

Temperature and Humidity Station

Build Your Own

Contents

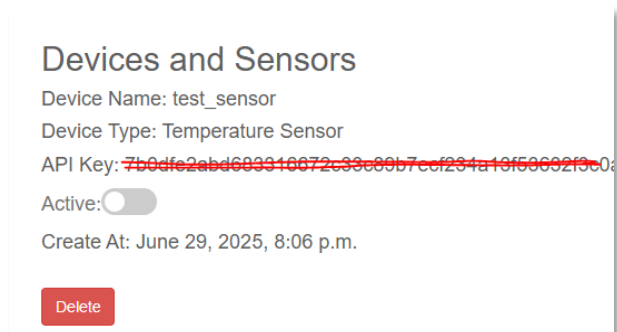
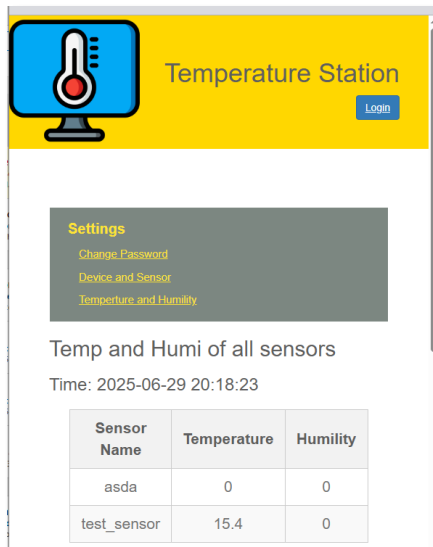
Introduction	2
ESP32 Wiring - LCD and Temp Sensor	2
Device Model	2
Temperature Sensor Wiring	3
LCD Wiring.....	3
Flash MircroPython firmware to ESP32	4
Python code for ESP32	5
Setup Django Webserver	5
We are using Ubuntu server VM or Raspberry Pi OS and just need to enable SSH with full access ..	5
Install Django Framework and necessary Apps.....	6
Create virtual environment and install Django	6
Create web applications and install dependent libraries.....	7
Copy the source code files and install.....	7
Register / create API key for sensor device	8
Update the main.py in Thonny with the Device Name and API Key to the ESP32 device	9

Introduction

There are 2 parts in this project. The first one is to build a LCD display clock using ESP32, 2x16 LCD module and temperature sensors. It will show the current date time and the sensor information. You can build as many as display clocks and put that round your building with different type of IoT sensors as you design.



Part 2 is to build a server that collect all the sensor data and present the latest sensor information such as temperature and humidity on a webserver. The ESP32 sensors will send data every 2 mins using API calls. API Keys are stored on the server. User can disable / enable any key resulting allow or reject the data from individual sensor.



ESP32 Wiring - LCD and Temp Sensor

Device Model

- ESP32

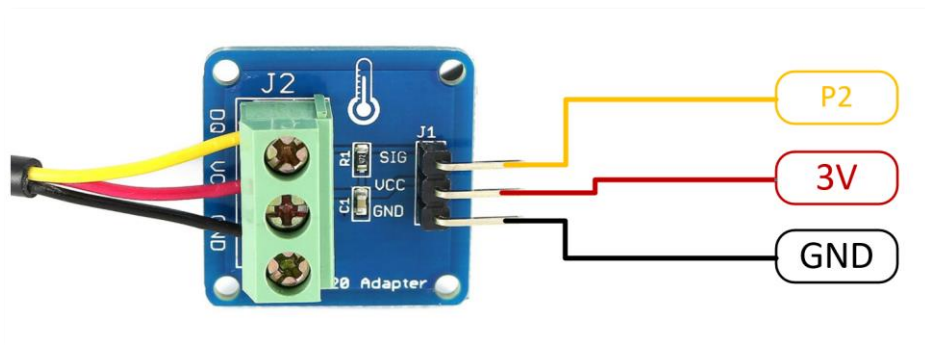
CP2102 WIFI Bluetooth Module 38 pin

- LCD Display
LCD1602 1602 LCD Module 16x2 Character LCD Display
IIC I2C Serial Interface Adapter Module
- Temperature Sensor
DS18B20 with adaptor

Temperature Sensor Wiring

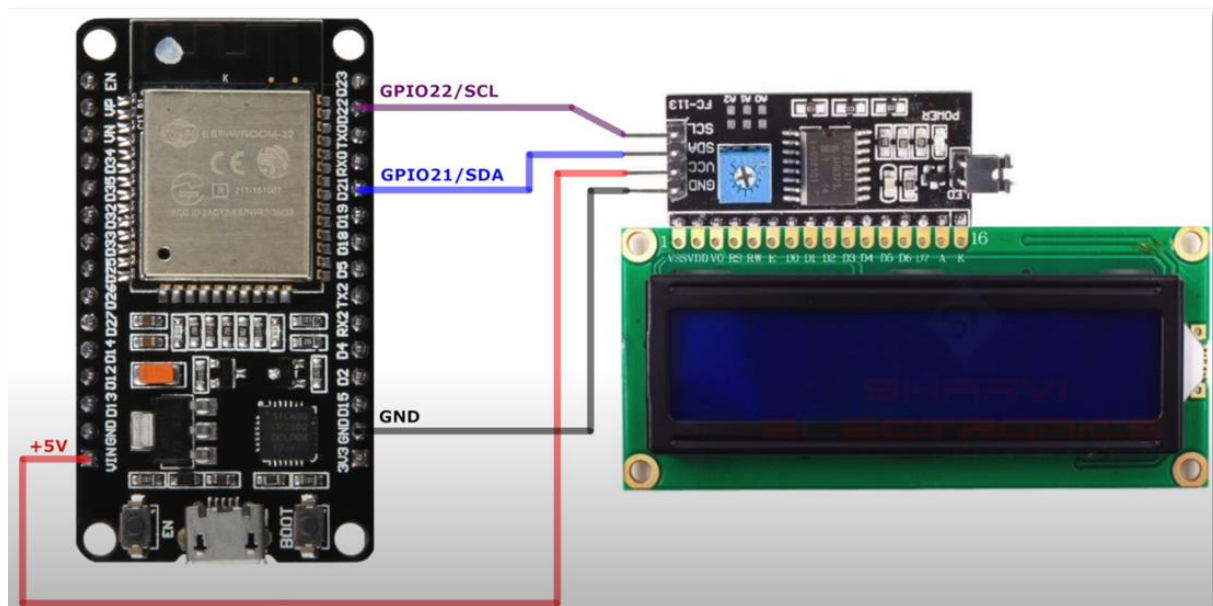
DS18B20 can work on either 3v or 5v. I use 3v to reserve the 5v pin for LCD display

VCC <--> 3V, GND <--> Ground, SIG <--> GPIO 2



LCD Wiring

VCC <--> 5V, GND <--> Ground, SDA <--> GPIO21, SCL <--> GPIO22



Flash MicroPython firmware to ESP32

- Version: ESP32_GENERIC-20250415-v1.25.0.bin

This version support https which required for API call or error with "ImportError: no module named 'ssl'"

- Set ESP32 to download model

hold down the BOOT button (often labeled as GPIO0) and then briefly press and release the EN (enable/reset) button

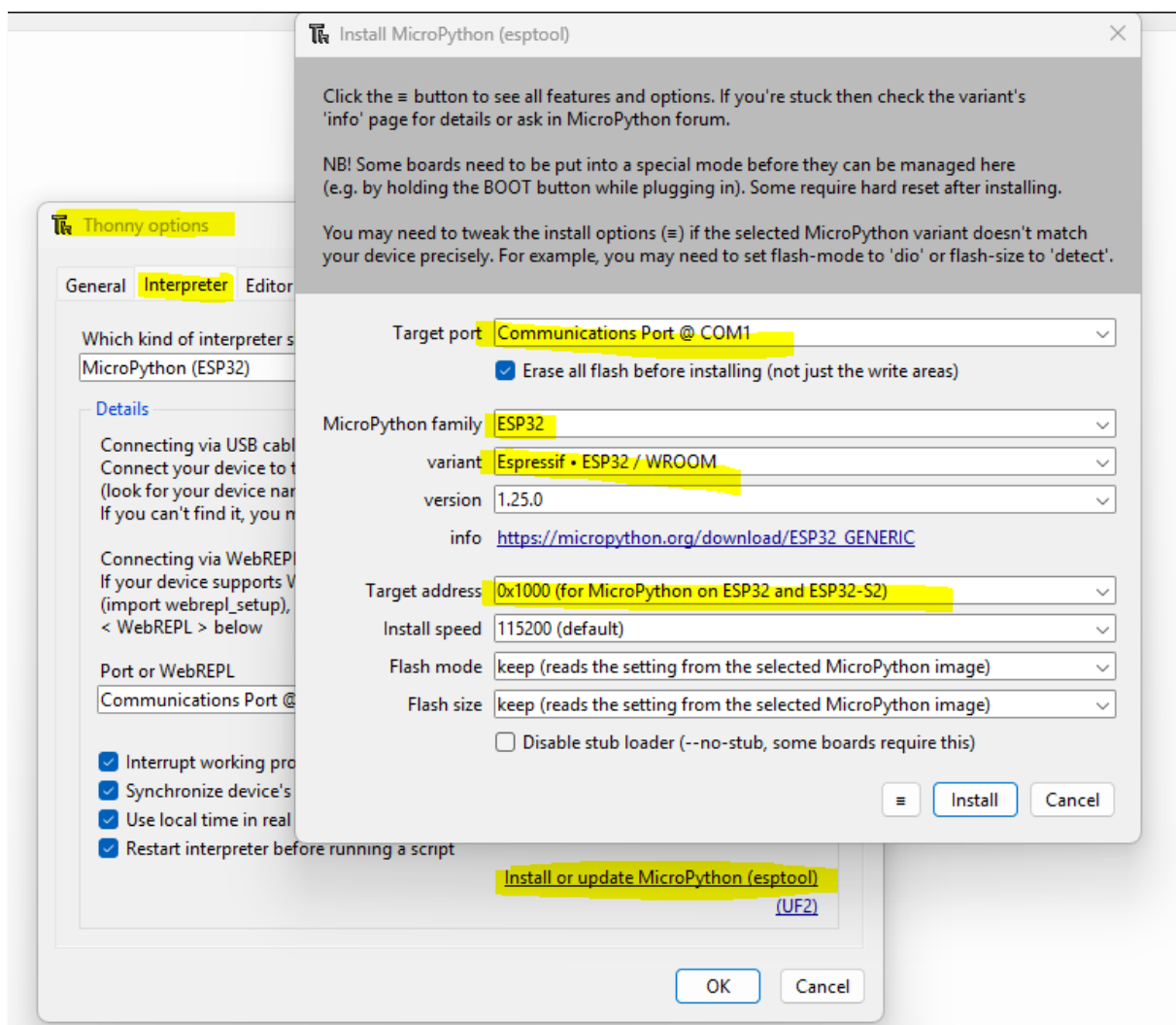
- Using windows with python installed to flash firmware

```
python -m esptool --chip esp32 --port COM3 erase_flash
```

```
python -m esptool --chip esp32 --port COM3 --baud 460800 write_flash -z 0x1000  
C:\Path\To\ESP32_GENERIC-20250415-v1.25.0.bin
```

- Using Thonny application to flash firmware (Optional)

Google it. It is easy. You will need Thonny to copy all python code to ESP32 anyway.



Python code for ESP32

- Download and install Thonny for Windows (free Tools)
- Connect the ESP32 in Thonny

Tools → Thonny Options → Interpreter → Select MicroPython (ES32) → Select the Communication Port @ COM?? → OK

- Upload below file to ESP32 including folder

/lib/i2c_lcd.py

/lib/lcd_api.py

/lib/ntptime.py

/main.py

/display_LCD1602.py

/sensor_DS18B20.py

/local_time.py

/upload.py

- Edit the main.py file to meet your need

Wifi settings

Server IP, Device name and API Key

Day Light saving settings

Timezone settings

LCD pin numbers

Temperature ping number

Setup Django Webserver

We are using Ubuntu server VM or Raspberry Pi OS and just need to enable SSH with full access

- Ubuntu server 24.04.01 (minimal footprint install / 2 CPU / 2G RAM / 20Gb Disk)
 - If the system running on VM. Make sure all disk space has been used.

sudo lvextend -l +100%FREE /dev/ubuntu-vg/ubuntu-lv

sudo resize2fs /dev/mapper/ubuntu--vg-ubuntu—lv

```
user@filesafer:~$ df -h
Filesystem                Size      Used Avail Use% Mounted on
tmpfs                     192M        976K  192M   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 18G       4.0G    13G   24% /
tmpfs                     960M         0   960M   0% /dev/shm
tmpfs                     5.0M         0    5.0M   0% /run/lock
/dev/sda2                 1.8G       99M    1.6G   7% /boot
tmpfs                     192M       12K    192M   1% /run/user/1000
```

- Raspberry Pi 3 or 4B
 - Install Raspberry Pi OS Lite (no Desktop needed)

Install Django Framework and necessary Apps

sudo apt-get update

sudo apt-get install python3-django python3-pip python3-venv

- If you are using Ubuntu server 20.04 with minimal footprint installation

sudo apt-get install nano cron iputils-ping

sudo nano /etc/resolv.conf

add line: **nameserver 8.8.8.8**

Create virtual environment and install Django

cd /

sudo mkdir Automation

sudo chmod 777 Automation

python3 -m venv Automation

cd Automation

source bin/activate

pip3 install Django

```

user@filesafer:~$ cd /
user@filesafer:/$ sudo mkdir Automation
user@filesafer:/$ sudo chmod 777 Automation/
user@filesafer:/$ python3 -m venv Automation
user@filesafer:/$ cd Automation
source bin/activate
pip3 install Django
Collecting Django
  Downloading django-5.2.2-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref>=3.8.1 (from Django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from Django)
  Downloading sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
  Downloading django-5.2.2-py3-none-any.whl (8.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 6.3 MB/s eta 0:00:00
  Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
  Downloading sqlparse-0.5.3-py3-none-any.whl (44 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.4/44.4 kB 3.8 MB/s eta 0:00:00
Installing collected packages: sqlparse, asgiref, Django
Successfully installed Django-5.2.2 asgiref-3.8.1 sqlparse-0.5.3
(Automation) user@filesafer:/Automation$
(Automation) user@filesafer:/Automation$

```

Create web applications and install dependent libraries

- Perform below commands under directory /Automation within the virtual environment

django-admin startproject Thermometer .

python3 manage.py startapp Login

python3 manage.py startapp TempLog

pip3 install ipcalc six yt_dlp django-sslserver

- Above process creates below folder structure

Thermometer

TempLog

Login

Copy the source code files and install

We will use sftp to copy the source files into created directories from above

If you are using windows computer you can use MoxaXterm free application to do this. Linux user can perform this task natively using Terminal.

- Start from the directory where you have downloaded the Django Source Code. SFTP to the server

cd /Automation

put -R *

python3 manage.py makemigrations

python3 manage.py migrate

- Create superuser for the Web App. This account is for you to login to the Web GUI

python3 manage.py createsuperuser

```
(Automation) user@filesafer:/Automation$ python3 manage.py createsuperuser
Username (leave blank to use 'user'): user
Email address:
Password:
Password (again):
Superuser created successfully.
(Automation) user@filesafer:/Automation$
```

- Setup application auto-start (My ubuntu user account called "user" where used in @reboot code below)

sudo nano /etc/crontab

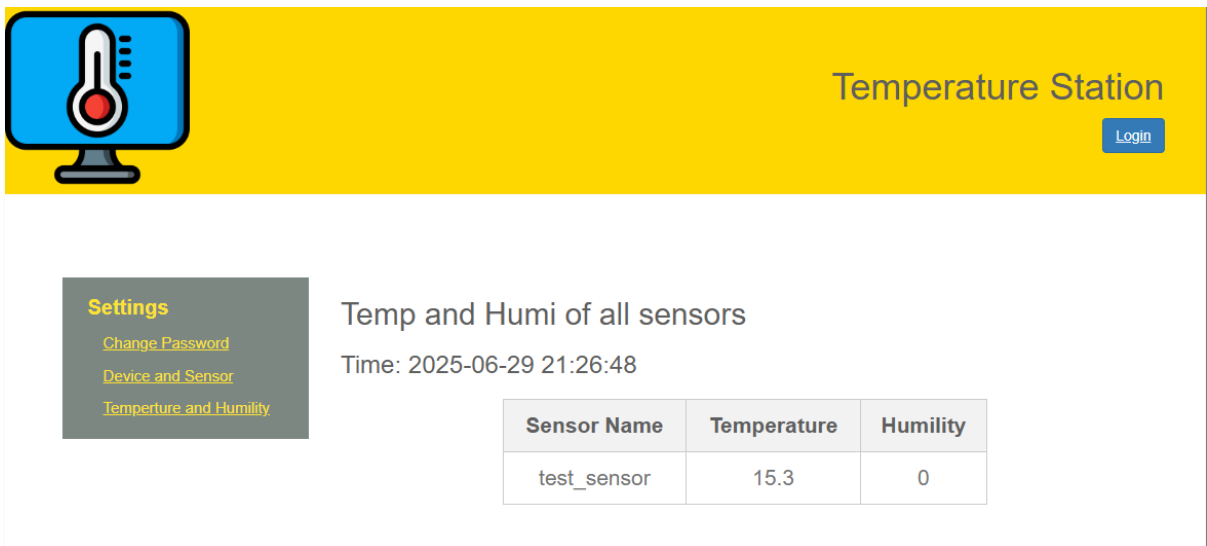
add below line to the bottom and save / exit

@reboot user /bin/bash -c "/Automation/run.sh"

- Reboot the server and access the web app via browser

sudo reboot now

https://<your-server-ip-address>:8000



The image shows the web interface of a 'Temperature Station'. The header is yellow with a blue thermometer icon on the left and the title 'Temperature Station' on the right, with a 'Login' button. The main content area is white. On the left, there is a 'Settings' sidebar with links: 'Change Password', 'Device and Sensor', and 'Temperture and Humility'. The main content displays 'Temp and Humi of all sensors' and the current time 'Time: 2025-06-29 21:26:48'. Below this is a table with sensor data.

Sensor Name	Temperature	Humility
test_sensor	15.3	0

Register / create API key for sensor device

Device and Sensor → Register New Sensor → Put in a name for Device Name → Select the sensor type → Click Register

Settings

- [Change Password](#)
- [Device and Sensor](#)**
- [Temperture and Humility](#)

Devices and Sensors

Sensor Name:

Sensor Type:

Is Active: ☒

[Register](#)

Update the main.py in Thonny with the Device Name and API Key to the ESP32 device

- In order to allow the ESP32 to send data to the server

Devices and Sensors

Device Name: **LivingRoom_1**

Device Type: Temperature Sensor

API Key: **514734ba937900c2df92c165f5b0f109566b8ba82f15272ea537fe073fde9ae**

Active: ☒

Create At: June 29, 2025, 9:33 p.m.