Development of Voice-Controlled Automatic Chessboard Games for People with Physical Disabilities

Ferdiansyah Farandi[1], Azzam Muzakhim Imammuddin[2*], Putri Elfa Mas'udia[3]

[1,2]Digital Telecommunication Network Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

[3]Telecommunication Engineering Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

[1]2041160003@student.polinema.ac.id, [2]azam@polinema.ac.id, [3]putri.elfa@polinema.ac.id

***Abstract*— The development of chess board games in the field of technology focuses more on artificial intelligence, but less on people with physical disabilities, especially those with motor impairments who rely on human assistance to move pieces. The method in this research is Hidden Markov model used to calculate the probability of voice recognition on Pocketsphinx and Web Speech API. Voice input uses a filtered microphone which is first filtered by Pocketsphinx to generate commands to move the robot arm which utilizes a stepper motor device to move the pieces programmed on the Raspberry pi device and second filtered by the Web Speech API for reading the location of the pieces and notation using chess.js and chessboard.js with simulation. In ten trials of voice recognition Pocketsphinx and Web Speech API showed good to excellent results with percentage of success and WER values of 94% and 5.78% respectively for Pocketsphinx, and 98.75% and 99% on Web Speech API with WER values of 1.25% and 0.83% respectively in normal and eating movement conditions. The robot arm shows 100% accuracy in three trials with constant time and varies according to the notation distance from the initial position of the robot arm.**

Keywords*— Chess Board Game, Disabilities, Pocketsphinx, Voice Control, Web Speech API.

I. INTRODUCTION

Chess is a strategy game and is one of the oldest games in the world, it is also very popular around the world from small children to adults playing this game. The chess game consists of two players who each hold black and white pieces that are moved alternately.

Technological developments also influenced the development of technology in chess games, this technological development began in the 1700s [1], although it was later discovered that the machine was fake, but the machine helped popularize and encourage research and technological development in chess games. The development of chess on computer games first began [2].

Chess games are growing until now, but many of our friends have difficulties with physical constraints [3], the development carried out is always focused on AI to achieve victory in each game, but not focused on people with disabilities, especially people with disabilities with high levels of motor impairment must rely on the help of others to move their pieces on the chessboard according to their instructions which can reduce the level of fairplay in the game also can result in a reduction of the feeling of independence and self-esteem [4].

In previous research entitled "Development of Automatic Chess Board Games Controlled by Voice for People with Physical Disabilities" is considered to be raised in this study. In this research I combine previous research related to the use of voice input in chess game simulations as a basis for positioning chess pieces in the development of this chess board game, then the use of a robot arm was chosen because the explanation of people with physical disabilities as described in the previous point, was chosen in moving this chess piece, using the working principle of a 3d printer and combined with an electromagnet device as a substitute for a hand that functions to take chess pieces that have been attached to a small iron plate at the top.

All of these devices are controlled through a microphone which is a device that can be attached to mobile phones and Tablets in an effort to improve the quality of recorded data from sound sources [5]. The microphone voice input sensor module which is processed using Pocketsphinx which is used to move the robot arm by relying on very fast response and minimal resource consumption, thus enabling the development of speech recognition suiTable for desktop applications, command and control and dictation applications [6]. The Web Speech API is a collection of client-side JavaScript functions embedded in modern web browsers [7]. it allows web developers to leverage the latest machine learning-based speech recognition technologies, in this case such as controlling a chess simulation through voice recognition from the library on a Raspberry Pi, which is one of the most popular Single Board Computer (SBC), with the low-cost SBCs bring together external hardware, sensors and controller interfaces, with user-friendly programming capabilities, high connectivity, and desktop functionality. It is hoped that people with physical

*Corresponding author

disabilities will be helped and make it easier for them to play chess games with this development.

The stepper motor driver will translate the command in the form of a pulse signal into a movement by a stepper motor, which is one type of drive that is widely used in CNC Engraving machines, 3D printing, and conveyor systems that require precision [9]. The robot will use 3 Axis stepper motors, which are used in machines that function as miniature CNC (Computer Numerical Control) machines, or machines that aim to create movement in the form of patterns or characters to move the arm [10]. It uses three stepper motor components on the x, y, z axes to support the operation of the voice-controlled chess robot arm, utilizing a Cartesian model that is based on the orthogonal x, y, z axes, combined with stepper motors to drive this 3D printer [11]. The DRV8825 is controlled by a step/direction interface. The step/direction interface takes a square wave and a logic high/low signal to control the speed and direction of the motor, respectively [12]. The rising edge of the square wave moves the stepper motor one step, and therefore, the frequency of the square wave determines the speed at which the motor turns [12].

Automatic Speech Recognition (ASR) systems are evaluated based on their accuracy in transcribing spoken language into text. The Hidden Markov Model (HMM) includes several symbols that represent different components in its formulation as shown in equation (3), Where M is the HMM, S is the set of HMM hidden states, O is the set of observation vectors, A is the set of jump probabilities from the self-hop hidden state and the next jump, and B is the observed probability density [13].

The Word Error Rate (WER) is a commonly used metric for evaluating ASR systems. It is the ratio between the number of substitution, insertion, and deletion errors in a hypothesis and the number of words in a reference [14]. A WER of 5-10% is considered good quality and ready to use. A WER of 20% is accepTable, but you may need to consider doing more training. A WER of 30% or more signifies poor quality and requires customization and training [15].

II. METHOD

This research uses the type of research making / development. Where the answer to the problem formulation needs the stages of making, designing, preparing tools and materials, parameters to form a feasible automatic chess board game.

A. Hardware System Block Diagram

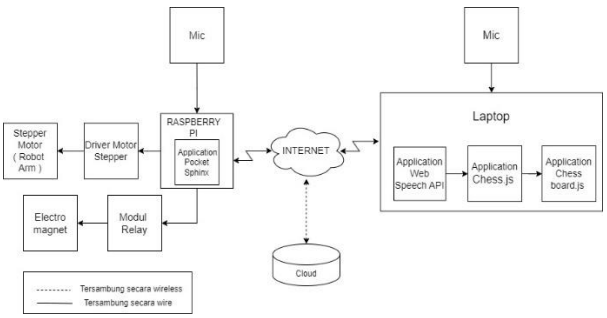

Figure 1. The System Block Diagram

Based on the system block diagram Fig. 1, it can be explained into several parts. There is two microphone as voice input connected to the Raspberry pi and laptop.

The laptop receives digital signal input data that is passed to the Web Speech API. The cloud database on the audio web speech API requires an internet network on the laptop to access it on the cloud database. The audio data is managed and matched by the Web Speech API then converted into text which is forwarded to chess.js. Then in chess.js is interpreted in a chess move in the form of FEN (Forsyth-Edwards Notation) format which is then sent to chessboard.js. In chessboard.js the display is visualized based on the FEN (Forsyth-Edwards Notation) format. Furthermore, the laptop and raspberry are connected to the internet network intermediary on wifi, to run the program manually in the initial position on the robot arm. Then on the Raspberry Pi the voice input is forwarded to pocketsphinx which calls the stepper motor program to forward the appropriate rotation command on the chess coordinates to the stepper motor driver. The stepper motor driver will translate the command in the form of a pulse signal into a movement by stepper motor which is one type of drive that is widely used in CNC Engraving machines, 3D printing and conveyor systems that require precision. The pocketsphinx is also used to call the electromagnet program which functions to send signals to the relay module. Then the relay module translates the signal into a motion that activates and turns off the electromagnet device.

This system integrates the Web Speech API as voice input, converting spoken commands into textual representations of chess moves. These textual inputs are then processed and rendered on a virtual chessboard interface, providing a simulated environment for gameplay and strategic exploration. The system are recognized by Pocketsphinx and translated into a motor control commands. Stepper motors are utilized to precisely position a robotic arm, while electromagnets are activated to moving chess pieces on the chessboard. This integration of virtual and physical components through voice control creates an inclusive system, allowing individuals with disabilities to engage in chess gameplay.

B. System Flowchart

The design of voice recognition relies on the Web speech API and Pocket Sphinx, where the Web Speech API as a web-based voice recognition so that it can be utilized to integrate the use of voice recognition into web page applications. Pocket Sphinx is an open-source library that is flexible in speech recognition because it can be tailored to specific needs, plus its use does not require the internet. The similarity between the two uses the Hidden Markov Model method which uses a language model to calculate each probability of a word that might be spoken. As an illustration from the beginning of the voice can be up to a command on the web speech API and pocket sphinx in the form of a flowchart as follows:

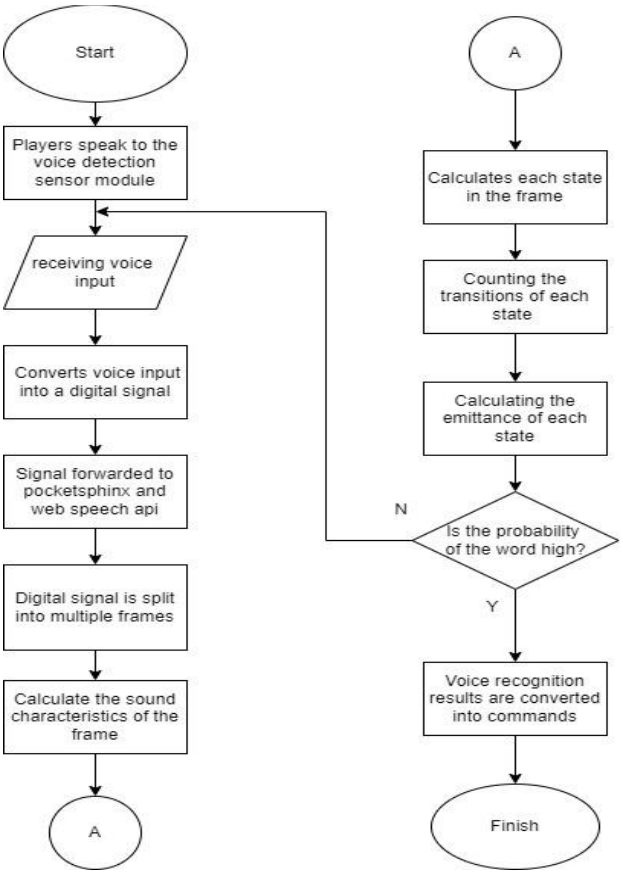

Figure 2. HMM Flowchart

Based on the system block diagram Fig. 2, the process begins when players start talking into a dynamic microphone, which receives the voice input. This type of microphone operates by having sound pressure move a membrane connected to a coil of wire; the membrane's movement produces changes in the magnetic field, which are then converted into electrical signals by an electronic circuit. These electrical signals are subsequently transformed into digital signals, which are forwarded to Pocket Sphinx and the Web Speech API. To facilitate probability calculations, the digital signal is split into small frames. Then, calculating the voice characteristics of the frames, the calculated characteristics are frequency, and spectrum. The calculation of voice characteristics involves extracting features from the processed voice signal using the Mel-Frequency Cepstral Coefficient (MFCC), namely frequency (Y(k)), is the frequency domain signal after processing Fast Fourier Transform (FFT) which is expressed as follows in equation (1).

$$Y(k) = \sum_{n=0}^{N-1} y(n) \cdot e^{\frac{-2j\pi kn}{N}} \quad (1)$$

Spectrum, serves as the energy spectrum of the frequency domain signal after the filtering process. Expressed as follows in equation (2).

$$S_{(m)} = ln(\sum_{k=0}^{N-1} P(\omega) \cdot Hm(k)) \quad (2)$$

After that, calculating each state in the frame is in the HMM determining which state most likely represents the sound in the frame. Expressed as follows in equation (3).

$$M = \{S, O, A, B, \pi\} \quad (3)$$

Where M is the HMM, S is the set of HMM hidden states, O is the set of observation vectors, A is the set of jump probabilities from the self-hop hidden state and the next jump, B is the observed probability density [8]. The value π = is the initialization probability value of the HMM state, aij is the transition probability. Calculate the transition or pause at each state to another state. With the formula expressed as follows in equation (4).

$$a_t(i) = \sum_{j=1}^{N} a_{t-1}(j) \cdot a_{ji} \cdot b_i(o_t) \quad (4)$$

Where N is the total number of states is the transition probability from state j to state i and is the emission probability from state i to observation at time t. Calculate the emittance of each state, which is the emittance of the possibility to produce a certain output from one state. With the formula expressed as follows in equation (5).

$$b_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j) \quad (5)$$

Where is the emission probability from state j to observation at time t + 1. If the probability is high with the command that I want in chess notation to move the chess or simulation, it will be converted into a command. The corresponding speech recognition result is converted and forwarded into a command.

C. *Simulation Flowchart*

The flowchart illustrates the process of reading the position of chess pieces and generating notation using chess.js and chessboard.js for simulation as shown in Fig. 3.

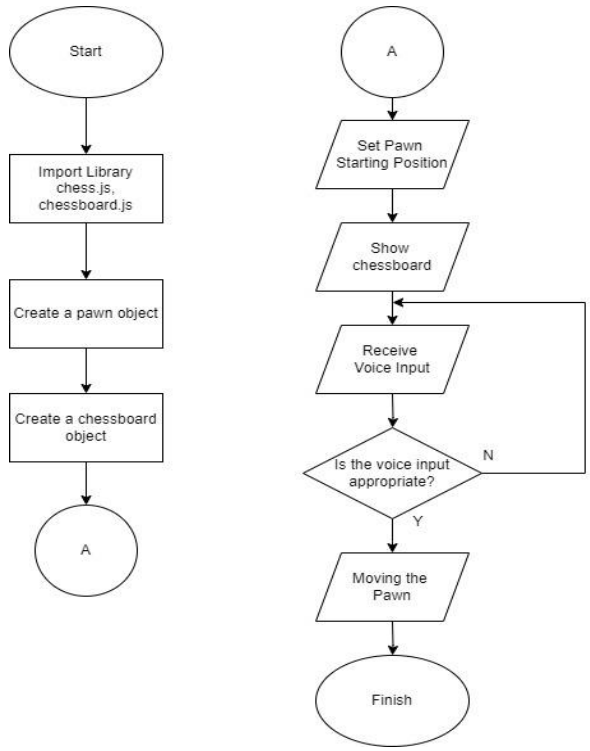

Figure 3. Simulation Flowchart

This flowchart in Fig. 3 shows the steps in the program on the chessboard game simulation, First the program flow starts by importing the libraries needed to run the program. These libraries include libraries to handle the chessboard, pieces, and

moves. Second Defining the piece object in chess.js, namely creating a piece object for type and color, then creating a movement object for the initial position of the piece and the destination position of the piece and other movements such as promotion. Next, specifying objects on chessboard.js, namely creating board objects on the board will consist of a combination of letters and numbers. After that, in the initial position set section, create a program to set the initial position of the pieces on the chessboard according to international chess rules. Then, receive voice input into the microphone which will be selected. After that, Selecting whether the sound captured by the microphone is a movement command or not, if so then it is forwarded to determine the movement, if the sound input is not appropriate then the program will wait until there is a new input that is appropriate. Last, determine the movement and move the chess piece according to the command that enters the voice input.

D. Robot Arm Flowchart

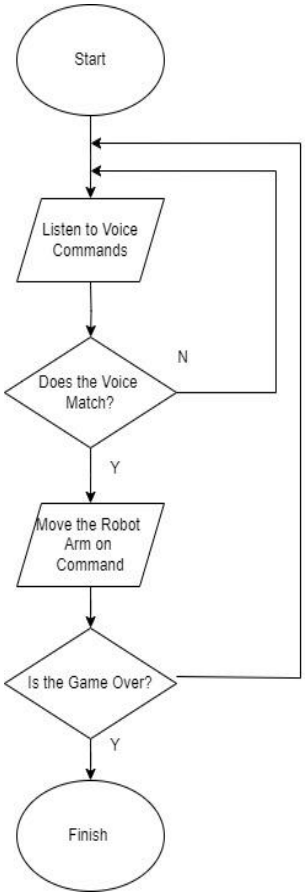

Figure 4. Robot Arm Flowchart

The flowchart in Fig. 4 refers to robot arm for the steps described as follows. First, listen to voice commands using the microphone sensor to listen to voice commands from the user. The microphone sensor is a sound sensor that can detect sound waves. This sensor will convert the sound waves into electrical signals. This electrical signal will then be processed by the robot arm to be identified as a voice command. Second, by using the web speech API algorithm to process the electrical signal from the microphone sensor. The web speech API algorithm is an algorithm that can be used to recognize voice commands. This algorithm will compare the electrical signal from microphone sensor with the database of voice commands that have been trained. So that it can find out whether the voice is appropriate or not. If the electrical signal from the microphone matches one of the voice commands in the database, the Web Speech API algorithm will return a "true" value and move the robot arm according to the player's command and vice versa, if the electrical signal from the microphone does not match one of the voice commands in the database, the Web Speech API algorithm will return a "false" value and return to listening to the voice input until the voice command matches. Then, move the robot arm according to the command i.e. if the value returned by the Web Speech API algorithm is "true", then the robot will move its arm according to the voice command received. The robot will use a 3 Axis stepper motors are used in machines that function as miniature CNC (Computer Numerical Control) or machines which aims to create movement in the form of patterns or characters to move the arm. Next, it will check whether the game is over, if the player gives keyboard input to stop the process then it is considered to finished the game.

E. Hardware Circuit Schematic

This schematic circuit illustrates how the various components connect and interact to form a chess robot system that can be operated via voice commands. Raspberry Pi plays a role in processing voice commands and controlling physical actuators on stepper motors and electromagnets to move the chess pieces on the board. the overall automatic chess board game aid shown in Figure 5 as follows:

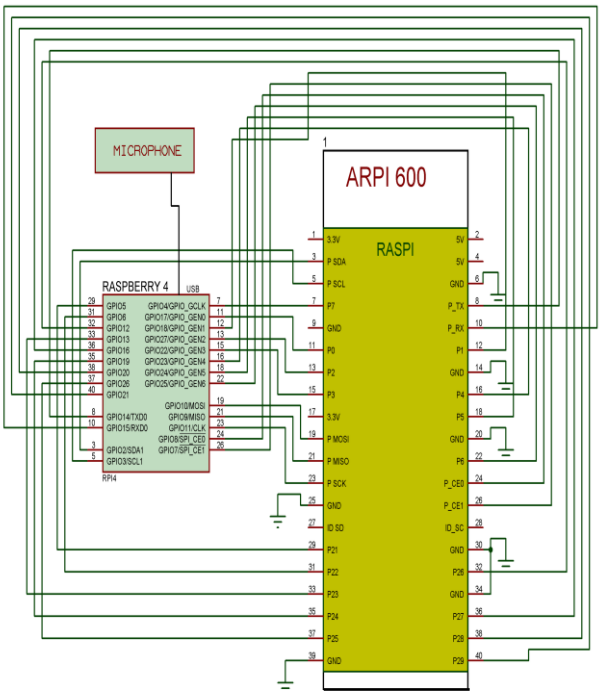

Figure 5. Robot Arm Flowchart

The schematic in Fig. 5 shows the implementation of the technology designed to enable people with physical disabilities to play chess independently through voice commands. The main components in this system include a microphone as the voice input, a Raspberry Pi 4 as the main processing unit and as the controller of the physical actuators.

The microphone captures the user's voice command, which is then processed by the Raspberry Pi using Pocketsphinx for voice recognition. The recognized commands are passed through the ARPI 600 to the CNC shield, which will control the NEMA 17 stepper motor and electromagnet.

The use of the DRV8825 motor driver ensures precise control over the motion of the stepper motor, while relays regulate the activation of the electromagnet to lift and place the chess pieces in a normally closed condition. The DRV8825 is controlled by a step/direction interface and The step/direction interface takes a square wave and a logic high/low signal to control the speed and direction of the motor, respectively. The rising edge of the square wave moves the stepper motor one step. Therefore, the frequency of the square wave determines the speed at which the motor turns.

III. RESULTS AND DISCUSSION

This chapter presents the comprehensive results derived from the software and hardware design implementations. It is includes the outcomes of sensor testing and their subsequent application.

A. Mechanical Design Implementation Results

Figure 6. Chess Board Design

The design in Fig. 6 uses three stepper motor components on the x, y, z axes to support the operation of the voice controlled chess robot arm utilizes a cartesian model that is based on the orthogonal x, y, z axes combined with using stepper motors to drive this 3d printer.

B. Chess Position Simulation Implementation Result

In Fig. 7 shows the display of the chess position where there is a chess board, then a description of the chess notation and the status of the player who must walk which will alternate when the input is appropriate and there is a separate button for start stop to turn on and off the microphone that captures voice input from the user.

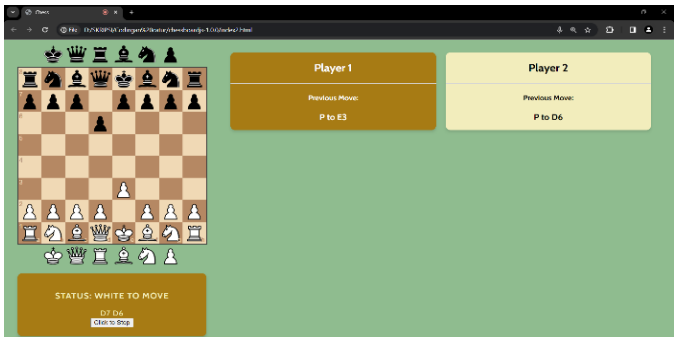

Figure 7. Simulation Design

C. Overall Test Result

The test uses chess notation codes and basic moves in chess without en passant and castling rules where each square on the chess board is given a certain code, namely letter codes a, b, c, d, e, f, g, h, and number codes 1, 2, 3, 4, 5, 6, 7, 8, which are then combined to form a coordinate that is used in making it easier to track the position of the pieces in the game, learn tactics, and understand the mistakes made , as shown in Fig. 8.

8	a8	b8	c8	d8	e8	f8	g8	h8
7	a7	b7	c7	d7	e7	f7	g7	h7
6	a6	b6	c6	d6	e6	f6	g6	h6
5	a5	b5	c5	d5	e5	f5	g5	h5
4	a4	b4	c4	d4	e4	f4	g4	h4
3	a3	b3	c3	d3	e3	f3	g3	h3
2	a2	b2	c2	d2	e2	f2	g2	h2
1	a1	b1	c1	d1	e1	f1	g1	h1
	a	b	c	d	e	f	g	h

Figure 8. Chess Notation

Then the code that determines the pieces and how each piece moves according to the basic moves in chess without en passant or rokade rules is as follows. The King with the symbol “K”, the King can move one square in any direction (front, back, left, right, diagonal). Queen with symbol “Q”, the Queen can move freely in any direction (front, back, left, right, diagonal) with unlimited number of plots. Rook / Rook with symbol “R”, Rook can move freely in any direction (front, back, left, right) with unlimited number of plots. Bishop / Minister with symbol “B”, the Minister can move freely in any diagonal direction with unlimited number of plots. Knight / Horse with symbol “N”, the Horse has a unique movement in the shape of the letter “L” and the Horse moves two plots in one direction (front, back, left, right), then one plot in the perpendicular direction (diagonal). Pawn / Warrior with the symbol “P”, the Warrior can move one square forward. On his first move, the soldier has the option to move two squares forward. The soldier can also capture an opponent's piece that is one square ahead diagonally.

By understanding these piece and movement codes, the determination and reading of piece positions are more readable and easy to analyze.

Testing the Accuracy of Web Speech API on Chess Positions

This test is conducted to test the accuracy of the web speech API to recognize the voice and the similarity of the output that appears. Where this test is divided into several parts, first the normal movement (movement other than eating and promotion) and the movement of eating other pieces and both movements are divided into two for each black and white color between pieces and each section contains commands, output, and accuracy. The accuracy of the test will produce different values for each individual based on various factors, one of which is clarity of pronunciation, pauses in mentioning between notations, as well as factors of surrounding environmental conditions and noise levels as shown in Table I as follows:

TABLE I
SIMULATION TEST

Normal Move White Pawn		
No	**Accuracy**	**Total**
1	100 %	23
2	90 %	0
3	80 %	1
Normal Move Black Pawn		
No	**Accuracy**	**Total**
1	100 %	22
2	90 %	2
3	80 %	0
Black Pawn White Pawn		
No	**Accuracy**	**Total**
1	100 %	22
2	90 %	1
3	80 %	1
Black Pawn Black Pawn		
No	**Accuracy**	**Total**
1	100 %	22
2	90 %	2
3	80 %	0

Testing the Accuracy of Pocketsphinx on Robot Arm

Testing of voice input processing on Pocketsphinx is done ten times using the chess notation voice commands that have been programmed, resulting in output output will be in the form of text that is the same as the spoken voice commands.

The test in Table 2 shows the measurement of the level of accuracy calculated from the percentage of pocketsphinx voice commands that have a base design in English which has been adjusted to the use of Indonesian words in the voice command. This is due to the use of the words a and e which are difficult to recognize in English on Pocketsphinx, so in order to improve the accuracy of voice recognition on Pocketsphinx, the word has been modified so that pronunciation in Indonesian can be recognized properly as shown in Table II as follows:

TABLE II
POCKETSPHINX TEST

No	Voice Recognition	Output	Accuracy
1	ahsatu	Moving A1	100 %
2	baysatu	Moving B1	100 %
3	cheysatu	Moving C1	100 %
4	daysatu	Moving D1	100 %
5	ehsatu	Moving E1	100 %
6	efsatu	Moving F1	100 %
7	geysatu	Moving G1	100 %
8	hasatu	Moving H1	100 %
9	ahdua	Moving A2	100 %
10	baydua	Moving B2	100 %
11	cheydua	Moving C2	100 %
12	daydua	Moving D2	100 %
13	ehdua	Moving E2	100 %
14	efdua	Moving F2	100 %
15	geydua	Moving G2	100 %
16	hadua	Moving H2	100 %
17	ahtiga	Moving A3	100 %
18	baytiga	Moving B3	100 %
19	cheytiga	Moving C3	100 %
20	daytiga	Moving D3	100 %
21	ehtiga	Moving E3	100 %
22	eftiga	Moving F3	100 %
23	geytiga	Moving G3	100 %
24	hatiga	Moving H3	100 %
25	ahempat	Moving A4	100 %
26	bayempat	Moving B4	100 %
27	cheyempat	Moving C4	100 %
28	dayempat	Moving D4	100 %
29	ehempat	Moving E4	100 %
30	efempat	Moving F4	100 %
31	geyempat	Moving G4	100 %
32	haempat	Moving H4	100 %
33	ahlima	Moving A5	100 %
34	baylima	Moving B5	100 %
35	cheylima	Moving C5	100 %
36	daylima	Moving D5	100 %
37	ehlima	Moving E5	100 %
38	eflima	Moving F5	100 %
39	geylima	Moving G5	100 %
40	halima	Moving H5	100 %
41	ahenam	Moving A6	70 %
42	bayenam	Moving B6	80 %
43	cheyenam	Moving C6	70 %
44	dayenam	Moving D6	90 %
45	ehenam	Moving E6	90 %
46	eflima	Moving F6	90 %
47	efenam	Moving G6	90 %
48	haenam	Moving H6	70 %
49	ahtujuh	Moving A7	90 %
50	baytujuh	Moving B7	90 %
51	cheytujuh	Moving C7	90 %
52	daytujuh	Moving D7	90 %
53	ehtujuh	Moving E7	90 %
54	eftujuh	Moving F7	90 %
55	geytujuh	Moving G7	90 %
56	hatujuh	Moving H7	90 %
57	ahdelapan	Moving A8	70 %
58	baydelapan	Moving B8	90 %
59	cheydelapan	Moving C8	90 %
60	daydelapan	Moving D8	90 %
61	ehdelapan	Moving E8	80 %
62	efdelapan	Moving F8	70 %
63	geydelapan	Moving G8	80 %
64	hadelapan	Moving H8	90 %

The results are still generally positive with accuracy predominantly reaching 100%. A drop in accuracy occurred in some tests with 90% accuracy in a quarter of the total trials, as well as a drop in accuracy, with 80% and even 70% in some very rare cases. Despite significant noise interference from the surrounding environment, the device was still able to produce accurate data in most cases with a total average of 94%. This shows that the quality of the tool is still high even when faced with diverse environmental challenges.

Testing the Accuracy of Robot Arm

This test involves the calculation of time assisted using a stopwatch in units of seconds in three trials. In this test running the movement of chess pieces, the robot arm operates using servo motors regulated by drv8825 with the results shown in Table III:

TABLE III
ROBOT ARM ACCURACY TEST

No	Command	Output	Movement Time	Accuracy
1	A1	Moving A1	46.61	100 %
2	B1	Moving B1	37.74	100 %
3	C1	Moving C1	31.71	100 %
4	D1	Moving D1	25.86	100 %
5	E1	Moving E1	20.59	100 %
6	F1	Moving F1	15.76	100 %
7	G1	Moving G1	10.00	100 %
8	H1	Moving H1	5.13	100 %
9	A2	Moving A2	53.05	100 %
10	B2	Moving B2	44.18	100 %
11	C2	Moving C2	38.15	100 %
12	D2	Moving D2	32.30	100 %
13	E2	Moving E2	27.03	100 %
14	F2	Moving F2	22.30	100 %
15	G2	Moving G2	16.44	100 %
16	H2	Moving H2	11.57	100 %
17	A3	Moving A3	57.47	100 %
18	B3	Moving B3	48.60	100 %
19	C3	Moving C3	42.57	100 %
20	D3	Moving D3	36.72	100 %
21	E3	Moving E3	31.45	100 %
22	F3	Moving F3	26.62	100 %
23	G3	Moving G3	20.86	100 %
24	H3	Moving H3	15.99	100 %
25	A4	Moving A4	62.49	100 %
26	B4	Moving B4	53.62	100 %
27	C4	Moving C4	47.59	100 %
28	D4	Moving D4	41.74	100 %
29	E4	Moving E4	36.47	100 %
30	F4	Moving F4	31.64	100 %
31	G4	Moving G4	25.88	100 %
32	H4	Moving H4	21.01	100 %
33	A5	Moving A5	69.90	100 %
34	B5	Moving B5	61.03	100 %
35	C5	Moving C5	55.01	100 %
36	D5	Moving D5	49.15	100 %
37	E5	Moving E5	43.88	100 %
38	F5	Moving F5	39.05	100 %
39	G5	Moving G5	33.29	100 %
40	H5	Moving H5	28.42	100 %
41	A6	Moving A6	74.78	100 %
42	B6	Moving B6	65.91	100 %
43	C6	Moving C6	59.88	100 %
44	D6	Moving D6	54.03	100 %
45	E6	Moving E6	48.76	100 %
46	F6	Moving F6	43.93	100 %
47	G6	Moving G6	38.17	100 %
48	H6	Moving H6	33.30	100 %
49	A7	Moving A7	81.76	100 %
50	B7	Moving B7	72.89	100 %
51	C7	Moving C7	66.86	100 %
52	D7	Moving D7	61.01	100 %
53	E7	Moving E7	55.74	100 %
54	F7	Moving F7	50.91	100 %
55	G7	Moving G7	45.15	100 %
56	H7	Moving H7	40.28	100 %
57	A8	Moving A8	88.62	100 %
58	B8	Moving B8	79.73	100 %
59	C8	Moving C8	73.71	100 %
60	D8	Moving D8	67.85	100 %
61	E8	Moving E8	62.58	100 %
62	F8	Moving F8	57.75	100 %
63	G8	Moving G8	51.99	100 %
64	H8	Moving H8	47.12	100 %

The role of current here in the drv8825 stepper motor driver is also very important where if the current is higher, the movement of the tool is getting faster which will reduce the travel time of the tool on each notation, while a lower current will make the stepper motor unresponsive or restrained in its movement due to the load given, but overheating will be an obstacle that can damage the stepper motor and other components, so it is necessary to match the current with the needs and not make the driver component too hot. In this test, the x-axis of the stepper motor driver uses a current input of 1.43 A with a multimeter tool

The current on the x-axis used in the stepper motor of 1.43 A where the current value of 1.43 A is still the maximum current limit that can be received by the stepper motor nema 17hs4401s which is 1.7 A based on the datasheet.

The results in Table 3 of the robot arm have an accuracy that reaches 100%. This means that all 64 commands given were successfully executed correctly. Nevertheless, there is a difference in variation in the various times required by the robot to complete each command, with an average time of 44.40 seconds.

As for the speed, some commands were successfully completed in a fast time, such as H1 which only required 5.13 seconds considering that in Figure 4.9 the H1 notation of the robot arm movement is very close to the initial position of the robot arm, followed by G1 which required 10.00 seconds, and H2 which required 11.57 seconds. Meanwhile, there are certain commands that take longer than average, with the A8 command being the longest with a time of 88.62 seconds, followed by A7 for 81.76 seconds, and B8 with a time of 79.73 seconds which is influenced by the initial position of the robot arm walking. Overall, the performance of the robot arm is arguably consistent in achieving high accuracy, although there are still variations in the time it takes to execute each command.

World Error Rate (WER)

The Word Error Rate (WER) is an industry standard for measuring speech recognition models. WER counts the number of incorrect words identified during recognition, and divides that number by the total number of words provided in the human-labeled transcript (N), as for the WER formula as follows in equation (6).

$$WER = \frac{I+D+S}{N} \times 100\ \% \qquad (6)$$

The words identified in the formula fall into three categories, Insertion (I) refers to words that were added incorrectly in the hypothesis transcript, Deletion (D) denotes words that were not detected in the hypothesis transcript, and Substitution (S) pertains to words that are substituted between the reference and the hypothesis. The quotient of these discrepancies is multiplied by 100 and displayed as a percentage of analyzable results.

A WER of 5-10% is considered good quality and ready to use. A WER of 20% is accepTable, but you may need to

consider doing more training. A WER of 30% or more signifies poor quality and requires customization and training.

The Word error rate (WER) is a commonly used metric for evaluating ASR systems. It is the ratio between the number of substitution, insertion and deletion errors in a hypothesis and the number of words in a reference. Based on the input voice of the database used in the chess board game, the input voice only has one word, so it is necessary to simplify the formula for the three categories with one category wichis substitution, that shows the results recognized as false or true which is shown in Table IV.

The results in Table IV show how many percentages appear so that it can be categorized as Webs Speech API-based chess simulation control in normal moving and eating conditions, namely 1.25% and 0.83% are categorized as very good quality. As for Pocketsphinx-based robot arm control with a percentage result of 5.78% is in the good quality category.

TABLE IV
ROBOT ARM ACCURACY TEST

Chess Simulation Control (Web Speeh API) Normal Moving and Eating Moving Conditions			
No	**Number of Tests/Word**	**Total Error**	**WER Percentage**
1	240	3	1.25 %
2	240	2	0.83 %
Controlling the Robot Arm (Pocketsphinx)			
No	**Number of Tests/Word**	**Total Error**	**WER Percentage**
3	640	37	5.78 %

IV. CONCLUSION

After testing and analyzing the device that has been made, it can be concluded that controlling chess pieces using Pocketsphinx to detect user voice input through a microphone produces positive performance with an average recognition accuracy of 94% and WER of 5.78% in various situations in 10 trials. The chess simulation developed utilizing the chess.js and chessboard.js libraries for visualization and piece movement, as well as the Web Speech API for voice recognition, showed piece movement and voice recognition accuracy of 98.75% and 99%, respectively, with WER of 1.25% and 0.83%, in 10 trials. The robot arm, which is operated using a servo motor governed by drv8825, showed movement accuracy reaching 100%, although there is a slight variation in the time to execute commands, overall the performance is consistent with the initial position, especially in controlling the z-axis.

REFERENCES

[1] Ranganathan, G., "AN ECONOMICAL ROBOTIC ARM FOR PLAYING CHESS USING VISUAL SERVOING", Journal of Innovative Image Processing, pp 141, India, Juni, 2020.

[2] Vickers, J., "Man vs Moore's Law: The declining relationship between chess and artificial intelligence 1950-1997", Best Undergraduate Dissertations of 2021, pp 5, Inggris, 2021.

[3] Parwata, I.G.H., Putra, I.K.G.D., Sutramiani, N.P., "Penerapan Web Speech API pada Game Catur Berbasis Suara", pp 22, Bali, April, 2019.

[4] Pozzi L., "A Robotic Assistant for Disabled Chess Players in Competitive Games", International Journal of Social Robotics, pp 1-2, 2023.

[5] Bhakti, M. A. C., "Analisis Perbedaan Tekanan Suara di Mikrofon Internal dengan Mikrofon Eksternal pada Aplikasi Soundmeter berbasis Ponsel", Journal of Computer Science, 2021.

[6] Aditya, R., Muid, A., Ristian U., "Tempat Sampah Otomatis Speech Recognition Menggunakan Pocketsphinx", Jurnal Ilmiah Ilmu Komputer Mulawarman, pp 39-42, Pontianak, 2020.

[7] Lonescu T. B., Schlund, S., "Programming Cobots By Voice: A Human-Centered, Web-Based Approach", Journal of Human-Machine Interaction Group, Vienna, 2020.

[8] Gamess, E., Hernandez, S., "Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests", International Journal of Advanced Computer Science and Applications, pp 819, New York, 2022.

[9] Wibowo, B. C., Nugraha, F., "Stepper Motor Speed Control Using Start-Stop Method Based On PLC", Jurnal Teknik Elektro dan Komputer, Kudus, 2021.

[10] Suhendra, A., "Stepper Motor Speed Control Using Start-Stop Method Based On PLC", pp 25, Batam, April, 2021.

[11] Wicaksono, R. A., "Rancang Bangun dan Simulasi 3D Printer Model Cartesian Berbasis Fused Deposition Modelling", Jurnal Engine: Energi, Manufaktur, dan Material, pp 53-55, Pontianak, 2021.

[12] Carillo, L., McMahon, "3D Printing by Selective Laser Sintering (SLS) ", Journal of Dept. of Electrical and Computer Engineering, 2022.

[13] Chen, Y., "A HIDDEN MARKOV OPTIMIZATION MODEL FOR PROCESSING AND RECOGNITION OF ENGLISH SPEECH FEATURE SIGNALS", Journal of Intelligent Systems, pp 717-719, Shanghai, China, April, 2022.

[14] Park, C., Lu, C., Chen, M., Hain, T., "Fast Word Error Rate Estimation Using Self Supervised Representations For Speech And Text", Speech and Hearing Research Group, 2023.

[15] Microsoft, "Test accuracy of a custom speech model", <URL: Test accuracy of a custom speech model - Speech service - Azure AI services | Microsoft Learn, January, 2024.