UNIVERSIDADE DE COIMBRA

Mechatronics Project

Robotic Drummer

Date: 22/05/2024

by

Stefano Trenti, 2023252452
José Dinis Antunes Ferreira, 2019244998

Under the supervision of

Prof. Mahmoud Tavakoli

Department of Computer Engineering
University of Coimbra

Contents

1 Introduction **1**

2 Materials and Electrical Design **1**
2.1 Materials 1
2.2 Electrical Design 1

3 Motion control/coding **2**

4 GUI implementation and Extra part **5**

5 Reliability **6**

6 Conclusion **6**

1 Introduction

The goal of this project is to design and implement a robotic drummer that can replicate the movements of a human drummer. The robot will be built using an Arduino UNO and servo motors. The project will involve understanding the mechanics of drumming, translating human motion into robot movement, and programming the robot to play different rhythms at various speeds and volumes. Additionally, a graphical user interface (GUI) will be developed to control the robot's drumming parameters. The project will also focus on evaluating the system's reliability and exploring the possibility of adding more servos to achieve different types of drum beats.

2 Materials and Electrical Design

2.1 Materials

The robotic drummer is designed using an Arduino UNO platform as the microcontroller. The Arduino UNO is a popular open-source electronics platform based on easy-to-use hardware and software. It is responsible for the logic of the system and controlling the servo motors that actuate the robotic arm. The arm consists of three servo motors:

- **Elbow Servo (S0):** This servo controls the up-and-down movement of the forearm, simulating the elbow joint of a human drummer.
- **Wrist Servo (S1):** This servo controls the back-and-forth movement of the hand, simulating the wrist joint.
- **Finger Servo (S2):** This servo is responsible for the precise movement of the drumstick, allowing for different types of beats.

The servos are powered and controlled directly by the Arduino UNO, utilizing its pulse width modulation (PWM) capabilities. While the Arduino UNO has a limited current output, it is sufficient for driving these small servos under the expected load conditions of the robotic drummer.

2.2 Electrical Design

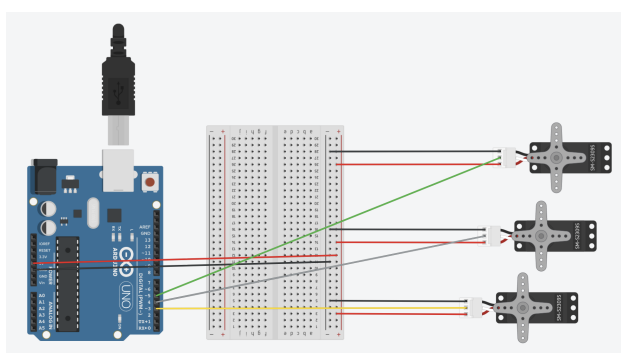

Figure 1: Electrical circuit of the system

The electrical design of the robotic drummer is relatively simple, utilizing an Arduino UNO as the central controller and an external power supply for the servo motors. The Arduino UNO is capable of digital and analog input/output operations. In this project, the Arduino is responsible for generating the pulse width modulation (PWM) signals that control the position of the WS-MG996R servo motors.

The robotic drummer uses three WS-MG996R servo motors to control the movement of the arm and drumstick. These servos are standard hobby servos commonly used in robotics and hobbyist projects due to their affordability, ease of use, and relatively high torque output.

The servos are powered directly from an external power source, which is plugged into a wall outlet. This external power source is necessary to provide sufficient current to drive the servo motors, especially during high-speed or high-torque movements. The Arduino UNO, while capable of powering the servos under normal conditions, may not be able to provide enough current for demanding operations, leading to performance issues or even damage to the board.

The external power source is a standard AC-to-DC adapter that converts the mains voltage to a suitable DC voltage for the servos. This DC voltage is then connected to the servo motors and the Arduino UNO, providing a stable and reliable power supply for the entire system. The Arduino UNO controls the servos by sending PWM signals to their control wires, which are connected to the Arduino's digital pins. The Arduino code defines the desired angles for each servo based on the desired drumming motion and sends the corresponding PWM signals to the servos to achieve those angles.

3 Motion control/coding

The motion control of the robotic drummer is implemented through the Arduino code, which defines functions to compute the angles of the elbow and wrist servos using cubic and quintic polynomial equations.

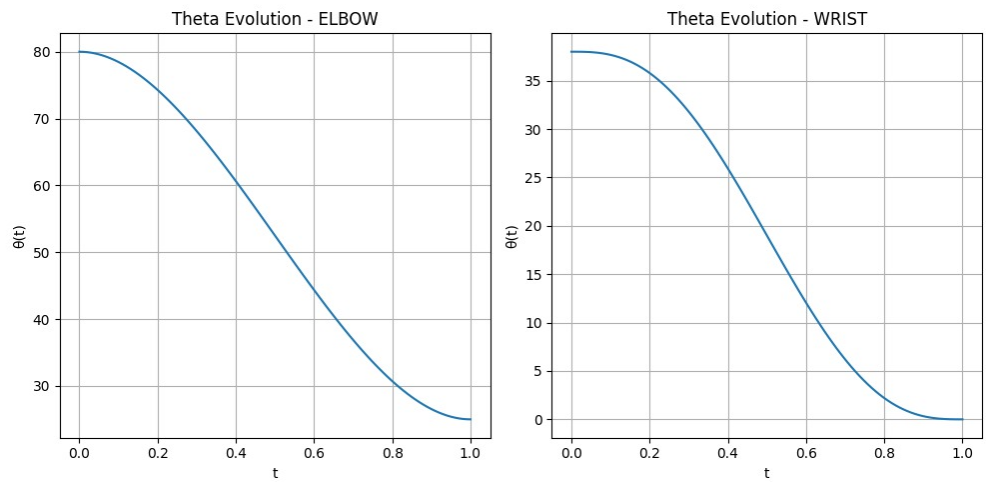

Figure 2: Theta evolution

The graphs represent the angular displacement of the elbow and wrist servos, respectively, over time (t). The x-axis (t) represents the normalized time, ranging from 0 (start of the movement) to 1 (end of the movement). The y-axis ($\theta(t)$) represents the angle of the servo in degrees.

- **Elbow Servo (S0)** The graph for the elbow servo shows a smooth, decreasing curve. This indicates that the elbow starts at a high angle (approximately 80 degrees) and gradually lowers to a final angle (approximately 25 degrees) over the course of the movement. This motion simulates the downward swing of a human drummer's elbow during a drum stroke.

- **Wrist Servo (S1)** The graph for the wrist servo also shows a decreasing curve, but with a steeper slope compared to the elbow servo. This indicates that the wrist starts at a high angle (approximately 35 degrees) and moves down to a lower angle (approximately 0 degrees) more quickly than the elbow. This motion simulates the wrist's rotation as it follows the elbow during the drumming action. The faster motion of the wrist is needed to give more energy to the drumstick imitating a "whipping" motion.

These equations ensure smooth and controlled movements of the robotic arm during drumming actions. The following flow chart follows the main steps that the main loop of the program follows.

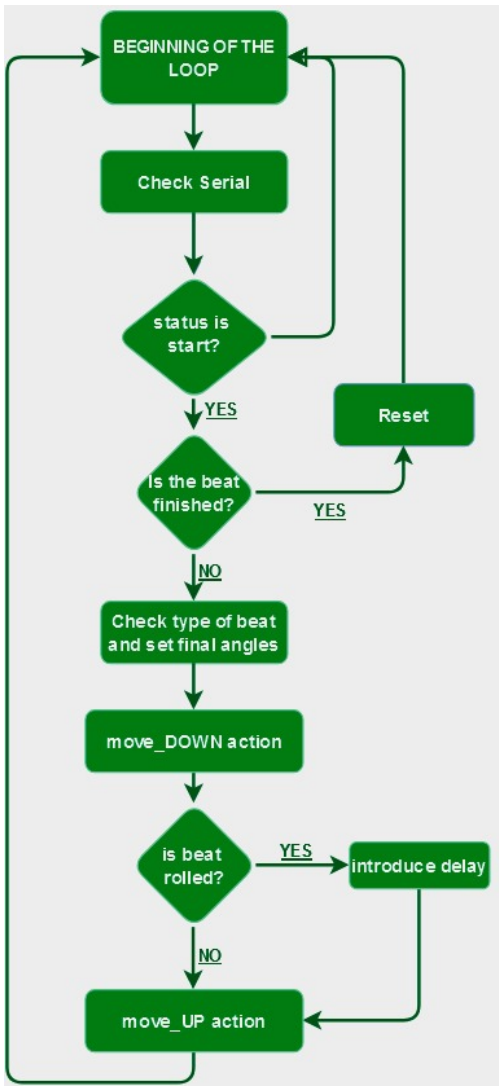

Figure 3: Loop

- **Beginning of the Loop**: The code enters a continuous loop, repeatedly executing the following steps.
- **Check Serial**: The code checks if any new variable values have been received through the serial port. These commands are sent from the GUI and contain information about the desired beat type, speed, volume and system state.
- **Status is Start?** : If the received command indicates a "start" status, the code proceeds to the next step. If not, it skips to the end of the loop and begins checking for new commands again.
- **Reset**: If the status is "start" and the beat is finished, the code resets any relevant variables or counters to prepare for a new drumming sequence.
- **Is the Beat Finished?**: The code checks if the current beat has been completed. If the beat is still in progress, it proceeds to the next step. If not, it goes back to step 2 to check for new commands.
- **Check Type of Beat and Set Final Angles**: The code determines the type of beat (open, muted, or rolled) based on the received command and sets the appropriate final angles for the finger servo motors.
- **Move DOWN Action**: The servo motors are then commanded to move to the calculated final angles following the computed trajectory, simulating the downward stroke of the drumstick hitting the drum.
- **Is Beat Rolled?**: The code checks if the current beat is a rolled beat. If it is, it proceeds to the delay step. If not, it skips to the upward motion.
- **Introduce Delay**: For a rolled beat, a delay is introduced to let the drumstick bounce simulating the continuous drumming action of a roll.
- **Move UP Action**: Finally the servo motors are commanded to return to their initial "up" position.
- **End of the Loop**: The code returns to the beginning of the loop, ready to check for new commands and repeat the drumming sequence.

The image on this page shows the flowchart of the MOVE_DOWN function which is responsible for the downward stroke of the robot. This function computes the angular trajectory for the various servos needed to accomplish the desired motion.

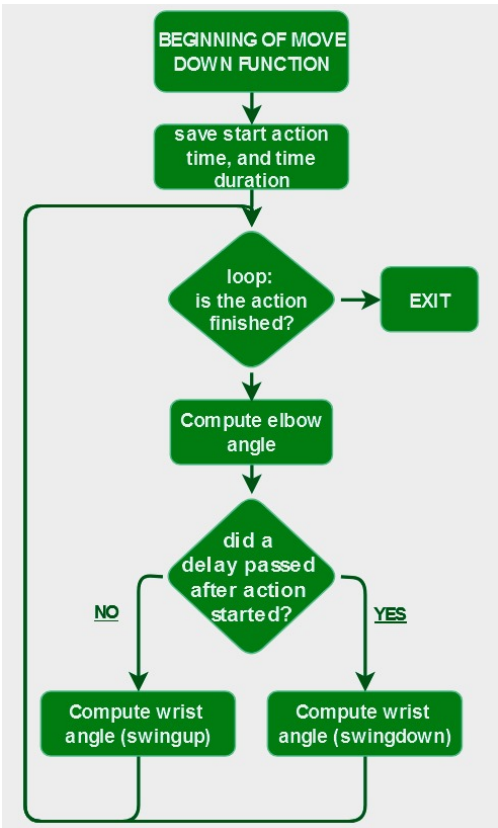

Figure 4: move_DOWN function

- **Beginning of move_DOWN function**: This indicates the start of the function execution.
- **Save start action time, and time duration**: The function records the current time and the intended duration of the downward movement. This is used to calculate the servo angles at each step of the movement.
- **Loop**: The function enters a loop that continues until the movement is complete.
- **Is the action finished?**: This is a conditional check to determine whether the desired movement duration has elapsed. If so, the function exits the loop and returns.
- **Compute elbow angle**: If the action is not finished, the function calculates the elbow servo angle. This angle determines the position of the forearm during the downward movement.
- **Did a delay pass after action started?**: This checks if a specific delay has passed since the start of the movement. The purpose of this delay is to allow the elbow servo to reach its desired position before the wrist servo starts moving.
- **Compute wrist angle (swingup/swingdown)**:
  - **If the delay has not passed (NO)**: The wrist angle is computed to move the wrist upwards, preparing for the downward swing.
  - **If the delay has passed (YES)**: The wrist angle is computed to move the wrist downwards, along with the drumstick, striking the drum.
- **Exit**: After the action is finished, the function exits, indicating the completion of the downward movement.

The code implementation for this control algorithm can be found here. Also a video demonstration of the robot movements along with other material can be found here for the video and here for all the material.

4 GUI implementation and Extra part

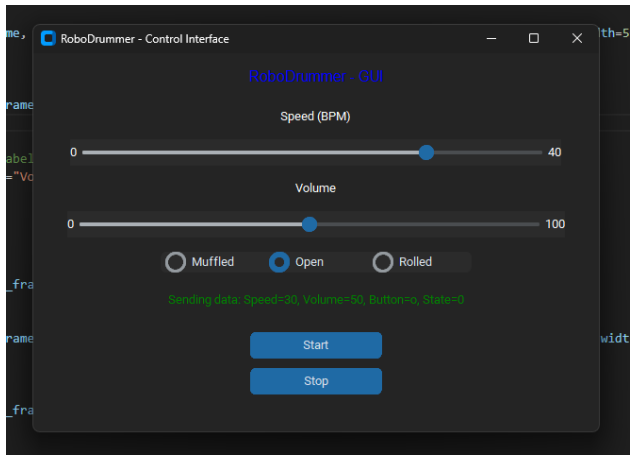

Figure 5: GUI implementation

A simple GUI was required to control the various parameters of the robot such as BPM, Volume and type of beat. This was implemented in Python using the Tkinter and CustomTkinter libraries. This program continuously communicates through Serial the following message: *message = f"Sending data: Speed=int(speed), Volume=int(volume), Button=button_state, State=state"* The range of values is: Speed = 0-40 bpm, Volume = 0-100%, Button = "m, o, r", State = 1 or 0 for start and stop. The code implementation for the GUI can be found here.

The robotic drummer is capable of producing three distinct drum beats: muffled, open, and rolled. These variations are achieved by manipulating the third servo motor (S2), which controls the drumstick's angle and movement.

- **Muffled Beat:** In this mode, the drumstick strikes the drum with a reduced force and a shorter rebound, resulting in a quieter and less resonant sound. This is accomplished by setting a specific hit angle for the drumstick servo (S2) that limits the impact and rebound of the drumstick.
- **Open Beat:** This is the standard drum beat, where the drumstick hits the drum with normal force and a full rebound, producing a clear and resonant sound. The hit angle for the drumstick servo (S2) is set to allow for a full swing and rebound.
- **Rolled Beat:** This beat involves the drumstick rapidly striking the drum multiple times in succession, creating a continuous and sustained sound. To achieve this, the code introduces a delay after the initial downward stroke of the drumstick, keeping the drumstick in contact with the drum for a short period before returning it to the "up" position. This process is repeated rapidly to create the rolling effect.

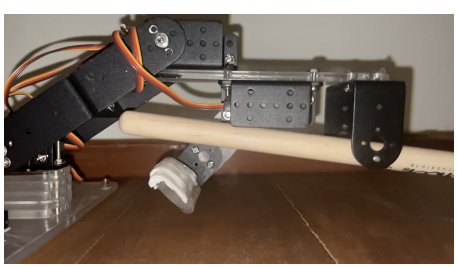

Figure 6: Open or rolled beat fingers

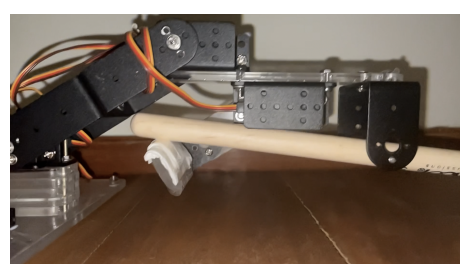

Figure 7: muffled

The selection of the beat type is controlled through the GUI, which sends commands to the Arduino. The Arduino code then interprets these commands and adjusts the hit angle of the drumstick servo (S2) accordingly to produce the desired beat.

5 Reliability

The relialibity of this system was proven by looking at the audio track from the video recordings of this system in motion. By measuring the time interval between the sound peaks for each beat the accuracy of the system was able to be confirmed (within 30 ms). Also by setting the peak volume value at 50% as a constant line we are able to see how when we raise the volume from the GUI this is reflected by the peaks exceeding this constant line. The same is true for lowering the volume and the different beat types.

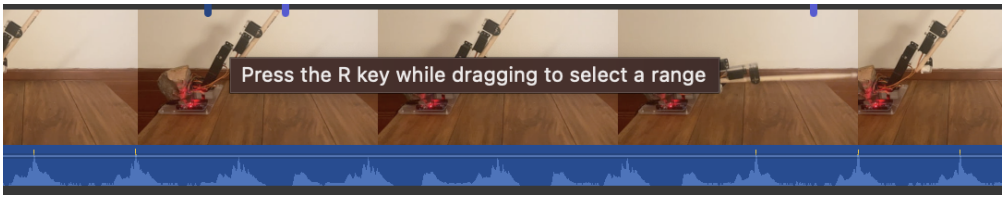

Figure 8: Different beats sound-wave with 75% volume and 30 bpm

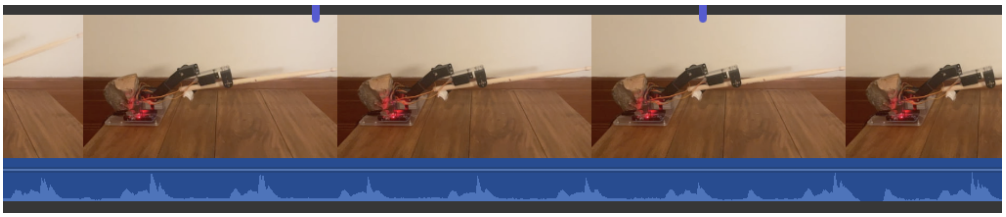

Figure 9: Different beats sound-wave with 25% volume and 30 bpm

6 Conclusion

The robotic drummer project successfully demonstrates the integration of mechanical, electrical, and software components to create a mechatronic system that can replicate human drumming motions. The use of servo motors allows for precise control of the robotic arm's movements, while the Arduino code implements the logic for generating different types of drum beats. The GUI provides a user-friendly interface for controlling the drummer's parameters, such as speed, volume, and beat type.

The project could be further enhanced by adding more servos to achieve a wider range of drum beats and by incorporating sensors to provide feedback on the drummer's performance. Additionally, the code could be optimized to improve the realism and fluidity of the drumming motions. Overall, the robotic drummer project serves as a valuable learning experience in mechatronics and robotics, showcasing the potential of these technologies in creative and artistic applications.