Modmixer: Relatório Final

:Universidade Federal de Santa Catarina
Campus Tecnológico de Joinville
Projeto integrador II

Grupo 2: Modmixer

1st Arthur Henrique Mallmann
15159540
Montagem CAD

2nd Assis Gabriel Greselle
16150414
Montagem Eletrônica

3rd Guilherme Turatto
16201836
Comunicação Serial

4th Gabriel Cesar Silveira
16203673
Interface gráfica

5th Gabriel Luiz Martins
Matricula

***Abstract*—Este relatório descreve o projeto e desenvolvimento de um controlador modular de som para aplicações Linux, abordando seu *hardware*, *software* embarcado e de interface gráfica.**

***Index Terms*—Controle, Som, Linux**

I. Introdução

Este trabalho propõe o projeto e desenvolvimento, de *hardware* e *software*, de um controlador de som de baixo custo para aplicações Linux em *desktops*. O usuário do sistema Modmixer deve ser capaz de aumentar ou diminuir o volume de qualquer aplicação sendo executada em seu computador.

Para este controle, o sistema conta com uma interface gráfica para atribuição das aplicações aos módulos. Os módulos físicos são dispositivos, conectados serialmente à porta USB, que possuem potenciômetros para definição do nível de volume desejado.

A execução da interface é feita, após a instalação das dependências listas no arquivo 'requirements.txt', pelo comando 'python main.py' no diretório dos arquivos. Apenas a conexão do módulo principal à porta USB é necessária para o Modmixer seja identificado.

O projeto do sistema foi divido entre seus integrantes nas seguintes áreas: interface gráfica, comunicação serial, eletrônica e montagem, modelagem CAD e impressão dos *cases*. A atribuição de funções segue a nomeação no cabeçalho.

II. Requisitos de projeto

O projeto foi subdivido em cinco requisitos funcionais, para que atendesse às expectativas apontadas. E destes, foram especificados os requisitos não funcionais. Os tópicos abaixo os descrevem.

A. Requisitos funcionais

Os requisitos funcionais são:

- RF1: Possuir interface gráfica para configuração dos módulos de hardware.
- RF2: O usuário deve poder controlar o volume de diferentes aplicações do computador a partir dos módulos.
- RF3: Funcionamento *plug and play*.
- RF4: Garantir funcionamento com diferentes quantidades de módulos.
- RF5: Envio de informações dos módulos para o computador através de uma única conexão USB.

B. Requisitos não funcionais

Os requisitos não funcionais foram idealizados para cada item anterior. Eles etão listados a seguir, e seu índice esta vinculado aos requisitos funcionais.

- RNF 1.1: Caixa de seleção de porta usb usada para conexão com módulo principal.
- RNF 1.2: Caixa de seleção para atrelar cada módulo conectado à aplicação desejada, nesta devem estar listadas todas as aplicações gerando saída de som.
- RNF 1.3: O número de caixas de seleção exibidas deve ser o mesmo número de módulos conectados.
- RNF 1.4: Exibir tela de aviso no caso de nenhum módulo esteja conectado.
- RNF 1.5: Exibir barra de preenchimento para indicar volume atual da aplicação.
- RNF 1.6: Botão de término de execução da interface.
- RNF 2.1: A sequência de conexão dos módulos deve corresponder à sequência de índices listados verticalmente na interface gráfica.
- RNF 2.2: Os limites mínimo e máximo do potenciômetro devem ser traduzidos aos níveis mínimo e máximo de áudio da aplicação atrelada.
- RNF 2.3: A seleção da aplicação na interface deve resultar apenas em modificação de volume da aplicação selecionada.
- RNF 3.1: Definição dos endereços do protocolo I2C em tempo de execução via comunicação auxiliar baseada no protocolo One-Wire.
- RNF 3.2: Ao responder a requisição de status via I2C, cada módulo deve sinalizar ao dispositivo principal se há conexão de um módulo depois dele e seu endereço.

- RNF 3.3: O dispositivo principal deve armazenar a quantidade de dispositivos conectados e seus respectivos endereços.
- RNF 4.1: O dispositivo principal deve atualizar a lista de dispositivos conectados sempre que uma requisição não for atendida por duas vezes consecutivas, removendo o dispositivo defeituoso ou desconectado.
- RNF 4.2: O módulo principal deve sinalizar ao programa de interface com o usuário sempre que houver alterações na lista de dispositivos conectados.
- RNF 4.3: O barramento de alimentação 3.3V (VCC e GND) e o barramento I2C (SCL e SDA) devem ter uma caminho direto na conexão entre os módulos e o dispositivo principal, sem depender do processamento dos módulos intermediários.
- RNF 5.: Módulo principal contendo adaptador USB para o microcontrolador ESP01.
- RNF 5.1: Comunicação serial entre o módulo master e o computador via USB com taxa de 9600bps.
- RNF 5.2: Comunicação entre o módulo principal e os demais módulos via I2C.
- RNF 5.3: Módulo principal deve realizar requisições de status (volume) aos módulos secundários periodicamente em um intervalo máximo de 50 milissegundos.
- RNF 5.4: Módulo principal deve enviar ao computador o status de todos os módulos periodicamente em um intervalo máximo de 100 milissegundos.

C. Premissas de desenvolvimento

No projeto, descrito nos dois tópicos anteriores, as seguintes considerações de ambiente e limitações de hardware foram feitas:

- Sistema Operacional Linux (Ubuntu 20.04 LTS) com todas as dependências necessárias instaladas.
- Microcontrolador ESP8266 módulo ESP-01 (para todos os módulos).
- Conexão entre o módulo principal e o computador através de adaptador USB próprio para ESP-01 (ESP01-USB).
- Número máximo de módulos conectados (corrente estimada para cada módulo: 100mA):
  - USB 2.0 (500mA): Módulo principal + 4 Módulos.
  - USB 3.0 (900mA): Módulo principal + 8 Módulos.
- Número mínimo de módulos conectados:
  - USB 2.0 (500mA): Módulo principal.
  - USB 3.0 (900mA): Módulo principal.

III. Desenvolvimento

Para que fosse executado em tempo viável, o ModMixer teve seu desenvolvimento de projeto dividido em quatro áreas entre os integrantes do grupo, a atribuição de funções segue a nomeação no cabeçalho. São elas:

- Interface gráfica.
- Comunicação serial
- Eletrônica e montagem
- Modelagem CAD e impressão dos *cases*

Resultando em vários diagramas, estes apresentados detalhadamente nas oportunidades em sala de aula conforme o cronograma da ementa.

A. Diagramas da Interface gráfica

1) Máquina de estados: Este diagrama ilustra a navegação das telas conforme os módulos são identificados no sistema.

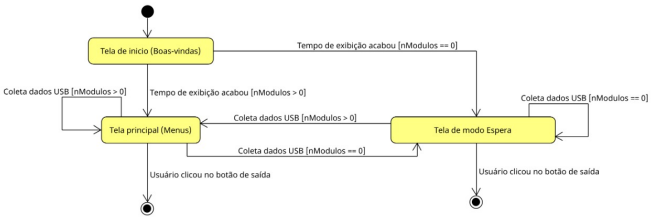

Fig. 1: Máquina de estados das telas

A tela principal de menu, que contém a interação com o usuário, está mostrada na figura abaixo.

Fig. 2: Tela principal

2) Diagrama de atividade: O diagrama de atividades mostra as chamadas de funções da classe principal do Modmixer, e as separa entre cada classe envolvida. Mostrando o fluxo geral da o sistema executado no computador.

3) Diagrama de classes: O problema de interface gráfica foi descrito em seu código Python por quatro classes: Este diagrama descreve os atributos e métodos de cada classe envolvida no problema.

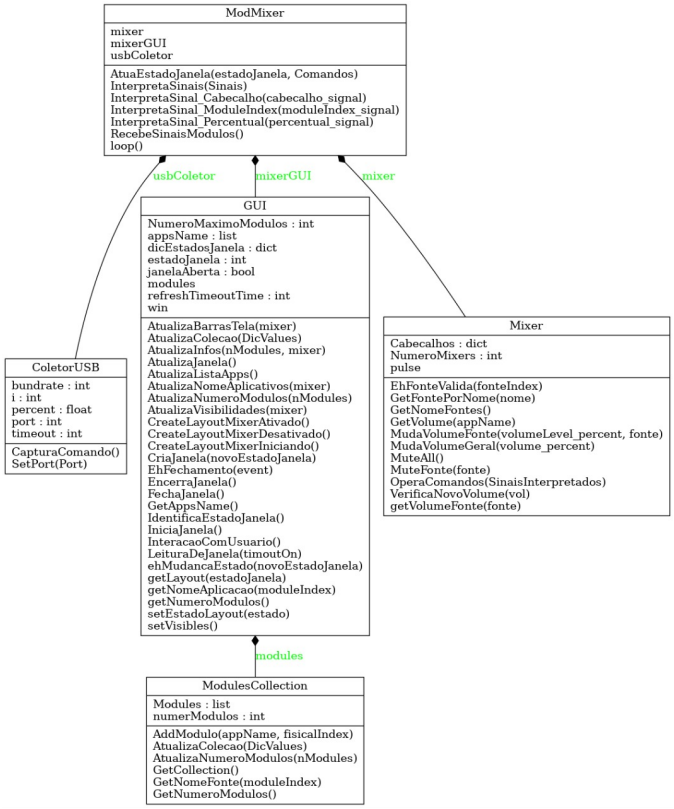

Fig. 4: Diagrama de classes

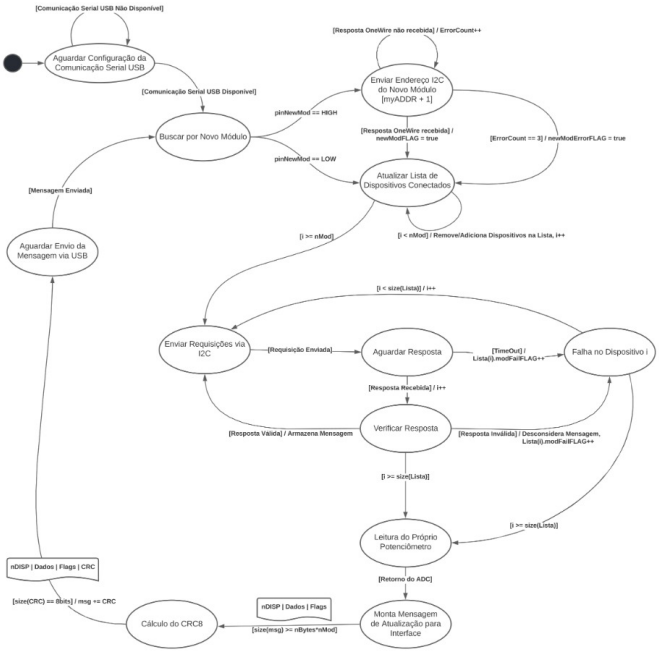

Fig. 5: Máquina de estados do módulo master

B. Diagramas da Comunicação serial

Para a modelagem da comunicação entre o módulo *master* e o computador, e entre módulos secundários; foram feitos dois diagrama de máquina de estados.

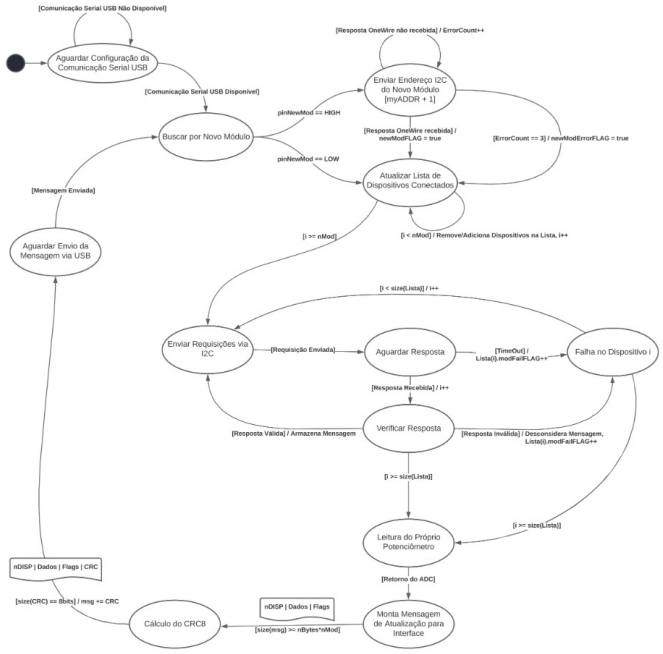

Fig. 6: Máquina de estados dos módulos secundários

IV. Testes

A. Testes da interface gráfica

Os testes da interface gráfica validam alguns dos critérios estabelecidos nos RNF1 e RNF2, porém se mostraram suficientes para o funcionamento do sistema.

Devido aos atrasos de projeto, a interface, após programada, foi testada com auxílio de um Arduino Nano, emulando os sinais de controle dos possíveis módulos.

1) Captura de informações: Neste teste o Arduino escreve uma sequência conhecida de números e letras no formato usado para controle de áudio: ’[’A’ até ‘Z’] [‘0’] [¡0.00 até 1.00]’

As mensagens são capturadas e comparadas com os valores esperados.

- Primeiro teste: compara o número de mensagens recebidas com o número esperado. Assim sabemos que nada foi perdido.
- Segundo teste: compara o conteúdo das mensagens com o esperado Assim sabemos que nossa captura de letras e valores são funcionais e, portanto, podem ser usadas nos sinais de controle.

As rotinas de teste resultaram em cem acertos cada.

2) Detecção de módulos: Mostra que a detecção e adaptação da interface está condizente com a alteração do número de módulos conectados. Mensagens crescentes em tamanho (indicando o aumento do número de módulos conectados) são recebidas numa forma de aumento conhecida. O número de módulos detectados e de itens visíveis no menu são salvos em cada iteração, e então as sequências são comparadas com a esperada.

- Primeiro teste: Verifica o número de módulos detectados
- Segundo teste: Verifica o número de itens visíveis na interface

As rotinas de teste resultaram em 596 acertos cada.

3) Seleção de aplicativos: Valida a listagem e seleção de aplicativos no menu da interface.

- Primeiro teste: Verifica-se que a lista de aplicativos selecionados condiz com a lista de fontes de som no sistema, comparando uma à outra em cada iteração.
- Segundo teste: Um sinal de controle é emitido alterando constantemente o volume de apenas um módulo. Verifica-se que o aplicativo selecionado na interface é o mesmo que tem seu volume alterado no sistema em cada iteração.

As rotinas de teste resultaram em acertos.

4) Teste de Controle de Áudio: O sinal de controle é emitido alterando o volume de duas aplicações, capturado e salvo em cada iteração. Junto a isso, os níveis de áudio do sistema são registrados. Para, então, comparar visualmente que o nível de áudio acompanha o sinal de controle gerado ao longo das iterações.

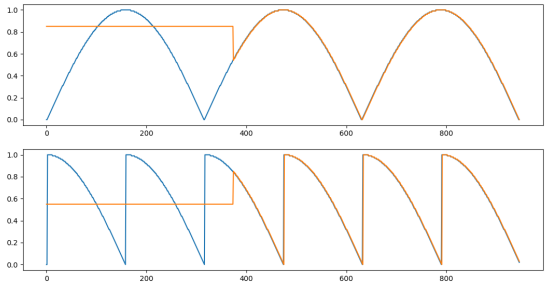

Fig. 7: Dois sinais de áudio seguindo seus sinais de comando

Este teste é validado por inspeção visual. Percebe-se que a curva de áudio (laranja) acompanha a de sinal de controle (azul) após certo instante de tempo, sendo este o momento de confirmação de atribuição do aplicativo na interface.

5) Medição de delay: É usado a biblioteca ‘time’ no registro de *timestamps* para que seja calculado os intervalos de tempo entre cada etapa do código.

- Primeiro teste: É feita a média do tempo entre a criação do objeto Modmixer e a primeira exibição do menu de seleção de 30 execuções da interface.
- Segundo teste: É feita a média geral do tempo registrado entre a captura do sinal de comando e a chamada de

função da biblioteca de alteração de volume em cada iteração de 30 execuções da interface.

As rotinas de teste resultaram na medição de 17.73s para o aparecimento do menu de seleção e 0.02s entre o sinal de controle e a atuação no sistema de som.

B. Testes da comunicação serial

1) Detecção de Módulos e Definição de novos Endereços: Ao iniciar a execução do módulo principal é enviado uma mensagem ao endereço padrão I2C (definido como 0x01), no qual todos os módulos secundários iniciam. Contudo, cada módulo secundário aguarda o módulo anterior ser configurado para ingressar no barramento I2C. Essa ordem de acesso ao barramento é garantida através de uma sinalização digital em nível alto, realizada em cascata entre os módulos.

Ao receber a mensagem no endereço padrão, os módulos secundários conferem o campo da soma de verificação e, caso não haja discrepâncias entre o valor recebido e a soma calculada internamente, passam para verificação do conteúdo da mensagem. Se no byte de flags for detectada a presença da flag para setar um novo endereço, esta operação será feita em sequência, redefinindo seu endereço no barramento I2C e liberando o próximo módulo a iniciar com o endereço padrão.

O resultado obtido é mostrado na figura abaixo e demonstra essa operação de comunicação através do endereço padrão e redefinição de endereço ocorrendo com 4 dispositivos. Este teste foi realizado no momento de setup do módulo principal, onde são descobertos quantos módulos já estão conectados inicialmente no barramento.

```
Dispositivos Configurados:  1 (ADDR = 0x0A)
Dispositivos Configurados:  2 (ADDR = 0x0B)
Dispositivos Configurados:  3 (ADDR = 0x0C)
Dispositivos Configurados:  4 (ADDR = 0x0D)

-----------------------------------------
 4 Dispositivos Configurados com Sucesso
-----------------------------------------
```

Fig. 8

2) Teste de Integridade das Mensagens: Neste teste foi realizada a comunicação com os dispositivos já configurados com objetivo de verificar a integridade das mensagens e detecção de erros. A detecção de erros nas mensagens pode ocorrer de duas formas: o número de bytes recebidos é diferente do número de bytes solicitados ou o byte contendo a soma de verificação contém um valor diferente do calculado internamente com base na mensagem recebida.

```
DISP:  0, err:  0
ADDR: 0x0A
size:  3 / 3, MATCH
data: MATCH
DISP_00[0]: 255
DISP_00[1]: 00
DISP_00[2]: ff
```

```
DISP:  1, err:  0
ADDR: 0x0B
size:  3 / 3, MATCH
data: MATCH
DISP_01[0]: 12
DISP_01[1]: 00
DISP_01[2]: 0c
```

```
DISP:  2, err:  0
ADDR: 0x0C
size:  3 / 3, MATCH
data: MATCH
DISP_02[0]: 36
DISP_02[1]: 00
DISP_02[2]: 24
```

```
DISP:  3, err:  0
ADDR: 0x0D
size:  3 / 3, MATCH
data: MATCH
DISP_03[0]: 40
DISP_03[1]: 00
DISP_03[2]: 28
```

Fig. 9

A figura acima apresenta o número do dispositivo (”DISP”), o erro de comunicação acumulado (”err”), o endereço do dispositivo (”ADDR”), o número de bytes solicitado e recebido (”size: solicitado / recebido”), a verificação do número de bytes e da soma de verificação, e, por fim, o conteúdo de cada um dos 3 bytes da mensagem (de acordo com o dataframe especificado na Figura (FIGS)). Ao desconectar um módulo, nenhum byte é recebido e a divergência entre requisição e resposta é detectada. Este caso é apresentado na Figura (TAL). Com isso, podemos ver o conteúdo e tamanho das mensagens recebidas e verificar a coerência com a detecção de erros na comunicação.

3) Teste de Variação do Volume: Por fim, foram plotados os gráficos do primeiro byte de cada mensagem, que corresponde ao volume do módulo.

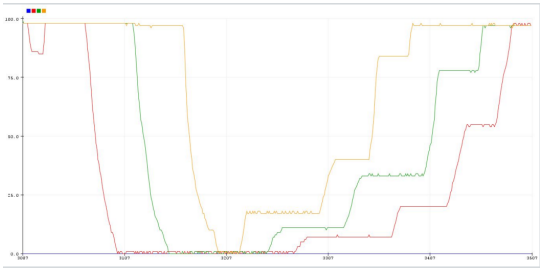

Fig. 10

V. Problemas encontrados

Alguns problemas no decorrer da execução do projeto levaram ao atraso das atividades relacionados à interface gráfica e comunicação serial e valem ser pontuados.

A biblioteca Pysimplegui usada montagem da interface gráfica apresentou algumas falhas imprevistas nos exemplos de documentação. O evento de fechamento da interface foi alvo de recentes mudanças pelos desenvolvedores. Este evento, geralmente referido como único, agora é tradado como dois momentos: intenção de fechar e fechamento completo.

Outro problema problema encontrado foi no uso da função ’readline’ da biblioteca Pyserial, usada para coleta de comandos vindo dos módulos pela USB. Esta função inicialmente deveria ser monitorada pelo *timeout* definido, porém nem sempre isto ocorre. A substituição pela função de utilidade similar ’read_until’ foi suficiente para não apresentar falhas novamente. Além disso, o tempo de *delay* entre operações de escrita e leitura não foi achado na documentação, e levou à falhas.

VI. Conclusão

O projeto proposto foi executado conforme descrito neste trabalho, demonstrando sua executabilidade dentro da demanda esperada. O aumento do tempo dedicado para o processo de testagem mostrou-se necessário.

Melhorias no sistema de comunicação e interface são possíveis, porém não necessárias para sua execução.

São sugeridos aprimoramentos em seu código de coleta de dados, como o uso de *Thread* dedicada para leitura da

porta USB. Além disso, é desejável refatorar o código embarcado de comunicação entre dispositivos para a diminuição de sua complexidade. Outra tarefa para trabalhos futuros é disponibilização, em forma de biblioteca, dos métodos necessários para as operações.