IOT (Internet of Things) INTERNSHIP BHARAT INTERN INTERNSHIP

Task-3 Health monitoring system:- IoT-Based Health Monitoring System

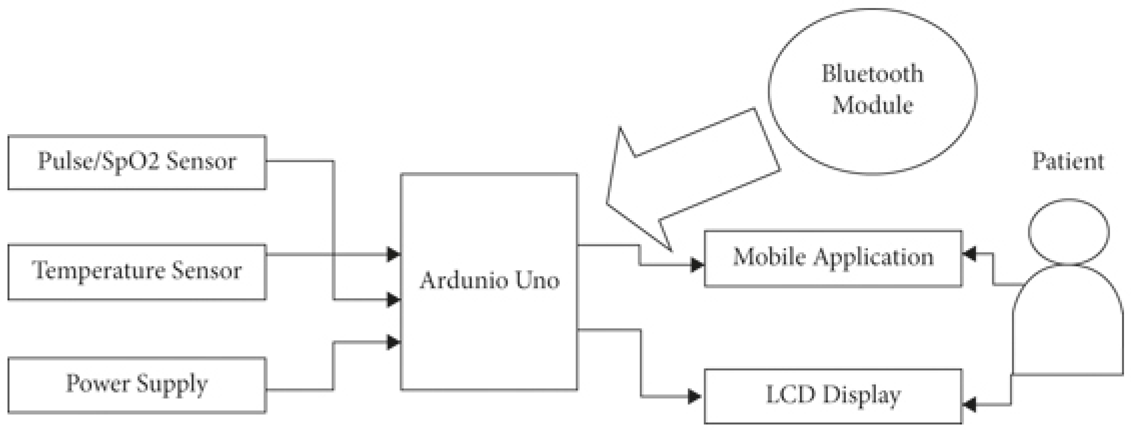
Name:- Sai Kalyan Praneeth velpuri

Introduction:-

This Project introduces a health monitoring system that leverages the Internet of Things (IoT) technology. With advancements in technology, healthcare professionals are constantly seeking innovative electronic devices to easily detect abnormalities in the human body. The use of IoT-enabled technologies offers new possibilities for developing non-invasive and novel clinical support systems. This project mainly focuses on a healthcare monitoring system designed specifically for individuals in rural areas of developing countries like Bangladesh, who lack immediate access to healthcare facilities and face financial constraints. Traditional methods of purchasing expensive medical instruments or frequent hospital visits are not viable for the average population. To address this issue, the proposed system measures vital signs such as body temperature, heart rate, and blood oxygen saturation levels (SpO2) in patients, individuals with high blood pressure, and diabetics. The collected data is then transmitted to a mobile application via Bluetooth. The mobile application, developed using the Android Studio inventor app, receives the data from the IoT device. The system comprises three layers: physical, logical, and application layers. The physical layer consists of sensors that collect data, while the logical layer processes the collected data, manages media access, and enables communication between sensors. The application layer utilizes the processed data from the logical layer to make informed decisions. The primary objective of the system is to enhance affordability and provide convenient access to personalized healthcare for the general population. This project presents an IoT-based solution that simplifies the utilization of complex medical devices at a minimal cost, allowing individuals to monitor their health from the comfort of their homes. To ensure accuracy, a 95 percent confidence interval with a maximum relative error of 5 percent is applied to all measurements of the patient's health parameters. The widespread adoption of these devices as support tools by the general public could significantly impact their quality of life in specific situations.

BLOCK DIAGRAM:-

Here, patients will measure their pulse rate and SpO2 using the max30100 sensor and body temperature using the Lm35 sensor, and patients can see measurement data in the mobile app and LCD display. The data will be shown in the mobile app with the help of a Bluetooth module that will receive data from the Arduino and save it in the cloud. From there, the data will be transferred into the mobile application, and the patients can view the measurement of the health parameters. After measuring the physiological vital data of the human body, it will be sent to the Arduino UNO, which will process the analog data into digital. After that, the data via Bluetooth module will appear on the mobile application. Measured data from the human body can be seen on the LCD display as well.



COMPONENTS USED :-

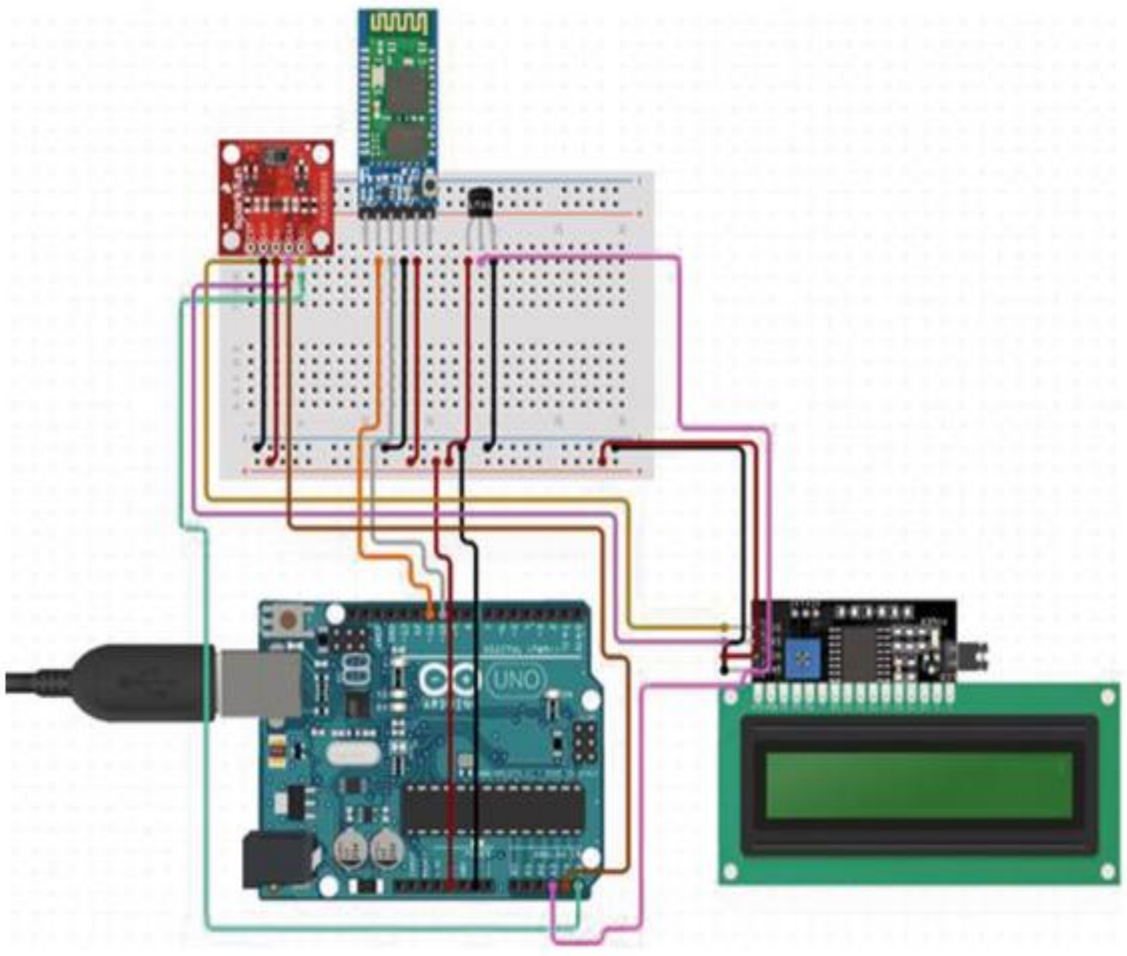
Table 1

Description of the components.

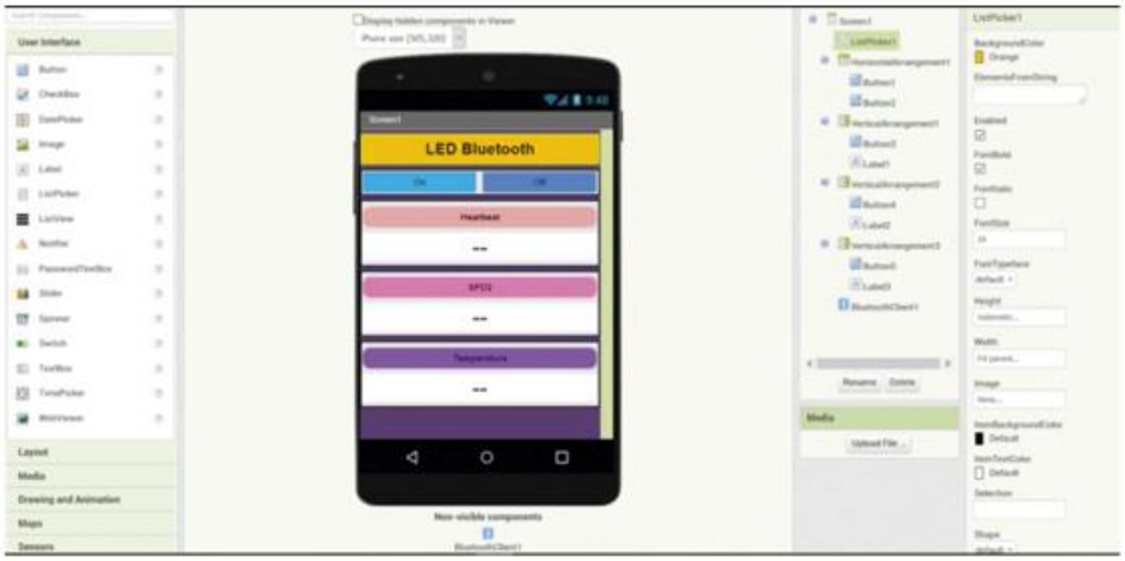
Components	Description
Arduino Uno	An open-source microcontroller based on the 8 bit ATmega238p microchip. It will work as an interface between the sensor and the mobile application described in this paper. This consists of additional components that help the microcontroller, such as a crystal oscillator, serial contact, voltage controller, etc. The Arduino Uno has 14 pins, including 6 pins for analog input, a link to the universal serial bus (USB), an ICSP header, and a reset button. The Arduino Uno provides an ICSP case.
Max 30100	The MAX30100 is a pulse oximetry and heart rate monitor sensor that can be powered by 1.8 V or 3.3 V power supply. A host microcontroller uses the I2C interface to communicate. The MAX30100 pulse oximetry subsystem consists of the ALC, a 16 bit sigma-delta ADC, and a patented disconnected time filter. It is ultra-low-powered, making it suitable for systems operating on batteries.
LM35	The LM35 is an integrated circuit temperature sensor with a temperature-dependent output voltage. It’s a compact, low-cost IC that can determine temperature anywhere between -55 and 150 degrees Celsius. It can easily be interfaced with any ADC or development platform, like an Arduino microcontroller. It can be linked.
Bluetooth module HC-05	The HC-05 is a module that can provide wireless functionality. We are using this module to set up communication between the Arduino and the mobile application. The HC-05 module can easily be coupled with microcontrollers because it uses the

	serial port protocol (SPP). Power the module simply by using +5V and connecting the module's rx pin to the tx and the tx module pin to the MCU rx, as shown below.
LCD display	A 16 × 2 LCD display with I2C can show 16 characters in 2 lines. This means that you will only have four pins for the LCD display: VCC, GND, and SDA, as well as SCL.
Jumper wires	Without the requirement for soldering, flexible wire with connections on each end can be connected to other jumper wires or a pin header.
Bread board	It's a construction base.

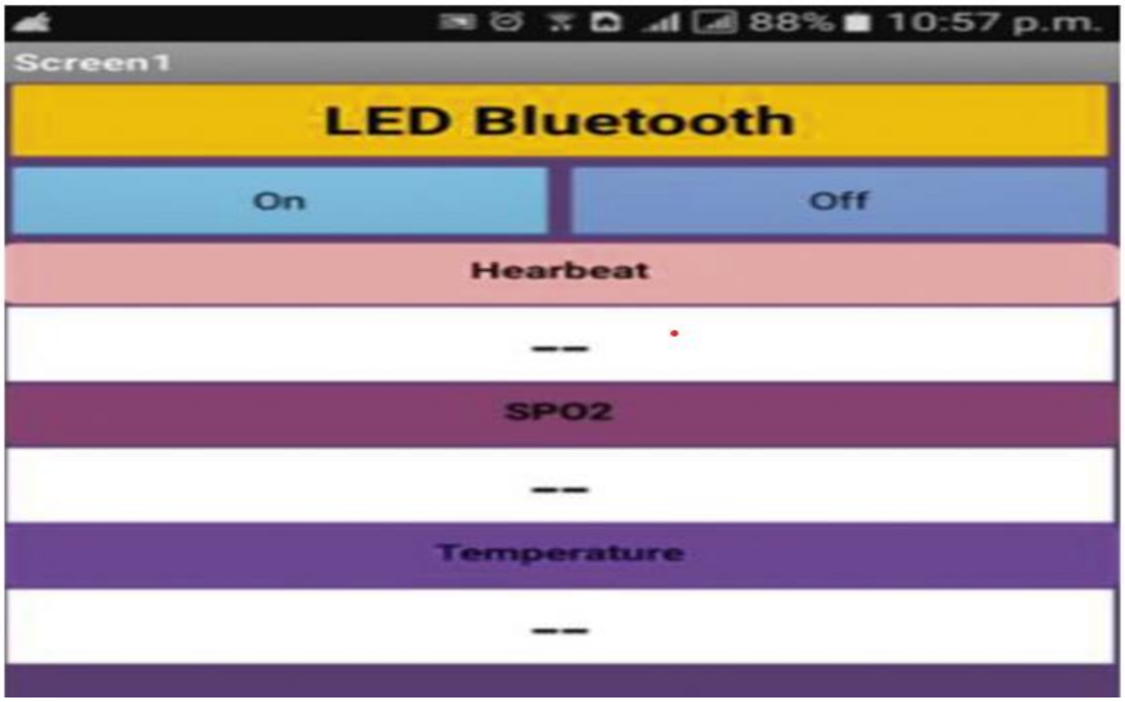
BLOCK DIAGRAM / PIN DIAGRAM:



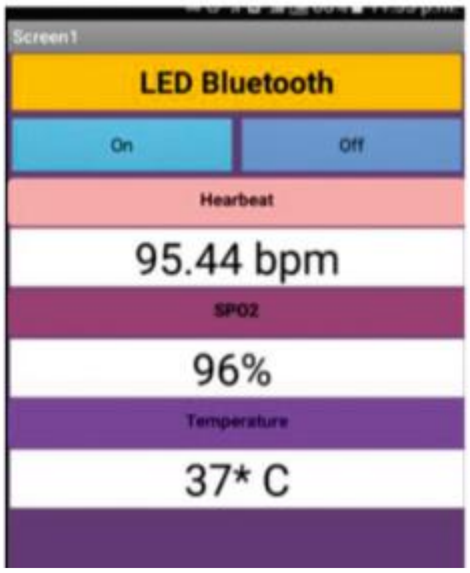
SOFTWARE APP INTERFACE



MOBILE APPLICATION BEFORE THE BLUETOOTH IS ON



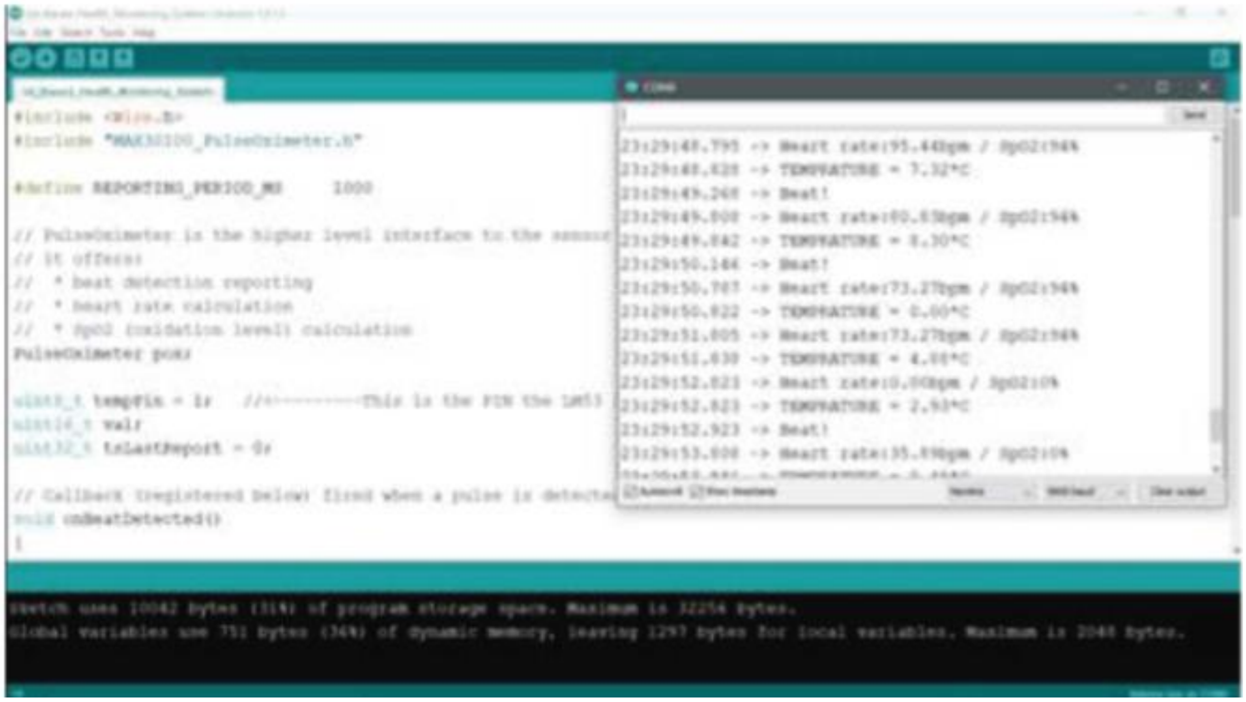
MOBILE APPLICATION AFTER THE BLUETOOTH IS ON



CODE

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>

// Define the pins for the sensors
const int bloodPressurePin = A0;
const int bloodSugarPin = A1;
const int heartRatePin = A2;

// Define the pins for the LCD display
const int rsPin = 12;
const int enPin = 11;
const int d4Pin = 10;
const int d5Pin = 9;

const int d6Pin = 8;

const int d7Pin = 7;


// Create a LiquidCrystal object

LiquidCrystal lcd(rsPin, enPin, d4Pin, d5Pin, d6Pin, d7Pin);


// Create a SoftwareSerial object

SoftwareSerial serial(2, 3);


// Initialize the sensors

void setup() {

  pinMode(bloodPressurePin, INPUT);

  pinMode(bloodSugarPin, INPUT);

  pinMode(heartRatePin, INPUT);


  // Initialize the LCD display

  lcd.begin(16, 2);


  // Initialize the SoftwareSerial object

  serial.begin(9600);

}


// Read the sensor values and send the data to the cloud

void loop() {

  int bloodPressure = analogRead(bloodPressurePin);

  int bloodSugar = analogRead(bloodSugarPin);
  int heartRate = analogRead(heartRatePin);

  // Send the data to the cloud
  serial.print(bloodPressure);
  serial.print(",");
  serial.print(bloodSugar);
  serial.print(",");
  serial.print(heartRate);
  serial.print("\n");

  // Update the LCD display
  lcd.setCursor(0, 0);
  lcd.print("Blood Pressure: ");
  lcd.print(bloodPressure);
  lcd.setCursor(0, 1);
  lcd.print("Blood Sugar: ");
  lcd.print(bloodSugar);
  lcd.setCursor(8, 0);
  lcd.print("Heart Rate: ");
  lcd.print(heartRate);
}
```

OUTPUT



```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#define REPORTING_PERIOD_MS     1000

// PulseOximeter is the higher level interface to the sensor
// it offers:
//  * beat detection reporting
//  * heart rate calculation
//  * SpO2 (oxidation level) calculation
PulseOximeter pox;

uint8_t tempPin = 1;   //<-----------This is the PIN the LM35
uint16_t val;
uint32_t tsLastReport = 0;

// Callback (registered below) fired when a pulse is detected
void onBeatDetected()
{
```

```
23:29:48.795 -> Heart rate:95.44bpm / SpO2:94%
23:29:48.828 -> TEMPRATURE = 7.32*C
23:29:49.268 -> Beat!
23:29:49.808 -> Heart rate:60.63bpm / SpO2:94%
23:29:49.842 -> TEMPRATURE = 8.30*C
23:29:50.144 -> Beat!
23:29:50.787 -> Heart rate:73.27bpm / SpO2:94%
23:29:50.822 -> TEMPRATURE = 0.00*C
23:29:51.805 -> Heart rate:73.27bpm / SpO2:94%
23:29:51.838 -> TEMPRATURE = 4.88*C
23:29:52.823 -> Heart rate:0.00bpm / SpO2:0%
23:29:52.823 -> TEMPRATURE = 2.93*C
23:29:52.923 -> Beat!
23:29:53.808 -> Heart rate:35.89bpm / SpO2:0%
```

```
Sketch uses 10042 bytes (31%) of program storage space. Maximum is 32256 bytes.
Global variables use 751 bytes (36%) of dynamic memory, leaving 1297 bytes for local variables. Maximum is 2048 bytes.
```