UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE MONITOREO DE INDICADORES DE SALUD MEDIANTE IOT PARA LA TOMA DE SIGNOS VITALES

AUTOR

Vela Mosquera, Juan Carlos

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en
MAGÍSTER EN ELECTRÓNICA Y AUTOMATIZACIÓN

TUTOR

Gómez Morales, Oscar Wladimir

Santa Elena, Ecuador

Año 2026

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA
Validar únicamente con FirmaEC

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
OSCAR WLADIMIR GOMEZ MORALES
Validar únicamente con FirmaEC

Ing. Oscar Gómez Morales, Ph.D.
TUTOR

Firmado electrónicamente por:
JUNIOR RAFAEL FIGUEROA OLMEDO
Validar únicamente con FirmaEC

Ing. Junior Figueroa Olmedo, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
LUIS ENRIQUE CHUQUIMARCA JIMENEZ
Validar únicamente con FirmaEC

Ing. Luis Chuquimarca Jiménez, Ph.D.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ
Validar únicamente con FirmaEC

Abg. María Rivera González, Mgtr.
SECRETARIA GENERAL
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por Vela Mosquera Juan Carlos, como requerimiento para la obtención del título de Magíster en Electrónica y Automatización.

TUTOR

Firmado electrónicamente por:
OSCAR WLADIMIR GOMEZ MORALES
Validar únicamente con FirmaEC

Ing. Oscar Gómez Morales, Ph.D.

Santa Elena, 2 de marzo de 2026

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Vela Mosquera Juan Carlos**

DECLARO QUE:

El trabajo de Titulación, Desarrollo de un prototipo de sistema de monitoreo de indicadores de salud mediante IOT para la toma de signos vitales previo a la obtención del título en Magíster en Electrónica y Automatización, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 2 de marzo de 2026

EL AUTOR

Firmado electrónicamente por:
JUAN CARLOS VELA MOSQUERA
Validar únicamente con FirmaEC

Juan Carlos Vela Mosquera

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado Desarrollo de un prototipo de sistema de monitoreo de indicadores de salud mediante IOT para la toma de signos vitales, presentado por el estudiante, Juan Carlos Vela Mosquera fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 9%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


CERTIFICADO DE ANÁLISIS
magister

Proyecto de Titulación-Juan Vela

9% Textos sospechosos

3% Similitudes (ignorado)
0 % similitudes entre comillas
< 1 % entre las fuentes mencionadas
0% Idiomas no reconocidos
9% Textos potencialmente generados por la IA

Nombre del documento: Proyecto de Titulación-Juan Vela.pdf
ID del documento: ef4d73be6671ecd6796158c9957afdf7d76a5de
Tamaño del documento original: 6,19 MB

Depositante: Oscar Wladimir Gómez Morales
Fecha de depósito: 2/3/2026
Tipo de carga: interface
fecha de fin de análisis: 2/3/2026

Número de palabras: 10.687
Número de caracteres: 88.489

TUTOR

Firmado electrónicamente por:
OSCAR WLADIMIR GOMEZ MORALES
Validar únicamente con FirmaEC

Ing. Oscar Gómez Morales, Ph.D.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **Vela Mosquera Juan Carlos**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de este proyecto de titulación con componentes de investigación aplicada y/o de desarrollo con fines de difusión pública, además apruebo la reproducción de este proyecto de titulación con componentes de investigación aplicada y/o de desarrollo dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor

Santa Elena, 2 de marzo de 2026

EL AUTOR

Firmado electrónicamente por:
JUAN CARLOS VELA MOSQUERA
Validar únicamente con FirmaEC

Juan Carlos Vela Mosquera

AGRADECIMIENTO

Agradezco en primer lugar a Dios, por permitirme estar presente en este momento, darme el conocimiento necesario y bendecirme cada día.

A mis Hijas Claris y mi bebe que en este momento que me encuentro realizando este trabajo se encuentra en la barriguita de mi bella esposa Mayra: ustedes me dan fuerzas y apoyo en cada paso que damos juntos como familia.

A las instituciones que han permitido mi desarrollo profesionalmente, un agradecimiento muy especial a la Universidad Estatal Península de Santa Elena (UPSE), y a todos sus magníficos docentes que impartieron sus grandes conocimientos en este proceso.

Y de forma muy particular, expreso mi profundo agradecimiento a mi tutor, el Ing. Oscar Wladimir Gómez Morales, por su guía imprescindible, su paciencia infinita y por compartir su vasto conocimiento, factores decisivos para el éxito técnico de este trabajo de titulación.

Juan Carlos, Vela Mosquera

DEDICATORIA

Dedico este logro y esfuerzo a Dios, por guiarme con su amor infinito y darme la fuerza para llegar hasta aquí.

A mi amada Mayra, a mis hijas que son mi bendición Claris y Madelaine: este logro es por y para ustedes, mi mayor motivación y mi hogar.

A mis padres, por su sacrificio, amor y ejemplo que me formaron.

A mis hermanos Marcelo, Gabriela y Leandro, así como mis primos Carlos y Andres, por su apoyo incondicional y su cariño constante.

Con todo mi corazón, este trabajo les pertenece.

Juan Carlos, Vela Mosquera

ÍNDICE GENERAL

TÍTULO ..... **I**

TRIBUNAL DE SUSTENTACIÓN ..... **II**

CERTIFICACIÓN ..... **III**

DECLARACIÓN DE RESPONSABILIDAD ..... **IV**

DECLARO QUE: ..... **IV**

CERTIFICACIÓN DE ANTIPLAGIO ..... **V**

AUTORIZACIÓN ..... **VI**

AGRADECIMIENTO ..... **VII**

DEDICATORIA ..... **VIII**

ÍNDICE GENERAL ..... **IX**

ÍNDICE DE TABLAS ..... **XI**

ÍNDICE DE FIGURAS ..... **XII**

RESUMEN ..... **XIII**

ABSTRACT ..... **XIV**

INTRODUCCIÓN **1**

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL **3**

1.1. Revisión de literatura 3

1.2. Desarrollo teórico y conceptual 6

1.2.1. Signos Vitales 6

1.2.2. Automatización 8

1.2.3. Sensores 8

1.2.4. Protocolo de comunicación 12

1.2.5. Plataforma de monitoreo 13

1.2.6. Normativas de uso médico 14

CAPÍTULO 2. METODOLOGÍA 15

2.1. Selección de Hardware 17

2.2. Selección de Software 19

2.3. Selección de Plataforma IoT 21

2.4. Diseño Electrónico 23

2.4.1. Simulación física del circuito Electrónico 23

2.4.2. Esquemático del circuito 24

2.5. Diseño y modelado CAD 26

2.6. Diagrama de Flujo de la programación del Sistema 27

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN 29

3.1. Construcción y Evaluación del prototipo 29

3.1.1. Conectividad VNC Viewer 29

3.1.2. Impresión en 3D 32

3.2. Análisis y Discusión de Resultados 41

CONCLUSIONES 43

RECOMENDACIONES 44

REFERENCIAS 45

ANEXOS 49

ÍNDICE DE TABLAS

Tabla 1 Comparaciones de trabajos similares.....3

Tabla 2 Parámetros de Signos vitales.....6

Tabla 3 Tipos de sensores en la Salud.....9

Tabla 4 Clasificación de protocolos de comunicación utilizados en sistemas IoT.....12

Tabla 5 Selección de Hardware.....17

Tabla 6 Selección de Hardware.....20

Tabla 7 Selección de Plataforma IoT.....22

Tabla 8 Diodos de protección electrónico.....25

Tabla 9 Tabulación de datos.....40

ÍNDICE DE FIGURAS

Figura 1 Metodología de investigación 15

Figura 2 *Diagrama de Flujo del prototipo* 16

Figura 3 Simulación de componentes electrónicos Físicos. 24

Figura 4 Esquemático del circuito electrónico 25

Figura 5 Diseño CAD del prototipo 26

Figura 6 Case del sensor PulseSensor 27

Figura 7 Diagrama de Flujo del programa 28

Figura 8 Interfaz VNC con Raspberry Pi 3B+ 29

Figura 9 Interfaz de la Raspberry por medio de VNC Viewer 30

Figura 10 Carga de archivo StandardFirmata 31

Figura 11 Monitor Serial de Arduino 32

Figura 12 Configuración de impresión en 3D 33

Figura 13 Prototipo Ensamblado 34

Figura 14 Prototipo en el antebrazo del Usuario 35

Figura 15 HMI del prototipo en funcionamiento 36

Figura 16 Usuario 2, observando sus signos vitales 37

Figura 17 Interfaz del prototipo. 38

Figura 18 Envío de datos a ThingSpeak 39

Figura 19 Gráfico de barra, de los datos de muestra. 41

RESUMEN

Este trabajo presenta el desarrollo de un prototipo de sistema de monitoreo de indicadores de salud que estuviesen basados en tecnologías de internet de las cosas, orientado a una medición de forma remota y continua de signos vitales. El estudio estaba fundamentado en un enfoque cuantitativo descriptivo apoyado por revisiones bibliográficas y un diseño experimental. El prototipo integró sensores biomédicos no invasivos para poder medir la frecuencia cardíaca, temperatura corporal y ambiental y otros apartados empleando una arquitectura maestro-esclavo basada en Arduino y Raspberry Pi. los datos capturados se los procesó localmente y se los transmitió a la nube por medio de la plataforma Thingspeak, permitiendo una visualización en tiempo real por medio de la interfaz gráfica desarrollada en python y tkinter. Los resultados evidenciaron Comunicaciones estables, además de una baja latencia y una adecuada confiabilidad en las mediciones que se realizaron en las pruebas

Palabras claves: Arduino, internet de las cosas, Monitoreo de salud, Raspberry Pi, signos vitales.

ABSTRACT

This work presents the development of a prototype health indicator monitoring system based on Internet of Things (IoT) technologies, aimed at the remote and continuous measurement of vital signs. The study was grounded in a descriptive quantitative approach supported by literature review and an experimental design. The prototype integrated non-invasive biomedical sensors to measure heart rate, body temperature, and ambient temperature, among other parameters, using a master–slave architecture based on Arduino and Raspberry Pi. The captured data were processed locally and transmitted to the cloud through the ThingSpeak platform, enabling real-time visualization via a graphical interface developed in Python and Tkinter. The results demonstrated stable communications, low latency, and adequate reliability in the measurements obtained during testing.

Keywords: Arduino, health monitoring, Internet of Things (IoT), Raspberry Pi, vital signs

INTRODUCCIÓN

La integración de sensores corporales con plataformas de Internet de las Cosas (IoT) ha crecido aceleradamente a nivel global como una solución para la monitorización remota continua, con el propósito de detectar o predecir el deterioro de la salud (Kristoffersson A, 2020). En el contexto mundial, el monitoreo de forma remota a diferentes pacientes ha podido consolidarse como una solución viable dentro de la atención sanitaria actual, considerando que permite una recolección y transmisión de signos vitales como la frecuencia cardíaca, temperatura o el ritmo respiratorio, sin la necesidad de que la persona se encuentre físicamente en el hospital utilizando Redes Inalámbricas de Área Corporal (WBANs) (Akkaş y otros, 2020). Estos sistemas permiten que los datos sean visualizados por personal médico y de salud y familiares mediante aplicaciones web o app móviles, facilitando decisiones basadas en datos corporales longitudinales, las cuales pueden ser analizadas mediante Inteligencia Artificial (Andrade y otros, 2024).

A nivel de América Latina y la región andina, la implementación de estas tecnologías enfrenta desafíos únicos relacionados con la infraestructura y la geografía. En estas regiones, la monitorización remota no solo busca minimizar costos sanitarios (Taherdoost, 2024), sino que se convierte en una necesidad crítica para poblaciones vulnerables en zonas de difícil acceso (Mohammed y otros, 2020).

En el caso específico de Ecuador, la situación epidemiológica revela una alta incidencia de enfermedades cardiovasculares, las cuales representan una de las principales causas de mortalidad (MSP, 2020). También este tipo de enfoques IoT permite integrar mecanismos con alertas tempranas para algún tipo de anomalías, lo que resulta de gran importancia sobre todo para el monitoreo de enfermedades crónicas. Por tanto, los desafíos técnicos incluyen un control y aseguramiento de la interoperabilidad entre dispositivos para garantizar una transmisión de datos (Abdulmalek y otros, 2022)

La implementación en sistemas de salud de tecnología IoMT, introduce también tipos de arquitecturas de comunicación que son especializadas, esto para que puedan soportar dispositivos inteligentes que puedan permitir una adecuada transmisión de datos biométricos por redes inalámbricas (Chen & Sheng, 2024).

Este prototipo involucra la integración de varios sensores, los cuales nos permite tener el registro de las variables como, Temperatura corporal la temperatura de medio ambiente, así como la señal del ritmo cardiaco BPM, para la adquisición de esta señal, se usan dos

sensores, que a pesar que adquieren la misma señal, son dos dispositivos totalmente diferentes a nivel de hardware, el cual se describirá en páginas posteriores,

El prototipo cuenta con una interfaz gráfica la cual fue desarrollado en el lenguaje de Python y su librería por defecto TKinter (Python, 2025), la cual muestra una interfaz HMI con los datos del paciente, para que el medico pueda observar las variaciones de estas variables y realice su análisis determinado, con la ayuda de las plataformas de IoT como Thingspeak, estas variables pueden ser observadas a nivel global por otros especialistas y tener un mejor análisis.

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

A continuación, se presenta el desarrollo del análisis bibliográfico, analizado para este trabajo, el cual nos permite tener una clara idea de investigaciones similares a la presentada en este escrito. La cual se extrajo de revistas científicas confiables y bases de datos de alto impacto.

1.1. Revisión de literatura

Se realizó una exhaustiva búsqueda bibliográfica sobre temas relacionados con la investigación en bases de datos reconocidas como Scopus, SciELO y National Library of Medicine, entre otras, con el objetivo de analizar los avances más significativos y actualizados al momento del desarrollo del proyecto. Como resultado de esta revisión, en la Tabla 1 se presenta una comparación de diez trabajos similares considerados los más relevantes para el estudio. No obstante, fue necesario examinar un número considerablemente mayor de investigaciones para llevar a cabo un análisis adecuado y fundamentado.

Tabla 1

Comparaciones de trabajos similares

N	Tema de investigación	Tecnologías IoT (Sensores/Microcontroladores)	Plataforma de Comunicación/Visualización	Aplicación Principal	Puntos Clave / Innovación
1	Sistema de Monitoreo de Salud Basado en IoT para Mejorar la Calidad de Vida (Abdulmalek S, 2022)	Sensores varios, M-Health/E-Health	Smartphones, Internet	Cuidado crítico, pandemias	Uso de smartphones para monitorización de salud en contextos de pandemia.
2	Dispositivos portátiles IoT y de monitorización de la salud como tecnologías habilitadoras para la mejora sostenible de la calidad de vida en entornos inteligentes (Zovko, 2023)	Sensores no invasivos, wearables	Redes observadoras de monitoreo de salud	Mejora de la calidad de vida en entornos inteligentes	Solución de monitoreo continuo no invasivo con wearables.

3	Uso de dispositivos sensores IoT para la gestión eficiente de sistemas sanitarios (Gopichand, 2024)	Dispositivos digitales inalámbricos interconectados	Red IoT	Anticipación, diagnóstico, tratamiento y monitoreo de pacientes	Red de dispositivos interconectados para recopilación y transmisión automática de datos.
4	MLRA-Sec: un modelo adaptativo e inteligente de evaluación de ciberseguridad para el Internet de las Cosas Médicas (IoMT) (Ksibi, 2024)	Dispositivos médicos conectados (CMD)	IoMT (Internet de las Cosas Médicas)	Monitoreo de progresión de enfermedades, seguimiento de pacientes	Énfasis en la seguridad y privacidad en IoMT.
5	Desarrollo prototipo de monitorización remota continua de pacientes en UCI en el domicilio (Thippeswamy, 2021)	AD8232, MAX30102, NTC	Monitoreo remoto en el hogar, Google Firebase	Pacientes de UCI en el hogar	Monitoreo continuo de pacientes de UCI en la comodidad de su hogar.
6	Sistema inteligente de monitorización de la salud para la medición en tiempo real de signos vitales (Afifi, 2025)	Sensores inteligentes, gadgets	Herramientas de análisis (IA)	Detección temprana de deterioro del paciente	Sistema rentable para monitoreo continuo con IA para decisiones clínicas.
7	Servicios médicos de salud con superinteligencia artificial y sistema inteligente vestible basado en IoT para monitorización remota de la salud (Elango, 2023)	Sensores, wearables (Super IA)	IoT	Monitoreo remoto de salud	Uso de IA y wearables para monitorear múltiples indicadores y riesgos como caídas.
8	Monitorización en tiempo real de signos vitales y gestión de datos utilizando un sistema de monitorización de salud basado en IoT de bajo coste (Mishra A. D., 2024)	Sensores (BP, temperatura sin contacto, SpO2, ECG), OLED, ESP8266	IoT de bajo costo, pantalla OLED	Monitoreo continuo de signos vitales	Sistema de bajo costo y tiempo real para monitoreo de signos vitales.

9	Desarrollo de un sistema de monitorización de pacientes asmáticos con IoT y rentable: integrando datos de salud y entorno interior con análisis estadístico y visualización de datos (Chakraborty, 2023)	MAX30100, DS18B20, DHT11, NodeMCU, Arduino	IoT, retroalimentación en tiempo real	Pacientes con asma	Monitoreo de indicadores de salud y parámetros ambientales para pacientes con asma.
10	Sistema impulsado por IoT para el seguimiento continuo de pacientes con enfermedades cardíacas tras la cirugía (Harez, 2024)	HW-827, DS18B20, ESP32	Plataformas conectadas a la nube	Pacientes post-cirugía cardíaca	Monitoreo continuo post-quirúrgico con sensores avanzados y conectividad inalámbrica.

Nota. Se comparó con 10 investigaciones similares al proyecto en desarrollo. Elaboración propia

En la tabla se puede observar que muchas de las investigaciones han sido desarrolladas en los últimos años, principalmente debido a la evolución de la tecnología IoT. Esta evolución ha permitido que numerosos proyectos en el área de la salud enfoquen sus aplicaciones hacia el Internet of Medical Things (IoMT), incorporando redes WBANs para el monitoreo de pacientes.

En los últimos años, este tipo de soluciones ha tenido un crecimiento significativo. Además, a raíz de la pandemia ocurrida en 2020, los problemas cardíacos y otras complicaciones de salud han evidenciado un incremento considerable, lo que ha generado mayor interés en el desarrollo de soluciones tecnológicas que permitan prevenir, monitorear o incluso pronosticar posibles afecciones futuras. Por esta razón, se ha puesto un énfasis importante en la implementación de sistemas de monitoreo remoto como alternativa viable dentro del sector salud.

La expansión de la telemedicina a nivel mundial, desde el año 2020 hasta la actualidad, se ha desarrollado investigaciones relacionadas a este tema, en la página de PubMed, el cual es una biblioteca Nacional de Medicina de los Institutos Nacionales de Salud de EE.

UU. (PubMed Central PMC, 2025), ha tenido un amplio crecimiento en investigaciones relacionas en este tema.

1.2. Desarrollo teórico y conceptual

En esta sección del capítulo, se revisa una variedad de fuentes bibliográficas, se presentan los conceptos teóricos generales de varios parámetros importantes dentro del prototipo propuestos que sustenta su desarrollo.

1.2.1. Signos Vitales

Los signos vitales constituyen un conjunto de parámetros fisiológicos que permiten evaluar el estado general de salud de una persona y detectar posibles alteraciones en el funcionamiento normal del organismo. (Hall, 2021)

Estos también muestran funciones esenciales del cuerpo, como el ritmo cardiaco, la frecuencia respiratoria, la temperatura y la presión arterial, estos parámetros cambian con respecto a la edad, el sexo, el peso, la capacidad para ejercitarse de la persona (MedlinePlus, 2025).

En la tabla 2, se muestra una breve descripción y los rangos normales de los signos vitales para un adulto sano promedio mientras está en reposo, así como los métodos tradicionales de medición:

Tabla 2

Parámetros de Signos vitales

Parámetro	Descripción Fisiológica	Rango Normal en Adultos	Métodos de Medición
Frecuencia Cardíaca (Pulso)	Número de veces que el corazón late por minuto. Refleja la actividad del sistema cardiovascular y la demanda de oxígeno del cuerpo. Un aumento puede indicar fiebre, ansiedad, ejercicio o insuficiencia cardíaca, mientras que una disminución puede ser signo de buena condición física o de problemas de conducción eléctrica del corazón.	60 - 100 latidos por minuto (lpm)	Manual: Palpación de arterias periféricas (radial, carótida).

Temperatura Corporal	Medida del calor interno del cuerpo, regulada por el hipotálamo. Es un indicador clave de la presencia de infección, inflamación o trastornos del sistema termorregulador. La fiebre es una respuesta inmunitaria común a patógenos.	36.5°C - 37.5°C (oral)	Manual/Tradicional: Termómetros de mercurio o digitales (bucal, axilar, rectal).
Frecuencia Respiratoria	Número de ciclos respiratorios (inhalación y exhalación) por minuto. Refleja la eficiencia del sistema respiratorio y el equilibrio ácido-base del cuerpo. Puede aumentar en casos de ansiedad, fiebre, neumonía o insuficiencia cardíaca, y disminuir por depresión del sistema nervioso central.	12 - 20 respiraciones por minuto (rpm)	Manual: Observación visual del movimiento del tórax durante un minuto.
Presión Arterial	Fuerza que ejerce la sangre contra las paredes de las arterias. Se expresa como dos valores: presión sistólica (presión máxima durante la contracción del corazón) y presión diastólica (presión mínima entre latidos). Es un indicador crítico del riesgo cardiovascular; la hipertensión es un factor de riesgo mayor para infartos y accidentes cerebrovasculares.	Sistólica: < 120 mmHg Diastólica: < 80 mmHg	Manual: Esfigmomanómetro de mercurio o aneroide con estetoscopio (método de auscultación de Korotkoff).
Saturación de Oxígeno en Sangre (SpO_2)	Porcentaje de hemoglobina en la sangre periférica que está saturada con oxígeno. Es un indicador directo de la eficacia de la oxigenación pulmonar y la perfusión tisular. Valores bajos (hipoxemia) pueden ser signo de enfermedades pulmonares, insuficiencia cardíaca o altitud extrema.	95% - 100%	Manual: Gasometría arterial (invasiva y no apta para monitoreo continuo).

Nota. Estos rangos varían dependiendo de las condiciones de la persona. Tomado de *Biblioteca Nacional de Medicina*, 2025, MedlinePlus (https://medlineplus.gov/spanish/ency/article/002341.htm)

Los signos vitales permiten obtener una visión rápida y no invasiva del estado funcional del cuerpo humano. Considerando que el organismo es un sistema complejo, se puede realizar una analogía con la conducción de un vehículo. En un automóvil, el tablero

muestra información fundamental sobre el funcionamiento de la máquina, como la temperatura del motor, la velocidad del vehículo o el nivel de aceite, parámetros que permiten al conductor conocer si el sistema está operando dentro de condiciones normales. Si uno o varios de estos valores se encuentran fuera de su rango adecuado, es necesario detener el vehículo y tomar las acciones correspondientes para evitar daños en el motor. (Vela Mosquera, 2024)

De manera similar, en el cuerpo humano los signos vitales cumplen la función de un "tablero de control", ya que permiten observar en un momento determinado el estado general del organismo. Estos parámetros proporcionan información clave sobre el funcionamiento de los sistemas fisiológicos, por lo que representan uno de los primeros indicadores que los profesionales de la salud evalúan al momento de realizar un diagnóstico o valorar la condición de un paciente. (Hall, 2021)

1.2.2. Automatización

La automatización en términos generales implica la creación de un ecosistema tecnológico autónomo donde múltiples componentes trabajan en conjunto y sincronizadamente para la adquisición de datos sin la intervención humana (Manzano Ramos, 2021).

Así la automatización culmina en la gestión y visualización de datos, estos datos se obtienen de forma longitudinal y son enviados a plataformas de monitoreo que pueden ser local, web, móvil o IoT, donde se almacena de forma segura, se analiza en tiempo real, y sus datos pueden ser interpretados por el tiempo y obtener un análisis cuantitativo evolutivo del usuario (Yogi, 2024).

Este tipo de modelos de atención remota puede ser considerado muy útil en poblaciones que sean vulnerables, por lo que este tipo de dispositivos IoT logra representar una tendencia que va creciendo en el monitoreo continuo y personalizado para análisis en la salud (Mohammed y otros, 2020).

1.2.3. Sensores

Según (Vijayalakshmi Subramanian, 2023), los sensores médicos se definen como dispositivos que integran la tecnología de la información electrónica con la biomedicina, facilitando los avances en la atención sanitaria mediante procedimientos diagnósticos inteligentes, micro, multiparámetros, de control remoto y no invasivos.

La monitorización de los signos vitales es fundamental en el análisis clínico, y estos están estrechamente relacionados a los sensores que se encuentran dentro del sistema del paciente para brindar una correcta recopilación de información y esta pueda ser visualizada por el profesional de la salud, ofreciendo una ventana directa a la detección temprana de alteraciones que podrían indicar el inicio de una patología o el deterioro de una condición existente (Vela Mosquera, 2024).

Los sensores médicos se clasifican principalmente si son invasivos o no invasivos y su principio de operación, en la tabla 3, se realiza un resumen de los diferentes tipos de sensores en el ámbito de la salud, esta distinción se basa en el grado de interacción física con el cuerpo humano, lo que influye directamente en la precisión de la medición.

Tabla 3

Tipos de sensores en la Salud.

Característica	Sensores No Invasivos	Sensores Invasivos
Definición	Dispositivos que miden parámetros fisiológicos sin penetrar la piel ni introducirse en cavidades corporales. Operan desde el exterior del cuerpo.	Dispositivos que requieren penetrar la piel, insertarse en vasos sanguíneos, órganos o tejidos internos para realizar la medición.
Nivel de Riesgo	Muy bajo. No presentan riesgos de infección, hemorragia o daño tisular. Ideales para uso prolongado y en entornos no clínicos.	Moderado a alto. Conllevan riesgos inherentes como infección, inflamación, coagulación, embolia o trauma tisular. Su uso está restringido a entornos controlados.
Confort del Paciente	Alto. Son generalmente cómodos, discretos y permiten la movilidad del paciente. Esencial para la adherencia en monitoreo a largo plazo.	Bajo. Pueden causar dolor, incomodidad e inmovilidad. Su tolerancia es limitada en el tiempo.

Precisión y Fiabilidad	Variable. Pueden ser susceptibles a interferencias externas (movimiento, luz ambiental, temperatura). La precisión ha mejorado significativamente con avances en procesamiento de señal.	Generalmente alta. Al medir directamente en el sitio de interés (p. ej., sangre arterial), ofrecen datos más directos y menos propensos a artefactos.
Costo y Mantenimiento	Bajo. Son dispositivos de bajo costo, a menudo desechables o de fácil limpieza. Ideales para implementaciones a gran escala.	Alto. Requieren procedimientos estériles, personal capacitado para su inserción y monitoreo constante, lo que incrementa los costos operativos.
Aplicaciones Típicas	Monitoreo ambulatorio, telemedicina, atención primaria, entornos educativos, prevención y bienestar. Sistemas IoT para el hogar.	Unidades de Cuidados Intensivos (UCI), cirugía, diagnóstico clínico especializado, monitoreo hemodinámico continuo, manejo crítico de pacientes.
Ejemplos Específicos	-Fotopletismografía (PPG): Sensor MAX30100/MAX30102 para medir frecuencia cardíaca y saturación de oxígeno (SpO_2) en el dedo o lóbulo de la oreja. -Termómetro digital: Sensor DS18B20 para medir temperatura corporal en axila o frente. -Electrocardiograma (ECG) de superficie: Electrodos adhesivos en el tórax para registrar la actividad eléctrica del corazón. -Esfigmomanómetro	- Catéter arterial: Con sensor de presión integrado para medir presión arterial invasiva (IBP) en tiempo real. - Sensor de glucosa subcutáneo: Electrodo implantado en el tejido intersticial para monitoreo continuo de glucosa. - Electrodos intracardiacos: Utilizados en estudios electrofisiológicos para mapear arritmias. - Gasometría arterial: Análisis

	oscilométrico: Manguito en el brazo para medir presión arterial de forma indirecta.	de una muestra de sangre extraída directamente de una arteria para medir gases sanguíneos (PaO_2, $PaCO_2$) y pH.
Relevancia en Sistemas IoT	Son la base de la mayoría de los prototipos y sistemas comerciales de salud conectada. Permiten la creación de ecosistemas de monitoreo remoto, continuo y de bajo costo	Su integración en sistemas IoT es compleja y limitada a contextos hospitalarios de alta especialización. La transmisión de datos de estos sensores ya es una práctica común en las UCI, pero su expansión a entornos no clínicos es poco probable debido a sus riesgos.

Nota. Adaptado de *Smart medical sensor*, 2023, Security and Privacy Issues in Internet of Medical Things (https://www.sciencedirect.com/science/article/pii/B9780323898720000058)

De acuerdo con (God y otros, 2024), los sensores versátiles permiten ofrecer un medio que sea fiable para poder supervisar la salud en pacientes, permitiendo detectar posibles deterioros en la salud antes de que se agraven. Al integrar redes IoT con diferentes sensores, se permite el envío de grandes volúmenes de datos a la nube para ser analizados de forma automática y almacenados adecuadamente.

La combinación de diferentes sensores corporales, microcontroladores y protocolos IoT permite facilitar modelos de atención que estén más centrados en el paciente y en enfermedades específicas (Khan & Duncan, 2025). La utilización de diferentes redes como Bluetooth o Wi-Fi que se encuentran integradas con este tipo de plataformas en la nube permite favorecer una escalabilidad entre diferentes dispositivos médicos y aplicaciones para un monitoreo (Abu-Jassar y otros, 2025).

Los sistemas de sensores embebidos se han convertido en el elemento clave de los avances en dispositivos médicos; La alta versatilidad que ofrecen permite el desarrollo de

nuevos dispositivos diagnósticos y avanzados de monitorización para pacientes tanto en entornos domésticos como hospitalarios (Arandia N, 2022).

1.2.4. Protocolo de comunicación

Los protocolos de comunicación involucran un conjunto de reglas y estándares que permiten el intercambio de información entre dispositivos dentro de una red. Estos protocolos establecen la forma en que los datos son estructurados y encapsulados, transmitidos y recibidos entre los diferentes componentes que tiene interconectado el sistema (Valvano, 2017).

En la tabla 4, se resume los diferentes protocolos que se usa en las diferentes etapas del prototipo, basándonos en una exhaustiva fuente bibliográfica, tanto desde los dispositivos de adquisición de datos que se encuentran entre los sensores y el paciente, así como los diferentes protocolos entre el maestro y las diferentes plataformas IoT.

Tabla 4

Clasificación de protocolos de comunicación utilizados en sistemas IoT

Nivel del sistema	Protocolo / Tecnología	Descripción	Aplicación en sistemas IoT
Dispositivos (Hardware)	Serial (UART/USB)	Protocolo de comunicación utilizado para transmitir datos de forma secuencial entre dispositivos electrónicos.	Comunicación entre microcontroladores y computadoras o sistemas embebidos.
Dispositivos (Hardware)	Firmata	Protocolo que permite controlar un microcontrolador desde un software externo mediante comunicación serial.	Permite que plataformas como Python controlen y reciban datos desde Arduino.
Dispositivos (Hardware)	I2C	Protocolo de comunicación síncrona utilizado para conectar múltiples sensores y dispositivos en un mismo bus de comunicación.	Comunicación entre sensores y microcontroladores.

Dispositivos (Hardware)	SPI	Protocolo de comunicación de alta velocidad utilizado para conectar sensores, memorias y periféricos a microcontroladores.	Transferencia rápida de datos entre dispositivos electrónicos.
Red / IoT	Wi-Fi (TCP/IP)	Tecnología de comunicación inalámbrica que permite conectar dispositivos a redes locales e Internet.	Conexión de dispositivos IoT a servidores o plataformas en la nube.
Red / IoT	HTTP	Protocolo de transferencia utilizado para enviar información entre dispositivos y servidores web.	Envío de datos desde dispositivos IoT hacia plataformas de monitoreo en la nube.
Red / IoT	MQTT	Protocolo ligero basado en publicación-suscripción diseñado para aplicaciones IoT.	Transmisión eficiente de datos en redes con bajo consumo de ancho de banda.
Red / IoT	CoAP	Protocolo optimizado para dispositivos IoT con recursos limitados y redes de baja potencia.	Comunicación entre sensores y servidores en entornos IoT.

Nota. Adaptado de Bahga, A., & Madisetti, *Internet of things*, 2015, A hands-on approach. Universities Press.

Los protocolos de comunicación dentro de un sistema IoT pueden clasificarse en dos niveles principales: protocolos de comunicación a nivel de dispositivos, utilizados para el intercambio de información entre sensores y microcontroladores, y protocolos de comunicación a nivel de red o IoT, los cuales permiten la transmisión de datos hacia plataformas en la nube o sistemas de monitoreo remoto (Al-Fuqaha, 2015).

1.2.5. Plataforma de monitoreo

Las plataformas de monitoreo son sistemas informáticos que permiten almacenar, visualizar y analizar los datos generados por dispositivos conectados. En el contexto del Internet de las Cosas aplicado a la salud, estas plataformas permiten centralizar la información proveniente de sensores biomédicos y presentarla a través de interfaces gráficas accesibles para médicos o personal sanitario (Gubbi, 2013).

Entre las funciones principales de estas plataformas se encuentran el almacenamiento de datos históricos, la visualización en tiempo real, el análisis de tendencias y la generación de alertas ante posibles anomalías en los parámetros monitoreados (Gubbi, 2013).

1.2.6. Normativas de uso médico

El desarrollo e implementación de dispositivos utilizados para monitoreo de salud debe cumplir con diferentes normativas y estándares internacionales que garantizan la seguridad, confiabilidad y calidad de los equipos médicos. Estas regulaciones establecen requisitos relacionados con el diseño, la fabricación, la validación clínica y la protección de los datos de los pacientes.

La norma ISO 13485 es reconocida internacionalmente para los sistemas de gestión de calidad en el diseño y fabricación de dispositivos médicos. Establece requisitos específicos que ayudan a las organizaciones a garantizar que sus dispositivos médicos cumplan tanto con las exigencias de seguridad y eficacia como de los clientes y las normativas (ISO, 2025).

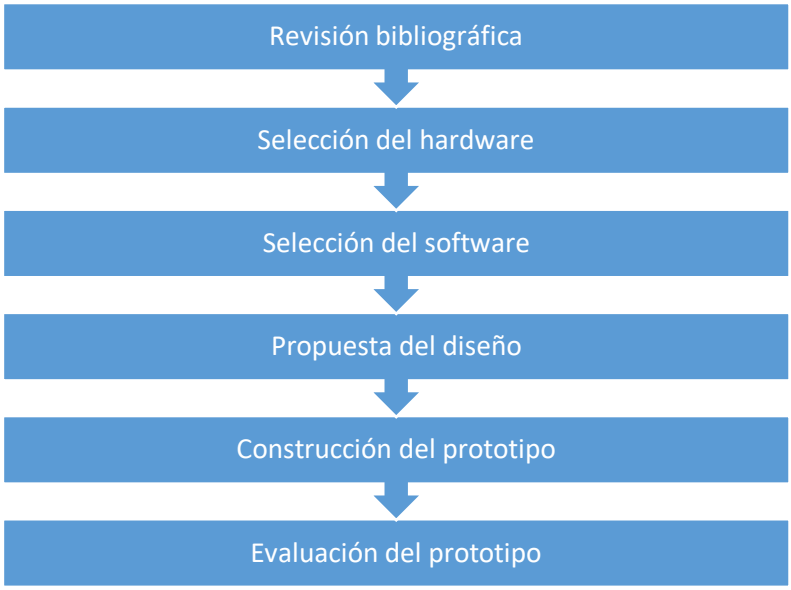
Organismos internacionales como la World Health Organization y la International Organization for Standardization han desarrollado normas que regulan el uso de dispositivos médicos y sistemas de monitoreo biomédico. Entre estas normas se encuentran estándares relacionados con seguridad eléctrica, compatibilidad electromagnética, gestión de calidad y protección de datos de salud (World Health Organization, 2025).

CAPÍTULO 2. METODOLOGÍA

La metodología aplicada para esta investigación se basa en un modelo cuantitativo descriptivo y consta de 7 etapas, como se indica en la Figura 1.

Figura 1

Metodología de investigación



Nota. Elaboración Propia

Para la primera parte del proceso investigativo del monitoreo de signos vitales a través de IoT mediante un prototipo, se realizó una revisión bibliográfica de estudios similares en revistas científicas almacenada en bases de datos como Scopus, ScienceDirect, Web of Science, Springer, entre otras. Seguidamente, se elige la mejor opción de tecnología a aplicar en el desarrollo del proyecto, además de la selección del hardware y del software, los cuales se utilizarán para la construcción del prototipo. Para las últimas etapas del proceso se diseñó un sistema de monitoreo de signos vitales utilizando IoT mediante la plataforma ThingSpeak juntamente con un diseño en Python utilizando la interfaz gráfica y la librería TkInter, además de utilizar varios sensores para el reconocimiento de variables.

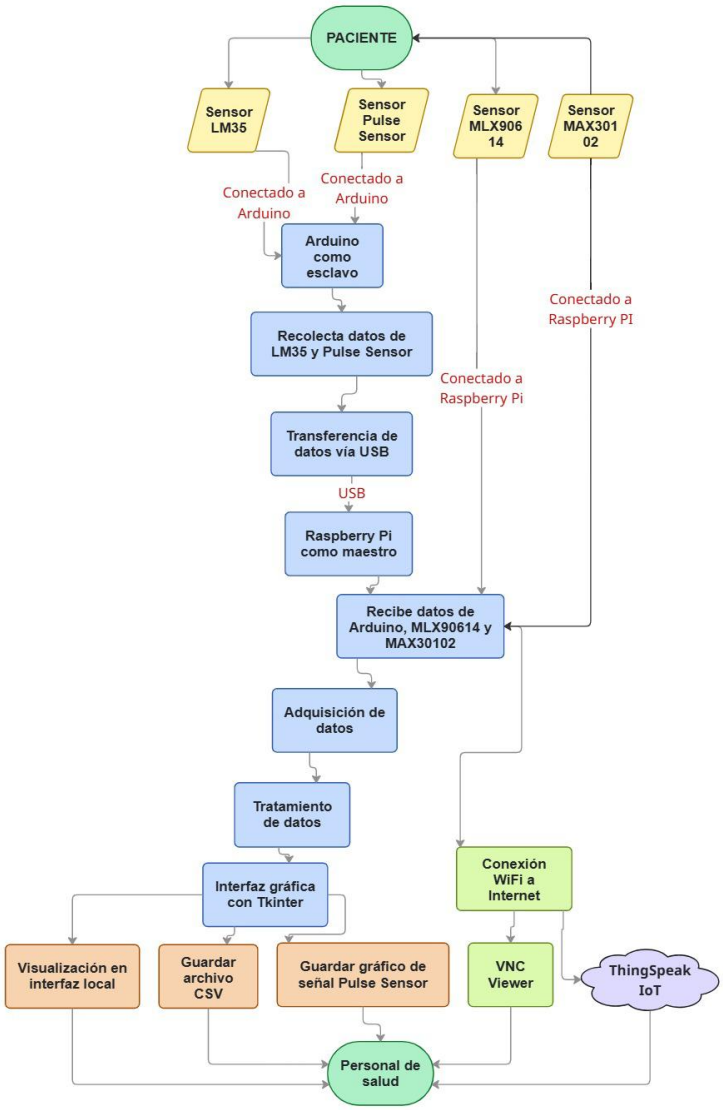
El sistema propuesto adopta una arquitectura distribuida en capas, como se ilustra en el diagrama de bloque de la Figura 2. Se divide en cinco módulos funcionales interconectados entre sí:

1. Adquisición de señales biomédicas
2. Comunicación y transferencia de datos
3. Procesamiento y análisis
4. Visualización y almacenamiento
5. Interfaz remota y nube

Esta separación permite modularidad, escalabilidad y mantenibilidad del sistema.

Figura 2

Diagrama de Flujo del prototipo



Nota. Secuencia de adquisición y procesamiento de datos de los sensores. Elaboración propia

2.1. Selección de Hardware

En la Tabla 5. Se puede observar los componentes seleccionados para este proyecto, mediante un cuadro comparativo con las diferentes opciones que tenemos en el mercado ecuatoriano.

En el existen diversas tarjetas electrónicas que cumplen con los requisitos del proyecto; sin embargo, se optó por una tarjeta electrónica Raspberry Pi 3B+ más un microcontrolador Arduino modelo Arduino Uno, los cuales funcionan de forma conjunta estableciendo una comunicación maestro-esclavo entre ellas. Este método de funcionamiento es funcional y acorde para este estudio, ya que el microcontrolador Arduino Uno cuenta con varias salidas digitales y analógicas que permiten leer la información de todos los sensores a utilizar. Además, se empleó un sensor LM35, el cual mide la temperatura, un sensor MLX90614, que es un sensor infrarrojo, y un sensor de pulso (pulse sensor), que es un sensor óptico por fotopletismografía (PPG).

Tabla 5

Selección de Hardware

Plataforma / Sensor	**Ventajas**	**Desventajas**	**Costo aprox. (USD)**	**Aplicabilidad en este proyecto**
Raspberry Pi 3B+	* Alto poder de procesamiento * Soporte nativo para Python, Tkinter, Matplotlib * Conectividad Wi-Fi/Ethernet * Puertos I²C, UART, GPIO, USB	* No tiene ADC analógico * Requiere módulo externo para señales analógicas * Mayor consumo de energía que microcontroladores	$35–40	Ideal como maestro: Procesa datos, genera interfaz gráfica y envía a la nube
Arduino Uno	* 6 entradas analógicas (ADC 10-bit)	* Bajo poder de procesamiento	$18–22	Ideal como esclavo: Adquisición confiable de

	* Fácil conexión con sensores analógicos (LM35, PulseSensor) * Compatibilidad con bibliotecas estándar (Firmata)	* Sin sistema operativo ni capacidad gráfica * Limitado almacenamiento y conectividad		señales biomédicas
ESP32	* Incluye Wi-Fi/Bluetooth * Dos núcleos, bajo costo * Toma de muestras rápida	* Menor precisión en ADC (ruido interno) * Difícil calibración del sensor PPG * Estabilidad limitada en lecturas analógicas continuas	$7–10	No recomendado: Señal PPG inestable; no ideal para ECG/PPG sin filtros externos
NodeMCU (ESP8266)	* Muy económico * Wi-Fi integrado * Fácil acceso a IoT	* Solo 1 entrada analógica * Alta variabilidad en voltaje de referencia * No soporta múltiples sensores analógicos	$4–6	Inadecuado: No puede leer LM35 y PulseSensor simultáneamente con precisión
BeagleBone Black	* Potente (similar a RPi) * Muchos puertos I/O	* Curva de aprendizaje alta * Escasa documentación educativa * Costo elevado	$55–60	Posible pero innecesario: Exceso de recursos para esta aplicación
LM35 (sensor de temperatura)	* Salida lineal (10 mV/°C) * Precisión ±0.5°C * Fácil integración	* Requiere contacto físico * Sensible a ruido eléctrico si no se filtra	$1.50	Seleccionado: Ideal para medir temperatura ambiente o por contacto

MLX90614 (IR)	* Medición sin contacto * Alta precisión (±0.1°C) * I²C directo	* Más costoso que sensores analógicos * Requiere 3.3V (no todos los modelos aceptan 5V)	$6–8	Seleccionado: Permite evitar contacto físico y mejorar higiene
PulseSensor	* Diseño optimizado para PPG * Circuito impreso con filtro RC * Open source y bien documentado	* Sensible a movimiento y luz ambiental * Requiere fijación adecuada (cinta, dedo)	$5	Seleccionado: Mejor relación calidad-precio para detección de pulso

Nota. Elaboración propia.

La combinación Raspberry Pi 3B+ y Arduino Uno fue seleccionado por su Complementariedad técnica entre ambos dispositivos, su bajo costo y su facilidad educativa.

Alternativas como ESP32 o NodeMCU reducen el tamaño, pero comprometen la estabilidad de las lecturas analógicas. En cambio, la arquitectura dual garantiza precisión, escalabilidad y mantenibilidad.

2.2. Selección de Software

En la tabla 6, podemos observar un cuadro comparativo respecto a la selección de software que mejor se acopla al diseño propuesto.

La selección del software se desarrolló de la misma forma que la de hardware, según las necesidades de la investigación, por lo que para la tarjeta lógica maestra se utilizó la programación Python y así poder establecer una comunicación con el microcontrolador Arduino, que es el encargado de recibir todos los datos de los sensores mediante la biblioteca PyFirmata. Además, para la creación de la interfaz gráfica se utiliza la biblioteca TkInter. Esta comunicación es de maestro a esclavo. Por otra parte, el Arduino, que es el esclavo, utiliza la librería Firmata para la comunicación con la tarjeta principal. Esta configuración se realiza en el entorno de desarrollo (IDE) de Arduino.

Tabla 6

Selección de Hardware

Característica	Sistema Propuesto	Arduino + Firmata + Python	MATLAB Live Script (Prototipo)	Node-RED + ESP32	Thinger.io + IoT Platform
Lenguaje principal	Python 3 + C++ (Arduino)	Python + Firmata	MATLAB	JavaScript (Node-RED)	C++ (ESP32) + JSON
Interfaz gráfica	Tkinter + Matplotlib (local)	Tkinter (lento con Firmata)	App Designer (licencia cara)	Dashboard web (básico)	Web dashboard (Thinger)
Latencia de visualización	< 120 ms (serial + root.after	250–500 ms (Firmata overhead)	> 300 ms (MATLAB GUI)	150–200 ms	200–400 ms (web latency)
Registro local	CSV + PNG (automático)	CSV (manual)	.mat (propio)	InfluxDB (requiere configuración)	MongoDB / SQLite
Envío a nube	ThingSpeak (HTTP GET)	ThingSpeak	ThingSpeak (Toolbox)	Ubidots / AWS IoT	Thinger.io nativo
Depuración en tiempo real	Consola con `[PULSO]`, `[TS]`	Limitada (sin print en Firmata)	Debug avanzado (pero costoso)	Flujo visual (Node-RED)	Logs remotos (limitados)
Requerimientos de hardware	Raspberry Pi 3B+/4 + Arduino	Mismo	PC con MATLAB License	ESP32 + Gateway	ESP32 + Internet estable
Curva de aprendizaje	Media (Python + Arduino básico)	Media-Alta (Firmata complejo)	Alta (MATLAB)	Baja-Media (Node-RED visual)	Media (JSON, APIs)
Escalabilidad	Alta (añadir sensores vía I²C/UART)	Media (Firmata	Baja (acoplado a MATLAB)	Alta (modular)	Media (plataforma cerrada)

		tiene límite de pines)			
Licencia	Open Source (GPL/MIT)	Open Source	Propietaria ($2,000+/año)	Open Source (Node-RED)	Freemium (Thinger)

Nota. Elaboración propia.

En este apartado se desarrollaron dos métodos para la comunicación entre el Arduino y la raspberry Pi para observa su eficacia, la primera fue con el uso del protocolo de comunicación Firmata, usado desde la librería de ArduinoIDE con el StandardFirmata, el cual permite que el Arduino actúe como una tarjeta de adquisición de datos controlada por la Rasberry.

El segundo método, fue desarrollar un sketch propio en Arduino, receptando los datos, encapsulándoles y enviándolos a la raspberry a través de su comunicación seria por el puerto USB, para su posterior procesamiento de datos.

La arquitectura serial + Raspberry Pi como maestro supera a Firmata en latencia y robustez, y es más accesible que soluciones basadas en MATLAB o plataformas cerradas.

2.3. Selección de Plataforma IoT

En la tabla 7, podemos Observar la selección mediante el cuadro comparativo de la plataforma IoT.

La programación de las tarjetas se las realiza por separado para luego poder hacer el enlace de los dos microcontroladores, así que el microcontrolador maestro es el encargado de administrar, dirigir y procesar toda la información obtenida del microcontrolador esclavo que contine los sensores. Entonces, la comunicación es por medio del puerto USB y todo se controla a través de la programación en Python por medio del controlador maestro.

Además, la información de los resultados se muestra por dos canales:

a) la interfaz nativa del microcontrolador y

b) la plataforma IoT.

Aunque la actualización de la información difiere para ambos se prefiere el ámbito local ya que no tiene retraso como la tiene la plataforma ThingSpeak que utiliza la IoT.

Tabla 7

Selección de Plataforma IoT

Plataforma	**ThingSpeak**	**Ubidots**	**Blynk**	**Thinger.io**	**AWS IoT Core**
Modelo de negocio	Gratuito (5 canales, 15s min)	Freemium (3 dispositivos gratis)	Freemium (1 proyecto gratis)	Freemium (10 dispositivos)	Pago por uso (costoso)
Facilidad de integración	Muy alta (HTTP GET/POST simple)	Alta (API REST + librerías)	Alta (app móvil + dashboard)	Media (requiere firmware específico)	Baja (AWS SDK, IAM, MQTT)
Latencia típica	1–3 s (por límite de escritura)	0.5–2 s	< 1 s (WebSocket)	1–2 s	0.2–1 s (MQTT)
Visualización en tiempo real	Gráficos básicos (web)	Dashboards personalizables	App móvil + web (muy intuitivo)	Web dashboard (limpio)	Requiere QuickSight o Grafana
Soporte para alertas	Email/SMS (premium)	Reglas + notificaciones	Botones, notificaciones, SMS	Webhooks, email	SNS + Lambda (complejo)
Seguridad	API Key (básica)	TLS + Token OAuth	TLS + Auth	TLS + JWT	IAM + X.509 + KMS
Documentación	Excelente (ejemplos en Python, Arduino)	Buena	Muy buena (tutoriales visuales)	Buena	Compleja (para expertos)

Ideal para	Proyectos educativos, prototipos rápidos	Emprendimientos escalables	Aplicaciones móviles rápidas	Prototipos con firmware personalizado	Soluciones industriales profesionales
Costo para 1 dispositivo	Gratis	Gratis (hasta 3 dispositivos)	Gratis (1 proyecto)	Gratis (10 dispositivos)	~$5/mes (mínimo viable)

Nota. Elaboración propia.

Se seleccionó ThingSpeak por su simplicidad, compatibilidad con los equipos a usar en este prototipo, documentación extensa y un modelo gratuito suficiente para prototipado educativo. No requiere configuración de MQTT ni certificados, lo que reduce la curva de aprendizaje.

2.4. Diseño Electrónico

El diseño electrónico se implementó sobre una placa de pruebas (protoboard), utilizando componentes comerciales de bajo costo y fácil adquisición. La arquitectura consta de tres bloques principales:

1. Plataforma maestra: Raspberry Pi 3 Model B v1.2
2. Plataforma esclava: Arduino Uno (como módulo de adquisición)
3. Sensores Fisiológicos: LM35, PulseSensor y MLX90614

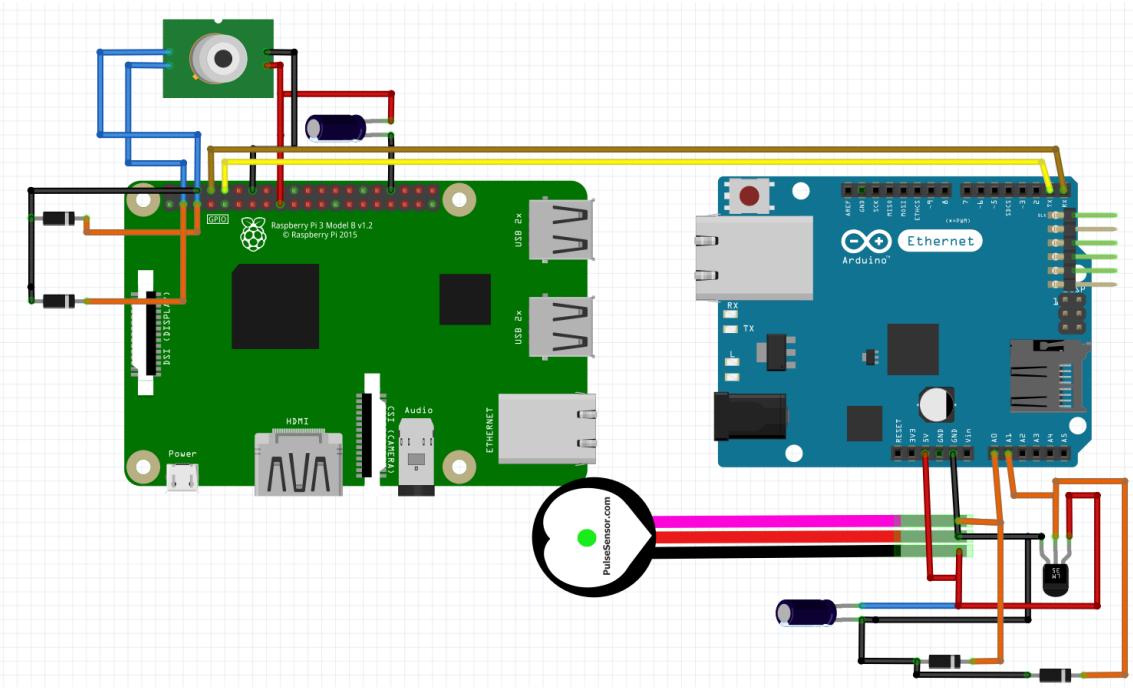
La comunicación entre Raspberry Pi y Arduino se realiza mediante USB tipo B–A, aprovechando el puerto serial nativo del Arduino y la capacidad de la Raspberry Pi para emular un terminal USB.

2.4.1. Simulación física del circuito Electrónico

Con la ayuda del software Fritzing, se simula la conexión electrónica entre la tarjeta de control y los sensores Fisiológicos respectivos como se observa en la figura 3.

Figura 3

Simulación de componentes electrónicos Físicos.



Nota. Se simula la conexión de los componentes físicos en cada pin de las tarjetas.

Elaboración propia.

Ponemos observar los componentes físicos que involucran el prototipo, así como sus respectivas protecciones como el uso de diodos 1N4148, de alta frecuencia en modo Clamping, para las protecciones de las señales SDA y SCL de los sensores que se conectan a la raspberry y de igual forma la protección en las señales análogas de los sensores que están conectados al Arduino.

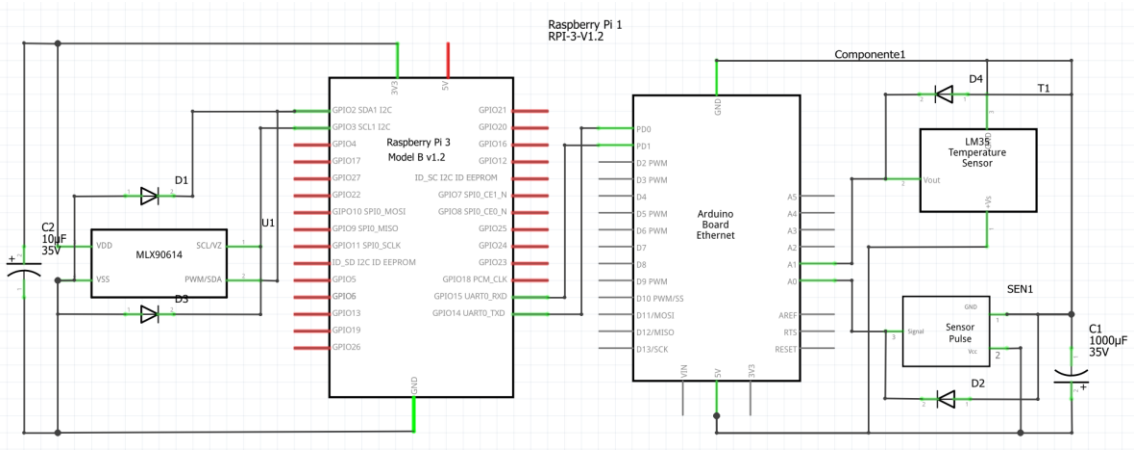
De esta misma manera ubicamos dos capacitores en las entradas de alimentación de cada sensor, en la entrada 3,3Vdc de la Raspberry Pi como en la entrada de 5Vdc de la tarjeta Arduino.

2.4.2. Esquemático del circuito

En la Figura 4. Podemos observar el esquemático para tener una visualización más clara de cada una de las conexiones y a los pines que le corresponde cada elemento.

Figura 4

Esquemático del circuito electrónico



Nota. Circuito esquemático elaborado em Fritzing. Elaboración propia.

La implementación física, se realizó sobre una placa de pruebas estándar de 102 puntos, los sensores Lm35 y Pulse sensor se conectan a la placa Arduino debido a sus entradas analógicas, en paralelo se conecta un capacitor electrolítico en sus entradas de alimentación, para Filtrar ruido de alta frecuencia y estabilizar la tensión de alimentación y señal, mejorando la calidad de la lectura analógica.

En la tabla 8, se detalla los diodos de protecciones ubicados en el circuito electrónico

Tabla 8

Diodos de protección electrónico

Punto del circuito	Tipo de protección	Componente recomendado
Entrada A0 (PulseSensor)	Clamping de señal	1N4148 x2 (a VCC y GND)
Entrada A1 (LM35)	Clamping de señal	1N4148 x2 (a VCC y GND)
Línea SDA (MLX90614)	Transitorio / ESD	1N4148 o TVS (ESD5Z5V0)
Línea SCL (MLX90614)	Transitorio / ESD	1N4148 o TVS (ESD5Z5V0)

Nota. Elaboración propia.

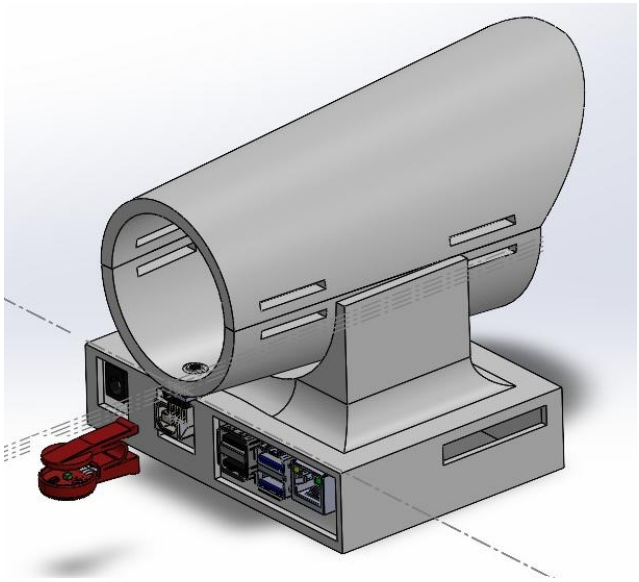
2.5. Diseño y modelado CAD

Con el uso del programa SolidWorks, se realiza un diseño en 3D, ubicando los sensores en puntos específicos, donde se recolectará los datos del paciente, este tiene el ensamble de los microcontroladores Arduino y Raspberry pi3 B+, en la parte inferior del diseño, conservando entradas para los puertos periféricos de cada uno, este diseño se puede imprimir en 3D.

Para mejor facilidad, en la Figura 5, observamos el diseño, pensado en la postura de un antebrazo, el cual el cliente solo asentara su brazo en el prototipo, y este encendiendo realizara la medición de sus signos vitales, así como realizar la conexión a internet y enviar los datos a la nube IoT y su grafico en área local.

Figura 5

Diseño CAD del prototipo

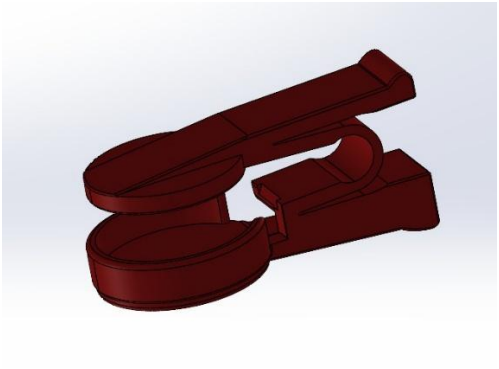


Nota. Diseño realizado en SolidWorks 2025. Elaboración propia.

Los case del sensor que miden el ritmo cardiaco como SensorPulse, se elaboró aparte, ya que este será ubicado en la yema de los dedos de los usuarios, en la figura 6, podemos observar el diseño de cada uno de ellos.

Figura 6

Case del sensor PulseSensor



Nota. Diseño elaborado en SolidWorks. Elaboración propia.

En los Anexos adjuntos se detalla todos los planos y dimensiones relacionados al case del prototipo.

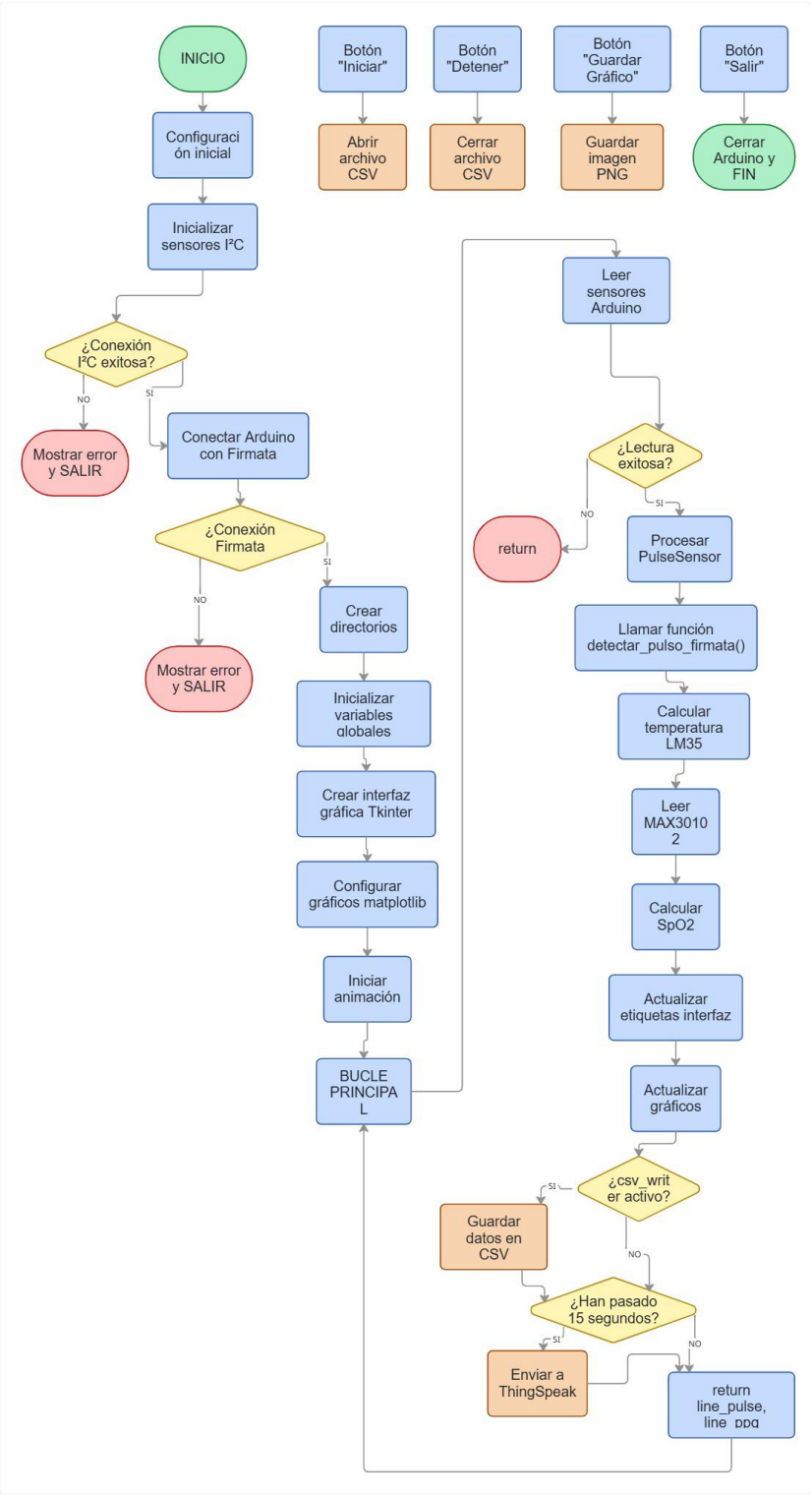
2.6. Diagrama de Flujo de la programación del Sistema

Se Realizo la programación en Python y en Arduino, la primera prueba se usa el protocolo Firmata en Arduino, y en Python se usa la librería PyFirmata, para usar la tarjeta Arduino como esclavo y como una tarjeta de adquisición de señal, a medida que realizamos pruebas, se pudo notar un ligero retraso al momento de enviar los datos a la nube de ThingSpeak.

En la Figura 10, podemos observar el diagrama de Flujo de esta programación, y en los Anexos tenemos toda el pseucódigo de la programación.

Figura 7

Diagrama de Flujo del programa



Nota. Elaboración propia.

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

3.1.Construcción y Evaluación del prototipo

3.1.1. Conectividad VNC Viewer

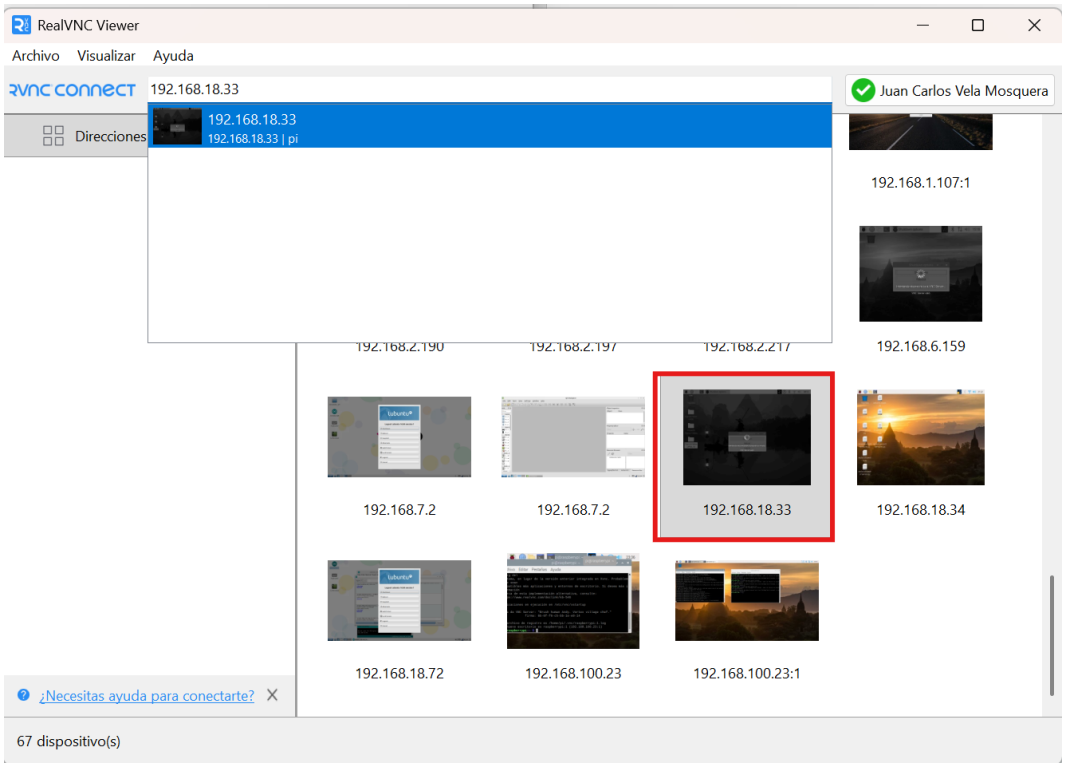
La aplicación VNC Viewer permite tener acceso remoto a la interfaz gráfica de la Raspberry Pi. De esta manera, es posible conectarse al sistema desde cualquier dispositivo que soporte la aplicación, ya sea una laptop, tablet o teléfono móvil.

La conexión se realiza dentro de un área de red local (LAN) utilizando la dirección IP asignada a la Raspberry Pi. Esta dirección IP es otorgada automáticamente por el router cuando el dispositivo se conecta mediante WiFi o cable Ethernet. Para identificar qué IP fue asignada, se puede utilizar cualquier aplicación de escaneo de red que permita visualizar los dispositivos conectados.

En este caso, como se muestra en la Figura 8, la dirección IP asignada a la Raspberry Pi fue *192.168.18.33*, la cual permitió establecer la conexión remota mediante VNC Viewer.

Figura 8

Interfaz VNC con Raspberry Pi 3B+

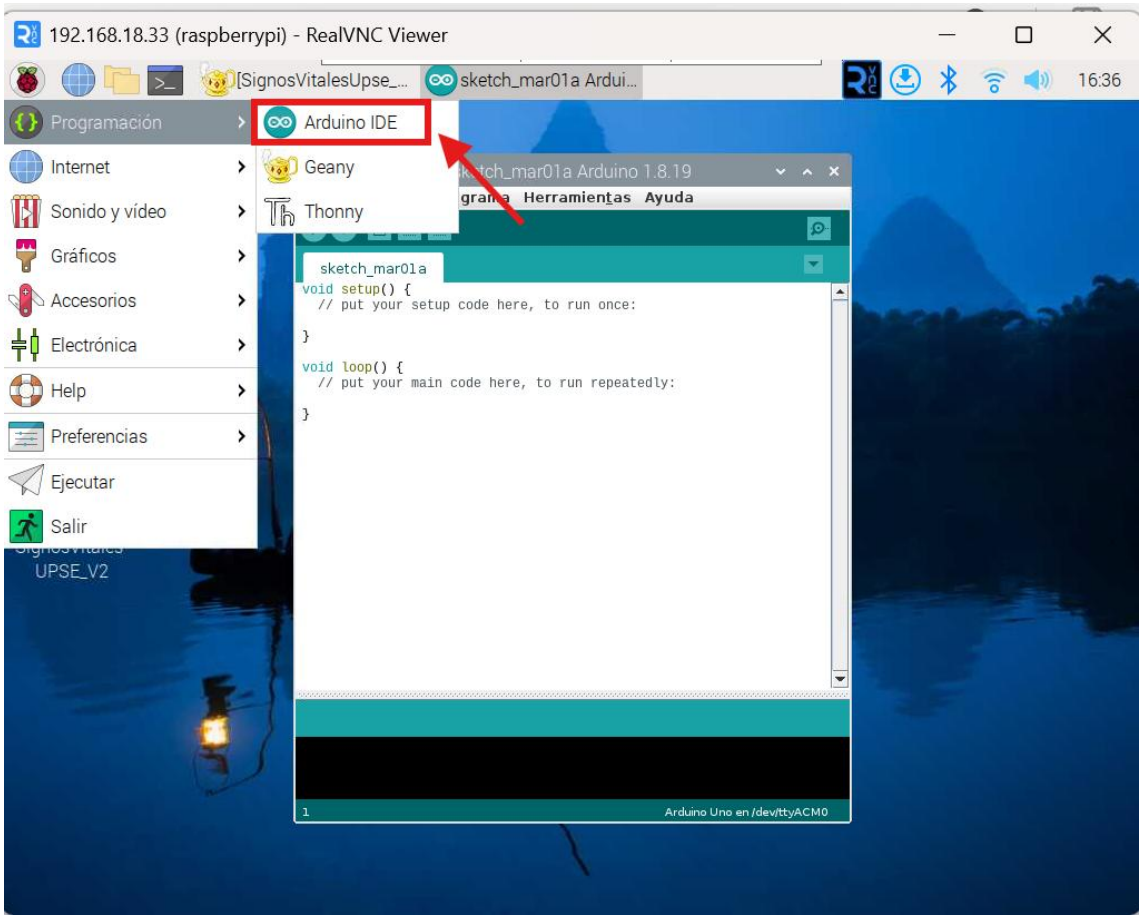


Nota. Interfaz de VNC para acceder a Raspberry en área local. Elaboración propia.

Una Vez ingresado por la IP, te pedirá un usuario y contraseña del acceso a la tarjeta Raspberry pi3 B+, en este caso se usó la contraseña por defecto que es *usuario: pi* y como *contraseña: raspberry*, una vez dentro estarás en la interfaz de la tarjeta Raspberry pi3 B+, como se observa en la figura 9.

Figura 9

Interfaz de la Raspberry por medio de VNC Viewer



Nota. El IDE de Arduino viene por defecto en la última actualización del SO de Raspberry. Elaboración propia.

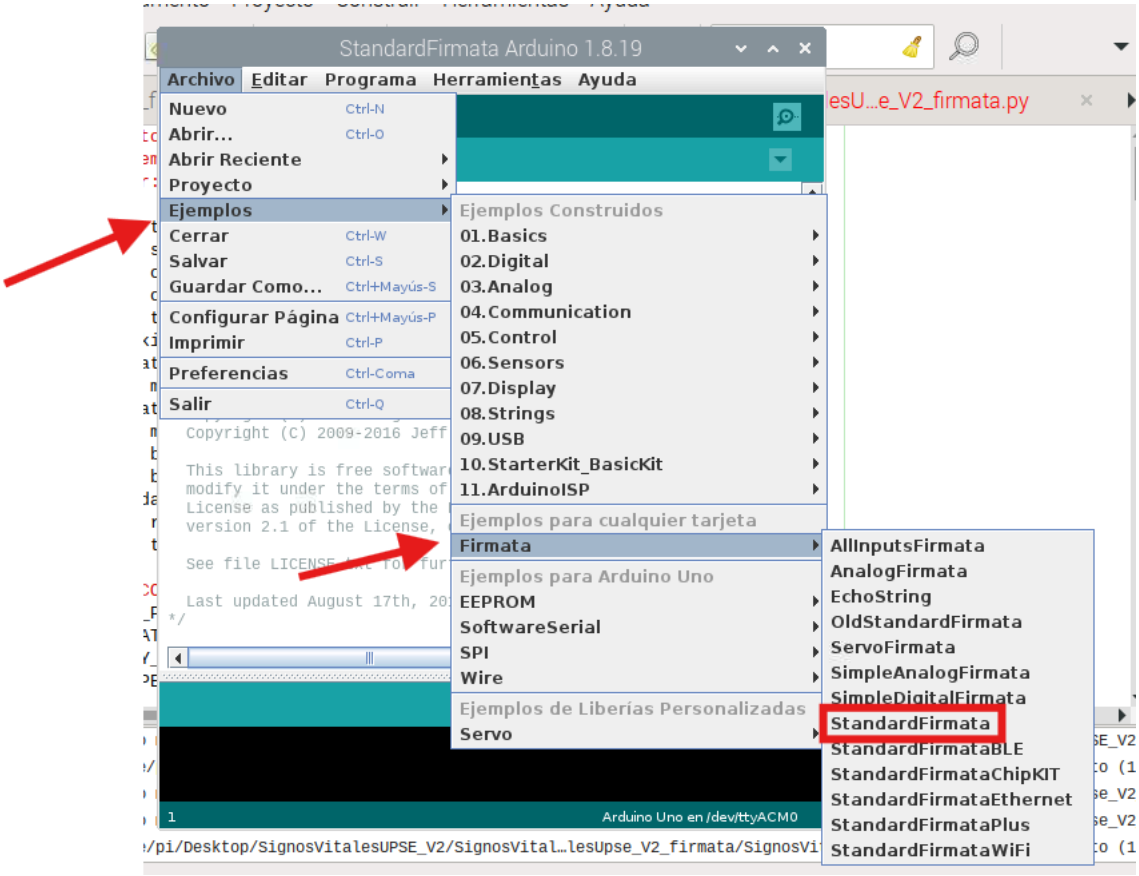
Ya dentro de la tarjeta se procede a realizar la programación en Python con el editor que prefieras, para este proyecto se usó el IDE Geany, y para Arduino su editor por defecto.

El IDE de Arduino está integrado dentro de la plataforma de Raspberry, al no ser el caso, se puede descargar e instalar en la misma tarjeta, la librería se debe de descargar con el administrador de librerías dentro de la misma interfaz, se debe descarga el archivo

StandardFirmata para cargar al Arduino y proceder con la compilación y carga del programa como se observa en la figura 10.

Figura 10

Carga de archivo StandardFirmata



Nota. Interfaz dentro de la placa Raspberry Pi3. Elaboración propia.

La estructura del código en Python en ambos casos es similar, con la diferencia que con el uso de firmata se debe usar la librería *PyFirmata import Arduino, útil,* y la estructura que este conlleva.

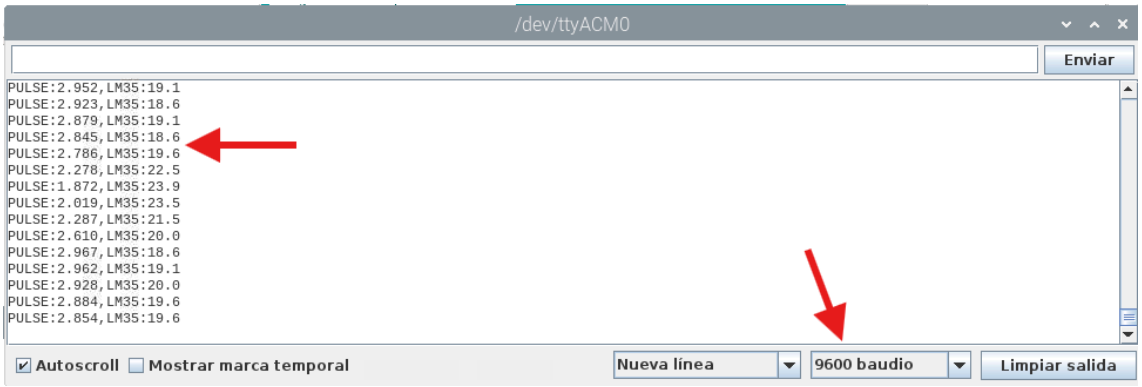
Mientras que, usando un Sketch, desarrollado desde cero, infiere en la programación desde Python, ya que al primero se debe usar la librería *import serial*, el cual abre el puerto serial de la Raspberry para comunicarse con el Arduino, el cual también tiene que abrir su puerto serial y ambos sincronizan su velocidad en 9600 baudio y poder enviar y recibir bien los datos. En los anexos se plasma todos los códigos usados en ambos casos para su posterior estudio, en la Figura 11, visualizamos el puerto serie del Arduino en el

segundo caso, esto para realizar test de los dos sensores que se encuentran en los puertos análogos A0 y A1, podamos ver el dato que se envía al maestro. Cabe recalcar que la linealización

de la señal se la realiza en la misma tarjeta Arduino, a diferencia del uso de firmata que se realizaba la linealización en Python.

Figura 11

Monitor Serial de Arduino



Nota. Uso del monitor Serial de Arduino para realizar test de los sensores. Elaboración propia.

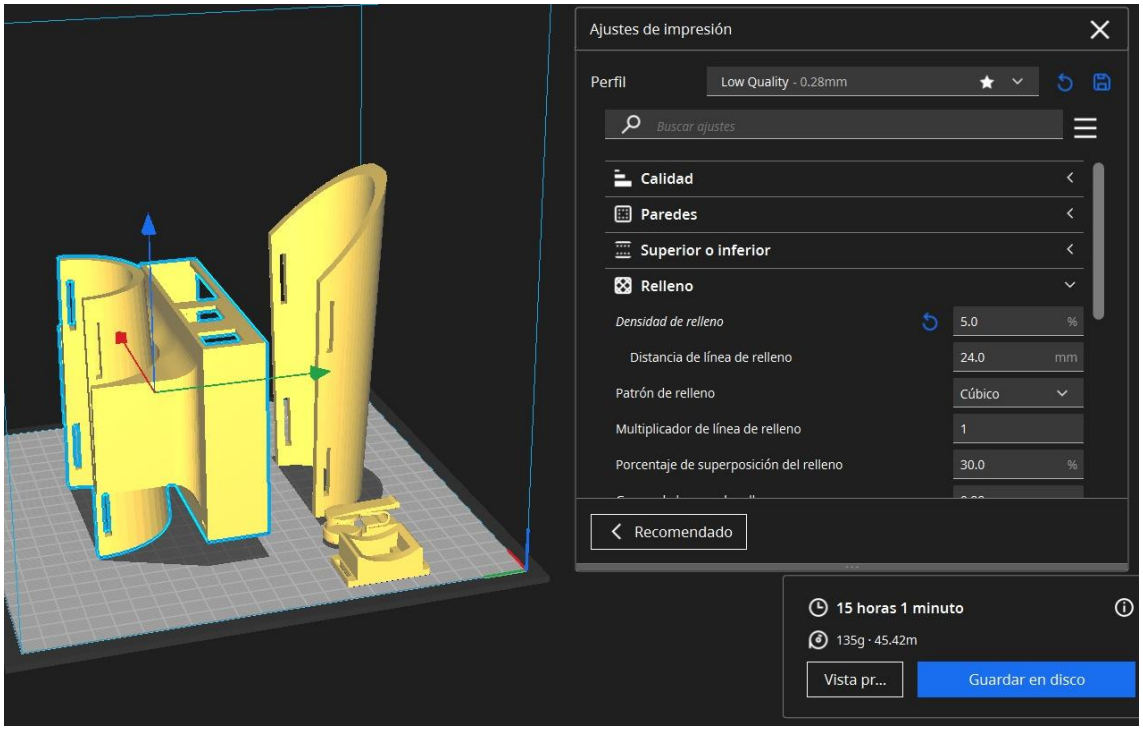
Así como en el monitor serial se realiza un test de los sensores, también se realiza un test por el terminal de Python, para observar los datos que vienen desde el Arduino, y del sensor que está conectado en la raspberry con el protocolo I2C, observamos el terminal con los datos listo para ser enviados a la plataforma IoT ThingSpeak,

3.1.2. Impresión en 3D

Se realiza la impresión de cada parte del case del prototipo, con una impresora 3D de modelo Ender 3pro, de la marca Creality, la cual mediante el software CURA, nos permite generar los archivos en código G, para enviarlo a la máquina de impresión, en la figura 12, se observa los parámetros de impresión como el tiempo aproximado de 15 horas de impresión para todos los elementos, con un relleno del 5% y una resolución de 0.28mm.

Figura 12

Configuración de impresión en 3D

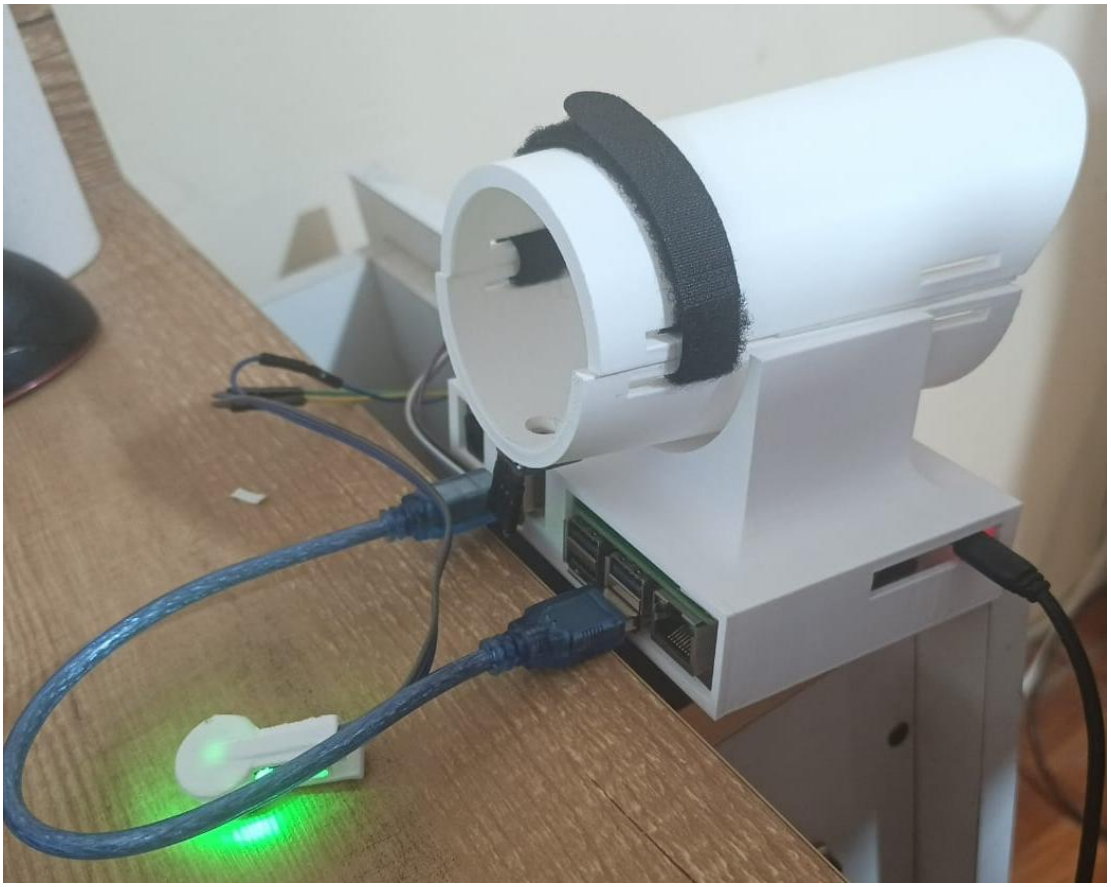


Nota. El código G, fue elaborado por el software Cura. Elaboración propia.

El prototipo fue implementado utilizando los sensores previamente descritos, interconectados mediante cables de cobre AWG 22. La Figura 13 muestra la implementación física del sistema ensamblado sobre una placa de pruebas, listo para operar. Inicialmente, se diseñó una carcasa funcional en 3D mediante modelado CAD, para optimizar la disposición de componentes y facilitar su uso ergonómico, en la figura 13 podemos ver como quedo una vez impreso el modelo previamente Elaborado.

Figura 13

Prototipo Ensamblado



Nota. Prototipo de sistema de monitoreo de signos vitales ensamblado y puesta en marcha. Elaboración propia.

La Figura 14, nos muestra la puesta en marcha del prototipo, en la imagen se aprecia como el usuario se coloca el dispositivo previo a las indicaciones de uso, sin dificulta y mostrando su eficiencia en diseño.

Figura 14

Prototipo en el antebrazo del Usuario

Nota. Elaboración Propia

En la Figura 15, el usuario observa sus datos corporales, así como la gráfica de la frecuencia cardiaca, gracias al acceso remoto a la interfaz de la Raspberry, se realiza un análisis haciéndole consulta a los usuarios, preguntándole que tan difícil se les hizo el uso del prototipo, si la interfaz es comprensible, y se recibieron respuestas positivas, aunque algunas recomendaciones respecto a la estructura para que sea más de goma y menos rígida, la cual es factible superarlo si se imprime en materiales Flexibles como el TPU.

Figura 15

HMI del prototipo en funcionamiento

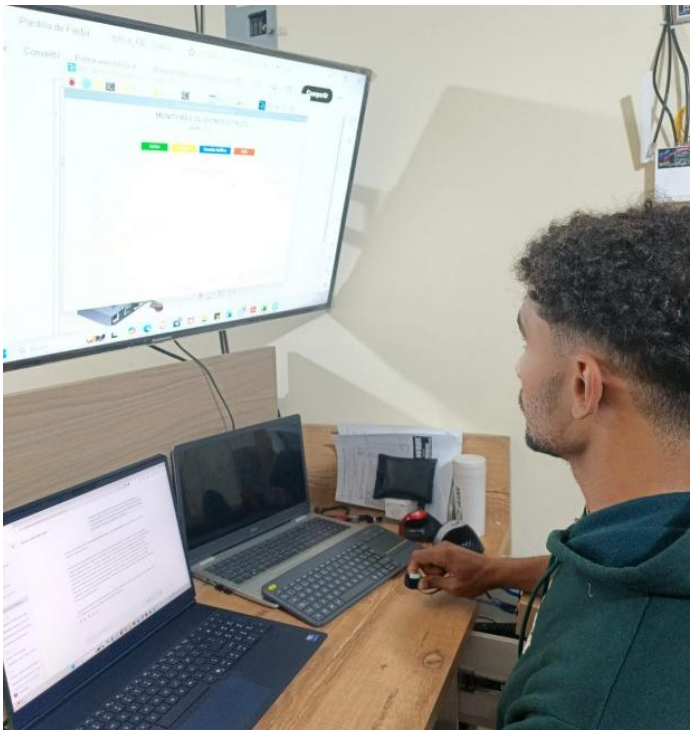
Nota. Elaboración Propia

Los sensores en las posiciones previamente enunciadas, permitiendo que estos recopilen los datos y sea procesado por la raspberry para posteriormente realizar alguna especie de tratamiento de datos. en la Figura 15 y 16. El usuario es capaz de observar las gráficas de sus signos vitales, así como el valor de su temperatura corporal y ambiente, los cuales serán guardados en un archivo .csv, y estos valores puedan ser analizados.

Además de tener un menú muy dinámico, que le permite al profesional que use el sistema de poder seleccionar entre opciones, las cuales les permite tomar los datos desde un inicio a un final de tiempo que el crea conveniente, así como capturar el grafico para ser usado como mejor vea necesario, esta toma de datos además de capturar los valores de los sensores, también guarda el día, y la hora de inicio y de fin de la toma de datos.

Figura 16

Usuario 2, observando sus signos vitales



Nota. Elaboración Propia

El sistema permite monitoreo remoto mediante Real VNC Viewer, que ofrece acceso gráfico seguro y eficiente a la interfaz alojada en la Raspberry Pi, facilitando el control sin necesidad de un monitor físico conectado directamente al dispositivo. Esta solución resulta especialmente útil en entornos educativos y de desarrollo, donde se requiere supervisión remota del sistema embebido.

En cuanto a la arquitectura de comunicación, se evaluaron dos enfoques:

1. Uso de pyFirmata para lectura remota de pines desde Python.
2. Implementación de un sketch personalizado que transmite datos estructurados por puerto serial (sin Firmata).

Los resultados mostraron diferencias significativas en latencia: el uso de pyFirmata introdujo un retardo promedio de 250–500 ms debido al protocolo cliente-servidor y al muestreo periódico, mientras que el sketch personalizado redujo la latencia a 50–120 ms, mejorando notablemente la sincronización entre la señal física y su representación en la interfaz gráfica desarrollada con tkinter y matplotlib.

La interfaz gráfica que observamos en la figura 17, es la que se ejecuta en la raspberry de forma local, desarrollada en Python, ejecuta y muestra en tiempo real:

- Señal PPG (gráfico),
- BPM, temperatura corporal (LM35) y ambiente (MLX90614),
- Estado del sistema.

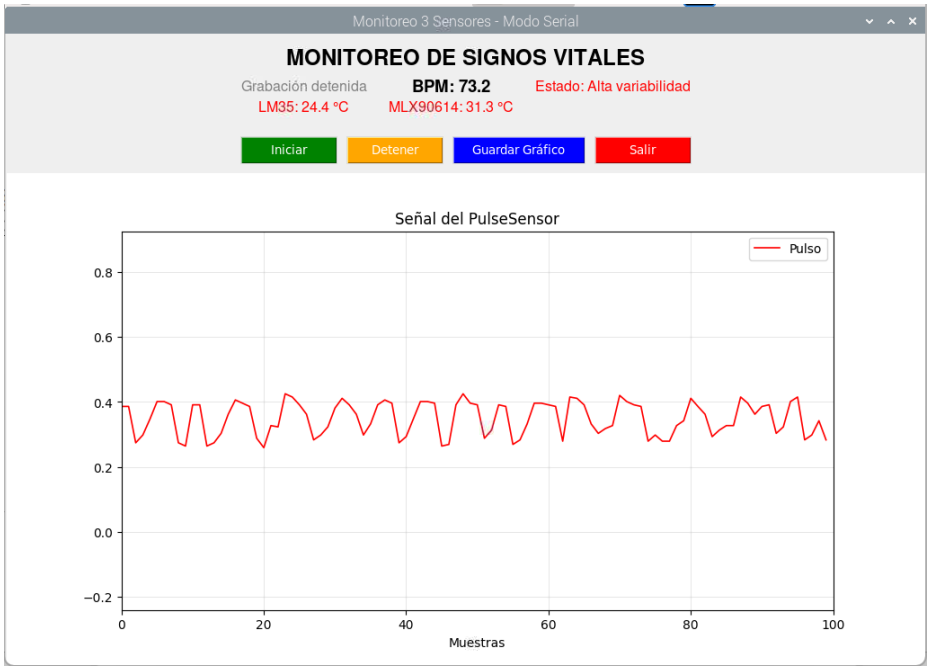
Actualización cada 50 ms mediante root.after(), sin bloqueo.

Cada uno de los widgets usados en la interfaz, realizan una tarea especifica que se describe a continuación.

- Iniciar: Activa adquisición, crea CSV y comienza el bucle.
- Detener: Detiene grabación y cierra el archivo.
- Guardar Gráfico: Exporta el gráfico actual como PNG con timestamp.
- Salir: Cierra limpiamente la app y libera recursos.

Figura 17

Interfaz del prototipo.

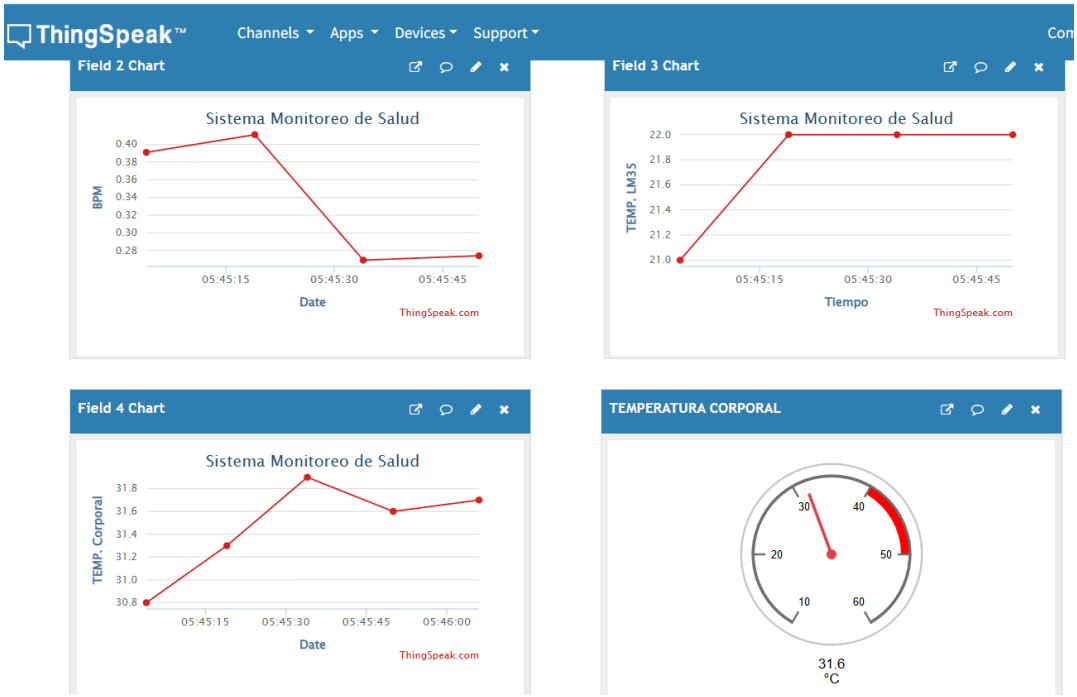


Nota. Elaboración propia

La ejecución y obtención de datos se realizan desde la consola, donde se visualizan los datos recolectados por los sensores biomédicos. Las lecturas se muestran en tiempo real los datos como la frecuencia cardíaca y la temperatura ambiente como la corporal; por lo tanto, se evidencia la correcta transmisión de información entre los dispositivos y la plataforma IoT como observamos en la Figura 18. Además, este registro confirma la baja latencia y la excelente estabilidad en la arquitectura de comunicación implementada,

Figura 18

Envío de datos a ThingSpeak



Nota. Envío de datos a ThingSpeak con un promedio de 15s de actualización en esta plataforma IoT. Elaboración propia.

Los resultados obtenidos durante la evaluación del prototipo se muestran en la Tabla 9, los datos demuestran que el prototipo es capaz de realizar un monitoreo remoto eficiente y estable de los signos vitales

Los resultados que se obtuvieron permiten evidenciar que este prototipo puede adaptarse de forma precisa a entornos con un sistema IoT, además de resultar ser 100% funcional y confiable para poder realizar un monitoreo de forma remota de signos vitales en las personas de prueba. Se puede registrar una baja latencia confirmando la eficiencia en la arquitectura que se tenía en la comunicación implementada, además de que la estabilidad

en las mediciones permitió validar una selección apropiada de los sensores biomédicos que se emplearon en este prototipo. Este tipo de hallazgos permite sustentar que el prototipo resulta ideal como un tipo de herramienta que brinde apoyo en aplicaciones de la salud digital y en esquemas para monitoreo continuo, aportando con esto una alternativa tecnológica que es de bajo costo, con potencial de poder integrarse en entornos clínicos y comunitarios adecuadamente.

Tabla 9

Tabulación de datos.

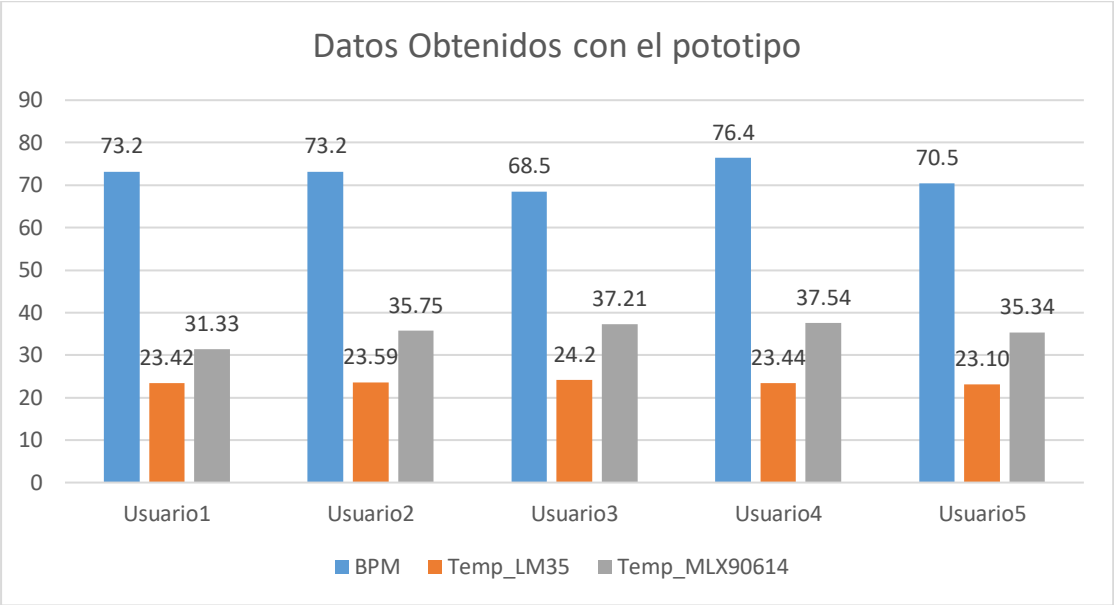
#Usuario	BPM	Temp_LM35	Temp_MLX90614
Usuario1	73.2	23.42	31.33
Usuario2	73.2	23.59	35.75
Usuario3	68.5	24.2	37.21
Usuario4	76.4	23.44	37.54
Usuario5	70.5	23.10	35.34

Nota. Se obtuvo el resultado de 5 usuarios, para determinar su eficiencia. Elaboración propia

Se procedió a graficar los datos obtenidos de los 5 usuarios, como se observa en la figura 19 a través de un gráfico de barras.

Figura 19

Gráfico de barra, de los datos de muestra.



Nota. Elaboración propia.

a los presentes en el mercado, además de ser opensource, nos permite seguir desarrollando mejoras y encontrando soluciones accesibles al problema de salud. Sin embargo, el tamaño reducido que se tenía para realizar las pruebas también limita el llegar a generalizar los resultados, pero este prototipo demostró ser una herramienta confiable si se lo llega a comparar con otro tipo de dispositivos que se encuentran en mercados existentes.

3.2. Análisis y Discusión de Resultados

Los datos de los usuarios muestran, que los latidos por minutos de los participantes, van desde los 68.5BPM hasta los 76.4BPM, que se encuentra en el rango normal de una persona adulta sedentaria que varían desde los 60BPM a 100BPM, Así como observamos que las temperaturas de cada usuario varia relativamente, estas llegan a bajar hasta los 31.33°C en un ambiente frío, como se observa la temperatura ambiente si permanece más estable, ya que no varía mucho las muestras se tomaron de manera consecutiva.

Estos valores varían porque los pacientes presentaban condiciones físicas muy diferentes, tanto en edad, sexo y condiciones físicas, adicional que se tomó en días diferentes y con

temperaturas medio ambientales diferentes, pero conservan una similitud, la población de muestra varían entre la edad de 25 años a 38 años.

El sistema propuesto no busca competir con dispositivos médicos certificados, sino ofrecer una plataforma abierta, reproducible y pedagógicamente efectiva para la enseñanza de sistemas embebidos, procesamiento de señales y IoT en salud. Su bajo costo y modularidad lo posicionan como una alternativa viable frente a soluciones comerciales opacas y costosas.

Cabe recalcar que este no es un sistema considerado para el uso medico completamente, ya que carece de certificaciones para su validación, pero deja la puerta abierta para un desarrollo accesible a bajo costo y funcional.

Los resultados que se obtuvieron son satisfactorios con investigaciones previas que permiten destacar la eficiencia que se tiene en estos sistemas IoT para monitoreos de signos vitales remotos en tiempo real. La baja latencia que presentó este dispositivo confirmó que la arquitectura de comunicación resultaba ser adecuada, en comparación con otros dispositivos en el mercado, la relación costo-beneficio de este proyecto supera

CONCLUSIONES

Los participantes evaluaron el prototipo como práctico, intuitivo y de fácil manejo, destacando la simplicidad en la colocación de los sensores y la claridad de la interfaz gráfica, que permite visualizar en tiempo real la señal PPG, el BPM y las temperaturas corporal y ambiental. La disponibilidad inmediata de los datos en formato estructurado (CSV) y su representación gráfica facilitan el análisis cuantitativo y cualitativo de las señales fisiológicas.

La implementación demostró que es posible construir un dispositivo funcional, de bajo costo y código abierto, capaz de registrar latidos por minuto (BPM), temperatura corporal y ambiente con precisión razonable. Los resultados mostraron valores de frecuencia cardíaca entre 68.5 y 76.4 BPM, dentro del rango fisiológico normal para adultos en reposo (60–100 BPM), validando la capacidad del sistema para capturar señales fisiológicas coherentes.

Además, se evaluaron dos estrategias de comunicación con Arduino: el uso de pyFirmata y un sketch personalizado con transmisión serial directa. Se comprobó que este último reduce significativamente la latencia (de ~500 ms a ~50–120 ms), mejora la fluidez del HMI y elimina problemas de buffer overflow, resultando en una actualización más precisa tanto del dashboard local como del canal en ThingSpeak.

El uso de pyFirmata introduce un alto overhead en la comunicación con Arduino, generando latencias de 250–500 ms, lo que afecta negativamente la visualización en la nube. En cambio, al utilizar un sketch personalizado que envía datos por serial, la latencia se reduce a 50–120 ms, mejorando significativamente la fluidez del sistema. La visualización en tiempo real es más efectiva con este enfoque, tanto en la interfaz local como en la plataforma IoT (ThingSpeak), donde se observa menor desfase y mayor coherencia entre la señal física y su representación remota.

RECOMENDACIONES

Preferir la comunicación serial directa frente a pyFirmata para aplicaciones en tiempo real.

Aunque pyFirmata facilita el prototipado inicial, introduce latencia significativa que afecta el envío a plataformas IoT como ThingSpeak. Se recomienda usar un sketch personalizado en Arduino que envíe datos estructurados por puerto serial, lo cual mejora la sincronización y precisión temporal.

Implementar protección eléctrica básica en todas las señales

Colocar diodos 1N4148 como clamping entre GND y las líneas analógicas (A0, A1), y entre VCC y señal, protege los pines del Arduino de picos de voltaje. De igual forma, se recomienda agregar resistores pull-up de 4.7 kΩ en las líneas I²C (SDA/SCL) para mejorar la estabilidad del bus.

Cambiar la contraseña predeterminada del usuario 'pi' en Raspberry Pi

Por motivos de seguridad, especialmente si se habilita SSH o acceso remoto mediante RealVNC, es fundamental cambiar la contraseña por defecto usando el comando passwd, evitando así accesos no autorizados al sistema.

REFERENCIAS

Abdulmalek , S., Nasir, A., Jabbar, W., & Almuhaya, M. (2022). IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review. *Healthcare, 10*(10). https://doi.org/10.3390/healthcare10101993

Abdulmalek S, N. A. (2022). IoT-Based Healthcare-Monitoring System towards Improving Quality of Life. *Healthcare (Basel), 10*. https://doi.org/10.3390/healthcare10101993

Abu-Jassar, A., Attar, H., Amer, A., & Lyashenko, V. (2025). Remote Monitoring System of Patient Status in Social IoT Environments Using Amazon Web Services Technologies and Smart Health Care. *International Journal of Crowd Science, 9*(2), 110 .125. Obtenido de https://www.sciopen.com/article/pdf/10.26599/IJCS.2023.9100019.pdf

Afifi, S. G. (2025). Smart Health Monitoring System for Realtime Measurement of Vital Signs. *nternational Conference on Artificial Intelligence, Computer, Data Sciences and Applications, 5*. https://doi.org/https://doi.org/10.1109/acdsa65407.2025.11166087

Akkaş, M., SOKULLU, R., & Çetin, H. E. (2020). Healthcare and patient monitoring using IoT. *Internet of Things, 11*, 1. https://doi.org/https://doi.org/10.1016/j.iot.2020.100173.

Al-Fuqaha, A. G. (2015). Internet of Things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials, 17*(4). https://doi.org/https://doi.org/10.1109/COMST.2015.2444095

Andrade, A., Tassinari Cabral, A., Bellini, B., & Facco Rodrigues, V. (2024). IoT-based vital sign monitoring: A literature review. *Smart Health, 32*, 2. https://doi.org/https://doi.org/10.1016/j.smhl.2024.100462

Arandia N, G. J. (2022). Sistemas de sensores embebidos en dispositivos médicos: requisitos y desafíos por venir. *Sensores (Basilea)*(3). https://doi.org/10.3390/s22249917

Chakraborty, A. G. (2023). Development of an IoT-enabled cost-effective asthma patient monitoring system: Integrating health and indoor environment data with

statistical analysis and data visualization. *Internet of Things, 24*. https://doi.org/https://doi.org/10.1016/j.iot.2023.100942

Chen , Q., & Sheng, N. (2024). A novel health monitoring system for vital signs using IoT. *Scientific Reports, 14*. Obtenido de https://www.nature.com/articles/s41598-024-69257-y

Elango, S. M. (2023). Super Artificial Intelligence Medical Healthcare Services and Smart Wearable System based on IoT for Remote Health Monitoring. *International Conference on Smart Systems and Inventive Technology (ICSSIT), 5*. https://doi.org/https://doi.org/10.1109/icssit55814.2023.10060928

God, I., Jagtap, S., Suryavanshi, C., & Pawar, B. (2024). IoT Based Health Monitoring System. *Applied Science and Engineering Technology*. https://doi.org/https://doi.org/10.22214/ijraset.2024.58454

Gopichand, G. S. (2024). Use of IoT sensor devices for efficient management of healthcare systems. *Discover Internet of Things, 4*. https://doi.org/https://doi.org/10.1007/s43926-024-00062-9

Gubbi, J. B. (2013). A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7).

Hall, J. E. (2021). *Guyton and Hall Textbook of Medical Physiology.* Elsevier.

Harez, J. B. (2024). IoT-Driven System for Continuous Monitoring of Heart Disease Patients Post-Surgery. *arXiv, 1*. https://doi.org/https://doi.org/10.48550/ARXIV.2410.11269

ISO. (10 de 12 de 2025). *www.iso.org*. Obtenido de https://www.iso.org/standard/59752.html

Khan , A., & Duncan , D. (2025). Remote Patient Monitoring Applications in Healthcare: Lessons from COVID-19 and Beyond. *Electrónica, 14*(15). https://doi.org/https://doi.org/10.3390/electronics14153084

Kristoffersson A, L. M. (2020). A Systematic Review on the Use of Wearable Body Sensors for Health Monitoring. *Sensors (Basel, Switzerland), 20*(5), 1-2. https://doi.org/10.3390/s20051502

Ksibi, S. J. (2024). MLRA-Sec: an adaptive and intelligent cyber-security-assessment model for internet of medical things (IoMT). *International Journal of Information Security, 24*. https://doi.org/https://doi.org/10.1007/s10207-024-00923-y

Manzano Ramos, E. A. (2021). Implementación de un Sistema de Monitoreo a nivel de prototipo de signos vitales: pulso, temperatura y saturación de oxígeno para pacientes. *Interfases, 14*. https://doi.org/https://doi.org/10.26439/interfases2021.n014.5168

MedlinePlus. (13 de 11 de 2025). *medlineplus.gov.* Obtenido de https://medlineplus.gov/spanish/ency/article/002341.htm

Mishra, A. D. (2024). Real-time Vital Signs Monitoring and Data Management Using a Low-Cost IoT-based Health Monitoring System. *Journal of Health Management, 26*. https://doi.org/https://doi.org/10.1177/09720634241246926

Mohammed, S., Saleh , A., Alshaikh, I., & Mohammed, A. (2020). Real-time health monitoring using IoT devices for patients with chronic conditions. *International Journal of Health Sciences, 4*(1), 214–228. https://doi.org/https://doi.org/10.53730/ijhs.v4nS1.15149

MSP. (29 de 09 de 2020). *Ministerio de Salud Pública*. Obtenido de https://www.salud.gob.ec/msp-previene-enfermedades-cardiovasculares-con-estrategias-para-disminuir-los-factores-de-riesgo/

PubMed Central PMC. (13 de 09 de 2025). *PMC*. Obtenido de https://pmc.ncbi.nlm.nih.gov/

Python. (20 de 10 de 2025). *Python.org*. Obtenido de https://docs.python.org/es/3.9/library/tk.html

Taherdoost, H. (2024). Wearable Healthcare and Continuous Vital Sign Monitoring with IoT Integration. *Computers, Materials and Continua,, 81*(1), 79-104. https://doi.org/https://doi.org/10.32604/cmc.2024.054378

Thippeswamy, V. S. (2021). Prototype Development of Continuous Remote Monitoring of ICU Patients at Home. *Instrumentation Mesure Métrologie, 20*. https://doi.org/https://doi.org/10.18280/i2m.200203

Valvano, J. W. (2017). *mbedded systems: Introduction to ARM Cortex-M microcontrollers.* CreateSpace.

Vela Mosquera, J. C. (2024). Propuesta de un sistema de monitoreo de indicadores de salud mediante IOT para la toma de signos vitales. *Revista Social Fronteriza, 4*. https://doi.org/https://doi.org/10.59814/resofro.2024.4(1)169

Vijayalakshmi Subramanian, S. D. (2023). Smart medical sensor. *Security and Privacy Issues in Internet of Medical Things*. https://doi.org/https://doi.org/10.1016/B978-0-323-89872-0.00005-8.

World Health Organization. (15 de 12 de 2025). *www.who.int*. Obtenido de https://www.who.int/

Yogi, K. S. (2024). Enhancing Healthcare Delivery Through IoT Applications in Remote Patient Monitoring and Telemedicine. *IEEE North Karnataka Subsection Flagship International Conference*.

Zovko, K. Š. (2023). IoT and health monitoring wearable devices as enabling technologies for sustainable enhancement of life quality in smart environments. *Journal of Cleaner Production, 413*. https://doi.org/https://doi.org/10.1016/j.jclepro.2023.137506

ANEXOS

Anexo 1: CÓDIGO DE PROGRAMACIÓN

1er Caso. con el uso del protocolo de comunicación Firmata y la librería PyFirmata

```
# SignosVitalesUpse_V2_firmata.py
# Sistema de monitoreo de signos vitales - Versión Final con depuración
# Autor: Juan Carlos Vela - UPSE

import time
import csv
import os
import tkinter as tk
from tkinter import Label, Button, Frame, messagebox
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.animation as animation
import board
import busio
from adafruit_mlx90614 import MLX90614
import requests
from pyfirmata import Arduino, util

# --- CONFIGURACIÓN ---
PORT_ARDUINO = '/dev/ttyACM0'
API_KEY_THINGSPEAK = '8FQPQNAL5O0J23E3'
THINGSPEAK_URL = 'https://api.thingspeak.com/update'

# Inicializar I2C para MLX90614 (Raspberry Pi)
try:
    i2c = busio.I2C(board.SCL, board.SDA)
    mlx = MLX90614(i2c)
    print("☑ MLX90614 detectado")
except Exception as e:
    print(f"❌ Error al conectar MLX90614: {e}")
    messagebox.showerror("Error", f"No se pudo conectar al sensor MLX90614: {e}")
    exit()

# Conectar Arduino con Firmata
try:
    arduino = Arduino(PORT_ARDUINO)
    it = util.Iterator(arduino)
    it.start()
    print("☑ Arduino conectado con Firmata")
except Exception as e:
    print(f"❌ Error al conectar Arduino: {e}")
    messagebox.showerror("Error", f"No se pudo conectar al Arduino: {e}")
    exit()

# Pines analógicos
pulse_pin = arduino.analog[0]  # PulseSensor → A0
lm35_pin = arduino.analog[1]   # LM35 → A1
```

```python
for pin in [pulse_pin, lm35_pin]:
    pin.enable_reporting()

# Carpetas
os.makedirs("datos", exist_ok=True)
os.makedirs("graficos", exist_ok=True)

# Variables globales
bpm_actual = 0.0
ultima_lectura_pulse = 0.0
tiempo_ultimo_pulso = None
historial_bpm = []
max_points = 100
datos_pulse = []
csv_file = None
csv_writer = None
start_time = None
ultima_vez_thingspeak = time.time()
INTERVALO_TS = 15


# --- DETECCIÓN DE PULSO CON DEPURACIÓN ---
def detectar_pulso(lectura, umbral=1.4):  # ☑ Umbral ajustado a tus datos (~1.1–2.2 V)
    global ultima_lectura_pulse, tiempo_ultimo_pulso, bpm_actual
    if lectura is None:
        return bpm_actual

    # Depuración: ver cada lectura
    print(f"[PULSO] L={lectura:.3f}V | U={umbral} | Ant={ultima_lectura_pulse:.3f}V")

    if lectura > umbral and ultima_lectura_pulse <= umbral:
        ahora = time.time()
        if tiempo_ultimo_pulso:
            delta = ahora - tiempo_ultimo_pulso
            if 0.3 < delta < 2:  # Rango válido: 30–200 BPM
                bpm = 60 / delta
                bpm_actual = round(bpm, 1)
                print(f"☑ PULSO DETECTADO → Delta={delta:.2f}s → BPM={bpm_actual}")
                historial_bpm.append(bpm_actual)
                if len(historial_bpm) > 10:
                    historial_bpm.pop(0)
        tiempo_ultimo_pulso = ahora
    ultima_lectura_pulse = lectura
    return bpm_actual


def estado_arritmia():
    if len(historial_bpm) < 5:
        return "Estable"
    desviacion = sum(abs(a - b) for a, b in zip(historial_bpm, historial_bpm[1:])) / max(len(historial_bpm)-1, 1)
    return "Alta variabilidad" if desviacion > 10 else "Estable"
```

```
# --- INTERFAZ GRÁFICA ---
root = tk.Tk()
root.title("Monitoreo de Signos Vitales - 3 Sensores")
root.geometry("1200x900")
root.configure(bg="#f0f0f0")

frame_top = Frame(root, bg="#f0f0f0")
frame_top.pack(pady=10)

Label(frame_top, text="MONITOREO DE SIGNOS VITALES",
font=("Helvetica", 18, "bold"), bg="#f0f0f0").grid(row=0, column=0,
columnspan=4, pady=5)

label_fecha = Label(frame_top, text="Estado: Esperando...",
font=("Helvetica", 12), bg="#f0f0f0", fg="gray")
label_fecha.grid(row=1, column=0, padx=10)

label_bpm = Label(frame_top, text="BPM: --", font=("Helvetica", 14,
"bold"), bg="#f0f0f0")
label_bpm.grid(row=1, column=1, padx=10)

label_estado = Label(frame_top, text="Estado: --", font=("Helvetica",
12), bg="#f0f0f0", fg="blue")
label_estado.grid(row=1, column=2, padx=10)

label_lm35 = Label(frame_top, text="LM35: -- °C", font=("Helvetica",
12), bg="#f0f0f0")
label_lm35.grid(row=2, column=0, padx=10)

label_mlx = Label(frame_top, text="MLX90614: -- °C",
font=("Helvetica", 12), bg="#f0f0f0")
label_mlx.grid(row=2, column=1, padx=10)


# --- BOTONES ---
def iniciar():
    global csv_file, csv_writer, start_time
    if csv_file:
        detener()
    start_time = datetime.now()
    filename =
f"datos/registro_{start_time.strftime('%Y%m%d_%H%M%S')}.csv"
    try:
        csv_file = open(filename, 'w', newline='', encoding='utf-8')
        csv_writer = csv.writer(csv_file)
        csv_writer.writerow(['FechaHora', 'BPM', 'Pulse_Volt',
'Temp_LM35', 'Temp_MLX90614'])
        label_fecha.config(text=f"Grabando:
{start_time.strftime('%d/%m %H:%M:%S')}")
        print(f"☑ Grabación iniciada: {filename}")
    except Exception as e:
        print(f"❌ Error CSV: {e}")


def detener():
```

```
    global csv_file
    if csv_file:
        csv_file.close()
        csv_file = None
        label_fecha.config(text="Grabación detenida")


def guardar_grafico():
    if datos_pulse:
        nombre =
f"graficos/grafico_{datetime.now().strftime('%Y%m%d_%H%M%S')}.png"
        fig.savefig(nombre, dpi=150, bbox_inches='tight')
        print(f"☑ Gráfico guardado: {nombre}")
        messagebox.showinfo("Éxito", f"Gráfico guardado
como:\n{nombre}")


def salir():
    detener()
    arduino.exit()
    root.quit()


frame_botones = Frame(root, bg="#f0f0f0")
frame_botones.pack(pady=10)

Button(frame_botones, text="Iniciar", command=iniciar, bg="green",
fg="white", width=10).grid(row=0, column=0, padx=5)
Button(frame_botones, text="Detener", command=detener, bg="orange",
fg="white", width=10).grid(row=0, column=1, padx=5)
Button(frame_botones, text="Guardar Gráfico", command=guardar_grafico,
bg="blue", fg="white", width=15).grid(row=0, column=2, padx=5)
Button(frame_botones, text="Salir", command=salir, bg="red",
fg="white", width=10).grid(row=0, column=3, padx=5)


# --- GRÁFICOS ---
fig, ax = plt.subplots(figsize=(10, 5))
line_pulse, = ax.plot([], [], 'r-', linewidth=1.2, label="Pulso")
ax.set_ylim(0, 5)
ax.set_xlim(0, max_points)
ax.grid(True, alpha=0.3)
ax.set_title("Señal del PulseSensor", fontsize=12)
ax.set_xlabel("Muestras")
ax.legend()

canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)


# --- ANIMACIÓN PRINCIPAL ---
def actualizar(frame):
    global datos_pulse, bpm_actual, ultima_vez_thingspeak

    # Leer sensores vía Firmata
    try:
```

```
            pulse_raw = pulse_pin.read()
            temp_lm35_raw = lm35_pin.read()
        except Exception as e:
            print(f"❌ Lectura fallida: {e}")
            return line_pulse,

        if pulse_raw is None or temp_lm35_raw is None:
            return line_pulse,

        # Convertir a voltaje
        pulse_volt = pulse_raw * 5.0
        temp_lm35 = temp_lm35_raw * 5.0 * 100  # 10 mV/°C → °C

        # Calcular BPM
        bpm_actual = detectar_pulso(pulse_volt, umbral=1.4)

        # Leer MLX90614
        try:
            temp_mlx = mlx.object_temperature
        except Exception as e:
            print(f"❌ MLX error: {e}")
            temp_mlx = None

        # --- ACTUALIZAR ETIQUETAS ---
        label_bpm.config(
            text=f"BPM: {bpm_actual:.1f}",
            fg="red" if bpm_actual < 50 or bpm_actual > 100 else "black"
        )
        label_lm35.config(
            text=f"LM35: {temp_lm35:.1f} °C",
            fg="red" if temp_lm35 < 35 or temp_lm35 > 40 else "black"
        )
        label_mlx.config(
            text=f"MLX90614: {temp_mlx:.1f} °C" if temp_mlx else
"MLX90614: ERROR",
            fg="red" if temp_mlx and (temp_mlx < 35 or temp_mlx > 40) else
"black"
        )
        label_estado.config(
            text=f"Estado: {estado_arritmia()}",
            fg="red" if "variabilidad" in estado_arritmia() else "blue"
        )

        # --- ALMACENAR PARA GRÁFICO ---
        datos_pulse.append(pulse_volt)
        if len(datos_pulse) > max_points:
            datos_pulse.pop(0)
        line_pulse.set_data(range(len(datos_pulse)), datos_pulse)

        # Escala dinámica
        if datos_pulse:
            min_y = min(datos_pulse) - 0.5
            max_y = max(datos_pulse) + 0.5
            ax.set_ylim(min_y, max_y)

        canvas.draw()
```

```
    # --- GUARDAR EN CSV ---
    if csv_writer:
        ahora_str = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        fila = [
            ahora_str,
            round(bpm_actual, 1),
            round(pulse_volt, 3),
            round(temp_lm35, 1),
            round(temp_mlx, 1) if temp_mlx else 0
        ]
        csv_writer.writerow(fila)
        csv_file.flush()

    # --- ENVIAR A THINGSPEAK CADA 15 SEGUNDOS ---
    ahora = time.time()
    if ahora - ultima_vez_thingspeak >= INTERVALO_TS:
        # ☑ Mostrar en consola lo que se envía
        print(f"[TS] Enviando → BPM:{bpm_actual:.1f},
Pulse:{pulse_volt:.3f}V, LM35:{temp_lm35:.1f}°C,
MLX:{temp_mlx:.1f}°C")

        try:
            params = {
                'api_key': API_KEY_THINGSPEAK,
                'field1': round(bpm_actual, 1),
                'field2': round(pulse_volt, 3),
                'field3': round(temp_lm35, 1),
                'field4': round(temp_mlx, 1) if temp_mlx else 0
            }
            response = requests.get(THINGSPEAK_URL, params=params,
timeout=5)
            print(f"[TS] Respuesta HTTP {response.status_code}:
{response.text}")
            if response.status_code == 200:
                ultima_vez_thingspeak = ahora
        except Exception as e:
            print(f"[TS] ERROR: {e}")

    return line_pulse,


# --- INICIAR ANIMACIÓN Y GUI ---
ani = animation.FuncAnimation(fig, actualizar, interval=200,
blit=False, cache_frame_data=False)
root.mainloop()
```

2do Caso. Con la librería Serial en Python

```
# monitoreo_serial.py
# Sistema sin Firmata — solo serial + MLX90614 en RPi
# Autor: Juan Carlos Vela - UPSE

import time
import serial
import csv
import os
import tkinter as tk
from tkinter import Label, Button, Frame, messagebox
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.animation as animation
import board
import busio
from adafruit_mlx90614 import MLX90614
import requests
import threading

# --- CONFIGURACIÓN ---
SERIAL_PORT = '/dev/ttyACM0'
BAUD_RATE = 9600
API_KEY_THINGSPEAK = '8FQPQNAL5O0J23E3'
THINGSPEAK_URL = 'https://api.thingspeak.com/update'

# Inicializar MLX90614 (Raspberry Pi)
try:
    i2c = busio.I2C(board.SCL, board.SDA)
    mlx = MLX90614(i2c)
    print("☑ MLX90614 listo")
except Exception as e:
    print(f"❌ MLX90614: {e}")
    messagebox.showerror("Error", f"No se conectó MLX90614: {e}")
    exit()

# Conectar Arduino por serial
try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=0.1)
    time.sleep(2)
    print("☑ Arduino conectado por serial")
except Exception as e:
    print(f"❌ Serial: {e}")
    messagebox.showerror("Error", f"No se conectó Arduino: {e}")
    exit()

# Carpetas
os.makedirs("datos", exist_ok=True)
os.makedirs("graficos", exist_ok=True)

# Variables globales
bpm_actual = 0.0
ultima_lectura_pulse = 0.0
tiempo_ultimo_pulso = None
```

```
historial_bpm = []
max_points = 100
datos_pulse = []
csv_file = None
csv_writer = None
start_time = None
ultima_vez_ts = time.time()
INTERVALO_TS = 15


# --- DETECCIÓN DE PULSO (ajustada a tu señal) ---
def detectar_pulso(lectura, umbral=1.4):
    global ultima_lectura_pulse, tiempo_ultimo_pulso, bpm_actual
    if lectura is None or lectura <= 0:
        return bpm_actual

    print(f"[PULSO] L={lectura:.3f}V | U={umbral} |
Ant={ultima_lectura_pulse:.3f}V")

    if lectura > umbral and ultima_lectura_pulse <= umbral:
        ahora = time.time()
        if tiempo_ultimo_pulso:
            delta = ahora - tiempo_ultimo_pulso
            if 0.3 < delta < 2:
                bpm = 60 / delta
                bpm_actual = round(bpm, 1)
                print(f"☑ Pulso detectado! Delta={delta:.2f}s →
BPM={bpm_actual}")
                historial_bpm.append(bpm_actual)
                if len(historial_bpm) > 10:
                    historial_bpm.pop(0)
        tiempo_ultimo_pulso = ahora
    ultima_lectura_pulse = lectura
    return bpm_actual


def estado_arritmia():
    if len(historial_bpm) < 5:
        return "Estable"
    desviacion = sum(abs(a - b) for a, b in zip(historial_bpm,
historial_bpm[1:])) / max(len(historial_bpm)-1, 1)
    return "Alta variabilidad" if desviacion > 10 else "Estable"


# --- INTERFAZ GRÁFICA ---
root = tk.Tk()
root.title("Monitoreo 3 Sensores - Modo Serial")
root.geometry("1200x900")
root.configure(bg="#f0f0f0")

frame_top = Frame(root, bg="#f0f0f0")
frame_top.pack(pady=10)

Label(frame_top, text="MONITOREO DE SIGNOS VITALES",
font=("Helvetica", 18, "bold"), bg="#f0f0f0").grid(row=0, column=0,
columnspan=4, pady=5)
```

```
label_fecha = Label(frame_top, text="Estado: Esperando...",
font=("Helvetica", 12), bg="#f0f0f0", fg="gray")
label_fecha.grid(row=1, column=0, padx=10)

label_bpm = Label(frame_top, text="BPM: --", font=("Helvetica", 14,
"bold"), bg="#f0f0f0")
label_bpm.grid(row=1, column=1, padx=10)

label_estado = Label(frame_top, text="Estado: --", font=("Helvetica",
12), bg="#f0f0f0", fg="blue")
label_estado.grid(row=1, column=2, padx=10)

label_lm35 = Label(frame_top, text="LM35: -- °C", font=("Helvetica",
12), bg="#f0f0f0")
label_lm35.grid(row=2, column=0, padx=10)

label_mlx = Label(frame_top, text="MLX90614: -- °C",
font=("Helvetica", 12), bg="#f0f0f0")
label_mlx.grid(row=2, column=1, padx=10)


# --- BOTONES ---
def iniciar():
    global csv_file, csv_writer, start_time
    if csv_file:
        detener()
    start_time = datetime.now()
    filename =
f"datos/registro_{start_time.strftime('%Y%m%d_%H%M%S')}.csv"
    try:
        csv_file = open(filename, 'w', newline='', encoding='utf-8')
        csv_writer = csv.writer(csv_file)
        csv_writer.writerow(['FechaHora', 'BPM', 'Pulse_Volt',
'Temp_LM35', 'Temp_MLX90614'])
        label_fecha.config(text=f"Grabando:
{start_time.strftime('%d/%m %H:%M:%S')}")
        print(f"☑ Grabación iniciada: {filename}")
    except Exception as e:
        print(f"❌ CSV error: {e}")


def detener():
    global csv_file
    if csv_file:
        csv_file.close()
        csv_file = None
        label_fecha.config(text="Grabación detenida")


def guardar_grafico():
    if datos_pulse:
        nombre =
f"graficos/grafico_{datetime.now().strftime('%Y%m%d_%H%M%S')}.png"
        fig.savefig(nombre, dpi=150, bbox_inches='tight')
        print(f"☑ Gráfico guardado: {nombre}")
```

```
        messagebox.showinfo("Éxito", f"Guardado: {nombre}")


def salir():
    detener()
    root.quit()


frame_botones = Frame(root, bg="#f0f0f0")
frame_botones.pack(pady=10)
Button(frame_botones, text="Iniciar", command=iniciar, bg="green",
fg="white", width=10).grid(row=0, column=0, padx=5)
Button(frame_botones, text="Detener", command=detener, bg="orange",
fg="white", width=10).grid(row=0, column=1, padx=5)
Button(frame_botones, text="Guardar Gráfico", command=guardar_grafico,
bg="blue", fg="white", width=15).grid(row=0, column=2, padx=5)
Button(frame_botones, text="Salir", command=salir, bg="red",
fg="white", width=10).grid(row=0, column=3, padx=5)


# --- GRÁFICOS ---
fig, ax = plt.subplots(figsize=(10, 5))
line_pulse, = ax.plot([], [], 'r-', linewidth=1.2, label="Pulso")
ax.set_ylim(0, 5)
ax.set_xlim(0, max_points)
ax.grid(True, alpha=0.3)
ax.set_title("Señal del PulseSensor", fontsize=12)
ax.set_xlabel("Muestras")
ax.legend()

canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)


# --- ACTUALIZACIÓN NO BLOQUEANTE (sin FuncAnimation) ---
def actualizar():
    global datos_pulse, bpm_actual, ultima_vez_ts

    # Leer todo lo disponible del Arduino
    if ser.in_waiting > 0:
        raw = ser.read(ser.in_waiting)
        try:
            linea = raw.decode('utf-8', errors='ignore').replace('\r',
'').replace('\n', '').strip()
            print(f"[SERIAL] '{linea}'")  # 👈 Depuración en consola

            if 'PULSE:' in linea and 'LM35:' in linea:
                partes = linea.split(',')
                if len(partes) >= 2:
                    try:
                        pulse_val = float(partes[0].split(':')[1])
                        temp_lm35 = float(partes[1].split(':')[1])

                        # Calcular BPM
                        bpm_actual = detectar_pulso(pulse_val,
umbral=1.4)
```

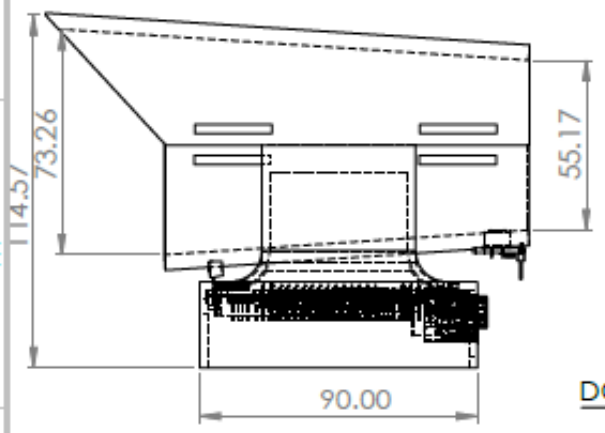
```
                        # Leer MLX90614
                        try:
                            temp_mlx = mlx.object_temperature
                        except:
                            temp_mlx = None

                        # Actualizar interfaz
                        label_bpm.config(
                            text=f"BPM: {bpm_actual:.1f}",
                            fg="red" if bpm_actual < 50 or bpm_actual
> 100 else "black"
                        )
                        label_lm35.config(
                            text=f"LM35: {temp_lm35:.1f} °C",
                            fg="red" if temp_lm35 < 35 or temp_lm35 >
40 else "black"
                        )
                        label_mlx.config(
                            text=f"MLX90614: {temp_mlx:.1f} °C" if
temp_mlx else "MLX90614: ERROR",
                            fg="red" if temp_mlx and (temp_mlx < 35 or
temp_mlx > 40) else "black"
                        )
                        label_estado.config(
                            text=f"Estado: {estado_arritmia()}",
                            fg="red" if "variabilidad" in
estado_arritmia() else "blue"
                        )

                        # Almacenar para gráfico
                        datos_pulse.append(pulse_val)
                        if len(datos_pulse) > max_points:
                            datos_pulse.pop(0)
                        line_pulse.set_data(range(len(datos_pulse)),
datos_pulse)

                        # Escala dinámica
                        if datos_pulse:
                            min_y = min(datos_pulse) - 0.5
                            max_y = max(datos_pulse) + 0.5
                            ax.set_ylim(min_y, max_y)

                        canvas.draw()

                        # Guardar CSV
                        if csv_writer:
                            ahora_str = datetime.now().strftime('%Y-
%m-%d %H:%M:%S')
                            fila = [
                                ahora_str,
                                round(bpm_actual, 1),
                                round(pulse_val, 3),
                                round(temp_lm35, 1),
                                round(temp_mlx, 1) if temp_mlx else 0
                            ]
```

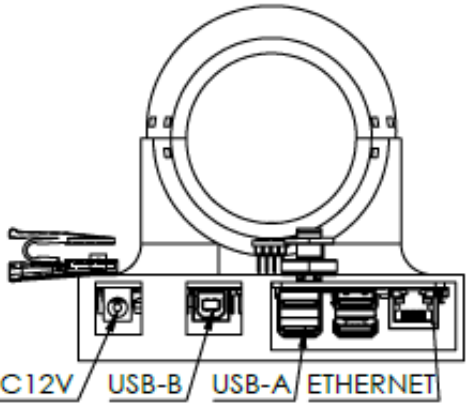
```
                            csv_writer.writerow(fila)
                            csv_file.flush()

                        # Enviar a ThingSpeak (cada 15 s)
                        ahora = time.time()
                        if ahora - ultima_vez_ts >= INTERVALO_TS:
                            print(f"[TS] Enviando: BPM={bpm_actual},
Pulse={pulse_val:.3f}, LM35={temp_lm35:.1f}, MLX={temp_mlx:.1f}")
                            try:
                                params = {
                                    'api_key': API_KEY_THINGSPEAK,
                                    'field1': round(bpm_actual, 1),
                                    'field2': round(pulse_val, 3),
                                    'field3': round(temp_lm35, 1),
                                    'field4': round(temp_mlx, 1) if
temp_mlx else 0
                                }
                                response =
requests.get(THINGSPEAK_URL, params=params, timeout=5)
                                print(f"[TS] Respuesta:
{response.status_code}")
                                if response.status_code == 200:
                                    ultima_vez_ts = ahora
                            except Exception as e:
                                print(f"[TS] Error: {e}")

                    except Exception as e:
                        print(f"❌ Parseo falló: {e} | Línea:
{linea}")

        except Exception as e:
            print(f"❌ Decode falló: {e} | Raw: {repr(raw)}")

    # Llamar nuevamente después de 50 ms → fluidez
    root.after(50, actualizar)


# --- INICIAR ---
root.after(100, actualizar)
root.mainloop()
```

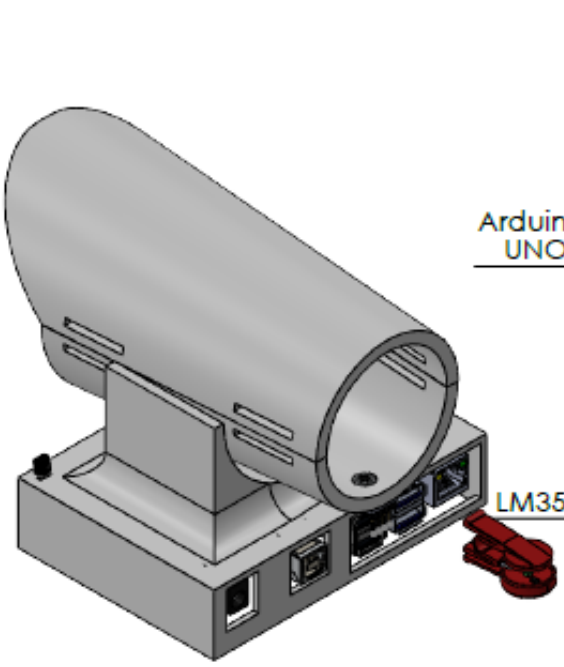
Programación de Sketch de Arduino

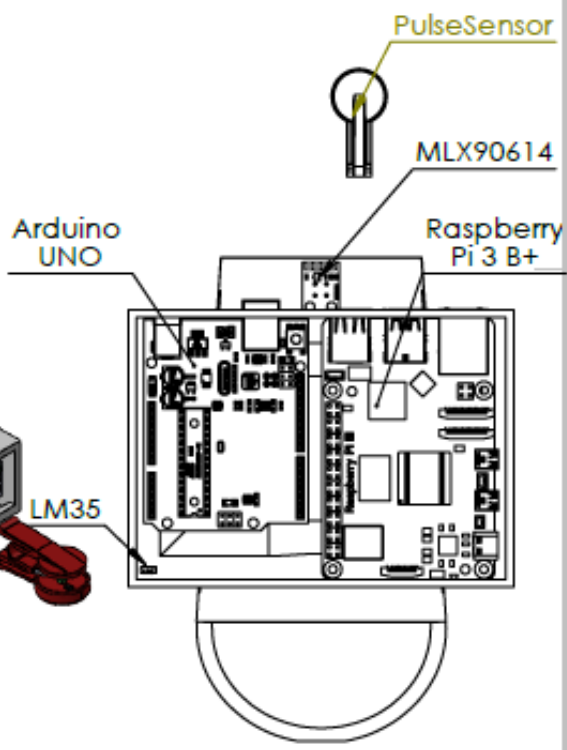
```
// sketch_arduino.ino
// Envía: PULSE:<v>,LM35:<t>
// Frecuencia: ~5 Hz (200 ms)
void setup() {
  Serial.begin(9600);
  while (!Serial) {} //
}

void loop() {
  int adc_pulse = analogRead(A0);  // PulseSensor → A0
  int adc_lm35 = analogRead(A1);   // LM35 → A1

  // Convertir a voltaje
  float pulse_volt = (adc_pulse * 5.0) / 1023.0;
  float temp_lm35 = (adc_lm35 * 5.0) / 1023.0 * 100; // °C

  // Enviar en formato robusto
  Serial.print("PULSE:");
  Serial.print(pulse_volt, 3);
  Serial.print(",LM35:");
  Serial.println(temp_lm35, 1);

  delay(200); // ~5 muestras/segundo
}
```

Anexo 2: PLANOS CAD

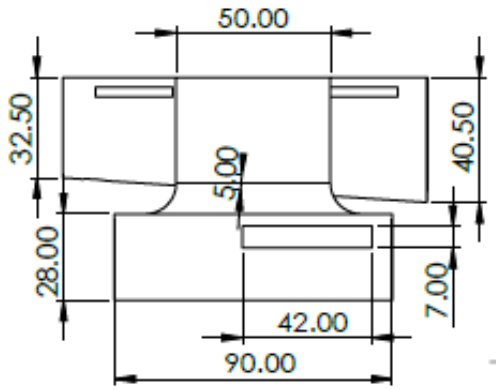


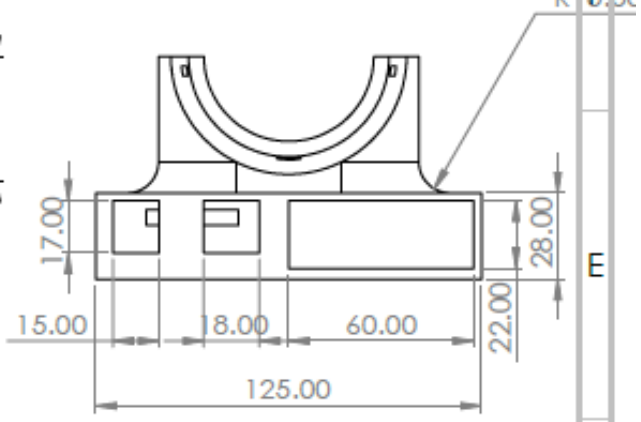


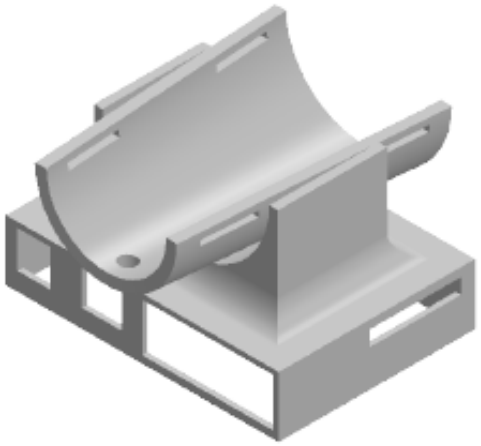


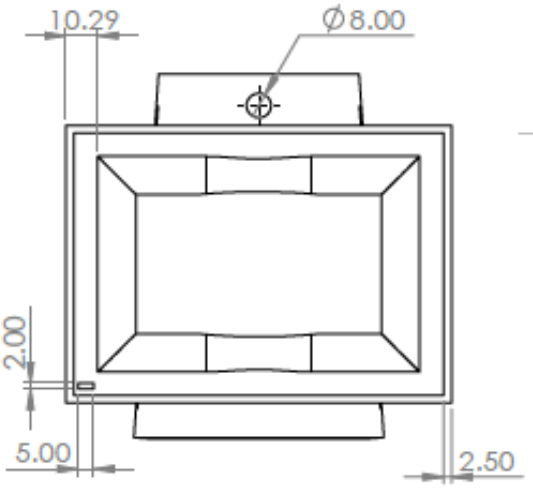


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
						UPSE	
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	Juan Vela					Sistema de Signos Vitales, Prototipo	
VERIF.	Oscar Gomez						
APROB.	Oscar Gomez						
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
				Varios		AB-ENS-001	
				PESO:		ESCALA 1:2	HOJA 1 DE 1

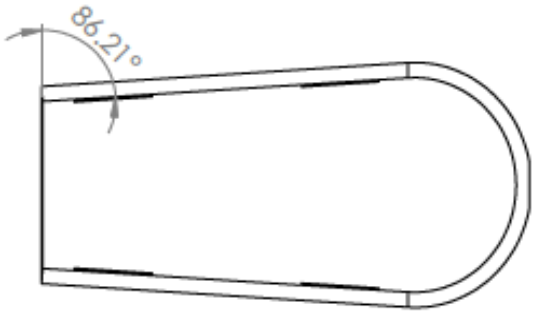


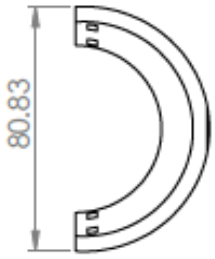


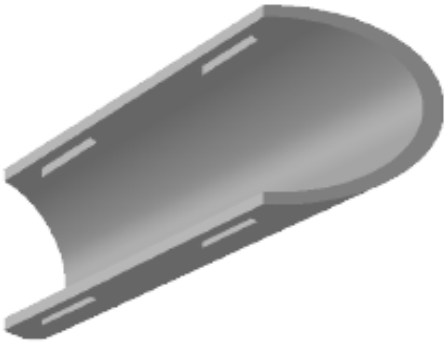


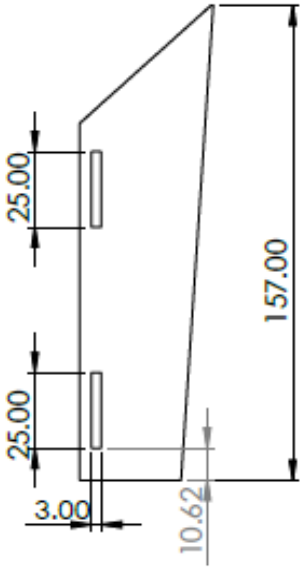


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN	
				UPSE		
	NOMBRE	FIRMA	FECHA	TÍTULO: Antebrazo Inferior		
DIBUJ.	JUAN VELA					
VERIF.	OSCAR GOMEZ					
APROB.	OSCAR GOMEZ					
FABR.						
CALID.				MATERIAL: PLA	N.º DE DIBUJO AB-PAR-002	A4
				PESO:	ESCALA:2	HOJA 1 DE 1

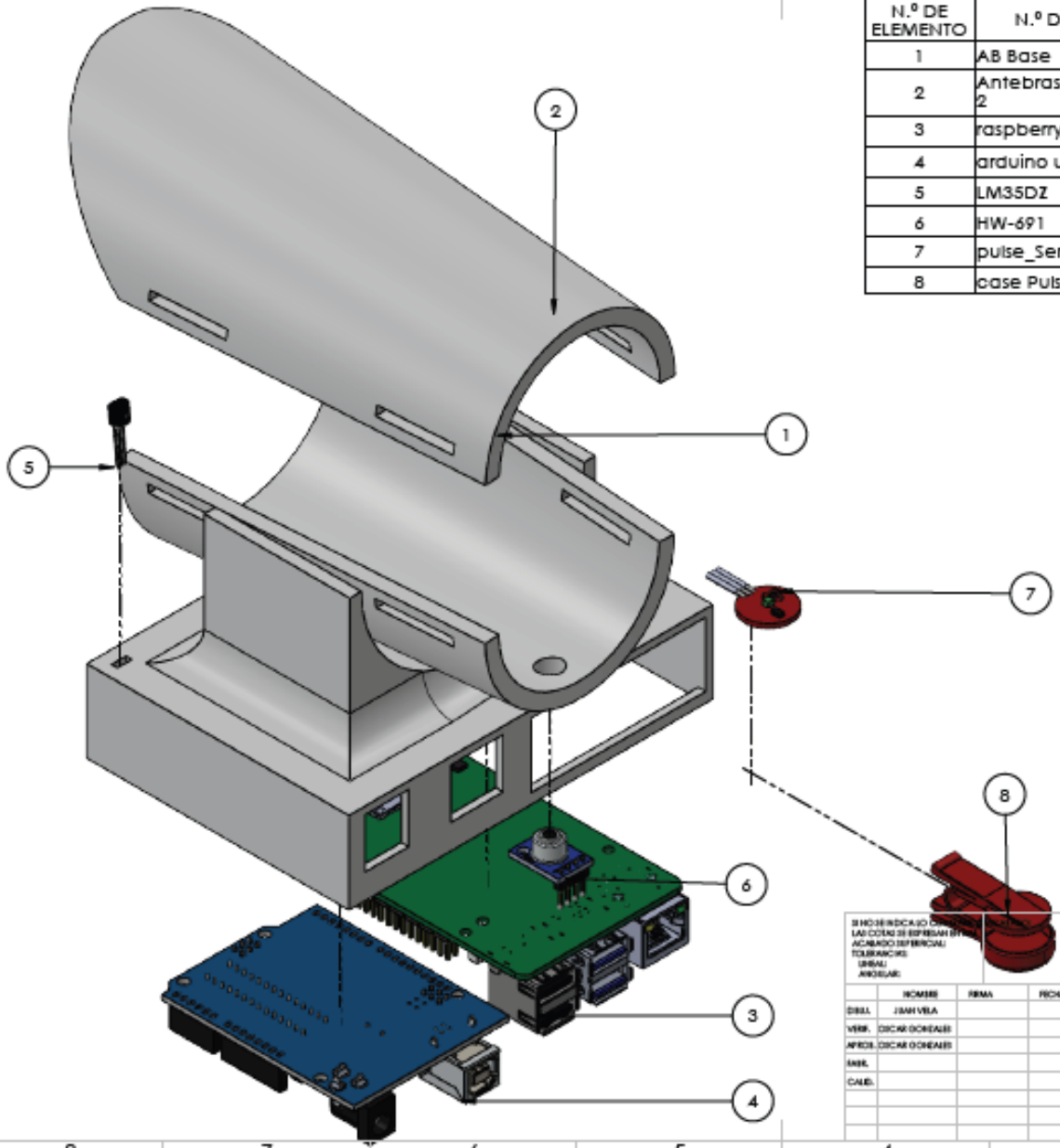








SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
						UPSE	
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	JUAN VELA					Antebrazo Superior	
VERIF.	OSCAR GOMEZ						
APROB.	OSCAR GOMEZ						
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO AB-PAR-001	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1



N.º DE ELEMENTO	N.º DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	AB Base		1
2	Antebraso Superior 2		1
3	raspberry_pi_5		1
4	arduino uno		1
5	LM35DZ		1
6	HW-691		1
7	pulse_Sensor		1
8	case PulseSensor		1

	NOMBRE	FIRMA	FECHA
DIBUJ.	JUAN VELA		
VERIF.	OSCAR GONZALES		
APROB.	OSCAR GONZALES		
MAND.			
CALID.			

MATERIAL: VARIOS

PESO:

ESCALA: 1:1

HOJA 1 DE 1

ELIMINAR Y ROMPER ARISTAS VIVAS

NO CAMBIE LA ESCALA

REVISIÓN

UPSE

TÍTULO:

EXTRUCCIÓN

N.º DE DIBUJO

AB-EXTR-001

A3