

Week1-Day1&2: Introduction to Embedded C, Simulation and mini-hands-on tasks

Learning Objective: Understanding Embedded-C concepts, Simulation of Projects Embedded systems basics.

We started the day by explaining the differences between Microcontroller, Arduino, NodeMCU and RaspberryPi, and we were taught the different specifications of these boards. Then, we began by explaining microcontrollers and microprocessors and their respective properties.

After that, the explanation of embedded C programming, its purposes, and the use of its syntaxes were given.

Hands-On activity: LED blinking

Input: Power supply, Code

Output: consecutively Segmented LED blinking

Tools: Proteus, Keil uVision

Logic: The program is designed in such a way that consecutive adjacent LEDs were to blink for a given delay time while the adjacent LED remained off, this was interswitched between them using delay.

Code:

```
#include <reg51.h>
```

```
void delay(unsigned int);
```

```
void main(void)
```

```
{
```

```
    while(1)
```

```
    {
```

```
        P1 = 0x80;
```

```

    delay(500);
    P1 = 0x40;
    delay(500);

        P1 = 0x20;

    delay(500);
    P1 = 0x10;
    delay(500);

        P1 = 0x08;

    delay(500);
    P1 = 0x04;
    delay(500);
    P1 = 0x02;
    delay(500);

        P1 = 0x01;

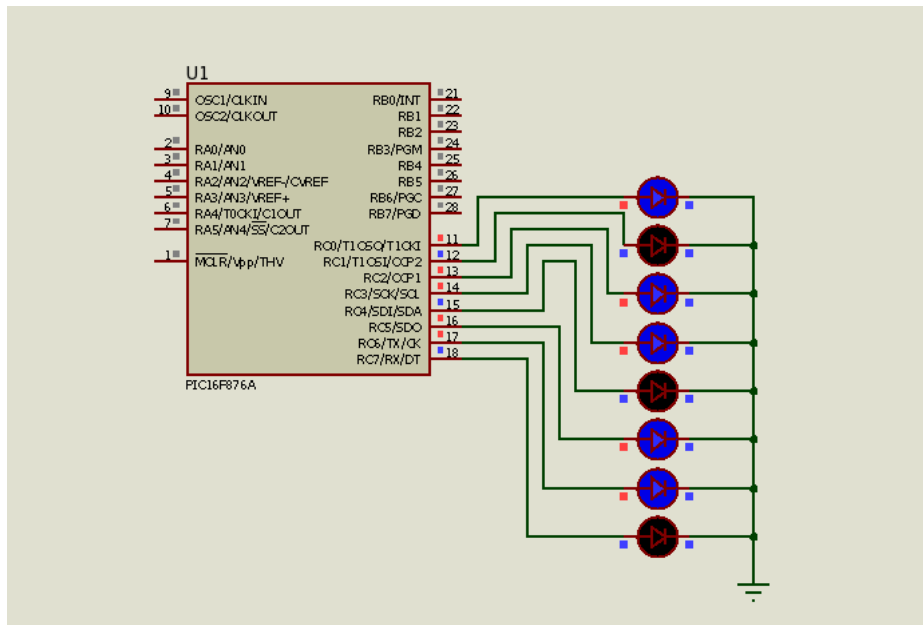
    delay(500);

        }
}

void delay(unsigned int t) // Function for delay
{
    unsigned int i, j;
    for(i = 0; i < t; i++)
        for(j = 0; j < 1275; j++);
}

```

Output:



Week1-Day3: introduction to Button interfacing with 8051

Learning Objective: Taking input from a button and respectively controlling the IC output based on the input generated.

Input: Push Buttons, Power supply, Code

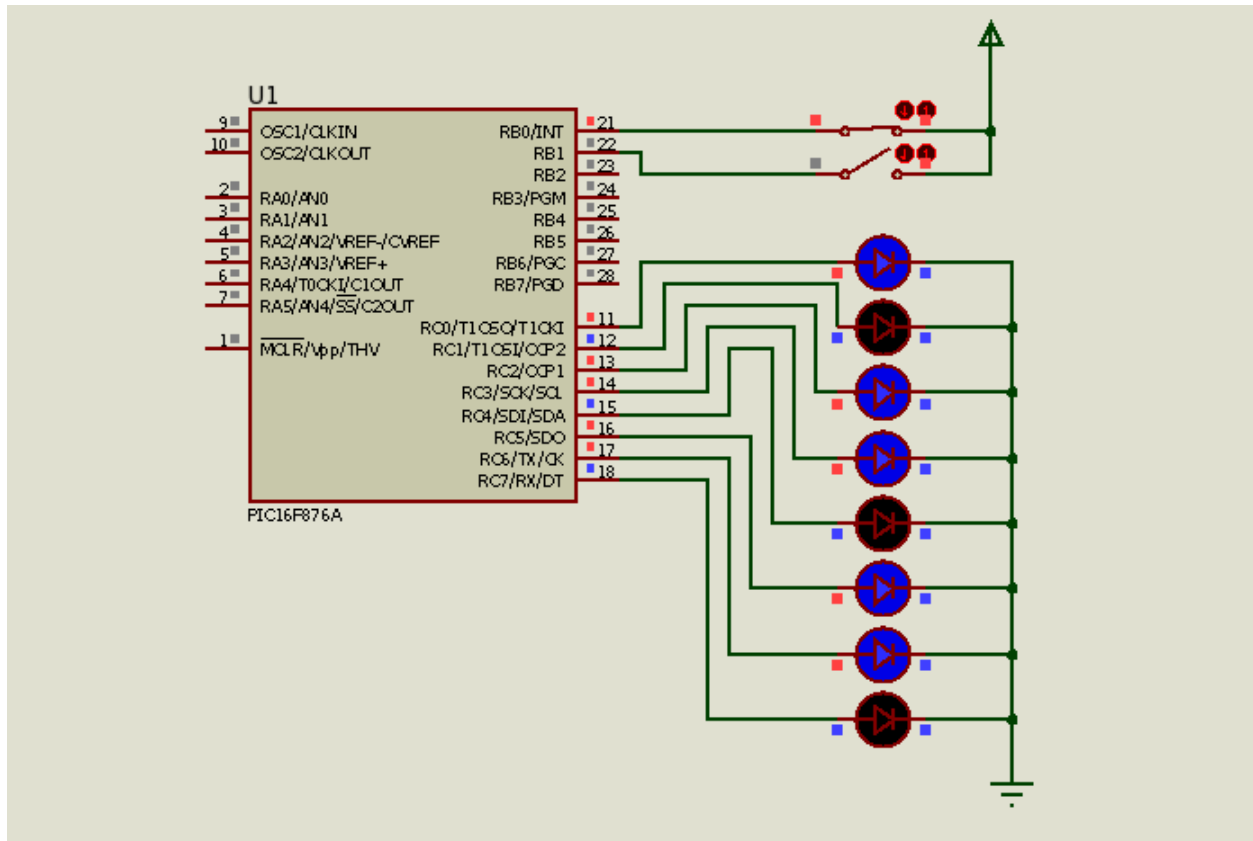
Output: LEDs

Tools: Proteus, Keil uVision

Logic: Push buttons are connected to any respective defined input pin of the microcontroller, signal has to be read from these pins through the button states, and the LEDs are controlled prior to the button states using the program.

Code

Output:



Common issues: Proteus version compatibility issues, Connection errors.

Week1-Day4: introduction to LCD interfacing with 8051

Learning Objective: basic knowledge of LCD, and interfacing it with 8051 microcontroller for informative display.

Input: Push Buttons, Power supply, Code

Output: LEDs, LCD

Tools: Proteus, Keil uVision

Logic: LCD (16 columns x 2 rows) has 16 pin for connection, with the main 2 for power(Vcc and GND), data pins(D0-D7) as per the LCD datasheet, these control and data pins has to configure in code for proper communication to the LCD to generate effective display, the code is designed to display the session of the day and the class strength.

Code: Code #include <reg51.h>

```
sbit rs=P2^0;
```

```
//sbit rw=P2^1;
```

```
sbit e=P2^2;
```

```
sbit ALE = P2^4;
```

```
sbit OE = P2^5;
```

```
sbit SC = P2^1;
```

```
sbit EOC = P2^7;
```

```
//Declaring the input selection pin
```

```
sbit ADDR_A = P2^0;
```

```
sbit ADDR_B = P2^1;
```

```
sbit ADDR_C = P2^2;
```

```
void delay(unsigned int);
```

```
void WriteCommandToLCD(unsigned char ch);
```

```
void WriteDataToLCD(unsigned char ch);
```

```
void WriteStringToLCD(unsigned char ch[]);
```

```
void main(void)
```

```
{
```

```
    //    unsigned char ch[]="ES TRAINING";
```

```
unsigned char ch1[]="GITAM UNIVERSITY, BANGALORE";
```

```
unsigned int j,k;
```

```
unsigned int MyData = 20;
```

```
//LCD Intialization
```

```
WriteCommandToLCD(0x38); //2 lines and 5×7 matrix
```

```
WriteCommandToLCD(0x01); //Clear display screen
```

```
WriteCommandToLCD(0x0E); //Display on, cursor off
```

```
WriteCommandToLCD(0x80); //Force cursor to the beginning ( 1st line)
```

```
WriteCommandToLCD(0x06); //Increment cursor (shift cursor to right)
```

```
//Sending Data to LCD
```

```
WriteStringToLCD("ES TRAINING"); //Sending the string to LCD
```

```
WriteCommandToLCD(0xc0); //Force cursor to the beginning ( 2nd line)
```

```
for(j=0;ch1[j]!='\0';j++)
```

```
{
```

```
    WriteDataToLCD(ch1[j]); // Sending one character to LCD
```

```
}
```

```
for(k=0;k<30;k++)
```

```
{
```

```
    WriteCommandToLCD(0x1c); // shift entire display right
```

```
}
```

```
while(1)
```

```
{
```

```
    WriteCommandToLCD(0x01); //Clear display screen
```

```

        WriteCommandToLCD(0x80); //Force cursor to the beginning
( 1st line)

        WriteStringToLCD("CLASS STRENGTH");//Sending String to
LCD

        WriteCommandToLCD(0xc0); //Force cursor to the beginning
( 2nd line)

        WriteDataToLCD((MyData / 10) + 48);//seperating the first
digit of MyData

        WriteDataToLCD((MyData % 10) + 48);//seperating the first
digit of MyData

        WriteStringToLCD(" STUDENTS"); //Sending String to LCD
    }
    //}
}

void delay(unsigned int t)/////Function for setting 1ms delay
{
    unsigned int i,j;
    for(i=0;i<t;i++)
        for(j=0;j<1275;j++);
}

void WriteCommandToLCD(unsigned char ch) //Function for sending Command
{
    e=1;
    rs=0;
    rw=0;

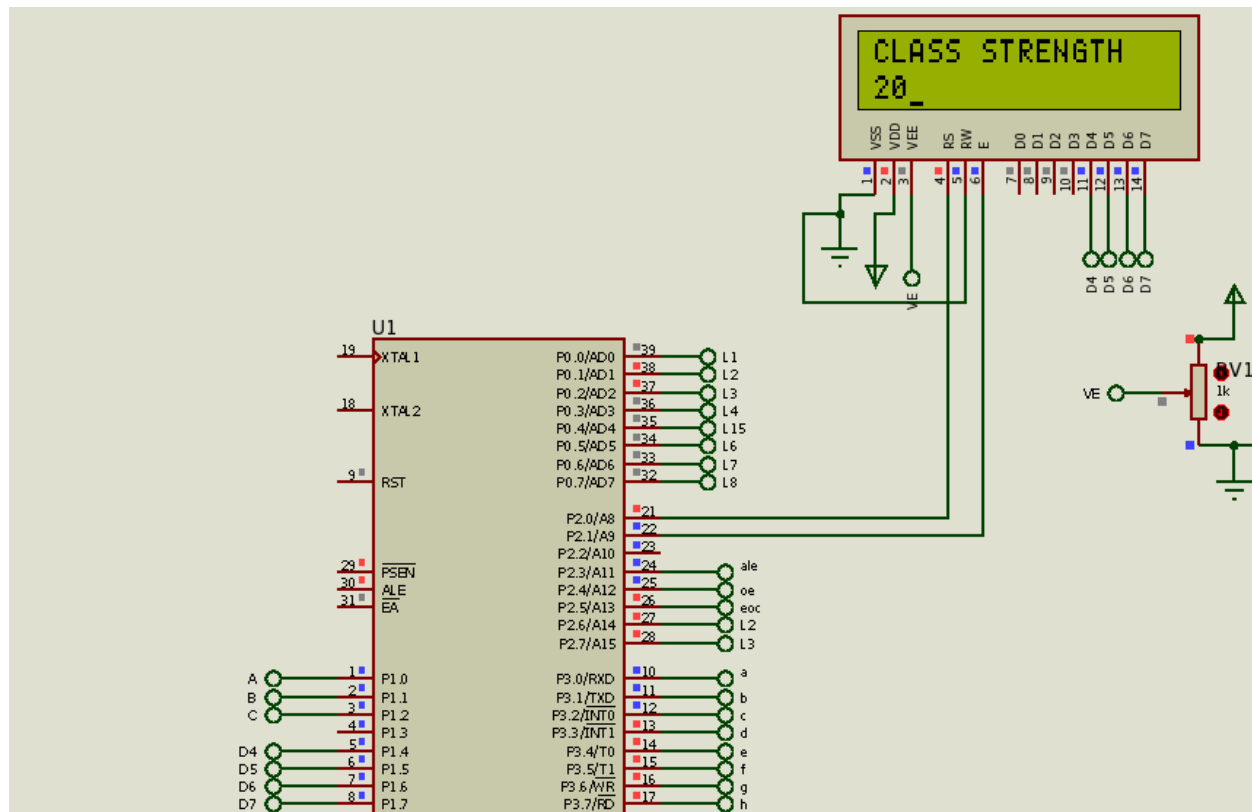
```

```

        P1= ch & 0xF0;
        e=0;
        delay(20);
    }
void WriteDataToLCD(unsigned char ch) //Function for sending Data
{
    e=1;
    rs=1;
    rw=0;
    P1=ch & 0xF0;
    e=0;
    delay(20);
}
void WriteStringToLCD(unsigned char ch[]) //Function for sending string
{
    int i;
    for(i=0;ch[i]!='\0';i++)
    {
        WriteDataToLCD(ch[i]);
    }
}

```

Output:



Week1-Day5: introduction to ADC interfacing with 8051

Learning Objective: basic knowledge of ADC, and interfacing it with 8051 microcontroller for analog input interfacing and conversion to digital output generation.

Input: Push Buttons, ADC0808 IC, Power supply, Code

Output: LEDs

Tools: Proteus, Keil uVision

Logic: ADCs convert **continuous analog signals** (like voltage variations from sensors) into **discrete digital signals** (represented by a series of 0s and 1s) that digital devices can understand.

Imagine an analog dial with infinitely many positions between minimum and maximum. An ADC takes a snapshot of the dial's position at a specific moment and translates it into a specific digital value.

Key Concepts:

- **Resolution:** This refers to the number of bits used to represent the analog signal's value. A higher resolution ADC (e.g., 12-bit) provides more precise digital values with more steps between minimum and maximum. (Think of having more positions on the dial to capture)
- **Sampling Rate:** This defines how often the ADC takes a sample of the analog signal. A faster sampling rate is necessary for accurately capturing rapidly changing signals. (Imagine taking more frequent snapshots of the dial's movement)
- **Analog Reference Voltage:** This voltage sets the upper limit of the analog signal that the ADC can convert. The digital output will range from 0 (for 0V) to a maximum value (usually corresponding to the reference voltage).

Code: #include <reg51.h>

```
sbit ALE = P2^4;
```

```
sbit OE = P2^5;
```

```
sbit SC = P2^6;
```

```
sbit EOC = P2^7;
```

```
//Declaring the input selection pin
```

```
sbit ADDR_A = P2^0;
```

```
sbit ADDR_B = P2^1;
```

```
sbit ADDR_C = P2^2;
```

```
//void MSDelay(unsigned int);
```

```
void MSDelay(unsigned int delay)
```

```
{
```

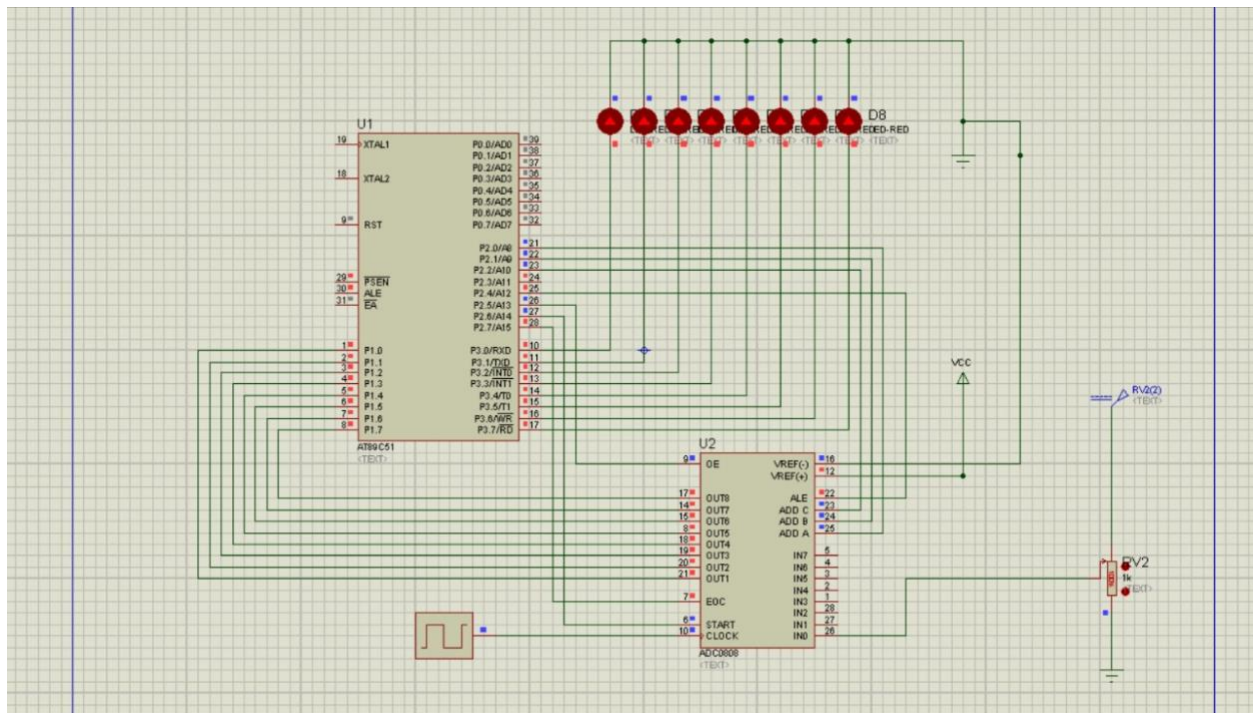
```

        unsigned int i,j;
        for(i=0;i<delay;i++)
            for(j=0;j<1275;j++);
    }
    void main()
    {
        unsigned char ADC_Value = 0;
        P1 = 0xFF;
        EOC = 1;
        ALE = 0;
        OE = 0;
        SC = 0;
        while(1)
        {
            ADDR_C = 0;
            ADDR_B = 0;
            ADDR_A = 0;
            MSDelay(10);
            ALE = 1;
            MSDelay(10);
            SC = 1;
            MSDelay(10);
            ALE = 0;
            SC = 0;
            while(EOC==1);
            //while(EOC==0);
            OE=1;
            MSDelay(10);
            ADC_Value= P1;
            P3 = ADC_Value;
            OE = 0 ;
        }
    }

```

}

Output:



DIY Task: ADC and LCD interface for Voltage Monitoring with LED indication using 8051 or PIC microcontroller

Objective: Monitoring Line Voltage Fluctuation using Microcontroller ADC input.

Input: Analog source, push Microcontroller, PIC16F876A, LEDs, LCD, Proteus, MicroC for PIC (IDE)

Output: LCD Voltage display with LED indications of Voltage levels.

Logic: >> If the voltage level is more than 3v, use two bulbs (Use LED for demonstration)

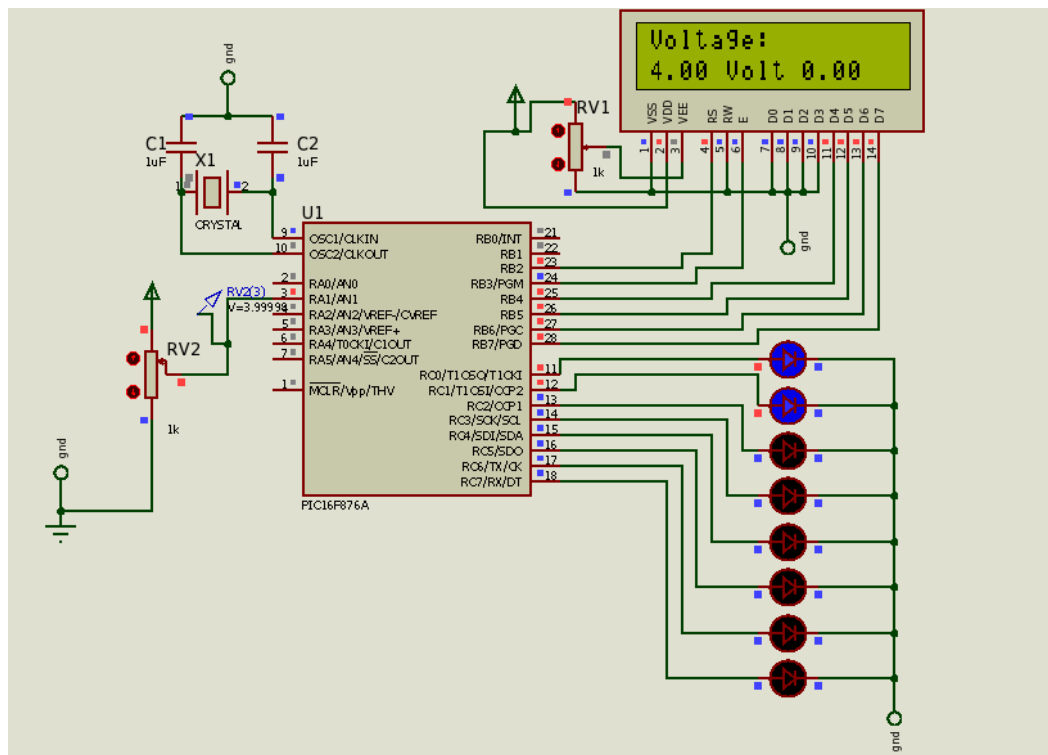
>> If the voltage level is 2-3 V, use only one bulb (Use LED for demonstration)

>> If it is less than 2 v, the system should switch off all the bulbs.

Connection: 6 pins from the PIC B-port were used for the LCD interface, the PORTC was used for the LEDs connection, Analog pin 1 was used for taking analog signal from the source, a crystal was used for the PIC clock efficiency, and all these configurations are assigned accordingly in the program.

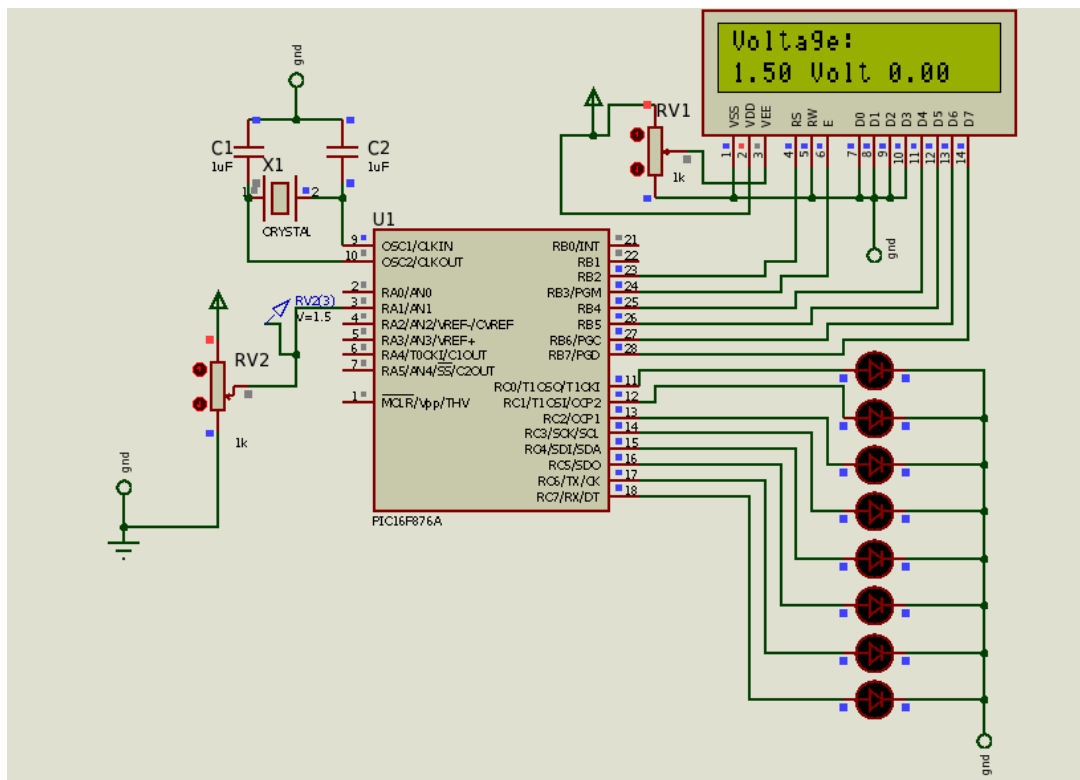
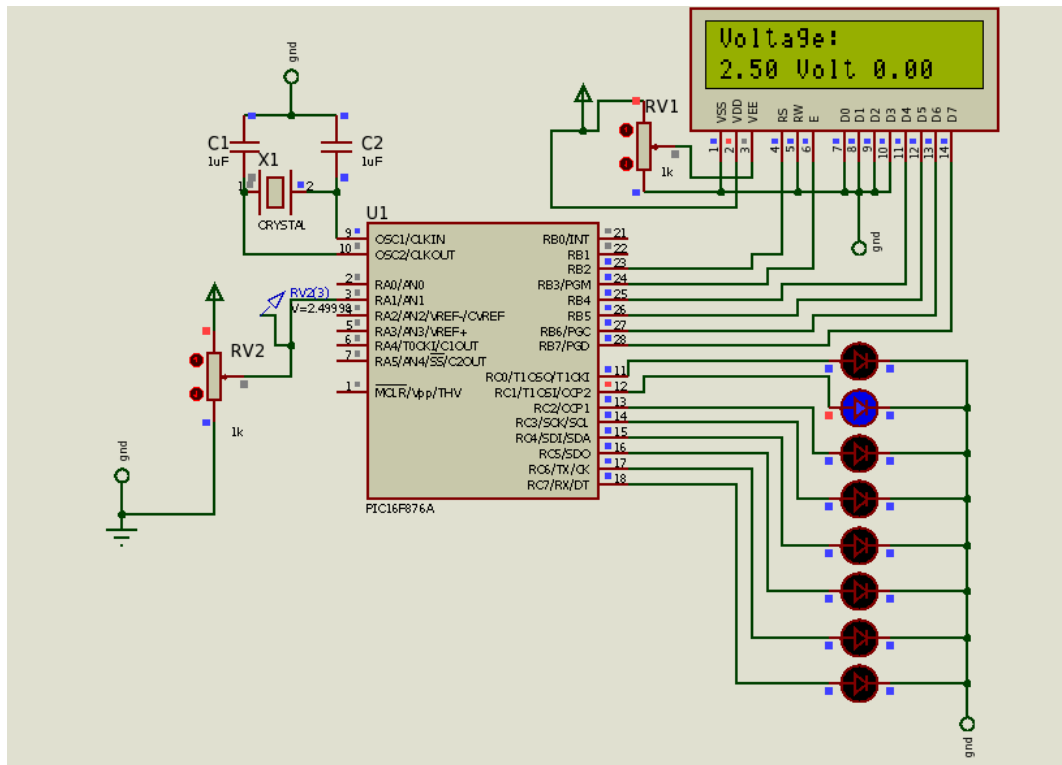
Code:

Output:



When Voltage is greater than 3V

When Voltage is between 2-3V



When Voltage is less than 2V

Week2-Day1: introduction to RaspberryPi

Learning Objectives: To gain Background knowledge and prerequisites of RaspberryPi

The trainer explained the evolution of RaspberryPi, how the ideation came into existence, the stages of its development and models, it started from Pi model A, with few features and was upgraded to Pi model B with additional features compared to model A, it was then upgraded to Pi model 2, 3, 4, and presently in Model 5, this evolution evolves its advancement in features on the track, it's a microprocessor that uses OS (operating system, previously Noobs, now PiOS) to operate modelled in Ubuntu/Linux structure/framework, presently it has a different version of board design from the Pi foundation and different educational Kit designed for hobbies and learning purposes.

Interfaces supported by the board were explained, and how they function compared to other microcontrollers or microprocessors, e.g I2C, SPI, M2HAT, etc., not neglecting its sophisticated features of visualization, connecting to external image visualizers as well as personal visuals, its onboard connectivity and ports functionalities were explained by the trainer, resources to broaden the basic knowledge and fundamentals of RaspberryPi were given and annotated.

Week2-Day2: Installation of RaspberryPi OS and Configuration

Learning Objectives: Onboarding usage configuration of RaspberryPi board

To start using the Pi board, as mentioned by the trainer and in the documentation on the process base, the OS(Ubuntu-based or Ubuntu core) has to be installed on the RaspberryPi board, which can be done in two ways:

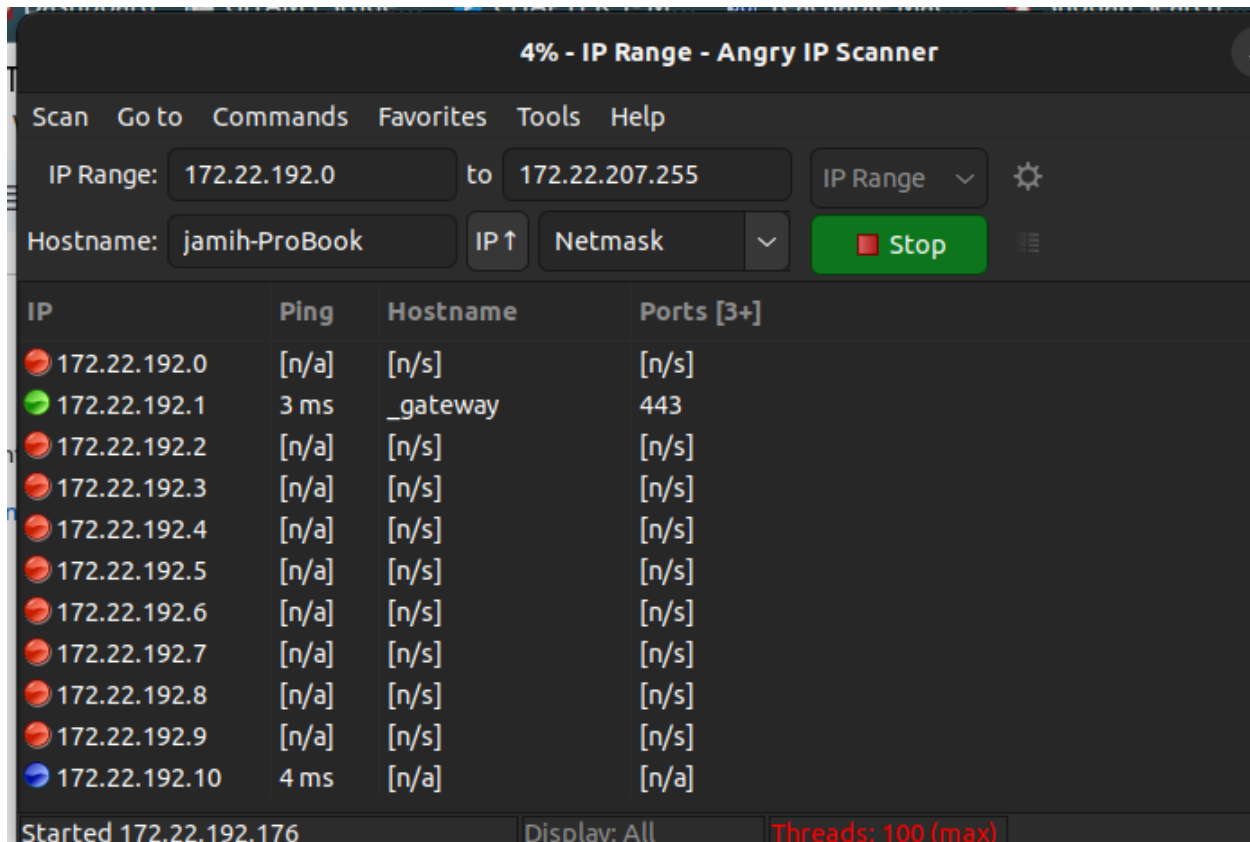
- a. Downloading it and uploading it through the Imager
- b. Using the imager to install it from the Pi foundation cloud to the mass storage with network connectivity

The steps for the installation and configuration are as follows:

- a. A memory card is to be inserted into a reader, which has to be inserted into the PC.
- b. Opening the Imager and selecting the board type, OS version and the storage to load the OS to.
- c. Performing the basic settings(device local name, device name, SSH auth, WiFi details, etc) before final approval of loading the OS to the external storage

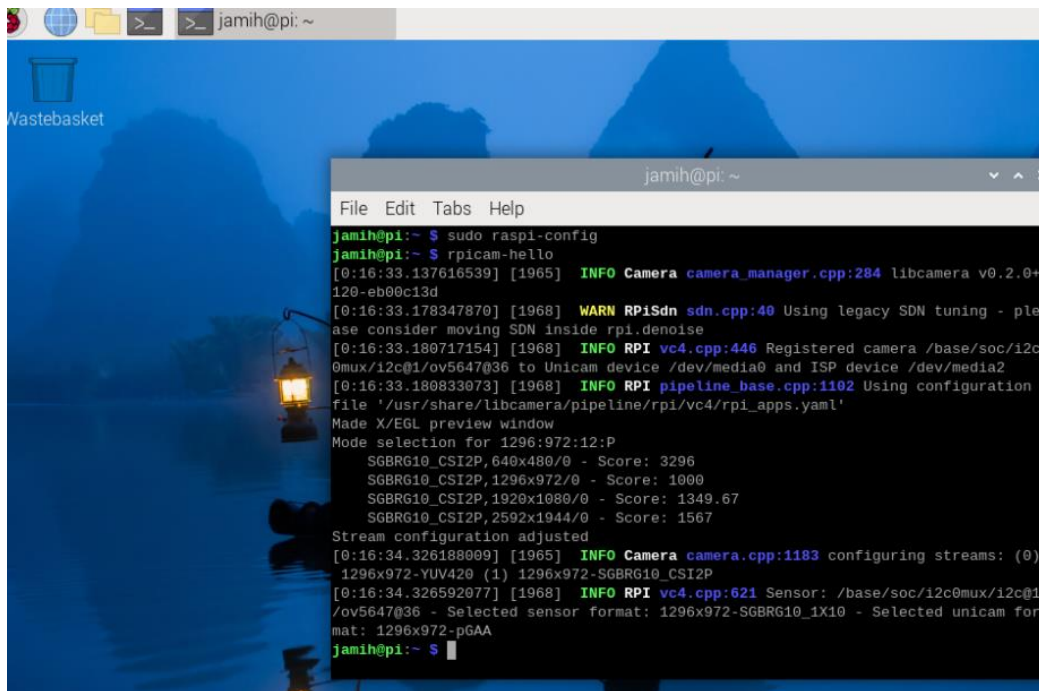


- d. Approving the settings and awaiting the full writing and verification process by the imager to the external storage.
- e. After the full installation, the memory card was removed from the reader and inserted into the Pi board, then the power was connected to the board and powered on while waiting for the board to bootload the OS in the card and connect to the network provided in the settings,
- f. After connectivity confirmation, we then installed an IP address scanner to determine the IP address of the Pi board (this is needed as we ran a headless configuration of the board) called "AngryIPScanner", provided both the PC and the Pi board are connected to the same network.



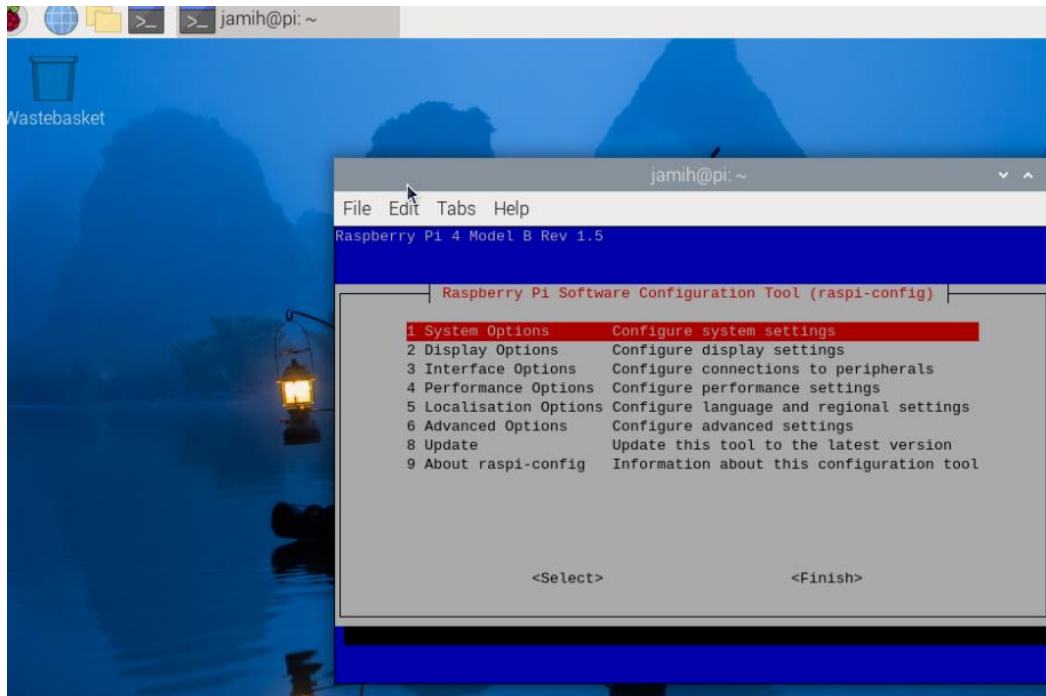
- g. After the IP has been determined, we then connect to the board using either way (Terminal for Linux users "ssh localname@ipaddress", installing Putty for Windows users) with the help of the IP address, authorization name and password was used to finally connect to the board.

- h. Once the board is finally connected, the terminal changes to the board's interface terminal (for Linux users), or a terminal will pop up from Putty (windows users).
- i. After this, the "sudo raspi-config" command was used to access the settings interface of the Pi OS to enable the remote display interface(VNC) connectivity and other settings.



The screenshot shows a Raspberry Pi desktop environment with a blue background featuring a lighthouse. A terminal window is open, displaying the following output:

```
jami@pi:~$ sudo raspi-config
jami@pi:~$ rpigam-hello
[0:16:33.137616539] [1965] INFO Camera camera_manager.cpp:284 libcamera v0.2.0+120-eb00c13d
[0:16:33.178347870] [1968] WARN RPiSdn sdn.cpp:40 Using legacy SDN tuning - please consider moving SDN inside rpi.denoise
[0:16:33.180717154] [1968] INFO RPI vc4.cpp:446 Registered camera /base/soc/i2c0mux/i2c01/ov5647@36 to Unicam device /dev/media0 and ISP device /dev/media2
[0:16:33.180833073] [1968] INFO RPI pipeline_base.cpp:1102 Using configuration file '/usr/share/libcamera/pipeline/rpi/vc4/rpi_apps.yaml'
Made X/EGL preview window
Mode selection for 1296:972:12:P
SGBRG10_CSI2P, 640x480/0 - Score: 3296
SGBRG10_CSI2P, 1296x972/0 - Score: 1000
SGBRG10_CSI2P, 1920x1080/0 - Score: 1349.67
SGBRG10_CSI2P, 2592x1944/0 - Score: 1567
Stream configuration adjusted
[0:16:34.326188009] [1965] INFO Camera camera.cpp:1183 configuring streams: (0) 1296x972-YUV420 (1) 1296x972-SGBRG10_CSI2P
[0:16:34.326592077] [1968] INFO RPI vc4.cpp:621 Sensor: /base/soc/i2c0mux/i2c01/ov5647@36 - Selected sensor format: 1296x972-SGBRG10_1X10 - Selected unicam format: 1296x972-pGAA
jami@pi:~$
```



- j. Installation of the VNC viewer was done, and using the IP address generated initially, we were able to connect to the Pi and the OS interface was displayed on the VNC viewer.



Once the VNC viewer displays this, the board is ready for use on our PC.

Finally, for the day, we made a simple connection of the Pi-Cam module to the board. We verified the simple initialization code for the camera bootup using the command **`"rpicam-hello --timeout 0"`**.



Above is the image taken by the Pi-cam module.

Week2-Day3: Completion of all tasks and update on Repository

Learning Objective: Completion of all pending tasks with upload to the GitHub repository and tracker sheet with the documented report

Hands-On Activity: LED blinking

Input: all documented task

Output: Documentation submitted

Activity: Students are to complete all pending tasks (given by trainers or profile-based), as well as update the tracker including pushing all respective documents into the repository.

Week2-Day4: Installation of Libraries and primary tasks verification

Learning Objective: Preparation of a ready-made RaspberryPi Board

Hands-On Activity: LED blinking, PWM, Button Interface, Analog Input

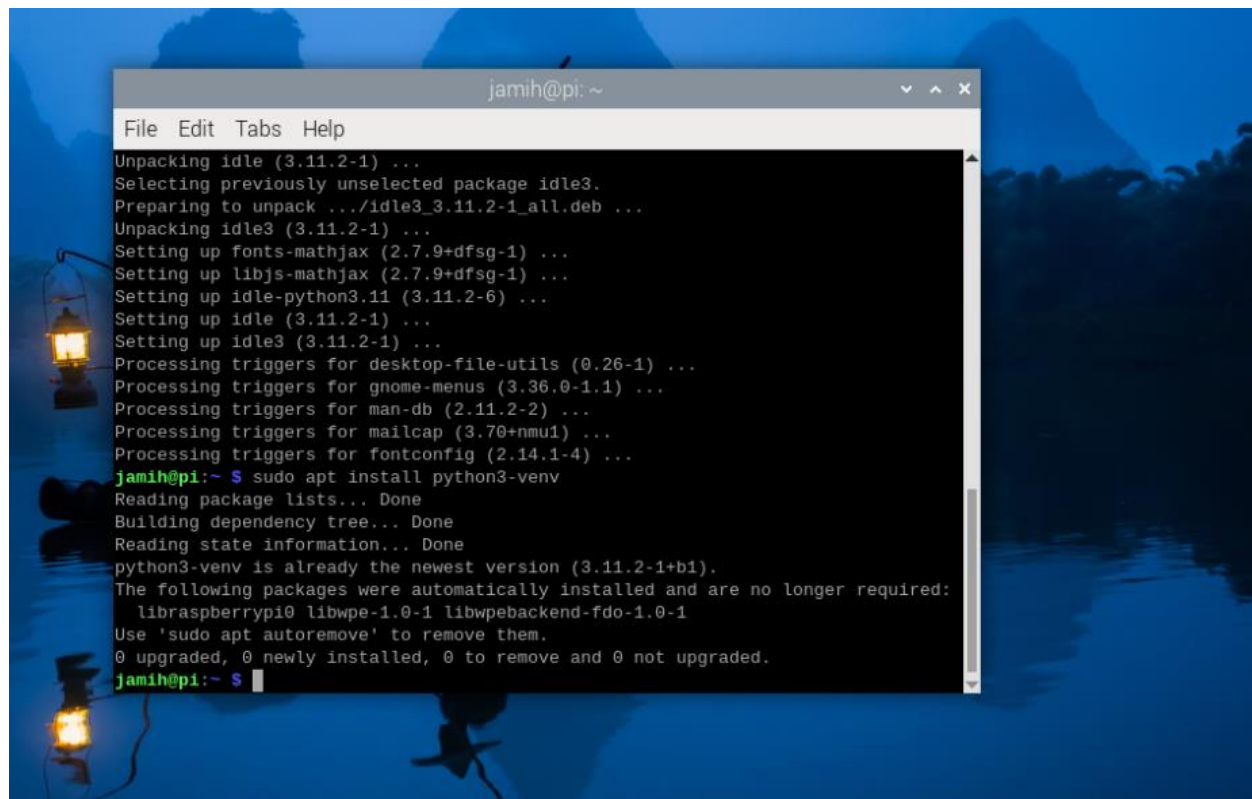
Input: Installation of Libraries and Configuration

Output: The working test of the aforementioned tasks.

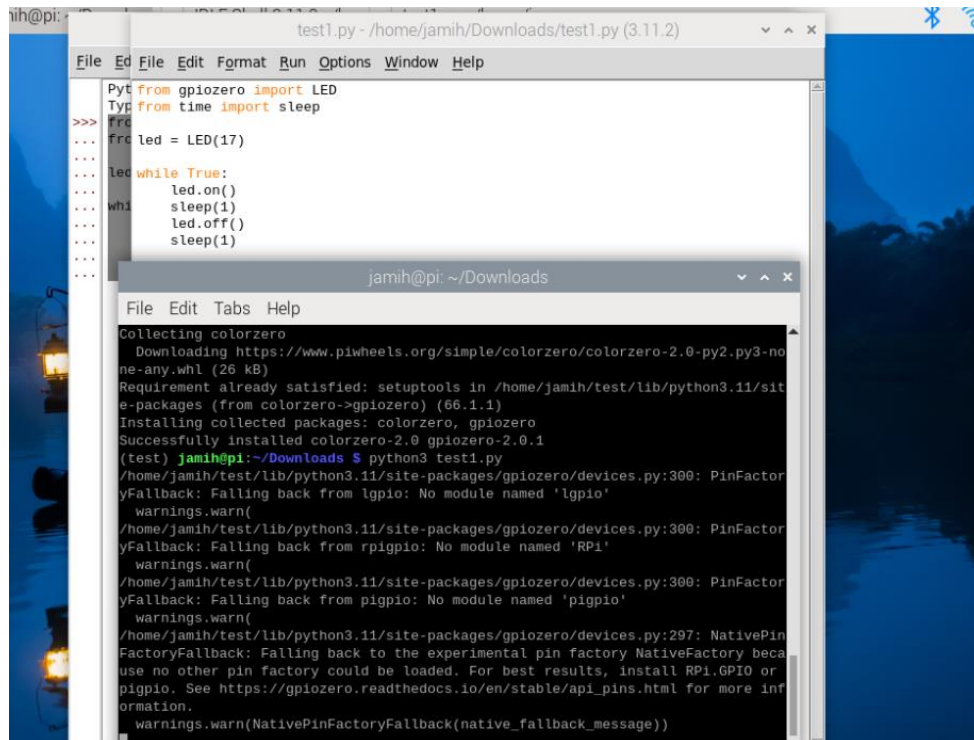
Tools: Raspberry Pi, Memory Card, Card Reader, Imager, VNC viewer, Push Button.

Activities (Stepwise process of PI board Configuration):

- a. Install idle3 for the programming interface using “sudo apt install idle3.”
- b. Installation of virtual environment library “sudo apt install python3-venv.”
- c. Create a virtual environment using
- d. Activate the virtual environment
- e. Install the GPIOzero used for programming and executing Raspberry Pi code
- f. Open the IDLE3 and create a .py file for the code
- g. Save the .py file in any directory
- h. Execute the file using “python3 filename.py”



Task1: Blinking of LED with the first created .py file and setting the hardware up



The screenshot shows a Raspberry Pi desktop environment with a blue background. Two windows are open. The top window is a text editor titled 'test1.py - /home/jamih/Downloads/test1.py (3.11.2)'. It contains the following Python code:

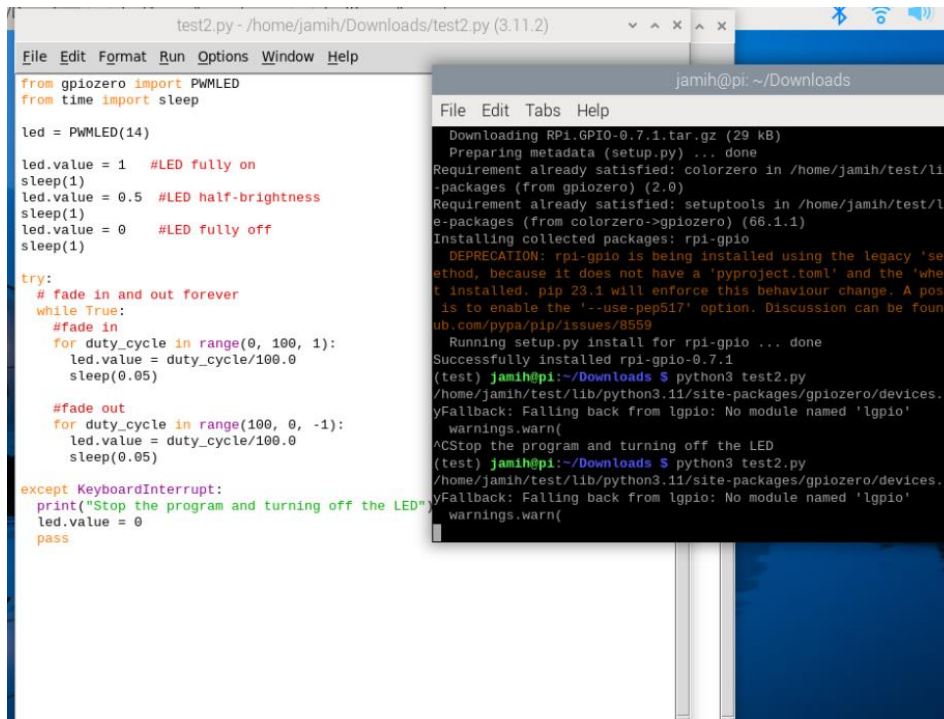
```
Pyt from gpiozero import LED
Typ from time import sleep
>>> frc led = LED(17)
...
... led while True:
...     led.on()
...     sleep(1)
...     led.off()
...     sleep(1)
... 
```

The bottom window is a terminal titled 'jamih@pi: ~/Downloads'. It shows the output of the command 'python3 test1.py'. The output includes the installation of 'colorzero' and 'gpiozero' from piwheels.org, followed by several warnings from the 'gpiozero' library regarding pin factories and fallbacks.

```
Collecting colorzero
  Downloading https://www.piwheels.org/simple/colorzero/colorzero-2.0-py2.py3-no
ne-any.whl (26 kB)
Requirement already satisfied: setuptools in /home/jamih/test/lib/python3.11/sit
e-packages (from colorzero->gpiozero) (66.1.1)
Installing collected packages: colorzero, gpiozero
Successfully installed colorzero-2.0 gpiozero-2.0.1
(test) jamih@pi:~/Downloads $ python3 test1.py
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:300: PinFactor
yFallback: Falling back from lgpio: No module named 'lgpio'
  warnings.warn(
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:300: PinFactor
yFallback: Falling back from rpigpio: No module named 'RPi'
  warnings.warn(
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:300: PinFactor
yFallback: Falling back from pigpio: No module named 'pigpio'
  warnings.warn(
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:297: NativePin
FactoryFallback: Falling back to the experimental pin factory NativeFactory beca
use no other pin factory could be loaded. For best results, install RPi.GPIO or
pigpio. See https://gpiozero.readthedocs.io/en/stable/api_pins.html for more inf
ormation.
  warnings.warn(NativePinFactoryFallback(native_fallback_message))
```


Task2: applying PWM on any GPIO with software or Hardware configuration on the PI board

Controlling the connected LED on any GPIO pin with the inbuilt “**PWMLED**” library gives the PWM configuration to the connected pin.



```
test2.py - /home/jamih/Downloads/test2.py (3.11.2)
File Edit Format Run Options Window Help

from gpiozero import PWMLED
from time import sleep

led = PWMLED(14)

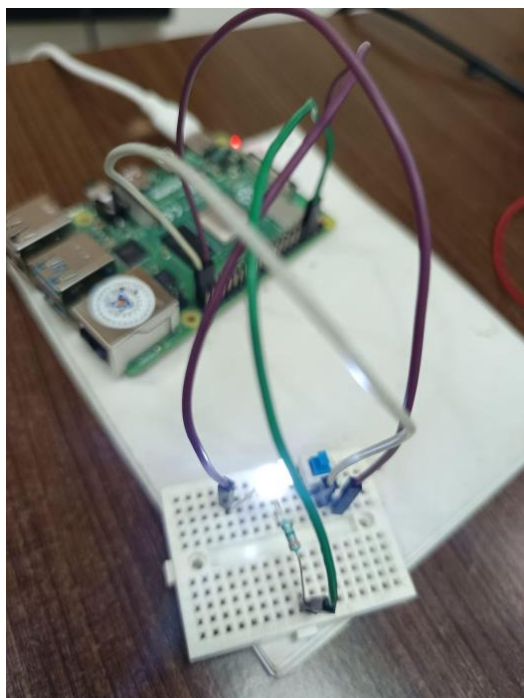
led.value = 1 #LED fully on
sleep(1)
led.value = 0.5 #LED half-brightness
sleep(1)
led.value = 0 #LED fully off
sleep(1)

try:
    # fade in and out forever
    while True:
        #fade in
        for duty_cycle in range(0, 100, 1):
            led.value = duty_cycle/100.0
            sleep(0.05)

        #fade out
        for duty_cycle in range(100, 0, -1):
            led.value = duty_cycle/100.0
            sleep(0.05)
except KeyboardInterrupt:
    print("Stop the program and turning off the LED")
    led.value = 0
    pass
```

```
jamih@pi: ~/Downloads
File Edit Tabs Help

Downloading RPi.GPIO-0.7.1.tar.gz (29 kB)
Preparing metadata (setup.py) ... done
Requirement already satisfied: colorzero in /home/jamih/test/lib/python3.11/site-packages (from gpiozero) (2.0)
Requirement already satisfied: setuptools in /home/jamih/test/lib/python3.11/site-packages (from colorzero->gpiozero) (66.1.1)
Installing collected packages: rpi-gpio
  DEPRECATION: rpi-gpio is being installed using the legacy 'setup.py' method, because it does not have a 'pyproject.toml' and the 'wheel' package is not installed. pip 23.1 will enforce this behaviour change. A possible solution is to enable the '--use-pep517' option. Discussion can be found at https://pip.pypa.io/issues/8559
Running setup.py install for rpi-gpio ... done
Successfully installed rpi-gpio-0.7.1
(test) jamih@pi:~/Downloads $ python3 test2.py
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:10: DeprecationWarning: Falling back from lgpio: No module named 'lgpio'
  warnings.warn(
^CStop the program and turning off the LED
(test) jamih@pi:~/Downloads $ python3 test2.py
/home/jamih/test/lib/python3.11/site-packages/gpiozero/devices.py:10: DeprecationWarning: Falling back from lgpio: No module named 'lgpio'
  warnings.warn(
```



Led blink/ PWM captured in the same state.

DIY Task1: Analog input on the Pi board

DIY Task2: Digital Input on the Pi board

Objective: Controlling an LED with a push button

Input: push button, Power supply

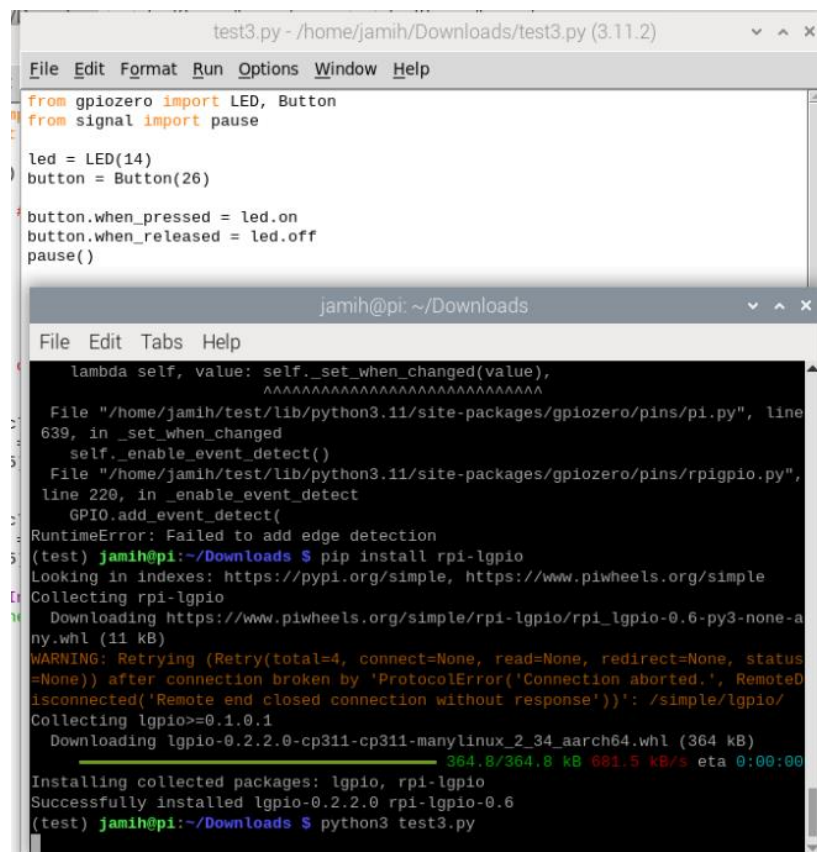
Output: LED

Connection: the push button **NO(Normally Open) pin** was connected to the **GPIO pin 26** of the board with a connection to the ground from the push button **GND pin**, the LED positive lead was connected to the **GPIO pin 14** of the Pi board with a resistor connected to the cathode lead of the LED to the ground pin of the Pi board.

Logic: Whenever the button is pressed, the LED should come up, and when the button is released, the LED is switched off with a slight delay in the processing between the button press and release.

Code: uploaded to GitHub

Output:



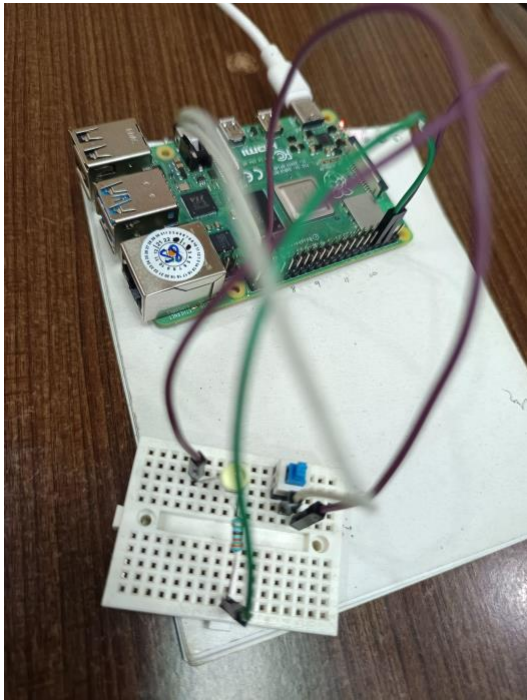
```
test3.py - /home/jamih/Downloads/test3.py (3.11.2)
File Edit Format Run Options Window Help
from gpiozero import LED, Button
from signal import pause

led = LED(14)
button = Button(26)

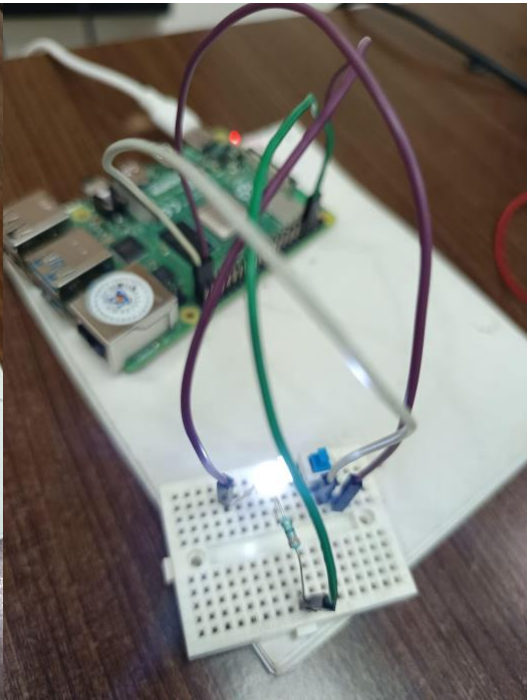
button.when_pressed = led.on
button.when_released = led.off
pause()

jamih@pi: ~/Downloads
File Edit Tabs Help

lambda self, value: self._set_when_changed(value),
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
File "/home/jamih/test/lib/python3.11/site-packages/gpiozero/pins/pi.py", line
639, in _set_when_changed
self._enable_event_detect()
File "/home/jamih/test/lib/python3.11/site-packages/gpiozero/pins/rpigpio.py",
line 220, in _enable_event_detect
GPIO.add_event_detect(
RuntimeError: Failed to add edge detection
(test) jamih@pi:~/Downloads $ pip install rpi-lgpio
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting rpi-lgpio
  Downloading https://www.piwheels.org/simple/rpi-lgpio/rpi_lgpio-0.6-py3-none-a
ny.whl (11 kB)
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status
=None)) after connection broken by 'ProtocolError('Connection aborted.', RemoteD
isconnected('Remote end closed connection without response'))': /simple/lgpio/
Collecting lgpio>=0.1.0.1
  Downloading lgpio-0.2.2.0-cp311-cp311-manylinux_2_34_aarch64.whl (364 kB)
364.8/364.8 kB @ 81.5 kB/s eta 0:00:00
Installing collected packages: lgpio, rpi-lgpio
Successfully installed lgpio-0.2.2.0 rpi-lgpio-0.6
(test) jamih@pi:~/Downloads $ python3 test3.py
```



when the button is released



when the button is pressed

Common issues:

- i. Libraries not found
- j. Version compatibility
- k. Hardware components range change
- l. Libraries compatibility (needed to install respective/affected library)