Visvesvaraya Technological University, Belagavi.

PROJECT REPORT

on

"IOT BASED SMART HEALTH MONITORING SYSTEM"

Project Report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

For the academic year 2019-20

Submitted by

USN	Name
1CR16EC409	HARISH S
1CR16EC437	SRIKANTH NG
1CR16EC442	SUSHMA G
1CR16EC444	VIKAS U

Under the guidance of

Internal Guide

Mr. KIRSHNA TEJA

Assistant Professor

Department of ECE

CMRIT,Bangalore

Department of Electronics and Communication Engineering

CMR Institute of Technology, Bengaluru – 560 037

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CELEBRATING 25 YEARS
CMR INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to Certify that the dissertation work **"IOT BASED SMART HEALTH MONITORING SYSTEM"** carried out by HARISH S, SRIKANTH NG, SUSHMA G,VIKAS U, USN:1CR16EC409, 1CR16EC437, 1CR16EC442, 1CR16EC444 , bonafide students of **CMRIT** in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the academic year **2019-20**. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
Mr. Krishna Teja	Dr.R. Elumalai	Dr. Sanjay Jain
Assistant Professor,	Head of the Department,	Principal,
Department of ECE,	Department of ECE,	CMRIT,
CMRIT, Bangalore.	CMRIT, Bangalore.	Bangalore.

External Viva

Name of Examiners	Signature & date
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose consistent guidance and encouragement crowned our efforts with success.

We consider it as our privilege to express the gratitude to all those who guided in the completion of the project.

We express my gratitude to Principal, **Dr. Sanjay Jain,** for having provided me the golden opportunity to undertake this project work in their esteemed organization.

We sincerely thank **Dr. R. Elumalai,** Head**,** Department of Electronics and Communication Engineering, CMR Institute of Technology for the immense support given to me.

We express my gratitude to our project guide Mr. Krishna Teja, Assistant Professor, for their support, guidance and suggestions throughout the project work.

Last but not the least, heartful thanks to our parents and friends for their support.

Above all, We thank the Lord Almighty for His grace on us to succeed in this endeavor.

Table of Contents

CHAPTER 1 **ERROR! BOOKMARK NOT DEFINED.**

INTRODUCTION **ERROR! BOOKMARK NOT DEFINED.**

1.1 Overview **Error! Bookmark not defined.**

CHAPTER 2 3

LITERATURE SURVEY 3

2.1 Survey 3

CHAPTER 3 **ERROR! BOOKMARK NOT DEFINED.**

METHODOLOGY 7

3.1 Architecture Blockdiagram 7

3.1.1System Architecture 7

3.2 PPG Concept 9

3.3 System Requirements 13

CHAPTER 4 14

HARDWARE 14

4.1ESP32 Microcontroller 14

4.1.1 Pin Mapping 15

4.1.2Flash layout 15

4.1.3Device Summary 17

4.2 GPS Module 20

4.2.1 Segments of GPS Communication 21

4.2.2 Working of GPS 21

4.2.3 GPS Receiver Technology 24

4.2.4 General Applications of GPS 26

4.2.5 GPS Receiver 30

4.3 Pulse Sensor 32

4.3.1 Pulse Sensor Anatomy 34

4.3.2Preparing the pulse sensor 35

4.3.3 Running the pulse sensor code 36

4.3.4 Trouble shooting 42

4.4 DHT11 sensor 43

4.4.1 Pin Identification and Configuration 43

4.4.2 DHT11 Specifications 44

4.4.3 Applications 45

4.5 Buzzer 46

4.5.1 Buzzer features and Specifications 47

4.5.2 Buzzer Applications 47

CHAPTER 5 48

SOFTWARE 48

5.1 Google Firebase cloud 48

5.1.1 Introduction to firebase 48

5.1.2 Firebase use case 49

5.1.3 Application Architecture 49

5.2 MIT App Inventor 54

CHAPTER 6 56

RESULTS AND DISCUSSION 56

CHAPTER 7 60

APPLICATIONS AND ADVANTAGES 60

7.1 Application 60

7.2 Advantage 60

CHAPTER 7	61
CONCLUSIONS	61
REFERENCES	62

Chapter 1

INTRODUCTION

1.1 Overview

IOT was first proposed by Kevin Ashton in 1999 . It is a communication network in which physical objects are interconnected with each other or with larger systems. This network collects billions of data from the very different devices we use in everyday life and transforms them into usable information . Today, there are about 20 billion devices in the world that interact with each other, and by 2025 it is estimated to go up to 75 billion devices . This shows that in the coming years cities that we live with IOT will become smart cities that will keep pace with the more paced and planned life . This transformation will offer us many opportunities to make our life easier . One of these important opportunities is the e-health services that are closely related to all of us .

IoT applications in the health sector are increasing day by day . People living in rural areas cannot benefit from preventive health services due to lack of infrastructure. As a result, deaths occur very early in these regions. . In addition, with the rapid aging of the world population, the needs of the elderly for life support are increasing with the change of family structure . In addition to chronic heart disease, there is a high probability that the patient will lose his or her life as a result of excessive fatigue of the heart during sleep at night, especially in Chronic Obstructive Pulmonary Disease (COPD) and Obstructive Sleep Apnea Syndrome (OSAS). Devices that provide continuous monitoring of these patients are very expensive and sensitive and require trained personnel to use them . It is possible for such patients to be followed up continuously with wearable health devices while maintaining their daily lives in the social environment. These wearable devices continuously measure the patient's heart values and, when a symptom of a heart attack has occurred, may send information about the patient's health condition to the family members and the doctor . HR and HRV are used primarily as a diagnostic tool for heart and non-cardiac diseases such as heart failure, aging, Parkinson's disease, diabetes . IoT is a new reality that completely changes our daily life. It is also a way to revolutionize modern health care by providing more personalized and preventive care.

Thanks to IOT technology, mutual information sharing among various smart devices has been facilitated anywhere in the world. In this environment, studies on smart health services, which can provide remote diagnosis of the disease, are also accelerated. . Thanks to low cost, low power consumption and high performance, devices that can collect patient heart data can be sent to the patient's family or doctor by smart phone applications . Continuous monitoring of a person's health through wearable biomedical devices is now possible with many wearable health kits. However, real-time analyzes and estimates, warnings and alarms on health hazards are not adequately addressed in these devices .

In this study, a wearable device is designed to measure vital values such as HR, HRV, GPS and CT, which directly concern heart health. The pulse sensor on the device and the heart related data from the patient's fingertip are analyzed with the ESP32 controller. The results of this analysis are transferred to the patient's mobile phone via Bluetooth connection.

Thanks to the "Smart Health" interface created with the MIT app inventor, the data transmitted to the mobile phone is displayed on the screen in real time. When the patient's vital parameters reach critical levels, an audible-visual alert is sent to the patient and family members with the Android-based application. Concurrently, this data and the patient's position information are sent to the patient's family members and the her/his doctor as e-mail and twitter notification. The main purpose of the device is to increase the chances of survival by providing medical assistance to the patient within the first few hours in case of a possible heart attack.

Materials and Methods

Parameters such as HR, HRV and BT, which are considered to be important health status indicators, have a tremendous diagnostic value. Until recently, continuous monitoring of these physiological parameters has been possible only in the hospital environment, and nowadays with the development of wearable technology, these parameters can be monitored accurately, continuously and in real time h.

Chapter 2

LITERATURE SURVEY

2.1 Survey

Design and Implementation of a Wearable Sensor Network System for IoT-Connected Safety and Health Applications Fan Wu ;Taiyang Wu ; Mehmet RasitYuce IEEE 2019

This paper presents a wearable sensor network system for Internet of Things (IoT) connected safety and health applications. Safety and health of workers are important for industrial workplace; therefore, an IoT network system which can monitor both environmental and physiological can greatly improve the safety in the workplace. The proposed network system incorporates multiple wearable sensors to monitor environmental and physiological parameters. The wearable sensors on different subjects can communicate with each other and transmit the data to a gateway via a LoRa network which forms a heterogeneous IoT platform with Bluetooth-based medical signal sensing network. Once harmful environments are detected and, the sensor node will provide an effective notification and warning mechanism for the users. A smart IoT gateway is implemented to provide data processing, local web server and cloud connection. After the gateway receives the data from wearable sensors, it will forward the data to an IoT cloud for further data storage, processing and visualization.

Smart Health Monitoring System through IOT S. Ananth ; P. Sathya ; P. Madhan Mohan IEEE 2019

IOT is one of the booming field in forthcoming years and plays a major role in the field of health care. IOT helps in connecting the people by empowering their health and wealth in a smart way through wearable gadgets. Recent improvements in wireless sensor networks have created a new trend in internet of things. Smart health is important application in Internet of things. Patients with abnormal health conditions can be quickly monitored through smart health care system and provide a rapid solution for the patients. This type of solutions can be encountered through wearable gadgets that continuously monitor the activity and condition of the patient in a predictable method. The main aim of this work is to provide an extensive research in capturing the sensor data's, analyzing the data and providing a feedback to patients based on different health parameters.

Smart and Wearable ECG monitoring system as a Point of Care (POC) device AbhinayVishwanatham ; Narendra Ch. ; S.R. Abhishek ; Ramakrishna Chaitanya R. ; Siva Sankara Sai Sanaga IEEE 2019

Cardiovascular diseases are the major cause of mortality rate in India with more than 50 % of the patients coming from a rural background. IoT plays a crucial role in developing point of care devices which cater to the healthcare demands of the rising rural populace. ECG monitoring system plays a key role as a diagnostic tool for cardiac abnormalities. Thus it becomes important to develop a portable point of care (POC) device at an affordable price so as to monitor the patients' cardiac health without interfering in their daily schedule. We developed an end to end health care workflow comprising three important modules. Firstly, a Bluetooth Low Energy (BLE) enabled portable 5-lead ECG monitor system with small form factor. Next, a smart phone based android application, which receives, plots and analyzes the data sent from ECG device. Finally, a remote server where the patient data and analysis reports are stored for future reference of a professional medical practitioner. The device was tested using real time data from a rural hospital and further the obtained ECG signatures were compared with standard GE and SIEMENS ECG machines. The results were validated by a cardiologist of a super specialty hospital.

An IoT-based wearable system using accelerometers and machine learning for fetal movement monitoring Xin Zhao ;Xianyi Zeng ; LudovicKoehl ; Guillaume Tartare ; Julien de Jonckheere ; Kehui Song IEEE 2019

Fetal movement is an essential index of fetal well-being. However, existing technologies in clinical applications cannot provide an effective, long-term and easily accessible way for fetal movement monitoring. This paper presents a wearable system using accelerometers and machine learning for automatic monitoring of fetal movement. To realize the concept of the e-health home care, Internet of Things (IoT) is applied on the system to connect all the terminal monitoring units to a control center. The whole system mainly consists of two parts: the local monitoring unit and the remote health evaluation unit, respectively. The local monitoring unit includes: (1) a garment integrating accelerometers for data acquisition and an embedded system for signal processing and machine learning, (2) an Android-based local monitoring platform used for visualization of statistics on fetal health status based on information received from the garment via Bluetooth. The local monitoring unit is also designed to alarm the mother if fetal movements reduce significantly or if the intensity is weakened. The remote health evaluation unit consists of a cloud computing platform and an expert system. Connecting each terminal local monitoring unit via the Internet, the remote health evaluation unit allows an expert to access the information at distance for further medical supervision, make advanced diagnosis as well as take actions in case of emergency. The cloud computing platform is also designed for convergence of medical data from each individual patient to build a general medical database for scientific research. The proposed local fetal movement monitoring unit along with the remote health evaluation platform could be regarded as a good example of IoT application in the field of smart textile and home healthcare.

Smart secure system for human health monitoring R Nikhil Nair ; K. A. Kiran IEEE 2019

As the population is expanding around the world, there is an immense need to provide appropriate medical assistance to them. To meet this need, sensors are one such revolution which can be used to monitor a large population in a simple manner. The main idea of this paper is to provide a secure way of monitoring a patient0s health [very confidential data] in a real time process. The system uses MEMS based sensors which are wearable. These sensors are used to diagnose or monitor the patient continuously and thereby the data is made to correspond with a raspberry pi which is then connected to the cloud. According to the data stored in the cloud, the patient0s state is alerted to family doctor, hospital or an emergency ambulance. The patient0s (VIPs or any of their relatives) data which are stored in the cloud have to be secured. In this system, the data from pulse rate sensor and ECG sensor is collected to Arduino UNO R3 development board and then transmitted to Raspberry Pi which acts as a server. Then the data is sent to the MySQL database encrypted using AES cryptography written in PHP language.

Pervasive Healthcare Monitoring System Mario Nava Perez ; Francisco J. Alvarez Mata ; Victor M. Zamudio Rodriguez ; Shumei Zhang IEEE 2018

An unhealthy lifestyle, such as lack of exercise, poor diet, and drinking alcohol excessively can increase the risk of chronic disease. With the growth of Internet of Things (IoT) and the emergence of the wearable devices, it is possible to monitor important aspects of our daily life, with the aim of encouraging healthier lifestyles. This paper proposes a personal healthcare system that will allow the user to track their daily activities such as exercise, sleeping, heart-rate and diet, using only a smart watch and phone. The monitored data, behavior analysis and feedback from user's doctor can be shared via cloud services, and visualized using charts and tables. Users also can chat with an artificial intelligent agent that deals with related questions via a mobile app.

Chapter 3

METHODOLOGY

3.1 ARCHITECTURE BLOCK DIAGRAM

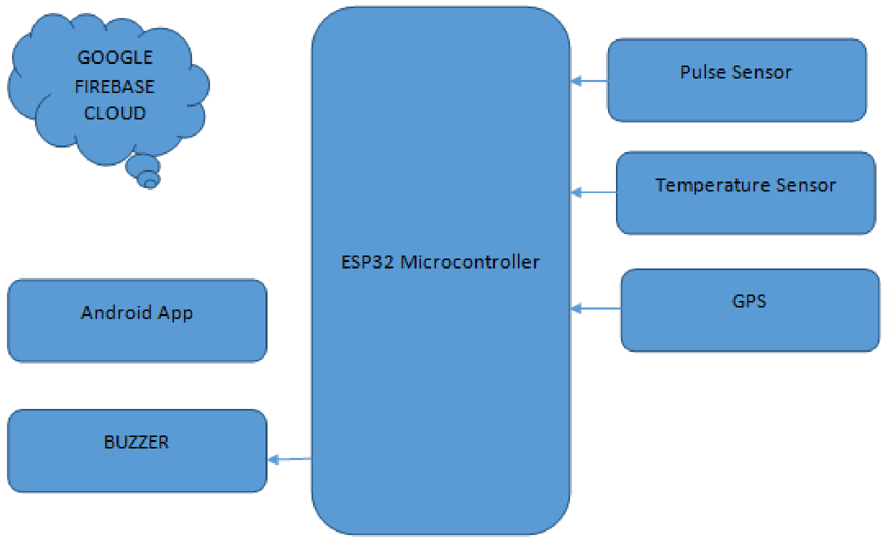


Figure 3.1 Block diagram

3.1.1 System Architecture

In smart health monitoring system, ESP32 is used as controller IOT module for communication. The pulse sensor is used to collect physiological signals and the sensitive temperature sensor is used for BT measurement. The general structure of the proposed system for smart health practice is shown in Figure 7. The system consists of ESP 32 Micro controller, MIT application, Firebase and physiological sensors.

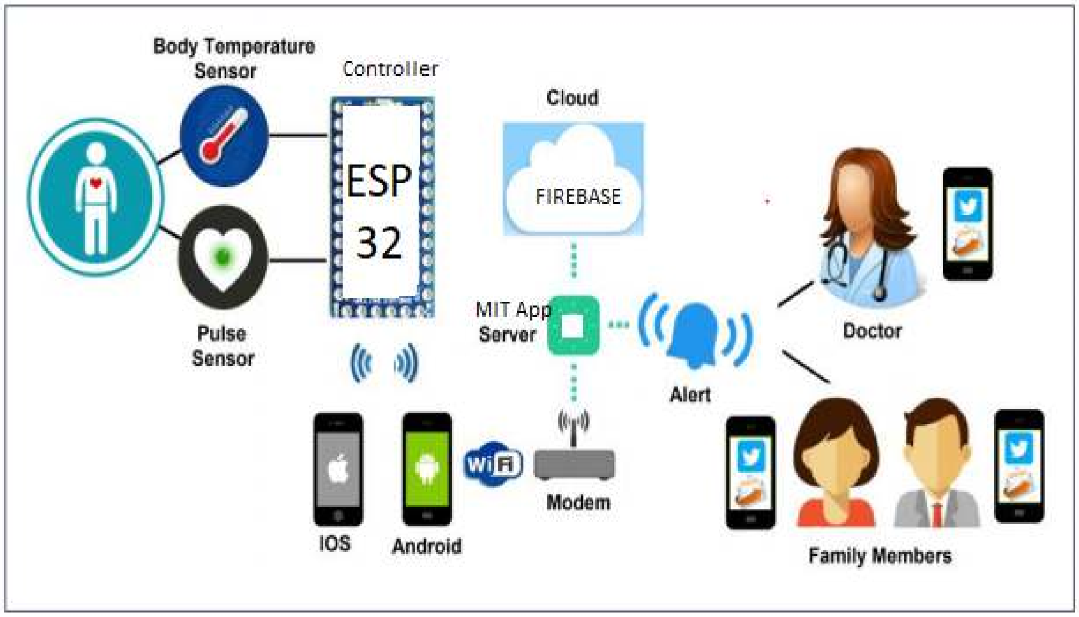


Figure 3.2 Structure of the System

Smart health measurement system built with the ESP32 controller. The Pulse Sensor measures the PPG signals that allow the calculation of heart-related values such as HR and HRV. This sensor is connected to the A0 analog input which converts 10 bits of the ESP 32 controller.

The Negative Temperature Coefficient (NTC) temperature sensor used for BT measurement is connected to A1 analog input of the controller. The NTC temperature sensor changes the resistance value depending on the temperature. The resistance value of the NTC is inversely proportional to the temperature and the sensitivity is very high. Due to its cheap, stable, and small structure, it is preferred in most sensitive temperature measurement applications. The sensor output is not linear, so the resistance values corresponding to each temperature measured in the program are written in a table. Generally, these values are presented in tables prepared with 1°C sensitivity by the manufacturers in web pages [20]. The table for 0.1°C sensitivity for 10K NTC used in CT measurement is created by intermediate value calculation method.

3.2 PPG Concepts

HR is the number of heart beats per minute and unit of HR is "bpm" (beats per minute). Every person's HR value can be different and many factors affect HR, such as body movements, emotional state, hunger, toughness, air temperature. HR values for ordinary people are considered normal between 60 and 100 beat per minute.

In athletes, normal HR can be reduced to 50 or even 40. For this reason, sports people over time can strengthen their heart muscles and pump more blood at a time than normal people. The famous cyclist racer Lance Armstrong has been reported to have thrown the ball 32 bpm at rest [18].

Apart from HR, very important heart-related values can be measured and depending on these, some diseases can be diagnosed. PPG is the recording of electrical activity in the heart to examine the functioning of the heart muscle and neural transmission system. In Figure 1 shows the values on a PPG signal that provide important findings about the heartbeat study.

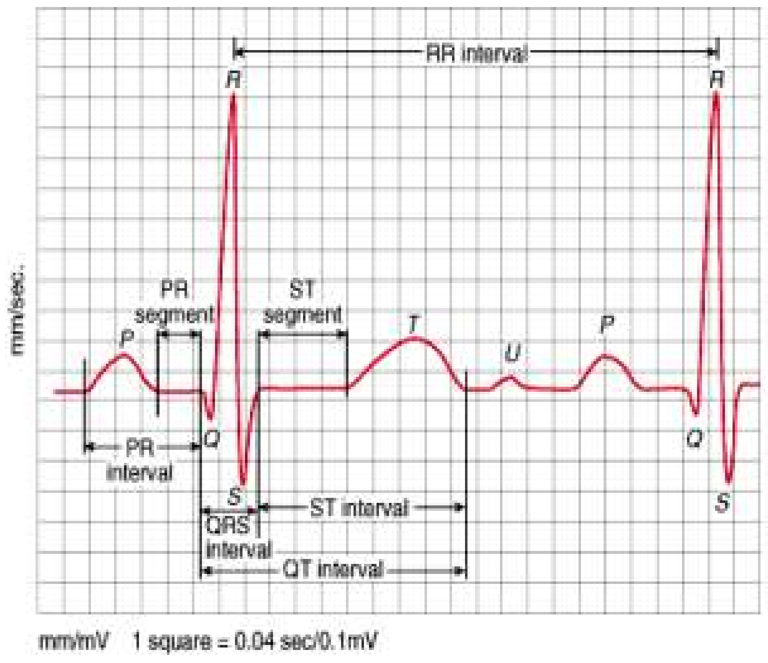


Figure 3.3 PPG signal.

Heart rate variability (HRV):

Heart rate variability is an important parameter in determining cardiac autonomic functions. HRV varies according to psycho physiological conditions such as stress, relief, exercise, as well as age and health. It is obtained by examining short-term (3-5 min) or long-term (24 h) signals.

PR interval : PR interval is the time between the P wave and the QRS complex. Normal PR interval is between 120-200 ms.

QRS interval: Normal QRS interval is between 70-100 ms.

RR interval: IBI (Inter beat interval): It's the time between two heart beats. This time varies depending on the heart rate. Normal RR interval is between 600-1000 ms.

QT interval: QT interval is the time from the beginning of the QRS complex to the end of the T wave. QT interval and HR are inversely proportional. It also varies with age and sex. Normal range is 320-440 ms.

ST interval : ST interval is inversely proportional to HR and the normal range is 0-150 ms.

Pulse Sensor

The heartbeat is detected by the reflection of the light emitted by the green LED on the APDS-9008 light sensor. In Figure 3.4 shows the circuit diagram of the pulse sensor. The low pass filter used at the output of the light sensor clears the signal from high frequency noise.

After the output of the filter, the signal is amplified by the MCP-6001 Op-Amp. The analog signal from this amplifier has a frequency of 0,5 to 4 Hz.

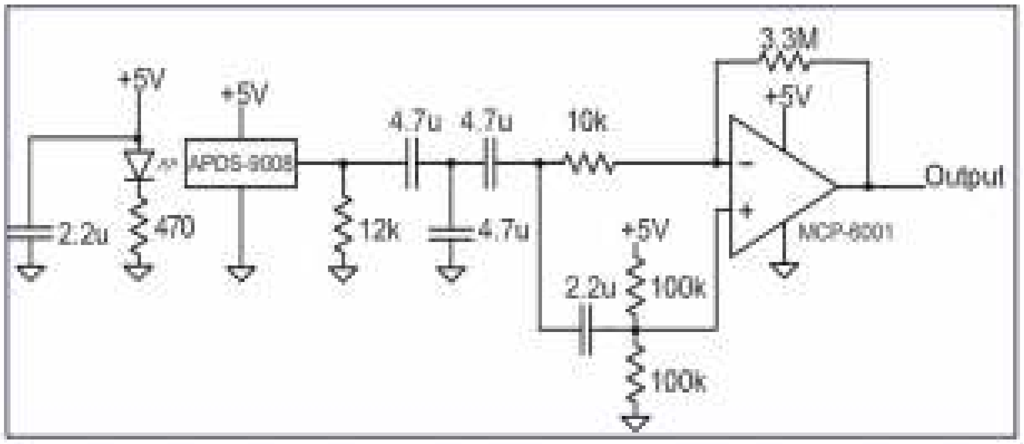


Figure 3.4 Pulse Sensor Circuit diagram.

The sensor output voltage, which varies from 0.25 to 1.5V, is converted into an output signal with an offset value of 512 adc. This output is normalized around V / 2 which is half the supply voltage. If the sensor cannot make a contact, the V / 2 value is taken from the sensor output

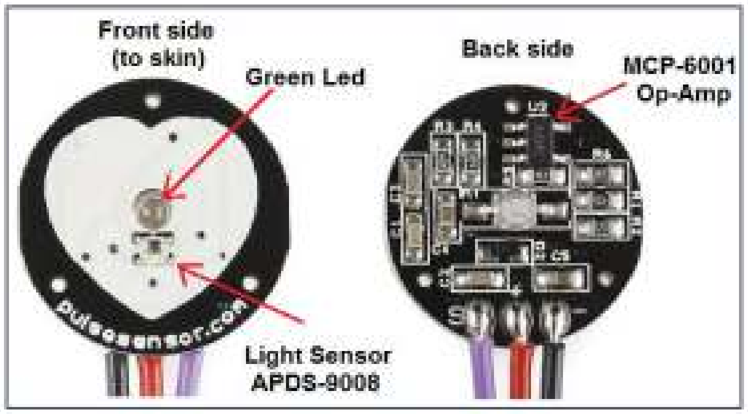


Figure 3.5 Pulse Sensor front and back sides.

The light emitted by the green LED in Figure 3.5 is reflected back to the APDS-9008 light sensor in every heart beat, depending on the oxygen saturation in the finger blood. This sensor responds to variations in light intensity and the amplitude of the output signal depends on the amount of light.

ESP32 Microcontroller

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs a TensilicaXtensa LX6 microprocessor in both dual-core and single-core variations and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.[2] It is a successor to the ESP8266 microcontroller.

Arduino Software (IDE)

The open source Arduino IDE editor allows you to write code and easily load it to the controllers via USB. The Arduino IDE supports many different controllers besides Arduino kits (Uno, Pro Mini, Mega, Due etc.). This software works on Windows, Mac OS X and Linux.

The Arduino IDE is written in the Java language and is based on the language named Processing/Wiring. The libraries are written in C and C ++ languages and compiled with AVR-GCC and AVR Libc. The code for the "Smart Health" has been developed using this interface. It is also compatible with advanced controllers such as Blynk, ESP, Onion Omega, Raspberry Pi, STM32. These specification of the controllers with bluetooth feature such as ESP32 can be used for closed circuit communication by consuming low energy on-line. Thanks to all these features Blynk; IOT offers solutions for developers at low prices and these users can also benefit from the cloud service at the same time.

3.3 SYSTEM REQUIREMENTS

Hardware Requirements

- ESP32 Microcontroller
- Pulse Sensor
- DHT11
- GPS Module
- Buzzer

Software Requirements

- Google Firebase cloud
- MIT App inventor
- Embedded C
- Arduino IDE

Chapter 4

HARDWARE

4.1 ESP32 Microcontroller

The DOIT Esp32 DevKit v1 is one of the development board created by DOIT to evaluate the ESP-WROOM-32 module. It is based on the ESP32 microcontroller that boasts Wi-Fi, Bluetooth, Ethernet and Low Power support all in a single chip.

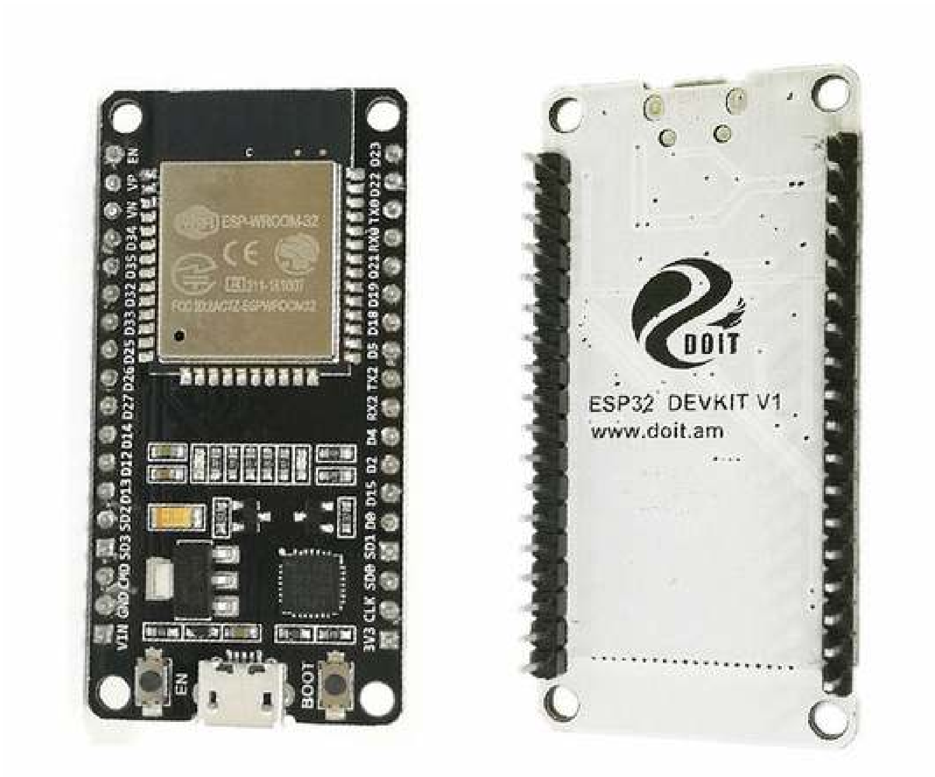


Figure 4.1 ESP32 Controller

4.1.1 Pin Mapping

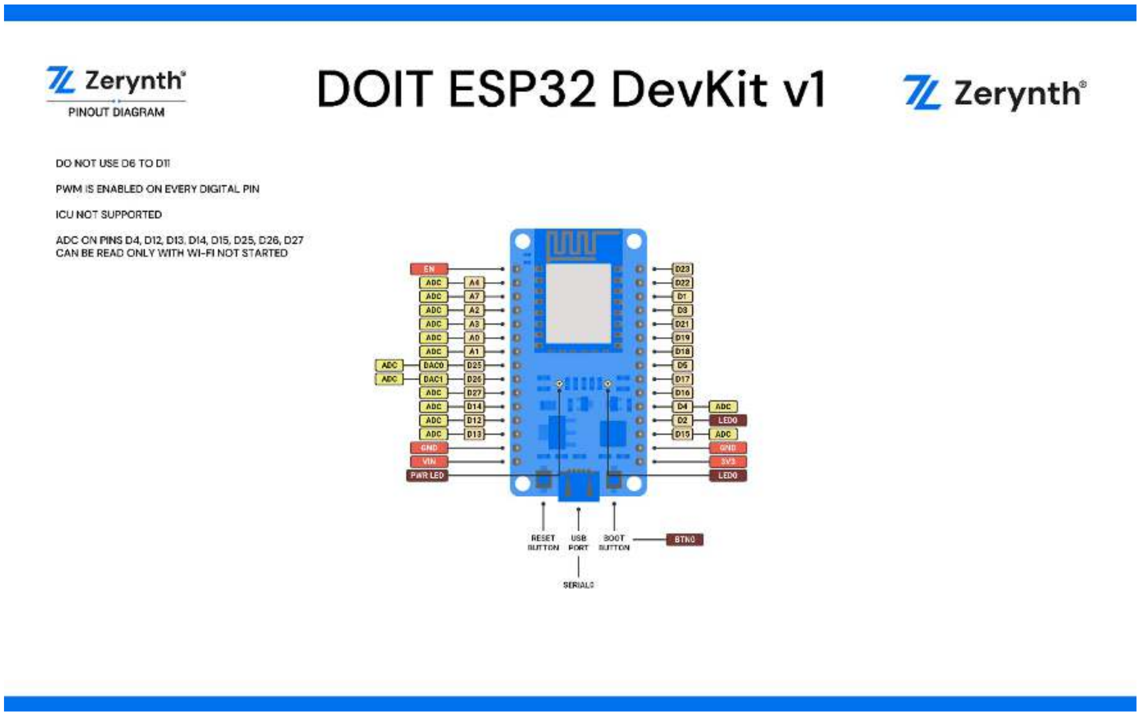


Figure 4.2 ESP32 pin mapping

4.1.2 Flash Layout

The internal flash of the ESP32 module is organized in a single flash area with pages of 4096 bytes each. The flash starts at address 0x00000, but many areas are reserved for Esp32 IDF SDK and Zerynth VM. There exist two different layouts based on the presence of BLE support.

In particular, for non-BLE VMs:

Start address	Size	Content
0x00009000	16Kb	Esp32 NVS area
0x0000D000	8Kb	Esp32 OTA data

Start address	Size	Content
0x0000F000	4Kb	Esp32 PHY data
0x00010000	1Mb	Zerynth VM
0x00110000	1Mb	Zerynth VM (FOTA)
0x00210000	512Kb	Zerynth Bytecode
0x00290000	512Kb	Zerynth Bytecode (FOTA)
0x00310000	512Kb	Free for user storage
0x00390000	448Kb	Reserved

For BLE VMs:

Start address	Size	Content
0x00009000	16Kb	Esp32 NVS area
0x0000D000	8Kb	Esp32 OTA data
0x0000F000	4Kb	Esp32 PHY data
0x00010000	1216Kb	Zerynth VM
0x00140000	1216Kb	Zerynth VM (FOTA)

Start address	Size	Content
0x00270000	320Kb	Zerynth Bytecode
0x002C0000	320Kb	Zerynth Bytecode (FOTA)
0x00310000	512Kb	Free for user storage
0x00390000	448Kb	Reserved

4.1.3 Device Summary

- Microcontroller: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 25
- Analog Input Pins (ADC): 6
- Analog Outputs Pins (DAC): 2
- UARTs: 3
- SPIs: 2
- I2Cs: 3
- Flash Memory: 4 MB
- SRAM: 520 KB
- Clock Speed: 240 Mhz
- Wi-Fi: IEEE 802.11 b/g/n/e/i:

- Integrated TR switch, balun, LNA, power amplifier and matching network
- WEP or WPA/WPA2 authentication, or open networks

Power

Power to the DOIT Esp32 DevKit v1 is supplied via the on-board USB Micro B connector or directly via the "VIN" pin. The power source is selected automatically.

The device can operate on an external supply of 6 to 20 volts. If using more than 12V, the voltage regulator may overheat and damage the device. The recommended range is 7 to 12 volts.

Connect, Register, Virtualize and Program

The DOIT Esp32 DevKit v1 comes with a serial-to-usb chip on board that allows programming and opening the UART of the ESP32 module. Drivers may be needed depending on your system (Mac or Windows) and can be download from the official Espressif documentation page. In Linux systems, the DevKit v1 should work out of the box.

For Linux Platform: to allow the access to serial ports the user needs read/write access to the serial device file. Adding the user to the group, that owns this file, gives the required read/write access:

- **Ubuntu** distribution –>dialout group
- **Arch Linux** distribution –>uucp group

Once connected on a USB port, if drivers have been correctly installed, the DevKit v1 device is recognized by Zerynth Studio. The next steps are:

- **Select** the DevKit v1 on the **Device Management Toolbar** (disambiguate if necessary);
- **Register** the device by clicking the "Z" button from the Zerynth Studio;
- **Create** a Virtual Machine for the device by clicking the "Z" button for the second time;
- **Virtualize** the device by clicking the "Z" button for the third time.

After virtualization, the DevKit v1 is ready to be programmed and the Zerynth scripts **uploaded**. Just **Select** the virtualized device from the "Device Management Toolbar" and **click** the dedicated "upload" button of Zerynth Studio.

Firmware Over the Air update (FOTA)

The Firmware Over the Air feature allows to update the device firmware at runtime. Zerynth FOTA in the DevKitC device is available for bytecode and VM.

Flash Layout is shown in table below:

Start address	**Size**	**Content**
0x00010000	1Mb	Zerynth VM (slot 0)
0x00110000	1Mb	Zerynth VM (slot 1)
0x00210000	512Kb	Zerynth Bytecode (slot 0)
0x00290000	512Kb	Zerynth Bytecode (slot 1)

For BLE VMs:

Start address	**Size**	**Content**
0x00010000	1216Kb	Zerynth VM (slot 0)
0x00140000	1216Kb	Zerynth VM (slot 1)
0x00270000	320Kb	Zerynth Bytecode (slot 0)
0x002C0000	320Kb	Zerynth Bytecode (slot 1)

For Esp32 based devices, the FOTA process is implemented mostly by using the provided system calls in the IDF framework. The selection of the next VM to be run is therefore a duty of the Espressif bootloader; the bootloader however, does not provide a failsafe mechanism to revert to the previous VM in case the currently selected one fails to start. At the moment this lack of a safety feature can not be circumvented, unless by changing the bootloader. As soon as Espressifrelases a new IDF with such feature, we will release updated VMs.

Secure Firmware

Secure Firmware feature allows to detect and recover from malfunctions and, when supported, to protect the running firmware (e.g. disabling the external access to flash or assigning protected RAM memory to critical parts of the system).

4.2 GPS module:

Fig 4.3 GPS Module

A GPS navigation device, GPS receiver, or simply GPS is a device that is capable of receiving information from GPS satellites and then to calculate the device's geographical position. It records the longitude and latitude of its position. Using suitable software, the device may display the position on a map, and it may offer directions. The Global Positioning System (GPS) is a global navigation satellite system (GNSS) made up of a network of a minimum of 24, but currently 30, satellites placed into orbit by the U.S. Department of Defense.

4.2.1 Segments of GPS communication:

This communication system consists of 3 segments namely Space segment (the satellites), Control segment(the ground stations), and User segment(GPS receiver and its user).

The space segment of GPS system consists of 24 satellites . The number may vary as new ones are launched and old ones are retired . Each satellite is in an 11,000-mile orbit Each satellite transmits low power radio signals on several frequen -cies . The signals travels “line of sight”, meaning it will pass through clouds , glass and plastic, but will not go through most solid objects such as buildings and mountains.

The control segment as implied by the name controls the GPS satellites by tracking them and then providing them corrected orbital and clock (time) information. There are five control stations around the world four unmanned monitoring stations and one “master control station”. The four unmanned receiving stationsconsta -ntly receive data from the satellites and then send that information to master control station . The master control station corrects the satellite data and , together with two antenna sites, sends (“uplinks”) the information to the GPS satellites.

The user segment consists of the GPS receiver and its user (boasters, pilots, hikers, hunters, the military etc.).

4.2.2 Working of GPS:

To start with, assume that all of the satellites and the receiver have a perfect internal clock. This is not the case, but it makes a good starting point. Each satellite transmits a coded signal. Consider this signal to be like the peaks and ridges along the edge of a super long key

This code is generated as a function of time. The receiver is also able to generate the same code. The receiver matches the incoming code to the internally generated code except that there is a delay caused by the signal’s travel time between the satellite and the receiver. The receiver measures how much it has had to shift the timing of its code to match the incoming code. Since the receiver knows how much time it took the signal to reach the receiver and the speed of travel of the signal, it can then calculate the distance from the satellite.

Although no clock is perfect, the satellites have atomic clocks pretty close. The clock in the GPS receiver is closer in technology to an inexpensive digital watch . Light travels at 186,000 miles per second. If the receiver time was off by 1/100 of a second the calculated distance would be off by 1,860 miles.

For each receiver to have its own cesium clock would make GPS technology prohibitively expensive and non-portable. What the GPS receiver does is to use cheap clock similar to a digital watch and add one more satellite to the calculation to correct the time in the receiver. The receiver shifts the time calculation back and forth so that all of the imaginary spheres around the satellites intercept at one point.

For three-dimensional navigation you need to receive four satellites. Think of it as one satellite for each dimension and one for the time. For two dimensional navigation you can scrape by with only receiving three satellites. If you know your altitude , the GPS can treat the center of the earth as a satellite reducing the number of required satellite by one .Your distance from the center of the earth is the radius of the earth plus your altitude . This is why aviation GPS models have barometric altimeter input and you may occasionally see a handheld GPS ask for your altitude during poor reception conditions.

The GPS receiver picks up two kinds of coded information from the satellites. One type of information, called “almanac” data, contains the approximate positions (locations) of the satellites. This data is continuously transmitted and stored in the memory of the GPS receiver’s it knows the orbits of the satellites and where each satellite is supposed to be. The almanac data is periodically updated with new information as the satellites move around.

Any satellite can travel slightly out of the orbit, so the ground monitor stations keep track of the satellite orbits, altitude, location and speed. The ground stations send the orbital data to the master control system, which intern sends corrected data up to the satellites. This corrected and exact position data is called the “ephemeris” data, which is valid for about 4-6 hours, and is transmitted in the coded information to the GPS receiver. So, having received the almanac and ephemeris data, the GPS receiver knows the position of the satellites at all times.

Time of essence:

Even though the GPS receiver knows the précised location of the satellites in the space , it still needs to know how far away the satellites are . So it can determine its position on earth. There is a simple formula that tells the receiver how far it is from each satellite. Your distance from a given satellite object equals the velocity of the transmitted signal multiplied by the time it takes the signal to reach the receiver.

Mathematical Analysis:

Let the 3D coordinates of the receiver be x1, y1, z1. As we have learnt that the receiver clock is not that accurate as the satellite clock , while we calculate the coordinates, the time error comes into consideration . Let the error be ‘dt’. The simple equation with which we calculate the position of the receiver is

$$\text{Sqrt}\,[(x-x1)2+(y-y1)2+(z-z1)2] = \text{velocity of light} * (T + dt)$$

Where T is the time taken for the signal to propagate from the satellite to the receiver. There are four unknowns in the equation. So we should get four equations from 4 randomly selected satellites out of 24 satellites . Now we calculate x1 ,y1 ,z1 and dt which gives the accurate position of the user .

Almanac data:

The unit stores data about where the satellites are located at any given time .This data is called the Almanac. Sometimes when the GPS unit is not tuned on for a length of time , the almanac can become outdated or “cold” When the GPS receiver is cold , it could take longer to acquire satellites . A receiver is considered “warm” when the data has been collected from the satellites within the last four to six hours.

Once the GPS has locked onto enough satellites to calculate a position , the user is ready to begin navigation . Most GPS units will display a position page or a page showing your position on a map(map screen) that will assist you in your navigation .

4.2.3 GPS Receiver Technology:

Most modern GPS receivers are a parallel multi-channel design. Older single-channel designs were once popular, but were limited in their ability to continuously receive signals in the toughest environments such as under tree cover. Parallel receivers typically have from between 5 to 12 receiver circuits, each devoted to one particular satellite signal , so strong locks can be maintained on all the satellites at all the times. Parallel channel satellites are quick to lock onto satellites when first turned on and they are unequaled in their ability to receive the satellite signals even in difficult conditions such as dense foliage or urban settings with tall buildings.

Sources of errors:

Civilian GPS receivers have potential errors due to the result of the accumulated errors primarily due to the following sources:

1. Ionosphere and troposphere delays.

2. Signal multi-path.

3. Recover clock errors.

4. Orbital errors.

5. Number of satellites visible.

6. Satellite geometry.

7. Intentional degradation of the satellite signal by the U.S military.

However the accuracy can be improved by combining the GPS receiver with a Differential GPS (DGPS) receiver.

Versions of GPS:

- Differential GPS:

  The idea of differential GPS is to install a GPS receiver at a known point. Since the location is known, this GPS compares the distance to each satellite and to what it should be and then rebroadcasts the error in digital format.

  GPS receivers so equipped can then use this known error in its position calculations. Basic GPS uses local receivers and local transmitters. In order to use differential GPS, you need a special differential receiver which then sends the signal to the GPS. Differential GPS is mainly a marine application and it is not widely used for recreational applications. One of the biggest advantages of differential GPS is that it helped to eliminate the purposeful errors caused by Selective Availability.

  Now that Selective Availability has been turned off, the level of accuracy increase from using differential GPS is significantly less. Additionally, WAAS, Wide Area Augmentation Service is common in most new GPS receivers.

- WAAS:

  Another differential technique is known as WAAS, Wide Area Augmentation Service. WAAS has 25 receivers scattered around the United States. A mathematical model of the satellite errors is created based on the measurements and the error correction values are then sent to a geo synchronous satellite to be rebroadcast.

  The advantage of WAAS over conventional differential GPS is that it is available in small handheld receivers without needing a separate receiver. In fact, almost every new GPS receiver is WAAS capable. WAAS was designed for aviation use. GPS is more accurate in laterally than for altitude. One of the goals of WAAS was to provide sufficient accuracy to allow GPS to be used to provide vertical guidance during an instrument approach. Most larger airports have something called an ILS (Instrument Landing System)which provides a “radio beam” down to the runway.

  However, many smaller airports do not have this expensive navigational infrastructure. There are many issues involved, but WAAS enabled GPS approaches with vertical guidance offer big safety improvements to runways without ILS equipment. The other goal of WAAS is not so much accuracy as it is integrity. If a satellite is sending a bad signal, it takes a few minutes to detect and stop broadcasting the signal.

Currently aviation receivers use satellite signals beyond the minimum required to cross check the accuracy of the signal. For example, if you need 4 satellites to determine a position and you are receiving 5 satellites, you can use the extra signal as a cross check. This is called RAIM, ReceiverAutonomous Integrity Monitoring. Part of WAAS provides integrity checking which is faster than what is offered through the basic GPS system.

4.2.4 General applications of GPS:

- Precision Positioning:

  Surveyors and map makers use GPS for precision positioning. GPS is often used to map the location of such facilities as telephone poles, sewer lines, and fire hydrants. Surveyors use GPS to map construction sites and property lines. Forestry, mineral exploration, and wildlife habitat management all use GPS to precisely define positions of important assets and to identify changes.

  During data collection, GPS points can be assigned codes to identify them as roads, streams, or other objects. These data can then be compared and analyzed in computer programs called Geographic Information Systems (GIS).
- Surveying With GPS:

  Surveying that previously required hours or even days using conventional methods can be done in minutes with GPS.

- Set Your Watch:

  Because GPS includes a very accurate time reference, the system is also widely used for timekeeping. GPS receivers can display time accurate to within 150 billionths of a second.

- GPS in Vehicles:

  Many types of GPS systems can be used on vehicles, providing the driver with the current position and a local map.

- Nautical Chart Error:

  The data collected from satellite navigation systems provide more accurate information for maps and nautical and aeronautical charts. This example demonstrates how charts are updated to prevent navigational mishaps.

- GPS Navigation in the Air:

  Pilots on long distance flights without GPS rely on navigational beacons located across the country. Using GPS, aircraft can fly the most direct routes between airports.

- GPS in the Cockpit:

  Pilots often rely on GPS to navigate to their destinations. A GPS receiver in the cockpit provides the pilot with accurate position data and helps him or her keep the airplane on course.

Interpreted sentences transmitted by GPS unit:

$GPAAM - Waypoint Arrival Alarm

$GPALM - GPS Almanac Data(Can also be received by GPS unit)

$GPAPB - Autopilot format "B"

$GPBOD - Bearing, origin to destination

$GPBWC - Bearing and distance to waypoint, great circle

$GPGGA - Global Positioning System Fix Data

$GPGLL - Geographic position, latitude / longitude

$GPGRS - GPS Range Residuals

$GPGSA - GPS DOP and active satellites

$GPGST - GPS Pseudo range Noise Statistics

$GPGSV - GPS Satellites in view

$GPHDT - Heading, True

$GPMSK - Control for a Beacon Receiver

$GPMSS - Beacon Receiver Status

$GPR00 - List of waypoints in currently active route

$GPRMA - Recommended minimum specific Loran-C data

$GPRMB - Recommended minimum navigation info

$GPRMC - Recommended minimum specific GPS/Transit data

$GPRTE - Routes

$GPTRF - Transit Fix Data

$GPSTN - Multiple Data ID

$GPVBW - Dual Ground / Water Speed

$GPVTG - Track made good and ground speed

$GPWPL - Waypoint location

$GPXTE - Cross-track error, Measured

$GPZDA - UTC Date / Time and Local Time Zone Offset

$GPRMC:

Recommended minimum specific GPS/Transit data:

$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a,m*hh

Field #

1= UTC time of fix

2= Data status (A=Valid position, V=navigation receiver warning)

3= Latitude of fix

4= N or S of longitude

5 = Longitude of fix

6= E or W of longitude

7= Speed over ground in knots

8 = Track made good in degrees True

9 = UTC date of fix

10= Magnetic variation degrees (Easterly var. subtracts from true course)

11= E or W of magnetic variation

12= Mode indicator,(A=Autonomous,D=Differential,E=Estimated,N=Data notvalid)

13= Checksum

Eg1.

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

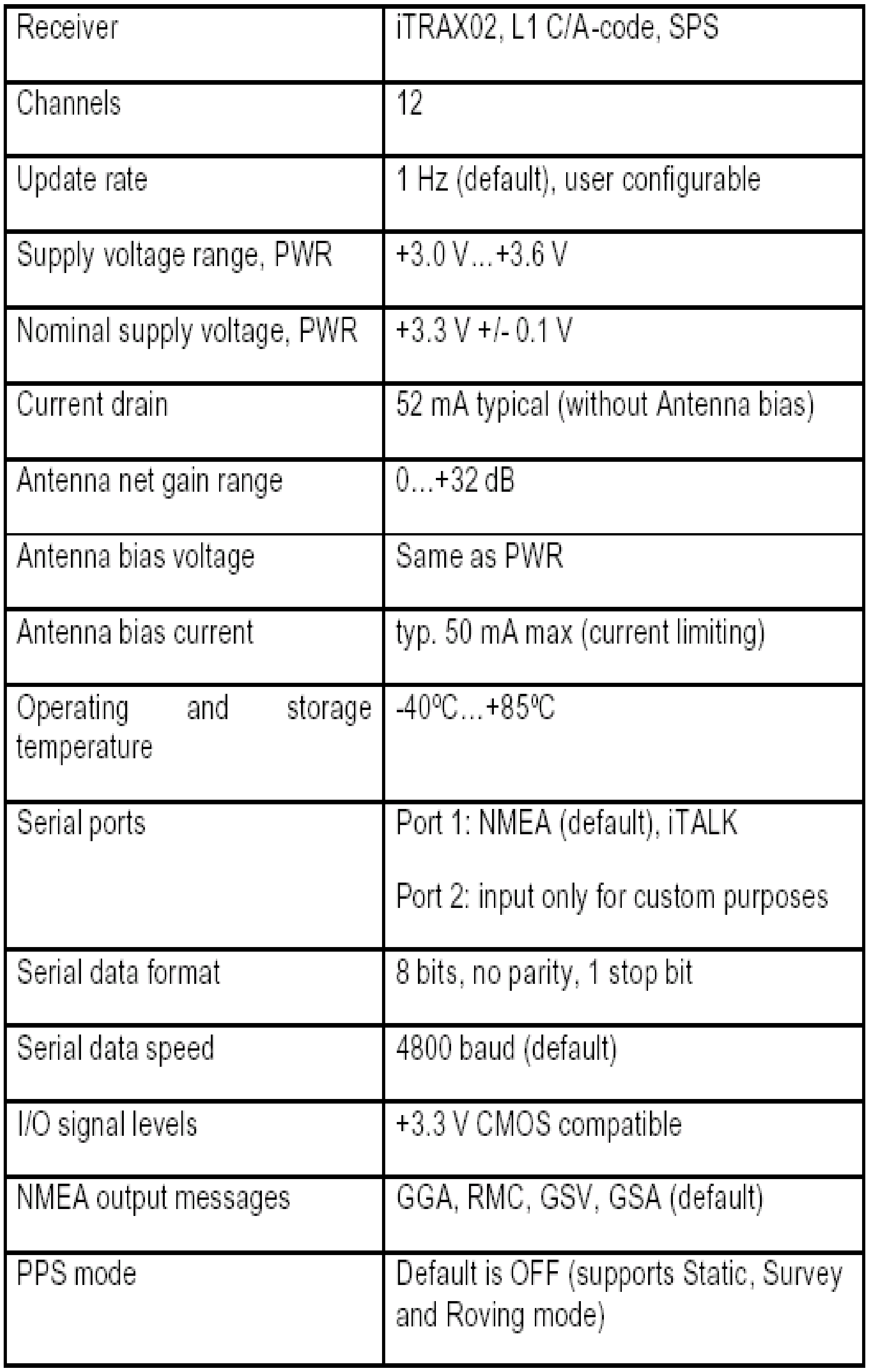
225446	Time of fix 22:54:46 UTC
A	Navigation receiver warning A = Valid position, V = Warning
4916.45,N	Latitude 49 deg. 16.45 min. North
12311.12,W	Longitude 123 deg. 11.12 min. West
000.5	Speed over ground, Knots
054.7	Course Made Good, degrees true
191194	UTC Date of fix, 19 November 1994
020.3,E	Magnetic variation, 20.3 deg. East
*68	mandatory checksum

4.2.5 GPS RECEIVER:

uTracker02-LLP

The uTracker02-LLP is a GPS receiver board, which consists of the Integrated iTRAX02 (ref #1,2) receiver electronics with a 8Mbit flash memory together with a regulator and a power-up-reset circuit. The nominal power supply and I/O signal levels are +3.3V.The module provides complete signal processing from antenna to serial data output in either NMEA messages or in proprietary talk binary protocol (configurable). A second serial port input is also optionally available for custom purposes. Also a PPS pulse can be provided for timing purposes. The module supports both passive and active GPS antennas with a +3.3V bias voltage, which includes also a protection against short circuit. The module hardware and firmware supports optionally an Antenna Bias Supervisor, which is able to detect false states of the active antenna bias current and voltage, either Short or Open state.

Specifications



Receiver	iTRAX02, L1 C/A-code, SPS
Channels	12
Update rate	1 Hz (default), user configurable
Supply voltage range, PWR	+3.0 V…+3.6 V
Nominal supply voltage, PWR	+3.3 V +/- 0.1 V
Current drain	52 mA typical (without Antenna bias)
Antenna net gain range	0…+32 dB
Antenna bias voltage	Same as PWR
Antenna bias current	typ. 50 mA max (current limiting)
Operating and storage temperature	-40ºC…+85ºC
Serial ports	Port 1: NMEA (default), iTALK Port 2: input only for custom purposes
Serial data format	8 bits, no parity, 1 stop bit
Serial data speed	4800 baud (default)
I/O signal levels	+3.3 V CMOS compatible
NMEA output messages	GGA, RMC, GSV, GSA (default)
PPS mode	Default is OFF (supports Static, Survey and Roving mode)

4.3 Pulse sensor

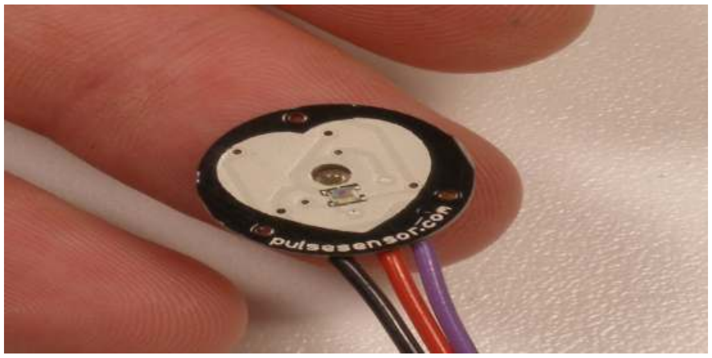


Figure 4.4(a) pulse sensor

Introduction:

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. It can be used by students, artists, athletes, makers, and game & mobile developers who want to easily incorporate live heart- rate data into their projects. The sensor clips onto a fingertip or earlobe and plugs right into Arduino with some jumper cables. It also includes an open-source monitoring app that graphs your pulse in real time.

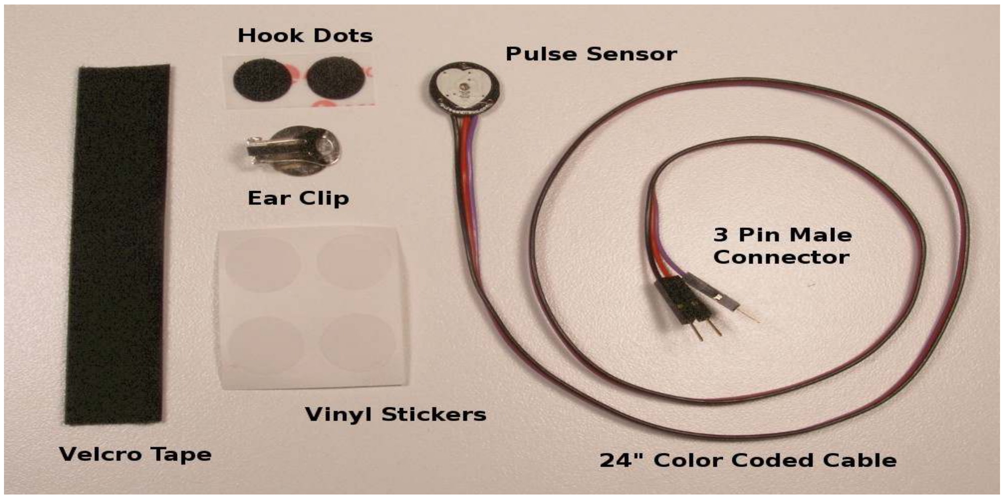


Figure 4.4(b) pulse sensor kit

T1he Pulse Sensor Kit includes:

1) A24-inchColor

Coded Cable, with (male) header connectors. It easy to embed the sensor into the project, and connect to an controller. No soldering is required.

2) An Ear Clip

An Ear Clip perfectly sized to the sensor. We searched many places to find just the right clip. It can be hot- glued to the back of the sensor and easily worn on the earlobe.

3) 2 Velcro Dots.

These are 'hook' side and are also perfectly sized to the sensor. Velcro dots very useful if you want to make a Velcro (or fabric) strap to wrap around a fingertip.

4) 3Transparent Stickers.

These used on the front of the Pulse Sensor to protect it from oily fingers and sweaty earlobes.

5) The Pulse Sensor has 3 holes around the outside edge which make it easy to sew it into almost anything.

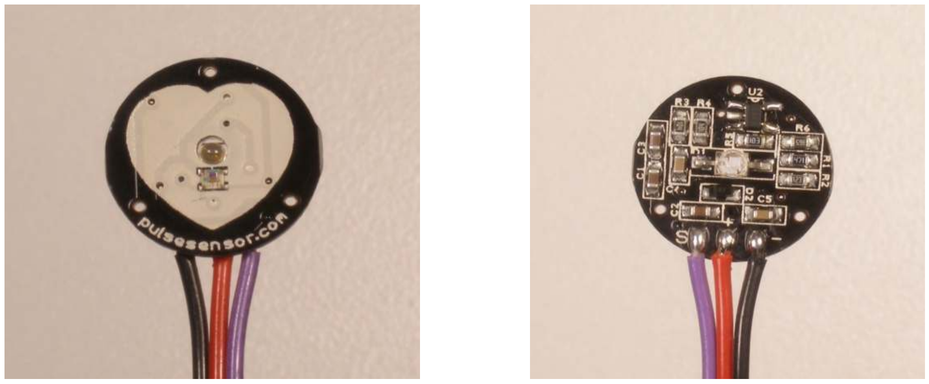


Figure 4.4(c)

4.3.1 Pulse Sensor Anatomy

The front of the sensor is the pretty side with the Heart logo. This is the side that makes contact with the skin. On the front you see a small round hole, which is where the LED shines through from the back, and there is also a little square just under the LED. The square is an ambient light sensor, exactly like the one used in cell phones, tablets, and laptops, to adjust the screen brightness in different light conditions.

The LED shines light into the fingertip or earlobe, or other capillary tissue, and sensor reads the light that bounces back. The back of the sensor is where the rest of the parts are mounted. We put them there so they would not get in the way of the of the sensor on the front. Even the LED we are using is a reverse mount LED. For more about the circuit functionality, check out the Hardware page.[needslink]

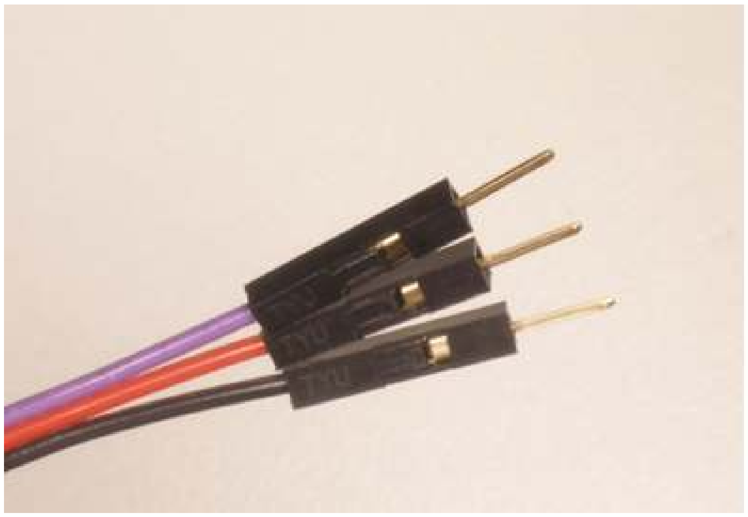


Figure 4.4 (d)

RED wire = +3V to +5V

BLACK wire = GND

PURPLE wire = Signal

4.3.2 Preparing the Pulse Sensor

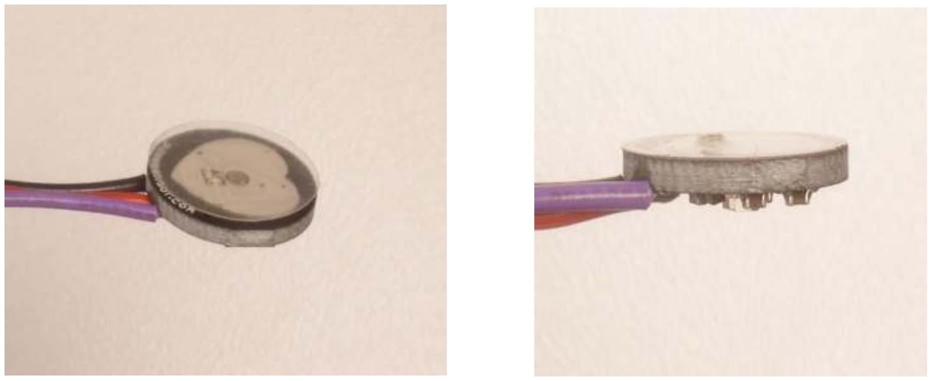


Figure 4.4(e)

Before start using the sensor we need to insulate the board from our (naturally) sweaty/oily fingers. The Pulse Sensor is an exposed circuit board, and if you touch the solder points, you could short the board, or introduce unwanted signal noise. We will use a thin film of vinyl to seal the sensor side. Then center it on the Pulse Sensor. It should fit perfectly.

When you are happy with the way it's lined up, squeeze it onto the face all at once! The sticker (made of vinyl) will kind of stretch over the sensor and give it a nice close fit. If you get a wrinkle, don't worry, just press it down really hard and it should stick. We gave you 4, so you can replace it if necessary.

That takes care of the front side. The vinyl sticker offers very good protection for the underlying circuit, and we rate it 'water resistant'. meaning: it can stand to get splashed on, but don't throw it in the pool!

If this is your first time working with Pulse Sensor, you're probably eager to get started, and not sure if you want to use the ear-clip or finger-strap (or other thing). The back of the Pulse Sensor has even more exposed contacts than the front, so you need to make sure that you don't let it touch anything conductive or wet.

The easiest and quickest way to protect the back side from undesirable shorts or noise is to simply stick a Velcro dot there for now. The dot will keep your parts away from the Pulse Sensor parts enough for you to get a good feel for the sensor and decide how you want to mount it. You’ll find that the Velcro dot comes off easily, and stores back on the little strip of plastic next to the other one.

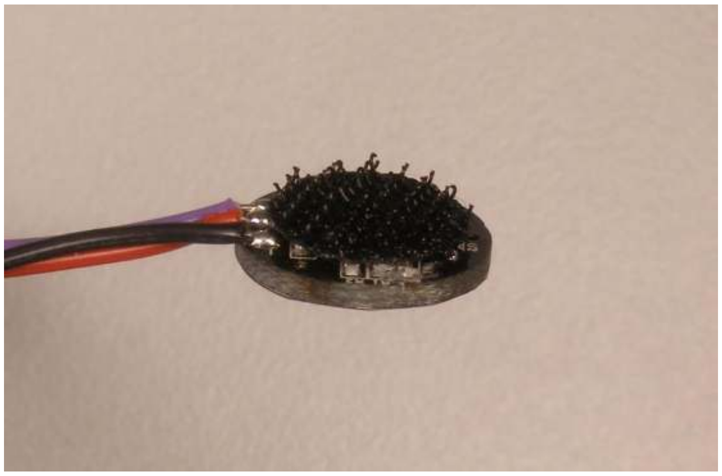


Figure 4.4(f)

4.3.3 Running The Pulse Sensor Code

Arduino code is called “PulseSensorAmped_Arduino-xx”

The Processing code is called “PulseSensorAmped_Processing-xx”

We strongly advise that you DO NOT connect the Pulse Sensor to your body while your computer or arduino is being powered from the mains AC line. That goes for charging laptops and DC power supplies. Please be safe and isolate yourself from from the power grid, or work under battery power.

Connect the Pulse Sensor to: +V (red), Ground (black), and Analog Pin 0 (purple) on your favorite Arduino, or Arduino compatible device, and upload the ‘Pulse Sensor Amped_Arduino-xx’ sketch.

After it’s done uploading, you should see Arduino pin 13 blink in time with your heartbeat when you hold the sensor on your fingertip. If you grip the sensor too hard, you will squeeze all the blood out of your fingertip and there will be no signal! If you hold it too lightly, you will invite noise from movement and ambient light. Sweet Spot pressure on the Pulse Sensor will give a nice clean signal. You may need to play around and try different parts of your body and pressures. If you see an intermittent blink, or no blink, you might be a zombie or a robot.

To view the heartbeat waveform and check your heart rate, you can use the Processing sketch that we made. Start up Processing on your computer and run the 'PulseSensorAmped_Processing-xx' sketch. This is our data visualization software, and it looks like this.

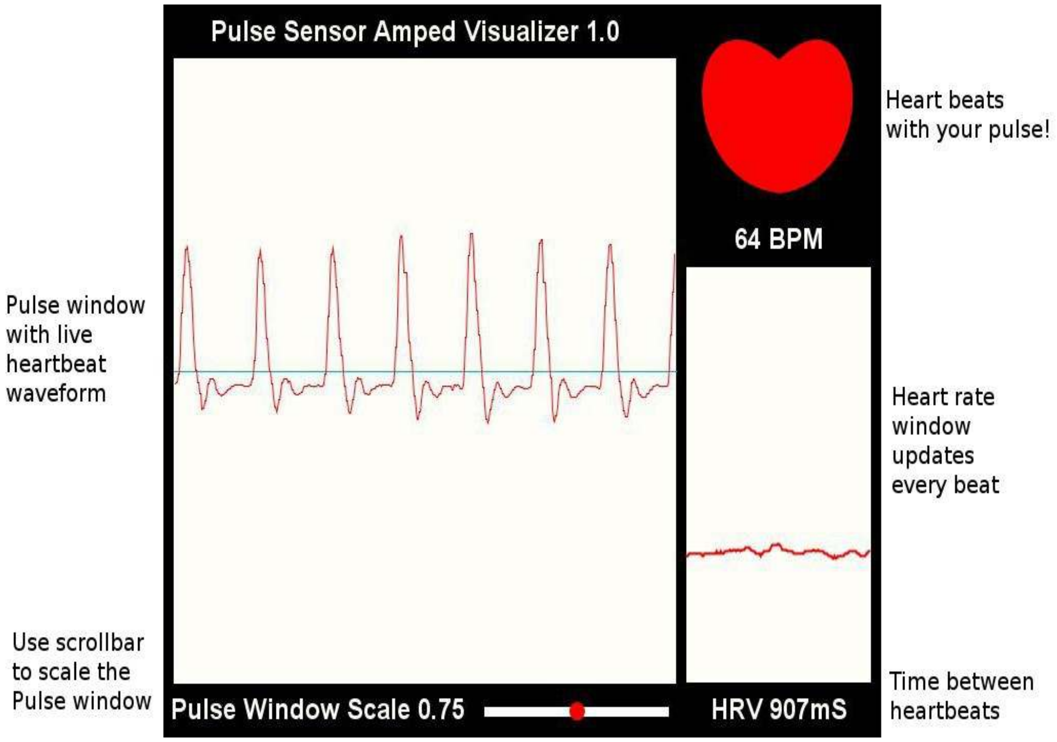

Pulse Sensor Amped Visualizer 1.0
Heart beats with your pulse!
64 BPM
Pulse window with live heartbeat waveform
Heart rate window updates every beat
Use scrollbar to scale the Pulse window
Pulse Window Scale 0.75
HRV 907mS
Time between heartbeats

Figure 4.5

The large main window shows a graph of raw sensor data over time. The Pulse Sensor Data Window can be scaled using the scrollbar at the bottom if you have a very large or very small signal. At the right of the screen, a smaller data window graphs heart rate over time. This graph advances every pulse, and the Beats Per Minute is updated every pulse as a running average of the last ten pulses. The big red heart in the upper right also pulses to the time of your heartbeat. When you hold the Pulse Sensor to your fingertip or earlobe or (fill in body part here) you should see a nice heartbeat waveform like the one above. If you don't, and you're sure you're not a zombie, try the sensor on different parts of your body that have capillary tissue. We've had good results on the side of the nose, middle of the forehead, palm, and lower lip. We're all different, original organisms. Play around and find the best spot on you and your friends. As you are testing and getting used to the sensor. You may find that some fingers or parts of fingers are better than others. For example, I find that when I position the sensor so that the edge of the PCB is at the bottom edge of my earlobe I get an awesome signal. Also, people with cold hands or poor circulation may have a harder time reading the pulse. Run your hands under warm water, or do some jumping-jacks!

It's really important to protect the exposed Pulse Sensor circuitry so the sweat of your fingertips or earlobe (or wherever) doesn't cause signal noise or a short circuit. This How-To uses hot glue, which can be removed easily or re-worked if you want to change where you've stuck your Pulse Sensor. We love hot glue!

The other things you'll need are:

Hot Glue Gun

Blue Tape (any tape should do ok)

Nail Trimmers (or your favorite trimming device. Flush-cut wire snips work well too)

First, attach the clear vinyl sticker to the front of your Pulse Sensor, as shown above. Then put a blob of hot glue on the back, right over the circuit. Size can be difficult to judge sometimes. What I meant was put a hot glue blob about the size of a kidney bean on the back side of the Pulse Sensor.

Then, while the glue is still very hot, press the Pulse Sensor glue-side-down onto the sticky side of a piece of blue tape (I believe that blue tape has magical properties, but if you don't have blue tape other kinds of tape will work just as well).

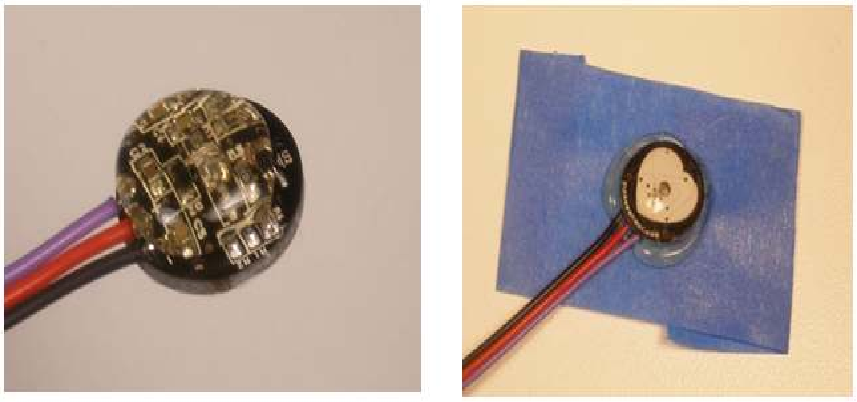


Figure 4.6(a)

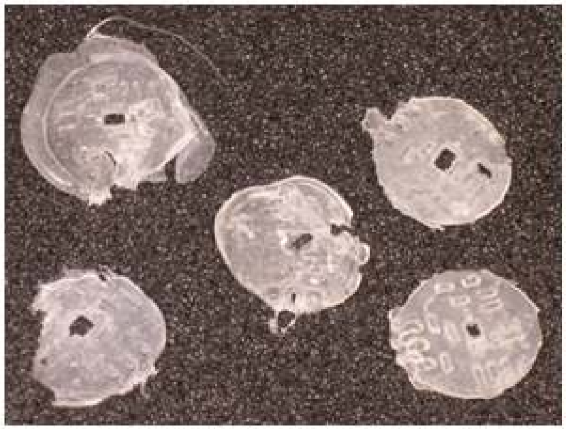


Figure 4.6(b)

The tallest thing on the back of the pulse sensor is in green LED housing right in the middle. Use it to make the hot-glue seal thin and strong. When you press evenly until the back of the LED touches, all the conductive parts will be covered with hot glue. If the glue doesn't ooze out all around, let it cool down, then peel if from the pulse sensor and try again. Once the glue has cooled down and has some body, you can peel it off easily.

Next put a small dab of hot glue on the front of the cables, where they attach to the pulse sensor circuit board. This will bond to the other glue that's there and act as a strain-relief for the cable connection can wear out over time.

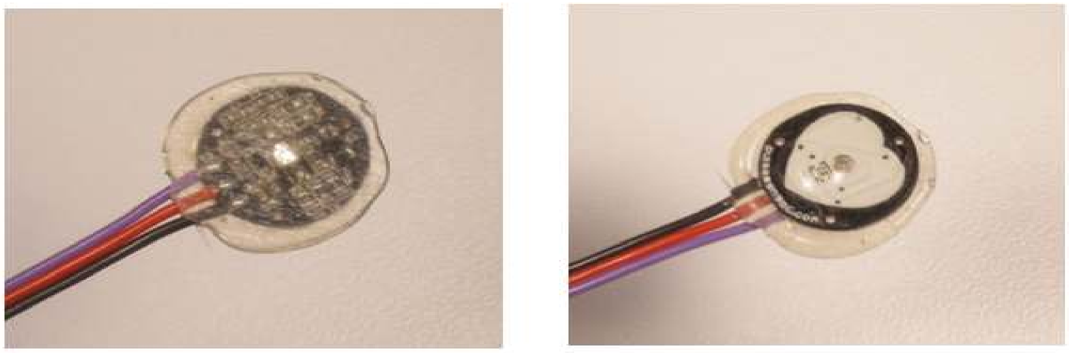


Figure 4.6(c)

Once the hot glue has cooled the blue tape will peel off very easily. Check your work to make sure that there are not exposed electrical connections. Next is trimming. Flush cutting wire snips work too. Take care not to clip the wires, and leave enough around the cable to act as a good strain-relief.

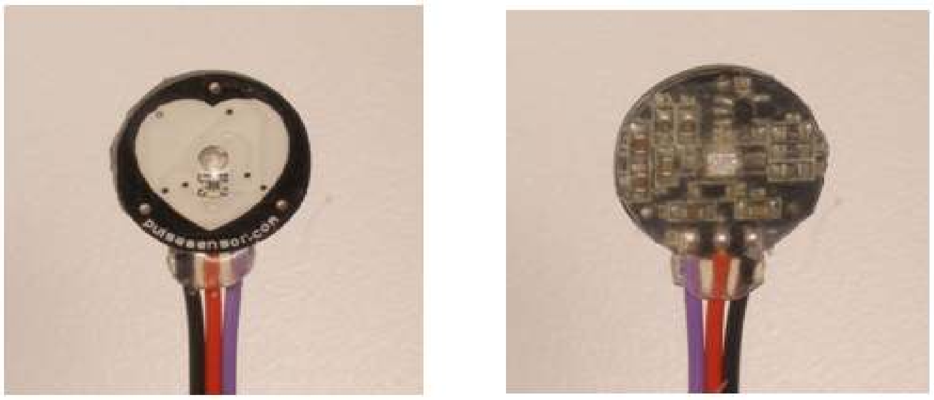


Figure 4.6(d)

This is the basic Pulse Sensor Hot Glue Seal, It’s also got the clear vinyl sticker on the front face. We’re calling this ‘Water Resistant’, ready to be handled and passed around from fingers to earlobes or whatever. It is not advised to submerge or soak the Pulse Sensor with this basic seal.

Now you can stick on the Velcro dot (included) and make a finger strap with the Velcro tape.

Attaching the Ear Clip

The earlobe is a great place to attach Pulse Sensor. Here's some instruction on how to do it. It is important to apply some strain relief to the cable connection where it meets the Pulse Sensor PCB. The little wire connections can wear out and break (or short on something) over time. We can do this with hot glue.

Attach a clear vinyl sticker to the front of the Pulse Sensor. Then, put a small dab of hot glue on the front of the cables right where they meet the PCB. Get some on the edge of the PCB too, that will help.

Next place the Pulse Sensor face down, and put a dab of glue about the size of a kidney bean on the back as illustrated above. Center the round part of the ear clip on the sensor and press it into the hot glue. The tallest component on the back is the plastic body of the reverse mount LED, and if you press it evenly it will help keep the metal of the ear clip from contacting any of the component connections.

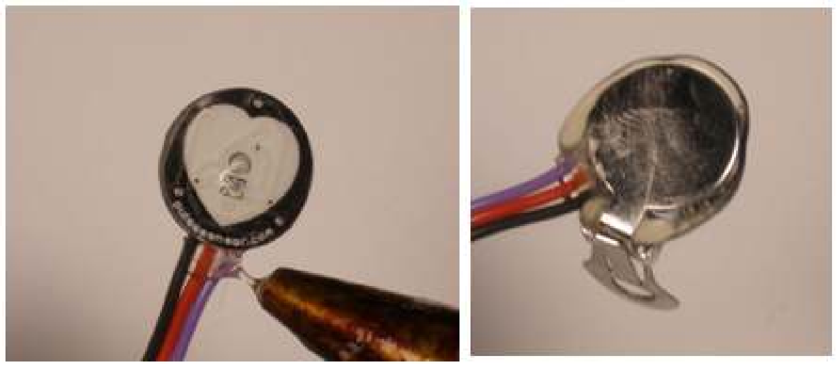


Figure 4.6(e)

Allow the hot glue to ooze out around the ear clip. That will ensure good coverage. Take care not to let the hot glue cover around the ear clip hinge, as that could get in the way of it working. Trimming is easy with nail clippers. Don't trim the wires by mistake.

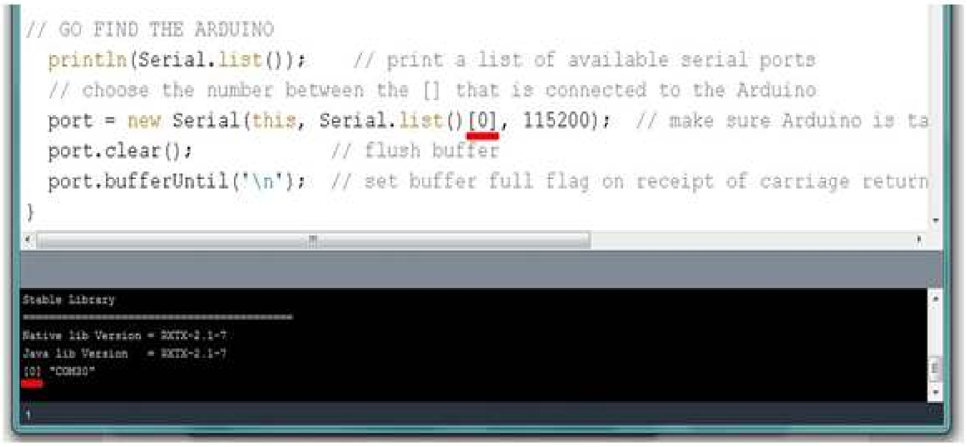
Figure4.6(f)

4.3.4 Trouble shooting:

Processing Sketch gives me a COM port error at startup.

Make sure you are plugged into an ESP32 board, that it is working correctly, and running our firmware.

Check to see if you have the right serial port. The code underlined in red should match the correct port number in the terminal window at the bottom of Processing IDE.



```
// GO FIND THE ARDUINO
  println(Serial.list());    // print a list of available serial ports
  // choose the number between the [] that is connected to the Arduino
  port = new Serial(this, Serial.list()[0], 115200);  // make sure Arduino is ta
  port.clear();              // flush buffer
  port.bufferUntil('\n');  // set buffer full flag on receipt of carriage return
}
```

```
Stable Library
=========================================
Native lib Version = RXTX-2.1-7
Java lib Version   = RXTX-2.1-7
[0] "COM30"
```

Processing gives an RXTX mismatch warning, then nothing happens

The RXTX mismatch problem can be resolved by making sure you are running the latest version of Processing and Arduino.

4.4 DHT11 Sensor

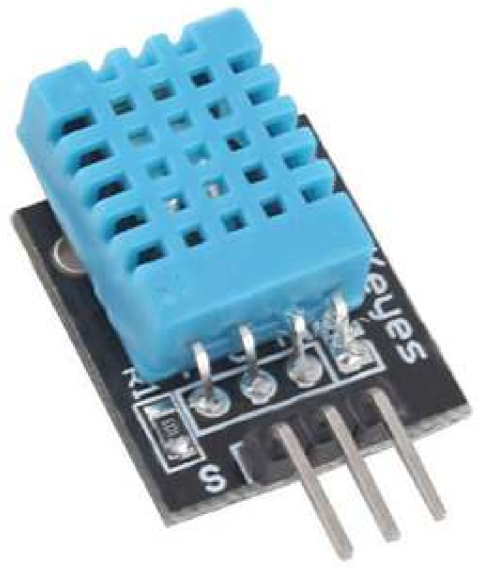


Figure 4.7 temperature and humidity sensor

4.4.1 Pin Identification and Configuration:

No:	Pin Name	Description
For DHT11 Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For DHT11 Sensor module		

1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

4.4.2 DHT11 Specifications:

- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: ±1°C and ±1%

Difference between DHT11 Sensor and module:

The DHT11 sensor can either be purchased as a sensor or as a module. Either way, the performance of the sensor is same. The sensor will come as a 4-pin package out of which only three pins will be used whereas the module will come with three pins as shown above.

The only difference between the sensor and module is that the module will have a filtering capacitor and pull-up resistor inbuilt, and for the sensor, you have to use them externally if required.

Where to use DHT11:

The DHT11 is a commonly used Temperature and humidity sensor. The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

How to use DHT11 Sensor:

The DHT11 Sensor is factory calibrated and outputs serial data and hence it is highly easy to set it up. The connection diagram for this sensor is shown below.

The data pin is connected to an I/O pin of the MCU and 5K pull-up resistor is used. This data pin outputs the value of both temperature and humidity as serial data.

The output given out by the data pin will be in the order of 8bit humidity integer data + 8bit the Humidity decimal data +bit temperature integer data + 8bit fractional temperature data +8bit parity bit.

4.4.3 Applications:

- Measure temperature and humidity
- Local weather station
- Automatic climate control
- Environment monitoring

4.6 Buzzer

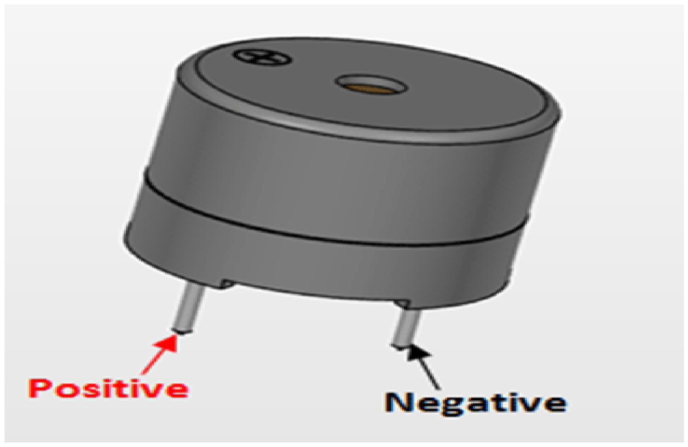


Figure4.8 buzzer

A **buzzer** is a small yet efficient component to add sound features to our project/system. It is very small and compact 2-pin structure hence can be easily used on breadboard, Perf Board and even on PCBs which makes this a widely used component in most electronic applications.

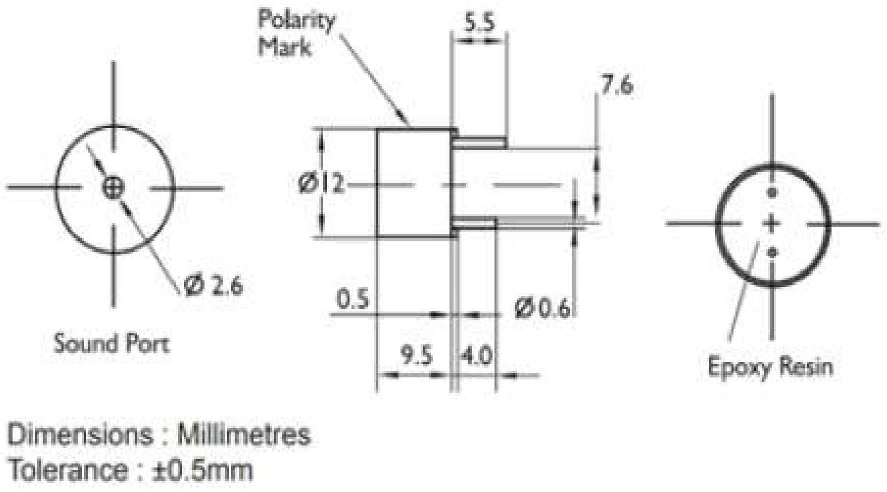


Figure 4.9 2D model of buzzer

There are two types are buzzers that are commonly available. The one shown here is a simple buzzer which when powered will make a Continuous Beeeeeeppp.... sound, the other type is called a readymade buzzer which will look bulkier than this and will produce a Beep. Beep. Beep. Sound due to the internal oscillating circuit present inside it. But, the one shown here is most widely used because it can be customised with help of other circuits to fit easily in our application.

This buzzer can be used by simply powering it using a DC power supply ranging from 4V to 9V. A simple 9V battery can also be used, but it is recommended to use a regulated +5V or +6V DC supply. The buzzer is normally associated with a switching circuit to turn ON or turn OFF the buzzer at required time and require interval.

4.5.1 Buzzer Features and Specifications

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz
- Small and neat sealed package
- Breadboard and Perf board friendly

4.5.2 Applications **Buzzer**

- Alarming Circuits, where the user has to be alarmed about something
- Communication equipments
- Automobile electronics
- Portable equipments, due to its compact size

Chapter 5

SOFTWARE

5.1 Google Firebase cloud

5.1.1 INTRODUCTION TO FIRE BASE

The broadening of the Internet and hardware advancements during the last decade have given birth to what is nowadays known as the reactive programming paradigm in the computer science field. Behind it, a single powerful reason appears and is challenging the ways of working of the software development industry around the world: *today, a single website handles as much traffic as the entire Internet just a few years ago* and this has implied that end users rely each time more on applications and also expect millisecond response times, 100% uptime (Bonér, Farley, Kuhn, & Thompson, 2016, p.2)

Objective and report structure

The following report aims to present the state-of-the-art of the concept of real-time databases and deep technical details about one of the most widely-used products using this new technology in real-world applications: Firebase. Altogether with the previous, a use case will also be presented to show a real example that implements Firebase as a database layer. Specifically, Firebase will be used to store tweets information coming directly from the Twitter API.

The report is structured as follows: first, an overall view of the traditional database management systems is presented, which will introduce the main weaknesses of these classical tools to deal with evolving collections, the main concept behind push-based databases. After a comparison between both approaches, real-time databases are introduced and some of the main vendors will be discussed. Later, an in-depth conceptualization will be made for Firebase database. Finally, a technical Firebase implementation will be shown followed by a short discussion and conclusion section.

5.1.2 Firebase use case

The objective of this section is to present a real-time application implementation using Firebase as the database layer.

The application uses Google Natural Language Processing (NLP) API to perform analysis of keywords and sentiments expressed in tweets obtained from the Twitter stream API. The processed data is stored in the firebase database which triggers updates in real-time in all connected clients devices. This implementation was developed based on the work done by the Google developer Sara Robinson (Robinson, 2017).

5.1.3 Application architecture

The architecture of the application consists on the following components: a client of Twitter stream API, a client of the NLP Google API, a Node server to perform the integration among the two external APIs and the Firebase database, and a web client which can be accessed by desktop and mobile devices to watch the updates in real-time.

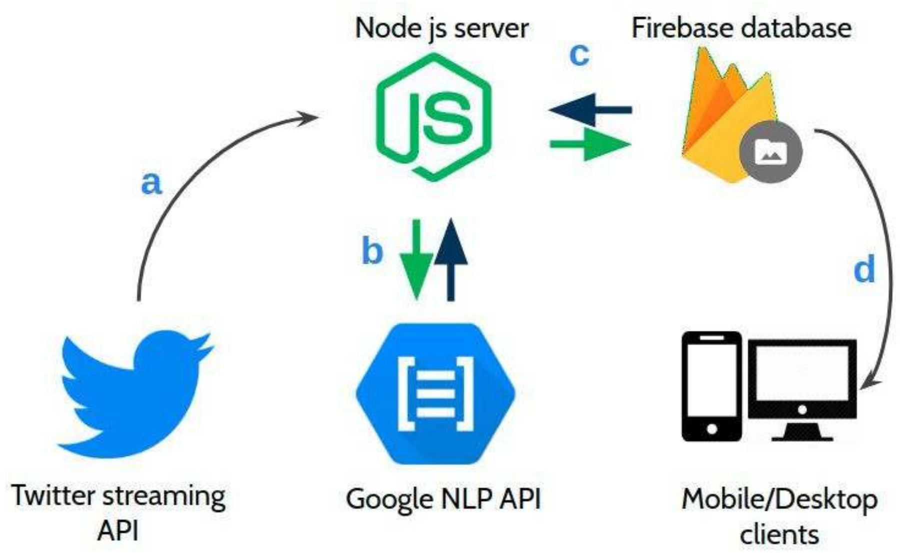


Figure5.1 Application architecture

Sentiment analysis

The Node server will permanently receive new tweets from the Twitter stream API (Figure 12.a) taking into account a set of specified filters. These filters could be a list of keywords which we desire to keep track of, the language of the tweets, etc.

After a new tweet that complies with the filter's criteria arrives, the Node server extracts the text from the tweet and sends it to the Google NLP service. The NLP service returns a set of properties related to the outcome of the tweet text analysis. Example of such properties are:

- The score of the prevailing emotion in the tweet (negative, positive or neutral) given a scale from -1 to 1.
- The list of symbols or tokens found after performing a syntactic analysis of the tweet text, describing if the identified token is a symbol, an adjective, verb, etc.

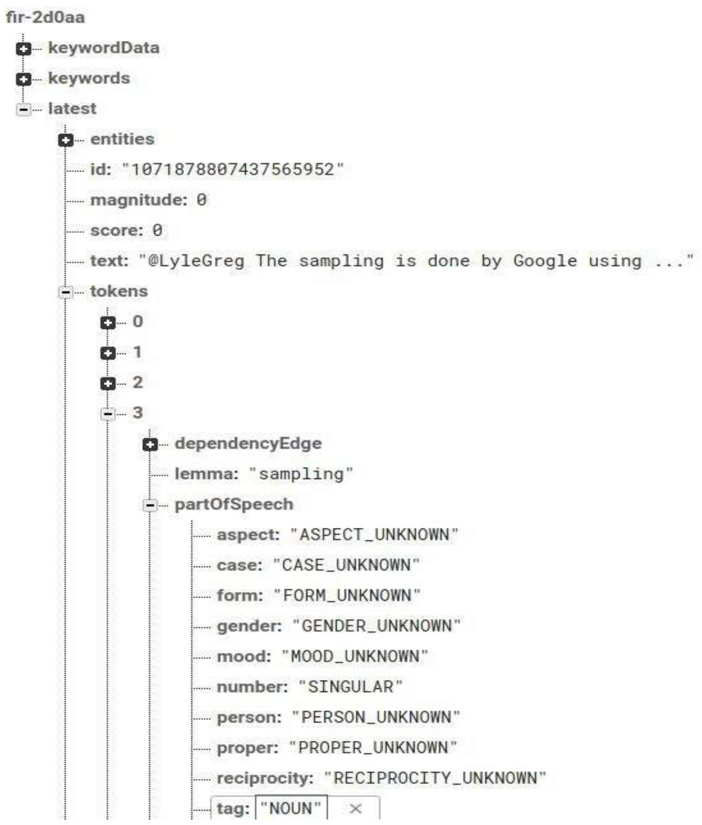
After obtaining the result of the tweet analysis through the NLP service, the resulting set of properties are saved into the Firebase database . In our example application, the properties are written in a child node of the root of the Firebase database, with the key “latest” which indicates the latest tweet that was processed, see Figure 13.

Firebase to raspberry pi logic

```
import pyrebase
import sys
#import Adafruit_DHT
import time

config = {
"apiKey": "AIzaSyBpBd9Nji8eJwaRZV7PRJM9DUpNHe58yyg",
"authDomain": "test123-85050.firebaseapp.com",
"databaseURL": "https://test123-85050.firebaseio.com",
        "storageBucket": "test123-85050"
   }
```

```
firebase = pyrebase.initialize_app(config)
db = firebase.database()
#print(db)
#db.child("/message").push(25)
db.update({"lan":74.85,"long":52.8})
values = db.child("long").get()
print(values.val())
```

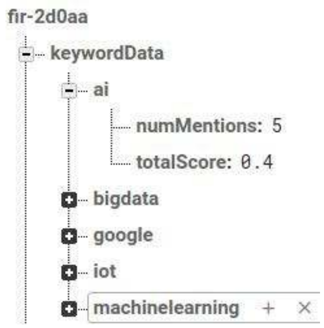


```
fir-2d0aa
  keywordData
  keywords
  latest
    entities
    id: "1071878807437565952"
    magnitude: 0
    score: 0
    text: "@LyleGreg The sampling is done by Google using ..."
    tokens
      0
      1
      2
      3
        dependencyEdge
        lemma: "sampling"
        partOfSpeech
          aspect: "ASPECT_UNKNOWN"
          case: "CASE_UNKNOWN"
          form: "FORM_UNKNOWN"
          gender: "GENDER_UNKNOWN"
          mood: "MOOD_UNKNOWN"
          number: "SINGULAR"
          person: "PERSON_UNKNOWN"
          proper: "PROPER_UNKNOWN"
          reciprocity: "RECIPROCITY_UNKNOWN"
          tag: "NOUN"
```

Figure5.1.1 Example of the Firebase database in our example application. The child-node ‘latest’ indicates the last tweet that was analyzed by the Google NLP service.

In order to show the real-time capabilities of Firebase, the Node server also performs an active listening on the “latest” node in our database. This means that every time this node is updated in the Firebase database with a new tweet information, a function is triggered in our Node server. In this function, the Node server retrieves the properties of the latest tweet, examines which of the tracked keywords are present in the tweet text, and the emotional scale (-1 to 1) associated with the tweet. Then, the Node server, writes a new child node with the name of the corresponding keyword inside the path “root”->“keywordData” in the database, with the following properties:

- numMentions: Number of times the keyword was found in the analyzed tweets until now.
- totalScore: The sum of the sentiment scores associated with the keyword in all the tweets that contained the keyword.



```
fir-2d0aa
 keywordData
   ai
     numMentions: 5
     totalScore: 0.4
   bigdata
   google
   iot
   machinelearning
```

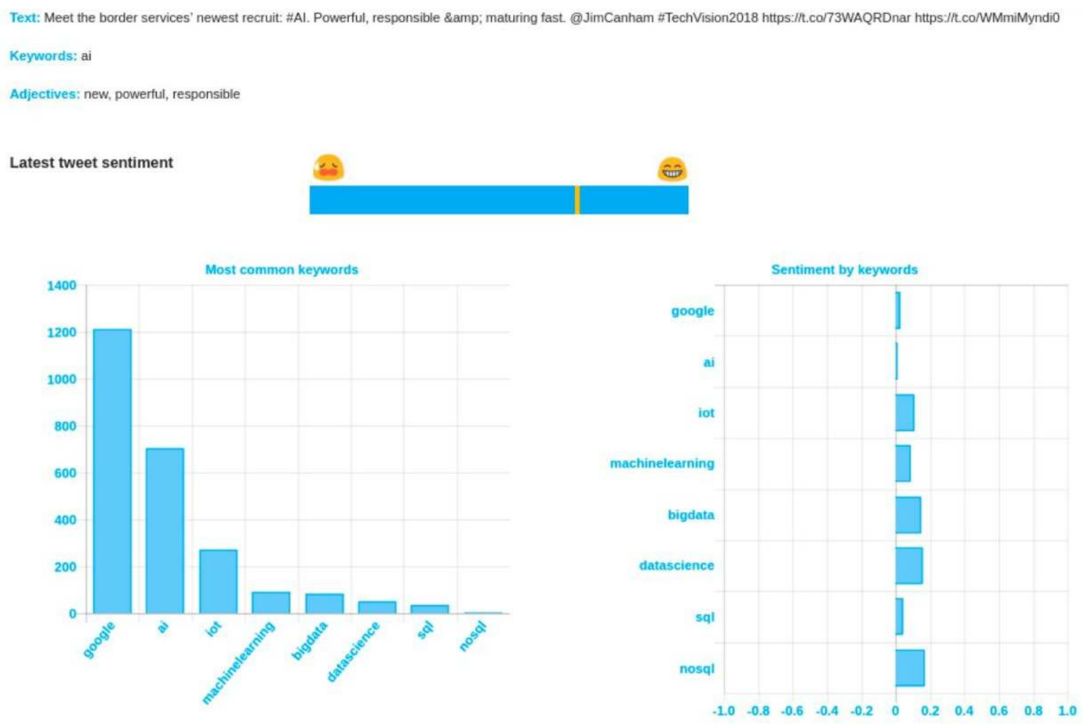
Figure 5.1.2 The keyword “ai” was found in 5 tweets already, and the sum of all the sentiments scores is 0.4 so far.

Visualization

Finally, mobiles and desktop clients can access the web application to visualize a dashboard related to the tweets The web application view is divided into 3 sectors.

In the upper part of the view, the latest tweet that matched one or more of our selected keywords is displayed with the corresponding adjectives that were found. The sentiment score of the tweet is displayed in a blue bar where the left extreme (sad face) represent negative sentiments (-1 in sentiment score), whereas the right extreme (happy face) represents positive sentiments (+1 in sentiment score). This section is updated instantly every time a new tweet is saved in the Firebase database.

The bottom left section is a graphic that represents the keywords found and the number of matches in all the analyzed tweets. The bottom right section is a graphic that represents the average of the sentiment score per keyword. Both graphics are updated in real-time when the Firebase database is updated, the graphics are sorted by the top 10 most mentioned keyword in the analyzed tweets. The web client view is shown in Figure 15.


Text: Meet the border services' newest recruit: #AI. Powerful, responsible &amp; maturing fast. @JimCanham #TechVision2018 https://t.co/73WAQRDnar https://t.co/WMmiMyndi0

Keywords: ai

Adjectives: new, powerful, responsible

Latest tweet sentiment

Figure 5.1.3 Web application updated in real-time with every update in the Firebase database.The complete code with the corresponding instruction on how to execute the application can be found in the following git repository: https://github.com/srpablino/ml-talk-demos

5.2 MIT App Inventor

MIT App Inventor is an intuitive, visual programming environment that allows everyone even children to build fully functional apps for smart phones and tablets. Those new to MIT App Inventor can have a simple first app up and running in less than 30 minutes. And what's more, our blocks-based tool facilitates the creation of complex, high-impact apps in significantly less time than traditional programming environments. The MIT App Inventor project seeks to democratize software development by empowering all people, especially young people, to move from technology consumption to technology creation.

App Inventor lets you develop applications for Android phones using a web browser and either a connected phone or emulator. The App Inventor servers store your work and help you keep track of your projects.

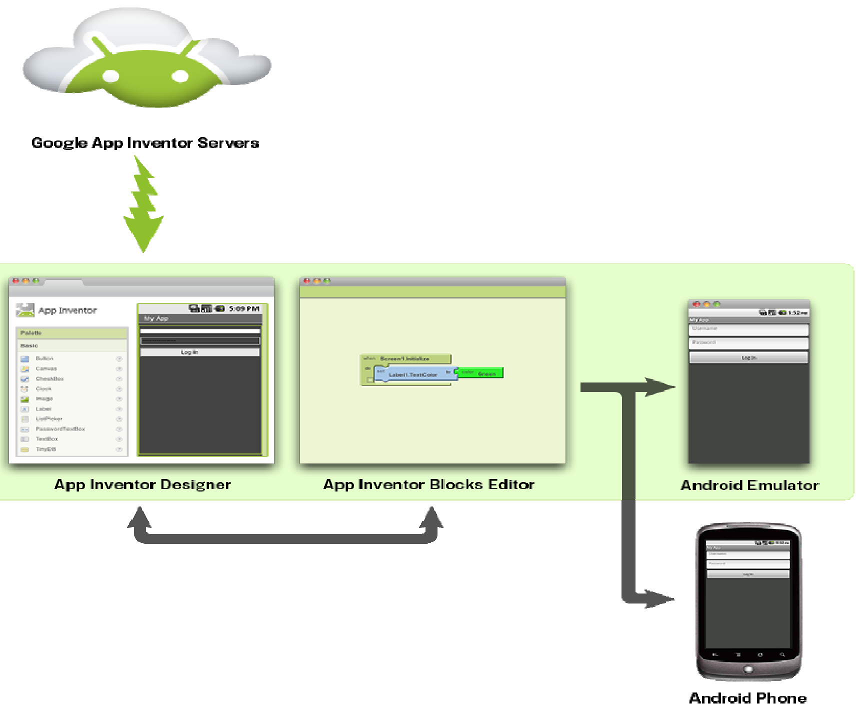


Figure5.2 Application architecture

You build apps by working with:

The App Inventor Designer, where you select the components for your app.

The App Inventor Blocks Editor, where you assemble program blocks that specify how the components should behave. You assemble programs visually, fitting pieces together like pieces of a puzzle.

Your app appears on the phone step-by-step as you add pieces to it, so you can test your work as you build. When you're done, you can package your app and produce a stand-alone application to install.

If you don't have an Android phone, you can build your apps using the Android emulator, software that runs on your computer and behaves just like the phone.

The App Inventor development environment is supported for Mac OS X, GNU/Linux, and Windows operating systems, and several popular Android phone models. Applications created with App Inventor can be installed on any Android phone. (See system requirements.)

Before you can use App Inventor, you need to set up your computer and install the App Inventor Setup package on your computer.

Chapter 6

RESULTS AND DISCUSSION

PRACTICAL DESIGN MODEL OF IOT SMART MONITORING SYSTEM

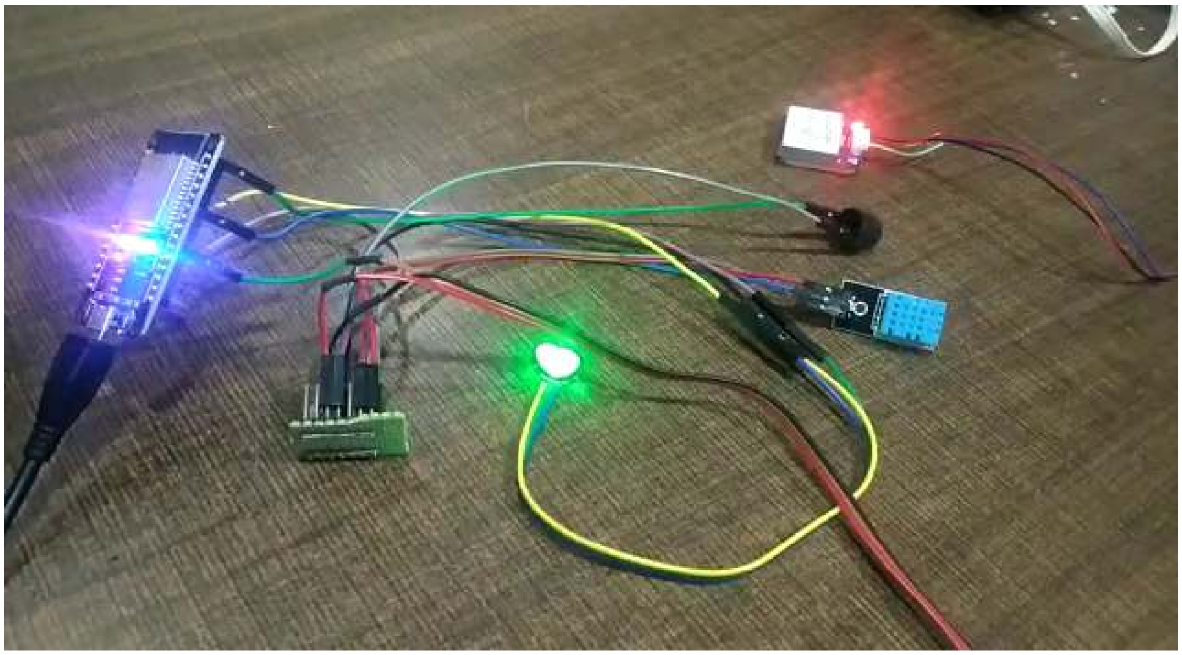
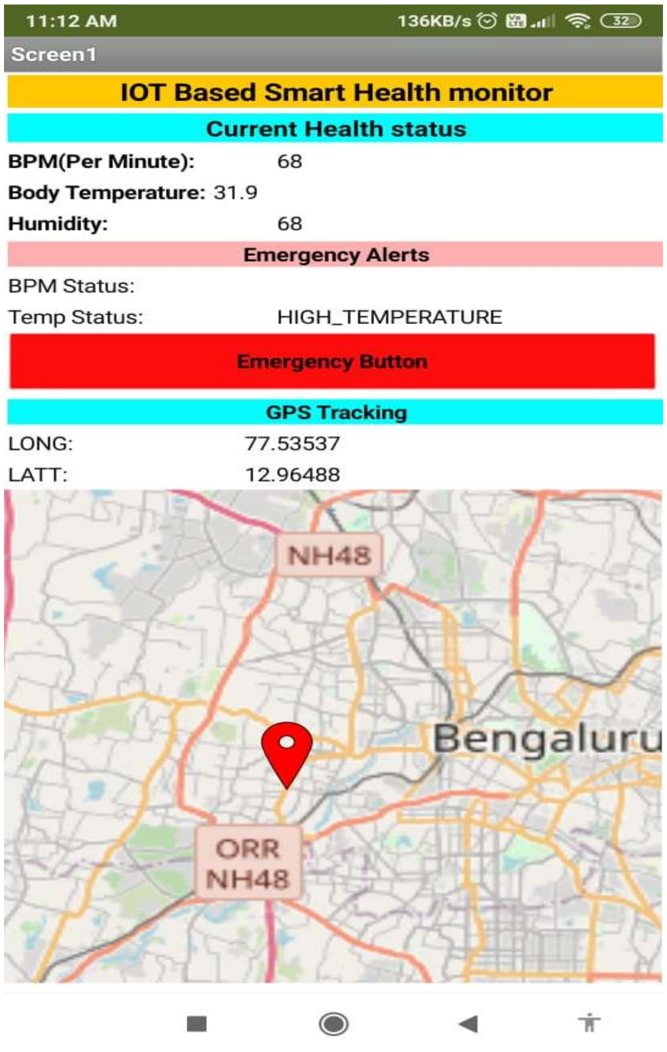


Figure 6.1 IOT smart monitoring system.

The above shows the practical design model of IOT smart monitoring system. Firstly the sensor captures the sensor data; analyze the data based on the parameters like heart rate, temperature, and humidity providing a feedback to doctors and the family members.



11:12 AM 136KB/s

Screen1

IOT Based Smart Health monitor

Current Health status

BPM(Per Minute): 68
Body Temperature: 31.9
Humidity: 68

Emergency Alerts

BPM Status:
Temp Status: HIGH_TEMPERATURE

Emergency Button

GPS Tracking

LONG: 77.53537
LATT: 12.96488

Figure 6.2 "SMART HEALTH" User Panel

The circuit is first installed on a board where necessary tests and trials were carried out. The inter beat interval (IBI) can be calculated using the PPG wave obtained from the pulse sensor. Microcontroller code is set to measure the IBI with the timing between moments when the signal crosses 50% of the wave amplitude during this fast upward ramp (PT), as shown in Figure 10. BPM is an average value from every previous 10 IBI beat value.

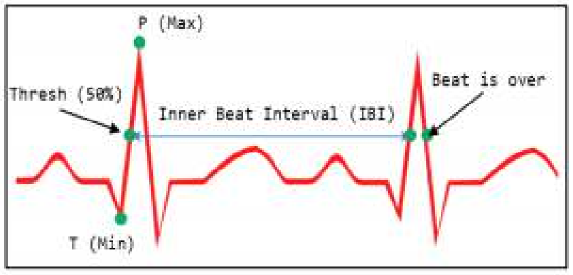


Figure 10. PPG signal for two consecutive heartbeats

High resolution sampling is very important to ensure that the timing between heartbeats is accurately measured. For this, Timer2 interrupt is set to 2ms. In this way a timing resolution with a frequency of 500 Hz is obtained. The analog output of the Pulse Sensor is then read when each interrupt occurs. The highest (P) and lowest (T) values of the PPG signal must be determined in order to obtain a correct amplitude measurement. The thresh variable is initialized at 512 and changes over the course of the study to find the point corresponding to 50% of the amplitude. In order to avoid noise and false readings, the time corresponding to 3/5 of the IBI value must be passed before the signal is updated. If the threshold value is greater than 50% of the signal amplitude and the time period is greater than 3/5 of the IBI, the IBI values are stored in a 10-element sequence.

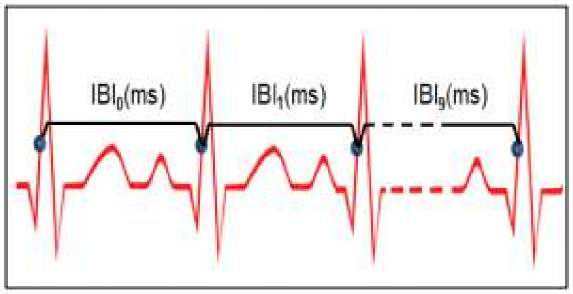


Figure 11. PPG signal for consecutive heartbeats

$$IBI_{avg} = \frac{(IBI_0 + IBI_1 + \cdots + IBI_9)}{10} \qquad (1)$$

$$BPM = \frac{60000}{IBI_{avg}} \qquad (2)$$

As shown in Figure 11, after a total of 10 consecutive IBI registrations, the BPM value is obtained in ms by taking the average of the total IBI values. The formulas for calculating the IBI value are given in equations 1 and 2. Thanks to the SMART HEALTH application installed on the mobile phone, all vital data sent wirelessly by Arduino Pro Mini can be displayed in real time. In Figure you can see the "SMART HEALTH" android interface.

Chapter 7

Applications and Advantages

7.1 Applications:

- This Project is mainly designed for elderly person in our home.
- Disable patients can use this project, when difficult to go to doctors on Daily basis.
- It is helpful when patient need to go for some emergency work.

7.2 Advantages:

- By this project we get Improved Treatment.
- Faster disease Diagnosis can be done.
- Proactive Treatment.
- We can reduce the Errors.
- It requires fewer components so its cost is low.
- Real time translation.
- It is very flexible to users.
- Patient health parameter data is stored over the cloud.

Chapter 8

CONCLUSION

Today, many services can be reached with internet technology and the number of applications that use this technology is constantly increasing. IOT technology is expanding day by day to include different sectors and applications. One of them is the smart health sector and this sector offers incredible opportunities for us with new applications. The monitoring of the patients, who should be kept under constant surveillance, in the hospital environment is very difficult with the existing infrastructure and methods. Patients under surveillance in hospitals are dependent on bedding and this makes the patients uncomfortable. Many health problems that require early diagnosis may cause vital problems for the patient because they cannot be monitored on time. Early diagnosis is unbelievably important for patients with heart disease.

In this study, a wireless patient monitoring system is developed that allows patients to be mobile in their social areas. The developed system continuously measures the heart rate and body temperature of the patient and provides monitoring and tracking through an android based interface. When the patient's vital data reaches a predetermined limit value, the mobile application alerts the patient and the people in the vicinity. This warning is made at a volume level that people near the patient can hear. If there is nobody in the vicinity of the patient who can help him, the patient's heart rate, body temperature, and coordination information are sent to family members and the doctor as e-mail and twitter notifications. The main purpose of the device is to make provide that they get medical aid as soon as possible, in case of a possible discomfort for heart diseases. So there will be an increased chance of survival of patients.

REFERENCES

1. Design and Implementation of a Wearable Sensor Network System for IoT-Connected Safety and Health Applications Fan Wu ;Taiyang Wu ; Mehmet RasitYuce IEEE 2019
2. Smart Health Monitoring System through IOT S. Ananth ; P. Sathya ; P. Madhan Mohan IEEE 2019
3. Smart and Wearable ECG monitoring system as a Point of Care (POC) device AbhinayVishwanatham ; Narendra Ch. ; S.R. Abhishek ; Ramakrishna Chaitanya R. ; Siva Sankara Sai Sanaga IEEE 2019
4. An IoT-based wearable system using accelerometers and machine learning for fetal movement monitoring Xin Zhao ;Xianyi Zeng ; LudovicKoehl ; Guillaume Tartare ; Julien de Jonckheere ; Kehui Song IEEE 2019
5. Smart secure system for human health monitoring R Nikhil Nair ; K. A. Kiran IEEE 2019
6. Pervasive Healthcare Monitoring System Mario Nava Perez ; Francisco J. Alvarez Mata ; Victor M. Zamudio Rodriguez ; Shumei Zhang IEEE 2018
7. Ashton, K, That 'internet of things' thing, RFID journal, 2009, 22(7), 97-114.
8. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M, Internet of Things (IoT): A vision, architectural elements, and future directions, Future generation computer systems, 2013, 29(7), 1645-1660.
9. Evans, D., The internet of things: How the next evolution of the internet is changing everything, CISCO white paper, 2011, 1, 1-11.
10. Ejaz, W., Naeem, M., Shahid, A., Anpalagan, A., Jo, M., Efficient energy management for the internet of things in smart cities, IEEE Communications Magazine, 2017, 55(1), 84- 91.

11. Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., Liljeberg, P., Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach, Future Generation Computer Systems, 2018, 78, 641- 658.
12. Thibaud, M., Chi, H., Zhou, W., Piramuthu, S., Internet of Things (IoT) in high-risk Environment, Health and Safety (EHS) industries: A comprehensive review, Decision Support Systems, 2018, 108, 79-95.
13. Ayón, C., Unpacking Immigrant Health: Policy, Stress, and Demographics, Race and Social Problems, 2018, 1-3.
14. Chaudhary, R., Jindal, A., Aujla, G. S., Kumar, N., Das, A. K., Saxena, N., LSCSH: Lattice-Based Secure Cryptosystem for Smart Healthcare in Smart Cities Environment, IEEE Communications Magazine, 2018, 56(4), 24-32.
15. Kakria, P., Tripathi, N. K., Kitipawang, P., A real-time health monitoring system for remote cardiac patients using smartphone and wearable sensors, International journal of telemedicine and applications, 2015, 8.
16. Majumder, S., Mondal, T., Deen, M. J., Wearable sensors for remote health monitoring, Sensors, 2017, 17(1), 130.
17. Saha, J., Saha, A. K., Chatterjee, A., Agrawal, S., Saha, A., Kar, A., Saha, H. N., Advanced IOT based combined remote health monitoring, home automation and alarm system. In Computing and Communication Workshop and Conference, IEEE 8th Annual, 2018, pp 602-606.