DESARROLLO DE UN PROTOTIPO DE MÁSCARA FACIAL ELECTRÓNICA PARA EL MONITOREO DE LA TEMPERATURA CORPORAL, RESPIRACIONES POR MINUTO Y SATURACIÓN DE OXÍGENO EN LA SANGRE DE PACIENTES EN OBSERVACIÓN MÉDICA DOMICILIARIA

Autor

CIRO ALFONSO PALLARES FRAGOZO

Director

VICTOR JULIO VARGAS SARMIENTO

MSc. Energías Renovables Y Sostenibilidad Energética

Ingeniería Electrónica

Departamento De Electrónica, Eléctrica, Sistemas Y Telecomunicaciones

Facultad De Ingenierías Y Arquitectura

Universidad De Pamplona

Pamplona, noviembre 2020

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES
PROGRAMA DE INGENIERÍA ELECTRÓNICA
TRABAJO PRESENTADO PARA OPTAR POR EL TITULO DE INGENIERO ELECTRÓNICO

TEMA:

DESARROLLO DE UN PROTOTIPO DE MÁSCARA FACIAL ELECTRÓNICA PARA EL MONITOREO DE LA TEMPERATURA CORPORAL, RESPIRACIONES POR MINUTO Y SATURACIÓN DE OXÍGENO EN LA SANGRE DE PACIENTES EN OBSERVACIÓN MÉDICA DOMICILIARIA

FECHA DE INICIO DEL TRABAJO: 19/09/2020
FECHA DE TERMINACIÓN DEL TRABAJO: 17/12/2020

NOMBRES Y FIRMAS DE AUTORIZACIÓN PARA LA SUSTENTACIÓN:

Ciro Alfonso Pallares Fragozo

Victor Julio Vargas Sarmiento
DIRECTOR

José Daniel Ramirez Corzo
DIRECTOR DE PROGRAMA

JURADO CALIFICADOR:

Victor Julio Vargas Sarmiento

José Daniel Ramirez Corzo

Andrés Orlando Páez Melo

PAMPLONA NORTE DE SANTANDER
COLOMBIA
14/12/2020

A mi madre Martha Noemi Fragozo Castilla, por su infinita paciencia y amor por mí, a mi director de trabajo de grado que me ha acompañado en la realización de este proyecto, a mi amiga Heidy Grisel Varela Avilés quien me apoyó desde el inicio hasta la culminación, a todas las personas que de una u otra manera han contribuido con mi formación como persona, a aquellos docentes que se han convertido en eje de apoyo para construir mis bases como un profesional, a la Universidad de Pamplona por darme tantos buenos momentos, mil gracias siempre los recordaré.

CONTENIDO

Tabla de figuras.

Tablas

Tabla de ecuaciones

RESUMEN

Se realizó un sistema de monitoreo electrónico en base a una mascarilla facial electrónica que por medio de seguimiento permite recolectar información acerca de parámetros de oxigenación en la sangre, frecuencia respiratoria y temperatura corporal, este prototipo de mascarilla multisensorial está pensado para el uso de pacientes en observación médica domiciliaria, con intervalos de medida correspondientes a la evaluación médica necesaria consultada a un especialista del sector de la salud. El prototipo aquí desarrollado tiene un sistema embebido Arduino Mega que es capaz de recibir los datos que le envían los sensores por el protocolo I2C y el puerto analógico para posteriormente enviarlo hacia una placa NodeMCU que es capaz de enviar la información recibida, a la nube en firebase para posteriormente visualizarla en un aplicativo móvil, obteniendo así, resultados equiparables a equipos comerciales de bajo costo y validado con un profesional de la salud, además al concluir este prototipo se abren puertas hacia nuevos desarrollos de este tipo de aplicaciones de la salud, posiblemente abriendo tendencia hacia el uso de dispositivos portátiles para diferentes mediciones importantes en materia de diagnóstico o prevención de enfermedades.

1. INTRODUCCIÓN.

1.1 PROBLEMA.

Debido a la situación actual en el mundo, el uso de mascarillas para el uso diario se ha convertido un tema común, podemos usar diferentes definiciones para referirnos a ellas tal como tapaboca(s), cubre boca(s), mascarilla de uso general no hospitalario, podemos afirmar que la demanda de este tipo de elemento ha crecido exponencialmente este año por la situación actual. En Colombia como en muchas partes del mundo, el precio de dichos elementos se ha incrementado considerablemente a pesar de que el presidente de la republica indicó que no se debían subir de precio dichos implementos como algunas otras cosas, pudimos notar que la práctica fue totalmente diferente. No solo se ha incrementado el uso de mascarillas, también ha habido un aumento en el número de pruebas que se han procesado para el virus del COVID-19, lo que ha provocado un traumatismo en los tiempos de toma de muestras, su procesamiento y la entrega de los resultados, teniendo de diferencia entre las fechas de inicio de síntomas y la de diagnóstico en promedio más de 10 días, dejando al paciente sin revisión en el lapso de un tiempo. [1]

1.2 JUSTIFICACIÓN.

Al desarrollarse este prototipo de máscara monitorizadora, se podrá tener acceso de forma económica y de fácil a un producto que pueda solventar uno de los problemas que se han presentado en todo el mundo por la situación actual, es decir, la monitorización de los signos vitales como: la temperatura, la oxigenación en la sangre y las respiraciones por minuto, sin ayuda de terceros, por ende, se desarrolló una solución ingenieril. La mascarilla electrónica incorpora sensores los cuales pueden transmitir los signos vitales del paciente hacia un microcontrolador de la familia Arduino, con esta información el paciente no deberá ocupar un espacio en el hospital, debido a que no se encontrará fuera de monitoreo con este sistema. Cabe aclarar que, aunque el proyecto puede estar dirigido a cualquier público, el uso en el personal de salud es muy importante, debido a que son estos profesionales quienes están presentes siempre con los posibles contagiados. Estos parámetros pueden orientar al médico a evidenciar si el paciente está o no está contagiado.

1.3 OBJETIVOS.

1.3.1 OBJETIVO GENERAL.

Desarrollar un prototipo de máscara facial electrónica para el monitoreo de la temperatura corporal, respiraciones por minuto y saturación de oxígeno en la sangre de pacientes que se encuentran en observación médica domiciliaria.

1.3.2 OBJETIVO ESPECIFICO.

1. Diseñar e implementar un sistema electrónico que se adapte a la mascarilla y permita monitorear la temperatura corporal, saturación de oxígeno en la sangre y respiraciones por minuto.
2. Diseñar, implementar y programar en un sistema embebido en el cual se registren los parámetros monitoreados y se comunique con un dispositivo de supervisión.
3. Programar el sistema de supervisión para evaluar la salud del paciente y generar recomendaciones.
4. Diseñar e imprimir una mascarilla facial en 3d que permita acoplar el sistema electrónico de monitoreo.
5. Acoplar la mascarilla, el sistema electrónico y el sistema embebido y verificar su correcto funcionamiento.
6. Hacer pruebas de validación del dispositivo apoyándose en personal médico especializado.

1.4 ANTECEDENTES.

Haciendo una revisión bibliográfica acerca de los proyectos realizados anteriormente con respecto al proyecto que aquí se presenta, encontramos algunos tipos de artículos científicos como el oxímetro experimental desarrollado por Eduardo M.G. Rodrigues a, Radu Godinaa, Carlos M.P. Cabrita b y João P.S. Catalão expuesto a continuación.

1.4.1 OXÍMETRO EXPERIMENTAL DE TIPO REFLECTANTE DE BAJO COSTO PARA SISTEMAS DE SALUD PORTÁTILES.

El advenimiento de la tecnología portátil es fundamental para la difusión dispositivos de salud de monitoreo portátil personal. Los desarrollos recientes de sensores biomédicos han disminuido el factor de forma y consumo de energía que se puede usar de forma permanente. Este artículo analiza un método reflectante de bajo costo. Un sistema de fotopletismografía (PPG) que utiliza una solución de circuito integrado dedicado (IC) como núcleo de un dispositivo portátil de control de la salud. Se realiza la medición de dos indicadores fisiológicos, a saber, la frecuencia del pulso (FC) y la saturación de oxígeno en sangre (SpO2). El trabajo analiza en profundidad la arquitectura de detección de señales PPG, que garantiza mediciones de alta resolución gracias a una unidad de conversión de analógico- digital delta-sigma. Las operaciones de filtrado digital de posprocesamiento se implementan para mejorar adquisición de ruido PPG para la extracción de señales fisiológicas. Se presenta un diseño completo del sistema y se realiza una evaluación detallada en un escenario de procesamiento en tiempo real. La plataforma de prueba se completa con una aplicación de gráficos basada en PC para el análisis de datos en línea y fuera de línea. Minimizar la disipación de energía es el principal desafío en un diseño portátil. Sin embargo, restringe la sensibilidad de la medición de la señal PPG al reducir la calidad de la señal. Utilizando el prototipo de consumo de energía desarrollado, se realizan estudios sobre la caracterización del consumo de energía y la calidad de la señal en diversas condiciones de trabajo. A continuación, se propone una figura de mérito de desempeño como principal contribución a la investigación, que aborda el tema de compromiso de consumo de energía y calidad

de la señal. Tiene como objetivo ser utilizado como un análisis de compensaciones entre estos dos criterios de diseño en conflicto. [2]

O la máscara impresa en 3D para situaciones de pandemia desarrollado por Gwen R. J. Swennena, Lies Pottelb y Piet E. Haers referenciada a continuación.

1.4.2 MÁSCARA IMPRESA EN 3D A MEDIDA, EN PANDEMIA EN CASO DE SITUACIONES DE CRISIS CON CARENCIA DE MÁSCARAS FFP2/3 DISPONIBLES COMERCIALES.

En el caso de situaciones de crisis pandémica, una falta crucial de material de protección como las mascarillas protectoras para los profesionales sanitarios. (PoC) y el prototipo se presenta, demostrando una máscara facial impresa tridimensionalmente (3D) reutilizable hecha a medida basada en materiales y técnicas (imágenes 3D e impresión 3D) con disponibilidad global. La mascarilla protectora 3D individualizada consta de dos componentes compuestos de poliamida reutilizables impresos en 3D (una mascarilla y un soporte de membrana de filtro) y dos componentes desechables (una banda de fijación de la cabeza y una membrana de filtro). Se utilizó diseño asistido por computadora (CAD) para producir las componentes reutilizables de la máscara facial 3D basados en exploraciones faciales individuales, adquirido utilizando un teléfono inteligente de nueva generación con dos cámaras y un escaneo facial solicitud. El modelado 3D puede ser realizado fácilmente por diseñadores CAD en todo el mundo con descargar software La membrana de filtro de soplado fundido no tejido desechable disponible de fabricantes industriales que producen máscaras protectoras FFP2 / 3 para pintura, construcción, agricultura e industria textil. Velcro fácilmente disponible Los sujetadores se utilizaron como una banda de fijación de cabeza desechable. Una limpieza y desinfección se propone el protocolo. Pruebas virológicas y de fugas de los componentes reutilizables de La mascarilla 3D, después de uno o varios ciclos de desinfección, aún no se ha realizado y es esencial antes de su uso en situaciones de la vida real. Esta PoC debe permitir al lector considerar hacer y / o probar virológicamente la descripción de mascarillas faciales impresas en 3D y hechas a medida en todo el mundo. El formato de lenguaje de teselación de superficie (STL) del original. [3]

Como también podemos encontrar desarrollos financiados por páginas de donaciones y desarrollos colombianos, todos ellos, enfocados en la protección y el uso de mascarillas fáciles como podremos ver a continuación con estos cuatro (4) tipos de mascarillas.

1.4.3 C-MASK.

De la empresa japonesa Donut Robotics, esta mascarilla es capaz de traducir hasta 8 idiomas diferentes, según sus desarrolladores, el sonido de esta mascara puede llegar hasta 10m hacia adelante, puede conectarse a una red de internet para transmitir y traducir los mensajes, la C-MASK es de plástico y se adapta a cualquier tipo de mascarilla convencional, se conecta a una app mediante Bluetooth, puede utilizar un micrófono incluido en el diseño para poder transcribir el habla a texto, realizar llamadas o amplificar la voz del usuario, la mascarilla tendrá un costo entre los 40 y 50 dólares americanos.[4]

1.4.4 LEAF MASK.

Con un diseño y una tecnología con patente pendiente, perteneciente a Redcliffe Medical Devices Inc. Leaf ™, es una máscara que cuenta con un recubrimiento hidrofóbico, que permite una superficie libre de líquidos, sus desarrolladores certifican que puede destruir el 99.99% de los microbios. Tiene un nano recubrimiento permanente que sobrevive a lavados regulares de agua o agua con jabón suave, para poder encontrarla actualmente se debe rellenar un formulario de inscripción a una lista de espera, pero en indiegogo al financiar el proyecto se conseguía una mascarilla en 49 dólares americanos.[5]

1.4.5 CIVILITY.

“Nueva generación de máscaras transparentes” es un proyecto en demanda actual en indiegogo, cuenta con dos mini ventiladores eléctricos, silenciosos según sus creadores, estos ventiladores aumentan la transpirabilidad dentro de la máscara y aumentan la extracción de aire hacia fuera de la misma, estos mini ventiladores tienen 8h de autonomía y el proyecto en indiegogo cuenta con packs de compra desde los 35 euros hasta los 7500 euros.[6]

1.4.6 EASYCO.

Es una máscara de respiración en actual desarrollo por 5 emprendedores colombianos, se inspiraron en las máscaras Easybreath fabricadas por la marca deportiva Decathlon y las transformaron en dispositivos que según ellos, harán más seguro el trabajo de los médicos y enfermeras, la máscara EasyCO, está compuesta por una máscara de snorkel, un acople en impresión 3D y un filtro intersurgical, los filtros intersurgical proporcionan una efectiva barrera ante el paso del coronavirus responsable del SARS en los circuitos respiratorios.[7]

2. MARCO TEÓRICO

En este capítulo se van a tratar los temas correspondientes a el diseño de una máscara facial multisensorial, además de la información relevante de sus componentes y de la plataforma en la cual se hará monitoreo en tiempo real de las variables de oxigenación, frecuencia respiratoria y temperatura corporal.

2.1 DISEÑO E IMPRESIÓN 3D.

En la actualidad, la impresión 3D ha tomado un gran auge en muchos campos de la ingeniería siendo parte de la industria 4.0, podemos imprimir casi cualquier cosa hoy en día, por ende, podemos encontrar adelantos en impresiones 3D que de una u otra forma están en contacto con la piel, como las máscaras de disfraces utilizadas en múltiples propósitos y otras más versátiles como las de gases. [8][9]

La impresión 3D tiene ventajas frente a otras tecnologías por su fácil reproducción, algunos investigadores han aprovechado el potencial de esta tecnología para crear máscaras que pueden hacer frente a la escasez de mascarillas por parte de la pandemia actual, logrando establecer una buena alternativa y abriendo la puerta a nuevas maneras de enfrentarla, algunos desarrolladores se han atrevido a más y han incorporado diferentes características a dichas máscaras, como en el caso de CLIU, esta es una máscara que permite la desinfección automatiza, permitiendo extender la vida útil de dicho elemento, dando la posibilidad de reutilización y desinfección de algunas máscaras por diferentes métodos, remojándolas en agua a una temperatura superior a 56C durante 30 minutos recomendado por la Comisión Nacional de Salud de la República Popular de China.[10][11[12]

El desarrollo de prototipos 3D es una técnica innovadora actualmente, nos permite darles vida a modelos en un entorno virtual y materializarlos en un entorno físico, hay diversos programas entorno al desarrollo de diseños 3D, entre los cuales cabe resaltar,

SolidWorks y autocad, que disponen diferentes herramientas para diseño 3D y entre ellas existen plataformas en la nube gratuitas como es el caso de TinkerCad, un programa de extensión de la empresa autocad.

2.2 TEMPERATURA CORPORAL.

La temperatura es un requisito previo básico para todas las formas de vida como también una dimensión física. Las características físicas de la materia dependen de la temperatura. Inclusive pequeñas variaciones pueden enmarcar cambios considerables, como los que pueden ser observados durante la transformación de un estado de agregación a otro. El metabolismo de los organismos vivos puede ser afectado en gran medida por la temperatura. Las tasas metabólicas pueden ser modificadas por temperaturas excesivamente bajas o excesivamente altas como también pueden modificar, alterar la función de los órganos y provocar daños en los tejidos. Es decir, la temperatura del entorno y la temperatura corporal adquieren una importancia fundamental para la vida y la salud.

Temperatura corporal es una medida relativa de calor que se le asocia al metabolismo del cuerpo humano, esta medida tiene gran importancia en la relación de la anestesia y termorregulación. La termorregulación es una de las funciones más importantes de cualquier organismo, el organismo humano en buenas condiciones de salud es capaz de regular la temperatura corporal central con una diferencia de +/- 0,2°C con respecto al valor normal. [13]

En la figura 1 se puede observar la diferencia de temperatura que presenta el cuerpo, dicha imagen se puede tomar como mapa de temperatura externa del cuerpo para poder hacer mediciones corporales diferentes a las convencionales (oral o rectal), en este mapa se puede notar que la zona de la cabeza (rostro) es la más viable para tomar una referencia de infecciones con respecto a temperatura elevada en el cuerpo.

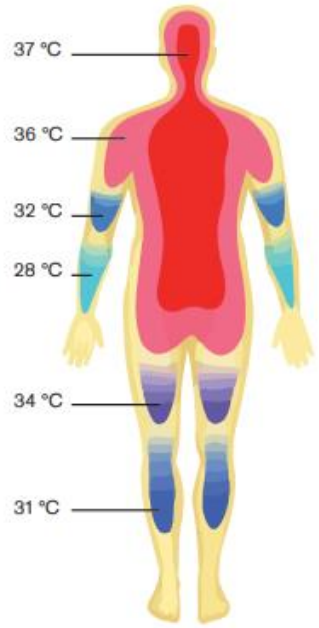


Fig. 1. Parte central del cuerpo y periferia.

Fuente: *Drägerwerk AG & Co. KGaA [13].*

2.2.1 TEMPERATURA CORPORAL CENTRAL.

Aunque la temperatura de las extremidades y de las regiones periféricas del cuerpo varían según las condiciones medioambientales, la temperatura de la parte central del cuerpo, que se compone de órganos internos, tejidos profundos y el cerebro, se mantiene bajo condiciones normales prácticamente constante. La zona termoneutral es el rango de temperaturas externas a las que la temperatura corporal central puede mantenerse sin la ayuda de sudoración o temblores y abarca un rango de entre 27 y 32°C para una persona adulta sin ropa en reposo. [13]

La temperatura corporal puede ser medida de diferentes maneras, tanto con contacto como sin contacto físico, existen diferentes tipos de sensores que se encargan de esto, como lo hace el MLX90614, siendo un sensor infrarrojo que puede medir temperatura sin entrar en contacto con la piel.

MLX90614.

Fig. 2. Circuito MLX90616 vista de frontal.

Fuente: *Autor.*

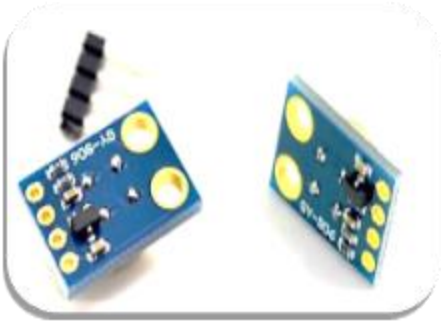


Fig. 3. Circuito MLX90616 vista de posterior.

Fuente: *Autor.*

En las figuras 2 y 3 se puede ver el sensor El MLX90614 por la parte superior y posterior respectivamente, dicho sensor está construido a partir de 2 chips desarrollados y fabricados por Melexis: el primero es el detector de termopila infrarroja MLX81101 y el segundo es el acondicionador de señal ASSP MLX90302, especialmente diseñado para procesar la salida del sensor IR.

SUBSISTEMA MLX90614.

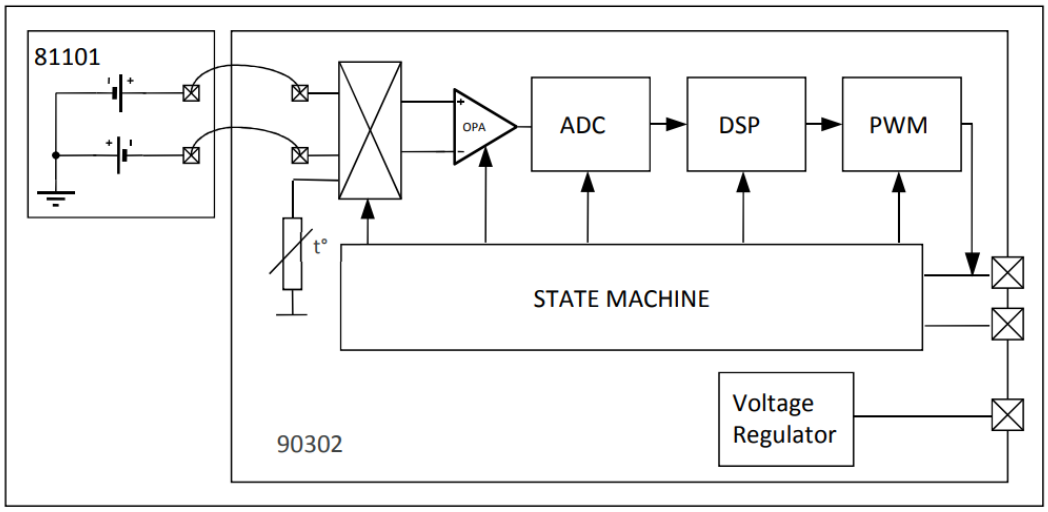


Fig. 4. Subsistemas en bloques MLX90614

Fuente: *Hoja de datos.*

PRINCIPIO DE PROCESAMIENTO DE SEÑALES.

En la figura 4 se puede apreciar los componentes internos del sensor MLX90614; el funcionamiento del MLX90614 está controlado por una máquina de estado interna, que cumple la función de controlar las medidas y cálculos de las temperaturas del objeto y la temperatura ambiental y realizar un posprocesamiento de las temperaturas para generarlas a través de la salida PWM o la interfaz compatible con SMBus. El ASSP admite 2 sensores de IR (el segundo no está implementado en el MLX90614xAx). La salida de los sensores IR se amplifica por un amplificador chopper de baja compensación y bajo ruido con ganancia programable, convertido por un modulador Sigma Delta a un solo flujo de bits y alimentado a un potente DSP para su posterior procesamiento. La señal es tratada por filtros de paso bajo FIR e IIR programables (por medio de la EEPROM que contiene) para una mayor reducción del ancho de banda de la señal de entrada, para lograr el rendimiento de ruido y la frecuencia de actualización deseados. La salida del filtro IIR es el resultado de la medición y está disponible en la RAM interna. Hay 3 celdas diferentes disponibles: una para el sensor de temperatura integrado y 2 para los sensores IR.

PRECISIÓN DE TEMPERATURA DEL MLX90614.

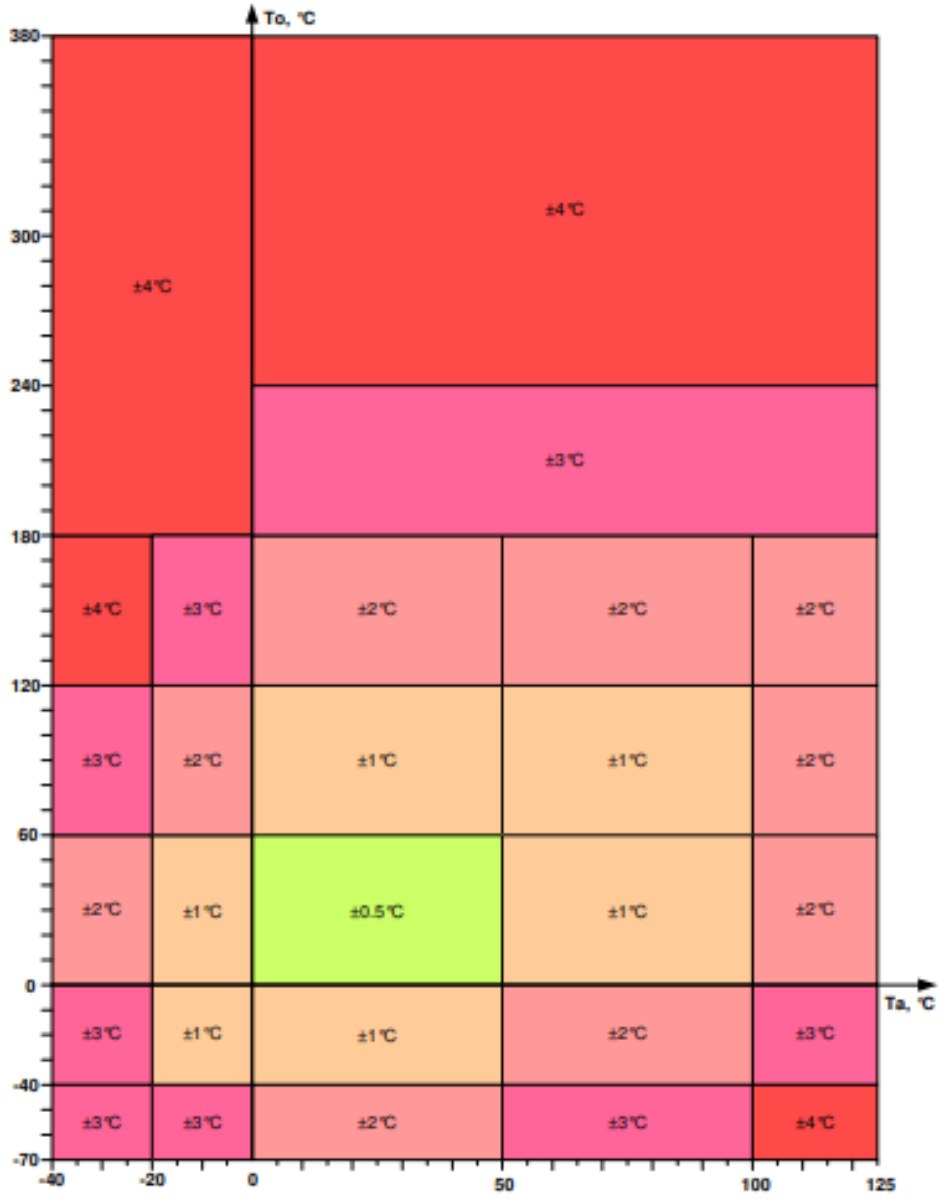


Fig. 5. Precisión de MLX90614 (Ta, To).

Fuente: *Hoja de datos.*

En la figura 5 se muestra la presión del MLX90614 con respecto a sus dos medidas principales Ta (temperatura ambiente) y To (temperatura de objeto apuntado). El rango

con menor error se encuentra entre 0° y 60° C, es decir, cuando la temperatura disminuye de 0 o aumenta de 60, el error aumenta.

TERMOMETRO MLX90614.

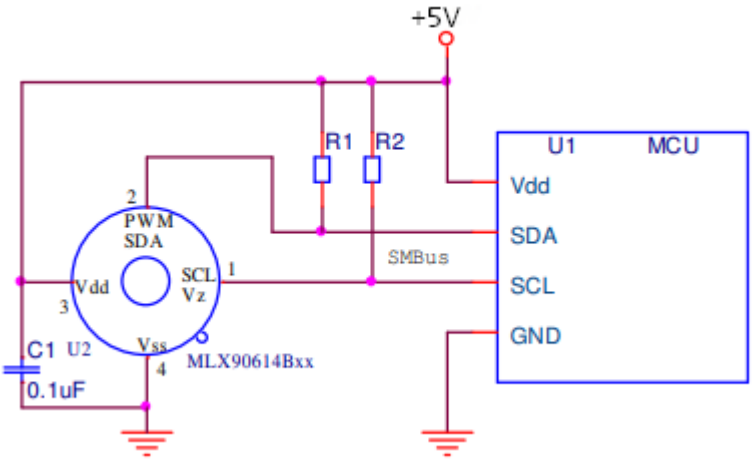


Fig. 6. Circuito del MLX90614.

Fuente: *Hoja de datos.*

El esquema de la figura 6 es el que podemos encontrar en el encapsulado de la figura 3 y debido a sus prestaciones antes mencionadas y su compatibilidad con el bus I2C lo utilizaremos con la librería "Adafruit_MLX90614_Library" mediante el llamado de la función "`Ob=mlx.readObjectTempC();`" para obtener la temperatura del objeto (rostro de la persona en este caso) (Más información en el anexo 1)

2.3 SATURACIÓN DE OXÍGENO EN LA SANGRE.

Para que todos los órganos y células del cuerpo humano funcionen normalmente, el oxígeno es esencial, el oxígeno entra en los alvéolos desde el aire circundante y entra en contacto con el sistema circulatorio e intercambia gases. El oxígeno se transfiere a la sangre y parte de él permanece en forma de oxígeno disuelto (O2, PaO2 en la sangre arterial), pero la mayor parte es absorbida por la hemoglobina y luego transportada a los

tejidos. Un gramo de hemoglobina transporta 1,36 ml de oxígeno (hemoglobina saturada al 100% (SaO2)). La cantidad de oxígeno entregado al tejido (contenido de oxígeno en sangre arterial, CaO2) depende principalmente de la cantidad de hemoglobina y su saturación en oxígeno (CaO2 = (Hgb x 1.36 x SaO2), mientras que la proporción de oxígeno disuelto es menor + (PaO2 x 0,0031)). Cuando se habla de saturación de oxígeno arterial (SaO2), se refiere a la proporción (%) de proteína de oxígeno en sangre en el volumen sanguíneo total. Cuando la saturación de oxígeno es más alta, la hemoglobina se vuelve roja brillante. La pulsioximetría, basa en la diferencia en la absorción de ondas de luz por la hemoglobina oxigenada y desoxigenada. [14]

VALORES NORMALES EN PERSONAS SIN AFECCIÓN RESPIRATORIA.

De acuerdo con la absorción de ondas de luz, el valor promedio de mediciones repetidas durante un período de tiempo se usa para calcular la saturación de oxígeno arterial (SaO2) de la hemoglobina pulsante (arterial). El resultado obtenido es el porcentaje de saturación de oxígeno de la hemoglobina (SaO2) en la sangre arterial porque se mide con un pulsioxímetro (SpO2), además de la cantidad de pulsos por minuto que pueden indicar la frecuencia. [14]

El oxígeno disuelto normal es absorbido por la hemoglobina, y la relación entre la presión parcial arterial de oxígeno y la saturación de oxígeno de la hemoglobina (la afinidad de la hemoglobina por el oxígeno) se describe en la curva de disociación de la hemoglobina sigmoidea en la figura 7.

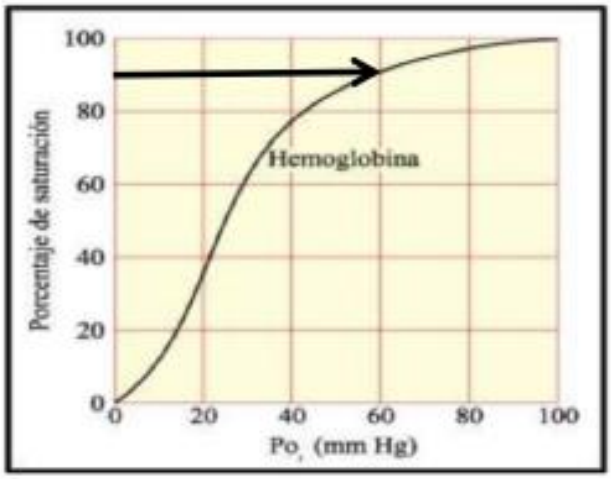


Fig. 7.Curva de Disociación de la Hemoglobina.

Fuente: *N. C. Dario Trujillo. MD, Rafael Colmenraes, Audery Piotrostanalky [14].*

Inicialmente, esta relación es inclinada. Un pequeño aumento de la presión de oxígeno aumentará significativamente la SaO2. Cuando la PaO2 alcanza unos 60 mmHg, la curva pierde su inclinación y se vuelve más horizontal. Cuando la PaO2 alcanza los 70 mmHg (la SaO2 representa aproximadamente el 90-92%), la curva se vuelve más plana y casi horizontal y, a pesar del aumento de la presión de oxígeno, si la hemoglobina alcanza el 100% de saturación (PaO2 de aproximadamente 100 mmHg), hay poco o ningún aumento de SaO2. Mirando la curva desde la dirección opuesta, la presión arterial de oxígeno puede De más de 100 mmHg a 65-70 mmHg y SaO2 de 100 a 90%. Cuando la PaO2 cae por debajo de 70 mmHg, especialmente por debajo de 60 mmHg, Un pequeño cambio en la PaO2 conducirá a una gran caída en la SaO2, por lo que Contenido de oxígeno en sangre arterial. La afinidad de la hemoglobina por el oxígeno y, por tanto, la relación entre La PaO2 y la SaO2 se ven afectadas por muchos factores, principalmente Temperatura, presión arterial de dióxido de carbono (PaCO2) y Concentración de iones de hidrógeno (pH). Agregue cualquiera de estos Los factores reducirán la afinidad de la Hb por el O2, lo que cambiará la curva Derecha y abajo, se necesita más presión de oxígeno para alcanzar la misma SaO2. PaO2 y SaO2 aumentan al aumentar la altitud A nivel del mar, la SaO2 normal se encuentra entre el 95% y el 100% como se muestra en la tabla 1.[14]

Altitud sobre nivel del mar	SaO2 % Hombres Media (IC95%)	SaO2% Mujeres Media (IC95%)
970mt	94,8 (94,1-95,4)	96,4 (95,7 – 97,1)
1520	95,5 (94,9-96,1)	95,6 (94,9 – 96,2)
1728	95,7 (95,3-96,2)	96,1 (95,6 – 96,6)
1923	95,1 (94,3-95,8)	96 (95,6-96,3)
2180	95,2 (94,6-95,9)	95,4 (94,9-95,9)
2600	93,6 (93,2-94)	94,4 (94,1-94,8)

Tabla 1 Saturación arterial de oxígeno en diferentes alturas en población sana en Colombia.

Fuente: *N. C. Dario Trujillo. MD, Rafael Colmenraes, Audery Piotrostanalky [14].*

En las aplicaciones biomédicas es común encontrar diferentes tipos de sensores que pueden ser empleados para medir magnitudes relevantes, como los sensores empleados en oximetría, un claro ejemplo de ellos es el sensor Max30102.

2.3.1 MAX 30102

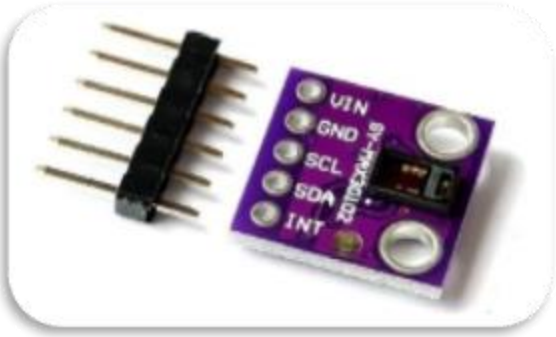


Fig. 8. Sensor MAX30102 vista frontal.

Fuente: *Autor.*

Fig. 9. MAX30102 vista trasera.

Fuente: *Autor.*

En las figuras 8 y 9 se puede ver el sensor MAX30102, este es un sensor del fabricante Maxim Integrated con suficientes prestaciones para este proyecto debido a que es un sensor utilizado en dispositivos wearables, es decir, dispositivos utilizables o que están en contacto con las personas.

El sensor MAX30102 funciona con una sola fuente de alimentación de 1.8V y una fuente de alimentación de 5V para los leds, contiene dos LED: un led rojo (660nm) y un led infrarrojo (920nm), un fotodetector, óptica especializada, filtro de luz ambiental entre 50 y 60Hz para poder utilizarlo en diferentes ambientes, y un conversor ADC delta sigma de 16 bits y de hasta 1000 muestras por segundo. Además de incorporar un sensor de temperatura interno para compensar los efectos de la temperatura en la medición, su transmisión de datos se lleva a cabo por el protocolo de comunicación I2C, lo que lo hace sencillo de utilizar con un microcontrolador de la familia Arduino.

SUBSISTEMAS DEL CIRCUITO INTEGRADO.

En la figura 10 se puede apreciar los diferentes subsistemas que contiene el sensor MAX30102, subsistemas que son detallados a continuación.

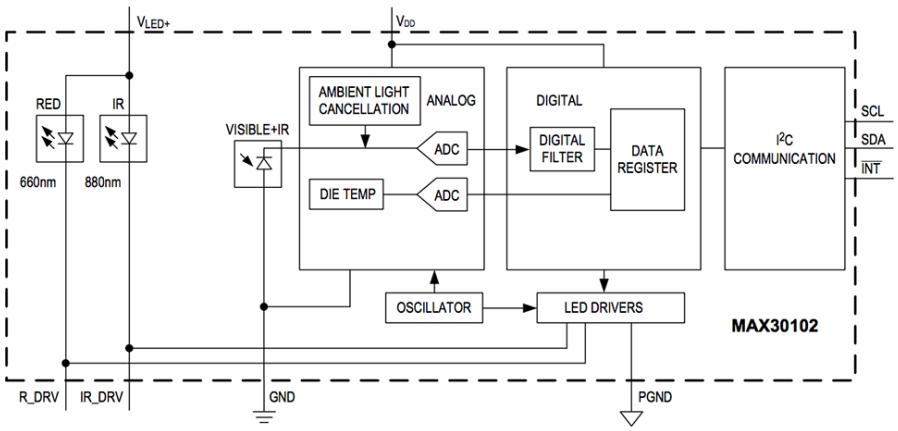


Fig. 10 Diagrama de bloques funcional del sensor MAX30102.

Fuente: *Hoja de datos.*

SUBSISTEMA DE SPO2.

El subsistema de SpO2 del MAX30102 contiene cancelación de luz (ALC), un sigma-delta de tiempo continuo ADC y un filtro de tiempo discreto patentado. El ALC tiene un circuito interno Track / Hold para cancelar la luz ambiental y aumentar el rango dinámico efectivo. El SpO2 ADC tiene rangos programables de escala completa de 2µA a 16µA. los ALC pueden cancelar hasta 200 µA de corriente ambiental.

El ADC interno es un convertidor continuo de sobre muestro sigma-delta con resolución de 18 bits. La frecuencia de muestreo del ADC es de 10,24 MHz. La tasa de datos de salida de ADC se puede programar de 50sps (muestras por segundo) a 3200sps. (El MAX30102 contiene también un sensor de temperatura.)

SENSOR DE TEMPERATURA.

El MAX30102 tiene un sensor de temperatura integrado en el chip para calibrar la dependencia de la temperatura del subsistema SpO2 este sensor de temperatura tiene una resolución inherente de 0.0625 ° C. Los datos de salida del dispositivo son relativamente insensibles a la longitud de onda del LED de infrarrojos, donde la longitud de onda del LED rojo es fundamental para la interpretación correcta de los datos. Un

algoritmo de SpO2 utilizado con la señal de salida MAX30102 puede compensar el error de SpO2 asociado con el ambiente cambios de temperatura.

A demás cuenta con controladores led para modular los pulsos que pueden ser emitidos desde el sensor.

CONTROLADOR LED.

El MAX30102 integra controladores LED rojos e IR para modular los pulsos de LED para mediciones de SpO2 y FC. La corriente del LED se puede programar de 0 a 50 mA con la tensión de alimentación adecuada. El ancho de pulso del LED puede ser programado de 69 µs a 411 µs para permitir que al algoritmo optimizar la precisión y el consumo de energía de SpO2 y HR basado en casos de uso.

FRECUENCIA DE MUESTREO Y RENDIMIENTO.

La frecuencia de muestreo máxima para el ADC depende del ancho de pulso seleccionado, que, a su vez, determina la resolución del ADC. Por ejemplo, si el ancho de pulso se establece en 69 µs entonces la resolución del ADC es de 15 bits y todas las frecuencias de muestreo son seleccionable. Sin embargo, si el ancho de pulso se establece en 411 µs, entonces las tasas de muestreo son limitadas.

RANGO DEL ADC PARA EL SpO2(18-Bits).

En la tabla 2 se pueden apreciar los rangos que puede tomar el ADC interno, dichos rangos pueden ser programados. Al cambiar el rango del ADC se puede aumentar o disminuir su resolución configurando un simple parámetro que será explicado en el capítulo de metodología.

TAMAÑO DEL LSB (pA)	ESCALA COMPLETA (nA)
7.81	2048
15.63	4096
31.25	8192
62.5	16384

Fuente: *Hoja de datos.*

Tabla 2 Rango del ADC.

FRECUENCIA DE MUESTREO DEL SpO2.

De igual manera se puede modificar el muestreo del algoritmo de SpO2 con los rangos mostrados en la tabla 3, la tasa de muestreo y el ancho del pulso están relacionados, la frecuencia de muestreo establece un límite superior en el tiempo del ancho de pulso.

SPO2	MUESTRAS POR SEGUNDO
1	50
2	100
3	200
4	400
5	800
6	1000
7	1600
8	3200

Tabla 3 Control de frecuencia de muestreo.

Fuente: *Hoja de datos.*

ANCHO DE PULSO LED.

En la tabla 4 se aprecia la configuración posible para el ancho de pulso del led con relación a la resolución del ADC, si el ancho de pulso se establece en 69 µs entonces la

resolución del ADC es de 15 bits y todas las frecuencias de muestreo son seleccionable. Sin embargo, si el ancho de pulso se establece en 411 µs, entonces las tasas de muestreo son limitadas.

ANCHO DEL PULSO	RESOLUCIÓN DEL ADC
69 (68.95)	15
118 (117.78)	16
215 (215.44)	17
411 (410.75)	18

Tabla 4 Control de ancho de pulso led.

Fuente: *Hoja de datos.*

MODO SPO2 (CONFIGURACIÓN PERMITIDA).

MUESTRAS POR SEGUNDO	ANCHO DE PULSO (us)			
	69	118	215	411
50	o	o	o	o
100	o	o	o	o
200	o	o	o	o
400	o	o	o	o
800	o	o	o	X
1000	o	o	X	X
1600	o	X	X	X
3200	X	X	X	X
RESOLUCIÓN (Bits)	15	16	16	18

Tabla 5 Configuración permitida para SpO2.

Fuente: *Hoja de datos.*

En la tabla 5 se puede ver la configuración admisible para dicho sensor comparando el muestreo por segundos, el ancho de pulso del led y la resolución en bits permitida, marcando con una X la configuración no accesible.

CIRCUITO TÍPICO PARA WEARABLE DE SALUD.

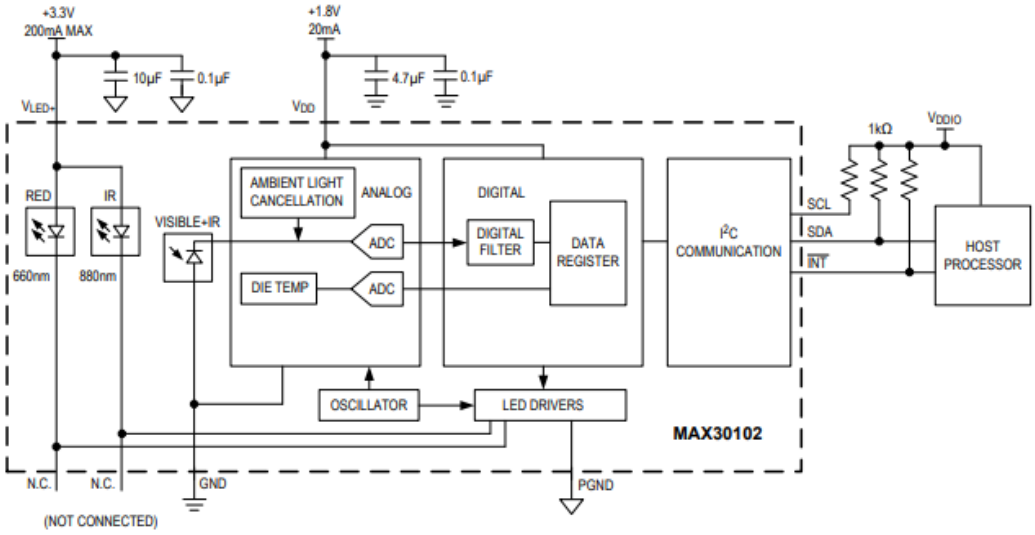


Fig. 11. Circuito para oximetría.

Fuente: *Hoja de datos.*

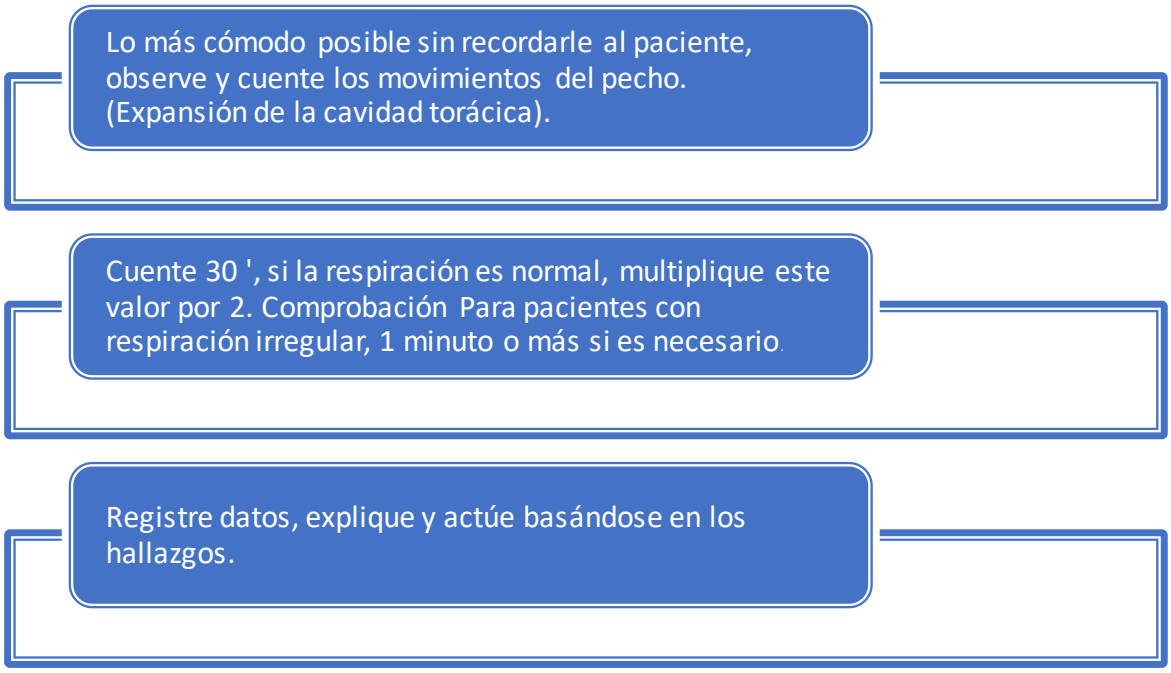
Este circuito que se encuentra en la Figura 11, es un circuito embebido para el uso en aplicaciones de oximetría donde se puede medir la oxigenación en la sangre y con la librería "SparkFun_MAX3010x_Pulse_and_Proximity_Sensor_Library", se puede configurar y hacer uso de este sensor. Cada registro puede ser modificado con la línea de código:

"`particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange);`"(Más información en el anexo 1)

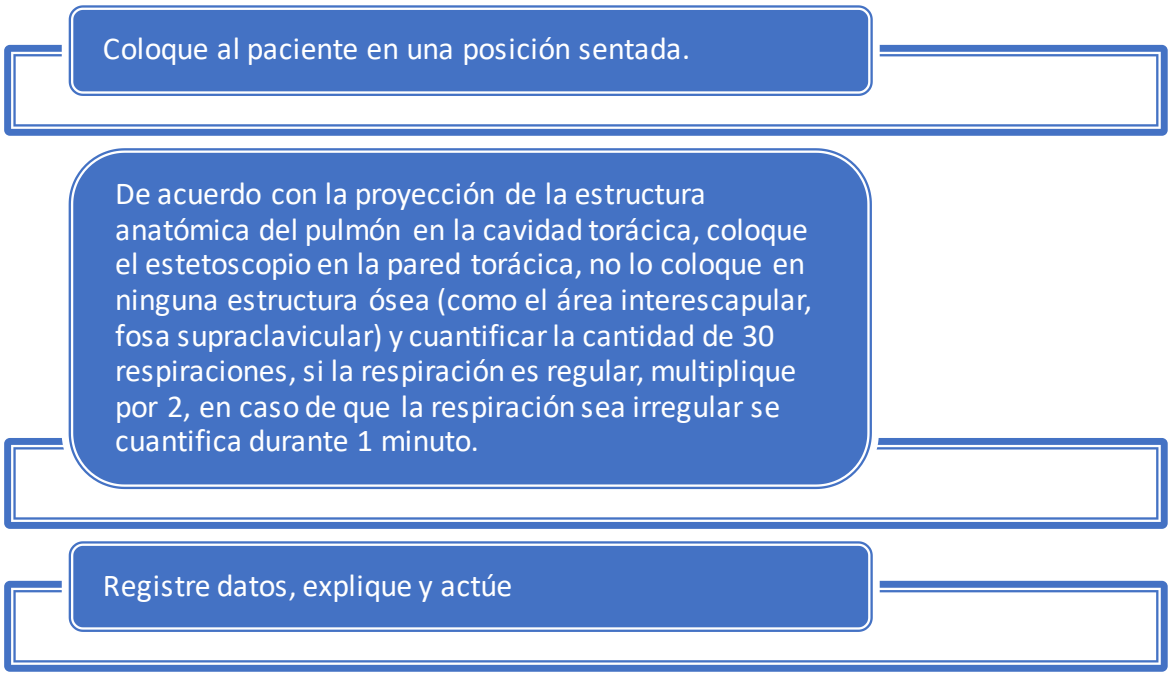
2.4 FRECUENCIA RESPIRATORIA.

Este es el número de veces que una persona respira por minuto, al medir las respiraciones es de importancia tener en cuenta el esfuerzo que realiza la persona para poder respirar, profundidad, ritmo, simetría, cada ciclo de respiración comprende de una fase de inspiración y otra de espiración. [15]

TÉCNICAS PARA EVALUAR LA FRECUENCIA RESPIRATORIA MEDIANTE INSPECCIÓN. [15]



TÉCNICA PARA EVALUAR LA FRECUENCIA RESPIRATORIA MEDIANTE AUSCULTACIÓN. [15]



VALORES NORMALES DE LA FRECUENCIA RESPIRATORIA EN PERSONAS SANAS.

EDAD	RESPIRACIONES POR MINUTO
Recién nacido	**30-80**
Lactante menor	**20-40**
Lactante mayor	**20-30**
Niños de 2 a 4 años	**20-30**
Niños de 6 a 8 años	**20-25**
Adulto	**15-20**

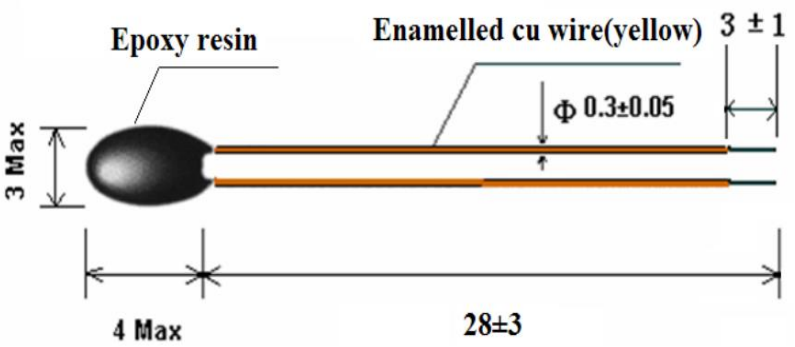
Tabla 6 Valores normales de la frecuencia respiratoria.

Fuente: *J. J. Talamas Márquez [15].*

En la tabla 6 se puede observar los valores de frecuencia respiratoria normales en personas sanas, diferenciados por su edad. Existen diferentes maneras de medir la frecuencia respiratoria, pero en este caso se puede medir tomando la variación que se distingue en la temperatura de las fosas nasales, mediante un termistor, los termistores son resistores que varían su resistencia mediante cambios de temperatura.

2.4.1 NTC3950 100K Ohm.

Fig. 12. Termistor NTC3950 100K Ohm.



Fuente: *Hoja de datos.*

El termistor tiene un encapsulado pequeño de vidrio, en el interior contiene oxido de semiconductor que hará la función de variar la resistencia dependiendo de la temperatura, el termistor aquí utilizado es un NTC de 100kOhm, ya que es de fácil acceso y para nuestra aplicación es suficiente, su respuesta es no lineal, pero en la hoja de datos se puede encontrar sus rangos de valores que van desde los -55°C hasta los 125°C. En la figura 13 se puede observar las curvas de los termistores NTC y PTC, cuando un termistor es NTC el valor de la resistencia del termistor disminuye a medida que la temperatura aumenta y cuando es PTC el valor de la resistencia del termistor aumenta a medida que la temperatura aumenta.

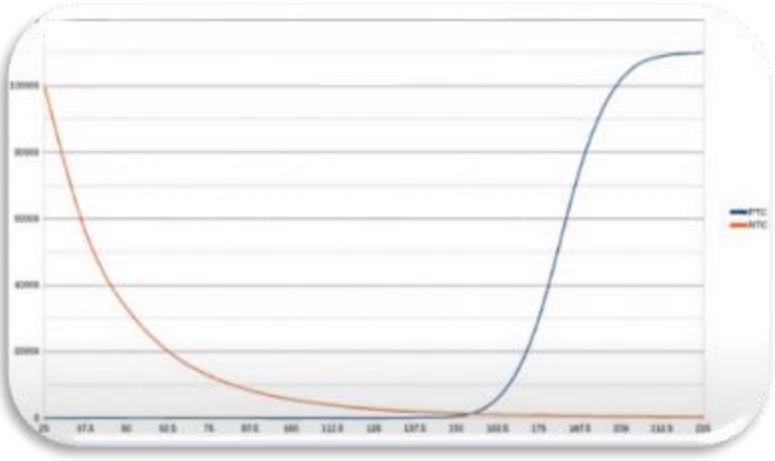


Fig. 13.. Curva de termistor NTC y PTC (Temperatura vs Resistencia).

Fuente: *Autor.*

2.5 COMUNICACIÓN INALÁMBRICA.

la comunicación inalámbrica permite enviar y recibir datos y voz entre dispositivos diferentes, mediante enlaces por radiofrecuencia, algunos sistemas embebidos como la familia Arduino soportan el protocolo de transmisión inalámbrica llamado Bluetooth que opera a los 2,4 GHz cuando se encuentra en un área reducida. [16]

2.5.1 PROTOCOLOS Y BUSES DE COMUNICACIÓN.

PROTOCOLO TPC/IP.

El acrónimo TCP / IP se refiere a un conjunto de protocolos utilizados para la comunicación de datos. Este conjunto necesita Su nombre proviene de sus dos acuerdos más importantes, Protocolo TCP (Protocolo de control de transmisión) y protocolo IP (Protocolo de Internet). Internet parece expandirse sin límite alguno, aunque el Protocolo TCP / IP siempre permanece igual: efecto, Internet ha hecho del protocolo TCP / IP un Estándar en todo tipo de aplicaciones telemáticas, incluidas redes locales y corporativas.

Es en esta zona llamada Intranet que el TCP / IP juega un papel cada vez más importante cada día. La popularidad de TCP / IP no se debe tanto a Internet como una serie de funciones que pueden satisfacer las necesidades actuales de transmisión de datos. En

el mundo, existen algunos destacados que hace referencia a que el estándar del protocolo TCP / IP es abierto, y es ampliamente compatible con varios sistemas, es decir, sus sistemas se pueden utilizar libremente y son desarrollados independientemente de los sistemas operativos y del hardware de los ordenadores, como también TCP / IP es adecuado para casi cualquier tipo de medio, no importa si se trata de Ethernet, Conexión ADSL o fibra óptica y el TCP / IP utiliza un esquema de direccionamiento, signa una dirección a cada dispositivo conectado incluso si toda la red es tan extensa, es única en toda la red. Al igual que Internet. [17]

ARQUITECTURA DEL PROTOCOLO TCP/IP.

El protocolo TCP / IP está en Capa OSI, por lo que el nivel de protocolo TCP / IP no cumple totalmente con los siete estándares establecidos por OSI. Hay una descripción del protocolo TCP / IP, que define de tres a cinco niveles. La figura 14 Representa el modelo TCP / IP de cuatro capas y su correspondencia con un modelo de referencia OSI. [17]

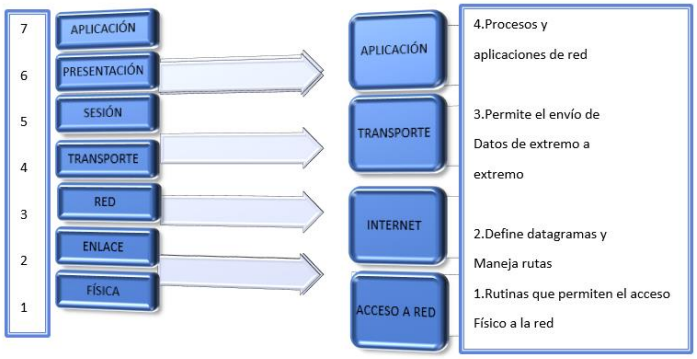


Fig. 14. Correspondencia del modelo OSI con TCP/IP.

Fuente: *A. E. Pérez [17].*

Los datos enviados a la red pasan por el protocolo TCP / IP de la capa de aplicación más alta al acceso de red más bajo. Después de recibir, pasan por la pila de protocolos en la

dirección opuesta. En estos recorridos, cada nivel ha aumentado o disminuido la información de control garantizando que los datos se transfieran correctamente. [17]

BUS DE COMUNICACIÓN I2C.

La abreviatura de I2C proviene de "Inter IntegratedCircuit", es una interfaz en serie que utiliza dos cables para llevar a cabo la comunicación. Esta interfaz la proporciona Philips Semiconductors A principios de los 80. Como el propósito de esta comunicación es conectar a través de 2 cables diferentes dispositivos, en este proyecto se utiliza par conectar los sensores de oxigenación en la sangre y la temperatura corporal, debido a que estos sensores disponen de dicho bus de comunicación. [18]

SDA (Serial Data Line): Es una línea para la transmisión de datos en serie y SCL (Serial Clock Line): Es la línea que transmite la señal de reloj generada por el dispositivo maestro para sincronización de datos y en la figura 15 se puede observar su estructura.[18]

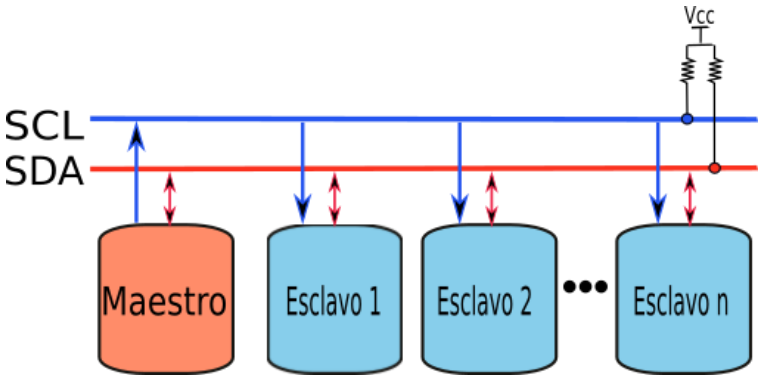


Fig. 15. Estructura del bus I2C.

Fuente: *I. T. de Querétaro [19].*

Ambas líneas son bidireccionales y solo se pueden utilizar dos estados. Activa baja O pasivo-alto, además de poder realizar una comunicación full-duplex entre todos los dispositivos conectados al circuito. La velocidad de transmisión de datos puede alcanzar

más de 100 kbps en modo estándar, más de 400 kbps en modo rápido y 3.4Mbps en modo de alta velocidad. [18]

HARDWARE DEL BUS I2C.

Se necesita una resistencia de "carga" o "pull up" a la salida del pin del microcontrolador que se utiliza para obtener una respuesta, está permitió conectar varias fuentes de datos en el mismo hilo. Hay un nivel alto en el bus cuando ningún dispositivo accede a él, también hay nivel alto cuando ningún dispositivo transmite un cero. Por otra parte, para que haya un nivel bajo, algún dispositivo debe poner un nivel bajo en el bus o si dos dispositivos escriben a la vez, en dicho caso prevalecerán los ceros. Para poder configurar las direcciones y tiempos de reloj que se utilizan en el proyecto se utiliza la librería wire.h programada en c que puede encontrarse en la plataforma Arduino. El cálculo de las resistencias de Pull-up depende de la capacidad del bus, de la tensión de alimentación, y del número de dispositivos conectados. [18]

COMUNICACIÓN SERIAL.

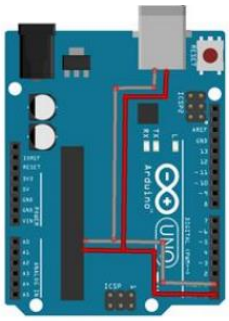
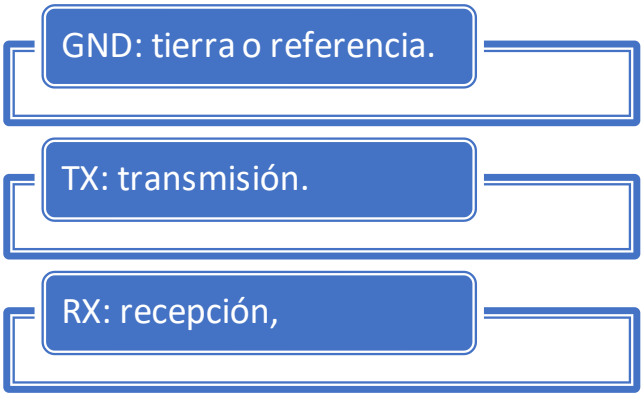


Fig. 16. Comunicación serial en Arduino.

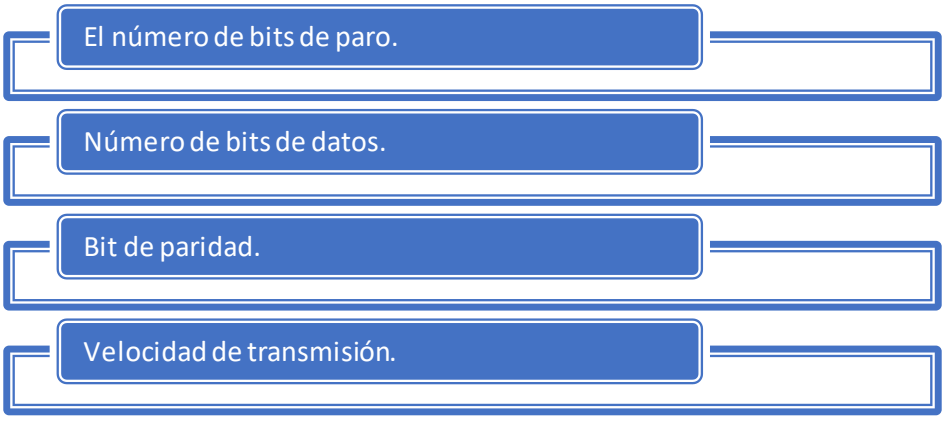
Fuente: *A. Uno and I. M. C. I. Ltda. [20]*

En la figura 16 se puede observar la comunicación de hardware serial que contiene el embebido Arduino uno, la familia Arduino contiene comunicación serial desde su gama más baja hasta su gama más alta. La comunicación serial es un tipo de protocolo de

comunicación entre dispositivos que es incluido de manera estándar en prácticamente cualquier computadora, pero también podemos encontrarlo incluido en placas de desarrollo y dispositivos embebidos como las placas de la familia Arduino. La comunicación serial RS232 es un protocolo común utilizado por dispositivos que son usados en instrumentación. La comunicación serial puede ser utilizada para control, depuración de código, adquisición de datos. Esta comunicación es típicamente utilizada para transmitir datos de formato ASCII. Se utilizan tres líneas de transmisión las cuales son:



Como este tipo de transmisión es asíncrona, se pueden enviar datos por una línea mientras estamos recibiendo por otra. La comunicación serial tiene características importantes como:



Es cierto que podemos transmitir grandes cantidades de datos, pero para que dos puertos se puedan comunicar, es necesario que tengan características iguales, en la

figura 17 puede observarse la ilustración de la transmisión de datos con sus características antes mencionadas. [19][20]

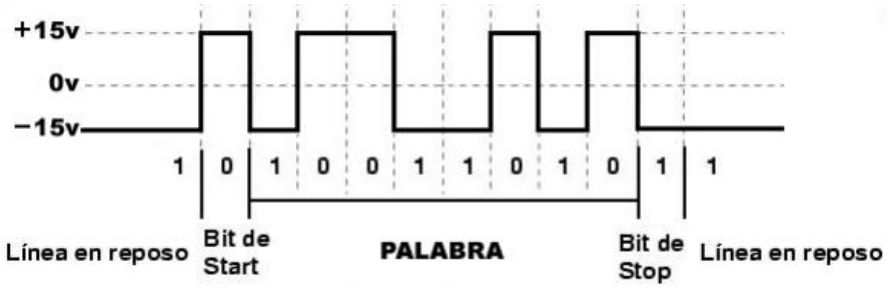


Fig. 17. Transmisión de datos por puerto serial.

Fuente: *I. T. de Querétaro. [19]*

Al hablar de velocidad de transmisión, hablamos del número de bits transmitidos por segundo y la unidad está en baudios (baudios); donde 100 baudios representarían 100bits por segundo y cuando nos referimos a los ciclos de reloj también nos referimos a velocidad de transmisión, es decir, si el protocolo hace un llamado a unos 4800 ciclos de reloj, entonces el reloj corre a 4800Hz, con lo cual, el puerto serial muestrea las líneas de transmisión a 4800Hz. Es común que las velocidades de transmisión sean de 4800, 9600 y 115200, debido a que al tener velocidades más altas se reduciría la distancia máxima posible entre dispositivos. [20]

2.6 ARDUINO MEGA

El Arduino Mega probablemente es el microcontrolador con más capacidad de la familia Arduino. Posee 16 entradas análogas, 54 pines digitales que funcionan como entrada/salida; un cristal oscilador de 16 MHz, un botón de reset, una conexión USB y una entrada para la alimentación de la placa. La comunicación entre la computadora y Arduino se lleva a cabo a través del Puerto Serie. Gracias a que posee un convertidor usb-serie sólo se necesita conectarlo a la computadora utilizando un cable USB como el de las impresoras.

2.7 FIREBASE Y APPINVENTOR

Firebase es una plataforma digital de la empresa Google que se utiliza para facilitar el desarrollo de aplicaciones web o móviles de una rápida y sencilla, Su objetivo principal, es mejorar el rendimiento de las apps mediante la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión desarrollada, segura y de fácil acceso para los usuarios, firebase puede combinarse con diferentes gestores de aplicaciones o entornos de programación, desde los más complejos hasta los más fáciles de manejar como lo es el caso de Appinventor, este último es un ide de programación en la nube que permite crear aplicaciones Android y comunicarlas con los servicios de firebase a través de programación en bloques de código.

3. METODOLOGÍA.

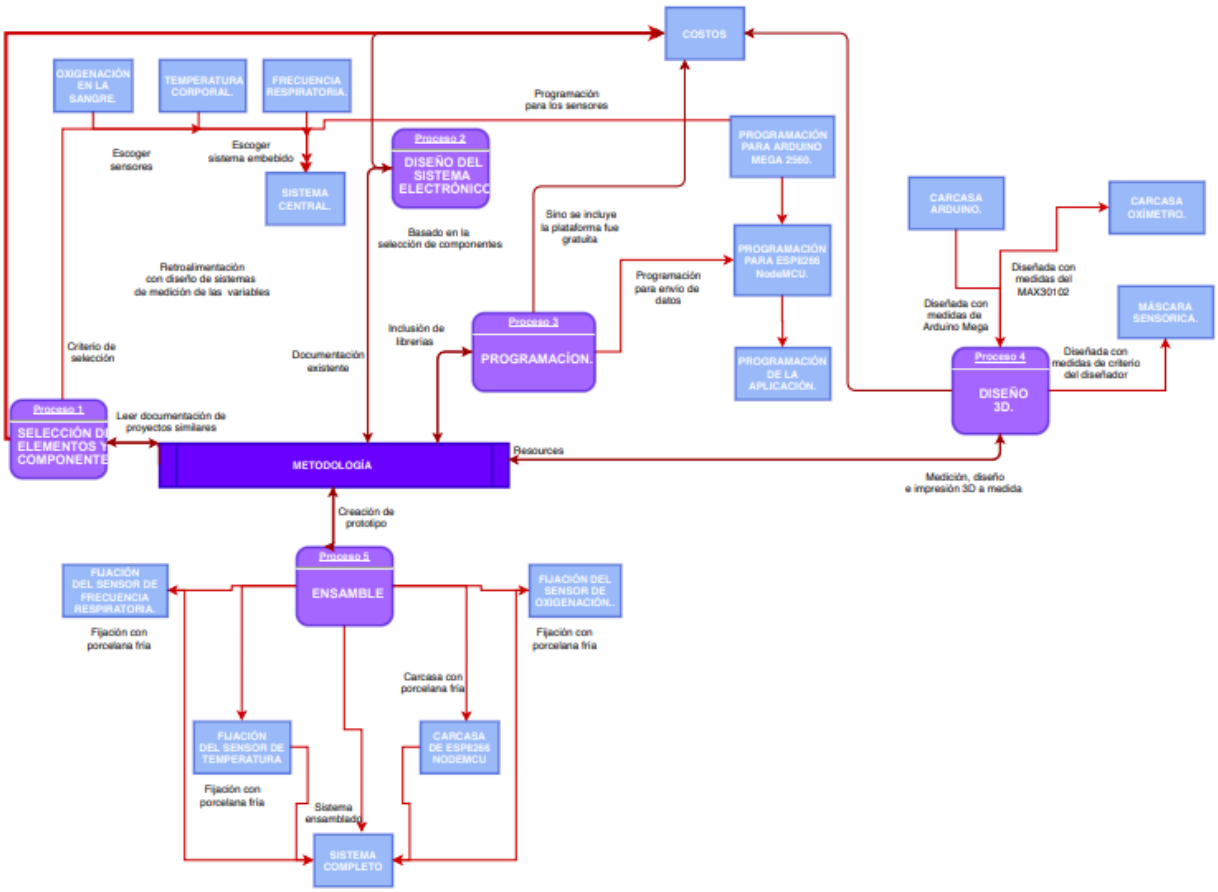


Fig. 18. Metodología.

Fuente: *Autor.*

En la figura 18 se observa la ruta metodológica ubicada por bloques que se seccionarán a continuación para poder llegar a una ruta de creación del prototipo como se muestra en la figura 19.

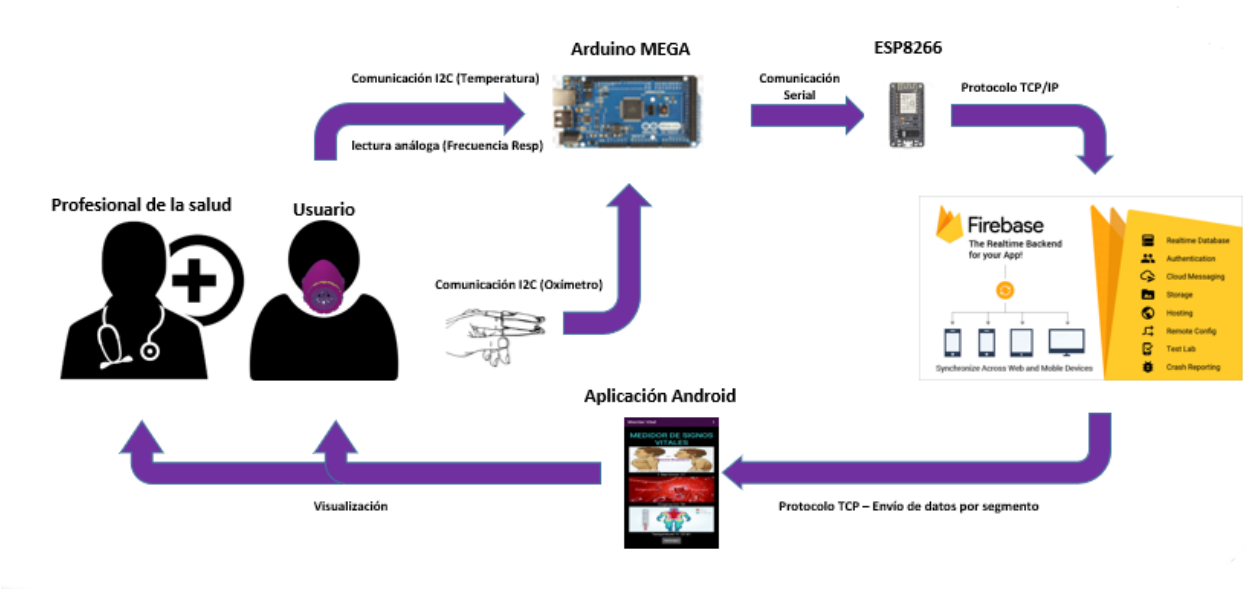


Fig. 19. Ruta de funcionamiento del prototipo.

Fuente: *Autor.*

3.1 SELECCIÓN DE ELEMENTOS Y COMPONENTES.

Después de estudiar con detenimiento lo que nos puede aportar la electrónica actual basada en los requerimientos de nuestra aplicación, es hora de escoger los sensores, hardware y protocolos de comunicación óptimos para el desarrollo de nuestro proyecto.

En el trabajo se planteó la medición de variables de: Oxigenación en la sangre, temperatura corporal y frecuencia respiratoria, para poder abarcarlos todos debemos se estudian por separado obteniendo casos diferentes de medición para cada uno.

3.1.1 OXIGENACIÓN EN LA SANGRE.

Al medir la oxigenación en la sangre se habla de saturación de oxígeno arterial (SaO2), con esto nos referimos a la proporción (%) de proteína de oxígeno en sangre en el volumen sanguíneo total. Cuando la saturación de oxígeno es más alta, la hemoglobina se vuelve roja brillante. La pulsioximetría se basa en la diferencia en la absorción de ondas de luz por la hemoglobina oxigenada y desoxigenada. Necesitaremos un oxímetro con un diodo que emita ondas de luz roja al infrarrojo cercano, tenga un fotodiodo en el

otro extremo para que detecte la luz transmitida o reflejada a través del tejido y un microprocesador que puede identificar y separar componentes pulsantes (arterias) y componentes no pulsantes. (vena) sin olvidarnos de la economía en este caso.

CRITERIO DE SELECCIÓN DE MAX30102.

NOMBRE	VCC	SENSOR SOLITARIO	ELIMINACIÓN DE RUIDO	ADC	I2C
MAX30100	5V	NO	SI	16 bits	SI
MAX30102	5V	SI	SI	16 bits	SI

Tabla 7 Comparación MAX30102 y MAX30100

Fuente: *Hojas de datos y precios de la tienda MercadoLibre.*

En la tabla 7 En el mercado colombiano podemos encontrar 2 tipos de sensores de oximetría o sensores utilizables para oximetría de la misma familia, con características similares en sus hojas de datos, pero con diseños un poco diferentes y precios fluctuantes, se primó el MAX30102 sobre el MAX30100 debido a su costo inferior y a que el sensor MAX30100 tiene unas resistencias cerca al lente que contiene los leds que podrían obstaculizar la medición en diferentes situaciones.

3.1.2 TEMPERATURA CORPORAL.

Para poder medir la temperatura corporal en esta aplicación es importante evitar el contacto con la piel, puesto que nuestro sensor se ubica en una mascarilla ubicada en el rostro, al evitar el contacto con el sensor prevendremos errores en mediciones y posibles focos de infecciones, dicho esto procedemos a escoger un sensor que facilite la medición sin tener contacto, como lo es el sensor MLX90614.

CRITERIO DE SELECCIÓN DE MLX90614.

NOMBRE	Vcc	CONTACTO	ERROR	RANGO	I2C
DS18B20	3.3V-5V	SI	± 0,5°C	-55° a 125º C	NO
Dht11	3.3V-5V	NO	± 2°C	0° -50° C	NO
LM35	4V-40V	SI	± 0,5°C	-55° a 150° C	NO
MLX90614	3V-5V	NO	± 0,5°C	-40° a 125°C	SI

Tabla 8 Criterio de selección del MLX90614

Fuente: *Hojas de datos y precios de la tienda MercadoLibre.*

En este caso como podemos ver en la tabla 8, a pesar de que en el mercado colombiano se puede encontrar diferentes tipos de sensores de temperatura, las características que ofrece el sensor MLX90614 son ideales para este prototipo, debido a que permite mediciones con un buen margen de error sin necesidad de tener contacto con el objetivo a medir, provocando que la aplicación sea menos invasiva; a pesar de que el sensor Dht11 tiene características similares y también puede medir sin contacto, el sensor MLX90614 es más adecuado para esta aplicación por su precisión.

3.1.3 FRECUENCIA RESPIRATORIA.

Para medir la frecuencia respiratoria se utilizan diferentes métodos, pero en esta sección utilizamos un método diferente a los convencionales, para medir la frecuencia respiratoria debemos conocer cuando una persona está inhalando y exhalando, por ende, luego de hacer un estudio de la manera más viable de medir dichos parámetros y adecuarlos a nuestro diseño, se decide optar por un termistor NTC que al aumentar la temperatura reduce su resistencia, se ubica cerca de las fosas nasales para poder notar el cambio en la temperatura cuando se produce una exhalación y una inhalación, estos datos se toman durante 1min para poder visualizar los resultados de la frecuencia respiratoria. En este caso al no contar con un sistema embebido o circuito integrado se procede a hacer el diseño del circuito y el cálculo para su utilización. Para poder utilizar el termistor con un microcontrolador se deben transformar sus valores en datos que pueda leer dicho embebido, de modo qué, si se utiliza la ecuación de Steinhart-Hart (ecuación 1) que es

un modelo de resistencia de un semiconductor a diferentes temperaturas, se pueden traducir los valores del NTC al microcontrolador de manera analógica. [21]

La ecuación es:

Ecuación 1 Steinhart-Hart

$$1/T = A + BlnR + C(lnR)\text{^}3 \quad (1)$$

Donde:

T: Temperatura (en grados Kelvin).

R: Valor de resistencia (en Ohms).

A, B, C: Coeficientes de Steinhart-Hart.

Para poder encontrar los coeficientes se puede revisar en la hoja de datos proporcionada por el fabricante y buscar los coeficientes, en caso de no ver esta información, se necesita conocer al menos tres puntos de operación, para esto, se utilizan tres datos de resistencia para tres temperaturas conocidas. Pero también se pueden recibir a través de simuladores de la web sino se cuenta con las condiciones ideales para la medición como en este caso, para este propósito se utiliza la web: "https://www.thinksrs.com/downloads/programs/Therm%20Calc/NTCCalibrator/NTCcalculator.htm" esta web es básicamente una calculadora de dichos coeficientes como se muestra en la figura 20, esta calculadora recibe los 3 valores de resistencia y temperatura que se pueden obtener de la hoja de datos, en este caso se tomaron 5°,25° y 45° C que corresponden a valores de:

5°C= Entre 246.103 y 255.737 kOhm.

25°C= Entre 99.000 y 101.000 kOhm.

45°C=Entre 42.865 y 44.462 kOhm.

Respectivamente, y se ingresan a la calculadora con valores aproximados para facilitar los cálculos.

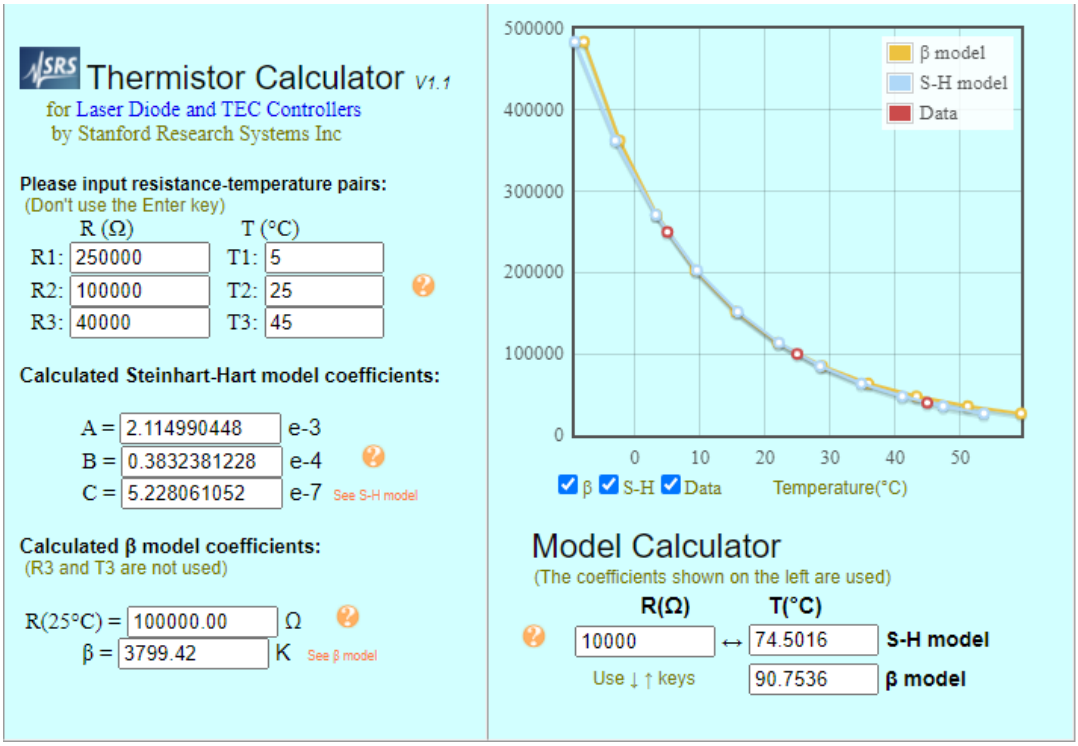


Fig. 20. Calculadora de coeficientes de Steinhart-Hart.

Fuente: *www.thinksrs.com*

Así se obtienen los coeficientes:

A= 2.114990448e-3

B= 0.3832381228e-4

C= 5.228061052e-7

Que posteriormente se utilizarán con un divisor de tensión para poder medir el voltaje en el Arduino.

3.1.4 SISTEMA CENTRAL.

Para la toma de todas las variables deseadas se necesita un embebido lo suficientemente poderoso y versátil que pueda acoplar todos los sensores y pueda transmitirlos posteriormente como es el caso de la tarjeta de desarrollo Arduino Mega 2560.

CRITERIO DE SELECCIÓN DEL ARDUINO MEGA 2560.

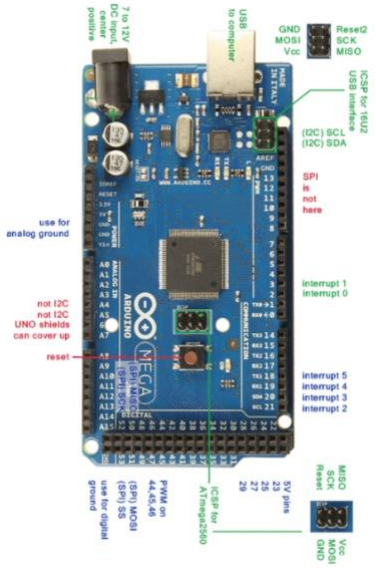
NOMBRE	Vcc	SRAM	EEPROM	AN I/O	DIG I/O
PSoC 5LP	1.7/5.5V	64KB	2KB	3	62
ARDUINO MEGA 2560	5 /12 V	8KB	4KB	16	54
EKTM4C1	5V	32KB	2KB	12	40
PIC18F4550	4.2V/5.5V	2KB	256B	13	35
RASPBERRY Pi 3	5V	1GB	SD	0	40

Tabla 9 Comparación de embebidos

Fuente: *Hojas de datos y precios de las tiendas MercadoLibre y Vistronica.*

Atendiendo a las necesidades del prototipo, se tiene en cuenta las especificaciones del sistema embebido como se muestra en la tabla 9, así como su implementación adicional y su compatibilidad con los sensores seleccionados previamente, como se puede notar, entre un **PIC18F4550** y un **Arduino MEGA 2560** existen bastantes similitudes por las que se podría decantar por el PIC debido a su menor tamaño de embebido, pero, al intentar implementar un algoritmo en dicho microcontrolador, se necesita un programador que tiene una implementación extra, debido a que dicho microcontrolador no cuenta con un programador interno, además de que en la plataforma de Arduino se pueden encontrar códigos y librerías que para el PIC no se encontrarían, lo que quiere decir que, se tendrían que implementar lo que resultaría ineficiente desde el punto de vista del tiempo de elaboración del prototipo.

ARDUINO MEGA 2560.



Fuente: *Hoja de datos.*

Con este embebido Arduino, visualizado en la figura 21, se hace uso de las bibliotecas mencionadas para poder adquirir los datos de oxigenación, temperatura (Ambos por el puerto I2C de este embebido) y respiración incluyendo los cálculos que se obtuvieron en la sección anterior, se procesan y se convierten en datos ASCII para poder enviarlos por el puerto serial, esperando la señal de un botón que controla el usuario a voluntad (Este Arduino cuenta con 3 puerto seriales) hacia el siguiente embebido que se encarga de llevar todo a la nube.

ESP8266 NodeMCU.

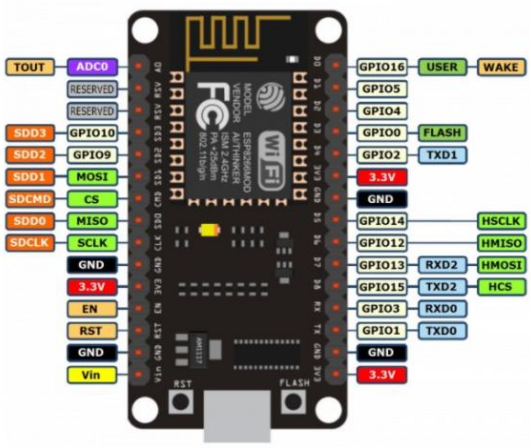


Fig. 22. ESP8266 NodeMCU.

Fuente: *Hoja de datos.*

CRITERIO DE SELECCIÓN DEL ESP8266 NodeMCU.

En la figura 22, se observa el embebido encargado de enviar los datos a la nube, es decir, el NodeMCU, es una placa Wifi de pequeñas dimensiones, es compatible con Arduino y puede ser utilizada en cualquier proyecto IoT (Internet de las cosas). Está montada alrededor del ya conocido ESP8266 y tiene todos sus pines en los laterales, incorpora un regulador de tensión integrado, así como un puerto USB de programación. Se puede programar con LUA o mediante el IDE de Arduino que es lo que se hace en este caso, para implementar el NodeMCU con el IDE de Arduino, sólo se debe instalar su librería y el programa lo reconocerá automáticamente, el NodeMCU puede enviar datos a firebase configurando pocas líneas de código gracias a la librería creada para Arduino, (Más información en el anexo 2)

3.2 DISEÑO DEL SISTEMA ELECTRÓNICO.

Los sistemas para cada medición se especifican en la sección anterior, en esta sección nos concentraremos en tenerlos todos en conjunto un conjunto. Los dos primeros sensores (**Temperatura** y **Oxigenación**) tienen prioridad al compartir el puerto I2C, pero

también gracias a esto su conexión se hace en paralelo y con dos resistencias de pull-up en el puerto, una desde Vcc hasta SCL y otra desde Vcc hacia SDA.

SENSOR DE OXIMETRÍA Y SENSOR INFRARROJO CONECTADOS EN PARALELO.

En la figura 23 se observa la conexión entre el sensor de oxigenación y el de temperatura, conectados mediante el protocolo I2C, para poder hacer uso de estos dos sensores mediante este puerto se usa la librería Wire.h de la plataforma de Arduino que facilita la conexión de dispositivos mediante los puertos I2C y así poder conectar ambos sensores.

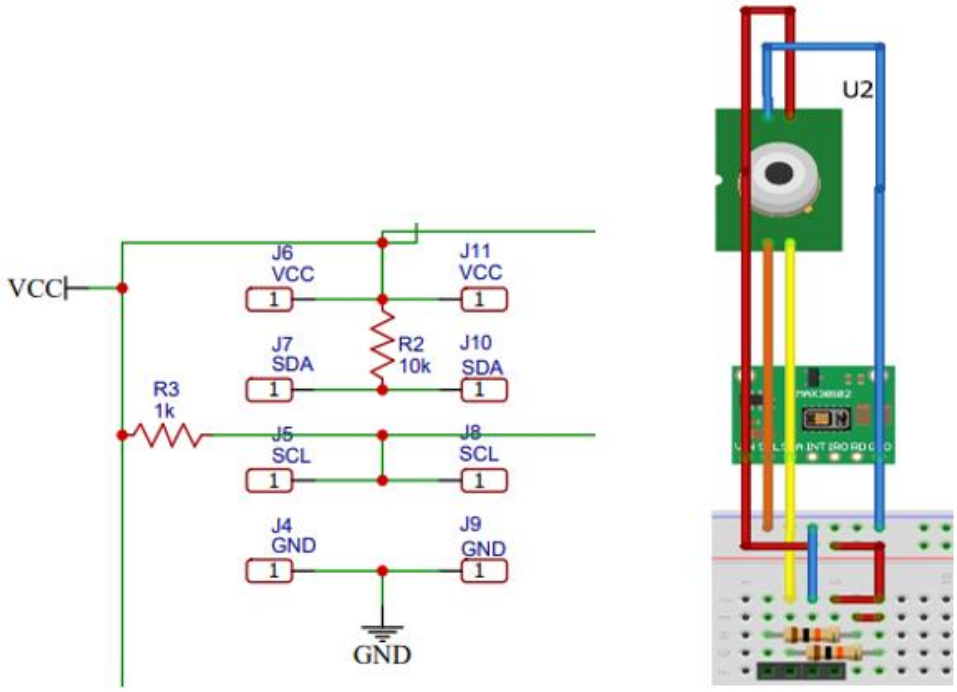


Fig. 23. Esquemático de conexión I2C.

Fuente: *Autor*

El intervalo de las resistencias pull-up válidas para SDA y SCL está limitado por las especificaciones de carga estática y limitado por la especificación de flanco ascendente

y descendente. El valor mínimo de la resistencia está determinado por la carga de corriente máxima que puede manejar el transistor de salida. El límite de especificación I2C es de 3 mA (para modo estándar y rápido) o de 20 mA (para modo rápido plus). [22]

Si se considera que cada dispositivo tiene una capacitancia de 10pf se obtiene 20pf por tener 2 dispositivos y se desprecia el cable por estar cerca al Arduino. Para modo estándar: I2C-bus: t = tiempo de subida = 1000 ns (1 µs), entonces RC = 1180.2ns. [22]

Para modo rápido: tiempo de subida del bus I2C = 300 ns

Vcc= 5V

Ecuación 2 Calculo de resistencias para i2c con frecuencia < 100khz

$$Freq < 100KHz \quad \frac{R_{\min} \quad (Vcc(5) - 0.4)}{3mA} = 1.5kOhms \quad R_{max} = \frac{1000ns}{C_{bus(20pf)}} = 50kOhms \tag{2}$$

Ecuación 3 Calculo de resistencias para i2c con frecuencia >100khz

$$Freq > 100KHz \quad \frac{R_{\min} \quad (Vcc(5)-0.4)}{3mA} 1.5kOhms, \quad R_{max} = \frac{300ns}{C_{bus(20pf)}} = 15kOhms \tag{3}$$

En la ecuación 2 se muestra el cálculo para las resistencias cuando la frecuencia es menor que 100KHz, como se utilizan en esta aplicación frecuencias mayores de 100kHz, se hace uso de la ecuación 3, utilizando un valor intermedio del rango de resistencias, es decir 10kOhms.

Para el sensor de **Frecuencia respiratoria,** se escogen las constantes que se utilizarán en el programa que se implementará en el Arduino y su montaje y los circuitos son sencillos, se coloca una resistencia del mismo valor del termistor para que trabaje como divisor de tensión y así poder leerla con un pin analógico del Arduino, como se muestra en la figura 24.

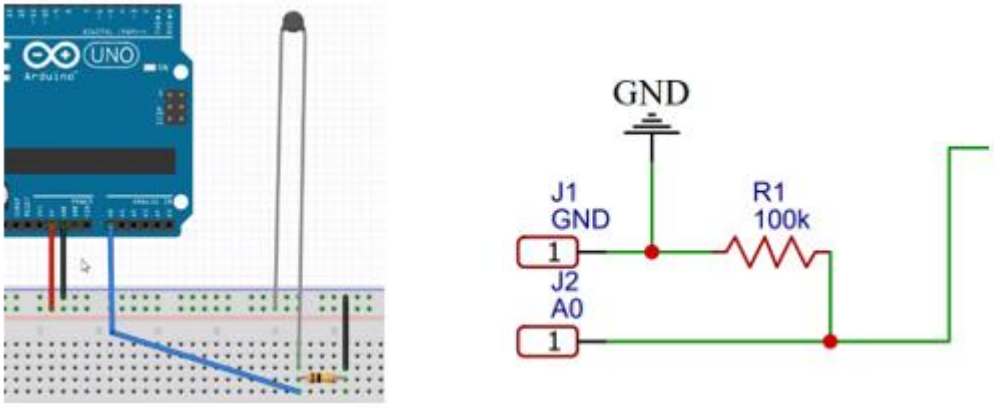


Fig. 24. Esquemático de termistor.

Fuente: *Autor.*

Una vez obtenidos los circuitos que le darán vida a la aplicación, el prototipo está casi listo, ahora se agregan indicadores para darle una idea al usuario de lo que está pasando en ese momento con el sistema electrónico, es decir, se encenderán mientras se esté realizando una medición (un led para la temperatura y la oxigenación y otro para la respiración), además se integra un pulsador para que el usuario pueda hacer la transferencia de datos del Arduino Mega hacia el NodeMCU, cuando esto suceda se iluminará un led en la placa receptora, avisando que se hizo la comunicación y que está recibiendo datos.

En la figura 25 se puede apreciar la conexión de tipo protoboard o montaje físico con el que se tendría una noción de las conexiones de manera visual y en la figura 26 se pueden ver dichas conexiones a manera de esquemático.

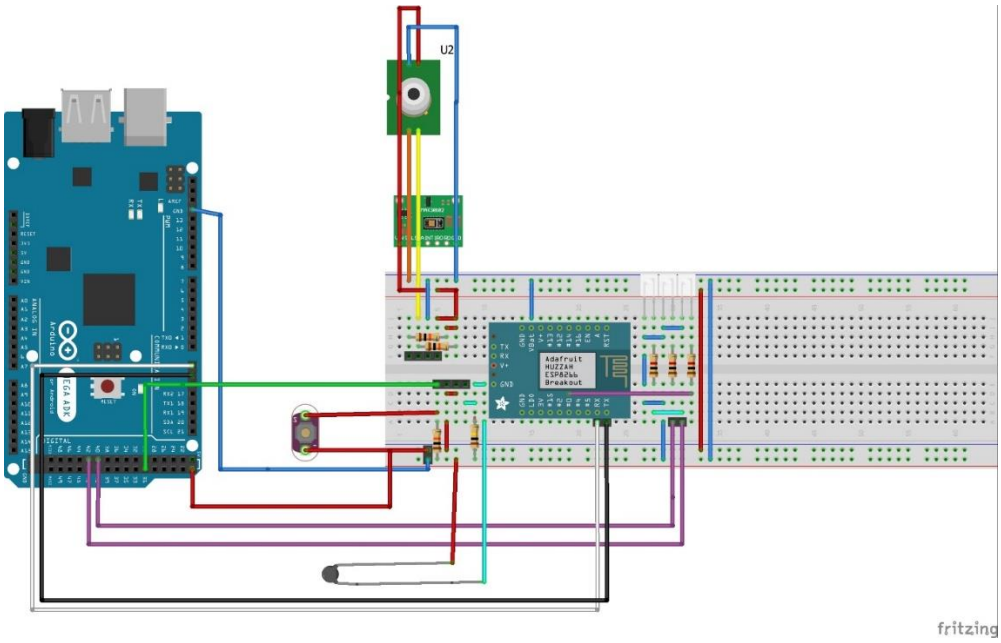


Fig. 25. Esquemático tipo protoboard del sistema electrónico completo.

Fuente: *Autor.*

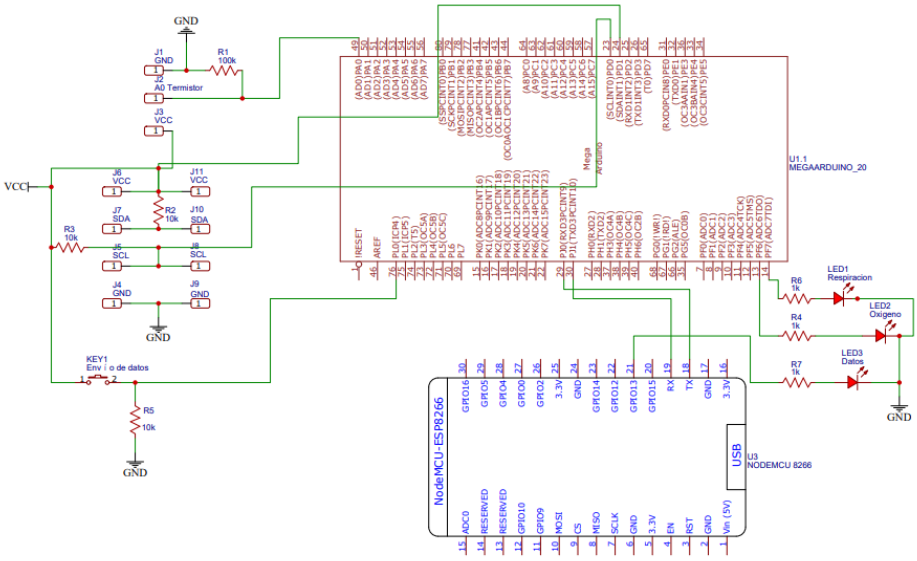


Fig. 26. Esquemático general de conexiones.

Fuente: *Autor.*

(PCB en el anexo 3)

RESISTENCIAS PARA LED.

Ecuación 4 Calculo de resistencias para led

$$Resistencia(Ohms) = \frac{Tensión\ de\ alimentación - Caida\ de\ tensión\ del\ LED}{Corriente\ dentro\ de\ lo\ admitible\ por\ el\ led} \quad (4)$$

Siguiendo la ecuación 4 se pueden intercambiar los indicadores.

Leds rojos: 1,2V de caída de tensión, 20mA de corriente

$$\frac{\mathbf{5v - 1,2V}}{\mathbf{20x10^{-3}}} = \mathbf{190\Omega}$$

$$\frac{\mathbf{5v - 1,2V}}{\mathbf{20x10^{-3}}} = \mathbf{190\Omega}$$

$$\frac{\mathbf{5v - 1,2V}}{\mathbf{20x10^{-3}}} = \mathbf{190\Omega}$$

Con una resistencia de 190Ohms es suficiente para funcionar correctamente, pero podremos colocar un valor comercial por encima y no habrá errores.

FRECUENCIA DE MUESTREO DEL DISPOSITIVO.

El prototipo sigue la guía de la toma de datos de frecuencia respiratoria, es decir, debido a que la frecuencia respiratoria se toma durante un minuto, el tiempo mínimo en el que podremos tomar muestras es de 1.1min para el dispositivo completo, dividiéndolo así:

Frecuencia respiratoria: 1min.

Oxigenación en la sangre: 1 seg.

Temperatura corporal: 1 seg.

La definición de la frecuencia de muestreo del dispositivo se estableció teniendo en cuenta lo anteriormente mencionado. La medición de la temperatura y la oxigenación puede ser constante o instantánea, pero la frecuencia respiratoria se establece durante 1min, no obstante, el dispositivo puede configurarse para hacer cuantas mediciones se deseen al día durante 1.1min cada una y así lo establezca el profesional de la salud encargado.

3.3 PROGRAMACIÓN.

3.3.1 PROGRAMACIÓN PARA ARDUINO MEGA 2560

Estas librerías son las que ayudan a darle vida a sistema, en una sección previa se habló de ellas, si las se repasan rápidamente, se encuentra la librería **Wire.h** que se encarga de controlar el bus I2C, la **"MAX30105.h"** que permite controlar el sensor **MAX30102,** con la diferencia que el sensor **MAX30105** contiene un led verde que no se utiliza en este algoritmo, así que es compatible. Se utiliza el algoritmo del SpO2 que contiene la librería proporcionada por la empresa **SparkFun Electronics,** dicha librería se encargará de controlar los registros del **MAX30102** para esta aplicación, y por último, la librería **Adafruit_MLX90614.h,** proporcionada por la empresa **Adafruit Industries** que se encargará de controlar los registros del sensor **MLX90614** y entregar una medida, en la figura 27 se pueden notar el llamado a las librerías utilizadas.

```
#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include <Adafruit_MLX90614.h>
```

Fig. 27. Librerías para Arduino Mega.

Fuente: *Autor.*

REGISTROS DEL SENSOR DE OXIGENACIÓN.

```
byte ledBrightness = 60; //Opciones: 0=Off a 255=50mA //brillo del led rojo
byte sampleAverage = 4; //Opciones: 1, 2, 4, 8, 16, 32 //promedio de la muestra
byte ledMode = 2; //Opciones: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green// modo de los leds
byte sampleRate = 100; //Opciones: 50, 100, 200, 400, 800, 1000, 1600, 3200 //frecuencia de muestreo
int pulseWidth = 411; //Opciones: 69, 118, 215, 411  //ancho del pulso
int adcRange = 4096; //Opciones: 2048, 4096, 8192, 16384  //Rango del ADC

particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Se configura el sensor



//lee las primeras 100 muestras y determina el rango de la señal
for (byte i = 0 ; i < bufferLength ; i++)
{
  while (particleSensor.available() == false) //pregunta por nueva data
    particleSensor.check(); //revisa el sensor por nueva data

  redBuffer[i] = particleSensor.getRed(); //Led Rojo
  irBuffer[i] = particleSensor.getIR(); //Led infrarojo
  particleSensor.nextSample(); //se termina con esta muestra y se procede a la siguiente
}

//calcula ritmo cardiaco y SpO2 despues de las primeras 100 muestras (primeros 4s de muestras)
maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength, redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);
```

Fig. 28.Configuración de registros y muestreo.

Fuente: *Autor.*

En la figura 28 se muestra la configuración de los registros mencionados en una sección anterior y se obtienen 100 muestras en 4 segundos. (Para más información, remítase al anexo 1)

SENSOR INFRARROJO.

```
,
 Ob=mlx.readObjectTempC(); // Medición de la temperatura con la librería <Adafruit_MLX90614.h>
 dato1=String(Ob);
```

Fig. 29.Medición de temperatura.

Fuente: *Autor.*

En la figura 29 se muestra el llamado de la función mlx.readObjectTempC, para el sensor de infrarrojos, luego sólo se debe guardar la temperatura en una variable. (Para más información, remítase al anexo 1)

SENSOR DE RESPIRACIÓN.

```
float R1 = 100000;              // resistencia fija del divisor de tension
float logR2, R2, TEMPERATURA;
float c1 = 2.114990448e-03, c2 = 0.3832381228e-04, c3 = 5.228061052e-07;
float actual=0;
float anterior=0;


float obtenerTemp(){
  Vo=analogRead(A0);  // lectura de A0
  R2=R1*(1023.0 /(float)Vo-1.0); // conversion de tension a resistencia
  logR2=log(R2);  // logaritmo de R2 necesario para ecuacion
  TEMPERATURA=(1.0/(c1+c2*logR2+c3*logR2*logR2*logR2));   // ecuacion S-H
  TEMPERATURA=TEMPERATURA-273.15;   // Kelvin a Centigrados (Celsius)
  return TEMPERATURA;
}
```

Fig. 30. Declaración de variables y creación de función de temperatura para respiración.

Fuente: *Autor.*

En la figura 30 se puede notar la creación de una función para leer la temperatura del termistor y aplicar la ecuación 1. Por otra parte, en la figura 31 se puede ver un cronómetro que puede ser configurado fácilmente, cambiando el valor de 240 que representa segundos, de esta manera se pueden tomar muestras cada vez que se desee, el valor mínimo es 61.

```
if(millis()-tiempo>=1000){
  tiempo2=tiempo2+1;
  tiempo=millis();
  if(tiempo2>=240){
  tiempo2=0;
  }
 }
```

Fig. 31.Cronómetro interno.

Fuente: *Autor.*

```
if (tiempo2<=60){//Durante el primer minuto de encendido el sistema se tomarán las muestras de respiración

  TEMPERATURA=obtenerTemp();//Se hace llamada a la función obtenerTemp
  actual=TEMPERATURA;

        if(actual>anterior+0.5)
        {anterior=actual;
          c=1;
          }
        if(actual<anterior-0.5){
          anterior=actual;
          c=0;
          }
         if(c!=d){
          cont=cont+1;
          d=c;
          }
```

Fig. 32. Toma de frecuencia respiratoria.

Fuente: *Autor.*

En la figura 32 se observa la creación de un cronómetro interno en el Arduino para que el termistor tome variaciones de temperatura durante un minuto, entre inhalación y exhalación. (Para más información, remítase al anexo 1)

ENVÍO DE DATOS

En la figura 33 se visualiza que para poder enviar los datos hacia el módulo ESP8266 NodeMCU se deben convertir dichos valores capturados, en variables de String. Se valida el valor medido del sensor de oxigenación en condicionales que convierten sus datos en información entendible.

```
if(validSPO2==0){
  dato2="invalido";
}else{
if(spo2>=92 && validSPO2==1){
  Oxi=spo2;
  dato=String(Oxi);
  dato2=dato;
}
 if(spo2<=91 && validSPO2==1){
  conta1=conta1+1;
   if(conta1>10){
    cont1=0;
    dato2="SPO2=_por_debajo_del_92%,_Recurra_a_consulta_médica";
   }
}
}
```

Fig. 33. Condiciones para conversión de datos.

Fuente: *Autor.*

En la figura 34 se observa que le Arduino espera la indicación del pulsador ubicado en el pin 31 de enviar los datos a través del puerto serial 3, que tenemos conectado al ESP8266 NodeMCU.

```
} int serial=digitalRead(31);
    if(serial==1){
     conteoresp=String(cont);
     signos=String(dato2+',')+String(dato1+',')+String(conteoresp); //Agrupación de los 3 datos de signos vitales para posterio
     Serial3.println(signos); }//Envío de datos por el serial del puerto 3 del Arduion MEGA
```

Fig. 34. Envío de datos por el serial 3.

Fuente: *Autor.*

(Para más información, remítase al anexo 1)

3.3.2 PROGRAMACIÓN PARA ESP8266 NodeMCU

Esta placa es un embebido pequeño con menores prestaciones que el Arduino Mega, pero contiene funciones de WIFI que se pueden utilizar en el mismo IDE de Arduino. La librería ESP8266WiFi.h nos permite controlar nuestra placa NodeMCU y la librería FirebaseESP8266.h que facilita la utilización del protocolo TCP/IP para poder conectarse al servicio de base de datos de Google: Firebase, en la figura 35 se puede observar el llamado a las librerías y funciones utilizadas en la placa ESP8266.

```
#include "FirebaseESP8266.h"
#include <ESP8266WiFi.h>
#include <Ticker.h>
#define FIREBASE_HOST "mascara-multisensorial.firebaseio.com" //Sin http:// o https://
#define FIREBASE_AUTH "T52G4unSkXBKFV4ecydbOkg5AyxKXEsLW3MMJBny"
#define WIFI_SSID "Martha"
#define WIFI_PASSWORD "ciro1969"
#include <Separador.h>
Ticker tiempo_1;

String path = "/Sensores";
//Define un objeto de Firebase
FirebaseData firebaseData;
```

Fig. 35. Librerías para NodeMCU.

Fuente: *Autor.*

Al definir los parámetros anteriores se puede crear una conexión entre la placa y Firebase por medio de la red del hogar y el Host creado en firebase" mascara-multisensorial.firebaseio.com"

CREACIÓN DE LOS PATHS EN FIREBASE.

```
if (Firebase.setString(firebaseData, path + "/Temperatura corporal", Temperatura)){InforSetLuzSensor();}
else{CausaError();}
if (Firebase.setString(firebaseData, path + "/Oxigenacion en la sangre", Oxigenacion)){InforSetLuzSensor();}
else{CausaError();}
if (Firebase.setString(firebaseData, path + "/Frecuencia respiratoria", Frecuencia)){InforSetLuzSensor();}
else{CausaError();}
if (Firebase.setString(firebaseData, path + "/Estado del Sensor de Oxigeno", ValorMensaje)){InforSetLuzSensor();}
else{CausaError();}
if (Firebase.setString(firebaseData, path + "/Estado de temperatura del paciente", ValorMensaje1)){InforSetLuzSensor();}
else{CausaError();}

Boolenvio=false;
tiempo_1.attach(10, funcion_1);
pinMode(led,OUTPUT);
digitalWrite(led,LOW);
```

Fig. 36. Creación de Paths en firebase.

Fuente: *Autor.*

Luego de configurar el NodeMCU y crear los paths como se muestra en la figura 36 la interfaz de la base de datos en firebase mostrará la información que puede verse en la figura 37. (Para más información, remítase al anexo 2)

Fig. 37. Base de datos en tiempo real.

Fuente: *Autor.*

3.3.3 PROGRAMACIÓN DE LA APLICACIÓN.

Fig. 38. Ruta de construcción de aplicaciones móviles.

Fuente: *Platzi.*

En la figura 38 se puede apreciar la ruta de creación de app móviles proporcionada por la empresa Platzi, necesitaremos un lenguaje de FrontEnd, un lenguaje de transporte de datos, un lenguaje de Backend y por último una base de datos, para usos prácticos el lenguaje de Frontend se desarrolla en APP inventor, (sotfware creado por el MIT para desarrollo de apps Android) firebase empaqueta el resto de los lenguajes en un solo servicio, por esta razón se utiliza en este proyecto.

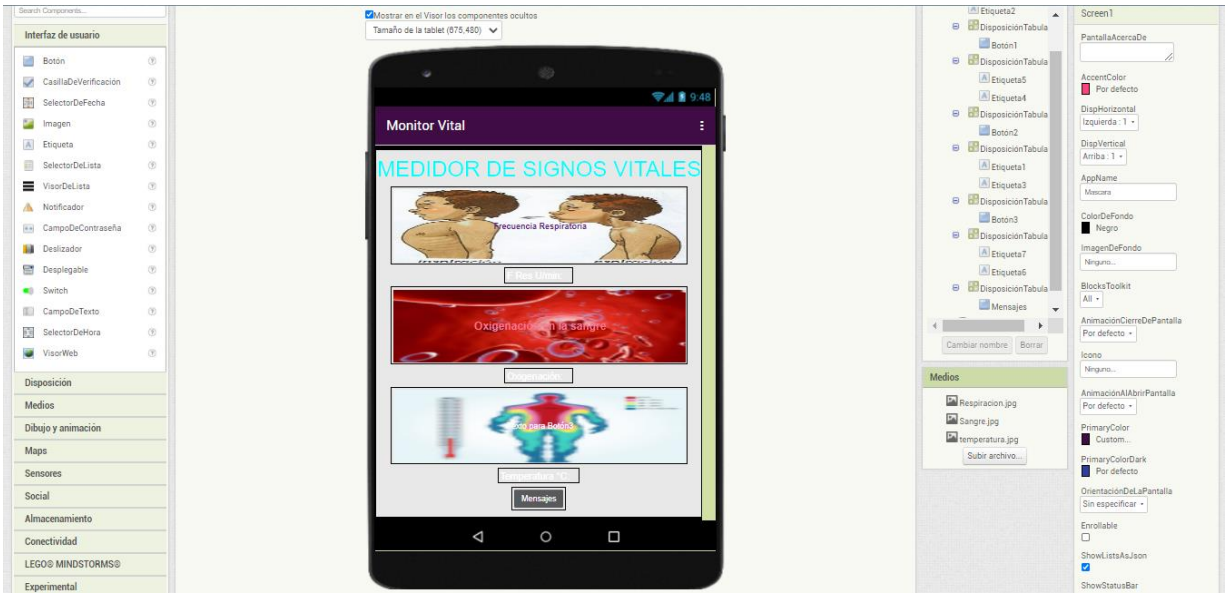


Fig. 39. Interfaz gráfica en App Inventor.

Fuente: *Autor.*

La interfaz gráfica de la figura 39 es el entorno de desarrollo de Appinventor, mientras que los códigos de bloques que aparecen en la siguiente figura, es decir, la figura 40, son la lógica que maneja y da vida a la interfaz de usuario de este prototipo.

Fig. 40. Bloques de programación en App Inventor.

Fuente: *Autor.*

La programación en bloques de App Inventor es muy intuitiva y al estar en español se puede entender fácilmente su lógica, al exportar nuestro código e interfaz, se obtiene la siguiente apk:

Fig. 41. Mensajes y alertas de los sensores

Fuente: *Autor.*

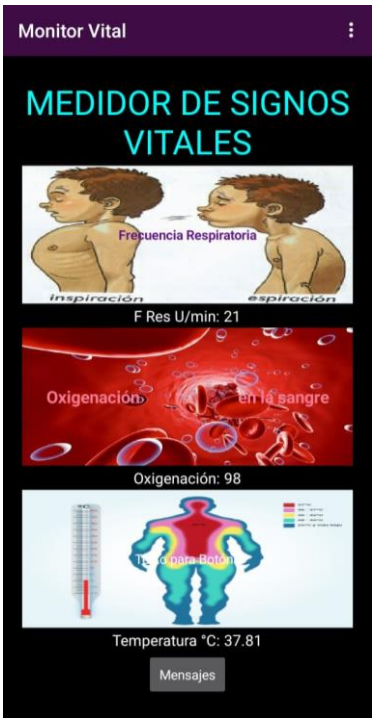


Fig. 42. Interfaz de monitoreo en app

Fuente: *Autor.*

En las figuras 41 y 42 se puede ver la interfaz de la aplicación móvil para Android, en la cual al presionar sobre las imágenes, se puede solicitar los datos que tenga la base de datos, siendo:

F Res U/min: el ciclo de frecuencia respiratoria.

Oxigenación: la saturación en la sangre, medida por nuestro oxímetro.

Temperatura: la temperatura medida por nuestro sensor infrarrojo.

Al solicitar información en “Mensajes” se puede ver si el oxímetro tomó muestras o no, por estar en conectado al paciente o no al presionar el botón “Oxi” y con el botón “Temp”

se podrá ver el estado de la temperatura en el paciente, incluyendo sólo dos estados “Fiebre” o temperatura normal.

3.4 DISEÑO 3D.

Para el desarrollo de las piezas necesarias en este proyecto se utiliza un software en la nube llamado Tinkercad, debido a su bajo consumo de recursos físicos y acceso gratuito a demás de público o privado (dependiendo la configuración, los objetos creados pueden ser públicos o no) las figuras 42 y 43 son imágenes de diseños 3D realizados en este software.

3.4.1 CARCASA ARDUINO.

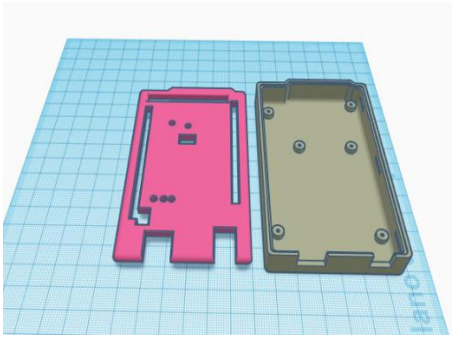


Fig. 44. Parte superior e inferior.

Fuente: *Autor.*

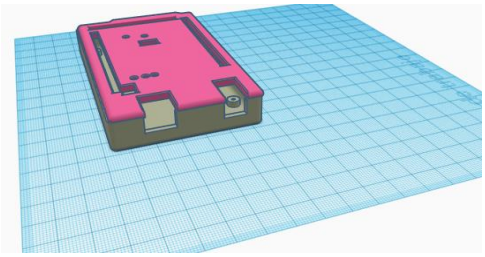


Fig. 43. Carcasa ensamblada.

Fuente: *Autor.*

Para la creación de dichas piezas se toma la medida de un Arduino Mega.

(para más información, remítase al anexo 4)

3.4.2 CARCASA OXÍMETRO.

Para la carcasa en el oxímetro se tomaron medidas basadas en un oxímetro comercial con 6cm de largo 3 de ancho y 3 de alto y el tamaño de nuestro sensor. Las figuras 45, 46 y 47 son diseños 3D de un oxímetro para el sensor MAX30102.

(Para más información, remítase al anexo 4)

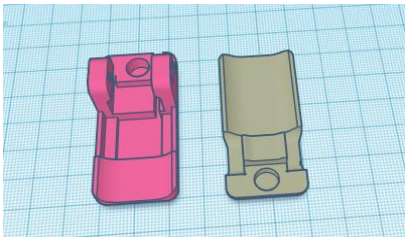


Fig. 45. Partes inferior y superior respectivamente.

Fuente: *Autor.*

Parte superior.

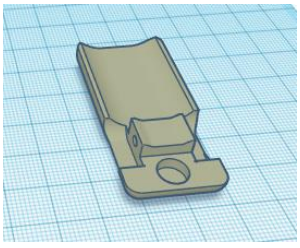


Fig. 47. Parte superior del oxímetro.

Fuente: *Autor.*

Parte inferior.

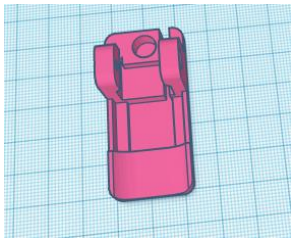


Fig. 46. Parte inferior del oxímetro

Fuente: *Autor.*

3.4.3 MÁSCARA SENSORICA.

CÁLCULOS DE DISEÑO DE LA MÁSCARA 3D

Para el diseño del tamaño de la mascarilla se toma de guía una mascarilla Zubi-Ola para polvo termosellada, desechable con un clip metálico recubierto que permite ajustarse a la nariz. Tomando como referencia sus bases de ajuste desde el ancho de la cara en los pómulos, desde el tabique hasta el mentón y por último la profundidad de la cavidad para

respirar en dicha máscara siendo estas medidas marcadas con rojo en la figura 48 de 12cm, 14cm y 8cm respectivamente.

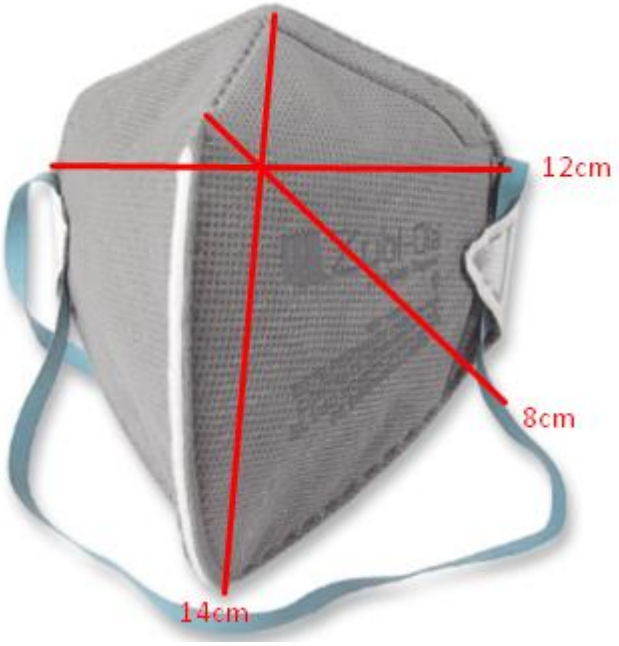


Fig. 48. Mascara de referencia Zubi-Ola

Fuente: *Autor.*

Obteniendo de esta manera unas referencias de medida a partir de una mascarilla comercial adaptable y accesible en el mercado actualmente.

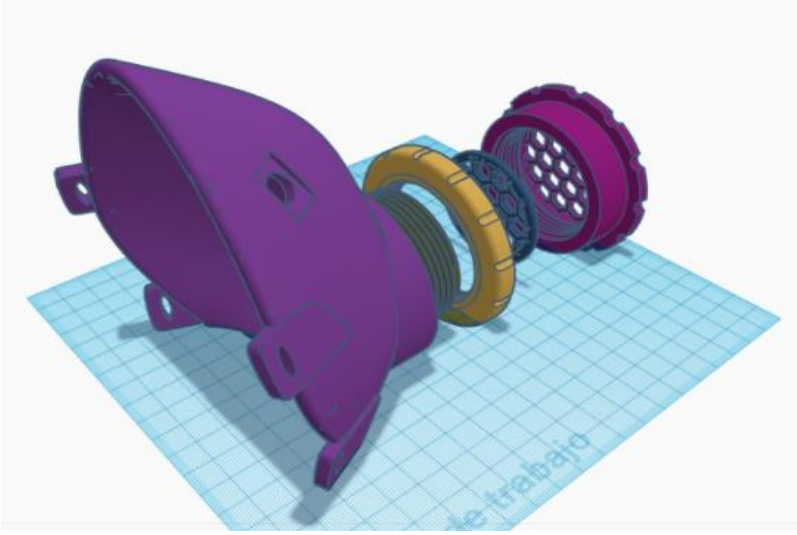


Fig. 49. Máscara con sus respectivos filtros.

Fuente: *Autor.*

En la figura 49 se detalla la máscara 3D diseña en Tinkercad, tomando de referencia las medidas anteriormente establecidas en la figura 46. (Para más información, remítase al anexo 4)

Para la ubicación del Sensor infrarrojo se modela dentro del programa Tinkercad, con las medidas del sensor, pero con algunas modificaciones para que pueda quedar sujeto a presión, como puede verse en la figura 50.

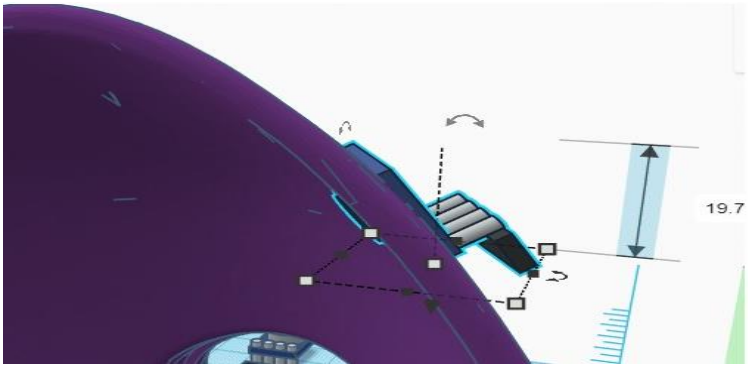


Fig. 50. Máscara con sensor infrarrojo incrustado.

Fuente: *Autor.*

3.5 ENSAMBLE.

Se colocan las fijaciones de los sensores y se apoya en porcelana fría, la cual se puede moldear y una vez se seque se puede limpiar con alcohol.

3.5.1 FIJACIÓN DEL SENSOR DE FRECUENCIA RESPIRATORIA.

Fig. 51. Fijación sensor de frecuencia respiratoria.

Fuente: *Autor.*

El sensor de frecuencia respiratoria se ubica en el interior de la máscara impresa en 3D, cerca de las fosas nasales para poder detectar los cambios de temperatura y registrarlos, como se puede observar en la figura 51.

3.5.2 FIJACIÓN DEL SENSOR DE TEMPERATURA.

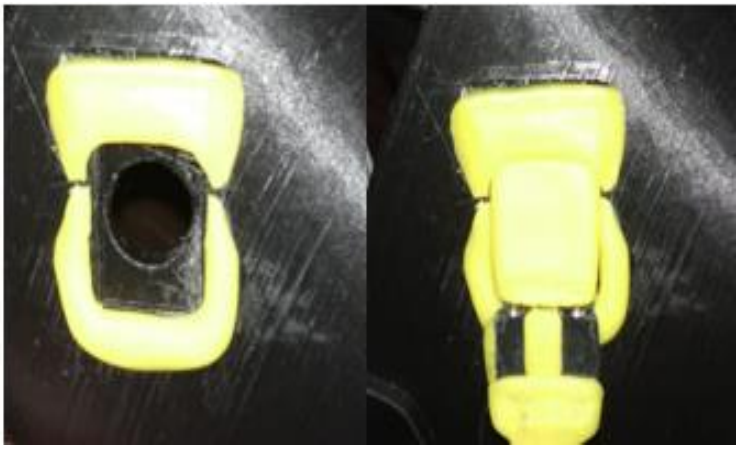


Fig. 52. Fijación para sensor infrarrojo.

Fuente: *Autor.*

En la fijación del sensor de temperatura, se puede notar que, al crear un espacio con porcelana fría para el sensor infrarrojo, se puede remover con facilidad para poder limpiar el área y volver a situarlo en el mismo lugar, esto se puede notar en la figura 52.

3.5.3 FIJACIÓN DEL SENSOR DE OXIGENACIÓN.

Fig. 53.Oxímetro removible en porcelanicrón.

Fuente: *Autor.*

La fijación del sensor de oxigenación en la sangre se lleva a cabo en porcelana fría, permitiendo moldear a conveniencia, como se puede ver en la figura 53.

3.5.4 CARCASA DE ESP8266 NODEMCU.

Fig. 54.Carcasa con porcelanicrón para ESP8266 NodeMCU.

Fuente: *Autor.*

En la figura 54 se obversa una carcasa para el ESP8266 creada en porcelanicrón o porcelana fría, para poder utilizar las conexiones requeridas y proteger la placa.

3.5.5 SISTEMA COMPLETO.

Fig. 55. Sistema incluido en madera.

Fuente: *Autor.*

El sistema completo es compacto y puede ser incluido en un estuche pequeño como en la figura 55.

Fig. 56. Botón en porcelanicrón.

Fuente: *Autor.*

El botón de la figura 56 se puede utilizar para tener el control del envío de datos hacia la base de datos en la nube.

Los filtros utilizados en la máscara que se observan en la figura 57, se hacen a partir de mascarillas N95 gracias a que este tipo de filtros pueden ser descontaminados con agua caliente. [23]

Fig. 57. Filtros de la máscara.

Fuente: *Autor.*

CONEXIONES FÍSICAS.

En la figura 58 se puede encontrar la guía de conexiones físicas del prototipo terminado, siendo IV: conexión del sensor de frecuencia respiratoria, III: conexión del sensor de oxigenación

II: conexión del sensor de temperatura, en las conexiones externas. Así mismo podemos notar las conexiones del Arduino y la NodeMCU, tanto externa como internamente.

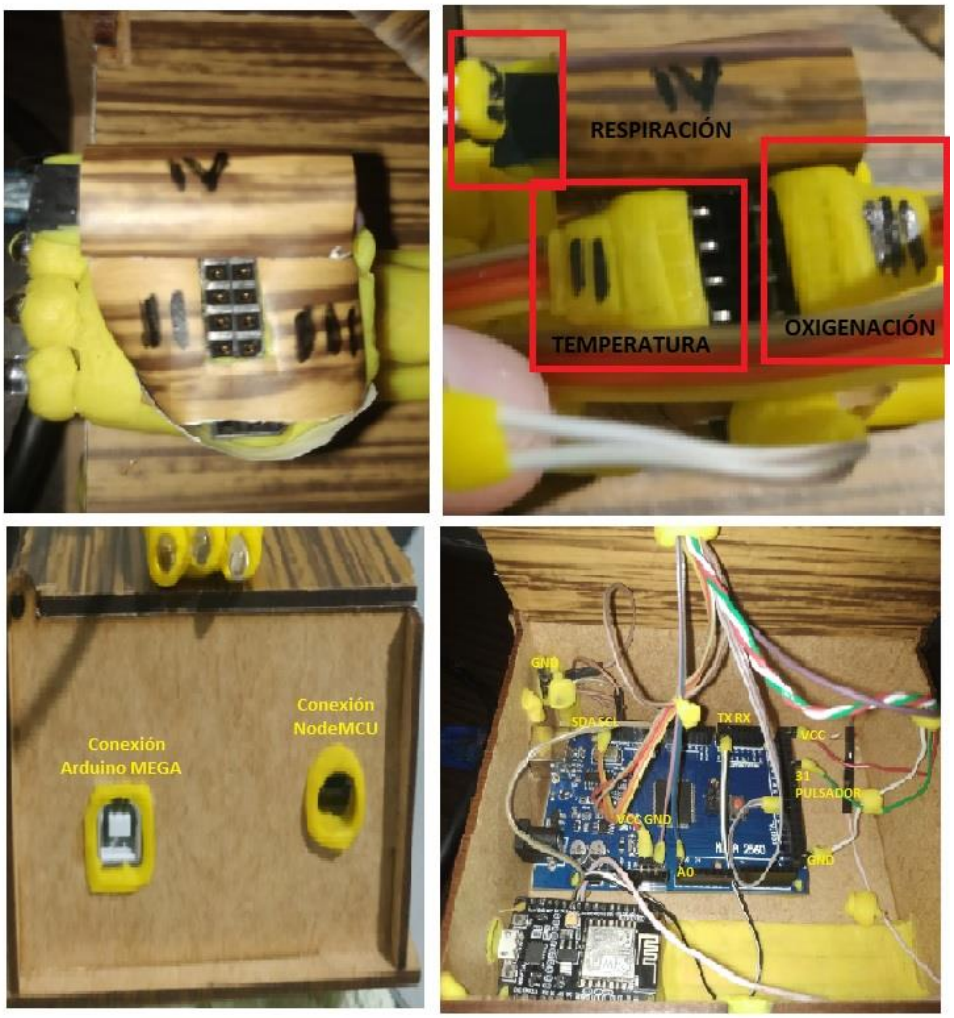


Fig. 58. Conexiones externas e internas.

Fuente: *Autor.*

Fig. 59. Sistema completo.

Fuente: *Autor.*

El prototipo terminado como lo muestra la figura 59, contiene los sensores: MAX30102 ubicado en una carcasa de porcelanicrón, el MLX90614 y el termistor fijados en la máscara, para poder medir la frecuencia respiratoria y la temperatura corporal. Dentro de la caja podremos encontrar el Arduino Mega y el ESP8266, con sus respectivas conexiones y en la parte superior de la caja, encontraremos el paquete de conexiones donde podremos conectar los sensores y un botón para controlar el envío de datos.

3.6 COSTOS DEL PROYECTO

Por último, se realiza el análisis de costos teniendo en cuenta todos los elementos que fueron utilizados en nuestro proyecto:

TABLA DE COSTOS		
ARTICULO	**CANTIDAD**	**PRECIO TOTAL**
ARDUINO MEGA 2560	1	38.500
ESP8266 NODEMCU	1	17.500
MAX30102	1	25.000
TERMISTOR NTC3950 100KOHM	1	3.500
MLX90614	1	35.000
PORCELANA FRÍA 125GR	1	2.000
CABLE BUS DE DATOS 2 METROS	1	7.500
REGLETA HEMBRA	2	2.000
REGLETA MACHO	1	1.000
CAJA DE CORTE LASER DE MADERA	1	3.500
CABLE MICRO USB	1	15.000
MASCARA COMPLETA IMPRESA EN 3D	1	94.800
COSTOS DE ENVÍOS	3	29.200
TOTAL	274.500	

Tabla 10 Costos.

Fuente: *Autor.*

4. RESULTADOS

4.1 COMUNICACIÓN

Al comunicar la placa Arduino Mega y la placa ESP8266 se obtiene una respuesta idónea utilizando el puerto serial este prototipo, como se muestra en la figura 58.

La velocidad a la que se reflejan los datos desde la placa ESP8266 van a depender completamente de nuestra velocidad de internet, pero al tratarse de pocas cadenas de datos ASCII, podremos ver resultados casi al instante.

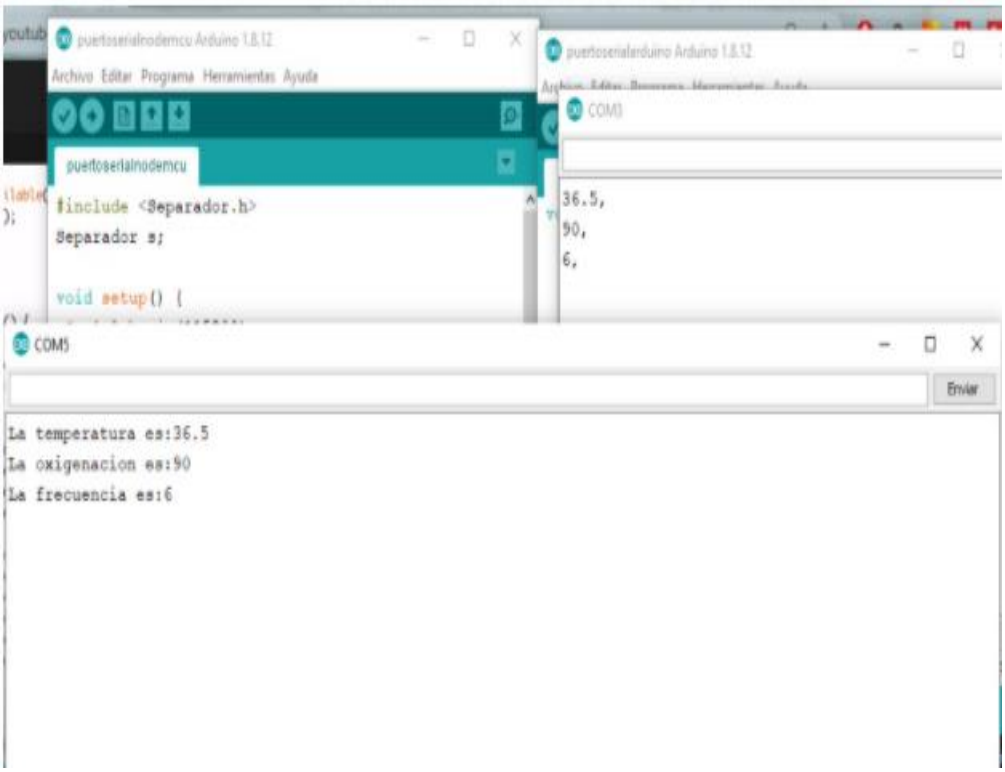


Fig. 60. Comunicación ESP8266 y Arduino MEGA.

Fuente: *Autor.*

En la figura 60 se puede notar que en la parte inferior (COM5) se muestra el envío de datos desde el Arduino MEGA y su recepción en la parte superior derecha (COM3) por parte del ESP8266 NodeMCU, completando una transferencia exitosa a través de protocolo serial.

4.2 PRUEBA Y VALIDACIÓN DEL SISTEMA

Para la validación y prueba del sistema, se realizan medidas con personas de diferentes edades y sexos, para comprobar que la mascarilla se ajusta a ellos como se puede ver en la figura 61.

Fig. 61. Mascarilla en diferentes rostros.

Fuente: *Autor.*

Se puede observar en la figura 61 que la máscara se ajusta a diferentes tipos de rostros, gracias a la forma en la que está diseñada con los parámetros establecidos.

4.3 PRUEBA DEL SISTEMA CON EQUIPOS COMERCIALES.

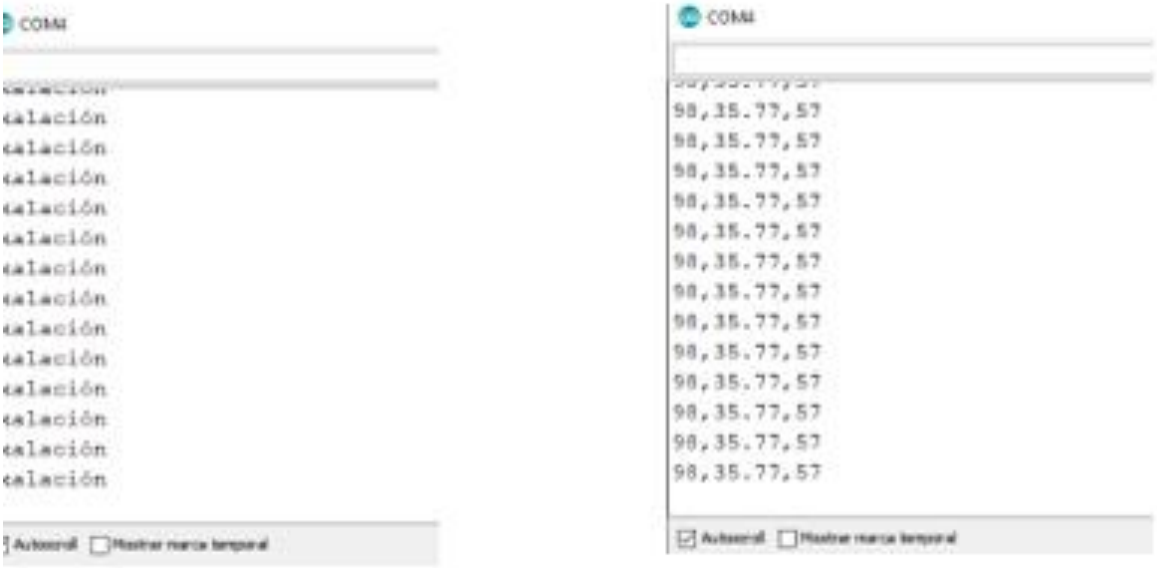


Fig. 62. Envío de datos por serial y medición del ciclo respiratorio.

Fuente: *Autor.*

Se puede evidenciar la toma de datos en la figura 62, en la izquierda se observa los ciclos de la frecuencia respiratoria y en la derecha los datos de: oxigenación en la sangre, temperatura corporal y frecuencia respiratoria respectivamente, separados por "," para su posterior lectura en el ESP8266 NodeMCU.

Fig. 63. Medición con equipos comerciales y monitoreo en la aplicación.

Fuente: *Autor.*

El sensor de infrarrojo que se está utilizando tiene un valor de error de ± 0.5, pero aún así entrega valores aproximados a los termómetros de uso doméstico como se observa en la figura 63, por ende, se procede a hacer la validación del sistema con un profesional de la salud, en este caso un paramédico ubicado en el país Puerto Rico hacia el país Colombia, a través de videollamada registrada en la plataforma ZOOM como se puede ver en la figura 64.

(El monitor presente en esta imagen es una guía para el programador)

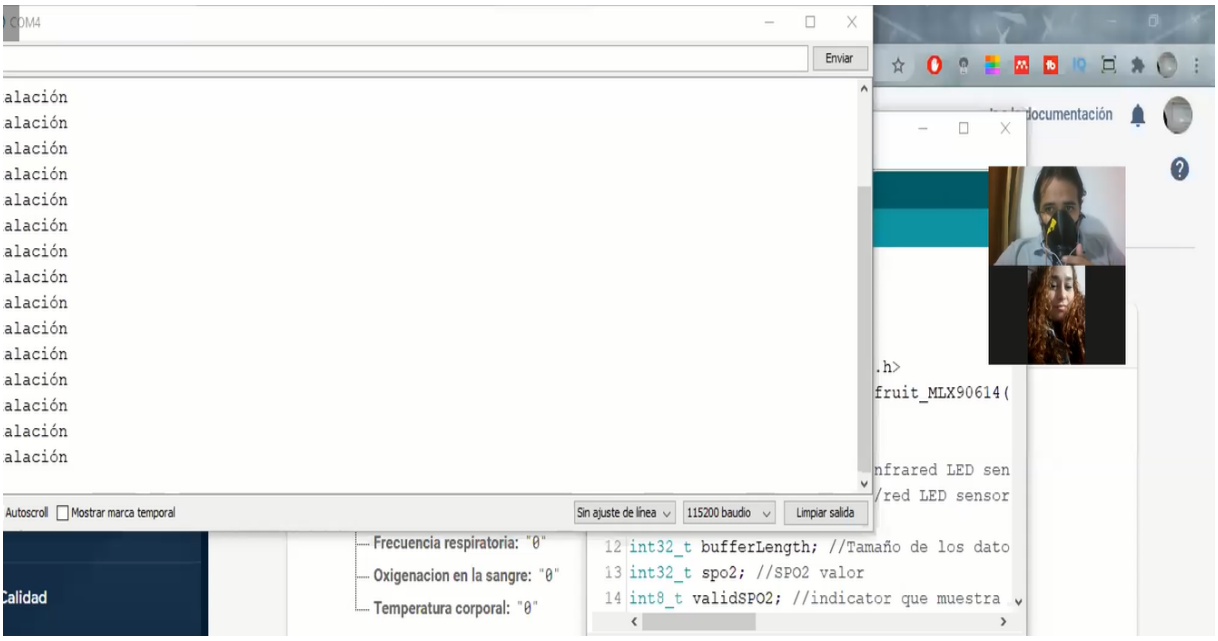


Fig. 64. Verificación de monitor en Colombia, vista desde puerto rico a través de videollamada en ZOOM.

Fuente: *Autor.*

El profesional (Paramédico) se encontraba verificando la frecuencia respiratoria en la videollamada y a través del monitor serial del Arduino, se podía notar que el sensor de frecuencia respiratoria tomaba los datos correspondientes entre los parámetros normales.

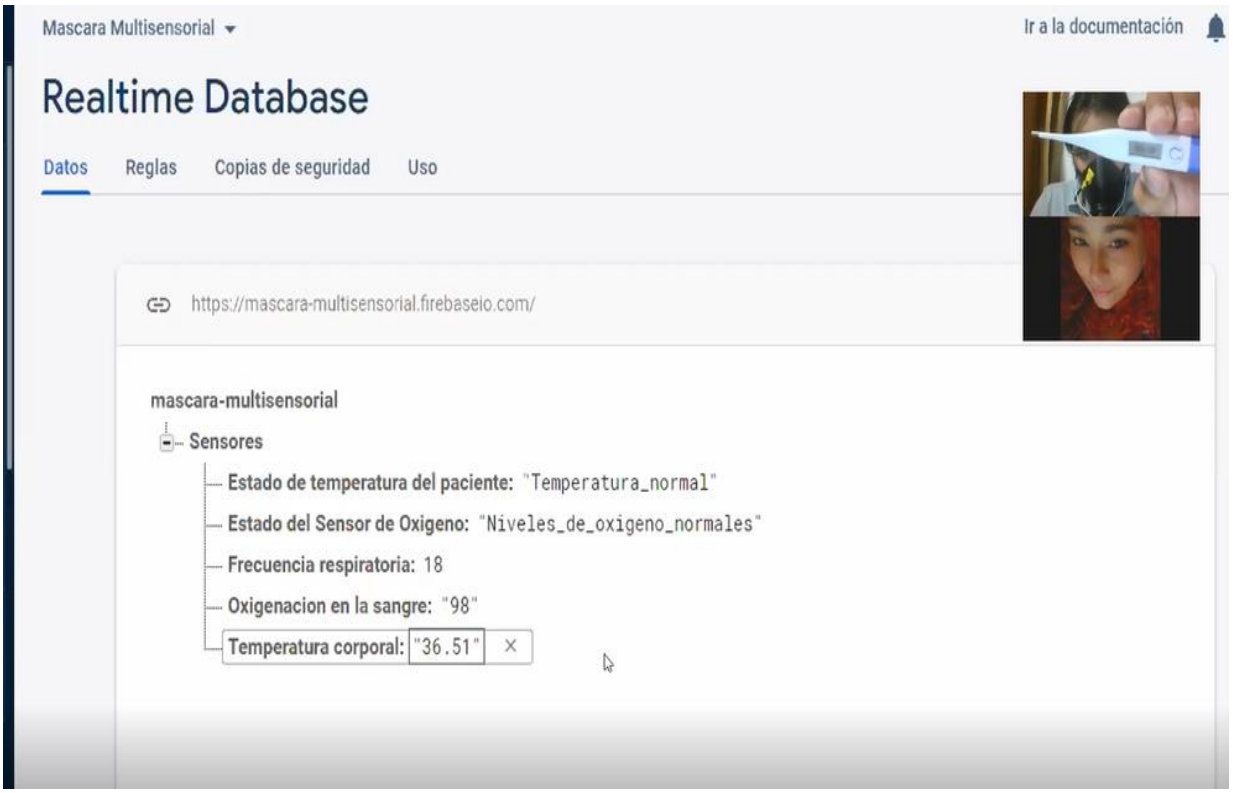


Fig. 65. Base de datos en tiempo real, vista desde ambos países.

Fuente: *Autor.*

Una vez que los datos fueron tomados y registrados en firebase como se observa en la figura 65, se procedió a verificar dichos valores con sensores comerciales, un termómetro MOKARWAY modelo T1 es el que puede notarse en la parte superior derecha de la pantalla, con este termómetro se valida el funcionamiento de la temperatura corporal medida por el prototipo.

Fig. 66. Verificación en la base de datos de los signos vitales comparando con equipos comerciales.

Fuente: *Autor.*

En la parte superior derecha de la figura 66, se puede notar el oxímetro MOKARWAY modelo M-01, con el cual se valida el funcionamiento de la oxigenación en la sangre medida y registrada en firebase por parte del sensor del prototipo.

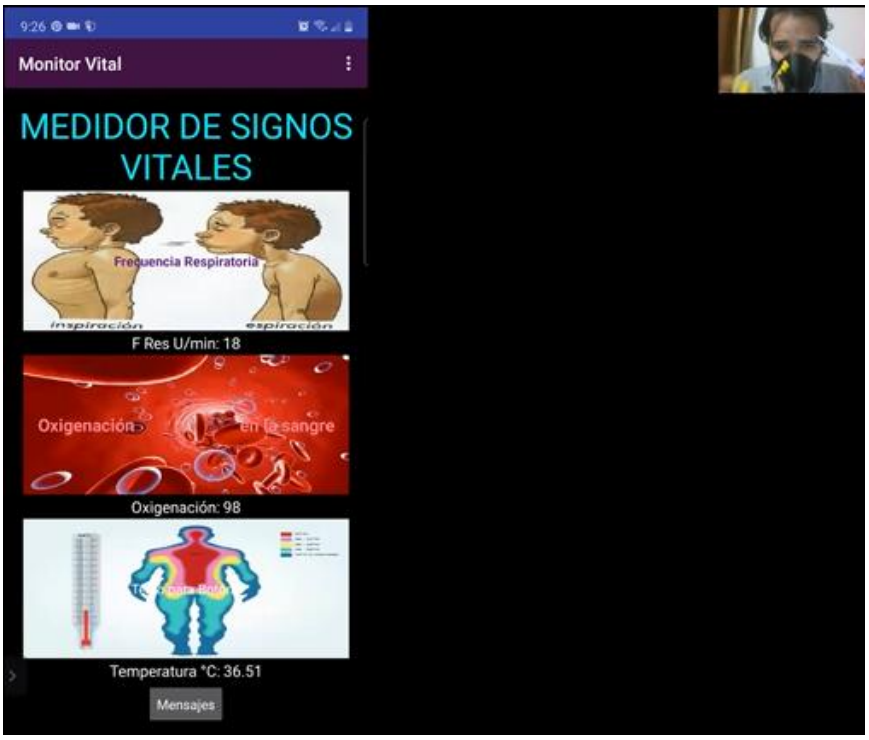


Fig. 67. Verificación de signos vitales en la aplicación móvil, desde Puerto Rico.

Fuente: *Autor.*

De esta manera se logra validar el funcionamiento del sistema, logrando establecer una monitorización de signos vitales, desde el país de Puerto Rico hacia el país de Colombia. La frecuencia respiratoria se encuentra en 18 ciclos por minuto, entre los rangos normales, la oxigenación que marcó el sistema fue de 98, el oxímetro comercial marcó 99, la temperatura de nuestro sistema marcó 36.51 y el termómetro de contacto 36.1, es decir, el sistema funciona entre los rangos normales como se puede observar en la figura 67.

4.4 DATOS ADICIONALES

El servicio de firebase guarda el registro de conexiones de usuarios que transcurren en el mes como se puede observar en la figura 66 (El paquete aquí contratado es gratuito y puede soportar hasta 100 usuarios).

Fig. 68. Número de conexiones en la base de datos.

Fuente: *Autor.*

En la figura 68 se puede notar el número de conexiones establecidas en la base de datos ubicada en firebase.

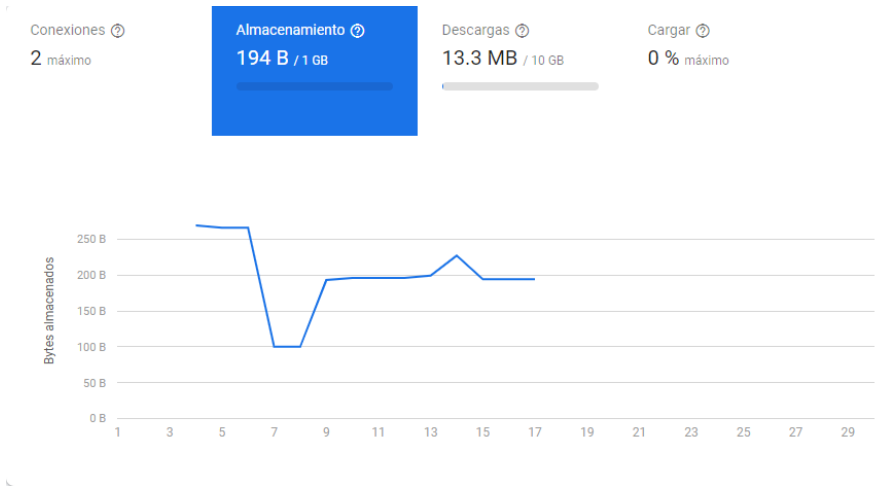


Fig. 69. Almacenamiento en la base de datos.

Fuente: *Autor.*

En la figura 69 se puede notar el almacenamiento de datos por parte de los sensores del prototipo y en la figura 70 se observa que cuando hay más de un usuario descargando datos de la base de datos, la descarga se multiplica.

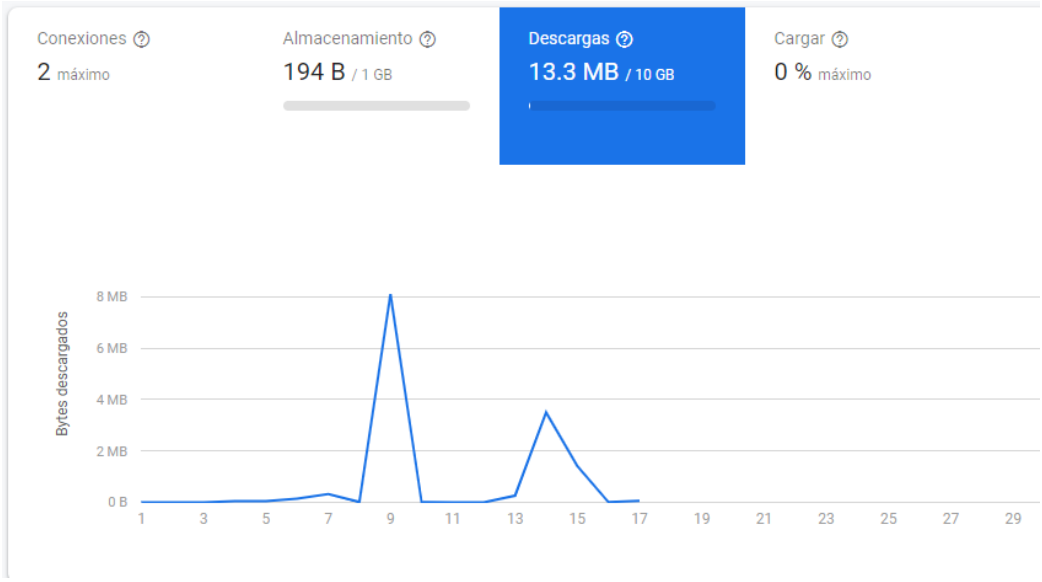


Fig. 70. Solicitudes en la base de datos.

Fuente: *Autor.*

De acuerdo con lo anterior, se utilizó el sistema 5 días seguidos, el primer día se cargó y descargó información en la base de datos más de una vez, para poder notar el impacto en el paquete gratuito de firebase con el que se cuenta en este proyecto, resultando en los valores de la figura 71:

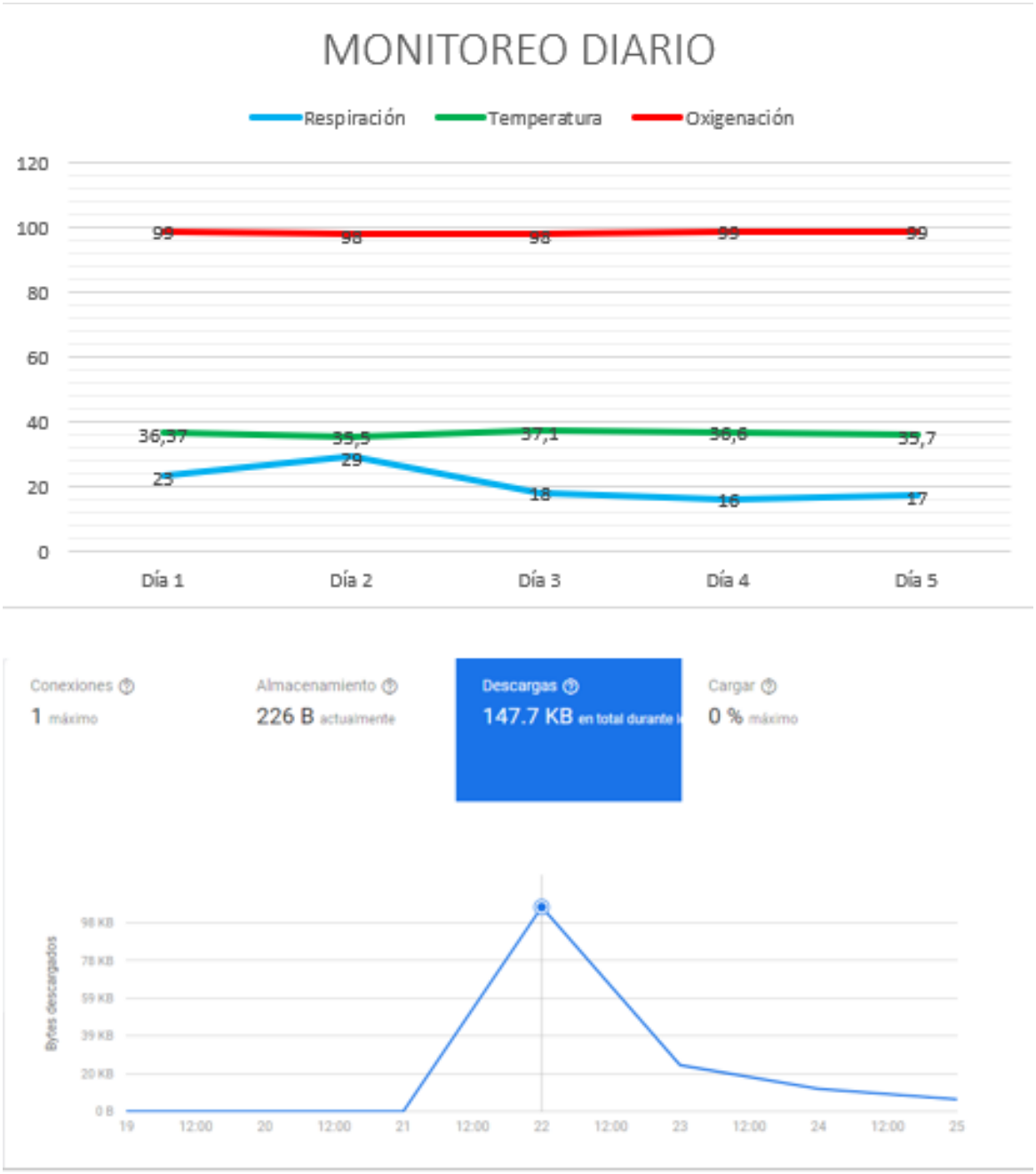


Fig. 71. Monitoreo diario y consumo en Firebase.

Fuente: *Autor.*

COMPARACIÓN Y PORCENTAJES DE ERROR DEL PATRÓN DE MEDIDA DE LA MÁSCARA CON EL PATRÓN DE MEDIDA DE SENSORES COMERCIALES.

TEMPERATURA MÁSCARA 3D	TEMPERATURA SENSOR COMERCIAL	OXIGENACIÓN MÁSCARA 3D	OXIGENACIÓN SENSOR COMERCIAL
36.5	36.1	99	99
35.5	35.8	99	99
37.1	37.3	98	99
36.6	36.5	98	99
35.7	36.1	99	99

Tabla 11 Medidas del sistema vs Medidas de sensores comerciales.

Fuente: *Autor.*

Una vez obtenido los datos y realizada la comparación de valores de temperatura y oxigenación con sensores comerciales, como se muestra en la tabla 11, se obtiene un porcentaje de error dada por la ecuación 5 y obteniendo los porcentajes de la tabla 12.

Ecuación 5 Porcentaje de error de medida de temperatura.

$$\frac{|Valor\ temperatura\ máscara - Valor\ temperatura\ comercial|}{Valor\ temperatura\ comercial} x100 \quad (5)$$

Ecuación 6 Porcentaje de error de medida de oxigenación

$$\frac{|Valor\ oxigenación\ máscara - Valor\ oxigenación\ comercial|}{Valor\ oxigenación\ comercial} x100 \quad (6)$$

% Error de temperatura	%Error de Oxigenación
1.10	0
0.83	0
0.53	1.01
0.27	1.01
1.10	0

Tabla 12 Porcentaje de error de medida

Fuente: *Autor.*

ANÁLISIS DE RESULTADOS CON RESPECTO A LOS ANTECEDENTES.

Al elaborar una revisión literaria de los temas a utilizar en este proyecto, se notó que además de los pilares en los que se desarrolla este prototipo, (oxigenación en la sangre, temperatura corporal y frecuencia respiratoria) pilares que sirvieron de guía y repercusión para todo este proyecto, hay que recordar, también podremos encontrar la inspiración y diferencias entre los antecedentes planteados en el marco teórico y nuestro prototipo culminado, desde la transcendencia de la investigación del oxímetro experimental de tipo reflectante, hasta las máscaras financiadas en páginas colaborativas, podemos notar que la impresión 3D puede ser utilizada en un sinfín de aplicaciones, como la presente y la antes mencionada en el artículo " MÁSCARA IMPRESA EN 3D A MEDIDA, EN PANDEMIA EN CASO DE SITUACIONES DE CRISIS CON CARENCIA DE MÁSCARAS FFP2/3 DISPONIBLES COMERCIALES" que podremos encontrar en el marco teórico y en la bibliografía, las máscaras mencionadas en los antecedentes, en las que este modelos se inspira son: C-MASK, LEAF MASK y CIVILITY, cabe mencionar que, aunque estas máscaras son una fuerte inspiración para este proyecto, se diferencian bastante de él, desde el diseño hasta la funcionabilidad de cada una, si tuviésemos que mencionar alguna característica común, esta sería la protección respiratoria en menor o mayor medida, pero al denotar las diferencias de cada máscara, podemos concluir que cada máscara se diferencia entre sí, aunque se podría utilizar características de cada una de ellas, por ejemplo la superficie hidrofóbica de la máscara **LEAF MASK**, característica con la que no cuenta nuestro prototipo, los ventiladores de la máscara **CIVILITY** que

aumentan la transpirabilidad dentro de la máscara, pero no pueden medir la frecuencia respiratoria como lo hace nuestro prototipo a pesar de no contar con ventilador alguno o el factor social de la máscara **C-MASK**, máscara que permite amplificar el sonido de la voz y traducir idiomas, pero no cuenta con medición de signos vitales de ningún tipo, como lo hace nuestro prototipo, diferenciándose así de todas las demás máscaras debido a su factor de medición de signos vitales.

5. CONCLUSIONES.

Al establecer un monitoreo de signos vitales se debe tener en cuenta la precisión de los sensores a utilizar, la precisión para la aplicación requerida y costo, si queremos hacer una aplicación de uso clínico es decir no domiciliario y no como dispositivos wearables los costos de los dispositivos y la precisión de los sensores será más alta, debido a que de esto depende la vida de una persona.

Atendiendo al debido análisis realizado de la implementación de sistemas embebidos en sistemas electrónicos de monitoreo, tanto en su software como en su hardware se proponen mejoras para aplicaciones específicas, sugiriendo la inclusión de embebidos con afinidad de aplicación o demás temas relacionados, así facilitaría la creación de nuevos sistemas que pueden convertirse en tendencia, al relacionar y entrelazar el internet de las cosas con las aplicaciones de biomédica.

La comunicación médica/paciente es importante, este prototipo podría llegar a tener varias aplicaciones que podrían ayudar a los médicos a generar diagnósticos en tiempo real, podría ser utilizado en pacientes con problemas en su patrón respiratorio y podría llevarse una bitácora organizada de su evolución, como por ejemplo los pacientes con COPD, es decir, pacientes con enfermedades en las vías aéreas, en algunas ocasiones para estos pacientes está contra indicado usar oxígeno en su tratamiento, ya que su nivel de oxigenación en la sangre es más bajo de lo normal.

La tecnología de los diseños 3D e impresoras 3D ha avanzado lo suficiente como para poder imprimir casi lo que sea, por ende, se deben aprovechar todos los recursos que se puedan, existe software gratuito de modelado 3D con suficientes prestaciones para elaborar diseños complejos sin consumir tantos recursos físicos propios, pero también se encuentran limitados algunos de ellos, en el caso de tinkercad no podemos generar planos en PDF de las piezas que construimos, pero si podemos exportar objetos y archivos STL que pueden ser impresos posteriormente, de las medidas con las que trabajamos en dicho software.

Dependiendo de qué tan versátil es un sensor se pueden suplir aplicaciones que no se habían planteado antes, al querer medir la oxigenación en un lóbulo de la oreja, para poder encapsular toda la electrónica de la máscara en un solo sitio, el sensor de oximetría que utilizamos no puede cumplir con dicha tarea, pero es probable que con otros arreglos de sistemas embebidos se pueda llegar a satisfacer, por ejemplo, un sensor con un microcontrolador integrado para este propósito, o un sistema embebido dedicado a aplicaciones de este tipo de temas de la salud, al contar con un embebido de estas características podría ser incluido fácilmente en dispositivos como el nuestro en cuestión.

El internet nos permite acortar brechas de comunicación, la telemedicina es una realidad, en este tiempo donde el contacto físico debe ser fuertemente limitado, nos vemos en la necesidad de contar con aplicaciones de esta índole, este proyecto puede abrir contribuir a una posible tendencia del internet, la medicina y los dispositivos wearables.

6. SUGERENCIAS

Teniendo en cuenta que el diseño aquí culminado es un prototipo y con la intención de ahondar más a futuro, con respecto a la selección de los sensores, se recomienda la implementación de sensores más precisos para obtener mejores resultados, mayor certeza en la medición y poder llegar a tener un uso más crítico y confiable.

El diseño del prototipo, al ser una máscara facial podría querer utilizarse en exteriores, por ende, el sistema electrónico puede llegar a compactarse más al imprimir un solo circuito preferiblemente flexible en China, para poder combinarlo con una máscara flexible.

El ambiente virtual se presta para aplicaciones cada vez más complejas, el software está en constante actualización, lo que conlleva a trabajar constantemente en actualización de aplicaciones que fueron creadas previamente, se recomienda actualizar la versión de la aplicación móvil e incluir el desarrollo para la empresa Apple, esto haría que se abarcaran más dispositivos, como también la inclusión de más pacientes en la app, de esta manera, un profesional de la salud podría monitorizar a diferentes pacientes en diferentes ubicaciones.

Se recomienda, para futuras impresiones 3D, realizar la materialización del diseño 3D en material flexible y especial para estar en contacto con la piel siendo higiénico, así podría adaptarse este prototipo a diferentes rostros y lo haría de manera más cómoda.

Se recomienda desinfectar con alcohol el dispositivo cada vez que se desee utilizar, hervir los filtros y seguir las recomendaciones de su médico, respecto a sus signos vitales.

Nota: esto es un prototipo y bajo ningún concepto reemplaza a un profesional de la salud.

7. BIBLIOGRAFÍA

[1] E. M. G. Rodrigues, R. Godina, C. M. P. Cabrita, and J. P. S. Catalão, "Experimental low cost reflective type oximeter for wearable health systems," *Biomedical Signal Processing and Control*, vol. 31, pp. 419–433, Jan. 2017, doi: 10.1016/j.bspc.2016.09.013.

[2] G. R. J. Swennen, L. Pottel, and P. E. Haers, "Custom-made 3D-printed face masks in case of pandemic crisis situations with a lack of commercially available FFP2/3 masks," *International Journal of Oral and Maxillofacial Surgery*, vol. 49, no. 5, pp. 673–677, May 2020, doi: 10.1016/j.ijom.2020.03.015.

[3] "smart mask | mysite." https://www.donutrobotics.com/c-mask (accessed Sep. 13, 2020).

[4] "Leaf Healthcare." https://www.leaf.healthcare/ (accessed Sep. 13, 2020).

[5] "CIVILITY - Next Gen Transparent Mask | Indiegogo." https://www.indiegogo.com/projects/civility-next-gen-transparent-mask?fbclid=IwAR0Kkgtn8Nsgf4VlyCt4ipIzz4fKQ9KASpWUtR9oRWmvdScWzEiFzv64weY#/ (accessed Sep. 13, 2020).

[6] "Inicio - Dispositivo de protección para el personal de la salud." https://www.easy-co.org/ (accessed Sep. 13, 2020).

[7] O. O. Fadare and E. D. Okoffo, "Covid-19 face masks: A potential source of microplastic fibers in the environment," *Science of the Total Environment*, vol. 737, Oct. 2020, doi: 10.1016/j.scitotenv.2020.140279.

[8] S. Jia, G. Guo, and Z. Xu, "A survey on 3D mask presentation attack detection and

," *Pattern Recognition*, vol. 98, Feb. 2020, doi:

10.1016/j.patcog.2019.107032.

[9] G. R. J. Swennen, L. Pottel, and P. E. Haers, “Custom-made 3D-printed face masks in case of pandemic crisis situations with a lack of commercially available FFP2/3 masks,” *International Journal of Oral and Maxillofacial Surgery*, vol. 49, no. 5, pp. 673–677, May 2020, doi: 10.1016/j.ijom.2020.03.015.

[10] D. Wang et al., “Can Masks Be Reused After Hot Water Decontamination During the COVID-19 Pandemic?,” Engineering, 2020, doi: 10.1016/j.eng.2020.05.016.

[11] “CLIU Mask - N99, BIO, Antimicrobial & A.C. Filters.” https://cliu.it/es/ (accessed Sep. 07, 2020).

[12] F. Torres and C. Fernández, “Sensores y detectores,” *Automática, Robótica y Visión Artificial*, p. 63, 2014.

[13] Drägerwerk AG & Co. KGaA, “La importancia de la temperatura corporal central Fisiopatología y métodos de medición,” p. 60, 2018.

[14] N. C. Dario Londoño Trujillo. MD, MSc, Rafael Acero Colmenraes, auderyPiotrostanalky, “Uso E Interpretación Pulso,” *Colombia. Ministerio de Salud y Protección Social de salud colombia*, vol. Convenio 5, pp. 6–13, 2016, [Online]. Available: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/VS/PP/ENT/uso-interprtn-oximetria-pulso.pdf.

[15] J. J. Talamas Márquez, “Habilidades Básicas III Toma de signos vitales,” *Universidad Juarez del Estado de Durango*, vol. 1, no. 1, p. 14, 2016, [Online]. Available: http://famen.ujed.mx/doc/manual-de-practicas/a-2016/03_Prac_01.pdf.

[16] A. E. Pérez, “Diseño de aplicación móvil para la comunicación inalámbrica de señales audiovisuales,” p. 4, 2013.

[17] A. E. Pérez, “Diseño de aplicación móvil para la comunicación inalámbrica de señales audiovisuales,” p. 4, 2013.

[18] A. Flores, “Sistema electrónico de comunicación paciente-personal,” 2015.

[19] I. T. de Queretaro., “Comunicaciones Digitales: Protocolos seriales (uC) ¿Qué es la comunicación serial ?,” 2019.

[20] A. Uno and I. M. C. I. Ltda., “Comunicación Serial.”

[21] M. Araya, “Sensores De Temperatura: Sensores De Temperatura,” *April 27, 2015*, pp. 4–5, 2015, [Online]. Available: http://snsoresdetemperatura.blogspot.com/2009/05/sensores-de-temperatura.html.

[22] J. Blum, “ The I 2 C Bus ,” *Exploring Arduino®*, no. October, pp. 199–221, 2019, doi: 10.1002/9781119405320.ch10.

[23] D. Wang et al., “Can Masks Be Reused After Hot Water Decontamination During the COVID-19 Pandemic?,” Engineering, 2020, doi: 10.1016/j.eng.2020.05.016.

ANEXOS

ANEXO 1

CÓDIGO INCORPORADO EN ARDUINO MEGA 2560

```
#include "FirebaseESP8266.h"//Librería firebase
#include <ESP8266WiFi.h> //Librería de control NodeMCU
#include <Ticker.h> //función de tiempo
Ticker tiempo_1;
#include <Wire.h> // Librería wire
#include "MAX30105.h"// Librería Max30105(Max30102)
#include "spo2_algorithm.h"// Librería SpO2
MAX30105 particleSensor;// Iniciar sensor
#include <Adafruit_MLX90614.h>// Librería MLX90614
Adafruit_MLX90614 mlx = Adafruit_MLX90614(); //llamada a la
Función del sensor

#define MAX_BRIGHTNESS 255 // define el brillo máximo del LED
uint32_t irBuffer[100]; //LED infrarrojo
uint32_t redBuffer[100];  //LED rojo sensor data

int32_t bufferLength; //Tamaño de los datos
int32_t spo2; //SPO2 valor
int8_t validSPO2; //indicator que muestra si el SPO2 calculado
es valido
```

```
  int32_t heartRate; //valor de ritmo cardiaco
  int8_t validHeartRate; //indicator que muestra si el ritmo
cardiaco calculado es valido


  byte readLED = 13; //pestañea con cada lectura
  int Oxygen = 9; //salida del porcentaje de oxigenación
  int indicadorRes=40;
  int indicadorOxi=42;
  int cont;
  int d;
  int cont1;
  int conta1;
  int Vo;
  int c = 0;


  float R1 = 100000;// resistencia fija del divisor de tensión
  float logR2, R2, TEMPERATURA;
  float c1 = 2.114990448e-03, c2 = 0.3832381228e-04, c3 =
5.228061052e-07; //Coeficientes de S-H
  //Declaración de variables
  float actual=0;
  float anterior=0;
  float Oxi;
  float Ob;


  unsigned long tiempo;
```

```
  unsigned long tiempo2=0;


  String dato1;
  String conteoresp;
  String dato;
  String dato2;
  String signos;



  void setup(){
  //Declaración de puertos seriales y modos de pin
    Serial.begin(115200);
    Serial3.begin(115200);
    mlx.begin();
    pinMode(Oxygen, OUTPUT);
    pinMode(pulseLED, OUTPUT);
    pinMode(readLED, OUTPUT);
    pinMode(indicadorRes, OUTPUT);
    pinMode(indicadorOxi, OUTPUT);



    // Initializar sensor
    if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //Usa el
puerto I2C por defecto, 400kHz velocidad
    {
```

```
      Serial.println(F("MAX30105 was not found. Please check wiring/power."));
      while (1);
    }

    Serial.println(F("Coloque el sensor en el dedo y presione alguna tecla"));
    while (Serial.available() != 0) ; //esperando a el usuario
    Serial.read();

    byte ledBrightness = 60; //Opciones: 0=Off a 255=50mA //brillo del led rojo
    byte sampleAverage = 4; //Opciones: 1, 2, 4, 8, 16, 32 //promedio de la muestra
    byte ledMode = 2; //Opciones: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green// modo de los leds
    byte sampleRate = 100; //Opciones: 50, 100, 200, 400, 800, 1000, 1600, 3200 //frecuencia de muestreo
    int pulseWidth = 411; //Opciones: 69, 118, 215, 411  //ancho del pulso
    int adcRange = 4096; //Opciones: 2048, 4096, 8192, 16384 //Rango del ADC

    particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange); //Se configura el sensor

   mlx.begin();
  }
```

```
  void loop(){
   //Cronómetro
    if(millis()-tiempo>=1000){
      tiempo2=tiempo2+1;
      tiempo=millis();
      if(tiempo2>=240){
      tiempo2=0;
      }
     }
  digitalWrite(indicadorRes,LOW);
  digitalWrite(indicadorOxi,LOW);


  if (tiempo2<=60){//Durante el primer minuto de encendido el
sistema se tomarán las muestras de respiración


    TEMPERATURA=obtenerTemp();//Se hace llamada a la función
obtenerTemp
    actual=TEMPERATURA;


          if(actual>anterior+0.5)
          {anterior=actual;
            c=1;
            }
          if(actual<anterior-0.5){
            anterior=actual;
            c=0;
```

```
            }
           if(c!=d){
            cont=cont+1;
            d=c;
            }
  digitalWrite(indicadorRes,HIGH);//Cuando se esté tomando
muestras de respiración el indicador de respiración está ON y el
de Oxigeno OFF
  digitalWrite(indicadorOxi,LOW);
  }
   if(tiempo2>=90){//Las muestras tomadas se borraran luego de
90seg
    cont1=cont/2;
    cont=0;
     }
    if(tiempo2>=61&&tiempo2<=64){//La función de tomar muestras
de oxigenación demora 4 segundos
      digitalWrite(indicadorRes,LOW);//Cuando se esté tomando
muestras de respiración el indicador de respiración está OFF y el
de Oxigeno ON
      digitalWrite(indicadorOxi,HIGH);
    bufferLength = 100; //la longitud del búfer de 100 almacena 4
segundos de muestras que se ejecutan a 25 muestras por segundo


    //lee las primeras 100 muestras y determina el rango de la
señal
    for (byte i = 0 ; i < bufferLength ; i++)
    {
```

```
      while (particleSensor.available() == false) //pregunta por
nueva data
        particleSensor.check(); //revisa el sensor por nueva data

      redBuffer[i] = particleSensor.getRed(); //Led Rojo
      irBuffer[i] = particleSensor.getIR(); //Led infrarojo
      particleSensor.nextSample(); //se termina con esta muestra
y se procede a la siguiente
    }

    //calcula ritmo cardiaco y SpO2 despues de las primeras 100
muestras (primeros 4s de muestras)
   maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength,
redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);

       //Tomando muestras continuamente de MAX30102. La
frecuencia cardíaca y la SpO2 se calculan cada segundo
      //colocando los primeros 25 conjuntos de muestras en la
memoria y desplazando los últimos 75 conjuntos de muestras al
principio
      for (byte i = 25; i < 100; i++)
      {
        redBuffer[i - 25] = redBuffer[i];
        irBuffer[i - 25] = irBuffer[i];
      }

      //se toma 25 conjuntos de muestras antes de calcular la
frecuencia cardíaca
```

```
      for (byte i = 75; i < 100; i++)
      {
        while (particleSensor.available() == false) //pregunta
por nueva data
          particleSensor.check(); //revisa el sensor por nueva
data
        digitalWrite(readLED, !digitalRead(readLED));//Parpadea
el LED integrado con cada lectura de datos
        redBuffer[i] = particleSensor.getRed();
        irBuffer[i] = particleSensor.getIR();
        particleSensor.nextSample(); //se termina con esta
muestra y se procede a la siguiente
          //Se pregunta por el
          if(validSPO2==0){
            dato2="invalido";
          }else{
          if(spo2>=92 && validSPO2==1){
            Oxi=spo2;
            dato=String(Oxi);
            dato2=dato;
          }
           if(spo2<=91 && validSPO2==1){
            conta1=conta1+1;
             if(conta1>10){
              cont1=0;

dato2="SPO2=_por_debajo_del_92%,_Recurra_a_consulta_médica";
```

```
            }
          }
         }
          Ob=mlx.readObjectTempC(); // Medición de la temperatura
con la librería <Adafruit_MLX90614.h>
          dato1=String(Ob);
       }
      //Después de recolectar 25 nuevas muestras, recalcular HR y
SP02
      maxim_heart_rate_and_oxygen_saturation(irBuffer,
bufferLength, redBuffer, &spo2, &validSPO2, &heartRate,
&validHeartRate);
    //}
    } int serial=digitalRead(31);
         if(serial==1){
          conteoresp=String(cont);

signos=String(dato2+',')+String(dato1+',')+String(conteoresp);
//Agrupación de los 3 datos de signos vitales para posterior
envío
          Serial3.println(signos); }//Envío de datos por el
serial del puerto 3 del Arduion MEGA
  }


  float obtenerTemp(){
    Vo=analogRead(A0);  // lectura de A0
    R2=R1*(1023.0 /(float)Vo-1.0); // conversion de tension a
resistencia
```

```
    logR2=log(R2);  // logaritmo de R2 necesario para ecuacion
    TEMPERATURA=(1.0/(c1+c2*logR2+c3*logR2*logR2*logR2));   // ecuacion S-H
    TEMPERATURA=TEMPERATURA-273.15;   // Kelvin a Centigrados (Celsius)
    return TEMPERATURA;
  }
```

ANEXO 2

CÓDIGO INCORPORADO EN ESP8266 NODEMCU

```
  #include "FirebaseESP8266.h"
  #include <ESP8266WiFi.h>
  #include <Ticker.h>
  Ticker tiempo_1;


  #define FIREBASE_HOST "mascara-multisensorial.firebaseio.com" //Sin http:// o https:// enlace de la base de datos
  #define FIREBASE_AUTH "T52G4unSkXBKFV4ecydbOkg5AyxKXEsLW3MMJBny" //Token de firebase
  #define WIFI_SSID "Nombre de la red"
  #define WIFI_PASSWORD "Contraseña"
  #include <Separador.h> //librería para separar carácteres
  String path = "/Sensores"; //creación del path Sensores
```

```
//Define un objeto de Firebase
FirebaseData firebaseData;
//Declaración de variables y funciones
void printResult(FirebaseData &data);
void CausaError(void);
void InforSetLuzSensor(void);
void InforGetLuzSensor(void);
unsigned long tiempo;
unsigned long tiempo1;
unsigned long tiempo2=0;
int cont;
float ValorMedidoOxigenacion=0;
float ValorMedidoTemperatura=0;
float ValorMedidoFrecuencia=0;
String ValorMensaje;
String ValorMensaje1;
String Oxigenacion="0";
String Temperatura="0";
String Frecuencia="0";
boolean Boolenvio=false;

int led=13;
int estadoled=LOW;
Separador s;
void setup() {
```

```
  //inicialización del serial
   Serial.begin(115200);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Conectando a ....");
    while (WiFi.status() != WL_CONNECTED)
    {
      Serial.print(".");
      delay(300);
    }
    Serial.println();
    Serial.print("Conectado con la IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);//conexión a
Firebase
    Firebase.reconnectWiFi(true);

    //Establezca el tiempo de espera de lectura de la base de
datos en 1 minuto (máximo 15 minutos)
    Firebase.setReadTimeout(firebaseData, 1000 * 60);

    //Tamaño y  tiempo de espera de escritura, tiny (1s), small
(10s), medium (30s) and large (60s).
    //tiny, small, medium, large and unlimited.
    Firebase.setwriteSizeLimit(firebaseData, "tiny");
```

```
  //Creación de paths de información

    if (Firebase.setString(firebaseData, path + "/Temperatura
corporal", Temperatura)){InforSetLuzSensor();}else{CausaError();}

    if (Firebase.setString(firebaseData, path + "/Oxigenacion en
la sangre",
Oxigenacion)){InforSetLuzSensor();}else{CausaError();}

    if (Firebase.setString(firebaseData, path + "/Frecuencia
respiratoria",
Frecuencia)){InforSetLuzSensor();}else{CausaError();}

    if (Firebase.setString(firebaseData, path + "/Estado del
Sensor de Oxigeno",
ValorMensaje)){InforSetLuzSensor();}else{CausaError();}

    if (Firebase.setString(firebaseData, path + "/Estado de
temperatura del paciente",
ValorMensaje1)){InforSetLuzSensor();}else{CausaError();}

      Boolenvio=false;

    tiempo_1.attach(10, funcion_1);// función de envío a firebase
cada 10 seg

    pinMode(led,OUTPUT);

    digitalWrite(led,LOW);


  }

  void loop() {

    Serial.println("-------------------------------------");

    Serial.println("ACTUALIZAR EL ESTADO DE LOS SENSORES");


    Serial.println(F("Coloque el sensor en el dedo y presione
alguna tecla"));
```

```
    while (Serial.available() == 0) ; //esperando a el usuario
    Serial.read();
     String signos=Serial.readString();
     Serial.println(signos);
       String Oxigenacion=s.separa(signos,',',0);
       String Temperatura=s.separa(signos,',',1);
       String Frecuencia=s.separa(signos,',',2);
     Serial.println(Oxigenacion);
     Serial.println(Temperatura);
     Serial.println(Frecuencia);
       estadoled=LOW;
    if (Boolenvio==true){
        Boolenvio==false;
        estadoled=HIGH;




       if(Oxigenacion=="invalido")
     {
        ValorMensaje="invalido";
        if (Firebase.setString(firebaseData, path + "/Estado del
Sensor de Oxigeno",
ValorMensaje)){InforSetLuzSensor();}else{CausaError();}
      }
```

```
    char carray2[Oxigenacion.length() + 2];// tamaño del array
        Oxigenacion.toCharArray(carray2, sizeof(carray2)); //Pone
Oxigenacion en un Array
        ValorMedidoOxigenacion=atof(carray2); //Convierte el
array en una variable float
        Serial.println(ValorMedidoOxigenacion);
        if(ValorMedidoOxigenacion>=92){
          ValorMensaje="Niveles_de_oxigeno_normales";
         if (Firebase.setString(firebaseData, path + "/Estado del
Sensor de Oxigeno",
ValorMensaje)){InforSetLuzSensor();}else{CausaError();}
       }
       if(ValorMedidoOxigenacion>=92){
        if (Firebase.setString(firebaseData, path + "/Oxigenacion
en la sangre",
Oxigenacion)){InforSetLuzSensor();}else{CausaError();}
       }

if(Oxigenacion=="SPO2=_por_debajo_del_92%,_Recurra_a_consulta_méd
ica")
     {  ValorMensaje=Oxigenacion;
        if (Firebase.setString(firebaseData, path + "/Estado del
Sensor de Oxigeno",
ValorMensaje)){InforSetLuzSensor();}else{CausaError();}
      }
    char carray1[Temperatura.length() + 1];// tamaño del array
    Temperatura.toCharArray(carray1, sizeof(carray1)); //Pone
Temperatura en un Array
```

```
    ValorMedidoTemperatura=atof(carray1); // Convierte el array
en una variable float
    Serial.println(ValorMedidoTemperatura);


     if(ValorMedidoTemperatura>=30){
         if (Firebase.setString(firebaseData, path +
"/Temperatura corporal",
Temperatura)){InforSetLuzSensor();}else{CausaError();}
         }
     if(ValorMedidoTemperatura<=37.5){
           ValorMensaje1="Temperatura_normal";
         if (Firebase.setString(firebaseData, path + "/Estado de
temperatura del paciente",
ValorMensaje1)){InforSetLuzSensor();}else{CausaError();}
          }
     if(ValorMedidoTemperatura>=37.6){
            ValorMensaje1="FIEBRE";
         if (Firebase.setString(firebaseData, path + "/Estado de
temperatura del paciente",
ValorMensaje1)){InforSetLuzSensor();}else{CausaError();}}
    char carray3[Frecuencia.length() + 1];// tamaño del array
    Frecuencia.toCharArray(carray3, sizeof(carray3)); //Pone
Oxigenacion en un Array
    ValorMedidoFrecuencia=atof(carray3); //Convierte el array en
una variable float
    if(ValorMedidoFrecuencia>1){
```

```
       if (Firebase.setFloat(firebaseData, path + "/Frecuencia
respiratoria",
ValorMedidoFrecuencia)){InforSetLuzSensor();}else{CausaError();}
      }
    Serial.println("-----------------------------------");
    Serial.println("ACTUALIZAR EL ESTADO DE LOS SENSORES");


   // if (Firebase.setFloat(firebaseData, path + "/Temperatura
corporal",
ValorMedidoTemperatura)){InforSetLuzSensor();}else{CausaError();}
  //  if (Firebase.setFloat(firebaseData, path + "/Oxigenacion en
la sangre",
ValorMedidoOxigenacion)){InforSetLuzSensor();}else{CausaError();}
  //  if (Firebase.setFloat(firebaseData, path + "/Frecuencia
respiratoria",
ValorMedidoFrecuencia)){InforSetLuzSensor();}else{CausaError();}


    Serial.println("-----------------------------------");
    Serial.println("   LEER EL ESTADO DE LOS SENSORES   ");
    if (Firebase.getString(firebaseData, path + "/Temperatura
corporal" )){InforGetLuzSensor(); }else{CausaError(); }
    if (Firebase.getString(firebaseData, path + "/Oxigenacion en
la sangre")){InforGetLuzSensor(); }else{CausaError(); }
    if (Firebase.getString(firebaseData, path + "/Frecuencia
respiratoria" )){InforGetLuzSensor(); }else{CausaError(); }
    if (Firebase.getString(firebaseData, path + "/Estado del
Sensor de Oxigeno" )){InforGetLuzSensor(); }else{CausaError(); }
```

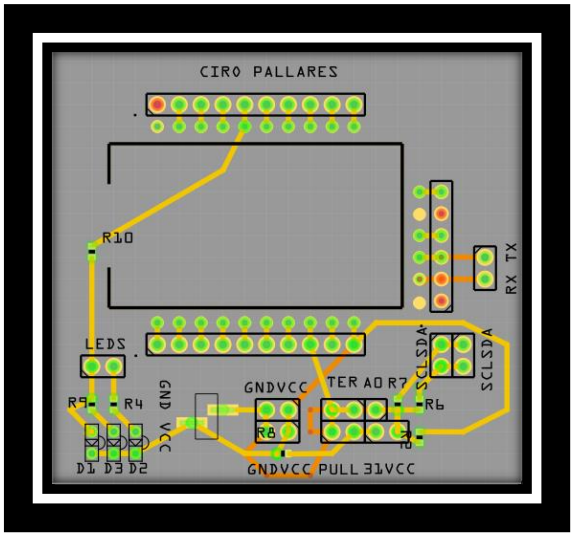
```
    if (Firebase.getString(firebaseData, path + "/Estado de
temperatura del paciente" )){InforGetLuzSensor();
}else{CausaError(); }
    if (Firebase.getString(firebaseData, path + "/Oxigenacion en
la sangre")){InforGetLuzSensor(); }else{CausaError(); }
   // delay(1000);
     digitalWrite(led,estadoled);
    }
    digitalWrite(led,estadoled);

    if(Serial.available()){
      serialEvento();
      }
  }
  void serialEvento(){
    String signos=Serial.readString();
    String Oxigenacion=s.separa(signos,',',0);
    String Temperatura=s.separa(signos,',',1);
    String Frecuencia=s.separa(signos,',',2);
    Serial.println("La oxigenacion es:"+Oxigenacion);
    Serial.println("La temperatura es:"+Temperatura);
    Serial.println("La frecuencia es:"+Frecuencia);
    }

  void InforGetLuzSensor(void)
  {
```

```
  Serial.println("Aprobado");
  Serial.println("Ruta: " + firebaseData.dataPath());
  Serial.println("Tipo: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
  Serial.print("Valor: ");
  printResult(firebaseData);
  Serial.println("------------------------------------");
  Serial.println();
}
void InforSetLuzSensor(void)
{
  Serial.println("Aprobado");
  Serial.println("Ruta: " + firebaseData.dataPath());
  Serial.println("Tipo: " + firebaseData.dataType());
  Serial.println("ETag: " + firebaseData.ETag());
  Serial.print("Valor: ");
  printResult(firebaseData);
  Serial.println("------------------------------------");
  Serial.println();
}

void CausaError(void)
{
  Serial.println("ERROR");
  Serial.println("RAZON: " + firebaseData.errorReason());
```

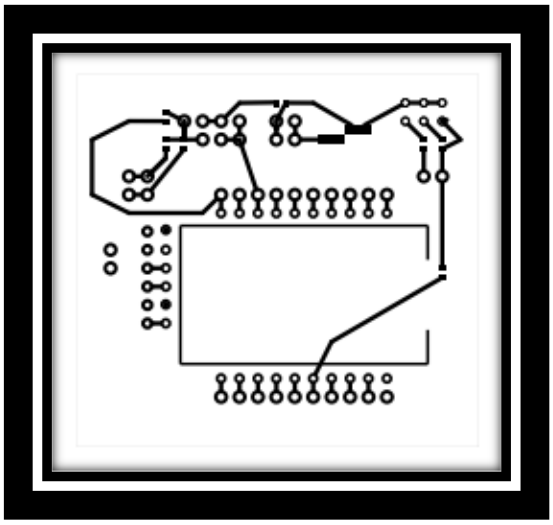
```
    Serial.println("------------------------------------");
    Serial.println();
  }


  void printResult(FirebaseData &data)
  {
      if (data.dataType() == "int")
          Serial.println(data.intData());
      else if (data.dataType() == "float")
          Serial.println(data.floatData(), 5);
      else if (data.dataType() == "double")
          printf("%.9lf\n", data.doubleData());
      else if (data.dataType() == "boolean")
          Serial.println(data.boolData() == 1 ? "true" :
"false");
      else if (data.dataType() == "string")
          Serial.println(data.stringData());
  }
  void funcion_1(void){
    Boolenvio=true;}
```

ANEXO 3

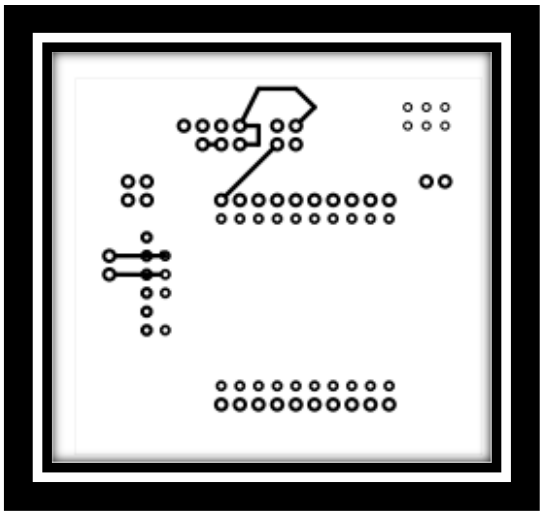
PCB COMPLETA.



PCB CAPA SUPERIOR.



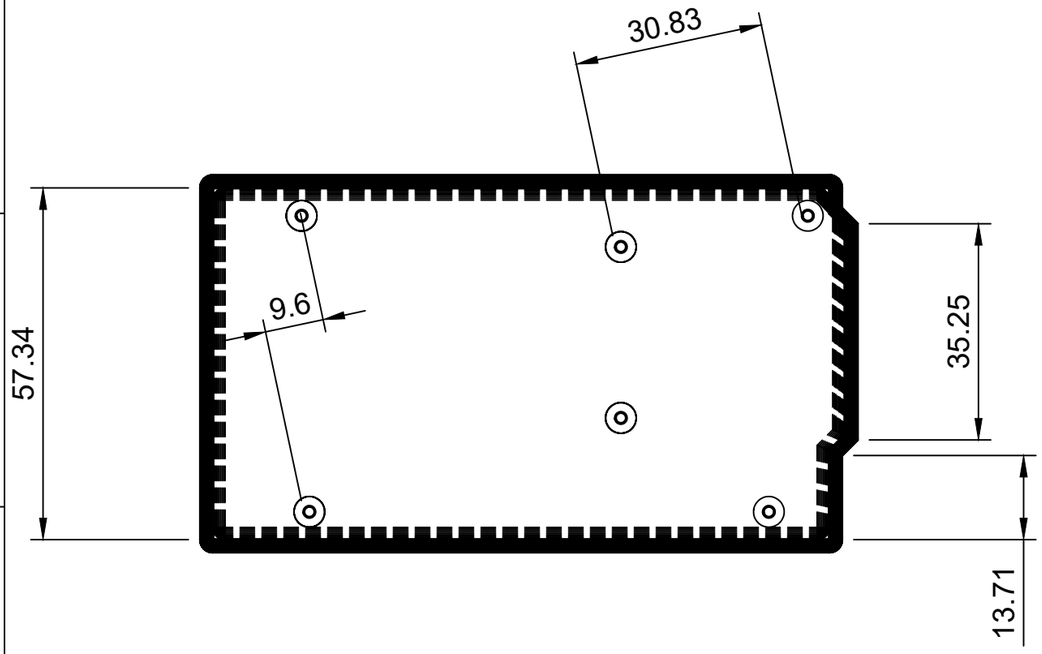
PCB CAPA INFERIOR.

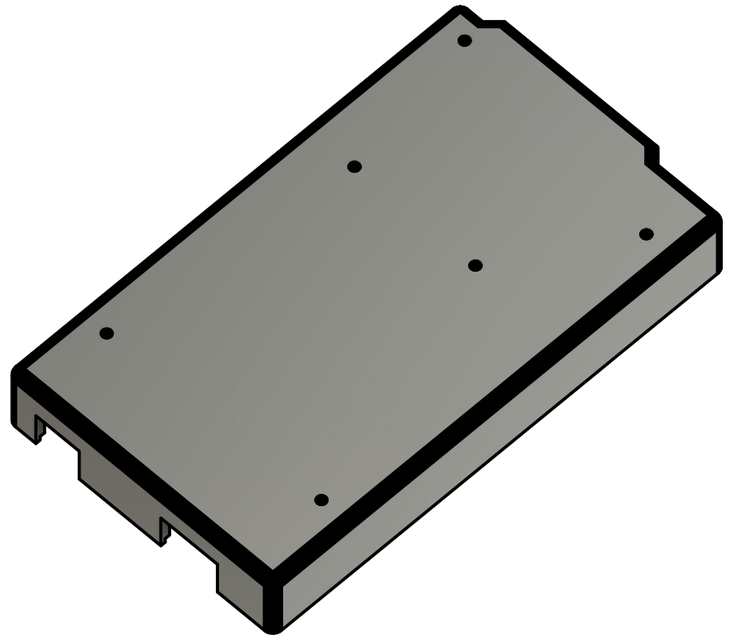


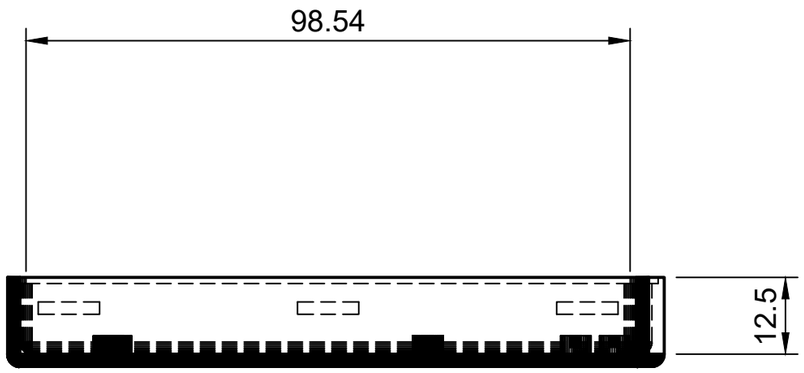
ANEXO 4

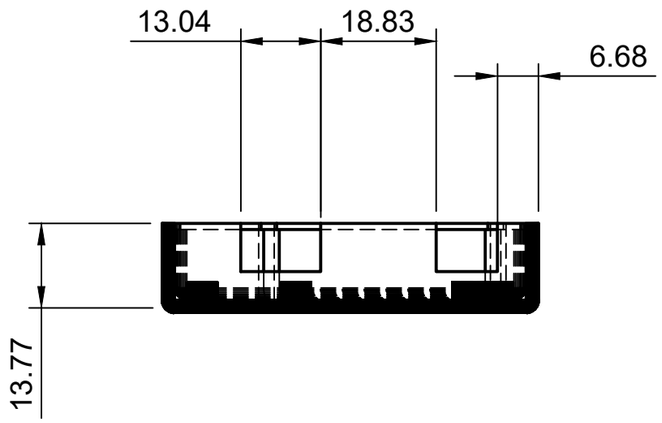
CARCASA ARDUINO MEGA 2560

PARTE INFERIOR DE LA CARCASA DEL ARDUINO MEGA



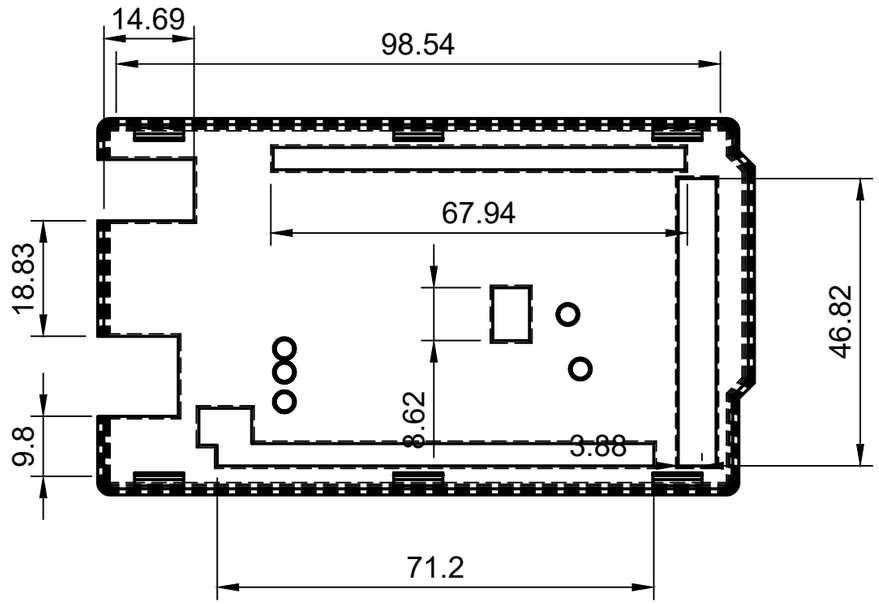


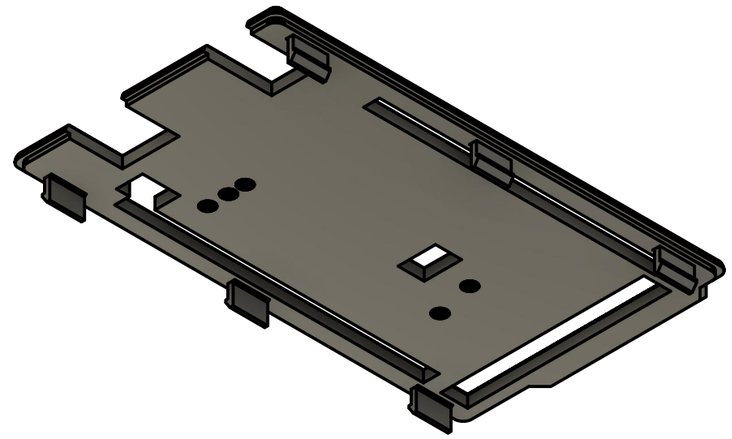


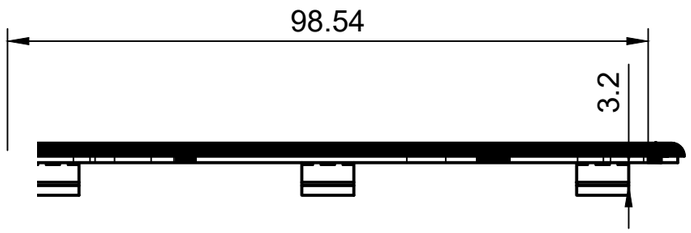


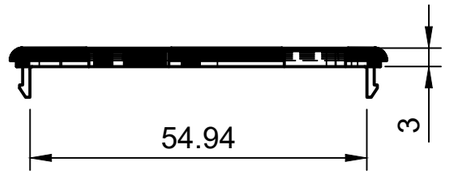
Dept.	Technical reference	Created by	Approved by
		Ciro Alfonso Pallares Fragozo 18/11/2020	
Medida establecida en milímetros		Document type	Document status
		Title: Arduino inferior	DWG No.
			Rev. / Date of issue / Sheet: 1/1

PARTE SUPERIOR DE LA CARCASA DEL ARDUINO MEGA



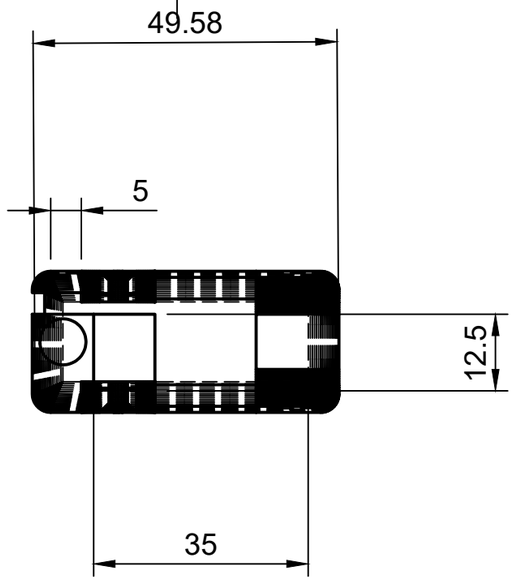


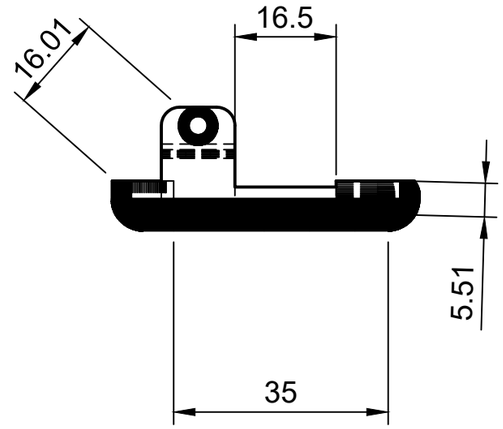


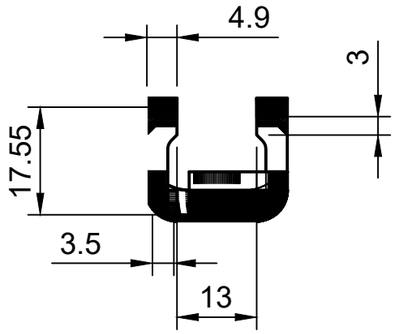


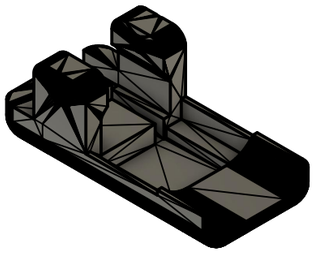
Dept.	Technical reference	Created by	Approved by
		Ciro Alfonso Pallares Fragozo 18/11/2020	
Medida establecida en milímetros		Document type	Document status
		Title: Arduino Superior	DWG No.
			Rev. / Date of issue / Sheet: 1/1

OXÍMETRO



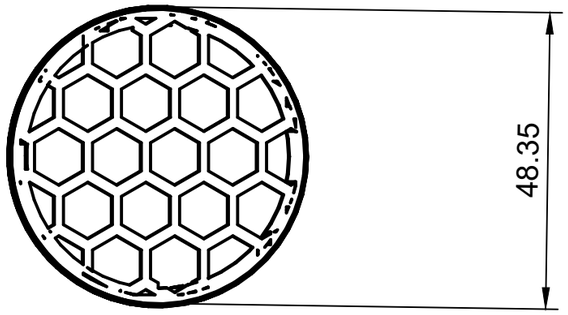


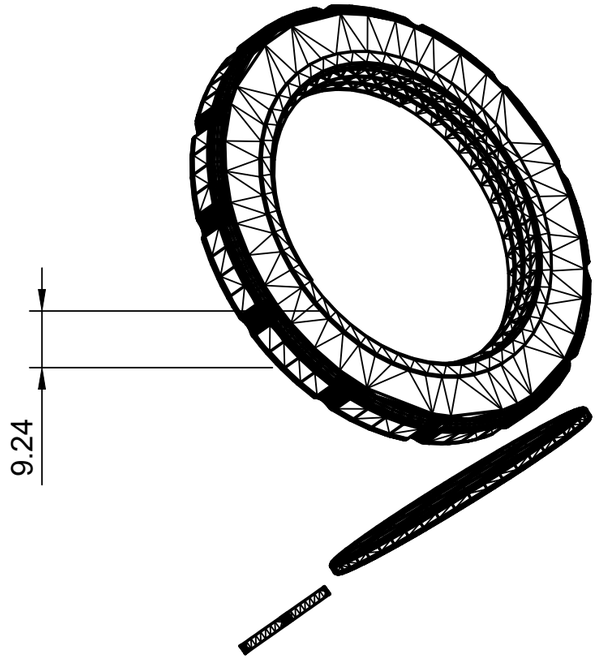


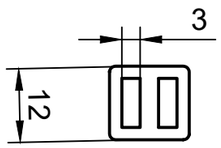


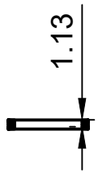
Dept.	Technical reference	Created by Ciro Alfonso Pallares Fragozo 18/11/2020	Approved by
Medida establecida en milímetros		Document type	Document status
		Title: Oxímetro	DWG No.
			Rev. / Date of issue / Sheet 1/1

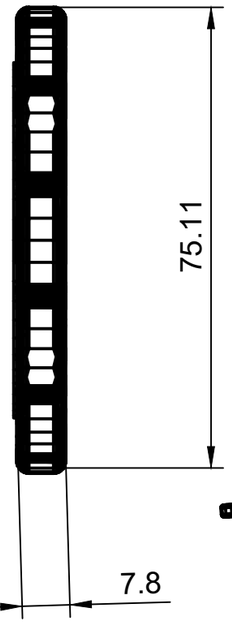
MÁSCARA (FILTRO, PIEZA PARA ASEGURAR LA MÁSCARA Y TUERCA SECUNDARIA)



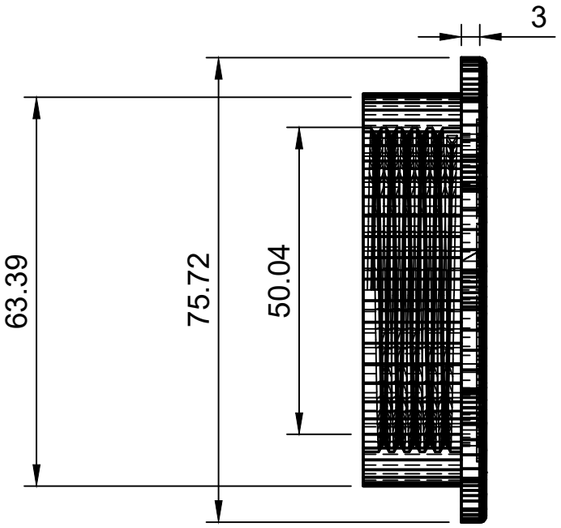


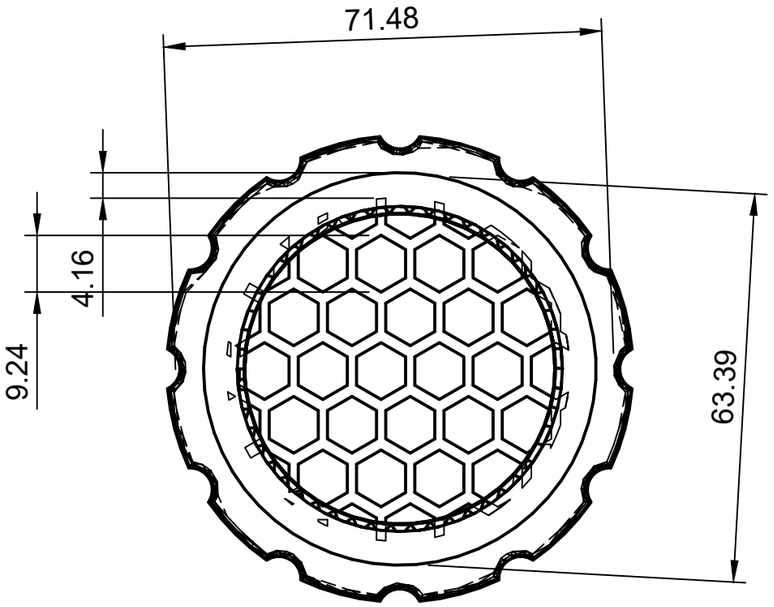


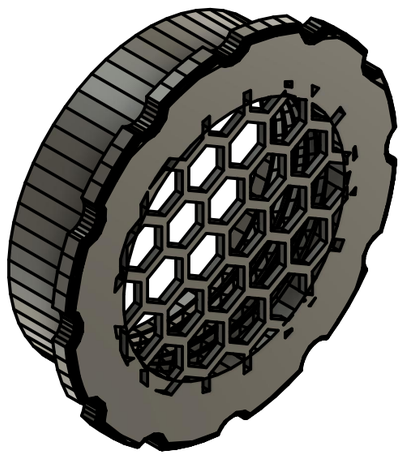




Dept.	Technical reference	Created by	Approved by
		Ciro Alfonso Pallares Fragozo 19/11/2020	
Medida establecida en milímetros		Document type	Document status
		Title: Filtro tuerca y fermo elastico	DWG No.
			Rev. / Date of issue / Sheet: 1/1

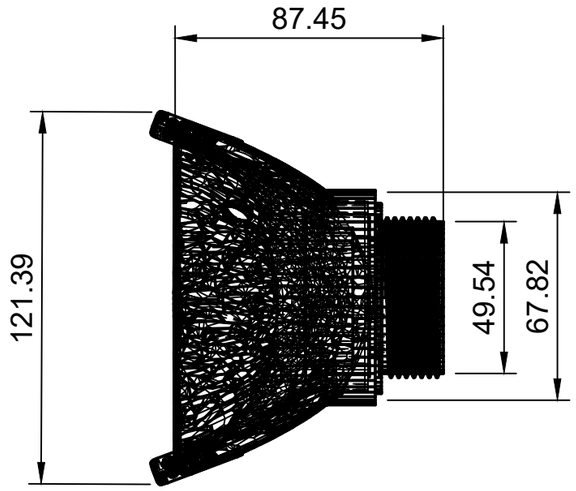


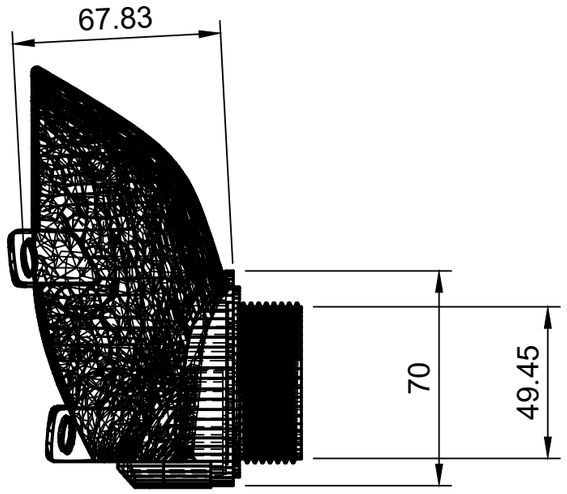


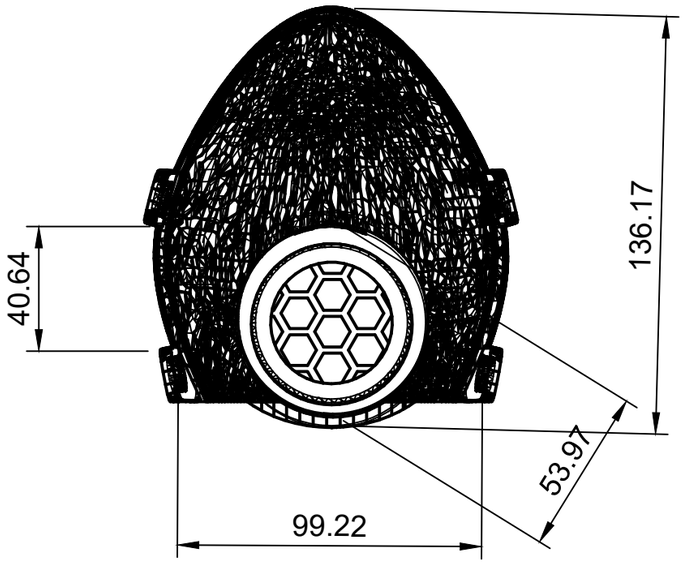


Dept.	Technical reference	Created by Ciro Alfonso Pallares Fragozo 19/11/2020	Approved by
Medida establecida en milímetros		Document type	Document status
		Title Tuerca filtro primario	DWG No.
			Rev. / Date of issue / Sheet 1/1

MÁSCARA (PIEZA PRINCIPAL)







Dept.	Technical reference	Created by Ciro Alfonso Pallares Fragozo 19/11/2020	Approved by
Medida establecida en milímetros		Document type	Document status
		Title: Máscara	DWG No.
			Rev. / Date of issue / Sheet 1/1