ARDUINO HEARTBEAT SENSOR

LAB REPORT

Submitted by

GANESH K [RA2111003010298]
HARIHARA KRISHNA[RA2111003010300]
SHREYA RANJAN[RA2111003010301]

Under the Guidance of

Ms. HEMA M
ASSISTANT
PROFESSOR
COMPUTING
TECHNOLOGIES
DEPARTMENT

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

ARDUINO HEARTBEAT SENSOR

LAB REPORT

Submitted by

GANESH K [RA2111003010298]
HARIHARA KRISHNA[RA2111003010300]
SHREYA RANJAN[RA2111003010301]

Under the Guidance of

Ms. HEMA M
ASSISTANT
PROFESSOR
COMPUTING
TECHNOLOGIES
DEPARTMENT

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
Chengalpattu District

BONAFIDE CERTIFICATE

RegisterNo.**RA2111003010298,RA2111003010300,RA2111003010301**

Certified to be the bonafide work done by **GANESH K, HARIHARA KRISHNA, SHREYA RANJAN** of II Year/IV Sem B.Tech Degree Course in the **Practical Software Engineering and Project Management 18CSC206J** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Kattankulathur during the academic year 2022 – 2023.

M. Hema (M. Hema)

LAB INCHARGE

M. HEMA
Assistant Professor
Department of Computing Technologies SRMIST – KTR

Department of Computing Technology
Kattankulathur
603 203
SRM IST

M. Pushpalatha

Head of the Department

Dr. Pushpalatha M
Professor & Head
Department of Computing Technologies SRMIST- KTR

Date : 2/5/23

ABSTRACT

A heartbeat sensor using an Arduino board is a simple yet effective way to measure heart rate. The sensor works by detecting changes in blood flow through the fingertip using a photoplethysmography (PPG) technique. The sensor consists of an infrared LED and a photodiode that are placed on either side of the fingertip. When the LED emits light, it penetrates the skin and reaches the blood vessels. The photodiode then detects the amount of light that is reflected back, which is proportional to the blood flow in the fingertip.

The Arduino board reads the analog signal generated by the photodiode and processes it using an algorithm to calculate the heart rate. The algorithm uses a Fast Fourier Transform (FFT) to analyze the signal and identify the frequency of the heartbeat. The calculated heart rate is then displayed on an LCD screen or sent to a computer or mobile device for further analysis.

Overall, a heartbeat sensor using an Arduino board is a cost-effective and easy-to-implement solution for monitoring heart rate in various applications, including fitness tracking, medical monitoring, and stress management

LIST OF FIGURES

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	1
Title of Experiment	To identify the Software Project, Create Business Case, Arrive at a Problem Statement
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K, Harihara Krishna Ramesh
Register Number	RA2111003010301
Date of Experiment	23-01-2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

6-2-2023

Staff Signature with date

Aim -

To Frame a project team, analyze and identify a Software project. To create a business case and arrive at a Problem Statement for the **HEARTBEAT SENSOR** Project.

Team Members -

S. No	Register No	Name	Role
1	RA2111003010298	GANESH K	Lead/Rep
2	RA2111003010301	SHREYA RANJAN	Member
3	RA2111003010300	HARIHARA KRISHNA RAMESH	Member

Project Title - HEARTBEAT SENSOR

Project Description -

Heart rate, body temperature and blood pressure monitoring are very important parameters of the human body. Doctors use various kinds of medical apparatus like thermometer for checking fever or body temperature, BP monitor for blood pressure measurement and heart rate monitor for heart rate measurement. In this project, we have built an Arduino based heartbeat monitor which counts the number of heartbeats in a minute. Here we have used a heartbeat sensor module which senses the heartbeat upon putting a finger on the sensor.

BUSINESS CASE

	23/01/2023
	Shreya Ranjan
	Member

THE PROJECT -

The principle behind the working of the Heartbeat Sensor is Photoplethysmography. According to this principle, the change in the volume of blood in an organ is measured by the changes in the intensity of the light passing through that organ.

IR LEDs are typically used as the light source in heartbeat sensors, while photodetectors such photodiodes, light-dependent resistors (LDRs), and phototransistors are typically used as the detectors. These two—a light source and a detector—can be set up in one of two configurations: a Transmissive Sensor or a Reflective Sensor.

THE HISTORY -

The heart rate monitor was in the realm of sci-fi not so very long ago. Researchers started checking heart rates around 1912, using water buckets as counterweights in the first laboratory model. The first electronic heart-monitoring tool, the electrocardiograph, was originally the size of a room, and even today we would certainly not want to carry one around (even if we could afford one). Looking forward to this problem, we have tried to design a real-time portable sensor using modern Arduino to analyze how the nonlinear signal recurrence plot of human heartbeat is done.

ESTIMATED COST -

Arduino Uno (1) Rs - 1,440

16 x 2 LCD Display (1) Rs - 270

10KΩ Potentiometer (1) Rs - 10

330Ω Resistor (Optional – for LCD backlight) Rs – 5

Push Button (1) Rs – 50

Heartbeat Sensor Module with Probe (finger based) (1) Rs-880

Mini Breadboard (1) Rs - 70

Connecting Wires Rs - 120

TOTAL Rs – 2,845 (approx.)

APPROACH -

Upload the code to Arduino UNO and Power on the system. The Arduino asks us to place our finger in the sensor and press the switch. Place any finger (except the Thumb) in the sensor clip and push the switch (button). Based on the data from the sensor, Arduino calculates the heart rate and displays the heartbeat in bpm. While the sensor is collecting the data, sit down and relax and do not shake the wire as it might result in a faulty value. After the result is displayed on the LCD, if you want to perform another test, just push the rest button on the Arduino and start the procedure once again.

LIMITATIONS -

The maximum number of programs an Arduino board can run concurrently is one. Multitasking is a feature of other rival boards like the Raspberry Pi. Because Arduino lacks the capacity to run multiple programs simultaneously without affecting system performance, we must close one sketch in order to start another, unlike multicore CPUs.

Most Arduino boards use microcontrollers, which aren't yet fully capable of performing at their best. To make creating sketches simple for new users, the Arduino development environment has been streamlined. All of this optimization reduces the microcontroller's overall power capacity. The performance will be much enhanced if the same microcontroller is utilized in AVR development.

BENEFITS -

Keeping track of heart rate can give an insight into the fitness level, heart health and emotional health. Many people are walking around with a resting heart rate that is too high, due to factors such as too much caffeine, dehydration, inactivity and persistent stress.

The heart rate of a human can be calculated with a straightforward setup that uses an Arduino UNO, 162 LCD, and heartbeat sensor module, which is a much easier and user-friendly setup that even our elders can easily operate at home without going to the hospital for the same thing. Moreover, Smart Watches and other pricey Heart Rate Monitors can also be replaced with this project as a very less expensive option.

RESULT -

Thus, the project team formed, the project is described, the business case was prepared and the prob'em statement was arrived.

M. Hema
6-2-2023

RESULT -

Thus, the project team formed, the project is described, the business case was prepared and the problem statement was arrived.

M. Hema
6-2-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	2
Title of Experiment	Identification of Process Methodology and Stakeholder Description
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna Ramesh (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	31/01/2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

13-2-2023

Staff Signature with date

Aim -

To identify the appropriate Process Model for the project and prepare Stakeholder and User Description.

Team Members -

Sl No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Project Title - HEARTBEAT SENSOR

Project Methodology -

The waterfall model was the methodology We used to design the heartbeat sensor system using a heart rate sensor. In order to construct the application, this model's four phases are used. This model is highly structured and well-organized. It will provide more information on each stage of the project's development as well as the system needs.

Staff Signature with date

Aim -

To identify the appropriate Process Model for the project and prepare Stakeholder and User Description.

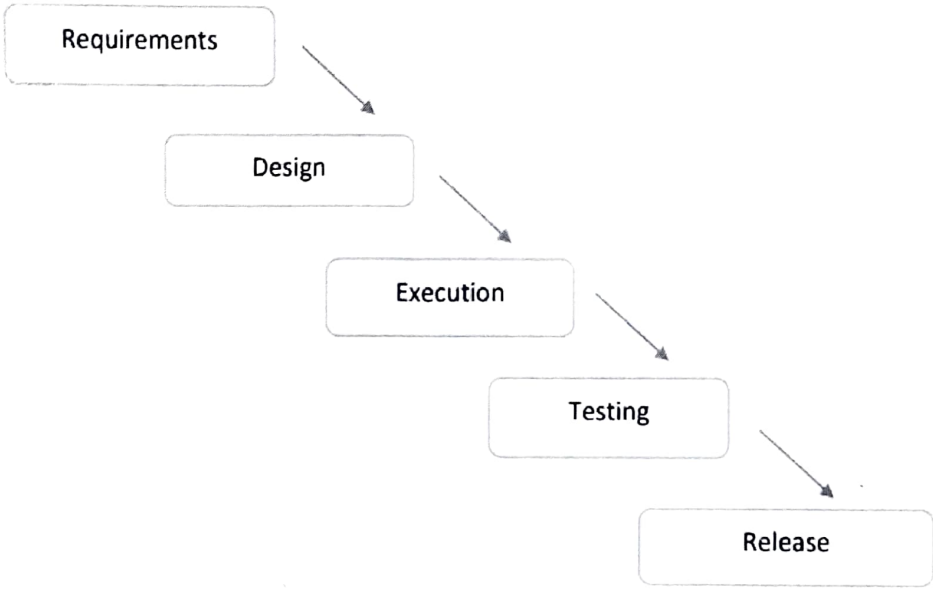
Team Members -

Sl No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Project Title - HEARTBEAT SENSOR

Project Methodology -

The waterfall model was the methodology We used to design the heartbeat sensor system using a heart rate sensor. In order to construct the application, this model's four phases are used. This model is highly structured and well-organized. It will provide more information on each stage of the project's development as well as the system needs.



The waterfall model is an activity project that splits each phase into a waterfall-like sequence through the phases of conception, initiation, analysis, design, construction, testing, production, or implementation, and maintenance. To prevent phase overlap, each phase must be finished before the following phase.

The sequential phases in Waterfall model are –

- Requirement Gathering and analysis – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- System Design – The requirement specifications from the first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- Implementation – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Waterfall Model".

Waterfall Model - Application -

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are –

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages -

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages -

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows –

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

Incorporate information to below table regarding stakeholders of the project -

Stakeholder Name	Activity/ Area /Phase	Interest	Influence	Priority (High(1)/ Medium(2)/ Low(3))
Project Leader	Lead the team in every aspect, Accountable for the entire project scope,team,success and failure.	High	High	1
Team members	Demand incentives,retain and upgrade skills,new product excitement and development.	High	High	2
Suppliers	Ensuring feasible and realistic in every aspect, managing divergence from budgeted cost.	Low	High	3
Deans and Department Head	Supports and endorses cooperation with university compliance and monitoring efforts related to the project.	High	Medium	2
Faculty	Guides for projects	High	High	2
End users	Use and provide feedback.	High	High	1
Sponsor	Aligning the project with strategy, and objectives.	Medium	Low	3
Alumni and development office	Creating opportunities for volunteerism and engagement to support long-term growth.	Medium	Low	3
IT department	Providing technical support to expertise, and ensuring the seamless integration	Medium	Medium	2

	of technology into the project plan			
Investors	Providing financial resources and support, evaluating potential risks and returns, and making decisions that can help for success of project	Low	Low	3
Resource manager	Overseeing and coordinating the allocation of human, ensuring efficient use of resources and resolve issues that may arise during the project	Medium	Low	3
Students	Bring fresh perspectives, enthusiasm, building valuable skills, and contributing to the overall success of the project.	Low	High	2

M. Sharma 13-2-2023

Result -

Thus the Project Methodology was identified and the stakeholders were described.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	3
Title of Experiment	System, Functional and Non-Functional Requirements of the Project
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna Ramesh (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	06.02.2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

15-2-2023

Staff Signature with date

Aim -

To identify the system, functional and non-functional requirements for the project.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Project Title - HEARTBEAT SENSOR

System Requirements -

- Arduino UNO x 1
- 16 x 2 LCD Display x 1
- 10KΩ Potentiometer
- 330Ω Resistor (Optional – for LCD backlight)
- Push Button
- Heartbeat Sensor Module with Probe (finger based)
- Mini Breadboard
- Connecting Wires

Functional Requirements -

1. Heart Rate Detection: The system should be able to accurately measure and display the heart rate of the user in beats per minute (BPM).
2. Heartbeat Visualization: The system should provide a visual representation of the heartbeat, such as a pulse graph or oscilloscope display.
3. User Input: The system should have a user interface that allows the user to input their age, weight, and height, which can be used to calculate the estimated maximum heart rate.
4. Heart Rate Range Monitoring: The system should monitor the heart rate range and alert the user if the heart rate goes above or below the normal range.
5. Alarm System: The system should have an alarm system that can be activated if the heart rate goes above or below the normal range.
6. Data Storage: The system should store the heart rate data and display it in a meaningful format, such as a chart or graph.
7. Compatibility: The system should be compatible with various sensors, such as pulse oximeters, ECG sensors, and photoplethysmogram (PPG) sensors, for maximum versatility.
8. Battery Life: The system should have a long battery life, allowing for extended use without the need for frequent recharging.
9. Connectivity: The system should have the ability to connect to other devices, such as smartphones, laptops, and tablets, for remote monitoring and data transfer.
10. User-Friendly Interface: The system should have a user-friendly interface that is easy to navigate and understand, with clear and concise instructions for use.

Non-Functional Requirements -

1. Accuracy: The system should accurately measure the heart rate and provide reliable results, with a margin of error of no more than ±5 BPM.
2. Responsiveness: The system should respond quickly to changes in heart rate and display the updated information in real-time.
3. Performance: The system should perform efficiently, with a fast processing speed and low latency, to provide smooth and uninterrupted data transmission.
4. Durability: The system should be durable and resistant to damage, with a rugged design that can withstand normal wear and tear.
5. Portability: The system should be compact and lightweight, allowing for easy transport and use in various settings.
6. Security: The system should be secure, with encrypted data transmission and storage to protect sensitive information.
7. Interoperability: The system should be interoperable with other devices and systems, allowing for seamless integration and data transfer.
8. Scalability: The system should be scalable, allowing for the addition of new sensors and features as needed.
9. Maintenance: The system should be easy to maintain and repair, with a user-friendly interface and clear documentation for troubleshooting and maintenance.
10. Compliance: The system should comply with relevant standards and regulations, such as those related to medical device safety and data privacy.

M. Flews 15-2-2023

Result -

Thus the requirements were identified and accordingly described.

SRM

INSTITUTE OF SCIENCE & TECHNOLOGY

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	4
Title of Experiment	Prepare Project Plan based on scope. Calculate Project effort based on resources and Job roles and responsibilities
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna Ramesh (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	20-02-2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

27-2-2023

Staff Signature with date

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	4
Title of Experiment	Prepare Project Plan based on scope, Calculate Project effort based on resources and Job roles and responsibilities
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna Ramesh (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	20-02-2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

M. Flex 27-2-2023

Staff Signature with date

Aim -

To Prepare Project Plan based on scope, Calculate Project effort based on resources, Find Job roles and responsibilities.

Team Members -

Sl No	Register No	Name	Role
1	RA2111003010298	GANESH K	Lead
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

1. Project Management Plan -

Focus Area	Details
Integration Management	Arduino board, Pulse sensor, OLED display, Power source, Circuitry to connect the components, Casing for the device. The heartbeat sensor project will use gradual integration as its integration technique. Before being integrated with other components, each component will first be integrated and evaluated independently. By using this method, any problems with the individual components will be found and fixed before they are integrated with other components. The components will be integrated using the next procedure: Step 1: Join the Arduino board with the pulse sensor. Step 2: Verify the pulse sensor is operating properly by testing it. Step 3: Join the Arduino board and OLED display. Step 4: Verify the OLED display's functionality by testing it. Step 5: Join the Arduino board to the power supply. Step 6: Verify the power source is operational by testing it.

Step 7: Connect the circuitry to join the parts together in accordance with the circuit design.
Step 8: Verify the circuitry's functionality by testing it.
Step 9: Within the case, attach the circuitry, the pulse sensor, the OLED display, and the power supply.
Step 10: Test the complete device to ensure it is working correctly.

The heart beat sensor project's components will be integrated using Arduino according to the integration management strategy. By using an incremental approach to integration, any problems with individual components will be found and fixed before integration with other components. Every stage of the integration process will include testing and validation to make sure the overall system is operating properly. The components will be integrated in accordance with the circuit design using configuration management, and all project-related paperwork will be kept in one location.

Scope Management

Scope management for a heart beat sensor system using Arduino involves defining the project's objectives, requirements, and deliverables, and then developing a plan to achieve them within the available resources, time, and budget.

step 1: Define the project's scope by figuring out its goals, the results that can be expected, and any limitations like time and money.
step 2: Identify project requirements by Determining the functional and non-functional requirements of the system, such as the sensitivity of the heart rate monitor, the accuracy of measurements, and the response time of the system.
step 3: Create a hierarchical framework that outlines the tasks, subtasks, and activities needed to finish the project and break it down into smaller, manageable pieces.
step 4: The heart rate monitor device, the software interface for displaying results, and any necessary accompanying documentation are a few examples of the specific deliverables that must be developed as part of the project.
step 5: Determine the people, supplies, and machinery needed to finish the job.
step 6: Create a project budget by estimating the total cost of the project, taking into account labor costs, material costs, and other costs.

step 7: The project's progress should be continuously monitored, and any required adjustments should be made to keep it on course and within the project's scope.

Quality Management

Quality management for a heart beat sensor system using Arduino involves ensuring that the system meets or exceeds the expected quality standards.

Develop quality requirements: Define the quality requirements for the system, such as accuracy, sensitivity, and response time, and ensure that these requirements are clearly communicated to all stakeholders.

Perform thorough testing: Perform comprehensive testing of the system components and software tools to ensure that they meet the quality requirements.

Conduct validation and verification: Conduct validation and verification tests to ensure that the system meets the desired performance and functionality standards.

Implement quality assurance processes: Implement quality assurance processes, such as peer reviews, to identify and address potential issues early on in the development process.

Use reliable components: Use reliable components for the system, such as sensors and microcontrollers, to minimize the risk of system failures.

Stay up-to-date with industry standards: Stay up-to-date with industry standards and best practices for heart rate monitoring systems to ensure that the system meets or exceeds these standards.

Risk Management

Risk management for a heart beat sensor system using Arduino involves identifying potential risks that could impact the project's success and developing strategies to mitigate those risks.

1) Technical risks: These risks include hardware or software failure, inadequate sensor sensitivity or accuracy, and compatibility issues with other systems. To mitigate these risks, you can perform thorough testing of the system components

and conduct simulations or trials to identify potential technical issues early on.

2) Schedule risks: These risks include delays in component delivery, unexpected changes in project requirements, and unforeseen complications during system integration. To mitigate these risks, you can develop contingency plans and build in extra time in the project schedule to account for potential delays or changes.

3) Resource risks: These risks include a shortage of personnel or equipment, unexpected costs, and changes in project priorities. To mitigate these risks, you can allocate resources effectively, closely monitor the project budget, and maintain open communication with stakeholders to ensure that any changes in priorities are addressed in a timely manner.

4) Security risks: These risks include unauthorized access to the heart rate data or the system itself, data loss, and breaches of privacy. To mitigate these risks, you can implement appropriate security protocols, such as encryption and access controls, and conduct regular security audits to identify and address potential vulnerabilities.

5) Regulatory risks: These risks include non-compliance with relevant regulations or standards, such as HIPAA or FDA guidelines for medical devices. To mitigate these risks, you can stay up-to-date on the latest regulations and standards, seek expert advice if necessary, and conduct thorough testing and validation of the system to ensure compliance.

2. Estimation -

2.1.Effort and Cost Estimation -

Activity Description	Sub-Task	Sub-Task Description	Effort (in hours)	Cost in INR
Project Management	Initiation,Planning, Specifications, Stakeholder Analysis, Execution, Monitoring and Control, Closure	application of processes, methods, skills, knowledge and experience to achieve specific project objectives according to the project acceptance criteria within agreed parameters.	2-4	₹0
Hardware	Electronic design	In hardware is an activity that involves designing and developing the software that will control or interface with a hardware device.	7-8	₹0
	Mechanical design	Designing and creating a hardware device's physical structure and parts entails choosing the product's materials, dimensions, and shape as well as any necessary manufacturing procedures.		
Embedded Software	Learning to program Arduino	To ensure that the embedded software reliably measures the heartbeat rate and produces trustworthy output, it must be carefully built, tested, and debugged.	2-4	₹0
	Configuring the sensors			
	Signal Processing			
Wireless Software	Send and receive data	Providing the bridge between wired and wireless systems and connections.	5-6	₹3000
	Give received data to user interface software			
Design the user screen or interface	Graph from received data	Confirm the user requirements (acceptance criteria)	3	₹1500

	Beats per minute reading	Giving digital output when the finger is placed on it.	1-2	₹1000
	Graphical design	Focusing on the logic of displaying elements in interactive designs, to optimize the user experience.	1-2	₹1000
Identify Data Source for displaying units of Energy Consumption	Relational, Multidimensio -nal, Dimensionally Modeled Relational	Go through Interface contract (Application Data Exchange) documents.	5	₹2500
	Documents	Documents and User Manuals.	2-4	₹0
Dissemination	Project plan	Making the results and deliverables of a project available to the stakeholders and to the wider audience.	9-10	₹0
	Final report			
	Final gala presentation			

Effort (hr)	Cost (INR)
1	500

2.2.Infrastructure/Resource Cost [CapEx] -

Infrastructure Requirement	Qty	Cost per qty	Cost per item
Arduino Uno Microcontroller	1	₹2300	₹2300
Heart Beat Sensor	1	₹300	₹300
OLED Display	1	₹175	₹175
Breadboard	1	₹135	₹135
Jumper Wires (Male-male wires)	20	₹0.5	₹10
₹Jumper Wires (Male-female wires)	20	₹0.5	₹10
Power Supply	1	₹300	₹300
Enclosure	1	₹400	₹400

2.3 Maintenance and Support Cost [OpEx] -

Category	Details	Qty	Cost per qty per annum	Cost per item
Operating Expenses	Incoming Arduino Uno microcontroller versions, Advanced Sensors, Latest Display etc.	1	₹2300	₹2300
	Latest Jumper Wires.	20	₹0.5	₹10
Planned Capital Expenditures	Purchasing and maintaining strategies, Replacing Long-Terms Assets.	1	₹3000	₹3000
Plan maintenance Cost	Routine Equipment Checking .	-	₹150	₹150
License	Software Database Middleware IDE (User Interface).	-	₹1000	₹1000
Expenditure of Spares Repair	Indicators, Inventory Reduction, Inventory Review.	-	₹800	₹800
Additional Buffer	Scheduling etc.	-	₹600	₹600

3. Project Team Formation -

3.1.Identification Team members -

Name	Role	Responsibilities
Ganesh K	Key Business User (Product Owner)	Provide clear business and user requirements
Ganesh K	Project Manager	Manage the project
Harihara Krishna	Business Analyst	Discuss and Document Requirements
Ganesh K	Technical Lead	Design the end-to-end architecture
Shreya Ranjan	Frontend Developer	Develop user interface
Ganesh K , Shreya Ranjan	Backend Developer	Storage of all data in the software.
Harihara Krishna	Tester	Define Test Cases and Perform Testing

3.2. Responsibility Assignment Matrix -

RACI Matrix / Activity	Team Members			
	Harihara Krishna (BA)	Shreya Ranjan (Developer)	Ganesh K (Project Manager)	Ganesh K (Key Business User)
Stakeholder Analysis	R	R	R/A	C
Future work Presentation	I	A	C	I
Information recollecting tool Analysis	R	R	R	R/I
Team Meeting	I	I	A	C
Questionnaire Application	A	A/C	R	I
System Engineering	I	R/A	C	I
Software Development	I/C	R	A	I
Hardware Development	I/C	R	R/A	I
Test Engineering	R/A	I/C	A	I/C
Result Analysis	R/A	C	A	R/A
Quality Assurance	R	R	R	I
Configuration Management	A	I/C	C	I
Integrated Logistic Support	I	R	C	I
Training	R	C	I	A
Fill Business Model Row	R/A	I/C	R/I	R/C
Result Presentation	C	R/C	A	R/A
Evaluation/ Presentation	R	R	A/I	A
User Requirement Documentation	A	C/I	I	R
Learning /Reflection	R	R	A	I

A	Accountable
R	Responsible
C	Consult
I	Inform

Result -

Thus, the Project Plan was documented successfully.

M. Ilavarasi
27-2-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	5
Title of Experiment	Prepare Work breakdown structure, Timeline chart, Risk identification table
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna Ramesh (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	27-02-2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

M-Shree 6-3-2023

Staff Signature with date

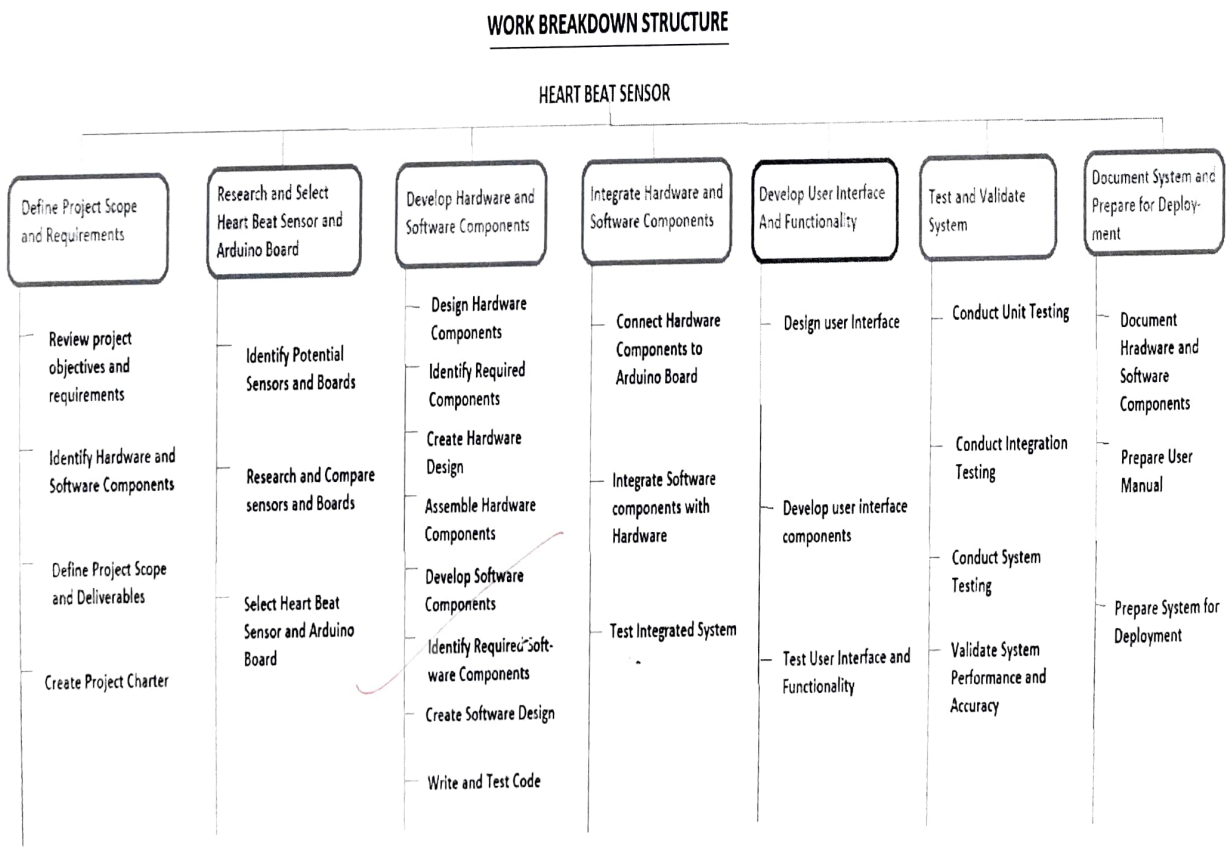
Aim -

To Prepare Work breakdown structure, Timeline chart and Risk identification table

Team Members -

Sl No	Register No	Name	Role
1	GANESH K	RA2111003010298	Rep/Leader
2	HARIHARA KRISHNA RAMESH	RA2111003010300	Member
3	SHREYA RANJAN	RA2111003010301	Member

WBS (Work Breakdown Structure) -



1. **Define project scope and objectives -**
    1.1 Determine the purpose and objectives of the project
    1.2 Identify the stakeholders
    1.3 Define the scope and limitations of the project

2. **Research heartbeat sensors and Arduinos -**
    2.1 Identify the types of heart beat sensors available
    2.2 Research the features and specifications of each sensor
    2.3 Determine the compatibility of each sensor with the Arduino

3. **Select appropriate components -**
    3.1 Select the heart beat sensor based on research
    3.2 Identify other necessary components (Arduino, breadboard, wires, resistors, etc.)
    3.3 Determine the quantity and specifications of each component

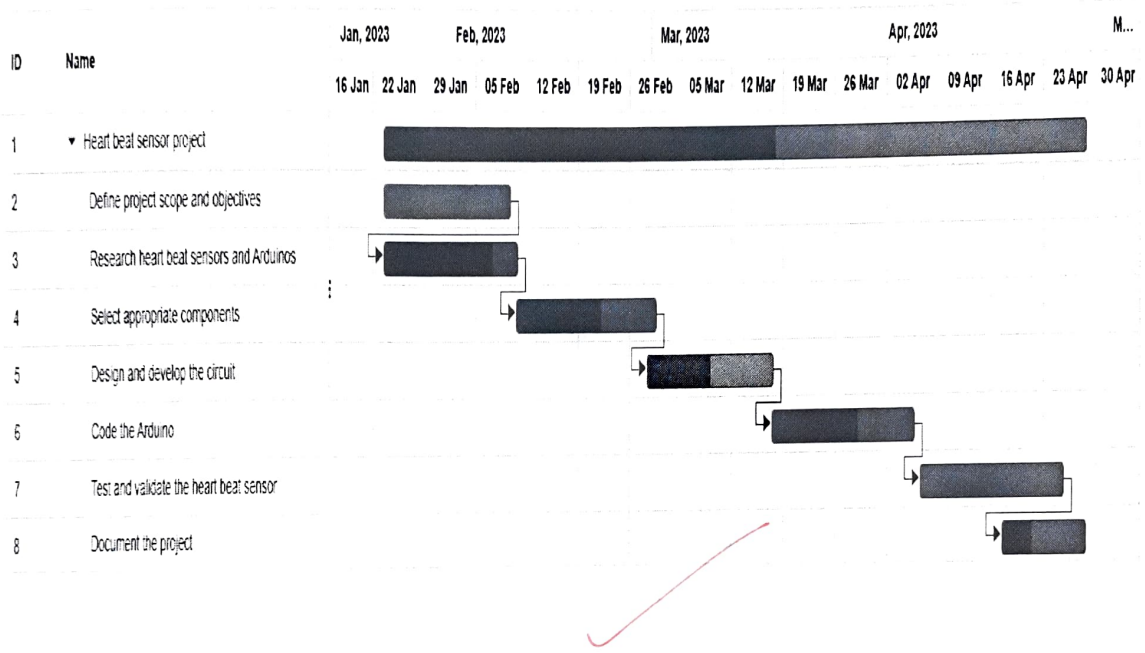
4. **Design and develop the circuit -**
    4.1 Create a schematic of the circuit
    4.2 Layout the components on a breadboard
    4.3 Connect the components according to the schematic
    4.4 Verify the circuit works as intended

5. **Code the Arduino -**
    5.1 Write the Arduino code to read the heart beat sensor
    5.2 Configure the Arduino to communicate with a computer
    5.3 Test the code to ensure it works as intended

6 **Test and validate the heart beat sensor -**
    6.1Test the heart beat sensor in different scenarios
    6.2 Validate the accuracy and consistency of the sensor
    6.3 Refine the circuit and code if necessary

7 **Document the project -**
    7.1 Create documentation for the project, including the circuit schematic, Arduino code, and testing results
    7.2 Write a user manual to guide users in operating the heart beat sensor
    7.3 Prepare a final report summarizing the project and its outcomes

TIMELINE – GANTT CHART -

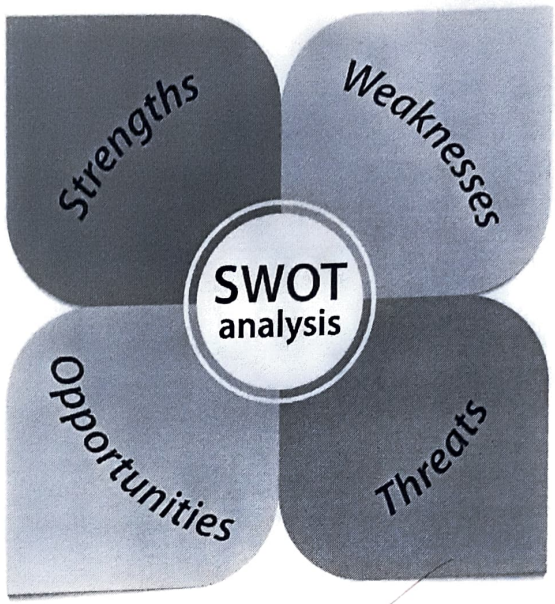


Name	Start Date	End Date	Time Taken	Progress%	Relaincy	Resources (Members)
Heart beat sensor project	Jan 23, 2023	Apr 28, 2023	70 days	56		Ganesh K, Harihara Krishna, Shreya Ranjan
Define project scope and objectives	Jan 23, 2023	Feb 09, 2023	14 days	100		Ganesh K
Research heartbeat sensors and Arduinos	Jan 23, 2023	Feb 10, 2023	15 days	80	2FS-14 days	Shreya Ranjan
Select appropriate components	Feb 10, 2023	Mar 01, 2023	14 days	60	3FS-1 days	Harihara Krishna
Design and develop the circuit	Feb 28, 2023	Mar 17, 2023	14 days	50	4FS-2 days	Shreya Ranjan, Ganesh K, Harihara Krishna
Code the Arduino	Mar 17, 2023	Apr 05, 2023	14 days	60	5FS-1 days	Ganesh K
Test and validate the heart beat sensor	Apr 06, 2023	Apr 25, 2023	14 days	0	6FS	Harihara Krishna
Document the project	Apr 17, 2023	Apr 28, 2023	10 days	35	7FS-7 days	Shreya Ranjan

Name	Start Date	End Date	Time Taken	Progress%	Relaincy	Resources (Members)
Heart beat sensor project	Jan 23, 2023	Apr 28, 2023	70 days	56		Ganesh K, Harihara Krishna, Shreya Ranjan
Define project scope and objectives	Jan 23, 2023	Feb 09, 2023	14 days	100		Ganesh K
Research heartbeat sensors and Arduinos	Jan 23, 2023	Feb 10, 2023	15 days	80	2FS-14 days	Shreya Ranjan
Select appropriate components	Feb 10, 2023	Mar 01, 2023	14 days	60	3FS-1 days	Harihara Krishna
Design and develop the circuit	Feb 28, 2023	Mar 17, 2023	14 days	50	4FS-2 days	Shreya Ranjan, Ganesh K, Harihara Krishna
Code the Arduino	Mar 17, 2023	Apr 05, 2023	14 days	60	5FS-1 days	Ganesh K
Test and validate the heart beat sensor	Apr 06, 2023	Apr 25, 2023	14 days	0	6FS	Harihara Krishna
Document the project	Apr 17, 2023	Apr 28, 2023	10 days	35	7FS-7 days	Shreya Ranjan

RISK ANALYSIS -

SWOT -



Strengths -

1. High accuracy and reliability of heart beat readings
2. Low cost of components and materials
3. Open source platform allows for easy customization and modification
4. Potential for use in various applications such as medical monitoring, fitness tracking,and sports performance analysis
5. Opportunity for learning and development of technical and programming skills

Weaknesses -

1. Requires a basic understanding of electronics and programming
2. Limited functionality compared to commercial heart rate monitors
3. May require calibration and adjustment to ensure accuracy
4. Limited market potential for commercialization
5. May not be suitable for all users, such as those with pacemakers or other medical conditions

Opportunities -

1. Growing demand for wearable health monitoring devices
2. Potential for partnership or collaboration with medical institutions, fitness centers, or sports teams
3. Opportunity to expand the functionality of the heart beat sensor through software updates and modifications
4. Potential for integration with other devices and technologies
5. Possibility of attracting investors or funding for further research and development

Threats -

1. Increasing competition in the health monitoring device market
2. Rapidly changing technology and potential obsolescence of current hardware and software
3. Potential for regulatory or legal issues related to medical device regulations and standards
4. Difficulties in marketing and distribution without proper resources and networks
5. Risks associated with data privacy and security

RMMM (Risk Mitigation Monitoring and Management) -

Risk	Mitigation	Monitoring	Management
Inadequate sensor accuracy	Test sensor accuracy with multiple users and devices. Implement calibration methods to improve accuracy.	Regularly review and compare sensor readings with other devices. Continuously calibrate the sensor to ensure accuracy.	Assign a dedicated team member to monitor and manage sensor accuracy. Set up automated alerts for inaccuracies.
Unforeseen technical difficulties	Develop contingency plans for technical issues. Have a backup plan in place in case of technical difficulties.	Regularly test hardware and software components. Monitor and document technical issues.	Assign a dedicated team member to manage technical issues. Implement regular team check-ins to discuss technical progress and issues.
Project delays	Monitor project progress regularly. Use agile methodology for iterative development.	Regularly update project timeline and milestones. Track progress against key	Assign a dedicated project manager to manage project timelines and milestones. Implement

	Implement contingency plans.	performance indicators (KPIs).	a change management process for unexpected delays.
Insufficient team resources	Increase team resources, delegate tasks effectively.	Regularly assess team member workload and availability. Document team member roles and responsibilities.	Assign a dedicated team member to manage team resources. Implement cross-training and knowledge-sharing activities.
Lack of programming and electronics knowledge	Ensure team members have the necessary skills or provide training. Seek external support if necessary.	Regularly assess team member knowledge and skills. Encourage continuous learning and development.	Assign a dedicated team member to manage team skills and knowledge. Implement a knowledge-sharing and training program.
Inability to secure funding	Develop a comprehensive funding plan with multiple options. Seek	Regularly review and update funding options. Document funding sources and requirements.	Assign a dedicated team member to manage funding. Implement a

	out potential investors and grants.		fundraising and grant application process.
Data privacy and security issues	Ensure data is stored securely and follow best practices for data privacy. Regularly monitor and update security measures.	Regularly assess data privacy and security risks. Implement regular security updates and patches.	Assign a dedicated team member to manage data privacy and security. Implement regular data privacy and security training for team members.

Result -

Thus, the work breakdown structure with timeline chart and risk table were formulated successfully.

6-3-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	6
Title of Experiment	Design a System Architecture, Use Case and Class Diagram
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	01-03-2023

Mark Split Up

S.No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

M. Ganesh 6-3-2023

Staff Signature with date

Aim -

To Design a System Architecture, Use case and Class Diagram

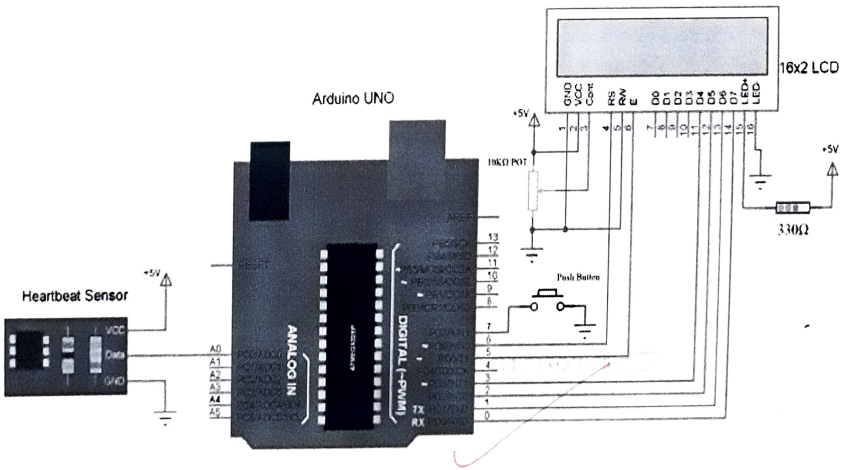
Team Members -

Sl No	Register No	Name	Role
1	GANESH K	RA2111003010298	Rep
2	HARIHARA KRISHNA RAMESH	RA2111003010300	Member
3	SHREYA RANJAN	RA2111003010301	Member

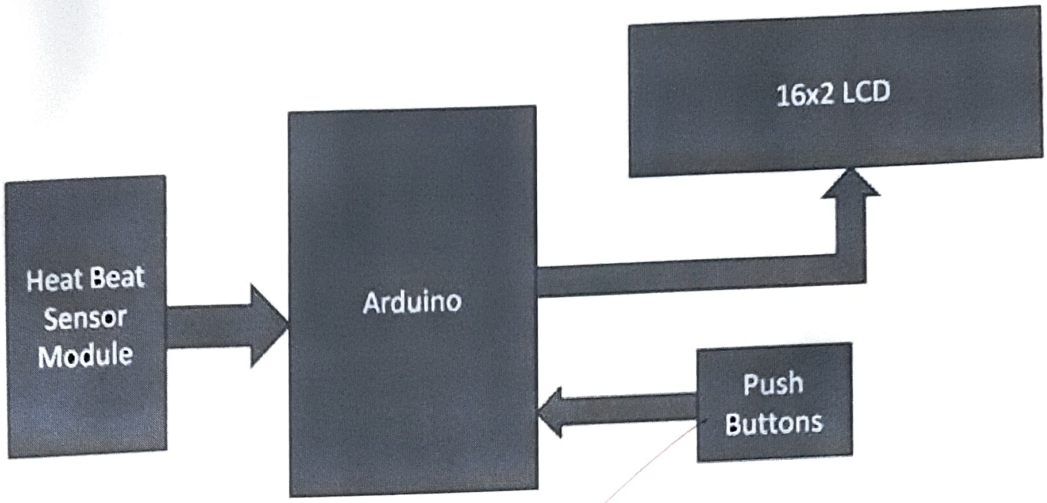
SYSTEM ARCHITECTURE -

a. Structural - wise -

Circuit Diagram

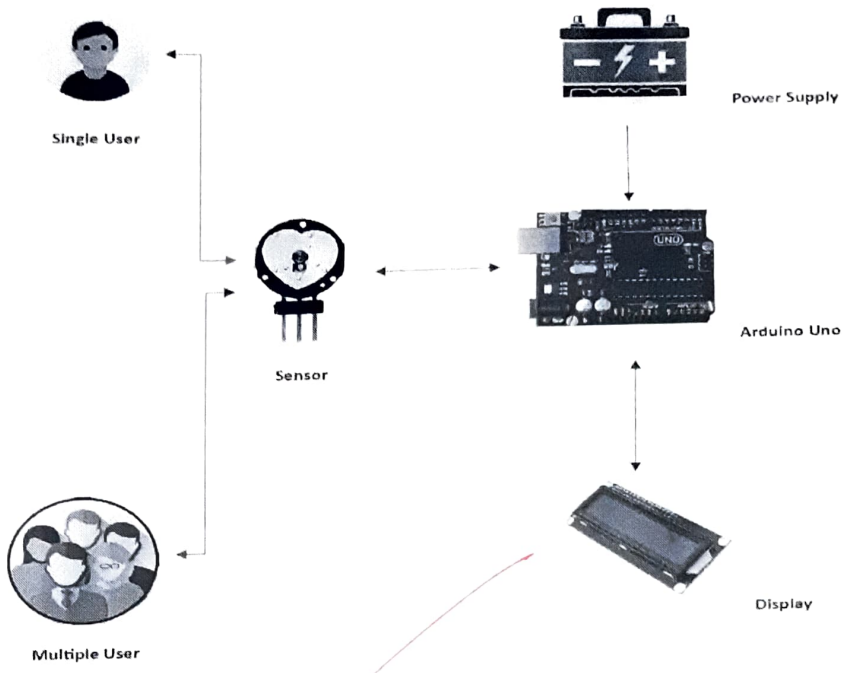


Block Diagram



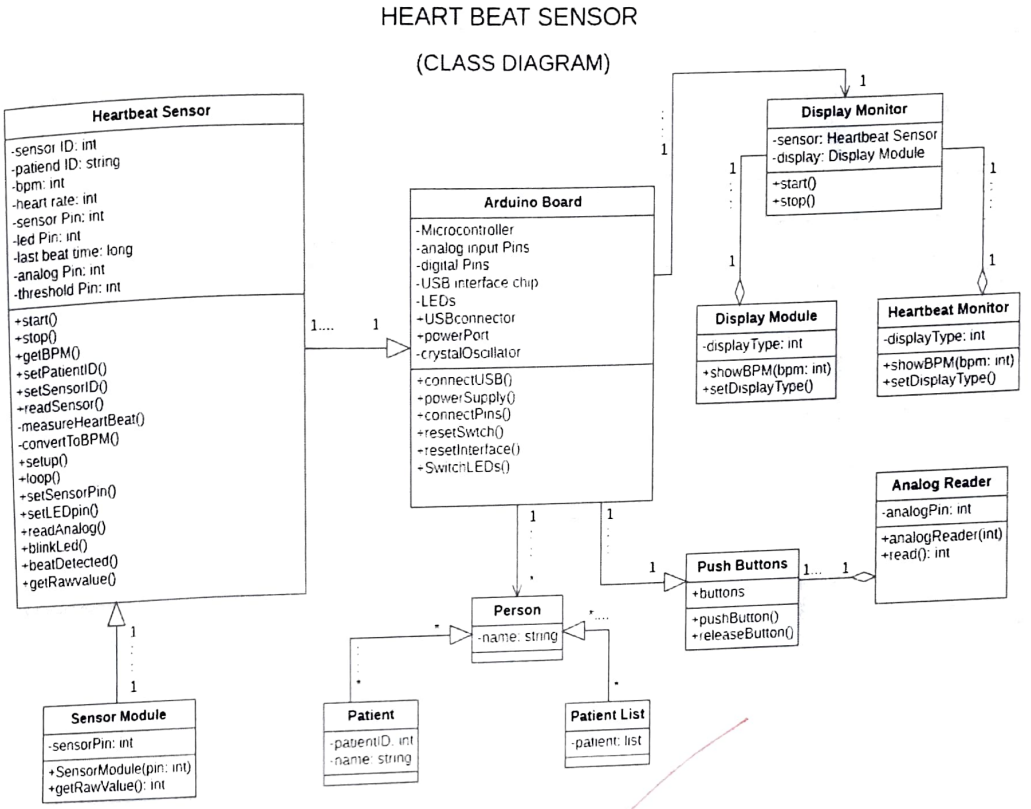
The circuit diagram for the Arduino-based heart rate monitor utilizing a heartbeat sensor is displayed in the arrangement diagram. The sensor contains a clip for attaching a finger and three pins for connecting the VCC, GND, and Data lines. The output pin of the heartbeat sensor module is directly linked to Arduino pin 8. There is a connection between Vcc and GND. The Arduino is connected to a 16x2 LCD in 4-bit mode. The Arduino pins 12, GND, and 11 are directly connected to the control pins RS, RW, and En. Moreover, the arduino's pins 5, 4, 3, and 2 are linked to data pins D4 through D7. Moreover, two push buttons have been added: one to activate the system for reading pulses and one for resetting the reading.Pressing the start button allows the Arduino to start counting pulses and establishing a five-second timer when we need to count heart rate. With respect to ground, this start push button is linked to pin 7 and the reset push button to pin 6 of the Arduino.

b. Behavioral - wise -



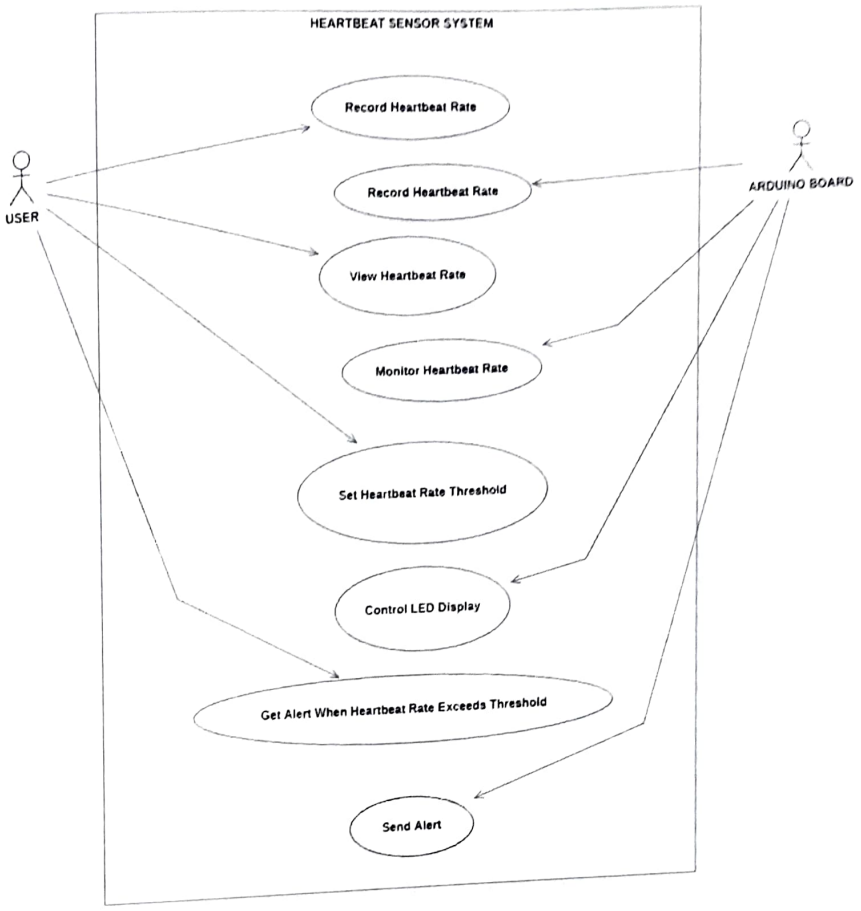
This system architecture of the Arduino UNO heartbeat sensor shows how the architecture behaves with respect to the input provided. Upload the code to Arduino UNO and Power on the system. The Arduino asks us to place our finger in the sensor and press the switch. Place any finger (except the Thumb) in the sensor clip and push the switch (button). Based on the data from the sensor, Arduino calculates the heart rate and displays the heartbeat in bpm. While the sensor is collecting the data, sit down and relax and do not shake the wire as it might result in faulty values. After the result is displayed on the LCD, if we want to perform another test, just push the reset button on the Arduino and start the procedure once again.

CLASS DIAGRAM -



This diagram shows different classes involved which are working coherently to make the Arduino Heartbeat Sensor work. In this, a heartbeat sensor class represents the sensor module which detects the fingerprint and sends the input to the internal architecture. A number of attributes and methods are displayed inside the class. Besides, an Arduino Board class is present which acts as the core of this diagram. Other classes include Display monitor to show the output Heart rate in BPM. Push buttons are present to enhance the interaction between users and the system. It also keeps a track of previous users with their respective names and heart rates. Relationships among various classes and multiplicities are shown to indicate how these classes are dependent on each other and are working cohesively for the optimal performance of the Arduino Heartbeat Sensor.

USE CASE DIAGRAM –



In this diagram, we can see the interactions between the user and the heartbeat sensor system, as well as the different components of the Arduino system that make up the functionality of the system. The user can interact with the system by recording the heartbeat rate, viewing the heartbeat rate, setting the heartbeat rate threshold, and getting an alert when the heartbeat rate exceeds the threshold. These interactions are facilitated by the Arduino system. The Arduino system is composed of several components, including the ability to record the heartbeat rate, monitor the heartbeat rate, control the LED display, and send an alert when the threshold is exceeded. These components work together to provide the necessary functionality for the heartbeat sensor system .

M. Ilavarasi 6-3-2023

Result -

Thus, the system architecture, use case and class diagram created successfully.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	7
Title of Experiment	Design a Entity relationship diagram
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	06-03-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

M. Hema 14-3-2023

Staff Signature with date

Aim -

To create the Entity Relationship Diagram

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

ENTITY RELATIONSHIP DIAGRAM AND NOTATIONS -

What is an ER Diagram ?

- Entity Relationship Diagrams (ER Diagrams), often called Entity Relationship Diagrams (ERDs), are diagrams that show the relationships between entity sets that are maintained in databases. In other words, ER diagrams aid in illuminating the logical organization of databases. Entities, attributes, and relationships serve as the foundation for ER diagram creation.

- ER Diagrams use a variety of symbols, including rectangles to represent entities, ovals to describe attributes, and diamond shapes to illustrate relationships.

- A flowchart and an ER diagram first have a striking similarity. But, the particular symbols and their meanings in the ER Diagram make this paradigm distinct. To represent the entity framework infrastructure, there is an ER Diagram.

What is the ER Model ?

Entity Relationship Model (ER Model), as its name suggests, is a high-level conceptual data model diagram. An effective database can be created by methodically analyzing the data requirements.

- The ER Model depicts real-world things and their connections. A great approach before building your database is to create an ER Model in a DBMS.

- ER Modeling aids in the methodical analysis of data requirements for the creation of a well-designed database. Therefore, it is recommended that you finish your ER modeling before putting your database into operation.

Why use ER Diagrams ?

Following are the main justifications for utilizing the ER Diagram:

- Aids in the definition of entity relationship modeling terminology.
- Give a glimpse of how your tables should connect to one another and what fields each table will include.
- Aids in the description of entities, properties, and relationships.
- ER diagrams may be converted into relational tables, making it simple to create databases.
- Database designers can utilize ER diagrams as a guide for integrating data into certain software applications.
- With the aid of an ERP diagram, the database designer is better able to comprehend the data that will be stored in the database.
- ERD Diagram enables you to explain to users the logical organization of the database.

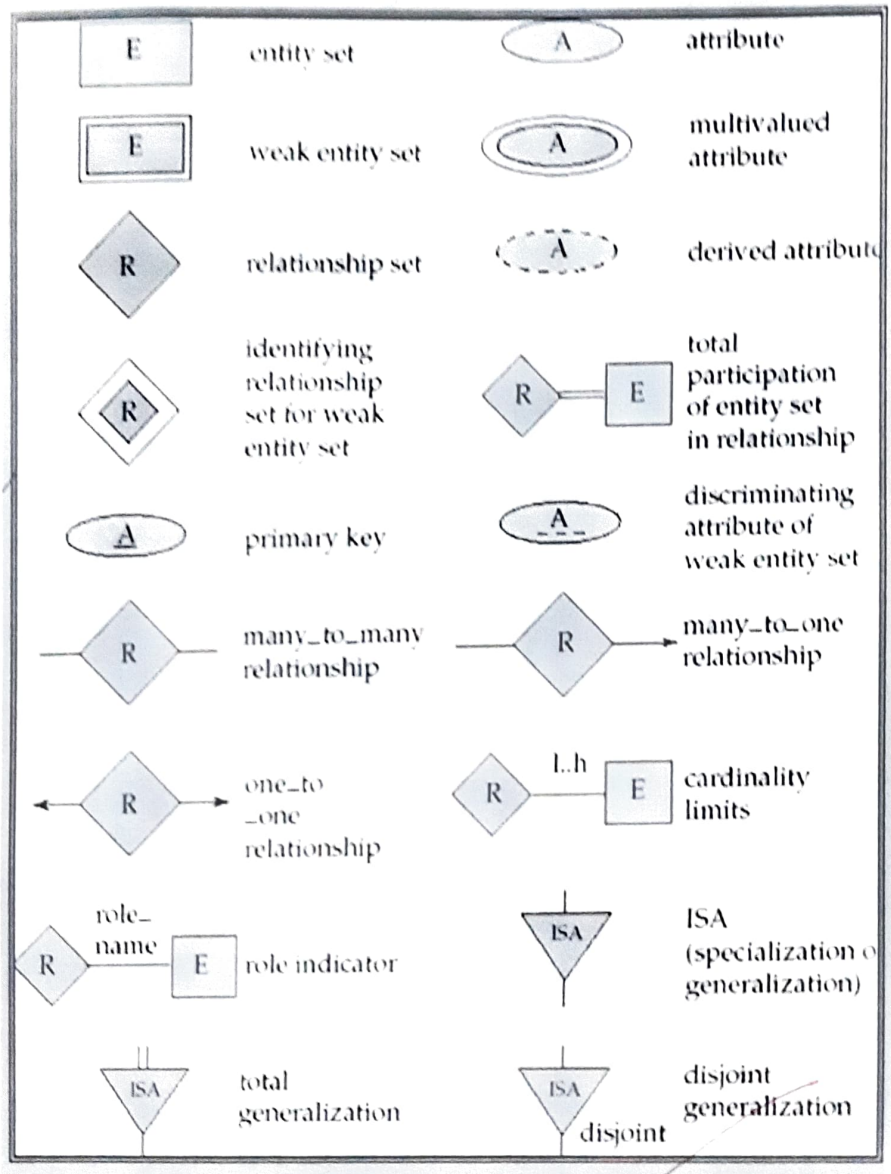
Components of the ER Diagram -

This model is based on three basic concepts: Entities, Attributes, Relationships

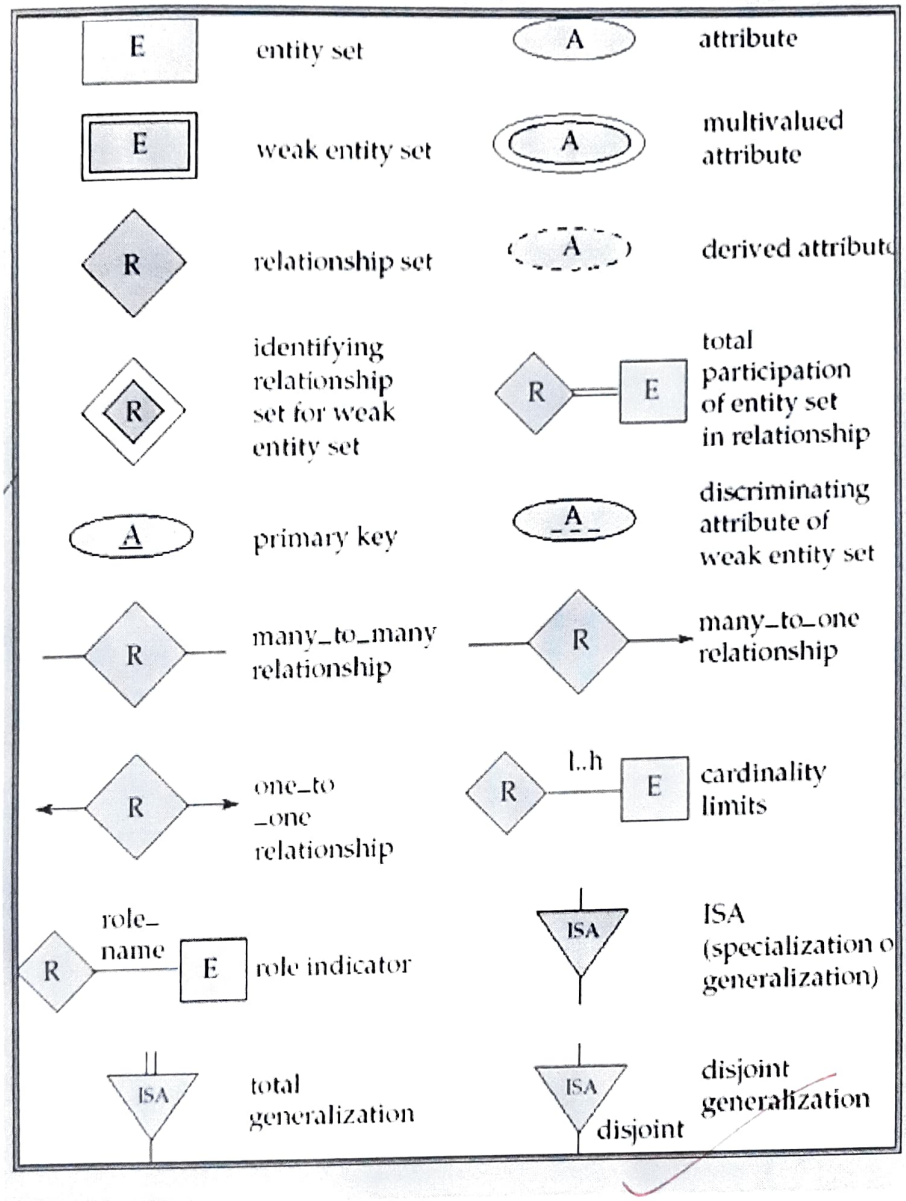
ER Diagram – Notations -

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Lines link attributes to entity sets and entity sets to relationship sets.
- Ellipses represent attributes.
- Double ellipses répresent multivalued attributes.
- Dashed ellipses denote derived attributes.
- Underline indicates primary key attributes.

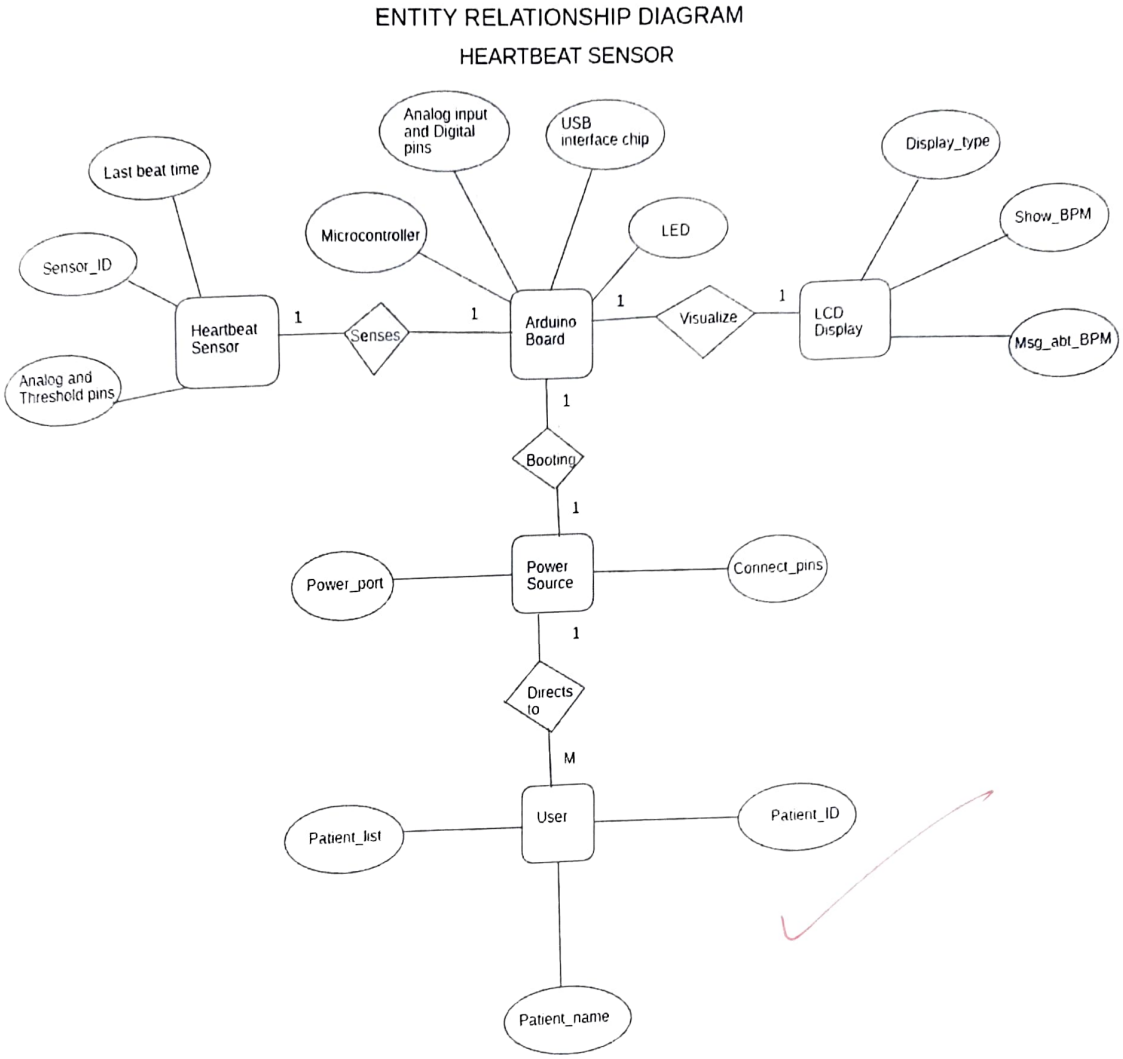
ER Diagram - Notations Representation -



ER Diagram - Notations Representation -



ER Diagram for ARDUINO HEARTBEAT SENSOR -



An entity relationship diagram (ERD) for a heartbeat sensor might include the following entities and relationships -

User - This entity represents the person who is wearing the heartbeat sensor. It might include attributes such as name,id and list of all the persons.

LCD Display - This entity represents the data that is collected by the heartbeat sensor. It might include attributes such as display_type,Show_BPM and Msg_abt_BPM.

Heartbeat Sensor - This entity represents the heartbeat sensor itself. It might include attributes such as Sensor_id, Last beat time, analog and Threshold pins.

Arduino Board - An Arduino board entity represents a physical device that serves as the core of a system. An Arduino board is a microcontroller board that allows for the creation of interactive objects and systems that can sense and control physical devices.It might include attributes such as Microcontroller, Analog and Digital pins, USB interface chip and LED.

Power Source - The power source entity could represent a physical device or source that provides power to other entities or components within a system. It might include attributes such as power_ports and connect_pins.

The relationships between these entities might include -

A person can have many instances of heartbeat data, but each instance of heartbeat data belongs to only one person.

A heartbeat sensor can collect many instances of heartbeat data, but each instance of heartbeat data is collected by only one heartbeat sensor.

A LCD Display will monitor many instances of heartbeat data, but each instance of heartbeat data is monitored on display.

An Arduino Board is an intermediate for creating this and working of those pins which is setted up with the Sensor.

Power source is connected to the Arduino for the functioning of this complete entity.

Overall, this ERD provides a visual representation of the data flow and relationships between the various entities involved in a heartbeat monitoring system, allowing for better design, implementation, and maintenance of the system.

14-3-2023

Result -

Thus, the entity relationship diagram was created successfully.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	8
Title of Experiment	Develop a Data Flow Diagram (Process-Up to Level 1)
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	09-03-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

M. Hema
14-3-2023

Staff Signature with date

Aim -

To develop the data flow diagram up to level 1 for the Heart beat Sensor System using Arduino Board.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Data Flow Diagram -

The DFD takes an input-process-output view of a system. That is, data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects are represented by labeled arrows, and transformations are represented by circles (also called bubbles). The DFD is presented in a hierarchical fashion. That is, the first data flow model (sometimes called a level 0 DFD or context diagram) represents the system as a whole. Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level.

The data flow diagram enables you to develop models of the information domain and functional domain. As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the application.
A few simple guidelines can aid immeasurably during the derivation of a data flow diagram:
(1) Level 0 data flow diagram should depict the software/system as a
single bubble;
(2) Primary input and output should be carefully noted;
(3) Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;
(4) All arrows and bubbles should be labeled with meaningful names;
(5) Information flow continuity must be maintained from level to level and
(6) One bubble at a time should be refined. There is a natural tendency to overcomplicate the data flow diagram. This occurs when you attempt to show too much detail too early or represent procedural aspects of the software in lieu of information flow.

Aim -

To develop the data flow diagram up to level 1 for the Heart beat Sensor System using Arduino Board.

Team Members -

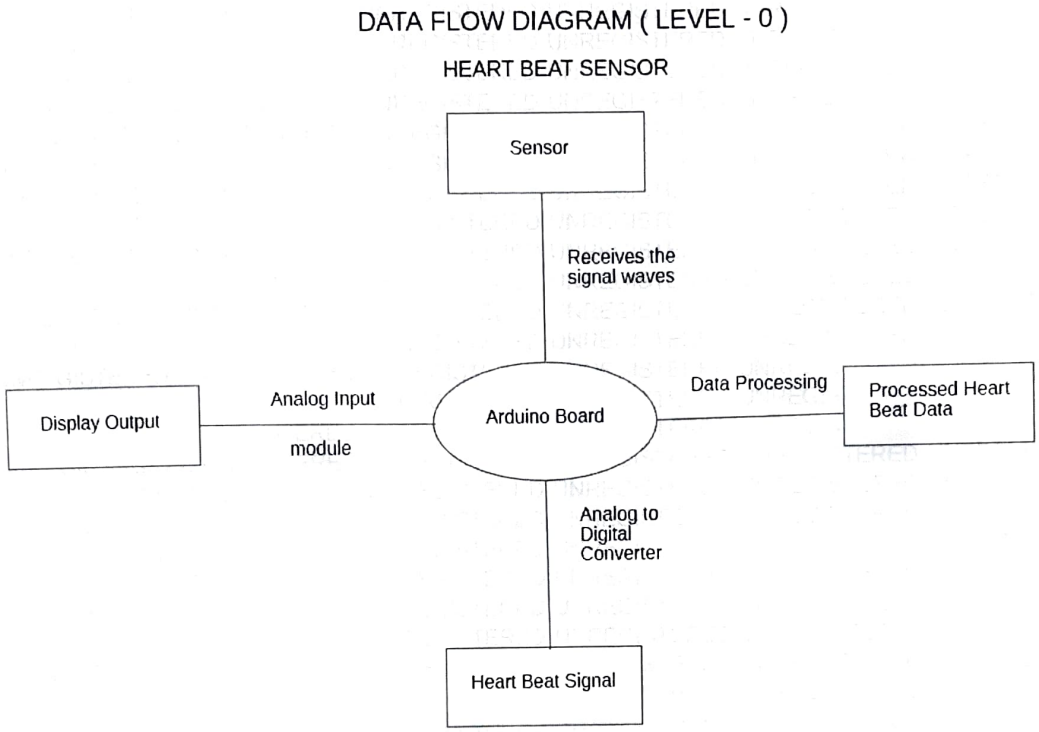
S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Data Flow Diagram -

The DFD takes an input-process-output view of a system. That is, data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects are represented by labeled arrows, and transformations are represented by circles (also called bubbles). The DFD is presented in a hierarchical fashion. That is, the first data flow model (sometimes called a level 0 DFD or context diagram) represents the system as a whole. Subsequent data flow diagrams refine the context diagram, providing increasing detail with each subsequent level.

The data flow diagram enables you to develop models of the information domain and functional domain. As the DFD is refined into greater levels of detail, you perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of data as it moves through the processes that embody the application.
A few simple guidelines can aid immeasurably during the derivation of a data flow diagram:
(1) Level 0 data flow diagram should depict the software/system as a
single bubble;
(2) Primary input and output should be carefully noted;
(3) Refinement should begin by isolating candidate processes, data objects, and data stores to be represented at the next level;
(4) All arrows and bubbles should be labeled with meaningful names;
(5) Information flow continuity must be maintained from level to level and
(6) One bubble at a time should be refined. There is a natural tendency to overcomplicate the data flow diagram. This occurs when you attempt to show too much detail too early or represent procedural aspects of the software in lieu of information flow.

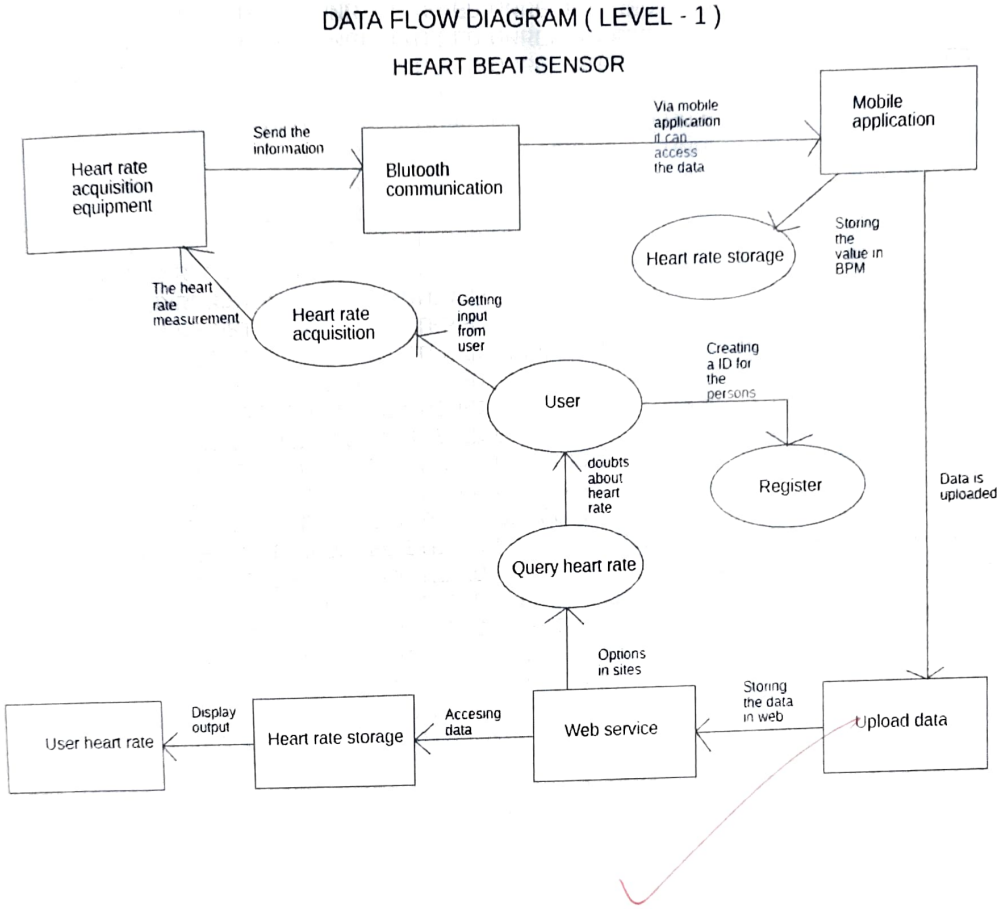
DFD (Level 0) -



The sensors in this diagram pick up the user's heartbeat and transmit a signal to the Arduino board. To obtain a more precise reading of the heartbeat, the analogue input module turns the signal into a digital value, which is then sent through the analog-to-digital converter.

This digital value is used by the data processing component to apply any necessary algorithms to remove noise and gather valuable data about the user's heartbeat. The processed heartbeat data is finally shown on the display component, which can be a visual display, a computer, or another device linked to the Arduino.

DFD (Level 1) -


DATA FLOW DIAGRAM (LEVEL - 1)
HEART BEAT SENSOR
Heart rate acquisition equipment
Send the information
Blutooth communication
Via mobile application it can access the data
Mobile application
Heart rate storage
Storing the value in BPM
The heart rate measurement
Heart rate acquisition
Getting input from user
User
Creating a ID for the persons
Register
doubts about heart rate
Query heart rate
Data is uploaded
Options in sites
User heart rate
Display output
Heart rate storage
Accesing data
Web service
Storing the data in web
Upload data

In the context of a heartbeat sensor system using Arduino, the heart rate acquisition equipment is a critical component that provides the input signal for the system. Without this equipment, the Arduino board would not have a way to detect the user's heartbeat and the system would not be able to function.

Bluetooth communications can be used in a heartbeat sensor system to transmit the heart rate data wirelessly from the Arduino board to other devices such as a smartphone, tablet, or computer. This can provide a more convenient way for users to monitor their heart rates, as they can view the data in real-time on a device without the need for any physical connections. It can also enable the heartbeat sensor system to send alerts or notifications to the user when their heart rate exceeds a certain threshold or falls below a certain range. This can be especially useful for individuals who need to monitor their heart rates closely, such as those with cardiovascular disease or athletes in training.

A mobile application can be used in a heartbeat sensor system to provide a more user-friendly and convenient way for users to monitor and manage their heart rate data. The mobile application can be designed to work in conjunction with the Arduino-based heartbeat sensor system and can provide a range of features and functionalities, such as: Real-time heart rate monitoring,Heart rate data storage and analysis,Alerts and notifications and, User profile management.

Web services can be used in a heartbeat sensor system to provide remote access to heart rate data and other information stored in the system. By integrating web services into the system, users can access their heart rate data from any location with an internet connection, and from a range of devices such as smartphones, tablets, laptops, and desktop computers. The use of web services can provide several benefits like - Remote data access, Real-time data updates, Data security and Integration with other systems.

Result -

Thus, the data flow diagrams have been created for the Heart beat Sensor System using Arduino Board.

14-3-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	9
Title of Experiment	Design a Sequence and Collaboration Diagram
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	10-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

21-4-2023

Staff Signature with date

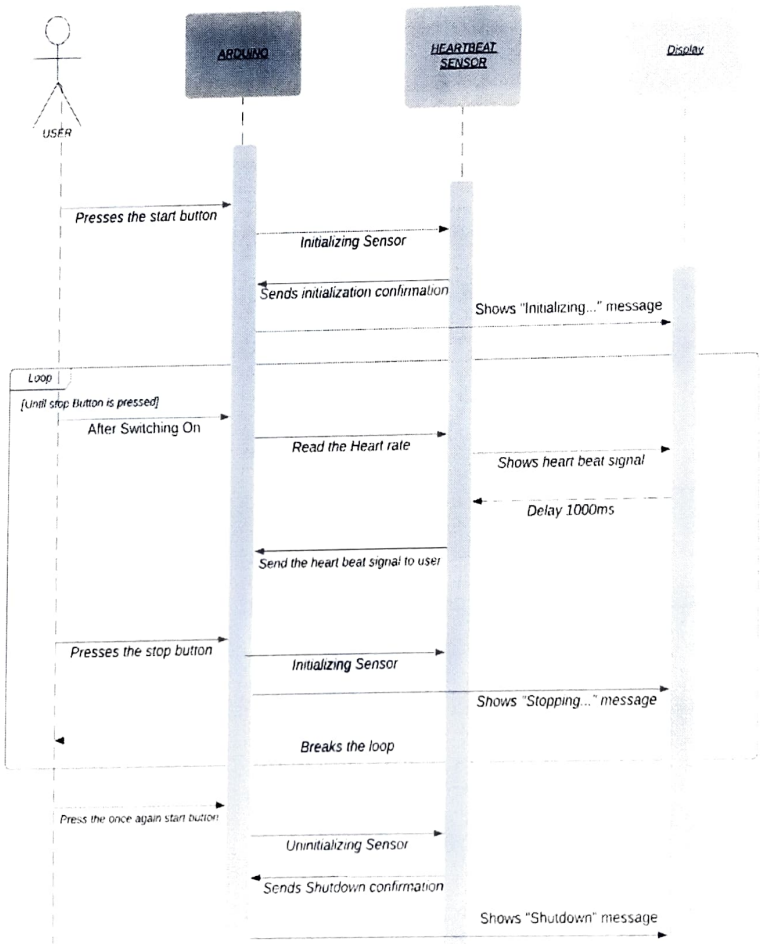
Aim -

To create the sequence and collaboration diagram for the Arduino Heartbeat Sensor.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA211003010301	SHREYA RANJAN	Member

Sequence Diagram -



A sequence diagram is a type of UML diagram that depicts the interactions between objects in a system over time. It illustrates the sequence of messages exchanged between objects to accomplish a task or a set of tasks.

In the context of an Arduino-based heartbeat sensor, a sequence diagram can show the sequence of events that occur when a user interacts with the device. For example, the diagram can show how the heartbeat sensor initializes, detects a heartbeat, and displays the heartbeat signal to the user.

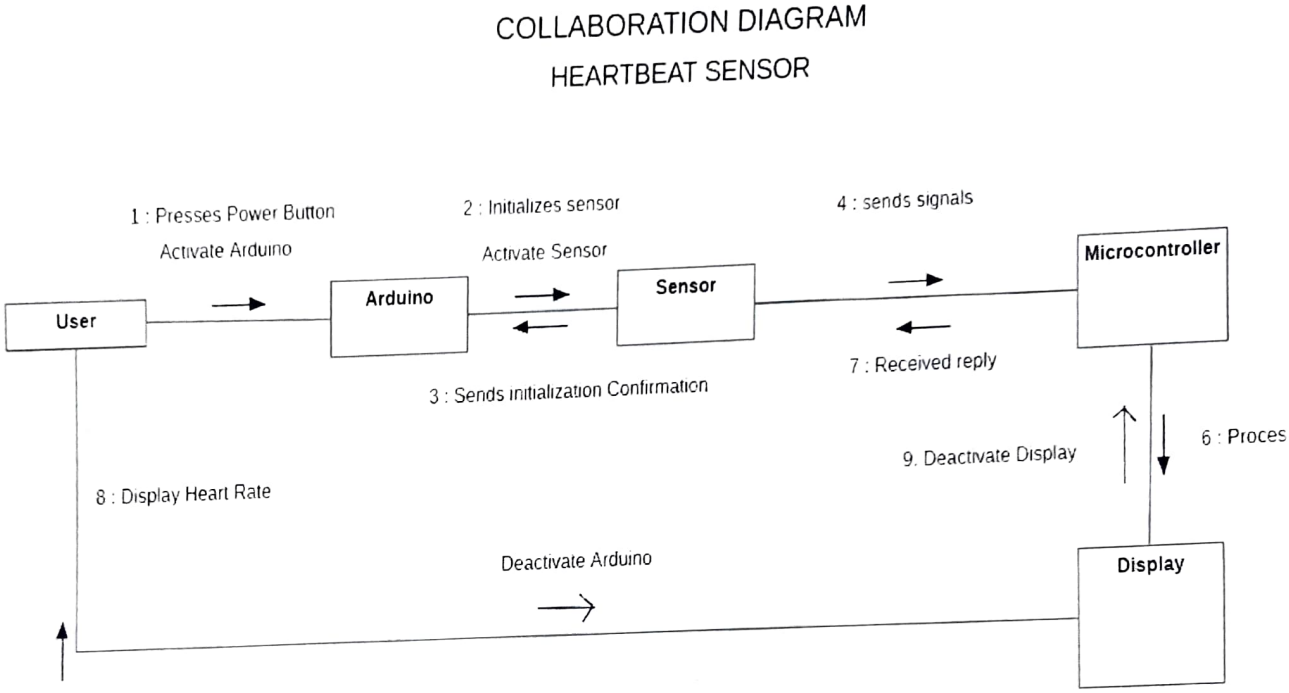
The sequence diagram typically consists of objects, which represent the entities or components in the system, and messages, which represent the actions or operations performed by the objects. The messages are typically arranged in a chronological order, with the horizontal axis representing time.

Each object has its lifeline, which represents the duration of the object's existence in the system. When an object receives a message, its lifeline is extended, and when it sends a message, a message arrow is created to represent the communication between the objects.

Sequence diagrams can be useful in the development and testing of software systems, as they can help identify potential issues or bottlenecks in the system's design. They can also be used as documentation to communicate the system's behavior and requirements to stakeholders.

Overall, a sequence diagram is a useful tool for visualizing the interactions between objects in a system and can help developers and stakeholders understand the system's behavior and requirements.

Collaboration Diagram -



In a collaboration diagram, the focus is on the objects and their interactions, rather than the sequence of messages like in a sequence diagram. In this diagram, we see the objects involved in the system: User, Arduino, HeartBeatSensor, and Display.

The collaboration starts with the User object pressing the power button, which activates the Arduino object. The Arduino then activates the HeartBeatSensor object to initialize the sensor. Once the sensor sends its initialization confirmation, the HeartBeatSensor object is deactivated and the Arduino activates the Display object to show an "Initializing..." message.

The loop represents the continuous process of the HeartBeatSensor object sending heartbeat signals to the Arduino object, which in turn activates the Display object to show the signal. This loop continues until the User object presses the stop button, at which point the loop is broken and the Arduino activates the Display object to show a "Stopping..." message.

After the loop is broken, the Arduino deactivates the HeartBeatSensor object to shut down the sensor and activates the Display object to show a "Shutdown" message. Finally, the Arduino object is deactivated, and the collaboration diagram ends.

Again, note that this is just one possible collaboration diagram for an Arduino Uno heart beat sensor system, and actual implementations may differ depending on the specific requirements and components used.

Result -

Thus, the sequence and collaboration diagrams were created for the Arduino Heartbeat sensor.

21-4-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	10
Title of Experiment	Develop a Testing Framework/User Interface
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	15-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

21-4-2023

Staff Signature with date

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	10
Title of Experiment	Develop a Testing Framework/User Interface
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	15-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

21-4-2023

Staff Signature with date

Aim -

To develop the testing framework and/or user interface framework for the Arduino Heartbeat Sensor.

Team Members:

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Executive Summary -

Test Plan -

Scope of Testing -

The scope of testing the software application of an Arduino-based heartbeat sensor includes verifying that the software is functioning correctly and accurately capturing and displaying heart rate data. This would involve testing the software's ability to read the sensor's data, interpret it correctly, and display it accurately. Additionally, the scope would include testing for any potential errors or bugs in the software that could affect its functionality.

Objectives -

The objectives of testing the software application of an Arduino-based heartbeat sensor would include ensuring that the software is accurate, reliable, and user-friendly. Specifically, the testing should aim to -

- Verify that the software accurately captures and displays heart rate data.

- Confirm that the software is reliable and functions correctly under various conditions.

- Identify and address any bugs or issues that could impact the software's performance.

- Ensure that the software is user-friendly and easy to use for both technical and non-technical users.

Approach -

To test the software application of an Arduino-based heartbeat sensor, the following approach could be taken -

- Define a set of test cases that cover all the functionalities of the software.

- Execute the test cases using the Arduino sensor and software, and record the results.

- Verify that the software accurately captures and displays the heart rate data.

- Test the software under different scenarios to ensure its reliability, including different environmental conditions and user inputs.

- Perform regression testing to ensure that changes or updates to the software do not break existing functionality.

- Use debugging tools to identify and address any bugs or issues found during testing.

- Gather feedback from users to ensure that the software is user-friendly and meets their needs.

- Overall, the testing approach should be thorough and systematic to ensure that the software application of the Arduino-based heartbeat sensor is accurate, reliable, and user-friendly.

Functional Testing -

Functional testing is a type of software testing that focuses on verifying that the software application or system meets its specified functional requirements. It involves testing the software's features and functionality by providing inputs and verifying that the expected outputs are produced.

The purpose of functional testing is to ensure that the software application or system works as intended and meets the user's needs. Functional testing is typically performed by testers who are not involved in the development of the software and may include black-box testing techniques, such as equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.

Functional testing can be performed manually or with the help of automated testing tools. It is an essential step in the software development life cycle, as it helps to identify defects or bugs in the software application or system and ensures that it meets the required specifications and user expectations.

The functional Testing of the software involves:

Sensor Data Reading - Verify that the software accurately reads the sensor data and stores it correctly in the database or other storage mechanisms.

Heart Rate Calculation - Verify that the software correctly calculates the heart rate from the sensor data.

Display - Verify that the software displays the heart rate in a clear and understandable format on the output device, such as an LCD or LED display.

User Interface - Verify that the user interface is user-friendly and easy to use for both technical and non-technical users.

Data Storage - Verify that the software correctly stores and retrieves data from the database or other storage mechanisms.

Non-functional Testing -

Non-functional testing is a type of software testing that focuses on evaluating the software's performance, usability, security, and other aspects that are not related to the software's functional requirements. Unlike functional testing, which focuses on the software's features and functionality, non-functional testing evaluates how well the software performs in various conditions, such as high load or stress, different environments, and security vulnerabilities.

The Non functional Testing of the software involves:

Performance Testing - Verify that the software can handle a large volume of data without crashing or slowing down. This would involve testing for factors such as response time, throughput, and resource utilization.

Compatibility Testing - Verify that the software works correctly with various types of sensors and hardware platforms.

Usability Testing - Verify that the software is easy to use and meets the user's expectations for usability, such as response time, user interface design, and error handling.

Security Testing - Verify that the software is secure and free from vulnerabilities, such as SQL injection or cross-site scripting (XSS).

Reliability Testing - Verify that the software is reliable and performs consistently under different scenarios, such as different environmental conditions and user inputs.

Overall, the functional testing ensures that the software application of an Arduino-based heartbeat sensor accurately captures, calculates, and displays heart rate data. The non-functional testing ensures that the software performs reliably, securely, and efficiently under various conditions and meets user expectations for usability.

The Non functional Testing of the software involves:

Performance Testing - Verify that the software can handle a large volume of data without crashing or slowing down. This would involve testing for factors such as response time, throughput, and resource utilization.

Compatibility Testing - Verify that the software works correctly with various types of sensors and hardware platforms.

Usability Testing - Verify that the software is easy to use and meets the user's expectations for usability, such as response time, user interface design, and error handling.

Security Testing - Verify that the software is secure and free from vulnerabilities, such as SQL injection or cross-site scripting (XSS).

Reliability Testing - Verify that the software is reliable and performs consistently under different scenarios, such as different environmental conditions and user inputs.

Overall, the functional testing ensures that the software application of an Arduino-based heartbeat sensor accurately captures, calculates, and displays heart rate data. The non-functional testing ensures that the software performs reliably, securely, and efficiently under various conditions and meets user expectations for usability.

Types of Testing, Methodology, Tools -

Category	Methodology	Tools Required
Functional Testing	Black Box Testing	Arduino IDE, Arduino Sensor
	Equivalence Partitioning	Multimeter, Oscilloscope
	Boundary Value Analysis	Function Generator
	Decision Table Testing	
	State Transition Diagrams	
Non- Functional Testing	Performance Testing	JMeter, LoadRunner, Gatling
	Compatibility Testing	Different types of sensors and hardware platforms
	Usability Testing	Usability Testing Frameworks, User Survey Tools
	Security Testing	Security Testing Frameworks, Penetration Testing Tools
	Reliability Testing	Fault Injection, Stress Testing, Failure Analysis Tools

Result -

Thus, the testing framework/user interface framework has been created for the Heartbeat Sensor.

21-4-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	11
Title of Experiment	Test Cases
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R(RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	18-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

21-4-2023

Staff Signature with date

Aim -

To develop the test cases manual for the Arduino Heartbeat Sensor.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Test Case -

Functional Test Cases -

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1	Sensor Data Reading	Read Sensor Data	1.Start the software application. 2.Place the sensor on the subject. 3.Verify that the sensor data is displayed on the output.	The software should read sensor data from the sensor.	The numerical heart rate value that is read from the sensor and displayed on the output device.	Pass	success
2	Heart Rate Calculation	Calculate HeartRate on the	1. Start the software application. 2. Verify that the calculated heart rate is displayed.	The software should calculate heart rate from sensor data.	It calculated no. of heartbeats per minute (BPM).	Pass	Success
3	Display	Display Heart Rate	1. Start the software application. 2. Verify that the displayed heart rate is accurate.	The software should display heart rate on the output device.	It displayed no. of heartbeats per minute (BPM) which is a critical indicator of cardiovascular health.	Pass	Success
4	User Interface	User Interface	1. Verify that the user interface is easy to use.	The user interface should be user-friendly and intuitive.	The user can see the heart rate in real-time and track changes over time.	Pass	Success

Non-Functional Test Cases -

Test ID (#)	Test Scenario	Test Case	Execution Steps	Expected Outcome	Actual Outcome	Status	Remarks
1.	Performance Testing	Stress Testing	1.Start the software application. 2. Generate a high volume of data. 3. Verify that the software responds quickly and efficiently.	The software should perform well under high load and stress.	It ensured that it is accurate and reliable in detecting and measuring heart rate.	Pass	Success
2.	Security Testing	Security Testing	1.Verify that the software is free from known vulnerabilities. 2. Test for SQL injection and cross-site scripting(XSS) .	The software should be secure and free from security threats.	It ensured that the user's personal and health information is kept and secure.	Pass	Success
3.	Compatibility Testing	Compatibility Testing	1.Verify that the software works on different hardware. 2. Verify that the software works on different operating systems.	The software should work on different hardware platforms. The software should work on different operating systems.	The outcome of compatibility testing is to verify that the software works on different hardware platforms and operating systems.	Pass	Success
4.	Usability Testing	Usability Testing	1.Verify that the user	The software should be user -	It was easy to use and provided	Pass	Success

			interface is easy to use. 2. Conduct a user survey to gather feedback.	friendly and intuitive.	consistent reading. No discomfort caused while wearing the heartbeat sensor.		
5	Reliability Testing	Reliability Testing	1.Test The software under different environmental conditions.	The software should perform consistently under different scenarios.	The outcome of reliability testing is to verify that the software can perform consistently and reliably over an extended period of time.	Pass	Success

Result -

Thus, the test case manual has been created for the Heartbeat Sensor.

21-4-2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	12
Title of Experiment	Manual Test Case Reporting
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Number	RA2111003010301
Date of Experiment	21-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

25-5-23

Staff Signature with date

Aim -

To prepare the manual test case report for the Arduino Heartbeat Sensor.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Manual Test Cases -

The following table shows the manual test cases plan and their status -

Category	Progress Against Plan	Status
Functional Testing	Completed all test cases on schedule	Completed
Performance Testing	Completed load testing, but not stress testing yet	Completed
Security Testing	Started penetration testing, but not complete yet	Completed
Usability Testing	Completed user testing, but not accessibility testing yet	Completed
Compatibility Testing	Completed testing on Windows, but not on Mac OS yet	Completed
Regression Testing	Completed initial round of testing, but more rounds needed	Completed
Reliability Testing	Completed initial testing, but more testing needed for extended period of time	In Progress

Test Cases Coverage -

Manual test case coverage and status for various functionalities are shown as -

Test Cases	Test Case Coverage (%)	Status
Heart Rate Sensor Reading	100%	Pass
Data Storage and Retrieval	90%	In Progress
User Interface	80%	In Progress
Error Handling	95%	Pass
Connectivity	70%	Pass
Battery Life	80%	Pass

This table provides an overview of the testing categories, the progress made against the plan, and the status of completion for each category. This table can be used to track the overall progress of testing and identify any areas that require further attention or resources.

Result -

Thus, the test case report has been created for the Arduino Heartbeat Sensor.

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

School of Computing

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC206J

Course Name: Software Engineering and Project Management

Experiment No	13
Title of Experiment	Provide the details of Architecture Design/Framework/Implementation
Name of the candidate	Shreya Ranjan
Team Members	Ganesh K (RA2111003010298) Harihara Krishna R (RA2111003010300)
Register Numbers	RA2111003010301
Date of Experiment	24-04-2023

Mark Split Up

S. No	Description	Maximum Mark	Mark Obtained
1	Exercise	5	5
2	Viva	5	5
	Total	**10**	10

25-5-23

Staff Signature with date

Aim -

To provide the details of architectural design/framework/implementation of arduino heartbeat sensor.

Team Members -

S No	Register No	Name	Role
1	RA2111003010298	GANESH K	Rep/Member
2	RA2111003010300	HARIHARA KRISHNA RAMESH	Member
3	RA2111003010301	SHREYA RANJAN	Member

Architectural Design -

The Arduino Heartbeat Sensor is a simple yet effective way to measure and monitor heart rate. This sensor can be used in a variety of applications, including fitness trackers, health monitors, and medical devices. In this article, we will discuss the architectural design of the Arduino Heartbeat Sensor.

The Arduino Heartbeat Sensor consists of a few key components, including a light emitter, a light detector, and an Arduino microcontroller. The light emitter is typically an LED that emits a light in the red or infrared spectrum, which is absorbed by the blood in the finger. The light detector is a photodiode that detects the amount of light that is transmitted through the finger. The Arduino microcontroller is responsible for processing the data from the light detector and calculating the heart rate.

The architecture of the Arduino Heartbeat Sensor can be divided into two main parts: the hardware and the software. The hardware consists of the physical components of the sensor, including the LED, photodiode, and Arduino microcontroller. The software consists of the code that runs on the microcontroller and processes the data from the sensor.

Hardware Architecture -

The hardware architecture of the Arduino Heartbeat Sensor is relatively simple. It consists of the following components:

LED- The LED is used to emit light in the red or infrared spectrum.

Photodiode- The photodiode is used to detect the amount of light that is transmitted through the finger.

Amplifier - The amplifier is used to amplify the signal from the photodiode.

Analog-to-Digital Converter (ADC)- The ADC is used to convert the analog signal from the amplifier into a digital signal that can be processed by the Arduino microcontroller.

Arduino Microcontroller -The Arduino microcontroller is responsible for processing the data from the ADC and calculating the heart rate.

Software Architecture -

The software architecture of the Arduino Heartbeat Sensor is responsible for processing the data from the hardware and calculating the heart rate. The software architecture consists of the following components:

Data Acquisition - The data acquisition component is responsible for acquiring the raw data from the ADC.

Signal Processing - The signal processing component is responsible for filtering and smoothing the raw data to remove noise and artifacts.

Peak Detection - The peak detection component is responsible for detecting the peaks in the signal, which correspond to each heartbeat.

Heart Rate Calculation - The heart rate calculation component is responsible for calculating the heart rate based on the time between each peak.

User Interface - The user interface component is responsible for displaying the heart rate to the user.

Code -

```
//code copied from arduino.cc
int pulsePin = A0;                 // Pulse Sensor purple wire connected to analog pin A0
int blinkPin = 13;                 // pin to blink led at each beat

// Volatile Variables, used in the interrupt service routine!
volatile int BPM;                  // int that holds raw Analog in 0. updated every 2mS
volatile int Signal;               // holds the incoming raw data
volatile int IBI = 600;            // int that holds the time interval between beats! Must be
seeded!
volatile boolean Pulse = false;    // "True" when User's live heartbeat is detected. "False"
when not a "live beat".
volatile boolean QS = false;       // becomes true when Arduoino finds a beat.

static boolean serialVisual = true;   // Set to 'false' by Default.  Re-set to 'true' to see Arduino
Serial Monitor ASCII Visual Pulse

volatile int rate[10];                    // array to hold last ten IBI values
volatile unsigned long sampleCounter = 0;          // used to determine pulse timing
volatile unsigned long lastBeatTime = 0;           // used to find IBI
volatile int P = 512;                     // used to find peak in pulse wave, seeded
volatile int T = 512;                     // used to find trough in pulse wave, seeded
volatile int thresh = 525;                // used to find instant moment of heart beat, seeded
volatile int amp = 100;                   // used to hold amplitude of pulse waveform, seeded
```

```
volatile boolean firstBeat = true;        // used to seed rate array so we startup with reasonable
BPM
volatile boolean secondBeat = false;      // used to seed rate array so we startup with
reasonable BPM

void setup()
{
 pinMode(blinkPin,OUTPUT);         // pin that will blink to your heartbeat!
 Serial.begin(115200);             // we agree to talk fast!
 interruptSetup();                 // 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);
}


// Where the Magic Happens
void loop()
{
  serialOutput();

 if (QS == true) // A Heartbeat Was Found
    {
     // BPM and IBI have been Determined
     // Quantified Self "QS" true when arduino finds a heartbeat
     serialOutputWhenBeatHappens(); // A Beat Happened, Output that to serial.
     QS = false; // reset the Quantified Self flag for next time
    }

  delay(20); //  take a break
}
```

```
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
}

void serialOutput()
{  // Decide How To Output Serial.
 if (serialVisual == true)
  {
    arduinoSerialMonitorVisual('-', Signal);   // goes to function that makes Serial Monitor
Visualizer
  }
 else
  {
    sendDataToSerial('S', Signal);     // goes to sendDataToSerial function
  }
}

void serialOutputWhenBeatHappens()
{
 if (serialVisual == true) //  Code to Make the Serial Monitor Visualizer Work
   {
    Serial.print(" Heart-Beat Found ");  //ASCII Art Madness
    Serial.print("BPM: ");
    Serial.println(BPM);
```

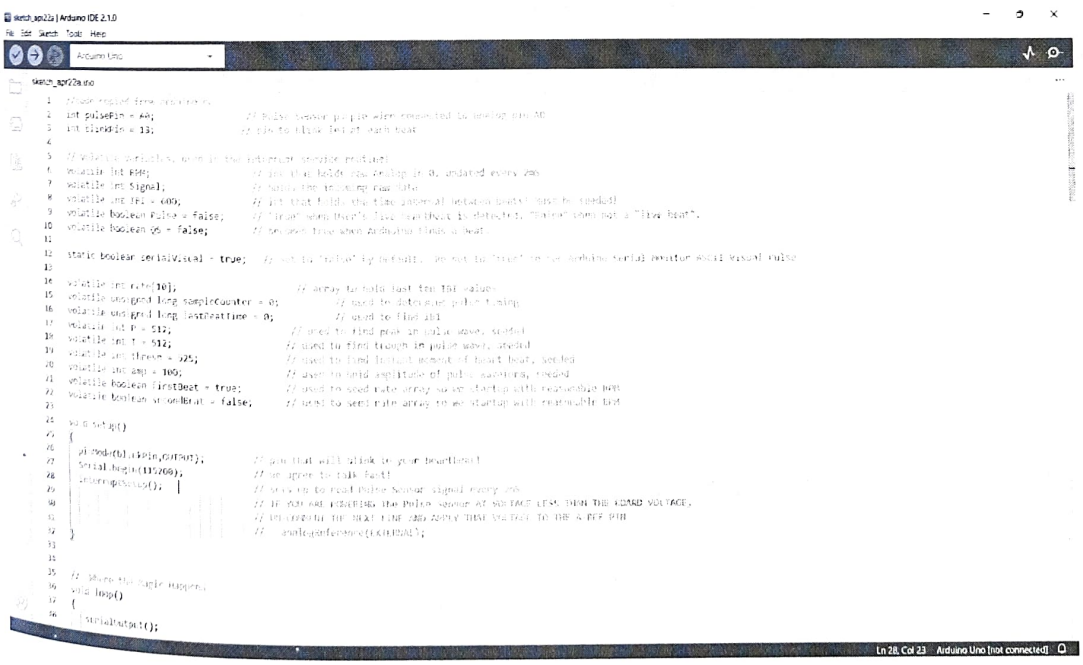
```
  }
else
  {
   sendDataToSerial('B',BPM);   // send heart rate with a 'B' prefix
   sendDataToSerial('Q',IBI);   // send time between beats with a 'Q' prefix
  }
}

void arduinoSerialMonitorVisual(char symbol, int data )
{
 const int sensorMin = 0;      // sensor minimum, discovered through experiment
 const int sensorMax = 1024;    // sensor maximum, discovered through experiment
 int sensorReading = data; // map the sensor range to a range of 12 options:
 int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
 // do something different depending on the
 // range value:
}


void sendDataToSerial(char symbol, int data )
{
  Serial.print(symbol);
  Serial.println(data);
}

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
```

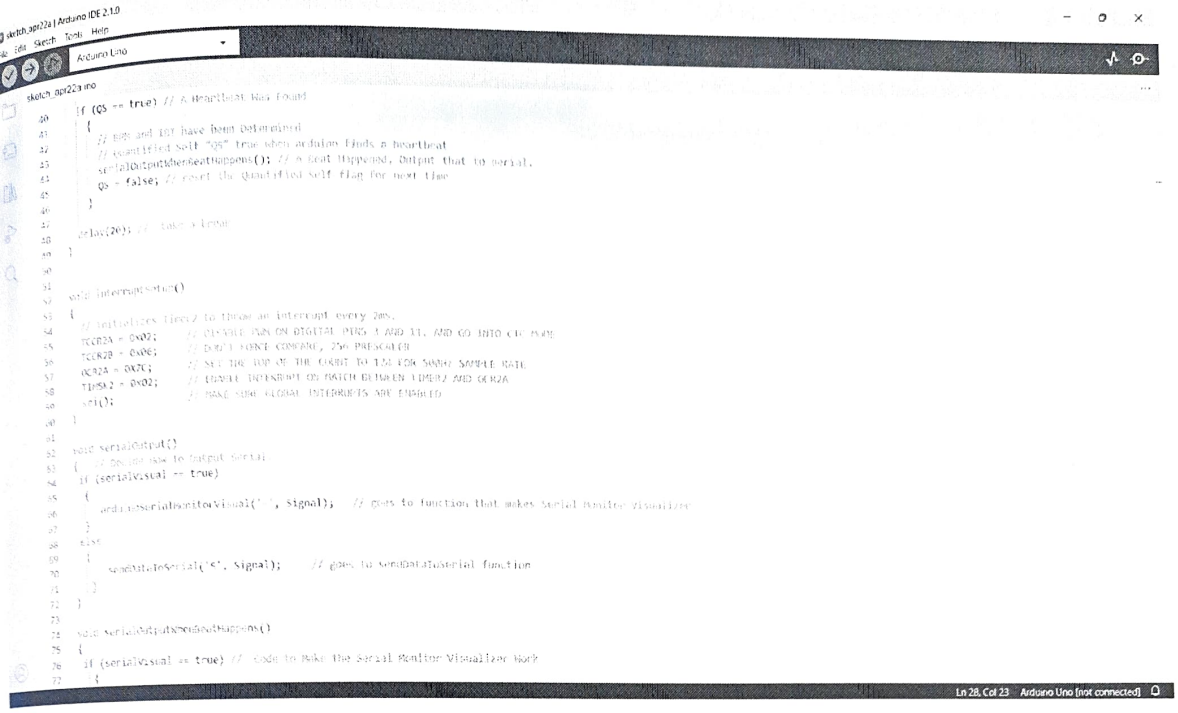
```
{
 if (Signal < T) // T is the trough
 {
  T = Signal; // keep track of lowest point in pulse wave
 }
}

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(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;
     }
    }

    if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
    {
```

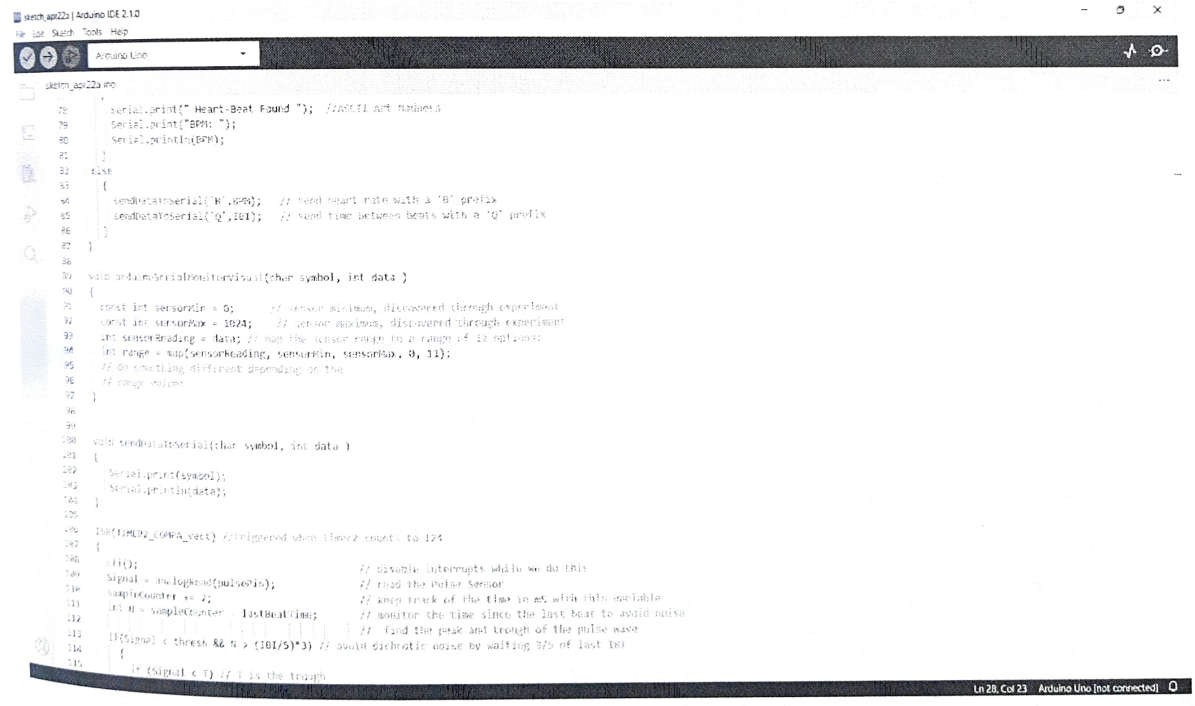
```
    firstBeat = false;                  // clear firstBeat flag
    secondBeat = true;                  // set the second beat flag
    sei();                              // enable interrupts again
    return;                             // IBI value is unreliable so discard it
   }
  // keep a 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;
 }
```

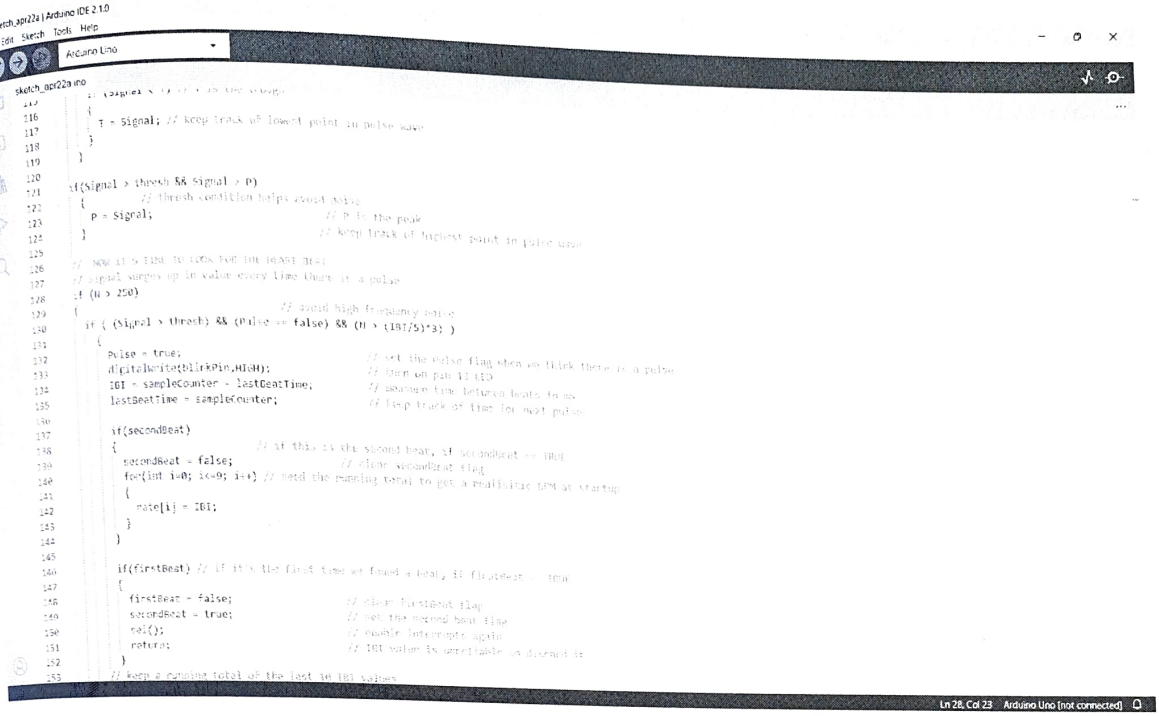
```
if (N > 2500)
                    // if 2.5 seconds go by without a beat
 {
  thresh = 512;                    // set thresh default
  P = 512;                         // set P default
  T = 512;                         // set T default
  lastBeatTime = sampleCounter;          // bring the lastBeatTime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


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

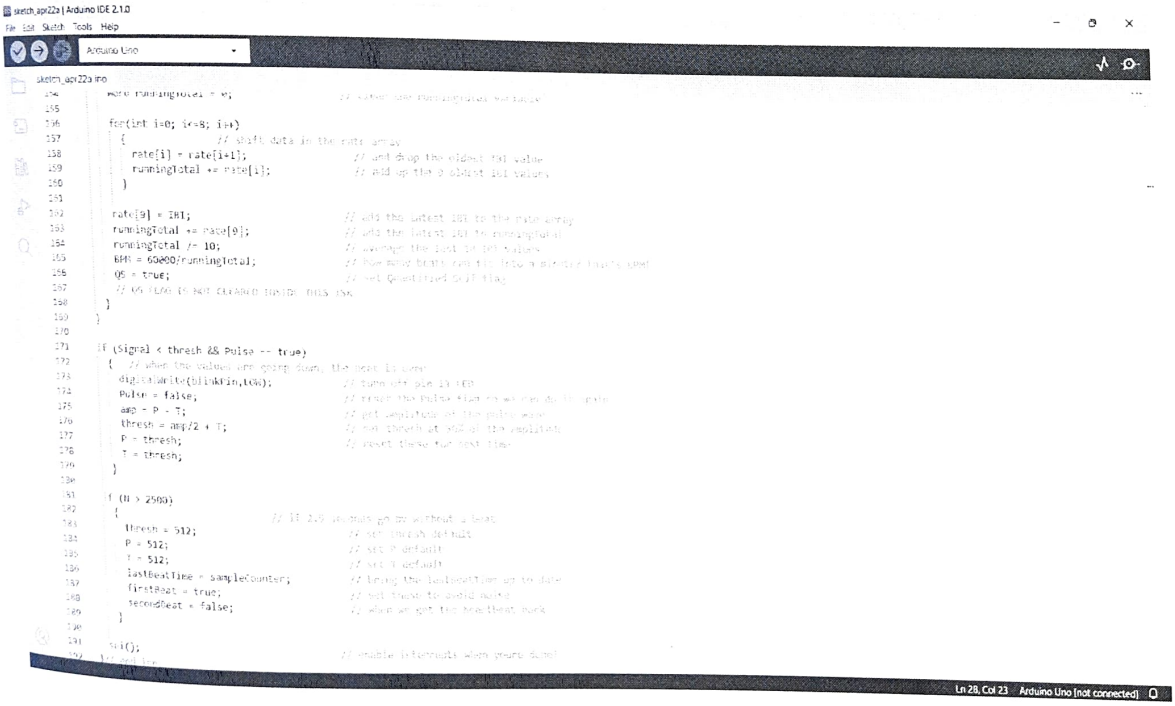
Implementation -



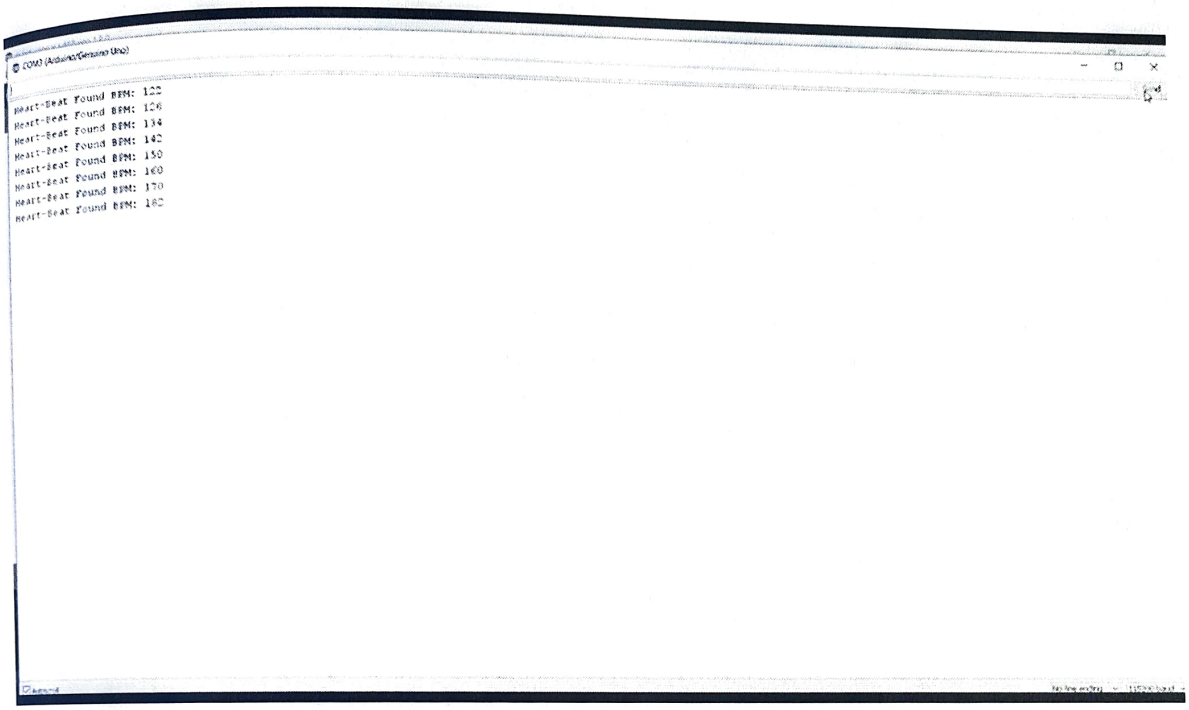

sketch_apr22a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_apr22a.ino
if (QS == true) // A Heartbeat Was Found
{
// BPM and IBI have been Determined
// Quantified Self "QS" true when Arduino finds a heartbeat
sendDataToSerial('B', BPM); // Send heart rate with a 'B' prefix
sendDataToSerial('Q', 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 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
}
void serialOutput()
{ // Decide how to output Serial.
if (serialVisual == true)
{
arduinoSerialMonitorVisual('', Signal); // goes to function that makes Serial Monitor Visualizer
}
else
{
sendDataToSerial('S', Signal); // goes to sendDataToSerial function
}
}
void serialOutputWhenBeatHappens()
{
if (serialVisual == true) // Code to Make the Serial Monitor Visualizer Work
{
Ln 28, Col 23 Arduino Uno [not connected]


sketch_apr22a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_apr22a.ino
Serial.print(" Heart-Beat Found "); //ASCII Art Madness
Serial.print("BPM: ");
Serial.println(BPM);
}
else
{
sendDataToSerial('B',BPM); // send heart rate with a 'B' prefix
sendDataToSerial('Q',IBI); // send time between beats with a 'Q' prefix
}
}
void arduinoSerialMonitorVisual(char symbol, int data)
{
const int sensorMin = 0; // sensor minimum, discovered through experiment
const int sensorMax = 1024; // sensor maximum, discovered through experiment
int sensorReading = data; // map the sensor range to a range of 12 options:
int range = map(sensorReading, sensorMin, sensorMax, 0, 11);
// do something different depending on the
// range value:
}
void sendDataToSerial(char symbol, int data)
{
Serial.print(symbol);
Serial.println(data);
}
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
Ln 28, Col 23 Arduino Uno [not connected]


sketch_apr22a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_apr22a.ino
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(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 realistic BPM at startup
{
rate[i] = IBI;
}
}
if(firstBeat) // if it's the first time we found a beat, if firstBeat == TRUE
{
firstBeat = false; // clear firstBeat flag
secondBeat = true; // set the second beat flag
sei(); // enable interrupts again
return; // IBI value is unreliable so discard it
}
// Keep a running total of the last 10 IBI values
Ln 28, Col 23 Arduino Uno [not connected]


sketch_apr22a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_apr22a.ino
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 1000
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 = false; // when we get the heartbeat back
}
sei(); // enable interrupts when youre done!
Ln 28, Col 23 Arduino Uno [not connected]

Output -



Result -

Thus, the details of architectural design/framework/implementation along with the screenshots were provided.

CONCLUSION

A heartbeat sensor using an Arduino board is a useful tool for monitoring heart rate in various settings. The sensor is based on a simple yet effective photoplethysmography technique that detects changes in blood flow through the fingertip. The Arduino board processes the sensor data using an algorithm that calculates the heart rate based on the detected signal. The heart rate can then be displayed on an LCD screen or sent to a computer or mobile device for further analysis.

The heartbeat sensor using an Arduino board is a cost-effective and easy-to-implement solution for monitoring heart rate. It can be used in a variety of applications, including fitness tracking, medical monitoring, and stress management. The sensor can be customized and integrated into various projects and systems, making it a versatile tool for measuring heart rate. Overall, the heartbeat sensor using an Arduino board is a valuable addition to the field of biometric monitoring and has the potential to improve health outcomes and enhance wellness.

2-5-23

REFERENCES

1. https://circuitdigest.com/microcontroller-projects/heartbeat-monitor-project-using-arduino
2. https://youtu.be/JlODP0riCJk
3. https://projecthub.arduino.cc/
4. https://hackr.io/blog/arduino-projects
5. https://all3dp.com/1/best-cool-arduino-projects/