

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA
CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA**

GREICILI DOS SANTOS FERREIRA

**DESENVOLVIMENTO DO FIRMWARE DE UMA PLACA DE
COMUNICAÇÃO CELULAR PROPRIETÁRIA**

FLORIANÓPOLIS, 2026.

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA
CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA**

GREICILI DOS SANTOS FERREIRA

**DESENVOLVIMENTO DO FIRMWARE DE UMA PLACA DE
COMUNICAÇÃO CELULAR PROPRIETÁRIA**

Trabalho de Conclusão de Curso submetido
ao Instituto Federal de Educação, Ciência e
Tecnologia de Santa Catarina como parte
dos requisitos para obtenção do título de
Bacharel em Engenharia Eletrônica.

Orientador:
Prof. Renan Augusto Starke, Dr.Eng.

FLORIANÓPOLIS, 2026

Ficha de identificação da obra elaborada pelo autor.

Ferreira, Greicili dos Santos
Desenvolvimento do firmware de uma placa de comunicação celular proprietária / Greicili dos Santos Ferreira; orientação de Renan Augusto Starke. - Florianópolis, SC, 2026.
68 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Eletrônica. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Firmware. 2. LPC2368. 3. MA510GL. 4. RL-RTX. 5. Comunicação celular. I. Augusto Starke, Renan. II. Instituto Federal de Santa Catarina. III. Desenvolvimento do firmware de uma placa de comunicação celular proprietária.

DESENVOLVIMENTO DO FIRMWARE DE UMA PLACA DE COMUNICAÇÃO CELULAR PROPRIETÁRIA

GREICILI DOS SANTOS FERREIRA

Este trabalho foi julgado adequado para obtenção do título de Bacharel em Engenharia Eletrônica e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 10 de julho de 2026.

Banca Examinadora:

Documento assinado digitalmente
 **RENAN AUGUSTO STARKE**
Data: 05/08/2026 10:29:25-0300
Verifique em <https://validar.iti.gov.br>

Prof. Renan Augusto Starke, Dr. Eng.

Documento assinado digitalmente
 **CHARLES BORGES DE LIMA**
Data: 04/08/2026 12:19:29-0300
Verifique em <https://validar.iti.gov.br>

Prof. Charles Borges de Lima, Dr. Eng.

Documento assinado digitalmente
 **DANIEL LOHMANN**
Data: 04/08/2026 11:29:50-0300
Verifique em <https://validar.iti.gov.br>

Prof. Daniel Lohmann, Msc. Eng.

Dedico esse trabalho a Deus,
pois sem Ele eu nada seria.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo dom da vida e por sua presença constante, que me amparou ao longo desta trajetória, concedendo sabedoria nos momentos de dificuldade, forças para continuar e fé para seguir acreditando e lutando até o fim.

Ao meu namorado, Júnior, pelo apoio incondicional durante este período desafiador. Obrigada por cuidar de mim, cozinhando para nós para que eu pudesse dedicar mais tempo aos estudos, por me incentivar nos momentos de desânimo, por me confortar quando as dúvidas surgiam e por ouvir pacientemente minhas constantes inquietações. Sou grata, sobretudo, por sempre me lembrar de que, no final, tudo dá certo e que, se ainda não deu certo, é porque não chegou no final.

Ao grupo Lógica-E, pela confiança depositada em mim, pela oportunidade de participar de uma etapa tão importante no desenvolvimento de um produto e por acreditar no meu potencial profissional.

Ao meu orientador, Prof. Renan Augusto Starke, pela paciência, pelo apoio, pelo tempo dedicado e pela confiança depositada em mim ao longo da realização deste trabalho.

Ao Instituto Federal de Santa Catarina (IFSC) e a todos os professores, pela formação de excelência proporcionada, pela infraestrutura oferecida e pelas oportunidades de pesquisa, que contribuíram significativamente para o meu desenvolvimento profissional e pessoal.

“Estudante: aplica-te com espírito de apóstolo aos teus livros, com a convicção íntima de que essas horas e horas são já — agora! — um sacrifício espiritual oferecido a Deus, proveitoso para a humanidade, para o teu país, para a tua alma.”
São Josemaria Escrivá – Sulco, 522

RESUMO

A empresa RMS Registradores de Energia, especializada no desenvolvimento de registradores eletrônicos para medição de grandezas elétricas em sistemas de baixa e média tensão, utiliza em seus equipamentos uma placa de comunicação responsável pela transmissão remota dos dados de medição ao servidor privado SEGECON por meio da rede celular. Essa placa é fornecida por terceiros, tornando a empresa dependente da propriedade intelectual do fabricante para a manutenção e evolução do *firmware*. Diante desse cenário, este trabalho apresenta o desenvolvimento de um *firmware* embarcado para essa placa, composta pelo microcontrolador LPC2368 e pelo modem celular MA510GL, cuja arquitetura é previamente definida e não permite modificações em sua configuração física. O objetivo do trabalho consiste em desenvolver um *firmware* capaz de intermediar a comunicação entre o registrador de energia — no qual participam deste processo a placa de comunicação e a placa RVL— e o servidor SEGECON, garantindo a transferência das informações de medição. Para o desenvolvimento da aplicação foi empregado o sistema operacional de tempo real RL-RTX, utilizando recursos como *Event Flags*, *Mailboxes*, *Memory Pools* e *User Timers*, além de comandos AT destinados à configuração do modem, ao estabelecimento da conexão com a rede celular e à comunicação com o servidor por meio da criação de *sockets* TCP. Também foi implementado o protocolo de comunicação utilizado pela RMS, que define a estrutura das mensagens e os mecanismos para a troca de informações entre o servidor SEGECON e a placa RVL. A validação do *firmware* foi realizada por meio da análise dos *logs* de comunicação gerados durante a etapa final de testes, contemplando o estabelecimento da conexão, a transferência de blocos de memória de massa e o envio periódico de mensagens KEEP-ALIVE. Durante a validação, foram executados 46 ciclos de comunicação, dos quais 36 ocorreram conforme o comportamento esperado, resultando em uma taxa de sucesso de aproximadamente 78%. Apesar da identificação de falhas relacionadas ao gerenciamento do encerramento da conexão TCP e à reinicialização de temporizadores, o sistema apresentou capacidade de recuperação automática, mantendo sua estabilidade operacional e preservando sua principal funcionalidade, que consiste na intermediação da comunicação entre o servidor SEGECON e a placa RVL.

Palavras-chave: *Firmware*. LPC2368. MA510GL. RL-RTX. Comunicação celular.

ABSTRACT

RMS Energy Meters, a company specialized in the development of electronic energy meters for measuring electrical quantities in low- and medium-voltage systems, uses a communication board in its devices to remotely transmit measurement data to its private server, SEGECON, through the cellular network. This communication board is supplied by a third-party manufacturer, making the company dependent on the supplier's intellectual property for firmware maintenance and further development. In this context, this work presents the development of embedded firmware for this communication board, which is based on the LPC2368 microcontroller and the MA510GL cellular modem. Since its hardware architecture is predefined, modifications to its physical configuration are not possible. The objective of this work is to develop firmware capable of mediating communication between the energy meter—which comprises the communication board and the RVL board—and the SEGECON server, ensuring the reliable transmission of measurement data. The application was developed using the RL-RTX real-time operating system, employing features such as Event Flags, Mailboxes, Memory Pools, and User Timers, as well as AT commands for modem configuration, cellular network connection establishment, and communication with the server through the creation of TCP sockets. The communication protocol adopted by RMS was also implemented, defining the message structure and the mechanisms for information exchange between the SEGECON server and the RVL board. Firmware validation was performed through the analysis of communication logs generated during the final testing phase, including connection establishment, mass memory block transfer, and the periodic transmission of KEEP-ALIVE messages. During the validation process, 46 communication cycles were executed, of which 36 behaved as expected, resulting in an approximate success rate of 78%. Although issues related to TCP connection termination management and timer reinitialization were identified, the system demonstrated automatic recovery capability, maintaining operational stability while preserving its primary functionality of mediating communication between the SEGECON server and the RVL board.

Keywords: Firmware. LPC2368. MA510GL. RL-RTX. Cellular communication.

LISTA DE FIGURAS

Figura 1 – Diagrama da comunicação entre o SEGECON e o registrador de energia	19
Figura 2 – Comunicação entre UE e eNB	20
Figura 3 – Arquitetura da rede celular	21
Figura 4 – Placa de comunicação	22
Figura 5 – Diagrama do LPC2368.....	23
Figura 6 – Interligação do VIC com o núcleo.....	24
Figura 7 – Mapeamento das fontes de interrupção nos canais do VIC	25
Figura 8 – Sistema para a geração dos <i>clocks</i> no LPC2368.....	25
Figura 9 – Diagrama funcional do PLL do LPC2368	26
Figura 10 – Fluxo de comunicação entre TA e TE durante a execução de comandos AT	30
Figura 11 – Transmissão de URC pelo modem	30
Figura 12 – Sintaxe comandos AT	31
Figura 13 – Estabelecimento de uma conexão utilizando <i>sockets</i> no modelo cliente-servidor.....	32
Figura 14 – <i>Frame</i> do protocolo de comunicação da RMS	34
Figura 15 – Pontos de acesso ao TX e RX da UART1 e UART2.....	41
Figura 16 – Circuito para obter os dados de TX e RX simultaneamente.....	42
Figura 17 – Processos realizados pela placa de comunicação	43
Figura 18 – <i>Hardware</i> implementado para o desenvolvimento do <i>firmware</i>	47
Figura 19 – Fluxograma do funcionamento do <i>firmware</i>	51
Figura 20 – Arquitetura do <i>firmware</i>	53

LISTA DE QUADROS

Quadro 1 – Comandos AT utilizados para a conexão com a rede celular e a comunicação com o servidor	32
Quadro 2 – Sequência de conjuntos de dados solicitados pelo SEGECON	35
Quadro 3 – Exemplo de troca de dados entre o SEGECON e o registrador	44
Quadro 4 – Exemplo de monitoramento e fechamento de conexão.....	44
Quadro 5 – Comunicação com a placa RVL e identificação da requisição do número de série	45
Quadro 6 – Modificação no arquivo <i>Startup.s</i>	48
Quadro 7 – Código de inicialização UART1	49
Quadro 8 – Rotina para o tratamento de interrupções da UART1.....	50
Quadro 9 – Envio do número de série, em resposta à solicitação do SEGECON.....	57
Quadro 10 – <i>Log</i> de transferência de blocos da memória de massa	57
Quadro 11 – Monitoramento da conexão e envio do <i>frame</i> KEEP-ALIVE, com o fechamento do <i>socket</i> TCP e início da nova comunicação.....	59

LISTA DE ABREVIATURAS E SIGLAS

3GPP	<i>3rd Generation Partnership Project</i>
ACK	<i>Acknowledgement</i>
APN	<i>Access Point Name</i>
ARM	<i>Advanced RISC Machine</i>
AT	<i>Attention</i>
CCO	<i>Current Controlled Oscillator</i>
CoAP	<i>Constrained Application Protocol</i>
CR	<i>Carriage Return</i>
CRC	<i>Cyclic Redundancy Check</i>
eNB	<i>Evolved Node B</i>
EPC	<i>Evolved Packet Core</i>
E-UTRAN	<i>Evolved Universal Terrestrial Radio Access Network</i>
FIFO	<i>First In, First Out</i>
FIQ	<i>Fast Interrupt reQuest</i>
GNSS	<i>Global Navigation Satellite System</i>
GPRS	<i>General Packet Radio Service</i>
I ² C	<i>Inter-Integrated Circuit</i>
I ² S	<i>Inter-IC Sound</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
IRQ	<i>Interrupt ReQuest</i>
ISR	<i>Interrupt Service Routine</i>
JTAG	<i>Joint Test Action Group</i>

LF	<i>Line Feed</i>
LTE-M	<i>Long Term Evolution for Machines</i>
LwM2M	<i>Lightweight Machine-to-Machine</i>
MM	Memória de massa
MQTT	<i>Message Queuing Telemetry Transport</i>
MTC	<i>Machine-Type Communications</i>
NB-IoT	<i>Narrowband Internet of Things</i>
PCLK	<i>Peripheral Clock</i>
PDP	<i>Packet Data Protocol</i>
PLL	<i>Phase-Locked Loop</i>
RTC	<i>Real-Time Clock</i>
RTOS	<i>Real-Time Operating System</i>
RL-RTX	<i>Real-Time Library - Real Time eXecutive</i>
SIM	<i>Subscriber Identity Module</i>
SRAM	<i>Static Random-Access Memory</i>
TCP	<i>Transmission Control Protocol</i>
TE	<i>Terminal Equipment</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>
UE	<i>User Equipment</i>
URC	<i>Unsolicited Result Code</i>
USB	<i>Universal Serial Bus</i>
VIC	<i>Vectored Interrupt Controller</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Justificativa	15
1.2	Definição do Problema	16
1.3	Objetivo Geral	16
1.4	Objetivos Específicos	16
1.5	Estrutura do Trabalho	17
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	Arquitetura do sistema	18
2.2	Comunicação celular	19
2.3	Placa de comunicação	21
2.3.1	Arquitetura de <i>hardware</i> da placa	22
2.3.2	Microcontrolador LPC2368	23
2.3.2.1	<i>VIC</i>	24
2.3.2.2	<i>Sistema de clock do LPC2368</i>	25
2.3.2.2.1	<i>PLL</i>	26
2.3.2.3	<i>UART</i>	27
2.3.3	Modem MA510GL	28
2.3.3.1	<i>Comandos AT</i>	29
2.3.3.2	<i>Comunicação por meio de sockets TCP</i>	31
2.3.3.3	<i>Principais comandos utilizados</i>	32
2.4	Protocolo de comunicação da RMS	33
2.5	Sistemas operacionais de tempo real	35
2.5.1	RL-RTX	36
2.5.2	Recursos utilizados do RL-RTX	37
2.5.2.1	<i>Event Flags</i>	37
2.5.2.2	<i>Memory Pool</i>	38
2.5.2.3	<i>Mailboxes</i>	38
2.5.2.4	<i>User Timer</i>	38
2.6	Resumo	39
3	METODOLOGIA	41
3.1	Obtendo os <i>logs</i> da comunicação	41
3.1.1	Comunicação com o módulo MA510GL	42
3.1.1.1	<i>Troca de dados entre o servidor e o registrador</i>	43
3.1.1.2	<i>Monitoramento da rede e fechamento da conexão</i>	44
3.1.2	Comunicação com a placa RVL	45
3.2	Configuração do ambiente de desenvolvimento	46
3.3	Desenvolvimento do <i>firmware</i>	47
3.3.1	Ajustes no arquivo <i>Startup.s</i>	48
3.3.2	Configuração das UARTs	48
3.3.3	Definição da estrutura do <i>firmware</i>	49
4	APRESENTAÇÃO DOS RESULTADOS	56
4.1	Cenário 1 – Estabelecimento da conexão com o servidor SEGECON	56
4.2	Cenário 2 – Transferência de blocos MM	57
4.3	Cenário 3 – Manutenção e encerramento da conexão	59

4.4	Análise e discussão dos resultados	60
5	CONSIDERAÇÕES FINAIS.....	62
5.1	Sugestões para trabalhos futuros	63
	REFERÊNCIAS.....	64

1 INTRODUÇÃO

Em 2024, o grupo Lógica-E¹, empresa com sede no bairro Forquilha, em São José/SC, onde a autora deste trabalho atua profissionalmente, integrou ao seu portfólio a empresa RMS Registradores de Energia², que atua há aproximadamente 30 anos no desenvolvimento de registradores eletrônicos de energia. A RMS é especializada no desenvolvimento de equipamentos destinados à medição de grandezas elétricas, como tensão e corrente de fase, fator de potência, frequência da tensão e distorção harmônica total de corrente e tensão, entre outras, em sistemas de baixa e média tensão.

Antes da aquisição da RMS, o grupo contava com quatro empresas: Cianove, voltada para soluções na área da construção civil; LeSense, responsável por fornecer tecnologias para coleta de dados no ambiente industrial; Lógica-E Engenharia de produto, dedicada ao desenvolvimento de soluções de engenharia, e GigaFlex, especializada na montagem de placas e produtos eletrônicos. A incorporação da RMS ampliou o portfólio do grupo e diversificou suas áreas de atuação. Contudo, essa integração também trouxe desafios técnicos internos, especialmente relacionados à manutenção e adaptação de equipamentos legados, tema central deste trabalho.

Um dos desafios com a incorporação dos produtos da RMS refere-se a uma placa de comunicação, utilizada por quatro modelos de registradores de energia para a transmissão de dados ao servidor privado, denominado SEGECON. Esta placa atua como interface entre o servidor e a placa RVL (responsável pelo processamento e armazenamento das medições), realizando a comunicação via *internet* por meio da infraestrutura de telefonia móvel. O principal problema é que ela é adquirida de um fornecedor externo, gerando dependência da propriedade intelectual de terceiros. Diante disso, busca-se reverter essa situação para que a sua produção e manutenção sejam feitas internamente.

Para atingir esse objetivo, o primeiro passo consiste no desenvolvimento de um novo *firmware* para a placa atualmente utilizada, permitindo a manutenção e

¹ GRUPO LÓGICA-E. **Home**. Disponível em: <https://logicae.com.br/>. Acesso em: 06 abr. 2026.

² RMS. **Home**. Disponível em: <https://rms.ind.br/>. Acesso em: 06 abr. 2026.

atualização das placas já instaladas e garantindo a continuidade da comercialização do produto enquanto uma nova solução de *hardware* não é desenvolvida. Dessa forma, o desenvolvimento do *firmware* envolve etapas como a análise da arquitetura de *hardware* da placa, a obtenção e análise de *logs* de comunicação para compreensão do funcionamento do protocolo de comunicação, a configuração do ambiente de desenvolvimento e, por fim, a realização de testes para validação do funcionamento do sistema.

1.1 Justificativa

A incorporação da empresa RMS ao grupo Lógica-E trouxe diversos desafios técnicos, entre os quais se destaca a utilização de uma placa de comunicação responsável por intermediar a transmissão de dados entre os registradores de energia e o servidor SEGECON. O principal problema associado a essa solução está no fato da placa ser adquirida de um fornecedor externo, o que configura um ponto de fragilidade, ao estabelecer uma relação de dependência e limitar o domínio sobre o funcionamento do sistema.

Nesse contexto, este trabalho justifica-se pela necessidade de desenvolver um novo *firmware* para a placa de comunicação, possibilitando maior controle sobre seu funcionamento e reduzindo a dependência de fornecedores externos. Dessa forma, torna-se viável a produção, manutenção e evolução da solução internamente. Adicionalmente, o desenvolvimento do *firmware* demanda um estudo aprofundado de sua operação, incluindo os mecanismos de tratamento e transmissão de dados, contribuindo para a geração de conhecimento técnico e para a documentação da tecnologia empregada.

Assim, o trabalho não apenas atende a uma demanda imediata da empresa, mas também contribui como etapa fundamental para o desenvolvimento futuro de uma nova placa, reduzindo a dependência de fornecedores externos para fortalecer a autonomia tecnológica do grupo.

1.2 Definição do Problema

Como desenvolver um *firmware* novo para a placa de comunicação existente, mantendo o *hardware* original, de modo a garantir a continuidade da comunicação atual entre os registradores de energia e o servidor de dados da RMS?

1.3 Objetivo Geral

Desenvolver o *firmware* para uma placa de comunicação existente, capaz de realizar a comunicação entre os registradores de energia e o servidor SEGECON, com base no *hardware* disponível e respeitando as limitações do microcontrolador utilizado.

1.4 Objetivos Específicos

- a) Analisar a arquitetura de *hardware* da placa de comunicação existente, identificando seus principais componentes, interfaces e limitações de processamento e memória.
- b) Analisar o protocolo de comunicação proprietário procurando compreender o seu funcionamento e as suas particularidades.
- c) Estudar o funcionamento do modem celular utilizado para o estabelecimento da conexão e envio de dados ao servidor SEGECON.
- d) Projetar a arquitetura do *firmware*, considerando as restrições impostas pelo *hardware* disponível.
- e) Implementar rotinas de comunicação entre o microcontrolador e o modem celular utilizado, incluindo configuração, estabelecimento de conexão e transmissão de dados.
- f) Implementar a comunicação entre a placa de comunicação e a placa RVL para a coleta e o tratamento dos dados.
- g) Implementar o envio dos dados coletados para o servidor remoto SEGECON.

- h) Realizar testes de funcionamento e validação do *firmware* desenvolvido, avaliando estabilidade, confiabilidade e desempenho da comunicação.
- i) Documentar a solução desenvolvida como base para futuros projetos de *hardware* e *firmware* próprios da empresa.

1.5 Estrutura do Trabalho

O presente trabalho está organizado em cinco capítulos. O primeiro Capítulo introduz o tema, abordando a justificativa, a definição do problema, os objetivos gerais e específicos e a estrutura do trabalho.

O segundo Capítulo reúne a fundamentação teórica necessária para o desenvolvimento do *firmware* da placa de comunicação da RMS. Inicialmente, é descrita a arquitetura do sistema analisado, seguida da apresentação da placa de comunicação e de seus principais componentes de *hardware*, com destaque para o microcontrolador LPC2368 e o modem celular MA510GL. Em seguida, são abordados a comunicação via infraestrutura de telefonia móvel, o protocolo de comunicação da RMS e os conceitos de sistemas operacionais de tempo real empregados no desenvolvimento do *firmware*.

O terceiro Capítulo descreve a metodologia adotada. Inicialmente, são apresentados os procedimentos utilizados para a obtenção e análise dos *logs* de comunicação entre o servidor SEGECON e o registrador de energia, envolvendo a placa RVL e a placa de comunicação. Na sequência, são detalhados o ambiente de desenvolvimento, os ajustes realizados no projeto base e as etapas de desenvolvimento do novo *firmware*, incluindo a configuração das UARTs e a definição da arquitetura baseada no sistema operacional RL-RTX.

O quarto Capítulo é dedicado à apresentação, análise e discussão dos resultados obtidos, avaliando o comportamento do *firmware* durante a comunicação com o servidor SEGECON e verificando sua conformidade com os requisitos definidos para o sistema.

Por fim, o quinto Capítulo apresenta as considerações finais do trabalho, destacando as principais contribuições alcançadas, as limitações identificadas e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para o desenvolvimento do *firmware* proposto. Inicialmente, é descrita a arquitetura do sistema no qual a placa de comunicação está inserida, bem como os principais componentes de *hardware* envolvidos no processo de comunicação. Em seguida, aborda-se o protocolo utilizado para a troca de informações entre os dispositivos, apresentando sua estrutura e os aspectos necessários para a compreensão de seu funcionamento. Por fim, são apresentados os conceitos relacionados aos sistemas operacionais de tempo real empregados em sistemas embarcados, com foco no RL-RTX, utilizado no desenvolvimento do *firmware*. Esses conhecimentos servem de base para as etapas de análise, implementação e validação apresentadas nos capítulos posteriores.

2.1 Arquitetura do sistema

A arquitetura do sistema é composta por três elementos principais: o registrador de energia, a infraestrutura de telefonia móvel e o servidor SEGECON. O registrador de energia realiza a aquisição e o armazenamento temporário das grandezas elétricas medidas, transmitindo-as ao servidor SEGECON por meio da rede celular. O SEGECON é responsável pelo recebimento e armazenamento centralizado das informações provenientes dos registradores da RMS.

Internamente, o registrador de energia é composto por duas placas que desempenham papéis essenciais no processamento e na transmissão dos dados de medição: a placa RVL e a placa de comunicação. A placa RVL é responsável pelo processamento e armazenamento das informações coletadas, enquanto a placa de comunicação, objeto de estudo deste trabalho, realiza o gerenciamento da conexão com a rede de telefonia móvel e a interface entre o servidor SEGECON e a placa RVL, viabilizando o compartilhamento dos dados de medição.

A Figura 1 apresenta o fluxo de comunicação entre o servidor SEGECON e o registrador de energia. Após o estabelecimento da conexão pela rede de telefonia móvel, o servidor envia uma requisição encapsulada em um *frame*, conforme o protocolo da RMS. A placa de comunicação recebe a mensagem, verifica sua

integridade, realiza o desencapsulamento e encaminha a solicitação à placa RVL, que fornece os dados requeridos. Em seguida, a placa de comunicação encapsula a resposta em um novo *frame* e a transmite ao servidor. Dessa forma, a placa de comunicação atua como intermediária entre a RVL e o SEGECON, sendo responsável pelo gerenciamento da comunicação.

Figura 1 – Diagrama da comunicação entre o SEGECON e o registrador de energia



Fonte: Elaboração própria (2026).

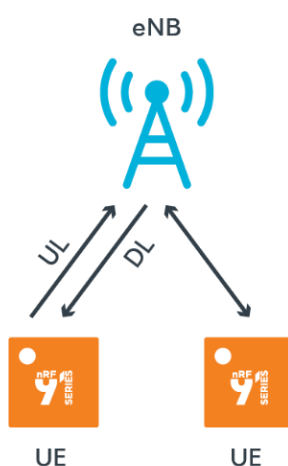
2.2 Comunicação celular

A *Cellular IoT* (*Cellular Internet of Things*) corresponde ao conjunto de tecnologias desenvolvidas pelo 3GPP (*3rd Generation Partnership Project*) para permitir a comunicação de dispositivos IoT por meio das redes celulares já existentes. Entre as principais tecnologias dessa categoria destacam-se o LTE-M (*Long Term Evolution for Machines*) e o NB-IoT (*Narrowband Internet of Things*), ambos desenvolvidos especificamente para aplicações de comunicação entre máquinas (*Machine-Type Communications – MTC*) (Nordic Semiconductor, 2026).

Diferentemente das aplicações voltadas para usuários humanos, como chamadas de voz e acesso à Internet, os dispositivos IoT geralmente transmitem pequenos volumes de dados, apresentam tráfego predominantemente no sentido dispositivo-rede (*uplink*) e podem tolerar maiores atrasos na comunicação. Além disso, uma única estação rádio base deve ser capaz de atender muitos dispositivos simultaneamente. Essas características motivaram o desenvolvimento de tecnologias mais adequadas a esse perfil de aplicação, resultando na criação do LTE-M e do NB-IoT pelo 3GPP (Nordic Semiconductor, 2026).

A arquitetura de uma rede celular pode ser dividida em dois grupos principais: os elementos de acesso à rede e os elementos do núcleo da rede. Os elementos de acesso são compostos pelos equipamentos do usuário (*User Equipment – UE*) e pelas estações rádio base (*Evolved Node B – eNB*). O UE corresponde ao dispositivo final conectado à rede, enquanto o eNB é responsável pela comunicação com os dispositivos localizados em sua área de cobertura. A comunicação entre ambos ocorre por meio de enlaces de envio de dados (*uplink*) e de recepção de dados (*downlink*) (Nordic Semiconductor, 2026). A Figura 2 ilustra a comunicação entre os dispositivos e a estação rádio base.

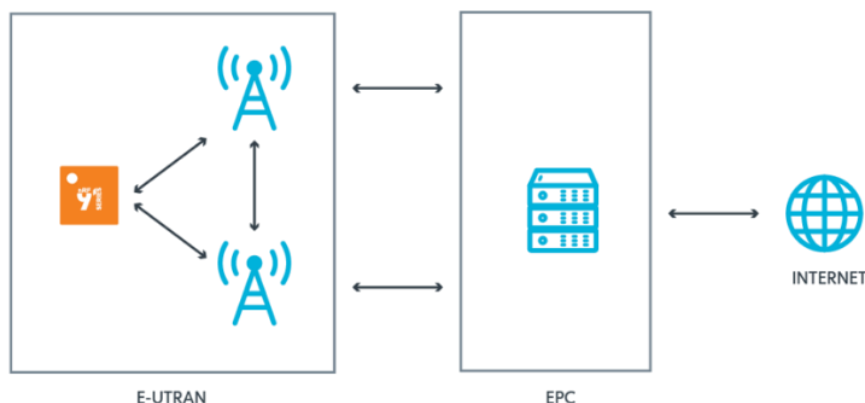
Figura 2 – Comunicação entre UE e eNB



Fonte: Nordic Semiconductor (2026).

Os dispositivos UE e as estações rádio base eNB compõem a E-UTRAN (*Evolved Universal Terrestrial Radio Access Network*), definida como a “tecnologia de comunicação sem fio utilizada nas redes celulares” (Nordic Semiconductor, 2026, tradução nossa). Além de realizar a comunicação entre os dispositivos e as estações rádio base, a E-UTRAN comunica-se com o núcleo da rede celular, denominado EPC (*Evolved Packet Core*), responsável por prover a conexão com a Internet e por gerenciar as conexões dos dispositivos, sendo constituído por diferentes entidades funcionais (Nordic Semiconductor, 2026). A Figura 3 apresenta uma visão simplificada dessa arquitetura, evidenciando a interação entre a E-UTRAN, o EPC e a Internet.

Figura 3 – Arquitetura da rede celular



Fonte: Nordic Semiconductor (2026).

Para que um dispositivo IoT possa acessar uma rede celular, é necessário que possua um cartão SIM (*Subscriber Identity Module*), responsável por armazenar as informações de identificação e autenticação do dispositivo junto à operadora. O SIM possibilita a realização do procedimento de registro na rede, denominado *attach procedure*, no qual o dispositivo se autentica e estabelece conexão com a infraestrutura celular (Nordic Semiconductor, 2026).

2.3 Placa de comunicação

Conforme descrito anteriormente, a placa de comunicação, apresentada na Figura 4, é responsável por intermediar a comunicação entre a placa RVL e o servidor SEGECON. Suas funções incluem o estabelecimento da conexão com a rede de telefonia móvel e o tratamento das mensagens trocadas, realizando o desencapsulamento e o encapsulamento dos dados de acordo com o protocolo de comunicação da RMS.

Figura 4 – Placa de comunicação



Fonte: Elaboração própria (2026).

2.3.1 Arquitetura de *hardware* da placa

A placa de comunicação é composta pelo microcontrolador LPC2368 e pelo modem celular MA510GL, além dos circuitos auxiliares necessários para alimentação, armazenamento e comunicação externa. O LPC2368 atua como unidade central de processamento, sendo responsável pelo gerenciamento das interfaces de comunicação e pela execução do *firmware* embarcado. O modem MA510GL fornece acesso à rede de telefonia móvel, permitindo a comunicação com o servidor SEGECON.

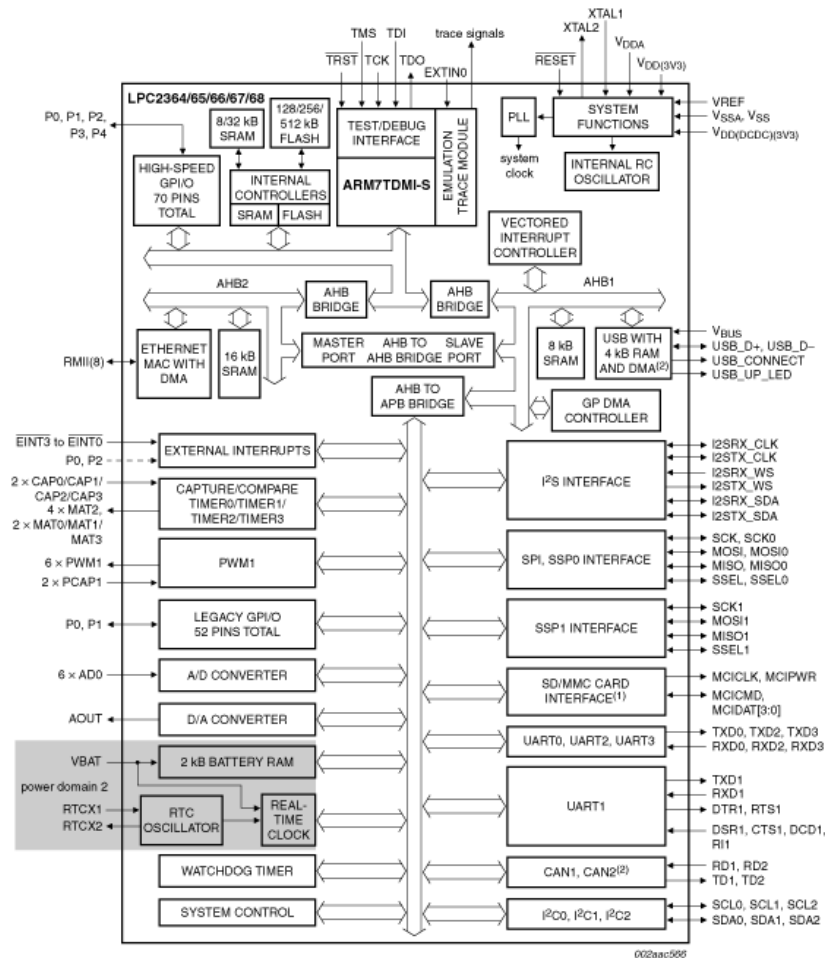
A gravação e a depuração do *firmware* podem ser realizadas por meio dos *pads* de depuração presentes na placa, que fornecem acesso à interface JTAG do microcontrolador. Alternativamente, a gravação pode ser efetuada pela interface USB, mediante um curto-circuito entre os pinos 1 e 5 dos *pads Boot* durante a inicialização, para que o microcontrolador entre no modo de programação.

2.3.2 Microcontrolador LPC2368

O microcontrolador LPC2368, desenvolvido pela NXP *Semiconductors*, é baseado no núcleo ARM7TDMI-S de 32 *bits* e foi projetado para aplicações embarcadas que demandam elevado nível de integração de periféricos e baixo consumo de energia (NXP, 2026). O dispositivo opera com frequência de até 72 MHz e dispõe de até 512 kB de memória *Flash* embarcada e 32 kB de memória SRAM principal disponível ao processador (NXP, 2013).

A Figura 5 apresenta a arquitetura em blocos do microcontrolador LPC2368. A estrutura é organizada em torno do núcleo ARM7TDMI-S e integra, em um único circuito, memórias, controladores de interrupção, módulos de comunicação e diversos periféricos, interligados por diferentes barramentos internos (NXP, 2007).

Figura 5 – Diagrama do LPC2368



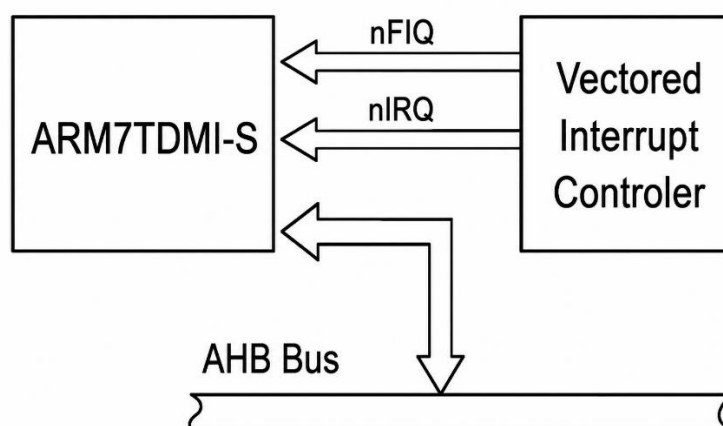
Fonte: NXP (2007, p. 8).

Nas seções seguintes são apresentados os principais blocos utilizados no desenvolvimento da aplicação, com destaque para o controlador de interrupções vetorizadas (VIC), o sistema de *clock* e as interfaces UART.

2.3.2.1 VIC

O *Vectored Interrupt Controller* (VIC) é o bloco responsável pelo controle e gerenciamento das interrupções no microcontrolador LPC2368, atuando como interface entre as fontes de interrupção e o núcleo ARM7TDMI-S. Conforme ilustrado na Figura 6, o núcleo dispõe de duas entradas de interrupção: IRQ (*Interrupt ReQuest*) e FIQ (*Fast Interrupt reQuest*), destinadas, respectivamente, às interrupções de uso geral e de alta prioridade, sendo ambas gerenciadas pelo VIC (Sousa, 2006, p. 43).

Figura 6 – Interligação do VIC com o núcleo



Fonte: Sousa (2026, p. 43).

O VIC possui 32 entradas de interrupção, podendo cada uma delas ser configurada individualmente como IRQ ou FIQ por meio do registrador *VICIntSelect* (NXP, 2007). Em geral, apenas uma fonte é configurada como FIQ, uma vez que esse tipo de interrupção apresenta a menor latência de atendimento (Sousa, 2006, p. 45).

As interrupções configuradas como IRQ utilizam o mecanismo de interrupções vetorizadas implementado pelo VIC. Nesse modo, cada fonte de interrupção pode ser associada ao endereço de sua respectiva rotina de atendimento à interrupção (ISR), além de possuir um nível de prioridade configurável por meio dos registradores *VICVectPriority*. Quando duas ou mais interrupções de mesma prioridade ocorrem simultaneamente, o VIC atende primeiro aquela associada ao

canal de menor numeração, conforme ilustrado na Figura 7, que apresenta o mapeamento entre as fontes de interrupção e os canais do controlador. Durante o atendimento de uma interrupção IRQ, o processador obtém diretamente do registrador *VICVectAddr* o endereço da rotina correspondente, reduzindo o tempo necessário para identificar a origem da interrupção e tornando o tratamento mais eficiente (NXP, 2007).

Figura 7 – Mapeamento das fontes de interrupção nos canais do VIC

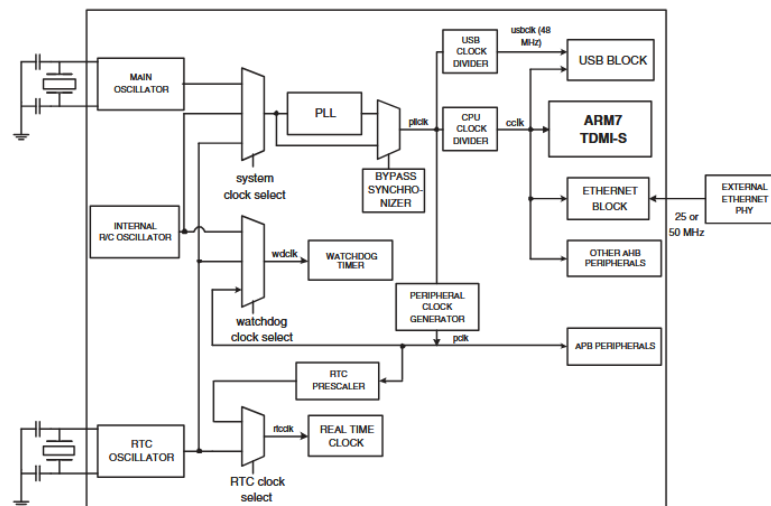
Bit	31	30	29	28	27	26	25	24
Symbol	I2S	I2C2	UART3	UART2	TIMER3	TIMER2	GPDMA	SD/MMC
Bit	23	22	21	20	19	18	17	16
Symbol	CAN1&2	USB	Ethernet	BOD	I2C1	AD0	EINT3	EINT2
Bit	15	14	13	12	11	10	9	8
Symbol	EINT1	EINT0	RTC	PLL	SSP1	SPI/SSP0	I2C0	PWM1
Bit	7	6	5	4	3	2	1	0
Symbol	UART1	UART0	TIMER1	TIMER0	ARMCORE1	ARMCORE0	-	WDT

Fonte: NXP (2007, p. 85).

2.3.2.2 Sistema de clock do LPC2368

O LPC2368 possui um sistema de geração e distribuição de *clock* responsável por fornecer sinais de sincronismo ao núcleo ARM7TDMI-S e aos diferentes periféricos do microcontrolador. Conforme ilustrado na Figura 8, esse sistema é composto por diferentes fontes de *clock*, pelo circuito PLL (*Phase-Locked Loop*), por divisores de frequência e pelos circuitos responsáveis pela distribuição dos *clocks* aos diversos blocos internos do microcontrolador (NXP, 2007).

Figura 8 – Sistema para a geração dos clocks no LPC2368



Fonte: NXP (2007, p. 33).

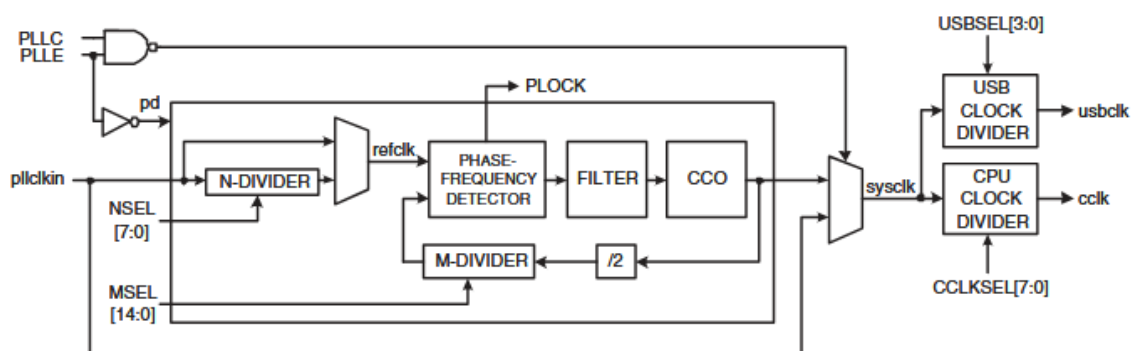
A frequência de operação do sistema pode ser obtida a partir de três fontes de *clock* independentes: o *Main Oscillator*, destinado ao uso de um cristal externo; o *Internal RC Oscillator*, oscilador interno utilizado durante a inicialização do microcontrolador; e o *RTC Oscillator*, dedicado ao funcionamento do relógio de tempo real. O *clock* proveniente do *Main Oscillator*, que opera na faixa de 1 MHz a 24 MHz, pode ser utilizado diretamente pelo processador ou aplicado ao circuito PLL para obtenção de frequências mais elevadas (NXP, 2007). Na placa de comunicação, o *Main Oscillator* recebe o sinal de um cristal de 20 MHz.

2.3.2.2.1 PLL

O circuito PLL é responsável por adequar a frequência proveniente da fonte de *clock* selecionada para atender às necessidades de operação do núcleo ARM7TDMI-S e dos demais periféricos do sistema. O PLL aceita frequências de entrada entre 32 kHz e 50 MHz, realizando internamente operações de divisão e multiplicação desta frequência antes de distribuí-la aos diferentes blocos do microcontrolador (NXP, 2007).

A Figura 9 apresenta o diagrama funcional do PLL implementado no LPC2368. Observa-se que a configuração do circuito é baseada principalmente nos parâmetros N e M, responsáveis, respectivamente, pela divisão da frequência de entrada (pllclkin) e pela multiplicação da frequência interna do PLL (NXP, 2007).

Figura 9 – Diagrama funcional do PLL do LPC2368



Fonte: NXP (2007, p. 38).

A configuração do sistema de *clock* do LPC2368 envolve principalmente os registradores SCS, CLKSRCSEL, PLLCFG, PLLCON, PLLFEED e CCLKCFG. Quando o *Main Oscillator* é utilizado como fonte de *clock*, sua inicialização é realizada por meio do registrador SCS, no qual o bit OSCEN habilita o oscilador e o bit

OSCRANGE define sua faixa de operação (1 MHz a 20 MHz ou 15 MHz a 24 MHz). Após a estabilização do oscilador, indicada pelo *bit* OSCSTAT, a fonte de *clock* utilizada pelo PLL é selecionada por meio do registrador CLKSRCSEL. Os parâmetros de multiplicação (M) e divisão (N) do PLL são configurados no registrador PLLCFG, enquanto a sua habilitação e a conexão ao sistema são controladas pelo registrador PLLCON. Por fim, a frequência aplicada ao núcleo ARM7TDMI-S é determinada pelo divisor configurado no registrador CCLKCFG, enquanto os *clocks* destinados à interface USB e aos demais periféricos são obtidos por divisores independentes, permitindo que cada bloco opere na frequência adequada (NXP, 2007).

2.3.2.3 UART

O LPC2368 possui quatro interfaces UART (*Universal Asynchronous Receiver/Transmitter*) destinadas à comunicação serial assíncrona. As quatro interfaces apresentam funcionamento semelhante, diferenciando-se apenas pela UART1, que disponibiliza sinais adicionais para controle de modem, permitindo a implementação de controle de fluxo por *hardware* (NXP, 2007). De acordo com o esquemático da placa de comunicação, a UART2 é utilizada na comunicação com a placa RVL, enquanto a UART1 é empregada na comunicação com o modem celular MA510GL.

A frequência utilizada pelas UARTs é proveniente do *Peripheral Clock* (PCLK), derivado do sistema de *clock* do microcontrolador. Os registradores PCLKSEL0 e PCLKSEL1 permitem configurar, individualmente para cada periférico, a frequência do PCLK, adequando-a às necessidades da aplicação (NXP, 2007).

Após definida a frequência do PCLK, a taxa de transmissão (*baud rate*) é configurada por meio dos registradores UnDLL, UnDLM e UnFDR. Os registradores UnDLL e UnDLM armazenam o divisor principal da taxa de transmissão, enquanto o registrador UnFDR permite realizar um ajuste fracionário por meio dos parâmetros MULVAL e DIVADDVAL, possibilitando a obtenção de uma ampla faixa de taxas de transmissão utilizando uma mesma frequência de entrada (NXP, 2007). A escrita nos registradores UnDLL e UnDLM somente é permitida quando o bit DLAB, localizado no registrador UnLCR, estiver em nível lógico um. Esse registrador também é responsável pela configuração do formato do quadro de comunicação, incluindo o

número de bits de dados, bits de parada e paridade (Sousa, 2006, p. 72). A taxa de transmissão das UARTs é determinada pela Equação (1), apresentada no *datasheet* da NXP (2007, p. 337).

$$UARTn_{baudrate} = \frac{PCLK}{16 \times UnDL} \times \frac{MULVAL}{(MULVAL + DIVADDVAL)} \quad (1)$$

Em que:

- PCLK é o *clock* do periférico;
- UnDL é o valor determinado pelos registradores UnDLM e UnDLL (UnDL = 256 x UnDLM + UnDLL);
- DIVADDVAL e MULVAL são os parâmetros específicos do gerador fracionário de *baud rate*, definidos no registrador UnFDR.

As UARTs possuem *buffers* FIFO (*First In, First Out*) independentes de transmissão e recepção com capacidade para 16 *bytes*. O funcionamento desses *buffers* é controlado pelo registrador UnFCR, no qual também é definido o valor de *trigger* para a geração da interrupção de recepção, podendo ser configurado para 1, 4, 8 ou 14 *bytes* recebidos. Os dados recebidos são lidos pelo registrador UnRBR, enquanto os dados a serem transmitidos são escritos no registrador UnTHR (Sousa, 2006, p. 73).

As interrupções da UART são identificadas por meio do registrador UnIIR, que informa a origem da interrupção pendente, como recepção de dados, esvaziamento do *buffer* de transmissão ou condições de erro na linha de comunicação (NXP, 2007). Esse mecanismo permite que a comunicação serial seja realizada de forma orientada a interrupções, reduzindo a necessidade de monitoramento contínuo pelo processador.

2.3.3 Modem MA510GL

O módulo MA510GL, desenvolvido pela *Fibocom*, é um módulo de comunicação celular de alta performance destinado a aplicações IoT. Baseado na

plataforma *Qualcomm* e compatível com a especificação 3GPP *Release 14*, o dispositivo suporta as tecnologias LTE-M (Cat.M1), NB-IoT (Cat.NB1/NB2) e GPRS, permitindo comunicação em diferentes redes móveis. O módulo também oferece suporte opcional a sistemas globais de navegação por satélite (GNSS) e incorpora interfaces industriais, como UART, USB, I²C e I²S, além de protocolos de comunicação voltados à IoT, como MQTT, CoAP e LwM2M (Fibocom, 2026).

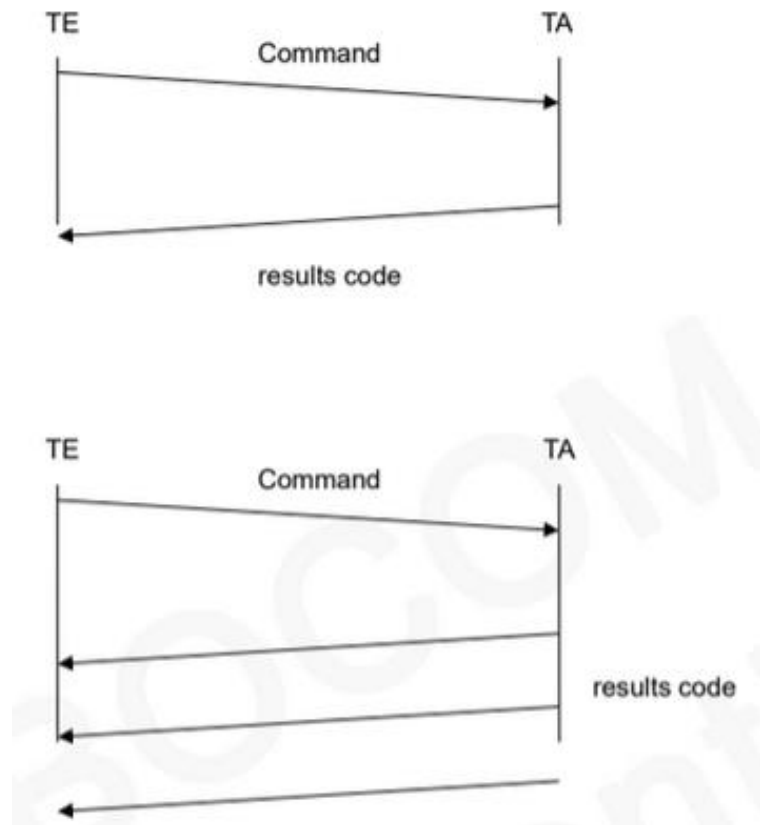
O modem MA510GL é responsável por estabelecer a comunicação entre a placa de comunicação e a rede celular, permitindo a troca de informações com o servidor SEGECON. A comunicação entre o LPC2368 e o módulo é realizada com comandos AT (*Attention*) por meio da UART1.

2.3.3.1 Comandos AT

Segundo a *Nordic Semiconductor* (2026, tradução nossa), “os comandos AT são a interface de *software* utilizada para controlar o modem [celular], definida como parte do padrão 3GPP”. Esses comandos são transmitidos em formato ASCII e iniciam com o prefixo AT, indicando ao modem que uma nova solicitação será enviada. O término de cada comando é sinalizado pelo caractere *Carriage Return* (CR), enquanto as respostas retornadas pelo modem utilizam, por padrão, a sequência *Carriage Return/Line Feed* (CR/LF) tanto no início quanto no final de cada mensagem (Fibocom, 2024).

A Figura 10 ilustra a comunicação entre o equipamento terminal (TE), representado pelo LPC2368, e o modem celular (TA), representado pelo MA510GL. De maneira geral, toda comunicação é iniciada pelo TE, que envia comandos AT ao modem. Em resposta, o modem retorna um ou mais códigos de resultado (*results code*), cuja finalidade é informar o resultado da operação ao TE (Fibocom, 2024).

Figura 10 – Fluxo de comunicação entre TA e TE durante a execução de comandos AT



Fonte: Fibocom (2024, p. 15).

Além das respostas geradas em decorrência dos comandos enviados pelo TE, o modem também pode transmitir mensagens assíncronas conhecidas como URC (*Unsolicited Result Codes*), conforme ilustrado na Figura 11. Diferentemente dos *results code*, essas mensagens não estão associadas a um comando previamente enviado, sendo utilizadas para informar eventos espontâneos, como registro na rede, perda de conexão, recebimento de dados ou alterações no estado da comunicação (Fibocom, 2024).

Figura 11 – Transmissão de URC pelo modem



Fonte: Fibocom (2024, p. 15).

A Figura 12 apresenta a sintaxe dos comandos AT suportados pelo MA510GL, sendo em “xxx” onde é informado o comando desejado e em “<Value>” os parâmetros de configuração. De acordo com a documentação da Fibocom (2024), os comandos podem ser classificados em quatro categorias: *Execute Commands*, utilizados para executar uma função específica; *Parameter Set Commands*, empregados para configurar parâmetros de operação; *Parameter Read Commands*, utilizados para consultar o valor atual de um parâmetro; e *Parameter Test Commands*, que permitem verificar os parâmetros e valores suportados por um determinado comando.

Figura 12 – Sintaxe comandos AT

Execute Command Syntax	AT+xxx ATxxx ATxxx;
Parameter Set Command Syntax	AT+xxx=<Value> ATxxx=<Value>
Parameter Read Command Syntax	AT+xxx? ATxxx?
Parameter Test Command Syntax	AT+xxx=? ATxxx=?

Fonte: Fibocom (2024, p. 17).

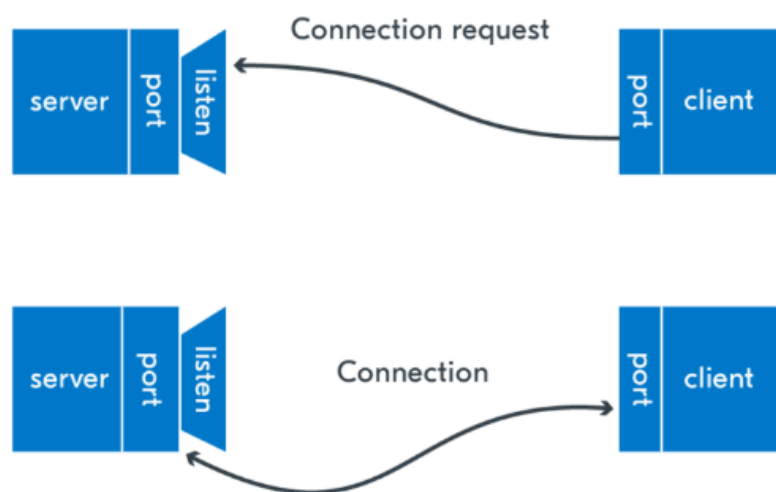
2.3.3.2 Comunicação por meio de sockets TCP

Apesar de o módulo MA510GL possuir compatibilidade com diversos protocolos para aplicações IoT, a comunicação entre a placa de comunicação e o servidor SEGECON foi implementada utilizando *sockets* TCP.

Segundo a *Nordic Semiconductor* (2026, tradução nossa), “um *socket* é uma estrutura de *software* que serve como um ponto final para enviar e receber dados em uma rede”. Os *sockets* são empregados no modelo cliente-servidor, no qual um dispositivo atua como servidor, permanecendo em estado de escuta (*listen*) em uma porta previamente definida, enquanto outro dispositivo atua como cliente, iniciando a conexão por meio do endereço IP e da porta do servidor. Conforme ilustrado na Figura

13, o cliente envia inicialmente uma requisição de conexão ao servidor. Após sua aceitação, estabelece-se um canal de comunicação bidirecional entre os dois dispositivos, permitindo que ambos enviem e recebam dados por meio de seus respectivos *sockets* (Nordic Semiconductor, 2026).

Figura 13 – Estabelecimento de uma conexão utilizando *sockets* no modelo cliente-servidor



Fonte: Nordic Semiconductor (2026).

2.3.3.3 Principais comandos utilizados

O estabelecimento da comunicação entre a placa de comunicação e o servidor SEGECON requer a utilização de um conjunto de comandos AT responsáveis pela configuração do modem MA510GL, pelo gerenciamento da conexão com a rede celular e pela comunicação de dados. O Quadro 1 apresenta os principais comandos utilizados para essas funcionalidades, bem como sua descrição e finalidade.

Quadro 1 – Comandos AT utilizados para a conexão com a rede celular e a comunicação com o servidor

Comando	Descrição	Finalidade
AT	Verificar comunicação com o modem	Confirmar a comunicação com o modem
AT+CPIN	Inserir PIN do SIM	Configurar senha do cartão SIM
AT+CSQ	Intensidade do sinal	Verificar a intensidade do sinal recebido e a taxa de erro de <i>bits</i>

AT+CREG	Estado de registro na rede	Verificar o registro do modem na rede celular
AT+CGDCONT	Definir contexto PDP	Configurar APN e parâmetros de acesso à rede de dados
AT+MIPCALL	Estabelecer conexão de dados	Obter um endereço IP válido na rede celular
AT+MIOPEN	Abrir um <i>socket</i> TCP	Criar uma conexão TCP com o servidor
AT+MIPSEND	Armazenar dados no <i>buffer</i> de transmissão	Preparar os dados para envio
AT+MIPUSH	Enviar dados para a pilha de protocolos	Efetivar a transmissão dos dados ao servidor
+MIPRTCP	Receber dados do <i>socket</i> TCP	Indicar o recebimento de dados do servidor
AT+MIPCLOSE	Fechar <i>socket</i> TCP	Finalizar a conexão com o servidor
+MIPSTAT	Indicar o estado da conexão	Informar alterações no estado da comunicação TCP

Fonte: Adaptado de Fibocom (2024).

Os comandos +MIPRTCP e +MIPSTAT correspondem a URCs emitidas pelo modem para indicar, respectivamente, o recebimento de dados pelo *socket* TCP e alterações no estado da conexão. Em conjunto com os demais comandos apresentados, possibilitam a realização das etapas necessárias para o funcionamento da aplicação, incluindo a configuração inicial do modem, o registro na rede celular, o estabelecimento da conexão de dados, a criação do *socket* TCP, a troca de informações com o servidor e o encerramento da conexão.

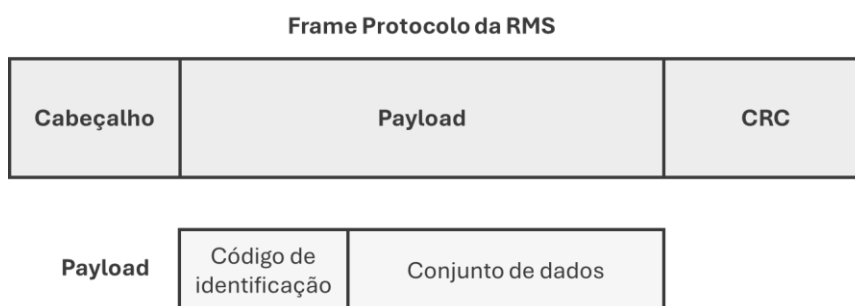
2.4 Protocolo de comunicação da RMS

O *frame* do protocolo de comunicação da RMS, apresentado na Figura 14, é composto por três campos principais: cabeçalho, *payload* e CRC (*Cyclic Redundancy Check*). O cabeçalho contém informações de sincronização e controle da mensagem, enquanto o *payload* armazena os dados de solicitação ou resposta. O campo CRC, por sua vez, contém o código de detecção de erros utilizado para validar a integridade dos dados recebidos. Todos os dados são transmitidos em formato

binário, sendo apresentados em notação hexadecimal (BRS Sistemas Eletrônicos, 2015).

O *payload* é formado por um código de identificação e por um conjunto de dados. Nas mensagens de requisição, seu tamanho é fixo em 66 *bytes* e seu conteúdo é definido pelo protocolo, de modo que cada código de identificação corresponde a uma solicitação específica. Nas mensagens de resposta, o tamanho do *payload* depende das informações solicitadas, podendo variar entre 258 e 1002 *bytes*. Por questões de confidencialidade industrial, o conteúdo específico dos campos e dos dados transportados não será detalhado neste trabalho.

Figura 14 – Frame do protocolo de comunicação da RMS



Fonte: Elaboração própria (2026).

O Quadro 2 apresenta a sequência de conjuntos de dados solicitados pelo servidor SEGECON, indicando o respectivo código de identificação e o tamanho da mensagem de resposta associada a cada conjunto. O protocolo de comunicação segue um modelo de requisição e resposta, no qual o servidor solicita um conjunto de dados específico e o registrador retorna as informações correspondentes. Para a maioria dos conjuntos de dados, uma única requisição é suficiente para a obtenção da resposta. A exceção é o bloco MM (Memória de Massa), que correspondem aos registros de medição armazenados no equipamento. Como o registrador pode armazenar uma quantidade significativa de registros, os dados são transmitidos em múltiplos blocos. Após o recebimento de cada bloco MM, o servidor envia uma mensagem de confirmação (ACK), identificada pelo código 06, indicando que os dados foram recebidos corretamente e autorizando a transmissão do próximo bloco. Esse processo se repete até que todos os registros armazenados sejam transferidos. Por fim, o conjunto de dados identificado pelo código 2C é exclusivo dos registradores do modelo MARH-28M, não sendo utilizado pelos demais equipamentos da RMS.

Quadro 2 – Sequência de conjuntos de dados solicitados pelo SEGECON

- Bloco Alterações (28), 258 bytes
- Bloco Faltas de Energia (25), 258 bytes
- Bloco Faltas de Energia DIC 1 (2A), 258 bytes
- Bloco Faltas de Energia DIC 2 (2A), 258 bytes
- Bloco Faltas de Energia DIC 3 (2A), 258 bytes
- Bloco Parâmetros (21), 258 bytes
- Blocos MM (26), 1002 bytes
- ...
- Bloco EVENTOS Frequência (2b), 787 bytes
- Bloco EVENTOS Frequência (2c), 882 bytes

Fonte: Elaboração própria (2026).

O protocolo da RMS também prevê uma mensagem padrão denominada KEEP-ALIVE, transmitida a cada 60 segundos na ausência de troca de dados. Sua finalidade é manter ativa a conexão e auxiliar na detecção de perdas de comunicação. Caso não ocorra nenhuma transmissão durante 120 segundos, a conexão deve ser encerrada e restabelecida. O *frame* KEEP-ALIVE não gera resposta do destinatário e pode ser enviado tanto pelo servidor quanto pela placa de comunicação, sendo esta última a responsável pela maior parte das transmissões dessa mensagem durante a operação do sistema (BRS Sistemas Eletrônicos, 2015).

2.5 Sistemas operacionais de tempo real

Em sistemas embarcados de baixa complexidade, é comum a utilização da abordagem *foreground/background*, também conhecida como superlaço (*superloop*), na qual um laço infinito executa sequencialmente as tarefas da aplicação, enquanto as interrupções tratam eventos síncronos e assíncronos. Embora simples de implementar, essa abordagem apresenta limitações quanto à previsibilidade temporal e à manutenção do sistema à medida que sua complexidade aumenta. Nesse modelo, as tarefas executadas no contexto do superlaço não possuem mecanismos de priorização, de modo que tarefas críticas podem ter sua execução atrasada por atividades menos importantes. Além disso, o gerenciamento de atividades

concorrentes torna-se mais complexo à medida que novas funcionalidades são incorporadas ao sistema (Denardin; Barriquello, 2019).

Como alternativa, podem ser empregados RTOS (*Real-Time Operating System*), que fornecem mecanismos de escalonamento, sincronização e comunicação entre tarefas, permitindo um gerenciamento mais eficiente dos recursos do sistema. Entre suas principais características destacam-se o comportamento determinístico, a multitarefa, o tratamento eficiente de interrupções e o escalonamento por prioridade. Esses sistemas também disponibilizam mecanismos de gerenciamento de memória e de alocação de recursos computacionais, reduzindo interferências entre tarefas e contribuindo para a estabilidade da aplicação. Além disso, sua arquitetura favorece a escalabilidade do sistema, possibilitando a inclusão de novas funcionalidades sem comprometer o desempenho das tarefas já existentes (IBM, 2026).

Existem diversos RTOS disponíveis para sistemas embarcados. Neste trabalho, foi adotado o RL-RTX, desenvolvido pela KEIL, devido à sua integração com a plataforma de desenvolvimento *μVision* e à compatibilidade com dispositivos baseados na arquitetura ARM7, a mesma utilizada pelo LPC2368. Em razão da disponibilidade de exemplos e bibliotecas consolidados para esse microcontrolador, o RL-RTX foi escolhido como base para o desenvolvimento do *firmware* proposto.

2.5.1 RL-RTX

O RL-RTX (*Real-Time Library - Real Time eXecutive*) é um sistema operacional de tempo real desenvolvido para aplicações embarcadas baseadas em microcontroladores ARM. Seu núcleo (*kernel*) fornece um conjunto de funções para criação, gerenciamento e escalonamento de tarefas, permitindo que diferentes atividades sejam executadas de forma concorrente e coordenada (KEIL, 2026).

O *kernel* é responsável pela troca de contexto entre tarefas e pelo gerenciamento de sua execução, podendo utilizar escalonamento preemptivo baseado em prioridades (KEIL, 2026). Dessa forma, tarefas mais críticas podem receber preferência de processamento, contribuindo para o atendimento dos requisitos temporais da aplicação.

Além do gerenciamento de tarefas, o RL-RTX disponibiliza mecanismos de comunicação e sincronização entre processos, permitindo o compartilhamento seguro de informações e recursos. Entre os recursos fornecidos pelo sistema destacam-se o *event flags*, *memory pools*, *mailboxes* e *user timers*, amplamente utilizados em aplicações multitarefa para sincronização de eventos e troca de dados entre tarefas (KEIL, 2026).

A utilização desses mecanismos contribui para a modularização do *software*, reduzindo sua complexidade e facilitando atividades de manutenção, testes e expansão futura da aplicação. Além disso, o RL-RTX oferece recursos de temporização e tratamento eficiente de interrupções, características importantes para sistemas embarcados que necessitam operar de forma previsível e confiável (KEIL, 2026).

2.5.2 Recursos utilizados do RL-RTX

O desenvolvimento do *firmware* fez uso de alguns dos recursos disponibilizados pelo RL-RTX para atender aos requisitos de comunicação, sincronização e temporização da aplicação. A seguir, são apresentados os principais mecanismos empregados, bem como suas respectivas funções dentro da arquitetura desenvolvida.

2.5.2.1 Event Flags

As *Event Flags* são mecanismos de sincronização utilizados para sinalizar a ocorrência de eventos entre tarefas e rotinas de interrupção. No RL-RTX, cada tarefa possui um conjunto de 16 *flags* que podem ser utilizadas para indicar diferentes eventos. Uma tarefa pode permanecer bloqueada aguardando uma ou mais *flags* e ser liberada para execução quando a condição esperada for atendida. O sistema permite que a tarefa aguarde simultaneamente múltiplos eventos, exigindo a ocorrência de todos eles (*AND*) ou de qualquer um deles (*OR*) para prosseguir sua execução (KEIL, 2026).

As *Event Flags* podem ser acionadas tanto por outras tarefas quanto por rotinas de interrupção, permitindo que tarefas sejam notificadas sobre a ocorrência de

eventos assíncronos. Dessa forma, uma rotina de interrupção pode apenas sinalizar a ocorrência de um evento por meio de uma *flag*, deixando o processamento mais complexo para uma tarefa específica. Essa abordagem contribui para reduzir o tempo de execução das rotinas de interrupção e melhorar a organização do sistema (KEIL, 2026).

2.5.2.2 *Memory Pool*

As *Memory Pools* são áreas de memória previamente alocadas e divididas em blocos de tamanho fixo, que podem ser requisitados e liberados pelas tarefas durante a execução do sistema. O gerenciamento desses blocos é realizado pelo próprio RL-RTX, permitindo a alocação dinâmica de memória de forma organizada e adequada a aplicações multitarefa. Esse mecanismo pode ser utilizado em conjunto com as *Mailboxes*, fornecendo os blocos de memória destinados ao armazenamento das mensagens trocadas entre tarefas (KEIL, 2026).

2.5.2.3 *Mailboxes*

As *Mailboxes* são mecanismos de comunicação que permitem o envio e o recebimento de mensagens entre tarefas. No RL-RTX, as mensagens são trocadas por meio de ponteiros para blocos de memória previamente alocados em uma *Memory Pool*, os quais armazenam os dados a serem compartilhados. Dessa forma, em vez de transferir diretamente o conteúdo da mensagem, a *Mailbox* transmite apenas o endereço do bloco de memória correspondente, tornando a comunicação mais eficiente, especialmente para mensagens de maior tamanho. Caso não exista nenhuma mensagem disponível, a tarefa receptora pode permanecer bloqueada até que uma nova mensagem seja enviada, sendo reativada pelo kernel quando a mensagem estiver disponível (KEIL, 2026).

2.5.2.4 *User Timer*

Os *User Timers* são temporizadores de *software* disponibilizados pelo RL-RTX para a execução de ações após um intervalo de tempo previamente definido.

Esses temporizadores operam no modo *single-shot* (disparo único), ou seja, realizam apenas uma contagem regressiva e são automaticamente removidos após expirarem, não sendo possível pausá-los ou reiniciá-los. Durante a execução do sistema, podem ser criados e eliminados dinamicamente pelas tarefas conforme a necessidade da aplicação (KEIL, 2026).

Ao criar um *User Timer*, são definidos um tempo de expiração e um parâmetro associado ao temporizador. Quando o tempo configurado é atingido, o *kernel* executa uma função de retorno (*callback*), fornecendo o parâmetro previamente informado. Esse parâmetro pode ser utilizado para identificar qual temporizador expirou e, conseqüentemente, determinar a ação a ser executada. Caso o temporizador seja removido antes do término da contagem, a função de retorno não é executada (KEIL, 2026).

Portanto, esse recurso é particularmente útil na implementação de mecanismos de temporização, como o controle de *timeouts* em aplicações embarcadas.

2.6 Resumo

O presente capítulo apresentou a fundamentação teórica necessária para o desenvolvimento do *firmware* proposto. Inicialmente, foi descrita a arquitetura do sistema, abordando o servidor SEGECON, o registrador de energia — composto pela placa RVL e pela placa de comunicação — e a rede celular, de modo a compreender a interação entre esses elementos e seus respectivos papéis no processo de transmissão dos dados de medição.

Em seguida, foram apresentados os conceitos relacionados à comunicação celular, contemplando os principais mecanismos envolvidos no estabelecimento da conexão com a rede móvel e na comunicação de dados com o servidor. Posteriormente, foi introduzida a placa de comunicação, objeto de estudo deste trabalho, destacando seus principais componentes: o microcontrolador LPC2368 e o modem MA510GL. A partir desse estudo, foram identificados os módulos internos do microcontrolador necessários para a implementação da aplicação, tais como o controlador de interrupções vetorizadas (VIC), o sistema de geração de *clock* e as

interfaces UART. Além disso, foram apresentados os comandos AT empregados na configuração do modem e no processo de comunicação com o servidor a partir da abertura do *socket* TCP.

Também foi contextualizado o protocolo de comunicação RMS, descrevendo a estrutura dos *frames* de comunicação e aspectos relevantes para o desenvolvimento do *firmware*, como o tamanho variável do *payload* e o envio periódico do *frame* KEEP-ALIVE em intervalos de um minuto na ausência de troca de dados.

Por fim, foram abordados os conceitos relacionados aos sistemas operacionais de tempo real (RTOS), com ênfase no RL-RTX e nos recursos empregados na aplicação, como *Event Flags*, *Memory Pools*, *Mailboxes* e *User Timers*.

A fundamentação teórica apresentada estabelece a base necessária para a compreensão das decisões de projeto, da metodologia adotada e das etapas de desenvolvimento do *firmware* descritas no capítulo seguinte.

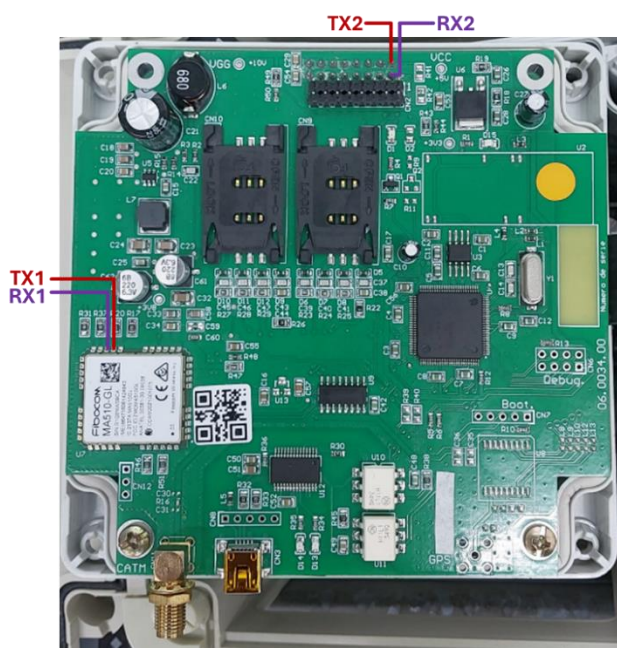
3 METODOLOGIA

O desenvolvimento do *firmware* da placa de comunicação foi conduzido em cinco etapas principais: obtenção dos *logs* de comunicação, análise da comunicação com o modem MA510GL, análise da comunicação com a placa RVL, configuração do ambiente de desenvolvimento e implementação do *firmware*.

3.1 Obtendo os *logs* da comunicação

A primeira etapa consistiu na obtenção dos *logs* de comunicação da placa original em funcionamento, com o objetivo de compreender o fluxo de mensagens trocadas entre os diferentes elementos do sistema. Para a captura das mensagens do modem MA510GL, foi utilizado o programa Terminal configurado para comunicação em formato ASCII e taxa de transmissão de 115200 bps. Já para a comunicação com a placa RVL, o monitoramento foi realizado em formato binário, com visualização em hexadecimal, e taxa de transmissão de 9600 bps. Os *logs* foram obtidos a partir dos sinais TX e RX das UARTs do microcontrolador, cujos pontos de acesso são apresentados na Figura 15. A UART1 é utilizada na comunicação com o modem MA510GL, enquanto a UART2 é empregada na comunicação com a placa RVL.

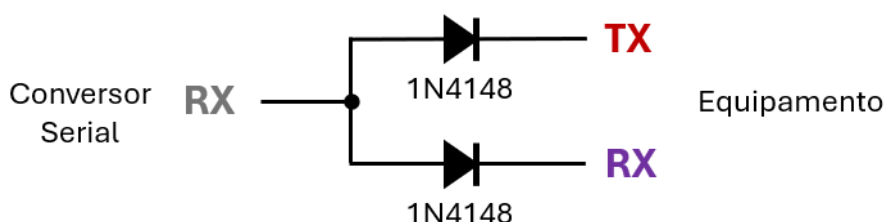
Figura 15 – Pontos de acesso ao TX e RX da UART1 e UART2



Fonte: Elaboração própria (2026).

Para possibilitar a captura simultânea dos dados transmitidos e recebidos pelas UARTs, foi utilizado o circuito apresentado na Figura 16. Os sinais TX e RX da interface serial monitorada são obtidos nos pontos indicados na Figura 15 e conectados às entradas do circuito, que combina ambos os sinais em um único ponto de monitoramento por meio de diodos, sendo utilizado o modelo 1N4148. Esse ponto é então conectado a um conversor serial ligado ao computador. Dessa forma, torna-se possível registrar, em uma única captura, tanto as mensagens transmitidas quanto as recebidas, permitindo a análise completa da comunicação entre os dispositivos.

Figura 16 – Circuito para obter os dados de TX e RX simultaneamente



Fonte: Elaboração própria (2026).

Adicionalmente, o programa Terminal foi configurado para registrar as mensagens em um arquivo contendo o *timestamp* de cada evento. Dessa forma, foi possível registrar cronologicamente os comandos e mensagens trocados pela placa de comunicação, desde o estabelecimento da conexão com a rede de telefonia móvel até a troca de dados entre o servidor SEGECON e o registrador de energia. A análise desses registros permitiu compreender a sequência de eventos e o fluxo de comunicação do sistema.

3.1.1 Comunicação com o módulo MA510GL

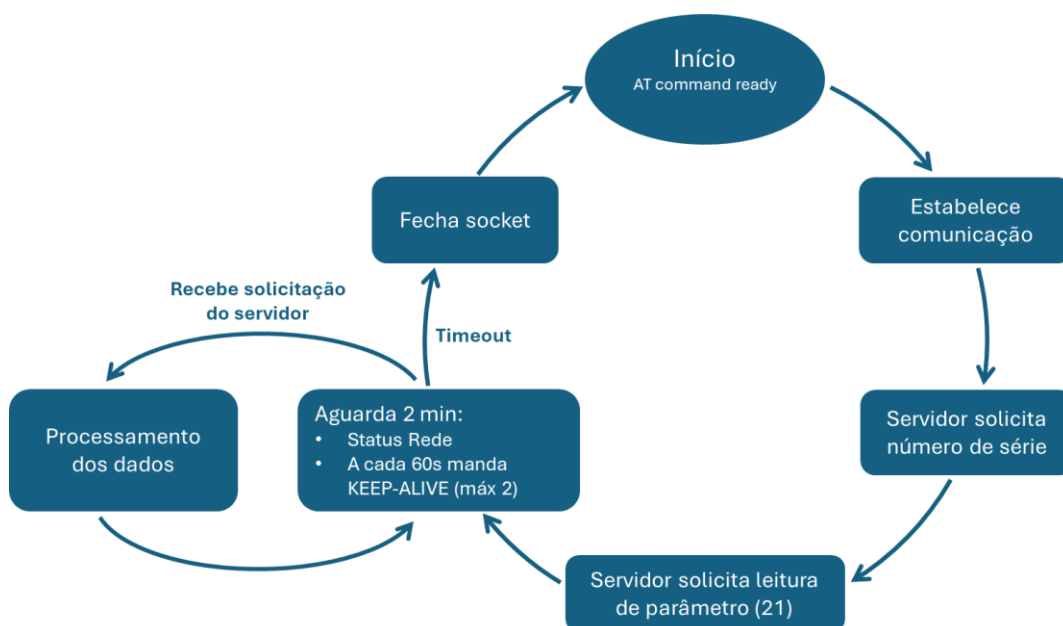
A análise dos *logs* de comunicação com o módulo MA510GL permitiu identificar o ciclo de operação apresentado na Figura 17. O processo inicia quando o modem envia a mensagem "*AT command ready*", indicando sua disponibilidade. Em seguida, o microcontrolador realiza o registro na rede de telefonia móvel e estabelece a comunicação com o servidor SEGECON por meio da abertura de um *socket* TCP.

Após a conexão ser estabelecida, o servidor solicita o número de série do equipamento e, uma vez recebida a resposta, requisita o conjunto de dados de código

21, referente aos parâmetros do registrador, conforme definido pelo protocolo da RMS. A partir desse momento, a placa realiza verificações periódicas do estado da conexão com a rede móvel enquanto aguarda novas requisições do servidor.

Na ausência de novas solicitações, uma mensagem KEEP-ALIVE é enviada a cada 60 segundos para manter o *socket* ativo. Caso não haja comunicação com o servidor após o envio de dois KEEP-ALIVE consecutivos, o *socket* é encerrado e o processo de conexão é reiniciado.

Figura 17 – Processos realizados pela placa de comunicação



Fonte: Elaboração própria (2026).

3.1.1.1 Troca de dados entre o servidor e o registrador

O Quadro 3 apresenta um exemplo de troca de mensagens entre o servidor SEGECON e o registrador de energia. O comando +MIPRTCP indica o recebimento de dados pela pilha TCP do *socket* configurado, correspondendo à solicitação enviada pelo servidor. Por questões de confidencialidade industrial, o significado dos campos presentes na mensagem não será detalhado neste trabalho.

A resposta do registrador é transmitida por meio do comando AT+MIPSEND. Em seguida, o comando AT+MIPPUSH é utilizado para transferir os dados do *buffer* acumulativo para a pilha TCP, efetivando o envio da mensagem ao servidor.

A obtenção do número de série é fundamental para o estabelecimento da comunicação com o servidor SEGECON, uma vez que essa é a primeira informação solicitada pelo servidor após a abertura da conexão.

Compreendido o fluxo de comunicação entre os elementos do sistema, a etapa seguinte consistiu no desenvolvimento do novo *firmware* para a placa de comunicação, de modo a atender aos requisitos de funcionamento identificados e aos objetivos propostos neste trabalho.

3.2 Configuração do ambiente de desenvolvimento

O ambiente de desenvolvimento integrado (IDE) escolhido para o desenvolvimento do *firmware* foi o *µVision*, da *Keil*. A escolha deve-se principalmente à disponibilidade de suporte nativo ao microcontrolador LPC2368, incluindo arquivos de configuração e inicialização específicos para o dispositivo. Ao selecionar o microcontrolador no ambiente, é disponibilizado automaticamente o arquivo *Startup.s*, desenvolvido em linguagem *assembly*, responsável pela inicialização de baixo nível necessária para a execução da aplicação. Por se tratar de um microcontrolador baseado em uma arquitetura ARM mais antiga, a utilização do *µVision* simplificou o início do desenvolvimento, além de disponibilizar exemplos de código e ferramentas de depuração compatíveis com o dispositivo.

A Figura 18 apresenta o *hardware* utilizado no projeto. A placa de comunicação foi conectada ao registrador de energia e ao adaptador de depuração ULINK2, empregado na gravação, simulação e depuração do *firmware* por meio da interface JTAG. Adicionalmente, foram realizadas modificações na placa para permitir a manipulação das UARTs utilizadas durante os testes. Essas modificações consistiram na inserção de *jumpers* entre as trilhas de comunicação, possibilitando a seleção entre o modo de operação normal e o modo de emulação, no qual um *terminal* serial pode assumir o papel da placa RVL ou do módulo MA510GL.

Figura 18 – Hardware implementado para o desenvolvimento do *firmware*



Fonte: Elaboração própria (2026).

3.3 Desenvolvimento do *firmware*

Inicialmente, o *firmware* foi desenvolvido utilizando uma arquitetura sequencial. Entretanto, visando uma melhor organização do *software* e a divisão das funcionalidades em tarefas independentes, optou-se pela adoção de um sistema operacional de tempo real (RTOS).

Definida a utilização de um RTOS, a primeira alternativa avaliada foi o FreeRTOS. No entanto, o exemplo disponível encontra-se desatualizado e sem suporte, demandando adaptações em arquivos de inicialização e trechos de código em linguagem *assembly* para sua utilização. Em razão dessas dificuldades, foram analisados os exemplos fornecidos pela IDE *μVision*, entre os quais se encontravam aplicações baseadas no RL-RTX.

Por apresentar integração nativa com a IDE *μVision* e exemplos compatíveis com o microcontrolador LPC2368, o RL-RTX foi validado e adotado como base para o desenvolvimento do *firmware*.

3.3.1 Ajustes no arquivo *Startup.s*

O projeto de exemplo fornecido pela IDE considera a utilização de um cristal de 12 MHz como fonte de *clock* principal. Entretanto, a placa de comunicação utiliza um cristal de 20 MHz, tornando necessária a adequação da configuração de *clock* do microcontrolador. Essa configuração é fundamental, pois define as frequências de operação da CPU e dos periféricos internos.

Para realizar essa adequação, foram ajustados os registradores SCS, PLLCFG e CCLKCFG. A configuração adotada buscou obter uma frequência de CPU próxima ao limite de 72 MHz, especificado no *datasheet* do LPC2368, e um *clock* dos periféricos (PCLK) de 16 MHz, facilitando os cálculos necessários para a configuração do *baud rate* das UARTs. Com as alterações realizadas, a CPU passou a operar em 64 MHz e, mantendo-se a configuração padrão dos divisores de *clock* dos periféricos, as UARTs passaram a operar com PCLK de 16 MHz. O Quadro 6 apresenta as modificações realizadas no arquivo *Startup.s*, destacadas pelas setas azuis.

Quadro 6 – Modificação no arquivo *Startup.s*

→	SCS_Val	EQU	0x00000030
→	CLKSRCSEL_Val	EQU	0x00000001
→	PLLCFG_Val	EQU	0x00000007
→	CCLKCFG_Val	EQU	0x00000004

Fonte: Elaboração própria (2026).

3.3.2 Configuração das UARTs

O *µVision* não disponibiliza uma interface gráfica para configuração dos periféricos, tornando necessária a manipulação direta dos registradores do microcontrolador. O Quadro 7 apresenta a implementação desenvolvida para a configuração da UART1, operando com interrupções e *baud rate* de 115200 bps, baseada no código apresentado por Dibal (2012, p. 61).

Inicialmente, os registradores PINSEL0 e PINSEL1 são configurados para atribuir aos pinos do microcontrolador as funções de transmissão (TX) e recepção (RX). Em seguida, é realizada a configuração do controlador de interrupções, definindo a função responsável pelo tratamento das interrupções da UART1, bem

como o tipo e a prioridade da interrupção. Posteriormente, são configurados os parâmetros de comunicação da UART. Os registradores U1FDR, U1DLL e U1DLM são utilizados na definição da taxa de transmissão, enquanto o registrador U1LCR estabelece o formato da mensagem, configurando a comunicação com 8 bits de dados, sem paridade e 1 bit de parada. Por fim, os registradores U1IER e U1FCR são configurados para habilitar as interrupções da UART1. A configuração adotada gera interrupções a cada *byte* recebido e na ocorrência de erros de comunicação, possibilitando o tratamento assíncrono dos dados. A mesma estrutura pode ser empregada na configuração das demais UARTs do microcontrolador, bastando utilizar os registradores correspondentes e ajustar os parâmetros desejados.

Quadro 7 – Código de inicialização UART1

```

/***** Config UART1 *****/
PINSEL0 &= ~0xC0000000;
PINSEL0 |= 0x40000000; /* Enable TxD1 */
PINSEL1 &= ~0x00000003;
PINSEL1 |= 0x00000001; /* Enable RxD1 */

U1IER = 0; /* Disable UART0 Interrupts */
VICVectAddr7 = (uint32_t) USART1_IRQHandler; /* Set Interrupt Vector */
VICIntSelect = 0x00000000; /* interrupt request assigned to IRQ */
VICVectPriority7 = 1; /* Define priority to interrupt */
VICIntEnable = (1 << 7); /* Enable Interrupt */

/* 15200 Baud Rate @ 16MHz PCLK Clock */
U1LCR = 0x83; /* 8 bits, no Parity, 1 Stop bit and DLAB = 1 */
U1FDR = 0xB1; /* Set fractional divider register */
U1DLL = 0x08; /* Set divisor latch LSB */
U1DLM = 0x00; /* Set divisor latch MSB */
U1LCR = 0x03; /* DLAB = 0 */

U1IER = 0x05; /* Enable UART1 RX and Error Interrupts */
U1FCR = 0x07; /* Enable and reset TX and RX FIFO. trigger level 0 */

```

Fonte: Elaboração própria (2026).

3.3.3 Definição da estrutura do *firmware*

Após a validação do funcionamento das UARTs, a etapa seguinte consistiu no desenvolvimento das rotinas responsáveis pelo gerenciamento da comunicação e processamento dos dados. Com a utilização do RTOS, essas funcionalidades foram estruturadas na forma de tarefas, cada uma responsável por uma etapa específica do processo de comunicação.

Conforme apresentado anteriormente, as interrupções geradas pelas UARTs são atendidas pela função associada ao registrador *VICVectAddr*. O Quadro 8 apresenta a implementação desenvolvida para a UART1, análoga à utilizada na UART2. Nessa rotina, os dados recebidos são armazenados em um *buffer* circular dedicado à respectiva UART, permitindo que cada interface de comunicação possua sua própria área de armazenamento temporário. Por se tratar de uma rotina de atendimento à interrupção, sua execução deve ser rápida e conter o mínimo de instruções possível. Dessa forma, a implementação limita-se à verificação da integridade dos dados recebidos, descartando aqueles que apresentam erro e armazenando os demais no *buffer* circular. A interpretação dos dados e a execução das ações correspondentes são realizadas por uma tarefa específica, acionada por meio da função *isr_evt_set*, referente ao recurso *Event Flag*.

Quadro 8 – Rotina para o tratamento de interrupções da UART1

```

/*-----
*                               USART1_IRQHandler: UART1 receive interrupt                               *
*-----*/
void USART1_IRQHandler(void) __irq {
    uint8_t IIRValue, LSRValue, Descarte;
    IIRValue = ((uint8_t) U1IIR >> 0x01);
    LSRValue = (uint8_t) U1LSR;
    /* Possível erro durante a recepção do dado */
    if ((IIRValue & 0x07) == 0x03) {
        UART1status = LSRValue;
        Descarte = U1RBR;
    } else if ((IIRValue & 0x07) == 0x02) {
        enqueue(&buffer_receive_UART1, (uint8_t) U1RBR); /* Armazena caractere recebido no buffer circular */
        isr_evt_set(0x01, uart1_rx_id); /* Chama tarefa Task_UART1_RX */
    }
    VICVectAddr = 0; /* Acknowledge Interrupt */
}

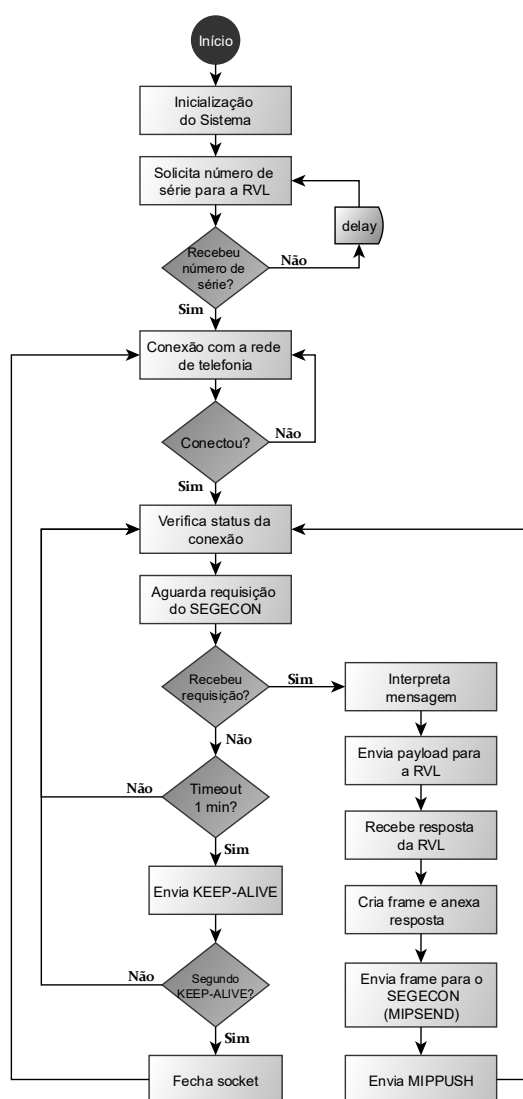
```

Fonte: Elaboração própria (2026).

A Figura 19 apresenta o fluxograma das etapas executadas pelo novo *firmware* para reproduzir o funcionamento da solução original e realizar a transmissão dos dados ao SEGECON. Inicialmente, são executadas as rotinas de inicialização do sistema, incluindo a configuração dos periféricos e a criação das tarefas do RTOS. Em seguida, o microcontrolador solicita o número de série à placa RVL, repetindo a requisição periodicamente até obter uma resposta. Após a obtenção do número de série, é iniciado o processo de conexão com a rede de telefonia móvel e de abertura do *socket* TCP para comunicação com o SEGECON. Uma vez estabelecida a

conexão, a placa permanece monitorando seu estado e aguardando requisições do servidor. Na ausência de mensagens do SEGECON por um período de um minuto, é enviado um *frame* KEEP-ALIVE. Caso não haja comunicação durante mais um minuto, o *socket* é encerrado e o processo de conexão é reiniciado. Por outro lado, ao receber uma requisição do servidor, o *firmware* interpreta o *frame* recebido, extrai o *payload* e o converte para o formato utilizado pela placa RVL. Após receber a resposta, o microcontrolador realiza as conversões necessárias, monta um novo *frame*, conforme o protocolo da RMS, e o transmite ao SEGECON. Como descrito anteriormente, após o envio da mensagem por meio do comando MIPSEND, é necessário executar o comando MIPUSH para efetivar sua transmissão pela pilha TCP do modem.

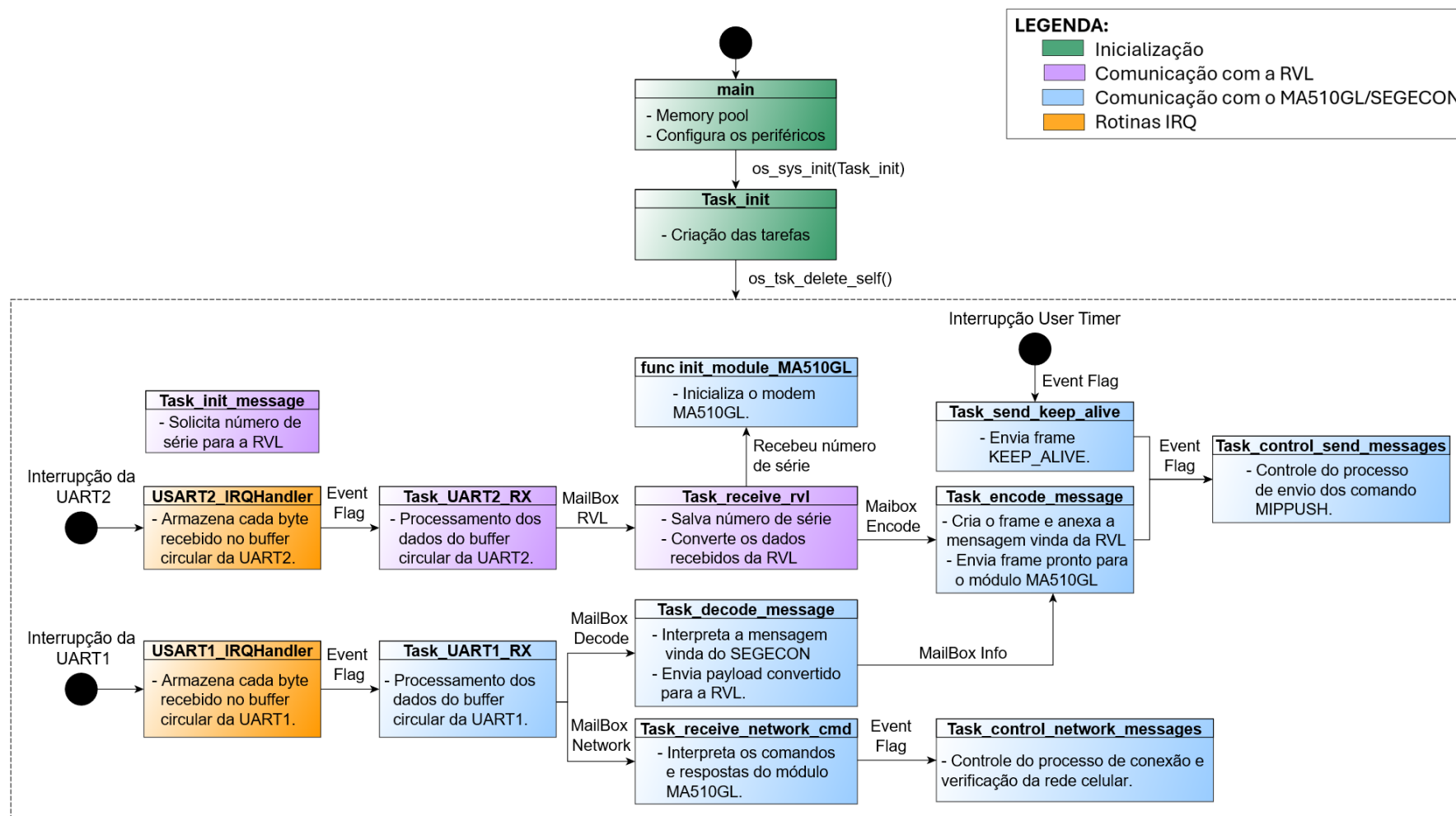
Figura 19 – Fluxograma do funcionamento do *firmware*



Fonte: Elaboração própria (2026).

Com base no fluxo de operação apresentado na Figura 19, foi desenvolvida a arquitetura do *firmware* mostrada na Figura 20, utilizando o sistema operacional de tempo real RL-RTX. A arquitetura foi estruturada em tarefas responsáveis pela inicialização do sistema, pela comunicação com a placa RVL, pela comunicação com o módulo MA510GL e pelo processamento das mensagens trocadas com o servidor SEGECON.

A arquitetura desenvolvida utiliza os principais mecanismos de comunicação e sincronização disponibilizados pelo RL-RTX. A troca de informações é realizada por meio de *Mailboxes*, que transportam ponteiros para as estruturas de dados alocadas em blocos de memória dedicados (*Memory Pools*), reduzindo o consumo de memória e evitando cópias desnecessárias de dados. A sincronização entre tarefas e rotinas de interrupção é feita por *Event Flags*, utilizadas para sinalizar a disponibilidade de novos dados para processamento. Já os *User Timers* são empregados na execução periódica de atividades temporizadas, como o envio do *frame* KEEP-ALIVE.

Figura 20 – Arquitetura do *firmware*

Fonte: Elaboração própria (2026).

A aplicação é iniciada pela função *main*, responsável pela configuração dos periféricos e pela criação dos *Memory Pools* destinados ao armazenamento das estruturas de dados cujos ponteiros são compartilhados entre as tarefas por meio de *Mailboxes*. Em seguida, o sistema operacional é inicializado e executa a tarefa *Task_init*, responsável pela criação das demais tarefas do sistema, encerrando sua execução após essa etapa.

Após a inicialização, a tarefa *Task_init_message* solicita periodicamente o número de série à placa RVL por meio da mensagem padrão apresentada no Quadro 5. Somente após o recebimento dessa informação é iniciado o processo de configuração do módulo MA510GL e de estabelecimento da conexão com a rede celular.

Conforme ilustrado na Figura 20, os dados recebidos pelas UARTs são inicialmente tratados pelas rotinas de interrupção, responsáveis apenas por armazenar os *bytes* recebidos em *buffers* circulares e sinalizar a disponibilidade de novos dados por meio de *Event Flags*. O processamento desses dados é realizado pelas tarefas *Task_UART1_RX* e *Task_UART2_RX*, que analisam os dados armazenados nos *buffers* circulares e os direcionam para as tarefas responsáveis pela interpretação das requisições do SEGECON, pelo tratamento das respostas da RVL e pelo controle da conexão com a rede celular.

As tarefas *Task_decode_message* e *Task_encode_message* realizam o processamento das mensagens trocadas entre o SEGECON e a RVL. A *Task_decode_message* valida a integridade da mensagem recebida do SEGECON, interpreta seus campos e converte o *payload* de ASCII hexadecimal para o formato binário esperado pela RVL. Além disso, encaminha à *Task_encode_message* as informações do cabeçalho da requisição que deverão ser reutilizadas na composição do *frame* de resposta. Após o processamento da requisição pela RVL, a tarefa *Task_receive_rvl* recebe a resposta enviada pela placa, converte os dados retornados para ASCII hexadecimal e os encaminha à *Task_encode_message*. Essa tarefa é responsável por montar o *frame* conforme o protocolo RMS e prepará-lo para envio ao SEGECON por meio do comando MIPSEND. O envio é finalizado pela tarefa *Task_control_send_messages*, que executa o comando MIPPUSH.

O controle da conexão com a rede celular é realizado pela tarefa *Task_control_network_messages*, enquanto o envio periódico do *frame* KEEP-ALIVE é executado pela tarefa *Task_send_keep_alive*, acionada por um *User Timer* do RL-RTX.

4 APRESENTAÇÃO DOS RESULTADOS

Este capítulo apresenta os resultados obtidos com o *firmware* desenvolvido por meio da análise dos *logs* de comunicação gerados durante sua operação. O objetivo é verificar o correto funcionamento das funcionalidades implementadas e sua conformidade com o comportamento esperado para o sistema.

Para isso, são analisados três cenários principais da operação do *firmware*: o estabelecimento da conexão com o servidor SEGECON e o recebimento da solicitação do número de série; o processamento da solicitação de um bloco MM (memória de massa), incluindo o envio do bloco subsequente após o recebimento da mensagem de confirmação (ACK, código 06); e o envio periódico do *frame* KEEP-ALIVE, bem como o encerramento da conexão. A análise desses cenários permite validar os mecanismos de comunicação implementados entre o SEGECON, a placa de comunicação e a RVL.

4.1 Cenário 1 – Estabelecimento da conexão com o servidor SEGECON

No Quadro 9 é apresentada a etapa de estabelecimento da comunicação com o servidor SEGECON. A resposta +MIPOPEN:1,1 indica a abertura bem-sucedida do *socket* TCP pelo módulo MA510GL. Em seguida, observa-se o recebimento de uma solicitação proveniente do servidor, identificada pela mensagem +MIPRTCP, e o envio da respectiva resposta por meio do comando AT+MIPSEND. Esse comportamento demonstra que o *firmware* recebeu, processou e respondeu corretamente à requisição recebida. Após o envio dos dados, é executado o comando AT+MIPPUSH, responsável por efetivar a transmissão da mensagem ao servidor, concluindo o ciclo de comunicação. Como a resposta enviada ao SEGECON depende das informações obtidas junto à RVL, o trecho apresentado também evidencia o correto funcionamento da comunicação entre os dispositivos envolvidos.

[illegible]

4.3 Cenário 3 – Manutenção e encerramento da conexão

No Quadro 11, observa-se a execução do mecanismo de supervisão da conexão implementado no *firmware*. Para isso, são enviados periodicamente os comandos MIPCALL?, CREG? e CSQ?, utilizados para verificar, respectivamente, o estado da conexão de dados, o registro do módulo na rede celular e a intensidade do sinal recebido.

Além do monitoramento da rede, é realizado o envio periódico do *frame* KEEP-ALIVE por meio dos comandos MIPSEND e MIPPUSH, com o objetivo de manter a conexão ativa na ausência de requisições do servidor SEGECON. Conforme apresentado no *log*, o primeiro frame KEEP-ALIVE foi transmitido às 15:23:16 e o segundo às 15:24:17, evidenciando o intervalo de 60 segundos configurado no *User Timer* do RL-RTX. Após a transmissão do segundo *frame*, a conexão é encerrada por meio do comando MIPCLOSE, concluindo o ciclo de comunicação.

Quadro 11 – Monitoramento da conexão e envio do *frame* KEEP-ALIVE, com o fechamento do *socket* TCP e início da nova comunicação.

```
15:23:16.520> AT+MIPSEND=1,"5B524D5300000001FE0E5D00BF40"
...
15:23:17.542> +MIPPUSH: 1,0
[...]
15:24:11.051> AT+MIPCALL?
15:24:12.057> +MIPCALL: 1,10.2.153.238
15:24:12.057> OK
15:24:12.057> AT+CREG?
15:24:13.061> +CREG: 0,1
15:24:13.061> OK
15:24:13.061> AT+CSQ?
15:24:14.081> +CSQ: 18,99
15:24:14.081> OK
15:24:14.081> AT+MIPCALL?
15:24:15.081> +MIPCALL: 1,10.2.153.238
15:24:15.081> OK
15:24:15.081> AT+CREG?
15:24:16.128> +CREG: 0,1
```

Primeiro frame KEEP-ALIVE enviado.

```

15:24:16.128> OK
15:24:16.128> AT+CSQ?
15:24:17.124> +CSQ: 18,99
15:24:17.124> OK
15:24:17.124> AT+MIPSEND=1"5B524D5300000001FE0E5D00BF40"
15:24:17.565> +MIPSEND: 1,0,2034 Segundo frame KEEP-ALIVE enviado.
15:24:17.565> OK
15:24:17.565> AT+MIPPUSH=1
15:24:18.132> +MIPPUSH: 1,0
15:24:18.132> OK
15:24:18.132> AT+MIPCLOSE=1
15:24:18.228> OK
15:24:22.955> +MIPCLOSE: 1,0
15:24:22.955> AT
15:24:23.972> OK

```

Fonte: Elaboração própria (2026).

4.4 Análise e discussão dos resultados

Durante a etapa de validação, o *firmware* permaneceu em operação entre 15h18 e 16h45, período no qual foram realizados 46 ciclos de comunicação com o servidor SEGECON. Dentre esses ciclos, apenas um envolveu a transferência dos blocos MM, na qual foram transmitidos 50 blocos. Nesse caso, a sequência de solicitações apresentada no Quadro 2 foi executada duas vezes durante a mesma sessão de comunicação. Essa variação não impacta o comportamento do *firmware*, uma vez que a placa de comunicação atua apenas como intermediária entre o SEGECON e a RVL, sem interferir na lógica de troca de dados definida por esses dispositivos.

Ao longo dos testes, não foram observados travamentos, reinicializações inesperadas ou perda de comunicação com o servidor, mesmo durante a transferência dos blocos de memória de massa, caracterizada pelo elevado volume de dados transmitidos e por blocos de 1002 *bytes*. Grande parte dos ciclos de comunicação seguiu o fluxo principal de operação apresentado na Figura 17. Os *logs* analisados demonstraram que o *firmware* foi capaz de processar e encaminhar corretamente as

informações entre o SEGECON e a RVL, mantendo a estabilidade operacional durante todo o período de avaliação. Entretanto, foram identificadas algumas falhas de funcionamento. Em uma ocasião, o *firmware* reenviou o comando MIPOPEN mesmo após ter recebido a resposta esperada do módulo. Além disso, foram registradas cinco ocorrências de encerramento da conexão em momento inadequado e cinco situações em que o *firmware* não identificou corretamente a URC +MIPSTAT: 1,2, enviada pelo modem para indicar o fechamento da conexão TCP. Essas falhas foram as que mais influenciaram os resultados da validação, refletindo diretamente na taxa de sucesso observada. A análise dos *logs* indicou que essas duas últimas falhas estão relacionadas. Em todos os casos, o *firmware* conseguiu restabelecer a operação após o encerramento da conexão, indicada pelo recebimento da URC +MIPSTAT: 1,2, porém o temporizador responsável pelo envio do *frame* KEEP-ALIVE não era reinicializado. Como essa URC era recebida após o primeiro envio do KEEP-ALIVE, o temporizador permanecia em execução durante o início da sessão seguinte. Como consequência, o comando MIPCLOSE era executado prematuramente, ocasionando o encerramento da nova conexão em momento inadequado.

Apesar dessas ocorrências, o *firmware* recuperou sua operação automaticamente em todos os casos, mantendo a execução normal do sistema e sem comprometer a transferência dos blocos da memória de massa. Considerando os 46 ciclos avaliados, 36 ocorreram conforme o comportamento esperado, resultando em uma taxa de sucesso de aproximadamente 78%. Os problemas identificados não foram considerados críticos, pois ocorreram na etapa de estabelecimento da conexão e, portanto, não ocasionaram perda de dados. Além disso, não impediram o funcionamento do sistema nem comprometeram sua principal funcionalidade, que consiste na intermediação da comunicação entre o SEGECON e a RVL.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo desenvolver um *firmware* para a placa de comunicação utilizada nos registradores de energia da RMS, permitindo o compartilhamento dos dados de medição com o servidor SEGECON por meio da rede celular.

Para isso, foi realizado inicialmente um estudo da arquitetura do sistema, abrangendo o protocolo RMS, os mecanismos de comunicação celular, o funcionamento do modem MA510GL e os principais recursos do microcontrolador LPC2368 empregados na aplicação. A análise dos *logs* obtidos a partir da placa original permitiu compreender o fluxo de comunicação existente e identificar os requisitos necessários para a implementação do novo *firmware*.

A solução desenvolvida utilizou o sistema operacional de tempo real RL-RTX, explorando recursos como *Event Flags*, *Mailboxes*, *Memory Pools* e *User Timers* para estruturar a aplicação de forma modular e facilitar a sincronização entre tarefas, periféricos e rotinas de interrupção. Além disso, foram implementados mecanismos para gerenciamento da conexão com a rede móvel, controle dos *sockets* TCP para a comunicação com o servidor SEGECON, processamento das mensagens com base no protocolo da RMS e comunicação com a placa RVL.

Os testes realizados demonstraram que o *firmware* foi capaz de estabelecer comunicação com o servidor SEGECON, responder corretamente às solicitações recebidas, realizar a transferência de blocos de memória de massa e manter a conexão ativa por meio do envio periódico do *frame* KEEP-ALIVE. Durante a etapa de validação, foram observadas algumas falhas relacionadas ao gerenciamento do encerramento da conexão e à reinicialização dos temporizadores, alterando o fluxo de operação esperado. Entretanto, tais ocorrências não comprometeram a funcionalidade principal do sistema, uma vez que o *firmware* apresentou capacidade de recuperação automática e manteve a operação estável durante todo o período de avaliação, sem perdas de dados de medição.

Dessa forma, conclui-se que os objetivos propostos foram atingidos, sendo desenvolvido um *firmware* funcional para a placa de comunicação, capaz de intermediar a troca de dados entre o SEGECON e a placa RVL.

5.1 Sugestões para trabalhos futuros

Como proposta para trabalhos futuros, recomenda-se inicialmente a correção das inconsistências identificadas durante a etapa de validação, bem como a ampliação do período de testes em campo, de modo a verificar a ocorrência de novas falhas e aprimorar a robustez e a confiabilidade do sistema.

Além disso, a solução atualmente desenvolvida não emprega mecanismos de criptografia para a transmissão dos dados. Nesse contexto, destaca-se que o modem MA510GL dispõe nativamente de recursos capazes de suportar a criptografia das comunicações, sem a necessidade de processamento por parte do microcontrolador, possibilitando a adoção de protocolos seguros em futuras versões do *firmware* sem o comprometer o desempenho.

Uma possibilidade de melhoria, em relação ao *hardware*, é a substituição do microcontrolador LPC2368 pelo LPC1768, também da NXP. Esse microcontrolador é compatível pino a pino e oferece melhoria no desempenho por apresentar um núcleo ARM Cortex-M3 e frequência máxima de operação de 100 MHz. A utilização de um Cortex-M3 permitiria a substituição do RL-RTX por um RTOS mais atual, como *FreeRTOS*.

Outra perspectiva de evolução está relacionada ao desenvolvimento de um novo servidor SEGECON. Durante o período de migração, a placa de comunicação poderá atuar de forma simultânea com ambas as infraestruturas, encaminhando os dados tanto para o servidor atualmente em operação quanto para o novo servidor em desenvolvimento, permitindo a validação progressiva da nova solução antes da descontinuação do sistema legado. Essa abordagem exigirá a otimização do mecanismo de gerenciamento de conexões implementado na tarefa *Task_control_network_messages*, além da implementação de funcionalidades capazes de suportar o gerenciamento concorrente de dois *sockets* TCP ativos.

REFERÊNCIAS

BRS SISTEMAS ELETRÔNICOS. **PROTOCOLO DE COMUNICAÇÃO GPRS**. 2015. *Documento confidencial de acesso restrito*.

DENARDIN, Gustavo Weber; BARRIQUELLO, Carlos Henrique. **Sistemas operacionais de tempo real e sua aplicação em sistemas embarcados**. São Paulo: Blucher, 2019. 474 p. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/9788521213970/pageid/0si>. Acesso em: 20 maio 2026.

DIBAL, P. Y. DESIGN OF A WIRELESS SENSOR NETWORK USING LPC2103 MICROCONTROLLER. **Arid Zone Journal Of Engineering, Technology And Environment**. Maiduguri, p. 53-66. ago. 2012. Disponível em: https://www.academia.edu/30581677/DESIGN_OF_A_WIRELESS_SENSOR_NETWORK_USING_LPC2103_MICROCONTROLLER. Acesso em: 05 nov. 2025.

FIBOCOM. **MA510**. Disponível em: https://www.fibocom.com/en/SeriesProduct/info_itemid_147.html. Acesso em: 15 jun 2026.

FIBOCOM. **MTC AT Commands User Manual: TCP&UDP**. 2024. *Documento confidencial de acesso restrito*.

FIBOCOM. **Fibocom_MA510&MC108&MC109_Series**: AT Commands. 2024. *Documento confidencial de acesso restrito*.

IBM. **O que é um sistema operacional em tempo real (RTOS)?** Disponível em: <https://www.ibm.com/br-pt/think/topics/real-time-operating-system>. Acesso em: 20 maio 2026.

KEIL. **Overview**. Disponível em: https://www.keil.com/support/man/docs/rlarm/rlarm_ar_overview.htm. Acesso em: 15 mar. 2026.

KEIL. **Product Description**. Disponível em: https://www.keil.com/support/man/docs/rlarm/rlarm_ar_product.htm. Acesso em: 15 mar. 2026.

KEIL. **Product Specification**. Disponível em: https://www.keil.com/support/man/docs/rlarm/rlarm_ar_product_spec.htm. Acesso em: 15 mar. 2026.

KEIL. **Defining Tasks**. Disponível em: https://www.keil.com/support/man/docs/rlarm/rlarm_ar_task_def.htm. Acesso em: 25 mar. 2026.

KEIL. External References. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_extern_ref.htm. Acesso em:
25 mar. 2026.

KEIL. NXP (founded by Philips) LPC2368. Disponível em:
<https://www.keil.com/dd/chip/4152.htm>. Acesso em: 15 mar. 2026.

KEIL. Tasks. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_cfgtask.htm. Acesso em: 25
mar. 2026.

KEIL. Your First RTX Application. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_your_first_app.htm. Acesso
em: 25 mar. 2026.

KEIL. Hardware Timer. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_cfgtimer.htm. Acesso em: 30
mar. 2026.

KEIL. Pre-emptive Scheduling. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_preemt.htm. Acesso em: 30
mar. 2026.

KEIL. Scheduling Options. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_schedopt.htm. Acesso em: 30
mar. 2026.

KEIL. System Task Manager. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_clock_task.htm. Acesso em:
30 mar. 2026.

KEIL. Timer Tick Interrupt. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_timer_tick.htm. Acesso em:
30 mar. 2026.

KEIL. User Timers. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_user_timers.htm. Acesso em:
30 mar. 2026.

KEIL. Event Flag Management Routines. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_evt_mgmt_funcs.htm. Acesso
em: 02 abr. 2026.

KEIL. Interrupt Functions. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_inter_func.htm. Acesso em:
02 abr. 2026.

KEIL. Using a Mailbox. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_mbx_using.htm. Acesso em:
02 abr. 2026.

KEIL. **Memory Allocation Routines**. Disponível em:
https://www.keil.com/support/man/docs/rlarm/rlarm_ar_mem_mgmt_funcs.htm.
 Acesso em: 03 abr. 2026.

NORDIC SEMICONDUCTOR (Noruega). **Lesson 1 – Basics of cellular IoT**.
 Disponível em: <https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-1-cellular-fundamentals/>. Acesso em: 10 maio 2026.

NORDIC SEMICONDUCTOR (Noruega). **LTE-M and NB-IoT**. Disponível em:
<https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-1-cellular-fundamentals/topic/lesson-1-lte-m-and-nb-iot/>. Acesso em: 11 maio 2026.

NORDIC SEMICONDUCTOR (Noruega). **AT commands**. Disponível em:
<https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-2-cellular-fundamentals/topic/lesson-2-at-commands/>. Acesso em: 12 maio 2026.

NORDIC SEMICONDUCTOR (Noruega). **Network coverage and SIM cards**.
 Disponível em: <https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-1-cellular-fundamentals/topic/lesson-1-network-coverage-globally/>. Acesso em: 12 maio 2026.

NORDIC SEMICONDUCTOR (Noruega). **Network programming**. Disponível em:
<https://academy.nordicsemi.com/courses/cellular-iot-fundamentals/lessons/lesson-3-cellular-fundamentals/topic/network-programming/>. Acesso em: 12 maio 2026.

NXP. **LPC2364/65/66/67/68**. 2013. Disponível em:
https://www.nxp.com/docs/en/data-sheet/LPC2364_65_66_67_68.pdf. Acesso em:
 10 nov. 2025.

NXP. **LPC2368FBD100**. Disponível em: <https://www.nxp.com/part/LPC2368FBD100>.
 Acesso em: 12 nov. 2025.

NXP. **UM10211**: LPC2364/66/68/78 User manual. 2007. Disponível em:
<https://docs.rs-online.com/62ce/0900766b80a5bed6.pdf#l4.5.2128352>. Acesso em:
 15 nov. 2026.

SOUSA, Daniel Rodrigues de. **Microcontroladores ARM7 (Philips, Família LPC213X)**: o poder dos 32 bits - teoria e prática. São Paulo: Érica, 2006. 278 p.