A Cloud Infrastructure for Large Scale Health Monitoring in Older Adult Care Facilities

Uchechukwu Gabriel David

University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/masters_theses_2

Part of the Computer Engineering Commons, Electrical and Computer Engineering Commons, and the Nursing Commons

Recommended Citation
https://doi.org/10.7275/24314898.0 https://scholarworks.umass.edu/masters_theses_2/1148

This Open Access Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.
A CLOUD INFRASTRUCTURE FOR LARGE SCALE HEALTH MONITORING IN OLDER ADULT CARE FACILITIES

A Thesis Presented
by
UCHECHUKWU DAVID

Submitted to the Graduate School of the University of Massachusetts Amherst in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

SEPTEMBER 2021

Electrical And Computer Engineering
Acknowledgments

A Big Thank You, to my family and my supervisor (Prof. Noh), who helped and supported me throughout the course of my Master’s program.
Technology development in the sub-field of older adult care has always been on the back-burner compared to other healthcare areas. But with increasing life expectancy, this is poised to change. With the increasing older adult population, the current older adult care facilities and personnel are struggling to keep up with demand. Research conducted in the Netherlands [1] found 33,000 older adults were awaiting admission into a home for the elderly showing that demand far exceeds availability. This huge demand for older adult care has resulted in a decrease in the quality of care being provided. A recent study involving older adults aged 65 and above [2] compared the quality of care given to older adults in nursing homes in the UK and found it to be inadequate. While it is true that giant strides have been made in the field of personal health and fitness [3], we have to acknowledge that these technologies have not found widespread adoption in the elderly communities for a number of reasons which include lack of education, cognitive impediments, low-income and techno-phobia [4]. We believe that older adult care technologies should be approached from a different perspective in order to maximize outcomes. Inventions in the health care space are a moving target and a significant degree of technical aptitude and interest is required to keep up with these changes. My research work will be focused on developing a distributed system
infrastructure that will enable large-scale monitoring of vital signals and early detection of emergency situations in nursing homes and assisted living communities. This new approach will increase automation in nursing homes leading to a reduction in running cost and an increase in capacity.
Contents

Acknowledgments .......................................................... i

Abstract ........................................................................ ii

List of Figures ............................................................... vi

1 Introduction .................................................................. 1
  1.1 Overview of the Current State of Smart Healthcare ................. 2
    1.1.1 Smart Watches and Fitness Bracelets .......................... 3
    1.1.2 Remote Patient Monitoring ...................................... 3
    1.1.3 Google Nest / Amazon Echo (Smart AI Gateways) ........... 3
  1.2 Brief Description of Our Proposed System and Its Advantages Over Existing Solution .................................................. 3

2 Methodology .............................................................. 6
  2.1 System Architecture Design .......................................... 6
  2.2 Overview of Tools and Technologies ................................. 10
    2.2.1 BLE (Bluetooth Low Energy) ................................... 10
    2.2.2 Django (Backend web development framework) ............. 10
    2.2.3 Reactjs (Frontend javascript library) ......................... 10
    2.2.4 Nodejs (Javascript Backend) ................................. 11
    2.2.5 Socket.io (Realtime communication library) ............... 11
  2.3 Node device (Middle-ware) Implementation ........................ 11
  2.4 Backend server implementation ...................................... 12
  2.5 Front-end implementation ............................................ 15
  2.6 DevOps (Development and Operations) .............................. 17
    2.6.1 Hardware ......................................................... 19
  2.7 Evaluation Setup ...................................................... 20
    2.7.1 Evaluation Metrics ............................................. 21

3 Results .................................................................... 22
  3.1 Experiment 1: Scalability ............................................ 22
  3.2 Experiment 2: Real-Time Operation ................................. 22
## CONTENTS

3.3 Experiment 3: Cost Analysis .................................................. 24  
3.4 Experiment 4: Usability ......................................................... 26  

4 Discussion .................................................................................................. 28  
4.1 Comparing With Existing Solutions ................................................. 29  

5 Conclusion ............................................................................................... 32  

A An appendix .............................................................................................. 33  

Bibliography ................................................................................................. 34
# List of Figures

1.1 RPM Architecture Design .............................................. 4
2.1 General System Design .................................................. 7
2.2 Cloud Infrastructure Design ............................................ 8
2.3 Pseudo-code for Node Firmware ....................................... 12
2.4 Database table for User .................................................. 12
2.5 Database table for Sensor Data ......................................... 13
2.6 Code snippet for Data Broadcast System ............................. 15
2.7 Image showing Front-end Application ................................. 16
2.8 Sub-components of User UI Object .................................... 17
2.9 User Object on-click ...................................................... 18
2.10 Docker Compose Configuration ........................................ 20
3.1 Scalable docker infrastructure ......................................... 23
3.2 Ping Response Time Vs Number of Users ............................ 23
3.3 Data transit time Vs Number of Users ................................. 24
3.4 Frontend Application ..................................................... 26
Chapter 1

Introduction

Government healthcare expenditure for older adults over the age of 65 accounts for about 30 percent of the total healthcare expenditure in the USA [17]. As the population continues to age, this fraction will likely grow. Another factor that contributes to the high cost of healthcare for older adults is the increase in chronic diseases. Older adults experience a higher risk of chronic diseases and will need more care. As more people begin to need regular assistance for their daily living, the existing healthcare services are not able to fulfill the demand. Different approaches are currently being explored to solve this problem, these include community living, virtual care and tele-health. Many nursing homes are now adopting a franchise model to increase supply and bring care closer to the seniors [18]. Community living and assisted in-home living have gained popularity in recent years because of the increasing number of older adults who prefer to live at home for as long as possible. But even while living at home, it is necessary to monitor high-risk older adults and notify caregivers when emergency situations are detected. Nursing homes remain the optimal solution for older adults who have health problems and as a result are not able to independently perform activities of daily living (ADL). Nursing homes are anything but cheap. On average, a private room in a nursing home costs $250 daily [17] [29]. This sums up to a little over $100,000 annually as shown in Table 1.1 [11]. Government aid (Medicare) and Long-term health insurance cover this cost for some individuals, but in most cases, this aid only kicks in after personal resources have been exhausted. This is unaffordable for the majority of people. The high cost of senior care can be attributed to the high number of highly trained personnel required to provide care and medical services.

In an effort to minimize cost, most older adult care facilities are understaffed. This makes it challenging to detect emergency health situations when they occur. Currently, care providers rely on the older adults to make a call when they are not feeling well. This works in some cases, but many emergency health situations may not be immediately obvious to the older adult, example: high/low blood pressure, oxygen saturation, irregular heart rhythms etc. This solution is also not suitable for older adults with reduced mobility and or cognitive impediments. The distributed monitoring system we are proposing will address the problems we have discussed above. Using a system of bio-sensors and wireless data transmitters, we
can collect bio-signals from each older adult and alert caregivers when vital signs go below or above a certain threshold. Using a system like this, caregivers will always be aware of the current condition of all the older adults in a facility and will be immediately notified when emergency conditions are detected.

A lot of the tasks performed by skilled and unskilled nursing staff in a care facility cannot be automated (e.g., walking, feeding, toileting, grooming, and general hygiene, etc.), but some tasks like routine ward rounds and vital signal monitoring are automatable. By automating vital signal monitoring, this system will allow care providers to focus on other un-automatable tasks. This will increase the quality of care provided as caregivers can now devote more time to each older adult.

The key components in our proposed system will be a network of sensors for data collection. Each older adult in the care facility will have a smart wearable device optimized for high-quality bio-signal collection. This bio-signal data will be wirelessly sent to a central control station for processing and storage. A real-time interface will also be provided to enable caregivers and nurses to view live patient bio-signals and take necessary actions in the case of an emergency.

To ensure that the system meets requirements, we evaluated out system using metrics like usability, scalability, real-time operation, and cost.

<table>
<thead>
<tr>
<th>State</th>
<th>Semi-Private</th>
<th>Private</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alaska</td>
<td>$26,403</td>
<td>$27,563</td>
</tr>
<tr>
<td>Connecticut</td>
<td>$12,623</td>
<td>$13,733</td>
</tr>
<tr>
<td>Hawaii</td>
<td>$12,167</td>
<td>$13,657</td>
</tr>
<tr>
<td>Massachusetts</td>
<td>$12,015</td>
<td>$12,755</td>
</tr>
<tr>
<td>New York</td>
<td>$11,756</td>
<td>$12,189</td>
</tr>
<tr>
<td>North Dakota</td>
<td>$11,027</td>
<td>$11,690</td>
</tr>
<tr>
<td>New Jersey</td>
<td>$10,696</td>
<td>$11,863</td>
</tr>
<tr>
<td>Delaware</td>
<td>$10,694</td>
<td>$10,950</td>
</tr>
<tr>
<td>New Hampshire</td>
<td>$10,342</td>
<td>$11,102</td>
</tr>
<tr>
<td>West Virginia</td>
<td>$10,281</td>
<td>$11,072</td>
</tr>
</tbody>
</table>

Table 1.1: Average monthly cost of a room in a nursing home [7][8].

1.1 Overview of the Current State of Smart Healthcare

There exists a plethora of smart home health solutions for both older adults and the general population. We will discuss some of the leading solutions and their shortcomings with regards to application in terms of the sub-field of senior care.
1.1.1 Smart Watches and Fitness Bracelets

Smartwatches and fitness bracelets are among the most significant innovations in recent times. They are very popular because of the compression of some of the important features found in mobile phones and computers into a smaller wearable form factor. They also incorporate a host of bio-sensors, which are used to collect and analyze bio-signals. Healthcare applications of smartwatches include [22]: activity monitoring, gesture recognition, heart rate monitoring, diabetes self-management, detection of seizures, etc. The signal quality from miniaturized wearable systems was initially not good enough for clinical use but advances in nanotechnology and biomedical engineering have led to wearable bio-sensors that are comparable to clinical-grade solutions. Examples include: Omron Heart-Guide [5], DexCom continuous glucose monitoring [19], AliveCor wireless ECG monitoring [10].

1.1.2 Remote Patient Monitoring

Remote patient monitoring is one of the big breakthroughs in older adult care in recent times. Remote patient monitoring (RPM), also known as telemedicine, uses digital technologies to collect medical and other forms of health data from individuals in one location and electronically transmit that information securely to health care providers in a different location for assessment and recommendations [10]. RPM greatly reduces the cost of treatment by replacing in-person visits with remote doctor-patient sessions. RPM is still a developing technology. While some research studies [1] have reported a noticeable impact of remote health monitoring on clinical outcomes, others [14] have found that RPM did not reduce readmission rates.

1.1.3 Google Nest / Amazon Echo (Smart AI Gateways)

Google Nest and Amazon Echo are both revolutionary products that have found use in the health care space. They provide an array of features ranging from smart AI assistants to the Internet of Things (IoT) Hub and multimedia entertainment devices. They are able to connect to smartwatches, fitness trackers, etc., and pull data from them. They are especially useful in helping users understand and take control of their data [24]. Amazon Echo/Alexa is also finding use in hospital room management [3].

1.2 Brief Description of Our Proposed System and Its Advantages Over Existing Solution

Our system borrows from the existing healthcare technologies and goes a step further, enabling scalability and large-scale use in assisted living facilities and nursing homes. We focused on building an end-to-end infrastructure that allows multiple features and sensor types to be easily integrated. Our system enables multiple older adults in a nursing home
or one older adult living independently to be monitored with reduced human intervention. It is comprised of bio-sensors, edge devices, and cloud computing resources.

Smart watches and fitness bracelets are well suited for personal use but are not optimized for use in community scenarios like nursing homes. Outstanding research by Frederic et al. [5] explains some of the challenges that hinder the widespread adoption of smartwatches among older adults. One of the major reasons is usability [19]. Most of the technology and interfaces in these smartwatches do not come naturally to older adults. The instructions and user manuals for these devices are also not targeted towards older adults. Our system will attempt to solve this usability issue by offloading most device management and configuration to the nurses and caregivers. The device in this case will require little to no input or setup configuration from the older adults.

Our proposed system shares some similarities with remote patient monitoring. Both systems leverage the additional computational power available in the cloud infrastructure to deliver healthcare services. RPM uses secure cloud servers [6] to establish communication and data transmission links between patients and their care providers. Our system will also use cloud servers to perform processing and storage of collected data allowing the wearable and sensor devices to consume less power. The major difference between the two systems is in the area of application. While RPM is designed for temporary monitoring of discharged (high risk) patients, our system is designed for long term monitoring of bio-signals in older adults. RPM is also meant to be a personalized solution. As a result, it does not take ad-
Smart AI Gateways (Google Nest and Amazon Echo) are designed to be universal problems solvers, we will compare and contrast these universal devices to our specialized eldercare devices.

- **Scalability:** Smart AI gateways (Google Nest/Amazon Echo) are designed for personal use. The high emphasis on privacy and data protection makes it difficult for third-party companies to build on top of this platform. To implement Amazon Echo in an older adult care facility you will need to procure and configure an individual device for each room/user. In contrast, our system will be designed for community environments. This will include a central configuration system that will be used to configure and manage multiple devices.

- **Power Consumption:** Smart AI gateway devices incorporate an extensive suite of features which include: AI assistant, calendar, smart home automation, Multimedia apps, etc. While these extra features are desirable, they increase power consumption resulting in decreased battery life. A more specialized solution will minimize power consumption by transitioning to a low-power mode during down periods.

- **Complexity:** Setting up Smart AI gateways is a non-trivial task even for a technology enthusiast. This presents an ease-of-use problem for older adults and additional setup and installation costs for nursing home operators. By offloading all configuration and setup functions to the care providers, our system will present a less steep learning curve for older adults.
Chapter 2

Methodology

We designed a system that borrows from the existing healthcare technologies and goes a step further, enabling scalability and large-scale use in assisted living facilities and nursing homes. We focused on building an end-to-end infrastructure that will allow multiple features and sensor types to be easily integrated. Our system enables multiple older adults in a nursing home or one older adult living independently to be monitored with reduced human intervention. It is comprised of bio-sensors, environmental sensors, edge devices, and cloud computing resources.

- Vital Signal Monitoring Device: This includes the vital signal monitoring systems and sensors. Our system is designed to be modular and accept multiple bio-signal devices. Some of the most common bio-signals include: ECG, heart rate, body temperature, and bio-impedance. These sensors will be controlled by a low-power wireless-enabled microcontroller which will also transmit sensor data to a node device.

- Node Devices: These devices serve as a middle-man between the low-power microcontrollers that control the sensors and the cloud. They will aggregate data from multiple sensor controllers and forward this data to the cloud. Node devices are designed to serve multiple users at a time. This will reduce the cost of large-scale applications. This is an advantage our system has over existing solutions.

- Cloud Devices: The cloud server will process and store all the data received from the node devices. It will also provide a real-time interface where nurses and private care providers can monitor multiple sensors in real-time.

My work focused on building the web infrastructure (Node device and Cloud software) to handle reception, transmission, storage and presentation of data.

2.1 System Architecture Design

A good system architecture should have some if not all of the following qualities:
• Understand-ability.
• Usability.
• Securability.
• Reliability and Availability.
• Testability.
• Scalability.
• Fault Tolerance

The system we developed is comprised of the Node devices and Cloud infrastructure. The node device collects data from multiple users in the same locality and forwards this data to the cloud service. Communication between the sensor devices on the users and the node device is accomplished using Bluetooth Low-Energy (discussed below). Some of the reasons why we chose this wireless technology over others like WiFi, Zigbee include: low power consumption, low unit cost and widespread adoption. The Node device is implemented on a Raspberry Pi Model 3b+ which has built-in BLE capability. The software which runs on the Node device is written using the Python programming language.

Python is an interpreted high-level general-purpose programming language. Python’s design philosophy emphasizes code readability with its notable use of significant indentation. It is one of the most widely used programming languages and as a result, software libraries and hardware drivers are readily available.
The Node device software receives data packets from the user sensor devices and forwards this data to the cloud infrastructure. The design of the cloud infrastructure represents the majority of the work accomplished in this project. We designed the cloud infrastructure to have very low-latency while supporting data-reception from node-device in different geographical locations. Our cloud platform is also able to stream data to multiple viewers (health officials and care-providers with authorized access).

Our cloud software is composed of the following parts:

- **Database**, stores user and sensor data.
- **REST API**, Consumed by node devices and connected web browsers.
- **Data Broadcast System**, broadcasts incoming data packets to subscribed web browsers.

![Pictorial representation of cloud infrastructure](image)

**Figure 2.2: Pictorial representation of cloud infrastructure**

**Database: PostgreSQL**

A database is a software program developed to create, edit, store and search files and records. There are so many types of databases available for software projects. The choice of the
database will depend on the system requirements. Examples of system requirements include: workload, latency requirements, available Hardware e.t.c.

For our application, we have chosen a general purpose Database, PostgreSQL, which works well is most use cases. PostgreSQL [27] is a powerful, open source object-relational database system with a strong reputation for reliability, feature robustness and performance. A relational database is a collection of data items with pre-defined relationships between them. These items are organized as a set of tables with columns and rows. Tables are used to hold information about the objects to be represented in the database. Each column in a table holds a certain kind of data and a field stores the actual value of an attribute. This relationship feature is particularly important in our application because we need to establish a relationship between each user and multiple sensor data types that might be available.

**REST API: Django Rest Framework**

An API is a set of definitions and protocols for building and integrating application software. It’s sometimes referred to as a contract between an information provider and an information user—establishing the content required from the consumer (the call) and the content required by the producer (the response) [21]. A REST API is an application programming interface that conforms to the constraints of REST architectural style and allows for interaction with web services. A RESTful API allows us to perform the following commands:

- **GET**, Retrieve a resource. This is used by connecting web apps (browsers) to retrieve user data from the database.

- **PUT**, To change the state of or update a resource. This is used by the Node devices, to updated or add sensor data to the database.

- **DELETE**, To remove an object or resource. This is used by the web apps to delete user or sensor data.

To facilitate ease of development and conform to the REST standard, we used a well-known REST Framework, Django Rest Framework.

**Data Broadcast System: Socket.io**

When multiple web browsers (care-providers) connect to the cloud application, we need a way to reliably broadcast sensor data to all of them. This is what the data broadcast system is responsible for. To avoid a system where multiple endpoints are constantly polling the cloud server for new data, we developed this data broadcast system to be event-based. I.e web browsers do not need to constantly check for new data, the register themselves with the data broadcast system and this system notifies them when new data is available.

This architecture greatly reduces the number of requests which frees up computation time on the cloud hardware.
2.2 Overview of Tools and Technologies

2.2.1 BLE (Bluetooth Low Energy)
Bluetooth Low Energy is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group (Bluetooth SIG) aimed at novel applications in healthcare, fitness, beacons, security and entertainment [28]. BLE is the most common wireless connectivity protocol in use. One reason for the incredible success of BLE technology is the tremendous flexibility it provides developers. Whether a product streams high-quality audio between a smartphone and speaker or transfers data between a tablet and medical device, The BLE technology has a solution for you.

All BLE devices use the Generic Attribute Profile (GATT). GATT has the following terminology:

- **Client**, A device that initiates GATT commands and requests, and accepts responses, for example, a computer or smartphone.

- **Server**, A device that receives GATT commands and requests, and returns responses, for example, a temperature sensor.

- **Characteristic**, A data value transferred between client and server, for example, the current battery voltage or sensor value.

- **Service**, A collection of related characteristics, which operate together to perform a particular function.

- **Descriptor**, A descriptor provides additional information about a characteristic.

2.2.2 Django (Backend web development framework)
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design [20]. It was designed to help developers take applications from concept to completion as quickly as possible. Django’s primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don’t repeat yourself. Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

In our application we used Django to develop the Web REST API which is responsible acts as an endpoint for node device "PUT" requests and web browser "GET" requests.

2.2.3 Reactjs (Frontend javascript library)
We used ReactJS to develop the Frontend (user graphical interface) of our application. This is the part of the application that the care providers can see and interact with. ReactJS is
an open-source front-end library for building user interfaces. It encourages the creation of reusable UI components, which present data that changes over time.

ReactJS makes it painless to create interactive UIs. Design simple views for each state in your application, and ReactJS will efficiently update and render just the right components when your data changes.

### 2.2.4 Nodejs (Javascript Backend)

Nodejs is another widely used Javascript library. It is mostly used for server-side scripting i.e performing tasks on the cloud server. As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. This is why we used Nodejs to develop the data broadcast functionality in our application.

In our application, it receives sensor data messages from the Django REST API and forwards these messages to subscribed web applications.

### 2.2.5 Socket.io (Realtime communication library)

Socket.io is an additional library which enables some of the broadcast features in our application. It enables real-time, bidirectional and event-based communication between the browser and the server. It runs both on the cloud server and the remote web browser.

### 2.3 Node device (Middle-ware) Implementation

The node device software is implemented using the python programming language. The software running on the user sensor devices expose a BLE characteristic for each sensor value. The Node device gets this data from the user devices by reading these characteristics. Once the initial BLE connection between the user device and Node device is setup, the Node device can request to be notified anytime the value of the characteristic (sensor) changes. This enables to avoid constantly polling the user device to see if new data is available. The node device and continue with other tasks and only get notified when new data is available.

User devices are also configured to send periodic messages called pings. These pings tell the world that this particular device is still active. This feature is particularly important in a healthcare setting. If a ping is not received from a user within a given period, we can assume that something has gone wrong and start making efforts to bring that device back online.

The REST API endpoints which the node device uses to communicate with the server are discussed in the next session.
CHAPTER 2. METHODOLOGY

Step 1: Scan for new BLE devices

Step 2: Connect to BLE devices matching sensor description

Step 3: Wait until message packet is received from connected BLE device

Step 4: Forward message to cloud server by calling REST API

Figure 2.3: Pseudo-code for Node Firmware

2.4 Backend server implementation

The backend server software is made up of the database, REST API, and the Data broadcast system. Starting with the database, our decision to use PostgreSQL was an easy one because it is very widely use and performs well in multiple use cases.

Databases require very detailed configurations. A strong username and password is needed to ensure restricted access to the database content. We also need to setup the table, row and column structure for the data types that will be stored in the database. This structure is described using an object called Models. Fig 2.4 shows the database table structure for each User. It contains, entries like the name, gender, age, unique device id e.t.c

```python
class User(models.Model):
    user = models.OneToOneField(User)
    name = models.CharField(max_length=30)
    gender = models.CharField(max_length=10)
    age = models.PositiveSmallIntegerField()
    device_id = models.CharField(primary_key=True)
    device_type = models.CharField(max_length=10)
    room_no = models.PositiveSmallIntegerField()
    birth_date = models.DateField(null=True)
    comments = models.CharField(max_length=128)
```

Figure 2.4: Database table for User

Fig 2.5 shows the database table for sensor data entry. Each sensor data entry must be matched to a specific user. This enables us to query the database for all data samples belonging to a specific user. We also store the time the the sensor sample was received. Having timestamped data is essential in biomedical applications. It can be used to identify
the exact instant an incident occurred. It is also used to generate a time-series representation of the data changes.

The REST API implemented using Django exposes the following end points.

- **http://localhost:8002/ping**
  This is the end point where ping requests are received. It accepts POST requests from the node device.

Sample ping request:

```json
POST {
    "device_id": "6D57CF753C85",
    "battery": 60,
    "time": 1624733671,
    "name": "Vicki Moates",
    "room_no": 16,
    "device_type": "RR",
    "gender": "female",
    "data": [{"value": 0, "time": 0}],
    "command": "ping"
}
```
• **http://localhost:8002/users/**

  This is the user data endpoint. It supports POST requests to create new users, GET requests to retrieve user information, DELETE requests to delete a user and PUT requests to update user data.

  Sample API call to create new user:

  ```
  POST {
    "user": {
      "url": "http://localhost:8002/users/988/",
      "username": "JohnWoodruff87",
      "email": "JohnWoodruff@gmail.com",
    },
    "name": "John Woodruff",
    "age": 67,
    "gender": "male",
    "device_id": "7BA56B64958A",
    "device_type": "SPO2",
    "room_no": 133
  }
  ```

• **http://localhost:8002/sensordata/**

  This is the sensor data endpoint consumed by the node device to add data to the database. It supports POST add new user sensor data, GET requests to retrieve sensor data based matching a specified filter, DELETE requests to delete a sensor data sample.

  Sample API call to add new data to database:

  ```
  POST {
    "device_id": "7BA56B64958A",
    "senior_id": "user1234",
    "time": "12423223",
    "value": "95.6",
    "device_type": "SPO2",
  }
  ```

When sensor data is received on **sensordata** API end point, it is forwarded to the data broadcast system. The data broadcast system is then responsible for broadcasting this data to all listening web browsers / web apps. The data broadcast system is able to accomplish this using web sockets. A web socket is a computer communication protocol that enables a
two-way interactive communication session between the user’s browser and a server. Socket.io builds upon web sockets to enable communication between one server and multiple browsers.

The data broadcast system is implemented in nodeJs. It is a simple application with one task, to receive data from main REST API and forward to all connected web browsers.

```javascript
async function run() {
    const sock = new zmq.Subscriber;
    sock.connect(`tcp://${zeromq_server}:${zeromq_port}`);
    sock.subscribe(zeromq_topic)
    console.log(`ZMQ sub connected to ${zeromq_server}:${zeromq_port}`);

    for await (const [topic, msg] of sock) {
        data = topic.toString().slice(zeromq_topic.length).trim()
        console.log(data)
        data = JSON.parse(data)
        io.emit(zeromq_topic, data) // Send to socket.io
    }
}
run()
```

Figure 2.6: Code snippet for Data Broadcast System

### 2.5 Front-end implementation

The front-end of a web application is the visible user interface that enables users to interact with the underlying system. This includes both user interface and the user experience. A good front-end application should be visual appealing, intuitive and performant. Our choice the front-end framework, ReactJS, is one of the most widely used.

Here are some of the points we considered while building the front-end interface:

- **Ease of use and intuitiveness**
  We wanted this front-end application to be easy to use and require very little learning time. This is crucial in a healthcare environment where delays or misunderstood data could lead to catastrophic results.

- **Speed**
  The user should be able to reach any part of the web application relatively quickly.

- **Differentiability**
  It should be easy to different abnormal sensor data (users in potentially critical condition) from users in normal condition.
Main Display Area

The Main Display Area is subdivided into 2 components:

1. **Watchlist Area**
   This is an area on the top of the page which is reserved for displaying all users in potentially critical conditions. As new sensor data comes in, the web application checks the data to see if this data value falls outside the threshold considered as normal. If it does, the user is marked as "potentially critical" and displayed in the Watchlist Area. This feature is very useful in a large group living center with hundreds of users. The care-provider can very quickly identify which users might be in a compromised situation.

2. **Main User Area**
   All other users whose sensor data is within normal thresholds is placed in this area. Care-providers can still see their current biosignal values and monitor the trends in these values.

![Watchlist and Other Users in the Main Display Area of Front-end Application](image-url)

In Fig 2.7 we can see that one user has sensor values (Body Temperature) which are outside the normal range. This could indicate a critical health condition or a sensor hardware error. This watchlist allows the care-provider to easily spot this and take the appropriate action.
User Object

The user object is the unique UI object that contains data to describe each user. It is designed to provide almost all the information about the current state of a user at one glance.

![Figure 2.8: Different components of the User UI Object](image)

The user object is also clickable. Clicking on it displays more information about the user including a time-series graph of recent sensor values. The time-series graph is particularly useful in understanding how a signal value has varied within a given period of time.

The gauge in the top right corner of Fig 2.9 acts a simplified interpretation of the user data. The different color zones represent different user states.

- Red: Danger! Critical condition.
- Green: Great Condition.
- Yellow: OK Condition.

2.6 DevOps (Development and Operations)

DevOps is a term that combines software development and technology operations. It describes the tools, setup procedures, and hardware requirements needed to develop and run
our program. One of the tools which we used to setup our entire development and production environment is **Docker**.

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers [7]. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. This concept of containers allows us to run the different sub-components of our application independently, as if they were running on their own machine.

A big advantage of docker is that it enables scalability out of the box. We can configure all our containers to run one a single machine during development and change the configuration to run on more powerful distributed hardware as demand and workload increases.

Docker also makes our software very portable. Once the initial docker configuration is setup, the configuration and installation procedure is not linked to any hardware configuration. We can move our software to a completely different hardware and everything would still work seamlessly.
CHAPTER 2. METHODOLOGY

We divided our cloud application into 4 docker containers:

- Database Container.
- REST API Container.
- Data Broadcast Container.
- Front-end Container.

These containers communicate with each other through predefined port mappings.

2.6.1 Hardware

All the development and test was carried out on my personal computer. This is only possible because of powerful virtualization software like docker.

Development Hardware Specs

- Model: ASUS
- CPU: Intel Core i7-7500U @2.7GHz x 4
- Memory: 16 GB
- OS: Ubuntu 18.05 LTS
- OS type: 64 bit

For more intensive tasks, we have access to a more powerful server computer in the Nursing Engineering Laboratory.

Lab Hardware Specs

- Model: Dell PowerEdge
- CPU: Intel Xeon @2.4GHz x 4
- Memory: 128 GB
- OS: Ubuntu Server
- OS type: 64 bit
2.7 Evaluation Setup

Evaluation and testing are an integral part of any project. There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas executing programmed code with a given set of test cases is referred to as dynamic testing. In our project, we have chosen dynamic testing.
To show that our project meets the required evaluation metrics, we also developed specific tests/experiments to evaluate certain components and aspects of the project.

2.7.1 Evaluation Metrics

As described in the initial project proposal, the following evaluation metrics were used to test our system and provide like for like comparisons with existing solutions.

- **Scalability**: Our system needs to maintain the real-time and availability requirements discussed earlier. We will simulate a scenario with 100 older adults. To run this test, we developed a test software that simulates an individual device.

- **Real-Time Operation**: We developed and ran tests to show that the time from the occurrence of an emergency event to notification on the control dashboard must be less than 10 seconds.

- **Cost**: We will perform a cost analysis and show that our proposed solution is cheaper than existing solutions. Knowing the comprising components and server computational requirements, we can estimate the operating cost of our system in a real-life scenario.

- **Usability**: By offloading most of the logic control to the central control center located in the cloud, we will show that this provides a more worry-free user experience for the older adults. In the control monitoring front-end, information should be laid out in such a way that more attention is drawn to the higher risk situation.
Chapter 3

Results

In this section, we will discuss the results of our tests and experiments and evaluate these results against metrics mentioned in the introduction section.

3.1 Experiment 1: Scalability

Scalability is the measure of a system’s ability to increase or decrease in performance and cost in response to changes in application and system processing demands. Ensuring scalability of a system involves careful choice of tools and sub-components that are individually scalable.

In our case, we have chosen docker, which brings default scalability to our system through the concept of containerization. Each sub-component of our system is separated into a docker container. Docker is able to spin up more instances of a container to meet increased demand and workloads.

We build a test program that simulated an individual user. Using this program we were able to simulate a scenario with 100 concurrent users. Our system remained performant even with this increased workload, thanks to the Docker infrastructure. Fig 3.2 shows a graph of ping response time vs the number of user devices.

Ping response time is the amount of time it takes for messages to travel from the user sensor device to the server. This value is a direct indicator of performance. We can see that even at 100 concurrent users, we are still able to achieve a response time less than 25ms. A value less than 30ms is generally considered to be very good [4].

3.2 Experiment 2: Real-Time Operation

A real-time system has been described as one which ”controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time”. A health-care monitoring system is a good example of a real-time system.
CHAPTER 3. RESULTS

Real-time programs must guarantee response within specified time constraints, often referred to as "deadlines". Real-time responses are often understood to be in the order of milliseconds, and sometimes microseconds.

In our program, we need to guarantee that is able to travel from the bio-sensor device to the front-end application in less than 10 seconds. In other words, if the biosensor on a user detects a critical (sensor values outside the normal range), this information should be
propagated to the server and displayed on the front-end application all within 10 seconds. This should hold true both in periods of high and low workloads.

We were able to test this metric using the data timestamp feature. All data is timestamped as soon as they arrive at the node device. To determine the transit/delivery time at the front-end application, all we had to do was check the current timestamp when the arrived at the front-end and get the difference. Fig 3.3 shows the result of this experiment. We can see that even in the case of very high workloads (100 concurrent users), the message is still delivered in under 2 seconds.

![Figure 3.3: Data transit time Vs Number of Users](image)

### 3.3 Experiment 3: Cost Analysis

Cost is a big part of any project. It is vital to ensure that a product/project is competitive cost-wise. To analyse the cost of our system, we need to estimate the cost of each individual sub-component. A big advantage of cloud software products is that the cost associated with running them is minimal. This is made possible because of the proliferation of cloud platforms like Amazon AWS[23] and Microsoft Azure [15].

These companies provide all the hardware and software tools needed to deploy and manage a software program. The company/developers do not have to purchase and maintain expensive cloud server hardware. These cloud platforms also handle a lot of the issues associated with developing large distributed systems at scale e.g availability, fault tolerance and reliability.

Amazon AWS is currently the largest cloud platform. We will perform our analysis using this platform. We will analyse cost for a use case with 100 concurrent users. For this task, we have selected the **t2.xLarge EC2 Instance** [25]. This Virtual server configuration provides a good mix of compute and memory capability.
CHAPTER 3. RESULTS

Hardware Description

- Platform: Amazon EC2 instance (t2.xLarge)
- CPU: Intel Xeon High Freq
- Number of Virtual CPUs: 4
- Memory (RAM): 16 GB

<table>
<thead>
<tr>
<th>Time Duration</th>
<th>AWS Instance</th>
<th>On-Demand Price/hr*</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily</td>
<td>t2.xlarge</td>
<td>$0.3756</td>
<td>$9.01</td>
</tr>
<tr>
<td>Weekly</td>
<td>t2.xlarge</td>
<td>$0.3756</td>
<td>$36.05</td>
</tr>
<tr>
<td>Monthly</td>
<td>t2.xlarge</td>
<td>$0.3756</td>
<td>$144.23</td>
</tr>
<tr>
<td>Quarterly</td>
<td>t2.xlarge</td>
<td>$0.3756</td>
<td>$432.69</td>
</tr>
<tr>
<td>Yearly</td>
<td>t2.xlarge</td>
<td>$0.3756</td>
<td>$1298.07</td>
</tr>
</tbody>
</table>

Table 3.1: Estimated cost of running our program on The AWS Virtual Server.

An addition cost involved with running a software service is the cost of maintenance and feature upgrades. Modern software solutions are often highly intricate pieces of technology that require regular updates and maintenance. This maintenance work can often be tedious and time-consuming and needs to be planned for well in advance. Industry experts estimate that over 90% of all costs related to a relatively modern piece of software are the regular maintenance costs.

Bugs and unforeseen behaviour are an unavoidable part of software products. After a product has been released, it needs to be monitored frequently so that these bugs can be detected and fixed in a timely manner. To ensure smooth configuration and operation of software programs, an IT administrator is often needed to handle management and configuration of the software setup. We need to add the cost of this maintenance to the total operation cost. Table 3.2 shows a breakdown of these maintenance costs.

<table>
<thead>
<tr>
<th>Task</th>
<th>Duration (Hrs/week)</th>
<th>Price/hr*</th>
<th>Monthly Total</th>
<th>Annual Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT Administrator</td>
<td>10</td>
<td>$50 - $75</td>
<td>$2400</td>
<td>$28800</td>
</tr>
<tr>
<td>Software Developer</td>
<td>5</td>
<td>$50 - $75</td>
<td>$1200</td>
<td>$14400</td>
</tr>
</tbody>
</table>

Table 3.2: Cost of Software Maintenance
CHAPTER 3. RESULTS

3.4 Experiment 4: Usability

Usability is a measure of how well a specific user in a specific context can use a product/design to achieve a defined goal effectively, efficiently and satisfactorily. Usability is a very subjective experience. We have measured usability in our case by asking for direct user feedback. We approached individuals in the healthcare field and they were able to find their way around the program with little or now problems.

![Image showing complete Frontend Application.](image)

However, there are some general guidelines that can be used to evaluate the usability of the system:

- **Learnability**: This is how quickly a new or novice user learns or relearns the user interface to conduct basic tasks. Our system excels in this area. The design is very simply layed out and it is very easy to see where each component is located and what is it used for. The watch-list area is also clearly separated from the rest of the users in a way that is intuitive. The individual user objects also change their appearance when a user moves a mouse over them. This indicates that the object can be clicked to reveal more information.

- **Efficiency**: This is defined as the speed with which a user can complete a task or accomplish a goal. Our system is very efficient. Most of the objects and tasks can be reached with fewer than 2 mouse clicks. A slight slow down in efficiency can arise when there are a lot of online users and the care-giver has to scroll up or down in search of specific users.
CHAPTER 3. RESULTS

• Error Tolerance: This refers to the ability of the system to help users avoid and recover from error. When an error occurs, the user is able to go back to a previous stable state by clicking the back arrow buttons on the browser. This makes it easy to recover from errors when they occur.

The front-end dashboard is visually appealing and attention is immediately drawn to the most important areas of the user object. Users who could possibly be in a critical condition are separated from the other users, making it easier for the care-provider to focus his attention. More detailed information and sensor data for each user is also provided and can be easily accessed by clicking on the user object.
Chapter 4

Discussion

The benefits of long-term bio-signal monitoring cannot be overemphasized. From timely assistance during potentially critical/abnormal condition to long-term health prediction, continuous bio-signal monitoring has found multiple uses. However, not all bio-signals are made equal. Some bio-signals provide more insightful data compared to others. To get the best outcome from continuous signal monitoring, it is important to select and monitor the correct bio-signal.

Blood-pressure is one of the most important (data-rich) bio-signals. Blood-pressure correlates well with the risk of advanced cardiovascular disease (ACVD). Also, hypertension, a medical condition in which systolic BP is above 140mm-Hg and/or diastolic Blood-pressure is above 90mm-Hg persistently, is a risk factor for coronary heart disease and the single most important risk factor for stroke. Systolic Blood-pressure is the maximum pressure in the arteries when the heart contracts. Diastolic BP is the minimum pressure in the arteries between the heart’s contractions. It is even more important in older adults as it provides a summary of cardiovascular and internal health. Numerous studies have confirmed the long-term benefits of continuous blood-pressure monitoring [2]. In spite of all these advantages, blood-pressure monitoring is rarely used in wearable continuous monitoring environments. This is because, there is currently no widely accepted solution for wearable blood-pressure monitoring. The inflated cuff-method is still the most accurate method for blood-pressure measurements. Some researchers have been able to get close approximations of blood pressure using other bio-signals. Maxim Integrated Semiconductors have developed a system, MAXIMBPT, which has shown good results when used for blood-pressure estimation. Maxim Blood Pressure Trending (MAXIMBPT) is a BP measurement system using a photoplethysmography (PPG) signal obtained from the fingertip. The system requires user-specific calibrations, and successive measurements are relative to this calibration point. The goal of the system is to make BP measurements readily available to the user through widely available consumer devices like smartphones or wellness watches. The system is most useful and accurate when measurements are taken in a resting state, as in traditional cuff-based BP systems. This application note contains evaluation methods and design specifications for the MaximBPT system. The most advanced blood-pressure sensing system is currently
being developed by a company, Blumio, which uses Wearable Millimeter-Wave Device for Contact-less Measurement of Arterial Pulses which are then passed through a signal processing algorithm to obtain Blood-Pressure value.

ECG bio-signal (Electrocardiogram), is another good indicator of general body health. It can be used to make different types of predictions, from abnormal hearth rhythms to stress levels. Measuring ECG signals is not as difficult as Blood-Pressure measurement and as a result, there are more solutions available for wearable ECG devices. An ECG device can be easily implemented in our system and be used to monitor trends like RR-Interval, Heart Rate and Heart-Rate-Variability. Our current test/evaluation setup simulates an ECG device that provides RR-Interval data at predefined frequency.

Other important biosignals to keep track of include: core body temperature and bio-impedance. These signals are more easily measured and are readily available in wearable form-factors.

Our system has been developed in such a that any new bio-signal sensor device can be easily added by creating a new tag for this sensor type in the data base. Fig 2.5 shows the sensor types which are currently available on the device.

### 4.1 Comparing With Existing Solutions

In this section we will be comparing our system with the already existing alternatives, to find out where our system improved on the existing technology and where it might be lacking behind.

The current options for continuous health monitoring for older adults include:

- Smart Watches and Fitness Bracelets.
- Smart AI Speakers/Gateways.
- Traditional Remote Health Monitoring Setup.
- Early Sense Remote Health Monitoring Technology.

Smart Watches and Fitness Bracelets include devices like the Apple Watch and Fit-bit Device, which are able to continuously monitor bio-signal data and store them according to a user profile. The main downside of these devices is that they are personal devices and are not designed to be scalable in a community living environment. The data the collect and store is only visible by the user. There is not avenue for care-providers to gain access to this data and intervene in situations where the older adult might be unable to call for help. Smart Watches and Fitness Bracelets, as currently designed, are not well suited for continuous for remote monitoring in older adult care, and as a result do not present a like for like comparison with our system. They are also expensive with a steep learning curve, all of which are factors our current design has improved upon.
CHAPTER 4. DISCUSSION

Smart AI Speakers/Gateways include devices like the Amazon Alexa and Google AI Assistant. These companies have built conversational models which are able to interact with a user like a human or care-provider would. Amazon Alexa is currently being implemented in hospitals, where it is used by admitted patients in a variety of ways, ranging from requesting for help and attention from care providers to controlling electronic devices in the room. This setup, in its current state, does not match some of the core futures our system provides. There is no bio-signal data being collected and the system relies on the user (older adult) being able to detect and report a deviation from his/her normal health condition. These devices are also cost prohibitive because they are designed and built as complex tools that are able to solve multiple problems.

Traditional Remote Health Monitoring is the most widely used method of remote health monitoring. In Traditional Remote Health Monitoring, a user is given a couple of bio-sensors and Graphical Interface. The user has to manually collect bio-signal data using the sensors and enter them in the provided graphical user interface (usually a mobile app or web application on a tablet). It is easy to see how this system is flawed. First, the quality of the data being collected is compromised because the bio-sensor devices have to be operated by the older adult, who are usually not properly trained in health care device operation. This method of monitoring is also not continuous because it relies on the user to routinely collect and enter the data. Usually, the user is only able to do this once or twice a day, therefore the care providers and not able to visualize and understand bio-signal trends throughout the course of a day. Our system really builds on this existing technology by providing seamless data collection and user experience. Data is monitored continuously without any user input.

EarlySense is one of the leading technology companies in the field of remote health monitoring [26]. The EarlySense Monitoring System has been developed to provide continuous monitoring of heart rate (HR), respiration rate (RR), and bed motion for patients in medical/surgical, oncology, orthopedics, isolation, post-partum, skilled nursing facilities, long term acute care, and rehabilitation settings [9]. EarlySense is a low-acuity continuous monitor. EarlySense consists of a sensor that is placed under the patient’s mattress, a bedside monitor, a central display station, and proprietary analytic software that runs on a PC. The system is based on a piezoelectric sensor, sensitive to applied mechanical strains. The system differs from other patient monitoring systems in that it is a contact-less device which eliminates the use of telemetry leads. While the patient is lying flat in bed, the system continuously records heart rate (HR), respiration rate (RR), and bed motion. Low-acuity systems do not provide cardiac wave-forms (rhythm strips).

The EarlySense Monitoring System provides a direct like for like comparison with our system. However our system does have some clear advantages.

- **Sensor Variety**: The EarlySense system only offers one sensor type (piezoelectric sensor). This limits the types of bio-signals that can be collected to only HR (Heart-Rate), RR (Respiratory Rate) and Motion. In contrast, our system has been built to
be modular and support any bio-sensor. This ensures that the data collection can be customized for each user.

- **Flexibility and User Experience**: The piezoelectric sensor used in the EarlySense system is placed under the mattress which the user lays on. This means that user data can only be collected when the user is laying down on the bed. It is easy to see how this limits the application of this system to critically ill and bedridden patients. This is a big problem because we also want to collect user data during periods when the user is not laying down. Our system supports any type of sensors. The wireless communication through Node devices means that the sensor devices can be designed to be wearable and the user can freely move around without compromising data collection.

- **Cost**: The EarlySense system is more expensive than our proposed system. Each user requires a sensor device and an electronic bedside monitor. This additional bedside monitor introduces additional cost. While this bedside monitor can be advantageous in an acute care setting (hospitals and post-surgery recovery facilities), they are not needed in a long-term care setting.

- **Power Consumption**: This is one area where the EarlySense system is marginally better than our system. The piezoelectric sensors used by EarlySense system have very low power consumption. Because the device is always placed in a fixed location under the mattress, it is powered by a connection to a wall socket. This means that it can run indefinitely as long as there is electric power supply. In contrast, our sensor devices are designed to be wearable which means that they will be battery powered. They will need to be recharged periodically. However, I believe that the freedom and user experience enabled by having a wearable device outweighs the battery life limitations.
Chapter 5

Conclusion

Health care for older adults is both expensive and of low quality. The high demand for older adult care personnel and facilities has led to a general decrease in care quality. Our system provides a smart remote monitoring solution that can provide a better aging experience for older adults and their families. Just like robots have been used to improve efficiency and accuracy in many fields, a smart network of sensors can be used to reinvent older adult care. By automating some of the simpler tasks performed by nurses and care-providers (ward rounds, routine check-ins), this system reduces the operating costs for nursing home operators and improve the quality of care for the older adults.

Historical bio-signal data will also be very instrumental in monitoring general health of older adults and making data-backed decisions/prescriptions. From remote skin temperature monitoring for COVID19 detection, to Arrhythmia detection using ECG Sensors, our system has the potential to a solution for many healthcare problems. There are some areas in which our work can be extended/improved:

- **Machine Learning on the Edge:** Edge ML is a technique by which Smart Devices can process data locally (either using local servers or at the device-level) using machine and deep learning algorithms, reducing reliance on Cloud networks. Using Edge ML, it is possible to perform signal processing and predictions on the Node device, thereby improving the latency of data transfer. This can be used for applications like Fall Detection. The current state of the art fall detection applications involve using An LSTM Neural Network (Long short-term memory) which is fed accelerometer and motion data from the user wearable devices. [16][12][13] Multiple researchers have shown that is able to achieve 90% accuracy for fall detection when using LSTM.
Appendix A

An appendix

The complete code for the project is located on GitHub at the following link:
https://github.com/uched41/grandma_su/tree/dev

An offline copy of the project will is also setup on the NE-Lab Server PC. Here are the steps to replicate this system on a different server or PC:

• Install Docker and Docker-Compose (https://docs.docker.com/get-docker)

• Clone GitHub repository.

• Open a shell terminal.

• Navigate to the Server directory.

• Run command `docker-compose up`. You might need to use `sudo` to gain the required system privileges. This command starts up the entire system (database, REST-API, Frontend application e.t.c.).

• The Frontend Application can be assessed on port 3000 of the localhost address. This can be changed to a public address depending on the user needs.

The Test folder contains the application used to simulate user sensor devices. Here are the steps needed to simulate a user device:

• Open a new shell terminal.

• Navigate to the Test/Server_Test Folder.

• Run command `python3 test1.py 5`, where 5 is the number of uses you want to simulate. This command will randomly create 5 (or n) new users and register them on the server. They should be immediately visible on the Frontend Application.

To stop the server and all the sub-components use `CTRL + C`. This will work for both the server and the test application.
Bibliography


