

الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

جامعة سعد دحلب البليدة
University SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculty of Technology

قسم الإلكترونيك
Department of Electronics



Final Year Project Report

Domain: Science and Technology

Sector: Electronics

Specialty: Electronics of embedded systems

presented by

Djouabi Karim

Theme

IoT-Based Real-Time Health Monitoring System

Supervised by: Mme. Bougherira

Année Universitaire 2023-2024

Remerciements

In the name of Allah, the Most Gracious and the Most Merciful, we thank Allah for all His blessing and strength that He has given us for carry out this modest project. we would like to to thanks all my teachers, especially my Promotor Mme Bougherira. They helped me understand so much we didn't know before, especially during write this thesis. And we express my sincere gratitude to my parents, family, and friends who helped and support me during development of this project, and we would thanks them for their valuable support.

ملخص:

يستفيد المشروع من إنترنت ESP32 تستكشف الأطروحة تطوير نظام مراقبة الصحة المدمج باستخدام متحكم لجمع البيانات الصحية في الوقت الفعلي من أجهزة الاستشعار ونقلها بشكل آمن إلى منصة (IoT) الأشياء التخزين السحابية. يتيح هذا الإعداد لمتخصصي الرعاية الصحية والمرضى الحصول على رؤى قيمة حول الحالة الصحية، وتسهيل الرعاية الاستباقية وتحسين نتائج الرعاية الصحية.

Résumé :

La thèse explore le développement d'un système embarqué de surveillance de la santé utilisant un microcontrôleur ESP32. Le projet exploite l'Internet des objets (IoT) pour collecter des données de santé en temps réel provenant de capteurs et les transmettre en toute sécurité à une plateforme de stockage cloud. Cette configuration permet aux professionnels de la santé et aux patients d'obtenir des informations précieuses sur l'état de santé, facilitant ainsi les soins proactifs et l'amélioration des résultats des soins de santé.

Abstract :

The thesis explores the development of an embedded health monitoring system using an ESP32 microcontroller. The project leverages the Internet of Things (IoT) to collect real-time health data from sensors and securely transmit it to a cloud storage platform. This setup allows healthcare professionals and patients to gain valuable insights into health status, facilitating proactive care and improved healthcare outcomes.

Lists of acronyms and abbreviations

- **RAM:** Random Access Memory
- **REST:** Representational State Transfer
- **RF:** Radio Frequency
- **RFID:** Radio Frequency Identification
- **SD:** Secure Digital
- **SDHC:** Secure Digital High Capacity
- **SGBD:** Systèmes de Gestion de Base de Données
- **SPI:** Serial Peripheral Interface
- **SQL:** Structured Query Language
- **SRAM:** Static Random Access Memory
- **Sub/Pub:** Subscriber/Publisher
- **TCP:** Transmission Control Protocol
- **UART:** Universal Asynchronous Receiver-Transmitter
- **UDP:** User Datagram Protocol
- **UIT:** Union Internationale des Télécommunication
- **URI:** Uniform Resource Identifier
- **URL:** Uniform Resource Locator
- **USB:** Universal Serial Bus
- **UUID:** Universally Unique Identifier

Contents

| | |
|---|----|
| General Introduction | 1 |
| Chapitre 1 Generalist | 3 |
| Introduction..... | 3 |
| 1.1 Definition of IoT | 3 |
| 1.1.1 How IoT work..... | 4 |
| 1.1.2 Application of IoT..... | 4 |
| 1.2 Cloud and Server Infrastructure | 6 |
| 1.2.1 Databases | 6 |
| 1.2.2 Cloud Services | 8 |
| 1.2.3 Security Measures | 8 |
| 1.3 Health Monitoring Measures..... | 9 |
| 1.4 Benefits of IoT in Health monitoring..... | 11 |
| 1.5 Components selected for this project | 12 |
| 1.5.1 ESP32-DevkitC-V4..... | 12 |
| 1.5.2 AD8232 (ECG Sensor) | 15 |
| 1.5.3 MAX30100 (Oximeter and Heart Rate Sensor)..... | 16 |
| 1.6 Software and Platforms..... | 17 |
| 1.6.1 Arduino IDE..... | 17 |
| 1.6.2 InfluxDB | 21 |
| 1.6.3 Grafana..... | 22 |
| Conclusion: | 23 |
| Chapitre 2 Conception | 24 |
| 2.1 System Architecture..... | 24 |
| 2.2 Hardware Design..... | 25 |
| 2.2.1 Microcontroller Layer | 25 |
| 2.2.2 Sensors Layer | 28 |
| 2.2.3 The circuit | 31 |
| 2.3 Software | 32 |
| 2.3.1 Arduino IDE | 32 |
| 2.3.2 Setup InfluxDB..... | 36 |
| 2.3.3 Setup NSSM..... | 40 |
| 2.3.4 Setup Grafana..... | 42 |
| Conclusion | 46 |
| Chapitre 3 Implementation and Result | 47 |

| | |
|--|----|
| Introduction..... | 47 |
| 3.1 Test and connect components | 47 |
| 3.1.1 Connect the MCU to Database..... | 47 |
| 3.1.2 Connect InfluxDB Grafana | 52 |
| 3.1.3 Test the sensors | 57 |
| 3.2 Result and interface..... | 61 |
| General Conclusion..... | 64 |
| References..... | 65 |

Figure List

| | |
|---|----|
| Figure 1 IoT system (imaginary drawing) | 3 |
| Figure 2 IoT system work | 4 |
| Figure 3 Example of a relational database | 6 |
| Figure 4 the different between SQL and non SQL Databases | 7 |
| Figure 5 TSDB example..... | 7 |
| Figure 6 ECG Signal..... | 9 |
| Figure 7 Pulse_oximetry..... | 10 |
| Figure 8 Average heart rates by age | 10 |
| Figure 9 Blood Pressure Table..... | 11 |
| Figure 10 ESP32-DevKitC V4 with ESP32-WROOM-32 soldered module..... | 12 |
| Figure 11 ESP32-WROOM-32D | 13 |
| Figure 12 The Pin diagram of the ESP32-devkitc-v4 | 15 |
| Figure 13 AD8232 Sensor | 15 |
| Figure 14 MAX30100..... | 16 |
| Figure 15 Arduino IDE interface | 18 |
| Figure 16 Arduino IDE Menus | 18 |
| Figure 17 Arduino IDE Tools-Bar | 19 |
| Figure 18 Arduino IDE – Code section..... | 20 |
| Figure 19 Arduino IDE – Statue section | 20 |
| Figure 20 Arduino IDE – Program notifications..... | 20 |
| Figure 21 InfluxDB Architecture | 21 |
| Figure 22 Grafana Dashboard | 22 |
| Figure 23 AD8232 Schematic | 29 |
| Figure 24 the functional block for the MAX30100 module | 30 |
| Figure 25 Scheamtic of circuit..... | 31 |
| Figure 26 ARduino IDE Setup 1 | 33 |
| Figure 27 ARduino IDE Setup 2..... | 33 |
| Figure 28 ARduino IDE Setup 3..... | 34 |
| Figure 29 ARduino IDE Setup 4..... | 34 |
| Figure 30 ARduino IDE Setup 5..... | 35 |
| Figure 31 ARduino IDE Setup 6..... | 35 |
| Figure 32 ARduino IDE Setup 7..... | 36 |
| Figure 33 InfluxDB Setup 1..... | 37 |
| Figure 34 INFLUXDB SETUP 2..... | 37 |
| Figure 35 INFLUXDB SETUP 3..... | 38 |
| Figure 36 INFLUXDB SETUP 4..... | 38 |
| Figure 37 INFLUXDB SETUP 5..... | 39 |
| Figure 38 INFLUXDB SETUP 6..... | 39 |
| Figure 39 NSSM Setup 1 | 40 |
| Figure 40 NSSM SETUP 2 | 40 |
| Figure 41 NSSM SETUP 3 | 41 |
| Figure 42 NSSM SETUP 4 | 41 |
| Figure 43 NSSM SETUP 5 | 42 |
| Figure 44 Grafana Setup 1 | 43 |

| | |
|--|----|
| Figure 45 GRAFANA SETUP 2..... | 43 |
| Figure 46 GRAFANA SETUP 3..... | 44 |
| Figure 47 GRAFANA SETUP 4..... | 44 |
| Figure 48 GRAFANA SETUP 5..... | 45 |
| Figure 49 GRAFANA SETUP 6..... | 45 |
| Figure 50 InfluxDB Configuration 1..... | 48 |
| Figure 51 INFLUXDB CONFIGURATION 2 | 48 |
| Figure 52 Synchronized to InfluxDB | 49 |
| Figure 53 Create Dashboard in InfluxDB | 50 |
| Figure 54 ADD CELL in InfluxDB | 51 |
| Figure 55 Read wifi_state in InfluxDB | 51 |
| Figure 56 INFLUXDB CONFIGURATION 3 | 51 |
| Figure 57 ADD Data sources in Grafana | 52 |
| Figure 58 Select Database in Grafana..... | 53 |
| Figure 59 Grafana Configuration 1 | 53 |
| Figure 60 GRAFANA CONFIGURATION 2 | 54 |
| Figure 61 ADD Dashboard in Grafana | 55 |
| Figure 62 GRAFANA CONFIGURATION 3 | 55 |
| Figure 63 Copy Script from InfluxDB 1 | 55 |
| Figure 64 COPY SCRIPT FROM INFLUXDB 2..... | 56 |
| Figure 65 Past Script in Grafana | 56 |
| Figure 66 Grafana visualisation | 57 |
| Figure 67 ECG Signal output..... | 58 |
| Figure 68 Heart rate and SpO2 Result | 61 |
| Figure 69 ESP32 conected to AD8232..... | 62 |
| Figure 70 ESP32 Connected to MAX30100 | 62 |
| Figure 71 AD8232 Signal in Grafana..... | 63 |

List of Tables

| | |
|---|----|
| Table 1 ESP32 Basic buttons and their role..... | 13 |
| Table 2 ESP32 Spécification..... | 14 |

General Introduction

The ever-growing realm of healthcare is constantly seeking advancements in remote patient monitoring. This thesis explores the development of an embedded health monitor project utilizing an ESP32 microcontroller. This project leverages the power of the Internet of Things (IoT) to collect real-time health data from sensors and transmit it securely to a cloud storage platform. By harnessing the capabilities of cloud storage and data visualization tools, healthcare professionals and patients can gain valuable insights into a patient's health status, enabling proactive care and improved healthcare outcomes.

This project utilizes an ESP32 microcontroller as its core. The ESP32's processing power, Wi-Fi connectivity, and low power consumption make it ideal for developing battery-powered wearable or portable health monitoring devices. The project will integrate various health sensors with the ESP32, enabling the collection of vital signs such as heart rate, blood oxygen levels, and potentially other health parameters depending on the specific application.

The collected health data from the sensors will be processed by the ESP32 and then securely transmitted to a cloud storage platform like InfluxDB. InfluxDB is a time-series database specifically designed to handle the high volume and frequent updates characteristic of sensor data. Storing the data in the cloud allows for remote access, facilitates analysis over time, and enables the creation of insightful visualizations.

Following data storage in the cloud, the project will leverage a visualization tool like Grafana to transform the raw sensor data into user-friendly dashboards and graphs. These visualizations will present the health data in a clear and concise manner, allowing healthcare professionals to easily monitor trends, identify potential health concerns, and make informed decisions about patient care.

In my thesis, We will explore the application of IoT in health monitoring. This application involves the visualization and observation of various health parameters, such as blood pressure, blood oxygen levels, and ECG signals. Existing embedded IoT systems for health monitoring offer a variety of functionalities. Some systems focus on monitoring vital signs like heart rate, blood pressure, and blood oxygen levels such as the smartwatches with built-in health sensors and wearable devices for continuous heart rate monitoring. Other systems target specific health conditions, such as glucose monitoring devices for diabetic patients or smart inhalers for asthma management. My work will be presented in three chapters:

Chapter 1: This chapter will delve into the fundamentals of IoT, explaining how it works. we will discuss the sensors used in my project, why they were chosen, and how they function.

Additionally, we will cover the software and programming languages employed in the project, providing justifications for these choices.

Chapter 2: This chapter will outline the strategy of my work, detailing the protocols used in the project. It will be divided into two parts: software configuration and hardware components and circuits.

Chapter 3: The final chapter will focus on implementing the project, showcasing the data and results obtained.

This structure will provide a comprehensive understanding of IoT and its application in health monitoring, highlighting the significance and potential of this technology in improving healthcare outcomes.

Chapitre 1 Generalist

Introduction

In this chapter, we will explore the fundamentals of the Internet of Things (IoT). We will discuss the components we have selected for our project, explaining why each component was chosen, how they work, and their specific applications. Additionally, we will highlight the importance of IoT and its impact on various industries.

1.1 Definition of IoT

The Internet of Things (IoT) is a vast network of everyday objects transformed into "smart" devices by embedding them with sensors, processing ability, and internet connectivity. These devices collect, share, and analyze data, enabling them to automate tasks, improve efficiency, and offer entirely new functionalities. IoT systems found everywhere, from smart home appliances to industrial machinery, IoT systems are revolutionizing how we live, work, and interact with the world around us.

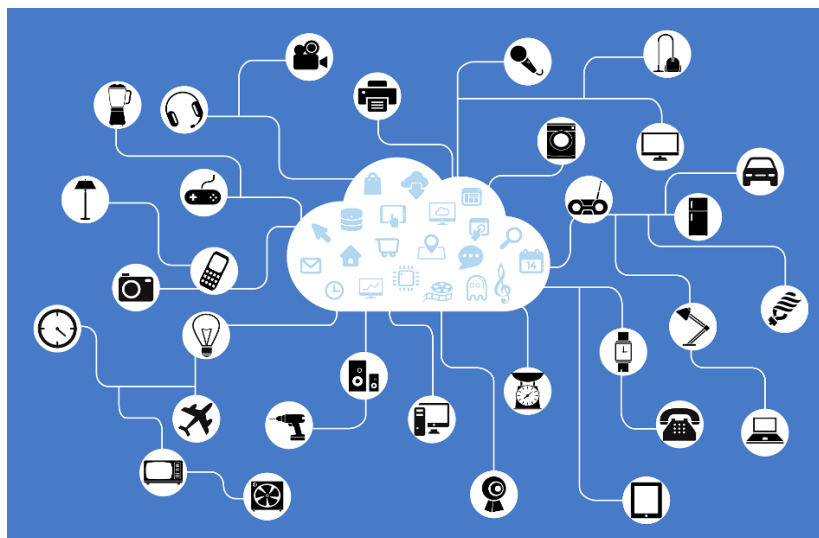


FIGURE 1 IOT SYSTEM (IMAGINARY DRAWING)

1.1.1 How IoT work

The idea of IoT is simple. Sensors collect data and send it to a processing unit within the device used. This data is then read, organized, and transmitted either wirelessly or via wired connections to the cloud. The cloud can be local or Public like Google cloud, Amazone cloud, firebase, etc (Meaning that the data can be accessed via a device connected to the same network or from anywhere in the world). The data is then integrated with software programs that analyze, organize, and display it in various formats such as tables, curves, graphs, and columns through an application or website. Smart devices such as computers, smartphones, and others are then connected to an IoT platform, allowing users to interact with the processed data.

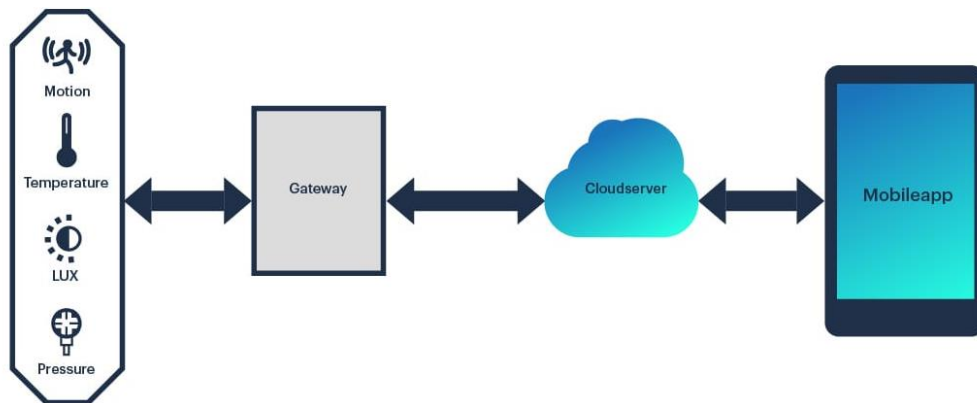


FIGURE 2 IOT SYSTEM WORK

1.1.2 Application of IoT

Here's the completed text about the Internet of Things (IoT) applications in both public and private sectors:

The Internet of Things (IoT) provides convenience to the user and ease of accessing things, reading measurements, and controlling them remotely. It also saves energy to a great extent.

The Internet of Things can be used in all fields and all sectors of public activities, such as:

- **Green mobility:**
 - Electric scooters and bikes: IoT can track availability, manage battery life, and enable remote unlocking.
 - Display of public transport schedules in real-time (Like ETUS Application): Real-time information on arrival times and delays improves passenger experience.
- **Management of traffic flows:**
 - Smart traffic lights can adjust based on real-time traffic data, optimizing flow and reducing congestion.
 - Sensors can detect accidents and automatically alert emergency services.
- **Surveillance cameras:**
 - IoT-enabled cameras can offer remote monitoring, improve response times to incidents, and deter crime.
- **Electric charging stations:**
 - Users can locate available stations, check charging status, and potentially reserve charging points remotely.

And the private activities sector in all fields like:

- **Smart homes:**
 - Control lighting, thermostats, appliances, and security systems remotely for convenience and energy savings.
 - Smart speakers and virtual assistants can automate tasks and provide hands-free control.
- **Wearable technology:**
 - Fitness trackers monitor heart rate, activity levels, and sleep patterns, promoting health and wellness.
 - Smartwatches offer notifications, contactless payments, and health monitoring.
- **Smart appliances:**
 - Ovens preheat remotely, washing machines notify when a cycle is complete, and refrigerators optimize energy usage based on contents.
- **Inventory management:**
 - Track inventory levels in real-time, automate reordering, and reduce stockouts.

- **Agriculture:**

- Sensors monitor soil moisture, temperature, and nutrient levels, allowing for precision farming and improved yields.

1.2 Cloud and Server Infrastructure

A robust cloud and server infrastructure is crucial for storing and managing the health data collected from the IoT devices in a Database can be access from the server.

1.2.1 Databases

A database is essential for the system to store and organize the collected sensor data. There is so many database types are considered depending on the specific data, we will mention the main of them :

a- Relational Databases(SQL):

These databases excel at storing structured data with well-defined relationships, making them ideal for user information management in online stores or social media platforms (e.g., user profiles, connections, friend suggestions). Relational databases are ideal for storing user information and timestamps associated with sensor data.

While they can store timestamps associated with sensor data, their structured format might not be the most efficient choice for handling large volumes of sensor readings. In such cases, time-series databases offer better optimization for time-stamped data.

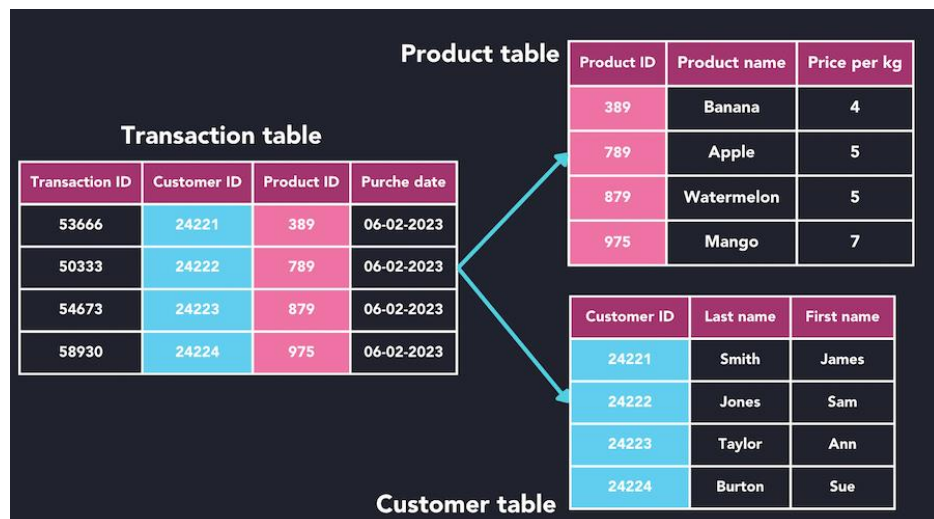


FIGURE 3 EXAMPLE OF A RELATIONAL DATABASE

b- Non-relational databases (NoSQL):

A non-relational database, also known as a NoSQL database, does not use the tabular schema of rows and columns found in traditional databases. Instead, it uses a storage model optimized for specific data types. These databases are highly flexible and scalable, making them ideal for handling large volumes of unstructured or semi-structured data. Common NoSQL storage models include document databases, key-value stores, and graph databases.

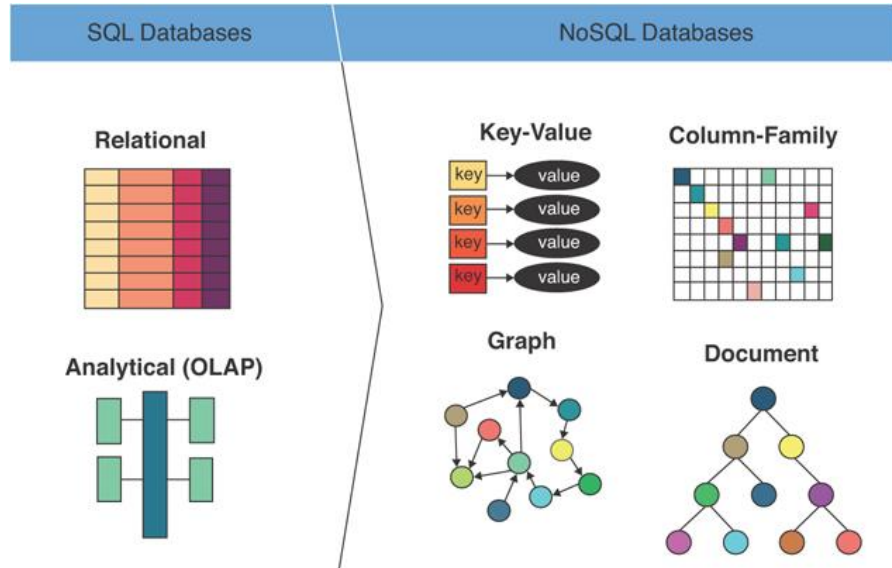
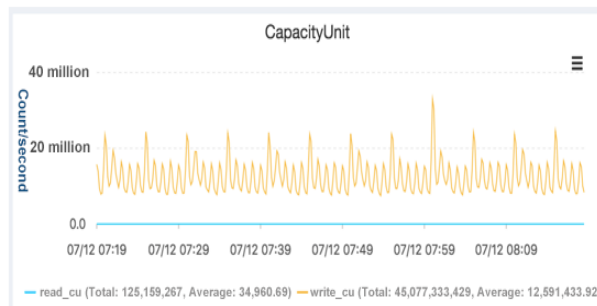


FIGURE 4 THE DIFFERENT BETWEEN SQL AND NON SQL DATABASES

c- Time-Series Databases (TSDBs):

TSDBs database are optimized for storing and analyzing data collected over time. They handle high data volumes efficiently and are ideal for health measurements (like ECG signals and heart rate), Surveillance camera, Trend Analysis, Trading and Real-time Applications.



Monitoring time series data

| Order No. | Time | Location | Event |
|-------------|------------------|-----------|--|
| 77165205838 | 2018-07-10 10:00 | Shanghai | Ship |
| 77165205838 | 2018-07-11 12:00 | Yuhang | Arrived in Hangzhou |
| 77165205838 | 2018-07-11 14:00 | Zhuantang | Start delivery and arrive at Apsara Park |
| 77165205838 | 2018-07-11 16:00 | Zhuantang | Accepted |

Status time series data

FIGURE 5 TSDB EXAMPLE

1.2.2 Cloud Services

Cloud services are application and infrastructure resources that exist on the Internet, these services, allowing customers to leverage powerful computing resources without having to purchase or maintain hardware and software, only buying a monthly or yearly subscribing.

Cloud services offer scalability and flexibility for data storage and processing. This system can explore options like :

- **Public Cloud:** Provides a readily available infrastructure on a pay-as-you-go basis.
- **Private Cloud:** Offers greater control and security for sensitive data, but may require dedicated hardware investment.
- **Hybrid Cloud:** Combines public and private cloud options, offering a balance between flexibility and security.

1.2.3 Security Measures

Using cloud include a sensitive information so ensuring the privacy and security of health data is paramount. The system will implement robust security measures, including:

- **Encryption:** Protecting data confidentiality both at rest and in transit.
- **Access Control:** Restricting access to authorized users and devices.
- **Regular Backups:** Ensuring data redundancy and recovery in case of system failures.

1.3 Health Monitoring Measures

The primary role of the health monitoring system is to enable seamless access to patient information without the need for physical visits to the doctor, thus paving the way for the development of electronic platforms for medical care that eliminate the necessity of traveling to clinics or hospitals. This is particularly beneficial for elderly individuals and those residing in remote areas, as it leverages technological advancements to create an efficient healthcare system.

The system operates by accurately collecting data on the user's health parameters and integrating this information into an IoT platform. This data can then be accessed remotely by healthcare professionals from any location. Furthermore, the system is designed to provide suggestions and advice to enhance patient recovery. In the event of a health emergency, the system is capable of issuing alert notifications and facilitating direct communication, including the sharing of critical information and the user's location with emergency services. The system will focus on collecting and analyzing the following health data:

a- ECG Signal :

Electrocardiogram (ECG) signals capture the electrical activity of the heart. ECG data analysis can help detect abnormalities like arrhythmias (irregular heartbeats) and potential heart problems.

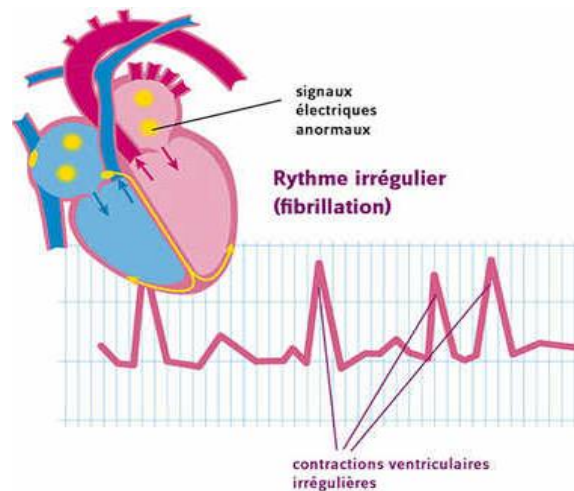


FIGURE 6 ECG SIGNAL

b- Blood Oxygen Saturation (SpO2):

Oxygen is breathed into the lungs. The oxygen then passes into the blood where the majority of the oxygen attaches to hemoglobin. Hemoglobin is a protein located inside our red blood cells that transports the oxygen through the bloodstream to the rest of our body and tissues. In this way, our body is given the oxygen and nutrients it needs to function.

SpO2 indicates the percentage of oxygen-carrying hemoglobin in the blood. Monitoring SpO2 can be crucial for individuals with respiratory conditions and those at risk of oxygen deprivation.

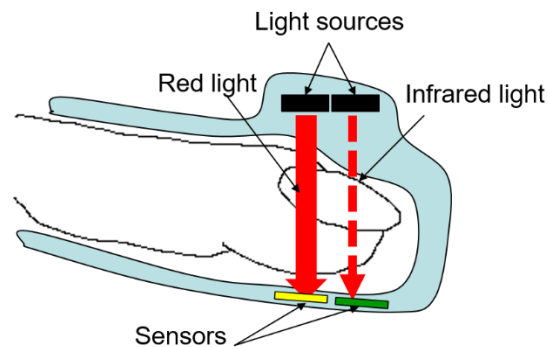


FIGURE 7 PULSE_OXIMETRY

c- Heart Rate :

Heart rate is the number of times the heart beats per minute. Monitoring heart rate can provide insights into overall cardiovascular health and identify potential issues like tachycardia (fast heart rate) or bradycardia

| Average heart rates by age | | |
|----------------------------|--|---|
| Age in years | Average maximum heart rate in beats per minute | Target heart rate range in beats per minute |
| 40 | 180 | 90 to 153 |
| 45 | 175 | 88 to 149 |
| 50 | 170 | 85 to 145 |
| 55 | 165 | 83 to 140 |
| 60 | 160 | 80 to 136 |
| 65 | 155 | 78 to 132 |

FIGURE 8 AVERAGE HEART RATES BY AGE

(slow heart rate). The maximum heart rate also depends on multiple factors. As people age, average maximum heart rate falls. A commonly used formula to determine our maximum heart rate is 220 minus our age in years (220min - age).

d- Blood Pressure

This can be added as an optional measurement for comprehensive health monitoring. Blood pressure readings can help identify hypertension (high blood pressure) and hypotension (low blood pressure), both of which require medical attention.

| Blood pressure | Systolic mmHg (top number) | | Diastolic mmHg (bottom number) |
|----------------|----------------------------|--------|--------------------------------|
| Low | Less than 90 | and | Less than 60 |
| Ideal | Less than 120 | and | Less than 80 |
| Normal | 120–129 | or | 80–84 |
| Normal to high | 130–139 | or | 85–89 |
| High | Above 140 | and/or | Above 90 |

FIGURE 9 BLOOD PRESSURE TABLE

1.4 Benefits of IoT in Health monitoring

Utilizing IoT in health monitoring offers a range of benefits:

- **Remote Patient Monitoring:** Physicians can remotely monitor patient health data, enabling timely intervention and improved care management.
- **Early Detection of Health Concerns:** Continuous health data collection can facilitate the early detection of potential health issues.
- **Improved Chronic Disease Management:** IoT systems can support patients with chronic conditions by monitoring vital signs and providing medication reminders.
- **Personalized Healthcare Insights:** Data analysis can provide personalized healthcare insights allowing for customized treatment plans and preventative measures.

1.5 Components selected for this project

Choosing parts to make this project a success includes respecting some conditions, such as cost. The cost of the parts must be within the limits of the benefit they provide and their ease of availability. Keeping this in mind, the following components were selected:

- ESP32-DevkitC-V4 Microcontroller.
- AD8232 Sensor.
- MAX30100 Sensor.

1.5.1 ESP32-DevkitC-V4

The ESP32-DevKitC V4 is a small-format ESP32-based development board made by Espressif that uses the ESP32-WROOM-32D microcontroller. Most I/O ports are distributed over pin headers on both sides for easy communication.

Developers can either hardwire devices or mount the ESP32-DevKitC V4 to the board.

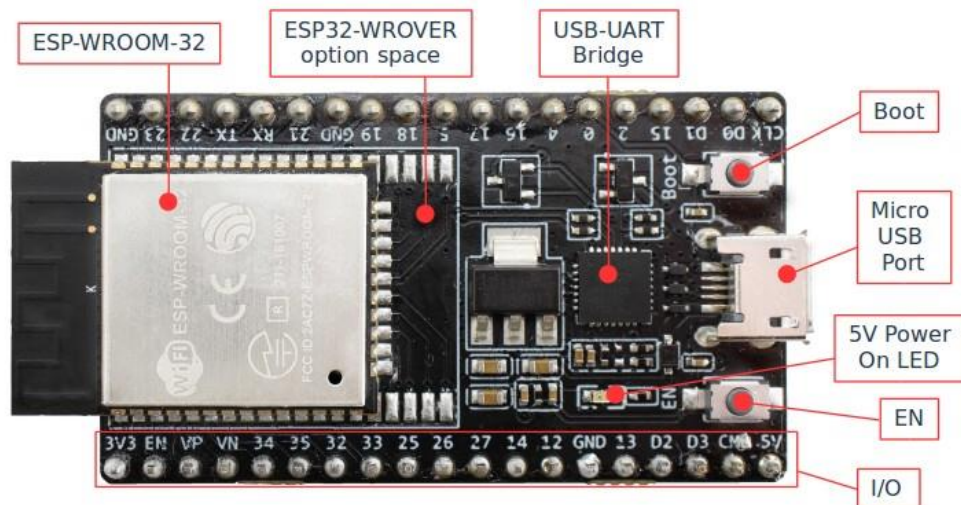


FIGURE 10 ESP32-DEVKITC V4 WITH ESP32-WROOM-32 SOLDERED MODULE

I will summarize the basic information about this controller in a table :

| Key element | The description |
|-------------|---|
| EN | Reset button |
| Boot | Download button. Holding Boot then pressing EN initiates firmware download mode to download the firmware via the serial port. |

| | |
|--------------------|---|
| USB-to-UART Bridge | The unique USB-UART bridge chip provides transfer rates of up to 3 Mbps. |
| Micro USB Port | USB interface. Power supply of the card as well as the communication interface between a computer and the ESP32-WROOM-32 module. |
| 5V Power On LED | Lights up when USB or external 5V power is connected to the board. |
| I/O (E/S) | Most of the ESP module pins are distributed across the pin headers on the board. we can program ESP32 to enable multiple functions such as PWM, ADC, DAC, I2C, I2S, SPI, etc. |

TABLE 1 ESP32 BASIC BUTTONS AND THEIR ROLE

ESP-32-WROOM-32D

This microcontroller included inside the esp32-DevkiC-V4 contains many features that are essential in creating our system, such as wifi, bluetooth, which gives us the ability to communicate wirelessly and send information by them.



FIGURE 11 ESP32-WROOM-32D

The naming convention is a little fuzzy, so I'll break it down:

- **ESP:** (a family of microcontrollers manufactured by Espressif Systems)
- **32:** (the device family)
- **WROOM:** (subdivision of the device family)
- **32D:** (The model number or model code)

ESP32-WROOM-32D are powerful generic Wi-Fi + Bluetooth + Bluetooth LE MCU modules that target a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, Music streaming and MP3 decoding.

I will summarize all its features in a table :

| Catégories | Articles | | Caractéristiques |
|-------------------|---|---|--|
| Wi-Fi | Protocoles | | 802.11 b/g/n (802.11n up to 150 Mbps) |
| | | | Agrégation A-MPDU et A-MSDU et prise en charge de l'intervalle de garde de 0,4 μ s |
| | Plage de fréquence centrale de fonctionnement canaliser | | 2412 ~ 2484 MHz |
| Bluetooth | Protocoles | | Spécification Bluetooth v4.2 BR/EDR et Bluetooth LE |
| | Radio | | Récepteur NZIF avec une sensibilité de -97 dBm |
| | | | Émetteur de classe 1, classe 2 et classe 3 |
| | | | AFH |
| Audio | | CVSD et SBC | |
| Hardware | Interfaces de modules | Carte SD, UART, SPI, SDIO, I2C, LED PWM, Moteur PWM, I2S, IR, compteur d'impulsions, GPIO, capteur tactile capacitif, ADC, DAC, Two-Wire Automotive, compatible avec ISO11898-1 | |
| | Architecture | | 32 bits |
| | RAM | | 512 KB |
| | Pins | | 36 Pins |
| | CPU Cores et clock | | 2 (dual core) 80-240 MHz |
| | Cristal intégré | | 40 MHz cristal |
| | Flash SPI intégré | | 4 MB |
| | Tension de fonctionnement/Alimentation | | 3.0 V ~ 3.6 V |
| | Courant de fonctionnement | | Moyen : 80 mA |
| | Courant minimum délivré par la puissance la fourniture | | 500 mA |
| | Plage de température ambiante de fonctionnement recommandée | | -40 °C ~ +85 °C |
| | Niveau de sensibilité à l'humidité (MSL) | | Level 3 |
| | Capteur sur puce | | Capteur à effet Hall |

TABLE 2 ESP32 SPECIFICATION

The complete drawing of the ESP32 disk board is shown in Figure 1.5.3. As we can see, each pin has been carefully labeled with its corresponding:

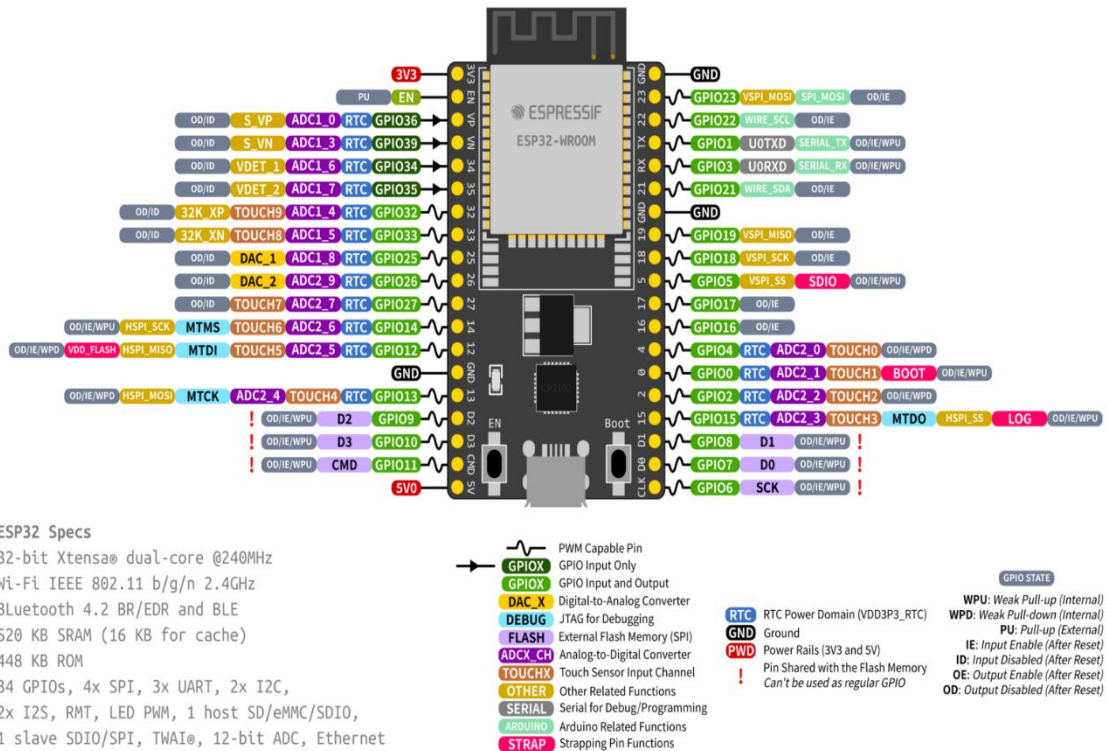


FIGURE 12 THE PIN DIAGRAM OF THE ESP32-DEVKITC-V4

1.5.2 AD8232 (ECG Sensor)

The AD8232 Sensor is an integrated signal conditioning block for ECG and other biopotential measurement applications. It is designed to extract, amplify, and filter small biopotential signals in the presence of noisy conditions, such as those created by motion or remote electrode placement, allowing precise monitoring of heart activity.

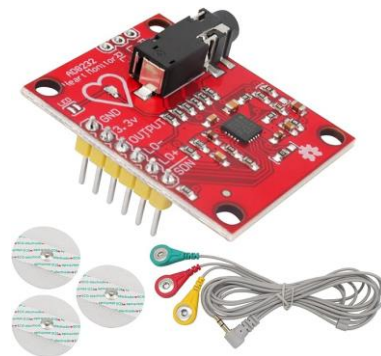


FIGURE 13 AD8232 SENSOR

This sensor has 6 Pinout:

- e- GND (Ground):** This is the ground reference for the circuit. It should be connected to the ground of the power supply.
- f- 3.3V (Power Supply):** This pin supplies the power to the AD8232 module. It typically requires a 3.3V power source.
- g- OUTPUT:** This pin outputs the analog ECG signal. This is the processed signal that can be fed to an analog-to-digital converter (ADC) for further processing or visualization.
- h- LO- (Leadoff Detection -):** This pin is used for lead-off detection, which indicates if an electrode is disconnected. It is connected to one of the inputs of the operational amplifier for leadoff detection.
- i- LO+ (Leadoff Detection +):** Similar to LO-, this pin is used for lead-off detection. It works with LO- to detect if electrodes are properly attached to the patient.
- j- SDN (Shutdown):** This pin is used to put the AD8232 into a low-power shutdown mode. When SDN is pulled high, the device goes into shutdown mode. When it is low, the device is active.

These pins allow the AD8232 to function effectively for ECG signal acquisition and processing, ensuring proper power, signal output, and lead-off detection capabilities.

1.5.3 MAX30100 (Oximeter and Heart Rate Sensor)

This integrated sensor combines pulse oximetry and heart rate monitoring capabilities. It uses light-emitting diodes (LEDs) and photodetectors to measure SpO₂ and heart rate.

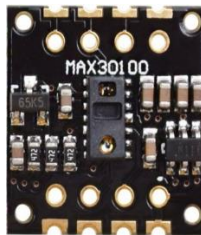


FIGURE 14 MAX30100

Here's the detailed pinout description for the MAX30100:

- k- GND (Ground):** Connect both of these pins to the ground of our circuit.
- l- RD (Red LED Anode):** This pin is the anode for the red LED. Connect this pin to the red LED's anode.
- m- IRO (IR LED Anode):** This pin is the anode for the infrared LED. Connect this pin to the IR LED's anode.
- n- INT (Interrupt):** This pin is used for interrupt signaling. The MAX30100 pulls this pin low to signal that data is ready to be read.
- o- VIN (Power Supply):** This pin should be connected to a voltage supply in the range of 1.8V to 3.3V.
- p- SDA (Serial Data) :** This is the I2C data line. Connect this pin to the I2C data line on our microcontroller.
- q- SCL (Serial Clock):** This is the I2C clock line. Connect this pin to the I2C clock line on our microcontroller.

1.6 Software and Platforms

1.6.1 Arduino IDE

The Arduino IDE is an open source program used to program Arduino boards. It is an integrated development environment, developed by arduino.cc. Allow writing and uploading code to Arduino boards and other supported boards. It consists of several libraries and a set of small project examples.

The Arduino software (IDE) is compatible with different operating systems to be used such as Windows, Linux, Mac OS, Arduino software is easy to use for beginners or advanced users. It is used to start programming electronics and robotics, and building interactive prototypes.

Finally we could say that Arduino IDE is a tool for developing new things. And create new electronic projects by anyone because it is easy to use, even children can learn using this program and boards.

- **Tools:** Provides various development tools and hardware settings such as selecting board type, serial monitor settings, managing libraries, and configuring programmer options.
- **Help:** Offers assistance and resources—accessing documentation, checking for updates, and finding additional online resources.

Under the Menus section, we find this bar:

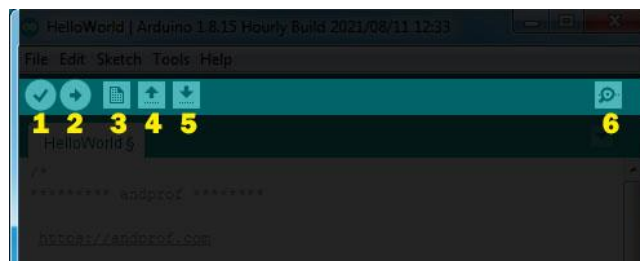
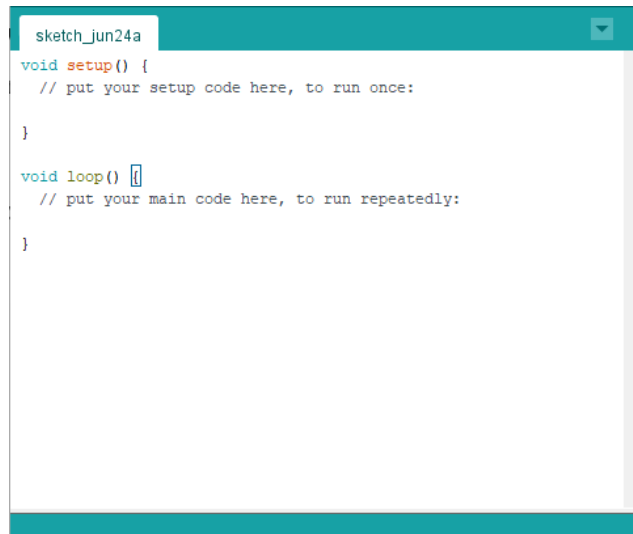


FIGURE 17 ARDUINO IDE TOOLS-BAR

The role of these tools is as follows:

- 1- **Verify:** this button use to review the code, or make sure that is free from mistakes.
- 2- **Upload:** this button is use to upload the code on the arduino board.
- 3- **New:** this button use to create new project, or sketch (sketch is the file of the code).
- 4- **Open:** is use when we want to open the sketch from sketchbook.
- 5- **Save:** save the current sketch in the sketchbook.
- 6- **Serial monitor:** showing the data which have been sent from arduino.

Also the code editor section:



```
sketch_jun24a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

FIGURE 18 ARDUINO IDE – CODE SECTION

This section for code editor is liberator of codes, is the white space in the program, in which codes are been writing, and modifying on it.

Under the code section we have statue bar setion:



FIGURE 19 ARDUINO IDE – STATUE SECTION

Status bar is a space can be found down the code editor, through it showing the status of operations completion like compiling, uploading...etc.

The last thing in down the black space is the program notifications section:

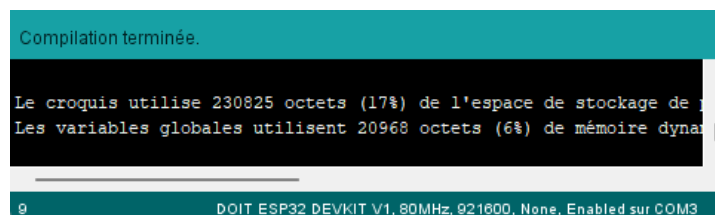


FIGURE 20 ARDUINO IDE – PROGRAM NOTIFICATIONS

This program notifications is showing us the mistakes of codes, and some problems that

can be face us during the programming process. And clarifies to our the type of the mistake or the problem which happened and it reason.

And it presents some instruction through it, which we have to apply to process the mistake or the problem.

1.6.2 InfluxDB

In this chapter, we previously talked about databases and their types.

InfluxDB is a non-relational, time-series database optimized for high-speed data ingestion and querying based on timestamps. It does not use SQL for querying but employs its own language, InfluxQL, tailored for time-series data. InfluxDB excels in storing and retrieving data indexed by time and date, making it ideal for applications like sensor data, metrics, and IoT devices. It supports real-time data ingestion and querying, allowing for immediate analysis and integration with machine learning models for tasks such as real-time forecasting and anomaly detection.

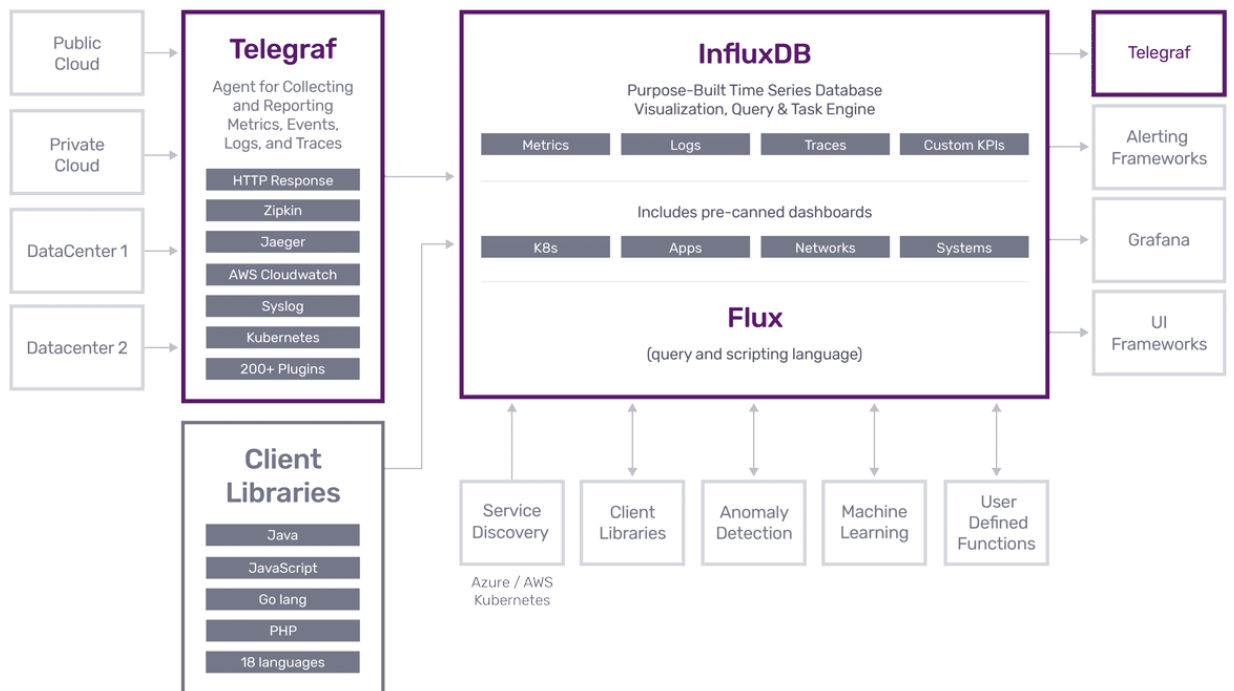


FIGURE 21 INFLUXDB ARCHITECTURE

1.6.3 Grafana

Grafana is a versatile open source platform known for its powerful visualization capabilities and support for various data sources, including databases like InfluxDB. It enables users to create customizable dashboards to effectively monitor and analyze time series data. Grafana's integration with InfluxDB enables seamless querying and visualization of time-stamped metrics, making it a popular choice among organizations for real-time monitoring and monitoring tasks. The flexible alerting system and extensive community-contributed plugins enhance its usefulness for building comprehensive monitoring solutions.



FIGURE 22 GRAFANA DASHBOARD

Conclusion:

This chapter provides a comprehensive overview of the Internet of Things (IoT) and its core components. We explored the basic concepts of the Internet of Things, including sensor data collection, processing, and communications. We have highlighted the various applications of IoT in both the public and private sectors, and how it's impact to the industries.

We then looked at the specific components that were chosen for this health surveillance project. We have explained the functions of ESP32-DevKitC-V4 microcontroller, AD8232 ECG sensor, MAX30100 oximeter and heart rate sensor. We also explored software and platforms that would be useful in developing this project, including the Arduino IDE for programming the microcontroller, InfluxDB for storing time series data, and Grafana for data visualization.

With this foundation, the next chapters will delve into the technical aspects of the project. We will explore the coding required to program the microcontroller to interact with sensors, data transfer protocols, and security considerations. We'll also detail configuring InfluxDB to efficiently store data and explore creating interactive dashboards with Grafana. Finally, the project will be comprehensively tested and improved to ensure that it provides accurate and reliable monitoring of health data.

Chapitre 2 Conception

Introduction :

This chapter delves into the design and planning of our health surveillance system. We will explore the necessary hardware and software components, identify their functions and how they work in harmony to achieve the project goal of collecting, processing, storing and visualizing health data.

The success of any project depends on planning and design before starting work. In this chapter, we will carefully dissect the design of our health monitoring system. We will embark on a journey to understand how the hardware and software components used work and interconnect, and define their functions, how they work, and their setup. This is what ultimately allows the system to achieve its primary goal of collecting, processing, storing and visualizing health data to enable users to access their health information remotely in a simple, easy-to-read interface.

2.1 System Architecture

2.1.1 Strategy

First of all, the most important thing in this project is the container that contains our information, the so-called database. The reason we chose influxdb is its ease of use and reliability. It is also open source, since it is open source, we can use it without licensing costs, which makes the project accessible to everyone. Many companies also rely on InfluxDB for their performance, which increases confidence in its capabilities.

The second thing is that we will connect our microcontroller to the database to make sure that it works well. This will be done via an access point, which can be a router or switch. This step is necessary after which we can add whatever we want, but the most important thing is that this stage works well.

Once the connection is established, we can connect the sensors to the ESP32. By writing code, we will be able to read the sensor data and send it to the database for storage. The modular approach to adding sensors later with code modifications allows the system to be easily expanded as needed.

Finally, we can connect the database to Grafana to visualize the data. This provides an easy-to-use interface to explore the health data collected.

However, it is important to note that we can also develop a custom application to display information directly from the database, bypassing Grafana. This approach provides flexibility in terms of user interface design.

2.2 Hardware Design

This section describes the physical components that make up our health monitoring system.

2.2.1 Microcontroller Layer

The ESP32 has 38 Pinout which helps us to connect many sensors without problem. It also contains communication protocols, some of which we mention:

Physical ports:

- **SPI (Serial Peripheral Interface):** The ESP32 actually has multiple SPI interfaces, but not all pins are user configurable. Here is the detail:
- **SPI0 and SPI1:** Mainly used internally to access flash memory. Not recommended for user applications.
- **SPI2 (often referred to as HSPI):** This is the most commonly used SPI interface for user applications. Pins:

Clock (SCLK): GPIO14 (configurable)

Master Out Slave In (MOSI): GPIO23 (configurable)

Master-in-Slave-Out (MISO): GPIO19 (configurable)

Chip Identification (CS): we can use any general purpose GPIO pin to identify the chip depending on our specific SPI device.

- **SPI3 (often referred to as VSPI):** Another user-configurable SPI interface. Pins:

Clock (SCLK): GPIO5 (fixed)

Master Out Slave In (MOSI): GPIO21 (fixed)

Master In Slave Out (MISO): GPIO18 (fixed)

Chip Identification (CS): we can use any general purpose GPIO pin to identify the chip depending on our specific SPI device.

- **I2C (integrated circuits):** The ESP32 has one I2C interface. Pins:

Clock (SCL): GPIO22 (fixed)

Data (SDA): GPIO21 (static)

Additional communication protocols:

- **UART (Universal Asynchronous Receiver/Transmitter):**

The ESP32 has multiple UART interfaces, but only UART0 is generally used for user applications. Pins:

RX (Receive): GPIO16 (fixed)

TX (transmission): GPIO15 (fixed)

I2S (Audio Inter-IC):

- **The ESP32 has two I2S interfaces.**

Specific pin assignments can vary depending on the chosen I2S interface and its configuration.

Wi-Fi Connectivity:

This integrated Wi-Fi capability allows the ESP32 to:

- **Transmit sensor data wirelessly:** Collected health data can be transmitted to a smartphone, computer, or cloud platform for further analysis, visualization, or storage.
- **Receive updates and configurations:** The ESP32 can receive firmware or configuration updates wirelessly, ensuring the system stays current.

Bluetooth Connectivity:

Adding to its versatility, the ESP32 also boasts Bluetooth connectivity. This enables:

- **Low-power data transmission:** Bluetooth Low Energy (BLE) allows for power-efficient data exchange with compatible devices like smartphones or smartwatches, ideal for continuous health monitoring without draining the battery.
- **Direct communication with peripherals:** The ESP32 can directly connect to Bluetooth-enabled devices like wireless headphones or smart scales, expanding the functionality of our health monitor.

Beyond Communication:

The ESP32 offers additional features that enhance our health monitor's capabilities:

- **Low Power Consumption:** A critical aspect for wearables, the ESP32 operates efficiently, maximizing battery life for extended monitoring periods.
- **Processing Power:** The ESP32 packs a processing punch, allowing for on-device data processing and analysis before transmission, potentially reducing data volume and improving efficiency.

By combining these features, the ESP32 empowers us to create a feature-rich health monitor that seamlessly collects, transmits, and analyzes vital health data.

2.2.2 Sensors Layer

a *ECG (Electrocardiogram):*

The AD8232 is an integrated circuit specifically designed for measuring weak bioelectrical signals, particularly those related to the heart, making it ideal for building electrocardiogram (ECG) devices. Here's a breakdown of its key features:

- **Amplifies and filters signals:** The AD8232 takes the tiny electrical signals from the heart and amplifies them to a usable level. It also filters out unwanted noise from the surrounding environment (like muscle movement or power line interference) to provide a clearer ECG signal.
- **Single lead input:** The AD8232 is designed for single-lead ECG measurement, typically using three electrodes attached to the body. More advanced ECG systems might use multiple leads for a more comprehensive picture of heart activity.
- **Low power consumption:** This is crucial for battery-powered wearable devices.
- **Easy to use:** The AD8232 is a relatively simple chip to integrate into our project with the help of supporting circuitry.

Connecting the AD8232 to ESP32:

The AD8232 doesn't directly connect to the ESP32 through SPI, I2C, or other communication protocols commonly used for sensors. the connection will be as the follow:

- **Analog output:** The AD8232 provides an analog output voltage that represents the amplified and filtered ECG signal.
- **Analog-to-Digital Converter (ADC):** This is where the ESP32 comes in. One of the ESP32's built-in ADC channels converts the analog output voltage from the AD8232 into a digital signal that the ESP32 can understand and process.

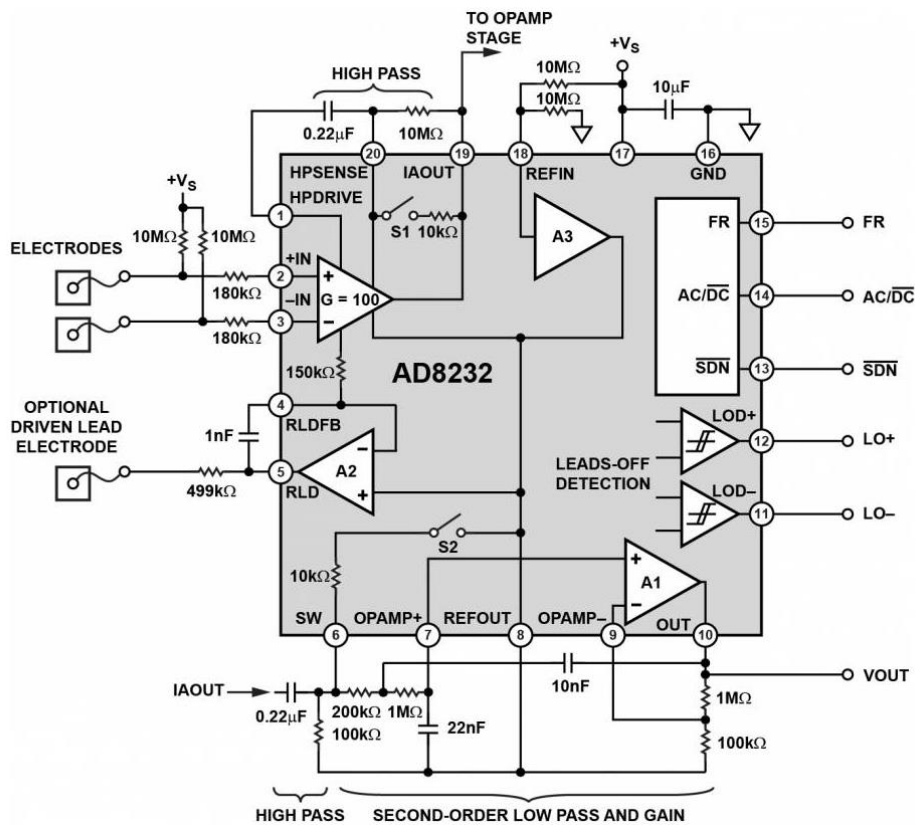


FIGURE 23 AD8232 SCHEMATIC

b *Pulse Oximetry (SpO₂):*

The MAX30100 sensor plays a vital role in health monitoring system, acting as a window to our user's blood oxygen saturation (SpO₂) and heart rate. This integrated pulse oximetry and heart-rate sensor solution simplifies the design process by incorporating several key components:

- **Dual LEDs:** The MAX30100 houses two light-emitting diodes (LEDs) – typically a red LED and an infrared (IR) LED. These LEDs emit light wavelengths that are differentially absorbed by blood depending on its oxygen saturation level.
- **Photodetector:** The sensor also integrates a photodetector, which captures the light that passes through the user's fingertip (or other designated body part) after interacting with the blood vessels.
- **Optimized Optics:** The design of the MAX30100 includes optimized optics to ensure efficient light transmission and collection, leading to more accurate measurements.

- **Low-Noise Analog Signal Processing:** Extracting the subtle variations in light absorption requires precise signal processing. The MAX30100 boasts low-noise analog signal processing circuitry, minimizing electrical noise and ensuring reliable data acquisition.

By combining these elements, the MAX30100 allows our health monitor to:

- **Measure Blood Oxygen Saturation (SpO2):** The sensor measures the ratio of oxygenated hemoglobin to total hemoglobin in the blood. This SpO2 value is a crucial indicator of respiratory health.
- **Track Heart Rate:** The pulsating nature of blood flow due to the heart's activity causes slight variations in light absorption. The MAX30100 detects these variations and translates them into heart rate measurements.

The compact size and integrated design of the MAX30100 make it an ideal choice for wearable health monitors. It simplifies the hardware design and reduces the number of discrete components needed, streamlining the development process.

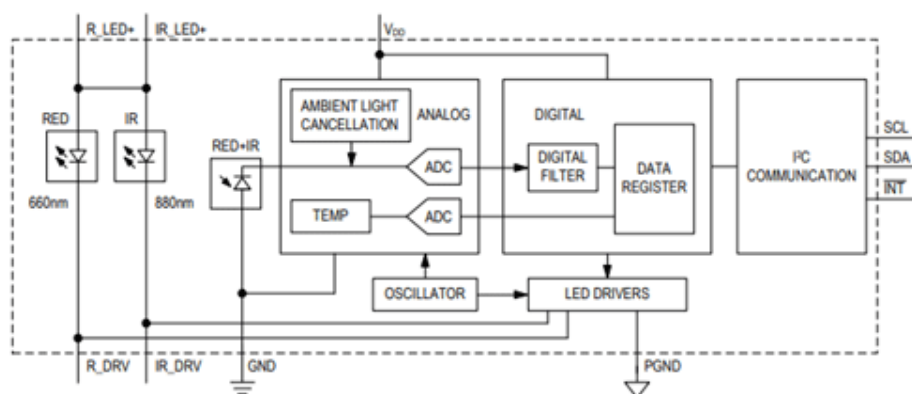


FIGURE 24 THE FUNCTIONAL BLOCK FOR THE MAX30100 MODULE

2.2.3 The circuit

The detail of Pin Connections to our circuit will be as the following:

Power and Ground Connections

- **MAX30100:**
 - VCC (Power) to 3.3V on the ESP32
 - GND (Ground) to GND on the ESP32
- **AD8232:**
 - 3.3V (Power) to 3.3V on the ESP32
 - GND (Ground) to GND on the ESP32

I2C Communication for MAX30100:

- SDA to GPIO 26 on the ESP32
- SCL to GPIO 24 on the ESP32

Analog and Digital Connections for AD8232:

- LO- (Leads Off Negative) to GPIO 32 on the ESP32
- LO+ (Leads Off Positive) to GPIO 33 on the ESP32
- OUTPUT (Analog Output) to GPIO 34 on the ESP32

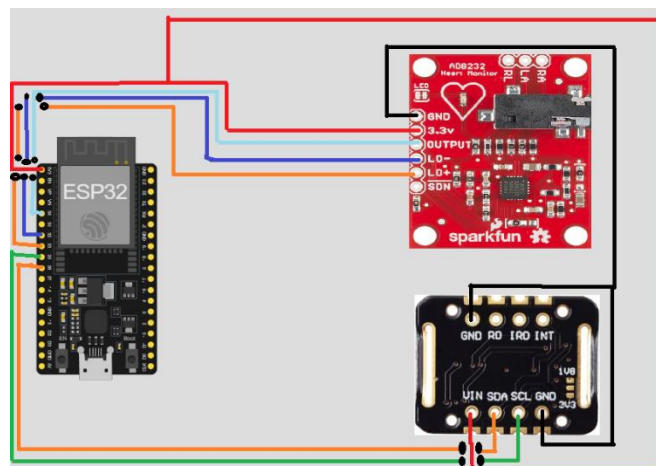


FIGURE 25 SCHEAMTIC OF CIRCUIT

2.3 Software Development

In this section we will use a virtual machine to repeat steps we did before to install those software and use them we mention that because we found it useful and maybe the reader want to know about that to test the system by the way. for start we need to go to Control panel in windows (I use windows 11 but it is the same thing in other windows). Next by clicking in Program and features or we can just write that

- **Panneau de configuration\Tous les Panneaux de configuration\Programmes et fonctionnalités** -

In search bar and we go there. Then on the right click Turn Windows features on or off and search for Windows Sandbox active this feature and restart our computer and it's work.

Note: This virtual machine window give a 4 Gb of ram and 40 Gb of Storage and internet access, but one when we close this window everything we have download it or install it there is gone.

2.3.1 Arduino IDE

First thing here we will install Arduino IDE, Install it and prepare it for ESP32 and Sensors by following this steps:

- Step 01: To download it we can just go to www.arduino.cc/en/Main/Software or just search in windows, Linux or Mac Store it is available there.
- Step 02: After install it and open it and go to **File> Preference** (we can also click on Ctrl + , as shortcut).

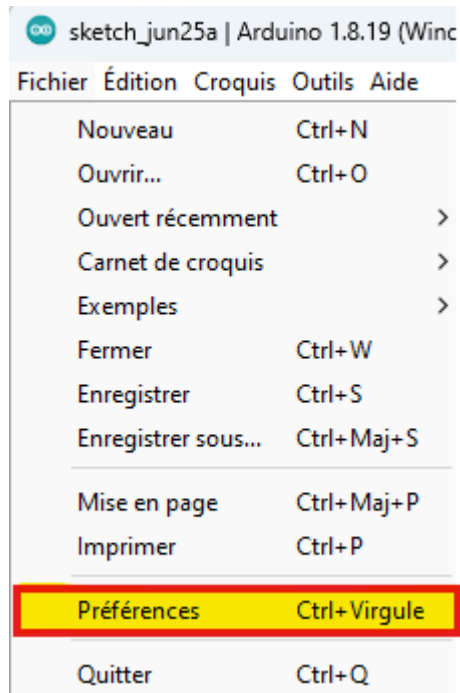


FIGURE 26 ARDUINO IDE SETUP 1

- Step 03: Copy and paste the following Link https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json into the Additional Boards Manager URLs.

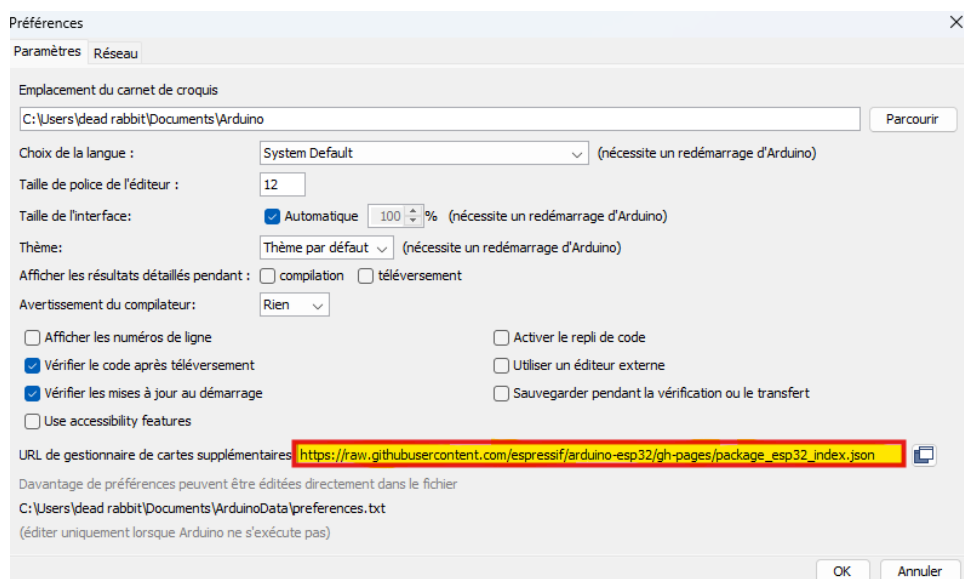


FIGURE 27 ARDUINO IDE SETUP 2

- Step 04: Open the Boards Manager by going to **Tools > Board > Boards Manager...**

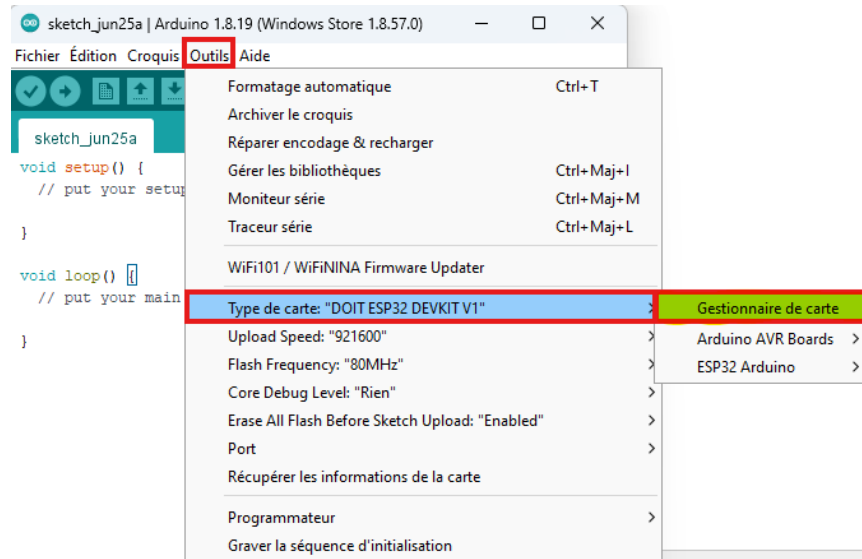


FIGURE 28 ARDUINO IDE SETUP 3

- Step 05: Then Search for **ESP32** in Search Bar and press install in the following library:

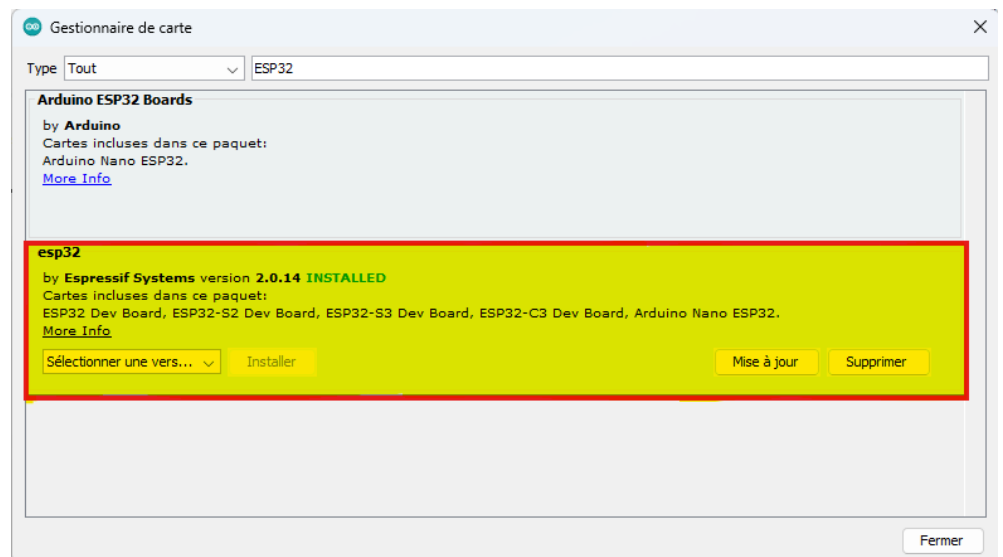


FIGURE 29 ARDUINO IDE SETUP 4

- Step 06: Go to **Sketch > Include Library > Manage Libraries** or by clicking at **Ctrl + maj + l**.

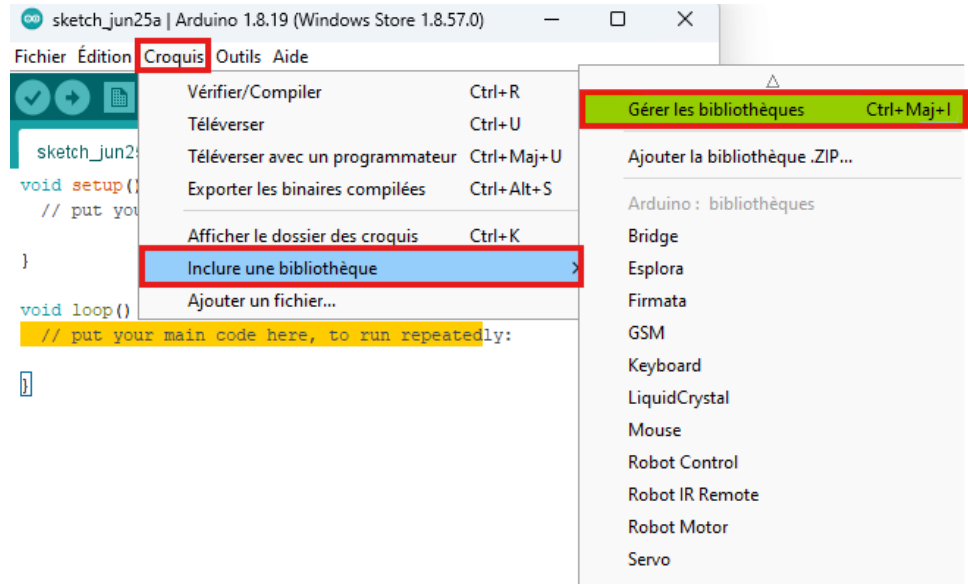


FIGURE 30 ARDUINO IDE SETUP 5

- Step 07: Type **MAX30100** in Search Bar and install these libraries. Just to note we install those two libraries because we have been testing two different sensors and both of them named **MAX30100**, and each one worked for a different sensor.

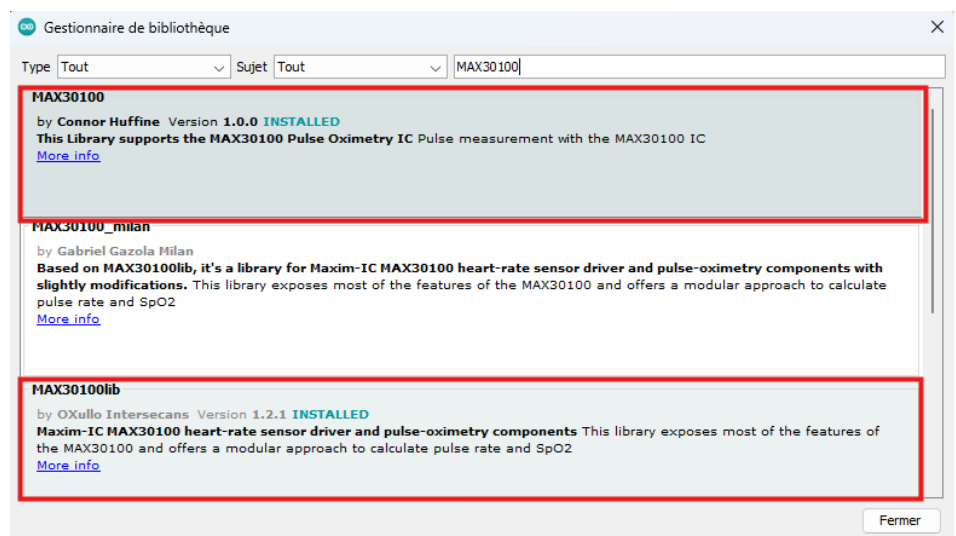


FIGURE 31 ARDUINO IDE SETUP 6

- Step 08: now we need to install InfluxDB library by searching for **influxdb** and install this library:

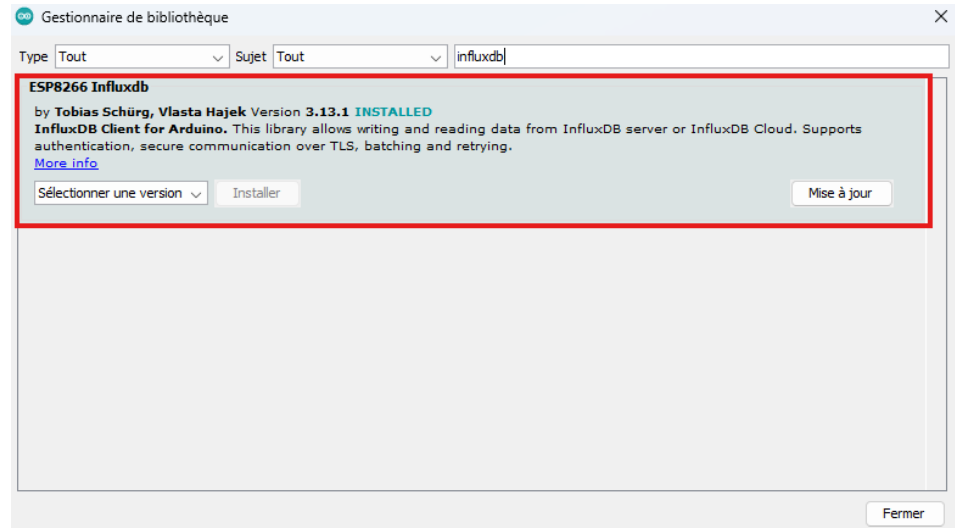


FIGURE 32 ARDUINO IDE SETUP 7

Note: other library like WI-FI and Bluetooth will included automatically when install ESP32 librarie, and AD8232 doesn't need library to work.

2.3.2 Setup InfluxDB

Before starting we want to make we know that we can use InfluxDB without install it just using the cloud to save and access to data it will limited because we will use another to store our data and that will be payable for sure, or we can download InfluxDB software in our machine that could be a normal PC, a Raspberry pi or any ARM device. For me we will run it in my pc we prefer PC because in that moment a low profil PC are cheap and can do better than a Raspberry pi.

To install and configure it we need to follow this steps:

- Step 01: First thing we to open the official website of InfluxDB from the link

<https://docs.influxdata.com/influxdb/v2/install/?t=Linux#command->

[line-examples](#) . After opening the link we will found a multiple option for the operating systems in my case we will chose windows.

Install InfluxDB

The InfluxDB v2 time series platform is purpose-built to collect, store, process and visualize metrics and events.

- [Download and install InfluxDB v2](#)
- [Start InfluxDB](#)
- [Download, install, and configure the `influx` CLI](#)

1. Download and install InfluxDB v2.

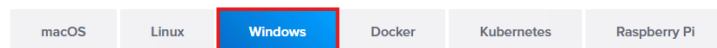


FIGURE 33 INFLUXDB SETUP 1

- Step 02: Scroll down and we will find download button click it and wait Influxdb v2 complete

InfluxDB and the influx CLI are separate packages
The InfluxDB server (`influxd`) and the `influx` CLI are packaged and versioned separately.
You'll install the `influx` CLI in a later step.



Expand the downloaded archive into `C:\Program Files\InfluxData\` and rename the files if desired.

FIGURE 34 INFLUXDB SETUP 2

- Step 03: Then we go to the folder where the InfluxDB file was downloaded and copy the Path, after that open the PowerShell and write this command

```
Expand-Archive .\influxdb2-2.7.6-windows.zip -DestinationPath  
'C:\Program Files\InfluxData\'
```

Replace the dot with the Path of the folder, in my case it will be like that:

```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\WDAGUtilityAccount> Expand-Archive .\influxdb2-2.7.6-windows.zip -DestinationPath 'C:\Program Files\InfluxData\'
>>

```

FIGURE 35 INFLUXDB SETUP 3

- Step 04: Run it, this will expand the archive file that we download it and move it to a folder named InfluxData in Program Files.

After installing InfluxDB Successfully we can run it by opening CMD by searching it or use the shortcut “Windows key + R” and write CMD, and run the command `cd C:\Program Files\influxdata`, that will allows CMD moving to directories of influxdata folder. Then write `influxd` and hold enter that will run influxdb and we will se something like that:

```

Administrator: C:\Windows\system32\cmd.exe - influxd
C:\Program Files\InfluxData>influxd
2024-06-25T19:28:01.507611Z info Welcome to InfluxDB {"log_id": "0q0kpfG000", "version": "v2.7.6", "commit": "5439e06286", "build_date": "2024-04-12T21:51:33Z", "log_level": "info"}
2024-06-25T19:28:01.609395Z info Resources opened {"log_id": "0q0kpfG000", "service": "bolt", "path": "C:\Users\WDAGUtilityAccount\influxdbv2\influxd.bolt"}
2024-06-25T19:28:01.609395Z info Resources opened {"log_id": "0q0kpfG000", "service": "sqlite", "path": "C:\Users\WDAGUtilityAccount\influxdbv2\influxd.sqlite"}
2024-06-25T19:28:01.618873Z info Using data dir {"log_id": "0q0kpfG000", "service": "storage-engine", "service": "store", "path": "C:\Users\WDAGUtilityAccount\influxdbv2\engine\data"}
2024-06-25T19:28:01.618873Z info Compaction settings {"log_id": "0q0kpfG000", "service": "storage-engine", "service": "store", "max_concurrent_compactions": 6, "throughput_bytes_per_second": 50331648, "throughput_bytes_per_second_burst": 50331648}
2024-06-25T19:28:01.619438Z info Open store (start) {"log_id": "0q0kpfG000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "op_event": "start"}
2024-06-25T19:28:01.619438Z info Open store (end) {"log_id": "0q0kpfG000", "service": "storage-engine", "service": "store", "op_name": "tsdb_open", "op_event": "end", "op_elapsed": "0.000ms"}
2024-06-25T19:28:01.620084Z info Starting retention policy enforcement service {"log_id": "0q0kpfG000", "service": "retention", "check_interval": "30m"}
2024-06-25T19:28:01.620076Z info Starting precreation service {"log_id": "0q0kpfG000", "service": "shard-precreation", "check_interval": "10m", "advance_period": "30m"}
2024-06-25T19:28:01.622611Z info Starting query controller {"log_id": "0q0kpfG000", "service": "storage-retrieval", "concurrency_quota": 1024, "initial_memory_bytes_quota_per_query": 9223372036854775807, "memory_bytes_quota_per_query": 9223372036854775807, "max_memory_bytes": 0, "queue_size": 1024}
2024-06-25T19:28:01.636417Z info Configuring InfluxQL statement executor (zeros indicate unlimited). {"log_id": "0q0kpfG000", "max_select_point": 0, "max_select_series": 0, "max_select_buckets": 0}
2024-06-25T19:28:01.646630Z info Starting {"log_id": "0q0kpfG000", "service": "telemetry", "interval": "5m"}
2024-06-25T19:28:01.647176Z info Listening {"log_id": "0q0kpfG000", "service": "tcp-listener", "transport": "http", "addr": ":8086", "port": 8086}

```

FIGURE 36 INFLUXDB SETUP 4

That mean that our database is working in our pc, to access to it just open any navigator and write this `localhost:8086` we must see this window:

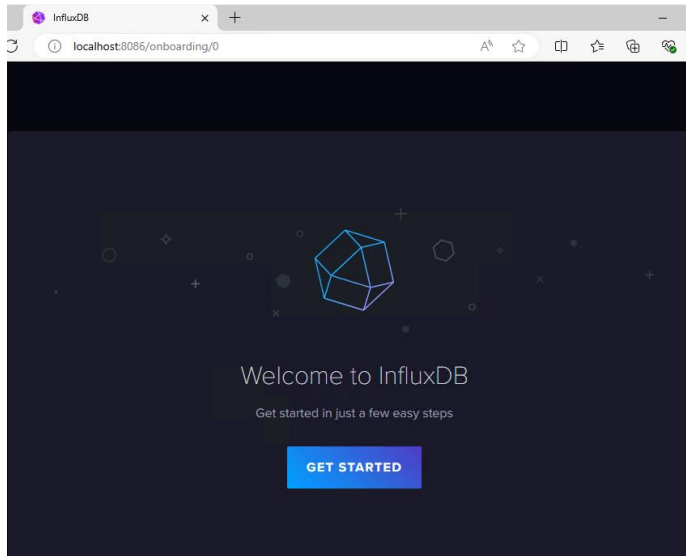


FIGURE 37 INFLUXDB SETUP 5

To complete configuring and setup our database we click - Get Started – and write username and password...etc, like that:

FIGURE 38 INFLUXDB SETUP 6

2.3.3 Setup NSSM

Each time we want to use our database we need to run CMD and write a specific command that's annoying and not practical because we will need a monitor in our PC server that's cost a lot and not everyone can do it.

To solve that there is a tool named NSSM that will run any service automatically, we just need to:

- Step 01: go to this link <https://nssm.cc/download> and download the last version of the software.

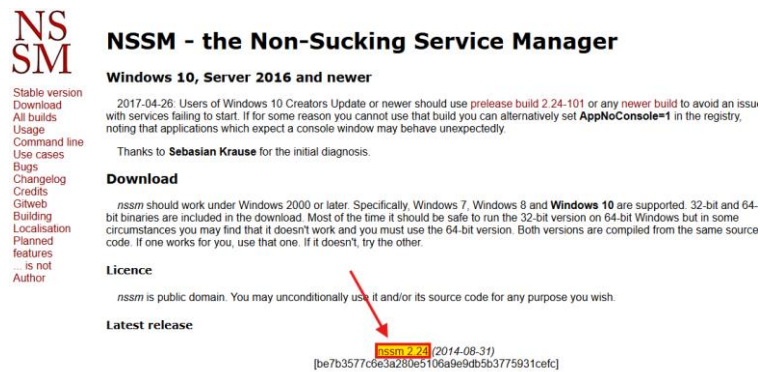


FIGURE 39 NSSM SETUP 1

- Step 02: Then we unzip the file, we will find two file 'win64' and 'win32' for 64 bit and 32 bit OS, open the correct folder and copy the Path then go to cmd and run this command `cd C:\Users\<user Account>\Downloads\nssm-2.24\nssm-2.24\win64`

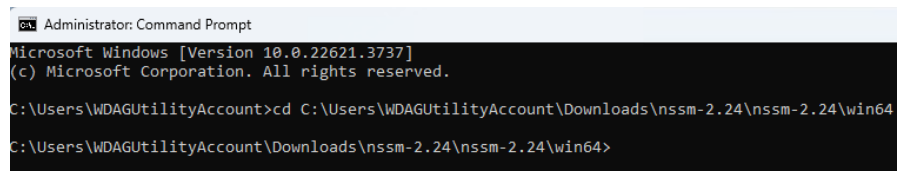


FIGURE 40 NSSM SETUP 2

- Step 03: Next run the command **nssm install** after run it we will see a little window we need to click at the three dots showing in the follow picture:

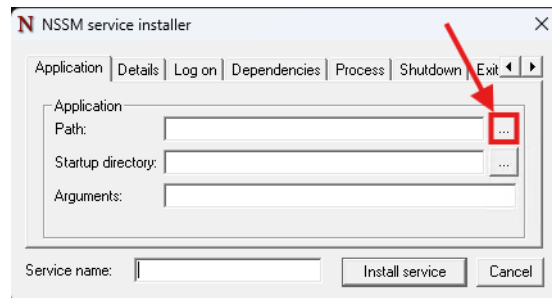


FIGURE 41 NSSM SETUP 3

- Step 04: Then chose InfluxDB folder and click on influxd.exe

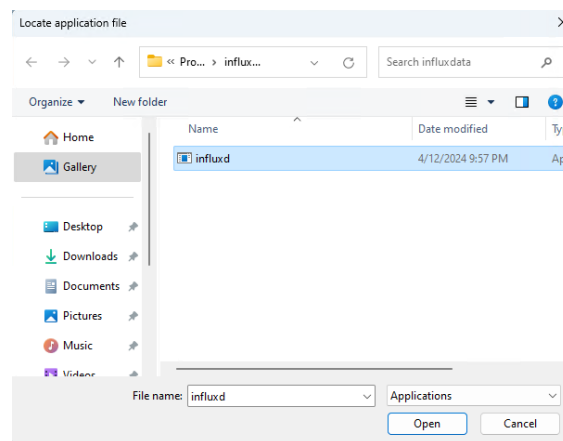


FIGURE 42 NSSM SETUP 4

- Step 05: Click open and chose a service name in my case we name it 'influxd' as the following:

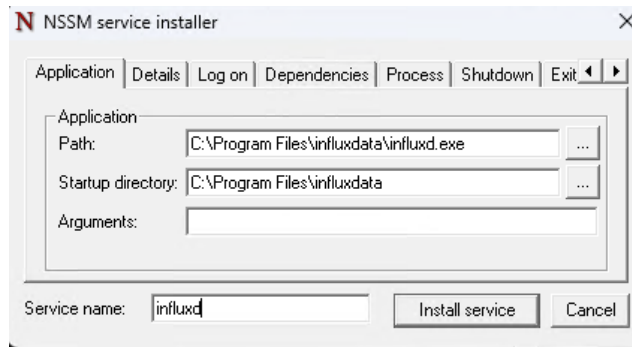


FIGURE 43 NSSM SETUP 5

- Step 06: Just restart the pc and access to InfluxDB from the navigator and we will see that it will run automatically.

2.3.4 Setup Grafana

Grafana is the same thing as InfluxDB we could use it locally or using cloud but we will go for locally because is totally free and enough for us because we just want to show our data in the same place, and we can also host our PC server and we could access to it in anywhere.

To install local Grafana we will follow those steps:

- Step 01: We need to go to the official site web of Grafana or just go to this link <https://grafana.com/grafana/download> and chose the correct platform in my case we am using windows

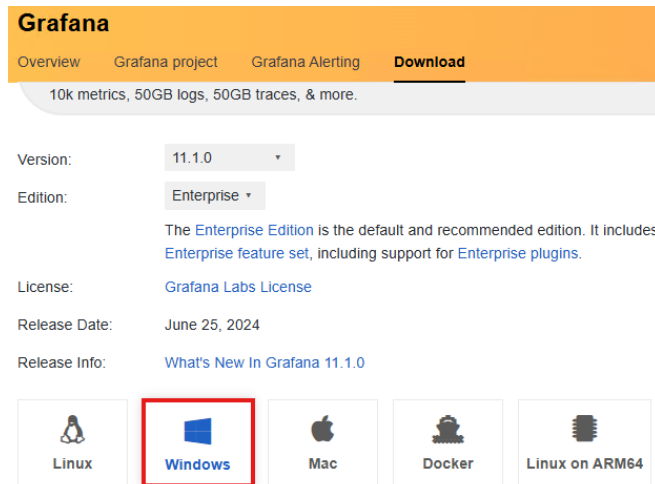


FIGURE 44 GRAFANA SETUP 1

- Step 02: After installing Grafana we double click and setup file and follow the installation steps like installing any program just next, next to finish.

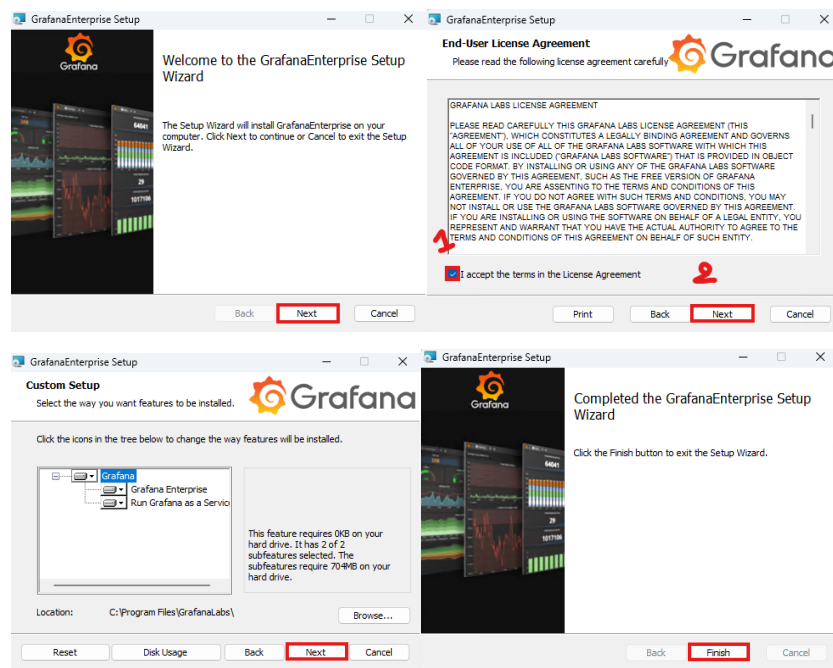


FIGURE 45 GRAFANA SETUP 2

- Step 03: When the installation is done, now we can access to Grafana using any browser and write **localhost:3000** and Grafana interface will showing up, username and password setting in default **admin**.

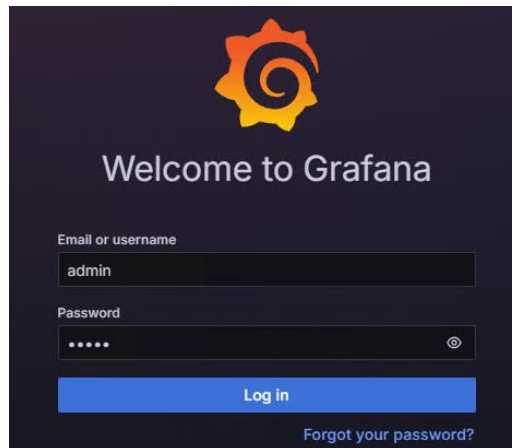


FIGURE 46 GRAFANA SETUP 3

- Step 04: In the first time when we sign In it will recommend we to change the password and use a new password and user name we can set a new password or just skip that and keep the default password.

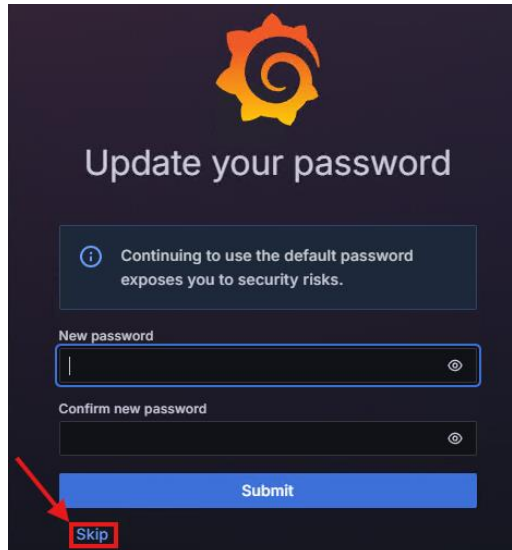


FIGURE 47 GRAFANA SETUP 4

And now Grafana is working in our local machine. Another more step to allow access to Grafana from any device in our local network we need to allow Grafana Through the Firewall:

- Step 05: First open control panel and go to **system and security > Windows Defender Firewall > Allow an app or Feature through Windows Defender Firewall** .
- Step 06: Click on allow modification and Find Grafana in the list and and check ensure to check both of two boxes **‘Privat’** and **‘Public’**.

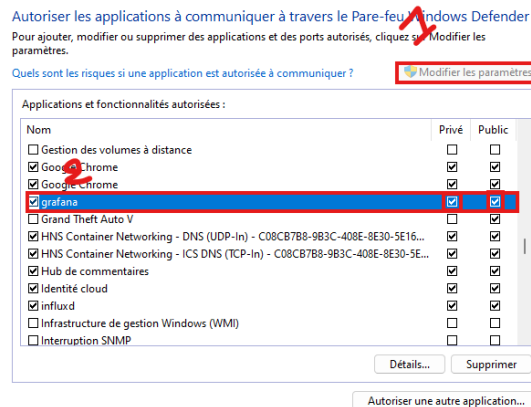


FIGURE 48 GRAFANA SETUP 5

Now we can access to Grafana from any device connected to the same network.

Note: to access to Grafana from our device we don't search for **localhost:3000** because that's for the exact machine we need to access to it by writing our pc IP address by going to router setting and look for our pc or easily we can go to CMD and write this command **ipconfig** in my case it will be **192.168.1.35:3000** we can fix or change our pc address for sure.

```

Carte Ethernet Ethernet 2 :
    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6 de liaison locale. . . . . : fe80::651b:767a:3f23:9b53%10
    Adresse IPv4. . . . . : 192.168.1.35
    Masque de sous-réseau. . . . . : 255.255.255.0
    Passerelle par défaut. . . . . : fe80::1%10
                                     192.168.1.1

Carte Ethernet vEthernet (Default Switch) :
    Suffixe DNS propre à la connexion. . . . :
    Adresse IPv6 de liaison locale. . . . . : fe80::975e:9926:e8b6:aa52%36
    Adresse IPv4. . . . . : 172.19.128.1
    Masque de sous-réseau. . . . . : 255.255.240.0
    Passerelle par défaut. . . . . :
  
```

FIGURE 49 GRAFANA SETUP 6

Conclusion

This chapter has outlined the design and planning considerations for a health monitoring system. It has explored the necessary hardware and software components, detailing their functions and how they work together to achieve the overall goal.

The system leverages an ESP32 microcontroller as the central processing unit due to its communication protocols, Wi-Fi and Bluetooth connectivity, and low power consumption. It then utilizes various sensors to collect health data, including the AD8232 for ECG and the MAX30100 for pulse oximetry. The collected data is transmitted wirelessly to a database, where InfluxDB is chosen for its ease of use, open-source nature, and reliability. Finally, Grafana is employed to visualize the health data in a user-friendly interface, allowing users to access their information remotely.

This chapter also provided a detailed guide on setting up the software components, including Arduino IDE for programming the ESP32, InfluxDB for data storage, NSSM to run InfluxDB as a service, and Grafana for data visualization. By following these steps, users can establish a functional health monitoring system for collecting, processing, storing, and visualizing health data.

Chapitre 3 Implementation and Result

Introduction

A well-designed health monitoring system relies on the seamless interaction of various hardware and software components. This chapter delves into the crucial processes of testing and connecting these elements to ensure the system functions as intended. We will explore how to establish communication between the microcontroller, responsible for collecting sensor data, and the database, where the data is stored for future use. Additionally, we will navigate the integration of Grafana, a powerful visualization tool, with InfluxDB, the chosen database, to transform raw data into user-friendly graphs and charts. Finally, we will test the functionality of the sensors themselves, verifying their ability to accurately capture health information.

By following the steps outlined in this chapter, we will gain the knowledge and skills necessary to test, connect, and verify the effectiveness of our health monitoring system, paving the way for reliable and informative health data collection.

3.1 Test and connect components

3.1.1 Connect the MCU to Database

First of all, we need to link our ESP32 microcontroller to our Database by access to InfluxDB and then we could find an example show we how we start we will explain those steps:

- Step 01: Go to any browser and write `localhost:8086` enter our password and user name and we are in, then in the right bar go to sources.

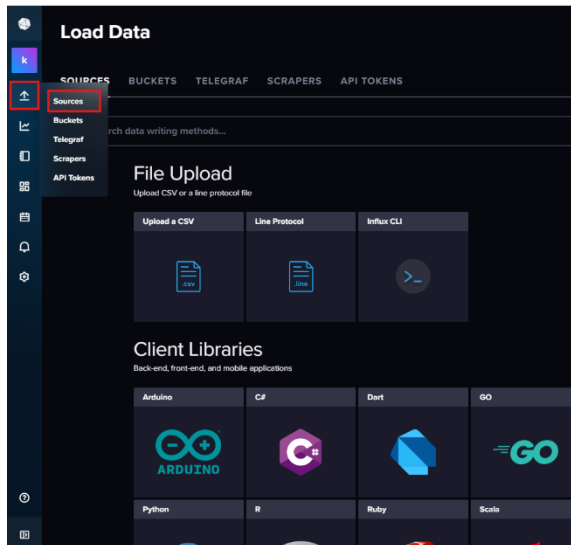


FIGURE 50 INFLUXDB CONFIGURATION 1

- Step 02: We will find a lot of programming language, in my case we will chose Arduino C, after choosing a language we will see some steps we had to follow to quick setup and link our microcontroller to the database, we will installing esp32 and InfluxDB libraries because we did it before in Arduino IDE section. After that we create a bucket to store a data that we want (e.g. a bucket can be for example a name of every patient) and copy the code creating by the database.

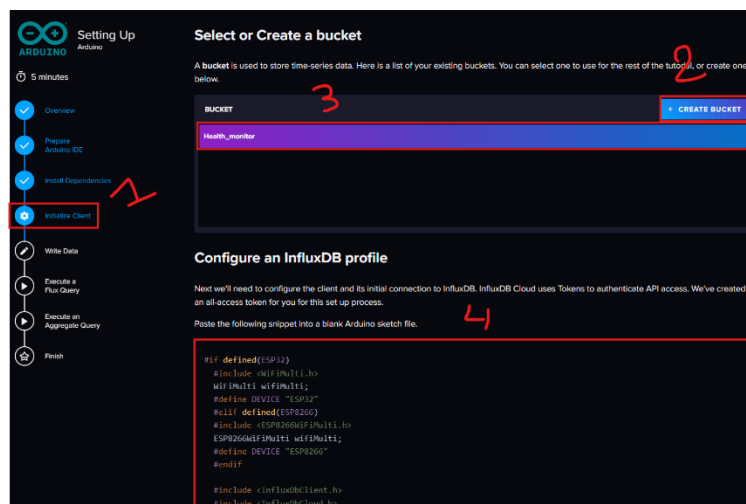


FIGURE 51 INFLUXDB CONFIGURATION 2

- Step 03: Paste the code in the Arduino IDE and change <YOUR_WIFI_SSID> with we WI-FI SSID name, and change <YOUR_WIFI_PASSWORD> with our WI-FI password and the <localhost> with our PC IP address . And upload the code to ESP32.
- Step 04: Open the serial monitor and we must see ‘Connected to **InfluxDB**: <http://192.168.1.35:8086>. that is mean we are connected to the database.

```
COM3
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1184
load:0x40078000,len:13260
load:0x40080400,len:3028
entry 0x400805e4
Connecting to wifi
Syncing time...
Synchronized time: Wed Jun 26 12:42:46 2024
Connected to InfluxDB: http://192.168.1.35:8086
```

Défilement automatique Afficher l'horodatage

FIGURE 52 SYNCHRONIZED TO INFLUXDB

To start writing data, we append lines of code to add tags to the dot at the end of the void setup() function.

For example to write our WI-FI SSID signal strength:

```
void setup() {  
  sensor.addTag("device", DEVICE);  
  sensor.addTag("SSID", WiFi.SSID());}
```

To read the value in serial monitor and send it to the database we will add those lignes in Void Loop() function:

```
void loop() {  
// Clear any previously read sensor data to ensure fresh readings  
  sensor.clearFields();  
// Read the current WiFi signal strength (RSSI)  
  int rssiValue = WiFi.RSSI();
```

```

// Add the RSSI value as a field named "rssi" to the sensor data
sensor.addField("rssi", rssiValue);

// Print a message indicating that data is being written
Serial.print("Writing: ");

// Convert sensor data to InfluxDB line protocol
Serial.println(sensor.toLineProtocol());

// Attempt to connect to the WiFi network
if (wifiMulti.run() != WL_CONNECTED) {
// Informative message if connection is lost}
Serial.println("Wifi connection lost");

// Try writing the sensor data to the InfluxDB database
if (!client.writePoint(sensor)) {
Serial.print("InfluxDB write failed: ");

// Print detailed error message for debugging
Serial.println(client.getLastErrorMessage());}

// Print a message indicating a one-second delay
Serial.println("Waiting 1 second");

// Introduce a one-second delay to control the frequency of data collection
and transmission
delay(1000);}

```

Next to visualize our data we need to follow this steps:

- Step 01: First of all go to the left strip and click on **dashboard** then **+CREATE DASHBOARD**.

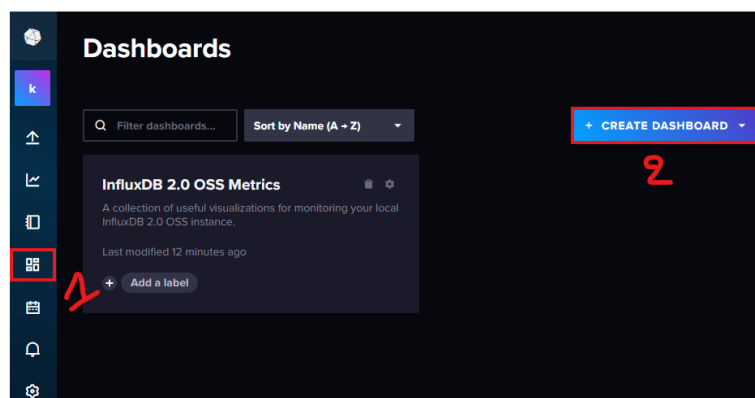


FIGURE 53 CREATE DASHBOARD IN INFLUXDB

- Step 02: The click on **ADD CELL**.

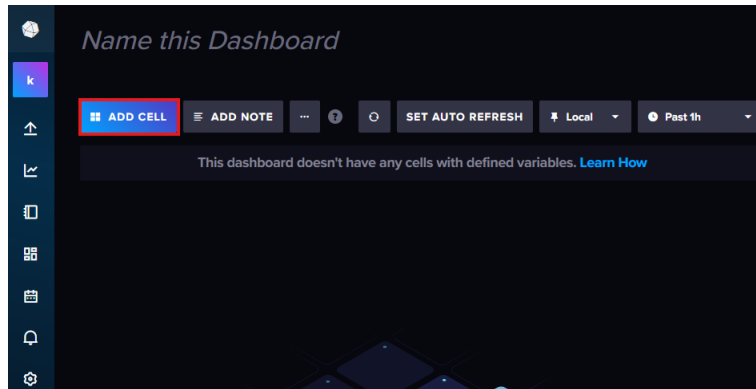


FIGURE 54 ADD CELL IN INFLUXDB

- Step 03: Chose **wifi_state** (we create this Bucket before) then **wifi-status** and our WI-FI SSID (in my case it is **D-Link2750U**) > **rss1** > **ESP32**

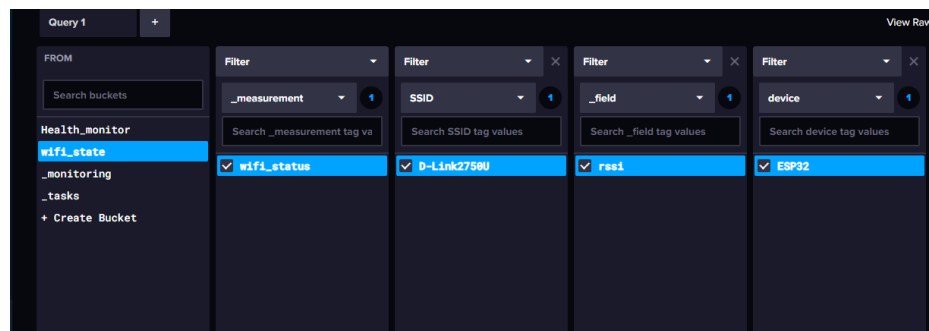


FIGURE 55 READ WIFI_STATE IN INFLUXDB

- Step 04: Finally click **SUBMIT** and the check sign, now we create our dashboard and we can monitor our Wi-Fi signal strength.

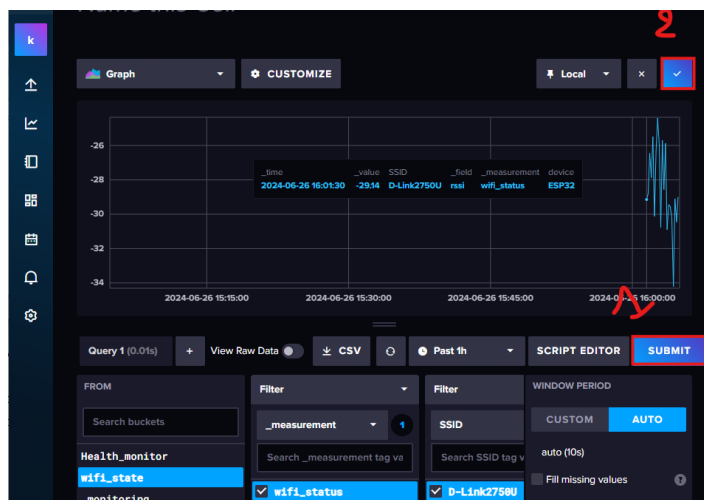


FIGURE 56 INFLUXDB CONFIGURATION 3

3.1.2 Connect InfluxDB Grafana

In InfluxDB we had refresh our data each time we want to see the new data because it is a database designed to store data not to visualize them, that is why we need Grafana. To link them we need to follow these steps:

- Step 01: Firstly we open any navigator and search for **localhost:3000** and enter our password for access to Grafana then in the left strip go to **Home > Menu > Connections > Data sources** and add new data source.

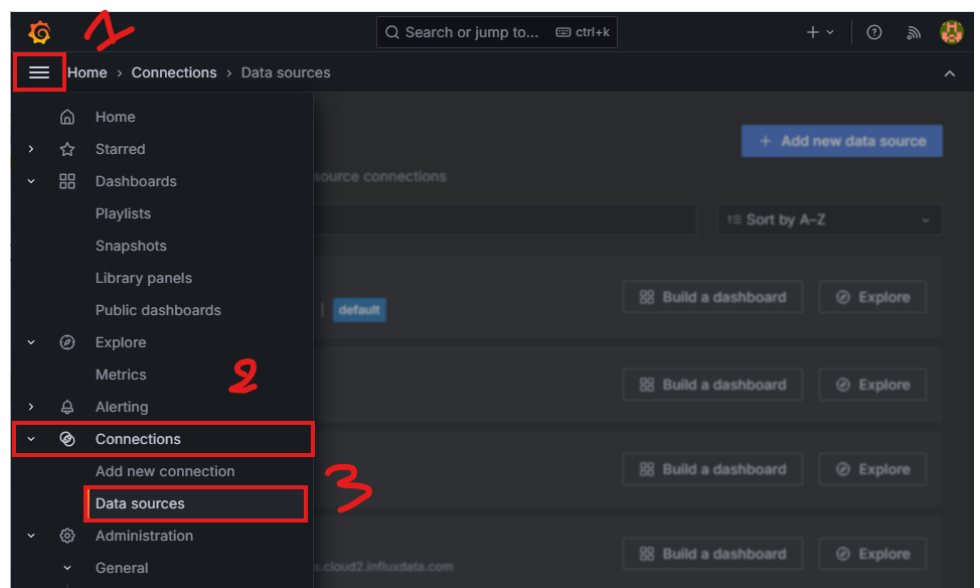


FIGURE 57 ADD DATA SOURCES IN GRAFANA

- Step 02: You will see a various Database suggestion search and chose InfluxDB.

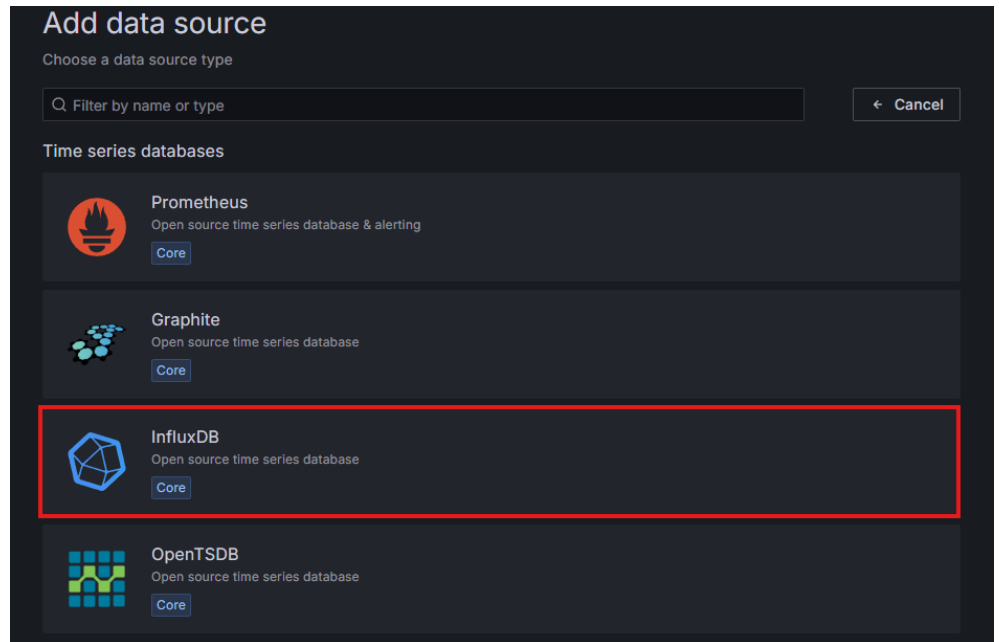


FIGURE 58 SELECT DATABASE IN GRAFANA

- Step 03: Then to set this new data source set the Query language to **Flux** and in the field of URL write <http://localhost:8086/>.

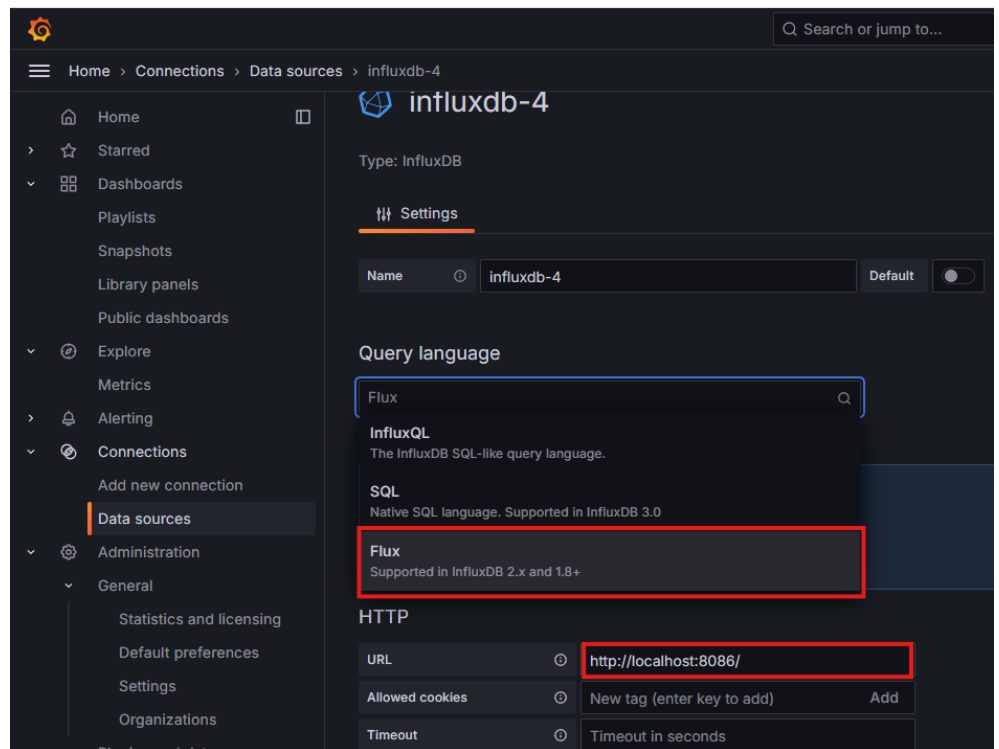


FIGURE 59 GRAFANA CONFIGURATION 1

- Step 04: Under that we unabled Basic auth and enter our Database **User** and **Password**, for the InfluxDB Details we enter our organization database name and the API Token (if we lost it we can create a new one), then in the Default Bucket we write what we want to visualize in this case we chose **wifi_state** Bucket, **click Save & test** and we are done.

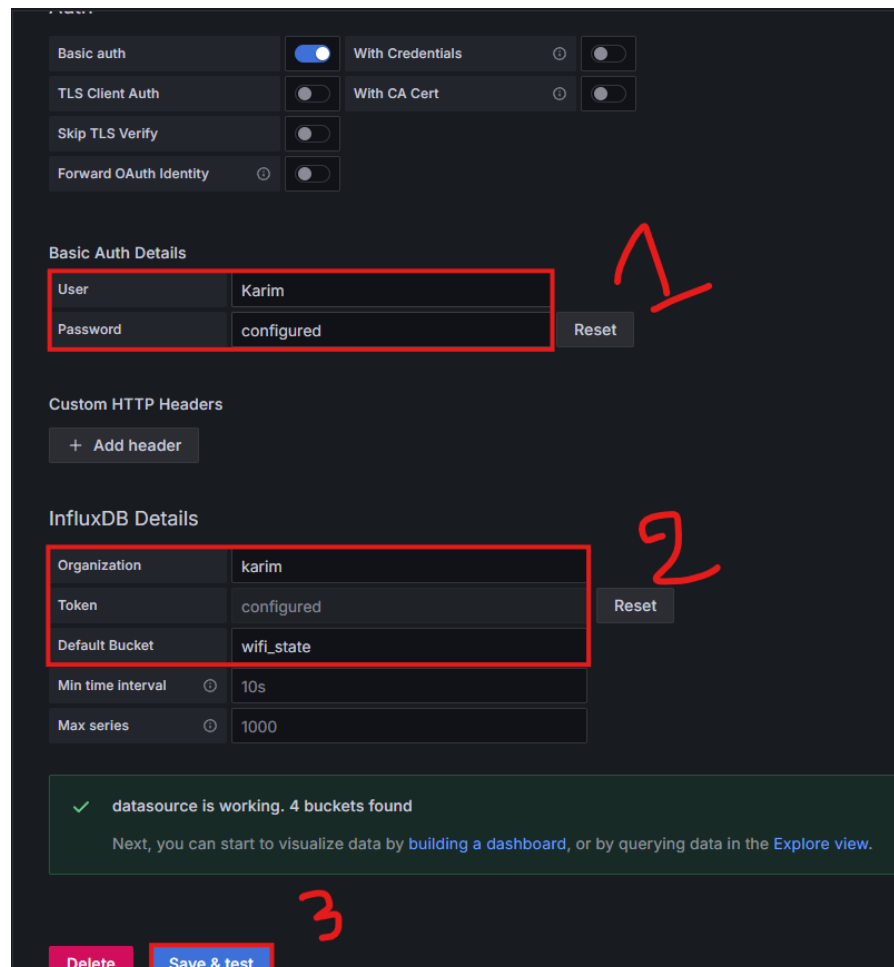


FIGURE 60 GRAFANA CONFIGURATION 2

- Step 05: Secondly, in the left strip again go to **Home > Menu > Dashboards** and click **+Add visualization**

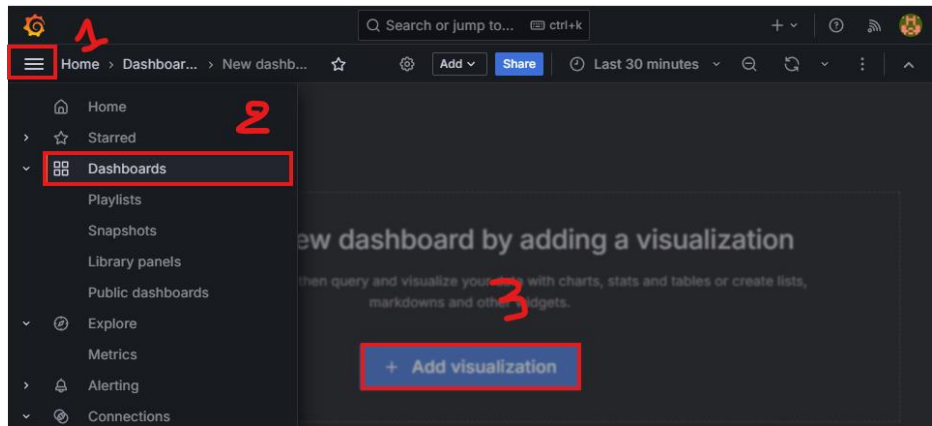


FIGURE 61 ADD DASHBOARD IN GRAFANA

- Step 06: Chose the Data source that we add it before and hold Enter.

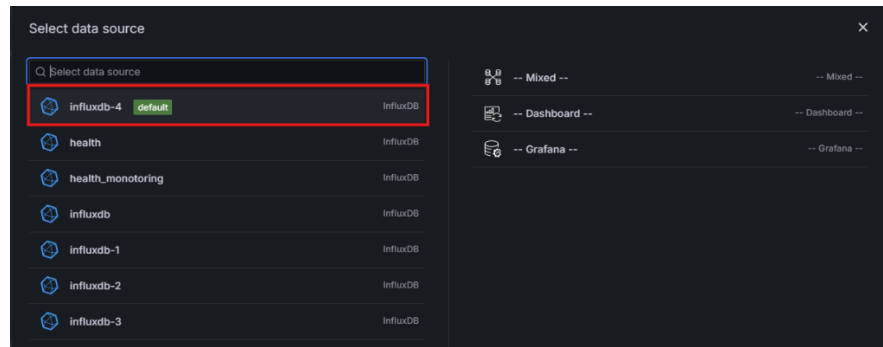


FIGURE 62 GRAFANA CONFIGURATION 3

- Step 07: The good thing in InfluxDB that it is generate the Flux Code automatically so we need go back to InfluxDB and chose our Dashboard and click **SCRIPT EDITOR**.

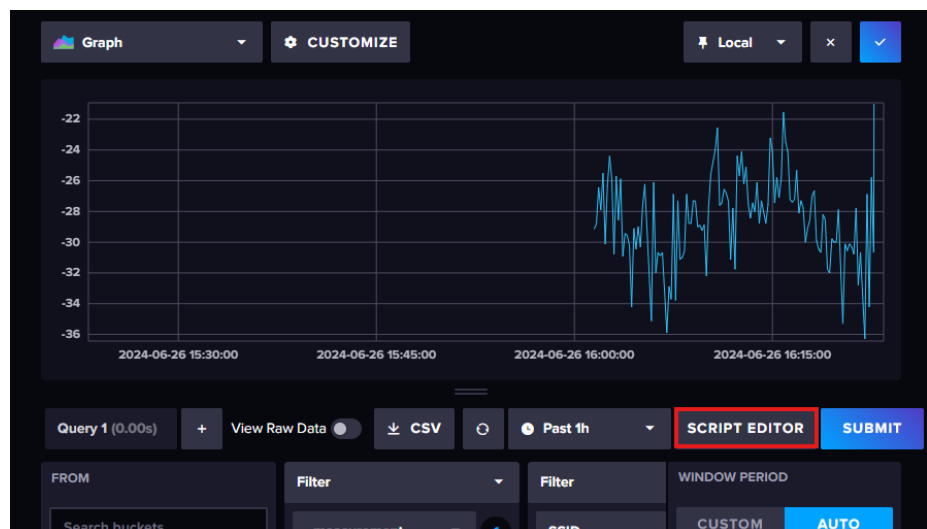


FIGURE 63 COPY SCRIPT FROM INFLUXDB 1

- Step 08: You can see the Flux code, we can copy it and paste it in Grafana script.

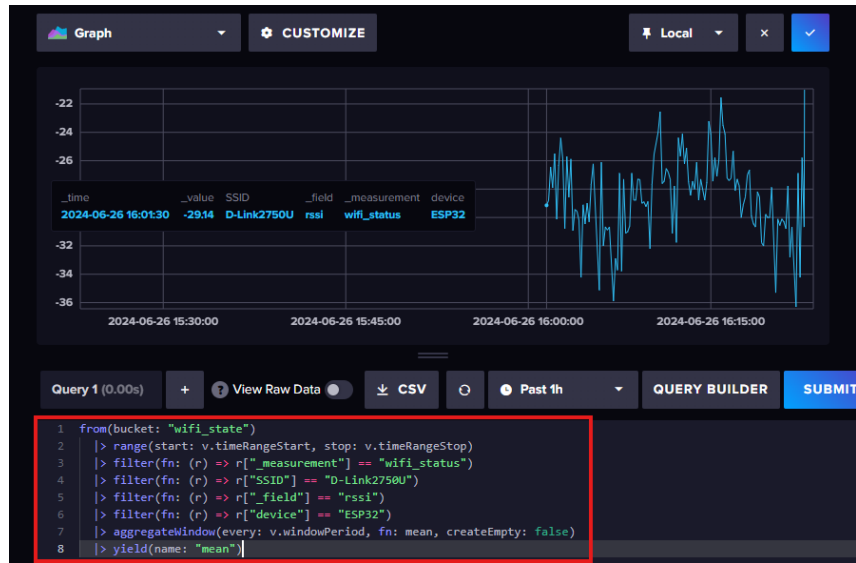


FIGURE 64 COPY SCRIPT FROM INFLUXDB 2

- Step 09: Go back to Grafana and paste the Flux Code in the script and click apply.

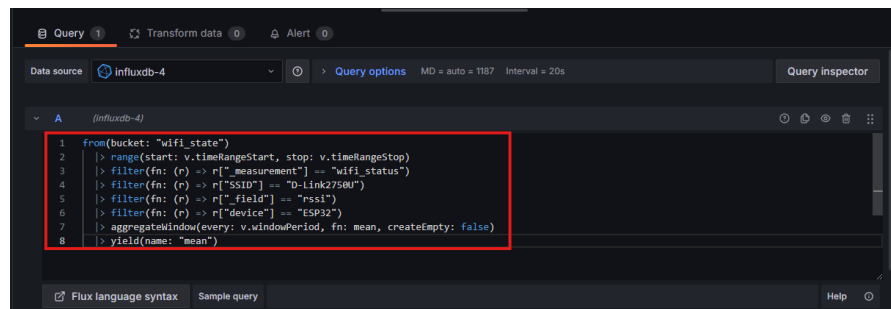


FIGURE 65 PAST SCRIPT IN GRAFANA

- Step 10: Finally save the Dashboard and chose a name for it, now we can visualize our Wi-Fi signal strength by the name in Graph or table...etc



FIGURE 66 GRAFANA VISUALISATION

3.1.3 Test the sensors

a AD8232

First we connect the sensor to the ESP32 and wiring the pin to each other

Then we write the following code that make us read data from the sensor using the exponential Filter

```

sketch_jun26b | Arduino 1.8.19 (Windows Store 1.8.57.0)
Fichier Édition Croquis Outils Aide
sketch_jun26b $
const int LOPlus = 32; // LO+ pin
const int LOMinus = 33; // LO- pin
const int OutputPin = 34; // Analog output from AD8232
const int SDNPin = 35; // Shutdown pin

float a = 0.9; // Smoothing constant
float yk_1 = 0; // Previous y(k-1) value

void setup() {
  Serial.begin(9600);

  // Setup pin modes
  pinMode(LOPlus, INPUT);
  pinMode(LOMinus, INPUT);
  pinMode(SDNPin, OUTPUT);

  digitalWrite(SDNPin, LOW); // Enable the AD8232 (SDN low)
}

void loop() {
  if (digitalRead(LOPlus) == 1 || digitalRead(LOMinus) == 1) {
    Serial.println('!'); // Leads off detection
  } else {
    float xk = analogRead(OutputPin);
    float yk = a * yk_1 + (1 - a) * xk; // Apply smoothing filter

    Serial.println(yk); // Send smoothed value

    yk_1 = yk; // Update previous y(k-1) value
  }

  delay(1); // Small delay to avoid serial saturation
}

```

After uploading the code to our MCU, open the **Serial Plotter** and we must get this result :

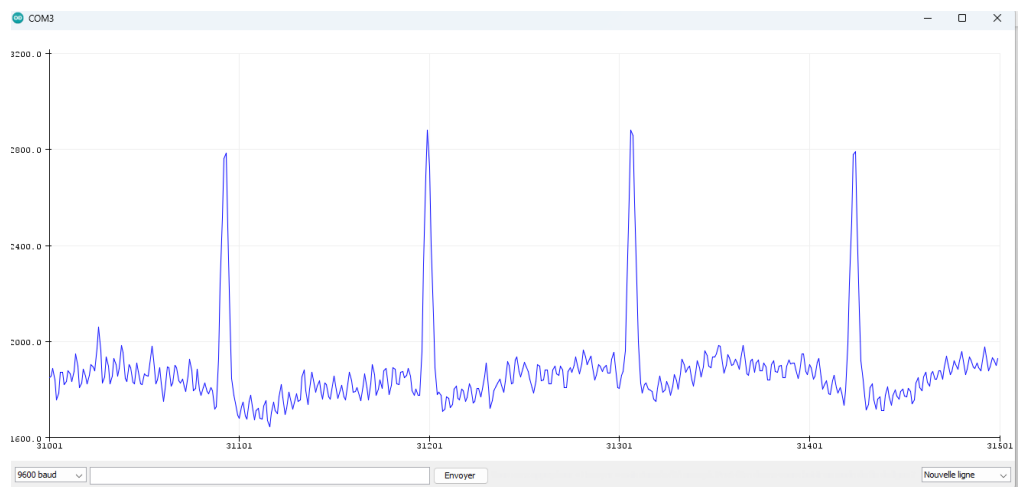



FIGURE 67 ECG SIGNAL OUTPUT

That is mean our sensor is working we just need to use a better wires to get better signal and we could use AI in the future to process this signal and get the data directly.

b MAX30100

We rite this code that is provide us the reading data of the **Spo2** and the **Heart Rate**:



```
max30100 | Arduino 1.8.19 (Windows Store 1.8.57.0)
Fichier Édition Croquis Outils Aide

max30100
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define SDA_PIN 25
#define SCL_PIN 26
#define REPORTING_PERIOD_MS 1000
#define SMOOTHING_WINDOW 5

PulseOximeter pox;
uint32_t tsLastReport = 0;

float heartRates[SMOOTHING_WINDOW] = {0};
float spO2s[SMOOTHING_WINDOW] = {0};
int currentIndex = 0;

void onBeatDetected()
{
    Serial.println("Beat detected!");
}

void setup()
{
    Serial.begin(115200);
    Serial.println("Initializing Pulse Oximeter...");

    // Initialize I2C communication with custom pins
    Wire.begin(SDA_PIN, SCL_PIN);

    // Initialize the PulseOximeter instance
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }

    // Adjust sensor settings if necessary
    pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA); // Adjust current bas

    // Register a callback for the beat detection
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

float average(float *array, int size) {
    float sum = 0;
```

```

float average(float *array, int size) {
    float sum = 0;
    for (int i = 0; i < size; i++) {
        sum += array[i];
    }
    return sum / size;
}

void loop()
{
    // Make sure to call update as fast as possible
    pox.update();

    // Print the read values once every second
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        heartRates[currentIndex] = pox.getHeartRate();
        spO2s[currentIndex] = pox.getSpO2();
        currentIndex = (currentIndex + 1) % SMOOTHING_WINDOW;

        float avgHeartRate = average(heartRates, SMOOTHING_WINDOW);
        float avgSpO2 = average(spO2s, SMOOTHING_WINDOW);

        if (avgHeartRate < 30.0 || avgHeartRate > 200.0) {
            avgHeartRate = 0.0; // Ignore unrealistic heart rate reading
        }
        if (avgSpO2 < 70.0 || avgSpO2 > 100.0) {
            avgSpO2 = 0.0; // Ignore unrealistic SpO2 readings
        }
        Serial.print("Heart rate:");
        Serial.print(avgHeartRate);
        Serial.print("bpm / SpO2:");
        Serial.print(avgSpO2);
        Serial.println("%");

        tsLastReport = millis();
    }
}

```

After wiring the sensor to the MCU and uploading this code, open the serial monitor to read the data of our **Heart Rate** and **SpO2** (Oxygen in blood)

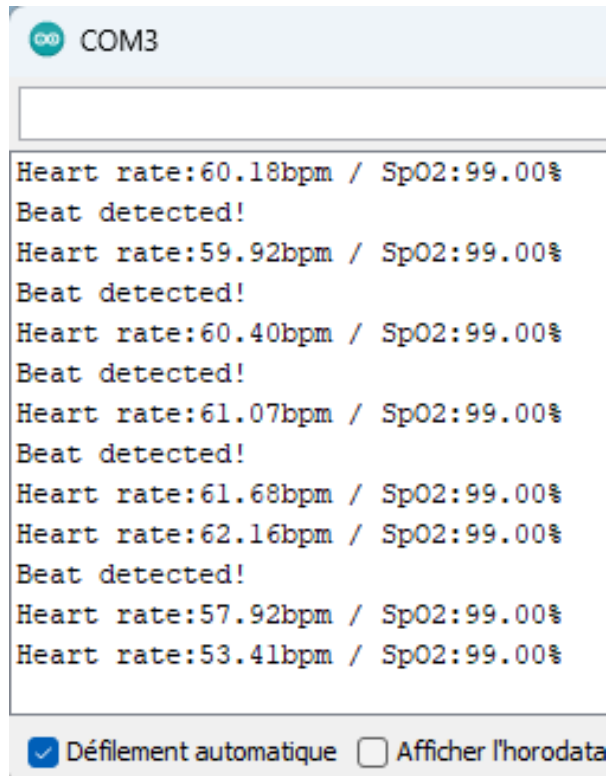


FIGURE 68 HEART RATE AND SPO2 RESULT

These results match the standards for measuring depends to age, and this means that our sensor is working well.

3.2 Result and interface

Upgrade the code :

Now to send data to our database, we follow the steps we talked about previously

And adding those line to the code:

```
sensor.addField("heart_rate", avgHeartRate);
```

```
sensor.addField("spO2", avgSpO2);
```

for the MAX30100 sensor, and this line for the AD8232 Sensor:

```
sensor.addField("smoothed_value", average);
```

Those two line allow to send data to the Bucket in the database.

Wiring :

After following the previous schematics we apply them into our sensors:

This montage for the AD8232:

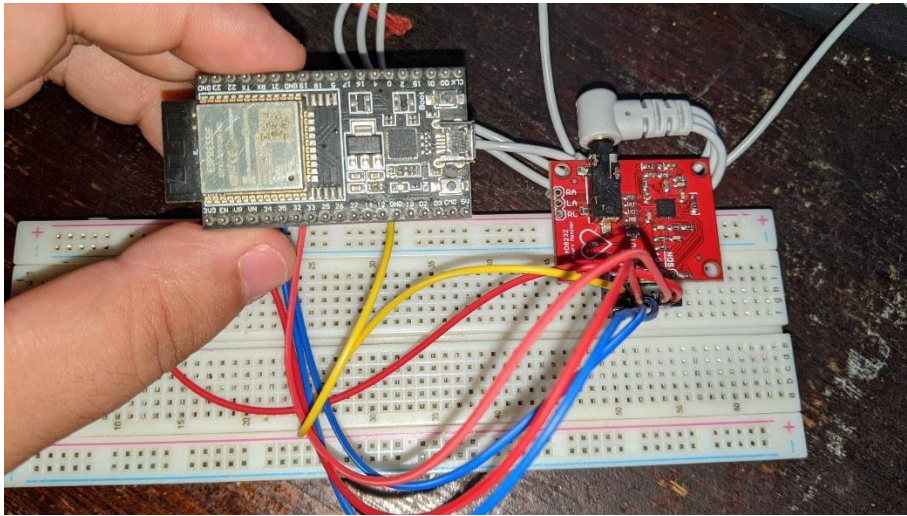


FIGURE 69 ESP32 CONNECTED TO AD8232

And this montage for the MAX30100:

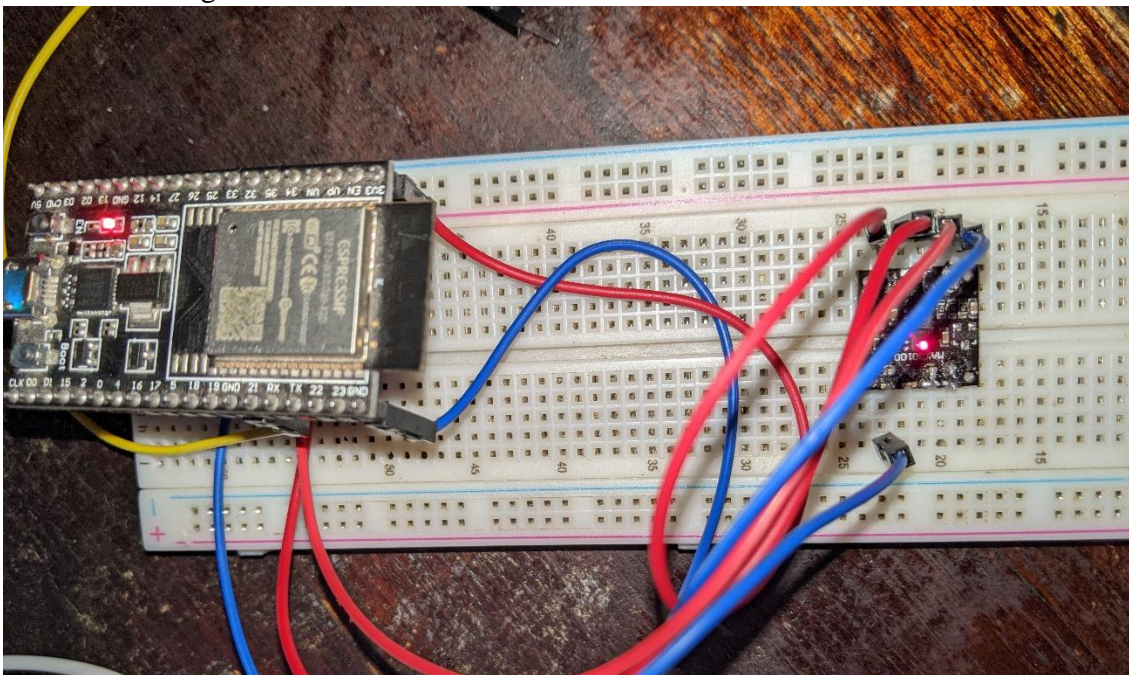


FIGURE 70 ESP32 CONNECTED TO MAX30100

Result:

Finally we access to our database and add a new visualization as we talk in the Chapter 2,

after that we go to Grafana and add a new dashboard and select the Data we want to read them:



FIGURE 71 AD8232 SIGNAL IN GRAFANA

Conclusion

This chapter has been an insightful exploration of the health monitoring system's capabilities. We've witnessed the culmination of the project's efforts through the presentation of the actual results. Data visualizations generated by Grafana, based on sensor readings stored in InfluxDB, provide a clear picture of the system's performance. By examining these visualizations, we gain valuable insights into the user's health status. In essence, this chapter has brought the health monitoring system to life. We've seen the tangible results it produces and gained a deeper appreciation for the remarkable technology that underpins these measurements.

General Conclusion

This thesis explores the development of a real-time health monitoring system using the ESP32 microcontroller, a versatile component in IoT applications. We also talked about how to use and program this controller and its operating environment, in addition to some health monitoring sensors. We talked about the AD8232, its operation, how it works, and the results it gives. The same thing applies to the MAX30100 sensor for measuring blood oxygen levels and heartbeats.

We also touched on the types of databases and their uses. We used the InfluxDB database and learned how to upload it to the local computer and how it works. We also talked about how to transfer data from the controller to the database and display content in grafana, which also taught us how to install and use it and the accurate and smooth results it displays.

Finally, we would like to point out that this project is developable in every way, as it has a good structure and powerful software capable of carrying a lot of data and processing it at a glance.

References

<https://www.influxdata.com/>

<https://grafana.com/>

<https://www.arduino.cc/>

<https://www.espressif.com>

<https://andprof.com/tools/what-is-arduino-software-ide-and-how-use-it/>

<https://pixabay.com/vectors/network-iot-internet-of-things-782707/>

<https://www.baianat.com/articles/how-iot-is-transforming-industries>

<https://questdb.io/glossary/relational-database/>

<https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases>

<https://www.pearsonitcertification.com/articles/article.aspx?p=3004582&seqNum=4>

<https://www.analog.com/en/resources/technical-articles/how-to-design-a-better-pulse-oximeter.html>

<https://www.health.harvard.edu/heart-health/what-your-heart-rate-is-telling-you>