An-Najah National University

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Computer Engineering Department

Hardware Graduation Project

The Devoted Caregiver

Students:

Laila Abu Safiya

Sojod Jamous

Supervisor:

Emad Natsheh

Acknowledgment

I would like to take this opportunity to express my heartfelt gratitude and appreciation to the following individuals and organizations who have played a significant role in the completion of my graduation project.

First and foremost, I would like to express my deepest appreciation to my supervisor, Dr.Emad Natsheh, for their unwavering guidance, invaluable insights, and continuous support throughout the entire duration of this project. Their expertise, patience, and dedication have been instrumental in shaping the direction of my research and ensuring its success.

I am also grateful to the faculty members and staff of the Computer Engineering department for their provision of excellent academic resources, facilities, and a nurturing environment that has fostered my intellectual growth. Their commitment to education and their passion for their respective fields have been a constant source of inspiration.

I would like to extend my sincere thanks to my classmates and peers who have been an integral part of this journey. Their camaraderie, engaging discussions, and willingness to collaborate have enriched my learning experience and made this project more enjoyable. I am truly grateful for their friendship and support.

My deepest appreciation goes to my family for their unconditional love, encouragement, and belief in my abilities. Their unwavering support and sacrifices have been the driving force behind my accomplishments. I am forever indebted to them.

Completing this graduation project has been a challenging yet fulfilling endeavor, and it would not have been possible without the unwavering support and contributions of these exceptional individuals and organizations. Thank you from the bottom of my heart for being a part of my academic journey and for helping me achieve this milestone.

Abstract

Cerebral palsy is a challenging medical condition with no known cure. The sole form of treatment available for individuals with cerebral palsy is physical therapy. This project introduces an innovative mobility chair that not only provides much-needed comfort and assistance but also incorporates essential aspects of physical therapy into the daily routine of these patients. The chair offers comprehensive massage therapy, providing back massages at different speeds to stimulate blood circulation, effectively addressing a common problem among patients with limited mobility. It also includes foot massage therapy using specially designed foot wraps, especially around the heel area according to Chinese foot mapping principles, which helps facilitate bowel movements, a recurring challenge for many individuals. The chair features a multi-touch movable footrest that stimulates the nerve pathways in the feet, facilitating neural communication with the brain. In addition, there is a hand-held balloon in the chair that gradually inflates and deflates to relieve tension in the hands, which helps relieve muscle spasms and spasms. For safety, the chair is equipped with a headrest sensor to ensure proper head position, preventing choking or discomfort to the patient. It also includes capabilities for monitoring health status, with a built-in wristwatch that measures temperature and heart rate, provides regular reports to the patient's family, and alerts healthcare providers in the event of irregular readings. The chair's functions are activated via a keypad, with each feature programmed at pre-set time intervals to ensure patient safety. In case of any issues related to head or hand balloon position, alerts are sent to family members for immediate attention. This mobility chair offers a comprehensive solution for patients with cerebral palsy and motor paralysis. It bridges the gap between daily life and essential physical therapy, providing comfort and support, all while enhancing the overall quality of life for both patients and their caregivers.

Furthermore, the project is implemented using Arduino and ESP32WROOM, utilizing DC motors and sensors, switches, keypads, and a SIM900 module to enable seamless operation and communication, showcasing a synergy between modern technology and compassionate care.

Contents

List of Figures

Chapter 1

Introduction

Cerebral palsy includes a group of motor conditions that do not deteriorate, are not contagious, and cause physical disability during human development in different parts of the body related to the performance of motor functions, and it is one of the most dangerous diseases that steal from humans the ability to move and, in some cases, loss of hearing or even speech.
Treatment of this disease depends on the rate of paralysis and the affected area. In the case of cerebral palsy, the damage occurs in the brain cells that do not regenerate and cannot be treated so far and can only be treated with physical therapy.
It was found that the number of patients with cerebral palsy in the city of Nablus is 144, according to the World Federation of the Handicapped.

1.1 Problem Statement

The main problem is that there is no type of wheelchair that supports and helps patients with cerebral palsy, as these patients suffer from weak movement and muscle weakness in various parts of the body, and they also need permanent physical therapy and frequent follow-up of their vital signs.

1.2 Objectives

The Devoted Caregiver aims to be easy to use. It will also facilitate and assist parents at home and provide assistance to them and the patient. It will provide physical therapy for the patient, monitor temperature and pulse, and send reports and warnings to both the family and the doctor in emergency cases.

1.3 Scope of work

1.*Physiotherapysystem* : This system massages several areas of the body, including the back and feet, to stimulate blood circulation and nerve impulses.

2.*Biosystem* : This system checks the patient's vital signs, which are the temperature and heart rate.

3.*Alarmsystem* : This system is responsible for sending warnings to families and doctors in the event of an emergency and sending them daily reports.

1.4 Significance

The Devoted Caregiver seamlessly integrates user-friendly medical care with cutting-edge IoT technology, mobile app connectivity, and an adjustable Physiotherapy system, all conveniently accessible from the comfort of home.

Chapter 2

Constraints and Earlier work

2.1 Constraints

1- Hardware Limitations When trying to execute two actions simultaneously on an Arduino Mega for different durations, it limits true simultaneous execution. because it operates on a single core.
2- Pulse Sensor(SEN-11574) was not giving accurate results.
3- The difficulty of 3D printing massage rollers to be of the appropriate size to provide the appropriate function.

2.2 Earlier work

1. An electronics course covering a range of topics related to electronic technology and systems.
2. A course on microcontrollers that covers both their theoretical underpinnings and real-world applications.
3. The Microcontrollers Lab covers topics like controlling stepper motors and provides hands-on experience with Arduino and its functionalities. 4. Scientific research and critical thinking teach students how to read scientific journals and write research papers using contemporary tools like LaTeX.

Chapter 3

Literature Review

Cerebral palsy is a disease that limits a person's ability to move and carry out normal daily functions. This disease was first discovered in 1860 by Dr. William Little. The main cause of this disease is the lack of oxygen in the brain, which leads to cell death. One of the reasons for the lack of oxygen is exposure to suffocation after childbirth, for example, or a traffic accident. Patients with cerebral palsy suffer from many problems, as they suffer from frequent cramps in their limbs, in addition to difficulty in excreting, and disturbances in heart rhythm and temperature. These problems vary in degree from one patient to another, but these symptoms persist throughout the patient's life.

To alleviate the suffering of cerebral palsy patients, traditional solutions are resorted to, such as bringing a physical therapist, as this helps relieve paand in, muscle stiffness, and improve movement in general. This therapist is brought periodically every week or perhaps periodically to prevent permanent muscle spasms. It can also prevent future health problems such as muscle tightness and to solve the problem of difficulty in passing, laxatives are taken to help facilitate stomach emptying.

From the previous solutions, it was found that these solutions, such as medicines, may not be permanently available to patients, and families may not have the money permanently to bring these medicines, in addition to the difficulty of bringing a physiotherapist for some cases. Parents may not inadvertently notice a high fever and heart attack. The patient has if we need something that alerts us if there is a change in temperature and heart rate. Therefore, it is better to find a solution to these problems that is obtained in an integrated way to alleviate the suffering that cerebral palsy patients suffer from.

Chapter 4

Methodology

4.1 System Architecture

4.1.1 Full Body

We use the chair for patients with cerebral palsy as a basic part to stabilize the moving parts and gears for the massage and to place the head support part on it.

Figure 4.1: Full Body

4.1.2 Back Massages

Massages vary in speed and technique to improve blood circulation, effectively addressing a common issue for patients with limited mobility.

Figure 4.2: Back Massages

4.1.3 Foot Massage

Foot massage therapy employs specialized foot wraps, particularly targeting the heel area based on Chinese foot mapping principles. This approach aids in promoting bowel movements, which can be a persistent challenge for many individuals. The chair is equipped with a versatile footrest that uses multiple touchpoints to activate nerve pathways in the feet, fostering enhanced neural communication with the brain.

Figure 4.3: Foot Massage

4.1.4 Hand-held Balloon

The chair incorporates a handheld balloon that gently expands and contracts, alleviating tension in the hands and effectively easing muscle spasms. In case of any issues related to hand balloon position, alerts are sent to family members for immediate attention

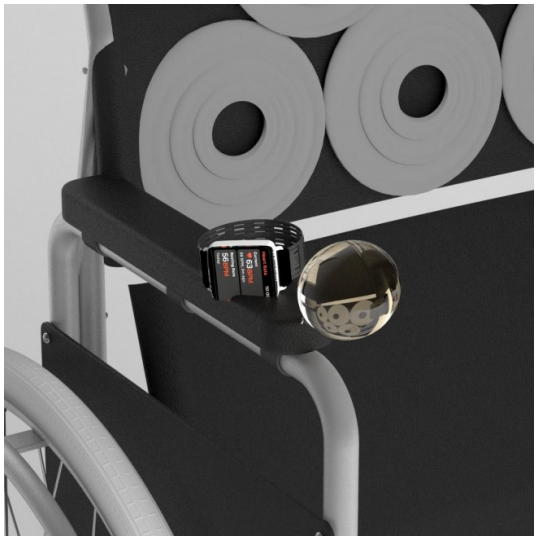

Figure 4.4: Hand-held Balloon

4.1.5 Built-in Wristwatch

It monitors both temperature and heart rate, offering routine reports to the patient's family using IoT and promptly notifying healthcare providers of any irregular readings throw emergency calls to parents.

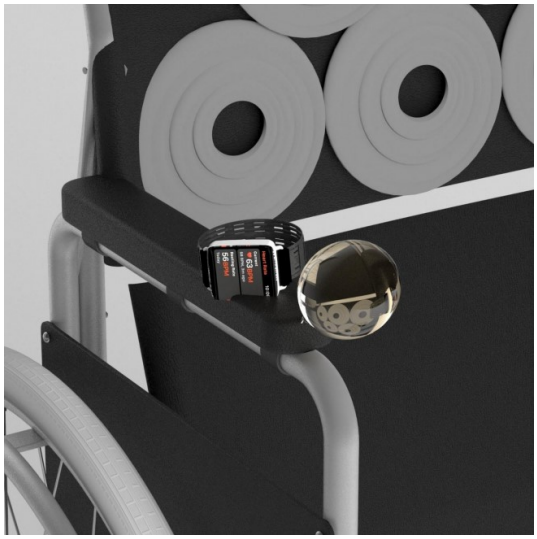

Figure 4.5: Built-in Wristwatch

4.1.6 Headrest

A headrest sensor is utilized to maintain the correct head position, preventing any potential discomfort or risk of choking for the patient.

Figure 4.6: Headrest

4.2 Processing Units and Used Devices

4.2.1 Arduino MEGA

A microcontroller board based on the ATmega2560 is the Arduino Mega 2560. It contains 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, 54 digital input/output pins (of which 15 can be utilized as PWM outputs), a USB connector, a power jack, an ICSP header, and a reset button. Everything required to sustain the microcontroller is contained in it.

We used an Arduino MEGA responsible for driving the keyboard, DC Motors, an air pump with a relay, and an Afficheur TFT 1.7.

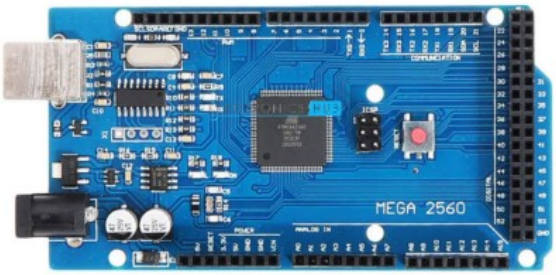

Figure 4.7: Arduino MEGA

4.2.2 Arduino Uno

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button.

we use Arduino Uno with a Pulse rate sensor, temperature sensor, and SIM900 to make a call when there is an emergency due to heart rate or temperature.

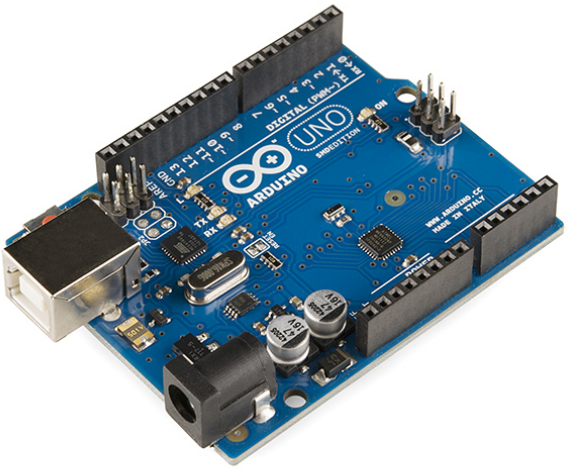

Figure 4.8: Arduino Uno

4.2.3 ESP32WROOM

ESP32-WROOM-32 is a powerful, generic Wi-Fi + Bluetooth® + Bluetooth LE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding. we use this controller for IoT to send the information of sensors to mobile applications with notifications and emails.

Figure 4.9: ESP32WROOM

4.2.4 Power Supply

A 24V 6A power supply was utilized in this project in order to provide high current to the stepper motor which we will explain later.

Figure 4.10: Power Supply

4.2.5 Afficheur TFT 1.7

A TFT 1.7-inch display is a type of small Thin Film Transistor (TFT) screen commonly used in various electronics projects for visual output. We use this display to show the main list for the users to get the instructions.

Figure 4.11: Afficheur TFT 1.7

4.2.6 0.96 OLED Display

0.96" (24 mm) monochrome graphical OLED display with a resolution of 128 x 32 pixels mounted on a 28 x 28 mm PCB. The effective display surface is 11 mm x 23 mm. It has a 4-pin connector to be used with an I^2C bus (with SCL and SDA signals). The display works with 5 V and 3.3 V applications.

We use this OLED to display the value of the temperature and heart rate, so we connect it with Arduino Uno which reads the value for temperature and heart rate.

Figure 4.12: 0.96 OLED Display

4.2.7 SIM900A

SIM900A Modem is built with Dual Band GSM/GPRS based SIM900A modem from SIMCOM. It works on frequencies 900/ 1800 MHz. SIM900A can search these two bands automatically. The frequency bands can also be set by AT Commands. The baud rate is configurable from 1200-115200 through AT command. The GSM/GPRS Modem is having internal TCP/IP stack to enable you to connect with internet via GPRS.

We connect it with Arduino UNO to make an emergency call in the event of an abnormality in the values of both temperature and heartbeat.

Figure 4.13: SIM900A

4.2.8 Micro air Pump

The micro air pump designed for an upper arm blood pressure monitor operates optimally within a voltage range of 3.7 to 12 volts. Its power consumption remains under 1 ampere and varies in direct relation to the airflow generated. With a leak rate of less than 3mmHg per minute, it ensures accurate and consistent pressure readings. This pump delivers an airflow ranging between 2.0 to 4.0 liters per minute, facilitating efficient functionality. Capable of reaching pressures exceeding 400mmHg.

We use an air pump with an Arduino Uno to pump air into the balloon, which helps move the patient's hand to reduce the contraction in the hand area and stimulate blood circulation in the body. It pumps air for a while and then stops until it helps open and reclose the patient's hand.

Figure 4.14: Micro air Pump

4.2.9 DC Motors

12V DC geared motors are compact motors that operate on 12 volts of direct current (DC). The gearing enhances torque, making these motors suitable for applications like robotics, automation, and small machinery.

We have used this motor in many parts responsible for massage for physical therapy in different areas, such as the back and feet. Each one operates in different processes and delays to achieve its goal.

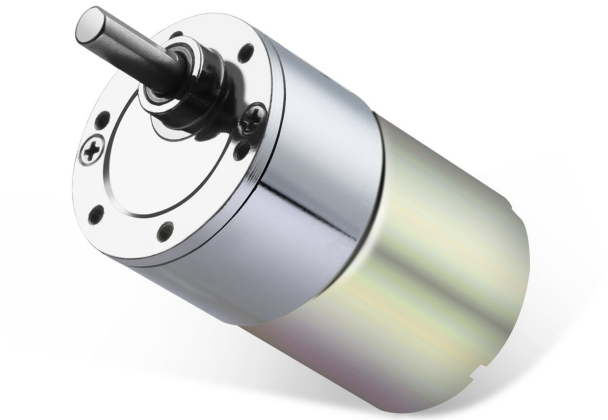

Figure 4.15: DC Motors

4.2.10 H-Bridge

An H-bridge is built of four switches that control the flow of current to a load. In the image above, the load is the M connecting the two sets of switches. Using one current source, you can drive current in two directions by closing two switches.

Our use of H-bridges with DC motors and Air Pump provides a versatile and efficient way to control various aspects of motor operation.

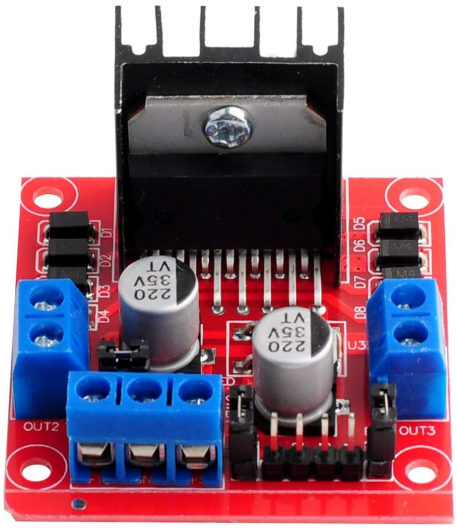

Figure 4.16: H-Bridge

4.2.11 Pulse Sensor - SEN11574

The SEN11574 Pulse Sensor is a device used to measure heart rate or pulse by detecting changes in blood volume in a fingertip. It measures heart rate or pulse by detecting changes in light absorption or reflection caused by blood flow through the fingertip.
So, we use it with Arduino Uno to get the value of the pulse rate, So that if the data is not normal SIM900A will make a call to the patient's parents.

Figure 4.17: Pulse Sensor

4.2.12 Tempreature Sensor DHT11

The DHT11 is a low-cost sensor used for measuring temperature (0 to 50°C) and humidity (20 - 80 RH). It provides digital output, making it easy to use with microcontrollers like Arduino. The module is encased in protective material with a few external pins for connectivity, often including power (VCC), ground (GND), and a single data pin for communication with a microcontroller. Also, we use it also the same as SEN11574 with Arduino Uno in emergency cases.

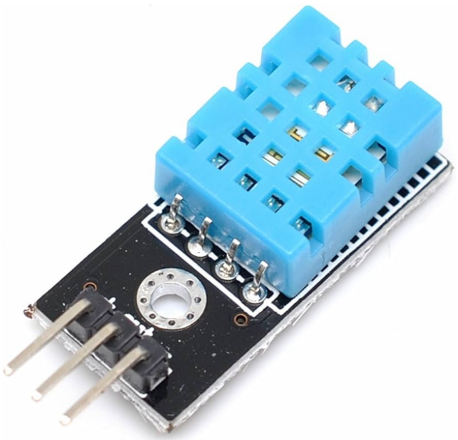

Figure 4.18: Tempreature Sensor DHT11

4.2.13 Keypad4*4

The 4×4 matrix keypad is an input device, it is usually used to provide input value in a project. It has 16 keys in total, which means it can provide 16 input values. The most interesting thing is it used only 8 GPIO pins of a microcontroller. We use it as an input device to select which part of the chair will work, each button is for a specific operation that can turn it on and off.

Figure 4.19: Keypad4*4

4.3 Operation Process

The system starts working if the patient sits in his designated place, then the system will check the place of the head and the hand-held balloon, then the system is activated to receive commands from the keypad.

4.3.1 Headrest process

when the system turns on at first it will check if the patient's head is in the right place, if not it will send an Alert by buzzer to inform the parents will check it. we check the Headrest using buttons in the place of head the that will be pressed when the head is in the right place.

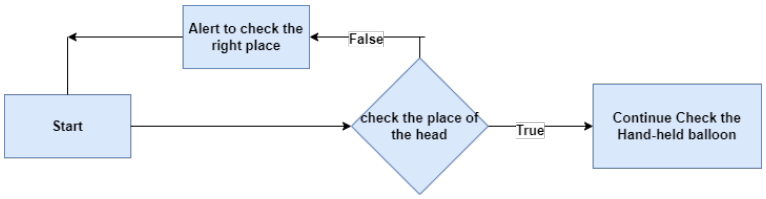

Figure 4.20: Headrest process

4.3.2 Hand-held Balloon Alerts

Here the system will check if the balloon is held correctly in the patient hand, and if not it will send alert for parents using buzzer.

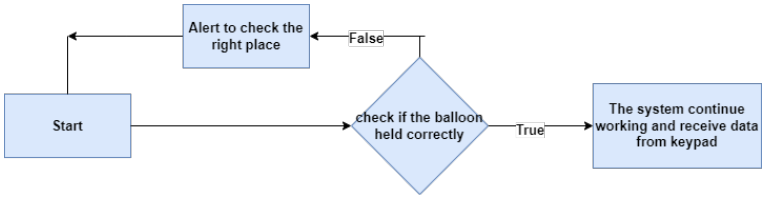

Figure 4.21: Hand-held Balloon process

Then each button in the keypad is specialized to start or stop a specific part, in addition to the possibility of starting many parts and stopping each of them individually or stopping them all together.

So if the user presses number 1 in the keypad the Back Massages Rollers will work and if presses it again it will turn off, number 2 will work for Foot Massages Rollers, 3 for Hand-held Balloon, and if we have different things work together we can turn off for all of them by pressing number 4. Here is the general diagram:

Figure 4.22: Basic instructions

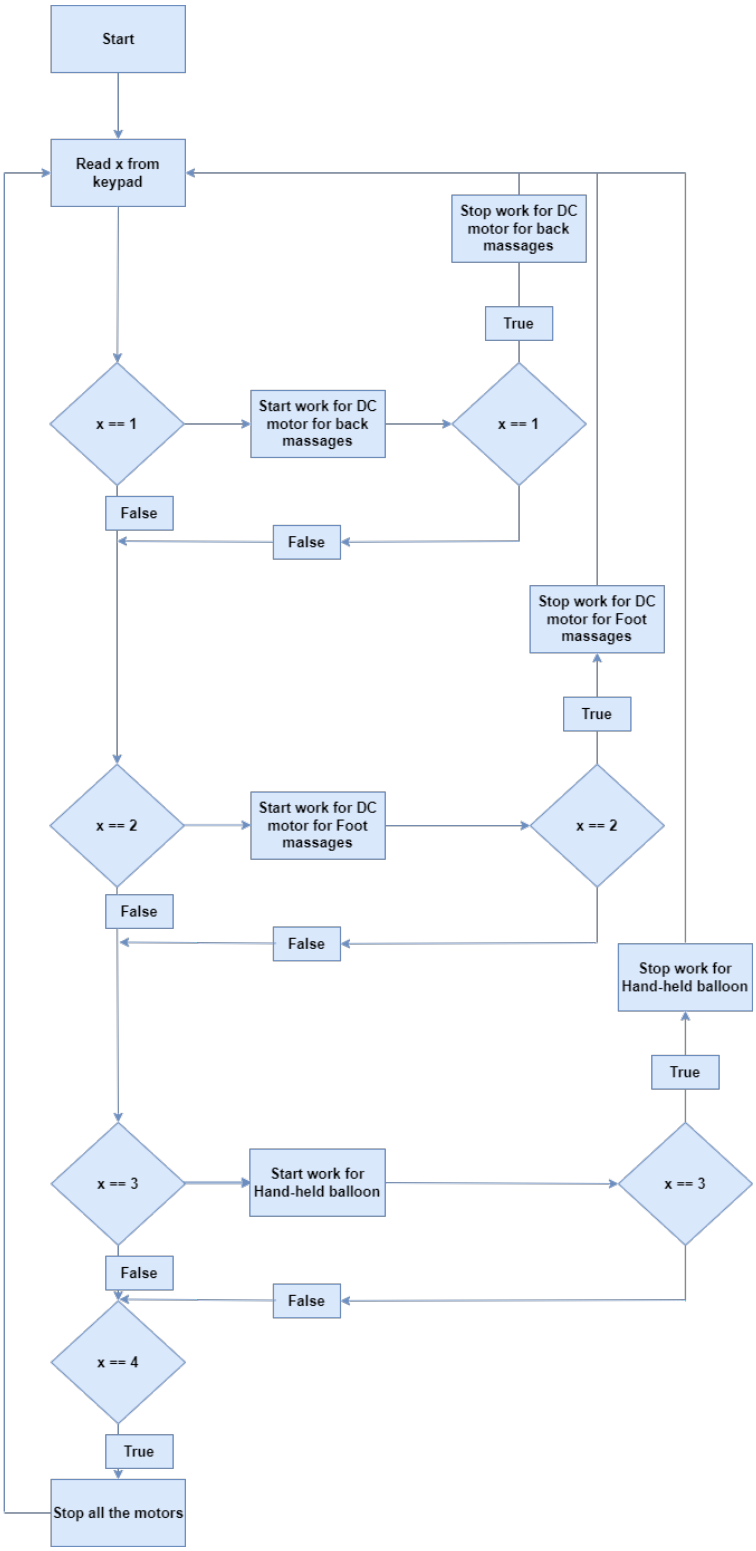

Figure 4.23: Full diagram

4.3.3 Back Massage process

In this process when we press on number 1 in the keypad that is connected to Arduino Mega the DC motors will start working, it will start rotating in one direction then after a specific time it will change its direction.

4.3.4 Foot Massage process

Also, it will start running when we press on number 2, It starts rotating clockwise and after a specified time, it reverses its direction. In this process, Arduino Mega will work with a DC motor.

4.3.5 Hand-held balloon process

when we press on number 3 it will begin by pumping air into the balloon to a certain extent to help open the patient's hand and move it from the continuous contraction. Then the air will be emptied from it and this process will be repeated until it stops.
In this process, the keypad will send the information to Arduino Mega to start working with Micro air Pump.

4.3.6 Wristwatch process

When the patient wears the watch it will start working, this watch is responsible for monitoring health status, it contains a Heart rate sensor and temperature sensor to check both pulse rate and patient temperature, so if there are any irregular reads it will make a call to patient parent for emergence, Also it will give regular reports to the patient's family.
So in this component, we use a Heart rate Sensor and temperature sensor connected with Arduino Uno connected with SIM900 for emergency calls, and we use ESP32WROOM for regular reports using Blynk.

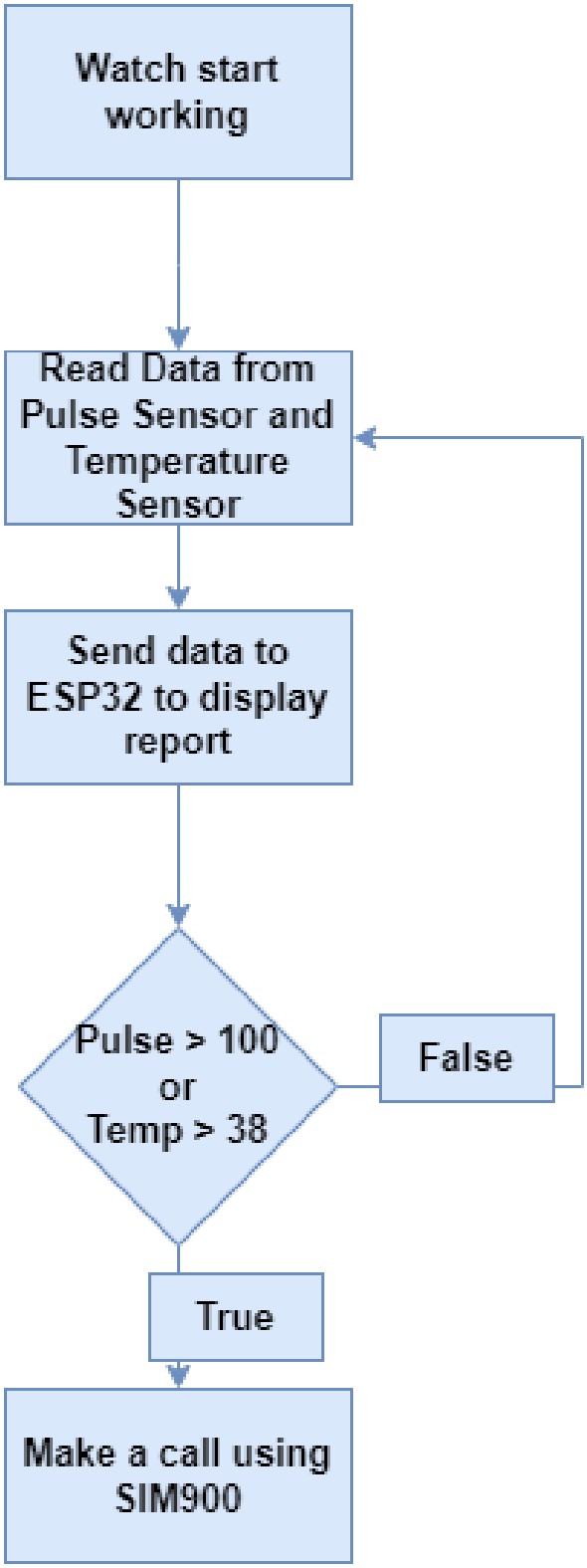

Figure 4.24: Wristwatch process

Chapter 5

Results and Discussion

5.1 Results

This specialized mobility chair is a holistic solution crafted for individuals coping with cerebral palsy and motor paralysis. It serves as a bridge between daily routines and crucial physical therapy, ensuring both comfort and essential support. Its design aims to elevate the overall quality of life for patients and their caregivers by seamlessly integrating therapy into everyday activities.

5.2 Discussion

The system introduces novel and advanced features, departing from conventional approaches. While it's currently in the developmental phase, it presents a solution that encompasses the potential to assist, it can aid a broader spectrum of patients down the line.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In summary, we envision this concept as a catalyst for a substantial transformation in the healthcare realm, potentially evolving into a coveted product. Its simplicity and intuitive design will render it inclusive for diverse individuals. This core objective has been the driving force behind our project since its inception.

6.2 Future Work

1— The chair converts to many positions:

a. wheelchair.

b. position bed.

c. standing position (assisting in the physiotherapy process).

2— Control the system via phone.

3— Using more advanced technological systems.

Chapter 7

Codes

7.1 Main Code for Arduino Mega

The first part of the codes represents the specialized part of the control panel responsible for controlling the massage process for various areas of the patient's body, in addition to the special balloon for hand massage. In this part, it is checked whether the head is in the correct place and ensures that the patient's hand is in the correct position to ensure the patient's correct position. Then, the various parts of the massage are allowed to be controlled through the control panel. Each part is operated with specific instructions and for a specific time so that it provides appropriate care for the patient.

```
#include <Keypad.h>
#include <SoftwareSerial.h>
#include <Adafruit_GFX.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ST7735.h>
#include <Wire.h>
#include <SPI.h>

#define TFT_CS    21
#define TFT_RST   19
#define TFT_DC    20
#define TFT_SCLK 24
#define TFT_MOSI 22
const int buttonPin1 = 28;//startbutton
const int buttonPin2 = 29;//head1
const int buttonPin3 = 30;//head2
const int buttonPin4 = 31;//handbutton
const int ledPin = 23;//bazer
bool button1Pressed = false;

```

```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
    TFT_RST);

SoftwareSerial bluetooth(0, 1);  // RX, TX for Bluetooth module

namespace CustomEnums {
  enum ProcessState { IDLE, ACTIVE };
  enum FlareState { INACTIVE, FLARE_ACTIVE = 2 };
}

const byte ROWS = 4;
const byte COLS = 4;
// char bluetoothCommand;
char hexaKeys[ROWS][COLS] = {
  {'D', 'C', 'B', 'A'},
  {'#', '9', '6', '3'},
  {'0', '8', '5', '2'},
  {'*', '7', '4', '1'}
};

byte rowPins[ROWS] = {10, 9, 8, 7};
byte colPins[COLS] = {6, 5, 4, 3};

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS)
    ;

const int relayPin = 2;
const int bottomofthefootPosPin = 26;
const int bottomofthefootNegPin = 27;
const int in1Pin = 14;
const int in2Pin = 15;
const int motorPosPin = 11;
const int motorNegPin = 12;

bool backRollersActive = false;
bool bottomofthefootRollersActive = false;
bool footRollersActive = false;
bool balloonActive = false;
bool startButtonPressed = false;

unsigned long backRollersStartTime = 0;
unsigned long bottomofthefootStartTime = 0;
unsigned long footRollersStartTime = 0;
unsigned long balloonStartTime = 0;

const unsigned long backRollersDuration = 120000;  // 2 minutes
const unsigned long balloonOnDuration = 30000;    // 1 minute
```

```
66
67 CustomEnums::ProcessState backRollersState = CustomEnums::IDLE;
68 CustomEnums::ProcessState bottomofthefootRollersState = CustomEnums::IDLE;
69 CustomEnums::ProcessState footRollersState = CustomEnums::IDLE;
70 CustomEnums::ProcessState balloonRollersState = CustomEnums::IDLE;
71 CustomEnums::FlareState backRollersFlareState = CustomEnums::INACTIVE;
72 CustomEnums::FlareState bottomofthefootRollersFlareState = CustomEnums::INACTIVE
    ;
73
74
75 void setup() {
76   Serial.begin(9600);
77   bluetooth.begin(9600);
78   pinMode(relayPin, OUTPUT);
79   pinMode(motorPosPin, OUTPUT);
80   pinMode(motorNegPin, OUTPUT);
81   pinMode(bottomofthefootPosPin, OUTPUT);
82   pinMode(bottomofthefootNegPin, OUTPUT);
83   pinMode(in1Pin, OUTPUT);
84   pinMode(in2Pin, OUTPUT);
85   pinMode(buttonPin1, INPUT_PULLUP);
86   pinMode(buttonPin2, INPUT_PULLUP);
87   pinMode(buttonPin3, INPUT_PULLUP);
88   pinMode(buttonPin4, INPUT_PULLUP);
89   pinMode(ledPin, OUTPUT);
90   tft.initR(INITR_BLACKTAB);
91   tft.fillScreen(ST7735_BLACK);
92
93   tft.drawRoundRect(3, 5, 122, 150,5, ST7735_BLUE);
94   tft.setCursor(30, 20);
95   tft.setTextColor(ST7735_GREEN);
96   tft.setTextSize(1);
97       tft.setCursor(10, 20);
98       tft.println("1-Back massage");
99       tft.setCursor(10, 40);
100      tft.println("2-Move the toucher");
101      tft.setCursor(10, 60);
102      tft.println("3-Hand balloon");
103      tft.setCursor(10, 80);
104      tft.println("4-Foot massage");
105      tft.setCursor(10, 100);
106      tft.println("5-Stop All");
107
108  tft.drawRect(8 , 125, 112, 25,ST7735_BLUE);
109  // tft.fillRect(11, 128, 8, 19, ST7735_RED);
110
111 }
```

```
112
113 void loop() {
114     processBluetoothCommand();
115
116
117 if (digitalRead(buttonPin1) == LOW) {
118         startButtonPressed = true;
119     } else {
120         startButtonPressed = false;
121     }
122
123     if (startButtonPressed && digitalRead(buttonPin2) == HIGH && digitalRead(
    buttonPin3) == HIGH && digitalRead(buttonPin4) == HIGH) {
124         digitalWrite(ledPin, HIGH);
125     } else if (startButtonPressed && digitalRead(buttonPin2) == LOW  ||
    digitalRead(buttonPin3) == LOW && digitalRead(buttonPin4) == HIGH) {
126         digitalWrite(ledPin, HIGH);
127     }
128    else if (startButtonPressed && digitalRead(buttonPin2) == HIGH  &&
    digitalRead(buttonPin3) == HIGH && digitalRead(buttonPin4) == LOW) {
129         digitalWrite(ledPin, HIGH);
130     } else if (startButtonPressed && digitalRead(buttonPin2) == LOW  &&
    digitalRead(buttonPin3) == LOW && digitalRead(buttonPin4) == LOW) {
131         digitalWrite(ledPin, LOW);
132     }
133     else {
134         digitalWrite(ledPin, LOW);
135     }
136
137
138
139
140
141   // Continue with keypad-based control
142   char customKey = customKeypad.getKey();
143
144   // Directly integrate functionality here
145   if (customKey) {
146     switch (customKey) {
147       case '1':
148         toggleBackRollers();
149         break;
150
151       case '2':
152         toggleFootRollers();
153         break;
154
```

```
155       case '3':
156         toggleBalloon();
157         break;
158 
159       case '5':
160         stopAll();
161         break;
162 
163       case '4':
164         togglebottomofthefootRollers();
165         break;
166 
167       case '6':
168         backRollersFlareState = CustomEnums::FLARE_ACTIVE;
169         bottomofthefootRollersFlareState = CustomEnums::FLARE_ACTIVE;
170         break;
171       // Add more cases as needed for additional commands
172 
173       default:
174         break;
175     }
176   }
177 
178   controlDevices();
179 }
180 
181 void processBluetoothCommand() {
182   if (bluetooth.available() > 0) {
183     char bluetoothCommand = bluetooth.read();  // Remove '\n' argument
184     Serial.print("Received command from Bluetooth: ");
185     Serial.println(bluetoothCommand);
186 
187     if (bluetoothCommand == '1') {
188       Serial.println("Command 1 received. Calling toggleBackRollers...");
189       toggleBackRollers();
190     } else if (bluetoothCommand == '2') {
191       toggleFootRollers();
192     } else if (bluetoothCommand == '3') {
193       toggleBalloon();
194     } else if (bluetoothCommand == '5') {
195       stopAll();
196     } else if (bluetoothCommand == '4') {
197       togglebottomofthefootRollers();
198     } else if (bluetoothCommand == '6') {
199       backRollersFlareState = CustomEnums::FLARE_ACTIVE;
200       bottomofthefootRollersFlareState = CustomEnums::FLARE_ACTIVE;
201     }
```

```
202   }
203 }
204
205
206 void toggleBackRollers() {
207   backRollersActive = !backRollersActive;
208   if (backRollersActive) {
209     backRollersStartTime = millis();
210     backRollersState = CustomEnums::ACTIVE;
211   } else {
212     backRollersState = CustomEnums::IDLE;
213     digitalWrite(motorPosPin, LOW);
214     digitalWrite(motorNegPin, LOW);
215   }
216 }
217
218 void togglebottomofthefootRollers() {
219   bottomofthefootRollersActive = !bottomofthefootRollersActive;
220   if (bottomofthefootRollersActive) {
221     bottomofthefootStartTime = millis();
222     bottomofthefootRollersState = CustomEnums::ACTIVE;
223   } else {
224     bottomofthefootRollersState = CustomEnums::IDLE;
225     digitalWrite(bottomofthefootPosPin, LOW);
226     digitalWrite(bottomofthefootNegPin, LOW);
227   }
228 }
229
230 void toggleFootRollers() {
231   footRollersActive = !footRollersActive;
232   if (footRollersActive) {
233     footRollersStartTime = millis();
234     footRollersState = CustomEnums::ACTIVE;
235   } else {
236     footRollersState = CustomEnums::IDLE;
237     digitalWrite(in1Pin, LOW);
238     digitalWrite(in2Pin, LOW);
239   }
240 }
241
242 void toggleBalloon() {
243   balloonActive = !balloonActive;
244    if (balloonActive) {
245   balloonStartTime = millis();
246     balloonRollersState = CustomEnums::ACTIVE;
247   } else {
248     balloonRollersState = CustomEnums::IDLE;
```

```
249     digitalWrite(relayPin, LOW);
250   }
251 }
252 
253 void stopAll() {
254   bottomofthefootRollersActive= false;
255   backRollersActive = false;
256   footRollersActive = false;
257   balloonActive = false;
258   digitalWrite(bottomofthefootPosPin, LOW);
259   digitalWrite(bottomofthefootNegPin, LOW);
260   digitalWrite(motorPosPin, LOW);
261   digitalWrite(motorNegPin, LOW);
262   digitalWrite(in1Pin, LOW);
263   digitalWrite(in2Pin, LOW);
264   digitalWrite(relayPin, LOW);
265   bottomofthefootRollersState= CustomEnums::IDLE;
266   backRollersState = CustomEnums::IDLE;
267   footRollersState = CustomEnums::IDLE;
268   balloonRollersState = CustomEnums::IDLE;
269   backRollersFlareState = CustomEnums::INACTIVE;
270   bottomofthefootRollersFlareState= CustomEnums::INACTIVE;
271 
272 }
273 
274 void controlDevices() {
275   controlBackRollers();
276   controlFootRollers();
277   controlBalloon();
278   controlbottomofthefootRollers();
279   controlBackRollersFlare();
280   controlbottomofthefootRollersFlare();
281 
282 }
283 
284 void controlBackRollers() {
285   if (backRollersState == CustomEnums::ACTIVE) {
286     unsigned long elapsedTime = millis() - backRollersStartTime;
287 
288     if (elapsedTime < backRollersDuration) {
289       digitalWrite(motorPosPin, HIGH);
290       digitalWrite(motorNegPin, LOW);
291     } else if (elapsedTime < (backRollersDuration + 60000)) {
292       digitalWrite(motorPosPin, LOW);
293       digitalWrite(motorNegPin, LOW);
294     } else {
295       backRollersStartTime = millis();
```

```
296       backRollersState = CustomEnums::ACTIVE;
297       backRollersActive = true;
298     }
299   } else {
300     backRollersStartTime = millis();
301     backRollersState = CustomEnums::IDLE;
302     backRollersActive = false;
303   }
304 }
305 
306 void controlbottomofthefootRollers() {
307   if (bottomofthefootRollersState == CustomEnums::ACTIVE) {
308     unsigned long elapsedTime = millis() - bottomofthefootStartTime;
309 
310     if (elapsedTime < backRollersDuration) {
311       digitalWrite(bottomofthefootPosPin, HIGH);
312       digitalWrite(bottomofthefootNegPin, LOW);
313     } else if (elapsedTime < (backRollersDuration + 60000)) {
314       digitalWrite(bottomofthefootPosPin, LOW);
315       digitalWrite(bottomofthefootNegPin, LOW);
316     } else {
317       bottomofthefootStartTime = millis();
318       bottomofthefootRollersState = CustomEnums::ACTIVE;
319       bottomofthefootRollersActive = true;
320     }
321   } else {
322     bottomofthefootStartTime = millis();
323     bottomofthefootRollersState = CustomEnums::IDLE;
324     bottomofthefootRollersActive = false;
325   }
326 }
327 
328 void controlFootRollers() {
329   if (footRollersState == CustomEnums::ACTIVE) {
330     static unsigned long prevFootRollersTime = 0;
331     unsigned long currentMillis = millis();
332     if (currentMillis - prevFootRollersTime >= 400) {
333       prevFootRollersTime = currentMillis;
334       static bool state = false;
335       digitalWrite(in1Pin, state);
336       digitalWrite(in2Pin, !state);
337       state = state;
338       digitalWrite(in1Pin, state);
339       digitalWrite(in2Pin, !state);
340       state = !state;
341 
342     }
```

```
343   } else {
344     digitalWrite(in1Pin, LOW);
345     digitalWrite(in2Pin, LOW);
346   }
347 }
348 
349 void controlBalloon() {
350   if (balloonRollersState == CustomEnums::ACTIVE) {
351     unsigned long elapsedTime = millis() - balloonStartTime;
352 
353     if (elapsedTime < balloonOnDuration) {
354       digitalWrite(relayPin, HIGH);
355     } else if (elapsedTime < (2 * balloonOnDuration)) {
356       digitalWrite(relayPin, LOW);
357     }else {
358       balloonStartTime = millis();
359       balloonRollersState = CustomEnums::ACTIVE;
360       balloonActive = true;
361     }
362     }else {
363       balloonRollersState = CustomEnums::IDLE;
364       balloonActive = false;
365   }
366 }
367 
368 
369 void controlBackRollersFlare() {
370   if (backRollersFlareState == CustomEnums::FLARE_ACTIVE) {
371     static unsigned long prevBackRollersFlareTime = 0;
372     unsigned long currentMillis = millis();
373     if (currentMillis - prevBackRollersFlareTime >= 120000) {  // Two minutes
374       prevBackRollersFlareTime = currentMillis;
375       backRollersState = CustomEnums::ACTIVE;
376       backRollersActive = true;
377       backRollersFlareState = CustomEnums::INACTIVE;
378     }
379   }
380 }
381 
382 void controlbottomofthefootRollersFlare() {
383   if (bottomofthefootRollersFlareState == CustomEnums::FLARE_ACTIVE) {
384     static unsigned long prevBackRollersFlareTime = 0;
385     unsigned long currentMillis = millis();
386     if (currentMillis - prevBackRollersFlareTime >= 120000) {  // Two minutes
387       prevBackRollersFlareTime = currentMillis;
388       bottomofthefootRollersState = CustomEnums::ACTIVE;
389       bottomofthefootRollersActive = true;
```

```
390         bottomofthefootRollersFlareState = CustomEnums::INACTIVE;
391       }
392     }
393 }
```

7.2 Main Code for Arduino Uno

Here is the code for Arduino Uno that will work in a hand watch to measure Pulse rate and temperature and make an emergency call. It also will display Values in 0.96OLED.

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <SoftwareSerial.h>

SoftwareSerial SIM900A(10,11);
#define DHTPIN A1
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

Adafruit_SSD1306 srituhobby = Adafruit_SSD1306(128, 64, &Wire);

#define sensor A0
#define Highpulse 540

int sX = 0;
int sY = 60;
int x = 0;
int Svalue;
int value;
long Stime = 0;
long Ltime = 0;
int count = 0;
int Bpm = 0;

void setup() {
  Serial.begin(9600);
  SIM900A.begin(9600);   // Setting the baud rate of GSM Module
  Serial.println ("SIM900A Ready");
  srituhobby.begin(SSD1306_SWITCHCAPVCC, 0x3C); // Address 0x3C for 128x32
  dht.begin();
  delay(1000);
  srituhobby.clearDisplay();
}

void loop() {
  Svalue = analogRead(sensor);
  Serial.println(Svalue);
  value = map(Svalue, 0, 1024, 0, 45);

```

```
  int y = 60 - value;

  if (x > 128) {
    x = 0;
    sX = 0;
    srituhobby.clearDisplay();
  }

  srituhobby.drawLine(sX, sY, x, y, WHITE);
  sX = x;
  sY = y;
  x++;

  float temperature = dht.readTemperature(); // Read temperature from DHT11
    sensor

  BPM();

  if(temperature >= 15){
    ImegancyCall();
  }

  srituhobby.setCursor(0, 0);
  srituhobby.setTextSize(1);
  srituhobby.setTextColor(SSD1306_WHITE);
  srituhobby.print("BPM :");
  srituhobby.setCursor(0, 16);
  srituhobby.setTextSize(1);
  srituhobby.setTextColor(SSD1306_WHITE);
  srituhobby.print("Temp: ");
  srituhobby.print(temperature);
  srituhobby.print(" C");
  srituhobby.display();

   if (SIM900A.available()>0)
   Serial.write(SIM900A.read());

}

void BPM() {

  if (Svalue > Highpulse) {
    Stime = millis() - Ltime;
    count++;

    if (Stime / 1000 >= 60) {
      Ltime = millis();
```

```
90        Serial.println(count);
91        if(count >= 100){
92        ImegancyCall();
93    }
94        srituhobby.setCursor(60, 0);
95        srituhobby.setTextSize(1);
96        srituhobby.setTextColor(SSD1306_WHITE);
97        srituhobby.print(count);
98        srituhobby.print("   ");
99        srituhobby.display();
100       count = 0;
101     }
102   }
103 }
104 void ImegancyCall()
105 {
106   Serial.println ("make Call");
107   SIM900A.println("AT+CMGF=1");
108   delay(1000);
109   Serial.println ("Set call Number");
110   SIM900A.println("ATD0599997156;\r\n"); //Mobile phone number to call
111   delay(1000);
112   Serial.println ("Call has been Done");
113 }
```

7.3 Main Code for ESP

This code will display the values for pulse rate and temperature in the Mobile Phone using Blynk.

```
    #define BLYNK_TEMPLATE_ID "TMPL6JlkL16e1"
#define BLYNK_TEMPLATE_NAME "The Devoted Caregiver"
#define BLYNK_AUTH_TOKEN "R_zACzOw1_T7gLoeRkDN1MCDRnL2VB8s"

#define BLYNK_PRINT Serial
#include <WiFi.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>

// Pulse Sensor Configuration
#define PULSE_SENSOR_PIN 32  // Connect Pulse Sensor to GPIO 32
int pulseSensorValue;        // Variable to store pulse sensor data
int lastPulseSensorValue = 0; // Variable to store the last pulse sensor data
unsigned long lastPulseSensorTime = 0; // Variable to store the last time pulse
    sensor data was sent to Blynk
#define PULSE_SENSOR_UPDATE_INTERVAL 2000  // Update pulse sensor data every 2
    seconds

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Jamousm";  // type your wifi name
char pass[] = "406699991";  // type your wifi password

BlynkTimer timer;

#define DHTPIN 26 // Connect Out pin to D2 in NODE MCU
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void sendSensor()
{
  // Read DHT sensor data
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Send DHT sensor data to Blynk
  Blynk.virtualWrite(V0, t);
  Blynk.virtualWrite(V1, h);
  Serial.print("Temperature: ");
```

```
  Serial.print(t);
  Serial.print("    Humidity: ");
  Serial.println(h);

  // Read pulse sensor data
  pulseSensorValue = analogRead(PULSE_SENSOR_PIN);

  // Update pulse sensor data every PULSE_SENSOR_UPDATE_INTERVAL milliseconds
  if (millis() - lastPulseSensorTime > PULSE_SENSOR_UPDATE_INTERVAL) {
    // Send pulse sensor data to Blynk
    Blynk.virtualWrite(V2, pulseSensorValue);
    Serial.print("Heart Rate: ");
    Serial.println(pulseSensorValue);

    lastPulseSensorTime = millis();
  }
}

void setup()
{
  Serial.begin(115200);

  Blynk.begin(auth, ssid, pass);
  dht.begin();
  timer.setInterval(100L, sendSensor);
}

void loop()
{
  Blynk.run();
  timer.run();
}
```

Bibliography

[1] Karen W Krigger. “Cerebral palsy: an overview”. In: *American family physician* 73.1 (2006), pp. 91–100.

[2] Kevin K Tremper. “Pulse oximetry”. In: *Chest* 95.4 (1989), pp. 713–715.

[3] Tiffany Field. “Massage therapy”. In: *Medical Clinics* 86.1 (2002), pp. 163–171.

[4] Arduino. *Arduino - Open-source electronics platform*. Accessed: Current date. Year the website was last updated or accessed. URL: `https://www.arduino.cc/`.

[1] [2] [3] [4]