"SMART HEALTH MONITORING SYSTEM "

AIM: Write an application using Raspberry Pi/Arduino for smart health monitoring system which records heart beat rate and temperature and also sends sms alerts if readings are beyond critical values.

COMPONENTS: Arduino Board, breadboard, connecting wires, temperature sensor (BF494B), pulse rate sensor (HW - 827), push button.

THEORY:

1. BF494B Temperature Sensor: The BF494B is a semiconductor temperature sensor that measures ambient temperature. It operates on the principle of resistance changes with temperature. When connected to a circuit, it provides accurate temperature readings, making it suitable for various applications like climate control and temperature monitoring.

2. Pulse Rate Sensor HW-827: The HW-827 is a pulse rate sensor designed to measure the heart rate or pulse rate of a person. It typically uses infrared light to detect blood flow changes in the skin. By attaching it to a fingertip, it can provide real-time pulse rate information, commonly used in fitness and health monitoring devices.

3. Push Button: A push button is a momentary switch that, when pressed, makes or breaks an electrical connection. It's a simple user interface component used to trigger actions in electronic circuits. Pressing the button completes a circuit, allowing current flow. Commonly found in applications like doorbells and digital devices for input.

4. Arduino Breadboard: An Arduino breadboard is a prototyping platform used for experimenting and creating electronic circuits with Arduino microcontrollers. It typically consists of a base with rows of interconnected sockets and a power rail, making it easy to connect and test components. This helps in developing and testing Arduino-based projects.

5. Connecting Wires: Connecting wires are electrical conductors used to establish connections between various components and devices in electronic circuits. They come in different types, such as jumper wires, male-female, and male-male connectors. These wires enable the flow of signals, power, and data within a circuit, allowing for proper functionality and interconnectivity.

PROCEDURE:

1. **Temperature Sensor (BF494B):**
   - Connect the sensor's VCC to 5V on the Arduino.

- Connect the sensor's GND to GND on the Arduino.
- Connect the sensor's analog output to an analog pin-A1 on the Arduino.

2. Pulse Rate Sensor (HW-827):

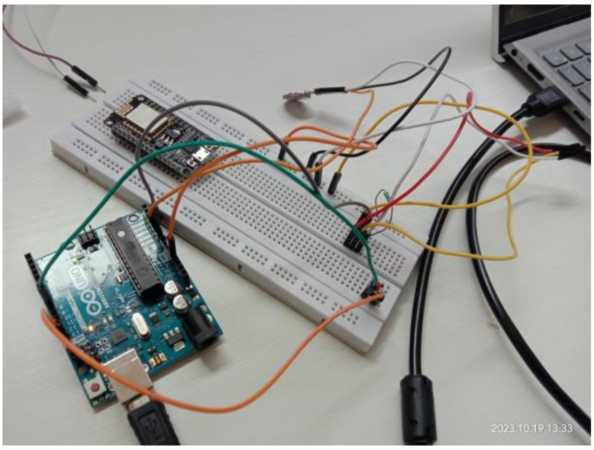
- Connect the sensor's VCC to 5V on the Arduino.
- Connect the sensor's GND to GND on the Arduino.
- Connect the sensor's analog output to another analog pin-A0 on the Arduino.

4. Push Button:

- Connect one terminal of the push button to a digital pin 8 on the Arduino.
- Connect the other terminal of the push button to GND.
- Use a pull-up resistor (e.g., 10kΩ) between the same digital pin 8 and 5V to enable the internal pull-up resistor.

Arduino Code:

```
 #define A 1.009249522E-03
#define B 2.378405444E-04
#define C 2.019202697E-07
int temperatureSensorPin = A1;
int pulseRateSensorPin = A0;
int buttonPin = 8;
bool buttonState = LOW;
bool lastButtonState = LOW;
long lastDebounceTime = 0;
long debounceDelay = 50;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  int temperatureReading = analogRead(temperatureSensorPin); // Read the analog value from the sensor

  int pulseRateReading = analogRead(pulseRateSensorPin);

  // Add your code to convert analog readings to temperature and pulse rate here
  float temperatureCelsius = convertToCelsius(temperatureReading);
  int pulseRateBPM = convertToBPM(pulseRateReading);

  buttonState = digitalRead(buttonPin);

  if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      // When the button is pressed, display the data on the Serial Monitor
      Serial.print("Temperature (°C): ");
      Serial.print(temperatureCelsius);
      Serial.print("\tPulse Rate (BPM): ");
      Serial.println(pulseRateBPM);
    }
    delay(50);
  }
  lastButtonState = buttonState;
}

float convertToCelsius(int analogReading) {
  // Convert analog reading to temperature in Celsius (adjust as needed)
  float resistance = (1023.0 / analogReading - 1.0) * 10000.0; // Convert analog reading to resistance (adjust the reference resistance as needed)

  // Calculate temperature in Kelvin using the Steinhart-Hart equation
float kelvinTemp = 1.0 / (A + B * log(resistance) + C * pow(log(resistance), 3));

// Convert Kelvin to Celsius
float celsiusTemp = kelvinTemp - 273.15;

  return celsiusTemp;
}

int convertToBPM(int analogReading) {
  // Convert analog reading to pulse rate in BPM (adjust as needed)
  float bps = map(analogReading, 0, 1023, 0, 200);
  return bps;
}
```



CONCLUSION: In conclusion, this IoT-based Health Monitoring System enhances healthcare by recording vital signs, detecting critical values, and sending timely SMS alerts for immediate attention.