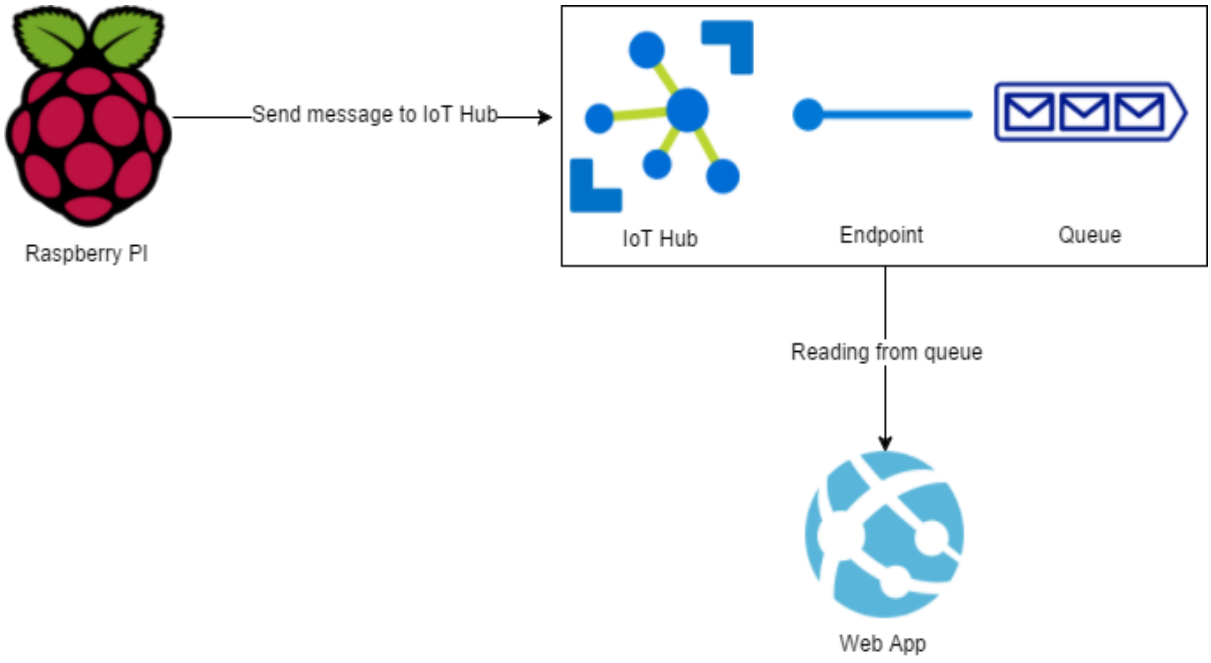
IOT SENSOR SYSTEM AND RASBERRY PI INTEGRATION

INTRODUCTION:

Building an IoT sensor system and integrating it with a Raspberry Pi involves several steps, including selecting and setting up the hardware, choosing appropriate sensors, programming the Raspberry Pi, and implementing communication protocols. Here's a step-by-step guide to help you get started



OBJECTIVE:

This project describes a room control and monitoring system using Internet of Things (IoT). It aims to help the user specifically the elderly to monitor their room automatically wirelessly though mobile application. This project using Raspberry Pi to communicate with Firebase cloud platform and mobile application developed using Android Studio. The system accumulates the temperature, humidity and gas surrounding the room and execute the output to the actuator to act and the sensors data are stored into the database. In mobile application, user can login to view the real-time sensors data with date and time.

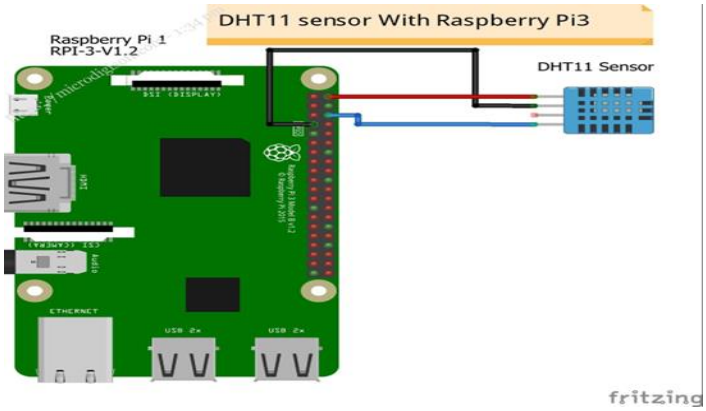
SENSORS:

The Raspberry Pi is well-loved as a general-purpose computer, commonly used for programming, hosting files, emulation, and browsing the web. However, one of the unique features of the Raspberry Pi is its GPIO header which allows it to interface with physical electronic components such as LEDs, switches, and servos, as well as sensors to measure a variety of things.

1.WEATHER STATION:

You'll need a Raspberry Pi model with a 40-pin GPIO header, a suitable weatherproof enclosure (if you plan to keep it outside), and a suitable high spot to install the external weather sensors. The Adafruit IO service is used set up a web dashboard that can be viewed from any device.

2.HUMIDITY AND TEMPERATURE:



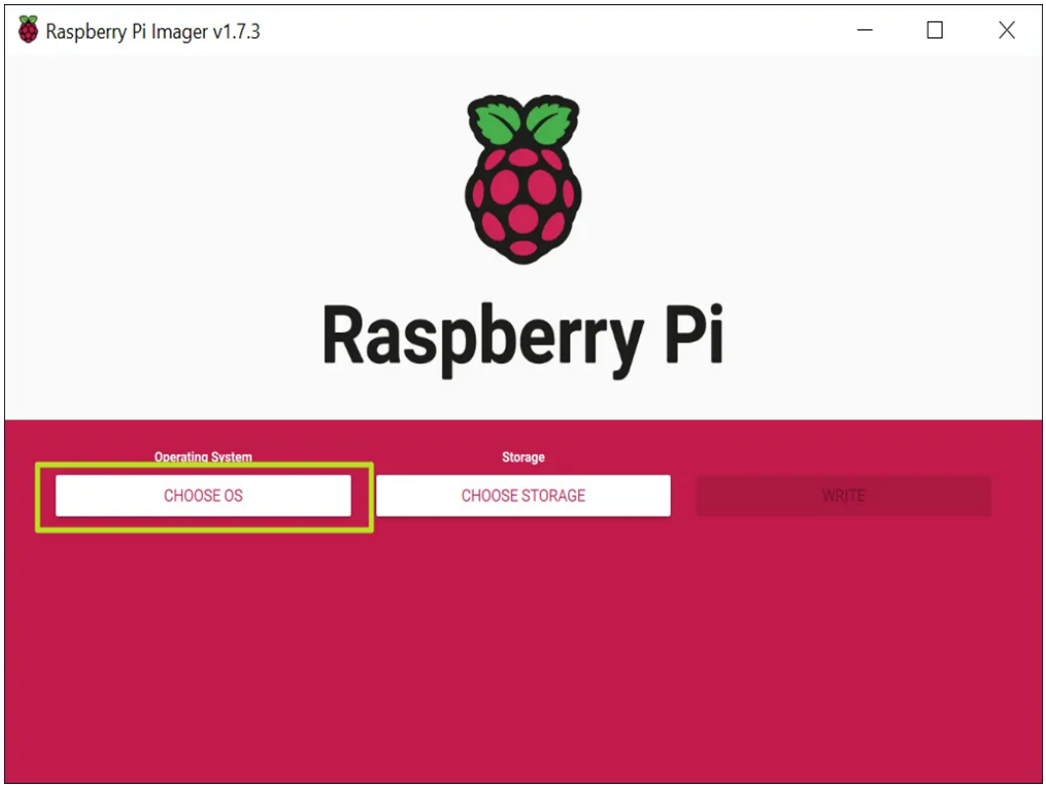
- *DHT11 sensor measures and provides humidity and temperature values serially over a single wire.*
- *It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.*
- *It has 4 pins; one of which is used for data communication in serial form.*
- *Pulses of different TON and TOFF are decoded as logic 1 or logic 0 or start pulse or end of a frame.*

SET UP RASPBERRY PI:

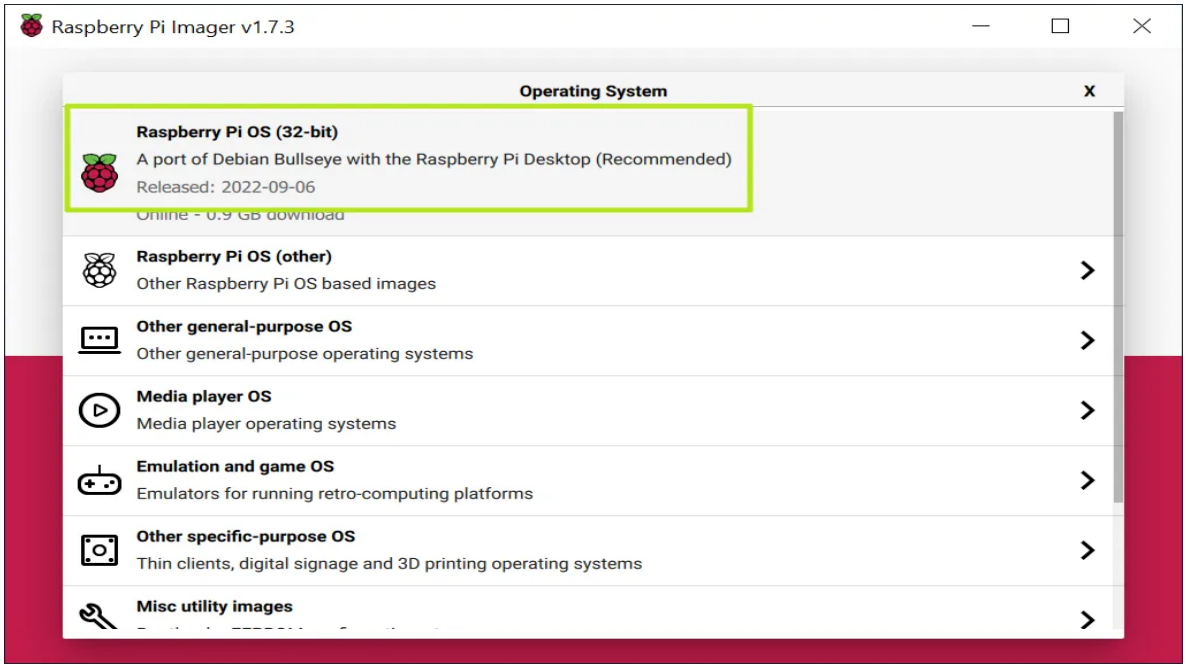
Downloading and Installing Raspberry Pi OS

Once you have all the components you need, use the following steps to create the boot disk you will need to set up your Raspberry Pi. These steps should work on a using a Windows, Mac or Linux-based PC (we tried this on Windows, but it should be the same on all three).

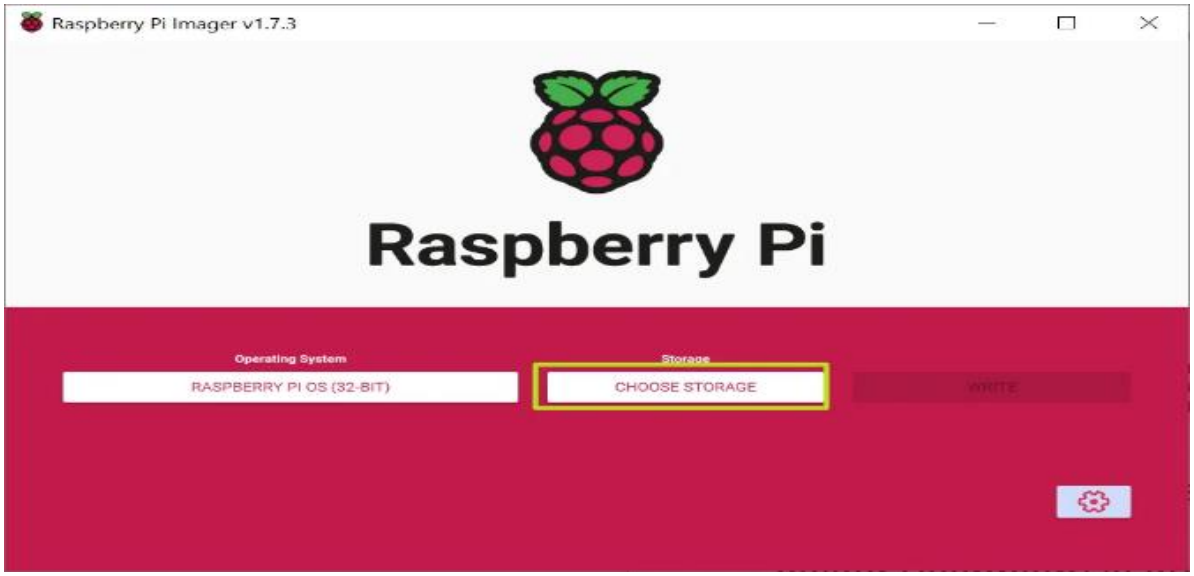
1. ***Insert a microSD card / reader*** *into your computer.*
2. ***Download and install the official Raspberry Pi Imager.*** *Available for Windows, macOS or Linux, this app will both download and install the latest Raspberry Pi OS. There are other ways to do this, namely by downloading a Raspberry Pi OS image file and then using a third-party app to "burn it," but the Imager makes it easier.*
3. ***Click Choose OS.***



Select Raspberry Pi OS (32-bit) *from the OS menu (there are other choices, but for most uses, 32-bit is the best).*



Click Choose storage *and* ***pick the SD card*** *you're using.*



Booting Your Raspberry Pi for the First Time

After you're done writing the Raspberry Pi OS to a microSD card, it's time for the moment of truth.

*1. **Insert the microSD card** into the Raspberry Pi.*
*2. **Connect the Raspberry Pi** to a monitor, keyboard and mouse.*
*3. **Connect an Ethernet cable** if you plan to use wired Internet.*
*4. **Plug the Pi in** to power it on.*
If you had used the Raspberry Pi Imager settings to create a username and password, you'll be able to go straight into the desktop environment, but if not, you will get a setup wizard.

```
import RPi.GPIO as GPIO                    #1
import time                                #2

pirsensor = 4                              #3
GPIO.setmode(GPIO.BCM)                     #4
GPIO.setup(pirsensor, GPIO.IN, GPIO.PUD_DOWN)  #5

previous_state = False                     #6
current_state = False

while True:                                #7
    time.sleep(0.1)                        #8
    previous_state = current_state         #9
    current_state = GPIO.input(pirsensor)  #10
    if current_state != previous_state:    #11
        if current_state:                  #12
            print("Motion Detected!")
```

- *Import the Requests Python library that helps you to send HTTP requests to a server easily without worrying about query strings, form-encoding your POST data, etc.*
- *Import the JSON library so that you can print out the data that was sent to the GCM server in JSON format.*
- *The end point for Google's GCM server.*
- *The API Key that you've obtained from Google. This identifies the application developer sending the push notification.*
- *The Registration ID(s) of the application receiving the notification. You get this Registration ID from the application after it has registered with Google. In the real world, this Registration ID should be sent to the server maintained by the developer to provide a complete list of Registration IDs of the app installed on the users' devices. If you want to send a push message to multiple recipients, separate the Registration IDs with commas (,).*
- *The HTTP header to sent to the GCM server to authenti*cate the identity of the sender.

Conclusion:

The Raspberry Pi is a powerful little beast and a great platform for building low-cost, but highly capable, embedded systems. The interfaces built into its GPIO connector make it easy to bolt on modules using simple low-cost electronics and a bit of configuration to create very functional and flexible system