**M S RAMAIAH INSTITUTE OF TECHNOLOGY**

(An Autonomous Institute, Affiliated to VTU)

MSR NAGAR, MSRIT POST, Bangalore-54

A Dissertation Report on

**HEALTH CARE MONITORING SYSTEM**

Submitted by

**VISHAL MAHULI** 1MS12CS134

**ANVITHA D RAO** 1MS12CS010

**VISHWANATH K.** 1MS12CS135

**ADARSH RAJ** 1MS12CS005

*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**

#### Contents

***Declaration i***

***Acknowledgements ii***

***Abstract iii***

***List of Figures***

***List of Tables***

1 **INTRODUCTION**

* 1. General Introduction………………. **7**
  2. Statement of the Problem…………..
  3. Objectives of the project……………
  4. Project deliverables……………
  5. Current Scope………………………
  6. Future Scope……………………….

1. **PROJECT ORGANIZATION 11**
   1. Software Process Models
   2. Roles and Responsibilities
2. **LITERATURE SURVEY 13**

3.1….Introduction

3.2…Main Body

3.3 Conclusion of Survey

1. **PROJECT MANAGEMENT PLAN 15**
   1. Schedule of the Project
   2. Effort Estimation
   3. Risk Identification
2. **SOFTWARE REQUIREMENT SPECIFICATIONS 19**

5.1 Product Overview

5.2 External Interface Requirements

5.2.1 User Interfaces

5.2.2 Hardware Interfaces

5.2.3 Software Interfaces

5.2.4 Communication Interfaces

5.3 Functional Requirements

5.3.1 Functional Requirement 1.1

:

5.3.n Functional Requirement 1.n

5.4 Software System Attributes

5.4.1 Reliability

5.4.2 Availability

5.4.3 Security

5.4.4 Portability

5.4.5 Maintainability

5.4.6 Performance

5.5 Performance Requirements

5.6 Database Requirement

5.7 Design Constraints

5.8 Other Requirements

1. **DESIGN 24** 
   1. Introduction
   2. Architecture Design
   3. Graphical User Interface
   4. Class Diagram and Classes (represent Inheritance, Aggregation and Association)
   5. Sequence Diagram
   6. Data flow diagram
   7. Metric calculation
2. **IMPLEMENTATION 29**
   1. Source Code
3. **TESTING 35**

**7.1** Results and Snapshots

1. **CONCLUSION & SCOPE FOR FUTURE WORK 46**
2. **REFERENCES 46**

### DECLARATION

I hereby declare that the project entitled “***HEALTHCARE MONITORING SYSTEM***” submitted for the " **PROJECT BASED LEARNING (CSPE734)** " is our original work and the project has formed the basis for the award of B.E degree in Computer Science.

Signature of the Students:

Place:

Date:

**ACKNOWLEDGEMENT**

First and foremost we would like to thank Mrs. Sowmya B. J (Professor), Department of Computer Science and Engineering, M. S. Ramaiah Institute of technology, Bangalore, who was also our supervisor, for their consistent guidance on each and every step of our project and also our HOD Dr. K.G.Srinivasa(Professor). Without their push and motivation, this project would have not been completed. Their continuous support and help made this project feasible. Without her support we would not have been able to excel in our project. Lastly, we would like to thanks all our friends and teammates for believing in each other and our capabilities.

# ABSTRACT

The internet of things with cloud computing is a new trend in managing online data and monitoring them .With the increase in the number of Internet users over the past decade has made internet a part of our life. IOT is one of the emerging Internet technologies which let the network devices sense and collect data around us and share it across the internet where people can utilize and process the data. Health care system using Iot that uses computers  to  provide an efficient and easy way to monitor the health of a patient remotely and provide immediate assistance when required. It employs the integration of cloud networking, wireless communication, to store the patient’s health details thus keeping the users updated about the patient’s health.

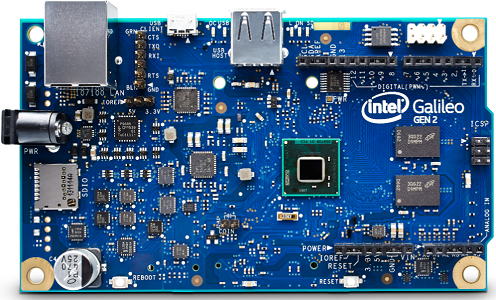
In this paper we present a health care system (HCS) using Intel Galileo gen2. In such a system the body sensors in/around the patient gather health information, which are stored in the cloud. When there is a high variation in body temperature and pulse rate, health care clinic staffs are alerted through which they can provide immediate assistance and alert the doctor and provide treatment by referring the catalogue. This system targets aged people who live alone with comparatively good health condition and also people  in critical conditions who need constant surveillance. This system is designed to be low cost and expandable.

**1. INTRODUCTION**

* 1. **General Introduction**
* **The Intel® Galileo Gen 2** development board is a microcontroller board based on the Intel® Quark**™**SoC X1000 application processor, a 32-bit Intel® Pentium® brand system on a chip (SoC). It is the first board based on Intel® architecture designed to be hardware and software pin-compatible with shields designed for the Arduino Uno\* R3.
* This platform provides the ease of Intel architecture development through support for the Microsoft Windows\*, Mac OS\*, and Linux\* host operating systems. It also brings the simplicity of the Arduino integrated development environment (IDE) software.
* The Intel Galileo Gen 2 board is also software-compatible with the Arduino software development environment, which makes usability and introduction a snap. In addition to Arduino hardware and software compatibility, the Intel Galileo Gen 2 board has several PC industry standard I/O ports and features to expand native usage and capabilities beyond the Arduino shield ecosystem. A full-sized mini-PCI Express\* slot, 100 Mb Ethernet port, Micro-SD slot, 6-pin 3.3V USB TTL UART header, USB host port, USB client port, and 8 Mbyte NOR Flash\* come standard on the board.

* Intel Galileo Gen 2 improves on Gen 1 by replacing the RS-232 console port with a 6-pin 3.3V USB TTL UART header. New additions to the Intel Galileo Gen 2 board include 12-bit pulse-width modulation (PWM), console UART1 redirection to Arduino\* headers, 12V Power-over-Ethernet (PoE) capability, and a power regulation system that accepts power supplies from 7V to 15V.

* The genuine Intel® processor and surrounding native I/O capabilities of the SoC provides for a fully featured offering for both the maker community and students alike. It will also be useful to professional developers who are looking for a simple and cost effective development environment to the more complex Intel® Atom™ processor and Intel® Core™ processor-based designs.



**Intel Galileo Gen2 Board**



Groove sensor kit

* 1. **Problem statement**

The personal health monitoring of each individual is considered very important because of rise in health problems in today's world. The increasing stressful lifestyle is taking maximum toll on the public health. The researchers and surveys often demonstrate that most of the major health ailments are the result of careless attitude towards the health ailments.

Moreover the increase in patients has also led to the decrease in the relative number of doctors per patient which results in vicious cycle where ignored or delayed diagnostics of an ailment makes the patient more dependent on doctor. These days it is advisable to each individual to monitor and maintain good health by using health monitoring devices which track their health conditions even in the absence of the doctor under normal and stable conditions. There are also old people staying at home independently whose health condition can be remotely accessed by their guardian and under unstable conditions can be given immediate treatment. Thus there is a need for software that utilizes the data available from the device, uploads it to the website, gets feedback from the doctors via internet and show health reports. Doctor should be able to get data anytime he wants for analysis.  Our project attempts to use the information obtained using such devices to give the detailed analysis of health of a patient/individual that can help in getting a prompt and timely assistance from doctor when there is high variations in body temperature or pulse rate.

* 1. **Project Objectives**
* Due to the increase in the number of patients and decrease in the number of doctors per patient the system aims in providing an efficient patient-focused, high quality, cost-effective services while promoting health and wellness.
* Continuous monitoring the health conditions of those patients who need constant survilance and old people living independently using appropriate sensors and thus letting the guardians live tension free.
* Collecting data from sensors continuously and storing them using cloud services and making it available to the guardian thus keeping them updated about the patients health condition.
* Efficient management of the data collected and stored and keeping track of the working conditions of the sensor.
* Thus provide a simple interfaced catalog which can be easily accessed and understood by the guardian and the doctor.
* Provide a highly available system which alerts the doctor when there is high variations in temperature and pulse rate which helps the doctor in attending to them as soon as possible.
* The health care system should benefit all people making it cost effective and easy to use and maintain.
  1. **Project Deliverables**
* Temperature and pulse rate measurement is achieved by parsing the raw data and extracting the actual values.
* Sending Gmail Alerts to the concerned doctor and the guardian of the patient whenever there is a high variation in patient’s body temperature or pulse rate is another important deliverable of the project.
* Constantly updating the temperature and pulse rate values to the local server database is the most important deliverable with respect to this project and it needs to be spontaneous in nature as the webpage must reflect the real time values.
* A webpage has been developed & configured and used to remotely access the temperature and pulse rate values measured by the sensors .
* Persistent offline storage is provided by using MySQL database.

**1.5 Current Scope of the System**

Our system is the first step towards the effective health monitoring and care systems. We have developed the system with temperature and pulse rate sensors. These sensors detect the body temperature variations & pulse beat respectively and is continuously recorded. These records maintained can be very useful for the patient’s medical history and can be analysed to see any specific patterns. Collection of such records can also help in analysing by general medical researchers on which diseases or virus infections trends. In case of any emergency such as heavy variation in body temperature or improper pulse rate, an email will be sent to the concerned doctor and the guardian whose email address will be mentioned for patient’s admission form. This part ensures that if at all any irregularities occur with patient, it will be dealt immediately and can be provided with the necessary help.

**2. PROJECT ORGANIZATION**

**2.1 Software Process Models**

Extreme programming is the best suited software development process model for wireless temperature monitoring system.

Extreme programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed *continuously*, i.e. the practice of pair programming.

XP describes four basic activities that are performed within the software development process: coding, testing, listening, and designing. Each of those activities is described below.

#### Coding

The advocates of XP argue that the only truly important product of the system development process is code – software instructions that a computer can interpret. Without code, there is no working product.

Coding can also be used to figure out the most suitable solution. Coding can also help to communicate thoughts about programming problems. A programmer dealing with a complex programming problem, or finding it hard to explain the solution to fellow programmers, might code it in a simplified manner and use the code to demonstrate what he or she means. Code, say the proponents of this position, is always clear and concise and cannot be interpreted in more than one way. Other programmers can give feedback on this code by also coding their thoughts.

#### Testing

Extreme programming's approach is that if a little testing can eliminate a few flaws, a lot of testing can eliminate many more flaws.

* Unit tests determine whether a given feature works as intended. A programmer writes as many automated tests as they can think of that might "break" the code; if all tests run successfully, then the coding is complete. Every piece of code that is written is tested before moving on to the next feature.
* Acceptance tests verify that the requirements as understood by the programmers satisfy the customer's actual requirements.

System-wide integration testing was encouraged, initially, as a daily end-of-day activity, for early detection of incompatible interfaces, to reconnect before the separate sections diverged widely from coherent functionality. However, system-wide integration testing has been reduced, to weekly, or less often, depending on the stability of the overall interfaces in the system.

#### Listening

Programmers must listen to what the customers need the system to do, what "business logic" is needed. They must understand these needs well enough to give the customer feedback about the technical aspects of how the problem might be solved, or cannot be solved. Communication between the customer and programmer is further addressed in the *planning game*.

#### Designing

From the point of view of simplicity, of course one could say that system development doesn't need more than coding, testing and listening. If those activities are performed well, the result should always be a system that works. In practice, this will not work. One can come a long way without designing but at a given time one will get stuck. The system becomes too complex and the dependencies within the system cease to be clear. One can avoid this by creating a design structure that organizes the logic in the system. Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.

**2.2 Roles and Responsibility:**

1.**Vishal:** Is our team leader who was responsible for assigning tasks to team members and also ensuring that it goes according to the plan so that  project could be completed on time with the specified requirements. Also was instrumental in conducting brain storming sessions and writing the code for sending an email alert. Responsible for writing the abstract and introduction of the project in research paper, final report and presentation.

2.**Vishwanath:** Was assigned the task of writing code for pushing data into the local server and creating the front end for the user. Created an easy and efficient web interface. Also was responsible in writing the design and implementation part of the research paper. Played an important role in the final report and presentation as well.

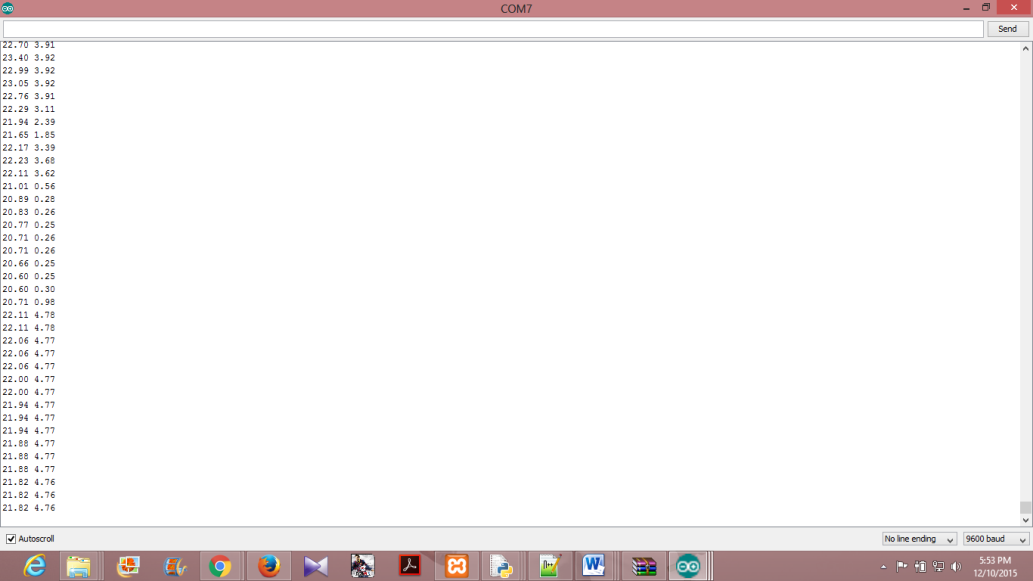
3.**Anvitha:** Was assigned the task of extracting raw data from the sensors and parsing it to obtain the temperature values in degree Celsius and also writing code for obtaining temperature values from the patient’s body to the IDE. Testing and comparison part of the research paper was her responsibility along with the final report and presentation.

4.**Adarsh:** Was assigned the task of extracting raw data from the sensors and parsing it to obtain the pulse values  and also writing code for obtaining pulse values from the patient’s body to the IDE. Scope and future scope part of the research paper was his responsibility along with the final report and presentation.

**3. Literature Survey**

**3.1 Reading temperature values from sensors**

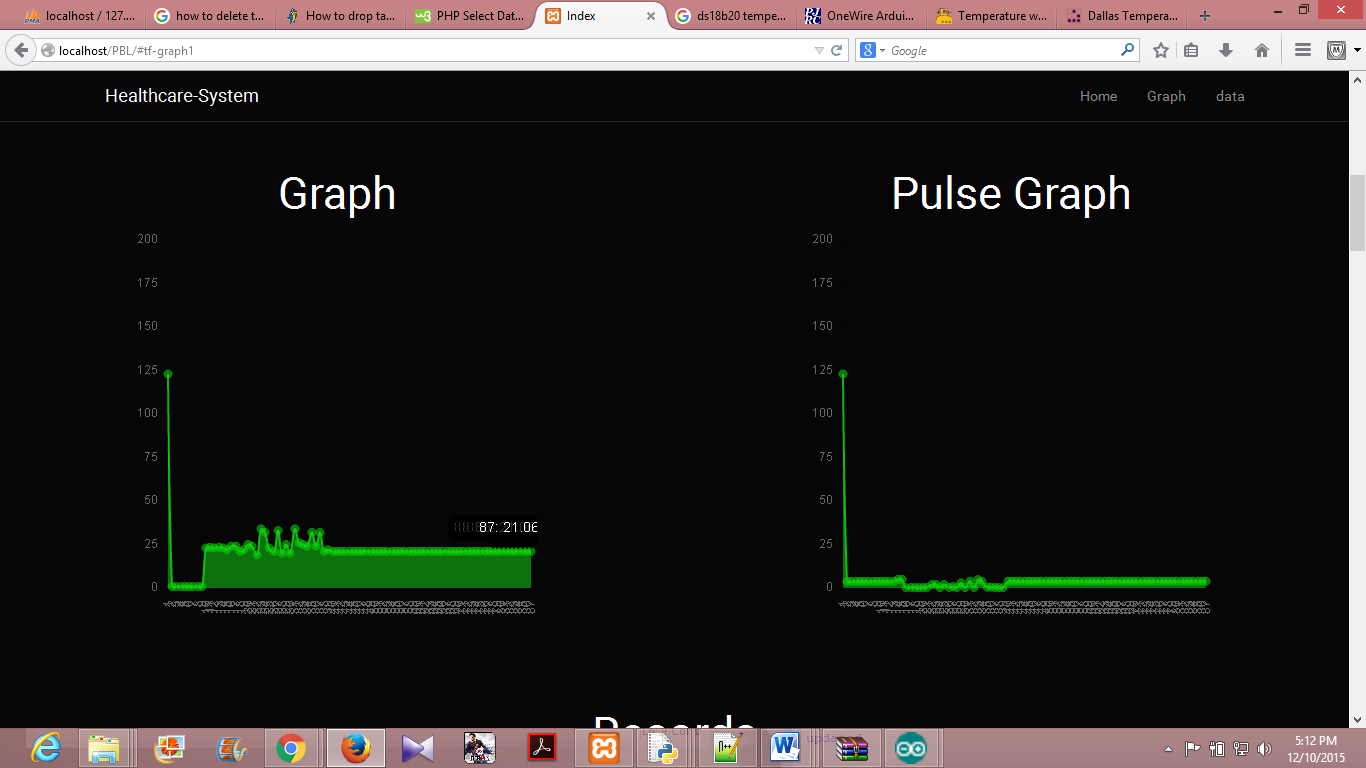
The data is read from the Groove temperature sensor and TCRT1000 heartbeat rate sensor and the values collected are passed on to the Galileo Gen2 board for further processing.



**3.2 Gmail alerts and Cloud Updates**

The temperatures and pulse rates are constantly updated to the local web server. When the temperature exceeds the threshold value, an email to the intended recipient’s Gmail account is sent from the system. The values collected are reflected from the values read to the webpage (front end). The Gmail alerts are implemented using python code where the sender’s username, password and the intended recipient’s username must be known for sending the Email.

Further on the webpage, apart from providing display facilities also provides pictorial representation of data in the form of graphs . This helps in analyzing the data that is collected over a period of time.



**4. PROJECT MANAGEMENT PLAN**

**4.1 Project Schedule**

In project management, a **schedule** is a listing of a project's milestones, activities, and deliverables, usually with intended start and finish dates. Those items are often estimated in terms of resource allocation, budget and duration, linked by dependencies and scheduled events. A schedule is commonly used in project planning and project portfolio management parts of project management. Elements on a schedule may be closely related to the work breakdown structure (WBS) terminal elements, the Statement of work, or a Contract Data Requirements List.

Before a project schedule can be created, the schedule maker should have a work breakdown structure (WBS), an effort estimate for each task, and a resource list with availability for each resource. If these components for the schedule are not available, they can be created with a consensus-driven estimation method like Wideband Delphi. The reason for this is that a schedule itself is an estimate: each date in the schedule is estimated, and if those dates do not have the buy-in of the people who are going to do the work, the schedule will be inaccurate.

In order for a project schedule to be healthy, the following criteria must be met

* The schedule must be constantly (weekly works best) updated.
* The EAC (Estimation at Completion) value must be equal to the baseline value.
* The remaining effort must be appropriately distributed among team members (taking vacations into consideration).

**4.2 Effort Estimation**

**Software development effort estimation** is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete, uncertain and noisy input. Effort estimates may be used as input to project plans, iteration plans, budgets, investment analyses, pricing processes and bidding rounds.

There are many ways of categorizing estimation approaches, see for example. The top level categories are the following:

* Expert estimation: The quantification step, i.e., the step where the estimate is produced based on judgmental processes.
* Formal estimation model: The quantification step is based on mechanical processes, e.g., the use of a formula derived from historical data.
* Combination-based estimation: The quantification step is based on a judgmental and mechanical combination of estimates from different sources.

The evidence on differences in estimation accuracy of different estimation approaches and models suggest that there is no “best approach” and that the relative accuracy of one approach or model in comparison to another depends strongly on the context .This implies that different organizations benefit from different estimation approaches. Findings, summarized in that may support the selection of estimation approach based on the expected accuracy of an approach include:

* Expert estimation is on average at least as accurate as model-based effort estimation. In particular, situations with unstable relationships and information of high importance not included in the model may suggest use of expert estimation. This assumes, of course, that experts with relevant experience are available.
* Formal estimation models not tailored to a particular organization’s own context, may be very inaccurate. Use of own historical data is consequently crucial if one cannot be sure that the estimation model’s core relationships (e.g., formula parameters) are based on similar project contexts.
* Formal estimation models may be particularly useful in situations where the model is tailored to the organization’s context (either through use of own historical data or that the model is derived from similar projects and contexts), and it is likely that the experts’ estimates will be subject to a strong degree of wishful thinking.

**4.3 Risk Identification**

**Risk management** is the identification, assessment, and prioritization of risks (defined in ISO 31000 as *the effect of uncertainty on objectives*) followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate eventsor to maximize the realization of opportunities. Risk management’s objective is to assure uncertainty does not deflect the endeavor from the business goals.

Risks can come from various sources: e.g., uncertainty in financial markets, threats from project failures (at any phase in design, development, production, or sustainment life-cycles), legal liabilities, credit risk, accidents, natural causes and disasters as well as deliberate attack from an adversary, or events of uncertain or unpredictable root-cause. There are two types of events i.e. negative events can be classified as risks while positive events are classified as opportunities. Several risk management standards have been developed including the Project Management Institute, the National Institute of Standards and Technology, actuarial societies, and ISO standards. Methods, definitions and goals vary widely according to whether the risk management method is in the context of project management, security, engineering, industrial processes, financial portfolios, actuarial assessments, or public health and safety.

Risk sources are more often identified and located not only in infrastructural or technological assets and tangible variables, but also in human factor variables, mental states and decision making. The interaction between human factors and tangible aspects of risk highlights the need to focus closely on human factors as one of the main drivers for risk management, a "change driver" that comes first of all from the need to know how humans perform in challenging environments and in face of risks (Daniele Trevisani, 2007). As the author describes, «it is an extremely hard task to be able to apply an objective and systematic self-observation, and to make a clear and decisive step from the level of the mere "sensation" that something is going wrong, to the clear understanding of how, when and where to act. The truth of a problem or risk is often obfuscated by wrong or incomplete analyses, fake targets, perceptual illusions, unclear focusing, altered mental states, and lack of good communication and confrontation of risk management solutions with reliable partners. This makes the Human Factor aspect of Risk Management sometimes heavier than its tangible and technological counterpart»

Strategies to manage threats (uncertainties with negative consequences) typically include avoiding the threat, reducing the negative effect or probability of the threat, transferring all or part of the threat to another party, and even retaining some or all of the potential or actual consequences of a particular threat, and the opposites for opportunities (uncertain future states with benefits).

**5. SOFTWARE REQUIREMENT SPECIFICATIONS**

1. **Product overview**

Healthcare Monitoring System: The aim of this project is to measure the temperature and pulse rate of the patient’s body using Galileo Gen2. We have used one Groove temperature sensors and aTCRT1000 pulse rate sensor to measure the real time temperature and pulse rate and update it in the local server database. The temperatures and pulse rate values are constantly monitored and printed on the console. Also an email alert is sent to the doctor and guardian in case if any high variation recorded in patient’s body. These recorded values can be later analysed for further efficient medication.

1. **External Interface Requirements**
2. **User Interfaces**

The real time temperature and pulse rate can be accessed from an interactive user interface webpage customized for Groove temperature sensor and TCRT1000 pulse rate sensors and Intel Galileo Gen2. The requirements of the user interfaces are that it needs to be highly interactive, efficient, and attractive but should yet be simple and must possess a sleek look. Also the interface must be able to accept input from a variety of devices and display the information in an organized manner using graphs. The user interfaces include a webpage.

1. **Hardware Interfaces**

Intel Galileo Gen2 is the hardware device which is responsible for co-coordinating various activities with respect to the project. It is the core component of the project. Further Groove Temperature sensors and TCRT1000 pulse rate sensors are used to measure the body temperature and heartbeat rate. **Galileo Gen2** along with the **sensors** acts as the hardware interface to the healthcare monitoring system project.

1. **Software Interfaces**

A arduino code is written to read the raw temperature and pulse rate and it’s parsed to obtain the actual temperature which is then updated to the database and also if the body temperature or pulse rate exceeds the threshold value then an email alert is sent to the doctor and the guardian using a Python code. Thus the **Python code & Arduino code** together with the **library routines** act as the software interface to the wireless temperature monitoring system project.

1. **Communication Interfaces**

An end user can communicate with the project over the internet using the webpage. The user can communicate with the sensors and Galileo Gen2 board using the Arduino IDE. SMTP is the communication protocol used to send Emails to the intended recipient.

1. **Functional Requirements**

* **Functional Requirement 1.1** Database Access. The user can access the data i.e. the temperature readings from the database that is stored on the SD card of raspberry pi (offline storage)
* **Functional Requirement 1.2** Gmail Alert. When the temperature exceeds the threshold value an Email is sent along with a photo of the surroundings by using SMTP.
* **Functional Requirement 1.3** E-healthCare application. This application remotely connects to the terminal of Intel Galileo Gen2 and can be used for obtaining the current temperature and heartbeat rate of the patient;s body.
* **Functional Requirement 1.4** Wired Connectivity: LAN Cable is used so that the sensors continuously measure the temperature and this can be uploaded to local server.
* **Functional Requirement 1.5** Temperature Measurement. Groove temperature sensor is used to measure the body temperature.
* **Functional Requirement 1.6** Heartbeat Rate Measurement : TCRT1000 sensor is used to measure the pulse rate of the patient’s body.
* **Functional Requirement 1.7** Database. The database is used for storing temperature values and providing offline storage.

1. **Software System Attributes**

* **Reliability:** This system is available all time and mainly the reliability of the system depends on the sensors and the board. As the lifespan of board and sensors is more, the reliability factor is also more.
* **Availability:** As it’s the health care system, there has to be continuous sensing of data and monitoring. As a result they have to be available all time.
* **Security:** The catalogue is made available only to the doctors and the hospital authorities. A separate login page is made available only to the guardian through a login page which requires authentication. Thus the data is made secure by making it available only to the concerned people. Any modification to the database shall be synchronized and be done only by the database admin.

**4.4 Portability:**

* The health care system developed is portable and can run over different operating systems.

**4.5 Maintainability**

* The health care system is easily maintainable as it has fewer components and its design is not complicated. As the lifespan of sensors and the board is high, so it requires less maintenance. The system will provide capability to back-up the data.

**4.6 Performance**

The system shall give responses in 1 second after checking the patient’s information. The user-interface screen shall respond within 5 seconds. The System must support 1 person at a time.

1. **Performance requirements**

The only way in which systems will meet their performance targets is for them to be specified clearly and unambiguously. It is a simple fact that if performance is not a stated criterion of the system requirements then the system designers will generally not consider performance issues. While loose or incorrectly defined performance specifications can lead to disputes between clients and suppliers. In many cases performance requirements are never ridged as system that does not fully meet its defined performance requirements may still be released as other consideration such as time to market.

In order to assess the performance of a system the following must be clearly specified:  
• **Response Time**- The webpage gets updated within 2 minutes but the database updates and mailing it gets done within 1minute.  
• **Workload**- Even if the workload on the system is increased the performance doesn’t degrade as Intel Galileo Gen 2 has a RAM of 256MB.The workload will be continuous and uniform as the temperatures have to be measured throughout the day.  **• Scalability**- The number of users accessing the temperature and pulse rate from the webpage will just be the doctor.  
• **Platform-** The performance of the system depends on the platform that the application is run.

1. **Database Requirement:**

A table named Patient should be created to store the necessary information about the patient such as Name, Phone No, Address, Pid, Parent/Guardian info, Ward No, Disease, Doctor\_Name. Patient identification number:Pid will be the key attribute of the table used to associate it with another table Catalog. Catalog table stores the values such as heart rate, pulse rate, body temperature per every patient including the date and time of every reading.  
**Database Update:**  
The values read by the sensors will be continuously passed on to the database at specific intervals of time. The database gets filled with these values on a normal basis. Whenever the authorized person wishes to access the database for information or for update, he/she must go through an authentication protocol and then will be able to access or update it.

1. **Design Constraints**

* **Hard drive space**: The amount of space an application needs for storage and execution purpose is major design limitation as Intel Galileo Gen2 uses a SD card for persistent storage.
* **Application memory Usage**: The amount of memory space needed for an application to run is an important limitation in the design as Intel Galileo Gen2 has only 256MB RAM and it needs to effectively utilize it.
* **Budget**: The amount of money that can be spent in the overall development of the project restricts the design of the system.
* **Application Quality:** If the quality of the application is high then it means that the design is highly effective. Thus the quality of the application is an important deciding factor in the design of an application.

**6.DESIGN**

The proposed model of the health care system is as below described. The model consists of different sensors like temperature and pulse sensor. In the proposed model the temperature and pulse rate  variations of a person are monitored. Initially the Intel Galileo  gen2 connects to the internet through LAN  . When the connection is established it will start reading the parameters of sensors like p1, p2 etc. The threshold levels for the required sensors are set as t1, t2 etc. The sensors data are collected and sent to the local server . The data can be analyzed anywhere any time. If the sensor parameters are greater than the threshold level then an email is sent to the corresponding guardian and the health care clinic staff and the required action is taken based on the condition of the patient for the controlling of  parameters which have crossed the threshold value. The temperature and the pulse rate are stored on the local server for analysis using mysql. Even under normal conditions the guardian or the doctor  can check the patients catalog to see the patients past health conditions which helps them in detecting the problem .The user can also monitor the patients health through the internet via web interface provided.Thus we have continuous monitoring of patients body temperature and pulse , in turn keeping the clinical staff and the guardian updated of the patients health.

**Proposed System Feature**

The proposed system is a distributed health care system, consists of server and  sensors. Server controls and monitors the various sensors, and can be easily configured to handle more hardware interface module (sensors). The Intel Galileo Gen2 development board, with built in LAN port , makes the data available all time . Health System can be accessed from the web browser of any local PC in the same LAN using server IP, with appropriate web browser through server real IP (internet IP).  LAN  is selected to be the network infrastructure that connects server and the sensors.We are creating  a local server through xampp in which data is stored using mysql database.A web interface is created using PHP.

**Software design**

**Front End Design:**

* PHP is an HTML-embedded, server-side scripting language designed for web development. PHP stands for Hypertext Preprocessor.PHP is an open source software and it is free to download and use.Allows web developers to create dynamic content that interacts with databases. PHP supports various databases like MySQL, Oracle, Sybase, Solid, PostgreSQL, Informix etc.In this project we are using mysql.It consistes of php tags as well as html as html is embedded in php.These tags are used for processing as well as telling how the content has to be delivered.The user can see only the html content where as the php contents are hidden.
* An email that is sent using google’s gmail.

**BACK END DESIGN:**

XAMPP  is a [free and open source](https://en.wikipedia.org/wiki/Free_software) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [web server](https://en.wikipedia.org/wiki/Web_server) [solution stack](https://en.wikipedia.org/wiki/Solution_stack) package  consisting mainly of the [Apache HTTP Server](https://en.wikipedia.org/wiki/Apache_HTTP_Server), [MariaDB](https://en.wikipedia.org/wiki/MariaDB) [database](https://en.wikipedia.org/wiki/Database), and [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)) for scripts written in the[PHP](https://en.wikipedia.org/wiki/PHP) and [Perl](https://en.wikipedia.org/wiki/Perl) [programming languages](https://en.wikipedia.org/wiki/Programming_language).  makes it extremely easy for developers to create a local web server.  Everything needed to set up a web server – server application (Apache), database (Mysql), and scripting language (PHP) – is included in an extractable file.Using this the front end is created.

**Algorithm Design**

* Connect the Galileo gen2 board and configure it .
* Connect the temperature and heartbeat sensors to the board along with LAN cable.
* Run the code for extracting the raw data acquired from the sensors and start the local server.
* While(read data from sensors every 10 seconds)

{

Temp\_sensor= analog data read from sensor; // read the temperature sensed from groove temperature sensor.

Heartbeat\_sensor=analog\_data\_read from sensor; // read the heartbeat rate from the Tcrt1000.

celsius= Temp\_sensor\*0.0583497053; //apply the formula to convert the data read from sensor to convert to celcius.

pulse = Heartbeat\_sensor \* (5.0 / 1023.0); // apply the formula to convert the data read from sensor to appropriate range.

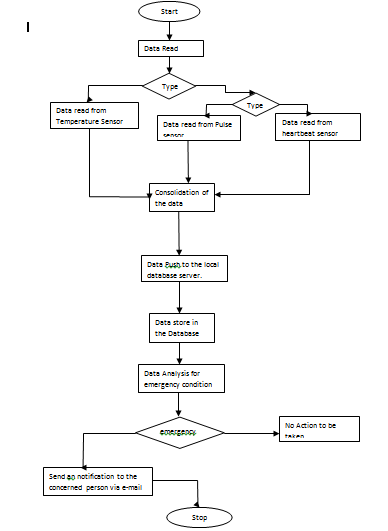
while(collected\_data>threshold) // check for higher variation in collected data

alert email sent to the doctor and guardian; // alert email

update the Celsius and pulse into the local server database;

reflect the changes in the front end;

}

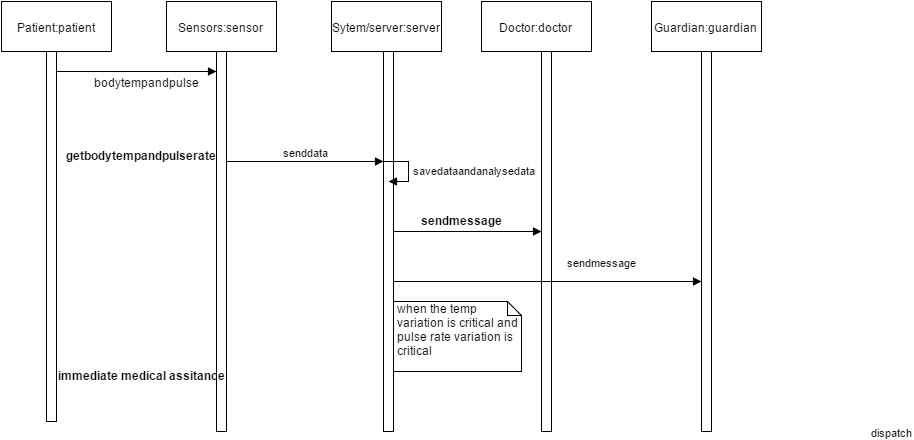


DATA FLOW DIAGRAM FOR TEMPERATURE AND PULSE RATE SENSOR

**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**.

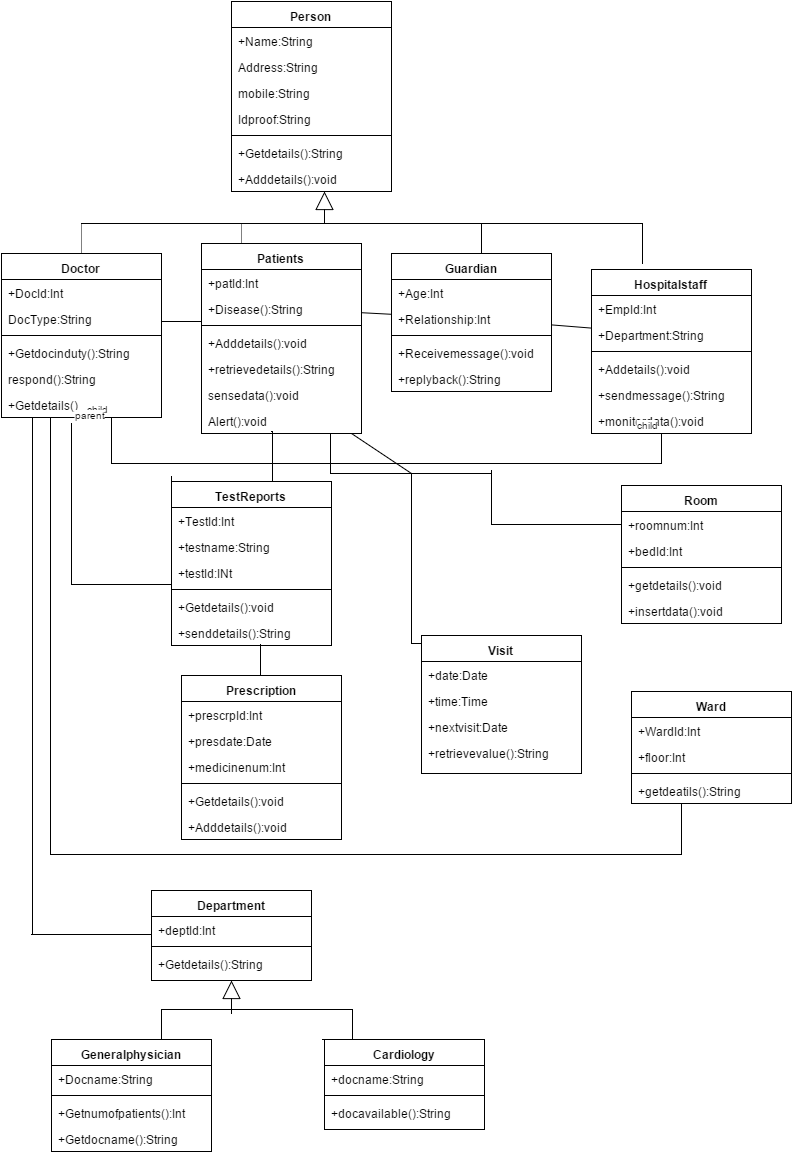
A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.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

****

**CLASS DIAGRAM**

* 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.

A class diagram is static in nature because it doesn’t represent the sequence of activities and also doesn’t bother about how system’s behavior varies from time to time due to uncertainties.

****

**METRIC CALCULATIONS**

**7. IMPLEMENTATION**

**Implementation** is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment. Many implementations may exist for a given specification or standard. For example, web browsers contain implementations of World Wide Web Consortium-recommended specifications, and software development tools contain implementations of programming languages.

A special case occurs in object-oriented programming, when a concrete class implements an interface; in this case the concrete class is an *implementation* of the interface and it includes methods which are *implementations* of those methods specified by the interface.

**Computer programming** (often shortened to **programming**, sometimes called **coding**) is a process that leads from an original formulation of a computing problem to executable computer programs. Programming involves activities such as analysis, developing understanding, generating algorithms, verification of requirements of algorithms including their correctness and resources consumption, and implementation (commonly referred to as coding of algorithms in a target programming language. Source code is written in one or more programming languages. The purpose of programming is to find a sequence of instructions that will automate performing a specific task or solving a given problem. The process of programming thus often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

Related tasks include testing, debugging, and maintaining the source code, implementation of the build system, and management of derived artifacts such as machine code of computer programs. These might be considered part of the programming process, but often the term "software development" is used for this larger process with the term "programming", "implementation", or "coding" reserved for the actual writing of source code. Software engineering combines engineering techniques with software development practices.

**SOURCE CODE:**

* **To obtain sensor values**

int outputpin= 0;

float celsius=0.0;

int rawvoltage=0;

const int analogIn = A1;

void setup()

{

Serial.begin(9600);

}

void loop()

{

int rawvoltage= analogRead(analogIn);

int sensorValue = analogRead(A0);

float millivolts= (rawvoltage/1024.0) \* 5000;

float celsius= rawvoltage\*0.0583497053;

float voltage = sensorValue \* (5.0 / 1023.0);

// Serial.print("sensor = " );

// Serial.print(rawvoltage);

// Serial.print("\t output = ");

Serial.print(celsius);

Serial.print(" ");

Serial.println(voltage);

delay(1000);

}

**TO mail in case of emergency**

import smtplib

import serial #Import Serial Library

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

sport="com6"

asO=serial.Serial(sport, 9600)

# me == my email address

# you == recipient's email address

me = "my@email.com"

you = "vishwa.kulkarni@gmail.com"

you1 = "vishalmahuli8@gmail.com"

# temp=[];

# Create message container - the correct MIME type is multipart/alternative.

msg = MIMEMultipart('alternative')

msg['Subject'] = "Temperature Emergency"

msg['From'] = me

msg['To'] = you

# Create the body of the message (a plain-text and an HTML version).

text = "There's an Emergency, The patient's Body Temperature is going crazy! He needs Help!"

# Record the MIME types of both parts - text/plain and text/html2

part = MIMEText(text, 'plain')

# Attach parts into message container.# According to RFC 2046, the last part of a multipart message, in this case

# the HTML message, is best and preferred.

msg.attach(part)

msg1 = MIMEMultipart('alternative')

msg1['Subject'] = "Heart Beat Emergency"

msg1['From'] = me

msg1['To'] = you

# Create the body of the message (a plain-text and an HTML version).

text1 = "There's an Emergency, The patient's Heart Beat is going crazy! He/She needs Help!"

# Record the MIME types of both parts - text/plain and text/html2

part1 = MIMEText(text1, 'plain')

# Attach parts into message container.# According to RFC 2046, the last part of a multipart message, in this case

# the HTML message, is best and preferred.

msg1.attach(part1)

# Send the message via local SMTP server.

mail = smtplib.SMTP\_SSL('smtp.googlemail.com', 465)

mail.ehlo

mail.login('iamyourdoctor134@gmail.com', 'pblproject')

temparature=[]

pulse=[]

while(1==1):

if(asO.inWaiting()>0):

temp=asO.readline()

r=temp.split();

temparature.append(r[0])

pulse.append(r[1])

print temparature

print pulse

if(r[0] > 22):

mail.sendmail(me, you, msg.as\_string())

mail.sendmail(me, you1, msg.as\_string())

if(r[1] >1 and r[1] < 3):

mail.sendmail(me, you, msg1.as\_string())

mail.quit()

**TO PUSH THE DATA TO A SPECIFIED IP**

/\*

Web Server

A simple web server that shows the value of the analog input pins.

using an Arduino Wiznet Ethernet shield.

Circuit:

\* Ethernet shield attached to pins 10, 11, 12, 13

\* Analog inputs attached to pins A0 through A5 (optional)

\*/

#include <SPI.h>

#include <Ethernet.h>

// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network:

byte mac[] = { 0x98, 0x4F, 0xEE, 0x01, 0x86, 0x81 };

IPAddress ip(172,1,14,96);

// Initialize the Ethernet server library

// with the IP address and port you want to use

// (port 80 is default for HTTP):

EthernetServer server(80);

int outputpin= 0;

float celsius=0.0;

int rawvoltage=0;

void setup() {

// Open serial communications and wait for port to open:

delay(5000);

Serial.begin(9600);

Ethernet.begin(mac, ip);

server.begin();

Serial.print("server is at ");

Serial.println(Ethernet.localIP());

}

void loop() {

// listen for incoming clients

EthernetClient client = server.available();

int rawvoltage= analogRead(A1);

int sensorValue = analogRead(A0);

float millivolts= (rawvoltage/1024.0) \* 5000;

float celsius= rawvoltage\*0.0583497053;

float voltage = sensorValue \* (5.0 / 1023.0);

Serial.print("Temperature in Celsius is = ");

Serial.println(celsius);

delay(1000);

Serial.print("Heart beat Voltage is = ");

Serial.println(voltage);

delay(1000);

if (client) {

Serial.println("new client");

// an http request ends with a blank line

boolean currentLineIsBlank = true;

while (client.connected()) {

if (client.available()) {

char c = client.read();

Serial.write(c);

// if you've gotten to the end of the line (received a newline

// character) and the line is blank, the http request has ended,

// so you can send a reply

if (c == '\n' && currentLineIsBlank) {

// send a standard http response header

client.println("HTTP/1.1 200 OK");

client.println("Content-Type: text/html");

client.println("Connection: close");

client.println();

client.println("<!DOCTYPE HTML>");

client.println("<html>");

// add a meta refresh tag, so the browser pulls again every 5 seconds:

client.println("<meta http-equiv=\"refresh\" content=\"5\">");

// output the value of each analog input pin

//for (int analogChannel = 0; analogChannel < 6; analogChannel++) {

//int sensorReading = analogRead(analogChannel);

// client.print("analog input ");

//client.print(analogChannel);

//client.print(" is ");

//client.print(sensorReading);

client.print("Temperature in Celsius is = ");

client.println(celsius);

client.println("<br />");

client.print("Heart beat Voltage is = ");

client.println(voltage);

client.println("<br />");

//}

client.println("</html>");

break;

}

if (c == '\n') {

// you're starting a new line

currentLineIsBlank = true;

}

else if (c != '\r') {

// you've gotten a character on the current line

currentLineIsBlank = false;

}

}

}

// give the web browser time to receive the data

delay(1);

// close the connection:

client.stop();

Serial.println("client disonnected");

}

}

**8. TESTING**

**Software testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* meets the requirements that guided its design and development,
* responds correctly to all kinds of inputs,
* performs its functions within an acceptable time,
* is sufficiently usable,
* can be installed and run in its intended environments, and
* Achieves the general result its stakeholder’s desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently..

## Testing levels

There are generally four recognized levels of tests: unit testing, integration testing, component interface testing, and system testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

### Unit testing

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis, metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

### Integration testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

### Component interface testing

The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values Unusual data values in an interface can help explain unexpected performance in the next unit. Component interface testing is a variation of black-box testing, with the focus on the data values beyond just the related actions of a subsystem component.

### System testing

System testing, or end-to-end testing, tests a completely integrated system to verify that it meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

### Operational Acceptance testing

Operational Acceptance is used to conduct operational readiness (pre-release) of a product, service or system as part of a quality management system. OAT is a common type of non-functional software testing, used mainly in software development and software maintenance projects. This type of testing focuses on the operational readiness of the system to be supported, and/or to become part of the production environment. Hence, it is also known as operational readiness testing (ORT) or Operations Readiness and Assurance (OR&A) testing. Functional testing within OAT is limited to those tests which are required to verify the *non-functional* aspects of the system.

In addition, the software testing should ensure that the portability of the system, as well as working as expected, does not also damage or partially corrupt its operating environment or cause other processes within that environment to become inoperative.

## Testing types

### Installation testing

An installation test assures that the system is installed correctly and working at actual customer's hardware.

### Compatibility testing

A common cause of software failure (real or perceived) is a lack of its compatibility with other application software, operating systems (or operating system versions, old or new), or target environments that differ greatly from the original (such as a terminal or GUI application intended to be run on the desktop now being required to become a web application, which must render in a web browser). For example, in the case of a lack of backward compatibility, this can occur because the programmers develop and test software only on the latest version of the target environment, which not all users may be running. This results in the unintended consequence that the latest work may not function on earlier versions of the target environment, or on older hardware that earlier versions of the target environment was capable of using. Sometimes such issues can be fixed by proactively abstracting operating system functionality into a separate program module or library.

### Continuous testing

Continuous testing is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with a software release candidate. Continuous testing includes the validation of both functional requirements and non-functional requirements; the scope of testing extends from validating bottom-up requirements or user stories to assessing the system requirements associated with overarching business goals.

### Software performance testing

Performance testing is generally executed to determine how a system or sub-system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

*Load testing* is primarily concerned with testing that the system can continue to operate under a specific load, whether that be large quantities of data or a large number of users. This is generally referred to as software scalability. The related load testing activity of when performed as a non-functional activity is often referred to as *endurance testing*. *Volume testing* is a way to test software functions even when certain components (for example a file or database) increase radically in size. *Stress testing* is a way to test reliability under unexpected or rare workloads. *Stability testing* (often referred to as load or endurance testing) checks to see if the software can continuously function well in or above an acceptable period.

There is little agreement on what the specific goals of performance testing are. The terms load testing, performance testing, scalability testing, and volume testing, are often used interchangeably.

Real-time software systems have strict timing constraints. To test if timing constraints are met, real-time testing is used.

### Usability testing

Usability testing is to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application.

### Accessibility testing

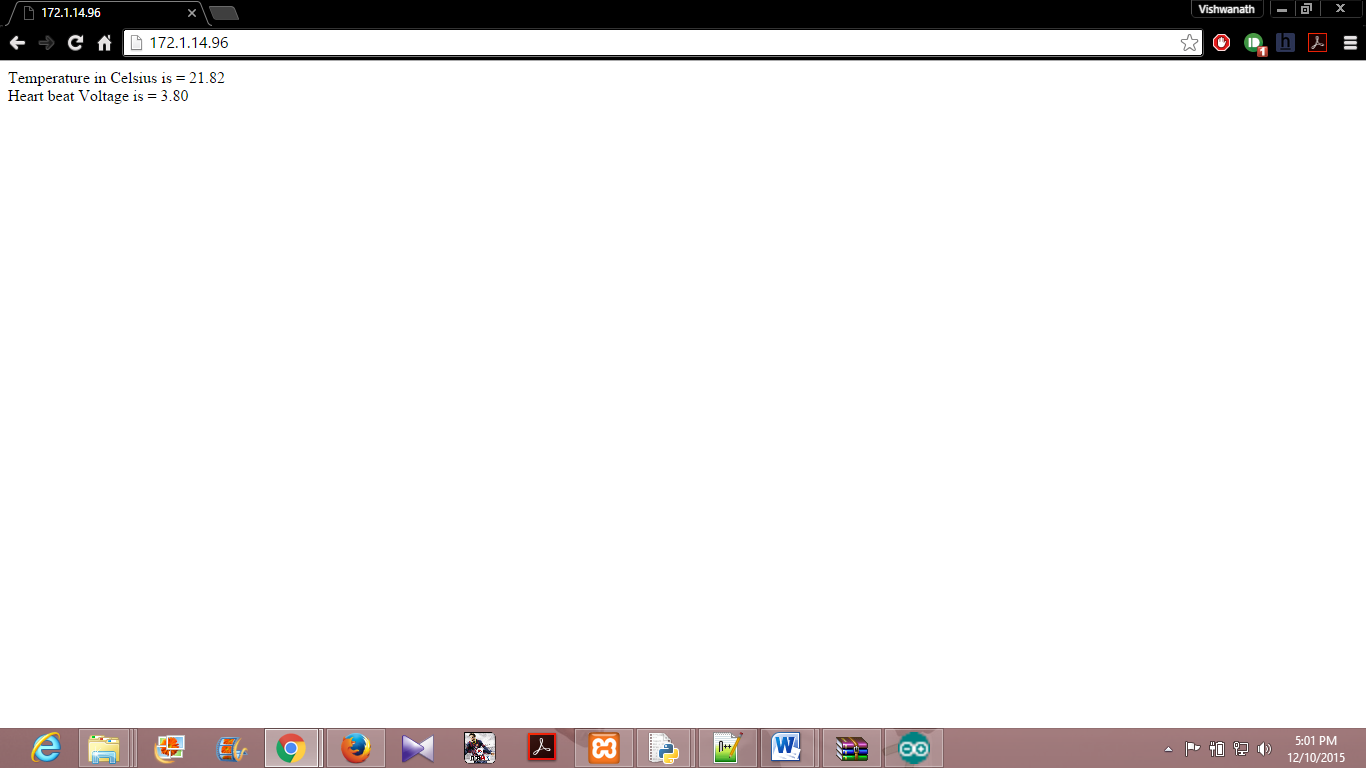
Accessibility testing may include compliance with standards such as:

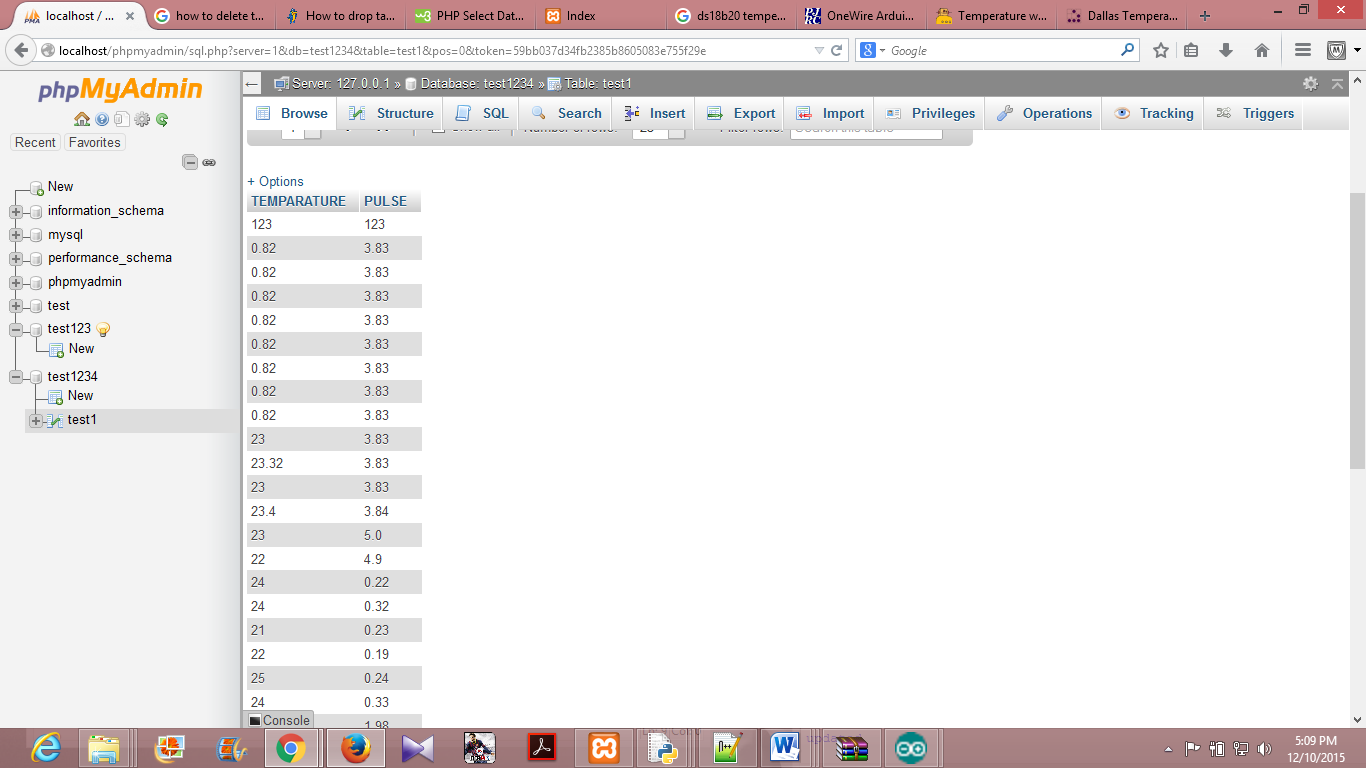
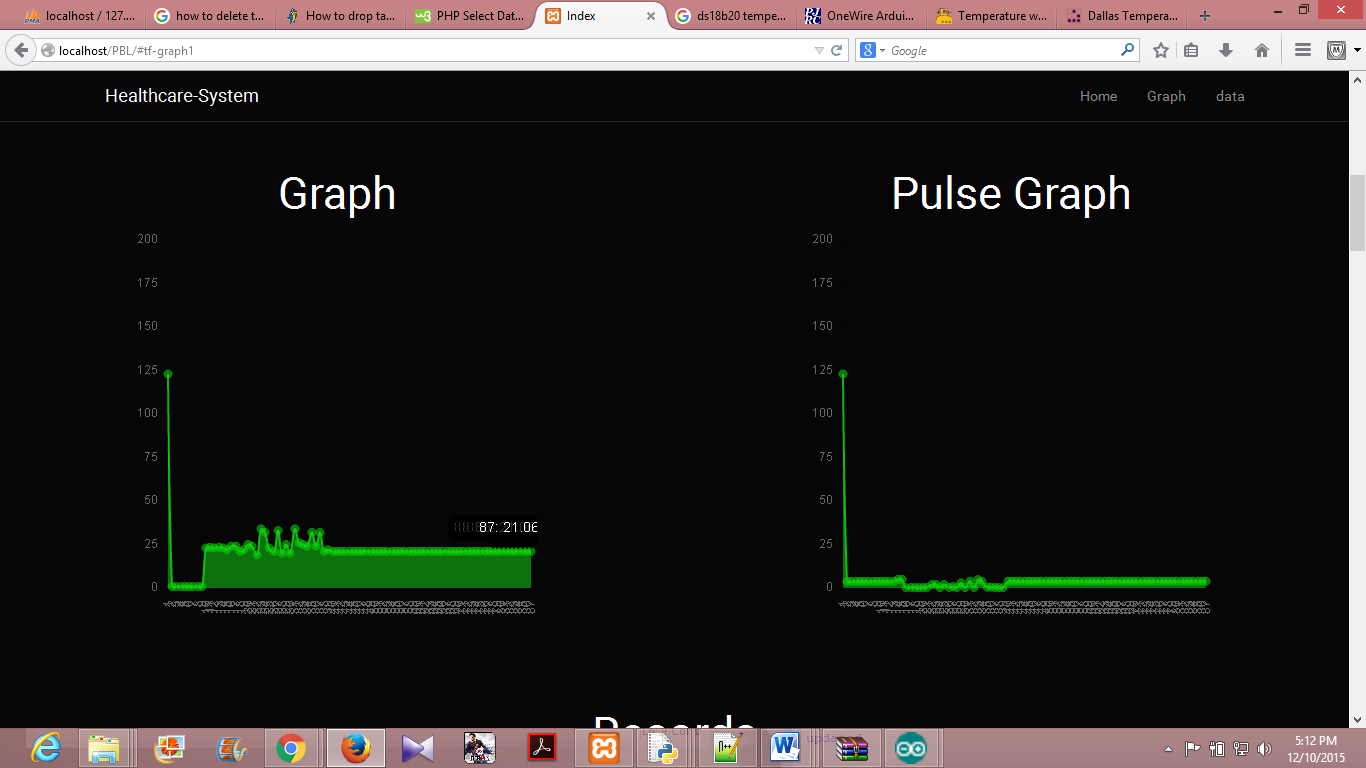
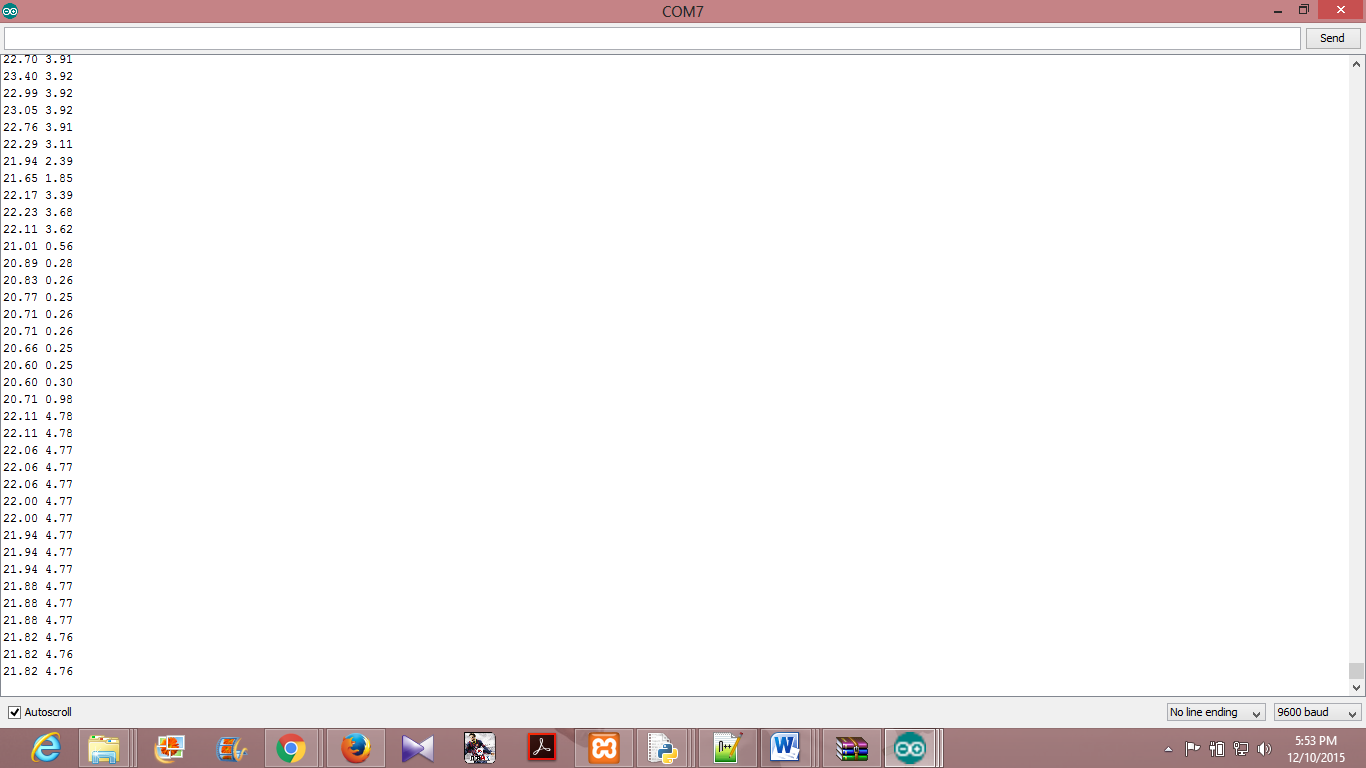
* Americans with Disabilities Act of 1990
* Section 508 Amendment to the Rehabilitation Act of 1973
* Web Accessibility Initiative (WAI) of the World Wide Web Consortium (W3C)

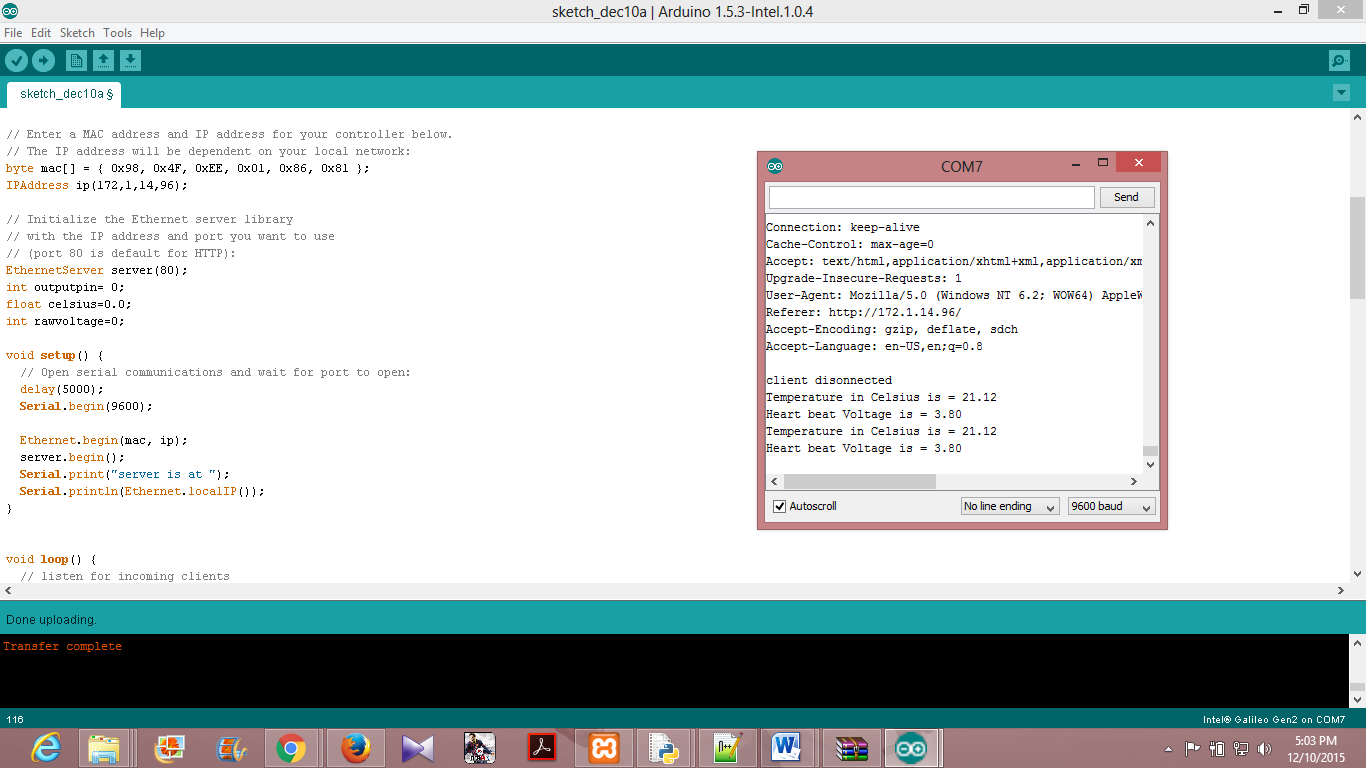
### Security testing

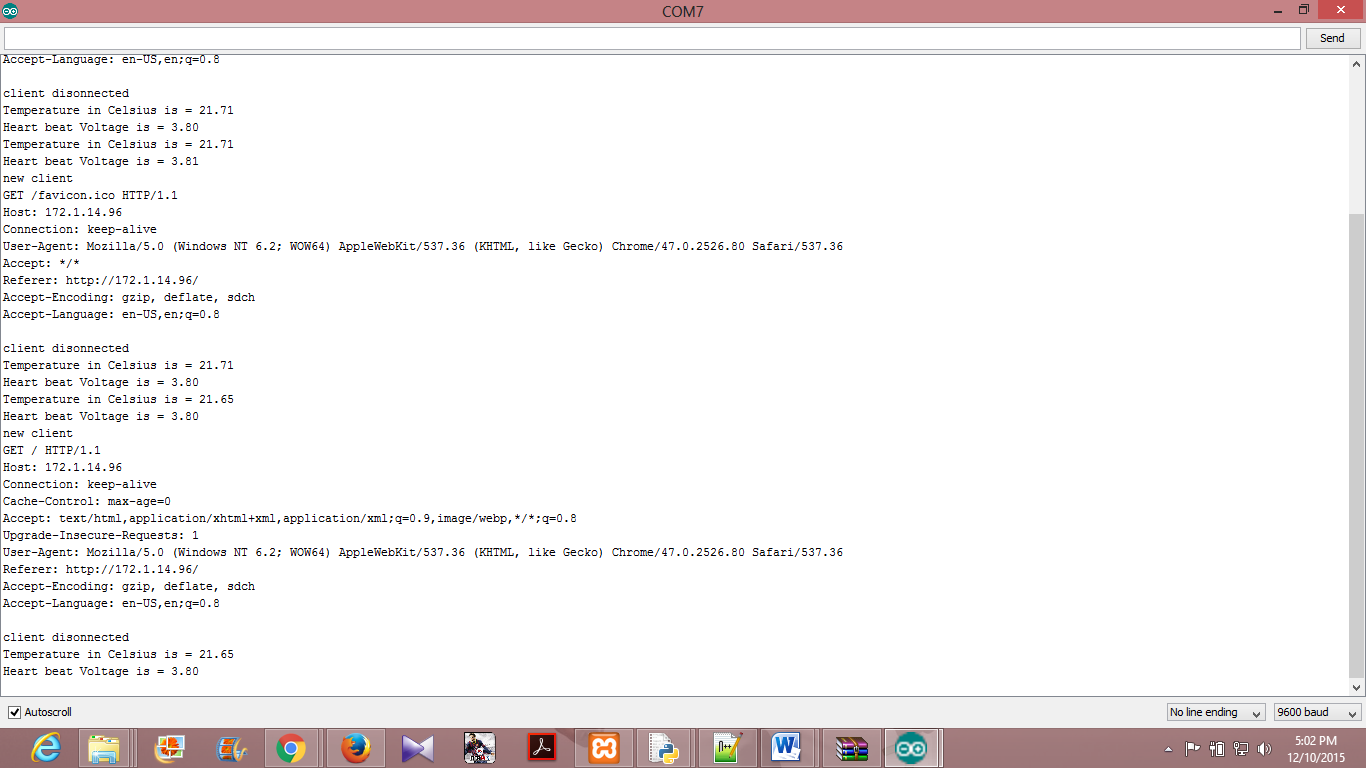
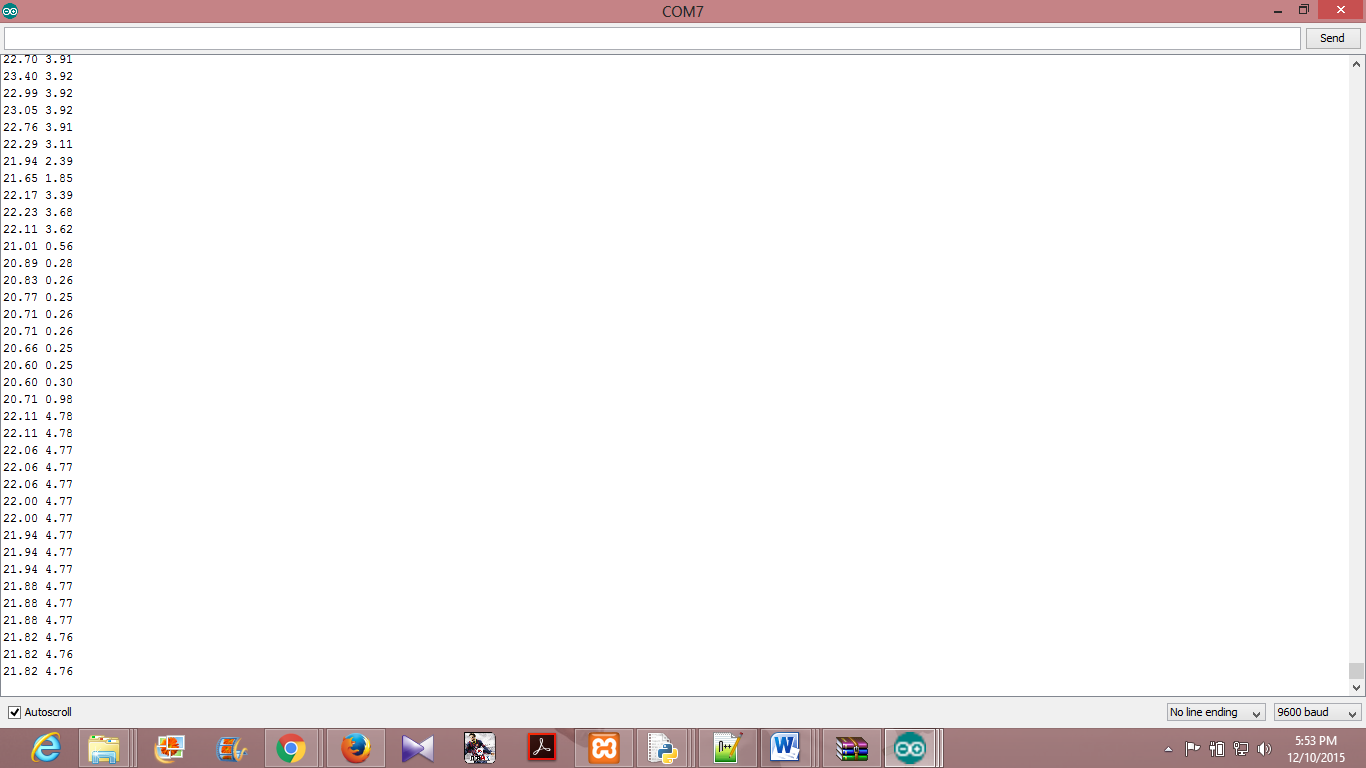
Security testing is essential for software that processes confidential data to prevent system intrusion by hackers.

The International Organization for Standardization (ISO) defines this as a "type of testing conducted to evaluate the degree to which a test item, and associated data and information, are protected to that unauthorized persons or systems cannot use, read or modify them, and authorized persons or systems are not denied access to them.

**SNAPSHOTS: **

****

****

****

**9. CONCLUSION AND SCOPE FOR FUTURE WORK**

The healthcare monitoring system is successful in its objective of providing an efficient and compelling way of health care of patients. The basic aim to reduce the manual effort put in such task and to maintain a separate catalogue for patients. This will help the doctors for recognizing any trends and making further decisions on medication and treatment for that patient. Such systems can help in reduction in also maintaining the reports for the patients.

**Future scope of health care system:**

Using this system as framework, the system can be expanded to include various other sensors which could include body movement sensors, pressure sensors etc..and storing it onto the cloud. This will further help in even more effective monitoring of patients. This kind of a system with respective changes can be implemented in the hospitals for disable people or in nursing homes or at old age homes will serve the people better.

**10. REFERENCES**

[1]Python Software[US],[online] Available: https://pypi.python.org/pypi

[2] Galileo Gen2 ,[online]Available:

https://www.arduino.cc/en/Guide/IntelGalileoGen2

[3]  Claire Rowland, Elizabeth Goodman, Martin Charlier, Ann Light & Alfred Lui “Designing Connected Products”,

[4] [online]Available: http://enterpriseiotbook.com

[5] [online]Available: https://software.intel.com/en-us/iot/library/galileo-getting-started

[6] Madni, A.M., "Smart Configurable Wireless Sensors and Actuators for System of Systems (SoS) Applications". Keynote Address, Proceedings, IEEE SMC Intl. Conf. on System of Sytems Engineering, Los Angeles, April 2006.

[7]. Von Lubitz, D. and N. Wickramasinghe, 2006 "Healthcare and

Technology: The Doctrine of Networkcentric Helahtcare," Intl. J Electronic Healthcare(IJEH), vol. 4, pp.322-344.

[8] Introducing Intel Galileo as a development platform of smart sensor

Evolution, opportunities and challenges,School of Engineering and Advanced