DESIGN AND IMPLEMENTATION OF AN AUDIO LINKED REMOTE-CONTROLLED CAR

2423748

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: This report discusses the design and implementation of a remote-controlled car which utilises an audio link between a controller and the car. The implemented solution utilises an app to emit tones at specific frequencies that are detected by a microphone on the car. These frequencies dictate the movement of the car as there are commands associated with. A second order bandpass filter is introduced to ensure that external noise is minimised and so that only tones associated with commands are processed. A Fast Fourier Transform is then performed on the filtered signal to extract the intended frequencies. The motors are controlled via a H-bridge motor driver. The final solution performed satisfactorily once tested as it was able to complete a lap around a racetrack without hinderance from the external noise present in the environment.

Key words: Audio Linked, Bandpass Filter, Fast Fourier Transform, Microcontroller, Remote Controlled Car

1. INTRODUCTION

The aim of this report is to discuss the design and implementation of a Remote-Controlled (RC) that utilises an audio signal to control the movement of the car. The functional car is required to operate in a non-ideal environment it is required to compete with other RC cars in a race. A further analysis of the project is conducted in the sections that follow.

Section 2 provides a background on the project requirements, constraints, and previously implemented solutions. Section 3 discusses the design and implementation of the RC car. Section 4 analyses the system after test were conducted. Lastly, Section 5 provides an overall evaluation of the system.

2. BACKGROUND

2.1. Project Requirements

The key component of this project is that the vehicle should be designed such that an audio-based link exists between the controller and the RC car. Additionally, the car should be able to operate optimally in an environment with external noise as the car would be simultaneously competing with other RC cars on the same racetrack. However, the latter requirement was eased prior to testing and each car was required to individually perform a lap around the racetrack.

2.2. Constraints

- The audio signal that the car is meant to receive is required to be within the human audible range of 20Hz to 20kHz.
- The voltage supplied to the motors should not exceed 5V.
- A standardised chassis and DC motors are used without enhancing any aspects of these items.

2.3. Success Criteria

For successful implementation, the final design should meet all requirements set out for the system and be able to complete a lap around the test track with the driver having complete control over the car.

2.4. Previous Solutions

Numerous design variations of the aspects used in this project have all been utilised in previous designs implemented by others. Inspiration which was drawn from Panduino's project which utilised the Arduino Fast Fourier Transform (FFT) library, "arduinoFFT.h", for a "FFT Spectrum Analyzer [1].

3. DESIGN AND IMPLEMENTATION

3.1. Preliminary Design

Before reaching the implemented solution, a preliminary solution was tested which utilised an ElecHouse voice recognition module, connected to an Arduino Uno, to receive the commands that control the car. Whilst this solution was functional, it provided no noise immunity and could only operate in an ideal environment. In an attempt to improve the noise immunity of the module, a second order bandpass filter was attached to the module and tones of specific frequencies were used instead of voice commands. However, this was not an operable solution and thus an alternate system design was implemented.

3.2. Improved Implementation

The alternate solution consisted of both analog and digital subsystems, which together were used to improve the noise immunity capabilities of the system. This was done by using the second order bandpass filter implemented in the preliminary stage, and thereafter a Fast Fourier Transform (FFT) was performed on the analog signal transmitted to the microcontroller.

The block diagram in Fig 1. of Appendix A. illustrates the breakdown of the system and how each of its components interact with one another.

3.3. *Microcontroller*

An ESP32 was opted for over an Arduino Uno as the ESP32 incorporates a 32-bit microcontroller unit (MCU) with a dual-core capable of operating at a clock speed of up to 240 MHz, which is significantly faster than the 8-bit MCU Arduino with a clock speed of 16Mhz [2].

3.4. *Microphone*

The microphone located on the car and is used to detect and receive the audio signals from the controller unit. For the ESP32 to process the signal that the microphone receives, the signal needs to be amplified by a gain. Therefore, a rather simplistic approach was taken for this subsystem as the MAX9814 microphone amplifier module was acquired. The module's microphone has an audio detection range of 20Hz to 20Khz and the built-in amplifier has an adjustable gain which was set to the default maximum gain of 60dB [3].

A microphone and amplifier circuit could have been designed and built for a fraction of the price of the module, however due to time constraints this was not feasible, and the additional costs were incurred to procure the module.

3.5. *Band Pass Filter*

To ensure that only frequencies associated with certain commands are processed, a second order bandpass filter designed by Bhatt [4] was used to reduce the range of frequencies that can be transmitted to the ESP32. A second order filter was sufficient for the system as there was a reduced need to cut out a wide range of frequencies due to the constraints surrounding the race being eased.

Certain adjustments were made to fit the specifications of the system and the finalised filter in Fig. 2 of Appendix A was implemented. Since a sampling frequency of 7000Hz was used (See Section 3.6) , the upper threshold of the filter had to be less than, or equal to half of the sampling frequency in order to avoid aliasing. Therefore, the bandpass filter was adjusted to have a center frequency of 2800Hz and a passband bandwidth of 300Hz.

To power the filter, both a positive and negative supply were required for the LM741 op-amp that was used. The minimum recommended supply voltage for the LM741 is ±10V, however after testing different supply voltages it was found that the LM741 performed optimally with ±9V [5] . Therefore, a dual-rail supply using 9V batteries were used to power the op-amp.

3.6. *Fast Fourier Transform*

Once the amplified and filtered analog signals are transmitted to the ESP32 microcontroller, a FFT is performed using the "arduinoFFT.h" library. This extracts the frequency of the audio signals with the greatest magnitude.

For the FFT to produce accurate results, numerous samples are gathered at a specific sampling frequency. A maximum of 256 samples were used at a sampling frequency of 7000 Hz. These specifications were based off the processing capabilities of the ESP32, as increasing the sampling frequency and number of samples further resulted in the FFT producing unpredictable and erroneous results.

To ensure that commands are only carried out when required and not due to stray frequencies that are within the passband range, a counter system was introduced. A sequence of twenty consecutive frequencies that are associated with the range of a command are required before the command can be carried out by the car.

3.7. *Motor Control*

Once the extracted frequency from the FFT falls within the range of specific frequencies defined in Table 1. Of Appendix B, a Pulse-Width Modulation (PWM) signal associated with the command is sent from the ESP32 to the DC motors via the H-Bridge motor driver. The H-bridge allowed for straightforward control over the motors, allowing it to move forward, left, right and to come to a stop, all by setting the duty cycle of the PWM according to the Table 2. in Appendix B. Initially, a command for the car to move backwards was accounted for but was later removed as it was not required.

Due to the constraints, the motor driver was supplied by a 5V power bank. However, only 3.4V are utilised by the motors as the motor driver has an inherent voltage drop off, measured at 1.6V, to power its internal circuitry [6].

3.8. *Controller App*

A simple android app was coded in Android Studio for the RC car control unit, as shown in Fig 3. of Appendix A. The user interface consists of six buttons, five of which when pressed, play continuous tones of frequencies that are associated with specific motor commands. The sixth button is used to stop any tone that is playing from the app.

The app is designed such that if a button is pressed while a tone is already being emitted, the tone that is playing will immediately stop and the sound associated with the most recent button press will start playing. Additionally, pressing two buttons simultaneously does not have any adverse effects as only the tone associated with the button that is released at a later stage is played.

4. TESTING AND ANALYSIS

4.1. Strengths

Despite needing twenty consecutive frequency values for a command to be carried out, the response time of the car's movements were still swift. Thus, successfully preventing any external influence on the car without having to forgo on the quality of the movement. The reduced voltage that was supplied to the motors proved to be advantageous as having finer control over the movement of the car due to a slower speed (due to a low voltage) yielded better results than having the car move at a faster speed but with less control (due to an increased voltage supply).

4.2. Weaknesses

When testing the FFT, it was detected that the extracted frequency deviates from the actual frequency by an offset error of approximately 30Hz. This error was accounted for in the motor's code as the range of the frequencies were adjusted. Additionally, a slight flaw was observed in the movement of the car when both motors operated at a 100% duty cycle as the car moved slightly to the left rather than going fully forwards. To minimise this effect as much as possible, the duty cycle of the right motor was reduced to from 100% to 95%.

5. EVALUATION

5.1. Time Management

Overall, the time spent on the project was well managed considering as a functional solution was timeously implemented despite setbacks from the preliminary solution. To achieve this, tasks were divided amongst group members and were strictly adhered according to the breakdown provided in Table 3. of Appendix B. Additionally the Gantt chart in Table 5. of Appendix B provides a breakdown of the time spent on each task for this project

5.2. Cost of Implementation

Due to there being no cost constraints, a total of R657 was incurred to implement the design. A breakdown of the costing is shown in Table 4. of Appendix B Whilst there are cheaper and more feasible solutions that exist, the implemented solution is a reasonable one as pre-existing modules were used to meet the project deadlines.

5.3. Overall Evaluation

Despite the slight flaws in certain subsystems, the overall solution can be deemed successful as the system functioned as required when all the subsystems were linked. The responsiveness exhibited by the car was excellent as the car was able to detect commands from a maximum distance of 15m without an observable delay. The implemented filter was sufficient in reducing the noise from the surrounding as the car was able to complete the testing race lap in just under 90 seconds without being hindered.

5.4. Future Recommendations

For greater control over the car, each command can be associated with a smaller range of frequencies, thus allowing for more commands within the same passband. This would require an improved interface for the app as the current interface is only feasible for the present car design. Additionally, if the control over the car can be further improved on, as previously mentioned, and there are no power constraints, it would be feasible to supply the motors with a higher voltage which would allow for increased speed.

6. CONCLUSION

In summation, a remote-controlled car was implemented to compete in a race with other cars. The main requirement of the car was that the link between the controller and the car should be audio based. Therefore, a solution that utilised an app to emit tones at specific frequencies that were detected by a microphone on the car. These frequencies dictate the movement of the car. A bandpass filter was used to ensure that external noise was minimised and only tones associated with commands were processed. A FFT was performed on the filtered signal to extract the intended frequencies that control the motors of the car. The overall implementation was successful as the car was able to uphold all the constraints and requirements.

REFERENCES

[1] Pandauino, "*1024 Samples FFT Spectrum Analyzer Using an Atmega1284"*., https://www.instructables.com/1024-Samples-FFT-Spectrum-Using-an-Atmega1284/. Last Accessed: 24 May 2023.

[2] Benne, "ESP32 vs Arduino Speed Comparison," *Makerguides.com*, https://www.makerguides.com/esp32-vs-arduino-speed-comparison/ Last Accessed: 23 May, 2023.

[3] "Adafruit AGC Electret Microphone Amplifier - MAX9814." https://cdn-learn.adafruit.com/downloads/pdf/adafruit-agc-electret-microphone-amplifier-max9814.pdf, Last accessed: 23 May, 2023.

[4] Bhatt A., Designing a band pass filter. Designing a band pass filter. https://www.engineersgarage.com/designing-of-band-pass-filter-2/. Last accessed 24 May 2023.

[5] Texas Instruments, LM741 Operational Amplifier, 2015.

[6] STMicroelectronics, L298 Dual Full-Bridge Driver, n.d

APPENDIX A

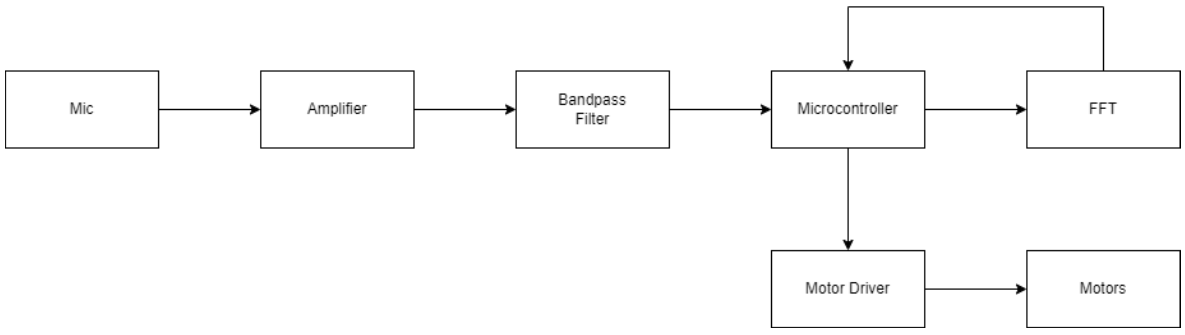

Figure 1: Block Diagram of the implemented system.

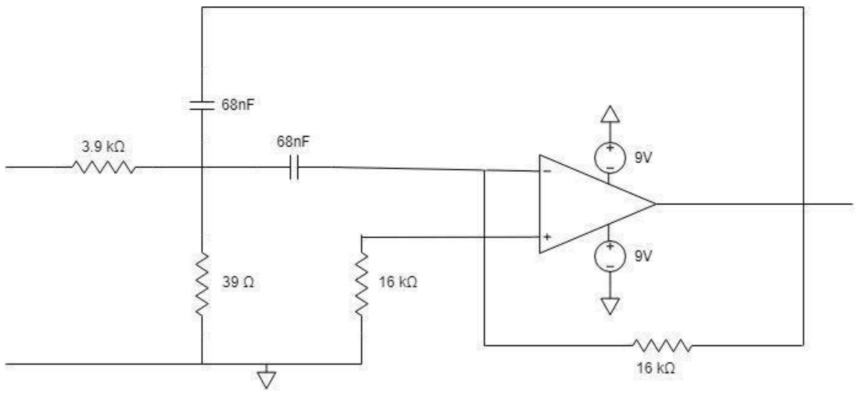

Figure 2: Circuit diagram of the second order bandpass filter used in the circuit.

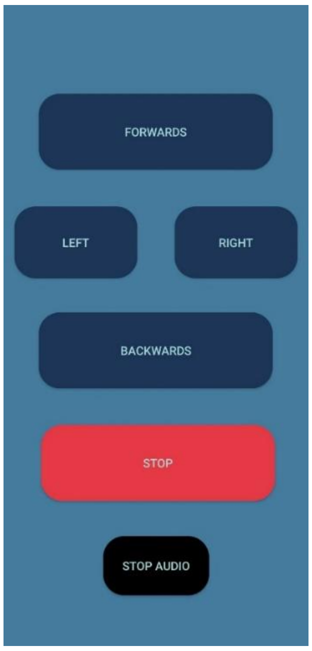

Figure 3: User interface of the control app

APPENDIX B

Table 1: Frequencies associated with each command on the car.

Car Command	Range of Frequencies to activate command (Hz)	Frequency Produced By The App (Hz)
Stop	2700 to 2765	2710
Left	2770 to 2815	2775
Right	2820 to 2870	2830
Forward	2875 to 2925	2885

Table 2: Duty cycle associated with each command on the car.

Car Command	Duty Cycle of Left Wheel (%)	Duty Cycle of Right Wheel (%)
Stop	0	0
Left	60	80
Right	100	60
Forward	100	95

Table 3: Task breakdown of each team member

Task	Participant
Research	Engineer 1, Engineer 2 & Engineer 3
Filter Design	Engineer 1 & Engineer 2
FFT	Engineer 1 & Engineer 3
Motor Control	Engineer 1 & Engineer 3
App Development	Engineer 3
Construction and Configuration	Engineer 1, Engineer 2 & Engineer 3
Testing	Engineer 1, Engineer 2 & Engineer 3

Table 5: Costing of components of the system

Component	Quantity	Price
ESP32	1	R179
MAX9184 Microphone	1	R98
L298N H-Bridge Motor Driver	1	R52
Power banks	2	R300
9 V Batteries	2	R18
Misc. components		<R10
Total		R657

Table 4: Gantt Chart of the time spent on each component of the project.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13
Task													
Preliminary Report Writing	🟥	🟥	🟥	🟥									
Peer Reviews							🟦						
Presentation						▨	▨						
Design and Implementation					🟩	🟩	🟩	🟩	🟩	🟩			
Testing of Solution											🟨		
Final Report Writing												🟧	🟧