IOT based Paralysis Patient Monitoring System

Table of Contents

CHAPTER 1: Introduction:

1.1 Background:

- Muscles that are paralyzed cannot be intentionally or independently moved. It may be either transitory or ongoing.
- The most frequent causes are multiple sclerosis, spinal cord damage, and stroke. Paresis, a severe weakness, can result in paralysis, a total loss of movement.
- The goal of treatment is to help a person adjust to life with paralysis by making them as independent as possible, even though there are cutting-edge methods for curing or treating paralysis patients.
- Our objective is to create a gadget that can retrain a patient's mobility while allowing them to use it independently and keeping the cost low enough for them to pay for it out of pocket.

1.2 ABSTRACT:

- The IOT-based paralysis patient health care system is designed to assist the patient in communicating various messages.
- Patient temperature, pulse and oxygen levels are sensed along with a few hand gestures and displayed on IOT server via Wi-Fi module.
- In this project we are sensing health parameters like Heart Rate, Body Temperature and SpO2 (Blood Oxygen) level.
- The health parameters are displayed on a web browser on Wi-Fi enabled devices like smart phones or Laptop.
- The heartbeat rate is determined physically by using a stethoscope where the likelihood of mistake is high on the grounds that the heartbeat rate is in the middle of 70 to 90 every moment whose event is under 1 sec, so this gadget can be considered as an excellent option.

- The IOT-based paralyzed patient health care system is a system meant to assist the patient in communicating different messages to physicians, nurses, or loved ones while sitting at home or in the workplace over the internet.
- To deliver this capability, the system employs microcontroller-based circuitry. It employs a hand motion recognition circuit as well as a receiver and transmitter circuit.
- The hand motion circuit detects hand motions with an accelerometer and wirelessly transmits this information to the receiver system through Rf.
- The receiver system is designed to receive and process these commands and transmit the data online over to the Blynk IOT platform.
- The Blynk IOT, then displays this information online, to receive patient information.

1.3 Problem statement:

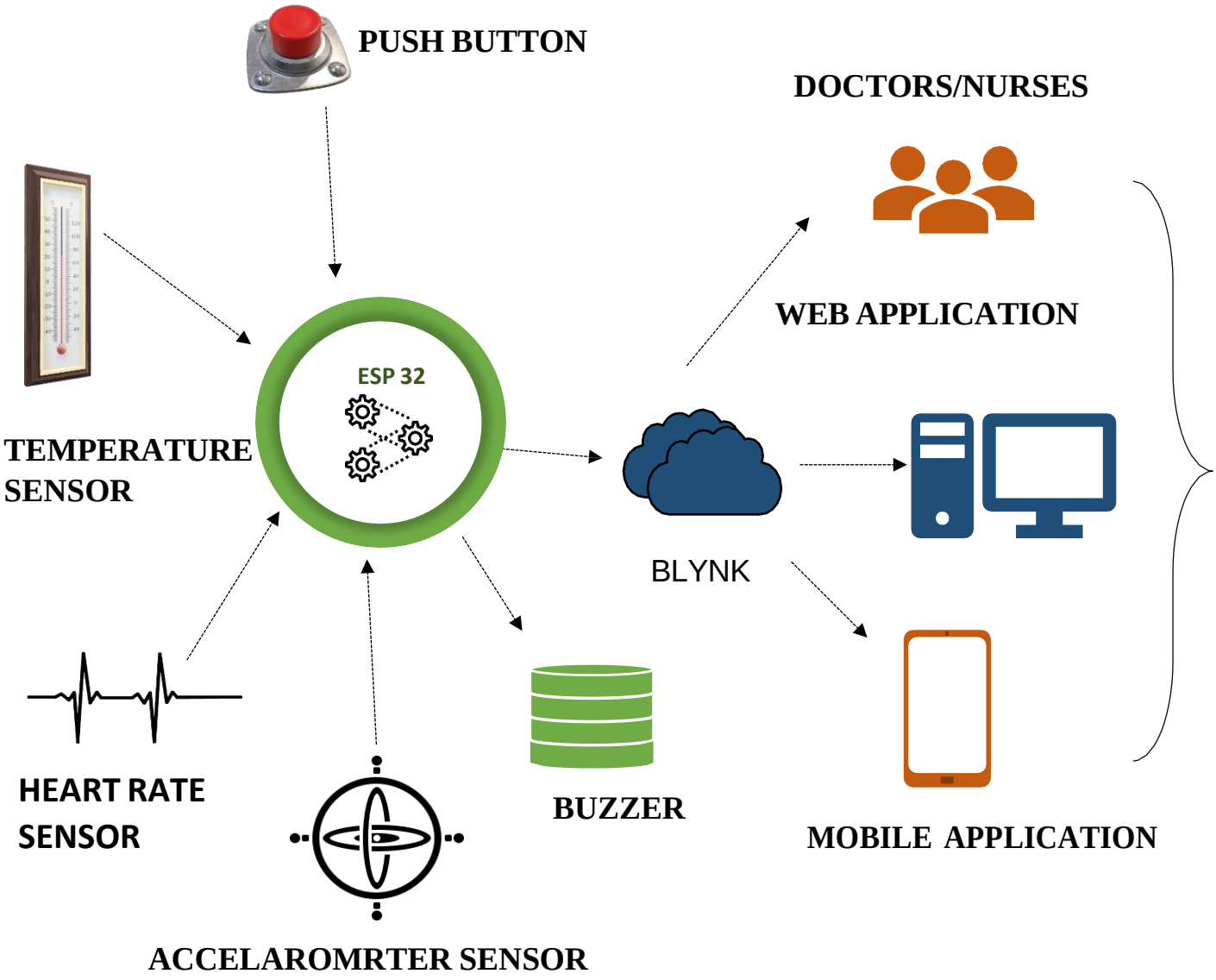
- Everyone is aware that paralysis is the inability to independently move your muscles. Unfortunately, even their closest friends and family members cannot understand their basic needs.
- Additionally, it is crucial to continuously monitor their health.
- Consequently, we need a system that informs us of their fundamental needs.
- Taking this point as priority to our project we came up with a proposed system as mentioned below.

1.4 Proposed system:

- The IOT-based paralyzed patient health care system is a tool created to help patients connect with loved ones or medical professionals online while they are at home or at work.
- It will make it easier for them to interact with others.
- The lives of those who have paralysis could one day be improved by this technology.

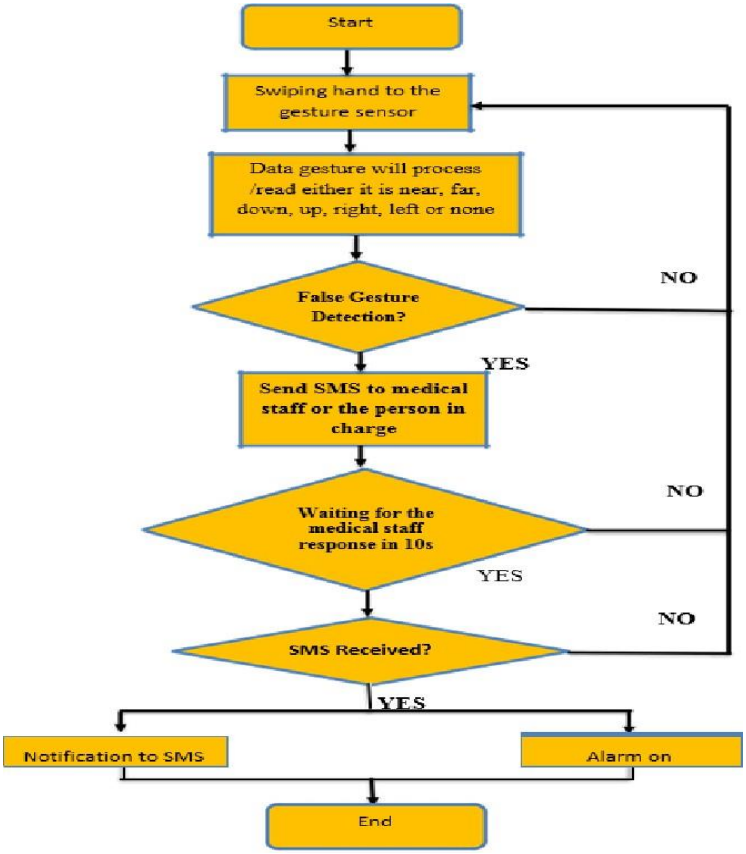
- Wi-Fi-based technology is used by the system to achieve this functionality.
- Both a healthcare monitoring circuit and a hand motion recognition circuit are used.
- Utilizing an accelerometer, the hand motion circuit detects hand movements. A limited number of sensors are being employed to keep track of the patient's health.
- A Wi-Fi module is used to send this data to a server.
- The messages are displayed to the affected parties using the IOT platform Blynk.

CHAPTER 2: System Architecture



System Architecture Description:

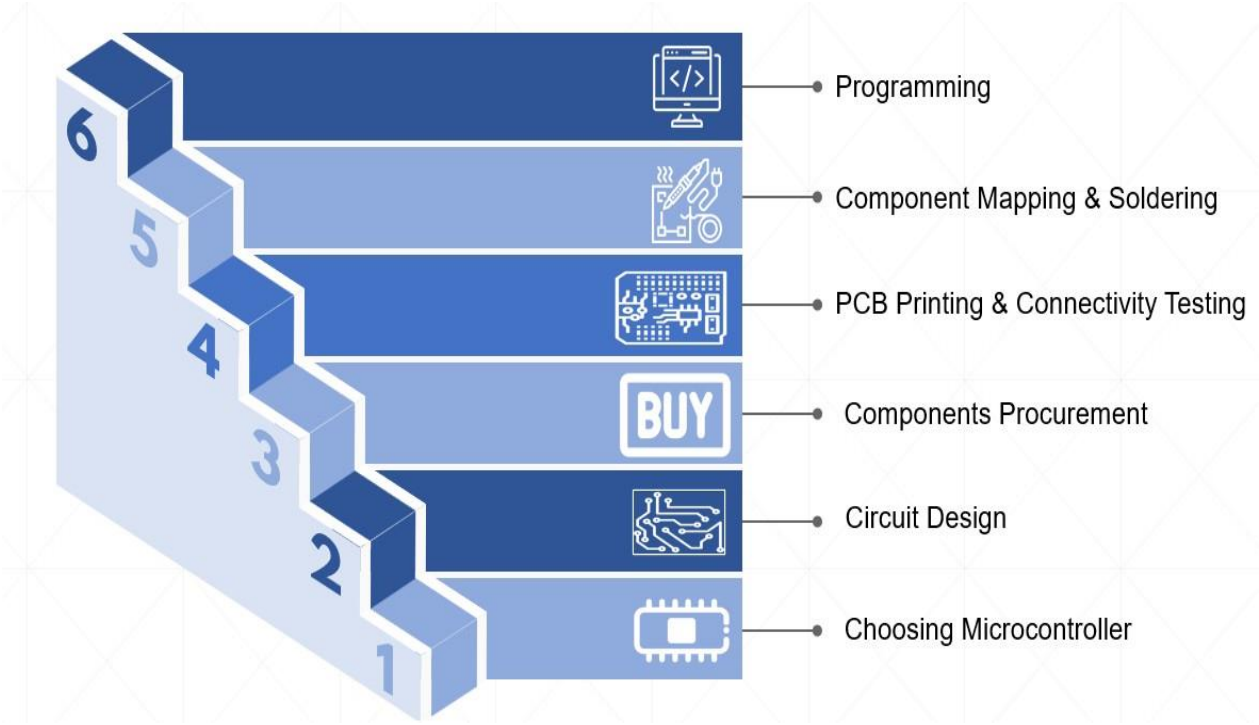
1. Temperature sensor: DS18B20 temperature detector is used to read patient body temperature.
2. Heart Rate Sensor: MAX30100 Pulse Oximeter and Heart Rate sensor is used to measure both blood oxygen levels and heart rate
3. Accelerometer Sensor: ADXL335 can measure both static and dynamic acceleration caused by motion, shock, or vibration, as well as gravity.
4. Push (Panic) Button: A panic alarm is an electrical device used to alert someone in an emergency.
5. Buzzer: When the Panic Button is pressed, this sound is played.
6. ESP32: ESP32 is generally available for low price, low-power system on a chip microcontroller in addition to integrated Wi-Fi and dual mode Bluetooth.
7. Blynk: It can manage equipment remotely, display sensor data, record data, and visualize it.
8. End Applicants: Doctors, nurses, and loved ones can access patient statistics via the internet while sitting at home or at work.



FSMD OF HAND GESTURE SENSING & NOTIFYING

CHAPTER 3: DESIGN DETAILS

3.1 PROJECT DEVELOPMENT STAGES:



1. Choosing MicroController:

- We have chosen ESP 32 Microcontroller, as it can provide an interface in between Wifi and Bluetooth functions on the main functionality of the processor.
- If we have connected salvage device with out our notice, then ESP32 decrease the connectivity in-between stack over head and the main application of the processors.

2. Circuit Designing:

- We have used a software known as EasyEDA.
- This software was helpful to design our circuit diagram and try an accurate simulation.

3. Components Procurement:

- By using EasyEDA simulator, we have chosen different types of components and by different magnifications, then taken accurate values of the outputs and finalized the components.
- After finalizing the components, we have ordered the components through online markets.

4. PCB Printing and Connectivity Testing:

- Our next step is to design the PCB boarding printing and connect all the components on the PCB.
- After connecting the components, test each component by using a multi-meter.

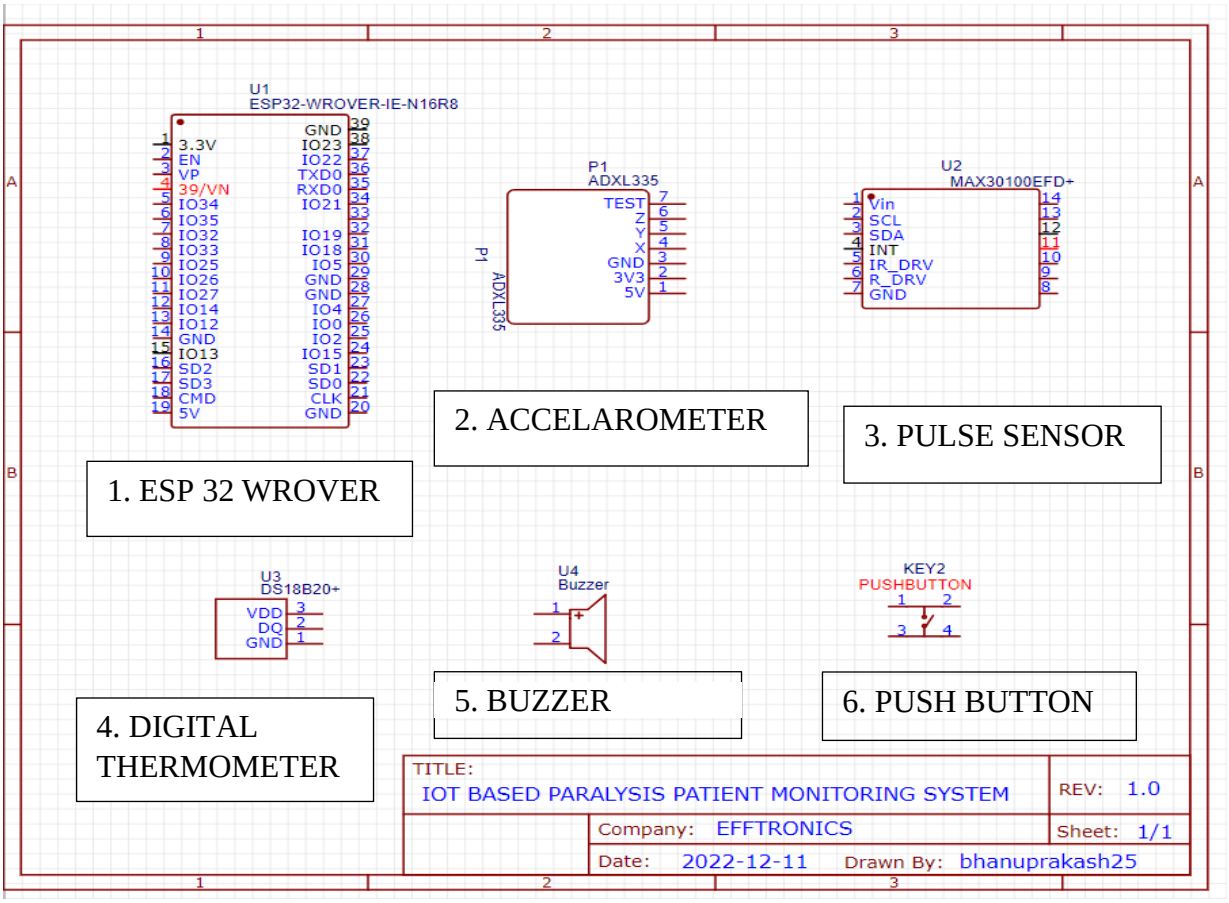
5. Components Mapping and soldering:

- Components are to be plotted properly without any over lapping of the Jumper wires.
- Soldering need to be done without any short circuiting.

6. Programming:

- We have chosen Arduino IDE, to code our program in C++ language.

3.2 Components:



3.2.1. ESP32 WIFI Microcontroller:

The ESP32 is a chip that uses TSMC's 40nanometers reduced power technology to combine 2.4GHz Wi-Fi and Bluetooth. Optimized for optimal power and RF efficiency for lot of applications and even more power scenarios. This Chip series also has the ESP32-S0WD (NRND), ESP32-U4WDH, ESP32-D0WD (NRND), ESP32-D0WDQ6 (NRND), ESP32-D0WDQ6-V3 (NRND), ESP32-D0WD (NRND), ESP32-D0WDQ6 (NRND), ESP32-D0WD-V3

The chip version v1 is used in the ESP32-D0WD (NRND), ESP32-D0WDQ6 (NRND) and ESP32-S0WD (NRND)

ESP32-D0WDR2-V3The ESP32-D0WD-V3, ESP32-U4WDH ESP32-D0WDR2-V3, and ESP32-D0WDQ6-V3 use chip versions v3.0 or v3.1 (NRND). Section 7 contains model numbers and

ordering information. For more information on chip adjustments, see the ESP32 Processor Revision v3.0 User Guide and this Series SoC Revisions., less than

- ESP32-S0WD (NRND), ESP32-D0WD (NRND) and ESP32-D0WDQ6 (NRND) are based on chip version v1 or chip version v1.1.
- ESP32-D0WD-V3, ESP32-D0WDR2-V3, ESP32-U4WDH, and ESP32-D0WDQ6-V3 (NRND) are based on chip version v3.0 or chip version v3.1. See Section 7 for part numbers and ordering information. For details on chip revisions, see the ESP32 Chip Revision v3.0 User Manual and ESP32 Series SoC Revisions.

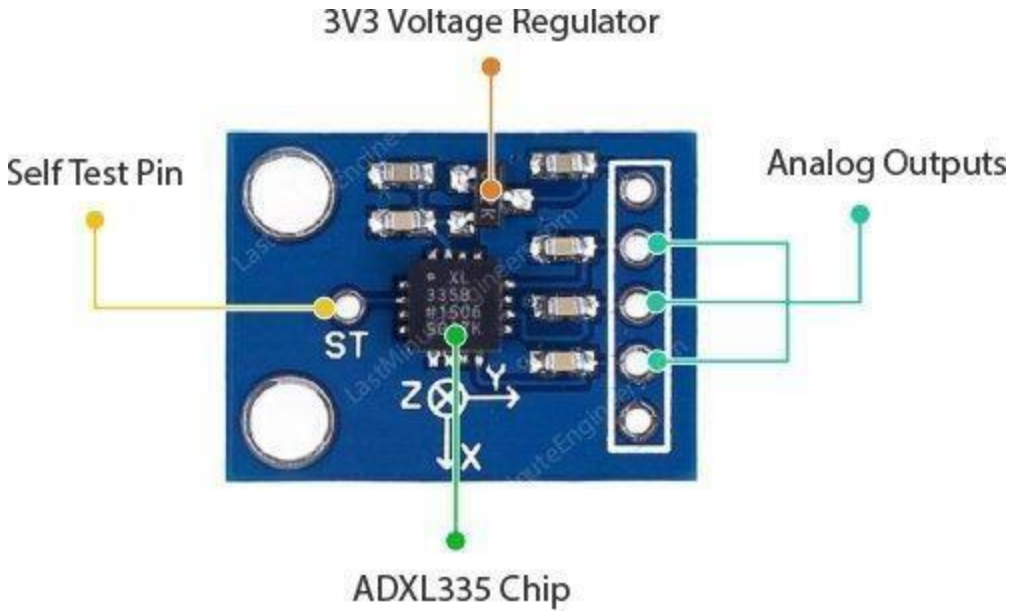
Specifications:

Operating Voltage	2.2V to 3.6V
GPIO	36 ports
ADC	14 ports
DAC	2 ports
Flash memory	16 Mbyte
SRAM	250 Kbyte
Clock Speed	Up to 240 MHZ
Wi-Fi	2.4 GHz
Sleep Current	2.5 µA

Figure:

3.2.2 ADXL335 Accelerometer Sensor

- An accelerometer is a gadget that measures the acceleration of any object. It monitors acceleration using analog inputs in three dimensions, such as X, Y, and Z. It is a low-noise and low-power gadget.
- When used to measure acceleration, it is interfaced with any form of controller, such as a microprocessor or an Arduino, for example. It is mostly utilized in construction machines such as drilling, driving piles, and demolition, as well as human activity machines like running, walking, dancing, and skipping.
- It is easily accessible in the market or online. Figure shows a basic ADXL 335 accelerometer.



- A small, low-power, low-noise triple axis accelerometer from Analog Devices - ADXL335 - is at the heart of the module. It can measure both static and dynamic acceleration caused by motion, shock, or vibration, as well as gravity.
- Every pin of the ADXL335 is broken out to a 6-pin, 0.1″ pitch header in this breadboard-friendly module, including 3 analog outputs for X, Y, and Z axis measurements, 2 supply pins, and a self-test pin.

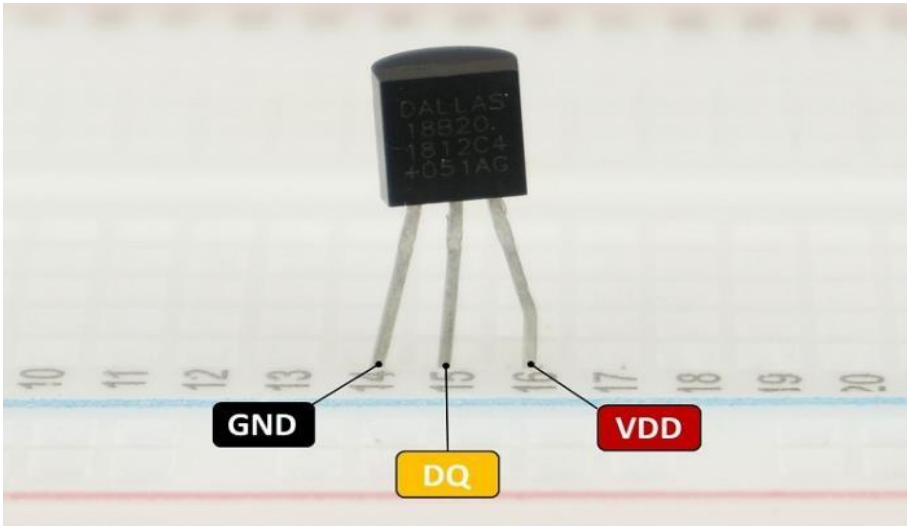
3.2.3 MAX30100 PULSE OXIMETER AND HEART RATE SENSOR:

- In this project, we will connect a MAX30100 Pulse Oximeter Sensor to an Arduino. The MAX30100 Sensor can measure both blood oxygen levels and heart rate. To view the SpO2 and BPM values, we can use any display, such as a 16x2 LCD display. SpO2 (blood oxygen concentration) is expressed as a percentage, whereas BPM (heart rate/pulse rate) is expressed as a few beats per minute. The MAX30100 is a sensor solution for pulse oximetry and heart rate monitoring. It detects pulse oximetry and heart rate signals by combining two LEDs, a photodetector, optimized optics, and low-noise analog signal processing. This sensor can be used with any microcontroller, such as Arduino, ESP8266, or ESP32, to easily measure the patient's health parameters.

- The MAX30100 is powered by 1.8V and 3.3V power supplies and can be turned off via software with negligible standby current, allowing the power supply to always remain connected.

3.2.4 DS1820B TEMPERATURE SENSOR:

- The DS18B20 temperature detector is a digital temperature detector with a single wire. This means that communicating with the Arduino requires only one data line (and GND).
- It can be powered by an external power source or by determining power from the data line (known as "parasite mode"), which eliminates the requirement for an external power source.
- The detectors have a reported delicacy of/-0.5 deg C in the temperature range of -10 deg C to 85 deg C.



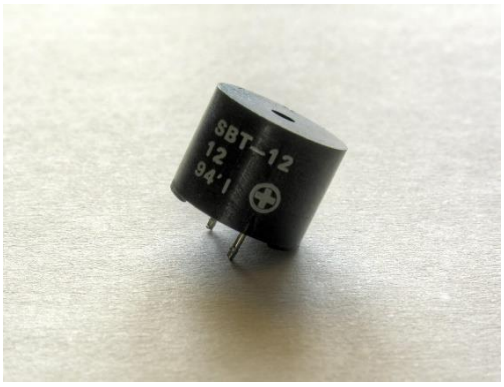
DS18B20	Arduino
GND	Ground
DQ	Any digital pin (with 4.7k Ohm pull-up resistor)
VDD	5V (normal mode) or GND (parasite mode)

3.2.5 BUZZER:

- A buzzer is a simple but effective way to add sound to our project/system.
- It is a popular component in many electronic applications due to its small and compact 2-pin shape, which allows it to be easily used on breadboards, Perf Boards, and even PCBs.
- The internal oscillating circuit generates sound.
- It is easily customizable with the help of additional circuits to fit in our application. Simply connect this buzzer to a 4V to 9V DC power supply to use it.
- A standard 9V battery can also be used, but a controlled +5V or +6V DC supply is recommended.

Buzzer Pin Configuration

Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit



3.2.6 PUSH(PANIC) BUTTON

- A panic alarm is an electrical device designed to help notify someone in an emergency where there is a hazard to people or property.
- A concealed panic alarm button is frequently, although not always, used to activate a panic alarm.

- These buttons can be linked to a monitoring center or a local monitoring station through a silent alarm or an audible bell/siren.
- The alert can be used to summon help from local security, police, or emergency services.
- Some systems can additionally turn on closed-circuit television to record or evaluate the event.

3.3 Software Description:

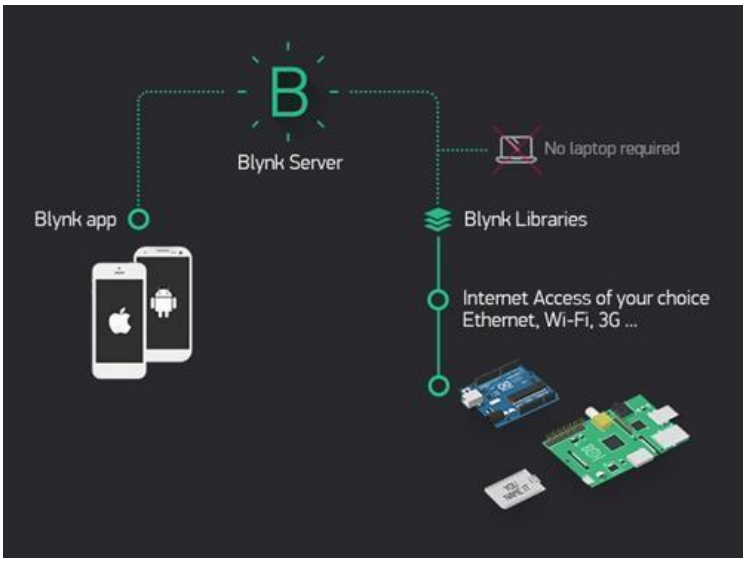
3.3.1 Blynk Server:

- Blynk is designed for the creation of smartphone applications that integrate with a variety of other applications.
- The Blynk platform's primary goal is to make developing mobile phone applications as simple as possible.
- Blynk was created with the Internet of Things in mind.
- It can remotely manage equipment, display sensor data, record data, visualize it, and perform several other exciting tasks.
- The platform consists of three primary components:
    - Blynk App - allows you to develop amazing interfaces for your projects by utilizing the numerous widgets we provide.

    - Blynk Server - responsible for all smartphone-to-hardware communication You can use our Blynk Cloud or host your own private Blynk server locally. It's open source, can manage thousands of devices, and can even be run on a Raspberry Pi.

- ❖ Blynk Libraries - Enable communication with the server and handle all incoming and outgoing commands for all popular hardware platforms.
- ❖ Blynk Platform has two ways to communicate
- ❖ Smartphone App: The smartphone app is an app editor. It is used to create Blynk projects, each of which can contain many widgets and link to different devices. Projects can also be shared on this site.

- ➢ Server: Blynk library for JavaScript and Python is available. Cloud Server is simple to use and begins instantly on any host having a Java runtime environment.
- ➢ We can have complete control over our data and access by using a private server.
- ➢ It has a low latency and a high level of security.
- ➢ Blynk is accessible via online and smartphone.
- ➢ The benefit of using a smartphone application is that it informs us whenever we need it.
- ➢ As we receive notifications on our smartphones, we have a better interface.
- ➢ The Blynk App is available for both iOS and Android devices.



3.3.2 **Arduino IDE:**

- The Arduino Integrated Development Environment (IDE), sometimes known as the Arduino Software (IDE), features a code editor, a message area, a text terminal, a toolbar with basic function buttons, and a variety of menus.
- It communicates with the Arduino hardware to upload and interact with programs.
- The Arduino code is written in C++, with a few unique methods and functions that we'll go through later.
- C++ is a simple to learn programming language.
- When you make an Arduino sketch (code file), it is analyzed and compiled to machine language.
  - The Arduino programming language is separated into three sections:

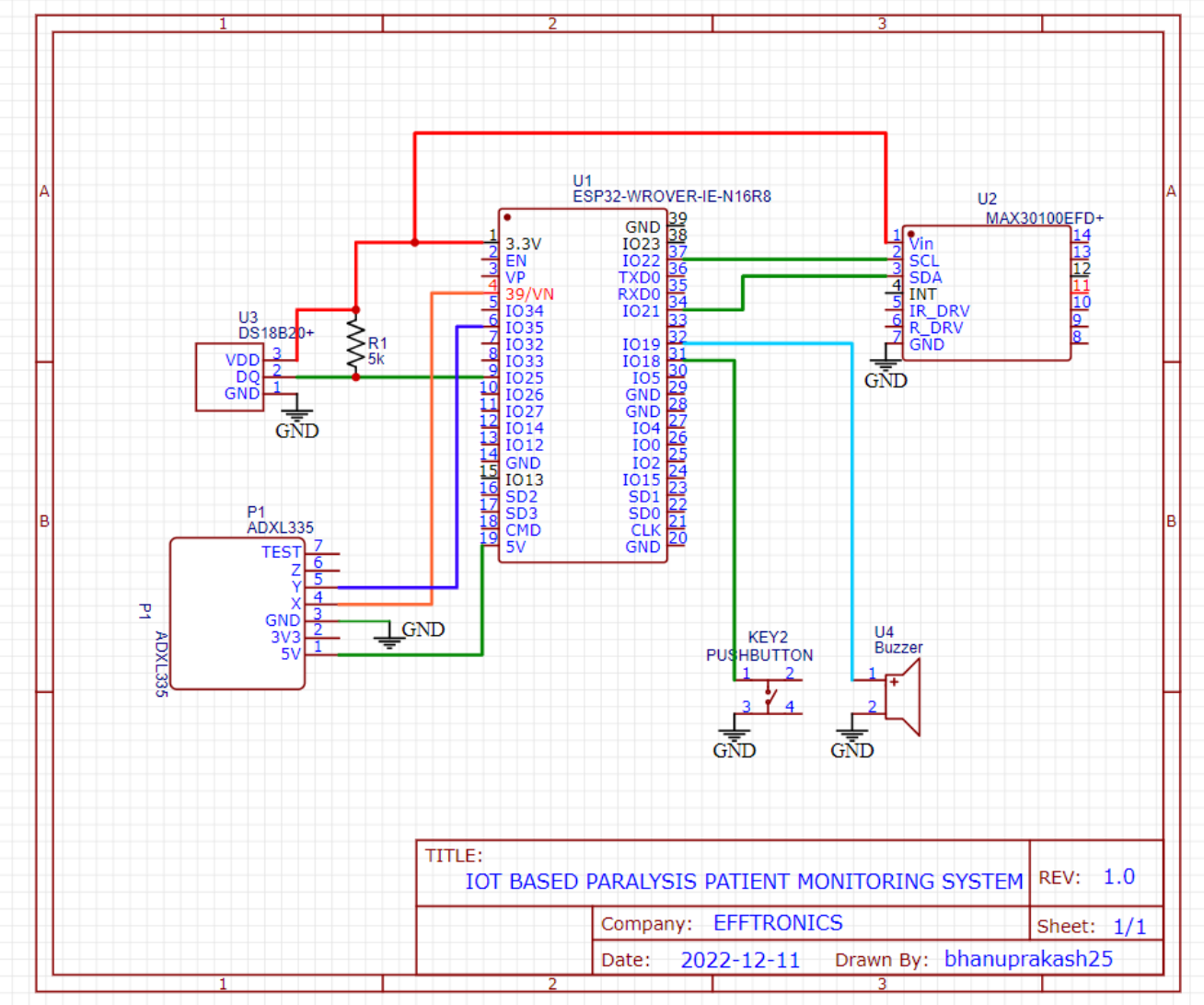
•Functions: For controlling the Arduino board and performing computations.

•Values (variables and constants): Arduino data types and constants.

•Structure: The elements of Arduino (C++) code.

CHAPTER 4: Prototyping and Demonstration

4.1 CIRCUIT DESIGNING:



For any embedded system microcontroller is heart of it.

- It performs all the arithmetic and logical operation and control all the peripheral device connected to it according to the code written.
- Our project main aim is to monitor the paralyzed patient health.

- To serve the purpose of heartrate and SpO2 measurement, Max30100 sensor is employed and for temperature measurement DS18b20 sensor is used.
- The heartrate sensor also called pulse sensor monitors the variation in blood volume in any part of the body which results in a change in the quantity of light passing froms that organ.
- Serial clock (SCL) and Serial Data (SDA) pins of pulse sensor are connected to I/O ports of ESP32.
- Temperature sensor is connected of ESP32 to give the change in the voltage in the sensor.
- The main purpose of an accelerometer sensor is that the patient can interact with others with the specific gesture.
- The ADXL335 operates on the capacitive principle. It operates on the concept that the capacitance within the sensor varies as an acceleration is applied.
- Xout and Yout of the sensor are ESP32.
- Push button is used as a panic button to notify if any help is required.
- Buzzer is used to alarm people nearby whenever help is required.

4.2 OPERATION:

- Arduino IDE must be programmed beforehand so that it can carry out the specified task.
- After programming, the board is connected to computer by using USB serial converter. Hardware configurations like pins are to be programmed in the Arduino IDE.
- ESP32 must be flashed via USB prior to use. Then, we require blynk libraries for programming, and while programming, we must input our internet-connected Wi-Fi router's SSID and password so that the Wi-Fi module can connect to the server we intend to use for IoT.
- Blynk is the IOT platform that we are using in our project. After logging in into Blynk a new project is created. Board and hardware details are given into the created blynk.
- An Authorization called Auth token is a unique identifier which is generates when a project is being created. This is required to link the hardware to our smartphone/web application.
- Auth token should be given in the code and upload to Wi-Fi module connected.

- When the patient is in contact with the heartrate and temperature sensor, they sense the respective values. The accelerometer senses the orientation of the hand (when connected to the

hand). ESP32 which is the main core of the project, analyzes the data sensed by all the sensor as well as the push button state.

- It uses the predefined sensor libraries of pulse and temperature sensors to convert the sensed values into temperature and heartrate of the patient.
- Also, in the case of accelerometer it converts the orientation into basic messages thet the patient wants to convey.
- This processed data is sent to IOT server i.e., Blynk using the SSID, password and Auth token.

Blynk server displays the pulse rate, oxygen levels and temperature values.

Along with that Blynk mobile application can notify others the hand movement messages and panic button messages (Help messages) to the users.

4.3 OUTPUT AND DEMO:

DEMO VALUES THAT ARE DISPLAYED ON BLYNK WEB APPLICATION:

NOTIFICATIONS ON BLYNK MOBILE APP:

4.4 CODE:

```
#define xpin 39

#define ypin 35

#define buz 19

#define sw 18

#define BLYNK_TEMPLATE_ID "TMPL4zlwX9hK"

#define BLYNK_DEVICE_NAME "Temperature"

#define BLYNK_AUTH_TOKEN "m_83vktKCtf21JNd58ogHYv-tGIn7rg3"


#include <OneWire.h>

#include <DallasTemperature.h>
```

```
const int oneWireBus = 25;

OneWire oneWire(oneWireBus);

DallasTemperature sensors(&oneWire);


#include <Wire.h>

#include "MAX30100_PulseOximeter.h"

#define BLYNK_PRINT Serial

#include <WiFi.h>

#include <WiFiClient.h>

#include <BlynkSimpleEsp32.h>


#define REPORTING_PERIOD_MS 1000


char auth[] = BLYNK_AUTH_TOKEN;            // You should get Auth Token in the Blynk App.

char ssid[] = "701_ABHRSU";                        // Your WiFi credentials.

char pass[] = "7138700654";


PulseOximeter pox;

float BPM, SpO2;

uint32_t tsLastReport = 0;
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  sensors.begin();
  pinMode(buz,OUTPUT);
  pinMode(sw,INPUT_PULLUP);
  Blynk.begin(auth, ssid, pass);


    if (!pox.begin())
    {
        Serial.println("FAILED");
        for(;;);
    }
    else
    {
        Serial.println("SUCCESS");
       // pox.setOnBeatDetectedCallback(onBeatDetected);
    }
}
int x,y,SpO22;
void loop() {
  // put your main code here, to run repeatedly:
    pox.update();
```

```
 Blynk.run();
 BPM = pox.getHeartRate();
 SpO2 = pox.getSpO2();
    if(digitalRead(sw)==LOW)
    {
      digitalWrite(buz,HIGH);
      Serial.print("help me...!");
       Blynk.logEvent("panic","Please Help Me ... !!!!");
      delay(2000);
    }
    else
    {
      digitalWrite(buz,LOW);
    }

if (millis() - tsLastReport > REPORTING_PERIOD_MS)
{

    Serial.print("Heart rate:");
    Serial.print(BPM);
    Serial.print(" bpm / SpO2:");
    Serial.print(SpO2);
    Serial.println(" %");
```

```
        Blynk.virtualWrite(V1, 120.6);
        Blynk.virtualWrite(V2, 94);
        tsLastReport = millis();
    }
    else
     mydata()
}
void mydata()
{
sensors.requestTemperatures();
    float temperatureC = sensors.getTempCByIndex(0);
    float temperatureF = sensors.getTempFByIndex(0);
    Serial.print("temperature: ");
    Serial.print(temperatureC);
    Serial.println("ºC");
    Blynk.virtualWrite(V3,temperatureC);
x = analogRead(xpin);
y = analogRead(ypin);
Serial.print("X= ");
Serial.println(x);
Serial.print("Y= ");
Serial.println(y);
delay(250);
```

```
if((y>=2000)&&(y<=2400))
    {
      Serial.println("I am thirsty");
      Blynk.logEvent("notify1","I am thirsty ... !!!!");
      delay(2000);
      }
    else if((y>=1300)&& (y<=1500))
    {     Serial.println("I am hungry");
      Blynk.logEvent("notify2","I am hungry ... !!!!");
      delay(2000);
      }
      else if((x>=1400) && (x<=1600))
    {     Serial.println("I need to use restroom");
      Blynk.logEvent("notify3","I need to use restroom ... !!!!");
      delay(2000);
      }
      else if((x>=2000) && (x<=2400))
    {     Serial.println("I want to sit");
      Blynk.logEvent("notify3","I want to sit....!!!!");
      delay(2000);     }
      else
      {
        Serial.println("stop");}}
```

CHAPTER 5: CONCLUSION:

This project is very useful paralysis patients monitoring. If sensors are kept connected, then we can get continuous data of temperature and pulse values. Also, we can further add other sensors like blood pressure etc., to monitor health parameters continuously. This project is very useful paralysis patients monitoring. If sensors are kept connected, then we can get continuous data of temperature and pulse values. Also, we can further add other sensors like blood pressure etc., to monitor health parameters continuously. Furthermore, we can also implement an idea where the doctors or nurses get notified or get an SMS immediately when the health parameters reach the threshold values so that they can act upon immediately.

One disadvantage using hand gestures for sensing messages is that hand movements cannot be given fully paralyzed patients. Hence the hand sensing feature cannot be implemented on them. Also, this system requires internet for communication. Hence it is of no use where there is no internet connections.

However, this would be advantageous to patient's caretakers when they have access to internet and when they are able to give few gestures.

REFERENCES:

https://ieeexplore.ieee.org/document/9441910

http://www.ijirset.com/upload/2020/october/23_HealthBox.PDF

https://www.dnatechindia.com/basic-working-pulse-oximeter-sensor.html

https://www.hnhcart.com/blogs/sensors-modules/a

https://docs.blynk.io/en/