Smart Health Monitoring System

Md. Abdullah Al Muntasir (ET231033)

Md. Minhaj Uddin Raiyan (ET231034)

Course Code: EEE 3506 Course Title: Microprocessor & Interfacing sessional Course Teacher: Dr. Sikder Sunbeam Islam Department: EEE Section: 5AM Date of Submission: 16/06/2025	Remarks:

Items	Remarks (by teacher)	Items for Report	Remarks (by teacher)
A. Used Arduino Uno or PIC Microcontroller.		(i)Front page,	
B. Used Sensor (it is must).		(ii)Title	
C. Proper sensing operation.		(iii)Pointed features of the design	
D. Properly alerted the owner.		(iv) circuit diagram	
E. Proper Simulation output shown.		(v)Description of program/codes,	
F. Hardware implemented properly.		(vi) Proteus (simulation) output snapshot,	
		(vii) hardware working snap shots,	
		(viii) Assessment of issues relevant to this work	
		(ix)Proof of CEP	
		(x) contribution of members.	

ii. **Title:** ***Smart Health Monitoring System*.**

iii. **Pointed Features of the proposed work:**

- This device measures heart rate (BPM), oxygen saturation (SpO_2), and body temperature.
- Alerts the user in case of abnormal readings (e.g., high fever or high pulse).
- Displays all health parameters on a 16x2 I2C LCD.
- LED alert system gives real-time warning when health values exceed safe limits.
- Designed as a wearable or portable device for basic health diagnostics.

iv. **Circuit Diagram & Operation:**

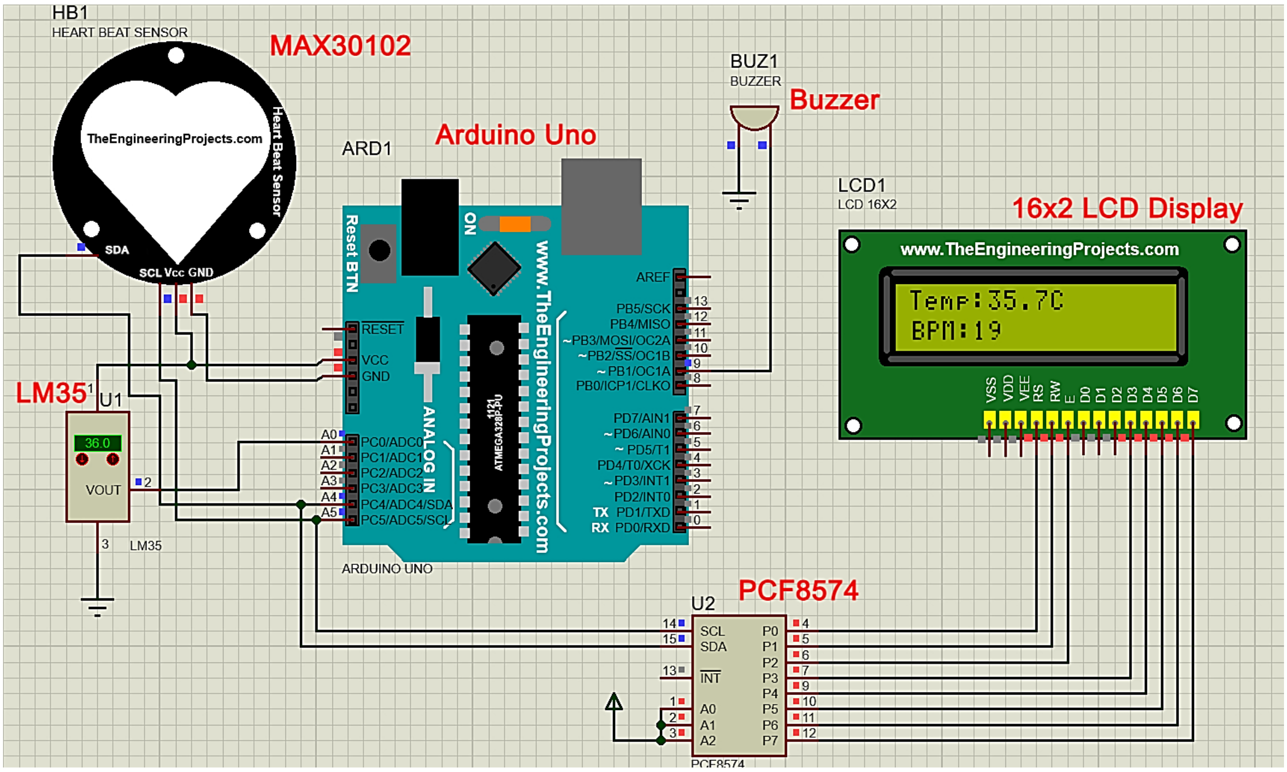

Fig-1: Circuit Diagram of the Project

Operation:

In this circuit two sensors were used: MAX30102 which is a Heartbeat sensor & LM35 which is a temperature sensor.

__**MAX30102**:__ The MAX30102 sensor works by shining red and infrared light into the skin and measuring how much it is absorbed by the blood. As the heart beats, blood flow changes, affecting how much light is absorbed. The sensor uses this to detect heart rate. It also compares the absorption of red and IR light to estimate oxygen saturation (SpO_2) in the blood. It communicates digitally via I2C.

__**LM35**:__ The LM35 is an analog temperature sensor. It produces a voltage that increases linearly with temperatures specifically, 10 millivolts for every degree Celsius. This voltage is read by the Arduino's analog pin and converted to temperature using a simple formula.

The Health Monitoring System continuously measures a person's heart rate, blood oxygen level (SpO_2), and body temperature using the MAX30102 and LM35 sensors. The data is displayed on a 16x2 I2C LCD. If the heart rate exceeds 100 BPM or the temperature goes above 100°F, the system triggers a visual alert by blinking an LED and showing a warning message on the screen. This ensures quick awareness of potential health issues in real time.

To better understand the operation, a **Flowchart** has been provided.

Fig-2: Flow Chart of the System

v. Description of program/codes:

Program	**Description**
#include <Wire.h> #include <LiquidCrystal_I2C.h> #include "MAX30105.h" #include "heartRate.h"	Enables I2C communication Library for 16x2 I2C LCD Library for MAX30105 pulse oximeter Contains heartbeat detection functions

MAX30105 particleSensor; LiquidCrystal_I2C lcd(0x27, 16, 2);	Create sensor object Set up 16x2 LCD at I2C address 0x27

const byte RATE_SIZE = 3;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute;
int beatAvg;
	Size of BPM averaging array
	Array to store last 3 BPM readings
	Index for BPM readings
	Time of last detected beat
	Stores latest BPM
	Stores average BPM

unsigned long lastDisplayUpdate = 0;
unsigned long lastBlinkTime = 0;
int blinkCount = 0;
bool blinking = false;
bool ledState = false;
unsigned long lastLoopTime = 0;
	For LCD update timing
	For blinking LED timing
	Blink counter
	Whether alert is active
	LED ON/OFF state

void setup() {
  Wire.setClock(400000);
  lcd.init();
  lcd.backlight();
  pinMode(ALERT_LED_PIN, OUTPUT);
	Initializes & Setting up LCD Display

if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
  lcd.setCursor(0, 0);
  lcd.print("MAX30105 ERROR");
 while (1);
  }
	Start MAX30105
	Display error if sensor fails
	Stop execution

particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x3F);
  particleSensor.setPulseAmplitudeIR(0x3F);
	Configure sensor settings
	Set red & IR led brightness

lcd.setCursor(0, 0);
  lcd.print("Health Monitor");
  lcd.setCursor(0, 1);
  lcd.print("Initializing...");
  delay(1500);
  lcd.clear();
}
	Show welcome message
	Wait 1.5 seconds
	Clear LCD

void loop() {	runs repeatedly
unsigned long now = millis();	Get current time
long irValue = particleSensor.getIR(); long redValue = particleSensor.getRed();	Read IR & red signal

int lm35Reading = analogRead(LM35_PIN);
  float temperatureC = (lm35Reading * 5.0 / 1023.0) * 100.0;
  float temperatureF = (temperatureC * 9.0 / 5.0) + 26.0;
	Read temperature sensor
	Convert to °C
	Convert to °F (+ offset)

int spo2 = 0;
  if (redValue > 10000 && irValue > 10000) {
  float ratio = (float)redValue / irValue;
  spo2 = 110 - (ratio * 25);
  if (spo2 > 100) spo2 = 100;
if (spo2 < 0) spo2 = 0;
}
	Initialize SpO$_2$ value
	Only calculate if values are valid
	Red/IR ratio
	Approximate SpO$_2$

if (irValue < 50000) {
    if (now - lastDisplayUpdate > 1000) {
      lcd.clear();
	If strap is not worn, show a warning on the display.

lcd.setCursor(0, 0);
lcd.print("Wear The Strap");
lcd.setCursor(0, 1);
lcd.print("Properly...");
lastDisplayUpdate = now;
}
return;
}	

if (checkForBeat(irValue)) {
long delta = now - lastBeat;
lastBeat = now;
beatsPerMinute = 60 / (delta / 1000.0);
if (beatsPerMinute > 20 && beatsPerMinute < 255) {
rates[rateSpot++] = (byte)beatsPerMinute;
rateSpot %= RATE_SIZE;
 	
beatAvg = 0;
for (byte x = 0; x < RATE_SIZE; x++)
beatAvg += rates[x];
beatAvg /= RATE_SIZE;
}
}
	Heartbeat detection, calculation, store BPM & calculating average BPM

if (now - lastDisplayUpdate > 1000) {
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("BPM:");
lcd.print(beatAvg);
lcd.print(" Sp:");
lcd.print(spo2);
lcd.setCursor(0, 1);
lcd.print("Temp:");
lcd.print(temperatureF, 0);
lcd.print((char)223); // Degree symbol
lcd.print("F");
lastDisplayUpdate = now;
}
	Update LCD every second

if (!blinking) {
if (temperatureF >= 100) {
startBlinking("FEVER ALERT!", "Temp High!");
} else if (beatAvg >= 100) {
startBlinking("HIGH BPM ALERT!", "Check Heart!");
}
}
	Alert conditions

void startBlinking(String line1, String line2) {
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(line1);
lcd.setCursor(0, 1);
lcd.print(line2);
blinking = true;
blinkCount = 0;
lastBlinkTime = millis();
}
	Alert function

``void handleBlink(unsigned long now) {
  if (blinking && now - lastBlinkTime >= 250)
{
    ledState = !ledState;
    digitalWrite(ALERT_LED_PIN, ledState);
    lastBlinkTime = now;
 	
    if (!ledState) blinkCount++;
 	
    if (blinkCount >= 3) {
      blinking = false;
      digitalWrite(ALERT_LED_PIN, LOW);
    }
  }
}```
	LED blinking logic

vi. **Proteus Simulation Snapshot:**

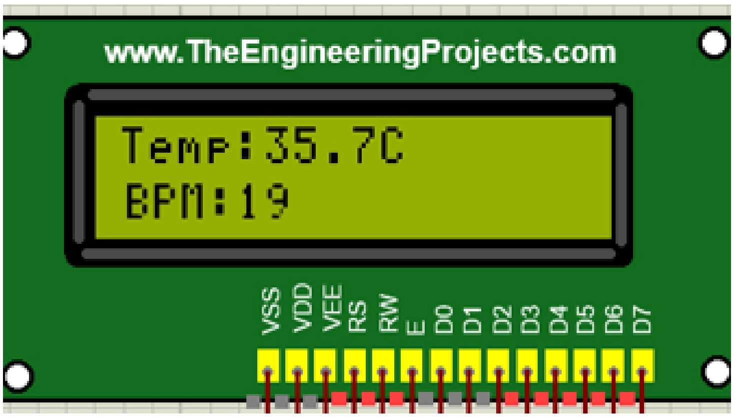

Fig-3: Snapshot of Proteus output – Showing body Temp & BPM

Fig-4: Snapshot of Proteus output – Fever Alert

vii. Hardware working snapshot:

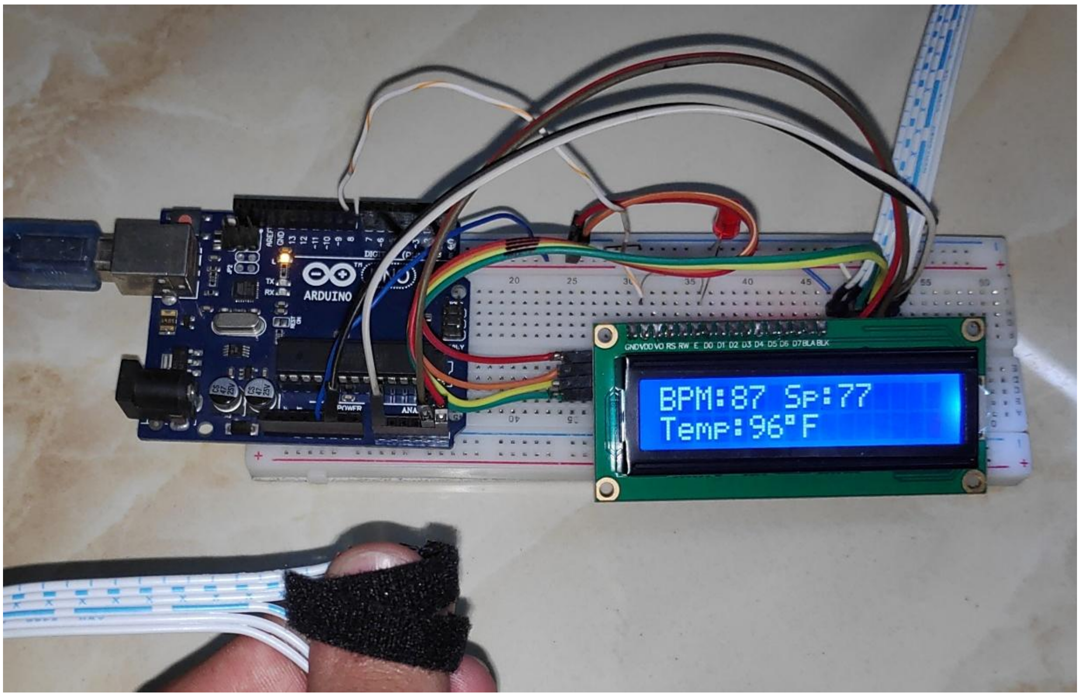

Fig-6: Snapshot of hardware output

viii. Assessment of Considered Issues:

a) **Public health issue:** Assists in early detection of fever and abnormal pulse rates.
b) **Safety issue:** Depends on good sensor contact; false readings may occur otherwise.
c) **Cultural issue:** Encourages health monitoring in underdeveloped or rural communities.
d) **Societal issue:** Useful for elderly care, especially where regular checkups are difficult.
e) **Legal issue:** Requires compliance with basic safety and electrical certifications.
f) **Environmental & Sustainability issue:** Low power usage; can battery powered.

g) **Sustainability assessment:**

- Open-source hardware makes components reusable in other projects.
- Can be repaired or upgraded easily (modular design).
- Extremely low power consumption (milliwatt range).
- Can be powered by USB or rechargeable batteries.
- Very affordable: <$20 for complete setup.
- Promotes innovation in low-resource settings (e.g. rural healthcare).
- Encourages local assembly, reducing dependency on imports.
- Easy to repair and maintain.
- It can be mass-produced or adapted for specific community needs.

ix. Justification of CEP (Complex Engineering Problem):

We are claiming the project as **CEP** (Complex Engineering Problem) **for the following attainment of WK, WP, WA:**

WK3: Knowledge of Electronics, microcontroller was used.

WK4: Specialized knowledge on sensor was used.

WK5: Circuit design knowledge was applied.

WK6: Simulation and Hardware tools were used.

WK7: Societal issue was considered.

WP1: WK3-WK6 are addressed.

WP3: Multiple ways of design and microcontroller coding is possible.

WP4: The project work is involved with infrequently encountered issues as **medical** related matters that is not studied in electrical engineering discipline.

WA1: The project involves diverse resources (People, money, equipment etc)

x. Contribution of the members:

Students ID	Name	Role	Contribution	Signature
ET231033	**Md. Abdullah Al Muntasir**	**Team Leader**	i. Planned the total work steps, ii. Bought the equipment, iii. Contributed to design, iv. Hardware implementation, v. Data Collection. **	
ET231034	**Md. Minhaj Uddin Raiyan**	**Member**	i. Bought the equipment, ii. Contributed to design, iii. Hardware implementation, iv. Simulation, v. Report Writing. **	