Acta Scientific COMPUTER SCIENCES

Volume 5 Issue 8 August 2023

Research Article

Design of a 3D Simulated MIDI-led Robotic Drummer

Mohamed Shahul Mubeen Ahamed Kabir Ribayee, David Reid and Emanuele Lindo Secco*

School of Mathematics, Computer Science and Engineering, Liverpool Hope University, UK

***Corresponding Author:** Emanuele Lindo Secco, School of Mathematics, Computer Science and Engineering, Liverpool Hope University, UK.

Received: May 26, 2023

Published: July 05, 2023

Abstract

This paper presents the design and development of a robotic device for musical performance. The robot is conceived to interact with MIDI interface and be able to perform music by means of a set of housing combined with servo actuators and a Maker Pi RP2040. All components are integrated into a Robotic Drummer which allow the percussion of multiple drums in accordance with the MIDI track. At this stage, assembly and performance have been simulated.

Keywords: Robotics Musical Instrument; Music Interaction; Low-cost Robotics

Introduction

The field of robotics has witnessed significant advancements in recent years, transforming the way we perceive automation and human-machine interactions. The field of music production, among the various fields of potential applications of Robotics, has emerged as a fascinating domain that combines technology, creativity, and performance. This project report explores the development of a robotic drummer, a simple prototype designed to augment musical performances and expand the possibilities of rhythmic expression.

The primary objective of this project is to design a robotic drummer capable of emulating the rhythm and dynamics of a human drummer, with the optional accessibility of a Musical Instrument Device Interface (hereafter referred to as MIDI) controller, using a Digital Audio Workstation (hereafter referred to as DAW). Throughout this report, the different stages of the development process will be delved into, starting from the initial conceptualization and design phase, to the implementation and testing of the robotic drummer. The selection and integration of components such as drums, motors, and the control system will be discussed as well.

Additionally, the software aspects of the project will be explored, with a focus on the utilization of CircuitPython as the Programming Language for controlling the robotic drummer. The development of logic to interpret user input, read drumming patterns, and dynamically respond to changes in the environment or performance requirements will be discussed as well.

Materials and Methods

The core of this project relies on the Maker Pi RP2040 board, powered by the Raspberry Pi RP2040 microcontroller. It features a Dual ARM Cortex-M0+ running at 133MHz and 264kB on-chip SRAM. It can also support up to 16MB of off-chip Flash memory, similar to the Raspberry Pi RP2040 [1]. The board's key advantage for this project lies in its built-in dual channel DC motor driver, 4 servo motor ports, 13 GPIO status indicator LEDs, and 7 Grove I/O connectors for sensors. These features make it an ideal choice for quick prototyping, allowing users to focus on coding [2,3]. The board comes preloaded with CircuitPython [3], simplifying coding and making it more user-friendly.

Micro servo motors are selected for the drummer robot's actuators. The idea is to design and 3D print a mallet link that attaches to the servo horn. When the servo motor is activated, the mallet will follow the horn and strike the percussion instrument.

Autodesk Fusion 360 (Autodesk®) is utilized as the software of choice for the design of the mallet, servo mount and the base. Once the designs are complete, the models are prepared for production. To showcase different views and dimensions, orthographic and isometric views are created and displayed below.

After finalizing the designs, the next step involves exporting the 3D models as. stl files. This file format is widely used for 3D printing and ensures compatibility with PrusaSlicer, a slicing software that is employed to optimize the models for printing on a Prusa Printer.

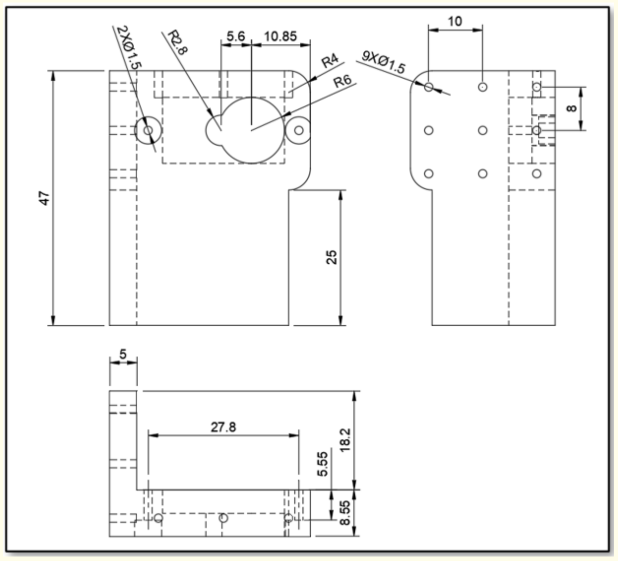

Figure 1: Schematic of the Servo Motor Housing.

Using PrusaSlicer, the exported .stl files are loaded, and the necessary print settings are configured. In this case, a layer thickness of 0.3 mm with an infill of 5% is chosen for the base, and 0.15 mm thickness with 15% infill for the mallet and the servo holder. These values are arrived by a trial-and-error basis, considering the quality and strength of the final product and speed of the print. After slicing the models, their respective G-codes are exported to then feed into the Prusa Printer. PLA is chosen as the print material for all the parts.

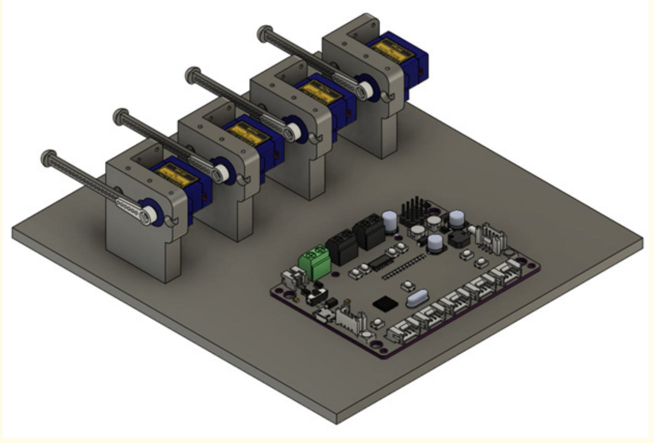

Figure 2: Integrated 3D assembly of the Servo Motor Housing and Maker Pi RP240.

Software

The code is written in Python with the Mu Editor, as it provides easy access to the board as well as the Read-Evaluate-Print-Loop (REPL) module. REPL provides a quick way of testing and debugging the Maker Pi before writing in the whole code to the board, using the Serial monitor which operates the same as a Python Shell.

First off, Mu Editor is opened and the mode is set to CircuitPython to enable REPL. Then the code reported in the Initialization and Setting sections was integrated.

Initialization

```
# import libraries
import time
import board
import pwmio
import digitalio as dio
import adafruit_midi
from adafruit_motor import servo
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff
import usb_midi

#define init position
def init_pos(a, b):
 for i in range(len(notes_list)):
 led_list[i].value = 1
 servo_list[i].angle = a
 time.sleep(0.5)
 led_list[i].value = 0
 servo_list[i].angle = b

# initialize some variables
global servo_retract_angle
global servo_hit_angle
global notes_list
global led_list
global servo_list

servo_retract_angle = 50        # calibrated according to the design
servo_hit_angle = 10            # calibrated according to the design
servo_frequency = 200           # default value for the servos
notes_list = [72, 74, 76, 77]   # C4, D4, E4 and F4 notes
led_list = []
servo_list = []

# initialize the LED pins to use (debugging)
led_pins = [board.GP0, board.GP1, board.GP2, board.GP3]
for pin in led_pins:
led = dio.DigitalInOut(pin)
led.direction = dio.Direction.OUTPUT
led_list.append(led)

# initialize the servo pins (actuating)
servo_pins = [board.GP12, board.GP13, board.GP14, board.GP15]
for servo_pin in servo_pins:
 servo_motor = pwmio.PWMOut(servo_pin, duty_cycle = 2 ** 15,
frequency = servo_frequency)
 motor = servo.Servo(servo_motor)
 servo_list.append(motor)

# make a reset button
button = dio.DigitalInOut(board.GP20)
button.direction = dio.Direction.INPUT
button.pull = dio.Pull.UP

# initialize USB MIDI input
midi = adafruit_midi.MIDI(midi_in=usb_midi.ports[0], in_chan-
nel=0)

# main loop
while True:

 # listen to MIDI messages on channel 1 of Ableton Live
msg = midi.receive()

 # checks for MIDI input and actuates the servos accordingly
for i in range(len(notes_list)):
 if isinstance(msg, NoteOn) and msg.note == notes_list[i]:
 led_list[i].value = 1
 servo_list[i].angle = servo_hit_angle
if isinstance(msg, NoteOff) and msg.note == notes_list[i]:
 led_list[i].value = 0
 servo_list[i].angle = servo_retract_angle

 # RESET code
if not button.value:
 init_pos(servo_hit_angle, servo_retract_angle)

 time.sleep(0.01) # to avoid CPU overloading
```

Such a system is then saved and labelled with code.py and copied to the CIRCUITPY drive inside the Maker Pi.

This makes the Pi run the code automatically on start-up, and unlocks the feature of using a dedicated power source to connect to the board; that way it will not depend on the computer's power through a USB cable.

Software Setup

The project incorporates the adafruit_midi library, which plays a crucial role in its implementation. The MIDI class provided by the library is utilized to configure the Maker Pi as an output device, allowing seamless recognition by Ableton. Subsequently, by employing the .receive() method, the Maker Pi becomes capable of listening to MIDI inputs originating from any source, utilizing Channel 1.

To capture note-on inputs from the MIDI device, the project utilizes the NoteOn object, while the NoteOff object is employed to detect note-off inputs [4]. These objects effectively enable the Maker Pi to respond to incoming MIDI signals and perform the necessary actions accordingly.

In addition, the adafruit_motor library is employed to facilitate servo motor control. This library offers essential methods and functionalities, including the generation of Pulse Width Modulation (PWM) cycles specifically tailored for servos. By leveraging these features, precise actuation of the servo motors becomes achievable.

To ensure compatibility and ease of code replication across various microcontrollers, the project also integrates the board library. This library provides a standardized set of classes that correspond universally to commonly used microcontroller platforms. This uniformity simplifies the process of replicating and reusing the code, eliminating the need for extensive pin configuration checks.

Figure 3 displays a representative session in Ableton Live with few MIDI notes written on the MIDI track.

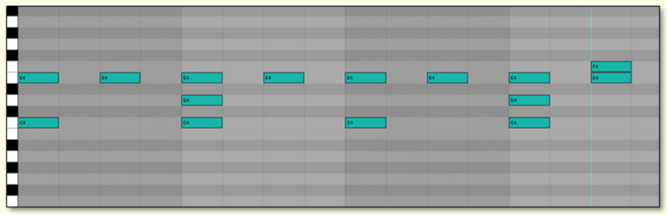

Figure 3: A MIDI Track in Ableton Live.

Once the MIDI track is set to output to CircuitPython Audio, the C4, D4, E4 and F4 notes control the servos connected to GP12, GP13, GP14 and GP15 respectively, of the board (i.e. the Maker Pi RP2040 [5]). It is also important to notice that the ON information from the MIDI track actuates the corresponding servo to the servo_hit_angle value, while Note OFF actuates to the servo_retract_angle.

Hardware setup

The Maker Pi RP2040 is powered up by connecting to the computer via a USB cable, and the power switch is turned ON. A new session in Ableton is opened and a MIDI track is set to send MIDI information to CircuitPython Audio through Channel 1. If needed, the computer's typing keyboard can be used as a MIDI keyboard to operate or possibly debug the robot live. Alternatively, a MIDI device can be connected to the computer, and can be set as an input to the same MIDI track to control the robot through a MIDI device. The mallet linked to the servos via a mechanical linkage are in turn actuated, which is designed in a way to hit the percussion elements when the notes are played, and retract when the former are stopped playing. The servo_hit_angle is where the mallet hits the percussion, and the servo_retract_angle is where it retracts to. These two variables can be adjusted according to the design.

Applications and extensions

A similar approach could be adopted in making robots which play other musical instruments, provided that the hardware and circuit are modified accordingly. Human-robot design [8,9] and interaction could also be investigated while the end-user communicates with the instrument with different means [10-14]. Since the robot is coded in a procedural manner, if in any case the coded is opted for reuse or reference, updating just a few variables according to the hardware changes will work perfectly, as long as the coding environment is kept the same. For example, 6 servos can be used to make a Robotic Guitarist, or 10 servos to play a Lyre Harp [6]. The possibilities are endless and with the right procedures and skill level, a whole Robotic Music Band can be made. The integration with a DAW also makes the robot play well with live instruments as it can be controlled with a MIDI interface connected to the computer, rather than to playback a pre-defined MIDI track.

Limitations

The processing and parsing of MIDI information typically require significant memory resources, which can result in delays when simultaneously processing multiple notes. This delay can lead to inconsistencies in servo actuation, causing a slight gap between notes and making it challenging to maintain synchronization with the desired tempo. Consequently, when servo motors are utilized as actuators, the drum's performance is limited. Higher tempos, particularly those above 150 BPM, may introduce further difficulties, and accurately playing 16th-length notes becomes problematic.

Conclusion

The development of a CircuitPython-based robotic drummer using Maker Pi RP2040 has been explored. The outcome of this project shows the potential of designing innovative musical instruments which are low-cost and that push the boundaries of traditional performance. Clearly, a set of further experiments should be performed in order to properly validate the proposed system. Moreover, a set of other actuator and sensors could be also considered and integrated within the system in accordance with other low-cost robotics systems. Such systems could be explored in terms of their different interaction with the musical instruments. In this context, other type of human-robot interaction could also be investigated while the end-user communicates with the instrument with different means.

Clearly the proposed system could also be improved with a further design where the selection of the actuators would take into account the possibility of incrementing the percussion speed vs the execution of more dynamic and fast tracks. At the same time, the system would benefit of a more interactive set of sensors allowing, for example, the visual reading of the musical score.

Through this project, the art of music is combined with robotic technology, opening new avenues for human-machine collaboration in the world of music.

Acknowledgements

This work was presented in coursework form in fulfilment of the requirements for the MSc in Robotics Engineering for the student Mohamed Shahul Mubeen Ahamed Kabir Ribayee under the supervision of EL Secco and D Reid from the Robotics Lab and AI Lab, School of Mathematics, Computer Science and Engineering, Liverpool Hope University.

Bibliography

1. Raspberry Pi RP2040 Documentation.
2. CytronTechnologies Maker Pi RP2040 Official Website.
3. CytronTechnologies Maker Pi RP2040 GitHub.
4. Adafruit MIDI Library API Documentation.
5. Maker Pi RP2040 Datasheet.
6. MIDI Controlled Robot Lyre with CircuitPython in Adafruit Official Website.
7. D Elstob and EL Secco. “A low cost EEG based BCI Prosthetic using motor imagery”. *International Journal of Information Technology Convergence and Services* 6.1 (2016): 23-36.
8. EL Secco and C Moutschen. “A Soft Anthropomorphic and Tactile Fingertip for Low-Cost Prosthetic and Robotic Applications”. *EAI Transactions on Pervasive Health and Technology* 4 (2018): 14.
9. VD Manolescu and EL Secco. “Design of an Assistive Low-Cost 6 d.o.f. Robotic Arm with Gripper”. 7th International Congress on Information and Communication Technology (ICICT 2022), Lecture Notes in Networks and Systems 1 (2020): 39-56.
10. K Brown., *et al*. “A Low-Cost Portable Health Platform for the Monitoring of Human Physiological Signals”. The 1st EAI International Conference on Technology, Innovation, Entrepreneurship and Education, (2017).
11. M Ormazabal and EL Secco. “A low cost EMG Graphical User Interface controller for robotic hand”. Future Technologies Conference (FTC 2021)”. *Lecture Notes in Networks and Systems* 2 (2021): 459-475.
12. EL Secco., *et al*. “A CNN-based Computer Vision Interface for Prosthetics’ application”. EAI MobiHealth 2021 - 10th EAI International Conference on Wireless Mobile Communication and Healthcare (2022): 41-59.
13. EL Secco., *et al*. “A CNN-based Computer Vision Interface for Prosthetics’ application”. EAI MobiHealth 2021 - 10th EAI International Conference on Wireless Mobile Communication and Healthcare (2022): 41-59.
14. Buckley N., *et al*. “A CNN sign language recognition system with single and double-handed gestures”. IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC) (2021): 1250-1253.