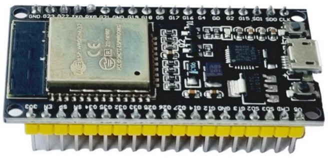
ESP32 Environmental Monitoring Station

A compact ESP32-based project that measures temperature, humidity, pressure, and gas levels using DHT22, BMP280, and MQ sensors.

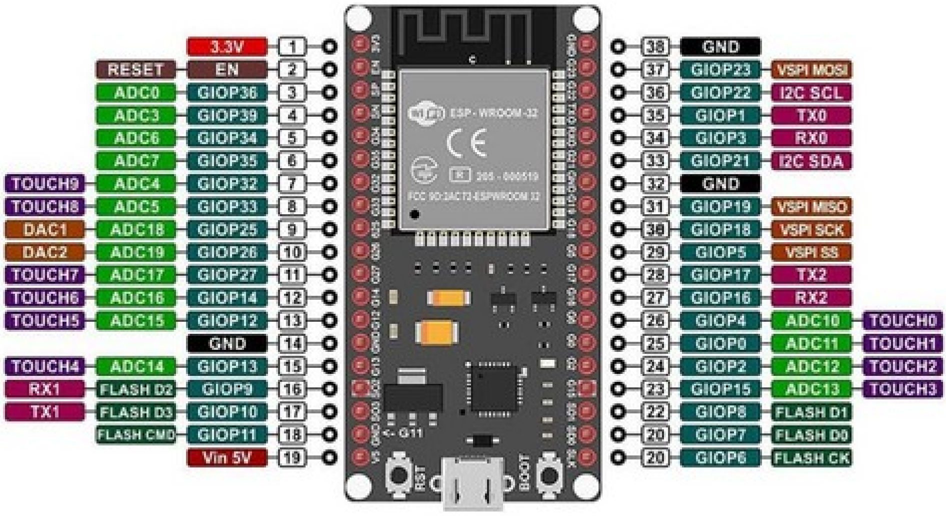
Components used →

1. ESP32 (38 Pin) WiFi + Bluetooth NodeMCU-32 Development Board

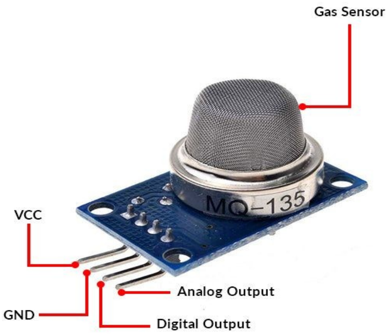


Features	**ESP32**
Micro-controller	Xtensa Dual-Core 32-bit LX6 with 600 DMIPS
Frequency (MHz)	160 to 240
Bluetooth	Bluetooth 4.2 and BLE
SRAM	Yes
GPIO Pins	38 (32 Usable Pins)
PWM	16 Channels
ADC	12-bit
TWAI	Yes
Power Consumption	Approx. 80~90mA
Working Temperature	-40°C to 125°C

Voltage needed: 3.3V logic, powered via USB (5V input)

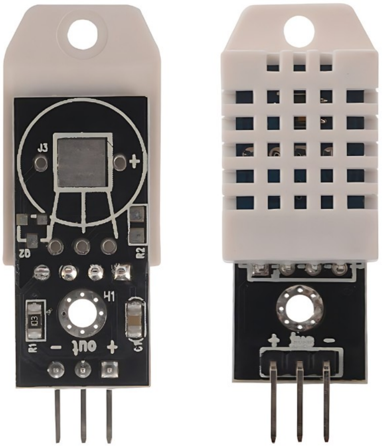


2. MQ-135 Gas Sensor Module For NH3, Alcohol, benzene, smoke , CO2 Detector Module



- This gas sensor works between 2.5V to 5.0V.
- Uses around 150mA of current while running.
- It can detect gases like NH3, NOx, CO2, alcohol, benzene, and smoke.
- When powered at 5V, it gives out a digital signal from 0V to 5V in TTL logic.
- it also provides an analog output in the same voltage range. Typically, it operates at 5V.

3. DHT22 Humidity and Temperature Sensor Module (AM2302)



DHT22 Temp/Humidity Sensor: 3.3V or 5V (recommended 3.3V with ESP32)

Measuring range Temperature -40 to 80 °C

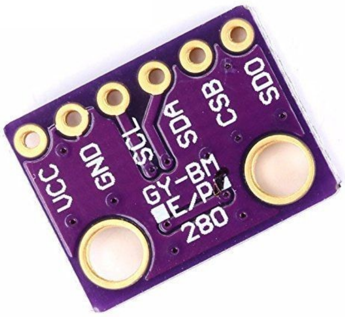
Measuring Range Humidity 0 to 99.9%RH

Resolution Temperature 0.1 °C

Resolution Humidity 0.1%RH

4. BMP280 Sensor Module

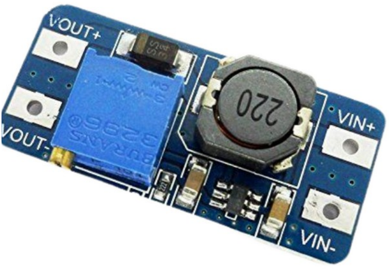
Operating Voltage	1.71V to 3.6V
Operating Temperature	-40 to +85 deg. Celsius
Peak current	1.12mA
Operating Pressure	300 hPa to 1100 hPa



5. Witty Fox 3.7V 2600mAh Li-Ion Battery

Voltage	3.7V
Battery Capacity	2600 mAh
Battery Type	Lithium-Ion
Battery Dimension (Height x Diameter)	66mm x 22mm

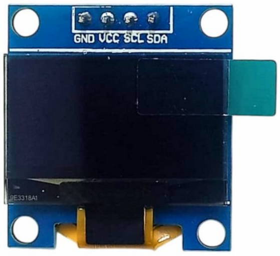
6. MT3608 DC-DC Boost Module (2V-24V)



The MT3608 2A Max DC-DC Step Up Power Module Booster Power Module is a budget-friendly module that enables the user to step-up a 2 to 24V input voltage to a 5 to 28V output at a maximum of 2A.

Model	MT3608 DC-DC Boost Module
Input Voltage	2 to 24V
Max. Output Current (A)	2
Efficiency	93%

7. 0.96 Inch OLED Display Module SPI/I2C 4pin Blue Color



Display Size	0.96 inch
Display Type	OLED Display
Resolution	128 x 64 Pixels
Driving Voltage	3.3-5V
Interface Type	IIC

8. Other components: Touch sensor, Switch, Buck Converter

Software Setup

- **Language:** MicroPython / CircuitPython
- **IDE:** Thonny
- **Libraries Required:**
    - `dht` (for DHT22)
    - `bmp280` (for BMP280)
    - `ssd1306` (for OLED)
    - `machine`, `time`, `network` (ESP32 basics)

Pin Connections (I2C & GPIO)

Component	Pin on Component	Connects to ESP32 GPIO	Notes
BMP280	VCC	3V3	3.3 V supply
	GND	GND	Common ground
	SCL	GPIO 22	I2C clock
	SDA	GPIO 21	I2C data
SSD1306 OLED	VCC	3V3	3.3 V supply
	GND	GND	Common ground
	SCL	GPIO 22	I2C clock (shared with BMP280)
	SDA	GPIO 21	I2C data (shared with BMP280)
DHT22	VCC	3V3	3.3 V supply
	GND	GND	Common ground
	DATA	GPIO 4	10 kΩ pull-up to 3.3 V recommended
MQ Gas Sensor	VCC	3V3	3.3 V supply (warms up slower than at 5 V)
	GND	GND	Common ground
	DO	GPIO 15	Digital alarm signal
	AO	GPIO 34	

Test Code Examples

```
from machine import Pin, I2C, ADC
import ssd1306
import dht
import time
from bmp280 import BMP280

# ==== I2C Setup ====
i2c = I2C(0, scl=Pin(22), sda=Pin(21))
devices = i2c.scan()
print("I2C scan:", devices)
oled = ssd1306.SSD1306_I2C(128, 64, i2c)

# ==== Sensors ====
dht_sensor = dht.DHT22(Pin(19))     # DHT22 module on GPIO19
mq135 = ADC(Pin(34))           # MQ135 analog output
mq135.atten(ADC.ATTN_11DB)        # 0–3.3V range

# BMP280 setup
try:
    bmp = BMP280(i2c, addr=0x76)
    bmp_ok = True
except Exception as e:
    print("BMP280 init failed:", e)
    bmp = None
    bmp_ok = False

# ==== TTP223 Touch Switch ====
touch = Pin(4, Pin.IN, Pin.PULL_DOWN)  # GPIO4
window = 0
last_touch_time = 0
debounce_ms = 200  # 200ms debounce

# ==== Helper Functions ====
def mq_quality(value):
    if value < 800:
        return "Excellent"
    elif value < 1500:
        return "Good"
    elif value < 2500:
        return "Moderate"
    elif value < 3200:
        return "Bad"
    else:
        return "Worst"

def read_sensors():
    # DHT22
    try:
        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
        if hum < 0 or hum > 100:  # ignore invalid readings
            hum = None
    except:
        temp, hum = None, None

    # BMP280
    if bmp_ok:
        try:
            pres_pa = bmp.pressure
            pres_hpa = pres_pa / 100
            if pres_hpa > 1020:
                pres_desc = "High"
            elif pres_hpa < 1000:
                pres_desc = "Low"
            else:
                pres_desc = "Normal"
        except:
            pres_hpa = None
            pres_desc = "ERR"
    else:
        pres_hpa = None
        pres_desc = "ERR"

    # MQ135
    try:
        gas = mq135.read()          # raw ADC
        gas_label = mq_quality(gas)  # qualitative label
    except:
        gas = None
        gas_label = "ERR"

    return temp, hum, pres_hpa, pres_desc, gas, gas_label

# ==== Main Loop with 4 windows ====
while True:
    try:
        # Touch to cycle windows
        t = time.ticks_ms()
        touch_state = touch.value()
        if touch_state == 1 and t - last_touch_time > debounce_ms:
            window = (window + 1) % 4
            print("Window changed:", window)
            last_touch_time = t
        # Read sensors
        temp, hum, pres_hpa, pres_desc, gas, gas_label = read_sensors()

        # Display
        oled.fill(0)
        if window == 0:
            oled.text("Temp & Hum", 0, 0)
            oled.text("Temp: {}C".format(int(temp) if temp else "--"), 0, 16)
            oled.text("Hum: {}%".format(int(hum) if hum is not None else "--"), 0, 32)
        elif window == 1:
            oled.text("Pressure", 0, 0)
            oled.text("Status: {}".format(pres_desc), 0, 16)
            oled.text("Value: {:.1f} hPa".format(pres_hpa) if pres_hpa else "--", 0, 32)
        elif window == 2:
            oled.text("Air Quality", 0, 0)
            oled.text("MQ: {}".format(gas_label), 0, 16)
            oled.text("Raw: {}".format(gas if gas else "--"), 0, 32)
        elif window == 3:
            oled.text("DETAIL VIEW", 0, 0)
            oled.text("Temp: {}C Hum: {}%".format(int(temp) if temp else "--", int(hum) if hum is
not None else "--"), 0, 12)
            oled.text("Pres: {:.1f} hPa".format(pres_hpa) if pres_hpa else "--", 0, 24)
            oled.text("Gas: {} / {}".format(gas if gas else "--", gas_label), 0, 36)
        oled.show()

        time.sleep(1)

    except Exception as e:
        print("Loop error:", e)
        time.sleep(2)
```