International Journal of Artificial Intelligence & Robotics (IJAIR) E-ISSN: 2686-6269
Vol.1 No.1, 2019, pp.16-25

Indonesian Sign Language API (OpenSIBI API) as The Gateway Services for Myo Armband

Moh. Zikky[1]*, Rizky Yuniar Hakkun[2], Buchori Rafsanjani[3]
[1,2,3]Department of Creative Multimedia Technology, Politeknik Elektronika Negeri Surabaya, Indonesia
[1]zikky@pens.ac.id, [2]rizky@pens.ac.id, [3]buchorirafsanjani@yahoo.com
*corresponding author

ABSTRACT

We create an API (Application Programming Interface) for Indonesian Sign Language (SistemIsyarat Bahasa Indonesia/SIBI) which is called OpenSIBI. In this case study, we use the Myo Armband device to capture hand gesture data movement. It uses five sensors: Accelerometer, Gyroscope, Orientation, Orientation-Euler, and EMG. First, we record, convert and save those data into JSON dataset in the server as data learning. Then, every data request (trial data) from the client will compare them using k-NN Normalization process. OpenSIBI API works as the middleware which integrated to RabbitMQ as the queue request arranger. Every service request from the client will automatically spread to the server with the queue process.As the media observation, we create a client data request by SIBI Words and Alphabeth Game, which allows the user to answer several stages of puzzle-game with Indonesian Sign Language hand gesture. Game-player must use the Myo armband as an interactive device that reads the hand movement and its fingers for answering the questions given. Thus, the data will be classified and normalized by the k-NN algorithm, which will be processed on the server. In this process, data will pass OpenAPI SIBI (which connected to RabbitMQ) to queue every incoming data-request. So, the obtained data will be processed one by one and sent it back to the client as the answer.

As the result, beside the advantage of adaptive input of several data resources, we got the effectife time-process with OpenSIBI scheme. It resulted a shortest processing time around 0.113 seconds and the longest time around 0.199 seconds, it was the best time-processing if compared to the previous research. However, it also resulting a good approach in guessing the data input. As the experiment notes, we got 100% correct gesture translation with A - Z Alphabeth and 25 simple words data testing.

Keywords: SIBI API, Indonesian Sign Language, Myo Armband, Sign Language Dataset, OpenSIBI.

I. INTRODUCTION

As the observed system of this research, we made the Learning Game of Indonesian Sign Language using Myo Armband which is integrated with OpenSIBI. OpenSIBI is the OpenAPI managed every request given by the client before being sent to the server. OpenSIBI itself is integrated with RabbitMQ as the message broker in receiving several client's inputs.

Educational Game has the role of a client which will request the server with JSON transmitted data. This game was developed using Unity game engine and integrated with Myo SDK provided by Myo Armband device itself. Meanwhile,on the server-side, we install programs that process the output data of hand movements sent by client using k-NN algorithm. Furthermore, its server also providing the OpenSIBI, which works as the bridge of data requests from clients. Its OpenAPI was also connected to RabbitMQ that works as the message broker which receives every request from several clients in a realtime.

Thus, when the client sends request data, it will flow to the OpenSIBI. Then if the data have several clients that send it at the same time, it will flow the queue section at RabbitMQ which will be processed one by one by the server and sent it back to the client's request as the sign language words data.

A. Myo Armband's Feature Extraction

In the previous work, AnggaRahagiantoetc made the hand gesture classification of Myo Armband data [1]. They captured the Indonesian Sign Language (SIBI) movement and extracted its data with normalization process in five Myo armband's sensors. Thus, all sensor data are processed by neural network algorithm to make easier in matching data with its trial data given.

In their experiment as shownin Figure 1, resulted accuracy of data around 93.08% on average, its experiment used the alphabet A until Z with 260 times (6 times in training for every word/alphabet). So, it means Myo Armband has a powerful sensor in capturing every single gesture movement in the hand and fingers.

 DOI: 10.25139/ijair.v1i1.2026

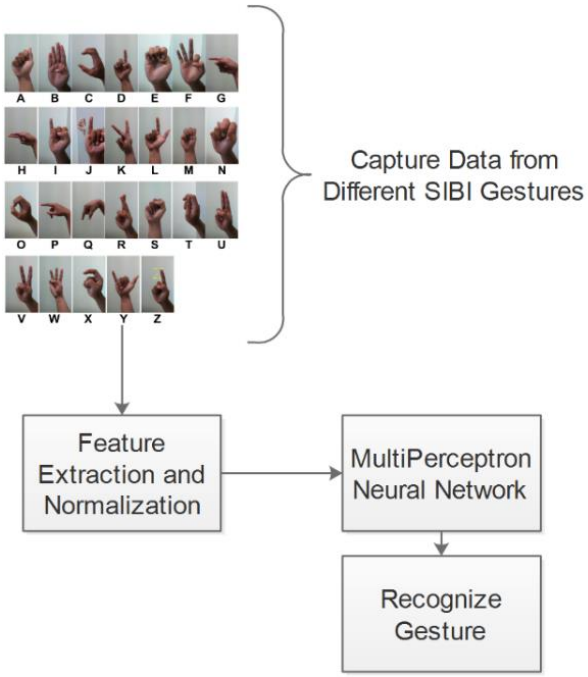


Fig. 1. Extraction Process of Myo Armband Data Sensor [1]

B. Microservice

Microservice is used as the approach feature ofmoduledevelopment in application that arranges a large service request. The implementation of thisOpenSIBI API was aimed to make all commercial hand motion sensor devices are compatible and easy to use with the data embedded, such as MyoArmband data, Leap Motion Controller, Hi5 VR Glove, and so on. The microservice architecture is the right choice to support a variety of commercial devices and it can be adjusted easily with user-friendly UI to communicate with other service sets [2].

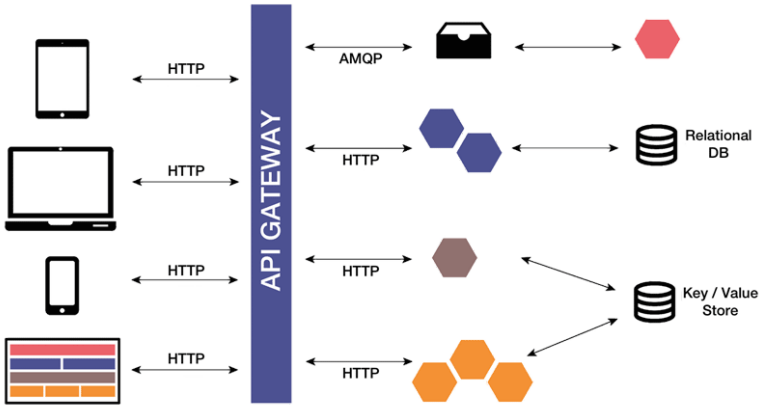


Fig. 2. The example of Microservices Architecture

The Microservice architecture pattern significantly affects the relationship between applications and databases. Instead of sharing a single database schema with other services, each service has its own database scheme. In another perspective, this approach actually is contrary to the idea of an enterprise-wide data model. Thus, it often results data redundancies. However, work with independent database for each service scheme is very important if we want to take a benefit from microservice architecture. We can use the type of database and programming language whatever we want.

So, the main microservice task is dividing the services into smaller parts where the services are interconnected with each other. Thus, the user can use the various technologies in every service created. For implementation, we can't connect several platforms directly, we have to make the API Gateway such as for web access, android, iOS, etc. Its API gateway has tasks such as load-balancing, caching, access control, API metering, and monitoring.

C. RabbitMQ

RabbitMQ is a message broker written with Erlang programming language. Message brokeritself in the programming paradigm is a software thatmanagesthe messages sent by sender to ensure its messagewell- delivered to receiver[3].

DOI: 10.25139/ijair.v1i1.2026

This rabbitMQ can be describedas the post office and its postman. He will receive a message sent by the sender then forward its message to the recipient. Furthermore, that message flows as asynchronous behavior, where the sender does not have to wait a report from the recipient related to its message for effective reason and rabbitMQ can directly handle the other process. So, the sender as simple as sends a message to the message broker, then that message will be handled by the message broker.

The mechanism of the message broker task is sending the message handled by the message to the queue, and its message will be forwarded one by one to the recipient. One advantage of this method is the guaranty for everymessages sent by the sender will be received by the recipient. It is different if we compare tothe synchronous method, where the sender must wait until receivinga response messagefrom receiver, after that a system just can do another process [3]

Fig. 3. RabbitMQ trademark logo

Some terms that frequently used RabbitMQ are:

- *Producer***;** Producing is not different from sending. Logic programming that sends messages is called producers.
- *Queue*; Queue has attributes such as the name for the post box in RabbitMQ. Many producers can send many messages to one queue and also many consumers can receive data from one queue**.**

*Consuming***;** Consuming is the same as accepting. The consumer is a logic programming that waits for receiving messages. As the consumer or broker, both do not have to be on the same host

II. METHOD

SIBI Dataset; a result collection of data training that has been recorded to several previous respondents. To understand the outline of the design system process in this research, Figure 4 shows how every process approach flows and connected each other.

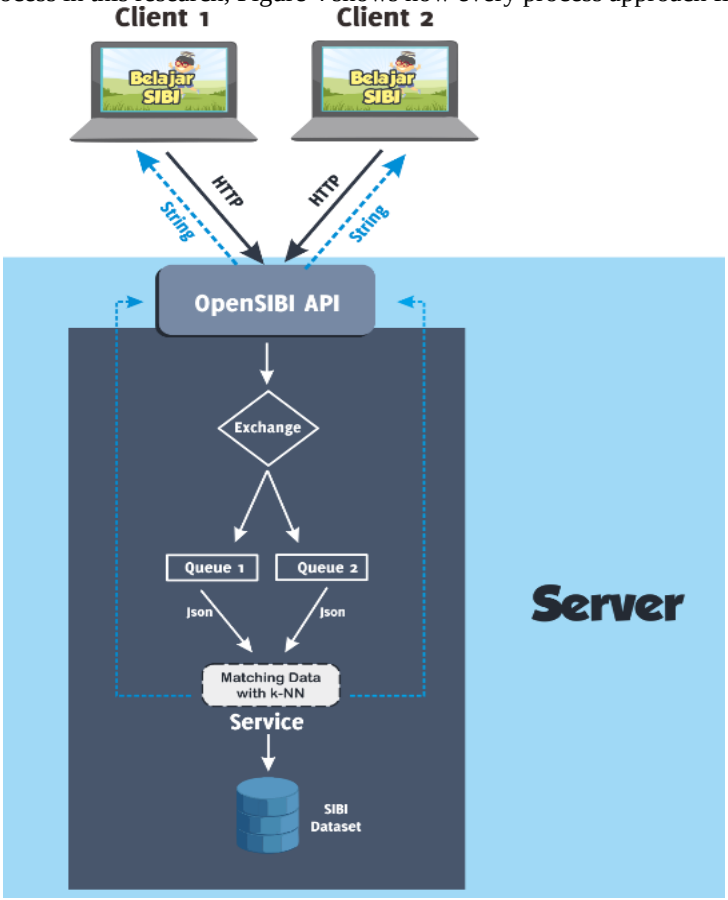


Fig.4. Design System

DOI: 10.25139/ijair.v1i1.2026

As showed in Figure 4, generally the main engine of this system is located in the server. The server must handle the various data input adaptively and it must manages the data request to the right location and time with the effective queued. OpenSIBI API became the essential part because it is the connector to the data input. So in this research, we made this gateway with simple port which can be connected to various devices data. So, if we look the ingredient of this system, they are:

a. Client; this element is an application of educational game SIBI Learning that sends the output data request to the host or server in JSON format.
b. Server; this section contains the processing of data that has been sent by the client. Thus, it is embedded in an OpenSIBI API to bridge requests from clients before processed on the server.
c. OpenSIBI API; an OpenAPI developed to bridge the data sent by the clients. The data requested by the client will be queued at a Message Broker, RabbitMQ. RabbitMQ will process its data in the order of the queue and send it to the desired service, a service here means the process of classification of data using the k-NN algorithm.
d. DataMatching with k-NN; after the queue process, data flow to the k-NN service, then it will be processed and the output will be sent back to the client in a string format.

III. CONFIGURATION, RESULT, AND ANALYSIST

A. Capturing Process of Dataset

In the process of recording the sign language gesture,we captured its hand movement in several respondents. There are several steps of preparation at this recording:

1) Prepare the Myo Armband device

In the first preparation, it must be ensured that the connection between Myo Armband and Laptop/PC is connected properly. In Myo Armband Manager, after clicking the Connect button and the indicator is indicated with green. Thus, there is a notification in the lower right corner for calibration, it indicates that Myo is well connected with the PC/laptop.

2) Calibration for Data Recording

Figure 5 shows the UI program that has been prepared to record hand movement data that will use for SIBI dataset. In this process, the respondent must stand or sit up straight with the right hand clenched position. This condition is used to determine the relative coordinate of hand position's starting point. The example position of calibration as shown in Figure 6.

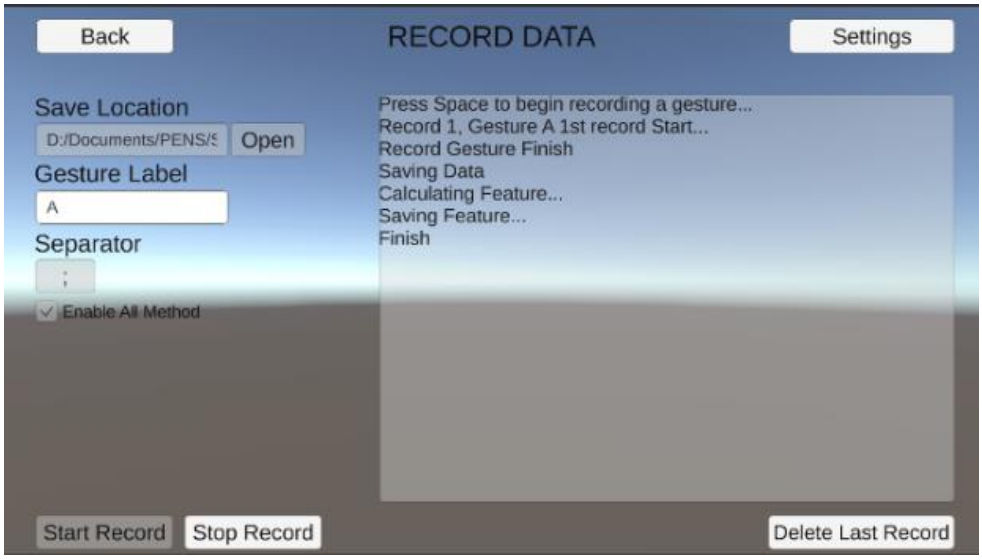


Fig. 5. Interface program for data recording using Unity

DOI: 10.25139/ijair.v1i1.2026

Fig. 6. Hand clenched position when calibrating Myo Armband

3) The Process of Data Recording

For the experiment as an example of recorded data, we record Alphabeth A until Z and 25 simple words as shown in Table I.

TABLE I
LIST OF RECORDED WORDS

Child	Dog	What	Father	Good
Hallo	Mother	Wife	You	Eat
Orange	Uncle	Plane	White	I am
Smell	Blue	Worm	Hey	Who
Why	They	Drink	Grand Mother	Husband
Snake				

Table 1 showed a list of recorded words based on the dictionary of Indonesian Sign Language (SIBI). So the total of alphabet word 26 added by 26 simple words are 52 gestures that stored in the dataset. All movement data both alphabet and simple words are recorded in once time and produce one feature data file. Thus, the feature data is normalized and calculated with min-max process. Until this process, a set of dataset is ready for use.

B. Server Configuration

On the server-side, there are several software installations that must be obtained and configured in advance to ensure the process of sending data runs well.

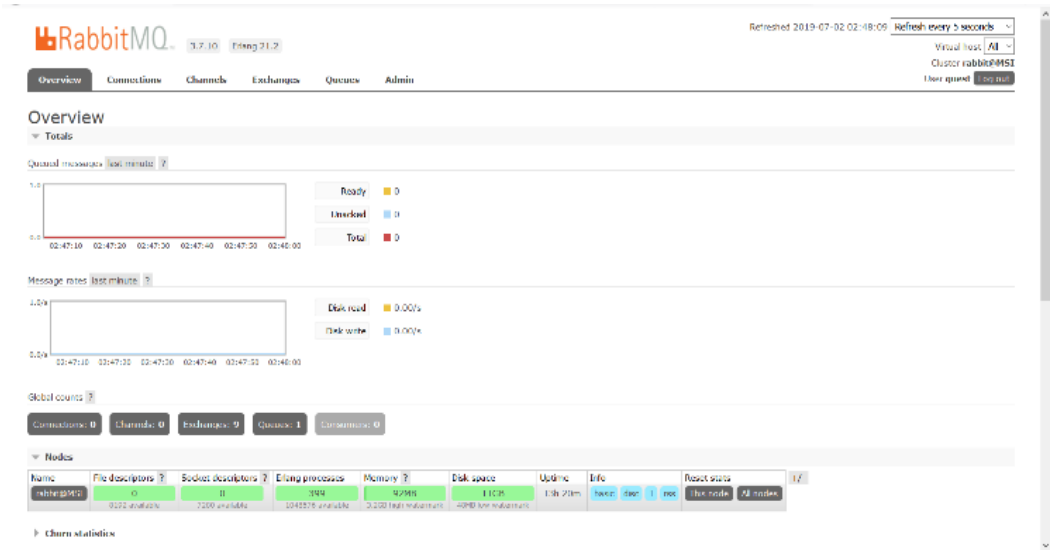
1. RabbitMQ Configuration

RabbitMQ can be installed on the computer server, as usual, another software. But, we also must install Erlang. Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability [4]. In this case, Erlang is needed by RabbitMQ to run this service well.After the installation is complete, there is a configuration in order to be able to access the RabbitMQ Management page. At the beginning of the installation, enable some RabbitMQ plugins with the Command Prompt command below by entering the RabbitMQ installation directory (as default, a directory has existed in C:\program files\RabbitMQ Server\rabbitmq_server-1.7\plugins\). If we have enabled the plugin in the previous process, it will look Figure 7. RabbitMQ Management can be accessed with the following hosts and ports. After this configuration, RabbitMQ will be standby and ready to accept data requests.

```
rabbitmq-plugins enable rabbitmq_management
```

 DOI: 10.25139/ijair.v1i1.2026

 E-ISSN: 2686-6269




```
http://localhost:15672/
Username : guest
Password : guest
```

Fig. 7. RabbitMQ Management User Interface

2. *Service Configuration of k-NN*

K-NN Service is configured to connect a RabbitMQ. Because k-NN, RabbitMQ, and OpenAPI SIBI services will relate to each other.For K-NN service as shown in Figure 8 is used for connecting service with the available host and then connected to RabbitMQ. Thus, RabbitMQ will create a queue that will be used to accommodate requests from the client. After the connection and queue have been created, the dataset is ready to be loaded into this service.

```
0 references
public static void Main(string[] args)
{
    Console.WriteLine(Directory.GetCurrentDirectory());

    Console.ReadLine();

    DataManager manager = new DataManager();

    var factory = new ConnectionFactory() { HostName = "localhost" };
    using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {
        channel.QueueDeclare(queue: "rpc_queue", durable: false,
          exclusive: false, autoDelete: false, arguments: null);
        channel.BasicQos(0, 1, false);
        var consumer = new EventingBasicConsumer(channel);
        channel.BasicConsume(queue: "rpc_queue",
          autoAck: false, consumer: consumer);


        Console.WriteLine("Server Sudah tersambung");


        manager.read_dataset(Environment.CurrentDirectory + "\\f_normalized.csv", ';', out train_values, out train_labels);
        manager.read_minmaxdata(Environment.CurrentDirectory + "\\f_minmax.csv", ';', out train_minmaxdata);


        Console.WriteLine("Sudah load dataset");
```

Fig. 8. The example of k-NN Configuration program

 DOI: 10.25139/ijair.v1i1.2026

3. Web Services and OpenSIBI API Configuration

On the server also prepared a Container that can accommodate the API created. Here the author uses the XAMPP service Web as its container. After the XAMPP installation is complete, copy the OpenSIBI API that was created into the htdocs folder that is in the XAMPP directory. But before Apache on XAMPP is run, the created API must be configured first and have a connection to RabbitMQ. Here the author uses the PHP programming language to build OpenAPI SIBI. The configuration model as shown in Figure 9.

```
class SibiRpcClient
{
    private $connection;
    private $channel;
    private $callback_queue;
    private $response;
    private $corr_id;

    public function __construct()
    {
        $this->connection = new AMQPStreamConnection(
            'localhost',
            5672,
            'guest',
            'guest'
        );
        $this->channel = $this->connection->channel();
        list($this->callback_queue, ,) = $this->channel->queue_declare(
            "",
            false,
            false,
            true,
            false
        );
        $this->channel->basic_consume(
            $this->callback_queue,
            '',
            false,
            true,
            false,
            false,
            array(
                $this,
                'onResponse'
            )
```

Fig. 9. OpenSIBI API Configuration

At the beginning of the connection made, it must ensure that the host, port, username, and password connections are filled correctly and in accordance with the previous service made.

C. Analysist

1. The Observation of Single Client Request

In this observation, we create a generate's random data request that sends by the player. The purpose of this observation is to ensure that every data is sent correctly. Figure 9 shows how the command request processed. This process produces JSON data in the form of featured extraction random based on a dataset that is then sent to the server using HTTP requests with a POST method. When system sends a JSON data, it is also followed by a GUID that is generated randomly in Unity engine. It works to indicate that the data sending and received data are appropriate and not interchangeable with the other client services, mostly in a multi-client manner. In this observation scenario, we run a client and server with localhost in one computer. The results of this observation indicate that the request data from the client is processed properly and the data is returned based on the GUID. It also shows the data returned to the client is the match to what was written in the command prompt on the server.

DOI: 10.25139/ijair.v1i1.2026

Fig. 9.Preview command of observation of single client request

2. The Observation of Multi-Client Request

For further observation, we run two client window processes that built from Unity and run with different data requests. Figure 10 shows how this observation runs. Each process as shown in Figure 10 generates JSON random data based on the dataset and sent it to the server. Every sending data is also identified by an ID to indicate that the data sent and received are appropriate and not exchanged with another client. In this observation scenario, either server or client is still run in one computer. As a result of this observation indicates that the requests data sent by each program are not interchangeable with the other one. And it also shows how independent this client, if one client sends a request, the other client is not influenced by this duty.

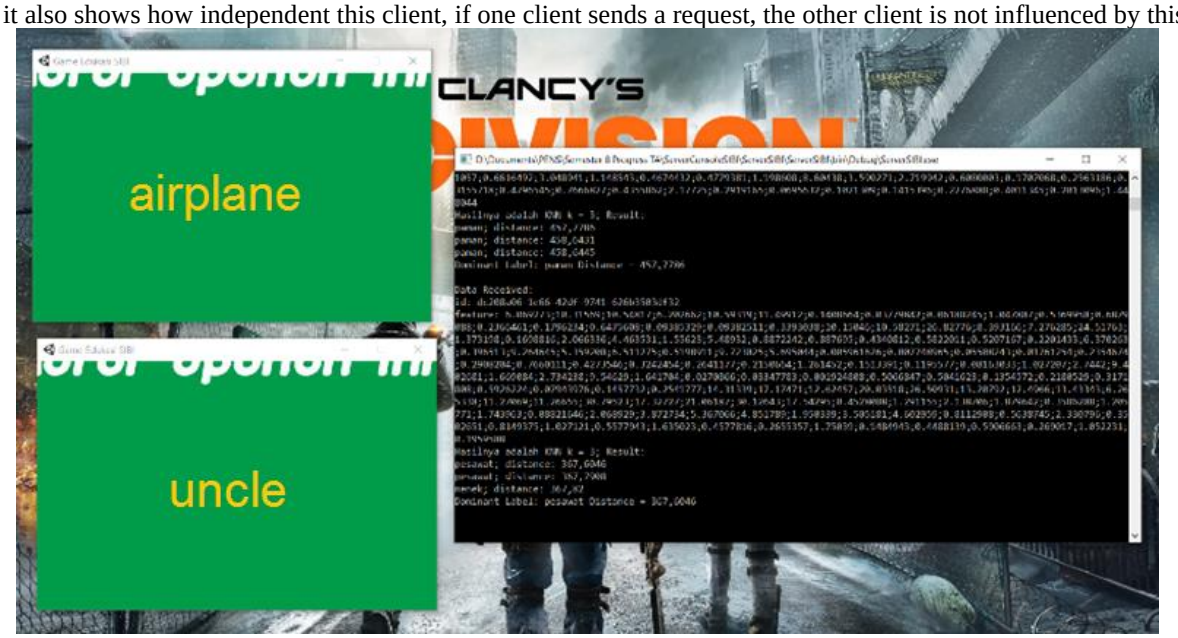


Figure 10. The window observation of double client request

3. The Observation of Application Analysist (Multi Clients)

The next observation is using players who play SIBI sign language. This observation aims to see the performance of the server when receiving many requests from players. In this observation, several client laptops with Myo Armband were used. In this observation, we used one window for server, three players with Myo Armband. Each player or client will try to request data for 10 times in the real-time to test whether each request is processed properly or not.

DOI: 10.25139/ijair.v1i1.2026

Fig.11. Player Observation using Myo Armband (as the Client)

As the result of this observation process as shown in Figure 11, the display of traffic in RabbitMQ also has varied results, which depends on the request from the client and the performance of the internet connection used. If there was a stack of processes, it will be put into the queue first and then processed. Figure 12 shows a RabbitMQ server that handles the incoming request data from the client.

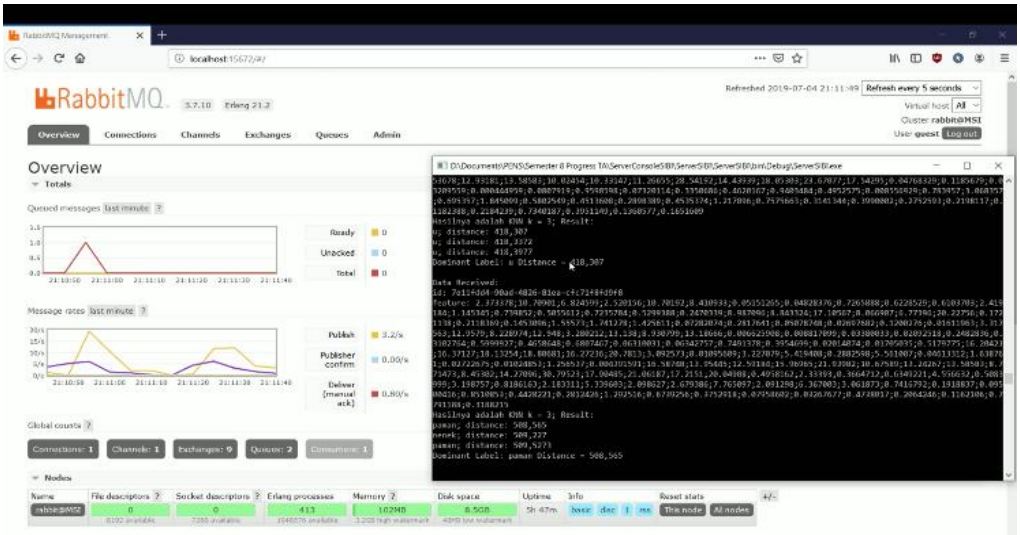


Fig. 12. RabbitMQ server preview that receives the request data from clients

4. *Analysis of Data Request based on Time's Consumed*

As the analysis results showed on Tabel 2, that list has observed from random sign language data. Thus, we collect the consuming time in one request when producing a word or letter from the client.

TABLE II
PERFORMANCE ANALYSIST REGARDING ITS TIME'S CONSUME

No.	*Word*	*Time consumed*	*Status*
1	I am	0,148s	Sent

DOI: 10.25139/ijair.v1i1.2026

No.	*Word*	*Time consumed*	*Status*
2	Blue	0,113s	Sent
3	I am	0,133s	Sent
4	A	0,199s	Sent
5	Blue	0,149s	Sent
6	Who	0,149s	Sent
7	Who	0,118s	Sent
8	Blue	0,133s	Sent
9	Who	0,133s	Sent
10	Uncle	0,147s	Sent
	Average	0,141s	Sent

From the data results in Table 2, it can be concluded that the clients with ten times data requests from Myo Armband have been work successfully by k-NN on the server with the shortest processing time around 0.113 seconds (the word Blue in number 2) and the longest time is letter A with 0.199 seconds. This time is calculated and started after hand movement data is obtained and send it to the server until it backs to the client.

IV. CONCLUSION

This OpenSIBI API has been capable to receive several requests from five clients with Myo Armband in real-time with 100% successful performance. Thus, this performance resulted a shortest processing time around 0.113 seconds and the longest time around 0.199 seconds. So, best on this good result, for further implementation, it can be developed and implemented into various other devices, for example the data development of leap motion controller, Manus Glove VR, and so on. On the server-side, it also can be developed into the other functions, such as hand movements characteristic for post-stroke rehabilitation (hand stroke), simulation of throwing virtual objects, and so on.

REFERENCES

[1] AnggaRahagiyanto, Achmad Basuki, Anwar Raditiya, *Hand Gesture Classification for Sign Language Using Artificial Neural Network*, 2017, PoliteknikElektronika Negeri Surabaya.
[2] Chris Richardson, Pattern: Microservice Architecture,https://microservices.io/, accesed on 22 Juli 2019
[3] Mahesh Singh, RabbitMQ: Understanding Message Broker, www.3pillarglobal.com/, accesed on 22 Juli 2019
[4] Official Web of Erlang Programming Language, *https://www.erlang.org/*, 2018, accesed on 22 Juli 2019.

 DOI: 10.25139/ijair.v1i1.2026