EC 535 Project: Electronic Instrument Simulator

Hin Lui Shum, Xulun Huang

30th April, 2024

1 Introduction

In the realm of embedded systems and digital music technology, our final project introduces an innovative electronic musical instrument utilizing the BeagleBone Black. This project synthesizes traditional musical interaction with modern computational capabilities, aiming to create a user-friendly and versatile musical experience. This report outlines the technical details, successes, and learning outcomes of developing an electronic instrument that not only simulates traditional music playing but also incorporates advanced digital functionalities.

1.1 Project Overview and Objectives

The motivation for this project stems from a personal connection to music and the transformative potential of modern technology in enhancing musical engagement. By integrating real-time sound generation of various instruments and adding capabilities like recording, playback, and MIDI conversion, our project addresses both the nostalgic appeal of traditional electronic keyboards and the demands of contemporary music production.

The primary objective was to develop a system that allows users to seamlessly switch between instruments like piano and guitar, and to interact with the device through a straightforward terminal-based menu. This approach was designed to prioritize auditory feedback and simplify the user interface, which originally included an LCD screen for displaying notes.

2 System Architecture

2.1 Hardware Implementation

- **BeagleBone Black revC:** Main computing unit.
- **16GB Micro SD Card:** Storage for operating system and applications.
- **Basic Circuit Components:** Breadboard, connecting wires, push buttons, and 10k Ohms resistors.
- **Audio Output and Connectivity:** USB port speaker, optional USB Hub, and USB WiFi adapter for network connectivity.

2.2 Wiring Configuration

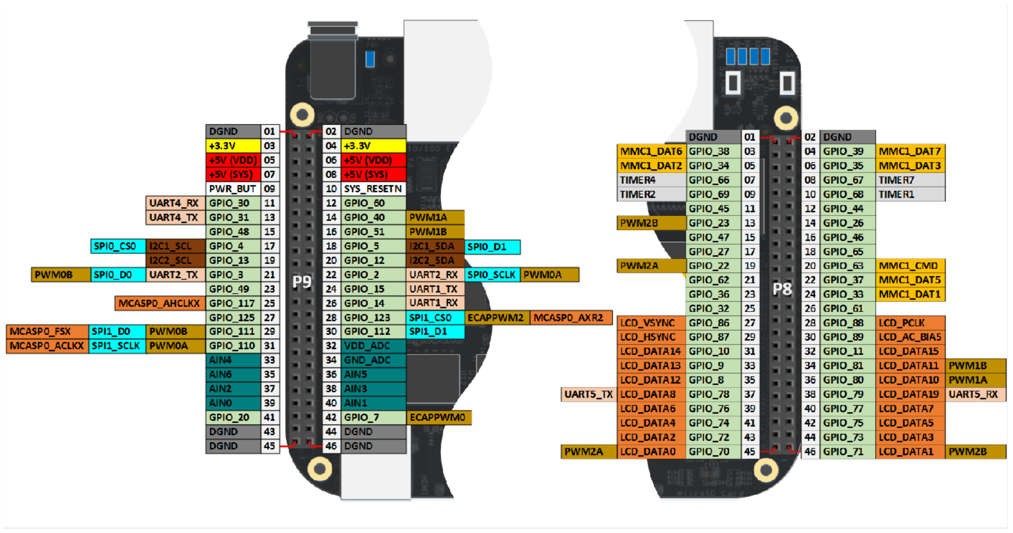


Figure 1: BeagleBone Black Pinout

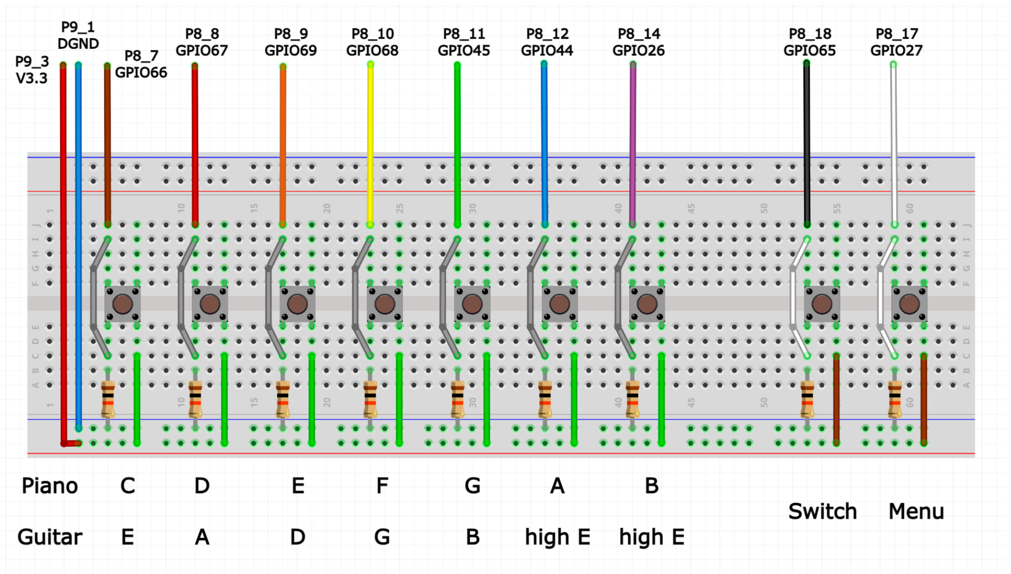


Figure 2: Circuit Design

2.3 Software Used

- **Operating System:** Am335x Debian 11.7 on BeagleBone.
- **Programming:** Python 3.x for script development.
- **Libraries:** Pygame for audio output handling and Adafruit_BBIO.GPIO for GPIO pin control.
- **Utilities:** Balena Etcher used for flashing SD cards.

3 Discussion of Failure

During the initial planning and implementation phases of our project, we encountered significant challenges related to audio processing within the Linux kernel module using C language. Recognizing these complexities, we reached out to Professor Montazam via email seeking guidance in late March. Unfortunately, due to unavailability or other constraints, we did not receive a response in time to influence our early decisions, thus we were compelled to make strategic decisions independently.

The primary technical challenges involved managing low-level operations such as buffer management, audio data streaming, and precise control over the DAC (Digital-to-Analog Converter). These tasks were further complicated by the absence of pre-installed audio management tools like ALSA (Advanced Linux Sound Architecture) on our image files from EC535 lab machines, USB ports for the speaker were also inaccessible.

Given these constraints, and the necessity to adhere closely to our project proposal, we decided to transition our development environment. We opted to flash offical Am335x Debian 11.7 on our BeagleBone board, which supports a wide range of high-level functionalities. This strategic pivot was driven by the need to leverage Python, a language that offers extensive library support and simplifies several aspects of audio processing.

4 Adaptations and Innovations

During the project's development, we faced a significant setback due to the limitations of the pre-installed image on lab machines, which did not support the integration of the LCD board initially planned. This limitation prevented us from implementing our original idea of displaying musical notes on a staff directly on the LCD screen. Despite this hurdle, we sought alternative ways to enrich the project's value and utility.
.

To compensate for the content loss and make the most out of our chosen development environment, we leveraged the flexibility and capabilities of Python. We introduced a groundbreaking feature by integrating our system with an online Large Language Model API. This new functionality allows our device to parse recorded musical messages into MIDI files, transforming raw audio data into a universally recognizable format that can be used in various digital music platforms.
.

The details and technical implementation of this innovative feature will be discussed in the next section, highlighting how we extended beyond the initial scope of our project to explore advanced applications and provide additional utility.

5 Successful Implementations

5.1 Instrument Sound Simulation

The core functionality of our electronic musical instrument involves the simulation of different instrument sounds, which is currently implemented for piano and guitar. This feature is facilitated by a streamlined setup comprising seven keys, each programmed to trigger specific notes corresponding to the selected instrument.

System Design and Flexibility:
The system's architecture is designed with modularity in mind, allowing for easy expansion. Currently, the system includes:

Technical Implementation:

- **Piano and Guitar Sounds:** Users can switch between these two instruments, providing a diverse auditory experience suitable for various musical styles and performances.

- **Sound Libraries:** Utilizes Python's Pygame library for sound output, which supports a wide range of audio formats and simplifies the process of adding new sounds.
- **GPIO Integration:** Each key's interaction is managed via GPIO pins on the BeagleBone Black, demonstrating effective use of hardware resources.

5.2 Implementation of Recording and Playback Functionalities

Our project successfully integrates advanced recording and playback functionalities, allowing users to capture and reproduce musical performances with high fidelity. This dual functionality enhances the user experience by enabling precise record and play back of the notes played.

Recording Functionality:

- **Multi-Instrument Recording:** The system captures each key press along with the timing and the specific instrument being used. This allows for complex recordings involving multiple instruments to be captured in a single session.
- **Data Capture and Storage:** Every aspect of the performance, including the sequence of notes, their timing, and the instrument selection, is recorded and stored. This data is crucial for accurate playback and further analysis.

Playback Functionality:

- **Accurate Reproduction:** The playback mechanism uses the stored timestamps to ensure that each note is reproduced exactly as it was played, mirroring the original performance with precision. This feature is essential for users who wish to analyze their playing, correct mistakes, or relive their musical expressions.
- **Educational Value:** By providing a faithful reproduction of the recorded music, the system serves as an invaluable tool for educational purposes, helping musicians refine their skills and technique through immediate feedback and consistent practice.

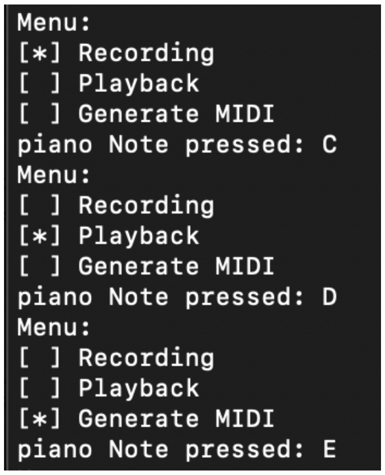
Technical Highlights:

- **Efficient Time-Stamping:** The system's ability to record precise time intervals between notes is a technical achievement that underpins the high fidelity of playback. This timing accuracy ensures that the nuanced dynamics and rhythm of the original performance are maintained.

5.3 Interactive Menu Design

The interactive menu of our electronic musical instrument is a notable achievement in the project, demonstrating effective use of limited input hardware and software design to enhance user interaction. This menu allows users to select functionalities such as recording, playback, and MIDI file generation, using just a few push buttons.

Design and Interaction:
The menu is designed to cycle through options using one button, with another using one of the key note to select the current option. his implementation showcases an exemplary use of limited hardware to achieve a functional and user-friendly interface. Not only it was efficient but also user-friendly, providing clear visual feedback on the terminal. Each option is highlighted with an asterisk (*) when selected, simplifying the user interface.



```
Menu:
[*] Recording
[ ] Playback
[ ] Generate MIDI
piano Note pressed: C
Menu:
[ ] Recording
[*] Playback
[ ] Generate MIDI
piano Note pressed: D
Menu:
[ ] Recording
[ ] Playback
[*] Generate MIDI
piano Note pressed: E
```

Figure 3: Interactive Menu Design

Technical Highlights:

- **State Machine Architecture:** Utilizes a state machine approach for clear and robust state transitions based on user inputs.
- **Efficient GPIO Utilization:** The design minimizes the number of GPIO pins required, leveraging them to handle multiple functionalities efficiently.
- **Software Debouncing:** Implements software debouncing to ensure reliable button press detection, enhancing the responsiveness of the menu.

5.4 Successful Implementation of MIDI Generation Feature Using OpenAI

The MIDI generation feature marks a significant technological advancement in our project, enabling the conversion of recorded musical events into MIDI files. This capability was realized through the innovative use of OpenAI's language model, which provide custom Python script tailored specifically for our needs whenever prompted.

Technical Integration with OpenAI:

- **Automated Script Creation:** Using openai API, we automated the complex process of generating Python scripts capable of parsing and converting our uniquely structured musical event data—tuples containing instrument types, notes, and precise timings—into standard MIDI format.
- **High Compatibility:** The generated MIDI files are universally compatible, allowing them to be seamlessly imported into any Digital Audio Workstation (DAW) for advanced musical production and editing.

Purpose and Benefits of the MIDI Feature:

- **Enhanced Accessibility:** Converting recordings into MIDI format opens up new possibilities for musicians to manipulate their performances within professional DAWs, bridging the gap between simple recordings and complex music production.
- **Facilitates Musical Exploration:** The MIDI files enable users to edit each note's properties extensively, such as pitch and duration, fostering creativity and detailed musical analysis.

6 Knowledge and Skills Acquired

Throughout the course of this project, we have significantly expanded my technical knowledge and skill set, particularly in the areas of embedded systems design, software development, and system integration. These skills not only contributed to the success of this project but have also prepared me for future challenges in embedded systems, software development, and beyond. Here are the key technical learnings from the project:

- **Embedded System Setup:** We learned how to configure and manage an embedded system using BeagleBone Black. This included installing and setting up a Linux-based operating system solely through terminal commands, which sharpened my command-line proficiency and understanding of Linux environments.
- **Networking and Peripheral Integration:** We gained practical experience in configuring network settings and connecting peripherals such as USB ports and WiFi adapters. This skill is crucial for developing IoT devices and other network-connected systems.
- **Audio System Configuration:** Setting up the audio system on a device without a graphical interface taught us about the underlying mechanisms of sound management in Linux, including driver selection, audio output routing, and software library usage for audio playback (Pygame).
- **Advanced Programming and API Integration:** Developing the system required advanced programming skills in Python, particularly in integrating with complex APIs like OpenAI's language models. This process involved understanding API documentation, handling JSON data structures, and writing robust code to interact with external services.

7 Project Resources

7.1 GitHub Link

`https://github.com/jshumhl/EC535-Electronic-Instrument-Simulator`

7.2 Youtube Video Demo

`https://youtu.be/dtELEZ3XwHg`

8 References

- `https://wiki.archlinux.org/title/Wpa_supplicant`
- `https://www.jeffgeerling.com/blog/2022/playing-sounds-python-on-raspberry-pi`
- `https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/gpio`
- `https://marcclifton.wordpress.com/2015/12/04/setting-up-beaglebone-black-with-debian-and-ssh-with-windows/`
- `https://files.beagle.cc/file/beagleboard-public-2021/images/am335x-debian-11.7-iot-armhf-2023-09-02-4gb.img.xz`
- `https://old.beagleboard.org/static/START.htm`
- `https://www.youtube.com/watch?v=c81tmb7WJxw`
- `https://jetforme.org/2015/01/podtique/`