

**Program: ESE 4009\_1**

**INSTRUCTOR:** Prof**.** Mike Aleshams

# Group: 2

|  |  |  |
| --- | --- | --- |
| **Student Name** | **Student ID** | **Signature\*** |
| Christy Rachel Philip | C0765535 | CHRISTY RACHEL PHILIP |
| Fahad Rahman | C0769871 | FAHAD RAHMAN |
| James M Chacko | C0777192 | JAMES M CHACKO |
| Premil Prasannan | C0777191 | PREMIL PRASANNAN |

*\*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties.*

**Health monitoring headphone using IoT**

**Description of the latest similar system**

Humans now are more cautious about their health. People nowadays go to hospitals for a full-body check-up regularly. For instance, there have been many incidents of accidents in Canada because of a lack of health monitoring. These incidents lead to the development of specific portable health monitoring devices that people can use daily. There are Apple Watch, Fitbit, Samsung watch for these purposes. With the help of IoT, data is transmitted to the user's smartphone, and all the health data can be analysed within the phone using a customised application.

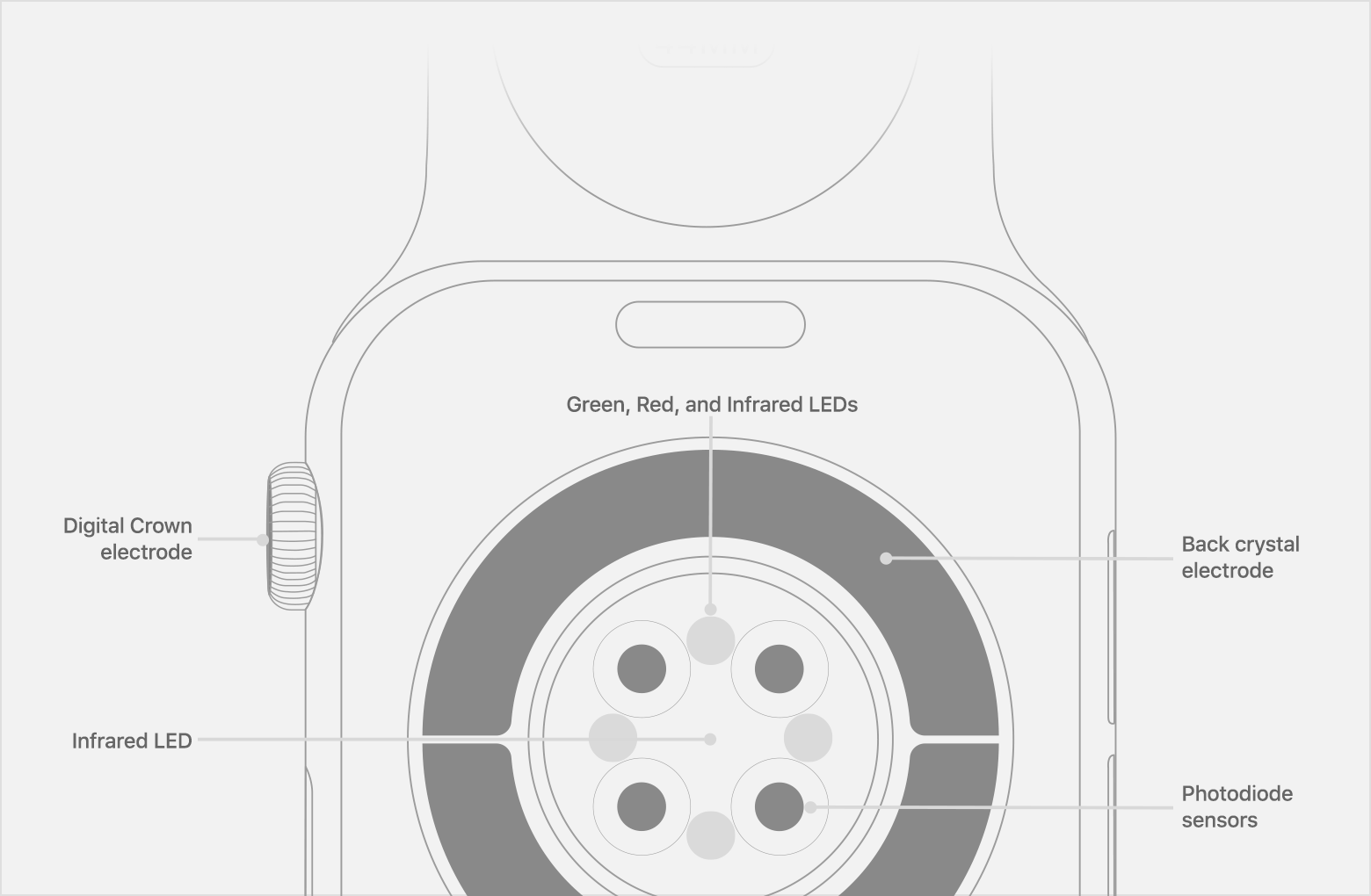


Figure 1:Existing system Diagram

The existing product is attached to the wrist like a watch. The product consists of a heart rate sensor that includes an Infrared sensor, UV sensor and photodiode to detect the heart rate. Apple explains on its website that it uses what is known as “photoplethysmography.” “This technology, while difficult to pronounce, is based on a simple fact: Blood is red because it reflects red light and absorbs green light.” Apple Watch uses green LED lights, paired with light-sensitive photodiodes, to detect the amount of blood flowing through your wrist. “When your heart beats, the blood flow in your wrist — and the green light absorption — is greater” (Saltzman, 2021). The new Series 5 runs on a new Apple S5 chip inside: 64-bit dual-core S5 processor, up to 2x faster than the S3 processor (includes W3 wireless chip).

**Limitations of the latest similar system:**

Even though a similar system uses advanced technologies, there are few limitations to the system.

* The smartwatch is now on the market at a higher price. The new apple watch series six cost around $600 including Tax
* Blood Oxygen app measurements are not intended for medical use, including self-diagnosis or consultation with a doctor, and are only designed for general fitness and wellness purposes (Apple inc, 2020)
* Apple Watch Series 6 (GPS + Cellular) can use a cellular connection for Emergency SOS. To use Emergency SOS on an Apple Watch without cellular, your iPhone needs to be nearby. If your iPhone isn’t nearby, your Apple Watch needs to be connected to a known Wi-Fi network, and you must set up Wi-Fi Calling (Apple Inc, 2020)
* The watch uses sensors to take measurements from the wrist. The accuracy of these measurements is not that great.
* The product does not have the sensors or features to take the temperature measurement of an individual.
* The specific watch only works with the help of an iPhone
* Low Battery life.

**Initial Solution**

Here we are introducing our product as a solution. This health monitoring device consists of temperature sensors, a body surface electrode, and an SPO2 sensor directly connected to the users' bodies. All the signals from these peripheral devices are transferring to the microcontroller. With the help of IoT, data is transmitted to the user's smartphone, and all the health data can be analyzed within the phone using a customized application (Andrade, 2019).

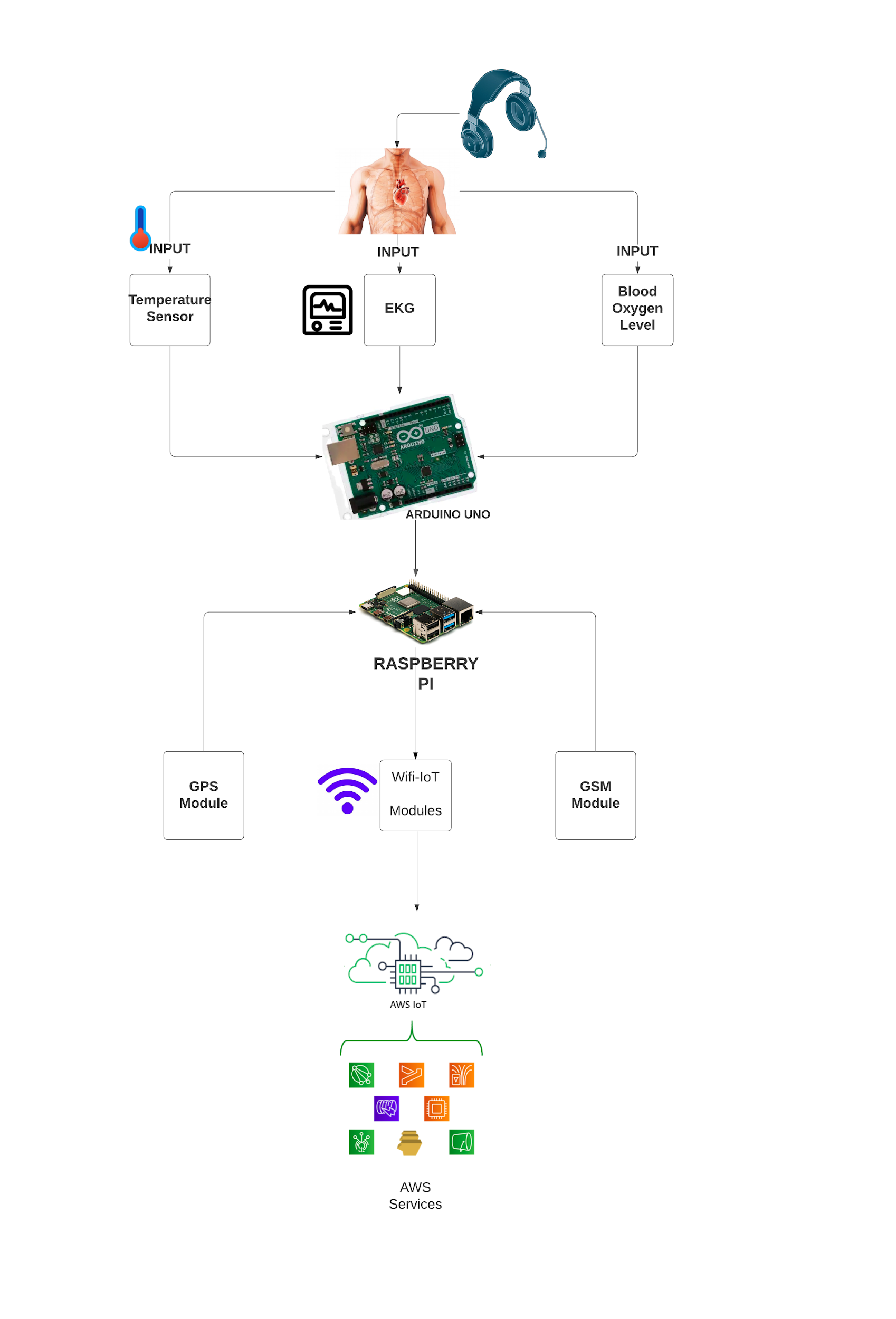
Here the temperature sensor is to check the temperature, the Surface electrode is to measure the EKG, and the SPO2 sensor verifies the blood oxygen level. Accuracy will be better because the sensors are attached to body parts(heart, neck)

Moreover, if the user feels an emergency health failure, an additional switch is added in the design to call 911 by pressing this switch directly. We can also add a facility to automatically identify a patient's emergency condition by itself and call an emergency automatically. In addition to this, since we are adding a GPS module to this, we can access the patient’s location.

As this is connected as part of Bluetooth headphones, it is user-friendly. Along with enjoying music, they can monitor their body temperature, blood oxygen level and EKG.

Above all, the cost of the product is less than one-third of the currently available product. Compatibility with any phone is another great feature of this product.

**Block Diagram**

****

**Hardware and Software Requirement**

**Major Hardware components Required**

Raspberry pi

Arduino UNO

Thermistor (Temperature sensor)

Pulse Oximeter/Spo2 Sensor

Surface Electrode/ECG Sensor

Battery

GSM Module

GPS Module

**Major Software Requirements**

KiCad

Amazon Web Services

Raspbian

**Limitations of Initial Solution**

In the initial solution, we planned to use Raspberry pi as the main processor. But after some deep research, we found that this processor has certain limitations.

Our project is on IoT, and it has a lot of concerns regarding RTOS. Raspberry pi is not the best processor for IoT projects. Raspberry Pi doesn’t have an inbuilt Wifi or Bluetooth module. So we need to connect external modules to the microprocessor. However, combining all these modules may increase the size and weight of the prototype and have size constraints.

Moreover, the Pi does not have a web server of its own. We need to buy a web server like AWS or Azure for this IoT project. One of the other concerns was the lack of ADC and DAC converter. Raspberry pi does not have ADC and DAC converter, and it may require the use of other slave devices for this purpose. In our initial solution, we needed to use Arduino as a slave device to convert the data from sensors, and they are transferred to the Raspberry pi.

**Alternative solution**

We can use the ESP32 microcontroller instead of Raspberry Pi to resolve this issue, which will help eliminate these problems up to certain limits.

ESP32 is a popular low-power system on chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth, making it suitable for designing and prototyping IoT solutions.

ESP32 also has its own webserver. So we don’t have to use AWS or other cloud services for IoT.

**Major features of ESP32**

* **Wifi and Bluetooth:** This one has an inbuilt wifi module, which helps to avoid an additional wifi module (Explore Embedded, n.d.)
* **Web Server:** As ESP32 has its web server, we don’t need other web servers like AWS or Microsoft Azure. (Explore Embedded, n.d.)
* **Low power consumption:** TheESP32-S2 co-processor is based on the RISC-V architecture. The power consumption is much lower. The ULP co-processor is active when the CPU is disabled in sleep modes and consumes a lot less power. (Explore Embedded, n.d.)
* **Cost-effective:** The cost of this microcontroller is only $14.27, which makes it more cost-effective than Raspberry pi, which costs about $65.
* **ADC converter:** The ESP32 integrates two 12-bit SAR (Successive Approximation Register) ADCs, supporting a total of 18 measurement channels. (Explore Embedded, n.d.)

**Includes FreeRTOS:** The Espressif Internet Development Framework (ESP-IDF) uses FreeRTOS to use the two high-speed processors better and manage the numerous built-in peripherals. It is done by creating tasks. It Includes a tickless mode for low-power applications. RTOS objects (tasks, queues, semaphores, software timers, mutexes and event groups) can be created using either dynamically or statically allocated RAM.

Many manufacturers produce SoC with freeRTOS support. Expressif included freeRTOS in its latest version ESP – IDF. There are currently two methods to programming the ESP32: the ESP-IDF and the ESP32 Arduino Core. Espressif IoT Development Framework is a set of open-source libraries and tools to facilitate the deployment of apps to ESP32s FreeRTOS.The Espressif Internet Development Framework (ESP-IDF) uses FreeRTOS to make better use of the two high-speed processors and manage the numerous built-in peripherals. It is done by creating tasks (Kumar, 2017)

*esp\_system.h*: This inclusion configures the peripherals in the ESP system. Think of it as system initialization. It's like setting up all the components of your bike, before you could fire the engine (Explore Embedded, n.d.)

*freertos/FreeRTOS.h*: Include configuration to run freeRTOS on ESP32.

When your task is in the running state and performing its action, it is said to be in its Active State. When your task is not running or waiting for an event to occur, it is said to be in its Blocked State. What this means is, if there are 2 functions that both need to perform periodic tasks only one function can run at a time. This means that one function would be in the running state and the other would be in the blocked state. The components directory holds all the 'C' code for the ESP32. It contains all the components that make up the ESP32. It includes Drivers for numerous peripherals, the bootloader, bt(Bluetooth), freeRTOS. (Espressif, n.d.)

In our project, there will be four major tasks. At the same time, we need to take temperature, Heart rate and blood oxygen level. So these tasks will be performed at the same time and Esp32 supports multitasking. Moreover, the highest priority will be given to sending notifications. That is when the value from the sensor goes beyond a certain level the highest priority will be given to send alert messages and notifications.

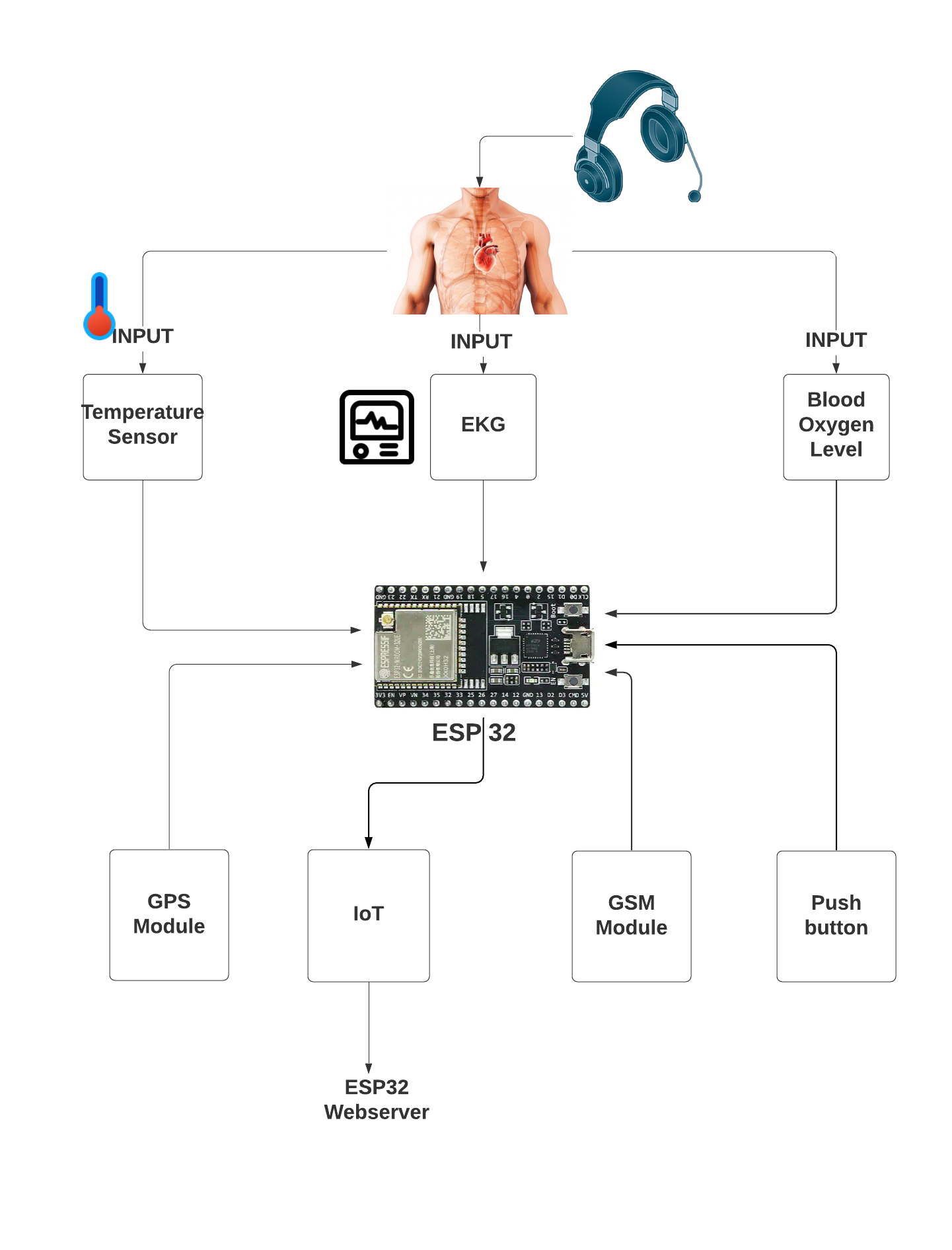
**Limitation Of ESP32**

Being a powerful 32-bit microcontroller with integrated Wi-Fi, full TCP/IP stack for internet connection, and Bluetooth ESP32 have some limitations.

There is insufficient processing power for fast WiFi or CPU-intensive activities while using the WiFi, too few GPIOs, one ADC pin, and no onboard USB. The ESP8266 does not provide any means to secure the code or the data it stores. Anyone with physical access to the device can read your more sensitive information, such as Wi-Fi credentials. It also has less SRAM and ROM, but again it supports more prominent external memories.

**FINAL SOLUTION**

**Block Diagram**

****

**FEATURES**

* **Hardware Changes**

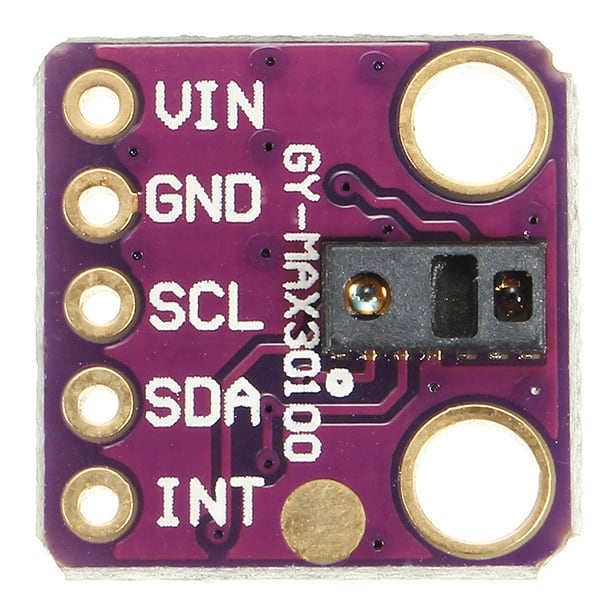
**Processor**

In the initial solution, we proposed using Raspberry pi as the central processing unit. But, we changed it to ESP32 because of its properties.ESP32 is a low cost, low size processor that has better features than Raspberry pi. It has an inbuilt wifi module which makes it a better processor for IoT projects.

**Surface Electrodes**

In the proposed plan, we planned to use the surface electrodes for taking the ECG. Most of the patients don't prefer surface electrode to be connecting to their bodies. These electrodes may affect the Aesthetics of the prototype. It will be uncomfortable to people if they connect these electrodes during their day to day life. Adding these electrodes may increase the size of our project.

The solution to this was to take ECG using a sensor. When we did some research, we came across the Max30100 Pulse Oximetry sensor and Heart rate sensor. This sensor can be used for blood oxygen and ECG and does not need any extra sensor pads or surface electrodes to find heart rate. (Maxim Integrated, 2014)



**Software Change**

ESP32 has its own built-in web server and wifi modules. We can access the ESP32 web server by typing the ESP32 IP address on a browser in the local network. We can create the server by coding in the Arduino IDE. (Last Minute Engineers, n.d.)

**Hardware and Software Requirement**

* **Major Hardware components Required**

ESP32

Thermistor (Temperature sensor)

Max30100 Pulse Oximetry sensor

Heart rate sensor

Battery

GSM Module

GPS Module

Push button

Power supply

* **Major Software Requirements**

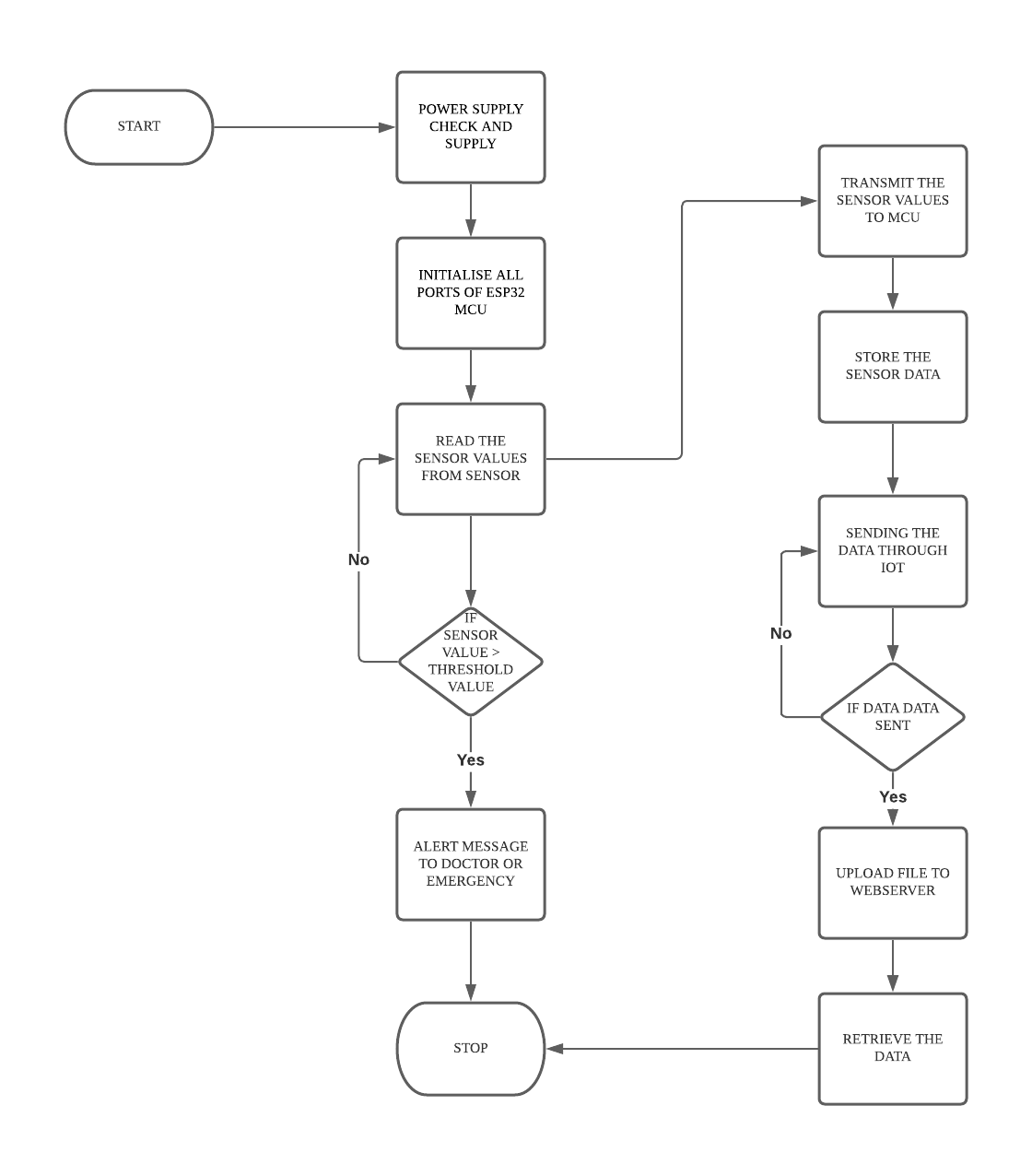
KiCad

ESP32 Web server

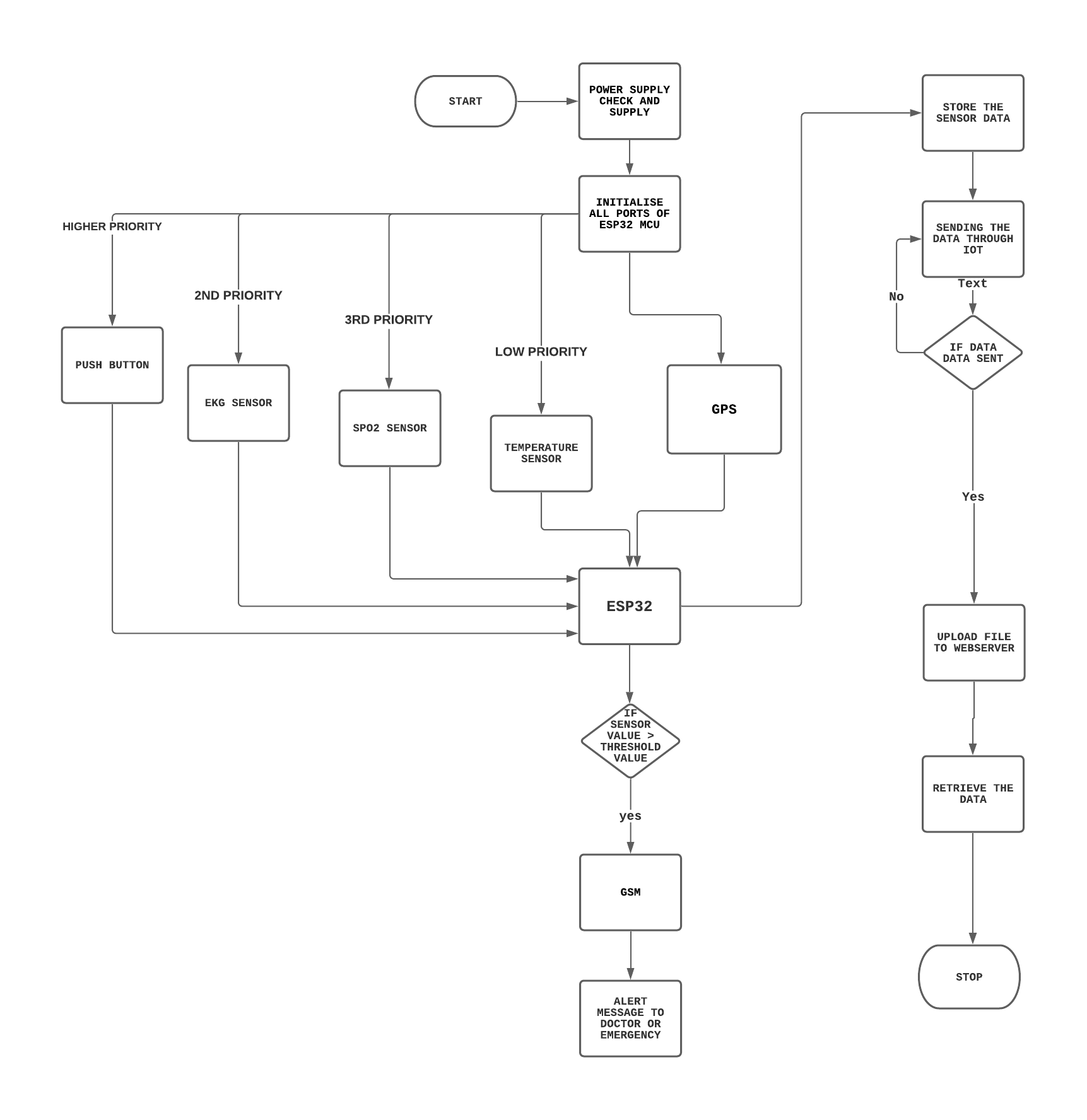
Arduino IDE

C/C++(Coding language)

**FLOW CHART**

****

**UPDATED FLOWCHART**

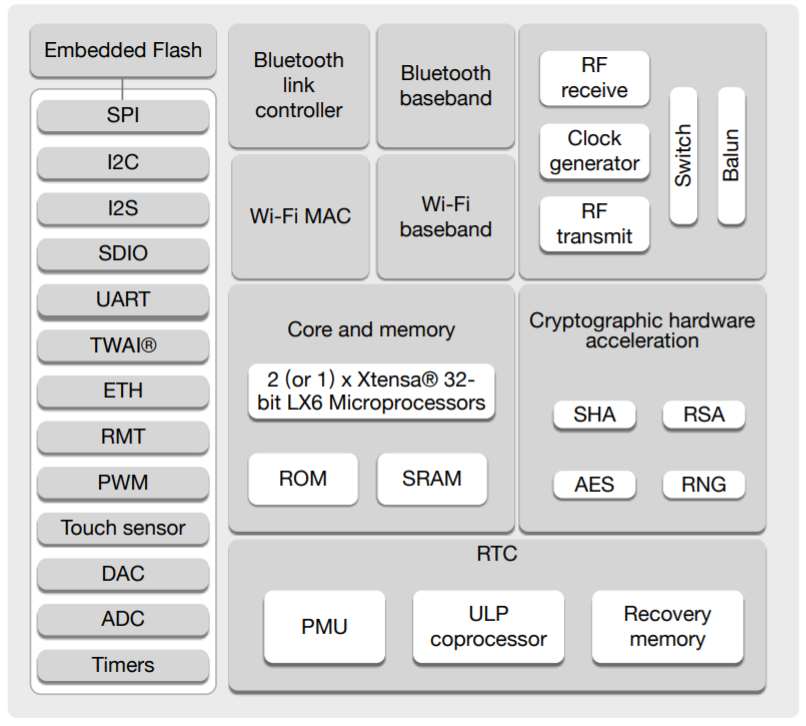


**Milestones (Deliverables and Time Schedule)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start Date** | **End Date** | **Person-In-Charge** |
| Project Proposal | May 10 | June 3 | Group |
| Ordering Components | June 4 | June 7 | James |
| Testing the components | June 8 | June 10 | Christy |
| ESP32 setup | June 11 | June 14 | Fahad |
| Interfacing with GSM | June 15 | June 17 | Premil |
| Interfacing with GPS | June 18 | June 23 | Christy |
| Interfacing with Temperature Sensor | June 24 | June 30 | James |
| Interfacing with EKG Sensor | July 1 | July 7 | Premil |
| Interfacing with SPO2 Sensor | July 8 | July 12 | Fahad |
| Interfacing with Push Button | July 13 | July 14 | Christy |
| Real-Time Operation | July 15 | July 19 | James |
| Power Management | July 20 | July 22 | Premil |
| Cloud Storage Interfacing | July 23 | July 26 | Fahad |
| Schematic Capture design | July 27 | July 29 | Premil |
| PCB Layout design | July 28 | August 4 | Fahad |
| Code Integration | August 5 | August 11 | Christy |
| PCB Implementation | August 12 | August 16 | James |
| Project Report | August 18 | August 18 | Group |
| Final Demonstration | August 19 | August 19 | Group |

**Communication and Programming Standards:**

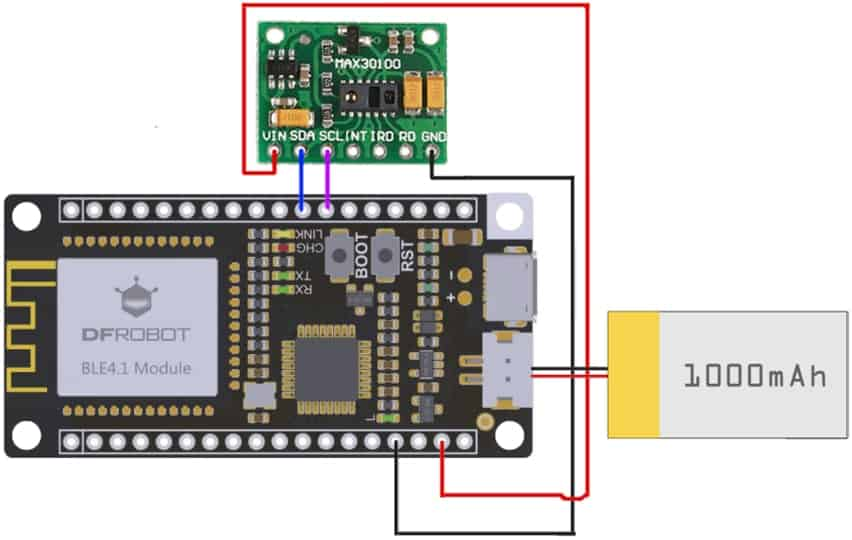
Functional Block Diagram of ESP32

****

The main protocol used in ESP32 are

1. 3 x SPI(Serial Peripheral Interface) (wifi)
2. 2 x I2C
3. 2 X I2S
4. 3 X UART
5. Ethernet MAC interface
6. CAN 2.0
7. IR(TX/RX)

In our project, we are interfacing MAX30100 Pulse Oximeter with ESP32 with the I2C communication protocol. The MAX30100 has I2C Pins. To connect its SDA pin to GPIO21 & SCL pin to GPIO22 of ESP32 Board. The power supply required by MAX30100 is 3.3V. So connect its VCC terminal to 3.3V of ESP32. The circuit diagram and connection is given below.

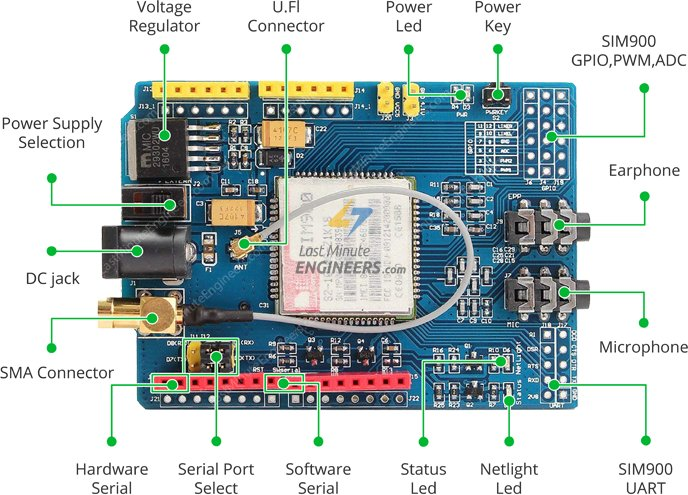


I2C is a very common communication protocol that we are using in our project to interface our MAX30100 with ESP32. The ESP32 has two I2C bus interfaces that can serve as an I2C master or slave. In these cases, the ESP32 is the master chip and the MAX30100 is the slave. I2C means Inter-Integrated circuit, and it is a synchronous, multi-master, multi-slave communication protocol. We can connect,

1. multiple slaves to one master
2. multiple masters controlling the same slave

I2C communication protocol uses two wires to share information. One is used for the clock signal (SCL) and the other is used to send and receive data (SDA). The SDA and SCL lines are active low, so they should be pulled up with resistors. Typical values are 4.7k Ohm for 5V devices and 2.4k Ohm for 3.3V devices. Connecting an I2C device to an ESP32 is normally as simple as connecting GND to GND, SDA to SDA, SCL to SCL and a positive power supply to a peripheral, usually 3.3V.

For GSM and GPS modules we will use UART communication because by using UART we don't need any clock for synchronizing the output. In our project the GSM module is used to alert the user.



Temperature sensor DS18B20 uses one way communication. So we can connect the temperature sensor to the MCU using a GPIO pin and we don't need any special communication protocol.

The ESP32 microcontroller has an inbuilt wifi and we require wifi for IoT.ESP32 implements TCP/IP, full 802.11 b/g/n/e/i WLAN MAC protocol, and Wi-Fi Direct specification. This means ESP 32 can speak to most of the WiFi Routers out there when used in station(client) mode. Also it is able to create an Access point with full 802.11 b/g/n/e/i.

ESP32 also supports Wi-Fi Direct. Wifi-Direct is a good option for peer-to-peer connection without the need of an access point. The Wifi-Direct is easier to set up and the data transfer speeds are much better than bluetooth.

For transferring the data to the cloud we use MQTT(Message Queuing Telemetry Transport). MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc. (MQTT, n.d.)

For programming we use C language and code will follow MISRA coding c guideline (Embedded staff, 2002)

* All codes shall conform to ISO 9899 standard C, with no extensions permitted
* The condition of an if-statement and the condition of an iteration statement shall have boolean type.
* All object and function identifiers shall be declared before use.
* Not to use non-constant pointers to functions.

**Environmental, Legal and Ethical Ramifications**

For every project there will be concern regarding ethical and environmental issue.Our proposed project is based on Iot based Health monitoring.So.lot of ethical and environmental standards should be considered while designing.Major environmental and ethical concerns are listed below

* Our project includes use of more than one sensor and most of the sensors use IR rays for their working.There will be concern about the safety of the product in certain demographics of society.Some consider over radiation of IR rays causes cancer and environmental pollution.
* Since our project is entirely based on Iot we use the internet for its working.So there will be concern regarding the privacy of our data.Hackers might attack and may lead to stealing of our private data.
* The Data is stored on third party storage like Esp32 webserver but when it is produced massively we will require to use Amazon web service or Azure for cloud service.So they may charge higher charges for using their feature and the safety is also a concern while using their products.These companies might sell our public data to other mnc without the users acknowledgement. This could be a legal and ethical issue
* Another issue which may be an ethical issue is that our product is a health monitoring device,So the users may question the accuracy of our product and how precisely it will take our data from the human body.
* Environmental factors could be the use of the material to create the health monitoring device and its design standards.The material used must be environment friendly and must not cause any pollution.
* In this project we can try to make sure that connections are made secure in the proper casing and also that the proper cooling is provided in case our main processor gets hot due to the processing of data.

As the prototype we are designing is based on Health monitoring the data storage will be given high priority.when the project is used in the real world application the safety will be given more priority.The components used are using less power the power consumption can be reduced.

**Bibliography**

Andrade, D. (2019, July 17). *Use a Raspberry Pi to Communicate with Amazon AWS IoT*. hackster.io. Retrieved June 1, 2021, from https://www.hackster.io/dansku/use-a-raspberry-pi-to-communicate-with-amazon-aws-iot-e37525

Apple inc. (2020, December 21). *Apple Watch Series 6*. Technical Specifications. Retrieved June 1, 2021, from https://support.apple.com/kb/SP826?locale=en\_CA

Embedded staff. (2002, Juy 1). *Introduction to MISRA C*. embedded. Retrieved June 1, 2021, from https://www.embedded.com/introduction-to-misra-c/

Espressif. (n.d.). *FreeRTOS*. Overview. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html

Explore Embedded. (n.d.). *Overview of ESP32 features. What do they practically mean?* Explore Embedded. https://www.exploreembedded.com/wiki/Overview\_of\_ESP32\_features.\_What\_do\_they\_practically\_mean%3F

Explre Embedded. (n.d.). *Hello World with ESP32 Explained*. Hello World. https://exploreembedded.com/wiki/Hello\_World\_with\_ESP32\_Explained

Kumar, S. (2017, August 1). *ESP32 : Introduction to FreeRTOS*. ICIRCUIT. Retrieved June 1, 2021, from https://icircuit.net/esp32-introduction-freertos/1297

Last Minute Engineers. (n.d.). *Create A Simple ESP32 Web Server In Arduino IDE*. Last minute Engineers. https://lastminuteengineers.com/creating-esp32-web-server-arduino-ide/

Maxim Integrated. (2014, September 24). *MAX30100*. Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health. Retrieved June 1, 2021, from https://www.maximintegrated.com/en/products/sensors/MAX30100.html

MQTT. (n.d.). *MQTT: The Standard for IoT Messaging*. Why MQTT. https://mqtt.org

Saltzman, M. (2021, June 2021). *How Does Apple Watch’s Heart Rate Monitor Work?* Dummies. https://www.dummies.com/consumer-electronics/wearables/apple-watch/how-does-apple-watchs-heart-rate-monitor-work/