Grupo 2: Modmixer

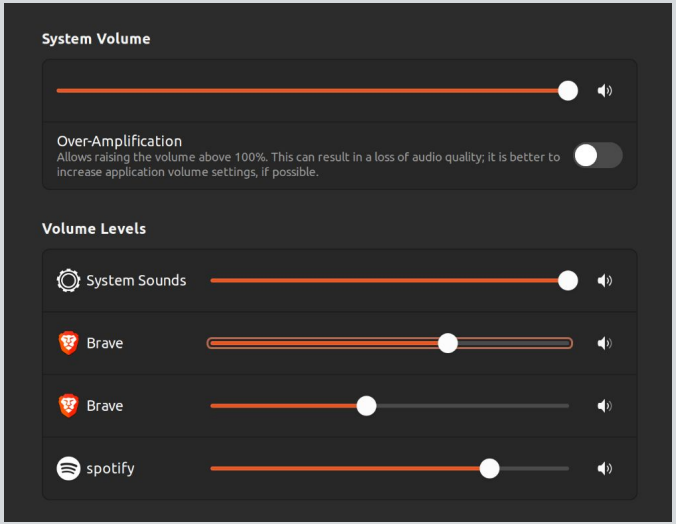
Assis G. , Arthur H., Gabriel C., Gabriel M. e Guilherme T.

Engenharia Mecatrônica | CTJ

UFSC
Joinville

Problemática apresentada

- Controle de volume em múltiplas aplicações
- Especialmente no caso de uso em desktops



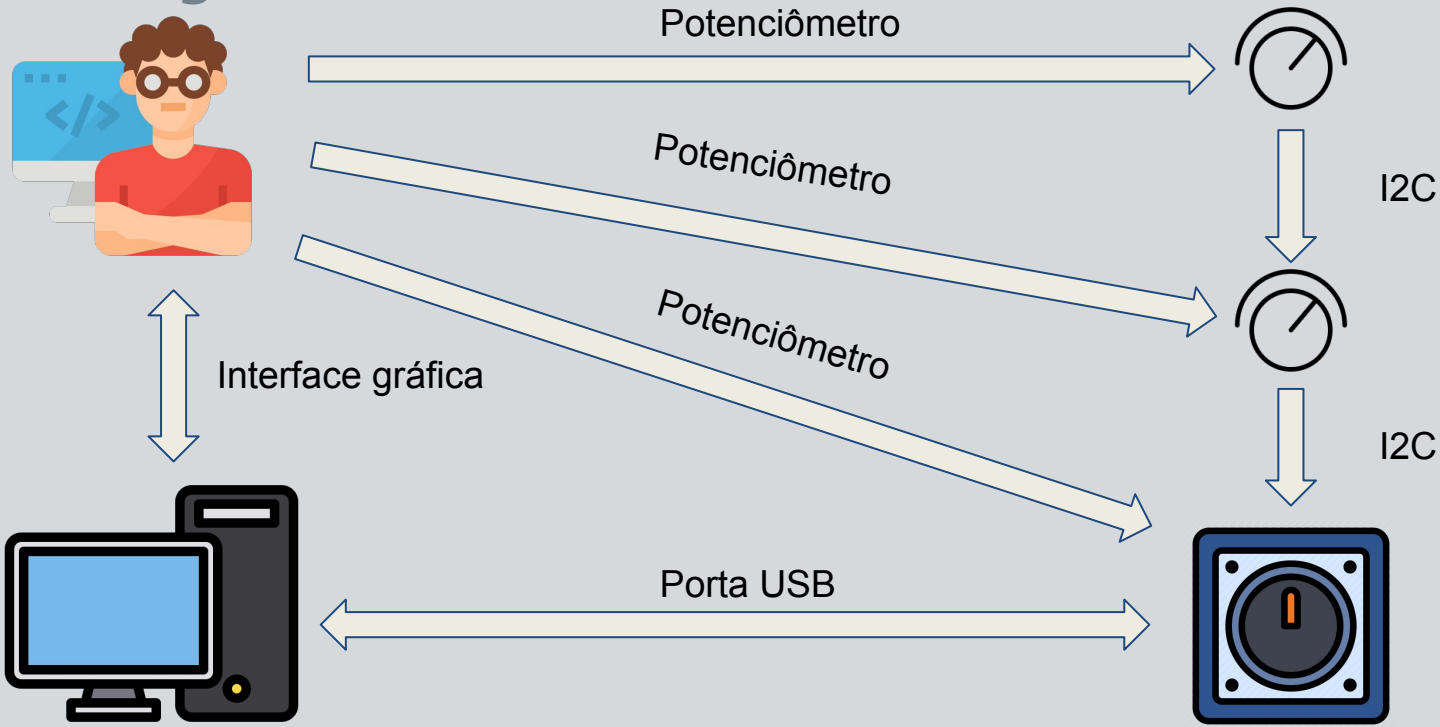
Solução esperada

- **Atrelar o volume de alguma aplicação à um controle físico em hardware**
- **Possibilitar extensão modular**

Público alvo

- Prossumidores
- Entusiastas de tecnologia da informação
- Criadores de conteúdo digital

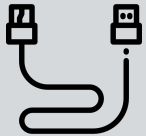
Projeto base



Divisão do problema

- **Interface gráfica: Gabriel César Silveira**

- **Comunicação serial: Guilherme Turatto**



- **Montagem eletrônica: Assis G.**

- **Impressão aditiva: Gabriel Luiz Martins e Arthur Henrique Mallmann**

Requisitos Funcionais

- **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;

- **RF1: Possuir interface gráfica para configuração dos módulos de hardware:**
  - O usuário não deve precisar selecionar a porta USB
  - Caixa de seleção para atrelar cada módulo conectado à aplicação desejada, nesta devem estar listadas todas as aplicações gerando saída de som;
  - O número de caixas de seleção exibidas deve ser o mesmo número de módulos conectados;
  - Exibir tela de aviso no caso de nenhum módulo esteja conectado;
  - Exibir barra de preenchimento para indicar volume atual da aplicação;
  - Botão de término de execução da interface;

- **RF2: O usuário deve poder controlar o volume de diferentes aplicações do computador a partir dos módulos:**
  - A sequência de conexão dos módulos deve corresponder à sequência de índices listados verticalmente na interface gráfica;
  - Os limites mínimo e máximo do potenciômetro devem ser traduzidos aos níveis mínimo e máximo de áudio da aplicação atrelada;
  - A seleção da aplicação na interface deve resultar apenas em modificação de volume da aplicação selecionada;

Requisitos Não Funcionais

- **RF3: Funcionamento plug and play:**
  - Alimentação de todos os módulos provida pela conexão USB;
  - Conexão entre os módulos através do conector DB9;
  - Cada módulo deve conter um conector DB9 macho (para se conectar ao dispositivo já existente) e outro fêmea (para possibilitar a conexão de novos módulos);
  - O módulo principal deve possuir uma conexão USB macho com cabo (para conexão com o computador) e uma conexão DB9 fêmea (para conexão com o primeiro módulo);

Requisitos Não Funcionais

- **RF4: Funcionamento com diferentes quantidades de módulos:**
  - O dispositivo principal deve atualizar a lista de dispositivos conectados sempre que uma requisição não for atendida por três vezes consecutivas, removendo o dispositivo defeituoso ou desconectado (*modFailFLAG*);
  - O módulo principal deve sinalizar ao programa de interface com o usuário sempre que houver alterações na lista de dispositivos conectados (*newModSuccessFLAG*);
  - 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;

Requisitos Não Funcionais

- **RF5: Envio de informações dos módulos para o computador através de uma única conexão USB:**
  - Módulo principal contendo adaptador USB para o microcontrolador ESP01;
  - Comunicação serial entre o módulo master e o computador via USB com taxa de 9600bps;
  - Comunicação entre o módulo principal e os demais módulos via I2C;
  - Módulo principal deve realizar requisições de status (volume) aos módulos secundários periodicamente em um intervalo máximo de 50 milissegundos;
  - Módulo principal deve enviar ao computador o status de

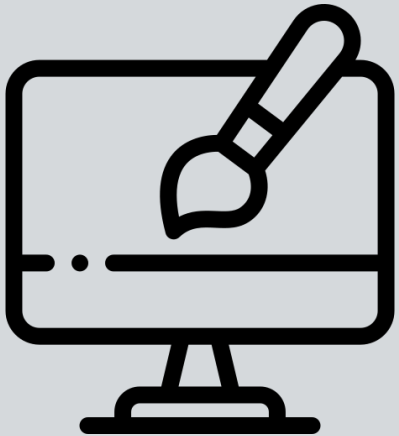
Considerações

- Sistema Operacional Linux (Ubuntu 20.04 LTS);
- Microcontrolador ESP8266 módulo NodeMCU (para todos os módulos);
- Conexão entre o módulo principal e o computador através da porta micro USB integrada ao modulo;
- 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;

Projeto da Interface Gráfica

Integrante responsável: Gabriel César Silveira

RF1 e RF2



Máquina de estados das telas

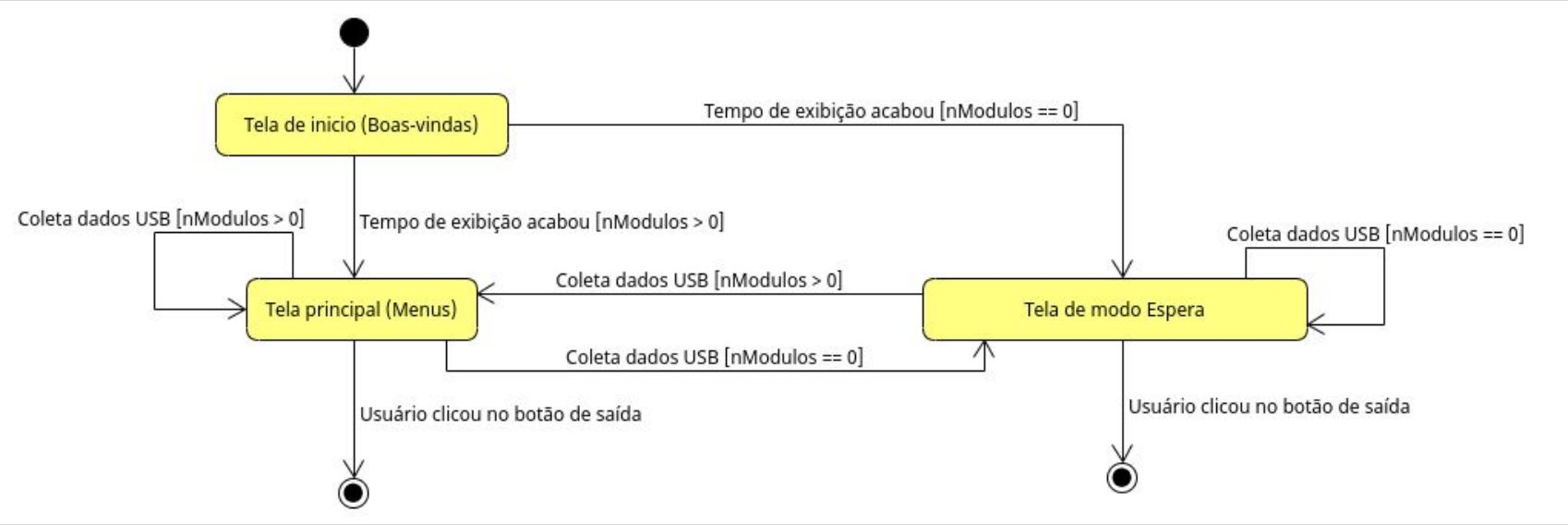
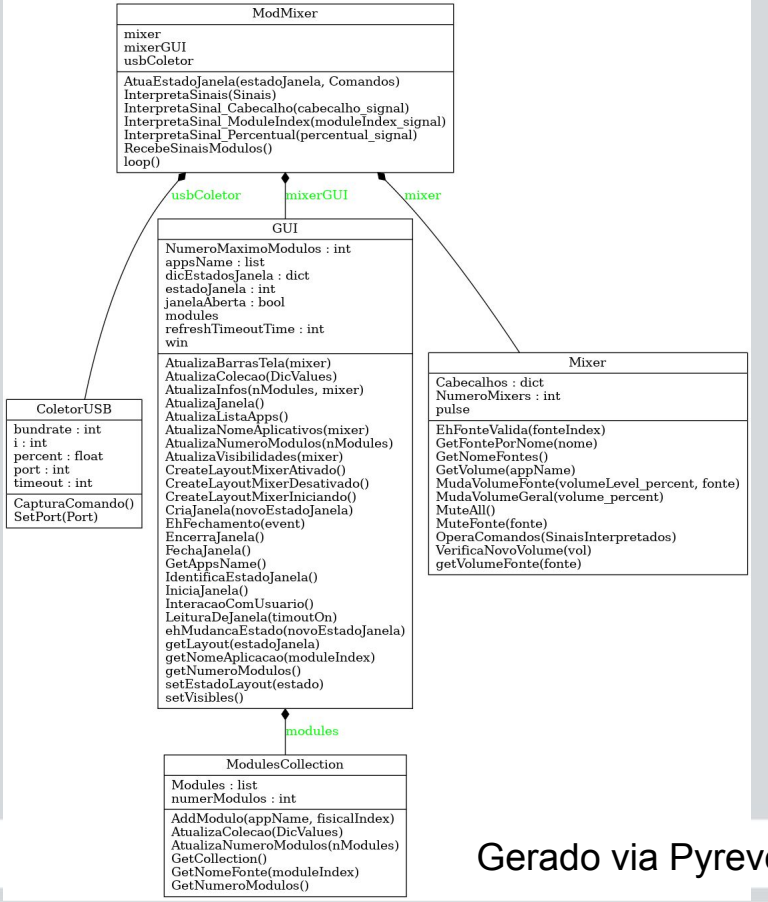


Diagrama de classes



Gerado via Pyreverse

UFSC

Diagrama de atividades

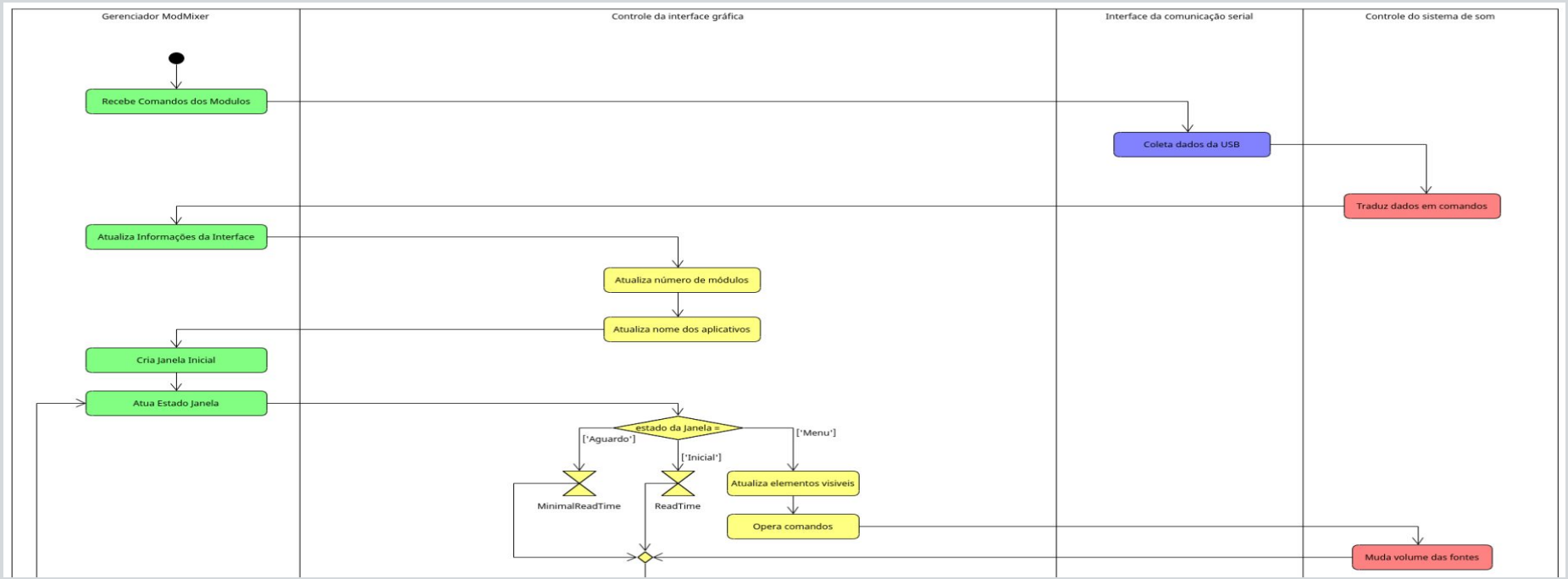
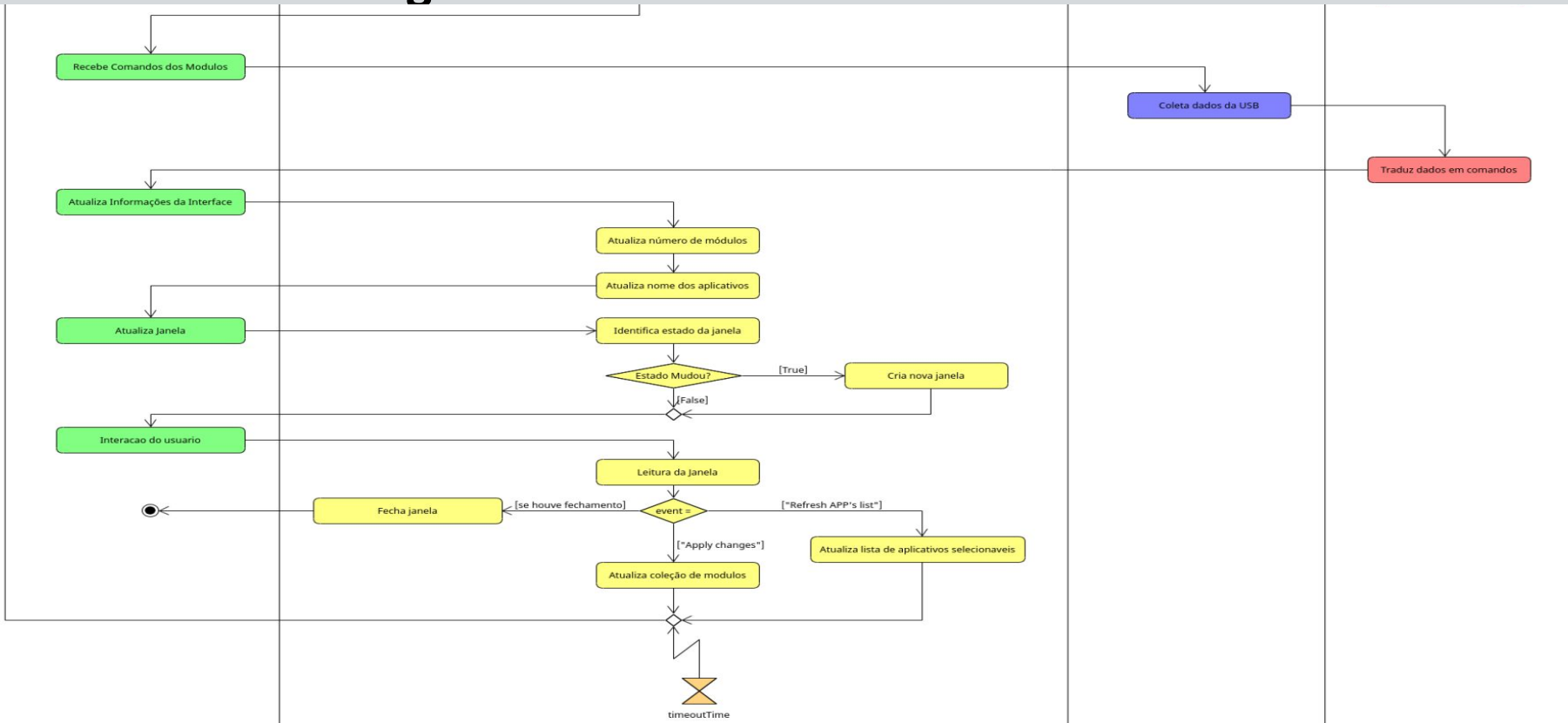


Diagrama de atividades



Testes Unitários

- Captura de informações
- Detecção de módulos
- Seleção de aplicativos
- Controle de áudio
- Medição de delays

Todos foram efetuados emulando diferentes entradas USB com um Arduino Nano.

Teste de Captura de Informações

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.

Resultados:

```
linux@gabrielvoid:~/Área de Trabalho/Integrador 2/Implementacoes/Implementacao e Teste/Testes$ python main_TesteComandos.py
Teste de numero de itens identificados:  100 / 100
Teste de identificacao correta:  100 / 100
```

Teste de Detecção de Módulos

Mostramos 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.
Primeira teste: Verifica o número de módulos detectados
Segundo teste: Verifica o número de itens visíveis na interface
Resultados:

```
linux@gabrielvoid:~/Área de Trabalho/Integrador 2/Implementacoes/Implementacao e Teste/Testes$ python main_TesteDeteccao.py
Teste de deteccao, status de sucesso:  True
Teste de visibilidades, status de sucesso:  True
```

Teste de Seleção de Aplicativo

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.

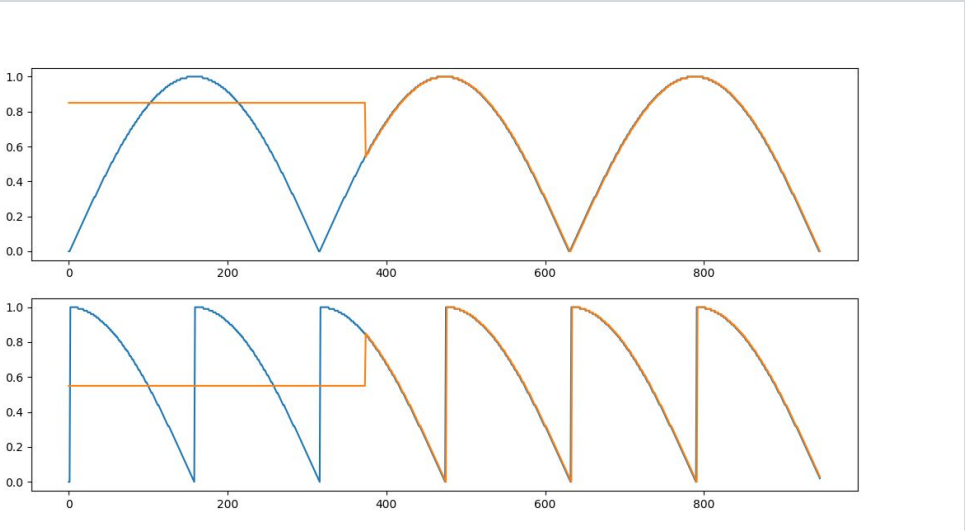
Resultados:

```
linux@gabrielvoid:~/Área de Trabalho/Integrador 2/Implementacoes/Implementacao e Teste/Testes$ python main_TesteSelecao.py
Teste de englobamento:  596 / 596
Teste de selecao de aplicativo:  596 / 596
```

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.

Resultado:



Medição de delays

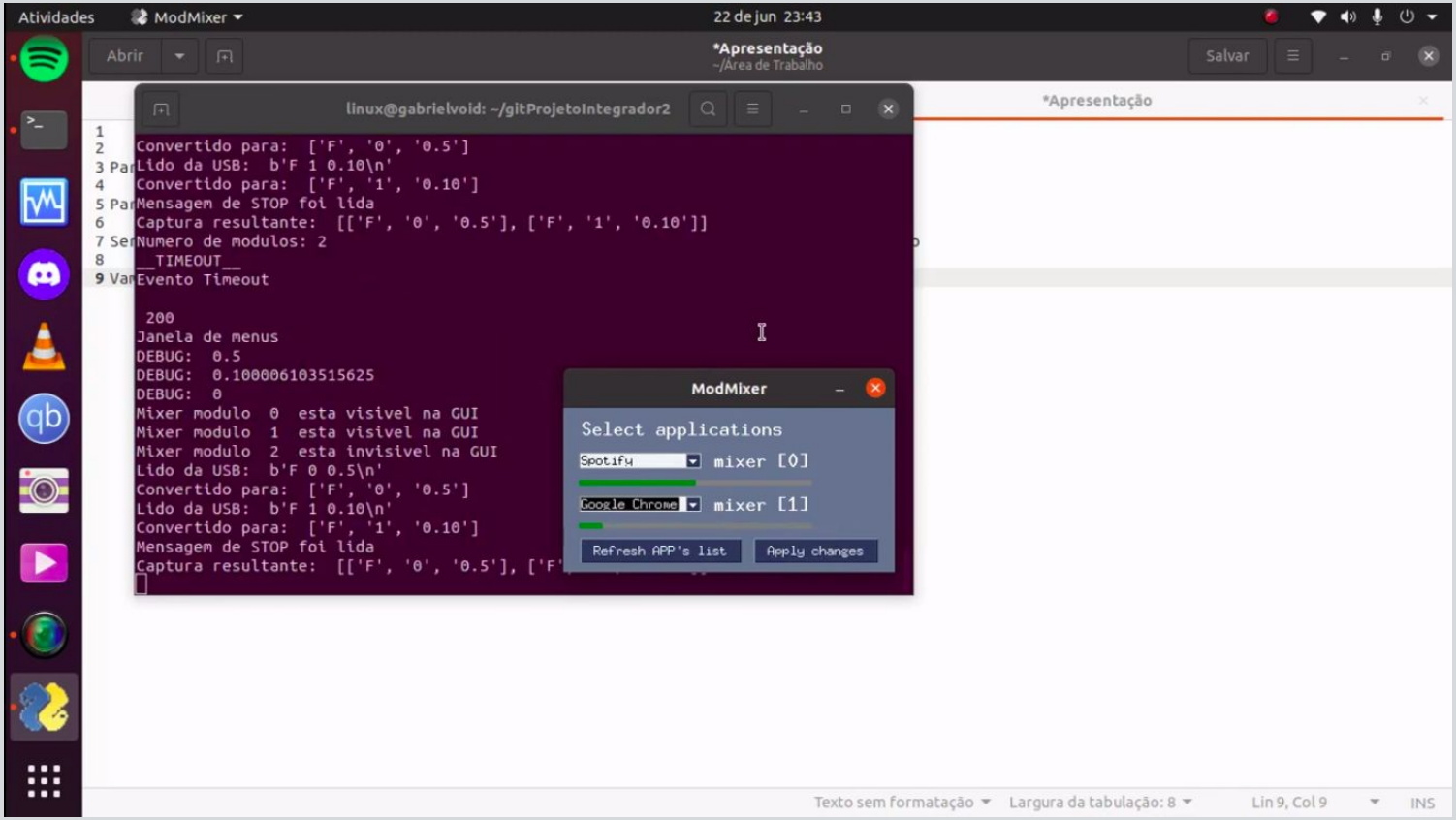
É usado a biblioteca ‘time’ no registro de *timestamps* para que seja calculado os intervalos de tempo entre cada etapa do código.

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.

Resultados (em segundos):

```
linux@gabrielvoid:~/Área de Trabalho/Integrador 2/Implementacoes/Implementacao e Teste/Testes$ python main_TesteDelays.py
Tempo entre a criacao do objeto e a primeira leitura de janela:  17.7287246465683
Tempo medio geral entre um sinal de mudanca de volume e sua atuacao(chamada PulseAudio):  0.02086561706626187
```

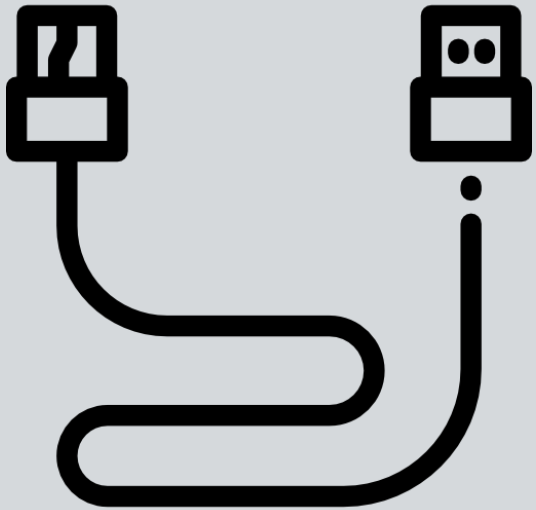

Atividades
ModMixer
22 de jun 23:43
Abrir
*Apresentação
~/Área de Trabalho
Salvar
*Apresentação
linux@gabrielvoid: ~/gitProjetoIntegrador2
Convertido para: ['F', '0', '0.5']
Lido da USB: b'F 1 0.10\n'
Convertido para: ['F', '1', '0.10']
Mensagem de STOP foi lida
Captura resultante: [['F', '0', '0.5'], ['F', '1', '0.10']]
Numero de modulos: 2
__TIMEOUT__
Evento Timeout
200
Janela de menus
DEBUG: 0.5
DEBUG: 0.100006103515625
DEBUG: 0
Mixer modulo 0 esta visivel na GUI
Mixer modulo 1 esta visivel na GUI
Mixer modulo 2 esta invisivel na GUI
Lido da USB: b'F 0 0.5\n'
Convertido para: ['F', '0', '0.5']
Lido da USB: b'F 1 0.10\n'
Convertido para: ['F', '1', '0.10']
Mensagem de STOP foi lida
Captura resultante: [['F', '0', '0.5'], ['F'
ModMixer
Select applications
Spotify
mixer [0]
Google Chrome
mixer [1]
Refresh APP's list
Apply changes
Texto sem formatação
Largura da tabulação: 8
Lin 9, Col 9
INS


ModMixer
Select applications
Spotify
mixer [0]
Google Chrome
mixer [1]
Refresh APP's list
Apply changes

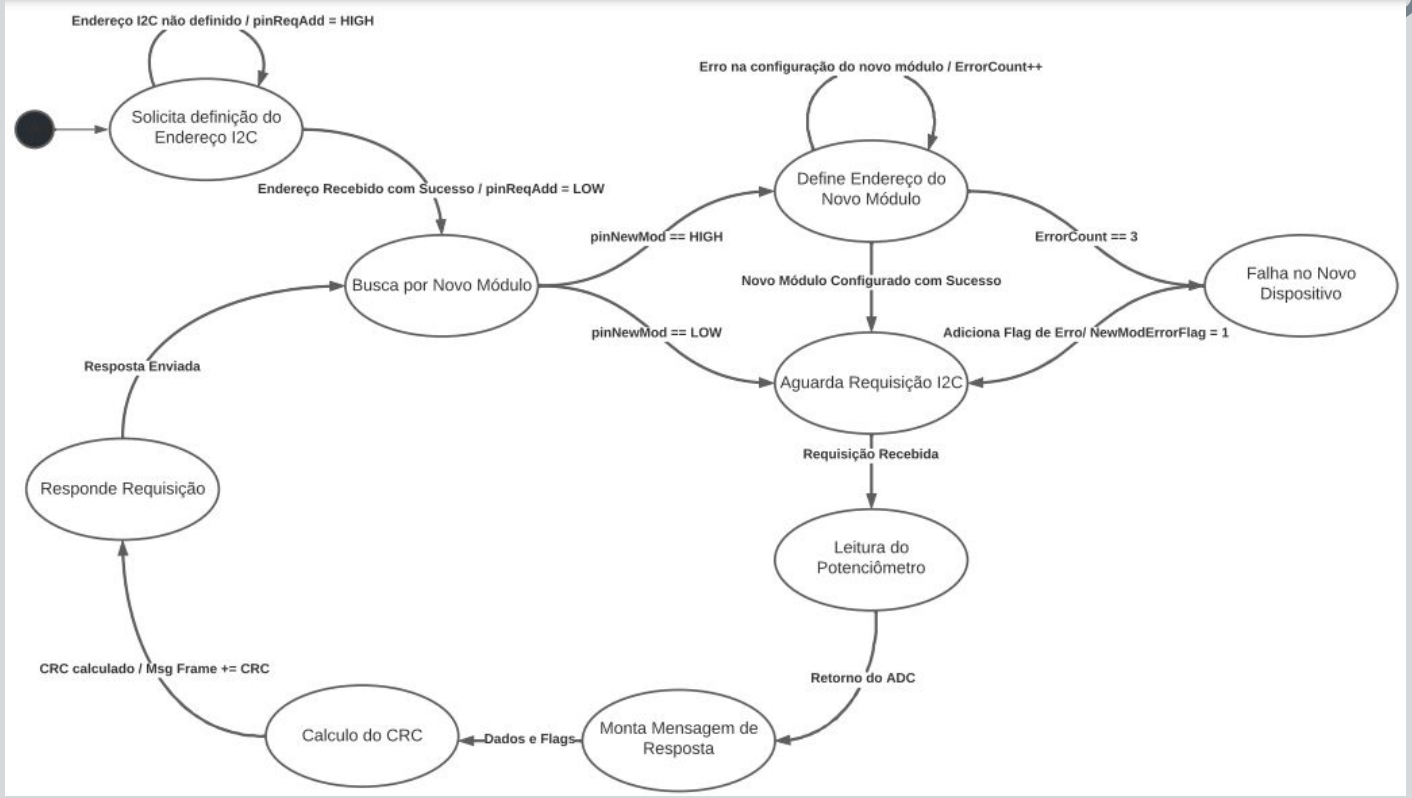
Projeto da Comunicação

Integrante responsável: Guilherme Turatto

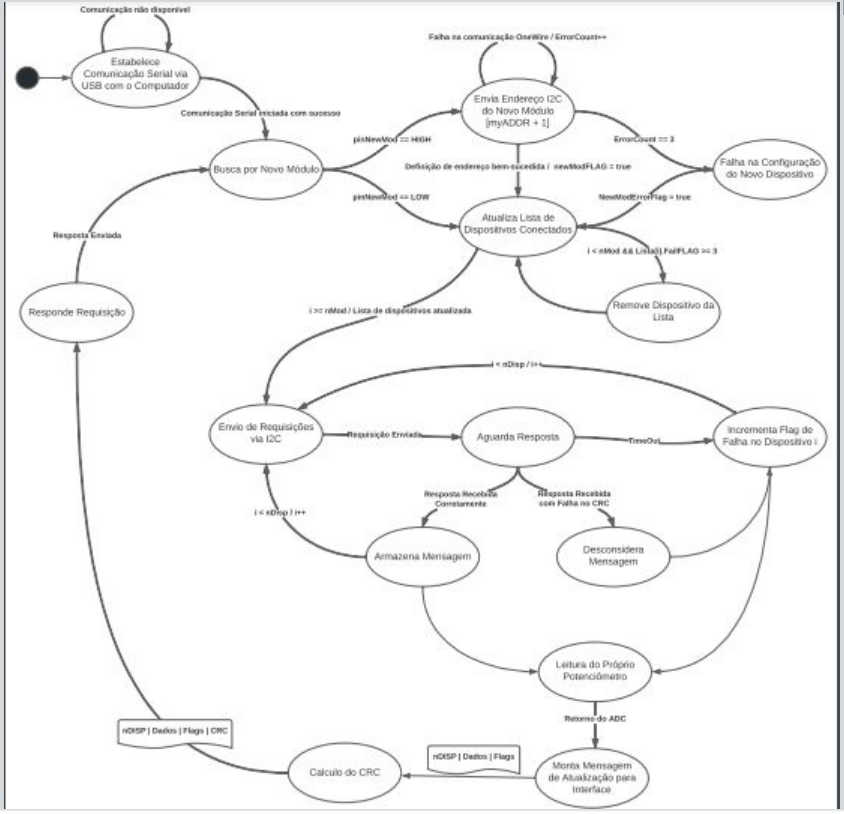
RF3, RF4 e RF5.



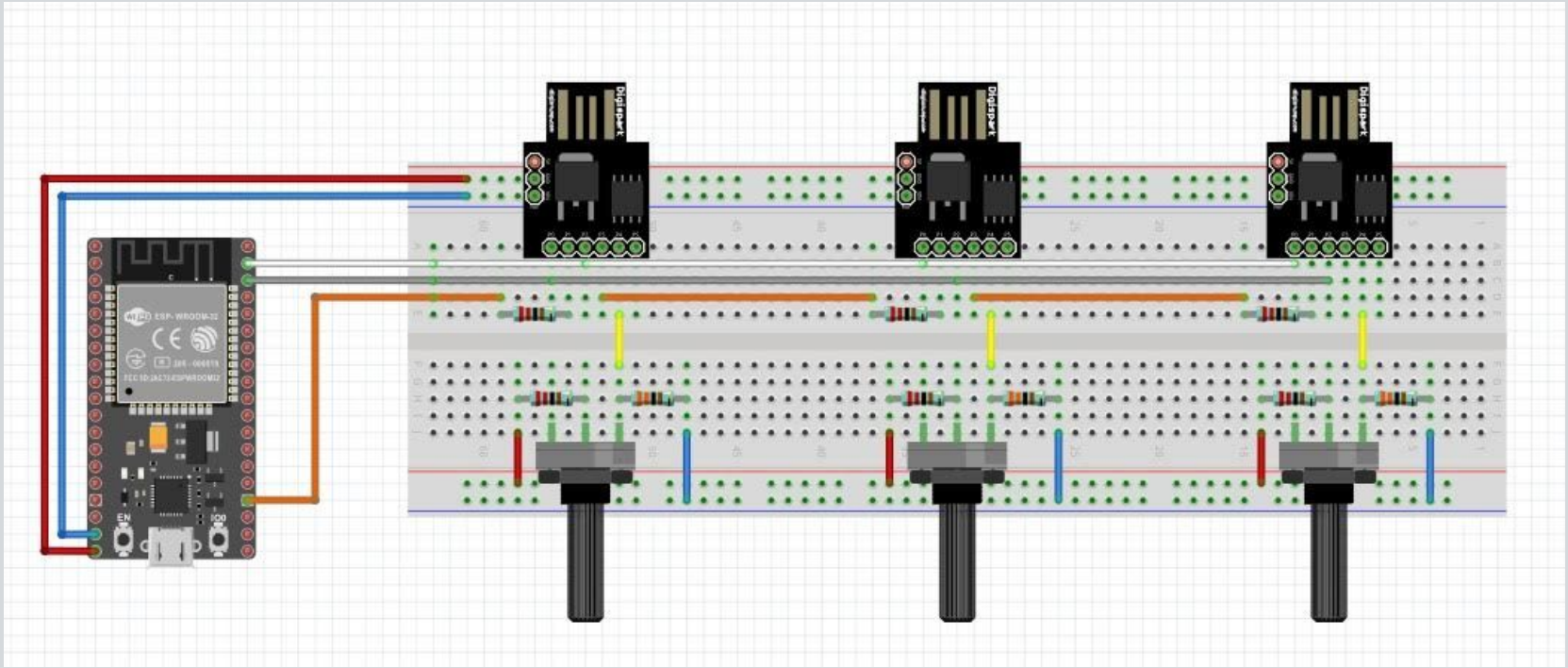
FSM - Módulos Secundários

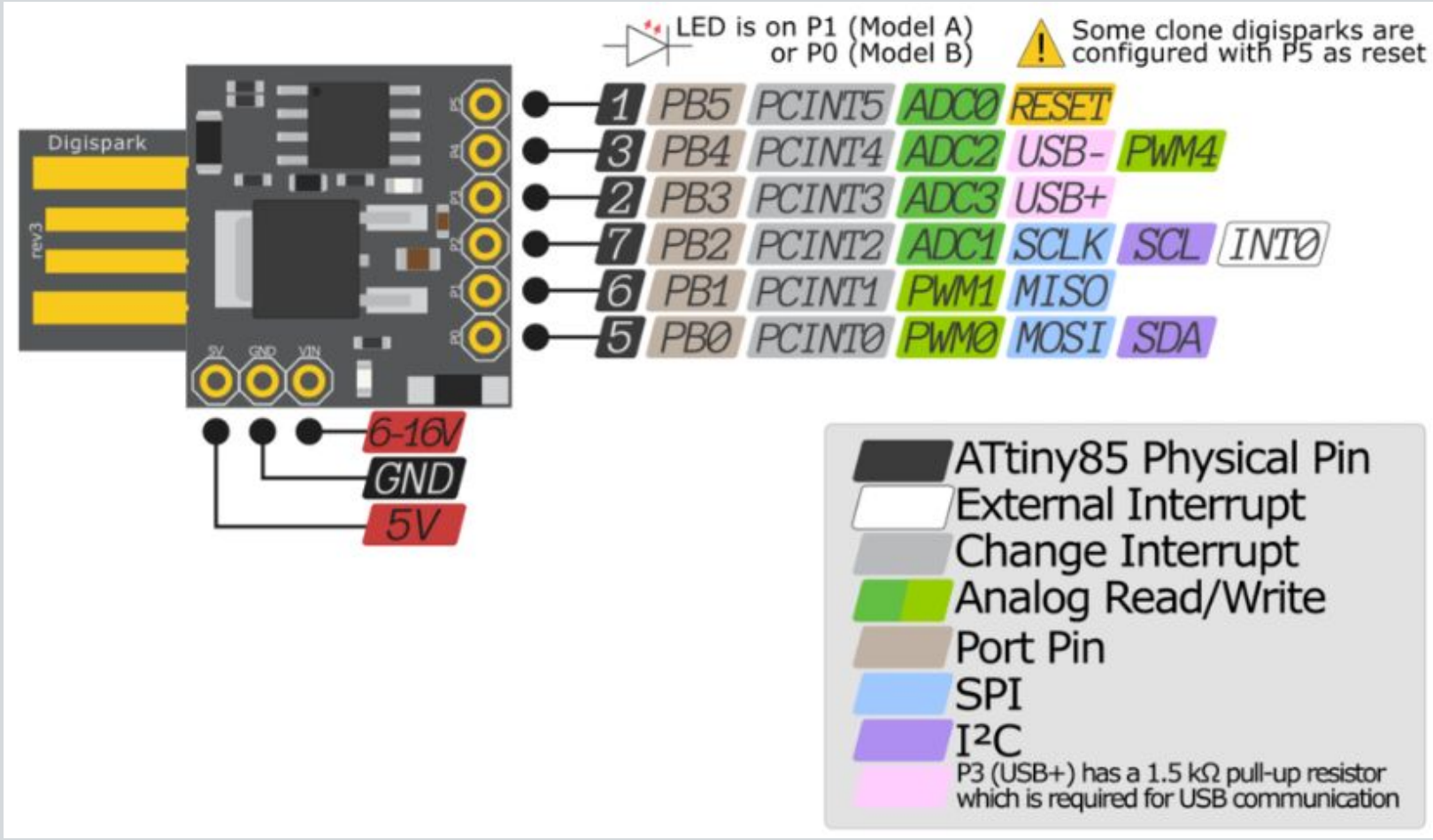


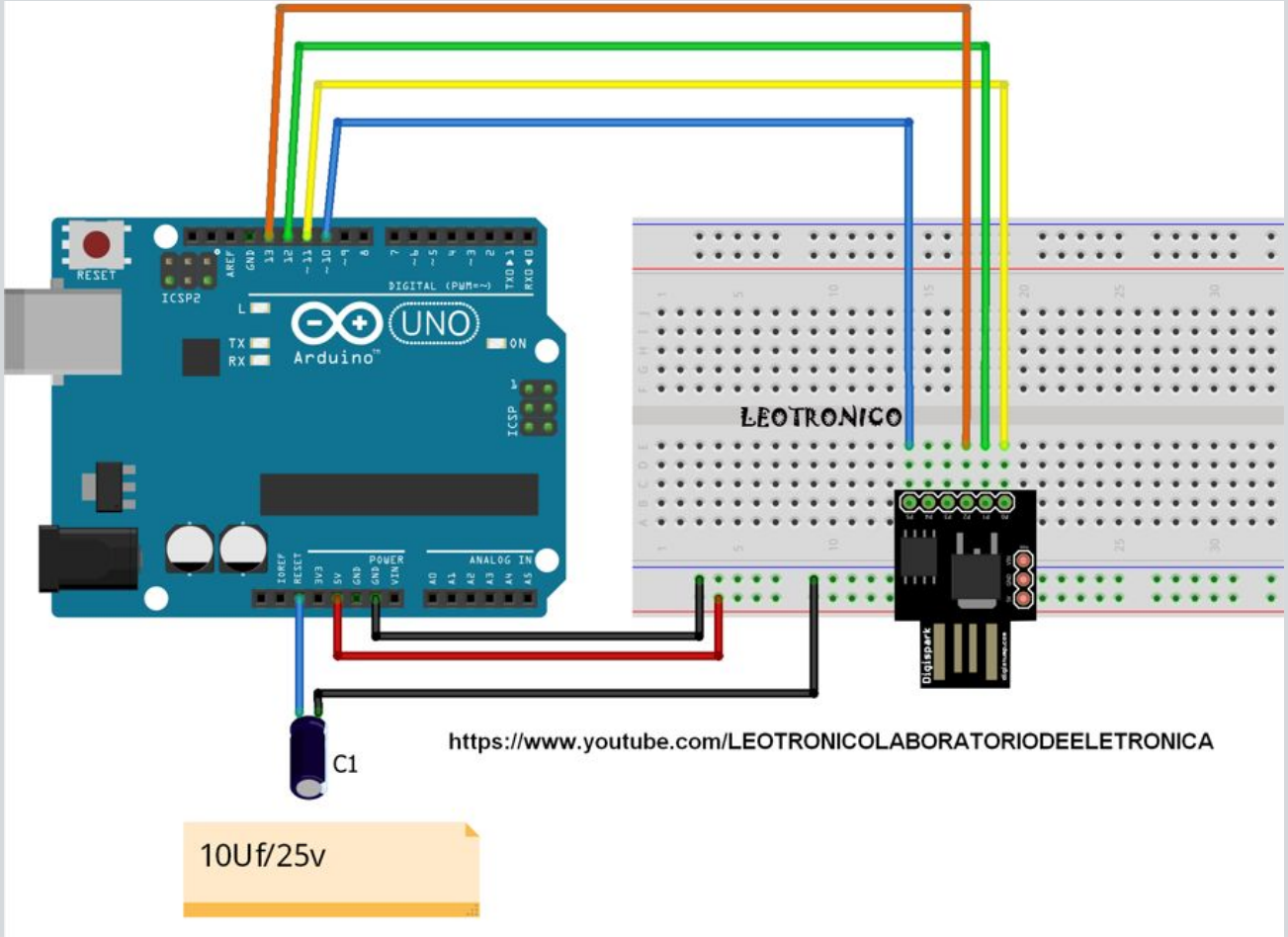
FSM - Módulo Principal



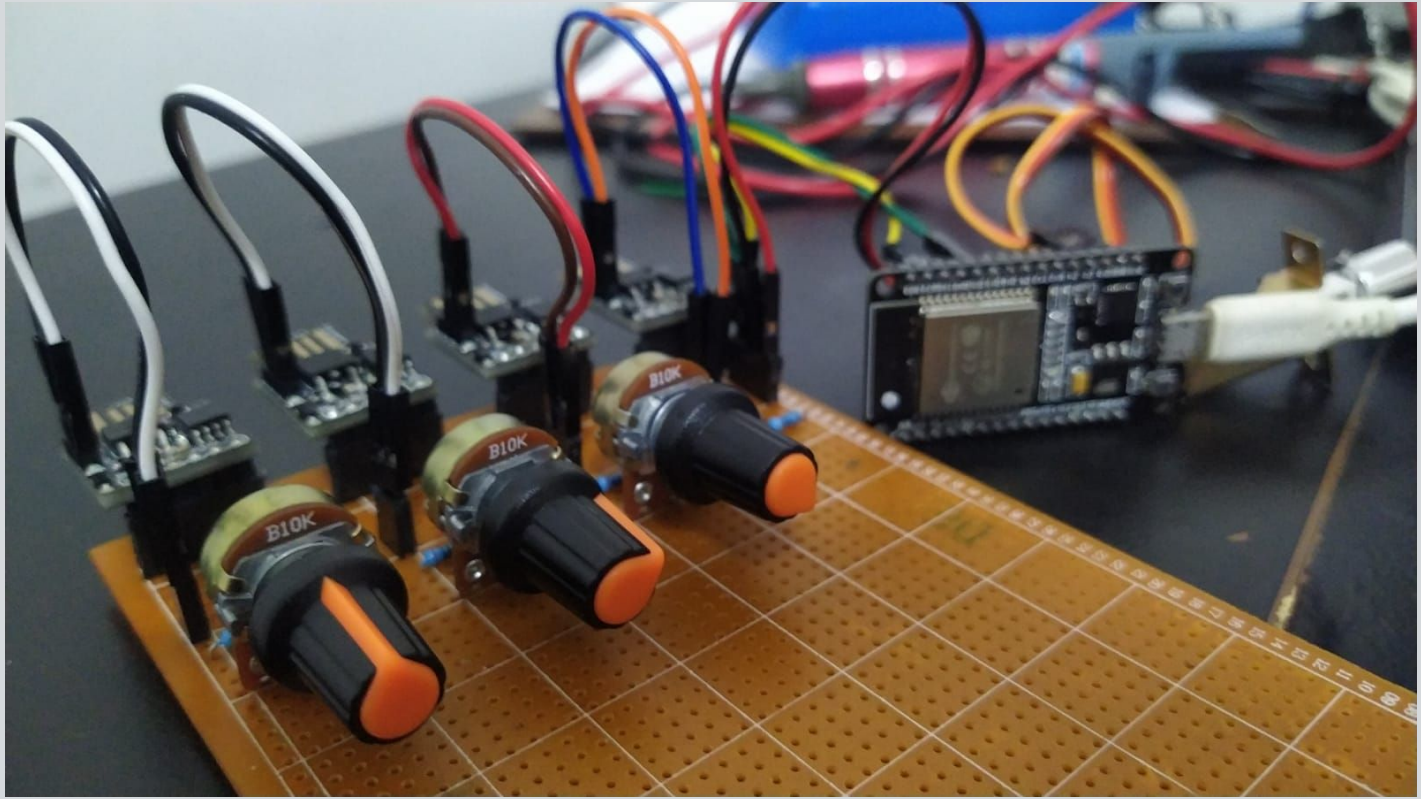
Circuito de Testes



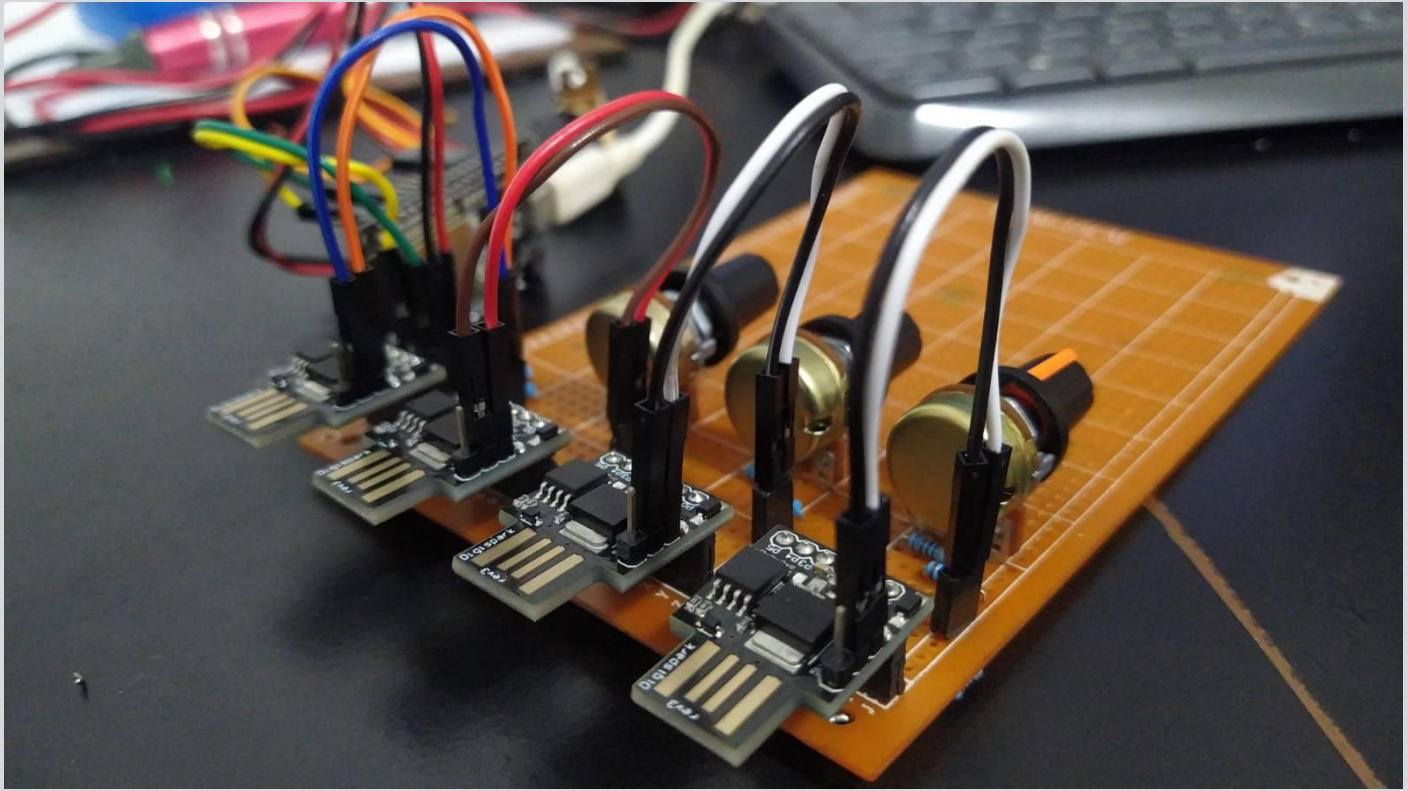

LED is on P1 (Model A)
or P0 (Model B)
Some clone digisparks are
configured with P5 as reset
Digispark
rev3
5V
GND
VIN
1 PB5 PCINT5 ADC0 RESET
3 PB4 PCINT4 ADC2 USB- PWM4
2 PB3 PCINT3 ADC3 USB+
7 PB2 PCINT2 ADC1 SCLK SCL INT0
6 PB1 PCINT1 PWM1 MISO
5 PB0 PCINT0 PWM0 MOSI SDA
6-16V
GND
5V
ATtiny85 Physical Pin
External Interrupt
Change Interrupt
Analog Read/Write
Port Pin
SPI
I²C
P3 (USB+) has a 1.5 kΩ pull-up resistor
which is required for USB communication


RESET
ICSP2
UNO
Arduino
TX
RX
ON
ICSP
DIGITAL (PWM~)
POWER
ANALOG IN
LEOTRONICO
C1
https://www.youtube.com/LEOTRONICOLABORATORIODEELETRONICA
10Uf/25v

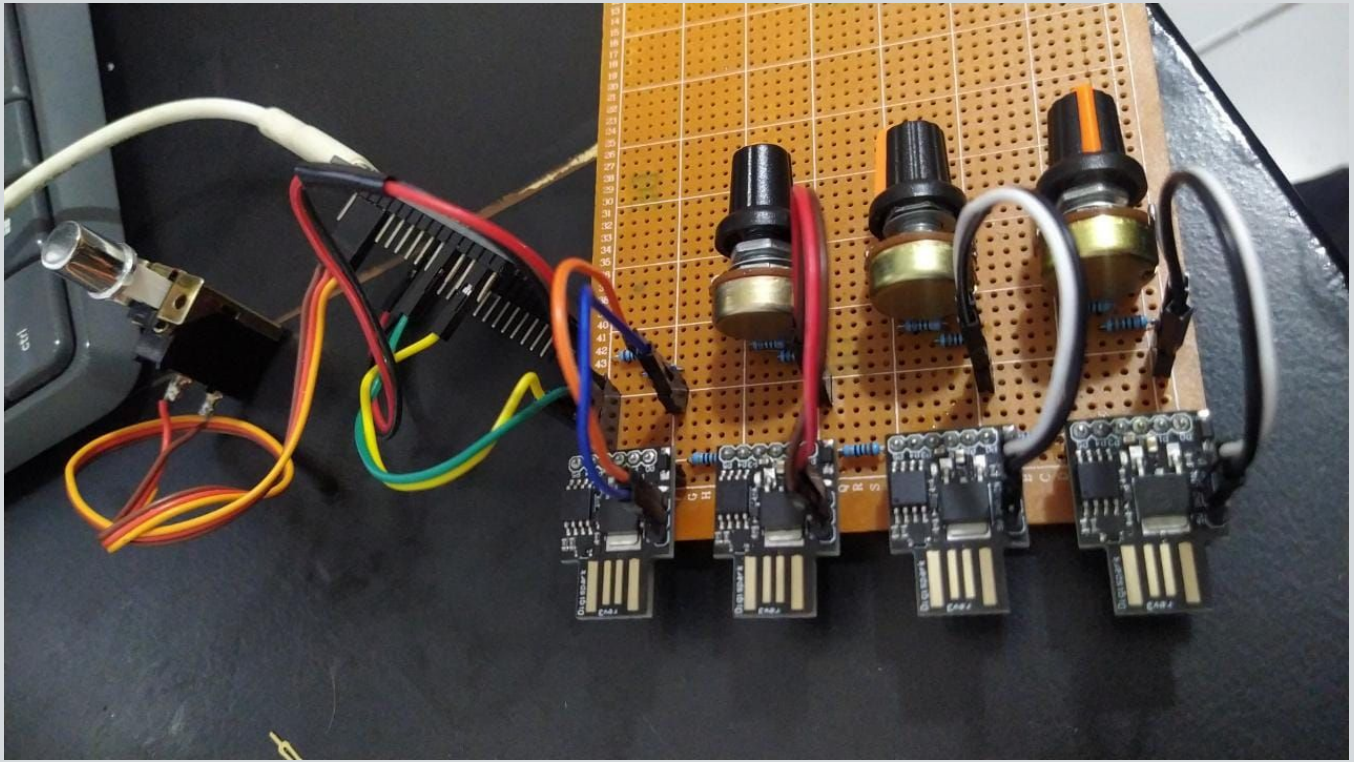
Circuito de Testes



Circuito de Testes



Circuito de Testes



Estrutura de Mensagens

Mensagem de Requisição I2C			
Nome	**ID**	**FLAGS**	**CHECKSUM**
Tamanho	1byte	1byte	1byte
Faixa de Valores	0 - 127	0-255	0-255
Função	Próprio endereço I2C ou novo endereço	Indicam funções a serem executadas pelo módulo	Soma de verificação

Mensagem de Resposta I2C			
Nome	**DADOS**	**FLAGS**	**CHECKSUM**
Tamanho	1byte	1byte	1byte
Faixa de Valores	0 - 100	0-255	0-255
Função	Leitura analógica já convertida	Indicam parâmetros de funcionamento	Soma de verificação

Mensagem de Requisição

Mensagem de Requisição I2C			
Nome	**ID**	**FLAGS**	**CHECKSUM**
Tamanho	1byte	1byte	1byte
Faixa de Valores	0 - 127	0-255	0-255
Função	Próprio endereço I2C ou novo endereço	Indicam funções a serem executadas pelo módulo	Soma de verificação

Flags de Requisição

Flags de Requisição I2C		
Bits	**FLAG**	**Função**
7	setAddrFLAG	Indica que o módulo deve atualizar seu endereço para o endereço recebido em ID
6	nextModResetFLAG	Indica que o módulo subsequente deve ser resetado
5		
4		
3		
2		
1		
0		

Mensagem de Resposta

Mensagem de Resposta I2C			
Nome	**DADOS**	**FLAGS**	**CHECKSUM**
Tamanho	1byte	1byte	1byte
Faixa de Valores	0 - 100	0-255	0-255
Função	Leitura analógica já convertida	Indicam parâmetros de funcionamento	Soma de verificação

Flags de Resposta

Flags de Resposta I2C		
Bits	**FLAG**	**Função**
7	internalFailFLAG	Indica que o módulo está com alguma falha
6	newModResetedFLAG	Indica que o módulo subsequente foi resetado
5		
4		
3		
2		
1		
0		

Soma de Verificação

```
byte Checksum(const byte *msg, uint8_t len)
{
    byte c = 0;
    uint8_t i = 0;
    while(i < len)
        c ^= msg[i++];
    return c;
}
```

Auto-Calibração

```
// Read analog input from potentiometer
CurrentAnalogRead = analogRead(pinReadPotentiometer);

// Adapt limits to the real components values
if(CurrentAnalogRead > MAX_ANALOG_VALUE)
    MAX_ANALOG_VALUE = CurrentAnalogRead;
else if(CurrentAnalogRead < MIN_ANALOG_VALUE)
    MIN_ANALOG_VALUE = CurrentAnalogRead;
```

Teste de Detecção e Definição de novos Endereços

```
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
-----------------------------------------
```

Teste de Integridade das Mensagens

```
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
```

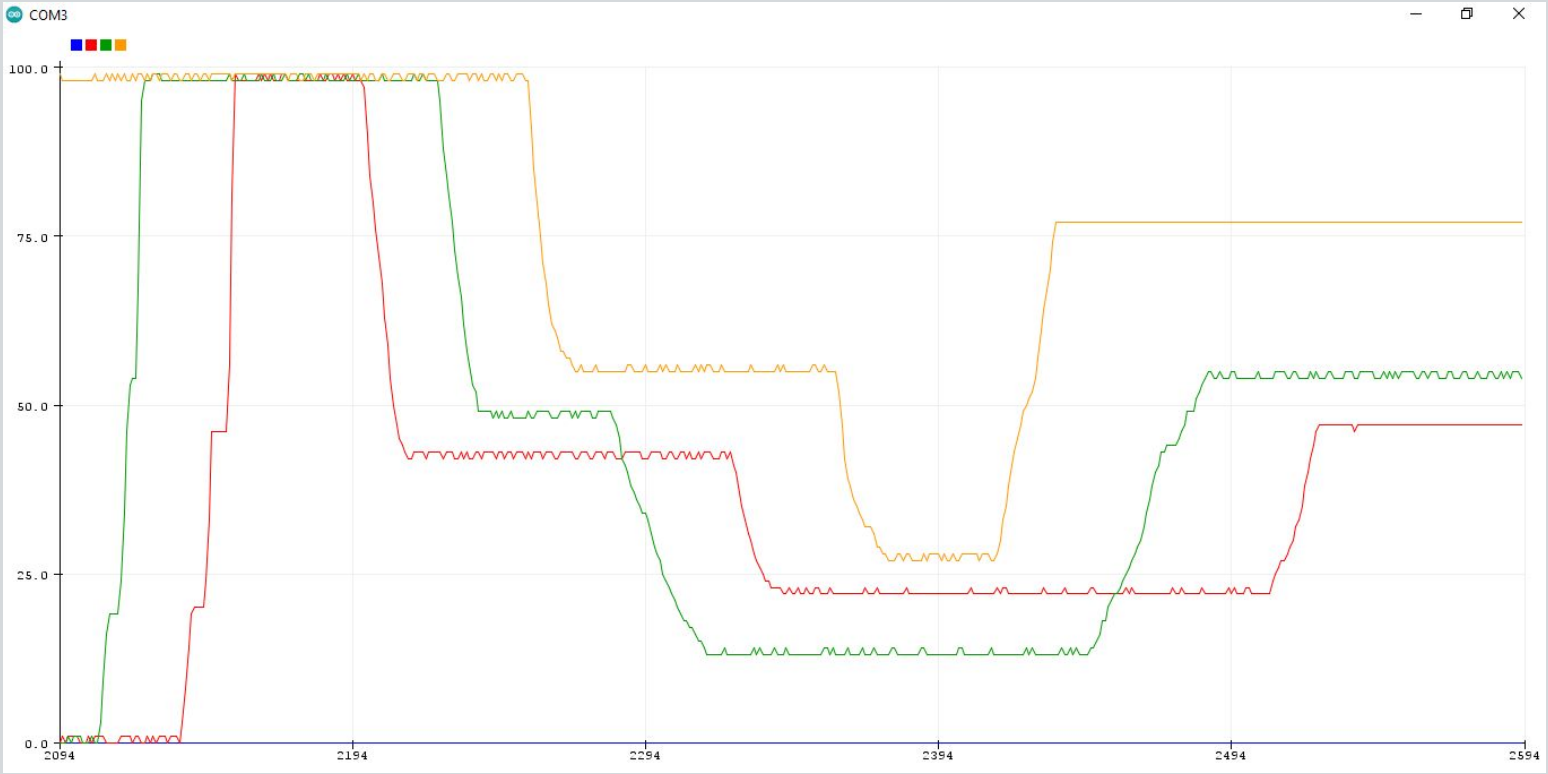
```
DISP:   2, err:   0
ADDR: 0x0C
size:  3 / 3, MATCH
data: MATCH
DISP_02[0]: 36
DISP_02[1]: 00
DISP_02[2]: 24
```

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

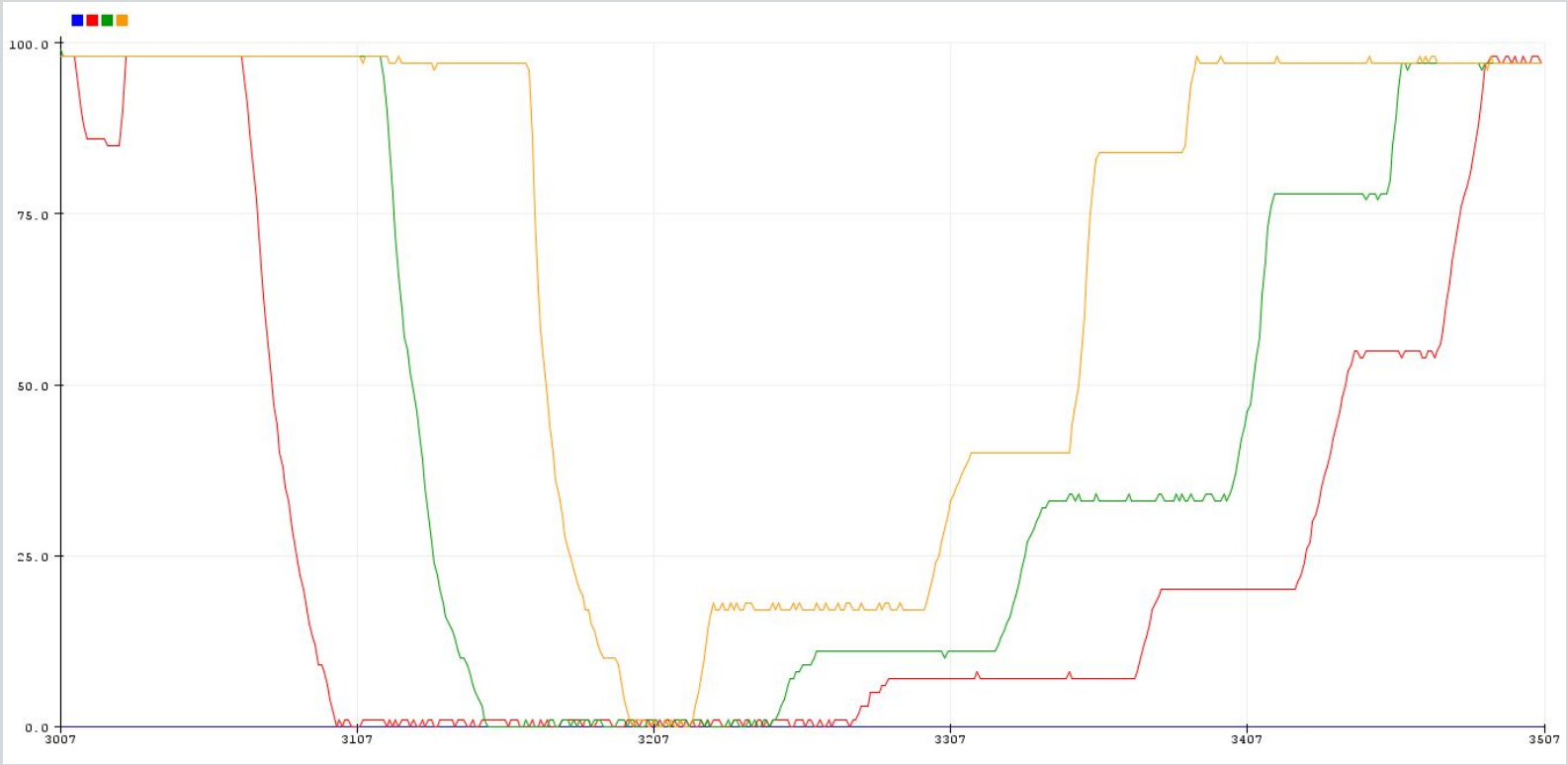
Teste de Integridade das Mensagens

```
-----------------------------------
DISP:   0, err:   4
ADDR: 0x0A
size:  3 / 0, MISMATCH  <<--- !!!
data: MATCH
DISP_00[0]: 00
DISP_00[1]: 00
DISP_00[2]: 00
-----------------------------------
```

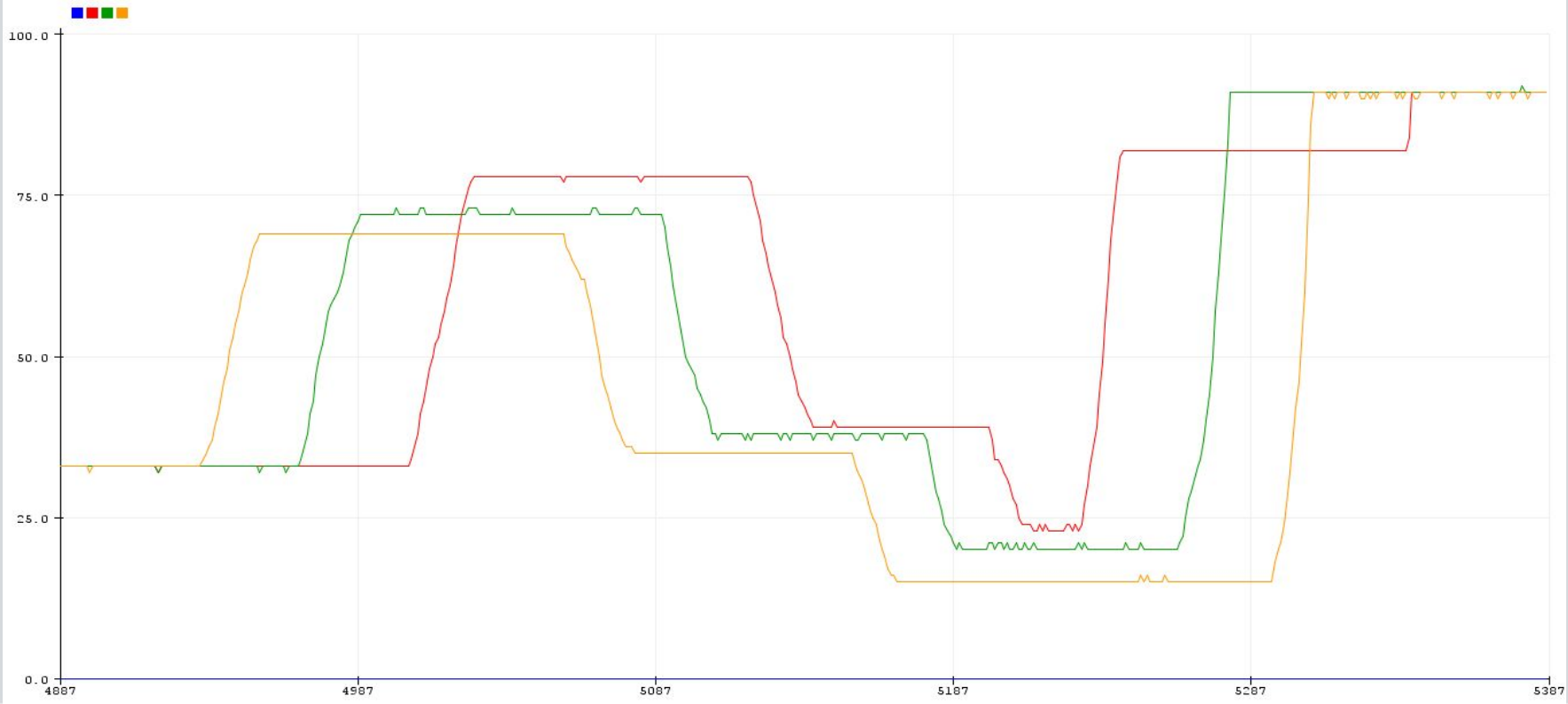
Variação de Volume



Variação de Volume



Variação de Volume



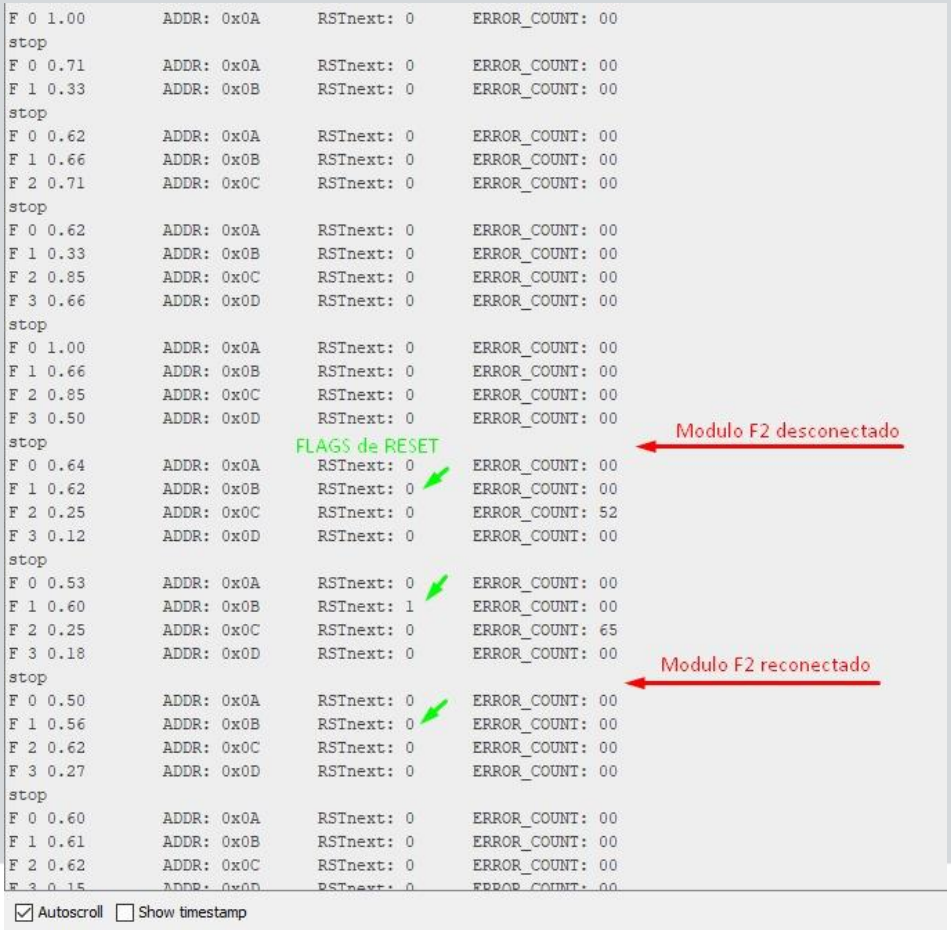
Variação de Volume

```
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
----------------------------------------

255     66      80      48
255     66      80      47
255     83      80      47
255     66      80      48
```

Verificação da flag de reset



Verificação da flag de reset

UFSC

Projeto do diagrama elétrico

Integrante responsável: Assis G Greselle

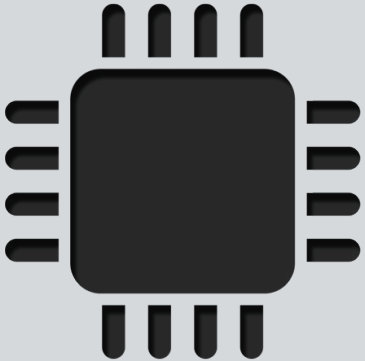
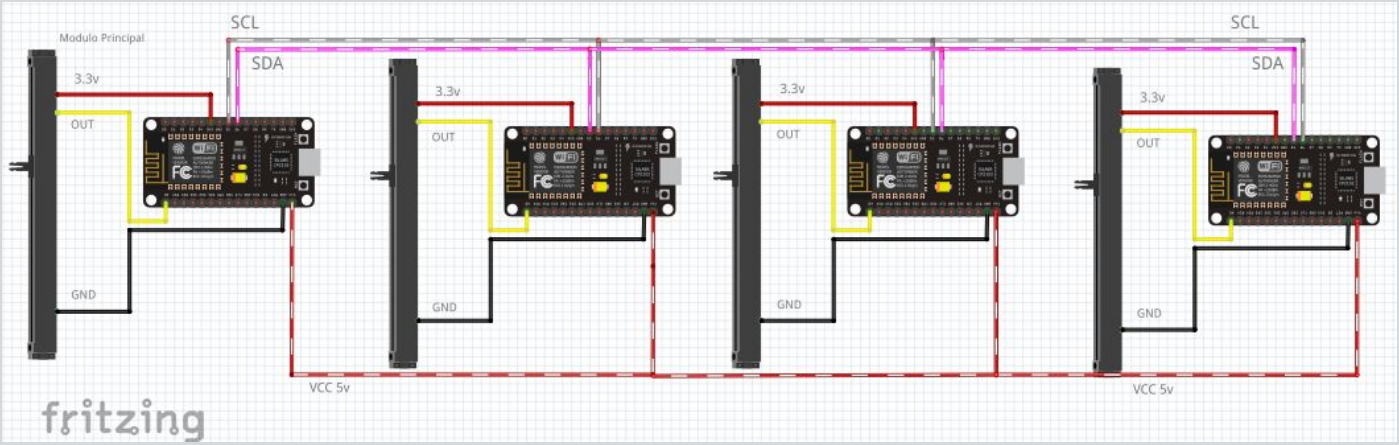
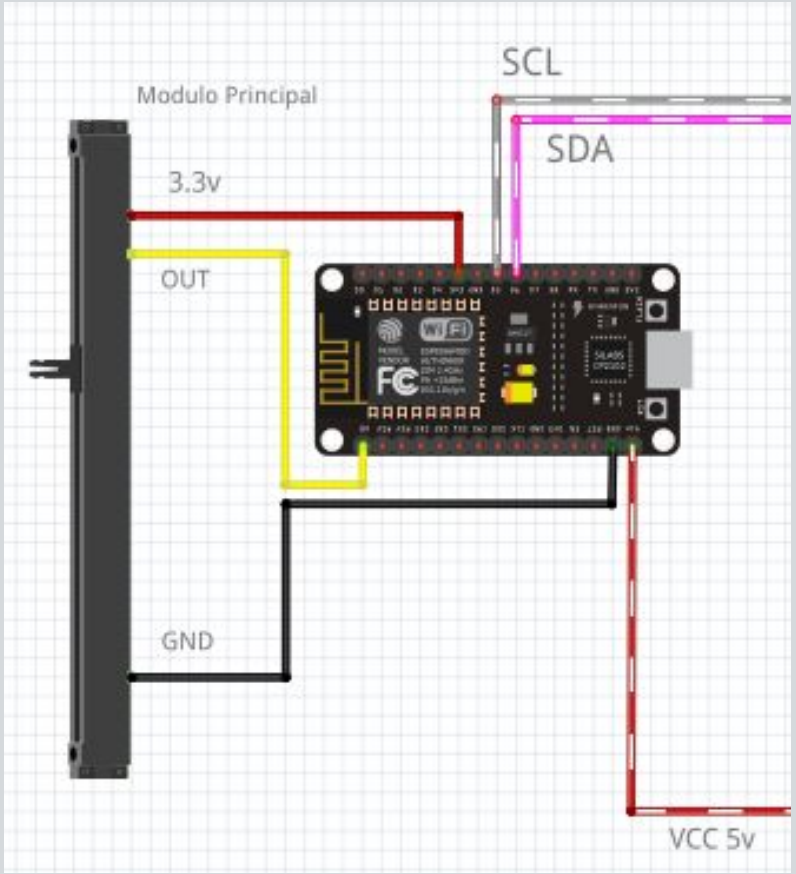


Diagrama Elétrico



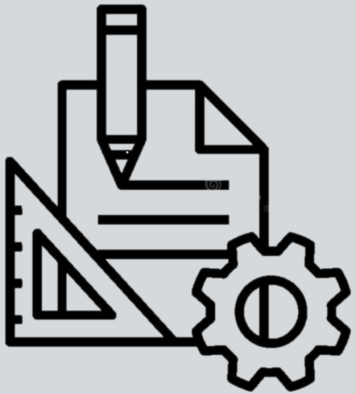
- **RF2:** O usuário deve poder controlar o volume de diferentes aplicações do computador a partir dos módulos;
- **RF3:** Funcionamento plug and play;
- **RF4:** Garantir funcionamento com diferentes quantidades de módulos;



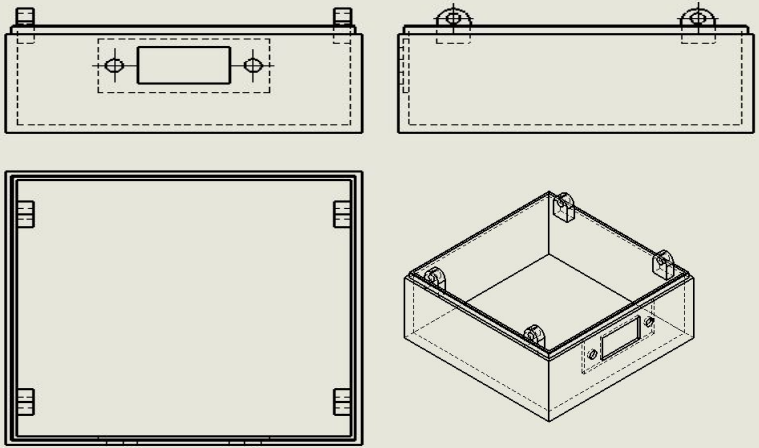
- **RF5:** Envio de informações dos módulos para o computador através de uma única conexão USB;

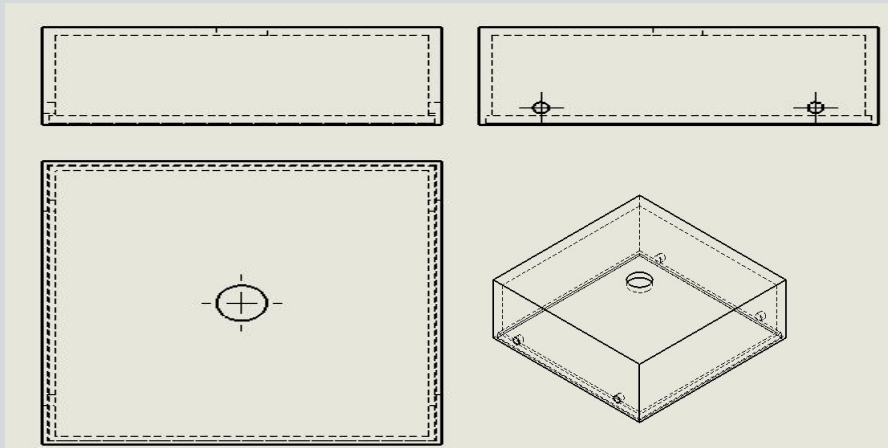
Projeto CAD

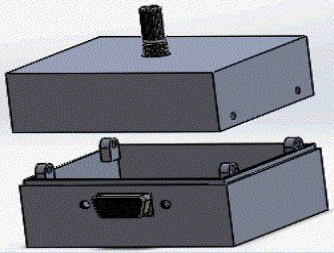
Integrante responsável: Gabriel Luiz Martins



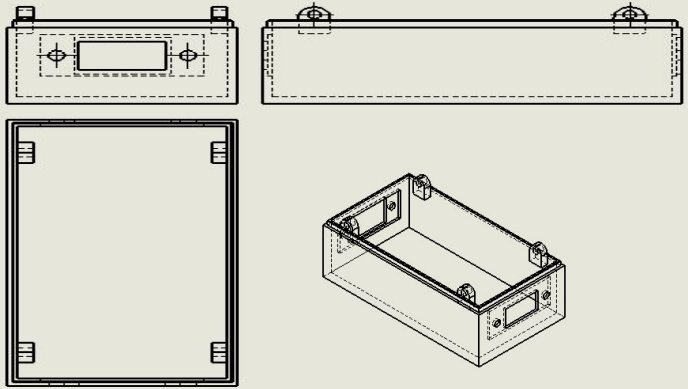
Projeto Master

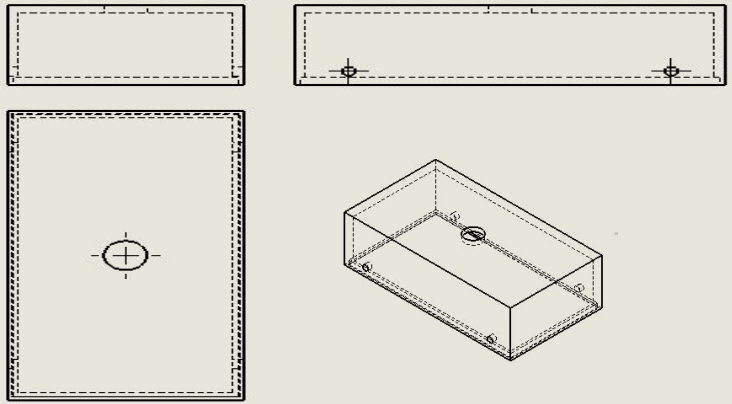


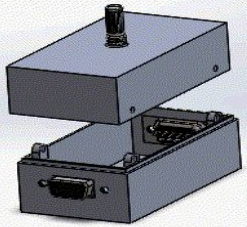




Projeto Slave







Obrigado

UFSC
Joinville