UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Inteligencia Artificial aplicada en el diseño de un dispositivo para registrar el número de contactos de una persona en el marco de COVID - 19

TESINA

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Guillermo Martínez Martínez

DIRECTORA DE TESINA

M.I. Rosa Itzel Flores Luna

Ciudad Universitaria, Cd. Mx., 2022

Agradecimientos

A Dios sobre todo, por la vida y salud de mi familia, seres queridos y la mía. Por la oportunidad y que bajo su voluntad pueda yo desenvolverme en estas áreas del conocimiento.

A mi familia, mi madre Angélica Martínez, por su amor maternal infinito, ser inspiración de mi visión de y hacia la vida, entusiasta natural cuyo acervo de valores, educación y apoyo incondicional son fundamentos del yo actual y por su siempre cariñosa motivación en mi vida, mostrándome su confianza. A mi padre Guillermo Martínez, por su incondicional apoyo, cariño, formación, educación y guía, de valores intrínsecos y por siempre motivarme a alcanzar mis más altos sueños, confiando en mí. A mi hermana Geraldine y hermano Moisés, quienes me muestran su cariño y confianza, de quienes sus sueños tengo certeza lograrán con su visión y capacidades únicas. Este capítulo de vida es producto del fundamental apoyo e inspiración que me han dado; inherentemente estoy formado en gran medida por las únicas contribuciones de cada uno, por lo tanto, me gustaría expresarles a ti Mamá, Papá, Geral y Moy, que es un capítulo compartido, les pertenece tanto como a mí. Mil gracias.

A mis abuelos, Victoria y Pedro, Elia y Guillermo, por con tanto cariño ejercer el rol de tronco y raíces de la familia, en analogía con el Árbol del Tule en Oaxaca, México. Por aconsejar, apapachar y continuar educando a mis padres y sus nietos. Por sus consejos y su incondicional cariño cálido y sabio, por sus historias con experiencias y perspectivas de vida que me han dado. Asimismo, mis tías y tíos que han apoyado.

Agradecer a la M.I.Itzel Flores, por su valiosísimo tiempo y entusiasmo en dedicar a la atención del desarrollo de este trabajo. Dr. Edmundo Rocha por su valiosísimo tiempo y proponerme colaborar en el proyecto y por orientación y apoyo desde semestres tempranos. Al M.A. Yair Bautista , por el entusiasmo que ha fomentado en mí desde “Jóvenes hacia la investigación” en 2015 así como la orientación, entusiasmo y apoyo generalizado a sus estudiantes. Así mismo al sínodo Mtra. Fernanda Merino e Ing. Ana Marissa Juárez.

A la Universidad Nacional Autónoma de México, mi máter, por la formación

integral académica y cultural. A mis profesores Margarita Puebla, en Ética y Literatura, Héctor Sánchez de Ing. Económica, José M Téllez-G de Máquinas Eléctricas, Sergio Crail de Cálculo Integral, que me han transmitido su pasión y entusiasmo.

A la Fundación Robert Bosch México, que cree y apuesta en el talento joven de universidades en México con acciones como el programa MexCellence, del cual me siento comprometido y agradecido de haber pertenecido.

A Leslie Sánchez, por los enésimos momentos de escucha en ambos sentidos y gran amistad genuina que la ENP. 9 formó y la F.I. consolidó. Con quien la reflexión permeada de su visión y valores siempre encaja en cada i-ésima charla. Por conectarnos en la virtualidad, a dedicarnos a nuestras actividades y poder concentrarnos en nuestros proyectos.

A Gabriela Haideé, por su amistad auténtica que se cultivó desde ENP 9, y continúa nutriéndose viendo al otro crecer. Por incontables charlas, escucha, silencios y anécdotas; estando siempre en lo dark y en lo bright en apoyo incondicional. A quien estimo increíblemente y cuya personalidad suma invariablemente del ambiente donde esté.

A Eduardo, por la amistad de más larga duración que tengo desde primer año de secundaria y siempre estar, escuchando con empatía en diversos momentos y rebotando percepciones. Me entusiasma el crecimiento de ambos en tan distintas áreas.

A Miguel, amistad con quien la reflexión, introspección y simplezas tienen igual cabida, por las innumerables experiencias, horas de escucha recíproca, kilómetros en bicicleta, y compartir enfoques.

A Víctor y Saúl, profesionistas de formación que compartimos, mis compitas, amistades que el viaje por la facultad me dió y a quienes estimo muchísimo y con quienes - las risas no faltaron - y ni faltarán,

A Ingrid R., amistad que Guadalajara creó con horas de escucha recíproca. A Josu y Fernando, compitas desde ENP 9 y durante FI. A Karla R, Tania O., Ruelas. Agradecer también a aquellos con quien he compartido momentos y en este momento quizá se me están pasando.

A la mascota de la familia, Lula, la perrita más cariñosa y juguetona, cuyos sentimientos me transmite a través de sus ojos.

Sin planearlo, terminé de pulir estas últimas líneas de agradecimientos estando en una banca del Centro Cultural Universitario, a un costado de la Sala Nezahualcóyotl un día parcialmente despejado 23 de agosto de 2022 con G. Cerati de fondo.

Abstract

English:

Ongoing research is addressing the impact human mobility and thus, contacts, have on the development of diseases. It is stated by Schläpfer et al on their work "Human mobility impacts many aspects of a city, from its spatial structure to its response to an epidemic" (Schläpfer et al., 2021). On the grounds of the latter, the aim of this work is to explore and pose a solution which enables the prediction of the number of contacts an individual is exposed to.

The data acqusition process supported by several technologies as well as the analysis development will be stated and discussed. A foremost contribution of this solution is that key insights are provided by the Logistic Regression Machine Learning Model implemented by the author, enabling a wide scope for epidemiological models to be enhanced, leading to better understanding of certain diseases. Consequently, these insights encourage data-driven decisions to make feasible successful public policies.

Español:

Existe actualmente investigación que se encarga del impacto que la movilidad humana y, consecuentemente, los contactos, tienen en el desarrollo de enfermedades. Está descrito por Schläpfer et al. en su trabajo ["La movilidad humana impacta muchos aspectos de una ciudad, desde su estructura espacial a su respuesta a una epidemia"] (Schläpfer et al., 2021) . A partir de lo anterior, el objetivo de este trabajo es explorar y proponer una solucion la cual habilite la predicción del número de contactos a los cuales un individuo es expuesto.

El proceso de adquisición de datos apoyado de varias tecnologías así como el desarrollo del análisis serán indicados y discutidos. Una principal contribución de esta solución es que perspectivas clave serán proveídas por un Modelo de Regresión Logística de Machine Learning implementado por el autor, habilitando un amplio rango de modelos epidemiológicos a ser mejorados, lo cual lleva a un mejor entendimiento de determinadas enfermedades. Consecuentemente, estas percepciones fomentarán las decisiones basadas en datos para crear viables y exitosas políticas públicas.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

La movilidad del ser humano ha sido variable en la historia. Esta ha sido un factor determinante en el destino, prevalencia u ocaso de ciertas culturas y sociedades durante el tiempo que la especie humana ha estado habitando el planeta.

Schläpfer et. al. menciona *"Human mobility impacts many aspects of a city, from its spatial structure to its response to an epidemic. It is also ultimately key to social interactions, innovation and productivity"* [La movilidad humana impacta muchos aspectos de una ciudad, desde su estructura espacial a su respuesta a una epidemia. Es incluso ultimadamente clave para las interacciones sociales, la innovación y productividad] en su trabajo "The universal visitation law of human mobility"(Schläpfer et al., 2021) lo cual marca una primera pauta para entender la relevancia de la movilidad humana cuando tiene que responder frente a una epidemia.

1.2. Objetivo

Diseñar de un dispositivo que permita registrar el número de contactos que tiene una persona, en el marco de la pandemia de COVID - 19.

Como objetivos específicos, se encuentra: Aplicación de aprendizaje máquina nutrido con información de sensores percibiendo el entorno a través de un arreglo de elementos de electrónica y haciendo uso de las tecnologías de dispositivos móviles soportados con servicios en la nube.

1.3. Motivación

Así como se establece en el trabajo "El número reproductivo básico R_0: consideraciones para su aplicación en la salud pública" de Rindenhour, Kowalik y Shay publicado en 2018, previo al surgimiento de la pandemia de la enfermedad COVID - 19, causada por el virus SARS - CoV - 2, se presenta a las estimaciones del parámetro R_0 como medidas coadyuvantes al entendimiento de la transmisión de una enfermedad. Mencionan que *"... la estimación del R_0 en una población determinada es útil para entender la transmisión de una enfermedad en ella. Si se considera el R_0 en el contexto de otros parámetros epidemiológicos importantes, su utilidad puede consistir en que permite conocer mejor un brote epidémico y preparar la respuesta de salud pública correspondiente"* (Ridenhour et al., 2018)

En ese sentido, el presente trabajo de tesis *"Inteligencia Artificial aplicada en el diseño de un dispositivo para registrar el número de contactos de una persona en el marco de COVID-19"* nace como una propuesta de solución de elemento coadyuvante a la estimación de parámetros útiles en modelos epidemiológicos a través del entendimiento del comportamiento social de un individuo.

A través de diversas herramientas: Electrónica, Infraestructura IoT (Internet of Things), Conexiones a la nube, Análisis Data Science del dataset obtenido y con Modelos de Machine Learning se logra esta propuesta de dispositivo. El deseo es aportar información valiosa en la comprensión de enfermedades donde el contacto físico forma un factor de riesgo para asesorar en la toma de decisiones en favor del bienestar y salud del grupo demográfico en cuestión.

1.4. Planteamiento del problema

En el contexto por la pandemia de COVID-19 un elemento fundamental en la transmisión de la enfermedad son los contactos físicos. Existe una problemática en el seguimiento del número de contactos que tiene un individuo durante su día a día. En este trabajo se aborda la problemática de obtención de información de contactos y datos con mayor exactitud para estimaciones en el número de contactos de una persona.

1.5. Contribuciones

Este trabajo propone un dispositivo que hace uso de múltiples tecnologías y herramientas: Electrónica, IoT, Cloud, Data Science y Machine Learning para estimar el número de contactos de una persona. El objetivo es que estas predicciones puedan ser usadas para contribuir al mejoramiento de modelos epidemiológicos y un mayor entendimiento de la dinámica de ciertas enfermedades para coadyuvar en el establecimiento de políticas de salud pública con base en evidencia.

1.6. Estructura del documento

Este documento está dividido en nueve capítulos de la siguiente manera

- **Capítulo 1. Introducción.**

  Se presenta Estado del Arte, Objetivo, Motivación, Planteamiento del problema, Contribuciones y Estructura del documento.

- **Capítulo 2. Necesidades y Especificaciones.**

  Se establecen las necesidades y especificaciones que actúan como directriz en el diseño y desarrollo del dispositivo.

- **Capítulo 3. Creando una Solución de Dispositivo.**

  Se propone una solución de dispositivo: cómo se avanza en el proceso.

- **Capítulo 4. Instrumentación y Electrónica.**

  Profundiza y detalla el proceso de instrumentación y electrónica utilizados en la solución.

- **Capítulo 5. Aplicación Móvil: App Android.**

  Se detalla el proceso de la aplicación y razón de toma de decisiones, así como las pantallas con las cuales el usuario interactúa.

- **Capítulo 6. Almacenamiento en la nube: Google Firebase Realtime Database.**

  Profundiza en la estructura de datos en que son almacenados en la nube en la solución de Google.

- **Capítulo 7. Análisis Exploratorio de Datos.**

  Análisis en el cual se encuentran patrones y entendimiento de los datos recopilados. Se discuten los resultados y correlaciones más relevantes.

- **Capítulo 8. Inteligencia Artificial y Modelos de Machine Learning: Resultados y Discusión.**

  Aborda el Modelo de Inteligencia Artificial así como profundización en su funcionamiento, resultados y la discusión de los mismos.

- **Capítulo 9. Conclusiones y trabajo a futuro.**

  Informa lo abordado en todo el documento, detalla el potencial y se esbozan potenciales mejoras en trabajo a futuro.

Adicionalmente, se cuenta con cuatro apéndices con los códigos desarrollados durante todo el proceso.

- Apéndice A. Códigos de Aplicación Móvil Android.
- Apéndice B. Códigos de Funcionamiento del dispositivo.
- Apéndice C. Códigos de tratamiento de datos.
- Apéndice D. Códigos de Machine Learning y de tratamiento de información geográfica.

Capítulo 2

Necesidades y Especificaciones

2.1. Introducción

En este capítulo se abordarán especificaciones y necesidades asociadas al dispositivo. Para la realización de esta sección del documento se analizaron estructuras utilizadas previamente propuestas en trabajos donde un dispositivo que pasó a través del proceso de diseño es pieza clave.

De esta manera, un recurso bibliográfico en la materia es el trabajo de Ulrich y Eppinger "Diseño y desarrollo de productos" establecen capítulos correspondientes a distintas partes específicas del proceso del desarrollo de un producto. La forma modular de capítulos de este libro, incluso en intención de los autores, fomentan que aquellos *" ... profesores, estudiantes y practicantes pueden tener fácil acceso sólo al material que encuentren más útil..."* (Ulrich and Eppinger, 2013).

En ese sentido, durante el desarrollo de este capítulo se muestran aquellas necesidades identificadas en este dispositivo tomando como referencia el trabajo de Ulrich y Eppinger con las especificaciones consecuentes.

Se abordará el concepto *wearable* cuando se esté describiendo al dispositivo. Los wearables también son conocidos como tecnología vestible, una definición es: *"Son dispositivos electrónicos inteligentes incorporados a la vestimenta o usados corporalmente como implantes o accesorios que pueden actuar como extensión del cuerpo o mente del usuario. Se ha popularizado con el consumo exponencial de los relojes inteligentes y los seguidores de actividad. Aparte de usos comerciales, esta tecnología está siendo incorporada a la navegación de sistemas, avances en la industria textil y la salud."* (28).

2.2. Necesidades

En esta sección se trabajan las necesidades mediante reglas propuestas en el trabajo de Ulrich y Eppinger, de tal suerte que se tendrán cinco directrices que guiarán la identificación de necesidades:

- "Qué" y no "cómo".
- Especificidad.
- Positiva, no negativa.
- Un atributo del producto.
- Evitar "debe" y "debería".

Bajo estas directrices es posible esbozar las siguientes necesidades identificadas para el dispositivo estilo wearable:

1. El dispositivo recaba información mensurable del entorno en que se encuentra operando.
2. El dispositivo cuenta con conexión Bluetooth a un dispositivo móvil durante la operación.
3. El dispositivo cuenta con batería suficiente para un día de uso de aproximadamente 8 horas.
4. Las información del dispositivo es guardada correctamente en una base de datos en la nube.
5. La información del dispositivo es suficiente para que los modelos de machine learning cuenten con métricas de desempeño altas, maximizando los positivos verdaderos y los negativos verdaderos, reduciendo al máximo aquellos falsos negativos y falsos positivos.
6. La aplicación asociada al dispositivo permite una interacción suave y amigable con el usuario.
7. El código del microcontrolador, desarrollo de aplicación móvil, análisis de datos y de aprendizaje máquina cuentan con estandarización y buenas prácticas.
8. El dispositivo opera normalmente en individuos de estaturas distintas.

En próximas iteraciones del desarrollo del dispositivo será posible identificar más necesidades de acuerdo a las condiciones del ambiente, grupos demográficos y mayores alcances de entendimiento del entorno por parte del wearable.

2.3. Especificaciones

De acuerdo con lo estudiado por Ulrich y Eppinger, se establecen especificaciones con fin de que el grupo de desarrollo tenga con detalles precisos y medibles, "...lo que el producto tiene que hacer." (Ulrich and Eppinger, 2013). En ese sentido, se presentarán especificaciones y las necesidades asociadas.

Bajo la premisa de que deben ser medibles, se presenta la columna de Unidades, para tener la referencia adecuada de estas métricas. Asimismo, en la referencia mencionada se sugiere un nivel de importancia de más baja con un nivel 1 hasta importancia más alta con un nivel 5.

Identificación de especificaciones y necesidades asociadas					
No. de Esp	Especificación	Valores	Unidades	Importancia	Necesidad Asociada
1	Mide temperatura	10 a 40	[°C]	5	1
2	Mide distancia	0.1 a 3	[m]	5	1
3	Conexión Bluetooth	Continua	-	5	2
4	Batería y autonomía	8	[h]	4	3
5	User-friendly	Alta	-	5	6
6	Conexión a nube	24 al día	[h]	5	4
7	Cloud Storage	1	[GB]	5	4
8	Data disponible	24 al día	[h]	5	5
9	Calidad de código	Buenas prácticas	-	4	7
10	Adaptabilidad	+Campo sensores	-	4	8

Tabla 2.1: Especificaciones del dispositivo y necesidades asociadas

Como es mostrado en la tabla 2.1, se muestran los valores sobre los que oscilarán estas especificaciones. En cuanto a la medición de temperatura se habla de

este rango de 10 a 40 [°C] pues el subconjunto de las temperaturas de una persona en estado normal y una con fiebre se incluye en el rango establecido. Asimismo, se incluye la temperatura ambiente estándar de 25 [°C].

De acuerdo a diversas fuentes, el distanciamiento social se ha convertido clave en la prevención de contagios de COVID-19, en ese sentido, que un sensor sea capaz de medir distancias en estos rangos y mayores es necesario, es así como se propone un máximo de tres metros y un mínimo de 10 centrímetros.

En cuanto a la Conexión Bluetooth, será necesaria de forma continua para poder hacer la transferencia de la información del entorno captada por los sensores, procesadas con el microcontrolador y enviadas a la app móvil a través de Bluetooth.

Una vez dentro del dispositivo móvil la información necesitará conexión a la nube para poder hacer la sincronización a la base de datos, se sugiere siempre esté conectado el dispositivo para lograr tener la información más reciente en todo momento. Un almacenamiento de al menos 1 GB se considera suficiente para primeras iteraciones de la versión de desarrollo del dispositivo, esta capacidad será escalable al número de usuarios que se tengan.

Autonomía del dispositivo es relevante pues tendrá que alimentar el conjunto de electrónica, de tal forma que pueda funcionar al menos, en un lapso de ocho horas al día.

El grupo demográfico será variado lo cual hace necesario que el diseño de la aplicación sea lo más amigable posible e intuitiva con el usuario para su correcto funcionamiento.

Los análisis de información serán ejecutados continuamente, de tal suerte que la información se necesita disponible en todo momento, lo cual es posible con la mayoría de proveedores de almacenamiento en la nube.

La programación de los elementos implícitos en cada una de las partes de desarrollo del dispositivo serán de considerables líneas de código, lo cual hace un caso de uso para la implementación de buenas prácticas: fomentará una transferencia de conocimiento más sostenida y suave para un escalamiento del proyecto.

Finalmente, un campo de “visibilidad” de los sensores es necesario pues con toda seguridad los usuarios tendrán un rango de estaturas que difieran por varias unidades de centímetros, incluso varias decenas, en ese sentido, un mayor alcance es requerido.

Capítulo 3

Creando una Solución de Dispositivo

3.1. Introducción

En este capítulo se profundiza en la problemática a resolver y se presentará la propuesta de solución: se abordarán los sistemas de los cuales el dispositivo se compone, flujo de trabajo y la finalidad de este.

3.2. Definición del Problema

La dinámica de contacto social forma parte del día a día del ser humano. En ese sentido, cuando en un contexto donde el número de contactos físicos tiene implicaciones en el comportamiento de una enfermedad y consecuentemente, en los modelos epidemiológicos, se crea el espacio para una solución que apoye en la estimación de este número. Existiendo sensores que aportan información del entorno en que se encuentran a través de variables mensurables como temperaturas, distancias, ubicación de latitud y longitud en el planeta, se plantea la siguiente solución que aporta datos hacia la estimación de número de contactos de una persona.

3.3. Propuesta de Solución

Tomando en cuenta la identificación de necesidades y generación de especificaciones, se trabajará en las siguientes áreas de la solución:

- Descripción del entorno a través de variables medibles.

- Instrumentación de sensores y electrónica necesaria.
- Aplicación móvil para el usuario.
- Conexión a base de datos en la nube.
- Análisis exploratorio de los datos.
- Algoritmos de Inteligencia Artificial en Machine Learning.

Con la integración de los elementos mencionados anteriormente se desarrollará la solución en las secciones de capítulos de este documento. En la Figura 3.1 se muestra un diagrama del comportamiento de la solución.

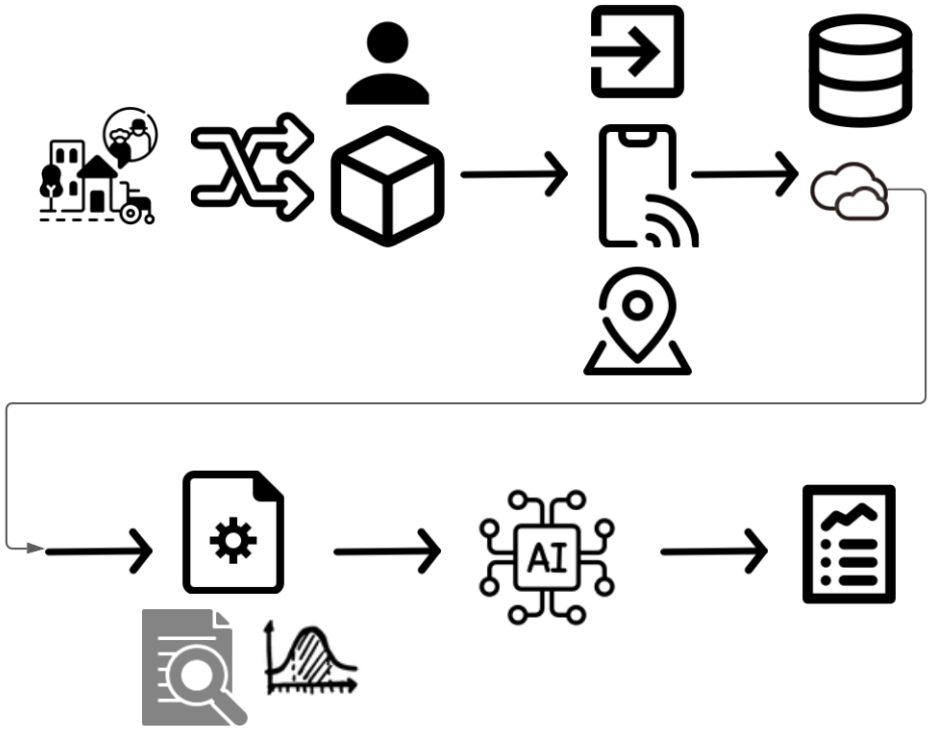


Figura 3.1: Diagrama de la propuesta de solución (31)

3.4. Resumen

Como es mostrado en la Figura 3.1 el flujo queda de la siguiente manera:

- Adquisición de datos del entorno.
- Recopilación de variables de entorno por el dispositivo que el usuario está utilizando.
- Envío de datos a la aplicación móvil para continuar el flujo. En esta etapa también son recopilados los datos de ubicación geográfica.
- Envío a base de datos en la nube.
- Consumo de datos en la nube para ejecución de análisis exploratorio de datos.
- Entrenamiento y test de modelos de Inteligencia Artificial.
- Resultados.

Capítulo 4

Instrumentación y Electrónica

4.1. Introducción

Durante este capítulo se detalla la instrumentación y la sinergia de estos elementos con la electrónica necesaria para su funcionamiento, se abordan características de estos elementos así como el papel que juegan dentro del dispositivo completo.

4.2. Sensores

Para las variables que se medirán en este dispositivo se generaron propuestas de sensores potenciales de colocar. La selección de los que finalmente se instrumentaron se detalla en cada uno.

4.2.1. Medición de Temperatura

Teniendo como necesidad detectar presencia humana se realizó una investigación de opciones de sensores potenciales para el caso de uso del dispositivo; el siguiente producto desarrollado por OMRON habilita la detección de presencia humana. De tal forma, el sensor de temperatura utilizado en el dispositivo es OMRON D6T44L06, ilustrado en la Figura 4.1.

El principio de operación es que una lente recibe los rayos infrarrojos irradiados por objetos con temperatura. Esto excita con una fuerza electromotriz al sensor y con un circuito analógico es calculada la temperatura observada.

Una de las principales ventajas de este sensor, es que a diferencia de los

Figura 4.1: Sensor OMRON D6T44L06 (4)

sensores piroeléctricos convencionales utilizados para detectar presencia, que solo son sensibles a cambios en la radiación, el sensor OMRON D6T44L06 continúa con la detección a pesar de que el sujeto en cuestión quede estacionario.

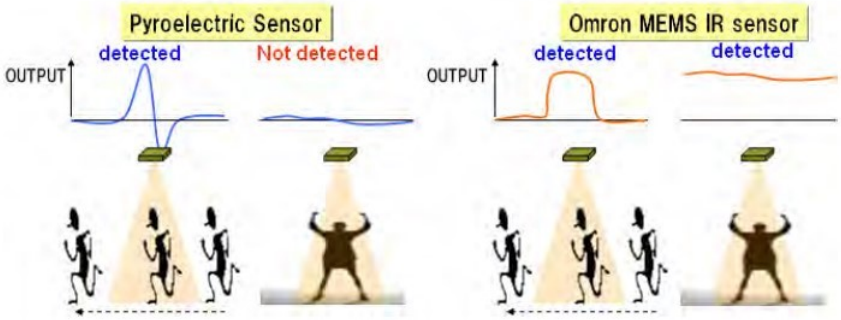


Figura 4.2: Diferencia entre OMRON y sensores piroeléctricos convencionales (1)

Como es mostrado en la Figura 4.2 el sensor OMRON es capaz de continuar detectando la presencia humana a pesar de que esta esté estática. El campo de visión es amplio, mostrado en la Figura 4.3.

Las mediciones de este sensor son a través de una matriz de temperaturas que se comunican a través de un protocolo de comunicación I^2C. El comportamiento es ilustrado en 4.4.

Se seleccionó el sensor para ser el principal proveedor de información del entorno del dispositivo. El costo de éste sensor, en la fecha de compra del 10 de agosto de 2020, fue de 38.94 dólares estadounidenses.

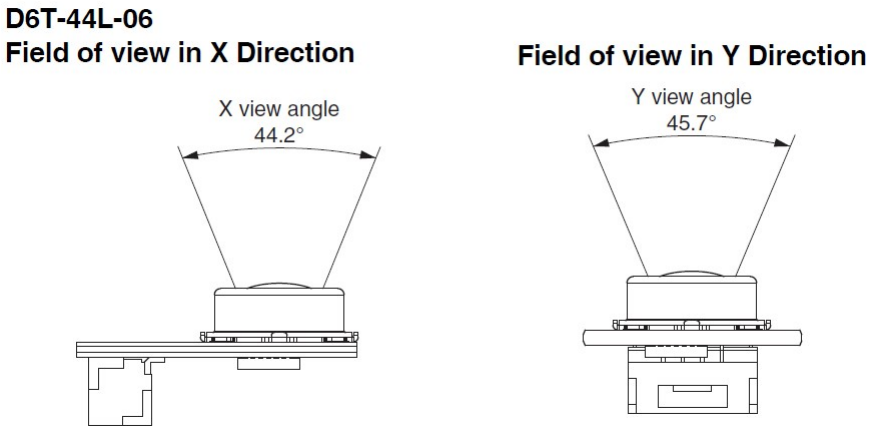


Figura 4.3: Campo de visión del sensor D6T44L06 (4)

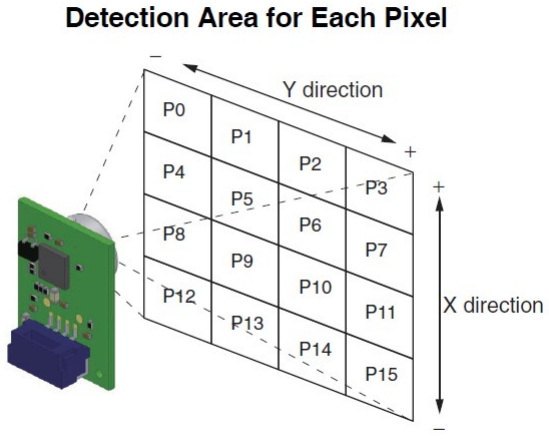


Figura 4.4: Matriz de temperaturas del sensor D6T44L06(4)

4.2.2. Medición de Distancia

Durante el caso de uso de medición de contactos, la distancia a la cual se encuentra el individuo en cuestión de las demás personas, se convierte un factor importante.

Para sortear esta necesidad, se cuenta con principalmente dos principios de operación de sensores de distancia: ópticos y ultrasónicos. La oferta y disponibilidad de ambos tipos es extensa. El caso del dispositivo lo hace propenso a ser operado en condiciones de exteriores: existirá luz solar, un ambiente donde ondas

electromagnéticas del espectro visible y no visible pueden causar interferencias en las mediciones en sensores cuyo principio de operación es óptico; durante la formación en la Facultad de Ingeniería, haciendo uso de este tipo de sensores, lo expresado ha sido empíricamente comprobado.

Los sensores cuyo principio de funcionamiento es ultrasónico son preferidos para el uso que se le dará al dispositivo. Finalmente, para el prototipo del dispositivo se decidió hacer uso del sensor HC-SR04, cuyo rango de medición va de los 2 [cm] a 400 [cm], se puede observar en la Figura 4.5

Figura 4.5: Sensor de distancia HC-SR04 (24)

Este sensor alimentará de información de distancia al microcontrolador, en el cual se hará enfásis más adelante. El principio de funcionamiento es el ilustrado en la Figura 4.6, donde a través de una señal de trigger proporcionada por el microcontrolador se envían pulsos ultrasónicos desde el emisor del sensor, mientras que una señal echo está encendida durante el tiempo que tarda en regresar el pulso ultrasónico al receptor. El pulso es rebotado por el objeto obstáculo a una distancia determinada.

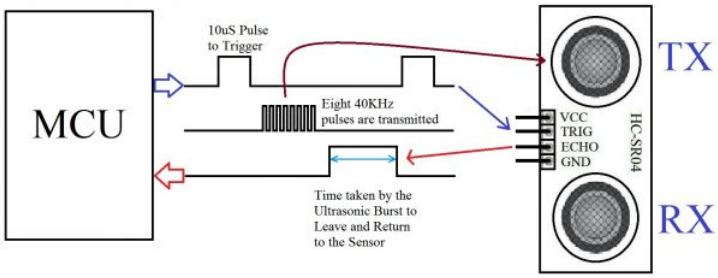


Figura 4.6: Funcionamiento del sensor ultrasónico HC-SR04 (11)

Esta información dará entendimiento de la distancia a la cual fueron los contactos.

4.3. Módulo Bluetooth

A partir de la necesidad de que el dispositivo tenga conexión con una app móvil, se propuso la vía Bluetooth. El módulo que se conecte al dispositivo debe ser capaz de enviar información al dispositivo móvil actuando como *maestro* y su vez, recibir instrucciones, de tal forma que actúa como *esclavo*. Habiendo identificado estas circunstancias, los módulos Bluetooth de bajo costo considerados fueron los siguientes:

- HC-05
- HC-06

El módulo HC-05 es capaz de actuar como maestro y esclavo, siendo superior en especificaciones técnicas sobre el HC-06 que solo permite actuar como esclavo. De esta forma, se eligió al módulo HC-05 para el dispositivo; en la Figura 4.7 se muestra.

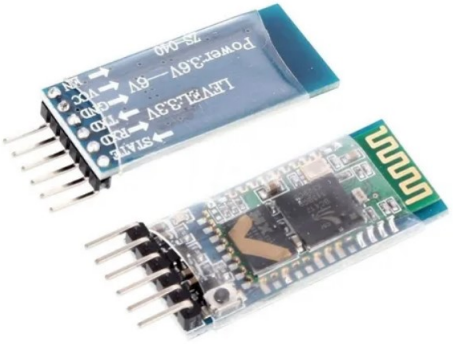


Figura 4.7: Módulo Bluetooth HC-05 (21)

Este módulo se comunicará con el microcontrolador con el puerto serial.

4.4. Microcontrolador

El microcontrolador para el dispositivo requiere que pueda leer la comunicación I^2C proveniente del sensor de temperatura, así como puertos digitales y analógicos tanto para la electrónica restante, como LEDs auxiliares y conexión con el sensor de distancia ultrasónico. Finalmente también requiere que soporte la comunicación serial para el módulo Bluetooth. Algunos de los MCU (Microcontroller Unit) propuestos fueron los siguientes:

- PIC16F887 - Microchip Technology (23)
- Arduino UNO - Arduino Open Source electronics platform. Plataforma de desarrollo basada en el microcontrolador ATmega328P (7)
- ESP32 - Espressif Systems (12)
- Texas Instruments MSP430G2553 - Texas Instruments INC (22)
- Arduino Nano - Arduino Open Source electronics platform Plataforma de desarrollo basada en el microcontrolador ATmega328 (6)

Con características necesarias y suficientes, el Arduino Nano en caso de aplicación del dispositivo encaja adecuadamente, mostrándose en la Figura 4.8

Figura 4.8: Arduino Nano (6)

La cantidad de pines es suficiente, con una velocidad de reloj de 16 [MHz] que permite la aplicación de los elementos sin mayor problema y el voltaje de operación estándar de 5[V]. Con pines útiles para la comunicación I^2C y protocolo de comunicación serial soportado. Adicionalmente, las dimensiones son reducidas.

Especificaciones técnicas del Arduino Nano son las expresadas en la Tabla 4.1 con datos obtenidos de Arduino (6)

Especificaciones Técnicas Arduino NANO	Valor
Microcontrolador	ATmega328
Arquitectura	AVR
Voltaje de operación	5[V]
Memoria Flash	32 [KB] - Bootloader: 2 [KB]
SRAM	2 [KB]
Velocidad de reloj	16 [MHz]
Pines IN Analog	8
EEPROM	1 [KB]
DC corriente por I/O Pins	40 [mA] (I/O Pins)
Voltaje de entrada	7-12 [V]
Pines I/O Digitales	22 (6 de PWM)
Salidas PWM	6
Consumo de energía	19 [mA]
Tamaño PCB	18 x 45 [mm]
Peso	7 [g]

Tabla 4.1: Especificaciones Técnicas del Arduino Nano - Valores tomados de la tienda oficial de Arduino (6)

4.5. Circuito electrónico

Para la construcción física del dispositivo se utilizó como guía un esquema de diagrama de circuito electrónico, donde se toman en cuenta los resistores necesarios en la comunicación I^2C, así como LEDs indicadores de estatus de conexiones, auxiliares en el proceso de diseño de las comunicaciones Bluetooth, de sensores y adquisición de mediciones. El diagrama es mostrado en la Figura 4.9.

Se puede ver en la Figura 4.9 el sensor de temperatura OMRON D6T es representado de forma de rectángulo sin embargo, su forma real se encuentra en la Figura 4.1.

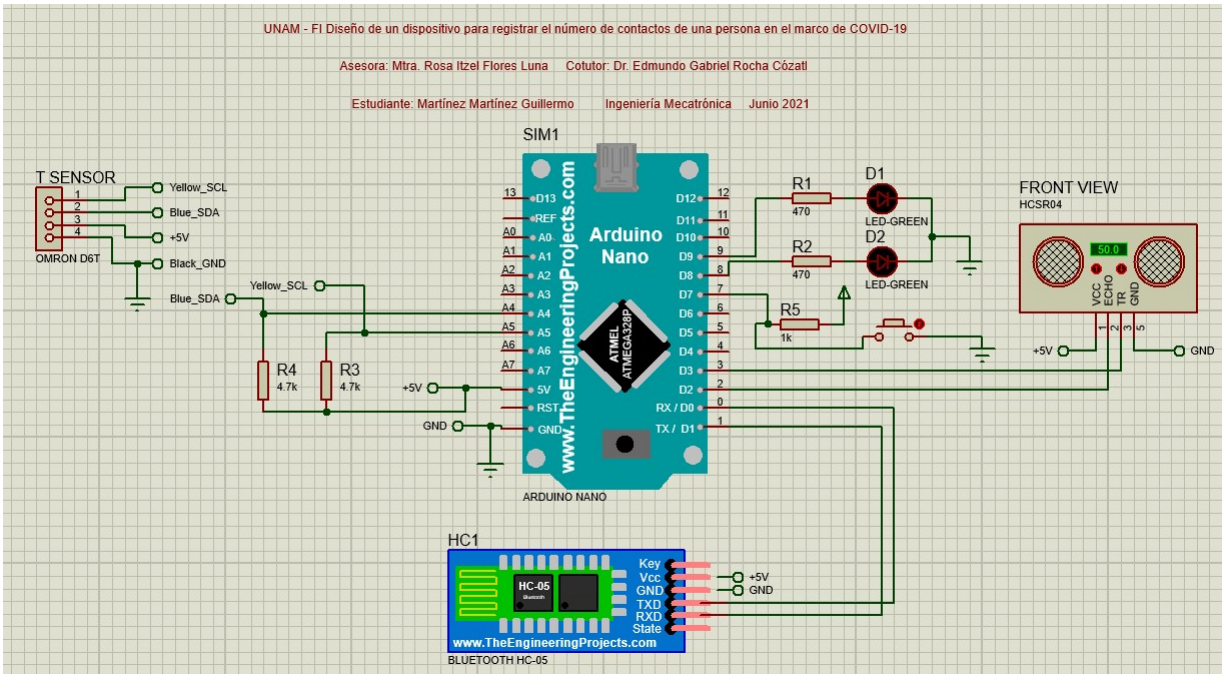


Figura 4.9: Diagrama circuito electrónico del dispositivo

4.6. Diseño de PCB

Con la finalidad de la escalabilidad del proyecto se contempló también el diseño de una Printed Circuit Board (PCB), su diseño es mostrado en las figuras 4.10 y 4.11.

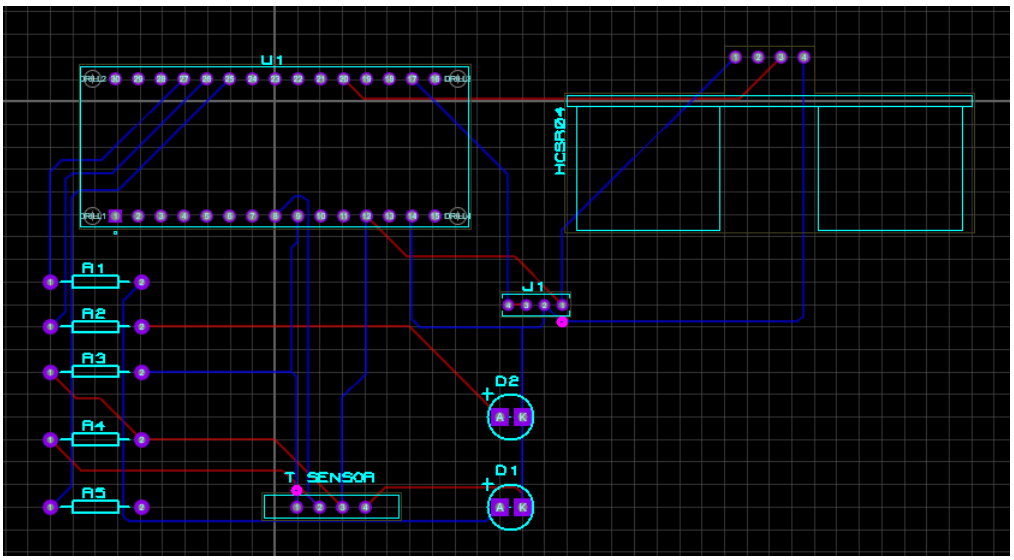


Figura 4.10: Layout de PCB con los elementos del circuito

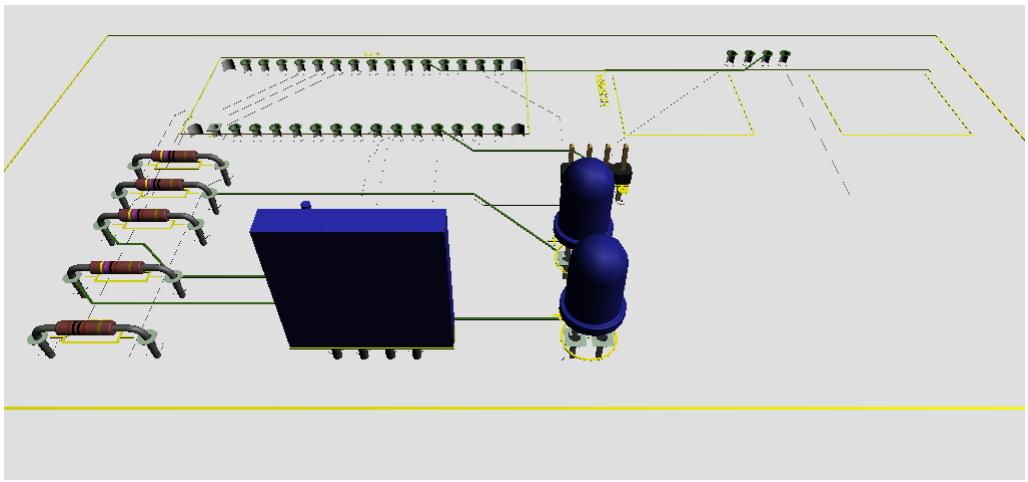


Figura 4.11: Vista 3D del resultado de la tarjeta PCB impresa

Capítulo 5

Aplicación Móvil: App Android

5.1. Introducción

En este capítulo se aborda el desarrollo de la aplicación móvil que se conectará con el dispositivo: es la entrada de la información del dispositivo hacia la interacción con el usuario y con la nube.

5.2. Seleccionando Android

Deseando que el dispositivo funcione con un número elevado de individuos para agregar diversidad de datos a las muestras, se decidió tomar el sistema operativo móvil cuya porción del mercado sea mayoritaria. De acuerdo con datos de Statcounter, con datos en su reporte "Mobile Operating System Market Share Worldwide" que comprende el periodo de enero de 2009 a junio de 2018 en la Figura 5.1, es evidente la creciente dominancia de Android. Ahora bien, con datos del mes de Julio 2021, Android participa con un 72.18 % en el mercado de sistemas operativos móviles. (20).

5.3. Android Studio

De acuerdo con el recurso web de Android Developers (19), se menciona que *"Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that*

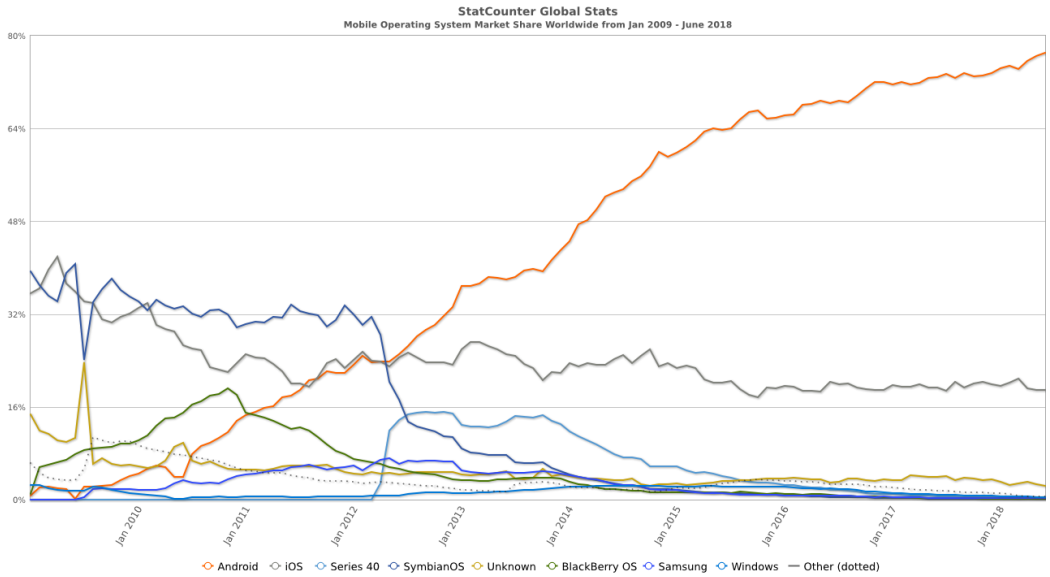


Figura 5.1: Dominancia en el mercado mundial de sistemas operativos móviles de enero 2009 a junio 2018. (20)

enhance your productivity when building Android apps" [Android Studio es el Entorno de Desarrollo Integrado (IDE) oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA . Además del potente editor de código y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que mejoran la productividad al crear aplicaciones para Android...] (19).

Con base en lo anterior, Android Studio se convierte a la herramienta oficial y el entorno adecuado para el desarrollo de la app del dispositivo.

5.4. Lenguaje de programación: Kotlin vs Java

Dentro de Android Studio se puede elegir entre programar utilizando el más reciente lenguaje Kotlin o bien el más tradicional lenguaje Java. A continuación se muestran un poco sobre ambos y por qué se eligió Kotlin.

5.4.1. Java

Lanzado en 1995 y desarrollado por James Gosling en Sun Microsystems, es un lenguaje de código abierto, de propósito general y un lenguaje de programación orientado a objetos. Adicionalmente, es un lenguaje estáticamente tipado, revisando tipos en el tiempo de compilación. Comparte similitudes con C y C++ pero ofrece menos servicios de bajo nivel (18)

Es un excelente lenguaje entendible lo cual hace que varios programadores lo elijan. Adicionalmente, ha tenido una presencia durante varios años lo cual hace que la documentación disponible sea más mayor y que su probabilidad de ser reemplazado sea baja. (18)

5.4.2. Kotlin

Este lenguaje de programación es más joven que Java, se introdujo en 2016, de código abierto que también puede compilar a bytecode y ejecutarse en una Java Virtual Machine, permitiendo que funcione en casi cualquier plataforma (18). Es compatible con Java e inspirado en él, Kotlin pretende expresar una versión mejorada que sea más limpia, más sencilla, más rápida de compilar y que suponga una mezcla de programación orientada a objetos y funcional (18).

Ha cobrado adopción generalizada en el desarrollo de diversas apps en el mercado (17). Entre las características más promocionadas por Android Studio (17) se encuentran, entre otras:

- Expresivo y conciso: Las modernas características del lenguaje Kotlin le permiten centrarse en la expresión de sus ideas y escribir menos código repetitivo.
- Código más seguro Con @Nullable y @NonNull incluidos en su sistema de tipos, Kotlin te ayuda a evitar NullPointerExceptions. Las aplicaciones Android que utilizan Kotlin tienen un 20 % menos de probabilidades de fallar.
- Interoperable Kotlin es 100 % interoperable con el lenguaje de programación Java, así que puedes tener tan poco o tanto de Kotlin en tu proyecto como quieras.

5.4.3. Decisión: Kotlin

Ante lo presentado y con la documentación creciente de Kotlin y su amplia adopción por la comunidad programadora, se decidió utilizar Kotlin para este

proyecto. Adicionalmente, la cantidad de líneas de código utilizadas para realizar ciertas tareas en Java son significativamente más que las requeridas en Kotlin.

5.5. Sobre el idioma

Es importante mencionar que el desarrollo de esta app está pensado en alcanzar un universo de usuarios cuya magnitud, entre mayor, es mejor, pues la disponibilidad de datos incrementará. Esto aumenta la confianza en resultados y estadísticas creadas a partir de un número de muestras mayor. En ese sentido, el idioma en el cual la aplicación esté disponible es relevante. En esta versión se encuentra disponible en dos idiomas: Español e Inglés, idioma que es automaticamente seleccionado a partir del idioma principal del dispositivo móvil del usuario.

5.6. Sobre el código

El código base de toda esta aplicación se encuentra en el Apéndice 1 del presente trabajo. Se encontrarán los siguientes códigos:

- Android Manifest: Es un archivo XML el cual contiene la metadata relevante del proyecto de aplicación Android. Dentro se encontrarán permisos que la aplicación requiere al dueño del dispositivo, el nombre del proyecto, así como nombres de los procesos activity y otras configuraciones.
- Gradle: Es un kit de herramientas de compilación avanzado utilizado para automatizar y gestionar el proceso de compilación. A su vez, permite definir configuraciones personalizadas y flexibles (5)
- Splash Screen Activity: Código en lenguaje Kotlin que incluye la lógica del proceso activity de splash screen
- Select Device Activity: Código en lenguaje Kotlin que incluye la lógica del proceso activity de la selección de dispositivos.
- Measurement: Código en lenguaje Kotlin donde se define la clase de estructura de una unidad de medición para la base de datos.
- LocTest: Código en lenguaje Kotlin donde se realizan las pruebas de localización con GPS y con la Red celular.

- Control Activity: Código en lenguaje Kotlin donde se encuentra la mayor parte de la lógica de la aplicación. Aquí se incluye la conexión con la base de datos Firebase, comunicación Bluetooth y comportamiento de los botones de la interfaz de usuario en la app.
- Archivos XML correspondientes a la parte gráfica de las actividades de la apliación

5.7. Actividades de la aplicación

5.7.1. Splash Screen

Una splash screen es un recurso gráfico utilizado generalmente como la primera impresión de experiencia de usuario con la aplicación móvil. Se imprime una imagen durante los primeros segundos de ejecución de esta. Esta aplicación cuenta con una splash screen que el usuario visualiza de la forma expresada en la figura 5.2.

Figura 5.2: Splash Screen de la aplicación

5.7.2. Select Device Activity

Inmediatamente después de la splash screen continúa una ventana donde al usuario se le muestra un mensaje donde se le solicita primero lea el manual de uso del dispositivo para poder continuar a usarlo. El usuario verá una pantalla donde podrá seleccionar qué desea hacer: Hacer un test de ubicación o directamente conectarse al dispositivo. Esta pantalla puede verse en la Figura 5.3. Es en este momento donde al usuario se le solicitarán los permisos de Bluetooth y Ubicación la primera vez que utilice la aplicación.

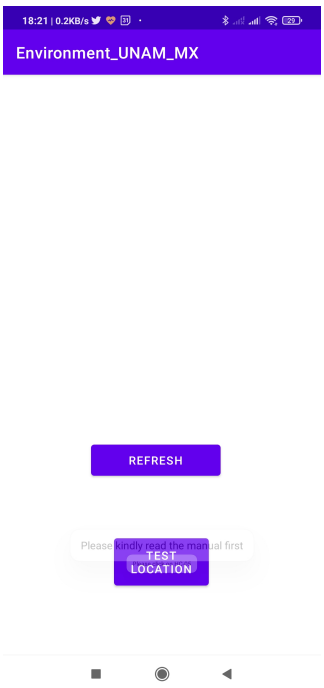


Figura 5.3: Actividad siguiente de la splash screen

Al momento de seleccionar el boton de refresh, se desplegarán los diversos equipos Bluetooth que el usuario ha utilizado, adicionalmente a este dispositivo. Si existieran diversos equipos ya utilizados, se deberá hacer scroll para visualizar los demás dispositivos Bluetooth han sido vinculados como se muestra en las figuras 5.4 y 5.5.

El dispositivo cuenta con un módulo Bluetooth HC-05 cuyo nombre es el mismo, HC-05. Sin embargo, es relevante mencionar que este nombre se puede modificar.

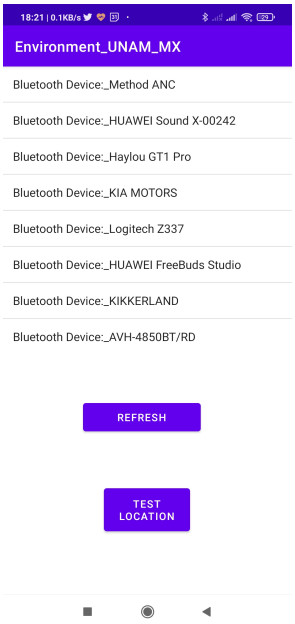


Figura 5.4: Selección de Dispositivos Bluetooth 1

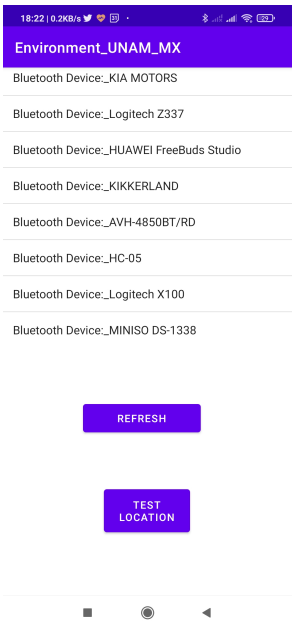


Figura 5.5: Selección de Dispositivos Bluetooth 2

5.7.3. Control Activity

Una vez seleccionado el dispositivo, aparecerá un mensaje que indica que la conexión Bluetooth se encuentra en progreso como lo muestra la Figura 5.6. Se señala que estos mensajes son expresados en el código como "Toast" para que el usuario pueda verlos durante un tiempo definido.

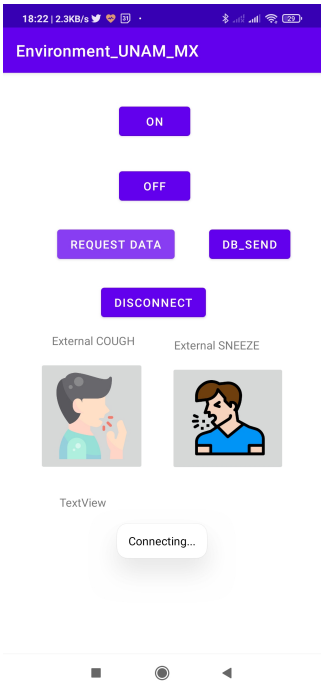


Figura 5.6: Conectando al dispositivo

Una vez conectado, dentro de la actividad de control es posible reportar cuando una persona tose o estornuda, cuando estos botones interactivos son presionados aparecerá un mensaje que reporta al usuario que en memoria se ha guardado esta información (Figura 5.7).

Asimismo, es posible dar una visualización local de las mediciones que se encuentra tomando el dispositivo en ese momento. En la Figura 5.8 es posible ver en el cuadro de texto información de medición desplegada: temperaturas, distancia y variable de etiqueta.

Finalmente, existe un botón para que, si es deseado, se inserte de forma manual la medición actual del dispositivo. El flujo de información al presionar este botón es el siguiente:

1. A través de comunicación Bluetooth se envía un caracter al dispositivo.

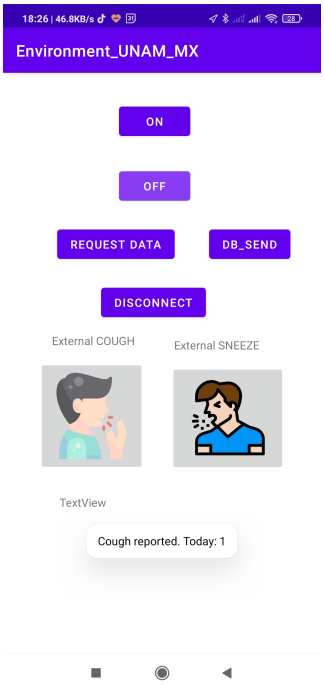


Figura 5.7: Conectando al dispositivo

2. Este caracter es recibido por el módulo Bluetooth y enviado a la plataforma de desarrollo Arduino.

3. El código de Arduino al momento de recibir este caracter hace la solicitud de información a través de protocolo I^2C al sensor de temperatura y en paralelo comienza una medición con el sensor ultrasónico.

4. Una vez recibida esta información, el microcontrolador la ordena en una cadena separada por comas y la envía a través del módulo Bluetooth a la app.

5. La app, al recibir una cadena con las características de una medición, comienza un proceso de parseo y a través de una clase crea el objeto que será enviado a la base de datos.

6. Internamente la app registra también los valores de ubicación, de tiempo y de número de personas con tos o estornudo. Posteriormente, se completa el objeto que será insertado en la base de datos.

7. La app, al estar en un dispositivo con conexión a internet, y habiendo creado ya la instancia de base de datos de Firebase, ejecuta la inserción del objeto JSON.

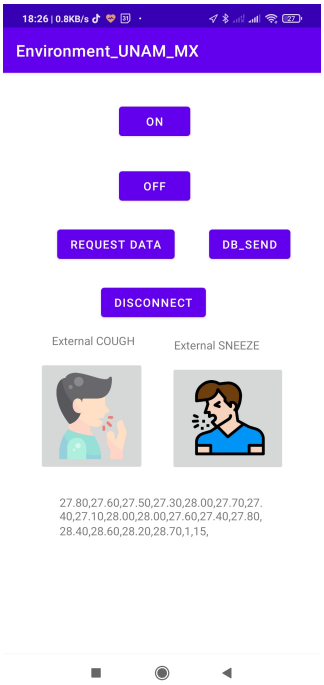


Figura 5.8: Ejemplo de solicitud de medición y despliegue al dispositivo

8. De esta forma, queda completo el flujo de información.

5.7.4. Ubicación

El servicio de ubicación funciona bajo el hecho de que el usuario ha otorgado los permisos necesarios que la app en su momento de ejecución solicita. En este contexto, existe un botón de test de ubicación el cual solicita la información tanto del GPS como de la red, los despliega y hace una ejecución continua de visualización de datos de ubicación. Durante este mismo proceso, se indica cuál de las dos mediciones cuenta con mayor exactitud. En la Figura 5.9 se visualiza el primer funcionamiento de ubicación y en la Figura 5.10 el continuo funcionamiento de actualización de datos de localización.

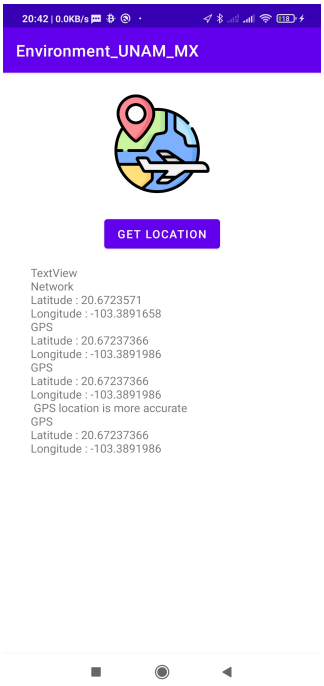


Figura 5.9: Primera solicitud de información de ubicación

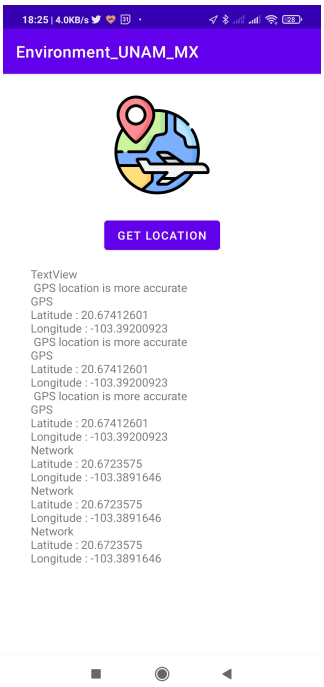


Figura 5.10: Continua solicitud de información de ubicación

Capítulo 6

Almacenamiento en la nube: Google Firebase Realtime Database

6.1. Introducción

En este capítulo se abordará el almacenamiento de la información recolectada por el dispositivo hacia la base de datos en la nube y el servicio elegido para esta objetivo. Se explicará la lógica de la estructura dentro de la base de datos así como el significado de los elementos.

6.2. Seleccionando Firebase Realtime Database

De acuerdo con Google en el sitio oficial de su producto Firebase, se define a The Firebase Realtime Database de la siguiente forma *"The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data"*[La base de datos en tiempo real Firebase es una base de datos en la nube. La información es almacenada en formato JSON y sincronizada en tiempo real para cada cliente conectado. Cuando usted construye en diversas plataformas con nuestro iOS, Android y JavaScript SDK, todos sus clientes comparten una instancia de Base de Datos en tiempo real y automáticamente recibe actualizaciones con la información más reciente.](13)

6.3. Estructura de los datos

La información es almacenada en formato JSON, el cual es acrónimo de (JavaScript Object Notation) siendo un formato ligero de intercambio de datos. El formato es de llave-valor. Un ejemplo sencillo de este formato es el siguiente:

```
{
  "no_cuenta": "0011223344",
  "edad": "24",
  "datos_identidad":
        {
        "nombre_pila": "Guillermo",
        "apellido_paterno":"Martinez",
        "apellido_materno":"Martinez",
        }
   "datos_geograficos":
        {
        "entidad":"CDMX,
        "pais":"Estados Unidos Mexicanos"
        }
}
```

6.4. Información de cada objeto JSON

Dentro de esta base de datos, cada objeto JSON representa una medición. Cada objeto contendrá la información siguiente. En la Figura 6.1 es posible observar resaltado en amarillo el título de la base de datos, expresada como "environment-unam-mx-default-rtdb" dentro de la cual existen llaves con el formato $measure\_i$ donde i es la i-ésima medición que es almacenada dentro de la base de datos. Cada una de estas mediciones contará con siete llaves para ingresar a los valores. A continuación se describen.

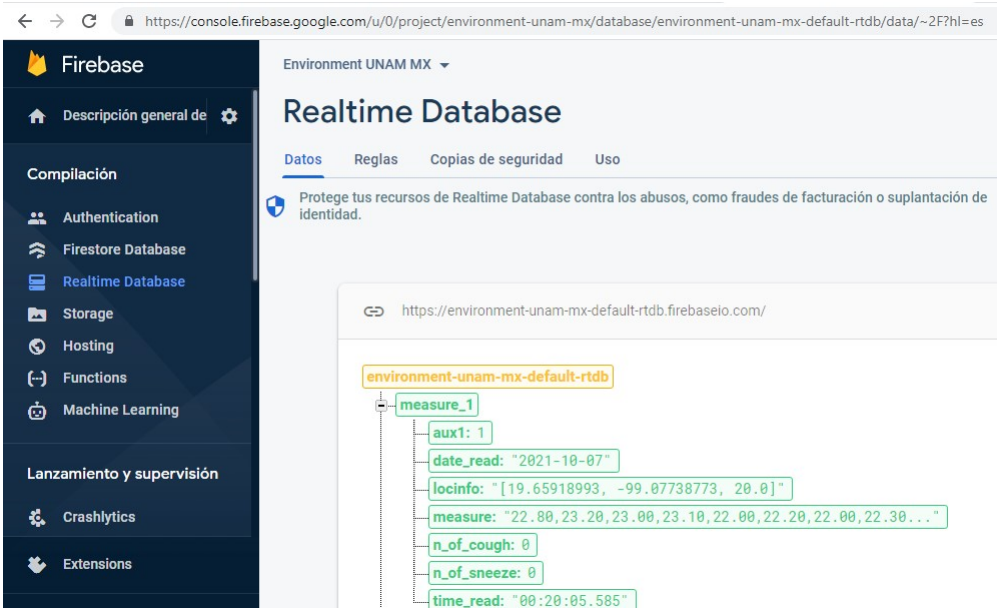


Figura 6.1: Visualización interactiva de una medición dentro de la base de datos Realtime Database

- Llave $aux1$: Variable cuya rol es ser un campo que en el futuro puede ser utilizado con el objetivo de añadir funcionalidades extras al dispositivo. Su valor default es 1.
- Llave $date\_read$: Variable que guarda el año, mes y día en que se realiza la medición
- Llave $locinfo$: El valor es una lista de tres elementos que hacen referencia la ubicación. Esta ubicación es obtenida ya sea por el GPS del dispositivo o por la Red a la cuál esté conectada. Se tomará la de mayor exactitud posible. Comprende latitud, longitud y exactitud en metros.
- Llave $measure$: Es una variable de tipo cadena de texto (string) de 19 elementos separados por comas. Serán las mediciones de la matriz de temperaturas, tomando los 16 valores de la matriz de 4x4, más uno adicional de temperatura de referencia interna del sensor. Los dos valores restantes serán la distancia en centímetros conseguida por el sensor ultrasónico y finalmente, un valor binario de 0 o 1 que indicará la etiqueta de esta medición. Un 1 significará contacto y un 0 significará no contacto.
- Llave n_of_cough : Cantidad de personas que tosieron hasta el momento de la medición.

- Llave *n_of_sneeze* : Cantidad de personas que estornudaron hasta el momento de la medición.
- Llave *time_read* : Hora del día a la cual fue tomada la medición.

Las mediciones en objeto JSON al momento de ser almacenadas en la base de datos, tendrán la estructura siguiente:

```
"measure_1" :{
  "aux1" :1,
  "date_read" :"2021-10-07",
  "locinfo" :"[19.65918993, -99.07738773, 20.0]",
  "measure" :"22.80,23.20,23.00,23.10,22.00,22.20,22.00,22.30,22.
     20,22.80,22.10,22.20,22.00,22.30,22.20,22.30,27.70,0,138,\n
     ",
  "n_of_cough" :0,
  "n_of_sneeze" :0,
  "time_read" :"00:20:05.585"
},
"measure_10" :{
  "aux1" :1,
  "date_read" :"2021-10-07",
  "locinfo" :"[19.65919553, -99.07742789, 300.0]",
  "measure" :"22.70,22.10,21.90,21.90,22.60,21.90,21.70,21.80,23.
     00,22.30,22.20,21.70,22.80,22.20,21.90,21.80,26.00,0,163,\n
     ",
  "n_of_cough" :0,
  "n_of_sneeze" :0,
  "time_read" :"00:20:43.588"
},
"measure_100" :{
```

```
20      "aux1" :1,
21      "date_read" :"2021-10-07",
22      "locinfo" :"[19.65918013, -99.07749139, 20.0]",
23      "measure" :"29.50,34.20,35.40,30.20,33.70,34.70,35.10,32.20,34.
            40,35.40,33.60,30.80,33.00,34.80,33.90,30.50,27.30,1,12,\n"
            ,
24      "n_of_cough" :0,
25      "n_of_sneeze" :0,
26      "time_read" :"00:24:30.735"
27    },
```

6.5. Configuración de Base de Datos

Dentro de un ambiente de una solución que se encuentra en producción es necesario establecer ciertas reglas para la base de datos. Actualmente, con el objetivo de desarrollo de este dispositivo, las configuraciones se encuentran en un estado que permite ensayos rápidos, es decir, es flexible ante cambios que se requieran para poder probar distintos conceptos.

6.5.1. Reglas

Las reglas de una base de datos realtime database se escriben de igual forma en formato JSON. El dispositivo se conecta a esta base de datos en modo test para poder ejecutar procesos de lectura y escritura. En una primera versión generada de default se tenía una configuración para que el 14 de abril de 2021 se vencieran los permisos de lectura y escritura. Manualmente han sido modificados para que hasta el año 2052 se puedan hacer estas operaciones.

```
1  {
2    "rules": {
3      ".read": "now < 2618376400000", // 2021-4-14
4      ".write": "now < 2618376400000", // 2021-4-14
```

```
5   }
6 }
7 // Allow read/write access to all users under any conditions
8 // Warning: **NEVER** use this rule set in production; it allows
9 // anyone to overwrite your entire database.
```

Es importante hacer énfasis que para un ambiente de producción es necesario establecer reglas determinadas para que los usuarios hagan operaciones de lectura y escritura solo bajo ciertas condiciones. De otra forma, las reglas actuales permiten al usuario realizar todos los cambios que desee, escenario a evitar en un estado operativo de la solución. Estas configuraciones sirven perfectamente para las pruebas realizadas con el dispositivo y validar que todas las operaciones de lectura y escritura se estén realizando correctamente.

6.5.2. Configuración de google-services.json

Este archivo es necesario para que Firebase se pueda agregar a una app Android. El documento json de configuración llamado google-services.json permite una integración directa al proyecto en Android Studio. El archivo contiene información y valores secretos para una vinculación exitosa. Dada la información sensible presentada en el archivo mencionado, es poco viable exponerlo en el presente documento.

Capítulo 7

Análisis Exploratorio de Datos

7.1. Introducción

En este capítulo se aborda el análisis de los primeros ensayos del sensor de temperatura junto con el de distancia. Posteriormente se muestra un análisis exploratorio de datos, técnica de ciencia de datos, a un conjunto de información almacenada en la base de datos de Firebase producto de toma de muestras etiquetadas.

7.2. De los conjuntos de datos

Este análisis estará enfocado principalmente en tres conjuntos de datos.

- Conjunto de datos de testeo al sensor OMRON D6T y sensor ultrasónico HC-SR04 en condiciones controladas.
- Conjunto de datos de adquisición de mediciones etiquetadas con "Sí Contacto" y "No Contacto" con el objetivo de utilizarse en aprendizaje máquina supervisado.
- Conjunto de datos para mostrar localizaciones en mediciones.

7.3. Condiciones controladas: OMRON D6T y HC-SR04

Estos datos surgen de las primera aproximaciones a los sensores para comprender su funcionamiento y entendimiento de estos mismos como sistemas de manera aislada.

7.3.1. De la adquisición de datos

Estando los sensores empotrados en la pared se colocarón a una altura de 1.50 metros. Esta altura se determinó pues el sujeto de pruebas principal fue el autor del documento Guillermo Martínez, cuya estatura es de 1.82 metros. De tal forma, 1.50 metros corresponde a la posición de su esternón, lugar aproximado donde el dispositivo será usado. Se usó la configuración mostrada en la Figura 7.1.

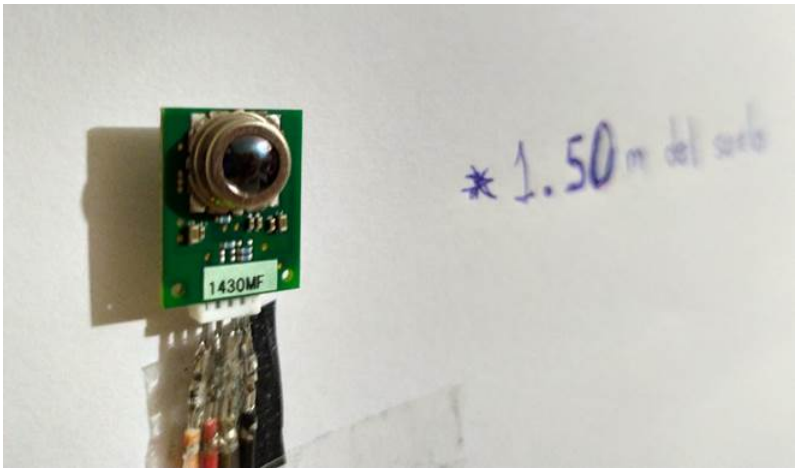


Figura 7.1: Configuración inicial de las pruebas (Martínez-Martínez, 2021)

Desde esta posición el sensor de temperatura se comunica con la plataforma de desarrollo Arduino ubicada en una protoboard, esto es ilustrado en la Figura 7.2.

El lugar de pruebas y de adquisición local de pruebas queda configurado como se muestra en las figuras 7.4 y 7.3.

Al cabo de unas primeras mediciones fue posible ver que el sensor contaba con una rotación no deseada que era causada por una débil sujeción a la pared, que

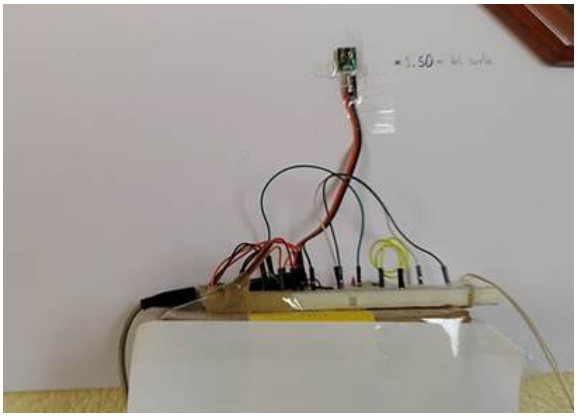


Figura 7.2: Conexión al circuito (Martínez-Martínez, 2021)

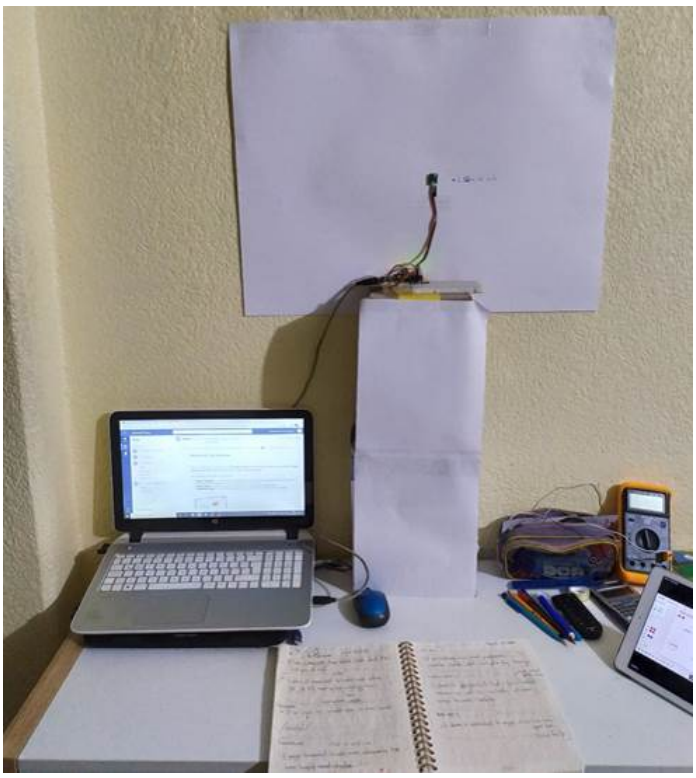


Figura 7.3: Módulo de adquisición de datos (Martínez-Martínez, 2021)

hace las veces de tierra mecánica en esta configuración. En ese sentido, se decidió abordar un cambio a esta configuración de tal manera que el sensor quedara sujeto mediante un soporte que tuviera tres grados de libertad: Rotación en los 3 ejes. Se solicitó manufacturar un elemento de sujeción con un proveedor de manufactura

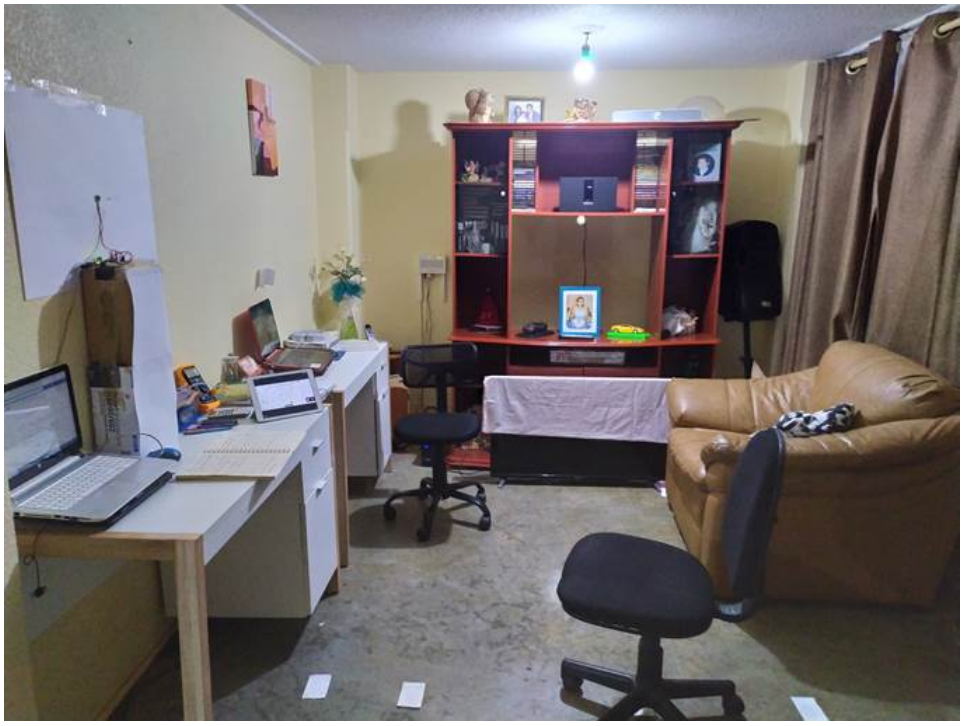


Figura 7.4: Entorno completo de primera adquisión de datos (Martínez-Martínez, 2021)

aditiva (36), mostrado en la Figura 7.5.

Quedando el sensor empotrado en la pared como es mostrado en la Figura 7.6.

Bajo estas condiciones que fue posible realizar la toma de mediciones exploratorias. Finalmente, el sensor ultrasónico se colocó sobre del sujeto de pruebas, en este caso, el autor del presente trabajo Guillermo Martínez para que el sensor tuviese una pared completa como elemento de soporte al rebote de ondas ultrasónicas.

De esta forma, para diversos experimentos fue posible tomar mediciones con diversas combinaciones de variables, ángulos en los tres posibles ejes y distancia del individuo al sensor.

Las mediciones sobre las que mostrarán los análisis cuentan con la siguiente configuración:

- Ángulos en 0 [°]
- Sensor de temperatura a 1.5 [m] del suelo

Figura 7.5: Elemento de sujeción de tres grados de libertad realizado con manufactura aditiva (Romero-Rivas, 2021)

Figura 7.6: Sensor empotrado en la pared con el elemento de sujeción de tres grados de libertad (Martínez-Martínez, 2021)

- Sensor de distancia colocado a la altura del esternón del individuo de prue-

bas, equivalente a 1.5 [m]

- La distancia será entre el individuo y el sensor.

La toma de mediciones se realizó a través de un script de Python 3.7 cuyo código se encuentra en los Apéndices del presente trabajo. Asimismo, el código utilizado para el funcionamiento del dispositivo en el microcontrolador Arduino también se encuentra en los Apéndices de este documento.

7.3.2. Análisis: Matrices de temperatura

Dado que se cuenta con una matriz de 16 elementos de temperatura, una representación de mapa de calor es adecuada para tener una interpretación gráfica del comportamiento de las temperaturas. De tal forma, se generaron animaciones GIF de aproximadamente un minuto de duración donde el mapa de calor muestra cambios de color mientras la distancia varía. El código que genera estas animaciones de igual manera se encuentra en la sección de Apéndices. Un ejemplo de un frame de esta animación se muestra en la Figura 7.7

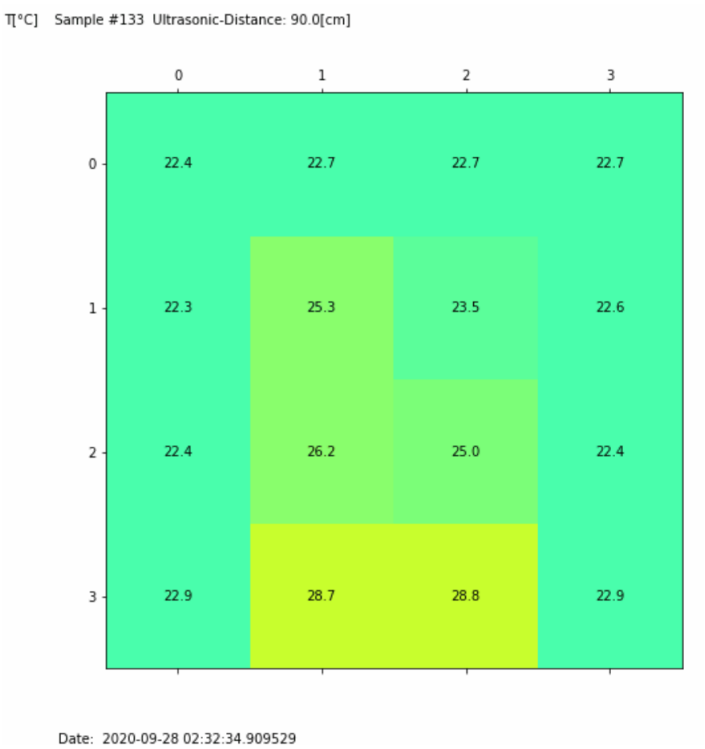


Figura 7.7: Frame de animación de mediciones (Martínez-Martínez, 2021)

El caso en particular mostrado en la Figura 7.7 corresponde a una medición realizada el 28 de septiembre de 2020 a las 02:32 hrs, siendo la muestra 133 de la animación. La distancia en ese momento es de 90 [cm].

Esta representación es útil para comprender el comportamiento de la matriz de temperaturas cuando se encuentra con un individuo de frente a una distancia variable. A continuación se mostrarán tres muestras con el respectivo mapa de calor en las Figuras 7.8 7.9 7.10.

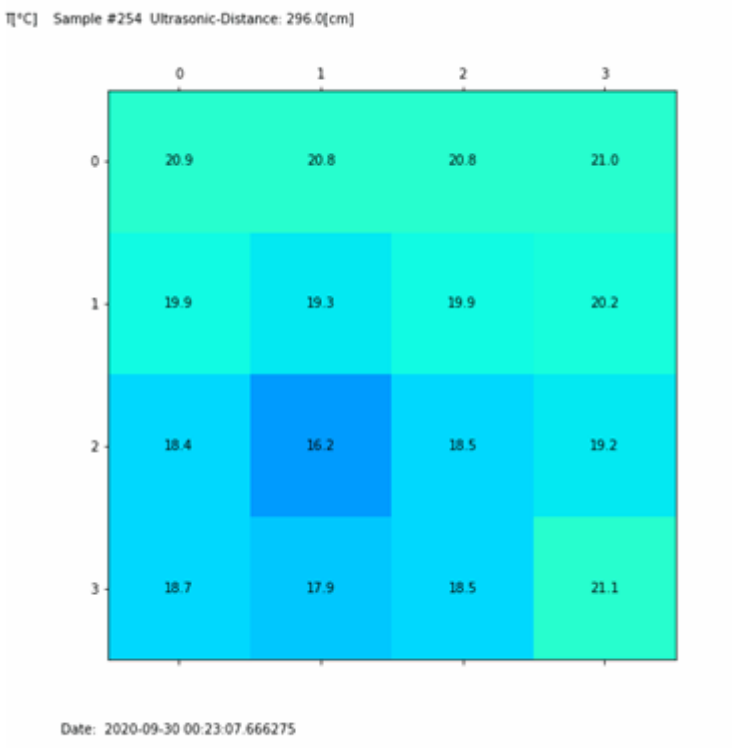


Figura 7.8: Frame de animación de mediciones: Caso alejado (Martínez-Martínez, 2021)

En la Figura 7.8 el individuo se encuentra alejado del sensor de temperatura pues está a 296 [cm], de esta manera, la mayoría de los elementos de esta matriz muestran temperaturas cercanas a la ambiente. Es perceptible también que los elementos superiores detectan cierta elevación de temperatura. Estos elementos apuntan a la región cercana de la cabeza del individuo.

Por otra parte, una vez que el individuo se acerca un poco más al sensor a una

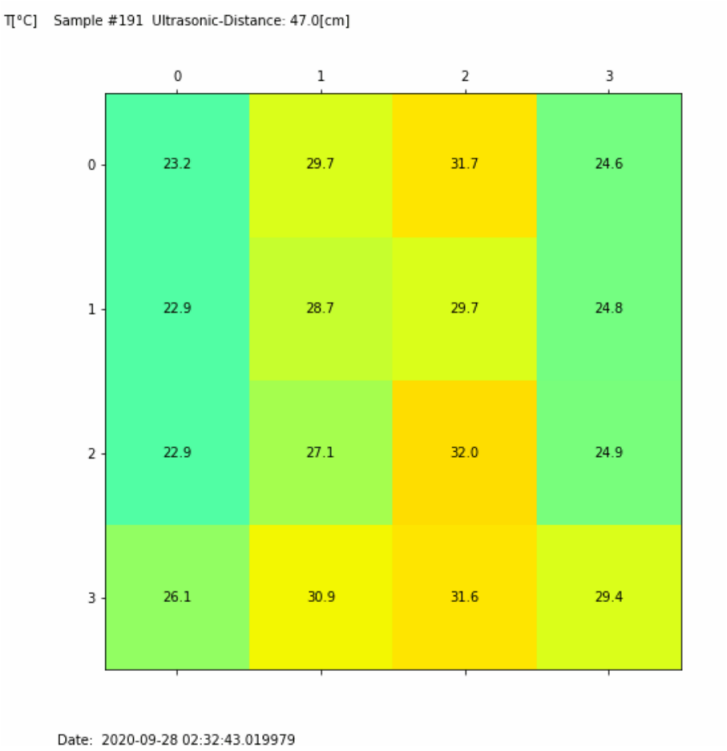


Figura 7.9: Frame de animación de mediciones: Caso distancia media (Martínez-Martínez, 2021)

distancia que podría considerarse un contacto bastante cercano (Figura 7.9), de 47 [cm], es definitivamente percetible la presencia de temperaturas significativamente más elevadas que la ambiental, pues se encuentran algunas mayores a 30 [°C].

Finalmente, ante un contacto totalmente cercano, el cual podría ser prácticamente un saludo de beso o abrazo, a una distancia de 5 [cm], la matriz de temperaturas muestra elevaciones bastante cercanas a la temperatura corporal de un individuo, son visibiles valores de 35 [°C]. Se muestra en la Figura 7.10.

7.3.3. Análisis: Comportamiento de valores de matrices de temperatura

Un análisis interesante en este caso de uso es conocer el comportamiento general de cada uno de estos elementos de la matriz durante la ejecución de estas medidas. Los segmentos de animación mostrados anteriormente corresponden a un conjunto de 300 mediciones.

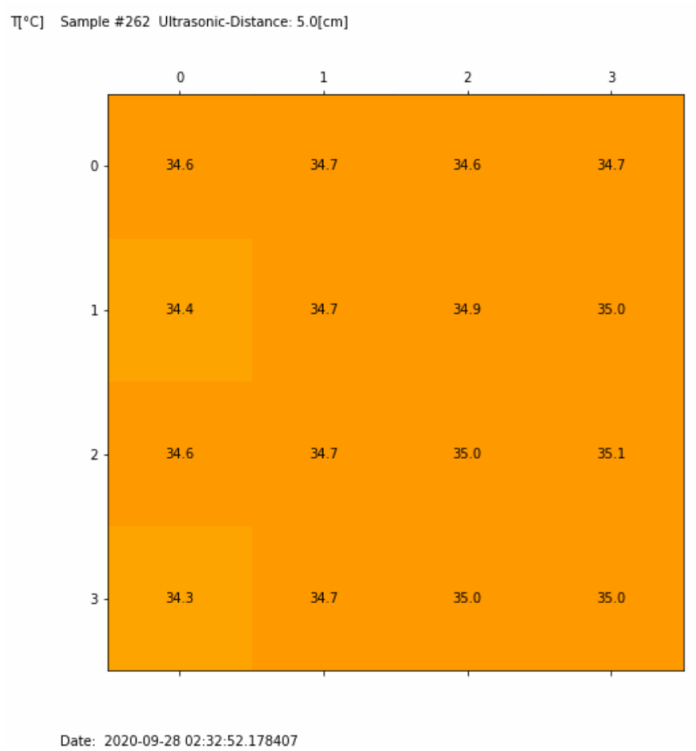


Figura 7.10: Frame de animación de mediciones: Caso cercano (Martínez-Martínez, 2021)

A continuación cada medición será representada por un punto dentro de una gráfica de Temperatura vs Distancia generada para cada uno de los elementos de la matriz. El eje de las abscisas será distancia en centímetros y el de las ordenadas temperatura en grados Celsius.

En las figuras 7.11 y 7.12 es perceptible un comportamiento para cada elemento de la matriz. El valor de la temperatura es inversamente proporcional al valor de la distancia. Esto marca un primer patrón que indicaría la presencia de un contacto o no contacto.

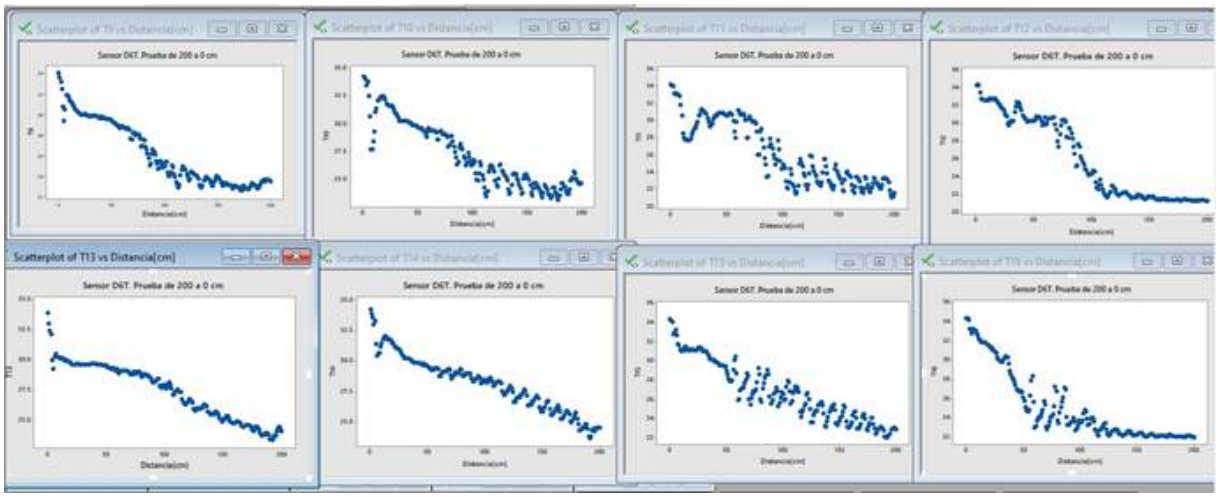


Figura 7.11: Primeros ocho elementos de la matriz (Martínez-Martínez, 2021)

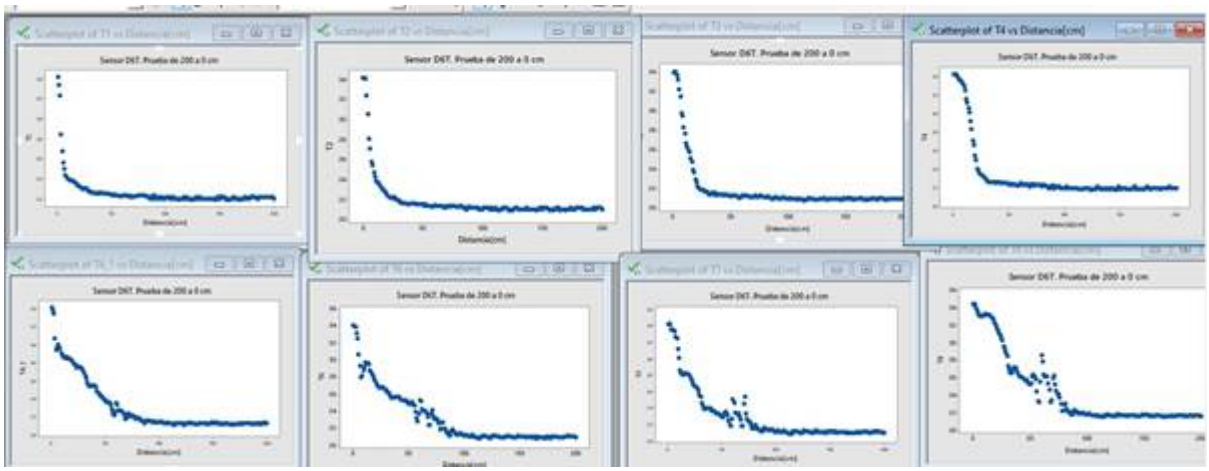


Figura 7.12: Útimos ocho elementos de la matriz (Martínez-Martínez, 2021)

7.4. Mediciones etiquetadas para aprendizaje máquina supervisado

El objetivo de estas mediciones es distinto al de la Sección 7.3, en aquella sección el objeto principal es conocer el comportamiento de la toma de mediciones de los sensores y dar el primer acercamiento. En esta sección el objetivo es, una vez con el dispositivo funcional, tomar mediciones que se encuentren etiquetadas

de con un “0” si corresponde a No contacto y con un “1” en caso de Sí contacto. Tener etiquetas permite ejecutar un subconjunto la Inteligencia Artificial llamado Aprendizaje Máquina Supervisado.

7.4.1. De la adquisición de datos

El dispositivo es colocado sobre el individuo que ejecuta las mediciones de la forma en que es representado en la Figura 7.13. Una vez teniendo el dispositivo,

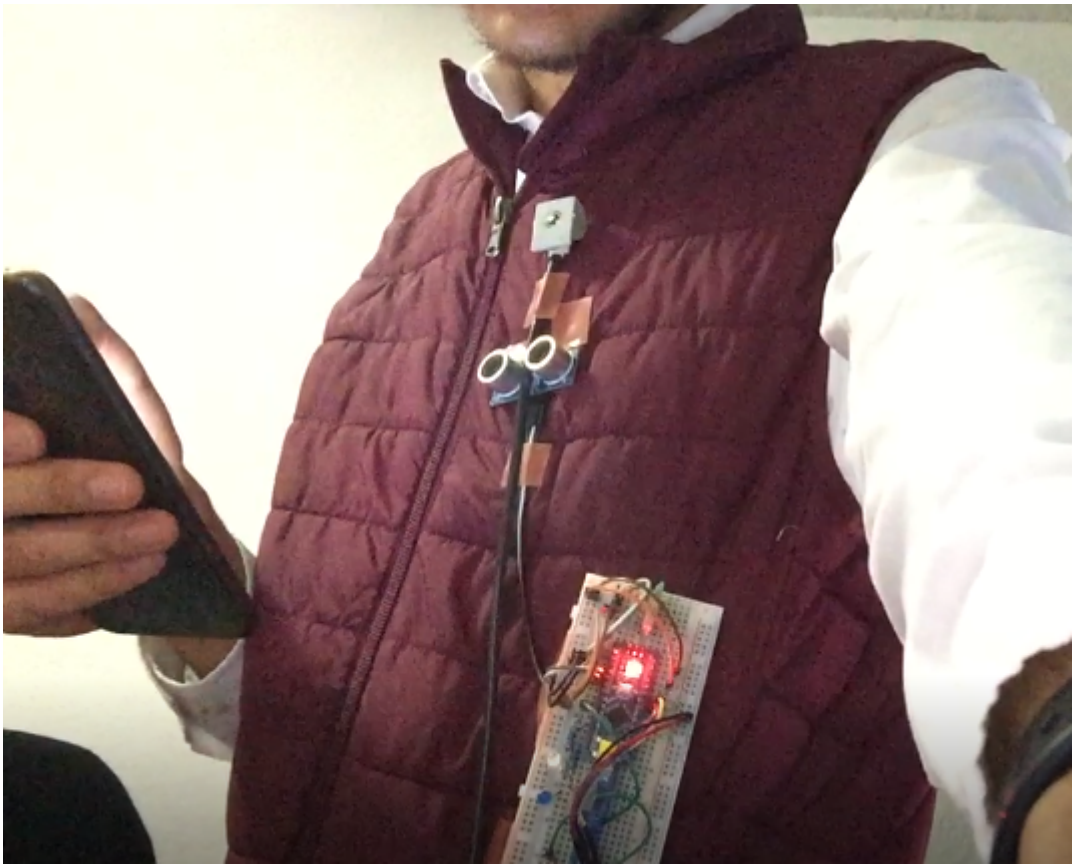


Figura 7.13: Posición del dispositivo en el individuo de pruebas (Martínez-Martínez, 2021)

el individuo camina y toma mediciones etiquetadas, es decir, ante un no contacto mediciones son enviadas a la base de datos Firebase. De la misma manera mediciones para el caso de sí contactos son sincronizadas con la base de datos.

7.4.2. Etiquetado de datos

El criterio utilizado para generar este etiquetado de mediciones es el siguiente: Una medición es etiquetada como un contacto cuando la distancia física con otro individuo es menor a 1.5 [m]. Todos aquellas mediciones que no se encuentran bajo estos criterios, son etiquetadas como no contactos.

7.4.3. De la base de datos Firebase al ambiente local de Python

A través de un código en Python los datos son obtenidos de esta base de datos y localmente son pasados por un proceso de transformación de objetos JSON a una forma tabular, que en Python a través de la librería Pandas es llamada "DataFrame". El código para ello se encuentra en los apéndices de este documento.

7.4.4. Comportamiento de variables: Resultados y discusión

A continuación se mostrará un análisis descriptivo de este conjunto de datos. Se estarán analizando 323 mediciones, en las cuales se encuentran etiquetas de "Contacto" representadas con un 1 y "No contacto" con un 0.

En la Figura 7.14 se observa que las mediciones de cero, es decir, que representan un "No contacto" representan la mayor parte de los datos.

Es importante mencionar que las temperaturas representan un papel relevante en las variables que ingresan al modelo.

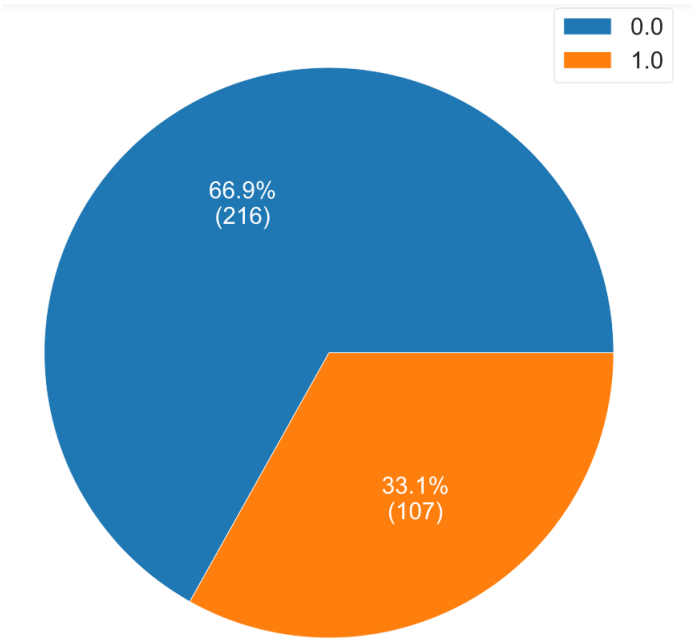
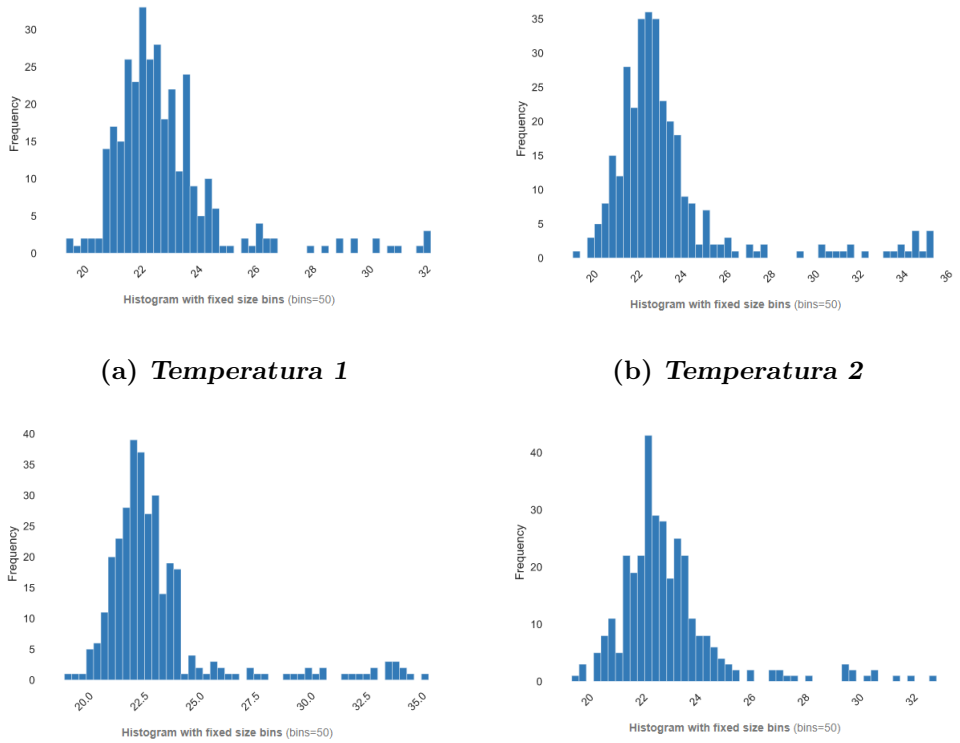


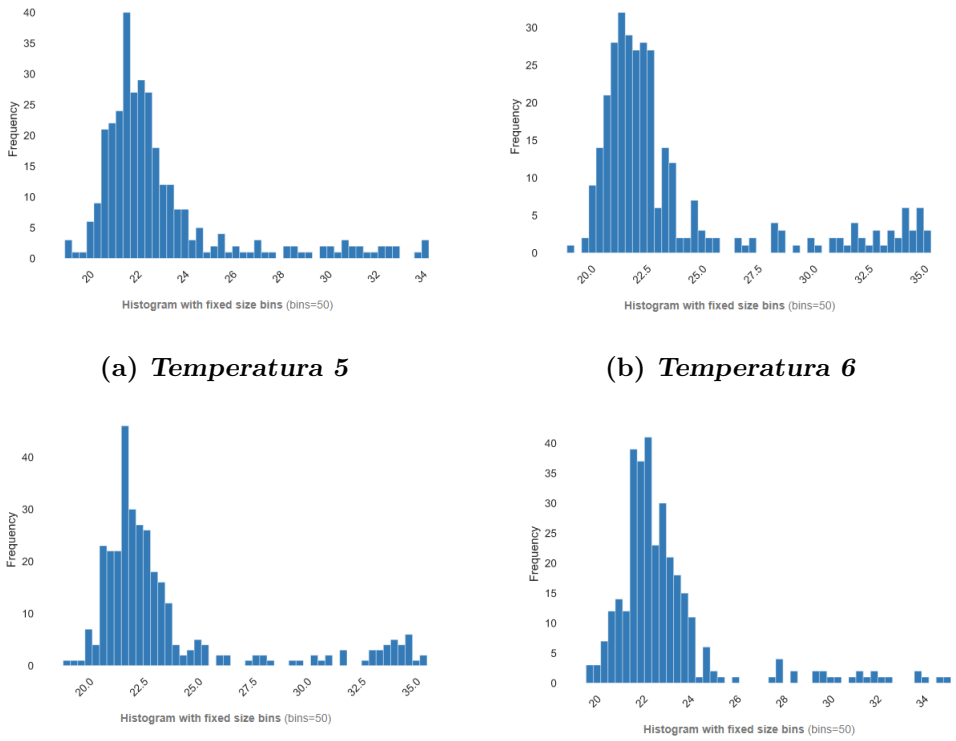
Figura 7.14: Distribución de mediciones etiquetadas (Martínez-Martínez, 2021)



(a) ***Temperatura 1*** (b) ***Temperatura 2***

(c) ***Temperatura 3*** (d) ***Temperatura 4***

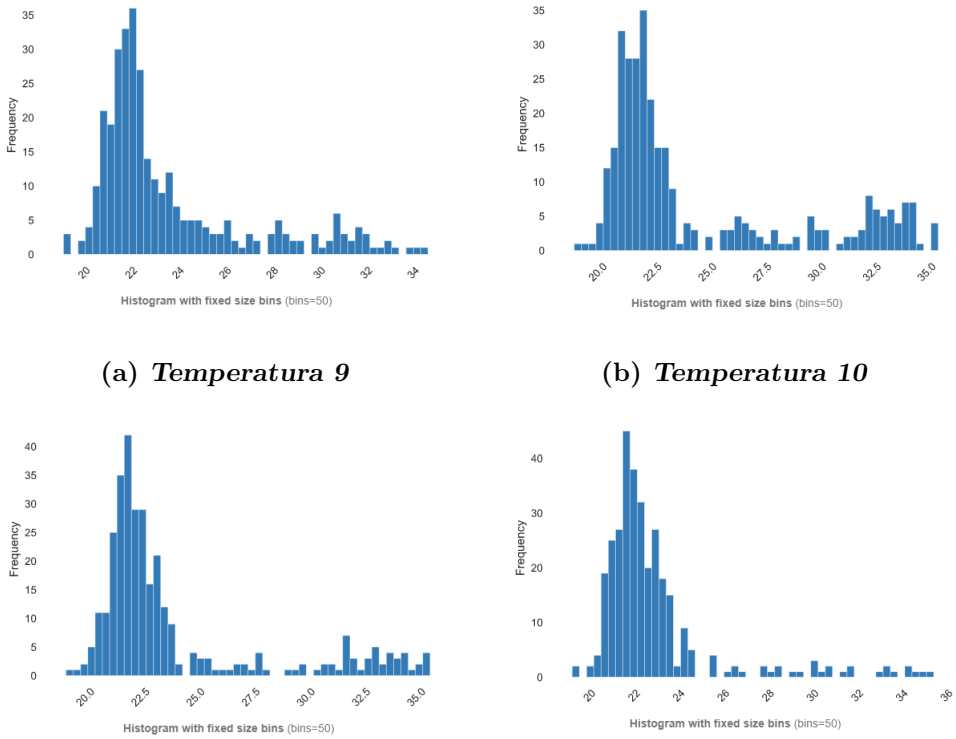
Figura 7.15: Histograma de distribución de Temperaturas 1, 2, 3 y 4. Abscisas representan temperatura [°C] y ordenadas la frecuencia de estas



(a) ***Temperatura 5*** (b) ***Temperatura 6***

(c) ***Temperatura 7*** (d) ***Temperatura 8***

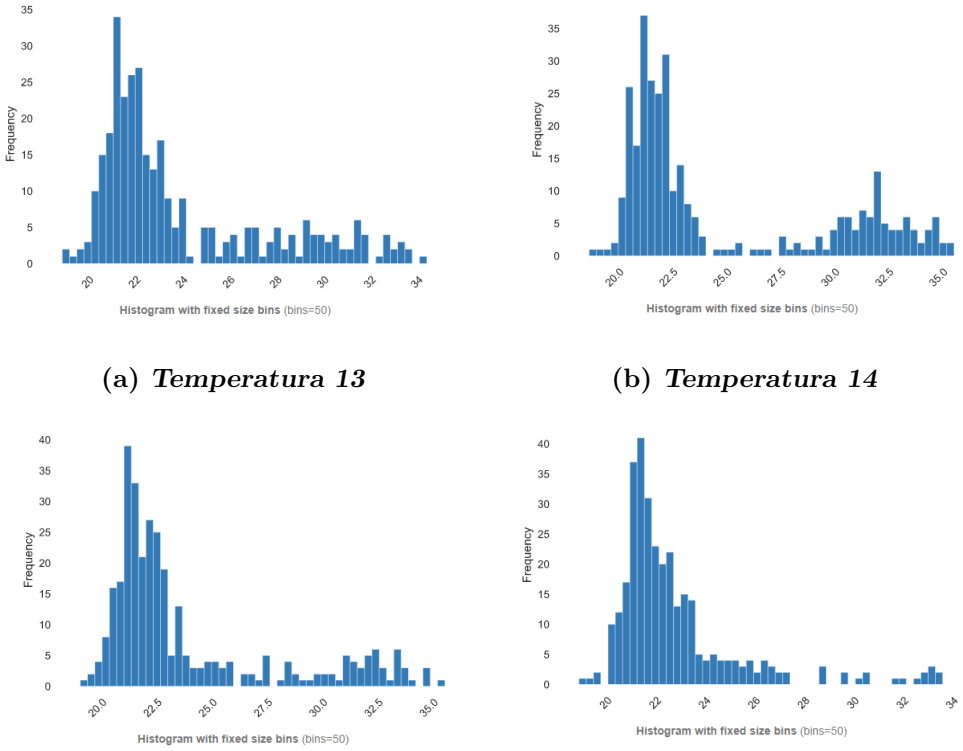
Figura 7.16: Histograma de distribución de Temperaturas 5, 6, 7 y 8 Abscisas representan temperatura [°C] y ordenadas la frecuencia de estas



(a) ***Temperatura 9*** **(b)** ***Temperatura 10***

(c) ***Temperatura 11*** **(d)** ***Temperatura 12***

Figura 7.17: Histograma de distribución de Temperaturas 9, 10, 11 y 12. Abscisas representan temperatura [°C] y ordenadas la frecuencia de estas



(a) ***Temperatura 13*** **(b)** ***Temperatura 14***

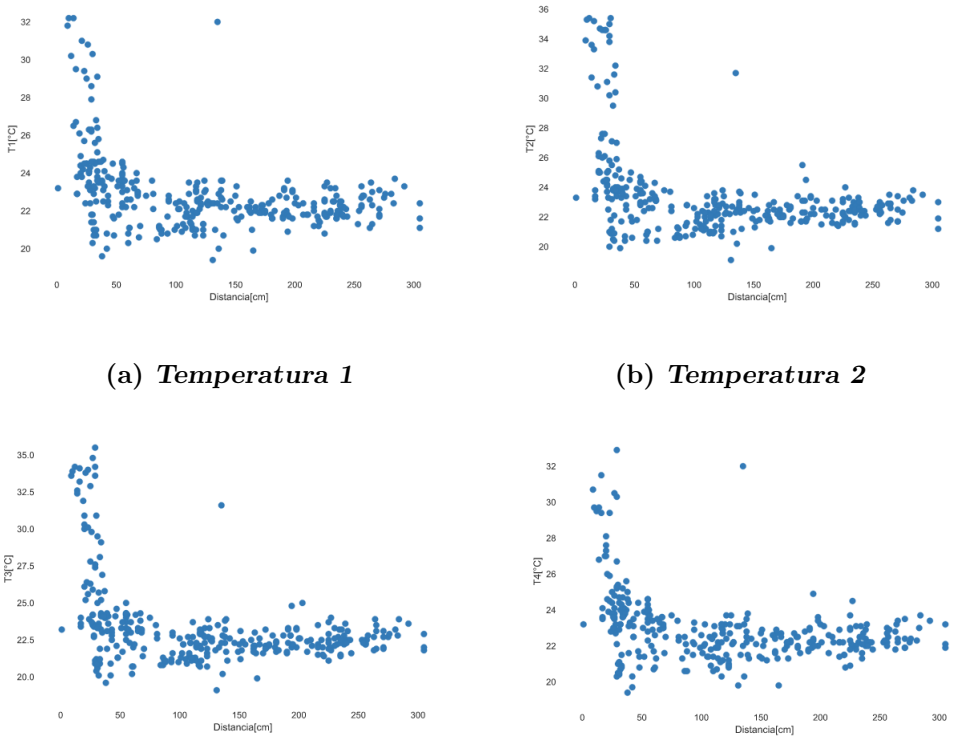
(c) ***Temperatura 15*** **(d)** ***Temperatura 16***

Figura 7.18: Histograma de distribución de Temperaturas 13, 14, 15 y 16. Abscisas representan temperatura [°C] y ordenadas la frecuencia de estas

Se enumeran los histogramas de distribución de temperaturas de los dieciséis elementos de la matriz, formados de la siguiente forma.

- Primer renglón: Figuras 7.15(a), 7.15(b), 7.15(c), 7.15(d)
- Segundo renglón: Figuras 7.16(a), 7.16(b), 7.16(c), 7.16(d)
- Tercer renglón: Figuras 7.17(a), 7.17(b), 7.17(c), 7.17(d)
- Cuarto renglón: Figuras 7.18(a), 7.18(b), 7.18(c), 7.18(d)

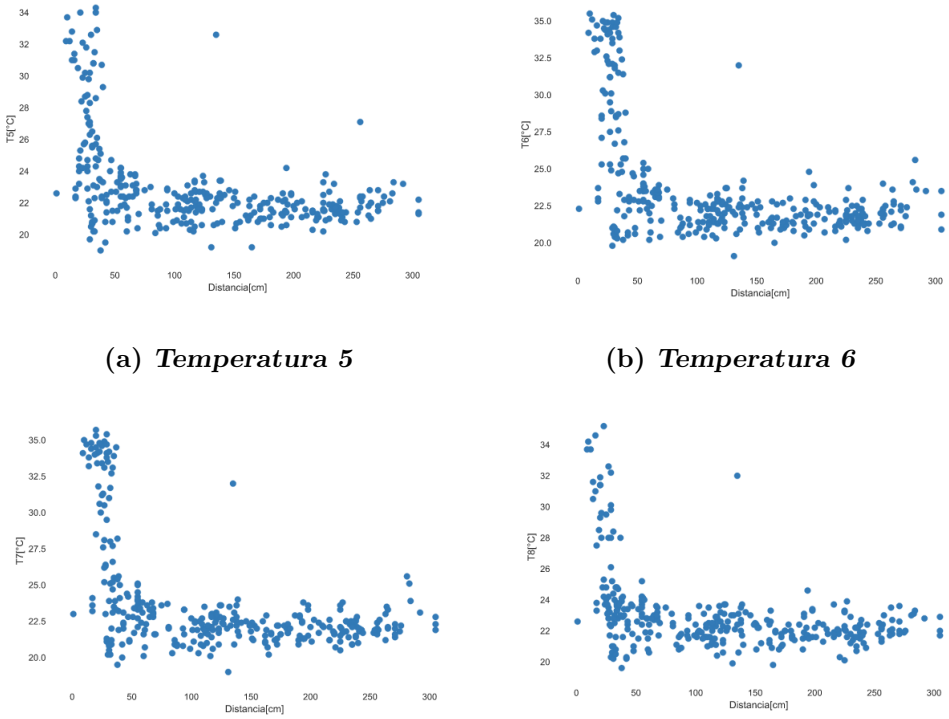
Es posible notar que las temperaturas cercanas a la ambiente de 22 [°C] forman el conjunto más grande en el histograma. La totalidad de elementos de la matriz mencionada muestran algunos otros datos en el conjunto cercano que sobrepasa los 30 [°C] explicados por aquellas mediciones de contactos cuya distancia es relativamente cercana. Este efecto de distancia será analizado en las gráficas de Temperatura vs Distancia.



(a) ***Temperatura 1*** **(b)** ***Temperatura 2***

(c) ***Temperatura 3*** **(d)** ***Temperatura 4***

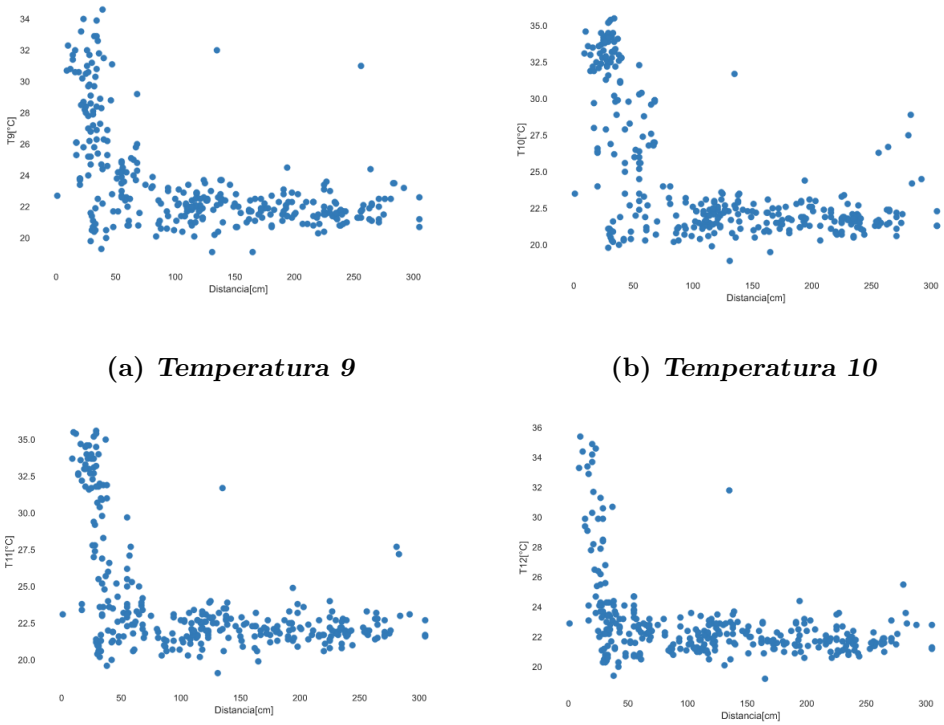
Figura 7.19: Temperaturas 1, 2, 3 y 4 [°C] vs Distancia [cm]



(a) ***Temperatura 5*** (b) ***Temperatura 6***

(c) ***Temperatura 7*** (d) ***Temperatura 8***

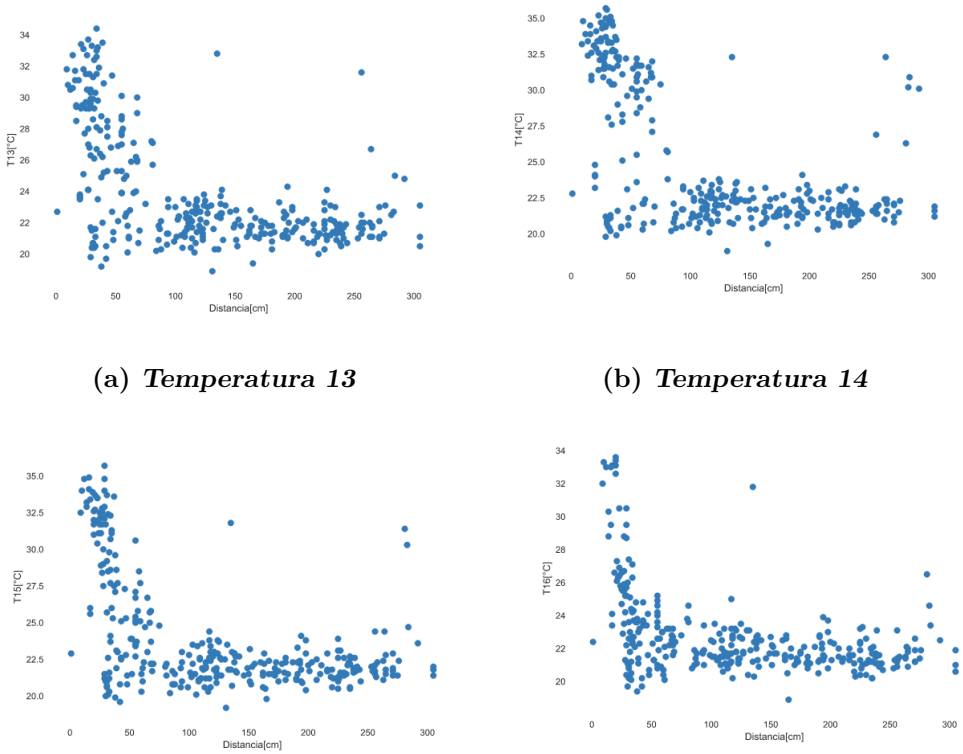
Figura 7.20: Temperaturas 5, 6, 7 y 8 [°C] vs Distancia [cm]



(a) ***Temperatura 9*** **(b)** ***Temperatura 10***

(c) ***Temperatura 11*** **(d)** ***Temperatura 12***

Figura 7.21: Temperaturas 9, 10, 11, 12[°C] vs Distancia [cm]



(a) ***Temperatura 13***

(b) ***Temperatura 14***

(c) ***Temperatura 15***

(d) ***Temperatura 16***

Figura 7.22: Temperaturas 13, 14, 15 y 16 [°C] vs Distancia [cm]

Se enumeran las figuras de temperaturas vs distancia de los dieciséis elementos de la matriz a continuación:

- Primer renglón: Figuras 7.19(a), 7.19(b), 7.19(c), 7.19(d)
- Segundo renglón: Figuras 7.20(a), 7.20(b), 7.20(c), 7.20(d)
- Tercer renglón: Figuras 7.21(a), 7.21(b), 7.21(c), 7.21(d)
- Cuarto renglón: Figuras 7.22(a), 7.22(b), 7.22(c), 7.22(d)

En todas las gráficas la tendencia es clara: Temperatura y Distancia son inversamente proporcionales.

Lo anterior debido a que el sensor de temperatura es capaz de tomar la medición con mayor exactitud cuando la distancia es menor.

Adicionalmente, es posible ver algunos valores atípicos cuya naturaleza no se alinea con la tendencia general. Se consideran atípicos debido al ruido en mediciones y son particularmente conocidos como outliers del dataset.

7.4.5. Patrones en las variables

Una herramienta para encontrar patrones en los datos suele ser la matriz de correlaciones de Pearson. De acuerdo con el Capítulo 6 de "Selection of Variables and Factor Derivation" por Nettleton, David, la correlación de Pearson es el método más común para usar con variables numéricas; asigna un valor entre -1 y 1, donde 0 es no correlación, 1 es una correlación total positiva y -1 es una correlación total negativa. Es interpretado de la siguiente forma: un valor de correlación de 0.7 entre dos variables indicaría que existe una relación positiva y significativa. Una correlación positiva significa que si la variable A sube, entonces B también subirá, mientras que si el valor de la correlación es negativa, entonces si A crece, B se reduce (29).

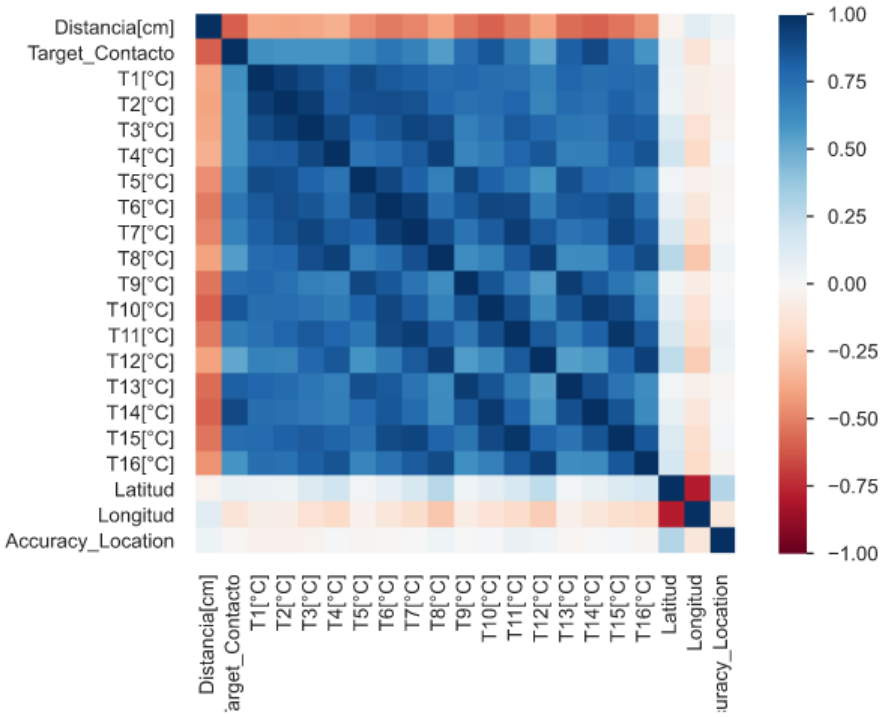


Figura 7.23: Matriz de Pearson para mediciones (Martínez-Martínez, 2021)

La Figura 7.23 de Matriz de Pearson para las mediciones evidencia tanto en la primera columna así como en el primer renglón completos, el comportamiento inversamente proporcional de la temperatura con la distancia, lo cual confirma lo interpretado líneas anteriores en el presente documento.

7.5. Datos de localización

Este conjunto de datos corresponde a mediciones con el dispositivo realizadas dentro de un supermercado en Guadalajara, Jalisco, México. Se mostrará el comportamiento de la ubicación de las mediciones.

7.5.1. De la adquisición de datos

Estando en el supermercado antes mencionado, se recolectó un conjunto de mediciones. Se solicitó de manera verbal autorización al gerente en turno para poder utilizar el dispositivo en sus instalaciones. De este dataset se recuperan las varibles de posicion geográfica para analizar su comportamiento. El instrumento de medición siempre es autodeterminado por la aplicación. Como criterio de decisión de tomar GPS o Red Celular, la app decanta su decisión por el cual tenga mayor exactitud en el momento de la solicitud de ubicación.

7.5.2. Comportamiento de variables geográficas

El dataset es tratado a manera de DataFrame con el módulo de Pandas para Python. La información de Latitud y Longitud es pasado a una forma de puntos que se almacenan en una columna de geometría tal cual es mostrado en la Tabla 7.24

Posteriormente, para poder graficar la información con herramienta geopandas en python es necesario contar con archivos extensión .shp (shape). En este caso: Jalisco, México para la ciudad de Guadalajara, los archivos se consiguieron del University of California Berkeley GeoData Repository (8).

Con lo anterior cubierto, es posible mapear en la figura 7.25 cada una de estas mediciones sobre un mapa. Dada la cercanía geográfica de todas las mediciones dentro del estado de Jalisco se observan todos los datos en la ciudad de Guadalajara.

	Date_Measure	Time_Measure	Latitude	Longitude	geometry	measure
0	2021-09-26	20:12:33.099	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
1	2021-09-26	20:12:39.448	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
2	2021-09-26	20:12:41.868	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
3	2021-09-26	20:12:45.485	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
4	2021-09-26	20:13:00.107	20.675603	-103.391042	POINT (-103.39104 20.67560)	Measurement
...	...	...	...	...	...	...
66	2021-09-26	20:23:04.425	20.674617	-103.394292	POINT (-103.39429 20.67462)	Measurement
67	2021-09-26	20:23:16.890	20.674617	-103.394292	POINT (-103.39429 20.67462)	Measurement
68	2021-09-26	20:23:37.080	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
69	2021-09-26	20:12:21.494	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement
70	2021-09-26	20:12:23.984	20.669061	-103.390681	POINT (-103.39068 20.66906)	Measurement

Figura 7.24: Información de manera tabular (Martínez-Martínez, 2021)

Figura 7.25: Ubicación geográfica de mediciones en la ciudad de Guadalajara, Jalisco, México. (Martínez-Martínez, 2021)

Capítulo 8

Inteligencia Artificial y Modelos de Machine Learning: Resultados y Discusión

8.1. Introducción

En este capítulo se presenta la aplicación de un Modelo de Machine Learning, un subconjunto de Inteligencia Artificial, al caso de uso del dispositivo. Se explica la base matemática del modelo utilizado así como específicos adicionales del proceso y la comparación del desempeño de este contra otros potenciales. Adicionalmente, se muestran los resultados y la discusión de su signficado.

8.2. Inteligencia Artificial

John McCarthy, científico de computación y cognitivo, fue uno de los fundadores de la Inteligencia Artificial, él en 2004 en el Departamento de Ciencias de Computación de la Universidad de Stanford, California, EUA, propone la siguiente definición:

" It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." [Es la ciencia e ingeniería de hacer máquinas inteligentes, especialmente, programas de computadora. Está relacio-

nado con la tarea similar de utilizar computadoras para entender la inteligencia humana, pero la Inteligencia Artificial no tiene que limitarse a sí misma a métodos biológicamente observables] (McCarthy, 2004)

IBM, por su parte, define a IA como:

"Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind."[La Inteligencia Artificial aprovecha los ordenadores y las máquinas para imitar la capacidad de resolución de problemas y la toma de decisiones de la mente humana] (14)

Gartner en su glosario en línea propone la siguiente definición:

'Artificial intelligence (AI) applies advanced analysis and logic-based techniques, including machine learning, to interpret events, support and automate decisions, and take actions'"[La Inteligencia Artificial aplica análisis avanzado y técnicas basadas en lógica, incluyendo al aprendizaje máquina, para interpretar eventos, apoyar y automatizar decisiones y tomar accines.](16)

8.3. Machine Learning

De acuerdo con IBM, Machine Learning, en español "Aprendizaje máquina" es una rama de la Inteligencia Artificial y de ciencia de computación la cual se enfoca en el uso de datos y algoritmos para imitar la forma en que los humanos aprenden, incrementando su exactitud gradualmente.(15)

En la Figura 8.1 IBM representó a Machine Learning como un subconjunto de la Inteligencia Artificial, junto con otro concepto determinado Deep Learning, el cual pertenece al universo de Machine Learning.

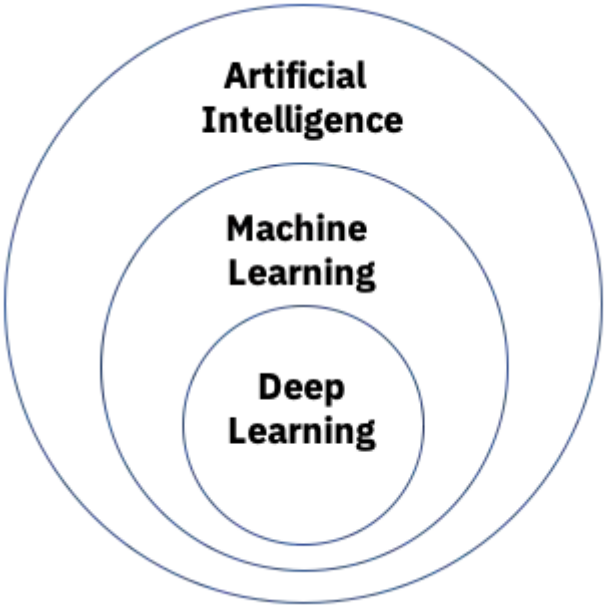


Figura 8.1: Machine Learning como un subconjunto de la Inteligencia Artificial (14)

8.3.1. Casos de uso de Machine Learning en el mundo real

De acuerdo con IBM (15) algunos casos de uso reales son los siguientes:

- Reconocimiento de voz
- Servicio al cliente
- Visión por computadora
- Motores de recomendación
- Trading automático de acciones

8.3.2. Métodos de Machine Learning

Muchos modelos de Machine Learning están definidos por la presencia o ausencia de influencia humana en los datos crudos, ya sea que una recompensa se ofrezca (al algoritmo), retroalimentación específica o etiquetas que se utilicen (26). De acuerdo con Nvidia.com (27) existen diferentes modelos de Machine Learning como los siguientes:

- Supervised Learning: En español, aprendizaje supervisado, se caracteriza por tener un conjunto de datos que ha sido pre-etiquetado y clasificado por usuarios que le permiten al algoritmo ver qué tan exacto es el desempeño.
- Unsupervised Learning: En español, aprendizaje no supervisado, se caracteriza por tener un conjunto de datos que no está etiquetado t un algortimo identifica patrones y relaciones dentro de los datos sin ayuda de los usuarios.
- Semisupervised Learning: En español, aprendizaje semisupervisado, se caracteriza por tener data estructurada y no estructurada, la cual guía al algortimo en su camino de hacer conclusiones independientes. La combinación de ambos tipos de datos en un conjunto de entrenamiento permite a los algoritmos de aprendizaje máquina aprender a etiquetar datos no etiquetados.
- Reinforcement learning: En español, aprendizaje por refuerzo, se utiliza un sistema de recompensa/castigo", ofreciendo retroalimentación al algoritmo para aprender las acciones que "deberealizar para obtener las recompensas.

8.3.3. Componentes de un algoritmo de aprendizaje supervisado

El caso de uso del dispositivo entra dentro del aprendizaje supervisado dado que el conjunto de datos se encuentra etiquetado. Para ilustrar el concepto de datos etiquetados se muestra la Figura 8.2 donde para cada imagen de entrada se tiene identificada la clasificación de perro/gato, mientras que para los no etiquetados, no se tiene la clasificación.

Figura 8.2: Ilustración de datos etiquetados y datos no etiquetados (9)

Los componentes típicos de un Algoritmo de Machine Learning Supervisado son (26):

- Proceso de decisión: Un conjunto de procedimientos, cálculos u otros pasos que toma los datos y regresa una predicción en el tipo de patrones de los datos que el algoritmo está tratando de encontrar.
- Función de error: Un método para medir qué tan buena fue la predicción a través de comparación con los ejemplos previamente conocidos (cuando estos están disponibles). Esta función se plantea la pregunta: ¿El proceso de decisión predijo correctamente? Si no fue así, ¿Cómo cuantificar qué tan mala fue la predicción?
- Un proceso de actualización u optimización: Es donde el algoritmo observa los errores y entonces actualiza el proceso de decisión mediante el cual el algortimo llega a la decisión final, de esta manera la siguiente vez el error será de menor magnitud.

8.4. El problema de clasificación

Definitivamente el problema central del uso del dispositivo es clasificar la información de las mediciones como Contacto o No contacto, en ese sentido, Machine Learning será utilizado para abordar el tema de *clasificación*. Particularmente será llamada clasificación binaria pues solo existen dos alternativas.

8.5. Regresión Logística

El modelo de Inteligencia Artificial, perteneciente al subconjunto de Machine Learning supervisado, que soporta el funcionamiento con datos obtenidos del dispositivo será *Regresión Logística*, o mejor conocido en la literatura como *Logistic Regression*.

8.5.1. Definición

IBM define a la regresión logística como un tipo de análisis estadístico (también conocido como logit model) que es usualmente utilizado para analíticos predictivos y modelado, y extiende aplicaciones en machine learning. En el enfoque de analíticos las variables dependientes son finitas o categóricas: ya sea A o B (regresión binaria) o un rango finito de opciones A, B, C o D (regresión multinomial). Es utilizado en software estadístico para entender la relación entre variables dependientes y una o más variables indepenientes a través de estimación de probabilidades utilizando una ecuación de regresión logística (25).

8.5.2. Del nombre de regresión logística

Dado que el problema de clasificación es binario, la variable dependiente "y" de "Contacto" tomará solamente valores entre 0 y 1, es decir, $y \in \{0, 1\}$. Intuitivamente, no tendría sentido que la variable tome valores mayores que 1 o menores que 0 (Ng, 2003).

De tal forma, seleccionaremos la función 8.1 para la salida de la variable "y", la cual será llamada "g" la cual es también conocida como función logística o función sigmoide. En la Figura 8.3 es posible visualizar el comportamiento de la función.

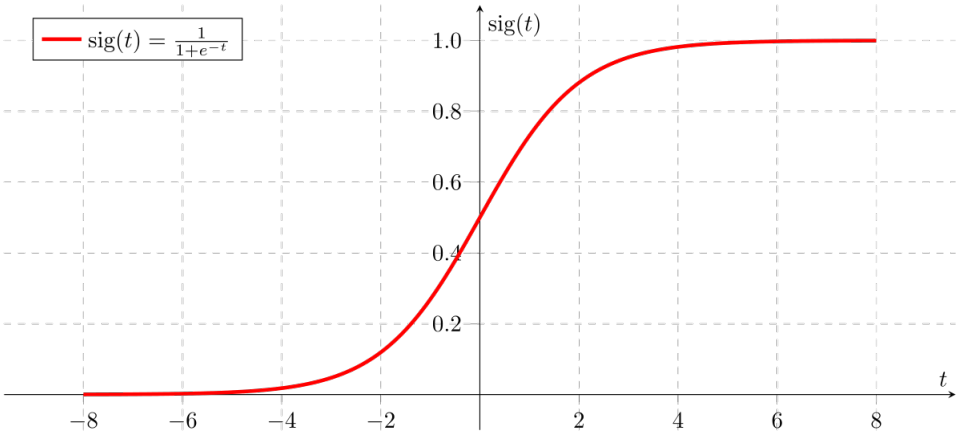


Figura 8.3: Función sigmoide (2).

$$g(z) = \frac{1}{1 + e^{-z}} \tag{8.1}$$

Es relevante notar que la función sigmoide $g(z)$ tiende a 1 cuando $z \to \infty$, y $g(z)$ tiende a 0 cuando $z \to -\infty$. Lo cual deja a $g(z)$ siempre entre los valores de 0 y 1 (Ng, 2003).

Un recurso bastante útil es la derivada de esta función, de tal manera se esboza en 8.2.

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1 + e^{-z}} \\ &= \frac{1}{(1 + e^{-z})^2}\left(e^{-z}\right) \\ &= \frac{1}{(1 + e^{-z})} \cdot \left(1 - \frac{1}{(1 + e^{-z})}\right) \\ &= g(z)(1 - g(z)). \end{aligned} \tag{8.2}$$

8.5.3. Del funcionamiento de la regresión logísitica

En la literatura las características que representan, en nuestro caso, mediciones, se les llama “features”. Estas características en el dispositivo son las temperaturas y la distancia. Dentro del funcionamiento de la regresión logística existen matrices de peso denotados por la letra “W” y valores de sesgo (de bias) con la letra “b”. Las características unidas crearán una matriz “X” que reúne estas características. Con σ se hará referencia a la función sigmoide presentada en 8.1.

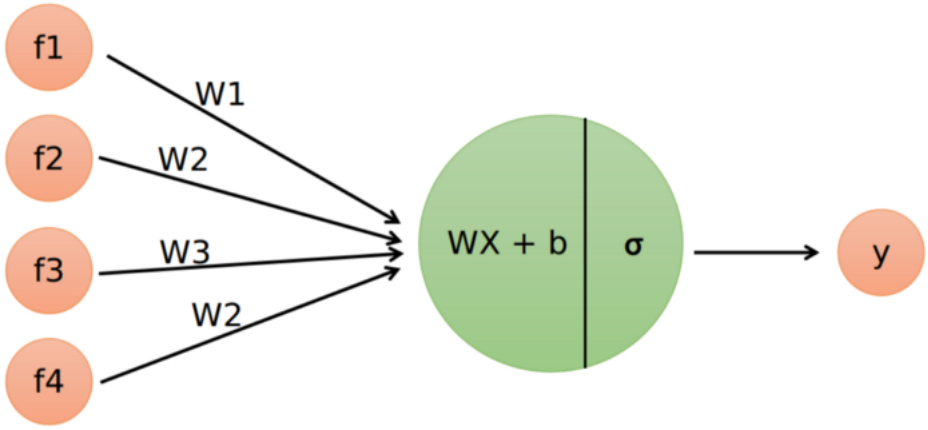


Figura 8.4: Modelo de flujo de funcionamiento de regresión logística (3).

El flujo del funcionamiento de la regresión logística es descrito en la Figura 8.4. Es posible ver en la Figura 8.4 que el proceso comienza con la entrada de características y termina en la salida de etiqueta de predicción.

El proceso de estimación o predicción de la etiqueta está dada por la multiplicación de matrices de pesos y las caracteríticas más el valor de sesgo. A esto se le aplica la función sigmoide y se arroja el valor estimado $\hat{y}$.

$$\hat{y} = \sigma\left(w^T x + b\right), \quad \sigma(z) = \frac{1}{1+e^{-z}} \tag{8.3}$$

8.5.4. Funciones matemáticas relevantes

Será necesario esbozar algunas ecuaciones que servirán para la descripción del funcionamiento de la regresión logística.

$$z = w^T x + b \tag{8.4}$$

$$\hat{y} = a = \sigma(z) \tag{8.5}$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1-y)\log(1-a)) \tag{8.6}$$

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1-y^{(i)}\right) \log\left(1-\hat{y}^{(i)}\right) \tag{8.7}$$

El flujo de funcionamiento presentado en 8.4 es también explicado por estas ecuaciones de la siguiente manera.

- En la ecuación 8.4 se presenta el proceso principal de multiplicación de las características de entrada por la matriz de pesos más el vector de sesgo (bias).
- En la ecuación 8.5 se presenta la aplicación de la función sigmoide a la salida de 8.4. Esta es la estimación de 0 o 1.
- Adicionalmente, la función de pérdida 8.6 muestra una manera de medir el error de predicciones durante el proceso de entrenamiento.
- Por su parte, la ecuación 8.7 representa la función de costo. Esta implica la suma de la función de pérdida dividida entre el número de muestras "m". Esta será una función que se buscará minimizar. El valor que esta función toma indica qué tan bien se están comportando los pesos y sesgos en el set de entrenamiento.

8.5.5. Conjunto de entrenamiento y de test

En Inteligencia Artificial, en particular en Modelos de Machine Learning supervisados se suele dividir el conjunto de datos en primera instancia, dos conjuntos: Entrenamiento y Test. Usualmente se considera 80-70 % para entrenamiento y 20-30 % para test. En este caso en particular se está utilizando de la siguiente forma:

- 75 % para conjunto de entrenamiento
- 25 % para conjunto de test

Del total de 323 mediciones utilizadas, una vez que se dividió el dataset en la proporción descrita de manera aleatoria, la cantidad utilizada en cada uno es la siguiente:

- 242 mediciones para entrenamiento
- 81 mediciones para test

8.5.6. Estatus de Balance de datos

Durante el proceso es relevante conocer el balance de datos. Esto significa observar la proporción de aquellas mediciones que son etiquetadas con 1 en comparación con aquellas que son 0. A continuación se mostrará el estatus del balance en los tres conjuntos:

- Completo
- Entrenamiento
- Test

Algunas definiciones para contexto de este estatus son las siguientes:

- "Target positivo" significa un 1, es decir, un contacto.
- " No target positivo" significa un 0, es decir, no contacto.

Para el dataset de completo de 323 mediciones se encuentra la siguiente proporción:

- Target positivo = 107
- Porcentaje de target positivo = 33.126934984520126 %
- No Target positivo = 216
- Porcentaje de no target positivo = 66.87306501547987 %
- Total = 323

Dataset de entrenamiento con 242 mediciones:

- Target positivo = 78
- Porcentaje de target positivo = 32.231404958677686 %
- No Target positivo = 164
- Porcentaje de no target positivo = 67.76859504132231 %
- Total = 242

Dataset de test con 81 mediciones

- Target positivo = 29

- Porcentaje de target positivo = 35.80246913580247 %
- No Target positivo = 52
- Porcentaje de no target positivo = 64.19753086419753 %
- Total = 81

Esto es un conjunto de datos que tiene un desbalance notable pero no extremo. Se entiende que existe un desbalance cuando una de las clases posee la mayoría de los datos. Se puede realizar entrenamiento y evaluar con F1-score dado el caso desbalanceado.

8.5.7. Gradiente descendiente

Dado que la función 8.7 de costo se requiere minimizar para obtener un punto mínimo asociado a pesos y sesgo determinados, se introducirá el concepto de gradiente descendiente. Se ilustra en la Figura 8.5. El objetivo es llegar al mínimo global de la función a través de iteraciones que reducen el error del algoritmo, de tal forma que cuando se encuentre un punto mínimo se habrán encontrado los valores de peso y sesgo adecuados para el modelo. Qué tan rápido sean los pasos en esta búsqueda de mínimo estará dado por el concepto de "learning rate", en español, tasa de aprendizaje, representado con α en la Figura 8.5

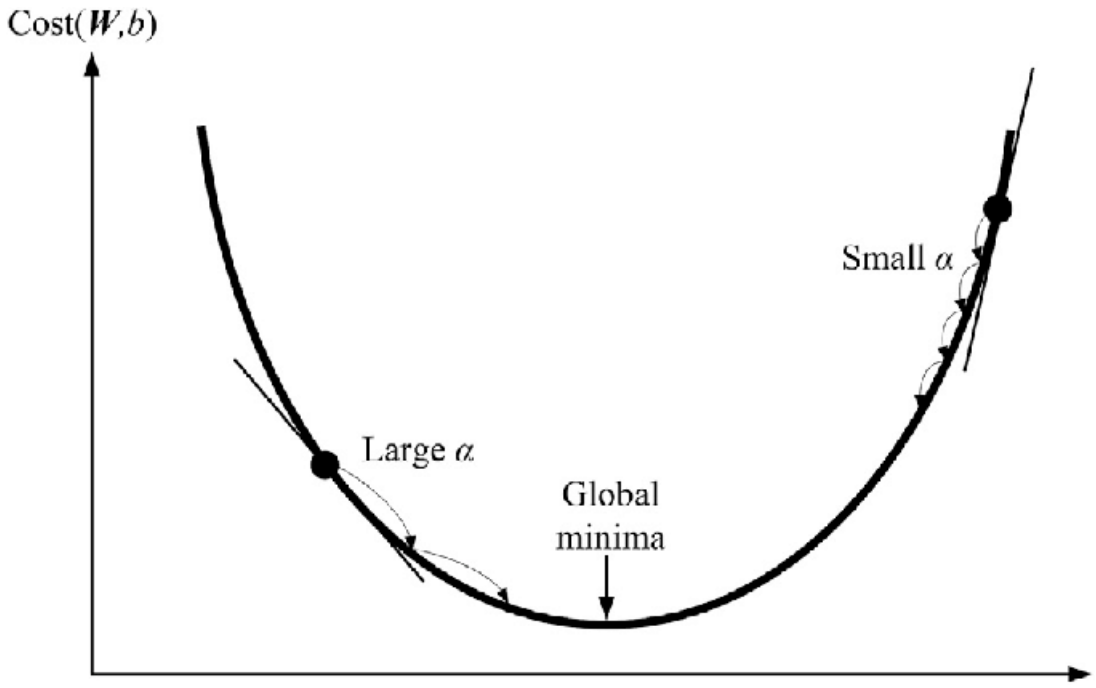


Figura 8.5: Función de costo y actualización de pesos y sesgo (30)

8.5.8. Actualización de pesos y de sesgo

En cada iteración a través de la tasa de aprendizaje, cuyos valores suelen estar entre 0.0 y 1.0, se actualizan los pesos y sesgos de la siguiente forma mostrada en las ecuaciones 8.8.

$$w_{new} = w - \alpha * \frac{\delta J}{\delta w}$$
$$b_{new} = b - \alpha * \frac{\delta J}{\delta b} \tag{8.8}$$

8.6. Resultados y Discusión del Modelo de Regresión Logística

Se presentarán los resultados y métricas utilizadas. Durante este proceso, se discutirá su significado e impacto en el caso de uso.

8.6.1. Matriz de confusión

Una matriz de confusión provee información útil para evaluar el desempeño de un modelo de Machine Learning supervisado. Es ilustrada en la figura 8.6. Aquellos valores representados como "Actual" representan las etiquetas verdaderas de valores 0 y 1, mientras que "Predicted" representa las predicciones (10). Esta matriz contrasta si las predicciones fueron acertadas o no.

En el caso de clasificación binaria existen cuatro valores internos los cuales son definidos de la siguiente forma.

- TP, en inglés True Positive, en español Verdadero Positivo, significa que el valor de predicción es 1 y la realidad también es 1.
- FP, en inglés False Positive, en español Falso Positivo, se define cuando la predicción es 1, pero el valor real es 0.
- FN, en inglés False Negative, en español Falso Negativo, se da cuando la predicción es 0, sin embargo la realidad es 1.
- TN, en inglés True Negative, en español Verdadero Negativo, existe cuando tanto el valor de predicción como el real son 0.

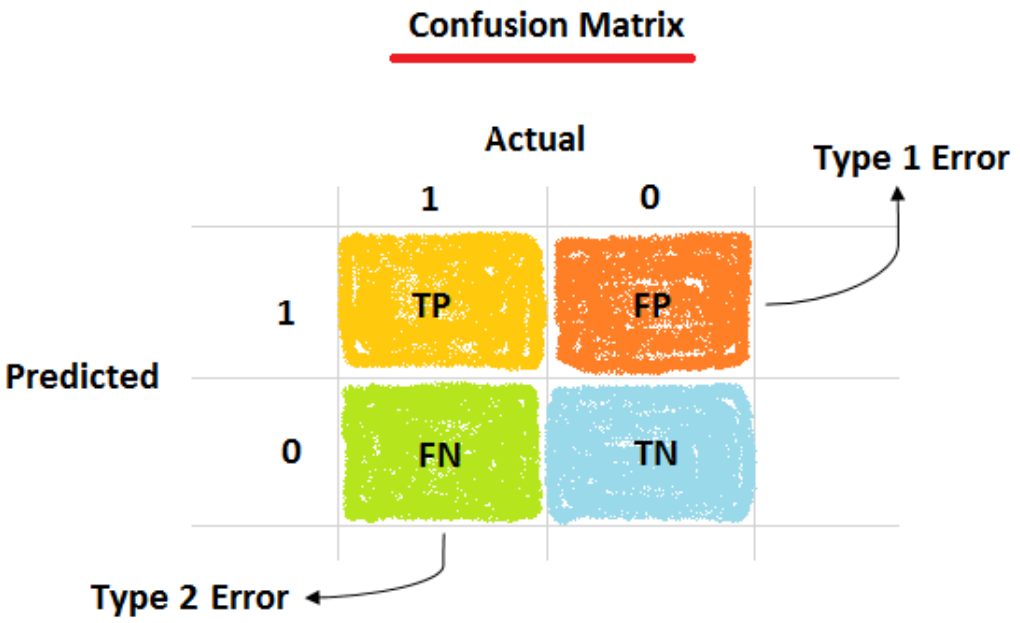


Figura 8.6: Matriz de confusión (10)

8.6.2. Errores Tipo 1 y Tipo 2

Estos dos tipos de errores aparecen cuando el modelo no predice todas las mediciones del conjunto de manera exacta y algunas de ellas son clasificadas incorrectamente, lo cual es un error (10). Se presentarán ambos los cuales también son ilustrados en la Figura 8.6.

- Error Tipo 1: Este error ocurre cuando el valor real es 0 pero el modelo predijo 1. En términos estadísiticos, este valor dice el nivel de significancia α * (10). El valor de alpha es elegido por quien realiza el análisis, entonces el error depende del valor elegido. Esto es un Falso Positivo.

  *Nota importante, esta α es distinta de la presentada anteriormente como tasa de aprendizaje.

- Error Tipo 2: Este error ocurre cuando el valor real es 1 pero el modelo predijo 0. En términos estadísticos, este error llamado β es principalmente

dependiente en el tamaño de muestra y la varianza (10). Esto es un Faso Negativo.

8.6.3. Métrica: Exactitud

Mide qué tan cercano es el valor de predicción al valor real.Se calcula de la siguiente forma en la ecuación 8.9.

$$\text{Exactitud } = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \tag{8.9}$$

8.6.4. Métrica: Precisión

Se define con base en los valores True Positive (Verdaderos Positivos) sobre el total de los 1 de predicción (TP+FP). Se calcula como es motrado en la ecuación 8.10.

$$\text{Precision } = \text{TP}/(\text{TP} + \text{FP}) \tag{8.10}$$

8.6.5. Métrica: *Recall*

Es también conocida como exhaustividad o tasa de cierto o tasa de verdaderos positivos. Indica qué tan bueno es el estimador o modelo para predecir valores positivos. La manera de calcular se encuentre en la ecuación 8.11.

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \tag{8.11}$$

8.6.6. Métrica: F1-Score

Esta métrica es también una media armónica de precisión y exhaustividad. F1-score indica la exactitud en términos de precisión y recall. Esta métrica es más adecuada para los problemas que tienen más dispersión, lo cual llevaría a existencia outliers. Es útil en clases desbalanceadas (10). Esta utilidad hace que sea una métrica a evaluar en el comportamiento del modelo pues este valor tiene embebidos los de precision y exhaustividad. Para calcularlo se muestra la expresión 8.12.

$$\text{F1-score } = 2^{*} \text{ precision } ^{*} \text{ recall } /(\text{ precision } + \text{ recall }) \tag{8.12}$$

8.6.7. Resultados y Discusión

La exactitud de este modelo es del 97 %, lo cual es una exactitud alta. Sin embargo, para evaluar al modelo es necesario tomar encuenta otras métricas como las mencionadas anteriormente. El Modelo de Machine Learning elegido tiene la siguiente Matriz de Confusión.

$$MatrizConfusionLR = \begin{bmatrix} 51 & 1 \\ 1 & 28 \end{bmatrix}$$

Los valores son los siguientes:

- TP: Verdaderos Positivos = 51
- FP: Falsos Positivos = 1
- FN: Falsos Negativos = 1
- TN: Verdaderos Negativos = 28

Es posible visualizar que tanto para el error tipo 1 y error tipo 2 solo se tiene un caso para cada uno. Fuera de ello, la clasificación está funcionando adecuadamente.

En este caso en particular, tanto precision como recall toman el valor de 0.98 lo cual influencía directamente al valor de F1-score, tomando un valor de 0.98 de igual manera. Eso sin duda indica una buena clasificación para ambas clases en el conjunto de datos desbalanceado.

Por otra parte, el modelo cuenta con la Matriz de Pesos W que tiene una estructura de (n_x,1) es decir, el número de renglones será equivalente al número de características y el columnas es el número de nodos. Para la regresión logística es solo 1. Por su parte, la Matriz de sesgo b contará con la estructura de (1,1).

A continuación se muestra con el mismo estilo de python los valores de las matrices en 8.6.7.

$$W = \begin{bmatrix} -0.01096605 \\ -1.50103241 \\ -0.26674058 \\ 0.16814001 \\ -0.46744182 \\ -0.2942094 \\ 0.34503892 \\ 1.79838073 \\ -0.24771069 \\ -0.34957081 \\ -0.52353762 \\ 0.03664752 \\ -1.50774391 \\ 0.92828383 \\ 0.9455384 \\ 0.26256895 \\ 0.57139162 \end{bmatrix}$$

De la misma manera, el valor de la matriz de sesgo está en la expresión 8.6.7.

$$b = \begin{bmatrix} -.63588659 \end{bmatrix}$$

Lo anterior alza el siguiente tema de discusión para casos futuros, ¿Qué es preferible? Procurar todos los contactos siempre registrarlos a costa de algunos falsos positivos o bien, no detectar todos los contactos y contar con falsos negativos. Este análisis es bastante común en el momento de selección de un Modelo de Machine Learning. Con base en ello se decide cuál es el tipo de error que se requiere minimizar tanto como sea posible.

8.7. Test de otros algoritmos de Machine Learning

Algunos otros modelos supervisados fueron puestos a prueba con algunos resultados menos satisfactorios y se documentan en el presente trabajo con el objetivo de tener una referencia de lo que sería un mal comportamiento.

- Modelo RBF SVM (Radial Basis Function - Support Vector Machine). En español, Máquina de Soporte Vectorial utilizando Función de Base Radial.

Su matriz de confusión muestra un sesgo hacia falsos negativos con 29 de ellos. Definitivamente no es un buen modelo.

Model accuracy: 64.19 %

$$MatrizConfusionRBFSVM = \begin{bmatrix} 52 & 0 \\ 29 & 0 \end{bmatrix}$$

- Modelo Decision Tree, en español, Ärbol de decisión se comportó de la siguiente forma, teniendo más falsos positivos que la Regresión Logística.

  Model accuracy: 96.29 %

$$MatrizConfusionDecisionTree = \begin{bmatrix} 50 & 2 \\ 1 & 28 \end{bmatrix}$$

- Modelo Red Neuronal, a pesar de contar con más capas ocultas y más nodos de cálculos que la Regresión Logística (modelo menos complejo), no tuvo un mejor desempeño. Esta red neuronal tiene un falso negativo y un falso positivo adicionales en contraste con la Regresión Logística. Model accuracy: 95.06 %

$$MatrizConfusionNeuralNet = \begin{bmatrix} 50 & 2 \\ 2 & 27 \end{bmatrix}$$

Existen modelos adicionales tales como el AdaBoost que pertenencen a una familia conocida como Boosting Algorithms caracterizados por usualmente tener mejor desempeño que los modelos más conocidos, por mencionar algunos: redes neuronales, árboles de decisión y regresión logística. En este caso, se evaluó adicionalmente un Algoritmo de AdaBoost. Sin embargo, el único cambio fue que tuvo un error menos en falsos negativos. En el caso de uso es viable utilizar un modelo más simple y sencillo como la Regresión Logística cuando la diferencia es tan pequeña.

Capítulo 9

Conclusiones y trabajo a futuro

9.1. Conclusiones

Durante este trabajo se ha documentado una propuesta de solución con un dispositivo cuya finalidad es lograr conocer y predecir el número de contactos que una persona tiene. Un gran elemento diferenciador de esta solución es la aplicación de tecnologías escalables tales como apps móviles y la nube, lo cual involucra la digitalización de todo el proceso. Adicionalmente, la implementación de un Modelo de Inteligencia Artificial, particularmente de Machine Learning supervisado, habilita una automatización ágil del proceso de estimación de contactos. La respuesta binaria de predicción del modelo permitirá encajar en diversos modelos epidemiológicos. El mayor valor de esta solución es la orientación que los datos de predicciones otorguen a modelos complementarios para el diseño, implementación y evaluación de políticas públicas en la materia, impactando directamente en el beneficio de la sociedad.

9.2. Trabajo a futuro

La funcionalidad e infraestructura están aseguradas en el dispositivo solución actual. Existe oportunidad de exploración en el diseño físico del dispositivo donde se detallen mayor abanico de oportunidades y evaluarse con base en criterios profundizados y pivotados con el usuario. Esto añadirá facilidad de uso y a un usuario con mejor conocimiento de las funcionalidades.

De la aplicación móvil, asegurar la lectura de mediciones y en caso contrario, un manejo de excepciones de posibles errores. Una correcta identificación de excepciones mediante pruebas de estrés permitirá que la aplicación sea más robusta.

Además, una integración y autenticación de usuarios permitirá que la app en un mismo dispositivo móvil funcione para diversos individuos. En cuanto a la base de datos, es deseable que para estar en un ambiente de producción se configuren los usuarios así como los permisos a los que estos tienen acceso. Adicionalmente, que se configurase algún servicio adicional de base de datos relacional como MongoDB con una infraestructura de réplicas en distintas ubicaciones geográficas aumentaría la disponibilidad de escritura y lectura. Dentro del dominio de escritura, herramientas como write concern serán útiles para asegurar la integridad de los datos.

Del Modelo de Machine Learning, entre mayor información exista la predicción será mejor. En ese sentido, incrementar el proceso de obtención de datos a través de un uso masivo de dispositivos podrá crear un dataset útil para entrenamiento de modelos de Machine Learning e incluso incursionar en el Aprendizaje Profundo (Deep Learning) obteniendo un mejor desempeño. Esto también habilitará la implementación de hiperparámetros en el proceso de aprendizaje del modelo, lo cual incrementa su desempeño y el abanico de experimentación. Adicionalmente, para enfrentar el futuro problema de desbalanceo es útil implementar técnicas de Undersampling como Random Undersampler y Oversampling como SMOTE (Synthetic Minority Oversampling Technique), las cuales potencialmente mejorarán el desempeño del modelo.

En cuanto a los sensores que actualmente forman parte del dispositivo un trabajo a futuro es incrementar la cantidad de variables físicas que el dispositivo es capaz de conocer de su entorno, tales como sensores de luz UV para conocer el estatus de un lugar abierto al aire libre o uno cerrado durante el dia. Otra variable es el sonido, de esta forma se puede predecir si las frecuencias que son recabadas corresponden a una conversación, lo cual aumentaría la intensidad del contacto.

Sobre la autonomía del análisis de Machine Learning, en este momento se encuentra en computadora de manera local, sin embargo las herramientas utilizadas como Scikit-learn de Python permite que el modelo sea escalable a una solución de servicios en la nube. Para tales efectos una propuesta de solución podría ser Amazon Web Services (AWS) con un bucket de S3 que esté albergando archivos estáticos del modelo entrenado y que este mismo pueda estar haciendo predicciones cada cierto tiempo con herramientas de como Apache Aiflow con un Directed Acyclic Graph (DAG) para tener un scheduler de las tareas a ejecutar. Esta misma herramienta puede estar alimentando una base de datos con las predicciones, dicha base puede estar en un servicio en la nube como Mongo Atlas para el caso de base de datos no relacional, o en el caso relacional una base de datos tipo SQL montada en una instancia elástica EC2 en AWS. La implementanción de estas herramientas dará escalabilidad y facilidad de ejecución masiva con el modelo. Se puede pensar en un proceso de DevOps para esta implementación.

Bibliografía

[1] (2013). *White Paper : D6T-44L/D6T-8L*. OMRON Electronic Components. -. xiii, 14

[2] (2018). *Logistic Regression - Detailed Overview*. Saishruthi Swaminathan - Towards Data Science. https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc (accessed: 08.12.2021). xv, 75

[3] (2020). *A logisitic regression from scratch. Translate Tensorflow to plain Numpy*. Dennis Bakhuis - Towards Data Science. https://towardsdatascience.com/a-logistic-regression-from-scratch-3824468b1f88 (accessed: 08.12.2021). xv, 76

[4] (2020). *OMRON D6T MEMS Thermal Sensors - High Sensitivity Enables Detection of Stationary Human Presence*. OMRON Corporation Electronic and Mechanical Components Company. Cat. No. A216-E1-05. xiii, xiii, xiii, 14, 15

[5] (2021). *Android Developers - Android Studio - User guide*. Google Developers. https://developer.android.com/studio/build (accessed: 17.10.2021). 26

[6] (2021a). *Arduino NANO*. Arduino. https://store.arduino.cc/usa/arduino-nano (accessed: 09.08.2021). xiii, 18, 19

[7] (2021b). *Arduino UNO REV3*. Arduino. https://store.arduino.cc/usa/arduino-uno-rev3 (accessed: 09.08.2021). 18

[8] (2021). *Berkeley Library GeoData*. University of California, Berkeley. https://geodata.lib.berkeley.edu/ (accessed: 02.12.2021). 66

[9] (2021a). *Chapter 3 - Machine Learning Types and their applications. Python Machine Learning and Data Science - 101 Series*. Souravi Sinha -

Medium.com. https://souravisinha.medium.com/chapter-3-machine-learning-types-and-their-applications-7b9f3fb8fc21 (accessed: 06.12.2021). xv, 73

[10] (2021b). *Confusion Matrix in Machine Learning. Classification metric in supervised learning.* Amit Chauhan - Medium. https://medium.com/analytics-vidhya/confusion-matrix-in-machine-learning-91b6e2b3f9af (accessed: 11.12.2021). xv, 81, 82, 83

[11] (2021). *Electrosome Interfacing HC-SR04 Ultrasonic Sensor with PIC Microcontroller.* Ligo George. https://electrosome.com/hc-sr04-ultrasonic-sensor-pic/ (accessed: 08.08.2021). xiii, 16

[12] (2021). *ESP32.* ESPRESSIF SYSTEMS (SHANGHAI) CO., LTD. https://www.espressif.com/en/products/socs/esp32 (accessed: 09.08.2021). 18

[13] (2021). *Firebase Realtime Database.* Google Developers. https://firebase.google.com/docs/database (accessed: 14.09.2021). 35

[14] (2021a). *IBM Cloud Learn Hub - What is Artificial Intelligence (AI)?* IBM. https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (accessed: 06.12.2021). xiv, 70, 71

[15] (2021b). *IBM Cloud Learn Hub - What is Machine Learning?* IBM. https://www.ibm.com/cloud/learn/machine-learning (accessed: 06.12.2021). 70, 72

[16] (2021). *Information Technology - Garner Glossary.* Gartner. https://www.gartner.com/en/information-technology/glossary (accessed: 06.12.2021). 70

[17] (2021). *Kotlin and Android Studio.* Android Developers - Google Developers. https://developer.android.com/kotlin (accessed: 14.08.2021). 25

[18] (2021). *KOTLIN VS JAVA: THE 12 DIFFERENCES YOU SHOULD KNOW.* Mariana Berga, Rute Figueiredo. https://www.imaginarycloud.com/blog/kotlin-vs-java/ (accessed: 14.08.2021). 25

[19] (2021). *Meet Android Studio.* Android Developers - Google Developers. https://developer.android.com/studio/intro (accessed: 14.08.2021). 23, 24

[20] (2021). *Mobile Operating System Market Share Worldwide.* Statcounter Global Stats. https://gs.statcounter.com/os-market-share/mobile/worldwide/ (accessed: 11.08.2021). xiii, 23, 24

[21] (2021a). *Modulo Bluetooth Hc05 Mercado Libre*. DIPMECATRONICA. https://bit.ly/3Asyr85 (accessed: 09.08.2021). xiii, 17

[22] (2021). *MSP430G2553 16 MHz MCU with 16KB Flash, 512B SRAM, comparator, UART/SPI/I2C, timer*. Texas Instruments Incorporated. https://www.ti.com/product/MSP430G2553 (accessed: 09.08.2021). 18

[23] (2021). *PIC16F887*. Microchip Technology Inc. https://www.microchip.com/en-us/product/PIC16F887 (accessed: 09.08.2021). 18

[24] (2021b). *Ultrasonic Ranging Module HC-SR04*. ElecFreaks. -. xiii, 16

[25] (2021c). *What is logistic regression?* IBM. https://www.ibm.com/topics/logistic-regression (accessed: 06.12.2021). 74

[26] (2021). *What is machine learning?* University of California, Berkeley. https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/ (accessed: 06.12.2021). 72, 73

[27] (2021). *What is the difference between Supervised, Unsupervised, Semi-supervised and Reinforcement Learning?* Nvidia. https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/ (accessed: 06.12.2021). 72

[28] (2022). *Tecnología vestible*. Wikipedia.org. https://es.wikipedia.org/wiki/Tecnología$_v$*estible*(*accessed* : 03.07.2022). 5

[29] David, N. (2014). Selection of variables and factor derivation. *Commercial Data Mining, Processing, Analysis and Modeling for Predictive Analytics Projects*. 63

[30] Man Gyun Na, Young Do Koo, Y. J. A. C.-H. K. (2018). Nuclear reactor vessel water level prediction during severe accidents using deep neural networks. *Nuclear Engineering and Technology*. xv, 80

[31] Martínez-Martínez, G. (2021a). *Created by the author on Lucidchart online tool*. www.lucidchart.com. xiii, 10

[32] Martínez-Martínez, G. (2021b). *Pictures and figures made by the author*. xiii, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, xiv, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53, 64, 67, 68

[33] McCarthy, J. (2004). What is artificial intelligence. *Computer Science Department, Stanford University*. 70

[34] Ng, A. (2003). *CS229 Lecture notes*. Winter 2003, University of Stanford. 74, 75

[35] Ridenhour, B., Kowalik, J. M., and Shay, D. K. (2018). El número reproductivo básico r0: consideraciones para su aplicación en la salud pública. *American Journal of Public Health*, 108:S455–S465. 2

[36] Romero-Rivas, V. (2021). *Additive manufacturer provider*. XIV, 44, 45

[37] Schläpfer, M., Dong, L., O'Keeffe, K., Santi, P., Szell, M., Salat, H., Anklesaria, S., Vazifeh, M., Ratti, C., and West, G. B. (2021). The universal visitation law of human mobility. *Nature*, 593:522–527. VII, VII, 1

[38] Ulrich, K. T. and Eppinger, S. D. (2013). *Diseño y desarrollo de productos*. Mc Graw Gill Education. 5, 7

Apéndice A

Códigos de Aplicación Móvil Android

A.1. Android Manifest XML

Listing A.1: Manifest of the Android Project

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.environment_unam_mx"
    android:versionCode="1"
    android:versionName="V1_GPS_Blue_D6T_Ultra_working">
    <!-- These are the required permissions to use Bluetooth
        android:roundIcon="@mipmap/ic_launcher_round" -->
    <!-- Internet:Location Fine: GPS and Coarse: Network location -->
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.
        BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.
        ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.
        ACCESS_COARSE_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/log_app_v1"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```
        android:theme="@style/Theme.Environment_UNAM_MX">
        <activity android:name=".LocTest"></activity>
        <activity
            android:name=".SelectDeviceActivity"
            android:screenOrientation="portrait" />
        <activity
            android:name=".SplashScreenActivity"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.AppCompat.DayNight.
                NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.
                    LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".ControlActivity"
            android:screenOrientation="portrait" />
    </application>

</manifest>
```

A.2. Build Gradle

Listing A.2: Build Gradle app

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    //The next one is totally neccessary to avoid unresolved reference error
    id 'kotlin-android-extensions'
    //id: 'com.google.gms.google-services'
}
apply plugin: 'com.google.gms.google-services'
//apply plugin: 'com.android.application'
//apply plugin: 'kotlin-android'
```

```
//apply plugin: 'kotlin-android-extensions'

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.2"

    defaultConfig {
        applicationId "com.example.environment_unam_mx"
        minSdkVersion 24
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner
            "
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
                optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
}

dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.firebase:firebase-database:19.7.0'
```

```
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    //This is necessary for the toast to work
    //Adds a lot of extra funcionality, like simple toast.
    implementation "org.jetbrains.anko:anko-common:0.8.3"
}
```

A.3. Splash Screen

Listing A.3: Splash Screen Kotlin File

```
package com.example.environment_unam_mx
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler

class SplashScreenActivity : AppCompatActivity() {

    companion object {
        var globalVar = "Created"
    }

    lateinit var handler: Handler
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_splash_screen)

        handler = Handler()
        handler.postDelayed({
            val intent = Intent(this,SelectDeviceActivity::class.java) //This
                is the part that
            // initializes the Next Activity
            startActivity(intent)
            finish()
```

```
        },4000) // Delaying that amount of miliseconds to open Main
            Activity

    }
}
}
```

Listing A.4: Splash Screen XML File

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".SplashScreenActivity"
    >


    <ImageView
        android:id="@+id/covidlogosplash"
        android:layout_width="350dp"
        android:layout_height="291dp"
        android:layout_marginTop="36dp"
        android:src="@drawable/covidresponseun"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.491"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"></ImageView>

    <ImageView
        android:id="@+id/unamlogosplash"
        android:layout_width="294dp"
        android:layout_height="198dp"
        android:layout_marginTop="8dp"
        android:src="@drawable/unamlogo"
        app:layout_constraintBottom_toTopOf="@+id/txtbottomsplash"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/covidlogosplash"></
            ImageView>

    <TextView
        android:id="@+id/txtbottomsplash"
        android:layout_width="277dp"
        android:layout_height="134dp"
        android:layout_marginBottom="32dp"
        android:fontFamily="sans-serif-black"
        android:text="@string/SplashScreenWelcome"
        android:textSize="30dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

A.4. Select Device Activity

Listing A.5: Select Device Activity Kotlin File

```
package com.example.environment_unam_mx
import android.app.Activity
import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothDevice
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.AdapterView
import android.widget.ArrayAdapter
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main.select_device_layout.*
import org.jetbrains.anko.toast

class SelectDeviceActivity : AppCompatActivity() {
    //A late-init var is a variable which is not initialized in
       that moment (late) (init)
```

```
//To make sure m_bluetoothAdapter can actually be a nullable
   object we add a ? (a lateinit
//cannot be a nullable object)
//These variables could be private, otherwise other classes can
    actually affect and interact
//with them, so I'm changing them to private.
private var m_bluetoothAdapter: BluetoothAdapter?=null
private lateinit var m_pairedDevices: Set<BluetoothDevice>
private lateinit var namedevice: String
private val REQUEST_ENABLE_BLUETOOTH = 1

//A companion objects works when I'd like to access them from
   other classes.
//When we are going to use them as the key for our extras prior
    intent extras so when
// we are moving data from one page to other
companion object{
    val EXTRA_ADDRESS: String="Device_address"
}

//onCreate starts whenever the page is started
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //For data base
    //var database = FirebaseDatabase.getInstance().reference
    //database.setValue("UNAM_Mexico")
    //


    setContentView(R.layout.select_device_layout)

    m_bluetoothAdapter=BluetoothAdapter.getDefaultAdapter()
    toast(getString(R.string.What2Do1))
    toast(getString(R.string.What2Do2))
    //Make sure it is not null
    if(m_bluetoothAdapter==null){
        toast(getString(R.string.BTnotsupported))
        return
    }
    //With !! that is saying that's not going to be null (
       Should work fine)
```

```
        //These lines are targeting to let us enable or disable
            bluetooth
        if (!m_bluetoothAdapter!!.isEnabled){
            val enableBluetoothIntent = Intent(BluetoothAdapter.
                ACTION_REQUEST_ENABLE)
            startActivityForResult(enableBluetoothIntent,
                REQUEST_ENABLE_BLUETOOTH)
        }
        //onClick listener to refresh list
        //We use curly braces{}
         select_device_refresh .setOnClickListener{pairedDeviceList()}
        btntestloc.setOnClickListener{
            //Calling another activity (ControlActivity, pointing
                out the address)
            val intent2 = Intent(this,LocTest::class.java)
            startActivity(intent2)
        }



    }


    private fun pairedDeviceList(){
        //toast(getString(R.string.try1))
        //I'm getting the devices
        m_pairedDevices=m_bluetoothAdapter!!.bondedDevices
        //Creating the ArrayList
        val list : ArrayList<BluetoothDevice> =ArrayList()
        val list2 : ArrayList<String> = ArrayList()
        if (!m_pairedDevices.isEmpty()){
            for(device:BluetoothDevice in m_pairedDevices){
                namedevice= device.getName()
                 list .add(device)
                 list2 .add(getString(R.string.name)+namedevice)
                //toast("1st is called: "+mmm)
                Log.i("Device",""+device)
            }

        } else {
            toast(getString(R.string.NoPairedDevFound))
```

```
        }
        //Show in the context the item stylings and the data:
        val adapter=ArrayAdapter(this, android.R.layout.simple_list_item_1,
            list )
        val adapter2=ArrayAdapter(this,android.R.layout.simple_list_item_1,
            list2)
        //Showing the list of devices
        //select_device_list.adapter=adapter
        aux_list .adapter=adapter2


        //I just need position so that I send just underscores
//        select_device_list.onItemClickListener = AdapterView.
    OnItemClickListener{_,_,position,_ ->
//            val device:BluetoothDevice = list[position]
//            val address: String = device.address
//            //Calling another activity (ControlActivity, pointing
    out the address)
//            val intent = Intent(this,ControlActivity::class.java)
//            intent.putExtra(EXTRA_ADDRESS,address)
//            startActivity(intent)
//        }

        aux_list .onItemClickListener=AdapterView.OnItemClickListener{_,_,
            position,_ ->
        val device:BluetoothDevice = list[position]
        val address: String = device.address
        //Calling another activity (ControlActivity, pointing out
            the address)
        val intent = Intent(this,ControlActivity::class.java)
        intent.putExtra(EXTRA_ADDRESS,address)
        startActivity(intent)
    }


    }

    //To tell the user the BT enabling was successful or not
    override fun onActivityResult(requestCode: Int,resultCode:Int,data:
        Intent?){
        super.onActivityResult(requestCode,resultCode,data)
```

```
        if (requestCode == REQUEST_ENABLE_BLUETOOTH){
            if(resultCode== Activity.RESULT_OK){
                if(m_bluetoothAdapter!!.isEnabled){
                    toast(getString(R.string.BTenabled))
                } else{
                    toast(getString(R.string.BTdisabled))
                }
            } else if (resultCode == Activity.RESULT_CANCELED){
                toast(getString(R.string.BTen_cancelled))
            }

        }
    }

}
```

Listing A.6: Select Device Activity XML File

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".SelectDeviceActivity">

<!--   <ListView-->
<!--       android:id="@+id/select_device_list"-->
<!--       android:layout_width="200dp"-->
<!--       android:layout_height="596dp"-->
<!--       app:layout_constraintBottom_toTopOf="@+id/select_device_refresh
    "-->
<!--       app:layout_constraintStart_toStartOf="parent"-->
<!--       app:layout_constraintTop_toTopOf="parent"-->
<!--       app:layout_constraintVertical_bias="0.0" />-->

    <Button
        android:id="@+id/select_device_refresh"
        android:layout_width="160dp"
        android:layout_height="50dp"
```

```
        android:layout_marginEnd="124dp"
        android:layout_marginBottom="216dp"
        android:text="@string/refresh"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <ListView
        android:id="@+id/aux_list"
        android:layout_width="399dp"
        android:layout_height="381dp"
        app:layout_constraintBottom_toTopOf="@+id/select_device_refresh"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias ="0.0" />

    <Button
        android:id="@+id/btntestloc"
        android:layout_width="117dp"
        android:layout_height="70dp"
        android:layout_marginBottom="80dp"
        android:text="@string/TestLoc"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias ="0.498"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.5. Control Activity

Listing A.7: Control Activity Kotlin File

```
package com.example.environment_unam_mx
import android.app.ProgressDialog
import android.bluetooth.BluetoothAdapter
```

```
import android.bluetooth.BluetoothDevice
import android.bluetooth.BluetoothSocket
import android.content.Context
import android.os.AsyncTask
import android.os.Build
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.database.FirebaseDatabase
import kotlinx.android.synthetic.main.control_layout.*
import java.io.IOException
import java.util.*
import com.example.environment_unam_mx.SplashScreenActivity.Companion.
    globalVar
import java.time.LocalTime
import com.example.environment_unam_mx.LocTest
import java.time.LocalDate

class ControlActivity: AppCompatActivity() {
    //Since we have an inner class, and always member variables are
        being accessed inside
    // we need to make the variables companion objects
    //Is able to be used, accessed in other classes.
    companion object {
        //The next explanations was taken on November 1st, 2020
            from
        // https://stackoverflow.com/questions/13964342/android-how
            -do-bluetooth-uuids-work
        //An Android phone can connect to a device and then use the
            Service Discovery Protocol
        // (SDP) to find out what services it provides (UUID).
        //In Bluetooth, all objects are identified by UUIDs. These
            include services, characteristics
        // and many other things. Bluetooth maintains a database of
            assigned numbers
        // for standard objects, and assigns sub-ranges for vendors
            (that have paid enough for
        // a reservation). You can view this list here:
```

```
    // https://www.bluetooth.com/specifications/assigned-
       numbers/
    //: I've solved that problem by hardcoding the UUID for
       Serial port service
    //as per this answer (using UUID.fromString
       ("00001101-0000-1000-8000-00805f9b34fb")
    var m_myUUID: UUID = UUID.fromString("
       00001101-0000-1000-8000-00805f9b34fb")

    //To create a Bluetooth socket to send info
    var m_bluetoothSocket: BluetoothSocket? = null

    //Certain methods are deprecated in Java, but they still
       work for our purposes.
    lateinit var m_progress: ProgressDialog
    lateinit var m_bluetoothAdapter: BluetoothAdapter
    var m_isConnected: Boolean = false
    lateinit var m_address: String
    var sneezecounter = 0
    var coughcounter = 0
    var dbcounter = 0
    lateinit var string_key: String
    lateinit var string_aux: String
    lateinit var string2_aux: String
    lateinit var string3_aux: String
    lateinit var string_received: String



}

@RequiresApi(Build.VERSION_CODES.O)
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    var database = FirebaseDatabase.getInstance().reference
    database.setValue("UNAM_Mexico_ControlActivity")

    //To set the UI
    setContentView(R.layout.control_layout)
```

```
//This way we know for a fact that we will be getting the
    right value from the extras.
//Here I made a change, the original one was
//m_address = intent.getStringExtra
m_address = intent.getStringExtra(SelectDeviceActivity.
    EXTRA_ADDRESS).toString()

ConnectToDevice(this).execute()
//This is where I send the information.
control_led_on .setOnClickListener {
    sendCommand("a")
    Toast.makeText(this, "Mandaste una a", Toast.LENGTH_LONG)
        .show()
}
 control_led_off .setOnClickListener {
    sendCommand("b")
    //sendCommand("c")
}


control_led_disconnect .setOnClickListener { disconnect() }

cloud_db_send.setOnClickListener{
    sendCommand("c")
     string_received  = receiveData()
    dbcounter += 1
    string_key  = "measure_"+ dbcounter.toString()
    //database.setValue("UNAM_Mexico_ControlActivity_"+
        dbcounter.toString() )
    var prueba1 = 1
    var prueba2 = 2
    var ubicaciones = ""
    //var classLocTest = LocTest()
    ubicaciones = globalVar
    //ubicaciones = objeto.getLocation()
    //Log.i("-------------------Will try:","to create
        LocTest")
    //var anotherclass = LocTest()
    //Log.i("-------------------Done:","Created LocTest")
    //ubicaciones = anotherclass.getLocation()
```

```
        //Log.i("-------------------Done:", "Created ubicaciones
            "+ubicaciones)
        //Time
        val currentTime = LocalTime.now()
        val currentDate = LocalDate.now()
        var current_time_to_db = currentTime.toString()
        var current_date_to_db = currentDate.toString()
        database.child(string_key).setValue(Measurement(string_received,
            coughcounter, sneezecounter,ubicaciones, prueba1,
            current_time_to_db, current_date_to_db))
    }

    btnaux.setOnClickListener{
        sendCommand("c")
        //Log.i("Entr al btnauxsetOnClickListener", "Entr al
            btnauxetOnClickListener")
        string_received =receiveData()
        //Here is where I have the received information. I'll
            try to send it to the cloud.
        textView_received.text= string_received
    }

    btncough.setOnClickListener {
        coughcounter = coughcounter + 1
        Log.i(" Entr  al setOnClickListener", "Entr al
            setOnClickListener")
        string_aux = getString(R.string.CoughReported)
        Log.i("done get string", "done get string")
        string2_aux = coughcounter.toString()
        Log.i("done get string2", "done get string2")
        string3_aux = string_aux + " " + string2_aux
        Log.i("concat", "concat")
        Toast.makeText(this, string3_aux, Toast.LENGTH_LONG).show()
        //Toast.makeText(this, R.string.CoughReported , Toast.
            LENGTH_LONG).show()}//+ coughcounter.toString()
    }


    btnsneeze.setOnClickListener {
        sneezecounter = sneezecounter + 1
```

```
            Log.i(" Entr  al setOnClickListener", "Entr al
                setOnClickListener")
            string_aux = getString(R.string.SneezeReported)
            Log.i("done get string", "done get string")
            string2_aux = sneezecounter.toString()
            Log.i("done get string2", "done get string2")
            string3_aux = string_aux + " " + string2_aux
            Log.i("concat", "concat")
            Toast.makeText(this, string3_aux, Toast.LENGTH_LONG).show()
            //Toast.makeText(this, R.string.SneezeReported, Toast.
                LENGTH_LONG).show()}//+ sneezecounter.toString()
        }
    }
        //It will send data to the Arduino
        private fun sendCommand(input: String) {
            //If the bluetoothSocket is not null, it is necessary to
                send it as a Byte Array
            //Just like I do when I encode from UTF-8 on Python
            if (m_bluetoothSocket != null) {
                try {
                    m_bluetoothSocket!!.outputStream.write(input.toByteArray
                        ())
                } catch (e: IOException) {
                    //It will show why it failed in the case when
                        that happened.
                    e.printStackTrace()
                }
            }

        }
//    fun InputStream.readUpToChar(stopChar: Char): String {
//        val stringBuilder = StringBuilder()
//        var currentChar = this.read().toChar()
//        while (currentChar != stopChar) {
//            stringBuilder.append(currentChar)
//            currentChar = this.read().toChar()
//            if (this.available() <= 0) {
//                stringBuilder.append(currentChar)
//                break
//            }
//        }
```

```
//        return stringBuilder.toString()
//    }

        private fun receiveData(): String {
            val buffer=ByteArray(1024)
            var bytes: Int
            var bytestwo: Int
            lateinit var readMessage: String
            var num_av:Int
            var stpchar: Char = 'c'
            Log.i(" Entr  al receiveData()", "Entr al receiveData()")
            if(m_bluetoothSocket!=null){
                num_av= m_bluetoothSocket!!.inputStream.available()
                Log.i(" Entr  a", "num_av= m_bluetoothSocket!!.
                    inputStream.available()")
                Log.i("And1", "num_av is"+num_av.toString())
               try{
                    while(num_av<107){
                        num_av= m_bluetoothSocket!!.inputStream.available()
                        Log.i(" Entr  al ", "while num_av=
                            m_bluetoothSocket!!.inputStream.available()
                            ")
                        Log.i("And2", "num_av is"+num_av.toString())
                    }
                    Log.i("Will try","to enter bytes= m_bluetoothSocket
                        !!.inputStream.read(buffer)")
                    bytes= m_bluetoothSocket!!.inputStream.read(buffer)//
                        Necessary !! to
                    Log.i("Successfully done: ","bytes=
                        m_bluetoothSocket!!.inputStream.read(buffer)")
                    // express is not empty
                    Log.i(" Entr  al bytes= m_bluetoothSocket!!.
                        inputStream.read(buffer)", "Entr al bytes=
                        m_bluetoothSocket!!.inputStream.read(buffer)")
                    readMessage=String(buffer,0, bytes)//The bytes array
                        and the
                    // buffer using UTF-8

                } catch(e: IOException){
                    e.printStackTrace()
                    readMessage = "ErrorToRead"
```

```
            }
        }
        return readMessage

    }


    //Disconnect function
    private fun disconnect() {
        //The same condition, if the bluetooth socket is not
            null.
        if (m_bluetoothSocket != null) {
            try {
                m_bluetoothSocket!!.close()
                m_bluetoothSocket = null
                m_isConnected = false
            } catch (e: IOException) {
                //Shows why it has failed in case it happen to
                    happen
                e.printStackTrace()
            }
        }
        finish()
    }

    //Called as initialized When we attempt to connect with
        Bluetooth which extends AsyncTask
    //It requires the context to be passed in
    private class ConnectToDevice(c: Context) : AsyncTask<Void, Void,
        String>() {

        //We need a constructor for this class and its variables
        private var connectSuccess: Boolean = true
        private val context: Context

        //the last one needs to be initialized in the next lines
            .
        //It might want to us to do it differently, field leak
            issue. Complaining about memory
        init {
            this.context = c
```

```
}

override fun onPreExecute() {
    super.onPreExecute()
    //m_progress = ProgressDialog.show(context,R.string.
        Connecting.toString(),R.string.PlsWait.toString()
        ) // DIALOG PROGRESS
    Toast.makeText(this.context, R.string.Connecting, Toast.
        LENGTH_LONG).show()
    //toast("Something")
    //toast(R.string.Connecting.toString())
    //toast("FUNCIONA PLIS ")
    //toast(R.string.Connecting.toString())

}

//We add the ? to it to cannot return null if it needs.
override fun doInBackground(vararg params: Void?): String? {
    try {
        if (m_bluetoothSocket == null || !m_isConnected) {
            m_bluetoothAdapter = BluetoothAdapter.
                getDefaultAdapter()
            //The address we got that from the intent
            val device: BluetoothDevice = m_bluetoothAdapter.
                getRemoteDevice(m_address)
            //To set up the socket with the device with my
                UUID
            //Sets up the connection between our phone and
                our Bluetooth device we want to
            // connect to.
            m_bluetoothSocket = device.
                createInsecureRfcommSocketToServiceRecord(
                m_myUUID)
            //It stops from looking for other devices
                trying to connect to thereby saving
            //our backs, basically saving battery! and
                resources.
            BluetoothAdapter.getDefaultAdapter().cancelDiscovery
                ()
            m_bluetoothSocket!!.connect()
```

```
                }

            } catch (e: IOException) {
                connectSuccess = false
                e.printStackTrace()
            }
            return null
        }

        override fun onPostExecute(result: String?) {
            super.onPostExecute(result)
            if (!connectSuccess) {
                Log.i("data", R.string.CouldntConnect.toString())
            } else {
                m_isConnected = true
            }
            //m_progress.dismiss() //DIALOG PROGRESS
        }

    }

}
```

Listing A.8: Control Activity XML File

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

    <Button
        android:id="@+id/control_led_on"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="36dp"
        android:text="@string/on"
        app:layout_constraintHorizontal_bias="0.47"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/cloud_db_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/DBSend"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.531"
        app:layout_constraintStart_toEndOf="@+id/btnaux"
        app:layout_constraintTop_toBottomOf="@+id/control_led_off"
        tools:ignore="MissingConstraints" />

    <Button
        android:id="@+id/control_led_off"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/off"
        app:layout_constraintHorizontal_bias="0.47"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/control_led_on" />

    <Button
        android:id="@+id/control_led_disconnect"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/disconnect"
        app:layout_constraintHorizontal_bias="0.46"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnaux" />

    <ImageButton
        android:id="@+id/btncough"
        android:layout_width="131dp"
        android:layout_height="137dp"
```

```
        android:layout_marginStart="44dp"
        android:adjustViewBounds="false"
        android:cropToPadding="false"
        android:scaleType="fitCenter"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/btnsneeze"
        app: layout_constraintHorizontal_bias ="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app: layout_constraintVertical_bias ="0.612"
        app:srcCompat="@drawable/coughdos" />

    <ImageButton
        android:id="@+id/btnsneeze"
        android:layout_width="142dp"
        android:layout_height="132dp"
        android:layout_marginEnd="44dp"
        android:scaleType="fitCenter"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app: layout_constraintVertical_bias ="0.616"
        app:srcCompat="@drawable/sneezedos" />

    <TextView
        android:id="@+id/textViewbtncough"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="60dp"
        android:text="@string/Coughbtn"
        app:layout_constraintBottom_toTopOf="@+id/btncough"
        app:layout_constraintEnd_toStartOf="@+id/textView2btnsneeze"
        app: layout_constraintHorizontal_bias ="0.011"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app: layout_constraintVertical_bias ="0.955" />

    <TextView
        android:id="@+id/textView2btnsneeze"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginEnd="76dp"
        android:text="@string/Sneezebtn"
        app:layout_constraintBottom_toTopOf="@+id/btnsneeze"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app: layout_constraintVertical_bias ="0.955" />

    <TextView
        android:id="@+id/textView_received"
        android:layout_width="252dp"
        android:layout_height="145dp"
        android:layout_marginBottom="52dp"
        android:text="TextView"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app: layout_constraintHorizontal_bias ="0.496"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btnaux"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="@string/RqstData"
        app:layout_constraintEnd_toEndOf="parent"
        app: layout_constraintHorizontal_bias ="0.27"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/control_led_off"
            />/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

A.6. Location Activity Kotlin File

Listing A.9: Location Activity Kotlin File

```
package com.example.environment_unam_mx
import android.Manifest
import android.annotation.SuppressLint
```

```
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.location.Location
import android.location.LocationListener
import android.location.LocationManager
import android.os.Build
import android.os.Bundle
import android.provider.Settings
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import kotlinx.android.synthetic.main. activity_loc_test .*
import com.example.environment_unam_mx.SplashScreenActivity.Companion.
    globalVar
import java.util.*

private const val PERMISSION_REQUEST = 10

class LocTest : AppCompatActivity() {

    lateinit var locationManager: LocationManager
    public var hasGps = false
    public var hasNetwork = false
    public var locationGps: Location? = null
    public var locationNetwork: Location? = null

    public var permissions = arrayOf(Manifest.permission.
        ACCESS_FINE_LOCATION, Manifest.permission.
        ACCESS_COARSE_LOCATION)

    public var locmediciones = "NotAvailable"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout. activity_loc_test )
        disableView()
        //Next lines are to ensure I got permissions from the user.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (checkPermission(permissions)) {
                enableView()
            } else {
```

```
                requestPermissions(permissions, PERMISSION_REQUEST)
                Toast.makeText(this, R.string.GrantPerm, Toast.
                    LENGTH_LONG).show()
            }
        } else {
            enableView()
        }

    }
    public fun disableView() {
        btngetloc.isEnabled = false //Disable button
        btngetloc.alpha = 0.5F //It looks like disabled
    }

    public fun enableView() {
        btngetloc.isEnabled = true
        btngetloc.alpha = 1F
        btngetloc.setOnClickListener { globalVar=getLocation()}
        Toast.makeText(this, R.string.LocOk, Toast.LENGTH_SHORT).show
            ()
    }
    //The lint tool helps find poorly structured code that can
        impact the reliability and
    // efficiency of your Android apps and make your code harder to
         maintain.
    @SuppressLint("MissingPermission")

    public fun getLocation(): String {
        var latitud_measured_gps = "NoData"
        var longitud_measured_gps = "NoData"
        var loc_accuracy_gps = "NoData"
        var latitud_measured_network = "NoData"
        var longitud_measured_network = "NoData"
        var loc_accuracy_network = "NoData"
        locationManager = getSystemService(Context.LOCATION_SERVICE)
            as LocationManager
        hasGps = locationManager.isProviderEnabled(LocationManager.
            GPS_PROVIDER)
        hasNetwork = locationManager.isProviderEnabled(LocationManager.
            NETWORK_PROVIDER)
        if (hasGps || hasNetwork) {
```

```
if (hasGps) {
    Log.d("CodeAndroidLocation", "hasGps")
    locationManager.requestLocationUpdates(LocationManager.
        GPS_PROVIDER, 5000, 0F, object : LocationListener {
        override fun onLocationChanged(location: Location) {
            if (location != null) {
                locationGps = location
                txtviewlocation.append("\nGPS ")
                txtviewlocation.append("\nLatitude : " +
                    locationGps!!.latitude)
                txtviewlocation.append("\nLongitude : " +
                    locationGps!!.longitude)
                Log.d("CodeAndroidLocation", " GPS Latitude
                    : " + locationGps!!.latitude)
                Log.d("CodeAndroidLocation", " GPS Longitude
                     : " + locationGps!!.longitude)
                latitud_measured_gps = locationGps!!.latitude.
                    toString()
                longitud_measured_gps = locationGps!!.longitude.
                    toString()
                loc_accuracy_gps = locationNetwork!!.accuracy.
                    toString()
                globalVar = Arrays.toString(arrayOf(
                    latitud_measured_gps,longitud_measured_gps,
                    loc_accuracy_gps))
            }
        }

        override fun onStatusChanged(provider: String?, status:
            Int, extras: Bundle?) {

        }

        override fun onProviderEnabled(provider: String) {

        }

        override fun onProviderDisabled(provider: String) {

        }
```

```
        })

        val localGpsLocation = locationManager.
            getLastKnownLocation(LocationManager.
            GPS_PROVIDER)
        if (localGpsLocation != null)
            locationGps = localGpsLocation
    }
    if (hasNetwork) {
        Log.d("CodeAndroidLocation", "hasGps")
        locationManager.requestLocationUpdates(LocationManager.
            NETWORK_PROVIDER, 5000, 0F, object :
            LocationListener {
            override fun onLocationChanged(location: Location) {
                if (location != null) {
                    locationNetwork = location
                    txtviewlocation.append("\nNetwork ")
                    txtviewlocation.append("\nLatitude : " +
                        locationNetwork!!.latitude)
                    txtviewlocation.append("\nLongitude : " +
                        locationNetwork!!.longitude)
                    Log.d("CodeAndroidLocation", " Network
                        Latitude : " + locationNetwork!!.latitude)
                    Log.d("CodeAndroidLocation", " Network
                        Longitude : " + locationNetwork!!.longitude)
                    latitud_measured_network = locationNetwork!!.
                        latitude.toString()
                    longitud_measured_network = locationNetwork!!.
                        longitude.toString()
                    loc_accuracy_network = locationNetwork!!.accuracy
                        .toString()
                    globalVar = Arrays.toString(arrayOf(
                        latitud_measured_network,
                        longitud_measured_network,
                        loc_accuracy_network))
                }
            }

            override fun onStatusChanged(provider: String?, status:
                Int, extras: Bundle?) {
```

```
            }

            override fun onProviderEnabled(provider: String) {

            }

            override fun onProviderDisabled(provider: String) {

            }

        })

        val localNetworkLocation = locationManager.
            getLastKnownLocation(LocationManager.
            NETWORK_PROVIDER)
        if (localNetworkLocation != null)
            locationNetwork = localNetworkLocation
    }

    if(locationGps!= null && locationNetwork!= null){
        if(locationGps!!.accuracy > locationNetwork!!.accuracy){
            txtviewlocation.append("\n Network location is more
                accurate")
            txtviewlocation.append("\nNetwork ")
            txtviewlocation.append("\nLatitude : " +
                locationNetwork!!.latitude)
            txtviewlocation.append("\nLongitude : " +
                locationNetwork!!.longitude)
            Log.d("CodeAndroidLocation", " Network Latitude : "
                 + locationNetwork!!.latitude)
            Log.d("CodeAndroidLocation", " Network Longitude :
                " + locationNetwork!!.longitude)
        }else{
            txtviewlocation.append("\n GPS location is more
                accurate")
            txtviewlocation.append("\nGPS ")
            txtviewlocation.append("\nLatitude : " + locationGps
                !!.latitude)
            txtviewlocation.append("\nLongitude : " + locationGps
                !!.longitude)
```

```
                    Log.d("CodeAndroidLocation", " GPS Latitude : " +
                        locationGps!!.latitude)
                    Log.d("CodeAndroidLocation", " GPS Longitude : " +
                        locationGps!!.longitude)
                }
            }

        } else {
            startActivity(Intent(Settings.
                ACTION_LOCATION_SOURCE_SETTINGS))
        }

        //return arrayOf(latitud_measured_gps,longitud_measured_gps
            ,loc_accuracy_gps,latitud_measured_network,
            longitud_measured_network,loc_accuracy_network).toString
            ()
        return Arrays.toString(arrayOf(latitud_measured_gps,
            longitud_measured_gps,loc_accuracy_gps,latitud_measured_network,
            longitud_measured_network,loc_accuracy_network))
    }

    public fun checkPermission(permissionArray: Array<String>): Boolean
        {
        var allSuccess = true
        for (i in permissionArray.indices) {
            if (checkCallingOrSelfPermission(permissionArray[i]) ==
                PackageManager.PERMISSION_DENIED)
                allSuccess = false
        }
        return allSuccess
    }
    override fun onRequestPermissionsResult(requestCode: Int, permissions:
        Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
            grantResults)
        if (requestCode == PERMISSION_REQUEST) {
            var allSuccess = true
            for (i in permissions.indices) {
                if (grantResults[i] == PackageManager.
                    PERMISSION_DENIED) {
                    allSuccess = false
```

```
                    val requestAgain = Build.VERSION.SDK_INT >= Build.
                        VERSION_CODES.M &&
                        shouldShowRequestPermissionRationale(permissions[i])
                    if (requestAgain) {
                        Toast.makeText(this, R.string.PermDenied, Toast.
                            LENGTH_SHORT).show()
                    } else {
                        Toast.makeText(this, R.string.Go2Settings, Toast.
                            LENGTH_SHORT).show()
                    }
                }
            }
            if (allSuccess)
                enableView()

        }
    }
}
```

Listing A.10: Location Activity XML File

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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=".LocTest">

    <Button
        android:id="@+id/btngetloc"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/GetLoc"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.262" />
```

```
    <TextView
        android:id="@+id/txtviewlocation"
        android:layout_width="324dp"
        android:layout_height="467dp"
        android:text="TextView"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btngetloc"
        app:layout_constraintVertical_bias="0.504" />

    <ImageView
        android:id="@+id/imviewlocation"
        android:layout_width="131dp"
        android:layout_height="122dp"
        app:layout_constraintBottom_toTopOf="@+id/btngetloc"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/location" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

A.7. Measurement Class

Listing A.11: Measurement Class Kotlin File

```
package com.example.environment_unam_mx
class Measurement {
    var measure = ""
    var n_of_cough = 0
    var n_of_sneeze = 0
    var locinfo = ""
    var aux1 = 0
    var time_read = ""
    var date_read = ""
    constructor(measure:String, n_of_cough:Int, n_of_sneeze:Int,locinfo:
        String, aux1:Int, time_read:String, date_read:String){
```

```
        this.measure = measure
        this.n_of_cough = n_of_cough
        this.n_of_sneeze  = n_of_sneeze
        this. locinfo  = locinfo
        this.aux1 = aux1
        this.time_read = time_read
        this.date_read = date_read

    }
}
```

Apéndice B

Códigos de funcionamiento del dispositivo

B.1. Microcontrolador Arduino: Sensor D6T, Comunicación Bluetooth y Sensor Ultrasónico

```
//Universidad Nacional Autnoma de Mxico – Facultad de Ingeniera .
//Tesis: Inteligencia Artificial aplicada en el diseo de un dispositivo
    para registrar el nmero de contactos de una persona en el marco de
    COVID-19
// Ingeniera Mecatrnica . Noviembre 2020.
//Asesora de tesis: Mtra. Rosa Itzel Flores Luna
//Cotutor de tesis: Dr. Edmundo Gabriel Rocha Czatl
//Estudiante: Guillermo Martnez Martnez
//Bluetooth_D6T_Ultrasonico

//----------D6T and Distance Variables and
    requirements----------//
#include <Wire.h>
#include <WireExt.h>

#define D6T_addr 0x0A // Address of OMRON D6T is 0x0A in hex
#define D6T_cmd 0x4C // Standard command is 4C in hex
```

```
#define echoPin 2 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 3 //attach pin D3 Arduino to pin Trig of HC-SR04
int rbuf[35]; // Actual raw data is coming in as 35 bytes. Hence the needed
    for WireExt so that we can handle more than 32 bytes
float tdata[16]; // The data comming from the sensor is 16 elements, in a 16
    x1 array
float t_PTAT;

const int boton=7;
int val_boton;


// defines variables
long duration; // variable for the duration of sound wave travel
int distance=0; // variable for the distance measurement
int distance1=0;
int distancepast=0;
int contador=0;

//--------------------Bluetooth variables and requirements
    ----------------------//
//(None)

void setup() {
  // -------Setup D6T and distance---//
  Wire.begin();
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  pinMode(13, OUTPUT);
  pinMode(boton,INPUT);
  //----------Setup Bluetooth----//
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  if(Serial.available()>0)
  {
```

```
    char data=Serial.read();
    switch(data)
    {
      case 'a': digitalWrite(8,HIGH);break;
      case 'b': digitalWrite(8,LOW);break;
      case 'c':
      //If a request is received, then send the data.
      d6tandultrasonic();
      digitalWrite(9, HIGH);
      delay(300);
      digitalWrite(9,LOW);
      break;
      default: break;
    }
    //Serial.println(data);
    }
}


void d6tandultrasonic()
{
  int i;
      // Begin a regular i2c transmission. Send the device address and then
          send it a command.
      Wire.beginTransmission(D6T_addr);
      Wire.write(D6T_cmd);
      Wire.endTransmission();
      delay(5);
      val_boton=digitalRead(boton);
      if(val_boton==0)
        val_boton=1;
      else
        val_boton=0;
      // This is where things are handled differently. Omron D6T output
          data is 35 bytes and there is a limit here on what Wire can receive.
          We use WireExt to read the output data 1 piece at a time. We store
          each byte as an element in rbuf.
      if (WireExt.beginReception(D6T_addr) >= 0) {
        i = 0;
        for (i = 0; i < 35; i++) {
          rbuf[i] = WireExt.get_byte();
```

```
      }
      // end the reception routine
      WireExt.endReception();

      // Do something with temperature compensation that we don't seem
          to use currently.NO SE USA ESTA LINEA.
      t_PTAT = (rbuf[0]+(rbuf[1]<<8))*0.1;
      //Aqu se juntan los Most Significant Bit y Less Significant Bit
      // Calculate the temperature values: add the low temp and the bit
          shifted high value together. Multiply by 0.1
      for (i = 0; i < 16; i++) {
        tdata[i]=(rbuf[(i*2+2)]+(rbuf[(i*2+3)]<<8))*0.1;
      }
    }


    // Use a for loop to output the data. We can copy this from serial
        monitor and save as a CSV
    for (i=0; i<16; i++) {
      Serial.print(tdata[i]);
      Serial.print(",");
    }
    Serial.print(t_PTAT);
    Serial.print(",");
    Serial.print(val_boton);
    Serial.print(",");
    //Serial.print("\n");
    //Comienzo sensor distancia
      digitalWrite(trigPin, LOW);
delayMicroseconds(5);
// Sets the trigPin HIGH (ACTIVE) for 10 microseconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Reads the echoPin, returns the sound wave travel time in microseconds
duration = pulseIn(echoPin, HIGH);
// Calculating the distance
distance1=duration * 0.034 / 2;
if(distance1<320)
{
  distance = distance1;
```

```
    }
    else
    {distance=distancepast;
    }
    // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  //Serial.print("Distance: ");

  distancepast=distance;
       Serial.print(distance);
       Serial.print(",");
       Serial.print("\n");
      // Serial.print(contador);
       //Serial.print("\n");
       //contador=contador+1;
       delay(70);//Para tener aprox todo en 60 segundos (300 muestras) porque
            cada ciclo se tarda 130 ms en el micro ms el delay.
//Esto es para ver el contenido del buffer que almacena cada valor en bruto
  //     for (i=0; i<sizeof(rbuf); i++) {
      //    Serial.print(rbuf[i]);
    //      Serial.print(",");
    //    }
    //    Serial.print("\n");
     //       delay(5000);
}
```

B.2. Script de adquisición de datos local

Listing B.1: Adquisición de datos local - Antes de integración con el dispositivo.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 18 16:53:15 2020

@author: Guillermo Martnez Martnez
Universidad Nacional Autnoma de Mxico. Facultad de Ingeniera.
Servicio Social - Tesis.
 Direccin de Tesis: Mtra. Rosa Itzel Flores Luna
```

```
 Colaboracin  con Dr. Edmundo Gabriel Rocha Czatl
Estudiante: Guillermo Martnez  Martnez .      Ingeniera   Mecatrnica
No. de cuenta. 313157886. Contacto: guillermo.martinezmartinez@outlook.com
" Inteligencia  Artificial  aplicada en el  diseo  de un dispositivo para
    registrar
 el  nmero  de contactos de una persona en el marco de COVID-19"
 ------------------------------------------------------------------------

 Versin  Funcional al 19 de noviembre de 2020 21:51 hrs
 Recopilacin  en excel y datos sin problema
 Creacin  de las animaciones con imgenes  en GIF funcional
Valores propios de la  matriz y determinante
 Aado  valores propios
"""
#pip install  pyserial  must be installed for the  first  time (In a new
    machine)
#pip install  celluloid  must be installed for the  first  time (In a new
    machine)
import serial
import time
#import pandas as pd
from openpyxl import Workbook
import datetime
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from celluloid  import Camera
test_name='Sensor_moviendose2'############
matrix_sensor_order=4
sample_size  = 300
book = Workbook()
sheet  = book.active
headers=("Sample","Time","Ultrasonic-Distance[cm]","Determinant","PTAT[
    C],")
headers=list(headers)
head_temps=["T"+str(j+1)+"[C]"  for j in range(matrix_sensor_order**2)]
head_eigvalues=[]
for j  in range(matrix_sensor_order):
    head_eigvalues.append("Eigenvalue"+str(j+1))
    head_eigvalues.append("Eig-val"+str(j+1)+"_Modulus")
    head_eigvalues.append("Eig-val"+str(j+1)+"_RealPart")
```

```
        head_eigvalues.append("Eig-val"+str(j+1)+"_ImPart")
#head_eigvalues=["Eigenvalue"+str(j+1) for j in range(matrix_sensor_order)]
        #This just adds Eig1,Eig2,...,Eig-n
headers+=head_temps+head_eigvalues
headers=tuple(headers)
sheet.append(headers)
ser=serial.Serial('COM6',9600)
ser.flushInput()
distancia=0
distancias = []
distanciapatron = []
contador = 0
start_time = time.time()
temps=[]
ims=[]
MatrizTemps,aux2,aux3,aux4 =[],[],[],[]
det_matrix=0
heatmap=plt.cm.jet
time.sleep(0.010)
ser.flushInput()
fig, ax = plt.subplots( figsize =(8,8))
camera = Camera(fig)
while contador < sample_size:
        ser_bytes = ser.readline()
        extreme1 = len(ser_bytes)-2#Largo del byte codificado menos el resto de
            la codificacin
        decodeddata = ser_bytes[0:extreme1].decode("utf-8")#Decodifico desde
            UTF-8 (El que usa arduino)
        #val_flotante = float(decodificados)
        temps1=decodeddata.split(',')
        temps=[float(i) for i in temps1]
        #----------------------Ya tengo los datos en temp
            -------#
        MatrizTemps=[temps[4-1]]
        aux2=[temps[8-1]]
        aux3=[temps[12-1]]
        aux4=[temps[16-1]]
        PTAT=temps[17-1]
        estado_boton=temps[18-1]
        distancia=temps[19-1]
```

```
argfor=range(2,0-1,-1)#Es desde que k es igual a 0 hasta -1 (sin tocar el
    -1) con incrementos de -1
for k in argfor:
    MatrizTemps.append(temps[k+matrix_sensor_order*0])
    aux2.append(temps[k+matrix_sensor_order*1])
    aux3.append(temps[k+matrix_sensor_order*2])
    aux4.append(temps[k+matrix_sensor_order*3])
ArrangedTempsMatrix=np.matrix([MatrizTemps,aux2,aux3,aux4])
det_matrix=np.linalg.det(ArrangedTempsMatrix)#Computing Determinant
    of the ArrangedTempsMatrix
eig_values_matrix, eig_vectors_matrix=np.linalg.eig(ArrangedTempsMatrix)
eig_modulus=abs(eig_values_matrix)
eig_real=eig_values_matrix.real
eig_imag=eig_values_matrix.imag
print(ArrangedTempsMatrix)
MatrizFlatten=ArrangedTempsMatrix.flatten()
instante = datetime.datetime.now()
datos_para_excel=[MatrizFlatten.item(i) for i in range(
    matrix_sensor_order**2)]
#=str(datos_para_excel)[1:-1]
#array_to_append = [contador,datos_para_excel,instante,distancia,PTAT]
#Array to append es una tupla, le agrego lo que quiero meter, el contador
    , los datos y lo dems
array_to_append=(contador,instante,distancia,det_matrix,PTAT)
for j in datos_para_excel: array_to_append=array_to_append+(j,)#Voy
    creciendo la tupla aadiendo temps(elemento de cada rengln)
for j in range(matrix_sensor_order):
    array_to_append+=(str(eig_values_matrix[j]),)
    array_to_append+=(eig_modulus[j],)
    array_to_append+=(eig_real[j],)
    array_to_append+=(eig_imag[j],)
contador+=1
#--------------------Begins the graphic interface
    ----------------------#
headgraph='T[C]      '+'Sample #'+str(contador) +' '+' Ultrasonic-
    Distance: '+str(distancia)+'[cm]'+' Determinant: '#+str(det_matrix)
ax.text(0,-1,headgraph,va='center',ha='center')
min_val, max_val =5,45
intersection_matrix = ArrangedTempsMatrix
im=ax.matshow(intersection_matrix, cmap=heatmap,vmin=min_val,vmax=
    max_val)
```

```
    for i in range(matrix_sensor_order):
        for j in range(matrix_sensor_order):
            c = intersection_matrix[j,i]
            ax.text(i, j, str(c), va='center', ha='center')
    ax.text(0,matrix_sensor_order,'Date:  '+str(instante),va='center',ha='
        center')
    camera.snap()
    #ims.append([im])#To append each image as a single-element list into the
        ims list which contains frames
    #----------------------------Ends the graphic interface
        ------------#
    sheet.append(array_to_append) #Used to write on the excel file
    #det=np.linalg.det(matriztipoNumpy)
book.save(test_name+'.xlsx')
end_time = time.time()
totaltime_taken=end_time - start_time
print("Total time taken by this loop: ", totaltime_taken)
sample_time_secs=totaltime_taken/sample_size
#ani = animation.ArtistAnimation(fig2, ims, interval=sample_time_secs*1000,
    blit=True,repeat_delay=5000)
#plt.show()
animation_temps = camera.animate()
#animation.save('celluloid_legends.gif', writer = 'imagemagick')
print("Saving animation... Please wait for confimation")
animation_time_start=time.time()
animation_temps.save(test_name+'.gif', writer = 'pillow')
animation_time_end=time.time()
animation_time=animation_time_end-animation_time_start
ser.close()
print("All done.")
print("Time spent saving and creatin animation: ",animation_time)
#Eache time new run change GIF and XLSX files names --------
    IMPORTANT test_name
```

Apéndice C

Códigos de tratamiento de datos

C.1. Datos en la base de datos a tratamiento local

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 7 00:44:08 2021
UPDATE: OCTOBER 07, 2021
@author: Guillermo Martnez  Martnez
National Autonomous University of Mexico. School of Engineering.
Universidad Nacional Autnoma de  Mxico .  Facultad de  Ingeniera  .
Thesis Director:  Mtra. Rosa Itzel  Flores  Luna
Collaboration with Dr. Edmundo Gabriel Rocha Czatl
Student: Guillermo  Martnez  Martnez .      Mechatronics Engineering
Student ID: 313157886. Contact (Main): guillermo.martinezmartinez@outlook.
    com
Contact (Secundary): guillermo.martinez@comunidad.unam.mx
" Artificial  Intelligence  applied  to  the design  of  a device  to  register  a
    person's  number of contacts amid
COVID-19 context"
" Inteligencia  Artificial  aplicada en el  diseo  de un dispositivo  para
    registrar
 el  nmero  de contactos de una persona en el marco de COVID-19"

"""
import numpy as np
import pandas as pd
```

```
import copy
#Import databae module
#from firebase_admin import db
# pip install  firebase-admin
#pip install  Pyrebase
#Get a database reference to our posts.
#pip install  requests  --upgrade
import pyrebase
firebaseConfig  = {
    } #JSON de informacin sensible
firebaseConfig2  = {
} #JSON de informacin sensible
firebase  = pyrebase.initialize_app(firebaseConfig)
#ref = db.reference('https://environment-unam-mx-default-rtdb.firebaseio.
    com/')
db = firebase.database()
#Retrieve data
measure_1 = db.get()
mediciones = []
for measurements in measure_1.pyres:
    mediciones.append(measurements.item[1])
#print(mediciones)

#----------Creating DataFrame
#cont_df = []
aux1 = []
date_read = []
locinfo  = []
measuree = []
n_of_cough = []
n_of_sneeze  = []
time_read = []
for i in range(len(mediciones)):
    aux1.append(mediciones[i]["aux1"])
    date_read.append(mediciones[i]["date_read"])
    locinfo.append(mediciones[i]["locinfo"])
    measuree.append(mediciones[i]["measure"])
    n_of_cough.append(mediciones[i]["n_of_cough"])
    n_of_sneeze.append(mediciones[i]["n_of_sneeze"])
    time_read.append(mediciones[i]["time_read"])
df = pd.DataFrame()
```

```
df["VarAux1"] = aux1
df["Date_Measure"] = date_read
df["Location_Information"] = locinfo
df["Measure_Whole"] = measuree
df["N_of_cough"] = n_of_cough
df["N_of_sneeze"] = n_of_sneeze
df["Time_Measure"] = time_read

#df.to_pickle("./07_october_2021_home.pkl")
df = pd.read_pickle("./07_october_2021_home.pkl")

#-------------------- After that I might start sorting out the
    values ----------------------------#
string_measurements_raw = []
for i in range(df.shape[0]):
    string_measurements_raw.append(copy.deepcopy(df["Measure_Whole"][i]).
        strip("\n"))

smr_2 = []
for i in range(len(string_measurements_raw)):
    temp_element = (copy.deepcopy(string_measurements_raw[i]).split(","))
    if str(temp_element[-1]) == "," or not str(temp_element[-1]):
        temp_element.pop()
    else:
        pass #That means everthing is fine.
    try:
        casted_values = list(map(float, temp_element))
    except:
        casted_values = "ERROR"
        print("Failed at: {} when i is {} and its size is {}".format(
            temp_element,i, len(temp_element)))
        print("Trying to solve that problem...")
        if (not str(temp_element[0])):
            temp_element.pop(0)
            print("Solved!")
            print("That is now:{} when i is {} and its size is {}".format(
            temp_element,i, len(temp_element)))
            casted_values = list(map(float, temp_element))
        else:
            print("Couldn't solve ... ERROR2 will be shown.")
            casted_values = "ERROR2"
```

```
    smr_2.append(casted_values)


to_avoid_lose_string_measurements_raw = copy.deepcopy(
    string_measurements_raw)
#SPLIT BY COMMAS STRING_MEASUREMENTS_RAW

matrix_sensor_order = 4
sorted_measurements_matrix = []
boolean_contact = []
PTATs = []
distancias = []
flatten_measurements = []

for temps in smr_2:
    MatrizTemps=[temps[4-1]]
    aux2=[temps[8-1]]
    aux3=[temps[12-1]]
    aux4=[temps[16-1]]
    PTAT=temps[17-1]
    estado_boton=temps[18-1]
    distancia=temps[19-1]
    argfor=range(2,0-1,-1)#Es desde que k es igual a 0 hasta -1 (sin tocar el
        -1) con incrementos de -1
    for k in argfor:
        MatrizTemps.append(temps[k+matrix_sensor_order*0])
        aux2.append(temps[k+matrix_sensor_order*1])
        aux3.append(temps[k+matrix_sensor_order*2])
        aux4.append(temps[k+matrix_sensor_order*3])
    ArrangedTempsMatrix=np.array([MatrizTemps,aux2,aux3,aux4])
    #det_matrix=np.linalg.det(ArrangedTempsMatrix)#Computing
        Determinant of the ArrangedTempsMatrix
    #eig_values_matrix,eig_vectors_matrix=np.linalg.eig(
        ArrangedTempsMatrix)
    #eig_modulus=abs(eig_values_matrix)
    #eig_real=eig_values_matrix.real
    #eig_imag=eig_values_matrix.imag
    #print(ArrangedTempsMatrix)
    MatrizFlatten=ArrangedTempsMatrix.flatten()

    sorted_measurements_matrix.append(ArrangedTempsMatrix)
```

```
        boolean_contact.append(estado_boton)
        PTATs.append(PTAT)
        distancias.append(distancia)
        flatten_measurements.append(MatrizFlatten)

df["Distancia[cm]"] = distancias
df["Target_Contacto"] = boolean_contact

for i in range(len(flatten_measurements[0])):
            df["T"+str(i+1)+str("[C]")] = np.nan


for i in range(df.shape[0]):
    for k in range(len(flatten_measurements[0])):
            df["T"+str(k+1)+str("[C]")].iloc[i] = flatten_measurements[i][k]


#df.to_pickle("./oct_21_df_ready_to_ds_ml.pkl")

test_read = pd.read_pickle("./oct_21_df_ready_to_ds_ml.pkl")
#Ready a usar Jupyter Notebook
```

Apéndice D

Códigos de Machine Learning y de tratamiento de información geográfica

D.1. Notebook de Machine Learning

D.1.1. Inteligencia Artificial aplicada en el Diseño de un dispositivo para registrar el número de contactos de una persona en el marco de COVID-19

Author: Guillermo Martínez Martínez
National Autonomous University of Mexico. School of Engineering.
Universidad Nacional Autónoma de México. Facultad de Ingeniería.
Dissertation Director: Mtra. Rosa Itzel Flores Luna
Collaboration with Dr. Edmundo Gabriel Rocha Cózatl.
Student: Guillermo Martínez Martínez. Mechatronics Engineering
Student ID: 313157886. Contact (Main): guillermo.martinezmartinez@outlook.com
Contact (Secundary): guillermo.martinez@comunidad.unam.mx
"Artificial Intelligence applied in the design of a device to register a person's number of contacts among the ###### COVID-19 context"

```
import pandas as pd
import numpy as np
import copy
import ast
from pandas_profiling import ProfileReport
```

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#https://towardsdatascience.com/easy-steps-to-plot-geographic-data-on-
    a-map-python-11217859a2db
#https://towardsdatascience.com/exploratory-data-analysis-with-pandas-
    profiling-de3aae2ddff3
#https://towardsdatascience.com/geopandas-101-plot-any-data-with-a-
    latitude-and-longitude-on-a-map-98e01944b972
```

```
import sklearn.linear_model
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import TimeSeriesSplit
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian
    Process",
         "Decision Tree", "Random Forest", "Neural Net", "AdaBoost",
         "Naive Bayes", "QDA", "LogisticRegression"]

 classifiers  = [
    KNeighborsClassifier(3),
```

```
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    GaussianProcessClassifier(1.0 * RBF(1.0)),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1, max_iter=1000),
    AdaBoostClassifier(),
    GaussianNB(),
    QuadraticDiscriminantAnalysis(),
    LogisticRegression()]
```

```
#import descartes
```

```
#import geopandas as gpd
```

```
#!pip install geopandas
```

```
#!conda install -c conda-forge geopandas
```

```
#!pip install gdal
```

```
#"!pip install pipwin
#!pip install gdal
#pip install fiona
#pip install geopandasl
```

```
df = pd.read_pickle("./oct_21_df_ready_to_ds_ml.pkl")
#globalVar = Arrays.toString(arrayOf(latitud_measured_network,
    longitud_measured_network,loc_accuracy_network))
```

```
df
```

323 rows × 21 columns

```
df["Latitud"] = np.nan
df["Longitud"] = np.nan
df["Accuracy_Location"] = np.nan
for i in range(df.shape[0]):
    list_latitude_longitude_accuracy = ast.literal_eval(df["
        Location_Information"][i])
    df["Latitud"].iloc[i] = list_latitude_longitude_accuracy[0]
```

```
    df["Longitud"].iloc[i] = list_latitude_longitude_accuracy [1]
    df["Accuracy_Location"].iloc[i] = list_latitude_longitude_accuracy [2]
df
```

C:3-packages.py:1637: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy self._setitem_single_block(indexe value, name)

Date_Measure
Location_Information
Time_Measure
Distancia[cm]
Target_Contacto
T1[°C]
T2[°C]
T3[°C]
T4[°C]
T5[°C]
...
T10[°C]
T11[°C]
T12[°C]
T13[°C]
T14[°C]
T15[°C]
T16[°C]
Latitud
Longitud
Accuracy_Location
323 rows × 24 columns

```
df = df.drop('Location_Information', 1)
```

```
df
```

323 rows × 23 columns

D.1.1.1. DataFrame ready to be analyzed

```
#prof = ProfileReport(df)
#prof. to_file ( output_file ='EDA_dataset_V1.html')
```

D.1.1.2. Will now prepare DataFrame for Machine Learning Models.

```
curr_columns = df.columns.to_list ()
curr_columns
```

['Date_Measure', 'Time_Measure', 'Distancia[cm]', 'Target_Contacto', 'T1[°C]', 'T2[°C]', 'T3[°C]', 'T4[°C]', 'T5[°C]', 'T6[°C]', 'T7[°C]', 'T8[°C]', 'T9[°C]', 'T10[°C]', 'T11[°C]', 'T12[°C]', 'T13[°C]', 'T14[°C]', 'T15[°C]', 'T16[°C]', 'Latitud', 'Longitud', 'Accuracy_Location']

```
df_to_ml = copy.deepcopy(df[['Distancia[cm]',
 'Target_Contacto',
 'T1[ C ]',
 'T2[ C ]',
 'T3[ C ]',
 'T4[ C ]',
 'T5[ C ]',
 'T6[ C ]',
 'T7[ C ]',
 'T8[ C ]',
 'T9[ C ]',
 'T10[ C ]',
 'T11[ C ]',
 'T12[ C ]',
 'T13[ C ]',
 'T14[ C ]',
 'T15[ C ]',
 'T16[ C ]' ]])
df_to_ml
```

323 rows × 18 columns

```
y = copy.deepcopy(df_to_ml["Target_Contacto"])
df_to_ml.drop('Target_Contacto', axis=1, inplace=True)
X = df_to_ml
```

```
X
```

Figura D.1: png

323 rows × 17 columns

y

0 0.0 1 0.0 2 1.0 3 1.0 4 1.0 ... 318 1.0 319 1.0 320 1.0 321 1.0 322 1.0 Name: Target_Contacto, Length: 323, dtype: float64

```
X_train, X_test, Y_train, Y_test = train_test_split (
    X, y, test_size = 0.25, random_state = 42)
```

```
target_ok = Y_train.tolist().count(1)
no_target_ok = Y_train.tolist().count(0)
print("Target positivo = {}".format(target_ok))
print("Porcentaje de target positivo = {} %".format(target_ok/len(Y_train)
    *100))
print("No Target positivo = {}".format(no_target_ok))
print("Porcentaje de no target positivo = {} %".format(no_target_ok/len(
    Y_train)*100))
print("Total = {} ".format(len(Y_train)))
```

Target positivo = 78 Porcentaje de target positivo = 32.231404958677686 % No Target positivo = 164 Porcentaje de no target positivo = 67.76859504132231 % Total = 242

```
sns.histplot(data =pd.DataFrame(Y_train).astype(int), x="Target_Contacto")
plt.title("Target 0  y Target 1 ")
```

Text(0.5, 1.0, 'Target 0 y Target 1')

```
for i in range(len(classifiers)):
    clf = classifiers[i]
    print("----Currently working on model--- {}".format(names[i]))
    clf.fit(X_train, Y_train)
    score = clf.score(X_test,Y_test)*100
    print("Model accuracy: {} %".format(score))
    predictions = clf.predict(X_test)
```

```
y_true = np.ravel(Y_test)
y_pred = predictions
cm = confusion_matrix(y_true, y_pred)
print("Confusion_matrix is as follows: \n {}".format(cm))
target_names = ["No Contacto", "S Contacto"]
print( classification_report (y_true, y_pred, target_names=target_names))
```

—-Currently working on model— Nearest Neighbors Model accuracy: 97.53086419753086 % Confusion_matrix is as follows: [[50 2] [0 29]] precision recall f1-score support

No Contacto 1.00 0.96 0.98 52 Sí Contacto 0.94 1.00 0.97 29

accuracy 0.98 81 macro avg 0.97 0.98 0.97 81 weighted avg 0.98 0.98 0.98 81

—-Currently working on model— Linear SVM Model accuracy: 98.76543209876543 % Confusion_matrix is as follows: [[51 1] [0 29]] precision recall f1-score support

No Contacto 1.00 0.98 0.99 52 Sí Contacto 0.97 1.00 0.98 29

accuracy 0.99 81 macro avg 0.98 0.99 0.99 81 weighted avg 0.99 0.99 0.99 81

—-Currently working on model— RBF SVM Model accuracy: 64.19753086419753 % Confusion_matrix is as follows: [[52 0] [29 0]] precision recall f1-score support

No Contacto 0.64 1.00 0.78 52 Sí Contacto 0.00 0.00 0.00 29

accuracy 0.64 81 macro avg 0.32 0.50 0.39 81 weighted avg 0.41 0.64 0.50 81

—-Currently working on model— Gaussian Process

C:3-packages_classification.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use zero_division parameter to control this behavior. _warn_prf(average, modifier, msg_start, len(result))

Model accuracy: 98.76543209876543 % Confusion_matrix is as follows: [[51 1] [0 29]] precision recall f1-score support

No Contacto 1.00 0.98 0.99 52 Sí Contacto 0.97 1.00 0.98 29

accuracy 0.99 81 macro avg 0.98 0.99 0.99 81 weighted avg 0.99 0.99 0.99 81

—-Currently working on model— Decision Tree Model accuracy: 96.29629629629629 % Confusion_matrix is as follows: [[50 2] [1 28]] precision recall f1-score support

No Contacto 0.98 0.96 0.97 52 Sí Contacto 0.93 0.97 0.95 29

accuracy 0.96 81 macro avg 0.96 0.96 0.96 81 weighted avg 0.96 0.96 0.96 81

—-Currently working on model— Random Forest Model accuracy: 95.06172839506173 % Confusion_matrix is as follows: [[49 3] [1 28]] precision recall f1-score support

No Contacto 0.98 0.94 0.96 52 Sí Contacto 0.90 0.97 0.93 29

accuracy 0.95 81 macro avg 0.94 0.95 0.95 81 weighted avg 0.95 0.95 0.95 81

—-Currently working on model— Neural Net Model accuracy: 96.29629629629629 % Confusion_matrix is as follows: [[50 2] [1 28]] precision recall f1-score support

No Contacto 0.98 0.96 0.97 52 Sí Contacto 0.93 0.97 0.95 29

accuracy 0.96 81 macro avg 0.96 0.96 0.96 81 weighted avg 0.96 0.96 0.96 81

—-Currently working on model— AdaBoost Model accuracy: 98.76543209876543 %
Confusion_matrix is as follows: [[51 1] [0 29]] precision recall f1-score support
No Contacto 1.00 0.98 0.99 52 Sí Contacto 0.97 1.00 0.98 29
accuracy 0.99 81 macro avg 0.98 0.99 0.99 81 weighted avg 0.99 0.99 0.99 81
—-Currently working on model— Naive Bayes Model accuracy: 97.53086419753086 %
Confusion_matrix is as follows: [[51 1] [1 28]] precision recall f1-score support
No Contacto 0.98 0.98 0.98 52 Sí Contacto 0.97 0.97 0.97 29
accuracy 0.98 81 macro avg 0.97 0.97 0.97 81 weighted avg 0.98 0.98 0.98 81
—-Currently working on model— QDA Model accuracy: 98.76543209876543 %
Confusion_matrix is as follows: [[51 1] [0 29]] precision recall f1-score support
No Contacto 1.00 0.98 0.99 52 Sí Contacto 0.97 1.00 0.98 29
accuracy 0.99 81 macro avg 0.98 0.99 0.99 81 weighted avg 0.99 0.99 0.99 81
—-Currently working on model— LogisticRegression Model accuracy: 97.53086419753086 %
Confusion_matrix is as follows: [[51 1] [1 28]] precision recall f1-score support
No Contacto 0.98 0.98 0.98 52 Sí Contacto 0.97 0.97 0.97 29
accuracy 0.98 81 macro avg 0.97 0.97 0.97 81 weighted avg 0.98 0.98 0.98 81
C:3-packages_model_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1): STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in: https://scikit-learn.org/stable/modules/preprocessing.html Please also refer to the documentation for alternative solver options: https://scikit-learn.org/stable/modules/linear_model.h regression extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

D.1.1.3. Neural Network

Testing Neural Network Performance over this dataset

```
#For 3 hidden layers of, say, 100, 50, and 25 units respectively, it would be
    (1000,500,250,125,50)
hidden_layer_sizes_tuple = (1000,500,250,120,60,30,15,7)
clf2 = MLPClassifier(solver='lbfgs', alpha = 1e-5, hidden_layer_sizes =
    hidden_layer_sizes_tuple,
                         random_state = 1, max_iter = 10000)
clf2.fit(X_train, Y_train)
```

MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(1000, 500, 250, 120, 60, 30, 15, 7), learning_rate='constant', learning_rate_init=0.001, max_fun=15000, max_iter=10000, momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False)

```
NN_predictions = clf2.predict(X_test)
```

```
score = clf2.score(X_test,Y_test)*100
print("Model accuracy: {} %".format(score))
```

```
Model accuracy: 97.53086419753086 %
```

```
y_true = np.ravel(Y_test)
y_pred = NN_predictions
cm = confusion_matrix(y_true, y_pred)
```

```
cm
```

```
array([[50, 2], [ 0, 29]], dtype=int64)
```

D.1.1.4. Using Logistic Regression

```
clf
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None,
penalty='l2', random_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)
```

```
score = clf.score(X_test,Y_test)*100
print("Model accuracy: {} %".format(score))
predictions = clf.predict(X_test)
y_true = np.ravel(Y_test)
y_pred = predictions
cm = confusion_matrix(y_true, y_pred)
print("Confusion_matrix is as follows: \n {}".format(cm))
target_names = ["No Contacto", "S Contacto"]
print( classification_report (y_true, y_pred, target_names=target_names))
```

```
Model accuracy: 97.53086419753086 % Confusion_matrix is as follows: [[51 1]
[ 1 28]] precision recall f1-score support
No Contacto 0.98 0.98 0.98 52 Sí Contacto 0.97 0.97 0.97 29
accuracy 0.98 81 macro avg 0.97 0.97 0.97 81 weighted avg 0.98 0.98 0.98 81
```

```
weight_matrix = clf.coef_
weight_matrix
```

```
array([[-0.01096605, -1.50103241, -0.26674058, 0.16814001, -0.46744182, -0.2942094
, 0.34503892, 1.79838073, -0.24771069, -0.34957081, -0.52353762, 0.03664752, -
1.50774391, 0.92828383, 0.9455384 , 0.26256895, 0.57139162]])
```

```
weight_matrix.shape
```

```
(1, 17)
```

```
bias_matrix = clf.intercept_
bias_matrix
```

```
array([-0.63588659])
```

```
clf.get_params()
```

{'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': None, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}

D.1.1.5. Saving and exporting model

```
#import pickle
#filename = "model_dec_2021.sav"
#pickle.dump(clf, open(filename, "wb"))
```

```
# save the model to disk
#filename = 'finalized_model.sav'
#pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
#loaded_model = pickle.load(open(filename, 'rb'))
#result = loaded_model.score(X_test, Y_test)
#print(result)
```

```
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

```
0.9753086419753086
```

```
loaded_model
```

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True, intercept_scaling=1, l1_ratio=None, max_iter=100, multi_class='auto', n_jobs=None, penalty='l2', random_state=None, solver='lbfgs', tol=0.0001, verbose=0, warm_start=False)

```
jupyter nbconvert --to markdown DataScience_and_Model_Nice.ipynb
```

```
pandoc --listings -f markdown -t latex test.md -o test.tex
```

D.2. Notebook de análisis de ubicación geográfica

```
import pandas as pd
import matplotlib.pyplot as plt
import descartes
import geopandas as gpd
from shapely.geometry import Point, Polygon
import copy
%matplotlib inline
```

```
df = pd.read_pickle("./ df_to_location_analysis .pkl")
```

```
df
```

Date_Measure
Time_Measure
Latitud
Longitud
Accuracy_Location
0
2021-09-26
20:12:33.099
20.669061
-103.390681
800.000
1
2021-09-26
20:12:39.448
20.669061
-103.390681
800.000
71 rows × 5 columns

```
df.rename(columns={'Latitud':'Latitude'}, inplace=True)
```

```
df.rename(columns={"Longitud":"Longitude"}, inplace=True)
```

```
df
```

```
Date_Measure
    Time_Measure
    Latitude
    Longitude
    Accuracy_Location
    0
    2021-09-26
    20:12:33.099
    20.669061
    -103.390681
    800.000
    1
    2021-09-26
    20:12:39.448
    20.669061
    -103.390681
    800.000
    2
    2021-09-26
    20:12:41.868
    20.669061
    -103.390681
    800.000
    3
    2021-09-26
    20:12:45.485
    20.669061
    -103.390681
    800.000
    71 rows × 5 columns
```

```
street_map = gpd.read_file("./jal_entidad.shp")
```

```
#fig, ax = plt.subplots( figsize  = (15,15))
#street_map.plot(ax = ax)
```

```
geometry = [Point(xy) for xy in zip (df["Longitude"], df["Latitude"])]
```

```
geometry[:3]
```

```
[<shapely.geometry.point.Point at 0x1da0aa295e0>,
 <shapely.geometry.point.Point at 0x1da0aa29d60>,
 <shapely.geometry.point.Point at 0x1da0aa29820>]
```

```
#geometry
```

```
len(geometry)
```

```
71
```

```
df["geometry"]=geometry
df
```

```
    POINT (-103.3906808 20.6690606)
    70
    2021-09-26
    20:12:23.984
    20.669061
    -103.390681
    800.000
    POINT (-103.3906808 20.6690606)
    71 rows × 6 columns
```

```
df["measure"] = "Measurement"
```

```
df
```

```
    71 rows × 6 columns
```

```
geo_df = gpd.GeoDataFrame(df,
                          crs="EPSG:4326",
                          geometry = geometry)
```

```
fig , ax = plt.subplots( figsize =(15, 15))
street_map.plot(ax = ax, alpha = 0.4, color = "grey")
```

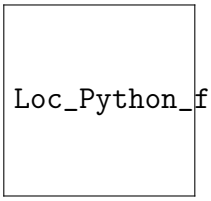


Figura D.2: png

```
geo_df[geo_df["measure"] == "Measurement"].plot(ax = ax, markersize = 20,
    color = "blue", marker = "o", label = "Mediciones en la Ciudad de
    Guadalajara")
plt.legend(prop = {"size":15})
```

```
<matplotlib.legend.Legend at 0x1da0b307040>
```