UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

ELC1048 - PROJETOS DE SISTEMAS EMBARCADOS

Professor: Carlos Henrique Barriquelo

RELATÓRIO DE PROJETO DE SISTEMAS EMBARCADOS

Data: 12/07/2023

Curso: Engenharia de Computação

Grupo:

- Daniel Youssef de Hollanda Lopes
- Davi Sehnem Castro
- Diego Henrique Nyland
- Gabriel Scaramussa
- Guilherme Ribeiro Silveira
- Leonardo Mello Foletto
- Lucas Rocha Alberti
- Meryane Fernandes Machado

DISPOSITIVO IOT PARA COLETA DE DADOS DE TEMPERATURA, LUMINOSIDADE E ACELERAÇÃO

Resumo

Neste relatório, será mostrado o processo de portabilidade para a plataforma *SAM R21 Xplorer Pro*, de um projeto de redes de sensores sem fio, composta por um sensor de temperatura, luminosidade, e um sensor de orientação, desenvolvido com base no protocolo *OpenThread*. O sistema deve permitir o monitoramento do estado de conexão da rede, dos sensores e da taxa de entrega das mensagens.

***Palavras-chave**: sistemas embarcados, portabilidade, OpenThread, sensores.*

1. INTRODUÇÃO

Os sistemas embarcados são dispositivos eletrônicos integrados em outros sistemas e desempenham um papel crucial na era da Internet das Coisas (IoT). Com o avanço da IoT, os sistemas embarcados se tornaram essenciais para a conectividade e comunicação entre dispositivos e sistemas. Esses sistemas possibilitam a troca de dados e informações em tempo real, facilitando a interconexão de dispositivos e a criação de redes inteligentes. Eles são responsáveis por processar e transmitir informações de forma eficiente, permitindo a automação de processos e a tomada de decisões inteligentes.

Assim, foi proposto o desenvolvimento de uma aplicação de rede de sensores sem fio (IoT) com base no projeto OpenThread. Por se tratar de um trabalho de grande porte foram divididas duas equipes onde a primeira chamada de equipe APP, responsável pela implementação na camada de Software e Aplicação e a equipe HAL responsável pela camada de abstração de hardware, sendo esta última o foco deste relatório.

Este relatório aborda o desenvolvimento de uma aplicação de rede de sensores sem fio (IoT) com base no projeto OpenThread. Foram formadas duas equipes: a equipe APP, responsável pela camada de Software e Aplicação, e a equipe HAL, focada na camada de abstração de hardware. Este relatório se concentra nas contribuições da equipe HAL, que desenvolveu drivers e bibliotecas para o projeto.

O objetivo do projeto é o desenvolvimento de uma aplicação de rede de sensores sem fio (IoT) com base no projeto OpenThread (equipe APP) e portabilidade do projeto para a plataforma SAM R21 Xplorer Pro (equipe HAL).

A implementação deste protocolo é feita para a comunicação através da internet entre duas placas SAM R21. Uma delas é conectada aos kits de extensão “I01-XPRO”, que possui um sensores de temperatura (AT30TS75A-SS8M-T) e um de luminosidade (TEMT6000), e ao kit “BNO055 XPLAINED PRO”, que possui um sensores de orientação. Esta placa age como um transmissor dos sinais coletados dos periféricos. O outro age como um receptor destes sinais, e realiza o processamento para a visualização pelo usuário.

Por fim, este relatório apresenta o que foi realizado pela equipe HAL, quais objetivos foram alcançados, quais as limitações encontradas e qual o resultado final encontrado pela equipe.

2. DESENVOLVIMENTO TEÓRICO

2.1. Objetivos:

- Realizar a portabilidade do projeto OpenThread RTOS (OTRTOS) que utiliza FreeRTOS, LwIP para o HW do laboratório (SAM R21 Xplorer Pro).
- Realizar os testes necessários pela equipe de APP para rodar a aplicação na plataforma de HW e no OT-RTOS.
- Realizar a demonstração do projeto e documentar os passos necessários para portar a aplicação: como compilar, usar, testar e executar a demonstração.

2.2. O sistema deve permitir o monitoramento:

- Do estado da rede.
- Da conexão dos dispositivos.
- Da taxa de entrega das mensagens

2.3. Tabela de materiais utilizados

- Computadores laboratório
  - Sistema Operacional: Ubuntu 18.04
- Placas de desenvolvimento:
  - ARM SAM R21 Xplorer Pro

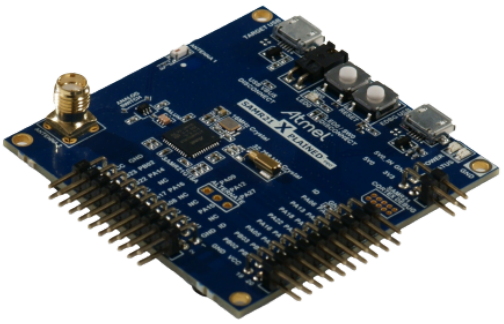

Imagem 1 - ARM SAMR21.

- Sensores
  - Sensor Atmel 42078-IO1 Xplained-Pro

Imagem 2 - Sensor IO1 Xplained Pro.

- Sensor Atmel BNO055-AN011 Xplained-Pro

Imagem 3 - Sensor BNO055.

2.4. Aplicação dos sensores:

- IO1 Xplained Pro - Luz ambiente (TEMT6000)

O sensor consiste em um fototransistor com um resistor em seu coletor. Essencialmente, quanto maior a luminosidade do local, maior será a corrente que passa pelo transistor, gerando assim uma queda de tensão maior no resistor de coletor. Essa tensão é posteriormente lida e convertida pelo conversor analógico-digital do microcontrolador.

Abaixo visualizamos a conexão do phototransistor, retirada do datasheet do IO1 Xplained Pro:

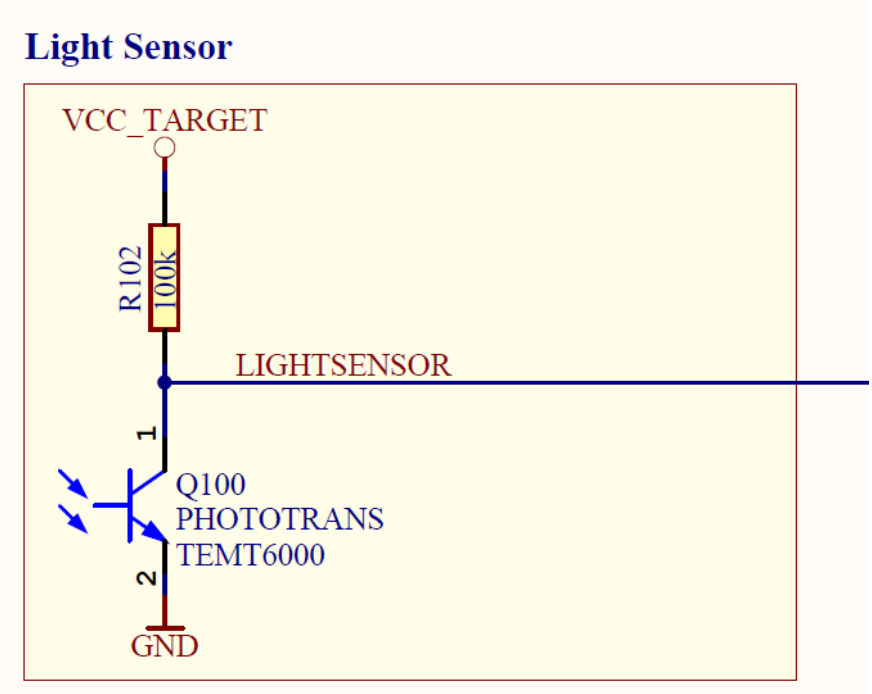

Imagem 04 - Phototransistor do IO1 Xplained Pro

A tensão $Vcc\ target$ utilizada será $3.3V$, que é a mesma da alimentação dos microcontroladores, conforme podemos observar na figura a seguir, retirada do *datasheet* do *SAMR21:*

Table 42-2. General Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V_{DD}	Power supply voltage	Voltage on VDDIN, VDDIO and VDDANA(2)	1.8(1)	3.3	3.6	V
$V_{DD1.8}$	Power supply voltage (on AVDD and DVDD)	External supply voltage(3)	1.7	1.8	1.9	V
V_{DDANA}	Analog supply voltage		1.8(1)	3.3	3.6	V
T_A	Temperature range		-40	25	125	°C
T_J	Junction temperature		-	-	100	°C

Notes: 1. With BOD33 disabled. If the BOD33 is enabled, check Table 42-17.
2. Even if an implementation uses the external 1.8V voltage supply $V_{DD1.8}$ it is required to connect V_{DD}.
3. AT86RF233 register 0x10 (VREG_CTRL) needs to be programmed to disable internal voltage regulators and supply blocks by an external 1.8V supply, refer to "Voltage Regulators (AVREG, DVREG)" on page 983.

.

Imagem 05 - Tensão de alimentação típica do *SAMR21.*

Portanto, equacionando temos:

$$I_c = \frac{(V_{CC\ TARGET} - V_{LIGHTSENSOR})}{R_C} = \frac{(3.3V - V_{LIGHTSENSOR})}{100K\Omega}$$

A imagem a seguir foi retirada do datasheet do phototransistor, e representa a relação entre a corrente do coletor com a luminosidade do ambiente. Podemos aproximar a relação por:

$$Luminosidade = \frac{I_c \cdot 2lux}{1\mu A}$$

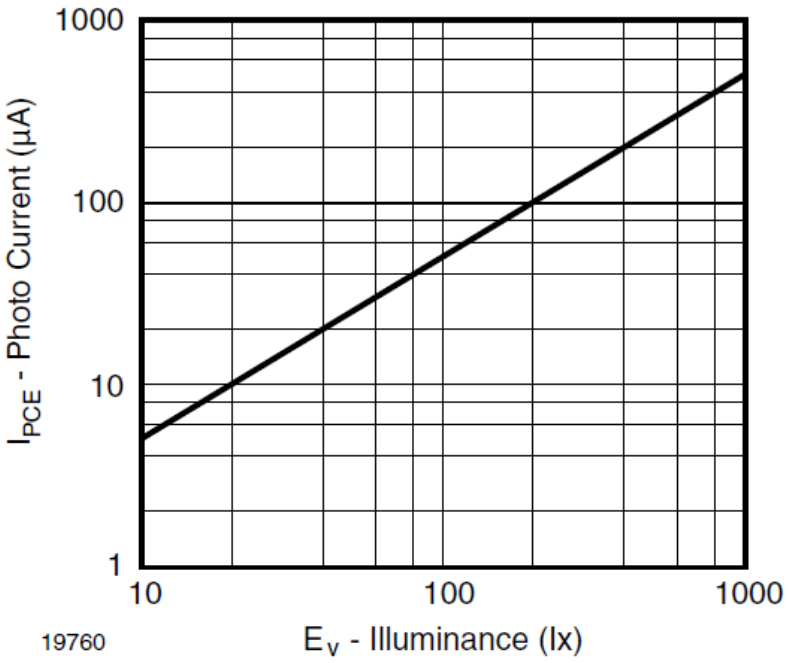

Imagem 06 - Relação lux por corrente do phototransistor

Juntando as duas equações, temos:

$$Luminosidade = \frac{(3.3V - V_{LIGHTSENSOR}) * 2\ lux}{100K\Omega * 1\mu A}$$

$$Luminosidade = (3.3 - V_{LIGHTSENSOR}) * 20$$

- BNO055 Xplained Pro - Direcional

Consiste em um sensor de nove eixos, composto por um microcontrolador e três sensores individuais, sendo eles um acelerômetro, magnetômetro e um giroscópio.

O sensor também é capaz de identificar a falta ou não de movimento, e reagir de acordo, através de interrupções. Dessa forma, caso não haja movimento após determinado tempo, o sensor entra em repouso, e cessa a transmissão de dados. Quando este detecta algum movimento, é gerada uma nova interrupção que reinicia o envio dos dados do sensor. O formato padrão do envio dos dados de posição e situação de operação é mostrado na Imagem 7.

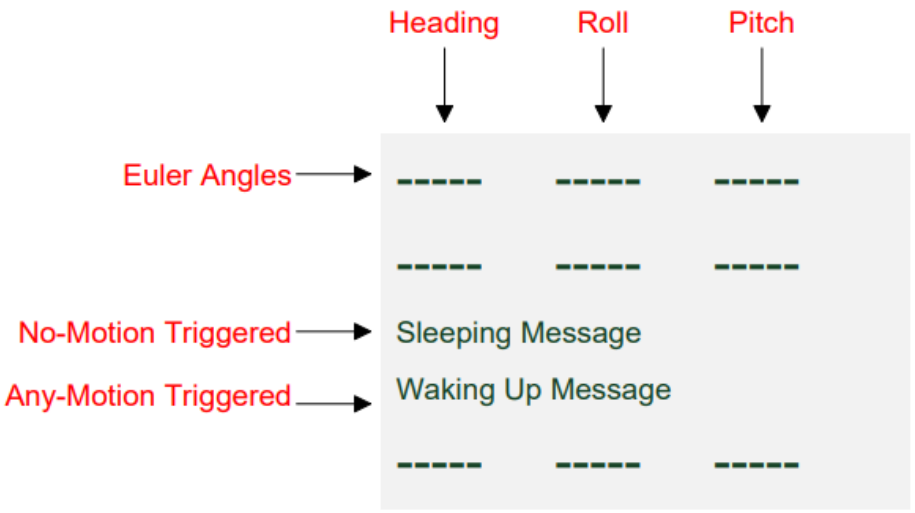

Imagem 07 - Formato padrão da visualização de dados do BNO055

3. DESENVOLVIMENTO PRÁTICO

3.1. Condicionamento dos Sensores

O primeiro sensor em que se trabalhou foi o sensor de iluminância presente no IO1 Xplained Pro. Para realizar seu condicionamento, foi utilizado o ambiente IDE, disponibilizado juntamente com o dispositivo, para gerar os códigos que permitem sua utilização, configurados com base nas especificações de pinagem fornecidas pela documentação de *design* do componente.

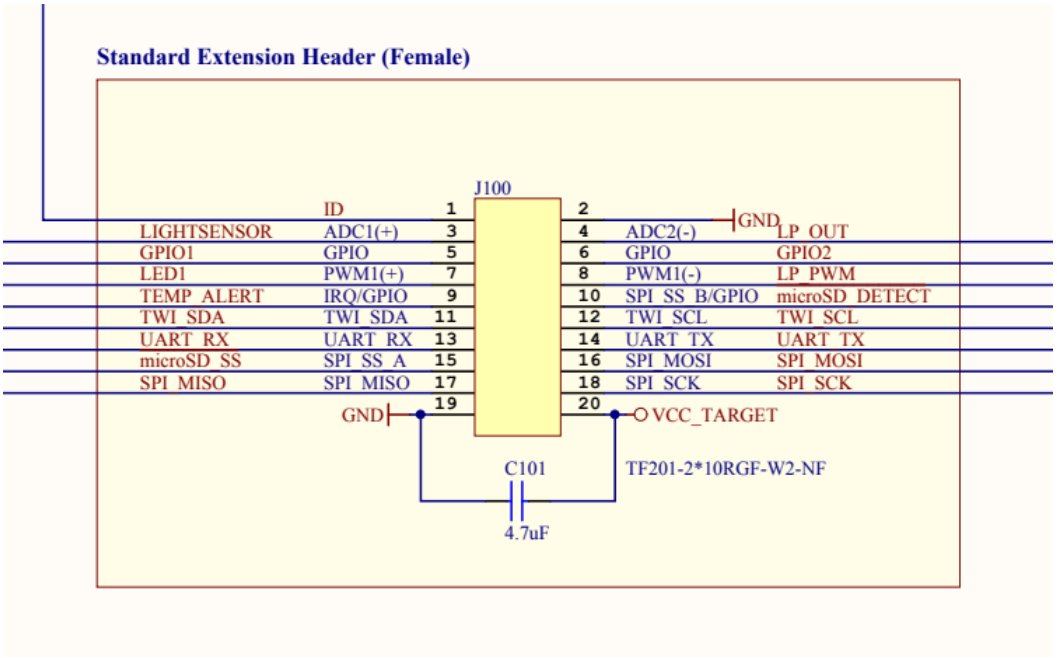

Imagem 08 - Pinagem da Documentação de *Design*

Dessa forma, configurou-se o pino PN06 do sensor como a entrada de sinal analógica do sensor de iluminância, e utiliza-se o conversor AD da placa para quantizá-lo em um sinal digital. Em código, é feita a função 'GET_light_sensor( )' que lê o valor quantizado pelo ADC e, primeiro recalcula o valor de tensão correspondente, e então relaciona este resultado com o valor de iluminância correspondente, retornando-o ao final da operação.

A implementação do sensor de temperatura é mais complexa que a do sensor de iluminância, devido a implementação por comunicação I2C. Isso acontece pois é necessário a leitura e escrita de registradores específicos da placa SAMR21. Apesar destas configurações não estarem disponíveis para este modelo, elas estavam para a placa SAMD21. Dessa forma, foram utilizados os arquivos destinados para a SAMD21 como base, e realizou-se a adaptação para a placa SAMR21, que é utilizada no projeto. Com a comunicação devidamente configurada, mais uma vez utilizou-se o ambiente IDE disponibilizado para determinar os pinos utilizados.

Devido a comunicação I2C, são necessários dois pinos para realizar a transferência dos dados. O primeiro, setado como o pino PN16, é a linha de dados serial. O segundo, definido como o pino PN17, é a linha de *clock* serial.

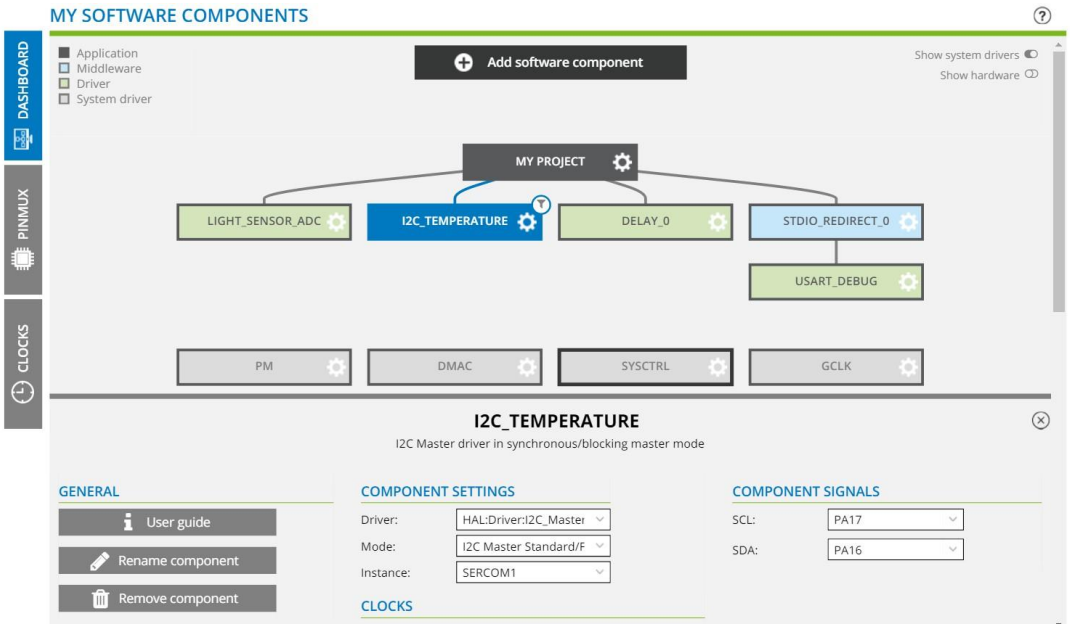

Imagem 09 - Ambiente IDE

Para obter o valor da temperatura em código, utiliza-se a função ‘GET_temperature_sensor( )’. Diferente do caso anterior, todo condicionamento do sinal é feito automaticamente pelos componentes, e portanto não é preciso realizar a conversão do sinal digital para o valor correspondente de temperatura.

Para a implementação do sensor direcional, mais uma vez utiliza-se o ambiente IDE e as informações da documentação de *design* fornecidas. Primeiro, realiza-se a configuração da utilização dos LEDs da placa. Nota-se, pela documentação, que estes são dependentes somente do estado alto/baixo do pino correspondente, e portanto sua implementação é simples.

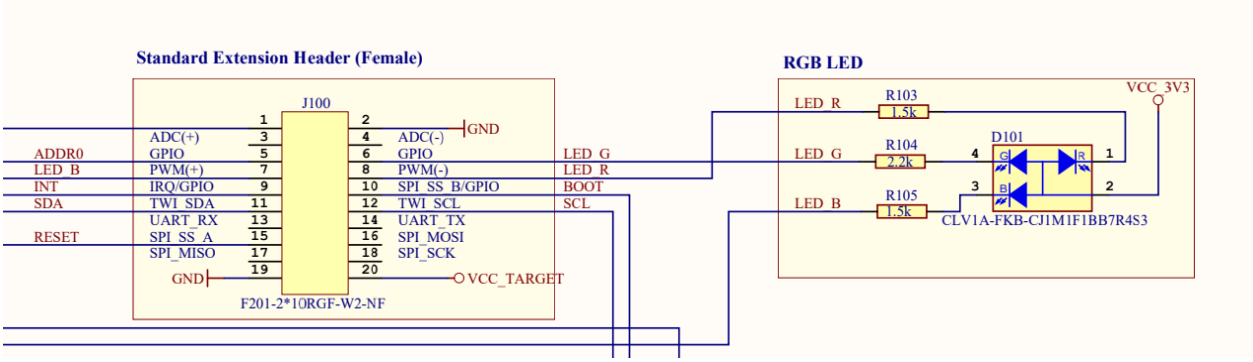

Imagem 10 - Documentação da Pinagem dos LEDs

Em seguida, de acordo com as informações de documentação, é preciso definir o tipo de comunicação que será utilizado. Como mencionado na seção anterior, é possível escolher entre I2C e SPI, de acordo com o nível dos pinos PS0 e PS1. Por padrão, a comunicação está definida como I2C.

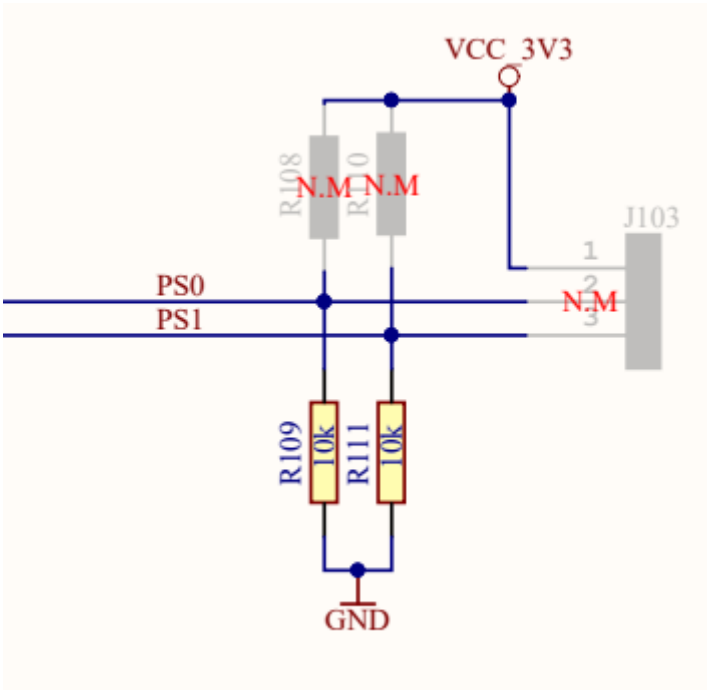

Imagem 11 - Definição em *Hardware* do Tipo de Comunicação

Para mudar esta configuração, seria necessário modificar o *hardware* do sensor, ao substituir a conexão existente entre os pinos e o GND através dos resistores R109 e R111, para uma conexão com VCC, que também requereria adicionar os resistores R108 e R110. Como não sabíamos se uma modificação física da placa era permitida, manteve-se a comunicação por I2C.

Para realizar a configuração do sensor através desta comunicação, adotou-se o mesmo procedimento utilizado para o sensor de temperatura, já que mais uma vez a SAMR21 não possui as configurações prontas, enquanto que a SAMD21 possui. Entretanto, não foi possível realizar a adaptação necessária para fazer a comunicação com a SAMR21, e portanto não conseguimos realizar a medição direcional utilizando os sensores do BNO055. Caso a comunicação venha a ser implementada corretamente, as configurações para o tratamento dos dados coletados já estariam prontas, e estes poderiam ser visualizados/transmitidos sem problemas.

Para testar se os dados coletados dos sensores estão corretos, foi implementada a comunicação por cabo USB entre a placa e o computador. Os resultados são mostrados na Figura abaixo. Ambos os sensores funcionam de acordo com o esperado, entretanto nota-se que o sensor de iluminância é limitado em sua operação, já que ele não identifica valores abaixo de um certo *threshold.* Dessa forma, seria mais útil utilizá-lo não para detectar um valor específico de LUX, mas sim para determinar se o ambiente está iluminado ou não.

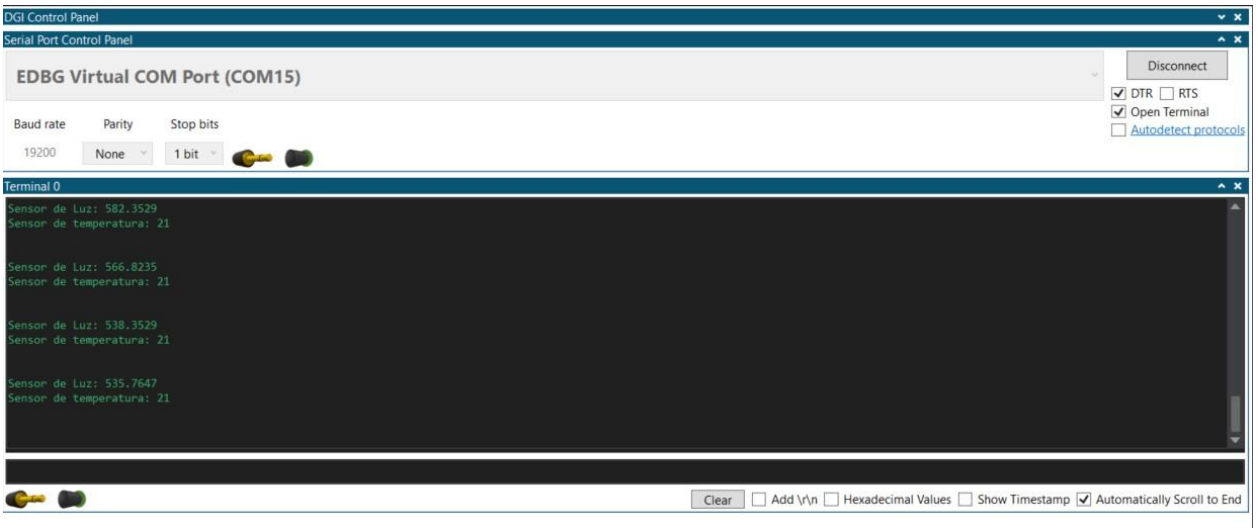

Imagem 12 - Dados dos Sensores Visualizados em Terminal

3.2. Comunicação de Rede

O OpenThread tem uma infraestrutura, que possibilita o desenvolvimento de redes de malha eficientes e confiáveis, com suporte para uma ampla gama de dispositivos conectados, com isso, foi utilizado um simulador para fazer a comunicação de Rede utilizando as bibliotecas OpenThread. Primeiramente através de um terminal linux realizamos o processo de "Building" do exemplo disponibilizado através do github "ot-samr21", para obtermos o arquivo ("ot-cli-ftd") necessário para realizar os seguintes procedimentos:

- Simulação de uma rede Thread:

Essa etapa foi criada uma rede no OpenThread onde foi possível fazer algumas simulações junto ao software Docker e o sistema operacional Linux Ubuntu 18.04.6. Após a criação dessa rede os dispositivos utilizados foram conectados e verificados.

- Autenticação de nós com comissionamento:

Neste segundo passo, foi necessŕio utilizar Commissioner que é o servidor de autenticação escolhido para novos dispositivos Thread. Pois, para que haja tráfego entre o IPv6 global entre os dispositivos e a Internet usando um roteador de borda do Thread é necessaŕio que os nós sejam autenticados.

Para realizar esta autenticação um dispositivo precisa funcionar como um comissário, onde o utilizador fornece as credenciais de rede necessárias para que os dispositivos participem da rede.

O arquivo “ot-cli-ftd” pode ser encontrado através no repositório na pasta main e os logs de simulação na pasta main/network_simulation_tests. A seguir podemos observar os testes realizados no terminal linux(simulação):

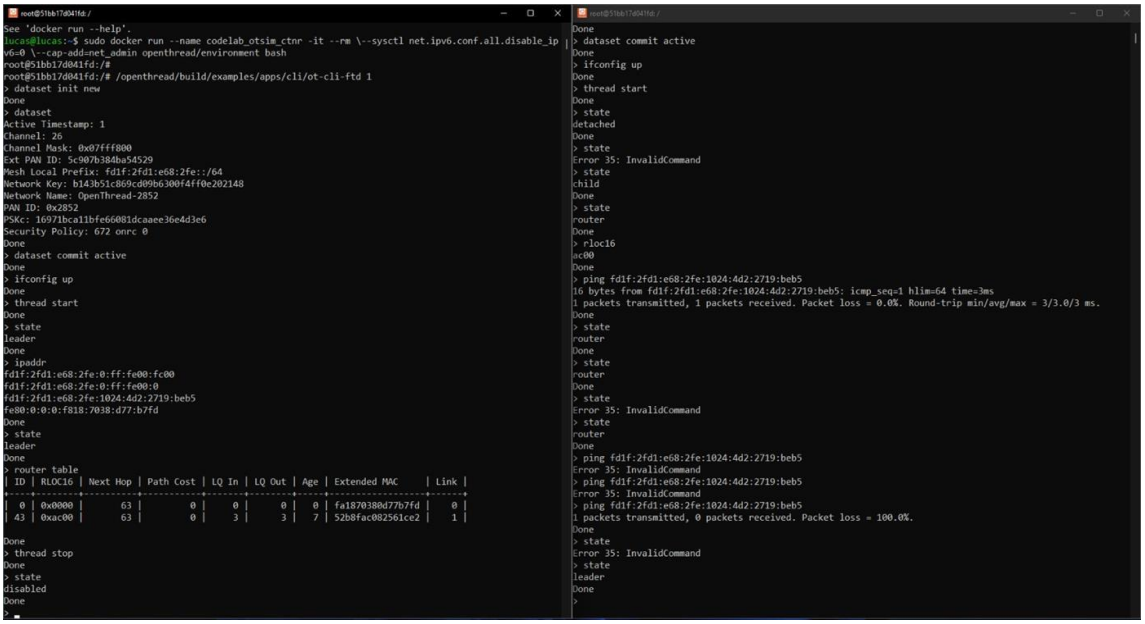

Imagem 13 - Execução da simulação de uma rede Thread.

- Execução da comunicação:

A partir da simulação, realizamos o processo de “Flash Binaries” para o hardware através do Microchip Studio, onde foi possível executar os testes a seguir:

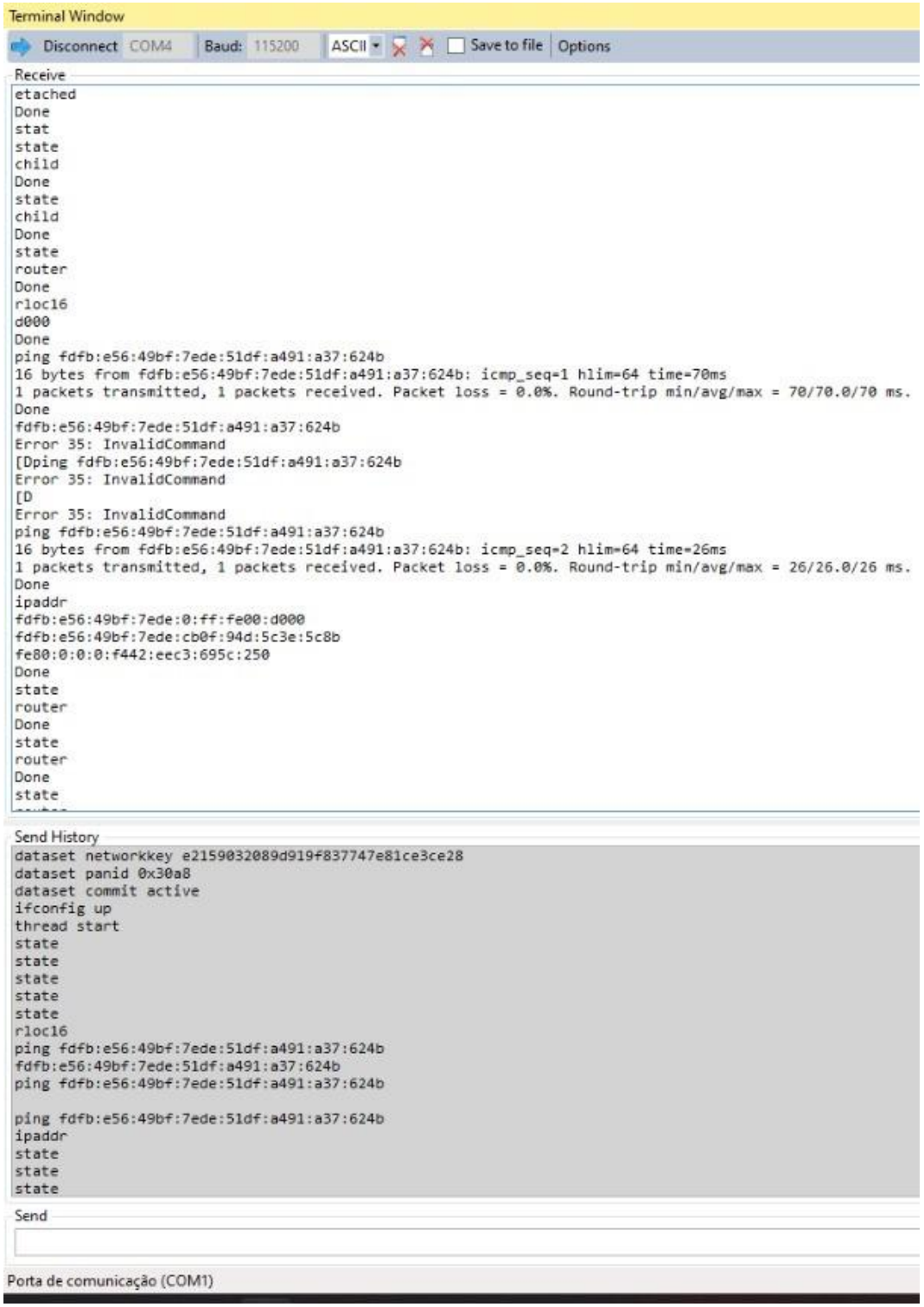

Imagem 14 - Execução da simulação de uma rede Thread Nó 1.

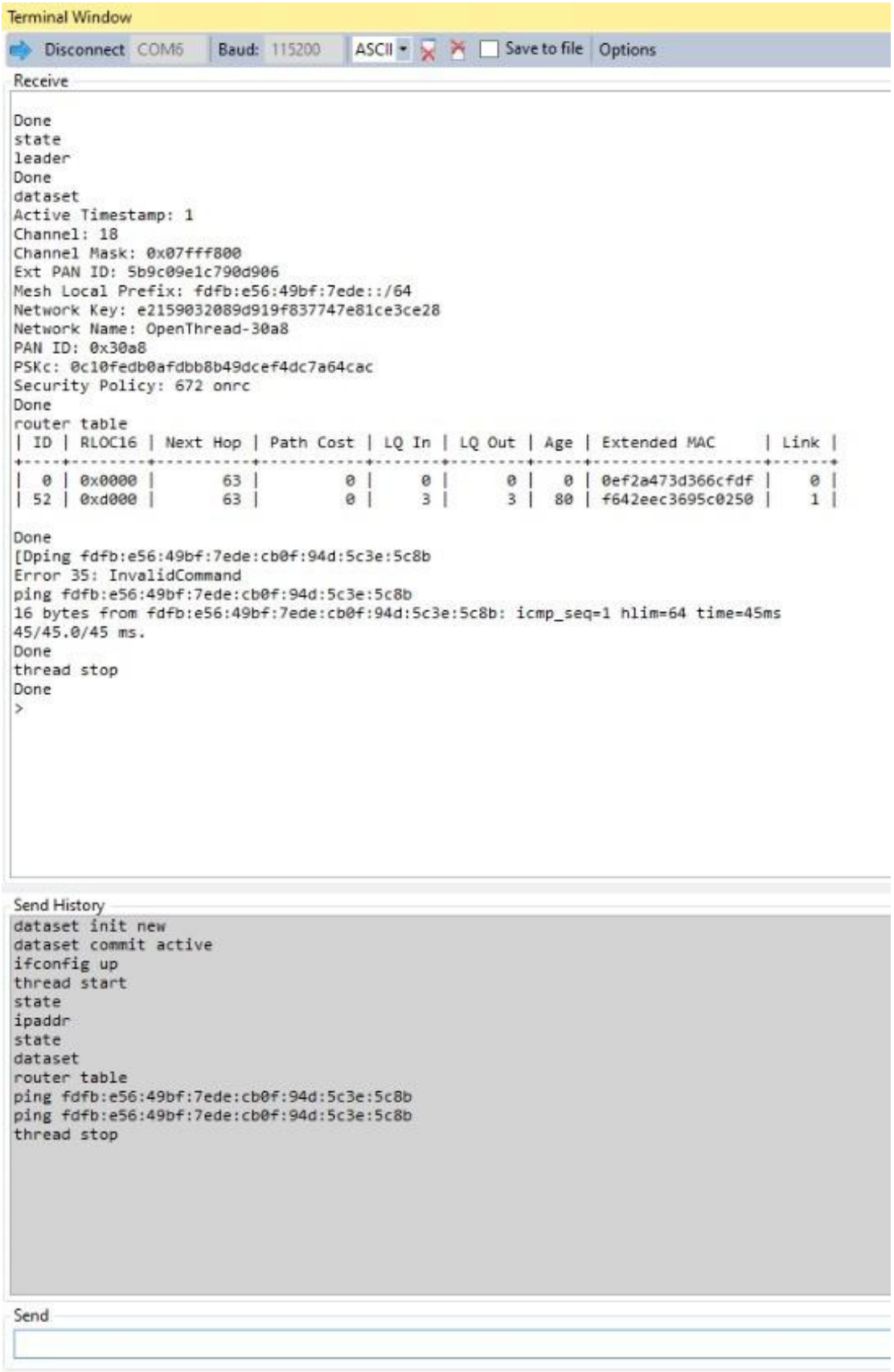

Imagem 15 - Execução da simulação de uma rede Thread Nó 2.

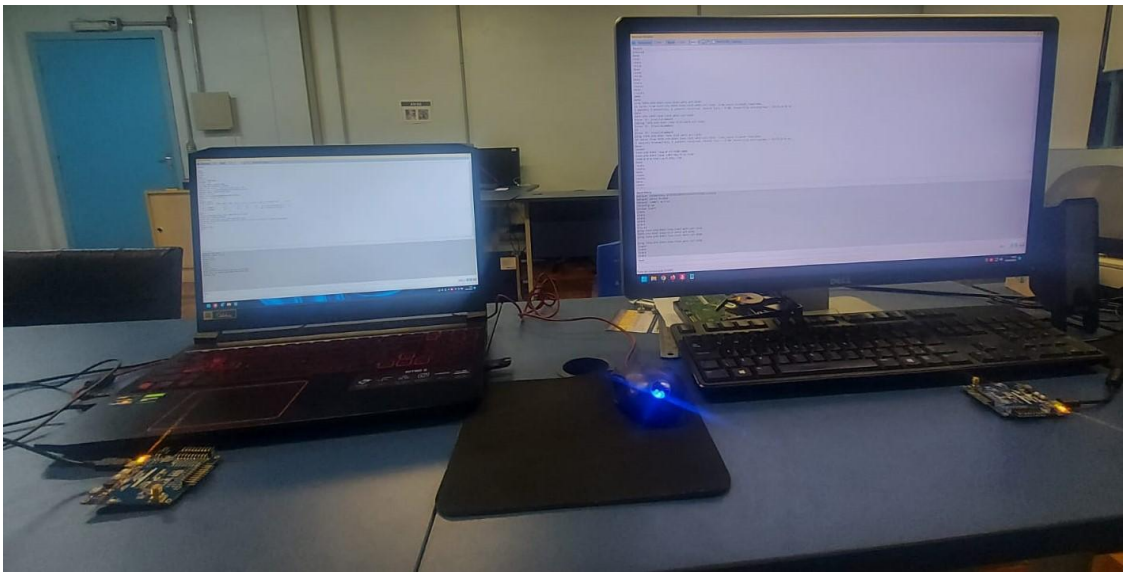

Imagem 16 - Execução da simulação de uma rede no hardware SAM R21 Xplorer Pro.

4. CONCLUSÃO

A equipe HAL trabalhou na portabilidade do projeto para a plataforma SAM R21 Xplorer Pro. A plataforma SAM R21 Xplorer Pro é uma placa de desenvolvimento baseada no microcontrolador SAM R21 da Microchip Technology. Foi realizada a adaptação do código da aplicação para que ele pudesse ser executado com sucesso na plataforma SAM R21 Xplorer Pro.

Durante o desenvolvimento houve modificações nos drivers de hardware, configurações de pinos, otimizações de desempenho e outras tarefas necessárias para garantir a compatibilidade e a funcionalidade adequada, porém o resultado final não foi como esperado.

Na parte de teste dos sensores houve um avanço significativo, conforme documentação disponível no repositório. O sensor de iluminação apresentou certa facilidade em sua implementação, porém o sensor de temperatura apresentou um nível de complexidade maior.

Quanto a comunicação foi realizada a simulação de uma rede Thread utilizado o OpenThread, como resultado obtido foi possível dar ping entre os dispositivos utilizados e a obtenção dos pacotes enviados, além disso, foi possível visualizar os dispositivos na tabela da rede.

Portanto, a execução desse projeto foi bem desafiadora e exigiu muito tempo de pesquisa e testes, apesar dos objetivos não serem alcançados na sua totalidade o trabalho se mostra promissor para eventuais consultas e melhoramentos. Em relação ao conhecimento adquirido acredita-se que com as trocas de experiências houve a concretude da aprendizagem além de ter sido uma boa experiência em como trabalhar e desenvolver projetos de modo colaborativo.

5. REFERÊNCIAS

- [1]. Repositório Github.
  <https://github.com/llucasroot/ot-rtos_elc1048>.

- [2]. Repositório OpenThread RTOS.
  <https://github.com/openthread/ot-rtos>.

- [3]. Projeto de Sistemas Embarcados. Moodle UFSM.
  <https://ead06.proj.ufsm.br/pluginfile.php/4022753/mod_resource/content/1/AULA_TDD.pdf>.

- [4]. Repositório OpenThread on SAMR21 Example.
  <https://github.com/openthread/ot-samr21>.

- [5]. Software Microchip.
  <https://www.microchip.com/en-us/tools-resources#xplained_pro>.

- [6]. Arm GNU Toolchain.
  <https://developer.arm.com/Tools%20and%20Software/GNU%20Toolchain>.

- [7]. Arm GNU Toolchain.
  <https://www.microchip.com/en-us/tools-resources/develop/libraries/advanced-software-framework>.

- [8]. OpenThread.
  <https://openthread.io/?hl=pt-br>.