Gestión remota de dispositivos IOT mediante técnicas de mensajería instantánea empleando Bots

Carlos Binker[1], Hugo Tantignone[1], Guillermo Buranits[1], Eliseo Zurdo[1], Diego Romero[1], Rubén Darío Moreira[1], Maximiliano Frattini[1]

[1]Universidad Nacional de La Matanza, Florencio Varela 1903 (B1754JEC) -- San Justo, Buenos Aires, Argentina
{cbinker, htantignone, gburanits, djromero, rmoreira}@unlam.edu.ar; {ezurdo, mfrattini}@alumno.unlam.edu.ar

Abstract. Este trabajo plantea como eje principal gestionar un hardware IOT remotamente mediante mensajería instantánea de texto, empleando *Telegram* [1]. Telegram permite definir *bots* [2] (aféresis de Robots) y administrarlos a través de un *token* que la misma aplicación genera en forma aleatoria a través de su bot principal, denominado *BotFather* [3] . Telegram brinda una *API* [4], que permite que los bots interactúen con nuestro sistema. El bot estará alojado en el chip ESP32. El ESP32 es un SOC (System on chip) [5]. Este SOC incorpora WIFI, bluetooth, sensores, conversores AD y DA, etc. Como caso de estudio se propone controlar mediante un bot a un conjunto de leds y a un sensor; este último elemento será simulado mediante un pulsador. Será necesario para ello el empleo de interrupciones. Además se requerirá el uso de Telegram (versión web o móvil) y el entorno de programación Arduino. Los comandos del bot se activarán anteponiendo el símbolo "/", o bien configurando teclas para cada comando.

Keywords: Telegram, API, Bot, ESP32, IOT

1 Introducción

Cuando se quiere acceder a un host en internet se lo puede hacer a través de su dirección IP o de su nombre, que se extrae de una tabla donde se encuentran mapeadas las direcciones IP con sus respectivos nombres de Host (El Domain Net System o DNS, se encarga de actualizar las tablas de direcciones y nombres). Con el crecimiento de internet y la proliferación de computadoras personales y dispositivos dentro de las redes privadas con servicios de comunicación y acceso a internet brindados por ISPs se hizo necesario el desarrollo de un método para acceder desde internet a los hosts de la red local conectada al router. La asignación de la dirección IP para los clientes se puede hacer en forma estática (se mantiene la dirección IP a lo largo del tiempo) o en forma dinámica (la dirección IP puede variar en el tiempo). La asignación dinámica le impediría a un cliente externo tener la dirección actualizada a la cual comunicarse, por lo cual será necesario un servicio de publicación de las direcciones IP actualizadas. Ver Fig. 1. Para evitar esto, ya que no siempre es fácil tener una IP pública fija se propone

la solución de mensajería por medio de Telegram mediante el empleo de bots, ya que nos independizamos de la capa de red. En nuestro caso nuestro host es un SOC ESP32, que a través de su IP privada accedemos a él a través de la red WIFI local conectada a Internet. El SOC (ESP32) estará configurado en modo STATION y albergará el bot.

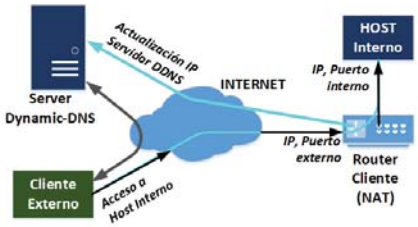

Fig. 1. Esquema de un servicio NO-IP usando una técnica de DNS dinámico

1.1 Esquema de la Plataforma Telegram

Telegram está creado sobre una serie de servidores distribuidos. Dichos servidores, para comunicarse entre sí, utilizan un protocolo propio llamado *MTProto* [6]. La razón del uso de este protocolo propietario es la mejora en seguridad como también el envío de mensajes, sobre todo vídeo e imagen. Haciendo un poco de historia podemos ver que hay dos versiones de MTProto. La primera versión se utilizó en 2014. Los mensajes en MTProto eran cifrados con el algoritmo SHA-1 [7]. Por un reporte en enero de 2015, en donde el investigador Juliano Rizzo reveló un error en el funcionamiento de SHA-1 que originó una vulnerabilidad al interceptar los mensajes, en 2016 se señaló un posible reemplazo del SHA-1 por el SHA-2. Por lo tanto en 2017, se lanza la segunda versión. El cifrado se reemplazó a SHA-256 con mayor cantidad de bytes de carga útil. Para complementar lo dicho observar la Fig. 2.

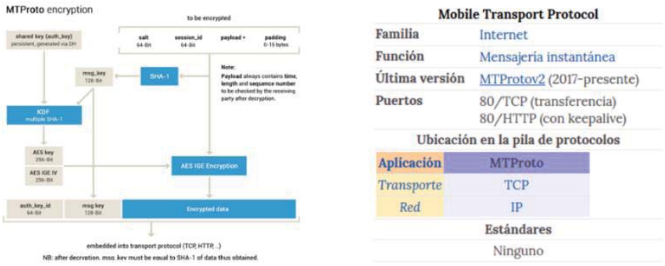

Fig. 2. Encriptación de mensajes Telegram mediante SHA-1. Ubicación del MTPro en el stack de protocolos de comunicaciones

1.2 API de Telegram asociada para el manejo de bots

La API de Telegram permite la interacción de los bots con un sistema. En nuestro caso ese sistema será un hardware a controlar remotamente a través de una interfaz de

usuario, que es en concreto el bot. Los bots de Telegram son cuentas especiales que no están ligadas a un número de teléfono. Al igual que la cuenta de un usuario, el acceso a la misma es por medio del *Alias*, que se accede anteponiendo **@** al nombre del bot, o bien se accede por el propio nombre. Estas cuentas sirven como una interface para albergar código que puede estar ejecutándose en cualquier lugar del mundo en un servidor. Su uso es totalmente transparente, ya que los usuarios no necesitan conocer nada absolutamente sobre cómo el protocolo de encriptación MTProto funciona. Los servidores de Telegram manejarán todo lo referente a la encriptación de los mensajes y la comunicación con la API de una manera muy sencilla. Esta comunicación con los servidores de Telegram a través de la API se da vía una simple conexión https. Todas las consultas a la API Telegram Bot deberán realizarse de esta manera:

https://api.telegram.org/bot<token>/METHOD_NAME.

Se crearán dos tipos de bots, uno que suministra los comandos anteponiendo el símbolo **"/"** y el otro bot suministra los comandos a través de un teclado. El bot creado para la primera situación se llama CACIC_2019, mientras que el bot creado para la segunda situación (comandos a través de teclas) se llama CACIC_2019_Key. A título de ejemplo con el token suministrado para CACIC_2019, la forma de acceder desde la web sería a siguiente:

https://api.telegram.org/bot904682755:AAHG6B1SsF3j6jJdMXJxUklcGBaUg2lEcaU/getme

La API soporta los métodos http GET y POST. Ante una petición (request), la respuesta (reply) de la API bot es un objeto JSON como el siguiente:

{**"ok":**true**,"result"**:{"id":904682755,"is_bot":true,"first_name":"CACIC_2019", "username":"cbinker5bot"}}

Tal como se observa, la API siempre devuelve un campo de tipo Boolean denominado **"ok"** y otro campo denominado **"result"**, en donde puede tener un campo opcional String denominado "*description"* con una descripción del resultado.

Si la petición hacia el bot fue satisfactoria, el campo "ok" recibido en el objeto JSON es igual a *true* y si además el resultado de la consulta fue localizado se nos devolverá el campo "result" con todos los resultados, tal como se indica en el ejemplo precedente En cambio, si la respuesta del campo "ok" es igual a *false,* se devolverá un código de error, tal como el siguiente ejemplo.

{"ok":false,"error_code":401,"description":"Unauthorized"}

En la Fig. 3 se puede observar claramente la interacción entre todos los componentes, es decir entre el cliente, el bot, la API de Telegram y el hardware remoto a controlar.

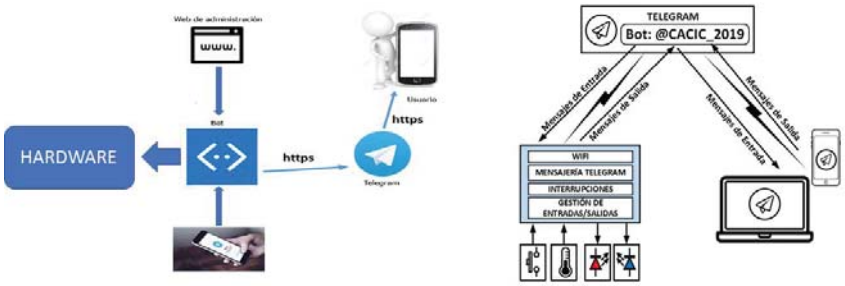

Fig. 3. –Interacción entre usuario (cliente), el bot, la API de Telegram y el hardware remoto

2 SOC ESP 32

Este dispositivo es un SOC que está adquiriendo una enorme popularidad debido a su bajo costo y la implementación de diferentes entornos de programación, tales como LUA y Python. Pero su enorme potencial radica en el hecho de que soporta también el IDE de Arduino y trabajar con él es como si realmente estuviéramos trabajando sobre alguna de las placas Arduino, que como ya sabemos goza de una enorme comunidad abocada al desarrollo de código y de librerías. Todo esto puede ser utilizado en forma transparente para este dispositivo y así evitamos la utilización de un doble microcontrolador. Este dispositivo se destaca por sobre todo porque ya resuelve todo lo concerniente a la comunicación de dispositivos a través de redes, ya que soporta WIFI, Bluetooth, Hardware criptográfico, SPI, I2C, I2S y hasta Ethernet. Además incorpora un sensor de temperatura y sensores Touch capacitivos, además de conversores DAC, ADC, etc. Posee un microprocesador con dos núcleos de 32 bits, lo que lo hace ideal para procesamiento paralelo. Cualquier pin del ESP32 puede configurarse para el control de interrupciones. A continuación exponemos el diagrama en bloques para dar una idea general de su potencialidad, como así también el correspondiente PIN-OUT, (ver Fig.4 para mayor detalle).

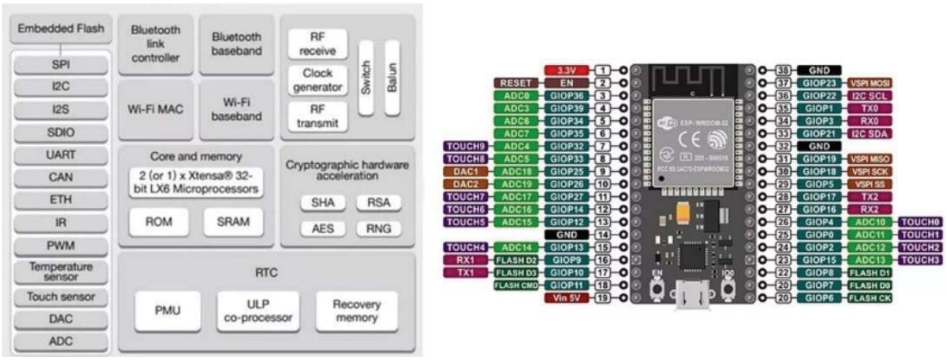

Fig. 4. Diagrama en bloques del ESP32

3 Caso de estudio empleando ESP32, Telegram y Arduino IDE

El prototipo de hardware que se utilizará es el mostrado en la Fig. 5:

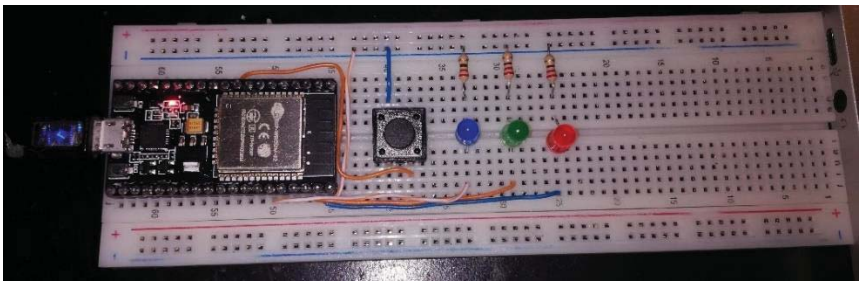

Fig. 5 - Hardware a controlar por el bot de Telegram empleando el NodeMCU ESP32

El bot que lo controlará tiene el siguiente aspecto (ver Fig. 6), el mismo fue configurado a través de los comandos del BotFather, y lo apreciamos tanto en su versión web cómo móvil. También, puede configurarse el bot para que los comandos se introduzcan a través de un teclado, tal como puede observarse en la Fig. 7

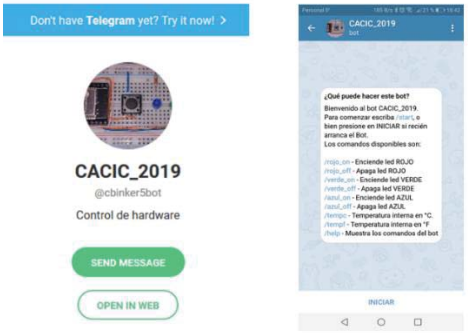

Fig. 6. - Bot que controlará el hardware (CACIC_2019), version web y móvil por comandos

Recordemos que para invocar al bot se usa el alias y éste se corresponde con el nombre del bot. Para acceder en forma web debe escribirse el siguiente comando:

t.me/cbinker5bot (acceso al bot por comandos utilizando **"/"**), donde en este caso cbinker5bot es el username del bot y se corresponde con el Alias (@cbinker5bot). En cambio para el acceso al bot que emplea un teclado para los comandos, el acceso web es *t.me/cacickeybot*. Notar también que obligatoriamente el administrador del bot debe terminar en bot.

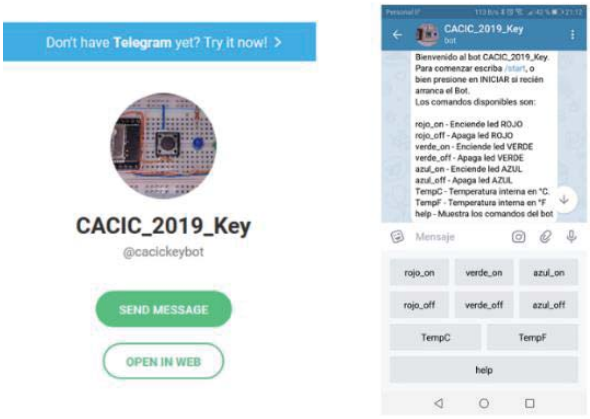

Fig. 7. Bot que controlará el hardware (CACIC_2019_Key), version web y móvil por teclado

4 Desarrollo de la experiencia

4.1 Software empleando el IDE de Arduino

El software que controlará remotamente el hardware por medio del bot es el siguiente (se exponen las partes más sobresalientes y se omiten algunos saltos de líneas), dada su extensión:

4.1.1. Bot que utiliza los comandos anteponiendo el símbolo "/"

1. Para ambos fuentes con sendos bots, las librerías a emplear son las siguientes:

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

Para el correcto funcionamiento, debe estar instalada la librería ArduinoJson version 5.13.5, ya que es una dependencia del resto de las librerías intervinientes.

2. Variable de entorno para el sensor de temperatura interno del ESP32

```
#ifdef __cplusplus
extern "C" { uint8_t temprature_sens_read();}
#endif
```

3. Definición de variables globales:

PULSADOR en el GIOP 25, y pongo los leds en los pines 23, 22 y 21.

```
const int pulsador=25; const int led_rojo=23;
```

```
const int led_verde=22; const int led_azul=21;
```

4. Código para manejo de interrupciones:

```
portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
bool pirTriggered = false;// Al pulsar pasa a true
long chatId = 464201191; // chatId del usuario que
interactúa con el bot
String msg = "Pulsador presionado"; // Mensaje al bot
```

5. Manejo de la interrupción por flanco descendente "Rising"

```
void IRAM_ATTR handleInterruptRising() {
portENTER_CRITICAL_ISR(&mux); Serial.println("Rising");
pirTriggered = true; portEXIT_CRITICAL_ISR(&mux); }
```

6. Instanciación de las credenciales para la conexión WIFI (ssid y password)

```
  const char* ssid = "IPLAN-306232"; // nombre del ssid
  const char* password = "NDUND4XYGYK3";
```

7. Instanciación del Bot de Telegram

```
const String botToken =
"904682755:AAHG6B1SsF3j6jJdMXJxUklcGBaUg2lEcaU";
```

8. Instanciación del cliente ssl para comunicarse con la web API de Telegram.

```
WiFiClientSecure sslPollClient;
```

9. Instanciación de la clase client (argumentos: el Token y un cliente seguro ssl)

```
TelegramBotClient client(botToken, sslPollClient);
```

10. Declaración de función destinada a recibir mensajes desde el Bot de Telegram.

```
void onReceive (TelegramProcessError tbcErr,
JwcProcessError jwcErr, Message* msg)
{Serial.println("onReceive") …;
```

11. Ejemplo de tratamiento de un comando (por ejemplo encender el led rojo)

```
if (msg->Text == "/rojo-on" )    {
client.postMessage(msg->ChatId,String("Encendiendo el
led rojo")); digitalWrite(led_rojo, HIGH); }
```

12. Arranque del WIFI utilizando las credenciales definidas

```
void setupWiFi()
{
Serial.println();Serial.printf("Intentando conectar a
la red %s ",ssid); Serial.println();
WiFi.begin(ssid, password); //Método que arranca WiFi.
while (WiFi.status() != WL_CONNECTED) {
```

```
delay(500); Serial.print(".");} // Mientras no conecta
imprimo ....
Serial.println(); Serial.println("OK");
Serial.print("IP address");
Serial.println(WiFi.localIP());
```

13. Imprimo OK y muestro la IP asignada mediante DHCP por el router WIFI.

```
// Imprimo la intensidad de la señal en dbm.
Serial.print("Strength ...: ");
Serial.println(WiFi.RSSI()+ String(" dbm"));
Serial.println();}
```

14. Función setup, se ejecuta una sola vez.

```
// Habilito como salida el pin GIOP23. Lo mismo para los otros leds.
void setup() {pinMode(led_rojo, OUTPUT);
```

15. Manejo de interrupciones

```
pinMode(pulsador, INPUT_PULLUP); // Resistencias
internas de Pull Up.
attachInterrupt(digitalPinToInterrupt(pulsador),
handleInterruptRising, RISING);
```

16. El cliente llama a las funciones para el procesamiento de datos o errores.

```
client.begin( onReceive, onError);
```

17. Pregunto si se disparó un evento, en este caso si se oprimió un pulsador en loop.

```
void loop() {
if (pirTriggered) {
client.postMessage(chatId, msg); // ChatId user
pirTriggered = false; // pirTriggered en true}
```

18. Para procesar los datos recibidos, este método deberá ser invocado en forma continua dentro del main loop.

```
client.loop();
```

4.1.2. Bot que utiliza los comandos ingresados mediante teclado

En esta situación el código se modifica de la siguiente manera:

En la función onReceive, para instalar el teclado se ejecuta la siguiente sentencia:

```
if (msg->Text == "/start") client.postMessage(msg-
>ChatId,String("Bienvenido. Ingrese los comandos indicados.
Escriba /help para mas información.\n"),board);
```

La clave es la palabra reservada **board**. El teclado ya se instala al darle *start* sólo una vez. El teclado se activa con las siguientes sentencias:

// Añadiendo cuatro filas al teclado

```
String row1[] = {"rojo-on","verde-on","azul-on"};
String row2[] = {"rojo-off","verde-off","azul-off"};
String row3[] = {"TempC", "TempF"};
String row4[] = {"help"};
```

// Configurando el teclado, cantidad de teclas por fila. Observar sólo va "**;**" al final

```
push(3, row1)
push(3, row2)
push(2, row3)
push(1, row4);
```

4.2 Resultados obtenidos

Se indican en la Fig. 8 los resultados obtenidos desde el bot para el control del hardware. Este bot emplea comandos anteponiendo el símbolo de "/". En la Fig. 9 se ve el resultado de aplicar el bot con los comandos a través de un teclado en la pantalla.

Fig. 8. Resultados arrojados por el bot con comandos precedidos por **"/"**

Fig. 9. - Resultados arrojados por el bot con comandos dados por teclas

Fig. 10. – Información recepcionada en ambos bots al activarse un sensor (PULSADOR). Se muestra el mensaje Pulsador presionado.

Ejemplo de Capturas del monitor serie del JSON recibido al prender y apagar el led rojo:

{"ok":true,"result":{"message_id":1464,"from":{"id":*904682755*,"is_bot":true,"first_name":"CACIC_2019","username":"cbinker5bot"},"chat":{"id":*464201191*,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1565008733,"text":"Encendiendo el led rojo"}}

{"ok":true,"result":{"message_id":1466,"from":{"id":*904682755*,"is_bot":true,"first_name":"CACIC_2019","username":"cbinker5bot"},"chat":{"id":*464201191*,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1565008746,"text":"Apagando el led rojo"}}

5 Conclusiones y trabajo futuro

1. La mensajería constituye un excelente método de control a través de Internet sin la necesidad de caer en un esquema de un servicio NO-IP como lo explicado en la introducción.
2. Cada vez es mucho más el interés por el empleo de bots y se desarrollan constantemente nuevos frameworks a fin de facilitar la escritura del código.
3. En un futuro trabajo se prevee la utilización de bots pero para que se comuniquen directamente entre dispositivos sin intervención humana.

6 Referencias

1. https://telegram.org/
2. Designing bots. 1st Edition. Amir Shevat. Publicación: mayo del 2017. Editorial OREILLY. ISBN-13: 978-1491974827
3. https://core.telegram.org/bots#6-botfather
4. https://core.telegram.org/bots/api
5. Internet of Things Projects with ESP32. Agus Kurniawan. Publicación: 30 de marzo de 2019. Editorial Packt. ISBN-13: 978-1789956870
6. https://core.telegram.org/mtproto
7. INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY (Paper). CHAITYA B. SHAH, DRASHTI R. PANCHAL. Volume 2, Issue X, Oct 2014. ISSN 2320-6802.