International Journal of Engineering and Management Research
e-ISSN: 2250-0758 | p-ISSN: 2394-6962
https://ijemr.vandanapublications.com

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

Wireless Control System for A Smart Robot

Shrikanth Shirakol[1], Apeksha Konnur[2], Nikhil V Jujaganv[3], Revanasiddappa[4] and Rakshita Tembadmani[5]
[1]Assistant Professor, Department of Electronics and Communication, SDM College of Engineering and Technology, INDIA
[2]Student, Department of Electronics and Communication, SDM College of Engineering and Technology, INDIA
[3]Student, Department of Electronics and Communication, SDM College of Engineering and Technology, INDIA
[4]Student, Department of Electronics and Communication, SDM College of Engineering and Technology, INDIA
[5]Student, Department of Electronics and Communication, SDM College of Engineering and Technology, INDIA

[1]Corresponding Author: shrikanthks09@gmail.com

Received: 19-03-2024 Revised: 05-04-2024 Accepted: 21-04-2024

ABSTRACT

This paper delineates the development of an Android mobile application tailored for manual control of a robot vehicle via wireless Bluetooth technology. The application prioritizes user interaction through both visual and voice-based commands, offering a versatile control interface. Through the graphical interface, users can actively monitor the robot's proximity to obstacles, facilitated by an integrated ultrasonic sensor positioned at the robot's front. The necessity for a mobile robot prototype based on a differential gear system was paramount to the application's evolution. The project's core objective lies in achieving precise control through voice commands, employing an Android application integrated with a microcontroller. The seamless connection between the Android app and the robot vehicle is established through Bluetooth technology, granting users control via on-screen buttons or spoken commands. A preliminary preparatory session is integral to ensuring the smooth operation of the robot, with a dedicated code employed for instructing the controller effectively. This innovative amalgamation of innovative technology and user-friendly control mechanisms holds promising applications across various domains, from industrial automation to assistive technologies, underscoring the project's potential for real-world impact.

Keywords— Android Application, Bluetooth, MIT App Inventor, Robot Control, Ultrasonic Sensor

I. INTRODUCTION

A novel Android application has been developed for the manual or automatic control of a 4-wheel-drive robot. Bluetooth technology facilitates seamless communication between the smartphone and the robot. The application, crafted using MIT App Inventor, comprises three primary sections: one for establishing Bluetooth connectivity, exiting the application, and resetting the microcontroller card; a manual control interface featuring directional, speed, LED, and fan controls; and an automatic control interface for resetting the position and configuring target axis and speed parameters. Testing procedures involved the Arduino Uno microcontroller and HM-10 Bluetooth module to validate precise information processing. [1].

The study explores the potential of smartphones and Bluetooth in robotics. Bluetooth, developed by Ericsson in 1994, facilitates file transfer and data exchange within an 8–10-meter range, making it suitable for home use.

The research emphasizes smartphones as versatile communication hubs. Speech recognition is introduced as a user-friendly method for robot control, enhancing multitasking and user convenience. The system operates independently of the robot's main intelligence, enabling effortless control through voice commands. The research underscores the growing need for precise and efficient robot control methods in the advancing field of robotics. The study also delves into closed-loop systems using speech recognition for commanding a robot. Hardware considerations include robot mechanical design, motor selection, and electronic devices, while software involves algorithms for converting words to target points and controlling precise robot movement. [3].

Additionally, a voice-controlled robot with a camera is introduced. The Android smartphone translates voice commands to text, which is communicated to the robot via Bluetooth. The robot, equipped with a camera, scans surroundings by rotating 360 degrees, calculates distances, and executes movements based on user commands [2].

In conclusion, the research contributes to robotics by presenting a comprehensive Android app for manual and automatic control, integrating Bluetooth communication and speech recognition for an enhanced user experience. The study's focus on efficient control mechanisms aligns with the rapidly evolving landscape of robotic technology [4].

Human beings are eagerly engaged in the quest for innovative means of interfacing with machines. One such endeavour is the development of a smart vacuum

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

cleaner, aimed at streamlining the cleaning process for users. This project encompasses a fusion of hardware and software components, including a microcontroller, motor shield, sensor array, an Android application, and a Bluetooth module facilitating communication between the hardware and software components.[10]

II. METHODOLOGY

The following parts are included in the circuit that was built.

The Arduino Uno is a single-board microcontroller that facilitates easy application access with a variety of interactive items and their surroundings [8]. The Arduino Uno microcontroller board is based on the ATmega328. It features an ICSP header, a power jack, a reset button, six analog inputs, four digital I/O pins, ceramic resonators operating at 16 MHz, and a USB port. It includes all of the support that the microcontroller needs. It is easily connected to a laptop with a USB cord [4]. It can be powered by a battery or a two AC to DC adaptor to get going. Because of the serial driver chip, the Uno is not like any other board because it does not utilize the FTDI USB. Instead, it has the Atmega16U2 (Atmega8U2 up to version R2) configured as a USB to serial converter. The Arduino Integrated Development Environment (IDE) is a cross-platform Java program It has a code editor that highlights syntax, matches braces, and automatically indents code. It can also compile and post programs on the board with a single click. We refer to an Arduino software or code as a "sketch."

The HC-05 Bluetooth module is a serial port protocol module, meaning that it communicates with the microcontroller serially. There are other gadgets with Bluetooth connectivity, such as the HC-06 and HC-05. In comparing the two modules, the HC-06 can only function as a slave, whereas the HC-05 can operate as both a slave and a master, allowing it to both accept and provide connections to other devices. We are utilizing it in slave mode for our project, which implies it will accept the connection from the Android app [4]. It runs on a (3.3-6) V supply, thus we need to utilize a voltage divider (the Rx pin of the Bluetooth module supports 3.3V) when the Tx pin of the Arduino board connects to it. If we don't, the module would just crash. Since the module accepts the exact same power source as the Arduino board, there is no such constraint for the Tx pin. Either "0000" or "1234" is the default password for the module, which must be used to connect it to the Android app [8]. The purpose of the L298 H Bridge Motor Driver is to regulate the rotation of the DC motors. In other words, it is an H-bridge motor driver that gives our robotic car direction. L298 operates between 5 and 35 volts. The motors A and B are connected to two screw terminal blocks in the module. There is also a screw terminal block for the ground pin, the motor's VCC, and a 5V pin that can be used as an input or output [12].The power source we are using will determine this; if it is less than or equal to 12 volts, it will offer output; if it is greater than or equal to 12 volts, it will require input to prevent other components from burning out. This aids in regulating the motor driver's pace as well; when a jumper is present, the motor driver runs at its fastest [5]. Alternatively, in the event that the jumper is not present, the motor speed can be controlled by connecting a PWM input (0-255) numeric value.

Nearly all mechanical motions that we observe in our surroundings are powered by electric motors. Energy conversion can be accomplished by electrical machines. Both mechanical and electrical energy are produced by motors. One of the most fundamental parts of the hundreds of household appliances we use on a daily basis is the motor driver. Motors are used in many different applications, such as hydroelectric power generators, cars, food blenders, and many more [2].

2.1 App Based Control

Using a mobile device to operate a robot like a smartphone or a tablet that runs the Android operating system must have Bluetooth module [3]. The robot must be built before it can be managed while the Bluetooth module must be included in the project. Robotic parts for the model included structure (for example a car chassis), the controller, a Bluetooth module, electric motors, motor drivers, and other parts like batteries, power cables, wheels, etc. It is not enough to have an Android device and a robot. These two devices must be connected and programmed to share information. On the Android side, there must be an application with an interface to be utilized by the user to instruct the robot [5].

This project develops an Android application to operate a 4-wheel drive mobile robot, whose design and prototype are realized, both manually and automatically. MIT App Inventor's designer and block editor are used to create the application's programming and interface. According to research on this topic published in the literature, mobile robots can only be manually operated via specially created Android applications. There are options for both automatic and manual control in this application [12]. There are three components to the application interface. The application's main screen appears in the first section, followed by the manual control screen in the second and the automatic control screen in the third.

To close an application and transfer data for resetting the microcontroller card, it is feasible to establish a Bluetooth connection between the robot and a smartphone or tablet via the main screen. It is possible to pair the robot with a smartphone or tablet through the home screen to end an application and provide data for resetting the microcontroller card. [3]. Position reset,

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

targeted axis, and speed data are transmitted to the robot for automated control from the automated control screen. The created application was installed on the smartphone after it was compiled. Figure 1 depicts how the robot moves when given commands from a smartphone application.

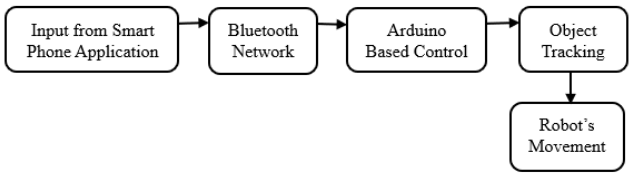

Figure 1: App Based User Interface

This is the screen that provides manual control over the mobile robot. There are buttons for direction, speed adjustment, indicators, a return button, and the fan and headlights on the displayed manual control panel. The robot can be moved in the desired direction with the help of these buttons. [1]. Figure 2 shows the front-end view of the button-based user interface.

Steps involved in Front-End Design Procedure are as discussed. Log in to MIT App Inventor. Click on "Start New Project" and give the project a name. Design the User Interface (UI): Move and drop UI components from the palette to create the interface. Use components such as four buttons for forward, backward, left, and right, and an additional button for stopping the robot. Adding Buttons: Drag five "button" components onto the screen. You can label these buttons with arrow symbols (↑, ↓, ←, →) or with text like "Forward," "Backward," "Left," "Right," and "Stop".

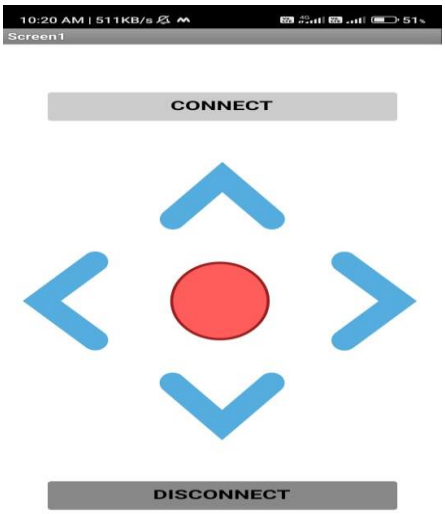

Figure 2: Button Based User Interface

Design Layout: Arrange the buttons in a layout that resembles arrow keys. You might use a grid arrangement or position them in a way that makes sense for your design. Adjust Button Properties: Customize the appearance of the buttons as needed. You can modify properties like text size, button size, and background color to produce the UI visually appealing. Figure 3 displays the backend schematic of the button-based user interface.

Figure 3: Backend schematic Diagram

Working on the back end: When the "Connect" button is clicked, the program scans for nearby Bluetooth devices and displays a list of them in the "Connect List Picker" component. The user chooses which Bluetooth device they want to connect to from the list. When the user decides a device, programme connects to it. Once the program is linked to the device, the user can send it single-byte numbers by clicking the "ForwordButton1" and "BackwardButton5" buttons. When the user clicks the "Disconnect" button, the programme disconnects from the device. Figure 4 shows the detailed block schematic of the back-end schematic.

Here is a more detailed explanation of each block in the program:

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

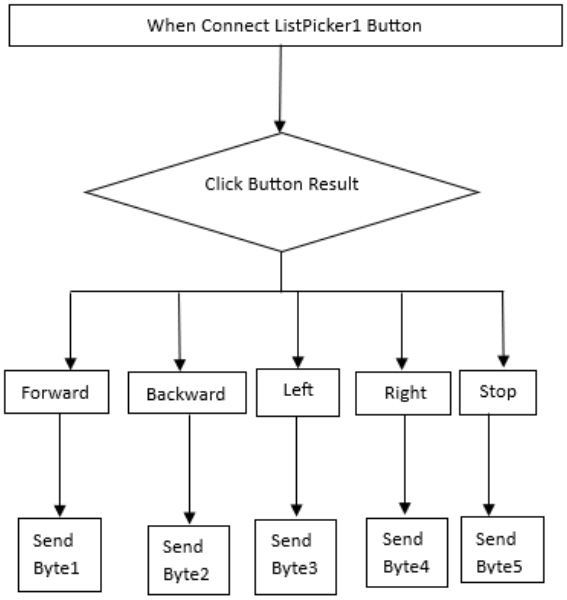

Figure 4: Block schematic of the backend

At first, we use ConnectListPicker1 before picking. This block is executed when the user opens the list of Bluetooth devices. It sets the "ConnectListPicker1" component's visible property to true, so that the list is presented to the user. Then Connect ListPicker1 After Picking is used. This block is executed when the user decides on a Bluetooth device from the list. It gets the address of the selected device and connects to it using the "BluetoothClient1.Connect" block. After the Bluetooth Client1 Connected the block is executed when the program successfully connects to the Bluetooth device. It sets the "StopButton3" component's visible property to true so that the user can disconnect from the device. Then we use Forword Button1 this block is executed when the user clicks the "ForwordButton1" button. It sends the single-byte number 30 to the Bluetooth device using the "BluetoothClient1.Send1ByteNumber"block. Backward Button5.Click: This block is executed when the user clicks the "BackwardButton5"button. It sends the single-byte number 5 to the Bluetooth device using the "BluetoothClient1.Send1ByteNumber"block Stop Button3. Click, this block is executed when the user clicks the "Disconnect" button. It disconnects from the Bluetooth device using the "BluetoothClient1.Disconnect" block. Once you have created the program, you can download it to your Android device and test it out.

2.2 Voice Controlled Robot

Voice is one of the most basic modes of social interaction and spoken language communication [7]. We use a very simple technique to speak commands to a robotic automobile. First, we turn all human queries into text using Google's speech-to-text converter. All of this is included in the Android app that we use. Next, the Bluetooth module of the robotic automobile receives the command in text format. This Bluetooth module acts as a bridge to transport data between the microcontroller of an automobile and an Android app. After receiving the text command, the microcontroller modifies the robotic car's movement [2]. Voice commands via a smartphone are used to operate the robot's motions, as seen in Figure 5.

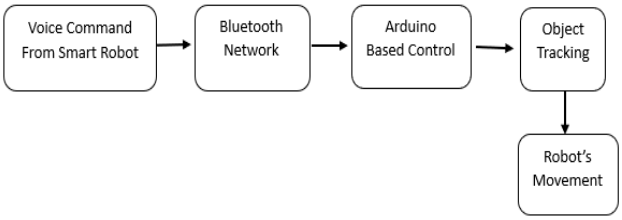

Figure 5: Voice control block diagram

The building specifics of the prototype for a "Voice-controlled Object Tracking Smart Robot" are briefly covered in this section [5]. The user can control the robot's movement by speaking commands into a smartphone running Android OS. These commands are processed by an internet cloud server application, which then transforms them into text format. The commands include things like forward, reverse, left, right, stop, and so on. The command data is wirelessly transferred from the smartphone's Bluetooth module to the robot's HC-05 Bluetooth module. A specific program that operates on the microcontroller contrasts the voice command with a list of pre-programmed keyword values. This in turn regulates the actuators that move the robot.

The voice-controlled robot's adaptability makes it useful in a number of contexts, such as home automation, helping people with mobility issues, and educational settings. Voice commands are easy to use and inclusive for users of different ages and technological backgrounds due to their intuitive nature [7]. a complete voice-activated robotic system that blends a flexible robotic platform with cutting-edge speech recognition technology. The amalgamation of these technologies exhibits potential for augmenting human-robot interaction, opening doors for inventive applications, and making a positive impact on the wider robotics domain.

The Android smartphone received the spoken command; these signals can be utilized in automated speech recognition (ASR) systems, which employ Bluetooth to connect the control unit and translate speech to text [7]. The robot should use a camera that is attached to it to locate the object when a voice instruction reaches the control unit. At each time interval, the robot will take a snap, which is then processed to complete a 360-degree rotation. The robot will determine the distance between the object and itself if it detects the thing. The instructions for

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

downloading the "Arduino Bluetooth Control" app from the Google Play Store are as follows. Verify that the HC-05 and your smartphone are paired, and that the password is "1234". Then, from the app search list, we selected the HC-05 option. Click the voice command now. To propel the car forward, articulate the word "forward". To make the car go backward, say "backward." To turn the car left, say "left." To allow the car to turn in the proper direction, articulate "right." To stop the car, say "stop." After using Bluetooth, disconnect the connection. The front-end view of the voice-based user interface is displayed in Figure 6.

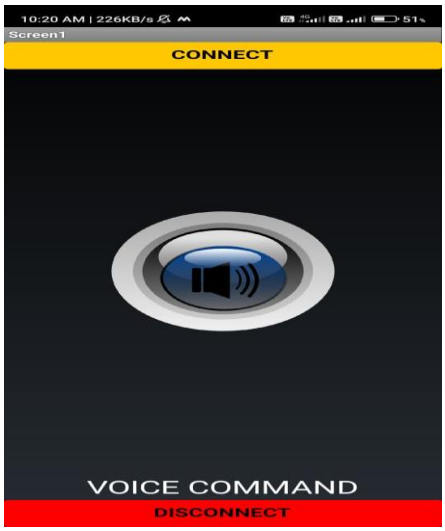

Figure 6: Voice Based User Interface

Voice instructions are used to control the robot's motions. These orders are supplied through an Android OS-based smart phone. An online cloud server is used by an application to translate the voice signal into text format in real time. The prototype robot's Bluetooth module receives this SMS command via the smartphone [5]. After receiving the command signal, the prototype robot's microcontroller gets the commands from the Bluetooth module.

Subsequently, the microprocessor produces a control signal that directs the DC motors to perform the intended movements. We're using a smart phone for wireless communication here. The prototype robot and Bluetooth receiver are interfaced with the microcontroller ports on the receiver side. Additionally, the smart phone's Bluetooth range determines how far the robot can operate. The distance between the mouth and the smartphone speaker, in addition to the internet connection, affects speech-to-text processing.

User Interface Elements

The designer view uses the designer view to drag and drop components onto the canvas. Components may include buttons, labels, sliders, and canvases. Buttons for Movement. Create buttons for different movements (forward, backward, left, and right). Assign appropriate icons or labels to the buttons. Slider for speed control. Implement a slider for controlling the robot's speed. Set the range based on the capabilities of your robot. Control logic helps to block View (Blocks Editor) [4]. Switch to the block view to define the logic behind the UI components. Button Click Events Create event handlers for button clicks to send corresponding commands to the robot. Use Bluetooth or Wi-Fi components to establish communication with the robot. Slider Value Changed helps implement logic to handle changes in the slider value. adjust the speed of the robot based on the slider position. The Companion App uses the MIT AI2 Companion app to test your app in real-time on an Android device. Ensure that the UI components respond correctly to user inputs. Hardware integration helps to connect your app to the actual robot hardware (e.g., via Bluetooth or Wi-Fi). Test the app with the physical robot to ensure proper communication. App descriptions provide a clear description of the app's purpose and functionality. Include instructions on how to connect the app to the robot.

Help screens use additional screens or pop-ups with instructions or troubleshooting tips. By following these steps and leveraging the visual programming capabilities of MIT App inventor, you can create a functional and user-friendly front end for controlling a wireless smart robot. Reminisce about thoroughly testing the app with the actual robot and iterating on the design based on user feedback and testing results**.** Figure 7 shows the process of connecting to the Bluetooth Client.

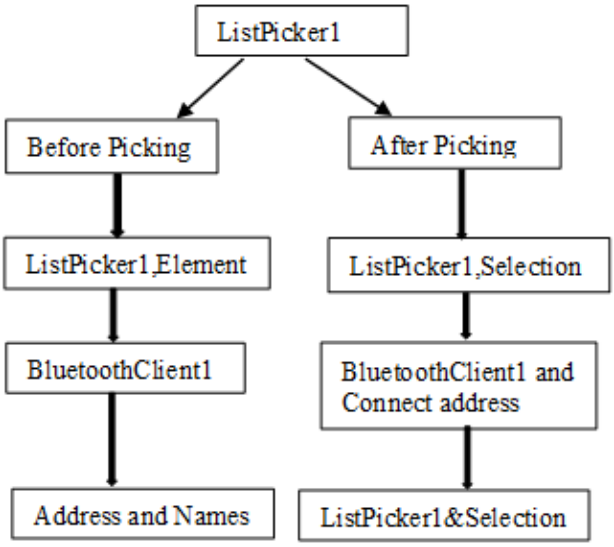

Figure 7: ConnectListPicker1

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

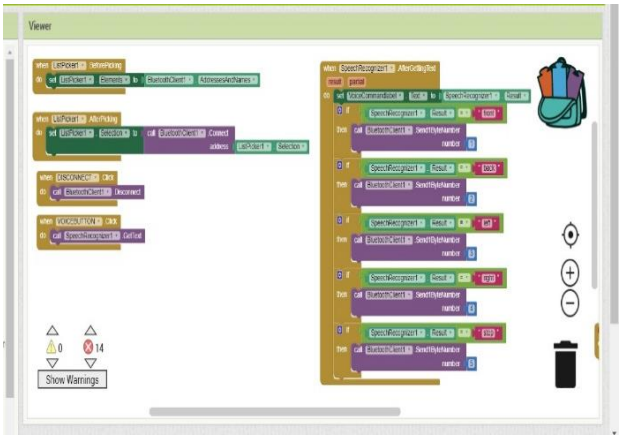

Figure 8: Backend of Voice based user Interface

Figure 8 shows the back end of voice-based user interface.

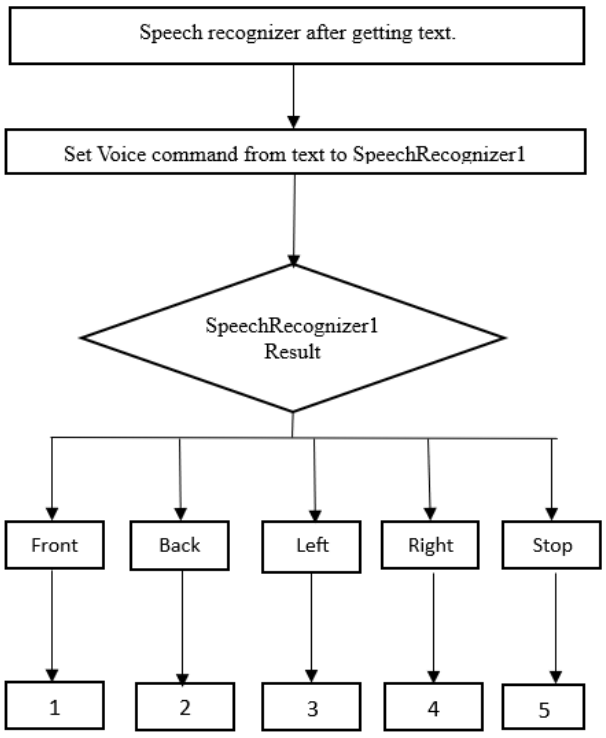

Figure 9: Block Diagram of Backend schematic of Voice based user Interface

Figure 9 shows the block diagram of back end working of voice-based user interface. MIT App Inventor is primarily designed for creating front-end applications; we can create simple backend-like functionality using cloud-based services. Creating a program interface is remarkably straightforward, involving the placement of selected objects within the designated area and visually determining their position and dimensions. However, for the program to not only appear aesthetically pleasing but also to function effectively, it needs to incorporate cryptic elements. To achieve this, users can click on the "Open the Block Editor" button, which prompts the download of a file. Upon opening the file for the first time, the block editor initiates and prompts the user to specify a path to the Command directory. Users can download the program installer compatible with their system from [5]. Upon installation, the program automatically copies the installation path, which can then be pasted into the Block Editor panel [9]. This approach outlines the creation of a backend-like structure for a wireless smart robot using MIT App Inventor.

1. Cloud-Based Backend: Use a cloud based service like Google Firebase as a simple backend. Set up a database to store information such as robot status, commands, and user authentication.

2. User Authentication: Integrate user authentication utilizing the authentication functionalities offered by the selected cloud service. This implementation guarantees that only authorized users have the ability to control the robot, enhancing security and preventing unauthorized access.

3. Database Structure: Design the database structure to store information about the robot, such as its status, battery level, and location. Include a table for storing commands to be executed by the robot.

4. MIT App Inventor Integration: Use the Web component in MIT App Inventor to make HTTP requests to the cloud-based backend. Design the app to send commands and receive real-time updates about the robot's status.

5. User Interface Elements: Create buttons and sliders in MIT App Inventor for user input. Use labels to display real-time information such as battery level and robot status.

6. Control Logic: Use MIT App Inventor's visual programming blocks to implement the logic for sending commands to the backend. Retrieve and display real-time information from the backend.

7. Security Measures: Ensure that the cloud-based backend enforces secure communication (e.g., HTTPS). Implement proper authentication and authorization mechanisms.

8. Testing: Test the MIT App Inventor app with the cloud-based backend. Ensure that commands are sent correctly and that real-time updates are reflected in the app.

9. Error Handling: Implement error-handling mechanisms in the MIT App Inventor app to manage issues like network failures or unsuccessful commands. Provide clear feedback to the user in case of errors.

2.3 Obstacle Avoidance

The robot could avoid obstacles by making use of an ultrasonic sensor. Obstacle avoidance is useful for

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

detecting obstacles and avoiding collisions. This has been added as a safety feature. The robot obtains the information from the surrounding area through a mounted ultrasonic sensor. According to widespread belief, an ultrasonic sensor is the best option for obstacle detection due to its affordability and long range. The ultrasonic sensor works in the 2cm - 400cm range. It has four pins, one for +5V power supply, one neutral/ground pin, one signal pin to trigger the transmitter and one echo pin to obtain the results [6]. In local navigation, the environment surrounding the robot is unknown, or only partially known, and sensors must be used to detect the obstacles, and a collision avoidance system must be incorporated inside the robot to avoid the obstacles. Similar to how dolphins navigate underwater, the HC-SR04 ultrasonic sensing element uses an echo sounder to determine how close an object is. Ultrasonic waves move at a speed of 343 m/s. It takes a lot of speed for MCUs in microcontrollers to measure precisely. In fifteen seconds, the waves reflect off the surface four meters distant. Humans are not harmed by ultrasonic waves. The majority of applications for ultrasonic sensors involve measuring distance [13]. It offers amazing high-accuracy, consistent readings, and non-contact detection in an intuitive package ranging from two to four hundred centimeters. It can operate without being hindered by dark or sunshine materials. Its transmitter and receiver modules are attached.

Working Principle

The flag with a short and high recurrence is emitted by the ultrasonic sensor. These propagate at the speed of sound within the atmosphere. A multivibrator is also included with the ultrasonic sensor; it is fixed to the base. Ultrasonic sensors are used in many different applications, such as automatic door openers and instruction alarm systems. The ultrasonic sensor has a high-performance threshold and is small in size. [13].

A vibrator and a resonator are combined to create a multivibrator. The vibration creates an undetectable wave that is sent by the resonator. The anode, which generates a 40 kHz wave, and the indicator, which detects the 40 kHz wave and relays an electrical flag to the microcontroller, are the two actual components of the ultrasonic sensor. By keeping a safe distance from obstructions, the robot can detect and identify a protest thanks to the ultrasonic sensor. The robot will move in accordance with the given code until it encounters an empty area, avoiding any roadblocks or impediments along the way. Obstacle avoidance of this type is very helpful in businesses that require automated supervision, such as in locations where human safety may be compromised. Depending on the requirements, additional sensors such as light, line, ultrasonic, and ultrasound sensors can be added to create this robot. Figure 10 illustrates how the intelligent robot navigates barriers by utilizing its ultrasonic sensor's capabilities.

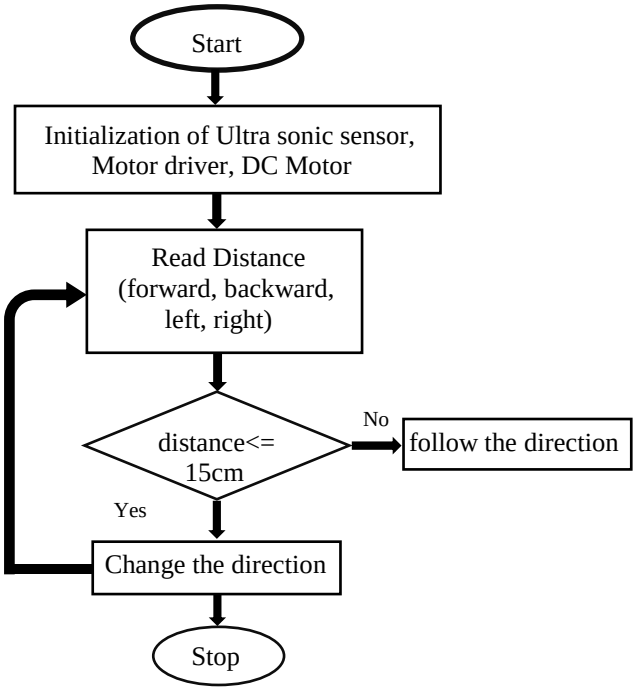

Figure 10: Flow diagram of operation of Smart Robot

III. CONCLUSION

In this study, an Android application was developed with MIT App Inventor for Bluetooth -based control of a 4-wheel-drive mobile robot. While the application interface was created in the MIT App Inventor designer, it was made with the drag-and-drop method in the block editor. When the application installed on the phone is run, the Bluetooth connection is provided on the main screen that opens, and the transition to other screens can be selected. An Arduino microcontroller was used to test that the values to be sent from the application are sent correctly when the buttons are pressed. A connection was established between the HC-05 Bluetooth module attached to the Arduino and the smartphone. In the program written to receive data on Arduino, the values sent with the application were transferred to the serial screen of the Arduino IDE program and examined, and it was shown that the values were transferred as planned. In line with the information sent to the Arduino, manual and automatic control of the four-wheeled mobile robot was carried out. The automatic speed and position control of the mobile robot was controlled with a PID in line with the reference values sent from the application to the robot microcontroller. In the given graphics, it is presented that the mobile robot is controlled effectively manually and

Peer Reviewed & Refereed Journal
Volume-14, Issue-2 (April 2024)
https://doi.org/10.5281/zenodo.11057143

successfully captures the reference values with automatic control. The designed application can be installed on different Android-based phones or tablets, it can connect with any processor used in Bluetooth protocol; and can be used on different robots according to the program that the user will write on the microcontroller Through the use of wireless Bluetooth technology, our study has successfully proven the design and execution of an Android mobile application that allows for manual control of a mobile robot. Our concept stands out because to its focus on voice control capability, which gives consumers an engaging and simple way to direct the robot's motions.

REFERENCES

[1] A. Top & M. Gökbulut. (2022). Android application design with mit app inventor for bluetooth based mobile robot control. *Wireless Personal Communications, 126*, 1403–1429.

[2] Chaudhry, A., Batra, M., Gupta, P., Lamba, S. & Gupta, S. (2019). Arduino based voice controlled robot. *International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 415-417. IEEE.

[3] M Saravanan & B Selvababu**.** (2015). Voice-controlled object tracking smart robot. *IEEE Conference Publication, IEEE Xplore*, pp. 40-45.

[4] Jan Nádvorník & Pavel Smutný. (2019). Android-based wireless controller for military robot using bluetooth technology. *IEEE Conference Publication, IEEE Xplore*, pp. 49-55.

[5] Makula, P., A. Mishra, A. Kumar, K. Karan & V. K. Mittal. (2015). Voice-controlled object tracking smart robot. In: *Proceedings of IEEE 3rd International Conference on Electronics, Computing and Communication Technologies (CONNECT 2015). IIIT Bengaluru, India*.

[6] Budiharto, Widodo, et al. (2019). Android-based wireless controller for military robot using bluetooth technology. *2nd World Symposium on Communication Engineering (WSCE)*. IEEE.

[7] Ankit, Vaghela, Patel Jigar & Vaghela Savan. (2016). Obstacle avoidance robotic vehicle using ultrasonic sensor, android and bluetooth for obstacle detection. *Int. Res. J. Eng. Technol, 3*, 339-348.

[8] Lee, W., Seong, J. J., Özlü, B., Shim, B. S., Marakhimov, A. & Lee, S. (2021, February 17). Biosignal sensors and deep learning-based speech recognition: A review. *Sensors, 03*, 339-348.

[9] Akilan, T., Chaudhary, S., Kumari, P. & Pandey, U. (2020). Surveillance robot in hazardous place using IoT technology. In: *2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 775-780. IEEE.

[10] Ansari, Aiman, Yakub Ansari, Saquib Gadkari & Aarti Gokul. (2015). Android app based robot. *International Journal of Computer Science and Information Technologies, 6*, 1598-1600.

[11] Singh, Mandeep, Gurmohan Singh, Jaspal Singh & Yadwinder Kumar. (2021). Design and validation of wearable smartphone based wireless cardiac activity monitoring sensor. *Wireless Personal Communications, 119*(1), 441-457.

[12] Hong, SeongYong & YongHyun Hwang. (2020). Design and implementation for IOT based remote control robot using block-based programming. *Issues in Information Systems, 21*(4), 317-330.

[13] Vairavan, R., S. Ajith Kumar, L. Shabin Ashiff & C. Godwin Jose. (2018). Obstacle avoidance robotic vehicle using ultrasonic sensor, Arduino controller. *International Research Journal of Engineering and Technology (IRJET), 5*(02).

[14] Shirakol, Shrikanth & S. S. Kerur. (2023). An improved vlsi architectural design of discrete cosine transform based on the loeffler-dct algorithm. *International Journal of Intelligent Engineering & Systems, 16*(5), 173-184.

[15] A. Angadi, S. Umarani, T. Sajjanar, R. Karnam, K. Marali & S. Shirakol. (2023). Architectural design of built in self-test for vlsi circuits using LFSR. *International Conference on Applied Intelligence and Sustainable Computing (ICAISC), Dharwad, India*, pp. 1-7. DOI: 10.1109/ICAISC58445.2023.10200604.