<https://randomnerdtutorials.com/esp32-sim800l-send-text-messages-sms/>

<https://www.youtube.com/watch?v=VM-viZHZvAs>

// SIM card PIN (leave empty, if not defined)

const char simPIN[] = "";

#define SMS\_TARGET "+351XXXXXXXXX"

bool smsSent = false;

// Configure TinyGSM library

#define TINY\_GSM\_MODEM\_SIM800 // Modem is SIM800

#define TINY\_GSM\_RX\_BUFFER 1024 // Set RX buffer to 1Kb

#include <Wire.h>

#include <TinyGsmClient.h>

#include <OneWire.h>

#include <DallasTemperature.h>

// Setup a oneWire instance to communicate with any OneWire devices

OneWire oneWire(oneWireBus);

// Pass our oneWire reference to Dallas Temperature sensor

DallasTemperature sensors(&oneWire);

float temperatureThreshold = 28.0;

// GPIO where the DS18B20 is connected to

const int oneWireBus = 13;

// TTGO T-Call pins

#define MODEM\_RST 5

#define MODEM\_PWKEY 4

#define MODEM\_POWER\_ON 23

#define MODEM\_TX 27

#define MODEM\_RX 26

#define I2C\_SDA 21

#define I2C\_SCL 22

// Set serial for debug console (to Serial Monitor, default speed 115200)

#define SerialMon Serial

// Set serial for AT commands (to SIM800 module)

#define SerialAT Serial1

// Define the serial console for debug prints, if needed

//#define DUMP\_AT\_COMMANDS

#ifdef DUMP\_AT\_COMMANDS

#include <StreamDebugger.h>

StreamDebugger debugger(SerialAT, SerialMon);

TinyGsm modem(debugger);

#else

TinyGsm modem(SerialAT);

#endif

#define IP5306\_ADDR 0x75

#define IP5306\_REG\_SYS\_CTL0 0x00

bool setPowerBoostKeepOn(int en){

Wire.beginTransmission(IP5306\_ADDR);

Wire.write(IP5306\_REG\_SYS\_CTL0);

if (en) {

Wire.write(0x37); // Set bit1: 1 enable 0 disable boost keep on

} else {

Wire.write(0x35); // 0x37 is default reg value

}

return Wire.endTransmission() == 0;

}

void setup() {

// Set console baud rate

SerialMon.begin(115200);

// Keep power when running from battery

Wire.begin(I2C\_SDA, I2C\_SCL);

bool isOk = setPowerBoostKeepOn(1);

SerialMon.println(String("IP5306 KeepOn ") + (isOk ? "OK" : "FAIL"));

// Set modem reset, enable, power pins

pinMode(MODEM\_PWKEY, OUTPUT);

pinMode(MODEM\_RST, OUTPUT);

pinMode(MODEM\_POWER\_ON, OUTPUT);

digitalWrite(MODEM\_PWKEY, LOW);

digitalWrite(MODEM\_RST, HIGH);

digitalWrite(MODEM\_POWER\_ON, HIGH);

// Set GSM module baud rate and UART pins

SerialAT.begin(115200, SERIAL\_8N1, MODEM\_RX, MODEM\_TX);

delay(3000);

// Restart SIM800 module, it takes quite some time

// To skip it, call init() instead of restart()

SerialMon.println("Initializing modem...");

modem.restart();

// use modem.init() if you don't need the complete restart

// Unlock your SIM card with a PIN if needed

if (strlen(simPIN) && modem.getSimStatus() != 3 ) {

modem.simUnlock(simPIN);

}

// Start the DS18B20 sensor

sensors.begin();

}

void loop() {

sensors.requestTemperatures();

// Temperature in Celsius degrees

float temperature = sensors.getTempCByIndex(0);

SerialMon.print(temperature);

SerialMon.println("\*C");

if((temperature > temperatureThreshold) && !smsSent){

String smsMessage = String("Temperature above threshold: ") +

String(temperature) + String("C");

if(modem.sendSMS(SMS\_TARGET, smsMessage)){

SerialMon.println(smsMessage);

smsSent = true;

}

else{

SerialMon.println("SMS failed to send");

}

}

delay(5000);

}

**How the Code Works**

In the previous example, we’ve already explained how to initialize the SIM800L and all the required configurations. So, let’s skip to the relevant parts for this project.

First, type your SIM card PIN. If it’s not defined, you can leave this variable empty.

const char simPIN[] = "";

Then, add the phone number you want to send the SMS to. The number should be in international format, otherwise it won’t work.

#define SMS\_TARGET "+351XXXXXXXXXXX"

Define your temperature threshold. We’ve set it to 28 degrees Celsius.

float temperatureThreshold = 28.0;

Create a variable to keep track if an SMS was sent or not.

bool smsSent = false;

The temperature sensor is connected to GPIO 13, but you can use any other GPIO.

const int oneWireBus = 13;

**Related content:** [ESP32 DS18B20 Temperature Sensor with Arduino IDE (Single, Multiple, Web Server)](https://randomnerdtutorials.com/esp32-ds18b20-temperature-arduino-ide/)

In the loop(), get the temperature readings.

sensors.requestTemperatures();

// Temperature in Celsius degrees

float temperature = sensors.getTempCByIndex(0);

SerialMon.print(temperature);

SerialMon.println("\*C");

**Upload the Code**

After inserting the recipient’s phone number and SIM card pin code, upload the sketch to your ESP32.

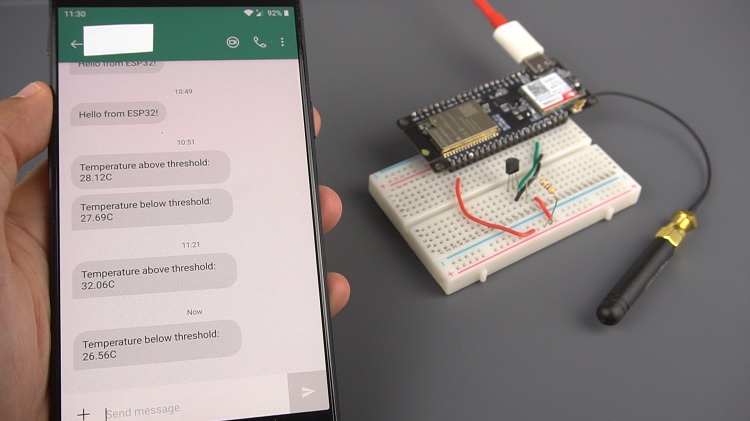
* Go to **Tools**> **Board**and select the**ESP32 Dev module**
* After that, go to **Tools**> **Port**and select the COM port the ESP32 is connected to
* Then, press the **Upload**button

After a few seconds, the code should be successfully uploaded.

You can also open the Serial Monitor at a baud rate of 115200 to see the current sensor readings.

If I put my finger on top of the sensor, the temperature will start increasing.When it goes above 28ºC, it sends an SMS.

When the temperature goes below the threshold, I’ll receive another SMS.



To make this project battery powered, I recommend using [deep sleep mode and wake up the ESP32 every hour](https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/) to check the current temperature, because if you use the code in this project it will drain your battery quickly.

Other articles/projects with the T-Call ESP32 SIM800L board:

* [ESP32 Publish Data to Cloud without Wi-Fi (TTGO T-Call ESP32 SIM800L)](https://randomnerdtutorials.com/esp32-sim800l-publish-data-to-cloud/)
* [TTGO T-Call ESP32 with SIM800L GSM/GPRS (in-depth review)](https://makeradvisor.com/ttgo-t-call-esp32-with-sim800l-gsm-gprs/)

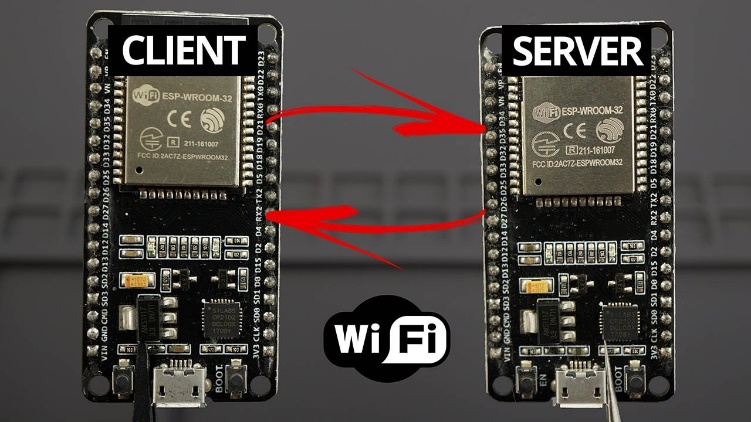
Learn more about the ESP32 with our resources:

* [More ESP32 Projects and Tutorials](https://randomnerdtutorials.com/projects-esp32/)

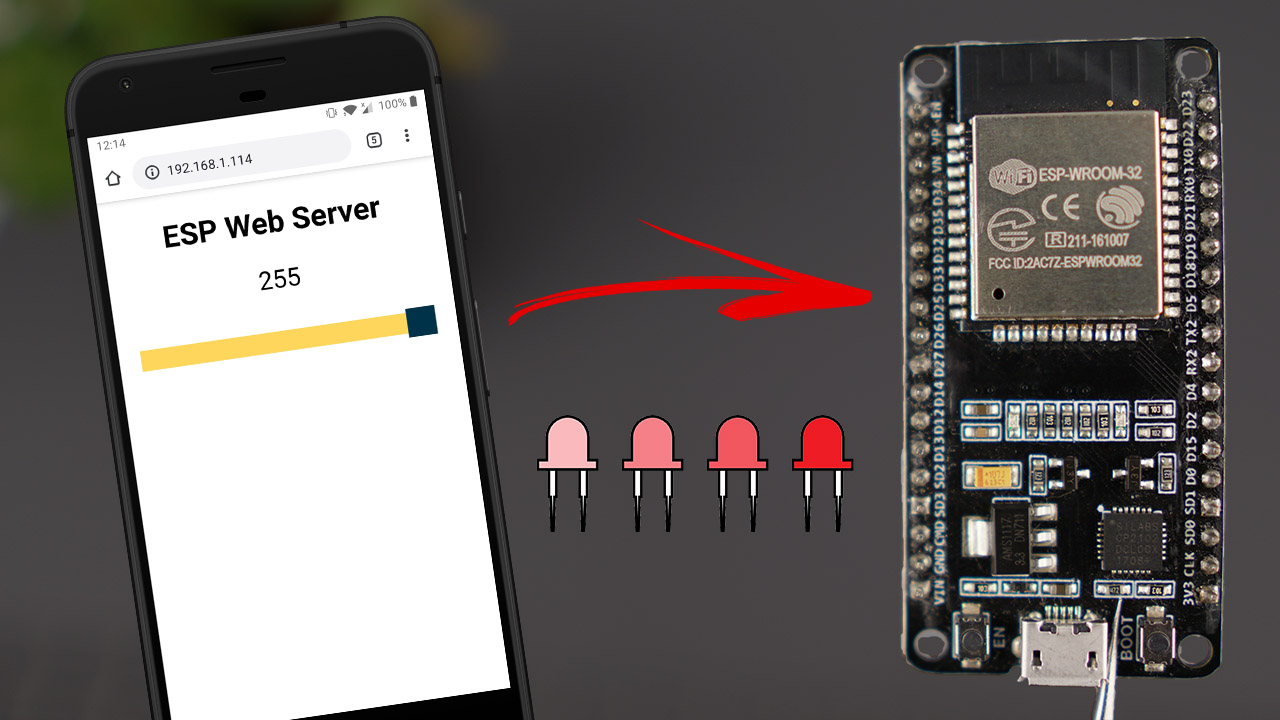
**Recommended Resources**

[**Build a Home Automation System from Scratch »**](https://randomnerdtutorials.com/build-a-home-automation-system-for-100/)With Raspberry Pi, ESP8266, Arduino, and Node-RED.

[**Home Automation using ESP8266 eBook and video course »**](https://randomnerdtutorials.com/home-automation-using-esp8266/)Build IoT and home automation projects.

[[](https://randomnerdtutorials.com/esp32-client-server-wi-fi/)](https://randomnerdtutorials.com/esp32-client-server-wi-fi/)

[ESP32 Client-Server Wi-Fi Communication Between Two Boards](https://randomnerdtutorials.com/esp32-client-server-wi-fi/)

[[](https://randomnerdtutorials.com/esp32-web-server-slider-pwm/)](https://randomnerdtutorials.com/esp32-web-server-slider-pwm/)

[ESP32 Web Server with Slider: Control LED Brightness (PWM)](https://randomnerdtutorials.com/esp32-web-server-slider-pwm/)

[ESP32 OLED Display with Arduino IDE](https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/)

[ESP32/ESP8266 MicroPython Web Server – Control Outputs](https://randomnerdtutorials.com/esp32-esp8266-micropython-web-server/)

[Power ESP8266 with Mains Voltage using Hi-Link HLK-PM03 Converter](https://randomnerdtutorials.com/esp8266-hi-link-hlk-pm03/)

[ESP8266 NodeMCU HTTP POST with Arduino IDE (ThingSpeak and IFTTT.com)](https://randomnerdtutorials.com/esp8266-nodemcu-http-post-ifttt-thingspeak-arduino/)

#include <Wire.h>

#include <TinyGsmClient.h>

void setup() {

int PW\_KEY 4; // pin modem enable

int RST 5; // pin modem reset

int POWER\_ON 23; // pin modem power

pinMode(PW\_KEY, OUTPUT);

pinMode(RST, OUTPUT);

pinMode(POWER\_ON, OUTPUT);

digitalWrite(PW\_KEY, LOW);

digitalWrite(RST, HIGH);

digitalWrite(POWER\_ON, HIGH);

// SDA 21 , SCL 22

// Definido puertos, iniciaize mode

// Set GSM module baud rate and UART pins

int RX = 26;

int RX = 27;

SerialAT.begin(115200, SERIAL\_8N1, RX, TX);

delay(3000);

modem.restart(); // or modem.init();

}

**Enviar SMS**

if(modem.sendSMS("+351XXXXXXXXX", "Enviando mensaje"));

{

Serial.println("Enviando");

}

**Recibir SMS**

String str3="";

while(modem.available()) {

char c = Serial.read();

str3 += c;

}

Top of Form