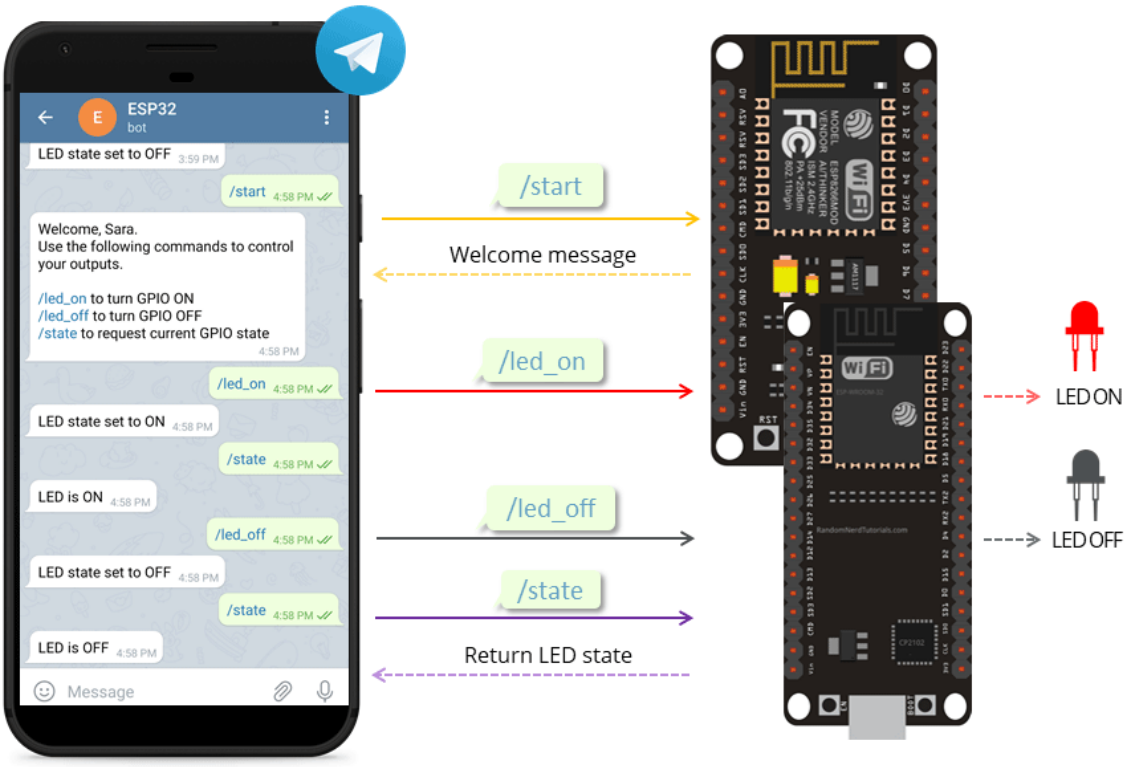
Telegram: ESP32 Control LED

Crear un bot de Telegram para interactuar con la ESP32 y cambiar el estado de un LED. Solo necesita tener acceso a Internet en su teléfono inteligente.

A continuación, una descripción general del proyecto:

- Crear un bot de Telegram para su ESP32
- Iniciar una conversación con el bot ESP32
- Envíar un mensaje **/led_on** al bot ESP32, la placa ESP32 recibe el mensaje y enciende el LED
- Enviar el mensaje **/led_off** para apagar el LED;
- Enviar el mensaje **/state** para obtener el estado del LED;
- Enviar el mensaje **/start** para recibir un mensaje de bienvenida con los comandos para controlar la placa ESP32;
- La ESP32 solo responderá a los mensajes provenientes de su ID de cuenta de Telegram.



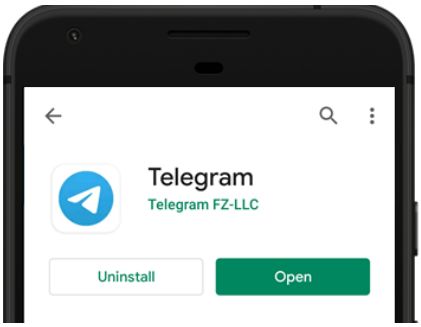
TELEGRAM

Telegram Messenger es un servicio de mensajería instantánea y voz sobre IP basado en la nube. Puede instalarlo fácilmente en su teléfono inteligente (Android y iPhone) o computadora (PC, Mac y Linux). Es gratis y sin anuncios. Telegram te permite crear bots con los que puedes interactuar.

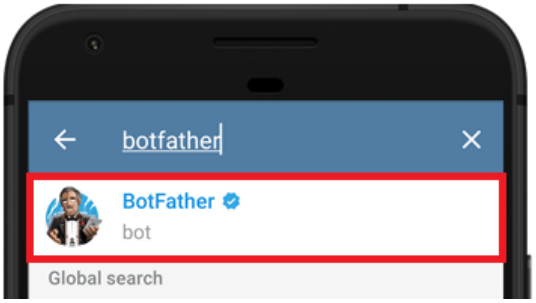
“Los bots son aplicaciones de terceros que se ejecutan dentro de Telegram. Los usuarios pueden interactuar con los bots enviándoles mensajes, comandos y solicitudes en línea. Controlas tus bots usando solicitudes HTTPS a la API de Telegram Bot ”. La ESP32 interactuará con el bot de Telegram para recibir y manejar los mensajes y enviar respuestas.

CREACIÓN DE UN BOT DE TELEGRAM

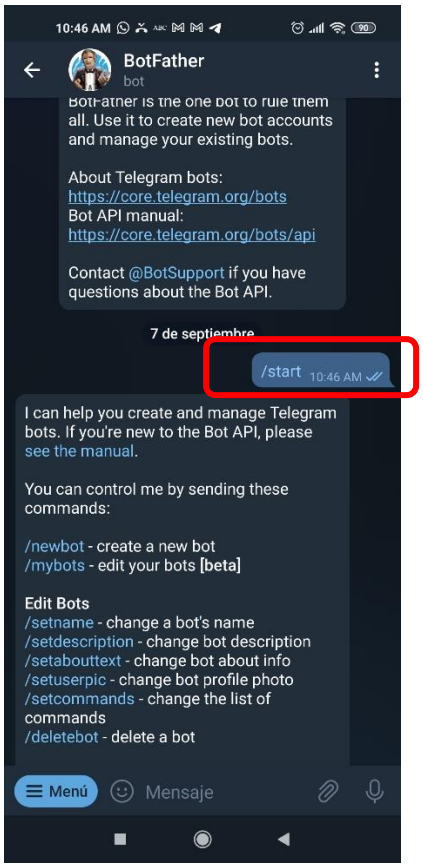
Ir a Google Play o App Store, descargue e instale Telegram.



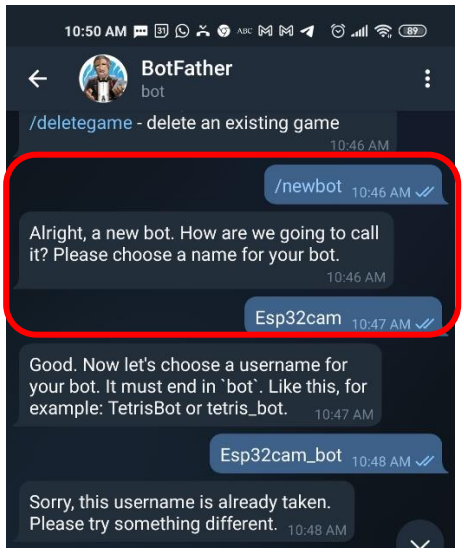
Abre Telegram y sigue los siguientes pasos para crear un Telegram Bot. Primero, busque " botfather " y haga clic en BotFather como se muestra a continuación. O abra este enlace t.me/botfather en su teléfono inteligente.



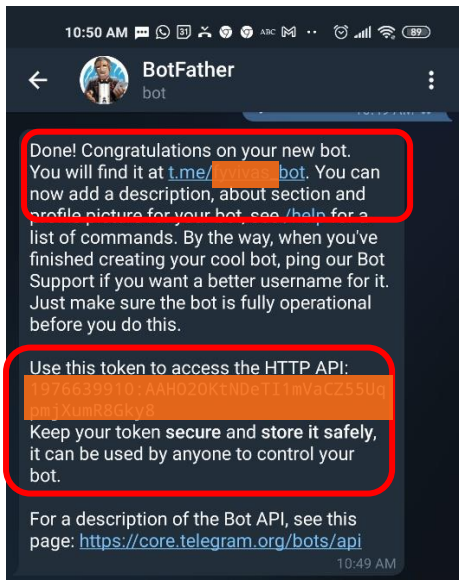
La siguiente ventana debería abrirse y se le pedirá que haga clic en el botón **\start**



Escriba / newbot y siga las instrucciones para crear su bot. Dale un nombre y un nombre de usuario.



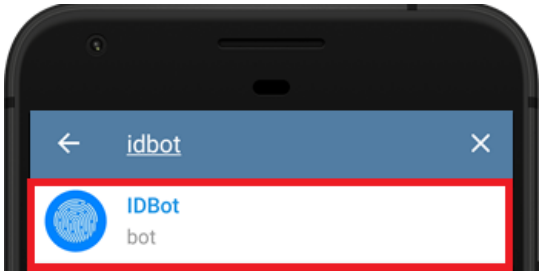
Si su bot se crea correctamente, recibirá un mensaje con un enlace para acceder al bot y al **bot token** . Guarde el bot token porque lo necesitará para que el ESP32 pueda interactuar con el bot.



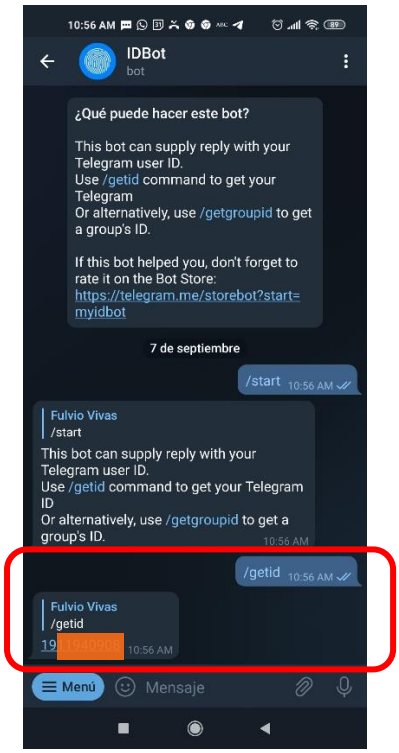
OBTENGA SU ID DE USUARIO DE TELEGRAM

Cualquiera que conozca el nombre de usuario de su bot puede interactuar con él. Para asegurarse de que ignoramos los mensajes que no son de nuestra cuenta de Telegram (o de cualquier usuario autorizado), puede obtener su ID de usuario de Telegram. Luego, cuando su bot de telegrama recibe un mensaje, el ESP32 puede verificar si la identificación del remitente corresponde a su identificación de usuario y manejar el mensaje o ignorarlo.

En su cuenta de Telegram, busque " **IDBot** " o abra este enlace t.me/myidbot en su teléfono inteligente.



Inicie una conversación con ese bot y escriba **/ getid** . Recibirá una respuesta con su ID de usuario. Guarde la identificación de usuario.



BIBLIOTECA DE BOTS DE TELEGRAM UNIVERSAL

Para interactuar con el bot de Telegram, usaremos la biblioteca de bots de Telegram universal creada por Brian Lough que proporciona una interfaz fácil para la API del bot de Telegram.

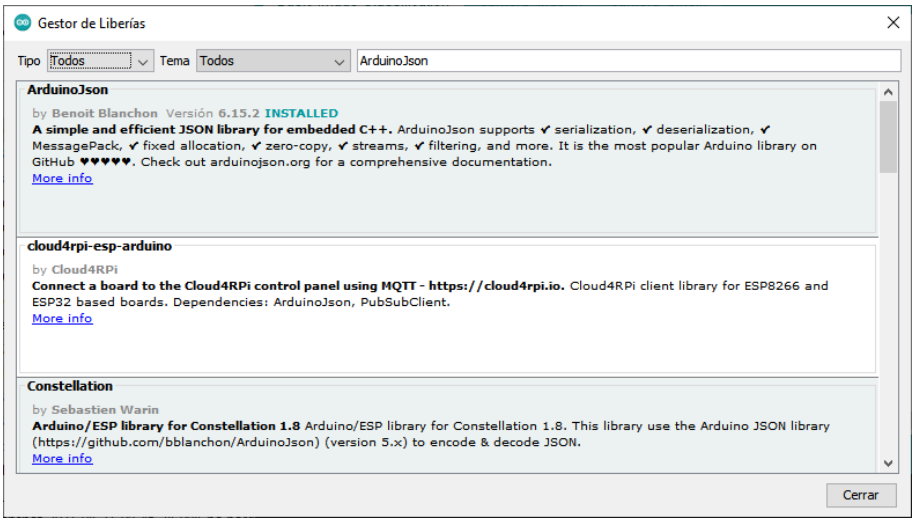
1. Haga clic aquí (https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot/archive/master.zip) para descargar la biblioteca Universal Arduino Telegram Bot .
2. Ir a Programa > Incluir Librería > Añadir Biblioteca .ZIP
3. Agregue la biblioteca que acaba de descargar.

BIBLIOTECA ARDUINOJSON

También debe instalar la biblioteca ArduinoJson . Siga los siguientes pasos para instalar la biblioteca.

1. Ir a Programa > Incluir Librería > Administrar bibliotecas.
2. Busque "ArduinoJson".

3. Instale la biblioteca con la versión 6.5.12.



CÓDIGO

Copie el siguiente código del IDE de Arduino. Debe insertar sus credenciales de red (SSID y contraseña), el token de Telegram Bot y su ID de usuario de Telegram. Además, verifique la asignación de pines para la placa ESP32 que está utilizando.

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

// Wifi network station credentials
#define WIFI_SSID "YOUR_SSID"
#define WIFI_PASSWORD "YOUR_PASSWORD"
// Telegram BOT Token (Get from Botfather)
#define BOT_TOKEN "XXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

const unsigned long BOT_MTBS = 1000; // mean time between scan messages

WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);
unsigned long bot_lasttime; // last time messages' scan has been done

const int ledPin = LED_BUILTIN;
int ledStatus = 0;

void handleNewMessages(int numNewMessages)
{
 Serial.print("handleNewMessages ");
 Serial.println(numNewMessages);

 for (int i = 0; i < numNewMessages; i++)
 {
  String chat_id = bot.messages[i].chat_id;
  String text = bot.messages[i].text;

  String from_name = bot.messages[i].from_name;
  if (from_name == "")
```

```
    from_name = "Guest";

  if (text == "/ledon")
  {
    digitalWrite(ledPin, LOW); // turn the LED on (HIGH is the voltage level)
    ledStatus = 1;
    bot.sendMessage(chat_id, "Led is ON", "");
  }

  if (text == "/ledoff")
  {
    ledStatus = 0;
    digitalWrite(ledPin, HIGH); // turn the LED off (LOW is the voltage level)
    bot.sendMessage(chat_id, "Led is OFF", "");
  }

  if (text == "/status")
  {
    if (ledStatus)
    {
      bot.sendMessage(chat_id, "Led is ON", "");
    }
    else
    {
      bot.sendMessage(chat_id, "Led is OFF", "");
    }
  }

  if (text == "/start")
  {
    String welcome = "Welcome to Universal Arduino Telegram Bot library, " + from_name + ".\n";
    welcome += "This is Flash Led Bot example.\n\n";
    welcome += "/ledon : to switch the Led ON\n";
    welcome += "/ledoff : to switch the Led OFF\n";
    welcome += "/status : Returns current status of LED\n";
    bot.sendMessage(chat_id, welcome, "Markdown");
  }
 }
}

void setup()
{
 Serial.begin(115200);
 Serial.println();

 pinMode(ledPin, OUTPUT); // initialize digital ledPin as an output.
 delay(10);
 digitalWrite(ledPin, HIGH); // initialize pin as off (active LOW)

 // attempt to connect to Wifi network:
 Serial.print("Connecting to Wifi SSID ");
 Serial.print(WIFI_SSID);
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
```

```
 secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root certificate for api.telegram.org
 while (WiFi.status() != WL_CONNECTED)
 {
  Serial.print(".");
  delay(500);
 }
 Serial.print("\nWiFi connected. IP address: ");
 Serial.println(WiFi.localIP());

 Serial.print("Retrieving time: ");
 configTime(0, 0, "pool.ntp.org"); // get UTC time via NTP
 time_t now = time(nullptr);
 while (now < 24 * 3600)
 {
  Serial.print(".");
  delay(100);
  now = time(nullptr);
 }
 Serial.println(now);
}

void loop()
{
 if (millis() - bot_lasttime > BOT_MTBS)
 {
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

  while (numNewMessages)
  {
   Serial.println("got response");
   handleNewMessages(numNewMessages);
   numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  }

  bot_lasttime = millis();
 }
}
```

Credenciales de red

Inserte sus credenciales de red en las siguientes variables.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

or

```
#define WIFI_SSID "YOUR_SSID"
#define WIFI_PASSWORD "YOUR_PASSWORD"
```

ID de usuario de Telegram

Inserta tu ID de chat. El que tienes del IDBot.

```
String CHAT_ID = "XXXXXXXXXX";
```

Token de Telegram Bot

Inserte el token de Telegram Bot que obtuvo de Botfather en el BOTtoken variable.

```
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
```

or

```
#define BOT_TOKEN "XXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

Cree un bot con el token y el cliente definidos anteriormente.

```
UniversalTelegramBot bot(BOT_TOKEN, secured_client);
```

Cree una variable para conectar el pin del LED (LED_BUILTIN).

```
const int ledPin = LED_BUILTIN;
int ledStatus = 0;
```

la funcion handleNewMessages () maneja lo que sucede cuando llegan nuevos mensajes.

```
void handleNewMessages(int numNewMessages) {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);
```

Obtenga el ID de chat para un mensaje en particular y guárdelo en la variable chat_id. El ID de chat identifica quién envió el mensaje.

```
String chat_id = String(bot.messages[i].chat_id);
```

Si el chat_id es diferente de su ID de chat (CHAT_ID), significa que alguien (que no eres tú) ha enviado un mensaje a tu bot. Si ese es el caso, ignore el mensaje y espere el siguiente mensaje.

```
if (chat_id != CHAT_ID){
  bot.sendMessage(chat_id, "Unauthorized user", "");
  continue;
}
```

De lo contrario, significa que el mensaje fue enviado por un usuario válido, así que lo guardaremos en la variable texto.

```
String text = bot.messages[i].text;
Serial.println(text);
```

La variable from_name guarda el nombre del remitente.

```
String from_name = bot.messages[i].from_name;
```

Si recibe el mensaje **/start** , enviaremos los comandos válidos para controlar el ESP32.

```
if (text == "/start") {
  String welcome = "Welcome , " + from_name + "\n";
  welcome += "Use the following commands to interact with the ESP32 \n";
  welcome += "/ledon : to switch the Led ON\n";
```

```
    welcome += "/ledoff : to switch the Led OFF\n";
    welcome += "/status : Returns current status of LED\n";
  bot.sendMessage(CHAT_ID, welcome, "");
}
```

Enviar un mensaje al bot es muy sencillo. Solo necesitas usar sendMessage () en el objeto bot y pasar como argumentos el ID de chat del destinatario, el mensaje y el modo de análisis.

```
bool sendMessage(String chat_id, String text, String parse_mode = "");
```

En nuestro ejemplo, enviaremos el mensaje a la ID almacenada en el CHAT_ID variable (que corresponde a su ID de chat personal) y envíe el mensaje guardado en la variable welcome.

```
bot.sendMessage(CHAT_ID, welcome, "");
```

NOTA: el LED_BUILTIN trabaja con lógica invertida, con un LOW enciende el LED y un HIGH apaga el LED.

Si recibe el mensaje **/ledon** , enciende el LED.

```
if (text == "/ledon")
   {
    digitalWrite(ledPin, LOW); // turn the LED on (HIGH is the voltage level)
    ledStatus = 1;
    bot.sendMessage(chat_id, "Led is ON", "");
   }
```

Si recibe el mensaje **/ledoff** , apaga el LED.

```
if (text == "/ledoff")
   {
    digitalWrite(ledPin, HIGH); // turn the LED on (LOW is the voltage level)
    ledStatus = 0;
    bot.sendMessage(chat_id, "Led is OFF", "");
   }
```

Finalmente, si recibe el mensaje **/status**, obtiene el estado del LED.

```
if (text == "/status")
   {
    if (ledStatus)
    {
     bot.sendMessage(chat_id, "Led is ON", "");
    }
    else
    {
     bot.sendMessage(chat_id, "Led is OFF", "");
    }
   }
```

DEMOSTRACIÓN

Suba el código a la placa ESP32. No olvide ir a Herramientas > Placa y seleccionar la placa ESP32. Vaya a Herramientas > Puerto y seleccione el puerto COM al que está conectada su placa. Después de cargar el código, presione el botón EN/RST integrado de ESP32 para que comience a ejecutar el código. Luego, abra el Monitor Serial para verificar qué está sucediendo en segundo plano.

Ir a la cuenta de Telegram y abra una conversación con su bot. Envíe los siguientes comandos y observe cómo responde el bot:

- **/start** muestra el mensaje de bienvenida con los comandos válidos;
- **/ledon** enciende el LED;
- **/ledoff** apaga el LED;
- **/status** envía el estado del LED a su cuenta de Telegram.

Al mismo tiempo, en el monitor serial, debería ver que el ESP32 está recibiendo los mensajes.

Si intenta interactuar con su bot desde otra cuenta, recibirá el mensaje "Usuario no autorizado".