Health & Covid Cap

A Project Report

On

ESS Lab Mini Project

Table of Contents

[Problem Statement 3](#_Toc104756938)

[Objective 3](#_Toc104756939)

[Proposed Solution 4](#_Toc104756940)

[Features 4](#_Toc104756941)

[UN Sustainable Development Goals (SDGs) satisfied 4](#_Toc104756942)

[Components 5](#_Toc104756943)

[Component List 5](#_Toc104756944)

[Raspberry Pi Pico 5](#_Toc104756945)

[MPU6050 6](#_Toc104756946)

[MLX90614 6](#_Toc104756947)

[MAX30102 6](#_Toc104756948)

[HC-SR04 7](#_Toc104756949)

[LM35 7](#_Toc104756950)

[HC-05 7](#_Toc104756951)

[1KΩ and 2.2KΩ Resistors 7](#_Toc104756952)

[Buzzer and 220Ω Resistor 8](#_Toc104756953)

[3.7V, 2000mAh Lithium Polymer (LiPo) Battery 8](#_Toc104756954)

[TP4056 LiPo Charging Module 8](#_Toc104756955)

[5V Boost Converter 8](#_Toc104756956)

[Slide Switch 8](#_Toc104756957)

[Hardware 9](#_Toc104756958)

[Block Diagram 9](#_Toc104756959)

[Circuit Diagram & Explanation 10](#_Toc104756960)

[Program on the Raspberry Pi Pico 12](#_Toc104756961)

[Basic Flowchart 12](#_Toc104756962)

[Code (Arduino IDE) 13](#_Toc104756963)

[Mobile Application Code 21](#_Toc104756964)

[Output 22](#_Toc104756965)

[Pictures & Video of Circuit Implementation 24](#_Toc104756966)

[Conclusion 25](#_Toc104756967)

[Limitations 25](#_Toc104756968)

[Future Scope 26](#_Toc104756969)

[References & Bibliography 26](#_Toc104756970)

[Contribution of the Group Members 26](#_Toc104756971)

# Problem Statement

People face various health and health-monitoring related concerns:

* People want to monitor each and every health aspect since to Covid-19 Pandemic.
* Social distance is very important, but people forget about it easily.
* Existing Devices are very expensive and unaffordable.
* There isn’t any ‘single’ device with all the required features.
* Predicting a possibility of Covid patient is also very important.
* We want to design a Cap with all the necessary features.

# Objective

* To design a proper health monitoring system:

This system must be effective. Smaller systems made for Health Monitoring usually lack on their effectiveness and performance. Their accuracy is also quite questionable.

* To make it user friendly:

This device should be usable to all the age groups and must be very easy to use. Knowledge and practice required to successfully operate it must be minimal.

* To allow users to perform successful self-analysis with this device:

Users must be in a position to firmly analyse their health aspects with this device. They must be able to decide whether they should visit a doctor or not.

* To help people predict a possibility of being a covid positive:

Covid can sometimes cause no symptoms. Users should be able to analyse themselves and decide about a possibility of infection so they could quarantine themselves.

* To alarm people about proximity of a person with fever:

To stop the spread of the virus, people need to stay away from possible carriers. The most common symptom of this infection is fever. People need to be sure that the people they are in contact with have not caught fever. They need to stay away from the people who are symptomatic. The system must be able to measure someone else’s temperature in order to check if they have fever or not.

* To successfully maintain social distancing and avoid in-person interaction, the key to beat the virus.
* To incorporate all these features in a single handy and comprehensible device:
* All the features must be performed by a single device which is portable and easy to use.
* To make this device affordable:

The device must reach out to all and everyone must be able to purchase it. Only then, the impact will be big enough to decrease the spread of the virus.

# Proposed Solution

We propose a solution, a handy medical can that is capable of giving overall health report of the patient.

It is a single device that integrates many features and devices.

It is easy to use and inexpensive.

It helps the user in maintaining social distancing.

It will reduce the burden on the medical staff as people will be capable of checking their basic health aspects. They won’t go to the hospital for small problems.

It will also reduce cost by replacing bulky medical equipment.

## Features

* Measurement of Self Body Temperature at Forehead
* Measurement of Self Heart Rate
* Measurement and Maintenance of Social Distancing
* Measurement of Someone Else’s Body Temperature
* Analysis of Sleep Quality
* Controlled by Smartphone App
* Output on Smartphone App

# UN Sustainable Development Goals (SDGs) satisfied

Target 3.3

By 2030, end the epidemics of AIDS, tuberculosis, malaria and neglected tropical diseases and combat hepatitis, water-borne diseases and other communicable diseases



Target 9.5

Enhance scientific research, upgrade the technological capabilities of industrial sectors in all countries, in particular developing countries, including, by 2030, encouraging innovation and substantially increasing the number of research and development workers per 1 million people and public and private research and development spending



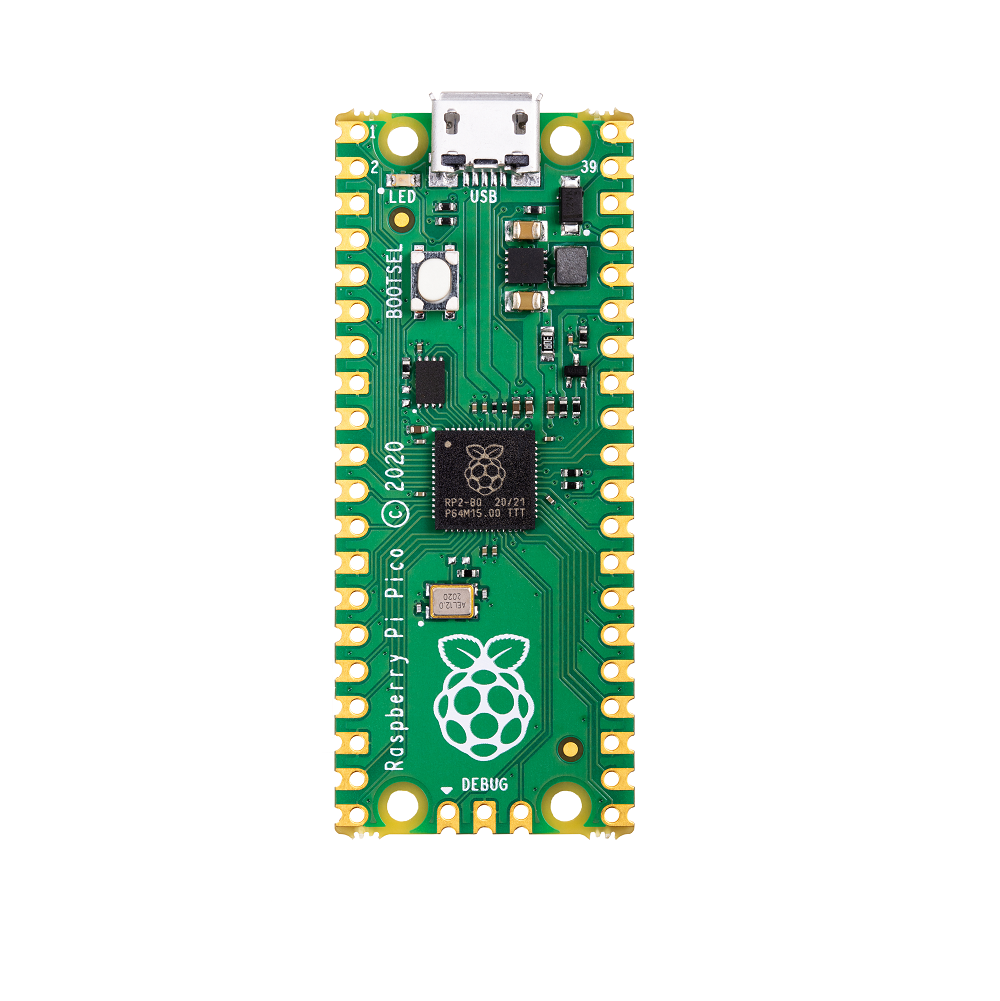
Target 11.3

By 2030, enhance inclusive and sustainable urbanization and capacity for participatory, integrated and sustainable human settlement planning and management in all countries

# Components

## Component List

* Raspberry Pi Pico
* MPU6050
* MLX90614
* MAX30102
* HC-SR04
* LM-35
* HC-05
* 1kΩ, 2.2 kΩ Resistor
* 220ΩmReistor
* Buzzer
* 3.7 V, 2000mAh LiPo Battery
* TP4056 LiPo Charging Module
* 5V Boost Converter
* Slide Switch



## Raspberry Pi Pico

Raspberry Pi Pico is Raspberry Pi’s 1st Microcontroller Board released in 2021. It is tiny, fast and cheap.

It is built around RP2040 Microcontroller Chip (IC-Integrated Circuit) designed by Raspberry Pi itself.

RP2040 features a dual-core Arm Cortex-M0+ processor with 264KB internal RAM and support for up to 16MB of off-chip Flash. A wide range of flexible I/O options includes I2C, SPI, and — uniquely — Programmable I/O (PIO). These support endless possible applications for this small and affordable package.

Specifications of Raspberry Pi Pico:

* RP2040 microcontroller chip designed by Raspberry Pi
* Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz
* 264KB on-chip SRAM & 2MB on-board QSPI Flash
* 26 multifunction GPIO pins, including 3 analog inputs
* 2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels
* 1 × USB 1.1 controller and PHY, with host and device support
* 8 × Programmable I/O (PIO) state machines for custom peripheral support
* Supported input power 1.8–5.5V DC
* Operating temperature -20°C to +85°C
* Castellated module allows soldering direct to carrier boards
* Drag-and-drop programming using mass storage over USB
* Low-power sleep and dormant modes
* Accurate on-chip clock
* On-Chip Temperature sensor (uses the 4th analog channel of the ADC)
* Accelerated integer and floating-point libraries on-chip

The Raspberry Pi Pico has Micro-USB for Serial Communications with a PC over UART Port0. This is used for programming the Pico and showing Output of the circuit on the PC.

It can be programmed using MicroPython (usually on Thonny IDE) and C using any IDE support these languages. It can also be programmed using Arduino IDE using its version of C/C++.

We have used Arduino IDE.

## MPU-6050 Gyro Sensor Module at Rs 250/piece | Girgaon | Mumbai| ID: 12778161062MPU6050

We needed a Gyroscope sensor. But tradition Gyroscope would have used 3 Analog Inputs, which we couldn’t afford as we need another one for LM35; Raspberry Pi Pico has only 3. We were already using I2C bus for other sensors, thus we chose to go with an I2C version of Gyroscope to save Pins.

The most readily available and cheapest gyroscope with required features was MPU-6050, which is not only a gyroscope but also an accelerometer, i.e., it is a 3-Axis IMU (Inertial Measurement Unit). It automatically does all the calculations and stores it in its memory, we just have to extract it using I2C communication. It does not take any additional pins because I2C bus allows up to 112 different Slave devices to be connected directly.

It operates at 3.3V Logic.

## MLX90614

MLX-90614 is very fascinating sensor. It uses I2C communication to communicate its readings to the Master Device.

It automatically calculates Ambient (its own) and Target Temperatures and stores it in its memory. It does this by using Stephan-Boltzmann law.

We just have to extract the readings from its memory which we do by I2C communication. It is available directly (just the component) and in module form (with pull up resistors and capacitor). The Module form is very expensive thus we chose to go with the direct component.

It operates at 3.3V Logic.

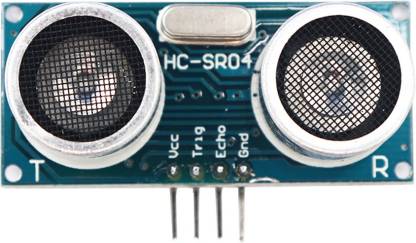
## MAX30102

This sensor measures Heart Rate and Blood Oxygen Levels. It can be attached to Finger and Earlobe. It was the only easily available and cheap sensor available for the purpose. It uses I2C communication thus takes no extra pins as we are already using the I2C bus for two other sensors.

It sends light and the decodes the reflected pattern to sense the parameters. For example, in measuring Heart Rate, this reflected pattern, after removal of noise and processing, has peaks and valleys, the period between two peaks, or valleys is the time for one heart beat (Beat Period), using this, Beat Frequency and Beats Per Minute (bpm) can be evaluated.

It operates at 3.3V Logic.

## HC-SR04

HC-SR04 Ultrasonic Distance Sensor is one of the most famous sensors among enthusiasts. Its operation is very simple and it works very fast. It is easily available and is very cheap.

It is easy to use and very easy to code. Its accuracy is fascinating. There some other versions of this sensors available but this is the most simple, easy to use and readily available one.

LIDAR sensors are a choice over this sensor owing to its high speed, high accuracy and compact form factor, but are very expensive.

It operates at 3.3V Logic.

## LM35

LM35 is a very precise and easy to use analog temperature sensor. Its accuracy is very high. Its output volage in Volts is just 1/100th of temperature in degree centigrade.

This means this is capable of measuring positive centigrade temperatures under normal supply voltages. But we can sense negative temperature as well by using specialized circuit with a negative voltage source. We chose this sensor over TMP36 because it is much more accurate in the required range and conversion of voltage to temperature is very simple.

It can be powered with 4-20V.

## HC-05 - Bluetooth Module - with Pins (&quot;hc05&quot;,&quot;hc-5&quot;)HC-05

HC-05 is a Bluetooth Module. It is an easy to use and cheap connectivity solution. It is easy to set up and easy to pair with other devices. It is very cheap compared to a Wi-Fi Module. Although Wi-Fi is better in terms of connectivity, on advantage Bluetooth allows is that one can connect a smart device to multiple Bluetooth device. So, this will not interrupt use of other devices like earphones while using the Health & Covid Cap.

It operates at 3.3V Logic and can pe powered with 3.6-6V

## Smart Electronics 1w Metal Film Color Ring Resistor 0.1r/1r/10 Ohm / 100 Ohm / 1k/10k / 100k / 1m 1bag=200pieces - Buy Resistors,Metal Film Resistor,100 Ohm Resistor Product on Alibaba.com1KΩ and 2.2KΩ Resistors

We are using devices operating at both 3.3V and 5V Logic. A 3.3V Logic O/P is compatible with 5V Logic I/P because of Noise Margin. But %V Logic O/P at a 3.3V Logic Input can easily damage the latter device.

The HC-05 Echo pin is at 5V, but goes to Raspberry Pi Pico at 3.3V. Thus, we need a voltage divider to shift the logic level.

1KΩ and 2.2KΩ Resistors form the Voltage Divider, mapping the 5V Logic Level to 3.33V Logic Level (which is acceptable to Raspberry Pi Pico).

## Piezo Buzzer 5V – FactoryForward IndiaBuzzer and 220Ω Resistor

Buzzer makes a good output device when acquiring user’s attention is required in case of an emergency.

To reduce the sound amplitude and current, we added a 220Ω Resistor in series to the buzzer.

## 3.7V, 2000mAh Lithium Polymer (LiPo) Battery

Have used Lithium Polymer Battery to power the project. LiPo Batteries are very compact and easy to use.

But they need to be used carefully because they cannot sustain mechanical or electrical (short circuit, over/undercharge) abuse.

They are available as single and multicell form. We have used Single Cell which is has a nominal voltage of 3.7V. It has a capacity of 2000mAh.

## TP4056 LiPo Charging Module

LiPo batteries need to be charged in a very specific way. We need to first charge at Constant Current (CC), then at Constant Voltage (CV). This is called CC-CV Method.

Also, we need some protection features like Short-Circuit, Over-Current, Over-Charge and Under-Charge Protection.

All these features are included in TP4056 LiPo Charging Module. We connect the load at the Output terminals of the Module and the LiPo Battery at Battery terminals. We connect the Charging Voltage to the USB port provided or input terminals.

These modules are available with many types of USB ports: Mini-USB, Micro-USB, Type-C. We are using Type-C variant.

## 5V Boost Converter

We need 5V for some the circuit. But LiPo battery has nominal voltage of 3.7V. So, we need to Boost it using a Boost Converter.

We are using a 5V Boost Converter.

It can take 0.9-5V at its input to generate stable, regulated 5V.

The Output is obtained at a USB-A port. But we don’t need it. So, we have de-soldered it and soldered a pair of wires at the output pins instead.

## IC-205 3-Pin Slide Switch ON-OFFSlide Switch

It switches the connection between TP4056 Module and Boost Converter, thus acts as ON-OFF Switch for the circuit.

# Hardware

This system has a lot of inputs and output.

It is a complex Internet-of-Things (IoT) System.

The main output with readings is displayed on the smartphone (communicated through Bluetooth).

The circuit also receives commands from the smartphone through Bluetooth.

Buzzer is also an output indicating an emergency situation.

## Block Diagram

All the sensors are inputs to the Raspberry Pi Pico.

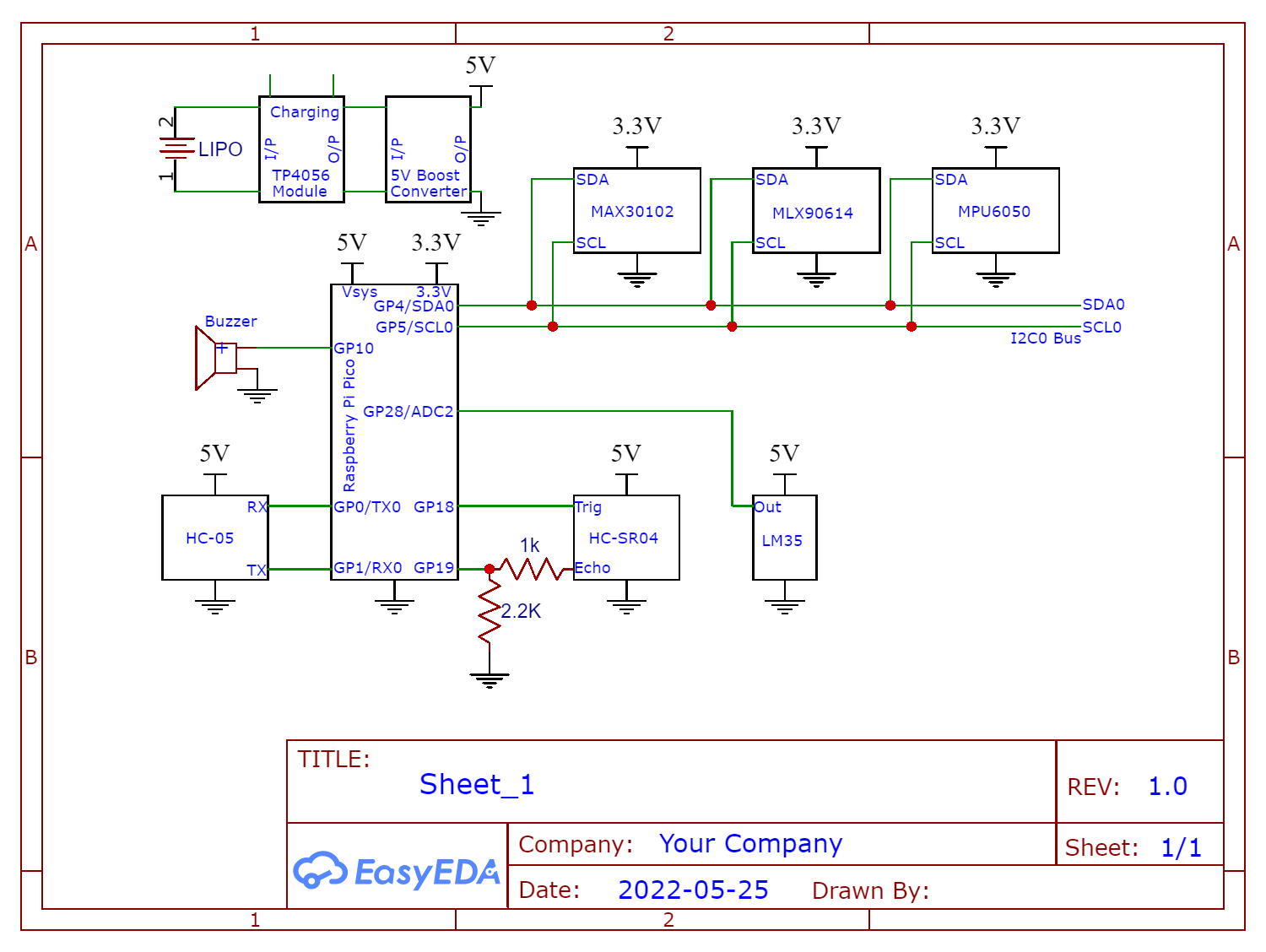
Buzzer is an output to Raspberry Pi Pico.

The main output is shown on the Smartphone.

Also commands from Smartphone are given to the Raspberry Pi Pico.

This happens over Bluetooth through HC-05 (Bluetooth Module).

## Circuit Diagram & Explanation



3.7V, 2000mAh Lithium Polymer Battery generates Power for the System.

It is connected to TP4056 LiPo Charging Module to manage charging and adding Short Circuit Protection.

Our System needs both 5V and 3.3V for operating. So, we boost the output of TP4056 Charging Module to 5V using 5V Boost Converter.

To generate 3.3V we use the Raspberry Pi Pico’s onboard Buck-Boost Converter which can take inputs from 1.8V to 5.5V to generate stable 3.3V for Pico and other modules on the 3.3V Pin. The input to the Buck-Boost Converter is at Pin Vsys.

The 3.3V Pin of the Pico and GND (Ground) is connected to all the devices requiring 3.3V to operate: MPU6050, MLX90614, MAX30102.

The 5V Output Pin of the 5V Boost Converter Module and GND (Ground) is connected to all the devices requiring 5V to operate: HC-SR04, LM35, HC-05.

Raspberry Pi Pico has 2 I2C Bus, I2C0 and I2C1, with separate controllers. Each Bus has two lines, SDA and SCL. The actual GP (GPIO) pins to which the bus is connected can be configured from a list of available choices.

We use GP4 and GP5 as SDA0 and SCL0 (SDA and SCL lines of I2C0 Bus) respectively for the I2C Bus.

I2C uses only these two lines to communicate to multiple devices and sensors. Here we have connected the SDA0 and SCL0 pins of Raspberry Pi Pico to the SDA and SCL pins of:

I²C communication is one of the *communication protocols* that is used by Raspberry Pi Pico. A Communication Protocol is a standardized way devised by people to establish Digital Communication between Electronic Devices.

The Advantage of this communication protocol is that it uses only 2 wires called collectively as I²C Bus (SDA for Data and SCL for Clock) and we can directly connect many devices (up to 112) to this bus with multiple Masters and Slaves, and each master can establish communication with any other device at any given time.

* MPU6050
* MLX90614
* MAX30102

The Trigger Pin of Ultrasonic Distance Sensor (HC-SR04) is connected to GP18 Pin (as Output) of Raspberry Pi Pico and the Echo Pin is connected, through a Voltage Divider to GP19 (as Input) of Raspberry Pi Pico.

The Output Pin of Temperature Sensor (LM35) is connected to ADC2 (GP28) Pin of Raspberry Pi Pico.

A Buzzer is used as output device and is connected to GP10 of Raspberry Pi Pico.

Raspberry Pi Pico has 2 UART Bus, UART0 and UART1, with separate controllers (besides another one reserved for Serial Communications with the PC). Each Bus has two lines, TX (Transmitter) and RX (Receiver). The actual GP (GPIO) pins to which the bus is connected can be configured from a list of available choices.

We use GP0 and GP1 as TX0 and RX0 (TX and RX lines of UART0 Bus) respectively for the UART Bus.

UART communication is one of the *communication protocols* that is used by Raspberry Pi Pico. UART stands for Universal Asynchronous Receiver Transmitter.

The TX pin is the Transmitter Pin and RX pin is the Receiver Pin. Thus, if we want to establish UART communication between two devices, we connect TX Pin of first to RX Pin of second and RX Pin of first to TX Pin of second.

We connect the TX Pin of Bluetooth Module (HC-05) to RX0 Pin of Raspberry Pi Pico, and its RX Pin to TX0 Pin of Raspberry Pi Pico to establish UART Communication between the two devices.

# Program on the Raspberry Pi Pico

## Basic Flowchart

There are three modes in our system namely – Normal Mode, Sleep-Monitoring Mode and Sleep Result Mode.

At each loop, we receive the current mode from the Smartphone. All the modes are continuously being checked.

***Normal Mode***

The Normal mode is the default mode.

It is used to measure the selected parameters –

Ultrasonic Sensor (HC-SR04) is used to measure the distance between the user and the closest individual,

Temperature Sensor (LM35) is used to measure the user’s temperature at the forehead.

IR Temperature Sensor (MLX90614) is used to measure the temperature of the closest individual to the user.

Heart Rate and Oximeter Sensor (MAX30102) is used to measure the heart rate of the user in bpm.

According to the pre-decided thresholds, an actuator in this case a buzzer is used to inform the user of the unusualness of one or more parameters (if Distance<=1m or Target Temperature>=38°C).

This is then further transmitted to Phone via Bluetooth connectivity using our Bluetooth module HC-05 to the Smartphone and displayed on out designed Application.

***Sleep-Monitoring Mode***

Gyroscope-Accelerometer sensor (MPU6050) is used to measure the turns of the user in bed (angular velocity ωz). If the angular velocity ωz is greater than the threshold of 50°/s we increment the turns counter else we continue.

On detection of a turn, a delay is necessary to ensure the turns do not increment as the user is still on his current turn as the ωz is being continuously monitored. This gives the user time to settle.

The current count of turns is transmitted to Smartphone and displayed in the application.

***Sleep Result Mode***

We conclude the User’s sleep as either Deep *Sleep,* *Disturbed Sleep* or Bad Sleep on the basis of the count of turns.

The result is and the final count of turns transmitted to Smartphone and displayed in the application.

## Code (Arduino IDE)

The code was made efficient by using minimum number of libraries. Most of the programming was done by using without any external libraries and only the default Arduino IDE Functions.

We had to avoid all higher levels of abstractions, like libraries, to be more efficient with the coding.

To avoid using too many libraries, we had to study each component and its datasheet from its original manufacturer in order to device an algorithm for the purpose that device served in our project.

The code was optimized by properly structuring it. This was done by declaring and defining a separate function for each separate task. Each function was properly called in the void setup() and void loop() functions in the proper order for the system to work properly.

We have used Arduino IDE’s C Language (modified C) to write this code. Because it is a modification of C, it is one of the most efficient and close-to-hardware High-Level Language.

#include<Wire.h>

#include "MAX30105.h"

#include "heartRate.h"

#define echoPin 19

#define trigPin 18

#define TempPin A2

#define buzzer 10

# define NORMAL\_MODE 0

# define SLEEP\_MONITORING\_MODE 1

# define SLEEP\_RESULT\_MODE 2

#define deepSleep 10

#define disturbedSleep 20

MAX30105 particleSensor;

int mode = 0;

String mode\_str = "0";

float dist;

float bdTemp;

float irTemp;

long heartValue; long lastBeat = 0; long beatPeriod = 1000; float bpm;

int gZr; float gZ; int turns = 0;

int tempAr;

int irTempr;

const float gZ\_thresh = 50;

float pause = 1500;

void normal\_mode();

void sleep\_monitoring();

void sleep\_result();

void calc\_dist();

void calc\_bdTemp();

void calc\_irTemp();

void calc\_bpm();

void calc\_gZ();

void display\_results();

void setup()

{

pinMode(trigPin,OUTPUT);

pinMode(echoPin,INPUT);

pinMode(buzzer, OUTPUT);

Serial1.begin(9600);

Wire.begin();

//Setting up the MAX30102

particleSensor.begin(Wire, 100000);

particleSensor.setup(); //Configure sensor with default settings

//Waking up the MPU6050

Wire.beginTransmission(0x68);

Wire.write(0x6B);

Wire.write(0x00);

Wire.endTransmission(true);

}

void loop()

{

if (Serial1.available()) //Fetch the mode from the phone

{

mode\_str = Serial1.readStringUntil(';');

if (mode\_str == "0")

{

mode = NORMAL\_MODE;

}

else if (mode\_str == "1")

{

mode = SLEEP\_MONITORING\_MODE;

turns = 0;

}

else if (mode\_str == "2")

{

mode = SLEEP\_RESULT\_MODE;

}

}

if (mode == SLEEP\_MONITORING\_MODE)

{

sleep\_monitoring();

}

else if (mode == SLEEP\_RESULT\_MODE)

{

sleep\_result();

}

else if (mode == NORMAL\_MODE)

{

normal\_mode();

}

}

//FUNCTIONS FOR DIFFERENT MODES -------------------

void normal\_mode()

{

calc\_dist();

calc\_irTemp();

calc\_bdTemp();

calc\_bpm();

display\_results();

}

void sleep\_monitoring()

{

calc\_gZ();

if(abs(gZ)>gZ\_thresh)

{

turns++;

delay(pause);

}

display\_results();

}

void sleep\_result()

{

display\_results();

}

//FUNCTIONS FOR DIFFERENT OPERATIONS -------------------

void calc\_dist() //Calculates the Distance

{

digitalWrite(trigPin,LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH); //Triggering the Sensor

delayMicroseconds(10);

digitalWrite(trigPin,LOW);

dist=(pulseIn(echoPin,HIGH)\*0.034)/2;

}

void calc\_bdTemp() //Calculates the Body Temperature

{

bdTemp = (analogRead(TempPin)\*330.0)/1023 - 12.0;

}

void calc\_irTemp() //Calculates the Target Temperature

{

//Getting raw readings

Wire.beginTransmission(0x5A);

Wire.write(0x06);

Wire.endTransmission(false);

Wire.requestFrom(0x5A,3);

tempAr=Wire.read(); //Ambient Temperature

irTempr=Wire.read()<<8 | Wire.read(); //Target Temperature

//Converting the raw data

irTemp = (irTempr/50.0)-273.15;

}

void calc\_gZ() //Calculates the z component of Angular Velocity

{

//Getting raw readings

Wire.beginTransmission(0x68);

Wire.write(0x47); //Send commant to read from 0x3B Register

Wire.endTransmission(true);

Wire.requestFrom(0x68, 2);

//Burst Reading 0x47, 0x48 and storing

gZr=Wire.read()<<8 | Wire.read();

//Conversion of raw data

gZ=((250.0\*gZr))/32750;

if (gZ>250.0 || gZ<-250.0)

{

gZ=0;

}

}

void calc\_bpm() //Calulates the Heart Rate

{

while(true)

{

heartValue = particleSensor.getIR();

if (checkForBeat(heartValue) == true)

{

lastBeat = millis();

break;

}

}

while(true)

{

heartValue = particleSensor.getIR();

if (checkForBeat(heartValue) == true)

{

beatPeriod = millis() - lastBeat;

break;

}

}

bpm = 60 / (beatPeriod / 1000.0);

}

void display\_results() //Display the O/P on phone

{

if (mode==NORMAL\_MODE)

{

Serial1.print(dist);

Serial1.print(";");

Serial1.print(irTemp);

Serial1.print(";");

Serial1.print(bdTemp);

Serial1.print(";");

Serial1.print(bpm);

Serial1.print(";");

Serial1.print("\n");

if (dist<=100 || irTemp>=38)

{

digitalWrite(buzzer, HIGH);

}

else

{

digitalWrite(buzzer,LOW);

}

}

if (mode==SLEEP\_MONITORING\_MODE)

{

Serial1.print(turns);

Serial1.print(";");

Serial1.print("\n");

}

if (mode==SLEEP\_RESULT\_MODE)

{

Serial1.print(turns);

Serial1.print(";");

if (turns<=deepSleep)

{

Serial1.print("0"); //You had a deep sleep!

Serial1.print(";");

}

else if (turns<=disturbedSleep)

{

Serial1.print("1"); //You had a disturbed sleep!

Serial1.print(";");

}

else

{

Serial1.print("2"); //You had a disturbed sleep!

Serial1.print(";");

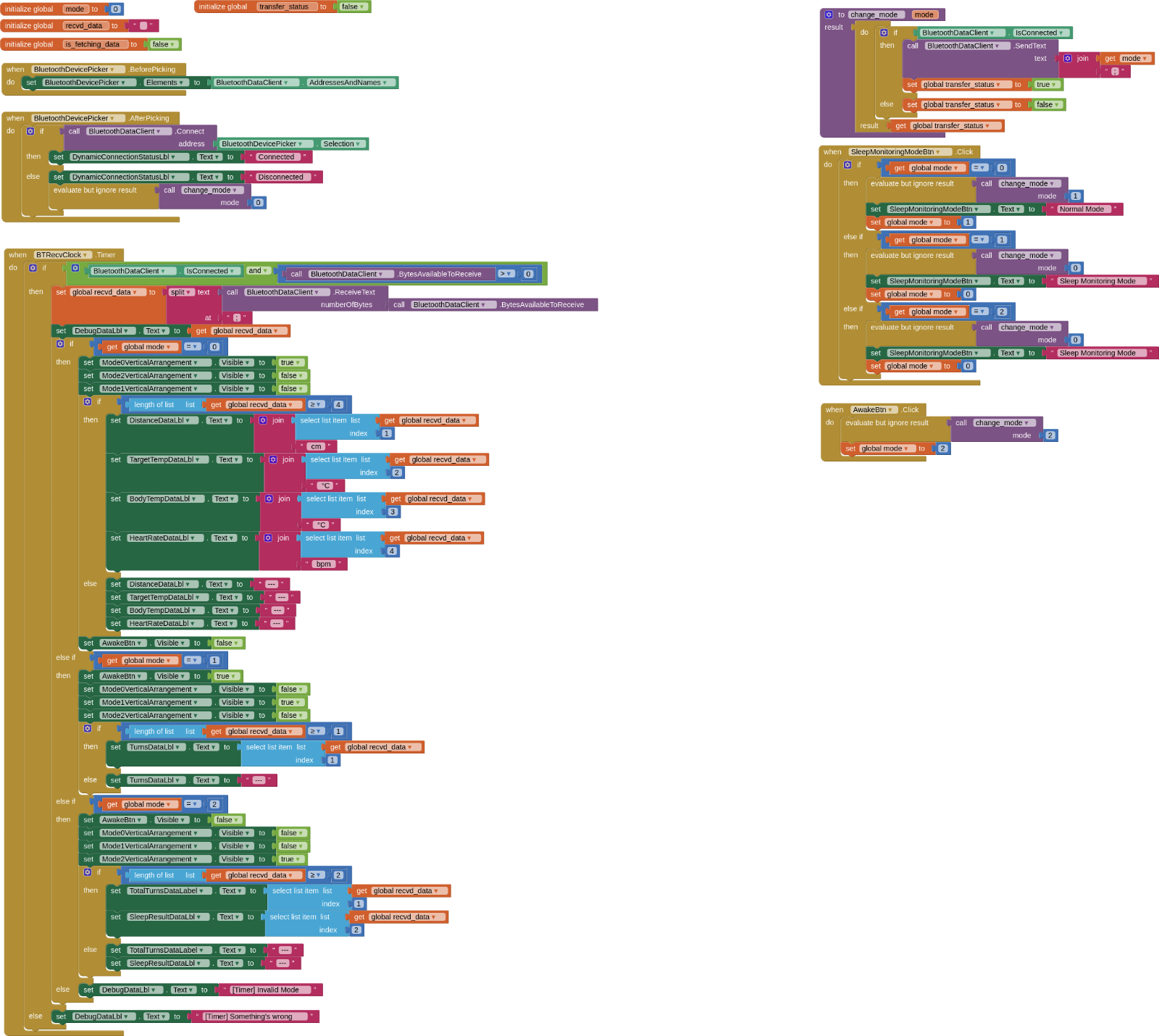
}

Serial1.print("\n");

}

}

# Mobile Application Code



We have used MIT App Invertor to create a custom Smartphone (Android) Application for acting as an Interface between the circuit and the user.

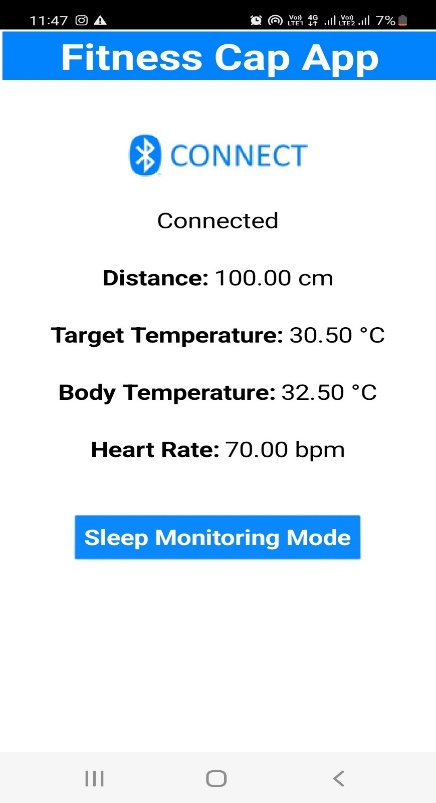
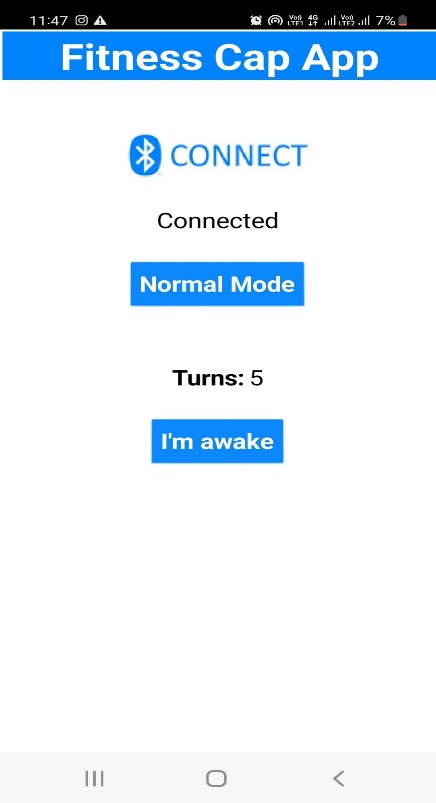
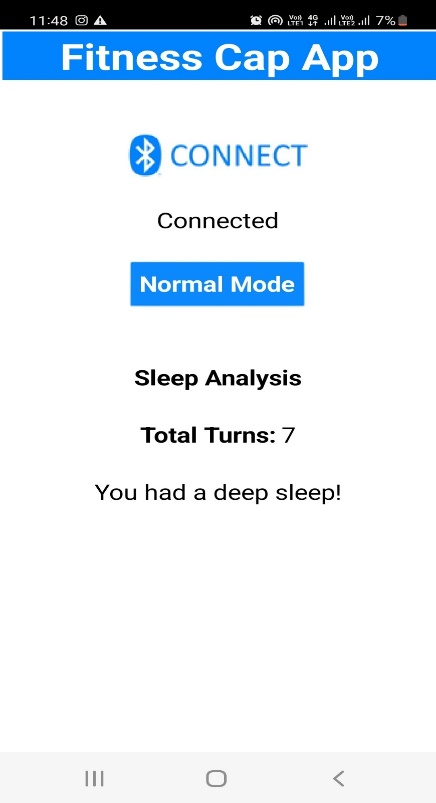
The above is the Block-Code for the Application.

This circuit communicates with the smartphone over Bluetooth.

This application acts an interface to the circuit for:

* Displaying all the outputs
* Controlling the mode in which the circuit is operating

# Output



Sleep Result Mode

Sleep-Monitoring Mode

Normal Mode

We can click on Connect button to connect the smartphone to the Cap.

As we know, there are three modes in our system namely – Normal Mode, Sleep-Monitoring Mode and Sleep Result Mode.

Normal Mode is the default mode. We toggle to all the three modes. We can click on Sleep-Monitoring Mode button to turn on Sleep Monitoring Mode. Then, we can click on the ‘I’m awake’ button to get the Sleep Result Mode.

From the Sleep-Monitoring Mode and Sleep Result Mode, we can go back to Normal Mode by Clicking Normal Mode Button.

***Normal Mode***

From the output we can see that it has measured following parameters: -

Distance: - Measures the distance between user and closest individual which is measured in centimetres using ultrasonic sensor (HC-SR04). User is notified via the buzzer if an object comes within 1m range.

Target Temperature: - The approaching object’s temperature is measured and notified. It is measured in degree Celsius using IR Temperature sensor (MLX-90614). If target’s temperature is equal to or above 38°C, user is notifier via the buzzer

Body Temperature: - The user’s temperature is being constantly measured and notified. It is measured in degree Celsius using temperature sensor (LM-35).

Heart Rate: - The user’s pulse is measured and notified. It is measured in beats per minute using Oximeter sensor (MAX-30102).

***Sleep-Monitoring Mode***

It measures and monitors the number of head rotations while sleeping.

For every rotation (with angular speed above 50°/s) it increments the number of turns and displays on the app. It uses Gyroscope-Accelerometer sensor (MPU6050) for measuring number of turns.

***Sleep Result Mode***

If the user moved his/her head frequently during sleep his/her sleep prediction is made and displayed. It displays the kind of sleep the user had either deep sleep, disturbed sleep or bad sleep.

If the number of turns is in the range 0-10 it shows user had deep sleep.

If the number of turns is in the range 11-20 it shows user had disturbed sleep.

If the number of turns is above 20 it shows user had bad sleep.

In the screenshot, as the user had 7 turns, it showed the user had a deep sleep.

# Pictures & Video of Circuit Implementation

# Conclusion

With the advent of technology, wearable health devices are increasingly helping people to better monitor their health status both at an activity/fitness level for self-health tracking and at a medical level providing more data to clinicians with a potential for earlier diagnostic and guidance of treatment. With seamless tracking of physical activity, sleep patterns, weight, glucose, heart rate and more, doctors can use this technology to collect a large amount of useful data about their patients. However, wearable technology also has potential to save time for patients.

Health & Covid Cap can also be designed with a range of other features to enhance customer experience. The cap is not only useful during a pandemic of viral disease (like Covid-19), but it is also useful in maintaining one’s personal health. There are various devices for different purposes, but they are usually very expensive and unaffordable to the most. There is no single device with all the necessary features to ensure safety especially during pandemic. The Health & Covid Cap helps in tackling all these problems as it is integrated into a single, portable and affordable device, disguised in the form of cap.

The main objective was to create a cap that’s suitable and easy for everyone’s use so that they can protect themselves and stay fit. The Health & Covid Cap is very handy and it’s comprehensible. It’s so simple to use that anyone can use. The existing functions with output on the app are implemented to improve wearers experience. In future we can also enhance and update the cap with different features.

The Health & Covid Cap can be extremely useful for people who are more health conscious, who wants to keep a track of their Health & Covid Cap and they want to monitor each aspect to ascertain their safety.

# Limitations

Every technology or device has some limitations and Health & Covid Cap is no exception. Even though it has many uses in their present form, there are several issues that should be addressed and many features could be added.

The Cap won’t be able to work without a Smartphone/Tablet as it shows the output on it, Thus, it depends entirely on the phone for its working.

Naturally, another important limitation is the battery life of such a device. Fitting a good battery life on such a small factor is tough. We used LiPo battery but that also has its own disadvantages, for example, it is not safe and won’t allow any abuse.

# Future Scope

The Health & Covid Cap has wide future scope as health is always everybody’s concern, a lot of enhances can be done in future, as in we can modify the project by adding a feature i.e., we can show the persons vaccine dose just by using phone number and cowin beneficiary ID and it can display vaccination details more such kind of features can be added.

We can add new features as and when we require, enhancement can easily be implemented under various situations and according to our future needs. It can further be expanded with voice interactive system facility. A feedback system can also be included which provides the state of the Health & Covid Cap and through that we can also be aware of what exact feature does the wearer wants. Design of the cap can also be modified; providing the customer with a large range of options to select according to their type and comfort.

# References & Bibliography

Muhammad Ali Mazidi - AVR Microcontroller and Embedded Systems: Using Assembly and C – 2010 – Prentice Hall

Muhammad Ali Mazidi - Arduino Programming from Beginning to Advanced– 2019 – MicroDigitalEd.com

MPU6050 Register Map- <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>

MLX90614 Datasheet- <https://www.melexis.com/en/documents/documentation/datasheets/datasheet-mlx90614>

LM35 Datasheet- <https://www.ti.com/lit/ds/symlink/lm35.pdf>

# Contribution of the Group Members

|  |  |  |
| --- | --- | --- |
| Yash Patil | 191060078 | App User Interface and Code - Normal Mode |
| Vinayak Wali | 191060075 | App User Interface and Code - Sleep Monitoring and Sleep Result Mode |
| Mayuri Angule | 191061006 | RPi Pico Code and Hardware - HC-SR04, HC-05, Bluetooth Implementation, Layout Design |
| Mohammed Ayaz Quadri | 191060079 | RPi Pico Code and Hardware - MAX30102, MPU6050; Circuit Design, Bluetooth Implementation |
| Shreyash Wasnik | 191060076 | RPi Pico Code and Hardware – MLX90614, LM35, Bluetooth Implementation, Circuit Mode Design |