Incorporating MIDI Functionality into an Acoustic Accordion

Adam Lauz

1 Introduction

Accordions, traditionally recognized as acoustic musical instruments, operate by utilizing air pressure to induce vibrations in internal metallic reeds, resulting in the production of audible sound. In recent years, there has been a notable convergence of technology and music, leading to the development of digital accordions. This evolution encompasses the creation of entirely digital accordion instruments, as well as the integration of electronic components into conventional acoustic accordions. These advancements enable the generation of digital soundscapes reminiscent of those produced by digital keyboards.

The transformation of an acoustic accordion into a digital sound-producing instrument typically involves the integration of physical sensors within the accordion apparatus. These sensors serve as intermediaries between the mechanical actions of playing the accordion and the digital realm. They are responsible for capturing key press actions and transmitting corresponding signals. Subsequently, these signals are transformed into Musical Instrument Digital Interface (MIDI) messages, which then interface with a dedicated sound module to produce an extensive range of auditory tones and effects.

The primary objective of this endeavor is the conversion of a traditional acoustic accordion into a versatile digital musical instrument, capable of producing a wide array of sounds and effects.

2 Outline

In this paper, we explore the integration of MIDI functionality into an acoustic accordion, transforming it into a versatile digital musical instrument. Our discussion is structured as follows:

1. Introduction: We introduce the traditional acoustic accordion and the recent convergence of technology and music in the development of digital accordions. Our primary goal is outlined.

2. Overview: We provide an overview of key concepts and technologies central to our project, including Arduino development, music-related use cases, MIDI, digital audio, analog Hall effect sensors, and multiplexers.

3. Components: We detail the essential components used in our project, encompassing sensors, the Arduino Uno microcontroller, multiplexers, an electronic matrix board, and a personal computer (PC) acting as a sound module.

4. High-Level Architecture: We delve into the core of our system, focusing on the Arduino Uno microcontroller, sixteen-channel multiplexer, and Hall effect sensors, as depicted in Figure 1.

5. Detailed Diagram: We present schematic diagrams illustrating the connections and interactions between various components in our system. These diagrams exclude the accordion itself for clarity, focusing solely on the electrical circuitry.

6. Key Press Detection Mechanism: We explain how physical key presses on the accordion are captured and translated into digital signals using linear-analog Hall effect sensors, as shown in Figure 3.

7. Algorithm Overview: We outline the algorithm that facilitates the integration of MIDI functionality into the acoustic accordion. This algorithm includes sensor initialization, MIDI message generation, and continuous sensor monitoring, as described in Algorithms 1, 2, and 3.

3 Overview

In this section, we provide a brief overview of key concepts and technologies central to our project, including Arduino development, music-related use cases, MIDI, digital audio, analog Hall effect sensors, and multiplexers.

Arduino Development: Arduino is an open-source electronics platform that offers a user-friendly, versatile environment for creating interactive electronic projects. Arduino boards are equipped with microcontrollers that can be programmed to read sensors, control various hardware components, and communicate with other devices. The Arduino Integrated Development Environment (IDE) provides a user-friendly interface for writing, compiling, and uploading code to Arduino boards.

Music-Related Use Cases: Arduino has found widespread application in the realm of music and sound. Musicians and makers use Arduino for a variety of music-related projects, ranging from instrument building and sound synthesis to interactive performances and installations. Arduino's flexibility and accessibility make it a valuable tool for exploring new avenues of musical expression.

MIDI (Musical Instrument Digital Interface): MIDI is a standardized protocol used in electronic musical instruments and computers for communicating musical information. MIDI messages include data on note events, control changes, pitch bends, and more. It serves as a universal language for connecting and controlling musical devices, enabling the exchange of musical data between instruments and software.

Digital Audio: Digital audio processing involves the conversion of analog sound waves into digital data and subsequent manipulation for various audio effects and synthesis. Arduino-based projects often leverage digital audio pro-

cessing to generate, modify, or process sound in real-time. This capability is essential for creating digital music instruments.

Analog Hall Effect Sensors: Hall effect sensors are transducers that respond to magnetic fields. Analog Hall effect sensors, such as the 49E model, produce an analog voltage output proportional to the strength of the magnetic field they are exposed to. These sensors are commonly used in musical instrument applications to detect the position and movement of magnetic elements, such as magnets on keys or strings, for precise control and triggering of musical events.

Multiplexers: Multiplexers (or Muxes) are electronic devices that allow multiple inputs to be routed to a single output. In our project, we utilize sixteen-channel analog-to-digital multiplexers (CD74HC4067) to efficiently manage a large number of analog sensor inputs. Multiplexers enable us to interface a substantial array of analog Hall effect sensors with the limited analog input pins available on the Arduino board.

Understanding these fundamental concepts and technologies is crucial for comprehending the integration of MIDI functionality into an acoustic accordion and the design of our digital music instrument system.

4 Components

The essential components employed in this project include:

- Sensors: A total of 40 Hall effect sensors (specifically, 49E analog sensors) and 40 miniature magnets.
- 1 Arduino Uno microcontroller with a USB cable for interfacing and control.
- 3 sixteen-channel analog-to-digital multiplexers (CD74HC4067) to manage the sensor inputs efficiently.
- An electronic matrix board to securely hold and interconnect the sensors.
- A personal computer (PC) utilized as a sound module to process and produce the desired audio output.

5 High-Level Architecture

The core of this system is centered around the Arduino Uno microcontroller, the sixteen-channel multiplexer, and the Hall effect sensors, as illustrated in Figure 1.

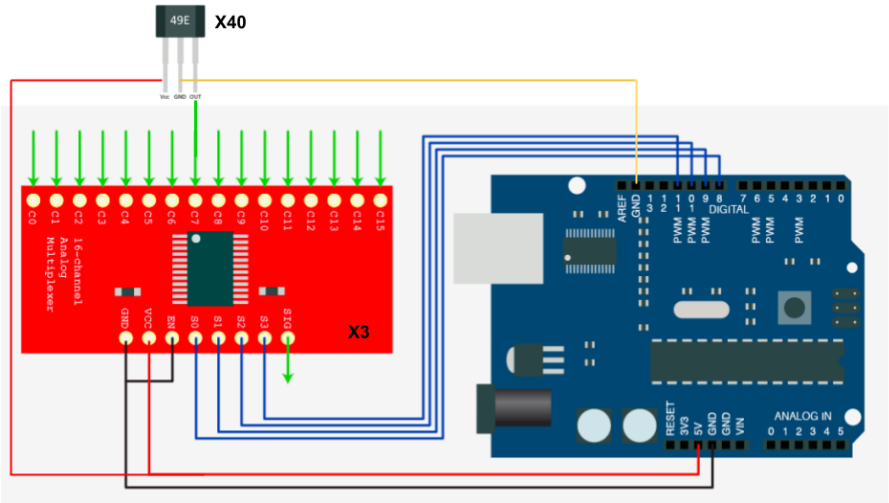

Figure 1: High-Level Architecture

6 Detailed Diagram

The schematic diagram and components (Figures 2 and 3) outlines the connections and interactions between the various components in the system. Notably, for clarity, the accordion itself is omitted from the diagram, focusing exclusively on the electrical circuitry.

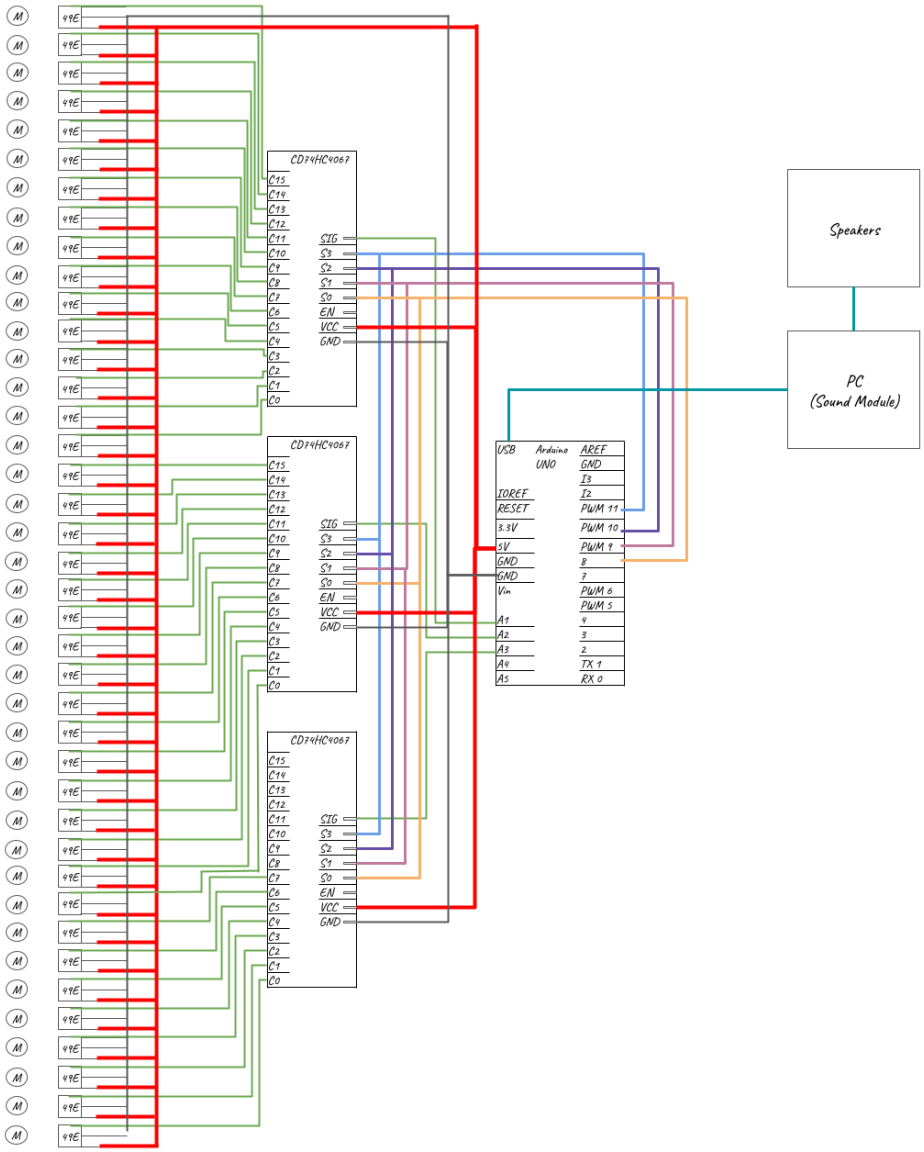

Figure 2: Detailed Diagram

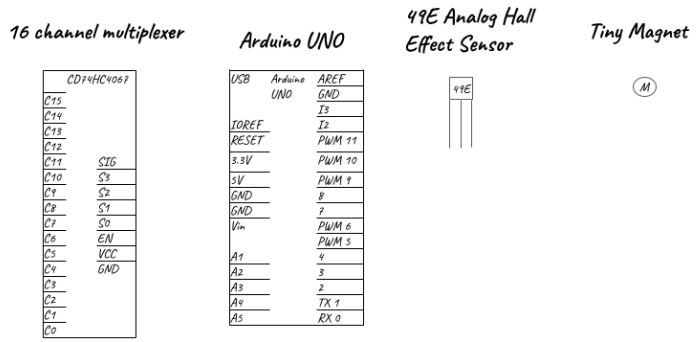

Figure 3: Diagram components

7 Key Press Detection Mechanism

Figure 4 serves to elucidate the method by which physical key presses on the accordion are captured and translated into digital signals. The linear-analog Hall effect sensors, such as the 49E model, produce signals of varying strength in response to the proximity of magnets.

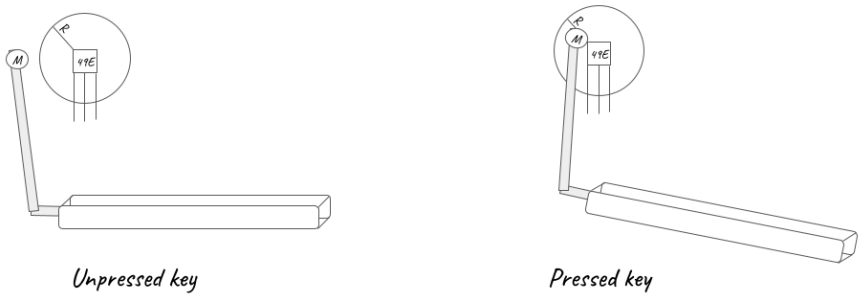

Figure 4: Key Press Detection Mechanism

8 Algorithm Overview

The system's algorithm orchestrates the integration of MIDI functionality into the acoustic accordion, enabling the generation of digital soundscapes. The algorithm consists of key components, including sensor initialization, MIDI message generation, and continuous sensor monitoring.

Algorithm 1 Sensor Initialization

```
1: procedure initializeSensors
2:     for i ← 0 to 2 do                          ▷ Iterate through multiplexers
3:         setMuxPins(i)                          ▷ Set multiplexer select pins
4:         for j ← 0 to 15 do                     ▷ Iterate through sensor pins
5:             readSensor(i, j)                   ▷ Read and initialize sensor values
6:         end for
7:     end for
8: end procedure
```

Algorithm 2 Continuous Sensor Monitoring

```
1: while true do                                          ▷ Continuously monitor sensors
2:     for i ← 0 to 2 do                                  ▷ Iterate through multiplexers
3:         setMuxPins(i)                                  ▷ Set multiplexer select pins
4:         for j ← 0 to 15 do                             ▷ Iterate through sensor pins
5:             value ← readSensor(i, j)                   ▷ Read sensor value
6:             updateSensorState(i, j, value)             ▷ Update sensor state
7:         end for
8:     end for
9:     sendMidiMessages      ▷ Send MIDI messages based on sensor states
10: end while
```

Algorithm 3 MIDI Message Generation

```
1: procedure sendMidiMessages
2:     for i ← 0 to 2 do                                  ▷ Iterate through multiplexers
3:         for j ← 0 to 15 do                             ▷ Iterate through sensor pins
4:             if isInZone(i, j) then     ▷ Check if sensor is in the active zone
5:                 channel ← getMidiChannel(i, j)         ▷ Determine MIDI
   channel
6:                 note ← getMidiNote(i, j)               ▷ Determine MIDI note
7:                 velocity ← calculateVelocity           ▷ Calculate velocity
8:                 noteOn(channel, note, velocity)        ▷ Send MIDI Note On
   message
9:             else
10:                noteOff(channel, note, velocity)       ▷ Send MIDI Note Off
   message
11:            end if
12:        end for
13:    end for
14: end procedure
```