Proyecto TI basado en IoT para implementación de estrategia en tecnología inalámbrica

Guillermo Javier Romero Murcia

Universidad Distrital Francisco José de Caldas

Facultad de Ingeniería

Especialización en Proyectos Informáticos

Proyecto de grado

Bogotá D.C.

2023

Proyecto TI basado en IoT para implementación de estrategia en tecnología inalámbrica

Guillermo Javier Romero Murcia

Director: Beitmantt Cárdenas Q

Proyecto de grado para optar al título Especialista en Proyectos Informáticos

Universidad Distrital Francisco José de Caldas

Facultad de Ingeniería

Especialización en Proyectos Informáticos

Proyecto de grado

Bogotá D.C.

2023

Contenido

Resumen

Este proyecto se enfocó en diseñar un sistema automatizado que permita la comunicación entre dispositivos a través de una red, potenciando así la eficiencia y conectividad de estos bajo el concepto de *Internet of Things* (IoT).

Este sistema se implementó de forma experimental usando protocolos y tecnologías *IoT* con el objetivo de dar solución a un caso de estudio localizado en la puerta principal de una vivienda, en el cual analiza este caso a través de una metodología basada en escenarios, en donde el escenario que se implementa tiene específicamente actores llamadas cosas o dispositivos, un dispositivo es una puerta y los otros son televisores conectados en red. La puerta tiene la función de capturar video (*streaming*, descarga continua) y publicarlo en la red *IoT* de la vivienda cuando alguien llega de visita y timbra, tomando la decisión de enviar de forma local la imagen al televisor más cercano que ha detectado al usuario, cuando este se encuentra dentro de la vivienda y vía internet al teléfono del usuario, cuando el sistema detecta que no se encuentra dentro de la casa.

Para ello se propuso un código fuente basado en el de lenguaje de programación *Python,* el cual se instaló en unas *Raspberry Pi3 (RPi3)* con un sistema operativo *Raspbian Stretch with Desktop* las cuales ayudarán para la toma del video, mostrar el video en los televisores o pantallas y detectar a la persona que está dentro de la vivienda.

También se usará la aplicación *Telegram* para la notificación en el teléfono.

La propuesta muestra, un sistema práctico donde se puede implementar las herramientas disponibles para la elaboración de un proyecto informático, el cual busca dar una solución a un problema en un entorno residencial.

Palabras clave*: IoT* (*Internet of things*), *Python*, *Raspberry Pi3 (RPi3), Telegram*

Introducción

La implementación del *Internet of things* (IoT) se está convirtiendo en una necesidad en la vida cotidiana. En este contexto, se dio a conocer una propuesta que ofrece una solución innovadora para la implementación de *IoT* en una vivienda, la propuesta se centró en la creación de una red la cual contiene unos dispositivos conectados, llamados *SmartThings*, que se comunican entre sí de forma automatizada.

Para el desarrollo de esta propuesta se tomaron tres *SmartThings* la cuales están conectadas a través de una red de estrella, el cual contiene un punto de acceso convencional a Internet (*Router* o *Access Point*).

Para lograr apropiar el concepto que se planteó desde el inicio y durante la investigación, se toma como base un problema cotidiano, en el que se espera que las *SmartThings* puedan resolver el problema por sí mismas y de forma autónoma, haciendo más eficiente la conexión a Internet cuando se tienen un verdadero y justificado motivo para usarlo.

Para ello es necesario introducir el concepto de Inteligencia artificial a través de la comunicación máquina a máquina (*M2M, Machine to Machine*) o (*Machine to peer*) haciendo uso de las distintas tecnologías y protocolos dispuestos actualmente por el creciente y cambiante mundo del *IoT*.

Algunas de estas tecnologías no fueron tenidas en cuenta por la dimensión que se le dio a esta propuesta, pero sí se tuvieron en cuenta las más conocidas como *Bluetooth* y *WiFi.*

Título e identificación del tema de investigación.

Título: Proyecto TI basado en *IoT* para implementación de estrategia en tecnología inalámbrica.

Tema de Investigación: Red *IoT*.

Estudio del problema de investigación.

Cuando una persona llega a la vivienda y timbra, se generan varios casos posibles, dentro de los cuales podemos encontrar:

- La persona que llega no sabe si hay alguien o si debe volver en otro momento.
- Quienes están dentro de la vivienda, no saben quién está en la puerta tocando el timbre. (vecino, familiar, encomiendas, vendedores, domicilios, vigilante, o incluso personas con malas intenciones).
- Pérdida de información como: un mensaje urgente de un vecino, llegada de correo (encomiendas, citaciones, correos certificados, pasaportes, Tarjetas de crédito, etc**.**).
- Puede perder la visita de un pariente, amigo, un técnico contratado para la instalación o mantenimiento de un servicio, la señora del servicio.
- Una persona que quiera ingresar al predio sin estar autorizado (“Ladrón”).

De los casos anteriores es necesario identificar a la persona y tipo de acción que va a realizar, para ello es necesario generar un control de acceso remoto para abrir y cerrar la puerta con un sistema inteligente donde pueda identificar rutinas diarias de una persona, como por ejemplo:

- Cuando llega un familiar o amigo a quedarse en la casa.
- Cuando llega la ruta con los hijos.
- Cuando llega la señora del servicio.
- Cuando llega correo, encomiendas, compras como electrodomésticos.
- También es el caso de citaciones por correo certificado o incluso si se espera un pasaporte o algún documento importante.
- La instalación o mantenimiento de algún servicio contratado.
- Los sistemas de emergencia locales (Policía, bomberos, ambulancias, Asistencia médica domiciliaria.
- Abrirle la puerta a la mascota.

Planteamiento del problema de investigación.

La existencia de un sistema de seguridad es esencial para garantizar la protección y tranquilidad de las personas en cualquier tipo de vivienda. Sin embargo, en muchos casos, los usuarios no están al tanto de quiénes visitan su vivienda, especialmente cuando no están en casa o no pueden atender a los visitantes.

La falta de información sobre las visitas que se realizan en la vivienda puede generar problemas de seguridad, ya que se desconoce quién está ingresando y saliendo del lugar. Además, si una persona desconocida ingresa al predio sin estar autorizada, puede causar preocupación y malestar entre los residentes.

Formulación del problema.

Teniendo en cuenta lo planteado en el antecedente, se puede establecer la pregunta de investigación: ¿Cómo informar o alertar a una persona cuando alguien está tocando el timbre de su vivienda de manera efectiva y segura?

Sistematización del problema.

- ¿La vivienda que no cuentan con personal de seguridad, como hacen las personas para saber quién está tocando el timbre?
- ¿Los residentes de la vivienda cuentan con la suficiente tecnología para validar quién es la persona que está tocando el timbre?
- ¿Qué consecuencias negativas representa si una persona no autorizada ingresa a un conjunto residencial?
- ¿Por qué las tecnologías actuales no son suficientes para determinar el ingreso de una persona a un conjunto residencial?

Objetivos de la investigación.

A continuación, se presentan los objetivos que enmarcan el desarrollo de este proyecto:

Objetivo general.

Diseñar un proyecto TI basado en *IoT* para la implementación de un sistema con tecnología inalámbrica.

Objetivos específicos

1. Identificar las tecnologías más relevantes para la implementación de una red inalámbrica *IoT*.
2. Proponer un marco de trabajo para la elaboración del proyecto.
3. Diseñar una red *IoT* basada en un escenario en un ambiente residencial.

Justificación de la investigación.

En términos prácticos, esta investigación ofrece la oportunidad de utilizar la tecnología disponible para implementar un nuevo dispositivo o "cosa", ya sea mediante la adaptación de uno existente o adquiriendo uno nuevo en el mercado.

Es importante destacar que actualmente existen algunas alternativas similares a la propuesta presentada en este trabajo. Una de ellas es el uso de timbres digitales con cámaras conectadas a Internet, como el Arlo Pro4 (2021), que se instala sobre la puerta o pared mediante una base y tornillos. Sin embargo, es importante tener en cuenta que este tipo de sistemas puede ser físicamente vulnerable al estar instalado externamente en el hogar. En contraste, la solución propuesta en este proyecto se implementa dentro del hogar, lo que evita vulnerabilidades físicas.

Otra solución que aborda el problema de manera similar a la propuesta presentada en este proyecto es el sistema de la cámara con acceso continuo a la descarga de video desde la puerta, como el Nest Labs, Inc. (2019). Nest Cam Indoor. Este sistema no cuenta con un modo seguro de autenticación y el video se publica continuamente, lo que mantiene el puerto abierto y lo hace susceptible a ataques externos. Esto significa que cualquier persona podría tener acceso a la visualización de lo que la cámara transmite. En comparación con la solución propuesta en este proyecto.

Desde otro punto de vista, si un usuario desea implementar el proyecto por sí mismo, puede utilizar este documento como guía. Al final, obtendrá un sistema adaptable de bajo costo que le permitirá convertir su puerta existente en una puerta con beneficios tecnológicos y de seguridad.

Hipótesis de la investigación.

Basados en la identificación del problema, se plantea la siguiente hipótesis:

La implementación de una red *IoT* en una vivienda aumenta la sensación de seguridad del residente al poder monitorear de manera efectiva la persona que está tocando el timbre, reduciendo la incidencia de robos y la disminución de la incertidumbre.

Marco Referencial.

Se encuentra compuesto por el marco teórico y conceptual con el fin de soportar la investigación:

Marco teórico

Cuando Kevin Ashton en 1999, presentó su idea revolucionaria llamó la atención de los directivos de la empresa: "Internet de las cosas". Con esta expresión, Ashton quería demostrar cómo la conexión ubicua de los objetos en cualquier momento y lugar estaba evolucionando. Desde entonces, han pasado casi dos décadas y el término se ha utilizado para describir una amplia gama de avances tecnológicos que han surgido a medida que se ha implementado esta idea.

El surgimiento de *IoT* ha marcado el inicio de una nueva era en la que la comunicación entre objetos a través de Internet permite recopilar información y utilizarla para tomar decisiones y acciones que ayuden a las personas en su vida diaria. Es por eso que *IoT* se ha convertido en el concepto principal que describe este fenómeno tecnológico en el que los objetos cotidianos se conectan y comunican a través de una red para recopilar y compartir datos que mejoran la vida de las personas. (ITU,2005).

Después de 18 años, cuando la ITU hizo referencia a una nueva dimensión de Internet, en la que cualquier cosa podía conectarse en cualquier momento y lugar, lo que dio lugar al

surgimiento del concepto de Internet de las cosas (*IoT*). Si bien la tecnología ya estaba disponible en ese momento, su adopción no fue generalizada.

Según el grupo de soluciones empresariales basadas en Internet de CISCO (IBSG, Internet Business Solutions Group), *IoT* se define como el momento en que se conectan más dispositivos a Internet que personas. (Evans, 2011).

Según datos de la consultora McKinsey, en la actualidad se conectan aproximadamente 127 nuevos dispositivos a Internet cada segundo en todo el mundo. Se prevé que el Internet de las cosas (*IoT*) tendrá un impacto económico anual de entre 3.9 y 11.1 billones de dólares en 2025, gracias a la implementación de diversos modelos de negocio. (MGI, 2018)

A pesar de que se han llevado a cabo numerosas iniciativas relacionadas con diversas tecnologías, la adopción inicial del Internet de las cosas (IoT) fue lenta. Sin embargo, en la actualidad, el IoT está experimentando un gran crecimiento y se espera que para los próximos años el 95% de todos los nuevos productos incluyan tecnología de IoT. (Paneta,2017).

Marco conceptual.

- **Beacon**

  Como señala Birks, "los Beacons son pequeños dispositivos de bajo consumo de energía que utilizan tecnología Bluetooth Low Energy (BLE) para enviar señales a dispositivos móviles cercanos" (2017, p. 123), Según Chang y Wang, "los Beacons se utilizan para transmitir información a dispositivos móviles cercanos, como notificaciones push, anuncios personalizados y promociones" (2018, p. 567).

- **Bluetooth LE**

  Según Mishra y Kumar, "Bluetooth LE es una tecnología de bajo consumo de energía que permite la transmisión de datos de forma inalámbrica a corta distancia" (2019, p. 456), Como señala Khan, "Bluetooth LE se utiliza en dispositivos de baja

potencia, como sensores y wearables, para transmitir datos de forma inalámbrica a otros dispositivos" (2020, p. 234), y como señala Alqadah, "Bluetooth LE ofrece una mayor velocidad de transmisión de datos y una mayor seguridad en comparación con el Bluetooth convencional, lo que lo hace ideal para aplicaciones que requieren una alta velocidad de transmisión y una mayor seguridad" (2021, p. 567).

- **Internet de las cosas *(Internet of Thing* (IoT))**

  Según Atzori, Iera y Morabito, "El Internet de las cosas (IoT) es una red de dispositivos interconectados que utilizan sensores y tecnologías de comunicación inalámbrica para recopilar y transmitir datos a través de Internet" (2010, p. 244), de igual forma Como señala Gubbi, Buyya, Marusic y Palaniswami, "El IoT se utiliza en aplicaciones como el hogar inteligente, la salud electrónica, la fabricación inteligente y la agricultura inteligente" (2013, p. 315), sin envergo como indica Al-Fuqaha, Guizani, Mohammadi, Aledhari y Ayyash, "El IoT plantea desafíos en términos de privacidad y seguridad de los datos, ya que los dispositivos IoT recopilan y transmiten grandes cantidades de datos personales" (2015, p. 4).

- **Python**

  "Python es un lenguaje de programación interpretado, lo que significa que los programas escritos en Python no necesitan ser compilados antes de ser ejecutados" (Rossum, 1995), "Python es un lenguaje de programación multiparadigma que admite programación orientada a objetos, programación imperativa y programación funcional" (Van Rossum, 2000) y "La filosofía de diseño de Python se centra en la legibilidad del código, la simplicidad y la eficiencia" (Van Rossum, 2003).

- **SBC (Single Board Computer)**

"Los SBC son dispositivos completos que incluyen procesador, memoria, almacenamiento y conectividad en una sola placa. Son ideales para aplicaciones que requieren una solución compacta y de bajo costo" (Guthrie, 2017).

"Los SBC son populares entre los desarrolladores y fabricantes de dispositivos IoT debido a su capacidad para integrarse fácilmente en sistemas más grandes y a su bajo costo" (Hughes, 2018).

"Los SBC son una opción atractiva para los proyectos de hardware de código abierto debido a su disponibilidad y su capacidad para ejecutar software libre y de código abierto" (Upton, 2015).

- **Telegram**

"Telegram es una aplicación de mensajería que ofrece seguridad y privacidad, junto con una gran cantidad de funciones y características que la hacen una opción popular entre los usuarios" (Krishna, 2020). "Una de las características clave de Telegram es su enfoque en la privacidad y la seguridad, con cifrado de extremo a extremo y opciones de autodestrucción de mensajes" (Lomas, 2018). "Telegram es una aplicación multiplataforma, disponible en una variedad de dispositivos y sistemas operativos, lo que la hace fácilmente accesible para usuarios de todo el mundo" (Rouse, 2019).

"Telegram también ofrece una API abierta que permite a los desarrolladores crear bots personalizados para realizar diversas funciones y tareas dentro de la aplicación" (Krishna, 2020).

"Telegram también permite a los usuarios crear y unirse a canales y grupos públicos para discutir y compartir información sobre diversos temas" (Lomas, 2018).

Aspectos metodológicos.

A continuación, se explica el tipo de investigación y el diseño metodológico a implementar con el fin de cumplir con los objetivos del proyecto.

Tipo de investigación.

En este proyecto propositivo, tuvo como objetivo encontrar una posible solución al problema planteado. Hace algunos años, la solución más común para atender llamadas telefónicas mientras se estaba fuera de casa era el uso de contestadores automáticos. Sin embargo, con la popularización de los teléfonos móviles, esta solución se volvió obsoleta ya que estos dispositivos son portátiles y permiten recibir y hacer llamadas desde cualquier lugar. Actualmente, gracias a la integración de internet en los objetos cotidianos, es posible desarrollar una función similar en cualquier dispositivo conectado a la red.

Método y metodología.

Teniendo en cuenta que la opción más recomendable para el problema es una puerta que permite la comunicación con otros dispositivos en red y que puede ampliar todas las funciones que se ajusta a lo requerido, su funcionamiento permite que una puerta la cual no estaba diseñada para tener tecnología ahora la tenga, se propone aplicar un método analítico el cual facilitará el diseño de este dispositivo.

La metodología que contiene el desarrollo de este proyecto, describe las diferentes actividades a realizar.

Alcance y Limitaciones

Alcance

La solución propuesta para abordar el problema planteado se basa en la implementación de tres *RPi3* conectadas a monitores y un módulo de cámara 1080. La primera *RPi3*, conectada al módulo de cámara y a un monitor, se encargará de capturar un video en *Streaming* y guardarlo. Además, enviará una notificación a través de la aplicación Telegram para alertar al usuario sobre la presencia de alguien en la puerta. También incluirá un sistema de detección constante de usuarios cercanos mediante *Bluetooth-beacon*, lo que permitirá visualizar la imagen transmitida por la cámara en un monitor cercano.

Las otras dos *RPi3* se utilizarán para controlar los televisores o monitores dentro de la casa y convertirlos en dispositivos inteligentes capaces de mostrar en vídeo la imagen del visitante que se encuentre en la puerta. Esta imagen se visualizará en el televisor más cercano al usuario gracias al sistema de detección Bluetooth-beacon programado en cada RPi3.

Las tres *RPi3* se conectarán mediante Wi-Fi en la misma red, lo que permitirá a la puerta inteligente publicar el video capturado por la cámara en un televisor o monitor inteligente cuando un usuario esté cerca. El sistema decidirá automáticamente la conexión disponible para salir a internet solo en caso de que el usuario no se encuentre en el hogar, y la autenticación del usuario mediante contraseña garantizará la seguridad del sistema. Después de un tiempo determinado o cuando el usuario lo decida, se cerrará la descarga continua de video, lo que mejorará aún más la seguridad de la red al no dejar puertos abiertos o vulnerables.

En conclusión, el alcance de este proyecto se limita al desarrollo del documento el cual propone un prototipo para el funcionamiento del dispositivo propuesto.

Limitaciones

Las limitaciones de este proyecto se especifican de la siguiente manera:

- Tiempo para hacer la elaboración del producto.
- Implementación de inteligencia artificial.
- Poner en marcha el proyecto con los temas visto de PMI.

Productos esperados.

Documento estructurado que contiene la propuesta de la solución al problema planteado.

Plan de trabajo/Contenido propuesto.

Plan de trabajo

Teniendo en cuenta la metodología Scrum como framework para la elaboración de proyectos, se propone como base para la elaboración de este proyecto el cual se implementa de la siguiente forma:

- ***Creación del producto backlog****:*

  Se procede a identificar las tareas que se van a realizar en el proyecto y priorizarlas de mayor a menor según la importancia que se tenga.

- ***Estimación de tiempo para el desarrollo del proyecto (Sprint):***

  Teniendo organizado la lista de tareas que se van a realizar, se procede a estimar el tiempo que se va a tomar en la realización de cada una de ellas, para este proyecto se considera pertinente hacer entregables cada sema, el cual se puede tener mayor control y seguimiento.

- ***Ejecución del proyecto (Scrum Team):***
    - Etapa de desarrollo: Se procede con la creación del algoritmo, instalación del software y hardware, habilitar los protocolos TCP/IP que se van a utilizar.
    - Etapa de pruebas: Se procede hacer la búsqueda de cualquier falla o defecto y la validación de la implementación de los requerimientos que se indicaron.

- ***Finalización del proyecto:***

Teniendo el producto terminado, se procede hacer la socialización, la entrega de la documentación requerida y la implementación del producto en la puerta principal de la vivienda.

Contenido propuesto

A continuación, se plantea los capítulos del *contenido propuesto* que conformará el documento de la investigación:

Resumen

Introducción

Capítulo I descripción del proyecto

1.1. *Planteamiento del problema.*

1.2. *Objetivos.*

1.3. *Objetivo general.*

1.4. *Objetivos específicos.*

1.5. *Justificación.*

Cronograma de actividades.

		Cronograma															
		MES 1				**MES 2**				**MES 3**				**MES 4**			
No	**ACTIVIDAD**	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
1	**Creación del producto backlog**																
1.1	Creación del listado de tareas																
1.2	Dar prioridad a las tareas																
2	**estimación de tiempo**																
2.1	Estimación de tiempo por tarea																
2.2	Socialización al equipo																
3	**Etapa de desarrollo**																
3.1	Elaboración del algoritmo																
3.2	Configuración del modem																
3.3	Instalación de SW en las RPI3																
3.4	Configuración del Hardware																
3.5	Adecuación del cableado																
4	**Etapa de pruebas**																
4.1	Reporte de bug																
4.2	Corrección del bug																
4.3	Regresión de pruebas																

4.4	Elaboración de documentación															
5	**Entrega del Proyecto**															
5.1	Socialización															
5.2	Entrega del manual															
5.3	Instalación del proyecto															

Presupuesto y recursos económicos.

A continuación, se establecen los recursos y el presupuesto requeridos para el desarrollo del proyecto.

Tabla 1. Descripción general del presupuesto

RUBROS	**RECURSOS**		**TOTAL**
	Recursos propios	**Otras fuentes**	
Personal	$ 32.000.000		$ 32.000.000
Equipos	$ 6.000.000		$ 12.800.000
Software	$ 2.080.000		$ 2.080.000
Materiales	$ 5.000.000		$ 5.000.000
TOTAL			**$ 51.620.000**

Tabla 1.1 Descripción de los gastos de personal

Investigador / Experto/ Auxiliar	Formación académica	Función dentro del proyecto	DEDICACIÓN Horas / mes	RECURSOS		TOTAL
				Propios	Otras fuentes	
Guillermo Romero	Profesional	Investigador	50	$ 16.000.000		$ 16.000.000
Prof. Redes	Profesional		50	$ 16.000.000		$ 16.000.000
TOTAL						**$ 32.000.000**

Tabla 1.2 Descripción de los equipos que se planea usar

EQUIPO	JUSTIFICACIÓN	RECURSOS		TOTAL
		Propios	Otras fuentes	
Portátil 1	Estación de trabajo, investigador 1	X		$ 3.000.000
Portátil 2	Estación de trabajo, investigador 2	X		$ 3.000.000
Servidor	Equipo de cómputo servidor para almacenamiento y procesamiento de datos	X		$ 6.000.000
Raspberry	3 dispositivos instalados en los modem de los clientes para captura de datos	X		$ 800.000
TOTAL				**$ 12.800.000**

Tabla 1.3 Descripción del software que se planea adquirir

SOFTWARE	JUSTIFICACIÓN	RECURSOS		TOTAL
		Propios	Otras fuentes	
Windows server	Sistema operativo para servidor	X		$ 1.320.000
SQL server	Gestión de Bases de Datos	X		$ 560.000
Ofimática	Requerida para construcción documental	X		$ 200.000
TOTAL				**$ 2.080.000**

Tabla 1.4 Materiales y suministros

MATERIALES	JUSTIFICACIÓN	RECURSOS		TOTAL
		Propios	Otras fuentes	
Papelería	Impresión de documentos para las entregas del proyecto.	X		$ 300.000
Internet	Trabajo en documentos compartidos, consulta de información, envío de correos.	X		$ 400.000
Medios digitales	Entrega digital a la decanatura de la especialización	X		$ 20.000
Energía	Servicio público de energía utilizado durante el proyecto	X		$ 200.000

Cableado UTP	Conexión de las RPI	X		$ 3.680.000
Transportes	Desplazamiento del equipo de trabajo para el cumplimiento del plan de trabajo	X		$ 400.000
TOTAL				**$ 5.000.000**

Desarrollo de la propuesta.

La construcción de esta propuesta se realizó de la siguiente forma:

El diseño de la propuesta se planteó para hacer una implementación bajo esquema de CISCO con algunas de las características del IoT.

Lista de materiales.

Los materiales necesarios para el desarrollo de esta propuesta son:

- *3 placas Raspberry Pi 3 Modelo B.*
- *3 tarjetas de memoria Micro SD de mínimo 16 GB categoría 10*
- *1 módulo de cámara 1080 para Raspberry Pi de 5 Megapíxeles*
- *Imagen ISO de sistema operativo Raspbian Stretch with Desktop*
- *Aplicación Etcher.io*
- *2 puentes tipo hembra para conexión GPIO*
- *1 botón*
- *Aplicación para Android iBeaconDroid descargada desde PlayStore.*
- *3 cargadores de 2.5A*
- *3 monitores con puertos HDMI.*
- *3 cables HDMI*
- *Mouse y teclado con conexión USB.*
-

Instalación de sistema operativo

Haciendo uso de la herramienta *Etcher*, seleccione el sistema operativo y la Memoria SD en la cual se desea instalar el sistema operativo, como se indica en la Figura 1.

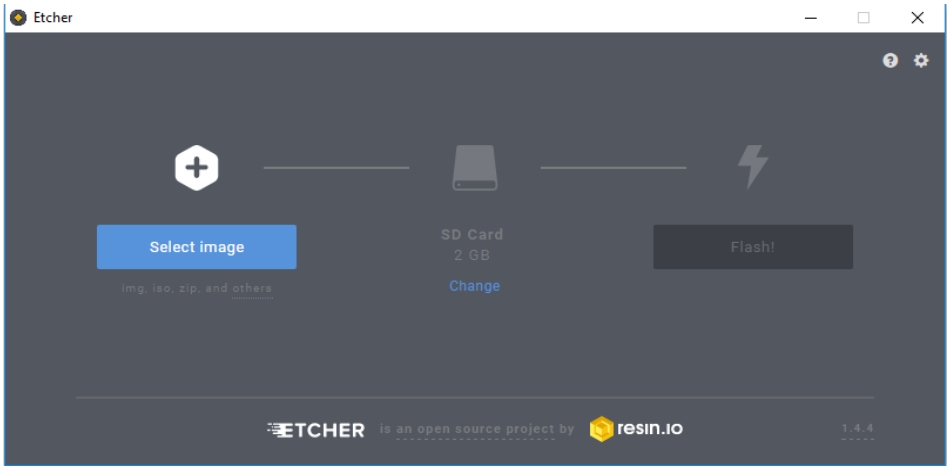

Figura 1. Programa Etcher, herramienta de instalación del Sistema Operativo en SD.

Fuente: El autor

Luego de finalizada la instalación del sistema operativo por medio de *Etcher*, se procede a colocar la tarjeta Micro SD dentro de una RPi3, conectar la Rpi3 a una pantalla y por medio de un cargador a una toma de corriente eléctrica.

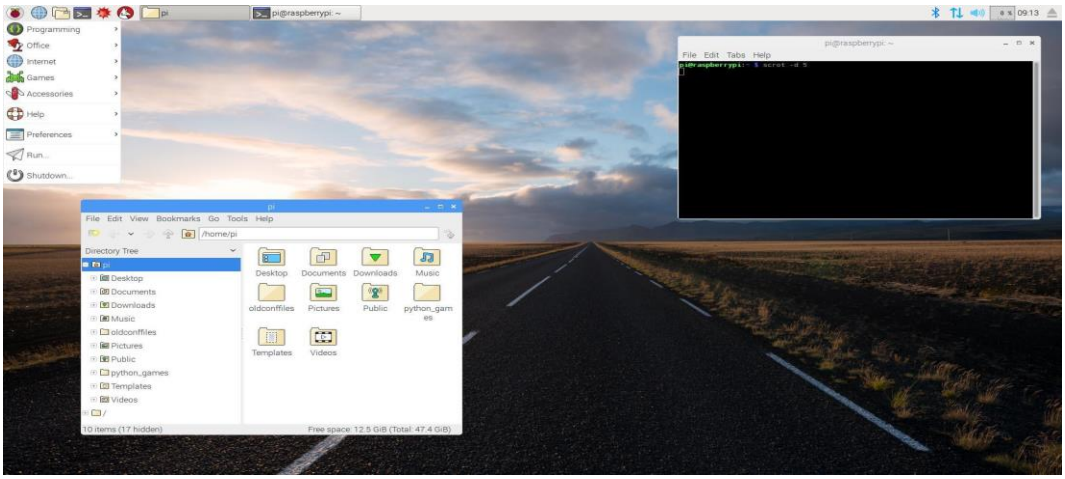

Figura 2. Escritorio del Sistema Operativo Raspbian sobre la RPi3-Model B.

Fuente: El autor

La Figura 2 muestra la primera visualización del Sistema Operativo Raspbian *Stretch with Desktop*, luego de su instalación en la tarjeta Micro SD.

Actualización del sistema operativo

Los siguientes comandos, se deben ejecutar en las terminales de cada Rpi3:

- *sudo apt-get update*
- *sudo apt-get upgrade*
- *sudo apt-get dist-upgrade*

Implementación de la cámara con la rpi3

En una RPi3 se instala los siguientes componentes:

- Se conecta la cámara en el puerto serial indicado para el manejo de este dispositivo, como indica la Figura 3.

Figura 3. RaspberryPi 3 modelo B Figura

Tomada de: https://www.raspberrypi.org/

Activación de la cámara por comando en el terminal

- *sudo raspi-config*

Con la ejecución de este comando aparecerá una pantalla de configuración, seleccione la opción *Interfacing Options, Camera,* el sistema preguntará si quiere habilitar la cámara, se indica *Yes* y *finish.*

Implementación descarga continua de video (*streaming*) para la cámara

Para implementar el sistema de descarga continua de video para la cámara se utiliza la herramienta llamada *mjpg_streamer,* para su instalación y descarga se usan los siguientes comandos, digitados en la terminal:

- *sudo apt-get install subversion libjpeg8-dev imagemagick libav-tools* ***cmake.***
- *sudo apt-get install miniupnpc*
- *sudo pip install telepot*
- *cd Desktop*
- *Git clone https://github.com/DedSecZero/BotTelegramIoT.git*

con el comando anterior se descarga las fuentes actualizadas en el escritorio, dentro del cual se creará una carpeta de nombre *BotTelegramIoT*, dentro de esta carpeta se encontrará lo siguiente:

- La publicación de la descarga continua de vídeo sobre la IP pública se realiza sobre el protocolo Internet Gateway Device IGD
- Directorio AutoBot, el cual contiene el programa principal en el cual se creó el Bot para la aplicación Telegram, este programa se llama botRaspberryPiIoT.py desarrollado en Python, para ver el código dirigirse al Anexo 1.
- Dentro del directorio AutoBot también se encuentra el programa timbre.py el cual se encarga de estar pendiente si alguien timbra en la puerta, en caso de que ocurra, crear un Backup en el la ruta /home/pi/Backup/ , dentro de la cual se encuentra los videos con la fecha y hora del momento en que timbraron, además enviará un mensaje por mensajería

instantánea al usuario principal notificándole que alguien ha tocado el timbre, de la misma formar enviará un link al usuario en el cual podrá ver por descarga continua de video la persona que se encuentra en ese momento frente a la puerta para visualizar el código fuente de timbre.py dirigirse al Anexo 2.

Configuración de *bot* en telegram.

Se debe ejecutar el siguiente comando:

- *sudo pip install telegram-send*

Inicia con una conversación por *Telegram* con el Bot *@BotFather*, como indica la Figura 4.

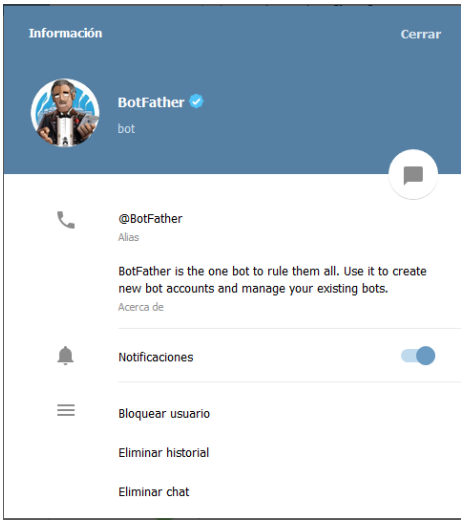

Figura 4. Bot de Telegram BotFather.

Fuente: https://telegram.me/BotFather

Para iniciar la conversación con el *BotFather* se puede escribe un mensaje y luego se presiona el botón *iniciar* que Telegram indica, cuando inicia la conversación con el Bot indicara que nos proporciona diferentes opciones como se referencia en la Figura 5:

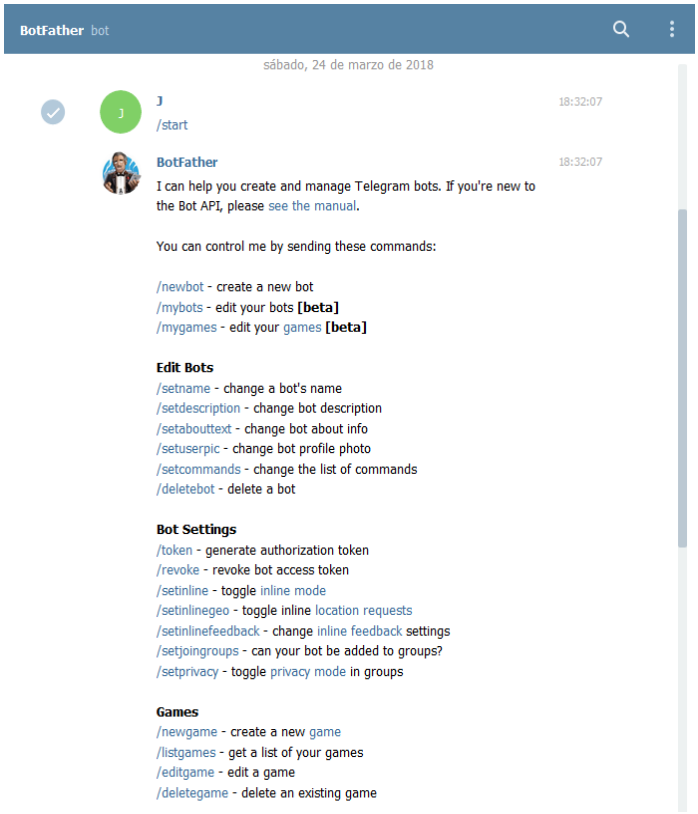

Figura 5. BotFather, funciones.

Fuente: Los Autores, https://telegram.me/BotFather

Se continua con la conversación usando el Bot *BotFather* para la creación del Bot.

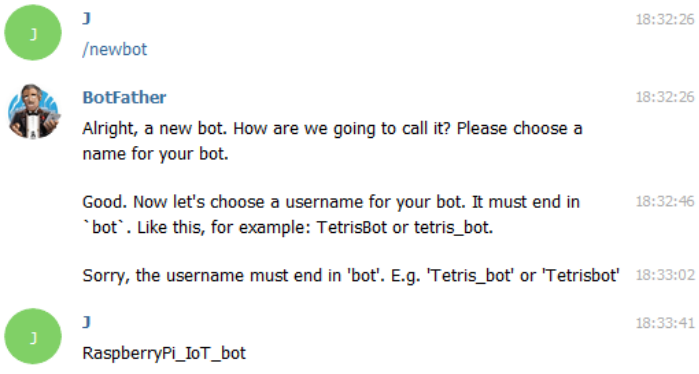

Figura 6. Creando Bot con BotFather.

Fuente: Los Autores, https://telegram.me/BotFather

La Figura 6 muestra la creación de un Bot propio, usando la función *newbot* y *BotFather* donde pide que le se le envié el nombre con el cual se creará el Bot.

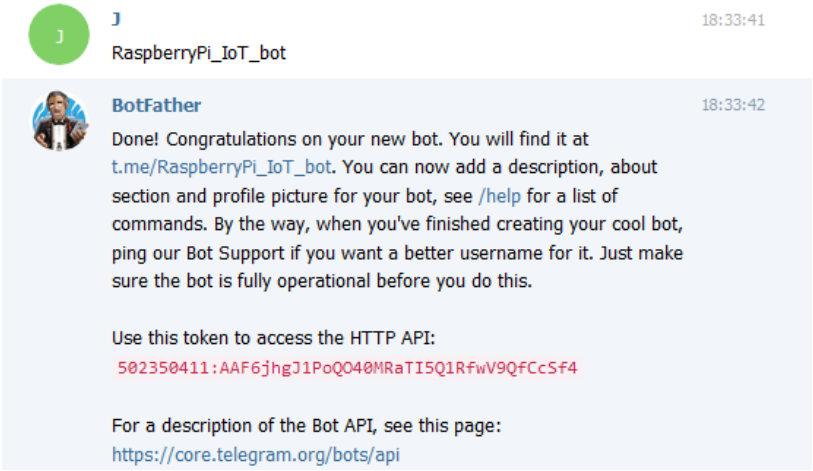

Figura 7. Token del Bot que se ha creado.

Fuente: Los Autores, https://telegram.me/BotFather

Luego de proporcionar el nombre con el cual sé quiere crear el Bot, *BotFather* proporcionara el *link* (*t.me/RaspberryPi_IoT_bot*) en el cual se puede encontrar el Bot, y adicionalmente proporciona el token con donde se puede crear el código fuente del Bot como indica la Figura 7, este token se debe colocar en el programa de *BotRaspberryPiIot.py.*

El token que *BotFather* proporcionado, servirá para configurar el Bot desde la terminal y poder establecer la comunicación, no solo desde el programa del Bot principal si no desde cualquier otro programa de Python, para esto se ejecuta el siguiente comando en la terminal:

telegram-send --configure

Al momento de ejecutar este comando, se debe ingresar el *token* que *BotFather* nos proporcionó como se referencia en la Figura 8:

```
root@DedSec:~# telegram-send --configure
Talk with the BotFather on Telegram (https://telegram.me/BotFather), create a bot and insert the token
› 605945594:AAF_FS027MVHyZ68ZyUcgztE5hn0Zc8GcxE
Connected with Test_RaspberryPiIoT_bot.

Please add Test_RaspberryPiIoT_bot on Telegram (https://telegram.me/Test_RaspberryPiIoT_bot)
and send it the password: 14826
```

Figura 8. Terminal de Linux Raspbian configurando telegram-send.

Fuente: Los Autores.

Luego de ingresar el token, envía un código, en este caso es14826, este código se envía al Bot:

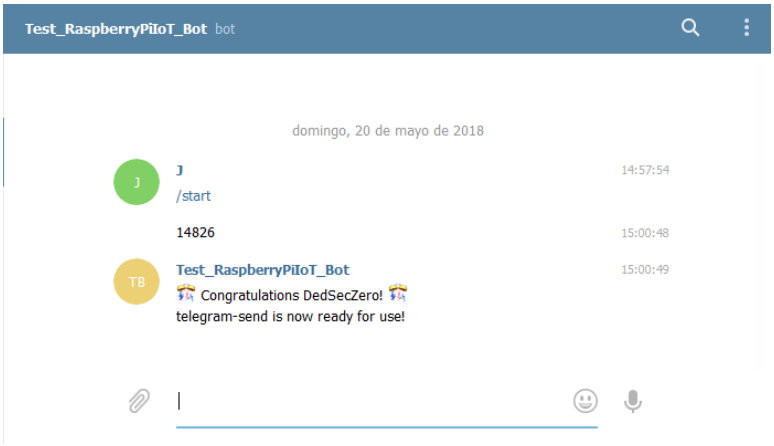

Figura 9. Bot Configurado.

Fuente: El autor, https://telegram.me/RaspberryPi_IoT_bot

Al ingresar el código como se indica en la Figura 9, se configura el Bot en la *RPI3*, para probarlo se ejecuta el comando:

telegram-send "Hola, Felicidades has configurado tu Bot exitosamente"

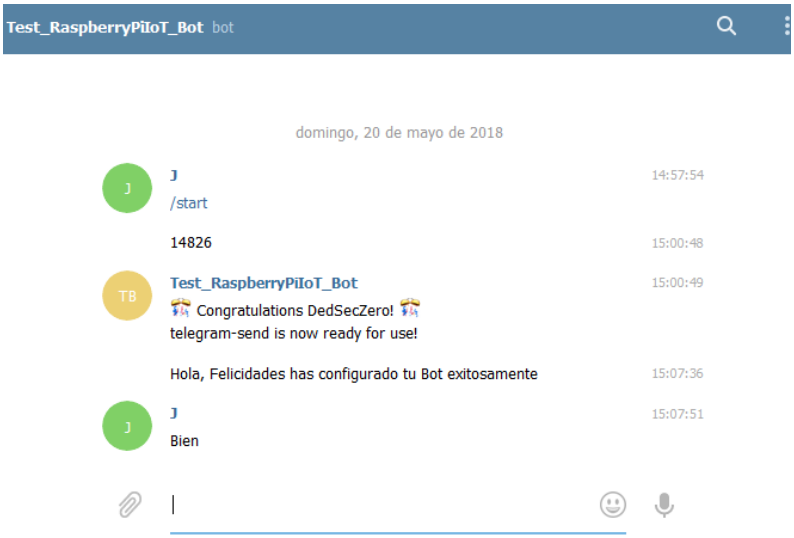

Figura 10. Pruebas de mensajería.

Fuente: Los Autores, https://telegram.me/RaspberryPi_IoT_bot

Al ejecutado se ha enviado un mensaje de parte del Bot por *Telegram* como se referencia en la Figura 10.

Detección de usuario por medio de *beacon* de bluetooth

La siguiente configuración debe realizarse en las 3 Rpi3.

Para configurar el uso de *Beacons* de *Bluetooth* e detectar al usuario se deben ejecutar los siguientes comandos por la terminal:

- *sudo pip install beacontools*
- *sudo apt-get install python-dev libbluetooth-dev libcap2-bin*
- *sudo setcap 'cap_net_raw,cap_net_admin+eip' $(readlink -f $(which python))*
- *pip install beacontools[scan]*
- *pip install pybluez*
- *pip install pybluez[ble]*
- *cd Desktop*

- *git clone https://github.com/DedSecZero/IBeaconIoT_TelnetTool.git*

Luego de ejecutar los comandos se descargarán las fuentes de Git en la carpeta *IBeaconIoT_TelnetTool* dentro del cual se encontrará una carpeta con el nombre *beacontools,* dentro de esta carpeta se encuentran las fuentes a instalar para que las RPi3 puedan detectar y hacer uso de Bluetooth Low Energy *BLE* detectando Beacons de Bluetooth.

Para realizar de manera más completa la instalación de *beacontools* se debe ejecutar los siguientes comandos por la terminal:

- *cd IBeaconIoT_TelnetTool/beacontools*
- *sudo python setup.py install*

Dentro del directorio *beacontools* se encuentra el directorio *examples, en esta carpeta* se encuentran diversos ejemplos de uso de Beacons de Bluetooth, para el desarrollo de la solución se crearon los programas en Python *telnet.py* ***(****ver código fuente en anexo 3****),*** *scanner_beacon.py* ***(****ver código fuente en anexo 4****),*** *openBrowser.py* ***(****ver código fuente en anexo 5****).***

Ejecución de cada componente

Para ejecutar los siguientes comandos se deben hacer en diferentes terminales:

- *Primera ventana**:** Iniciar programa del Bot:*
- *sudo Python*

/home/pi/Desktop/BotTelegramIoT/AutoBot/botRaspberryPiIot.py

- *Segunda ventana**:** Iniciar ejecución de programa timbre.py*
- *sudo Python /home/pi/Desktop/BotTelegramIoT/AutoBot/timbre.py*
- *Tercera ventana**:** Iniciar ejecución del programa telnet.py el cual hace la búsqueda del usuario cuando timbran en la puerta.*
- *Sudo Python*

/home/pi/Desktop/IBeaconIoT_TelnetTool/beacontools/examples/telnet.py

Nota: *las ejecuciones de la primera y segunda ventana solo se realizan en la RPi3 la cual contiene la camara, para las otras 2 RPi3 se ejecuta el comando de la tercera ventana.*

Diagramas secuencias de ejecución

Teniendo en cuenta el desarrollo de la solución y la funcionalidad completa del sistema de puerta inteligente se diseñaron los siguientes diagramas de secuencia (Figura 11) y de componentes (Figura 12) , los cuales brindan la perspectiva en orden de ejecución de los componentes:

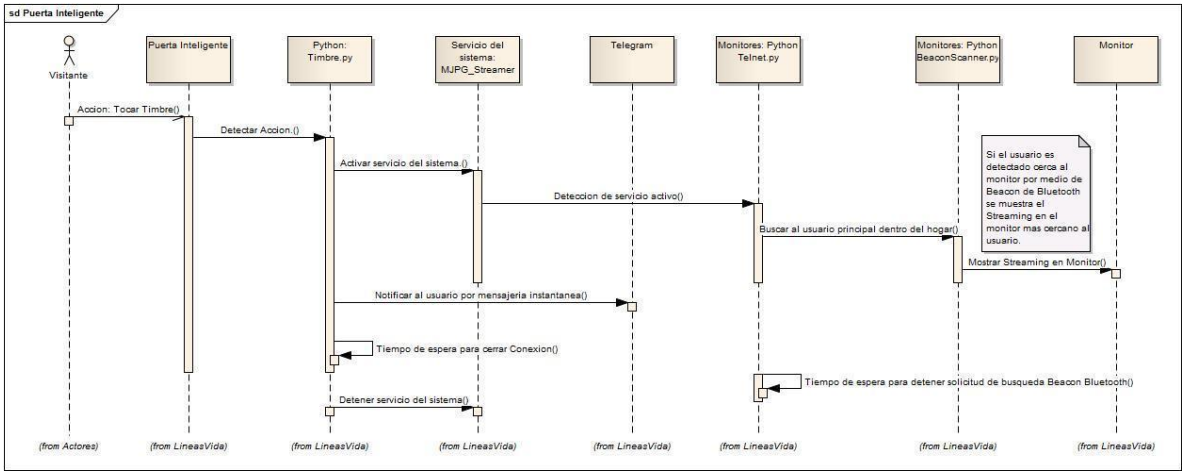

Figura 11. Secuencia de ejecución de los componentes en Rpi3 de la puerta inteligente. Fuente: El autor.

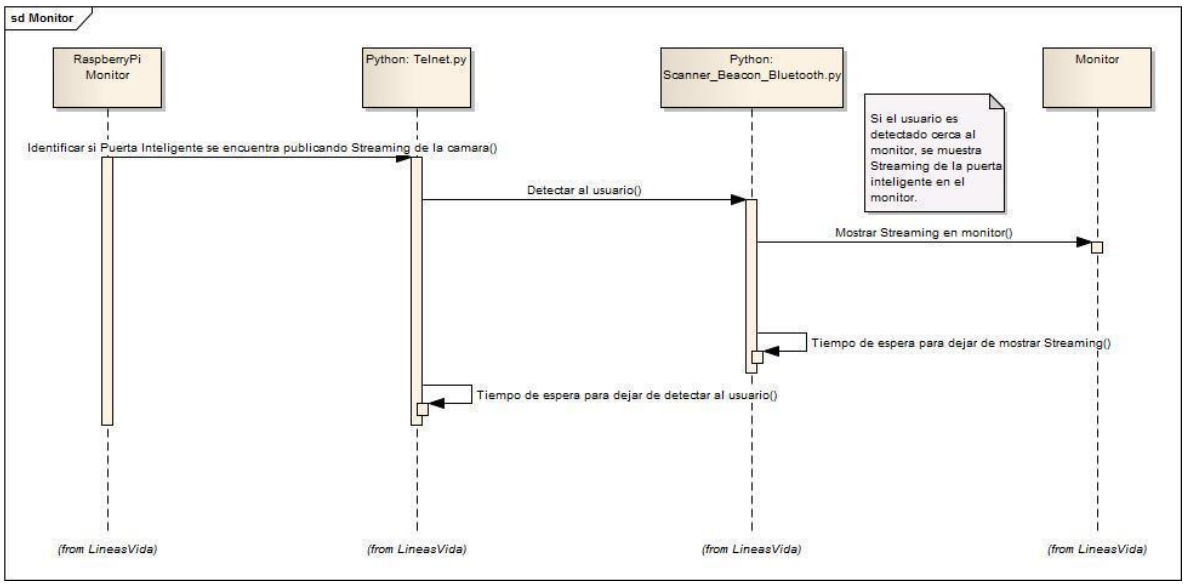

Figura 12. Diagrama de secuencia de los componentes de las 2 Rpi3 que detectan al usuario principal.

Fuente: E autor.

Diagrama de conexión en red de las RPI3

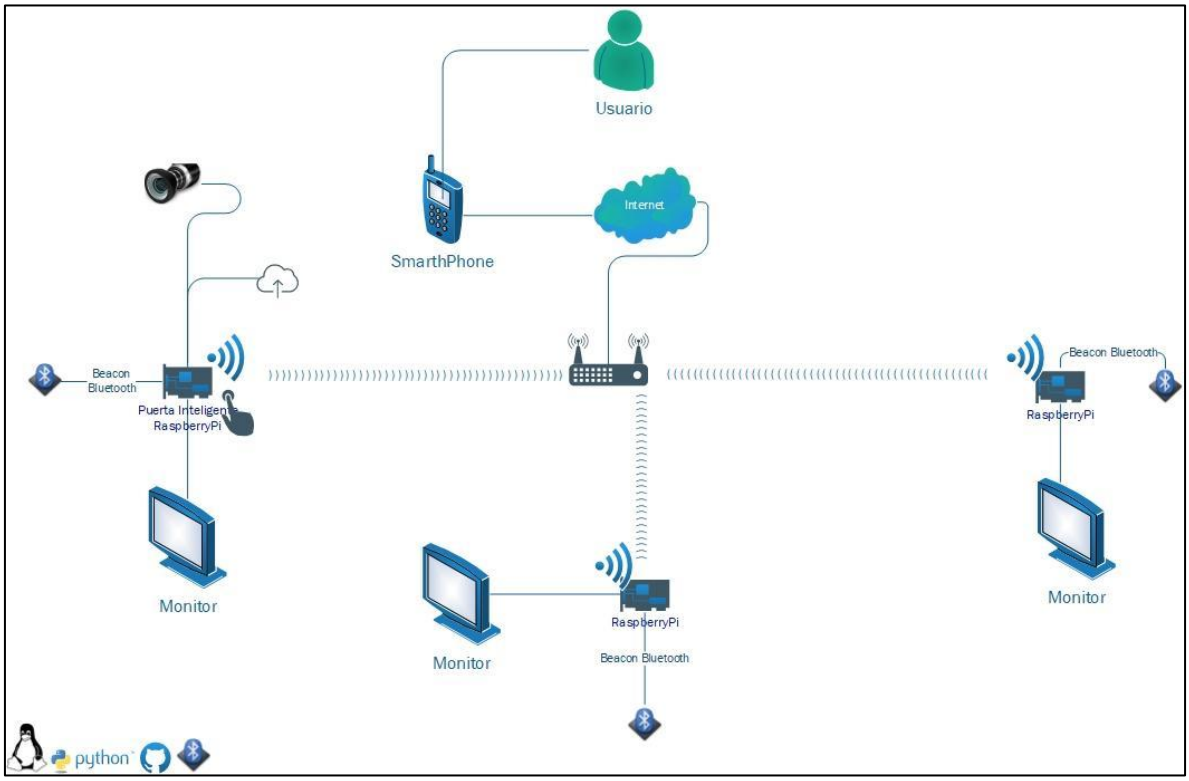

Figura 13. Diagrama de red y componentes.

Fuente: El autor.

La Figura 13 indica la distribución de los componentes dentro de la red del hogar, muestra la interacción del usuario con el sistema en caso de no encontrarse dentro del hogar. El lado izquierdo de la Figura se encuentra la Rpi3 encargada de la puerta inteligente, la cual tendra una cámara para visualizar a la persona que toca el timbre, cuenta también con la conexión al timbre, en la parte baja de la Figura y a la parte derecha se encuentran 2 Rpi3 encargadas de analizar si el usuario principal se encuentra dentro del hogar, en el caso de que se encuentre y alguien tocara el timbre, el monitor al cual el usuario se encuentre más cerca le publicara un video de descarga continua de la persona que se encuentra oprimiendo el timbre.

Conclusiones.

1. Desde el punto de vista de la especialización en proyectos informáticos, se logró implementar exitosamente buenas prácticas en el desarrollo del proyecto. Se aplicaron de manera efectiva los conceptos y herramientas adquiridos a lo largo de esta especialización.

2. Se puede desarrollar soluciones tecnológicas, las cuales ayudan a mejorar una existente o implementarla donde no existía.

3. El Internet de las cosas son dispositivos conectados a la red para múltiples funciones, donde su comunicación está dada por un *GateWay IOT* el cual permite que todos los dispositivos se comuniquen entre sí, sin importar qué protocolo de comunicación utilicen.

Bibliografía y referencias bibliográficas.

Bibliografía

Ing. Ariel Pisano, 2018, Gestión de Servicios Tecnológicos y de Telecomunicaciones (Tesis de Maestría) Universidad de San Andres, Buenos Aires Argentina.

(Evans, 2011) Internet de las cosas – Como la próxima evolución de Internet lo cambia todo, Cisco https://www.cisco.com/c/dam/global/es_mx/solutions/executive/assets/pdf/internet-of-things-iot-ibsg.pdf

(ITU,2015) ITU Internet Report 2005, “The Internet of Things”. 2005. https://www.itu.int/net/wsis/tunis/newsroom/stats/The-Internet-of-Things-2005.pdf

(MGI, 2018), McKinsey Global Institute, “The Internet of Things: How to capture the value of IoT” Mayo, 2018, https://www.mckinsey.com/~/media/McKinsey/Business%20Functions/McKinsey%20Digital/Our%20Insights/The%20Internet%20of%20Things%20How%20to%20capture%20the%20value%20of%20IoT/How-to-capture-the-value-of-IoT.pdf

(Paneta, 2017) Kasey Panetta, Gartner Top Strategic Predictions for 2018 and Beyond (03/10/2017) https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond

Referencias bibliográficas

Boccardo, Y., Montejano, G., & Riesco, D. (2016). Patrón de Diseño Beacon Action Manager para comunicar Aplicaciones Móviles (IoT). Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação, 1(6).

International Journal of Engineering Research & Technology (IJERT), 12 Diciembre 2016, IOT based Smart Home Security System with Alert and Door Acces Control using Smart Phone, Ijert, volumen 5.

Taewan Kim, Hyungsoo Park, and Yunmo Chung, 2013, Integration of Face Recognition and Sound Localization for a Smart Door Phone System, IEEE International Conference on Consumer Electronics (ICCE).

Matti Pietikäinen, Machine Vision Group, Departamento de Ingeniería Eléctrica e Informática, Universidad de Oulu, Finlandia.

One Media Group, 28/10/2014, Samsung Smart Door Lock Promotion Video, https://youtu.be/tfYTnjrlQQk.

Marcelo Pedra, 21/08/2016, Que es BlueTooth BLE, http://www.marcelopedra.com.ar/blog/2014/01/06/que-es-el-bluetooth-le/

Dave Evans, Abril 2011, Internet de las cosas Cómo la próxima evolución de Internet lo cambia todo, Cisco Internet Business Solutions Group (IBSG)

https://www.python.org/

https://telegram.org/

Garcia-Quesada, M. J. (2016). Bot para Telegram que ejecute Aventuras Conversacionales.

Garcia Segura, D. F., & Oliva Vallejos, M. A. (2018). Diseño de un Soft-Plc Basado en un Computador de Placa Reducida (SBC) Raspberry.

Beazley, D. M. (2009). Python essential reference. Addison-Wesley Professional.

Lutz, M. (2013). Learning Python. O'Reilly Media, Inc.

Rossum, G. V. (1995). Python reference manual. CWI.

Van Rossum, G. (2000). An introduction to Python for UNIX/C programmers. PythonLabs.

Van Rossum, G. (2003). Python tutorial. Python Software Foundation.

Guthrie, C. (2017). Single Board Computers: An Overview. IEEE Potentials, 36(1), 17-21.

Hughes, J. (2018). Why single-board computers are great IoT devices. Network World.

Novasom Industries (2019). Single Board Computer Applications.

Thompson, M. (2018). Single-Board Computers: A Programmer's Perspective. IEEE Software, 35(1), 96-98.

Upton, E. (2015). The Raspberry Pi single-board computer and Linux. Journal of Open Research Software, 3(1), e6.

Krishna, V. (2020). Telegram: The Secure Messaging App Explained. Security Boulevard.

Lomas, N. (2018). What is Telegram, and is it secure? Wired.

Rouse, M. (2019). Telegram. TechTarget.

Anexos.

Anexo 1 BOTRASPBERRYPIIOT.PY

```
#Autor: Johan Garcia - Guillermo Romero -IoT 2018
#Librerias
import telegram
from telegram.ext import *
import os
import sys
import time
import subprocess
```

```
import telepot,threading
from telepot.loop import MessageLoop


pf= '/usr/local/bin/pf'
ipaddr = '/usr/local/bin/ipaddr'
EXTERNAL_PORT = 54321
INTERNAL_PORT = 8080 #Puerto default del servicio mjpg_streamer


#Indicar el Id del Bot sobre el cual vamos a trabajar
mi_bot = telegram.Bot(token='###')
#Actualizar con regularidad el ID del bot
mi_bot_updater = Updater(mi_bot.token)
#Definicion de funcion Start
def start(bot, update, pass_chat_data=True):
   update.message.chat_id
   bot.sendMessage(chat_id=update.message.chat_id,text='Hola, Soy Brainiac')
   bot.sendMessage(chat_id=update.message.chat_id,text='Soy quien se encarga de estar
atento por si alguien llama a tu puerta')
#Definicion de funcion Close
def close(bot, update, pass_chat_data=True):
   update.message.chat_id
   bot.sendMessage(chat_id=update.message.chat_id,text='Lo siento pero vigilo 24/7, yo no
descanso Jefe.')
   #os.system('telegram-send --image noDormir.jpg --caption "Lo siento pero yo no descanso
Jefe"')
```

```
#Definicion de funcion ObtenerFoto
def obtenerFoto(bot, update,pass_chat_data=True):
   update.message.chat_id
   bot.sendMessage(chat_id=update.message.chat_id,text='Con gusto jefe dame un momento y te envio la foto...')
   os.system('rm foto.jpg')
   os.system('raspistill -o foto.jpg')
   #os.system('telegram-send --image foto.jpg --caption "Aqui tienes Jefe"')
   bot.send_photo(chat_id=update.message.chat_id,photo=open('foto.jpg','rb'))
#Definicion de funcion ObtenerVideo
def obtenerVideo(bot, update, pass_chat_data=True):
   update.message.chat_id
   bot.sendMessage(chat_id=update.message.chat_id,text='Con gusto jefe dame un momento y te envio el video...')
   os.system('rm video.h264')
   os.system('rm video.mp4')
   os.system("raspivid -o video.h264 -t 5000")
   os.system("MP4Box -add video.h264 video.mp4")
   os.system('telegram-send --file video.mp4 --timeout 40.0')
   bot.sendMessage(chat_id=update.message.chat_id,text='Lamento la demora, espero puedas disculparme')
#Definicion de funcion para inicio de streaming
def iniciarStreaming(bot,update,pass_chat_data=True):
   global pf,ipaddr
```

```
    #subprocess.call(['sudo', 'systemctl', 'start' , 'mjpg_streamer.service'])
    threading.Thread(target=StartMjpgService).start()
    update.message.chat_id
    #subprocess.call([pf,str(EXTERNAL_PORT),str(INTERNAL_PORT)])
    #out = subprocess.check_output([ipaddr])
    #ip = dict([line.split('=') for line in out.decode('ascii').strip().split('\n')])
    #reply = 'http://%s:%d/?action=stream' % (ip['External'], EXTERNAL_PORT)
    reply = 'http://192.168.43.29:8080/?action=stream'
    #if ip['External'] != ip['Public']:
    #   reply += '\nNo es posible realizar streaming'

    bot.sendMessage(chat_id=update.message.chat_id, text=reply)
#Definicion de funcion para finalizar streaming
def finalizarStreaming(bot,update,pass_chat_data=True):
    global pf,ipaddr
    update.message.chat_id
    #subprocess.call([pf, 'delete', str(EXTERNAL_PORT)])
    #subprocess.call(['sudo', 'systemctl', 'stop', 'mjpg_streamer.service'])
    StopMjpgService()
    bot.sendMessage(chat_id=update.message.chat_id,text='Streaming Finalizado')
#Listener de conversacion
def listener(bot,update,pass_chat_data=True):
    update.message.chat_id
    mensaje = update.message.text
    mensajeLower = mensaje.lower()
```

```
  if(mensajeLower == 'hola'):
    bot.sendMessage(chat_id=update.message.chat_id,text='Si deseas usar alguna de mis funciones puedes usar las siguientes: ')
    bot.sendMessage(chat_id=update.message.chat_id,text='/obtenerFoto ,  esta funcion hara que tome una foto y te la envie.')
    bot.sendMessage(chat_id=update.message.chat_id,text='/obtenerVideo, esta funcion me permite enviarte un video del momento, suelo demorarme un poco con esta tarea espero me disculpes')
    bot.sendMessage(chat_id=update.message.chat_id,text='/iniciarStreaming, iniciar streaming')
    bot.sendMessage(chat_id=update.message.chat_id,text='/finalizarStreaming, terminar streaming')
  elif(mensajeLower.find('preocupes')):
    bot.sendMessage(chat_id=update.message.chat_id,text='Eres el mejor')
  else:
    bot.sendMessage(chat_id=update.message.chat_id,text='lo siento pero no tengo nada que responder ante eso :Â´( ')
#Inicion y Fin de servicios
def StartMjpgService():
  os.system('/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/mjpg_streamer -i "/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/input_raspicam.so -fps 60 -quality 10 -x 400 -y 300" -o "/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/output_http.so -c SmartDoor:IoT2018"')
def StopMjpgService():
```

```
    os.system("killall mjpg_streamer")


#-----------------------------------------------Fin de Definiciones ----------------------------------------------
close_handler = CommandHandler('close',close)
start_handler = CommandHandler('start',start)
obtenerFoto_handler = CommandHandler('obtenerFoto',obtenerFoto)
obtenerVideo_handler = CommandHandler('obtenerVideo',obtenerVideo)
listener_handler = MessageHandler(Filters.text,listener)
iniStreaming_handler = CommandHandler('iniciarStreaming',iniciarStreaming)
finStreaming_handler = CommandHandler('finalizarStreaming',finalizarStreaming)


dispatcher = mi_bot_updater.dispatcher


dispatcher.add_handler(start_handler)
dispatcher.add_handler(close_handler)
dispatcher.add_handler(obtenerFoto_handler)
dispatcher.add_handler(obtenerVideo_handler)
dispatcher.add_handler(listener_handler)
dispatcher.add_handler(iniStreaming_handler)
dispatcher.add_handler(finStreaming_handler)
mi_bot_updater.start_polling()
mi_bot_updater.idle()
MessageLoop(mi_bot.token, iniciarStreaming).run_as_thread()
while True:
    pass
```

Anexo 2 TIMBRE.PY

```
import RPi.GPIO as GPIO
import time,os
import subprocess
import threading
def Backup():
   os.system("sudo python /home/pi/Desktop/BotTelegramIoT/AutoBot/backup.py")
def StartMjpgService():
   #os.system('/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/mjpg_streamer -i "./input_raspicam.so -fps 60 -quality 10 -x 400 -y 300" -o "./output_http.so -c SmartDoor:IoT2018"')
   os.system('/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/mjpg_streamer -i "/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/input_raspicam.so -fps 60 -quality 10 -x 400 -y 300" -o "/home/pi/Desktop/BotTelegramIoT/Streaming/mjpg-streamer/mjpg-streamer-experimental/output_http.so"')
def StopMjpgService():
   os.system("killall mjpg_streamer")


GPIO.setmode(GPIO.BCM)
GPIO.setup(23,GPIO.IN,pull_up_down=GPIO.PUD_UP) #BOTON
pf = '/usr/local/bin/pf'
ipaddr = '/usr/local/bin/ipaddr'
EXTERNAL_PORT = 54321
```

```
INTERNAL_PORT = 8080
try:
  while True:
    boton = GPIO.input(23)
    if boton == False:
      threading.Thread(target=Backup).start()
      print("Timbre Activado")
      time.sleep(5)
      os.system("telegram-send 'Hey ... estan timbrando ...'")
      time.sleep(5)
      #subprocess.call(['sudo','systemctl','start','mjpg_streamer.service'])
      threading.Thread(target=StartMjpgService).start()
      subprocess.call([pf,str(EXTERNAL_PORT),str(INTERNAL_PORT)])
      out = subprocess.check_output([ipaddr])
      ip = dict([line.split('=') for line in out.decode('ascii').strip().split('\n')])
      #reply = 'http://%s:%d/?action=stream' % (ip['External'],EXTERNAL_PORT)
      reply = 'http://%s:%d/?action=stream' % (ip['Internal'],INTERNAL_PORT)
      #reply = 'http://192.168.43.29:8080/?action=stream'
      os.system('telegram-send '+ reply)
      #reply = 'http://%s:%d/?action=stream' % (ip['External'],EXTERNAL_PORT)
      #if ip['External'] != ip['Public']:
      #   reply += '\n No es posible realizar streaming de manera Externa'

      #os.system('telegram-send '+ reply)
      time.sleep(40) #100
```

```
        os.system("telegram-send 'Comenzando a finalizar Streaming'")
        #subprocess.call([pf,'delete',str(EXTERNAL_PORT)])
        StopMjpgService()
        #subprocess.call(['sudo','systemctl','stop','mjpg_streamer.service'])
        os.system("telegram-send 'Streaming Finalizado'")
        pass
except:
   GPIO.cleanup()
```

Anexo 3 TELNET.PY

```
#Librerias
import threading
import sys,os
import telnetlib
import errno
from socket import error as socket_error
#Declaracion de variable que contiene la Ip a la cual se le realiza telnet
#HOST = "172.71.10.23"
HOST = "localhost"
#puerto = sys.argv[2]
#print(puerto)
#Definicion de funcion
def scanner():
   print("Ejecucion de Scanner en progreso")
```

```
   os.system("/usr/bin/python
/home/pi/Desktop/IBeaconIoT_TelnetTool/beacontools/examples/scanner_beacon.py")
   print("Escaneo Finalizado")
def validarTelnet():
#   global puerto
   try: #Validacion en caso de error
      #tn = telnetlib.Telnet(HOST,8080)
      while True:
#         print("haciendo telnet")
         tn = telnetlib.Telnet(HOST,8080) #8080)
#         puerto = 8080
         print("Connect  ",HOST,":",8080) #imprimir connect en caso exitoso
         scanner()
         break
   except socket_error as serr: #control de error
      if serr.errno != errno.ECONNREFUSED:
         raise serr


#while True: #ciclo para que el programa se ejecute siempre
#threading.Thread(target=validarTelnet).start() #ejecutar hilo de validacion de telnet
while True:
   validarTelnet()
   pass
#pass # continar ciclo hasta interrupcion de teclado o servicio por parte del usuario
```

Anexo 4 SCANNER_BEACON.PY

```
import time
import webbrowser
import sys
import os
import subprocess
from beacontools import BeaconScanner, IBeaconFilter
import threading, commands


rangeMax = -40
def openBrowserCam():
   #os.system("/usr/bin/python
/home/pi/Desktop/IBeaconIoT_TelnetTool/beacontools/examples/openBrowser.py")
   #os.system("sudo su - pi -c '/usr/bin/python
/home/pi/Desktop/IBeaconIoT_TelnetTool/beacontools/examples/openBrowser.py'")
   os.system("/bin/su - pi -c '/usr/bin/python
/home/pi/Desktop/IBeaconIoT_TelnetTool/beacontools/examples/openBrowser.py'")
def openBrowserBlack():
    os.system("sudo chromium-browser -no-sandbox
file:///home/pi/Desktop/prueba2/beacontools/examples/index.html")


def closeBrowser():
    #os.system("sudo killall chromium-browser")
    os.system("/usr/bin/killall chromium-browse")
```

```
def callback(bt_addr, rssi, packet, additional_info):
    print("aqui, <%d>" %(rssi))
    if rssi > rangeMax:
        print("aqui, <%d>" %(rssi))
        output = commands.getoutput('ps -A')
        print('Ejecucion Hilo Navegador')
        threading.Thread(target=openBrowserCam).start()
        time.sleep(30) #del
        closeBrowser()
    else:
        output = commands.getoutput('ps -A')
        if 'chromium-browse' in output:
            closeBrowser()


# scan for all iBeacon advertisements from beacons with the specified uuid
print("Iniciando Hilo de Scanner")
scanner = BeaconScanner(callback,
    device_filter=IBeaconFilter(uuid="00000003-0000-0000-0000-000000000000"))


scanner.start()
print(threading.current_thread())
time.sleep(30)
print("Stop externo")
scanner.stop()
sys.exit(0)
```

Anexo 5 OPENBROWSER.PY

```
import os,time

print("Abriendo Navegador")

#os.system("/usr/bin/chromium-browser -no-

sandbox  http://172.71.10.23:8080/?action=stream")

os.system("/usr/bin/chromium-browser -no-sandbox

http://192.168.43.29:8080/?action=stream")

#os.system("su - pi -c '/usr/bin/chromium-browser -no-sandbox

http://192.168.43.29:8080/?action=stream'")


print("Delay fin python navegador")

time.sleep(30)
```