Simple MIDI Controller

Adithya Shastry

November 30, 2018
CS342:Project 4

1 Overview

In this project I constructed a simple MIDI controller using the piezo element, a potentiometer, and a capacitive touch sensor. The MIDI controller allows the user to play notes and change the frequency of notes on the fly! The project uses two inputs, the touch sensor will be using the I2C communication protocol and the dial will be using an ADC(Protocol that converts analog signals to digital signals on the ATmega board) to set the proper frequency for the sound. Finally, the piezo element will serves as an output by outputting sound. I wanted to design this project because I want to play some sick beats!

2 Part Requirements

Part	Amount
Computer with Arduino IDE	1
Metro Mini	1
Data cable	1
Breadboard(Full or Half Sized)	1
MPR121 Capacitive Touch Sensor	1
Piezo Element	1
Insulated Copper Wire	about 0.5m
strips of conductive tape	33
Alligator Clip Wires	Optional based on need
Pontentiometer	1
ATmega Data sheet	1
MPR121 Data sheet	1

Table 1: Part Requirements

3 Circuit Implementation

When creating the keyboard, make sure to put one strip of tape below and over the wire so that there is no issue in terms of the wire conducting current. You can also use anything that can conduct current! In fact, I made a separate iteration of the MIDI controller that used fruits and other household items as keys instead of the copper strips. For these types of creative improvements on the project, it is best to use alligator clips to connect the object to the capacitive touch sensor.

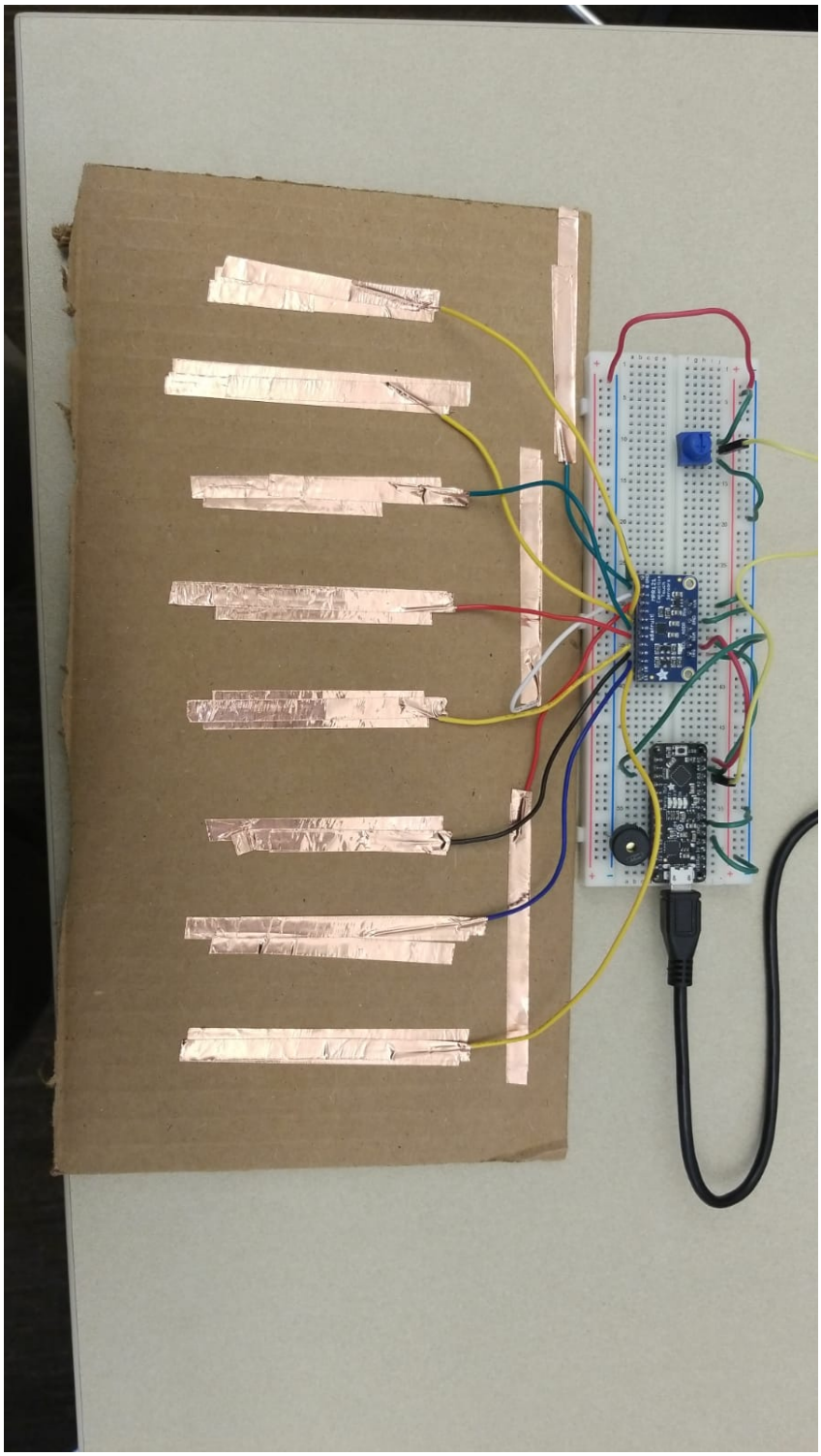

Figure 1: Hardware Implementation

4 Coding Implementation

```
1
2  #include <avr/io.h>
3  #include <Wire.h>
4  #include "Adafruit_MPR121.h"
5  #include "USART.h"
6
7  float frequency [12] = {16.35, 17.32, 18.35, 19.45,
8  20.60, 21.83, 23.12, 24.50,
9                          25.96, 27.50, 29.14, 30.87
10                        };
11 int octave = 4;
12 int duration[12] = {500, 500, 250, 250, 250, 1000, 500,
13                    500, 750, 250, 250, 750
14                   }; // in ms
15
16 Adafruit_MPR121 cap = Adafruit_MPR121();
17
18 /* For tracking MPR121 state changes (touches and
19 //releases) */
20 uint16_t currTouched = 0;
21 int note;
22 char t[100];
23 int ms = 0;
24
25
26 uint16_t sensedADC=0;
27 float mean=0;
28 char msg[124];
29 int angle;
30
31
32
33
34 ISR(ADC_vect){
35   int N=10;
36   sensedADC=ADCL;
37   sensedADC|=(ADCH &0x03)<<8;
38   mean=mean *float(N-1)/float(N)+float(sensedADC)/float(N);
39
```

```
40
41    angle=int(mean*(360.0/1023.0));
42  //  sprintf(msg, "online mean: \t%d\n",int(mean));
43  //  printString(msg);
44    sprintf(msg, "online angle: \t%d\n",angle);
45    printString(msg);
46  }
47
48
49
50
51
52  ISR(TIMER1_COMPA_vect) {
53
54
55  }
56
57  ISR(TIMER0_COMPA_vect){
58
59  }
60  int main() {
61
62
63    init();
64
65    initUSART();
66
67   if ( !cap.begin(0x5A) ) {
68   // Test that the sensor is up and running
69
70      while ( true );
71    }
72
73    currTouched = cap.touched();
74     // Get the currently touched pads
75
76    // TODO: Configure the GPIO
77
78
79    //Configure pad 3 to be an input
80    DDRC = 0b00100000;
81    DDRD = 0;
```

```
82    PORTD = 0b00000100;
83
84    // Configuring Timer 1 to generate a sound wave
85    DDRB = 0x02; // PB 1 is output
86    TCCR1A = 0b01000000; // CTC mode - pg 141 and toggling
87    //COM1A?
88    //TCCR1B = 9;
89    OCR1AH = 0x77;
90    OCR1AL = 0x71;
91
92
93
94  //WE will use TIMER0 for the ADC
95   TCCR0A = 2;
96    TCCR0B = 2;
97    TIMSK0 = 2;
98    OCR0A  = 200;
99    //Configure an analog read from pad 3
100 ADMUX=0b01000011;
101 ADCSRA=0b11101111;
102 ADCSRB=0b00000011;//Using TIMER0_COMPA_vect
103
104
105
106
107
108   //TODO: configure the interrupt (but do not enable its
109 //ISR)
110   EICRA = 0b00000010;
111   /* Global interrupt enable */
112   SREG |= 0x80;
113
114   /* Control loop */
115   while ( true ) {
116
117     printString("");
118
119     // TODO: If the interrupt flag is high, the MPR121's
120     //state has changed.
121
122     // TODO: Outside of an ISR, we have to manually clear
123     //the interrupt flag.
```

```
124       // TODO: Read in the new MPR121 state
125       if (EIFR & 0b00000001) {
126         ADCSRA=0b01101111;//THis will set the ADC to off
127
128         EIFR=1;
129         currTouched = cap.touched();
130         ADCSRA=0b11101111;//Re-enable the ADC
131
132         if ( currTouched > 0 ) { // a capacitive touch pad is being touched
133
134           note = int(log(currTouched) / log(2));
135           if (note >= 0 && note <= 11) {
136             TCCR1B = 0b00001001;
137             float t = frequency[note] * pow(2, octave + 1);
138             //This will change the tone number based on the angle
139             t=t-angle;
140             t = 16000000 / t - 1;
141             OCR1AH = (uint16_t(t) & 0xFF00) >> 8;
142             OCR1AL = (uint16_t(t) & 0x00FF);
143           }
144
145         } else {// all capacitive touch pads have been released
146           TCCR1B = 0x00;
147
148         }
149         //EIFR = 0xFF;
150
151       }
152     }
153     return 0;
154 }
```

5 Testing and Possible Further Improvements

Testing the functionality of the MIDI controller was done qualitatively by attempting to change the tone of the notes being played and hearing the result. I also had my friends try to play the MIDI controller in order to see if the design of the project was easy for the average user to pick up and use without much trouble. Further improvements for the project might be to add more keys to the board and more ways to change the sound coming out of the piezo element.

6 Conclusion

Overall, this was a very fun project to work on not only because it gave me a cool project to show off to my friends, but also because it helped me really understand how to deal with issues related to conflicting interrupts and the priority with which they are handled and also how to deal with importing libraries into projects and understanding the things they are changing on the board.