A Study of Continuous Heartbeat Monitoring System Using The Internet of Things Approach

2022, SEPTEMBER

Eko Sakti Pramukantoro

Graduate School of Interdisciplinary Science and Engineering in Health Systems

(Doctor‘s Course)

OKAYAMA UNIVERSITY

Dissertation submitted to
Graduate School of Interdisciplinary Science and Engineering in Health Systems
of
Okayama University
for
partial fulfillment of the requirements
for the degree of
Doctor of Philosophy.

Written under the supervision of

Professor Akio Gofuku

and co-supervised by
Professor Jinlong WU
and
Professor Masanobu ABE

OKAYAMA UNIVERSITY, September 2022.

TO WHOM IT MAY CONCERN

We hereby certify that this is a typical copy of the original doctor thesis of
Eko Sakti Pramukantoro

Signature of

the Supervisor

Seal of

Prof. Akio Gofuku

Graduate School of Interdisciplinary
Science and Engineering in
Health Systems

Contents

Abstract

Regularly monitoring heart conditions may help avoid heart disease, which can escalate to life-threatening scenarios. The heartbeat is monitored by examining the Electrocardiogram (ECG) pattern. Equipment such as a Holter monitor is utilized to obtain ECG data. Next, the physician will analyze the recording to seek the pattern regarding abnormalities. Conducting regular checkups can be challenging due to non-technical and technical aspects. An example of the non-technical aspect is a situation in a pandemic that leads to difficulties in making an appointment with a physician or other things like busyness. The technical aspect is related to the technology for conducting a regular checkup. Recording a cardiac activity using a Holter monitor has a drawback that limits the patient's movement, especially for long-term recording. In some cases, it is necessary to conduct a long-term recording of ECG because the irregular heartbeat may not appear during short examinations in health care facilities. For this case, flexible ECG equipment is preferred. Moreover, interpreting a long ECG recording will burden medical staff. Thus, an automated ECG analysis is needed.

This thesis presents a study of a continuous heartbeat monitoring system using the internet of things approach. I design frameworks for heartbeat monitoring that offer flexibility and continuous monitoring of heartbeat conditions. Our framework consist of of three modules: *things*, *middleware*, and *applications*. In this case, *things* are a medical sensor device with a gold standard cardiac sign and may replace the Holter monitor with limitations in active mobility. *Middleware* is the gateway to forward data from things to the applications. Then, *applications* usually consist of resources for analyzing and visualizing the collected data. I implement two frameworks to provide different data analysis functions. First, I use a fog-computing framework for real-time processing. Second, I use cloud computing for batch processing. In the implementation of cloud-based monitoring, I use micro-service architecture. I use three virtual machines, the first for application and database, the second for the classifier based on RR interval (RRi), and the last for the classifier based on morphology. A web service is provided for each module to communicate.

As the first contribution of this thesis, I investigate a wearable device, Polar H10, as a sensor to record cardiac activity. Polar H10 offers better measurement in the RRi of a person's heart rate while performing various activities. It produces several formats of cardiac parameters such as Heart Rate (HR), RRi, and ECG. The experiment results show that the produced RRi from Polar H10 is suitable for real-time and continuous heartbeat detection, while ECG is suitable for batch processing.

As the second contribution of this thesis, I developed a middleware with a Bluetooth Low Energy (BLE) framework to obtain RRi and ECG data from Polar H10 in a continuous and long-term way. The middleware is designed to overcome interoperability issues by enabling (1) multi-protocol communication; and (2) standardizing sensor data format.

Polar H10 is equipped with BLE as the communication protocol. I conduct experiments with several BLE frameworks such as *Core Bluetooth*, *PyGATT*, and *BLEAK*. Due

to its cross-platform capabilities, I selected BLEAK to be integrated into our middleware. In terms of data transmission capabilities, good data transmission can be achieved by having the *received signal strength indicator* Received Signal Strength Indicator (RSSI) above -80 dBm from Polar H10 to middleware. Its communication range reached 16 m in an indoor with obstacles environment and 45 m in a line-of-sight space.

The received data size could become significant in the continuous and long-term recording. Thus, selecting an appropriate tool for data storage management is essential since it needs to store the analysis results. MongoDB is a scalable database suitable for storing varied data types. Our investigation shows that it works by storing data in the document. I use a topic to distinguish one sensor data from another. Moreover, I provide personal data management, which allows access to users' cardiac data. This function was developed in a web-based interface using the Django framework.

As the third contribution of the thesis, I propose a heartbeat classifier based on RR interval data for real-time and continuous heartbeat monitoring using the Polar H10 wearable device. Several machine learning and deep learning methods were used to train the classifier using the MIT-BIH arrhythmia database. Although it is a mature database, it has an imbalance of classes. I applied oversampling methods to achieve higher classifier accuracy to overcome this issue. In the training process, I also compare intra-patient and inter-patient paradigms on the original and oversampling datasets to achieve higher classification accuracy and the fastest computation speed. As a result, with a constraint in RRi data as the feature, the random forest-based classifier implemented in the system achieved up to 99.67% for accuracy, precision, recall, and F1-score. I also conducted experiments involving healthy people to evaluate the classifier in a real-time monitoring system.

Finally, as the last contribution of the thesis, I propose the second classifier based on ECG morphology. ECG morphology is a feature based on the shape of *PQRS* waves. I train the classifier using a dataset from the MIT-BIH arrhythmia database to classify ECG data from Polar H10. Thus, I split the dataset into 70% for training, 10% for validation, and 20% for testing. Convolutional Neural Network (CNN) is used to extract features automatically, and I use a multi-layer perceptron (MLP) for classification. As a result, our morphology-based classifier can achieve an accuracy of 97%, followed by 96%, 97%, and 96% for precision, recall, and F1-score, respectively. I have implemented the classifier based on ECG morphology into our cloud-based monitoring system. The result shows that our system can predict ECG data from Polar H10 and visualize the results.

In the future, I would like to extend the implementation for real experimental studies in corporations with a medical professional to identify the type of heart disease and other real-case scenarios where users perform more vigorous activities, such as sports.

Acknowledgement

千里の道も一歩から
A journey of a thousand miles begins with a single step.

This dissertation is only possible with the enormous support I received during this study. Here, I would like to express my heartfelt gratitude to everyone who has helped me by volunteering their valuable time and effort.

First and foremost, I am indebted to my supervisor, Professor Akio Gofuku. His continuous support and guidance have helped my research in a way that should be expressed only with the deepest appreciation. I had valuable experiences in this laboratory with the vast opportunity given to me.

I would like to also express my gratitude to indebted to my co-supervisors, Professor Jinglong Wu and Professor Masanobu Abe. With their helpful comments, the outcome of my research could be improved with new perspectives.

This endeavor also would not have been possible without such a great environment of the Interface Systems Laboratory. Professor Tetsushi Kamegawa and Dr. So Shimooka have continuously provided many constructive comments during the interim presentations. The technical staff, Mr. Mitsunobu, and the secretary, Ms. Hiroko Natsume, for their utmost support during my study in the laboratory. Also, all lab members who made this journey possible.

I would like to acknowledge the Ministry of Education, Culture, Sports, Science, and Technology of Japan (MEXT) for financially supporting my Ph.D. study at Okayama University.

Finally, I must express my profound gratitude to my family and friends. Thank you for your unending encouragement, support, and prayers.

Okayama, Japan
September 2022

Acronyms

ECG Electrocardiogram

RRi RR interval

HR Heart Rate

RSSI Received Signal Strength Indicator

BLE Bluetooth Low Energy

MLP multi-layer perceptron

CNN Convolutional Neural Network

SVM Support Vector Machine

ANN Artificial Neural Network

RF Random Forest

WHO World Health Organization

CVD Cardiovascular Disease

MQTT Message Queuing Telemetry Transport

CoAP Constrained Application Protocol

LE Low-Energy

Wi-Fi Wireless Fidelity

IoT Internet of Things

IoHT Internet of Health Things

HTTP Hypertext Transfer Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

PPG Photoplethysmogram

HRM Heart Rate Measurement

SDK Software Development Kit

AAMI Association for the Advancement of Medical Instrumentation®

JSON JavaScript Object Notation

N Normal Beat

SVEB Supraventricular Ectopic Beat

VEB Ventricular Ectopic Beat

F Fusion Beat

Q Unknown Beat

GAP Generic Access Profile

GATT Generic Attribute Profile

L2CAP Logical Link Control and Adaptation Protocol

ATT Attribute Protocol

SMP Security Manager Protocol

HCI Host Controller Interface

UUID Universal Unique Identifier

HRV Heart Rate Variability

JWT JSON Web Token

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The World Health Organization (WHO) stated that Cardiovascular Disease (CVD) is the leading cause of death. According to their calculations, CVD killed 17 million individuals in 2005, accounting for 30 percent of mortality globally. 7.2 million individuals died from heart attacks, while 5.7 million died from strokes. CVD-related mortality will reach 23.6 million by 2030 if trends continue [1]. The primary causes of cardiovascular disease include tobacco use, high blood pressure, diabetes, overweight, uncontrolled physical activity, and a poor diet.

A heart disease that leads to life-threatening situations can be prevented by conducting regular heartbeat condition monitoring [1]. The common procedure for early detection of heart disease is to conduct a heartbeat measurement using an electrocardiogram (ECG) that records the electrical impulses during cardiac contraction and relaxation in the heart. The ECG recording provides information regarding cardiac function [2]. Equipment such as a Holter monitor is utilized to obtain ECG data. Next, the physician will analyze the recording to seek the pattern regarding abnormal patterns. Conducting regular checkups can be challenging due to non-technical and technical aspects. An example of the non-technical aspect is a pandemic that leads to difficulties in making an appointment with a physician or other things like busyness. The technical aspect is related to the technology for conducting a regular checkup. Recording a cardiac activity using a Holter monitor has a drawback that limits the patient's activity, especially for long-term recording.

Currently, flexible ECG equipment is available as wearable devices such as chest traps, fitness devices, smartwatches, or armbands. Initially, those devices are intended for fitness equipment. At the same time, continuously recording ECG is necessary because irregular heartbeat may not appear during short examinations in health care facilities. Adopting a wearable device will also enable heartbeat monitoring in the long term. Nonetheless, a chest strap such as Polar H10 can replace a Holter monitor to record cardiac activity [3]. Unlike a Holter monitor, this device can continuously measure users' heart rates during any activity [4]. It produces several formats of cardiac parameters such as heart rate (HR), RR interval (RRi), and electrocardiography (ECG) [5]. Additionally, automatic analysis of heart conditions that classify sensor data should be designed alongside the wearable device so that it will not burden the medical staff with interpreting a long heartbeat recording.

At the same time, the massive attention on the internet of things (Internet of Things (IoT)) in recent years, where heterogeneous devices are connected, has extended the bor-

ders and capabilities of physical things and virtual components. The implementation of IoT in healthcare is known as the internet of things in healthcare (Internet of Health Things (IoHT)) [6]. The concept internet of things was invented by Kevin Ashton [7]. Fundamentally, IoT consists of *things*, *middleware*, and *application* that are connected over the internet. The wearable heart monitoring device, such as Polar H10, can be an example of *things*. While the devices are typically resourced constrained, thus the processing of the recorded data should be done at the application on the internet site. Therefore, *middleware* acts as a collector and data forwarder from the sensor to the internet-based application. The internet-based *application* consists of data storage, a web application that enables users to manage their data, and analytic and prediction functions. Therefore, the wide-opened opportunity to develop automatic heartbeat monitoring using the IoT approach will greatly improve healthcare services.

1.2 Contributions

This dissertation proposes a continuous heartbeat monitoring system using the internet of things (IoT) approach. Instead of using a conventional device (e.g., Holter monitor), it adopts a wearable device that enables long-term monitoring and flexible participant activity during monitoring. This system aims to provide continuous heart condition monitoring and early detection of cardiovascular disease using the IoT approach.

1.2.1 Investigation of Wearable Sensor Devices and BLE Frameworks for Continuous Data Retrieval

The first contribution of this dissertation is the investigation of continuous data retrieval from wearable devices through the BLE framework. Polar H10 is adopted as the health sensor device which produces several cardiac signs such as RR interval and ECG of a user. I also presented data collection from the device using a BLE framework for long-term data recording. Moreover, a middleware is proposed to answer the interoperability problems in IoT. The middleware is equipped with a BLE interface to build communication with low-energy devices, data standardization using JSON format, and an HTTP interface for forwarding sensor data to other devices, such as cloud applications.

1.2.2 Proposal of A Heartbeat Classifier for Continuous Prediction using a Wearable Device

The second contribution of this dissertation is the proposal of a heartbeat classifier to predict output data from a wearable device [8, 9]. I explore Polar H10 output data as features in the heartbeats classifier. Starting from HRV, RR interval, and ECG morphology features. The classifier proposed to predict heartbeat into five classes, namely, normal beat (N), supraventricular ectopic beat (SVEB), ventricular ectopic beat (VEB), fusion beat (F), and unknown beat (Q), following the described classes by the Association for the Advancement of Medical Instrumentation® (AAMI). Since HRV is not suitable for multi-class classification due to the limited dataset. Thus, I focus on the first classifiers based on RR intervals for continuous monitoring and real-time prediction. The second classifier is based on ECG morphology to predict ECG signals from Polar H10.

1.2.3 Proposal of IoT-based Monitoring Framework for Early Detection of Cardiovascular Disease

Finally, as the last contribution, I extend this study by designing and implementing an IoT-based cardiovascular monitoring system based on RRi and ECG data of a wearable device [10, 11]. Two kinds of applications are proposed: fog-based and cloud computing. In fog-based applications consist of the sensor and middleware. In comparison, cloud computing-based consists of web applications and data analysis, which runs on a cloud computing environment. The application based on fog computing is provided for real-time or stream monitoring and prediction. The second application is based on cloud computing for multi-analysis and scalability. The system also provides several classifiers based on data type from polar H10 classifiers for RR interval data and ECG data. Moreover, the frameworks are also used to test the capability of the proposed classifiers. Finally, the application is intended to self-monitor of user's heartbeat condition.

1.3 Organisation of the Dissertation

The remaining parts of this dissertation are organized as follows. Chapter 2 reviews related studies in the literature on IoT-based healthcare services as preliminaries. Chapter 3 presents the investigation of wearable sensor devices and BLE frameworks for continuous data retrieval. Chapter 4 presents the proposal of a heartbeat classifier for continuous prediction using the RR interval data from a wearable device. Chapter 5 presents the proposal of an IoT-based monitoring framework for the early detection of cardiovascular disease. Finally, chapter 6 concludes this study with some future works.

Chapter 2

Literature Review

This chapter presents reviews of the literature on the studies of IoT-based heart monitoring and predictions using wearable devices.

2.1 Internet of Things

Kevin Ashton coined the term Internet of things (IoT) in 1999 [7]. As shown in Figure 2.1, IoT is a system that consists of things and applications on the internet. The characteristics of things are constrained devices; some have limited computational and connectivity but have the capability to produce important data. In this case, things work as sensors. With those characteristics, data from sensors need to deliver to devices with computational capability. These jobs are handled by middleware. Middleware act as a gateway and bridging sensor with application on the internet. The application in IoT runs on a cloud computing environment. This application offers the capability to analyze collected data from sensors into meaningful information.

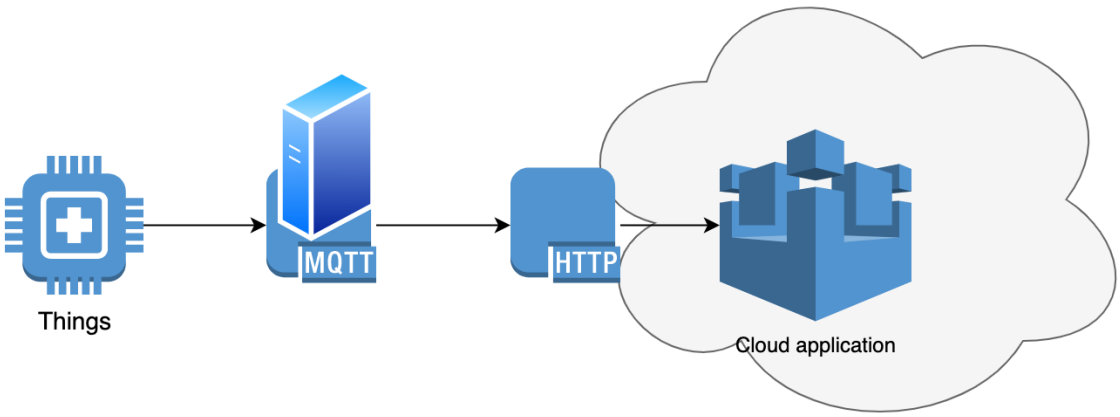

Figure 2.1: Internet of things architecture

Several protocols also play an important role in IoT, such as Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), BLE, and Zigbee. MQTT is used for protocol communication in things areas. In the MQTT scheme, Sensor becomes a publisher; the middleware will work as a broker, then some other device will act as a consumer or subscriber. Instead of MQTT, other protocols such as CoAP, BLE,

and Zigbee are suitable for delivering data from sensors to other devices. Those protocols work on a license-free band of 2.4 GHz. MQTT and CoAP run on top of Wireless Fidelity (Wi-Fi), where Wi-Fi offers higher bandwidth but also needs higher power consumption. Zigbee and BLE are more suitable for low-energy devices. However, ZigBee demands more time to send data compared to other technology. BLE was built for the Low-Energy (LE) device and offered frequency hopping to address interference with other frequencies. Thus, BLE is properly used as protocol communication with a low-energy health sensor device [12].

IoT systems can be implemented in several sectors, in the medical field known as the Internet of Medical Things. In medical areas, health sensor devices were used to produce health vital signs, such as heart rate sensors and blood pressure sensors. Thus, cloud applications consist of analyzing and visualizing collected sensor data. Several authors have proposed a monitoring system based on IoT in healthcare areas. Wang proposed a system using wearable devices to support the physical activity of volleyball athletes [13]. Damian proposed IoT based application to observe the location and actions of the elderly as an autonomous supporting system [14]. Jabeen offers IoT-based systems that are advantageous for isolated regions because usually, a cardiologist is not available in a rural area [15]. In their work, a biosensor was placed in an isolated area to gather health data, and then data was passed into a cloud environment to be analyzed. The cloud-based automatic analyzer offers excellence compared to the traditional analysis of health data. Traditional rule-based diagnosis is inefficient due to an enormous amount of data that requires a medical expert. The problem becomes more challenging with the limited availability of medical experts. Then, a method to analyze the collected data from sensors becomes necessary to gain useful information.

On the other hand, data can reveal previously unknown information about health and lead to discoveries. Many solutions for data analysis based on cloud computing have been proposed. Hossain suggested a system for monitoring the elderly utilizing an ECG platform and a one-class Support Vector Machine (SVM) classifier [16]. The IoT-based system is flexible and could overcome the centralized system, which is prone to overwhelmed computing capacity [17]. IoHT is believed to bring transformation in healthcare by enabling anytime, anywhere, and analyzing patient data as personal healthcare.

2.1.1 Challenges in IoT

The use of IoT for monitoring heartbeat conditions must consider several aspects such as:

1. Standardization
   A factory-based system with a sensor device usually works independently and primarily connects with a smartphone application. However, it lacks compatibility to interact with other devices from a different vendor. This condition raises interoperability issues. Interoperability is defined by Desai [18] and Razzaque [19] as three terms that lead to a device can not communicate with other devices. Those terms are described as follows:

   (a) Interoperability of networks. This interoperability relates to media transmissions such as BLE, WIFI, 6LoWPAN, and LPWAN.

   (b) Semantic Interoperability. Semantic is related to data format. Each sensor cloud produces a different data format.

(c) Syntactical Interoperability. Syntactical related to protocol communication include CoAP, MQTT, HTTP, and BLE.

A middleware that offers several ways of communication could be a solution to Interoperability in IoT. The middleware should provide communication features that can communicate with sensor devices using low-level protocol communication and interact with other devices using high-level protocol communication. Moreover, the middleware should offer to format or standardize data format, i.e., JSON. Thus, data from one device is readable by other devices. BLE is a known protocol developed to connect among resource constraint devices. In BLE, one connection consists of a minimum controller and peripheral. In IoT cases, the sensor is peripheral, and the middleware is a controller. The quality signal on BLE is measured by the strength of RSSI on the controller side [20]. While for high-level communication, Hypertext Transfer Protocol (HTTP) is excellent protocol communication. HTTP runs on top Transmission Control Protocol/Internet Protocol (TCP/IP) protocol and can deliver data using several messaging services such as web service. Standardization using web services could solve interoperability problems [21].

2. Accuracy of health sensors devices
   Currently, the sensing technology to record heart vital signs form as a pulse sensor, blood pressure sensor, Photoplethysmogram (PPG), and ECG. Mainly those devices intended for fitness utilities. A sensor that can replace the Holter monitor to record heartbeat data requires a health monitoring system. Polar H10 device has promised to substitute a Holter device among all available devices on the market. The study was done by Rahel, which compared an ECG Holter monitor and Polar H10 on subjects during rest and activity. The results suggest that Polar H10 has a gold standard for measuring RR interval assessment in several activities (low until high) [3].

3. Continuous monitoring
   The abnormal heartbeat pattern may not be seen during a short recording. If using Holter for long-term recording will be a burden for movement, as an alternative, I replace the Holter monitor with a wearable sensor device. Katrina shows the capability of Polar H10 for continuous long-term recording [22]. Usually, wearable sensor devices are equipped with a smartphone application to obtain heartbeat data. In this case, a smartphone act as middleware. However, the smartphone-based application lacks computation capability and duration of the recording. Thus, a desktop application could be a solution to obtain data from the sensor continuously. A desktop-based application can communicate with the sensor devices using BLE frameworks such as BLEAK, Core Bluetooth, and PyGATT.

4. Data storage management
   The data from the sensor device can vary, and collected data grow over time. Thus, a database that could address the challenge of volume, velocity, variety, veracity, and value is required. These databases are known as No-SQL. Several No-SQL databases, such as MongoDB, HBase, and Cassandra, are suitable options [23].

5. Data analysis
   Data analysis in an IoT environment aims to extract meaningful information from

collected sensor data. In the healthcare domain, i.e, for a heartbeat monitoring system, data analysis is provided to classify normal beats and abnormal beats. Physicians can easily determine the abnormal beats, however However, the problem becomes complicated and tiring for professionals to analyze long ECG recordings in a short time. The human eye is inappropriate for detecting the ECG signal's morphological variation. Thus a classifier based on machine learning that can automatically classify data based on sensor data characteristics is required [6].

2.2 Polar H10

Polar H10 is a LE device that can sense heart activity in various output formats. A coin battery powers this device. With the simple and small size, Polar H10 offers an accurate measure of RR interval compared to multi-lead ECG devices. Polar H10 uses a single lead ECG wrapped with an elastic-polymer strap. By using BLE, Polar H10 sends the data to other parties. In this manner, cardiac sign data that can be sent using BLE is Heart Rate HR and RR interval RRi. As shown in Figure 2.2 [24] through the BLE framework, I utilize Polar H10 to measure cardiac signs. This study uses Heart Rate Measurement (HRM)and ECG measurement. Initially, the polar H10 will send the advertisement to another device (as a sign of the Peripheral's presence); then, the Host will initiate a request by sending data 02 00 00 01 82 00 01 01 0E 00 to PMD control. Those sequence of data means a request to start measurement of ECG. The Host determines the duration of the recording. Thus, to stop recording Host needs to initiate by sending closing data 03 00 to PMD control. The technical specification of ECG recording by Polar H10 uses a single-lead with a sample rate of 130 Hz.

Polar also offers to access their proprietary data, such as EGC, through their Software Development Kit (SDK). Regarding the Polar documentation, this device works optimally in the range of 5 meters. Although it can work in various ranges between 10-35 meters, the open field can reach 100 meters. However, several variables can affect this range, such as interference (Wi-Fi and another Bluetooth signal nearby) and obstacles like a wall in an indoor area. The maximum communication range may vary in each device and depend on the environment. Such as the wall can reduce the communication range of this sensor. However, in some cases, the signal can bounce through the wall. The last thing that can reduce range is high air humidity [4]. Instead of meters, Received Signal Strength Indicator RSSI can be used as a unit to measure distance is BLE. RSSI is the signal strength received by the Rx node that comes from the Tx node. According to the literature, BLE can work properly below -70 dB [25].

2.3 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a communication mechanism that allows low-energy devices to interact with other devices. BLE is structured in three main stacks. The first stack is Application. The Application provides the interface and profile of the device for the user. The second is the Host. This stack consists of Generic Access Profile (GAP), Generic Attribute Profile (GATT), Logical Link Control and Adaptation Protocol (L2CAP), Attribute Protocol (ATT), Security Manager Protocol (SMP), and Host Controller Interface (HCI). The last stack consists of Host Controller Interface (HCI), Link Layer (LL), and Physical Layer (PHY) called Controller [25].

To develop applications based on BLE, understanding protocol ATT, GAP, and GATT is a major concern. ATT handle data organization and formulates into attribute such as Universal Unique Identifier (UUID). GATT works by exposing the service and characteristics of the device, whereas service is a function offered by the device, and characteristic is a data value that is offered in service. Description of the profile is coded into 16-bit UUID numbers, i.e., GATT characteristics and Object type. Furthermore, GATT provides a function to exchange profile information through BLE links. Information in the attribute and profile is regulated by Bluetooth Special Interest Group (SIG). All stakeholders must comply with that agreement. Table 2.1 and 2.2 [24] is an examples of UUIDs that have been agreed upon and used in this study.

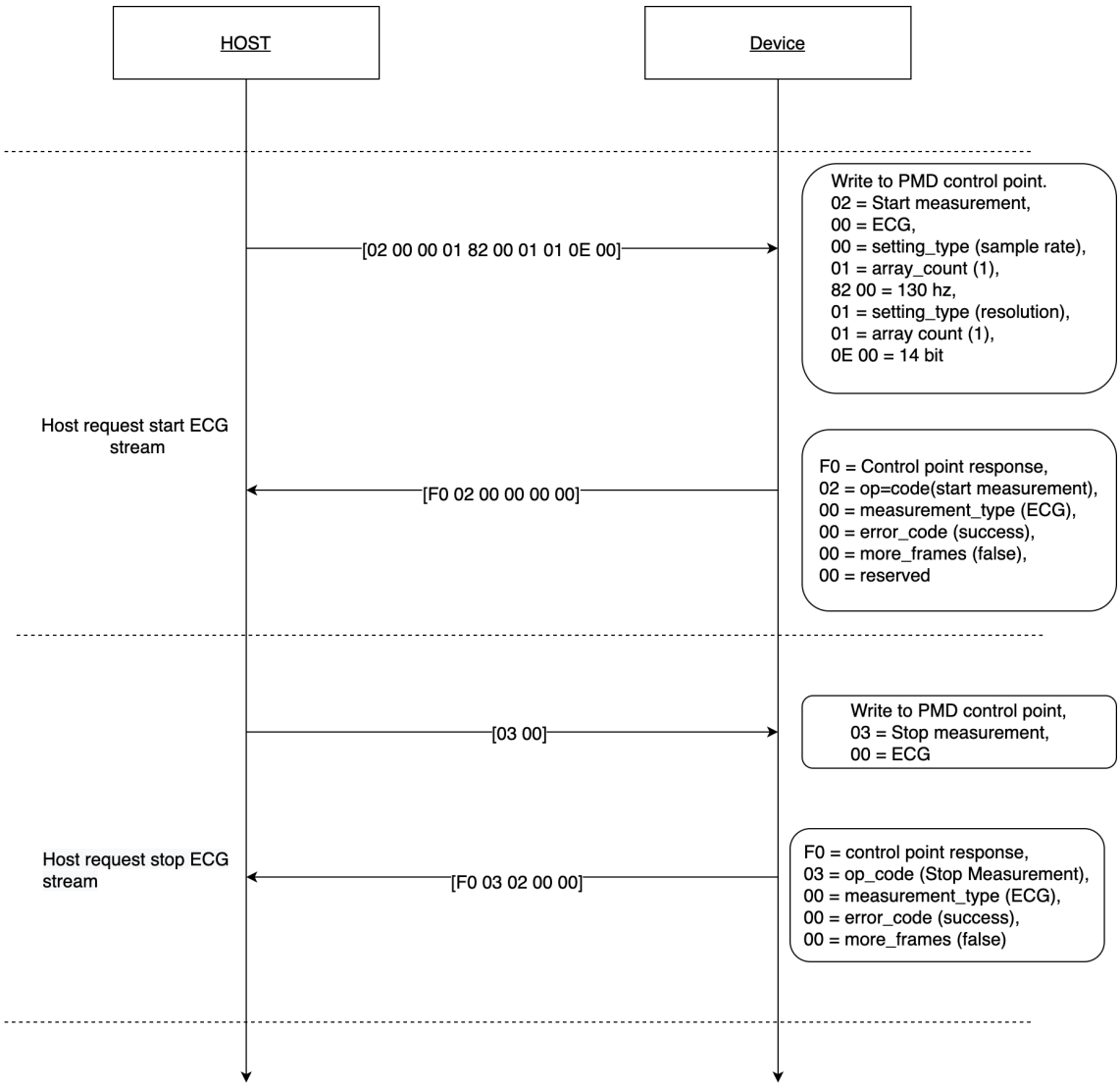

Figure 2.2: Request ECG stream

Table 2.1: GATT Characteristic and Object Type

UUID	**Issuing for**
0x2A37	Heart Rate Measurement
0x2A38	Body Sensor Location

Table 2.2: GATT Services

UUID	**Issuing for**
UUID	Issuing for
0x180D	Heart Rate
0x180F	Battery
0x180A	Device Information
0x1804	Tx Power

The correlation between ATT, GAP, and GATT is as follows; Generic Access Profile (GAP) defines rule discover devices, establish connections, manage the connection, and more. In general, the GAP has the task of maintaining advertising and connection. While Generic Attribute Profile (GATT) handles how the device is discovered, how to exchange profile and data and read and write data. How BLE addresses interoperability from a different vendor is through GATT. GATT uses ATT to exchange data, and how to organize the data is called a service.

BLE makes communication by broadcasting and connection. Broadcast is used for sending advertising packets, while the connection is a permanent connection that consists of a central (master) or peripheral (slave) to exchange information periodically. Most BLE communication started with broadcast advertisement. Two or more devices can make communication among BLE devices at the same time. It consists of a master and slave, which is called a Piconet. Theoretically, one Piconet can make a network consisting of 8 BLE devices. Another form of BLE network topology is scattered, the master of a slave being connected in another Piconet.

As shown in Figure 2.3-a [25], Piconet is an ad hoc network that possibly consists of a maximum of 8 devices. Device A is a master or central, while device B is peripheral or slave. Figure 2.3-b [25] is another form of Bluetooth network called scattered. Scatternet is a network consisting of a minimum of one Piconet that communicates with another. Thus, in BLE, one central (master) can communicate or receive data from other peripherals (slave).

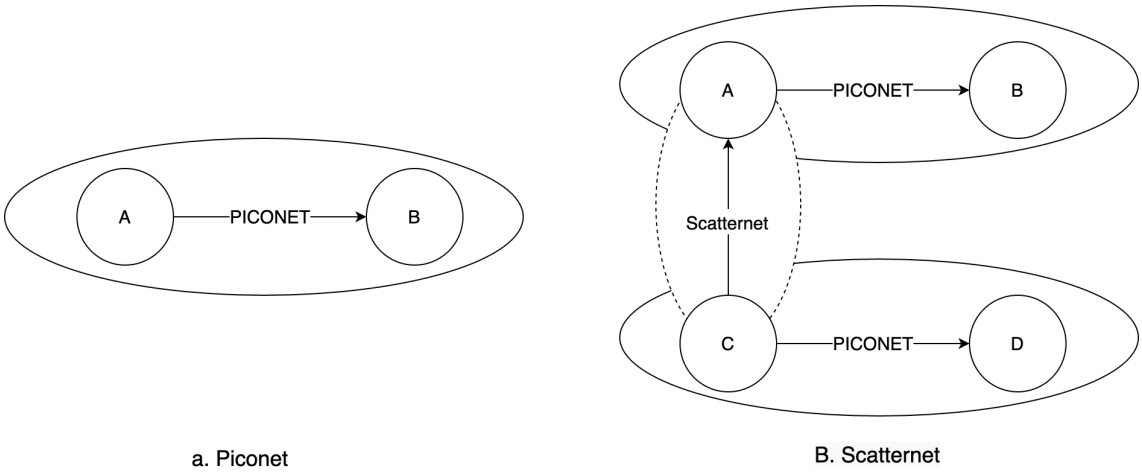

Figure 2.3: Bluetooth topology

2.3.1 BLE Frameworks

Recently low-energy (LE) devices are already equipped with Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) and BLE. BR/EDR is the code for mentioned Bluetooth classic radio. Moreover, BLE software is available for developing apps that utilize BLE as a communication mechanism. Several BLE frameworks are available freely on smartphones and desktop environments, Such as PyGATT, Core Bluetooth, and BLEAK. Core Bluetooth is a framework that provides a function to communicate with LE or BR/ED devices on macOS applications. Core Bluetooth is available for iOS or macOS applications to interact with low-energy peripheral devices such as Polar H10. PyGATT, a Python module for Bluetooth Low Energy Generic Attribute Profile (GATT). This module was developed to communicate to sensor devices related to the device‘s GATT descriptors. This module comes as an interface on a desktop computer, whereas all peripherals are used to connect with a smartphone. While Bleak offers the capability runs on multi-operating system environments.

The BLE framework was developed to establish wireless communication between low-energy (LE) devices with a smartphone. The LE device could produce health vital signs as a fruitful data source for further analytic processes. However, sending data continuously is not recommended due to the smartphone's limited power and computation resources. Thus, the BLE framework for desktop applications was developed. Nevertheless, compatibility becomes an issue in the developing system interacting with LE devices. This study evaluated and experimented with existing BLE frameworks, Core Bluetooth and PyGATT. Two programs representing each BLE framework were developed, and Polar H10 was used as an LE sensor device. This experiment was evaluated by observing parameters such as compatibility of software and hardware, sensor data, connection mechanism, and reachable range communication.

Wireless technology is a major concern regarding communication between sensors and middleware. As one of the developed wireless communication with the lowest power consumption, Bluetooth low energy (BLE) can be a suitable communication protocol for a low-energy health sensor device. However, the compatibility and interoperability issues of low-energy (LE) sensor devices (i.e., Polar H10) in medical applications should be supported by a proper framework to create low-energy communication between sensors and desktop computers. Moreover, a significant evaluation is needed to ensure the system

works properly [12].

2.4 Automated Heartbeat Classification

Anomaly in heart conditions can be recognized according to the heartbeat characteristics on an ECG recording, where the pattern correlates with the heart condition's state. Usually, medical experts will determine the state of a patient's heart condition by the shape or morphology of the ECG waves. However, manually determining the pattern is challenging and laborious for professionals, especially for long ECG recordings. Moreover, the human eye can be inappropriate for detecting the morphological variation of the ECG waves. Thus, the use of computational techniques for automatic classification is needed.

The benefit of an automated heartbeat classifier combined with a wearable heart sensor device enables real-time detection of abnormalities in our heartbeats. The Association for the Advancement of Medical Instrumentation® (AAMI) defines heartbeats into five classes [26]. As shown in Table 2.3, those beats are categorized as normal (N), supraventricular ectopic beat (SVEB), ventricular ectopic beat (VEB), fusion beat (F), and unknown beat (Q). Among all classes, SVEB and VEB are categorized as problems in our heart condition, where VEB is related to heart failure [27] and SVEB is related to atrial fibrillation [28].

A comprehensive survey on heartbeat classification using machine learning was presented by Luz [29] while using deep learning was presented by Ebrahimi [30]. One of the differences between classification using machine learning and deep learning methods is feature extraction. Deep learning offers automatic feature extraction, while machine learning mainly uses handcrafted features. The reports of automatic heartbeat classification are varied. Some use different classes and databases, thus leading to unfair comparison—unfortunately, only a few follow AAMI recommendation [6]. The Automated heartbeat classification requires several features to distinguish between normal and abnormal beats. Those features are extracted from electrocardiography recordings, such as the RR interval series, the morphology of ECG waves, and wavelets. Afterward, a machine learning or deep learning method was used as a classifier.

Lin [31] explored the combination of a normalized RR interval and morphological ECG waves as features. It used the linear discriminant to classify normal, supraventricular, and ventricular beats. As a result, normalized RR intervals increase the classifier's performance. Tsipouras uses 3 RRi features (R1, R2, and R3); thus, the rule-based and deterministic automaton is used to classify normal, premature ventricular contraction, ventricular flutter/fibrillation, and two heart blocks [32]. Lian uses a method to map RR interval to detect atrial fibrillation [33]. Xiang uses CNN as feature extraction to obtain time intervals between two RR intervals and morphological features as one-dimensional data, thus using a multi-layer perceptron(MLP) to classify VEB and SEB [33]. Sannino uses RR interval features consisting of previous RR, post RR, local average within 10 seconds slidings from the previous window, and average 10 RR interval window within 5 minutes. They use ANN as a binary classifier to predict normal and abnormal beats [34]. Ankita uses R-peak and RR interval as a feature and uses hybrid CNN to classify 16 classes of heartbeat [35]. Jose did an investigation of feature selection for heartbeat classification. He suggests that using normalized RR intervals could increase the classifier's performance [36]. Mondejar uses features such as RR interval, normalized RR interval, high order statistic, HBF coefficients, and wavelet transform, thus using a support vector machine (SVM) to classify each feature [37]. Developing automatic heartbeat classifica-

tion systems on resource-constrained devices is challenging, e.g., discovering an optimal mixture of features and classifiers [36].

Table 2.3: AAMI recommendation of heartbeats

Normal (N)	**Supraventricular Ectopic Beat (SVEB)**	**Ventricular Ectopic Beat (VEB)**	**Fusion Beat (F)**	**Unknown Beat (Q)**
Normal beat (N)	Atrial premature beat (A)	Premature ventricular contraction (V)	Fusion of ventricular and normal beat (F)	Paced beat (/)
Left bundle branch block (L)	Aberrated atrial premature beat (a)	Ventricular escape beat (E)		Fusion of paced and normal beat (f)
Right bundle branch block (R)	Nodal (junctional) premature beat (J)			Unclassified beat (Q)
Atrial escape beat (e)	Supraventricular premature beat (S)			
Nodal junctional escape beat (j)				

2.5 MIT-BIH Arrhythmia Database

MIT-BIH Arrhythmia Database contains 48 recordings with 30 minutes of ECG records for each subject. Those records are 100, 104, 108, 113, 117, 122, 201, 207, 212, 217, 222, 231, 101, 105, 109, 114, 118, 123, 202, 208, 213, 219, 223, 232, 102, 106, 111, 115, 119, 124, 203, 209, 214, 220, 228, 233, 103, 107, 112, 116, 121, 200, 205, 210, 215, 221, 230, 234. The frequency sampling used in this ECG recording is 360 Hz. Figure 2.4 shows the stub of the ECG signal from record number 116. This database consists of two signals upper signal and a lower signal. The upper signal uses limb lead II and modified lead II (MLII) in the chest area. Normal PQRS usually uses this kind of signal. The second signal is V1 obtained with lead V1 located in chess. This study uses a signal from MLII because normal QRS complexes are usually prominent in this signal. All the ECG recordings are already annotated by cardiologists [38]. Thus, to obtain the value of MLII from this database, I use the WFDB software package.

In 1949, the US Bureau of Standards issued a statement and a paper by Gilford recommending that ECG recordings have a sampling rate of no more than 200 samples per second for practical purposes. Einthoven analyzed QRS morphology recorded at various sampling rates and found no added utility in capturing ECG signals at sampling rates

greater than 100 samples per second for practical applications [39].

Table 2.4: Sample information of record 116

Parameter	value
frequency sampling	360
signal length	650000
number signal	2
base data	none
base time	none
units	mV, mV
signal name	MLII, V1
comments	’68 M 1453 1629 x2’, ’None’, ’There are two PVC forms.’
sample signal MLII	-0.32, -0.32, -0.32 . . . -0.975, -0.96, -1.28

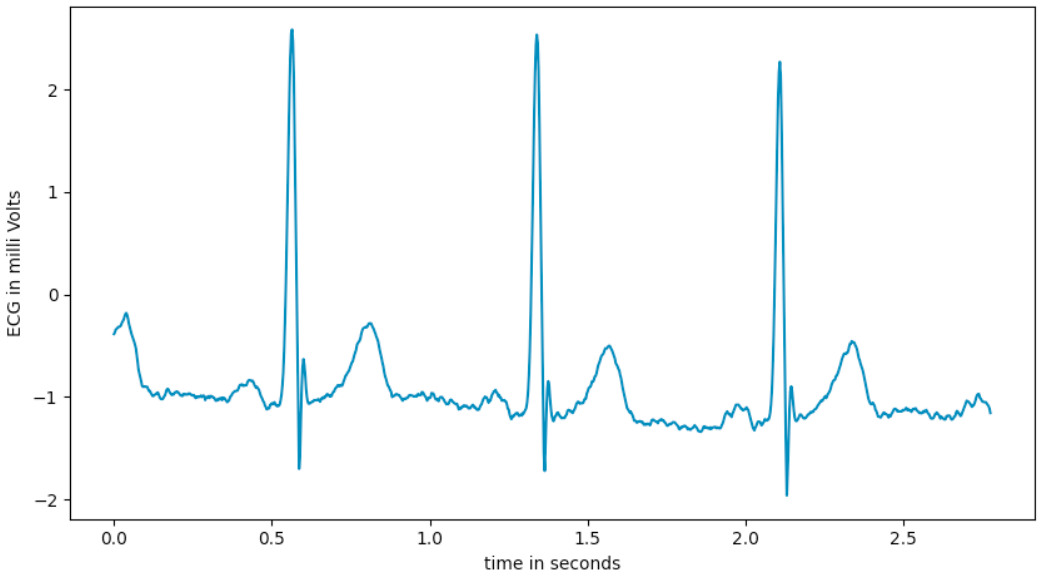

Figure 2.4: Sample of plotting signal 116

Chapter 3

Investigation of Wearable Sensor Devices and BLE Frameworks for Continuous Data Retrieval

This chapter presents the things and middleware domain of our proposed systems. Moreover, the detail of the sensor and middleware is discussed. In this study, things are a Polar H10 as a sensor device, while middleware is an application based on the BLE framework, which runs on a desktop computer.

3.1 Things

Currently, several flexible ECG equipment is available as wearable devices such as chest traps, fitness devices, smartwatches, or armbands. Those devices are intended for fitness equipment. Nonetheless, a chest strap such as Polar H10 can replace a Holter monitor to record cardiac activity [3]. This device is better than a Holter monitor for measuring the RR interval of a person's heart rate and RR interval while they are moving, running, cycling, swimming, and other activity at the gym [4]. A coin battery powers it for up to 30 hours of active usage. While being used on our chest, it does not affect our movement. Polar H10 is also equipped with Bluetooth Low Energy (BLE) to interact with other equipment [40]. It can produce several formats of cardiac parameters such as heart rate (HR), RR interval (RRi), and electrocardiogram (ECG) [5]. For those reasons, I use Polar H10 as a health sensor device. As shown in Figure 3.1, the Polar H10 consist of a strap and devices.

Figure 3.1: Polar H10

To obtain data from Polar H10, I use the BLE framework. Polar H10 already defines the universally unique identifier (UUID) to access service, as shown in Table 3.1. For example, to access heart rate and RR interval, UUID 00002a37-0000-1000-8000-00805f9b34fb should be used. Meanwhile, for ECG UUID FB005C80-02E7-F387-1CAD-8ACD2D8DF0C8 is used.

Table 3.1: UUID used in this study

Paremeter	**UUID**
Model number	00002a26-0000-1000-8000-00805f9b34fb
Heart rate measurement	00002a37-0000-1000-8000-00805f9b34fb
Battery level	00002a19-0000-1000-8000-00805f9b34fb
PMD service	FB005C80-02E7-F387-1CAD-8ACD2D8DF0C8
PMD Control	FB005C81-02E7-F387-1CAD-8ACD2D8DF0C8
PMD data	FB005C82-02E7-F387-1CAD-8ACD2D8DF0C8

3.2 Middleware

In this study, I develop a middleware to receive data from Polar H10 and forward data to a cloud application. Moreover, the middleware also performs formatting data to standardize sensor data into JavaScript Object Notation (JSON) format. Figure 3.2 shows or middleware‘s architecture. I provide two interface communications, BLE to receive data and HTTP to deliver data to other applications in cloud or fog computing environments.

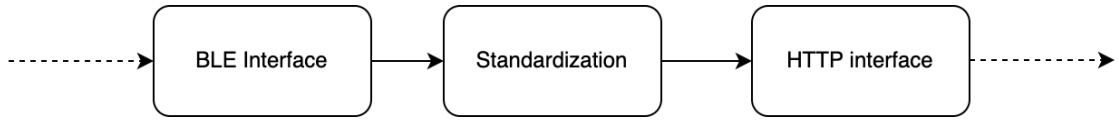

Figure 3.2: Middleware architecture

The BLE interface was developed with a function to connect with Polar H10. Algorithm 1 shows a function on middleware to request RR interval data from Polar H10. Initially, I define all UUIDs required to build a connection at the initialization state. Since the middleware is intended to be able to work in several operating systems, I defined two kinds of ways to address the sensor. The first address uses the MAC address used by Windows and GNU/Linux operating systems, while macOS uses UUID to address the devices due to security reasons. Thus a function provided by the BLEAK framework is used to connect middleware with Polar H10. As shown in the Algorithm 1 line 9, the function defines how long the connection should be made. In the example, state 60 means the connection will stay active until 60 seconds; after that, the middleware will send a request to

stop the connection. The mechanism to access ECG data from polar H10 slightly differs from RR interval data. It is required to ask for PMD control as shown in Algorithm 1 line 12.

After receiving data from Polar H10, regardless of RR interval or ECG data, I need to convert the value from byte to float. Algorithm 2 shows how to covert RR interval value, while Algorithm 3 converts ECG value. After getting the value, thus, data is standardized using JSON format as shown in Figure 3.3. The standardizing is done by adding some information to the data, which I call a payload. The payload consists of information regarding the name of the sensor's device, time, topic, and sensor data. I use the topic to differentiate one sensor's data from another. An example topic used in this study is $/RR\_pramukantoro$. The last function in our middleware is forwarding data to other applications, in Algorithm 4 shows the data will be delivered to the application on apps.belajardisini.com through web service.

Algorithm 1 BLE Connection RR interval and ECG

1: **Initialize:**
   $MODEL\_NUMBER \leftarrow 00002a26-0000-1000-8000-00805f9b34fb$,
   $HR\_MEASUREMENT \leftarrow$
   $00002a37-0000-1000-8000-00805f9b34fb$,
   $BATTERY\_LEVEL \leftarrow 00002a19-0000-1000-8000-00805f9b34fb$,
   $PMD\_SERVICE \leftarrow$
   $FB005C80-02E7-F387-1CAD-8ACD2D8DF0C8$,
   $PMD\_CONTROL \leftarrow$
   $FB005C81-02E7-F387-1CAD-8ACD2D8DF0C8$,
   $PMD\_DATA \leftarrow FB005C82-02E7-F387-1CAD-8ACD2D8DF0C8$
2: **if** $platform.system! = DARWIN$ **then**
3: $ADDRESS \leftarrow 3199F0CD-316C-4D6E-BD21-936AA57C8B3F$
4: **end if**
5: **function** BLEAKCLIENT(ADDRESS)
6: $model\_number \leftarrow$ await $client.read\_gatt\_char(MODEL\_NUMBER)$
7: $battery\_level \leftarrow$ await $client.read\_gatt\_char(BATTERY\_LEVEL)$
8: awaits $client.start\_notify(HR\_MEASUREMENT, ConvertData)$
9: awaits $asyncio.sleep(60.0)$
10: awaits $client.stopnotify(HR\_MEASUREMENT)$
11: **end function**
12: **function** RUN(ADDRESS)
13: $model\_number \leftarrow awaitclient.read\_gatt\_char(MODEL\_NUMBER)$
14: $battery\_level \leftarrow awaitclient.read\_gatt\_char(BATTERY\_LEVEL)$
15: awaits $client.write\_gatt\_char(PMD\_CONTROL, ECG\_WRITE)$
16: awaits $client.start\_notify(PMD\_DATA, data\_conv)$
17: awaits $asyncio.sleep(3.0)$
18: awaits $client.stop\_notify(PMD\_DATA)$
19: **end function**
20: running loop of $asyncio.get\_event\_loop()$
21: $loop.run\_until\_complete(run())$

Algorithm 2 convert byte to RR interval data

```
function CONVERTDATA(sender, value)
    byte0 ← value[0]
    res ← {}
    res["hrv_uint8"] ← (byte0 ∧ 1) = 0
    res["ee_status"] ← (byte0 ≫ 3) = 1
    res["rr_interval"] ← (byte0 ≫ 4) = 1
    if then res["hrv_uint8"]:
        res["hr"] ← value[1]
        i ← 2
    else:
        res["hr"] ← (value[2] ≪ 8) ∨ value[1] i+ = 2
    end if
    if then res["rr_interval"]:
        res["rr"] ← []
        while do i < len(value)
            res["rr"].append((value[i+1] ≪ 8) ∨ value[i])
            i+ = 2
            RRIval ← str((res["rr"]))[1 : −1]
            if then","inRRInterval:
                a ← RRIval.split(",")
                data1, data2 ← a[0], a[1]
                RRI.append(data1)
                RRI.append(data2)
            else:
                RRI.append(RRIval)
                payload ← {'device' : device, 'time' : time, 'topic' : topic, 'payload' : { 'rri' :
RRIval}}
            end if
        end while
    end if
end function
```

Algorithm 3 Convert byte to ECG data

```
function CONVERTDATA(sender, value)
    if then value[0] == 0x00:
        timestamp ← convert_to_unsigned_long(data, 1, 8)
        step ← 3
        samples ← data[10 :]
        offset ← 0
        while do offset < len(samples):
            ecg ← convert_array_to_signed_int(samples, offset, step)
            offset+ = step
            ecg_session_data.extend([ecg])
            ecg_session_time.extend([timestamp])
            payload ← {'device' : device, 'time' : time, 'topic' : topic, 'payload' : {'ecg' :
float(ecg)}}
        end while
    end if
end function
```

Algorithm 4 Sending sensor‘s data

Initialize:
$HOST \leftarrow apps.belajardisini.com$,
$PORT \leftarrow 80$,
$DEVICE \leftarrow 45HOST : 6s : ec : re : t1 : 23$,
$TOPIC \leftarrow /RR\_pramukantoro$
, $TOKEN \leftarrow eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9\ldots$

function SENDDATA(payload)
    try
        $r \leftarrow requests.post(\text{"}http://HOST:PORT/gateway/api/post\text{"}), payload, headers,$
        $Authorization$
    catch Expected exception as error
        Resend data
    end try
end function


```
{
    "_id": "62a6ba31c5c22df3e1d8eaab",
    "device": "po:la:r7:53:eb:c2",
    "payload": {
        "rri": "594"
    },
    "time": "2022-06-13 - 13:16:49.390087",
    "topic": "/RR_pramukantoro"
}
```


Figure 3.3: Format data in JSON

3.3 Experiment on Receiving Data from Sensor to Middleware

To ensure the middleware works as intended, I conducted experiments based on developed middleware to obtain RR interval and ECG data from Polar H10. The following are samples of data received by middleware:

1. Sample of received RR interval data
   -33,-62,-52,-62,-81,-59,-26,-11,-4,-9,-45,-69,-67,-79,-103,-117,-129,-129,-107,-79,
   -59,-69,-112,-155,-162,-148,-141,-139,-127,-119,-131,-131,-110,-100,-119,-136...
2. Sample of received ECG data
   -33,-62,-52,-62,-81,-59,-26,-11,-4,-9,-45,-69,-67,-79,-103,-117,-129,-129,-107,-79,
   -59,-69,-112,-155,-162,-148,-141,-139,-127,-119,-131,-131,-110,-100,-119,-136...

Using our middleware, data from Polar H10 are sent continuously for as long as desired. Figure 3.4 shows the plotting of the ECG signal obtained by Polar H10. Polar H10 uses High amplitude. Regarding the ECG wave or ECG morphology, the PQRS shape is easily observed.

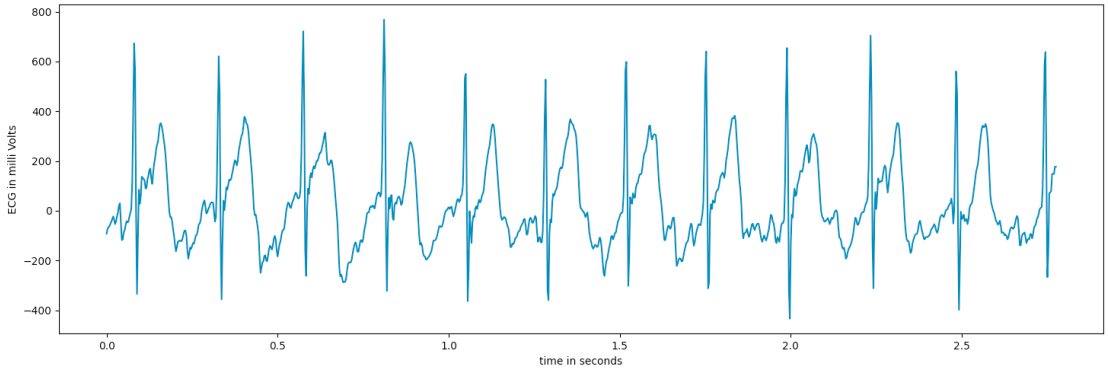

Figure 3.4: Plotting Polar H10

3.4 Experiment on The Middleware‘s Performance

This subsection presents the evaluation result of the Polar H10 as a vital sign sensor device and BLE frameworks. Three application that communicates with Polar H10 is developed. One is based on Core Bluetooth, the second is based on BLEAK, and the Third is based on PyGATT. Those programs represent a multi-environment operating system. The evaluation of this study is done by observing parameters such as compatibility of operating system and hardware, sensor data, connection mechanism, and reachable range. Core Bluetooth and BLEAK were implemented using Mac Book Pro hardware with OS version 10.15 and PyGATT on a Desktop computer that runs Ubuntu 20.04. Later on, this program is called the Central. Those programs equipped functions to communicate with Polar H10 and send data to cloud applications through a restful web service. The programs initially scan the LE advertisements that contain information on the GATT characteristics of heart rate measurement (HRM). This information is coded into 16-bit UUID numbers and regulated by Bluetooth SIG as 0x2A37. In this way, the Central will find any Peripheral that offers heart rate measurement service, i.e., Polar H10. Afterward, a connection between Central and Peripheral can be made by specific identifiers like name or mac address.

Regarding communication capability, the maximum communication range may vary depending on the environment. According to the Polar documentation, this device works optimally in the range of 5 meters. However, it can work in various ranges between 10-35 meters and reach 100 meters [41] in the open field. Instead of spatial distance, the Received Signal Strength Indicator (RSSI) can be used as an index to measure distance in BLE. RSSI is the signal strength received by the Rx node that comes from the Tx node. The optimum RSSI for reliable delivery is -70 dB, while lower than -80 dB is not recommended [25].

3.4.1 Performance on Hardware Compatibility

Table 3.2 shows program development's operating system and hardware compatibility results. The result on Mac Book Pro (13-inch, 2020) is unstable; the program runs well once the computer turns on. On the second run, the program cannot receive any advertisement message from Polar H10. It is caused by sharing hardware and resources with Wi-Fi called the Hand-off mechanism [42]. The solution is to add BLE hardware by using a USB dongle. Then the program works properly on Mac Book Pro (13-inch, 2020).

Regarding operating system compatibility, BLEAK outperforms other BLE frameworks. BLEAK can run smoothly on multi-operating systems. Based on this reason, the further study, I employ BLEAK as the BLE framework in our middleware.

Table 3.2: Device Compatibility

BLE frameworks	Device	Result
PyGATT	Mac Book Pro (13-inch, 2020), macOS 10.15	Unstable
PyGATT	Mac Book Pro (13-inch, 2017), macOS 10.15	Stable
PyGATT	Mac Book Pro (15-inch, 2019), macOS 10.15	Stable
PyGATT	Desktop computer, Ubuntu-20.04 LTS	Stable
BLEAK	Mac Book Pro (13-inch, 2020), macOS 10.15	Stable
BLEAK	Desktop PC, Ubuntu-20.04 LTS	Stable
BLEAK	Mac Book Pro (13-inch, 2017), macOS 10.15	Stable
BLEAK	Mac Book Pro (15-inch, 2019), macOS 10.15	Stable

3.4.2 Performance in Line-of-Sight Environment

To evaluate the communication range in BLE, indices of both distance and RSSI is used. While using the distance as an index, measurement is done by manually changing the distance of one device from another, i.e. from 0 meters until communication does not occur. While using RSSI as the index, the measurement was done by calculating signal strength received by the central (receiver). For the measurement of communication range, the distance between one central and one peripheral was deployed in an obstacle environment. As shown in Figure 3.5, this measurement was held at hallway location 9, and the maximum available indoor distance was 50 meters. The results of this experiment are shown in Figure 3.6. The maximum distance for proper communication using Mac Book Pro (13-inch, 2020) with a BLE dongle is 25 meters, with a threshold of tolerable RSSI is -80 dBm. At the same time, optimum communication on Mac Book Pro (13-inch, 2017) can reach 45 meters.

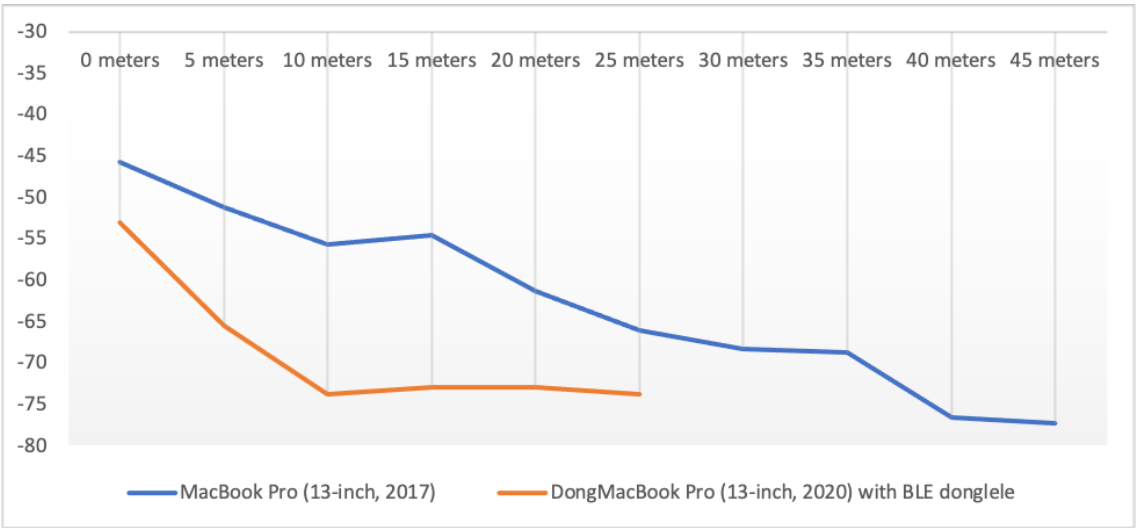

Figure 3.5: The communication range between central and peripheral.

3.4.3 Performance in Complex Indoor Environment

The experiment was conducted in a laboratory room to measure the communication capability of middleware between Polar H10 in a non-line-of-sight (NLoS) environment. Figure 3.6 illustrates a sketch of our laboratory's floor plan. One middleware and nine locations send sensor data to the middleware. I used two kinds of Central in this experiment, the first Mac Book Pro (13-inch, 2017) and the second Mac Book Pro (13-inch, 2020), with a BLE dongle. As the peripheral, I used two Polar H10s worn by different persons. In each location, the user wears the sensor for 5 minutes. The peripheral to the central sequentially range from location 0 to 9 is as follows: 0 meters, 7 meters, 8 meters, 3.5 meters, 12 meters, 14 meters, 15 meters, 7 meters, 17 meters, and 16 meters.

As a result, using Mac Book Pro (13-inch, 2017), data from peripherals at 9 locations can be received by the central. The received RSSI is shown in Figure 3.7. At positions 0 to 7, RSSI is still in a good state, while at positions 8 and 9, the RSSI is bad though the communication still occurs. On the Mac Book Pro (13-inch, 2020), as shown in Figure 3.8, the communication has occurred from position 0 until 3. A Good RSSI only happened at position 0, while the RSSI nearly reached the minimum threshold for good communication in other positions.

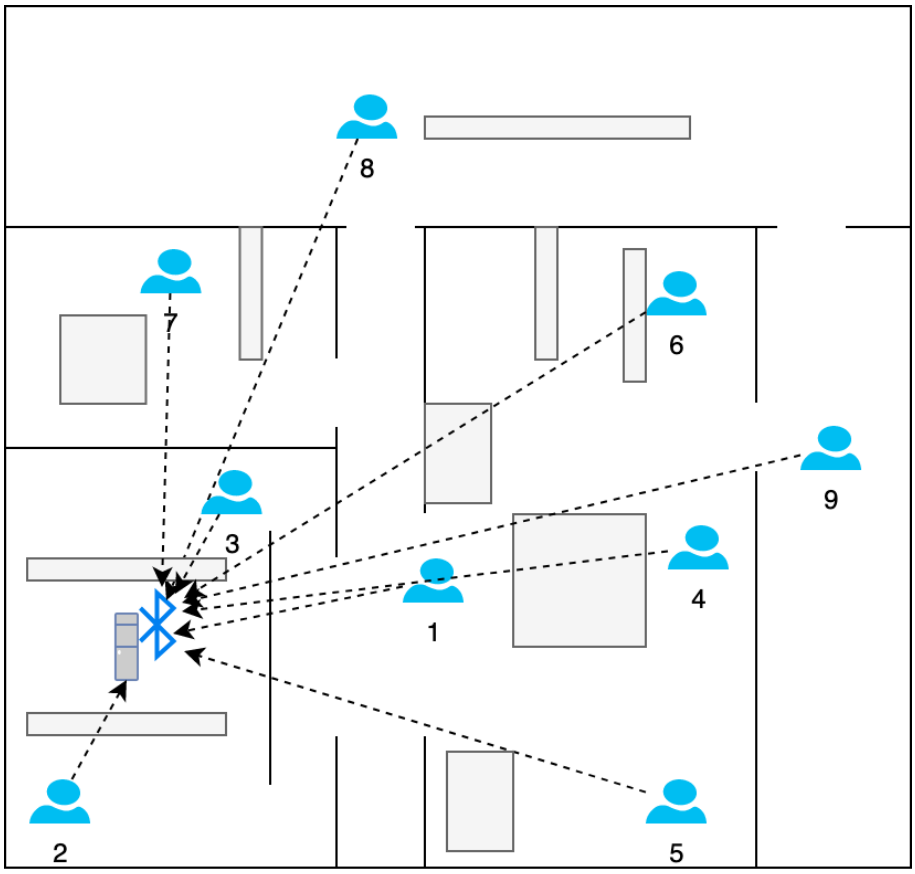

Figure 3.6: An environment of the experiment.

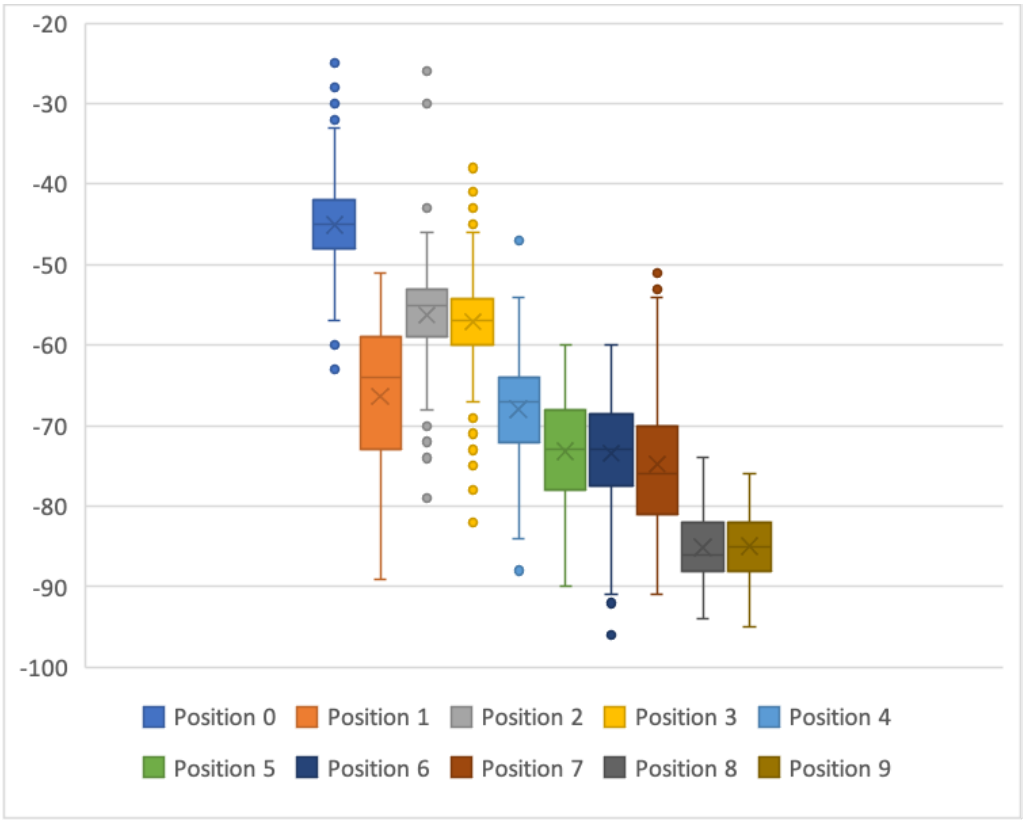

Figure 3.7: The RSSI on Mac Book Pro (13-inch, 2017).

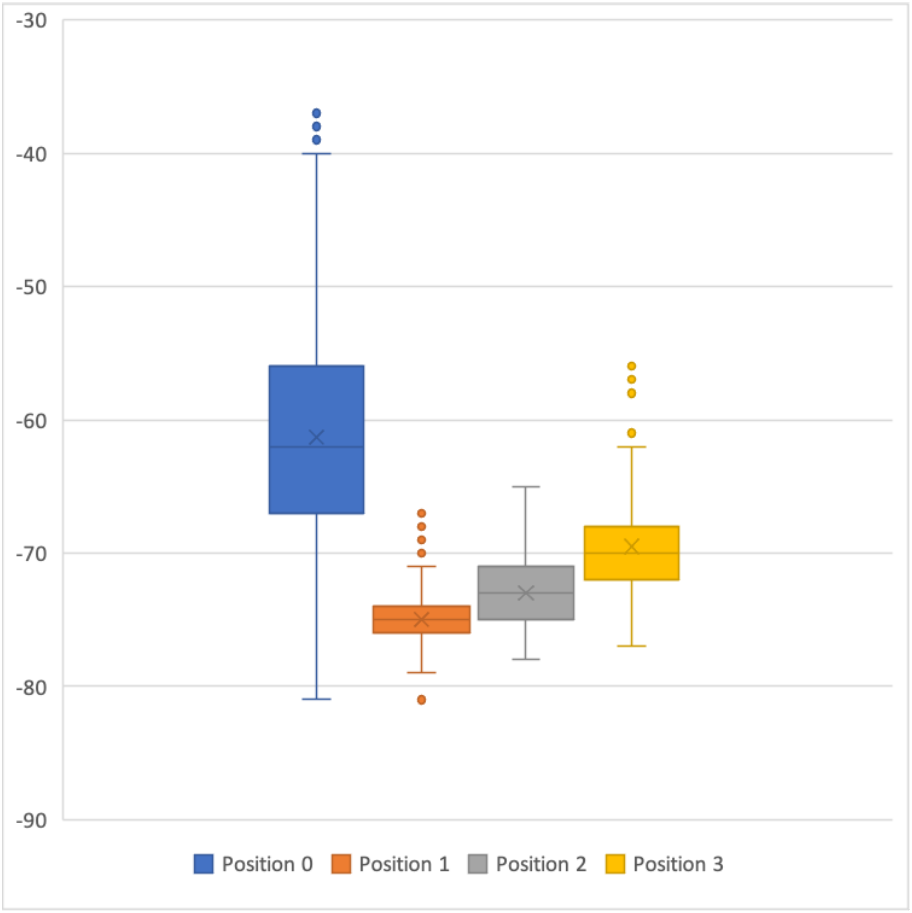

Figure 3.8: The RSSI on Mac Book Pro (13-inch, 2020) with BLE dongle.

3.5 Experiment sending data from middleware to cloud application

In this section, I present the result of an experiment in continuously delivering sensor data to the cloud application. Our cloud application runs on a virtual private server located in apps.belajardisini.com. The experiment involved a healthy person who wore a Polar H10. During one h, middleware requests RRi data, while middleware requests ECG data for the next hour. As the result shown in Figure 3.9 and Figure 3.10. The middleware will send data after receiving data from Polar H10. During one h, Polar h10 produces 7152 RR interval data and is sent every second. Sometimes in one second, two or more RR interval data are delivered. The experiment concluded there is no problem delivering RR interval data by middleware to the cloud. However, there is a bottleneck in delivering ECG data during one h. Polar H10 produces more ECG samples than the middleware's capability to send data to cloud applications in one second. As shown in Table 3.3 in one second, Polar H10 produces 73 items or samples of ECG data. Thus delivering ECG data continuously is not recommended. It must store ECG data before being sent to the cloud application. This kind of way will become batch processing instead of real-time processing.

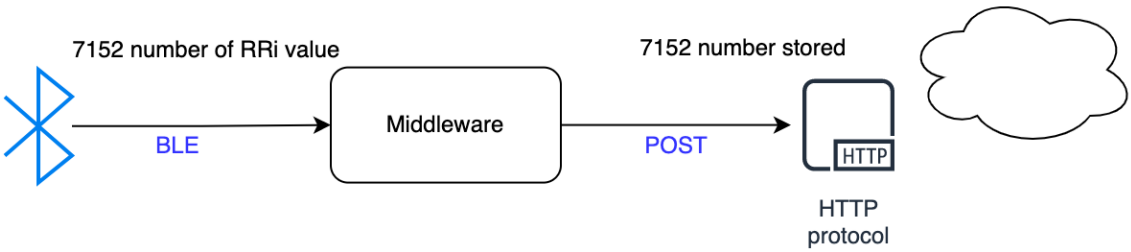

Figure 3.9: RRI 1 hour recording real-time recording

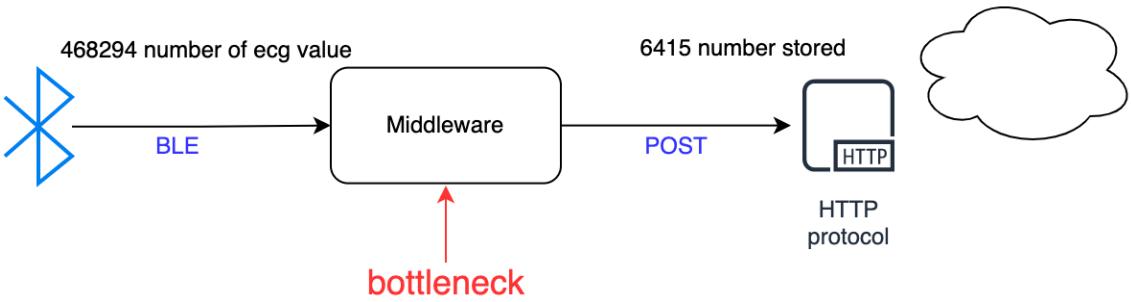

Figure 3.10: ECG 1 hour recording real-time recording

Table 3.3: Comparison number of sensor‘s data based on ECG and RR interval

Time	**ECG data**	**RR interval data**
1 minutes	7738 items	134 items
1 second	73 items	3 items
2 second	219 items	4 items
2 second	365 items	7 items

3.6 Summary

This chapter discussed our proposed systems' Things and middleware domain.

Chapter 4

A Heartbeat Classifiers for Continuous Prediction Using a Wearable Device

In this chapter, I present several heartbeats classifiers proposed in this study. Starting from a classifier based on heart rate variability (HRV), RR interval, and ECG morphology features. Regarding the feature for training, the classifier is adapted to the Polar H10 output data types: RR interval and ECG data.

4.1 Heart Rate Variability-based Classifier

In heart rate variability (HRV) measurement, a Holter monitor is typically used to observe ECG data. However, since this device lacks mobility, alternative wearable devices with a similar capability of recording cardiac data are preferred. On the other hand, the application provided alongside wearable devices are mostly installed in smartphone environments, which cause significant concern for users' privacy. It is better to develop our application to calculate sensitive data.

This section presents the proposed application that utilizes a commercialized wearable wireless device (Polar H10) to collect RR interval and ECG data to perform an HRV measurement using the time, frequency, and nonlinear domains. Furthermore, I also examine RR interval and ECG data produced by Polar H10. This study was done by prototyping and experimenting involving participants in a laboratory environment.

4.1.1 Obtaining Heart Rate Variability Data

The heart rate variability (HRV) measurement shows the condition of cardiac activity using a device to record electrocardiography (ECG) data, then calculates its value using several methods in HRV. To do that, using A Holter device is typical because of its ability to produce gold standard ECG data. However, this device is expensive for personal use and can burden high mobility. A wearable wireless ECG electrode is another alternative device to record ECG data. However, an application is necessary for conducting data analysis based on HRV measurement.

Nowadays, a wearable wireless device for recording ECG data is available at a reasonable price for personal use. Among all those devices, Polar H10 appears to be the most accurate and precise in continuously (>24 hours) doing HRV measurement [22]. Excellent work from Gilgen [3] compared RR interval data from Polar H10 with Medilog®

AR12plus as Holter monitor. The result shows the quality of RR interval produced by Polar H10 is valid, and they suggest the chest strap as the gold standard in measuring heart activity. Additionally, wireless devices available in the market are already equipped with applications to process data. i.e., fitness application. Some of them open their software development kit (SDK) to enable others to develop a custom application for their purpose. However, most of the provided applications are available on smartphones.

A smartphone-based application has limited power and computational resources and is not multi-platform. A program that runs on desktop environments can be proposed as the solution. Instead of using their SDK, using BLE frameworks, desktop-based applications can communicate to the wearable wireless device as a sensor like Polar H10 [8]. Furthermore, there is a concern about data privacy if using a provided application from a commercialized ECG wearable device [22]. Connecting a vital sensor to its developed or open-source application is better. To develop the own application to retrieve data from the Polar H10 device, the developer needs to choose one of two kinds of data offered by the Polar H10 device: a heart rate plus RR interval and ECG raw data. Both of them can collect through BLE communication. The question is, are data from RR interval and ECG data equal?

This study tried to answer the privacy concern and multi-environment application and examine both data produced by the Polar H10 sensor. An application based on Python was proposed. This application uses Bleak as a BLE framework to create communication with Polar H10 using BLE protocol communication. Moreover, an HRV measurement was provided to process the received data from the sensor [43]. And also, a T-test was conducted to find equality of RR interval data with ECG raw data from Polar H10.

According to the literature, [44], HRV measurement consists of three methods: time domain, frequency domain, and nonlinear. Those methods calculate based on the statistical approach of RR interval that is already filtered and becomes NN interval. As shown in Figure 4.1, the RR interval is an interval between the QRS complex in continuous ECG data. The first method has several parameters, such as the standard deviation of the NN interval (SDNN). The second parameter is the average RR interval (SDANN) standard deviation. This parameter was used to calculate the average standard deviation on 5 minutes segment on the 24-hour recording. The third is the root mean square of adjacent NN interval (RMSSD), NN50, the difference of interval between NN longer than 50 milliseconds, to distribute the density of NN interval into a geometric pattern. The standard deviation of the successive difference of RR interval (SDSD).

Power spectral density (PSD) was used in the frequency domain to gain information on the frequency component. Variables in the frequency domain are high frequency (HF) (0.14 and 0.40 Hz), low frequency (0.04 and 0.15Hz), and very low frequency (0-0.04Hz). LF reflects the activity of PNS, while LF is SNS. The last method is nonlinear, and this method is calculated using the Poincaré plot to visualize the RRI interval by SD1 and SD2 parameters. SD1 indices the sharp fluctuation of HR, while SD2 correlates with a short and long flux of HR.

Among those three methods, the time domain is best for long-term recording, while the frequency domain for short recording while nonlinear is rarely used. Regarding the interpretation of HRV measurement on health condition, several parameters to indicate the acute condition was described by SDNN$<$50 ms and HRV triangular index $<$ 15, SDNN¡100ms, and HRV triangular index $<$ 20 [45].

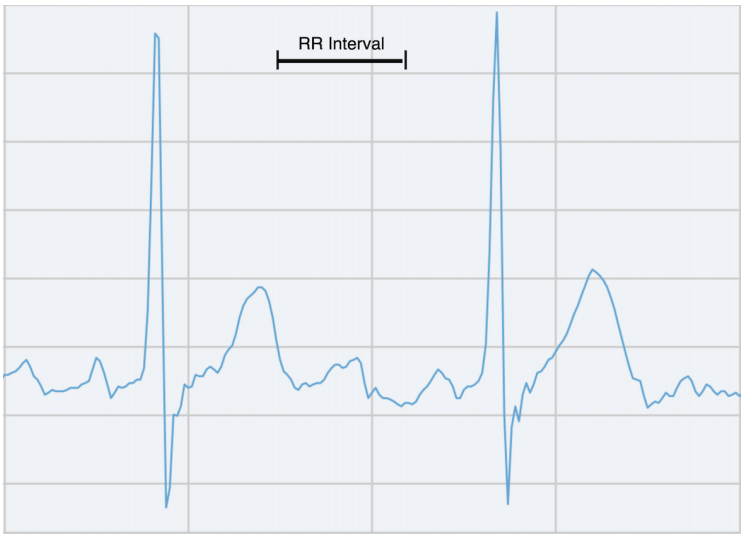

Figure 4.1: The RR Interval series

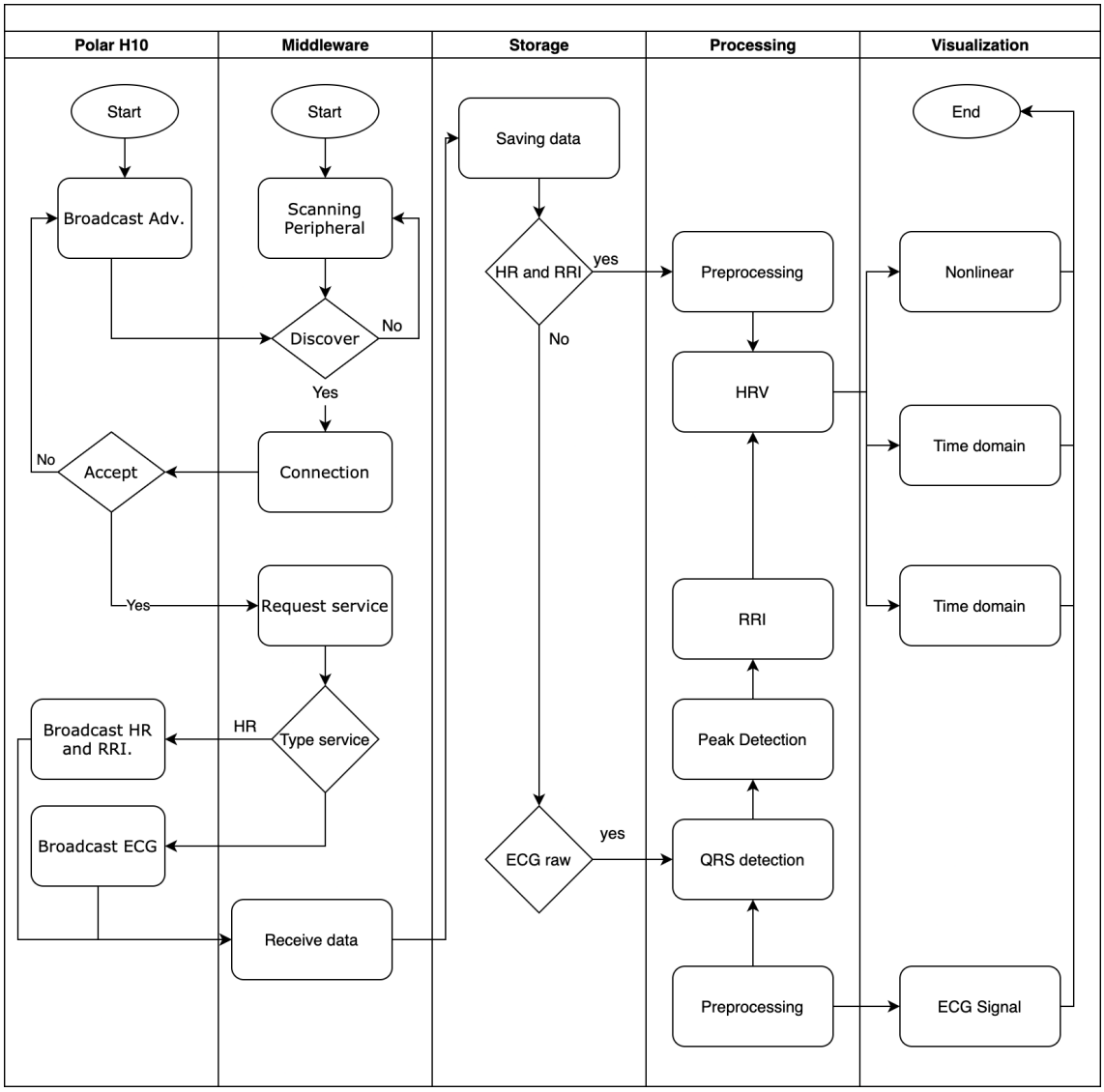

Figure 4.2: A system architecture of HRV application

4.1.2 Prototyping and Experiment

This study was done by prototyping an application and conducting an experiment. As shown in Figure 4.2, the prototype applications consist of several subsystems. As a health sensor, Polar H10 was used. This sensor has a gold standard for measuring cardiac activity. For middleware, I developed a program based on the Python programming language. Inside the middleware, several functions are provided. The primary function is making communicate with the sensor device. This function was built from the BLEAK module. BLEAK was chosen because of its capability to run on several operating systems. Other functions are standardizing data from the sensor nodes and the function to save received data into a storage system. After data is successfully stored, data are processed in the data process subsystem. Two functions based on HRV measurement are provided, first for processing data based on RR interval and processing data based on ECG raw data. Compared to ECG data, data already in the RRI series are much easier to process. At the same time, ECG data need several processes before an HRV measurement is conducted. From signal filtering, detection QRS complex, peak detection, and the last is measure RR interval [46]. Several authors had proposed a method for prepossessing ECG data; the Hearthpy module was employed in this study. After RR interval data is available, the HRV methods can be calculated. HRV measurement consists of time domain, frequency domain, and nonlinear. The last subsystem is visualization; a mat-plot module was used to visualize computed data.

An experiment was done involving a healthy participant wearing a Polar H10 device in the Laboratory room. In this study, data are collected in short-time windows, which means data are collected every 5 minutes using a developed application. At the same time, the participant does a study on his desk. Both RR interval and ECG data were collected separately from the same participant.

4.1.3 Experiment on HRV Data

The developed application can run in Linux, macOS, and Windows. Thus, by using this application, there's no limitation on the operating system. During collecting data from the participant, measurement of RSSI was also conducted. The RSSI parameter is used for determining the quality of data transmission. Based on our experiment, the average RSSI is -45 dB with a standard deviation of 5, which means the quality during data transmission is good, and the possibility of data loss is low. The maximum of RSSI for indicating good transmitting data is 80 dB.

After data are collected, the HRV measurement function is conducted. The HRV measurement function works with both RR interval data and ECG data. As a result, in Figure 4.3, information based on time-domain variables such as means of RR interval is 775.25ms, SDNN is 440.72ms, RMSSD is 18.26ms, and NN50 is 4). Based on the time domain result, the participant can be concluded as having a health condition because the unhealthy state describes SDNN below 100ms [45]. Figure 4.4 shows HRV measurement results using frequency-domain variables consisting of VLF, VF, and HF. The frequency domain was computed using Welch's method. The peak frequency of VLF is 0.015Hz, while the peak frequency of LF is 0.054Hz, and the peak frequency of HF is 0.0183Hz. According to the literature, low VLF is associated with heart condition problems. A mental condition of a participant can be defined using a lower value of HF, i.e., worry or stress. At the same time, LF represents resting conditions. Figure 4.5 shows the result of a popular method to describe the nonlinear domain in HRV. In this domain, the values of

two variables are SD1 with 14.340ms and SD2 with 63.847ms. This map plots the entire RR interval data. SD1 is the standard deviation of axis x1 and expresses the circle's width, while SD2 is the standard deviation of x2 and expresses the circle's length. The ratio SD1-to-SD2 is 0.225 and correlates with low-frequency/High-frequency (LF/HF) in the frequency domain. The eclipse relates to the root mean square of successive differences between normal heartbeats (RMSSD) variables in the time domain.

Algorithm 5 HRV feature extraction

Inputs:
RR interval
Outputs:
FD, TD, NL
function FREQUENCY_DOMAIN(RR interval)
$FD \leftarrow pyhrv.frequency\_domain.welch\_psd(RRinterval)$
return FD
end function
function TIME_DOMAIN(RR interval)
$TD \leftarrow pyhrv.time\_domain.time\_domain(RRinterval)$
return FD
end function
function NONLINEAR_DOMAIN(RR interval)
$NL \leftarrow pyhrv.nonlinear.Poincare(RRinterval)$
return NL
end function

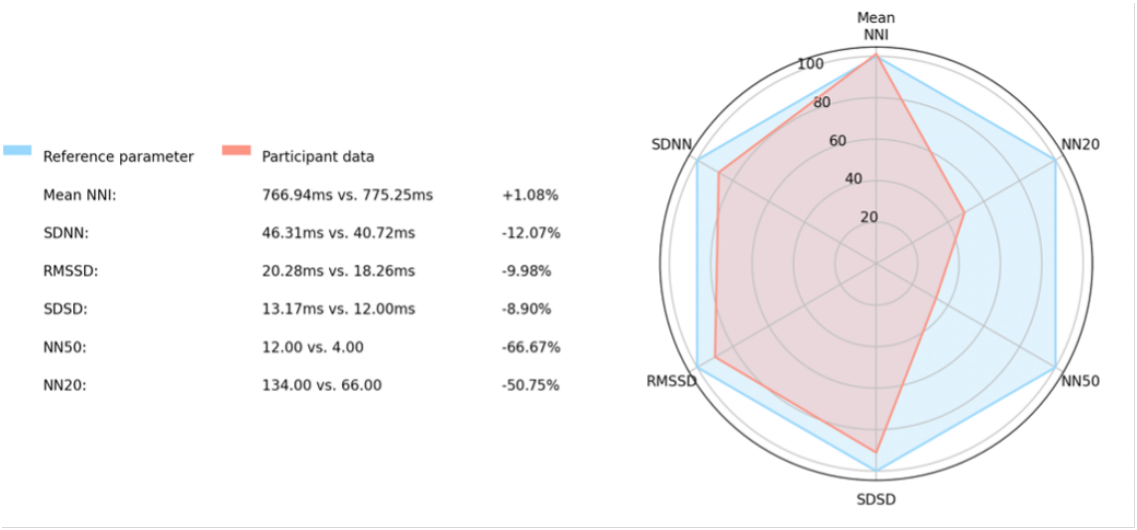

Figure 4.3: A result of time-domain patient data compared to reference data

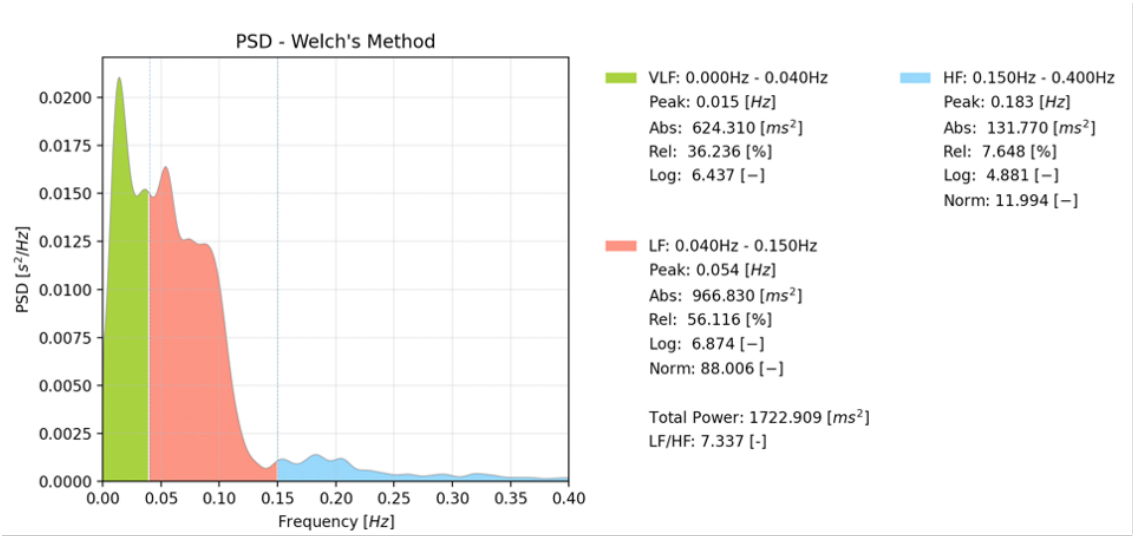

Figure 4.4: A result of frequency-domain calculation using Welch's method

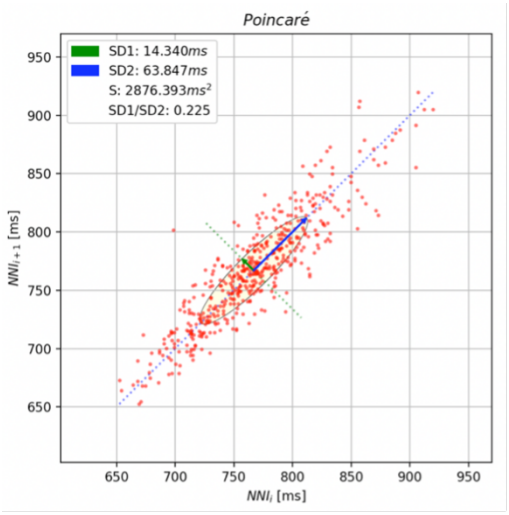

Figure 4.5: A result of the time-domain method using Poincare

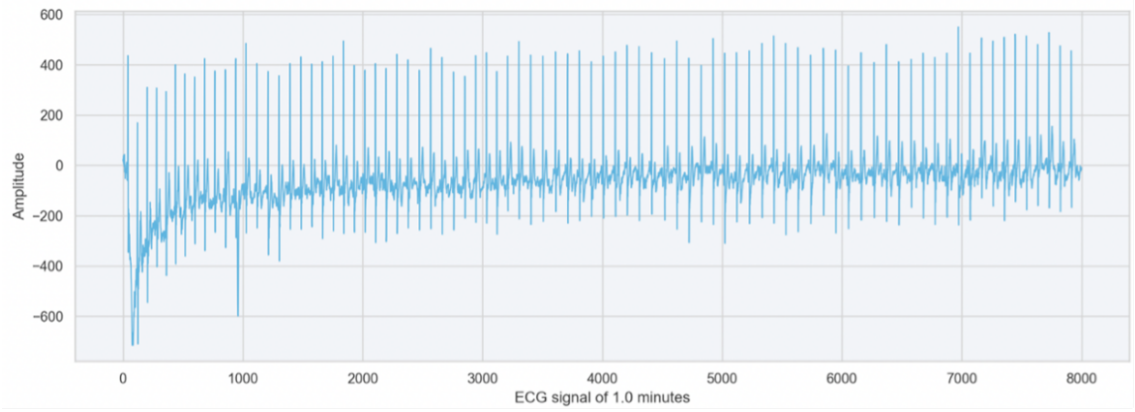

Figure 4.6: A visualization of 1 minute's slice ECG data

Table 4.1: A sample data from two kinds of data in 5 minutes

Types of data	NNI mean	Standard Deviation	Total RR interval
RR interval	766.9	46.2	487
ECG raw	766.9	48.2	391

To answer the equality of ECG and RR interval data for HRV, I collect 5 minutes of data for each. Figure 4.6 visualizes ECG data in 1 minute. From minute 0 until 7.5 seconds, there are some signal noises. Thus, those noises need to be removed. Several steps need to be done to process HRV measurement, from signal filtering, QRS, and peak detection, thus calculating the RR interval.

The p-value in a T-test was used to answer the equality of produced data between RR interval and ECG data. RR interval data was extracted from both data sources before performing the T-test. Table 1 is a summary of two kinds of data. As a result, the p-value is 0.17, which means the p-value is more prominent than 0.05 or 0.1. it can be concluded both data have identical averages. Figure 4.7, Figure 4.9, and Figure 4.8 result from the HRV measurement of ECG data.

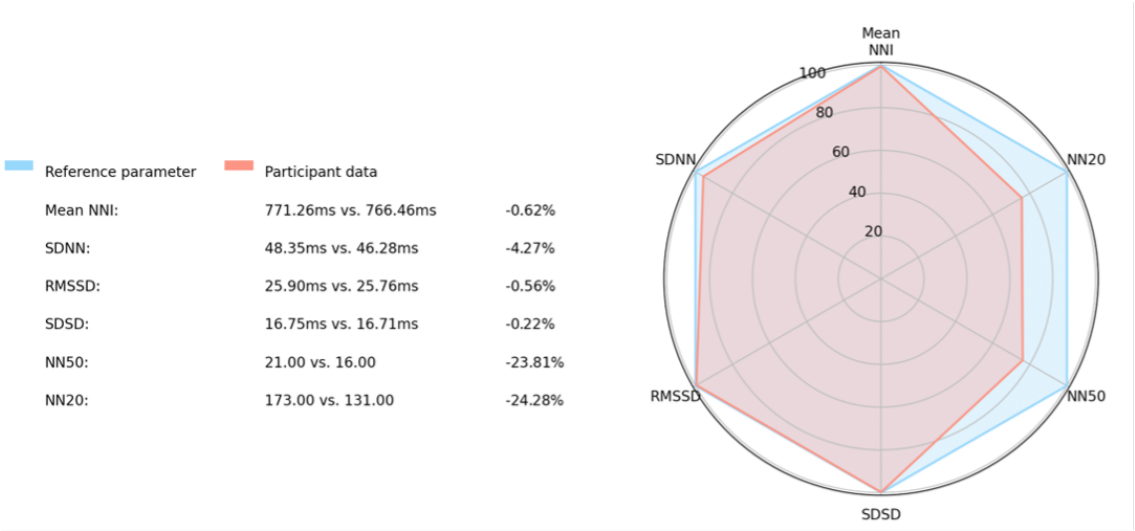

Figure 4.7: A result of time domain of ECG data

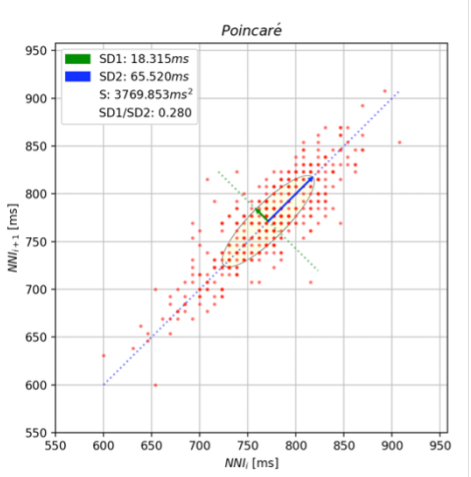

Figure 4.8: A result of frequency domain of ECG data

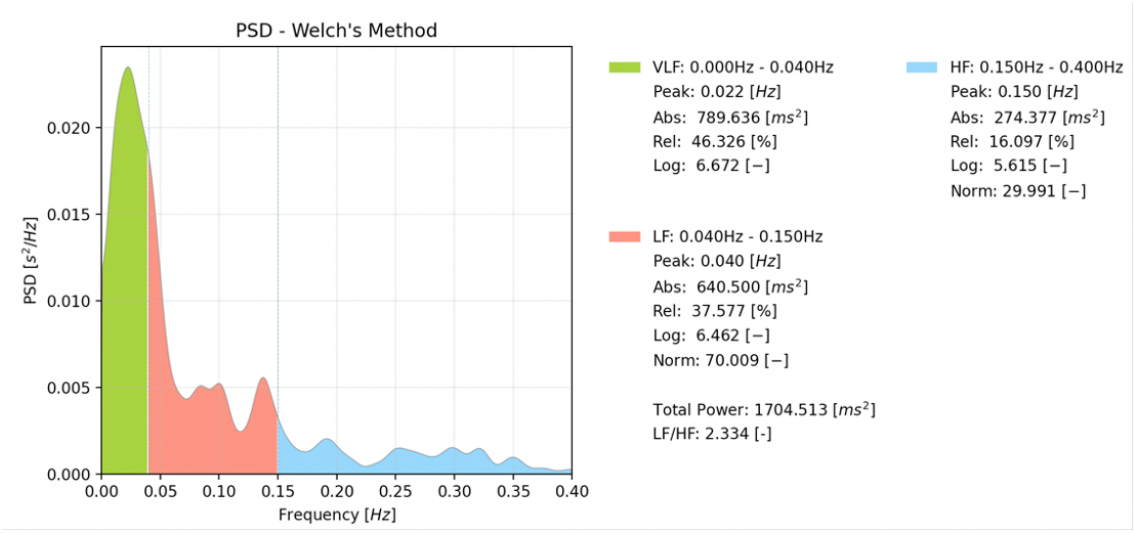

Figure 4.9: A result of a nonlinear domain of ECG data

4.1.4 Train HRV Classifier

Regarding the heart rate variability (HRV) feature for classification, there are three parameters of HRV such as:

1. Time-domain
   (a) The standard deviation of the NN interval (SDNN)
   (b) The standard deviation of the average RR interval (SDANN)
   (c) The root mean square of adjacent NN interval (RMSSD)
   (d) The difference in interval between NN longer than 50 milliseconds (NN50)
   (e) The standard deviation of the successive difference of RR interval (SDSD)
2. Frequency domain
   (a) The frequency-domain are high frequency (HF) (0.14 and 0.40 Hz)
   (b) Low frequency (LF) (0.04 and 0.15Hz)
   (c) shallow frequency (VLF) (0-0.04Hz)
3. Non-linear domain
   (a) SD1 indices the sharp fluctuation of HR
   (b) SD2 correlates with a short and long flux of HR

All those parameters are computed from the RR interval. HRV parameter can be used to define the state condition of our heartbeat. However, using HRV as a classification feature will limit to a normal and abnormal condition. I demonstrate the HRV as a feature for heartbeat classification. I generate HRV feature extraction for classification using a dataset from MIT-BIH arrhythmia (48 records) and MIT-BIH normal sinus (18 records).

From the data in Figure 4.10, I train the classifier using SVM. Afterward, I experimented with automatic prediction using Polar H10 and a classifier based on HRV. The result is shown in Figure 4.11. HRV can be used for heartbeat classification, only limited

to Normal and Abnormal. Thus I give the anomaly (A) label from MIT-BIH arrhythmia and the normal (N) label from MIT-BIH normal sinus.

		Time domain					Frequency domain		Non linear domain			
SEX	AGE	SDNN	SDANN	RMSSD	SDSD	NN50	TOTAL POWER	LF/HF	SD1	SD2	SD1/SD2	LABEL
M	45	139.2493004	94.52828496	129.2250359	126.7980456	3178	6.972933073	103923.6881	91.37589915	174.4431015	1.909071245	N
M	34	140.0115027	81.98404739	109.2124825	103.5851184	9655	6.311608876	61833.09881	77.2248869	182.3208307	2.360907708	N
F	38	119.0460455	63.83198188	121.6317664	119.0158826	5898	12.45664275	230728.6989	86.00664685	144.7297887	1.68277446	N
F	50	260.3416922	65.69730885	355.3341768	354.6947695	511	30.04739387	182701210	251.259206	269.1176111	1.071075625	N
F	66	37.24355178	2.335655936	53.18669311	35.89897287	586	0.052473737	931.8035817	37.60864102	36.76171483	0.977480543	A
F	73	64.81358379	24.31505633	94.94765381	88.22858933	159	0.247204344	2219.007064	67.13812986	62.39768444	0.92939265	A
F	24	260.8162153	59.95705065	434.6554259	308.7121576	1311	0.079902013	19174.24111	307.3477573	203.785919	0.663046709	A
M	63	39.87692903	4.816074059	47.36901794	37.85617683	197	0.182893027	1107.60435	33.49484163	44.67643496	1.333830309	A
F	87	124.0647217	53.77333824	146.8942485	125.5015197	736	0.454551262	8766.475889	103.8699101	141.3555219	1.36089	A

Figure 4.10: Sample data after feature extraction using HRV methods

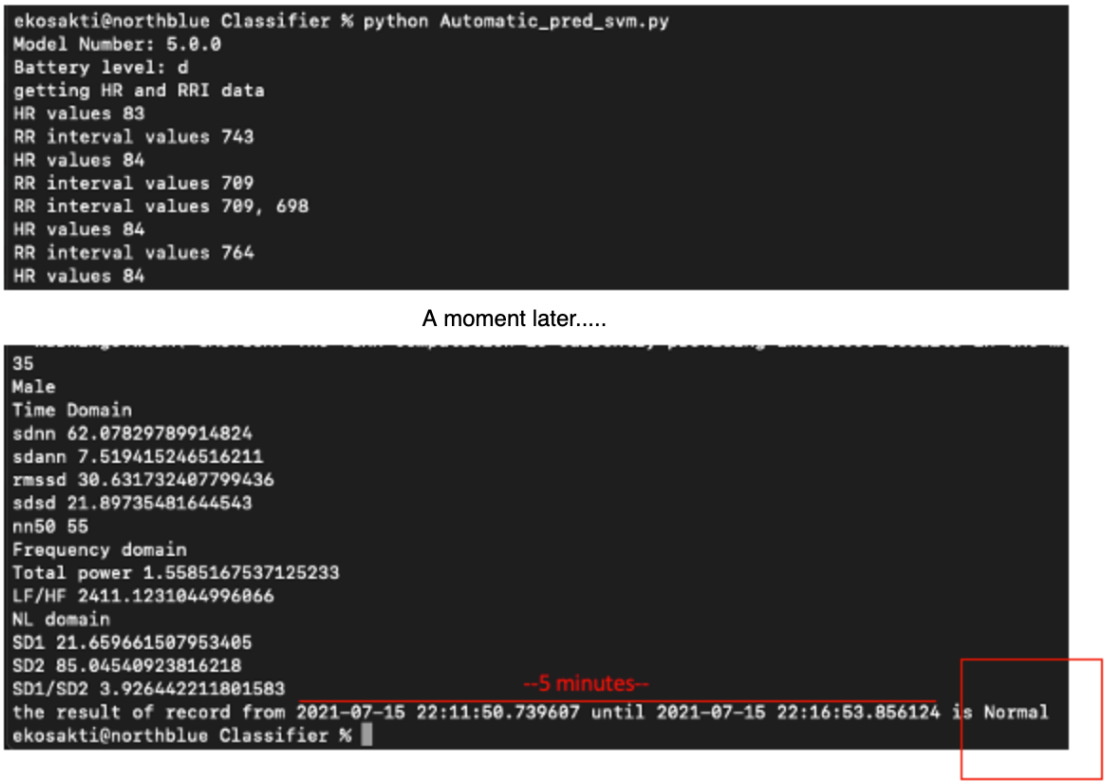

Figure 4.11: Experiment with HRV classification

4.2 RR interval-based Classifier

This section presents our heartbeat classifier based on RR interval features. The classifier was trained using a dataset from the MIT-BIH arrhythmia database [38]. Even though this dataset is imbalanced (imbalanced data would impact classification accuracy), this data has already been labeled, annotated, and is publicly available. The dataset consists of 48 recordings of patients' data. Each data has a 30-minute ECG recording. Among 48 recording number, 102, 104, 107, and 217 is omitted for training data because they consist of paced rhythm. Furthermore, I extract features for classification using this database.

RR interval data is measured from the distance of the two R peaks in each ECG wave (PQRS). This variable can reflect the physical condition [22]. Detecting the R wave in the ECG recording is needed to calculate the RR interval. In this case, I used Pan-Tompkins Algorithm [47]. Thus, calculate the distance from one R wave to the next detected R wave. After the RR interval's value is known, I calculate the RR interval series as one feature within 42 windows of RR interval data. There are several types of RR interval series, as shown in Table 4.2. I extract the RR interval series as a feature from the training and testing data. The RR interval series has the following characteristics: RR0, RR-1, RR+1, RR0/avgRR, RR-1/avgRR, RR-1/RR0, RR-1/RR0, RR+1/avgRR, RR+1/RR0. An average RR interval in the period window is required to calculate a normalized RR interval. Usually, the average RR interval is calculated in a patient-wise way. Patient-wise means calculating the average RR of all recorded data. In a real-time scenario, especially in stream processing, calculation of entire recorded data is impossible because data keep growing. Thus it is suggested to compute previously known data. In this study, feature extraction uses 42 previous RR intervals to minimize computational time and speed up the classification process. For this reason, feature extraction for training classifier from MIT-BIH arrhythmia database, the average RR interval is calculated from 42 windows of the previous RR interval. The RR interval can be computed into nine features; thus, it does not need a feature selection due to its low complexity.

Table 4.2: RR interval feature series.

Features Series	**Descriptions**
RR0	Current RRi value
RR-1	Previous RRi value
RR+1	Next RRi value
RR0/avgRR	Current RRi/average of RRi within 42 s
tRR0	(CurrentRR-averageRR)/stddevRR
RR-1/avgRR	Previous RRi/average of RRi
RR-1/RR0	Previous RRi/ current RRi within 42 s
RR+1/avgRR	Next RRi, average of RRi within 42 s
RR+1/RR0	Next RRi, current RRi

Table 4.3: Distribution of heartbeats class in MIT-BIH data.

	Original	**ROS**	**SMOTE**	**ADASYN**
number of N	90,125	90,125	90,125	90,125
number of S	2781	90,125	90,125	90,332
number of V	7009	90,125	90,125	89,215
number of F	803	90,125	90,125	90,293
number of Q	15	90,125	90,125	90,120

Finding effective features for classifiers in the automated heartbeat classification systems is still an open problem. In this study, I use R0, R-1, R+1, and RRI normalization because these features can achieve maximum accuracy by RRI series. I have done a feature selection experiment using only R0 the accuracy achieved 70% while R0, R1, R-1 = 99.17%. I also extend distance of RR series become R-2, R-1,R0,R+1, R+2 achieve accuracy 99.09% and R-3,R-2, R-1,R0,R+1, R+2 , R+3 achieved accuracy 99.05%. By this result, expanding the value of the RR feature will lower the accuracy because it becomes far from the labeled beat. As I concluded, R-1, R0, and R+1 are more discriminate for heart rate classification. Thus I added a normalized RR-interval feature. Using normalized features combined with RR intervals can increase the classification accuracy [36]. Saenz-Cogollo demonstrated using normalized features combined with RR interval can increase the classification accuracy [36]. Additionally, I use $tRR0$ introduced by [32] as the timing relationship between successive R waves. As shown in Table 4.2 I use nine features for developing an automated heartbeat classifier.

4.2.1 Train the RR interval-based Classifiers

I train the classifier using inter-patient and intra-patient paradigms with the MIT-BIH arrhythmia dataset to create the best classifier based on those features. The inter-patient paradigm means that the training and testing data come from different patient recordings. Later, it is called *protocol splitting* because many previous studies used this method to split the training and testing data [48]. At the same time, in the intra-patient paradigm, the data for training and testing may come from the same patient recording, which later is called *random splitting*. The protocol splitting will make the classifier work harder because the model will classify new data [29]. The splitting data based on inter-patient is defined as shown in 4.4. While in intra-patient, the scheme of splitting data is done randomly, selecting 70% from available data as training data and the remaining as testing data.

For classification methods, several machine learning and deep learning are used to classify five classes of heartbeats. I use Scikit learn library in Python to train the model using Decision Tree (DT), Gradient Boosting (GB), k-Nearest Neighbors (KNN), Multilayer Perceptron (MLP), Random Forest (RF), and Support Vector Machine (SVM). The training parameter is shown in Table 4.5.

Table 4.4: Splitting dataset.

Types	Record's numbers
training dataset	101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230
testing dataset	100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210,212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234

Table 4.5: Model parameters.

Model	Parameter
DT	default
GB	estimator = 100, learning rate = 0.1, max. depth = 3, random state = 0.
kNN	k = 3.
MLP	network solver = adam, alpha=1e-5, hidden layer = 128, input layer = 9 output layer = 5, max iteration = 600, random state = 42.
RF	tree = 30, random state = 42.
SVM	kernel = RBF, gamma = 0.8, C = 1.

Table 4.6: Deep learning model summary.

Layer (Type)	Output Shape	Param
dense (Dense)	314,857, 9	80
dense_1 (Dense)	314,857, 64	576
dense_2 (Dense)	314,857, 128	8320
dense_3 (Dense)	314,857, 512	66,048
dense_4 (Dense)	314,857, 128	65,664
dense_5 (Dense)	314,857, 64	8256
dense_6 (Dense)	314857, 5	325

For deep learning, I use tensor flow to train the model using sequential with artificial neural networks (ANN) [49]. The summary of the model is shown in Table 4.6. There are seven layers with nine nodes at the input layer, five at the output layer, and five at the hidden layer. The activation function is ReLu and $softmax$, $kernelregularizer$ (l2) is 0.0001, the optimizer is *adam*, and the loss function is $sparsecategoricalcross-entropy$. Four evaluation metrics, such as accuracy, precision, recall, and F1-score, are used to evaluate the classifiers.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.1}$$

$$Precision = \frac{TP}{TP+FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.3}$$

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4.4}$$

Evaluation is done by validating the model with data testing. The accuracy is a metric to measure the correctness of the predicted class with the true class in the dataset. The precision parameter defines a correct prediction class divided by all numbers resulting from prediction, known as the positive predicted value. At the same time, recall is used to measure the actual value of the predicted class identified correctly or known as sensitivity. The F1-score measure the balance between precision and recall, especially in the imbalance dataset. For the first model, I use several machine learning to train a classifier by splitting the data using protocol from [48] and random split as an intra-patient paradigm. For intra-patient training and testing data, I split randomly from the whole recording by 70% for training and 30% for testing. As shown in Table 4.7, I have three kinds of data splitting mechanisms. The first is protocol split, random split, and random split of oversampled data. Thus, I am conducting the training for those splitting for each classification method. I perform training five times.

Table 4.7: Dataset overview.

	Protocol Split		**Random Split**		**Oversampling**	
	Train	**Test**	**Train**	Test	**Train**	**Test**
number of N	45,866	44,259	63,150	26,975	63,050	27,075
number of S	944	1837	1973	808	63,225	26900
number of V	3788	3221	4845	2164	63042	27,083
number of F	415	388	536	267	63,076	27049
number of Q	8	7	9	6	63,044	27,081
Total	51,021	49,712	70,513	30,220	315,437	135,188

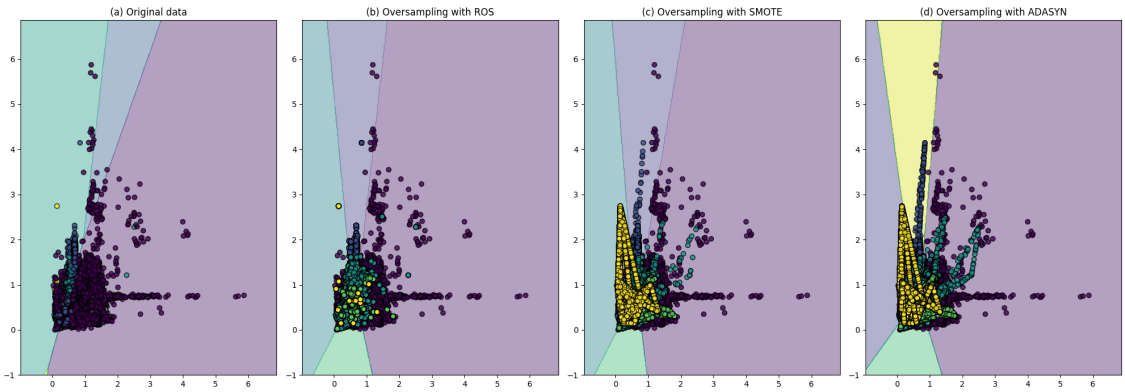

Figure 4.12: Decision boundary using logistic regression

4.3 Training Results of RR interval-based Classifiers

I conducted three schemes for training the classifier based on the dataset splitting scheme. The first scheme uses inter-patient splitting, and the second uses the intra-patient with a random split on the original dataset. The third scheme is intra-patient with a random split on the over-sampled dataset. The result of the first training is shown in Table 4.8. The highest accuracy was achieved by SVM based classifier with 92.57% and 90.23%, 92.57%, 90.81% for precision, recall, and F1-score consecutively. At the same time, the Neural Network-based classifier achieved the accuracy of 92.50% and 91.36%, 92.50%, 91.46% for precision, recall, and F1-score consecutively. As a supplement for those results, I present the confusion matrix at Table 4.9 and Table 4.10, where the horizontal value is the result of prediction by the classifier the vertical is an actual label. As shown in the confusion matrix, the result is not so good, several values are predicted in the wrong class, and both the classifiers cannot predict the Q class (the Q class is predicted as the N class). This result is caused by many overlapping data features with other classes, in Figure 4.12 shows with original data, i.e., minority class (with the yellow dot is appear inside another class). As stated by [48] the way of data splitting will burden the classifier, especially with imbalanced data.

The second training was conducted using a random dataset split with 70% for training and 30% for testing. The training and testing were done in five times repetitions. The ANN-based classifier achieved the highest accuracy with 96.25% and 96.07%, 96.35%, 96.09% for precision, recall, and F1-score. Random Forest-based classifier yields 96.22% accuracy with 95.94%, 96.21%, 95.89% for precision, recall, and F1-score, respectively. Based on the confusion matrix shown in Table 4.12 and Table 4.17 there is still a missmatch by the classifier to predict actual label. Overall, the accuracy of each classifier is better than the protocol split. The minority class (Q) by the classifiers based on inter-patient and intra-patient are classified as a normal class. Several works reported skipping the minority class and focusing on classifying N, S, and V class [29].

The third training was done by intra-patient scheme using over-sampled data by Random Oversampling, SMOTE, and ADASYN. The number of data for training is 315437 and 135188 for testing data. In this configuration, the data for each class is nearly equal. As a result, the maximum accuracy achieved is 99.67% by the Random Forest-based classifier. The precision, recall, and F1-score are 99.67%, 99.67%, and 99.67%. The second highest accuracy is the Decision Tree classifier with 99.31%, 99.32%, 99.31%, 99.31%

for accuracy, precision, recall, and F1-score, respectively. Table 4.14 shows the result of all classifiers using a training third training scheme. Based on the oversampling method, Random oversampling is dominant compared to other oversampling methods in terms of classifier accuracy. The way the ROS works by duplicating the minority class may lead to this dominance. However, the classifier trained using SMOTE also gives good results that achieve 98.15% accuracy by random forest classifier. As shown in confusion matrix Table 4.15, Table 4.16, and Table 4.17 the overlap causing miss-prediction by the classifier is fewer compared to the confusion matrix based on training classifiers using scheme one and two. These classifiers can recognize the F and Q classes, while the classifier based on training one and two schemes failed to predict the F and Q classes. Regarding the training result in Table 4.14 classification accuracy from several algorithms decreases, such as Gradient boosting, neural network, and support vector machine. This is due to changes in the dataset becoming non-linear. An algorithm that works better on non-linear data shows good results in accuracy, such as a random forest.

Table 4.18 shows the comparison of our classifier with previously proposed classifiers. The trained classifier in this study has competitive performance among previously reported classifiers. Moreover, our classifier only uses a simple feature from the RR interval series. Some classifiers can achieve higher accuracy compared to those previously reported.

Table 4.8: Result of machine learning using protocol split.

Method	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
DT	89.15	88.30	89.15	88.64
GB	89.08	89.10	89.08	88.44
KNN	90.76	88.42	90.76	89.42
NN	92.50	91.36	92.50	91.46
RF	91.81	89.24	91.81	90.29
SVM	92.57	90.23	92.57	90.81
ANN	91.44	88.59	91.04	89.72

Table 4.9: Confusion matrix SVM.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	43588	49	622	0	0
	S	1159	79	599	0	0
	V	808	64	2349	0	0
	F	385	0	3	0	0
	Q	6	1	0	0	0

Table 4.10: Confusion matrix NN.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	43170	199	872	18	0
	S	789	279	768	1	0
	V	619	55	2535	12	0
	F	382	0	5	1	0
	Q	6	0	1	0	0

Table 4.11: Result of machine learning using random split dataset.

Method	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
DT	94.08	94.12	94.08	94.10
GB	95.57	95.29	95.57	95.21
KNN	95.08	94.50	95.08	94.53
NN	95.82	95.53	95.82	95.46
RF	96.22	95.94	96.21	95.89
SVM	95.05	93.97	95.05	94.35
ANN	96.35	96.07	96.35	96.09

Table 4.12: Confusion matrix RF.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	26,734	20	206	15	0
	S	129	594	85	0	0
	V	420	43	1698	3	0
	F	211	0	6	50	0
	Q	6	0	0	0	0

Table 4.13: Confusion matrix ANN

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	26,776.2	28	214.6	15.2	0
	S	112.8	632.6	88.6	0	0
	V	392.6	44.6	1663	10.8	0
	F	178	0	9.4	46.6	0
	Q	5.75	0.75	0.5	0	0

Table 4.14: Result of training using over-sampled dataset

	Accuracy (%)			Precision (%)			Recall (%)			F1-Score (%)		
	R	**S**	**A**	**R**	**S**	**A**	**R**	**S**	**A**	**R**	**S**	**A**
DT	**99.31**	96.50	96.08	99.32	96.49	96.07	99.31	96.50	96.08	99.31	96.49	96.06
GB	89.57	86.73	78.26	89.62	86.70	77.94	89.57	86.73	78.26	89.55	86.67	78.01
KNN	**98.93**	97.55	97.49	98.99	97.71	97.56	98.97	97.68	97.49	98.96	97.64	97.44
NN	89.88	90.06	84.48	90.17	90.19	84.54	89.88	90.06	84.48	89.81	89.96	84.23
RF	**99.67**	98.15	98.08	99.67	98.15	98.09	99.67	98.15	98.08	99.67	98.14	98.07
SVM	87.87	87.43	79.83	87.93	87.46	79.59	87.87	87.43	79.83	87.78	87.32	79.39
ANN	**97.51**	96.20	95.85	97.54	96.22	95.81	97.51	96.20	95.85	97.49	96.18	95.83

Table 4.15: Confusion matrix RF-ROS.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	26,626	32	370	44	2
	S	0	26,900	0	0	0
	V	0	0	27083	0	0
	F	0	0	0	27,049	0
	Q	0	0	0	0	27,081

Table 4.16: Confusion matrix DT-ROS.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	26,148	141	576	205	5
	S	0	26,900	0	0	0
	V	0	0	27083	0	0
	F	0	0	0	27,049	0
	Q	0	0	0	0	27,081

Table 4.17: Confusion matrix ANN-ROS.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	24,773.4	255.8	943.4	1073.6	27.8
	S	84.2	26,670.75	146.5	2.4	0
	V	334.4	123.8	26,277.75	310.4	0.4
	F	50	0	13.6	26,992	0
	Q	0	0	0	0	27,082

Table 4.18: Works comparison following AAMI recommendation.

Classifier	**Feature**	**No. of Features**	**Data Scheme**	**Class**	**Accuracy (%)**
Ensemble SVM [37]	RRi, HOS, wavelet, time domain, morphology	45	Inter-patient	5	94.5
Random Forest [36]	RRi, HBF, time domain, morphology	6	Inter-patient	5	96.14
Naïve bayes [50]	HOS	4	Inter-patient	5	94
SVM [51]	RRi, DCT Random projection	33	Inter-patient	5	93.1
Ensemble of BDT [52]	RRi, DCT random projection	33	Inter-patient	5	96.15
Ensemble SVM [53]	RRi, Random projection	101	Inter-patient	5	93.8
Deep neural network [54]	RRi, Wavelet, HOS, morphology	45	Inter-patient	4	89.2
This work (SVM)	RRi	9	Inter-patient	5	92.57
This work (NN)	RRi	9	Inter-patient	5	92.50
This work (RF)	RRi	9	Intra-patient	5	96.22
This work (ANN)	RRi	9	Intra-patient	5	**96.35**
This work (RF)	RRi	9	Intra-patient(O)	5	**99.67**
This work (DT)	RRi	9	Intra-patient(O)	5	**99.31**

4.4 Morphology-based Classifier

In this section, I presented the classification based on ECG morphology. The classifier is trained using an ECG signal from the MIT-BIH arrhythmia database to predict the ECG signal from Polar H10. The classification classes of a heartbeat I use from the AAMI recommendation are N, SVEB, VEB, F, and Q. Before developing the classifier. I need to confirm whether the ECG signal from both sources is similar.

4.4.1 Comparison ECG Signal from the Dataset and Polar H10

MIT-BIH arrhythmia dataset consists of 48 recordings with 30 minutes, equal to 650,000 samples of ECG data. The sampling frequency is 360Hz, which means each second consists of 360 models, while Polar H10 uses 130 Hz. Figure 4.13 shows the plotting of ECG recording from a dataset with record 116, while Figure 4.14 Plotting ECG signal from Polar H10. Both figures are data within 30 minutes of ECG data. I also re-sample the signal of Polar H10 become 360 Hz and re-sample data from Polar H10 from 180 Hz to 360Hz. As a result, the ECG sample from the Polar H10 recording became 648,509, as shown in Figure 6. Furthermore, I am plotting 4 seconds of ECG signal from several recordings, such as record numbers 116, 100, 101, 102, and record from Polar H10. The noticeable differences are polar H10 using high voltage while recording from MIT-BIH arrhythmia is low voltage. However, the amplitude of each ECG signal is different.

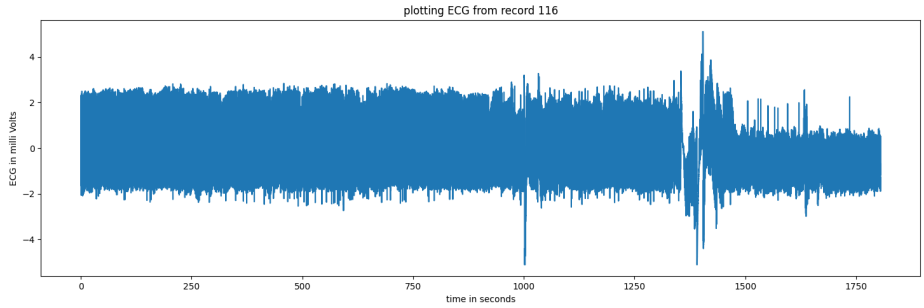

Figure 4.13: Plotting recording record 116

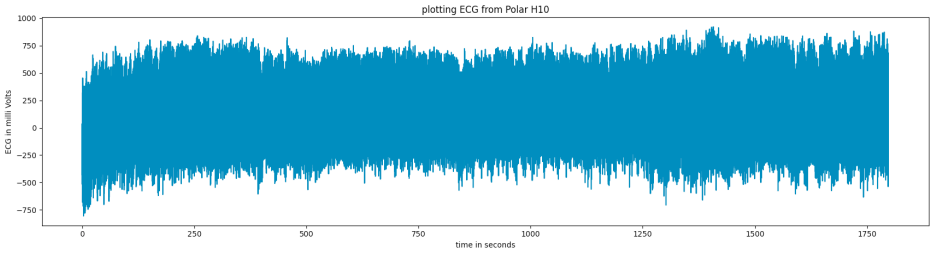

Figure 4.14: Plotting ECG recording Polar H10

An ECG morphology or PQRS wave classifier is trained using an arrhythmia database to classify ECG signal output from Polar H10. The morphology is regarding the shape of PQRS waves. The required signal for classification is a complete shape in which the sign of PQRS is visible. Figure 4.15 plotting from record 116, Figure 4.16 plotting from Polar H10, Figure 4.17 plotting from record 100, and Figure 4.18 plotting from record 101, are

consisting of the PQRS waves. Thus, I conclude the shape or morphology of PQRS waves from MIT-BIH arrhythmia and Polar H10 recording are similar.

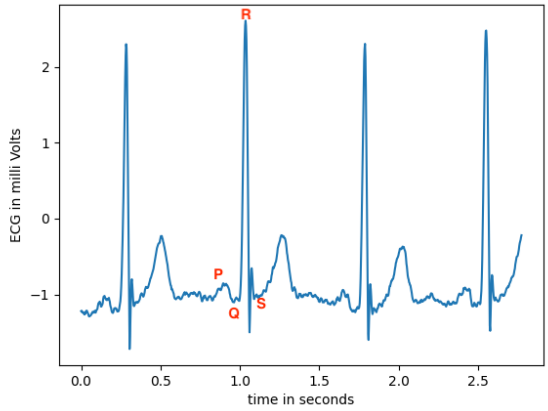

Figure 4.15: Plotting 4 second of ECG recording record 116

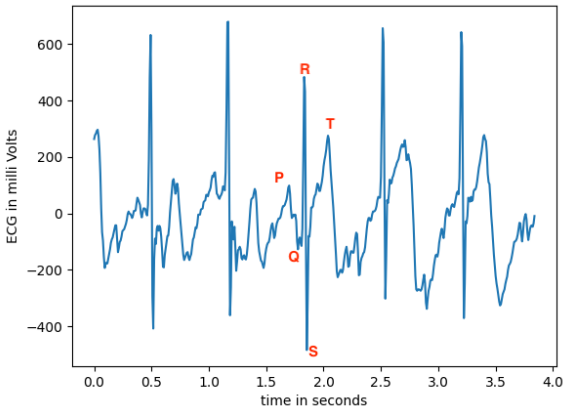

Figure 4.16: Plotting 4 second of ECG recording Polar H10

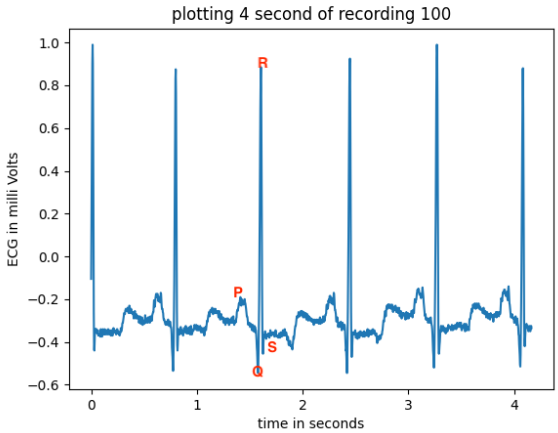

Figure 4.17: Plotting 4 second of ECG recording record 100

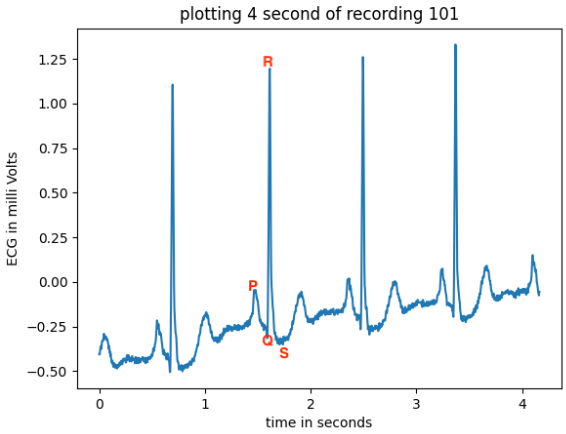

Figure 4.18: Plotting 4 second of ECG recording record 101

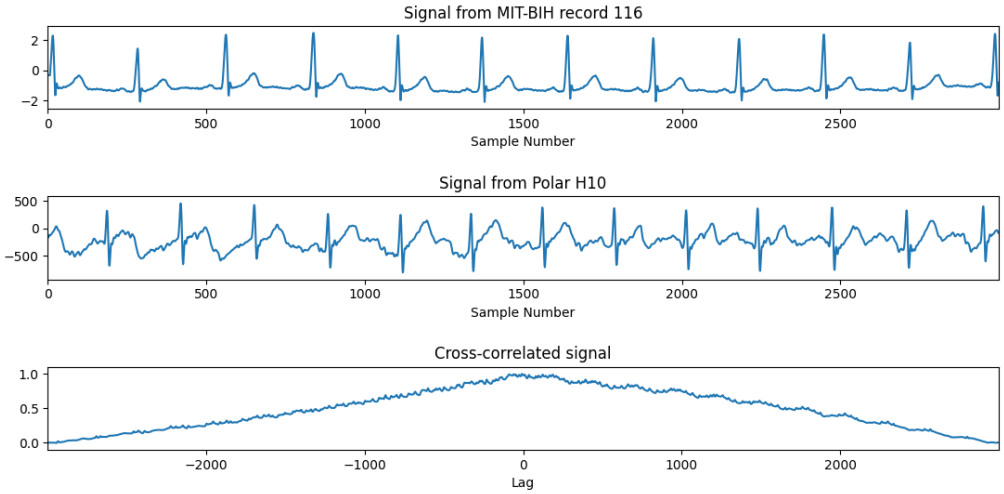

Figure 4.19: cross-correlation ECG signal from record 116 and Polar H10

In more detail regarding similarities between two signals, I use a cross-correlation method. This method measures a level of correlation between ECG signal from record 116 and Polar H10. The similarity indicates by a value from -1 until 1, where 1 is closely similar and -1 there is no similarity. As shown in Figure 4.19, the result of cross-correlation concluded those two signals are similar. When the lag is 0, the value nearly reaches 1. Lag is the indication location of two signals, where 0 is two signals in the same position.

4.4.2 Training the Morphology-based Classifier

The training classifier step consists of pre-processing, generating images, defining the architecture, training, and evaluation.

The pre-processing step converts signals from MIT-BIH arrhythmia records into 2D images of each labeled class. Each image represents the PQRS waves from the records, an image taken by the center in the R peak. The records are 101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230, 100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231,

232, 233, 234. How to extract and plot the signal shown in Algorithm 6. The feature for training shown in Figures 4.20 4.21, 4.22, 4.23, 4.24. Thus, The images play as input for 2D CNN with a size 128x128 gray-scale image. As a result, I have 100.606 images of ECG signals for training the classifier.

The CNN architecture is shown in Figure 4.25. Our model consists of 18 layers. The first layer is the input layer, where the input dimension of images is 128 x 128. I use Rectified Linear Unit (relu) for the activation function and apply a penalty to maintain the model overfitting by *regularizers.l*1_*l*2. To standardize input to the next layer, I use batch normalization. Thus I implement convolution operation starting with kernel size 128 x 128, followed by 64x64, 128x128, 128x128, 256x256. After each convolution layer, I implement MaxPooling and Dropout layers. MaxPooling helps to prevent over-fitting by conducting down-samples of the input, as well as by reducing the computational cost. At the same time, Dropout will prevent the model from selecting neurons randomly. In CNN, convolution is the process of automatic feature generation. After the convolution layer, I implement the Flatten layer. In the Flatten layers, output data from the convolution layer is converted into a 1-dimensional array. Thus the 1-dimensional array will become input in a dense layer for classification. The next layer is the Dense layer. The dense layer in this study aimed to classify the signal into five classes. In dense layer I use activation $softmax$, optimizer *adam*, and loss function with $categorical\_crossentropy$.

I train the classifier using MIT-BIH arrhythmia with an intra-patient paradigm with a random split for 70% training, 10% validation, and 20% testing. Number datasets for training, validation, and testing are 70.422, 9.053, and 21.131, respectively. Other training parameters are epoch 30, batch size 64, and regularizers 0.0001. Thus, I saved the best model based on those parameters in h5 file format for other purposes, such as implementing the classifier in the data analysis at cloud applications. The h5 is Hierarchical Data Format. Training duration time is 2h 08m and 12s

Algorithm 6 Morphology feature extractions

```
Initialize:
    label = N, S, V, F, Q
function GET_SIGNAL(record)
    signal ← wfdb.rdsamp(record)
    annotation ← wfdb.rdann(record)
    for s doignal in all recording
        Plot signal
        Save Plot for each label
        image ← grayscale
    end for
end function
```

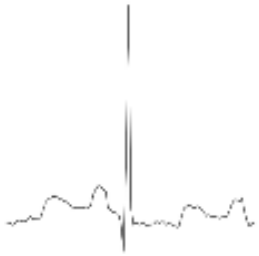

Figure 4.20: Sample of label N

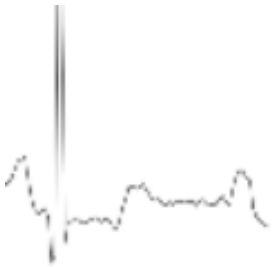

Figure 4.21: Sample of label S

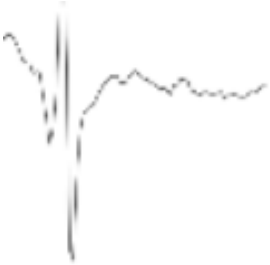

Figure 4.22: Sample of label V

Figure 4.23: Sample of label F

Figure 4.24: Sample of label Q

I use parameters such as accuracy, precision, recall, F1-score, and confusion matrix to evaluate the model from the training step. As a result, I achieved precision, recall, and F1-score 97%. Table 4.19 shows the confusion matrix from our classifier. Where reference is the actual label and classifier is the prediction result by the classifier.

Table 4.19: Confusion matrix CNN based.

		Classifier				
		n	**s**	**v**	**f**	**q**
Reference	N	13,371	13	90	5	0
	S	153	266	20	0	0
	V	62	3	981	6	0
	F	39	0	20	62	0
	Q	3	0	0	0	0

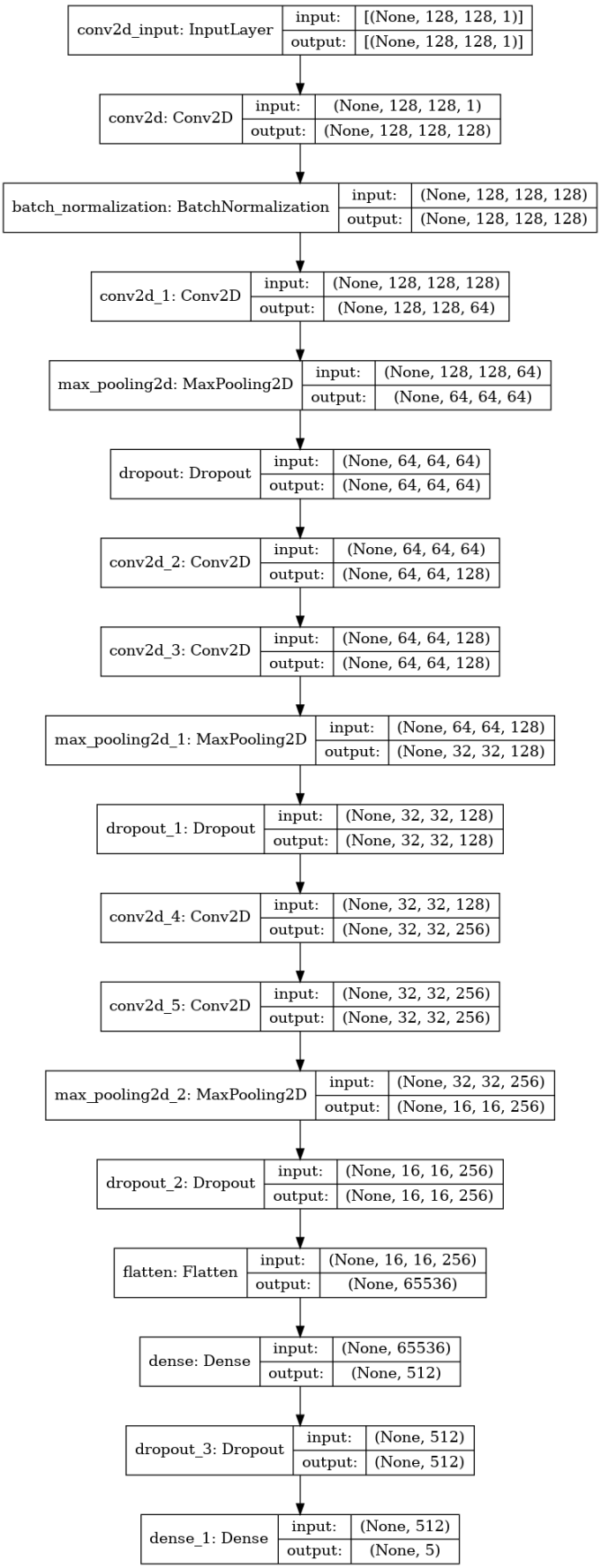

Figure 4.25: Our CNN model

4.4.3 Comparison with Others ECG-based Classifier

In this section, I present the comparison of other proposed classifier-based ECG signals. As a result, our classifier achieves competitive results compared to another classifier that uses other features such as Continuous Wavelet Transform, Short-time Fourier transform, and Wavelet.

Table 4.20: Works comparison ECG based classifiers.

Classifier	Feature	Class	Accuracy (%)
2D-CNN [55]	Continuous wavelet transform	8	99.02
2D-CNN [56]	Short-time Fourier transform	8	98.92
2D-CNN [57]	Wavelet	5	99.43
2D-CNN [13]	Continuous wavelet transform, RR interval	5	98.7
2D-CNN [58]	Dual-beat coupling	5	96.5
2D-CNN (This works)	Morphology	5	97

4.5 Summary

In this chapter, I presented the discussion related to the proposal for Heartbeat classification. The classifier will classify output data from Polar H10 into five classes: normal beat (N), supraventricular ectopic beat (SVEB), ventricular ectopic beat (VEB), fusion beat (F), and unknown beat (Q). The classifier is trained using a dataset for the MIT-BIH arrhythmia database. Three kinds of heartbeat classifiers are presented. A classifier based on HRV, RRi, and ECG morphology features. The study's challenge is achieving a heartbeat classifier with sufficient accuracy based on the output data from Polar H10 as the classification feature.

Chapter 5

An IoT-based Monitoring Framework for Early Detection of Cardiovascular Disease

In this chapter, I present the application of our IoT framework. The framework consists of things and middleware, web applications, and data analysis applications. As shown in Figure 5.1, I also define two kinds of applications: fog-based and cloud computing. In fog-based applications consist of Things and middleware. In comparison, cloud computing-based consists of web applications and data analysis, which runs on a cloud computing environment. The application based on fog computing is provided for real-time or stream monitoring and prediction. The second application is based on cloud computing for multi-analysis and scalability. The application is intended as self monitor our user's heartbeat condition. Moreover, the frameworks are also used to test the capability of our proposed classifiers.

5.1 Proposed IoT Architecture

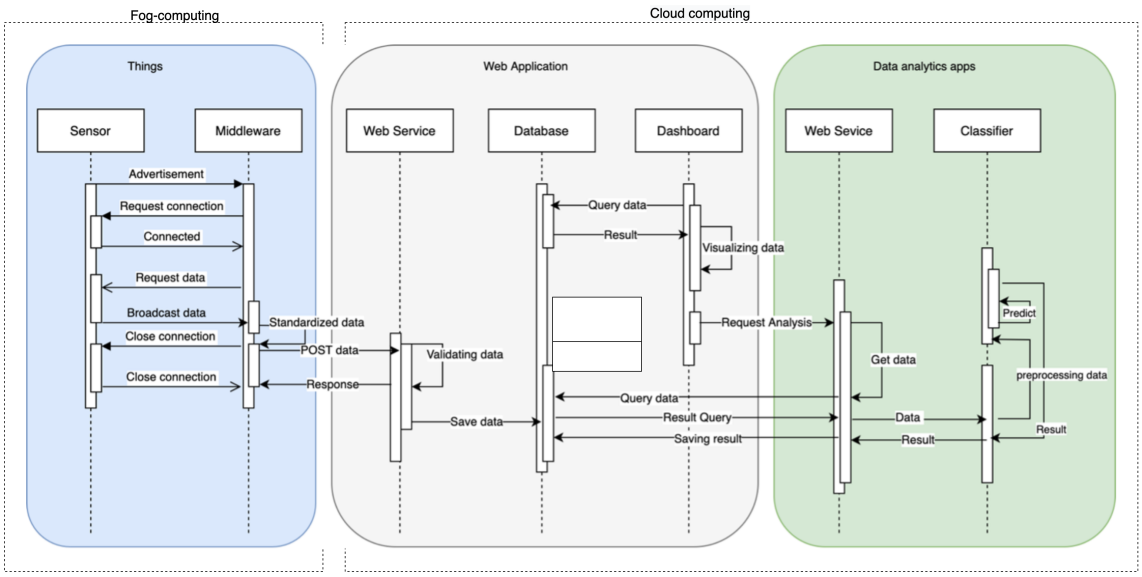

Figure 5.1: IoT apps Architecture

The proposed system was built with a modular approach to allow flexibility in application development, as depicted in Figure 5.1. The suggested system comprises three modules: things, web application, and data analysis.

1. Sensors and middleware are components of the Things system. The sensor is a chest strap device (Polar H10) that generates heartbeat data. I use the polar H10 because it has a gold standard data output for determining heartbeat conditions. This device can generate heart rate, RR interval, and ECG data based on a person's heartbeat. For middleware devices, I use a desktop computer instead of a smartphone. Smartphone-based applications are resource-constrained and unsuitable for the long-term recording of health data. I developed a middleware from a desktop-based BLE framework. A BLEAK framework is chosen over all other free and open BLE frameworks because of its ability to run on several operating system environments. As a result, it will address the issue of operating system compatibility. The middleware offers several functions, such as maintaining communication with the Polar H10, standardizing data by converting raw data from the sensor into JSON format, and passing data to cloud applications using the POST method.

2. A web application is developed with several subsystems, such as a web service, a database, and a web-based application. Those modules' explanations are as follows. Web services are used to receive data from middleware. The communication is provided using the POST method over HTTP protocol. For authentication, I utilize JSON Web Token to ensure only legitimate middleware can send data. The token is created by users through web applications while registering their devices. Instead of the token, the web service will validate the topic of data. The topic is used as a marker to distinguish types of heartbeat data. For example, $/RRinteval$ is a topic for RR interval data. If received data is validated, thus the data will be stored as a collection in the MongoDB database. The web service will send a response to middleware whether the received data is valid or unable to validate. I chose MongoDB as a database management system since the data from different middleware will be varied (such as RR interval data or ECG data). MongoDB supports the flexibility of storing user data as a collection. I make a collection of each user's data. I also provide a web-based interface to let users manage their credential information, devices, topics, analysis tools, and visualization of analysis results.

3. An analytic data module is developed with data analysis functions and web service for communication. Data analysis in this module will appear in other applications as a service. Users can access the analytic data service through the dashboard application. There are two options for the user to access the analytic service: RR interval classifier and ECG classifier. If users want to analyze their data, they should choose their data and determine the analytic function in the dashboard. Thus, the dashboard application will tell the data analysis through a web service to process the desired data. The data analysis system will ask the database to collect the user's heartbeat data based on the user's topic. If the RR interval classifier is selected, the raw data from the RR intervals will be extracted into the RR interval feature series. After that, the classifier created in the training phase will predict the heartbeat types. The heartbeat prediction result will appear in the user dashboard and be stored in the database as a history prediction.

5.2 Fog-Computing Framework

In this section, I provided an experiment using our classifier and developed a system to monitor heartbeats continuously in real-case scenarios. This experiment involved a healthy person in measuring the capabilities of a classifier to predict data continuously and as a preliminary experiment to validate our developed system. I choose the classifier with accuracy above 96% for each method among all the classifiers. The experiment runs for 20 minutes for each classifier. Our previous study concluded that Polar H10 and middleware could maintain good communication by receiving signal strength (RSSI) above -80dBm until 50 meters at no obstacle environment and 16 meters at obstacle environment [40]. Thus, the participants freely move as long as 50 meters within the middleware.

As shown in Figure 5.2, our experiment uses Polar H10 as a sensor, middleware, classifier, and visualizer. The middleware, classifier, and visualizer are run on a personal computer. The fog computing-based application works as follows :

1. The middleware initiates communication through BLE with Polar H10. In this experiment, I use BLEAK as the BLE framework.

2. After communication establishes, middleware requests heart rate measurement. Then middleware will wait for RR interval data from Polar H10.

3. The Polar H10 will send data by broadcast every second. The data consists of RR interval and heart rate.

4. The middleware will listen and wait until it receives 42 RR interval data. The classification process will start if 42 RR interval data are collected.

5. The classification process is started with feature extraction to form 9 kinds of RR interval series. Those features are RR0, RR-1, RR+1, RR0/avgRR, RR-1/avgRR, RR-1/RR0, RR-1/RR0, RR+1/avgRR, RR+1/RR0.

6. The classifier will predict the RR interval series to determine the class.

7. The prediction result is visualized in the command line interface (CLI), as shown in Figure 5.3. Figure 5.3 consists of information regarding the recording time, extracted feature, heart rate, prediction result, and computation time.

8. Finally, the middleware closes the connection with Polar H10 after 20 minutes of recording.

The classifier's performance is presented in Table 5.1. Classifiers based on Random Forest have the longest average processing time with 0.108851 seconds. The classifier with the fastest processing time is Decision Tree with 0.00035943 seconds. During 20 minutes, the number of beats varies; most prediction results are normal beats. The average inference time of the classifiers is less than 1 second, and they can give prediction results within one second. Thus, the classifier is suitable for continuous and real-time prediction of a heartbeat. I also provide information regarding RSSI during the experiment. As shown in Table 5.1, the average RSSI is above -80dBm, which indicates the transmission data between the sensor and middleware is in good condition while the participant moves around the middleware.

I also conducted an experiment involving six healthy people to evaluate our developed system. The four participants are male, and two are female. Their ages also varied. I used a random forest classifier trained with random oversampling in this experiment. The experiment runs for 30 minutes for each participant. I also measure the received signal strength indicator (RSSI) for the quality of received data from Polar H10. Table 5.2 shows the number of beats in 30 minutes from each participant is varied. All of the received beats are predicted as normal. According to the RSSI, I concluded the transmission data is in a good state, which is under -80dBm. The value of RSSI also indicates the distance between the participant with the middleware device. The more excellent value of RSSI means the participant is close to middleware.

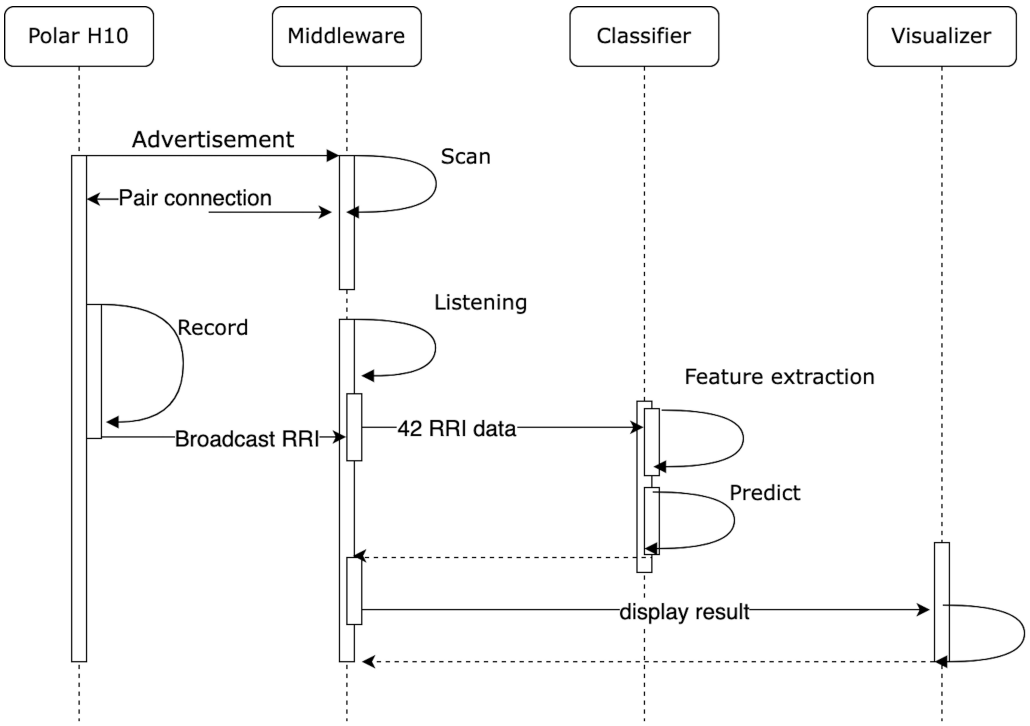

Figure 5.2: Class diagram of real-time monitoring system

```
-------------- 2022-04-22 - 17:37:45.116565 -------------
feature of RRI series (668.0, 663.0, 662.0, 1.0453444614180856, 1.3841982729848976, 1.037520026826633,
0.9925149700598802, 1.0359551399083424, 0.9910179640718563)
Your heartrate is 99
Your hearbeat is Normal
computation time :  0.030233144760131836
------------------------------------------------------------
-------------- 2022-04-22 - 17:37:46.114068 -------------
feature of RRI series (662.0, 666.0, 680.0, 1.037346565682946, 1.1796562263234776, 1.0436145207626013,
1.0060422960725075, 1.0655523635413946, 1.027190332326284)
Your heartrate is 98
Your hearbeat is Normal
computation time :  0.02966594696044922
------------------------------------------------------------
-------------- 2022-04-22 - 17:37:47.111681 -------------
feature of RRI series (666.0, 680.0, 658.0, 1.0440819678250157, 1.4096320962049336, 1.0660296368183344,
 1.021021021021021, 1.0315404426859767, 0.987987987987988)
Your heartrate is 98
Your hearbeat is Normal
computation time :  0.03921675682067871
------------------------------------------------------------
```

Figure 5.3: An output of real-time prediction of heartbeat

Table 5.1: Experiment result on healthy person within 20 min.

	Average Processing Time (Second)	**Found Beat**					**Number Beats**	**Average RSSI (dBm)**
		N	**S**	**V**	**F**	**Q**		
RF	0.108739	1172	0	0	0	0	1172	−49.15
ANN	0.043825	1172	0	0	0	0	1172	−62.80
ANN-ROS	0.043033	1177	0	0	0	0	1177	−69.98
DT-ROS	0.00035	1169	0	0	0	0	1169	−67.23
RF-SMOT	0.10885	1177	0	0	0	0	1177	−64.63
KNN-ROS	0.00194	1171	0	0	0	0	1177	−52.47
RF-ROS	0.10563	1176	0	0	0	0	1176	−45.032

Table 5.2: Result from the of experiment on six healthy people within 30 min.

Participant	**Age**	**Gender**	**Found Beat**					**Average RSSI (dBm)**
			N	**S**	**V**	**F**	**Q**	
1	33	M	1764	0	0	0	0	−63.7
2	34	M	1773	0	0	0	0	−59.1
3	36	M	1753	0	0	0	0	−59.3
4	35	M	1773	0	0	0	0	−46.9
5	28	F	1752	0	0	0	0	−72.8
6	33	F	1772	0	0	0	0	−60.5

5.3 Cloud-based Framework

This section presents the developed application that runs on cloud computing environments. As shown in Figure 5.4, there are three instances in a cloud application, the first instance for handling User credentials, data management, authentication, and database, the second instance for data analysis based on RR interval, and the third is an instance for ECG morphology analysis. Each instance does communication through a web service. In the first instance, the web service is provided to receive data from middleware, the logic of this web service shown in algorithm 7, while in the data analysis instance, a web service is provided to handle request analysis from the first instance, as shown in algorithm 8

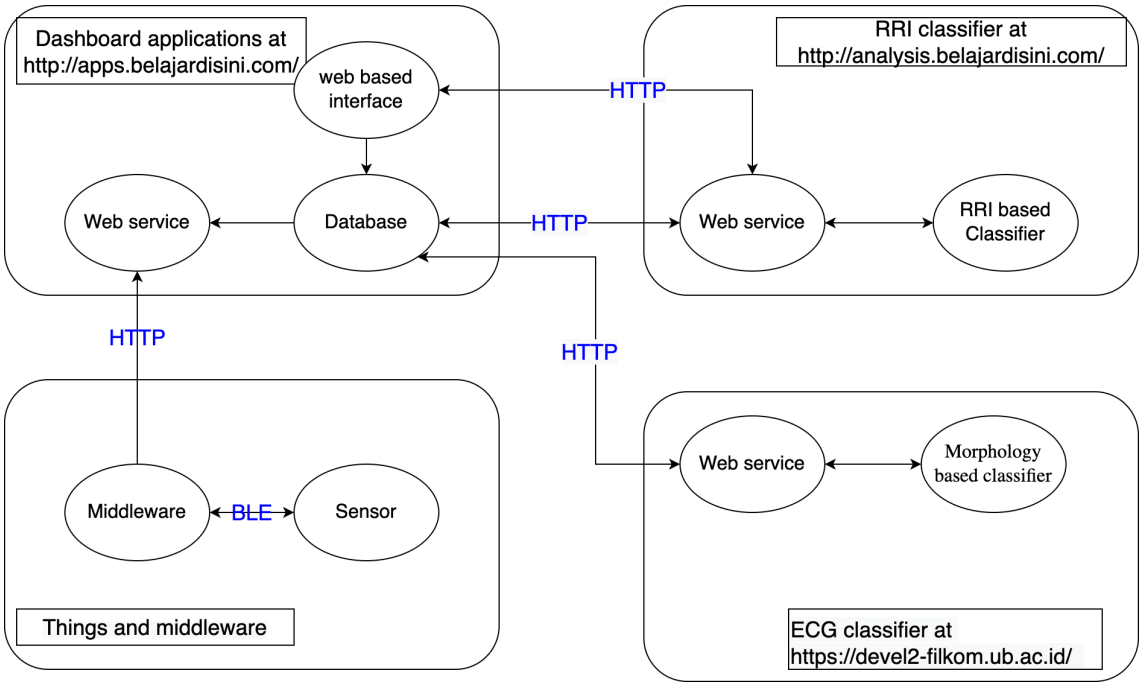

Figure 5.4: Micro-service Architecture

Algorithm 7 web service on cloud apps

```
1: function LISTENING_DATA(sensor_data)
2:     if topic and token = valid then
3:         store data to database
4:     end if
5: end function
```

5.3.1 User, Data Management, Authentication, and Database Instance

A web-based interface allows the user to access and edit their data. To use this system, the user must be registered to define their identity, devices, and topics and get a JWT token. The JWT token is an authentication mechanism to send user data through the Restful API. Authentication using the JWT token is chosen because it has better security than a typically used username and password. The identity of users, including their age and weight, will be used for further analysis. Users can define more than one device and

Algorithm 8 web service on data analysis

```
1: function (request)
2:     if request = valid then
3:         gets data based on request
4:         call feature extraction
5:         call predict function
6:     end if
7: end function
```

topic. As previously known, some health devices can produce data with different data formats i.e., electrocardiogram (ECG), R-R Interval (RR), and Heart Rate (HR). User must specify their topic to define their data in the user management subsystem. Thus, the topic and JWT token are required for the IoT gateway to send data to the application.

The cloud application and the rest application were developed based on the Django Frameworks. Restful API provides an interface to server request-response with an IoT gateway in our design. As shown in Figure 5.4, communication is started by the IoT gateway gathering Polar H10 data. In middleware, data from the sensor is given a topic and wrapped in JSON format afterward. Thus, data is delivered to a restful web service. The IoT gateway uses an HTTP POST method to perform delivery mechanisms and requires ECG data, topics, and JWT tokens. After receiving data from the middleware, the restful web service will validate the received JWT token and topic before data is stored in MongoDB. A failure response is delivered to the middleware if the JWT token and topic are invalid.

In this system, MongoDB was chosen to store user data. Our design consists of several collections: topics, users, and device data. The user collection consists of user credentials and identity information. The topic collection consists of registered topics by the users, and device data consist of detailed information on wearable sensor device. The implementation was done at a virtual private server.

5.3.2 Data Analysis Instance

As shown in Figure 5.4, I provide two kinds of data analysis functions in this study. The first is to predict data based on RR interval features, and the second is data analysis based on ECG morphology features. The classifiers from the previous section are being implemented in the data analysis section. I implement the data analysis instance in two virtual machines, as shown in Table 5.3.

Table 5.3: Virtual machine specification

Data analysis	Specification
RR Interval-based	1 CPU, 1GB RAM
ECG based	104 CPU, 48 GB GPU, 250 RAM

Figure 5.5 show the provided analysis tool for the user. The prediction function starts with the user selecting the analysis feature, such as ECG and RRi classification. Then the user will select the data that will be analyzed. After that, our system will do the rest.

The data analytic instance will analyze user data. After the analysis is done, the results will be displayed and saved. As shown in Figure 5.6, the result of heartbeat detection based on RRi is presented. The information consists of 7.150 Normal beats during one h recording. While in Figure 5.7, the result of ECG morphology-based heartbeat classification is presented. The information consists of plotting ECG signal and marker of detected anomaly beat, such as SVEB, VEB, F, and Q. While Normal beat is not presented in the ECG plotting.

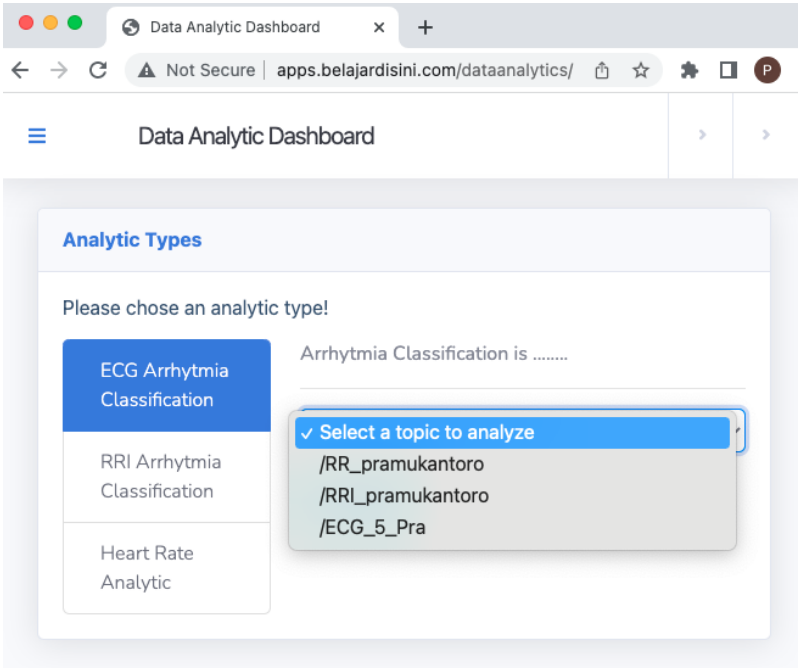

Figure 5.5: Data analysis menu

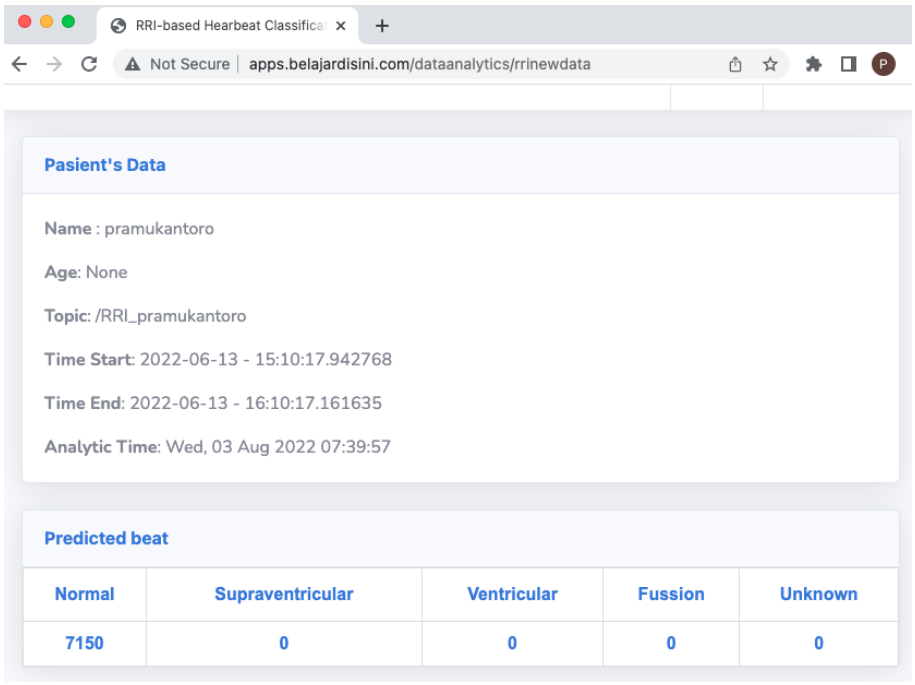

Figure 5.6: Result prediction using RRI

Figure 5.7: Result of ECG heartbeat classification

5.4 Comparison to other IoT platforms in Healthcare

As shown in Table 5.4, several authors already proposed IoT-based architecture. The closed work is Jaben [6], who claims their classifier yield 98% accuracy but uses a dataset from the UCI repository. In this matter, the achieved accuracy cannot be compared. As mentioned before, I use the MIT-BIH database in this study. I offer a complex system based on the IoT approach to detecting cardiovascular disease using commercialized wearable sensors that produce RR interval and ECG-based data compared to previous works. According to real-life testing, the system is working as expected and can predict heartbeat types based on data from polar H10.

Table 5.4: Works comparison ECG based classifiers.

Proposed	Sensing	Processing
Distributed computational framework based on IoT [17]	Monitoring heart rate football players using smartwatch and chess strap sensor	Detection of heartbeat anomaly. However, what kind of anomaly is unclear because the author focuses on validating the system's flexibility. Using several scenarios of computational taken position
A design methodology based on a real-life scenario [14]	Using the accelerometer, gyroscope, magnetometer, altimeter, and heart rate monitor	Processing of user localization and detecting fall accident
A fog-based IoT system for remote areas [15]		Using UCI repository and machine learning to predict 8 class cardiovascular disease.
this work	chess strap sensor to obtain ECG and RR interval data	Using two kind classifiers based on machine learning and deep learning to classify 5 class beat type

5.5 Summary

In this chapter, I presented the discussion of our contribution related to the proposal of an IoT framework for continuous heartbeat monitoring and the proposal of a fog computing framework for real-time processing.

Chapter 6

Conclusion and Recommendations

This dissertation presented a proposal for a continuous heartbeat monitoring system using the internet of things approach. This system adopts a wearable device that enables long-term monitoring and provides continuous heart condition monitoring and early detection of cardiovascular disease using the IoT approach.

Firstly, I presented the investigation of continuous data retrieval from wearable devices through the BLE framework. Polar H10 is adopted as the health sensor device which produces several cardiac signs such as RR interval and ECG of a user. I also presented data collection from the device using a BLE framework for long-term data recording. Moreover, a middleware is proposed to answer the interoperability problems in IoT. The middleware is equipped with a BLE interface to build communication with low-energy devices, data standardization using JSON format, and an HTTP interface for forwarding sensor data to other devices such as cloud applications. The result shows the feasibility of the proposed middleware for long-term data collection of cardiac signs from Polar H10 using a BLE framework.

Secondly, I proposed a heartbeat classifier to predict output data from a wearable device where I explored Polar H10 output data as features in the heartbeats classifier. Starting from HRV, RR interval, and ECG morphology features. The classifier proposed to predict heartbeat into five classes, namely, normal beat (N), supraventricular ectopic beat (SVEB), ventricular ectopic beat (VEB), fusion beat (F), and unknown beat (Q), following the described classes by the Association for the Advancement of Medical Instrumentation® (AAMI). Since HRV is not suitable for multi-class classification due to the limited dataset. Thus, I focused on RR intervals for continuous monitoring and real-time prediction as the first classifier. The second classifier is based on ECG morphology to predict ECG signals from Polar H10. The evaluation showed the prediction of the heart condition of the proposal achieved 99.67% and 97% using the RR interval and ECG morphology data, respectively.

Finally, I extended this study by designing and implementing an IoT-based cardiovascular monitoring system based on RRi and ECG data of a wearable device. I proposed two kinds of applications: fog-based and cloud computing. In fog-based applications consist of the sensor and middleware. In comparison, cloud computing-based consists of web applications and data analysis, which runs on a cloud computing environment. The application based on fog computing is provided for real-time or stream monitoring and prediction. The second application is based on cloud computing for multi-analysis and scalability. The system also provides several classifiers based on data type from polar H10 classifiers for RR interval data and ECG data. Moreover, the frameworks are also used to

test the capability of our proposed classifiers. The evaluation conducted on several healthy participants showed the feasibility of the application to provide real-time self-monitoring and prediction of users' heart conditions.

In future works, I would like to extend the implementation for real experimental studies by incorporating a medical professional to identify the type of heart disease and other real-case scenarios where users perform more vigorous activities, such as sports.

Bibliography

[1] Charlton J, Murphy ME, Khaw KT, Ebrahim SB, and Davey Smith G, "Cardiovascular diseases.," 1997.

[2] L. S. Lilly, "Pathophysiology of heart disease: A collaborative project of medical students and faculty," *Pathophysiology of Heart Disease: A Collaborative Project of Medical Students and Faculty*, pp. 1–467, 2015.

[3] R. Gilgen-Ammann, T. Schweizer, and T. Wyss, "RR interval signal quality of a heart rate monitor and an ECG Holter at rest and during exercise," *European Journal of Applied Physiology*, vol. 119, no. 7, pp. 1525–1532, 2019.

[4] Polar Electro, "Polar H10 Heart Rate Sensor System," *Polar Research and Technology*, vol. 1, no. 6, pp. 6–11, 2019.

[5] Polar, "Polar SDK."

[6] P. R. F. Rodrigues, J. M. da Silva Monteiro Filho, and J. P. do Vale Madeiro, "The issue of automatic classification of heartbeats," *Developments and Applications for ECG Signal Processing: Modeling, Segmentation, and Pattern Recognition*, pp. 169–193, 2018.

[7] K. Ashton, "That 'Internet of Things' Thing, in the real world things matter more than ideas," *RFID Journal*, 2009.

[8] E. S. Pramukantoro and A. Gofuku, "A study of Real-Time HRV Analysis Using a Commercial Wearable Device," *ACM International Conference Proceeding Series*, pp. 216–220, 2021.

[9] E. S. Pramukantoro and A. Gofuku, "A heartbeat classifier for continuous prediction using a wearable device," *Sensors*, vol. 22, p. 5080, 7 2022.

[10] E. S. Pramukantoro and A. Gofuku, "A real-time heartbeat monitoring using wearable device and machine learning," *2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 270–272, 2022.

[11] E. S. Pramukantoro and A. Gofuku, "Prototype of multi-layer personal cardiac monitoring system for data interoperability problem," *ACM International Conference Proceeding Series*, pp. 84–89, 2020.

[12] A. H. Omre and S. Keeping, "Bluetooth low energy: Wireless connectivity for medical monitoring," *Journal of Diabetes Science and Technology*, vol. 4, no. 2, pp. 457–463, 2010.

[13] T. Wang, C. Lu, Y. Sun, M. Yang, C. Liu, and C. Ou, "Automatic ECG classification using continuous wavelet transform and convolutional neural network," *Entropy*, vol. 23, no. 1, pp. 1–13, 2021.

[14] D. Dziak, B. Jachimczyk, and W. J. Kulesza, "IoT-based information system for healthcare application: Design methodology approach," *Applied Sciences (Switzerland)*, vol. 7, no. 6, 2017.

[15] F. Jabeen, M. Maqsood, M. A. Ghazanfar, F. Aadil, S. Khan, M. F. Khan, and I. Mehmood, "An IoT based efficient hybrid recommender system for cardiovascular disease," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1263–1276, 2019.

[16] M. S. Hossain and G. Muhammad, "Cloud-assisted Industrial Internet of Things (IIoT) - Enabled framework for health monitoring," *Computer Networks*, vol. 101, pp. 192–202, 2016.

[17] H. Mora, D. Gil, R. M. Terol, J. Azorín, and J. Szymanski, "An IoT-based computational framework for healthcare monitoring in mobile environments," *Sensors (Switzerland)*, vol. 17, no. 10, 2017.

[18] P. Desai, A. Sheth, and P. Anantharam, "Semantic Gateway as a Service Architecture for IoT Interoperability," in *Proceedings - 2015 IEEE 3rd International Conference on Mobile Services, MS 2015*, pp. 313–319, 2015.

[19] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Cla, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[20] J. Tosi, F. Taffoni, M. Santacatterina, R. Sannino, and D. Formica, "Performance evaluation of bluetooth low energy: A systematic review," *Sensors (Switzerland)*, vol. 17, no. 12, 2017.

[21] D. Allen, P. Denning, S. Jolly, M. Sabbouh, and P. Silvey, "Workshop on Web services," 2001.

[22] K. Hinde, G. White, and N. Armstrong, "Wearable Devices Suitable for Monitoring Twenty Four Hour Heart Rate Variability in Military Populations," *Sensors*, vol. 21, p. 1061, feb 2021.

[23] E. S. Pramukantoro, D. Primanita Kartikasari, and R. A. Siregar, "Performance evaluation of MongoDB, cassandra, and HBase for heterogenous IoT data storage," *Proceedings of ICAITI 2019 - 2nd International Conference on Applied Information Technology and Innovation: Exploring the Future Technology of Applied Information Technology and Innovation*, pp. 203–206, 2019.

[24] Bluetooth SIG, "Assigned Numbers is a list of numbers, codes, and identifiers in the Bluetooth specifications. The Assigned Numbers list is updated when new values are assigned.," 2020-11-30. Last accessed 17 September 2020.

[25] Bluetooth SIG, "characteristic heart rate measurement."

[26] A.-A. EC57, "Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," *Association for the Advancement of Medical Instrumentation*, 1998.

[27] J. W. Dukes, T. A. Dewland, E. Vittinghoff, M. C. Mandyam, S. R. Heckbert, D. S. Siscovick, P. K. Stein, B. M. Psaty, N. Sotoodehnia, J. S. Gottdiener, and G. M. Marcus, "Ventricular Ectopy as a Predictor of Heart Failure and Death," *Journal of the American College of Cardiology*, vol. 66, pp. 101–109, jul 2015.

[28] T. Acharya, S. Tringali, M. Bhullar, M. Nalbandyan, V. K. Ilineni, E. Carbajal, and P. Deedwania, "Frequent Atrial Premature Complexes and Their Association with Risk of Atrial Fibrillation," *American Journal of Cardiology*, vol. 116, no. 12, pp. 1852–1857, 2015.

[29] E. J. d. S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey," *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 144–164, 2016.

[30] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ECG arrhythmia classification," *Expert Systems with Applications: X*, vol. 7, 2020.

[31] C. C. Lin and C. M. Yang, "Heartbeat classification using normalized RR intervals and morphological features," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[32] M. G. Tsipouras, D. I. Fotiadis, and D. Sideris, "An arrhythmia classification system based on the RR-interval signal," *Artificial Intelligence in Medicine*, vol. 33, no. 3, pp. 237–250, 2005.

[33] J. Lian, L. Wang, and D. Muessig, "A simple method to detect atrial fibrillation using RR intervals," *American Journal of Cardiology*, vol. 107, no. 10, pp. 1494–1497, 2011.

[34] G. Sannino and G. De Pietro, "A deep learning approach for ECG-based heartbeat classification for arrhythmia detection," *Future Generation Computer Systems*, vol. 86, pp. 446–455, 2018.

[35] A. Tyagi and R. Mehra, "Intellectual heartbeats classification model for diagnosis of heart disease from ECG signal using hybrid convolutional neural network with GOA," *SN Applied Sciences*, vol. 3, no. 2, 2021.

[36] J. F. Saenz-Cogollo and M. Agelli, "Investigating feature selection and random forests for inter-patient heartbeat classification," *Algorithms*, vol. 13, no. 4, 2020.

[37] V. Mondéjar-Guerra, J. Novo, J. Rouco, M. G. Penedo, and M. Ortega, "Heartbeat classification fusing temporal and morphological information of ECGs via ensemble of classifiers," *Biomedical Signal Processing and Control*, vol. 47, pp. 41–48, 2019.

[38] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, pp. 45–50, 2001.

[39] L. G. Tereshchenko and M. E. Josephson, "Frequency content and characteristics of ventricular conduction," *Journal of Electrocardiology*, vol. 48, pp. 933–937, nov 2015.

[40] E. S. Pramukantoro and A. Gofuku, "A study of bluetooth low energy (BLE) frameworks on the IoT based heart monitoring system," *LifeTech 2021 - 2021 IEEE 3rd Global Conference on Life Sciences and Technologies*, pp. 108–110, 2021.

[41] Polar Electro, "what is the bluetooth transmission range of polar h9/h10 hr sensor?."

[42] Apple Developer, "Accessory Design Guidelines for Apple Devices."

[43] P. M. C. Gomes, P. Margaritoff, and H. Silva, "pyHRV: Development and evaluation of an open-source python toolbox for heart rate variability (HRV)," *Proc. Int'l Conf. on Electrical, Electronic and Computing Engineering (IcETRAN)*, pp. 822–828, 2019.

[44] Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology, "Heart Rate Variability : Standards of Measurement , Physiological Interpretation , and Clinical Use," 2014.

[45] F. Shaffer and J. P. Ginsberg, "An Overview of Heart Rate Variability Metrics and Norms," *Frontiers in Public Health*, vol. 5, 2017.

[46] V. Gent, "Heart Rate Analysis for Human Factors: Development and Validation of an Open Source Toolkit for Noisy Naturalistic Heart Rate Data," *Humanist-Vce.Eu*, pp. 13–14, 2018.

[47] J. Pan and Tompkins,W. J, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, pp. 230–236–230–236, 1985.

[48] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[50] L. B. M., N. M. M., N. W. M. João, Souza, M. V. G., P. P. R. F., and V. H. C. A., "A novel electrocardiogram feature extraction approach for cardiac arrhythmia classification," *Future Generation Computer Systems*, vol. 97, no. 1, pp. 564–577, 2019.

[51] S. Chen, W. Hua, Z. Li, J. Li, and X. Gao, "Heartbeat classification using projected and dynamic features of ECG signal," *Biomedical Signal Processing and Control*, vol. 31, pp. 165–173, 2017.

[52] R. Ghorbani Afkhami, G. Azarnia, and M. A. Tinati, "Cardiac arrhythmia classification using statistical and mixture modeling features of ECG signals," *Pattern Recognition Letters*, vol. 70, pp. 45–51, 2016.

[53] H. Huang, J. Liu, Q. Zhu, R. Wang, and G. Hu, "A new hierarchical method for inter-patient heartbeat classification using random projections and RR intervals," *BioMedical Engineering Online*, vol. 13, no. 1, 2014.

[54] Y. Li, Z. He, H. Wang, B. Li, F. Li, Y. Gao, and X. Ye, "CraftNet: A deep learning ensemble to diagnose cardiovascular diseases," *Biomedical Signal Processing and Control*, vol. 62, 2020.

[55] A. Ullah, S. U. Rehman, S. Tu, R. M. Mehmood, Fawad, and M. Ehatisham-Ul-haq, "A hybrid deep CNN model for abnormal arrhythmia detection based on cardiac ECG signal," *Sensors (Switzerland)*, vol. 21, no. 3, pp. 1–13, 2021.

[56] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of arrhythmia by using deep learning with 2-D ECG spectral image representation," *Remote Sensing*, vol. 12, no. 10, 2020.

[57] X. Xu and H. Liu, "ECG heartbeat classification using convolutional neural networks," *IEEE Access*, vol. 8, pp. 8614–8619, 2020.

[58] X. Zhai and C. Tin, "Automated ECG Classification Using Dual Heartbeat Coupling Based on Convolutional Neural Network," *IEEE Access*, vol. 6, pp. 27465–27472, 2018.