A Project Report

on

IOT BASED HEALTH MONITORING HEADPHONE

For the award of

ONTARIO GRADUATE CERTIFICATE

Submitted by,

Christy Rachel Philip (C0765535)

Fahad Rahman (C0769871)

James M Chacko (C0777192)

Premil Prasannan (C0777191)

Under the guidance of

Dr. Mike Aleshams

ESE-4009 Embedded System Design Project

(May 2021 - August 2021)

Abstract

This project aims to develop a health monitoring headset with IoT to monitor the everyday health of an individual at all times. This is an IoT-connected health care device, which is connected as a part of Bluetooth headphones. It consists of EKG sensor, pulse oximeter, temperature sensor, GPS and GSM to obtain various health-related data such as blood oxygen level, temperature, EKG and patients' live location.

The device can be connected to a mobile phone through IoT. The user's information such as height, weight and previous history can add in the webserver earlier itself. Using this, the family doctor can access it directly and monitor the patient's condition every day and communicate with them. Along with this, it has a push-button as well, so in case of emergencies, patient’s can press it by themselves and send information to 911 and doctor.

Acknowledgement

We would like to express our most profound appreciation to everyone at Embedded Systems Design Engineering Class in Lambton College in Toronto who provided us with the possibility to complete this project. Special gratitude to project guide Dr. Mike Aleshams whose engagement and contribution helped us coordinate our project and filing the final project report.

We would also like to extend our appreciation to Dr. Takis Zourntos, Course Coordinator, Embedded Systems Design Engineering, for project preparation guidance.

Heartfelt gratitude to our parents and our teachers from school and college, for their love and encouragement, without whom we would never have enjoyed so many opportunities.

Lambton College in Toronto is a great place to study and work, mainly because of its supporting faculty members. Thanks to all my colleagues in college for supporting and encouraging our idea.

CHRISTY RACHEL PHILIP

FAHAD RAHMAN

JAMES M CHACKO

PREMIL PRASANNAN

Table of Contents

List of Figures

Chapter I

Introduction

Overview

This project is about the development of a health monitoring headphone using IoT, which helps the client monitor their body temperature, blood oxygen level and EKG along with enjoying music. 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). 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.

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.

Problem Statement

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

So we are introducing our project as a solution for this problem.

Goal and Objectives

- **Goal**

Develop health monitoring headphones using IoT, which helps the client monitor their body temperature, blood oxygen level and EKG along with enjoying music by the end of the summer 2021 academic term.

- **Objective**
    1. Connecting a host processor ESP32 to our main sensors like MAX30102 Pulse Oximetry sensor and Heart rate sensor and DS18B20 temperature sensor.
    2. Adding GPS, GSM module and push button for emergency situation
    3. Configuring Wi-Fi and server capabilities to ESP32.

The Scope of the Project

Deliverable

1. DS18B20 (Temperature sensor)
2. Max30100 Pulse Oximetry sensor
3. ESP32 Microcontroller
4. Rechargeable Li-ion Battery
5. GSM Module
6. GPS Module
7. Push-button

Milestones

- June 4, 2021 - Ordering Components
- June 8, 2021 – Testing the components
- June 11, 2021–ESP32 setup
- June 15, 2021 – Interfacing with GSM
- June 18, 2021 – Interfacing with GPS
- June 24, 2021– Interfacing with Temperature Sensor
- July 1, 2021 - Interfacing with EKG Sensor
- July 8, 2021 –Interfacing with SPO2 Sensor
- July 13, 2021 – Interfacing with Push Button
- July 15, 2021 –Real-Time Operation
- July 20, 2021 – Power Management
- July 23, 2021 – Cloud Storage Interfacing
- July 27, 2021 – Schematic Capture design
- July 28, 2021 –PCB Layout design
- August 05, 2021 –Code Integration
- August 11, 2021 –PCB Implementation
- August 12, 2021 – Project Report
- August 18, 2021 – Final Demonstration

Limitations

1. ESP32 has insufficient processing power for fast WiFi or CPU-intensive activities while using the WiFi, too few GPIOs, one ADC pin, and no onboard USB.

2. Anyone with physical access to the device can read your more sensitive information, such as Wi-Fi credentials.
3. ESP32 has less SRAM and ROM, but again it supports more prominent external memories.

Outcomes and Benefits

- The user will obtain all the sensor readings of temperature and max30102 pulse oximetry and heart rate sensor.
- 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
- For emergency conditions, the patient can call 911 using the push button and GSM module.
- Lightweight and portable to carry along with enjoying music.

Procedure and Methodology

- After testing the components, we interface the EKG, SPO2, temperature sensor , Pushbutton, GSM module with our ESP32.
- Attached real time operation techniques to the project with help of priority and threads.
- Added power management techniques by forcing ESP32 into deep sleep mode.
- Upload the source code to Esp-32 module using the arduino ide
- Powering Up the module using th Li-Ion battery
- Press the boot button in the ESP-32 module to boot the module and to turn on the system
- Attach the finger to the Max30102 sensor to get the EKG and SP02
- Open Blynk App in your smartPhone to view the result

Chapter II

Literature Review

A number of assessments on the theme of Wireless-Sensors methods were done before as projects reports or, as research papers on IoT based Patient Health Monitoring Systems. In an article published in 2018, International Conference on Advances in Computing and Communication Engineering named “An IoT based Patient Health Monitoring System" they explained about patient’s health monitoring system based on IoT uses internet to effectively monitor patient health and helps the user monitoring their loved ones drom work and saves lives. The following figure (Fig.2.1) shows its engineering diagram.

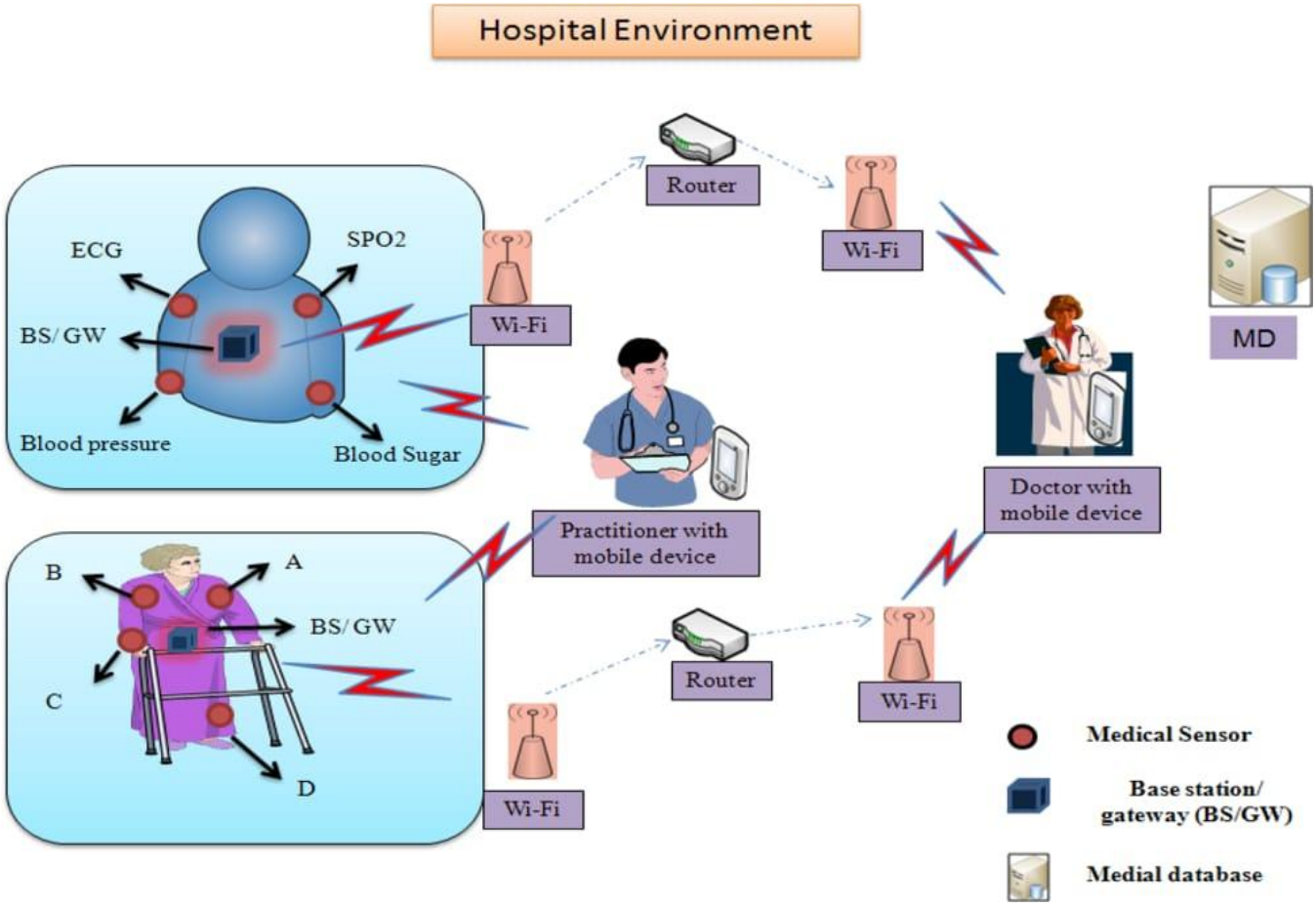

Fig.2.1: Engineering Diagram of the current proposal

A remote sensor system (WIRELESS-SN) may be remotely organized which comprises structurally dispersed self-sufficient gadgets that utilization sensors will screen physical or Ecological states. These self-sufficient devices, alternately nodes, consolidated for routers will make An WIRELESS-SN framework. Sensor networks are those ways of gathering the majority of the data required Toward advanced mobile environments, if in constructions, efficacies, productions, home-based, crafts, conveyance frameworks mechanization, or somewhere else. Late radical Also revolutionary war-fare control measures necessitate conveyed nets for radars that might have a chance to be extended. What's more have self establishing abilities. For such presentations, seriatim cables alternately wiring is as rule illogical. That advanced mobile passage is outlined should empower WIRELESS-SN Also state-funded correspondence networks with right one another(with consistent internetworking. In this design, the passage comprises from claiming vital controller component, databank (D-B), Wireless-SN component, Wireless-LAN, Moreover mobile GSM component.

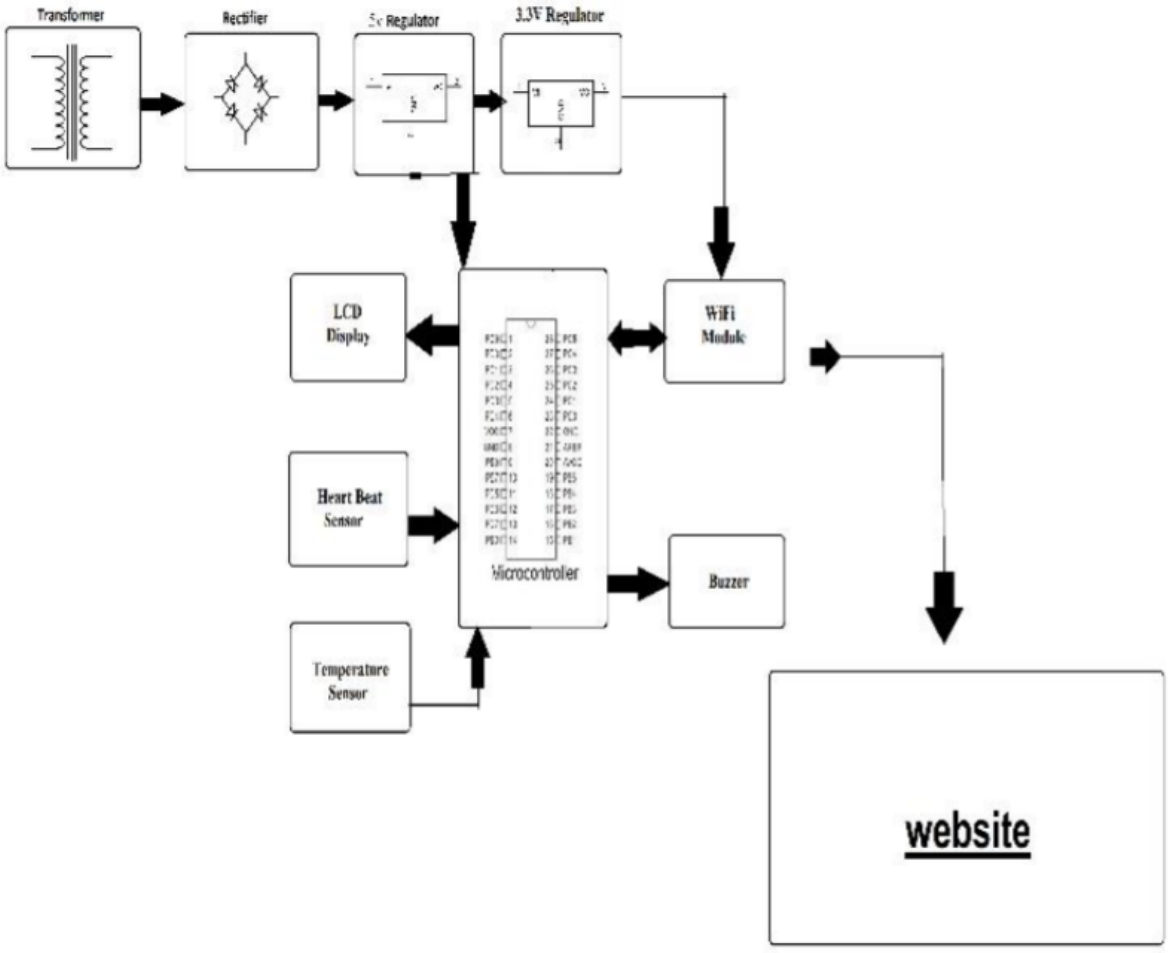

Fig.2.2 Architectural diagram of existing proposal

Structure from claiming passage the disseminated estimation hubs correspond remotely with a focal gateway, which gives an association of the wired universe the place the information could make collect, process, dissect and display your estimation information. With augment separation Also dependability in An remote sensor network, you camwood use routers with expansion an additional correspondence connection between conclusion hubs and the passage. Those passages incorporate three outer correspondence components (ECCM): Wireless-SN Unit, Wireless-LAN AiP, Furthermore mobile module GsM. Wireless-SN unit, ahead unique pointer, is essentially utilized to getting information bundles starting with those centre of Wireless-SN; on the additional pointer, it will be recycled to direct instructions of the Wireless-SN or particular sensing hubs. This executes the decorum interpretation and gives those corporal systems the middle of passage Furthermore Wireless-SN. In this project, a MiB520/22 US-B edge table attaching for those basin hubs Previously, WIRELESS-SN may be utilized Concerning illustration of the WIRELESS-SN component. A GSM unit will be necessary after conveyance SMS of the sub-scribers utilizing GsM systems, alternately transmitting information with secluded servers over GPRS whether essential. GTi-64 is a canny GsM/GPRS controller node that put composed GsM nets for those 1810/1905 mega-Hz r-f groups. That correspondence could make acknowledged through SMS above GsM or SMS In GPRS utilizing customary during guidelines. Wireless-LAN AiP needs some assignments. In so as should unite with those internets, it acts Concerning illustration An customer pc acknowledges the IP address allocated Toward those web server. Second, it sets dependent upon an ad-hoc system to subscribers Furthermore framework maintainers with the goal that they might unite with the keen passage with smartphone or PDA effectively.

Chapter III

Requirements / Analysis

Hardware

This health monitoring IoT headphone requires a Bluetooth headset, ESP32 Board, Wi-Fi, Bluetooth capabilities along with some sensors and GSM Modules. We used software and designed our own customized PCB using the schematic design which helped us connect all the wires.

Product fits for both personal and professional purposes in healthcare. It can be used as a daily healthcare product for every individual irrespective of time and place. This device will help the patient as well as doctors to monitor their daily health. In case of emergency, because of its emergency messages, this might act as a lifesaver.

We require the following list to make a complete health monitoring headphone:

Hardware Requirements:

- ESP 32 Board with GSM Module.
- Max30102 (Combination of Pulse Oximetry sensor and Heart Rate Sensor)
- Thermistor (Temperature sensor)
- GPS Module
- Push-button

  Here we need two pushbuttons, one for an emergency call and another one for text messages.

- Connecting Wires
- Jumper Wires
- Basic Electronic Components: Multimeter (to check on voltages and currents at various nodes basically for debugging the circuit), Wire Cutters (Cut off the wires of the required lengths to make the circuit look clean), Strippers (To remove the insulation and connect)
- Breadboard (Used to make the circuit for testing before final PCB design and implementation)
- Soldering Equipment (Soldering Iron: Solder the components on the PCB.

➢ Desoldering Gun: Removes the excess of soldering or if soldered incorrectly, Soldering Wire: used as electrical conducting to join the connection)

Software Requirements:

➢ Blinky IoT Application

All the data will be recorded in this IoT based application.

➢ Arduino IDE

Using Arduino IDE we are uploading the code to the ESP32 board.

➢ EasyEDA:

A PCB design and simulation tool enabling hardware engineers to write EDA (Electronic Design Automation). We use this tool for PCB design and layout. It could be used as a simulation but Beaglebone simulation may not be possible as P a g e 15 SPICE libraries are currently not present in the software. Hence software simulation of the Beaglebone circuits cannot be done but the PCB can be designed using this software

➢ C/C++(Coding language)

For programming, we use C language, and code will follow MISRA coding c guidelines (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 a boolean type.
- All object and function identifiers shall be declared before use.
- Not to use non-constant pointers to functions.

Power Requirement:

➢ Rechargeable Li-ion Battery is using to get external power

Block Diagram

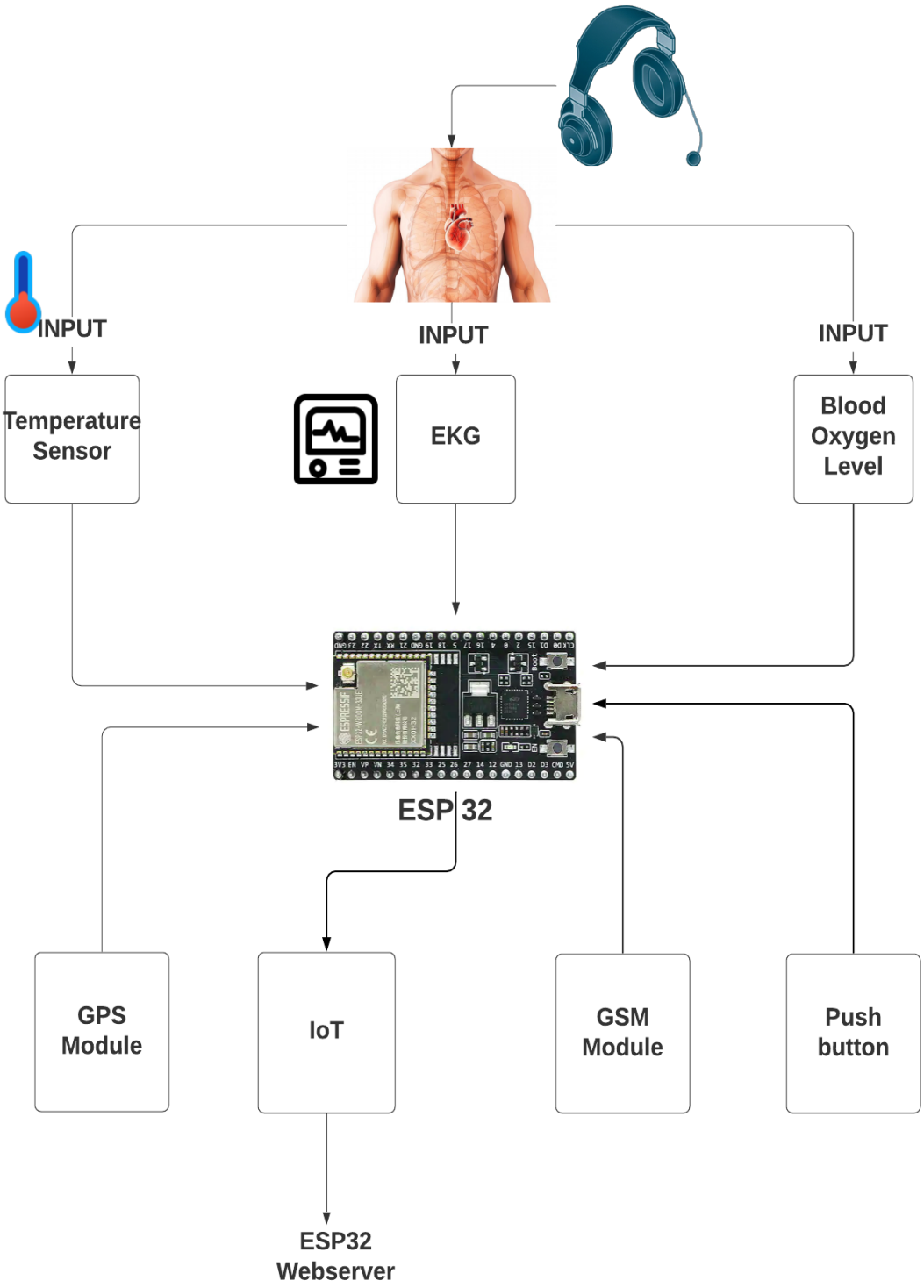

Figure 3.1 Block Diagram

Flowchart

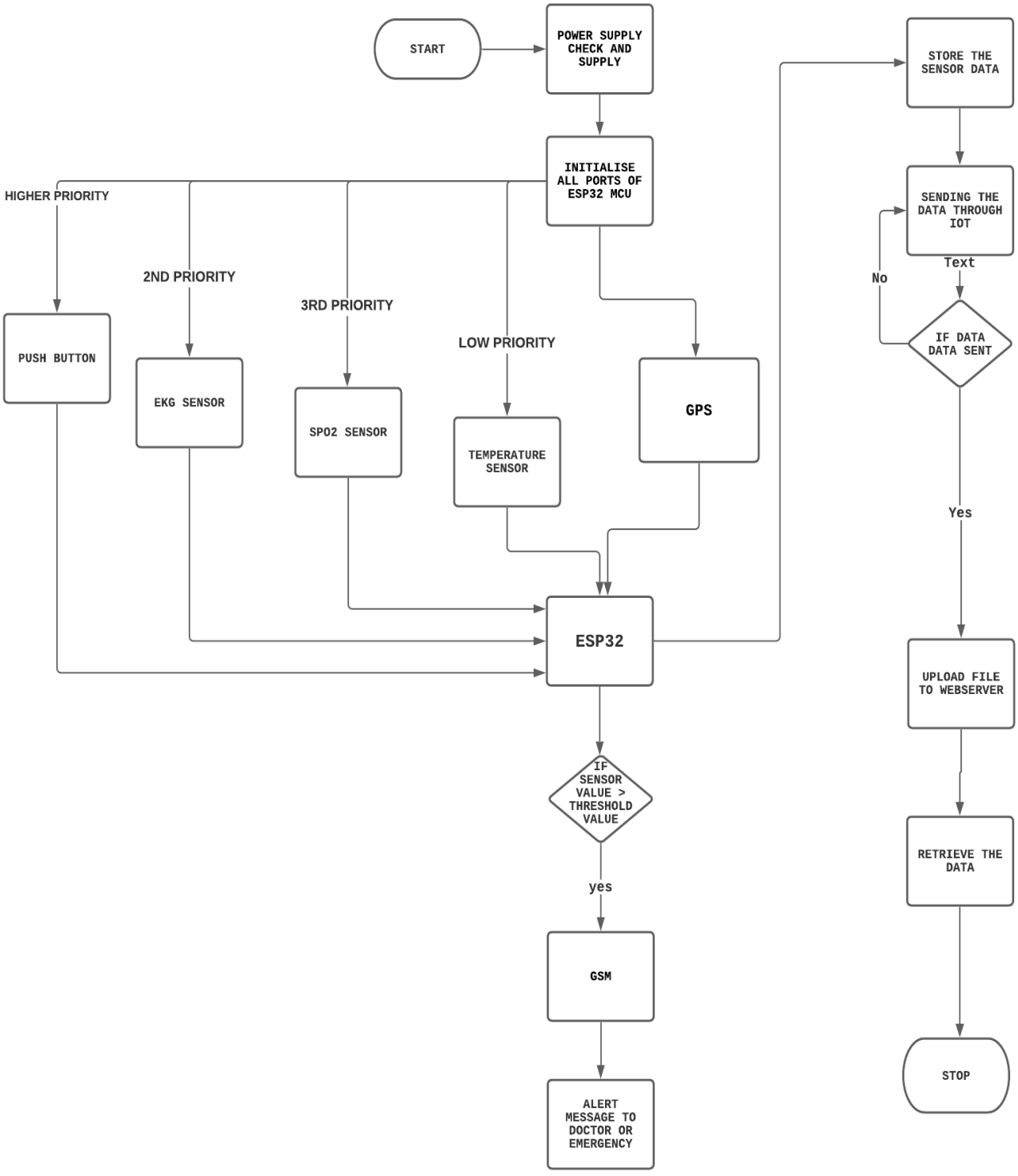

Fig. 3.2. Flowchart

Chapter IV

Design

Knowing Hardware

Host Processor: TTGO T-CALL ESP32 Microcontroller Board.

The ESP32 is an updated version of the ESP8266. It offers both Bluetooth, GSM and WiFi modules in-built in it. It is faster and is available in a dual-core design. It is also capable of operating in an ultra-low-power mode, ideal for battery-powered applications.

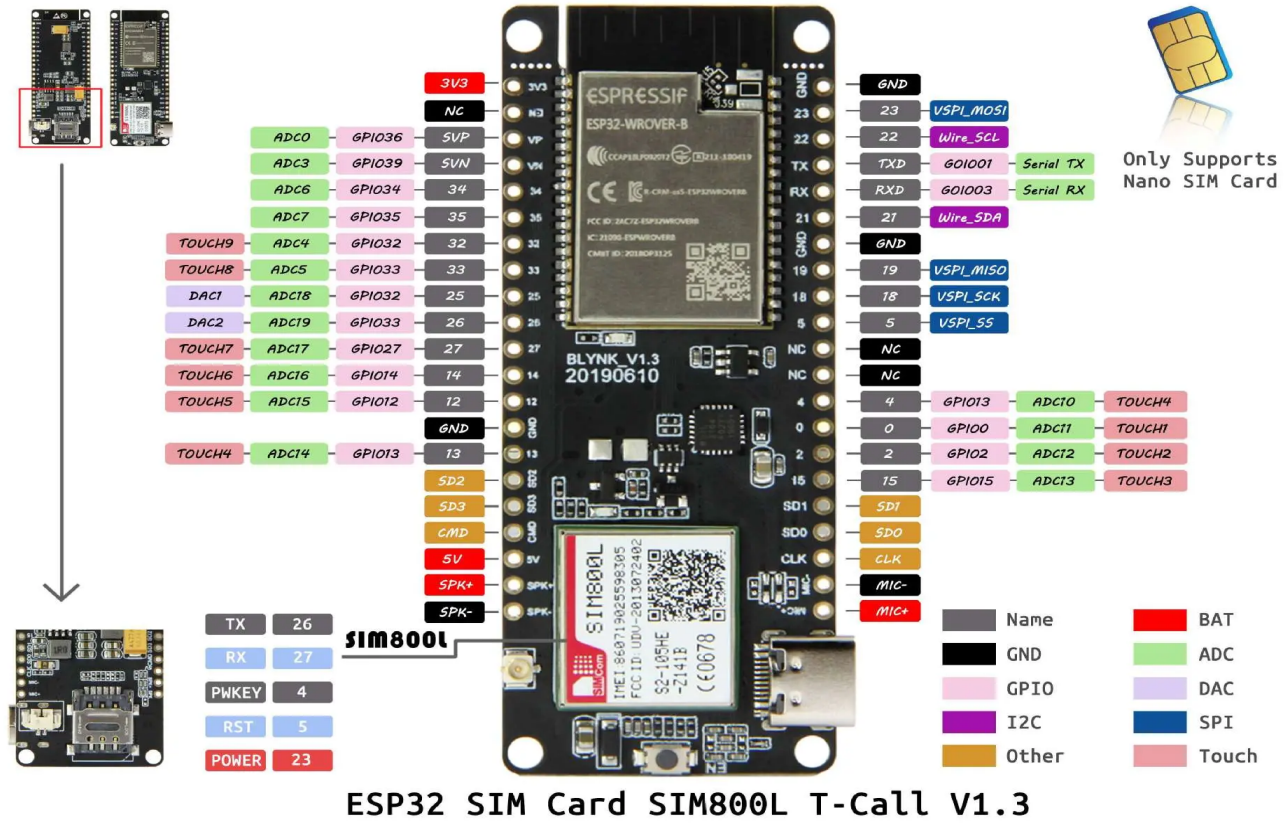

Fig.4.1. ESP32 Microcontroller Board

The ESP32 is powered by a micro-USB connector. Plug a USB cable with micro-B plug into the micro-USB socket on the board and the other end into a PC USB port to power up the board. A regulator on the board supplies the ESP-WROOM-32 module with 3.3V derived from the USB 5V.

SIM800L

SIM800L is an inbuilt GSM module in the TTGO-T-CALL ESP32 development board. Besides Bluetooth and wifi, we can communicate with ESP32 through SMS and phone call. Additionally, we can connect ESP32 to the internet using the sim card data plan. The operating voltage of the chip is from 3.4V to 4.4V, which makes it an ideal candidate for a direct LiPo battery supply. All the necessary data pins of the SIM800L GSM chip are broken out to 0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART. The module supports a baud rate from 1200 bps to 115200 bps with Auto-Baud detection.SIM800L is a quad-band GSM/GPRS module(combines GPS technology for satellite navigation), that works on frequencies GSM850MHz, EGSM 900 MHz, DSC 1800 Mhz, and PCS1900MHz. SIM800L features GPRS multi-slot class 12 / class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

Specification of SIM800L

- Supply voltage: 3.8V - 4.2
- Power consumption:
  - sleep mode < 2.0mA
  - idle mode < 7.0mA
  - GSM transmission (avg): 350 mA
  - GSM transmission (peek): 2000mA
- Module size: 25 x 23 mm
- Interface: UART (max. 2.8V) and AT commands
- SIM card socket: microSIM (bottom side)
- Supported frequencies: Quad Band (850 / 950 / 1800 /1900 MHz)
- Status signalling: LED
- Working temperature range: -40 do + 85 ° C

Major features

- 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: The ESP32-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.

MAX30102 Sensor

MAX30102 sensor is a combination of pulse oximetry and heart-rate monitor sensor. MAX30102 has an integrated red LED with an infrared LED. It has an I2C interface so it is very easy to use.Max30102 sensor can be used as both;

- Heart rate monitoring sensor
- SPO2 detecting sensor

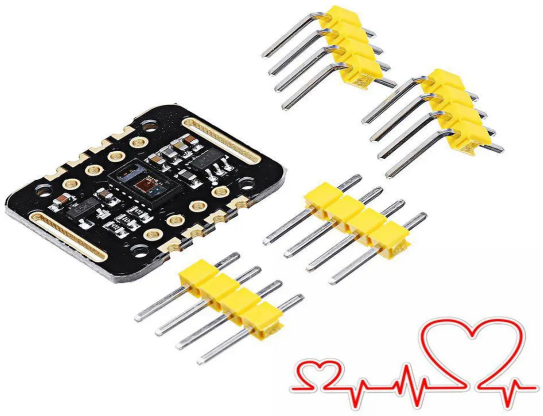

Fig.4.2. MAX30102 Sensor

The MAX30102 operates on a single 1.8V power supply and a separate 5.0V power supply for the internal LEDs. Communication is through a standard I2C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices.

Push-Buttons

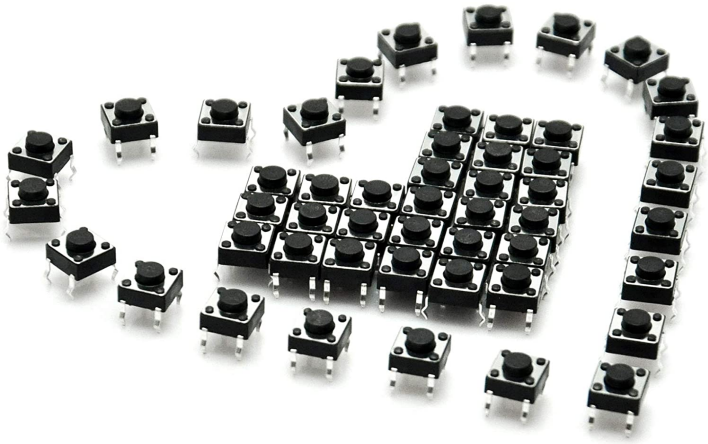

Fig.4.3.Push-button

Push-Buttons are normally open tactile switches. Push buttons allow us to power the circuit or make any particular connection only when we press the button. Simply, it makes the circuit connected when pressed and breaks when released. A push-button is also used for triggering the SCR by the gate terminal.

A push-button helps in closing an open electronic circuit or the reverse temporarily during the time it is pressed. It has a small spring that helps it to return to its original position when the user takes the pressure off the button.

There are mainly two types of push-buttons:

- ➢ Normally open (NO)
- ➢ Normally closed (NC)

Normally open (NO) Push Button

A circuit with a normally open push-button is open by default. When the user presses the button, the circuit becomes complete and the current starts flowing through it.

Normally closed (NC) Push Button

A circuit with a normally closed push-button is closed by default. The circuit breaks by pressing the button.

Technical Specifications

- ➢ Mode of Operation: Tactile feedback
- ➢ Power Rating: MAX 50mA 24V DC
- ➢ Insulation Resistance: 100Mohm at 100v
- ➢ Operating Force: 2.55±0.69 N
- ➢ Contact Resistance: MAX 100mOhm
- ➢ Operating Temperature Range: -20 to +70 ℃
- ➢ Storage Temperature Range: -20 to +70 ℃

Pinout Explanation

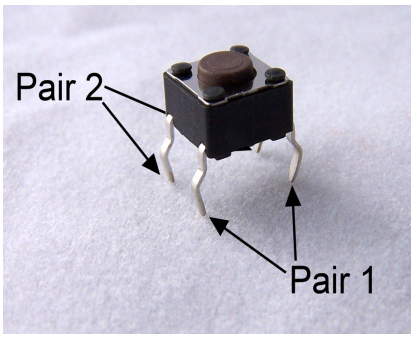

Fig.4.4. 2 Pairs of pins

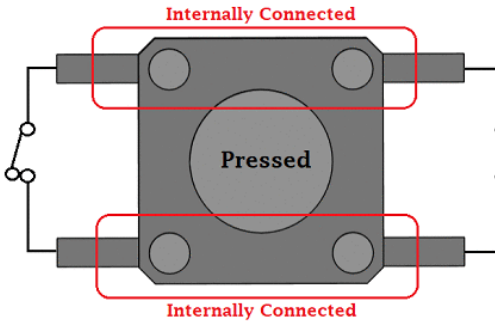

Fig.4.5. Internal connections.

For push-button switches, pin 1 and 2 are internally connected and pin 3 and 4 are internally connected. The above images made it clear. So if we give a connection to pin 1, it will get on pin 2 as well. Similarly for pins 3 and 4. When we establish a connection between pin 1 and 3 or 2 and 4, then the button would be pressed, else it would be in the released state as shown in the figure.

NEO-6M GPS Module.

This module is a NEO-6M GPS chip from u-blox. It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking while consuming only 45mA supply current.

Unlike other GPS modules, it can do up to 5 location updates a second with 2.5m Horizontal position accuracy. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second.

One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces the power consumption of the module to just 11mA making it suitable for power-sensitive applications like GPS wristwatches.

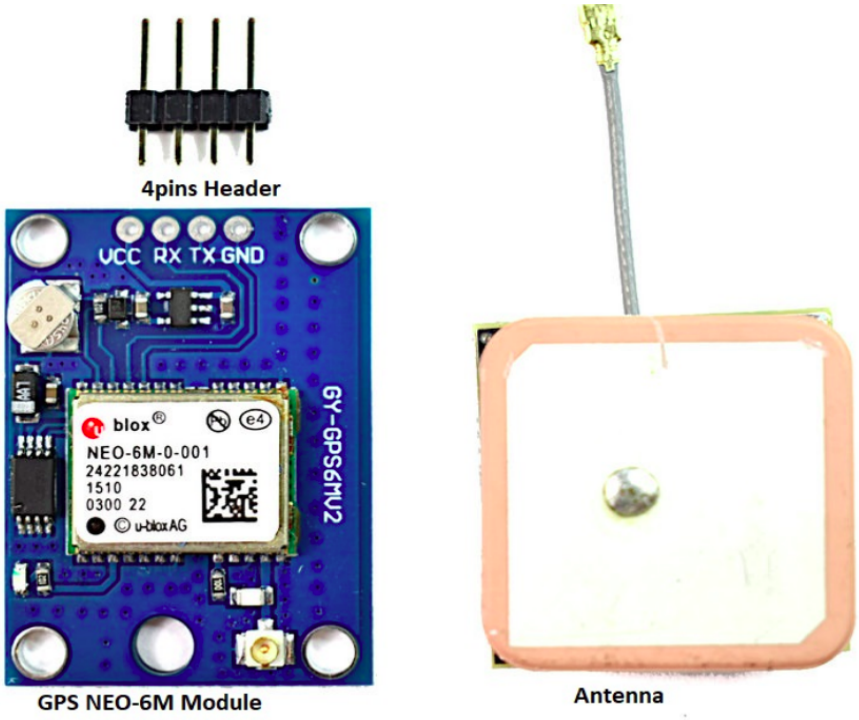

Fig.4.6. Neo 6M GSM Module

Specifications

- ➢ Power supply: 2.7V to 3.6V
- ➢ Default baud rate: 9600 bps
- ➢ Current consumption: 45mA.
- ➢ Up to 22 satellites on 50 channels tracking
- ➢ Up to 5 location updates a second with 2.5m Horizontal position accuracy.
- ➢ Navigation Sensitivity: Up to -161 dBm
- ➢ Navigation Update Rate: 1Hz
- ➢ Comes with an external antenna and built-in EEPROM.
- ➢ Interface: RS232 TTL with Serial Baud Rate 4800-230400 (default 9600)
- ➢ Operating Temperature: -40°C ~ 85°C
- ➢ Communication Protocol: UBX Binary, RTCM and Standard NMEA output

Temperature Sensor: DS18B20

DS18B20 is a 1-Wire interface Temperature sensor manufactured by Dallas Semiconductor Corp. The interface requires only one digital pin for two way communication with a microcontroller.

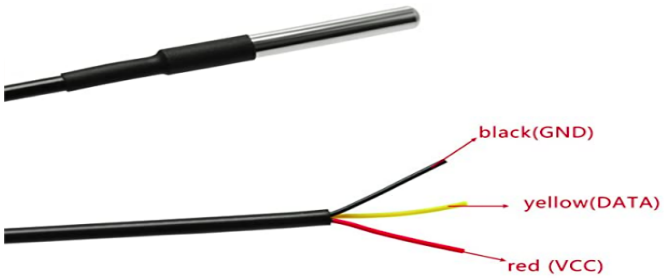

Fig.4.7. DS18B20 Temperature Sensor

The sensor comes usually in two form factors. One that comes in the TO-92 package looks exactly like an ordinary transistor. Another one is a waterproof probe style which can be more useful

when you need to measure something far away, underwater or under the ground.

Specifications

- ➢ Usable temperature range: -55 to 125 °C (-67 °F to +257 °F)
- ➢ 9 to 12 bit selectable resolution.
- ➢ Uses 1-Wire interface- requires only one digital pin for communication.
- ➢ Unique 64 bit ID burned into the chip.
- ➢ Multiple sensors can share one pin.
- ➢ ±0.5 °C Accuracy from -10 °C to +85 °C.

Schematic and PCB Design

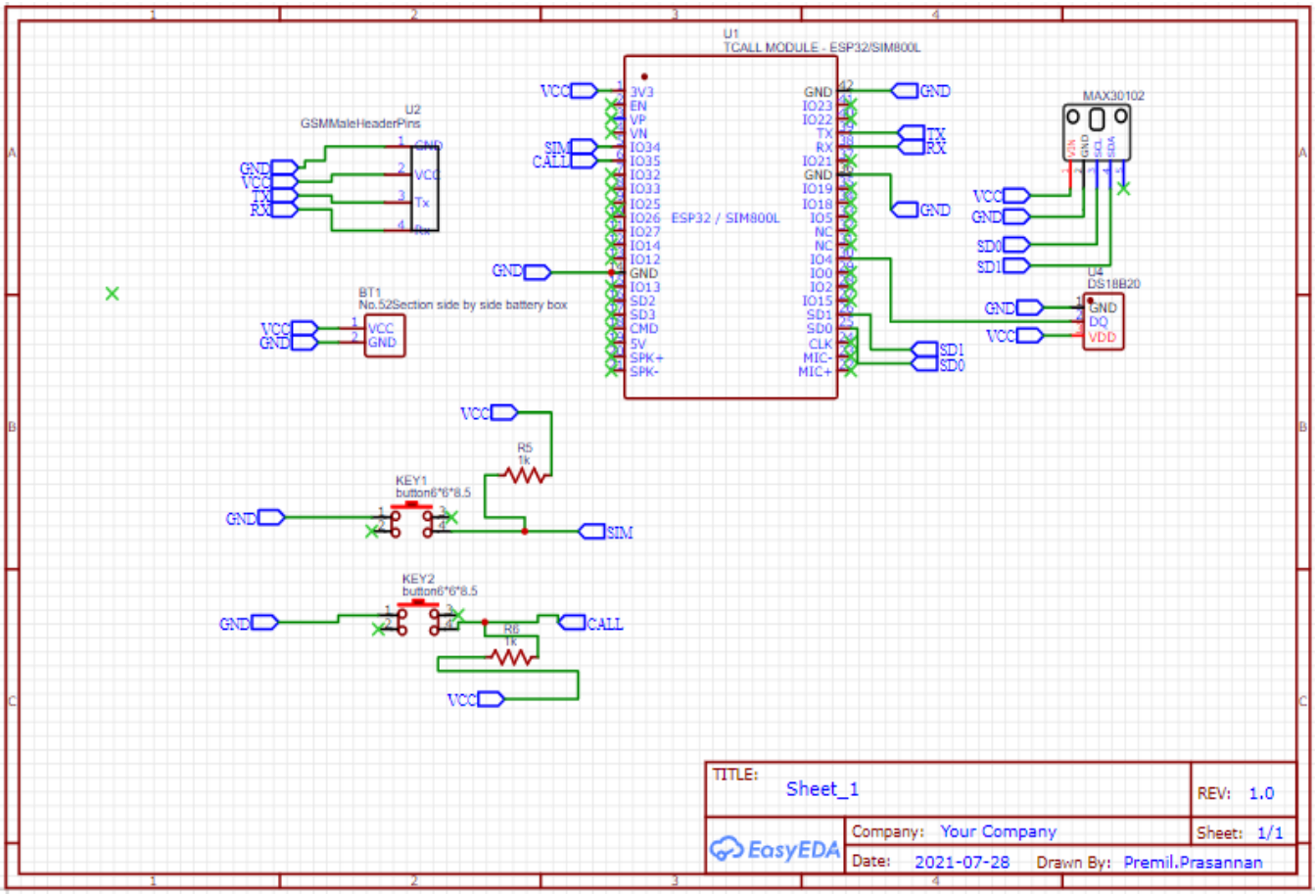

Fig. 4.8 Schematic Design

PCB Design 3D View

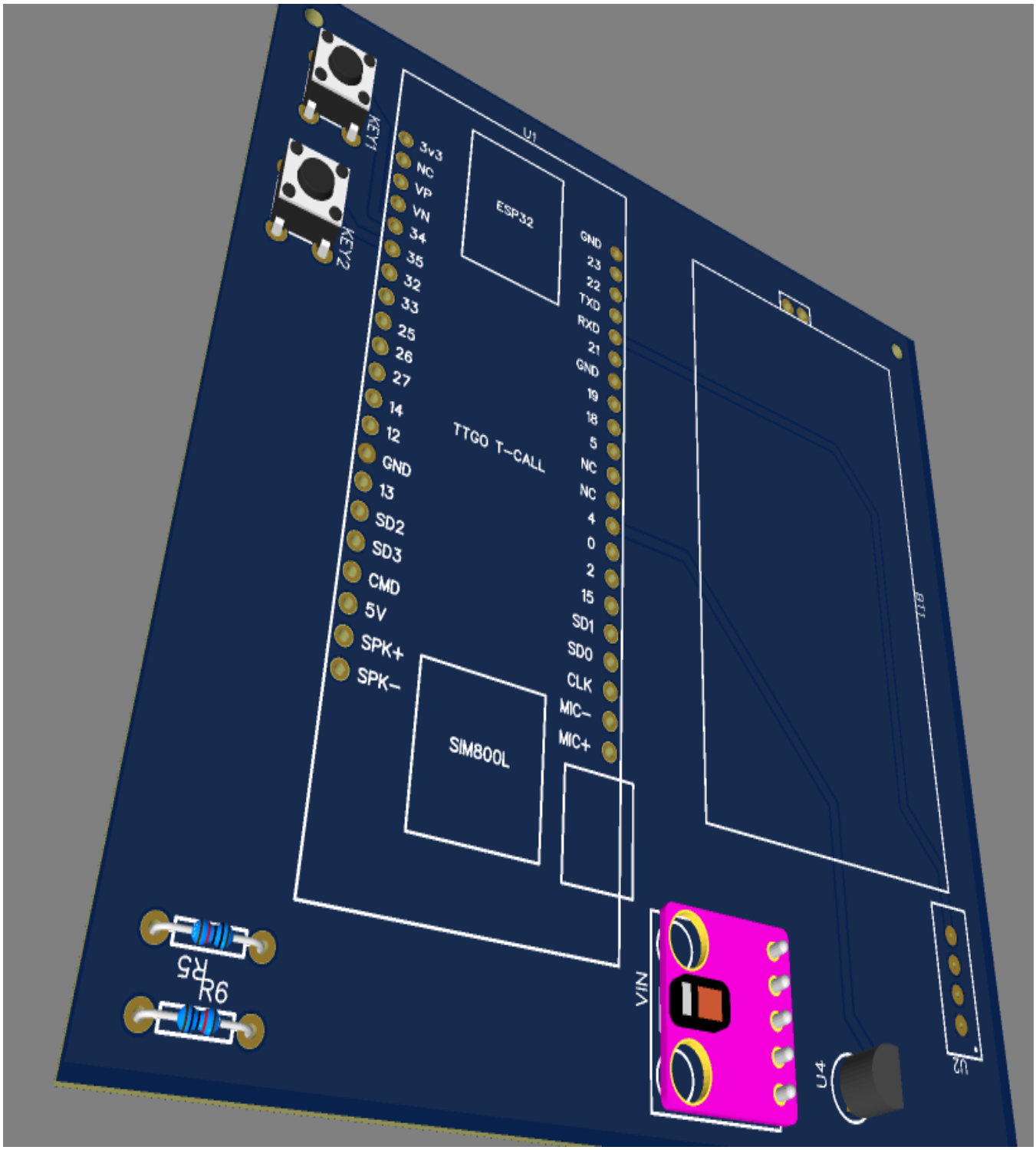

Fig.4.9. PCB Design: 3D View

PCB Board image

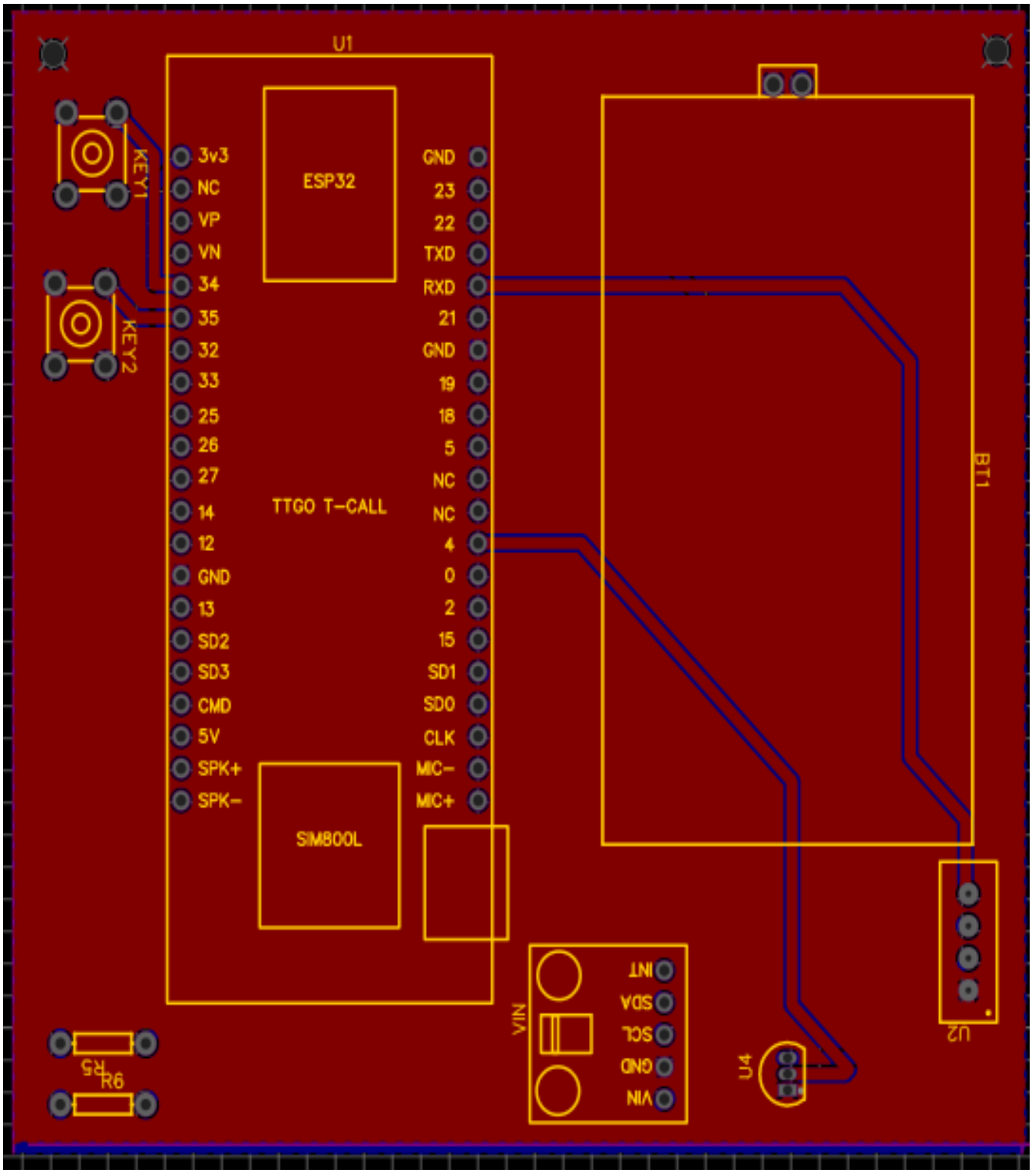

Fig. 4.10. PCB Board

Software Section: Blynk Application

Blynk is the most popular IoT platform to connect your devices to the cloud. It is well-known for allowing users to design apps to control their IoT devices, analyse telemetry data, and manage your deployed products at scale. Blynk App - allows you to create amazing interfaces for your projects using various widgets we provide. Blynk Server - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your private Blynk server locally.

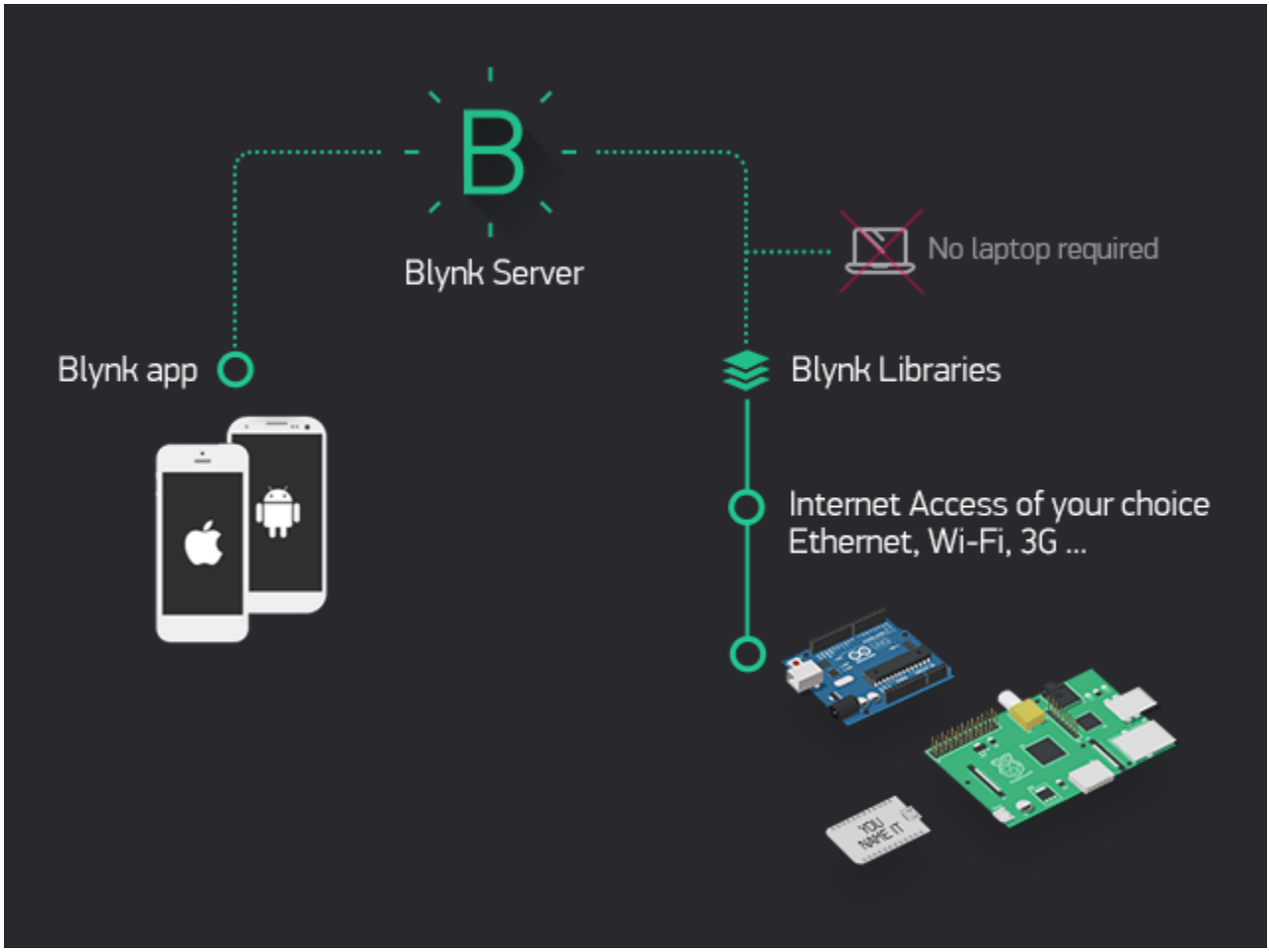

Fig. 4.11. Blynk App Working

Chapter V

Implementation and Test

ESP32 setup

The ESP32 is an updated version of the ESP8266. It offers both Bluetooth, GSM and WiFi modules in-built in it. It is faster and is available in a dual-core design. It is also capable of operating in an ultra-low-power mode, ideal for battery-powered applications.

The ESP32 is powered from a micro-USB connector. Plug a USB cable with micro-B plug into the micro-USB socket on the board and the other end into a PC USB port to power up the board. A regulator on the board supplies the ESP-WROOM-32 module with 3.3V derived from the USB 5V.

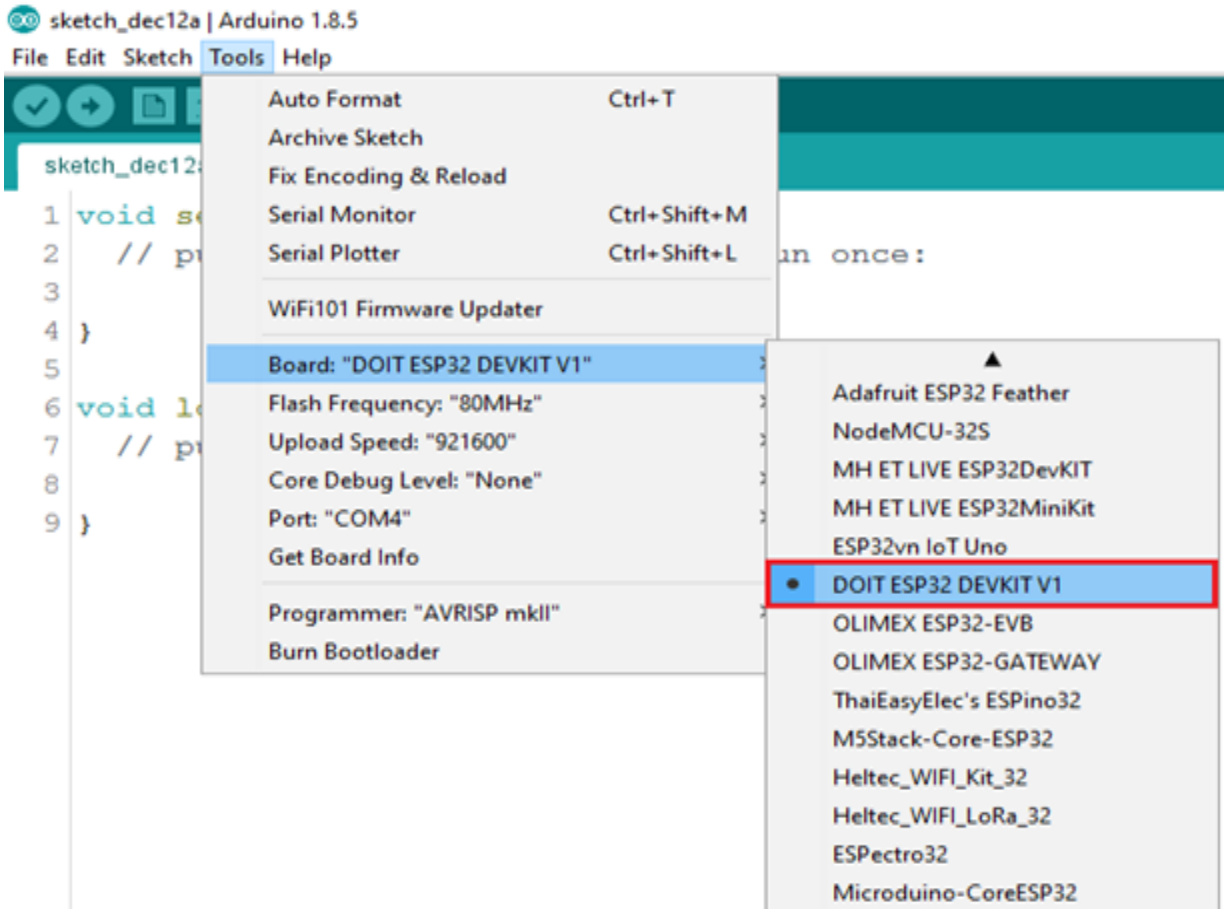

Fig.5.1 ESP32 Set up in Arduino IDE

Interfacing ESP32 with DS18B20 Temperature Sensor

Circuit Diagram

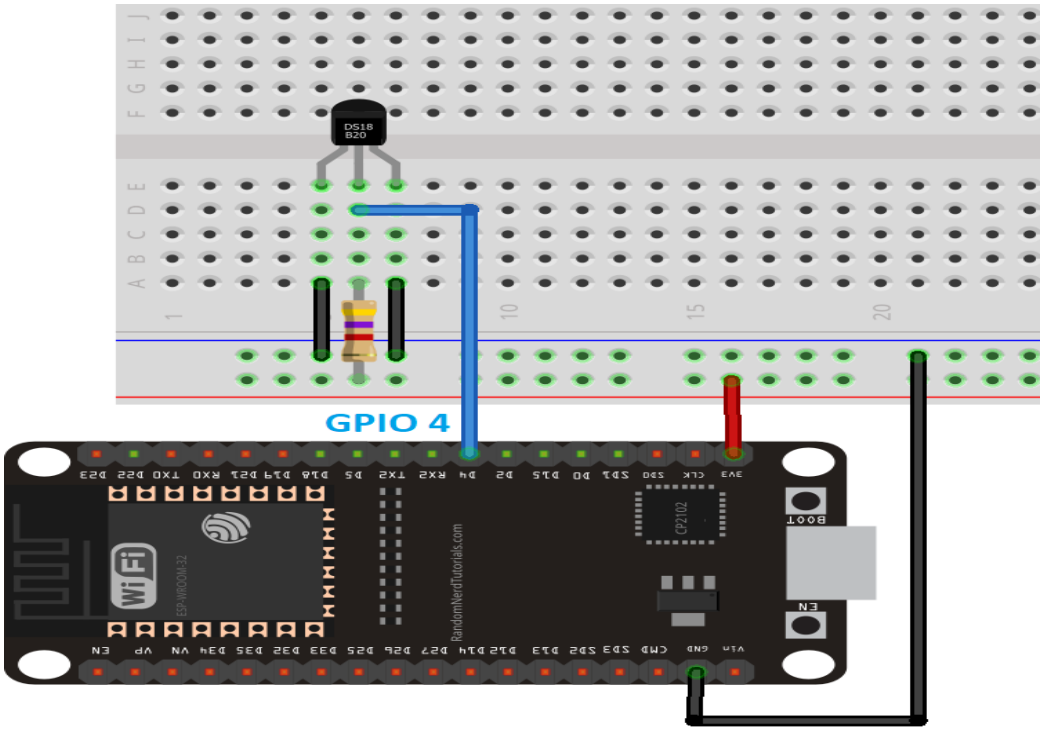

Fig.5.2. Interfacing ESP32 with DS18B20 Temperature Sensor

Connections

Connect the Vin pin of DS18B20 to 3.3 pin of ESP32.

Connect the GND pin of DS18B20 to the GND pin of ESP32.

Yellow Stripe is data that goes to GPIO pin 4 on ESP32.

Then add a 4.7k pull-up resistor between the signal and power pin to keep the data transfer stable

Interfacing ESP32 with SPO2 Sensor

The sensor is integrated pulse oximetry and heart-rate monitor sensor solution. It combines two LED's, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse and heart-rate signals. It operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.

The device has two LEDs, one emitting red light, another emitting infrared light. For pulse rate, only the

infrared light is needed. Both the red light and infrared light is used to measure oxygen levels in the blood.

When the heart pumps blood, there is an increase in oxygenated blood as a result of having more blood. As the heart relaxes, the volume of oxygenated blood also decreases. By knowing the time between the increase and decrease of oxygenated blood, the pulse rate is determined.

It turns out, oxygenated blood absorbs more infrared light and passes more red light while deoxygenated blood absorbs red light and passes more infrared light. This is the main function of the MAX30100: it reads the absorption levels for both light sources and stores them in a buffer that can be read via I2C.

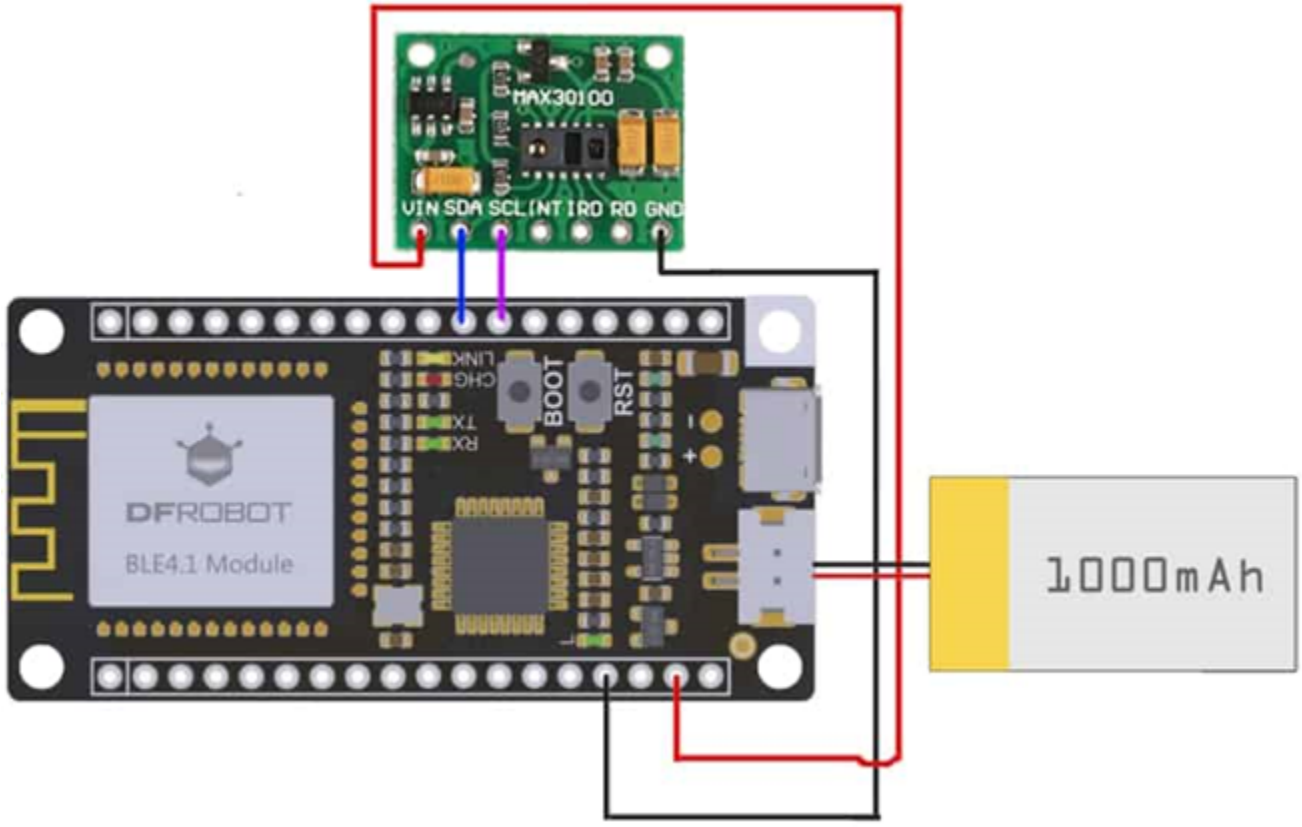

Fig.5.3. Interfacing ESP32 with SPO2 Sensor

How does an oximeter work?

A pulse oximeter is composed of the sensor (or probe) and the monitor with the display. The probe is on the finger and is detecting the flow of blood through the finger. This is displayed as a pulse wave on the monitor. A pulse wave must be present to demonstrate that a pulse is being detected. In this figure, the patient has a pulse rate of 72 beats / minute and an SpO2 of 98%. This monitor describes the pulse rate as the heart rate.

All pulse oximeter probes (finger or ear) have light emitting diodes (LEDs) which shine two types of red

light through the tissue. The sensor on the other side of the tissue picks up the light that is transferred through the tissues. The oximeter can determine which of the haemoglobin is in pulsatile blood (arterial) and can then determine the SPO2 of arterial blood in the peripheral circulation.

Hardware Connections (ESP32 Arduino):

-VIN = 3.3V

-GND = GND

-SDA = 21 (or SDA)

-SCL = 22 (or SCL)

Fig.5.4. Output in blynk App

Interfacing ESP32 with EKG Sensor

The main concern of our project is to monitor our client's health along with enjoying music. In our headphones, we connected different sensors to monitor their body temperature, blood oxygen level, and EKG. The EKG sensors are used to monitor the client's heart rate and keep a record of the average heart rate every second.MAX30102 is the sensor used for this purpose. 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.

The device has two LEDs, one emitting red light, another emitting infrared light. For pulse rate, only the

infrared light is needed. Both the red light and infrared light is used to measure oxygen levels in the blood.The device has two LEDs, one emitting red light, another emitting infrared light. For pulse rate, only the infrared light is needed. Both the red light and infrared light is used to measure oxygen levels in the blood.It turns out, oxygenated blood absorbs more infrared light and passes more red light while deoxygenated blood absorbs red light and passes more infrared light. This is the main function of the MAX30100: it reads the absorption levels for both light sources and stores them in a buffer that can be read via I2C.

In this project we will connect the heart-rate sensor to the 3V ESP32 pin.As said before, this sensor has an I2C interface. Therefore, we need to use 4 wires.This sensor has several pins anyway we don't need to use them. To measure the heart-rate and the pulse oximetry, the ESP32 must connect to the MAX30102 in this way:

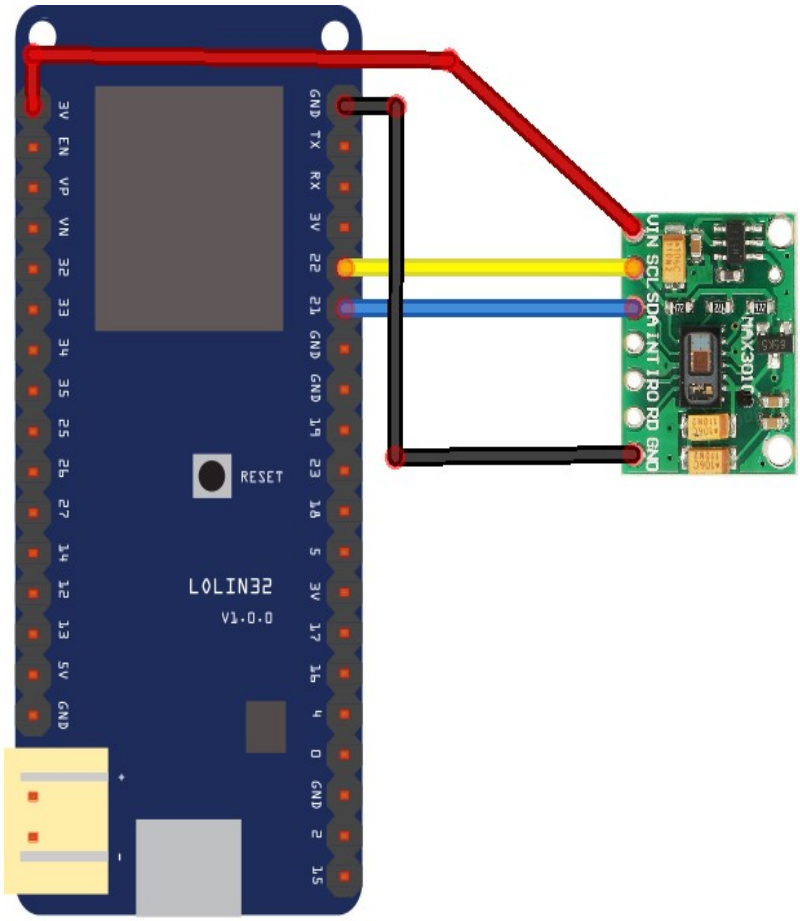

Fig. 5.5 Interfacing ESP32 with EKG Sensor

Interfacing ESP32 with GSM

A GSM module is a chip or circuit that will be used to establish communication between a mobile device or a computing machine and a GSM or GPRS system. The modem (modulator-demodulator) is a critical part here.The TTGO-T-CALL is an ESP32 development board which combines the SIM800L GSM module and ESP32 module. Besides bluetooth and wife we can communicate with ESP32 with SMS and phone call.Additionally we can connect ESP32 to the internet using the sim card data plan. This can be a great use for the IOT based project which has no nearby access to a wifi router.

SIM800L is an inbuilt GSM module in TTGO-T-CALL ESP32 by Sim.Its comes with a patch antenna. The operating voltage of the chip is from 3.4V to 4.4V, which makes it an ideal candidate for a direct LiPo battery supply. This makes it a good choice for embedding into projects without a lot of space. All the necessary data pins of the SIM800L GSM chip are broken out to 0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART. The module supports a baud rate from 1200 bps to 115200 bps with Auto-Baud detection.SIM800L is a quad-band GSM/GPRS module(combines GPS technology for satellite navigation), that works on frequencies GSM850MHz, EGSM 900 MHz, DSC 1800 Mhz, and PCS1900MHz. SIM800L features GPRS multi-slot class 12 / class 10 (optional) and supports the GPRS coding schemes CS-1, CS-2, CS-3 and CS-4.

Connections

1. SIM800_TX = Pin 26
2. SIM800_Rx = Pin 27
3. SIM800_PWkey = Pin 4
4. SIM800_RST = Pin 5
5. SIM800_POWER = Pin 23

In our project, 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 are using GSM module for this purpose.Here i am trying to send an sms using our GSM module SIM800L to my smartphone.In order to send sms am following some steps as mentioned below.

Steps for send an SMS with the TTGO T-Call ESP32 SIM800L

Step1 : ESP32 add-on Arduino IDE

Step 2:Prepaid SIM Card (SMS plan)

Step 3:Instal Libraries

Interfacing ESP32 with GPS

Connections

The GPS module will be powered with 3.3V power supply. It communicates with the ESP32 via serial communication using the TX and RX pins available on the 4 pins header.

- ➢ Connect the VCC pin of the GPS module to the ESP32 3.3V pin.
- ➢ Connect the GPS ground pin to the ESP32 ground pin.
- ➢ Now, connect the RX pin of the GPS module to the TX pin of the ESP32. Similarly, connect the TX pin of the GPS module to the RX pin of the ESP32 (RX and TX pins of the ESP32 are defined in the software).
- ➢ Last, connect your ESP32 to the computer through a USB cable. The ESP32 will be powered from the USB 5V.

Circuit Diagram

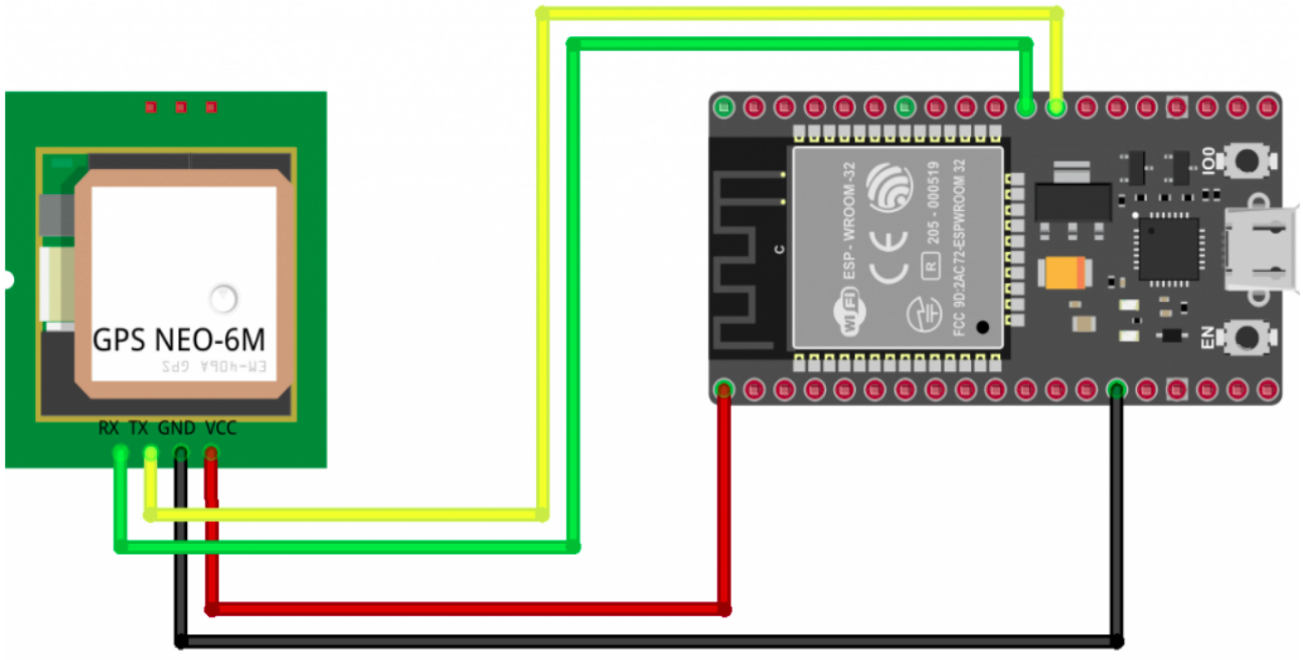

Fig.5.6.Interfacing ESP32 with GPS

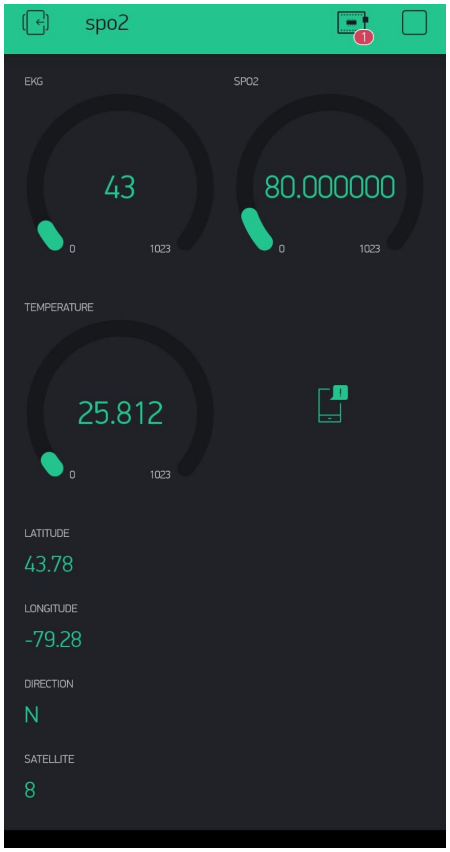

Fig.5.7 Output in Blynk App

<u>Interfacing ESP32 with Push Button</u>

Components Required

For Location Tracking

- ➢ TTGO TCALL from ESP32
- ➢ Neo6M GPS Module
- ➢ 3.7 V 600 mAh battery

For emergency features

- ➢ Micro Switches x 2
- ➢ Resistors x 2

Software part

- ➢ Arduino IDE
- ➢ Blynk Application

Circuit Diagram

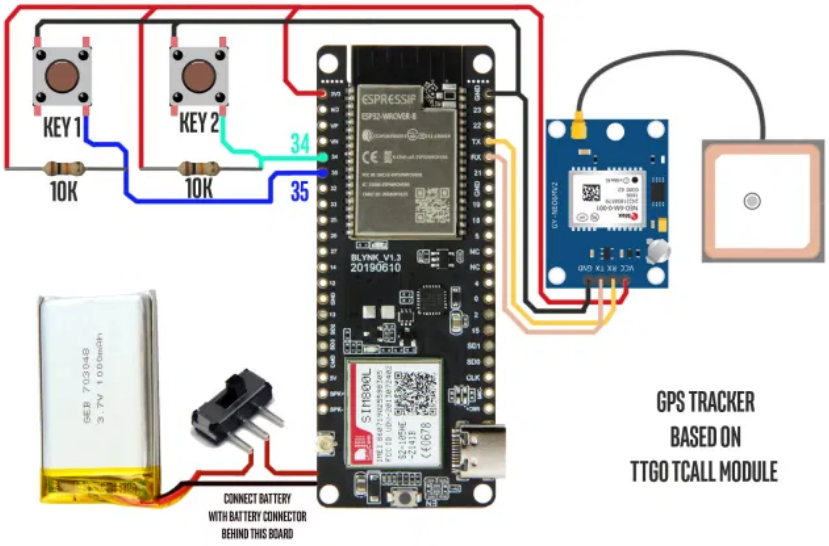

Fig.5.7. Interfacing ESP32 with Push Button

Working

The working of the project is quite simple. We have configured the Blynk application to use GSM as the communication protocol. In the Blynk part we have provided a Map to show the location and 5 value displays to display Latitude, Longitude, Speed, Direction and Number of satellites.

Now, as soon as the device is powered up it gets its location by using the GPS module and then pushes it to the Blynk server through GSM which is then received and displayed on the Blynk application.

Talking about the emergency features. We have added two microswitches in the PCB itself. Pressing one of which initiates a call to the mobile number mentioned in the code and the other sends an predefined emergency message to that number.

Output Sample

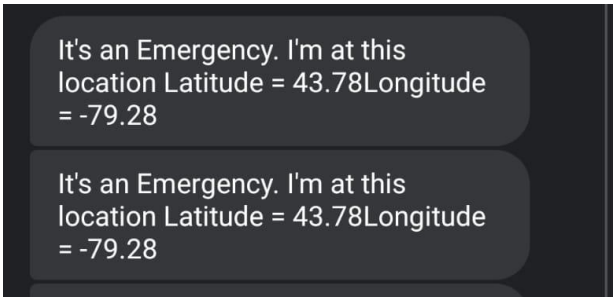

Fig.5.8. Output message

Real Time Operating System (RTOS)

The ESP32 comes with 2 Xtensa 32-bit LX6 microprocessors: core 0 and core 1. So, it is dual core. When we run code on Arduino IDE, by default, it runs on core 1. In this post we'll show you how to run code on the ESP32 second core by creating tasks. You can run pieces of code simultaneously on both cores, and make your ESP32 multitasking.

When we upload code to the ESP32 using the Arduino IDE, it just runs – we don't have to worry which core executes the code.

There's a function that you can use to identify in which core the code is running.

```
xPortGetCoreID()
```

If you use that function in an Arduino sketch, you'll see that both the setup() and loop() are running on core 1. Test it yourself by uploading the following sketch to your ESP32.

```
void setup() {

 Serial.begin(115200);

 Serial.print("setup() running on core ");

 Serial.println(xPortGetCoreID());

}



void loop() {

 Serial.print("loop() running on core ");

 Serial.println(xPortGetCoreID());
```

```
}
```

The Arduino IDE supports FreeRTOS for the ESP32, which is a Real Time Operating system. This allows us to handle several tasks in parallel that run independently.

Tasks are pieces of code that execute something. For example, it can be blinking an LED, making a network request, measuring sensor readings, publishing sensor readings, etc.

To create tasks you need to follow the next steps:

1. Create a task handle. An example for Task1:

```
TaskHandle_t Task1;
```

2. In the `setup()` create a a task assigned to a specific core using the `xTaskCreatePinnedToCore` function. That function takes several arguments, including the priority and the core where the task should run (the last parameter).

```
xTaskCreatePinnedToCore(
    Task1code, /* Function to implement the task */
    "Task1", /* Name of the task */
    10000,  /* Stack size in words */
    NULL,  /* Task input parameter */
    0,  /* Priority of the task */
    &Task1,  /* Task handle. */
    0); /* Core where the task should run */
```

3. After creating the task, you should create a function that contains the code for the created task. In this example you need to create the `Task1code()` function. Here's how the task function looks like:

```
Void Task1code( void * parameter) {
 for(;;) {
   Code for task 1 - infinite loop
   (...)
 }
}
```

The for(;;) creates an infinite loop. So, this function runs similarly to the loop() function. You can use it as a second loop in your code, for example.

If during your code execution you want to delete the created task, you can use the vTaskDelete()function, that accepts the task handle (Task1) as argument:

```
vTaskDelete(Task1);
```

Plug and Play the System

Upon successful interfacing of all the hardware and software with the ESP-32 our last requirement was that we needed to make sure that the system runs without the help of any other devices like the laptop or anything.

Thus, to do this we followed the following

Step 1: Compile the code and upload the source code which you want to execute when the system boots up

Step 2: Connect the li-ion battery to the module

Step 3: Use the bot button in the Eesp 32 module to boot the module and for compiling the code

Step 4: Place the finger to Max3102 sensor to get the reading

Chapter VI

Evaluation

Introduction

This chapter discusses the success of a solution meeting the user requirement. It will also evaluate testing methods used during development. Finally, the chapter will go on discussing methods of solving ESP-32 boards and using Sensors discrepancy. This chapter will include all solutions provided by manufacturing companies.

Minimum Requirements

Minimum product output will be a smartphone running Blynk Application which can show various results from our sensors.We need a max30102 Sensor and Esp32 working board for the implementation of our project.

Troubleshooting

Troubleshooting This section of the chapter will tell you about the various we faced during the entire course of the project. The issues and their solutions are as follows:

1. Getting the hardware ready was a problem as some of the GSM may or may not be compatible with the ESP32 and since this was the first time, we were using GSM thus we had some confusion. To resolve this issue, we had to search many issues and thus after looking links we found that either SIM900 or SIM800 would be compatible thus since TTGO-TCALL have inbuild Sim800l we chose this module

2. EasyEDA software is quite good enough to have almost all of the components present but some of the components like the Max30102,TTGO-TCAL, were not present thus with help of the online version. We edit some other components which may suit our project

4. The GPS module Neo-6M was not detecting signal while inside the building so we had to use an extra antenna to resolve the issue

5.Sometime Our Internet Service provider(ISP) denied our access to the blynk server using wifi.So we needed to call them and made changes in our DNS.

Chapter VII

Conclusion

The aim of this project is to develop a product which helps to monitor their health conditions irrespective of their time or location. This would become a great help for people who are diagnosed with different health issues such as heart problems, Blood pressure abnormalities and so on. Since it is attached to their body along with an emergency push button and GPS which shares location of the patient to the app, relatives can access them easily without any delay in case of emergencies. This will help them to save their lives. So in conclusion we could say this is a life saver.

Future Work

In the future, we can make this product in large scale with better size reduction and we can upgrade it by attaching inside the headset itself as a small chip.

Chapter VIII

User’s Guide

About the Product

This product is an IoT-connected health care device, which is connected as a part of Bluetooth headphones. It consists of EKG sensor, pulse oximeter, temperature sensor, GPS and GSM to obtain various health-related data such as blood oxygen level, temperature, EKG and patients' live location.

Powering Up the system

To power up the system you will require a 5 v,2000 mah Li-ion battery. When the battery is connected to ESP-32 board the Power LED will be ON and after some time the top LEDs must start blinking in a repeated heartbeat pattern. To make the GSM module connect to the suitable cellular network. Insert a SIM card at the bottom of the module and switch on the supply. When the supply is on the power LED turns on. Next, next to the power LED press the boot button key for approximately 2 seconds this will connect the system to the desired network. The board will be booted and will run the code in real time.

Connecting to wifi and blynk server

We need to provide the authentic wifi ssid and password in the source code inorder to enable wifi in the Esp-32 module.The system will automatically upload the data to blynk server once it get the data.The user need to download the Blynk app in the smartphone to view the result.

How to place fingers on max30102 Sensor

Max30102 is a ensor used to measure Heart rate and blood oxygen in real time.The sensor have a glass part which emit IR and visible red light where we need to place the finger.Don't press hard on the glass part and users must fully cover the glass part for an accurate result

References

1. Github. (2020, April 14). Asksensors. https://github.com/asksensors/Connect-ESP32-GPS-MQTT
2. Techiesms. (2020, June 26). Github. https://github.com/techiesms/GPS-Tracker-via-GSM-using-TTGO-TCALL-Module-/tree/master/ GPS Tracker with Call SMS
3. Tech Explorations. (n.d.). What is Blynk? A quick review. https://techexplorations.com/guides/blynk/1-what-is-blynk/
4. Github. (2021, March 28). LilyGo T-Call SIM800 Series. https://github.com/Xinyuan-LilyGO/LilyGo-T-Call-SIM800
5. Components 101. (n.d). Temperature Sensor DS18B20 https://components101.com/sensors/ds18b20-temperature-sensor#:~:text=The%20DS18B20%20is%20a%201,solutions%2C%20mines%20or%20soil%20etc.&text=It%20can%20measure%20a%20wide,of%20±5°C.
6. ESP32-Powered Heart Rate and Pulse Oximetry Monitoring with MAX30102(Mar 30, 2021). Espressif. https://www.espressif.com/en/news/ESP32_and_MAX30102
7. Blynk help centre(n.d).Keep your void loop() clean http://help.blynk.cc/en/articles/2091699-keep-your-void-loop-clean
8. Interfacing Temperature sensor with ESP32(n.d).Random nerd Tutorials https://randomnerdtutorials.com/esp32-ds18b20-temperature-arduino-ide/
9. Starting Electronics. (2017, September 21). ESP-WROOM-32 Testing and First Use of ESP32. https://startingelectronics.org/articles/ESP32-WROOM-testing/#:~:text=The%20first%20and%20most%20basic,on%2Dboard%20regulator%20is%20working
10. K.Pradeep.(2012, December).E-SAP: Efficient-Strong Authentication Protocol for Healthcare Applications Using Wireless Medical Sensor Networks. https://www.researchgate.net/figure/Patient-monitoring-using-a-wireless-medical-sensor-network-in-a-hospital-environment_fig1_221967627?hcb=1
11. Andrade, D. (2019, July 17). Use a Raspberry Pi to Communicate with Amazon AWS IoT. hackster.io. https://www.hackster.io/dansku/use-a-raspberry-pi-to-communicate-with-amazon-aws-iot-e3752

5

12. Apple inc. (2020, December 21). Apple Watch Series 6. Technical Specifications. https://support.apple.com/kb/SP826?locale=en_CA
13. Embedded staff. (2002, July 1). Introduction to MISRA C. embedded. https://www.embedded.com/introduction-to-misra-c/
14. Espressif. (n.d.). FreeRTOS. Overview. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html
15. 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
16. Explore Embedded. (n.d.). Hello World with ESP32 Explained. Hello World. https://exploreembedded.com/wiki/Hello_World_with_ESP32_Explained
17. Kumar, S. (2017, August 1). ESP32 : Introduction to FreeRTOS. ICIRCUIT. https://icircuit.net/esp32-introduction-freertos/1297
18. 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/
19. Maxim Integrated. (2014, September 24). MAX30100. Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health. https://www.maximintegrated.com/en/products/sensors/MAX30100.html
20. MQTT. (n.d.). MQTT: The Standard for IoT Messaging. Why MQTT. https://mqtt.org
21. 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/
22. Starting Electronics. (2017, September 21). ESP-WROOM-32 Testing and First Use of ESP32. https://startingelectronics.org/articles/ESP32-WROOM-testing/#:~:text=The%20first%20and%20most%20basic,on%2Dboard%20regulator%20is%20working
23. Interfacing DS18B20 1-Wire Digital Temperature Sensor with Arduino, (n.d.). Last Minute Engineers. https://lastminuteengineers.com/ds18b20-arduino-tutorial/
24. Interfacing Temperature sensor with ESP32(n.d).Random nerd Tutorials https://randomnerdtutorials.com/esp32-ds18b20-temperature-arduino-ide/

25. Components 101. (n.d). Temperature Sensor DS18B20
https://components101.com/sensors/ds18b20-temperature-sensor#:~:text=The%20DS18B20%20is%20a%201,solutions%2C%20mines%20or%20soil%20etc.&text=It%20can%20measure%20a%20wide,of%20±5°C.
26. Components 101. (2018, March 6). Push button Switch.
https://components101.com/switches/push-button
27. Stempedia. (n.d.). What is Tactile Switch or Push Button?
https://thestempedia.com/tutorials/what-is-tactile-switch-or-push-buttons/
28. HDK. (2007. May 29). Tactile switches, Data Sheet.
https://components101.com/asset/sites/default/files/component_datasheet/Push-Button.pdf
29. Microcontrollers Lab (n.d.). Push button with ESP32 – GPIO pins as digital input
https://microcontrollerslab.com/push-button-esp32-gpio-digital-input/
30. Techiesms. (2020, June 26). Github.
https://github.com/techiesms/GPS-Tracker-via-GSM-using-TTGO-TCALL-Module-/tree/master/GPS Tracker with Call SMS
31. Priyanshu. (2020, June 26). Techiesms.
https://techiesms.com/iot-projects/gsm-gprs-based-gps-tracker-with-calling-sms-features/
32. Tech Explorations. (n.d.). What is Blynk? A quick review.
https://techexplorations.com/guides/blynk/1-what-is-blynk/
33. Adafruit. (n.d.). Adafruit Ultimate GPS.
https://learn.adafruit.com/adafruit-ultimate-gps?view=all#built-in-logging
34. UBlox. (n.d.). NEO-6 u-blox 6 GPS Modules, Data Sheet
https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf
35. Last minute Engineers. (n.d.). Interface ublox NEO-6M GPS Module with Arduino.
https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/
36. Guru. (2020, April). GPS Tracking using ESP32 and IoT Platform over MQTT, The Ask Sensors Blog.
https://blog.asksensors.com/iot-cloud-based-gps-tracking-esp32-gps-neo-6m-module/