Table of Content

2. Defining a function to get the weather forecast from the API
3. Defining a function to update the e-Paper Display
4. Creating a function to handle the button press
5. Setting up the main loop

V. Customizing the Display
A. Adjusting the text, font size, and arrangement
B. Adding images or shapes to the display

VI. Testing and Troubleshooting
A. Running the script
B. Identifying and resolving potential issues

VII. Conclusion
A. Summary of the project
B. Potential improvements and future work

VIII. References
A. Relevant resources and documentation
B. Acknowledgments and attributions

I. INTRODUCTION

A. Brief description of the project

This project aims to create an indoor weather station using a Raspberry Pi, a DHT22 temperature and humidity sensor, a Waveshare 7.5-inch Multi-Color e-Paper Display, and an LED. The weather station will display indoor temperature and humidity, as well as outdoor weather conditions obtained from a weather API. A push-button will be used to update the displayed information, while the LED will indicate if the indoor temperature exceeds a specified threshold, simulating the activation of an air conditioner.

B. Importance and applications of the project

The indoor weather station project serves as a practical application of Raspberry Pi and various sensors, providing an engaging learning experience for beginners in electronics and programming. It demonstrates the integration of hardware components, software development, and API usage, offering insights into various aspects of IoT and smart home projects.

The weather station can be utilized in various contexts, such as residential, educational, or commercial settings, for monitoring indoor conditions and staying informed about outdoor weather. By having instant access to indoor and outdoor weather information, users can make informed decisions about heating, cooling, and ventilation, leading to increased energy efficiency and cost savings.

Moreover, this project can serve as a foundation for more advanced projects, such as integrating other sensors or adding smart home automation capabilities. For example, it can be expanded to control an actual air conditioning system, trigger

alerts based on temperature or humidity thresholds, or connect to a smartphone app for remote monitoring and control.

OVERVIEW OF THE COMPONENTS AND TECHNOLOGIES USED

Raspberry Pi: A compact and versatile single-board computer that serves as the central processing unit for the project. It is responsible for reading data from the DHT22 sensor, obtaining weather information from the API, updating the e-Paper display, and controlling the LED.

DHT22 temperature and humidity sensor: A low-cost, high-precision sensor for measuring indoor temperature and humidity levels. It provides digital output, which is read by the Raspberry Pi using a Python library.

Waveshare 7.5-inch Multi-Color e-Paper Display: A high-resolution, low-power display that is suitable for showing static or infrequently updated information, making it an ideal choice for this project. The display is controlled by the Raspberry Pi using the Waveshare e-Paper library.

Push-button: A simple input device used to trigger the update of information displayed on the e-Paper screen. The button is connected to a GPIO pin of the Raspberry Pi, which detects the button press and updates the display accordingly.

LED: A visual indicator that simulates the activation of an air conditioner when the indoor temperature exceeds a specified threshold. It is controlled by the Raspberry Pi using a GPIO pin.

Weather API: An online service that provides real-time weather information, such as temperature, humidity, and weather conditions. The project uses the API to obtain outdoor weather data, which is displayed on the e-Paper screen alongside the indoor sensor readings.

Python: The programming language used for developing the software component of the project. Python libraries are used to communicate with the hardware components and weather API, as well as to handle the processing and display of the data.

II. COMPONENTS AND MATERIALS

A. List of required components

1. Raspberry Pi
2. DHT22 temperature and humidity sensor
3. Waveshare 7.5-inch Multi-Color e-Paper Display
4. Push-button
5. LED
6. 330 Ohm resistor
7. Breadboard
8. Jumper wires
9. Weather API key

B. Brief description of each component

Raspberry Pi: A small, affordable single-board computer used in a wide range of applications, from hobbyist projects to industrial applications. In this project, it acts as the central controller for reading data from the DHT22 sensor, interacting with the weather API, updating the e-Paper display, and controlling the LED.

FIG 1: A Raspberry Pi

DHT22 temperature and humidity sensor: A digital sensor that measures temperature and humidity with high accuracy and provides a digital output. It is widely used in various environmental monitoring projects and can operate in a wide range of conditions.

FIG 2: DHT22 Temperature and Humidity Sensor

Waveshare 7.5-inch Multi-Color e-Paper Display: An e-ink display with high resolution, low power consumption, and a wide viewing angle. E-paper displays are ideal for applications where the content does not change frequently, making it perfect for displaying weather data in this project. The display can show black, white, and red colors and comes with a hat or adapter for easy connection to the Raspberry Pi.

FIG 3: E-Paper Display

Push-button: A simple, momentary switch that is used to trigger an update of the information displayed on the e-Paper screen. When the button is pressed, the

Raspberry Pi detects the event and updates the display with the latest data from the DHT22 sensor and the weather API.

FIG 4: Push button

LED: A light-emitting diode that serves as a visual indicator in this project. The LED simulates the activation of an air conditioner when the indoor temperature exceeds a specified threshold. It is connected to a GPIO pin of the Raspberry Pi, which controls its on/off state.

FIG 5: An LED

330 Ohm resistor: A resistor used to limit the current flowing through the LED when it is connected to a GPIO pin of the Raspberry Pi. This helps protect both the LED and the Raspberry Pi from potential damage caused by excessive current.

FIG 6: A Resistor

Breadboard: A solder-less prototyping board used to connect and test electronic components in a circuit. In this project, the breadboard is used to connect the push-button, LED, and the 330 Ohm resistor to the Raspberry Pi.

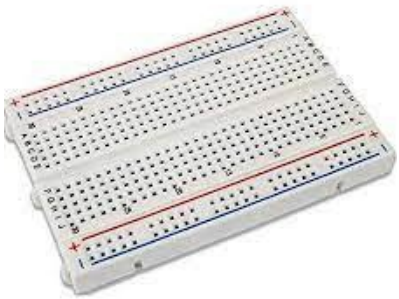

FIG 7: A Breadboard

Jumper wires: Wires with connector pins at each end, used to establish connections between the components and the Raspberry Pi. They can be easily plugged into the breadboard and the GPIO pins on the Raspberry Pi.

FIG 8: Jumper Wires

Weather API key: A unique identifier provided by a weather data provider, such as OpenWeatherMap or Weather API, which grants access to their weather data. The API key is used in the Python script to request weather data for a specific location, which is then displayed on the e-Paper screen.

III. HARDWARE ASSEMBLY

A. DHT22 sensor connection

- Connect the DHT22 sensor's VCC pin to the 3.3V pin on the Raspberry Pi.
- Connect the DHT22 sensor's GND pin to a ground pin on the Raspberry Pi.
- Connect the DHT22 sensor's data pin (usually marked as S or OUT) to the desired GPIO pin on the Raspberry Pi (in our example, we use GPIO 4).
- Connent a 10Kohm resistor between the data pin and VCC pin

B. Push-button connection

- Connect one side of the push-button to a GPIO pin on the Raspberry Pi (in our example, we use GPIO 3).
- Connect the other side of the push-button to a ground pin on the Raspberry Pi.
- Enable the internal pull-up resistor for the GPIO pin connected to the button in the Python script using the following line of code:

  GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

C. LED connection

- Connect the LED's anode (longer leg) to a GPIO pin on the Raspberry Pi (in our example, we use GPIO 14).
- Connect the LED's cathode (shorter leg) to a ground pin on the Raspberry Pi through a suitable resistor (e.g., 330 ohms) to limit the current and protect the LED

D. e-Paper Display connection

For connecting the Waveshare e-Paper display with a connector that has DC, CLK, DIN, BUSY, RST, GND, and 3V3 pin-outs, follow these steps:

- Connect the 3V3 pin on the display connector to the 3.3V pin on the Raspberry Pi.
- Connect the GND pin on the display connector to a ground pin on the Raspberry Pi.
- Connect the following pins between the display connector and the Raspberry Pi:

  DC (display) to SPI0 CE0 (GPIO 8) (Pi)

  CLK (display) to SPI0 SCLK (GPIO 11) (Pi)

  DIN (display) to SPI0 MOSI (GPIO 10) (Pi)

  BUSY (display) to a GPIO pin (GPIO 24) (Pi)

  RST (display) to a GPIO pin (GPIO17) (Pi)

After connecting all the components, your hardware assembly is complete, and you can proceed with the software setup and running the Python script.

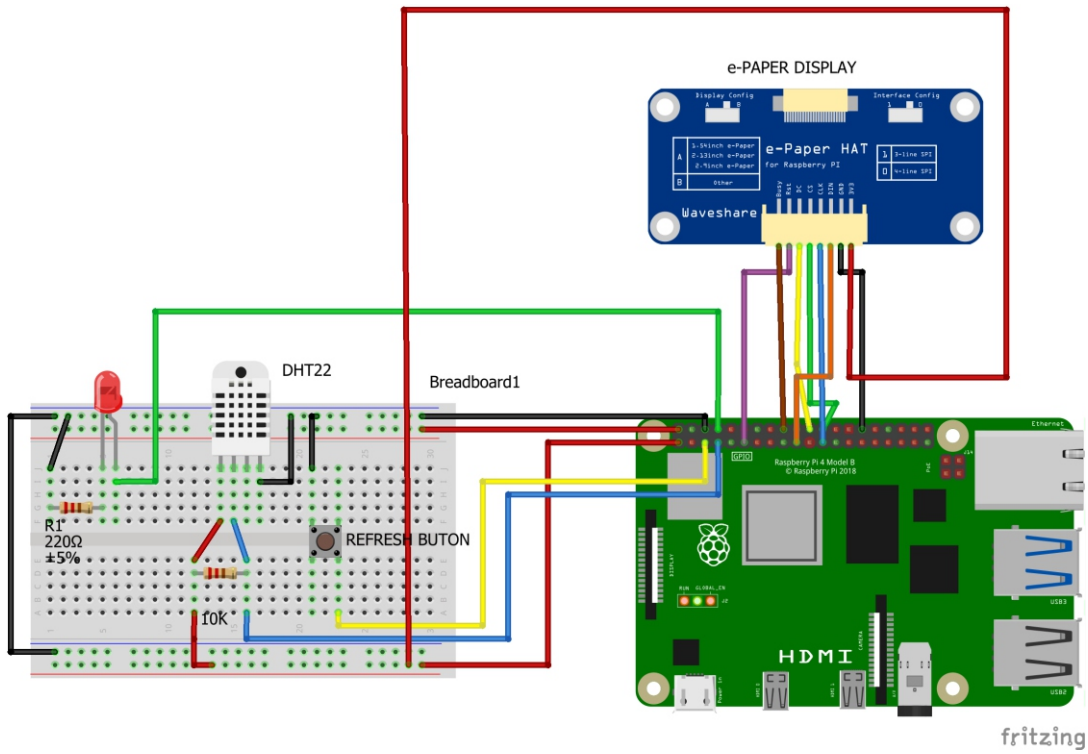

FIG 9: Circuit Connection Diagram

IV. SOFTWARE SETUP

A. Installing necessary libraries and dependencies

Updating Raspberry Pi

Before installing any libraries and dependencies, it's essential to update the Raspberry Pi to ensure you have the latest packages and security fixes. Open a terminal and run the following commands:

```
sudo apt update
sudo apt upgrade -y
```

Installing Python, pip, and necessary libraries

Python comes preinstalled on Raspberry Pi OS, but you may need to install pip (Python package manager) and some other required libraries. Run the following commands:

```
sudo apt install -y python3-pip
sudo pip3 install RPi.GPIO
sudo pip3 install Adafruit_DHT
sudo pip3 install requests
sudo apt-get install libopenjp2-7 libtiff5
```

Installing Waveshare e-Paper library

To install the Waveshare e-Paper library, clone the GitHub repository and install the required packages. Run the following commands:

```
git clone https://github.com/waveshare/e-Paper.git
cd e-Paper/RaspberryPi_JetsonNano/python/
sudo python3 setup.py install
```

B. Developing the Python script

Importing necessary libraries and configuring GPIO pins

Import the required libraries at the beginning of your Python script. Then, configure the GPIO pins according to your setup and hardware connections.

```
import time
import requests
import json
import RPi.GPIO as GPIO
import Adafruit_DHT
from PIL import Image, ImageDraw, ImageFont
from waveshare_epd import epd7in5_V2
import datetime
```

Defining a function to get the weather forecast from the API

Create a function to fetch weather data from the OpenWeatherMap API. Ensure you have a valid API key, and replace "PUT YOUR ID HERE" with your API key.

```
def weather():
  global weather_response
  global forecast_response
  weather_url=
f"https://api.openweathermap.org/data/2.5/weather?q={CITY_NAME}&appid={WEATHER_API}"
  weather_response = requests.get(weather_url).json()
```

Defining a function to update the e-Paper Display

Create a function called update_screen() that updates the e-Paper display with the latest data, including time, date, and weather information.

```
def update_screen():
  global weather_response
  weather()
  # Create a new image, draw text, and display the image on the e-Paper
```

Creating a function to handle the button press

Define a function called button_pressed_callback() that will be called when the button is pressed. This function should call the refresh_screen() function to update the display.

```
def button_pressed_callback(channel):
    update_screen()
```

Setting up the main loop

In the main loop, add an event listener for the button press and keep the script running with a time.sleep() function.

```
GPIO.add_event_detect(BUTTON_PIN,GPIO.FALLING,
callback=button_pressed_callback, bouncetime=200)
try:
    update_screen()
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    # Clean up resources when the script is interrupted
    epd.init()
    epd.Clear()
    epd7in5_V2.epdconfig.module_exit()
    GPIO.cleanup()
```

Now, your Raspberry Pi is set up with the required software and libraries, and the Python script is ready to run.

V. CUSTOMIZING THE DISPLAY

A. Adjusting the text, font size, and arrangement

To modify the text displayed on the screen, edit the content within the draw.text() functions in the Python script.

To change the font size, adjust the numerical value in the ImageFont.truetype() function when defining the font. You can create multiple font sizes using different variables.

To rearrange the text on the screen, adjust the x and y coordinates within the draw.text() functions. These coordinates define the position of the top-left corner of the text block.

B. Adding images or shapes to the display

To add an image, use the Image.open() function to load the image file and the image.paste() function to place it on the display canvas.

To draw shapes (such as lines, rectangles, or circles), use the drawing functions provided by the ImageDraw module, like draw.line(), draw.rectangle(), or draw.ellipse().

VI. TESTING AND TROUBLESHOOTING

A. Running the script

Save the Python script on the Raspberry Pi and make sure all required libraries are installed.

Run the script using the command python3 script_name.py in the terminal, replacing script_name.py with the name of your Python script.

B. Identifying and resolving potential issues

If the script encounters any errors, carefully read the error messages in the terminal to identify the problem.

Check your code for syntax errors, incorrect variable names, or incorrect pin assignments.

Ensure the hardware connections are secure and correct, and that the Raspberry Pi and display are properly powered.

VII. CONCLUSION

A. **Summary of the project**

This Raspberry Pi project utilizes a DHT22 sensor, a push button, an LED, and a Waveshare e-Paper display to create an indoor temperature and humidity monitoring system with weather forecast information. The Python script queries a weather API for data and updates the e-Paper display with the current time, date, indoor and outdoor temperature, humidity, and AC status. The LED indicates when the AC is active, and the button triggers an update of the displayed information.

B. Potential improvements and future work

1. Incorporate additional sensors, such as air quality or light sensors, to provide more comprehensive environmental monitoring.
2. Use a smaller or larger e-Paper display to fit specific use cases or design preferences.
3. Expand the display capabilities by incorporating graphs, charts, or more detailed weather forecast information.
4. Optimize the code to reduce power consumption, enabling battery-powered operation.
5. Add wireless connectivity, such as Wi-Fi or Bluetooth, to remotely monitor and control the system.

III. REFERENCES

A. Relevant resources and documentation

- Raspberry Pi official website: https://www.raspberrypi.org/
- Waveshare e-Paper official documentation: https://www.waveshare.com/wiki/7.5inch_e-Paper_HAT
- DHT22 sensor documentation: https://learn.adafruit.com/dht/connecting-to-a-dhtxx-sensor
- OpenWeatherMap API documentation: https://openweathermap.org/api

B. Acknowledgments and attributions

- Raspberry Pi Foundation for creating and maintaining the Raspberry Pi platform.
- Waveshare for producing the e-Paper display and providing support.
- Adafruit for the DHT22 sensor and the corresponding library.
- OpenWeatherMap for providing the weather API used in this project.