Disciplina: Projeto de Placa de Circuito Impresso | Professor(a): Joabel Moia

Curso: Engenharia Eletrônica

Câmpus Florianópolis

Alunos: André Luiz Gonçalves Toldo e Leonardo Besh

Projeto Final: Guitar Hero

Introdução

O intuito do projeto foi criar um dispositivo que fosse capaz de tocar notas musicais e trocar o instrumento que está sendo tocado com base na posição do dispositivo (X,Y,Z). O programa foi desenvolvido em *Cpp* e gravado em um Atmega328P.

Desenvolvimento

1. Componentes e Shields

Foram utilizados:

1x Arduino Uno

1x Shield MP3 Vs1053b

1x KeyPad 4x4

1x Shield MPU 9050

1x Fone de ouvido

2. Uso de cada componente

Arduino: foi usado como centro de processamento de todos os sinais

Shield MP3: serve para gerar os sons, para isso usamos a função MIDI desse componente (Ver MP3 Shield MIDI).

KeyPad: foi usado como dispositivo para iniciar o som de cada tecla, porém poderiam ser usadas qualquer matriz de botão, por exemplo: o braço de uma guitarra do *guitar hero* como era o objetivo inicial do trabalho, (o código foi feito para

suportar apenas 16 botões, porém pode ser adaptado facilmente para mais ou menos botões)

Shield MPU 9050: Usado para adquirir a posição do dispositivo em (X,Y,Z).

3. Bibliotecas Usadas

Foram incluídas as bibliotecas twi, usart0, timer2 (apesar de incluída, não foi usada), globalDefines e keypad do professor Leandro Schwarz.

Twi e usart0 foram usadas para inicializar os periféricos I2C e UART do atmega para comunicar com o MPU 9050 e o MP3 Shield respectivamente.

Keypad foi usado para a adquirir uma entrada de nota e o globalDefines contém definições importantes para o Atmega.

4. MP3 Shield MIDI

Para o funcionamento do MP3 Shield o seu pino MP3-RST tem que estar em HIGH para shield funcionar, após isso para habilitar o MIDI tem-se que dar um reset no Shiel (ou fazer quando ele estiver ligando) um pulso (100 ms) em seu GPIO 01 (HIGH - LOW - HIGH), assim a porta MIDI-IN entrará em funcionamento e o chip poderá trocar MIDI por essa porta.

O protocolo MIDI é baseado em mensagens transmitidas via protocolo UART com 0 bits de paridade 1 stop bit e 8 bits de dados por mensagem, para completar uma comunicação são usadas de 2 a 3 mensagens. Artigo usado como referência: https://www.cs.cmu.edu/~music/cmsip/readings/MIDI%20tutorial%20for%20programmers.html.

O programa feito para realizar a comunicação se vale do periférico UART e de uma estrutura chamada MIDI ou Midi_t, contendo as características do MIDI *status, data byte 1, data byte 2, data sent*. Sendo, o status o byte que define a função dos próximos bytes, *data* são os próprios dados, *data sent* são quantos bytes mandar (2 ou 3, contando com o status) e *chanel* canal do MIDI (suporta 16 canais no máximo, usa o GM1 *General Midi 1*).

O fone de ouvido foi conectado no MP3 Shield, mas esse shield poderia ser conectado a uma caixa de som também, o único cuidado a se tomar é que a sua referência de tensão (GBUF, ou sleeve do conector) é de 1,25 V, portanto é

necessário um circuito desacoplador de CC, em caso da saída ser conectado a um terra comum ao Shield e o amplificador ou caixa de som.

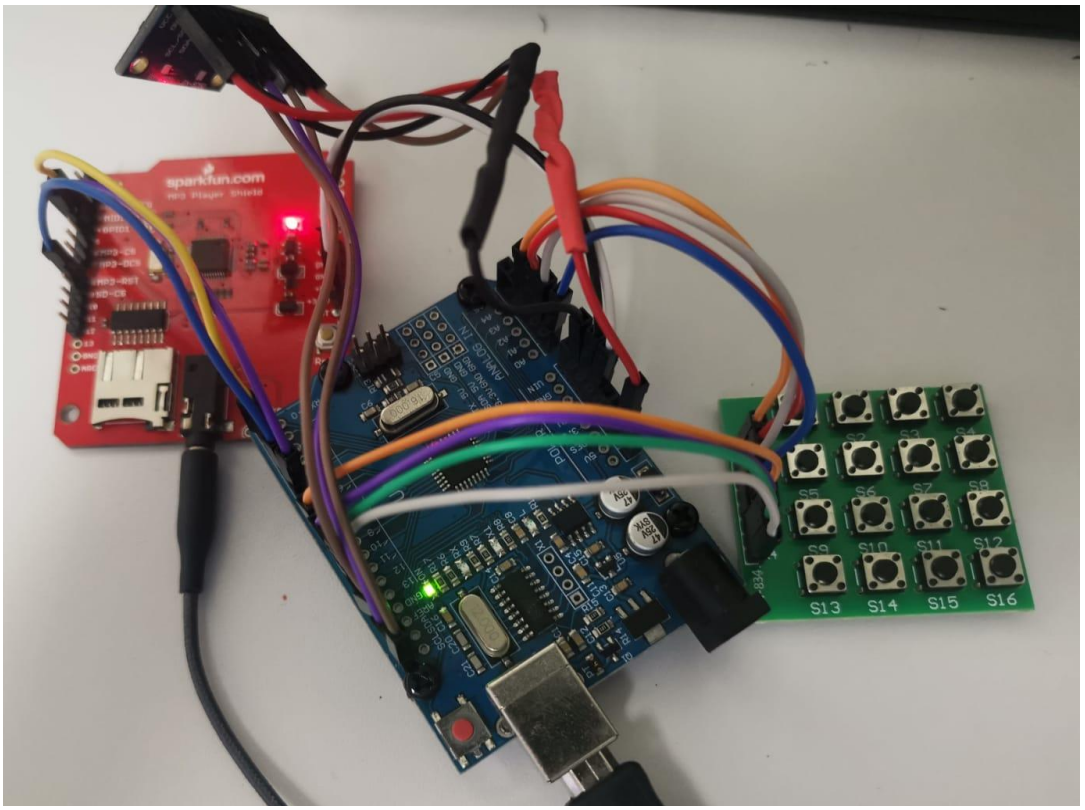
5. MPU 9050

Com o MPU 9050, o único cuidado que precisamos tomar é verificar se a comunicação com o módulo está funcionando, para isso, inicializamos o I2C usando a biblioteca e mandamos o arduino ler o registrador que contém o endereço do MPU 9050 e depois, se a comunicação é bem sucedida, ler os 3 bytes de aceleração (X,Y,Z) para tirar os valores anteriores do acelerômetro.

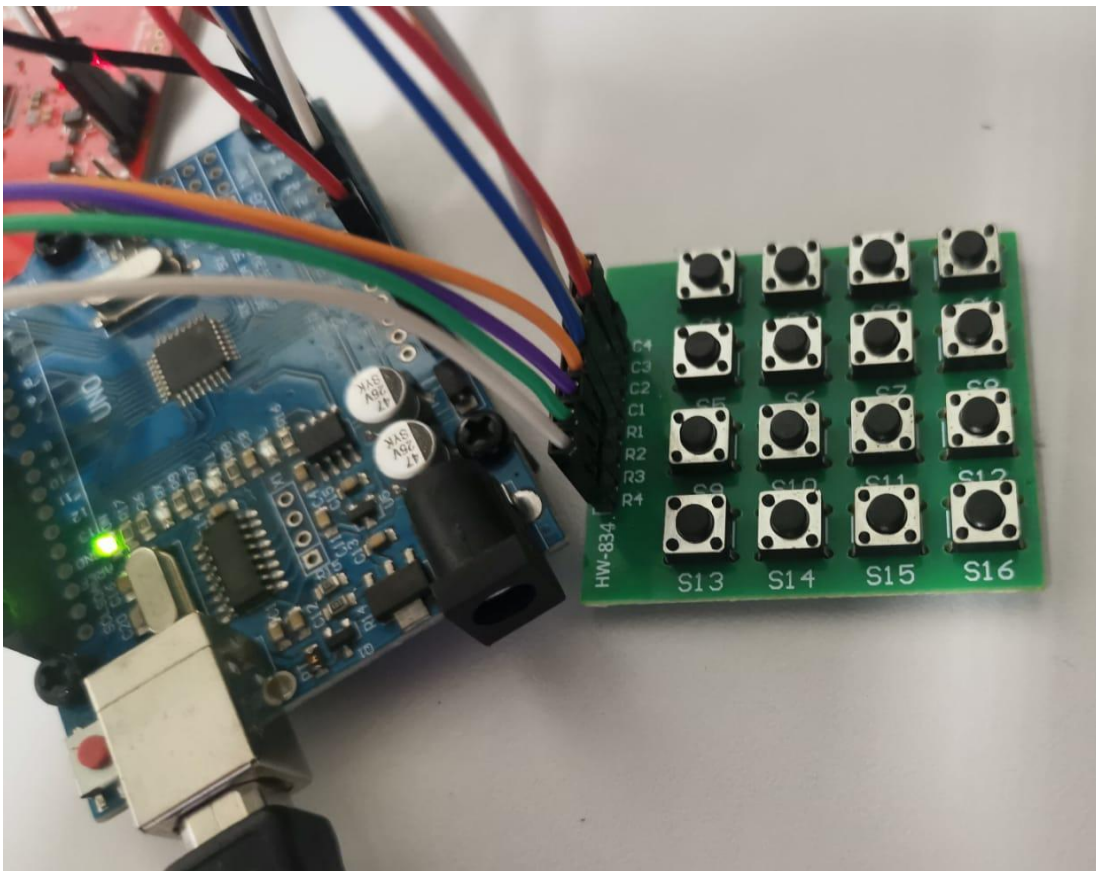
Depois de inicializado nós usamos esse Shield para nos dizer em que faixa de valores estavam as componentes X,Y e Z da aceleração, e atribuímos um instrumento para cada faixa.

Não usamos mais nenhuma funcionalidade do MPU 9050, como giroscópio e magnetômetro, nem mesmo a pilha interna que ele possui para armazenar os valores periodicamente.

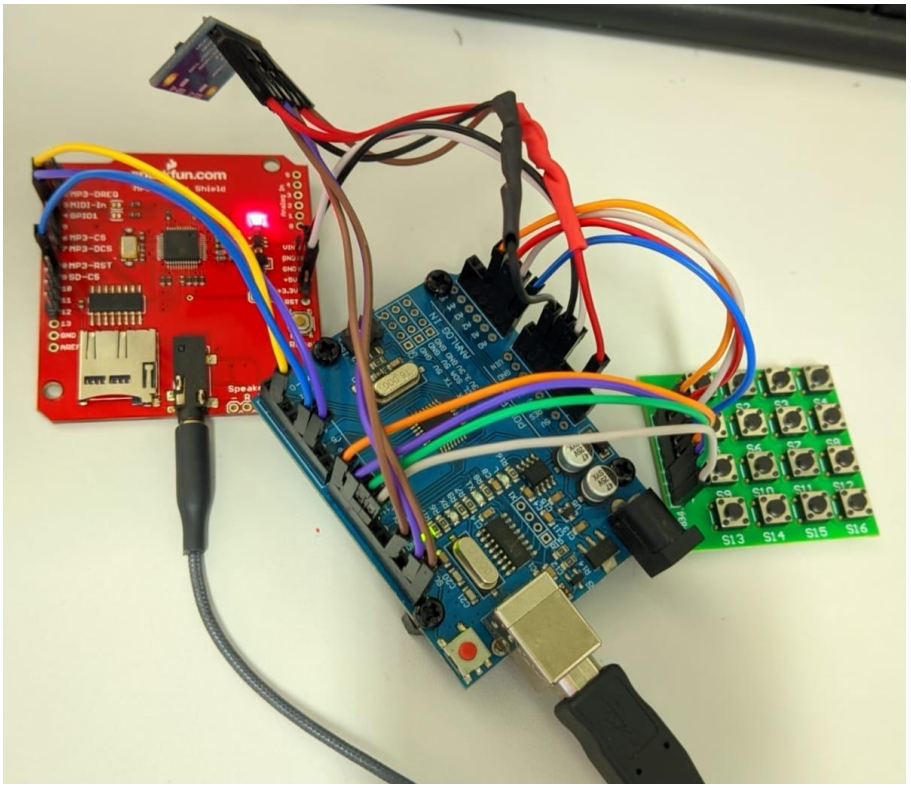
6. Conexões



Fonte: Autoria Própria



Fonte: Autoria Própria



Fonte: Autoria Própria

7. Código

Como o código final ficou com 735 linhas e contém músicas descritas em MIDI é mais fácil colocá-los em um repositório GIT, todos os códigos estão em: https://github.com/xamafilmes/Project_MIDI_Atmega.

Os pontos principais:

```
// os returns em cada instrução foram usados para
que a função não tentasse
// transmitir outro dado para o UDR0 sem esse
estar zerado denovo
void usartTransmissionBufferEmptyCallback()
{
    // De acordo com o valor em
Midi_Variable.DATA_SENT manda os dados em
Midi_Variable.STATUS_BYTE
    // para noteon e noteoff manda 3 dados, para
troca de instrumento manda 2 dados
    if((Midi_Variable.DATA_SENT & 0b111) >= 4) {
        UDR0 = Midi_Variable.STATUS_BYTE;
        Midi_Variable.DATA_SENT =
Midi_Variable.DATA_SENT - 4;
        return ;
    }
    if((Midi_Variable.DATA_SENT & 0b111) == 3) {
        UDR0 = Midi_Variable.DATA_BYTE1;
        Midi_Variable.DATA_SENT =
Midi_Variable.DATA_SENT - 2;
        return;
    }
    if((Midi_Variable.DATA_SENT & 0b111) == 2) {
        UDR0 = Midi_Variable.DATA_BYTE1;
        Midi_Variable.DATA_SENT = 0;
```

```
        clrBit(UCSR0B, UDRIE0); // disenables
interrupt on the UDREn
        return;
    }
    if((Midi_Variable.DATA_SENT & 0b111) == 1) {
        UDR0 = Midi_Variable.DATA_BYTE2;
        Midi_Variable.DATA_SENT = 0;
        clrBit(UCSR0B, UDRIE0); // disenables
interrupt on the UDREn
        return;
    }
}
```

E também:

```
    twi.init(10000);

    twi.setDevice(MPU9250_SLAVEADRESS); //
endereço do módulo
//    // envia o que esta no buffer para o
registrador do módulo
    // nao usa twi.writeReg();
//    // lê o que esta no buffer para o
registrador do módulo

    // Lê os valores dos registradores high do
módulo
    twi.readReg(0x3B, &AccelX, 1);
    twi.readReg(0x3D, &AccelY, 1);
    twi.readReg(0x3F, &AccelZ, 1);
```

```
    while(1) {

        twi.readReg(0x3B, &AccelX, 1);
        twi.readReg(0x3D, &AccelY, 1);
        twi.readReg(0x3F, &AccelZ, 1);
        // Converte os valores do acelerômetro
para valores com sinal
        AccelXConv = (int8_t)AccelX;
        AccelYConv = (int8_t)AccelY;
        AccelZConv = (int8_t)AccelZ;

        // valores para instrumento 0; 16 orgao;
46 harpa; 79 ocarina;
        //                          26 guitarra
jazz; 19 orgam igreja;
        // Seleciona o instrumento de acordo com o
acelerômetro
        if(AccelZConv > 49) {
            instrumento = 0;
        }
        if(AccelZConv < -49) {
            instrumento = 16;
        }
        if(AccelYConv > 49) {
            instrumento = 19;
        }
        if(AccelYConv < -49) {
            instrumento = 26;
        }
        if(AccelXConv > 49) {
```

```
            instrumento = 46;
        }
        if(AccelXConv < -49) {
            instrumento = 79;
        }
```

Considerações Finais

Apesar do inicialmente planejado fosse implementar esse projeto em uma guitarra de brinquedo a falta de tempo levou a execução em um keypad de 4x4, mas o projeto pode ser transportado sem muitos ajustes em código para o brinquedo.

O pooling das notas no teclado não deixaram que houvesse a formação de acordes, mas uma solução para isso é a utilização do timer 2 do atmega328P para cronometrar o tempo de desligamento das notas, além de um buffer contendo a nota a ser desligada e sua oitava.

Em caso do leitor não ter entendido muito bem como o MIDI funciona, é muito recomendado ler o artigo referência do MIDI e ao longo da leitura comparar com o código desenvolvido nesse projeto.