<https://makeradvisor.com/10-iot-development-boards/>

<https://makeradvisor.com/10-iot-development-boards/>



The TTGO T-Call is an ESP32 development board that besides Wi-Fi and Bluetooth also combines a SIM800 GSM/GPRS module. This can be very handy for projects in remote locations, because you can connect the ESP32 to the internet using your SIM card data plan.

You can also control outputs or monitoring sensors just by sending SMS or making phone calls to your ESP32.

//SIM800 TX is connected to Arduino D8

#define SIM800\_TX\_PIN 8

//SIM800 RX is connected to Arduino D7

#define SIM800\_RX\_PIN 7

* SIM800 5v/4v **↔**Arduino 5v
* SIM800 GND (either one) **↔**Arduino GND
* SIM800 SIM\_TXD **↔**Arduino D8 (read through for the reason)
* SIM800 SIM\_RXD **↔**Arduino D7 (read through for the reason)

Connect module with Arduino as mentioned above or with any changes depending on the module being used. Insert a valid SIM card and connect the Arduino with Arduino IDE.

Simple Serial Communication

Below is the simplest program usable to interact with SIM800 :

#include <SoftwareSerial.h>

//SIM800 TX is connected to Arduino D8

#define SIM800\_TX\_PIN 8

//SIM800 RX is connected to Arduino D7

#define SIM800\_RX\_PIN 7

//Create software serial object to communicate with SIM800

SoftwareSerial serialSIM800(SIM800\_TX\_PIN,SIM800\_RX\_PIN);

void setup() {

  Serial.begin(9600);

  while(!Serial);

  serialSIM800.begin(9600);

  delay(1000);

  Serial.println("Setup Complete!");

}

void loop() {

  //Read SIM800 output and print it in Arduino IDE Serial Monitor

  if(serialSIM800.available()){

    Serial.write(serialSIM800.read());

  }

  //Read Arduino IDE Serial Monitor inputs and send them to SIM800

  if(Serial.available()){

    serialSIM800.write(Serial.read());

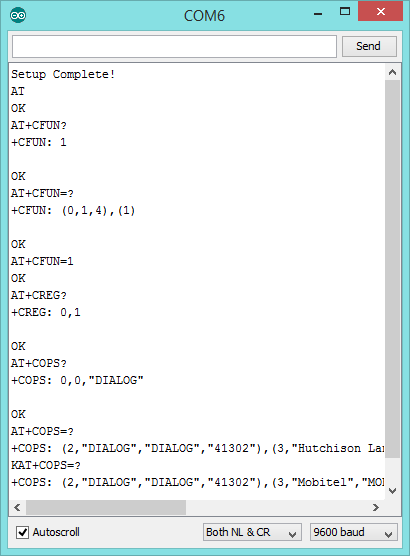
  }

}

Once upload is complete start the Arduino Serial Monitor from Tools menu. Set Baud rate to 9600 and **in drop down left to boud rate selection set “Both NL and CR”**.

Once done, you can freely send [AT commands](https://www.google.com/search?q=sim800+at+commands) to SIM800 and see the output in real time. Few examples :

AT – is to check if interface is working fine.  
AT+CFUN – is used to set phone functionality  
AT+CFUN? – returns currently set value for AT+CFUN  
AT+CFUN=? – returns all possible values that can be set for AT+CFUN (similar to help)  
AT+CFUN=1 – is to sent AT+CFUN to 1 (full functionality)  
AT+CREG? – to get network registration information. stat=1 means you are registered with home network  
AT+COPS? – returns currently registered operator details  
AT+COPS=? – returns all the operators available



Sending SMS with SoftwareSerial

In below code delay of 1 second is used after each command to give necessary time for SIM800 to respond to each command. With this approach it is not possible to clearly identify any ERROR conditions because program will not ready the responses sent. Proper method of doing this is by checking each response against an expected value. This is already handled in most of the Arduino libraries including “Seeeduino” library we’ll be using below. Hence, for this stage, 1 second delay is used for the sake of simplicity.

Note : Replace 07194XXXXX with mobile number SMS should be sent to.

#include <SoftwareSerial.h>

//SIM800 TX is connected to Arduino D8

#define SIM800\_TX\_PIN 8

//SIM800 RX is connected to Arduino D7

#define SIM800\_RX\_PIN 7

//Create software serial object to communicate with SIM800

SoftwareSerial serialSIM800(SIM800\_TX\_PIN,SIM800\_RX\_PIN);

void setup() {

  Serial.begin(9600);

  while(!Serial);

  //Being serial communication with Arduino and SIM800

  serialSIM800.begin(9600);

  delay(1000);

  //Set SMS format to ASCII

  serialSIM800.write("AT+CMGF=1\r\n");

  delay(1000);

  //Send new SMS command and message number

  serialSIM800.write("AT+CMGS=\"07194XXXXX\"\r\n");

  delay(1000);

  //Send SMS content

  serialSIM800.write("TEST");

  delay(1000);

  //Send Ctrl+Z / ESC to denote SMS message is complete

  serialSIM800.write((char)26);

  delay(1000);

  Serial.println("SMS Sent!");

}

void loop() {

}

IM800 Libraries

With a quick [Google search](https://www.google.com/search?q=SIM800+arduino+libraries) you will be able to find several SIM800 Arduino libraries.

After going through source codes of several libraries my selection was “Seeeduino\_GPRS” library which provides basic SIM800 features as well as additional set of GPRS related features.

Sending SMS with Seeeduino Arduino library

Note : Seeeduino library assumes that TX connected to D8 and RX is connected to D7 on Arduino. This is the reason we used relevant pins at first place. If you need to connect SIM800 with any other Arduino pin, you will have to modify the library source (gprs.h) and add a new constructor. Library uses MIT license.

* Clear steps relevant to installing Arduino library is available at : <https://www.arduino.cc/en/Guide/Libraries>
* Seeeduino\_GPRS library is available for download at : <https://github.com/Seeed-Studio/Seeeduino_GPRS>

Once library is installed in Arduino IDE File menu, Examples section you will find “Seeeduino\_GPRS” library and withing examples you will find “GPRS\_SendSMS” example which reads as follows :

/\*

Sketch: GPRS Connect TCP

Function: This sketch is used to test seeeduino GPRS's send SMS func.to make it work,

you should insert SIM card to Seeeduino GPRS and replace the phoneNumber,enjoy it!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

note: the following pins has been used and should not be used for other purposes.

  pin 8   // tx pin

  pin 7   // rx pin

  pin 9   // power key pin

  pin 12  // power status pin

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

created on 2013/12/5, version: 0.1

by lawliet.zou(lawliet.zou@gmail.com)

\*/

#include <gprs.h>;

#include <SoftwareSerial.h>;

GPRS gprs;

void setup() {

  Serial.begin(9600);

  while(!Serial);

  Serial.println("GPRS - Send SMS Test ...");

  gprs.preInit();

  delay(1000);

  while(0 != gprs.init()) {

      delay(1000);

      Serial.print("init error\r\n");

  }

  Serial.println("Init success, start to send SMS message...");

  gprs.sendSMS("07194XXXXX","hello,world"); //define phone number and text

}

void loop() {

  //nothing to do

}

If you go through Seeeduino library you will notice that it is possible to send commands directly for any advanced use cases. For examples there are library methods such as :

* sendCmdAndWaitForResp(const char\* cmd, const char \*expectedResp, unsigned timeout)
* sendCmd(const char\* cmd)
* waitForResp(const char \*resp, unsigned int timeout)

Hence, you could simply correctly rewrite the SMS sending application as below (reinvent the wheel) :

[?](https://www.ayomaonline.com/programming/quickstart-sim800-sim800l-with-arduino/)

|  |  |
| --- | --- |
| 01  02  03  04  05  06  07  08  09  10  11  12  13  14 | if(0 != gprs.sendCmdAndWaitForResp("AT+CMGF=1\r\n", "OK", DEFAULT\_TIMEOUT)) { // Set message mode to ASCII      ERROR("ERROR:CMGF");      return;  }  delay(500);  if(0 != gprs.sendCmdAndWaitForResp("AT+CMGS=\"07194XXXXX\"\r\n","&gt;",DEFAULT\_TIMEOUT)) {      ERROR("ERROR:CMGS");      return;  }  delay(1000);  gprs.serialSIM800.write(data);  delay(500);  gprs.serialSIM800.write((char)26);  return; |

= = = = = = =

#### ****Qué es TTGO T-Call?****

TTGO T-Call es una tarjeta de desarrollo basada en el ESP32 que integra el módulo SIM800L GSM / GPRS de igual manera cuenta con interfaz para conectar y cargar baterías de Lipo de hasta 500mA. Esta tarjeta de desarrollo esta orientada para soluciones de Internet de las cosas (IoT).

#### ****¿Para qué sirve TTGO T-Call?****

Sirve para desarrollar prototipos Wifi y Bluetooth relacionados al Internet de las cosas “IOT” con la ventaja de recibir y enviar mensajes de textos SMS cómo realizar llamadas, de igual manera esta tarjeta permite conectar al ESP32 a internet usando una SIM card con plan de datos.

Contamos con una amplia variedad de [dispositivos IoT](https://unitelectronic.wpengine.com/categoria-producto/tarjetas_desarrollo/serie_esp/) que podrán ayudarte a realizar más fácil tus proyectos.

### **ESPECIFICACIONES Y CARACTERÍSTICAS**

* Modelo: T-Call V1.3 20190610
* Incluye:
  + 1 pieza – Antena flexible FPC 40x15mm con conector U.FL/Ipex
  + 2 piezas – Tira header macho de 21 pines
  + 1 pieza – Conector cable JST PH de 2mm
* La placa cuenta con:
  + ESP32-WROVER-B
  + SIM800L
  + USB a TTL: CP2104
  + USB: Tipo C
  + Porta Tarjeta SIM: Compatible Nano SIM
  + Conector JST de 1.25 mm para cargar batería de Lipo de 3.7V
  + Conector para antena IPEX/IPX SMD
  + Botón de reinicio
  + Interfaz: UART, SPI, SDIO, I2C, PWM, PWM, I2S, IRGPIO, sensor táctil de condensador, ADC, DAC.
* Características eléctricas:
  + 5V 1A
  + Corriente de carga para batería Lipo: 500mA
  + Corriente de trabajo y en sueño: En trabajo 70 mA y en sueño 300uA
  + Rango de temperatura: -40 ℃ ~ + 85 ℃
* Especificación de software
  + Modo Wi-Fi: Estación / SoftAP / SoftAP + Estación / P2P
  + Mecanismo de seguridad: WPA / WPA2 / WPA2-Enterprise / WPS
  + Tipo de cifrado: AES / RSA / ECC / SHA
  + Desarrollo de software: Soporte de desarrollo de servidor en la nube / SDK para el desarrollo de firmware del usuario
  + Protocolo de red: IPv4, IPv6, SSL, TCP / UDP / HTTP / FTP / MQTT
  + Configuración de usuario: Instrucciones AT +, servidor en la nube, android / iOSapp
  + OS: FreeRTOS
* Especificaciones inalambricas
* WIFI:
  + Estándar: FCC / CE-RED / IC / TELEC / KCC / SRRC / NCC
  + Protocolo: 802.11 b / g / n (802.11n, velocidad de hasta 150Mbps) Polimerización A-MPDU y A-MSDU, soporte 0.4μS Intervalo de protección
  + Rango de frecuencia: 2.4GHz ~ 2.5GHz (2400M ~ 2483.5M)
  + Potencia de transmisión: 22 dBm
  + Distancia de comunicación: 300 m
* Bluetooth:
  + Protocolo bluetooth v4.2BR / EDR y estándar BLE
  + Radiofrecuencia: con -97dBm sensibilidad NZIF receptor Clase-1, Clase-2 y Clase-3 emisor AFH
  + Frecuencia de audio: frecuencia de audio CVSDSBC
* 2G GSM / GPRS a través del módulo SIM800L
  + Banda cuádruple: 850/900/1800 / 1900MHz
  + Interfaz: Serial UART
  + Controlado por Comandos AT

### **TUTORIALES**

* [CDMX-Electronica: Como programar el TTGO T-Call con el IDE Arduino](https://cdmxelectronica.com/producto/ttgo-t-call-v1-3-esp32-con-sim800l-240mhz/#Tutorial-CDMXElectronica-TTGO)
* [Enviar SMS T-Call ESP32 SIM800L](https://randomnerdtutorials.com/esp32-sim800l-send-text-messages-sms/)
* [Publicar datos a la nube usando la TTGO T-Call ESP32 SIM800L](https://randomnerdtutorials.com/esp32-sim800l-publish-data-to-cloud/)

### **DOCUMENTACIÓN Y RECURSOS**

#### ****Datasheet´s:****

* [ESP32-WROVER-B](https://www.espressif.com/sites/default/files/documentation/esp32-wrover-b_datasheet_en.pdf)
* [IP5306](https://416w7b49llop13pkg72rweoc-wpengine.netdna-ssl.com/wp-content/uploads/2019/10/IP5306-REG-V1.4.pdf)
* [SIM800L Hardware](https://416w7b49llop13pkg72rweoc-wpengine.netdna-ssl.com/wp-content/uploads/2019/10/SIM800_Hardware-Design_V1.08.pdf)
* [Comandos AT SIM800L](https://416w7b49llop13pkg72rweoc-wpengine.netdna-ssl.com/wp-content/uploads/2019/10/sim800_series_at_command_manual_v1.01.pdf)

#### ****TUTORIAL****

#### ****Cómo programar el TTGO T-Call con Arduino IDE****

#### ****Instalar las tarjetas ESP32 a Arduino IDE****

Para comenzar este tutorial debes de tener instalado el software de Arduino IDE en su ultima versión, posterior mente carga las tarjetas de la familia ESP32 sólo debes de copiar el siguiente link, después

Abrir el IDE de Arduino,

Seleccione Archivo/ Preferencias

Dirígete al “Gestor de URLs” y pega

#### ****https://dl.espressif.com/dl/package\_esp32\_index.json****

En el siguiente GIF podrás guiarte para cargar de manera correcta las tarjetas de ESP32. En caso de tener cargadas las tarjetas de ESP8266, solo debes de poner una “,” al final de la URL y pegar la URL de las tarjetas del ESP32. Si ya tienes las tarjetas cargadas puedes omitir este paso.

Ya que pegaste la URL debes de cerrar el IDE de Arduino y volverlo abrir, después dirigirte a la pestaña de “Herramientas” dar clic en “Placa” y después a “Gestor de tarjetas”, se abrirá una ventana y en la casilla de búsqueda ingresar “esp32” y seleccionar las placas “esp32 by Espressif Systems” he instalarla. En el siguiente GIF podrás observar el proceso que debes hacer para instalar correctamente las tarjetas.

Instalar las tarjetas “esp32 by Espressif Systems” al IDE de Arduino

#### ****Respuesta a comando AT****

Ya que se cargaron las tarjetas de ESP32 al IDE de Arduino lo siguiente que realizaremos será verificar que la placa esta funcionado correctamente, esto lo podemos verificar de manera muy rápida ya que esta placa cuenta con un programa pre-cargado de fabrica para tener comunicación con el módulo de manera serial y poder mandarle y recibir respuesta mediante comandos AT.

#### ****Para realizar esta primer prueba de funcionamiento siga los siguientes pasos:****

1. Insertar el nano SIM al módulo
2. Conectar la antena
3. Conectar el módulo a la computadora mediante un cable USB de tipo C este cable debe de ser de transferencia de datos en caso de que utilices un cable de sólo carga este no lo detectara la computadora. Ya que este conectada la tarjeta a la computadora en la tarjeta se intentara conectar a la red celular, se encenderá y se apagara el led indicador de conexión y cuando se establezca la comunicación el led encenderá y se apagara más lento.
4. Abrir el Arduino IDE
5. Abrir el monitor serie y definir 115200 baudios y tendremos la siguiente repuesta:

