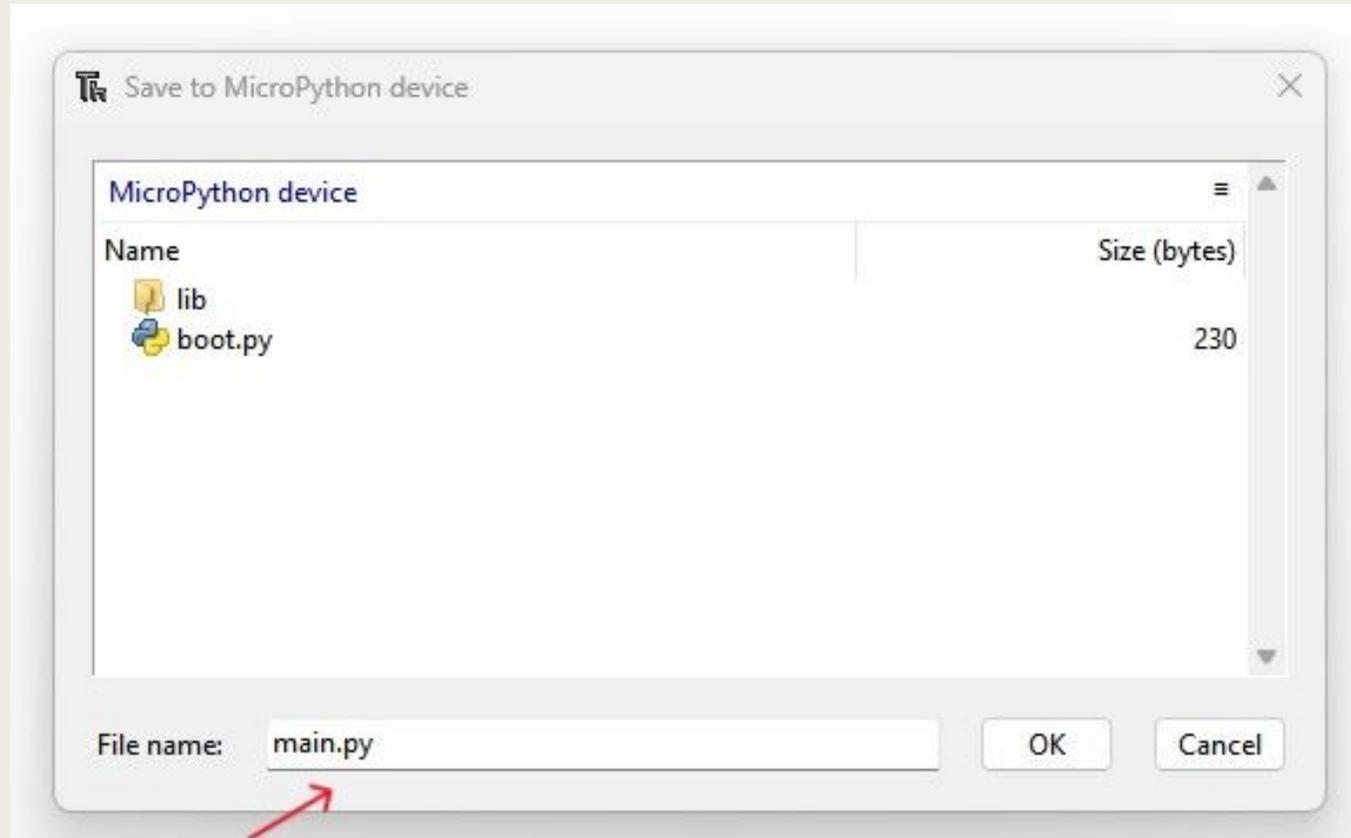
# Uploading the Code

3. Select the location as **“MicroPython device”**.



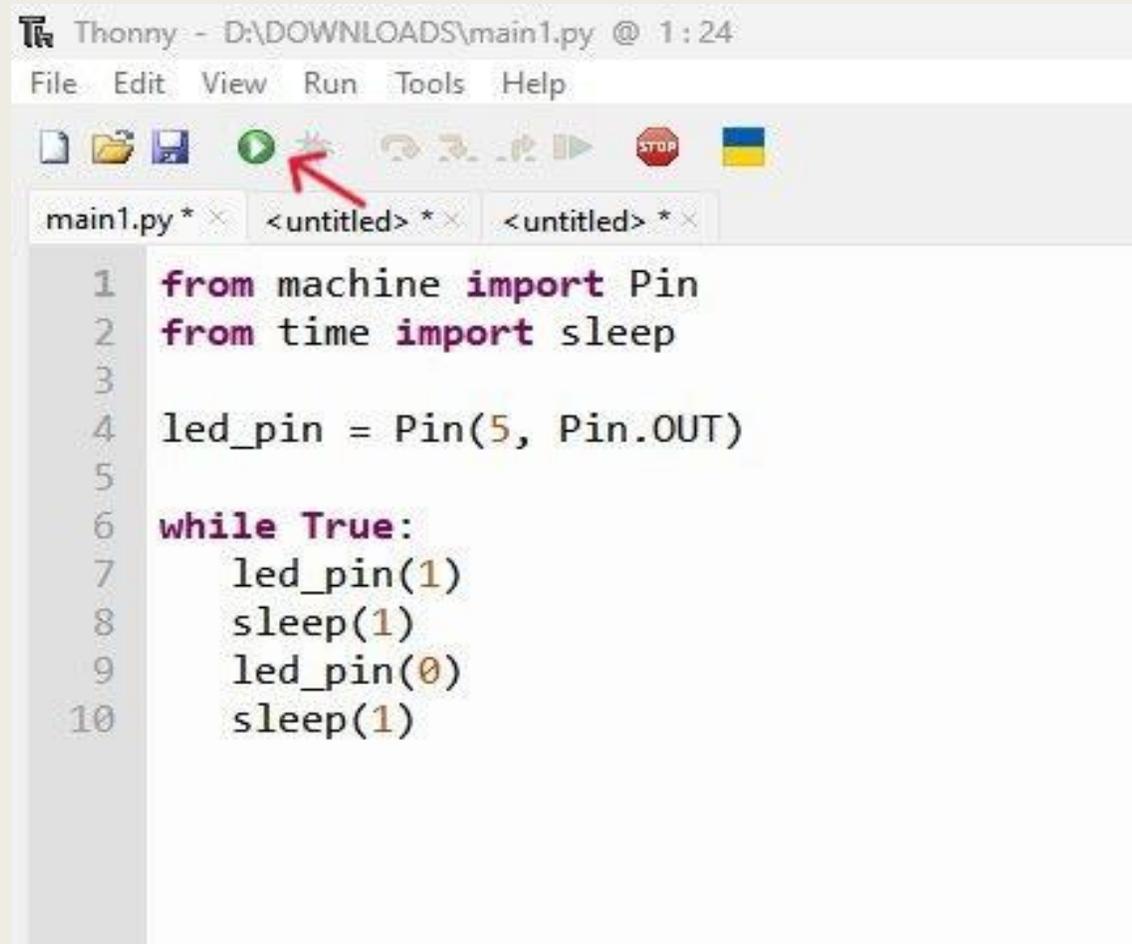
# Uploading the Code

4. Give the name “**main.py**” to the file and then click on the **OK** button to save.



# Uploading the Code

5. Then click on the **RUN** button as shown. Your code will start executing.



```
Thonny - D:\DOWNLOADS\main1.py @ 1:24
File Edit View Run Tools Help
[Icons: File, Folder, Save, Run, Stop, Refresh, Undo, Redo, Play, Stop, Flag]
main1.py * x <untitled> * x <untitled> * x
1 from machine import Pin
2 from time import sleep
3
4 led_pin = Pin(5, Pin.OUT)
5
6 while True:
7     led_pin(1)
8     sleep(1)
9     led_pin(0)
10    sleep(1)
```

# Observing the LDR values in Shell Window

1. We will find instantaneous LDR values in the shell window with a delay of 1 second. These values will change as the amount of light falling on the LDR changes.

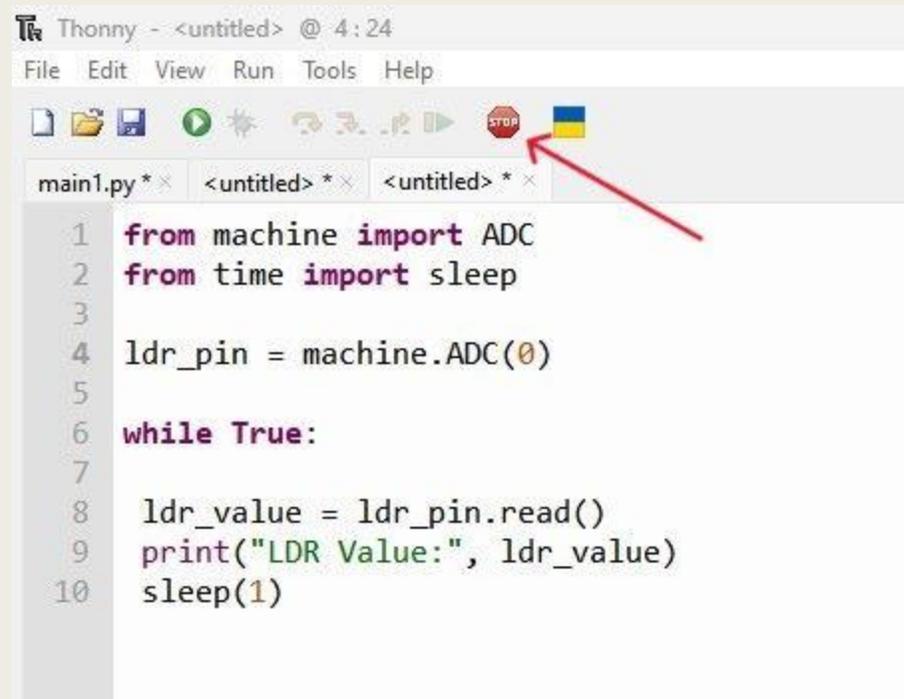
```
Shell x
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
LDR Value: 430
LDR Value: 452
LDR Value: 465
LDR Value: 449
```

A red bracket and arrow on the right side of the terminal output highlight the four LDR value lines.

# Observing the LDR values in Shell Window

2. After our examining of LDR values we can click on the **STOP** button to stop executing the code.



```
Thonny - <untitled> @ 4:24
File Edit View Run Tools Help
main1.py * <untitled> * <untitled> *
1 from machine import ADC
2 from time import sleep
3
4 ldr_pin = machine.ADC(0)
5
6 while True:
7
8     ldr_value = ldr_pin.read()
9     print("LDR Value:", ldr_value)
10    sleep(1)
```

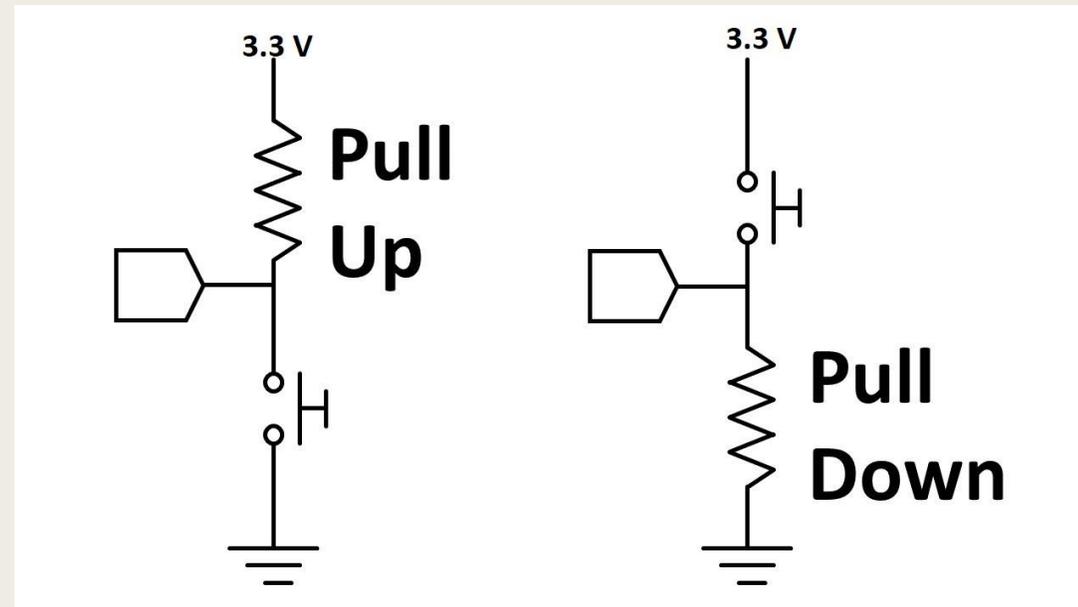
# Digital Sensor – Tactile Switch

- *An electric switch is a device – usually electromechanical – used to open and close an electric circuit. This disables and enables the flow of electric current, respectively.*
- *Using this behavior of a switch, we can easily detect the presence of certain objects. We can detect button presses (Keypad Inputs), we can use them to restrict the movement of certain actuators (Limit Switches/Endstop Switches) etc.*



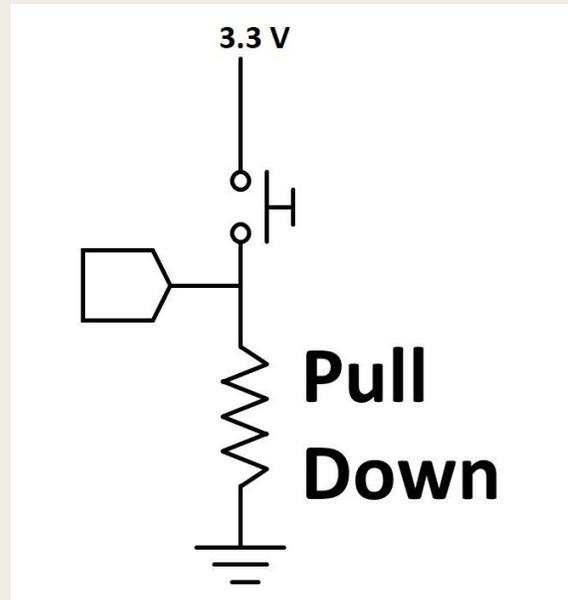
# Connecting a Switch

- Depending upon the connection configuration, a switch can either provide 0 or provide 1 when it is pressed.
- We need **PULL-UP** or **PULL-DOWN** resistors to connect a switch with a microcontroller. Otherwise, we risk damaging the input pin of the microcontroller by allowing a huge amount of current to flow to it.



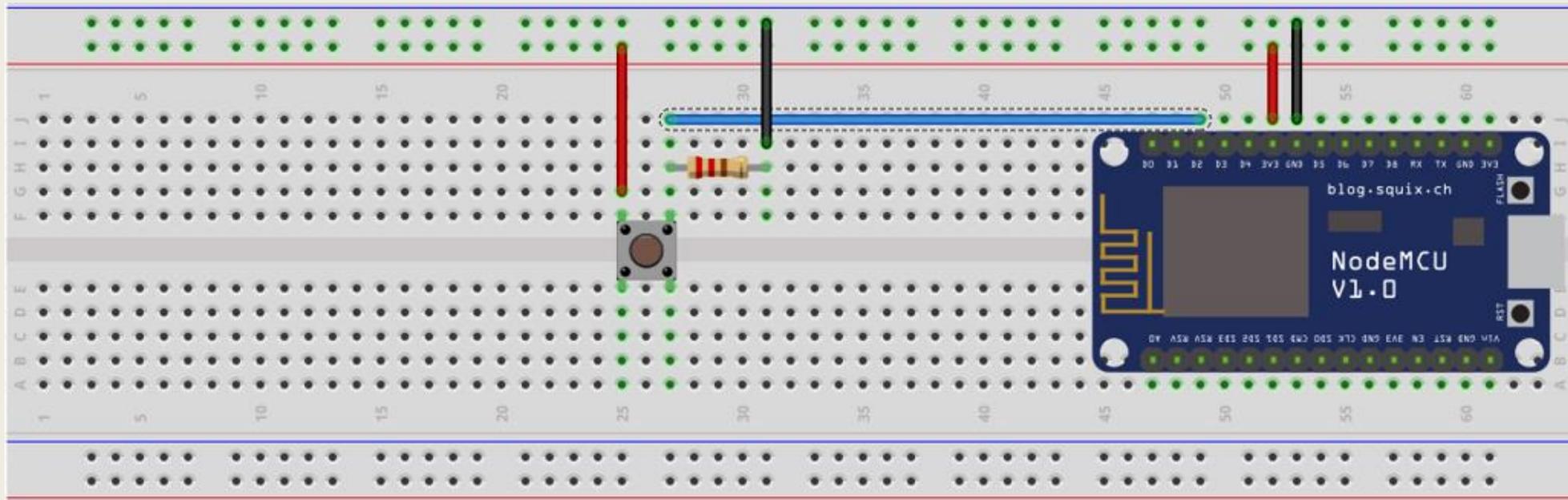
# Connecting a Switch

- *We will be using the PULL-DOWN mode.*
- *In this mode, if the switch is pressed, it will give OUTPUT as 1 (True/High)*
- *When the switch is not pressed, it will give OUTPUT as 0 (False/Low).*



# Connecting a Switch

- Place the NodeMCU as shown in the connection.
- Place the switch in a manner that you can connect the resistor in the pull down mode.
- We will use **D2** pin of the NodeMCU which is the **GPIO4** pin.
- Provide the 3.3V connection and GND connection as shown.



# Code to read the state of a switch

```
Thonny - MicroPython device :: /main.py @ 13 : 13
File Edit View Run Tools Help
[ main.py ] x
1 from machine import Pin
2 from time import sleep
3
4
5 push_button = Pin(4, Pin.IN)
6
7 while True:
8     logic_state = push_button.value()
9     if logic_state == True:
10        print('Switch is Pressed')
11    else:
12        print('Switch is Not Pressed')
13    sleep(0.1)
```



**Thank You!!!**