EAST WEST UNIVERSITY

An Intelligent Cardiac Monitor Using Internet of Things (IoT)

By

Asadujjaman

ID: 2011-3-60-017

Supervised by

Former Senior Lecturer

K.M. Imtiaz-Ud-Din

Dept. of CSE at East West University

The project has been submitted to the Department of the Computer Science & Engineering at East West University in the partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering.

May, 2016

DECLARATION

The project has been submitted to the Department of the Computer Science & Engineering. East West University in the partial fulfillment of the requirement for the degree of Bachelor of Science in CSE performed by me under supervision of Former Senior Lecturer K.M. Imtiaz-Ud-Din, Dept. of CSE at East West University. This is also needed to certify that, the project work is under the course ‘Project Work (CSE-499)’. I, hereby, declare that this project has not been submitted elsewhere for the requirement of any degree or diploma or any other purposes.

Signature of the candidate

(Asadujjaman)

Abstract

In this project, some IoT device will help user to maintain a proper diet chart during they taking their meal. There is a device which can measure human heart rate and after measuring this user can send heart rate data to a server, after they can read those data and get more information from server by an application. This application is develop for android devices. This whole thing is based on IoT architecture. Using IoT architecture I will ensure this service to my end user.

Asadujjaman

Letter of Acceptance

This project is submitted By Asadujjaman, Id : 2011-3-60-017 to the department of Computer Science and Engineering, East West University, Dhaka Bangladesh is accepted as satisfy for the partial fulfillment of the of the requirement for the degree of Bachelor of Science

Computer Science and Engineering on May 8, 2016.

Board of Examiners

ACKNOWLEDGEMENTS

It has been great pleasure for me to develop IoT related service for Heart Patient. I have gathered sufficient knowledge and experience during this project.

Most Importantly I would like to thank my honorable teacher and supervisor, Former Senior Lecturer K.M. Imtiaz-Ud-Din, Dept. of CSE at East West University. Who guided me to proper analysis of the system and helped to develop an elegant and efficient system. It was a great pleasure to study and work with many gifted people who influenced us in many ways. First of all, I would like to thank my project Supervisor K.M. Imtiaz-Ud-Din. He encouraged me to seek out the clearest and deepest description of theoretical ideas as well as experimental findings. I am very grateful to him for his continuous support, advice and guidance.

Finally I would like to convey my Special thanks to my family and friends whom have always given me tremendous Support. Without their love and encouragement I would not had achieve this far.

Asadujjaman

➢ Table of Content

Chapter 1

Introduction

1.1 What is IOT?

The "Internet of things" (**IOT**) is becoming an increasingly growing topic of conversation both in the workplace and outside of it. It's a concept that not only has the potential to impact how we live but also how we work. But what exactly is the "Internet of things" and what impact is it going to have on us, if any? There are a lot of complexities around the "Internet of things" but I want to stick to the basics. Lots of technical and policy-related conversations are being had but many people are still just trying to grasp the foundation of what the heck these conversations are about.

Let's start with understanding a few things.

Broadband Internet is become more widely available, the cost of connecting is decreasing, more devices are being created with Wi-Fi capabilities and sensors built into them, technology costs are going down, and smartphone penetration is sky-rocketing. All of these things are creating a "perfect storm" for the **IOT**.

This is the concept of basically connecting any device with an on and off switch to the Internet (and/or to each other). This includes everything from cellphones, coffee makers, washing machines, headphones, lamps, wearable devices and almost anything else you can think of. This also applies to components of machines, for example a jet engine of an airplane or the drill of an oil rig. As I mentioned, if it has an on and off switch then chances are it can be a part of the **IOT**. The analyst firm Gartner says that by 2020 there will be over 26 billion connected devices… That's a lot of connections (some even estimate this number to be much higher, over 100 billion). The **IOT** is a giant network of connected "things" (which also includes people). The relationship will be between people-people, people-things, and things-things [1].

We are using internet in our daily life. We are all connected through the internet. But the new internet is not just only connecting people but it's all about connecting **things**. Which means all **things** are connected through the internet. So its name is Internet of **things**.

Now the question is what actually **things** are? Here **things** mean only those stuffs which has a **sensor**, which can communicate with other **things** through the **internet** and can share information during that time.

Basically when a human tries to communicate with other human, he/she needs five types of sensors those are touch, taste, smell, hearing and sight. Those sensors helps human to communicate with others. Similarly different types of **things** has different types of sensors.

Using those sensors they can take percepts from environment and communicate with other **things** through internet.

Figure 1.1: Internet of Things [9]

1.2 Purpose of the project

Here I am trying to implement an application for heart patient using IOT Architecture. My main targeted group of people for this project is "Heart Patients". Every Heart patients always have to maintain some sort of diet plan for their meal. This project will help them to maintain their diet plan more strictly and properly. And that is my main goal of this project. I developed this application for them. The device and my application will help patient to measure their heart rate easily and after that, their relatives or him/herself will get a suggestion for further instruction or more information about emergency treatment. And you have to use a mobile app to get this suggestion. And I also developed this mobile application.

1.3 Objective

To help human to lead a healthy life is the main objective of my Project.

- Help human to lead a life with proper guideline.
- Help heart patients to keep in touch with relatives and doctors.
- Ensure To keep heart rate in normal state.
- At last, to help them to maintain a certain meal plan to save them from sudden heart attack.

1.4 Scope

Following are the scope of the developed system:

For User:

1. To use this service user has to complete their registration process. In registration user need to use their phone number as a “username”.
2. The device will be configured by this username or phone number.
3. User need to use Wi-Fi network to run this device, and for the limitation user need to use specific SSID and Password which has also configured in the device.
4. User also need to use username and password for login process.
5. After successful login they will get some information about what should they do for maintaining a diet plan.

For Management:

1. Only **Management** has the control, so he can configure that device by using username or phone number.

1.5 Benefits for user and medical science view:

The Followings are the benefits of the user and admin

User View: This service will help a heart patient to maintain a diet chart before taking meal or medicine. By using this service patient will be monitored by their relatives and doctors also.

Medical Science View: This may help medical science to provide better service to their heart patient.

Chapter 2

State of the Art

2.1 Internet of things: 4 free platforms to build IoT project

Build IoT project is a process that could involve the step of finding free platforms. As we know, Internet of things is a set of physical objects that use network support to exchange data. These objects can be sensors, software, boards and so on. This is an interesting ecosystem where the software can be connected directly to real hardware or devices. The most known board for building internet of things project are Arduino (with its several versions) and Raspberry. Integrating these devices with cloud platforms is possible to collect and analyze data, create "smart" object that can be controlled remotely. One way to control such devices is using smartphones like Android and iOS devices. Dev Boards like Arduino or Raspberry are cheap and everyone can experiment IoT projects. *Cloud IoT platforms help developers and maker to build IoT project and test them fast and easily*. [2]

2.2 IoT Platforms analysis for building projects

Cloud IoT platforms provides several kind of services that can be very useful in building **IoT project**:

- cloud data store data
- Event logic
- Platforms integration

Cloud data store enables developers to store data sent from different board (like Arduino or Raspberry); for example, it is possible to store values read from a sensor. This information can be visualized using a graph or analyzed with other tools.

Event logic is web based programming logic that can be used to trigger some action when an event occurs. Using this kind of platforms is possible to implement some "business logic" using just a web interface without knowing much about the board we are using for the project. Usually, the logic is like IF-THEN, for example if an event occurs then do this action. An event can be a signal read from a sensor and the action can be an email or a SMS.

Platforms integration is a set of "adapters" that implements specific protocol so that it is possible without writing a line of code mix different internet services to make a chain of actions. For example, using Arduino with Ethernet shield is possible to send an alert via SMS when a value read from a sensor is higher than a threshold level [2].

2.3 Build IoT project with free platform: Description

Below we can find a list of **IoT Cloud platforms** that can be useful to create **IoT projects** with a brief description.

Temboo: This is a very interesting platform that provides services to integrate Arduino, Raspberry and other platforms with different internet services (like SMS, Email and so on). This platform uses **choreos** that are connectors toward external services, so that events in Arduino, like sensor signals, can be transformed in different kind of events. Moreover it provides some logic like IF-THEN.

CarrIoTs: This is another interesting platform that enables smart devices to store data. It uses the data stream concept to enable devices to send data. This platform moreover has a rule management system so that you can implement custom logic directly on the web. It can moreover send Email, SMS and Twitter messages

NearBus: This proposes a different approach respect to other platforms. Usually the basic concept that stands behind IoT platforms is connecting the device (Arduino, Raspberry and so on) to the cloud so that these boards can send data. NearBus provides a different way: it maps the device into the cloud so that it gets a part of the cloud itself. It uses an **Agent** to accomplish this task and it is possible to control this agent directly from the web using a set of API.

Ubidots: This platforms support several kind of board and can be used to store data in the cloud. It offers data capture, data visualization with a built-in dashboard, rules management (or event management). With the built-in dashboard, it is possible to see in real time the graph built on the data sent by the device. It supports several kind of visualization.

Platform	Data store	Services integration	Data visualization	SDK API	Event/rule mngt	Free account
Temboo	No	Yes (about 2000 choreos)	No	Yes	Yes	Yes
CarrIoTs	Yes	Yes(Email, SMS, Twitter)	No	Yes	Yes	Yes
NearBus*	No	No	No	Yes	Yes	Yes
Ubidots	Yes	Yes(Email, SMS, Twitter, Web service)	Yes	Yes	Yes**	Yes

*Nearbus offers a different approach so it is quite difficult to categorize it
**It offers a set of API easy to use

The table above summarizes some of aspects of these platforms, that i think they are important. The aim of this comparison is to provide some high-level information about existing **IoT platforms** [2].

2.4 Framework in IOT

Bluemix is the latest cloud offering from IBM®. It enables organizations and developers to quickly and easily create, deploy, and manage applications on the cloud. Bluemix is an implementation of IBM's Open Cloud Architecture based on Cloud Foundry, an open source Platform as a Service (PaaS). Bluemix delivers enterprise-level services that can easily integrate with our cloud applications without needing to know how to install or configure them.

- What is Cloud Foundry?

Cloud Foundry is an open source platform as a service (PaaS) that lets us quickly create and deploy applications on the cloud. Because of its open source roots, Cloud Foundry is not vendor specific and does not lock us into proprietary software or cloud infrastructure. Cloud Foundry abstracts the underlying infrastructure needed to run a cloud, letting us focus on the business of building cloud applications. The beauty of Cloud Foundry is that it provides choice. Developers and organizations can choose:

Development Frameworks: Cloud Foundry supports Java™ code, Spring, Ruby, Node.js, and custom frameworks.

Application Services: Cloud Foundry offers support for MySQL, MongoDB, PostgreSQL, Redis, RabbitMQ, and custom services.

Clouds: Developers and organizations can choose to run Cloud Foundry in Public, Private, VMWare and OpenStack-based clouds.

Cloud Foundry's ability to provide choice comes through buildpacks, a convenient way to package frameworks and runtimes. Buildpacks can be community based, custom built, or built from scratch. In other words, if we cannot find a framework or service buildpack that suits our needs, we could modify an existing buildpack or create our own. By using buildpacks, companies are able to provide enterprise-level services like the Bluemix cloud offering.

- What is Bluemix?

Bluemix is an implementation of IBM's Open Cloud Architecture, based on Cloud Foundry, that enables us to rapidly create, deploy, and manage our cloud applications. Because Bluemix is based on Cloud Foundry, we can tap into a growing ecosystem of runtime frameworks and services. In addition to providing additional frameworks and services, Bluemix provides a dashboard for us to create, view, and manage our applications and services as well as monitor our application's resource usage. The Bluemix dashboard also provides the ability to manage organizations, spaces, and user access.

Bluemix provides access to a wide variety of services that can be incorporated into an application. Some of these services are delivered through Cloud Foundry. Others are delivered from IBM and third party vendors. New and enhanced services are added to the catalog often.

Some of the commonly used runtimes are:

- Node.js
- PHP
- Python
- Ruby

2.5 My Project's Architecture

In my project I did not use any specific framework or any platform. I just wanted to keep it simple. Though I might follow a basic architecture.

Here is my project architecture:

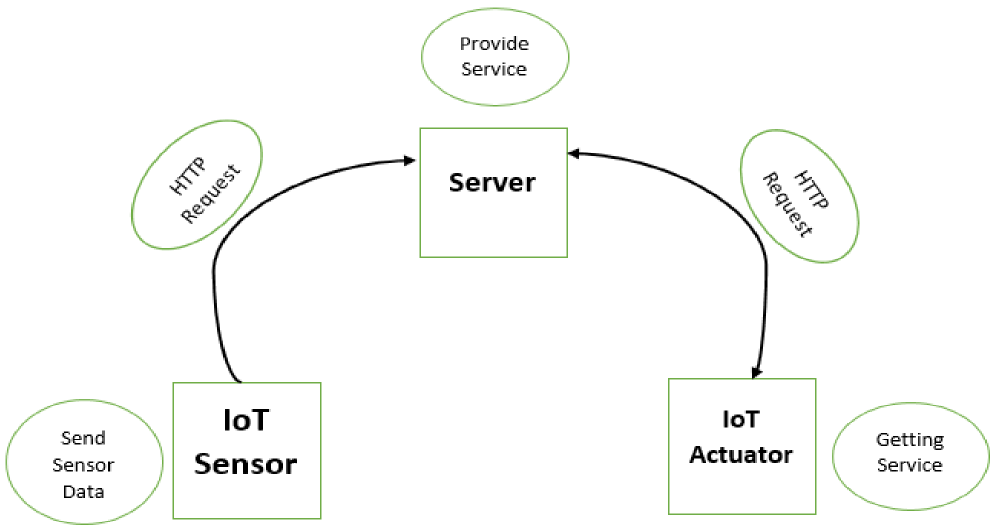


Fig 2.4: Project Architecture

There is a sensor in my Project which can detect human heart rate, then it will send heart rate data to a server. Those data will be saved in a database. Here I am using http get request to send data from the sensor.

There is also a mobile application in my project, where user can see their heart rate related information. For that they need to login by using there **username** and **password.** Heart rate related information is provided by the server, for that I need to send http get and post request to that server. Based on those request server will provide some service to user, and user will get that service by using that mobile application.

So this is the architecture which I am following to develop my IoT based project, where I did not use any framework or any specific platform. Anyone can develop any kind of IoT based project following this architecture and this is the advantage of my project.

2.6 About IoT Related Projects

The Internet of Things (IoT) is defined in many different ways, and it encompasses many aspects of life from connected homes and cities to connected cars and roads, roads to devices that track an individual behavior and use the data collected for push services. Some mention one trillion Internet-connected devices by 2025 and define mobile phones as the eyes and ears of the applications connecting all of those connected things. By these internet of things billions objects can communicate over worldwide over a public, private internet protocol network In 2010, the number of everyday physical objects and devices connected to the Internet was around 12.5 billion.

Smart cities, Smart cars, Public safety, Smart Industries and Environmental Protection has been given the high intention for future protection by IoT Ecosystem .For the development the government of Europe, Asia and America has consider the Internet of Things has area innovation and growth. Many visionaries have seized on the phrase Internet of Things to refer to the general idea of things, especially everyday objects, that are readable, recognizable, locatable, addressable, and/or controllable via the Internet, irrespective of the communication means (whether via RFID, wireless LAN, wide- area networks, or other means).

Radio Frequency Identification (RFID) and sensor network technologies will rise to meet this new challenge, in which information and communication systems are invisibly embedded in the environment around us. This results in the generation of enormous amounts of data which have to be stored, processed and presented in a seamless, efficient, and easily interpretable form. This model will consist of services that are commodities and delivered in a manner similar to traditional commodities.

Due to internet of things hospitals are shifting to remote self-monitoring for patients. Due self-monitoring it gives the patient greater freedom and independence for their health and free the equipment for emergency propose for patients. Internet of Things (IoT) is a new revolution of the Internet. Internet of Things (IoT) is can be said the expansion of internet services. It provides a platform for communication between objects where objects can organize and manage themselves. It makes objects themselves recognizable. The internet of things allows everyone to be connected anytime and anywhere. Objects can be communicated between each other by using radio frequency identification (RFID), wireless sensor network (WSN), Zigbee, etc. Radio Frequency identification assigns a unique identification to the objects. RFID technology is used as more secure identification and for tracking/locating objects, things, vehicles.

2.7 Projects based on IoT

Here I am going to talk about some existing project about IoT. Which are pretty similar to my project.

2.7.1 <u>Smart Security Solutions based on Internet of Things (IoT)</u>

Here is the first example; this project is all about controlling a security system. This project is developed by “Chirag M. ShahȦ*, Vamil B. SangoiȦ and Raj M. VishariaȦ” and they are from

"Electronics and telecommunication Engineering Department, D.J.Sanghvi College of Engineering, Vile Parle, Mumbai-4000056, India".

Summary of this Project:

With increasing demand from the industry for better access control systems, this paper was an attempt to make the conventional access control systems smarter and thereby decreasing the risks of breaking in into the places where these access control systems will be installed. EKTM4C123GXL is the development board which is used. Data from RFID reader and Biometric sensors are serially transmitted to the microcontroller. If valid fingerprint data or valid card no. is received, the microcontroller sends a signal to the Wi-Fi module1. The Wi-Fi module2 present at the door receives that signal and trips the relay according to the signal received. This is how the door opens. Also the Wi-Fi module sends a signal to the PC via the same Wi-Fi network. Hence the logs of people trying to access the door are maintained in the PC.

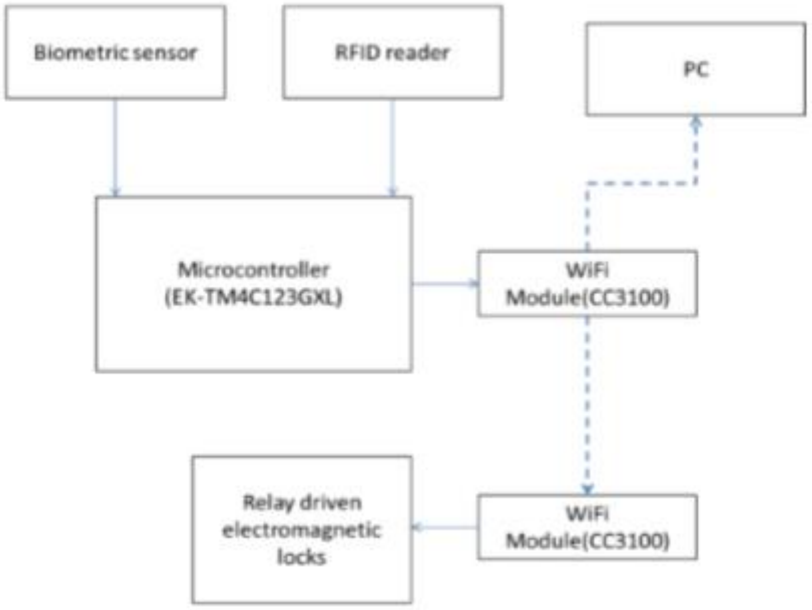


Figure 2.6.1.1: Block Diagram [4]

The smart card of the person is read by the RFID reader or wiegand reader near the door. The reader typically transmits a signal of 125 KHz. The card is a passive component with no power source. When it comes in proximity of the reader, the reader induces some voltage and hence the card transmits a unique 16 bit card number to the reader. The reader then transmits this card number to the microcontroller via the two data pins (D0, D1). Wiegand protocol is used for transmission. The 26 bit wiegand format is shown below. In the figure 4.2.1.2, the 1st bit is the even parity bit. This even parity is for the first 13 bits. This bit is followed by the 8 bit facility code (0-255). The facility code provides on more layer of security. This code is used in cases where the employees of 2 companies have the same card number. But they can be differentiated with the help of the facility code. The 8 bit facility code is followed by a 16 bit card number (0-65535). The last bit is the odd parity bit. The odd parity bit accounts for bits 1426. [4]

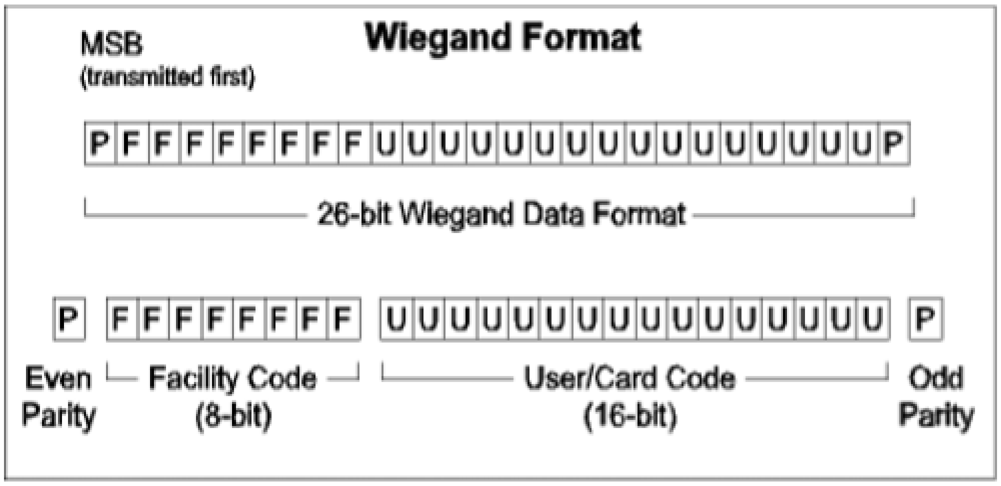


Figure 2.6.1.2: Wiegand Protocol [4]

2.7.2 <u>Intelligent Healthcare Service by using Collaborations between IoT Personal Health Devices</u>

Here is another example, which is about Health Care service using IoT. This project is developed by “ByungMun Lee* and Jinsong Ouyang” and they are from “Dept. of Computer Science, Gachon University, Korea, Dept. of Computer Science, California State University Sacramento, USA”. [6]

Summary of this Project:

Ubiquitous health (UH) service was a model in which individual medical data was measured by a ubiquitous personal health device (UHD), and then sent to the health server to provide feedback to medical experts and patients. Thus, most researches were focused on the function of sending the measured biomedical data to the server. Due to this reason, the analysis and processing function of medical data were mostly conducted in the server.

As the concept of IoT (Internet of Things) was introduced, researches which attempt to apply the IOT model in different fields are being progressed. If IoT technique is applied to UH, then UHD will break away from the simple functions of indicating measured data and sending them to the server and execute autonomous information exchange with neighboring systems (UHDs, gateway, server) and provide comprehensively assessed feedback immediately to the patient. For instance, a blood pressure which is above 140mmHg is generally assessed as hypertension. If the blood pressure measured from a patient with symptoms of diabetes mellitus is 135mmHg, then an intellectualized feedback service which assesses the condition as stage 1 hypertension instead of prehypertension can be provided. This can be provided only when the mutual relationship between risk factors of the disease is identified.

In this research, they propose an intelligent healthcare service model that can enable personal health device to recognize the relationship between mutual diseases and risk factors and provide intellectualized feedback to the patient. In addition, suggestion is made for the assessment algorithm of intellectualized processing essential during the modeling procedure and collaboration application protocol between personal health devices. Furthermore, it is presented the effectiveness of the proposed model through experiments. [5]

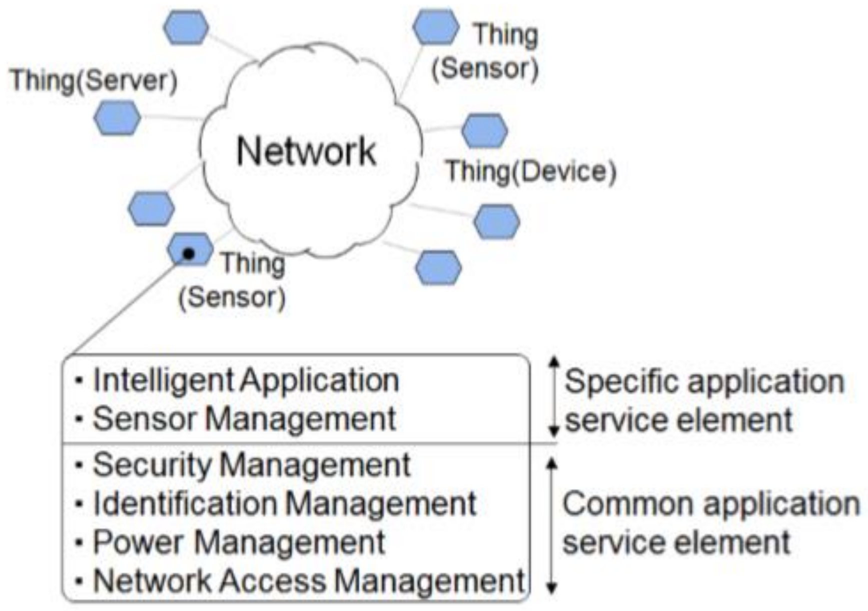

Thing(Server)
Thing (Sensor)
Network
Thing(Device)
Thing (Sensor)
• Intelligent Application
• Sensor Management
• Security Management
• Identification Management
• Power Management
• Network Access Management
Specific application service element
Common application service element

Figure 4.2.2.1: Conceptual Model for Internet of Things and its functionalities [5]

Chapter 3

Project Specification

Previously I was talking about my project architecture. Here I will describe each part briefly.

3.1 Knowing the Each Part of the Project

I have divided my project in three parts to follow that architecture, those are:

- Hardware
- Manage Data in Server site
- Develop Application for Smart Phone

3.1.1 Hardware: Here I have developed a device for detecting human heart rate. For that I need a sensor which can detect human heart rate. To support that sensor I need to connect it with Arduino, and I also need a ESP8266 Wi-Fi module to connect my device with internet

- **Pulse Sensor**

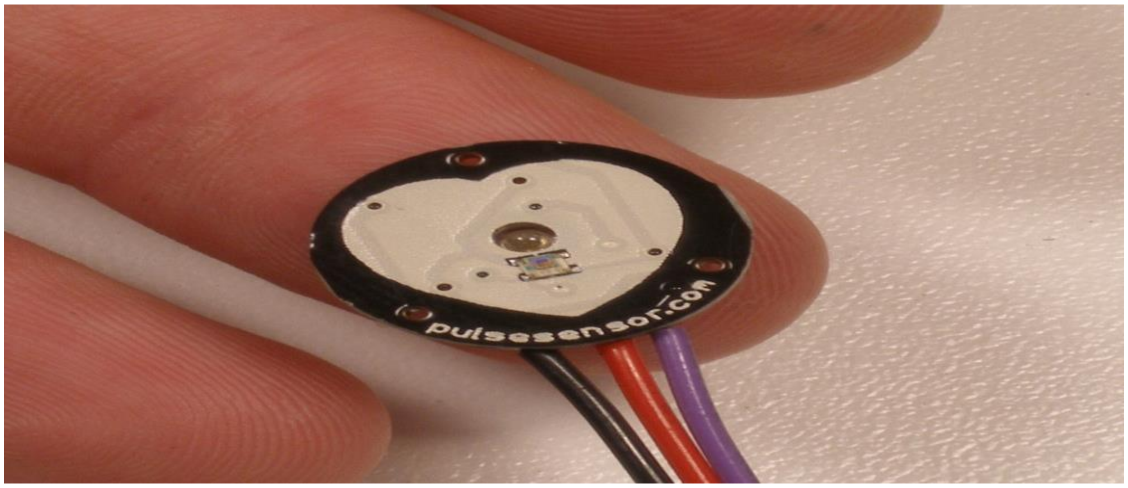


Figure 3.1.1.1: Pulse Sensor

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.

How the Pulse Sensor Works

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 we 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 cellphones, 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 amount of light that bounces back. The other side 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 sensor on the front. Even the LED we are using is a reverse mount LED.

The cable is a 24” flat color coded ribbon cable with 3 male header connectors.

RED wire = +3V to +5V

BLACK wire = GND

PURPLE wire = Signal;

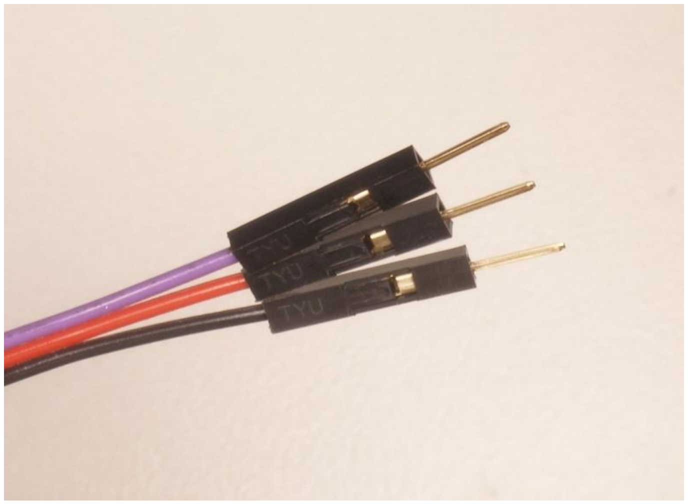


Figure 3.1.1.2: Pulse Sensor Pin Out

The Pulse Sensor can be connected to arduino, or plugged into a breadboard. Before we get it up and running, we need to protect the exposed circuitry so you can get a reliable heartbeat signal.

- **Arduino**

Here I am using Arduino Mega to connect my pulse sensor and Esp8266 Wi-Fi module. The Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

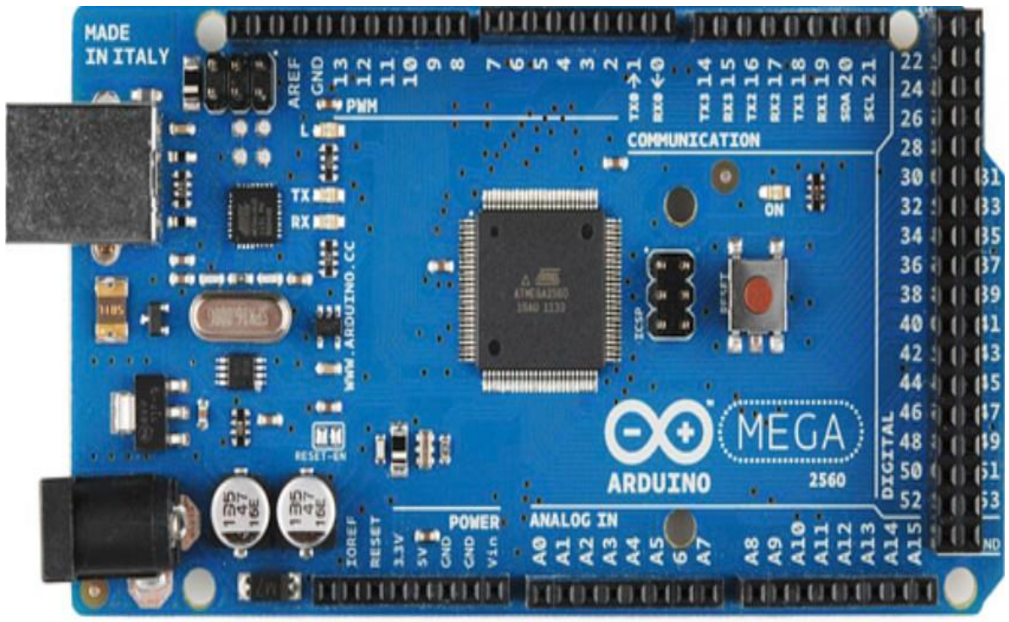


Figure 3.1.1.3: Arduino Mega

- **ESP8266 Wi-Fi Module**

ESP8266 Wi-Fi module is used to connect my device with internet. This very important part in my project. Because without this thing I couldn’t able to send my data to server.

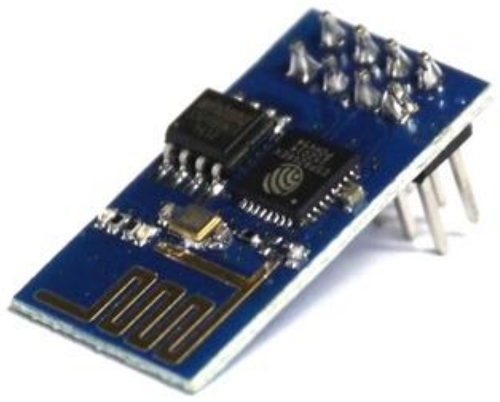


Figure 3.1.1.4: ESP8266 Wi-Fi Module

ESP8266 is an impressive, low cost Wi-Fi module suitable for adding Wi-Fi functionality to an existing microcontroller project via a UART serial connection. The module can even be reprogrammed to act as a standalone Wi-Fi connected device just need to add power. The feature list is impressive and includes: 802.11 b/g/n protocol Wi-Fi Direct (P2P), soft-AP Integrated TCP/IP protocol stack.

The hardware connections required to connect to the ESP8266 module are fairly straight-forward but there are a couple of important items to note related to power: The ESP8266 requires 3.3V power–do not power it with 5 volts! The ESP8266 needs to communicate via serial at 3.3V and does not have 5V tolerant inputs, so we need level conversion to communicate with a 5V microcontroller like most Arduinos use.

- **Liquid Crystal Display**

This will help us to read data from that device, that means it will show us heart rate value in its display.

Figure 3.1.1.5: Liquid Crystal Display

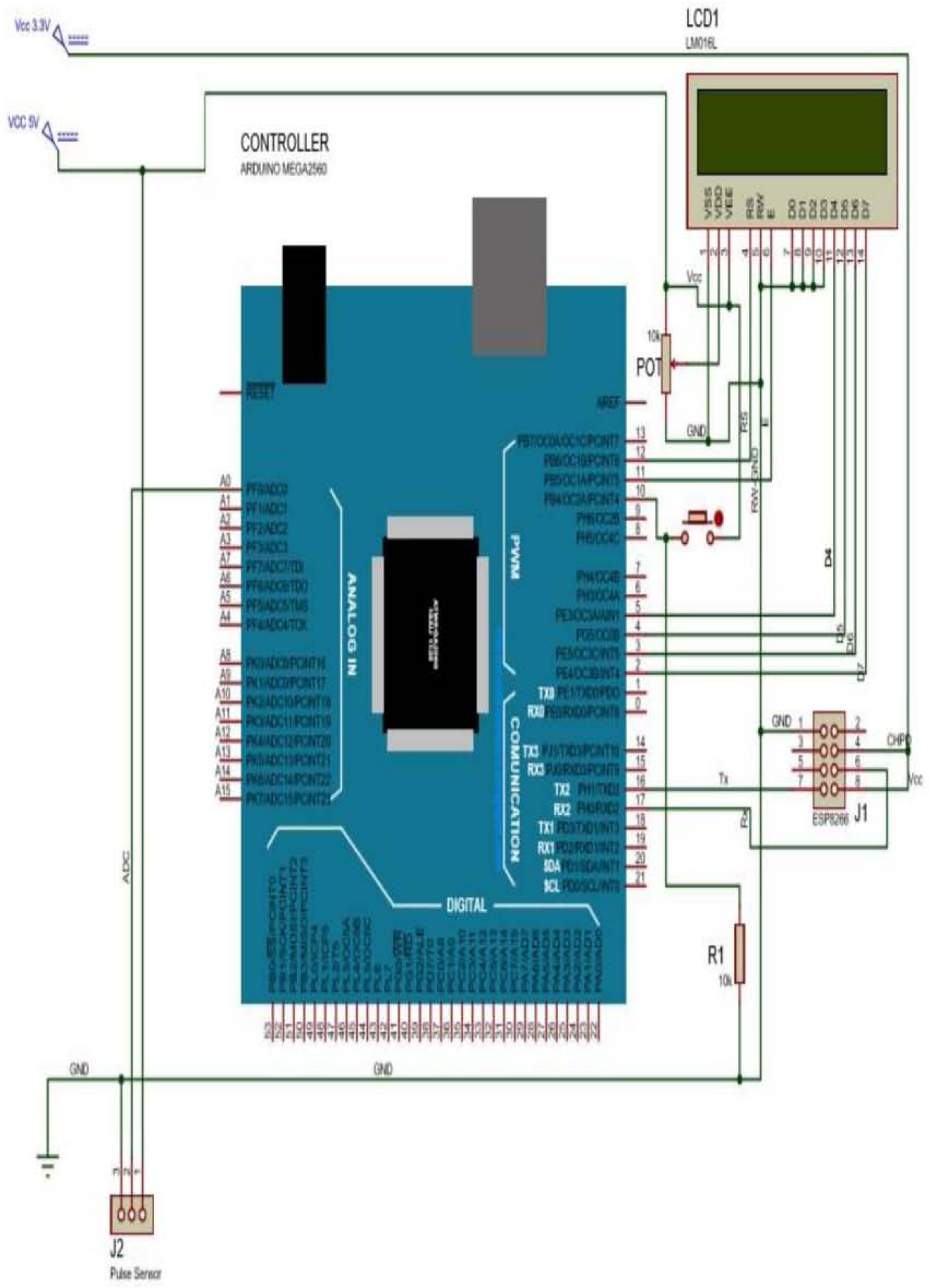


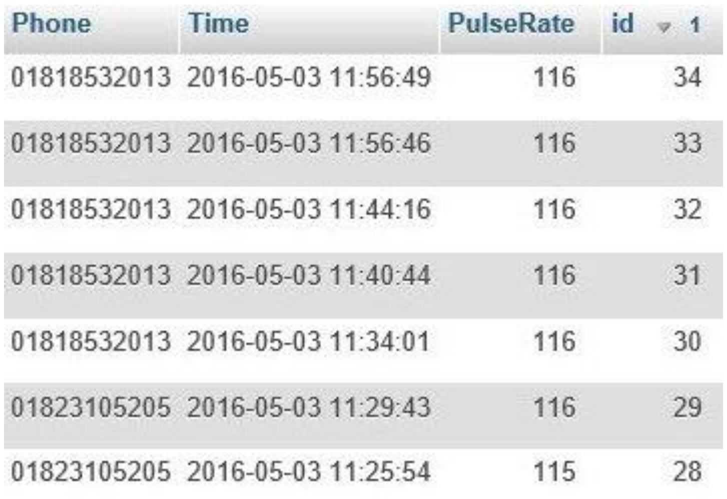
Figure 3.1.1.5: Circuit Diagram for the Device

3.1.2 Manage Data in Server:

In my project I need to send my data to a server. To manage those data I need to use a database where I can store my data.

Basically when a user hit the server from a device, that means the device is sending get request to the server. By sending this request user stored their heart rate data in database table. From that device this URL request will be sending;

"http://localhost:8080/MyHeartRate/PatientPulse.php?beats=100"



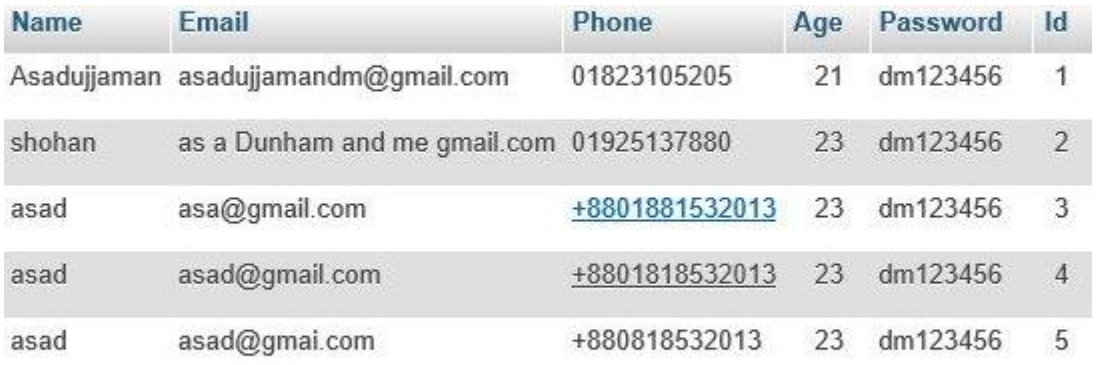
Phone	Time	PulseRate	id ▼ 1
01818532013	2016-05-03 11:56:49	116	34
01818532013	2016-05-03 11:56:46	116	33
01818532013	2016-05-03 11:44:16	116	32
01818532013	2016-05-03 11:40:44	116	31
01818532013	2016-05-03 11:34:01	116	30
01823105205	2016-05-03 11:29:43	116	29
01823105205	2016-05-03 11:25:54	115	28

Figure 3.1.2.1: Patient Pulse info table

So if we see that URL we can see we are sending GET request to a server, when we send this request a php file which called PatientPulse.php is hit. In this file there is some php code which will help the device to store data in database table.

After storing heart rate data in database table user need to see those data to get more information about their heart rate. I am trying to give them a suggestion how to maintain a proper diet chart during their meal. So when they need those information they need to use my application which can give them those information. To manage this service a user need to complete a registration process by which they can store their personal information in database table. For registration process user will send post request to a server

"http://localhost:8080/MyHeartRate/registration.php" this is the post request where user can store their personal information in database table. Here we can see a registration.php, in that file there is some php code which will help to store user personal information.



Name	Email	Phone	Age	Password	Id
Asadujjaman	asadujjamandm@gmail.com	01823105205	21	dm123456	1
shohan	as a Dunham and me gmail.com	01925137880	23	dm123456	2
asad	asa@gmail.com	+8801881532013	23	dm123456	3
asad	asad@gmail.com	+8801818532013	23	dm123456	4
asad	asad@gmai.com	+880818532013	23	dm123456	5

Figure 3.1.2.2: Patient Info Table

Previously I told that user need to see their heart rate information, for this firstly they need to login by using their username and password. For login we also send post request in server. If username and password is valid user then can see their heart rate information.

"http://localhost:8080/MyHeartRate/login.php" this is the post request; when user try to login in my application this request URL is send by them. Here we can see a login.php file which can help user to login in my application, if he/she gives valid username or password.

3.1.3 Develop Application for Smart Phone

I have developed a mobile application which can give user a proper suggestion about how to maintain a proper diet chart during their meals. By using this, user can also see their heart rate information. This application can be used by anyone, mainly heart patients, their relatives and doctors. By using this they can monitor their relatives and patients. At first user need to complete registration process where they need to provide their personal information. To configure that device for user, we need their username. Based on their username data will be stored in the database table.

By this application I need to provide some service to my user. At first user will send their heart rate data to a server. So I need those data to make a meaningful suggestion. For that I need to fetch all heart rate data from database for a specific user to show him/her some suggestion. Data will be processed in my application, that means which kind of message I need to provide to a specific user, this decision will make in this application based on user heart rate data.

This application is for android device. User who has an android device only they can download and install it to their phone. Mainly there are four activities in this application.

- Login
- Registration
- Show Heart Rate data for a specific user.
- Show some suggestion based on a specific heart rate.

If anyone has relatives or family members who are heart patient, they can use this application to monitor their relatives from anywhere they want. They don't need to monitor the device. They just need to fill the login form with their username, so they can get to know about the condition of their relatives or family members. Device only need for a patient who can send his/her hear rate data to the server.

Chapter 4

IoT & Its Protocol

4.1 Protocol

In information technology, a protocol is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities. Protocols exist at several levels in a telecommunication connection. For example, there are protocols for the data interchange at the hardware device level and protocols for data interchange at the application program level. In the standard model known as Open Systems Interconnection (OSI), there are one or more protocols at each layer in the telecommunication exchange that both ends of the exchange must recognize and observe. Protocols are often described in an industry or international standard.

The TCP/IP Internet protocols, a common example, consist of:

- Transmission Control Protocol (TCP), which uses a set of rules to exchange messages with other Internet points at the information packet level
- Internet Protocol (IP), which uses a set of rules to send and receive messages at the Internet address level
- Additional protocols that include the Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP), each with defined sets of rules to use with corresponding programs elsewhere on the Internet

  There are many other Internet protocols, such as the Border Gateway Protocol (BGP) and the Dynamic Host Configuration Protocol (DHCP).

4.2 Protocol Using in IoT Operation

Devices must communicate with each other (D2D). Device data then must be collected and sent to the server infrastructure (D2S). That server infrastructure has to share device data (S2S), possibly providing it back to devices, to analysis programs, or to people. From 30,000 feet, the protocols can be described in this framework as:

• MQTT: a protocol for collecting device data and communicating it to servers (D2S)

• XMPP: a protocol best for connecting devices to people, a special case of the D2S pattern, since people are connected to the servers

• DDS: a fast bus for integrating intelligent machines (D2D)

• AMQP: a queuing system designed to connect servers to each other (S2S)

Each of these protocols is widely adopted. There are at least 10 implementations of each. Confusion is understandable, because the high-level positioning is similar. In fact, all four claim to be real-time publish-subscribe IoT protocols that can connect thousands of devices. And it's true, depending on how you define "real time," "things," and "devices."

Nonetheless, they are very different indeed! Today's Internet supports hundreds of protocols. The IoT will support hundreds more. It's important to understand the class of use that each of these important protocols addresses.

The simple taxonomy in Figure 2 frames the basic protocol use cases. Of course, it's not really that simple. For instance, the "control plane" represents some of the complexity in controlling and managing all these connections. Many protocols cooperate in this region. **[7]**

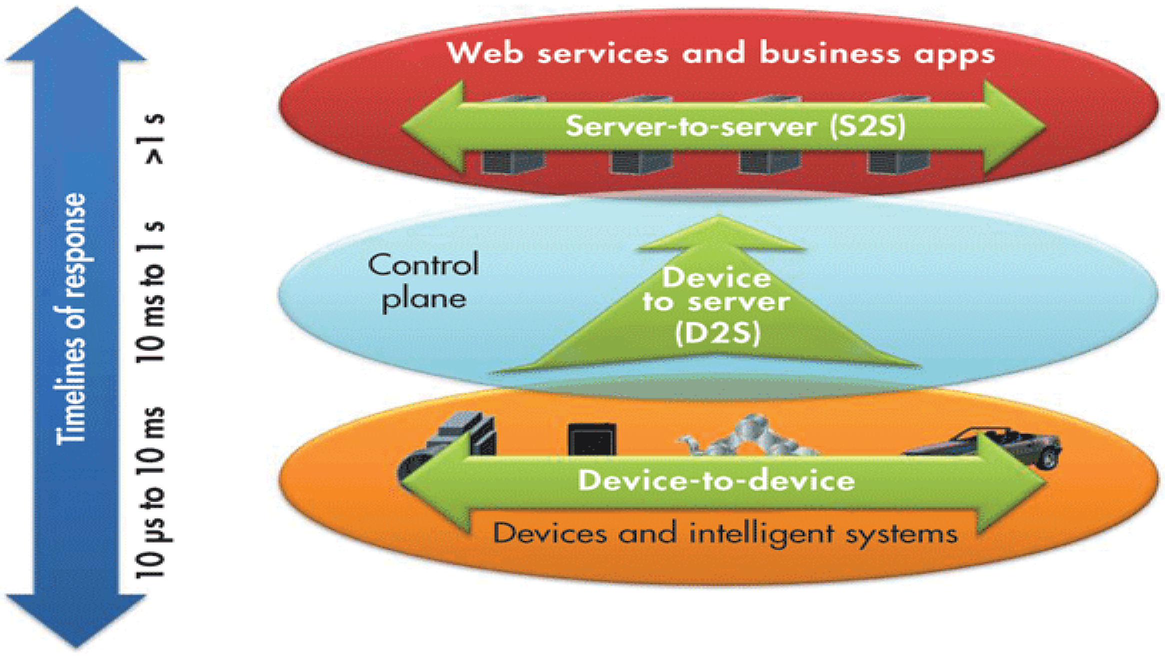


Figure 4.2: IoT protocols need to address response time. [7]

4.2.1 MQTT

MQTT, the Message Queue Telemetry Transport, targets device data collection. As its name states, its main purpose is telemetry, or remote monitoring. Its goal is to collect data from many devices and transport that data to the IT infrastructure. It targets large networks of small devices that need to be monitored or controlled from the cloud.

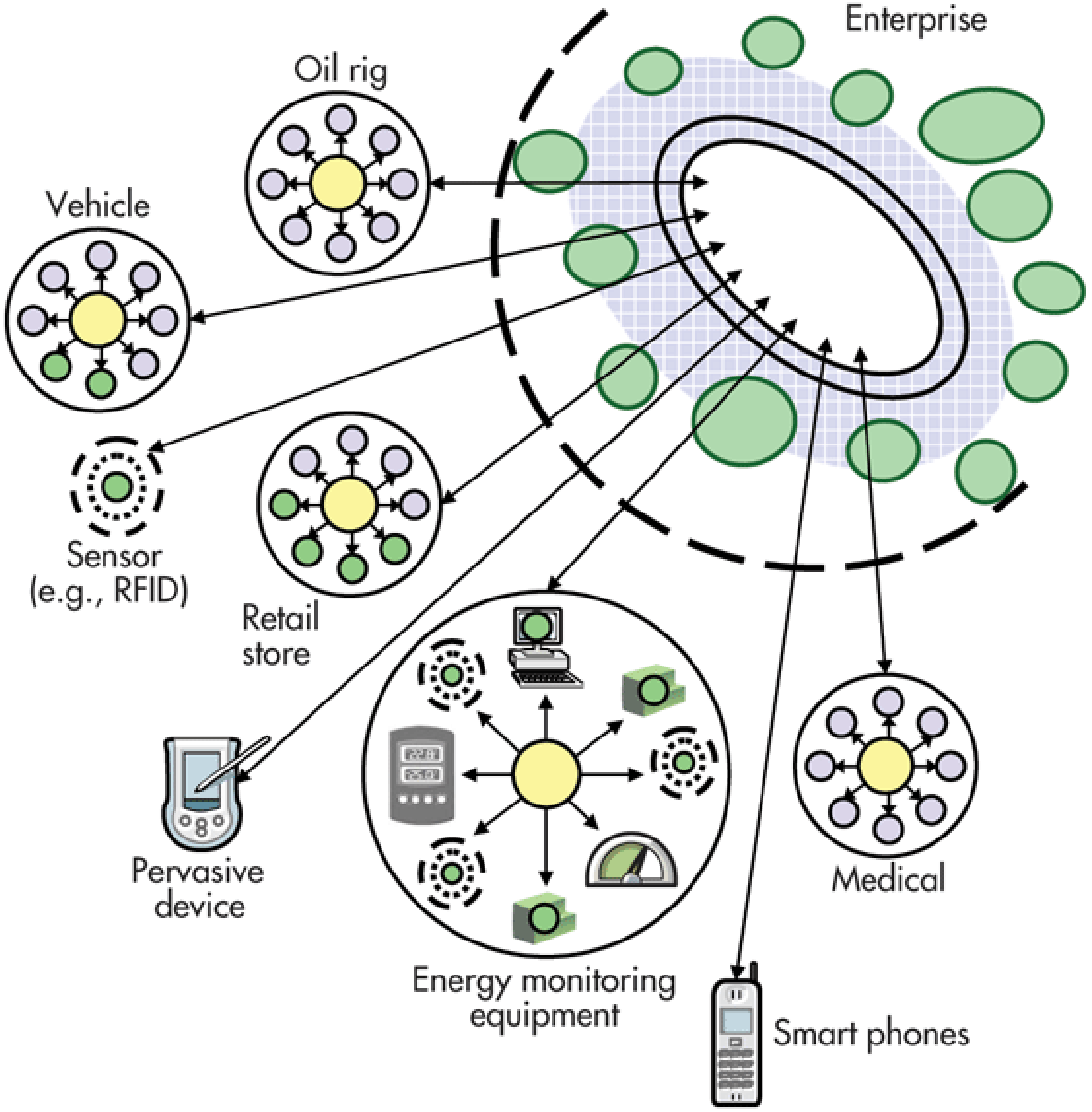


Figure 4.2.1: Message Queue Telemetry Transport (MQTT) implements a hub-and-spoke system. [7]

MQTT makes little attempt to enable device-to-device transfer, nor to "fan out" the data to many recipients. Since it has a clear, compelling single application, MQTT is simple, offering few control options. It also doesn't need to be particularly fast. In this context, "real time" is typically measured in seconds.

A hub-and-spoke architecture is natural for MQTT. All the devices connect to a data concentrator server, like IBM's new Message Sight appliance. You don't want to lose data, so the protocol works on top of TCP, which provides a simple, reliable stream. Since the IT infrastructure uses the data, the entire system is designed to easily transport data into enterprise technologies like Active MQ and enterprise service buses (ESBs).

MQTT enables applications like monitoring a huge oil pipeline for leaks or vandalism. Those thousands of sensors must be concentrated into a single location for analysis. When the system finds a problem, it can take action to correct that problem. Other applications for MQTT include power usage monitoring, lighting control, and even intelligent gardening. They share a need for collecting data from many sources and making it available to the IT infrastructure. **[7]**

4.2.2 XMPP

XMPP was originally called "Jabber." It was developed for instant messaging (IM) to connect people to other people via text messages. XMPP stands for Extensible Messaging and Presence Protocol. Again, the name belies the targeted use: presence, meaning people are intimately involved.

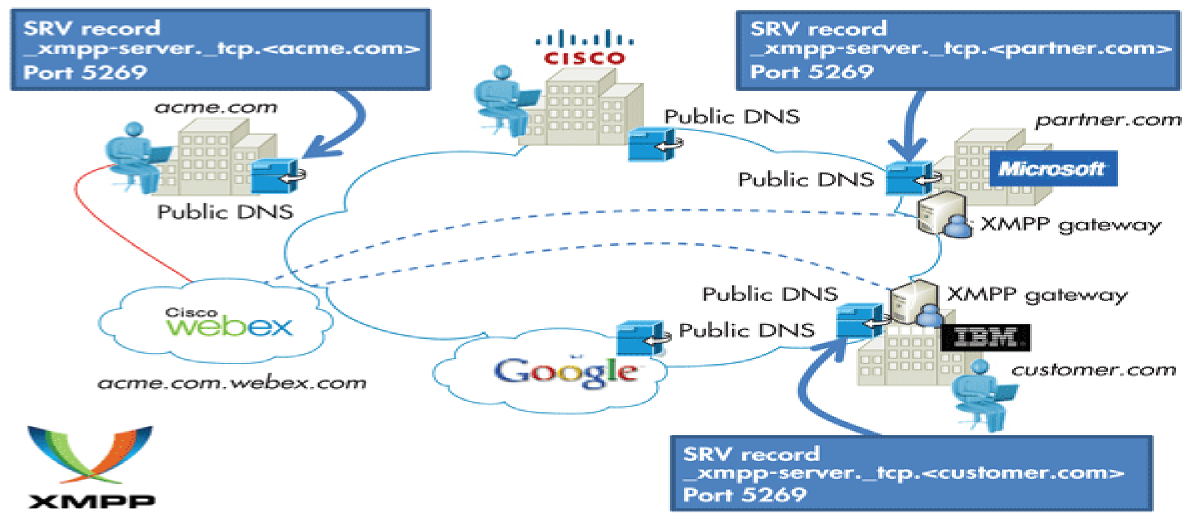


Figure 4.2.2: The Extensible Messaging and Presence Protocol (XMPP) provides text communication between points. [7]

XMPP uses the XML text format as its native type, making person-to-person communications natural. Like MQTT, it runs over TCP, or perhaps over HTTP on top of TCP. Its key strength is a name@domain.com addressing scheme that helps connect the needles in the huge Internet haystack.

In the IoT context, XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case. It's not designed to be fast. In fact, most implementations use polling, or checking for updates only on demand. A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets severs push messages. But "real time" to XMPP is on human scales, measured in seconds.

XMPP provides a great way, for instance, to connect your home thermostat to a Web server so you can access it from your phone. Its strengths in addressing, security, and scalability make it ideal for consumer-oriented IoT applications. **[7]**

4.2.3 DDS

In contrast to MQTT and XMPP, the Data Distribution Service (DDS) targets devices that directly use device data. It distributes data to other devices *(Fig. 5)*. While interfacing with the IT infrastructure is supported, DDS's main purpose is to connect devices to other devices. It is a data-centric middleware standard with roots in high-performance defense, industrial, and embedded applications. DDS can efficiently deliver millions of messages per second to many simultaneous receivers.

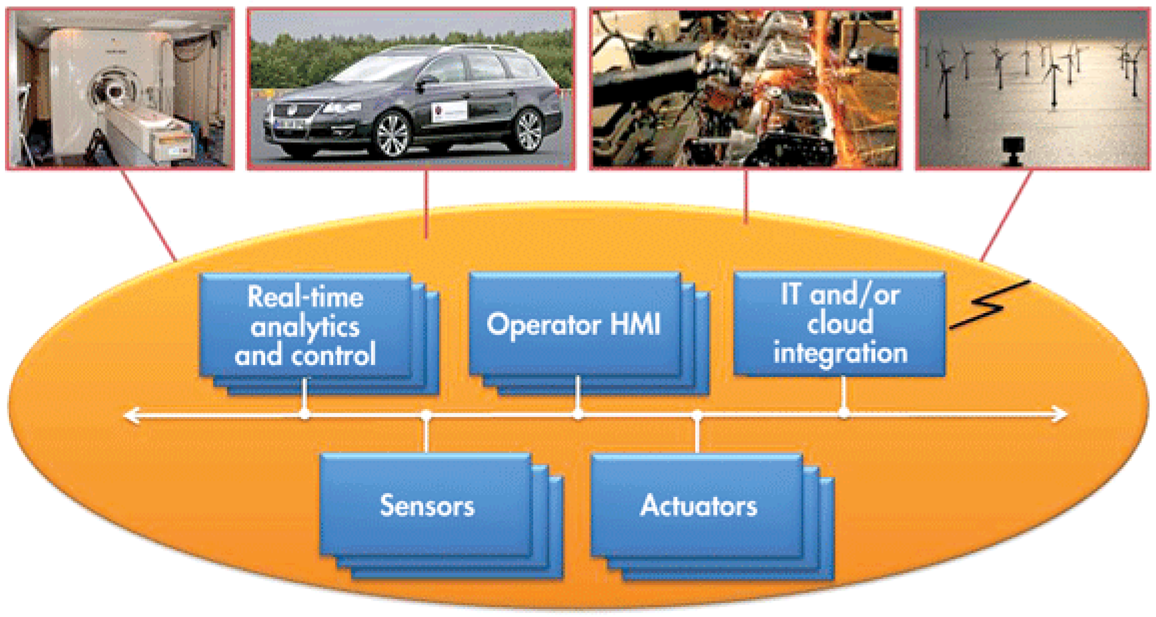


Figure 4.2.3: Data Distribution Service (DDS) implements a publish/subscribe architecture. [7]

Devices demand data very differently than the IT infrastructure demands data. First, devices are fast. “Real time” is often measured in microseconds. Devices need to communicate with many other devices in complex ways, so TCP’s simple and reliable point-to-point streams are far too restrictive. Instead, DDS offers detailed quality-of-service (QoS) control, multicast, configurable reliability, and pervasive redundancy. In addition, fan-out is a key strength. DDS offers powerful ways to filter and select exactly which data goes where, and “where” can be thousands of simultaneous destinations. Some devices are small, so there are lightweight versions of DDS that run in constrained environments.

Hub-and-spoke is completely inappropriate for device data use. Rather, DDS implements direct device-to-device “bus” communication with a relational data model. RTI calls this a “DataBus” because it is the networking analog to a database. Similar to the way a database controls access to stored data, a data bus controls data access and updates by many simultaneous users. This is exactly what many high-performance devices need to work together as a single system.

High-performance integrated device systems use DDS. It is the only technology that delivers the flexibility, reliability, and speed necessary to build complex, real-time applications. Applications include military systems, wind farms, hospital integration, medical imaging, asset-tracking systems, and automotive test and safety. DDS connects devices together into working, distributed applications at physics speeds. **[7]**

4.2.4 AMQP

Finally, the Advanced Message Queuing Protocol (AMQP) is sometimes considered an IoT protocol. AMQP is all about queues *(Fig. 6)*. It sends transactional messages between servers. As a message-centric middleware that arose from the banking industry, it can process thousands of reliable queued transactions.

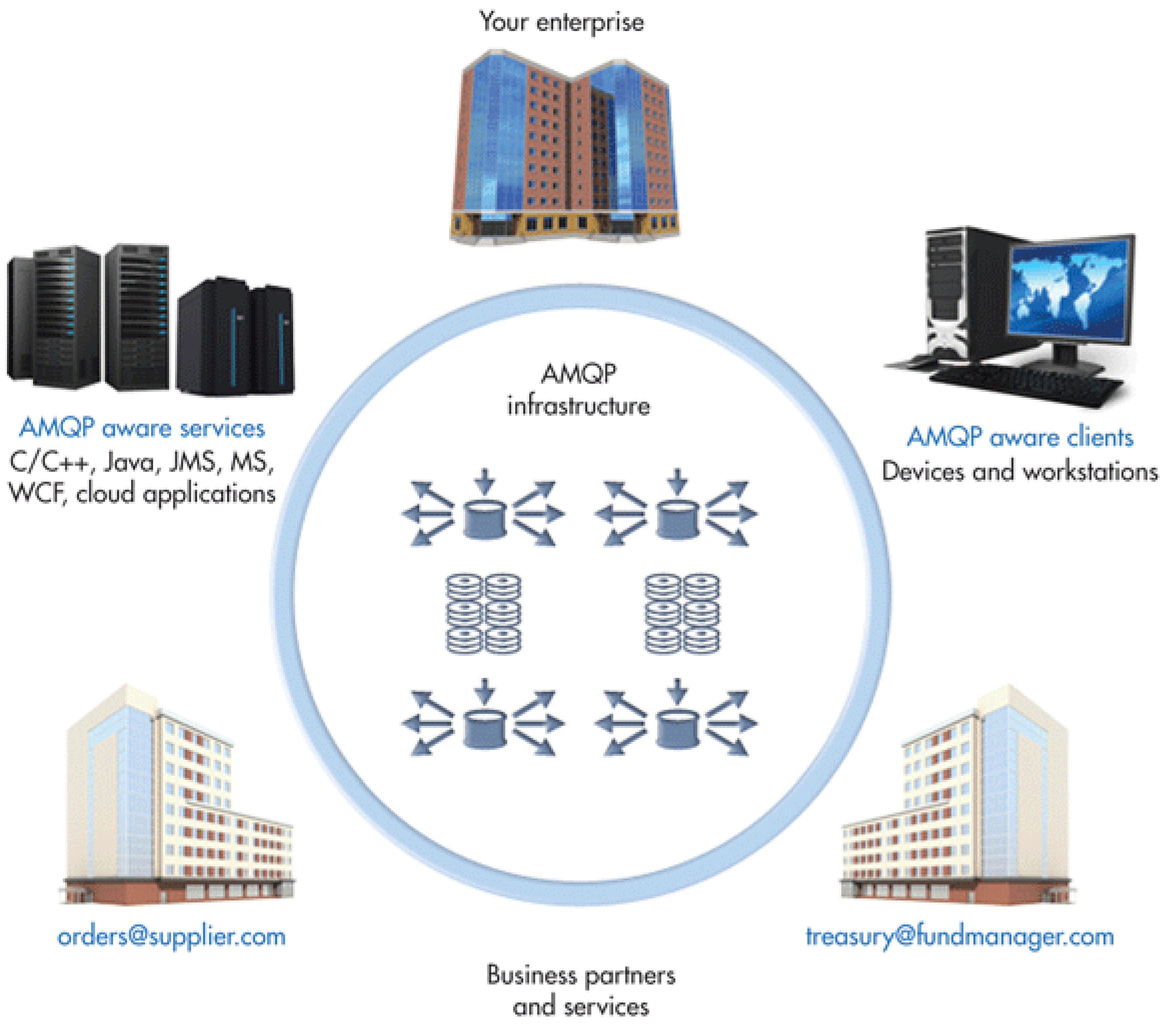


Figure 4.2.4: The Advanced Message Queuing Protocol (AMQP) is messages-centric middleware that arose from the banking industry. [7]

AMQP is focused on not losing messages. Communications from the publishers to exchanges and from queues to subscribers use TCP, which provides strictly reliable point-to-point connection. Further, endpoints must acknowledge acceptance of each message. The standard also describes an optional transaction mode with a formal multiphase commit sequence. True to its origins in the banking industry, AMQP middleware focuses on tracking all messages and ensuring each is delivered as intended, regardless of failures or reboots. AMQP is mostly used in business messaging. It usually defines “devices” as mobile handsets communicating with back-office data centers. In the IoT context, AMQP is most appropriate for the control plane or server-based analysis functions. **[7]**

4.3 Protocol Used in my Project:

Here I was trying to give an overview about IoT's protocols and frameworks in previous sections. Those will help us to make IoT projects. I did not used any framework or specific platform to develop my project but for the purpose of communication I need to follow a protocol. Which is quite similar to XMPP. In my project user send data from device to server and to get those data from server it will be using a phone.

For successful communication it requires sending and receiving data operation managed by HTTP GET and POST request. Usually XMPP used for person to person communication. But in IoT we know two or many things/device is connected with each other through the internet. They need to communicate with each other. So for successful communication they need to follow a protocol, and to make this happen I am using XMPP. Where my device can send data to the server and other device which is phone can read data from server.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Nowadays Internet became more popular thing. Without this we can’t live a single day. Internet is continuously changing our life style. Internet now crossed its boundary and changed the concept, nowadays internet is not only connecting people it also connecting things/devices. Where devices can communicate with each other through internet.

This new concept of internet of things covers a wide area, where anyone can take part by making different kind of project. For communication devices need sensors. Various kinds of sensors can help people to make different kind of project in IoT.

So I was trying to implement this new concept in my project. I tried make this simple by using IoT concept. In our country the total number of Internet subscribers has reached **61.288 million** at the end of **March, 2016[8]**. That means smartphone user is also increasing day by day in our country. So this is the best time to change the idea about using internet in our country.

Bangladesh is a developing country, most of the people in our country are not so rich. So they are not able to pay high amount for any kind of technology which will help them to lead a healthy life style. So my intension was clear, I was trying to develop a device for them which is really not so expensive. By using this device I hope they will get some benefit.

5.2 Future

People became more and more dependent on technology. This technology can able to change human thought and life style. In our daily life we are attached with technology. IoT will change human life style in near future. Because this new concept has a wide area.

People in Bangladesh are not so health conscious. So I tried to provide them a online service based on IoT.

In my project I will provide a service to my user by a mobile application, I know this is not enough to serve my user. If I want to guide them in proper way I need to help form hospital. Initially there is no sector for hospitals or medicals to control my service. Where they can serve their patients directly. To create more opportunity for patients and hospitals, I need to involve the hospital’s helps. Where they can monitor their patient more effectively.

5.3 Limitation

In my project there are some limitation for user. I didn’t give them any option to change their Wi-Fi SSID or Password. They have to use default SSID and password which will be configured in that device, and before using that device, they need to ensure that they are in same SSID network, or they can open hotspot from their mobile or any other device by using the default SSID and Password. The device then connect with internet if matching SSID and password is found. To give them proper suggestion I need more time to make decision. Initially I couldn’t be able to give them a best service. To ensure a good service we need to analyze their data more and more.

5.4 References:

[1] http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#1a1222d56828

[2] http://www.survivingwithandroid.com/2015/12/internet-of-things-free-iot-platforms-2.html

[3] https://www.ibm.com/developerworks/cloud/library/cl-bluemixfoundry/

[4] http://www.engpaper.net/free-research-papers-iot-internet-of-thing.htm

[5] http://www.sersc.org/journals/IJBSBT/vol6_no1/17.pdf

[6] http://www.sersc.org/journals/IJSH/vol9_no1_2015/21.pdf

[7] http://electronicdesign.com/iot/understanding-protocols-behind-internet-things

[8] http://www.btrc.gov.bd/content/internet-subscribers-bangladesh-march-2016

[9]https://www.google.com/search?q=internet+of+things&espv=2&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiovLSk4cHMAhWCBY4KHS0ICK0Q_AUICSgD&biw=1366&bih=643#imgrc=uvcHhzX_YrXrOM%3A

BIBLIOGRAPY

1. **http://www.w3schools.com/**

2. **https://www.arduino.cc/**

3. **https://github.com/esp8266/Arduino**

4. **http://pulsesensor.com/**

5. **http://www.tutorialspoint.com/android/**

6. **https://www.youtube.com/watch?v=x0I5vJfaRIU&list=PLe60o7ed8E-TpIFm6EXK1CEKz0ubPDJb-**

<u>Server Site Code</u>

<u>File Name</u>: init.php

```
<?php
    $db_name = "myheartrate";
    $mysql_user = "root";
    $mysql_pass = "";
    $server_name = "localhost";

    $con = mysqli_connect($server_name,$mysql_user,$mysql_pass,$db_name);

    if(!$con)
    {
        //echo "Connection Error...".mysqli_connect_error();
    }
    else
    {
        //echo "<h3> Database connection Success...</h3>";
    }
?>
```

<u>File Name</u>: registration.php

```
<?php
    require "init.php";

    $name = $_POST["Name"];
    $email = $_POST["Email"];
    $phone = $_POST["Phone"];
    $age = $_POST["Age"];
    $password = $_POST["Pass"];

    $sql_query_for_check_existing_user = "select * from patientinfo where Phone like '$phone';";
    $result = mysqli_query($con,$sql_query_for_check_existing_user);


    if(mysqli_num_rows($result)>0)
    {
        echo "This Phone ".$phone." Number Already Exist...";
    }
    else
    {
        $sql_query = "insert into patientinfo values('$name','$email','$phone','$age','$password');";
        mysqli_query($con,$sql_query);
    }
    mysqli_close($con);
?>
```

<u>File Name</u>: login.php

```
<?php
    require "init.php";

    $phone =   $_POST["Phone"];
    $userPass = $_POST["Pass"];

    $sql_query = "select Name from patientinfo where Phone like '$phone' and Password like '$userPass';";

    $result = mysqli_query($con,$sql_query);

    if(mysqli_num_rows($result)>0)
    {
        $row = mysqli_fetch_assoc($result);
        $Name = $row["Name"];
        echo  "Login Success...Welcome";
    }
    else
    {
        echo "Login Failed......Try Again..";
    }
    mysqli_close($con);
?>
```

File Name: Time.php

```
<?php
    require "init.php";

    $phone = $_GET["Phone"];
    $beats = $_GET["beats"];
    $sql_query = "INSERT INTO `patientpulse`(`Phone`,`PulseRate`) VALUES ('$phone','$beats');";

    if (mysqli_query($con,$sql_query))
    {
        echo "New record created successfully";
    }
    else
    {
        echo "Error: " . $sql . "<br>" . mysqli_error($con);
    }
    mysqli_close($con);
?>
```

Arduino Code:

File Name: ArduBeat.ino

```
#include "esp8266.h"
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

char PAGE[]="/MyHeartRate/WriteInFile.php?beats=";
char *WEBPAGE = (char*)malloc((sizeof(char)*strlen(PAGE))+3);

//  VARIABLES for heartbeat
int pulsePin = 0;                 // Pulse Sensor purple wire connected to analog pin 0
int blinkPin = 13;                // pin to blink led at each beat
int fadePin = 5;                  // pin to do fancy classy fading blink at each beat
int fadeRate = 0;                 // used to fade LED on with PWM on fadePin

// these variables are volatile because they are used during the interrupt service routine!
volatile int BPM;                   // used to hold the pulse rate
volatile int Signal;                // holds the incoming raw data
volatile int IBI = 600;             // holds the time between beats, the Inter-Beat Interval
volatile boolean Pulse = false;     // true when pulse wave is high, false when it's low
volatile boolean QS = false;        // becomes true when Arduoino finds a beat.



void sendDataToServer(int BPM);

void setup()
{
     pinMode(blinkPin,OUTPUT);         // pin that will blink to your heartbeat!
     pinMode(fadePin,OUTPUT);          // pin that will fade to your heartbeat!
     Serial.begin(9600);             // we agree to talk fast!
   Serial1.begin(9600);         //Open software serial for chatting to ESP

    // sets up to read Pulse Sensor signal every 2mS
    // IF YOU ARE POWERING The Pulse Sensor AT VOLTAGE LESS THAN THE BOARD VOLTAGE,
    // UN-COMMENT THE NEXT LINE AND APPLY THAT VOLTAGE TO THE A-REF PIN
    // analogReference(EXTERNAL);

   lcd.begin(16, 2);

     Serial.println(F("Adafruit's ESP8266 Demo"));


     //test if the module is ready
     if(! espReset()) {
         Serial.println("Module didn't respond :(");
         debugLoop();
     }

     Serial.println(F("ESP Module is ready! :)"));

     //connect to the wifi
     byte err = setupWiFi();

     if (err) {
         // error, print error code
         Serial.print("setup error:");
         Serial.println((int)err);
         debugLoop();
     }

     // success, print IP
     uint32_t ip = getIP();
     Serial.print("ESP setup success, my IP addr:");
     if (ip) {
         Serial.println(ip, HEX);
         } else {
         Serial.println("none");
     }

     sendCheckReply("AT+CIPSTO=0", "OK");

   interruptSetup();

}


//  Where the Magic Happens
void loop()
{
  //sendDataToProcessing('S', Signal);     // send Processing the raw Pulse Sensor data
  if (QS == true){              // Quantified Self flag is true when arduino finds a heartbeat
        fadeRate = 255;        // Set 'fadeRate' Variable to 255 to fade LED with pulse
        sendDataToProcessing('B',BPM);   // send heart rate with a 'B' prefix

         sendDataToServer(BPM);
        lcd.setCursor(0, 1);// the data to send culminating in a carriage return
        lcd.print(BPM);
        sendDataToProcessing('Q',IBI);   // send time between beats with a 'Q' prefix
        QS = false;                      // reset the Quantified Self flag for next time
     }
  delay(20);                             //  take a break
}

void sendDataToServer(int BPM){

  char strBuff[50];

  itoa(BPM,strBuff,10);

   //size= strlen(strBuff);
   strcpy(WEBPAGE,PAGE);
   strcat(WEBPAGE,strBuff);
   Serial.print("Request URL:");
   Serial.println(WEBPAGE);


  ESP_GETpage(HOST, 8080, WEBPAGE);

  Serial.println(F("**********REPLY***********"));
  Serial.print(replybuffer);
  Serial.println(F("**************************"));

  sendCheckReply("AT+CIPCLOSE", "OK");


  //debugLoop();

    delay(3000);
  }

  void sendDataToProcessing(char symbol, int data ){
    Serial.print(symbol);  //symbol prefix tells Processing what type of data is coming
    Serial.println(data);

  }
```

File Name: Interrupt.ino

```
volatile int rate[10];                    // used to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find the inter beat interval
volatile int P =512;                      // used to find peak in pulse wave
volatile int T = 512;                     // used to find trough in pulse wave
volatile int thresh = 512;                // used to find instant moment of heart beat
volatile int amp = 100;                   // used to hold amplitude of pulse waveform
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable BPM
volatile boolean secondBeat = true;       // used to seed rate array so we startup with reasonable BPM


void interruptSetup(){
  // Initializes Timer2 to throw an interrupt every 2mS.
  TCCR2A = 0x02;     // DISABLE PWM ON DIGITAL PINS 3 AND 11, AND GO INTO CTC MODE
  TCCR2B = 0x06;     // DON'T FORCE COMPARE, 256 PRESCALER
  OCR2A = 0X7C;      // SET THE TOP OF THE COUNT TO 124 FOR 500Hz SAMPLE RATE
  TIMSK2 = 0x02;     // ENABLE INTERRUPT ON MATCH BETWEEN TIMER2 AND OCR2A
  sei();             // MAKE SURE GLOBAL INTERRUPTS ARE ENABLED
}


// THIS IS THE TIMER 2 INTERRUPT SERVICE ROUTINE.
// Timer 2 makes sure that we take a reading every 2 miliseconds
ISR(TIMER2_COMPA_vect){                         // triggered when Timer2 counts to 124
    cli();                                      // disable interrupts while we do this
    Signal = analogRead(pulsePin);              // read the Pulse Sensor
    sampleCounter += 2;                         // keep track of the time in mS with this variable
    int N = sampleCounter - lastBeatTime;       // monitor the time since the last beat to avoid noise

//  find the peak and trough of the pulse wave
    if(Signal < thresh && N > (IBI/5)*3){       // avoid dichrotic noise by waiting 3/5 of last IBI
        if (Signal < T){                        // T is the trough
            T = Signal;                         // keep track of lowest point in pulse wave
         }
       }

    if(Signal > thresh && Signal > P){          // thresh condition helps avoid noise
        P = Signal;                             // P is the peak
       }                                        // keep track of highest point in pulse wave

  //  NOW IT'S TIME TO LOOK FOR THE HEART BEAT
  // signal surges up in value every time there is a pulse
if (N > 250){                                   // avoid high frequency noise
```

```
  if ( (Signal > thresh) && (Pulse == false) && (N > (IBI/5)*3) ){
    Pulse = true;                               // set the Pulse flag when we think there is a pulse
    digitalWrite(blinkPin,HIGH);                // turn on pin 13 LED
    IBI = sampleCounter - lastBeatTime;         // measure time between beats in mS
    lastBeatTime = sampleCounter;               // keep track of time for next pulse

         if(firstBeat){                         // if it's the first time we found a beat, if firstBeat ==
TRUE
             firstBeat = false;                 // clear firstBeat flag
             return;                            // IBI value is unreliable so discard it
            }
         if(secondBeat){                        // if this is the second beat, if secondBeat == TRUE
            secondBeat = false;                 // clear secondBeat flag
               for(int i=0; i<=9; i++){         // seed the running total to get a realisitic BPM at
startup
                    rate[i] = IBI;
                    }
            }

    // keep a running total of the last 10 IBI values
    word runningTotal = 0;                   // clear the runningTotal variable

    for(int i=0; i<=8; i++){                // shift data in the rate array
          rate[i] = rate[i+1];              // and drop the oldest IBI value
          runningTotal += rate[i];          // add up the 9 oldest IBI values
        }

    rate[9] = IBI;                          // add the latest IBI to the rate array
    runningTotal += rate[9];                // add the latest IBI to runningTotal
    runningTotal /= 10;                     // average the last 10 IBI values
    BPM = 60000/runningTotal;               // how many beats can fit into a minute? that's BPM!
    QS = true;                              // set Quantified Self flag
    // QS FLAG IS NOT CLEARED INSIDE THIS ISR
    }
}

  if (Signal < thresh && Pulse == true){     // when the values are going down, the beat is over
      digitalWrite(blinkPin,LOW);            // turn off pin 13 LED
      Pulse = false;                         // reset the Pulse flag so we can do it again
      amp = P - T;                           // get amplitude of the pulse wave
      thresh = amp/2 + T;                    // set thresh at 50% of the amplitude
      P = thresh;                            // reset these for next time
      T = thresh;
     }

  if (N > 2500){                             // if 2.5 seconds go by without a beat
      thresh = 512;                          // set thresh default
      P = 512;                               // set P default
      T = 512;                               // set T default
      lastBeatTime = sampleCounter;          // bring the lastBeatTime up to date
      firstBeat = true;                      // set these to avoid noise
      secondBeat = true;                     // when we get the heartbeat back
     }

  sei();                                     // enable interrupts when youre done!
}// end isr
```

File Name: esp8266.h

```
    #ifndef ESP8266_H
    #define ESP8266_H

    #include <Arduino.h>

    #define SSID "asad_wifi"      //your wifi ssid here
    #define PASS "18532013"    //your wifi key here

    //  www.adafruit.com/testwifi/index.html
    #define HOST "192.168.1.101"
    //#define WEBPAGE "/iot/"
```

```
#define PORT  "8080"

#define ESP_RST 12

// Use software serial (check to make sure these are valid softserial pins!)
#define ESP_RX 9
#define ESP_TX 10
//SoftwareSerial softser(ESP_RX, ESP_TX); // RX, TX
Stream *esp = &Serial1;

// can also do
// Stream *esp = &Serial1;

#define REPLYBUFFSIZ 255
char replybuffer[REPLYBUFFSIZ];
uint8_t getReply(char *send, uint16_t timeout = 500, boolean echo = true);
uint8_t espreadline(uint16_t timeout = 500, boolean multiline = false);
boolean sendCheckReply(char *send, char *reply, uint16_t timeout = 500);


enum {WIFI_ERROR_NONE=0, WIFI_ERROR_AT, WIFI_ERROR_RST, WIFI_ERROR_SSIDPWD, WIFI_ERROR_SERVER,
WIFI_ERROR_UNKNOWN};

boolean ESP_GETpage(char *host, uint16_t port, char *page) {
    String cmd = "AT+CIPSTART=\"TCP\",\"";
    cmd += host;
    cmd += "\",";
    cmd += port;
    cmd.toCharArray(replybuffer, REPLYBUFFSIZ);

    getReply(replybuffer);

    if (strcmp(replybuffer, "OK") != 0) {
        // this is OK! could be a version that says "Linked"
        if (strcmp(replybuffer, "Linked") != 0) {
            sendCheckReply("AT+CIPCLOSE", "OK");
            return false;
        }
    }

    String request = "GET ";
    request += page;
    request += " HTTP/1.1\r\nHost: ";
    request += host;
    request += "\r\n\r\n";

    cmd = "AT+CIPSEND=";
    cmd += request.length();
    cmd.toCharArray(replybuffer, REPLYBUFFSIZ);
    sendCheckReply(replybuffer, ">");

    Serial.print("Sending: "); Serial.println(request.length());
    Serial.println(F("*********SENDING*********"));
    Serial.print(request);
    Serial.println(F("*************************"));

    request.toCharArray(replybuffer, REPLYBUFFSIZ);

    esp->println(request);

    while (true) {
        espreadline(3000);  // this is the 'echo' from the data
        Serial.print(">"); Serial.println(replybuffer); // probably the 'busy s...'

        // LOOK AT ALL THESE POSSIBLE ARBITRARY RESPONSES!!!
        if (strstr(replybuffer, "wrong syntax"))
        continue;
        else if (strstr(replybuffer, "ERROR"))
        continue;
```

```
            else if (strstr(replybuffer, "busy s..."))
            continue;
            else break;
        }

        if (! strstr(replybuffer, "SEND OK") ) return false;

        espreadline(1000);  Serial.print("3>"); Serial.println(replybuffer);
        char *s = strstr(replybuffer, "+IPD,");
        if (!s) return false;
        uint16_t len = atoi(s+5);
        //Serial.print(len); Serial.println(" bytes total");

        int16_t contentlen = 0;
        while (1) {
            espreadline(50);
            s = strstr(replybuffer, "Content-Length: ");
            if (s) {
                //Serial.println(replybuffer);
                contentlen = atoi(s+16);
                Serial.print(F("C-Len = ")); Serial.println(contentlen);
            }
            s = strstr(replybuffer, "Content-Type: ");
            if (s && contentlen) {
                int16_t i;
                char c;

                for (i=-2; i<contentlen; i++) {  // eat the first 2 chars (\n\r)
                    while (!esp->available());
                    c = esp->read(); //UDR0 = c;
                    if (i >= 0) {
                        replybuffer[i] = c;
                    }
                }
                replybuffer[i] = 0;
                return true;
            }
        }
        //while (1) {
        //  if (esp.available()) UDR0 = esp.read();
        //}
    }


boolean getVersion() {
  // Get version?
  getReply("AT+GMR", 250, true);
}

boolean espReset() {
  getReply("AT+RST", 1000, true);
  if (! strstr(replybuffer, "OK")) return false;
  delay(2000);

  // turn off echo
  getReply("ATE0", 250, true);

  return true;
}

boolean ESPconnectAP(char *s, char *p) {

  getReply("AT+CWMODE=1", 500, true);
  if (! (strstr(replybuffer, "OK") || strstr(replybuffer, "no change")) )
    return false;

  String connectStr = "AT+CWJAP=\"";
  connectStr += SSID;
  connectStr += "\",\"";
  connectStr += PASS;
  connectStr += "\"";
  connectStr.toCharArray(replybuffer, REPLYBUFFSIZ);
  getReply(replybuffer, 500, true);
  espreadline(5000);
  Serial.print("<-- "); Serial.println(replybuffer);

  return (strstr(replybuffer, "OK") != 0);
}


byte setupWiFi() {
  // reset WiFi module
  Serial.println(F("Soft resetting..."));
  if (!espReset())
    return WIFI_ERROR_RST;

  delay(1000);

  Serial.println(F("Checking for ESP AT response"));

  if (!sendCheckReply("AT", "OK"))
    return WIFI_ERROR_AT;

  getVersion();
  Serial.print(F("Firmware Version #")); Serial.println(replybuffer);

  Serial.print(F("Connecting to ")); Serial.println(SSID);
  if (!ESPconnectAP(SSID, PASS))
    return WIFI_ERROR_SSIDPWD;

  Serial.println(F("Single Client Mode"));
  if (!sendCheckReply("AT+CIPMUX=0", "OK"))
        return WIFI_ERROR_SERVER;

  return WIFI_ERROR_NONE;
}

// NOT IMPLEMENTED YET!
uint32_t getIP() {
  getReply("AT+CIFSR", 500, true);

  return 0;
}




/************************/
uint8_t espreadline(uint16_t timeout, boolean multiline) {
  uint16_t replyidx = 0;

  while (timeout--) {
    if (replyidx > REPLYBUFFSIZ-1) break;

    while(esp->available()) {
      char c =  esp->read();
      if (c == '\r') continue;
      if (c == 0xA) {
        if (replyidx == 0)   // the first 0x0A is ignored
          continue;

        if (!multiline) {
          timeout = 0;         // the second 0x0A is the end of the line
          break;
        }
      }
      replybuffer[replyidx] = c;
      // Serial.print(c, HEX); Serial.print("#"); Serial.println(c);
      replyidx++;
    }

    if (timeout == 0) break;
    delay(1);
  }
  replybuffer[replyidx] = 0;  // null term
  return replyidx;
}

uint8_t getReply(char *send, uint16_t timeout, boolean echo) {
  // flush input
  while(esp->available()) {
     esp->read();
  }

  if (echo) {
    Serial.print("---> "); Serial.println(send);
  }
  esp->println(send);

  // eat first reply sentence (echo)
  uint8_t readlen = espreadline(timeout);

  //Serial.print("echo? "); Serial.print(readlen); Serial.print(" vs "); Serial.println(strlen(send));

  if (strncmp(send, replybuffer, readlen) == 0) {
    // its an echo, read another line!
    readlen = espreadline();
  }

  if (echo) {
    Serial.print ("<--- "); Serial.println(replybuffer);
  }
  return readlen;
}

boolean sendCheckReply(char *send, char *reply, uint16_t timeout) {
  getReply(send, timeout, true);

/*
  for (uint8_t i=0; i<strlen(replybuffer); i++) {
    Serial.print(replybuffer[i], HEX); Serial.print(" ");
  }
  Serial.println();
  for (uint8_t i=0; i<strlen(reply); i++) {
    Serial.print(reply[i], HEX); Serial.print(" ");
  }
  Serial.println();
  */
  return (strcmp(replybuffer, reply) == 0);
}

void debugLoop() {
  Serial.println("========================");
  //serial loop mode for diag
  while(1) {
    if (Serial.available()) {
      esp->write(Serial.read());
      delay(1);
    }
    if (esp->available()) {
      Serial.write(esp->read());
      delay(1);
    }
  }
}

#endif
```

Android Code

File Name: MainActivity.java

```
package com.example.shotlu.heartrate;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

public class MainActivity extends Activity{

    EditText etPhoneNumber,etPassword;
    String loginUserName, loginPassword;
    boolean isLoggedIn = false;
    LogIn logIn = new LogIn();
    Button btLogin;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etPhoneNumber = (EditText) findViewById(R.id.etPhone);
        etPassword = (EditText) findViewById(R.id.etPass);
    }

    public void userReg(View view)
    {
        startActivity(new Intent(this,Reistration.class));
    }

    public void userLogin(View view)
    {
        loginUserName = "01823105205";//etPhoneNumber.getText().toString();//
        loginPassword = "dm123456";//etPassword.getText().toString();//
        String method="login";
        Bundle  basket = new Bundle();
        basket.putString("loginUserName",loginUserName);
        FormValidation formValidation = new FormValidation(loginUserName,loginPassword);

        if(formValidation.ValidateLogin())
        {
//            String method = "login";
//            BackGroundTask backGroundTask = new BackGroundTask(this);
//            backGroundTask.execute(method, loginUserName, loginPassword);
            new LoginBackGroundTask(this).execute(method,loginUserName,loginPassword);
        }
        else
        {
            Toast.makeText(getBaseContext(), formValidation.ValidateLoginMessage(),
Toast.LENGTH_LONG).show();
        }
    }

    class LoginBackGroundTask extends AsyncTask<String,Void,String>
    {
        AlertDialog alertDialog;
        Context ctx;
        Reistration reistration;


        String userPhone;

        LoginBackGroundTask(Context ctx)
        {
            this.ctx = ctx;
        }

        @Override
        protected void onPreExecute()
        {
            alertDialog = new AlertDialog.Builder(ctx).create();
            alertDialog.setTitle("Login Information...");
        }

        @Override
        protected String doInBackground(String... params)
        {
            String loginUrl = "http://192.168.1.101:8080/MyHeartRate/login.php";//192.168.1.102

            String method = params[0];
            String loginUsername = params[1];
            String loginPassword = params[2];

            try
            {
                URL url = new URL(loginUrl);
                HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
                httpURLConnection.setDoInput(true);
                OutputStream outputStream = httpURLConnection.getOutputStream();

  BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));

                String data = URLEncoder.encode("Phone", "UTF-
8") + "=" + URLEncoder.encode(loginUsername, "UTF-8")
                        + "&" + URLEncoder.encode("Pass", "UTF-
8") + "=" + URLEncoder.encode(loginPassword, "UTF-8");

                bufferedWriter.write(data);
                bufferedWriter.flush();
                bufferedWriter.close();
                outputStream.close();
                InputStream inputStream = httpURLConnection.getInputStream();

  BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream, "iso-8859-1"));
                String response = "";
                String line = "";
                while ((line = bufferedReader.readLine()) != null) {
                    response += line;
                }
                bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
                return response;
            }
            catch (MalformedURLException e)
            {
                e.printStackTrace();
            }
            catch (IOException e)
            {
                e.printStackTrace();
            }

            return null;
        }

        @Override
        protected void onProgressUpdate(Void... values) {
            super.onProgressUpdate(values);
        }

        @Override
        protected void onPostExecute(String result)
        {

            if (result.equals("Login Success...Welcome"))
            {
                //Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
                IsLoggedIn();
            }
            else if (result.equals("Login Failed......Try Again.."))
            {
                Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
            }
        }
    }

    public void IsLoggedIn()
    {
        finish();
        Toast.makeText(this.getApplicationContext(), "LOG IN SUCCESSFULL", Toast.LENGTH_LONG).show();
        Intent i = new Intent(getApplicationContext(), PatientPulseDetails.class);
        i.putExtra("new_variable_name", loginUserName);
        startActivity(i);
    }
}
```

File Name: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tool="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_height="match_parent"
   android:layout_width="match_parent">

    <TextView
       android:layout_width="wrap_content"
       android:text="Login"
       android:textStyle="bold"
       android:textSize="50dp"
       android:layout_gravity="center_horizontal"
       android:layout_height="wrap_content"
       android:layout_marginBottom="30dp"
       />

    <TextView
       android:layout_width="wrap_content"
       android:text="Username or Phone"
       android:layout_height="wrap_content"
       />

    <EditText
```

```
        android:id="@+id/etPhone"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content"
        />

    <TextView
        android:layout_width="wrap_content"
        android:text="Password"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etPass"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:inputType="textPassword"
        android:layout_height="wrap_content"
        />

    <Button
        android:id="@+id/btLogin"
        android:text="Login"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="userLogin"/>

    <Button
        android:id="@+id/btRegister"
        android:text="Register Here"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="userReg"/>

</LinearLayout>
```

File Name: Registration.java

```
package com.example.shotlu.heartrate;

import android.app.Activity;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class Reistration extends Activity {

    EditText etName,etEmail,etPhone,etPass,etConPass,etAge;
    String name,phone,pass,email,age,confermPass;


    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reistration);
        etName=(EditText) findViewById(R.id.etName);
        etEmail=(EditText) findViewById(R.id.etEmail);
        etPhone=(EditText) findViewById(R.id.etPhone);
        etPass=(EditText) findViewById(R.id.etPass);
        etConPass=(EditText) findViewById(R.id.etConPass);
        etAge=(EditText) findViewById(R.id.etAge);
    }

    public void userReg(View view)
    {
            name = etName.getText().toString();
            phone = etPhone.getText().toString();
            pass = etPass.getText().toString();
            confermPass = etConPass.getText().toString();
            email = etEmail.getText().toString();
            age = etAge.getText().toString();

            FormValidation formValidation = new FormValidation(name,phone,email,pass,confermPass,age);

            if(formValidation.ValidateRegistration())
            {
                String method = "registration";
                BackGroundTask backGroundTask = new BackGroundTask(this);
                backGroundTask.execute(method, name, email, phone, age, pass);
                Toast.makeText(getBaseContext(),"registration successful",Toast.LENGTH_LONG).show();
            }

            else
            {
Toast.makeText(getBaseContext(),formValidation.ValidatenRegistrationMessage(),Toast.LENGTH_LONG).show();
            }
    }

    public void userBack(View view)
    {
       finish();
       Intent intent = new Intent(this,MainActivity.class);
       startActivity(intent);
    }
}
```

File Name: activity_registration.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tool="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_height="match_parent"
   android:layout_width="match_parent">

   <ScrollView
      android:layout_width="fill_parent"
      android:layout_height="499dp">
       <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="vertical">
    <TextView
       android:layout_width="wrap_content"
       android:text="Registration"
       android:layout_marginBottom="20dp"
       android:textSize="50dp"
       android:layout_gravity="center_horizontal"
       android:layout_height="wrap_content"
       />

    <TextView
       android:layout_width="wrap_content"
       android:text="Name"
       android:layout_height="wrap_content"
       />
    <EditText
       android:id="@+id/etName"
       android:layout_width="match_parent"
```

```
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content"
        />

    <TextView
        android:layout_width="wrap_content"
        android:text="Use your Phone number as your Username"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etPhone"
        android:text=""
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content"
        />

    <TextView
        android:layout_width="wrap_content"
        android:text="Email"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content"
        />

    <TextView
        android:layout_width="wrap_content"
        android:text="Age"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etAge"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:layout_height="wrap_content"
        />

    <TextView
        android:layout_width="wrap_content"
        android:text="Pass"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etPass"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:inputType="textPassword"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_width="wrap_content"
        android:text="ConPass"
        android:layout_height="wrap_content"
        />
    <EditText
        android:id="@+id/etConPass"
        android:layout_width="match_parent"
        android:layout_marginBottom="10dp"
        android:inputType="textPassword"
        android:layout_height="wrap_content"
        />

    <Button
        android:id="@+id/btRegister"
        android:text="Registration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="userReg"/>

    <Button
        android:id="@+id/btRegisterBack"
        android:text="Back"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="userBack"/>
        </LinearLayout>
   </ScrollView>
</LinearLayout>
```

File Name: BackGroundTask.java

```
package com.example.shotlu.heartrate;

import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

public class BackGroundTask extends AsyncTask<String,Void,String> {
    AlertDialog alertDialog;
    Context ctx;
    MainActivity mainActivity;
    Reistration reistration;
    boolean check = false;
    public LoginResponse loginResponse = null;

    String userPhone;

    BackGroundTask(Context ctx) {
        this.ctx = ctx;
    }


    @Override
    protected void onPreExecute() {
        alertDialog = new AlertDialog.Builder(ctx).create();
        alertDialog.setTitle("Login Information...");
    }

    @Override
    protected String doInBackground(String... params) {

        String regUrl = "http://192.168.1.101:8080/MyHeartRate/registration.php";//192.168.1.102
        String loginUrl = "http://192.168.1.101:8080/MyHeartRate/login.php";//192.168.1.102

        String method = params[0];


        /////////////////////////For Registration/////////////////////////
        if (method.equals("registration")) {
            String userName = params[1];
            String userEmail = params[2];
            userPhone = params[3];
            //int userAge = Integer.parseInt(params[4]);
            String userAge = params[4];
            String userPass = params[5];

            try {
                URL url = new URL(regUrl);
                HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
                OutputStream OS = httpURLConnection.getOutputStream();
                BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(OS, "UTF-8"));
                String data = URLEncoder.encode("Name", "UTF-8") + "=" + URLEncoder.encode(userName, "UTF-
8")
                        + "&" + URLEncoder.encode("Email", "UTF-
8") + "=" + URLEncoder.encode(userEmail, "UTF-8")
                        + "&" + URLEncoder.encode("Phone", "UTF-
8") + "=" + URLEncoder.encode(userPhone, "UTF-8")
                        + "&" + URLEncoder.encode("Age", "UTF-
8") + "=" + URLEncoder.encode(String.valueOf(userAge), "UTF-8")
                        + "&" + URLEncoder.encode("Pass", "UTF-
8") + "=" + URLEncoder.encode(userPass, "UTF-8");
                bufferedWriter.write(data);
                bufferedWriter.flush();
                bufferedWriter.close();
                OS.close();
                InputStream IS = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(IS, "iso-8859-
1"));

                String response = "";
                String line = "";
                while ((line = bufferedReader.readLine()) != null) {
                    response += line;
                }
                bufferedReader.close();
                IS.close();
                httpURLConnection.disconnect();
                return response;
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }

        }


        /////////////////////////For Login/////////////////////////

        else if (method.equals("login")) {
            String loginUsername = params[1];
            String loginPassword = params[2];

            try {
                URL url = new URL(loginUrl);
                HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
                httpURLConnection.setRequestMethod("POST");
                httpURLConnection.setDoOutput(true);
                httpURLConnection.setDoInput(true);
                OutputStream outputStream = httpURLConnection.getOutputStream();

  BufferedWriter bufferedWriter = new BufferedWriter(new OutputStreamWriter(outputStream, "UTF-8"));

                String data = URLEncoder.encode("Phone", "UTF-
8") + "=" + URLEncoder.encode(loginUsername, "UTF-8")
                        + "&" + URLEncoder.encode("Pass", "UTF-
8") + "=" + URLEncoder.encode(loginPassword, "UTF-8");

                bufferedWriter.write(data);
                bufferedWriter.flush();
                bufferedWriter.close();
                outputStream.close();
                InputStream inputStream = httpURLConnection.getInputStream();

  BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream, "iso-8859-1"));
                String response = "";
                String line = "";
                while ((line = bufferedReader.readLine()) != null) {
                    response += line;
                }
                bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();
                return response;
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return null;
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }

    @Override
    protected void onPostExecute(String result)
    {

        if (result.equals("Registration Successfull!!"))
        {
            Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
            alertDialog.setMessage(result);
            alertDialog.show();
        }
        else if (result.equals("Registration failed!!!!") || result.equals("This Phone " + userPhone + "
Number Already Exist..."))
        {
            Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
            alertDialog.setMessage(result);
            alertDialog.show();
        }
        else if (result.equals("Login Success...Welcome"))
        {
            Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
            check = true;
        }
        else if (result.equals("Login Failed......Try Again.."))
        {
            Toast.makeText(ctx, result, Toast.LENGTH_LONG).show();
            check = false;
        }
        super.onPostExecute(result);
    }


}
```

File Name: FormValidation.java

```
package com.example.shotlu.heartrate;

/**
 * Created by Shotlu on 3/25/2016.
 */
public class FormValidation {
    String message;
    String name,phone,email,password,confermPassword,age;

    /************************************ Registration From
Validation*******************************************/

  public FormValidation(String name, String phone, String email,String password, String confermPass, Strin
g age)
    {
        this.name=name;
        this.phone=phone;
        this.email=email;
        this.password=password;
        this.confermPassword=confermPass;
        this.age=age;
    }

    public FormValidation(String userName, String password)
    {
        this.phone = userName;
        this.password = password;
    }

    private boolean ValidName()
    {
        if(name.length()>0 && name != null)
        {

            return true;
        }
        else
        {
            message = "name is not valid!";
            return false;
        }
    }

    private boolean ValidEmail()
    {
        String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\.+[a-z]+";
        if(email.length()>0 && email != null && email.matches(emailPattern))
        {
            return true;
        }
        else
        {
            message = "email is not valid!";
            return false;
        }
    }

    private boolean ValidPhone()
    {
        /*String numberPattern = "^\\+[0-9]{10,13}$";
        if(phone.length()>0 && phone != null && phone.matches(numberPattern))
        {
            return true;
        }
        else
        {
            message = "phone number is not valid!";
            return false;
        }*/
        return true;
    }

    private boolean ValidPassword()
    {
        if(password.length()>0 && password != null)
        {
            return true;
        }
        else
        {
            message = " password is not valid!";
            return false;
        }
    }

    private boolean ValidConfermPassword()
    {
        if(confermPassword.length()>0 && confermPassword != null && confermPassword.matches(password))
        {
            return true;
        }
        else
        {
            message = " password does not match!";
            return false;
        }
    }

    private boolean ValidAge()
    {
        int validAge = Integer.parseInt(age);
        if(validAge>0 && validAge < 150)
        {
            return true;
        }
        else
        {
            message = " age is not valid!";
            return false;
        }
    }

    public boolean ValidateRegistration()
    {

  if(ValidName()&& ValidPhone() && ValidEmail() && ValidPassword() && ValidConfermPassword() && ValidAge()
)
        {
            return true;
        }
        else
        {
            return  false;
        }
    }

    public String ValidatenRegistrationMessage()
    {
        return message;
    }

    public boolean ValidateLogin()
    {
        if(ValidPhone() && ValidPassword())
        {
            return true;
        }
        else
        {
            return  false;
        }
    }
    public String ValidateLoginMessage()
    {
        return message;
    }
}
```

File Name: Contacts.java

```
package com.example.shotlu.heartrate;

/**
 * Created by omee on 3/27/2016.
 */
public class Contacts {

    public Contacts(String Phone, String Time, String PulseRate)
    {
        this.setPhone(Phone);
        this.setTime(Time);
        this.setPulseRate(PulseRate);
    }

    private String phone,time,pulseRate;


    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getPulseRate() {
        return pulseRate;
    }

    public void setPulseRate(String pulseRate) {
        this.pulseRate = pulseRate;
    }
}
```

File Name: ContactAdapter.java

```
package com.example.shotlu.heartrate;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

public class ContactAdapter extends ArrayAdapter  {

    List list = new ArrayList();

    public ContactAdapter(Context context, int resource) {
        super(context, resource);
    }
    public void add(Contacts object) {
        super.add(object);
        list.add(object);
    }

    @Override
    public int getCount() {
        return list.size();
    }

    @Override
    public Object getItem(int position) {
        return list.get(position);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View row;
        row = convertView;
        ContactHolder contactHolder;
        if (row == null) {
            LayoutInflater layoutInflater
= (LayoutInflater) this.getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            row = layoutInflater.inflate(R.layout.row_layout, parent, false);
            contactHolder = new ContactHolder();
            contactHolder.txPhone = (TextView) row.findViewById(R.id.tx_phone);
            contactHolder.txTime = (TextView) row.findViewById(R.id.tx_time);
            contactHolder.txPulseRate = (TextView) row.findViewById(R.id.tx_pulseRate);
            row.setTag(contactHolder);
        } else {
            contactHolder = (ContactHolder) row.getTag();
        }
        Contacts contacts = (Contacts) this.getItem(position);
        contactHolder.txPhone.setText(contacts.getPhone());
        contactHolder.txTime.setText(contacts.getTime());
        contactHolder.txPulseRate.setText(contacts.getPulseRate());
        return row;
    }

    static class ContactHolder {
        TextView txPhone, txTime, txPulseRate;
    }
}
```

File Name: PatientPulseDetails.java

```
package com.example.shotlu.heartrate;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;

public class PatientPulseDetails extends Activity {

    String jsonSTRING;
    static String value;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_patient_pulse_details);
    }

    public void getJson(View view)
    {
        new BackgroundTask().execute();
        //ShowJsonValue(jsonSTRING);
    }

    class BackgroundTask extends AsyncTask<String,Void,String>
    {
        String jsonUrl;
        String jsonString;


        @Override
        protected void onPreExecute() {
            Bundle extras = getIntent().getExtras();
            if (extras != null) {
                value = extras.getString("new_variable_name");
            }
            jsonUrl
="http://192.168.1.101:8080/MyHeartRate/json_get_data.php?phone="+value;//192.168.43.88
        }

        @Override
        protected String doInBackground(String... params) {
            try {

                URL url = new URL(jsonUrl);
                HttpURLConnection httpURLConnection = (HttpURLConnection)url.openConnection();
                InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
                StringBuilder stringBuilder = new StringBuilder();

                while((jsonString = bufferedReader.readLine())!=null)
                {
                    stringBuilder.append(jsonString);
                }

                bufferedReader.close();
                inputStream.close();
                httpURLConnection.disconnect();

                return stringBuilder.toString().trim();

            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }

        @Override
        protected void onProgressUpdate(Void... values) {
            super.onProgressUpdate(values);
        }
```

```java
        @Override
        protected void onPostExecute(String result) {
            //TextView textView = (TextView)findViewById(R.id.textview);
            //textView.setText(result);
            jsonSTRING=result;
            ShowJsonValue(jsonSTRING);
        }
    }

    public void ShowJsonValue(String jsonSTRING)
    {
        if(jsonSTRING==null)
        {
            Toast.makeText(getApplicationContext(), "First Get JSON", Toast.LENGTH_LONG).show();
        }
        else
        {
            Intent intent = new Intent(this, DisplayListView.class);
            intent.putExtra("jsonData",jsonSTRING);
            startActivity(intent);
        }
    }

    public void dataBack(View view)
    {
        finish();
        Intent intent = new Intent(this,MainActivity.class);
        startActivity(intent);
    }
}
```

File Name: activity_patient_pulse_details.xml

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tool="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_height="match_parent"
   android:layout_width="match_parent">

    <TextView
       android:layout_width="wrap_content"
       android:text="Heart Rate"
       android:textSize="50dp"
       android:layout_gravity="center_horizontal"
       android:layout_height="wrap_content"
       />

    <Button
       android:id="@+id/btGetData"
       android:text="Get Pulse Info"
       android:textStyle="bold"
       android:layout_gravity="center_horizontal"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_marginTop="20dp"
       android:onClick="getJson"/>

    <Button
       android:id="@+id/btDataBack"
       android:text="Logout"
       android:textStyle="bold"
       android:layout_gravity="center_horizontal"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_marginTop="20dp"
       android:onClick="dataBack"/>

</LinearLayout>
```

File Name: DisplayListView.java

```
package com.example.shotlu.heartrate;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class DisplayListView extends AppCompatActivity {

    String jsonString;
    JSONObject jsonObject;
    JSONArray jsonArray;
    ContactAdapter contactAdapter;
    ListView listView;
    public static String pulse="";
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_list_view);
        listView = (ListView) findViewById(R.id.listview);
        contactAdapter = new ContactAdapter(this,R.layout.row_layout);
        listView.setAdapter(contactAdapter);
        jsonString = getIntent().getExtras().getString("jsonData");

        try
        {
            jsonObject = new JSONObject(jsonString);
            jsonArray = jsonObject.getJSONArray("server_response");
            int count=0;
            String phone,time,pulseRate;

            while(count <jsonArray.length())
            {
                JSONObject JO = jsonArray.getJSONObject(count);
                phone = JO.getString("Phone");
                time = JO.getString("Time");
                pulseRate = JO.getString("PulseRate");
                Contacts contacts = new Contacts(phone,time,pulseRate);

                contactAdapter.add(contacts);
                count ++;
            }
        }
        catch (JSONException e)
        {
            e.printStackTrace();
        }

        listView.setOnItemClickListener((new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Contacts st = (Contacts) parent.getItemAtPosition(position);
                //String phone = st.getPhone();
                pulse = st.getPulseRate();
                Toast.makeText(getBaseContext(), pulse + " is selected ", Toast.LENGTH_LONG).show();

                finish();
                Intent intent = new Intent(DisplayListView.this,ShowSelectedItem.class);
                intent.putExtra(pulse,pulse);
                startActivity(intent);
            }
        }));
    }

    public void getBack(View view)
    {
        Intent intent = new Intent(this, PatientPulseDetails.class);
        startActivity(intent);
    }
}
```

File Name: activity_display_list_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   tools:context="com.example.shotlu.readdata.DisplayListView">

    <ListView
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:id="@+id/listview"
       android:layout_above="@+id/btBack" />

    <Button
       android:id="@+id/btBack"
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:layout_alignParentBottom="true"
       android:layout_alignParentLeft="true"
       android:layout_alignParentStart="true"
       android:onClick="getBack"
       android:text="Back"
       />

</RelativeLayout>
```

File Name: ShowSelectedItem.java

```
package com.example.shotlu.heartrate;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class ShowSelectedItem extends AppCompatActivity{

   TextView tvMessageShow;
   Button backButton;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_show_selected_item);

      String message= null;
      int pulseRate=0;
      tvMessageShow = (TextView) findViewById(R.id.tvMessage);

      pulseRate = Integer.parseInt(getIntent().getStringExtra(DisplayListView.pulse));
```

```
        if(pulseRate >= 110)
        {
            message="high pressure";
            tvMessageShow.setText(message);
        }
        else if(pulseRate <= 90 && pulseRate >= 70)
        {
            message="Normal Pressure";
            tvMessageShow.setText(message);
        }
        else if(pulseRate<70)
        {
            message = "Low Pressure";
            tvMessageShow.setText(message);
        }
    }

    public void clickBack(View view)
    {
        finish();
        Intent intent = new Intent(this,PatientPulseDetails.class);
        startActivity(intent);
    }
}
```

File Name: activity_show_selected_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.shotlu.heartrate.ShowSelectedItem">

    <TextView
        android:id="@+id/tvMessage"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:text="Hi"
        android:textColor="#ff0000"
        android:textSize="30dp"
        android:textAppearance="?android:textAppearanceLarge"
        android:textStyle="italic"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_margin="155dp"
        />

    <Button
        android:id="@+id/btListBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="back"
        android:onClick="clickBack"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        />

</RelativeLayout>
```