**M S Ramaiah Institute of Technology**

(An Autonomous Institute, Affiliated to VTU)

MSR nagar, MSRIT post, Bangalore-54

A Design Document on

**Smart Health care Monitoring System Using Raspberry pi 2**

Under the guidance of

Mrs. S. Rajarajeswari.

Submitted by

Himanshu Kumar 1MS12CS039

Suhail T N 1MS12CS115

Manoj more S 1MS13CS412

Suresh V 1MS13CS421

*In partial fulfillment for the award of the degree of*

# *Bachelor of Engineering in Computer Science & Engineering*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**M.S. RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute, Affiliated to VTU)**

**BANGALORE-560054**

[www.msrit.edu](http://www.msrit.edu), **May 2015**

**INTRODUCTION**

Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the project should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behavior models, and other supporting requirement information? Design Document is to provide a description of the design of a system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the project and system to be built. Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.

**Modularization**, Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently. Modular design unintentionally follows the rules of ‘divide and conquer’ problem-solving strategy this is because there are many other benefits attached with the modular design of software.

Advantage of modularization:

* Smaller components are easier to maintain
* Program can be divided based on functional aspects
* Desired level of abstraction can be brought in the program
* Components with high cohesion can be re-used again
* Concurrent execution can be made possible
* Desired from security aspect

**Concurrencies**, Back in time, all software are meant to be executed sequentially. By sequential execution we mean that the coded instruction will be executed one after another implying only one portion of program being activated at any given time. Say, a software has multiple modules, then only one of all the modules can be found active at any time of execution. In software design, concurrency is implemented by splitting the software into multiple independent units of execution, like modules and executing them in parallel. In other words, concurrency provides capability to the software to execute more than one part of code in parallel to each other. It is necessary for the programmers and designers to recognize those modules, which can be made parallel execution.

**Coupling and Cohesion**, When a software program is modularized, its tasks are divided into several modules based on some characteristics. As we know, modules are set of instructions put together in order to achieve some tasks. They are though, considered as single entity but may refer to each other to work together. There are measures by which the quality of a design of modules and their interaction among them can be measured. These measures are called coupling and cohesion.

Thus this document initially describes the different modules present in the project. It is then followed by the algorithms that are used to implement the modules. Later the system architecture is described which is followed by the description of the GUI and then various UML diagrams like class diagram, sequence diagram and data flow diagram are given. Finally, the document ends with the list of references used in developing this document.

The different modules needed and their description for Smart Health Care Monitoring system are given below

1. **Heart Beat Sensor** **Module**: This module is responsible for identifying the change in heart rate, The Heart Beat Sensor provides a simple way to study the heart's function. This sensor monitors the flow of blood through Finger. As the heart forces blood through the blood vessels in the Finger, the amount of blood in the Finger changes with time. The sensor shines a light lobe (small High Bright LED) through the ear and measures the light that is transmitted to LDR. The signal is amplified, inverted and filtered, in the Circuit. Usually Heart rate will be calculated for 1Minute. For a healthy human being we get heart rate of 72 pulse rate per 1 minute. For real time applications we cannot wait for 1minute each time because if there is any disturbance in calculating for ex: if patient is not properly keep his finger inside the device means again we have to take the readings. For this we are following averaging & sampling method. In this we are calculating heart rate for each 5 seconds & we are replacing that value in array of 12 characters. After replacing each value we will add the entire array. After the 12th value we will replace the 13th value on the 1st array element so that we will get average heart rate value for 1 minute. For 5second approximately we get rating of 6.
2. **Raspberry pi 2 model B Module:** Raspberry Pi is a mini computer which is of the size of a credit card . The operating system is called Raspbian OS which is simple and is optimized for Raspberry pi. It’s an open source operating system based on Debian. Once the operating system has been loaded in the Raspberry Pi using the SD card which is of class 10 or higher. The Raspberry pi 2 it has A 900mhz quad-core Arm cortex-a7 CPU, 1gb Ram, 4 USB ports, GPIO 40 pins, Full HDMI port, Ethernet port, Combined 3.5mm audio jack and composite video, Camera interface (CSI),Display interface (DSI),Micro SD card slot ,Video core 1v3d graphics core.
3. **Temperature Sensing Module (LM35):** The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4˚C at room temperature and ±3⁄4˚C over a full −55 to +150˚C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35’s low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1˚C in still air. The LM35 is rated to operate over a −55˚ to +150˚C temperature range.
4. **RFID Module:** Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders. The technology requires some extent of cooperation of an RFID reader and an RFID tag. Radio frequency identification (RFID) technology has been in use for several decades to track and identify goods, assets and even living things. Recently, however, RFID has generated widespread corporate interest as a means to improve supply chain performance. Market activity has been exploding since Wal-Mart's June 2003 announcement that its top 100 suppliers must be RFID-compliant by January 2005. Mandates from Wal-Mart and the Department of Defense (DoD) are making many companies scramble to evaluate, select and implement solutions that will make them compliant with their customers' RFID requirements and additional retailers and other large supply chain channel masters are likely to follow suit.
5. **GSM Module:** Global System for Mobile communications (GSM: originally from Grouped Special Mobile) is the most popular standard for mobile phones in the world. Its promoter, the GSM Association, estimates that 82% of the global mobile market uses the standard. GSM is used by over 3 billion people across more than 212 countries and territories. Its ubiquity makes international roaming very common between mobile phone operators, enabling subscribers to use their phones in many parts of the world. GSM differs from its predecessors in that both signalling and speech channels are digital, and thus is considered a second generation (2G) mobile phone system. This has also meant that data communication was easy to build into the system.
6. **GPRS Module:** General Packet Radio Services (**GPRS**) is a packet-based wireless communication service that promises data rates from 56 up to 114 Kbps and continuous connection to the Internet for mobile phone and computer users.
7. **Database Module:** This module’s function is to insert the data into the Android Application and provide persistent storage for all the activities performed by the user and will be helpful when the user/doctor needs to monitor the user’s activities.
8. **Email Module:** This module’s function is to send an Email alert to the intended recipient’s Gmail account along with the current position of the user which is obtained using GPS.
9. **Web Services (AWS):** AWS IoT is a platform that enables you to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline.
10. **Android Application Module:** This module’s function is to display the Data generated from the various Sensors that are connected to Raspberry pi. Different data such as Heart rate, temperature , location and User unique RFID is are taken into consideration and are notified to the guardian and Doctor .Basically this module gives the overall status of the patient to the intended person

**Algorithm Design**

Input: Raspberry Pi with all the Sensor connections.

Output: Updating Heart Rate Sensor values and Temperature value to AWS and Android Application Module, sending Email Alert and SMS Alerts.

1. R 🡨 Raw data from the Heart Rate sensors.
2. Ra 🡨 P (R) // parsing the raw data to obtain the actual analog voltage values.
3. Rd 🡨 ADC(Ra) //Converting to digital voltage value.
4. Monitor(Rd) /\* Displaying appropriate messages and digital voltage value on Monitor. \*/
5. App 🡨 Rd // Send data to Android Application.
6. Loop (Keep Track of Threshold value Being Sensed)

Send SMS Alert.

Send Email Alert using phone’s Internet with location as attachment.

Buzzer 🡨 ON

1. If (Temperature Exceeds)

Variable1 🡨 Store Time, Date and Location.

Wc++

1. Else IF (Heart Rate Data Exceeds)

Variable2🡨 Store Time, Date and Location.

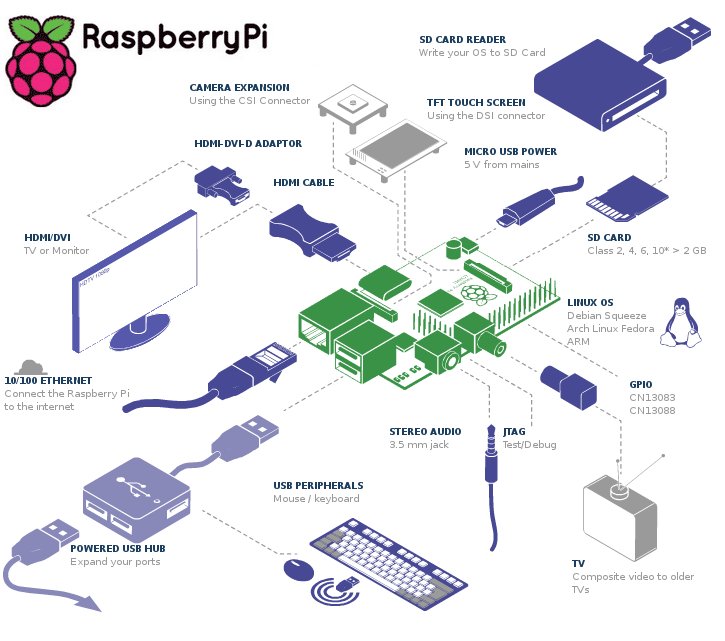
Wc++

1. Loop back to step 1.

**ARCHITECTURE DESIGN**

A **system architecture** or **systems architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behaviour) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. The different module present in this project has been described in the introduction. The system architecture is given in Figure1.



**Figure1. Raspberry pi System Architecture**

RFID Tag

GPS /GPRS

Raspberry PI 2

Wearable device

Heart Monitor

Temperature Monitor

*Amazon Web* Services

Data Base

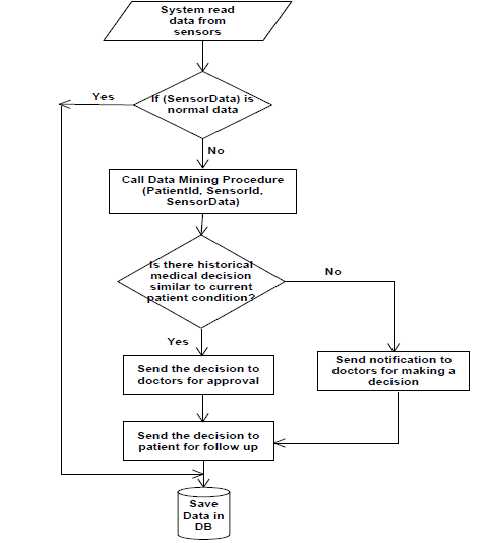
Monitor \Laptop



Heart rate and temperature data can be analyzed



**Figure2. Project Architecture.**

****

**Figure3. Flow Diagram.**

**GRAPHICAL USER INTERFACE**

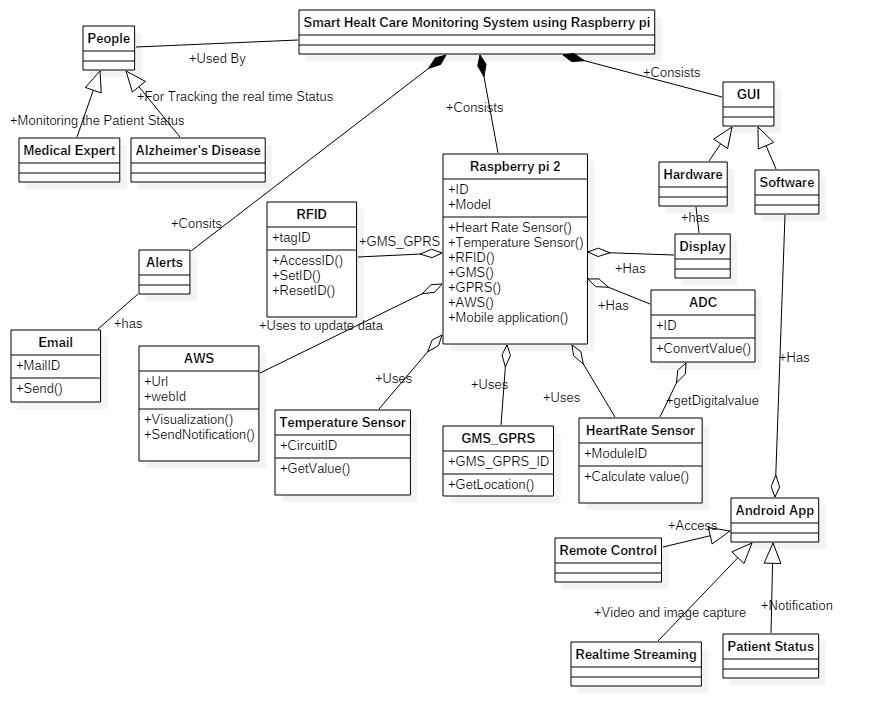
The main graphical user interface is an Android Application which communicates with the Raspberry pi board and also which notifies about the patient status. This application initially has a welcome login page for authentication with two buttons, namely login and clear. When the user enters the correct username and the password the login succeeds and redirects the application to the main page shows the graph of user heart rate and temperature. If either the username or the password is invalid, then it prompts the user to enter the correct credentials. Thus the user’s status such as Heart rate, Temperature, Location, RFID can be tracked both on the AWS and cell phone. Further the Android application also all user to look at the location from the Raspberry pi camera and capture the real time picture. The GUI has been planned to designed very efficiently so that it’s user friendly and at the same time has a good look and feel.

Also there is an monitor or laptop which is directly connected to the Raspberry pi board which provides a restricted user interface as it allows the user to only view the data being displayed on it and have complete control over the board using the VNC client module and this VNC control is also available on the Mobile in the form of Android application.

**CLASS DIAGRAM**

Classes are depicted as boxes with three sections, the top one indicates the name of the class, the middle one lists the attributes of the class, and the third one lists the methods. By including both an attribute and a method box in the class I'm arguably making design decisions in my model, something I shouldn't be doing if my goal is conceptual modeling. Another approach would be to have two sections, one for the name and one listing responsibilities. This would be closer to a [CRC model](http://agilemodeling.com/artifacts/crcModel.htm) (so if I wanted to take this sort of approach I'd use CRC cards instead of a UML class diagram). I could also use class boxes that show just the name of the class, enabling me to focus on just the classes and their relationships. However, if that was my goal I'd be more likely to create an [ORM diagram](http://agilemodeling.com/artifacts/ormDiagram.htm) instead. In short, I prefer to follow AM's [*Apply the Right Artifact(s)*](http://agilemodeling.com/practices.htm#ApplyTheRightArtifacts) practice and use each modeling technique for what it's best at.

* In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
* The figure in the next slide represents the class diagram for wireless temperature monitoring system which uses association, aggregation, composition and generalization and many more features of the class model.
* The relationship among various classes and their interdependencies are effectively modeled using the class diagram.
* The class diagram for this project is shown in Figure2.

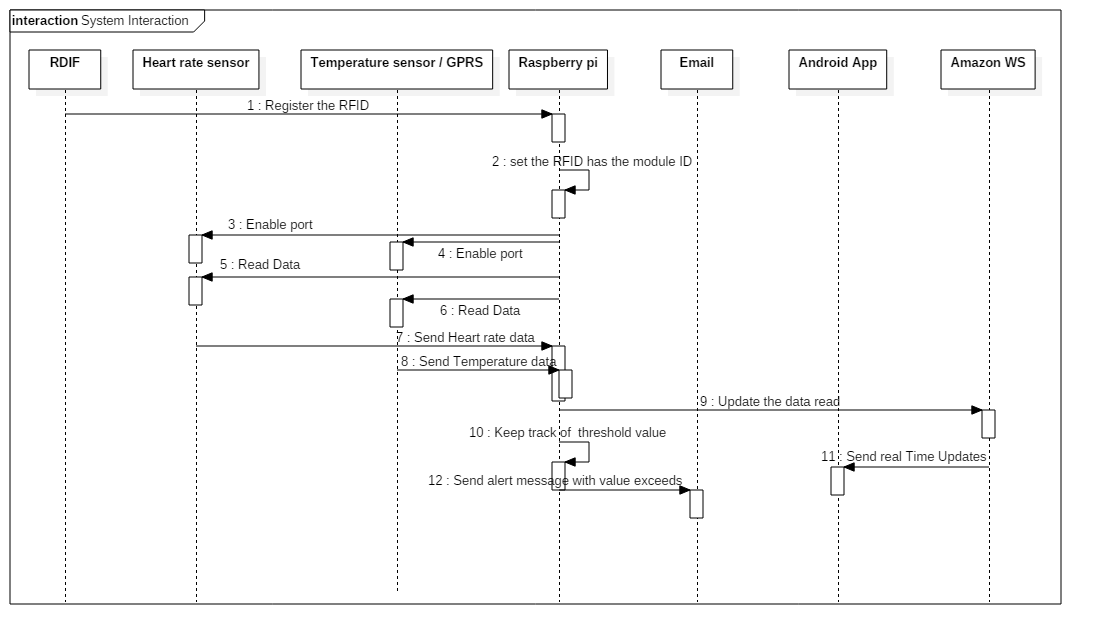


**Figure4. Class Diagram**

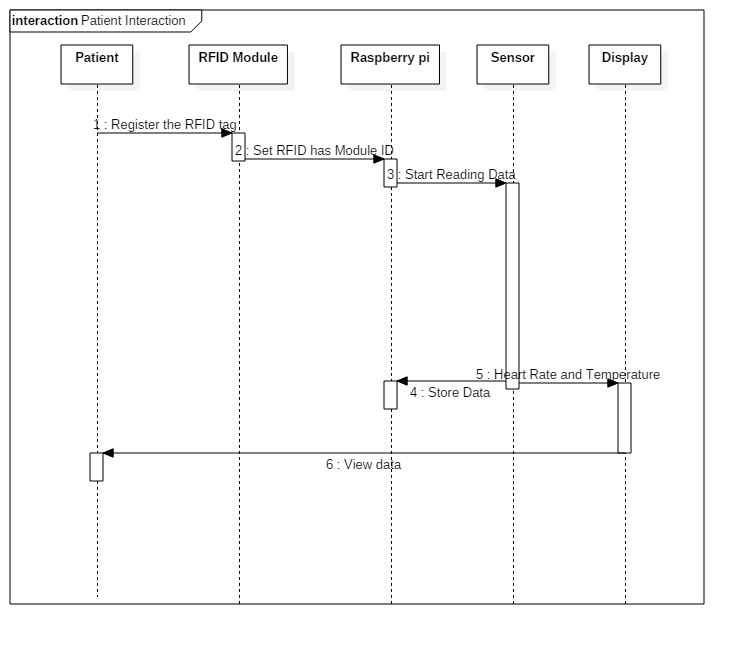
**SEQUENCE DIAGRAM**

**Sequence diagram** is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

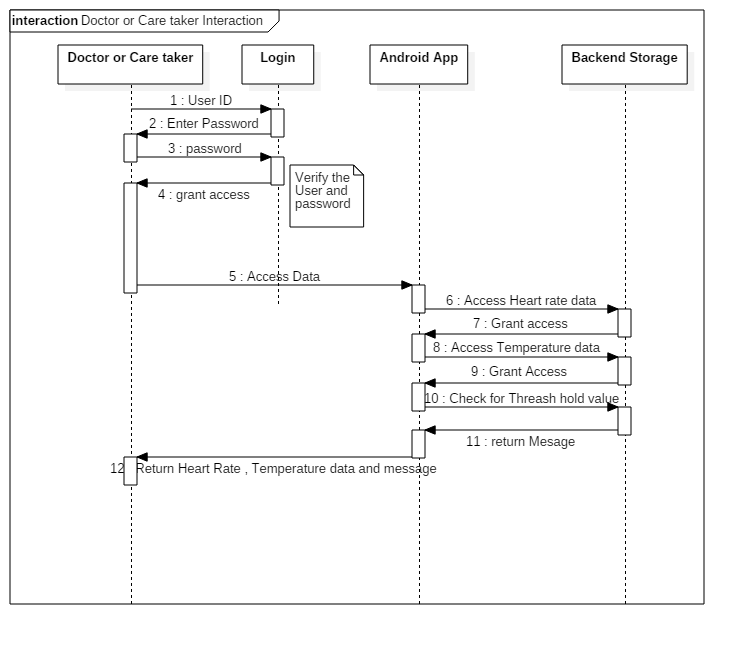
A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. The sequence diagrams for this project are as follows.

****

**Figure5. Sequence Diagram for Smart Health Care Monitoring.**

****

**Figure6. Sequence Diagram for Patient interaction.**

****

**Figure7. Sequence Diagram for Doctor Interaction.**

**DATA FLOW DIAGRAM**

A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

Appropriate Domain of Application:

* DFDs are excellent guide for validating the compatibility of the process and designs of the system. This is because in order to design applications successfully, especially large ones, the design of both the processes and the data stores is important. In addition, the data must be consistent with each other. For example, there must be process to store the data in the data stores and the data stores must supply the data views accessed by the processes. Since DFDs depict the relationships between processes, data store, and data views, this made DFD the perfect guide for validating compatibility.
* DFDs are appropriate diagrams for designing high-level application architecture. This is because it is a fact that the larger the application is to be developed the more important the architecture is. For example, building a box does not need an architect but a 10-story building does. In most architectural design, they are represented as diagrams because diagrams are the best way to depict multiple relationships among multiple components. This is applicable to software design, too and DFDs helps tremendously in showing the architecture design of the system r application.
* DFDs are especially useful for depicting system flow charts. DFDs are used to show the flows of data among batch-job steps.

**Untitled Diagram1.png**

**Figure8. Data Flow Diagram.**