Departamento: Ingeniería e Investigaciones Tecnológicas

Programa de acreditación:
CyTMA2

Programa de Investigación[1]:

Código del Proyecto: C2-ING059

Título del proyecto:

Interacción entre sistemas basados en IOT y redes de datos dual stack

PIDC: ☐
Elija un elemento.

PII: ☐
Elija un elemento.

Director: Binker Carlos Alberto

Director externo:

Codirector: Tantignone Hugo Raúl

Integrantes: Buranits Guillermo, Romero Diego, Moreira Rubén Darío

Investigador Externo, Asesor- Especialista, Graduado UNLaM:

Alumnos de grado: (Aclarar si tiene Beca UNLaM/CIN)

Zurdo Eliseo Alfredo, Frattini Maximiliano

Alumnos de posgrado:

Resolución Rectoral de acreditación: N° 367/19

Fecha de inicio: 01/01/2019

Fecha de finalización: 31/12/2020

[1] Los Programas de Investigación de la UNLaM están acreditados con resolución rectoral, según lo indica la Resolución HCS Nº 014/15 sobre **Lineamientos generales para el establecimiento, desarrollo y gestión de Programas de Investigación a desarrollarse en la Universidad Nacional de La Matanza**. Consultar en el departamento académico correspondiente la inscripción del proyecto en un Programa acreditado.

A. Desarrollo del proyecto (adjuntar el protocolo)

Debido a la pandemia se dificultó la presentación de trabajos en congresos, ya que el material de trabajo nos quedó en su mayoría en la Universidad, dificultando así enormemente la tarea. No obstante aquí exponemos el desarrollo del proyecto en su fase final.

A.1. Grado de ejecución de los objetivos inicialmente planteados, modificaciones o ampliaciones u obstáculos encontrados para su realización

En relación con los objetivos planteados para el año 2020, se considera que los mismos se han alcanzado, a pesar de las dificultades relacionadas con la pandemia. Lamentablemente no fue un año sencillo. La mayor parte de los elementos adquiridos para el proyecto y el instrumental asociado requerido (multímetros, osciloscopio digital, etc.) han quedado en la Universidad y eso no nos permitió poder cumplimentar las tareas con la rigurosidad que hubiéramos deseado. No obstante ello, en la primera parte del año nos enfocamos al estudio de la "Interacción de dispositivos IOT con Internet". Se había propuesto analizar el *gns3* como software de simulación para dicho fin, pero la realidad es que se observó que no se adaptaba a los requerimientos de IOT. En cambio, se analizó otro software de simulación de redes de CISCO, que sí admite con creces la interacción con dispositivos IOT, permitiendo simular sensores y actuadores (que se pueden configurar en JavaScript y Python) y además como se utiliza en la materia *Redes de computadoras*, se concluyó que era una excelente oportunidad para transferir conocimiento hacia la cátedra. En el mismo eje temático de la simulación IOT, también con el Packet Tracer de CISCO, se realizó un estudio pero basado exclusivamente en IPv6, pensado como pilar del IOT dado el enorme crecimiento exponencial que está teniendo la interconexión de dispositivos, tales como sensores, actuadores, etc. En el anexo se amplía con más detalle la investigación realizada. También hemos publicado un artículo en la Revista Científica del Departamento de Ingeniería de la UNLaM. El mismo se expone también en el anexo con su constancia. Finalmente la última etapa del proyecto se cumplimentó también de manera exitosa. Se desarrolló un sistema domótico de control de iluminación del hogar. El mismo está basado en el ESP8266. Este popular SOC (System of Chip) se utilizó en conjunto con un relé, permitiendo así el empleo de iluminación hogareña a 220V. El chip concretamente utilizado es el ESP-01s y el módulo de relé es la placa ESP-01 Relay. Ambos dispositivos montados pueden observarse en la Figura 1.

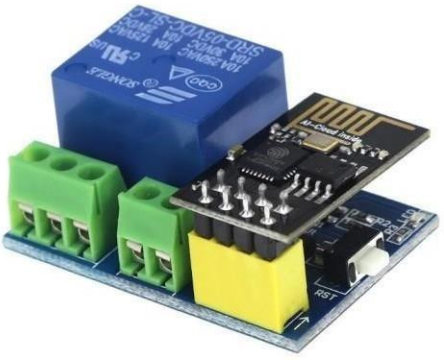

Figura 1 - Placa ESP-01s Relay y módulo ESP-01s

El encendido de las diferentes luminarias del hogar se lleva a cabo mediante un acceso web, a través de la red WIFI del hogar. Se implementaron dos escenarios:

1. **Control web interno en cada ESP-01s**. Cada dispositivo ESP-01s basado en el chip ESP8266, aloja un servidor web propio que controla una luminaria por medio de una dirección IP fija.

La página muestra un botón que permite encender o apagar la luz. Cada dispositivo se ha montado en una caja plástica contenedora, para que los circuitos impresos no toquen la caja metálica embutida donde irá insertado el módulo WIFI. Se ha construido la caja empleando una impresora 3D. Se ha logrado algo muy funcional y práctico, ya que como valor agregado, esta pequeña caja incorpora también la fuente de alimentación (5V DC, a partir de los 220 V de línea), pudiendo llegar a constituirse en un producto comercial. Además el sistema permite controlar la iluminación de la manera tradicional a través de una tecla o bien mediante WIFI, por ejemplo desde un teléfono móvil. Además, apelando al concepto de circulación dentro del hogar, se permite controlar una carga desde dos posiciones diferentes, una fija mediante una tecla de combinación y la otra posición móvil desde otro punto de la casa mediante la red WIFI, ya sea desde un celular, tablet, smart tv, etc. El módulo WIFI y la llave se conectan formando el característico circuito de llave de combinación, ver más detalle en el anexo.

A su vez, este escenario admite dos configuraciones posibles:

a) Llave de combinación en caja separada a donde se encuentra el módulo WIFI
b) Llave de combinación en la misma caja en donde se encuentra el módulo WIFI

El caso **a)** obedece a la situación real en donde ya en una casa existen llaves de combinación en cajas diferentes, mientras que el caso **b)**, simula la condición anterior, pero tanto el módulo WIFI como la llave de combinación se montan en la misma caja, dando la versatilidad al sistema de control manual o control inalámbrico vía web.

2. **Control web externo basado en la nube**. En este caso se empleó un servicio gratuito de hosting a través del sitio www.000webhost.com. En dicho sitio, se albergó una página web conectada a una base de datos SQL. Esta base de datos contendrá los botones de comando para el control de la iluminación. También el hecho de tener una base de datos nos habilita, por ejemplo, a leer información de sensores y enviarlos a la base de datos para su almacenamiento y posterior utili- zación, pero estas acciones quedarán pendientes para el siguiente proyecto. En este caso se em- pleó un Arduino Mega y un solo ESP-01s para vincular la placa a Internet. A diferencia del esce- nario anterior, debe instalarse el Arduino Mega en algún sector de la casa y desde allí realizar todas las conexiones hacia las cargas a controlar, es decir desde cada pin GPIO del Arduino Mega, deberá salir un cable de control que es el que activará el relé para el encendido de cada luminaria, lo cual dificulta la instalación eléctrica del hogar, y esto no es siempre factible, ya que por lo general las casas tienen caños de 5/8 de pulgada que dificulta la introducción de la cinta pasa- cables. En este caso los módulos a emplear se pueden ver en la Figura 2.

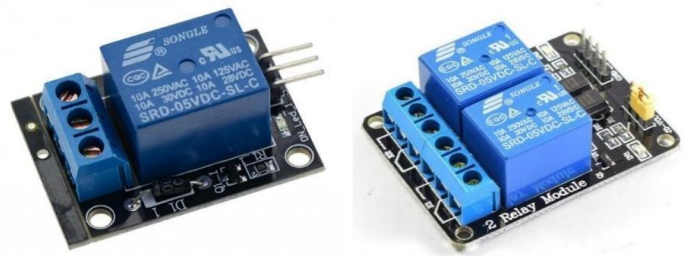

Figura 2 – Placa para comandar una o dos cargas (luminarias en nuestro caso)

B. Principales resultados de la investigación

B.1. Publicaciones en revistas

Publicación N° 1	
Autores	Carlos Binker, Hugo Tantignone, Eliseo Zurdo, Guillermo Buranits, Diego Romero, Rubén Darío Moreira, Maximiliano Frattini
Título del artículo	ACCESO REMOTO A DISPOSITIVOS IOT MEDIANTE TÉCNICAS DE MENSAJERÍA EMPLEANDO BOTS Y FREERTOS
N° de fascículo	1 (uno)
N° de Volumen	5 (cinco)
Revista	ReDDI – Revista Digital del Departamento de Ingeniería
Año	2020
Institución editora de la revista	Universidad Nacional de La Matanza
País de procedencia de institución editora	Argentina
Arbitraje	NO
ISSN:	2525-1333
URL de descarga del artículo	https://reddi.unlam.edu.ar/index.php/ReDDi/article/view/117
N° DOI	

B.2. Libros

Libro 1	
Autores	
Título del Libro	
Año	
Editorial	
Lugar de impresión	
Arbitraje	Elija un elemento.
ISBN:	
URL de descarga del libro	
N° DOI	

B.3. Capítulos de libros

Autores	
Título del Capitulo	
Título del Libro	
Año	
Editores del libro/Compiladores	
Lugar de impresión	
Arbitraje	Elija un elemento.

B.4. Trabajos presentados a congresos y/o seminarios

Autores	
Título	
Año	
Evento	
Lugar de realización	
Fecha de presentación de la ponencia	
Entidad que organiza	
URL de descarga del trabajo (especificar solo si es la descarga del trabajo; formatos pdf, e-pub, etc.)	

B.5. Otras publicaciones

Autores	
Año	
Título	
Medio de Publicación	

C. Otros resultados. Indicar aquellos resultados pasibles de ser protegidos a través de instrumentos de propiedad intelectual, como patentes, derechos de autor, derechos de obtentor, etc. y desarrollos que no pueden ser protegidos por instrumentos de propiedad intelectual, como las tecnologías organizacionales y otros. Complete un cuadro por cada uno de estos dos tipos de productos.

C.1. Títulos de propiedad intelectual. Indicar: Tipo (marcas, patentes, modelos y diseños, la transferencia tecnológica) de desarrollo o producto, Titular, Fecha de solicitud, Fecha de otorgamiento

Tipo	Titular	Fecha de Solicitud	Fecha de Emisión

C.2. Otros desarrollos no pasibles de ser protegidos por títulos de propiedad intelectual. Indicar: Producto y Descripción.

C.2.1

Producto	Descripción
Control de iluminación WIFI – Escenario 1	Ver Anexo I, punto C.2.1

C.2.2

Producto	Descripción
Control de iluminación WIFI – Escenario 2	Ver Anexo I, punto C.2.2

C.2.3

Producto	Descripción
Informe simulación IOT	Ver Anexo I, punto C.2.3

D. Formación de recursos humanos. Trabajos finales de graduación, tesis de grado y posgrado. Completar un cuadro por cada uno de los trabajos generados en el marco del proyecto.

D.1. Tesis de grado

Director (apellido y nombre)	Autor (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.2 Trabajo Final de Especialización

Director (apellido y nombre)	Autor (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título del Trabajo Final

D.3 Tesis de posgrado: Maestría

Director (apellido y nombre)	Tesista (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.4 Tesis de posgrado: Doctorado

Director (apellido y nombre)	Tesista (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título de la tesis

D.5 Trabajos de Posdoctorado

Director (apellido y nombre)	Posdoctorado (apellido y nombre)	Institución	Calificación	Fecha /En curso	Título del trabajo	Publicación

E. Otros recursos humanos en formación: estudiantes/ investigadores (grado/posgrado/ posdoctorado)

Apellido y nombre del Recurso Humano	Tipo	Institución	Período (desde/hasta)	Actividad asignada[2]
Eliseo Zurdo	Alumno	UNLAM	01/01/20 – 31/12/20	**1.** Participó en la ETAPA 4 y 5. Ver GANTT FPI-002. **2.** En la ETAPA 5 se abocó a lo concerniente a la programación de dispositivos IOT, dado que ahí se destaca su mayor conocimiento. Ver GANTT FPI-002.
Maximiliano Frattini	Alumno	UNLAM	01/01/20 – 31/12/20	**1.** Participó en la ETAPA 4 y 5. Ver GANTT FPI-002. **2.** En la ETAPA 5 se abocó a lo concerniente a la programación de dispositivos IOT, dado que ahí se destaca su mayor conocimiento. Ver GANTT FPI-002.

F. Vinculación[3]: Indicar conformación de redes, intercambio científico, etc. con otros grupos de investigación; con el ámbito productivo o con entidades públicas. Desarrolle en no más de dos (2) páginas.

G. Otra información. Incluir toda otra información que se considere pertinente.

H. Cuerpo de anexos:

- Anexo I: Copia de cada uno de los trabajos mencionados en los puntos B, C y D, y certificaciones cuando corresponda.[4]
- Anexo II:
  - FPI-013: Evaluación de alumnos integrantes. (si corresponde)
  - FPI-014: Comprobante de liquidación y rendición de viáticos. (si corresponde)
  - FPI-015: Rendición de gastos del proyecto de investigación acompañado de las hojas foliadas con los comprobantes de gastos.
  - FPI-035: Formulario de reasignación de fondos en Presupuesto.
- Anexo III: Alta patrimonial de los bienes adquiridos con presupuesto del proyecto (FPI 017)
- Nota justificando baja de integrantes del equipo de investigación.

Carlos Alberto Binker

_____ _____

Firma y aclaración
del director del proyecto.

Lugar y fecha : San Justo, 17/2/21

[2] Descripción de la/s actividad/es a cargo (máximo 30 palabras)

[3] Entendemos por acciones de "vinculación" aquellas que tienen por objetivo dar respuesta a problemas, generando la creación de productos o servicios innovadores y confeccionados "a medida" de sus contrapartes.

[4] En caso de libros, podrá presentarse una fotocopia de la primera hoja significativa o su equivalente y el índice.

Código	FPI-009
Objeto	Guía de elaboración de Informe de avance de proyecto
Usuario	Director de proyecto de investigación
Autor	Secretaría de Ciencia y Tecnología de la UNLaM
Versión	5
Vigencia	03/9/2019

ANEXO I

B.1
Revista digital del Departamento de Ingeniería ReDDI

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.: XX - Nro. XX (MES-AÑO)

[Tipo de documento: *artículo original*]

ACCESO REMOTO A DISPOSITIVOS IOT MEDIANTE TÉCNICAS DE MENSAJERÍA EMPLEANDO BOTS Y FREERTOS

REMOTE ACCESS TO IOT DEVICES THROUGH MESSAGING TECHNIQUES USING BOTS AND FREERTOS

Carlos Alberto BINKER [1], Hugo TANTIGNONE [1], Eliseo Alfredo ZURDO [1], Guillermo Buranits[1]

[1]Departamento de Ingeniería e Investigaciones Tecnológicas. UNLAM
cbinker@unlam.edu.ar
htantignone@unlam.edu.ar
ezurdo@alumno.unlam.edu.ar
gburanits@unlam.edu.ar

Resumen:

Este trabajo tiene como eje principal gestionar un hardware IOT remotamente mediante mensajería instantánea de texto, empleando Telegram. Telegram permite definir bots (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 (el padre de todos los bots). Telegram brinda una API , que permite que los bots interactúen con nuestro sistema. En nuestro caso, esa interacción se dará para controlar un microcontrolador que opera con dispositivos IOT, el destacado chip ESP32. El ESP32 es un SOC (System on chip). Este SOC incorpora WIFI, bluetooth, sensores, conversores AD y DA, etc. Como caso de estudio se propone controlar un conjunto de leds, empleando el ESP32 usando un bot, por lo que se requerirá el uso de Telegram (versión web o móvil) y el entorno de programación Arduino. Además como valor agregadose hará uso de multitarea empleando ambos núcleos del SOC utilizando técnicas FreeRTOS.

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

Abstract:

The focus of this work is to manage an IOT hardware remotely through instant text messaging, using Telegram. Telegram allows you to define bots (apheresis of Robots) and manage them through a token that the same application randomly generates through its main bot, called BotFather (the father of all bots). Telegram provides an API, which allows bots to interact with our system. In our case, that interaction will occur to control a microcontroller that operates with IOT devices, the outstanding ESP32 chip. The ESP32 is a SOC (System on chip). This SOC incorporates WIFI, Bluetooth, sensors, AD and DA converters, etc. As a case study, it is proposed to control a set of LEDs, using ESP32 using a bot, so the use of Telegram (web or mobile version) and the Arduino programming environment will be required. In addition, as an added value, multitasking will be used using both SOC cores using FreeRTOS techniques.

Palabras Clave: *Mensajería, Robots, Sensores, Internet de las cosas, API*

Key Words: *Telegram, Bots, Sensors, IOT, API*

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

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 FIGURA. 1. Para evitar esto, ya que no siempre es fácil tener una dirección IP pública sepropone la solución de mensajería por medio de Telegram mediante el empleo de bots. En nuestro caso nuestro host es un SOC ESP32, que a través de su IP accedemos a él a través de una red WIFI conectada a Internet. El SOC estará configurado en modo STATION y tomará su dirección IP mediante DHCP.

2. Descripción de la tecnología de mensajería Telegram y del SOC ESP 32

2.1 Esquema de la plataforma de mensajería instantánea Telegram

Telegram está creado sobre una serie de servidores distribuidos. Dichos servidores, para comunicarse entre sí, utilizan un protocolo propio llamado MTProto. 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. 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 FIGURA 2.

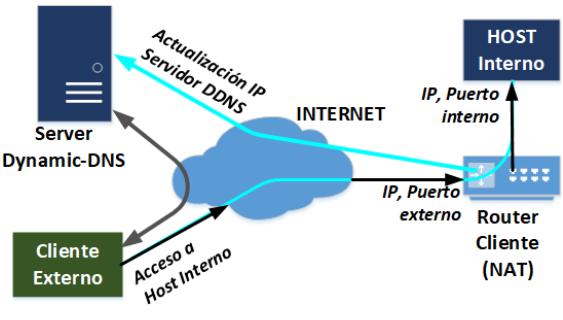

FIGURA 1 - Esquema de un servicio NO-IP usando una técnica de DNS dinámico

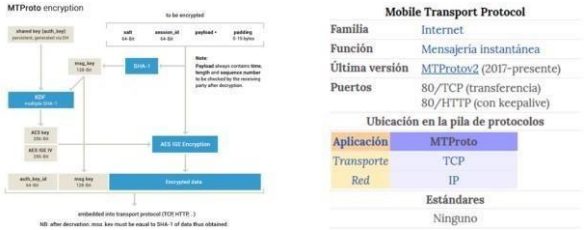

FIGURA 2 - Encriptación de mensajes Telegram mediante SHA-1. Ubicación del MTPro en el stack de protocolos de comunicaciones

2.2 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 como 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 creará un bot que suministra los comandos anteponiendo el símbolo "/". Dicho bot se denomina CONAIISI_2019. A título de ejemplo con el token suministrado para CONAIISI_2019, la forma de acceder desde la web sería:

https://api.telegram.org/botxxxx32320:AAGgmhH6_H69CiQcbr4hPS-CZs0M18oxxxx/getme

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

{"ok":true,"result":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"}
}

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"}

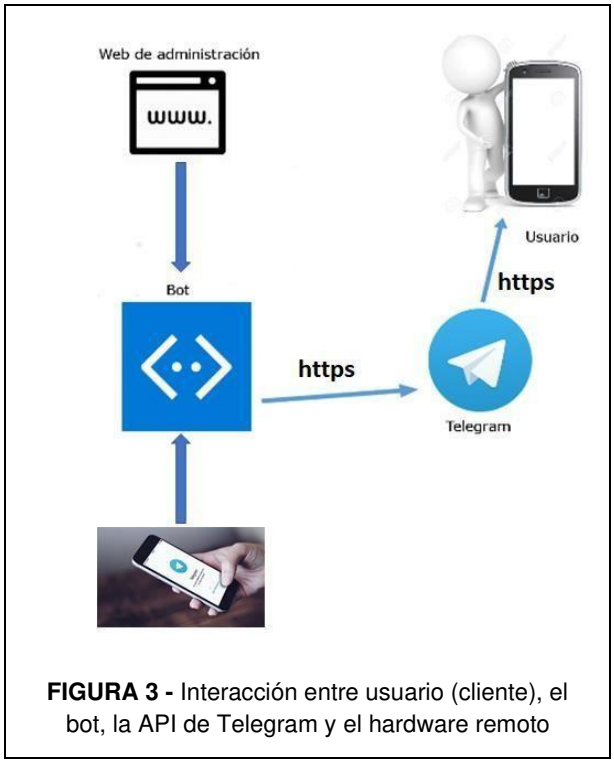

FIGURA 3 - Interacción entre usuario (cliente), el bot, la API de Telegram y el hardware remoto

En la FIGURA 3 se puede observar claramente la interacción entre todos los componentes, es decir entre el

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

cliente, el bot, la API de Telegram y el hardware remoto a controlar

2.3 Descripción del SOC ESP32

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, Ethernet, etc. Además, incorpora un sensor que mide la temperatura interna del dispositivo y sensores Touch capacitivos y de efecto Hall, además de conversores DAC y ADC. Posee un microprocesador con dos núcleos paralelo. Cualquier pin del ESP32 puede configurarse para el control de interrupciones. Para más detalle se expone el diagrama en bloques para dar una idea general de su potencialidad, como así también el correspondiente PIN-OUT, (ver FIGURA 4 y FIGURA 5 respectivamente).

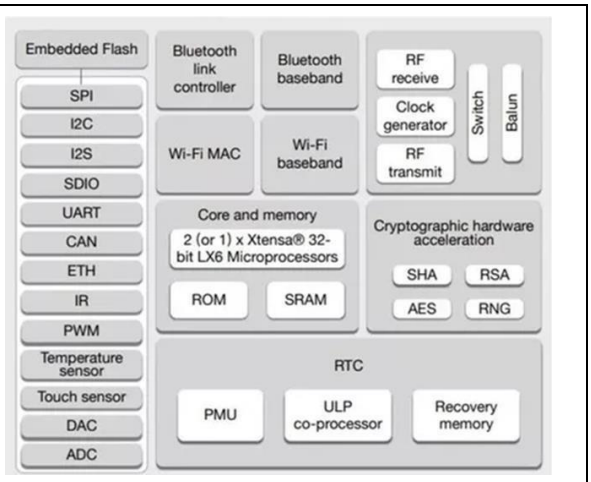

FIGURA 4 - Diagrama en bloques del ESP32

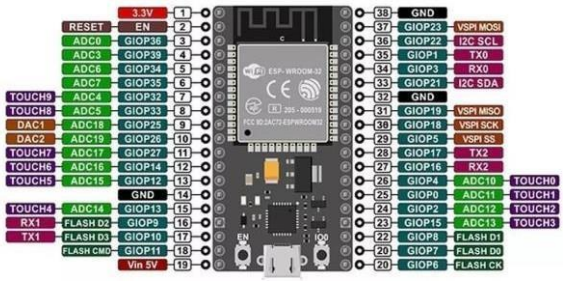

FIGURA 5 – Pin out ESP32. Placa Nodemcu ESP 32 DEV Module

3. Descripción de un Sistema RTOS

Un Sistema RTOS es un sistema operativo de tiempo real. Existen muchas aplicaciones en las cuales los tiempos de respuesta son un factor decisivo para el correcto funcionamiento, como por ejemplo los mandos de un avión o el sistema de accionamiento ABS de frenado en un automóvil son ejemplos más que ilustrativos de la exigencia requerida. El concepto es simple, dos tareas que pueden correr o no en un mismo núcleo realizan operaciones diferentes. Para conectar dichas tareas es necesario la existencia de una cola que transfiera los datos de una tarea a la otra. Por ejemplo una tarea A es productora de datos y otra tarea B es consumidora de los datos que genera la tarea A. Dichos datos se envían a

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

través de una cola que deberá crearse previamente. A continuación se ilustra el procedimiento descripto en la FIGURA 6.

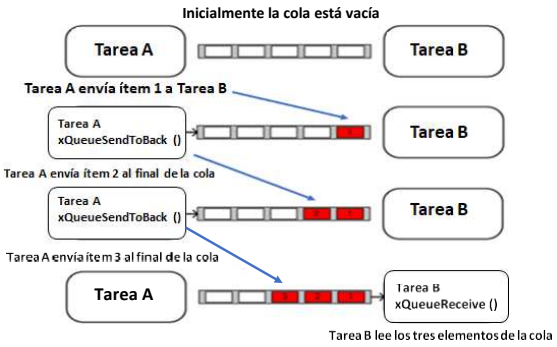

FIGURA 6 – Envío de datos de una tarea a otra a través de una cola

xQueueCreate
[Queue management]

queue.h

```
QueueHandle_t xQueueCreate( UBaseType_t uxQueueLenght,
                            UBaseType_t uxITemSize ) ;
```

xQueueSend
[Queue Management]

queue.h

```
BaseType_t xQueueSend( QueueHandle_t xQueue,
                       const void * pvItemToQueue,
                       TickType_t xTicksToWait
                     );
```

xQueueReceive
[Queue Management]

queue.h

```
Base_Type xQueueReceive(
                          QueueHandle_t xQueue,
                          void * pvBuffer,
                          TickType_t xTicksToWait
                        ) ;
```

FIGURA 7 – Funciones xQueueCreate, xQueueSend y xQueueReceive, elementos básicos de RTOS.

Existen 3 funciones principales básicas que son las quese utilizarán en toda experiencia de RTOS: Las funciones son: *xQueueCreate*, *xQueueSend* y *xQueueReceive*. La descripción de las mismas se detalla en la FIGURA 7.

Finalmente y yendo al caso concreto del ESP32, el framework de Arduino ya viene montado sobre un sistema operativo RTOS en donde las funciones setup() y loop() se implementan de la siguiente manera:

main.cpp

```
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp-task-wdt.h"
#include "Arduino.h"
    TaskHandle-t loopTaskHandle = NULL;
    #if CONFIG_AUTOSTART_ARDUINO
    bool loopTaskWDTEnabled;
    void loopTask(void *pvParameters)
    {

       setup();
       for(;;) {
          if(loopTaskWDTEnabled){
             esp-task-wdt-reset();
          }
          loop();
       }
    }
    extern "C" void app-main()
    {
       loopTaskWDTEnabled = false;
       initArduino();
      xTaskCreateUniversal(loopTask, "loopTask", 8192,
NULL,          1,          &loopTaskHandle,
CONFIG_ARDUINO_RUNNING_CORE);
    }
    #endif
```

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

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

t.me/c2ing59_bot

(acceso al bot por comandos utilizando "/"), donde en este caso c2ing59_bot es el username del bot y se corresponde con el Alias (@c2ing59_bot).

4. Caso de estudio empleando el SOC ESP32, Telegram y el IDE de Arduino con FreeRTOS

4.1 Prototipo a emplear en el caso de estudio

El prototipo de hardware que se utilizará es el mostrado a continuación en la FIGURA 8:

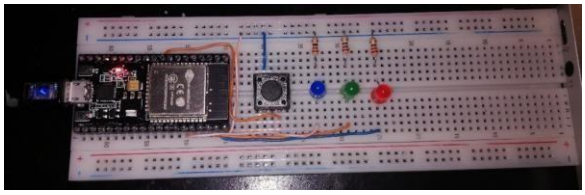

FIGURA 8 – Hardware a controlar por el bot de Telegram empleando el NodeMCU ESP32

FIGURA 9 – Bot que controlará el hardware (CONAIISI_2019), version móvil por comandos

FIGURA 10 – Bot que controlará el hardware (CONAIISI_2019), version web por comandos

El bot que lo controlará tiene el siguiente aspecto (ver FIGURA 9 y FIGURA 10), el mismo fue configurado a través de los comandos del BotFather, y lo apreciamos tanto en su versión web cómo móvil.

5. Desarrollo de la experiencia

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:

5.1 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.

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

5.2 Variable de entorno para el sensor de temperatura interno del ESP32

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

5.3 Definición de variables globales y constantes:

```
// Defino un PULSADOR en el GIOP 25, el cual se
// corresponde con el pin 9 del ESP32 Dev Module
const int pulsador=25;
// Defino los leds en los pines 23, 22 y 21, los cuales se
// corresponden con los pines 37, 36 y 33 del ESP32 Dev
// Module
const int led_rojo=23;
const int led_verde=22;
const int led_azul=21;
// Defino las constantes a emplear en el switch (ver
Tarea2)
#define ROJO 23
#define VERDE 22
#define AZUL 21
#define PRENDER  1
#define APAGAR   -1
//Estas variables determinarán el status del sistema a
//controlar, es decir qué leds están encendidos o apagados
byte rojo = 0;
byte verde = 0;
byte azul = 0;
//Defino el Descriptor para la Cola de Mensajes
QueueHandle_t xqueue;
//Defino el Descriptor para Tarea2
TaskHandle_t TaskHandle2 = NULL;
```

5.4 Código para manejo de interrupciones:

```
portMUX-TYPE mux =
portMUX_INITIALIZER_UNLOCKED;
// Cuando se presiona el pulsador esta variable pasa a true
bool pirTriggered = false;
//Código de la rutina de interrupción del pulsador por
//flanco descendente “Rising”
void IRAM_ATTR handleInterruptRising()
{
Serial.println("handleInterruptRising->"+
String(xPortGetCoreID()));
portENTER_CRITICAL_ISR(&mux);
Serial.println("Rising");
// Al ingresar a este rutina pirTriggered se pone a true
  pirTriggered = true;
  portEXIT_CRITICAL_ISR(&mux); }
// El chatId correspondiente al alias del usuario que
// interactúa con el bot
long chatId;
// Mensaje a enviar al bot
String msg = "Pulsador presionado";
```

5.5 Credenciales WIFI e instanciación de objetos

```
    //Instanciación de las credenciales para la conexión
WIFI //(ssid y password)

    const char* ssid     = " IPLAN-306232 ";
    const char* password = "xxxxxxxxx";

    //Instanciación del Bot de Telegram dando el token
//suministrado por BotFather

    //Nombre de Bot: CONAIISI_2019
```

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

```
const String botToken = "xxxx32320:AAGgmhH6_H69CiQcbr4hPS-CZs0M18oxxxx";

//Instanciación del cliente ssl utilizado para comunicarse //con la web API de Telegram

WiFiClientSecure sslPollClient;

//Instanciación de la clase client pasando como //argumentos el Token del Bot y un cliente seguro ssl

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

5.6 Declaración de función destinada a recibir mensajes desde el Bot de Telegram

```
void onReceive (TelegramProcessError tbcErr, JwcProcessError jwcErr, Message* msg)
```

Esta función corresponde a un evento que se disparará ante un comando suministrado al bot. Dicho evento funcionará en el núcleo 1. A tal efecto se imprime en el monitor serie dicho suceso con el siguiente comando:

```
Serial.println("onReceive En núcleo -> " + String(xPortGetCoreID()));
```

La variable xPortGetCoreID(), suministra el núcleo en donde se está ejecutando una tarea determinada. A continuación se expone como ejemplo el encendido del led verde:

```
if (msg->Text == "/verde-on" )
{
  client.postMessage(msg->ChatId, String("Encendiendo led verde"));
  SendMessage(VERDE * PRENDER);
  verde = 1;
 }
```

La función SendMessage() envía un mensaje desde una tarea que opera en un determinado núcleo hacia otra tarea que puede o no operar en un núcleo diferente. Para ello, se emplea una cola que opera como interface entre ambas tareas. En este caso la tarea a realizar es la de prender o apagar un led (ya que el manejo del hardware se realiza en el núcleo 0 mientras que el manejo de Telegram y la funcionalidad concerniente a WiFi se procesa en el núcleo 1.

Otras tareas a realizar en onReceive()

```
// Control de status acerca de los leds
 if (msg->Text == "/status" )    {
  if (!rojo && !verde && !azul)
    client.postMessage(msg->ChatId, String("Led rojo apagado\nLed verde apagado\nLed azul apagado"));
  else if (!rojo && !verde && azul)
  client.postMessage(msg->ChatId, String("Led rojo apagado\nLed verde apagado\nLed azul encendido\n"));
  else if (!rojo && verde && !azul)
  client.postMessage(msg->ChatId, String("Led rojo apagado\nLed verde encendido\nLed azul apagado\n"));
  else if (!rojo && verde && azul)
  client.postMessage(msg->ChatId, String("Led rojo apagado\nLed verde encendido\nLed azul encendido\n"));
  else if (rojo && !verde && !azul)
  client.postMessage(msg->ChatId, String("Led rojo encendido\nLed verde apagado\nLed azul apagado\n"));
  else if (rojo && !verde && azul)
  client.postMessage(msg->ChatId, String("Led rojo encendido\nLed verde apagado\nLed azul encendido\n"));
  else if (rojo && verde && !azul)
```

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

```
    client.postMessage(msg->ChatId, String("Led rojo encendido\nLed verde encendido\nLed azul apagado\n"));
   else
    client.postMessage(msg->ChatId, String("Led rojo encendido\nLed verde encendido\nLed azul encendido\n"));
  }
//Envío de un mensaje al bot en donde indica la //temperatura interna del ESP32 expresada en °C y °F ante //la petición /tempC o /tempF

if (msg->Text == "/tempc" ) client.postMessage(msg->ChatId, String((temprature_sens_read() - 32) / 1.8) + String(" °C"));

 if (msg->Text == "/tempf" ) client.postMessage(msg->ChatId, String((temprature_sens_read()) + String(" °F")));
```

5.7 Declaración de función invocada en el caso de que sucedan errores

```
Void onError (TelegramProcessError tbcErr, JwcProcessError jwcErr)
```

5.8 Empleo de las Funciones SendMessage() y ReceiveMessage()

5.8.1 SendMessage()

```
   void SendMessage( int message ) {

   //Envío al monitor serie para ver en que núcleo se está //ejecutando la función

   Serial.println("SendMessage(" + String(message) + ") En núcleo -> " + String(xPortGetCoreID()));
   if ( xqueue != 0 )
    {
   //Envía una tarea a la cola xqueue (ver 5.3), en este caso //corresponde al encendido o apagado de los leds.

   xQueueSend( xqueue, ( void * ) &message, ( TickType_t ) 0 ); }
    }
```

5.8.2 ReceiveMessage()

```
int ReceiveMessage()
{

 int msg = 0;
 if ( xqueue != 0 )
 {
//Receive a message on the created queue. Block for 10 //ticks if a message is not immediately available.

  if ( xQueueReceive( xqueue, &( msg ), ( TickType_t ) 10 ) )
   {
   }
 }
 Serial.println("ReceiveMessage(" + String(msg) + ") En núcleo -> " + String(xPortGetCoreID()));
 return msg;
}
```

5.9 Declaración del setup() y del setup2()

En el setup() se configurarán los valores iniciales correspondientes al núcleo 1, estas tareas corresponderán al manejo del WiFi del dispositivo ESP32, y al llamado al evento onReceive() que es el encargado de procesar las

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

peticiones del bot de Telegram. También acá se indican

las tareas a desarrollarse en los distintos núcleos del dispositivo.

5.9.1 setup()

```
void setup() {

//Arranco el puerto serial ante todo, para que al
//imprimir ya esté activo

Serial.begin(115200);

Serial.println("setup En núcleo -> " +
String(xPortGetCoreID()));
setupWiFi(); // Enciendo el WIFI.
//Creo Cola de Mensajes para la comunicación entre
//procesos con 100 lugares

xqueue = xQueueCreate(100, sizeof( int ));

 if (xqueue == NULL) {
  Serial.println("Error creating the queue");
  exit(-1);
 }


//Implementación del método begin del objeto client,
para //las denominadas “callback functions” a las que el
cliente //llamará ante la recepción de datos o de un error.

client.begin(onReceive, onError);

//Declaración de la tarea a realizarse en el núcleo 0

//Task2 - Loop en donde se ejecutará la tarea.
//"loopTask2" - Descripción de la tarea, se ejecutará en un
//nuevo loop infinito dado precisamente por for (;;)
//4096 - Tamaño del stack
//NULL - Parámetro a pasarle a la tarea
//1 - Prioridad
//&TaskHandle2 - Definida como variable global del
tipo //TaskHandle-t, ver 5.3)
//0 - Núcleo en el que se ejecutará la tarea
xTaskCreateUniversal(Task2, "loopTask2", 4096,
NULL, 1, &TaskHandle2, 0);
}
```

5.9.2 setup2()

```
void setup2() {
Serial.println("setup2 En núcleo -> " +
String(xPortGetCoreID()));
 setupLEDs();
 setupPULL();
 }
```

Las funciones setupLEDs() y setupPULL() aparecen declaradas previamente en el archivo fuente. Expondremos sólo la función setupPULL() por tratarse del manejo de una interrupción.

```
void setupPULL()
{
Serial.println("setupPULL En núcleo -> " +
String(xPortGetCoreID()));
pinMode(pulsador, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(pulsador),
handleInterruptRising, RISING);
}
```

6. Funciones loop() y Task2()

6.1 loop()

```
//Loop Principal de la Arduino. Se encargará del manejo
//de Wifi y Telegram.
void loop()
{
Serial.println("loop En núcleo -> " +
String(xPortGetCoreID()));
delay(300);
//Pregunto si se disparó un evento, en este caso si se
//oprimió un pulsador.
```

```
if (pirTriggered) {
Serial.println("Se presionó el pulsador En núcleo -> " +
String(xPortGetCoreID()));
// Le paso el ChatId del usuario que interactúa con el bot
client.postMessage(chatId, msg);
pirTriggered = false;
//pirTriggered en true significa que se oprimió el pulsador
  }
// Para procesar los datos recibidos, este método deberá ser
//invocado en forma continua dentro del main loop.
  client.loop();
}
```

6.2 Task2()

```
//Se encarga del control del Hardware sobre el Núcleo  0,
//esto es sobre los actuadores, que en este caso de
//simulación son los leds, y el control del pulsador.
void Task2(void *pvParameters)
{
  setup2(); // NOTAR que se ejecuta solo una vez
//A continuación forzamos el loop infinito con un for
for (;;) {
Serial.println("loop En núcleo -> " +
String(xPortGetCoreID()));
   delay(300);

   int msj = 0;
   msj = ReceiveMessage();
switch ( msj )
   {
     case (ROJO):
       digitalWrite(led_rojo, HIGH);
Serial.println("Pender Rojo en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     case (ROJO*APAGAR):
       digitalWrite(led_rojo,  LOW);
Serial.println("Apagar Rojo en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     case (VERDE):
       digitalWrite(led_verde, HIGH);
Serial.println("Prender Verde en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     case (VERDE*APAGAR):
       digitalWrite(led_verde, LOW);
Serial.println("Apagar Verde en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     case (AZUL):
       digitalWrite(led_azul, HIGH);
Serial.println("Prender Azul en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     case (AZUL*APAGAR):
       digitalWrite(led_azul, LOW);
Serial.println("Apagar Verde en Núcleo -> " +
String(xPortGetCoreID()));
       break;
     default:
       msj = 0;
    };
  }
}
```

7. Resultados obtenidos

7.1 Capturas del bot

Se indica a continuación en las figuras de la 11 a la 18, los resultados obtenidos desde el bot para el control del

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

hardware. Este bot emplea comandos anteponiendo el símbolo de "/", como ya se hizo mención anteriormente.

FIGURA 11 – Estado inicial al invocar al bot desde Telegram

FIGURA 12– Estado del bot al apretar /start

FIGURA 13 – Comandos disponibles al apretar /

FIGURA 14– Respuesta del bot al encender y apagar los leds

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

FIGURA 15 - Indicación de la temperatura interna del ESP32 en °C y en ° F

FIGURA 16 – Estado del bot al apretar /**help**

FIGURA17 - Encendiendo todos los leds y verificando el status de los 3 leds

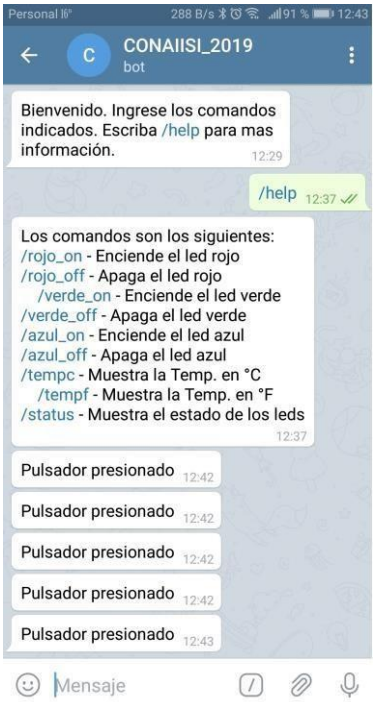

FIGURA 18 - Envío de alarma de Pulsador

7.2 Capturas Monitor serie

7.2.1 Configuración setup() y setup2()

14:02:25.584 -> setup En núcleo -> 1
14:02:25.584 -> SetupWiFi En núcleo -> 1
14:02:25.584 -> Intentando conectar a la red IPLAN-306232
14:02:26.193 -> ..
14:02:26.709 -> OK
14:02:26.709 -> IP address: 192.168.1.11
14:02:26.709 -> Strength: -44 dbm
14:02:26.709 ->
14:02:26.709 -> loop En núcleo -> 1
14:02:26.709 -> setup2 En núcleo -> 0
14:02:26.709 -> setupLEDs En núcleo -> 0
14:02:26.709 -> setupPULL En núcleo -> 0

7.2.2 Capturas de comandos del bot con /

a) /rojo_on

14:02:31.475 -> onReceive En núcleo -> 1
14:02:33.678 -> SendMessage(23) En núcleo -> 1
14:02:33.678 -> loop En núcleo -> 1
14:02:33.725 -> ReceiveMessage(23) En núcleo -> 0
14:02:33.725 -> Prender Rojo en núcleo -> 0

14:02:34.287 ->

{"ok":true,"result":{"message_id":272,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566955,"text":"Encendiendo led rojo"}}ReceiveMessage(0) En núcleo -> 0

b) /verde_on

14:02:36.631 -> onReceive En núcleo -> 1
14:02:38.694 -> SendMessage(22) En núcleo -> 1
14:02:38.694 -> loop En núcleo -> 1
14:02:38.787 -> ReceiveMessage(22) En núcleo -> 0
14:02:38.787 -> Prender Verde en núcleo -> 0
14:02:39.303 ->

{"ok":true,"result":{"message_id":275,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566960,"text":"Encendiendo led verde"}}ReceiveMessage(0) En núcleo -> 0

c) /azul_on

14:02:41.881 -> onReceive En núcleo -> 1
14:02:43.944 -> SendMessage(21) En núcleo -> 1
14:02:43.944 -> loop En núcleo -> 1
14:02:43.944 -> ReceiveMessage(21) En núcleo -> 0
14:02:43.944 -> Prender Azul en núcleo -> 0

14:02:44.553 ->

{"ok":true,"result":{"message_id":276,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566965,"text":"***Encendiendo led azul***"}}ReceiveMessage(0) En núcleo -> 0

d) /temp_C

14:02:46.897 -> onReceive En núcleo -> 1

14:02:49.475 -> loop En núcleo -> 0

14:02:49.522 ->

{"ok":true,"result":{"message_id":278,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","u

sername":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566970,"text":"***53.33 u00b0C***"}}ReceiveMessage(0) En núcleo -> 0

e) /temp_F

14:02:53.694 -> onReceive En núcleo -> 1
14:02:56.366 -> loop En núcleo -> 0

14:02:56.413 ->

{"ok":true,"result":{"message_id":280,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566977,"text":"***128 u00b0F***"}}ReceiveMessage(0) En núcleo -> 0

f) /status

14:03:01.185 -> onReceive En núcleo -> 1

14:03:03.857 -> loop En núcleo -> 0

14:03:03.904 ->

{"ok":true,"result":{"message_id":282,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568566984,"text":"**Led rojo encendido\nLed verde encendido\nLed azul encendido**"}}ReceiveMessage(0) En núcleo -> 0

g) /help

14:26:51.812 -> onReceive En núcleo -> 1
14:26:54.466 -> loop En núcleo -> 0

14:26:54.513 ->

{"ok":true,"result":{"message_id":284,"from":{"id":859432320,"is_bot":true,"first_name":"CONAIISI_2019","username":"c2ing59_bot"},"chat":{"id":464201191,"first_name":"Carlos","username":"Pqj23","type":"private"},"date":1568568415,"text":"Los comandos son los siguientes:\n/rojo_on - Enciende el led rojo\n/rojo_off - Apaga el led rojo\n /verde_on - Enciende el led verde\n/verde_off - Apaga el led verde\n/azul_on - Enciende el led azul\n/azul_off - Apaga el led azul\n/tempc - Muestra la Temp. en u00b0C\n /tempf - Muestra la Temp. en u00b0F\n/status - Muestra el estado de los leds","entities":[{"offset":33,"length":8,"type":"bot_command"},{"offset":65,"length":9,"type":"bot_command"},{"offset":99,"length":9,"type":"bot_command"},{"offset":133,"length":10,"type":"bot_command"},{"offset":165,"length":8,"type":"bot_command"},{"offset":197,"length":9,"type":"bot_command"},{"offset":227,"length":6,"type":"bot_command"},{"offset":263,"length":6,"type":"bot_command"},{"offset":295,"length":7,"type":"bot_command"}]}}ReceiveMessage(0) En núcleo -> 0

8. Conclusiones y trabajo futuro

1. Observando las marcas de tiempo obtenidas en las capturas del monitor serie podemos establecer los siguientes valores:

El valor promedio desde que se ingresa al evento onReceive() y hasta que se envía la tarea desde el núcleo 1 con SendMessage() de aprender o apagar un led es:

(2.203 + 2.063 + 2.063)/3 = 2.11 S

El valor promedio que se tarda en procesar la cola, es decir desde que se envía una tarea con SendMessage()

desde el núcleo 1 y hasta que esa misma tarea se recibe con ReceiveMessage() en el núcleo 0 es:

(47 + 93 + 0)/3 = 46.67 ms

Finalmente el valor promedio ponderado desde que se envía la leyenda de led prendido o apagado en el monitor serie del ESP32 y hasta que llega ese mensaje al bot a través de la API de Telegram es:

(564 + 516 + 609)/3 = 563 ms

2. 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.

3. 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.

4. Gracias al sistema de mensajería instantánea de Telegram se da una excelente solución al tema de seguridad ya que se brinda un mecanismo de cifrado de canal extremo a extremo y se resuelve de manera simultánea la autenticación con intercambio de credenciales entre el bot y los distintos usuarios que componen Telegram, como así también entre los distintos usuarios y Telegram.

5. En un futuro trabajo se prevé la utilización de bots pero para que se comuniquen directamente entre dispositivos sin intervención humana (Sistemas machine to machine M2M).

6. También se prevé en un futuro trabajo el empleo de una Raspberry Pi para poder manejar más núcleos y así poder ejecutar mayor cantidad de tareas, aumentando de manera notable la performance del sistema.

9. Referencias y Bibliografía

[1] https://www.freertos.org/a00116.html

[2] https://www.freertos.org/a00117.html

[3] https://www.freertos.org/a00118.html

[4] https://telegram.org/

[5] Designing bots. 1st Edition. Amir Shevat. Publicación: mayo del 2017. Editorial OREILLY. ISBN-13: 978-1491974827

[6] https://core.telegram.org/bots#6-botfather

[7] https://core.telegram.org/bots/api

[8] Internet of Things Projects with ESP32. Agus Kurniawan. Publicación: 30 de marzo de 2019. Editorial Packt. ISBN-13: 978-1789956870

[9] https://core.telegram.org/mtproto

[10] 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.

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

Recibido: A completar por el Editor. Formato: AAAA-MM-DD
Aprobado: A completar por el Editor. Formato: AAAA-MM-DD
Hipervínculo Permanente: A completar por el Editor
Datos de edición: Vol. [A completar por el Editor]-Nro. [A completar por el Editor]-Art. [A completar por el Editor]
Fecha de edición: Formato: AAAA-MM-DD

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

ReDDI

San Justo, 22 de octubre de 2020

Certificamos que el artículo

"ACCESO REMOTO A DISPOSITIVOS IOT MEDIANTE TÉCNICAS DE MENSAJERÍA EMPLEANDO BOTS Y FREERTOS",

de Carlos Alberto Binker, Hugo Tantignone, Eliseo Alfredo Zurdo, Guillermo Buranits, ha sido publicado en el Volumen: 5 - Número 1 (Agosto-2020) de la revista digital ReDDI ISSN: 2525-1333.

Dra. Bettina Donadello
Secretaria de Investigaciones Tecnológicas

Mg. Jorge Eterovic
Decano

Universidad Nacional de La Matanza

DIIT Departamento de Ingeniería e Investigaciones Tecnológicas

ISSN: 2525-1333

Revista Digital del Departamento de
Ingeniería e Investigaciones
Tecnológicas de la Universidad
Nacional de La Matanza

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

C2.1

Control de iluminación WIFI

Escenario 1

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza
ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

En este escenario, como se expresó en el resumen, cada dispositivo ESP-01s aloja un servidor web propio que controla una luminaria por medio de una dirección IP fija. El módulo WIFI y la llave se conectan formando el característico circuito de llave de combinación, tal como puede observarse en la Figura 1. Tanto NA como NC son los bornes normalmente abierto y normalmente cerrado del relé respectivamente y el terminal denominado *común*, permite conectar la llave a cualquiera de estos dos bornes. En el caso del relé, el mismo se activa con un nivel cero (LOW) en la entrada del pin I00 del ESP-01s, cuyo diagrama se muestra en la Figura 2. Por otro lado, la Figura 3 permite comprender cómo el relay se activa con un nivel lógico cero en el terminal I00 del ESP-01s.

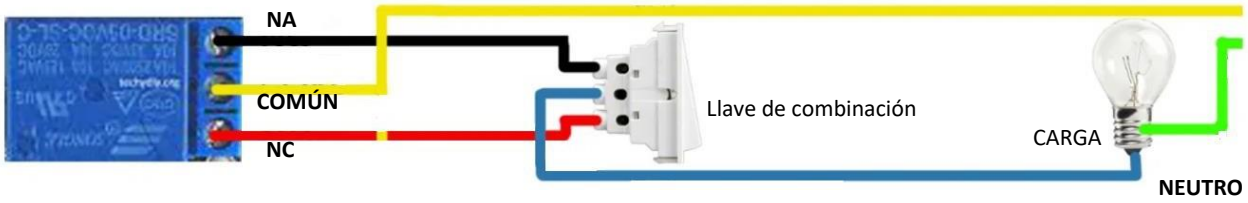

Figura 1 – Circuito de combinación entre la llave y el relay de la placa ESP-01s Relay.

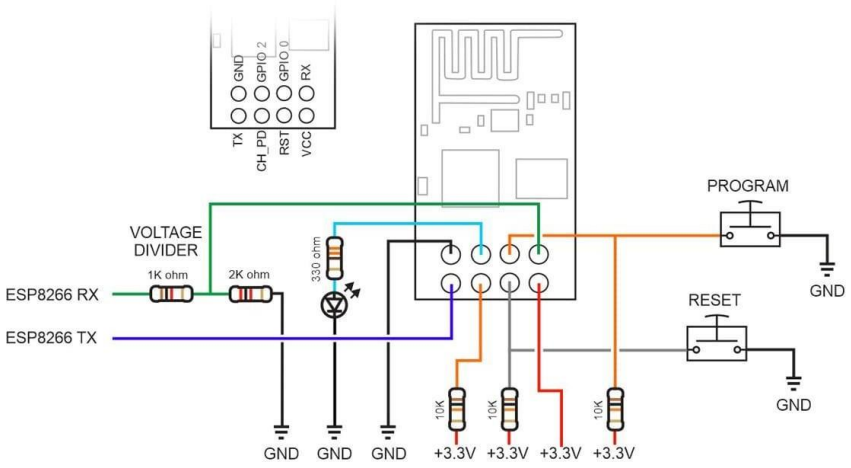

Figura 2 – Pin out del microcontrolador WIFI ESP-01s

Revista Digital del Departamento de
Ingeniería e Investigaciones
Tecnológicas de la Universidad
Nacional de La Matanza

ISSN: 2525-1333. Vol.: XX - Nro. XX (MES-AÑO)

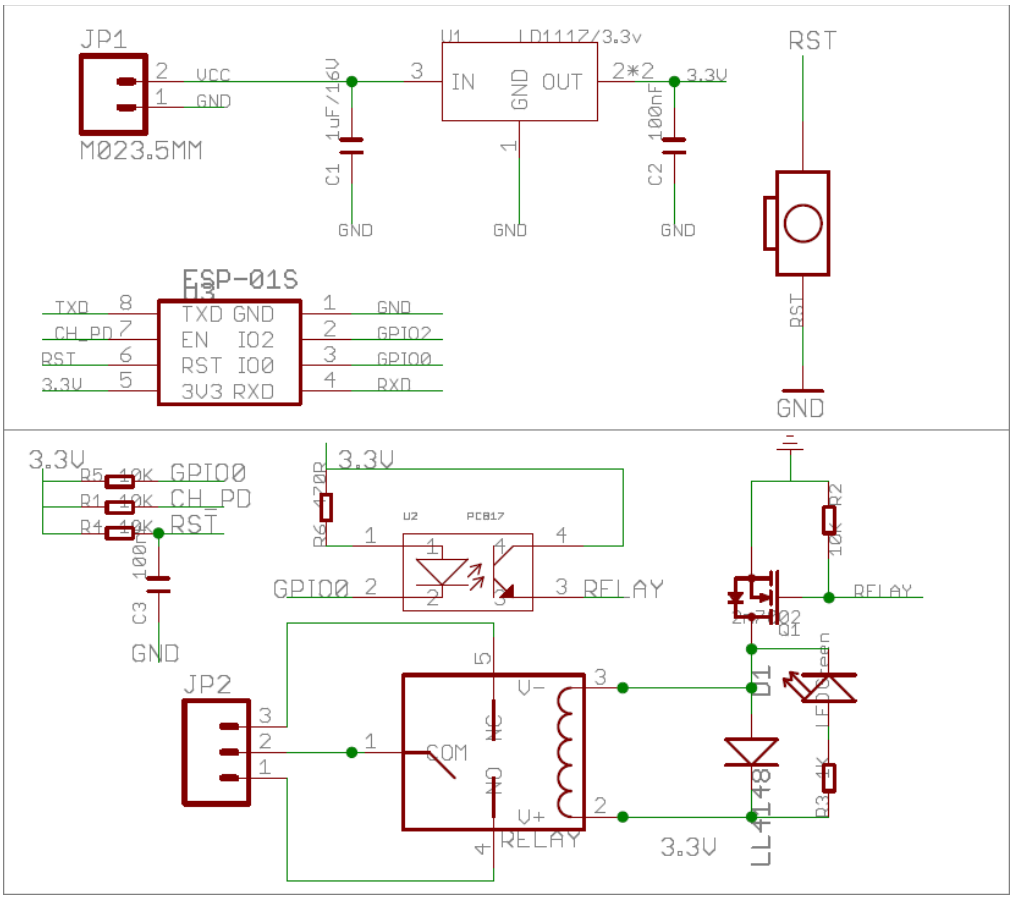

Figura 3 – Esquema eléctrico de la placa Relay ESP-01 Relay

A su vez A su vez, este escenario admite dos configuraciones posibles:

a) Llave de combinación en caja separada a donde se encuentra el módulo WIFI
b) Llave de combinación en la misma caja en donde se encuentra el módulo WIFI

La configuración **a)**, obedece a la situación real en donde ya en una casa existen llaves de combinación en cajas diferentes, mientras que la configuración **b)**, simula la condición anterior, pero tanto el módulo WIFI como la llave de combinación se montan en la misma caja eléctrica (que por lo general es de 10 cm x 5 cm), dando así la versatilidad al sistema de control ya sea manual o bien inalámbrico vía web. La caja en la que se montó el módulo WIFI ESP-01s con la Placa Relay es la siguiente (se muestran las medidas de la caja en milímetros como así también se indican los terminales de conexión), ver Figura 4 y 5:

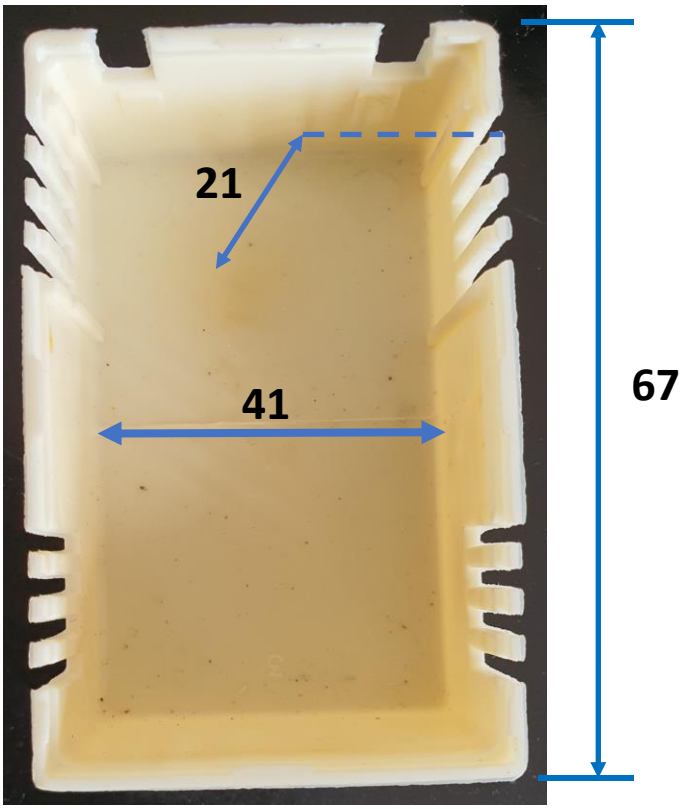

Figura 4 – Caja plástica de montaje del módulo WIFI (medidas en mm)

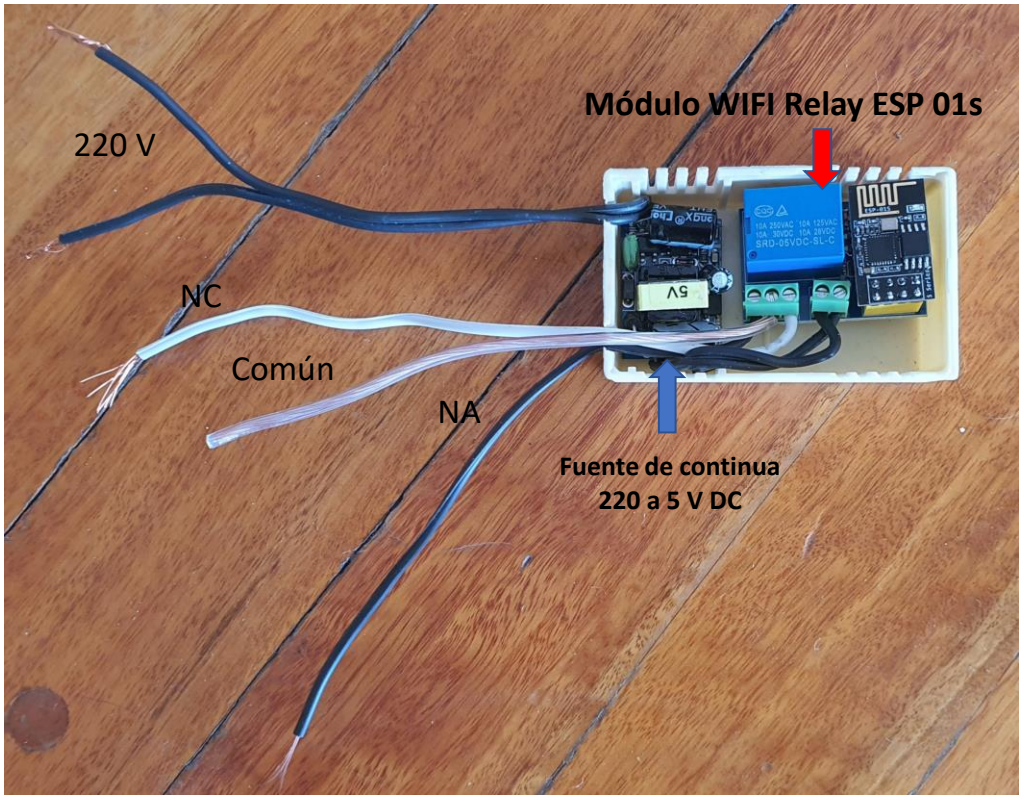

Figura 5 – Caja plástica de montaje del módulo WIFI

Como se había mencionado, cada módulo WIFI se maneja con un botón ***ON/OFF*** el cualestá inserto en una página HTML con los correspondientes estilos dados por CSS. Dicha página programada en HTML y con los estilos dados por CSS es albergada en la memoria flash del dispositivo ESP8266. Para ello es necesario utilizar el concepto de web asíncrona. A continuación se indica el código HTML utilizado, ver Figura 6.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Servidor Web Esp8266</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <h1 class="titulo">Servidor Web ESP8266</h1>

    <div class="caja">
        <p><button class="button2 button" onclick=location.href="/ON-OFF"> ON/OFF</button></p>
        <p class="titulo">Estado: %ESTADO_RELE%</p>
    </div>

</body>
</html>
```

Figura 6 – Código HTML subido a la flash del ESP-01 s que funciona como web server

El código Arduino es el siguiente (ver Figuras 7, 8 y 9):

```
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <FS.h>


const char* ssid = "Multicanal_2.4";
const char* password = "xxxxxx";

IPAddress ip(192,168,0,101);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);

const int Pin_Rele = 0;
String Estado_Pin;

AsyncWebServer server(80);

String processor(const String& var)
{
  if(var == "ESTADO_RELE")
    {
     if(digitalRead(Pin_Rele)==1)
     {
      Estado_Pin= "OK";
     }
     else
     {
      Estado_Pin = "OK";
     }
    return Estado_Pin;
    }
}
```

Figura 7 – Código Arduino I

```
void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
  pinMode(Pin_Rele, OUTPUT);

  if(!SPIFFS.begin()){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }
  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.config(ip, gateway, subnet);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println(WiFi.localIP());
  Serial.println(String(digitalRead(2))); // El GPIO02 está en estado alto
  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });
    // Route to load style.css file
  server.on("/estilos.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/estilos.css", "text/css");
  });
  // Route to set GPIO to HIGH
  server.on("/ON-OFF", HTTP_GET, [](AsyncWebServerRequest *request){
    digitalWrite(Pin_Rele, !digitalRead(Pin_Rele));
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });
      server.begin();
}
```

Figura 8 – Código Arduino II

```
void loop() {

}
```

Figura 9 – Código Arduino III

Figura 10 – Cliente WeB – Botón ON/OFF

En este código, la esencia es el uso de la función "*proccesor"*, incorporada a la librería *ESPAsyncWebServer.h*; finalmente se muestra también el cliente web que comanda a cada llave, ver Figura 10. Por lo tanto la luminaria se enciende tanto desde el botón ON/OFF del navegador web como con la tecla de combinación física indistintamente. A continuación mostramos el prototipo armado simulando la instalación eléctrica real de una casa. Ver Figura 11.

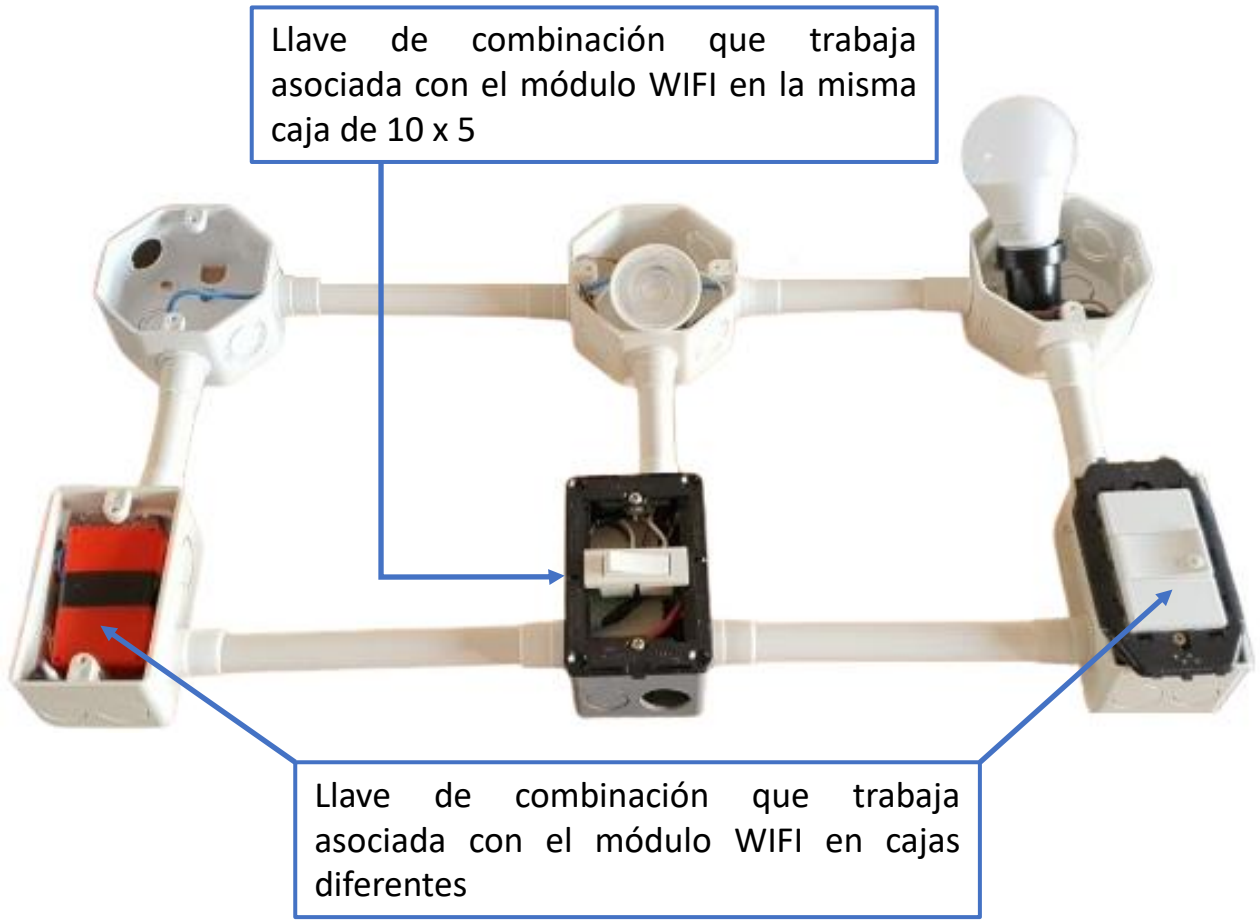

Figura 11 – Prototipo de simulación de una instalación eléctrica real de un hogar

En la caja eléctrica rectangular de la izquierda, se alberga sólo el módulo WIFI, tal como se indica en la Figura 5 (esta caja roja es la que se imprimió con una impresora 3D). Este módulo trabaja asociado con la llave de combinación que se montó en la caja eléctrica rectangular de la derecha. Esta alternativa corresponde a la opción **a)** de este escenario. Por otro lado, en la caja central se alberga tanto el módulo WIFI (Figura 5) y la llave de combinación. Se hace notar que este caso se corresponde con la alternativa **b)** del escenario descripto. Asimismo la marca de la llave empleada es de la marca *Cambre*, de la línea *Siglo 21*, ya que este modelo por su altura entra de forma satisfactoria en la caja de 10 x 5 cm, Ver Figura 12. Finalmente de manera complementaria a la Figura 1, en la Figura 13 se puede apreciar el circuito de combinación constituido por la tecla física y la otra llave, simulada por el Relay a través de los contactos común, NA (normalmente abierto) y NC (normalmente cerrado), comandado por el módulo ESP8266, en su variante de placa ESP-01s. En este caso se presenta una situación bien real, tal lo constituye la circulación a través de una escalera y con la posibilidad de comandar la luz desde la planta baja o desde el primer piso, por ejemplo.

Figura 12 – Llave de combinación marca Cambre, línea Siglo 21.

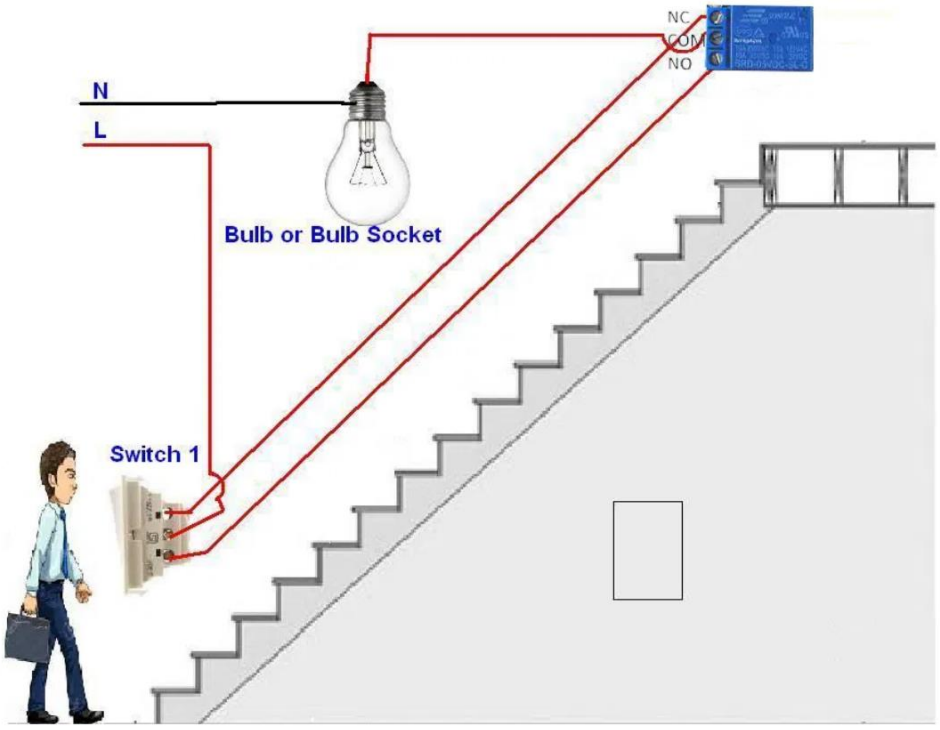

Figura 13 – Esquema circuital de combinación entre la llave de tecla y el relay comandado por el ESP-01s

C2.2

Control de iluminación WIFI

Escenario 2

En este escenario, el control web es externo y está basado en la nube. En este caso se empleó un servicio gratuito de hosting a través del sitio *www.000webhost.com*. En dicho sitio, se albergó una página web conectada a una base de datos MariaDB. Esta base de datos contiene los botones de comando para el control de la iluminación. También el hecho de tener una base de datos nos habilita, por ejemplo, a leer información de sensores y enviarlos a la base de datos para su almacenamiento y posterior utilización, pero estas acciones quedarán pendientes para el siguiente proyecto, empleando otros protocolos de comunicación más robustos, como por ejemplo MQTT. En este caso se empleó un Arduino Mega y un solo ESP-01s para vincular la placa a Internet. A diferencia del escenario anterior 2.1, debe instalarse el Arduino Mega en algún sector de la casa y desde allí realizar todas las conexiones hacia las cargas a controlar; es decir desde cada pin GPIO del Arduino Mega, deberá salir un cable de control que es el que activará el relé para el encendido de cada luminaria, lo cual dificulta la instalación eléctrica del hogar, y esto no es siempre factible, ya que por lo general las casas tienen caños de 5/8 de pulgada que dificultan la introducción de la cinta pasacables. En este caso los módulos a emplear se pueden ver en la Figura 14

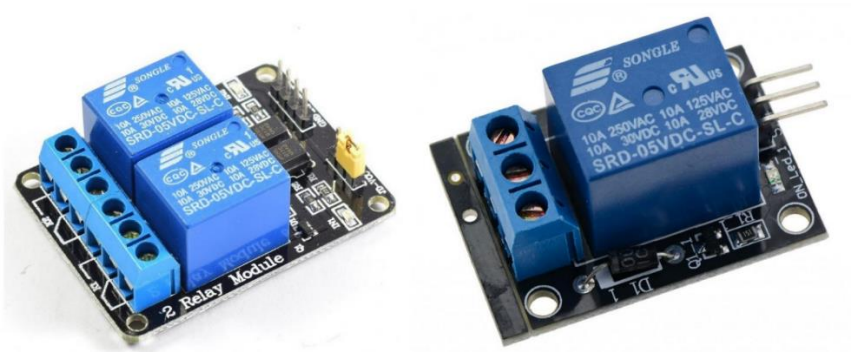

Figura 14 – Módulos Relay a emplear para control de luminaria con Arduino Mega 2560

El panel de control web, cuyos botones se hayan definidos en una tabla de la base de datos MARIADB que se encuentra hosteada en 000webhostapp, tiene el siguiente aspecto, ver Figura 15, (https://rum-analog.000webhostapp.com/index.php):

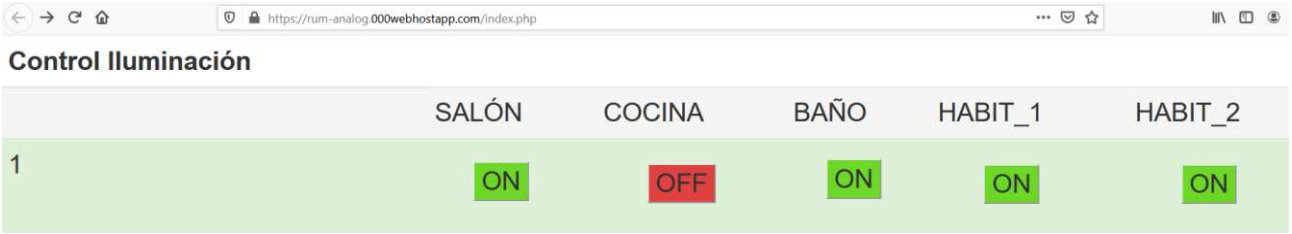

Figura 15 – Dashboard para control de iluminación

El esquema circuital es el siguiente, ver Figura 16:

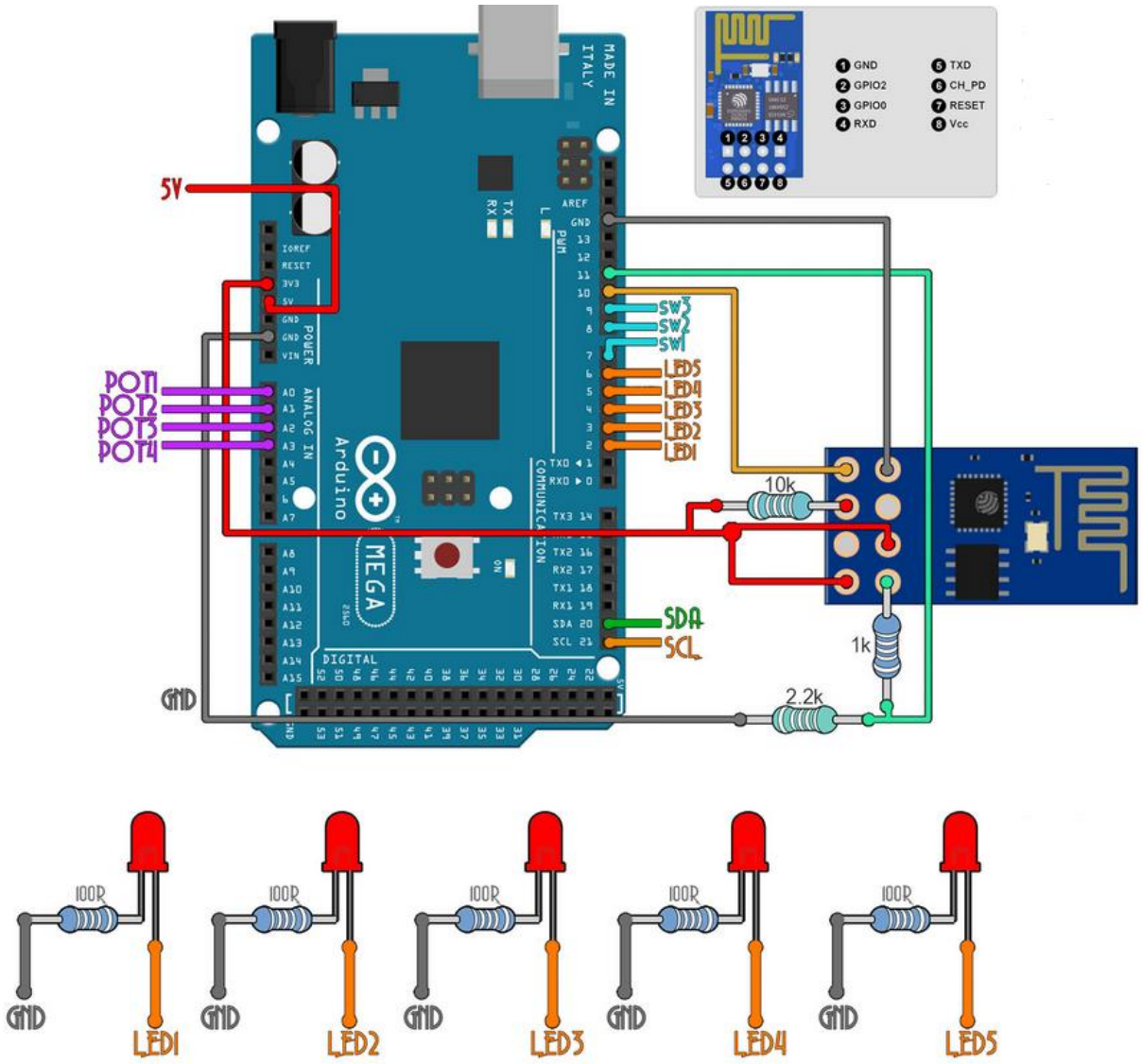

Figura 16 – Esquema circuital

Nótese que desde la base de datos también pueden añadirse otros botones de control, como por ejemplo los potenciómetros (indicados en el esquema eléctrico como POT1 a POT4, los cuales sensarían valores analógicos que serían enviados a la base de datos. Para simplificar el estudio, sólo se han considerado los botones de encendido y apagado remoto vía web para los LEDS. Por lo tanto para resumir en este escenario, desde la página web indicada en la Figura 15, se activan los botones a ON o a OFF (incluso esto se ha hecho con un cambio de color usando el framework de bootstrap, etc.) y luego desde el ESP-01s se leen los estados booleanos de estos botones albergados en la base de datos, procediéndose al encendido o apagado de las luminarias.

C2.3

Informe de simulación IOT empleando Packet Tracer de CISCO

Interacción de elementos IOT programables y gestión remota de los mismos a través de una red de datos

Resumen. Este trabajo de investigación plantea de qué manera los dispositivos IOT [1] interaccionan entre sí a través de una red de datos. Se estudia cómo se registran los dispositivos en un servidor remoto (IOT Registration Server) [2] y de qué forma los mismos pueden ser accedidos desde Internet para su gestión, ya sea desde una computadora, tablet o bien desde un smartphone vía la red celular 3G/4G. Se simulan condiciones de entorno (Environments) para ciertas variables y en base a ellas se crean reglas que permiten un determinado comportamiento de los dispositivos IOT. Dada la coyuntura actual de pandemia a raíz del covid-19, el realce de las simulaciones de redes de datos adquiere una relevancia trascendental, y además nos da la pauta de cuál es el modelo que debemos abordar para una realización real con componentes físicos tales como ESP 32, Raspberry PI, sensores y actuadores reales, etc. A tal efecto se propone un caso de estudio empleando la plataforma Paquet Tracer de CISCO, la cual incorpora dispositivos IOT configurables mediante JavaScript [3], Python [4] y Visual.

Keywords: IOT, IOT Registration Server, JavaScript, Environments

1. Introducción

1.1 Descripción de dispositivos IOT en plataforma Paquet Tracer

La plataforma Paquet Tracer permite simular una extensa variedad de dispositivos IOT. Los mismos podríamos catalogarlos en dispositivos sensores y actuadores. Los sensores o detectores con los que cuenta la plataforma en la versión utilizada (versión 7.3) son 21 (veintiuno), mientras que los actuadores son 14 (catorce). Observar la Figura 1.

Figura 1. Sensores y actuadores con posibilidad de ser conectados a una placa Arduino

Los actuadores y los sensores indicados permiten ser conectados al MCU que tiene el sistema. El MCU simula una placa del tipo Arduino uno [5] y permite ser programada tanto en JavaScript, Python o Visual (Blockly Programming) [6]. También hay cuatro categorías más de dispositivos, los cuales permiten ser utilizados o bien como componentes (es decir que se pueden conectar al MCU) o bien como *Smart devices* (dispositivos inteligentes), los cuales estos últimos permiten ser conectados directamente a la red a través de un adaptador, ya sea éste RJ-45 o WIFI y pueden ser registrados directamente en un servidor remoto para su gestión. En cambio los elementos sensores y actuadores que se conectan a un MCU no se registrarán en el IOT registration server, sino que en este caso lo que se registrará en el servidor es el MCU asociado, el cual conecta a estos dispositivos sensores y/o actuadores. A continuación se observa una figura en donde se expone estas cuatro categorías de dispositivos más la MCU (Arduino uno). Ver Figura 2.

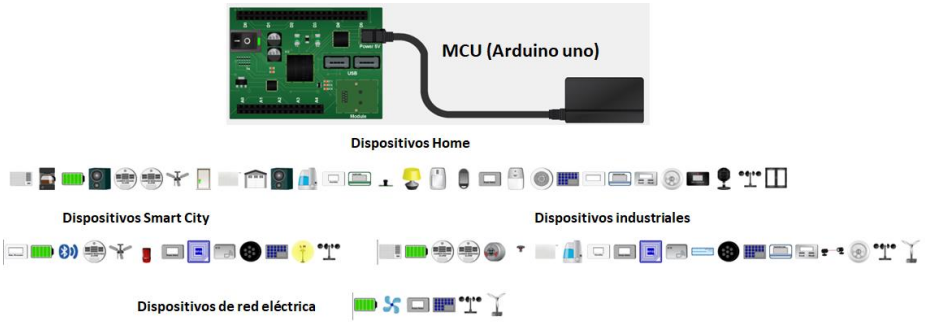

Figura 2. Dispositivos para empleo masivo como smart devices o componentes y MCU

1.2 Descripción de un dispositivo IOT en Paquet Tracer

Desde la solapa *I/O Config* se puede elegir si el detector trabajará como un sensor que irá conectado al Arduino uno o si trabajará como un smart device (dispositivo inteligente). En la solapa *Programming*, se accede al main del código JavaScript, Python o Visual. En la solapa *Config* se configura la parte de red del elemento y algo muy importante, desde aquí se lo vincula al servidor remoto (IOT Registration Server). Cuando al colocar la dirección IP más el usuario y el password, el dispositivo se vincula correctamente, el botón inferior derecho pasa de *Connect* a *Refresh.* Esta situación indica que el dispositivo se ha registrado exitosamente en el servidor remoto. A continuación se muestra también un ejemplo de un detector de monóxido de carbono (sensor) y de un display LCD (actuador), ambos configurados por software. Todos los dispositivos tienen la misma estructura. Ver Figura 3.

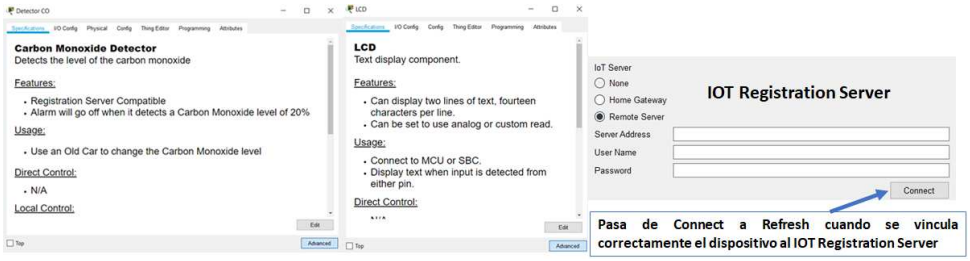

Figura 3. Detector de monóxido de carbono (CO) y LCD más IOT Registration Server

2. Diseño del caso de estudio empleando Paquet Tracer con IOT

2.1 Planteo de la topología a sintetizar mediante la plataforma Paquet Tracer

Se sintetizará empleando el Paquet Tracer de CISCO (versión 7.3) la siguiente topología de red (ver Figura 4).

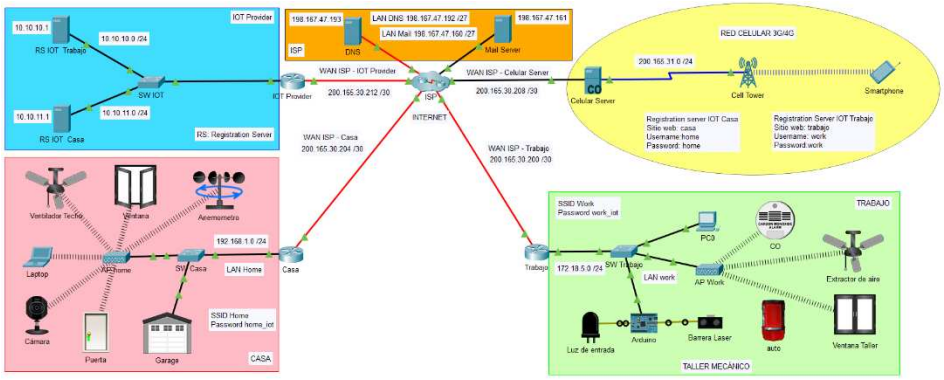

Figura 4. Topología correspondiente al caso de estudio

En esta topología se distinguen varias zonas. Tenemos tres routers de borde, denominados *Casa*, *IOT Provider* y *Trabajo* más un servidor de acceso a la red celular 3G/4G denominado *Celular Server*. Por otro lado, las zonas en color rosa (CASA), la zona celeste (IOT Provider), y la zona verde (TRABAJO) se conectan al ISP (Internet Service Provider), que es el operador de acceso a Internet. También la red celular tiene acceso a Internet, es decir se conecta con el ISP a través del denominado Celular Server y es a través de él mediante el cual los usuarios móviles a través de la Cell Tower se conectan a Internet entre otros servicios (en esta caso la idea es mostrar como un usuario se conecta por su celular a los IOT RS (IOT Registration server). Hay un IOT RS *Casa* que registra los dispositivos IOT del hogar, mientras que el IOT RS *trabajo* registra los dispositivos IOT del lugar de trabajo, en este caso se plantea como ejemplo un *taller mecánico*. A continuación se expone la tabla de direccionamiento de los routers de la topología (Tabla 1) y del swicht IOT (Tabla 2). El ruteo se ha realizado mediante OSPF versión 2 [7] de área única (área 0).

	RED	Dirección	Máscara	Interfaz de host
Casa	LAN Home	192.168.1.0	/24	g0/1 DG 192.168.1.254, resto IOT DHCP
	WAN casa-ISP	200.165.30.204	/30	g0/0/0 – 200.165.30.206 - SFP
Trabajo	LAN Work	172.18.5.0	/24	g0/1 DG 172.18.5.254, resto IOT DHCP
	WAN Trabajo-ISP	200.165.30.200	/30	g0/0/0 – 200.165.30.202 - SFP
ISP	WAN casa-ISP	200.165.30.204	/30	g0/1/0 – 200.165.30.205 - SFP
	WAN Trabajo-ISP	200.165.30.200	/30	g0/0/0 – 200.165.30.201 - SFP
	WAN ISP-IOT Prov.	200.165.30.212	/30	g0/2/0 – 200.165.30.213 - SFP
	WAN ISP-Celular	200.165.30.208	/30	g0/0 – 200.165.30.209 - Cooper
	LAN DNS Server	198.167.47.192	/27	g0/3/0 – 198.167.47.222 - SFP
	LAN Mail Server	198.167.47.160	/27	g0/1 – 198.167.47.190 - Cooper
IOT Provider	WAN ISP–IOT Prov.	200.165.30.212	/30	g0/2/0 – 200.165.30.214 – SFP
	Vlan Trabajo, VLAN 10	10.10.10.0	/24	g0/0.10 – 10.10.10.254
	Vlan casa, VLAN 11	10.10.11.0	/24	g0/0.11 – 10.10.11.254

Tabla 1. Direccionamiento IPv4 de los routers – SFP: Fibra óptica, Cooper: cobre

Switch	VLAN	Puerto de acceso / Trunk
SW IOT	10, Trabajo	f0/1, access mode
	11, Casa	f0/2, access mode
	10, 11	g0/1, trunk mode

Tabla 2. VLANS y puertos de acceso y trunk del swicht IOT del IOT Provider

A continuación se presenta la tabla con el direccionamiento correspondiente a los servidores utilizados en la topología objeto de estudio (ver Tabla 3).

Servidor	RED	Máscara	Interfaz de host	Default Gateway
IOT RS Trabajo	10.10.10.0	/24	10.10.10.1, Vlan 10	10.10.10.254
IOT RS Casa	10.10.11.0	/24	10.10.11.1, Vlan 11	10.10.11.254
DNS	198.167.47.192	/27	198.167.47.193	198.167.47.222
Mail Server	198.167.47.160	/27	198.167.47.161	198.167.47.190
Celular Server	200.165.30.208	/30	200.165.30.210, Backbone	200.165.30.209
	200.165.31.0	/24	Cell Tower - DHCP	200.165.31.254

Tabla 3. Direcciones IPv4 de los servidores de la topología

El resto de los dispositivos IOT (tanto en la casa como en el trabajo), adquieren sus direcciones IP mediante DHCP server. Los servidores DHCP se encuentran configurados en los routers perimetrales *Casa* y *Trabajo* respectivamente. Algunos de los dispositivos IOT se hallan vinculados a la red mediante WIFI y otros mediante cable RJ-45 conectados directamente a los switches de LAN denominados respectivamente como *SW casa* y *SW trabajo*. Para las conexiones WIFI se emplean dos access points denominados respectivamente como AP home y AP work, que utilizan el estándar 802.11ac, es decir permiten trabajar en las bandas de 2,4 y 5 Ghz respectivamente. El *Celular server* también brinda direcciones IPv4 públicas mediante DHCP a los smartphones u otros dispositivos inalámbricos como laptops o tablets. Recordar que en nuestro estudio emplearemos un smartphone para conectarnos vía la red celular 3G/4G a los IOT Registration Servers *Casa* y *Trabajo*. Se han asociado registros del tipo A en el DNS para acceder por nombres en la web. Las asociaciones realizadas son: **casa** que corresponde a la URL cuya IPv4 es 10.10.11.1, mientras que la URL **trabajo** se corresponde con la dirección IPv4 10.10.10.1. Los SSID y las credenciales de acceso a los IOT RS se encuentran indicadas en la plataforma de estudio. Como criterio para el default gateway (DG) en todos los casos se ha tomado la última dirección IP de host disponible. La conectividad IP es plena en toda la topología, es decir se puede hacer ping desde un dado dispositivo a cualquier otro del sistema, siendo esto condición sin ecuanon para la obtención de los resultados dados más adelante en el punto 3.3 correspondiente a resultados alcanzados.

2.2 Diseño de las condiciones a cumplir por los dispositivos IOT en el entorno Casa y en el entorno Trabajo (reglas establecidas en el IOT RS)

Entorno casa

Se proponen las siguientes condiciones o reglas: cuando el anemómetro detecta viento se abre la ventana y se apaga el ventilador de techo, por el contario cuando no hay viento detectado por el anemómetro, se cierra la ventana y se prende el ventilador de techo. Estas dos condiciones se programan mediante dos reglas en el servidor de registración IOT Casa. La tercera condición es cuando la cámara web detecta movimiento pone en la condición de *lock* a la puerta, es decir la traba. Por el contrario sin detección de movimiento, si la puerta está abierta (unlock) permanecerá en dicho estado (destrabada) y si está cerrada (lock) permanecerá en esa condición. Ver más adelante en resultados alcanzados las capturas correspondientes a dichas condiciones configuradas en el IOT registration server Casa.

Entorno Trabajo

En este caso el marco que se propone es el de un *taller mecánico* al cual ingresan vehículos de manera asidua. Aquí se utiliza una placa Arduino uno con un sensor *Trip Ware*, cuya función es que cuando ingresa un vehículo al taller mecánico se interrumpe un láser que está transmitiendo dicho sensor. Esta interrupción es detectada como un *flanco ascendente* y procesada en el programa hecho en *JavaScript* de la MCU (Arduino uno) de forma tal que se realiza un conteo de los vehículos. Además cada vez que se detecta un vehículo se enciende un LED (que simula una luminaria eléctrica de entrada de mayor potencia) por un lapso de 10 segundos. Cada 5 vehículos contados se envía un mail al *Mail Server*, el cual es registrado por un cliente de mail que alberga el smartphone o bien una PC o laptop. Cada vez que se recibe un mail por parte del Arduino el contador de vehículos es puesto a 0 (cero).

Por otro lado se dispone también en el mismo taller de un detector de monóxido de carbono (CO) que incorpora una alarma que cuando supera un determinado porcentaje de nivel de CO se activa. En esta situación el dispositivo está programado para que cuando se supere en un 20% el nivel de CO se dispare la alarma abriéndose la ventana y encendiéndose un extractor de aire a la máxima velocidad (el extractor de aire tiene dos velocidades). La fuente de CO se obtiene del escape defectuoso de un auto que se emplea como variable de entrada. Para accionar el auto y por ende la generación de CO se debe apretar la tecla ALT más la tecla izquierda del mouse sobre el auto rojo. Al apagarse el motor del auto y cuando el nivel de CO desciende por debajo de un 5% se cierra la ventana y se apaga el extractor de aire. Ver más adelante en resultados alcanzados las capturas correspondientes a esta situación.

3. Desarrollo de la experiencia

En este apartado se aportará la experiencia de vislumbrar la *interacción* del mundo del networking con los dispositivos IOT. Para ello primero haremos una síntesis preliminar de los dispositivos utilizados y finalmente presentaremos las capturas gráficas más relevantes de los resultados obtenidos en el caso de estudio planteado.

3.1 Herramientas utilizadas

Para la síntesis de la topología de red mencionada con anterioridad emplearemos los siguientes elementos de hardware y software:

1. PC Dell Inspiron 15 modelo 5584, Intel Core™ I7-8565U Processor, 32 GB RAM
2. Sistema operativo Windows 10 Education
3. Paquet Tracer versión 7.3.0.0838
4. Tres routers de borde modelo 1941con dos interfaces de fibra óptica (SFP)
5. Un router modelo 2911 (ISP), con 4 interfaces SFP
6. Dos Servidores IOT registration servers y un Celular Server para acceso a 3G/4G
7. Un servidor de mail y un servidor para DNS
8. Tres switches modelo 2960
9. Dos access points (AP) norma 802.11 ac
10. Una torre para telefonía celular (Cell Tower)
11. Un smartphone

Nótese que las interfaces de fibra óptica SFP permite simular la WAN tendida entre el ISP y los diferentes routers de borde, siendo ésta la realidad física del verdadero funcionamiento. La Figura 5 muestra los paneles posteriores de los routers 1941 y 2911 respectivamente.

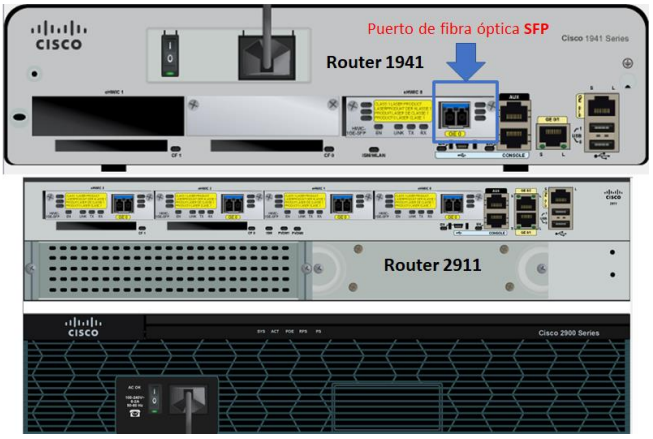

Figura 5. Routers 1941 y 2911, con puertos de fibra óptica SFP a 1GB/s

Los dispositivos IOT son los siguientes:

CASA

1 - Ventilador de techo, 2 - Ventana, 3 - Anemómetro, 4 - Cámara web, 5 - Portón de Garage y 6 - Laptop.

TRABAJO (Taller mecánico)

1 - Placa Arduino (MCU) con led y barrera láser (Trip Ware), 2 - Detector de monóxido de carbono (CO), 3 - Extractor de aire, 4 - Ventana, 5 - auto (para la simulación del CO emanado por el caño de escape), 6 - PC del tipo Desktop con cliente de mail.

3.2 Configuración de los routers de borde y del ISP

En todos los routers se ha configurado OSPF versión 2 de área única (área 0). A continuación se indican las tablas de ruteo correspondientes a los cuatro routers de la topología en estudio (ver Figura 6).

```
ISP#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 200.165.30.210 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 2 subnets
O       10.10.10.0/24 [110/2] via 200.165.30.214, 03:33:16, GigabitEthernet0/2/0
O       10.10.11.0/24 [110/2] via 200.165.30.214, 03:33:16, GigabitEthernet0/2/0
     172.18.0.0/24 is subnetted, 1 subnets
O       172.18.5.0/24 [110/2] via 200.165.30.202, 03:33:26, GigabitEthernet0/0/0
O    192.168.1.0/24 [110/2] via 200.165.30.206, 03:33:16, GigabitEthernet0/1/0
     198.167.47.0/24 is variably subnetted, 4 subnets, 2 masks
C       198.167.47.160/27 is directly connected, GigabitEthernet0/1
L       198.167.47.190/32 is directly connected, GigabitEthernet0/1
C       198.167.47.192/27 is directly connected, GigabitEthernet0/3/0
L       198.167.47.222/32 is directly connected, GigabitEthernet0/3/0
     200.165.30.0/24 is variably subnetted, 8 subnets, 2 masks
C       200.165.30.200/30 is directly connected, GigabitEthernet0/0/0
L       200.165.30.201/32 is directly connected, GigabitEthernet0/0/0
C       200.165.30.204/30 is directly connected, GigabitEthernet0/1/0
L       200.165.30.205/32 is directly connected, GigabitEthernet0/1/0
C       200.165.30.208/30 is directly connected, GigabitEthernet0/0
L       200.165.30.209/32 is directly connected, GigabitEthernet0/0
C       200.165.30.212/30 is directly connected, GigabitEthernet0/2/0
L       200.165.30.213/32 is directly connected, GigabitEthernet0/2/0
S*   0.0.0.0/0 [1/0] via 200.165.30.210
```

```
IOT_Provider#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 200.165.30.213 to network 0.0.0.0

     10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C       10.10.10.0/24 is directly connected, GigabitEthernet0/0.10
L       10.10.10.254/32 is directly connected, GigabitEthernet0/0.10
C       10.10.11.0/24 is directly connected, GigabitEthernet0/0.11
L       10.10.11.254/32 is directly connected, GigabitEthernet0/0.11
     172.18.0.0/24 is subnetted, 1 subnets
O       172.18.5.0/24 [110/3] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
O    192.168.1.0/24 [110/3] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
     198.167.47.0/27 is subnetted, 2 subnets
O       198.167.47.160/27 [110/2] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
O       198.167.47.192/27 [110/2] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
     200.165.30.0/24 is variably subnetted, 5 subnets, 2 masks
O       200.165.30.200/30 [110/2] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
O       200.165.30.204/30 [110/2] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
O       200.165.30.208/30 [110/2] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
C       200.165.30.212/30 is directly connected, GigabitEthernet0/0/0
L       200.165.30.214/32 is directly connected, GigabitEthernet0/0/0
O*E2 0.0.0.0/0 [110/1] via 200.165.30.213, 03:34:18, GigabitEthernet0/0/0
```

```
Casa#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 200.165.30.205 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 2 subnets
O       10.10.10.0/24 [110/3] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
O       10.10.11.0/24 [110/3] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
     172.18.0.0/24 is subnetted, 1 subnets
O       172.18.5.0/24 [110/3] via 200.165.30.205, 03:31:09, GigabitEthernet0/0/0
     192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C       192.168.1.0/24 is directly connected, GigabitEthernet0/1
L       192.168.1.254/32 is directly connected, GigabitEthernet0/1
     198.167.47.0/27 is subnetted, 2 subnets
O       198.167.47.160/27 [110/2] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
O       198.167.47.192/27 [110/2] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
     200.165.30.0/24 is variably subnetted, 5 subnets, 2 masks
O       200.165.30.200/30 [110/2] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
C       200.165.30.204/30 is directly connected, GigabitEthernet0/0/0
L       200.165.30.206/32 is directly connected, GigabitEthernet0/0/0
O       200.165.30.208/30 [110/2] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
O       200.165.30.212/30 [110/2] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
O*E2 0.0.0.0/0 [110/1] via 200.165.30.205, 03:31:08, GigabitEthernet0/0/0
```

```
Trabajo#show ip route
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 200.165.30.201 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 2 subnets
O       10.10.10.0/24 [110/3] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
O       10.10.11.0/24 [110/3] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
     172.18.0.0/16 is variably subnetted, 2 subnets, 2 masks
C       172.18.5.0/24 is directly connected, GigabitEthernet0/1
L       172.18.5.254/32 is directly connected, GigabitEthernet0/1
O    192.168.1.0/24 [110/3] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
     198.167.47.0/27 is subnetted, 2 subnets
O       198.167.47.160/27 [110/2] via 200.165.30.201, 03:32:28, GigabitEthernet0/0/0
O       198.167.47.192/27 [110/2] via 200.165.30.201, 03:32:28, GigabitEthernet0/0/0
     200.165.30.0/24 is variably subnetted, 5 subnets, 2 masks
C       200.165.30.200/30 is directly connected, GigabitEthernet0/0/0
L       200.165.30.202/32 is directly connected, GigabitEthernet0/0/0
O       200.165.30.204/30 [110/2] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
O       200.165.30.208/30 [110/2] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
O       200.165.30.212/30 [110/2] via 200.165.30.201, 03:32:18, GigabitEthernet0/0/0
O*E2 0.0.0.0/0 [110/1] via 200.165.30.201, 03:32:28, GigabitEthernet0/0/0
```

Figura 6. Tabla de ruteo de todos los routers (*Casa*, *Trabajo*, *ISP* e *IOT_Provider*)

3.3 Resultados alcanzados

A continuación, mostramos las capturas más relevantes de los dispositivos IOT en su interacción con la red de datos. Con esto se corrobora el funcionamiento de este caso de estudio.

3.3.1 Escenario CASA

Las reglas se aplican en el IOT Registration Server Casa. Ver Figuras 7 y 8 respectivamente.

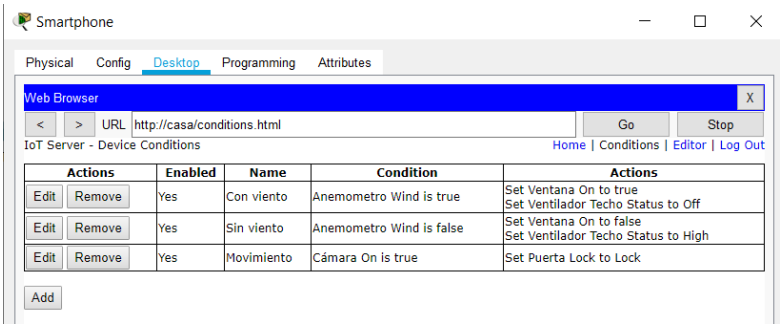

Figura 7. Reglas a aplicar (*Conditions*) al entorno del escenario Casa

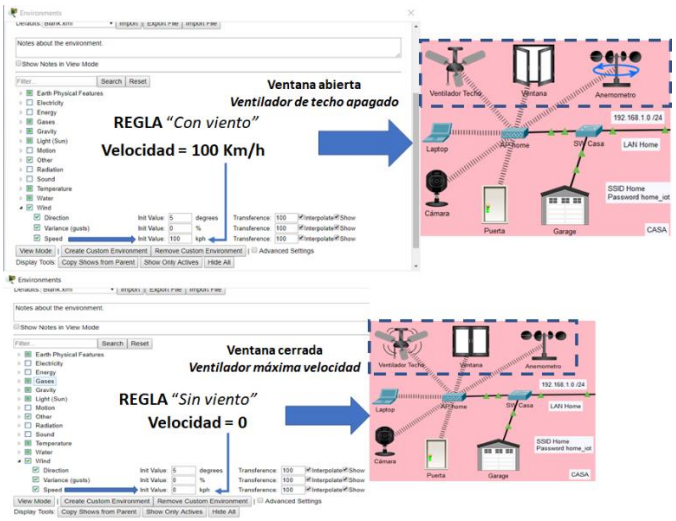

Figura 8. REGLAS con y sin viento. 100 Km/h (con viento) y 0 Km/h (sin viento)

3.3.2 Escenario TRABAJO

A continuación se muestran las capturas más relevantes de este escenario de simulación. Las reglas se aplican en el IOT Registration Server Trabajo.

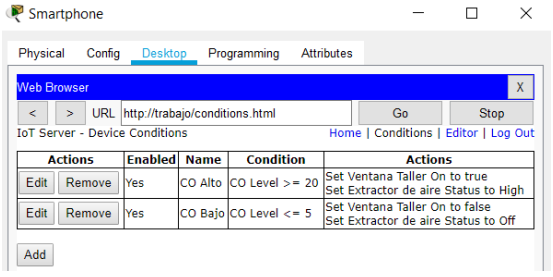

Figura 9. Reglas a aplicar (Conditions) al entorno del escenario Trabajo.

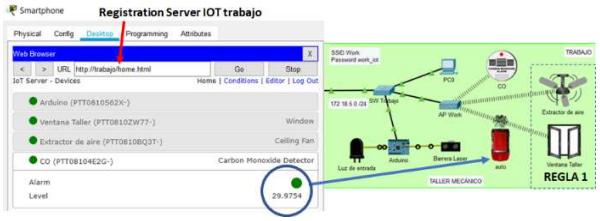

Figura 10. REGLA_1 – Apertura ventana y encendido del extractor de aire para CO >= 20%

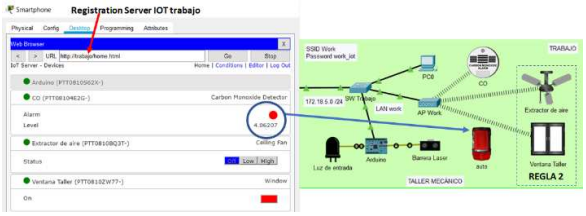

Figura 11. REGLA_2 – Cierre ventana y apagado del extractor de aire para CO <= 5 %

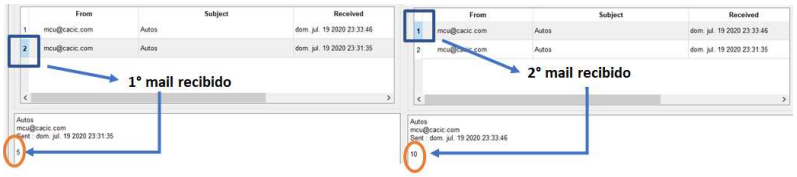

Figura 12. Recepción de mails cada 5 autos contabilizados.

4. Conclusiones y trabajo futuro

1. Con este estudio puede inferirse que todo dispositivo IOT puede ser registrado en un servidor remoto y sus datos (variables sensadas) accedidos mediante un servicio web a través de Internet o bien en forma local desde la intranet.
2. Como trabajo futuro se pretende incursionar en el protocolo MQTT como estándar en el manejo de datos mediante suscripción a canales de información. Como ejemplo de ello puede implementarse la plataforma https://ubidots.com/
3. A partir de este paper se pretende poder hacer transferencia de esta investigación hacia las cátedras de redes de computadoras, a fin de incorporar este nuevo paradigma que tiene que ver con el gran mundo del IOT, haciendo foco también en el empleo casi obligatorio del protocolo IPv6 debido a la inmensa cantidad de dispositivos.
4. A futuro conviene el uso de 6LOWPAN que es un estándar que posibilita el uso de IPv6 sobre redes basadas en el estándar IEEE 802.15.4

Referencias

1. https://www.argentina.gob.ar/jefatura/innovacion-publica/ssetic/grupo-de-trabajo/iot
2. http://static-pt-assets.s3.amazonaws.com/tutorials72.htm
3. https://www.javascript.com/
4. https://www.python.org/
5. https://www.arduino.cc/
6. https://www.packettracernetwork.com/internet-of-things/blocky-programming.html
7. https://tools.ietf.org/html/rfc2328

ANEXO II

Código	FPI-009
Objeto	Guía de elaboración de Informe final de proyecto
Usuario	Director de proyecto de investigación
Autor	Secretaría de Ciencia y Tecnología de la UNLaM
Versión	5
Vigencia	03/9/2019

FPI-013

Código	FPI-009
Objeto	Guía de elaboración de Informe final de proyecto
Usuario	Director de proyecto de investigación
Autor	Secretaría de Ciencia y Tecnología de la UNLaM
Versión	5
Vigencia	03/9/2019

Unidad Académica: Departamento de Ingeniería e Investigaciones Tecnológicas
Código: C2-ING059
Título del Proyecto: Interacción entre sistemas basados en IOT y redes de datos dual stack
Director del Proyecto: Carlos Alberto Binker
Programa de acreditación: PROINCE CyTMA2: **X**
Fecha de inicio: 01/01/2019
Fecha de finalización: 31/12/2020

1. Datos del alumno
Apellido y Nombre: Zurdo Eliseo Alfredo
DNI: 26353759
Unidad Académica: DIIT
Carrera que cursa: Ingeniería Informática
Período evaluado: 2020

2. Dictamen de evaluación de desempeño del alumno:
Colocar una cruz donde corresponda

2.1 Satisfactorio: **X**
2.1 No satisfactorio:

Fundamentos del dictamen:

Eliseo ha cumplido con todas las etapas del proyecto tal lo planificado durante 2020. Ha tenido una participación muy activa y comprometida con el grupo de investigación. Dada su enorme experiencia en programación, constituye una pieza clave para el desarrollo del equipo de investigación.

3. Propuesta de continuidad en el proyecto (si corresponde según duración estimada)
Colocar una cruz donde corresponda

3.1 Continuar en el presente proyecto:
3.2 No continuar en el presente proyecto:

Fundamentos del dictamen:

Dado el gran nivel de actuación y compromiso que ha tenido el alumno Eliseo Alfredo Zurdo es que determino su continuidad en futuros proyectos de investigación. Además quiero destacar que éste es el tercer proyecto consecutivo del cual participa y siempre lo ha hecho con total predisposición y compromiso para con el director del proyecto, sus compañeros de grupo y para las autoridades del DIIT.

San Justo, 17/2/2021		**Carlos Alberto Binker**
Lugar y fecha	Firma del Director	Aclaración de firma

Unidad Académica: Departamento de Ingeniería e Investigaciones Tecnológicas
Código: C2-ING059
Título del Proyecto: Interacción entre sistemas basados en IOT y redes de datos dual stack
Director del Proyecto: Carlos Alberto Binker
Programa de acreditación: PROINCE CyTMA2**: X**
Fecha de inicio: 01/01/2019
Fecha de finalización: 31/12/2020

1. Datos del alumno
Apellido y Nombre: Frattini, Maximiliano Gabriel
DNI: 26.849.323
Unidad Académica: DIIT
Carrera que cursa: Ingeniería en Informática
Período evaluado: 2020

2. Dictamen de evaluación de desempeño del alumno:
Colocar una cruz donde corresponda

2.1 Satisfactorio: **x**
2.1 No satisfactorio:

Fundamentos del dictamen:

Maximiliano ha cumplido con todas las etapas del proyecto tal lo planificado durante 2020. Ha tenido una participación muy activa y comprometida con el grupo de investigación. Dada su enorme experiencia en programación, constituye una pieza clave para el desarrollo del equipo de investigación.

3. Propuesta de continuidad en el proyecto (si corresponde según duración estimada)
Colocar una cruz donde corresponda

3.1 Continuar en el presente proyecto:
3.2 No continuar en el presente proyecto:

Fundamentos del dictamen:

Dado el gran nivel de actuación y compromiso que ha tenido el alumno Maximiliano Gabriel Frattini es que determino su continuidad en futuros proyectos de investigación. Cabe destacar que por ser ésta su primera participación en proyectos de investigación, el nivel alcanzado resulta favorable.

San Justo, 17/2/2021		**Carlos Alberto Binker**
Lugar y fecha	Firma del Director	Aclaración de firma