

[esp8266](#), [arduino](#), [telegram](#), [nodemcu](#)

Enviar mensajes desde el ESP8266 mediante Telegram

Lo primero que necesitamos en crear nuestro propio bot. Para ello instalamos Telegram en nuestro móvil e iniciamos un chat con **@BotFather**.

- Escribimos **/newbot** y nos saldrá un mensaje preguntándonos el nombre que le vamos a poner a nuestro bot
- Una vez escrito el nombre de nuestro bot y al pulsar enviar seguidamente nos preguntará un nombre de usuario.
- Ponemos un nombre de usuario que debe de terminar con **bot**
- Ahora nos saldrá un mensaje en el que nos dará una dirección y un token de acceso
- Iniciamos un chat con nuestro bot
- Si escribimos algo y enviamos el bot nos responde con lo mismo

Una vez que hemos escrito algo en nuestro chat con el bot procederemos a ver como podemos conocer el identificador (chat_id) ya que lo vamos a necesitar posteriormente . Para ello abrimos nuestro navegador y escribimos lo siguiente :

```
https://api.telegram.org/bot<token>/getUpdates?offset=0
```

donde token es el identificador que nos había enviado el @bootFather.

Un ejemplo sería :

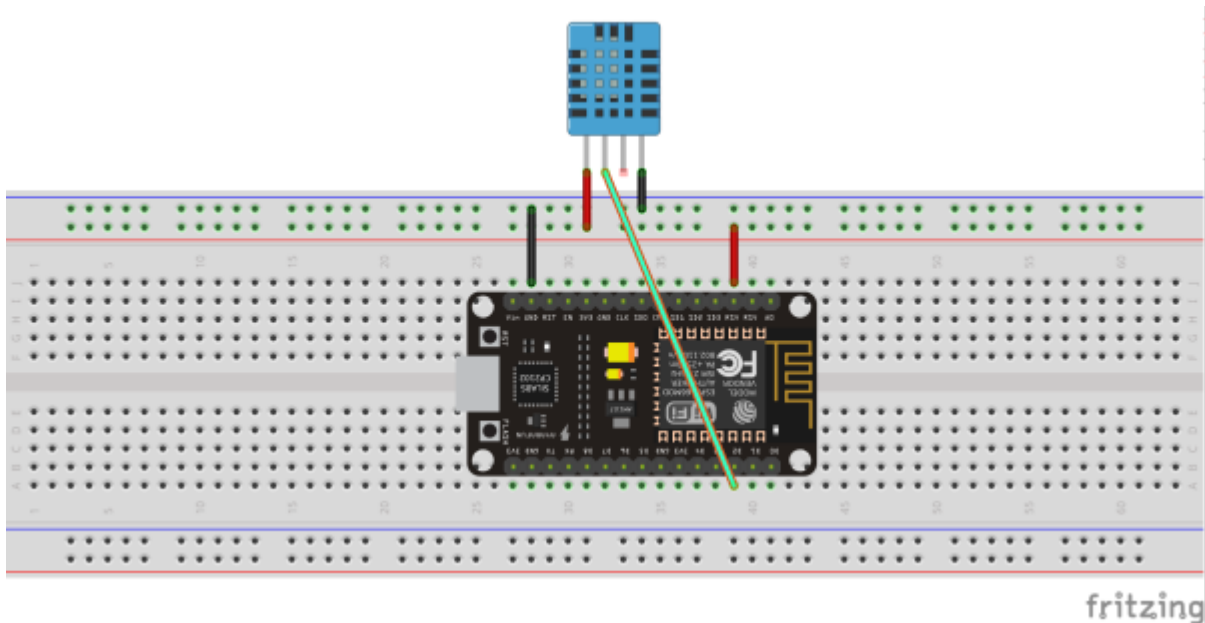
```
https://api.telegram.org/bot266211256:ABE1X7V2MJSx0VB-iK_a_4dXi6f0Uf405hj/getUpdates?offset=0
```

y en el navegador aparecería un resultado como el siguiente:

```
{"ok":true,"result":[{"update_id":546293813,"message":{"message_id":29,"from":{"id":546293813,"first_name":"Intrusos"},"chat":{"id":204236614,"first_name":"Intrusos","type":"private"},"date":1473372888,"text":"Hola"}}]}
```

donde podemos observar que el chat_id es →204236614

Una vez creado nuestro bot vamos a utilizar el mismo circuito que utilizamos para el servidor web, pero ahora haremos que nos envíe la temperatura y la humedad cuando se lo pidamos mediante telegram



```
/* Código original de https://github.com/gusman126/arduino_telegram_bot
   Modificado por wiki.intrusos.info
*/
```

```
#include <WiFiClientSecure.h>
#include <ESP8266WiFi.h>
#include "DHT.h"
```

```
// Definimos los parámetros de conexión a la WIFI
const char *ssid = "intrusos"; // no superior a 32 caracteres
const char *pass = "xxxxxxxxxx"; // contraseña wifi
int status = WL_IDLE_STATUS;
```

```
// Datos del Bot de Telegram
String BOTtoken = "bot2xxxxxxxx:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
//token Ojo hay que poner bot y seguido el token
String Chat_id = "xxxxxxxxxx"; // Chat_id
String Texto_enviar = "";
String Texto_recibido = "";
String Update_id = "";
String anterior_upd = "";
String Nueva_upd = "";
String Respuesta = "";
```

```
// Variables del código de tiempo
int Inicio;
int Termino;
int Intervalo = 15000;
unsigned long elapsed = 0;
unsigned long previous;
boolean respondio = false;
```

```
// Pin del ESP8266 al que está conectado.
// El GPIO 4 corresponde al D2 del ESP8266-12E NodeMCU v3
#define DHTPIN 4

// tipo de sensor DHT
#define DHTTYPE DHT11    // DHT 11

// Inicializa el sensor
DHT dht(DHTPIN, DHTTYPE);

WiFiClientSecure client; // inicio del cliente seguro
IPAddress server(149, 154, 167, 200); // IP de api.telegram.org

void setup() {

    Serial.begin(115200);

    // Conecta a la WIFI
    WiFi.begin(ssid, pass);
    /// }

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("Conectado a la red WiFi");
    Serial.println("Dirección IP: ");
    Serial.println(WiFi.localIP());

    // Comprobamos la conexión a Telegram
    if (client.connect(server, 443)) {
        Serial.println(".... conectado a Telegram");
    }
    // y enviamos el texto de inicio
    Enviar_texto(" Inicio del Sistema .....");

    // Comprobamos el ultimo mensaje
    Ultimo_msg();
    previous = millis();
}

void loop() {
    elapse();
    Leer_msg(); // leemos el ultimo mensaje

    // Comprobamos que haya pasado xx seg desde la ultima vez
    if (elapsed > 500) {
```

```
anterior_upd = Update_id; // Guardamos la anterior Update
Ultimo_msg (); // comprobamos el ultimo mensaje
delay(1000); // Esperamos a recibir los datos
Leer_msg(); // Leemos los datos
busca_upd_id(Respuesta); // buscamos la Update_id y la guardamos
busca_texto(Respuesta); // Buscamos el Texto del mensaje
// Si ha cambiado la Update_id seguimos con el codigo
if (anterior_upd != Nueva_upd) {
    //Serial.println("Es diferente Update");
    Responder_mensaje(Texto_recibido);
} else {
} // No hacemos nada si es el mismo Upd_id
}
} // Fin Loop

// Orden para buscar el texto del mensaje
void busca_texto( String Rsp ) {
    Texto_recibido = "";
    int start = Rsp.indexOf("text") + 7 ; // Buscamos el indice ( numero ) de
la palabra "text" y le añadimos 7
    int fin = Rsp.indexOf("}}}}") - 1; // Buscamos el indice del texto }}} y
le restamos uno
    Texto_recibido = (Rsp.substring(start, fin)); // Guardamos el resultado en
la variable
}

//Orden para buscar la Update_id
void busca_upd_id( String Rsp ) {
    anterior_upd = Update_id; // Guardamos la anterior Update_id para
comprobar
    int start = Rsp.indexOf("update_id") + 11 ; // Buscamos el indice del
texto y le añadimos 11
    int fin = Rsp.indexOf("message") - 2; // Buscamos el indice del texto y
le restamos 2
    Update_id = Rsp.substring(start, fin); // Guardamos la Update_id
    Nueva_upd = Rsp.substring(start, fin); // Volvemos a guardar la Update_id
pero en la variable de nueva
}

// Orden para pedir el ultimo mensaje, vemos que se usa el Offset=-1&limit=1
para mostrar solo el ultimo
void Ultimo_msg () {
    if (client.connect(server, 443)) {
        // client.println("GET /botxxxx/getUpdates?offset=-1&limit=1");
        client.println("GET /" + BOTtoken + "/getUpdates?offset=-1&limit=1");
    }
    previous = millis(); // Guardamos los milisegundos para comprobar que haya
pasado X tiempo entre lecturas
}
```

```
//Leemos el mensaje completo y lo añadimos a una variable caracter por caracter
void Leer_msg () {
    Respuesta = ""; // Vaciamos la variable
    while (client.available()) { // Mientras no lo lea todo seguira leyendo
        char inChar = client.read(); // Lee el caracter
        Respuesta += inChar; // Añadimos caracter a caracter el mensaje
    }
}

//Orden para comprobar el tiempo entre lecturas
void elapse() {
    elapsed = millis() - previous;
}

//Orden para enviar cualquier texto a Telegram
void Enviar_texto( String Texto_enviar ) {
    if (client.connect(server, 443)) {
        client.println("GET /" + BOTtoken + "/sendMessage?chat_id=" + Chat_id +
"&text=" + Texto_enviar + "");
    }
}

//Aqui añadiremos las ordenes de respuesta del arduino
void Responder_mensaje ( String mensaje ) {

    if (mensaje == "Estado") {
        Enviar_texto("Conectado");
        respondio = true;
    }
    else if (mensaje == "Temperatura") {
        float t = dht.readTemperature(); // Obtiene la Temperatura en Celsius
        Enviar_texto(String(t) + "°C");
        respondio = true;
    }
    else if (mensaje == "Humedad") {
        float h = dht.readHumidity(); // Obtiene la Humedad
        Enviar_texto(String(h) + "%");
        respondio = true;
    }

    if (respondio == true) { // mostramos el texto que se ha entendio
        Serial.println("El Texto : " + mensaje + " Lo he entendio perfectamente");
    }
    else {
        Serial.println("El Texto : " + mensaje + " No Lo he entendio");
    }

    respondio = false ; // Dejamos en falso que entendio el mensaje
}
```

```
}
```

```
////////// Fin del codigo
```

Referencias

- https://github.com/gusman126/arduino_telegram_bot
- https://github.com/Lstt2005/ESP8266_I.O.Broker/tree/master/Arduino/Telegram/TelegramBot-master
- <https://github.com/Casajasma/TelegramBot-Library>
- https://create.arduino.cc/projecthub/Arduino_Genuino/telegram-bot-library-ced4d4
- <http://trasteandoarduino.com/2016/03/21/telegram-contronlando-tu-servidor-hablandole-a-un-bot/>
- <http://www.xatakamovil.com/aplicaciones/llegan-los-bots-a-telegram-como-crear-el-tuyo-propio>

From:

<https://intrusos.info/> - **LCWIKI**

Permanent link:

<https://intrusos.info/doku.php?id=electronica:esp8266:telegram&rev=1473715535>

Last update: **2023/01/18 14:15**

