UNIVERSIDAD TECNOLÓGICA ISRAEL

ESCUELA DE POSTGRADOS

MAESTRÍA EN TELEMÁTICA

MENCIÓN: CALIDAD EN EL SERVICIO

(Aprobado por: RPC-SO-19-No.300-2016-CES)

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE MAGISTER

Título:
Sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles
Línea de Investigación:
Telecomunicaciones y sistemas informáticos aplicados a la producción y la sociedad.
Autor:
Juan David Chimarro Amaguaña, Ing.
Tutor:
Ing. Fidel David Parra Balza, Ph.D

Quito - Ecuador

2020

CERTIFICADO DE APROBACIÓN

Yo, Ing. Fidel David Parra Balza, Ph.D portador de la C.I. 1757469950

en mi calidad de Tutor del trabajo de investigación titulado:

Sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles

elaborado por: Ing. Juan David Chimarro Amaguaña, estudiante de la Maestría en Telemática, mención en Calidad del Servicio de la UNIVERSIDAD TECNOLÓGICA ISRAEL (UISRAEL), para obtener el Título de Magister, me permito declarar que luego de haber orientado, estudiado y revisado la tesis de titulación de grado, la apruebo en todas sus partes.

Lugar y fecha: Quito, 22 de febrero del 2020

Firma

Ing. Fidel David Parra Balza Ph.D

DEDICATORIA

Dedico este Master a mi familia por haberme apoyado tanto en los momentos difíciles, como los satisfactorios y estar a mi lado durante el tiempo que he trabajado para lograr esta nueva meta en mi vida, ya que con esfuerzo y dedicación se puede alcanzar el éxito.

AGRADECIMIENTOS

A Dios y a mi familia por motivarme día a día para lograr los objetivos.

A la Universidad Tecnológica Israel, a mi Tutor de Tesis, docentes y a todas las personas que forman parte de la Escuela de Postgrados, por dar una educación de calidad.

Y a mis amistades de la UISRAEL, del trabajo INTSUPERIOR, y a las personas con las que he compartido momentos especiales siempre los tengo presentes.

RESUMEN

El proyecto de investigación tuvo por objeto desarrollar un sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles, el enfoque metodológico de la investigación fue basado en una investigación aplicada ya que se centra específicamente en buscar la aplicación o utilización de conocimientos teóricos, a la vez que se adquieren otros, después de implementar y sistematizar. La técnica de levantamiento de información que se usó fue la observación indirecta ya que la investigación se llevó acabo por revisión de bibliografía. La teleoperación del brazo robótico se basó sobre la arquitectura del Cloud Computing usando las API y Services de Google Cloud Platform y permitió una interacción directa con Firebase y esta plataforma a su vez, tienen enlaces directos con dispositivos móviles, se hizo uso de protocolos de comunicación IoT (Internet of Things) y pudo conectarse con una la tarjeta Wifi, lo que consistió en tener el control de todas las variables del robot de forma remota en tiempo real, carga y descarga de información de la base de datos, seguridad de la información y obtener estadísticas sobre el uso de la aplicación. Como resultados se valoró el funcionamiento del sistema teleoperado obteniendo información sobre el tráfico tele-transmitido, tiempos de latencia, errores, eventos, estos parámetros permiten mejorar el rendimiento y la calidad del servicio de la aplicación.

Palabras claves: teleoperación, Firebase, Cloud Computing, aplicaciones móviles, IoT, tiempos de latencia.

ABSTRACT

The purpose of the research project was to develop an integrated system for the operation of a teleoperated robotic arm in real time using the Firebase platform with the use of mobile devices, the methodological approach of the research was based on an applied research since it focuses specifically in seeking the application or use of theoretical knowledge, while others are acquired, after implementing and systematizing. The information gathering technique that was used was indirect observation since the investigation was carried out by literature review. The teleoperation of the robotic arm was based on the architecture of Cloud Computing using the APIs and Services of Google Cloud Platform and allowed a direct interaction with Firebase and this platform in turn, have direct links with mobile devices, communication protocols were used IoT (Internet of Things) and was able to connect with a Wifi card, which consisted of having control of all robot variables remotely in real time, loading and unloading information from the database, information security and obtain statistics on the use of the application. As results, the operation of the teleoperated system was evaluated obtaining information on the tele-transmitted traffic, latency times, errors, events, these parameters allow to improve the performance and the quality of the service of the application.

Keywords: Teleoperation, Firebase, Cloud Computing, Mobile Applications, IoT, latency times.

INDICE DE CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

Los avances tecnológicos de la actualidad han permitido, que gran cantidad de dispositivos electrónicos puedan conectarse a la red de Internet, como: sensores, máquinas industriales, manipuladores robóticos, automóviles, electrodomésticos, entre otros, estos se han combinado para crear redes interconectadas de comunicación, así también, se empieza a potenciar nuevos servicios en áreas como: la industria del entretenimiento, telemanufactura, teleoperación, que forma parte de los sistemas distribuidos en tiempo real, es decir, están enfocados a la transmisión de datos en tiempo real, para controlar dispositivos en forma remota. Sin embargo, este proceso requiere la utilización de diferentes plataformas, cada una de ellas con una gran cantidad de librerías, las cuales requieren un trabajo exhaustivo y complejo para poder integrarlas en un sistema teleoperado en tiempo real, esto trae como consecuencia, costos elevados por la cantidad de horas – hombre invertidas para lograr este cometido y como consecuencia tiene algunas limitantes como compatibilidad de hardware, escalabilidad y versatilidad.

Es por esto que surge la necesidad de desarrollar un sistema de teleoperado para un brazo robótico mediante la integración de diferentes librerías, en conjunto con dispositivos móviles ya que es importante abordar desafíos exigentes y puedan mejorar el rendimiento de estos sistemas en tiempo real y a la vez que se puede recopilar información sobre el manipulador robótico y cumplir con estándares de calidad y servicio, por tal razón es de gran interés la investigación y el desarrollo tecnológico de estos sistemas actuales.

La teleoperación se añade a un brazo robótico con el fin de integrar con los sistemas distribuidos en tiempo real, con la finalidad de demostrar la operación de un dispositivo en forma remota, para esto fue necesario hacer una revisión de diferentes estudios relacionados con el tema en estudio como es “Sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles”.

En ese sentido, (Guzmán, Torres, & Galeano, 2014) en su trabajo titulado “Propuesta de un modelo de gestión de servicios en la nube para la manipulación de sistemas robóticos con el uso de dispositivos móviles”, sugiere la utilización de un servidor que permita almacenar información sobre las especificaciones de cada uno de los dispositivos móviles. Al mismo tiempo propone implementar servicios web que operan desde la nube para crear una interfaz

para el usuario de tal manera que facilite la teleoperación del robot por medio de la captura de datos y de una heurística. Los métodos que se realizó en esta investigación se exploran varias alternativas como SW-SOAP es para el control de un dispositivo robótico a partir de los servicios web y el SW Restfull la cual se resalta la simplicidad, la capacidad de transmitir datos a través de protocolos HTTP. Los resultados obtenidos demuestran la utilidad de considerar arquitecturas como el Cloud Computing que, mediante servicios web, habiliten la integración de usuario-robot y servidor y el modelo de teleoperación basado en servicios web RestFul, está fundamentado en los criterios de escalabilidad y versatilidad del software.

Esta investigación aporta al presente trabajo en como considerar la arquitectura en la nube para integrar un modelo de teleoperación basado en servicios web. Proponiendo un modelo mediante servicios web fundamentando criterios de escalabilidad y versatilidad.

Otro documento presentado por (Manzi, Fiorini, & Limosani, 2016) menciona en su trabajo *"Use Case Evaluation of a Cloud Robotics Teleoperation System"* en el cual describe una robótica de nube genérica sistema de teleoperación que permite controlar en tiempo real un robot (conectado con una red 4G) que tiene su transmisión de video como retroalimentación. El sistema propuesto se basa en la nube de Plataforma Azure y sobre tecnologías web recientes, particularmente se probó el sistema con una comunicación entre países en tiempo real para evaluar su rendimiento proporcionando el valor de rendimiento de la comunicación y el retraso promedio entre paquetes recibidos consecutivos en ambos lados robot y teleoperación. Los resultados que obtuvieron mostraron que la tecnología elegida permite tener actuaciones en tiempo real en términos de video y comandos de velocidad de transmisión.

Esta investigación contribuye al actual proyecto ya que permite considerar el tipo de red la cual estaba conectada a una red 4G suficiente para recibir comandos de velocidad y enviar datos de transmisión para permitir un control en tiempo real del robot sobre Nube. Además, el uso de la tecnología web permite utilizar esta solución de teleoperación en cualquier dispositivo moderno, como tableta y teléfono inteligente, lo que brinda una mayor posibilidad para la interacción humano-robot.

De la misma forma, (Alsalemi & Alhomsi, 2017) menciona en su trabajo titulado *"Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation"*, propone implementar un sistema de simulación para médicos en tiempo real, el procedimiento

requirió robustez y coordinación en la arquitectura de red por el cual se hizo uso de Internet de las cosas (IoT), el servicio Firebase como método de comunicación y se discutió las características como baja latencia, compatibilidad de hardware y gestión de datos. Firebase demostró ser una adecuada solución de comunicación que satisfizo la naturaleza urgente de entrenamiento médico porque cada unidad puede acceder en tiempo real a la base de datos y almacenar / leer cualquier parámetro una vez que sea necesario y podrá verificar cambios que ocurrieron, tal técnica pudo aumentar la eficiencia del sistema de comunicación Se establece una topología en estrella entre el dispositivo móvil (unidad) y Firebase (donde Firebase es el Hub) permitirá que el sistema de comunicación se mantenga en operación incluso si alguna unidad en la red no funciona o tenga una mala conexión y además puedan ser fácilmente integrado con el sistema actual y poder establecer condiciones de la vida real.

Esta investigación apoya al presente proyecto ya que permite considerar las características que brinda el servicio Firebase en la teleoperación y la interacción que tiene con los dispositivos móviles como baja latencia, robustez, control remoto y compatibilidad de hardware.

Adicionalmente (Toquica, Benavides, & Motta, 2019) menciona en su trabajo *"Web Compliant Open Architecture for Teleoperation of Industrial Robots"* propone una solución compatible basada en la web para la teleoperación de un robot industrial como una contribución a los crecientes requisitos tecnológicos en la industria, utilizando lenguajes de programación de código abierto y protocolos de comunicación estándar para proporcionar una solución flexible para aplicaciones académicas e industriales. Con los desarrollos recientes de los conceptos de la Industria 4.0, como sistemas ciberfísicos (CPS), internet de las cosas (IoT) y Cloud Computing, existe la necesidad de robustez y eficiencia en arquitecturas compatibles con cualquier proveedor de robótica industrial. La arquitectura propuesta que se desarrolló fue teniendo en cuenta los enfoques actuales que generalmente se utilizan en sistemas robóticos basados en la nube, toma el concepto de software como servicio (SaaS) para difundir servicios en orden y teleoperar un robot. Estos servicios se realizarán por el usuario remoto a través de una programación de aplicaciones interfaz (API) desarrollada en una programación de lenguaje de código abierto y se presentó una integración completa de en herramientas extendidas de JAVA para desarrollar aplicaciones basadas en web como parte de la arquitectura propuesta, demostrando la flexibilidad de la solución descrita en este trabajo con la compatibilidad con

sistemas operativos existentes para reducir los costos de implementación, todos los servicios disponibles se almacenarán en un entorno relacional de base de datos que se clasifican por tipos. Se incluyó la herramienta Cloud, ya que la propuesta es una aplicación basada en web donde los usuarios pueden tener acceso remoto a través de la arquitectura de computación en la nube lo que hace posible controlar un robot industrial desde cualquier punto del planeta con conexión a internet.

Esta investigación aporta al presente trabajo para considerar los desarrollos recientes de la revolución tecnológica actual, y tomar en cuenta las características que permiten integrar el software como servicio (SaaS) para difundir los servicios en orden y teleoperar un robot, usando la arquitectura de computación en la nube e implementación en plataformas de programación de lenguaje de código abierto y reducir los costos.

Los sistemas teleoperados conectados en tiempo real sin importar el lugar donde se encuentre el operario y el lugar donde la aplicación remota este instalada, sirven para realizar diversas tareas de alto riesgo como eliminación de cargas explosivas, la telemedicina, la detección, la manipulación, la explotación minera y la industria química.

Objeto de estudio

En este trabajo de investigación se estudiarán los sistemas teleoperados, en especial se analizará las plataformas que integran los sistemas robóticos teleoperados en tiempo real, ya que actualmente estos sistemas requieren altos recursos económicos en desarrollo porque deben cumplir criterios como la interacción con dispositivos móviles, escalabilidad, versatilidad, baja latencia, robustez, control remoto, compatibilidad de hardware y por ello resulta complejo integrarlas.

Para poder llevar a cabo esta investigación se establece como objetivo general:

Desarrollar un sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles.

Como objetivos específicos se plantean los siguientes:

- Establecer la arquitectura de red de la teleoperación usando el Cloud Computing para la optimización de recursos.
- Generar una aplicación móvil en Android Studio para el control del brazo robótico teleoperado.
- Gestionar la teleoperación en la plataforma Firebase para la minimización de procesos de integración.
- Valorar el funcionamiento en función al cumplimiento de los estándares de calidad y servicio para la medición del rendimiento del sistema teleoperado.

En el presente trabajo de investigación tendrá por alcance desarrollar un sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles, el brazo robótico en el cual se implementará la teleoperación tiene 5 grados libertad, y podrá realizar trabajos de manipulación *pick and place,* el robot tendrá como controlador una tarjeta de Arduino Uno y módulo Wifi ESP8266 que permitirá al robot conectarse a una red Wifi y al servicio Firebase el cual gestionará la información del estado del robot en tiempo real, también se desarrollará una aplicación móvil que será compatible con el sistema operativo Android en la cual se podrá teleoperar el robot y tendrá información sobre el estado del robot en tiempo real. Las pruebas del robot se las realizarán en el Laboratorio de Telecomunicaciones de la Universidad Tecnológica Israel, la investigación tendrá un periodo de tiempo de investigación y desarrollo de 6 meses donde se adjuntará el cronograma de trabajo.

Desde el punto de vista de los conceptos teóricos en esta investigación se fundamentará en las concepciones de autores reconocidos dentro del ámbito de la teleoperación, desarrollo de software y robótica. Estos extraídos de la biblioteca virtual ProQuest Ebook Central de la Universidad Tecnológica Israel y revistas científicas relacionadas con el presente proyecto.

Esta investigación será estructurada en 3 capítulos que son: capítulo 1 es el marco teórico el cual consiste en desarrollar la teoría que va a fundamentar el proyecto con base al planteamiento del problema que se ha realizado. Capítulo 2 es la descripción del proceso investigativo en la cual la metodología es una de las etapas en que se divide la realización de un trabajo en fases y

de esta manera, la metodología de investigación elegida es la que va a determinar la manera como se va a recabar, ordenar y analizar los datos obtenidos. Capítulo 3 se describe la propuesta, el producto, la aplicación, estudio de casos y análisis de resultados. Finalmente se darán las conclusiones o discusiones y recomendaciones sobre trabajos futuros.

CAPÍTULO I

1. MARCO TEÓRICO

1.1. Sistemas integrados en tiempo real

1.1.1. Descripción Sistemas integrados en tiempo real

Los sistemas integrados en tiempo real se describen como tecnologías y soluciones digitales que tienen como características principales: la comunicación, los sistemas de información, integración de sistemas, internet y movilidad, estos puntos son descritos por (Gavilán, 2019) quien afirma que: "(…) por la función de comunicación se consigue trasladar una información digital entre dos puntos (dos máquinas) distantes". (p. 49)

> Los sistemas de información son un conjunto de hardware y software que proporcionan una funcionalidad, una solución, pero en el fondo, se tiende a centrar en el software porque es el elemento diferencial, lo que hace a un sistema realmente distinto de otro (p. 53).
>
> La integración de sistemas, si solo se tratase de relacionar dos sistemas el problema sería de moderadas dimensiones: se establece esa interfaz y punto. Pero cuando hablamos de decenas o centenas de sistemas diferentes se genera una explosión combinatoria de interfaces que reclama una solución más global. Esa solución es SOA (*Service Oriented Architecture*) (p. 65).
>
> En SOA cada sistema expone dos cosas: unos servicios y un modelo de información de intercambio, en las arquitecturas SOA actuales, normalmente los servicios se implementan mediante los denominados *Web Services*, donde se utilizan protocolos de intercambio propios de Internet HTTP (*HyperText Transfer Protocol*), la información de intercambia en formato XML (*Extensible Markup Language*) o JSON (*JavaScript Object Notation*) y el modelo de información puede reflejarse en los denominados esquemas XML, no se profundizará en el significado de estos términos técnicos porque no es preciso realmente para entender la importancia de SOA (p. 67).
>
> Internet se ve, pues, como una gran nube, a la cual se conectan tanto los equipos de usuarios, como ordenadores personales o smartphones, como servidores o redes locales completas. Sin que el usuario y ni siquiera las empresas deban conocer el

detalle, Internet sirve como un mecanismo de interconexión entre millones de equipos usando protocolos comunes tales como el TCP/IP (p.75).

La movilidad uno de los rasgos más distintivos del panorama digital actual es la creciente supremacía de lo móvil, a medida que las comunicaciones móviles han potenciado las comunicaciones de datos con mayor ancho de banda (4G o LTE) o próximamente la tecnología 5G, estos equipos hoy en día denominamos smartphones, se utilizan como un ordenador personal y tienen capacidad para ejecutar aplicaciones, navegar por Internet, almacenar datos y las interfaces de usuario más centradas en lo táctil y la voz, dada la dificultad de los teclados en el tamaño de los equipos. (p. 83)

Para contrastar la descripción de los sistemas integrados en tiempo real (Garrell & Agüella, 2019) cita que:

Si se dispone de un dispositivo con conexión a internet se puede saber, desde cualquier punto del planeta donde se encuentre, en qué horario tiene un transporte público, reservar una plaza en un hotel, ver las obras maestras de un museo prestigioso, escuchar un concierto de música, leer una tesis doctoral de un científico, teleoperar un dispositivo o máquina desde la otra punta del mundo y todo eso en tiempo real. (p. 29).

Otro factor esencial que ha hecho posible la integración de sistemas es la espectacular progresión de las telecomunicaciones. A las telecomunicaciones por cable se han añadido las comunicaciones por satélite, las diferentes conectividades inalámbricas para cortas distancias, las progresivas ampliaciones de las amplitudes de banda suministradas y, por supuesto, la expansión territorial a cada vez más puntos del planeta, a pesar de que la telefonía móvil fue creada a finales de los años setenta para comunicación de solo voz, la tecnología ha evolucionado hasta posibilitar que un teléfono celular pueda actuar con cualquier tipo de datos digitales: voz, datos, audiovisuales, televisión, geolocalización. La telefonía inalámbrica ya lleva cuatro generaciones y va por la quinta (5G) (p. 33).

Con base en los autores citados los sistemas integrados en tiempo real se describen como una tecnología para manejar varios procesos y vincular cada uno de ellos mediante el

intercambio de información en tiempo real desde 2 puntos distantes a través de la combinación entre hardware y software, donde el software puede estar basado en arquitecturas actuales como el SOA ya que es una solución global para la integración combinatoria de diferentes tipos de interfaces; y el hardware que son los dispositivos móviles, ordenadores o máquinas que están conectados a la red de Internet, los cuales pueden estar usando protocolos comunes como el TCP/IP.

1.1.2. Aplicaciones de los Sistemas integrados en tiempo real como transformación digital

Las aplicaciones de estos sistemas integrados en tiempo real en la actualidad marcan tendencias en el mundo digital, y así lo plantea (Garrell & Agüella, 2019) y menciona que:

> Los pilares del progreso tecnológico, muchos de los avances tecnológicos que constituyen la base de la industria 4.0, y que se detalla en la figura 1, que transformará dramáticamente la producción: las celdas aisladas y poco optimizadas se convertirán en un flujo de producción totalmente integrado, automatizado y optimizado, y llevarán la fábrica a una mayor eficiencia y productividad. Las interrelaciones tradicionales entre proveedores, productores y clientes experimentarán cambios importantes, así como las relaciones entre humanos y máquinas (p. 50).

Figura 1: Las tecnologías de la Industria 4.0. Fuente: Mario Prieto (2017). Obtenido de: https://smart-lighting.es/industria-4-0-mejorar-eficiencia-equipos-industriales/

Para este trabajo de investigación se describirán las tecnologías que se van a usar para el desarrollo de la teleoperación de un brazo robótico, tales como el *cloud computing*, internet de las cosas (IoT), internet industrial de las cosas (IIoT), robots industriales, en lo cual (Gavilán, 2019) dice lo siguiente:

> El *Cloud Computing* (computación en la nube), no es más que trasladar los servidores con las aplicaciones corporativas desde máquinas aisladas en las empresas o en centros de proceso de datos corporativos a unos centros de procesos de datos ofrecidos por terceros y a los cuales se accede a través de Internet, a través de la nube. Es simple de entender como se muestra en la figura 2, pero el *cloud computing* trae consigo dos importantes cambios de paradigma en la gestión de los sistemas de información de una empresa:
>
> - Por un lado, las aplicaciones pasan a estar no en la propia empresa, sino en centros de proceso de datos fuera de la empresa y operados por un tercero, un proveedor de servicios.

- Por otro, se pasa de un modelo de inversión/compra de sistemas a otro de pago por uso. De un modelo de propiedad del software a otro de acceso como servicio (p. 89).

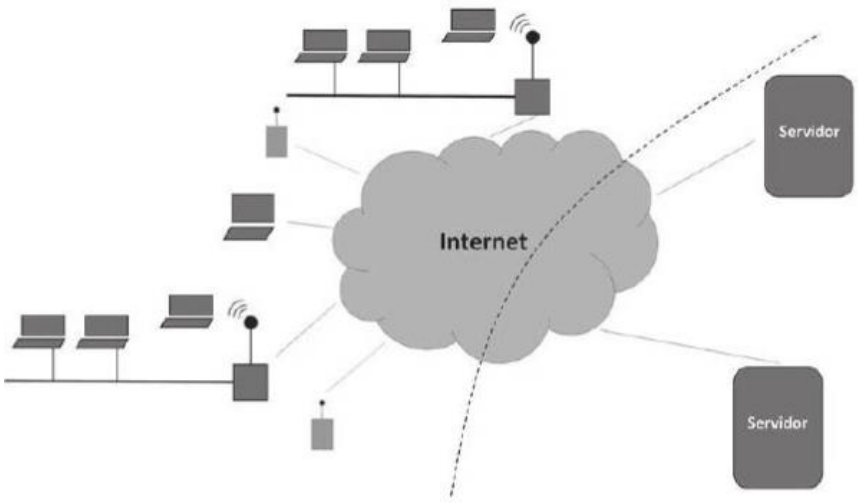


Figura 2: Entorno complejo con varios servidores. Fuente: Gavilán, Ignacio G. R. (2019). La carrera digital, ExLibric, 2019. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/uisraelsp/detail.action?docID=5810034.

Del Internet de las personas al Internet de las cosas (Gavilán, 2019), dice que:

> El internet que hemos conocido hasta ahora es una compleja red en cuya periferia, en los puntos de acceso a Internet, se sitúan fundamentalmente ordenadores, tablets y smartphones, unos equipos con los cuales trabajan las personas, que son las que, en último término, utilizan Internet para comunicarse. En Internet de las cosas lo que se encuentra en la periferia de la red y dialoga ya no son personas, sino dispositivos, equipos, máquinas. Así, por ejemplo, en Internet de las cosas entran en diálogo aparatos de medida (electricidad, agua, gas, etc.), electrodomésticos y equipos domóticas, automóviles, robots entre otros dispositivos (p. 140).

Con respecto al internet industrial de las cosas y los brazos robóticos industriales (Garrell & Agüella, 2019) dice que:

> Los brazos robóticos industriales, provistos de sensores y manipuladores que pueden hacer tareas muy variadas según los programas de control alojados en los ordenadores que los controlan, el abaratamiento de los microprocesadores y la profusión de sensores de todo tipo a precios asequibles ha hecho posible que sean cada vez más las empresas que ofrecen microrobots y diferentes componentes de robótica para automatizar pequeños talleres, vehículos de exploración, o pequeños brazos industriales. El descenso espectacular de los precios ha hecho posible que también se fabriquen un buen grupo de robots destinados a la formación, a la investigación o al simple entretenimiento (p. 56).

> Actualmente, solo algunos de los sensores y máquinas de los procesos de producción están conectados en red y hacen uso de la informática integrada, el internet industrial de las cosas (IIoT), se añade la posibilidad de descentralizar partes del control de los procesos, de interconectar mediante tecnologías estándar muchos más dispositivos y productos de forma que se enriquezca la informática integrada. Esto permite que los dispositivos de campo se comuniquen e interactúen entre ellos y con más controladores descentralizados, según sea necesario como se puede observar en la figura 3. Descentralizar el análisis y la toma de decisiones permite tener mejores respuestas en tiempo real. (p. 66)

Figura 3: Esquema del Internet en la Industria 4.0. Fuente: Tulip (2019) *The Ultimate Guide to Industrial IoT for Manufacturers.* Obtenido de: https://tulip.co/resources/iiot-for-manufacturers/

En las aplicaciones de los sistemas integrados en tiempo real, menciona algunas tecnologías en las cuales describe las grandes ventajas que ofrece y se menciona las más importantes, la interacción entre humano máquina cada vez más eficientes, el Cloud Computing cambia a un nuevo paradigma para la gestión de la información administrada por terceros y se accede a los procesos corporativos a través de internet y la nube, el internet de las cosas es una arquitectura de red que integra a los ordenadores, tablets y smartphones que trabajan con las personas y con ello se puede interactuar con electrodomésticos y equipos de domótica, automóviles, robots entre otros dispositivos, cabe recalcar que se diseñan e implementan un buen grupo de dispositivos que son destinados a la formación, a la investigación o al entretenimiento.

1.2. Sistemas robóticos teleoperados

1.2.1. Descripción de los sistemas robóticos teleoperados

Los sistemas robóticos teleoperados actualmente están aplicados en varias áreas tales como la industria, salud, investigación y así lo describe (Barrientos, Peñín, & Balaguer, 2007) y plantea que,

> Las tecnologías de teleoperación tiene una apertura a nuevos campos de aplicación, como el espacial, la cirugía, nuclear y submarino, con la ayuda de nuevos desarrollos en computadores digitales y la integración de tecnologías propias de la robótica y la inteligencia artificial. Se han desarrollado, además, nuevos elementos que mejoran la eficiencia de la manipulación, como pueden ser novedosos diseños de dispositivos de entrada y nuevas técnicas de control, como puede ser el control supervisado. La teleoperación, por tanto, puede considerarse hoy en día como una tecnología madura que forma parte de la robótica, aunque sus orígenes fueran bien distintos. En ambas tecnologías el objetivo fundamental es el realizar el control de un brazo articulado de varios grados de libertad, con la diferencia que en la teleoperación es el operador, y no un programa, el que comanda en todo momento al manipulador, con la posibilidad de que existan diversos modos de acoplamiento entre ambos (p. 483).

> Teleoperación y telemanipulación es la acción que realiza un ser humano al operar o gobernar a distancia un dispositivo. La telerrobótica es un conjunto de

tecnologías que comprenden la monitorización y reprogramación a distancia de un robot por un ser humano. La telepresencia es una situación o circunstancia que se da cuando un ser humano tiene la sensación de encontrarse físicamente en un lugar remoto. La telepresencia se consigue proporcionando coherentemente al ser humano suficiente cantidad de información sobre el entorno remoto. Elementos de un sistema de teleoperación se muestra en la figura 4 que al articularse permitan su correcto funcionamiento. (p. 487)

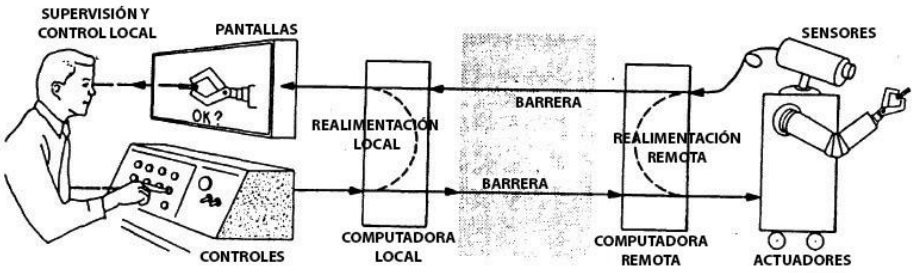


Figura 4: Elementos básicos de un sistema de teleoperación. Fuente: Emmanuel Nuño Ortega, "Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente" (2004, p.6)

Los componentes de un Sistemas robóticos teleoperados lo cita (Cerón, 2009) en su artículo y menciona que:

> Un sistema teleoperado se compone principalmente de una estación de teleoperación, un sistema de comunicación y esclavo, el esclavo puede ser un manipulador o un robot móvil equipado con un manipulador ubicado en un entorno remoto. La estación de teleoperación permite controlar al esclavo a distancia por medio del sistema de comunicación, el cual permite transmitir las señales de control hacia el esclavo y, a su vez, recibir señales de información sobre el estado de éste en la estación de teleoperación a través de un canal de comunicación que puede ser una red de computadores, un enlace de radio frecuencia o microondas (p. 3).

Con respecto al sistema de control de los sistemas robóticos teleoperados (Barrientos, Peñín, & Balaguer, 2007), habla de:

La importancia del operador en un sistema de teleoperación radica, entonces, en que cierra el bucle de control del sistema global. El operador actúa como un controlador, generando señales de actuación sobre el dispositivo de control a partir de la realimentación visual de información, y en su caso de fuerza, procedente de la zona remota con se ve en la figura 5. (p. 507)

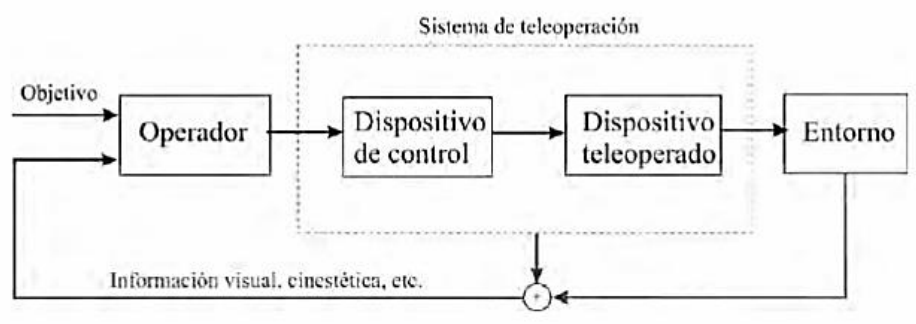


Figura 5: El operador como controlador del sistema de teleoperación. Fuente: Barrientos, Antonio, et al. Fundamentos de robótica (2a. ed.), McGraw-Hill España, 2007. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/uisraelsp/detail.action?docID=3199830.

Control de sistemas con retardo temporal, la existencia de retardos en la comunicación supone uno de los mayores problemas en la estabilidad de los sistemas de teleoperación, es decir, que se tarde un determinado tiempo en realizar la operación.

Estrategias para contrarrestar el retardo temporal, la solución más inmediata es adoptar una estrategia de mover y esperar. A continuación, se presentan las técnicas más habituales para contrarrestar los efectos del retardo temporal:

- Acomodación activa remota: consistente en modificar la referencia de posición del manipulador mediante cálculos por computador que reciben información sobre las fuerzas de contacto de un sensor fuerza/par situado en el extremo del manipulador como se puede observar en la figura 6.

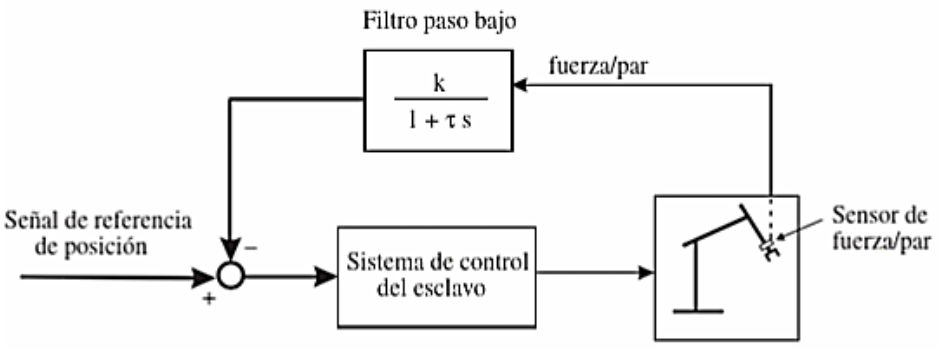


Figura 6: Esquema de implementación de la acomodación activa remota. Fuente: Barrientos, Antonio, et al. Fundamentos de robótica (2a. ed.), McGraw-Hill España, 2007. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/uisraelsp/detail.action?docID=3199830.

- Visualizadores predictivos: sistemas de visualización en los que se muestra al operador en tiempo real mediante simulación el resultado de las acciones que realiza, sin que tenga que esperar a recibir la información visual del resultado real. Para ello es necesario tener un modelo preciso de la zona remota (dispositivo teleoperado más entorno) y realizar las respectivas extrapolaciones (simulaciones) del mismo en el tiempo.
- Embragues temporales y espaciales: consisten en un refinamiento más de los visualizadores predictivos. En el caso del embrague espacial, los movimientos que realiza el operador, cuyo resultado ve mediante el visualizador predictivo, no son inmediatamente enviados a la zona remota, sino que se comprueba su resultado y si no es el adecuado el operador puede repetir la operación cuantas veces quiera, hasta que, una vez satisfecho, mande los comandos adecuados a la zona remota. En el caso del embrague temporal, el operador puede salir de sincronismo con la operación real, y con ayuda del visualizador predictivo acelerar el proceso en las operaciones fáciles y frenarlo en las difíciles. Incluso se podría realizar la prueba de distintas alternativas. El sistema guarda los datos y los envía a la velocidad adecuada al sistema real (p. 521).

El control supervisado y tele-programación, (Barrientos, Peñín, & Balaguer, 2007) explica que:

> Se entiende por control supervisado cuando uno o más operadores se encuentran programando y recibiendo intermitentemente información del entorno remoto en el que un bucle de control autónomo controla al sistema teleoperado en función de los comandos del operador y la tele-programación consiste en mezclar la programación convencional de robots con los conceptos de teleoperación, y se puede considerar como un tipo de control supervisado. Un operador realiza la tarea de teleoperación en un entorno local, con un bucle local de control, utilizando ya sea un simulador predictivo de la zona remota o una maqueta o representación física del mismo. El sistema analiza la tarea y de la misma infiere una serie de comandos de alto de nivel a ser ejecutados por el sistema teleoperado para replicar la ejecución de la tarea en la zona remota. Cuando la medida de los sensores reales detecta grandes discrepancias con el modelo local que ha servido para programar la tarea, el sistema se para y da paso al control del operador. Por supuesto, la tarea de teleoperación local puede, y debe, contar con reflexión de fuerzas sobre el operador para aumentar su efectividad. (p. 548).

1.3. Plataforma Firebase

Las características sobre la plataforma Firebase (Ruiz, 2017) dice lo siguiente:

> Según su propia definición, Firebase es un conjunto de herramientas orientadas a la creación de aplicaciones de alta calidad, al crecimiento de los usuarios y a ganar más dinero. Se describe la plataforma como una suite de diferentes aplicaciones que harán más fácil el desarrollo de las aplicaciones. Firebase permite programar aplicaciones compatibles con Android, iOS, Java Script, Node JS, Python, C++, Unity. El logo de la Plataforma Firebase como se ve en la figura 7.

Figura 7: Plataforma Firebase. Fuente: Sitio oficial de Firebase. Obtenido de: https://firebase.google.com/docs

Los servicios que ofrece Firebase en la nube para integrarse con las aplicaciones son:

1.3.1. Cloud Firestore

Cloud Firestore es una base de datos flexible y escalable, usa una base de datos NoSQL en la nube a fin de almacenar y sincronizar datos para la programación en servidores (cliente/ servidor), dispositivos móviles y la Web desde Firebase y Google Cloud Platform. Al igual que Firebase Realtime Database, mantiene tus datos sincronizados entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet. Cloud Firestore también ofrece una integración sin interrupciones con otros productos de Firebase y Google Cloud Platform, incluido Cloud Functions ver la figura 8.

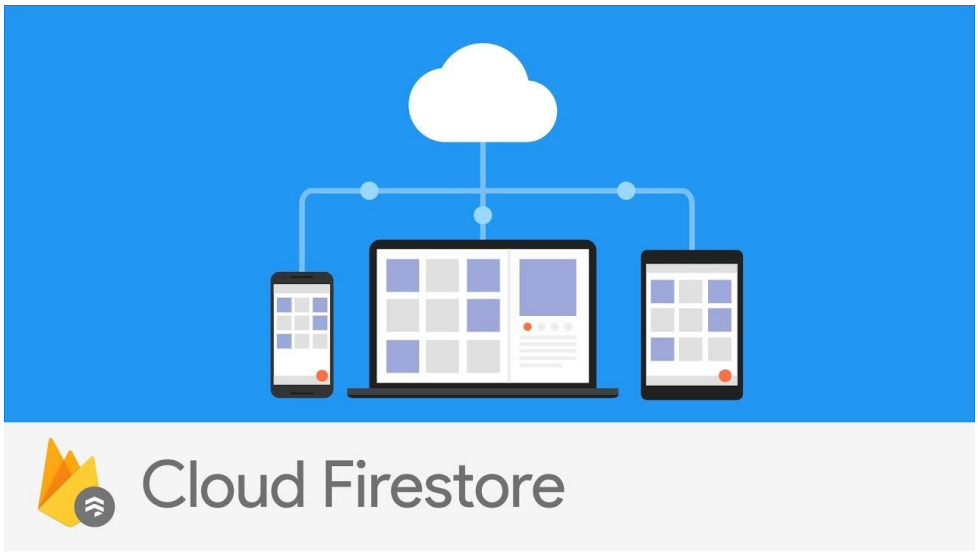


Figura 8: Firebase Cloud Firestore. Fuente: Sitio oficial de Firebase. Obtenido de: https://firebase.google.com/docs/firestore/

1.3.2. Realtime Database

Con la base en tiempo real de Firebase (Ruiz, 2017) añade que:

> Se podrá guardar todos los datos que se requiera en la aplicación, ya que permite actualizar los datos en los componentes automáticamente. Los datos se almacenan en formato JSON y se pueden agregar reglas para permitir requests con token o solo desde una URL.

1.3.3. Authentication

Con respecto a la herramienta Authentication (Firebase, 2019), describe que:

> La mayoría de las apps necesitan identificar a los usuarios. Conocer la identidad de un usuario permite que una app guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos del usuario.
>
> Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios en tu app. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook y Twitter, y mucho más.
>
> Firebase Authentication se integra estrechamente en otros servicios de Firebase y aprovecha los estándares de la industria como OAuth 2.0 y OpenID Connect, por lo que se puede integrar fácilmente con tu backend personalizado como se ve en la figura 9.

Figura 9: Firebase Authentication. Fuente: Sergio Mesa (2019). Obtenido de: https://medium.com/@mesasergio/creaci%C3%B3n-de-componente-autenticaci%C3%B3n-y-registro-con-angular-firebase-2-92c8926fd09d

1.3.4. Crashlytics

Firebase Crashlytics es una herramienta liviana que permite informar fallas en tiempo real, hacer un seguimiento de los problemas de estabilidad que afectan la calidad de tu app, priorizarlos y corregirlos. Crashlytics agrupa las fallas de forma inteligente y destaca las

circunstancias en las que se produjeron, lo que te permite ahorrar tiempo en la solución de problemas.

Descubre si una falla específica afecta a muchos usuarios. Recibe alertas cuando la gravedad de un problema aumenta repentinamente. Determina qué líneas de código están provocando fallas.

1.3.5. Performance Monitoring

Firebase Performance Monitoring es un servicio que permite obtener estadísticas sobre las características de rendimiento de las apps web, para iOS y Android.

Usa el SDK de Performance Monitoring para recopilar datos de rendimiento de tu app y, luego, revisa y analiza esos datos en Firebase console. Performance Monitoring te ayuda a comprender dónde y cuándo se puede mejorar el rendimiento de tu app, de manera que puedas usar esa información para solucionar problemas de rendimiento.

1.3.6. Cloud Messaging

Con FCM, puedes notificar a una app cliente que un correo electrónico nuevo o que otros datos están disponibles para la sincronización. Puedes enviar mensajes de notificación para volver a atraer a más usuarios y aumentar su retención. Para los casos prácticos de mensajería instantánea, un mensaje puede transferir una carga de hasta 4 KB a una app cliente.

1.4. Desarrollo de aplicaciones en dispositivos móviles

El desarrollo de aplicaciones móviles según (Santacruz, 2014), describe que:

> Las aplicaciones móviles no son aplicaciones de escritorio adaptadas para dispositivos con pantallas pequeñas, son por el contrario, aplicaciones diferentes por varias razones: (i) la capacidad para comunicarse desde cualquier lugar cambia la interacción del usuario con la aplicación, (ii) la interfaz de usuario para una pantalla y teclados pequeños difiere de forma significativa de la interfaz de una aplicación diseñada para un ordenador de sobremesa o un portátil, (iii) los tipos de canales de comunicación son diferentes, los dispositivos móviles incorporan capacidades de voz, mensajería, información de geolocalización y vídeo conferencia (en algunos teléfonos). Las mejores aplicaciones para móviles integran estas capacidades para optimizar la interacción del usuario con los datos y, por último, (iv) la naturaleza de las redes inalámbricas, aunque las redes ofrecen capacidades de datos de banda ancha, estas pueden variar, dependiendo de la

calidad de la señal y de la disponibilidad de conexión de la red, en particular si se trata de usuarios móviles. (p. 15)

Las nuevas tendencias con respecto a las comunicaciones de móviles (Agusti, Bernardo, & Casadevall, 2010) menciona que:

> La cuarta generación 4G con tecnologías como: Long Term Evolution (LTE), LTE Avanzado, LTE Advanced (LTE-A), High Speed Packet Access (HSPA+) y el estándar IEEE 802.16; ofrecen servicios basados completamente en el Protocolo de Internet (IP), con velocidades de transferencia hasta de 100 Mbps y con Calidad de Servicios, Quality of Service (QoS) (p. 377).

Paralelamente dentro del desarrollo de aplicaciones móviles (Gasca, Camargo, & Medina, 2014), dice que:

> El crecimiento de las redes, la evolución de los teléfonos móviles ha conllevado a la integración de diversas tecnologías a estos dispositivos, tecnologías como WiFi, Bluetooth, GPS, infrarrojo, touchscreen, USB, entre otras. Esto ha permitido que el teléfono celular sea compatible con una amplia gama de dispositivos y pueda sincronizarse con otros equipos para el intercambio de información (p. 22).

1.4.1. Diseño de aplicaciones para dispositivos móviles

El diseño de aplicaciones para dispositivos móviles (Santacruz, 2014) considera cuatro enfoques principales que se presentan en la siguiente tabla:

Tabla 1: Beneficios e inconvenientes de los diferentes enfoques para el desarrollo de aplicaciones para dispositivos móviles

Arquitectura	Beneficios	Inconvenientes
Cliente Nativo	- Aplicaciones sofisticadas y control sobre el entorno local. - Capacidades multitarea sobre muchas plataformas.	- El más alto nivel de esfuerzo de desarrollo. - Diferente código base para diferentes dispositivos.

Arquitectura	Beneficios	Inconvenientes
Java ME (*Java Platform Micro Edition*)	- El mismo código base puede soportar múltiples aplicaciones. - Aplicaciones más sofisticadas.	- Algunos límites de capacidad (no multitarea). - Requiere prueba y adaptación para las plataformas.
Navegador Web	- Desarrollo rápido - No requiere mantenimiento del código cliente. - Métodos Web 2.0 disponibles. - Trabajo con un amplio rango de dispositivos móviles.	- Menos sensible y capas que las aplicaciones nativas.
Midleware Móvil	- Alto nivel de capacidad con reducido esfuerzo de desarrollo.	- Tasas adicionales de licencia. - Curva de aprendizaje y esfuerzo de integración potencialmente grande.

Nota. Tabla adaptada por (Santacruz, 2014)

1.4.2. Tecnologías para dispositivos móviles

Sobre la tecnología para dispositivos móviles (Gasca, Camargo, & Medina, 2014) menciona que (…) "En la actualidad, la mayoría de los servicios móviles están desarrollados en: HTML 5, WAP, Java 2 Micro Edición (J2ME), C#, Silverlight, .NET, entre otros. También, en aplicaciones nativas para los Sistemas Operativos de los móviles (S.O. del móvil), como: Android, Symbian, iOS y MeeGo" (p. 22).

Las principales tecnologías y sistemas operativos para dispositivos móviles que hay actualmente según (Santacruz, 2014), informa que:

> Android es una plataforma formada por un conjunto de software en estructura de pila (*software stack*) que incluye un sistema operativo, software para conectar aplicaciones (*middleware*) y aplicaciones base. El SDK (*Software Development Kit*) de Android proporciona varias herramientas y API (*Applications Programming Interface*) que son necesarias para desarrollar aplicaciones Android. Estas aplicaciones se desarrollan en lenguaje Java. Android está desarrollado por OHA (*Open Handset Alliance*), una agrupación de 78 compañías para desarrollar estándares abiertos para dispositivos móviles y que está liderada por Google (p. 19).
>
> Palm OS ha tenido una gran evolución de versiones desde la 1.0 hasta la 5.0, luego pasó a llamarse Palm OS Cobalt. Este último es un sistema operativo basado en Linux, que evolucionó para denominarse webOS en 2009, el primer dispositivo que lo incluía fue Palm Pre. La compañía Palm Inc, fue adquirida en 2010 por HP y, actualmente, HP utiliza webOS en sus dispositivos móviles y tablets, como Pixi, Veer y HP TouchPad.
>
> A mediados de 2007 la tecnología Apple ofreció el iOS (inicialmente llamado IPhone OS), desarrollado originalmente para el IPhone y con él, una nueva definición del teléfono móvil. Más tarde fue introducido en el iPod Touch y actualmente en el IPad. Evolucionaron las versiones, las cuales incluían el Spotlight (para realizar búsquedas en el dispositivo), también ofrecía la posibilidad de incluir la API de Google Maps, las operaciones de copiar/cortar/pegar, interconexión por BlueTooth o P2P y librerías GPS, con una interfaz mejorada y funcionalidades como la presencia de un asistente personal Siri, facilidades para la sincronización

sin cables, un centro de notificaciones mejorado, el servicio de iMessage, la navegación web con pestañas, entre otras (p. 22).

En la tabla 2 se resalta las características de los sistemas operativos como se puede observar

Tabla 2: Sistemas operativos de dispositivos móviles

Característica	**Symbian**	**Windows Phone**	**IPhone**	**Palm**	**Android**	**BlackBerry**
Versión	Nokia Belle	Windows Phone 8 o Apollo	iOS 10.15.2	Garner OS 5.5	Android 9 Pie	OS 7.0
Dispositivos	Smartphone	Smartphone PDA	IPhone	PDA	Smartphones	Smartphones
Interfaz	Apuntador Teclado	Apuntador Teclado	Touch	Apuntador Teclado	- Touch - Apuntador Touch	Apuntador Teclado
SDK	Gratuito	Gratuito/IDE pagado	IPhone Delevoper Program	Gratuito	Gratuito	Gratuito
Aplicaciones	Nativas y JME	Nativas Compact Framework	Nativas y JME	Nativas y JME	Nativas y JME	JME
Firma	Obligatoria	Opcional	Obligatoria	Opcional	Opcional	Opcional

Característica	Symbian	Windows Phone	IPhone	Palm	Android	BlackBerry
Proveedores	Nokia, Sony Ericsson, Samsung, Siemens.	HP, HTC, Samsung, Dell.	IPhone	Familia Palm	HTC, LG, Samsung	Familia BlackBerry

Nota. Tabla adaptada por (Santacruz, 2014)

1.4.3. Plataformas de desarrollo de aplicaciones móviles

Los lenguajes para programación de aplicaciones móviles se pasarán a presentar las herramientas de desarrollo y compilación, (Santacruz, 2014) describe que:

> El desarrollo de aplicaciones Android se realiza con un grupo de herramientas que son suministradas en el SDK. La utilización de este grupo de herramientas puede ser de dos formas: utilizando un Entorno de Desarrollo Integrado (IDE) en combinación con un plugin llamado ADT (*Android Development Tools*) o bien desde la línea de comandos. Se puede utilizar cualquier IDE, actualmente lo más común es usar Android Studio. Algunas de las herramientas más importantes de línea de comandos son:
>
> - Android: crea y actualiza proyectos Android y crea y elimina AVD.
> - Android Emulator: ejecuta aplicaciones en una plataforma Android emulada.
> - Android Debu Bridge: es una interfaz para conectar la aplicación con un emulador o con un móvil Android. Permite instalar aplicaciones, ejecutar comandos en línea, etc.
> - Ant: permite compilar y construir proyectos generando el fichero. apk instalable de la aplicación.
> - Keytool: permite generar una clave privada que se usa para firmar y autenticar el fichero.apk. Esta herramienta forma parte del JDK.

- Jarsigner: es una herramienta para firmar el fichero.apk con una clave privada generada con la herramienta. Keytool. Esta herramienta también forma parte del JDK (p. p. 38).

Para obtener el SDK es necesario inscribirse en el iOS Dev Center (el sitio oficial de desarrolladores de Apple), desde donde se puede descargar gratuitamente. Si una vez creada la aplicación se quiere comercializar, es necesario enviarla al App Store, lo cual requiere registrarse en el programa de desarrolladores de Apple y pagar una cuota de $99. El desarrollador fija el precio de la aplicación y obtiene el 70% de las ventas realizadas a través del App Store. Un dispositivo Apple solo ejecuta aplicaciones firmadas por Apple, por lo que después de asignado el permiso de desarrollador, Apple expide un certificado que permite hacer pruebas en nuestro iPhone (p. 43).

1.4.4. Aplicaciones en dispositivos móviles ventajas y desventajas

En el ecosistema de aplicaciones móviles, (Ramírez, 2013) lista las siguiente ventajas y desventajas:

Muchas de las aplicaciones aprovechan varias capacidades. Por ejemplo, una aplicación de realidad aumentada aprovecha varias de ellas:

- GPS, para conocer la posición geográfica y saber qué se debe mostrar.
- Brújula, para saber la orientación actual
- Acelerómetro, para saber cuál es la orientación exacta de nuestro dispositivo superponer las capas.
- Cámara, para poder captar nuestro alrededor y así ampliar la información (en ocasiones, incluso, varias cámaras).
- Conexión a Internet, para poder obtener la información para ampliar nuestra realidad. Esta conectividad puede venir mediante Internet móvil o WiFi, entre otros.
- Capacidad de procesamiento gráfico muy mejorada, con chips de aceleración gráfica potentes

Se tiene una serie de beneficios relacionados con la ubiquidad en lo que respecta a las aplicaciones para dispositivos móviles:

- El dispositivo móvil es el primer medio de comunicación masivo real, dado que es capaz de llegar a casi todos los usuarios y en todo momento.
- El primer medio de comunicación permanentemente encendido, pues es capaz de captar y enviar información aun cuando está apagado (apagado en el sentido del usuario, es decir, cerrado, pero no totalmente apagado).
- Primer medio que está siempre con el usuario.
- Medio masivo que tiene incorporado un sistema de pago, que en este caso es mediante la operadora.
- Acceso total al contexto, con todas las posibilidades que eso conlleva. Consigue las mejores experiencias de usuario.
- Posibilidad de gestión de interrupciones en la aplicación o en las capacidades del dispositivo. Desde saber si tenemos conexión de datos o conexión de localización hasta tener información sobre la batería.
- Son relativamente fáciles de desarrollar si solo se contempla una plataforma.
- Se pueden distribuir por los canales conocidos de aplicaciones que permita la plataforma, con lo que se pueden vender más fácilmente.
- Todas las novedades llegan primero a este tipo de aplicaciones, pues es en este tipo de aplicaciones donde se prueban.

Como desventajas se producen:

- Limitación de la vida de las baterías. Esto significa que, al utilizar muchas de estas capacidades (por ejemplo, el GPS), hacemos que nuestro terminal pierda autonomía, pues se necesita destinar energía a estos periféricos, y lo mismo sucede con el resto de capacidades. En la materia que incumbe, el desarrollo de aplicaciones, tenemos que tener esto muy presente a la hora de diseñar y escribir nuestras aplicaciones, ya que puede afectar mucho a su rendimiento.
- Vulneración de la privacidad. Nuestros dispositivos móviles contienen cada vez más información personal y confidencial, y requieren acceso a esta información para poder sacarle el mayor partido y conseguir integrarse en el contexto. Por eso, siempre que realicéis una aplicación que requiera acceso a

información o acceda a datos de otras fuentes (como las redes sociales), debéis tener el permiso explícito del usuario, y este se debe poder revocar.

- Necesidades de hardware. Por ejemplo, la necesidad de mayor velocidad de transmisión. Si se encuentra en una zona con poca cobertura para nuestra red de transmisión de datos, puede ocurrir que nuestras aplicaciones no funcionen correctamente.
- Necesidades de inversión no previstas debido a novedades del mercado. Si aparece una fuente de fragmentación nueva (por ejemplo, un nuevo dispositivo), debemos invertir en dar soporte a ese nuevo dispositivo. Esta inversión puede suponer no tener que hacer nada o realizar un desarrollo nuevo.
- Portar aplicaciones es costoso. En el caso de querer realizar una aplicación para más de una plataforma, se complica el desarrollo, debido a los problemas de la fragmentación.
- Dependiendo de la plataforma elegida, puede haber fragmentación dentro de cada plataforma, debido a los diferentes tipos de dispositivos o versiones de la plataforma.
- No existe un estándar, por lo que cada plataforma ofrecerá sus peculiaridades.
- Normalmente, para desarrollar, distribuir o probar estas aplicaciones en dispositivos reales, es necesario tener una licencia de pago, dependiendo de la plataforma.
- Las ganancias por estas aplicaciones suelen repartirse entre el creador de la aplicación y la plataforma de distribución.

En este capítulo se explicó todo el marco teórico sobre los sistemas integrados en tiempo real en la cual se describió las tecnologías de la industria 4.0 como es el *cloud computing*, el internet de las cosas (IoT), cabe recalcar que son procesos que trabajan en la red de Internet y usan la informática integrada. Se habló de la teleoperación y telemanipulación que es la acción de operar un dispositivo de forma remota. La telerrobótica es la monitorización y reprogramación de un robot a distancia. Luego se describió las características de la plataforma Firebase la cual es un conjunto de herramientas que se encuentran integradas en la nube. Sobre

la plataforma Firebase se puede desarrollar cloud computing y esto hará más fácil el desarrollo de las aplicaciones. Firebase permite programar aplicaciones compatibles con Android, iOS, Java Script, Node, Python, C++, Unity. Sobre el desarrollo de aplicaciones móviles se ha proporcionado información sobre los sistemas operativos con los que trabajan los dispositivos móviles.

CAPÍTULO II

2. DESCRIPCIÓN DEL PROCESO INVESTIGATIVO

2.1. Enfoque metodológico de la investigación.

En este capítulo se plantearon los aspectos metodológicos y prácticos que se utilizaron para desarrollar un Sistema Integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles. El tipo de investigación será aplicada, debido a que tiene por objetivo resolver un determinado problema o planteamiento específico, enfocándose en la búsqueda y consolidación del conocimiento para su aplicación y, por ende, para el enriquecimiento del desarrollo cultural y científico. Considerando que la Investigación Aplicada se basa en una necesidad social práctica por resolver, algunos ejemplos de ella corresponden a los siguientes: cómo mejorar las técnicas para otorgar durabilidad a un producto, solución a un problema de producción, ver la figura 10.

En ese sentido, (Vargas, 2009), asegura que este tipo de investigación llamada también "práctica o empírica", se caracteriza:

> … porque busca la aplicación o utilización de los conocimientos adquiridos, a la vez que se adquieren otros, después de implementar y sistematizar la práctica basada en investigación. El uso del conocimiento y los resultados de investigación que da como resultado una forma rigurosa, organizada y sistemática de conocer la realidad. (P. 59)

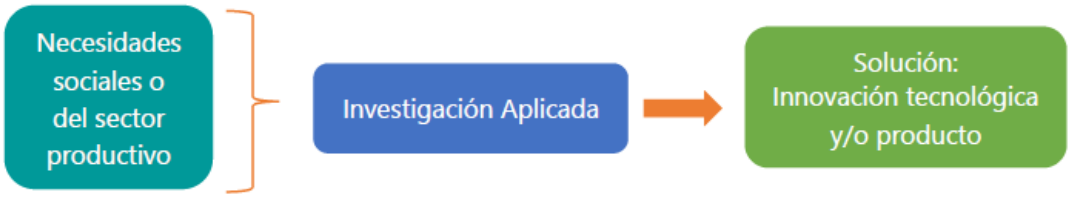


Figura 10: Investigación aplicada. Fuente: Duoc UC (2018). Obtenido de: http://www.duoc.cl/biblioteca/crai/definicion-y-proposito-de-la-investigacion-aplicada

Para el presente trabajo de investigación se dio una innovación tecnológica con respecto a la forma de integración de las nuevas tecnologías de la industria 4.0, las cuales se describieron en

el capítulo anterior, de tal manera que se integre todas estas tendencias en los sistemas de teleoperación. Así, la investigación, tenga como objetivo la aplicación de conocimientos para, reducir costes de implementación y mejorar la calidad y servicio de los sistemas robóticos teleoperados.

2.2. Técnica de recolección de datos

Como técnicas de recolección de datos que se utilizó es observación indirecta según (Rodriguez, 2005), se presenta "cuando el investigador corrobora los datos que ha tomado de otros, ya sea de testimonios orales o escritos de personas que han tenido contacto de primera mano con la fuente que proporciona los datos" (p.98).

Fortaleciendo el concepto de observación indirecta, (Huamán, 2005) menciona que:

> Es indirecta cuando el investigador entra en conocimiento del hecho o fenómeno observando a través de las observaciones realizadas anteriormente por otra persona. Esto ocurre cuando se apoya en libros, revistas, informes, grabaciones, fotografías, etc., relacionadas con lo que estamos investigando, los cuales han sido conseguidos o elaborados por personas que observaron antes lo mismo que nosotros (p. 16).

Tomando en cuenta esta técnica como levantamiento de información la cual es la observación indirecta ya que se realiza un análisis de la situación actual sobre el impacto tecnológico de la industria 4.0 y como se relaciona con los sistemas robóticos teleoperados, a través de sitios web oficiales, foros, revistas y publicaciones.

2.3.Situación actual

En la actualidad, las aplicaciones de producción han avanzado a una escala global, lo que ha permitido que se desarrollen nuevas formas para realizar estos procesos usando nuevas tecnologías, como son aplicaciones en la nube, sistemas teleoperados, internet de cosas, ciberseguridad, sistemas integrados en tiempo real. En Ecuador he observado un progreso lento en este tipo de tecnologías, por tal motivo aportaré mediante el presente proyecto de investigación donde se involucre algunas de estas tecnologías que son parte de la Industria 4.0.

La industria 4.0 en Ecuador (Velásquez, 2018), consultora internacional menciona que:

> Será gradual gracias al avance tecnológico, la creación de ecosistemas y una participación del gobierno, las propias industrias, los gremios y asociaciones vinculados al sector. La Industria 4.0 modifica el proceso de fabricación tradicional

hacia un esquema interconectado entre los clientes, la industria y los proveedores. El ciclo de vida de un producto incluye la interacción inicial del cliente, el diseño, prototipado, desarrollo, producción, entrega y mantenimiento del producto y servicios asociados mediante la producción autónoma, digitalización del ciclo de vida del producto, la utilización de Sistemas Ciberfísicos (CPS), obtención de información en tiempo real por medio de sensores, cada vez más pequeños y a un menor costo, el internet de las cosas, uso de cloud computing, big data, robots trabajando colaborativamente con las personas, realidad aumentada para soporte remoto y especializado, inteligencia artificial, simulación y gemelos digitales, fabricación aditiva, visión artificial y ciberseguridad.

Los sistemas robóticos teleoperados forman parte de los sistemas ciberfísicos de integración los cuales se conectan en tiempo real y trabajan sobre las tecnologías de la industria 3.0. La transformación más profunda se produce por la digitalización y la posibilidad de conectar en tiempo real a todos los actores sociales mediante Internet según se aprecia en la figura 11.

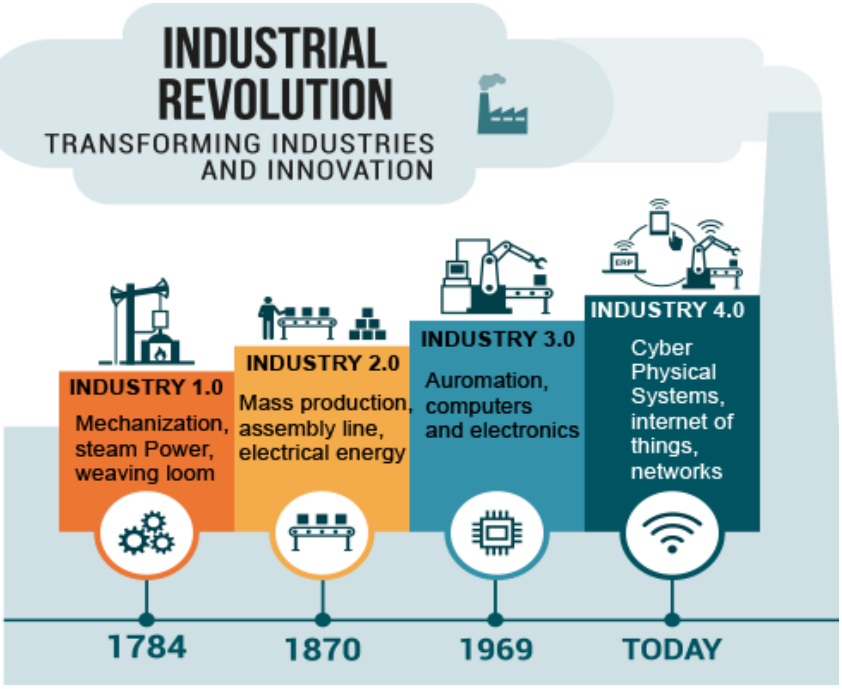


Figura 11: Industria 4.0. Fuente: Nordigi (2019). Obtenido de: https://nordigi.no/index.php/en/blog/24-nordigi-in-industry-4-0

El estudio realizado en Latinoamérica por (Basco, Beliz, & Garnero, 2018), sobre la industria 4.0, menciona que:

> En la transición a ciegas hacia la fábrica inteligente, las empresas gestionan sus actividades con altos niveles de incertidumbre; faltan capacidades para analizar los datos y para tomar decisiones en un contexto competitivo y cambiante. La matriz tecnológica cambia constantemente y de forma acelerada. El ciclo de vida de los productos se acorta considerablemente; algunos bienes caen en la obsolescencia mientras se configuran nuevos mercados de bienes y servicios "donde antes no había nada". La digitalización de la economía cambia las reglas de juego: las empresas tienen cada vez más información sobre sus clientes, pero al mismo tiempo, permite el ingreso repentino de nuevos competidores al mercado. Por lo tanto, se ven desafiadas a enfrentar una competencia creciente y escalable, y a tomar decisiones sobre una enorme cantidad de datos que muchas veces no tienen capacidad de interpretar. Sobre 2.000 directivos de nueve sectores industriales en 26 países, sólo el 20% de las empresas industriales reconoce tener capacidades avanzadas para el análisis de datos. El 51% considera necesario estimular el desarrollo de estas habilidades entre sus recursos humanos para eficientizar el proceso de toma de decisiones y reducir la incertidumbre.
>
> Desigualdad robótica tridimensional: crea, destruye y desplaza empleos. Crece la adopción de robots industriales, pero de forma concentrada en pocos países y en empresas de gran tamaño. La automatización de la producción es una tendencia creciente a nivel mundial; en el período 2010-2016 la producción de robots industriales creció a una tasa promedio anual del 12%, mientras que la dotación de robots industriales cada 10.000 habitantes, pasó de 66 unidades a 74 unidades en el mismo período. El capital robótico se concentra en pocos países ver la figura 12 y en empresas de tamaño grande, siendo la industria automotriz la principal adoptante de esta tecnología a nivel mundial. El uso del 75% de los robots industriales se localiza en cinco países: China, Estados Unidos, Corea, Japón y Alemania, los que, al mismo tiempo, resultan los principales productores de la tecnología. La reciente expansión de las capacidades cognitivas a las máquinas implica que tareas de

complejidad media también pueden ser automatizadas, generando pérdidas de empleo y desplazamiento de trabajadores a nuevas ocupaciones. Las economías más automatizadas, muestran tasas positivas de creación de empleo, lo que podría explicarse por el aumento de productividad propiciado por la incorporación de las nuevas tecnologías.

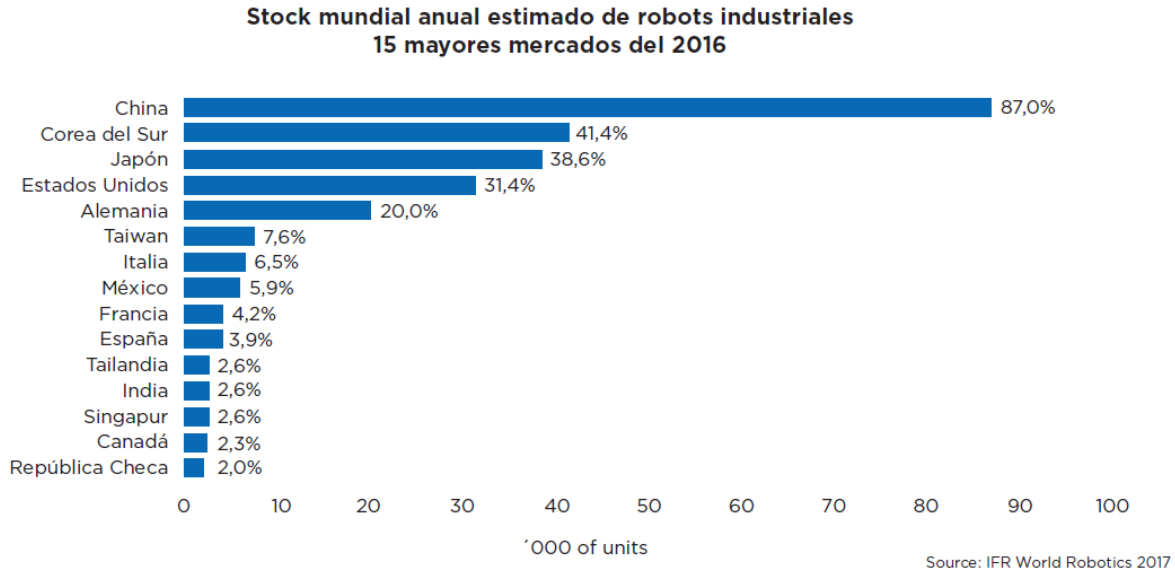


Figura 12: Stock mundial de robots industriales. Fuente: Paula Garnero (2018). Obtenido de: https://conexionintal.iadb.org/2018/09/03/industria-4-0-fabricando-el-futuro-3/

Las implicaciones y perspectivas futuras en el entorno de la industria 4.0, (Cortés, 2017) cita lo siguiente:

Hace una relación a 83 artículos revisados asociados con la industria 4.0 y sus tecnologías ver la figura 13, y halla que:

- El 42.5% de éstos buscan teorizar los conceptos asociados con la industria 4.0 y la manufactura inteligente.
- El 33.3% hace referencia a las tecnologías que sustentan las mismas.
- El 16.1% resaltan la importancia de los sistemas ciberfísicos, como los mecanismos o el medio tecnológico que permite la fusión del medio físico con el virtual.
- Y, el 8.1% está vinculado con las tecnologías que facilitan la integración de la cadena de suministro y valor.

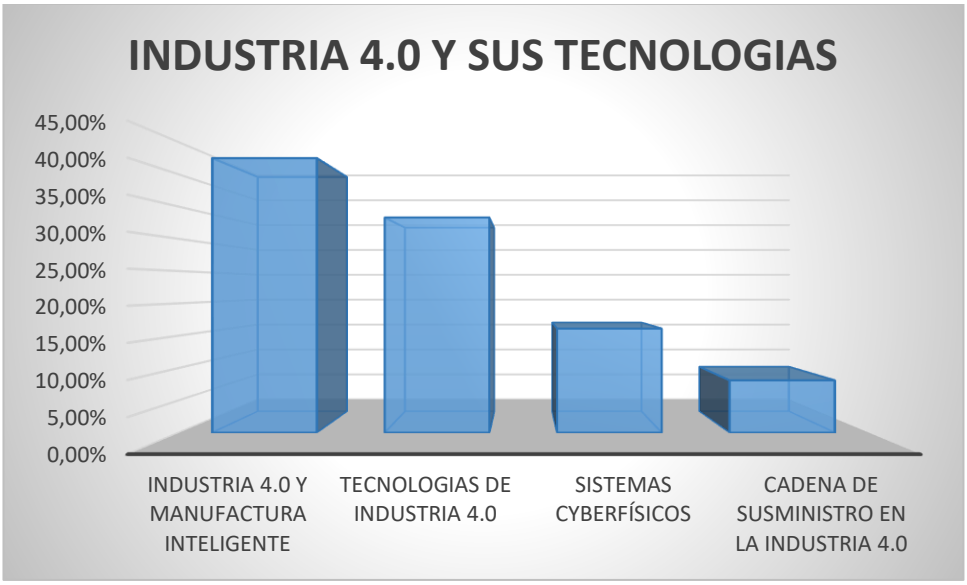


Figura 13: Industria 4.0 y sus tecnologías. Fuente: Ynzunza (2017). El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras. Obtenido de: https://www.redalyc.org/jatsRepo/944/94454631006/html/index.html#redalyc_94454631006_ref53

Se puede concluir que los sistemas ciberfísicos tienen un bajo porcentaje de desarrollo con relación a la industria 4.0 y manufactura inteligente, debido que resulta muy complejo integrar este tipo de sistemas con herramientas tradicionales de la anterior revolución industrial, por tal motivo en esta investigación se busca minimizar los procesos de integración haciendo uso de nuevas tecnologías que son parte de la industria 4.0.

Por otro lado, (Alsalemi & Alhomsi, 2017) menciona en su trabajo titulado “Real-Time Communication Network Using Firebase Cloud IoT Platform for ECMO Simulation” en el cual propone:

> …. implementar un sistema de simulación para médicos en tiempo real, el procedimiento requirió robustez y coordinación en la arquitectura de red por el cual se hizo uso de Internet de las cosas (IoT), el servicio Firebase como método de comunicación y se discutió las características como baja latencia, compatibilidad de hardware y gestión de datos. Firebase demostró ser una adecuada solución de comunicación que satisfizo la naturaleza urgente de entrenamiento médico porque cada unidad puede acceder en tiempo real a la base de datos y almacenar / leer

> cualquier parámetro una vez que sea necesario y podrá verificar cambios que ocurrieron, tal técnica pudo aumentar la eficiencia del sistema de comunicación Se establece una topología en estrella entre el dispositivo móvil (unidad) y Firebase (donde Firebase es el Hub) permitirá que el sistema de comunicación se mantenga en operación incluso si alguna unidad en la red no funciona o tenga una mala conexión y además puedan ser fácilmente integrado con el sistema actual y poder establecer condiciones de la vida real.

En tal virtud, esta investigación apoya al presente proyecto ya que permite considerar las características que brinda el servicio Firebase en la teleoperación y la estrecha relación que se tiene con la tecnología IoT ya que tiene una interacción entre dispositivos móviles y otros dispositivos conectados a la red de Internet cumpliendo con los siguientes parámetros: baja latencia, robustez, control remoto y compatibilidad de hardware.

En las telecomunicaciones (Llamas, 2019) habla sobre los protocolos de comunicación IoT y añade lo siguiente:

> Dentro del mundo del Internet de las cosas (IoT) se analiza los protocolos de comunicación disponibles para aplicaciones de IoT. Como sabemos el IoT es un campo en auge que, en forma muy resumida, consiste en que una gran cantidad de objetos cotidianos están o van a estar conectados entre sí. Un protocolo de comunicación es una serie de normas que se definen para que dos o más dispositivos puedan comunicarse y entenderse.
>
> Dentro de los protocolos de IoT seguramente el más popular es MQTT. Sin embargo, no sólo existe este protocolo en el campo del IoT. Tenemos muchas formas de comunicar realizar la comunicación M2M (machine-to-machine). Actualmente, con el desarrollo que han tenido las telecomunicaciones y el impulso que ha supuesto Internet, esto no resulta ningún problema.
>
> Condicionantes del M2M en IoT:
>
> - Gran cantidad de dispositivos.
> - Que sea escalable.

- La dependencia entre los dispositivos sea la menor posible, y deseablemente nula.
- Tenga interoperabilidad: es decir, que la solución funcione la mayor variedad de dispositivos, sistemas operativos, y lenguajes de programación.
- Gran número de comunicaciones simultáneas.
- Tenga seguridad de la información
- Acceder a los dispositivos fácilmente, por lo que se tendrá que lidiar con direcciones dinámicas y DHCP, posibles conexiones con mala latencia o ancho de banda, dependencia con la infraestructura de la red, firewalls, etc.

Las soluciones de comunicación en el IoT, que está siendo ampliamente empleada, es externalizar la comunicación un servicio de notificaciones centralizado. De hecho, es una infraestructura cada vez más habitual en informática, tanto en aplicaciones de IoT como para este caso de aplicación que es la teleoperación en tiempo real.

En definitiva, la solución consiste en disponer un servidor central que se encarga de recibir los mensajes de todos los dispositivos emisores, y distribuirlos a los receptores. De forma genérica se llamará a este servidor 'Router' o 'Broker'.

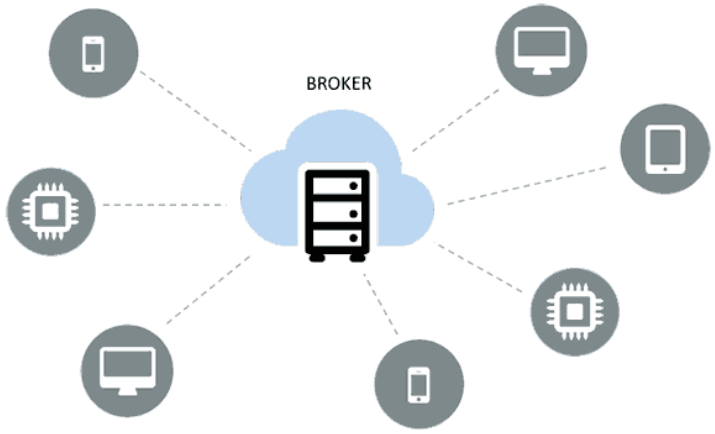


Figura 14: Servidor central denominado Broker o Router. Fuente: Luis llamas (2019). Obtenido de: https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/

En cuanto a metodologías que se puede encontrar en IoT (Chang, 2019) plantea lo siguiente:

Publish / Susbcribe (Pub/Sub)

La metodología Pub/Sub es un patrón de mensajería donde un agente, el 'Subscriber', informa al Router que quiere recibir un tipo de mensajes. Otro agente, el 'Publisher' puede publicar mensajes. El Router distribuye los mensajes a los Subscribers.

Router Remoder Procedure Calls (RRPC)

El RRPC es un patrón de ejecución remota de procedimientos donde un agente, llamado 'Callee', comunica al Router que proporciona un cierto procedimiento. Otro agente, llamado 'Caller', puede llamar a este procedimiento. El Router invoca el procedimiento en el Callee, recoge el resultado del proceso, y lo comunica al Caller que lo ha invocado.

Para los protocolos de IoT (Llamas, 2019), describe los siguientes:

Los protocolos destinados a aplicaciones de IoT, se describe a continuación algunos de los muchos protocolos M2M disponibles:

- MQTT (MQ Telemetry Transport) es un protocolo PubSub de Message Service que actúa sobre TCP. Destaca por ser ligero, sencillo de implementar. Resulta apropiado para dispositivos de baja potencia como los que frecuentemente tenemos en IoT. Está optimizado para el routing activo de un gran número de clientes conectados de forma simultánea.
- AMQP (Advanced Message Queuing Protocol) es un protocolo PubSub de Message Queue. AMQP está diseñado para asegurar la confiabilidad e interoperabilidad. Está pensado para aplicaciones corporativas, con mayor rendimiento y redes de baja latencia. No resulta tan adecuado para aplicaciones de IoT con dispositivos de bajos recursos.

- WAMP (Web Application Messaging Protocol) es un protocolo abierto que se ejecuta sobre WebSockets, y provee tanto aplicaciones de PubSub como rRPC.
- CoAP (Constrained Application Protocol) es un protocolo pensado para emplearse en dispositivos de IoT de baja capacidad. Emplea el modelo REST de HTTP con cabeceras reducidas, añadiendo soporte UDP, multicast, y mecanismos de seguridad adicionales.
- STOMP (Streaming Text Oriented Messaging Protocol, es un protocolo sencillo que emplea HTTP y mensajes de texto para buscar el máximo de interoperabilidad.
- XMPP (Extensible Messaging and Presence Protocol) es un protocolo abierto basado en XML diseñado para aplicaciones de mensajería instantánea.
- WMQ (WebSphere MQ) es un protocolo de Message Queue desarrolado por IMB.

Se añade a la situación actual, una lista de las aplicaciones y empresas que usan los servicios de Firebase:

- Square
- Twitch
- React
- Youtube
- Gmail
- Atlassian
- Alibaba
- The New York Times
- Shazam
- Trivago
- Duolingo
- Citrix

2.4. Metodología seleccionada

Las actividades planificadas para el desarrollo de un sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles está basada en la metodología planteada por (Drake, 2008), ya que tiene concordancia con los objetivos que se tiene en esta investigación. Por lo tanto, esta metodología está compuesta por varias fases que se describen a continuación:

En la primera fase se definirán las especificaciones de los módulos del sistema robótico teleoperado en tiempo real; en la segunda fase constará el esquema general del hardware y software a utilizarse en el presente proyecto de investigación; en la tercera, se conformará el organigrama general de la teleoperación; en la cuarta, se realizará la adaptación entre el hardware y el software; en la quinta, se constituirán los ordinogramas modulares y se realizará la selección de plataformas en el cual se desarrollará la codificación del sistema teleoperado; en la sexta fase, se ejecutará el desarrollo del software; en la séptima fase se llevará a cabo la implementación del hardware con el software, en la octava fase se detallarán las hojas de especificaciones de los componentes y protocolos de comunicación. Y en la novena fase se implementará el sistema integrado en tiempo real para la teleoperación y se realizarán las pruebas finales.

2.4.1. Fase I. definición de las especificaciones

En esta primera fase se define con la mayor precisión el funcionamiento del sistema a desarrollar, también se deben establecer los estímulos de entrada y salida sin explicar las razones.

2.4.2. Fase II. esquema general del hardware

En esta fase, se desarrolla en forma general los bloques funcionales que componen y las interconexiones entre sí de manera lógica. Se asume como bloques funcionales, cada una de las partes elementales del sistema encargada de hacer un único trabajo en particular.

2.4.3. Fase III. Ordinograma general

Siguiendo la secuencia, se establece un diagrama de flujo que se estima indicará el funcionamiento en forma general del sistema teleoperado, ya que sirve como base para el

desarrollo del software, el cual está sujeto a modificaciones. Se lo presenta en forma general de tal manera que muestre el funcionamiento del sistema integrado para la teleoperación en tiempo real.

2.4.4. Fase IV. Adaptación entre hardware y software

Luego de formar el organigrama general se plantea la arquitectura de red para la comunicación entre hardware y software, garantizado que la información entra y sale de forma correcta.

2.4.5. Fase V. Ordinogramas modulares y codificación del sistema

En esta fase cada uno de los siguientes bloques del diagrama de flujo se codifican individualmente, asegurándose que cada parte realice el trabajo en forma eficiente y segura, esto se logra codificando en forma independiente con el software seleccionado para obtener un óptimo resultado.

2.4.6. Fase VI. Especificaciones técnica y Protocolos de comunicación del hardware

En esta etapa se toma en cuenta las características de todos los dispositivos a utilizar, así como también las hojas de especificaciones técnicas y protocolos de comunicación de cada dispositivo que trabaja en red para la teleoperación.

2.4.7. Fase VII. Desarrollo del software

En esta fase se desarrolla, prueba y se depura tanto el programa de la aplicación móvil que transmitirá los mandos de teleoperación y el programa del controlador del robot que receptará los mandos de teleoperación hasta que el funcionamiento sea adecuado.

2.4.8. Fase VIII. Integración del hardware con el software

Para continuar se prueban la comunicación entre el dispositivo móvil y la plataforma Firebase, por otro lado, también se realiza la comunicación entre el controlador del robot y la plataforma Firebase de tal forma que se pueda confirmar que exista una correcta correspondencia entre ambos y que el sistema interactúe de manera eficiente.

2.4.9. Fase IX. Implementación del sistema integrado en tiempo real para la teleoperación y resultados finales

En esta última fase, se integran todos los dispositivos con sus respectivos protocolos de comunicación y los programas que se compilan, para llevar a cabo el funcionamiento general del sistema, y realizar las pruebas de rendimiento, latencia e integridad de la información para obtener los resultados finales.

2.5. Cuadro de fases metodológicas

Una vez planteada la metodología utilizada, se muestra en la tabla 3 con la planificación de las actividades realizadas para cumplir con los objetivos específicos presentados en la introducción del presente trabajo de investigación. Mostrando la concordancia entre éstos últimos con las fases y así como los recursos a empleados para lograrlas. Además, se presenta un cronograma con las actividades a ejecutar durante la elaboración del proyecto.

Tabla 3: Actividades y recursos para cumplir el objetivo general

Objetivos Específicos	Fases Metodológicas	Actividades	Recursos
Objetivo 1. Establecer la arquitectura de red de la teleoperación usando el Cloud Computing para la optimización de recursos.	Fase I. Definición de las especificaciones del sistema teleoperado en tiempo real.	Definición de los módulos que componen el funcionamiento del sistema.	Datos obtenidos en el capítulo I. Tabla de definición módulos elaborado en Word.
	Fase II. Esquema general del hardware del sistema teleoperado en tiempo real.	Desarrollo general de los bloques funcionales e interconexiones que componen el sistema.	Diagrama con los componentes principales del sistema elaborado en Word.
	Fase III. Ordinograma general del sistema teleoperado en tiempo real.	Establecimiento de un diagrama de flujo para indicar el funcionamiento general del sistema.	Diagramas de flujo del sistema elaborado en Word.

Objetivos Específicos	Fases Metodológicas	Actividades	Recursos
Objetivo 1. Establecer la arquitectura de red de la teleoperación usando el Cloud Computing para la optimización de recursos.	Fase IV. Adaptación entre hardware y software que forman parte del sistema de teleoperación.	Plantear la arquitectura de red para la comunicación entre hardware y software, garantizado que la información entra y sale de forma correcta.	Datos obtenidos en el capítulo I. Definición de arquitectura de red elaborado en Word
Objetivo 2 Generar una aplicación móvil en Android Studio para el control del brazo robótico teleoperado.	Fase V. Ordinogramas modulares y codificación del sistema teleoperado.	Codificación individual de los bloques del diagrama de flujo.	Datos obtenidos en la fase anterior. Lenguaje de Programación.
	Fase VI. Especificaciones técnica y Protocolos de comunicación del hardware del sistema teleoperado en tiempo real.	Se detallan las características técnicas de todos los dispositivos y los protocolos de comunicación	Hojas de especificaciones técnicas de los dispositivos
	Fase VII. Desarrollo del Software del sistema teleoperado en tiempo real.	Se desarrolla, prueba y se depura tanto el programa de la aplicación móvil y el programa del controlador del robot.	Programador de aplicaciones móviles. Programador de microcontroladores.

Objetivos Específicos	Fases Metodológicas	Actividades	Recursos
Objetivo 3 Gestionar la teleoperación en la plataforma Firebase para la minimización de procesos de integración.	Fase VIII. Integración del hardware con el Software del sistema teleoperado en tiempo real.	Integración entre el dispositivo móvil y la plataforma Firebase y la comunicación entre el controlador del robot y la plataforma Firebase	Plataforma Firebase
Objetivo 4. Valorar el funcionamiento en función al cumplimiento de los estándares de calidad y servicio para la medición del rendimiento del sistema teleoperado.	Fase IX. Implementación del sistema integrado en tiempo real para la teleoperación y pruebas finales	Se integran todos los dispositivos con sus respectivos protocolos de comunicación y los programas que se compilan, para llevar a cabo el funcionamiento general del sistema.	Programador de microcontroladores. Hojas de especificaciones técnicas de los dispositivos. Programador de aplicaciones móviles. Plataforma Firebase.

Nota. Tabla adaptada por el autor

2.6. Cronograma de actividades y recursos

El cronograma de actividades se lo presenta en el Anexo A.

CAPITULO III

3. PROPUESTA

Quedando constituidas según se expone a continuación:

3.1. Definición de las especificaciones del sistema robótico teleoperado

Se empieza definiendo los módulos que componen el funcionamiento del sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles, el brazo robótico en el cual se implementará la teleoperación tiene 5 grados libertad, y podrá realizar trabajos de manipulación *pick and place* por medio de un gripper, el robot tendrá como controlador una tarjeta de Arduino Uno y módulo Wifi ESP8266 que permitirá al robot conectarse a una red Wifi y al servicio Firebase el cual gestionará la información del estado del robot en tiempo real, también se desarrollará una aplicación móvil que será compatible con el sistema operativo Android en la cual se podrá teleoperar el robot y tendrá información sobre el estado del robot en tiempo real.

Tabla 4: Definición de módulos

Módulos	Definición	Selección
Módulo 1	Hardware	- Brazo robótico de morfología antropomórfica. - Tarjeta Wifi ESP8266. - Cámara Wifi. - Smartphone con el sistema operativo Android.
Módulo 2	Software de desarrollo	- Android Studio (Java) - IDE Arduino (Processing) - Emulador Nox 6 (Android)

Módulo 3	Plataforma de control de la teleoperación	- Google Cloud Platform
		- Firebase
		  - Realtime Database
		  - Authentication
		  - Analytics

Nota. Tabla adaptada por el autor

3.2. Esquema general del hardware del sistema robótico teleoperado

En esta fase, se desarrolla el esquema general de las interconexiones entre sí de manera lógica, cada una de las partes elementales del sistema encargada de hacer un único trabajo en particular.

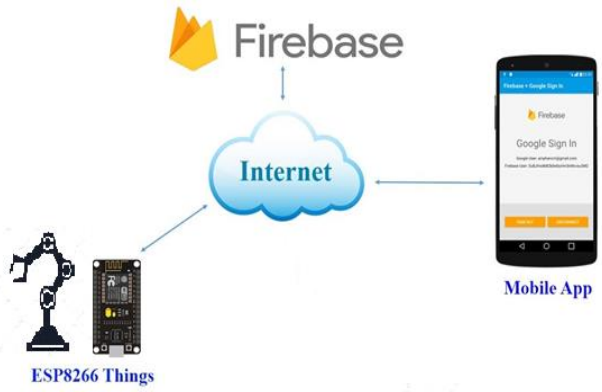


Figura 15: Esquema general del hardware. Fuente: Del autor

3.3. Ordinograma general del sistema robótico teleoperado

Siguiendo la secuencia, se indicará el funcionamiento en forma general del sistema teleoperado, ya que sirve de base para el desarrollo del proyecto. De tal manera que muestre el funcionamiento del sistema integrado para la teleoperación en tiempo real.

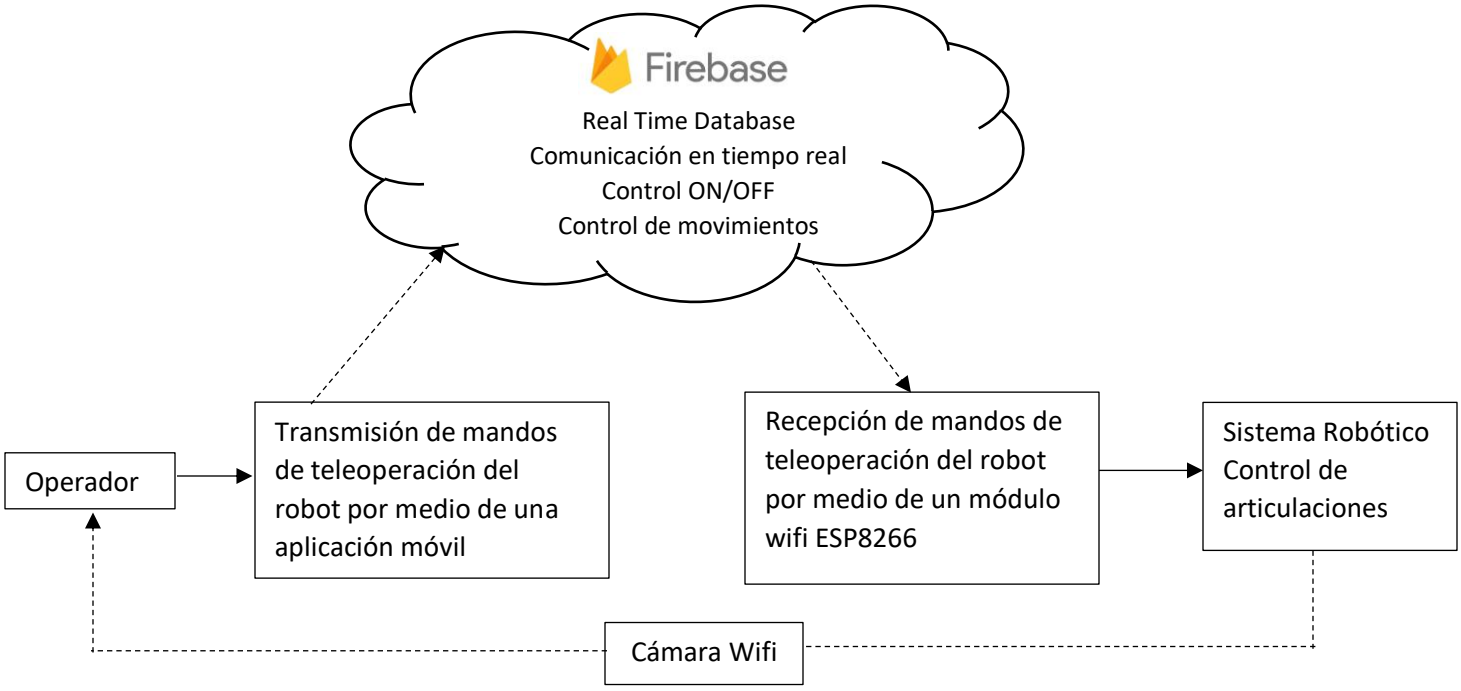


Figura 16: Ordinograma general del sistema teleoperado. Fuente: Del autor

3.4. Adaptación entre hardware y software del sistema robótico teleoperado

Luego de formar el organigrama general se plantea la arquitectura de red para la comunicación entre hardware y software, garantizado que la información entra y sale de forma correcta.

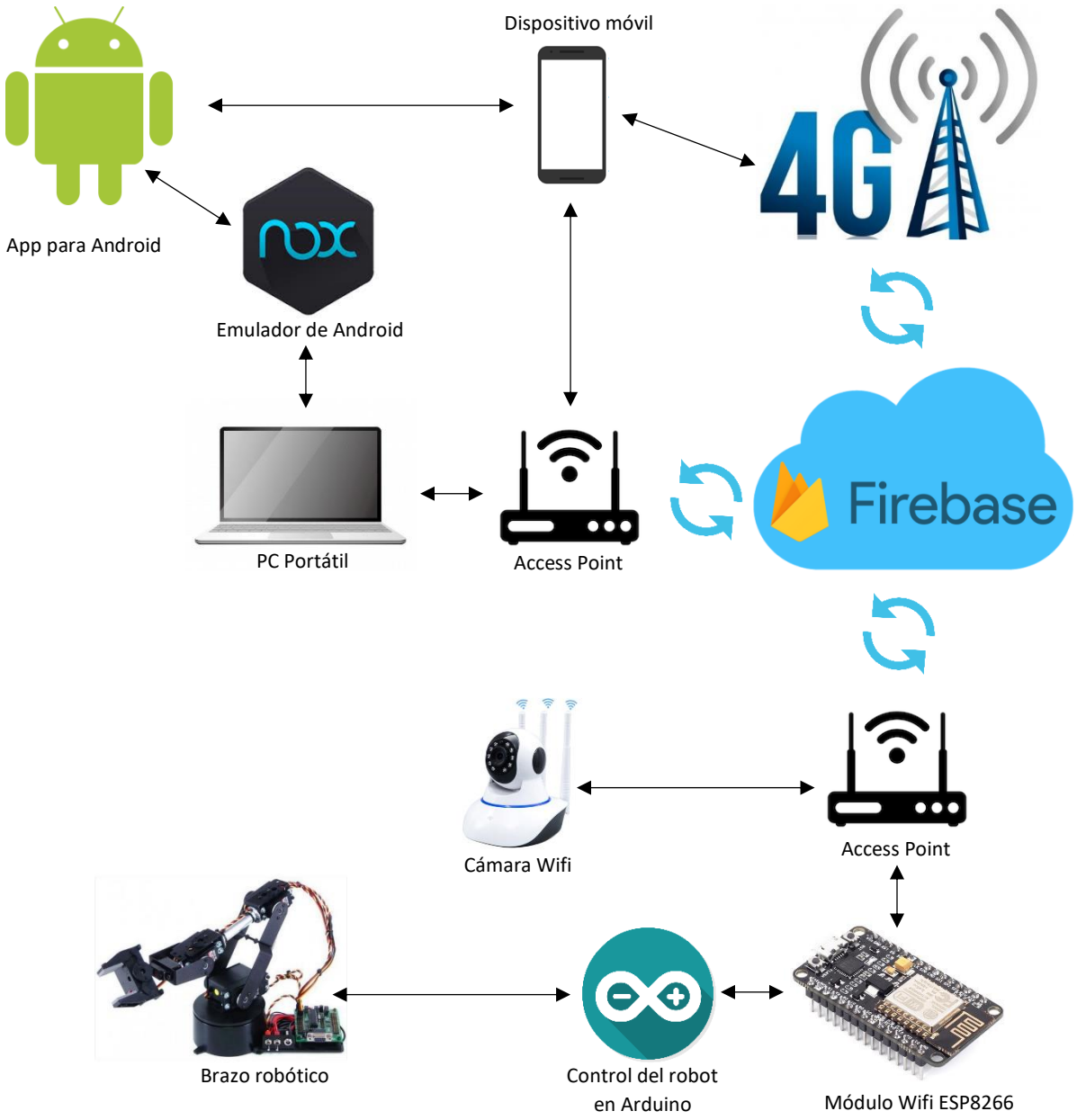


Figura 17: Arquitectura de comunicación entre hardware y software. Fuente: Del autor

3.5. Ordinogramas modulares y codificación del sistema robótico teleoperado

En esta fase cada uno de los siguientes bloques del diagrama de flujo se codifican individualmente, asegurándose que cada parte realice el trabajo en forma eficiente y segura, esto se logra codificando cada módulo en forma independiente con el software seleccionado para obtener un óptimo resultado.

Se agrega la lógica de Backend a los datos en tiempo real con Firebase y Google App Engine, ya que Firebase es una plataforma para crear aplicaciones móviles basadas en la web y Android, que ofrece almacenamiento de datos en tiempo real y sincronización automática entre herramientas de interfaz de usuario en los dispositivos, que son los Frontend ver la figura 18.

Frontend y Backend, (Chapaval, 2018), menciona qué:

> Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web y móvil que corren en el navegador o interfaz y que se encargan de la interactividad con los usuarios.
>
> Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas.

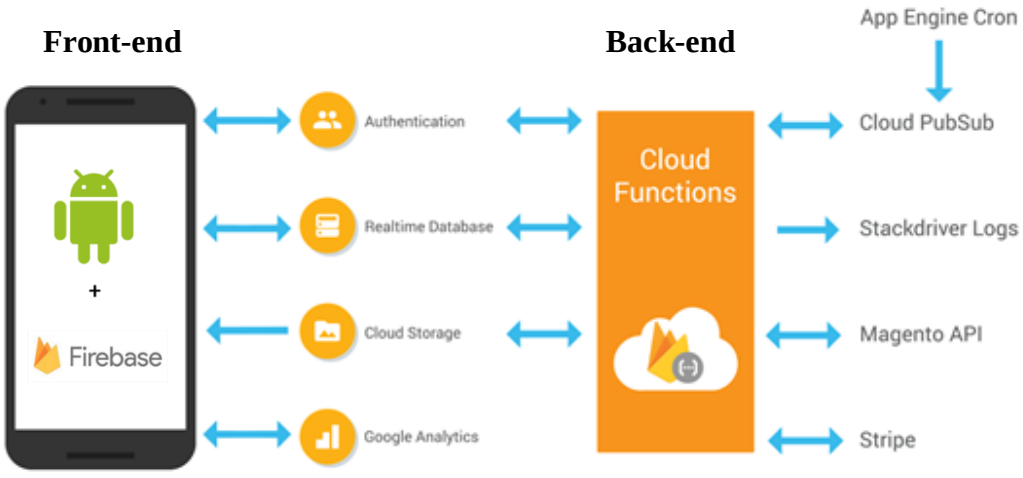


Figura 18: Lógica de comunicación Firebase Android. Fuente: David DeRemer (2019). Obtenido de: https://www.googblogs.com/hamilton-app-takes-the-stage/

Las herramientas que se usa para realizar aplicaciones de Cloud Computing, (Google Cloud, 2020), y dice que:

> El entorno flexible de App Engine es una plataforma de aplicaciones que supervisa, actualiza y escala el entorno de hosting; todo lo que necesitas hacer es escribir el código de tu servicio de backend móvil.
>
> Este entorno ejecuta el servicio de backend dentro de contenedores de Docker que puedes configurar. Esto significa que puedes llamar a binarios nativos, escribir en el sistema de archivos y hacer otras llamadas al sistema.
>
> Si tu aplicación necesita procesar datos del usuario o necesita organizar eventos, ampliar Firebase con el entorno flexible de App Engine te da el beneficio de poder sincronizar de forma automática datos en tiempo real, sin la necesidad de ejecutar tu código dentro de la zona de pruebas de App Engine como se ve en la figura 19.

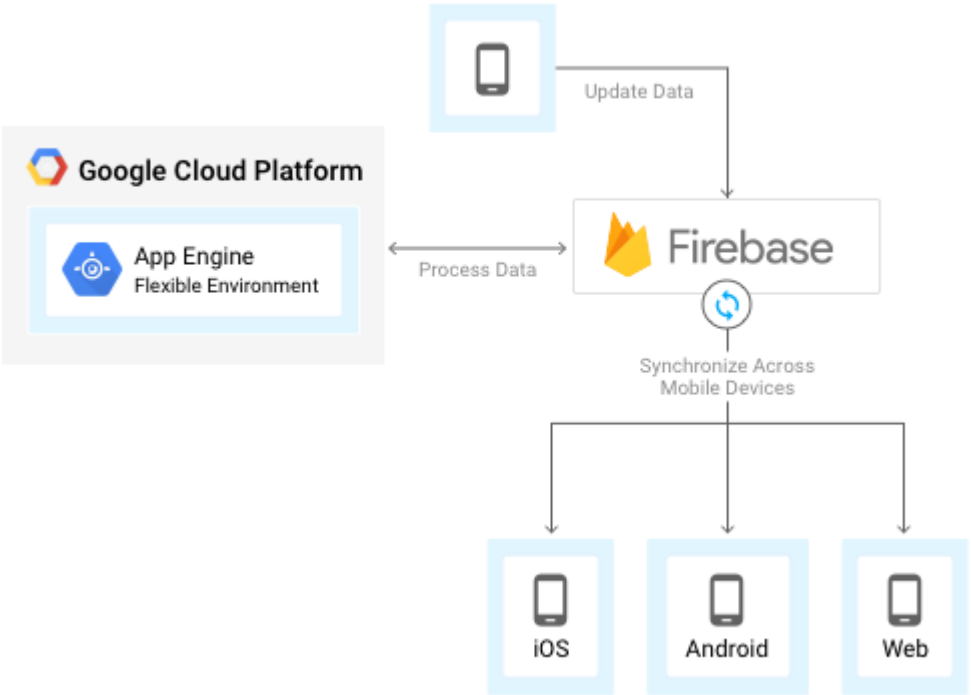


Figura 19: Entorno flexible App Engine. Fuente: Google Cloud Platform (2019). Obtenido de: https://cloud.google.com/solutions/mobile/mobile-app-backend-services?hl=es-419#firebase-appengine-standard

Recomendado para estas situaciones:

- Apps de Firebase que necesitan un servicio de backend para modificar los datos sincronizados y, a su vez, ese servicio necesita una configuración de servidor personalizada o bibliotecas de terceros no compatibles con el entorno estándar de App Engine.
- Servicios de backend que necesitan una conexión persistente a Firebase para recibir notificaciones de cambios en los datos. El entorno flexible puede mantener una conexión abierta durante 24 horas.

Los dispositivos móviles que se usarán son los Smartphone con sistema operativo Android y contendrán la aplicación móvil que realizará el control de mandos teleoperados en tiempo real y se desarrollará en la plataforma de Android Studio ya que contiene a Firebase como herramienta lo cual facilitará su implementación, para el presente proyecto de investigación se usará el Realtime Database de Firebase porque la base de datos es alojada en la nube y en la documentación de (Firebase, 2019), menciona las siguientes ventajas:

- No se tiene que preocupar de buscar un servidor e instalar software, Google se encarga de todo.
- Los datos se almacenan en formato JSON, se trata de una base de datos NoSQL.
- Sincronización en tiempo real.
- Cualquier cambio en los datos hace que los clientes reciban actualizaciones en cuestión de milisegundos.
- Permite apps multiplataforma iOS, Android y Web.
- Todos los clientes comparten la misma base de datos y reciben actualizaciones de forma automática con los datos más nuevos independientemente de su plataforma.
- Se dispone de un API REST para acceder a la base de datos desde otras plataformas.
- Si el dispositivo pierde la conexión a Internet la aplicación podrá seguir funcionando, mientras tanto se trabajará con una caché y cuando se recupere la conexión, la información será actualizada.

Figura 20: Realtime Database de Firebase. Fuente: The Engineer's Café (2019). Obtenido de: https://theengineerscafe.com/save-and-retrieve-data-firebase-android/firebase3/

Otra herramienta que se usará de Firebase es la Authentication porque permite añadir seguridad sobre la información en la aplicación, conocer la identidad de un usuario permite que una app guarde sus datos en la nube de forma segura, ver la figura 21.

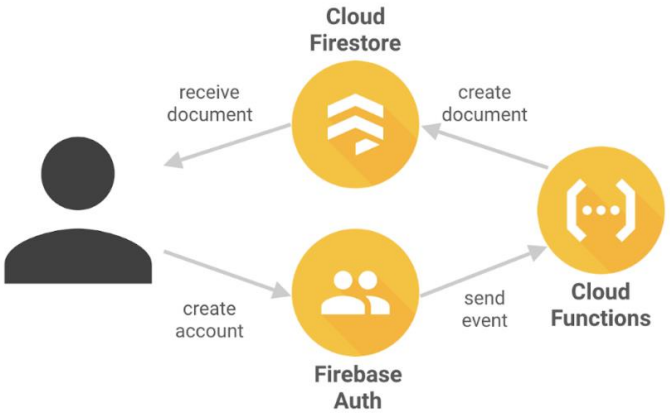


Figura 21: Seguridad con Firebase. Fuente: Doug Stevenson (2019). Obtenido de: https://medium.com/firebase-developers/patterns-for-security-with-firebase-offload-client-work-to-cloud-functions-7c420710f07

Firebase es un Backend como servicio. Y tiene un módulo llamado Firebase Authentication que apunta a operaciones relacionadas con el usuario. Mantiene internamente un registro de usuario donde puede realizar operaciones definidas como:

- Inicio de sesión (vía, dirección de correo electrónico, número de teléfono móvil)
- Registro
- Usuario registrado
- Cerrar sesión
- Contraseña olvidada
- Enviar correo de verificación a la dirección de correo electrónico asociada
- Actualizar contraseña / correo electrónico
- Actualizar información de perfil
- Borrar Usuario
- OpenID Connect (como Google SignIn, Facebook Sign In, Twitter SignIn y Github Sign In)

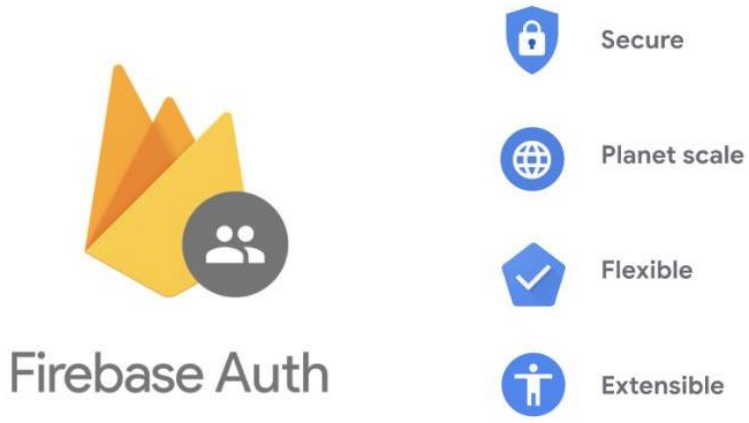


Figura 22: Authentication de Firebase. Fuente: Firebase (2019). Obtenido de: https://twitter.com/firebase/status/1126597715159392278

Para generar reportes sobre el uso, carga y descarga de información de la aplicación móvil para el sistema robótico teleoperado, también se debe realizar un análisis haciendo uso de Firebase Analytics que se detalla a continuación:

Google Analytics es una solución de análisis ilimitada y gratuita que ocupa un lugar central en Firebase. Analytics se integra a distintas funciones de Firebase y proporciona una capacidad ilimitada de generar informes sobre un total de hasta 500 eventos distintos que puedes definir con el SDK de Firebase. Los informes de Analytics te permiten entender claramente cómo se comportan tus usuarios para que puedas tomar decisiones fundamentadas en relación con el marketing de las apps y las optimizaciones del rendimiento.

Figura 23: Firebase Analytics. Fuente: Firebase documents (2019). Obtenido de: https://firebase.google.com/docs/analytics?utm_campaign=Firebase_announcement_education_general_en_05-18-16_&utm_source=Firebase&utm_medium=yt-desc

Para esta investigación se usará como dispositivo remoto para el control del brazo robótico el NodeMCU V3 es un poderoso chip que se puede programar de la misma forma que un Arduino®, sin embargo, no se ha usado para lo que fue diseñado principalmente; basado en el chip Wifi™ ESP8266, el NodeMCU V3 se puede usar para aplicaciones IoT, ya que tiene acceso a Internet de manera muy fácil, solo basta con una red Wifi™ y unas dependencias para conseguirlo.

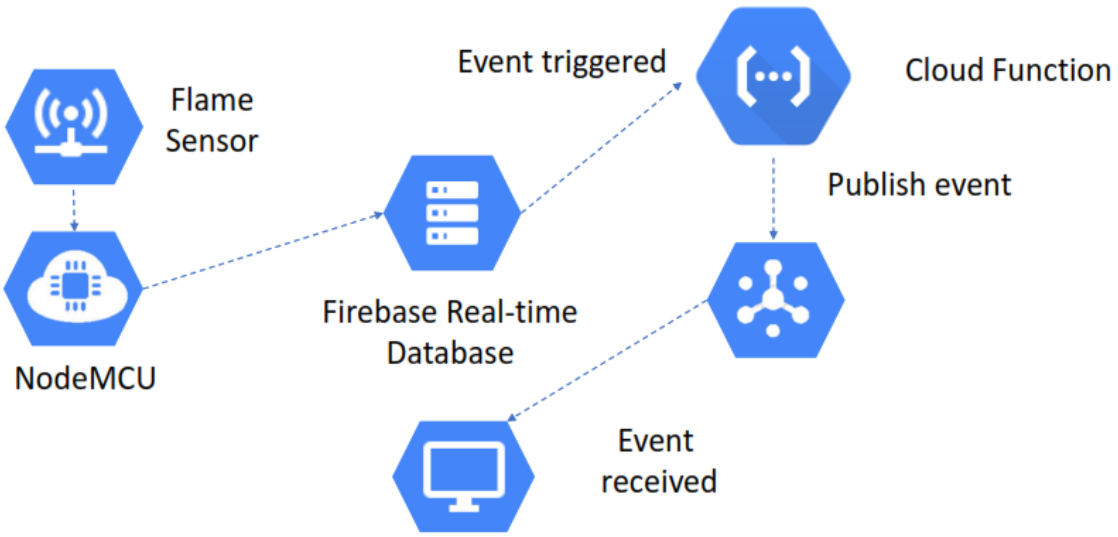


Figura 24: NodeMCU y Firebase. Fuente: Shang Yi Lim (2017). Obtenido por: https://www.slideshare.net/LimYi1/iot-google-cloud-functions-with-firebase/27

Para conectar correctamente la tarjeta Wifi™ ESP8266 mediante Arduino® y Firebase se requiere lo siguiente:

- Descargar la librería Firebase Arduino extended.
- Descargar la librería Arduino JSon versión 5.13.1.
- Tener cuenta de Google creada para acceder a Firebase, de lo contrario, no se podrá usar esta poderosa herramienta.
- Tener instalado el entorno de desarrollo de Arduino®, disponible en la página oficial.
- Tener instalada y configurada la librería ESP8266.

3.6. Especificaciones técnicas del hardware del sistema robótico teleoperado

En esta etapa se toma en cuenta las características de todos los dispositivos a utilizar, así como también las hojas de especificaciones técnicas y protocolos de comunicación de cada dispositivo que trabaja en red para la teleoperación.

3.6.1. Brazo Robótico con Cuatro Grados de Libertad AL5B Lynxmotion

La línea de brazos robóticos Brazo Robótico con Cuatro Grados de Libertad AL5B Lynxmotion, brinda un movimiento rápido y preciso, su estructura está compuesta por hombro plano, codo, movimiento de muñeca y una pinza de agarre funcional. Lynxmotion diseñó un

sistema asequible basado en un diseño robusto y de eficacia probada que durará y durará. El brazo robótico de aluminio está fabricado con los componentes del conjunto de servoejector de Lynxmotion para una flexibilidad máxima y capacidad de expansión, consta de soportes de aluminio anodizado negro, tubos y cubos de aluminio, componentes moldeados por inyección personalizados y componentes de precisión cortados con láser.

Figura 25: Brazo Robótico con Cuatro Grados de Libertad AL5B Lynxmotion. Fuente: RobotShop (2019). Obtenido de: https://www.robotshop.com/us/es/brazo-robotico-con-cuatro-grados-libertad-al5b-lynxmotion-solo-hardware.html

Tabla 5: Especificaciones físicas del brazo robótico

Especificaciones del hardware	**Dimensiones**
• Base (articulación 1) • Hombro (articulación 2) • Codo (articulación 3) • Muñeca pitch (articulación 4) • Muñeca yaw (articulación 5) • Gripper (efector final) • Alcance mediano de 7,5 " • Servomotores (Rango de movimiento por eje = 180 grados)	• Distancia (eje de base a codo) = 3,75 " • Distancia (eje codo-a-muñeca) = 5,00 " • Altura (brazo estacionado) = 6,375 " • Altura (alcanzando arriba) = 15,75 " • Media de alcance hacia adelante = 7,50 " • Apertura de la pinza = 1,25 "

Nota. Tabla adaptada por el autor

3.6.2. NodeMCU V3 - ESP8266

NodeMCU es una pequeña placa Wifi denominada ESP8266 que es compatible con el IDE de Arduino, revisar su datasheet en el Anexo B. adicional a esto (Brico Geek, 2020) y describe las características como se muestra a continuación:

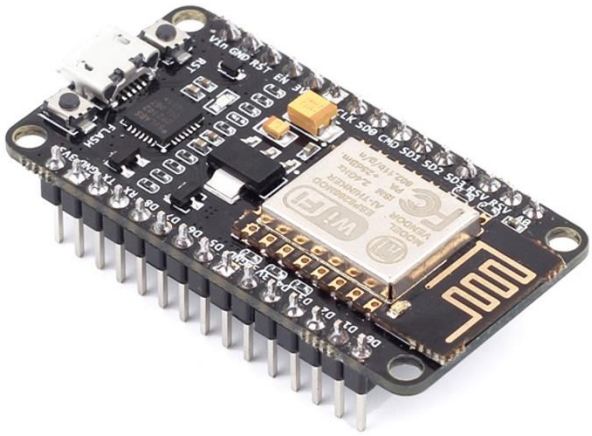


Figura 26: Tarjeta Wifi NodeMCU V3 - ESP8266. Fuente: Germán Martín (2018). Obtenido de: https://programarfacil.com/esp8266/como-programar-nodemcu-ide-arduino/

NodeMCU es una pequeña placa Wifi lista para usar en cualquier proyecto IoT. Está montada alrededor del ya conocido ESP8266 y expone todos sus pines en los laterales. Además, ofrece más ventajas como la incorporación de un regulador de tensión integrado, así como un puerto USB de programación. Se puede programar con LUA o mediante el IDE de Arduino.

Dispone de una extensa comunidad y documentación que te permitirán conectar el proyecto al mundo exterior mediante conexión Wifi. Debido a que utiliza un conversor USB CH340, normalmente el sistema operativo lo instala automáticamente, aunque dependiendo de los casos, puede que se necesite instalar el driver específico.

Características:

- Procesador: ESP8266 @ 80MHz (3.3V) (ESP-12E)
- 4MB de memoria FLASH (32 MBit)

- WiFi 802.11 b/g/n
- Regulador 3.3V integrado (500mA)
- Conversor USB-Serial CH340G / CH340G
- Función Auto-reset
- 9 pines GPIO con I2C y SPI
- 1 entrada analógica (1.0V max)
- 4 agujeros de montaje (3mm)
- Pulsador de RESET
- Entrada alimentación externa VIN (20V max)

Con respecto a esta tarjeta ESP8266 (Martín, 2018), resalta los siguiente:

NodeMCU fue una de las primeras placas de desarrollo con el microcontrolador ESP8266. Hasta entonces este chip solamente estaba disponible como placas ESP-xx como ESP01 o ESP12. Cuando fue presentada, no existía la integración de ESP8266 con el entorno de Arduino. No utilizaba un lenguaje compilado sino uno interpretado llamado LUA. Con la aparición de la integración con el IDE de Arduino, este firmware ha caído poco a poco en desuso. En cambio, la facilidad para conectarlo al ordenador para programarlo y la posibilidad de hacerlo desde el IDE de Arduino han hecho que el hardware siga disponible en su forma original y en forma de otras muchas placas derivadas.

A continuación, puedes ver el pineado de la NodeMCU original:

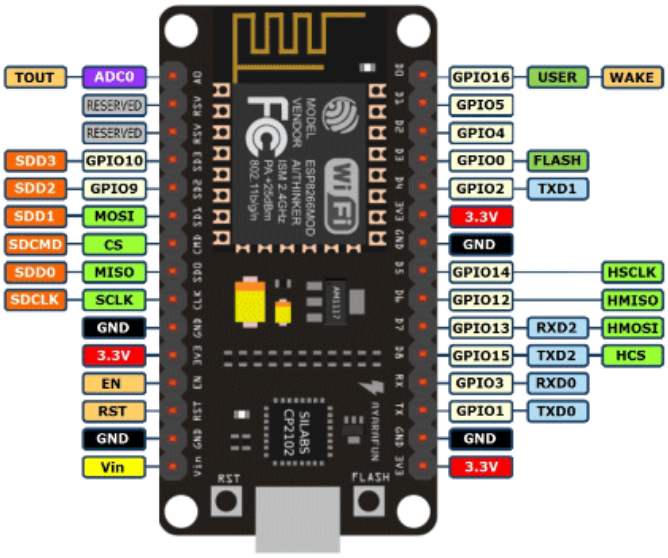


Figura 27: NodeMCU pineado. Fuente: Germán Martín (2018). Obtenido de: https://programarfacil.com/esp8266/como-programar-nodemcu-ide-arduino/

3.6.3. Dispositivos móviles con S.O. Android

Figura 28: Smartphone con S.O. Android. Fuente: Actualidad RT (2019). Obtenido de: https://actualidad.rt.com/actualidad/298453-reporte-seguridad-nokia-seguridad-android

SmartPhone es un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono móvil común. La característica más importante de todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar sus posibilidades,

como el procesamiento de datos y la conectividad o internet. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero.

Las características de un SmartPhone (Área tecnología, 2020), menciona las siguientes:

- Por su puesto, permite realizar llamadas telefónicas.
- Soporta correo electrónico y posibilidad de conexión a redes sociales.
- Cuenta con GPS.
- Permiten la instalación de programas de terceros.
- Utiliza cualquier interfaz para el ingreso de datos, como por ejemplo teclado QWERTY, pantalla táctil.
- Te permiten ingresar a Internet con tecnología 4G.
- Conectividad inalámbrica Wi-Fi.
- Poseen agenda digital, administración de contactos.
- Permitan leer documentos en distintos formatos, entre ellos los PDFs y archivos de Microsoft Office.
- Debe contar con algún sistema operativo móvil, tales como: Android, iOS, HarmonyOS, entre otros.
- Poseer memorias externas como microSD.
- Cámara trasera y delantera con muchos megapíxeles.
- Sincronización inalámbrica con otros dispositivos, como ordenadores portátiles o de sobremesa.
- Con un teléfono inteligente puedes hacer de todo al mismo tiempo, o lo que es lo mismo son multitareas.

Estos son todos los móviles que tienen Android 9 Pie al acabar 2018, según (García, 2018) menciona que:

> Android 9 Pie no haya llegado a algunos terminales, la mayoría de ellos gama alta de este año y el anterior. En la tabla 6 que se tiene más abajo se podrá ver todos los móviles que han sido actualizados a Android Pie 9 o que fueron lanzados con dicha versión, es decir, todos los dispositivos que cierran 2018 con esta versión del sistema operativo de Google.

La lista ha sido desarrollada por AOSMark, fue actualizada por última vez el 29 de diciembre del 2018 y está ordenada alfabéticamente según el nombre del fabricante:

Tabla 6: Smartphones con Android Pie 9

Marca	Modelo
ASUS	ZenFone 5 | ZenFone 5 Lite ZenFone 5Z (en Taiwán) Zenfone Max Pro (en beta)
GOOGLE	Pixel | Pixel XL Pixel 2 | Pixel 2 XL Pixel 3 | Pixel 3 XL
HTC	HTC U11 life
HUAWEI Y HONOR	Honor Play Honor 10 Honor View 10 Honor View 20 Huawei Nova 3i (en beta) Mate 10 | Mate 10 Pro | Mate 10 Lite Huawei P20 | P20 Pro Mate 20 | Mate 20 Pro | Mate 20 X | Mate 20 Lite
LG	LG G7 One LG G7 ThinQ (en beta)
MOTOROLA	Moto X4 Motorola one | Motorola One Power

Marca	Modelo
NOKIA	Nokia 6 Nokia 6.1 | Nokia 6.1 Plus Nokia 7 Plus Nokia 7.1 Nokia 8 Nokia 8.1
SAMSUNG	Galaxy S9 | Galaxy S9+ Galaxy Note 9 (en beta)
SONY	Xperia XZ Premium Xperia XZ1 | Xperia XZ1 Compact Xperia XZ2 | Xperia XZ2 Compact | Xperia XZ2 Premium
XIAOMI	Mi A1 Mi A2 Mi A2 Lite Mi MIX 2S Mi MIX 3 POCOPHONE F1 Mi 8 | Mi 8 Explorer Edition Mi 8 Lite | Mi 8 Pro (en beta) Xiaomi Mi MAX 3

Nota. Tabla adaptada por (García, 2018)

3.6.4. Cámara Wifi

Uno de los dispositivos que forman parte del sistema teleoperado, es una cámara Wifi, la cual permitirá un *feedback* visual en la teleoperación, ya que permite observar en tiempo real al brazo robótico de forma remota a través de una aplicación móvil. La configuración revisar desde el Anexo C al Anexo F.

Se adquirió una cámara Wifi en (Made-in-China, 2020) y presenta las siguientes características:

Figura 29: Cámara IP inalámbrica. Fuente: (Made-in-China, 2020). Obtenido de: https://es.made-in-china.com/co_newerton/product_P2p-Three-Antenna-WiFi-Security-2-0MP-Surveillance-Camera-1080P-Wireless-IP-Camera_ririgugig.html

Especificaciones técnicas:

- Distancia de infrarrojos(m): 5~15m
- Acción de alarma: FTP Foto,alarma,Fotos de correo electrónico local.
- La tecnología de infrarrojos:
- Nombre de marca: Howell
- CMOS Sensor.
- Compatible sistemas operativos: Windows 10,Windows 7, Mac OS, Windows 8,Windows XP
- El soporte de pared lateral
- Ángulo de visualización (Grado): 360grados
- Dimensiones (L x An x D)(mm): 120*100*100mm.
- Los sistemas móviles compatibles: Android, iOS, Windows Mobile
- Formato de compresión de vídeo: H.264

- La alta definición: 1080p (Full-HD).
- La salida de audio RCA: 1CH
- Conectividad: Red inalámbrica IP/
- Color: blanco
- El consumo de energía(W): 3W
- Estilo: Mini Cámara
- Instalación: Embedded
- La interfaz de red Wi-Fi/802.11/b/g
- Tipo: Cámara IP
- Marca: Sensor SONY
- La lente (mm): 2,8 mm
- Fuente de alimentación(V): 5V
- Número de modelo: NT-SP-C3

3.7. Desarrollo del software de la aplicación móvil y controlador del brazo robótico

En esta fase se desarrolla, prueba y se depura tanto el programa de la aplicación móvil que transmitirá los mandos de teleoperación y el programa del controlador del robot que receptará los mandos de teleoperación hasta que el funcionamiento sea adecuado.

Firebase para Android

La base de datos en tiempo real de Firebase (Firebase Realtime Database) es sin duda uno de los servicios más populares de la plataforma. Contar con la capacidad de almacenar datos en la nube es uno de los requerimientos de los que pocas aplicaciones actuales pueden escapar, y poder hacerlo sin necesidad de preocuparnos por toda la infraestructura de servidor necesaria es toda una ventaja.

Antes de ponernos a trabajar con Firebase desde un proyecto de Android Studio tendremos dirigirnos a la consola de desarrolladores, en esta ocasión la Consola de Firebase, para crear y configurar el nuevo proyecto y asociar a él nuestra aplicación.

Si es la primera vez que accedemos a la consola de Firebase, aparecerá un mensaje de bienvenida que directamente, invita a empezar a crear un nuevo proyecto.

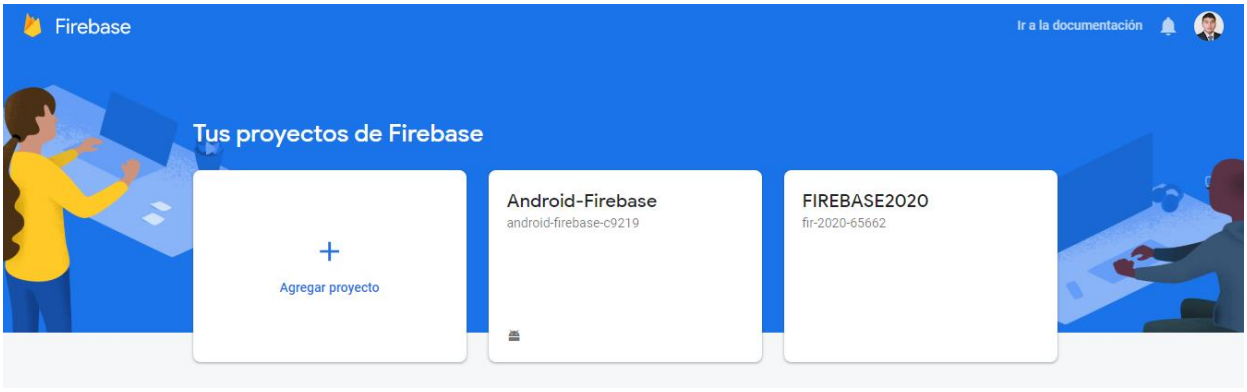


Figura 30: Consola de Firebase. Fuente: Del autor

Si es la primera vez que accedemos a la consola de Firebase, aparecerá un mensaje de bienvenida que directamente e invita a empezar a crear un nuevo proyecto.

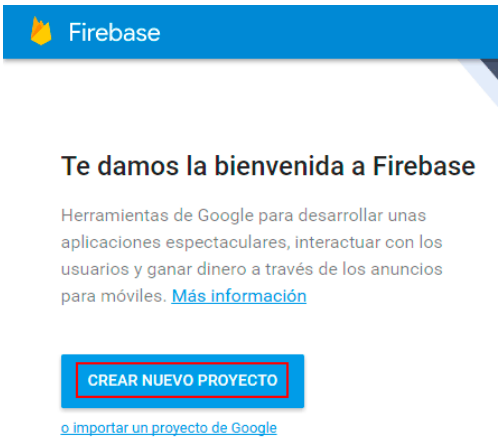


Figura 31: Crear un nuevo proyecto en Firebase. Fuente: Del autor

Si pulsamos la opción de “CREAR NUEVO PROYECTO” el sistema solicitará un nombre identificativo y la región a la que perteneces.

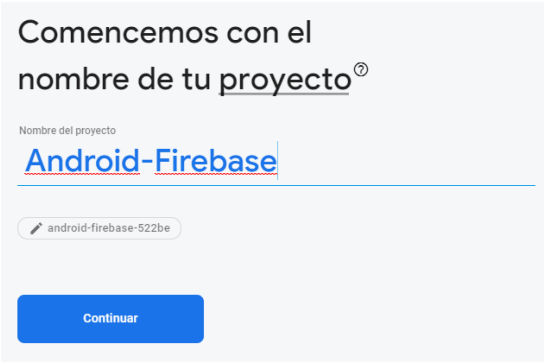


Figura 32: Nombre del proyecto en Firebase. Fuente: Del autor

Con estos simples datos quedaría creado el nuevo proyecto de Firebase. El siguiente paso será asociar una aplicación a dicho proyecto, en nuestro caso una aplicación Android, para lo que pulsaremos en la opción correspondiente a dicho sistema.

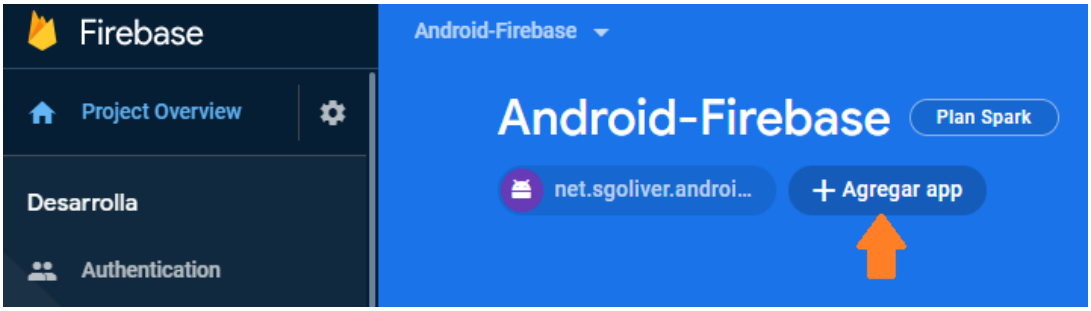


Figura 33: Agregar App a Firebase. Fuente: Del autor

Luego escogemos a la plataforma con la que vamos a desarrollar la aplicación para este proyecto se seleccionará la plataforma de Android.

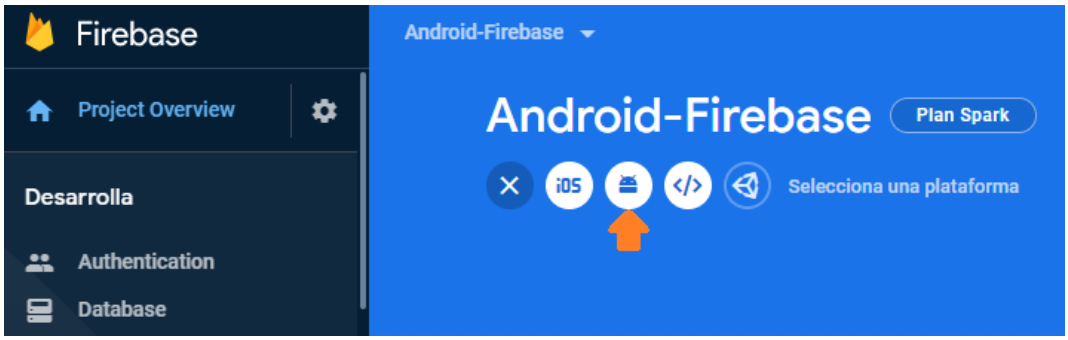


Figura 34: Plataforma Android con Firebase. Fuente: Del autor

Esto iniciará un pequeño asistente donde se solicitarán algunos datos como es el nombre del paquete java principal de nuestra aplicación Android que se encuentra en el archivo de Android Manifest y la huella digital SHA-1 del certificado con el que se firmará la aplicación, en principio usaremos como siempre la correspondiente al certificado de pruebas, pero se recuerda que se debe cambiar por el certificado de producción si se sube la aplicación a Google Play.

Toda aplicación Android debe ir firmada para poder ejecutarse en el dispositivo, tanto físico como emulado. Adicionalmente, durante el desarrollo de la misma, para realizar las pruebas y la depuración del código, aunque no seamos conscientes de ello también estamos firmado la aplicación con un certificado de pruebas. Pues bien, para obtener la huella digital del certificado con el que estamos firmando la aplicación podemos utilizar la utilidad keytool que se proporciona en el SDK de Java. Vamos a obtener el SHA1 del certificado de pruebas, que es el que se utilizará para probar en el emulador.

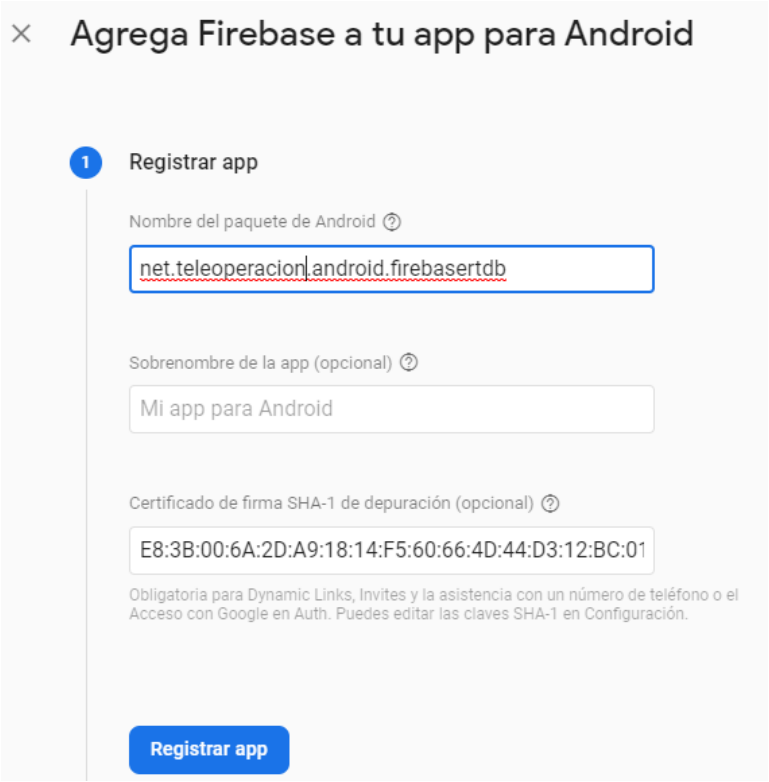


Figura 35: Agregar Firebase a la App para Android. Fuente: Del autor.

Para obtener el nombre del paquete debemos dirigimos al Android Manifest

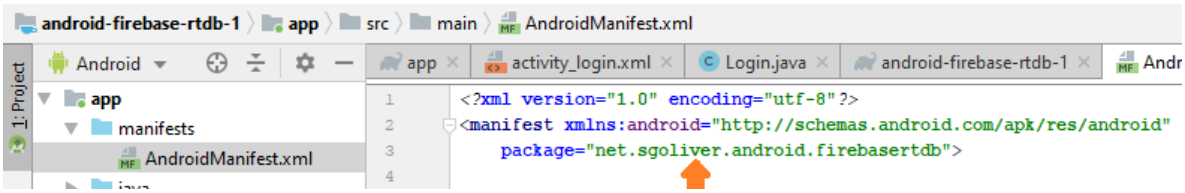


Figura 36: Nombre del paquete de la App Android. Fuente: Del autor

Obtención de la huella digital SHA-1 del certificado con el que se firmará la aplicación con los siguientes comandos:

```
cd C:\Program Files\Android\Android Studio\jre\jre\bin

keytool -list -v -keystore C:\Users\NOMBRE-EQUIPO\.android\debug.keystore -alias androiddebugkey -storepass android -keypass android
```

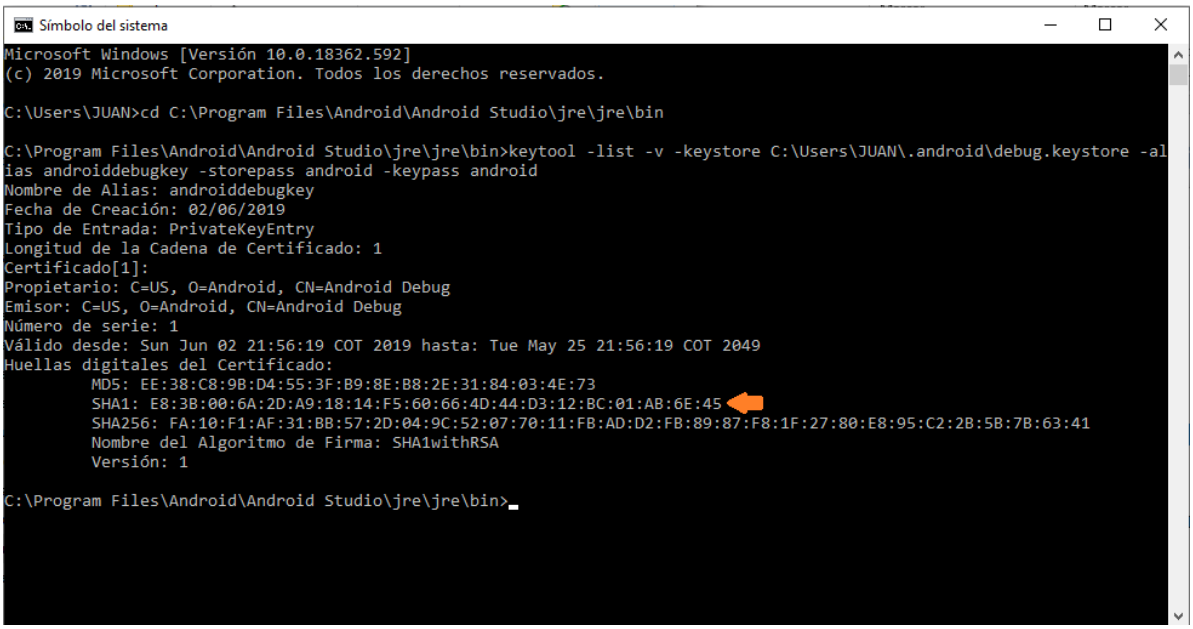


Figura 37: Generación de la huella digital SHA1. Fuente: Del autor

En este paso podrás descargar un fichero de configuración, en formato JSON, que tendremos que añadir a la aplicación Android una vez creemos el proyecto en Android Studio.

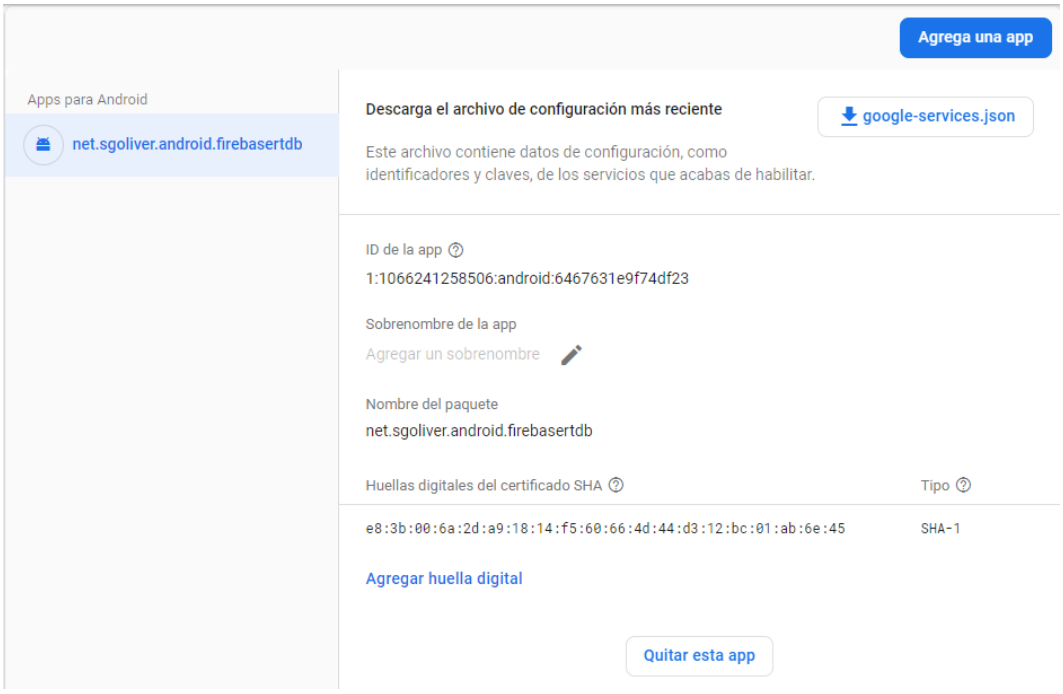


Figura 38: Google services en formato JSON. Fuente: Del autor

No se debe confundir una entidad con otra. Un mismo proyecto de Firebase puede tener asociadas varias aplicaciones, que además pueden ser de Sistemas distintos como: Android, iOS o Web.

Finalizados todos los preparativos en la consola de Firebase, podemos disponernos ya a crear nuestro proyecto en Android Studio. Crearemos un proyecto estándar, con API mínima 15 y utilizando la plantilla Empty Activity.

Creado el proyecto lo primero que vamos a hacer es colocar el fichero de configuración google-services.json que hemos descargado antes en su ubicación correcta. Debemos copiarlo a la carpeta "/app" situada dentro de la carpeta del proyecto.

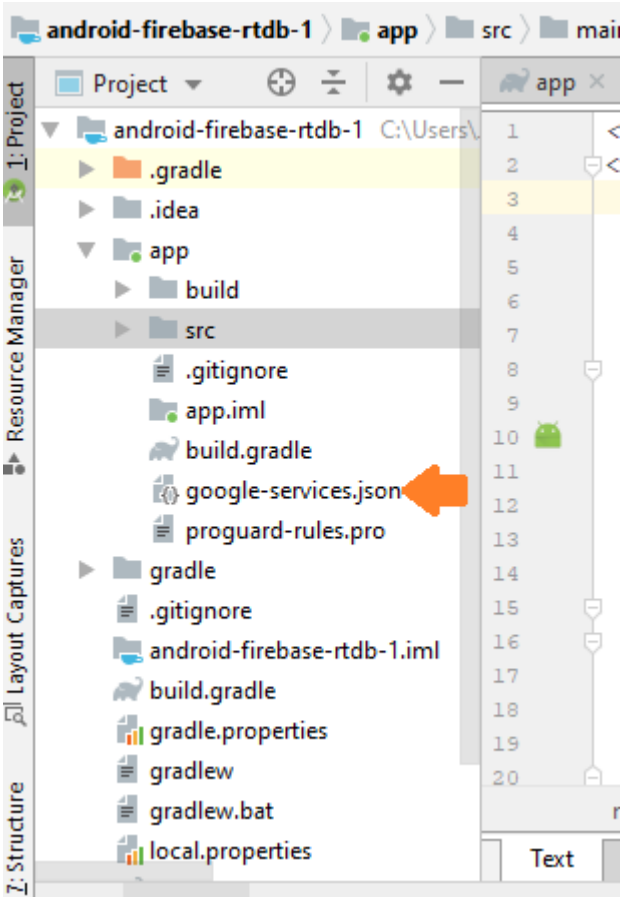


Figura 39: Google-services.json. Fuente: Del autor

El siguiente paso será añadir todas las dependencias de los servicios de Google en el build.gradle. Tendremos que modificar tanto el fichero build.gradle (Project: android-firebase-rtdb-1) situado a nivel de proyecto, como el build.gradle (Module: app) que es el módulo principal.

Dependencias build.gradle (Project: android-firebase-rtdb-1)

```
buildscript {
    repositories {
        jcenter()
        maven {
            // Dá acceso al repositorio de Google
            url 'https://maven.google.com/'
            name 'Google'
        }
        google()
    }

    dependencies {
```

```
        classpath 'com.android.tools.build:gradle:3.4.1'
        // Dá acceso a los Services y APIs de Google
        classpath 'com.google.gms:google-services:3.0.0'
    }
}

allprojects {
    repositories {
        jcenter()
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Dependencias en el build.gradle a nivel módulo principal

```
apply plugin: 'com.android.application'

android {
    // Escogemos la version con la se va a compilar la App (26)
    compileSdkVersion 26
    defaultConfig {
        applicationId "net.sgoliver.android.firebasertdb"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:design:26.1.0'
    implementation 'com.android.support:support-annotations:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'android.arch.lifecycle:extensions:1.1.1'
    androidTestImplementation('com.android.support.test.espresso:espresso-
core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
```

```
    })
    implementation 'com.android.support:appcompat-v7:26.1.0'
    // Implementamos el Service Realtime Database
    implementation 'com.google.firebase:firebase-database:9.4.0'
    // Implementamos la Authentication de Firebase
    implementation 'com.google.firebase:firebase-auth:9.4.0'
    // Implementamos la Authentication de Facebook
    implementation 'com.facebook.android:facebook-login:[5,6)'
    // Implementamos la Authentication de SingIn de Gmail
    implementation 'com.google.android.gms:play-services-auth:9.4.0'
    testImplementation 'junit:junit:4.12'
}
apply plugin: 'com.google.gms.google-services'
```

Ahora tenemos que dirigirnos a la consola y pulsamos sobre la opción Database del menú lateral izquierdo accederemos al panel de administración de nuestra base de datos. A la derecha veremos la información distribuida en 4 pestañas: Datos, Reglas, Uso y Copias de Seguridad.

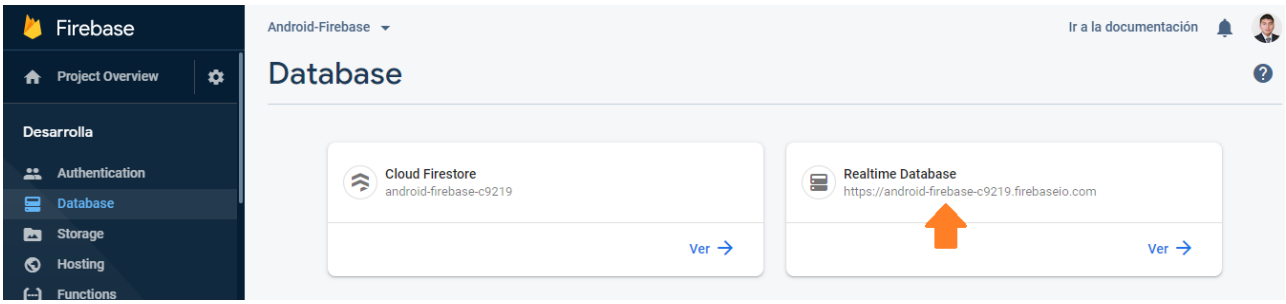


Figura 40: Realtime Database. Fuente: Del autor

Para el control de los movimientos del robot se crea un nodo y tendrá varios elementos hijo. Para ello volvemos a colocarnos encima del nodo raíz y pulsamos el símbolo (+) para añadir un nuevo elemento hijo. Le asignamos la clave Teleoperation, y sin indicar ningún valor pulsamos sobre su botón (+) para añadir sus elementos hijos, para este caso como elementos se tendrán los parámetros para teleoperar el brazo robótico como son: la base, codo, hombro, muñeca pitch, muñeca yaw las cuales representan el valor del ángulo en grados de cada articulación, el gripper con el que se va a controlar la herramienta de trabajo del robot en este caso sería el abierto y cerrado, estado del robot con el que se controlará el encendido y apagado y finalmente una variable robot para asignarle un nombre al brazo robótico como se ve en la figura 41.

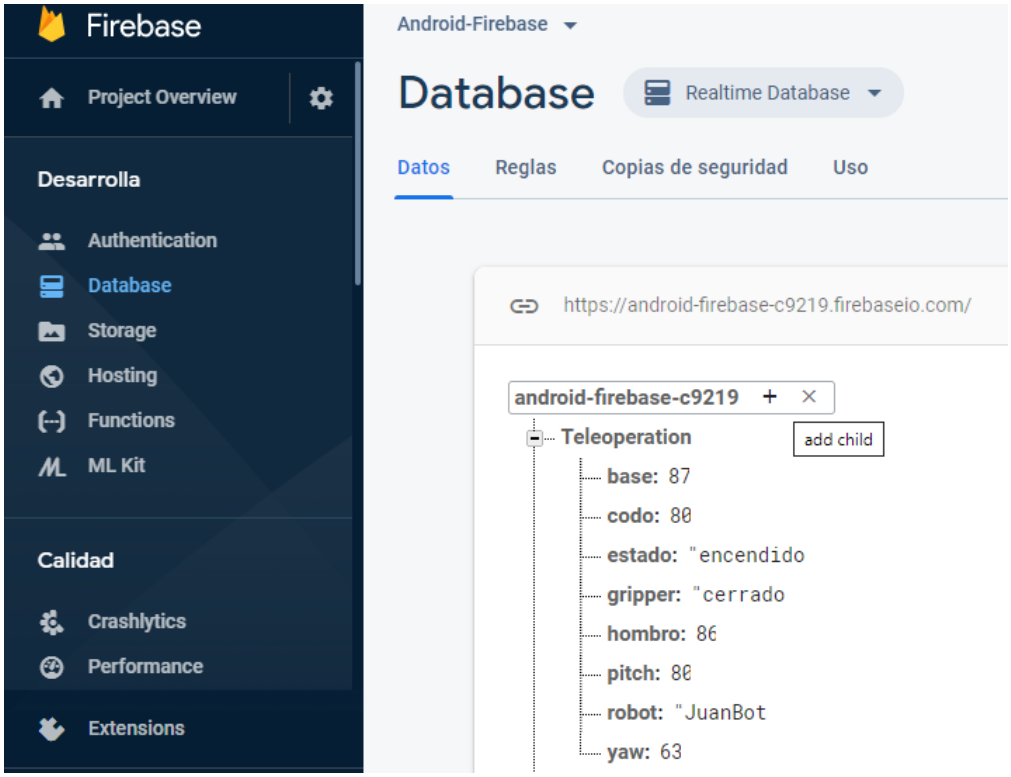


Figura 41: Nodo de teleoperación con Realtime Database. Fuente: Del autor

Cuando utilizamos la base de datos en tiempo real de Firebase la estrategia de obtención de información será distinta. Ya no necesitaremos que la aplicación consulte en distintos momentos un determinado dato por si éste ha cambiado, sino que directamente se suscribirá a dicho dato de forma que se notifique y reciba un nuevo valor automáticamente cada vez que éste cambie. Se podría decir de alguna forma que con las bases de datos tradicionales la iniciativa la debe llevar siempre la aplicación, y con Firebase gran parte de esa iniciativa pasa al lado de la base de datos. De esta forma, cualquier cambio que se produzca en los datos cuando un cliente actualiza parte de la información se trasladará de forma automática e inmediata a todos los clientes interesados en dicho dato, sin necesidad de implementar este mecanismo de notificación y refresco por nuestra parte.

Ahora se inicia el diseño de las interfaces de la aplicación móvil en Android Studio. Lo primero que haremos será crear una layout muy sencillo para nuestra pantalla principal, donde se mostrará los mandos de control del brazo robótico para el control de la teleoperación, ver la figura 42.

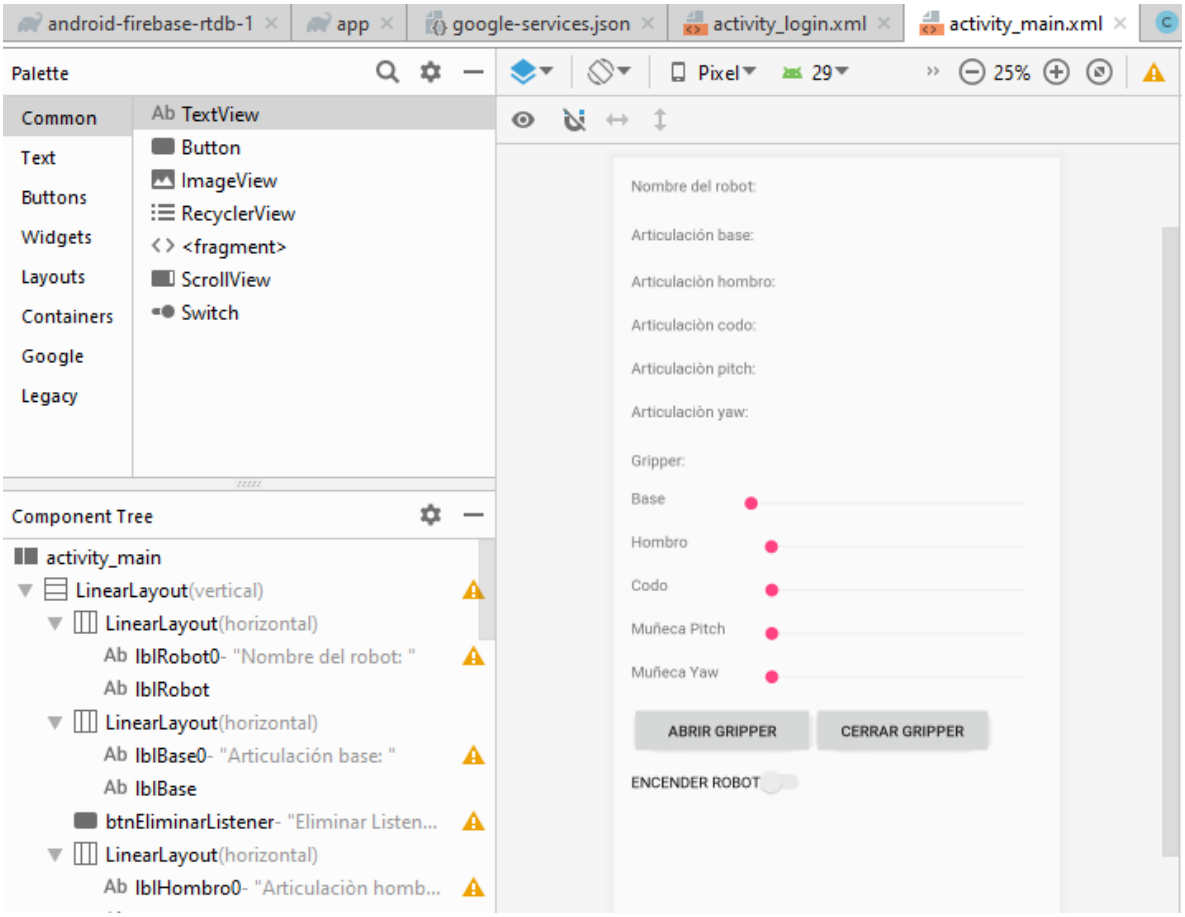


Figura 42: Activity para el control teleoperado del brazo robótico. Fuente: Del autor

Luego se tiene que realizar las configuraciones de Firebase en la clase MainActivity de Java. Lo primero que se tiene que hacer es crear una referencia a la base de datos. Esta referencia podría entenderse como un indicador que apunta a un lugar concreto de nuestra base de datos, aquel en el que estemos interesados. Aunque es posible definir una referencia al elemento raíz de la base de datos, casi nunca necesitaremos acceder a la totalidad de los datos, sino que bastará con un fragmento o sección concreta o, dicho de otra forma, solo necesitaremos los datos que cuelguen de un determinado nodo de nuestro árbol JSON.

Por empezar se crea una referencia al nodo base de forma que podamos suscribirnos a él para conocer su valor actual y estar informados de sus posibles cambios. Para ello crearemos en primer lugar un objeto de tipo DatabaseReference que almacenará la referencia a la base de datos. Para obtener una instancia a la base de datos de Firebase mediante getInstance() y después una referencia al nodo raíz de los datos con getReference() sin parámetros. A partir de esta primera referencia podemos crear cualquier otra más específica navegando entre los nodos

hijo mediante el método child(), que recibe como parámetro el nombre del subnodo al que queremos bajar en el árbol. En nuestro caso tendremos que acceder primero al subnodo Teleoperation como se muestra en el siguiente recuadro.

```
dbPrediccion =
    FirebaseDatabase.getInstance().getReference()
        .child("Teleoperation");
```

Obtenida la referencia ya podríamos suscribirnos a ella asignándole un listener de tipo ValueEventListener. Este listener tendrá dos métodos:

- onDataChange(). Se llamará automáticamente cada vez que se actualice la información del nodo actual o se produzca cualquier cambio en cualquiera de sus nodos descendientes. Se llamará por primera vez en el momento de suscribirnos, de forma que recibamos el valor actual del nodo y todos sus descendientes.
- onCancelled(). Se llamará cuando la lectura de los datos sea cancelada por cualquier motivo por ejemplo, porque el usuario no tiene permiso para acceder a los datos.

El primero de los métodos es por supuesto el más interesante, ya que es el que permitirá estar al tanto de cualquier cambio que se produzca en la información contenida a partir de la localización a la que apunta nuestra referencia. Como parámetro de este método recibiremos siempre un objeto de tipo DataSnapshot que contendrá toda la información del nodo. Un objeto DataSnapshot representa básicamente una rama del árbol JSON, es decir, contendrá toda la información de un nodo determinado, con su clave, su valor, y su listado de nodos hijos que a su vez pueden tener otros descendientes, por el que podremos navegar libremente. Podremos obtener la clave y el valor mediante los métodos getKey() y getValue() respectivamente. Los subnodos los podremos obtener en su totalidad mediante getChildren() los recibiremos en forma de listado de objetos DataSnapshot o bien podremos navegar a subnodos concretos mediante child("nombre-subnodo").

Cada vez que se llame al método onDataChange() simplemente recuperaremos el valor del nodo con getValue() y actualizaremos el campo correspondiente de la interfaz de la aplicación.

```
eventListener = new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {

lblBase.setText(dataSnapshot.child("base").getValue().toString()+ "°");

        Log.e(TAGLOG, "onDataChange:" +
dataSnapshot.getValue().toString());
```

```
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        Log.e(TAGLOG, "Error!", databaseError.toException());
    }
};
```

Si ejecutamos de nuevo la aplicación se podrán verificar los datos que asignaron desde la consola de Firebase, también podemos probar la aplicación móvil desde la PC usando el emulador NOX 6, como se muestra en el anexo G:

Figura 43: Datos en tiempo real entre la App y Firebase. Fuente: Del autor

Para iniciar la aplicación móvil, se implementó un Login con el cual se podrá ingresar a los mandos de control para la teleoperación del brazo robótico a través de la herramienta Authentication de Firebase, de esta manera se tendrá un alto nivel de seguridad hacia el funcionamiento de la misma. A continuación, se muestra los métodos de acceso hacia la aplicación, los cuales son: por correo/contraseña, Google y Facebook.

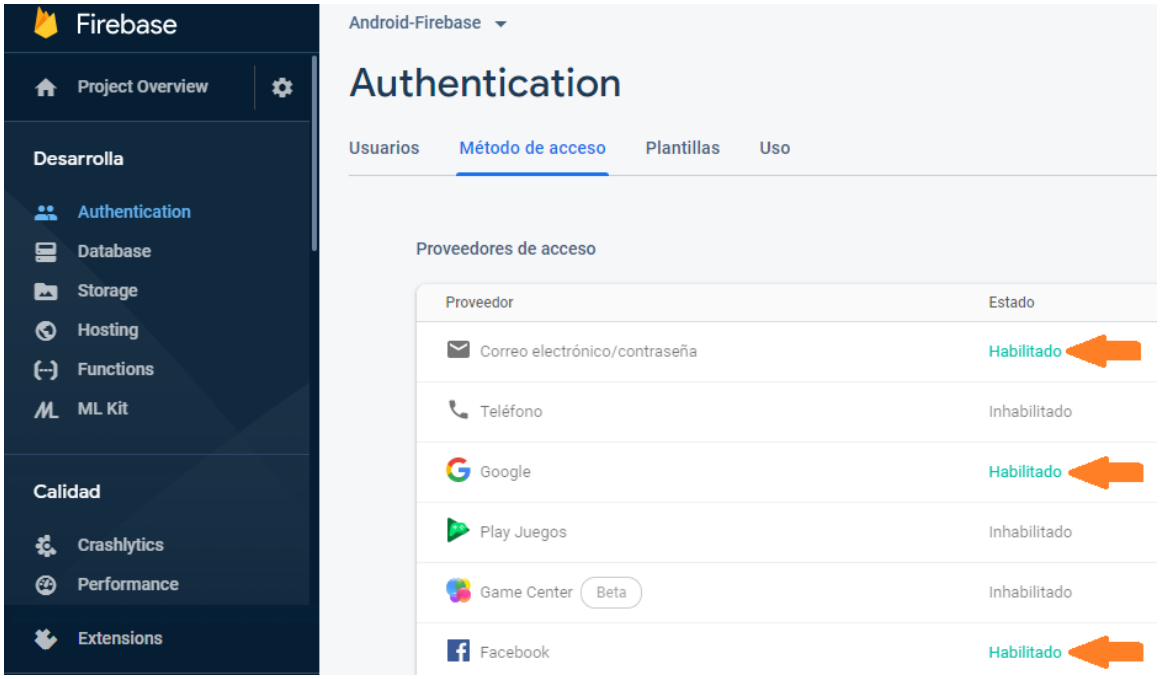


Figura 44: Métodos de acceso en Authentication de Firebase. Fuente: Del autor

Una vez realizado las configuraciones con cada método de acceso dentro de Authentication de Firebase se empieza el desarrollo de la actividad dentro de la aplicación móvil, y se agregarán los elementos dentro del Login para obtener el diseño como se puede observar en la figura 45. Además, se presenta la codificación de la aplicación de todas sus activities y functions desde el anexo H hasta el P.

Figura 45: Login de acceso de la aplicación móvil. Fuente: Del Autor

Esp8266 Firebase Connection

Ésta es la configuración que se utilizará en la placa NodeMCU – ESP8266, se necesita los siguientes componentes:

- ESP8266 (NodeMcu v3)
- Cuenta de google (firebase)

Paso 1: Configurar Arduino IDE, instalar la placa Esp8266:

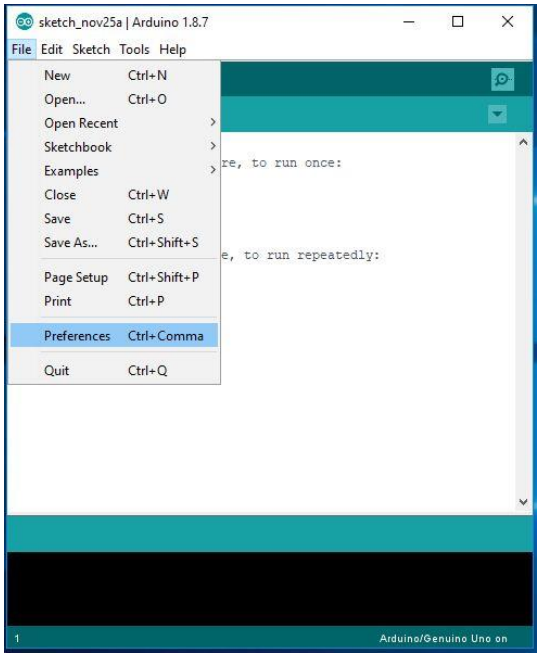


Figura 46: Configuración de ESP8266 en IDE Arduino paso 1. Fuente: Del autor

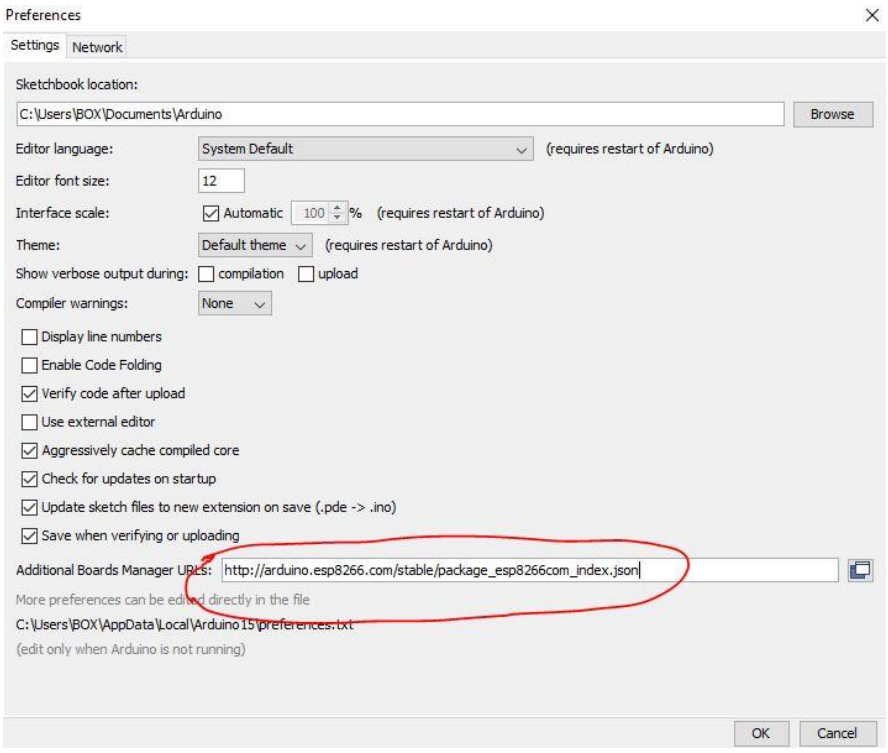


Figura 47: Configuración de ESP8266 en IDE Arduino paso 2. Fuente: Del autor

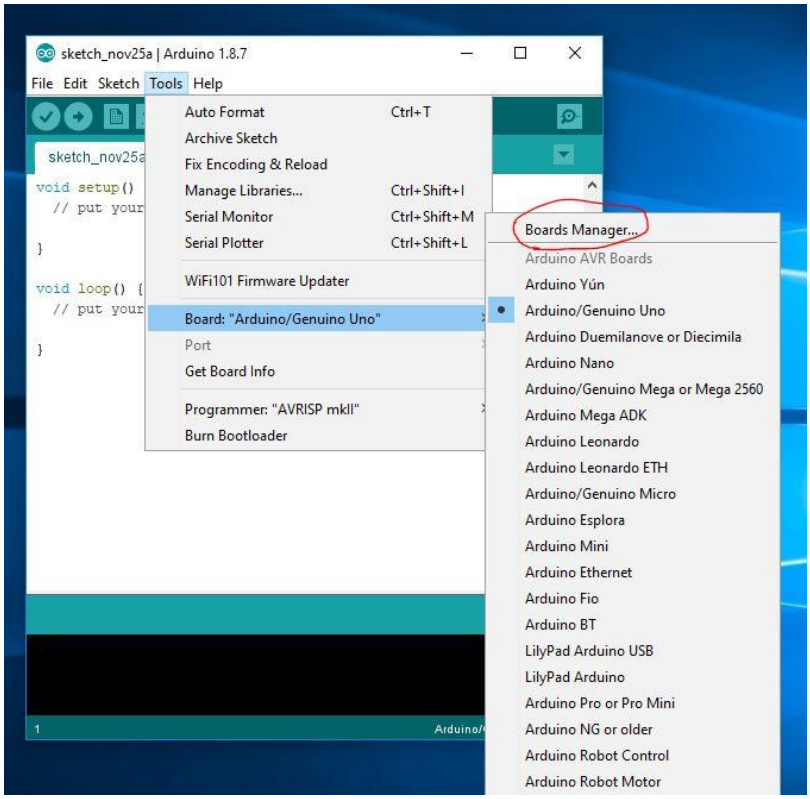


Figura 48: Configuración de ESP8266 en IDE Arduino paso 3. Fuente: Del autor

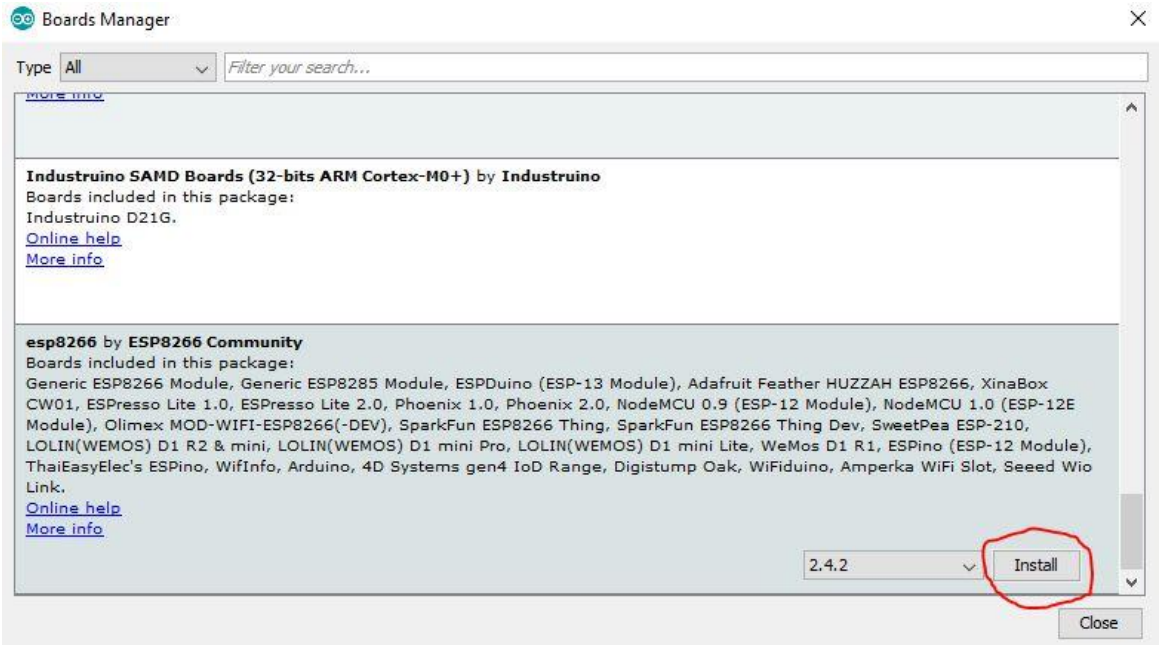


Figura 49: Configuración de ESP8266 en IDE Arduino paso 4. Fuente: Del autor

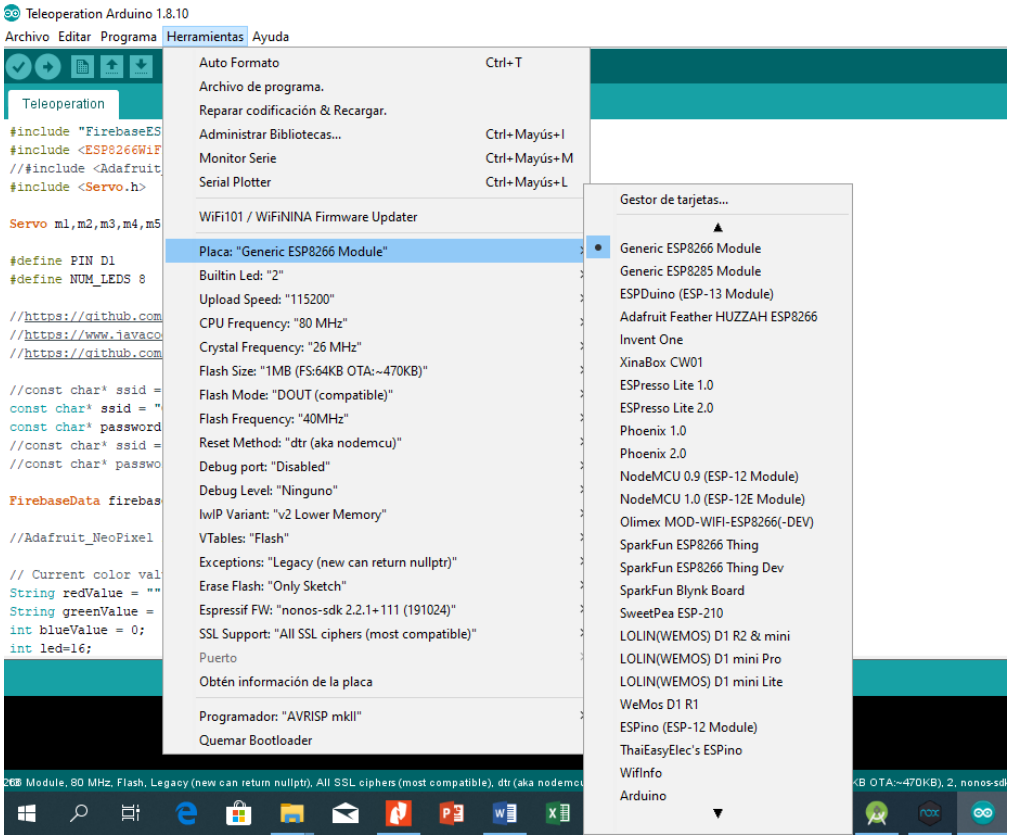


Figura 50: Configuración de ESP8266 en IDE Arduino paso 5. Fuente: Del autor

Paso 2: Configuración de Arduino IDE, Instale la Biblioteca Firebase ESP8266 Cliente

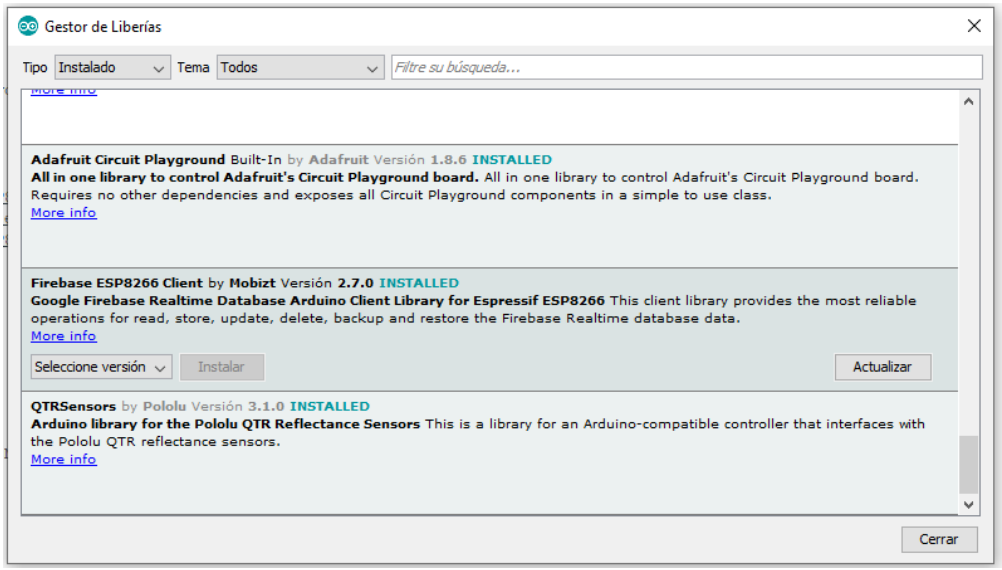


Figura 51: Añadir librería Firebase ESP8266 Client. Fuente: Del autor

Paso 3: Se inicializa las librerías que se usarán la teleoperación como se muestra:

```
//Librería de conexión con Firebase
//https://github.com/mobizt/Firebase-ESP8266
#include "FirebaseESP8266.h"
//Librería de la placa Wifi ESP8266
#include <ESP8266WiFi.h>
//Librería de servomotores
#include <Servo.h>
```

Paso 4: Se inicializa la comunicación de la tarjeta ESP8266 con una red Wifi como se muestra:

```
//Configuramos la red Wifi a la que se va a conectar
const char* ssid = "Wifi_Israel";
const char* password = "uisrael2015";
void setup() {
  Serial.begin(115200);
  connectWifi();
}
void connectWifi() {
  // Let us connect to WiFi
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println(".......");
  Serial.println("WiFi Connected....IP Address:");
  Serial.println(WiFi.localIP());

}
```

Paso 5: Buscamos el Firebase Host y Firebase Auth dentro de nuestro proyecto de Firebase para poder añadir en la configuración del controlador ESP8266

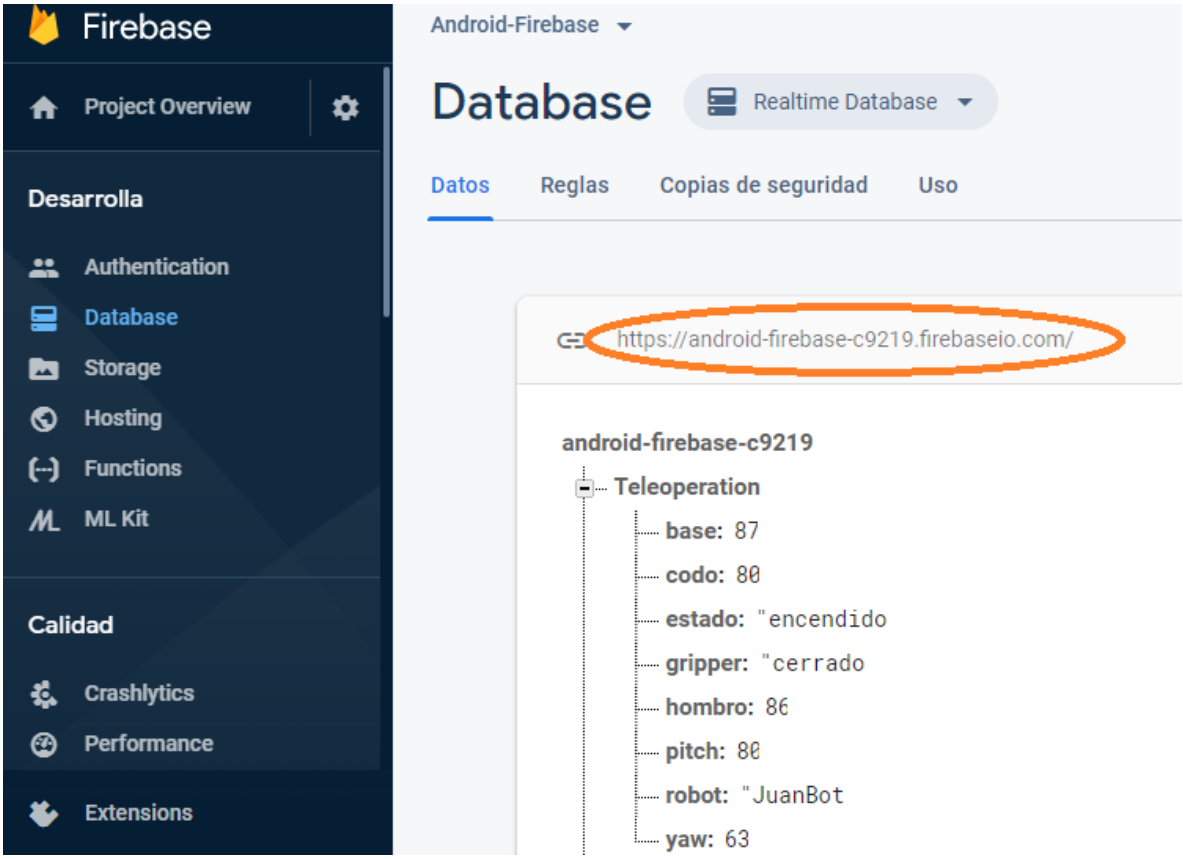


Figura 52: Firebase Host del Realtime Database. Fuente: Del autor

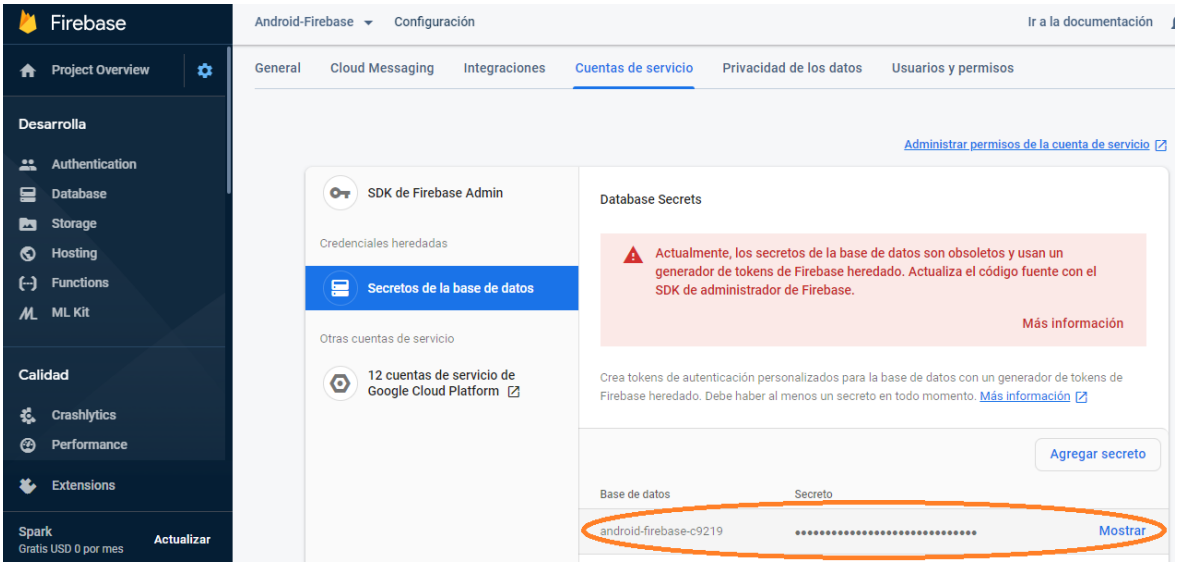


Figura 53: Firebase Auth, secretos de la base de datos. Fuente: Del autor

Paso 6: Se inicializa la comunicación entre la tarjeta ESP8266 con Firebase como se muestra:

```
//Creamos el objeto Firebase
FirebaseData firebaseData;

void setup() {
//Configuramos el objeto que se conectará a nuestro proyecto de
//Firebase
  Firebase.begin("https://android-firebase-c9219.firebaseio.com/",
"Q9886kW2ODbTVL9tAAkd2a7MMHht2Ja5j6LZvude");

}
```

Paso 7: Modo de acceso a los mandos de control de la teleoperación entre la tarjeta ESP8266 y Firebase Realtime Database:

```
void loop() {

  if (Firebase.getInt(firebaseData, "/Teleoperation/base")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m1.write(val);

    }
  }

  if (Firebase.getString(firebaseData, "/Teleoperation/estado")) {
    if  (firebaseData.dataType() == "string") {
      String val = firebaseData.stringData();
      if (val == "encendido") {
        digitalWrite(robot,HIGH);
      }
      if (val == "apagado") {
        digitalWrite(robot,LOW);
      }
    }
  }
}
```

Toda la codificación en IDE de Arduino para el control teleoperado del robot se adjunta en el Anexo Q.

3.8. Integración del hardware con el software según los protocolos de comunicación

Para continuar se prueban la comunicación entre el dispositivo móvil y la plataforma Firebase, por otro lado, también se realiza la comunicación entre el controlador del robot y la

plataforma Firebase de tal forma que se pueda confirmar que exista una correcta correspondencia entre ambos y que el sistema interactúe de manera eficiente.

Para integrar la teleoperación se usará los protocolos de comunicación para IoT, como mencionó (Chang, 2019) en su artículo dice que:

> Parte del auge del IoT es la popularidad de Internet y los Smartphone, la mejora de los sistemas de comunicación, y la aparición de dispositivos con conectividad pequeños, baratos y de bajo consumo como el ESP8266 o el ESP32. La comunicación entre dispositivos es la piedra central del IoT.

Sobre los protocolos de esta arquitectura en la nube (Google Cloud, 2020) menciona que:

> Cloud IoT Core admite dos protocolos para la conexión y comunicación del dispositivo: MQTT y HTTP. Los dispositivos se comunican con Cloud IoT Core a través de un "puente", ya sea el puente MQTT o el puente HTTP. El puente MQTT / HTTP es un componente central de Cloud IoT Core, como se muestra en la figura 54.

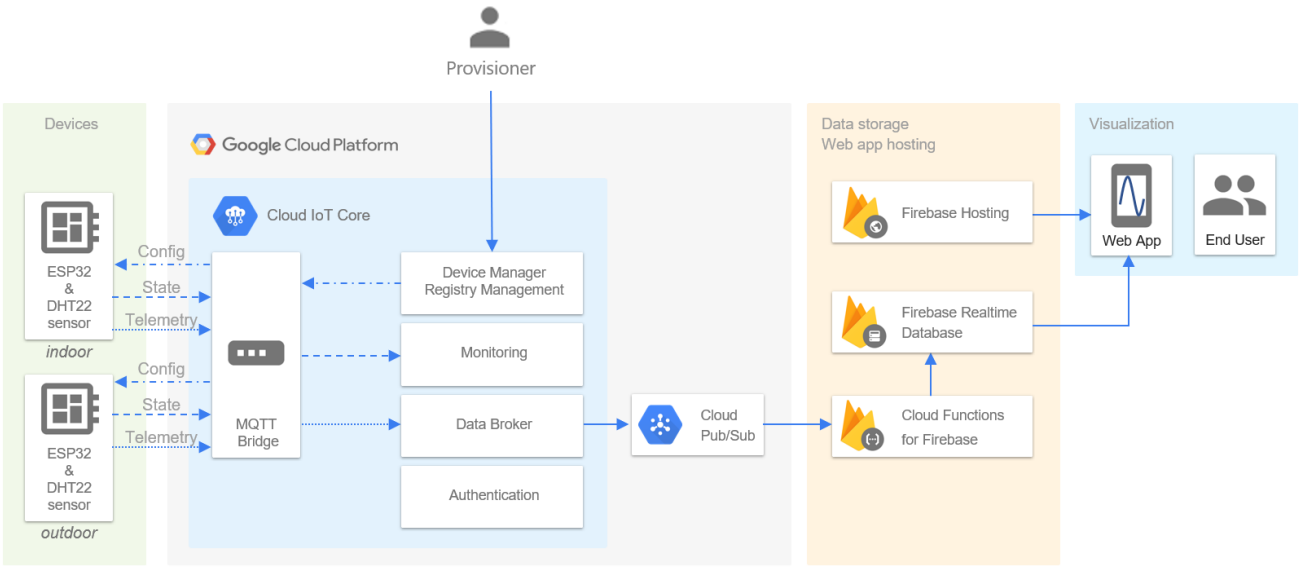


Figura 54: Arquitectura del proyecto: ESP32, Cloud IoT Core, Firebase y App Móvil. Fuente: Olivier Lourme (2019). Obtenido de: https://medium.com/free-code-camp/gcp-cloudiotcore-esp32-mongooseos-1st-5c88d8134ac7

Cuando crea un registro de dispositivo, selecciona protocolos para habilitar: MQTT, HTTP o ambos.

MQTT es un protocolo estándar de publicación / suscripción que los dispositivos integrados utilizan y admiten con frecuencia, y también es común en las interacciones de máquina a máquina.

HTTP es un protocolo "sin conexión": con el puente HTTP, los dispositivos no mantienen una conexión con Cloud IoT Core. En cambio, envían solicitudes y reciben respuestas. Cloud IoT Core solo admite HTTP 1.1 (no 2.0).

Considerar las siguientes características generales de cada protocolo en la siguiente tabla:

Tabla 7: Características del protocolo MQTT y HTTP.

MQTT	HTTP
• Menor uso de ancho de banda • Baja latencia • Mayor rendimiento • Admite datos binarios sin procesar	• Peso más ligero (fácil de comenzar; comandos de curvatura simples) • Menos problemas de firewall • Los datos binarios deben estar codificados en base64, lo que requiere más recursos de red y CPU

Nota. Adaptada por (Google Cloud, 2020). Obtenido de: https://cloud.google.com/iot/docs/concepts/protocols

Para este proyecto se usará el Google Cloud IoT Core como Broker, ya que interactúa con dispositivos ESP8266 y el Cloud Functions de la plataforma Firebase que a su vez interactúa con aplicaciones móviles.

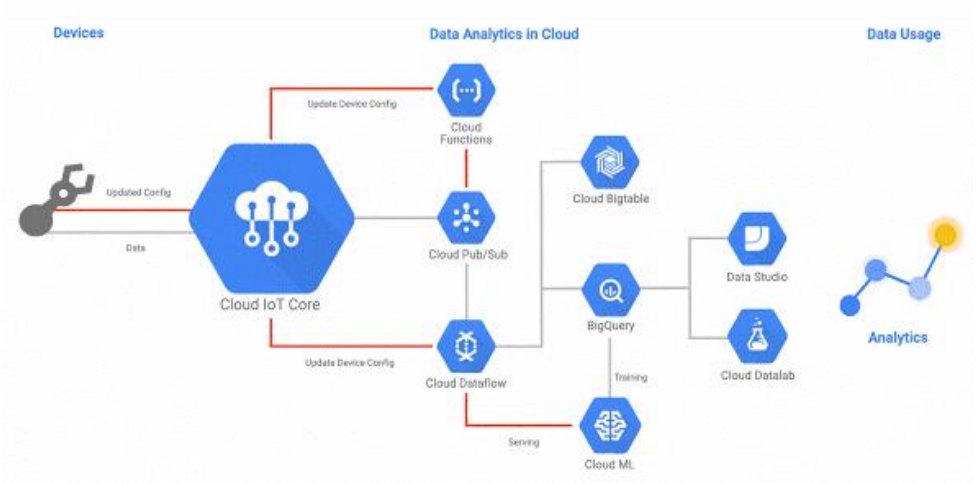


Figura 55: Flujo de datos a través de los servicios de Google Cloud Platform
Fuente: Alvaro Viebrantz (2019). Obtenido de: https://medium.com/google-cloud/build-a-weather-station-using-google-cloud-iot-core-and-mongooseos-7a78b69822c5

3.9. Implementación del sistema integrado en tiempo real para la teleoperación de un brazo robótico en la plataforma Firebase y obtención de resultados finales

En esta última fase, se integran todos los dispositivos con sus respectivos protocolos de comunicación y los programas que se compilan, para llevar a cabo el funcionamiento general del sistema, y realizar las pruebas de rendimiento, latencia e integridad de la información para obtener los resultados finales.

Una vez que se ha culminado con el desarrollo de un sistema integrado para la operación de un brazo robótico teleoperado en tiempo real mediante la plataforma Firebase con el uso de dispositivos móviles, se obtuvo lo siguiente:

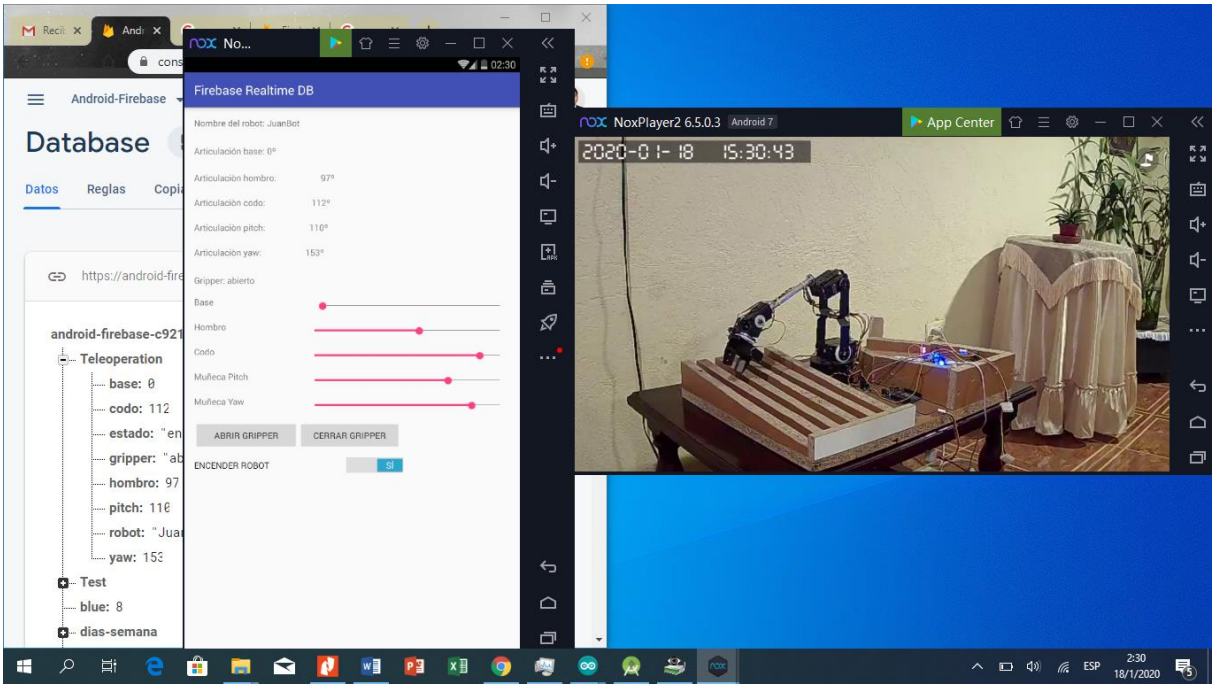


Figura 56: Sistema robótico teleoperado en tiempo real. Fuente: Del autor

Luego de realizar las pruebas de la aplicación móvil dentro de la plataforma Firebase se obtuvo un historial del número de usuarios o dispositivos nuevos en los cuales se instaló la aplicación móvil, también se registra el número de eventos que son los mandos de teleoperación que se envía hacia Firebase como se puede observar en la figura 57 y 58 respectivamente:

Figura 57: Registro de uso de la aplicación móvil. Fuente: Reporte de Firebase y Google Analytics

Figura 58: Usuarios por país. Fuente: Reporte de Google Analytics

Dentro de Firebase Analytics se puede obtener información sobre el tipo de dispositivo y el porcentaje de uso de cada función que tiene la aplicación como se puede ver en la figura 59.

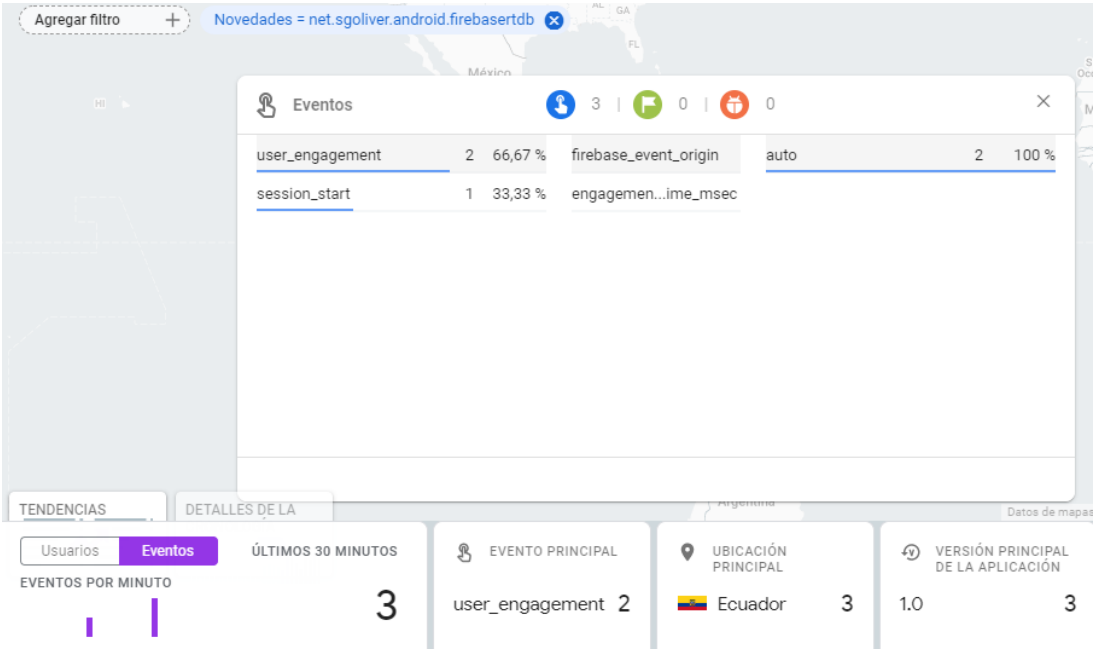


Figura 59: Firebase Analytics. Fuente: Firebase Platform

También se analizan los datos que otorga Facebook Analytics que es una herramienta que forma parte de Facebook developers, con la cual implementamos un modo de ingreso de usuarios con Authentication y le permita ingresar con una cuenta de Facebook a la aplicación

y dio reportes como número de usuarios que ingresaron con sus cuentas, tiempo promedio que demoran en ingresar y algunos datos destacados como se muestra en la figura 60.

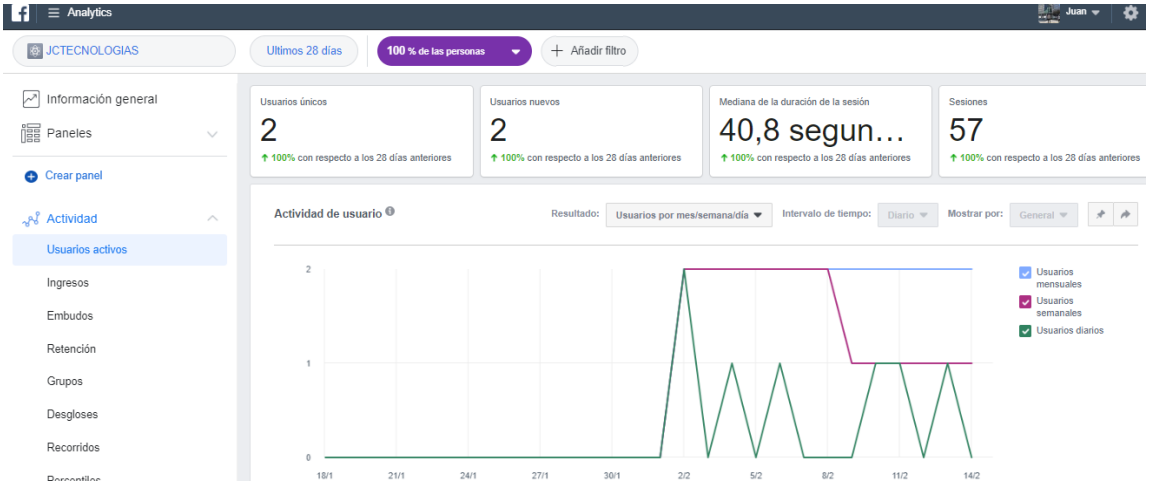


Figura 60: Registro de ingreso con cuenta de Facebook. Fuente: Facebook Analytics

Dentro de Authentication de Firebase en el modo de ingreso por correo y contraseña se obtuvo el reporte que se muestra en la figura 61.

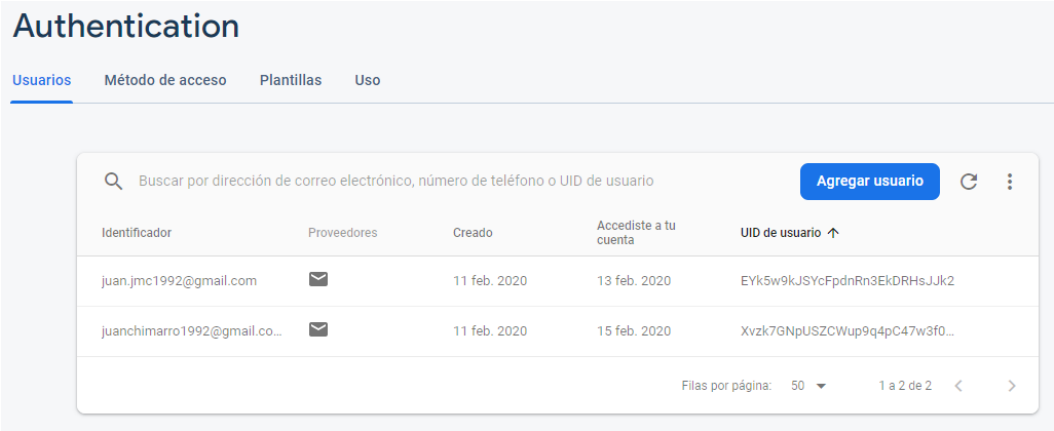


Figura 61: Registro de usuarios en Authentication de Firebase. Fuente: Firebase Platform

Ahora obtendremos la información sobre el uso de la base de datos en Realtime Database de Firebase en la cual se obtendrá el número de conexiones simultáneas, almacenamiento de la base de datos, carga y descarga de datos como se muestra a continuación:

La cantidad de conexiones simultáneas en tiempo real a tu base de datos como se ve en la figura 62.

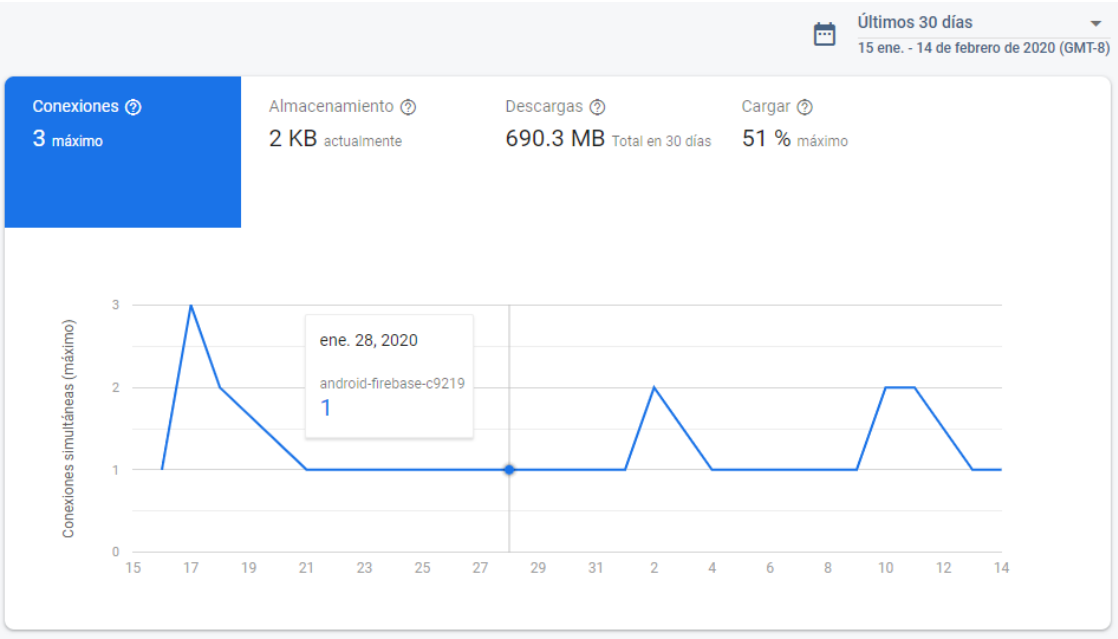


Figura 62: Conexiones simultáneas. Fuente: Realtime Database de Firebase

La cantidad de datos que se almacenan en tu base de datos, sin incluir el hosting de Firebase ni los datos almacenados mediante otras funciones de Firebase como se puede observar en la figura 63.

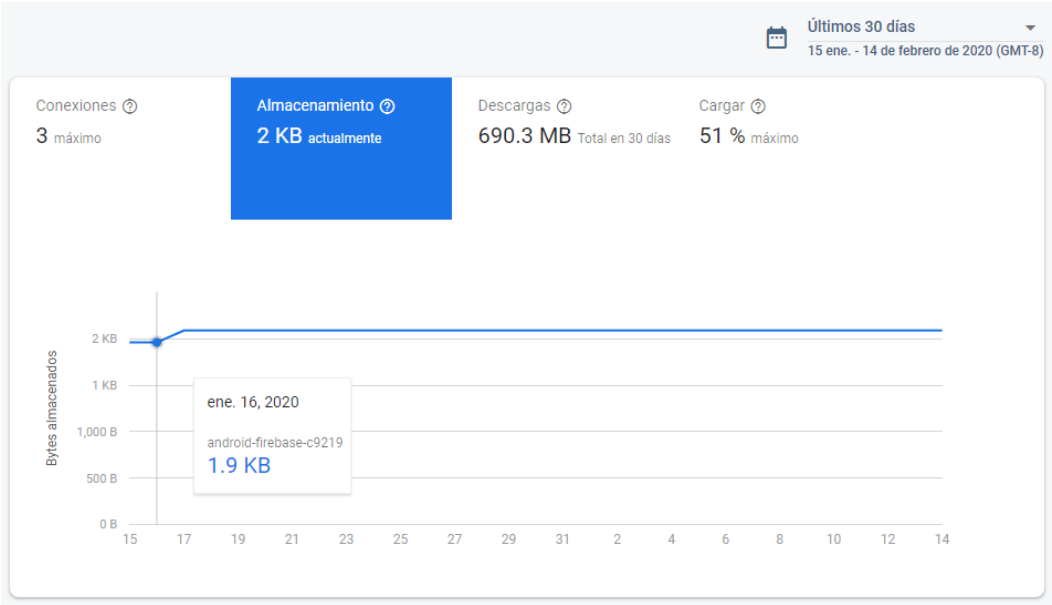


Figura 63: Almacenamiento de la base de datos. Fuente: Realtime Database de Firebase

Cuenta de descargas para todos los bytes descargados de tu base de datos, incluidos los gastos del protocolo de conexión y la encriptación SSL en intervalos de 1 minuto, ver la figura 64.

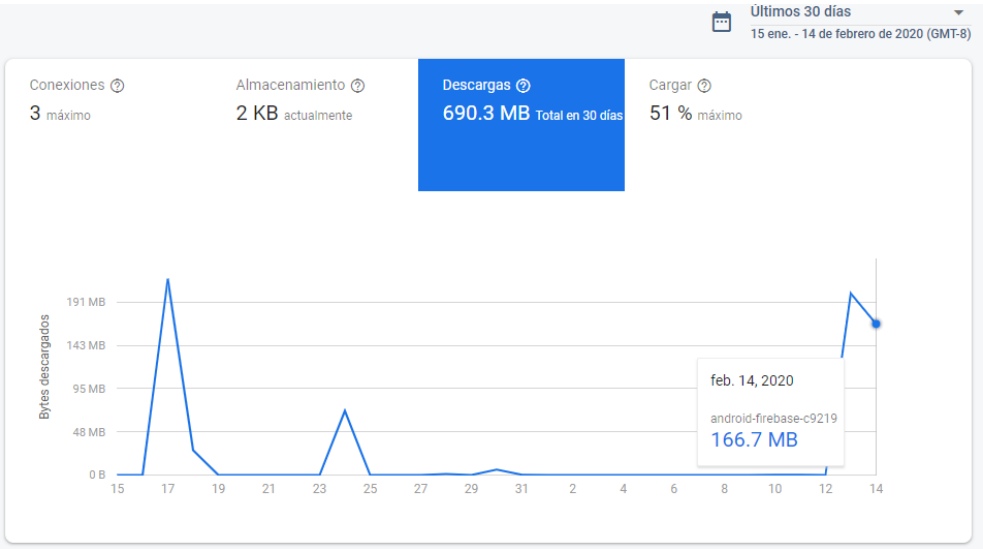


Figura 64: Descarga de datos desde le Realtime Database. Fuente: Realtime Database de Firebase

Porcentaje de la base de datos que está ocupado en el procesamiento de solicitudes (ya sea conexiones en tiempo real o solicitudes de la API de RESTful) durante intervalos de 1 minuto. Es posible que ocurran problemas de rendimiento a medida que te acerques al 100%, se muestra en la figura 65.

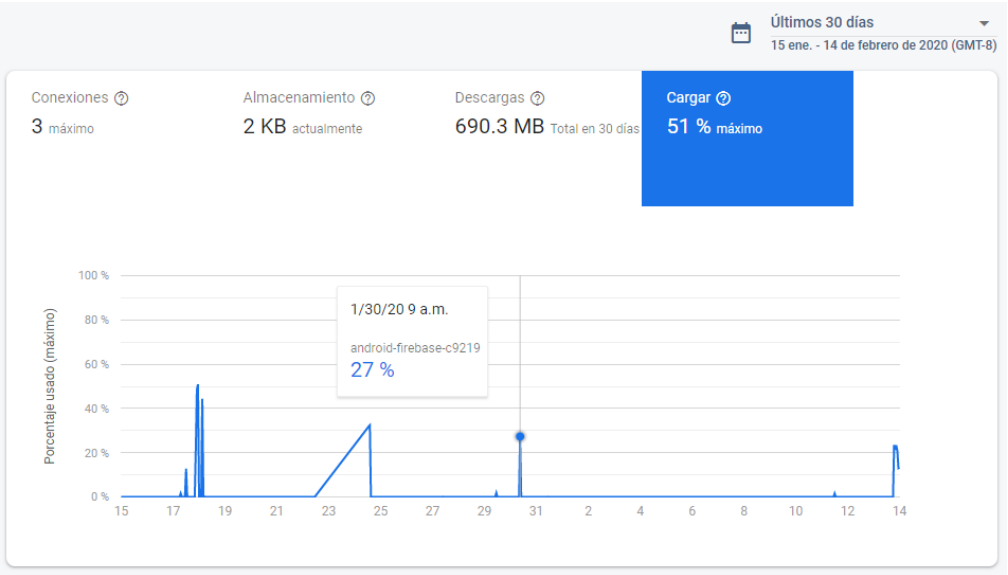


Figura 65: Carga de datos desde le Realtime Database. Fuente: Realtime Database de Firebase

Con respecto a la tarjeta ESP8266, se analizó el tiempo promedio de retardo que existe entre atributos, y el tiempo promedio de retardo de cada atributo cuando se produce un evento teleoperado, se muestra en las siguientes tablas 8 y 9 respectivamente. Para extraer los tiempos se usó la comunicación serial entre la tarjeta ESP8266 y la PC, ya que se comunican a una velocidad de 115200 Baudios y se pueden visualizar los datos en la interfaz serial del IDE de Arduino como se observa en la siguiente figura:

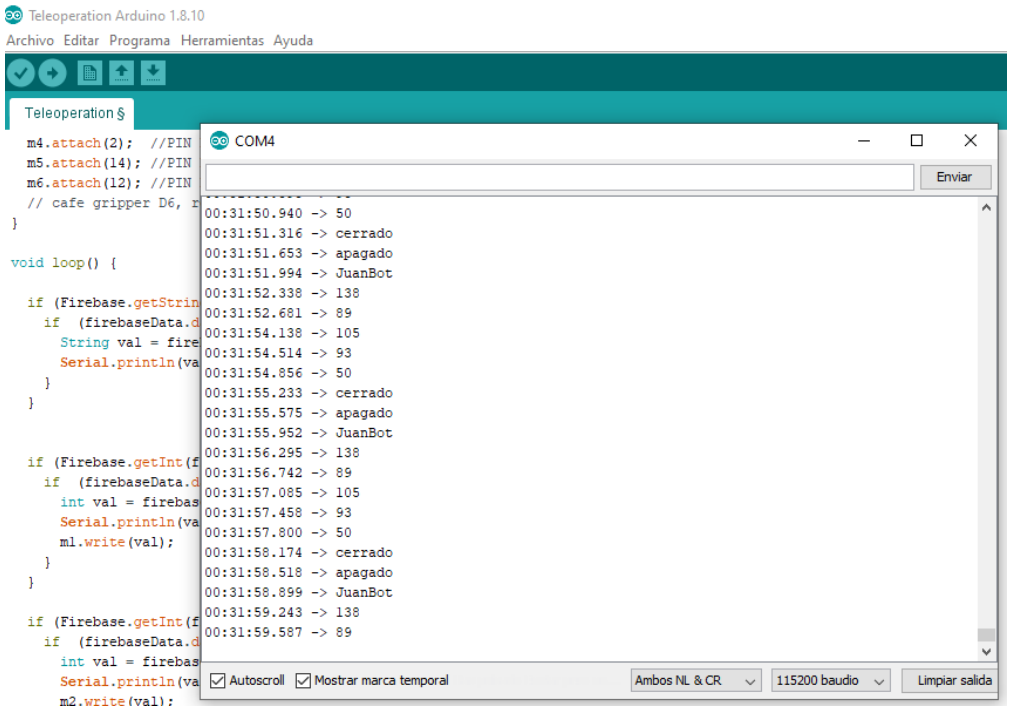


Figura 66: Salida de datos por el Serial del ESP8266. Fuente: Del autor

Tabla 8: Tiempo promedio de retardo que existe entre atributos

Iteración de datos por nodo	Hora de llegada del dato de Realtime al ESP8266	Atributo	Valor actual	Tiempo de retardo entre cada atributo (ms)	Promedio del tiempo de retardo entre cada atributo (ms)
1	21:18:22.422	Nombre robot	JuanBot	-	712,286
	21:18:23.213	Base	75°	791	
	21:18:23.863	Hombro	94°	650	
	21:18:24.689	Codo	69°	826	
	21:18:25.100	Pitch	70°	411	
	21:18:25.791	Yaw	122°	691	
	21:18:26.172	Gripper	cerrado	381	
	21:18:27.408	Estado del robot	apagado	1236	

Iteración de datos por nodo	Hora de llegada del dato de Realtime al ESP8266	Atributo	Valor actual	Tiempo de retardo entre cada atributo (ms)	Promedio del tiempo de retardo entre cada atributo (ms)
2	21:18:28.129	Nombre robot	JuanBot	721	660,375
	21:18:28.508	Base	75°	379	
	21:18:29.263	Hombro	94°	755	
	21:18:29.638	Codo	69°	375	
	21:18:30.735	Pitch	70°	1097	
	21:18:31.525	Yaw	122°	790	
	21:18:31.901	Gripper	cerrado	376	
	21:18:32.691	Estado del robot	apagado	790	
3	21:18:33.448	Nombre robot	JuanBot	757	563,000
	21:18:33.894	Base	75°	446	
	21:18:34.270	Hombro	94°	376	
	21:18:34.961	Codo	69°	691	
	21:18:35.721	Pitch	70°	760	
	21:18:36.098	Yaw	122°	377	
	21:18:36.478	Gripper	cerrado	380	
	21:18:37.195	Estado del robot	apagado	717	
4	21:18:38.021	Nombre robot	JuanBot	826	601,750
	21:18:38.673	Base	75°	652	
	21:18:39.428	Hombro	94°	755	
	21:18:39.809	Codo	69°	381	
	21:18:40.876	Pitch	70°	1067	
	21:18:41.256	Yaw	122°	380	
	21:18:41.632	Gripper	cerrado	376	
	21:18:42.009	Estado del robot	apagado	377	
5	21:18:42.389	Nombre robot	JuanBot	380	460,375
	21:18:42.767	Base	75°	378	
	21:18:43.554	Hombro	94°	787	
	21:18:43.934	Codo	69°	380	
	21:18:44.316	Pitch	70°	382	
	21:18:44.695	Yaw	122°	379	
	21:18:45.072	Gripper	cerrado	377	
	21:18:45.692	Estado del robot	apagado	620	
				Promedio total del tiempo de retardo entre cada atributo (ms)	**599,557**

Nota. Tabla adaptada por el autor

Como se puede observar en la tabla 8, se obtuvo un promedio de 599,557 milisegundos como retardo entre cada atributo lo cual representa un tiempo de latencia aceptable ya que se debe tomar en cuenta que se está trabajando en con una red Wifi con un ancho de banda de 4Mbps ya que se tiene un tiempo promedio de carga y descarga de 85 ms como se muestra en la figura 67. Adicionalmente se incluye el procesamiento y codificación de la información que debe realizar la tarjeta ESP8266.

Figura 67: Velocidad de la red Wifi CNT que usa la ESP8266. Fuente: Página oficial SpeedTest. Obtenido de: https://www.speedtest.net/result/9043471288

Tabla 9: El tiempo promedio de retardo de cada atributo cuando se produce un evento teleoperado.

Atributo	Hora de llegada del primer dato de Realtime al ESP8266	Valor inicio	Hora de llegada del siguiente dato de Realtime al ESP8266	Valor siguiente	Tiempo de retardo por cada atributo (ms)
Base	21:18:54.736	75°	21:18:57.962	138°	3226
Hombro	21:19:01.977	94°	21:19:05.343	89°	3366
Codo	21:19:02.354	69°	21:19:05.996	105°	3642
Pitch	21:19:09.709	70°	21:19:12.962	93°	3253
Yaw	21:19:10.086	122°	21:19:13.751	49°	3665
Gripper	21:19:17.455	cerrado	21:19:21.468	abierto	4013
Estado del robot	21:19:25.973	apagado	21:19:29.332	encendido	3359
				Promedio	**3503,429**

Nota. Tabla adaptada por el autor

Como se puede observar en la tabla 9 el tiempo promedio de retardo del evento que se produce en cada atributo es de 3503,429 milisegundos, eso indica que el tiempo de latencia es considerable por cada atributo lo cual puede representar una desventaja con respecto a la precisión, esto debido a la velocidad con la que trabaja el dispositivo móvil ya que se conecta a una red 3G o 4G dependiendo la zona, sin embargo, para el sistema teleoperado se usó una red móvil 4G con un ancho de banda de 10Mbps y un tiempo promedio de carga y descarga de 38 milisegundos para tener la comunicación de la manera más óptima posible como se puede verificar en la figura 68.

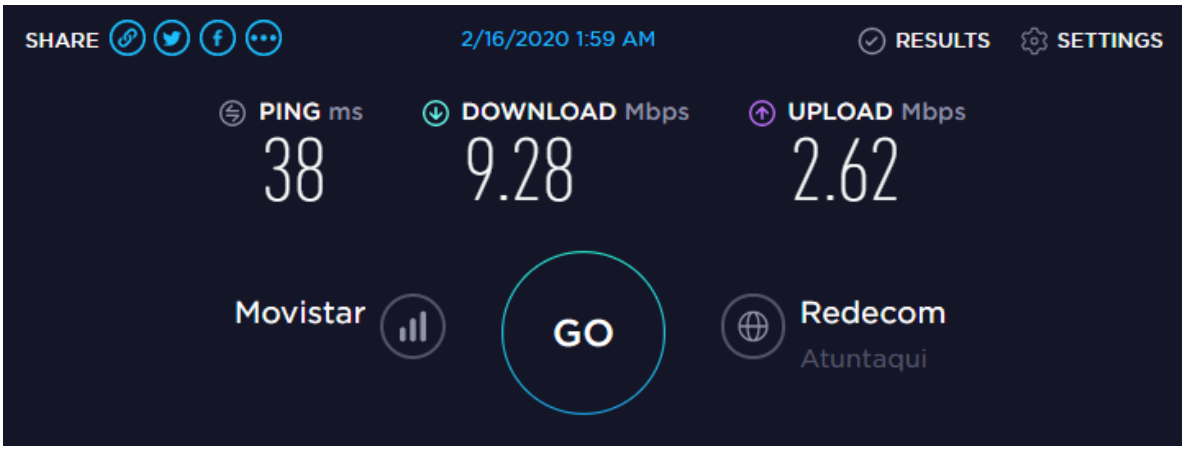


Figura 68: Velocidad de la red 4G Movistar que usa el dispositivo móvil. Fuente: Página oficial SpeedTest. Obtenido de: https://www.speedtest.net/result/a/5752021105

Finalmente se obtendrá los resultados de la API sobre el tráfico tele-transmitido, porcentaje de errores de ingreso a los controles de teleoperación y los tiempos de latencia que se tendrán a través de Google Cloud Platform que se muestran en las siguientes figuras.

En la figura 69 Google Identity Toolkit permite a los fabricantes de aplicaciones y sitios web admitir fácilmente múltiples opciones de autenticación para los usuarios finales. A través de este servicio se mide el tráfico con respecto al uso de la aplicación móvil. El servicio actualmente admite autenticación de contraseña además de inicio de sesión federado con Google, Facebook, Yahoo, Microsoft, Paypal y AOL. Incluso los desarrolladores no técnicos pueden agregar futuras opciones de inicio de sesión y migrar usuarios existentes con simples cambios de configuración.

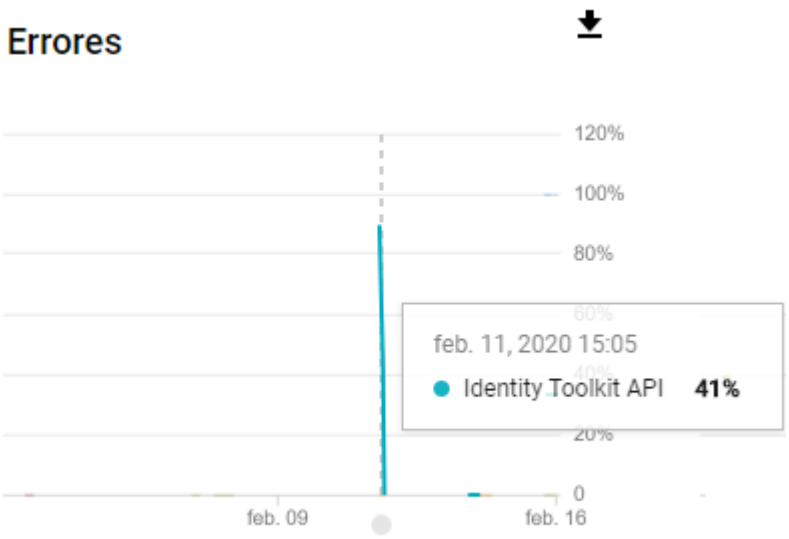
Métodos

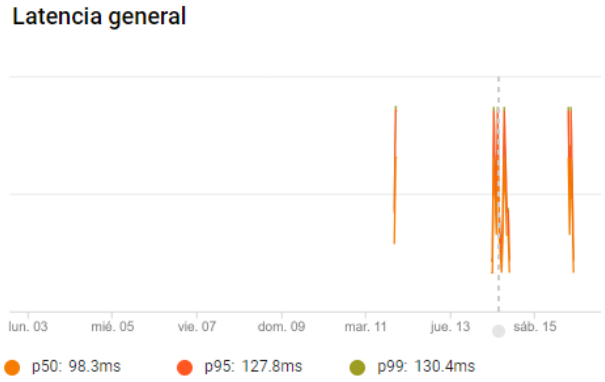
Método ↑	Solicitudes	Errores
identitytoolkit.accounts.lookup	17	0
identitytoolkit.accounts.resetPassword	2	0
identitytoolkit.accounts.signInWithPassword	33	0
identitytoolkit.accounts.signUp	55	0

Figura 69: Tráfico tele-transmitido de la Identity Toolkit API

Fuente: Google Cloud Platform

En la figura 70 se describe los datos del Token Service API le permite intercambiar un token de ID o un token de actualización por un token de acceso y un nuevo token de actualización. Puede usar el token de acceso para llamar de forma segura a las API que requieren autorización del usuario.



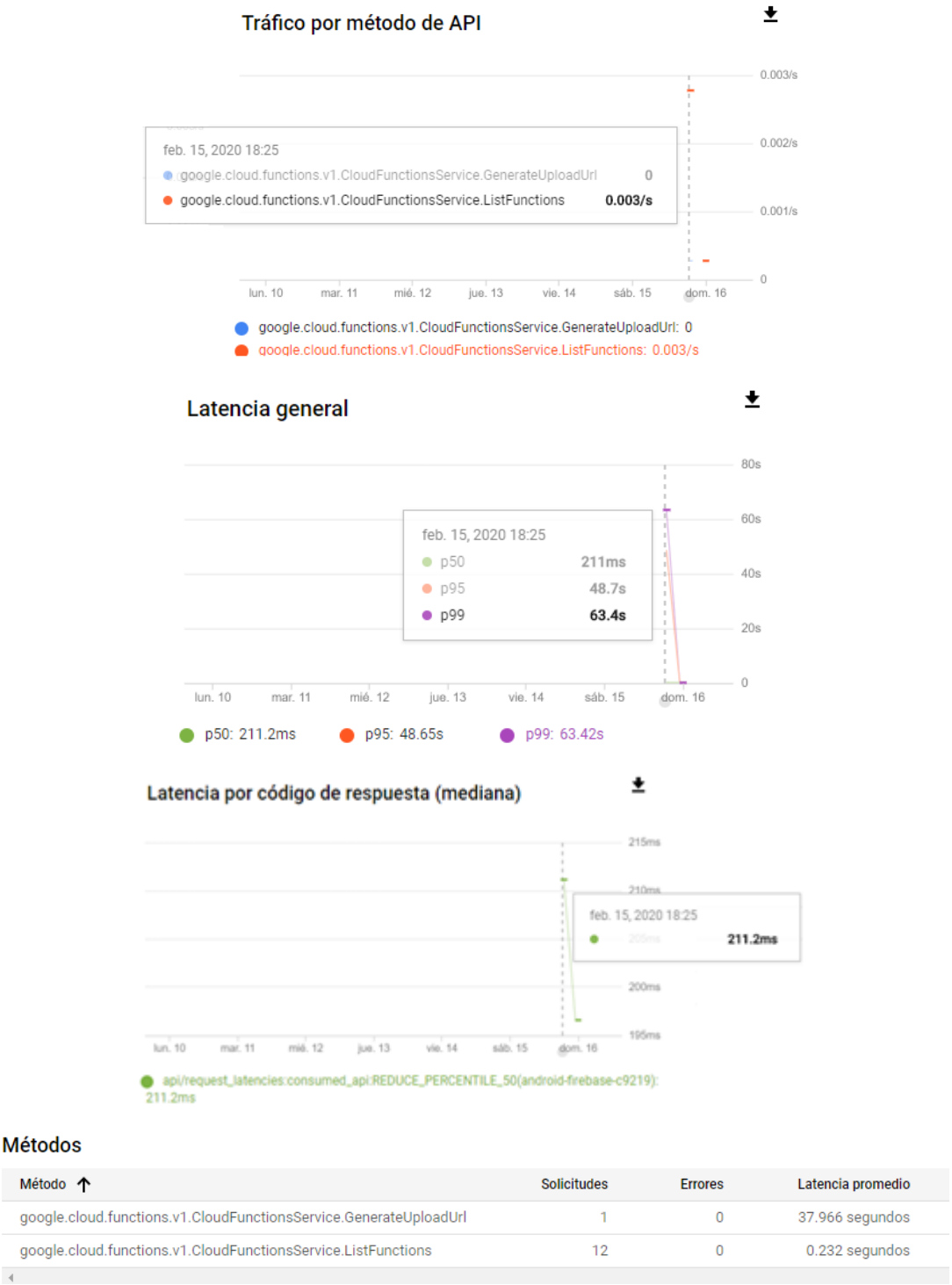


Métodos

Método	Solicitudes	Errores	Latencia promedio
google.identity.securetoken.v1.SecureToken.GrantToken	31	41.94 %	0.054 segundos
securetoken.googleapis.com.token	32	0	-

Figura 70: Token Service API, porcentaje de errores de ingreso y tiempos de latencia. Fuente: Google Cloud Platform

En la figura 71 el Cloud Functions API, gestiona funciones ligeras proporcionadas por el usuario ejecutadas en respuesta a eventos.

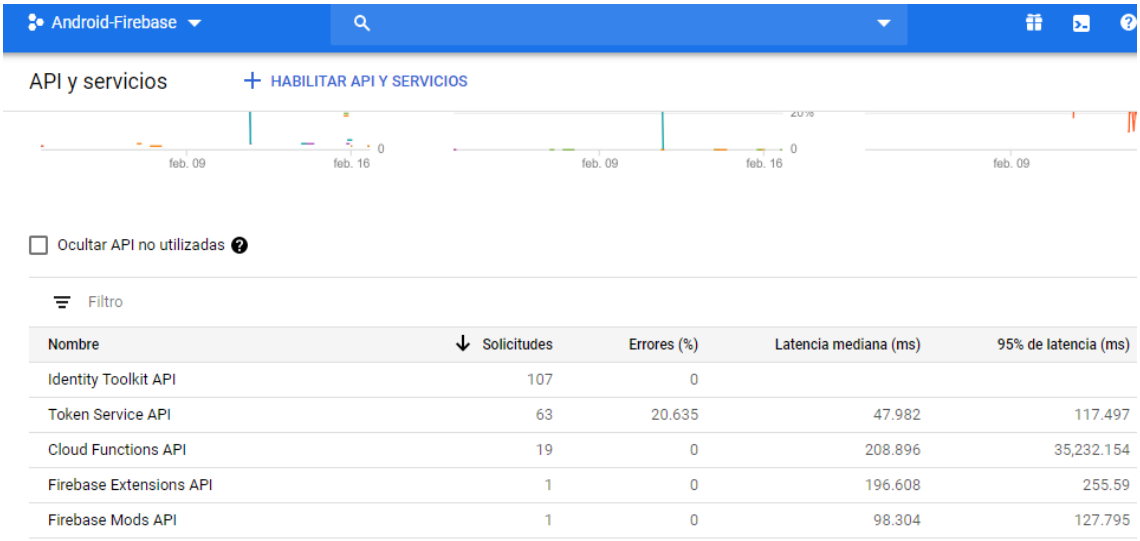


Métodos

Método	Solicitudes	Errores	Latencia promedio
google.cloud.functions.v1.CloudFunctionsService.GenerateUploadUrl	1	0	37.966 segundos
google.cloud.functions.v1.CloudFunctionsService.ListFunctions	12	0	0.232 segundos

Figura 71: Cloud Functions API, tráfico, errores y tiempos de latencia. Fuente: Google Cloud Platform

Google Cloud Platform también envía un reporte general de resultados de todas las API usadas para la teleoperación, como se puede verificar en la figura 72.



Nombre	Solicitudes	Errores (%)	Latencia mediana (ms)	95% de latencia (ms)
Identity Toolkit API	107	0		
Token Service API	63	20.635	47.982	117.497
Cloud Functions API	19	0	208.896	35,232.154
Firebase Extensions API	1	0	196.608	255.59
Firebase Mods API	1	0	98.304	127.795

Figura 72: Reporte de resultados de API y servicios. Fuente: Google Cloud Platform

Como se puede observar en la figura 72 Google Cloud Platform entregó tiempos de latencia aproximados a los que se midió en la interfaz serial de Arduino, con respecto al tráfico de datos del Cloud Functions API con Firebase se obtuvo tiempos de latencia entre cada atributo de 208,896 milisegundo y el tiempo de latencia por evento de cada atributo es de 35232,154 milisegundos, también se presentó un 20,635% de errores de ingreso con el Authentication de Firebase.

CONCLUSIONES

- La arquitectura de Cloud Computing aportó una reducción de recursos y costos ya que toda la infraestructura de la teleoperación se la implementó usando los APIs y Services de Google Cloud Platform y solo se requiere tener una cuenta de Gmail, dicha plataforma actúa como broker lo que permitió una interacción directa entre las herramientas de Firebase para el control de los mandos teleoperados por medio de dispositivos móviles y permite conectarse de forma remota a tarjetas de control que se conectan a la red de Internet en este caso se usó una ESP8266 y permitió controlar las acciones de teleoperación del brazo robótico a través del protocolos MQTT que es muy usado en aplicaciones IoT y para este sistema integrado en tiempo real y se alcanzó obtener un menor uso de ancho de banda, baja latencia y mayor rendimiento.
- La aplicación móvil que se generó, tiene la capacidad de conectarse a la red de Internet lo cual permite enlazar los controles y mandos del brazo robótico desde casi cualquier punto del planeta en tiempo real, la simplicidad de la aplicación permite una gran usabilidad, y cuenta con un potente sistema de seguridad ya que permite autenticar los usuarios.
- Para la gestión del proceso de integración de la teleoperación como gestor se usó Firebase ya que contiene herramientas como: Cloud Functions API, Realtime Database, Authentication y permitieron tener el control de todas las variables del robot de forma remota en tiempo real, carga y descarga de información de la base de datos, seguridad de la información, obtener estadísticas sobre el uso de la aplicación, además que se tiene compatibilidad de hardware.
- Se valoró el rendimiento del sistema robótico teleoperado a través de la información estadística que ofrece Firebase Analytics, Google Analytics, y Google Cloud Platform, donde se midió el tráfico tele-transmitido, tiempos de latencia, errores producidos, número de eventos, y se verificó el cumpliendo de los criterios de calidad y servicio, escalabilidad, confiabilidad y versatilidad del sistema, ya que toda esta infraestructura esta Dockerizada en Google.

RECOMENDACIONES

- Desarrollar una red dedicada de telecomunicaciones ya que va depender de una WLAN y una red de movilidad 3G/4G, y se necesita que los tiempos de latencia sean estables, y tanto el operador como el sistema remoto no pierdan comunicación.
- Implementar este sistema de teleoperación usando la plataforma Firebase a un nivel de producción para poder evaluar su desempeño.
- Estudiar el módulo ML Kit de Firebase en una red de robots ya que usa el aprendizaje automático para resolver problemas frecuentes en aplicaciones en la nube, de tal manera que tengan escalabilidad.
- Usar solo las variables necesarias dentro de la teleoperación ya que la información irrelevante puede incrementar tiempos de latencia y los errores lo cual sería una desventaja en este tipo de sistemas que funcionan en tiempo real.
- Probar este tipo de investigaciones sobre la tecnología 5G, de tal manera que se pueda mejorar la calidad de la comunicación y se pueda implementar en aplicaciones críticas como puede ser una cirugía médica.

REFERENCIAS BIBLIOGRÁFICAS

Agusti, R., Bernardo, F., & Casadevall, F. (2010). *LTE: NUEVAS TENDENCIAS EN COMUNICACIONES MÓVILES.* España: Fundación Vodafone España.

Alsalemi, A., & Alhomsi, Y. (2017). Real-Time Communication Network using Firebase Cloud IoT Platform for ECMO. *IEEE,* 178-182.

Área tecnología. (2020). *Área tecnología*. Obtenido de https://www.areatecnologia.com/Que-es-un-smartphone.htm?cv=1

Barrientos, A., Peñín, L. F., & Balaguer, C. (2007). *Fundamentos de robótica (2a. ed.).* España: McGraw-Hill España.

Basco, A., Beliz, G., & Garnero, P. (2018). *Industria 4.0 Fabricando el Futuro.* Buenos Aires: Banco Interaméricano de Desarrollo.

Brico Geek. (2020). *Brico Geek*. Obtenido de https://tienda.bricogeek.com/wifi/1033-nodemcu-v3-esp8266.html?cv=1

Cerón, A. (2009). Sistemas robóticos teleoperados. *Universidad Militar Nueva Granada*, 62-72.

Chang, R. (23 de 02 de 2019). *Smart Solutions Ec*. Obtenido de https://smartsolutionsec.net/2019/02/23/protocolos-de-comunicacion-para-iot/?cv=1

Chapaval, N. (2018). *platzi*. Obtenido de https://platzi.com/blog/que-es-frontend-y-backend/

Cortés, Y. (2017). El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras. *Redalyc*, ISSN: 1405-5597.

Drake, J. (2008). *Universidad de Cantabria.* Obtenido de https://www.ctr.unican.es/asignaturas/MC_OO/Doc/OO_08_I2_Proceso.pdf

Firebase. (17 de 11 de 2019). *Firebase*. Obtenido de https://firebase.google.com/docs/auth/

García, J. (31 de 12 de 2018). *xatakandroid*. Obtenido de https://www.xatakandroid.com/moviles-android/estos-todos-moviles-que-tienen-android-9-pie-al-acabar-2018-aosmark

Garrell, G., & Agüella, L. (2019). *La industria 4.0 en la sociedad digital.* Barcelona: Marge Books.

Gasca, M., Camargo, L., & Medina, B. (2014). Metodología para el desarrollo de aplicaciones móviles. *Redalyc*, 20-35.

Gavilán, I. (2019). *La carrera digital.* Málaga: ExLibric.

Google Cloud. (2020). *Google Cloud*. Obtenido de https://cloud.google.com/solutions/mobile/mobile-app-backend-services?cv=1&hl=es-419

Guzmán, J., Torres, I., & Galeano, P. (2014). Propuesta de un modelo de gestión de servicios en la nube para manipulación de sistemas robóticos con el uso de dispositivos móviles. *Lámpsakos*, 43-51.

Huamán, H. (2005). *Manual de técnicas de investigación, conceptos y aplicaciones.* Lima - Perú: IPLADEES S.A.C.

Llamas, L. (21 de 02 de 2019). *luisllamas*. Obtenido de https://www.luisllamas.es/protocolos-de-comunicacion-para-iot/

Made-in-China. (2020). *Made-in-China*. Obtenido de https://es.made-in-china.com/co_newerton/product_P2p-Three-Antenna-WiFi-Security-2-0MP-Surveillance-Camera-1080P-Wireless-IP-Camera_ririgugig.html?cv=1

Manzi, A., Fiorini, L., & Limosani, R. (2016). Use Case Evaluation of a Cloud Robotics Teleoperation System. *IEEE*, 208-211.

Martín, G. (2018). *Programar facil*. Obtenido de https://programarfacil.com/esp8266/como-programar-nodemcu-ide-arduino/?cv=1

Ramírez, R. (2013). *Métodos para el desarrollo de aplicaciones móviles.* Catalunya: Universidad Oberta de Catalunya.

Rodriguez, E. (2005). *Metodología de la Investigación.* Mexico: Universidad Juárez Autónoma de Tabasco.

Ruiz, M. (09 de 08 de 2017). *Openwebinars*. Obtenido de https://openwebinars.net/blog/que-es-firebase-de-google/?cv=1

Santacruz, L. (2014). *Programación multimedia y dispositivos móviles.* Madrid: RA-MA Editorial.

Toquica, J., Benavides, D., & Motta, J. (2019). WEB COMPLIANT OPEN ARCHITECTURE FOR. *IEEE*, 1408-1414.

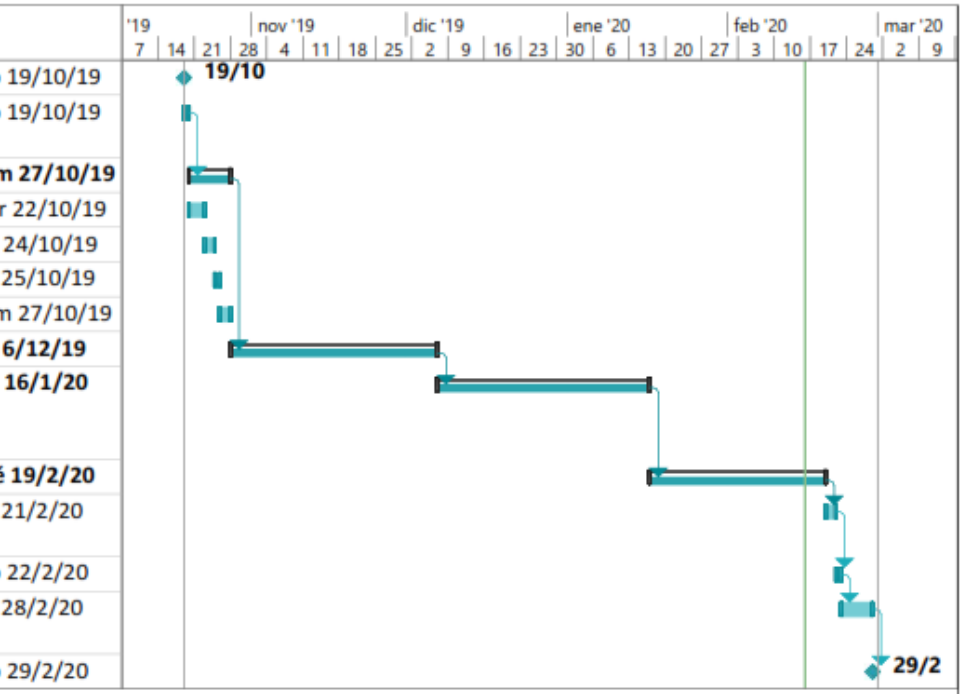
Vargas, Z. (2009). LA INVESTIGACIÓN APLICADA: UNA FORMA DE CONOCER. *Redalyc*, 155-165.

Velásquez, N. (09 de 04 de 2018). *itahora*. Obtenido de https://www.itahora.com/actualidad/la-industria-4-0-crea-valor-en-los-procesos-de-fabricacion/?cv=1

ANEXOS

Anexo A. Cronograma de actividades y recursos

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		Inicio	0 días	sáb 19/10/19	sáb 19/10/19
2		Inscripción del trabajo de titulación	1 día	sáb 19/10/19	sáb 19/10/19
3		**Introducción**	**7 días**	**dom 20/10/19**	**dom 27/10/19**
4		Antecedentes	3 días	dom 20/10/19	mar 22/10/19
5		Justificación	2 días	mié 23/10/19	jue 24/10/19
6		Alcance	1 día	vie 25/10/19	vie 25/10/19
7		Objetivos	2 días	sáb 26/10/19	dom 27/10/19
8		**Marco Teorico**	**30 días**	**lun 28/10/19**	**vie 6/12/19**
13		**Descripcion del Proceso Investigativo**	**30 días**	**sáb 7/12/19**	**jue 16/1/20**
19		**Propuesta**	**24 días**	**vie 17/1/20**	**mié 19/2/20**
30		Conclusiones y Recomendaciones	2 días	jue 20/2/20	vie 21/2/20
31		Entrega de escritos	1 día	sáb 22/2/20	sáb 22/2/20
32		Preparación defensa final	6 días	dom 23/2/20	vie 28/2/20
33		FIN	0 días	sáb 29/2/20	sáb 29/2/20



Proyecto: Proyecto Juan
Fecha: lun 17/2/20

Tarea	Resumen inactivo	Tareas externas
División	Tarea manual	Hito externo
Hito	solo duración	Fecha límite
Resumen	Informe de resumen manual	Progreso
Resumen del proyecto	Resumen manual	Progreso manual
Tarea inactiva	solo el comienzo	
Hito inactivo	solo fin	

Anexo B. Datasheet del NodeMCU V3 – ESP8266

NodeMcu v2 - ESP8266

Modelo ESP8266-NMCU

NodeMCU es una plataforma de desarrollo para Internet de las cosas (IoT) muy similar a Arduino. Te permite crear rápidamente proyectos que se pueden conectar a Internet por Wifi.

DESCRIPCIÓN

INFO

NodeMCU es una tarjeta de desarrollo similar a Arduino, especialmente orientada al Internet de las cosas (IoT). Está basado en el SoC (System on Chip) **ESP8266,** un chip altamente integrado, diseñado para las necesidades de un mundo conectado. Integra un potente procesador con Arquitectura de 32 bits (más potente que el Arduino Due) y conectividad Wifi.

Para el desarrollo de aplicaciones se puede elegir entre los lenguajes Arduino y Lua. Al trabajar dentro del entorno Arduino podremos utilizar un lenguaje que ya conocemos y hacer uso de un IDE sencillo de utilizar, ademas de hacer uso de toda la información sobre proyectos y librerías disponibles en internet. La comunidad de usuarios de Arduino es muy activa y da soporte a plataformas como el ESP8266.

NodeMCU viene con un firmware pre-instalado el cual nos permite trabajar con el lenguaje interpretado LUA, enviandole comandos mediante el puerto serial (CP2102). Las tarjetas NodeMCU y Wemos D1 mini son las plataformas mas usadas en proyectos de Internet de las cosas (IoT). No compite con Arduino, pues cubren objetivos distintos, incluso es posible programar NodeMCU desde el IDE de Arduino.

La tarjeta NodeMCU está diseñada especialmente para trabajar en protoboard. Posee un regulador de voltaje en placa que le permite alimentarse directamente del puerto USB. Los pines de entradas/salidas trabajan a 3.3V. El chip CP2102 se encarga de la comunicación USB-Serial.

ESPECIFICACIONES TÉCNICAS

- Voltaje de Alimentación (USB): 5V DC
- Voltaje de Entradas/Salidas: 3.3V DC
- SoC: ESP8266 (Módulo ESP-12)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Memoria Flash Externa: 4MB
- Pines Digitales GPIO: 17 (pueden configurarse como PWM a 3.3V)
- Pin Analógico ADC: 1 (0-1V)
- UART: 2
- Chip USB-Serial: CP2102
- Certificación FCC
- Antena en PCB
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- PLLs, reguladores, DCXO y manejo de poder integrados
- Potencia de salida de +19.5dBm en modo 802.11b
- Corriente de fuga menor a 10uA
- STBC, 1×1 MIMO, 2×1 MIMO

Anexo C. Manual configuración cámara IP Registro en la Aplicación Yoosee

1) Con su celular conectado al Wifi donde esta conectada la cámara entramos a la app, la cual nos pide registrarnos, dando clic en Registro rápido

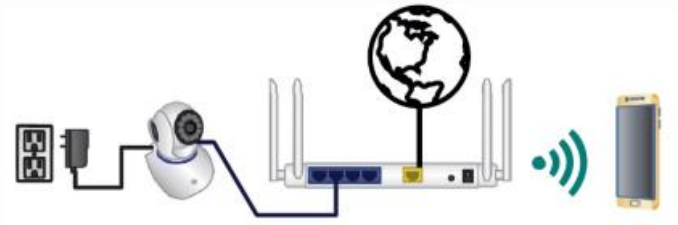
2) Se selecciona el método que se desea, ya sea mediante un número telefónico o un correo y se selecciona confirmar

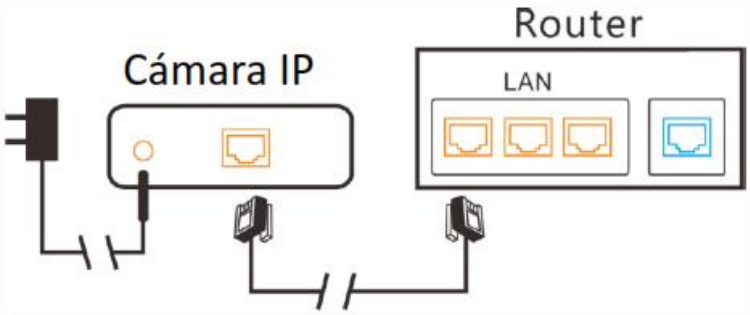
3) Una vez confirmado el registro, se procede a iniciar sesión para hacer uso de la aplicación

Anexo D. Manual configuración cámara IP conexión cableada

Instalación:

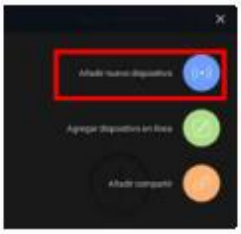
1) Se conecta el cable de RED a la cámara y al router
2) Luego se conecta el cable de corriente





3) Para añadir la cámara a la aplicación, se presiona el botón **+**

4) Se presiona **Añadir nuevo dispositivo**



5) Se selecciona el modo de conexión y presiona siguiente para continuar con el proceso

6) Se realiza la búsqueda del dispositivo y se selecciona el dispositivo encontrado

7) Se verifica el ID del dispositivo y se introduce la contraseña que posee escrita en la etiqueta de la cámara

8) Se le asigna un nombre para identificar la cámara y se confirma que la cámara será añadida

9) Se selecciona la cámara para iniciar la visualización desde la aplicación

Anexo E. Manual configuración cámara IP conexión remota

1) Para añadir la cámara a la aplicación, se presiona el botón **+**

2) Se presiona **Añadir nuevo dispositivo**

3) Se introduce el ID (que se encuentra en una etiqueta debajo de la cámara)

4) Se asigna un nombre al dispositivo a añadir

5) Se introduce la contraseña del dispositivo (se encuentra en la etiqueta debajo de la cámara)

6) Se presiona **Guardar** para finalizar la operación

Anexo F. Manual configuración cámara IP conexión inalámbrica

1) Para añadir la cámara a la aplicación, se presiona el botón +

2) Se presiona **Añadir nuevo dispositivo**

3) Se selecciona el modo de conexión y presiona siguiente para continuar con el proceso

6) Se introduce la contraseña del WIFI donde está conectado el teléfono y al cual se desea conectar la cámara

5) Se verifica que el equipo se encuentre encendido y se presiona el botón **He oído el timbre de aviso para conectar**

6) Se coloca el teléfono cercano a la cámara para que esta pueda ser escuchada

5) Se espera a que el dispositivo escuche el timbre emitido por el teléfono y se espera que se añada

7) Se verifica el ID del dispositivo y se introduce la contraseña que posee escrita en la etiqueta de la cámara

8) Se le asigna un nombre para identificar la cámara y se confirma que la cámara será añadida

9) Se selecciona la cámara para iniciar la visualización desde la aplicación

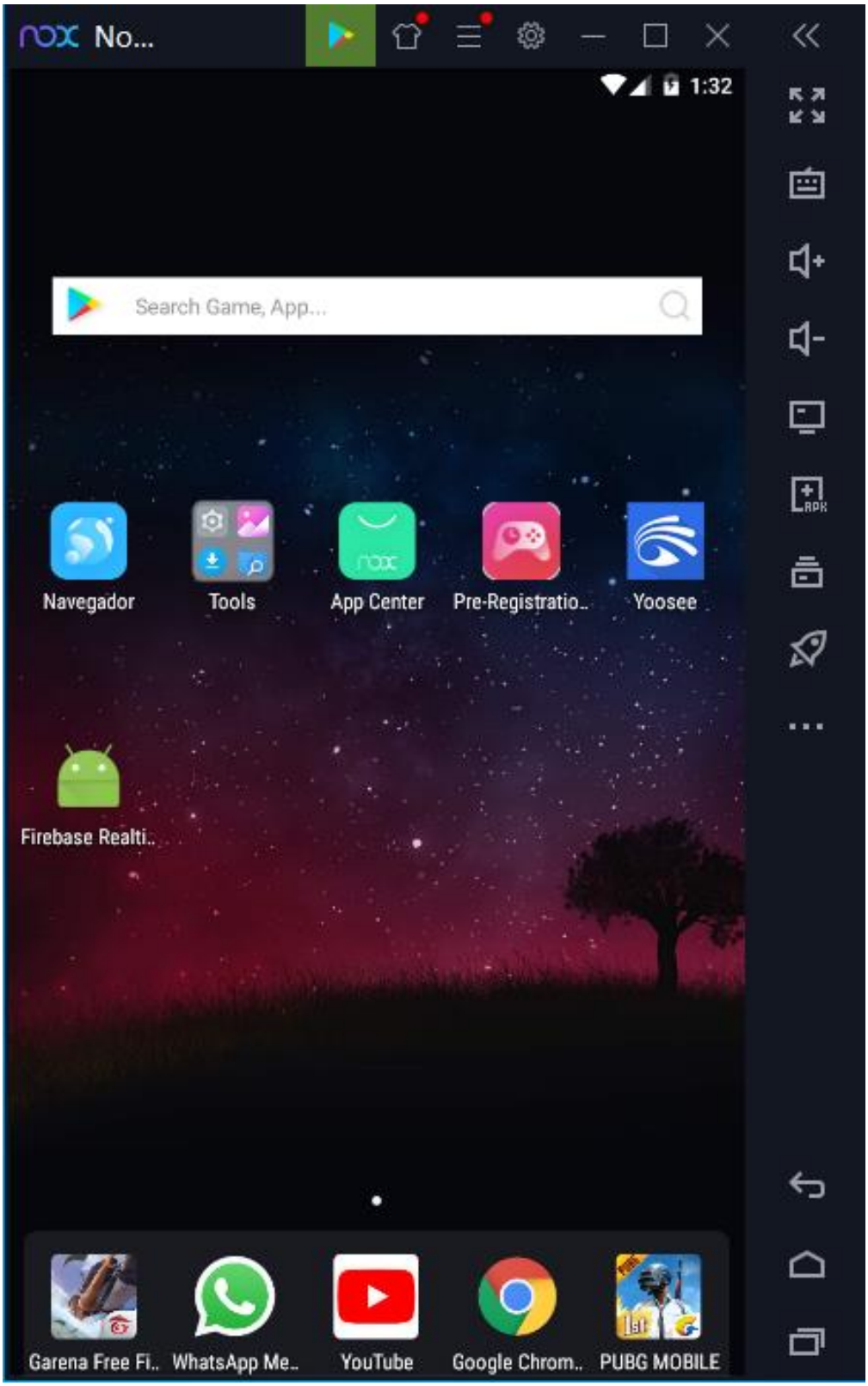
Movimientos

Para que la cámara gire, se deberá deslizar el dedo sobre la pantalla donde se está visualizando su cámara.

Deslizando para arriba o para abajo se conseguirá el movimiento vertical de hasta 120°.

Deslizando para la derecha o izquierda se conseguirá el movimiento horizontal de hasta 355°.

Anexo G. Emulador de Android en Windows 10 usando NOX 6



Anexo H. build.gradle(Proyect), dependecias de los API y Service de Firebase y Google en el

```
buildscript {
    repositories {
        jcenter()
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
        google()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.4.1'
        classpath 'com.google.gms:google-services:3.0.0'

        // NOTE: Do not place your application dependencies here; they
belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Anexo I. build.gradle(Module), dependecias de los API y Service de Firebase y Google en el

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "net.sgoliver.android.firebasertdb"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:design:26.1.0'
    implementation 'com.android.support:support-annotations:26.1.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    implementation 'android.arch.lifecycle:extensions:1.1.1'

androidTestImplementation('com.android.support.test.espresso:espresso-
core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-
annotations'
    })
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.google.firebase:firebase-database:9.4.0'
    implementation 'com.google.firebase:firebase-auth:9.4.0'
    implementation 'com.facebook.android:facebook-login:[5,6)'
    implementation 'com.google.android.gms:play-services-auth:9.4.0'
    testImplementation 'junit:junit:4.12'
}
apply plugin: 'com.google.gms.google-services'
```

Anexo J. Login desing en XMLA de Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Login">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="20dp">

        <EditText
            android:id="@+id/etUser"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Ingrese su correo"
            android:inputType="textEmailAddress" />

        <EditText
            android:id="@+id/etPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Ingrese su contraseña"
            android:inputType="textPassword" />

        <Button
            android:id="@+id/btnLogin"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:text="Ingresar" />


        <Button
            android:id="@+id/btnRegistrar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Registrar" />

        <com.facebook.login.widget.LoginButton
            android:id="@+id/login_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="30dp"
            android:layout_marginBottom="30dp" />
```

```
        <com.google.android.gms.common.SignInButton
            android:id="@+id/sign_in_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:layout_gravity="center_horizontal"
            android:orientation="horizontal">

            <Button
                android:id="@+id/sign_out_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:text="@string/sign_out" />

            <Button
                android:id="@+id/revoke_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_horizontal"
                android:text="@string/revocar" />

        </LinearLayout>

        <TextView
            android:id="@+id/txtNombre"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/txtNombre" />

        <TextView
            android:id="@+id/txtEmail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/txtNombre" />

    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Anexo K. Login Class en Java de Android Studio

```
package net.sgoliver.android.firebasertdb;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.facebook.AccessToken;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
import com.facebook.FacebookException;
import com.facebook.login.LoginManager;
import com.facebook.login.LoginResult;
import com.facebook.login.widget.LoginButton;
import com.google.android.gms.auth.api.Auth;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.auth.api.signin.GoogleSignInResult;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.OptionalPendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;


public class Login extends AppCompatActivity implements
GoogleApiClient.OnConnectionFailedListener {

    private SignInButton btnSignIn;
    private Button btnSignOut;
    private Button btnRevoke;
    private TextView txtNombre;
    private TextView txtEmail;

    private EditText etUser;
    private EditText etPassword;
    private Button btnLogin;
    private Button btnRegistrar;
    private LoginButton loginButton;
    CallbackManager callbackManager;
```

```
    private GoogleApiClient apiClient;
    private static final int RC_SIGN_IN = 1001;

    private ProgressDialog progressDialog;
    private static final String TAG = "MainActivity";

    //Llamamos al autenticador
    FirebaseAuth auth;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        callbackManager = CallbackManager.Factory.create();

        AccessToken accessToken = AccessToken.getCurrentAccessToken();
        boolean isLoggedIn = accessToken != null &&
!accessToken.isExpired();

        //Ahora Instanciamos
        auth=FirebaseAuth.getInstance();

        etUser = (EditText) findViewById(R.id.etUser);
        etPassword = (EditText) findViewById(R.id.etPassword);
        btnLogin = (Button)findViewById(R.id.btnLogin);
        btnRegistrar = (Button)findViewById(R.id.btnRegistrar);

        loginButton = (LoginButton) findViewById(R.id.login_button);
        loginButton.setReadPermissions("email", "public_profile");
        // If using in a fragment
        //loginButton.setFragment(this);

        // Callback registration
        loginButton.registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
            @Override
            public void onSuccess(LoginResult loginResult) {
                // App code

            }

            @Override
            public void onCancel() {
                // App code

            }

            @Override
            public void onError(FacebookException exception) {
                // App code
            }
        });
```

```
        LoginManager.getInstance().registerCallback(callbackManager,
                new FacebookCallback<LoginResult>() {
                    @Override
                    public void onSuccess(LoginResult loginResult) {
                        // App code
                        Intent intent = new
Intent(Login.this.getBaseContext(),
                                MainActivity.class);
                        startActivity(intent);
                    }

                    @Override
                    public void onCancel() {
                        // App code
                    }

                    @Override
                    public void onError(FacebookException exception) {
                        // App code
                    }
                });

        btnSignIn = (SignInButton)findViewById(R.id.sign_in_button);
        btnSignOut = (Button)findViewById(R.id.sign_out_button);
        btnRevoke = (Button)findViewById(R.id.revoke_button);
        txtNombre = (TextView)findViewById(R.id.txtNombre);
        txtEmail = (TextView)findViewById(R.id.txtEmail);

        //Google API Client

        GoogleSignInOptions gso =
                new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                        .requestEmail()
                        .build();

        apiClient = new GoogleApiClient.Builder(this)
                .enableAutoManage(this, this)
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .build();

        //Personalización del botón de login

        btnSignIn.setSize(SignInButton.SIZE_STANDARD);
        btnSignIn.setColorScheme(SignInButton.COLOR_LIGHT);
        btnSignIn.setScopes(gso.getScopeArray());

        //Eventos de los botones

        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent signInIntent =
Auth.GoogleSignInApi.getSignInIntent(apiClient);
                startActivityForResult(signInIntent, RC_SIGN_IN);
            }
```

```
        });

        btnSignOut.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

Auth.GoogleSignInApi.signOut(apiClient).setResultCallback(
                        new ResultCallback<Status>() {
                            @Override
                            public void onResult(Status status) {
                                updateUI(false);
                            }
                        });
            }
        });

        btnRevoke.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

Auth.GoogleSignInApi.revokeAccess(apiClient).setResultCallback(
                        new ResultCallback<Status>() {
                            @Override
                            public void onResult(Status status) {
                                updateUI(false);
                            }
                        });
            }
        });

        updateUI(false);

        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String userE=etUser.getText().toString();
                String passE=etPassword.getText().toString();
                //Ahora validamos por si uno de los campos esta vacío
                if(TextUtils.isEmpty(userE)){
                    //por si falta correo
                    Toast.makeText(Login.this,"Inserte
correo",Toast.LENGTH_SHORT).show();
                    return;
                }
                if(TextUtils.isEmpty(passE)){
                    //por si falta password
                    Toast.makeText(Login.this,"Inserte
contraseña",Toast.LENGTH_SHORT).show();
                    return;
                }

                //Ahora usamos el Auth para que se logee una vez
registrado
                auth.signInWithEmailAndPassword(userE,passE).
                        //Le pasamos la clase registro
                                addOnCompleteListener(Login.this, new
OnCompleteListener<AuthResult>() {
```

```
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {

                                if(!task.isSuccessful()){
                                    Toast.makeText(Login.this,"A ocurrido
un error",Toast.LENGTH_SHORT).show();
                                    return;
                                }


Toast.makeText(Login.this,"Bienvenido",Toast.LENGTH_SHORT).show();

                                Intent i = new
Intent(Login.this,MainActivity.class);
                                startActivity(i);
                            }
                        });
            }
        });

        btnRegistrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(Login.this,Registro.class);
                startActivity(i);
            }
        });
    }

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {
        Toast.makeText(this, "Error de conexion!",
Toast.LENGTH_SHORT).show();
        Log.e("GoogleSignIn", "OnConnectionFailed: " + connectionResult);
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        callbackManager.onActivityResult(requestCode, resultCode, data);
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == RC_SIGN_IN) {
            GoogleSignInResult result =
                    Auth.GoogleSignInApi.getSignInResultFromIntent(data);

            handleSignInResult(result);
        }

        //Toast.makeText(this, "No existe el producto con dicho código",
Toast.LENGTH_SHORT).show();
        Intent i = new Intent(this, MainActivity.class );
        startActivity(i);

    }
```

```
    private void handleSignInResult(GoogleSignInResult result) {
        if (result.isSuccess()) {
            //Usuario logueado --> Mostramos sus datos
            GoogleSignInAccount acct = result.getSignInAccount();
            txtNombre.setText(acct.getDisplayName());
            txtEmail.setText(acct.getEmail());
            updateUI(true);
        } else {
            //Usuario no logueado --> Lo mostramos como "Desconectado"
            updateUI(false);
        }
    }

    private void updateUI(boolean signedIn) {
        if (signedIn) {
            btnSignIn.setVisibility(View.GONE);
            btnSignOut.setVisibility(View.VISIBLE);
            btnRevoke.setVisibility(View.VISIBLE);
        } else {
            txtNombre.setText("Desconectado");
            txtEmail.setText("Desconectado");

            btnSignIn.setVisibility(View.VISIBLE);
            btnSignOut.setVisibility(View.GONE);
            btnRevoke.setVisibility(View.GONE);
        }
    }

    @Override
    protected void onStart() {
        super.onStart();

        OptionalPendingResult<GoogleSignInResult> opr =
Auth.GoogleSignInApi.silentSignIn(apiClient);
        if (opr.isDone()) {
            GoogleSignInResult result = opr.get();
            handleSignInResult(result);
        } else {
            showProgressDialog();
            opr.setResultCallback(new
ResultCallback<GoogleSignInResult>() {
                @Override
                public void onResult(GoogleSignInResult
googleSignInResult) {
                    hideProgressDialog();
                    handleSignInResult(googleSignInResult);
                }
            });
        }
    }

    private void showProgressDialog() {
        if (progressDialog == null) {
            progressDialog = new ProgressDialog(this);
            progressDialog.setMessage("Silent SignI-In");
            progressDialog.setIndeterminate(true);
```

```
            }

            progressDialog.show();
        }

        private void hideProgressDialog() {
            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.hide();
            }
        }
}
```

Anexo L. Registro usuario desing en XMLA de Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Registro">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="20dp">

        <EditText
            android:id="@+id/etUser"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Ingrese su correo"
            android:inputType="textEmailAddress" />

        <EditText
            android:id="@+id/etPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ems="10"
            android:hint="Ingrese su contraseña"
            android:inputType="textPassword" />


        <Button
            android:id="@+id/btnRegistrar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Registrar" />




    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

Anexo M. Registro Class en Java de Android Studio

```
package net.sgoliver.android.firebasertdb;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class Registro extends AppCompatActivity {

    private EditText etUser;
    private EditText etPassword;
    private Button btnRegistrar;
    FirebaseAuth auth;

    private ProgressDialog progressDialog;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);

        progressDialog=new ProgressDialog(this);

        //Instanciamos igualmente
        auth=FirebaseAuth.getInstance();

        etUser=(EditText) findViewById(R.id.etUser);
        etPassword=(EditText) findViewById(R.id.etPassword);
        btnRegistrar=(Button) findViewById(R.id.btnRegistrar);

        btnRegistrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String userE=etUser.getText().toString();
                String passE=etPassword.getText().toString();
                //Ahora validamos por si uno de los campos esta vacío
                if(TextUtils.isEmpty(userE)){
                    //por si falta correo
                    Toast.makeText(Registro.this,"Inserte
correo",Toast.LENGTH_SHORT).show();
                    return;
```

```
                }
                if(TextUtils.isEmpty(passE)){
                    //por si falta password
                    Toast.makeText(Registro.this,"Inserte
contraseña",Toast.LENGTH_SHORT).show();
                    return;
                }

                progressDialog.setMessage("En proceso");
                progressDialog.show();

                //Ahora usamos el metodo
                auth.createUserWithEmailAndPassword(userE,passE).
                        //Le pasamos la clase registro
                        addOnCompleteListener(Registro.this, new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {
                                //Task=Tareas devuelve si la tarea si se
cumple
                                //En este caso si se cumplio
                                Toast.makeText(Registro.this, "Usuario
registrado exitosamente", Toast.LENGTH_SHORT).show();
                                //Si no logra registrarse
                                if(!task.isSuccessful()){
                                    Toast.makeText(Registro.this,"Usuario
no se ha podido registrar",Toast.LENGTH_SHORT).show();
                                    return;
                                }
                                Intent i = new
Intent(Registro.this,Login.class);
                                startActivity(i);
                            }
                        });
            }
        });


    }
}
```

Anexo N. MainActivity mandos de control para el robot teleoperado desing en XMLA de Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="net.sgoliver.android.firebasertdb.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/lblRobot0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Nombre del robot: " />

            <TextView
                android:id="@+id/lblRobot"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/lblBase0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="Articulación base: " />

            <TextView
                android:id="@+id/lblBase"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp" />
```

```
        </LinearLayout>

        <Button
            android:id="@+id/btnEliminarListener"
            android:layout_width="wrap_content"
            android:layout_height="8dp"
            android:text="Eliminar Listener" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/lblHombro0"
                android:layout_width="21dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Articulaciòn hombro:" />

            <TextView
                android:id="@+id/lblHombro"
                android:layout_width="121dp"
                android:layout_height="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/lblCodo0"
                android:layout_width="158dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Articulaciòn codo:" />

            <TextView
                android:id="@+id/lblCodo"
                android:layout_width="288dp"
                android:layout_height="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/lblPitch0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="Articulaciòn pitch:" />
```

```
        <TextView
            android:id="@+id/lblPitch"
            android:layout_width="252dp"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/lblYaw0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Articulaciòn yaw:" />

        <TextView
            android:id="@+id/lblYaw"
            android:layout_width="258dp"
            android:layout_height="wrap_content"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/lblGripper0"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp"
            android:text="Gripper: " />

        <TextView
            android:id="@+id/lblGripper"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="5dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView2"
            android:layout_width="74dp"
            android:layout_height="51dp"
            android:layout_weight="1"
            android:text="Base" />
```

```
        <SeekBar
            android:id="@+id/seekBar1"
            android:layout_width="285dp"
            android:layout_height="30dp" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView3"
            android:layout_width="14dp"
            android:layout_height="51dp"
            android:layout_weight="1"
            android:text="Hombro" />

        <SeekBar
            android:id="@+id/seekBar2"
            android:layout_width="167dp"
            android:layout_height="30dp"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Codo" />

        <SeekBar
            android:id="@+id/seekBar3"
            android:layout_width="186dp"
            android:layout_height="30dp"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Muñeca Pitch" />
```

```
            <SeekBar
                android:id="@+id/seekBar4"
                android:layout_width="239dp"
                android:layout_height="30dp"
                android:layout_weight="1" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <TextView
                android:id="@+id/textView6"
                android:layout_width="wrap_content"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:text="Muñeca Yaw" />

            <SeekBar
                android:id="@+id/seekBar5"
                android:layout_width="233dp"
                android:layout_height="30dp"
                android:layout_weight="1" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="58dp"
            android:orientation="horizontal">

            <Button
                android:id="@+id/btnAbrirGripper"
                android:layout_width="168dp"
                android:layout_height="wrap_content"
                android:text="Abrir Gripper" />

            <Button
                android:id="@+id/btnCerrarGripper"
                android:layout_width="165dp"
                android:layout_height="wrap_content"
                android:text="Cerrar Gripper" />
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:orientation="horizontal">

            <Switch
                android:id="@+id/switch1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="ENCENDER ROBOT" />

            <Button
                android:id="@+id/btnSalir"
```

```
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Salir" />

        </LinearLayout>

    </LinearLayout>

</RelativeLayout>
```

Anexo O. MainActivity Class mandos de control para el robot teleoperado en Java de Android Studio

```
package net.sgoliver.android.firebasertdb;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.SeekBar;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.google.firebase.analytics.FirebaseAnalytics;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class MainActivity extends AppCompatActivity {
//public class MainActivity extends AppCompatActivity implements
SeekBar.OnSeekBarChangeListener {

    private static final String TAGLOG = "firebase-db";

    private TextView lblRobot;
    private TextView lblBase;
    private TextView lblHombro;
    private TextView lblCodo;
    private TextView lblPitch;
    private TextView lblYaw;
    private TextView lblGripper;
    private Button btnEliminarListener;
    private Button btnAbrirGripper;
    private Button btnCerrarGripper;
    private Button btnSalir;
    private SeekBar seekBar1;
    private SeekBar seekBar2;
    private SeekBar seekBar3;
    private SeekBar seekBar4;
    private SeekBar seekBar5;
    private Switch switch1;

    private DatabaseReference dbCielo;
    private DatabaseReference dbPrediccion;
    private ValueEventListener eventListener;

    //private FirebaseAnalytics mFirebaseAnalytics;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //mFirebaseAnalytics = FirebaseAnalytics.getInstance(this);


        lblRobot = (TextView)findViewById(R.id.lblRobot);
        lblBase = (TextView)findViewById(R.id.lblBase);
        lblHombro = (TextView)findViewById(R.id.lblHombro);
        lblCodo = (TextView)findViewById(R.id.lblCodo);
        lblPitch = (TextView)findViewById(R.id.lblPitch);
        lblYaw = (TextView)findViewById(R.id.lblYaw);
        lblGripper = (TextView)findViewById(R.id.lblGripper);
        btnEliminarListener =
(Button)findViewById(R.id.btnEliminarListener);
        btnAbrirGripper = (Button)findViewById(R.id.btnAbrirGripper);
        btnCerrarGripper = (Button)findViewById(R.id.btnCerrarGripper);
        btnSalir = (Button)findViewById(R.id.btnSalir);
        switch1 = (Switch) findViewById(R.id.switch1);
        switch1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            public void onCheckedChanged(CompoundButton buttonView,
boolean isChecked) {
                if (isChecked) {
                    // The toggle is enabled
                    dbPrediccion.child("estado").setValue("encendido");
                } else {
                    // The toggle is disabled
                    dbPrediccion.child("estado").setValue("apagado");
                }
            }
        });


        //--Asignamos las propiedades de nuestro control a la variable
SeekBar1 que es la Base
        seekBar1=(SeekBar)findViewById(R.id.seekBar1);
        // Valot Final
        seekBar1.setMax(180);
        seekBar1.setOnSeekBarChangeListener(
                new SeekBar.OnSeekBarChangeListener() {
                    //hace un llamado a la perilla cuando se arrastra
                    @Override
                    public void onProgressChanged(SeekBar seekBar,
                                                  int progress, boolean
fromUser) {
                        dbPrediccion.child("base").setValue(progress);

Toast.makeText(getApplicationContext(),String.valueOf(progress)+"
°",Toast.LENGTH_SHORT).show();
                    }
                    //hace un llamado  cuando se toca la perilla
                    public void onStartTrackingTouch(SeekBar seekBar) {
                    }
                    //hace un llamado  cuando se detiene la perilla
                    public void onStopTrackingTouch(SeekBar seekBar) {
```

```
                    }
                });
        //--Asignamos las propiedades de nuestro control a la variable
SeekBar2 que es el hombro
        seekBar2=(SeekBar)findViewById(R.id.seekBar2);
        // Valot Final
        seekBar2.setMax(30);
        seekBar2.setOnSeekBarChangeListener(
                new SeekBar.OnSeekBarChangeListener() {
                    //hace un llamado a la perilla cuando se arrastra
                    @Override
                    public void onProgressChanged(SeekBar seekBar,
                                                  int progress, boolean
fromUser) {

dbPrediccion.child("hombro").setValue(progress+80);

Toast.makeText(getApplicationContext(),String.valueOf(progress+80)+"
°",Toast.LENGTH_SHORT).show();
                    }
                    //hace un llamado  cuando se toca la perilla
                    public void onStartTrackingTouch(SeekBar seekBar) {
                    }
                    //hace un llamado  cuando se detiene la perilla
                    public void onStopTrackingTouch(SeekBar seekBar) {
                    }
                });
        //--Asignamos las propiedades de nuestro control a la variable
SeekBar2 que es el codo
        seekBar3=(SeekBar)findViewById(R.id.seekBar3);
        // Valot Final
        seekBar3.setMax(75);
        seekBar3.setOnSeekBarChangeListener(
                new SeekBar.OnSeekBarChangeListener() {
                    //hace un llamado a la perilla cuando se arrastra
                    @Override
                    public void onProgressChanged(SeekBar seekBar,
                                                  int progress, boolean
fromUser) {
                        dbPrediccion.child("codo").setValue(progress+45);

Toast.makeText(getApplicationContext(),String.valueOf(progress+45)+"
°",Toast.LENGTH_SHORT).show();
                    }
                    //hace un llamado  cuando se toca la perilla
                    public void onStartTrackingTouch(SeekBar seekBar) {
                    }
                    //hace un llamado  cuando se detiene la perilla
                    public void onStopTrackingTouch(SeekBar seekBar) {
                    }
                });
        //--Asignamos las propiedades de nuestro control a la variable
SeekBar2 que es el pitch
        seekBar4=(SeekBar)findViewById(R.id.seekBar4);
        // Valot Final
        seekBar4.setMax(90);
        seekBar4.setOnSeekBarChangeListener(
```

```
                new SeekBar.OnSeekBarChangeListener() {
                    //hace un llamado a la perilla cuando se arrastra
                    @Override
                    public void onProgressChanged(SeekBar seekBar,
                                                  int progress, boolean
fromUser) {

dbPrediccion.child("pitch").setValue(progress+45);

Toast.makeText(getApplicationContext(),String.valueOf(progress+45)+"
°",Toast.LENGTH_SHORT).show();
                    }
                    //hace un llamado  cuando se toca la perilla
                    public void onStartTrackingTouch(SeekBar seekBar) {
                    }
                    //hace un llamado  cuando se detiene la perilla
                    public void onStopTrackingTouch(SeekBar seekBar) {
                    }
                });

        //--Asignamos las propiedades de nuestro control a la variable
SeekBar2 que es el yaw
        seekBar5=(SeekBar)findViewById(R.id.seekBar5);
        // Valot Final
        seekBar5.setMax(180);
        seekBar5.setOnSeekBarChangeListener(
                new SeekBar.OnSeekBarChangeListener() {
                    //hace un llamado a la perilla cuando se arrastra
                    @Override
                    public void onProgressChanged(SeekBar seekBar,
                                                  int progress, boolean
fromUser) {
                        dbPrediccion.child("yaw").setValue(progress);

Toast.makeText(getApplicationContext(),String.valueOf(progress)+"
°",Toast.LENGTH_SHORT).show();
                    }
                    //hace un llamado  cuando se toca la perilla
                    public void onStartTrackingTouch(SeekBar seekBar) {
                    }
                    //hace un llamado  cuando se detiene la perilla
                    public void onStopTrackingTouch(SeekBar seekBar) {
                    }
                });

        dbPrediccion =
            FirebaseDatabase.getInstance().getReference()
                .child("Teleoperation");


        eventListener = new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                //Opción 1

lblRobot.setText(dataSnapshot.child("robot").getValue().toString());
```

```
lblBase.setText(dataSnapshot.child("base").getValue().toString()+ "°");

lblHombro.setText(dataSnapshot.child("hombro").getValue().toString()+
"°");

lblCodo.setText(dataSnapshot.child("codo").getValue().toString()+ "°");

lblPitch.setText(dataSnapshot.child("pitch").getValue().toString()+ "°");

lblYaw.setText(dataSnapshot.child("yaw").getValue().toString()+ "°");

lblGripper.setText(dataSnapshot.child("gripper").getValue().toString());
                // Valor Inicial

seekBar1.setProgress(Integer.parseInt(dataSnapshot.child("base").getValue
().toString()));

seekBar2.setProgress(Integer.parseInt(dataSnapshot.child("hombro").getVal
ue().toString())-80);

seekBar3.setProgress(Integer.parseInt(dataSnapshot.child("codo").getValue
().toString())-45);

seekBar4.setProgress(Integer.parseInt(dataSnapshot.child("pitch").getValu
e().toString())-45);

seekBar5.setProgress(Integer.parseInt(dataSnapshot.child("yaw").getValue(
).toString()));

if(dataSnapshot.child("estado").getValue().toString().equals("encendido")
){
                    switch1.setChecked(true);
                }else{
                    switch1.setChecked(false);
                }
                //Opcion 2
                Prediccion pred =
dataSnapshot.getValue(Prediccion.class);


                Log.e(TAGLOG, "onDataChange:" +
dataSnapshot.getValue().toString());
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                Log.e(TAGLOG, "Error!", databaseError.toException());
            }
        };

        dbPrediccion.addValueEventListener(eventListener);

        btnEliminarListener.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View view) {
```

```
                dbPrediccion.removeEventListener(eventListener);
            }
        });

        btnAbrirGripper.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                dbPrediccion.child("gripper").setValue("abierto");
            }
        });

        btnCerrarGripper.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                dbPrediccion.child("gripper").setValue("cerrado");
            }
        });

        btnSalir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(MainActivity.this,Login.class);
                startActivity(i);
            }
        });


    }

}
```

Anexo P. Android.Manifest en XMLA de Android Studio

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.sgoliver.android.firebasertdb">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".Registro"></activity>
        <activity android:name=".MainActivity" />
        <activity android:name=".Login">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id" />

        <activity
            android:name="com.facebook.FacebookActivity"

android:configChanges="keyboard|keyboardHidden|screenLayout|screenSize|or
ientation"
            android:label="@string/app_name" />
        <activity
            android:name="com.facebook.CustomTabActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />

                <category android:name="android.intent.category.DEFAULT"
/>
                <category
android:name="android.intent.category.BROWSABLE" />

                <data android:scheme="@string/fb_login_protocol_scheme"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Anexo Q. Control del brazo robótico con el IDE Arduino y NodeMCU V3 – ESP8266

```
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <Servo.h>

Servo m1,m2,m3,m4,m5,m6;  // create servo object to control a servo

//https://github.com/mobizt/Firebase-ESP8266/blob/master/README.md
//https://www.javacodegeeks.com/2019/07/esp8266-esp32-firebase-realtime-database-iot.html
//https://github.com/mobizt/Firebase-ESP8266

const char* ssid = "Wifi_Israel";
const char* password = "uisrael";

FirebaseData firebaseData;

int led=16;
int robot=13;


void setup() {
  Serial.begin(115200);
  pinMode(led,OUTPUT);
  pinMode(robot,OUTPUT);
  connectWifi();
  //leds.begin();
  Firebase.begin("https://android-firebase-c9219.firebaseio.com/",
"Q9886kW2ODbTVL9tAAkd2a7MMHht2Ja5j6LZvude");
  m1.attach(5);  // attaches the servo on pin D1 to the servo object
  m2.attach(4);  //PIN D2
  m3.attach(0);  //PIN D3
  m4.attach(2);  //PIN D4
  m5.attach(14); //PIN D5
  m6.attach(12); //PIN D6     16-->D0, 13-->D7, 15-->D8
  // cafe gripper D6, rojo yaw D5, naranja pitch D4, amarillo codo D3,
verde hombro D2, azul base D1
}

void loop() {

  if (Firebase.getString(firebaseData, "/Teleoperation/robot")) {
    if  (firebaseData.dataType() == "string") {
      String val = firebaseData.stringData();
      Serial.println(val);
    }
  }


  if (Firebase.getInt(firebaseData, "/Teleoperation/base")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m1.write(val);
    }
```

```
  }

  if (Firebase.getInt(firebaseData, "/Teleoperation/hombro")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m2.write(val);

    }
  }

  if (Firebase.getInt(firebaseData, "/Teleoperation/codo")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m3.write(val);

    }
  }

  if (Firebase.getInt(firebaseData, "/Teleoperation/pitch")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m4.write(val);

    }
  }

  if (Firebase.getInt(firebaseData, "/Teleoperation/yaw")) {
    if  (firebaseData.dataType() == "int") {
      int val = firebaseData.intData();
      Serial.println(val);
      m5.write(val);

    }
  }

  if (Firebase.getString(firebaseData, "/Teleoperation/gripper")) {
    if  (firebaseData.dataType() == "string") {
      String val = firebaseData.stringData();
      if (val == "abierto") {
        m6.write(45);
        digitalWrite(led,LOW);
        Serial.println(val);
         //setLedColor();
      }
      if (val == "cerrado") {
        m6.write(135);
        digitalWrite(led,HIGH);
        Serial.println(val);
      }
    }
  }

  if (Firebase.getString(firebaseData, "/Teleoperation/estado")) {
    if  (firebaseData.dataType() == "string") {
```

```
      String val = firebaseData.stringData();
      if (val == "encendido") {
        Serial.println(val);
        digitalWrite(robot,HIGH);
      }
      if (val == "apagado") {
        Serial.println(val);
        digitalWrite(robot,LOW);
      }
    }
  }
}

void connectWifi() {
  // Let us connect to WiFi
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println(".......");
  Serial.println("WiFi Connected....IP Address:");
  Serial.println(WiFi.localIP());

}
```