Research Results on System Development of the Research Project of a Self-Study System for Language Learning

Katsuyuki Umezawa
Dept. of Information Science,
Shonan Institute of Technology
Kanagawa, Japan
umezawa@info.shonan-it.ac.jp

Makoto Nakazawa
Dept. of Industrial Information Science,
Junior College of Aizu
Fukushima, Japan
nakazawa@jc.u-aizu.ac.jp

Manabu Kobayashi
Center for Data Science,
Waseda University
Tokyo, Japan
mkoba@waseda.jp

Yutaka Ishii
Faculty of Education,
Chiba University
Chiba, Japan
yishii@chiba-u.jp

Michiko Nakano
Faculty of Education and Integrated Arts and Sciences,
Waseda University
Tokyo, Japan
nakanom@waseda.jp

Shigeichi Hirasawa
Research Institute for Science and Engineering,
Waseda University
Tokyo, Japan
hira@waseda.jp

***Abstract*—This study proposes to develop a self-study system equipped with an artificial instructor who detects and advises learners and evaluates their language learning in a consolidated framework. Detecting the learners mean that the self-study system understands their learning conditions. In this paper, we describe the results of system development among our projects. In particular, we describe a study that detects various errors necessary for a self-study system and propose a system for efficiently collecting EEG (brain waves), which are biological information for detecting the learners.**

***Index Terms*—e-Learning, Self-study System, Language Learning, Simple EEG, Java**

I. Introduction

Recently, with the spread of large-scale open online courses, asynchronous distance learning where learners learn at their own pace has become widespread (Academic eXchange for Information Environment and Strategy [AXIES]). For this reason, there has been an increase in the demand for remote self-study. Nowadays, flipped classrooms are an emergent methodology to increase student independence during conventional classroom lectures. In the flipped classroom, self-study conducted before the face-to-face class plays an important role [1]. Unlike face-to-face learning where the learning time and content can be adjusted by looking at the student's facial expressions and learning attitudes, the current self-study system uses only pre-prepared learning content. In other words, the current self-study system treats all students equally, and it is not possible to provide detailed support according to the learning situation of each learner. We are confident that understanding the learning situation of each student in self-study will improve the quality and effectiveness of online learning.

978-1-6654-8336-0/22/$31.00 ©2022 IEEE

Regarding the target subjects, in 2016, the "School Curriculum Guideline" emphasized the early introduction of English education and programming education toward globalization and internationalization of education [2]. In Japan, programming education is made compulsory at junior high schools, and their government's growth strategy included "promotion of programming education from the compulsory education stage." In addition, the Japan Ministry of Education, Culture, Sports, Science, and Technology has announced that programming education in elementary schools will be compulsory. Also, the Ministry of Internal Affairs and Communications has announced a policy of developing one million new IT human resources by 2025, and they are promoting programming education.

In this research, we develop a self-study system for language learning using an artificial teacher. An artificial teacher or instructor is an agent that collects and analyzes the learner's biological information and learning history and automatically feeds the learner with the analyzed results. This artificial teacher embedded in this self-study system observes and analyzes the learner, presents them with suitable learning content, and gives appropriate instructions.

II. Related work

A. Learning history storage system

It is difficult to know the process of completing a program and what challenges the learner faced and how they solved it in the process. To understand them, it is evident the history of each version is not enough, and effective to record in detail how it has been changing for each operation of the user [3] [4]. Oomori, et al. [3] proposed ways to utilize the accumulated learning history, such as a function to display the operation history, set conditions and filter display items, and display / restore the source code in an arbitrary state in the past. Mori et al. [4] proposed a function that enables objective feedback to the student in question by developing and analyzing their detailed learning history in real-time. This allows the teacher to understand the learner's stumbling point. Despite this, all learners' accumulated history has not been automatically analyzed, and based on analyzed results, the quality needs improvement.

B. Application of EEG measurement to learning

It has been empirically known from psychological and brain science studies that the waveform of EEG can be used as an index of a mental state by observing it with related events [5]. To observe the human mental state, studies have been conducted using the α and the β wave obtained by subjecting the obtained brain waves to the discrete Fourier transform [6]. Researchers such as Giannitrapani et al. [7] have examined the connections between intellectual development and brain waves. They assumed that β waves were highly related to thinking states, and measured brain waves of healthy subjects that were undergoing an intelligence test. It was shown that the low-frequency component (low β wave) of the β wave predominated during the three major tests (reading comprehension, math, and figure alignment test), and the β wave estimated the thinking state. It was also shown to be effective to some extent as an index. Furthermore, we observed the human being's state of thinking and discovered that it was effective to measure the power spectrum of α and β wave, the ratio of α and β waves to the entire brain wave, or the ratio of α wave [8].

III. Overview of our Project

In our project, we develop a self-study system that measures each learner's biological information and learning history and presents learning material suitable for each student's learning conditions. This works even when that student is outside the class, and it is impossible for teachers to assist in real-time. To illustrate our research, we will conduct some experiments and evaluations using the developed system (see Fig. 1).

Our project can be divided into four main sections:

(1) Development of a self-study system equipped with an artificial teacher,

(2) Evaluation by experiment for English and programming languages,

(3) Integrated analysis of a learning log of different languages such as English and programming languages, and

(4) Research on possible substitutes with non-wearable measuring instruments to popularize the system.

As mentioned in (a), we develop a self-study system with the following four functions: (1) a function to judge careless mistakes from brain wave information and answer time, and to ask questions for which it is easy to make such mistakes, (2) a function that determines errors in spelling and grammar (syntax) of words (words in English and reserved words in programming languages) and that asks questions for which it is easy to make such mistakes, (3) a function that determines grammatically correct but logically erroneous answers and that asks questions for which it is easy to make such mistakes, and (4) a function that determines the state of the learner such as "not focusing on learning," "finds learning content too easy," "finds learning content too difficult," and "finds learning content incomprehensible or partially incomprehensible," and adjusts the difficulty level of the task in real-time, combining biological information such as brain waves and eye tracking information from the above (1)-(3) with learning history information.

In (b), the system developed in (a) is evaluated with experiments using English and programming languages. In (c), we perform analysis based on the results of the demonstration experiments to see whether there is any correlation between an individual learner's learning process for English and programming languages and whether synergy effects can be expected to be utilized for education of both topics. In (d), to disseminate our research results, we pursue the possibility of using non-wearable instruments that produce the same results (such as measuring blinks with a web camera) as judgment results with an electroencephalograph (EEG).

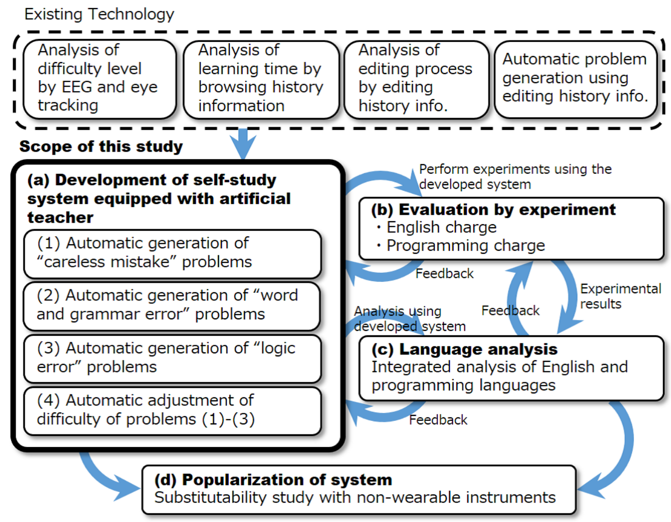

Fig. 1. Overview of our project

IV. Outcomes of our Research

In this study, we focus on the research item (a) shown by the thick line in Fig. 1. In this chapter, we describe the outcomes of (a)-(1) to (a)-(4) in Fig. 1.

A. How to judge careless mistakes

In this section, we describe how to judge careless mistakes based on how answer time and brain waves relate during Java language learning [9].

1) Overview: Three fourth-year students from the Shonan Institute of Technology participated in our research. During the experiment, we measured their brain wave information when they were learning programming. We focused on the relationship between the answer time of the problem and the brain wave and tried to detect any careless mistakes when they lost focus.

2) Experimental method: The learning target of the experiment is the Java language basics. The participants learn seven chapters. In each chapter, materials explain the contents, and 10 questions are included to measure comprehension. First, participants read, and learn the explanatory material of the first chapter. Then the brain wave is measured for five minutes. They answered ten questions about the first chapter. We measured their brain waves during the process. The measurement continued without stopping until the last question. The measurement time remains as a log, as it is possible to cut out the brain waves for each question. This process is repeated from the first to the seventh chapter.

3) Careless mistake judgment method: Here, we define careless mistakes as "wrong answers in a short time" and "wrong answers without thinking," i.e., "wrong answers because they did not answer properly." We proposed a careless mistake estimation method as follows: a) Calculate the average answer time each participant took to answer each question in the experiment. Specifically, we focused on questions learners answered incorrectly within a short time. At this point, the average value of the EEG is referred to while solving the target question. This is aimed at the target average. We also attribute the cumulative average as the average value of the questions from the first question to the preceding question of the target question (not including the target question). b) Compare the target average with the cumulative average and extract the value whose target average > the cumulative average. c) Perform an F-test (test for homogeneity of variance) and a t-test (test for MD) on the extracted average and the corresponding cumulative average. Check whether a significant difference is observed.

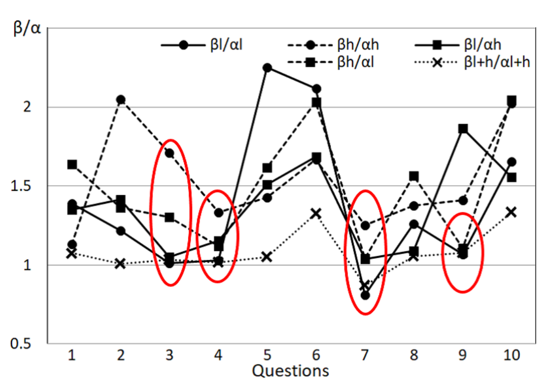

Fig. 2. Average EEG when answering each question in Chapter 6 of Experiment Participant 3

4) Experimental result: The first participant had some wrong answers. It is assumed that he was good at Java programming. The second participant scored a slightly higher number of wrong answers, but we estimated none as careless mistakes. After careful consideration, it is assumed that he erred. Fig. 2 shows the average EEG at the time the third participant answered each question in Chapter 6 of the Experiment. The circled part is where β/α is likely to go down. If this part has an incorrect answer, a short answer time, and a statistically significantly low β/α, it is judged as a careless mistake. In this example, only question 7 was judged to be a careless mistake. The third participant had many wrong answers. The following questions can be estimated as careless mistakes: question 9 in Chapter 2, question 9 in Chapter 4, question 7 in Chapter 5, question 7 in Chapter 6, and questions 3, and 9 in Chapter 8. We found that the third participant was more inclined to make careless mistakes at the end of each chapter.

B. Syntax error detection

In this section, we describe the process used to extract the source code including grammatical mistakes from the editing history data of the programming creation process [10].

1) Overview: A large number of learning logs were accumulated when approximately 90 students of the Shonan Institute of Technology took a 16-week programming class. These learning logs contain all the source code that was adjusted at the end of the program. Based on this information, a debug practice question extraction system that extracts source code containing grammatical errors and automatically generates questions for debug practice is developed.

2) Overall system configuration: We show the overall configuration of the debug exercise extraction system developed this time in Fig. 3. In Fig. 3, the extant editing history visualization system is used until the learning history is developed. The debug exercise extraction tool refers to the learning history developed by the editing history visualization tool, compares the correct source code with the source code, including errors, and extracting them. The extracted source code and errors are referenced and dispersed in the altered version of the editing history visualization system.

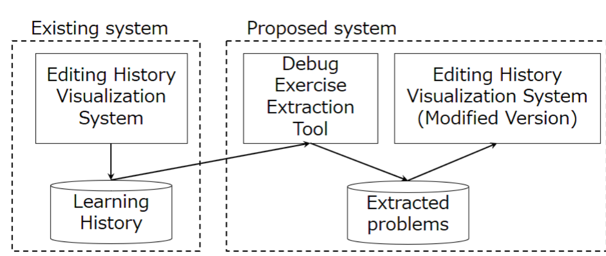

Fig. 3. Overall configuration of debug exercise extraction system

3) Debugging exercise extraction tool: In this section, we compare the debugging exercise extraction tool with the learning history developed by editing the history visualization system. Specifically, MyClass.java exists in the complete folder (last) for each problem, and MyClass.java is in the process of coding. These tools extract the number of errors and the number of misspelled characters for each mistake. The tool reconstructs the folder based on the extracted "number of mistakes" and "number of misspelled characters". Two differences exist between source codes: one is for correcting an error and the other is without error. Therefore, we only extracted the source code for correcting errors. The developed debugging exercise extraction tool is shown in Figure 4. As shown in Figure 4, we can indicate what is to be extracted - how many mistakes, misspelled characters, and which problems of the lesson.

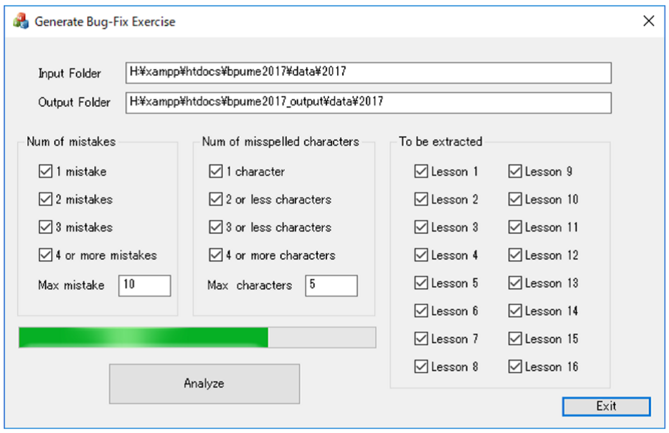

Fig. 4. Debugging exercise extraction tool

4) Extraction Algorithm: The extraction algorithm in this sector shows:

1) Repeat the following for the entire history of the editing history visualization system.
2) Check if last.info contains the word "end".
3) For all folders other than the last, check whether the word "errors" are described in the stdout file.
4) Differentiate between MyClass.java in the folder other than the last that describes the error and MyClass.java in the last folder[1]
5) At that time, it counts the number of mistakes and how many characters are included in one mistake.
6) According to the folder structure, the MyClass.java file is copied depending on the number of mistakes, and the number of misspelled characters contained in one mistake.

C. Logical error detection

This section describes the extraction of the source code including the logical error where the compiler does not output the error information from the editing history data of the programming creation process [11].

[1]In the Java version editing history visualization system, the class name containing the main function is MyClass.java.

1) Overview: The debugging exercise extraction system shown in the previous section targets syntax errors. In this section, the same learning history is used in analyzing the logical error. The compiler does not output error information because logical errors are not grammatical errors. Therefore, it is difficult to detect it mechanically. However, we were able to automatically detect logic errors that are prone to error by accumulating and analyzing a large amount of programming learning history. By this result, the learner can practice debugging to correct the logic error.

2) Class that collected data: We applied the flipped classroom method at our university for a period of16 weeks of actual lessons in three autumn semesters from 2017 to 2019. In this application, we accumulated learning logs from 183 students, 189 students, and 224 students for the three consecutive years, respectively.

3) Analysis results: The source program at that time is accumulated as learning history each time the learner presses the "Build & Execute" button when creating a program that solves these problems. We then analyzed the accumulated 203,436 source files (72,193, 61,721, and 69,522 in 2017, 2018, and 2019, respectively). When solving a task, the expectation is that "Build & Execute" will be performed after the copied source code has been significantly adjusted. Many learners copy and modify the source code of the previous problem. Specifically, when compared with the complete version of the source code, modifications in many places, or of many characters would be unsuitable for logic error analysis. Therefore, we exclude source code from being detected as a logic error when the "number of mistakes" and "number of misspelled characters" are < 10. Table I shows the number of detections for each logical error type and the percentage (%) of the total number of detections annually.

TABLE I
NUNBER OF DETECTIONS AND PERCENTAGE FOR EACH LOGIC ERROR TYPE

Type	Num. of Detections (Percentage %)		
	2017	2018	2019
Spaces	4,189 (19.94)	2,735 (15.12)	2,934 (14.85)
Comments	124 (0.59)	34 (0.19)	289 (1.46)
Strings	2,616 (12.45)	2,675 (14.78)	2,410 (12.20)
Brackets	1,140 (5.43)	1,164 (6.43)	1,228 (6.21)
For statements	2,771 (13.19)	2,223 (12.29)	2,577 (13.04)
While statements	152 (0.72)	108 (0.60)	152 (0.77)
If statements	1,453 (6.92)	1,122 (6.20)	1,384 (7.00)
Else statements	49 (0.23)	31 (0.17)	183 (0.93)
Println	89 (0.42)	101 (0.56)	93 (0.47)
Semicolons	869 (4.14)	649 (3.59)	649 (3.28)
Arrays	371 (1.77)	333 (1.84)	454 (2.30)
Variables	3,215 (15.30)	2,476 (13.68)	2,390 (12.10)
Numerics	1,447 (6.89)	2,006 (11.09)	2,313 (11.71)
Substitutions	279 (1.33)	252 (1.39)	331 (1.68)
Expressions	2,220 (10.57)	2,155 (11.91)	2,333 (11.81)
Other	26 (0.12)	29 (0.16)	40 (0.20)
Total	21,010 (100.0)	18,093 (100.0)	19,760 (100.0)

Table I shows that "Spaces" has the highest number of detections. These detections, along with changes to "Strings," are less important (than other logic error types) for programming comprehension. Many detections associated with programming control structures are “For statements” and “If statements”. Although “While statements” are also control structures, they are used infrequently, and therefore the number of detections is small. However, many detections of changes are registered in “Variables” and “Numbers”. This registration is attributed to the nature of the university lessons, as similar problems are solved successively. Although many “Expressions” are detected, these logic errors may require further analysis. The compiler output no error for those logic errors classified as “Other,” albeit source code contains a correction by double-byte detectable characters.

D. EEG measurement system

This section describes the development of a “Detect server” that measures brain wave information to detect learners and provide questions of appropriate difficulty.

1) Overview: In the conventional measurement method, during learning using a simple EEG, it was necessary for the experiment participants themselves or staff to manually perform the starting and stopping of electroencephalogram measurement. In some cases, the EEG signal was weak, and we did not notice that no data was available. Furthermore, since it is necessary to start the measurement for each participant, there is a drawback that the start and the end are different for each participant. To overcome these drawbacks, we developed an electroencephalogram collection system of conventional electroencephalogram data acquisition methods.

2) Proposed system: The proposed system for overcoming the drawbacks as shown in the previous section is presented in Fig. 5. Within this proposed system, the beginning, and end of EEG measurement can be instructed from the remote management server. Moreover, since the status of brain waves can be confirmed, there is no need for individual staff to support the participants in the experiment. The status of the total EEG can be confirmed, and after confirmation, the start of the acquisition of an electroencephalogram can be instructed. This eliminates the failure of not being aware that brain waves cannot be got (the signal was weak). Furthermore, there is an advantage that the time does not deviate for each data point since the brain wave data is accumulated in the server and the data acquisition time is unified by the server time.

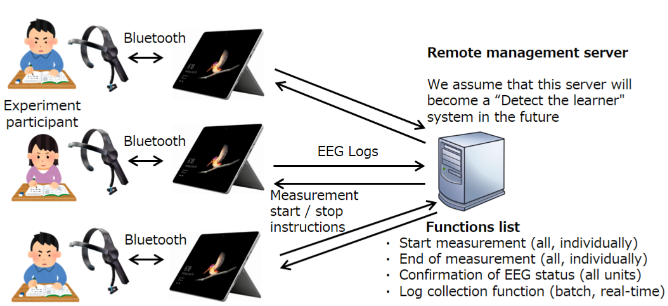

Fig. 5. Proposed EEG data acquisition system

3) Proposed system algorithm: The algorithm of the proposed system is shown in Fig. 6. The experiment participant first executes the login process with the experiment ID and password. The login information is sent to the server, and the connection between the client and the EEG headset is established. The experiment participant acquires brain wave data and status from the EEG headset and sends them to the server, which in turn saves the status in the control table, and saves the EEG data sent from the client in the data table when the measurement status is starting. If the measurement status is stopping, the EEG data sent from the client will be discarded. By this, we confirm the status saved in the server and then instruct to start measurement when the brain wave level of all the experiment participants reaches the normal value. When the time based on the experimental plan elapsed, we send an instruction to stop the measurement.

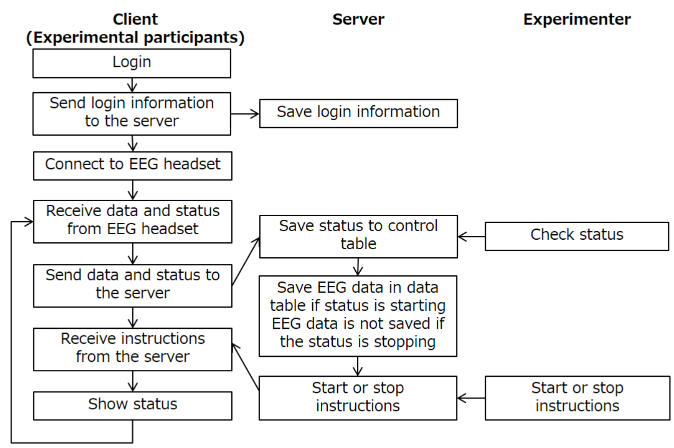

Fig. 6. Proposed EEG data acquisition system algorithm

4) Proposed system user interface: In Fig. 7, we show the startup screen on the experiment participant (user) side. The status of the EEG is displayed every second. A status of 200 indicates “a state in which brain waves are not taken due to poor contact or other reasons,” and 0 indicates a “state in which brain waves are taken normally.”

Fig. 8 shows an example of the status confirmation screen for the experimenter to confirm the EEG acquisition status of all the participants before the start of measurement.

During the experiment, we check the user status with the EegStatus command. From Fig. 8, it is easy to see that the status of user 11111111 is 0, whereas the status of other users remains 200. Also, we confirm that the status of all the members has become 0, and we start the measurement with the start command.

5) EEG data acquired and stored in the database: Fig. 9 shows the EEG data collected from the simple EEG to the server via the network and stored in the database. Observe that multiple types of brain waves (delta waves to high Gamma waves) of multiple experiment participants (1111111, 22222222, etc.) have been acquired.

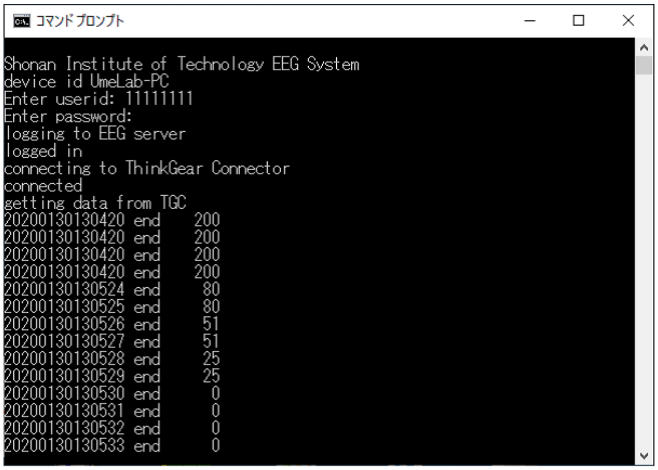

Fig. 7. User-side startup screen of the proposed system

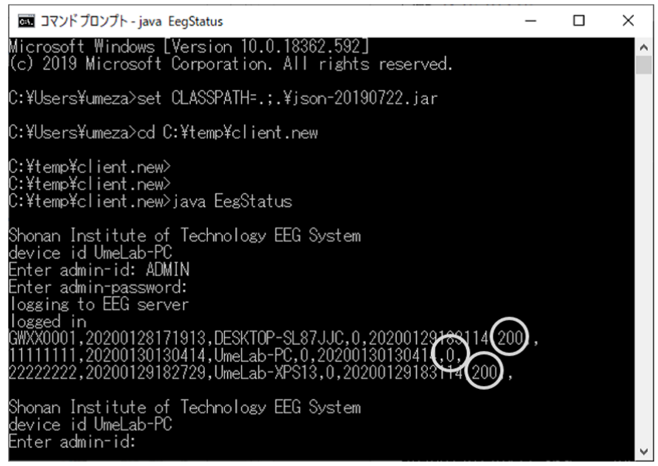

Fig. 8. Proposal system status confirmation screen

sdccrtdt	attension	meditation	delta	theta	lowAlpha	highAlpha	lowBeta	highBeta	lowGamma	highGamma	HostSABSNo	DevId
2020-01-29 18:17:43	57	61	366064	32937	12909	12179	6119	5337	4320	599	11111111	UmeLab-PC
2020-01-29 18:17:44	41	44	142244	68443	2487	5417	10939	13522	4533	3628	11111111	UmeLab-PC
2020-01-29 18:17:45	47	44	674140	42364	5187	15516	2325	10148	13378	2418	11111111	UmeLab-PC
2020-01-29 18:30:45	0	0	1681000	1202219	526665	234242	214614	578354	492786	567781	22222222	UmeLab-XPS13
2020-01-29 18:17:46	57	34	64750	64942	5764	6066	16259	34241	23844	17508	11111111	UmeLab-PC
2020-01-29 18:30:46	100	20	1172069	729880	91692	50703	37211	97643	70474	19012	11111111	UmeLab-PC
2020-01-29 18:30:46	0	0	425373	548621	249872	234027	39364	143406	135219	158474	22222222	UmeLab-XPS13
2020-01-29 18:30:47	78	48	868616	164951	112798	37869	5301	13078	27162	6401	11111111	UmeLab-PC
2020-01-29 18:30:47	0	0	33001	48136	19512	74989	25638	34913	28268	36047	22222222	UmeLab-XPS13

Fig. 9. EEG data acquired and stored in the database

V. Conclusion and Future Work

In this study, we succeeded in determining careless mistakes, extracting grammatical errors, and analyzing logical errors as part of our research on self-study systems for language learning. Furthermore, we developed a system that can efficiently collect brain waves from the participants of the experiment.

In the future, we plan to research evaluation experiments and integrated analysis of English and programming languages as well as research on possible substitutes with non-wearable measuring instruments to popularize the system.

Acknowledgement

Part of this research result was carried out as a part of the research project "Research on e-learning for next-generation" of Waseda Research Institute for Science and Engineering, Waseda University. Part of this work was supported by JSPS KAKENHI Grant Numbers JP21K18535, JP20K03082 and JP19H01721, and Special Account 1010000175806 of the NTT Comprehensive Agreement on Collaborative Research with Waseda University Research Institute for Science and Engineering. The research leading to this paper was partially supported by the grant as a research working group "ICT and Education" of JASMIN.

References

[1] K. Shigeta, "Flipped Classroom: Educational reform utilizing information technology," Jour nal of Information Processing and Management, vol.56, no.10, p.p. 677–683, (2014).

[2] Ministry of Education, Culture, Sports, Science and Technology, "What are you doing new and will continue to focus on?," Detailed contents of the 2017/2018 revised school curriculum guideline, (Mar. 2017). https://www.mext.go.jp/a_menu/shotou/new-cs/1383986.htm

[3] T. Omori and K. Maruyama, "A Method for Extracting Source Code Modifications from Recorded Editing Operations," Journal of the Information Processing Society of Japan (IPSJ), Vol.49, No.7, p.p. 2349–2359, (2008).

[4] K. Mori, T. Tanaka, H. Hashiura, A. Hazeyama, and S. Komiya, "Development of an Environment for Gleaning History Information of Programming on a Fine Granularity to Help with Programming Exercise Lesson," Technical Reports of the IPSJ, 2013-SE-179, vol.16, p.p. 1–6, (2013).

[5] N. Yoshimine "Mystery of the brain : Why the brain-waves oscillate rhythmically?," The bulletin of Tama University, p.p. 93–100, (2017).

[6] H. Berger "On the electroencephalogram in man," Archiv fur Psychiatrie and Nervenkrankheiten, 87, p.p. 527–570, (1929).

[7] D. Giannitrapani, "The role of 13-hz activity in mentation," The EEG of Mental Activities, p.p. 149–152, (1988).

[8] K. Yoshida, Y. Sakamoto, I. Miyaji, and K. Yamada, "Analysis comparison of brain waves at the learning status by simple electroencephalography," KES'2012, Proceedings, Knowledge-Based Intelligent Information and Engineering Systems, p.p. 1817–1826, (2012).

[9] K. Umezawa, M. Nakazawa, M. Kobayashi, Y. Ishii, M. Nakano and S. Hirasawa, "Detection of Careless Mistakes during Programming Learning using a Simple Electroencephalograph," Proceeding of the 15th International Conference on Computer Science and Education (IEEE ICCSE 2020), p.p. 72–77, (Aug. 2020).

[10] K. Umezawa, M. Nakazawa, M. Goto and S. Hirasawa, "Development of Debugging Exercise Extraction System using Learning History," Proceeding of the 10th The International Conference on Technology for Education (T4E 2019), p.p.244–245, (Dec. 2019).

[11] K. Umezawa, M. Nakazawa, M. Kobayashi, Y. Ishii, M. Nakano, and S. Hirasawa, "Analysis of Logic Errors Utilizing a Large Amount of File History During Programming Learning," Proceeding of the IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE2020), p.p. 232–236, (Dec. 2020).