INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

“SMART INTERFON”

SOLUCION ESTRATÉGICA EMPRESARIAL

Que para obtener el grado de MAESTRO EN SISTEMAS EMBEBIDOS

Presenta:

Juan Carlos López Garcia

Asesor:

Dr. Victor Miguel Hernández Maldonado

Aguascalientes Aguascalientes. Enero 2020

INFOTEC
POSGRADOS

AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA

MAESTRÍA EN SISTEMAS EMBEBIDOS

Ciudad de México, 13 de noviembre de 2020.
INFOTEC-DAIC-GCH-SE-0612/2020.

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado

SMART INTERFON

Desarrollado por el alumno **Juan Carlos López Garcia** y bajo la asesoría del **Dr. Victor Miguel Hernández Maldonado**; cumple con el formato de biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

Mtra. Julieta Alcibar Hermosillo
Coordinadora de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p.p Servicios Escolares

Agradecimientos

Quiero dedicar la culminación de este trabajo a mi familia principalmente a mi madre Ana María que siempre ha sido para mí un ejemplo de entereza, inteligencia y determinación quien este año 2020 ha tenido un problema de salud muy delicado, pero con la misma determinación de siempre lo ha afrontado todo en la vida está saliendo adelante. A mi padre Juan Lopez siempre un ejemplo de entrega y trabajo desinteresado a favor de la familia y de mi hermano Fernando que siempre ha mostrado una gran inteligencia sin embargo en esta situación difícil ha demostrado un corazón aún más grande.

También agradecer a todas las personas que nos han ayudado en esta situación de delicada salud de mi madre, desde un mensaje de ayuda y buenos deseos pasando por quien condujo desde otra ciudad para que un medicamento nos pudiera llegar a tiempo, hasta quien organizo una rifa para ayudarnos con recursos para seguir con los tratamientos. Una mención especial a los donadores de sangre quienes tuvieron que pasar incomodidades tales como ayuno, despertar temprano y recibir pinchazos en los brazos para hacer la donación.

A todos ellos he decidido hacerles esta pequeña dedicatoria como agradecimiento por el apoyo recibido.

Tabla de contenido

Introducción *1*
Capítulo 1. Contexto *4*
1.1 Aportación del proyecto 6
1.2 Contexto tecnológico 7
1.3 Especificación del problema 10
1.4 Especificación del problema tecnológico 11
Capítulo 2. Análisis y Diseño *15*
2.1 Método 15
2.2 Análisis y diseño de la primera iteración (iteración 1) 18
2.3 Análisis de riesgos 20
2.4 Análisis de casos de uso 21
Capítulo 3. Desarrollo y validación *36*
3.1 Display 36
3.2 Notificaciones 37
3.3 Comunicación 42
3.4 Captura de imagen en el dispositivo 44
3.5 Envió de imagen desde el dispositivo hacia el cliente android 45
3.6 Validación 46
Capítulo 4. Impacto 51
4.1 Impacto social 51
4.2 Impacto economico 51
Capítulo 5. Trabajo Futuro *55*
Conclusiones *59*
Bibliografia. *61*
Anexos *64*

Índice de figuras

Figura 1. Diferentes dimensiones de una s*mart city*…..……8
Figura 2. Esquema de metodología……..16
Figura 3. Proceso de desarrollo de un prototipo……..……17
Figura 4. Casos de uso……..21
Figura 5. Diagrama de secuencia persona en la puerta……..22
Figura 6. Diagrama secuencia comunicación SE a cliente android……...23
Figura 7. Diagrama secuencia comunicación cliente android a SE……..…24
Figura 8. Arquitectura de la solución red de área local (LAN)…….27
Figura 9. Arquitectura de hardware del SE…….....…28
Figura 10. Diagrama software del SE……..……...30
Figura 11. Diagrama de clases dispositivo para casa habitación……..……32
Figura 12. Pantalla login……..…….33
Figura 13. Pantalla alguien llamando a la puerta……..……33
Figura 14. Pantalla llamada en curso……....34
Figura 15. Pantalla LED 16x2……..……36
Figura 16. Conectividad pantalla LED con el convertidor I2C ……..…….37
Figura 17. Diagrama general de funcionamiento MQTT……..……...38
Figura 18. Diagrama de funcionamiento MQTT……..……..38
Figura 19. Verificación de estado del servicio mosquitto raspberryPi……..39
Figura 20. Cliente android recibiendo mensaje MQTT…….....40
Figura 21. Estructura de la tabla para gestión de clientes SE……..…..41
Figura 22. Datos de ejemplo de la tabla de BDD……..…41
Figura 23. Esquema de servidor de voz……..…42
Figura 24. Detección de rostro con Python…….……..45
Figura 25. Diagrama de comunicación entre componentes……46
Figura 26. Implementación física del dispositivo……49
Figura 27. Arquitectura para la solución utilizando conectividad 3g/4g…….55
Figura 28. Arquitectura microservicios en la nube……..……..…56

Índice de cuadros

Cuadro 1. Listado de dispositivos en el mercado……5
Cuadro 2. Listado de requerimientos funcionales……..…..20
Cuadro 3. Listado de requerimientos no funcionales……..…...20
Cuadro 4. Listado de casos de uso……...27
Cuadro 5. Lista de componentes y especificaciones ……....29
Cuadro 6. Validación de requerimientos funcionales……48
Cuadro 7. Validación de requerimientos no funcionales……..……49
Cuadro 8. Análisis de costos impacto económico……..……....52
Cuadro 9. Análisis de costos para la solución……52
Cuadro 10. Comparativa de costos……..52

Siglas y abreviaturas

3G: Tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS.

4G: Cuarta generación de telefonía móvil.

5G: Última generación de telefonía móvil.

Cloud Service: cualquier servicio disponible para los usuarios vía internet disponible mediante demanda.

FTP: File Transfer Protocol.

IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingeniería Eléctrica y Electrónica).

IoT: Internet of Things (Internet de las Cosas).

I2C: Protocolo de comunicación a través de bus de datos.

LAN: Local Área Network (Red de Área Local).

MQTT: Message Queuing Telemetry Transport

RTP: Real Time Transfer Protocol.

SE: Sistema Embebido.

SO: Sistema Operativo.

SQL: Structured Query Language.

SFTP: Secured File Transfer Protocol.

UDP: User datagram protocol.

UML: Unified Modeling Language

VAP: Advertisement Protocol(VLAN)

WAN: Wide Area Network (red de área amplia).

WiFi: Wireless Fidelity (Fidelidad inalámbrica).

XTP: Xpress Transport Protocol.

Introducción

Este proyecto busca ser el primer paso para el inicio de una unidad de consolidación de información lógica denominada *smart neiborhood*. Debido a la naturaleza del proyecto se pretende administrar todo lo relativo a un fraccionamiento privado. Para esto se plantea iniciar con el control de acceso a un coto privado mediante una plataforma cuyos principales componentes son un sistema embebido (SE) implementado sobre la plataforma raspberryPi además un cliente dentro de un dispositivo móvil implementado en android. En esta primera etapa toda la funcionalidad estará dentro del SE, es decir enviar y recibir voz, gestión de usuarios, notificaciones a todos los clientes, envió de la imagen de la persona que está llamando a la puerta etc. El desarrollo de toda la funcionalidad se plantea en la sección de desarrollo dentro de este documento, dicha funcionalidad en posteriores iteraciones de desarrollo se va a mudar a servicios cloud. Además, esta funcionalidad se planea que se amplié para tareas más robustas tales como recopilación de información como huellas dactilares, uso de reconocimiento facial, etc. Esta ampliación de funcionalidad se muestra en la sección de trabajo futuro dentro de este documento.

En la posterior sección de introducción dentro de este documento tenemos un amplio compendio de todas las diferentes unidades lógicas y conceptos que plantean diferentes autores para el uso de tecnologías IoT (Dodson, Sean, 2003), 3g, 4g, cloud services (Mell & Grance, 2011), tales como *smart city*, *smart environments*, etc. Se tomarán algunos de estos conceptos para este proyecto.

En el mercado existen algunos dispositivos para casa habitación los cuales buscan crear un ecosistema con la tendencia s*mart home* (Bing et al 2011) estos dispositivos planean la misma mejora que nosotros plantearemos en el desarrollo de esta solución con el enfoque de enviar la documentación inalámbricamente a una App instalada en un *smart phone.*

En la actualidad se tiene una amplia gama de líneas de desarrollo relacionado con *smart environments,* la idea de estos ambientes es recolectar información de un ambiente común para analizarla con el fin de mejorar las condiciones generales

tales como monitoreo de salud o asistencia a los individuos (Geetika et al., 2010). En esta propuesta se busca hacer de un fraccionamiento privado un espacio que cumpla con este tipo de requerimientos.

En cuanto a SE se refiere existen algunas variedades de un interfon para acceso y comunicación a la entrada a los condominios y/o edificios, el costo de estos dispositivos de comunicación es elevado debido al hardware y cableado por la extensa área que se debe de cubrir a través de la vivienda.

Capítulo 1

Contexto

Capítulo 1. Contexto

Actualmente se vive un grave problema de inseguridad en México; se estima que durante 2018 uno de cada tres mexicanos ha sido víctima de un delito[1] , entre los delitos más comunes el robo tipo asalto y robo a casa habitación. Algunos estudios señalan que la video vigilancia genera un efecto de desplazamiento en los actos delictivos[2]. Debido a estas razones la tendencia en cuanto desarrollo de vivienda en el país está enfocada en fraccionamiento privado. En diferentes regiones del país hay una gran tendencia a vivir en condominios privados[3] [4] [5] además que se estima que en ciudades como CDMX aproximadamente 5% de los mexicanos viven en departamentos[6] esto da como resultado un área de oportunidad para implementar soluciones de acceso. Es decir, vender el sistema embebido (SE) con su respectiva plataforma para dar el servicio de control de acceso a visitantes, así como la gestión y comunicación hacia adentro del condominio.

Las estadísticas[7] indican que alrededor del 80% de la población en México posee un *smartphone*, para el segmento de población a la cual está enfocada esta propuesta el porcentaje de personas que poseen esta tecnología es cercano al 100%. Se pretende aprovechar esta ventaja con la integración de nuevas tecnologías y brindar un servicio a los fraccionamientos privados.

---

[1] Encuesta Nacional de Victimización y Percepción sobre Seguridad Pública (ENVIPE) 2018, INEGI, Consultado 10 Nov 2019, https://www.inegi.org.mx/programas/envipe/2018/

[2]Seguridad con videovigilancia, UNAM, Consultado 10 Nov 2019, http://ciencia.unam.mx/contenido/infografia/51/seguridad-con-videovigilancia-

[3]Jesús Ángel Enríquez Acosta, Entre el miedo y la distinción. El estado actual del fraccionamiento cerrado en las ciudades fronterizas de Tijuana, Nogales y Ciudad Juárez, Scileo, Consultado 15 Nov 2019, http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-69612007000100001

[4]Informe del Mercado Inmobiliario 2018, LAMUDI, 2018, Consultado 10 Nov 2019 https://www.lamudi.com.mx/Informe-del-Mercado-Inmobiliario-2018/

[5]Ante la inseguridad los potosinos prefieren vivir en condominios, EL UNIVERSAL, Consultado 10 Ene 2018, https://sanluis.eluniversal.com.mx/metropoli/29-10-2018/ante-inseguridad-potosinos-prefieren-vivir-en-privadas-o-condominios

[6]¿Cuántos habitantes y hogares hay en México?, dineroenimagen, 2015, Consultado 10 Nov 2019, https://www.dineroenimagen.com/2015-07-29/59247

[7]Penetración de smartphones no se detiene, el eleconomista, Feb 2018, Consultado 10 Nov 2019, https://www.eleconomista.com.mx/empresas/Penetracion-de-smartphones-no-se-detiene-20180221-0022.html

Vivimos en una etapa donde existe la tecnología necesaria para poder desarrollar un dispositivo con su respectiva plataforma para comunicación entre condóminos y visitantes en la entrada de un fraccionamiento. Así que por ello resulta viable la propuesta que en este trabajo se presenta.

En el Cuadro 1 se muestran algunas de las opciones de hardware que existen además de sus principales características. Al momento de hacer esta comparativa solo había una opción para red LAN.

Durante este documento se explicará la visión que se tiene de esta solución buscando convertirse en una plataforma mucho más completa que solo controlar el acceso a un fraccionamiento privado y la interacción de los condóminos con los visitantes de un fraccionamiento.

Imagen	Producto	Marca	Precio	URL	RED	Comunicación a smartphone	Vision nocturna	Contro de perifericos	Resolucion de imagen	Megapixeles
	Videoportero DS-KV8102-IP	Hikvision	$2199 + Envio	https://www.abasteo.mx/index.php?cl=details&anid=1f9347b78c511f782dabde9fef7d3558&gclid=CjwKCAjw6IfoBRCiARIsAF6q06uiihKgRTzMM6MZm_odrOu_26jwpqtLIGxJnxJksFPvvrMzGazpwQMaAmzpEALw_wcB	Ethernet	No	Si	Si	1280 x 720	1MP
	Frente de Calle IP DS-KV8102-IM, Lector de Tarjetas, Altavoz, Aluminio	Hikvision	$1,959	https://www.abasteo.mx/index.php?cl=details&anid=1f9347b78c511f782dabde9fef7d3558&gclid=CjwKCAjw6IfoBRCiARIsAF6q06uiihKgRTzMM6MZm_odrOu_26jwpqtLIGxJnxJksFP	Ethernet	Si	Si	NE	1280 x 720	1.3MP
	Videoportero VTK-VTO6210B-VTH1560B, Altavoz/Micrófono, Interior/Exterior, Negro	Dahua	$5,729	https://www.abasteo.mx/Dahua-Videoportero-VTK-VTO6210B-VTH1560B-Altavoz-Microfono-Interior-Exterior-Negro.html?nosto=shop_api_detail2_1	Ethernet	No	Si	Si	1280 x 720	1MP
	Videoportero exteror IP para edificios	JR	$6,160	https://articulo.mercadolibre.com.mx/MLM-550002742-frente-de-calle-conexion-ip-wifi-a-smart-o-tablet-abre-chapa-_JM?matt_tool=18293428&matt_word&gclid=CjwKCAjwlZf3BRABEiwA8Q0q	Wifi	Si	No	Si	1281 x 720	1MP
	Portero exteror IP para edificios solo audio	JR	$9,500	https://articulo.mercadolibre.com.mx/MLM-626344831-portero-gsm-para-conjuntos-edificio-de-200-propietarios-y-1000-usuarios-control-de-acceso-para-tarjetas-rfid-o-llaveros-_JM?quantity=1&variation=32248746432	GSM	Si (llamada GSM)	No	Si	NA	NA

Cuadro 1. Listado de dispositivos en el mercado. Fuente: Elaboración propia.

1.1 Aportación del proyecto

Este proyecto de titulación de maestría está enfocado en solución estratégica no es tanto una investigación académica por lo tanto las aportaciones están más encaminadas a los resultados del prototipo. En general las aportaciones más relevantes son las relativas con la comunicación de los diferentes componentes de la plataforma. A continuación, se enumeran lo más relevante que se obtuvo:

La plataforma comercial raspberryPi demostró tener la suficiente potencia de hardware para soportar los múltiples procesos necesarios para la plataforma tales como librerías de audio, software de captura, base de datos embebida, comunicación bidireccional.

El SO Raspbian propio de la plataforma raspberryPi mostro compatibilidad con todo lo que decidimos instalar.

La librería *mosquitto* funciono para el envío de mensajes tipo *MQTT* entre componentes desarrollados entre diferentes tecnologías tales como android y Linux (Raspbian).

La interacción de componentes desarrollados en python y adroid funciono correctamente para el envió y recepción del *stream* de audio respectivamente.

Implementación dentro del cliente android de la librería mqtt para recibir notificaciones desde la raspberryPi.

El uso de la librería gst para la detección y configuración de los periféricos de audio dentro de la raspberryPi tales como micrófono y bocinas fue satisfactoria.

El uso de la librería VLC para utilizar la raspberryPi como servidor de voz RTP.

La implementación de la librería openCV para detección facial, este componente se desarrolló en python para centrar la imagen en el rostro de la persona que está utilizando el SE.

Implementación de las librerias RPi y smbus para el manejo de pantalla LCD 16x2.

Uso del convertidor I2C para optimizar el manejo del display LCD.

Todos los detalles de los puntos anteriores están en las secciones de *desarrollo y validación* además de en *conclusiones* dentro de este documento.

1.2 Contexto tecnológico

Fundamentalmente en este trabajo se propone un avance en la generación de una unidad lógica que se integre más fácilmente a una *smart city* como se plantea en los textos consultados (Su K. et al. 2011), (Krylovskiy et al. 2015) además de los diferentes tipos de tecnologías implementados en ella (Mitton et al., 2012). Actualmente en algunas publicaciones se habla de *smart home* (Wang et al. 2013), *smart living* (Lewis, 2008) y *smart environments* (Diane et al. 2007). Con respecto a *smart home* ya se tiene una línea de desarrollo muy adelantada referente a IoT para el uso cotidiano en una casa como plantea (Bing et al. 2011). *smart living* se refiere a la incorporación de dispositivos electrónicos para las actividades cotidianas en la vida además de la interacción del hombre con dichos dispositivos (Lewis, 2008).

De acuerdo con (Dameri et al. 2014) las características principales del *smart city* son:

- Producción y optimización de energía
- Bajar las emisiones de CO2
- Eficiencia en las construcciones
- Calidad de transporte
- Incrementar la calidad de los servicios.

Alineado al último punto este proyecto propone mejorar la calidad del servicio de comunicación entre las personas externas que llegan a tratar algún asunto al fraccionamiento tales como venta de productos en general o servicios básicos como gas, agua, etc. Y los condóminos dentro del fraccionamiento.

Además, otra de las bondades es poder comunicarse con las personas que usualmente prestan un servicio en el fraccionamiento ya sea trabajadores, jardineros, etc.

En relación con una *smart city* (Dameri et al. 2014) enumera las siguientes dimensiones, las cuales se recomienda considerar (Figura1):

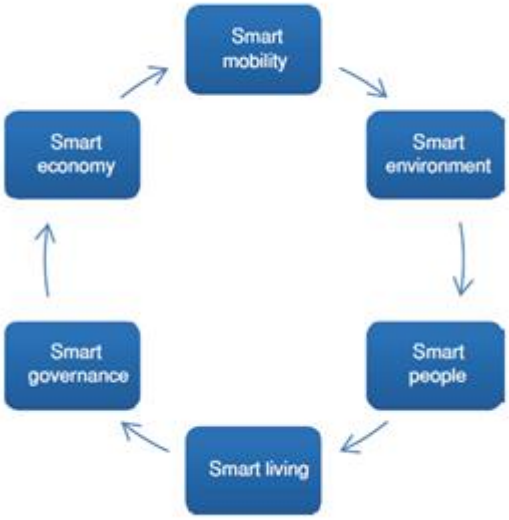


Figura 1. Diferentes dimensiones de una smart city

FUENTE: Dameri et al. 2014

En la Figura 1 se muestran los conceptos que una *smart city* busca promover tales como:

Smart Economy.

Relativo a la mejora en la competitividad con innovación. Este modelo se basa en conceptos para impulsar el desarrollo, la sostenibilidad y el atractivo para nuevas inversiones, los principales son: e-business, e-commerce, incremento de la productividad, empleo e innovación en TI y generación de servicios y nuevos productos, nuevos modelos y oportunidades de negocio y emprendimiento. Donde los puntos más importantes a tratar son los siguientes[8].

Apertura a la transformación

Innovación continúa

Internacionalización

Productividad

[8]Economía Smart, Universidad de Alicante, 2017, Consultado 10 Nov 2019, https://web.ua.es/es/smart/smart-economy-economia-inteligente.html

Smart People

Se refiere al desarrollo de capital humano y social por medio de participación comunitaria, utilizando como herramienta principal la tecnología, es decir teléfonos inteligentes, tabletas, internet, etc. Enfocado principalmente en las relaciones entre los ciudadanos y las ciudades comienzan a ir más allá de realizar consultas y trámites con la administración, consulta del estado del tráfico, del estado de la meteorología o de la consulta de callejeros[9]. Sin duda la tecnología proporciona la apertura de nuevos conceptos, como las s*mart city* y s*mart university,* se enfoca en:

Participación en la vida publica

Flexibilidad

Creatividad

Smart Goverment

Busca la participación activa a través de medios digitales entre gobierno y ciudadanos con el fin de facilitar tramites, mejorar servicios para mejorar la calidad de vida[10]. Cuyos principales objetivos son:

Participación en la toma de decisiones.

Mejora de servicios públicos y sociales.

Transparencia.

Smart Mobility

Enfocado principalmente al transporte de los ciudadanos. Los objetivos principales que deben plantearse para una movilidad inteligente se corresponden a la promoción de una movilidad sostenible, la cual garantice que la accesibilidad, los sistemas de transporte, los problemas ambientales y la gestión del aparcamiento respondan a las necesidades económicas, sociales y medioambientales de la ciudad[11]. Sus aspectos principales son:

Accesibilidad.

Disponibilidad.

---

[9] Smart People: comunidad Senspeople, Universidad de Alicante, 2017, Consultado 10 Nov 2019, https://web.ua.es/es/smart/smart-people-comunidad-senspeople.html

[10] Smart Government: Gobernanza del Futuro, Marcombo, 2017, Consultado 10 Nov 2019, http://www.librosmartcity.com/

[11]Smart Mobility, Planes Integrales de Movilidad, 2012, Consultado 10 Nov 2019, https://issuu.com/panoptica/docs/planes-integrales-de-movilidad-line

Reducir el impacto ambiental.

Innovación y seguridad en el sistema de transporte.

Sustentabilidad.

Smart Environment

Se enfoca en mejorar cualquier ambiente por medio de tecnología mediante el uso de sensores para obtener información y tomar decisiones operativas. Se busca la capacidad de optimizar los recursos naturales, preservar y proteger el medio ambiente, reducir los gases y residuos de manera sostenible, y de controlar y racionalizar el consumo de energía[12]. Busca principalmente:

Optimización de recursos.

Sustentabilidad del manejo de recursos.

Smart Living

Mejora en la calidad de vida mejorando los procesos del día a día mediante el uso de tecnología. Hace referencia a los nuevos estilos de vida mediante las TI, el comportamiento y el consumo. Los servicios inteligentes *smart living* actúan en los ámbitos de: salud, seguridad ciudadana, cultura, domótica en viviendas, proporcionando servicios inteligentes como e-salud, e-accesibilidad y e-turismo, todo ello con el objetivo de incrementar los niveles cohesión social, el capital y la seguridad en las urbes[13]. Sus principales aspectos son:

Calidad en la vivienda

Atractivos turísticos

Seguridad

1.3 Especificación del problema

Para la atención a los visitantes que llegan a la entrada de todo fraccionamiento privado se recomienda el uso de un interfon o personal de seguridad privada. En caso de contar con seguridad privada implica un gasto mensual en una o varias

[12]Smart Environment: Un entorno de calidad de vida, Universidad de Alicante, 2019, Consultado 10 Dic 2019, https://web.ua.es/es/smart/smart-environment-un-entorno-de-calidad-de-vida.html
[13] Smart Living: Microentorno de calidad en Smart Cities, Marcombo, 2016, Consultado 10 Dic 2019, http://www.librosmartcity.com/

personas durante 24 horas al día. Incluso esta opción implica cuestiones de seguridad tales como el tener que facilitar una identificación a la persona de la entrada.

Para el caso de que no se tenga personal de vigilancia muchas veces los fraccionamientos privados optan por que los mismos habitantes manejen el acceso. Esto implica descontrol en el fraccionamiento debido a que no queda ningún registro de las personas que ingresan. Además, para este caso también conlleva cuestiones de seguridad, debido que para poder darle acceso a personas que presten servicios o lleven mercancías se necesita que faciliten su teléfono celular.

1.4 Especificación del problema tecnológico

En relación con el problema tecnológico de esta primera etapa del proyecto que esta basado en la construcción de lo que tradicionalmente se conoce como *interfon*, aunque debido a los avances tecnológicos actualmente ya está más vigente el concepto de *video portero*.

El primer elemento de la solución tecnológica es el dispositivo, el interfon clásico que existen actualmente en el mercado se comunican por medio de señal analógica mediante cable. En este caso de nuestra propuesta se decidió implementar un sistema embebido utilizando la plataforma raspberryPi, en la sección *2.6.1 Arquitectura hardware dispositivo* se muestran los detalles de la implementación.

El segundo punto que se debe tener en cuenta la configuración de red sobre la cual va a desenvolverse el dispositivo. Actualmente las opciones de configuración de red son LAN o uso de cloud services esta opcion es similar a lo que tradicionalmente se llama topología WAN (Peterson & Davie, 2003) con la ventaja de que es mucho mas practica, cada opción esta enfocada en diferentes escenarios. En la sección *2.6 Arquitectura general* de la solución para red de área local en este documento se muestra la implementación sobre una red LAN y en la sección de trabajo futuro se plantea la solución utilizando cloud services. La diferencia principal funcionalmente hablando de ambas implementaciones es la cobertura del alcance de la plataforma, es decir para la red LAN solo se podrá comunicar el interfon con

clientes que estén dentro del rango físico de esta red (Nader F. Mir, 2006). Para la opción de cloud services el rango se amplía, para comunicarse solo se necesita que el interfon y el cliente tengan servicio de internet. Cabe destacar que la mayoría de las soluciones comercialmente hablando en el mercado trabajan sobre red LAN (tabla 1). Otra limitante en el mercado es la implantación de la red LAN cableada, esto eleva mucho el costo de la implementación, por esta razón hay pocos cotos privados que cuentan con esta implementación.

1.5 Objetivos

A continuación, se plantean los objetivos del proyecto tanto generales como particulares.

En primer lugar, se busca el desarrollo de un dispositivo para favorecer la comunicación por medio de voz, entre una persona que se encuentre en la entrada de un fraccionamiento privado con una persona dentro de su vivienda, esto se plantea a través una aplicación (cliente) instalada en el *smartphone*, además dicha aplicación recibirá una imagen de la persona con la cual se está comunicando en la puerta.

Tal y como se menciona en la introducción este proyecto está diseñado en ser solo la primera etapa de construcción para una plataforma completa dentro de fraccionamientos privados para la posterior integración de más servicios digitales implementados mediante la ayuda de *cloud services* tales como activación de puertas, guardar historial de visitas, guardar datos de las personas que visitaron el fraccionamiento, pudiéndose denominar esto como *smart neighborhood*. Esto a su vez está planeado para ser una pieza de un sistema más grande del cual ya se mencionó en la introducción tal como *smart city*.

Además, otro enfoque de este proyecto es con muy pocos cambios se posibilita a dar el servicio para una casa habitación, adaptando la funcionalidad de tocar el timbre y añadiendo el valor agregado que las tecnologías smart puede agregar.

1.5.1 Objetivos específicos

- Implementación de un sistema embebido (SE) como servidor de comunicaciones utilizando como plataforma una raspberryPi.
- Implementación de todo el software y configuraciones de la raspberryPi.
- Desarrollo de un prototipo cliente android para la comunicación con el SE

Capítulo 2

Análisis y Diseño

Capítulo 2. Análisis y Diseño

En este capítulo se va a elaborar el análisis y diseño a nivel técnico del proyecto. Primeramente, se define que metodología de desarrollo se va a utilizar y cuáles son los alcances, después se definen los requerimientos funcionales y no funcionales además de los diagramas de caso de uso y de secuencia los cuales nos ayudan a tener una idea más clara de cómo será la funcionalidad práctica para de aquí concretar clases y métodos con un enfoque ya en el desarrollo. Otro aspecto muy importante visto en este capítulo es la definición técnica de los componentes tanto de hardware como de software, para esto se elaboraron diagramas mostrando los componentes.

2.1 Método

La metodología utilizada en este proyecto es el modelo en *espiral* de *Bohem* (Sommerville, 2011), debido a su flexibilidad y que está basada en la construcción de prototipos e iteraciones, esto da la ventaja que al final de cada iteración se tiene un entregable que se adapta al desarrollo de la propuesta. La iteración tal como se define en el texto consultado (Sommerville, 2011), se define como la ocurrencia de cada una de las actividades necesarias para tener un entregable específico dentro de un tiempo determinado terminado.

En la Figura 2 se muestra la imagen de los pasos seguidos para el uso adecuado de esta metodología.

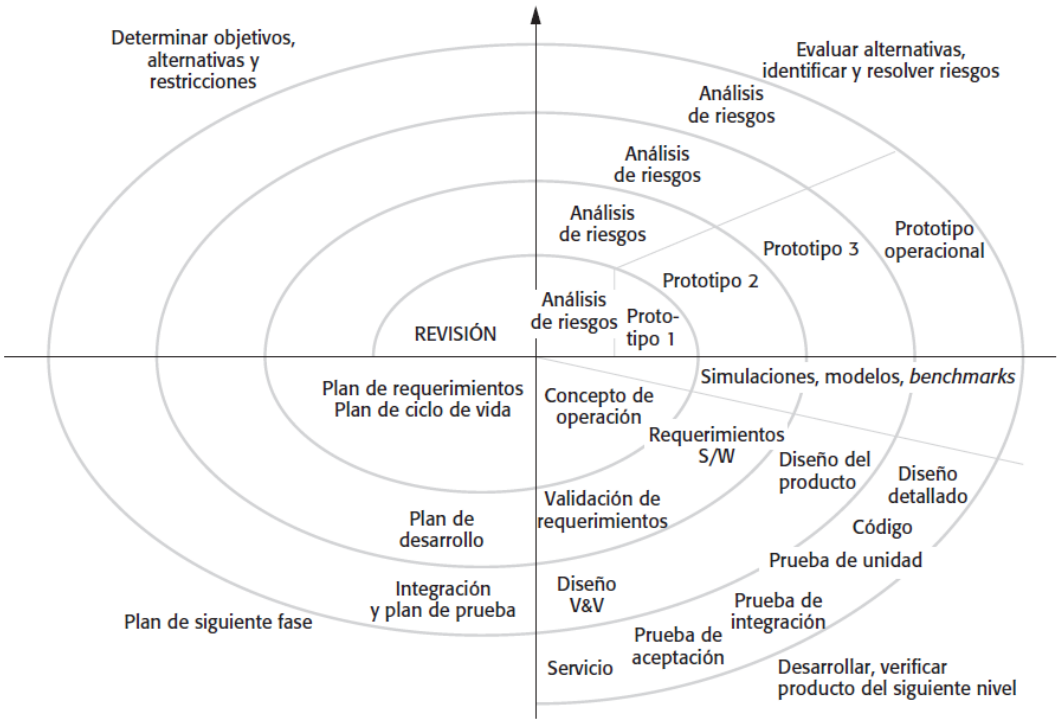


Figura 2 Esquema de metodología.

FUENTE: Sommerville, 2011 p48

En dicha referencia el autor plantea cuatro secciones para cada iteración las cuales son:

1. Establecimiento de objetivos: *Se definen objetivos específicos para dicha fase del proyecto. Se identifican restricciones en el proceso y el producto, y se traza un plan de gestión detallado. Se identifican los riesgos del proyecto. Pueden planearse estrategias alternativas, según sean los riesgos.*

2. Valoración y reducción del riesgo: *En cada uno de los riesgos identificados del proyecto, se realiza un análisis minucioso. Se dan acciones para reducir el riesgo. Por ejemplo, si existe un riesgo de que los requerimientos sean inadecuados, puede desarrollarse un sistema prototipo.*

3. Desarrollo y validación: *Después de una evaluación del riesgo, se elige un modelo de desarrollo para el sistema. Por ejemplo, la creación de prototipos desechables sería el mejor enfoque de desarrollo, si predominan los riesgos en la interfaz del usuario. Si la principal consideración son los riesgos de seguridad, el desarrollo con base en transformaciones formales sería el proceso más adecuado,*

entre otros. Si el principal riesgo identificado es la integración de subsistemas, el modelo en cascada sería el mejor modelo de desarrollo a utilizar.

4. Planeación: *El proyecto se revisa y se toma una decisión sobre si hay que continuar con otro ciclo de la espiral. Si se opta por continuar, se trazan los planes para la siguiente fase del proyecto.*

Como se menciona en la sección anterior (punto número tres) para prototipos es recomendable usar un enfoque de desarrollo el cual es nuestro deseo en esta primera iteración. En las siguientes iteraciones (ver sección de trabajo futuro) se va a tener integración de otros sistemas por lo cual se infiere que el mejor modelo sería el de cascada. Por lo tanto, en este trabajo se pretende establecer una sólida base en la cual el trabajo futuro pueda funcionar favorablemente.

Cada una de las secciones anteriores enlista a grandes rasgos lo que se tiene que considerar para terminar satisfactoriamente una iteración. A nivel más practico el autor plantea el siguiente diagrama para la creación de un prototipo basado en los pasos anteriores.

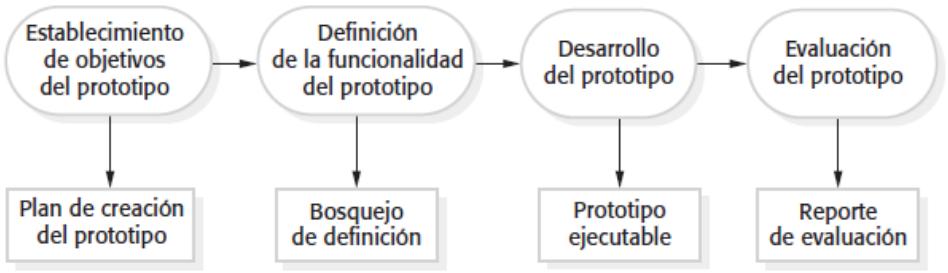


Figura 3. Proceso de desarrollo del prototipo.
FUENTE: Sommerville, 2011 p45

El alcance de este documento considera implementar un prototipo solo para la primera iteración enfocado en dar servicio para casa habitación, aunque un objetivo más amplio es llegar a la construcción de un proyecto para fraccionamientos privados, por esto mismo la etapa de *planeación* no está tomada en cuenta para este documento.

En la sección *2.2 Análisis y diseño* de la primera iteración comienza el proceso propuesto por la Figura 3, iniciando con el establecimiento de objetivos

posteriormente definiendo la funcionalidad mediante requerimientos y al final las etapas de desarrollo y resultados. Se muestran los detalles de iteraciones futuras.

Iteración 1

Creación de un prototipo funcional (Dispositivo y cliente android) para casa habitación que mediante red de área local (LAN) se comunique con las personas que están dentro de ella, en esta iteración de los entregables más destacados son: el cliente android, la comunicación mediante protocolo de transporte de voz RTP.

Iteración 2

Implementación de una plataforma para fraccionamiento privado utilizando una red LAN teniendo en cuenta el escenario de que en el fraccionamiento estuviera implementada esta topografía de red, el problema a resolver más destacado de esta iteración es seccionar a los usuarios para identificar a quienes se les tiene que enviar notificación debido a que viven en una misma casa

Iteración 3

Adaptación de la plataforma a servicios *cloud*, en esta iteración se van a trasladar varios de nuestros servicios a una plataforma en la nube esto dará como resultado una mejora directamente en cobertura del servicio debido a que no será necesario que todos los dispositivos estén conectados en la misma en red LAN para tener acceso a la plataforma. Creación de la App cliente en IOs.

2.2 Análisis y diseño de la primera iteración (iteración 1)

A continuación, vamos a iniciar el análisis y diseño de la primera iteración. Como específica la metodología se establecerán objetivos, además se efectúan los diferentes análisis tales como análisis de riesgos, análisis de casos de uso y diagramas de secuencia. Por último, se muestra las especificaciones de las diferentes arquitecturas (hardware, software, etc.) además de la arquitectura en general.

2.2.1 Establecimiento de objetivos específicos

A continuación, se enlistan los objetivos específicos para esta iteración, cada objetivo corresponde a un entregable y todos en su conjunto forman la plataforma funcional.

- Utilizar una rasberryPi para la construcción del interfon (sistema embebido).
- Instalar y evaluar el performance de un Linux embebido.
- Hacer todas las configuraciones pertinentes al sistema embebido relacionadas con la instalación de librerías, paqueterías y configuraciones.
- Crear un software en Android (cliente android) que sea capaz de recibir notificaciones por parte del SE además de entablar comunicación bidireccional.
- Diseñar e implementar la comunicación entre el SE y el cliente android.
- Pruebas de integración entre ambos componentes.

2.2.2 requerimientos funcionales

Como su nombre lo dice estos requerimientos son derivados de las funcionalidades que se necesitan para la plataforma. En el Cuadro 2 se enlistan los requerimientos funcionales.

Requerimiento ID	**Descripcion**
RF01	El sistema mandara señal de alerta a todos los teléfonos inscritos según corresponda.
RF02	El Interfon tendrá un buzzer simulando timbre.
RF03	El Interfon tendrá un led para verificar que este prendido.
RF04	El Interfon mandara stream de audio.
RF05	El Interfon recibira stream de audio.
RF06	El interfon capturara la imagen de la persona que está en la puerta.
RF07	El cliente Android mandara alerta cuando el inter fon está mandando señal.
RF08	El cliente Android tendrá un icono en el escritorio.

RF09	El cliente Android alertara como si fuera una llamada entrante.
RF09	El cliente Android mandara un stream de audio
RF10	El cliente android desplegara la imagen de la persona.
RF11	El cliente android guardara un historial.

Cuadro 2. Listado de requerimientos funcionales.

FUENTE: Elaboración Propia.

2.2.3 Requerimientos no funcionales

Estos requerimientos se desprenden de las necesidades que surgen para poder implementar las funcionalidades. En el Cuadro 3 se enlistan los requerimientos no funcionales.

Requerimiento ID	**Descripcion**
NF01	El cliente Android tendrá una actividad corriendo en background siempre.
NF02	El interfon tendrá Linux embebido.
NF03	La comunicación será mediante el protocolo RTP.
NF04	El Interfon manejara sus back Jobs corriendo siempre en Linux.
NF05	El interfon usara python para manejar perifericos tales como teclado y display
NF06	El interfon iniciara el stream de comunicación.
NF07	El interfon mandara notificacion a todos los clientes que corresponda utilizando el protocolo MQTT
NF08	El interfon tendra una BDD embebida para gestion de los clientes dados de alta
NF09	las condiciones de error se van a manejar conforme al Testing plan

Cuadro 3. Listado de requerimientos no funcionales.

FUENTE: Elaboración Propia.

2.3 Análisis de riesgos

En el modelo espiral de Bohem se plantea un análisis previo identificando los riegos potenciales. A continuación, presentamos los puntos de atención que se identificaron:

- La interferencia natural en la comunicación se debió a las condiciones físicas (paredes, estructuras de hierro/acero) etc.
- Incompatibilidad entre librerías de las diferentes plataformas.
- Bajo performance o mala calidad del audio.

Identificados los anteriores riesgos con la creación del prototipo y el uso del mismo bajo diferentes condiciones.

2.4 Análisis de casos de uso

Utilizando la metodología UML que también se plantea como parte del análisis en la metodología de espiral (Sommerville, 2011) se consultó más detalladamente en (Larman, 2003) se modela el siguiente caso de uso:

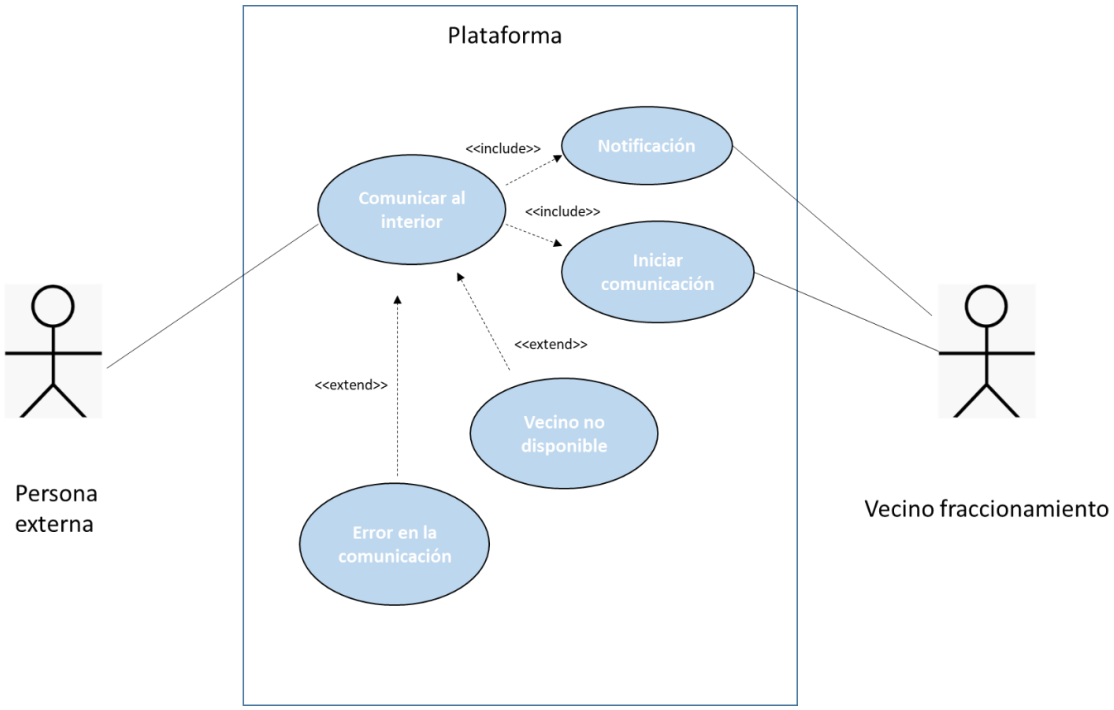


Figura 4. Casos de uso.

FUENTE: Elaboración Propia.

Como se observa en la Figura 4 tenemos dos actores principales para el uso del proyecto como lo estamos proponiendo los cuales son:

Persona externa: Cualquier persona que no vive en el fraccionamiento privado que desea ingresar o comunicarse con alguien del interior.

Vecino fraccionamiento: Todos los vecinos dentro del fraccionamiento que tienen instalado el cliente de android que tendrán la disponibilidad de comunicarse con la puerta de la entrada.

Con respecto a los casos de uso podemos ver que el *happy path* tiene dos diferentes acciones:

1.- Notificar a los clientes android: Posteriormente en el documento vamos a mostrar con detalle este proceso utilizando MQTT.

2.-Comunicación: explicaremos con detalle cómo se entabla la comunicación utilizando el protocolo RTP.

Además de esto tenemos los dos escenarios de error más común que se van a manejar los cuales son:

- Vecino no disponible.
- Error en la comunicación.

2.4.1 Análisis de diagramas de secuencia general

A partir del caso de uso se genera el siguiente diagrama de secuencia:

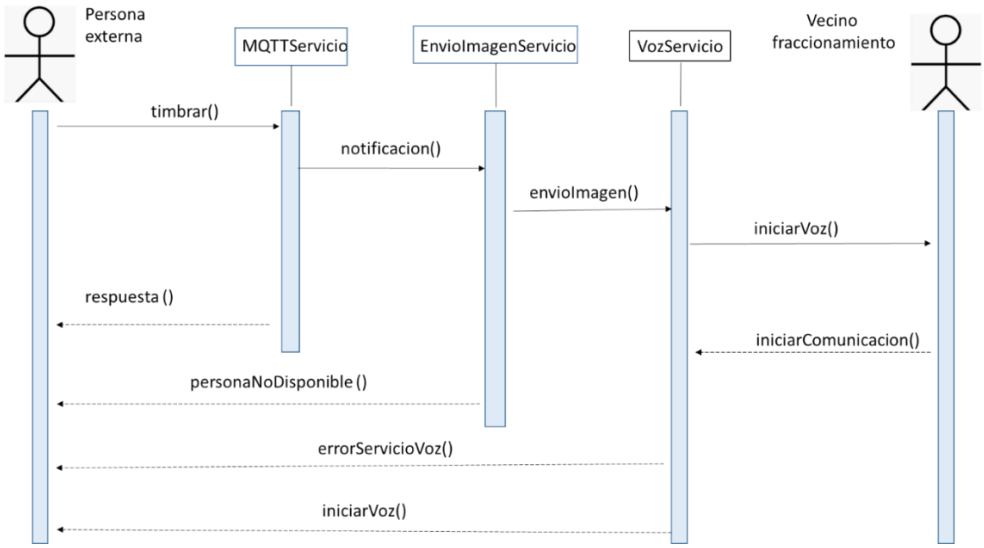


Figura 5. Diagrama de secuencia persona en la puerta.

FUENTE: Elaboración Propia.

En la sección anterior (Figura 5) diagrama de secuencia vemos la forma de interactuar de los diferentes componentes en orden lógico para el escenario de una persona llamando a la puerta, además de los escenarios de error más comunes

relacionados con este caso. A continuación, enlistamos cada servicio lógico propuesto y damos una breve explicación de cada uno de ellos.

MQTTServicio(): este servicio es el encargado de mandar las notificaciones a cada uno de los clientes además de recibir las respectivas respuestas, más adelante en el documento damos detalle de su implementación.

EnvioImagenServicio(): este servicio su tarea principal es capturar una imagen desde el SE y enviarla al cliente android, más adelante en este documento vamos a mostrar los detalles de la tecnología utilizada en ello.

VozServicio(): este servicio como se muestra en el diagrama será el encargado de enviar un *stream de voz* desde el SE hacia el cliente android además de recibir un *stream* desde el cliente y mandarlo a bocinas.

Además del diagrama de secuencia general se hace el diagrama de secuencia entre componentes en las próximas secciones mostramos con detalle cada uno de estos:

2.4.2 Diagramas de secuencia comunicación desde el SE hacia el cliente

En la Figura 6 se muestran las operaciones configuradas para la operación de comunicación del SE hacia el cliente android.

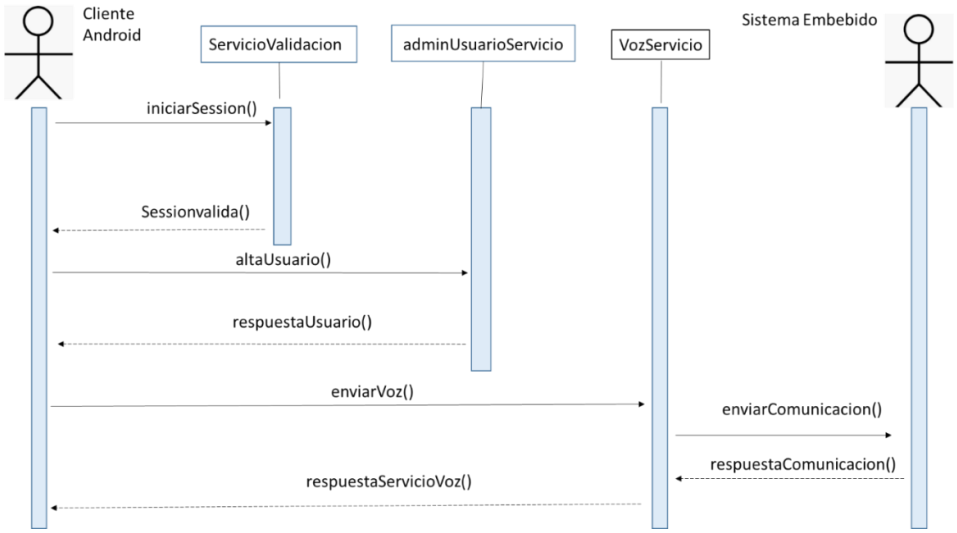


Figura 6. Diagrama secuencia comunicación SE a cliente android.

FUENTE: Elaboración Propia.

A continuación, se muestran los detalles de cada uno de los servicios listados en el diagrama:

usuarioValido(): manda el *request* al servicio de validación con unas credenciales (user/pass) y recibe una respuesta de true/false dependiendo si son válidas.

timbrar(): manda llamar al servicio de notificaciones para mandar un mensaje de notificación al cliente las respuestas posibles son (*éxito,error*)

iniciarVoz(): inicia el proceso de comunicación en caso de que el servicio no esté disponible mandara el código de error pertinente.

enviarComunicacion(): se inicia el *stream* de comunicación en dos vías.

2.4.3 Diagramas de secuencia comunicación desde el cliente hacia el SE

En la Figura 7 mostramos el diagrama de secuencia para el inicio de sesión desde el cliente android.

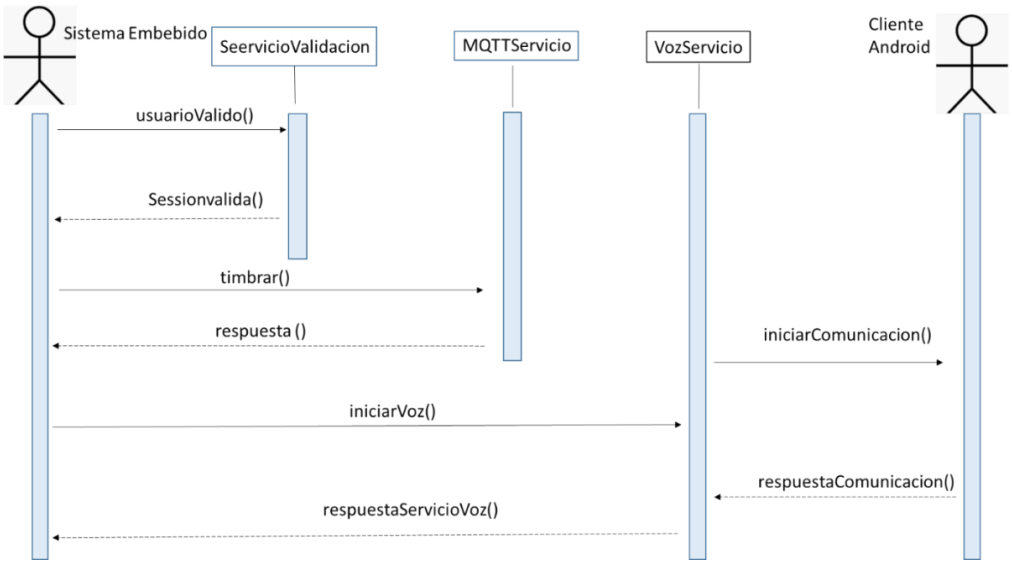


Figura 7. Diagrama de secuencia comunicación cliente android a SE.

FUENTE: Elaboración Propia.

A continuación, se explican las operaciones implementadas a ser accesibles desde este cliente:

*iniciarSession():*el cliente android tiene un usuario por defecto (admin/adminDoor123) con el cual se podrá acceder por primera vez a su vez desde este usuario.

altaUsuario(): una vez autenticados podremos dar de alta otros usuarios.
enviarVoz(): desde esta método se inicia el *stream* de comunicación.
iniciarComunicacion(): se refiere a la comunicación bidireccional es decir enviar y recibir desde el cliente android.

2.5 Plan de pruebas.

Con base en los escenarios en los casos de uso y además teniendo en cuenta condiciones generales de inicio comunes en todos los planes de pruebas se genera el plan de testing que se muestra en el Cuadro 4:

Título del caso
Happy path
Descripción
el SE y el cliente están en línea con conectividad y todas las condiciones optimas
Precondiciones
•La red LAN funciona correctamente. •el SE tiene conectividad. •el cliente android tiene conectividad. •dentro del SE están corriendo correctamente todos los servicios
Resultado esperado
Se debe entablar comunicación entre el SE y el cliente sin ningún problema
Título del caso
Sin red
Descripción
Sin conectividad de la red LAN
Precondiciones
•La red LAN esta caída.
Resultado esperado
el SE manda mensaje de error en la pantalla
Título del caso
cliente android no disponible
Descripción
ninguno de los integrantes de una familia subscritos a un tópico dado está disponible por la razón que sea
Precondiciones

•La red LAN funciona correctamente. •el SE tiene conectividad. •ningún cliente android perteneciente a ese tópico tiene conexión
Resultado esperado
el SE manda mensaje de error en la pantalla
Título del caso
servicio MQTT no disponible
Descripción
el servicio MQTT dentro del SE no está disponible
Precondiciones
•La red LAN funciona correctamente. •el SE tiene conectividad. •el servidor MQTT dentro del SE no está disponible
Resultado esperado
el SE manda mensaje de error en la pantalla
Título del caso
servicio RTP no disponible
Descripción
el servicio RTP dentro del SE no está disponible
Precondiciones
•La red LAN funciona correctamente. •el SE tiene conectividad. •el servidor RTP dentro del SE no está disponible
Resultado esperado
el SE manda mensaje de error en la pantalla
Título del caso
servicio FTP no disponible
Descripción
el servicio FTP dentro del SE no está disponible por lo cual no se podrá mandar la imagen
Precondiciones
•La red LAN funciona correctamente. •el SE tiene conectividad. •el cliente android tiene conectividad. •dentro del SE están corriendo correctamente todos los servicios
Resultado esperado
Se debe entablar comunicación entre el SE y el cliente sin ningún problema, la imagen se debe poner con la leyenda: *falla interna en la obtención de la imagen*

Cuadro 4. Listado de casos de uso.

FUENTE: Elaboración Propia.

2.6 Arquitectura general de la solución para red de área local

Para generar la comunicación entre los diferentes elementos dentro de la solución se tienen múltiples escenarios detectados, comunicación mediante red LAN (red de área local), 4g o cualquier red de telefonía celular.

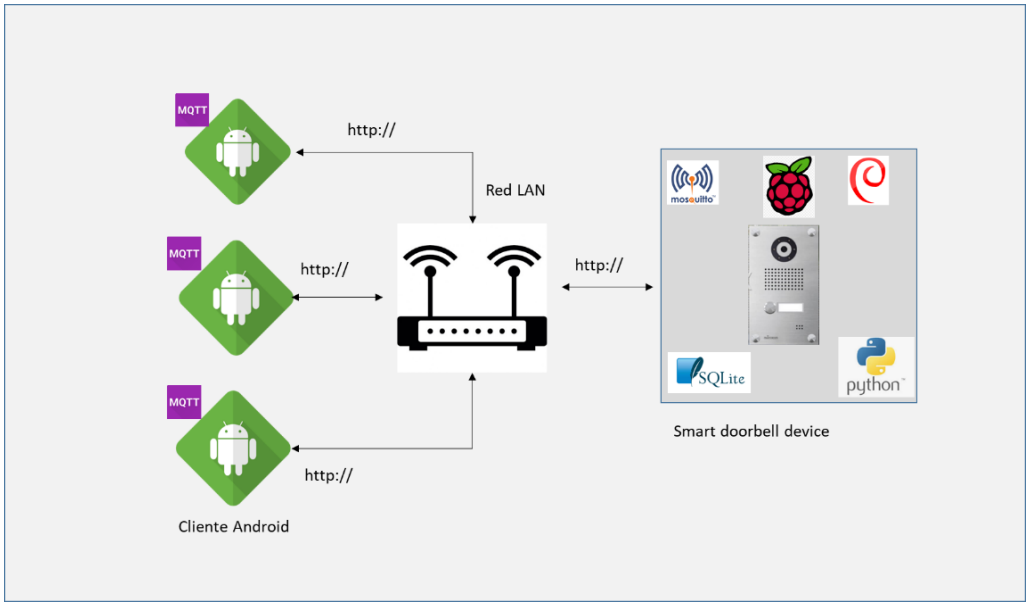


Figura 8. Arquitectura de la solución red de área local (LAN).

FUENTE: Elaboración Propia.

La solución propuesta al momento es mediante una red LAN esto implica que todos los dispositivos tanto android como el SE estarán conectados a la misma red, por lo tanto, deben tener visible el IP entre sí. Este escenario, aunque actualmente no es muy común es posible que se empiece a presentar al momento que las compañías telefónicas lleven el servicio de internet por fraccionamiento.

2.6.1 Arquitectura hardware dispositivo

A Continuación, se muestra el diagrama esquemático de componentes utilizados para la generación del prototipo.

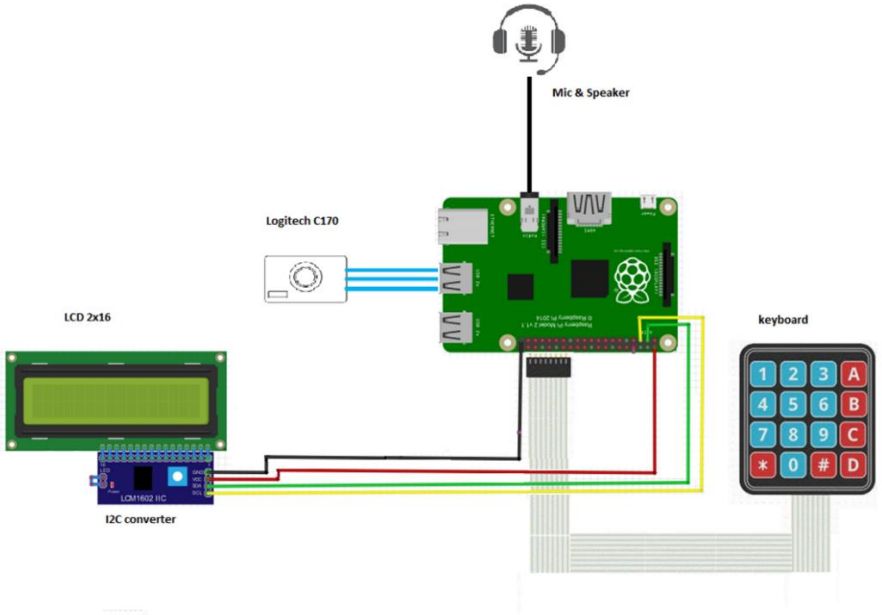


Figura 9. Arquitectura de hardware del SE.

FUENTE: Elaboración Propia.

Considerando las características del proyecto se optó usar como plataforma base una raspberryPi debido a la amplia gama de librerías compatibles con ella, así como la extensa documentación que existe en internet [14] [15].

Además, otra de las razones por la cual se eligió esta plataforma es por la compatibilidad de hardware, casi cualquier hardware tal como teclado, display, micrófono es compatible con raspberryPi.

Por último, el uso de un Linux embebido era una característica esencial en el proyecto. Un Linux embebido es de mucha utilidad para configurar rápidamente los periféricos. La lista de materiales para la implementación de este prototipo es la siguiente:

---

14 Raspberry Pi 4, RASPBERRY PI FOUNDATION, Consultado 10 Nov 2019, https://www.raspberrypi.org/

15 Raspberry Pi OS, RASPBERRY PI FOUNDATION, Consultado 10 Nov 2019, https://www.raspberrypi.org/downloads/raspbian/

Componente	Caracteristicas
Raspberry Pi Model B+ 	ARM Cortex A7 de cuatro núcleos a 900 MHz y 1Gb de SDRAM. Cuenta con tiene: • 100 Base Ethernet. • 4 puertos USB. • 40 pines GPIO. • Puerto Full HDMI. • Entrada para Micro SD card. • VideoCore IV 3D graphics core.
Logitech C170 	• USB 2.0 • Microfono integrado • Resolucion 1024x286 • Fotografia 5MP
Teclado matricial 	• 4x4 • 5V
Display LCD 	• Módulo de pantalla LCD con luz negra azul. • Gran ángulo de visión y alto contraste. • Controlador LCD integrado estándar de la industria HD44780 equivalente. • LCM tipo: caracteres • Puede mostrar 2 líneas X 16 caracteres. • Funciona con 5 V CC. • Módulo de dimensión: 80mm x 35mm x 9mm x mm • Tamaño del área de visión: 64,5mm x 16mm.
Bocina 	• Buzzer de alta velocidad • Voltaje de alimentación: 5V • Se puede variar la frecuencia con PWM • Medidas: 26 x 15 x 10mm

Cuadro 5. Lista de componentes y sus especificaciones para construcción del prototipo.

FUENTE: Elaboración Propia.

A continuación, describimos la funcionalidad de cada uno de los componentes con respecto al proyecto:

- ***Cámara***: esta cámara capturara la imagen de la persona que está llamado a la puerta y la enviara al dispositivo android.
- ***Micrófono***: se utilizará para enviar voz con el fin de interactuar con el cliente android
- ***Teclado Matricial***: Periférico que inicia el proceso al arribar una persona y llamar en la entrada a través de este teclado se va a

ingresar el número de casa o código al cual el usuario se quiere comunicar

- ***Display***: Su función será dar alguna retroalimentación de los estados del dispositivo tales como: ***en servicio***, ***llamando***, ***etc***.
- ***Bocina***: Mandara el sonido enviado desde el cliente de android además de dar un sonido de retroalimentación al cliente tal que parezca una llamada telefónica.

2.6.2 Arquitectura software dispositivo

En esta sección se muestra el desarrollo del software embebido. En la Figura 10 se muestra la configuración del software para el dispositivo, en el cual se hace notar que el software de python va a ser la parte medular del trabajo, esto es debido a su maleabilidad además que este lenguaje se puede comunicar a nivel de sistema operativo leyendo señales de los dispositivos. A su vez el SE se comunica con los componentes de más alto nivel tales como el servidor RTP, base de datos Embebida y el bróker de mensajería (MQTT).

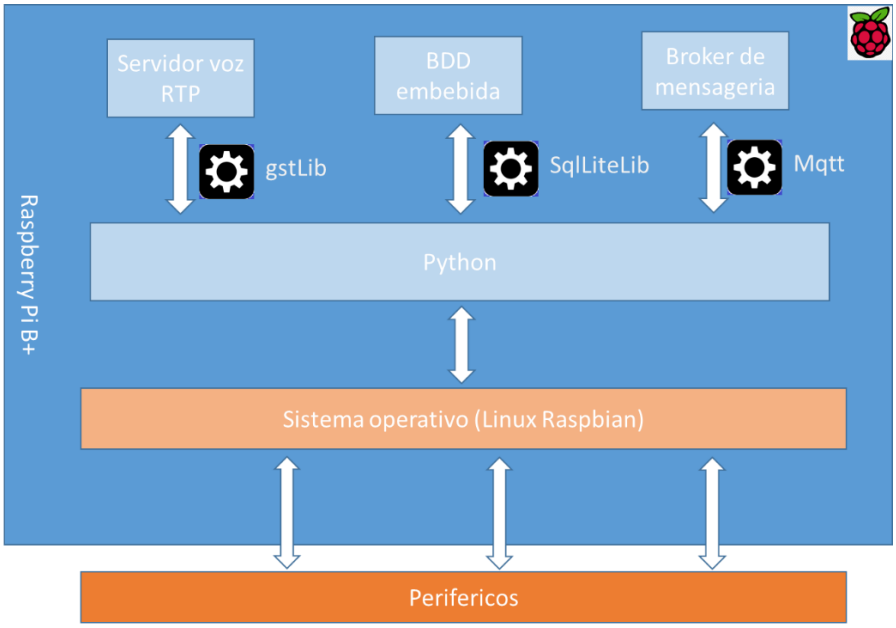


Figura 10. Diagrama de software SE.

FUENTE: Elaboración Propia.

2.6.3 Protocolo de comunicación RTP

Se decidió utilizar el protocolo RTP[16] que es un protocolo más robusto y diseñado específicamente para *stream* de audio y video. Este protocolo trabaja mucho mejor que otras opciones más ligeras tales como: VAT, UDP o XTP[17], debido a que está pensado para ser una plataforma de comunicación en diferentes escenarios. En posteriores iteraciones se planea uso de servicios cloud.

Dentro de algunos artículos que se consultaron a continuación se enlistan las principales ventajas a favor RTP[18](ver referencias al pie de página):

- A diferencia de mandar audio sobre el protocolo TCP (Tanenbaum & Wetherall p. 482) directamente en RTP si hay perdida de paquetes no se genera un retraso por la espera de los paquetes perdidos.
- TCP no soporta multicast. Si bien en esta iteración la comunicación será punto a punto a visión a futuro el muticast va a ser necesario.
- XTP [19] no está diseñado para streaming de datos su diseño es para otros tipos de datos enfocados en propósito general.
- Es un estándar muy utilizado hay librerías para la mayoría de las plataformas
- El Protocolo VAP actualmente está en desuso, no existe soporte o documentación para dicha tecnología.

Ventajas propias del protocolo RTP:

- Se puede utilizar un rango amplio de puertos además configurable en los firewalls
- Se puede configurar la calidad del audio/video
- Uso de diferentes códec.
- Permite comunicación bidireccional.
- Diseñado para aplicaciones en tiempo real.

16 Muhammet ErdalKaynam Hedayat, Real-time transport protocol stream detection system and method, 2012, Consultado 10 Nov 2019, https://patents.google.com/patent/US8306063B2/en
17 Some Frequently Asked Questions about RTP, Columbia, Consultado 10 Nov 2019, http://www.cs.columbia.edu/~hgs/rtp/faq.html#real-time
18 RtpStream. Google, 2018, Consultado 10 Nov 2019, https://developer.android.com/reference/android/net/rtp/RtpStream
19 Transport layer protocols and architectures for satellite networks, SatNext, Consultado 10 Nov 2019, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.7647&rep=rep1&type=pdf

2.7 Diseño software cliente android

A continuación, se muestra el diagrama de clases para la comunicación entre el cliente y el SE

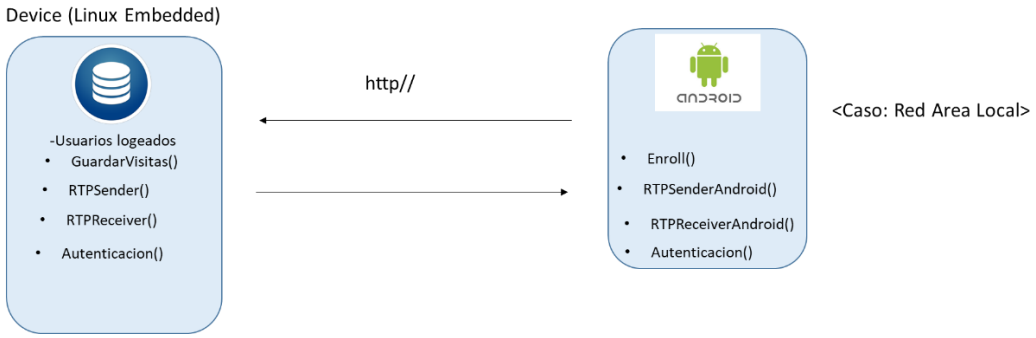


Figura 11. Diagrama de clases dispositivo para casa habitación.

FUENTE: Elaboración Propia.

Para esta fase del desarrollo se implementaron las siguientes clases dentro del cliente android.

Enroll()

Clase para dar de alta nuevos usuarios

RTPSender()

Clase para enviar el stream de voz hacia el dispositivo.

RTPReceiver()

Clase para recibir el stream de voz

ImageReceiver()

Clase para recibir la imagen desde el dispositivo

Autenticacion()

Clase que va a manejar el acceso al dispositivo

2.7.1 Pantallas cliente android

A continuación, se muestra el catálogo de las pantallas para el cliente android.

Login

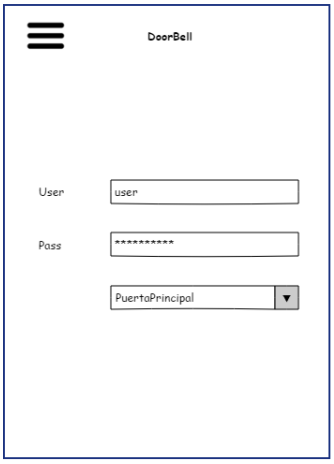


Figura 12. Pantalla de login.

FUENTE: Elaboración Propia.

En la Figura 12 se muestra la pantalla cuya funcionalidad es validar un usuario y contraseña. Para la primera vez que se autentifique el usuario va a tener un usuario y password por default, posteriormente se podrá dar de alta mas usuarios dentro de la misma App. En el scroll down que muestra la figura 12 aparecen los dispositivos disponibles al cual se requiere conectar se debe elegir dentro de dicha lista.

Llamando

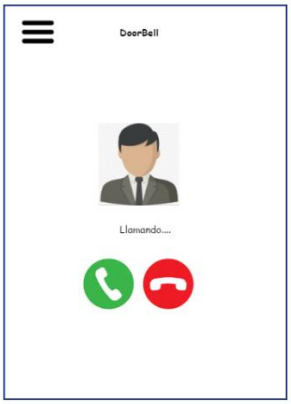


Figura 13. Pantalla alguien llamando a la puerta.

FUENTE: Elaboración Propia.

En la Figura 13 se muestra cuando alguien está llamando a la puerta aquí se atiende el llamado o se rechaza

Llamada en curso

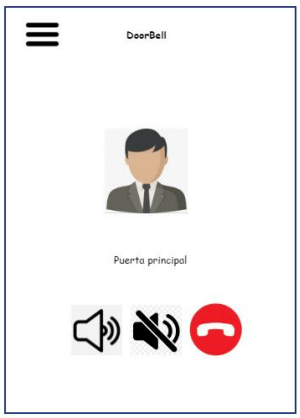


Figura 14. Pantalla llamada en curso.

FUENTE: Elaboración Propia.

Cuando existe una llamada en curso se despliega una pantalla tal como la Figura 14 semejante a cualquier pantalla normal de llamada aquí se puede hacer mute o colgar.

Capítulo 3

Desarrollo y validación

Capítulo 3. Desarrollo y validación

En esta sección se exponen los aspectos técnicos específicos de la implementación del SE y del cliente android. Para cada uno de los componentes se dan a conocer sus detalles técnicos tanto de configuración como de programación de código. Además, se hace una validación contra los objetivos planteados inicialmente en la sección de análisis y diseño para saber cuáles se pudieron lograr y cuales no fueron viables.

3.1 Display

Para saber el status del dispositivo se tiene un display lcd 16x2 utilizando un adaptador para el uso del protocolo de comunicación I2C[20] donde se deben tener los siguientes status:

-Llamando <número de casa>

-Llamada en proceso

*Figura 15**.** Pantalla LED 16x2*

FUENTE: *Elaboración Propia.*

El diagrama de conexión será el siguiente

[20] I2Cbus, Consultado 15 Oct 2019, https://www.i2c-bus.org/

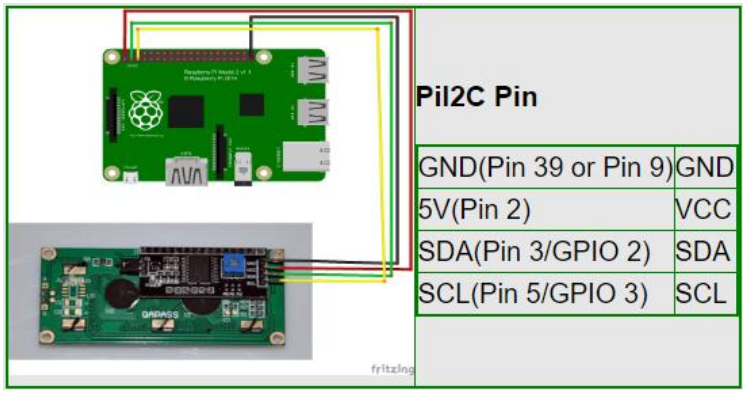


Figura 16. Conectividad pantalla LED con el convertidor I2C y los pines de raspberryPi

FUENTE: Elaboración Propia.

Se decidió utilizar un adaptador para usar I2C para utilizar un mínimo número de conexiones, como se muestra en la figura 15 solo se necesitan 4 conexiones. De otra manera estos displays necesitan más de 17 conectores.

El manejo del display se hizo con python se utilizaron las librerías:

- RPi.GPI
- smbus

Además de librerías de código abierto para ver este código completo consultar en el Anexo II.

3.2 Notificaciones

Se explica a detalle todo el proceso para lanzar notificaciones. El proceso utiliza el patrón de diseño Publisher/subscriber para esto ya existen librerías y herramientas en Raspbian. Además, se explica cómo se gestionan las notificaciones usando una BDD embebida y los diagramas de arquitectura en general de todo el proceso.

3.2.1 Configuración de notificaciones

Para las notificaciones hacia los usuarios desde el SE implementó un protocolo de mensajería basado en el patrón de diseño Publisher/subscriber [Karumanchi & Meda, 2012]. Este protocolo se basa en que una entidad pública mensajes en un topic y que diferentes clientes lo escuchan. La manera lógica de separar cada casa

será que todos los integrantes de una misma estarán subscritos al mismo topic. La implementación de este protocolo para raspberryPi es el MQTT[21] utilizando el proyecto open source Mosquitto[22].

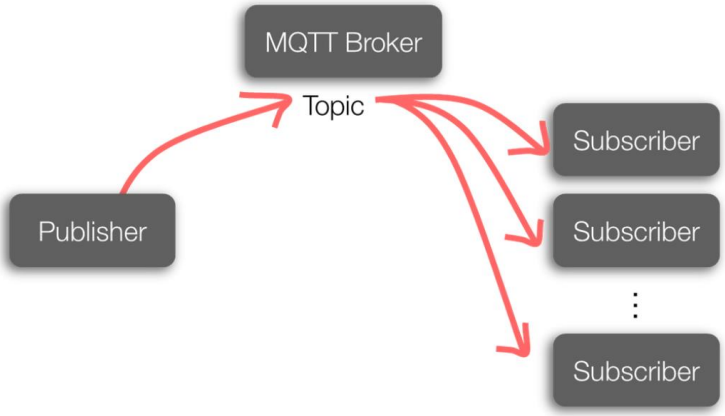


Figura 17 Diagrama general del funcionamiento general de MQTT

FUENTE: Karumanchi & Meda, 2012.

En el caso del prototipo el *publisher* va a ser la raspberry pi y los subscriptores serán los clientes en android que estarán escuchando un topic. Al momento de recibir un mensaje al topic quiere decir que alguien está llamando a la puerta, por lo cual iniciara la comunicación con el dispositivo.

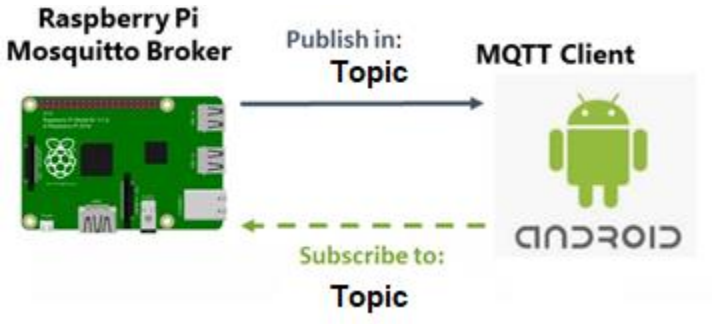


Figura 18. Diagrama de funcionamiento MQTT.

FUENTE: *Elaboración Propia*.

[21] MQTT, Consultado 15 Oct 2019, http://mqtt.org/
[22] Mosquitto MQTT Broker, Consultado 15 Oct 2019, https://mosquitto.org/

A continuación, vamos a mostrar los pasos para la configuración del servidor MQTT dentro de la raspberryPi.

Instalación de las librerías:

```
pi@raspberry:~ $ sudo apt update
pi@raspberry:~ $ sudo apt install -y mosquitto mosquitto-clients
```

Checar los servicios corriendo:

```
pi@raspberry:~ $ sudo systemctl enable mosquitto.service
```

Si el servicio se levanta correctamente vamos a ver una pantalla como la siguiente:

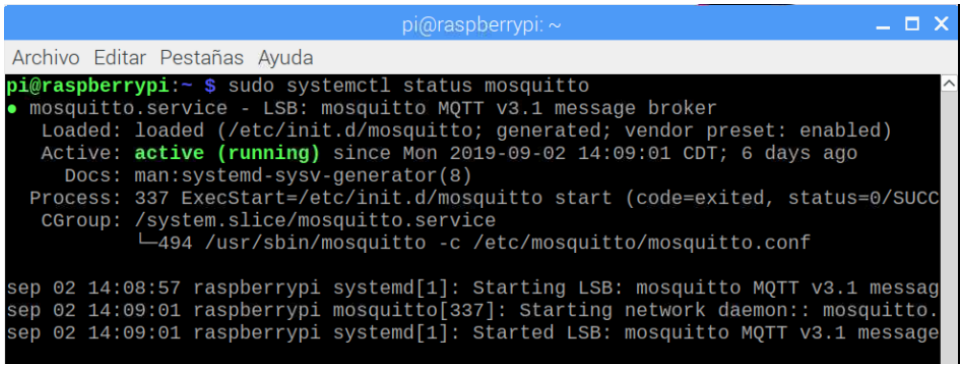


Figura 19. Verificación de estado del servicio mosquitto raspberryPi
FUENTE: Elaboración Propia.

Subscribir a un tópico:

```
pi@raspberry:~ $ mosquitto_sub -d -t testTopic
```

Publicar en un tópico (desde la raspberryPi):

```
pi@raspberry:~ $ mosquitto_pub -d -t testTopic -m "Hello world!"
```

En la Figura 20 se muestra como desde el cliente de android se recibe el mensaje, para esto el cliente está escuchando el mismo tópico llamado *testTopic* para ver el código con más detalle ir a la sección de Anexo V

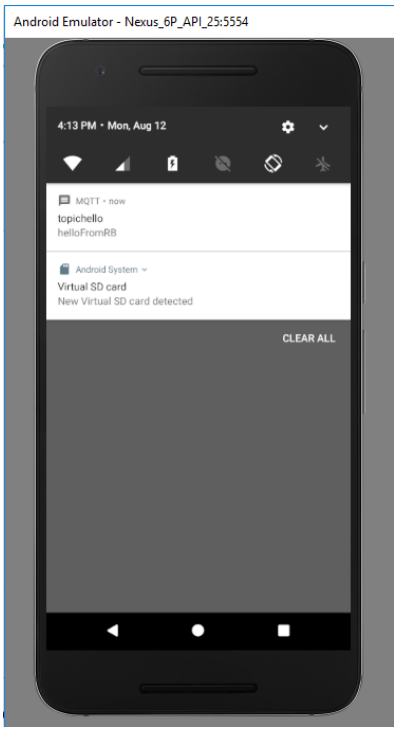


*Figura 20**.** Cliente android recibiendo mensaje MQTT.*

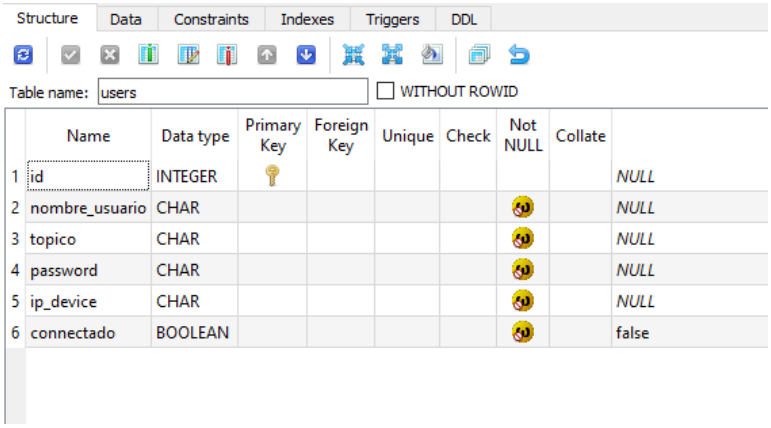
FUENTE: Elaboración Propia.

3.2.2 Gestión de las notificaciones

Para el uso y almacenamiento de todos los tópicos se utiliza una BDD embebida en, nuestro caso implementamos SQL Lite[23], se decidió usar esta BDD por la extensa documentación que se encuentra en la red además por la compatibilidad con python. Para la conectividad con python se utilizó la librería sqlite[24].

A continuación se muestra el schema de BDD:

[23] SQLLite, Consultado 10 Nov 2019, https://sqlite.org/index.html
[24]DB-API 2.0 interface for SQLite databases, Consultado 15 Nov 2019, https://docs.python.org/2/library/sqlite3.html



	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	
1	id	INTEGER							NULL
2	nombre_usuario	CHAR							NULL
3	topico	CHAR							NULL
4	password	CHAR							NULL
5	ip_device	CHAR							NULL
6	connectado	BOOLEAN							false

Figura 21. Estructura de la tabla para gestión de clientes SE.

FUENTE: Elaboración Propia.

Por el momento se tiene contemplada una sola tabla en la BDD la cual contiene los siguientes campos:

id: llave única de la tabla.

nombre_usuario: un identificador que se va a dar de alta mediante el cliente de android.

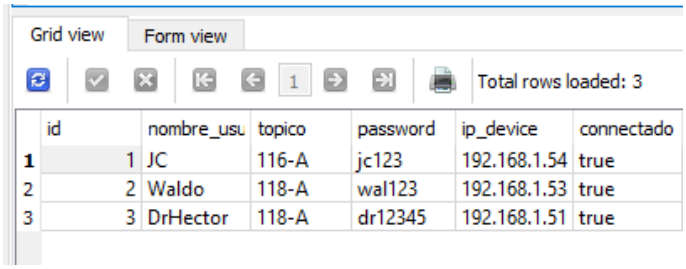
tópico: para esta iteración del componente va a ser el número de casa.

password: clave secreta que se dará de alta desde el cliente android.

ip_device: el ip del cliente este se dará de alta la primera vez desde e cliente android pero posteriormente se seguirá actualizando debido a que muchas veces este ip cambia dentro del contexto de una red LAN

conectado: campo tipo booleano que indicara con true/false si el dispositivo está conectado, para actualizar este estado se va a checar periódicamente si el dispositivo aún sigue conectado.

A continuación, se muestra un ejemplo de datos.



	id	nombre_usu	topico	password	ip_device	connectado
1	1	JC	116-A	jc123	192.168.1.54	true
2	2	Waldo	118-A	wal123	192.168.1.53	true
3	3	DrHector	118-A	dr12345	192.168.1.51	true

Figura 22. Datos de ejemplo de la tabla en BDD.

FUENTE: Elaboración Propia.

Como podemos observar el campo llamado tópico es el número de casa (así se maneja la lógica en esta iteración), esto quiere decir que cuando alguien llame a esa casa todos los habitantes de la misma van a recibir la notificación el que conteste primero la llama será el que se podrá comunicar con la persona en la puerta.

Debido a que en esta primera iteración está enfocada principalmente en la comunicación y notificaciones el schema de BDD es bastante simple.

3.3 Comunicación

En esta sección explica con detalle el proceso de comunicación desde el punto de vista técnico, así como sus diferentes componentes. También se muestran los diagramas de arquitectura para ejemplificar con más claridad las interacciones de los diferentes componentes.

3.3.1 Recibir voz desde el dispositivo hacia el cliente android

Para él envió de voz desde la puerta hacia el cliente android se tiene pensado que el SE sea un servidor de voz RTP, mientras que el cliente android va recibir el stream enviado por dicho servidor.

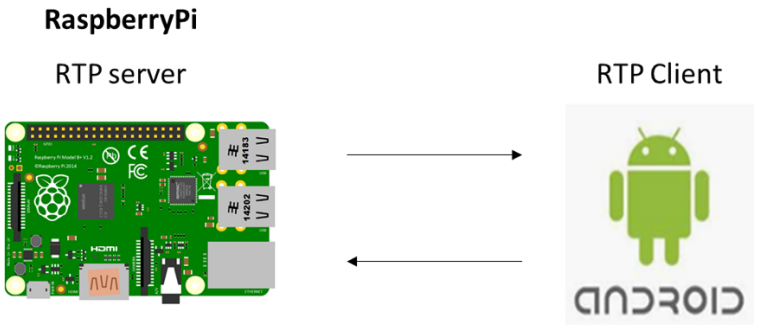


Figura 23. Esquema de servidor de voz.

FUENTE: Elaboración Propia.

Comenzando del lado del SE se está utilizando la librería VLC[25] para incorporar en servidor de voz RTP como se muestra a continuación:

vlc -vvv rtp://@192.168.1.72:22222

***Donde**:*

192.168.1.72 - Ip local del raspberryPi (SE)

22222: puerto a consultar

vvv: parámetro de entrada de la librería

Del lado del cliente existen librerías de android compatibles con RTP las cuales fueron implementadas en la siguiente sección se explica con mas detalle.

3.3.2 Cliente android recibiendo de voz desde el SE

En el punto anterior configuramos la rasberryPi(SE) como servidor de voz, en este punto se explicara cómo se va a escuchar la voz que está publicando la raspberryPi (para consultar el código completo relativo a recibir voz en el cliente android ver Anexo III).

Aquí la parte más destacable del código:

```
WifiManager wifiMngr = (WifiManager) getSystemService(WIFI_SERVICE);
WifiInfo wifiInfo = wifiMngr.getConnectionInfo();
int ip = wifiInfo.getIpAddress();
String ipAdress = Formatter.formatIpAddress(ip);
Log.d("ipAdressssss", ipAdress);

InetAddress inetAddressRemote = InetAddress.getByName(ipAdress); //Ip
Device (device) ip
InetAddress.getByName("192.168.1.68");InetAddress.getByAddress(new
byte[]{(byte) 192, (byte) 168, (byte) 1, (byte) 65});
Log.d("InetAddress", "InetAddress5......");
Log.d("get getLocalPort 2", String.valueOf(audioStream.getLocalPort()));
```

Un punto importante a destacar es que se debe tener acceso desde el Manifest al componente Wifi.

3.3.3 Cliente android enviando voz al SE

A continuación, se muestra el código más destacado para él envió de voz desde el cliente al SE (para consultar condigo completo ver Anexo IV).

```
AudioManager audio =  (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
```

[25] libVLC, Consultado 10 Nov 2019, https://wiki.videolan.org/LibVLC

```
audio.setMode(AudioManager.MODE_IN_COMMUNICATION);
AudioGroup audioGroup = new AudioGroup();
audioGroup.setMode(AudioGroup.MODE_NORMAL);
AudioStream audioStream = new
AudioStream(InetAddress.getByAddress(getLocalIPAddress ()));
audioStream.setCodec(AudioCodec.PCMU);
audioStream.setMode(RtpStream.MODE_NORMAL);
audioStream.associate(InetAddress.getByAddress(new byte[]{(byte) 192,
(byte) 168, (byte) 1, (byte) 72}), 22222);
audioStream.join(audioGroup);
```

En este código se especifica el ip del SE dentro de la red LAN y se genera un grupo virtual entre el actual cliente y el SE esto es por parte el protocolo RTP así es como pide la especificación. Este grupo se interrumpirá en cuanto la comunicación sea terminada.

3.3.4 Envió de voz desde el SE hacia el cliente

Como se muestra en la figura 23 la comunicación entre el SE y el cliente android debe ser bidireccional es decir ambos deben tanto enviar como recibir voz (ver anexo A). A continuación, mostramos como el dispositivo recibe la voz desde el cliente utilizando la librería de Linux gst[26]:

gst-launch-1.0 -v alsasrc device=plughw:1,0 \

! mulawenc ! rtppcmupay ! udpsink host=192.168.1.69 port=5001

Donde:

device=plughw:1,0 : especifica el numero de input es el conector 3.5 mm(donde van las bocinas).

host: ip del cliente android que está mandando el stream de audio

port: puerto donde está escuchando la raspberryPi.

3.4 Captura de imagen en el dispositivo

Una de las funcionalidades principales es relacionada con la cámara de video, esta se debe centrar en el rostro de la persona que está llamando a la puerta, para esto se utilizó python con la implementación de una librería en openCV . Se decidió este lenguaje debido a que ya tiene librerías implementadas para el reconocimiento

[26]Gstreamer, Consultado 10 Nov 2019, https://gstreamer.freedesktop.org/documentation/installing/on-linux.html?gi-language=c

facial, así como una extensa documentación en diferentes textos (M. A. O. Vasilescu & Terzopoulos, 2002). En la siguiente figura vemos el despliegue de la imagen. Para consultar el código activación cámara ver Anexo I

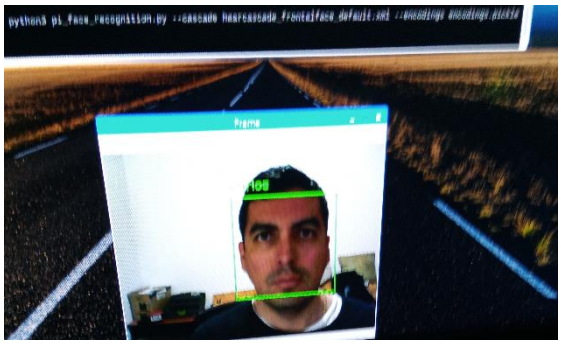


*Figura 24***.** *Detección de rostro con Python.*

FUENTE: *Elaboración Propia*.

Otra característica importante del dispositivo es que al momento de detectar un rostro se tomará la captura de imagen, dicha imagen será enviada por medio del protocolo de comunicación FTP[27] hacia el cliente android. Cabe destacar que la imagen solo será desplegada del lado del cliente android. En el SE el único despliegue de información es texto en la pantalla LCD de la figura 15.

3.5 Envió de imagen desde el dispositivo hacia el cliente android

Con respecto al envió de imagen desde el SE hacia el cliente android se implementó mediante un protocolo FTP en python a continuación se muestra el diagrama de funcionamiento.

[27]Managefiletransfer, Consultado 15 Oct 2019, https://managefiletransfer.wordpress.com/2011/08/02/mft-b2bconsulting/

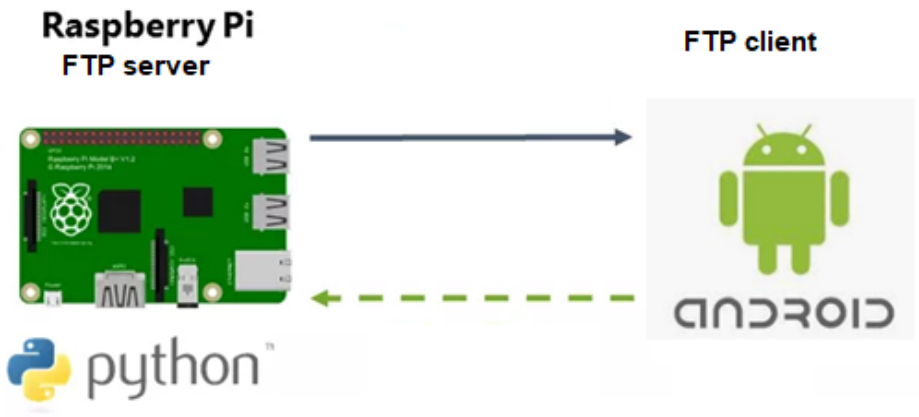


Figura 25. Diagrama de comunicación entre componentes protocolo FTP.

FUENTE: Elaboración Propia.

Se decidió utilizar este protocolo por las siguientes razones:

- Existe una amplia gama de librerías para este protocolo
- Hay versión segura del protocolo SFTP en caso de ser requerido para las posteriores implementaciones
- Es un protocolo para enviar archivos pesados
- Existe una amplia documentación
- Para el caso de las tecnologías que estamos usando existen librerías con compatibilidad

3.6 Validación

En esta sección se muestran los resultaos relacionados al seguimiento de lo que se planteó en el capítulo 2 referente a análisis de riesgos y los resultados que se obtuvieron tal como especifica la metodología.

3.6.1 Validación de riesgos

Tal como se mencionó en la sección de análisis y diseño se utiliza la metodología en espiral (Sommerville, 2011), en la sección 2 se mostraron los siguientes puntos en el análisis de riesgos:

1.-La interferencia natural en la comunicación se debió a las condiciones físicas (paredes, estructuras de fierro/acero) etc.

2.-Bajo performance o mala calidad del audio.

3.-Incompatibilidad entre librerías de las diferentes plataformas.

En cuanto al punto número uno y dos, relacionado con pruebas de conectividad y comunicación se obtuvieron buenos resultados, la calidad del audio fue buena y la comunicación se logró de forma clara. La captura de imágenes se vuelve algo pesada para la raspberryPi aunque funciono satisfactoriamente bien para este prototipo se usó el modelo 3B plus. Ya está disponible en el mercado el modelo 4, este modelo tiene más capacidad de procesamiento y mayor memoria RAM.

Como se mencionó anteriormente dependemos mucho del medio de transmisión y de la calidad de la señal para el caso de una red LAN normalmente no se tienen problemas en estos aspectos.

Con respecto al punto número tres, no hubo problemas de compatibilidad entre librerías se pudo hacer la notificación utilizando MQTT entre el SE y el cliente android, además se pudo hacer la comunicación en dos vías usando RTP para android y Linux.

Otro punto para considerar con respecto a la compatibilidad fue el uso de python para manejo de periféricos, en nuestro caso como se mostró en la fase de desarrollo todos los periféricos se pudieron configurar y manejar sin problemas.

3.6.2 Validación de requerimientos

Esta sección desglosa los requerimientos funcionales y los no funcionales siguiendo la clasificación que se plantea en el texto.

3.6.2.1 Validación de requerimientos funcionales

En esta sección se muestra el resultado de los requerimientos funcionales. En el Cuadro 6 se muestra el resultado de cada uno de los requerimientos funcionales planteados al inicio del análisis.

Requerimiento ID	Descripción	Estatus
RF01	El sistema mandara señal de alerta a todos los teléfonos inscritos según corresponda.	☑

RF02	El Interfon tendrá un buzzer simulando timbre.	☑
RF03	El Interfon tendrá un led para verificar que esta prendido.	☑
RF04	El Interfon mandara stream de audio.	☑
RF05	El Interfon recibirá stream de audio.	☑
RF06	El interfon capturara la imagen de la persona que está en la puerta.	☑
RF07	El cliente Android mandara alerta cuando el inter fon está mandando señal.	☑
RF08	El cliente Android tendrá un icono en el escritorio.	Siguiente etapa
RF09	El cliente Android alertara como si fuera una llamada entrante.	Siguiente etapa
RF09	El cliente Android mandara un stream de audio	☑
RF10	El cliente android desplegara la imagen de la persona.	☑
RF11	El cliente android guardara un historial.	Siguiente etapa

Cuadro 6. Validación de requerimientos funcionales

FUENTE: Elaboración Propia.

3.6.2.2 Validación de requerimientos no funcionales

En esta sección se muestra el resultado de los requerimientos no funcionales. En el Cuadro 7 se muestra el resultado de cada uno de los requerimientos no funcionales planteados al inicio del análisis.

Requerimiento ID	**Descripción**	**Estatus**
NF01	El cliente Android tendrá una actividad corriendo en background siempre.	Siguiente etapa
NF02	El interfon tendrá Linux embebido.	☑
NF03	La comunicación será mediante el protocolo RTP.	☑
NF04	El Interfon manejara sus back Jobs corriendo siempre en Linux.	☑
NF05	El interfon usara python para manejar periféricos tales como teclado y display	☑
NF06	El interfon iniciara el stream de comunicación.	☑
NF07	El interfon mandara notificación a todos los clientes que corresponda utilizando el protocolo MQTT	☑
NF08	El interfon tendrá una BDD embebida para gestión de los clientes dados de alta	☑

NF09	las condiciones de error se van a manejar conforme al testing plan	☑

Cuadro 7 Validación de requerimientos no funcionales.

FUENTE: Elaboración Propia.

A continuación, se muestra imagen del prototipo SE:

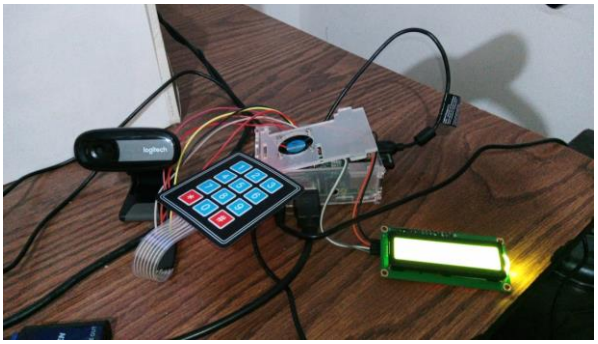


*Figura 26**.** implementación física del prototipo.*

FUENTE: Elaboración Propia.

Se adjuntan las siguientes direcciones de videos del funcionamiento.[28][29]

[28] Demo1, creado 20 Sep 2019 https://drive.google.com/file/d/13CEyScmJhAg0ZlCeUVBAOFgIW2ZtN9Q3/view?usp=sharing
[29] Demo2, creado 20 Sep 2019, https://drive.google.com/file/d/13Afad5Vc-R1NGGRbXiaRHq_5UnljfmqH/view?usp=sharing

Capítulo 4

Impacto

Capítulo 4. Impacto

En este capítulo se analiza el beneficio en el aspecto económico y en el aspecto social que traerá este proyecto al estar en el mercado.

4.1 Impacto social

El impacto social con respecto a la solución planteada se puede ver en cuanto a la seguridad, como se apuntó en la introducción en México la situación delictiva se ha vuelto una situación constante que necesita atenderse con tecnología vanguardista, los puntos a destacar dentro de este contexto son los siguientes.

- Desde adentro de cualquier casa o persona que tenga instalado el servicio en su celular podrá saber quién está llamando a la puerta
- Habrá un historial de las personas que han accedido al fraccionamiento esto podría ser útil en posteriores consultas o para las autoridades.
- Disparador de alertas en caso de ser necesarias

Otro impacto social es la calidad de vida debido a que se puede interactuar desde el celular con la persona que está llamando a la puerta.

4.2 Impacto económico

El impacto económico de esta implementación es notable debido a que potencialmente ayudará a prescindir de personal para el monitoreo de acceso a los fraccionamientos privados, además de que se podrá usar monitoreo remoto desde una central de control donde se pueden dar servicio a varios fraccionamientos esto dara como resultado un ahorro potencial cuantificable (ver Cuadro 10)

Numero de vigilantes	Salario minimo mensual
	$3121
	$6242
	$9363

Cuadro 8. Análisis de costos impacto económico tomando como base salario mínimo(año 2019).

FUENTE: Elaboración Propia.

A continuación, se hace una tabla del costo proyectado de la solución.

Costo del dispositivo	Pago unico
	$3000
monitoreo	$1200 mensuales

Cuadro 9. Análisis de costos para la solución.

FUENTE: Elaboración Propia.

Por último, mostramos una comparativa de potencial ahorro utilizando la solución.

Numero de vigilantes	Salario minimo mensual	costo solucion	ahorro mensual
	$3121	$1200	**$1921**
	$6242	$1200	**$5042**
	$9363	$1200	**$8163**

*Cuadro 10**.** comparativa de costos.*
FUENTE: Elaboración Propia.

Como podemos ver en el Cuadro 5 desde el primer caso donde solo se tiene un vigilante el ahorro es de más de 60% para el caso donde se tengan tres vigilantes se proyecta un ahorro del más de 85%.

Capítulo 5

Trabajo futuro

Capítulo 5. Trabajo Futuro

Para las siguientes iteraciones se pretende utilizar diferentes arquitecturas, el enfoque principal será cambiar de topología de red LAN a utilizar conectividad 4g/5g esto con el fin de utilizar las nuevas tecnologías, así como las diferentes propuestas de topología de res consultadas en otros textos [Podnar et al., 2016] [Slama et al., 2016].

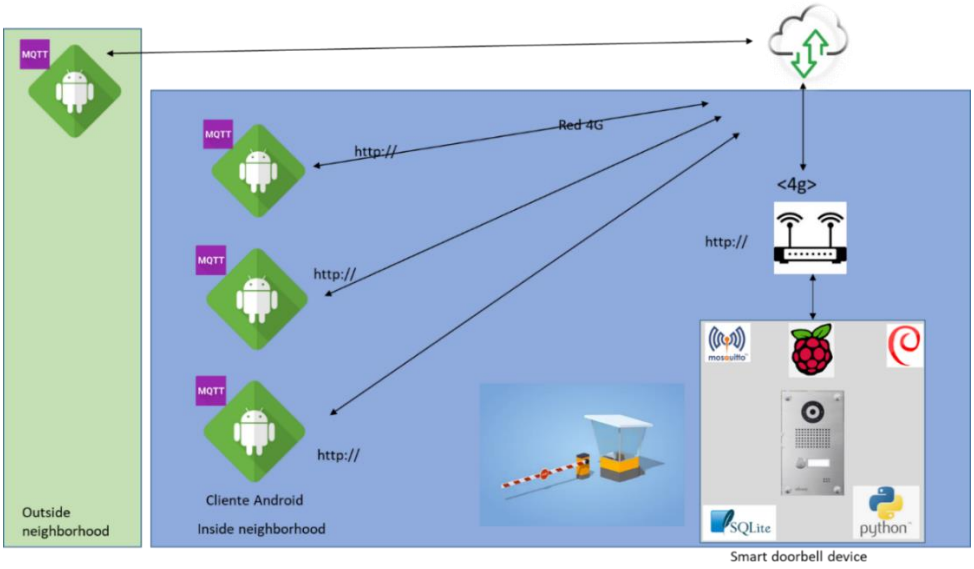


Figura 27. Arquitectura para la solución utilizando conectividad 3g/4g.

FUENTE: Elaboración Propia.

Para esta implementación usando servicios en la nube se tiene pensado implementar una arquitectura de microservicios[30] (Butzin et al. 2016)

Para esta nueva arquitectura algo de funcionalidad será removida del SE embebido y trasladada a un microservicio en la nube como muestra la siguiente Figura 28.

[30]What are microservices?, Consultado 10 Nov 2019, https://microservices.io/

En esta próxima implementación cada servicio va habitar en su propio contenedor Docker[31] que a su vez estará orquestado por Kubernetes[32] el cual es un software propio para administración de microservicios (Balalaie et al 2016).

Cada contenedor por sí mismo es independiente de los demás así además debe tener sus propios componentes tales como BDD, servidor de aplicaciones, etc.

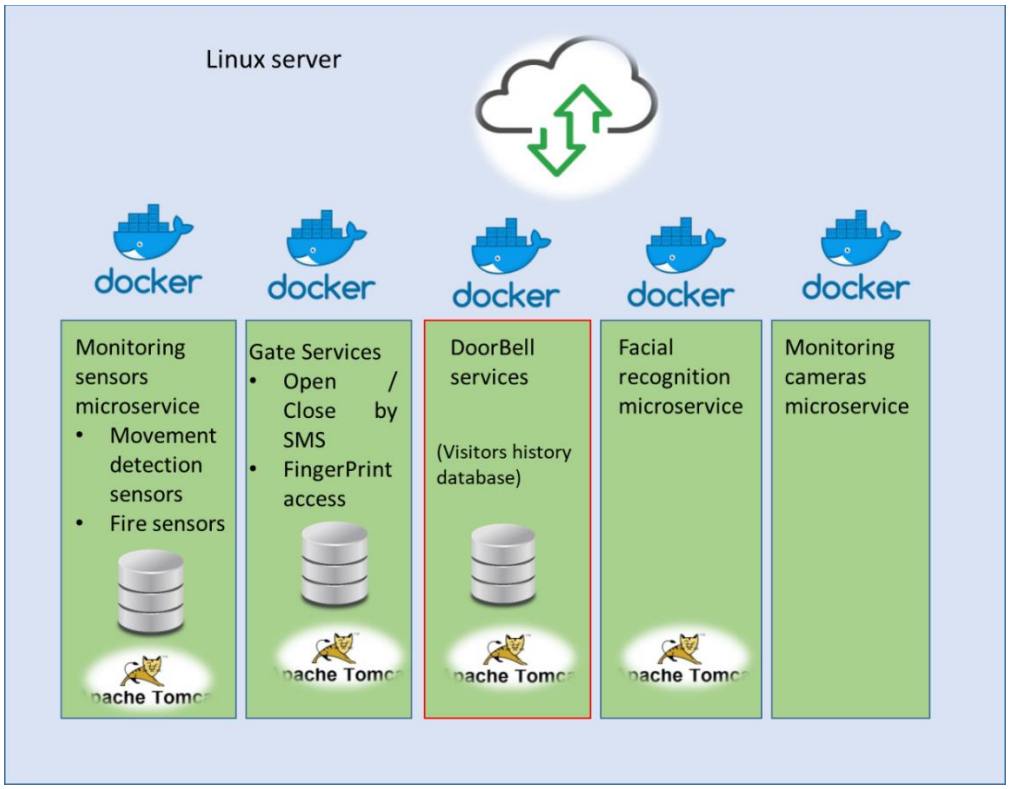


Figura 28. Configuración de microservicios en la nube.

FUENTE: Elaboración Propia.

Como se puede ver en la Figura 28 esta solución busca generar múltiples microservicios este tipo de arquitectura se adapta muy bien a nuestras necesidades debido a que se puede manejar un ambiente heterogéneo es decir se pueden instalar diferentes tipos de tecnologías en diferentes contenedores.

Esto da como resultado un fraccionamiento privado totalmente automatizado manejando tareas tales como abrir/cerrar la puerta, además se puede agregar un

[31] Docker, Consultado 10 Nov 2019, https://www.docker.com/
[32] Kubernetes, Consultado 10 Nov 2019, https://kubernetes.io/es/

lector de huella digital para que la persona acceda al fraccionamiento deje registro de sus datos dactilares. Por otra parte, se podría utilizar como un banco de consulta de imágenes para búsqueda de sospechosos de algún delito en la zona mediante una interfaz web o móvil para las autoridades policiacas.

Esto está alineado con la tendencia de *smart living* que se mencionó en la introducción, debido a que esta tecnología busca mejorar la calidad de la vivienda además de incrementar la seguridad de la misma, por otra parte, también se busca mejorar en general la función de un coto privado lo cual se ajusta a un *smart environment* como plantean los autores (Herring & Kaplan, 2003).

Por último, se muestra en la figura 29 un bosquejo de la visión final a donde planteamos llegar la cual sería una *smart city* que reciba y gestione información de diferentes *smart neighborhood,* esta información puede ser aplicada por ejemplo a recolección de basura, buscar criminales o personas extraviadas, etc.

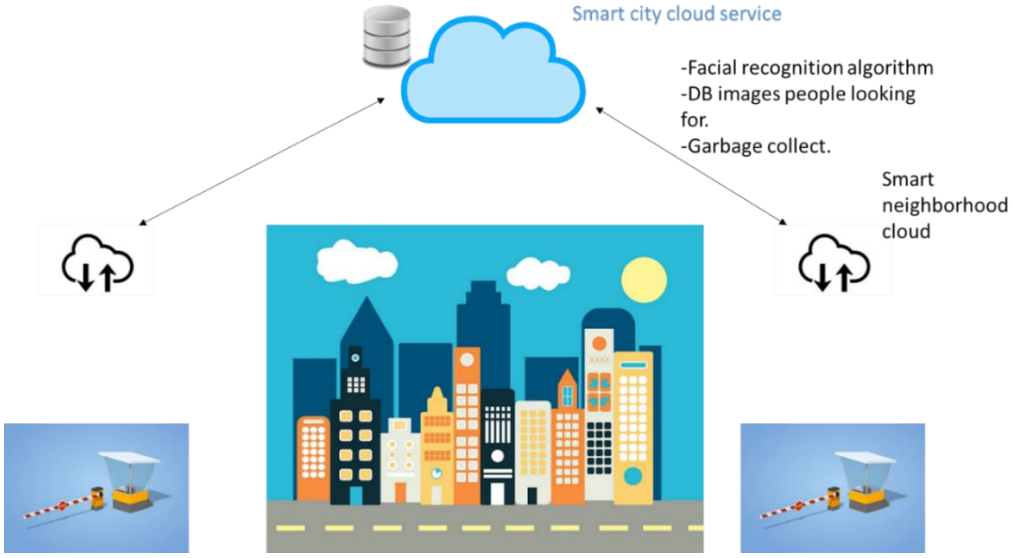


Figura 29. Smart city acumulando información por parte de diferentes smart neighborhood.

FUENTE: Elaboración Propia.

Conclusiones

Conclusiones

En este apartado se muestra las conclusiones y resultados que arrojo la realización de este proyecto, además de lo que se tiene proyectado para seguir desarrollando en las siguientes etapas de desarrollo de el mismo.

Como se mostró en la sección 3 servidor de mensajería MQTT embebido dentro de la raspberryPi funcionó adecuadamente es decir diversos clientes dispersos escuchaban las notificaciones enviadas por la raspberryPi.

Además de esto como se muestra en la sección 3.5 la raspberryPi fue capaz de capturar una imagen y enviarla al cliente android a través de un servidor FTP manera exitosa.

El stream RTP de voz envió y recibió audio de manera satisfactoria, es decir la fidelidad del sonido fue aceptablemente buena, así como de una respuesta razonablemente rápida. Para nosotros respuesta razonablemente rápida significa que al momento de requerir el servicio fue lanzado sin retraso.

Adicionalmente cabe señalar que el desarrollo de una plataforma que contiene un SE y tecnologías como android implica un gran reto, debido a que se necesita conocer diferentes tecnologías, además implica buscar la compatibilidad con cada uno de los diferentes componentes tanto de hardware y como de software para lo cual se necesita una fase de análisis muy profunda para todos los componentes de la plataforma.

Otro punto para destacar es la dependencia del medio de transmisión, en este caso la red ya sea alámbrica o inalámbrica, una velocidad de red lenta puede causar mala experiencia para el usuario en cuanto al uso y funcionamiento de la plataforma.

Como se menciona en el inicio del documento con este proyecto se busca crear una nueva unidad lógica que naturalmente se integra a un vecindario debido a que en un vecindario se tienen muchas funcionalidades en común, además se tiene integración de las personas en este tipo de actividades. En la sección de *trabajo futuro* se muestra el planteamiento además de los siguientes pasos basados

en lo que se está haciendo actualmente con respecto a las tendencias *smart environment, smart city*, planteadas en la sección de introducción.

Bibliografía

Balalaie A., Heydarnoori A. and Jamshidi P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software, vol. 33, no. 3, pp. 42-52.

Bing, L. Fu, Y. Zhuo and L. Yanlei (2011). Design of an Internet of Things-based smart home system. 2nd International Conference on Intelligent Control and Information Processing, Harbin.

Butzin B., Golatowski F., and Timmermann D. (2016). Microservices approach for the internet of things. Springer. IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin.

Dameri, R. P., & Rosenthal-Sabroux, C. (Eds.). (2014). Smart city: How to create public and economic value with high technology in urban space.

Diane J. Cook, Sajal K. Das. (2007). How smart are our environments? An updated look at the state of the art Pervasive and Mobile Computing Volume 3, Issue 2.

Slama Dirk, Puhlman Frank, Morris, Bhatnagar. (2016) Enterprise IOT.

Dodson, Sean. (2003) The internet of things. The Guardian.

Geetika S., Diane C. J.and Schmitter-Edgecombe. (2010) Recognizing independent and joint activities among multiple residents in smart environments. Journal of Ambient Intelligence and Humanized Computing.

Herring Charles and Kaplan Simon. (2003). University of Queensland, Component-Based Software Systems for Smart Environments.

Karumanchi, N., & Meda, S. R. (2012). Peeling Design Patterns: For Beginners & Interviews-Design Interview Questions.

Krylovskiy A., Jahn M. and Patti E. (2015). Designing a Smart City Internet of Things Platform with Microservice Architecture. 3rd International Conference on Future Internet of Things and Cloud, Rome.

Larman, C. (2003). UML y Patrones. Pearson Educación.

Lewis, T. (2008). Smart living: Lifestyle media and popular expertise (Vol. 15). Peter Lang.

Mell Peter and Grance Timothy (2011). The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technology.

Mitton, N., Papavassiliou, S., Puliafito, A. (2012). Combining Cloud and sensors in a smart city environment. J Wireless Com Network.

Nader F. Mir (2006) Computer and Communication Networks, Prentice Hall.

Podnar Zarko Ivanna, Broering Arne, Serrano Martha. (2016). Interoperability and open source solutions for internet of the things pag 12.

Peterson Larry L. and Davie Bruce. (2003). Computer Networks A Systems Approach.

Sommerville, I. (2011). Software Engineering. 1st ed.

Su K., Li J and Fu. H. (2011). Smart city and the applications. International Conference on Electronics, Communications and Control.

Tanenbaum, A. S., and Wetherall, D. (2002). Computer networks(pp. I-XVII). 4th Ed. Prentice hall.

M. A. O. Vasilescu and D. Terzopoulos. (2002). Multilinear image analysis for facial recognition.

Wang M., Zhang G., Zhang C., Zhang J. and Li C. (2013). An IoT-based appliance control system for smart homes. Fourth International Conference on Intelligent Control and Information Processing (ICICIP), Beijing. pp. 744-747.

Anexos

Anexos

Anexo I: Código activacion cámara.

```
import pygame
import pygame.camera
from pygame.locals import *

DEVICE = '/dev/video0'
SIZE = (640, 480)
FILENAME = '/home/pi/FTP/files/image4.jpg'

def camstream():
    pygame.init()
    pygame.camera.init()
    display = pygame.display.set_mode(SIZE, 0)
    camera = pygame.camera.Camera(DEVICE, SIZE)
    camera.start()
    screen = pygame.surface.Surface(SIZE, 0, display)
    capture = True
    while capture:
        screen = camera.get_image(screen)
        display.blit(screen, (0,0))
        pygame.display.flip()
        for event in pygame.event.get():
            if event.type == QUIT:
                capture = False
            elif event.type == KEYDOWN and event.key == K_s:
                pygame.image.save(screen, FILENAME)
    camera.stop()
    pygame.quit()
    return

if __name__ == '__main__':
    camstream()
```

Anexo II: Código manejo de notificaciones y lcd

```
import time
import RPi.GPIO as GPIO
##import rpi.gpio as GPIO
from keypad import keypad

# Simple string program. Writes and updates strings.
# Demo program for the I2C 16x2 Display from Ryanteck.uk
# Created by Matthew Timmons-Brown for The Raspberry Pi Guy YouTube channel

# Import necessary libraries for communication and display use
import lcddriver
import time
## Import MQTT libs
import paho.mqtt.client as mqtt
import paho.mqtt.publish as publish
broker_url = "192.168.1.66"
pub_topic = "topichello"
broker_port = 1883


# Load the driver and set it to "display"
# If you use something from the driver library use the "display." prefix first
display = lcddriver.lcd()
GPIO.setwarnings(False)

if __name__ == '__main__':
    # Initialize
    kp = keypad(columnCount = 3)

    # Main body of code
```

```
try:
    while True:
        # Remember that your sentences can only be 16 characters long!
          print("Writing to display")
          display.lcd_display_string("Teclee numero d", 1) # Write line of text to first line
of display
          display.lcd_display_string("casa seguido *", 2) # Write line of text to second line
of display
          ##time.sleep(2)                                        # Give time for the message to
be read
          ##display.lcd_display_string("Teclee numero d", 1)   # Refresh the first line of
display with a different message
          time.sleep(2)                                       # Give time for the message to be
read
          ##display.lcd_clear()                                  # Clear the display of any data
          ##time.sleep(0.5)                                        # Give time for the message
to be read
    ###### 4 Digit wait ######
          seq = []
          for i in range(4):
              digit = None
              while digit == None:
                  digit = kp.getKey()
              seq.append(digit)
              time.sleep(0.4)
    # Check digit code
          print(seq)
          display.lcd_clear()
          display.lcd_display_string(str(seq), 1)
          time.sleep(1)
          if seq == [2, 2, 0, '*']:
              print("Code accepted")
              display.lcd_display_string("Llamando...", 2)
              client = mqtt.Client()
              client.connect(broker_url,broker_port)
              client.publish(topic="topichello",       payload="RingFromPython",       qos=0,
retain=False)
              time.sleep(3)
              display.lcd_clear()
          else:
              print("Code fail")
              display.lcd_display_string("No existe tal numero", 2)
              time.sleep(3)
              display.lcd_clear()

except KeyboardInterrupt: # If there is a KeyboardInterrupt (when you press ctrl+c), exit the
program and cleanup
    print("Cleaning up!")
    display.lcd_clear()
```

Anexo III: Código envió audio cliente android al SE.

```
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Enumeration;
import android.media.AudioManager;
import android.net.rtp.AudioCodec;
import android.net.rtp.AudioGroup;
import android.net.rtp.AudioStream;
import android.net.rtp.RtpStream;
import android.os.Bundle;
import android.os.StrictMode;
import android.app.Activity;
import android.content.Context;
import android.util.Log;

public class MainActivity extends Activity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        StrictMode.ThreadPolicy policy= new
StrictMode.ThreadPolicy.Builder().permitAll().build();
        StrictMode.setThreadPolicy(policy);
        try {
            AudioManager audio =  (AudioManager) getSystemService(Context.AUDIO_SERVICE);
            audio.setMode(AudioManager.MODE_IN_COMMUNICATION);
            AudioGroup audioGroup = new AudioGroup();
            audioGroup.setMode(AudioGroup.MODE_NORMAL);
            AudioStream audioStream = new
AudioStream(InetAddress.getByAddress(getLocalIPAddress ()));
            audioStream.setCodec(AudioCodec.PCMU);
            audioStream.setMode(RtpStream.MODE_NORMAL);
            audioStream.associate(InetAddress.getByAddress(new byte[] {(byte)192, (byte)168,
(byte)1, (byte)19 }), 22222);
            audioStream.join(audioGroup);

        } catch (Exception e) {
            Log.e("----------------------", e.toString());
            e.printStackTrace();
        }
    }
    public static byte[] getLocalIPAddress () {
        byte ip[]=null;
        try {
            for (Enumeration<NetworkInterface> en = NetworkInterface.getNetworkInterfaces();
en.hasMoreElements();) {
                NetworkInterface intf = en.nextElement();
                for (Enumeration<InetAddress> enumIpAddr = intf.getInetAddresses();
enumIpAddr.hasMoreElements();) {
                    InetAddress inetAddress = enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress()) {
                        ip= inetAddress.getAddress();
                    }
                }
            }
        } catch (SocketException ex) {
            Log.i("SocketException ", ex.toString());
        }
        return ip;

    }
}
```

Anexo IV: Código para recibir audio en cliente android desde el SE.

```
import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.net.rtp.AudioCodec;
import android.net.rtp.AudioGroup;
import android.net.rtp.AudioStream;
import android.net.rtp.RtpStream;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.StrictMode;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.text.format.Formatter;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Enumeration;
```

```
public class MainActivity extends Activity {

    AudioStream audioStream;
    AudioGroup audioGroup;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d("On Create", "Create......");
        setContentView(R.layout.activity_main);
        StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitNetwork().build();
        StrictMode.setThreadPolicy(policy);
        AudioManager audio = (AudioManager)getSystemService(AUDIO_SERVICE);
        audio.setMode(AudioManager.MODE_IN_COMMUNICATION);
        Log.d("audio", "after audio......");
        audioGroup = new AudioGroup();
        audioGroup.setMode(AudioGroup.MODE_ECHO_SUPPRESSION);//MODE_ECHO_SUPPRESSION
        InetAddress inetAddress;
        try {
            Log.d("try", "try......");
            Log.d("get getLocalHost 000...", InetAddress.getLocalHost().toString());
           // Log.d ("get getLocalPort 0...",String.valueOf(audioStream.getLocalPort()));
            //inetAddress = InetAddress.getByName("192.168.1.67");
            byte[] adressLoc = getLocalIPAddress ();
            inetAddress = InetAddress.getByAddress(adressLoc);
            Log.d("InetAddress", "InetAddress1......");
            Log.d("adressLoc", adressLoc.toString());
            audioStream = new AudioStream(inetAddress);
            Log.d("InetAddress", "InetAddress2......");
            audioStream.setCodec(AudioCodec.PCMU);
            Log.d("InetAddress", "InetAddress3......");
            audioStream.setMode(RtpStream.MODE_NORMAL);
            Log.d("InetAddress", "InetAddress4......");

            WifiManager wifiMngr = (WifiManager) getSystemService(WIFI_SERVICE);
            WifiInfo wifiInfo = wifiMngr.getConnectionInfo();
            int ip = wifiInfo.getIpAddress();
            String ipAdress = Formatter.formatIpAddress(ip);
            Log.d("ipAdressssss", ipAdress);

            InetAddress inetAddressRemote = InetAddress.getByName(ipAdress); //local
(device) ip InetAddress.getByName("192.168.1.68");InetAddress.getByAddress(new byte[]{(byte)
192, (byte) 168, (byte) 1, (byte) 65});
            Log.d("InetAddress", "InetAddress5......");
            Log.d("get getLocalPort 2", String.valueOf(audioStream.getLocalPort()));

            audioStream.associate(inetAddressRemote, audioStream.getLocalPort());//5001
            Log.d("Port", "after Port......");
            ((TextView)findViewById(R.id.tv_port)).setText(" Port : " +
String.valueOf(audioStream.getLocalPort()));
            Log.d("TxtView", "after TxtView......");
            audioStream.join(audioGroup);
            Log.d("audioStream", "after audioStream......");
        }
        catch ( UnknownHostException e ) {
            e.printStackTrace();
        }
        catch ( SocketException e ) {
            e.printStackTrace();
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
```

```
    @Override
    public void onStart() {
        super.onStart();
        Log.d("On Start", "Start..");
    }

    public static byte[] getLocalIPAddress () {
        byte ip[]=null;
        try {
            for (Enumeration<NetworkInterface> en = NetworkInterface.getNetworkInterfaces();
en.hasMoreElements();) {
                NetworkInterface intf = en.nextElement();
                for (Enumeration<InetAddress> enumIpAddr = intf.getInetAddresses();
enumIpAddr.hasMoreElements();) {
                    InetAddress inetAddress = enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress()) {
                        ip= inetAddress.getAddress();
                    }
                }
            }
        } catch (SocketException ex) {
            Log.i("SocketException ", ex.toString());
        }
        return ip;

    }
}
```

Anexo V: Código para recibir notificaciones.

```
package com.app.androidkt.mqtt;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import android.support.annotation.NonNull;
import android.support.v4.app.NotificationCompat;
import android.util.Log;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallbackExtended;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public class MqttMessageService extends Service {

    private static final String TAG = "MqttMessageService";
    private PahoMqttClient pahoMqttClient;
    private MqttAndroidClient mqttAndroidClient;

    public MqttMessageService() {
    }

    @Override
    public void onCreate() {
        super.onCreate();
        Log.d(TAG, "onCreate");
        pahoMqttClient = new PahoMqttClient();
        mqttAndroidClient = pahoMqttClient.getMqttClient(getApplicationContext(),
Constants.MQTT_BROKER_URL, Constants.CLIENT_ID);

        mqttAndroidClient.setCallback(new MqttCallbackExtended() {
            @Override
            public void connectComplete(boolean b, String s) {

            }
```

```
        @Override
        public void connectionLost(Throwable throwable) {

        }

        @Override
        public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {
            setMessageNotification(s, new String(mqttMessage.getPayload()));
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

        }
    });
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(TAG, "onStartCommand");
    return START_STICKY;
}

@Override
public IBinder onBind(Intent intent) {
    // TODO: Return the communication channel to the service.
    throw new UnsupportedOperationException("Not yet implemented");
}

@Override
public void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "onDestroy");
}

private void setMessageNotification(@NonNull String topic, @NonNull String msg) {
    NotificationCompat.Builder mBuilder =
            new NotificationCompat.Builder(this)
                    .setSmallIcon(R.drawable.ic_message_black_24dp)
                    .setContentTitle(topic)
                    .setContentText(msg);
    Intent resultIntent = new Intent(this, MainActivity.class);

    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(MainActivity.class);
    stackBuilder.addNextIntent(resultIntent);
    PendingIntent resultPendingIntent =
            stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
    mBuilder.setContentIntent(resultPendingIntent);
    NotificationManager mNotificationManager =
            (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(100, mBuilder.build());
```