

# SKAPALAB

CONTROL DE ACCESO Y ORGANIZACION INTELIGENTE  
PARA SALAS DE TRABAJO



# SKAPALAB

## Control de acceso

*María Hervás Gutiérrez*

*Miguel Gámez Berral*

*Fernando Gómez de la Varga*

*David M. Ospina Eslava*



# Índice de contenido

---

|       |  |    |
|-------|--|----|
| 1     | Introducción y objetivos .....                   | 6  |
| 2     | Diseño Hardware .....                            | 6  |
| 2.1   | Identificador de usuario .....                   | 7  |
| 3     | Diseño Software .....                            | 9  |
| 3.1   | Flujo del programa .....                         | 9  |
| 3.1.1 | Bot de Telegram .....                            | 9  |
| 3.1.2 | Flujo del dispositivo de control de acceso ..... | 11 |
| 3.2   | Descripción del funcionamiento .....             | 11 |
| 3.2.1 | Bot de Telegram .....                            | 11 |
| 3.2.2 | Dispositivo de control de acceso.....            | 13 |
| 3.2.3 | Base de datos .....                              | 14 |
| 3.3   | Robustez del sistema.....                        | 15 |
| 3.3.1 | Bot de Telegram .....                            | 15 |
| 3.3.2 | Dispositivo de control de acceso.....            | 15 |
| 3.4   | Posibles mejoras y trabajo futuro .....          | 16 |
| 3.5   | Librerías .....                                  | 16 |
| 4     | Resultados y conclusiones .....                  | 17 |
| 5     | Manual de usuario.....                           | 18 |
| 5.1   | Manual del administrador de SkapaLab.....        | 18 |
| 5.1.1 | Puesta en marcha del sistema .....               | 18 |
| 5.1.2 | Registro de usuarios.....                        | 19 |
| 5.1.3 | Visualización de reservas .....                  | 20 |
| 5.2   | Manual del usuario miembro de SkapaLab.....      | 21 |
| 6     | Ficheros entregados .....                        | 21 |

## Índice de figuras

---

|   |    |
|---|----|
| Figura 1: Esquema de conexión. ....       | 7  |
| Figura 2: Identificador RFID. ....        | 7  |
| Figura 3: Lector RFID. ....               | 8  |
| Figura 4: Placa ESP8266. ....             | 8  |
| Figura 5: Reenvío de puertos. ....        | 18 |
| Figura 6: Configuración Bot. ....         | 19 |
| Figura 7: Registro de nuevo usuario. .... | 20 |
| Figura 8: Dashboard. ....                 | 20 |

## Índice de diagramas

---

|   |    |
|---|----|
| Diagrama 1: Esquema de la solución empleada. ....       | 6  |
| Diagrama 2: Bot de Telegram. ....                       | 9  |
| Diagrama 3: Proceso de reserva. ....                    | 10 |
| Diagrama 4: Programación de los módulos de acceso. .... | 11 |

## Índice de tablas

---

|                                      |    |
|--------------------------------------|----|
| Tabla 1: Topics de publicación. .... | 14 |
| Tabla 2: Topics de suscripción. .... | 14 |

# 1 Introducción y objetivos

Skapalab es un taller de acceso con suscripción con gran variedad de maquinaria y herramientas para que los usuarios puedan desarrollar sus proyectos sin necesidad de un gran desembolso económico. Al ser un espacio tan diverso, se presenta la dificultad de gestionar el uso de la maquinaria tanto para garantizar la seguridad del usuario, como para evitar solapamiento en las horas de uso.

Se propone una solución dentro del ámbito del “internet de las cosas” (IoT) usando distintas tecnologías. Se incorporan dispositivos de identificación por radiofrecuencia conectados a placas de desarrollo ESP8266 para restringir el acceso a determinadas salas del local y NodeRed y Bases de datos para la gestión del administrador del espacio. Para la interacción del usuario se utiliza la plataforma Telegram, Diagrama 1. Se logra así un sistema que permite a los usuarios acceder solo a las salas con la maquinaria o servicios incluidos en su suscripción. Además, se evitará que personas sin la formación necesaria utilicen maquinaria peligrosa y se expongan a los riesgos que eso conlleva. Por otro lado, se incorpora un servicio de reservas desde Telegram desde el que se informa sobre la disponibilidad del espacio y a través del cual el usuario puede reservar la sala durante un tiempo limitado.

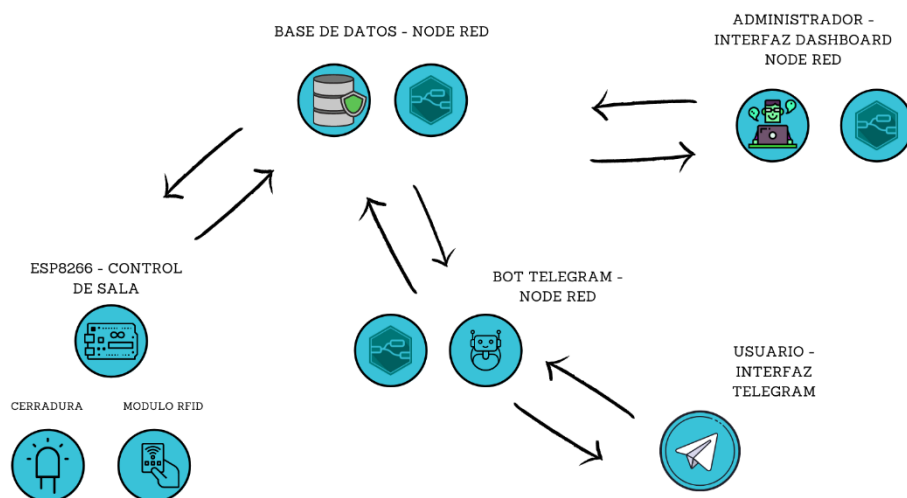
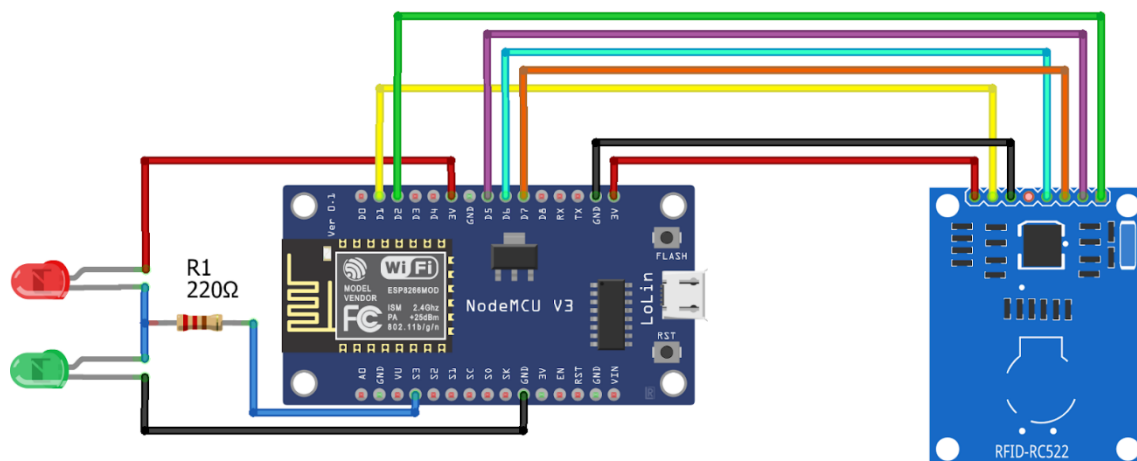


Diagrama 1: Esquema de la solución empleada.

## 2 Diseño Hardware

El sistema implementado se compone principalmente del sensor RFID y la placa de desarrollo NodeMCU, la cual nos facilita el uso del microcontrolador de Espressif ESP8266. Se comunican entre sí mediante el bus SPI, siguiendo la conexión que se observa en la Figura 1.



Para simular el funcionamiento de la cerradura electrónica hemos añadido dos leds, indicando con el verde que la puerta está abierta y con el rojo cerrada. Se controlan con el mismo pin, de manera que con el pin en estado “LOW” la cerradura estará cerrada, y en estado “HIGH” abierta.

## 2.1 Identificador de usuario

El sensor empleado para la identificación es un dispositivo de *identificación por radiofrecuencia, RDIF*. Con este módulo, y una tarjeta RDIF personal para cada cliente o socio del Skapalab, se controla el acceso a las distintas salas, según el nivel de autorización que tengan estos y sus reservas.

El sistema consta de dos elementos, uno pasivo y otro activo. Las tarjetas o llaveros de usuario denominadas TAGS, poseen un circuito con una pequeña memoria que almacena un ID unívoco que se usa para identificar al usuario. Estas tarjetas al acercarse al elemento activo, el lector RDIF, y entrar en el rango de la señal de radiofrecuencia, usan la energía de esta para activar su circuitería y permitir la lectura del contenido de su memoria. La estructura de los dispositivos se puede apreciar en la Figura 2.

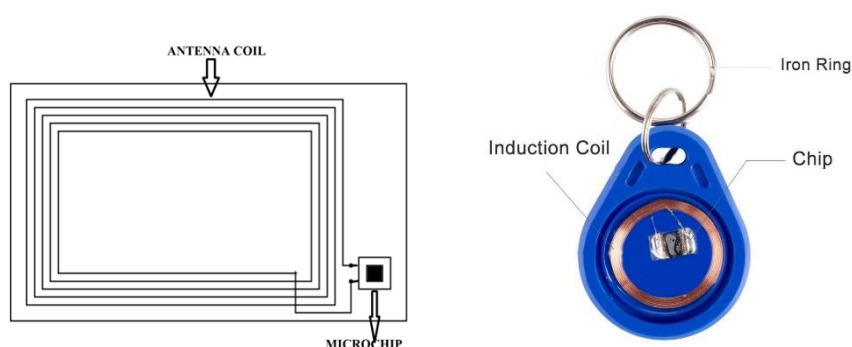


Figura 2: Identificador RFID.

El segundo elemento es el propio lector RDIF. En este caso para el sistema se ha utilizado concretamente el *Módulo RFID RC522* (Figura 3). Este módulo trabaja con un sistema de modulación y demodulación de 13.56MHz, frecuencia que usa la tecnología RFID. El módulo está diseñado para alimentarse a 3.3V, por lo que la misma placa ESP8266 podrá alimentarlo.



Figura 3: Lector RFID.

La comunicación del módulo con la placa se realiza por bus SPI, Serial Peripheral Interface. Este es un bus de comunicación estándar, tipo master-slave, entre dispositivos digitales. Se basa en un combinación serie compuesta por cuatro cables:

- **SCLK (Clock):** Señal de sincronización
- **MOSI (Master Output Slave Input):** Salida de datos del Master y entrada de datos al Esclavo.
- **MISO (Master Input Slave Output):** Salida de datos del Esclavo y entrada al Master.
- **SS/Select:** Para seleccionar un Esclavo, o para que el Master le diga al Esclavo que se active. De este tipo, hay tantas señales independientes como esclavos tenga el master.

La placa ESP8266 empleada posee dos grupos de pines preestablecido, para comunicación SPI (Figura 4).

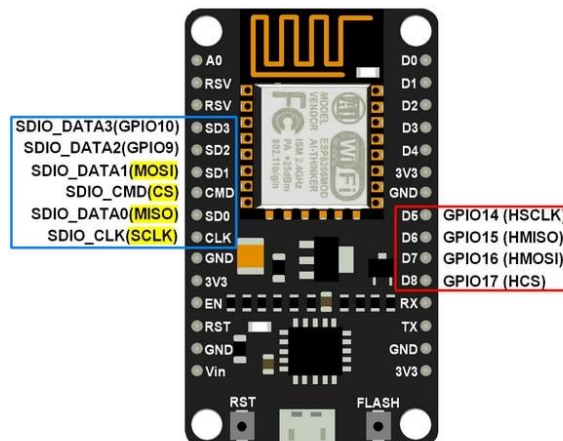


Figura 4: Placa ESP8266.

En este caso, como se puede apreciar en el esquemas de conexionado del dispositivo (Figura 1), se ha optado por los pines de las derecha.



## 3 Diseño Software

### 3.1 Flujo del programa

La programación del Bot de Telegram y de los dispositivos de control de acceso a las salas funcionan de forma independiente, por lo que se presentan en flujos separados en las posteriores secciones.

#### 3.1.1 Bot de Telegram

En el Diagrama 2 se muestra el funcionamiento general del Bot de Telegram. A parte, tenemos el procesos de reserva de horas para los distintos espacios en el Diagrama 3.

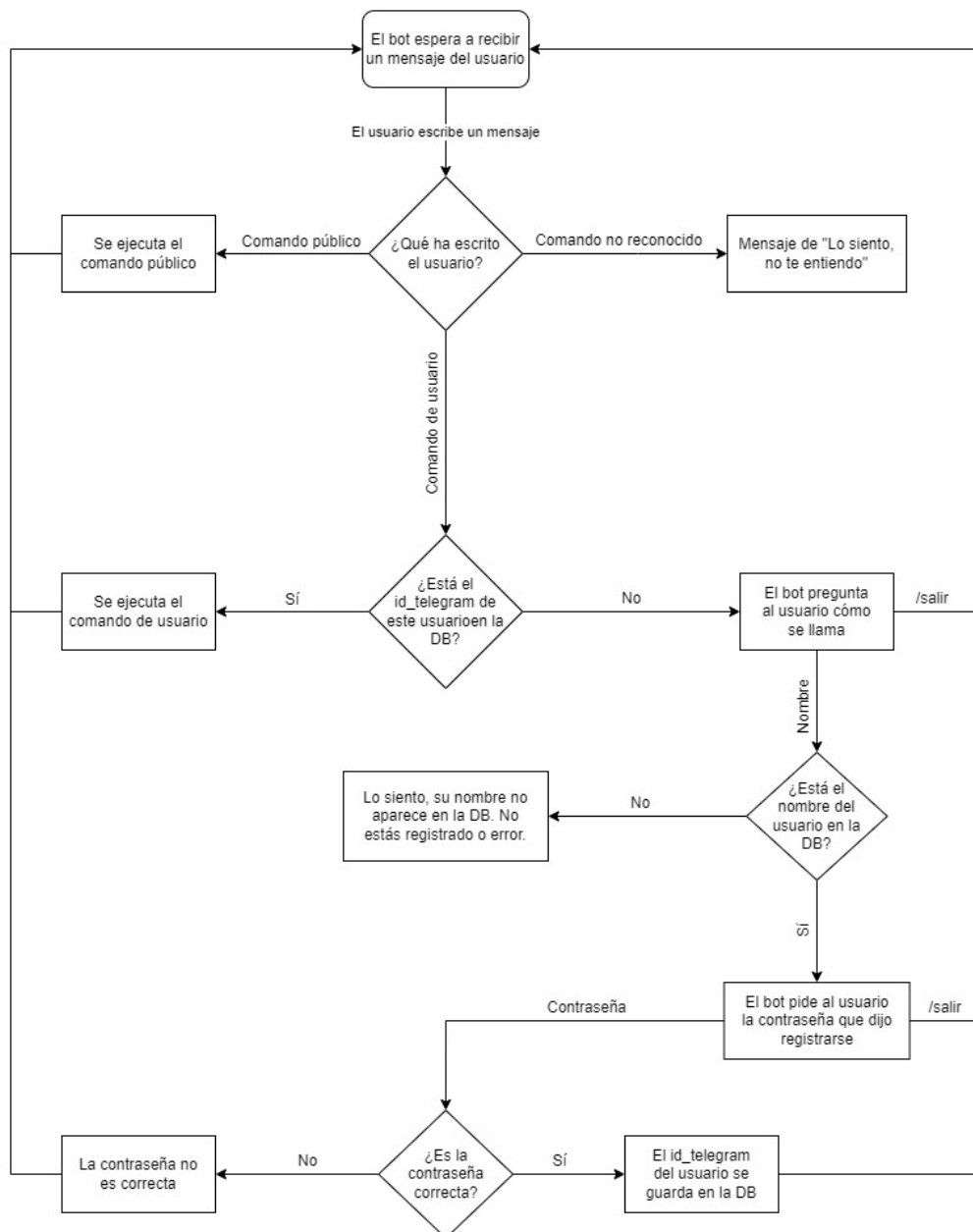


Diagrama 2: Bot de Telegram.

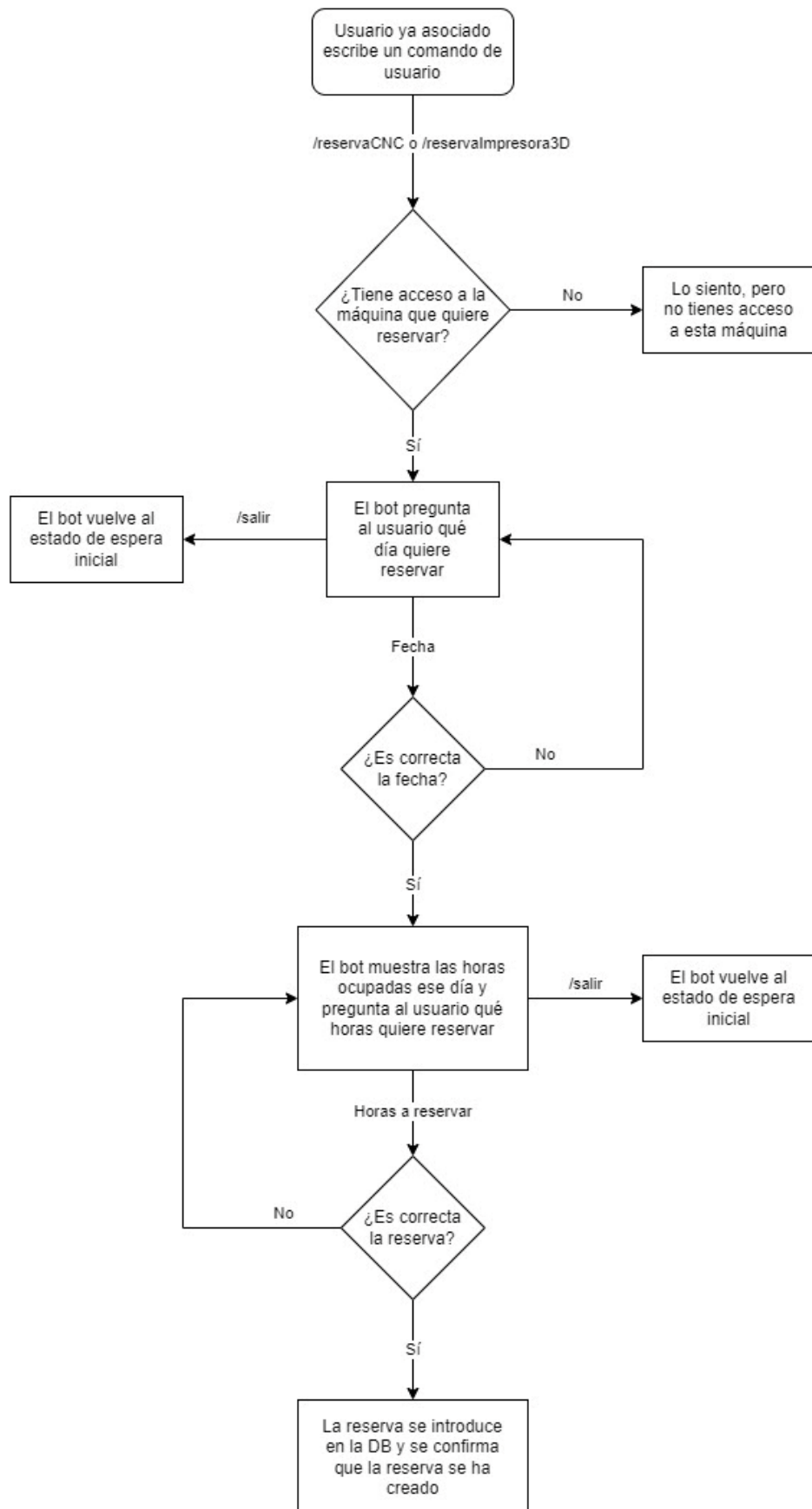


Diagrama 3: Proceso de reserva.

### 3.1.2 Flujo del dispositivo de control de acceso

El Diagrama 4 describe las funciones que realiza el módulo de cada puerta. Se encarga de gestionar cada ID que se detecte, y de abrir o cerrar la puerta según convenga. También se emplea una cache de *IDs comunes*, que serán los que tengan reserva y el administrador. Más adelante se describirán en detalle.

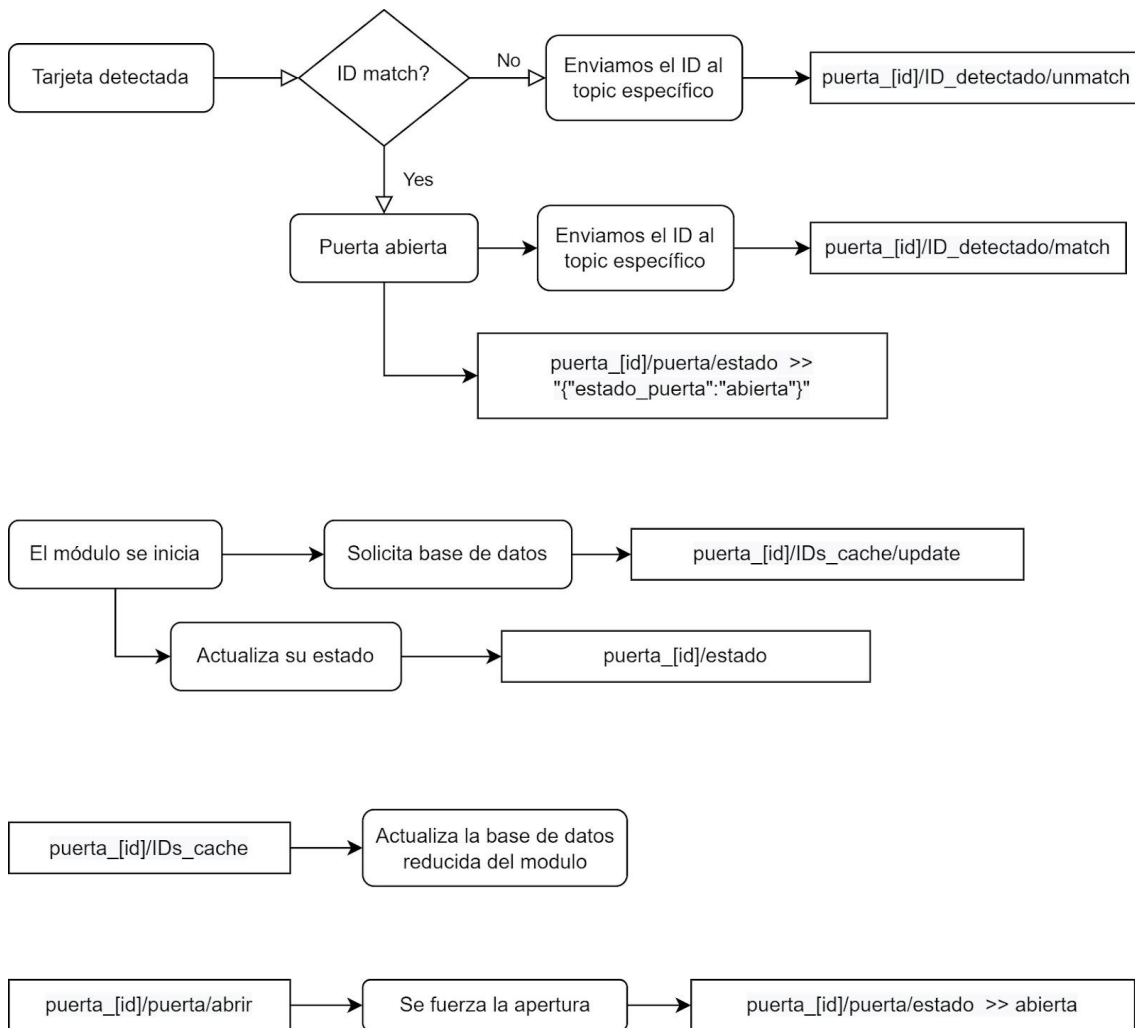


Diagrama 4: Programación de los módulos de acceso.

## 3.2 Descripción del funcionamiento

### 3.2.1 Bot de Telegram

El Bot de Telegram *SkapaLaBot* ha sido creado en NodeRed haciendo uso de la librería *node-red-contrib-chatbot* que, entre otras cosas, incluye funcionalidades avanzadas para el diseño de Bots de Telegram.

El Bot siempre está a la espera de que el usuario escriba algo con el nodo *Telegram Reciever*. Cuando el Bot recibe un mensaje, reconoce si el contenido del mensaje es un comando que tiene implementado haciendo uso del nodo *Rules*. Si el mensaje recibido no está dentro de lo que el Bot comprende, manda un mensaje al usuario, haciendo uso del nodo *Telegram Sender*, explicándolo e invita a ver el resumen de los comandos implementados haciendo uso de */start*.

El comando */start* muestra los distintos comandos disponibles. Los comandos están divididos en comandos en comandos públicos y comandos de usuario.

Los comandos públicos son de libre acceso por cualquier usuario que interactúe con el Bot. Entre ellos se encuentran:

- ***/start*** : Muestra un resumen de los comandos disponibles.
- ***/contacto*** : Muestra información de contacto de SkapaLab.
- ***/preguntasFrecuentes*** : Muestra preguntas frecuentes sobre SkapaLab.
- ***/asociarUsuario*** : Inicia el proceso de asociación de id de Telegram con un usuario registrado en la base de datos de SkapaLab.

Para acceder a los comandos de usuario, se comprueba en la base de datos si el id del chat asociado al mensaje que se está recibiendo corresponde a algún usuario registrado en la base de datos. Si existe, se ejecuta el comando. Si no existe, se comienza el proceso de asociación de usuario, también accesible a través del comando */asociarUsuario*.

El proceso de registro de un usuario en la base de datos lo hace de manera manual el administrador a través del Dashboard. En este registro, entre otra información, se pide una contraseña que se utiliza en el proceso de asociación del usuario.

En el proceso de asociación de usuarios, el Bot pide al usuario que escriba su nombre y queda a la espera de la respuesta de este. Se comprueba en la base de datos si el nombre introducido por el usuario se corresponde con el de algún usuario registrado manualmente. Si existe, se pide la contraseña que se proporcionó en el registro físico. Una vez más, se comprueba en la base de datos si el usuario tiene asociada dicha contraseña, si es correcta, el id del chat se almacena en los datos del usuario en la base de datos en el campo *id\_telegram* y se notifica al usuario de que la asociación ha tenido éxito.

Una vez asociado, el usuario tiene acceso a los comandos de usuario:

- ***/reservaCNC*** : comienza el proceso de reserva de la sala de la CNC.
- ***/reservaImpresora3D*** : comienza el proceso de reserva de la sala de la impresora 3D.

El proceso de reserva es totalmente análogo para ambas salas, excepto que se guardan por separado las reservas.

Cuando un usuario registrado y asociado introduce cualquiera de los comandos de reserva, se comprueba en la base de datos si, con la suscripción que tiene el usuario, tiene acceso a la sala correspondiente. Si así es, el Bot pide al usuario que introduzca la fecha que quiere reservar con el formato *dd/mm/aaaa*, y queda a la espera de la respuesta del usuario. Tras recibir la fecha, se comprueba que sea una fecha válida. Se comprueba el formato, que la fecha exista en el calendario y que sea igual o posterior a la fecha actual. Si se cumplen los requisitos, se muestran las horas

disponibles para reserva. Se muestran las horas desde la apertura hasta el cierre y con paréntesis se indican las horas que están ocupadas. Tras eso, el Bot indica al usuario que escoja las horas a reservar, mostrando un ejemplo para demostrar el formato. Si el formato es correcto y la franja horaria introducida no está ocupada, la reserva se guarda en la base de datos y se confirma al usuario de que se ha guardado.

Para evitar que un usuario se quede atrapado en un bucle de proceso, se ha implementado el comando */salir*, que permite interrumpir cualquier proceso y volver al estado inicial del Bot en el que muestra los comandos disponibles.

En varias ocasiones, en los procesos de asociación de usuario y de reserva, se hace uso de variables globales. Esto es debido a que a pesar de que el bloque *Telegram Sender* permita continuar una conversación encadenando mensajes, el mensaje que llega al bloque desaparece y es reemplazado por un mensaje nuevo que contiene la información recibida del usuario. Para conservar esa información a lo largo del flujo de la conversación ha sido necesario usar variables globales como "dia", "sala" o "nombre".

### 3.2.2 Dispositivo de control de acceso

La programación de los dispositivos se realiza en C++, lenguaje de programación de la placa NodeMCU que usa el chip ESP8266.

Se han incluido las diferentes librerías necesarias para uso de las distintas funcionalidades y módulos necesarios.

La programación del módulo se ha organizado en cuatro ficheros de un mismo proyecto.

El primer fichero *wifiHandler.h*, es el encargado de habilitar la conexión wifi y autenticar las credenciales de red y contraseña en la red a la que se debe conectar el módulo.

El segundo fichero *mqttHandler.h*, es el encargado de iniciar y gestionar la comunicación MQTT entre nuestro dispositivo, cliente, y el broker. En este fichero se crean los topics de suscripción y publicación, se rellenan las credenciales y dirección de servidor, y entonces se inicia la conexión y las respectivas suscripciones.

El fichero *procesoOTA.h* incluye las funciones necesarias para comprobar en el inicio si hay alguna actualización disponible en el servidor del broker.

Finalmente, el fichero *ArduinoCode.ino* contiene el funcionamiento principal del módulo. La primera función de este fichero, *procesa\_mensaje*, es la que se establece como *callback* y es encargada de deserializar y procesar los mensajes recibidos en los topics a los que se está suscrito. Se pueden recibir dos tipos de mensajes: el primero es el mensaje procedente del topic *puerta\_[id]/IDs\_cache*, en cuyo caso se actualiza la caché al contenido del mensaje, y el segundo es el mensaje procedente del topic *puerta\_[id]/puerta/abrir*, en cuyo caso se abrirá la puerta. Éste último comando está pensado para poder abrir la puerta desde el servidor de la base de datos o por el mismo administrador cuando lo requiera.

Luego se encuentran las funciones de *StringIDToByteID*, cuya función consiste en transformar el ID en string proveniente del broker mqtt a un ID en array de bytes para la cache. Y la función

*comparaUID* que compara el ID de la tarjeta leída y con los almacenados en la cache. Y finalmente las funciones abrir y cerrar puerta.

El programa *loop* consiste en la continua espera de la presencia de un TAG cerca del módulo para activar la lectura de este. Una vez detectada pasa a su lectura y posterior comparación con los IDs de la cache. Las dos primeras tareas se valen de funcione propias de la librería del módulo, *MFRC522.h*, *mfrc522.PICC\_IsNewCardPresent()* *ymfrc522.PICC\_ReadCardSerial()*, si alguna de estas funciones diera como resultado un valor booleano nulo, el bucle se reiniciará y ninguna acción sobre la apertura es realizada.

### 3.2.2.1 Listado de topics para comunicación

En las Tabla 1 y Tabla 2 se muestran los topics empleados para la comunicación vía MQTT con el broker y la base de datos.

| Topics (publicar)                       | Descripción  | Ejemplo                                   |
|---|--|---|
| <i>puerta_[id]/ID_detectado/unmatch</i> | Envía el ID detectado en caso de no coincidir.                               | <code>{"ID": "XX:XX:XX:XX"}</code>        |
| <i>puerta_[id]/ID_detectado/match</i>   | Envía el ID detectado en caso de coincidir, despues de abrir la puerta.      | <code>{"ID": "XX:XX:XX:XX"}</code>        |
| <i>puerta_[id]/puerta/estado</i>        | Al abrir o cerrar la puerta actualiza su estado.                             | <code>{"estado_puerta": "abierta"}</code> |
| <i>puerta_[id]/IDs_cache/update</i>     | Envía solicitud de la caché de IDs. Lo enviará cada vez que el µC se inicie. | <code>"true"</code>                       |
| <i>puerta_[id]/estado</i>               | Estado de conexión del dispositivo (se incluye el mensaje LWT).              | <code>{"online": true}</code>             |

Tabla 1: Topics de publicación.

| Topics (suscribir)              | Descripción  | Ejemplo   |
|---------------------------------|--|---|
| <i>puerta_[id]/IDs_cache</i>    | Se recibe el caché.                                    | <code>{"idComun1": "88:109:194:73",<br/>"idComun2": "80:98:231:164",<br/>"idComun3": "XX:XX:XX:XX",<br/>"idComun4": "XX:XX:XX:XX"}</code> |
| <i>puerta_[id]/puerta/abrir</i> | Recive la señal para abrir la puerta de forma directa. | <code>"true"</code>   |

Tabla 2: Topics de suscripción.

### 3.2.3 Base de datos

Los datos de los usuarios registrados y de las reservas realizadas a través de Telegram tienen que ser almacenados para su posterior uso. Para este propósito se utiliza la base de datos NoSQL MongoDB caracterizada por proporcionar un servicio rápido, sencillo y fácilmente escalable. La base de datos privada de este proyecto es "II8" y está instalada en el servidor *iot.ac.uma.es*. Para acceder a ella será necesario introducir como nombre de usuario "II8" y como contraseña "kS0Ooojy".

La base de datos se divide en dos colecciones: “usuarios” y “reservas”. En la primera se almacenan los datos del usuario registrado con el siguiente formato:

```
_id: "61fbbf27fe7923028c1d8cda"
nombre: "María Hervás"
fecha_vencimiento: "2022-04-03T23:00:00.000Z"
acceso_CNC: true
acceso_3D: true
password: "pato"
suscripcion: "Profesional"
fecha_inclusion: "2022-02-03T11:39:54.524Z"
id_telegram: 7842296896
id_NFC: "88:109:194:73"
```

En la colección de reservas se almacenan los datos de las peticiones de sala con el siguiente formato:

```
_id: "61fc257efe7923028c1d8cf3"
id_telegram: 7842296896
nombre: "María Hervás"
id_NFC: "123"
sala: "CNC"
fecha: "3/2/2022"
hora_inicio: 20
hora_fin: 21
fecha_creacion: "2022-02-03T18:57:02.757Z"
```

Los datos se almacenan en formato JSON y se incorpora en cada objeto la fecha antes de ser almacenados por si fuera necesario recuperar los datos de un intervalo de tiempo concreto.

### 3.3 Robustez del sistema

#### 3.3.1 Bot de Telegram

En el Bot de Telegram existen varias comprobaciones en cada paso de la reserva que garantizan que no hayan reservas erróneas. Al ser esta la única manera de reservar, aseguramos que no haya posibilidad de solape, duplicación o formato erróneo en las reservas de la base de datos.

Haciendo uso de la verificación de identidad a través del uso de la contraseña proporcionada en el registro físico, se asegura que sólo el usuario registrado correspondiente pueda asociar su id de Telegram.

#### 3.3.2 Dispositivo de control de acceso

Respecto a los dispositivos de las puertas, se emplea un subconjunto de identificadores guardados en memoria a modo de cache, de manera que si hay algún fallo en la red, los usuarios más probables (o el administrador) pueden seguir entrando con la tarjeta. Además, el código del microcontrolador revisa el cierre de la puerta cada 2 segundos, para que en ningún caso la puerta se quede abierta si ocurre algún evento inesperado. También incluye debugging a través del puerto serie para poder diagnosticar posibles fallos con alguna tarjeta, ya que el módulo RFID nos permite saber si una tarjeta puede ser leída o no.

### 3.4 Posibles mejoras y trabajo futuro

En la interfaz de Telegram se podrían implementar diversas mejoras: un mejor sistema de visualización de las reservas integrando Google Calendar, más comandos con más funciones disponibles, cambiar el modo de interacción de comandos por botones y menús. Para optimizar el uso de la maquinaria y dado que las impresiones en 3D suelen ser bastante largas, se considera la opción de habilitar reservas hasta después de la hora de cierre.

Para evitar el acceso o reserva de usuarios con una suscripción caducada, sería conveniente implementar la comprobación de la validez de la suscripción con el campo de la base de datos "fecha\_vencimiento". Tampoco se ha desarrollado una situación realista de la sala de impresión 3D ya que, en realidad, no hay una sola impresora sino varias. Se propone como trabajo futuro ampliar la posibilidad de reservas en más impresoras 3D.

Para mejorar el servicio proporcionado, se podría hacer uso de datos estadísticos como el número de accesos por usuario, número de intentos fallidos de acceso, número de horas reservadas por máquina o por usuario o histórico de aforo en cada sala entre otros. De esta forma pueden ajustarse el tipo de suscripciones o aumentar el número de recursos, por ejemplo. Estos datos podrían almacenarse en una colección de la base de datos y acceder a ellos fácilmente. Además, con respecto a la base de datos, sería oportuno habilitar un sistema de búsqueda y borrado o modificación de objetos dentro de la base de datos. En esta versión del proyecto solo se ofrece la posibilidad de incluir objetos o borrar toda la base de datos.

Por otro lado, para la implementación real sería necesario usar una cerradura electrónica, a la que le añadiríamos el dispositivo de control de acceso. Se podría añadir una interfaz gráfica a los dispositivos de control de acceso, instalando una pantalla. Así, si el acceso a un usuario ha sido denegado, se le puede mostrar en el momento cual ha sido el problema y que debe hacer si necesita acceder.

También sería conveniente añadir un sistema que detecte el paso de usuarios por las puertas, lo que permitiría controlar de manera fiable los aforos y la entrada a salas con más restricciones.

En cuanto a seguridad del dispositivo una encriptación punto a punto daría una mayor seguridad en la transmisión de datos.

### 3.5 Librerías

Para llevar a cabo el proyecto se han integrado varias librerías, las cuales nos permiten usar funciones avanzadas que han sido desarrolladas previamente.

Para el desarrollo del Bot de Telegram se han usado las siguientes:

- **node-red**: librería básica de NodeRed.
- **node-red-contrib-chatbot**: librería para crear chatbots para Telegram, Facebook Messenger, Viber, Twilio y Slack, con gran variedad de funcionalidades.
- **node-red-node-mongodb**: librería usada para para la base de datos.



Por otro lado, en la programación del módulo de control de acceso se han empleado las siguientes librerías:

- **Comunicación con el sensor RFID:**
  - SPI.h
  - MFRC522.h
- **Uso de datos formateados en JSON:**
  - ArduinoJson.h
- **Comunicación con el broker y gestión de actualizaciones:**
  - PubSubClient.h
  - ESP8266httpUpdate.h
  - ESP8266WiFi.h

## 4 Resultados y conclusiones

---

Tras la finalización del proyecto se proporciona un sistema funcional y con una robustez suficiente para su alcance actual, que cumple con los objetivos propuestos del proyecto, interactúa de forma correcta con el usuario siendo tolerante a errores en el origen de datos y proporciona feedback en caso de algún tipo de fallo en la interacción. Se desarrolla una interfaz intuitiva para usuarios sin conocimientos previos sobre el sistema, con operaciones rutinarias sencillas y explicadas de manera clara en el manual de usuario. El uso de Telegram como interfaz de usuario para la interacción y obtención de información se considera la óptima, al ser accesible de forma rápida desde cualquier dispositivo. Además, se implementa un sistema SCADA (supervisión, control y adquisición de datos) utilizando el paquete Dashboard para que el administrador pueda gestionar el servicio de la forma más eficiente.

En algunos aspectos, ésta implementación no es realista. No se utilizan cerraduras automáticas que aseguren que solo los usuarios autorizados puedan acceder a las salas, no se incluyen pantallas que informen al usuario sobre el proceso de acceso y no hay control del aforo real. Por este motivo, se considera que el proyecto necesita más etapas de desarrollo para ser considerado una versión final, completamente funcional e implementable.

Se ha intentado estructurar el proyecto de forma que sea fácilmente escalable y que sea posible aumentar el número de puertas con lector, habilitar más reservas o modificar las horas de apertura y cierre. También se mandan datos de acceso desde las puertas que, aunque en esta etapa del proyecto no sean utilizados, permiten la integración del almacenamiento y uso de datos estadísticos. Se adjunta toda la información necesaria para poder ampliar el servicio y todos los ficheros incluidos están debidamente comentados y descritos.

## 5 Manual de usuario

### 5.1 Manual del administrador de SkapaLab

#### 5.1.1 Puesta en marcha del sistema

Para instalar los dispositivos de control de acceso en cada puerta es necesario asignarles a cada uno un identificador. Para ello hay que definir el ID de cada módulo en el código del microcontrolador, además de configurar las credenciales para la conexión Wi-Fi y la dirección IP del del broker.

Automáticamente, los topics con los que se comunicara ese módulo en concreto comenzarán por *puerta\_ID/*, cambiando *ID* por *CNC* por ejemplo, si se quiere identificar la puerta del espacio de la CNC. Después será necesario usar esos mismos topics desde NodeRed, para controlar cada puerta independientemente.

Este sistema trabaja utilizando una máquina virtual que debe de ser instalada en el ordenador desde el que se quiera manejar el sistema. Se recomienda el uso de VirtualBox o VMWare, independientemente del sistema operativo. Una vez instalado el programa, se importa la máquina virtual adjunta al proyecto. Esta máquina virtual debe ejecutarse siempre que el sistema esté activo. Para poder acceder a los servicios de la máquina virtual, es importante configurar el reenvío de puertos en la configuración de la máquina (Figura 5). Para ello, habrá que modificar la IP anfitrión y colocar la IP de nuestro PC. La IP del PC puede encontrarse en el terminal de Windows introduciendo el comando *ipconfig* o desde Preferencias del Sistema > Red si se usa macOS.

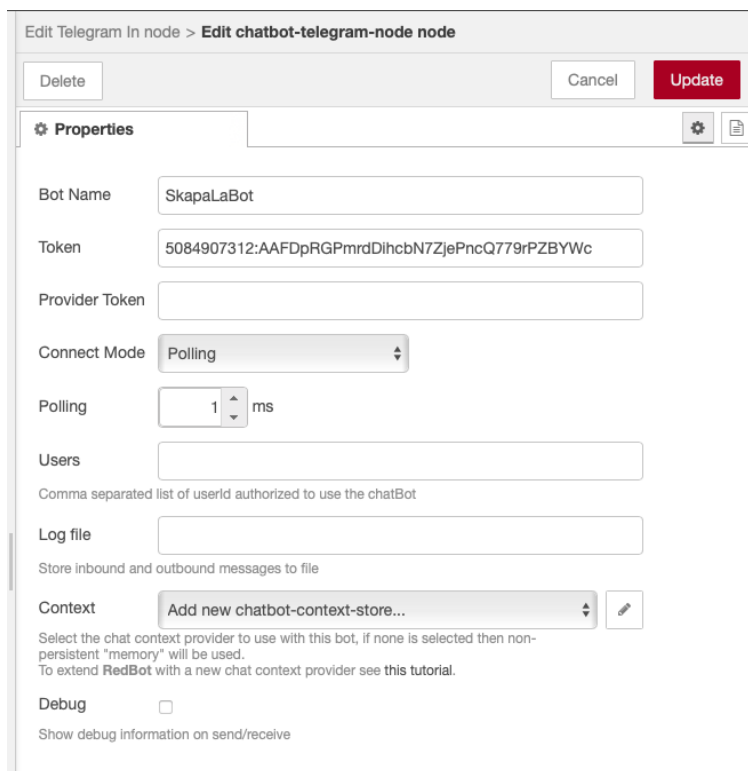
| Nombre | Protocolo | IP anfitrión | Puerto anfitrión | IP invitado | Puerto invitado |
|--------|-----------|--------------|------------------|-------------|-----------------|
| Rule 1 | TCP       | 192.168.1.15 | 1880             | 10.0.2.15   | 1880            |
| Rule 2 | TCP       | 192.168.1.15 | 1883             | 10.0.2.15   | 1883            |

Figura 5: Reenvío de puertos.

El siguiente paso será abrir NodeRed en el ordenador. NodeRed puede abrirse con cualquier navegador web tecleando en el buscador “localhost:1880/” o “IP anfitrión:1880” siendo IP anfitrión la IP utilizada para el reenvío de puertos. En caso de que no se abra NodeRed, comprobar que la IP en el reenvío de puertos sea correcta y que la máquina virtual esté en marcha. Una vez dentro de NodeRed, en el menú de la esquina superior derecha, pulsando en *Import* se accede a la pantalla de importación de archivos. Pulsando *Select files to import* se abrirá la carpeta de archivos del ordenador desde la que se deberá abrir el fichero .json adjuntado en el proyecto. Teniendo *Import to* con *New flow* seleccionado, se clic *import* y automáticamente se abrirán todos los flujos del proyecto. El fichero incluye cinco flujos de NodeRed:

- **SkapaLaBot:** Para el funcionamiento del Bot de Telegram.
- **Skapalab - USUARIOS:** Para el registro de nuevos usuarios en la base de datos y la visualización o borrado de estos registros.
- **Skapalab - RESERVAS:** Para la incorporación de nuevas reservas de salas en la base de datos y para borrar estas reservas.
- **Skapalab - VIS. RESERVAS:** Para la visualización de las reservas en el Dashboard.
- **Skapalab - mqtt:** Para el envío y recepción de mensajes entre NodeRed y las placas ESP8266 mediante el protocolo mqtt.

Para que el Bot funcione con NodeRed es importante asegurarse de que los nodos *Telegram Receiver* y *Telegram Sender* tengan la configuración correcta. Para ello, se hace doble click sobre el nodo a comprobar y haciendo click sobre el símbolo de editar a la derecha del primer desplegable que debe tener seleccionado SkapaLaBot. Debe aparecer en *Bot Name* “SkapaLaBot”, en *Token* “5084907312:AAFDpRGPMrdDihcbN7ZjePncQ779rPZBYWc”, en *Connect Mode* debe estar seleccionado “Polling”, y en *Polling*, 1 ms (Figura 6).



The image shows the configuration interface for a Telegram node in Node-RED. The title bar reads "Edit Telegram In node > Edit chatbot-telegram-node node". At the top, there are buttons for "Delete", "Cancel", and "Update". Below this is a "Properties" tab with a settings icon and a document icon. The configuration fields include: "Bot Name" (SkapaLaBot), "Token" (5084907312:AAFDpRGPMrdDihcbN7ZjePncQ779rPZBYWc), "Provider Token" (empty), "Connect Mode" (Polling), "Polling" (1 ms), "Users" (empty), "Log file" (empty), "Context" (Add new chatbot-context-store...), and a "Debug" checkbox. A note at the bottom states: "Select the chat context provider to use with this bot, if none is selected then non-persistent 'memory' will be used. To extend RedBot with a new chat context provider see this tutorial."

Figura 6: Configuración Bot.

Por otro lado, para asegurar que la base de datos y las conexiones MQTT funcionan, también es necesario comprobar que la configuración y autenticación son correctas. Para ello, en ambas usamos como usuario “II8” y como contraseña “kS00ooyj”. Además, para la conexión con el servidor, los nodos mqtt tienen que tener en el desplegable *Server* la dirección “iot.ac.uma.es:1883” y los nodos de la base de datos, “II8@iot.ac.uma.es”.

### 5.1.2 Registro de usuarios

El registro de un nuevo usuario se hace desde el Dashboard de NodeRed. El Dashboard se abre en el menú de la derecha de NodeRed. Se selecciona el panel *Dashboard* y se clicca el icono del cuadro con una flecha saliente ubicado en la parte superior. El Dashboard se compone de dos pestañas. El formulario de registro de usuarios se encuentra en la pestaña *Usuarios*.

Se deberá proporcionar un nombre completo (Nombre y Apellidos), un tipo de suscripción, la fecha de vencimiento de la suscripción y una contraseña para el registro. Además, con los switches se selecciona si se tiene acceso o no a las salas de CNC y de impresión 3D. Esto permitirá al usuario reservar esas salas o no. Se recomienda además que la contraseña sea breve y sencilla. Esta contraseña se usará una sola vez para vincular el id de Telegram (Figura 7).

**Registro**

**Registro Usuario**

Nombre Completo \*

☐ Arranque

☐ Entusiasta

☐ Maker

☐ Profesional

Fecha Vencimiento \*

03 02 2022

☐ Acceso CNC

☐ Acceso 3D

Contraseña \*

AÑADIR NFC CANCELAR

Figura 7: Registro de nuevo usuario.

Al pulsar el botón *Añadir NFC* se activará el lector de tarjetas NFC. Pasar por el lector la tarjeta que se quiera vincular a ese usuario y terminará el proceso de registro. Se puede cancelar el registro antes de activar el lector de tarjetas pulsando el botón *Cancelar*.

### 5.1.3 Visualización de reservas

El administrador tiene la posibilidad de visualizar las reservas activas en las distintas salas desde la pestaña *Reservas* en el Dashboard (Figura 8). Al pulsar el botón *Actualizar* aparecerán en pantalla las reservas y el estado de las salas.

| Reservas   |  |
|--|--|
| <p><b>Sala Impresión 3D</b></p> <p>Estado Actual 3D: No hay reservas en curso</p> <p><b>Más reservas hoy:</b></p> <p>Nombre: María<br/>Hora inicio: 17<br/>Hora fin: 19</p> <p>Nombre: María<br/>Hora inicio: 20<br/>Hora fin: 21</p> <p><b>Reservas próximos días:</b></p> <p>Nombre: María<br/>Fecha: 5/2/2022<br/>Hora inicio: 9<br/>Hora fin: 11</p> | <p><b>Sala CNC</b></p> <p>Estado Actual CNC: Hay reservas en curso</p> <p><b>Reserva Actual CNC:</b></p> <p>Nombre: María<br/>Hora inicio: 14<br/>Hora fin: 15</p> <p><b>Más reservas hoy:</b></p> <p><b>Reservas próximos días:</b></p> <p>ACTUALIZAR</p> <p>Última actualización hace: 3/2/2022 14:24:21</p> |

Figura 8: Dashboard.

## 5.2 Manual del usuario miembro de SkapaLab

Para obtener información de contacto y acceder a preguntas frecuentes sobre SkapaLab basta con comunicarse con el Bot de Telegram de SkapaLab. Para ello, en el buscador de Telegram se introduce “@SkapaLaBot” para encontrarlo por primera vez. Una vez encontrado el Bot, mandando el mensaje `/start` recibirás un resumen de los comandos que el Bot entiende.

Para poder reservar franjas horarias para usar las salas de CNC e Impresión 3D, necesitas estar registrado en la base de datos con una suscripción. Para ello, es necesario realizar el registro con el administrador. Tras completar el registro se te dará una tarjeta NFC. Una vez estés registrado, en Telegram, usando el comando `/asociarUsuario` y siguiendo las indicaciones del Bot podrás asociar tu usuario con tu cuenta de Telegram. Una vez hecho esto, el Bot te reconocerá como usuario registrado y tendrás acceso a los comandos de reserva.

Para acceder a una sala es necesario tener una reserva hecha a través del Bot. Puedes hacerlas escribiendo `/reservaCNC` o `/reservaImpresora3D`, y siguiendo las indicaciones del Bot. Una vez tengas la reserva, en la franja horaria reservada, podrás acceder a la sala seleccionada haciendo uso de la tarjeta que se te proporcionó en el registro con el administrador. Tan sólo es necesario acercarla al lector y una luz verde indicará que se ha abierto.

## 6 Ficheros entregados

---

- **SkapaLab\_NodeRed.json** : fichero en formato JSON que contiene los flujos de NodeRed del Bot de Telegram *SkapaLaBot*, así como la infraestructura de la base de datos y el Dashboard.
- **ubuntu\_iot\_vb.zip**: fichero para VirtualBox de la máquina virtual con el software necesario ya instalado (accesible desde el campus virtual).
- **ubuntu\_iot\_vmw.zip**: fichero para VMWare de la máquina virtual con el software necesario ya instalado (accesible desde el campus virtual).
- **ArduinoCode.zip**: carpeta comprimida con los códigos necesarios para el microcontrolador.
- [Repositorio en GitHub](#)