SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203.

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

EC3564- EMBEDDED SYSTEMS AND IOT DESIGN THEORY CUM LABOROTORY

LAB MANUAL

Regulation-2023

III YEAR / V SEM B.E ECE (2025- 2026 ODD SEMESTER)

Prepared by

Mr.K.R.Ganesh., Assistant Professor/ ECE

Dr.C.Amali., Associate Professor/ECE

Mr.S.Manikandan., Assistant Professor/ ECE

Dr.K.Lekha., Assistant Professor/ ECE

NAME	: ______________________
REGISTER NO	: ______________________
YEAR/SEMESTER	: ______________________

ABOUT OBSERVATION NOTES & PREPARATION OF RECORD

- This Observation contains the basic diagrams of the circuits enlisted in the syllabus of the **EC3564 - EMBEDDED SYSTEMS AND IOT DESIGN** course, along with the design of various components of the circuit and controller.
- The experiment's aim is also given at the beginning of each experiment. Once the student can design the circuit as per the circuit diagram, they are supposed to go through the instructions carefully and do the experiments step by step.
- They should note down the readings (observations) and tabulate them as specified.
- It is also expected that the students prepare the theory relevant to the experiment referring to prescribed reference books/journals in advance, and carry out the experiment after understanding thoroughly the concept and procedure of the experiment.
- They should get their observations verified and signed by the staff within two days and prepare & submit the record of the experiment when they come to the laboratory in the subsequent week.
- The record should contain experiment No., Date, Aim, Apparatus required, Theory, Procedure, and result on one side (i.e., Right-hand side, where rulings are provided) and Circuit diagram, Design, Model Graphs, Tabulations, and Calculations on the other side (i.e., Left-hand side, where no rulings are provided)
- All the diagrams and table lines should be drawn in pencil
- The students are directed to discuss & clarify their doubts with the staff members as and when required. They are also directed to follow strictly the guidelines specified.

EC3564 - EMBEDDED SYSTEMS AND IOT DESIGN

Theory cum laboratory

LABORATORY SYLLABUS

LIST OF PRACTICAL EXPERIMENTS:

- Programming Arithmetic and Logical Operations in 8051.
- Implementation of IoT using Python Programming.
- Study of ARM processor.
- Implementation of IoT using BLYNK - –Installation and Activation - Blinking an LED.
- Monitoring of temperature using sensor and Softcore Processors.
- Study of microcontrollers used in IoT.
- Logging of data in cloud using IoT.
- WAP for LED blink.
- Design of IoT System.

LIST OF EXPERIMENTS

INTRODUCTION TO KEIL µ VISION 5

INTRODUCTION TO KEIL µ VISION 5 SOFTWARE

The µVision5 IDE is a Windows-based software development platform that combines a robust editor, project manager, and make facility. µVision5 integrates all tools, including the C compiler, macro assembler, linker/locator, and HEX file generator. µVision4 helps expedite the development process of your embedded applications by providing the following:

- ✓ Full-featured source code editor
- ✓ Device database for configuring the development tool set
- ✓ Project manager for creating and maintaining your projects
- ✓ Integrated make facility for assembling, compiling, and linking your embedded applications
- ✓ Dialogs for all development tool settings
- ✓ True integrated source-level Debugger with high-speed CPU and peripheral simulator
- ✓ Advanced GDI interface for software debugging in the target hardware and for connection to Keil ULINK
- ✓ Flash programming utility for downloading the application program into Flash ROM
- ✓ Links to development tools manuals, device datasheets & users' guides

The Keil µVision5 IDE offers numerous features and advantages that help you quickly and successfully develop embedded applications. It is easy to use and guaranteed to help you achieve your design goals.

The installation steps for Keil software are given below:

1. Double click on Keil µvision4.exe file.
2. Then click on **Next**.
3. Tick the check box towards to license agreements and click **Next**.
4. Select Destination folder and click **Next**.
5. Fill the necessary text boxes and click **Next**.
6. Finally click on **Finish**.

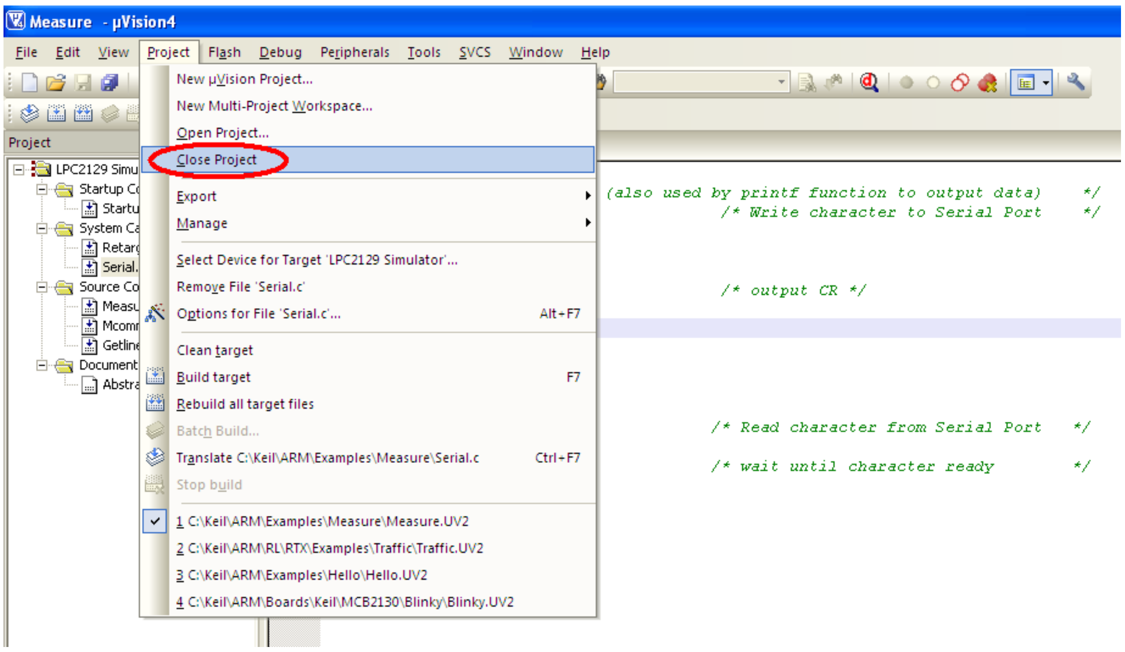
Software Flow

First open the icon keil µvision4 and the follow the steps are given below. The menu bar provides you with menus for editor operations, project maintenance, development tool option settings, program debugging, external tool control, window selection and manipulation, and on-line help. The toolbar buttons allow you to rapidly execute µVision4 commands. A Status Bar provides editor and debugger information. The various toolbars and the status bar can be enabled or disabled from the View Menu commands.

Creating a New Project

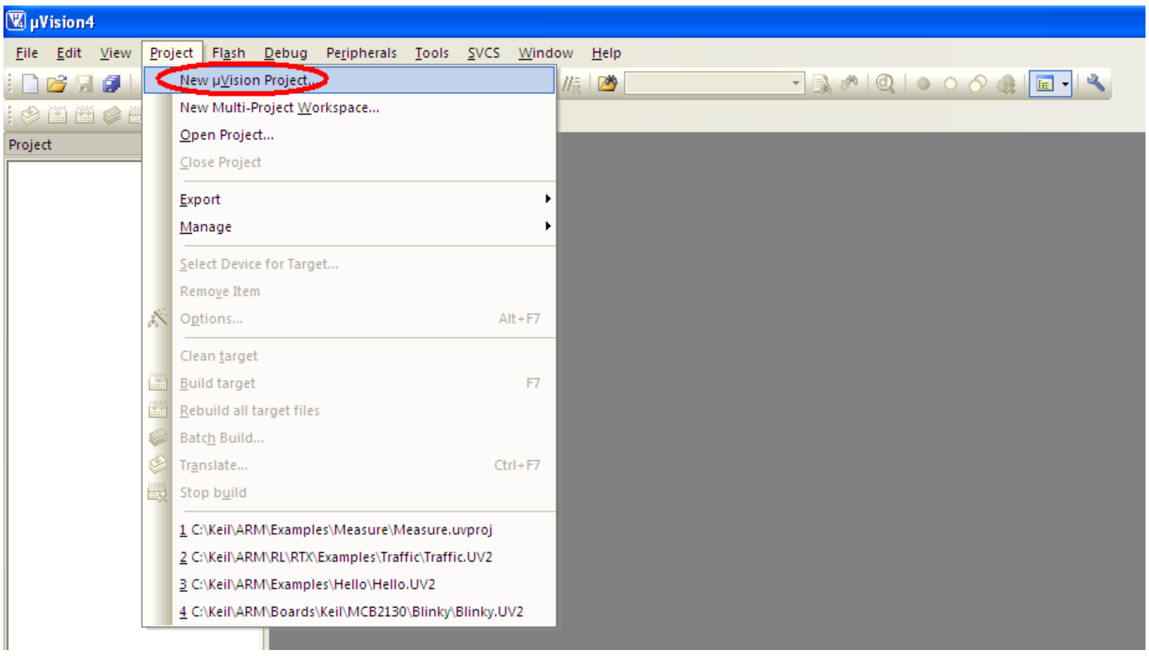
The mentioned procedures will explain the steps required to set up a simple application and to generate a HEX output.

STEP 1: Go to "**Project**" and close the current project "**Close Project**".

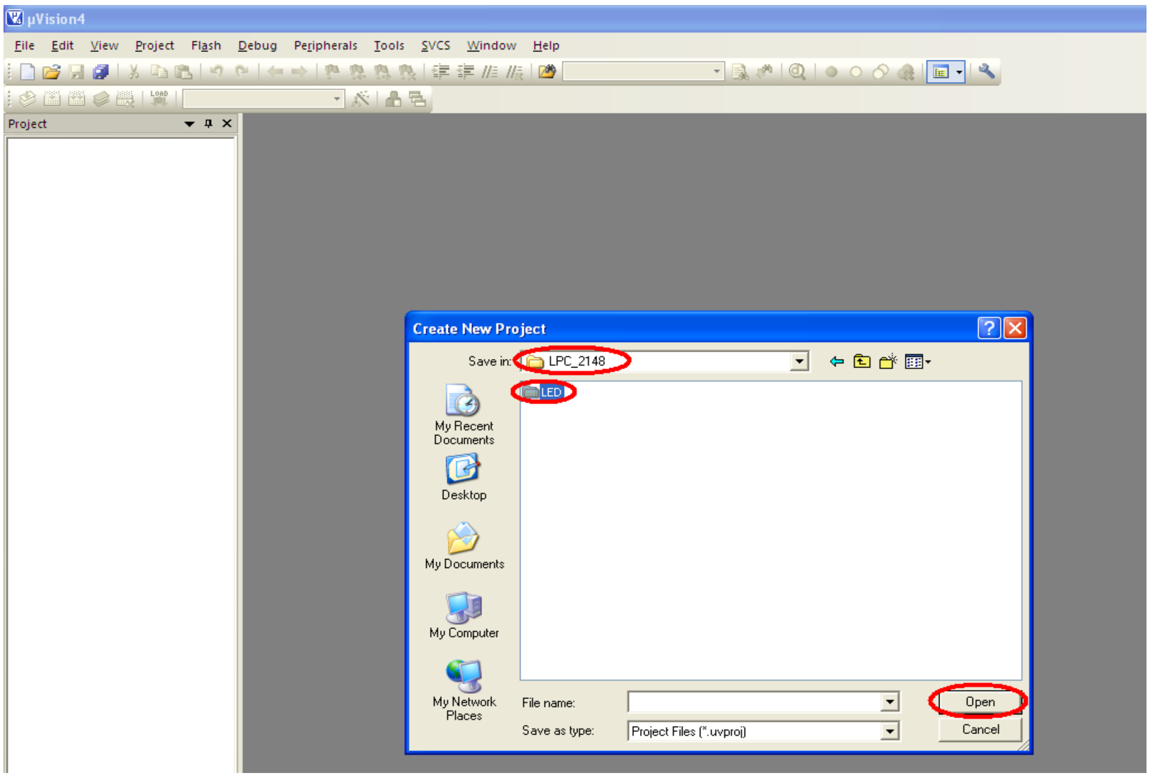

Measure - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target 'LPC2129 Simulator'...
Remove File 'Serial.c'
Options for File 'Serial.c'... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate C:\Keil\ARM\Examples\Measure\Serial.c Ctrl+F7
Stop build
1 C:\Keil\ARM\Examples\Measure\Measure.UV2
2 C:\Keil\ARM\RL\RTX\Examples\Traffic\Traffic.UV2
3 C:\Keil\ARM\Examples\Hello\Hello.UV2
4 C:\Keil\ARM\Boards\Keil\MCB2130\Blinky\Blinky.UV2
Project
LPC2129 Simu
Startup C
Startu
System Ca
Retarg
Serial.
Source Co
Measu
Mcomr
Getline
Document
Abstra

```
(also used by printf function to output data)     */
            /* Write character to Serial Port     */

            /* output CR */

          /* Read character from Serial Port    */
          /* wait until character ready         */
```

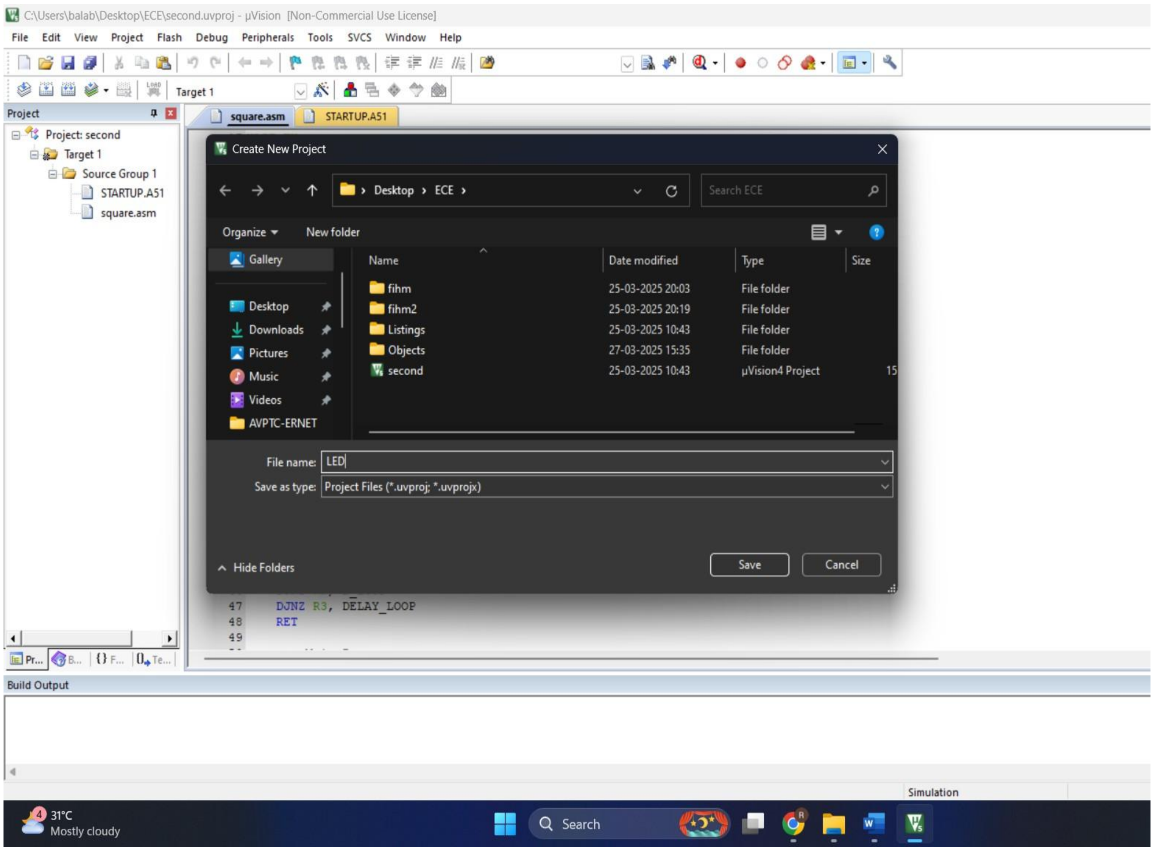
STEP 2: Go to the "**Project**" and click on "**New µvision Project**"


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 C:\Keil\ARM\Examples\Measure\Measure.uvproj
2 C:\Keil\ARM\RL\RTX\Examples\Traffic\Traffic.UV2
3 C:\Keil\ARM\Examples\Hello\Hello.UV2
4 C:\Keil\ARM\Boards\Keil\MCB2130\Blinky\Blinky.UV2
Project

STEP 3: A small window will pop up with the name "**Create New Project**" and can be created and select destination path.

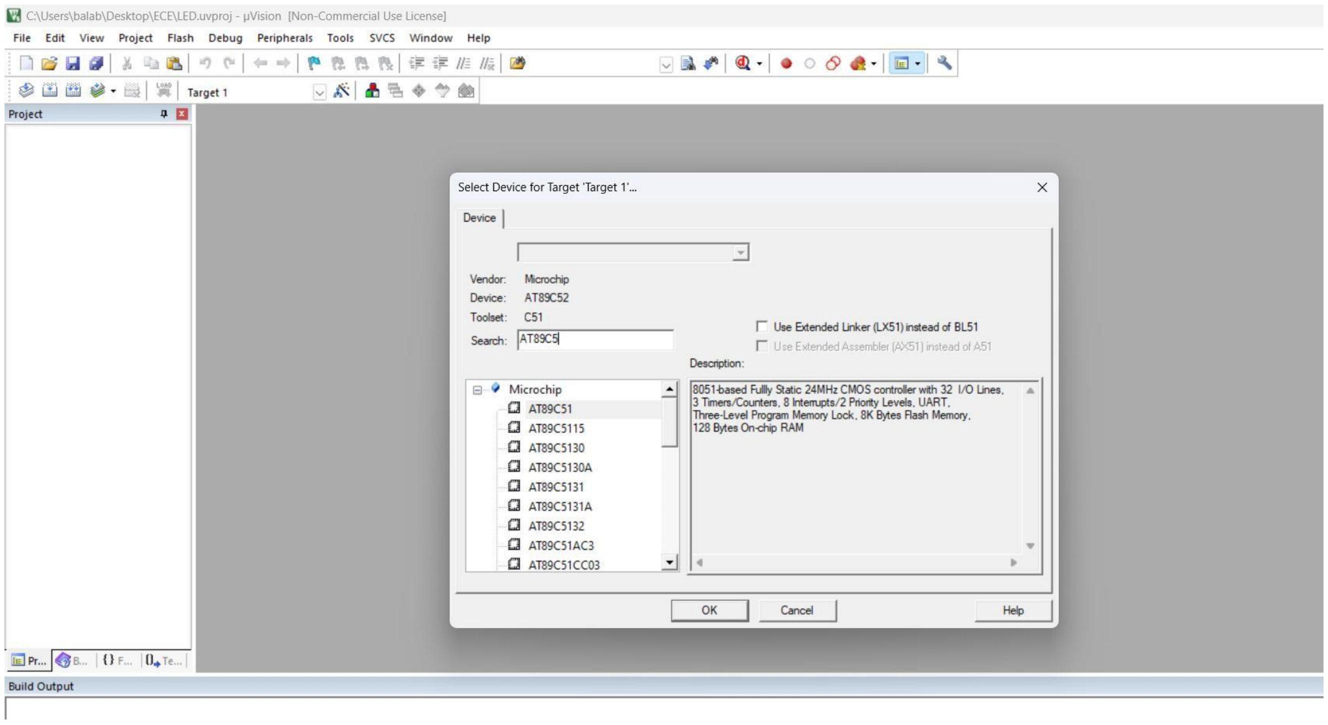


STEP 4: Create a folder and give a proper name that can be related to the Project.


C:\Users\balab\Desktop\ECE\second.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: second
Target 1
Source Group 1
STARTUP.A51
square.asm
square.asm
STARTUP.A51
Create New Project
Desktop › ECE ›
Search ECE
Organize New folder
Gallery
Desktop
Downloads
Pictures
Music
Videos
AVPTC-ERNET
Name Date modified Type Size
fihm 25-03-2025 20:03 File folder
fihm2 25-03-2025 20:19 File folder
Listings 25-03-2025 10:43 File folder
Objects 27-03-2025 15:35 File folder
second 25-03-2025 10:43 µVision4 Project 15
File name: LED
Save as type: Project Files (*.uvproj; *.uvprojx)
Hide Folders
Save
Cancel
Build Output
Simulation
31°C Mostly cloudy
Search

```
DJNZ R3, DELAY_LOOP
RET
```

STEP 5: A small window will pop up with the name "**Select Device for Target 'Target 1'**", and select the data base Microchip.


C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]

Select Device for Target 'Target 1'...

Device

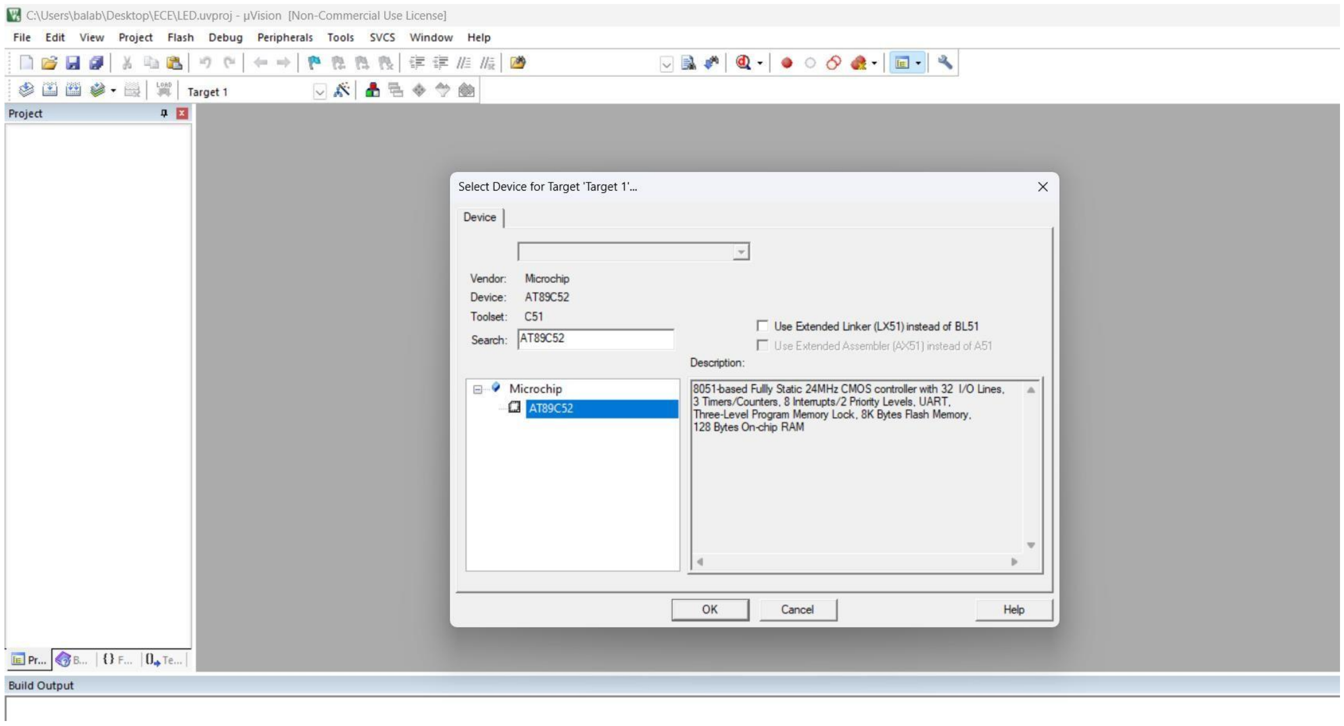
Vendor: Microchip
Device: AT89C52
Toolset: C51
Search: AT89C5

Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51

Description:
8051-based Fully Static 24MHz CMOS controller with 32 I/O Lines, 3 Timers/Counters, 8 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 8K Bytes Flash Memory, 128 Bytes On-chip RAM

Microchip
AT89C51
AT89C5115
AT89C5130
AT89C5130A
AT89C5131
AT89C5131A
AT89C5132
AT89C51AC3
AT89C51CC03

STEP 6: Within the Microchip, select **AT895C2.**


C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]

Select Device for Target 'Target 1'...

Device

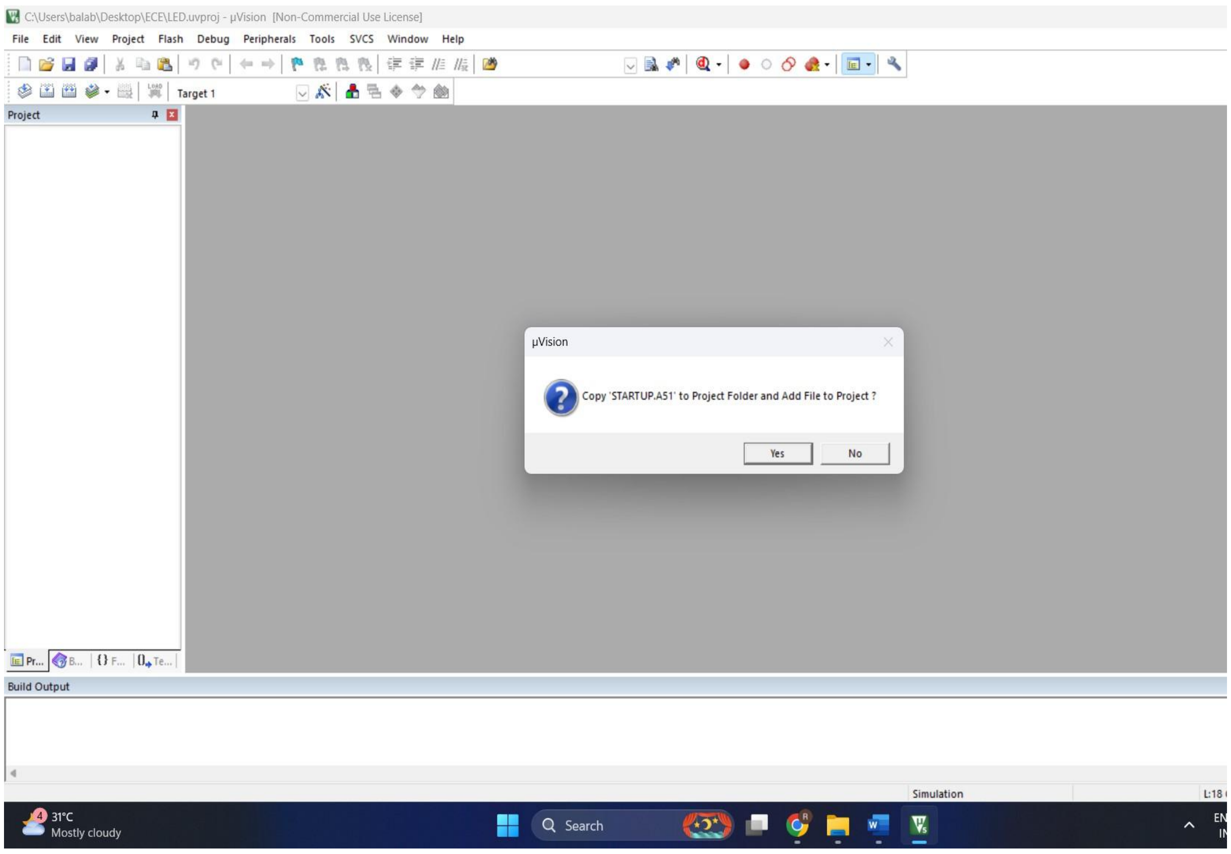
Vendor: Microchip
Device: AT89C52
Toolset: C51
Search: AT89C52

Use Extended Linker (LX51) instead of BL51
Use Extended Assembler (AX51) instead of A51

Description:
8051-based Fully Static 24MHz CMOS controller with 32 I/O Lines, 3 Timers/Counters, 8 Interrupts/2 Priority Levels, UART, Three-Level Program Memory Lock, 8K Bytes Flash Memory, 128 Bytes On-chip RAM

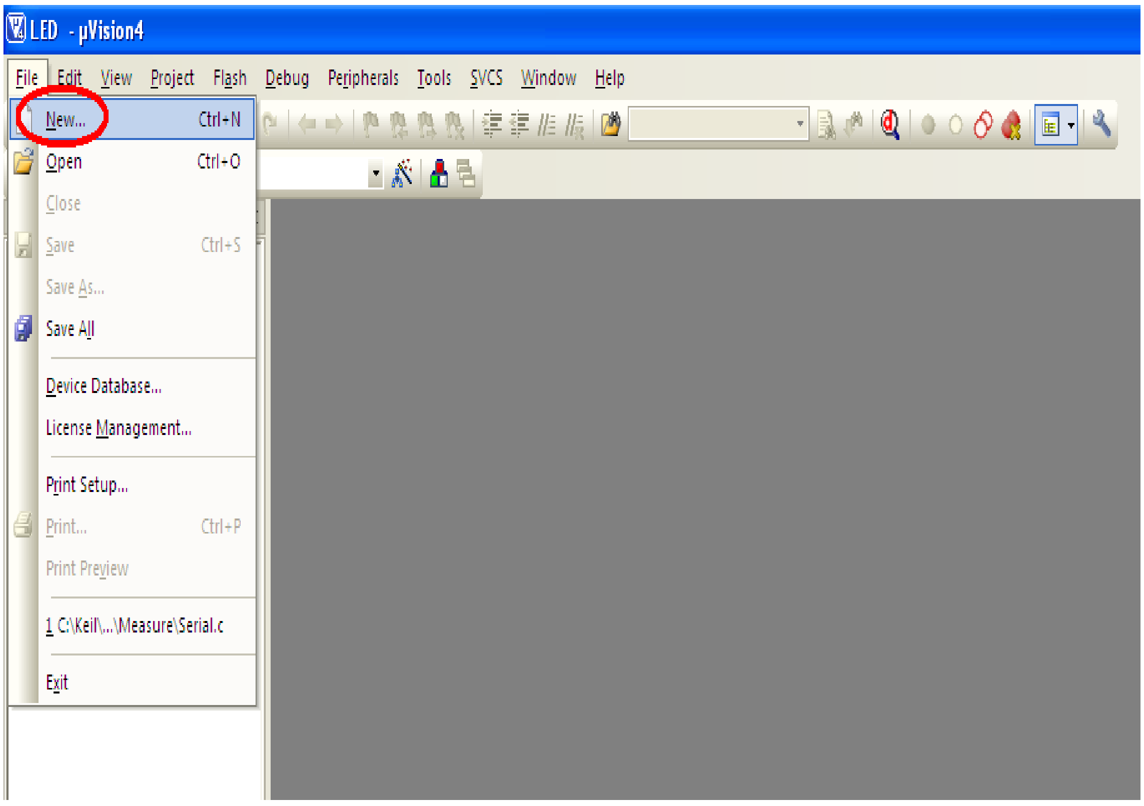
Microchip
AT89C52

STEP 7: Add Startup file to the project by clicking "**Yes**".

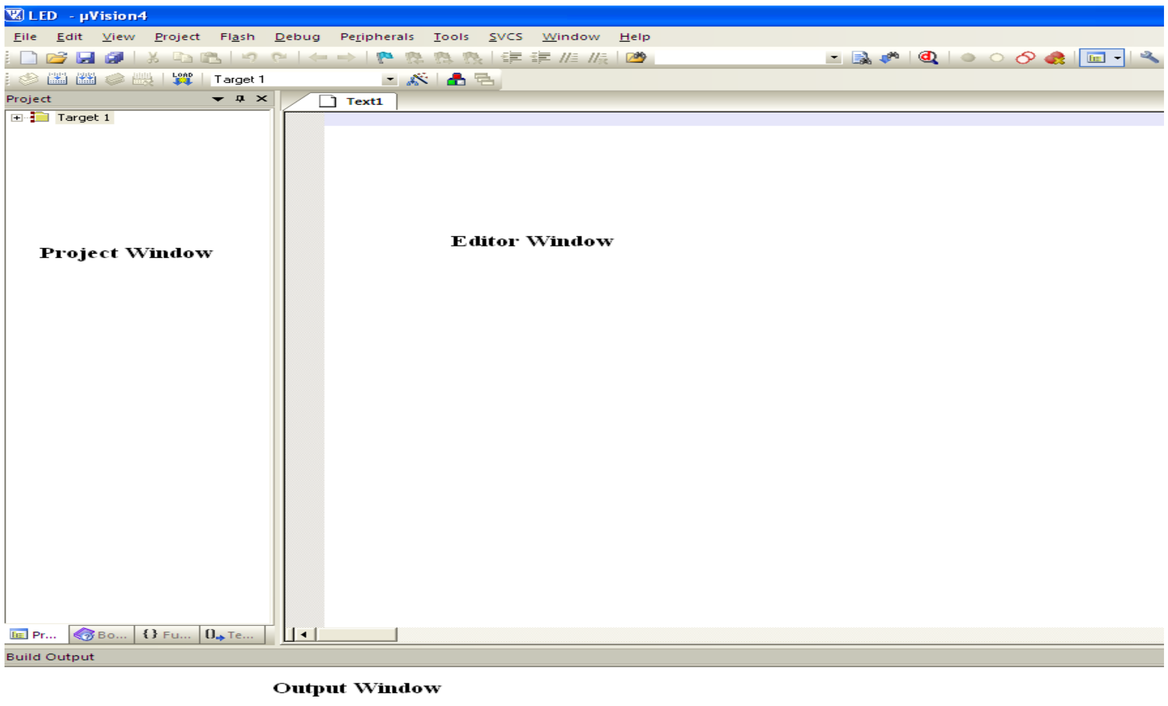

C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Build Output
µVision
Yes
No
Simulation

Copy 'STARTUP.A51' to Project Folder and Add File to Project ?

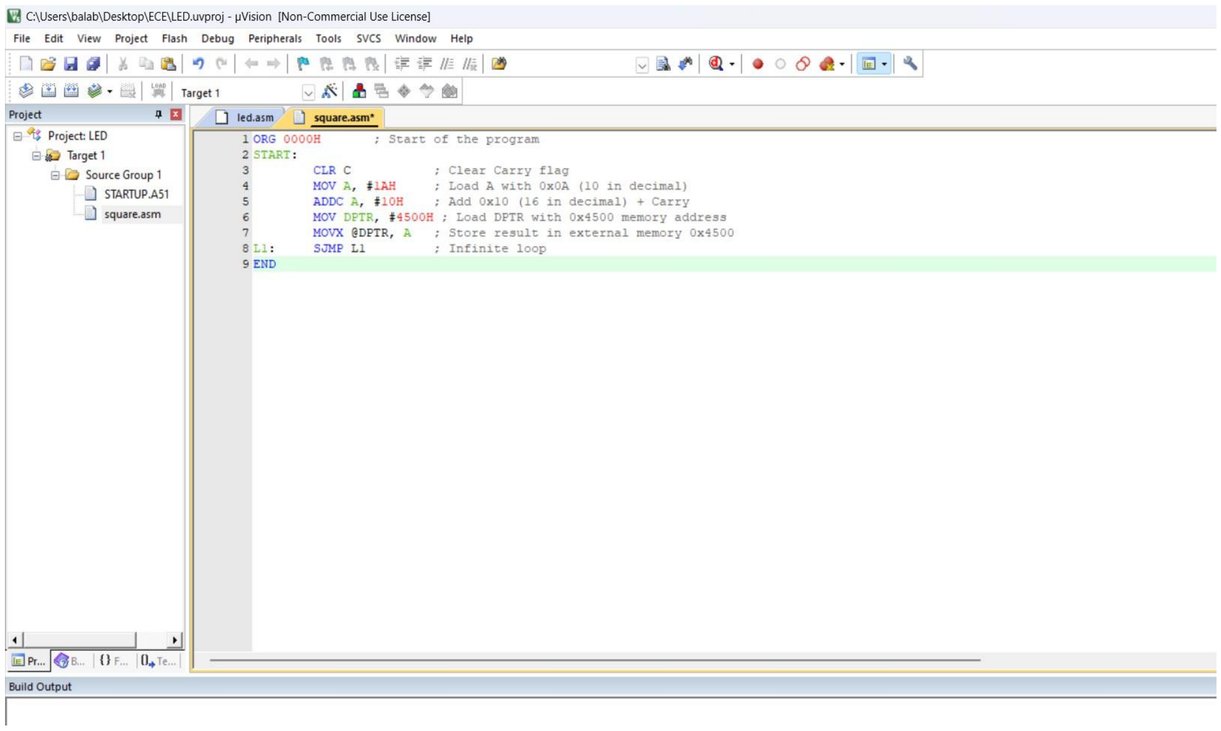
STEP 8: Next go to "**File**" and click "**New**".



STEP 9: There are the three main windows are available in the keil IDE. First one is Project Workspace, the second one is Editor Window and third one is Output Window.

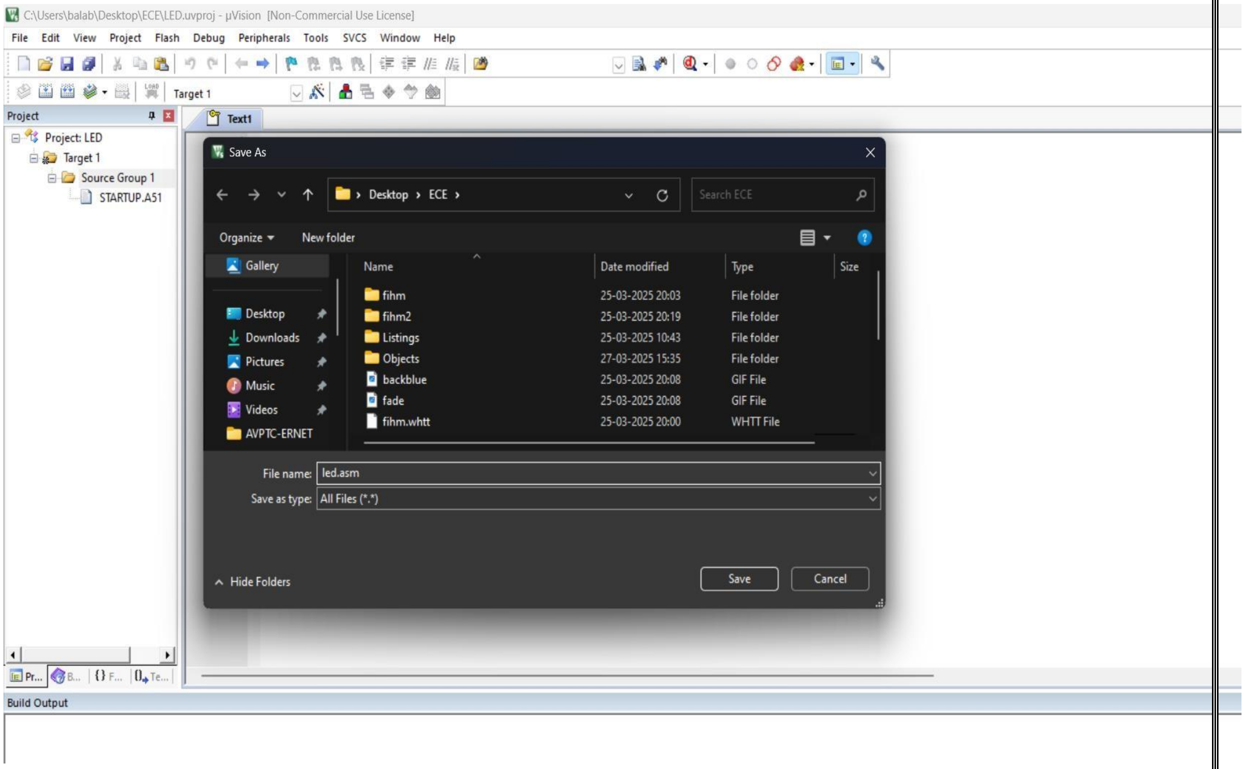


STEP 10: Can start to write *.asm code on the editor window.


C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: LED
Target 1
Source Group 1
STARTUP.A51
square.asm
led.asm
square.asm*
Pr... B... F... Te...
Build Output

```
ORG 0000H        ; Start of the program
START:
        CLR C            ; Clear Carry flag
        MOV A, #1AH      ; Load A with 0x0A (10 in decimal)
        ADDC A, #10H     ; Add 0x10 (16 in decimal) + Carry
        MOV DPTR, #4500H ; Load DPTR with 0x4500 memory address
        MOVX @DPTR, A    ; Store result in external memory 0x4500
L1:     SJMP L1          ; Infinite loop
END
```

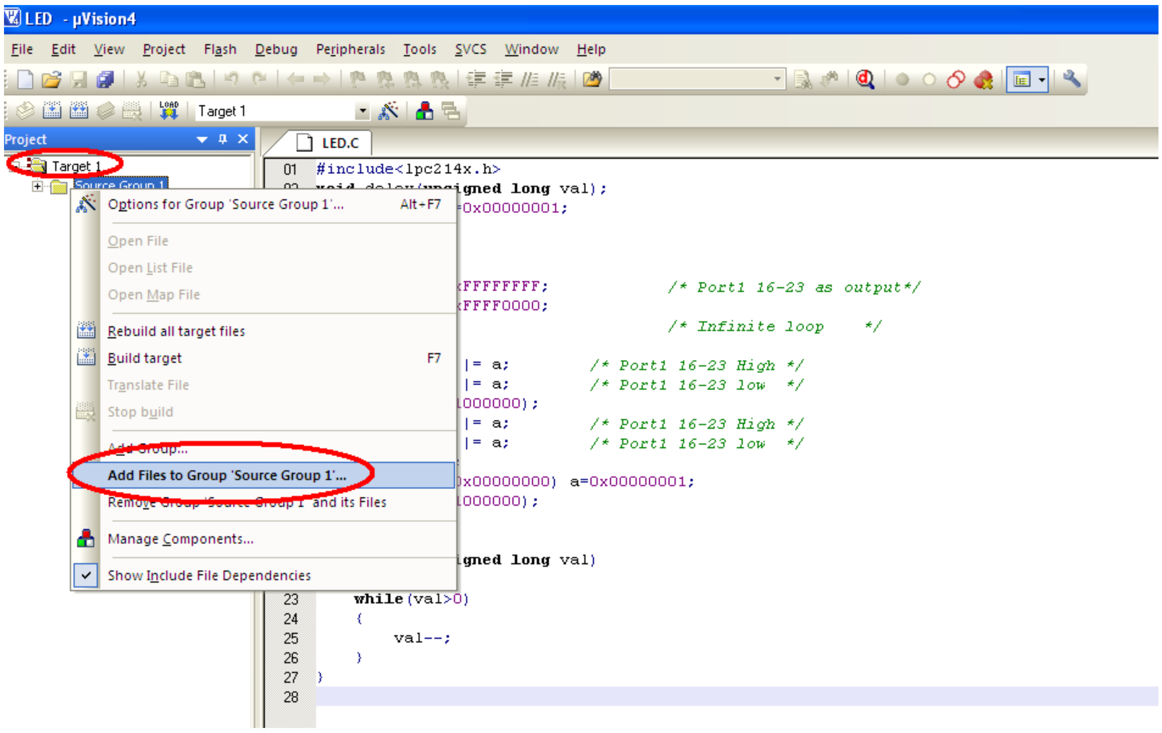
STEP 11: Can save the file, if the program is in "**C**" save as save as "**filename.ASM**".



Name	Date modified	Type	Size
fihm	25-03-2025 20:03	File folder	
fihm2	25-03-2025 20:19	File folder	
Listings	25-03-2025 10:43	File folder	
Objects	27-03-2025 15:35	File folder	
backblue	25-03-2025 20:08	GIF File	
fade	25-03-2025 20:08	GIF File	
fihm.whtt	25-03-2025 20:00	WHTT File	

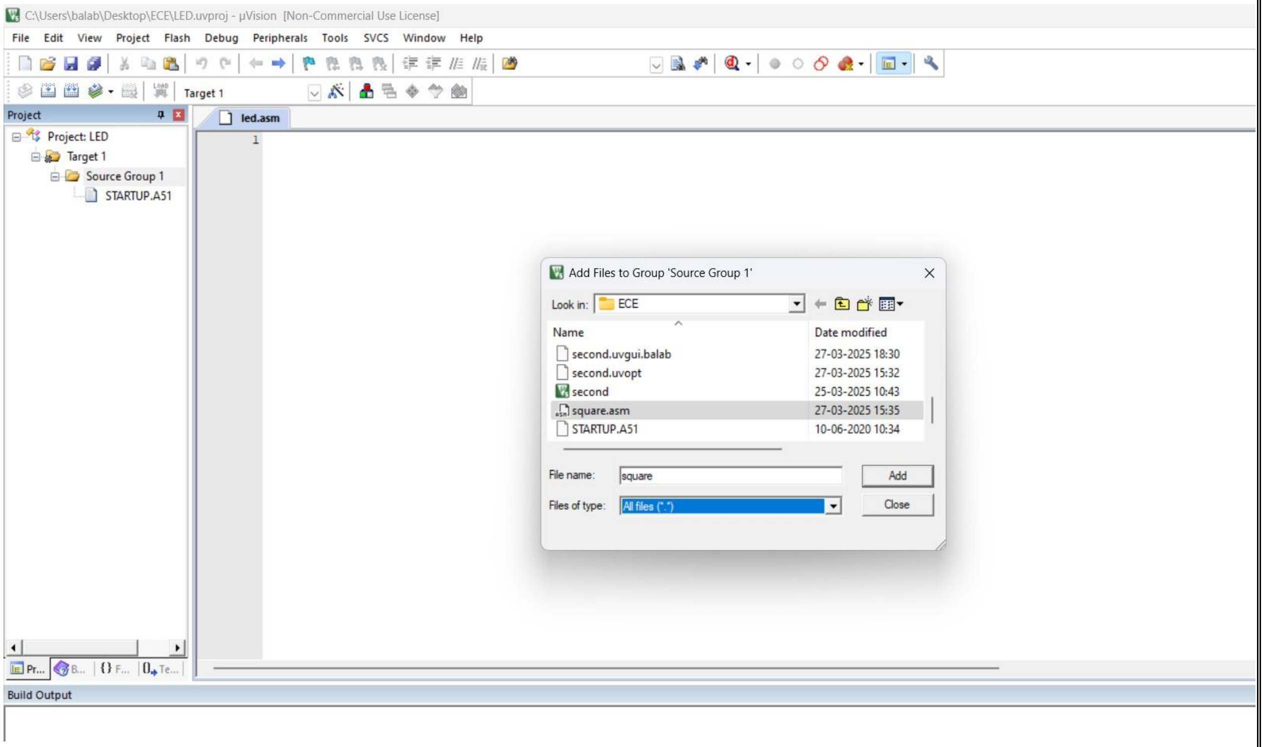
File name: led.asm
Save as type: All Files (*.*)

STEP 12: Add this source file to Group1, Go to the "**Project Workspace**" drag the +mark "**Target 1**" in that right click on "**Source Group1**" and click on "**Add Files to Group "Source Group1"**".


LED - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
LED.C
Options for Group 'Source Group 1'... Alt+F7
Open File
Open List File
Open Map File
Rebuild all target files
Build target F7
Translate File
Stop build
Add Group...
Add Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Manage Components...
Show Include File Dependencies

```
#include<lpc214x.h>
void delay(unsigned long val);
=0x00000001;

FFFFFFFF;             /* Port1 16-23 as output*/
FFFF0000;
                      /* Infinite loop    */

 |= a;         /* Port1 16-23 High */
 |= a;         /* Port1 16-23 low  */
000000);
 |= a;         /* Port1 16-23 High */
 |= a;         /* Port1 16-23 low  */

x00000000)  a=0x00000001;
000000);

gned long val)

    while(val>0)
    {
        val--;
    }
}
```

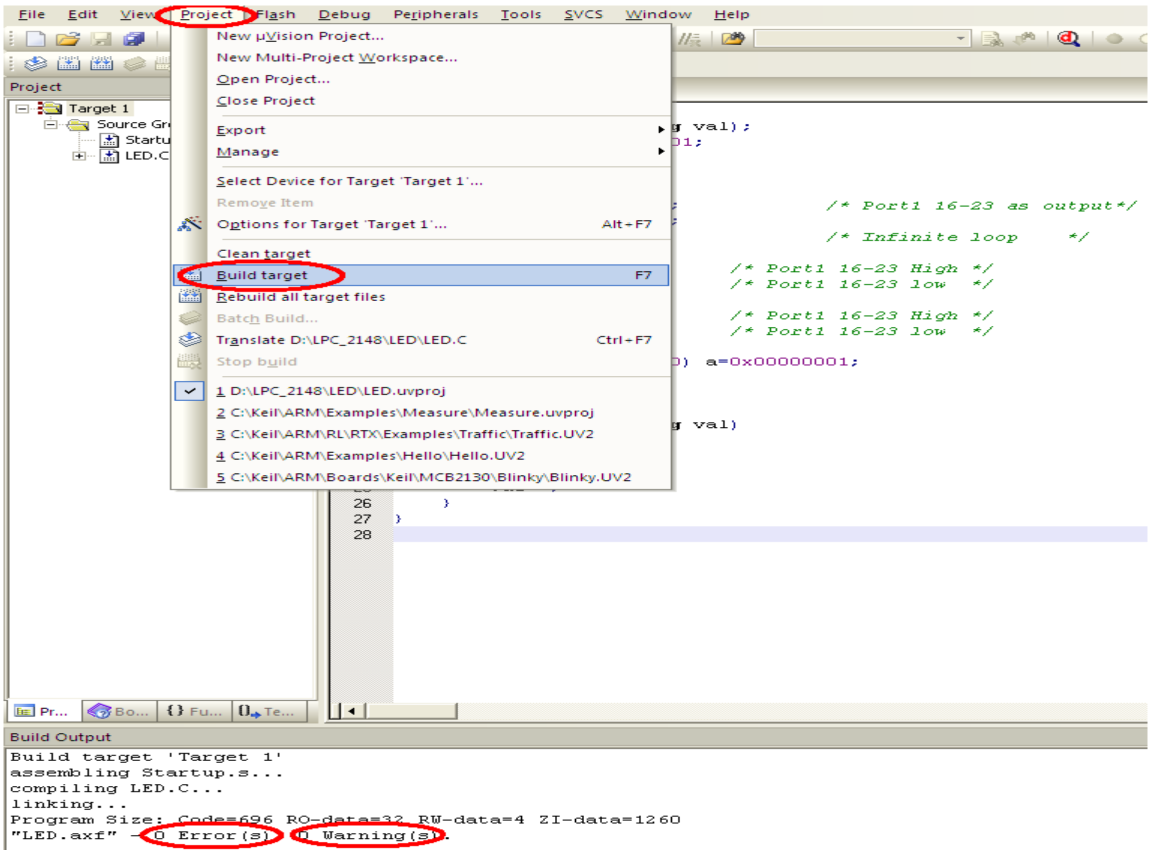
STEP 13: A small window will pop up with the name "**Add Files to Group Source Group1",** by default, the Files of type will be in **All Files (*.asm).** If the program is asm select **ASM Source file (*.s,*.src,*.a*).**


C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: LED
Target 1
Source Group 1
STARTUP.A51
led.asm
Add Files to Group 'Source Group 1'
Look in: ECE
Name
Date modified
Add
Close
Build Output

```
second.uvgui.balab   27-03-2025 18:30
second.uvopt         27-03-2025 15:32
second               25-03-2025 10:43
square.asm           27-03-2025 15:35
STARTUP.A51          10-06-2020 10:34
```

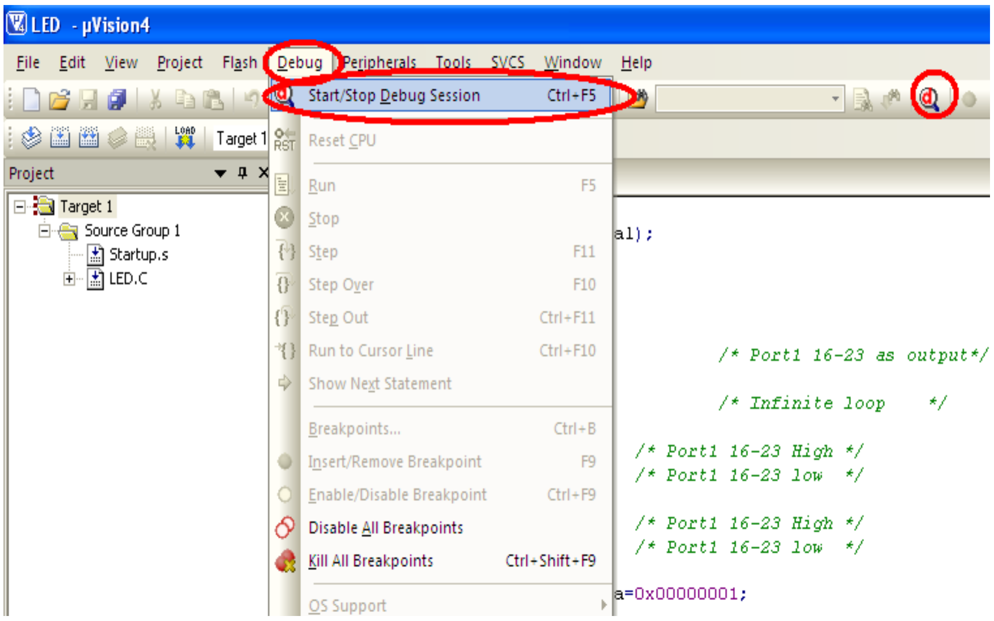
File name: square
Files of type: All files (*.*)

STEP 14: Then go to "**Project**" click on "**Build Target**" or **F7**. Output window will display related error and warning messages.


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Select Device for Target 'Target 1'...
Remove Item
Options for Target 'Target 1'... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate D:\LPC_2148\LED\LED.C Ctrl+F7
Stop build
1 D:\LPC_2148\LED\LED.uvproj
2 C:\Keil\ARM\Examples\Measure\Measure.uvproj
3 C:\Keil\ARM\RL\RTX\Examples\Traffic\Traffic.UV2
4 C:\Keil\ARM\Examples\Hello\Hello.UV2
5 C:\Keil\ARM\Boards\Keil\MCB2130\Blinky\Blinky.UV2
Project
Target 1
Source Gr
Startu
LED.C
Build Output

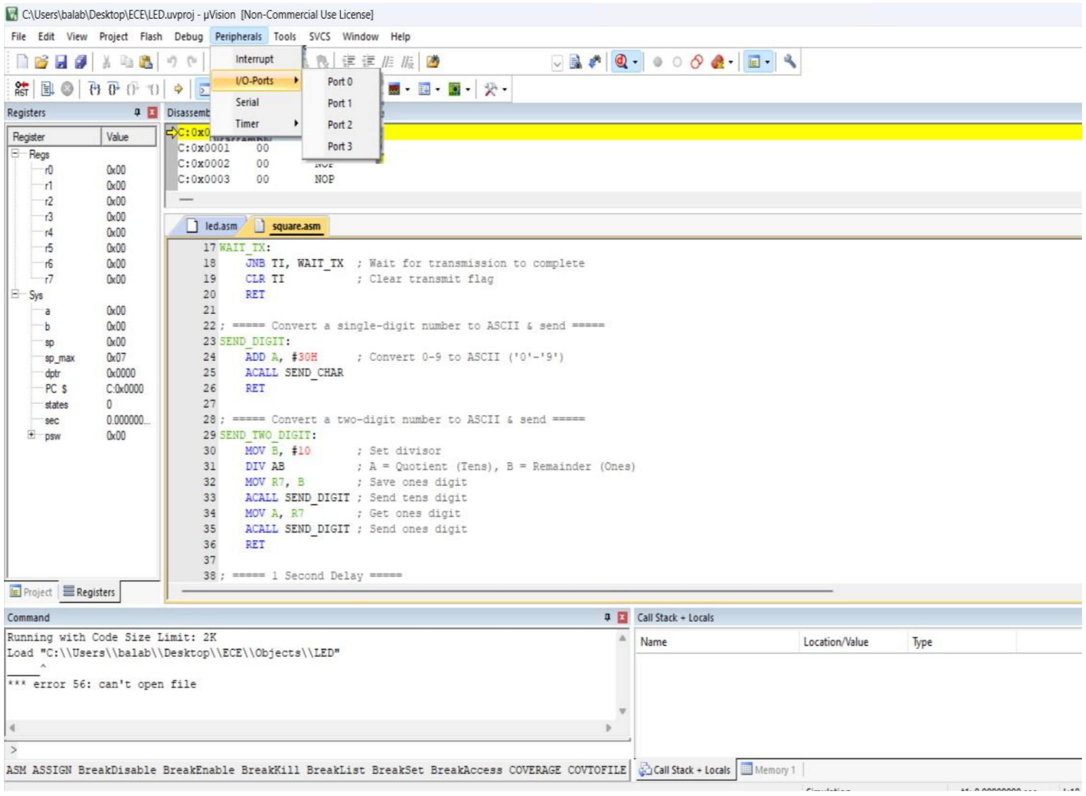
```
Build target 'Target 1'
assembling Startup.s...
compiling LED.C...
linking...
Program Size: Code=696 RO-data=32 RW-data=4 ZI-data=1260
"LED.axf" - 0 Error(s), 0 Warning(s).
```

Simulation
Part:
STEP 15: Go to "**Project**" menu, click on "**Rebuild all target Files**" and start **Debug**. From **View** menu can get **Memory Window** and from **Peripherals** can get I/O ports, Serial etc. For access internal memory type **i:0x_memory** location example: **i:0x30** and for external and program memory **x:0x_memory location**, **c:0x_memory location** respectively. From **Register** window we can edit and access the values also.


LED - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Target 1
Project
Target 1
Source Group 1
Startup.s
LED.C

```
al);

                /* Port1 16-23 as output*/

                /* Infinite loop    */

 /* Port1 16-23 High */
 /* Port1 16-23 low  */

 /* Port1 16-23 High */
 /* Port1 16-23 low  */
a=0x00000001;
```

STEP 16: If Output has to be seen on Port select Peripherals → IO Ports → Select Port



Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x00
sp_max	0x07
dptr	0x0000
PC $	C:0x0000
states	0
sec	0.000000...
psw	0x00

```
C:0x0001    00
C:0x0002    00
C:0x0003    00      NOP
```

```
WAIT_TX:
    JNB TI, WAIT_TX  ; Wait for transmission to complete
    CLR TI           ; Clear transmit flag
    RET

; ===== Convert a single-digit number to ASCII & send =====
SEND_DIGIT:
    ADD A, #30H      ; Convert 0-9 to ASCII ('0'-'9')
    ACALL SEND_CHAR
    RET

; ===== Convert a two-digit number to ASCII & send =====
SEND_TWO_DIGIT:
    MOV B, #10       ; Set divisor
    DIV AB           ; A = Quotient (Tens), B = Remainder (Ones)
    MOV R7, B        ; Save ones digit
    ACALL SEND_DIGIT ; Send tens digit
    MOV A, R7        ; Get ones digit
    ACALL SEND_DIGIT ; Send ones digit
    RET

; ===== 1 Second Delay =====
```

```
Running with Code Size Limit: 2K
Load "C:\\Users\\balab\\Desktop\\ECE\\Objects\\LED"
      ^
*** error 56: can't open file
```

```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
```

EXPERIMENTS USING 8051

EXP NO: 1	PROGRAMMING ARITHMETIC AND LOGICAL OPERATIONS IN 8051
DATE	

AIM:

To write 8051 Assembly Language Program to demonstrate arithmetic and logic operations using Keil simulator and execute it.

SOFTWARE REQUIRED:

S.No	Software Requirements	Quantity
1	Keil μvision5 IDE	1

INTRODUCTION TO 8051 SIMULATORS:

A simulator is software that will execute the program and show the results exactly to the program running on the hardware, if the programmer finds any errors in the program while simulating the program in the simulator, he can change the program and re-simulate the code and get the expected result, before going to the hardware testing. The programmer can confidently dump the program in the hardware when he simulates his program in the simulator and gets the expected results.

8051 controller is a most popular 8-bit controller which is used in a large number of embedded applications and many programmers write programs according to their application. So testing their programs in the software simulators is a way. Simulators will help the programmer to understand the errors easily and the time taken for the testing is also decreased.

These simulators are very useful for students because they do need not to build the complete hardware for testing their program and validate their program very easily in an interactive way.

List of 8051 Simulators:

The list of simulators is given below with their features:

1. MCU 8051: MCU 8051 is an 8051 simulator that is very simple to use and has an interactive IDE (Integrated Development Environment). It is developed by Martin Osmera and most important of all is that it is completely free. There are many features for this IDE they are

- ✓ It supports both C and assembly language for compilation and simulation

- ✓ It has an in-built source code editor, graphical notepad, ASCII charts, Assembly symbol viewer, etc. It also supports several 8051 ICs like at89c51, A89S52, 8051, 8052, etc.
- ✓ It will support certain electronic simulations like LED, 7segment display, LCD etc. which will help in giving the output when you interface these things to the hardware directly.
- ✓ It has tools like hex decimal editors, base converters, special calculator, file converters, inbuilt hardware programmers, etc.
- ✓ It has syntax validation, pop base auto-completion etc.

You can download this tool from https://sourceforge.net/projects/mcu8051ide/files/.

2. EDSIM 51: This is a virtual 8051 interfaced with virtual peripherals like 7 segment display, motor, keypad, UART etc. This simulator is exclusively for students developed by James Rogers,.

The features of this simulator are

- ✓ Have virtual peripherals like ADC, DAC with scope to display, comparator etc.
- ✓ Supports only assembly language
- ✓ IDE is completely written in JAVA and supports all the OS.
- ✓ Completely free and with user guide, examples, etc.

You can download this simulator from the https://www.edsim51.com/index.html.

3. 8051 IDE: This simulation software is exclusively for the Windows operating system (98/xp).

The features of this simulator are

- ✓ Text editor, assembler, and software simulate in one single program.
- ✓ Has facilities like Breakpoint setter, execute to break point, predefined simulator watch window, etc.
- ✓ It is available in both free version and paid version.

You can download this tool from https://www.acebus.com/win8051.htm

4. KEIL µVision: KEIL is the most popular software simulator. It has many features like interactive IDE and supports both C and assembly languages for compilation and simulation.

You can download and get more information from https://www.keil.com/c51/.

INSTALLATION OF KEIL SOFTWARE

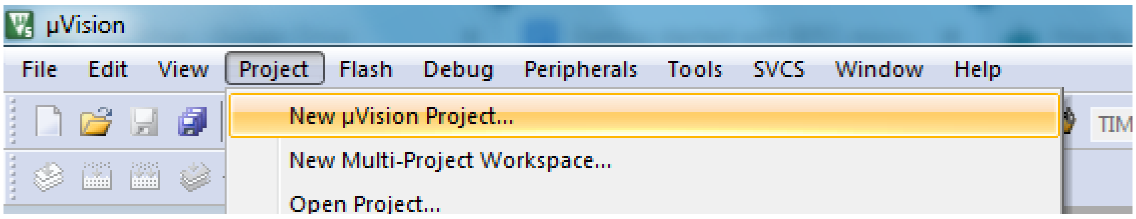
Set up Keil IDE for Programming

Keil µVision IDE is a popular way to program MCUs containing the 8051 architectures. It supports over 50 microcontrollers and has good debugging tools including logic analyzers and watch windows. In this article, we will use the AT89C51ED2 microcontroller, which has:

- 64 KB FLASH ROM
- On-chip EEPROM
- 256 Bytes RAM
- In-System programming for uploading the program
- 3 Timer/counters
- SPI, UART, PWM

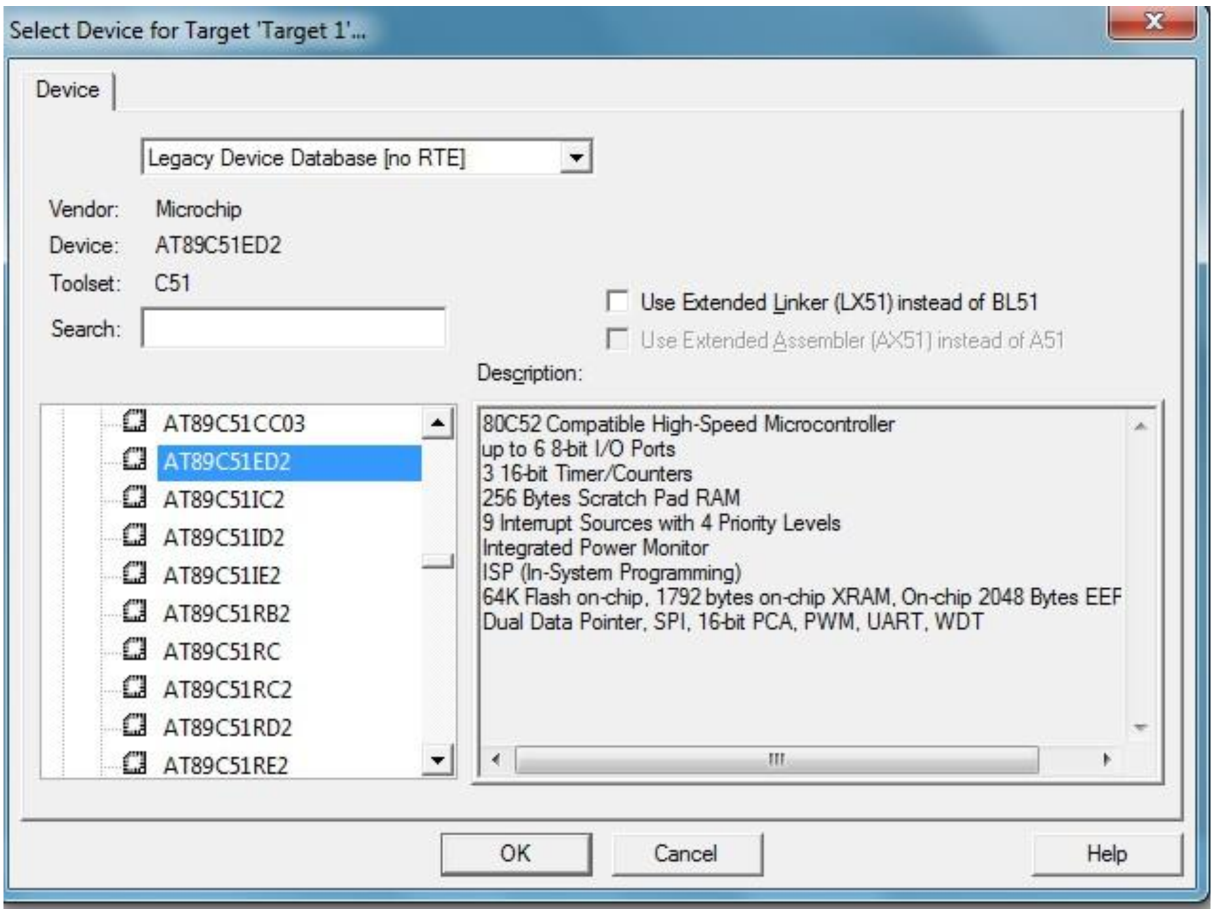
The Keil µVision icon.

To start writing a new program, you need to create a new project. Navigate to **project —> New µVision project**. Then save the new project in a folder.


µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...

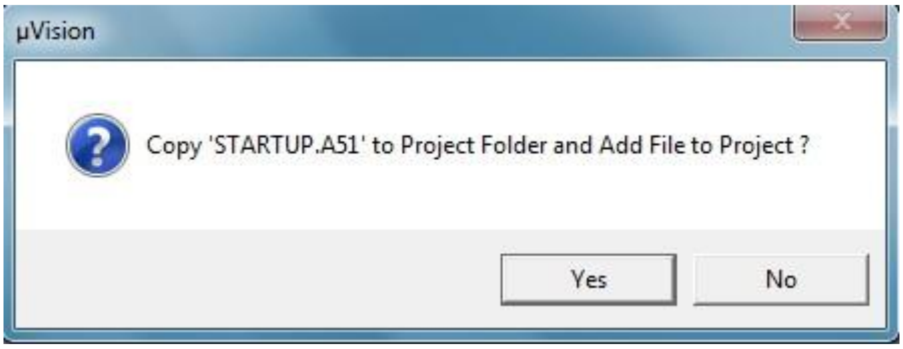
After saving the file, a new window will pop up asking you to select your microcontroller.

As discussed, we are using AT89C51/AT89C51ED2/AT89C52, so select this controller under the Microchip section (as Atmel is now a part of Microchip).



80C52 Compatible High-Speed Microcontroller
up to 6 8-bit I/O Ports
3 16-bit Timer/Counters
256 Bytes Scratch Pad RAM
9 Interrupt Sources with 4 Priority Levels
Integrated Power Monitor
ISP (In-System Programming)
64K Flash on-chip, 1792 bytes on-chip XRAM, On-chip 2048 Bytes EEF
Dual Data Pointer, SPI, 16-bit PCA, PWM, UART, WDT

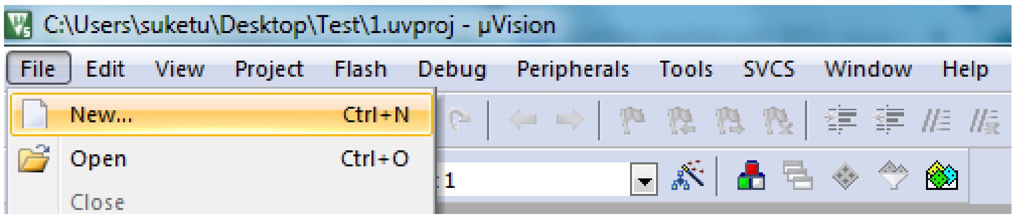
Select '**Yes**' in the next pop-up, as we do not need this file in our project.



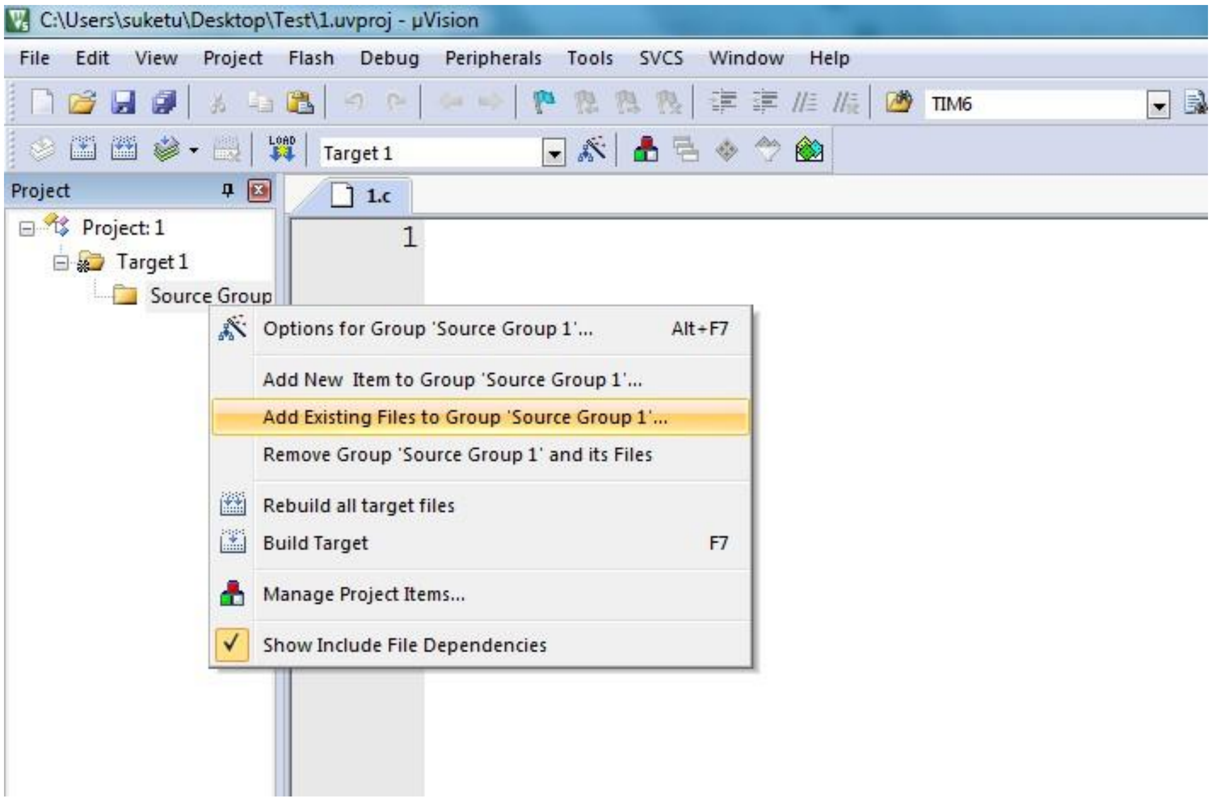
Copy 'STARTUP.A51' to Project Folder and Add File to Project ?

Our project workspace is now ready!

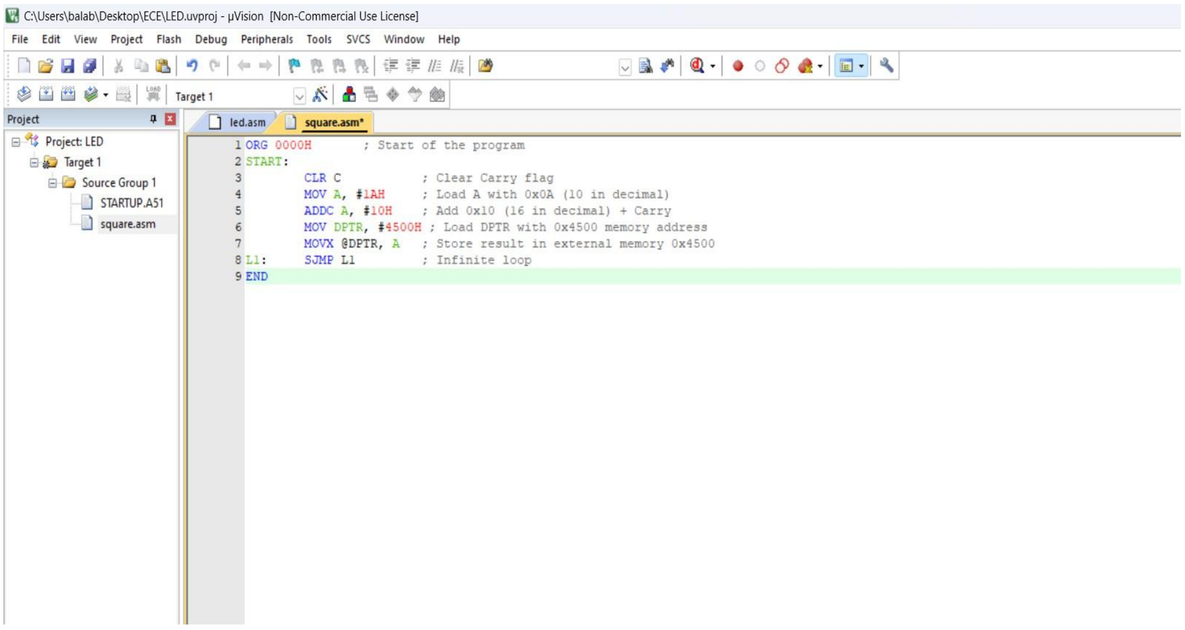
From here, we need to create a file where we can write our C code. Navigate to **File —> New**. Once the file is created, save it with .c extension in the same project folder.



Next, we have to add that .c or .asm file to our project workspace. Select **Add Existing** Files and then select the created .c or .asm file to get it added.


C:\Users\suketu\Desktop\Test\1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
TIM6
Target 1
Project
Project: 1
Target 1
Source Group
1.c
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies

The workspace and project file are ready.


C:\Users\balab\Desktop\ECE\LED.uvproj - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: LED
Target 1
Source Group 1
STARTUP.A51
square.asm
led.asm
square.asm*

```
ORG 0000H       ; Start of the program
START:
        CLR C          ; Clear Carry flag
        MOV A, #1AH    ; Load A with 0x0A (10 in decimal)
        ADDC A, #10H   ; Add 0x10 (16 in decimal) + Carry
        MOV DPTR, #4500H ; Load DPTR with 0x4500 memory address
        MOVX @DPTR, A  ; Store result in external memory 0x4500
L1:     SJMP L1        ; Infinite loop
END
```

PROCEDURE

1. Create a new project, go to “Project” and close the current project “Close Project”.
2. Next, Go to the Project New µVision Project and Create a New Project, Select the Device for the Target.
3. Select the device AT89C51ED2 or. AT89C51 or AT89C52
4. Add Startup file Next go to “File” and click “New”.
5. Write a program on the editor window and save it with .asm extension.
6. Add this source file to Group and click on “Build Target” or F7.
7. Go to debugging mode to see the result of simulation by clicking Run or step run.8.

PROGRAM:

8-BIT ADDITION

```
ORG 0000H       ; Start of the program
START:
      CLR C           ; Clear Carry flag
      MOV A, #1AH     ; Load A with 0x0A (10 in decimal)
      ADDC A, #10H    ; Add 0x10 (16 in decimal) + Carry
      MOV DPTR, #4500H ; Load DPTR with 0x4500 memory address
      MOVX @DPTR, A   ; Store result in external memory 0x4500
L1:   SJMP L1         ; Infinite loop
END
```

OUTPUT :

PROGRAM

8-BIT SUBTRACTION

```
ORG 0000H       ; Start of the program
START:      CLR C           ; Clear Carry flag (used in SUBB)
      MOV A,  #0AH    ; Load A with 0x0A (decimal 10)
      SUBB A, #05H    ; Subtract 0x05 (decimal 5) from A (A = A - 5 - CY)
      MOV DPTR, #4500H ; Load DPTR with address 0x4500H
      MOVX @DPTR, A   ; Store result of subtraction at external memory (XDATA) 0x4500
L1:   SJMP L1         ; Infinite loop to hold execution
END
```

OUTPUT :

PROGRAM

8-BIT MULTIPLICATION

```
ORG 0000H       ; Start of the program
START:
        MOV A, #05H       ; Load A with 0x05 (decimal 5)
        MOV B, #03H       ; Load B with 0x03 (decimal 3)
        MUL AB            ; Multiply A × B (Result stored in A & B)

        MOV DPTR, #4500H  ; Load DPTR with address 0x4500H
        MOVX @DPTR, A     ; Store the lower byte of the result (A) at 0x4500H

        INC DPTR          ; Increment DPTR to 0x4501H
        MOV A, B          ; Move higher byte of result (B) to A
        MOVX @DPTR, A     ; Store the higher byte at 0x4501H

L1:     SJMP L1           ; Infinite loop to hold execution
END
```

OUTPUT :

PROGRAM

8-BIT DIVISION

```
ORG 0000H       ; Start of the program
START:
        MOV A, #H       ; Load A with 0x15 (decimal 21)
        MOV B, #03H0E       ; Load B with 0x03 (decimal 3)
        DIV AB            ; Divide A by B (A ÷ B)

        MOV DPTR, #4500H  ; Load DPTR with address 0x4500H
        MOVX @DPTR, A     ; Store the quotient in external memory (XDATA 0x4500)

        INC DPTR          ; Increment DPTR to 0x4501H
        MOV A, B          ; Move the remainder to A
        MOVX @DPTR, A     ; Store the remainder in external memory (XDATA 0x4501)

L1:     SJMP L1           ; Infinite loop to hold execution
END
```

OUTPUT :

PROGRAM

LOGICAL OPERATION

```
ORG 0000H       ; Start of program
START:
   MOV A, #0F0H  ; Load A with 1111 0000 (F0H)
   MOV R1, #0AAH ; Load R1 with 1010 1010 (AAH)

   ; Perform AND operation
   ANL A, R1       ; A = A AND R1 (1010 0000)
   MOV DPTR, #4500H
   MOVX @DPTR, A   ; Store AND result in memory

   ; Perform OR operation
   ORL A, R1       ; A = A OR R1 (1010 1010)
   INC DPTR
   MOVX @DPTR, A   ; Store OR result in memory

   ; Perform XOR operation
   XRL A, R1       ; A = A XOR R1 (0000 0000)
   INC DPTR
   MOVX @DPTR, A   ; Store XOR result in memory

   ; Perform NOT (Complement) operation
   MOV A, #55H     ; Load A with 0101 0101 (55H)
   CPL A           ; Complement A (1010 1010)
   INC DPTR
   MOVX @DPTR, A   ; Store NOT result in memory

L1: SJMP L1      ; Infinite loop
END
```

OUTPUT :

RESULT:

Thus 8051 Assembly Language Program to demonstrate arithmetic and logic operations using Keil simulator is written and executed.

EXP NO: 2	IMPLEMENTATION OF IOT USING PYTHON PROGRAMMING
DATE	

AIM:

To write and execute program to implement IoT using Python programming.

SOFTWARE /HARDWARE REQUIRED:

S.No	Requirements	Quantity
1	Arduino IDE	1
2	Arduino Uno	1
3	LEDs	4
4	Resistros	4
5	Bread board	1

Importance of the Internet of Things

The Internet of Things (IoT) is transforming how we interact with technology, integrating it into every facet of our lives. Here are some key reasons why IoT is so important:

- Efficiency and Productivity: IoT enables automation in homes, factories, and offices, which can increase productivity and operational efficiencies.
- Real-Time Data: IoT devices provide real-time data that can be used for monitoring, analysis, and decision-making.
- Remote Control and Automation: IoT allows for the remote operation and automation of daily tasks.
- Improved Decision Making: With the vast amounts of data that IoT devices generate, businesses and individuals can make more informed decisions.
- Cost Reduction: IoT can help reduce costs by optimizing energy use and improving resource management.

IoT and Python's Role

The Internet of Things refers to a vast network of connected physical objects embedded with sensors, software, and other technologies. These objects collect and exchange data with the internet, integrating the physical and digital worlds. This convergence enhances automation, efficiency, and decision-making across various sectors including healthcare, agriculture, and manufacturing.

Python for IoT:

Python is favored for IoT due to its simplicity, readability, and a rich ecosystem of libraries. Python is a popular choice for developing Internet of Things (IoT) applications due to several compelling advantages:

- Simplicity and Readability: Python's syntax is clear and concise, making it easy to learn and use, especially for beginners.
- Extensive Libraries and Frameworks: Python has a vast ecosystem of libraries and frameworks that can significantly accelerate IoT development. Libraries like NumPy and Pandas facilitate data analysis.
- Platform Independence: Python is a cross-platform language, meaning it can run on various operating systems including Windows, macOS, Linux, and even on smaller devices like Raspberry Pi, which is popular for IoT projects.
- Community Support: Python has one of the largest programming communities, providing extensive support through forums, tutorials, and third-party software.
- Integration Capabilities: Python integrates well with other languages and technologies, which is vital in the IoT landscape where different types of technologies and protocols must work together seamlessly.
- Prototyping Speed: Python enables rapid prototyping—crucial in IoT development where adjusting to new requirements and testing ideas quickly can significantly cut down development time and cost.
- Data Handling and Analytics: IoT devices generate massive amounts of data, and Python's capabilities in handling and analyzing data are well-established. With libraries like Matplotlib for data visualization and Scikit-Learn for machine learning.

Python Platforms for IoT Development

Python on Raspberry Pi

Python on PyBoard

SP8266 and ESP32 with MicroPython

PROCEDURE

Control Arduino with Python and pyFirmata

Installing PyFirmata Module

You should have Python and pip Installed in your system. Then you can run the following command to install the PyFirmata module in your system.

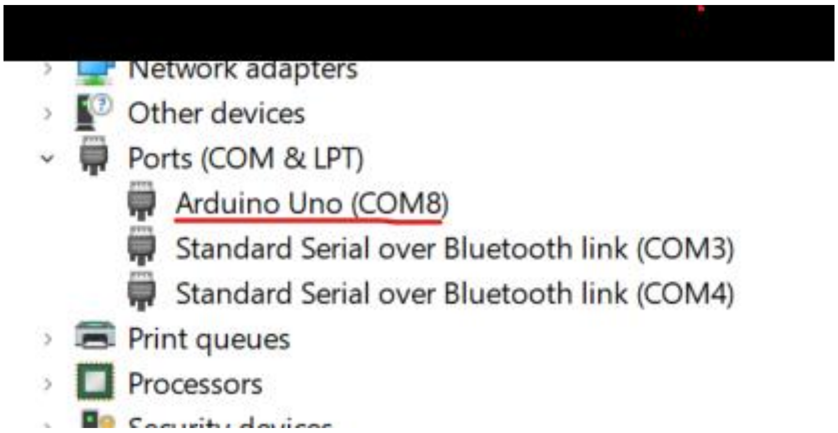
```
pip install pyFirmata
```

Upload “StandardFirmata” to Arduino

StandardFirmata is a code that helps Python get access to the Arduino board.

First, connect your Arduino to the computer/raspberry pi/laptop using the USB cable.

Know the port name the Arduino is connected to. In windows, the port name will be something like “COMx” (where x is an integer), while in Linux it will be a string starting with “/dev/tty”. You might find this information by opening Device Manager in Windows.



[redacted]

- Network adapters
- Other devices
- Ports (COM & LPT)
  - Arduino Uno (COM8)
  - Standard Serial over Bluetooth link (COM3)
  - Standard Serial over Bluetooth link (COM4)
- Print queues
- Processors

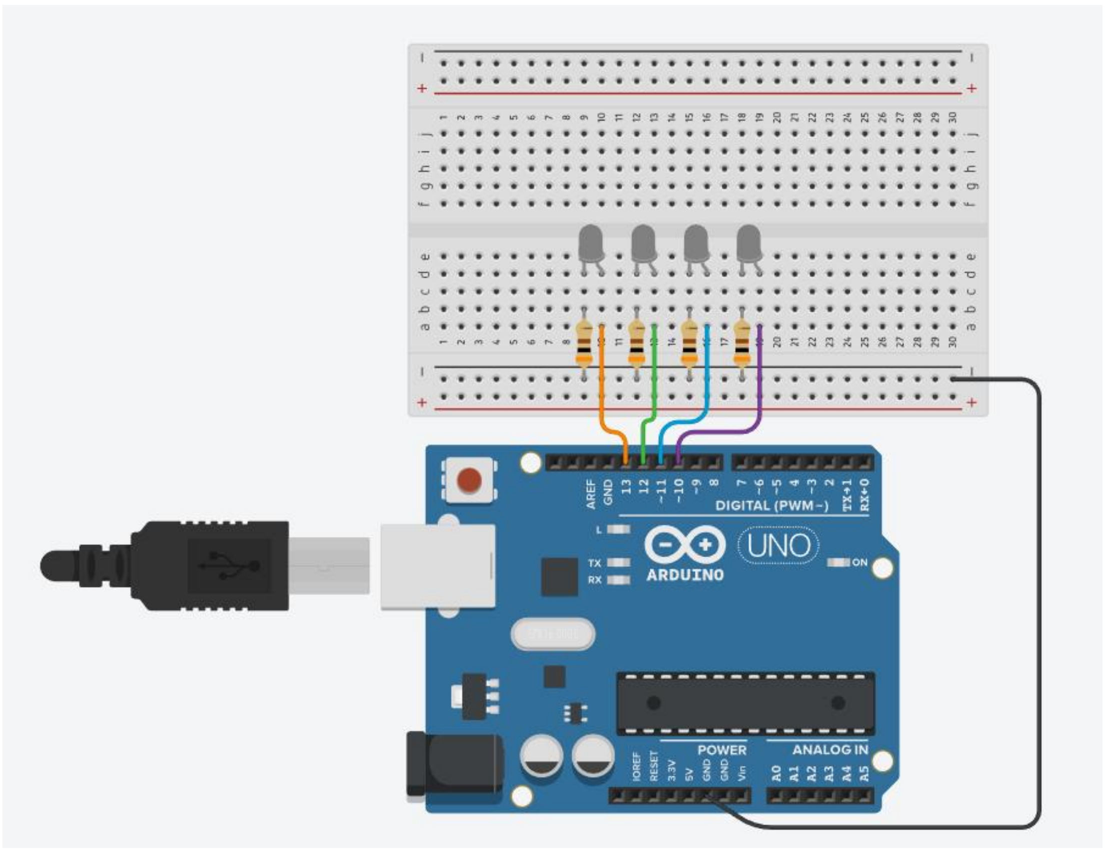
Next, you can open the Arduino IDE and follow the steps to upload the StandardFirmata to the board.

Get StandardFirmata: File -> Examples -> Firmata -> Standard Firmata

Specify Correct Board and Port: Tools -> Board -> Select Arduino UNO (or your own board) -> Tools -> Port -> Select your Port

Upload the StandardFirmata: Click on the upload button to upload the code to Arduino.

Making the connections



Make the connections like the image above. Here I have connected the 4 LEDs to the 13th, 12th, 11th, and 10th pins. There was no specific reason to connect them in that manner. You can use any other digital pin.

Python program:

```
from pyfirmata import Arduino
from time import sleep

# Connecting to the board
board = Arduino('COM8')

# initializing the LEDs
led1 = board.get_pin('d:13:o')
led2 = board.get_pin('d:12:o')
led3 = board.get_pin('d:11:o')
led4 = board.get_pin('d:10:o')

# wait for 1s at every count value
wait = 1

# initialise all to False (off)
val_1 = val_2 = val_3 = val_4 = False

# led4 is the least significant bit and led1 is the most significant bit
while True: # this is an infinite loop which won't end untill the terminal is killed
```

```
for ____ in range(2):
        for ___ in range(2):
                for __ in range(2):
                        for _ in range(2):
                                sleep(wait)
                                # Updating the values and printing them
                                led1.write(val_1)
                                led2.write(val_2)
                                led3.write(val_3)
                                led4.write(val_4)
                                print(int(val_1), int(val_2), int(val_3), int(val_4))

                                val_4 = not val_4
                        val_3 = not val_3
                val_2 = not val_2
        val_1 = not val_1
print("\n\n")
```

Output:

```
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 1 0
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
```

ESP8266 and ESP32 with MicroPython

Overview: ESP8266 and ESP32 are low-cost Wi-Fi microchips with full TCP/IP stack and microcontroller capability. They are highly popular for IoT projects due to their wireless capabilities.

IOT WITH ESP8266 and ESP32 :

- **Wireless Functionality:** Both chips support Wi-Fi, which is essential for IoT devices needing to communicate over the network.
- **Cost-Effective:** These chips are very affordable, making them a go-to choice for building networked machines at a lower cost.
- **MicroPython Support:** Running MicroPython on these devices allows developers to leverage

Python's simplicity to control hardware functionality easily and effectively.

- **Powerful Features:** The ESP32, being the successor to ESP8266, includes Bluetooth capabilities and better power efficiency, enhancing its usability in IoT projects that require low energy consumption and connectivity.

MicroPython Script:

1. **Installation:** First, install the esptool module using pip with the command:

   ```
   $ pip install esptool.
   ```

1. **Firmware:** Download the latest MicroPython firmware from the official website. Use esptool to flash this firmware onto your device. Remember to erase the flash memory of the board before installing the new firmware to ensure a clean setup.
2. **Development Environment:** You can write your MicroPython code on a standard computer using any compatible IDE designed for MicroPython. After coding, compile and transfer the script to the ESP8266 or ESP32's memory.

PROGRAM

Program for demonstrating how to control an LED with an ESP8266 or ESP32 using MicroPython:

```
from machine import Pin
import time

# Initialize a pin for the LED
ledPin = Pin(2, Pin.OUT)

# Toggle the LED on and off in a loop
while True:
    ledPin.on()  # Turn on the LED
    time.sleep(1)  # Wait for one second
    ledPin.off()  # Turn off the LED
    time.sleep(1)  # Wait for another second
```

Explanation:

- **Libraries:** We import Pin from machine to interact with GPIO pins, and time for handling delays.
- **LED Control:** We define ledPin as an output pin connected to the LED. The loop continuously turns the LED on and off every second, demonstrating basic pin control with MicroPython.

RESULT: **P**rogram to implement IoT using Python programming is written and verified.

EXPERIMENTS USING ARM

EXP NO: 3.A	**STUDY OF ARM PROCESSORS**
DATE	

Aim :
To study the ARM processor architecture, hardware features and installing OS in Raspberri Pi.

Hardware / Software Requirement:

Raspberry Pi

Overview of ARM Processors in IoT Applications:
ARM processors are the dominant architecture in the Internet of Things (IoT) landscape due to their energy efficiency, scalability, robust ecosystem, and cost-effectiveness. These processors power a wide range of IoT devices, from simple sensors to complex edge computing nodes, enabling real-time data processing, secure operations, and low-power consumption.

Key Features of ARM Processors for IoT
1. Energy Efficiency
ARM processors are based on the RISC (Reduced Instruction Set Computing) principle, which uses a simplified instruction set. This leads to lower power consumption and less heat generation, making ARM ideal for battery-powered and energy-constrained IoT devices.
2. Scalability and Flexibility
The ARM architecture is highly scalable, supporting a wide range of devices from low-end microcontrollers (e.g., Cortex-M series) to high-performance application processors (e.g., Cortex-A series).
Modular design allows customization for specific IoT use cases, such as wearables, smart home devices, and industrial sensors.
3. Real-Time Processing
Many IoT applications require real-time responsiveness. ARM's Cortex-M series is optimized for real-time tasks such as sensor data collection and control systems, ensuring timely and predictable performance.
4. Security
ARM processors integrate security features like TrustZone, which provides hardware-enforced isolation for secure execution of sensitive code and data. This is crucial as IoT devices often handle personal or critical operational data.
5. Robust Software Ecosystem

ARM offers extensive development tools, compilers, and libraries tailored for IoT, simplifying software development, debugging, and optimization.
The widespread adoption of ARM ensures strong community and vendor support, reducing development time and cost.

6. Connectivity
ARM-based chips often include built-in support for common IoT communication protocols such as Wi-Fi, Bluetooth, Zigbee, and cellular, enabling seamless device integration and networking.

Main ARM Processor Families Used in IoT

Series	Typical Use Cases	Key Features
Cortex-M	Sensors, wearables, smart meters	Ultra-low power, real-time processing, cost-effective
Cortex-A	Edge gateways, smart cameras	High performance, supports OS like Linux, multimedia
Cortex-R	Industrial control, automotive	Real-time, high reliability, error management

- Cortex-M Series: Widely used in microcontrollers for low-power, real-time IoT applications.
- Examples include Cortex-M0, M3, and M4, which balance performance and energy efficiency.

- Cortex-A Series: Found in more powerful IoT edge devices and gateways, supporting advanced operating systems and applications that require higher computational power.

- Cortex-R Series: Used in applications demanding high reliability and real-time performance, such as industrial automation and automotive systems.

Architectural Components and Innovations

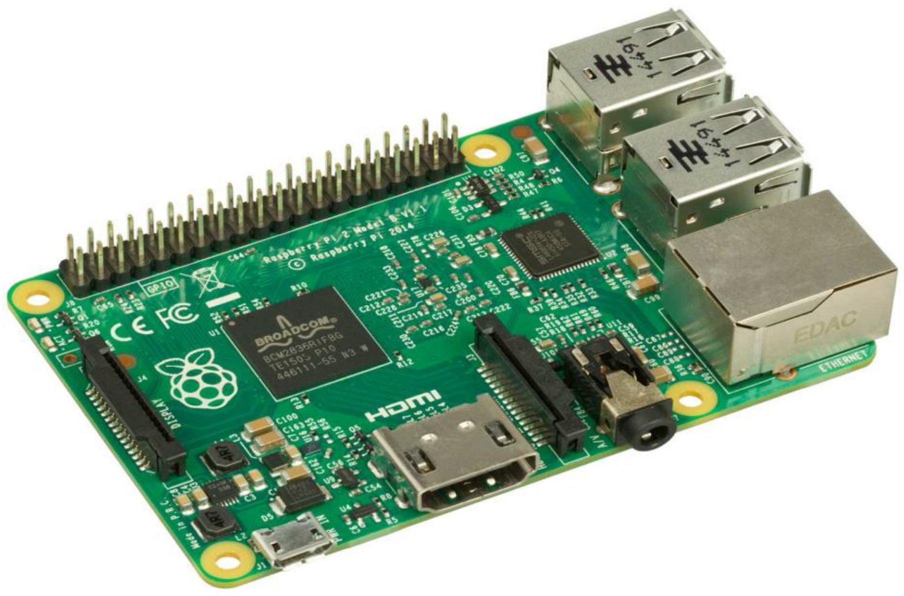
- Pipeline Architecture: Enables simultaneous processing of instruction stages, increasing throughput while maintaining energy efficiency.
- Instruction Sets: Evolve across generations (e.g., ARMv6, ARMv7, ARMv8, ARMv9), with enhancements for performance, security, and AI capabilities.
- DSP and ML Extensions: Features like Helium technology and DSP extensions in Cortex-M processors boost signal processing and machine learning performance at the edge.
- Custom Instructions: Allow for application-specific optimizations without fragmenting the software ecosystem.

Advantages of ARM in IoT

Raspberry Pi 2 (2015) single-board computer, featuring ARMv7 architecture

Raspberry Pi 2 (2015) single-board computer, featuring ARMv7 architecture

Cost-Effectiveness: ARM’s licensing model and widespread adoption drive down chip costs, making large-scale IoT deployment feasible.



Raspberry Pi 2 (2015) single-board computer, featuring ARMv7 architecture

Development Platforms: Popular boards like Raspberry Pi (Cortex-A) and Arduino (Cortex-M) use ARM processors, accelerating prototyping and development.

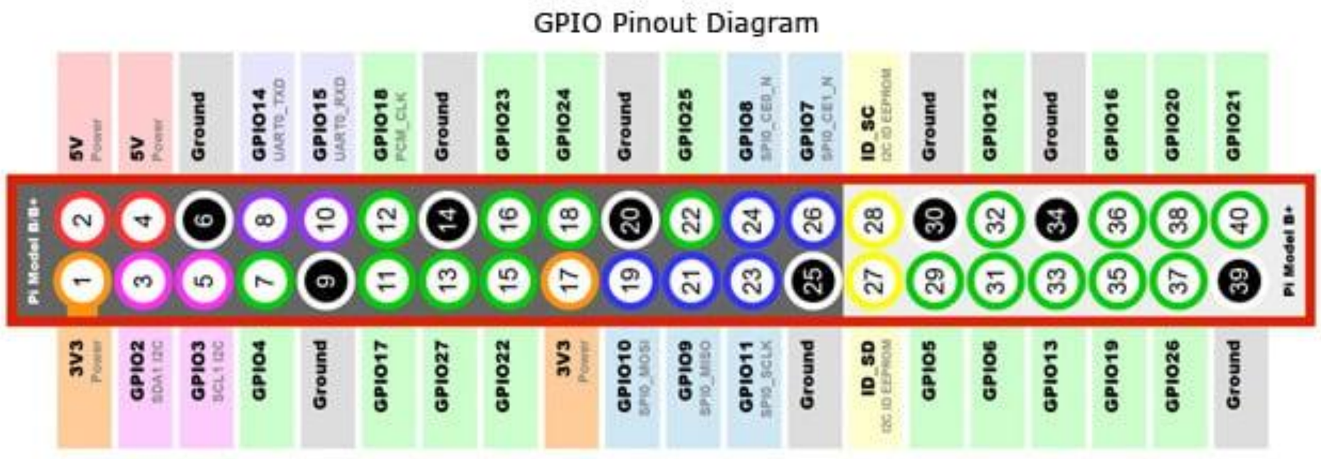
Example Applications

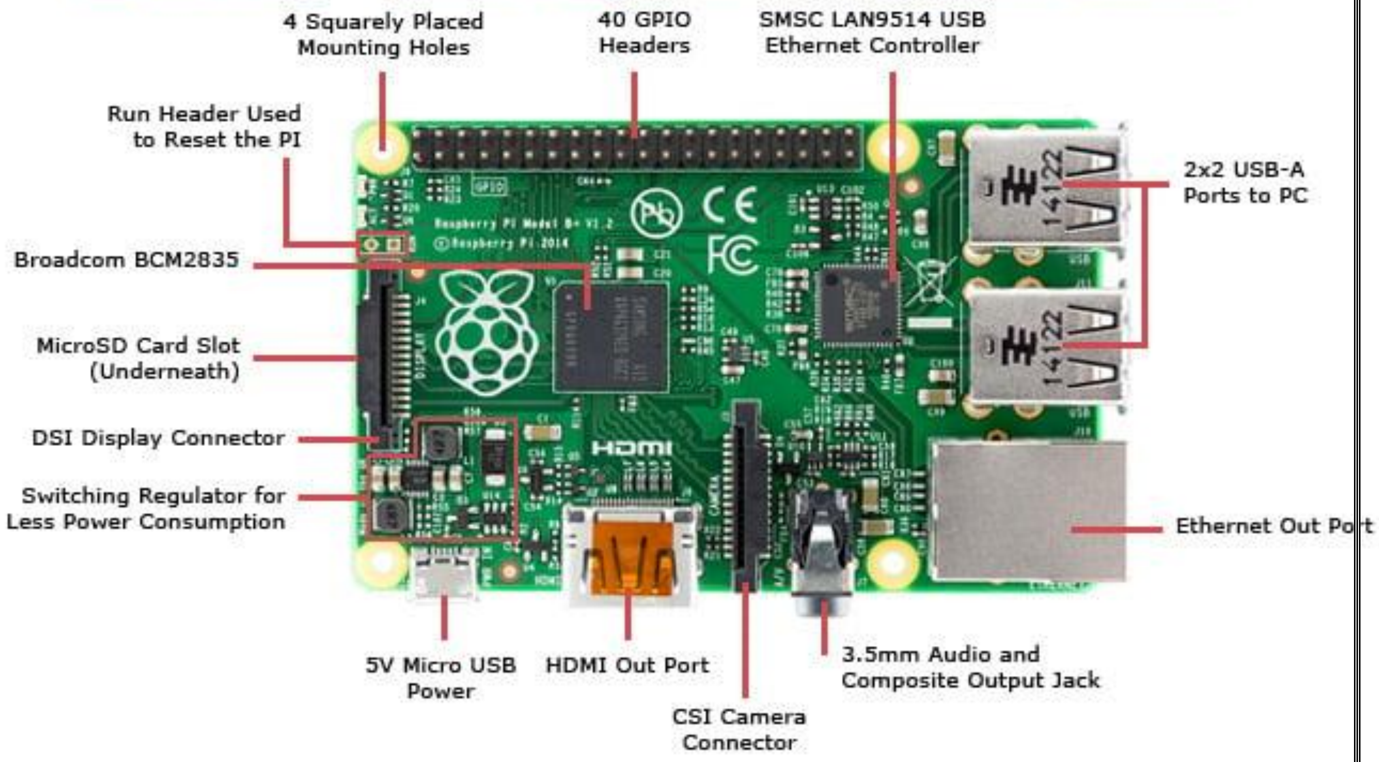
Smart Home Devices: Thermostats, security cameras, and smart lighting often use ARM Cortex-M or Cortex-A processors for efficient local processing and connectivity.

Wearables: Fitness trackers and smartwatches leverage ARM’s low-power operation for extended battery life.

Industrial IoT: ARM Cortex-R and Cortex-M processors power sensors, controllers, and gateways in manufacturing and logistics, providing real-time data and reliability.

HARWARE DETAILS :




4 Squarely Placed Mounting Holes
40 GPIO Headers
SMSC LAN9514 USB Ethernet Controller
Run Header Used to Reset the PI
2x2 USB-A Ports to PC
Broadcom BCM2835
MicroSD Card Slot (Underneath)
DSI Display Connector
Switching Regulator for Less Power Consumption
Ethernet Out Port
5V Micro USB Power
HDMI Out Port
CSI Camera Connector
3.5mm Audio and Composite Output Jack

Components of the Raspberry Pi :

- **USB ports: to connect a mouse, a keyboard, or other peripherals. It comes with two USB 3.0 and two USB 2.0 ports;**
- **Ethernet port: to connect to the internet using an Ethernet cable;**
- **Audio jack: to connect an audio device;**
- **CSI connector: to connect a camera with a CSI ribbon;**

- **HDMI connector: to connect a monitor or TV;**
- **Processor: is the brain of the Raspberry Pi;**
- **MicroSD card slot: to insert a microSD card to store your files and your operating system;**
- **MicroUSB power input: to power up your Pi;**
- **DSI connector: to connect DSI-compatible displays;**
- **Antenna: picks up wireless LAN and Bluetooth signals;**
- **GPIOs (general purpose input output pins): connect devices to interact with the outside world like sensors and outputs like LEDs and motors.**

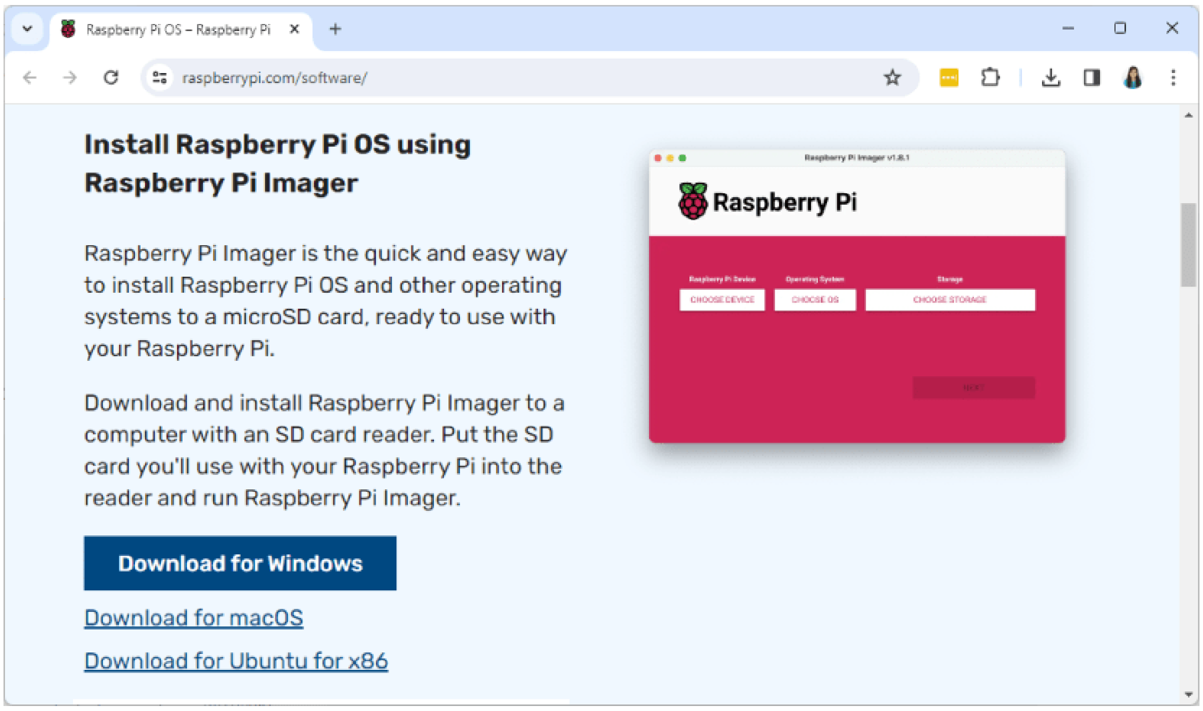
Installing the Operating System

There are several operating systems suitable for the Pi. The official distribution for the Raspberry Pi is Raspberry Pi OS and that's the one we recommend you install.

1) Start by connecting the microSD card to your computer.

**2) Go to the [Raspberry Pi Software page](). **

3) Select and download the Raspberry Pi Imager (a tool to flash the OS on the microSD card) for your computer's operating system.



Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi.

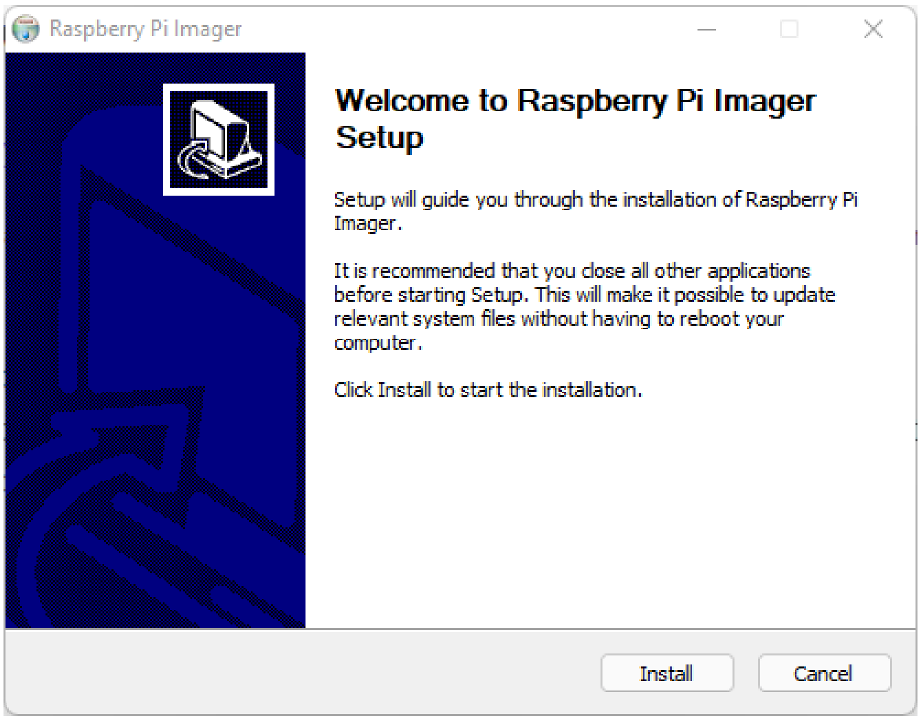
Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

Download for Windows

Download for macOS

Download for Ubuntu for x86

4) Click on the downloaded file to install the Raspberry Pi Imager.


Raspberry Pi Imager

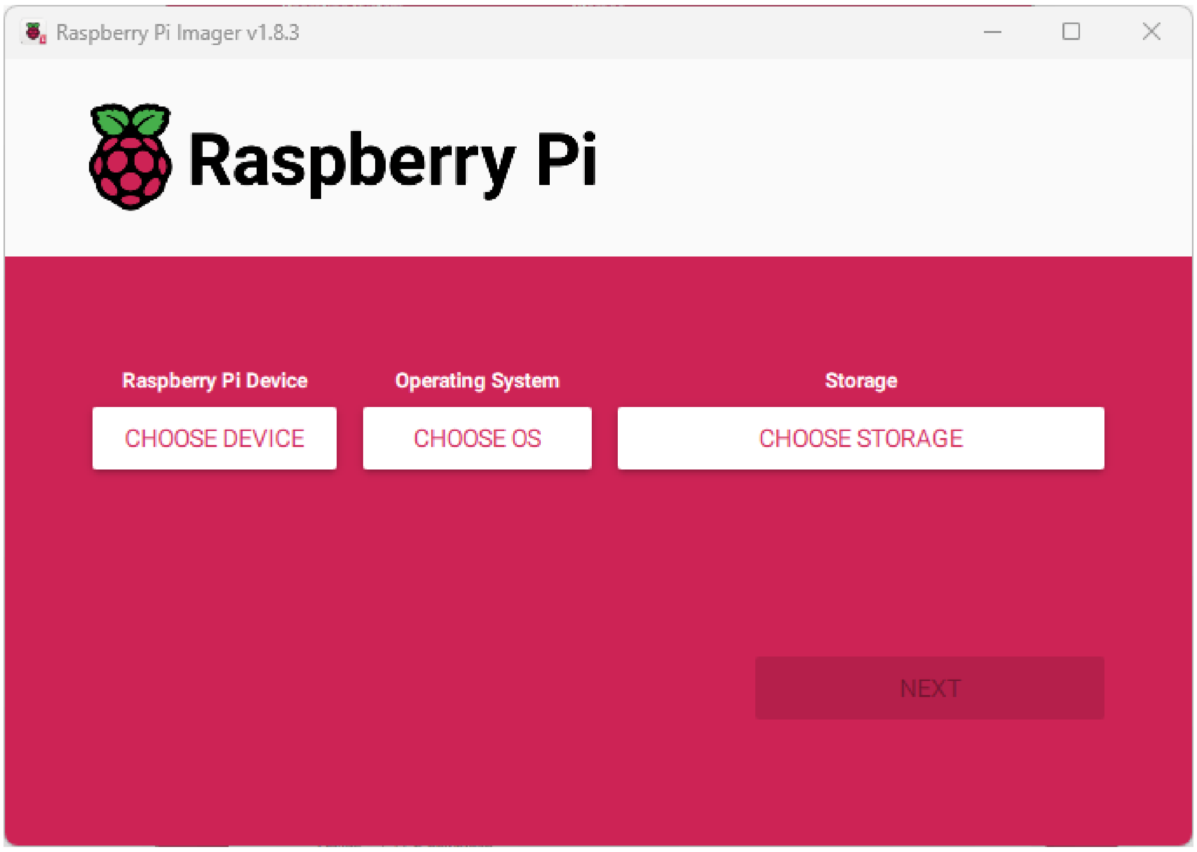
Welcome to Raspberry Pi Imager Setup

Setup will guide you through the installation of Raspberry Pi Imager.

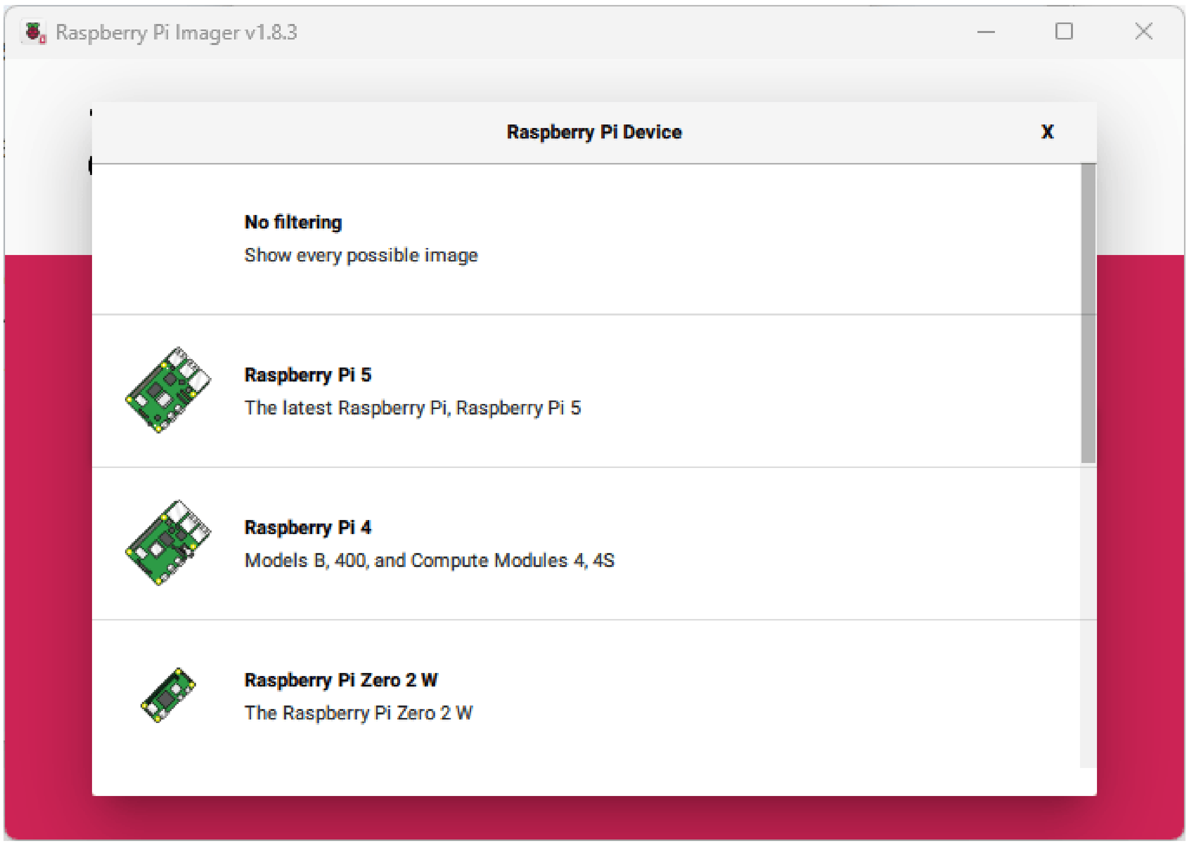
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Install to start the installation.

5) When the installation is complete, the Raspberry Pi Imager will open.


Raspberry Pi Imager v1.8.3

6) Click on Choose Device and select the Raspberry Pi board you’re using.


Raspberry Pi Imager v1.8.3

Raspberry Pi Device

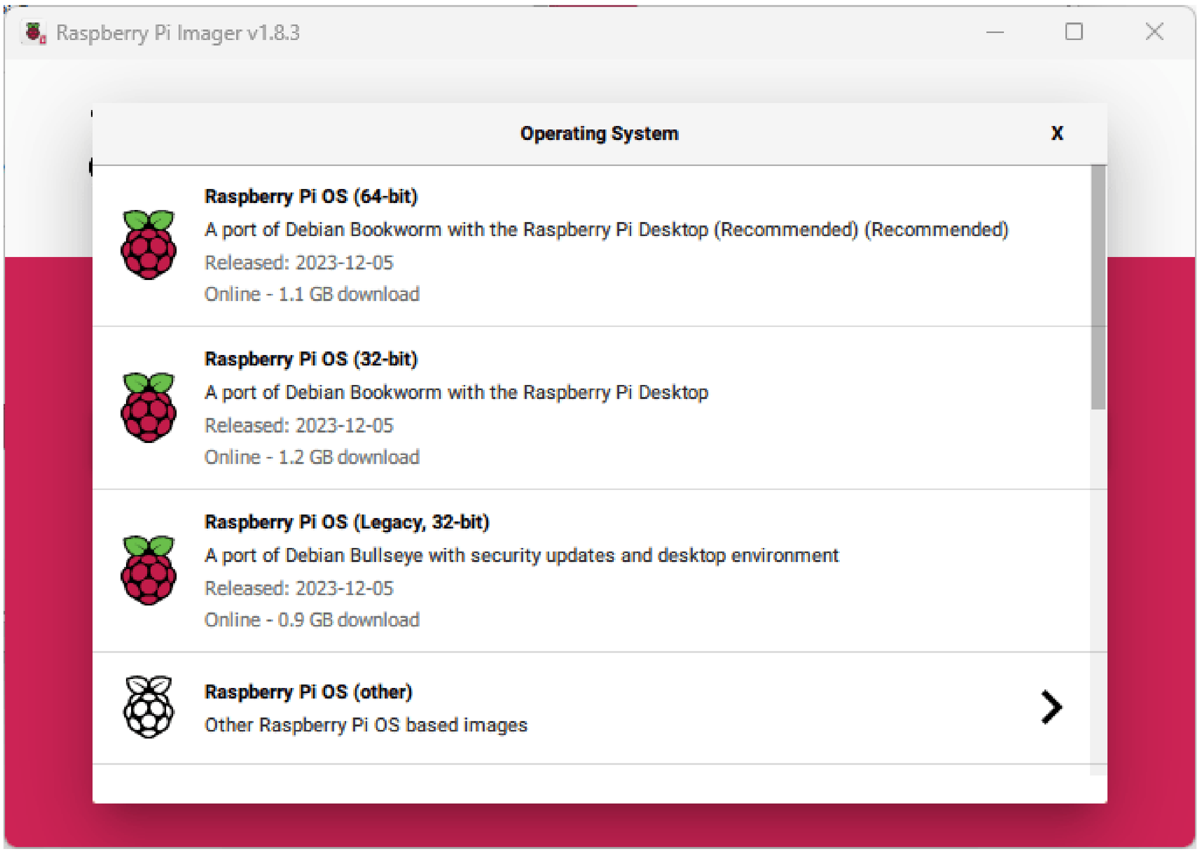
No filtering
Show every possible image

Raspberry Pi 5
The latest Raspberry Pi, Raspberry Pi 5

Raspberry Pi 4
Models B, 400, and Compute Modules 4, 4S

Raspberry Pi Zero 2 W
The Raspberry Pi Zero 2 W

7) Click on Choose OS to select the Operating System. Select the Raspberry Pi OS (32-bit) or (64-bit).


Raspberry Pi Imager v1.8.3

Operating System

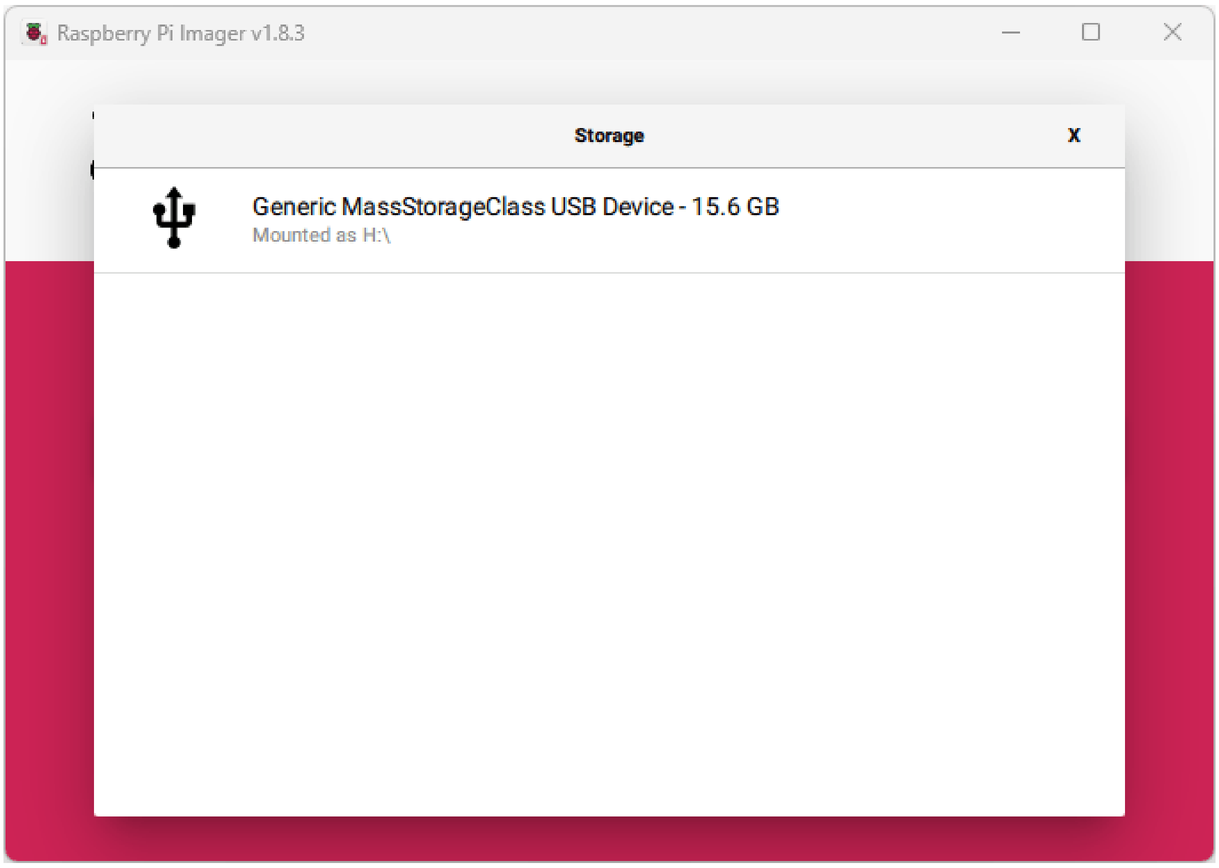
Raspberry Pi OS (64-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended) (Recommended)
Released: 2023-12-05
Online - 1.1 GB download

Raspberry Pi OS (32-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop
Released: 2023-12-05
Online - 1.2 GB download

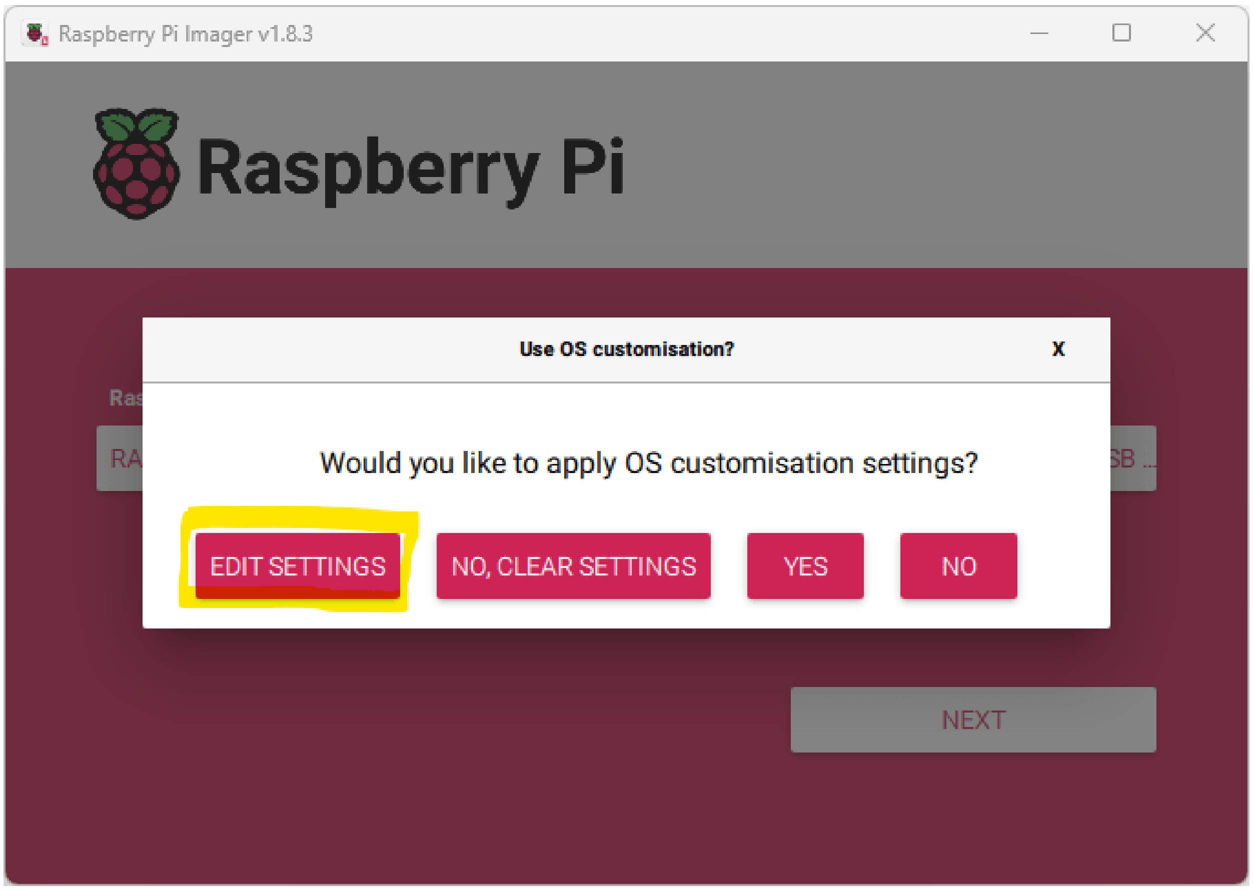
Raspberry Pi OS (Legacy, 32-bit)
A port of Debian Bullseye with security updates and desktop environment
Released: 2023-12-05
Online - 0.9 GB download

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

8) Choose storage. You must choose the microSD card where you want to install the OS.


Raspberry Pi Imager v1.8.3
Storage
X
Generic MassStorageClass USB Device - 15.6 GB
Mounted as H:\

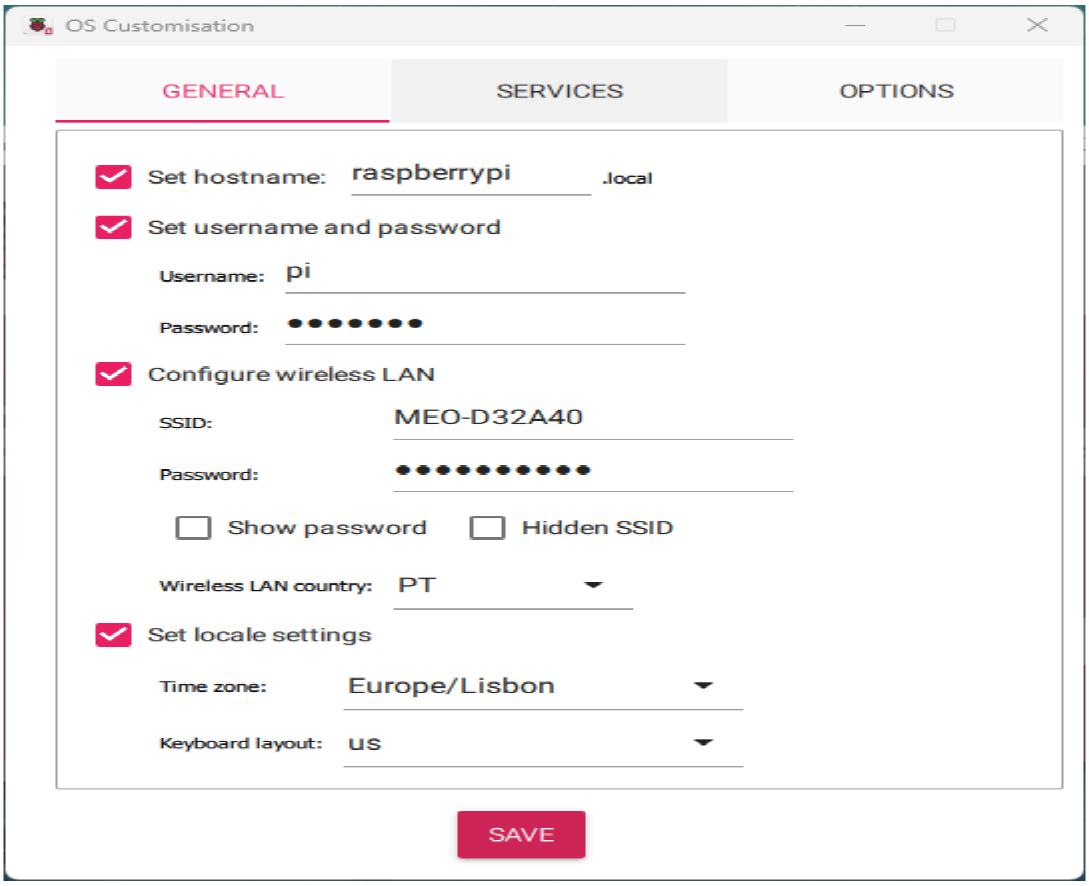
9) Click Next. Next, you'll be asked if you would like to apply customisation settings. Click on Edit Settings to set up the Wi-Fi credentials, and enable SSH.


Raspberry Pi Imager v1.8.3
Raspberry Pi
Use OS customisation?
X
Would you like to apply OS customisation settings?
EDIT SETTINGS
NO, CLEAR SETTINGS
YES
NO
NEXT

10) Under the GENERAL tab, you can set an hostname (the default will be *raspberrypi*), user, and

password, and set Wi-Fi with your local network credentials, so that you can connect to your Raspberry Pi using Wi-Fi later on. You can also select your timezone.

Don't forget to set your Wireless LAN country!



☑ Set hostname: raspberrypi .local

☑ Set username and password

Username: pi

Password: ●●●●●●●●

☑ Configure wireless LAN

SSID: MEO-D32A40

Password: ●●●●●●●●●●●

☐ Show password ☐ Hidden SSID

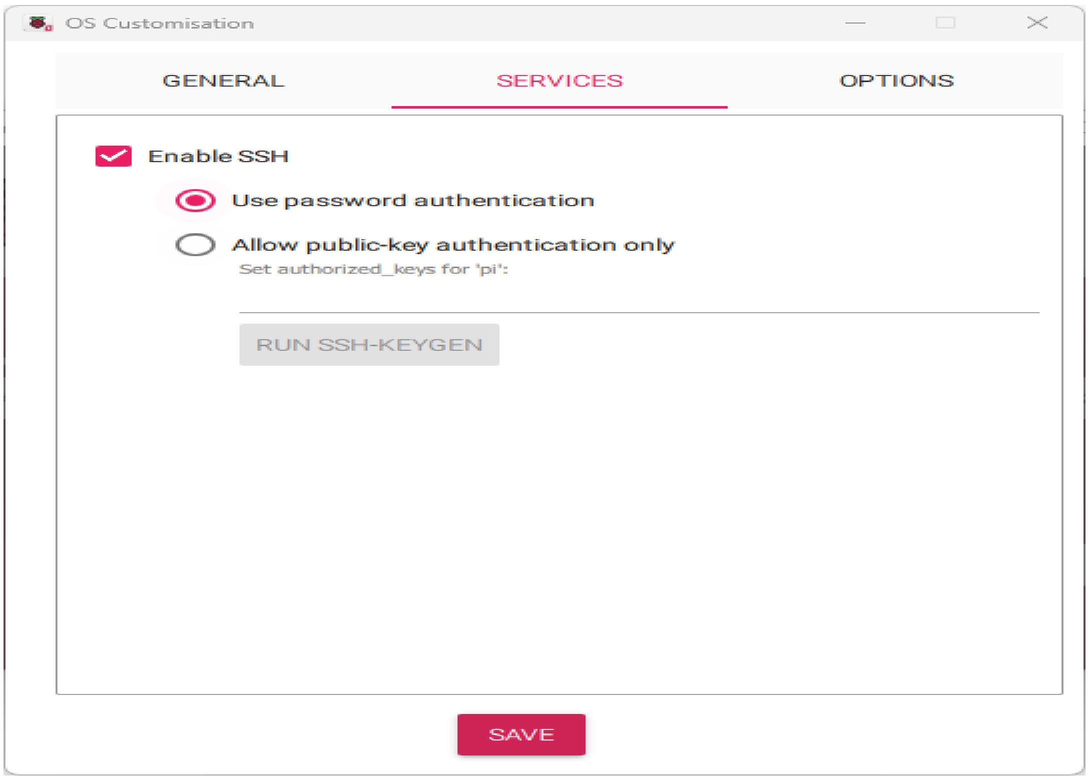
Wireless LAN country: PT

☑ Set locale settings

Time zone: Europe/Lisbon

Keyboard layout: us

11) Then, click on the SERVICES tab and enable SSH with password authentication.



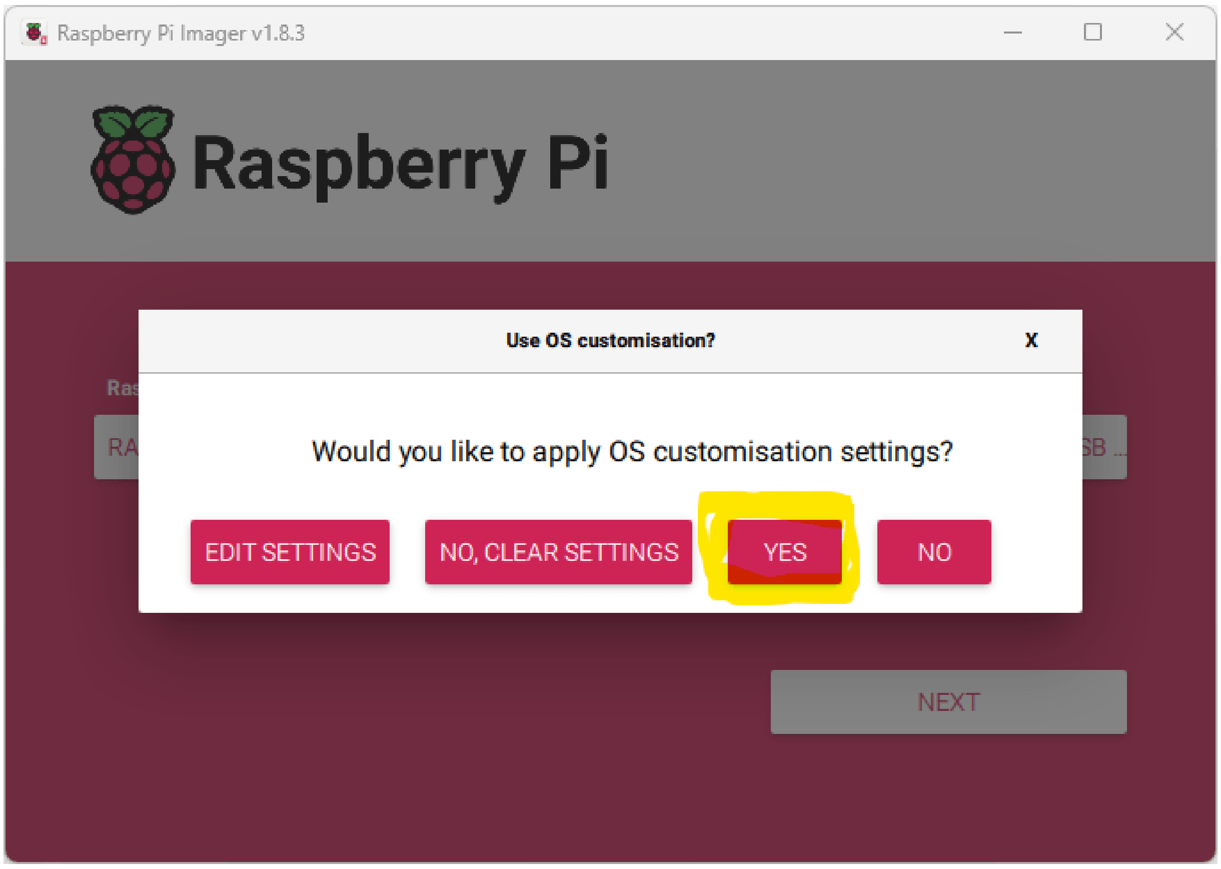
☑ Enable SSH

◉ Use password authentication

○ Allow public-key authentication only

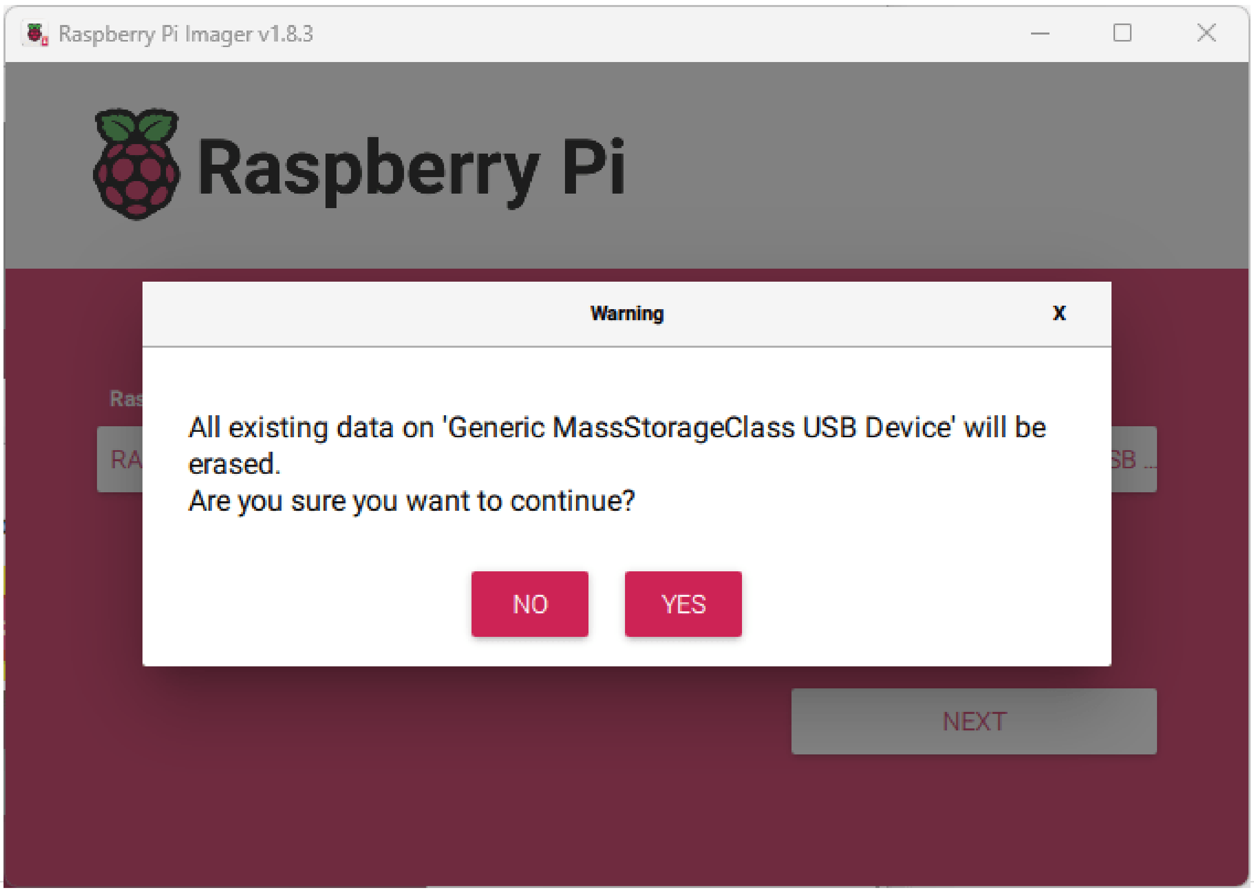
Set authorized_keys for 'pi':

12) Click Save. You'll be asked if you want to apply the OS customisation settings. Click YES.



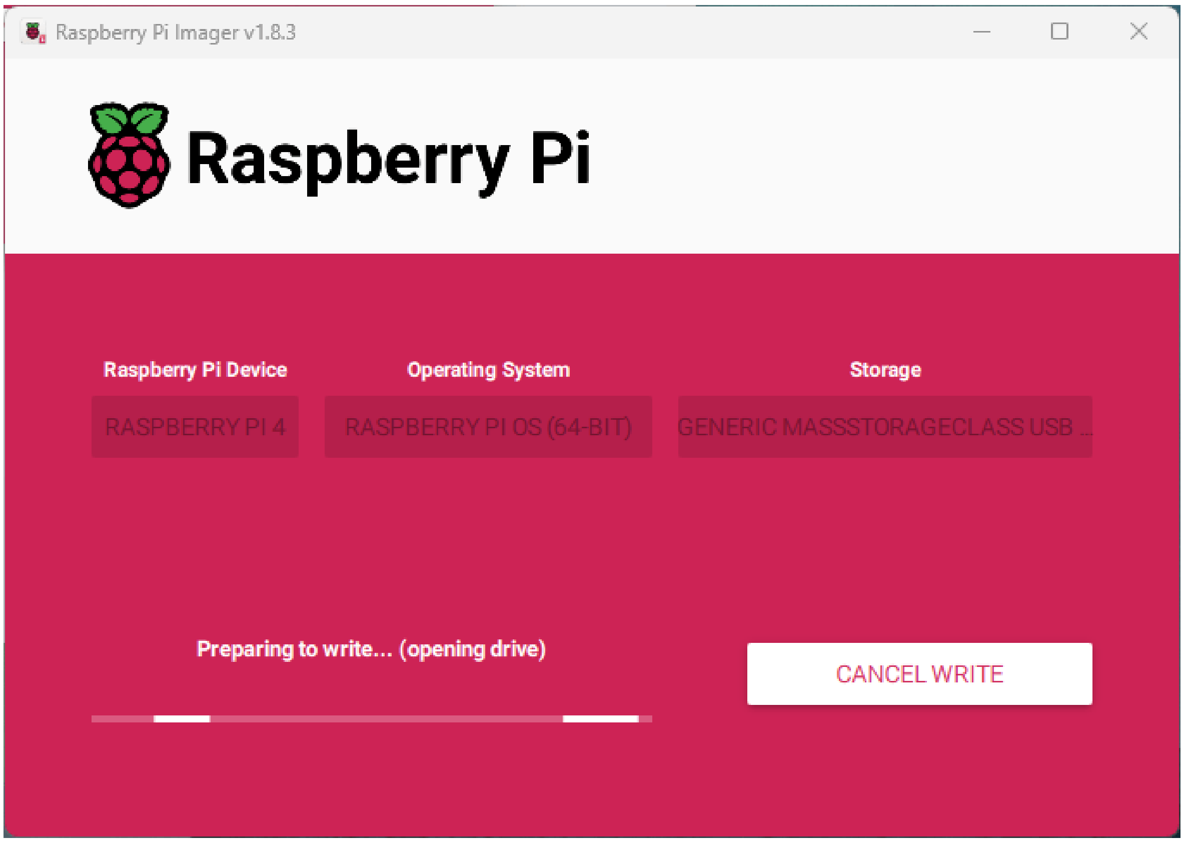
Would you like to apply OS customisation settings?

13) Finally, you'll be asked if you want to continue. Click YES to start burning the Raspberry Pi OS on the microSD card.



All existing data on 'Generic MassStorageClass USB Device' will be erased.
Are you sure you want to continue?

14) Wait a few minutes while it installs the Operating System.


Raspberry Pi Imager v1.8.3

Raspberry Pi

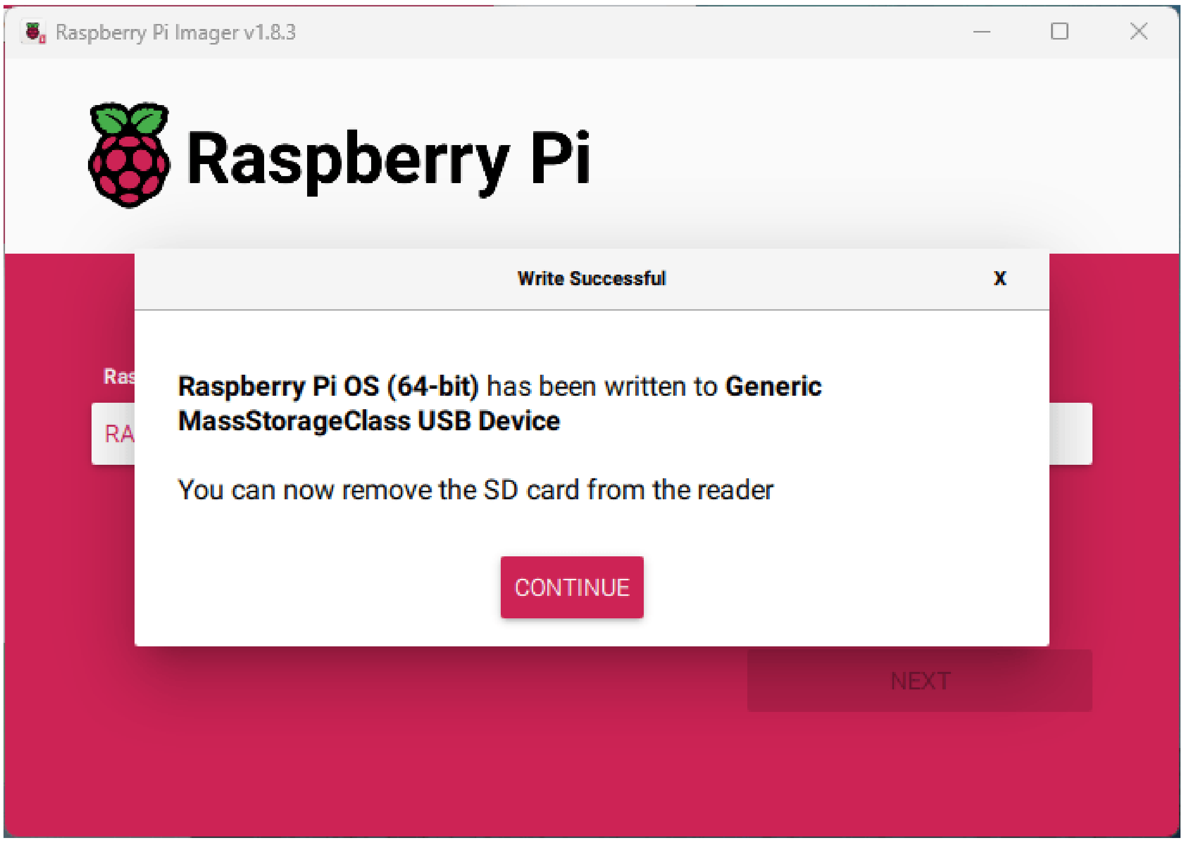
Raspberry Pi Device: RASPBERRY PI 4

Operating System: RASPBERRY PI OS (64-BIT)

Storage: GENERIC MASSSTORAGECLASS USB ...

Preparing to write... (opening drive)

15) When the installation is complete click on Continue. It will eject the microSD card safely.


Raspberry Pi Imager v1.8.3

Raspberry Pi

Write Successful

Raspberry Pi OS (64-bit) has been written to **Generic MassStorageClass USB Device**

You can now remove the SD card from the reader

15) Now, remove the card from your computer and insert it into your Raspberry Pi. Then, apply power to the Raspberry Pi to start it.

RESULT :

The features of ARM processors and specifically Raspberry Pi controller is studied. The installation as PC is done. Raspberry Pi can be used for IoT applications.

EXP NO: 3.B	PYTHON PROGRAMMING IN RASPBERRY PI
DATE	

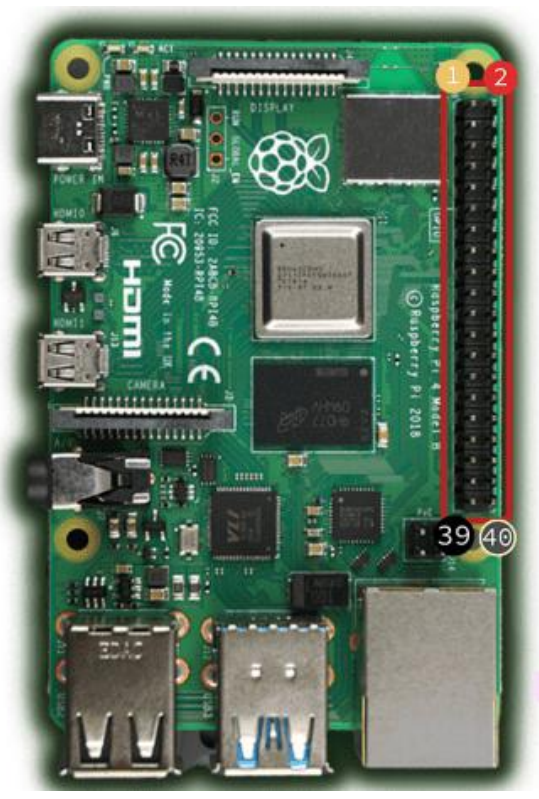
Aim :

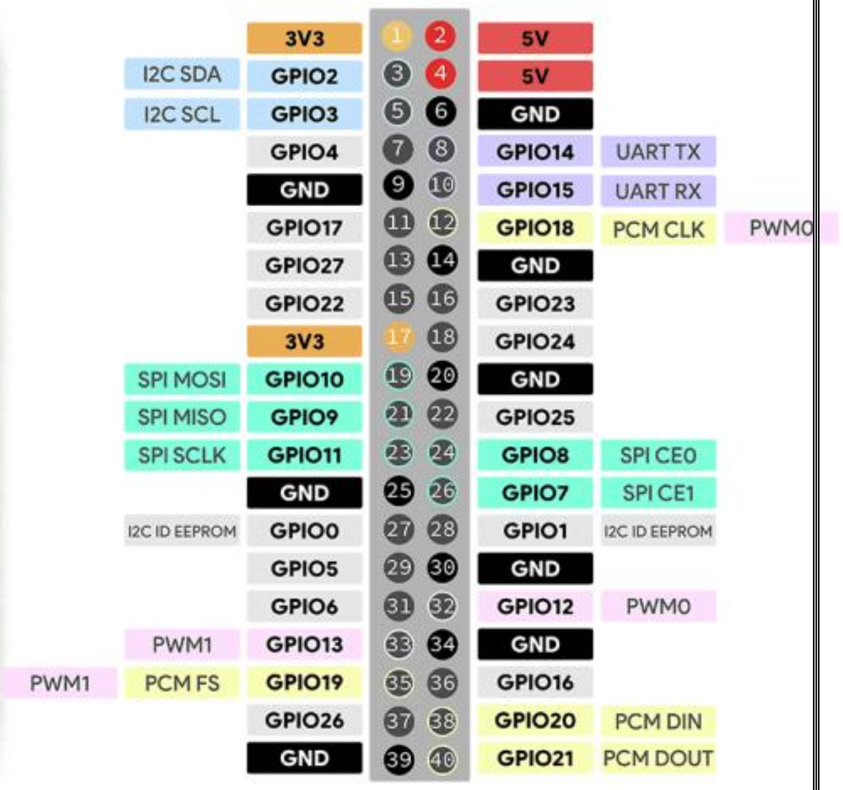
To develop Raspberry Pi platform and Python programming to implement LED blinking applications .

Hardware / Software Requirement:

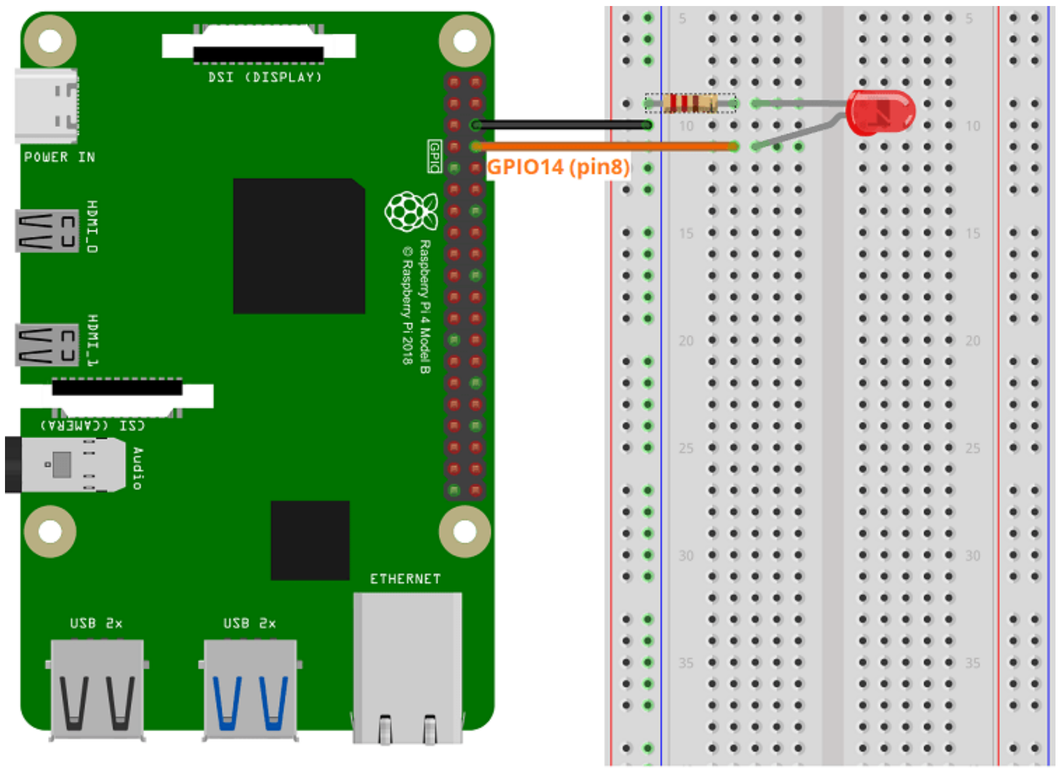
Raspberry Pi board	-1
Breadboard	-1
LED	-1
Resistors -220 ohm	-1
Connecting wires.	

Getting familiar with GPIOs:



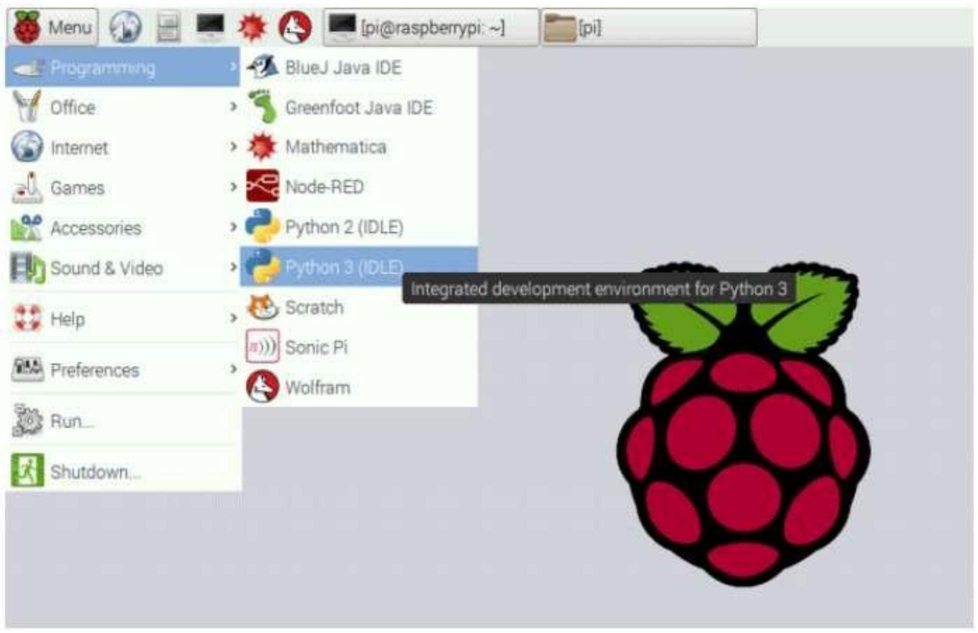

3V3 1 2 5V
I2C SDA GPIO2 3 4 5V
I2C SCL GPIO3 5 6 GND
GPIO4 7 8 GPIO14 UART TX
GND 9 10 GPIO15 UART RX
GPIO17 11 12 GPIO18 PCM CLK PWM0
GPIO27 13 14 GND
GPIO22 15 16 GPIO23
3V3 17 18 GPIO24
SPI MOSI GPIO10 19 20 GND
SPI MISO GPIO9 21 22 GPIO25
SPI SCLK GPIO11 23 24 GPIO8 SPI CE0
GND 25 26 GPIO7 SPI CE1
I2C ID EEPROM GPIO0 27 28 GPIO1 I2C ID EEPROM
GPIO5 29 30 GND
GPIO6 31 32 GPIO12 PWM0
PWM1 GPIO13 33 34 GND
PWM1 PCM FS GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20 PCM DIN
GND 39 40 GPIO21 PCM DOUT

Connection diagram:

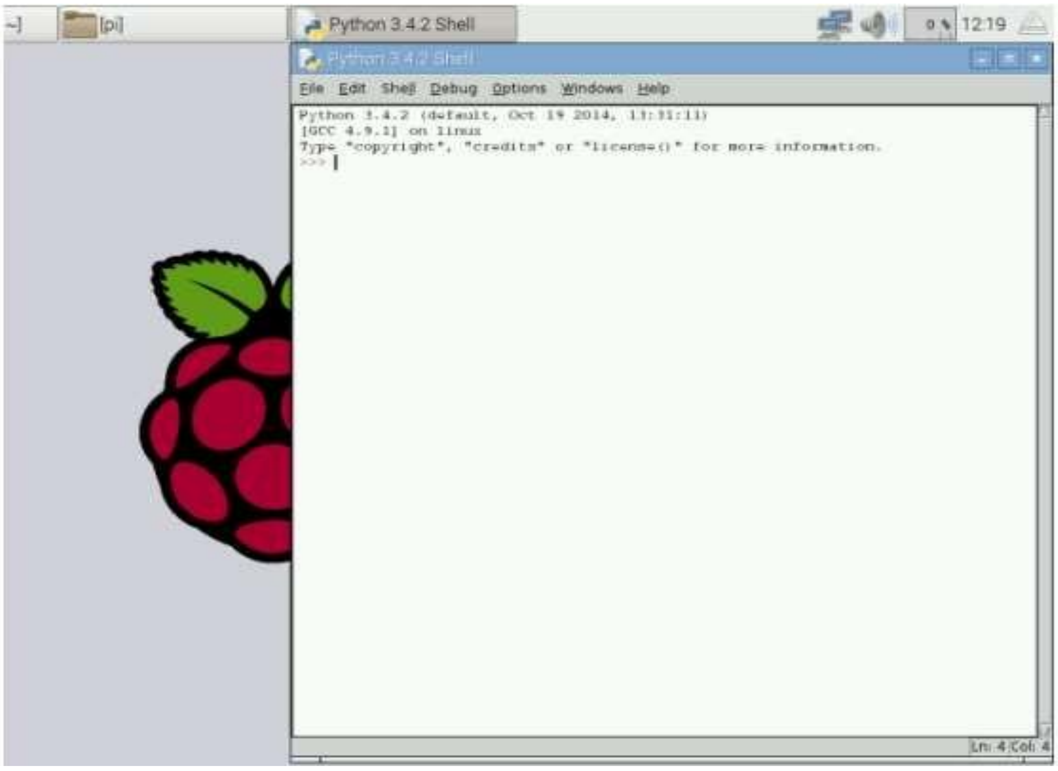


Procedure:

Step 1: On the desktop, go the Start Menu and choose for the PYTHON 3, as shown in figure below

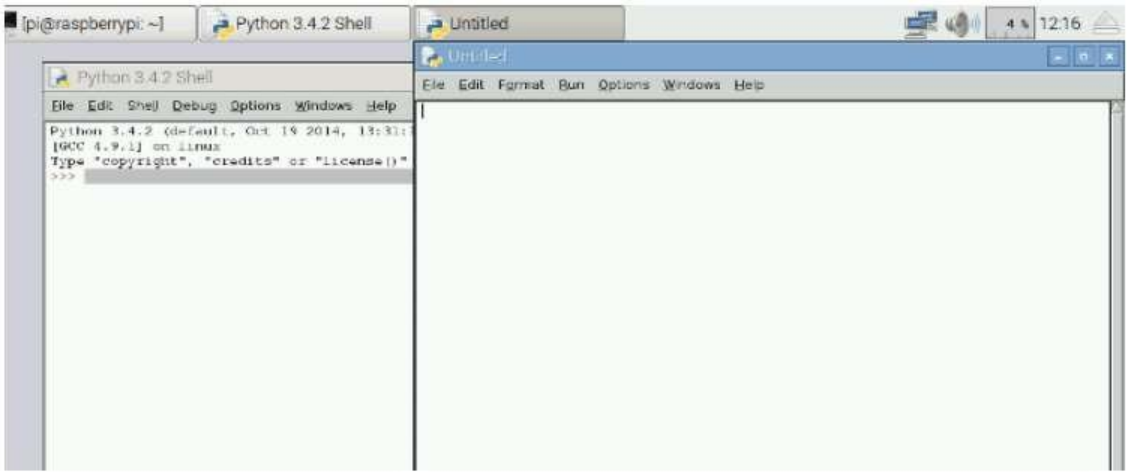


Step 2: After that, PYTHON will run and you will see a window as shown in below figure.

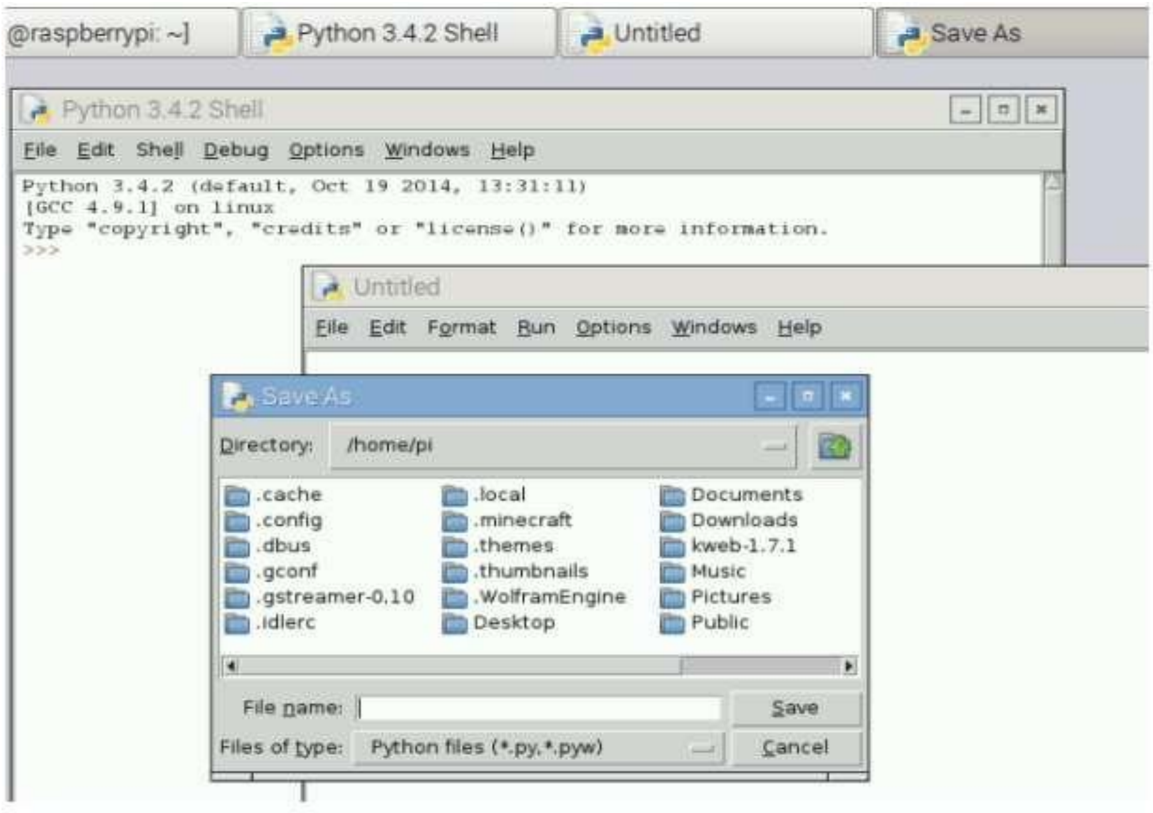


```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 
```

Step 3: After that, click on *New File* in *File Menu*, you will see a new window Open


[pi@raspberrypi: ~]
Python 3.4.2 Shell
Untitled
12:16
File Edit Shell Debug Options Windows Help
File Edit Format Run Options Windows Help

Step 4: Save this file as blinky on the desktop.

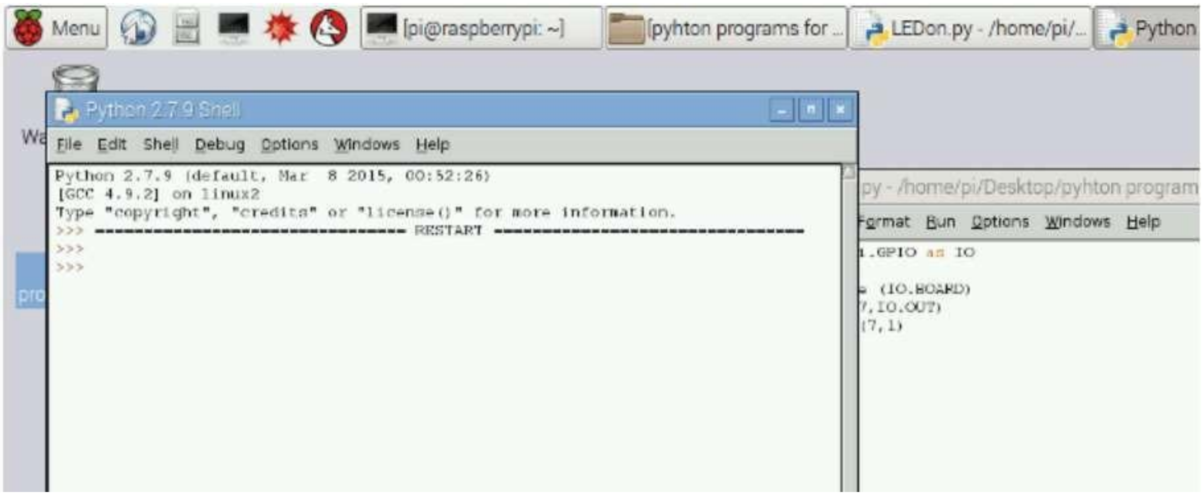


```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

Step 5: Connect 5v of Raspberry Pi board with LED Anode through GPIO ports

Step 6: Connect the cathode of LED to ground of Raspberry Pi board.

Step 7: After that write the program for blinky as given below and execute the program by clicking on “RUN” on “DEBUG” option.


Menu
[pi@raspberrypi: ~]
[pyhton programs for ...
LEDon.py - /home/pi/...
Python
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
py - /home/pi/Desktop/pyhton program
Format Run Options Windows Help

```
Python 2.7.9 (default, Mar  8 2015, 00:52:26)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
>>>
```

```
i.GPIO as IO

= (IO.BOARD)
7,IO.OUT)
(7,1)
```

Step 6: Verify the LED blinking status with the respective time delay given by the program

Program:

```
import Rpi.GPIO as GPIO import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(40.GPIO.OUT)
While True:
        GPIO.output(40,True)
        time.sleep(1)
         GPIO.output(40,False)
         time.sleep(1)
```

OUTPUT:

LED ON

LED OFF

Result:

Thus, the development of Raspberry Pi platform and Python programming to implement LED blinking applications was executed successfully and the output was verified.

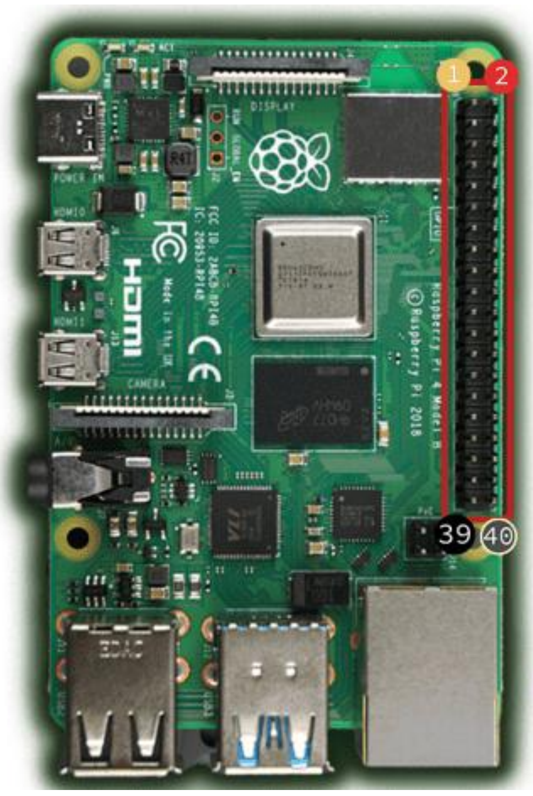
EXP NO: 3.C	**INTERFACING SENSORS WITH RASPBERRY PI**
DATE	

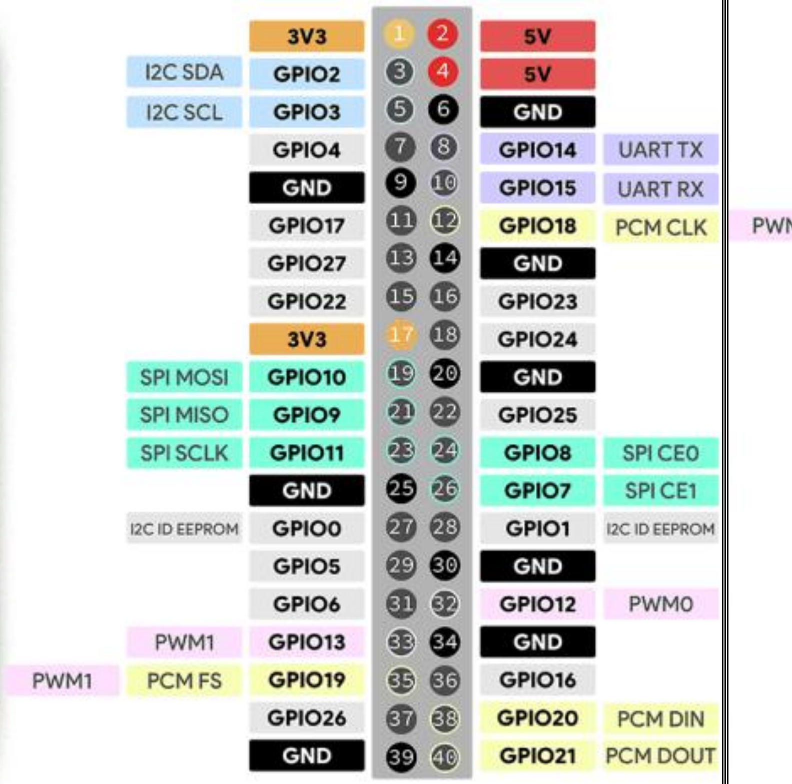
Aim :
To interface sensors with Raspberry Pi platform and Python programming to illustrate the operation of sensors.

Hardware / Software Requirement:

Raspberry Pi board	-1
Breadboard	-1
LED Different colours	-2
IR Sensor or LDR Sensor	-1
Connecting wires.	

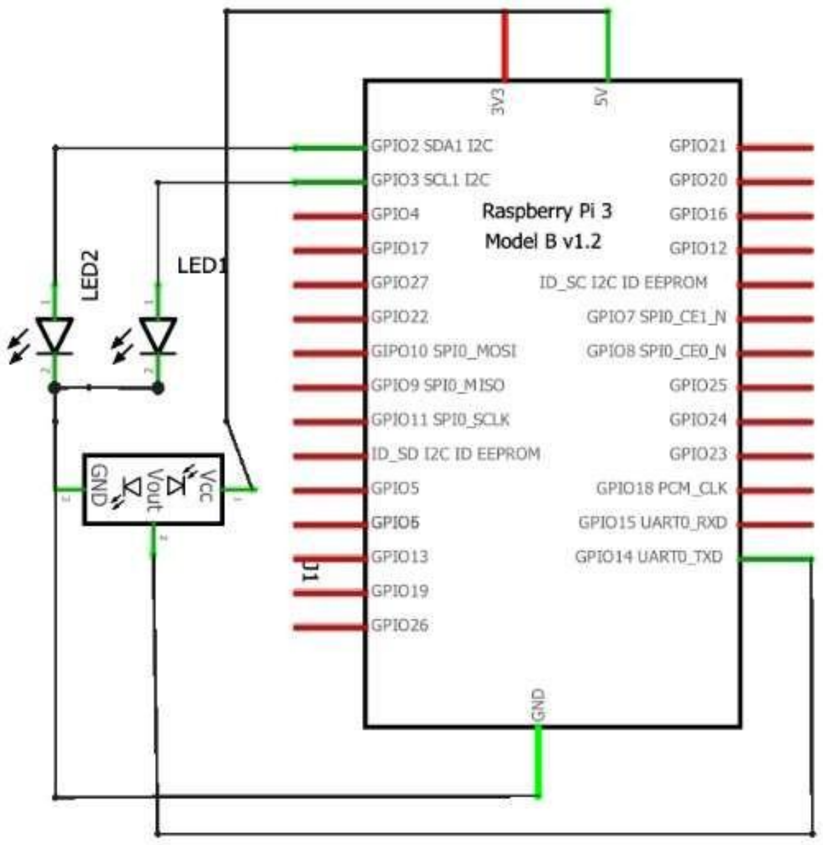
Getting familiar with GPIOs:





3V3 1 2 5V
I2C SDA GPIO2 3 4 5V
I2C SCL GPIO3 5 6 GND
GPIO4 7 8 GPIO14 UART TX
GND 9 10 GPIO15 UART RX
GPIO17 11 12 GPIO18 PCM CLK PWM
GPIO27 13 14 GND
GPIO22 15 16 GPIO23
3V3 17 18 GPIO24
SPI MOSI GPIO10 19 20 GND
SPI MISO GPIO9 21 22 GPIO25
SPI SCLK GPIO11 23 24 GPIO8 SPI CE0
GND 25 26 GPIO7 SPI CE1
I2C ID EEPROM GPIO0 27 28 GPIO1 I2C ID EEPROM
GPIO5 29 30 GND
GPIO6 31 32 GPIO12 PWM0
PWM1 GPIO13 33 34 GND
PWM1 PCM FS GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20 PCM DIN
GND 39 40 GPIO21 PCM DOUT

1



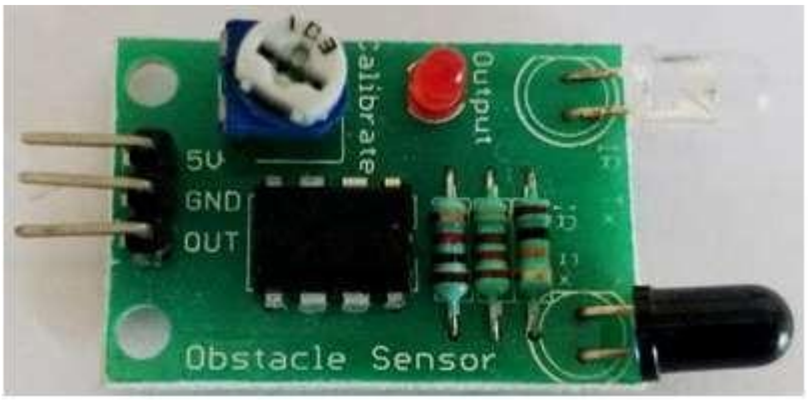
Circuit Connection

Theory: IR Sensor Module

IR (Infrared) Sensor IR (Infrared) Sensor works by emitting infrared signal/radiation and receiving of the signal when the signal bounces back from any obstacle. In other words, the IR Sensor works by continuously sending signal (in a direction) and continuously receive signal, if comes back by bouncing on any obstacle in the way.

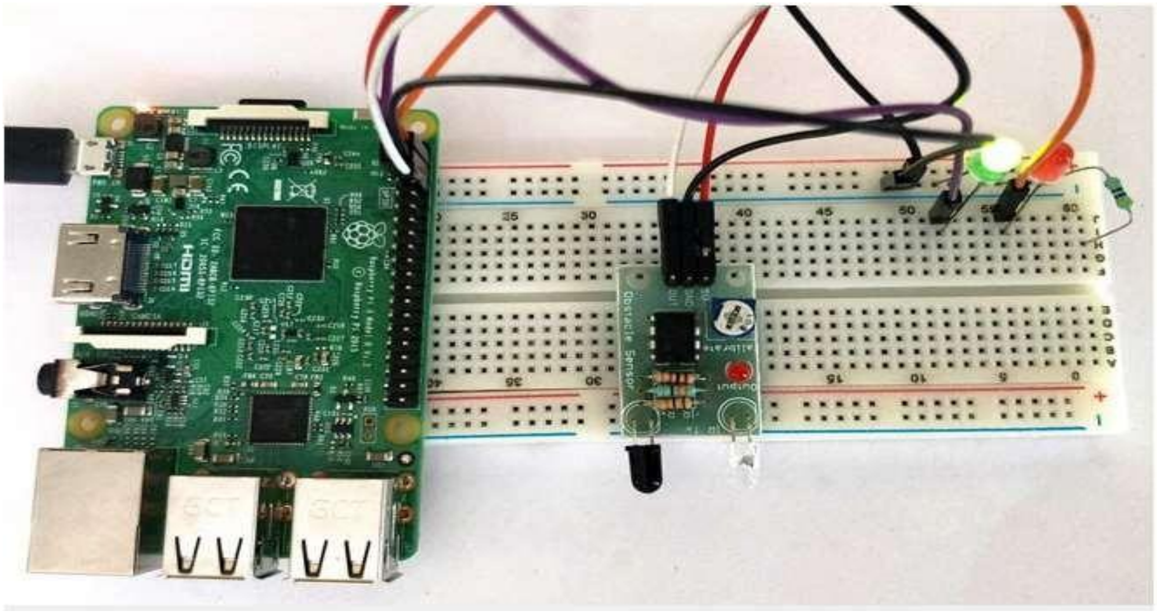
Components:

IR Sensor Emitter: This component continuously emits the infrared signal Receiver: It waits for the signal which is bounced back by obstacle
Indicator: On board LED to signal if obstacle is deducted by the sensor Output: Could be used as Input for further processing of the signal Ground: Ground/Negative point of the circuit
Voltage: Input 3.3V In this tutorial we will learn how we can Interface an IR sensor with Raspberry pi.



These sensors are most commonly use in small robots like line follower robot, Edge avoiding robot etc.. Simply putting, it can detect the presence of objects before it and also differentiate between white and black colour. So lets learn how to interface this sensor with Raspberry Pi. In this experiment, when there is no object in front of IR sensor then the Red LED remains turned on and soon as we put something in front of IR sensor then red LED turns off and Green LED turn on. This circuit can also serve as Security Alarm Circuit.

IR Sensor Module: IR sensors (Infrared sensor) are modules which detect the presence of objects before them. If the object is present it give 3.3V as output and if it is not present it gives 0 volt. This is made possible by using a pair of IR pair (transmitter and receiver), the transmitter (IR LED) will emit an IR ray which will get reflected if there is a object present before it. This IR ray will be received back by the receiver (Photodiode) and the output will be made high after amplified using an op-amp link LM358 The IR Sensor used in this assignment is like all IR sensor it has three pins which are 5V, Gnd and Out respectively. The module is powered by the 5V pin from Raspberry Pi and the out pin is connected to GPIO14 of Raspberry Pi. The potentiometer on top of the module can be used to adjust the range of the IR sensor.



Circuit Diagram and Explanation:

The circuit diagram for connecting Raspberry Pi with IR sensor is shown below.

As you can see the circuit diagram is very simple. We have directly powered the IR module from the 5V and Ground Pin of Raspberry Pi. The output pin of the IR module is connected to the GPIO14. We have also used two LED (Green and Red) to indicate the status of the object. These two LEDs are connected to GPIO3 and GPIO2 respectively. Since the GPIO pins of Raspberry Pi are 3.3V, a current limiting resistor is not mandatory. However if desired a resistor of value 470 ohms can be added between the ground pin of LEDs and Raspberry Pi. The whole circuit is powered by a 5V mobile charger through the micro USB port of the Raspberry pi.

PROGRAM:

```
import RPi.GPIO as IO

import time

IO.setwarnings(False)

IO.setmode(IO.BCM)


IO.setup(2,IO.OUT) #GPIO 2 -> Red LED as output

IO.setup(3,IO.OUT) #GPIO 3 -> Green LED as output

IO.setup(14,IO.IN) #GPIO 14 -> IR sensor as input


while 1:


    if(IO.input(14)==True): #object is far away

        IO.output(2,True) #Red led ON

    IO.output(3,False) # Green led OFF


    if(IO.input(14)==False): #object is near

    IO.output(3,True) #Green led ON

    IO.output(2,False) # Red led OFF
```

Procedure:

Step 1**:** Connections are made as per the circuit diagram.

Step 2: On the desktop, go the Start Menu and choose for the PYTHON 3

Step 3: After that, PYTHON will run and you will see a programming window Step 4: Click on *New File* in *File Menu*, you will see a new window

Step 5: After that write the program for IR sensor module and execute the program by clicking on “RUN” on “DEBUG” option.

Step 6: Verify the LED’s blinking status with respect to the program.

Result:

Thus, the interfacing sensor with Raspberry Pi platform was executed and the output was verified successfully.

EXP NO: 4	IMPLEMENTATION OF IOT USING BLYNK
DATE	

AIM:

To implement IoT using Blynk and demonstrate blinking of LED.

HARDWARE & SOFTWARE TOOLS REQUIRED:

S.No.	Software Requirements	Quantity
1	Blynk APP	1
2	ESP8266 NODE MCU	1

ABOUT BLYNK:

Blynk is an IoT platform with customizable mobile apps, private cloud, rules engine, and device management analytics dashboard, designed for easy and customizable Internet of Things applications. Designing dashboard on Blynk App for IoT projects is really easy, you just have to organize buttons, sliders, graphs, and alternative widgets onto the screen. We can also edit the widgets as per our requirements.

With the help of Blynk, the software side gets easier than the hardware. Blynk is perfect for interfacing with simple projects like monitoring the temperature of your room or turning lights on and off remotely Here, in this project we are controlling a LED using Blynk App and Esp8266. Earlier we have controlled the LED using ESP32. So, not only with ESP8266, it’s easy to interface Blynk app with Raspberry Pi, Arduino and other microcontrollers.

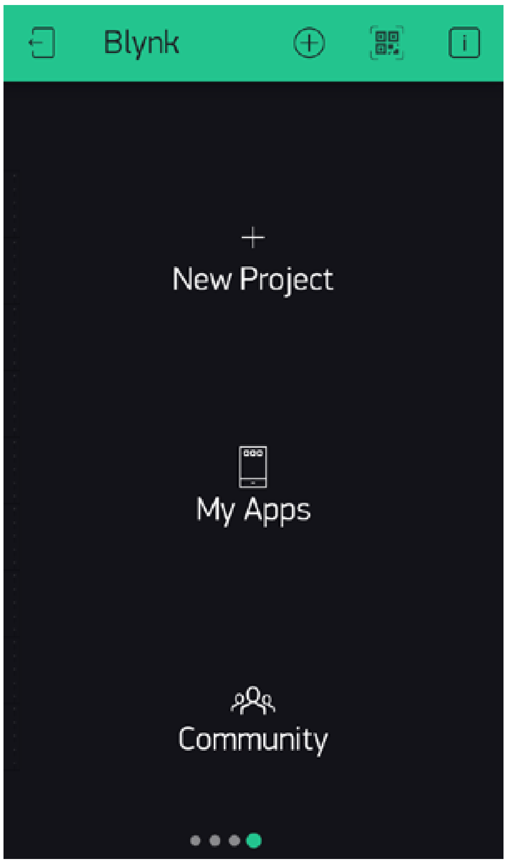
PROCEDURE:

Blynk App Dashboard Setup for Controlling LED

To control LED with Blynk app, download and install the Blynk app from Google or Apple app store.

Then, if you are new to Blynk app, create a new account by using your Email and password.

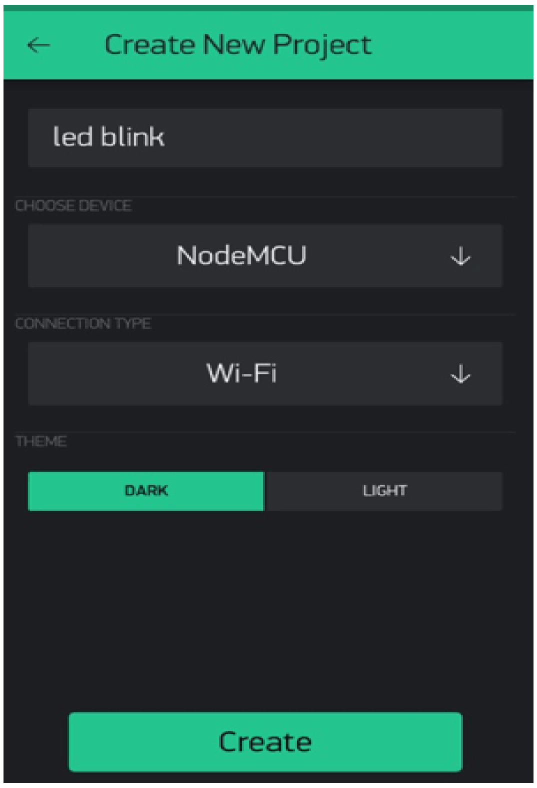
After sign up click on ‘New Project’ to start your project.



Now provide your project a name and as blynk app works with diffferent hardware models. Thus opt for your device from *choices*.

Here, we are using ESP8266 node MCU, so I am proceeding with node MCU.
While selecting your board opt for your association sort whether or not it's Wi-Fi or LAN or USB association.

After these steps click on 'Create' button to form your project.



Create New Project

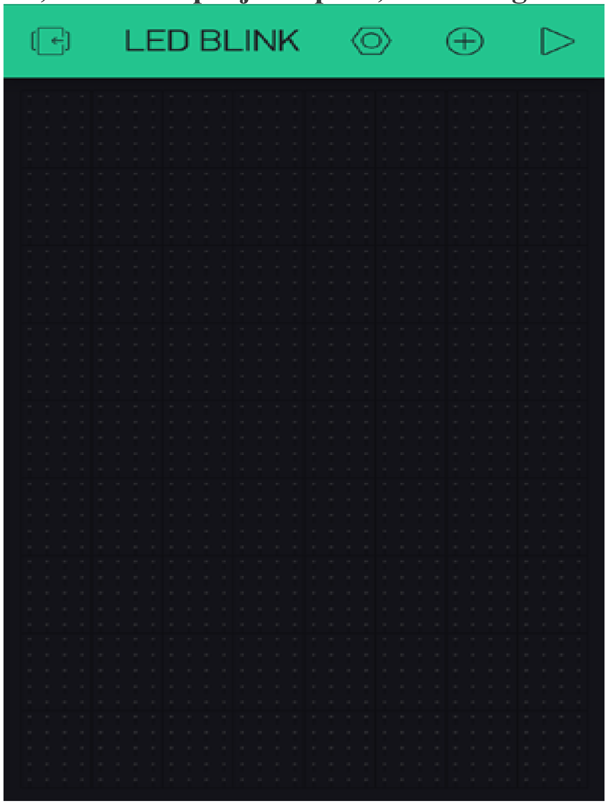
led blink

CHOOSE DEVICE: NodeMCU

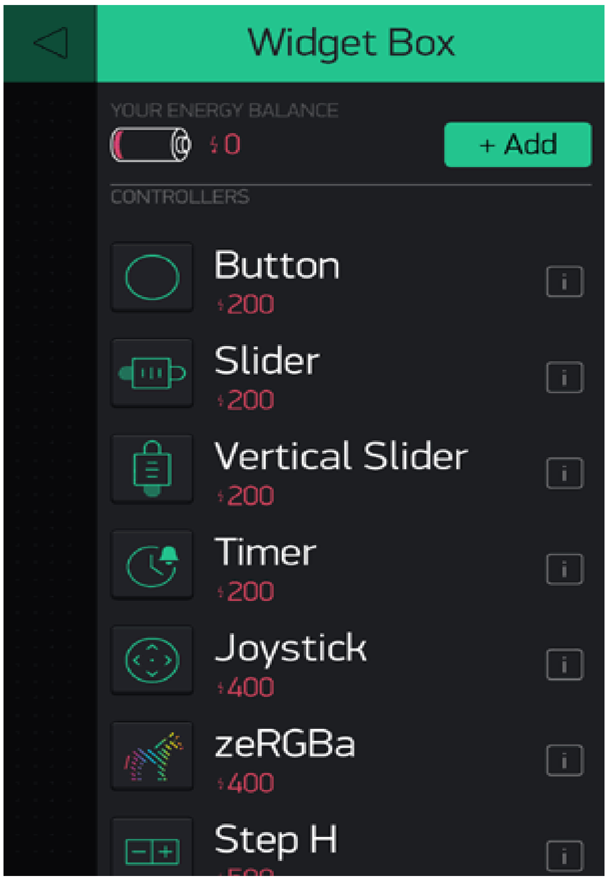
CONNECTION TYPE: Wi-Fi

THEME: [✓ DARK] LIGHT

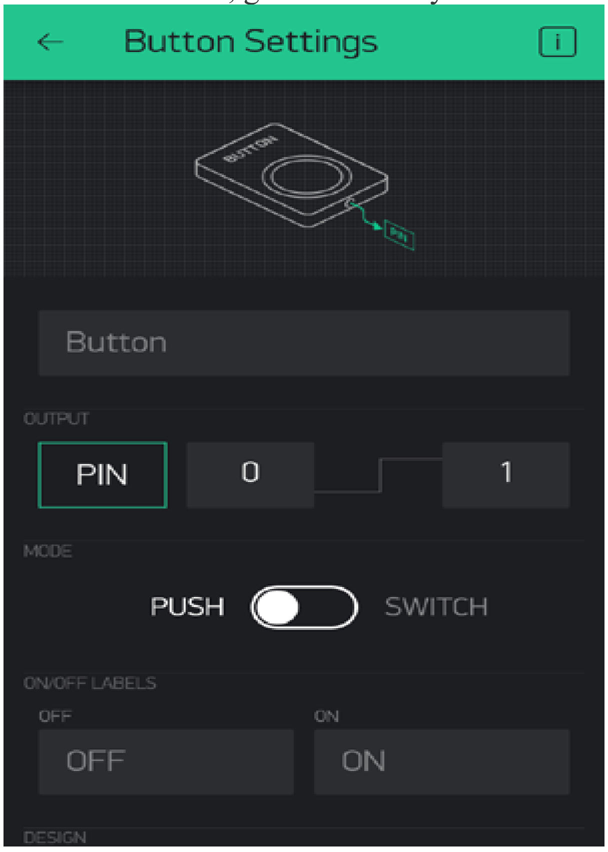
As, the blank project opens, add Widgets to it by clicking on Add button (Plus sign button)



Now after this click on 'Button' to add a button in your project.

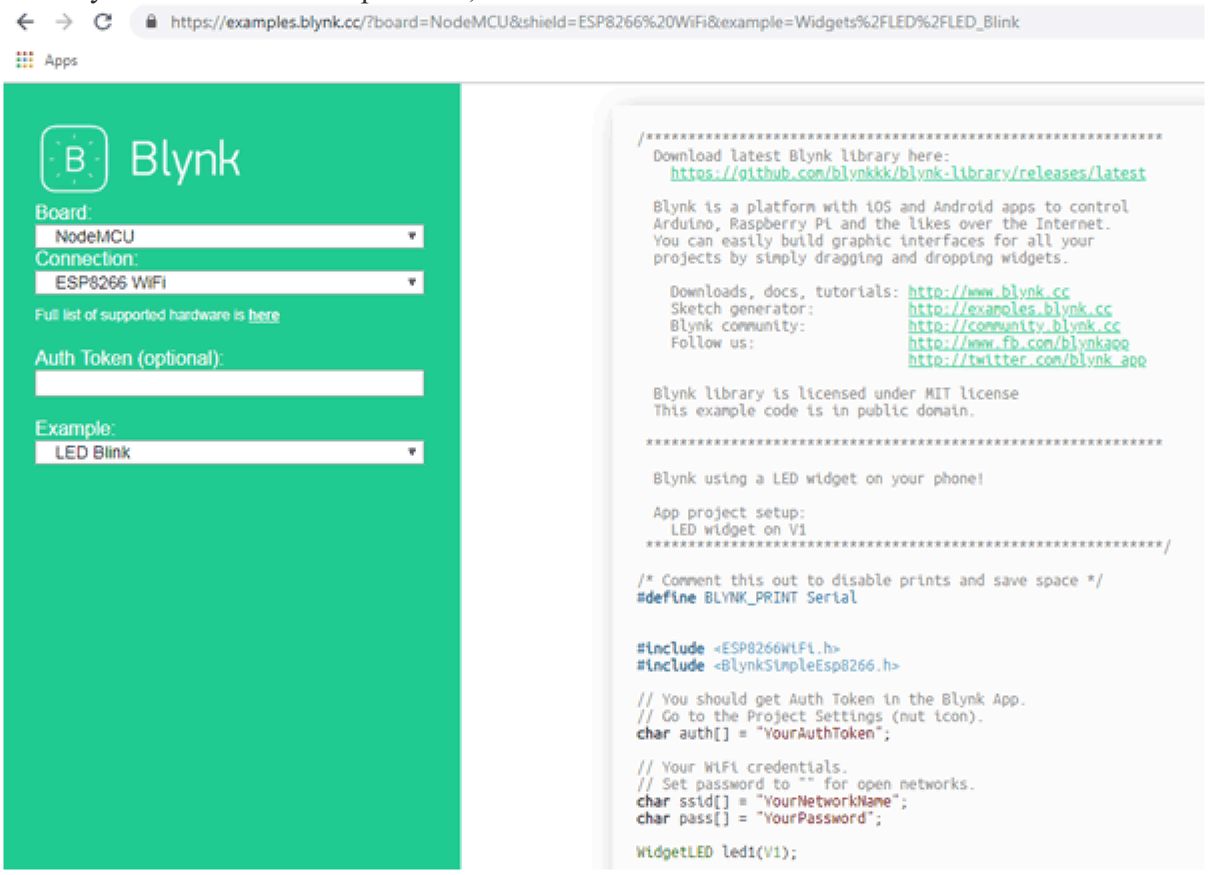


Now in button settings provide a name to your button. After assign the pin number to the 'OUTPUT'. Also, give names to your On/Off labels.



NODE MCU Setup for Blynk App

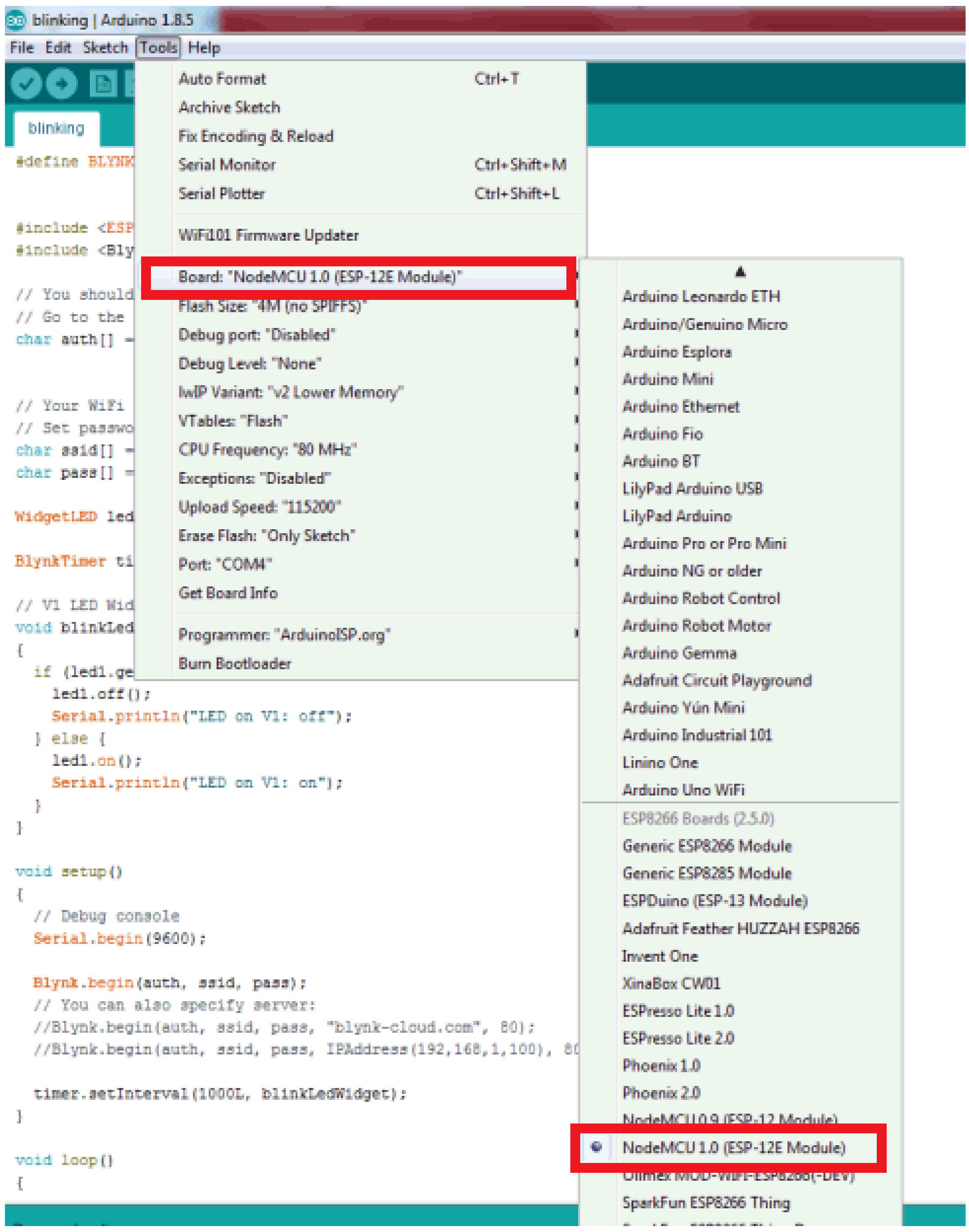
In your browser search Blynk code generator then open Blynk example browser, Select your board and example code, as shown below:


https://examples.blynk.cc/?board=NodeMCU&shield=ESP8266%20WiFi&example=Widgets%2FLED%2FLED_Blink
Apps
Blynk
Full list of supported hardware is here

Board: NodeMCU
Connection: ESP8266 WiFi
Auth Token (optional):
Example: LED Blink

```
/*************************************************************
  Download latest Blynk library here:
    https://github.com/blynkkk/blynk-library/releases/latest

  Blynk is a platform with iOS and Android apps to control
  Arduino, Raspberry Pi and the likes over the Internet.
  You can easily build graphic interfaces for all your
  projects by simply dragging and dropping widgets.

    Downloads, docs, tutorials: http://www.blynk.cc
    Sketch generator:           http://examples.blynk.cc
    Blynk community:            http://community.blynk.cc
    Follow us:                  http://www.fb.com/blynkapp
                                http://twitter.com/blynk_app

  Blynk library is licensed under MIT license
  This example code is in public domain.

 *************************************************************

  Blynk using a LED widget on your phone!

  App project setup:
    LED widget on V1
 *************************************************************/

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial


#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

WidgetLED led1(V1);
```

Open Arduino IDE and select Tools option as shown below:



Board: "NodeMCU 1.0 (ESP-12E Module)"
Flash Size: "4M (no SPIFFS)"
Debug port: "Disabled"
Debug Level: "None"
IwIP Variant: "v2 Lower Memory"
VTables: "Flash"
CPU Frequency: "80 MHz"
Exceptions: "Disabled"
Upload Speed: "115200"
Erase Flash: "Only Sketch"
Port: "COM4"
Programmer: "ArduinoISP.org"

```
#define BLYNK


#include <ESP
#include <Bly

// You should
// Go to the
char auth[] =


// Your WiFi
// Set passwo
char ssid[] =
char pass[] =

WidgetLED led

BlynkTimer ti

// V1 LED Wid
void blinkLed
{
  if (led1.ge
    led1.off();
    Serial.println("LED on V1: off");
  } else {
    led1.on();
    Serial.println("LED on V1: on");
  }
}

void setup()
{
  // Debug console
  Serial.begin(9600);

  Blynk.begin(auth, ssid, pass);
  // You can also specify server:
  //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
  //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 80

  timer.setInterval(1000L, blinkLedWidget);
}

void loop()
{
```

Note: You should have ESP8266 library installed in Arduino IDE.

Then program in Arduino IDE.

PROGRAM

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[ ] = "YourAuthToken";
// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[ ] = "YourNetworkName";
char pass[ ] = "YourPassword";
WidgetLED led1(V1);
BlynkTimer timer;
// V1 LED Widget is blinking
void blinkLedWidget()  // function for switching off and on LED
{
  if (led1.getValue()) {
    led1.off();
    Serial.println("LED on V1: off");
  } else {
    led1.on();
    Serial.println("LED on V1: on");
  }
}
void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, blinkLedWidget);
}
//In the loop function include Blynk.run() command.
void loop()
{
  Blynk.run();
  timer.run();
}
```

RESULT:

Thus, IoT using ESP8266 and Blynk App dashboard designed successfully and controlled the LED through Blynk app .

EXP NO: 5. A	MONITORING OF TEMPERATURE USING SENSOR AND ARDUINO UNO
DATE	

AIM:

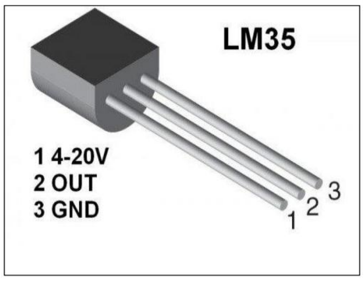
- To interface a temperature sensor with a Arduino Uno and To acquire, process, and display temperature data in real time.

Apparatus/Software Required

- Temperature sensor (e.g., LM35, thermistor, or digital sensor like DS18B20)
- Arduino Uno
- Breadboard and connecting wires
- Resistors (if using thermistor)
- (Optional) LCD module or 7-segment display for direct output

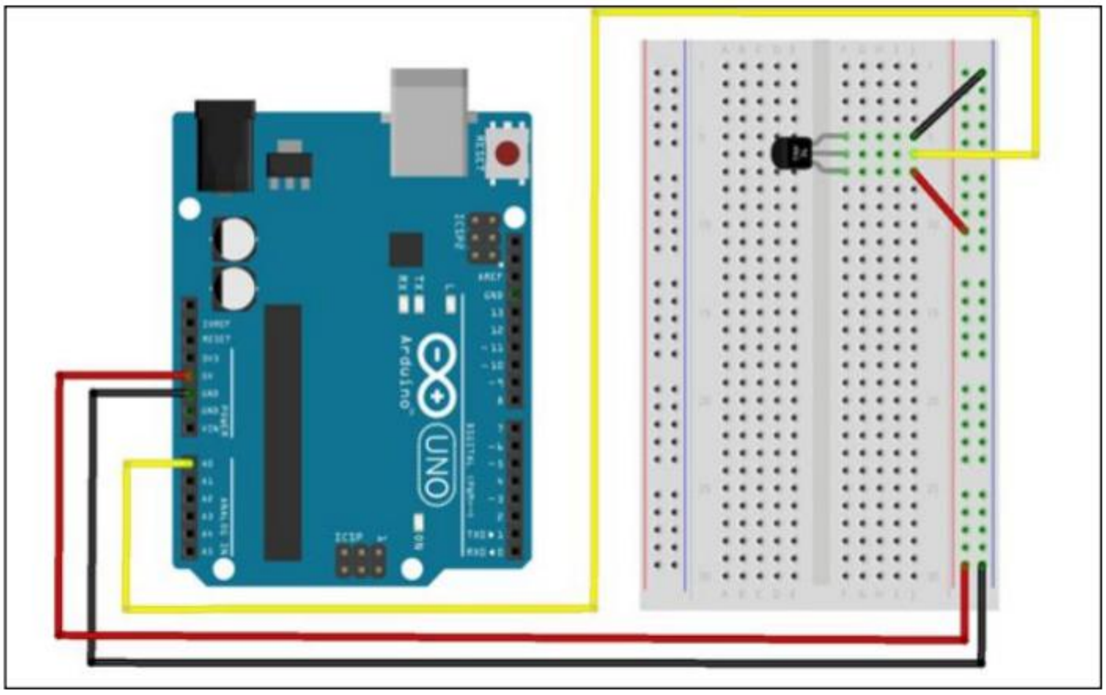
Theory

- **Temperature Sensor:**
  Converts temperature into an analog (voltage/resistance) or digital signal. Common types include:
  - LM35: Analog voltage proportional to temperature
  - Thermistor: Resistance changes with temperature
  - DS18B20: Digital output



Pin detail of LM 35

Connection diagram:



Procedure

Step 1: Sensor Circuit Setup

- For LM35 or thermistor:
    1. Connect the sensor according to its datasheet (e.g., LM35: VCC to 5V, GND to ground, output to analog input pin of FPGA board).
    2. For thermistor, use a voltage divider circuit and connect the divider output to an analog input.
- For digital sensor (e.g., DS18B20):
    1. Connect data, power, and ground lines as per datasheet.
    2. Use a pull-up resistor on the data line if required.

Step 2: Real-Time Data Acquisition and Monitoring

1. Power up the Arduino board and sensor circuit.
2. Open the Arduino IDE and program it.
3. Observe real-time temperature readings as processed by the softcore processor.
4. Optionally, display the temperature on an attached LCD or 7-segment display.

Step 3: Calibration and Validation (Optional)

1. Compare sensor readings with a reference thermometer at known temperatures.
2. Adjust the conversion formula or calibration constants in your code as needed.

Program : Implementing and monitoring the LM35 temperature Sensor:

```
 #include<LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 4, 5, 6, 7);
const int sensor=A1; // Assigning analog pin A1 to variable 'sensor'
float tempc;  //variable to store temperature in degree Celsius
float tempf;  //variable to store temperature in Fahreinheit
float vout;  //temporary variable to hold sensor reading
void setup()
{
pinMode(sensor,INPUT); // Configuring pin A1 as input
Serial.begin(9600);
lcd.begin(16,2);
delay(500);
}
void loop()
{
vout=analogRead(sensor);
Serial.print(vout)
vout=(vout*500)/1023;
tempc=vout; // Storing value in Degree Celsius
Serial.print(tempc)
tempf=(vout*1.8)+32; // Converting to Fahrenheit
lcd.setCursor(0,0);
lcd.print("in DegreeC= ");
lcd.print(tempc);
lcd.setCursor(0,1);
lcd.print("in Fahrenheit=");
lcd.print(tempf);
delay(1000); //Delay of 1 second for ease of viewing in serial monitor
        }
```

Explanation for the code:

Analog Read (sensor Pin);: Reads the analog voltage from the LM35 sensor connected to the sensor Pin.

float temperature C = (sensor Value / 1023.0) * 500.0;: Converts the analog voltage to temperature in Celsius using the formula provided in the LM35 datasheet. The value 1023.0 represents the Maximum value of the analog reading (corresponding to the maximum voltage), and 500.0 represents the temperature scaling factor for LM35.

Sample Observation Table

S.No.	**Reference Temp (°C)**	**Sensor Output (V or Digital)**	**Calculated Temp (°C)**

Result

- Successfully interfaced a temperature sensor with a Arduino Uno, Acquired and displayed real-time temperature data.

EXP NO: 5. B	**MONITORING OF TEMPERATURE USING SENSOR AND SOFTCORE PROCESSOR**
DATE	

AIM:

- To interface a temperature sensor with a softcore processor implemented on an FPGA platform and To acquire, process, and display temperature data in real time using the softcore processor.

Apparatus/Software Required

- FPGA development board (e.g., Intel/Altera DE0-Nano or Xilinx Spartan with Nios II, MicroBlaze, or AVR8 softcore processor)
- Temperature sensor (e.g., LM35, thermistor, or digital sensor like DS18B20)
- Breadboard and connecting wires
- Resistors (if using thermistor)
- PC with FPGA development tools:
    - Intel Quartus (for Nios II)
    - Xilinx Vivado (for MicroBlaze)
    - Arduino IDE or PlatformIO (if using AVR8 softcore)
- JTAG/USB programmer for FPGA
- Serial terminal software (e.g., PuTTY, Tera Term)
- (Optional) LCD module or 7-segment display for direct output

Theory

- **Softcore Processor:**
  A processor core (such as Nios II, MicroBlaze, or AVR8) implemented in the programmable logic of an FPGA. It can be customized and reconfigured, unlike fixed hardware microcontrollers.
- **Temperature Sensor:**
  Converts temperature into an analog (voltage/resistance) or digital signal. Common types include:
    - LM35: Analog voltage proportional to temperature
    - Thermistor: Resistance changes with temperature
    - DS18B20: Digital output

Procedure

Step 1: Sensor Circuit Setup

- For LM35 or thermistor:
    1. Connect the sensor according to its datasheet (e.g., LM35: VCC to 5V, GND to ground, output to analog input pin of FPGA board).
    2. For thermistor, use a voltage divider circuit and connect the divider output to an analog input.
- For digital sensor (e.g., DS18B20):
    1. Connect data, power, and ground lines as per datasheet.
    2. Use a pull-up resistor on the data line if required.

Step 2: Softcore Processor and FPGA Configuration

1. In the FPGA design tool (Quartus/Vivado), instantiate your softcore processor (Nios II, MicroBlaze, or AVR8).
2. Add required peripherals:
    - GPIO or ADC interface for sensor input
    - UART or display interface for output
3. Generate the hardware bitstream and program the FPGA.

Step 3: Software Development

1. Write embedded C/C++ code for the softcore processor to:
    - Initialize peripherals (ADC, UART, etc.)
    - Read the sensor value (analog or digital)
    - Convert raw sensor data to temperature using the sensor's transfer function or calibration curve
    - Output the temperature value to a serial terminal or display
2. Compile and download the program to the softcore processor via the FPGA toolchain.

Step 4: Real-Time Data Acquisition and Monitoring

5. Power up the FPGA board and sensor circuit.
6. Open the serial terminal on your PC and connect to the FPGA’s UART port.
7. Observe real-time temperature readings as processed by the softcore processor.
8. Optionally, display the temperature on an attached LCD or 7-segment display.

Step 5: Calibration and Validation (Optional)

3. Compare sensor readings with a reference thermometer at known temperatures.
4. Adjust the conversion formula or calibration constants in your code as needed.

Step 6: Data Logging or IoT Extension (Optional)

- Modify the code to log temperature data with timestamps or transmit data wirelessly (e.g., via Wi-Fi or Bluetooth module connected to FPGA).

Observation Table

S.No.	**Reference Temp (°C)**	**Sensor Output (V or Digital)**	**Calculated Temp (°C)**

Result

- Successfully interfaced a temperature sensor with a softcore processor on an FPGA.
- Acquired and displayed real-time temperature data using embedded software.

EXP NO: 6	**STUDY OF MICROCONTROLLERS USED IN IOT**
DATE	

AIM:

- To study the features, role and selection of microcontrollers in IoT systems and study working of sensor interfacing .

Apparatus / Software Required

- Arduino Uno or Mega board
- ESP8266 or ESP32 Wi-Fi microcontroller board
- STM32 Nucleo or Discovery board
- Raspberry Pi (for comparison)
- USB cables, breadboard, jumper wires, LEDs, sensors (e.g., DHT11/22)
- Computer with Arduino IDE, STM32CubeIDE, Thonny (for Raspberry Pi Pico), or other relevant IDEs

Theory

Microcontroller

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. It includes a processor core, memory, and programmable input/output peripherals.

Microcontrollers for IoT

Microcontrollers are ideal for IoT because they are small, low-power, and can interface with sensors, actuators, and communication modules to collect, process, and transmit data.

Common Microcontrollers in IoT:

- Arduino Series: Widely used for prototyping and education due to ease of use and community support.
- ESP8266/ESP32: Popular for built-in Wi-Fi/Bluetooth and low power consumption, ideal for wireless IoT devices.
- STM32 Series: High performance, low power, and rich connectivity options, used in industrial and commercial IoT.
- Raspberry Pi: A single-board computer with microcontroller capabilities, suitable for complex IoT systems and edge computing.
- ATtiny: Affordable and compact, good for simple IoT tasks.
- Texas Instruments MSP430/Tiva C: Known for low power and high integration, used in sensing and control.

- Infineon XMC/PSoC: Designed for IoT with a focus on energy efficiency and connectivity.

Procedure

Literature and Datasheet Study

1. List specifications (CPU speed, memory, power consumption, communication interfaces) for Arduino, ESP8266/ESP32, STM32, and Raspberry Pi using datasheets and online resources.
2. Compare features such as wireless connectivity, analog/digital I/O, and suitability for different IoT applications.

Hands-On Exploration

1. Setup the Development Environment:
    - Install Arduino IDE for Arduino/ESP boards.
    - Install STM32CubeIDE for STM32 boards.
    - Install Thonny or Raspberry Pi OS for Raspberry Pi.
2. Connect a simple sensor (e.g., DHT11/22 temperature and humidity sensor) to each board.
3. Write and upload a basic program to read sensor data and display it via serial monitor or onboard LEDs.
    - For ESP8266/ESP32, optionally send data to a cloud service or local web server.
4. Observe and record the ease of programming, speed of response, and power consumption (if possible).

Discussion and Comparison

1. Discuss the strengths and weaknesses of each microcontroller in terms of:
    - Ease of use
    - Connectivity (Wi-Fi/Bluetooth)
    - Power efficiency
    - Community and library support
    - Suitability for various IoT applications (smart home, health, industrial, etc.)
2. Summarize findings in a comparison table.

Observation Table

Board/MCU	CPU Speed	Memory	Connectivity	Power Use	Ease of Use	Typical IoT Use
Arduino Uno	16 MHz	2 KB	None	Low	Very easy	Prototyping, education
ESP8266	80 MHz	80 KB	Wi-Fi	Low	Easy	Smart home, wireless node
ESP32	240 MHz	520 KB	Wi-Fi/BT	Low	Easy	Wearables, automation
STM32F103	72 MHz	20 KB	UART/SPI/I2C	Low	Moderate	Industrial, medical
Raspberry Pi	700 MHz+	512 MB	Ethernet/WiFi	High	Moderate	Edge computing, gateway

Program to interface IR Sensor:

```
const int ProxSensor=A0;
int inputVal = 0;
void setup()
{
pinMode(7,OUTPUT);
// Pin 13 has an LED connected on most Arduino boards:
pinMode(ProxSensor,INPUT);//Pin A0 is connected to the output of proximity sensor
Serial.begin(9600);
}
```

```
void loop()
{
if(digitalRead(ProxSensor)==HIGH)
{
  //Check the sensor output
digitalWrite(7,HIGH);   // set the LED on
}
else
{
digitalWrite(7,LOW);    // set the LED off
}
inputVal = analogRead(ProxSensor);
Serial.println(inputVal);
delay(1000);
}
```

Result

Studied and compared the features of various microcontrollers used in IoT and sensor interfacing is implemented.

EXP NO: 7	LOGGING OF DATA IN CLOUD USING IOT
DATE	

AIM:

- To interface a temperature sensor with a Node MCU and log the data in Thinkspeak Cloud.

Apparatus/Software Required

Hardware

NodeMCU (ESP8266-based)

LM35 Temperature Sensor

Breadboard and jumper wires

Micro-USB cable

Software

Arduino IDE with ESP8266 board support

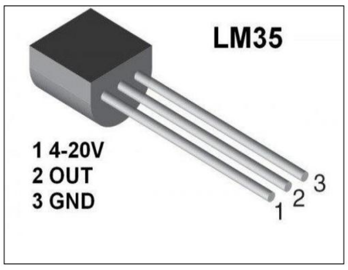
ThingSpeak account (free)

Libraries: ESP8266WiFi.h, ThingSpeak.h

Theory

Temperature Sensor:

Converts temperature into an analog (voltage/resistance) or digital signal. Common types include: LM35: Analog voltage proportional to temperature



Pin detail of LM 35

ThingSpeak API

ThingSpeak is a very good platform for IoT based projects. By using channels and web pages provided by ThingSpeak we can monitor any data over the Internet from anywhere and we can also control our system over the internet. ThingSpeak 'Collects' data from sensors, 'Analyze and Visualize' data and 'Acts' by triggering a reaction.

HARDARE/ SOFWARE SETUP

Circuit Connections

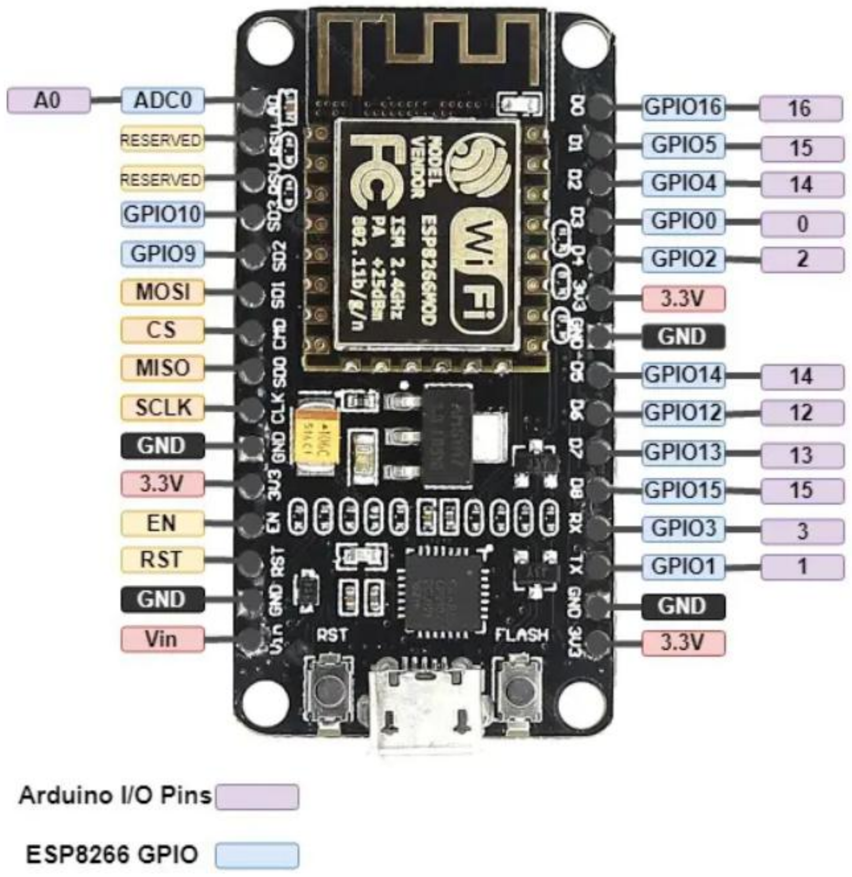
LM35 to NodeMCU

Pin 1 (VCC): Connect to NodeMCU’s 3.3V pin

Pin 2 (Output): Connect to NodeMCU’s A0 (Analog Input)

Pin 3 (GND): Connect to NodeMCU’s GND

NODE MCU PIN DETAILS



ThingSpeak Setup

Create a Channel

Log in to ThingSpeak → Channels → My Channels → New Channel

Name the channel (e.g., “Temperature Monitor”) and enable Field 1 (label: “Temperature”) .

Save and note the Write API Key .

Arduino IDE Configuration

Install Required Libraries

In Arduino IDE: Sketch → Include Library → Manage Libraries → Install ThingSpeak and

ESP8266WiFi

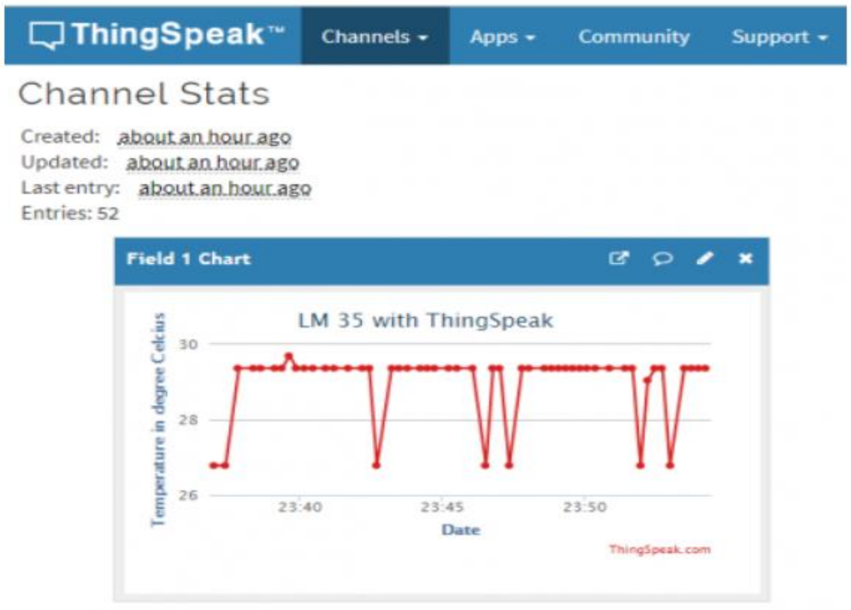
Procedure

1. First of all, user needs to create an account on ThingSpeak.com, then Sign In and click on ‘Get Started’.
2. Now go to the ‘Channels’ menu and click on the ‘New Channel’ option in the same page.
3. Now you will see a form for creating the channel, fill the Name and Description as per your choice. Then fill ‘Temperature’ in ‘Field 1’ field. Tick the check box ‘Make Public’ option below the form and finally Save the channel. Now your new channel is ready.
4. Now click on ‘API keys’ tab and note the Write and Read API key, here we are only using Write key. You need to copy and paste this key in the code (see below).
5. Now user need to upload the program to ESP8266 using Arduino IDE.
6. After uploading, open “PRIVATE VIEW” icon in ThingSpeak website and observe the monitored temperature value on graph as shown below.

Program :

```
#include "ThingSpeak.h"
#include <ESP8266WiFi.h>
const int LM35 = A0;
//---------- Enter you Wi-Fi Details---------//
char ssid[] = "vivo 1935"; //SSID
char pass[] = "Vivo 1935"; // Password
//-------------------------------------------//
WiFiClient  client;
unsigned long myChannelNumber = 1843008; // Channel ID here
const int FieldNumber = 1;
const char * myWriteAPIKey = "R8NVVUA948RBM0MQ"; // Your Write API Key here
void setup()
{
Serial.begin(115200);
WiFi.mode(WIFI_STA);
ThingSpeak.begin(client);
}
void loop()
{
if (WiFi.status() != WL_CONNECTED)
{
Serial.print("Attempting to connect to SSID: ");
Serial.println(ssid);
while (WiFi.status() != WL_CONNECTED)
{
WiFi.begin(ssid, pass);
Serial.print(".");
delay(5000);
}
Serial.println("\nConnected.");
}
int ADC;
float temp;
ADC = analogRead(LM35);  /* Read Temperature */
temp = (ADC * 3); /* Convert adc value to equivalent voltage */
temp = (temp / 10); /* LM35 gives output of 10mv/°C */
Serial.print("Temperature = ");
Serial.print(temp);
Serial.println(" *C");
delay(1000);
ThingSpeak.writeField(myChannelNumber, FieldNumber, temp, myWriteAPIKey);
delay(5000);
}
```

SAMPLE OUTPUT IN THINKSPEAK CLOUD



Channel Stats

Created: about an hour ago
Updated: about an hour ago
Last entry: about an hour ago
Entries: 52

IoT Temperature Data Logger Using ESP8266 and LM35 – ThingSpeak Graph

Observation Table

S.No.		

Result

- Successfully interfaced a temperature sensor with a Node MCU and Thinkspeak cloud.

EXP NO: 8.A	WAP(Wireless Application Protocol) FOR LED BLINK
DATE	

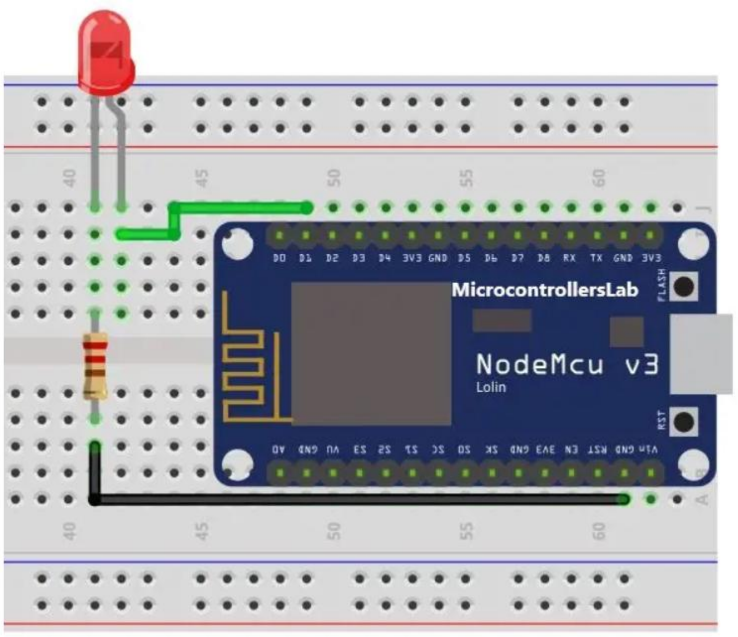
AIM:
To control (blink) an LED wirelessly using NodeMCU by implementing a simple web server (WAP-based interface) that allows the user to turn the LED ON or OFF from a browser on the same Wi-Fi network.

Materials Required

- NodeMCU (ESP8266)
- LED (any color)
- 220Ω resistor
- Breadboard and jumper wires
- Micro-USB cable
- Computer or smartphone with Wi-Fi capability
- Arduino IDE

Circuit Diagram

- **LED Anode (+):** Connect to NodeMCU D4 (GPIO2) via 220Ω resistor.
- **LED Cathode (–):** Connect to NodeMCU GND.



Procedure

1. **Hardware Setup:**
    - Assemble the circuit as described above.
    - Connect NodeMCU to your computer via USB.
2. **Software Setup:**
    - Open Arduino IDE.
    - Install the ESP8266 board package (if not already installed).
    - Select NodeMCU 1.0 (ESP-12E Module) and the correct COM port.
3. **Program NodeMCU as a WAP Web Server:**
    - The NodeMCU will create a simple web page accessible via Wi-Fi. Users can turn the LED ON or OFF by clicking buttons on the web page.

WAP-Based Web Server Code

```
#include <ESP8266WiFi.h>
// Replace with your network credentials
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
WiFiServer server(80);
#define LED_PIN D4



void setup() {
  Serial.begin(115200);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (!client) return;

  while (!client.available()) {
    delay(1);
  }

  String request = client.readStringUntil('\r');
  client.flush();

  // Parse request and control LED
  if (request.indexOf("/LED=ON") != -1) {
    digitalWrite(LED_PIN, HIGH);
  }
  if (request.indexOf("/LED=OFF") != -1) {
    digitalWrite(LED_PIN, LOW);
  }
  // HTML response
  String html = "<!DOCTYPE html><html><head><title>LED Control</title></head><body>";
  html += "<h1>ESP8266 LED Control (WAP)</h1>";
  html += "<p><a href=\"/LED=ON\"><button>ON</button></a></p>";
  html += "<p><a href=\"/LED=OFF\"><button>OFF</button></a></p>";
  html += "</body></html>";

  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println("Connection: close");
  client.println();
  client.println(html);
  delay(1);
}
```

Testing and Observations

1. Upload the code to NodeMCU.
2. Open Serial Monitor to find the IP address assigned to NodeMCU (e.g., 192.168.1.100).
3. On any device connected to the same Wi-Fi, open a browser and enter the IP address.
4. The web page will display ON and OFF buttons. Click to control the LED wirelessly.

Result

- The LED can be turned ON or OFF from any device on the same Wi-Fi network using a WAP-based web interface hosted by NodeMCU.

EXP NO: 8.B	WAP(Wireless Application Protocol) FOR LED BLINK USING BLYNK APP
DATE	

AIM:

To control (blink) an LED connected to a NodeMCU (ESP8266) using the Blynk app, enabling wireless control via a smartphone or web dashboard over Wi-Fi.

Materials Required

NodeMCU (ESP8266) development board

LED (any color)

220Ω resistor

Breadboard and jumper wires

Micro-USB cable

Smartphone with Blynk app installed

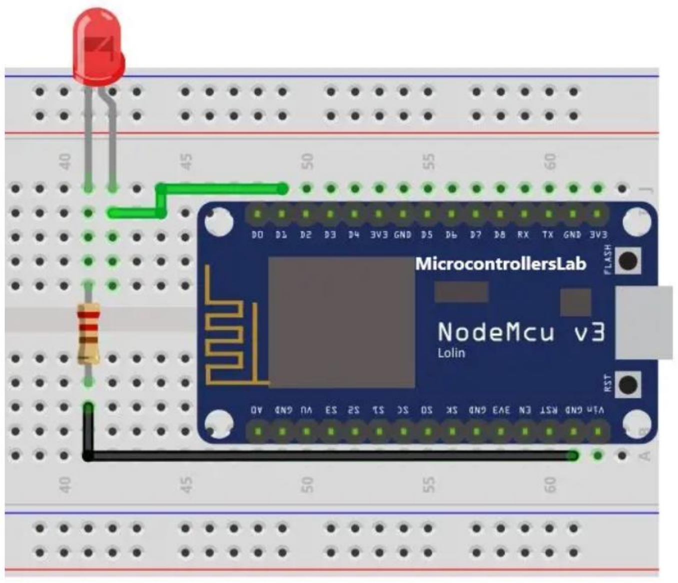
Arduino IDE with ESP8266 and Blynk libraries

Wi-Fi with internet access

Circuit Diagram

LED Anode (+): Connect to NodeMCU D1 (GPIO5) via a 220Ω resistor

LED Cathode (–): Connect to NodeMCU GND



Procedure

1. Hardware Setup

Assemble the circuit as described above, ensuring correct polarity for the LED and connection to D1 (GPIO5).

2. Blynk App Setup

Download and install the Blynk app from Google Play or Apple App Store.

Create a new project:

Name your project (e.g., "LED Blink").

Select device: NodeMCU or ESP8266.

Connection type: Wi-Fi.

Blynk will send an Auth Token to your email—save this for your code.

Add a Button widget to the project dashboard.

Tap the button, set the pin to D1 (or the GPIO you used).

Set mode to "Switch" for toggle control.

Optionally, label ON/OFF states.

3. Arduino IDE Setup

Install the ESP8266 board package and Blynk library in Arduino IDE.

Select the correct board (NodeMCU 1.0) and port.

5. **Programming the NodeMCU**

```
#define BLYNK_TEMPLATE_ID "YourTemplateID"

#define BLYNK_DEVICE_NAME "YourDeviceName"

#define BLYNK_AUTH_TOKEN "YourAuthToken"

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>

char auth[] = "YourAuthToken";          // Paste Auth Token from Blynk email

char ssid[] = "YourWiFiSSID";           // Enter your Wi-Fi SSID

char pass[] = "YourWiFiPassword";       // Enter your Wi-Fi password
```

```
void setup() {
  Blynk.begin(auth, ssid, pass);
  pinMode(D1, OUTPUT);                  // Set LED pin as output
}
void loop() {
  Blynk.run();
    }
```

- **Replace "YourAuthToken", "YourWiFiSSID", and "YourWiFiPassword" with your actual credentials.**

5. Upload and Test

- Upload the code to NodeMCU via Arduino IDE.
- Open the Blynk app, press the play (run) button to connect5.
- Tap the button widget to turn the LED ON or OFF wirelessly

RESULT:

Successfully studied how to integrate NodeMCU with the Blynk IoT platform for wireless device control.

EXP NO: 9	**DESIGN OF IOT SYSTEM**
DATE	

AIM:

To design and implement an IoT-enabled automatic door system that utilizes proximity detection to autonomously open the door.

THEORY :

Regular door opening and closing need manual physical power. It is ok when normal home doors. Where as in big size malls or industries we can't open and close manually. There are some electronic doors can be controlled by power. But those are static and can't be control from remote location. This system is designed to open the door automatically when a person comes near to the entrance of the door by sensing the distance.

DESCRIPTION:

This system includes Arduino uno receives the distance information from ultrasonic sensor and monitors continuously, when it goes below the activation distance it activates the servo motor.

Materials Required

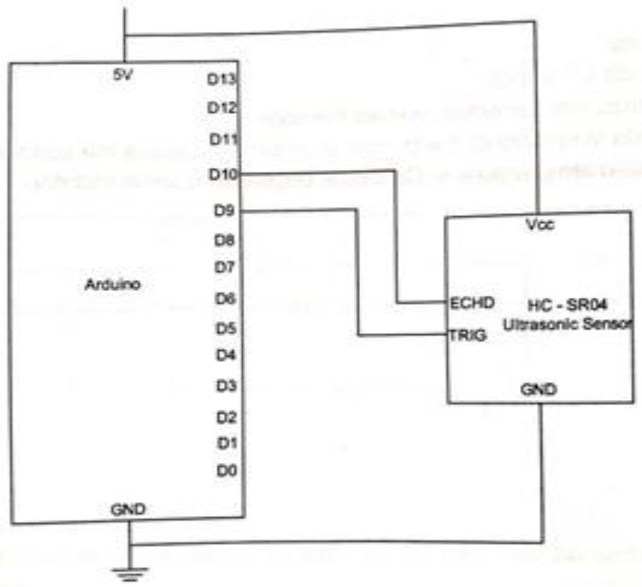
Ultrasonic sensor (HC-SR04)

Servo motor (SG90)

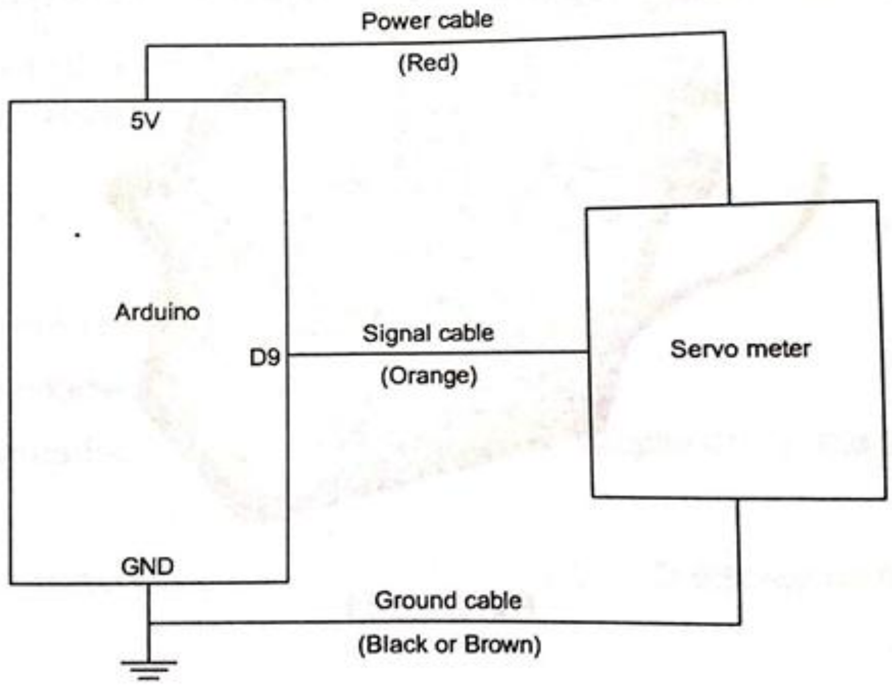
Program :

```
#include <Servo.h>
const int ActivationDistance = 10
const int trigPin = 3;
const int echoPin = 2;
long duration;
int distanceCm, distanceInch;
int servoPin = 9;
Servo servo;
int angle = 0;  // servo position in degrees
void setup()
 {
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
servo.attach(servoPin);
}
void loop()
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distanceCm = duration * 0.034 / 2;
distanceInch = duration * 0.0133 / 2;
delay(10);
if (distanceCm <= ActivationDistance)
{
  for(angle = 0; angle < 270; angle++)
{
servo.write(angle);
delay(15);
}
// now scan back from 180 to 0 degrees
for(angle = 180; angle > 0; angle--)
{
servo.write(angle);
delay(15);
}
 }
}
```

Ultrasonic Distance Measurement with Arduino



Circuit Diagram:



RESULT:

Thus an IoT-enabled automatic door system that utilizes proximity detection to autonomously open the door is designed and implemented.

ADDITIONAL EXPERIMENT BEYOND SYLLABUS

EXP NO: 10	DESIGN OF IOT SYSTEM – GAS LEAKAGE DETECTION SYSTEM
DATE	

AIM:

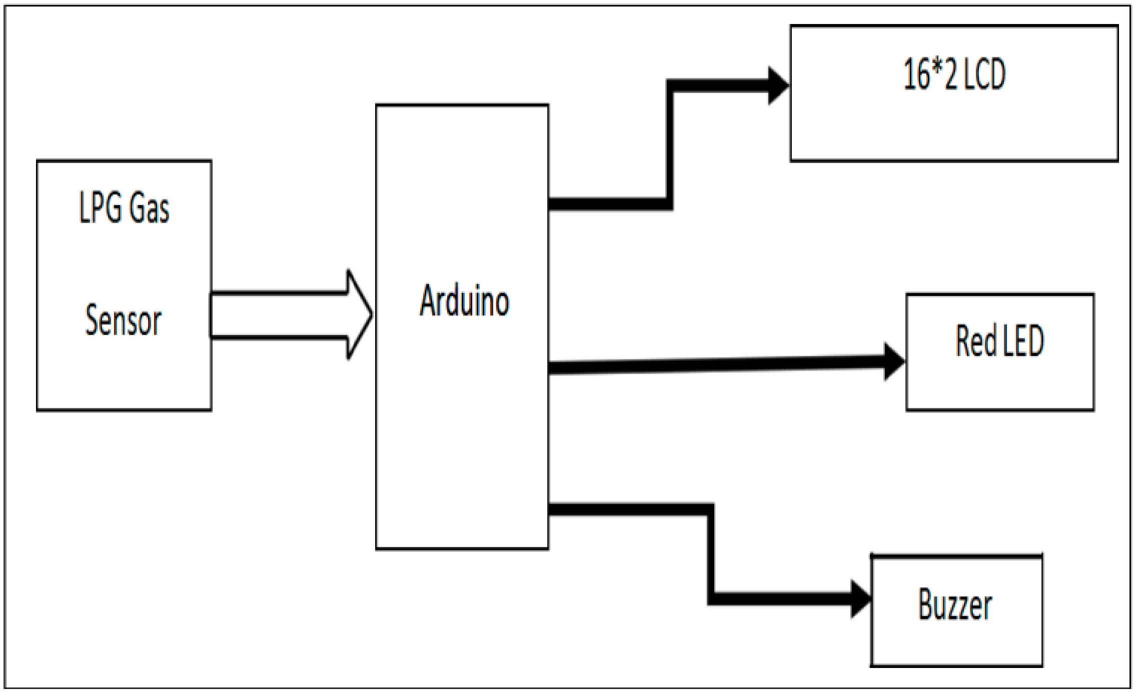
To design and implement an IoT-enabled gas leakage detection system and generates alarm.

THEORY :

Gas leakage is a serious problem and nowadays it is observed in many places like residences, industries, and vehicles like Compressed Natural Gas (CNG), buses, cars, etc. It is noticed that due to gas leakage, dangerous accidents occur. The Liquefied petroleum gas (LPG), or propane, is a flammable mixture of hydrocarbon gases used as fuel in many applications like homes, hostels, industries, automobiles, and vehicles because of its desirable properties which include high calorific value, less smoke, less soot, and meager harm to the environment. Liquid petroleum gas (LPG) is highly inflammable and can burn even at some distance from the source of leakage.

A gas leakage detector becomes vital and helps to protect people from the dangers of gas leakage There are different gas detection techniques used . This system is a low-cost advanced sensor-based gas leakage detector, alert and control system .

BLOCK DIAGRAM:



Materials Required

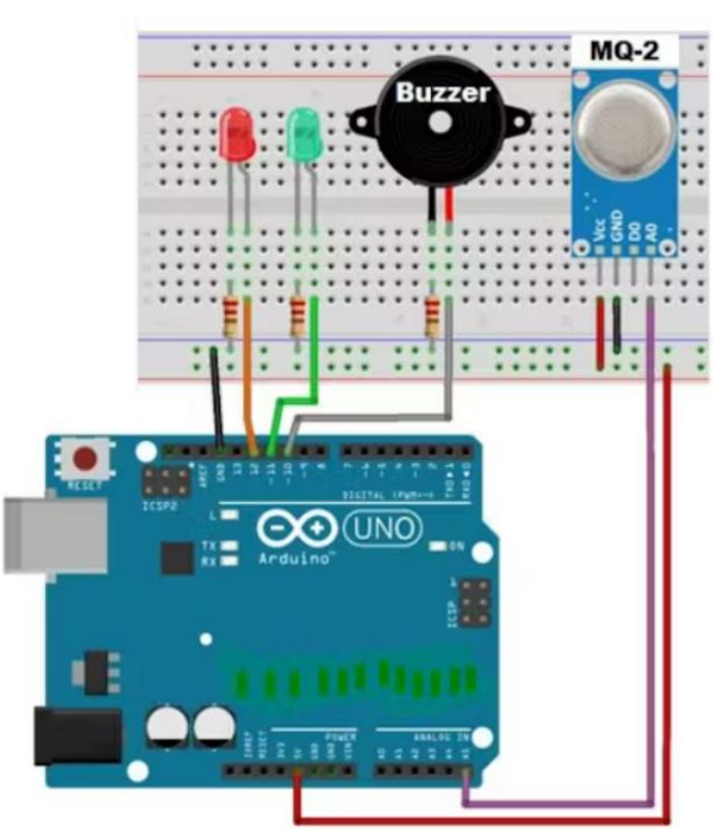
ARDUINO

MQ-6 GAS SENSOR

16*2 LCD

BUZZER

CIRCUIT DIAGRAM



Program :

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(5,6,8,9,10,11);
int redled = 2;
int greenled = 3;
int buzzer = 4;
int sensor = A0;
int sensorThresh = 400;
```

```
    void setup()
    {
    pinMode(redled, OUTPUT);
    pinMode(greenled,OUTPUT);
    pinMode(buzzer,OUTPUT);
    pinMode(sensor,INPUT);
    Serial.begin(9600);
    lcd.begin(16,2);
    }
void loop()
    {
    int analogValue = analogRead(sensor);
    Serial.print(analogValue);
    if(analogValue>sensorThresh)
    {
    digitalWrite(redled,HIGH);
    digitalWrite(greenled,LOW);
    tone(buzzer,1000,10000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("ALERT");
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("EVACUATE");
    delay(1000);
    }
    else
    {
    digitalWrite(greenled,HIGH);
digitalWrite(redled,LOW);
noTone(buzzer);
lcd.clear();
lcd.setCursor(0,0);



lcd.print("SAFE");
delay(1000);
lcd.clear();
lcd.setCursor(0,1);
lcd.print("ALL CLEAR");
delay(1000);
}
}
```

RESULT:

Thus, an IoT-enabled gas leakage detection system that utilizes LPG gas detection to raise alarm is designed and implemented.