Final Project Report

Citizen safety Device

Group 37

Yashvi Pipaliya (AU1841092)

Kesha Bagadia (AU1841011)

Yashvi Gandhi (AU1841033)

Manal Shah (AU1841026)

INDEX

CHAPTER : 1

Introduction and Motivation

In a world of boundless crime and with the advent of independent lifestyles, security is a growing concern, especially for children and women. But with all the technology available to us in recent times now, it's also not hard to build a safety device that can help raise an alarm and gather help. To help resolve this issue we aim to design a GPS based protection system that has dual security features. This system can be turned on by the person in case they even think they would be in trouble. It is useful because in a plain button press alerting system, in case the person is hit on the head from behind or is taken by surprise, they may never get the chance to press the panic button. Our system solves this problem. This feature can be turned on in advance and only the person authenticated to the devices can start the system by fingerprint scan. Once started the device requires the person to constantly press their finger on the system or, else the system will send her location to the authorized personnel number through SMS message as a security measure and also rings a buzzer continuously so that nearby people may realize the situation. The device uses a GPS sensor along with a GSM modem, LCD display, LEDs and Microcontroller based circuit to achieve this system.

CHAPTER : 2

Market Survey or Literature Survey of current products

As the concern of safety is not a new one, there have been an array of attempts to create application systems that provide an additional security layer to the user. In the past, many researchers have done a lot of work for citizen safety using different technologies. In one paper[1], the authors designed a smart ring device for women's safety using Raspberry Pi and a Raspberry Camera Module. The focus here is on helping the victim by sending the victim's real-time location and attacker's information to the police or to specific individuals. This device is manually controlled. If the victim is in trouble, the smartphone app can perform the above stated tasks by the emergency switch of the device in her hand. Although this device is working with the safety of women, due to the access being stationary, it will be difficult to get the assistance, if the victim is away from the access number.

The authors of this other paper[2] designed a smart shoe project emphasizing on two things. One is self-defense, and the other is to send the location of the victim to the precise access numbers. Raspberry Pi, Arduino Uno, GPS, GSM, etc. are used in this project. Raspberry Pi has been used for real-time photo and video streaming. In addition, GPS, GSM, and Electric taser are attached to Arduino Uno, providing real-time location and self defense of victims. This device also sends information about the Victim's location

---

[1] N. R. Sogi, P. Chatterjee, U. Nethra and V. Suma, "SMARISA: A Raspberry Pi Based Smart Ring for Women Safety Using IoT,"2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, 2018, pp. 451-454. https://www.semanticscholar.org/paper/SMARISA%3A-A-Raspberry-Pi-Based-Smart-Ring-for-Women-Sogi-Chatterjee/74f2c77ee69a945cd55d0c65e8429b6a536262ce.

[2] V. Sharma, Y. Tomar and D. Vydeki, "Smart Shoe for Women Safety,"2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan, 2019, pp. 1-4. https://ieeexplore.ieee.org/iel7/8913318/8923121/08923204.pdf.

to the fixation number, just like the previous device. Moreover, even though the self-defense provided here is beneficial, it is likely that the life of the victim will still be threatened if the number of attackers is more than one.

"Reach360" is another android application designed for women safety[3]. A victim can send her location and wary note to the police station, family member, compatriots, and admin by this app. Then the admin will forward the victim's notification to the users within the 100 meters of the victim. One App user can track another user by the unique code generated by the App. This system can help to secure women's movement but its benefit is limited considering two things. Firstly, this system would be more beneficial, if it was fully automated and it did not depend on the admin for finding users within the 100 meters of the victim. Also, if the victim is away from the police station, family members and the app users, then her safety would be compromised. This problem can be avoided by developing a system which manages some fixed volunteers for every area.

Here's a table that summarizes and compares the features of the products that we explored:

	SMARISA	**Smart Shoe for Women Safety**	**Reach360**
Micro-controller	Raspberry-pi	Raspberry Pi , Arduino	Mobile-Device
Wearable Device	Yes	Yes	No
Inbuilt camera	Yes	Yes	No
Contains GPS module	Yes	Yes	Yes

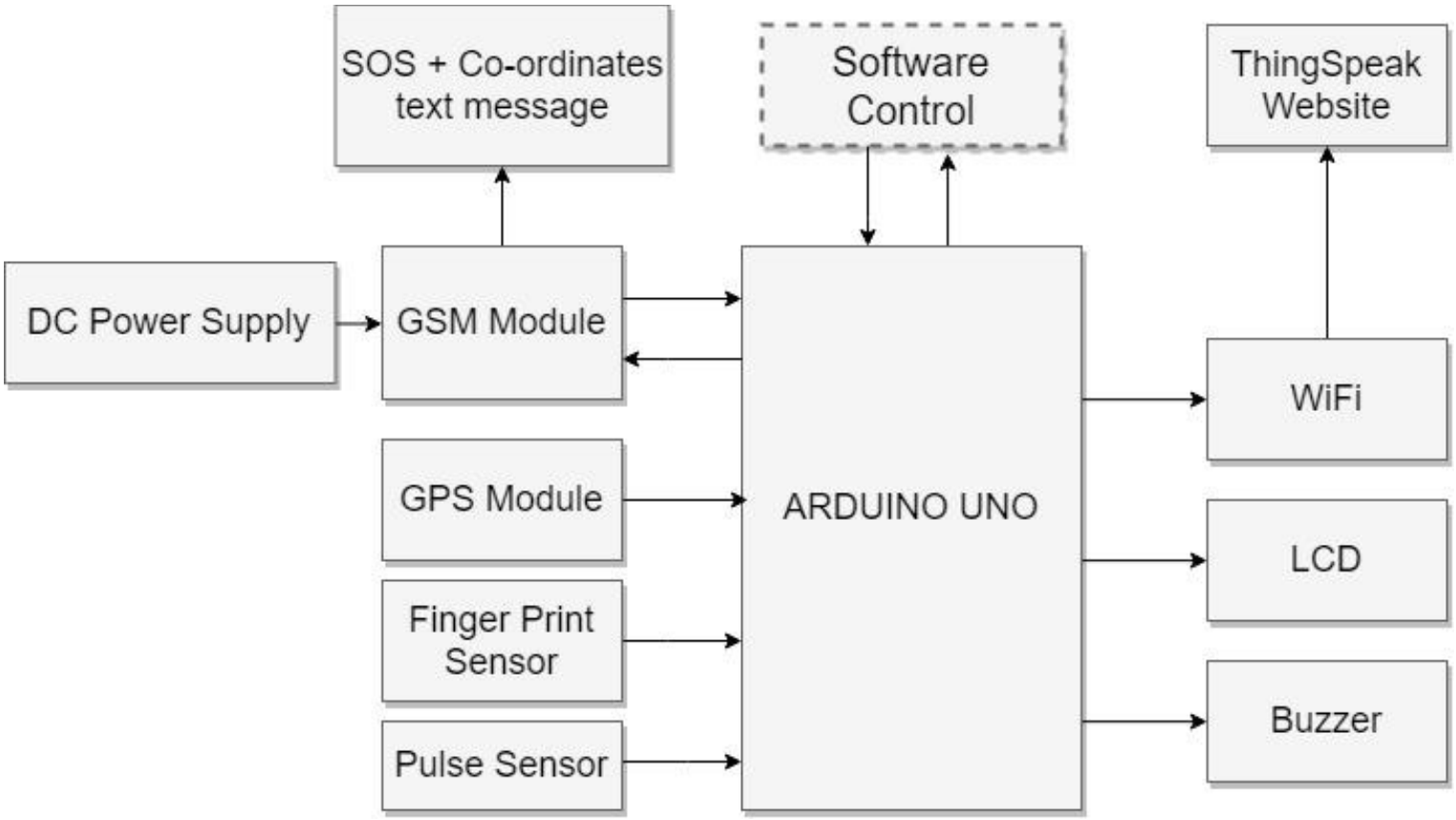
[3] S. Pandey, N. Jain, A. Bhardwaj, G. Kaur and V. Kumar, "Reach360: A comprehensive safety solution," 2017 Tenth International Conference on Contemporary Computing (IC3), Noida, 2017, pp. 1-3. https://ieeexplore.ieee.org/document/8284348.

Alert to emergency contact	Yes	No	Yes
Inbuild defence mechanism	No	Yes	No
Device holder	Ring	Shoe	Phone
Alarm	Yes	No	No
Weakness	Access is stationary, which will cause issues if the victim is away from the access number.	Self-defense provided will not be efficient if the number of attackers is more than one.	If the victim is away from the police station, family members and the app users, then her safety would be compromised.
Strength	Sends the attacker's information alongside the user's location and to the police and contacts.	Comes equipped with an electric taser that helps in self-defense.	Can get more immediate help if there are other users in the same area.

CHAPTER : 3

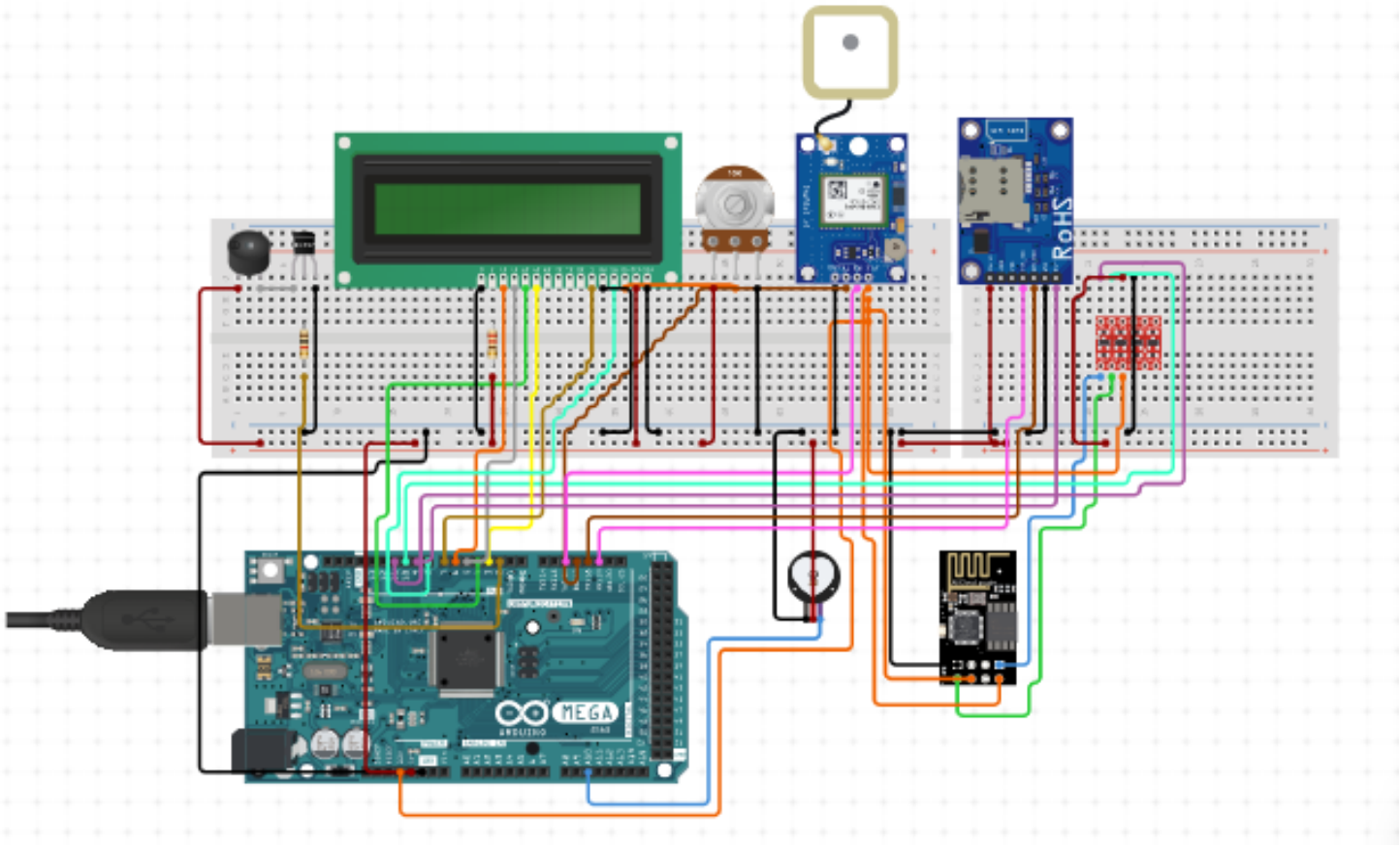
Block Diagram and Explanation

This project is designed with ARDUINO. It presents a safety detection system using GPS and GSM modems, fingerprint and pulse sensors and finally a WiFi module that helps one access information across the globe. This is a detection and messaging system. The GPS Receiver gets the location information from satellites in the form of latitude and longitude. If a finger is not detected on the fingerprint scanner in some time interval then the buzzer will go off. If a finger is still not detected after an additional time interval, the Arduino will collect information regarding the GPS coordinates of the person. It will then process this information and send it to the registered friends of the user via SMS using the GSM modem and the possible danger warning is displayed on LCD display. As an additional measure, constant input of the user's heartbeat is also monitored for sudden spikes due to fear and possible dangers. The same outputs of Buzzer and SMS will be received for the particular criteria. All this information will also be accessed through the internet via the WiFi module so that the friends of the user can be notified of the danger through an additional medium for security reasons.



CHAPTER : 4

Circuit Diagram



- **Fingerprint Scanner** - TTL (GT-511C3) will read and identify fingerprints using an on-board optical sensor and CPU. When the system is activated, it will take constant input from the user's finger pressing on it.

- **Heart Rate Pulse Sensor** will measure heart bpm. This again will be a constant input taken from the user when the system is active.

- **LCD 16x2** will be used to display intermediate messages and help with set up and status.

- **Buzzer** will ring and raise alarm as an output when danger is detected (heart bpm exceeds threshold or fingerprint input is not detected.

- **Ublox NEO-6M GPS Module** retrieves the user' location and provides the current time and date. Output of this module will be extracted when danger is detected and alert is to be sent to the

- **AE GSM MODEM SIM900A**, on insertion of a valid SIM card, can be used for voice calls, sending text messages and accessing the internet. When the system is active and there is any discrepancy in the inputs, like fingerprint is not detected or the heart bpm exceeds the threshold, it will be used to communicate alert messages and location coordinates to the user’s trusted contacts. This will be the output of the system.

- **ESP8266 Wifi Module** allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

CHAPTER : 5

Comparison

We have selected to work on the Arduino microcontroller for our project. Here are some of the features of Arduino UNO and its comparison with other microcontrollers like Raspberry Pi, BeagleBone, Intel's Galileo, Intel's Edison.

	Arduino	**Raspberry-pi**	**BeagleBone**	**Intel's Galileo**	**Intel's Edison**
Strengths	Easy to connect with some LED's, sensors, motors into the board directly	All the Storage is provided from a SD card. You can connect this to your network with an Ethernet Cable.	Similar to a Raspberry Pi but more powerful, Based on the TI Sitara AM335x, an application processor SoC containing an ARM Cortex-A8 core	Same size and shape as an SD card and containing a dual-core Intel Quark x86 CPU at 400 MHz communicating via Bluetooth and Wi-Fi	The Intel Edison module is a SoC that includes an Intel Atom 500MHz dual-core, dual-threaded CPU and an Intel Quark 100MHz microcontroller.
	The Arduino is a microcontroller. The arduino can be programmed, but can't run an operating system	Raspberry Pi are computers. Those devices can run an operating system alone	BeagleBone are computers. Those devices can run an operating system alone	Intel's Galileo is a microcontroller. It can be programmed i, but can't run an operating system	Intel's Editson is a microcontroller. It can be programmed i, but can't run an operating system
	Good combination of Digital and Analog pins.	Perfect availability of general-purpos e I/O pins	Abundant Digital and Analog pins are available .	Good combination of Digital and Analog pins.	Supports external storage Via. MicroSD card.
	Very versatile and extendable due to the availability of	Supports gigabit ethernet, WiFi,	Supports gigabit ethernet, WiFi, and Bluetooth	Supports external storage via. MicroSD card.	Supports all WiFi protocols and Bluetooth

	various external modules for different tasks.	and Bluetooth			
	Available in portable size	Video and Audio output ports are available	Enough memory storage(4GB) is available	Supports Gigabit, Bluetooth and Ethernet	Power adapter or USB port both can be used
	Light weight and comparatively cheapest		DDR3 RAM of 512 KB		Smallest in size

	Arduino	**Raspberry-pi**	**BeagleBone**	**Intel's Galileo**	**Intel's Edison**
Weaknesses	Very low clock frequency (16 MHz)	Requires standard 5V power supply. Highest clock frequency (1.2 GHz)	Does not support external memory (such as MicroSD cards).	Comparatively slower clock frequency (400 MHz) and the product is not longer supported by Intel	Comparatively slower clock frequency (500 MHz) and the product is not longer supported by Intel .
	Very low RAM(2KB).	Huge size	Huge size	Huge size and weight and not easily portable	Not cost effective
	Lowest storage (32KB)	No port for Analog pins.	Audio and video outputs are handled by a microHDMI connection.	. A compatible WiFi chip is must to use WIFi functionality	High Power consumption
	Very tedious and tiring connections	Supports only one WiFi protocol. .	Lack of reference materials available	Does not support audio and video output	Does not provide an Ethernet port.

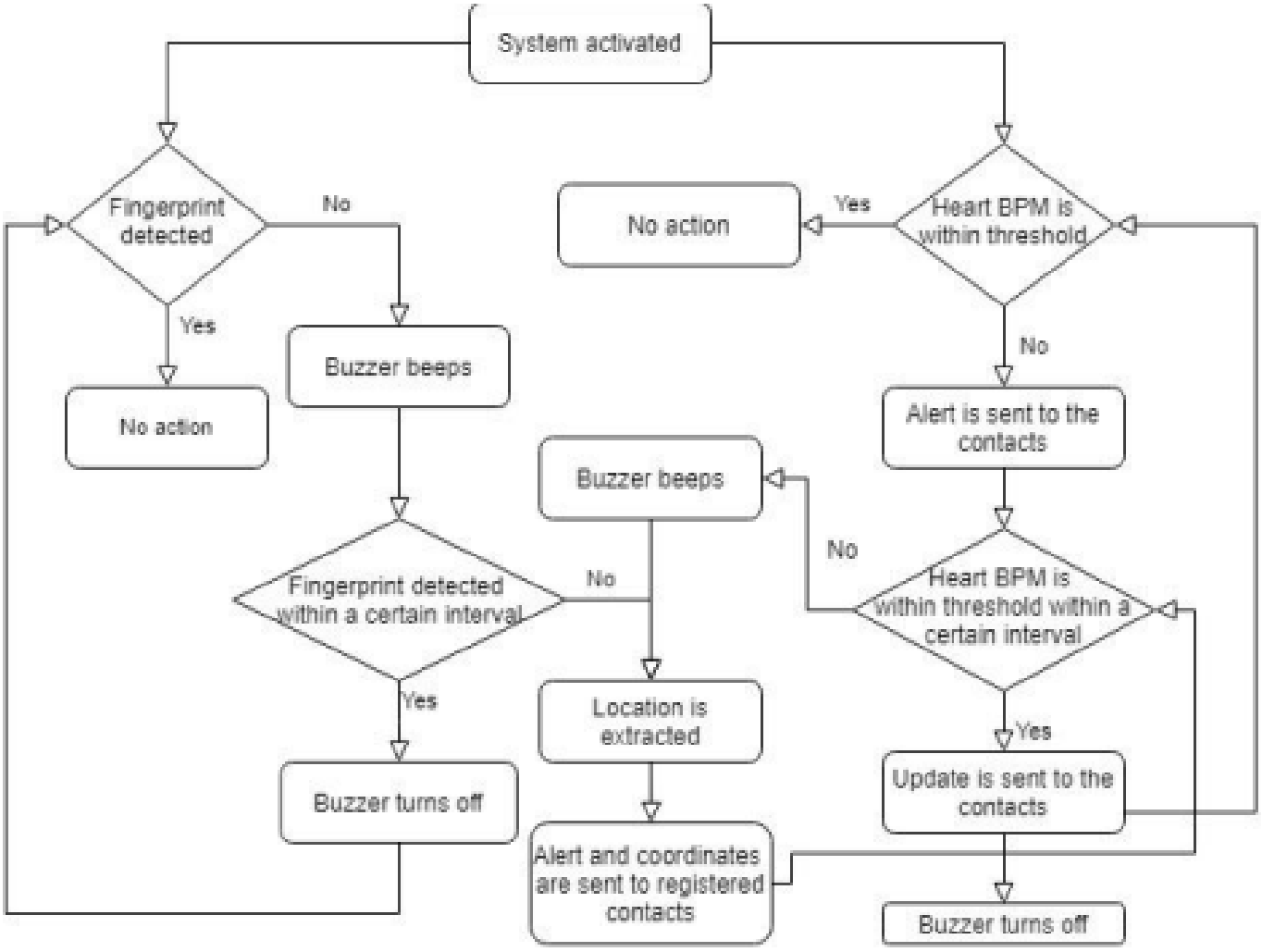
	Slightly high power consumption	Storage is not available, needs a MicroSD card to function.	Quite expensive		It is complicated and difficult to use.

General characteristics

	Arduino	**Raspberry-pi**	**BeagleBone**	**Intel's Galileo**	**Intel's Edison**
Price	$29.95	$35	$89	$50	$79.90
Processor	ATMega 328	ARM11	ARM Cortex-A8	Intel Quark	Intel Atom CPU
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"	4.8 x 2.8	25 x 4mm
Clock Speed	16MHz	700MHz	700MHz	500 MHz, 100MHz	400 MHz
Flash	32KB	SD Card	4GB(microSD)	4 GB eMMC	8 Mb
RAM	2KB	256MB	256MB	256MB	1GB
Input Voltage	7-12v	5v	5v	3.3 to 4.5 V	3V3, 5V
Analog Input	6 10-bit	N/A	7 12-bit	12-bit	12-bit
USB	N/A	USB 2.0	USB 2.0	USB 2.0	USB 2.0
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)	450mA (2.24W)	100mA (0.5W)
EEPROM	1KB	N/A	N/A	11KB	N/A
PWM	6	N/A	8	6	4
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux	Arduino IDE	Arduino, Python, Node.js

CHAPTER : 6

Program Flow chart



CHAPTER : 7

Sensors

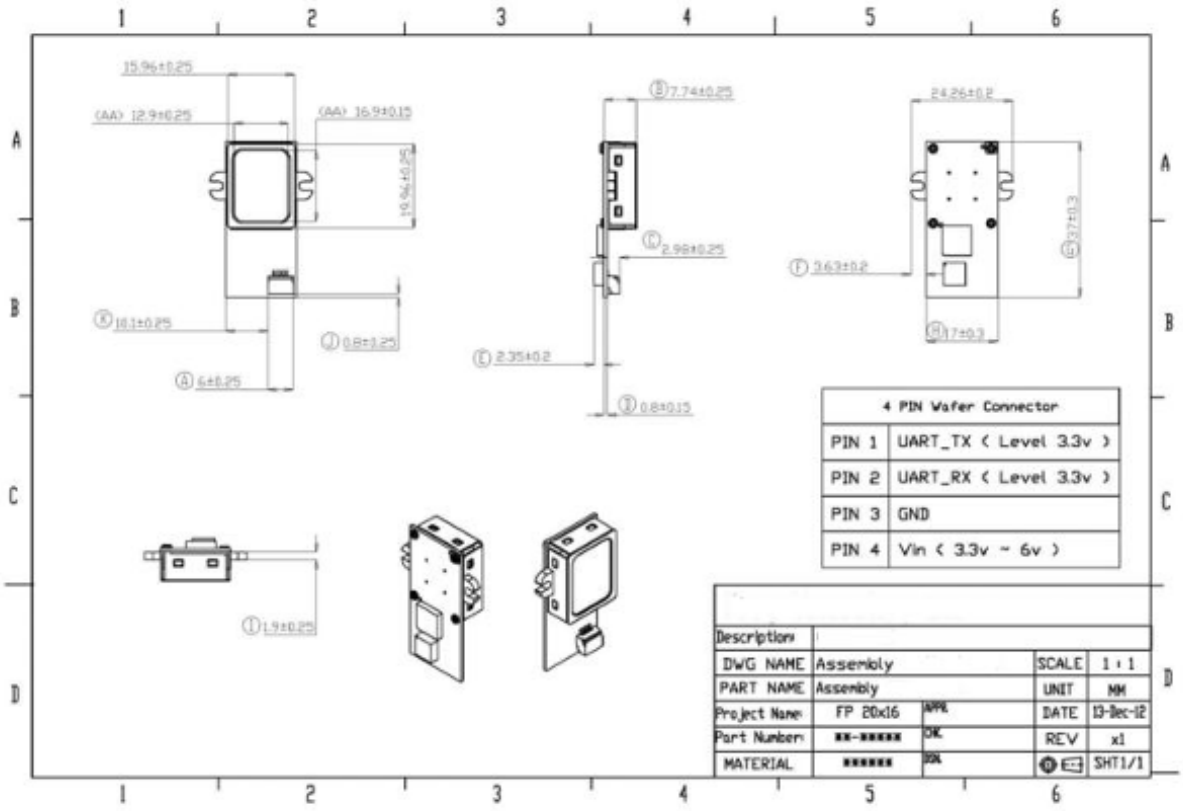
Fingerprint Scanner - TTL (GT-511C3)

1. **Details of operating principle with diagrams:**

   This device is one chip module with:

   - Fingerprint algorithm
   - Optical sensor

2. **Physical dimensions:**

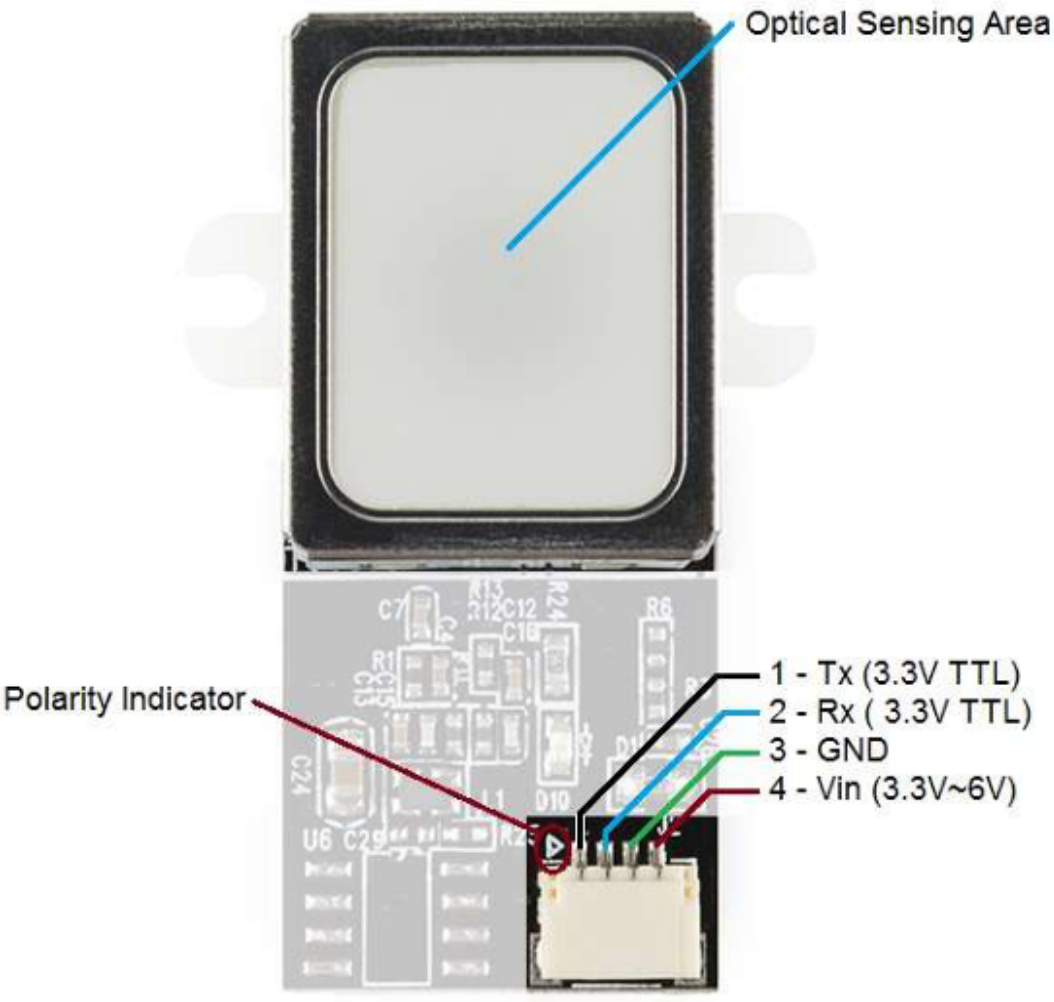


3. **Details of power ratings:**

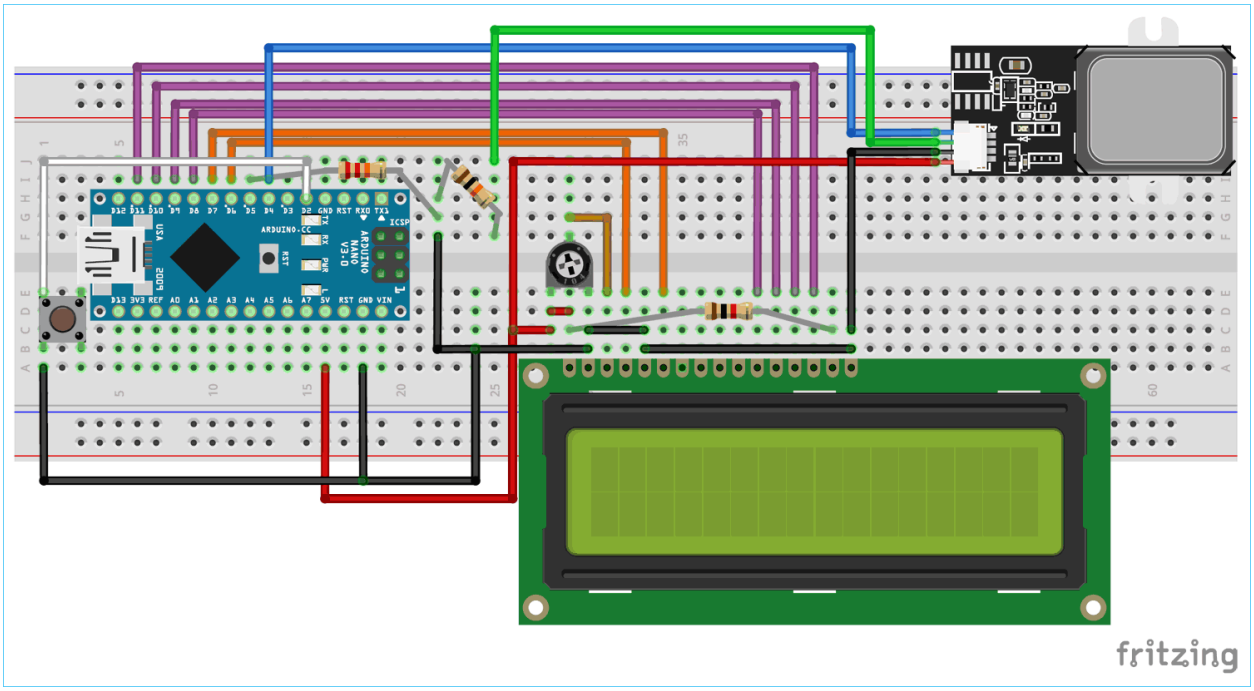
   Operating Voltage: 3.3V ~ 6Vdc

   Operating Current: < 130mA

4. **Pin diagram:**



5. **Type of interface with Arduino/Raspberry Pi:**



6. **Details of interfacing diagram:**

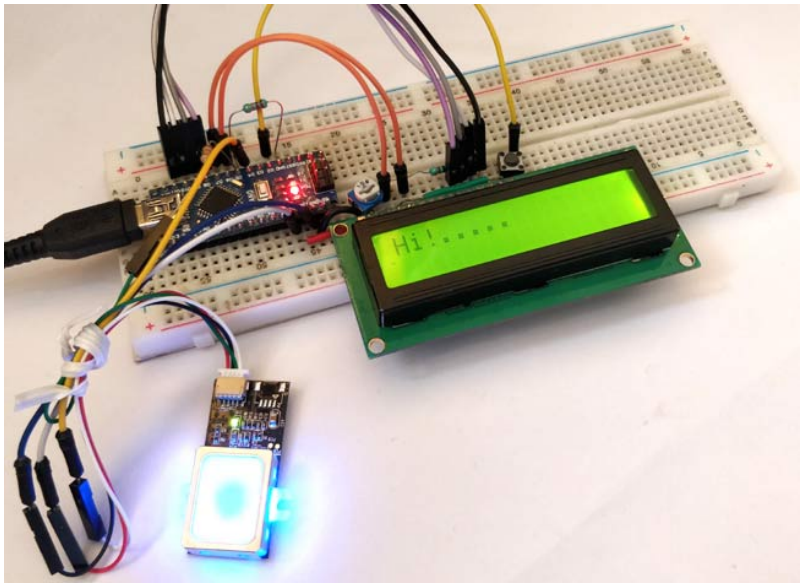
   The GT511C3 FPS has two power pins which can be powered by +5V pin of Arduino and two communication pins Rx and Tx which can be connected to any digital pin of Arduino for serial communication. Additionally, we have also added a push button and a LCD to display the sensor status.

7. **Code to communicate with the sensors**

```
 /* Connect Tx of FPS to Arduino Pin D4 and Rx of FPS to D5*/
#include "FPS_GT511C3.h"
#include "SoftwareSerial.h" //Software serial library
#include <LiquidCrystal.h> //Library for LCD


FPS_GT511C3 fps(4, 5); //FPS connected to D4 and D5
```

```
const int rs = 6, en = 7, d4 = 8, d5 = 9, d6 = 10, d7 = 11; //Mention the
pin number for LCD connection

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);//Initialize LCD method

void setup()
{
  Serial.begin(9600);
  lcd.begin(16, 2); //Initialise 16*2 LCD
  lcd.print("GT511C3 FPS"); //Intro Message line 1
  lcd.setCursor(0, 1);
  lcd.print("with Arduino"); //Intro Message line 2
  delay(2000);
  lcd.clear();
  fps.Open();         //send serial command to initialize fps
  fps.SetLED(true);   //turn on LED so fps can see fingerprint
  pinMode(2,INPUT_PULLUP); //Connect to internal pull up resistor as input
pin
}

void loop()
{
 if (fps.IsPressFinger())
  {
    fps.CaptureFinger(false);
    int id = fps.Identify1_N();
      lcd.clear();
      lcd.print("Detected");
      if (id==200) lcd.print("Unknown"); //If not recognised
      lcd.print(id);
      delay(1000);
  }
  else
  {
    lcd.clear(); lcd.print("Hi!....."); //Display hi when ready to scan
  }
}
```

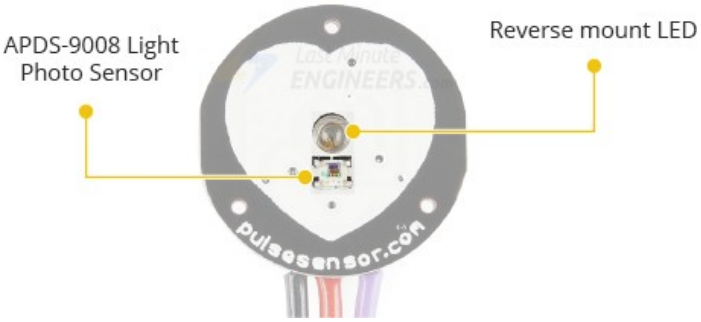
8. **Photos of working hardware:**



Heart Rate Pulse Sensor

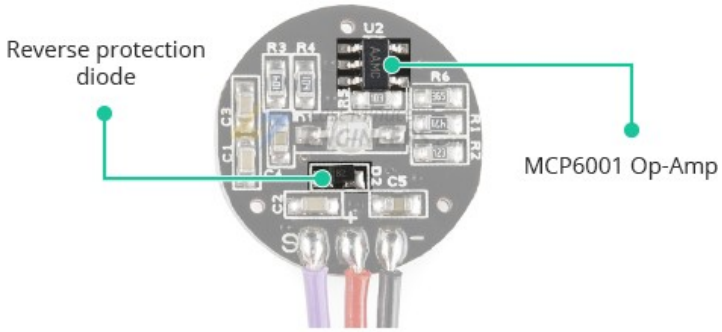
1. **Details of operating principle with diagrams**

   The front of the sensor is the side with the heart logo. This is where you place your finger. On the front side you will see a small round hole, from where the Kingbright's reverse mounted green LED shines.



   Just below the LED is a small ambient light photo sensor – APDS-9008 from Avago, similar to that used in cell phones, tablets and laptops, to adjust the screen brightness in different light conditions.

On the back of the module you will find the rest of the components including a microchip's MCP6001 Op-Amp and a bunch of resistors and capacitors that make up the R/C filter network. There is also a reverse protection diode to prevent damage if the power leads are accidentally reversed.



2. **Physical dimensions**

   L x W (PCB) 15.8mm (0.625″)

   Lead Length 20cm (7.8″)

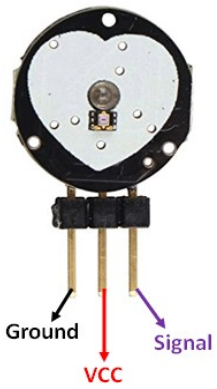
3. **Details of power ratings**

   VCC: 3.0 – 5.5V

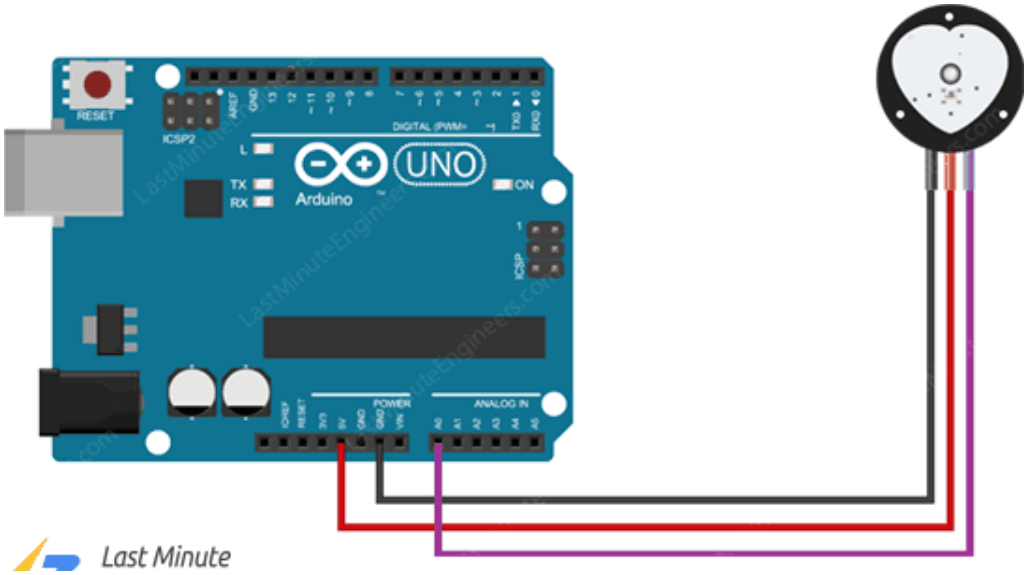
   IMax (Maximum Current Draw): < 4mA

   VOut (Output Voltage Range): 0.3V to Vcc

4. **Pin diagram**



5. Type of interface with Arduino



6. Details of interfacing diagram

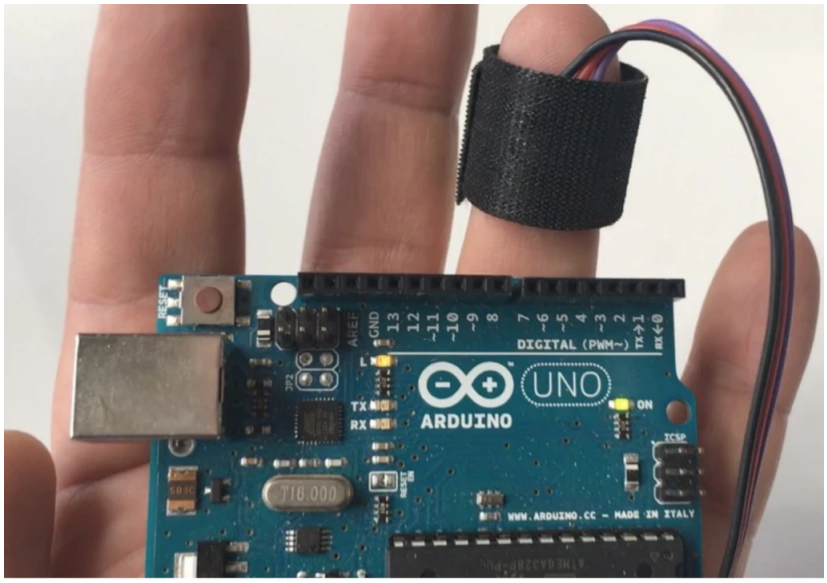
The module can be powered from 3.3 or 5V. The positive voltage connects to '+' and ground connects to '-'. The 3rd 'S' wire is the analog signal output from the sensor and this will connect to the A0 analog input of an Arduino.

7. Code to communicate with the sensors

```
int const PULSE_SENSOR_PIN = 0;    // 'S' Signal pin connected to A0
int Signal;                // Store incoming ADC data. Value can range from 0-1024
int Threshold = 550;       // Determine which Signal to "count as a beat" and which
to ignore.

void setup() {
       pinMode(LED_BUILTIN,OUTPUT);  // Built-in LED will blink to your heartbeat
       Serial.begin(9600);           // Set comm speed for serial plotter window
}

void loop() {

Signal = analogRead(PULSE_SENSOR_PIN); // Read the sensor value
```

```
Serial.println(Signal);                 // Send the signal value to serial plotter

if(Signal > Threshold){                 // If the signal is above threshold, turn on
the LED
digitalWrite(LED_BUILTIN,HIGH);
}
else {
digitalWrite(LED_BUILTIN,LOW);      // Else turn off the LED
}
delay(10);
}
```

8. Photos of working hardware

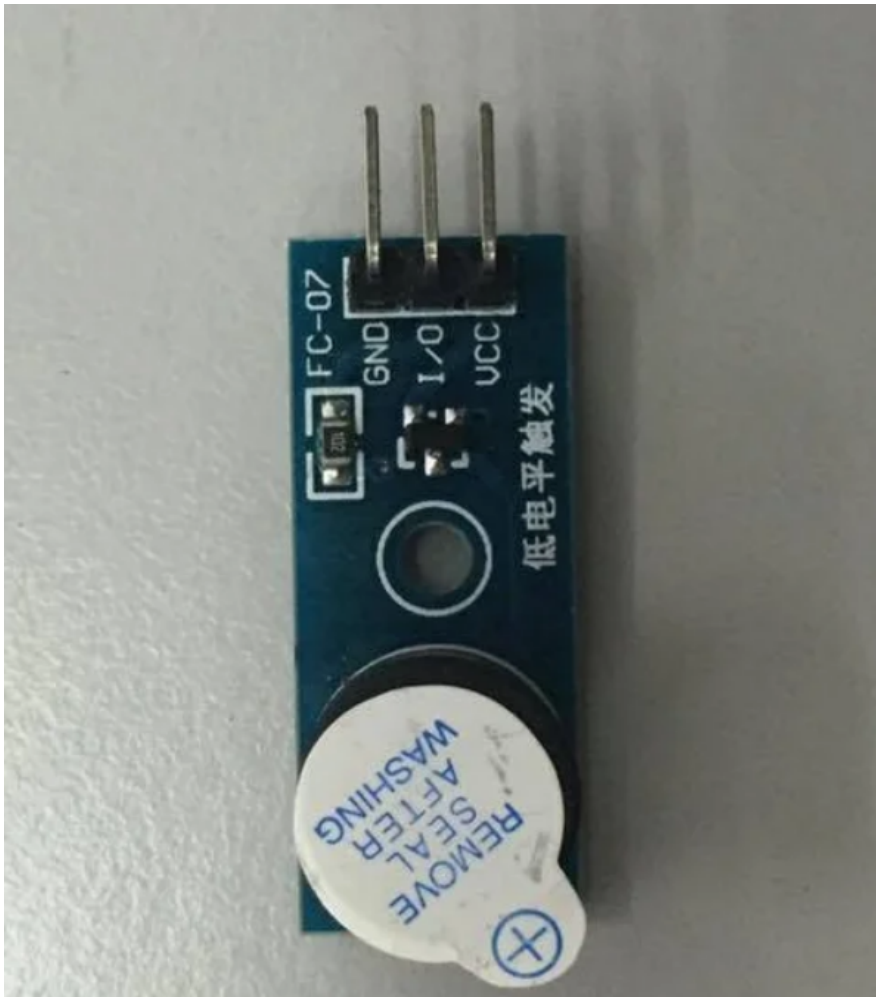


CHAPTER : 8

Actuators

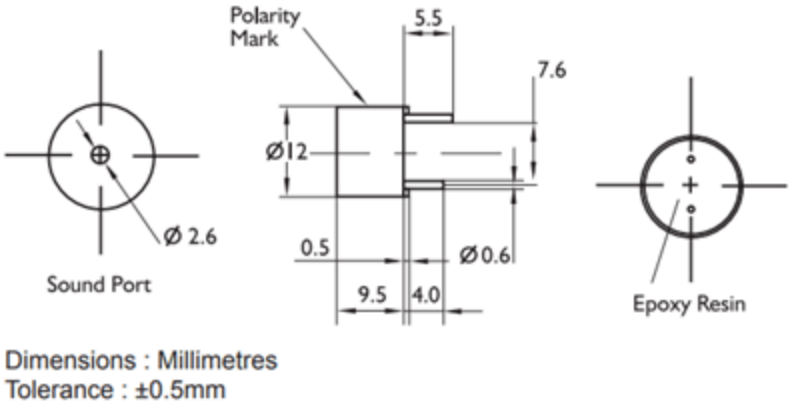
Buzzer

1. Details of operating principle with diagrams



An active buzzer sensor module has a built-in oscillation circuit, thus the sound frequency is fixed. It is able to generate the sound itself. So, you can simply turn it on and off with an Arduino pin, just like the way of turning on and off a Led which is connected to an Arduino board. Besides, this sensor starts beeping when it is being supplied with DC power supply.

2. Physical dimensions



3. Details of power ratings

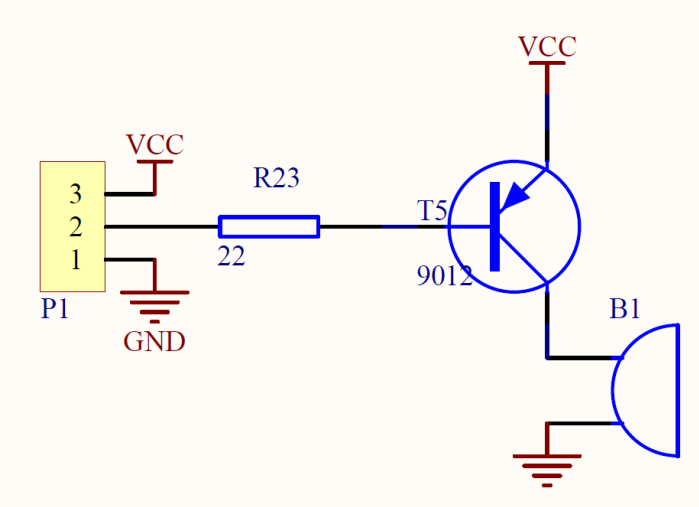
Operating voltage: 3.3V to 5V

Active low sensor module

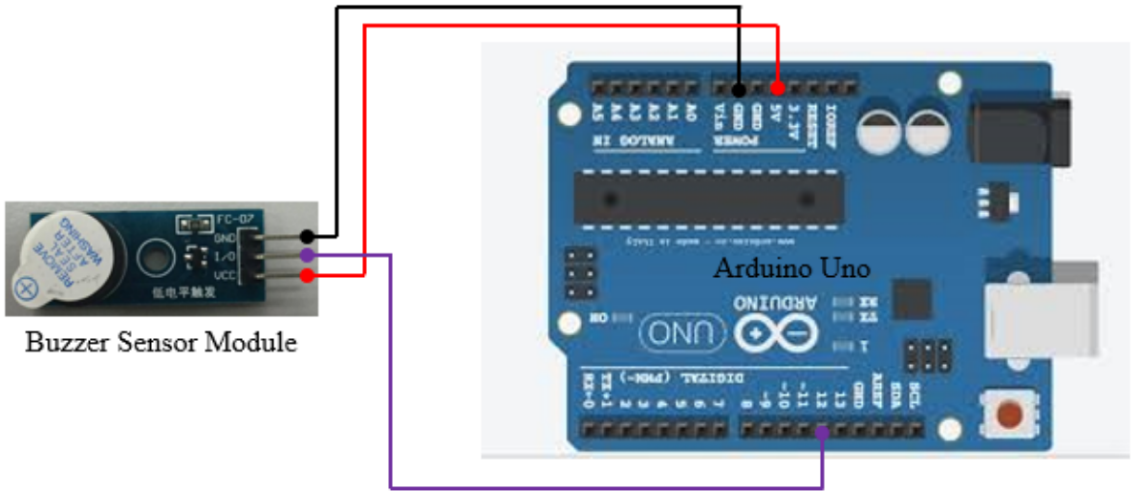
Driver transistor: S8550

Size: 3.2cm x 1.3cm

4. Pin diagram



5. Type of interface with Arduino/Raspberry Pi



6. **Details of interfacing diagram**

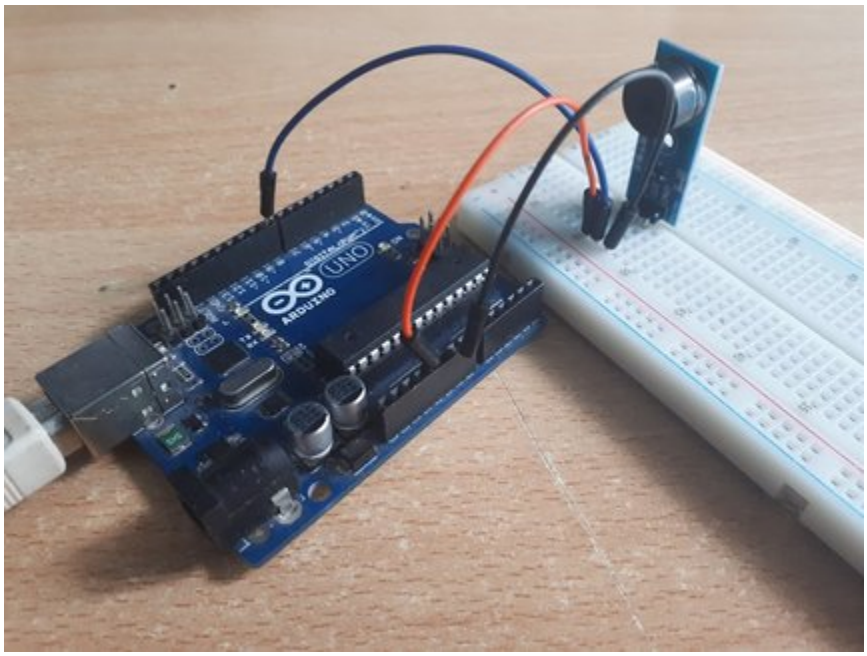
   Here, the cathode is connected to the ground while the anode is connected in series with the resistor to the pin 12 of the arduino.

7. **Code to communicate with the sensors**

```
int buzzPin= 12; // I/O-pin from buzzer connects here
const int wpm = 20; // Morse speed in WPM
const int dotL = 1200/wpm; // Calculated dot-length
const int dashL = 3*dotL; // Dash = 3 x dot
const int sPause = dotL; // Symbol pause = 1 dot
const int lPause = dashL; // Letter pause = 3 dots
const int wPause = 7*dotL; // Word pause = 7 dots

void setup()
{
pinMode(buzzPin,OUTPUT); // Set buzzer-pin as output
}
void loop()
{
dash();
dot();
dash();
dot();
delay(lPause-sPause); // Subtracts pause already taken

dash();
dash();
dot();
dash();
delay(wPause-sPause); // Subtracts pause already taken

}

void dot(){
digitalWrite(buzzPin, LOW); // Tone ON
delay(dotL); // Tone length
digitalWrite(buzzPin, HIGH); // Tone OFF
delay(sPause); // Symbol pause
return;
}
```

```
void dash(){
digitalWrite(buzzPin, LOW); // Tone ON
delay(dashL); // Tone length
digitalWrite(buzzPin, HIGH); // Tone OFF
delay(sPause); // Symbol pause
return;
}
```

8. **Include photos of working hardware.**



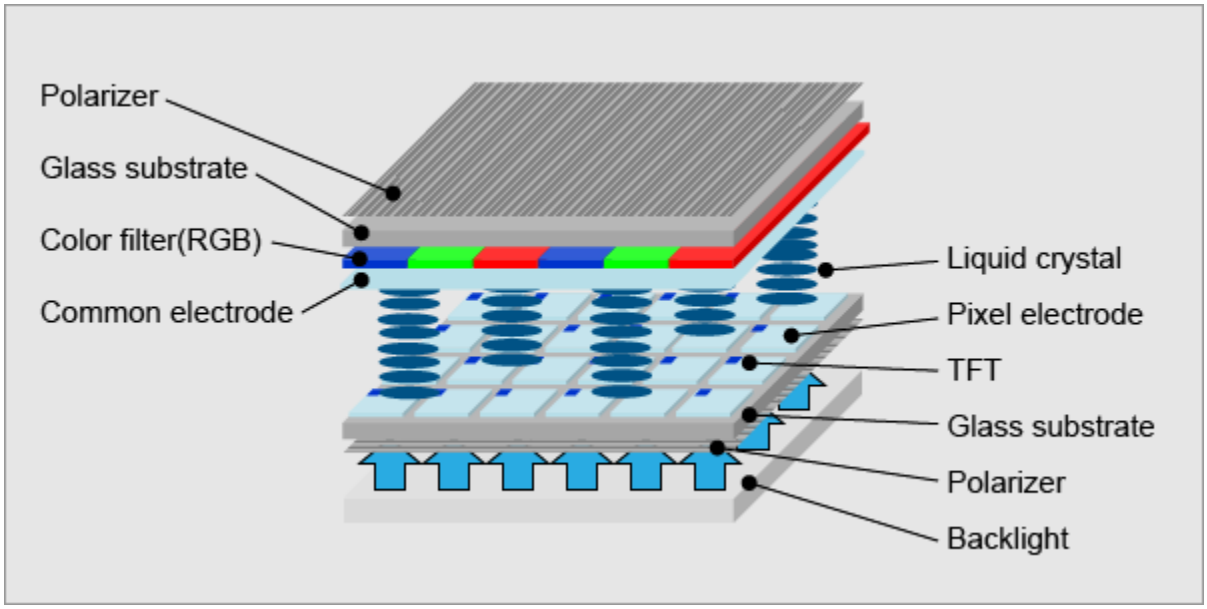
Displays

LCD Display 16X2

1. Details of operating principle with diagrams

In LCD 16×2, the term LCD stands for Liquid Crystal Display that uses a plane panel display technology, used in screens of computer monitors & TVs, smartphones, tablets, mobile devices, etc. Both the displays like LCD & CRTs look the same but their operation

is different. Instead of electron diffraction at a glass display, a liquid crystal display has a backlight that provides light to each pixel that is arranged in a rectangular network.



Every pixel includes a blue, red, green sub-pixel that can be switched ON/OFF. Once all these pixels are deactivated, then it will appear black and when all the sub-pixels are activated then it will appear white. By changing the levels of each light, different color combinations are achievable.

2. Physical dimensions

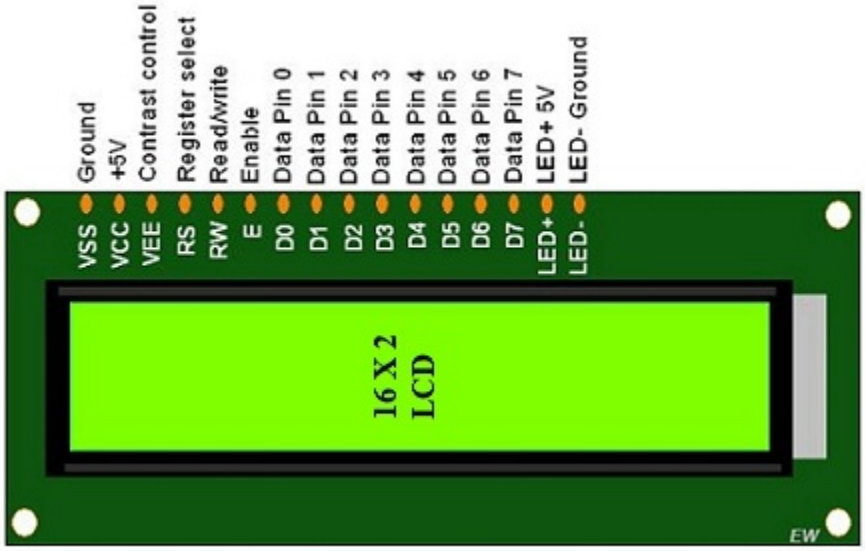
Size:	85.0 x 29.5 x 13.5 mm
Viewing area:	64.5 x 16.4 mm
Dot size:	0.56 x 0.61 mm
Character size:	3.00 x 5.23 mm
Weight:	35 g

3. Details of power ratings

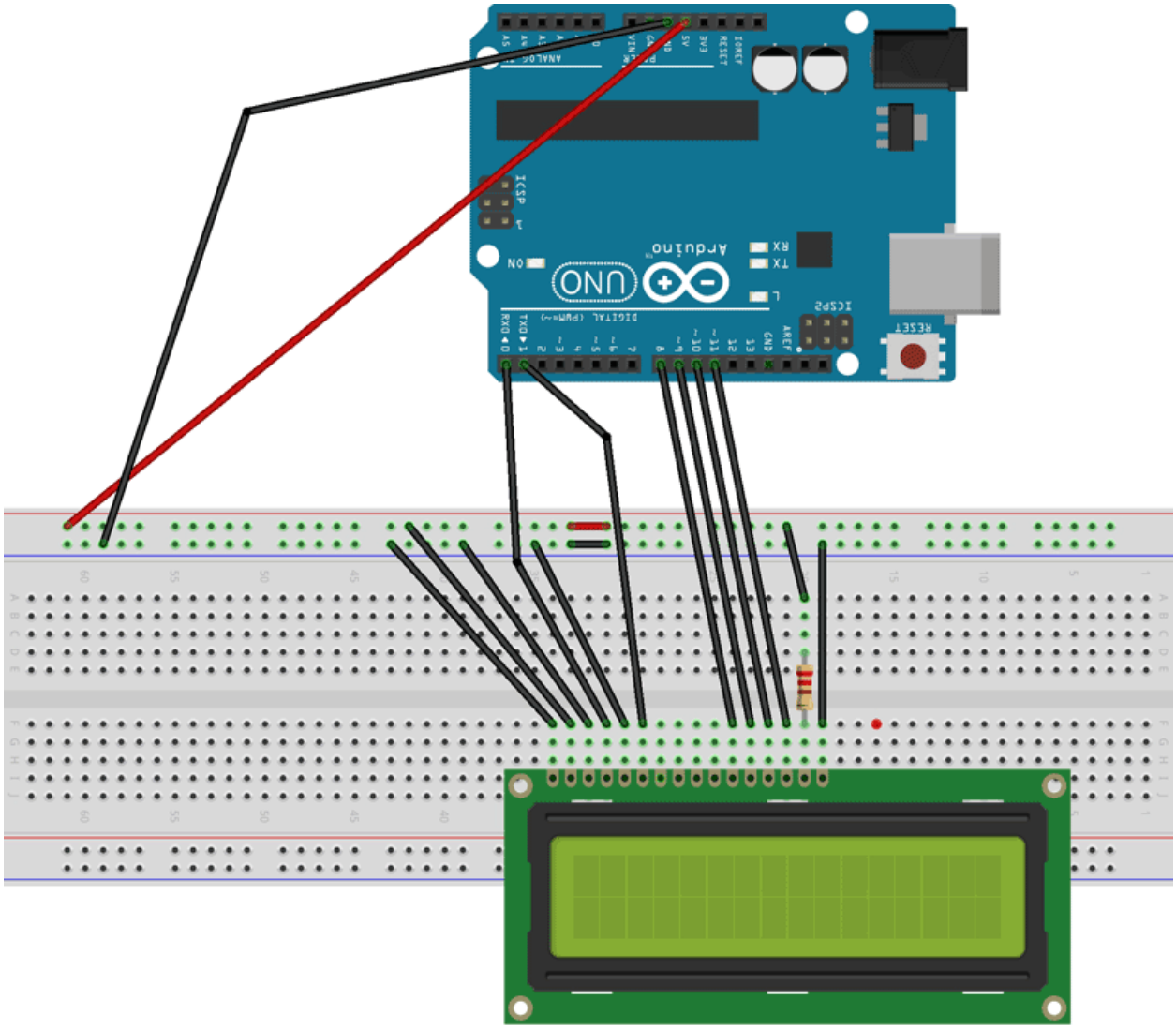
Operating Voltage: 4.7V to 5.3V

Operating Current: 1mA (without backlight)

4. Pin diagram



5. Type of interface with Arduino



6. Details of interfacing diagram

Before interfacing the LCD screen to the Arduino board, a pin header strip needs to be soldered to pin-14 or 16 of the LCD. We can notice this in the following circuit diagram. The following pins need to connect to wire the LCD to an Arduino board.

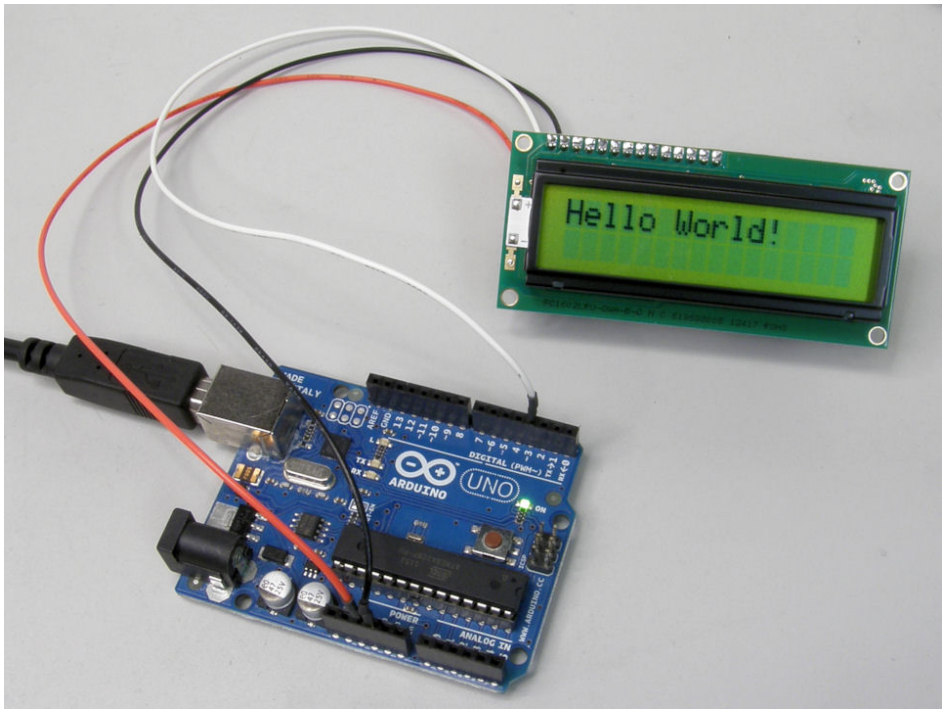
- RS pin of LCD to digital pin-12
- Enable pin is connected to digital pin-11
- D4 pin is connected to digital pin -5
- D5 pin is connected to digital pin- 4

- D6 pin is connected to digital pin-3
- D7 pin is connected to digital pin-2
- Read/Write pin is connected to GND
- VSS pin is connected to the GND terminal
- VCC pin is connected to 5V
- A 220-ohm resistor is connected from LED+ to 5V
- LED is connected to the GND terminal

7. Code to communicate with the sensors'

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 6, d5 = 5, d6 = 4, d7 = 3;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);  // set up the LCD's number of columns and rows:
  lcd.print("Hello World!");  // Print a text to the LCD.
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

8. Photos of working hardware



CHAPTER : 9

Details of website used

Technical Details

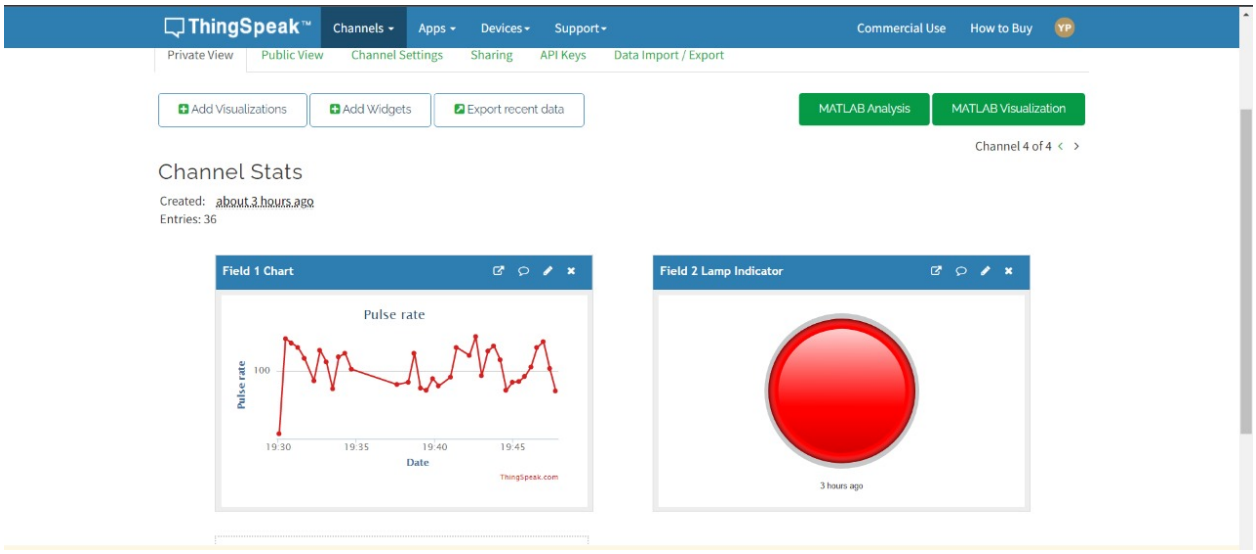
ThingSpeak is an IoT analytics platform service that allows us to aggregate, visualize, and analyze live data streams in the cloud. Data can be sent to ThingSpeak from our devices, create instant visualization of live data, and send alerts.

Some of the key capabilities of ThingSpeak include the ability to:

- Easily configure devices to send data to ThingSpeak using popular IoT protocols.
- Visualize our sensor data in real-time.
- Aggregate data on-demand from third-party sources.
- Use the power of MATLAB to make sense of our IoT data.
- Run our IoT analytics automatically based on schedules or events.
- Prototype and build IoT systems without setting up servers or developing web software.
- Automatically act on the data and communicate using third-party services like Twilio or Twitter.

In our project, ThingSpeak is used to retrieve data of the heart beats of the user from the pulse rate sensor. There is a status variable which indicates whether the person is in danger or not. If the person is in danger its value will be 1 and 0 otherwise.

Photo



Complete Code of Application

```
#include <SoftwareSerial.h>
#define RX 2
#define TX 3
String AP = "Internet";       // AP NAME
String PASS = "abcd"; // AP PASSWORD
String API = "20P5MYR9CGVV2Z32";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int valSensor = 1;
SoftwareSerial esp8266(RX,TX);


void setup() {
```

```
  Serial.begin(9600);
  esp8266.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
}

void loop() {
 valSensor = getSensorData();
 String getData = "GET /update?api_key="+ API +"&"+ field
+"="+String(valSensor);
sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
 esp8266.println(getData);delay(1500);countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");
}

int getSensorData(){
  return random(1000); // Replace with your own sensor code
}

void sendCommand(String command, int maxTime, char readReplay[]) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }

    countTimeCommand++;
  }

  if(found == true)
  {
```

```
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
  }

  if(found == false)
  {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }

  found = false;
}
```

CHAPTER : 10

Details of Communication Protocols

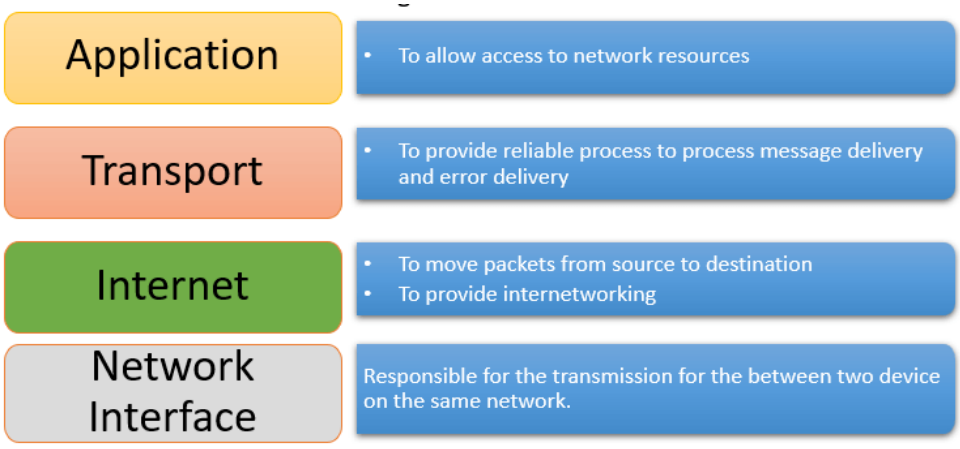
- **TCP/IP PROTOCOL**

  Usage :

  - The GSM Modem SIM900A has an **internal TCP/IP stack** to enable us to connect with the **internet via GPRS.** It is suitable for SMS, Voice as well as DATA transfer applications in the M2M interface.
  - The ESP8266 WiFi Module is a **self contained SOC with integrated TCP/IP protocol stack** that can give any microcontroller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

  Detail :

- TCP/IP stands for Transmission Control Protocol/ Internet Protocol. TCP/IP Stack is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.
- TCP/IP Model helps to determine how a specific computer should be connected to the internet and how data should be transmitted between them. It helps to create a virtual network when multiple computer networks are connected together. The purpose of the TCP/IP model is to allow communication over large distances.
- The functionality of the TCP IP model is divided into four layers, and each includes specific protocols. TCP/IP is a layered server architecture system in which each layer is defined according to a specific function to perform. All these four TCP IP layers work collaboratively to transmit the data from one layer to another.
    - Application Layer
    - Transport Layer
    - Internet Layer
    - Network Interface



Four Layers of TCP/IP model

Advantages :

- It helps to establish/set up a connection between different types of computers.
- It operates independently of the operating system.
- It supports many routing-protocols.
- It enables the internetworking between the organizations.

- TCP/IP model has a highly scalable client-server architecture.
- It can be operated independently.
- Supports a number of routing protocols

Disadvantages :

- TCP/IP is a complicated model to set up and manage.
- The shallow/overhead of TCP/IP is higher-than IPX (Internetwork Packet Exchange).
- In this model the transport layer does not guarantee delivery of packets.
- Replacing protocol in TCP/IP is not easy.
- It has no clear separation from its services, interfaces, and protocols.

- **NMEA PROTOCOL**

Usage :

- NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

Detail :

- NMEA 0183 is a combined electrical and data specification for communication between marine electronics such as echo sounder, sonars, anemometer, gyrocompass, autopilot, GPS receivers and many other types of instruments. It has been defined by, and is controlled by, the National Marine Electronics Association. It replaces the earlier NMEA 0180 and NMEA 0182 standards. In leisure marine applications it is slowly being phased out in favor of the newer NMEA 2000 standard,though NMEA0183 remains the norm in commercial shipping.
- The electrical standard that is used is EIA-422, although most hardware with NMEA-0183 outputs are also able to drive a single EIA-232 port. Although the standard calls for isolated inputs and outputs, there are various series of hardware that do not adhere to this requirement.
- The NMEA 0183 standard uses a simple ASCII, serial communications protocol that defines how data is transmitted in a "sentence" from one "talker" to multiple "listeners" at a time. Through the use of intermediate expanders, a talker can have

a unidirectional conversation with a nearly unlimited number of listeners, and using multiplexers, multiple sensors can talk to a single computer port.

- At the application layer, the standard also defines the contents of each sentence (message) type, so that all listeners can parse messages accurately.
- While NMEA0183 only defines an RS422 transport, there also exists a de facto standard in which the sentences from NMEA0183 are placed in UDP datagrams (one sentence per packet) and sent over an IP network.

- **UBX PROTOCOL**

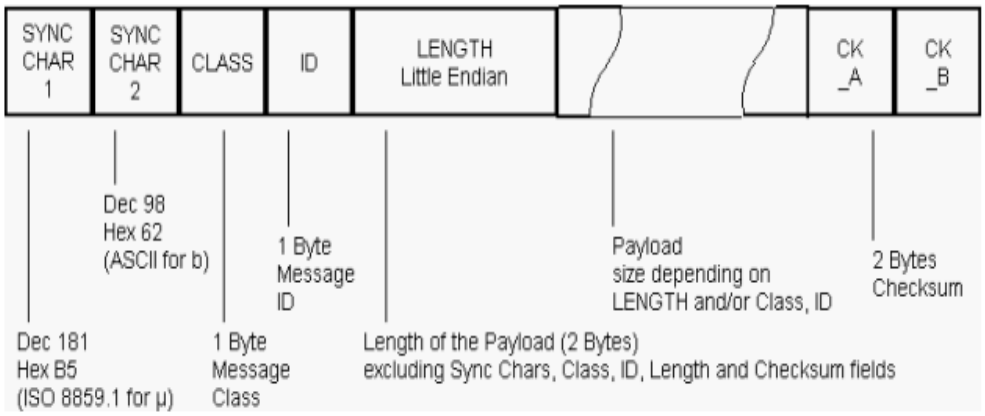
  Usage :

  - NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

  Detail :

  - u-blox GPS receivers use a u-blox proprietary protocol to transmit GPS data to a host computer using asynchronous RS232 ports. This protocol has the following key features:
    - Compact - uses 8 Bit Binary Data.
    - Checksum Protected - uses a low-overhead checksum algorithm
    - Modular - uses a 2-stage message identifier (Class- and Message ID)
  - u-blox receivers are fully configurable with UBX protocol configuration messages (message class UBX-CFG). The configuration used by the u-blox receiver during normal operation is called "Current Configuration". The Current Configuration can be changed during normal operation by sending any UBX-CFG-XXX message to the u-blox receiver over an I/O port. The u-blox receiver will change its Current Configuration immediately after receiving the configuration message. The ublox receiver always uses only the Current Configuration.

- A basic UBX Packet looks as follows:



- **RTCM PROTOCOL**

  Usage :

  - NMEA, UBX and RTCM protocols are used in the NEO-6M GPS Module.

  Detail :

  - RTCM SC-104, or often simply the RTCM standard, is a communication protocol for sending differential GPS (DGPS) to a GPS receiver from a secondary source like a radio receiver. The format does not define the source of the messages and has been used with systems as varied as longwave marine radio, communications satellite broadcasts, and internet distribution.
  - The first widely used version of the format was released in 1990 and was based on the 30-bit long packet used by the GPS satellites, known as a "frame". Each message started with a standardized two-frame header and then one or more data frames following. The frames were designed to be similar to GPS to make integration in GPS receivers easier, but had the disadvantage of having low channel efficiency and limiting the number of messages that could be sent in a given time.
  - A completely new message format was introduced in 2003 for version 3 of the standard which used a variable-length format to improve efficiency and increase

the number of messages that could be sent, which was important for real-time GPS corrections. The new standard also greatly increased the number of possible message types. As part of the standards process, the naming of the standard was changed, and version 3.1 became RTCM Standard 10403.1. As of 2021, the latest version is 3.2, or 10403.2

Table 1: RTCM-3 Message Structure (RTCM Standard 10403.1, 2006)

Preamble	Reserved	Message Length	Data Message	Checksum
8 bits	6 bits	10 bits	n bits	24 bits
0xD3	Not defined	Message Length in bytes	Variable length in bytes	QualComm definition CRC-24Q

Comparison Chart of Wi-Fi, Bluetooth and Zigbee

	Wi-Fi (IEEE 802 11B)	Bluetooth (IEEE 802.15.1)	Zigbee (IEEE 802.15.4)
Radio	DSSS	FHSS	DSSS
Data Rate	11 Mbps	1 Mbps	250kbps
Nodes per Master	32	7	64000
Data Type	Video, audio, graphics, pictures, files	Audio, graphics, pictures, files	Small data packet
Range (M)	100	10	70
Extendibility	Roaming possible	No	Yes
Battery Type	AAA battery power	Li-ion batteries	Alkaline batteries
Slave enumeration latency	Upto 3s	Upto 10s	30 ms

Complexity	Complex	Very complex	Simple
Frequency	2.4 GHz	2.4 GHz	850-930 MHz
Security	Authentication Service Set ID (SSID)	64 bit, 128 bit	128 bit AES and application layer user defined
Operating Environment	32℉-104℉	32℉-122℉	32℉-131℉
Battery Life	Hours	1 Week	> 1 Year
Success Metrics	Speed, flexibility	Compatible, easy to use	Reliability, power,cost

CHAPTER : 12

Complete Program

```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most
acurate BPM math.
#include <LiquidCrystal.h>
#include <PulseSensorPlayground.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
//defining RX and TX for ESP8266
#define RX 13
#define TX 6

//user's friends' numbers
long numbers[] = {9426684641, 9428492817, 9909414845, 6359506664,
8000940593};
```

```
//setting up variables for GPS
int state = 0;
const int pin = 9;
float gpslat, gpslon;
TinyGPS gps;
SoftwareSerial sgps(4, 5);
SoftwareSerial sgsm(2, 3);

//setting up variables for the ESP8266 module and creating an object for
the same
String AP = "One plus 8T";       // AP NAME
String PASS = "hijjok"; // AP PASSWORD
String API = "IKVU9ZNVRD26QDK8";   // Write API KEY
String HOST = "api.thingspeak.com";
String PORT = "80";
String field = "field1";
int countTrueCommand;
int countTimeCommand;
boolean found = false;
int dangerBit = 0;
SoftwareSerial esp8266(RX,TX);

//initialising object for LCD display
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// constants won't change. They're used here to set pin numbers:
const int buttonPin = 8;     // the number of the pushbutton pin
//const int ledPin =  13;      // the number of the LED pin
int buzzerPin = 7;
int counter = 0;
signed short minutes, secondes;
char timeline[16];
const int PulseWire = 0; // PulseSensor PURPLE WIRE connected to ANALOG PIN
0
const int LED13 = 13; // The on-board Arduino LED, close to PIN 13.
```

```
// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status
int Threshold = 300; // Determine which Signal to "count as a beat" and
which to ignore.

PulseSensorPlayground pulseSensor; // Creates an instance of the
PulseSensorPlayground

void setup() {
  // initialize the buzzer pin as output:
  pinMode(buzzerPin, OUTPUT);
  //pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  // initialize the pushbutton pin as an input:

  Serial.begin(9600); // For Serial Monitor

//initialise GSM and GPS objects
  sgsm.begin(9600);
  sgps.begin(9600);

  //configuring the ESP8266 module by setting up to send values
  esp8266.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=""+ AP +"","+ PASS +""",20,"OK");

// Configure the PulseSensor object, by assigning our variables to it.
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13); //auto-magically blink Arduino's LED with
heartbeat.
pulseSensor.setThreshold(Threshold);

// Double-check the "pulseSensor" object was created and "began" seeing a
signal.
if (pulseSensor.begin()) {
Serial.println("We created a pulseSensor Object !"); //This prints one time
at Arduino power-up, or on Arduino reset.
}
```

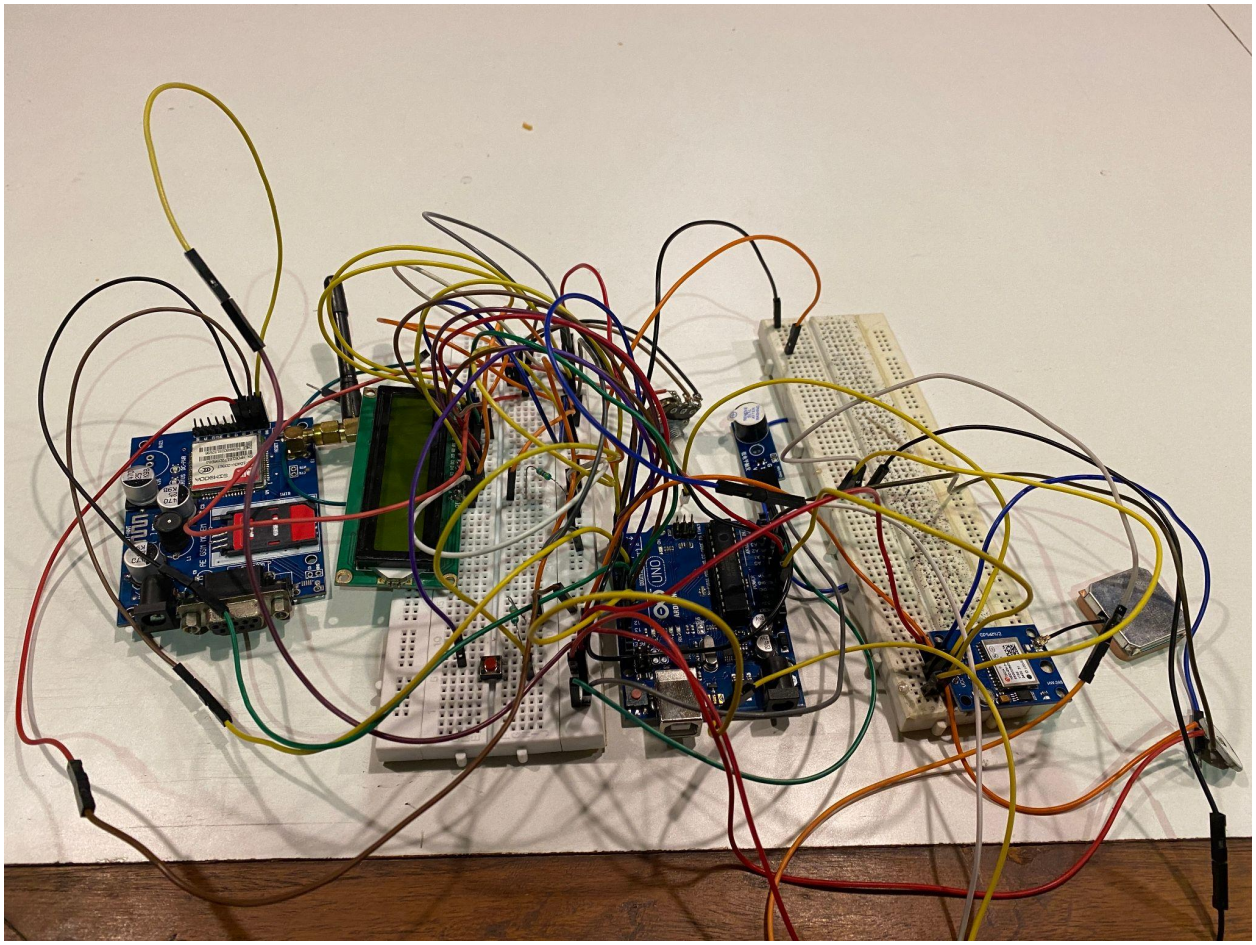
```
//initialises LCD
  lcd.begin(16, 2);
  lcd.print("Safety Device:");
}

void loop() {



 buttonState = digitalRead(buttonPin);
 int myBPM = pulseSensor.getBeatsPerMinute(); // Calls function on our
pulseSensor object that returns BPM as an "int".

// "myBPM" hold this BPM value now.
//if (pulseSensor.sawStartOfBeat()) { // Constantly test to see if "a beat
happened".
//Serial.println("♥ A HeartBeat Happened ! "); // If test is "true", print
a message "a heartbeat happened".
//Serial.print("BPM: "); // Print phrase "BPM: "
//Serial.println(myBPM);} // Print the value inside of myBPM.

  // check if user is unsafe
  if ((buttonState == LOW)||(myBPM>200)||(bpmSpike)) {
  digitalWrite (buzzerPin, LOW);
  delay(1000);
  counter ++;
  printf( "count =" + counter);
  if(counter > 4){
    dangerBit = 1;//to update the ThingSpeak LED
    sendNumbers(numbers);// to send SOS texts
    lcd.setCursor(0, 1);
  sprintf(timeline,"DANGER");// to update the danger status on LCD
  lcd.print(timeline);
  }
    // turn LED on:
    //digitalWrite(ledPin, HIGH);
}
  //user is safe
  if ((buttonState == HIGH)&&((myBPM<150)||(bpmSpike))  {
    // turn LED off:
    counter = 0;
```

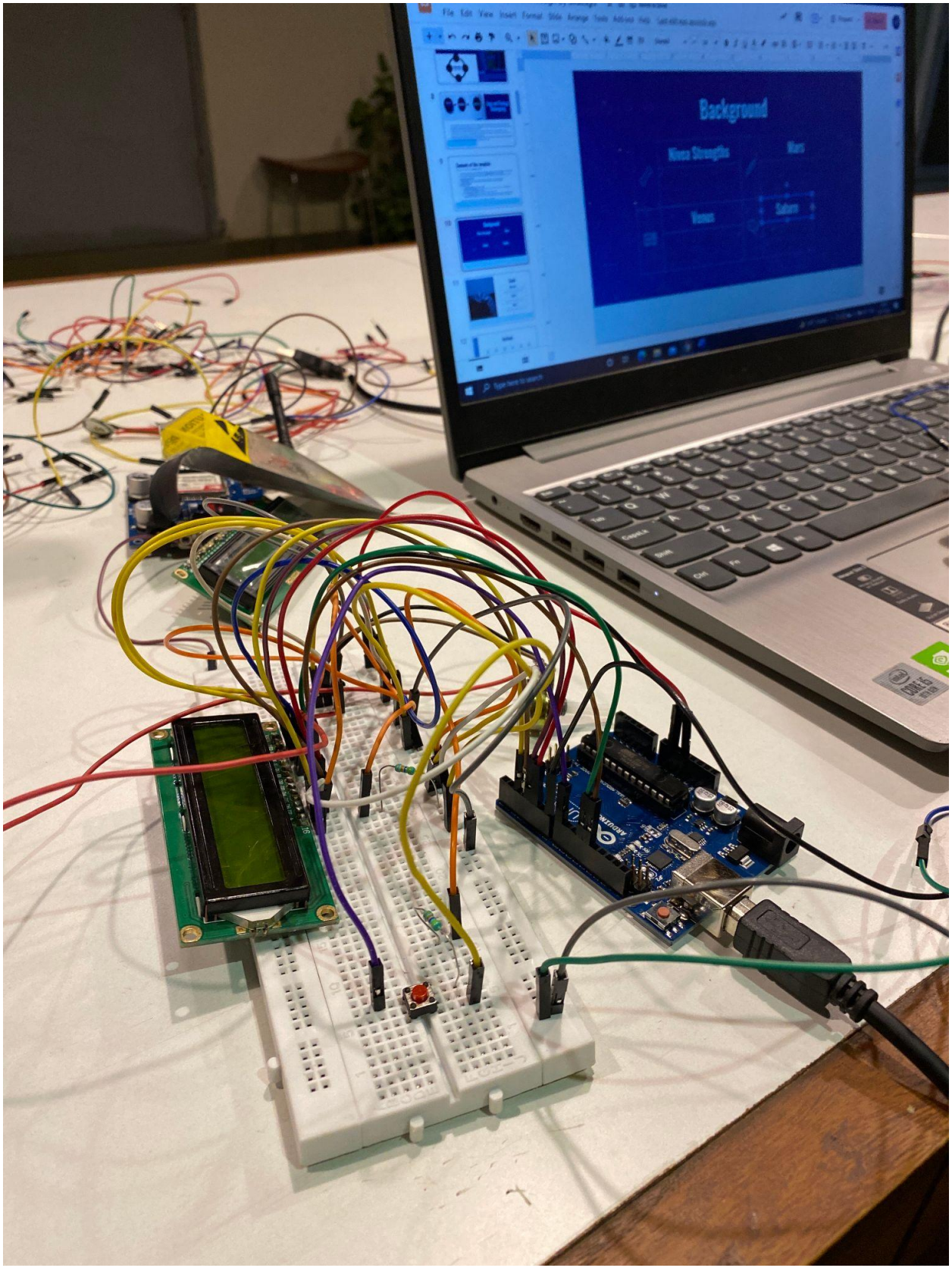
```
    digitalWrite (buzzerPin, HIGH);
    lcd.clear();
    lcd.print("Safety Device:");
    delay(1000);
  //  digitalWrite(ledPin, LOW);
  }
String getData = "GET /update?api_key="+ API +"&"+ "field1"
+"="+String(myBPM)+"&"+"field2"+"="+String(counter);
sendCommand("AT+CIPMUX=1",5,"OK");
sendCommand("AT+CIPSTART=0,"TCP",""+ HOST +"","+ PORT,15,"OK");
sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
esp8266.println(getData);
 delay(500);countTrueCommand++; // delay for sending data, net delay needed
of 1.5 seconds so this is deducting the 1s from before.
sendCommand("AT+CIPCLOSE=0",5,"OK");
}


void beep(unsigned char delayms) {                        // Created a
function for beep
  analogWrite(buzzerPin, 20);                                      // This
will set pin 11 to high
  delay(delayms);
// Giving a delay
  analogWrite(buzzerPin ,0);                                        // This
will set pin 11 to LOW
  delay(delayms);
// Giving a delay
}

boolean bpmSpike(int myBPM1){ // function to check spike in heartbeat
 int myBPM2 = pulseSensor.getBeatsPerMinute(); //to check the current
heartbeat against the previous one
 boolean check = false;
 if(myBPM2 - myBPM1 > 50) check = true;//if spike is greater than 50, its
possible the person has faced difficulty.
 return check;
}

void sendCommand(String command, int maxTime, char readReplay[]) {//
function to send command through ESP8266
```

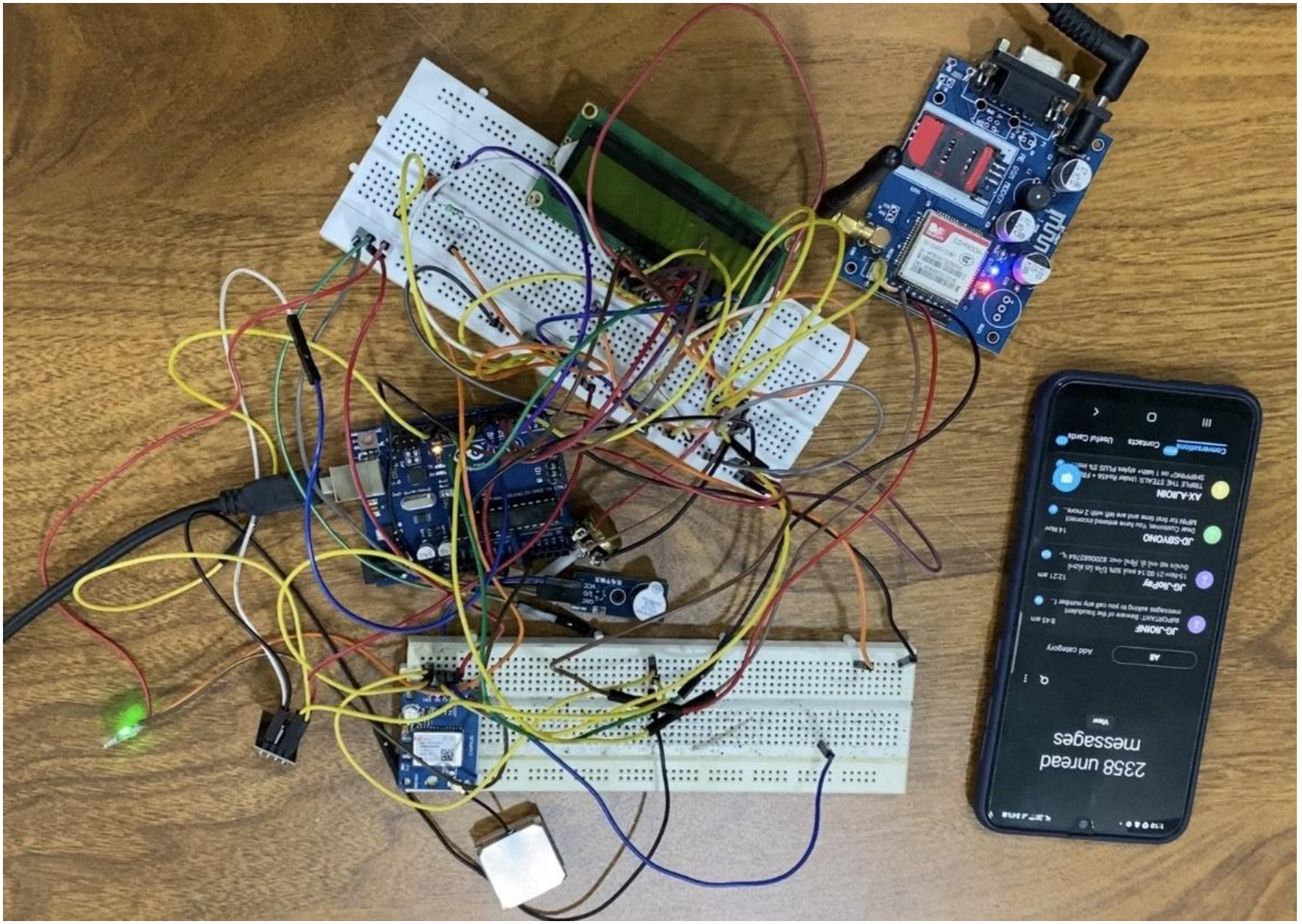
```
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  while(countTimeCommand < (maxTime*1))
  {
    esp8266.println(command);//at+cipsend
    if(esp8266.find(readReplay))//ok
    {
      found = true;
      break;
    }

    countTimeCommand++;
  }
  if(found == true)
  {
    Serial.println("OYI");
    countTrueCommand++;
    countTimeCommand = 0;
  }
  if(found == false)
  {
    Serial.println("Fail");
    countTrueCommand = 0;
    countTimeCommand = 0;
  }
  found = false;
 }

 void sendLocation(long phoneNumber){// function to send SOS messages to a
number
  sgps.listen();
  while (sgps.available())
  {
    int c = sgps.read();
    if (gps.encode(c))
    {
      gps.f_get_position(&gpslat, &gpslon);
    }
  }
```

```
    if (digitalRead(pin) == HIGH && state == 0) {
      sgsm.listen();
      sgsm.print("\r");
      delay(1000);
      sgsm.print("AT+CMGF=1\r");
      delay(1000);
      sgsm.print("AT+CMGS=\"+91" + String(phoneNumber) +  "\"\r");
      delay(1000);
      //The text of the message to be sent.
      sgsm.print("Latitude :");
      sgsm.println(gpslat, 6);
      sgsm.print("Longitude:");
      sgsm.println(gpslon, 6);
      delay(1000);
      sgsm.write(0x1A);
      delay(1000);
      state = 1;
    }
 if (digitalRead(pin) == LOW) {
      state = 0;
    }
}

void sendNumbers(long numbers[]){
 for(int i = 0; i++; i< numbers.length()){
   sendLocation(numbers[i]);}
}
```

Circuit Images

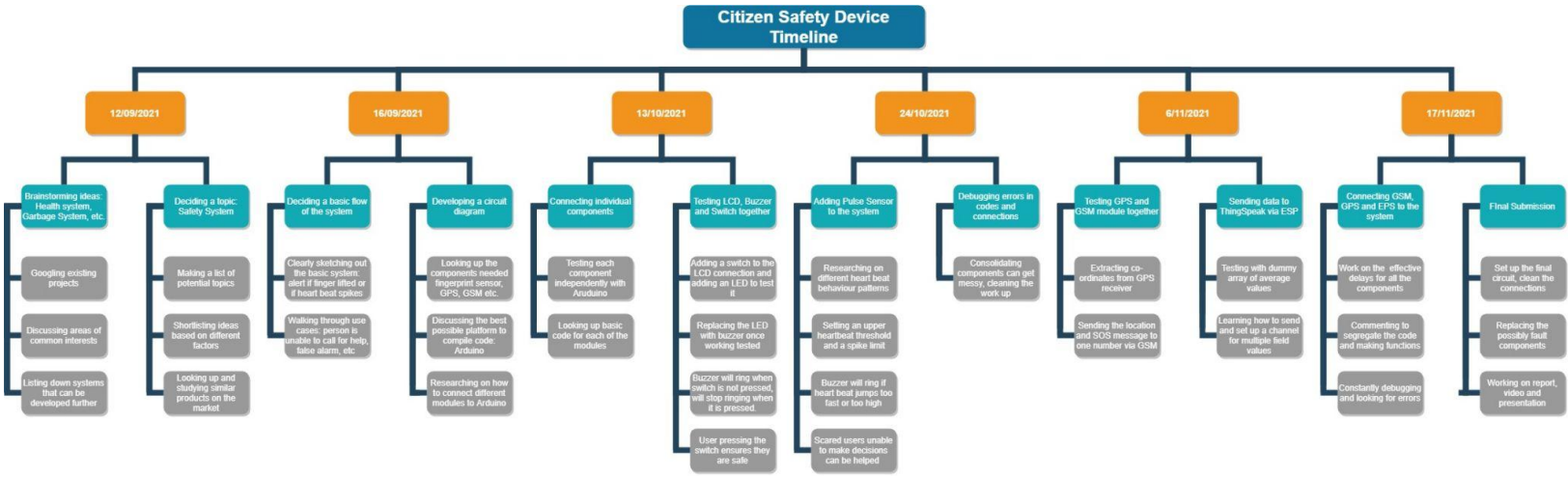



Background


2358 unread
messages

CHAPTER : 13

Summary



Potential Future Work

We have considered two scenarios to detect danger. First is using the Fingerprint as constant input and the other is the heart pulse monitor. If the pulse increases beyond a certain threshold danger would be detected and surroundings will be alerted about the same. But considering that there are multiple factors due to which the pulse rate would increase or decrease, for example the heart beat could increase if the person is running or jogging. Moreover sometimes the voice of Buzzer might scare the person because of which there could be a certain spike in pulse value. Thus using the heart pulse as a sign of danger would not be as effective as the fingerprint sensor. Hence we can work finding an optimum solution to detect danger using heart pulse monitor

We can work on adding another layer of security by sending the text messages or by calling various nearby police stations or helpline services to increase the safety in case of danger using the gsm and gps module.

CHAPTER : 14

References

- Muskan, T. Khandelwal, M. Khandelwal and P. S. Pandey, "Women Safety Device Designed Using IoT and Machine Learning," *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2018, pp. 1204-1210, doi: 10.1109/SmartWorld.2018.00210
- N. Viswanath, N. V. Pakyala and G. Muneeswari, "Smart foot device for women safety," *2016 IEEE Region 10 Symposium (TENSYMP)*, 2016, pp. 130-134, doi: 10.1109/TENCONSpring.2016.7519391.
- Prakash N., Udayakumar E., Kumareshan N., Gowrishankar R. (2021) GSM-Based Design and Implementation of Women Safety Device Using Internet of Things. In: Peter J., Fernandes S., Alavi A. (eds) Intelligence in Big Data Technologies—Beyond the Hype. Advances in Intelligent Systems and Computing, vol 1167. Springer, Singapore. https://doi.org/10.1007/978-981-15-5285-4_16
- Women Safety Device with GPS Tracking and Alerts Using Arduino (https://circuitdigest.com/microcontroller-projects/arduino-based-women-safety-device-for-emergency-alert-and-tracking)
- Womens Safety Device With GPS Tracking & Alerts (https://nevonprojects.com/womens-safety-device-with-gps-tracking-alerts/)

Appendix A

Datasheets

Fingerprint sensor

1. Concept

This device is one chip module with;

- fingerprint algorithm
- optical sensor

The major functions are the followings.

- High-accuracy and high-speed fingerprint identification technology
- Ultra-thin optical sensor
- 1:1 verification, 1:N identification
- downloading fingerprint image from the device
- Reading & writing fingerprint template(s) from/to the device
- Simple UART & USB communication protocol

Technical Specification

Item		Value
CPU		ARM Cortex M3 Core (Holtek HT32F2755)
Sensor		optical Sensor
Effective area of the Sensor		14 x 12.5(mm)
Image Size		202 x 258 Pixels
Resolution		450 dpi
The maximum number of fingerprints		200 fingerprints
Matching Mode		1:1, 1:N
The size of template		496 Bytes (template) + 2 Bytes (checksum)
Communication interface		UART, default baud rate = 9600bps after power on USB Ver1.1, Full speed
False Acceptance Rate (FAR)		< 0.001%
False Rejection Rate(FRR)		< 0.1%
Enrollment time		< 3 sec (3 fingerprints)
Identification time		< 1.0 sec (200 fingerprints)
Operating voltage		DC 3.3~6V
Operating current		< 130mA
Operating environment	Temperature	-20℃ ~ +60℃
	Humidity	20% ~ 80%

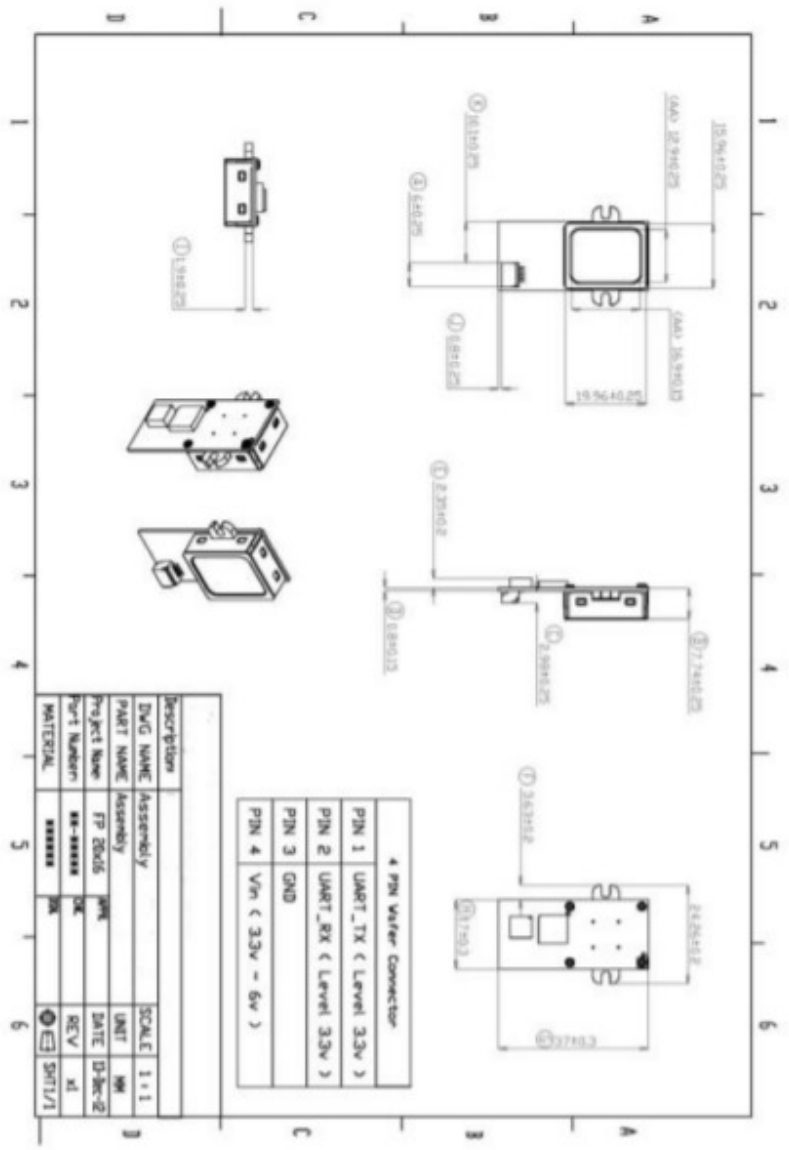
3. Protocol: Commands Summary

In a command packet *Command* can be one of below.

Number (HEX)	Alias	Description
01	*Open*	Initialization
02	*Close*	Termination
03	*UsbInternalCheck*	Check if the connected USB device is valid
04	*ChangeBaudrate*	Change UART baud rate
05	*SetIAPMode*	Enter IAP Mode In this mode, FW Upgrade is available
12	*CmosLed*	Control CMOS LED
20	*GetEnrollCount*	Get enrolled fingerprint count
21	*CheckEnrolled*	Check whether the specified ID is already enrolled
22	*EnrollStart*	Start an enrollment
23	*Enroll1*	Make 1st template for an enrollment
24	*Enroll2*	Make 2nd template for an enrollment
25	*Enroll3*	Make 3rd template for an enrollment, merge three templates into one template, save merged template to the database
26	*IsPressFinger*	Check if a finger is placed on the sensor
40	*DeleteID*	Delete the fingerprint with the specified ID
41	*DeleteAll*	Delete all fingerprints from the database
50	*Verify*	1:1 Verification of the capture fingerprint image with the specified ID
51	*Identify*	1:N Identification of the capture fingerprint image with the database
52	*VerifyTemplate*	1:1 Verification of a fingerprint template with the specified ID
53	*IdentifyTemplate*	1:N Identification of a fingerprint template with the database
60	*CaptureFinger*	Capture a fingerprint image(256x256) from the sensor
61	*MakeTemplate*	Make template for transmission

Number (HEX)	Alias	Description
62	*GetImage*	Download the captured fingerprint image(256x256)
63	*GetRawImage*	Capture & Download raw fingerprint image(320x240)
70	*GetTemplate*	Download the template of the specified ID
71	*SetTemplate*	Upload the template of the specified ID
72	*GetDatabaseStart*	Start database download, **obsolete**
73	*GetDatabaseEnd*	End database download, **obsolete**
80	*UpgradeFirmware*	Not supported
81	*UpgradeISOCDImage*	Not supported
30	*Ack*	Acknowledge.
31	*Nack*	Non-acknowledge.

8. Mechanical Dimensions



Pulse Sensor

General Description	Features
The Pulse Sensor is the original low-cost optical heart rate sensor (PPG) for Arduino and other microcontrollers. It's designed and made by World Famous Electronics, who actively maintain extensive example projects and code at: www.pulsesensor.com	• Includes Kit accessories for high-quality sensor readings • Designed for Plug and Play • Small size and embeddable into wearables • Works with any MCU with an ADC • Works with 3 Volts or 5 Volts • Well-documented Arduino library

Absolute Maximum Ratings	Min	Typ	Max	Unit
Operating Temperature Range	-40		+85	°C
Input Voltage Range	3		5.5	V
Output Voltage Range	0.3	Vdd/2	Vdd	V
Supply Current	3		4	mA

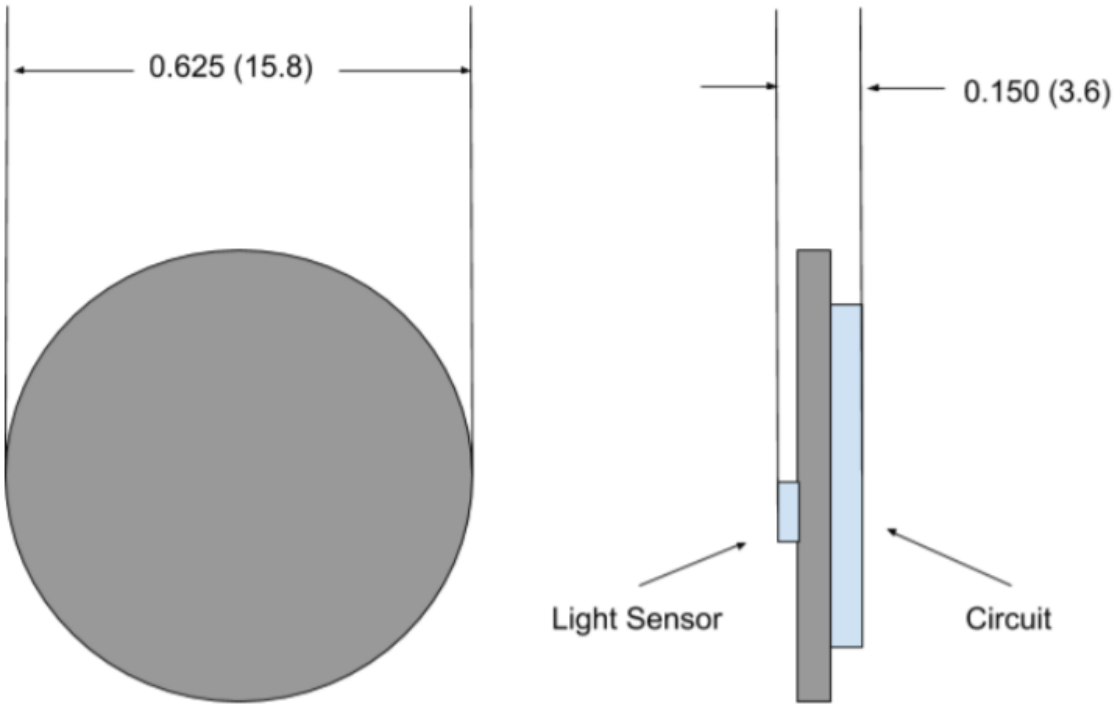
Pin Configuration

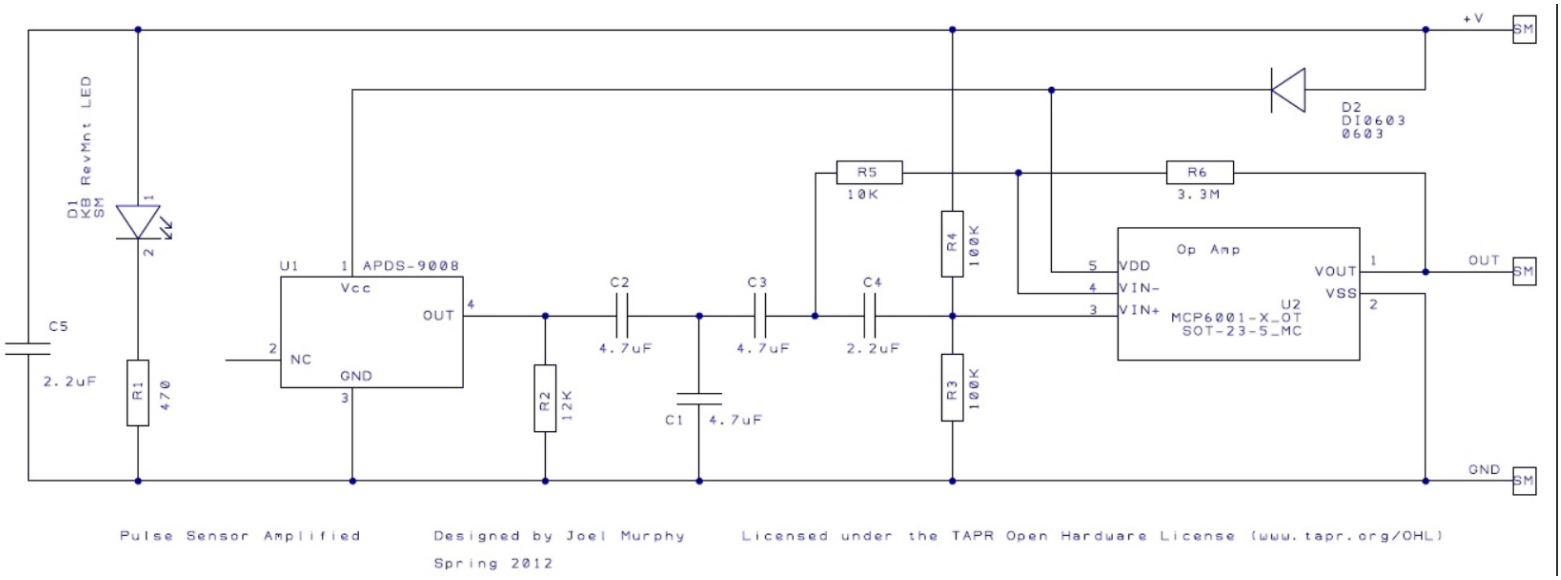
Pin Number	Pin Name	Wire Colour	Description
1	Ground	Black	Connected to the ground of the system
2	Vcc	Red	Connect to +5V or +3.3V supply voltage
3	Signal	Purple	Pulsating output signal.

Pulse Sensor Kit Contents

Pulse Sensor Optical Heart Rate Monitor

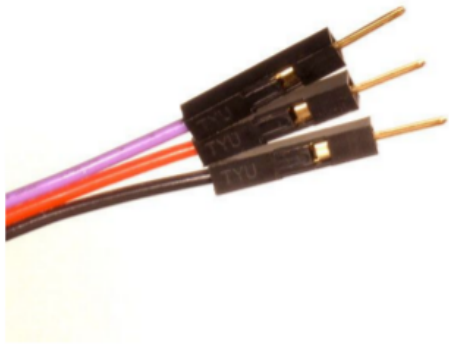
Physical Dimensions PCB inch(mm)

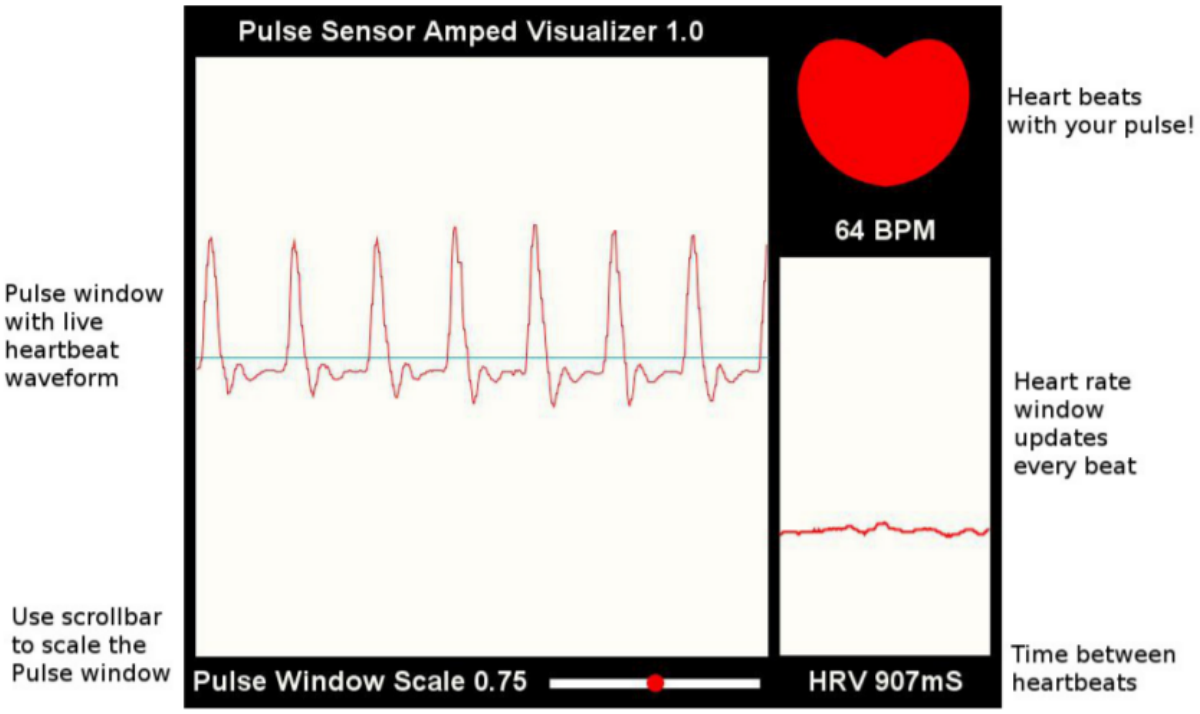




Cable Specs

- **Length 610 mm (24 inches)**
- **26 Gauge**
- **PVC Insulation, Ribbon Style**
- **Male Header Termination**
  - **Black Wire = GND**
  - **Red Wire = Vdd**
  - **Purple Wire = Pulse Signal**




Pulse Sensor Amped Visualizer 1.0
64 BPM
Pulse Window Scale 0.75
HRV 907mS
Heart beats with your pulse!
Pulse window with live heartbeat waveform
Heart rate window updates every beat
Use scrollbar to scale the Pulse window
Time between heartbeats

Buzzer

Buzzer

pro-SIGNAL

Features

- Black in colour
- With internal drive circuit
- Sealed structure
- Wave solderable and washable
- Housing material: Noryl

RoHS Compliant

Applications

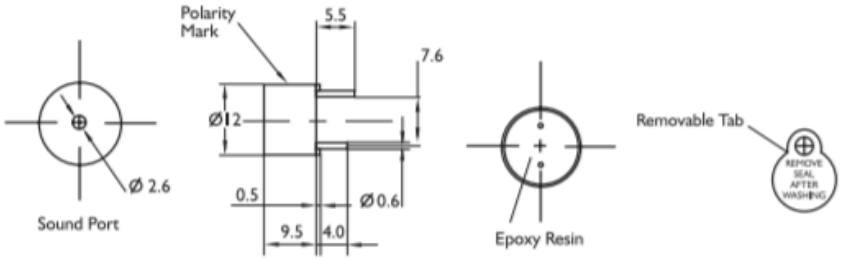
- Computer and peripherals
- Communications equipment
- Portable equipment
- Automobile electronics
- POS system
- Electronic cash register

Specifications:

Rated Voltage	: 6V DC
Operating Voltage	: 4 to 8V DC
Rated Current*	: ≤30mA
Sound Output at 10cm*	: ≥85dB
Resonant Frequency	: 2300 ±300Hz
Tone	: Continuous
Operating Temperature	: -25°C to +80°C
Storage Temperature	: -30°C to +85°C
Weight	: 2g

*Value applying at rated voltage (DC)

Diagram



Dimensions : Millimetres
Tolerance : ±0.5mm

Part Number Table

Description	Part Number
Buzzer, Electromech, 6V DC	ABI-009-RC

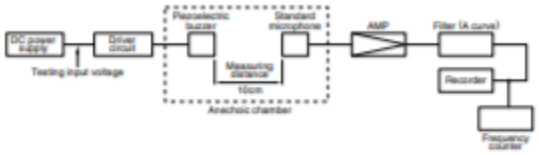
FEATURES

- The PS series are high-performance buzzers that employ unimorph piezoelectric elements and are designed for easy incorporation into various circuits.
- They feature extremely low power consumption in comparison to electromagnetic units.
- Because these buzzers are designed for external excitation, the same part can serve as both a musical tone oscillator and a buzzer.
- They can be used with automated inserters. Moisture-resistant models are also available.
- The lead wire type(PS1550L40N) with both-sided adhesive tape installed easily is prepared.

APPLICATIONS

Electric ranges, washing machines, computer terminals, various devices that require speech synthesis output.

SOUND MEASURING METHOD



SPECIFICATIONS AND CHARACTERISTICS

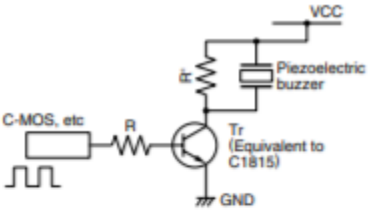
Type	Part No.	External dimensions			Characteristics		
		Outer diameter (mm)	Height (mm)	Pitch (mm)	Sound pressure (dB(A)/10cm)	Frequency (kHz)	Input voltage (Vo-p)[Rectangular wave]
PS12 Type	PS1240P02BT	ø12.2	6.5	5	70 min.	4	3
	PS1240P02CT3	ø12.2	3.5	5	60 min.	4	3
PS14 Type	PS1440P02BT	ø14	8	5	75 min.	4	3
	PS1420P02CT	ø14	11	5	70 min.	2	5
PS17 Type	PS1720P02	ø17	8	10	70 min.	2	3
	PS1740P02E	ø17	7.5	10	75 min.	4	3
	PS1740P02CE	ø17	4.6	10	60 min.	4	3
PS19 Type	PS1927P02	ø19	10.5 [excluding terminal]	20	90 min.	2.7	10
	PS1920P02	ø19	10.5 [excluding terminal]	20	80 min.	2	10
Others	PS1550L40N	ø15	1.6	—	Depend on the installation condition		

Type	Part No.	Applications	Features
PS12 Type	PS1240P02BT	For warning and alarm sounds of home appliances(air conditioners, refrigerators, fan forced heaters, cordless telephones, etc.)	• Compact • Automatic mountable • 12.7mm pitch radial taping
	PS1240P02CT3		• Thin type • Automatic mountable • 12.7mm pitch radial taping
PS14 Type	PS1440P02BT		• High sound pressure • Automatic mountable • 15mm pitch radial taping
	PS1420P02CT		• Low frequency tone • Automatic mountable • 15mm pitch radial taping
PS17 Type	PS1720P02		• Low frequency tone • High sound pressure
	PS1740P02E		• High sound pressure
	PS1740P02CE		• Thin type
PS19 Type	PS1927P02	For potted circuit (washing machines, drying machines, hot water supply systems, etc.)	• High sound pressure • Water-proof processing element
	PS1920P02		• Low frequency tone • Water-proof processing element
Others	PS1550L40N	Digital camera	• Compact, Thin type • Fix in both-sided adhesive tape

PRECAUTIONS FOR USE

- Do not apply DC bias to the piezoelectric buzzer; otherwise insulation resistance may become low and affect the performance.
- Do not supply any voltage higher than applicable to the piezoelectric buzzer.
- Do not use the piezoelectric buzzer outdoors. It is designed for indoor use. If the piezoelectric buzzer has to be used outdoors, provide it with waterproofing measures; it will not operate normally if subjected to moisture.
- Do not wash the piezoelectric buzzer with solvent or allow gas to enter it while washing; any solvent that enters it may stay inside a long time and damage it.
- A piezoelectric ceramic material of approximately 100μm thick is used in the sound generator of the buzzer. Do not press the sound generator through the sound release hole otherwise the ceramic material may break. Do not stack the piezoelectric buzzers without packing.
- Do not apply any mechanical force to the piezoelectric buzzer; otherwise the case may deform and result in improper operation.
- Do not place any shielding material or the like just in front of the sound release hole of the buzzer; otherwise the sound pressure may vary and result in unstable buzzer operation. Make sure that the buzzer is not affected by a standing wave or the like.
- Be sure to solder the buzzer terminal at 350°C max.(80W max.)(soldering iron trip) within 5 seconds using a solder containing silver.
- Avoid using the piezoelectric buzzer for a long time where any corrosive gas (H_2S, etc.) exists; otherwise the parts or sound generator may corroded and result in improper operation.
- Be careful not to drop the piezoelectric buzzer.

RECOMMENDED OPERATING CIRCUIT EXAMPLE



* Resistor to do charging and discharging to a piezoelectric element (Value of about 1kΩ is good efficiency).

Appendix B

Programming Review

Points to compare	C	C++	Java	Python
Paradigms	Procedural	Object Oriented	Object Oriented	Multi-Paradigm
Platform Dependency	Dependent	Dependent	Independent	Independent
Keywords	32	63	50 defined (goto, const unusable)	33
Preprocessor directives	Supported (#include, #define)	Supported (#include, #define)	Not Supported	Not Supported
Header files	Supported	Supported	Use Packages (import)	Use Packages (import)

Inheritance	No Inheritance	Supported	Multiple Inheritance Not Supported	Supported
Overloading	No Overloading	Supported	Operator Overloading Not Supported	Operator Overloading Not Supported
Pointers	Supported	Supported	No Pointers	No Pointers
Multi-threading and Interfaces	Not Supported	Not Supported	Supported	Supported
Database Connectivity	Not Supported	Not Supported	Supported	Supported
Code Translation	Compiled	Compiled	Interpreted	Interpreted
Exception Handling	No Exception handling	Supported	Supported	Supported

Appendix C

Trouble-shooting

1. **Ublox NEO-6M GPS Module**

   Problem: Not able to get any data from Neo-6M (GY-GPS6MV2) connected to Arduino

   Connections: GND - GND, TX - D3, RX - D2, VCC - 5V

   Code:

```
#include <SoftwareSerial.h>


// choose pins that connect to UBlox Neo 6m
int RXPin = 2;
int TXPin = 3;
int GPSBaud = 4800;

SoftwareSerial gpsSerial(RXPin, TXPin);


void setup() {
  // put your setup code here, to run once:
  // define pin modes for tx, rx:
  pinMode(RXPin, INPUT);
  pinMode(TXPin, OUTPUT);
  Serial.begin(9600);
  gpsSerial.begin(9600);
  Serial.write("Welcome to the GPS Show");
}

void loop() {
  // put your main code here, to run repeatedly:
  while(gpsSerial.available() > 0) {
    Serial.write(gpsSerial.read());
  }
}
```

Debugging & Troubleshooting:

After referring the data sheet (http://www.u-blox.com/images/downloads/Product_Docs/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf), we found that the baud rate should be set to 9600 for the module to work and GSM module's operating voltage is between 2.7V-3.6V, so changed the supply voltage to 3.3V.

VCC - 3.3V

int GPSBaud = 9600;

2. AE GSM MODEM SIM900A

Problem: The program is not loaded successfully.

Connections old: GND - GND, TX - 0, RX - 1, VCC - 5V, GSM Module Vol - 12V

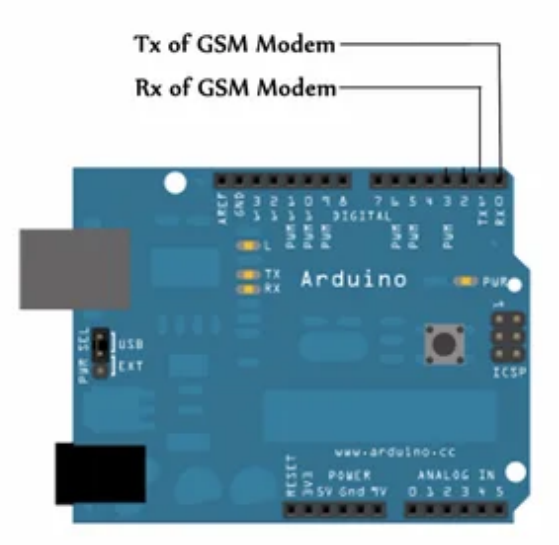
Connections new: GND - GND, TX - 9, RX - 10, VCC - 5V, GSM Module Vol - 12V

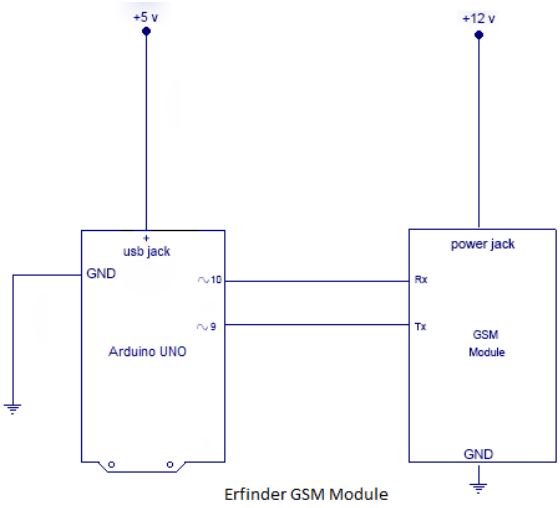
Debugging & Troubleshooting:

There are two ways of connecting a GSM module to an arduino. In any case, the communication between Arduino and GSM modules is serial. So we used serial pins of Arduino (Rx and Tx). So with this method, we connected the Tx pin of the GSM module to the Rx pin of the Arduino and the Rx pin of the GSM module to the Tx pin of Arduino.

Then, we loaded the programs to communicate with the gsm module to make it work.

The problem with this connection came while programming. Arduino uses serial ports to load programs from the Arduino IDE. If these pins are used in wiring, the program will not be loaded successfully to Arduino. So we had to disconnect wiring in Rx and Tx each time we ran the program. Once the program is loaded successfully, we can reconnect these pins and have the system working! To avoid this difficulty, We came up with an alternate method in which two digital pins of the arduino are used for serial communication. We need to select two PWM enabled pins of the arduino for this method. So we chose pins 9 and 10 (which are PWM enabled pins). This method is made possible with theSoftwareSerial Library of Arduino. SoftwareSerial is a library of Arduino which enables serial data communication through other digital pins of Arduino. The library replicates hardware functions and handles the task of serial communication.





Erfinder GSM Module

3. Heart Rate Pulse Sensor

Problem: Sensor not working properly

Debugging & Troubleshooting: According to the datasheet given, the operating current should be 3-4mA and it seemed that the current it received was more for some reason due to which it got short circuited.

Appendix D

Real Life Mounting of the system

Nowadays we see many monitoring device systems integrated in a smartwatch or in a smartphone or in any wearable technology. It's advantageous as we dont need to carry the whole circuit everywhere we go. Therefore, we will place this system in a wearable device for the feasibility and to properly serve its purpose of monitoring danger.

Generally, the system is encapsulated with a waterproof and weather resistant shielding so there won't be any problem unless the phone/watch is damaged. Thus they should be used carefully. Also, these days we find coating which helps in maintaining the temperature so that sensors can work properly. And the system will be using the power generated from the phone's/watch's battery.

Technology has never been free of the dualistic nature of reality—good or bad, right or wrong. Wearable devices are no different. While they do make life easier, they also pose a new security threat to personal privacy.

When users log in their personal data like phone numbers on websites, they actually have no idea about the developers behind the app or the companies selling them. Even if these are innocent providers, with no malicious intent, users still do not know if their personal data is stored on servers that have sufficient network security to ward off a hacker attack.

If personal information falls into the wrong hands, it can lead to a wide spectrum of cyber-abuse like identity theft, or targeted advertising. And also, surreptitiously tracking the user's online movements.

Since wearable smart devices can store and transmit information in seconds, users need to be mindful of how this information can be a goldmine for hackers and online criminals who intend to spring a cyber-based attack.

With so many features and functionalities, the battery management system (BMS) of a wearable takes more time to charge. Despite improvement in batteries' capacity in the last decade, the stored charge drains rapidly after a limited period of time. This decreases a wearable device's lifecycle and, as a result, the battery needs to be regularly replaced or changed to a rechargeable battery, which increases maintenance and cost. Also, a BMS that lasts for only four to six years is not helpful. Battery lifetime is a critical issue in wearable devices.

Small size factor is another critical issue for wearable devices. A battery's complex internal structure can significantly increase the size and cost of the devices, making it inconvenient for people to wear them.

Some of the new ways to achieve efficient power management are:

1. Bluetooth LE 5, Wi-Fi, or RFID. They consume low power (in microwatts) and offer effective data transfer facilities on an SoC.

2. Advanced CMOS technology. It has enabled the evolution of low-power circuits and systems. It allows the device to operate at a sub-microwatt power level.

3. Energy harvesting sources. With the shrinking transistor size, there is a need for efficient management of power sources. This means we should consider generating energy from thermal, vibration, or solar sources. However, energy harvesting brings some design challenges with itself, since harvested energy has voltage swings and depends on the availability and size of the harvester.

Appendix E

Real Life Scenarios

1. All the components are wired and are not composed into a neat system, therefore, it's only usable when it is converted into a compact, wearable device. The most important electronic component in wearable devices is the microcontroller. As these MCUs need to be small and at the same time perform more functions, integration becomes another important factor.

   But microprocessor speeds aren't always up to snuff and power concerns are paramount, because without power to bring them to life, electronics are just dead metal. Now, with better batteries, ever-increasing processor speed, ceaseless Internet connectivity and clever software programming, wearable technology seems possible.

2. Size- These devices must be small so that they can easily fit on to a wearable. Nevertheless, at the same time they must integrate more features in the same space. Technologies such as System-on-Chip (SoC) and chip scale packages (CSP) help to shrink the size. For example, Cypress offers PSoC (Programmable System-on-Chip) devices in multiple packaging options including WLCSP.
3. Power Consumption- Since wearable devices are battery powered, reducing the power consumption of these devices poses unique challenges. Wearable devices, unlike other mobile devices, are required to be always on and always connected because most of these are

monitoring devices. For example, a smart watch needs to always show the time and be connected to a mobile phone through a wireless link such as Bluetooth in order to receive alerts; a pedometer is required to continuously count the steps and report it to a mobile phone app; and similarly a heart rate monitor needs to be always monitoring and reporting. Yet battery capacity is inherently limited due to the requirement to reduce the overall size.

Therefore these devices need to operate at ultra-low power to conserve the battery life. This requirement drives special needs in MCU and firmware algorithms. 32-bit ARM architecture is a popular CPU technology for wearable devices as it provides best performance and energy efficiency. Also wireless technologies such as ANT+, Bluetooth Low Energy (BLE) are designed to consume low power.