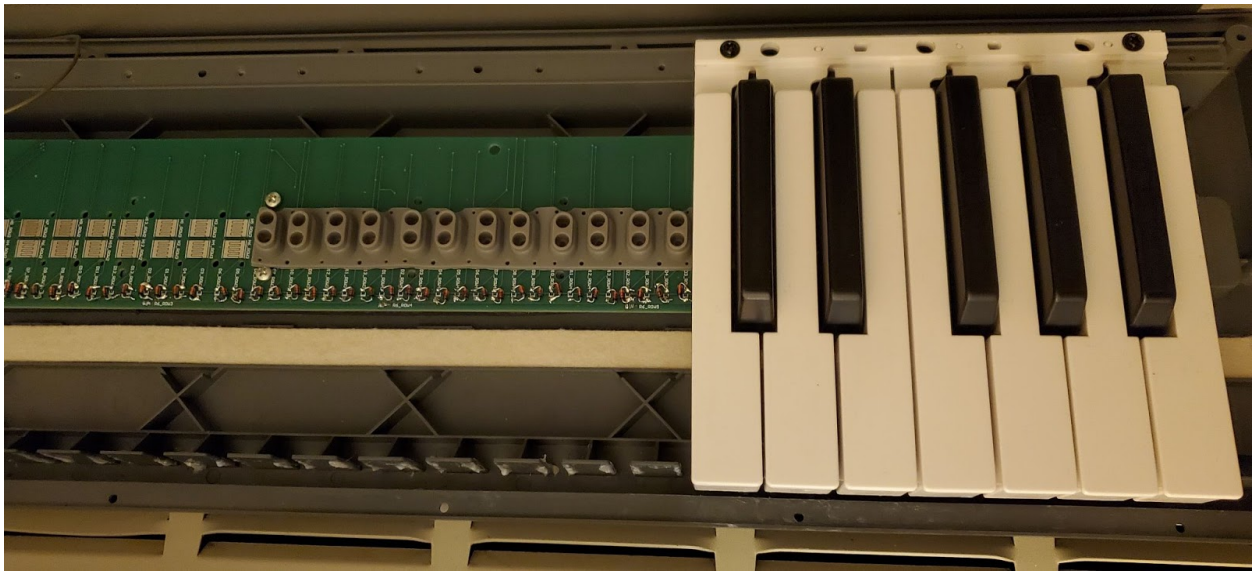
MIDUINO

Alex Ge
Richard Prescott

https://devpost.com/software/miduino?ref_content=my-projects-tab&ref_feature=my_projects



Final Project, Spring 2021
ESE350: Embedded Systems & Microcontroller Laboratory
University of Pennsylvania

Table of Contents

Abstract

For our project, we decided to construct and code a MIDI keyboard, which is a musical piano keyboard that is capable of transmitting information into a computer. MIDI is a protocol that sends information to musical software on the computer, such as Logic Pro X (or Garageband), which enables them to make sounds corresponding to played notes. We wanted to create such a keyboard to practice music during the pandemic, and we both have a love for music and creation. Since many of the practice rooms were closed, we sought out a keyboard for our own to experience music again.

Our solution and final product was a very ambitious keyboard, and we were able to achieve many of our goals. The final product contained a functional octave, with four notes that worked as intended, and others that tended to have glitches. Regardless of the actual functionality, we believe our project succeeded, as we learned a great deal of hardware and software while being able to actually create sound!

Motivation

Because of COVID, many of the practice rooms in our dorms have been closed, and we sought a way to recreate music while integrating embedded systems with our interests. A MIDI keyboard is a personal, portable way to practice music, and is extremely versatile due to its use of MIDI, which sends information that can be mapped to any instrument in software. This makes MIDI keyboards particularly useful for composition and music production. They are also just very cool!

Goals

A. Milestone

We expect to have a fully functioning push-button based keyboard, with a single octave of white keys only implemented. The circuitry should be done on a breadboard and interfaced with an Arduino Uno, and the device should be able to interact with a DAW on a computer, such as Logic Pro X in our case.

B. Final Demo

A good intermediate step would be to implement velocity sensing, but by the final product, we plan to have implemented 3 complete octaves of a full-sized, membrane-switch based keyboard. The keyboard should be velocity sensitive and have a potentiometer for volume control. If extra time permits, we may implement extra features, such as a servo motor metronome, LED lighting under the keys, a “dancing” actuator, or a sustain pedal.

Methodology

MIDI Protocol:
To implement MIDI protocol, we plan to write our own library, since MIDI is simply made up of UART commands (serial communication). More specifically, it is made up of 3 UART commands, which determine whether a note is on, which note is played, and velocity of the note respectively.
Switch Matrix:
For the circuitry, in order to reduce pin usage on the arduino, we plan to introduce a switch matrix with multiplexing. By tying the high ends of a group of switches together, we can functionally create a column. Then, by tying the low ends of another group of switches, we can create a row. Using the 16MHz clock of the arduino, we can then read the status of the board quickly to update what notes are being played at any time.
Velocity Sensing:
To implement velocity sensitivity, we plan on using Timer functionality. Each key will be made up of two switches, one closer to the back of the keyboard, and one closer to the front. Since the velocity does not really need to be exact and can be approximate, we will probably simply count the number of timer overflows between the first and second switches are pressed, which will vary depending on how hard or fast one presses the key. We can then translate that number into a velocity.
Volume Control:
For volume control, we will be simply feeding a potentiometer into the ADC pin, and use the ADC pin to convert the potentiometer to a continuous volume stream. Depending on the amount of volume into the ADC, the volume will change (which depends on the potentiometer).

Results

Goal A of Milestone was definitely achieved, as we were able to have a fully functional prototype with push-buttons! The push-button prototype was an octave, and was also velocity sensitive, so we were able to create a very comprehensive minimum viable product. The device was also able to easily communicate with Logic Pro X and transmit information seamlessly.

Goal B of Final Demo was not quite achieved, as the final product fell a bit short of our expectations. We were unable to achieve full 3-octave, velocity sensitive functionality, but we were able to achieve a functional octave of sounds. We were also able to successfully implement the switch matrix for some notes, and soldered the entire PCB to completion (although, with a little trouble, due to a mishap in the process). We also learned a lot of software through writing our own library, flashing with Atmel Ice, and implementing pin arrays.

Conclusion

This project was a tremendous learning experience, and we had a lot of fun throughout the process. We learned about switch matrices, hex flashing, programming with Atmel Ice, PCB Design, membrane switches, pad design, and integration between mechanical, electrical, and software parts.

We were very proud of our PCB, which we designed, routed, and soldered by ourselves. We were also very happy with the switch matrix configuration we designed, and with the software portion, which was heavily dependent on HIDUINO, external firmware that masked our Arduino into a MIDI Device, allowing Logic Pro X to recognize the Arduino for transmission purposes. We also were very proud of our final demo video, which we put a lot of effort into and had a lot of fun making.

The most successful part of the project was definitely the first milestone demo, in which we were able to actually hit all of our goals and objectives with the simple buttons! The buttons did not have many mechanical issues (unlike the membrane switches), which meant all of our theory followed through beautifully to the keyboard we created on a breadboard. We were able to get nice sounds in the way we expected with our code, and also implement velocity sensing by the end of the prototype.

However, there were many mishaps in the project, and we have a lot of steps that we would change, if we could. For example, the PCB was actually designed in the wrong

way-as in, the pads for velocity detection were ordered by Rows instead of Columns, which caused a large headache when it came to writing the code. We also forgot to order our components when we got our PCB, so we had to solder 64 through-hole diodes onto 0603 pads on our PCB over a single day! It was quite exhausting, but we got really good at soldering through-hole components by the end of it. We also would have changed our approach in the software for velocity-sensing implementation, as we were using nested if loops, but realized that we could instead capture the state of the keyboard at a certain fixed frequency, and that would have made the ordering by Rows mistake obsolete.

We definitely did not anticipate not having components, and the through-hole diodes actually hit our keys a little when we put them into the board. Additionally, we did not anticipate the 22 inch limit on the PCB, which then delayed the time until we were able to actually get the board to test the final product with. We definitely would have wanted more testing time, if possible.

Another unanticipated challenge were the membrane switches, which had many mechanical issues. For one, they debounced quite a bit on the actual keyboard, which made consistent testing quite difficult. Since they also bent in a weird way, the keys felt a little off depending on which way we oriented them.

As a next step, we of course could continue to refine the keyboard and try to implement all of the keys in the 3-octaves we wanted. We could also try to fix the faulty keys in our functional octave before moving on to the rest. Additionally, we could try the new software capturing implementation, and also move it from an Arduino to a more commercial and professional microcontroller. Finally, we could also order components and solder the correct surface mount pieces onto the board.

References

https://github.com/ddiakopoulos/hiduino
https://www.dribin.org/dave/keyboard/one_html/
https://www.youtube.com/watch?v=wyKStRyez5Y&t=609s&ab_channel=GreatScott%21
https://www.midi.org/specifications
http://www.openmusiclabs.com/learning/digital/input-matrix-scanning/keyboard/index.html
https://piecesofpi.co.uk/arduino-midi-keyboard-part-6-velocity-or-is-it-volume/

Appendix A

https://github.com/ddiakopoulos/hiduino

HIDUINO is firmware that was hex-flashed onto the Arduino which masked the Arduino as a MIDI input device on the computer, such that a DAW like Logic Pro X was able to recognize it as a MIDI device. Since we flashed it onto the 16u2, for the rest of the project we flashed code through an Atmel Ice directly to the Atmega328P, since we did not want to overwrite the 16u2.