From Bodily Sensors to the Cloud and Back

Design Document · Group 26

Client – Goce Trajcevski
Advisors – Goce Trajcevski, Liang Dong, Sung Min Kang

Isaac Zahau – *Front End/UI* – isanga@iastate.edu
Justin Worely – *Cloud Engineer* – jmworely@iastate.edu
John Kivley – *Electrical Engineer* – jekivley@iastate.edu
Richa Patel – *Database Engineer* – rppatel@iastate.edu
Michael Lauderback – *Embedded Systems Engineer* – mlauderb@iastate.edu

Team Website
`https://sddec20-26.sd.ece.iastate.edu`

Executive Summary

Development Standards and Practices

- IEEE standards
- Waterfall Software Development
- Circuit, Block, and Use Case diagrams

Summary of Requirements

- At least 2 sensors
- Data stored on the cloud
- Transmit and receive data securely
- User-friendly user interface

Applicable Courses from Iowa State University Curriculum

- EE 201 – Electric Circuits
- EE 230 – Electron Circuits and Systems
- EE 285 – Problem Solving Methods and Tools for Electrical Engineering
- EE 321 – Communication Systems I
- CPRE 281 – Digital Logic
- CPRE 288 – Embedded Systems I: Introduction
- CPRE 388 – Embedded Systems II: Mobile Platforms
- COM S 227 – Object-oriented Programming
- COM S 228 – Introduction to Data Structures
- COM S 319 – Construction of User Interfaces
- COM S 309 – Software Development Practices
- SE 329 – Software Project Management
- SE 339 – Software Architecture and Design

- COM S 362 – Object-Oriented Analysis and Design
- COM S 363 - Introduction to Database Management Systems
- COM S 474 – Introduction to Machine Learning

New Skills or Acquired Knowledge

- Gained knowledge of industry tools used for hardware design such as KiCad.
- Gained knowledge and skill in designing electrical hardware starting from an idea to the final product.
- Gained knowledge and honed skills in troubleshooting and debugging hardware designs and learning multiple methods to conduct troubleshooting.
- Gained knowledge in project documentation practices.
- Gained knowledge in Amazon DynamoDB
- Gained knowledge on AWS infrastructure.
- Gained knowledge on features of Spark and Cassandra
- Gained knowledge on different types of databases.
- Gained knowledge in MVC

Contents

List of Figures

List of Tables

1 Introduction

1.1 Acknowledgement

We would like to acknowledge Goce Trajcevski, Liang Dong, and Sung Min Kang. Goce Trajcevski is our advisor. He has been very helpful, guiding the project in a successful direction. Sung Min Kang and Liang Dong have been there to offer great advice in regards to sensors for our project. We would also like to acknowledge Sarah Radke, Jessica Terrell, and John Stankovic for they were the sources of our market survey.

1.2 Project and Problem Statement

General Problem:
Doctors cannot monitor all of their patients at once. At a medical center or hospital, a doctor must allocate their time accordingly to the needs of their patients. Some patients require more attention than others due to the severity of their condition. In addition, some patients wish to not remain in a medical center or hospital; however, their condition requires a higher level of monitoring from a medical professional.

General Solution:
To solve these and other possible needs, we are using remote data storage using a cloud platform and sensors placed in different locations on the human body. We will incorporate different sensors placed in key locations that all talk to a master control unit (MCU). This MCU will then process the data from all of the sensors. If the mobile application is open, it will transmit live statistics to the mobile application. Along with the mobile application the MCU will also send the data that was collected to the cloud platform. Once there, that data will be stored and analyzed to provide the user with multiple ways to view their statistics, as well as their medical professional. This way, it is easier for doctors to monitor multiple patients, as well as give the patient the freedom to leave the hospital or medical center since the doctors can now monitor the patient remotely.

1.3 Operational Environment

The end product is expected to be operational under various environments whether inside a hospital or home, or outside in sunshine, snow or rain. In addition, it is also expected to operate properly, even when exposed to sweat, repeated motion, mild detergents, and various clothing material.

1.4 Requirements

The following is an itemized list of requirements for the project.

- Use at least two on-body sensors that will measure various vitals of the human body to create an IoT system from monitoring bodily functions.
- Sensors must have low power consumption, be wearable, and have a dedicated power supply to ensure the device is portable. Furthermore it needs to be reliable to ensure the product can be used on a consistent basis collecting information accurately.
- The sensors must be able to transmit data to an MCU using digital signals where MCU relays the data to the user's phone. The MCU relays digital signals directly and it converts received analog signals to digital signals using an ADC before relaying the data.
- All electric hardware must be protected from outside elements when being worn on the body. The hardware cannot be affected by the rain, snow or sweat.
- The mobile application must be able to pull data from the database.
- The UI must be easy to navigate while providing details from every sensor. The visual aids should be easy to view and manipulate.
- All transmissions to and from the cloud need to be encrypted.
- The database must be cloud-based to store and access data in real time.

1.5 Intended Users and Uses

Our team conversed with multiple medical professionals and researched the work of John Stankovic, a professor at the University of Virginia whose research focuses on sensor-based health care. The end product will be wearable and easy to use. Furthermore, it will provide information regarding heart rate, temperature, and body weight all in real time.

1.6 Assumptions and Limitations

Below is an itemized list of assumptions driving the project.

- The maximum number of users will greatly vary over any given time.
- The product is currently limited to the United States of America.
- The users will agree to an information disclosure allowing bodily readings to be stored in a remote location.
- There will be a monthly subscription charge to access and store bodily data.

- The main body sensors will monitor temperature, heart rate, and weight.
- The end product will consist of multiple wearable items, each embedded with sensors for monitoring vital functions.
- Design and build our own MCU board to receive and relay the data from the sensors.
- The MCU will relay sensor data to the user's phone which will relay the data to the cloud.

Below is an itemized list of limitations driving the project.

- The cost of the final physical product should not exceed $100.
- Due to funding and limited time, less than ten products will be available for testing.
- Majority of testing will be with simulated users and devices.
- Due to the limited amount of users, scalability will not be able to be tested as effectively.
- The system must have a power supply consisting of batteries that will power the device for at least the majority of the day.
- The system must operate on 5.5VDC or less.
- Due to the nature of the data being collected, all data transferred must be encrypted.

1.7 Expected End Product and Deliverables

This product is a wearable device that includes multiple on-body sensors. The sensors will measure temperature, heart rate, and body weight of the user. This product will be portable enough for the users to take away from home and to use during everyday events. The product is also able to connect to a mobile phone to display live data when WiFi is not available. The device will use the bluetooth functionality to connect to the mobile phone and display the live data in a clean user interface. If WiFi is available, the mobile application will be able to pull in processed data from the cloud and display the data onto the application.

2 Specifications and Analysis

2.1 Proposed Approach

Our system will utilize sensor(s) to track different vitals. Based on the readings from one or more sensors, more data will be collected or a notification will be sent to the user, medical provider, or both. To accomplish this we will have one or more sensors collecting data and send that data to a local MCU. The local MCU will then compile that data and send the data to a user's mobile device. A user will utilize our mobile application to visualize and transmit the data received from the MCU to the cloud database.

When data is received by the database it will also be sent to a cloud server where data analytics will determine if there are any abnormalities in the data. If any abnormalities are found, the server will request more data from the MCU through the mobile application. The MCU will increase readings from active sensors, request data from inactive sensors, or both. Once more data is available for analysis the server will notify the user or medical provider about the possibility of a medical concern if necessary. If the server did not identify a medical concern it will notify the MCU through the mobile application to resume prior sensor monitoring.

2.2 Design Analysis

We discussed and researched databases, sensors to be tested, and packaging designs. We decided to focus on getting one sensor fully incorporated into the system . The first sensor we chose to incorporate was the pulse sensor. The pulse sensor is extremely easy to set up since it only has three connections. We also discussed the use of an ESP board for bluetooth/wifi connections between the MCU and the sensors. The ESP boards have been easy to use since there is a large community online of people who are using them in their own projects. Moreover, we have been laying out the design for the MCU using block diagrams and have been researching PCB designs that incorporate a microcontroller to assist us in designing our MCU. For the sake of testing, we are using separate dev kits to start writing MCU code. We have decided to use AWS DynamoDB as our database and have decided to use AWS cloud services. We have also set up a test application to verify the connection between the database and the mobile application.

2.3 Development Process

We are using the Agile development process per our advisor's recommendation.

2.4 Conceptual Sketch

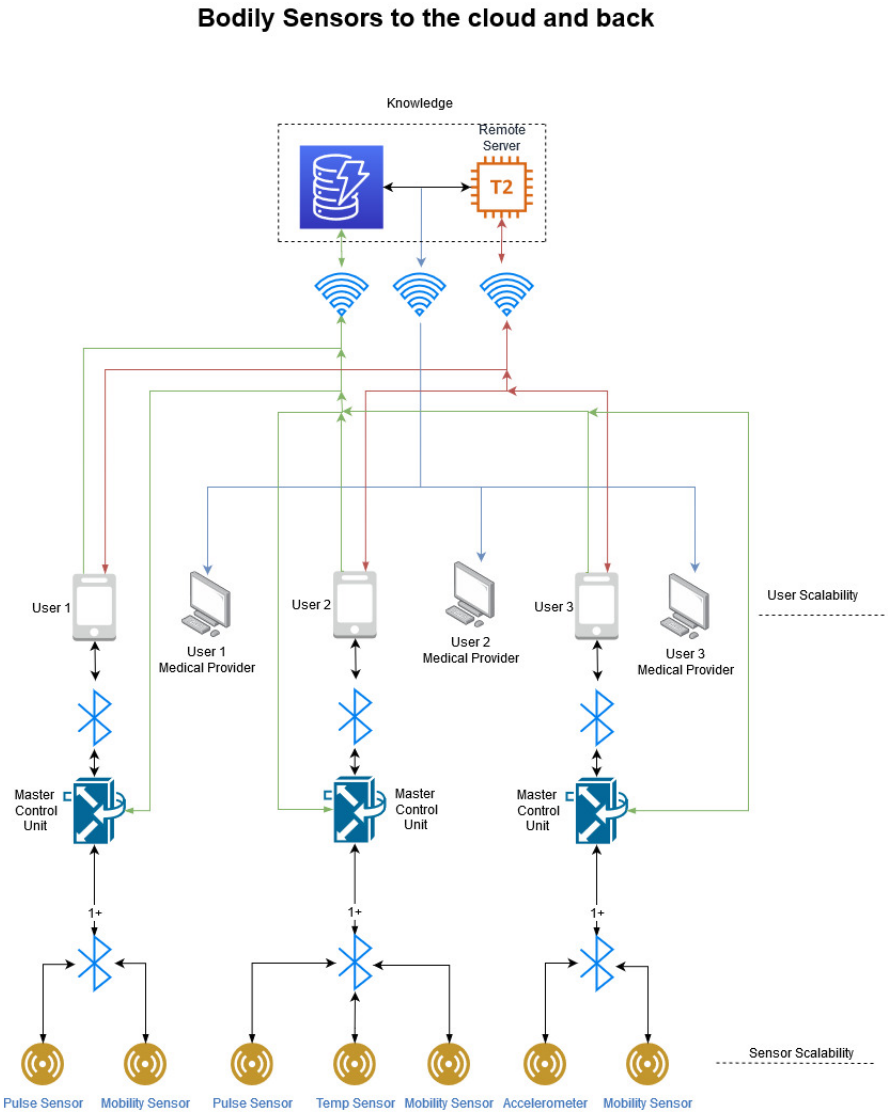

Figure 1: System Diagram

Figure 1 represents our entire system for the first three users. Users can have as many sensors as they choose, and the system is capable of adding more users.

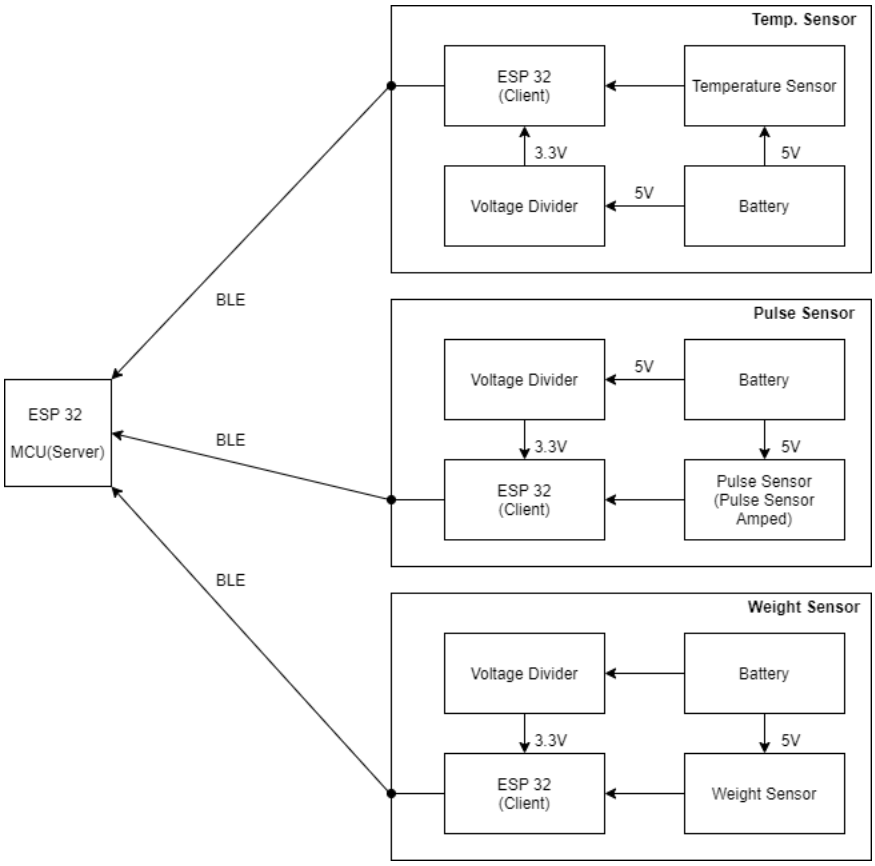

Figure 2: IoT Sensor Network

Figure 2 represents a sensor IoT network for a single user. Each user will have their own MCU (left side of the figure) which is equipped with an ESP 32 to communicate with the sensors via bluetooth. More information on the ESP 32 can be found in section 3.2. The MCU will be collecting data from the sensor to relay it to the user's mobile phone and the cloud. This user has three sensors: one to measure their body temperature, one to measure their pulse, and one to measure their weight. Each sensor has its own battery power supply as well as an ESP 32.

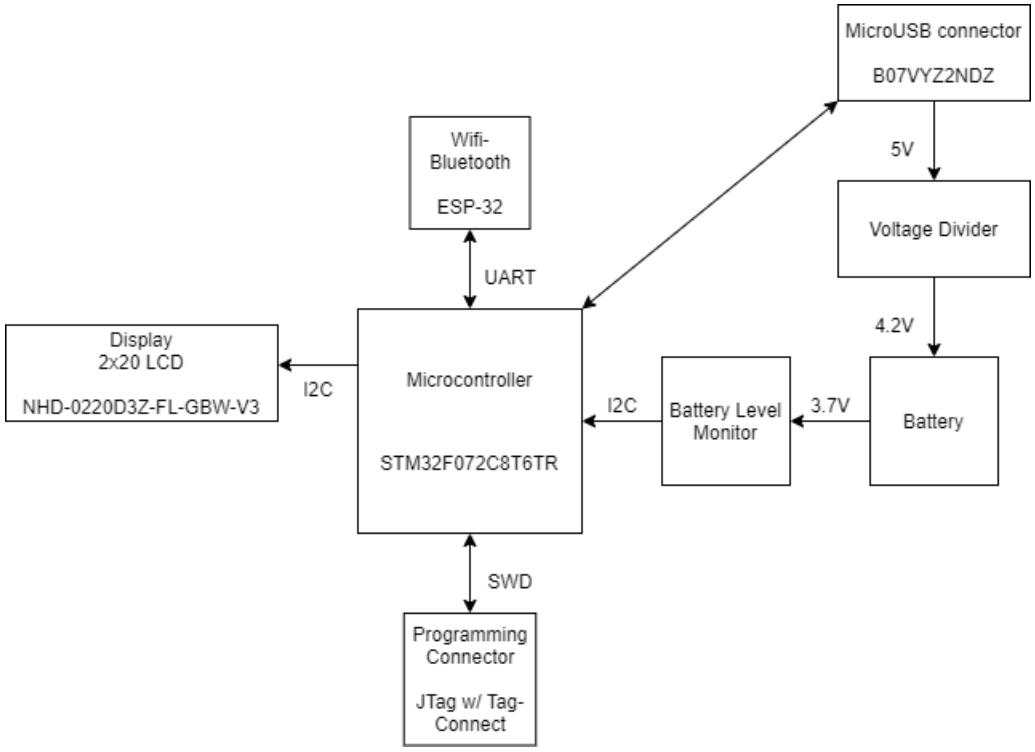

Figure 3: MCU Block Diagram

Figure 3 shows an outline of a single MCU. Each unit will be equipped with its own battery supply and an ESP 32 for wifi and bluetooth communication. In addition, it will have its own battery level monitor to determine the state of charge in the batteries, allowing the user to keep track of how much charge remains, and determine if the system needs to be recharged by plugging in external power to the micro usb connector. The display shows MCU-specific information to the end user like battery level, paired sensors, and network status. The Programming Connector is used to load code to the MCU and debug.

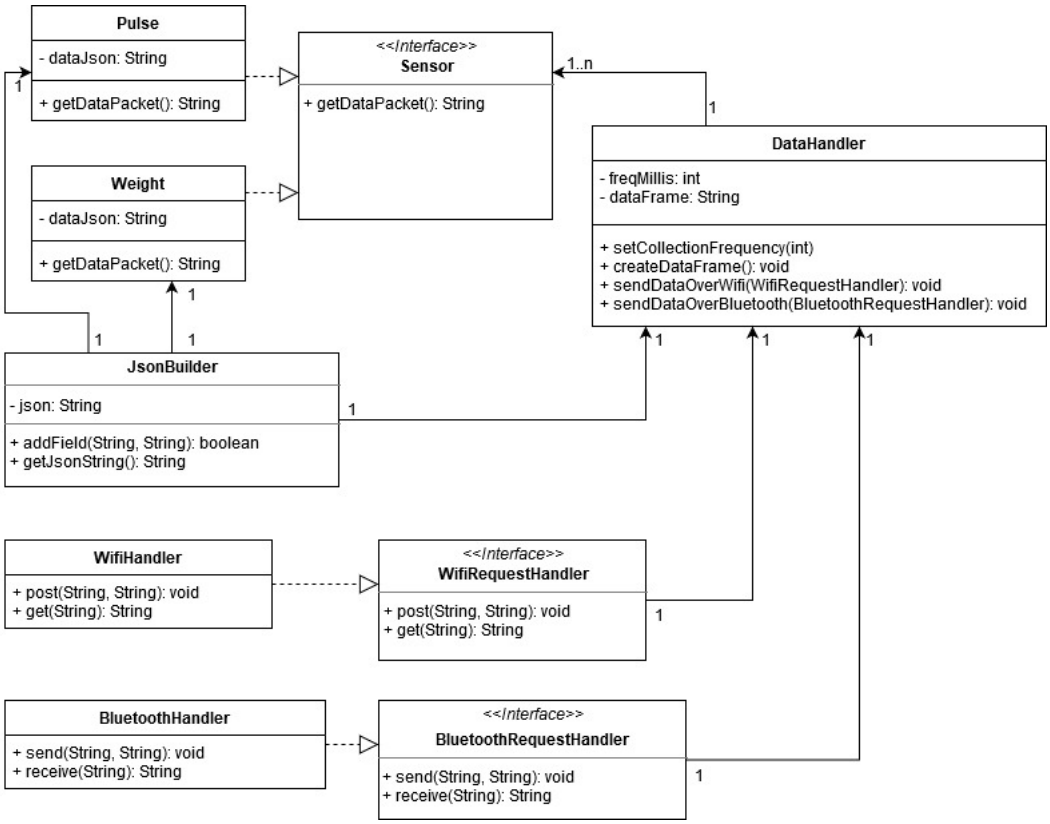

Figure 4: MCU Class Diagram

Figure 4 shows a class diagram of the MCU, illustrating how it will keep sensors scalable and how it will handle network requests for wifi and bluetooth. The sensors shown are pulse and weight.

3 Statement of Work

3.1 Previous Work and Literature

John Stankovic is a professor in Computer Science at the University of Virginia. His research consists of a smart home system for patients with various health problems such as dementia, depression, obesity, and epilepsy. Different sensors such as a trip sensor, pressure sensor, floor sensor and many other sensors are placed in a living space. These sensors will detect any irregular activity and will notify the patient's medical provider if needed[1].

Our product will be similar to Professor Stankovic's research with the exception of creating a smart home device. This product places its focus heavily on the on-body sensors and uses different sensors to detect any irregular patterns with the patient. Measurements from the sensors will be sent to the cloud, the cloud will process the

data, and send a message back to our device if any other measurement is needed. Our web application will be used by the medical provider to keep track of the patient's health.

3.2 Technology Considerations

Technology considerations are split up into seven groups: hardware/embedded, connection board options, pulse sensor options, battery options, user interface, and cloud services data analysis/databases.

3.2.1 Hardware/Embedded

To keep sensors small and provide the end user with a central hub for all sensors, we will need some kind of MCU. After consulting with ETG about options for the MCU, we narrowed our focus to 3: the Teensy board, Raspberry Pi Zero W, and a custom design [2].

The Teensy board is a small arduino board with limited GPIO. It has two analog-digital converters which can be useful for displaying waveforms, and because it is an arduino, it is comparably easy to program and test. The teensy board is also very cheap. The two largest drawbacks that make the teensy board an unviable option are that it does not have built in bluetooth or wifi. To implement both bluetooth and wifi, we would need a riser connecting the GPIO holes which makes the finished unit too bulky.

The Raspberry Pi Zero W is a very small form factor raspberry pi with built in wifi. Like the teensy board, it is quite cheap and easy to test [2]. The Pi Zero W also has a small, prebuilt battery pack, keeping to our form factor restrictions and eliminating the need to design battery technology. Unlike the teensy board, it does not have analog-digital converters, so clients will not be able to view raw waveforms and would have to rely on other tools to do so. The Pi Zero W does not have bluetooth, so, like the teensy board, we will have to use a riser to implement it. Perhaps the biggest drawback to the Pi Zero W is that it is overkill. To run any program, some version of linux needs to be installed. For our application, we do not need the HDMI port, nor the USB C ports. All we need are C libraries to run the MCU code, so using something like Linux From Scratch would be optimal.

The final option is a custom design. The greatest benefit of using a custom design is that we can implement everything exactly as needed, resulting in no extra hardware and no risers compromising form factor. Designing a custom system from scratch has its drawbacks as well. The system will be more difficult and expensive to test, and will be more time consuming to develop [3].

Given the three aforementioned options, we have chosen to build a custom design. The teensy board is too bare-bones, requiring risers to include all of the functionality we need which hurts form factor. The Pi Zero W, while small and in some ways convenient, was out of phase with what we are really looking for: not containing

some hardware we need while simultaneously containing some hardware we don't. The best option that gives us the form factor we are looking for and the functionality is a custom design.

3.2.2 Connection board options

The sensors will need a way to send the data they collect. In determining the connection options for the sensors, we spoke with ETG for advice on specific technology that will allow the sensors to relay data to and from our MCU. ETG pointed us in the direction of the ESP 8266 which is used in the roombas for the course CPRE 288 which all team members have taken. Taking a further look, the ESP 8266 enables the sensors to communicate via wifi and wifi only. It is also inexpensive, small in size, and has very low power consumption [4]. We found plenty of development kits available on the ESP website which allow us to easily test it's functionality.

When further exploring other ESP products we came across the ESP 32. The ESP 32 allows the sensors and MCU to communicate via wifi and bluetooth [5]. This alone would allow the MCU to utilize bluetooth to communicate with the sensors and relay the data to the cloud via wifi, unlike the ESP 8266. Furthermore, the ESP 32 is only slightly larger and more expensive than the 8266, however, it has stronger processing power, built-in SRAM and flash memory, and has development kits to allow us to test it's functionality on a breadboard [5]. In the end, we felt the ESP 32 was the best option because of it's dual capability of communicating via bluetooth and wifi as well as its relatively cheap price, built-in memory, and the development kits available to help us test.

3.2.3 Pulse Sensor options

The first set of sensors we explored were pulses sensors. Specifically we looked at two sensors, the MAXREFDES117 and the Pulse Sensor Amped. We chose these sensors because when we approached ETG for guidance, they recommended the Pulse Sensor Amped. After exploring many more sensors we found the MAXREFDES117 to be most appealing due to the fact that it's wearable, small in size, cheap, has low power requirements, and is provided with free algorithms for functionality with arduino microcontrollers. The downsides were that it may require 5.5 VDC for improved accuracy, and it is only compatible with arduino and mbed platforms [6].

When researching more into the pulse sensor amped, we found it to have many similar pros to the MAXREFDES sensor. It is small, wearable, and requires very little power. Furthermore, it comes with processing visualization software to help us test the sensor, it is compatible with many embedded platforms including arduino, and only needs a power, common, and data connection in the hardware design [7]. In the end, we determined the Pulse Sensor Amped to be the most viable option due its similar pros to the MAXREFDES in addition to its simplicity in design, its capability

with many platforms, and we can easily consult with ETG for any questions we have since they have experience working with the sensor as well.

For the design we have researched many types of sensors like pulse, weight, and mobility; however, to save time we chose to specifically focus on getting one sensor, the pulse sensor, functioning properly so that when we implement other sensors, the process will be much smoother and quicker.

3.2.4 Battery Options

Each individual sensor and master control unit will have its own battery power supply. We decided that it would be best to have replaceable coin cell batteries power the sensors so that when the sensor runs out of charge, there is minimal time of data loss since the user can replace the used batteries with new ones. In addition, we aim to keep the sensor circuit as simple as possible and having a rechargeable battery would require adding a charging port and battery level monitor.

For the master control unit, the user would be carrying it like how they would carry a mobile phone with them. Therefore, we decided to use a rechargeable, long lasting battery because we have more freedom in the design and we found the product would be easier to use if it could recharge like how the user would recharge their own mobile phone.

3.2.5 User Interface

Users will need to read measurements from the sensors and review processed data from the cloud. Our solution involves creating a mobile application and a web application for patients and medical providers. For the web application, we have done some research on frameworks and libraries including Angular and Vue. Angular has many advantages over Vue such as having a large support community and good server performance due to its caching and the non-requirement for extra packages to function. However, we ultimately decided to go with Vue for one reason [8]. Vue is very beginner friendly, which makes it a great choice for those who have never created a web application before. Vue has a small community, which means it does not have as much support with tools and debugging, but for this project, it should not be an issue since the scale is fairly small [9].

For our mobile application, we have two options: Android or IOS. Ideally, it would be great to have support for the two operating systems but due to time constraint, we decided to go with Android only. Both systems have extensive documentation on creating applications, but our team has much more experience with creating Android applications. Android also has a much bigger development community which makes debugging easier [10][11].

3.2.6 Cloud Services

Organizations can buy and maintain servers in a physical location. However, in the current era it is much more common to use a cloud provider to outsource these resources. This not only alleviates the burden of having to do normal maintenance, but also provides the benefits of having updated technology at no extra cost. To this end, we will use cloud services to host a remote server.

After doing research into both Microsoft's Azure [12] and Amazon's AWS(Amazon Web Services) [13] we decided to go with AWS. The reasons behind this choice were motivated by three main areas: Scalability, Cost, and Services. Our research showed that AWS offers much wider scalability when compared to Azure. While Azure might work better for smaller projects, AWS has a much greater ability to scale up with the project. For cost, we found that while Azure does offer relatively low costs, AWS's costs were even lower. AWS even offers a limited free tier that fits our teams needs for this project .When looking at services, we found that Azure is starting to invest more into different and unique services. However, AWS has a wider variety of services with easy intercommunication.

3.2.7 Data Analysis/Databases

While researching databases, we decided that Amazon DynamoDB would be the best NoSQL database [14]. This is because it gives the ability to store high quantities of data in a flexible way, and NoSQL databases tend to be faster than SQL databases.

We decided on Amazon DynamoDB because it is an excellent NoSQL database, and is serverless. DynamoDB is designed to be highly scalable with relatively low access times. DynamoDB also eases backups which are completed without performance reduction. The negative side to DynamoDB is that it can only be deployed within AWS, there is an additional storage cost for each item, and it is unable to do complex queries. We did some research on Cassandra but chose not to use it because the access times are slower, and querying does not have joins or subquery support [15]. We also did research on MongoDB and chose not to use it due to its difficulty to secure properly, and it does not have relational database capabilities.

3.3 Task Decomposition

The following table is a work breakdown summary, assigning task numbers and dependencies to tasks.

Task No.	Task Name	Contributors	Description	Dependencies

1	Hardware	John, Michael	Every hardware-based aspect of the project	
1.1	Sensor IoT Network	John	Scaleable bluetooth network of sensors	
1.1.1	Design Network	John	Create a block diagram for network architecture	
1.1.2	Order parts for network	John	Order sensors and testing dev kit	1.1.1
1.1.3	Assemble network	John	Assemble self-contained sensors	1.1.2
1.1.4	Test IoT network	John	Develop testing code to use as a template on the MCU	1.1.3
1.2	Custom MCU	Michael	Custom circuit that connects sensors to the cloud	
1.2.1	Design MCU architecture	Michael	Create a component diagram and schematic for the MCU	
1.2.2	Order components for MCU	Michael	Order components for the MCU including custom PCB	1.2.1
1.2.3	Assemble MCU components	Michael, John	Solder components to PCB	1.2.2

1.3	Integrate sensor network with custom MCU	Michael, John	Write code for MCU connecting sensor network	
1.3.1	Assemble sensors with MCU	Michael	Write MCU code	1.1, 1.2
1.3.2	Test sensor network with MCU	Michael	Refine Code	1.3.1
1.3.3	Calibrate sensors	Michael, John	Calibrate sensors so they generate accurate measurements	1.3.2
2	Database/Cloud	Richa, Justin	Cloud based aspects of the project	
2.1	AWS	Justin	Amazon Web Services	
2.1.1	Set up AWS server	Justin	Set up an AWS server	
2.1.2	Set up AWS DynamoDB	Richa	Set up AWS DynamoDB database	
2.2	Data Analytics	Richa	Develop AI code to interpret data	2.1
3	User Interface	Isaac, Richa, Michael, Justin	All UI aspects of the project	
3.1	Mobile Application	Isaac	Every mobile-based aspect of the project	
3.1.1	Integrate mobile application with database	Isaac, Richa	Connect mobile app to the AWS database	2.1.2
3.1.2	Integrate MCU with mobile application	Michael, Isaac	Connect mobile app to the MCU	1

3.2	Web Application	Justin, Isaac	Every web-based aspect of the project	2.1.2
3.2.1	Integrate web application with database	Isaac, Justin, Richa	Connect web app to the AWS database	2.1.2
3.3	Test Integration	Isaac, Justin	Test data flow from MCU to database	3.1, 3.2
4	Testing	Team	Test system as a whole, looking for issues in functionality and design	1,2,3

Table 1: Task Decomposition

Table 1 shows how the tasks for our project will be decomposed. Each task is assigned to a team member(s) who will be working on it. The dependencies of each task are listed in the far right column.

3.4 Possible Risks and Risk Management

Risk	**Severity**	**Mitigation Strategy**
High Expenses	High	Use of existing alert system(s) within AWS for notification when a service is reaching a repset usage threshold.
Lack of experience/knowledge	High	When faced with area's within the project where we were lacking experience, we consulted with experts in that area and would do extensive research into that area.
Injury from misuse	Medium	To minimize the injury from misuse, we took extra steps within the design to address possible causes of injury.

Malfunctioning system from normal usage over time	Low	We designed the system to be able to handle different environments and included tests for reliability into our design.
Data security	High	Planning is in progress and will be able to better address the risk next semester.

Table 2: Risk Mitigation

Table 2 shows the risks that are involved with creating and using our product. Each risk has its own severity and our strategy for dealing with each risk is listed under Mitigation Strategy.

3.5 Project Proposed Milestones and Evaluation Criteria

One key milestone is to have a fully finished design for our product by the end of the semester. In addition, we would like to have our sensor researched, ordered, and tested. We would also like to have our mobile application operating by early next fall. More milestones are discussed in the gantt chart.

3.6 Project Tracking Procedures

We will be using Trello as a method for tracking progression on each task that needs to be accomplished in order to create the final product. This method of SCRUM also documents the progression with time. We will also be using GitLab to keep track of changes in our codes.

3.7 Expected Results and Validation

Our desired outcome is to have a product that is accurate and secure. The final product needs to be able to gain users' trust the moment it is being presented. We can measure temperature, heart rate, movement, and body weight using external tools and compare the results to the ones we measured using our product.

4 Project Timeline, Estimated Resources, and Challenges

For our timeline we have split the team into software and hardware. Software is split into three sub-groups: user interface, server, and data.

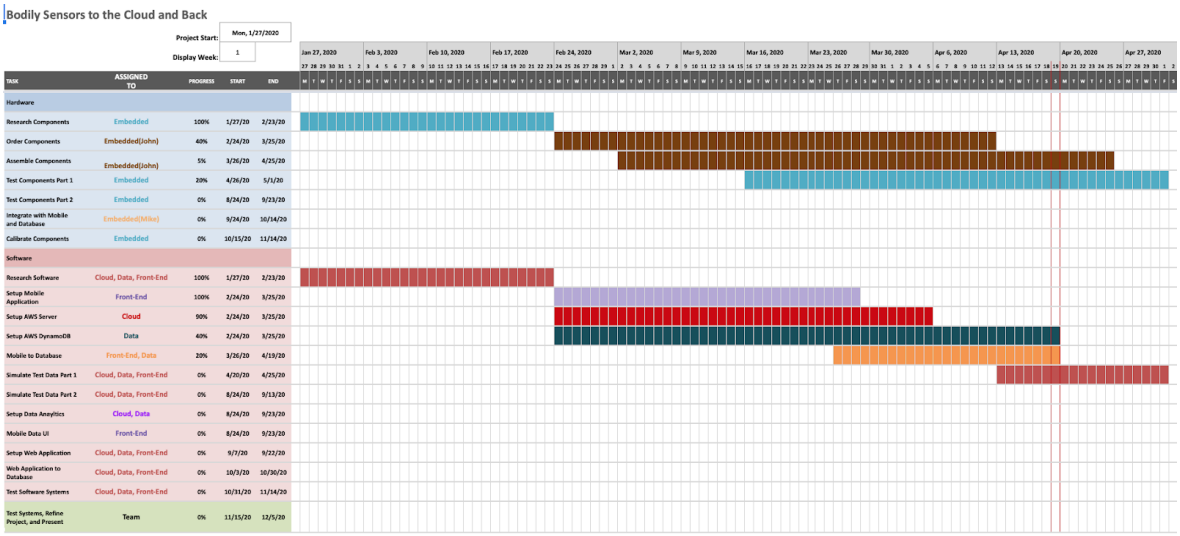

Figure 5: First Semester Gantt Chart

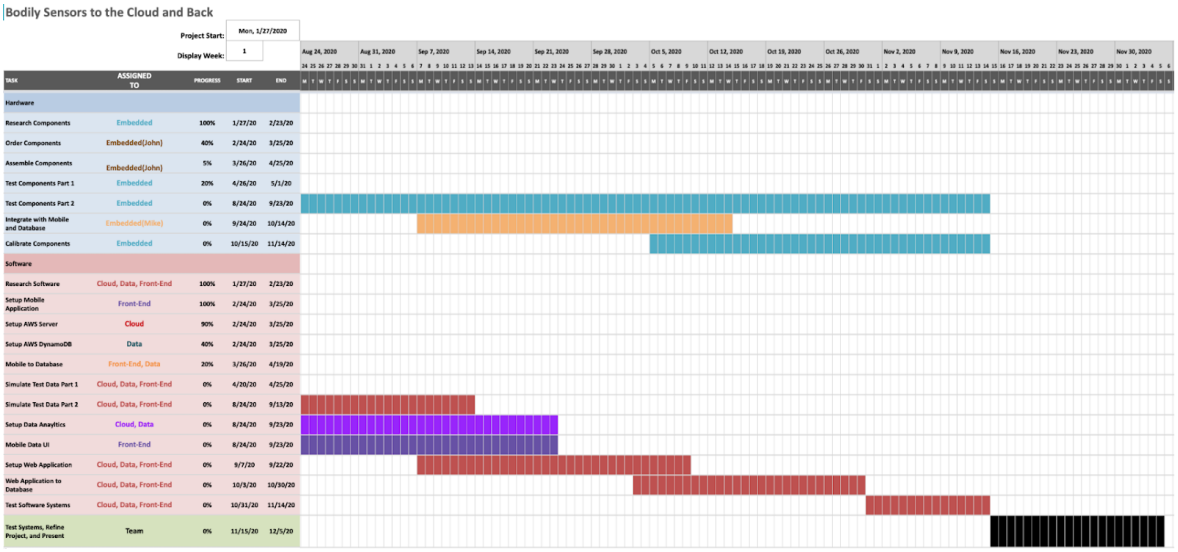

Figure 6: Second Semester Gantt Chart

4.1 Project Timeline

For our timeline we have split the team into software and hardware. Software is split into three sub-groups: user interface, server, and data.

4.2 Feasibility Assessment

Designing a microcontroller requires a lot of troubleshooting and debugging which can lead to unforeseen delays. Our approach to work around this is to test all sensor

functionality with a prebuilt MCU (an ESP 32 development kit) to ensure the IoT sensor network is functioning properly when we have to test our custom MCU.

4.3 Personnel Effort Requirements

Task	Estimated Time per Week
Cloud application	5 hours
Hardware application	9 hours
Database application	3 hours
Web application	3 hours
Mobile application	6 hours
Testing	25 hours

Table 3: Personnel Effort Requirements

4.4 Financial Requirements and Other Resource Requirement

For this project, we are assuming a budget of $500. The table below shows project costs.

Project Task/Tools	Quantity	Unit Cost	Total Est. Cost	% of Budget
ESP32 dev kits	4	$10.99	$43.96	8.792%
Pulse Sensor	1	$24.95	$24.95	4.99%
Weight Sensor	1	$9.00	$9.00	1.8%
Tag Connect Cable	1	$39.99	$39.99	7.998%
STM32 Microcontroller	2	$3.49	$6.98	1.396%
Custom PCB	2	$10	$20	4%
Micro-B USB connector	2	$0.87	$1.74	0.348%

100 nF caps	10	$0.163	$1.63	0.326%
10 nF caps	10	$0.05	$0.50	0.1%
4.7 uF caps	10	$0.206	$2.06	0.412%
AWS subscription[1]	9 months	$0-$30/month	$0-$270	0-54%
Total			**$190.88 - $460.88**	**38.176% - 92.176%**

Table 4: Financial Breakdown

We are spending, at most, 92.176% of the budget which leaves us $39.12 to continue adding sensors or eat up unforeseen costs. Because we are not valuing our labor, the surplus in budget would likely go to fund extra hardware in the event that original hardware is broken or becomes unusable.

5 Testing and Implementation

5.1 Interface Specifications

- Functionality between sensors and MCU (John)
- Functionality between MCU and mobile application (Michael & Isaac)
- Connectivity between mobile and database (Isaac & Richa)
- Connectivity between MCU and database (Michael & Richa)
- Functionality between database and cloud (Richa & Justin)
- Functionality between cloud and web application (Justin & Isaac)

5.2 Hardware and software

The following is an itemized list of software tools used in the project.

- Jest
- Mocha

[1]AWS subscription ranges from $0-$30 per month because of the free trial tier. For the scope of this project, we are trying to keep in the free tier. The $30 per month is representative of a small scale deployment with multiple users.

- JUnit
- Postman
- KiCad

Jest and Mocha are libraries that are used to test JavaScript codes. These libraries can be used to test the functionalities of Vue. JUnit is used to test Java applications. This unit testing tool will be used to test the functionalities of the mobile application. Postman is a platform for API development. It will be used to test the backend server. KiCad is a CAD design tool for printed circuit boards(PCBs). This tool will be used to design the IoT circuits as well as the MCU PCB.

5.3 Functional Testing

- Test sensitivity of the sensors using an oscilloscope. Make sure the sensitivity is not too high where it inhibits data collection and analysis. Adjust filtering or sampling frequency to clean up noisy signals. (John & Michael)
- Test ohmmeter circuit with resistive sensors to make sure the data collection is consistent and contains minimal noise. (John)
- Test bluetooth low energy mesh connections between sensors and MCU using serial messaging applications on our own cell phones and then testing. (John)
- Test to see if the mobile and web applications pull correct data from the database (Isaac & Richa)
- Test if measured data can be displayed on the mobile application live (Isaac)
- Test if the mobile application can push data to the database (Isaac & Richa)
- Test if new users can be added to the database (Isaac & Richa)
- Test if data is displayed to the correct user (Isaac & Justin)

5.4 Non-Functional Testing

- Test wear-ability and sensor connection distance to make sure the sensors can connect within at least a 10 foot range but not further than 15 feet. (Entire team)
- Test how long it takes to pull data and display it onto the frontend (Entire team)
- Test how fast users can login. The process should take no more than 30 seconds. (Entire team)

- Test the responsiveness of the web application. Visual lagging should be minimal. (Isaac, Richa, & Justin)
- Pen-testing for user data (Isaac & Justin)

5.5 Results

Testing this semester has mostly involved the hardware and embedded systems. Due to the COVID-19 pandemic causing the temporary closure of all labs on the Iowa State University campus, we had a limited window to utilize the labs and the equipment provided on campus. During our available time on campus, we tested the Amped pulse sensor and found promising results. The sensor sends an analog signal and we were able to observe our own heartbeats on the oscilloscope at an operating voltage of 5V and 3.3V DC. The one challenge we face in the future is filtering the noise in the signal. When we measured our pulse and moved our hand simultaneously, the signal experienced a lot of distortion so in the future we will have to either find a way to filter the signal using hardware or software. Furthermore, we were able to test our ESP 32 chip. We tested the chips bluetooth capabilities and we were able to send a serial message from the ESP 32 to our mobile phones. The next step is to establish an ESP 32 chip as an MCU and use the remaining chips and sensors to test a BLE mesh network and send data from a sensor to a mobile phone.

6 Conclusion

So far, we have started designing the MCU board, and sensors network. Our end goal is to interface the MCU with our sensors in an IoT environment and have the MCU send data from the sensors to the cloud and a mobile application where the user can then view the data and analytics from a mobile or web application.

To accomplish our goals in the time frame we have, our development groups need to work concurrently. While the hardware team designs the MCU, the cloud team will be setting up infrastructure with AWS and implementing stubs to ensure communication between the cloud and the web application is running smoothly. Developing in parallel reduces the time cost to developing this project and gives us more time for testing.

Our end product will allow medical professionals to analyze many vital functions of multiple patients at once with ease using our sensor network design. The information will be stored and interpreted using the cloud services we set up with our mobile and web applications. This will make the lives of medical professionals much less stressful when monitoring their patients. In addition, our product will give more patients the freedom to leave a medical center or hospital while receiving care if they wish to do so, and if they are given permission by their doctors, since our product is completely portable.

References

[1] "John A. Stankovic," *University of Virginia School of Engineering and Applied Science*, 06-Dec-2019. [Online]. Available: `https://engineering.virginia.edu/faculty/john-stankovic`. [Accessed: 25-Apr-2020].

[2] Sam Burnett - ETG, personal communication, Feb. 2020

[3] Teel, John, "How to Design Your Own Custom Microcontroller Board", *Predictable Designs*, 15-May-2018. [Online]. Available: `https://predictabledesigns.com/tutorial-how-to-design-your-own-custom-microcontroller-board-video-part1/`. [Accessed: 12-Feb-2020].

[4] Espressif Systems "ESP8266EX" ESP8266EX datasheet, Dec. 2015 [Revised Apr. 2020].

[5] Espressif Systems, "ESP32 Series," ESP32 datasheet, Aug. 2016 [Revised Jan. 2020].

[6] Maxim Integrated, "MAXREFDES117#: Heart-Rate and Pulse-Oximetry Monitor," System Board 6300 datasheet, Aug. 2016.

[7] Sparkfun Electronics, "Pulse Sensor," *Sparkfun Electronics*, SEN-11574. [Online]. Available: `https://www.sparkfun.com/products/11574`. [Accessed: Feb. 5, 2020].

[8] AngularJS. [Online]. Available: `https://docs.angularjs.org/guide/concepts`. [Accessed: 23-Apr-2020].

[9] "Overview," *Overview - vue.js*. [Online]. Available: `https://v1.vuejs.org/guide/overview.html`. [Accessed: 19-Apr-2020].

[10] "Meet Android Studio: Android Developers," *Android Developers*. [Online]. Available: `https://developer.android.com/studio/intro`. [Accessed: 19-Apr-2020].

[11] "Start Developing iOS Apps (Swift): Jump Right In", 08-Dec-2016. [Online]. Available: `https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/`. [Accessed: 23-Apr-2020].

[12] "Cloud Computing Services: Microsoft Azure," *Cloud Computing Services | Microsoft Azure*. [Online]. Available: `https://azure.microsoft.com/en-us/`. [Accessed: 19-Apr-2020].

[13] "Cloud Computing Services: Amazon Web Services," *Amazon Web Services (AWS) - Cloud Computing Services.*[Online]. Available: https://aws.amazon.com/. [Accessed: 19-Apr-2020].

[14] Amazon Web Services, Inc 2020. *Amazon Dynamodb - Overview.* [online] Available at: https://aws.amazon.com/dynamodb/ [Accessed 19 April 2020].

[15] Kumar, P., 2020. *Amazon Dynamodb Tutorial - A Complete Guide | Edureka Blog.* [online] Edureka. Available at: https://www.edureka.co/blog/amazon-dynamodb-tutorial#What-Is-DynamoDB? [Accessed 19 April 2020].

Appendix

Market Survey

John A. Stankovic - BP America Professor at the University of Virginia
Sarah A Radke - Nurse Practitioner Specialist in Whitefish Bay, WI
Jessica Terrell - Registered Nurse at Life Care Center of Jacksonville

MCU Programming References

The GNOME Project, *GTK+ 3*, (2020). Github Repository: https://github.com/GNOME/gtk. [Accessed: 09-Apr-2020].

The GNOME Project, *Glib*, (2020). Github Repository: https://github.com/GNOME/glib. [Accessed: 09-Apr-2020].

Boyini, Karthikeya, "Enum in C," *tutorialspoint.com*, 05-Nov-2018. [Online]. Available: https://www.tutorialspoint.com/enum-in-c. [Accessed: 09-Apr-2020].

Carnegie Mellon University, "C Coding Standards," *Carnegie Mellon University*, 27-Apr-2004. [Online]. Available: https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html. [Accessed: 16-Apr-2020].