MOBILE APP DEVELOPMENT USING MIT INVERTOR ON NODE-RED SERVICE

INTRODUCTION:

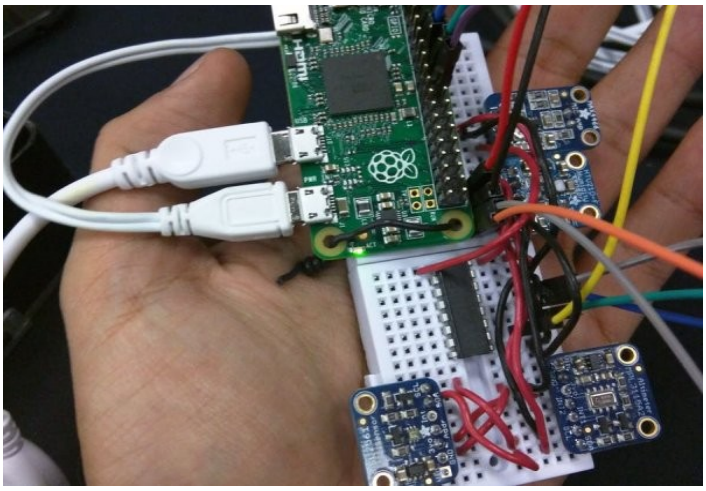
To read temperature and humidity data from a DHT11 or DHT22 sensor using raspberry.pi.Here is a python program using the Adafruit DHT library to accomplish this.

STEPS:

1. First, you'll need to install the Adafruit DHT library if you haven't already. Open a terminal on your Raspberry Pi and run the following command:

```
bash
```

CIRCUIT :



Preparing the Raspberry Pi

Installing the OS

1.- Download the latest Raspbian.

https://www.raspberrypi.org/downloads/raspbian

2.- Follow the instructions to install the Raspbian in your micro SD card.

if you are using Linux you can follow these steps to install Raspbian Lite on your micro SD card:

– Check the device name for your micro SD by running :

```
[james@fedora22 mnt] $ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        3.8G     0  3.8G   0% /dev
tmpfs           3.8G   79M  3.7G   3% /dev/shm
tmpfs           3.8G  1.5M  3.8G   1% /run
tmpfs           3.8G     0  3.8G   0% /sys/fs/cgroup
/dev/sda1       451G  175G  253G  41% /
tmpfs           3.8G  408K  3.8G   1% /tmp
tmpfs           769M  8.0K  769M   1% /run/user/42
tmpfs           769M   28K  769M   1% /run/user/1000
/dev/mmcblk0    7.4G  4.0K  7.4G   1% /run/media/james/3980-8C72
```

– Unmount your device with the following command

```
[james@fedora22 mnt] $ sudo dd bs=4M if=/home/james/Downloads/2015-11-21-raspbia
n-jessie-lite.img of=/dev/mmcblk0
[sudo] password for james:
347+1 records in
347+1 records out
1458569216 bytes (1.5 GB) copied, 218.893 s, 6.7 MB/s
[james@fedora22 mnt] $
```

1.- Expand the Filesystem and Enable I2C

- Login as user: pi password: raspberry
- Execute the command sudo raspi-config in the terminal
- Select Expand Filesystem and press Enter
- Select OK and you will return to the main menu
- Select Advanced Options
- Select I2C and press Enter
- Select Yes and press Enter
- Select OK and press Enter
- Select Yes and press Enter
- Select OK and you will return to the main menu
- Select Finish and press Enter
- Select Yes and press Enter to reboot the Raspberry pi

2.Create a Python script (e.g., temperature_humidity.py) and add the following code:

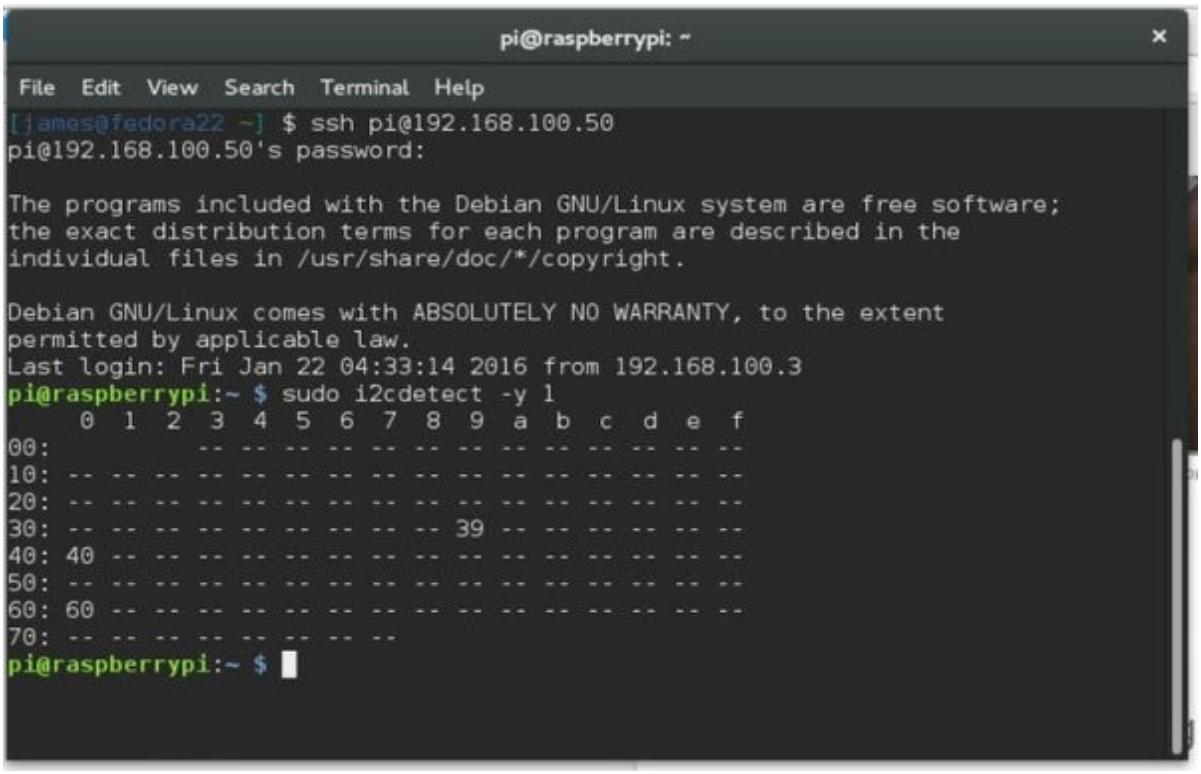
```
import Adafruit_DHT
import time

# Set the GPIO pin where your sensor is connected
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4  # Replace with the actual GPIO pin number

try:
    while True:
        humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
        if humidity is not None and temperature is not None:
            printf("Temperature: {temperature:.2f}°C, Humidity: {humidity:.2f}
%")
        else:
            printf("Failed to retrieve data from the sensor. Check the wiring.")
        time.sleep(2)  # You can adjust the sleep duration as needed
except KeyboardInterrupt:
    printf("Program terminated by user.")
except Exception as e:
    printf(f"Error: {str(e)}")
```

3.Save the script and run it:

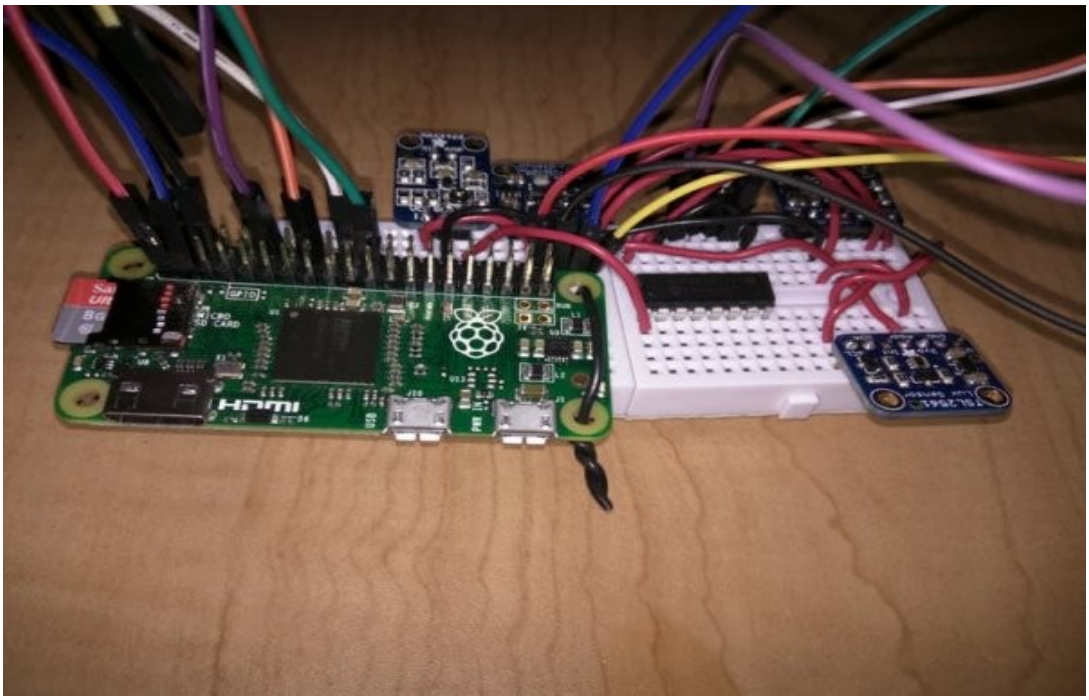
The script will continuously read the temperature and humidity values from the DHT sensor and print them to the terminal. Press **Ctrl+C** to stop the program.

Remember that the DHT11 and DHT22 sensors may have different pinouts and require different parameters, so make sure to adjust the code accordingly. Additionally, double-check your wiring and make sure you have the necessary permissions to access GPIO pins (usually, you need to run the script with superuser privileges or add your user to the group)

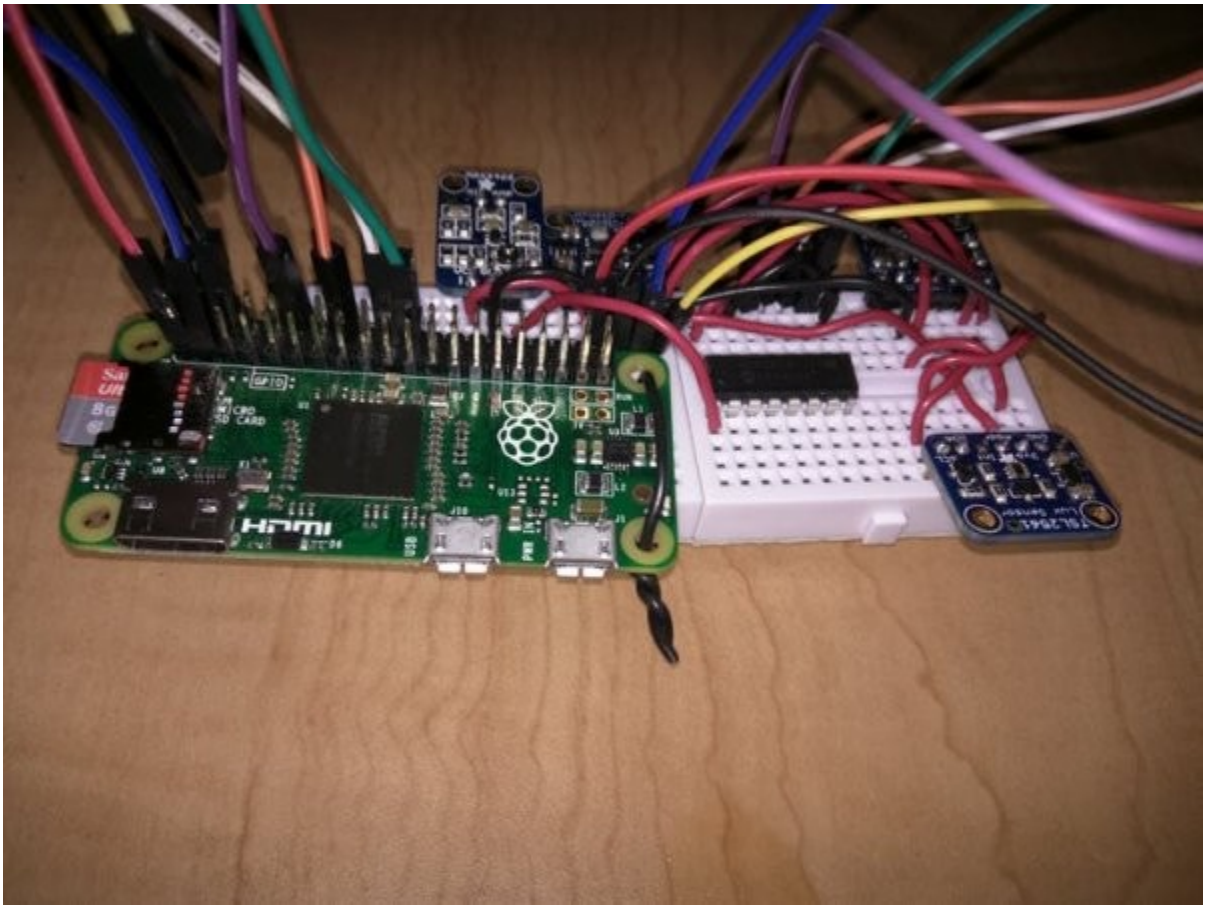
You should see the following output



Once you finish the wiring and sensor placement your device should look something like this.


HDMI

Once you finish the wiring and sensor placement your device should look something like this.



- **Define Your Environmental Monitoring Requirements:** Determine what kind of data you want to monitor and how you want to display it in your mobile app. This could include temperature, humidity, air quality, or any other relevant environmental parameters.

- **Set Up Hardware Sensors:** Connect environmental sensors (e.g., temperature sensors, humidity sensors, air quality sensors) to microcontrollers like Arduino or Raspberry Pi. These sensors will collect data from the environment.

- **Node-RED Configuration:** Use Node-RED to create flows that receive data from your sensors. You can use Node-RED's extensive library of nodes to interface with various sensors and devices. Transform the data and send it to a database or cloud service for storage and analysis.

- **Database or Cloud Service:** Store the data collected from the sensors in a database or cloud service. This can be a simple local database, cloud-based databases like Firebase, or IoT platforms like AWS IoT or Azure IoT.

- **Build the Node-RED-API Bridge:** Create an API in Node-RED to access the data you've collected. This API will provide a way for the MIT App Inventor to request and retrieve data from Node-RED.

- **MIT App Inventor:** Using MIT App Inventor, design and build your mobile app. You can create the app's user interface, define how the data will be displayed, and set up features for data retrieval. Use the Web component in MIT App Inventor to make HTTP requests to the Node-RED API.

- **Data Visualization:** Implement data visualization in your app, such as charts, graphs, or simple text displays, to present the environmental data in a user-friendly way.

- **Real-time Updates:** Use features in MIT App Inventor and Node-RED to enable real-time updates. For example, you can set up Node-RED to push data to the app whenever a new reading is available.

- **Testing and Debugging:** Test your mobile app and Node-RED flows thoroughly to ensure they work as expected. Debug any issues that may arise during testing.

- **Deployment:** Once your app and Node-RED flows are working correctly, deploy them to the target environment, such as a mobile device or a tablet. Users can then install and use the app for environmental monitoring.

- **Maintenance:** Regularly maintain and update your app and monitoring system to ensure data accuracy and reliability.

❖ Creating a complete Node-RED flow for environmental monitoring is a detailed process, and it depends on the specific sensors and hardware you are using. However, I can provide you with a simple example of a Node-RED flow that simulates environmental data (temperature and humidity) and sends it to a cloud-based database (Firebase) for storage. You can adapt and expand this flow according to your specific sensors and requirements.

Basic Node-RED flow:

- **Inject Node**: Use an "inject" node to simulate data. You can set it to inject data periodically (e.g., every minute).
- **Function Node (Temperature and Humidity Simulation)**: Use a "function" node to generate simulated temperature and humidity data. You can use JavaScript to create random values within the desired range.

Example function node code:

```
msg.payload = {
   temperature: Math.floor(Math.random() * 30) + 10,
   humidity: Math.floor(Math.random() * 70) + 30,
};
return msg;
```

- **Firebase Output Node**: Configure a Firebase output node to write the data to your Firebase Realtime Database.

    1. In the Firebase Output node, set up the Firebase configuration with your database URL and credentials.
    2. Configure the node to write the data to the appropriate Firebase path (e.g., "/environmental_data").

- **Debug Node**: Add a "debug" node to check the data before it is sent to Firebase. This is useful for debugging purposes.

- **Dashboard (Optional)**: If you want to visualize the data in Node-RED's dashboard, you can use dashboard nodes (e.g., "ui_chart" or "ui_text") to display the data.

Use in a Node-RED "function" node to generate a timestamp and send it as part of the payload:

```
var currentDate = new Date();

var timestamp = currentDate.getTime();

var formattedTimestamp = currentDate.toISOString();

var payload = {

   timestamp: timestamp,

   formattedTimestamp: formattedTimestamp

};

msg.payload = payload;

return msg;
```

```
const temperature = Math.floor(Math.random() * 30) + 10;

const humidity = Math.floor(Math.random() * 70) + 30;

const data = {

  temperature: temperature,

  humidity: humidity

};

const jsonData = JSON.stringify(data);

const xhr = new XMLHttpRequest();

const url = "https://your-server-endpoint.com/api/environment-
data";



xhr.open("POST", url, true);

xhr.setRequestHeader("Content-Type", "application/json");

xhr.onload = function() {

  if (xhr.status === 200) {

    console.log("Data sent successfully");

  } else {

    console.error("Failed to send data. Status code: " + xhr.status);

  }

};

xhr.send(jsonData);
```