MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
SEÇÃO DE ENSINO DE ENGENHARIA ELÉTRICA (SE/3)

Projeto do Sensor de Presença - Programação Aplicada

PROFESSOR:
NICOLAS SOUZA DE MELO MIRANDA DE OLIVEIRA

ALUNOS:
VICTOR NEVES SALGADO
FELIPE BARBOSA
LUIZ EDUARDO RICHARDT DE AZEVEDO

Resumo

Este relatório apresenta o desenvolvimento do projeto *Sensor AM312 com envio de dados via UDP*, disponível em `https://github.com/Edurichardt/Projeto_Sensor_Am312`. O sistema foi implementado em C++ para leitura de um sensor PIR (AM312) conectado ao canal ADC de uma plataforma Linux embarcada, e realiza o envio periódico das leituras para um computador remoto utilizando o protocolo UDP. O documento descreve a arquitetura do sistema, o funcionamento do software, e o protocolo de comunicação utilizado.

Sumário

1 Introdução

O projeto *Sensor AM312* tem como objetivo realizar a detecção de presença utilizando o sensor PIR AM312, cujo sinal de saída é lido por meio de um conversor analógico-digital (ADC) presente na placa embarcada. O sistema foi desenvolvido em linguagem C++ e projetado para ambientes Linux que possuam interface `IIO` (Industrial Input/Output), permitindo o acesso direto aos valores lidos pelo ADC.

Além da leitura local, o projeto implementa um mecanismo de comunicação com um computador remoto através do protocolo UDP, de modo a transmitir as informações obtidas periodicamente. Isso permite o monitoramento remoto e em tempo real do estado do sensor.

2 Arquitetura do Sistema

O sistema é composto por dois blocos principais: o módulo embarcado e o módulo de recepção remota.

- **Módulo embarcado:** executa a leitura do canal ADC, converte o valor em formato textual e envia o resultado pela rede utilizando o protocolo UDP. Esse módulo é responsável pela coleta de dados e transmissão.
- **Módulo remoto:** recebe os pacotes enviados pelo módulo embarcado e exibe os valores recebidos, podendo armazená-los para posterior análise.

O fluxo básico de funcionamento é o seguinte:

1. Inicialização do sistema e configuração do canal ADC.
2. Leitura periódica do valor proveniente do sensor AM312.
3. Comparação do valor com um limiar definido para determinar presença.
4. Envio do valor e/ou estado detectado via UDP.
5. Retorno ao início do ciclo.

3 Fluxograma do Sistema

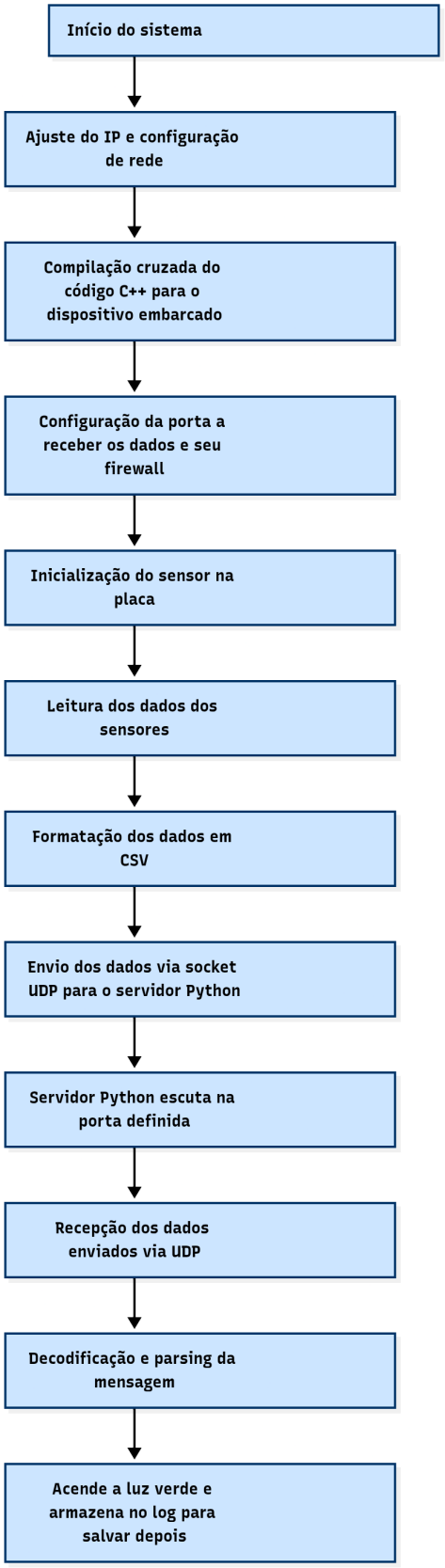

Figura 1: Fluxograma de funcionamento do sistema.

4 Configurações Iniciais do Sistema

Esta seção descreve as etapas de configuração inicial necessárias para a execução do projeto de leitura do ADC e envio de dados via UDP, utilizando uma placa **STM32** com Linux embarcado.

4.1 Configuração do Ambiente de Compilação Cruzada

O código C++ deve ser compilado em uma máquina Linux (ou máquina virtual) utilizando a *toolchain* apropriada para o processador ARM da placa. Para isso, primeiramente, extraia o SDK da ferramenta de compilação cruzada com o seguinte comando:

Listing 1: Extração da toolchain ARM

```
tar -xvf arm-buildroot-linux-gnueabihf_sdk-DK2.tar.gz
```

Após isso, escreva o código-fonte em C++ e salve-o com o nome desejado. Por exemplo:

Listing 2: Criação do arquivo-fonte principal

```
nano projeto.cpp
```

A seguir, compile o código utilizando a *toolchain* extraída:

Listing 3: Compilação cruzada do código C++

```
arm-linux-gnueabihf-g++ projeto.cpp -o projeto -pthread
```

O binário gerado (`projeto`) deverá ser transferido para a placa STM32 via pasta compartilhada ou protocolo de transferência.

4.2 Acesso à Placa via SSH

Com a placa conectada à rede, é possível acessá-la remotamente através do `SSH`. No terminal do sistema hospedeiro (ex.: Windows PowerShell ou terminal Linux), utilize o seguinte comando:

Listing 4: Acesso à placa via SSH

```
ssh root@<ip_da_placa>
```

Onde o IP fixado da placa utilizada é: 192.168.42.2.

4.3 Execução do Programa na Placa

Depois de acessar o sistema da placa via SSH, torne o arquivo binário executável e, em seguida, execute-o:

Listing 5: Execução do programa embarcado

```
chmod +x projeto
./projeto
```

O programa iniciará a leitura periódica do canal ADC e o envio de pacotes UDP para o computador remoto.

4.4 Identificação do Canal ADC

Para determinar qual canal do ADC está sendo utilizado, navegue até o diretório de dispositivos IIO e liste os arquivos disponíveis:

Listing 6: Acesso ao diretório de dispositivos IIO

```
cd /sys/bus/iio/devices/iio:device0/
ls
```

Em seguida, utilize o comando abaixo para verificar qual arquivo representa o canal ativo (aquele que retorna valor zero quando o pino está conectado ao GND):

Listing 7: Leitura do canal ADC

```
cat in_voltageX_raw
```

O número `X` indica o índice do canal correspondente (por exemplo, `in_voltage13_raw` para o canal 13).

4.5 Abertura de Porta UDP no Firewall do Windows

Para que o computador possa receber pacotes UDP enviados pela placa, é necessário liberar a porta utilizada (ex.: `5000`) no firewall. Os passos para isso são realizados na interface gráfica do Windows, conforme descrito no manual do projeto. Basicamente, deve-se abrir as configurações de Firewall do windows, depois ir em criar nova regra, e adicionar uma para a porta que se quer usar.

4.6 Configuração de Rede no Linux

Para que a comunicação UDP funcione corretamente entre a placa STM32 e o computador remoto, é necessário configurar tanto a interface de rede quanto o firewall do sistema Linux.

4.6.1 Definição de Endereço IP Fixo da Placa

Primeiro, veja qual a interface ethernet que seu computador usa, podendo ser eth0, enp1s0, etc.

Listing 8: Verificação da interface ethernet

```
ip link show
```

Defina o endereço IP fixo da interface Ethernet (por exemplo, `eth0`) com o seguinte comando:

Listing 9: Definição de IP fixo para a placa

```
sudo ip addr add 192.168.42.10/24 dev eth0
sudo ip link set eth0 up
```

Esse comando atribui à placa o IP `192.168.42.10` com máscara de sub-rede `/24`. Para verificar se a configuração foi aplicada corretamente:

Listing 10: Verificação do endereço IP configurado

```
ip addr show eth0
```

4.6.2 Liberação da Porta UDP 5000 no Firewall

Em sistemas Linux que utilizam o `ufw` (Uncomplicated Firewall), é necessário liberar a porta UDP 5000 para permitir o recebimento dos pacotes vindos da placa. Execute os seguintes comandos:

Listing 11: Liberação da porta UDP no firewall

```
sudo ufw allow 5000/udp
sudo ufw reload
```

Para confirmar que a regra foi adicionada corretamente:

Listing 12: Verificação das regras do firewall

```
sudo ufw status
```

4.7 Geração do Executável Python

A interface de recepção dos dados (arquivo `server.py`) pode ser empacotada em um executável para facilitar sua execução, utilizando o `pyinstaller`:

Listing 13: Geração do executável Python

```
pip install pyinstaller
pyinstaller --onefile --windowed server.py
```

Após a compilação, será gerado o arquivo `server.exe`, que pode ser executado diretamente sem a necessidade do interpretador Python.

5 Diagrama de Classes

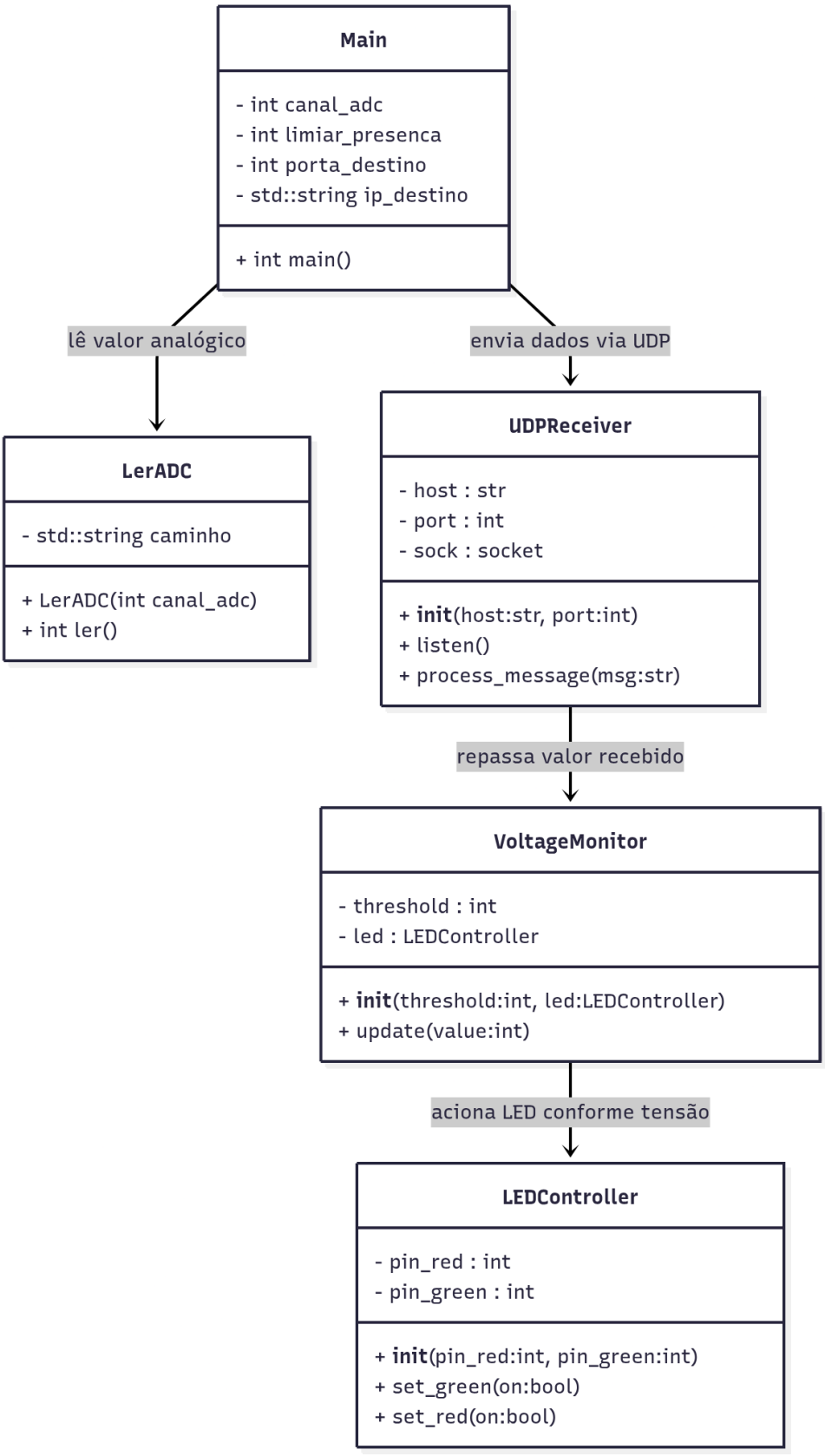

Figura 2: Diagrama de classes do sistema.

O código do projeto é estruturado em torno de classes que organizam as responsabilidades de leitura de dados e envio pela rede. Entre as principais, destacam-se:

- **Classe** `LerADC`: responsável por abrir e ler o valor de um canal ADC através do sistema de arquivos do Linux, utilizando a interface IIO.
- **Classe** `UDPSender`: encarregada de criar o socket UDP, configurar o endereço IP de destino e enviar os pacotes pela rede.
- **Função principal (`main`)**: instancia os objetos das classes, define parâmetros como intervalo de leitura e limiar de presença, e implementa o laço de execução contínuo.

6 Protocolo de Comunicação

O projeto utiliza o protocolo **UDP (User Datagram Protocol)** para a comunicação entre o dispositivo embarcado e o computador remoto. O UDP é um protocolo da camada de transporte da pilha TCP/IP, caracterizado por ser *não orientado à conexão*. Ele não realiza verificação de entrega nem garante a ordem dos pacotes, mas oferece baixa latência e simplicidade de implementação — qualidades desejáveis em sistemas de aquisição de dados periódicos.

Cada pacote enviado contém uma mensagem simples no formato:

```
adc,<valor>
```

onde `<valor>` representa a leitura do ADC em forma textual. Essas mensagens são enviadas a intervalos regulares (por exemplo, a cada 2 segundos) para o endereço e porta especificados no código.

O módulo receptor (executado no computador remoto) deve manter uma porta UDP aberta e escutando o tráfego para receber os dados e salvà-los em um arquivo .txt, caso o usuário queira. Em caso de perda de pacotes, o sistema não realiza retransmissão automática, o que é aceitável para aplicações de monitoramento contínuo onde pequenas perdas não comprometem o funcionamento.

7 Resultados e Testes

Durante os testes, o sistema foi executado em uma plataforma Linux embarcada conectada ao sensor PIR AM312. O canal ADC configurado correspondeu ao dispositivo `iio:device0`, e a leitura dos valores brutos foi validada com oscilações coerentes à detecção de movimento pelo sensor.

O envio via UDP foi validado utilizando um computador remoto na mesma rede local. Ao receber os pacotes, foi possível observar no terminal os valores enviados a cada 2 segundos, sem perdas significativas ou atrasos perceptíveis. Durante a leitura dos dados, o nosso servidor salvou cada vez em que houve presença, em um arquivo txt, a hora em que isso ocorreu.

8 Conclusão

O projeto demonstrou a viabilidade de um sistema simples e eficiente de monitoramento de presença, utilizando hardware acessível e comunicação baseada em UDP. A arquitetura

modular, separando leitura de sensores e transmissão de dados, permite futuras expansões, como inclusão de novos sensores, registro em banco de dados ou integração com uma interface gráfica para visualização em tempo real.

9 Referências

- `https://github.com/Edurichardt/Projeto_Sensor_Am312`
- Documentação do sensor PIR AM312.
- RFC 768 – User Datagram Protocol.
- Documentação da interface IIO do Linux.