Lab-1 (Part-1)

Raspberry Pi Programming

Lab Goal:

During this lab you will build some basic programs on Raspberry Pi such as blinking an LED, changing the intensity of the LED, and building a security camera.

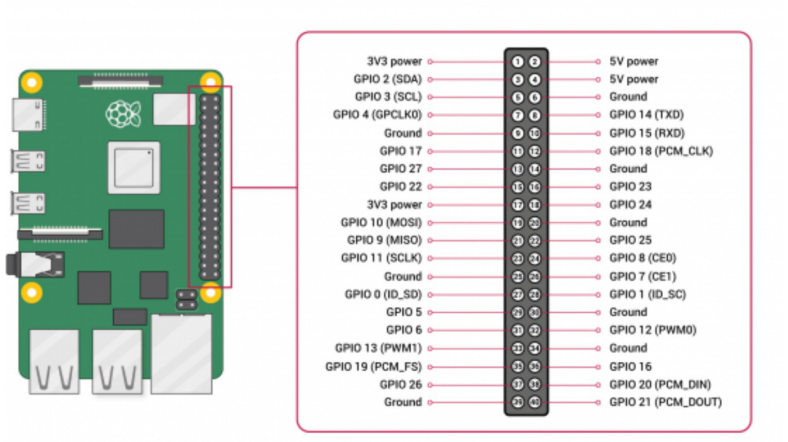
Background:

The Raspberry Pi (RPi) is a credit card-sized computer that is relatively low cost, and has been used for a wide variety of projects. The RPi runs a Linux distribution called Raspberry Pi OS. Raspberry Pi OS is a scaled down version of Debian Linux, built for the ARM architecture and including some useful packages.
This makes a Raspberry Pi a microcomputer (and not a microcontroller). Some basic characteristics of raspberry pi are:

Basis	Raspberry Pi
License	Both hardware and software of Raspberry Pi are closed source.
Control Unit	From ARM Family
Clock Frequency	Up to 1.5 GHz in Raspberry Pi 4 B
RAM	Requires large RAM (more than 1 GB)
CPU Architecture	64-bit
Logic level	Raspberry Pi's logic level is 3V.
Power Consumption	Consumes about 700 MW of power
Based on	Raspberry Pi is based on a microprocessor
Hardware Structure	Complex hardware Structure
Software	Raspberry Pi supports its own Linux-based operating system Raspberry Pi OS. You can also install the OS you like.
Internet	Raspberry Pi has a built-in Ethernet port and WiFi support.
Cost	Raspberry Pi boards are expensive.

Raspberry Pi GPIOs Overview:

GPIOs are the General Purpose Input and Output pins on Raspberry Pi. The GPIO enables you to switch devices on and off (output) or receive data from sensors and switches (input). This makes your Raspberry Pi a powerful System-on-Chip (SOC) that can be used for building a smart mirror, a weather station, an asset tracking robot that displays its coordinates on a web server, and many more. Below is a schematic of GPIO header for a raspberry Pi.



Components Required:

For this lab the components needed are:

- Raspberry Pi Power Supply
- Raspberry Pi Board
- Raspberry Pi Camera
- Raspberry Pi Sense Hat

Using Raspberry Pi Sense Hat

A Raspberry Pi Sense Hat is an add-on board that gives your Raspberry Pi a wide range of sensing capabilities. The add-on sensors include temperature, air pressure, humidity, color & brightness, orientation and IMU (accelerometer, gyroscope, magnetometer). It also consists of a 8x8 RGB LED matrix that allows the users to visualize data from the sensors. The LED matrix can be used to display messages and images. To get started with programming the inputs of Sense Hat or to access the outputs of the Sense Hat, connect it to your Raspberry Pi's GPIO Header. And power ON your Pi.

Exercise: Sensing the Environment

1. Connect the Sense Hat to your raspberry pi and boot your Pi.
2. Open the file, **sensehat_env.py**.
3. Review and run the code on IDE/terminal
4. It will display the current temperature, humidity and air pressure values (in C, %, millibars) on the IDE console.

8x8 LED Matrix :

he 8x8 LED matrix can be used for various applications such as showing colors on the LED screen, showing patterns/shapes or showing text on the matrix.

Our machines are programmed to store everything as 1s and 0s. These 1s and 0s are organized into sets of 8, called bytes, wherein a single byte can represent any number from 0 up to 255.

When we want to represent a color in a program, we can do this by defining the amounts of red, blue, and green that make up that color. These amounts are usually stored as a single byte and therefore as a number between 0 and 255. Here is a table showing the RGB values for each color.

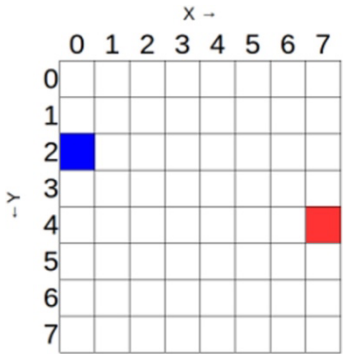
Red	Green	Blue	Colour
255	0	0	Red
0	255	0	Green
0	0	255	Blue
255	255	0	Yellow
255	0	255	Magenta
0	255	255	Cyan

Source:
https://github.com/raspberrypilearning/getting-started-with-the-sense-hat/blob/master/worksheet.md

So, in order to create the color blue, we initialize a variable blue = (0,0,255). Note that the integer values specify how bright each color should be; each value can be between 0 and 255 (brightest).

How can we access a single LED from the 8x8 LED Matrix?

In order to control each LED individually, we need to set pixels (LEDs) individually by using the ***sense.set_pixel(x_coordinate, y_coordinate, (R,G,B))*** function. Before we use this, it is important to understand how we describe each pixel. The Sense HAT uses a coordinate system like the one shown below. The origin of it is in the **top left (0,0)**.



- the blue pixel is at coordinates (0, 2)
- the red pixel is at coordinates (7, 4)

The program to replicate the above diagram (blue and red LED) will be as follows:

```
from sense_hat import SenseHat

sense = SenseHat()

sense.set_pixel(0, 2, (0, 0, 255))
sense.set_pixel(7, 4, (255, 0, 0))
```

Exercise: Display a Text on the LED Matrix

1. Open the file **senseHat_DisplayText.py**
2. Review the code and run the program in IDE/terminal.
3. It will start scrolling the text message on the LED Matrix.

***sense.show_message()* function will scroll the text on LED matrix.** By default, the scrolling speed is 0.1. The bigger the number the lower the speed. You can set the parameters such as *text_colour, back_colour,* and *scroll_speed* in the show_message() function to customize the speed and color of your message. Note that, the parameter *text*_colour alters the color of the text and is defined via three values to specify red, green, and blue. These are also called RGB values.

4. Uncomment lines #5, #6, and #11 to visualize how the text color, background_color, and scroll speed changes on your LED matrix.

Note: In order to display a single character, instead of using the *show_message()* command, we use the *show_letter() method.* We can change how the letter is displayed by using two of the same parameters, *text_colour* and *back_colour*. Single characters do not scroll, so there is no *scroll_speed* parameter for displaying a letter.

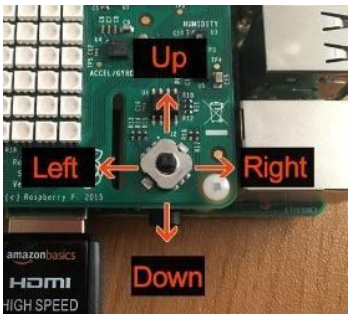
Checkpoint-1 (3 points):

Capture the current temperature, pressure and humidity values and display them on the LED matrix using different colors. Set scrolling speed to 0.05. Call the TA to check your program once you are done with this exercise. Change the temperature reading by touching/rubbing against the sensor or blowing onto the sensor. If the measured temperature changes by 1 degree Celsius from the initial reading, set the LED pixel on coordinate (3,3) to start blinking. Call the TA once you complete this exercise.

Hint: Use the set_pixel() method to control a single LED pixel from the 8x8 matrix. You can learn more about the sense-hat commands for measuring humidity/pressure and controlling LEDs from this link.

Using the Joystick

The Sense Hat Joystick is mapped to four keyboard cursor keys and the middle click corresponds to the Return Key.This implies that using the joystick has the same effect as pressing those cursor keys on the keyboard. .



1. Open the file **senseHat_joystick.py**
2. Review the code and run the program in IDE.
3. Press the middle-click of the joystick interface. The event will trigger the display of a single character on the LED matrix.

Checkpoint-2 (3 points):

Write a simple python program on your Raspberry Pi to start a camera preview window and capture an image when the middle-click of the joystick is pressed. Print the name of the file to the LED matrix when it's done.

You can learn more about how to program the Joystick handler here and here. Use the camera code below to help you:

```
from picamera2 import Picamera2, Preview
import time
picam2 = Picamera2()  ## Open the camera instance
camera_config = picam2.create_preview_configuration() ## Create a camera configuration suitable for preview
picam2.configure(camera_config) ## Configure the camera instance with the preview configuration
picam2.start_preview(Preview.QTGL) ## Start the Preview Window
picam2.start() ## Start the running Camera
time.sleep(2) ## Pause for 2 seconds
picam2.capture_file("test.jpg")## Capture and store the image as jpg
```

General Resources

SenseHat API: https://pythonhosted.org/sense-hat/api/

Extra Credit Checkpoint (3 points):

While experimenting with the temperature sensor, you might have noticed that the readings from these sensors exhibit significant fluctuations. Write a program that uses moving averaging to smooth out the sensor readings. Plot a graph for old temperature measurements w.r.t time along with the curated temperature measurements w.r.t time.