

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
BACHARELADO EM ENGENHARIA ELÉTRICA**

GUILHERME EICHSTÄDT

**SISTEMA DE COMUNICAÇÃO DE DISPOSITIVOS EM UMA
MICRORREDE DE ENERGIA**

JOINVILLE

2018

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
BACHARELADO EM ENGENHARIA ELÉTRICA**

GUILHERME EICHSTÄDT

**SISTEMA DE COMUNICAÇÃO DE DISPOSITIVOS EM UMA
MICRORREDE DE ENERGIA**

Trabalho de Conclusão de Curso submetido ao Bacharelado em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do Grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Marcello Mezaroba

Coorientador: Ms. Felipe Joel Zimann

JOINVILLE

2018

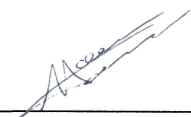
GUILHERME EICHSTÄDT

**SISTEMA DE COMUNICAÇÃO DE DISPOSITIVOS EM UMA
MICRORREDE DE ENERGIA**

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia Elétrica como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

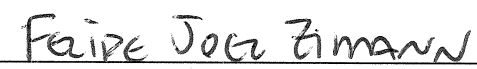
Banca Examinadora

Orientador:



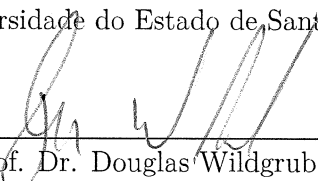
Prof. Dr. Marcello Mezaroba
Universidade do Estado de Santa Catarina

Coorientador:



Ms. Felipe Joel Zimann
Universidade do Estado de Santa Catarina

Membros:



Prof. Dr. Douglas Wildgrube Bertol
Universidade do Estado de Santa Catarina



Prof. Dr. Eduardo Bonci Cavalcante
Universidade do Estado de Santa Catarina

Joinville, 21 de Junho 2018.

AGRADECIMENTOS

A minha família que me apoiou e sempre me incentiva à dedicação aos estudos.

Ao professor Dr. Marcello Mezaroba que me auxiliou neste projeto e me recebeu no projeto de pesquisa proporcionando muito aprendizado e desenvolvimento.

Ao Ms. Felipe Joel Zimann pelo auxílio, apoio com correções, sugestões mas sobre tudo, ensinamentos transmitidos.

Ao Dr. Christofer Schwartz por toda a ajuda fornecida, sempre me auxiliando nos apertos da programação do software.

Aos professores da universidade que durante este tempo da graduação contribuíram com meu aprendizado.

Aos meus amigos e colegas que me ajudaram quando precisei e sempre estiveram ao meu lado durante a graduação.

*“Computadores são inúteis.
Eles conseguem apenas lhe dar respostas”
Pablo Picasso, 1881-1973*

RESUMO

Com a crescente necessidade do uso de energia elétrica, cresce também a necessidade de obtenção de novos meios de geração e distribuição. Procurando reduzir este problema e visando contar com energia elétrica em locais isolados uma alternativa é o uso das microrredes, onde a geração de energia ocorre mais próxima à carga. Para um bom funcionamento desta, é ideal o uso de um sistema que monitore seus dispositivos. Este trabalho propõe a composição de um conjunto de hardwares, softwares, firmwares e protocolos de comunicação que permitem o monitoramento dos dispositivos de uma microrrede. Além disso, o software desenvolvido permite a leitura dos dados através de uma conexão via UDP com um software externo para o processamento dos dados. Os hardwares proporcionam uma comunicação com ou sem fio permitindo mais mobilidade para o usuário. Para a validação do sistema foram utilizados dois estudos de caso, um com um inversor solar que utiliza uma linguagem de comunicação própria e outro com o uso de DSC com protocolo desenvolvido neste trabalho.

Palavras-chave: Microrrede, sistema de monitoramento, redes inteligentes, ESP8266, energia renovável

ABSTRACT

The need to obtain new ways of generation and distribution is growing due to the increasing of electric energy use. In order to reduce this problem and bring electricity in isolated locations, an alternative is the use of microgrids, where power generation occurs closer to the load. For a good operation of this, it is ideal to use a system that monitors those devices. This work proposes the composition of a set of hardware, software, firmware and communication protocols that allow monitoring devices in a microgrid. In addition, the developed software allows the reading of the data through a connection via UDP with external software. The hardware provides a wired or wireless communication allowing more mobility for the user. For the validation of the system two studies case were used, one with a solar inverter that uses its own communication language and another with the use of DSC with protocol developed in this work.

Keywords: Microgrid, monitoring system, smart grid, ESP8266, renewable energy

LISTA DE FIGURAS

1.1	Rede de monitoramento proposta.	15
2.1	Exemplo de estrutura de uma micro rede de energia.	19
2.2	Divisão dos controles de uma microrrede.	20
2.3	Exemplo de transmissão de dados utilizando o padrão RS-485.	22
2.4	Exemplo de transmissão de dados em uma comunicação utilizando RS-232.	23
3.1	Pinagem do módulo ESP8266-12E.	29
3.2	Modos de operação do ESP8266 conforme tipo de utilização.	30
3.3	Diagrama de blocos do conversor RS-232/RS-485.	32
3.4	Projeto em 3D (a) e placa de circuito impresso produzida (b) da placa de conversão RS485/RS232.	32
3.5	Diagrama de blocos da placa de comunicação para dispositivos RS485.	33
3.6	Comparação entre projeto 3D (a) e placa montada (b) da placa de comunicação para dispositivos RS485	34
3.7	Diagrama de blocos da placa de comunicação para dispositivos RS-232.	35
3.8	Comparação entre projeto 3D (a) e placa montada (b) da placa de comunicação para dispositivos RS-232.	35
4.1	Visão geral das camadas de protocolos utilizadas na comunicação entre dispositivos.	36
4.2	Visão detalhada das camadas de protocolos utilizadas na comunicação entre dispositivos.	37
4.3	Exemplo de mensagens enviadas entre o software de monitoramento e placa de comunicação com ESP8266.	41
4.4	Exemplo de mensagens enviadas pelo CCS para placa de comunicação com ESP8266 solicitando reinicialização.	43
4.5	Exemplo de mensagens enviadas entre CCS e software externo para aquisição de parâmetros.	47
4.6	Exemplo de mensagens enviadas entre CCS e software externo solicitando os valores das variáveis monitoradas em todos os dispositivos conectados ao CCS.	50
4.7	Exemplo de mensagens enviadas entre software criado (CCS) e software externo solicitando atualização dos valores das variáveis.	53
4.8	Exemplo de mensagens enviadas entre CCS e DSC solicitando parâmetros.	58
4.9	Exemplo de mensagens enviadas entre DSC e CCS para monitoramento dos valores das variáveis.	61

4.10	Exemplo de mensagens enviadas entre CCS e DSC solicitando atualização dos valores das variáveis.	63
4.11	Comunicação do inversor solar com AP para registro de um inversor.	66
4.12	Comunicação do inversor solar com CCS para verificação de informações.	69
4.13	Esquema do fluxo de comunicação com protocolos apresentados.	71
5.1	Esquema hierárquico entre componentes conectados à microrrede.	73
5.2	Definição das portas de comunicação UDP.	76
5.3	Janela principal do CCS na versão PRO 2.7 com a aba “Device 1” selecionada.	77
5.4	Janela principal do CCS na versão PRO 2.7 com a aba “PHB+” selecionada.	80
5.5	Janela de parâmetros extras do CCS na versão PRO 2.7.	81
5.6	Janela sobre o CCS na versão PRO 2.7.	82
5.7	Modelo de arquivo do tipo .txt de dados criado pelo software.	87
5.8	Modelo de arquivo do tipo .txt de erros criado pelo software.	87
5.9	Variáveis utilizadas para recebimento de mensagem pelo CCS no DSC.	89
5.10	Tela no Simulink em que o usuário efetua interações com dispositivos e requisição de mensagens separada em três áreas.	90
5.11	Blocos de interação no Simulink, via Matlab, que executam as funções programadas.	92
6.1	Depuração do ESP8266 ao ligar a placa de comunicação e efetuar o processo de conexão ao CCS.	93
6.2	Gráficos gerados através da simulação da interação da requisição do software externo com CCS.	95
6.3	Telas de softwares mostrando requisição de atualização solicitada pelo software externo para o DSC.	96
6.4	Gráficos de potência e temperatura do inversor solar gerados a partir dos dados salvos no log de dados gerados pelo CCS.	98
6.5	Telas de softwares mostrando requisição de dados solicitada pelo software externo para o inversor PHB.	99
A.1	Nomes dos objetos utilizados na janela principal na aba de dispositivos para um dispositivo genérico de número “XX”.	103
A.2	Nomes dos objetos utilizados na janela principal aba PHB+.	104
A.3	Nomes dos objetos utilizados na janela de parâmetros extras	105
B.1	Descrição de objetos utilizados no fluxograma apresentado	106
C.1	Descrição de objetos utilizados no fluxograma apresentado	134
D.1	Descrição de objetos utilizados no fluxograma apresentado	141
E.1	Descrição de objetos utilizados no fluxograma apresentado	146

LISTA DE TABELAS

2.1	Níveis de tensão de diferença entre sinais padronizados para o padrão RS-485.	22
2.2	Níveis de tensão padronizados pelo padrão RS-232.	23
2.3	Estrutura dos bytes do protocolo Modbus.	24
2.4	Principais diferenças entre modo de transmissão para Modbus RTU e Modbus ASCII.	25
2.5	Estrutura típica para comunicação Modbus em modo ASCII e RTU.	25
4.1	Valores configurados para o <i>control byte</i> e respectivos modos de operação. . . .	38
4.2	Estrutura dos Bytes do protocolo de comunicação.	38
4.3	Tabela com valores de <i>function code</i> e respectivas funções no protocolo ESP. .	39
4.4	Parâmetros contidos no campo <i>data</i> no exemplo de comunicação entre o CCS e o ESP8266 para o registro da placa de comunicação.	40
4.5	Análise das mensagens enviadas no processo de registro da placa de comunicação para o protocolo ESP.	42
4.6	Mensagem enviada pelo CCS solicitando a reinicialização do ESP.	43
4.7	Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com software externo.	44
4.8	Tabela de funções para comunicação com software externo.	45
4.9	Parâmetros no exemplo de comunicação entre o CCS e o ESP8266.	45
4.10	Análise das mensagens enviadas no processo de aquisição dos parâmetros no protocolo software externo (Continua).	48
4.11	Parâmetros contidos no campo <i>data</i> no exemplo de de comunicação entre o CCS e o software externo para a requisição dos valores das variáveis.	50
4.12	Análise das mensagens enviadas no processo de aquisição dos valores das variáveis no protocolo software externo.	52
4.13	Parâmetros contidos no campo <i>data</i> no exemplo de de comunicação entre o CCS e o software externo para a requisição de atualização dos valores das variáveis.	53
4.14	Análise das mensagens enviadas no processo de atualização dos valores das variáveis no protocolo software externo.	55
4.15	Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com DSC.	56
4.16	Tabela de funções para comunicação com DSC.	56

4.17	Parâmetros contidos no campo <i>data</i> no exemplo de de comunicação entre o CCS e o DSC para pedidos de parâmetros.	57
4.18	Análise das mensagens enviadas no processo de solicitação de parâmetros (Continua).	59
4.19	Parâmetros contidos no campo <i>data</i> no exemplo de de comunicação entre o CCS e o DSC para requisição do valor das variáveis.	61
4.20	Análise das mensagens enviadas no processo requisição do valor das variáveis.	62
4.21	Análise das mensagens enviadas no processo de requisição de atualização dos valores das variáveis.	64
4.22	Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com inversor PHB.	64
4.23	Mensagens enviadas no processo de registro de um inversor a rede.	67
4.24	Parâmetros contidos no campo <i>data</i> no exemplo de comunicação entre o CCS e o DSC para mensagem de pedido de registro.	68
4.25	Parâmetros contidos no campo <i>data</i> no exemplo de comunicação entre o CCS e o DSC para mensagem para alocar endereço de registro.	68
4.26	Análise das mensagens enviadas no processo de requisição de dados.	69
4.27	Parâmetros contidos no campo <i>data</i> no exemplo de de comunicação entre o CCS e o DSC para requisição do valor das variáveis e mensagem de confirmação de endereço.	70

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Objetivos e metodologia	14
1.2	Trabalhos similares	15
1.3	Estrutura do trabalho	16
1.4	Artigo submetido	16
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Geração distribuída	17
2.2	Microrredes	18
2.3	Redes inteligentes	20
2.4	Protocolos e modelos de comunicação	21
2.4.1	Padrão serial RS	21
2.4.1.1	Padrão RS-485	22
2.4.1.2	Padrão RS-232	23
2.4.2	Modbus	24
2.5	Internet das coisas (IoT)	26
2.6	Rede sem fio - Wifi	26
3	HARDWARE	28
3.1	ESP8266	28
3.1.1	Modos de operação do ESP8266	29
3.2	PHB3000-SS	31
3.2.1	Placa de conversão RS-485/RS-232	31
3.2.2	Placa de comunicação Wi-fi para dispositivos RS-485	32
3.3	Digital Signal Controller - DSC	34
3.3.1	Placa de comunicação Wi-fi para dispositivos RS-232	34
4	PROTOCOLOS DE COMUNICAÇÃO	36
4.1	Camada de comunicação control byte	37
4.2	Protocolo ESP	38
4.2.1	Registro da placa de comunicação	39
4.2.2	Reiniciar ESP	43

4.3	Protocolo software externo	44
4.3.1	Aquisição de parâmetros	45
4.3.2	Requisição dos valores das variáveis	50
4.3.3	Requisição de atualização dos valores das variáveis	53
4.4	Protocolo DSC	55
4.4.1	Aquisição de parâmetros	56
4.4.2	Requisição do valor das variáveis	61
4.4.3	Requisição de atualização dos valores das variáveis	62
4.5	Protocolo PHB	64
4.5.1	Registro de um inversor na rede	65
4.5.2	Requisição dos valores das variáveis	68
4.6	Integração entre diferentes dispositivos e protocolos	71
5	SOFTWARES E FIRMWARES	73
5.1	Central Control Software	74
5.1.1	Janelas versão PRO 2.7	76
5.1.1.1	Janela principal	76
5.1.1.2	Aba PHB+	79
5.1.1.3	Janela de parâmetros extras	81
5.1.1.4	Janela sobre	82
5.1.2	Programação e funcionamento	82
5.1.2.1	Funções	83
5.1.2.2	Arquivos de textos gerados via CCS	86
5.2	Firmware ESP8266	87
5.2.1	Funcionamento	88
5.3	Firmware DSC	88
5.3.1	Funcionamento	89
5.4	Software Externo	90
5.4.1	Funcionamento	91
6	RESULTADOS OBTIDOS	93
6.1	Estudo de caso I - DSC	94
6.2	Estudo de caso II - Inversor solar	97
6.3	Disposições finais	100
	REFERÊNCIAS BIBLIOGRÁFICAS	101

APÊNDICE A – ESTRUTURA DOS OBJETOS	103
APÊNDICE B – FLUXOGRAMA DE FUNCIONAMENTO DO CCS	
VERSÃO PRO 2.7	106
APÊNDICE C – FLUXOGRAMA DE FUNCIONAMENTO DO FIRMWARE	
DO ESP8266	134
APÊNDICE D – FLUXOGRAMA DE FUNCIONAMENTO DO FIRMWARE	
DO DSC	141
APÊNDICE E – FLUXOGRAMA DE FUNCIONAMENTO DO SOFTWARE	
EXTERNO (MATLAB)	146

1 INTRODUÇÃO

Sejam por questões ambientais ou, até mesmo, pela possibilidade de geração de energia em locais remotos, a utilização de tecnologias limpas para geração de energia, se torna cada vez mais viável diminuindo a utilização dos sistemas de geração convencionais, compostos basicamente de usinas termoeletricas, hidráulicas e nucleares. Com o crescimento do uso destas tecnologias cresce também a necessidade de desenvolver e utilizar meios de geração de energia distribuída onde o próprio usuário pode gerar a energia elétrica que vai utilizar, gerenciando e controlando os meios que as compõem tornando essencial o surgimento de novas técnicas de gerenciamento e controle que possam aproveitar o melhor desempenho da rede criada.

O aumento e criação desse tipo de sistema trouxe o problema de conectividade, ou seja, como o usuário terá controle sobre a geração e aspectos gerais da geração distribuída. Neste contexto surgiu o conceito de redes inteligentes que utilizam de diversos métodos de comunicação e processamento para controle e gerenciamento da geração. Esta pode incorporar diversos tipos de hardwares e protocolos, o que exige da indústria um esforço adicional para criação destes sistemas.

1.1 OBJETIVOS E METODOLOGIA

Este trabalho tem o objetivo em propor uma solução de comunicação para monitoramento de equipamentos conectados em uma microrrede de energia. Para sanar tal problema, diversos componentes foram criados e estudados para compor a rede.

Para gerenciar os dispositivos conectados, optou-se por centralizar o controle da rede através de um software que coordena os dados. Observando as propostas disponíveis no mercado, não é encontrada uma interface que integre diferentes tipos de protocolos, desta forma foi criado um software central recebe o nome de *Central Control Software - CCS*.

Definido o software de gerenciamento, procurou-se uma forma de conectar o mesmo aos dispositivos monitorados. Para este fim, foram criadas placas de conversão e comunicação. Para fornecer ao usuário mais mobilidade quanto a rede montada, optou-se em proporcionar a transmissão dos dados sem fio. Para isso, as placas de comunicação desenvolvidas contém microcontrolador que suporta comunicação Wifi.

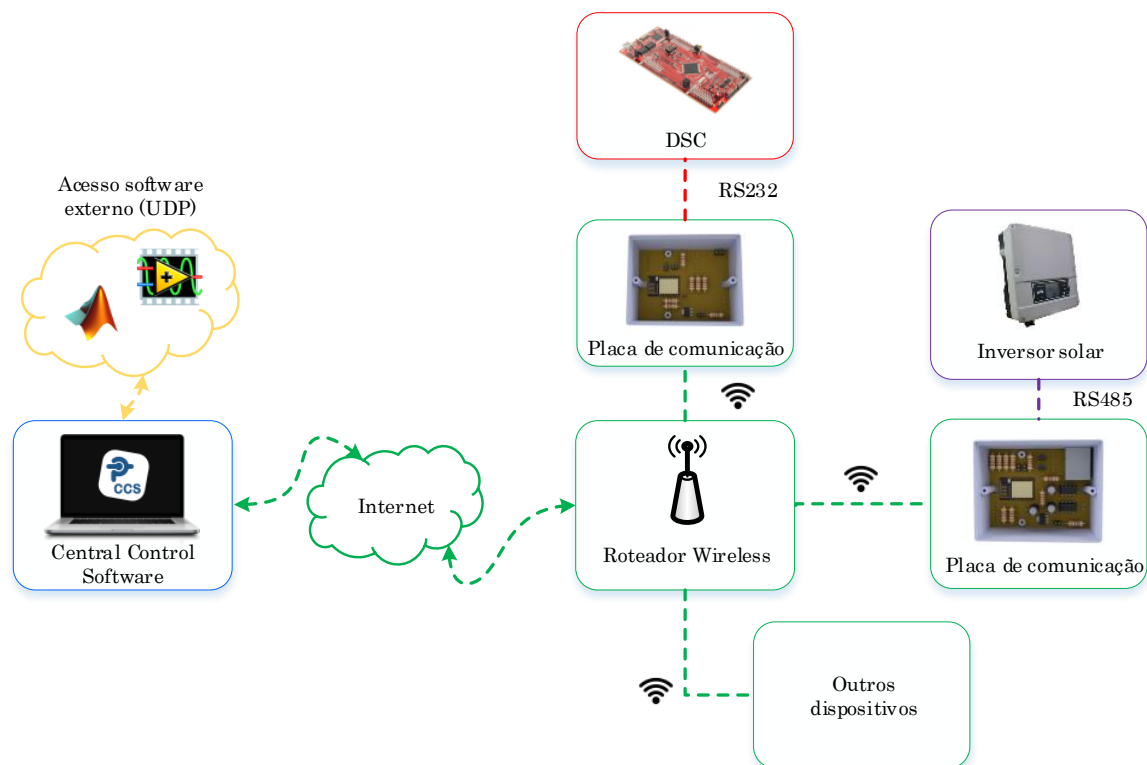
Para o estudo e validação do sistema dois dispositivos foram selecionados, um inversor solar e um DSC (Digital Signal Controller). O primeiro consiste de um produto

comercial, que contém seu próprio protocolo, desta forma fez-se necessário o estudo do mesmo. Já o DSC, um produto que permite customização, não possui um protocolo definido, desta forma, neste trabalho foi desenvolvido um protocolo para o mesmo.

Após o estabelecimento do software de monitoramento verificou-se que este não permite nenhuma manipulação complexa das variáveis requisitadas, para isso permitiu-se a conexão deste, com softwares externo. Assim, um software que permita uma conexão UDP pode se conectar com o CCS e requisitar dados para efetuar alguma modificação ou tratamento dos dados.

A Fig. 1.1 mostra a rede proposta por este trabalho. O software desenvolvido é acessado de um computador que pode está conectado a um software externo. Através da conexão de internet há o roteador wireless que está conectado as placas de comunicação, que por sua vez, conectam-se aos dispositivos a serem monitorados.

Figura 1.1 – Rede de monitoramento proposta.



Fonte: Elaborado pelo autor (2018).

1.2 TRABALHOS SIMILARES

No contexto desta aplicação podem ser citados vários trabalhos relacionados. Em Treter et al. (2015), foi proposto o desenvolvimento de um sistema de aquisição de dados e armazenamento em servidor web, onde os autores apresentam um sistema composto

por um microcontrolador e uma base de dados SQL (*Structured Query Language*), porém, é possível observar que o sistema criado utiliza uma biblioteca comercial permitindo a conexão de dispositivos apenas de uma fabricante. Com uma proposta similar, porém aplicada em um parque fotovoltaico, em Tsagaris e Triantafyllidis (2012), é apresentado um sistema de monitoramento que utiliza CLPs (*Controlador Lógico Programável*) com o uso de um sistema OPC (*Open Platform Communications*).

Comparando os sistemas analisados com o sistema aqui proposto é possível descrever algumas diferenças. Primeiramente, o sistema proposto não utiliza nenhum tipo de biblioteca comercial para determinação do protocolo de comunicação dos dispositivos, permitindo que o sistema fique mais dinâmico e genérico, permitindo a conexão de diversos tipos de dispositivos de variados fabricantes. Com o hardware proposto, ainda é possível uma conexão via Wifi (UDP - *User Datagram Protocol*) que possibilita a conexão sem uso de softwares de terceiros como um sistema OPC citado no segundo caso analisado.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado na seguinte estrutura. O capítulo 2 apresenta a fundamentação teórica, analisando aspectos e conceitos básicos necessários para a compreensão e criação do sistema proposto. O capítulo 3 aponta o hardware utilizado para a execução do projeto com as características principais de cada componente utilizado. No capítulo 4 temos a descrição dos protocolos utilizados e funções. Já o capítulo 5 expõe o software de comunicação criado para a comunicação dos dispositivos. Finalmente, no capítulo 6 são discutidos os resultados obtidos e conclusões.

1.4 ARTIGO SUBMETIDO

G. Eichstädt, F. J. Zimann, C. Schwartz, M. Mezaroba. **“A solution proposal for communication and management of converters operating in microgrids”**. 2017 Brazilian Power Electronics Conference (COBEP), Juiz de Fora, 2017, pp. 1-6.

2 FUNDAMENTAÇÃO TEÓRICA

A necessidade de maior aproveitamento da energia, no que tange à procedência, disponibilização e correta utilização, vem proporcionando o crescimento de novas tendências, que se baseiam na geração descentralizada de energia, constituindo os chamados sistemas de geração distribuída, ou *distributed generation*. Juntamente desses sistemas, cresce o interesse em desenvolver redes elétricas com conexão de forma ainda mais flexíveis, possibilitando a operação autônoma e eficaz, definindo assim as microrredes ou *micro grid*, que ao interagirem com o comportamento e ações dos usuários conectados a rede recebendo o nome de redes inteligentes, ou *smart grids* (VASQUEZ et al., 2010). Estes três conceitos definem ferramentas de forte expansão que estão sendo utilizadas para minimizar os problemas de energia sendo o foco deste trabalho.

Para a implementação dos métodos que viabilizam a construção tanto das microrredes quanto das redes inteligentes, é imprescindível a utilização de sistemas de comunicação e aquisição de dados, que tornam possível a interação entre os diferentes componentes dos sistemas, assim como a interpretação dos parâmetros adquiridos. Com a aquisição e interpretação dos dados é possível atuar nos componentes das redes elétricas, de forma a atingir o nível desejado e recomendado para o bom desempenho da mesma. Com o aumento dos métodos de comunicação em redes foi necessário criar padrões para que o usuário pudesse utilizar vários tipos de dispositivos em diversas aplicações. Neste cenário entram os protocolos de comunicação que regulamentam o acesso dos componentes conectados à rede.

Este capítulo visa definir os conceitos básicos para poder compreender os componentes de uma microrrede de energia e a estrutura para a comunicação deste sistema.

2.1 GERAÇÃO DISTRIBUÍDA

A vulnerabilidade dos sistemas de distribuição de energia, com relação às catástrofes naturais e também a necessidade de existir energia em locais isolados, resultaram em uma busca de sistemas capazes de tornar o acesso à energia mais facilitado, proporcionando maior independência na geração de energia. Neste caso, o modo de geração distribuída pode ser uma, solução, pois proporciona que a geração de energia ocorra em diversos locais utilizando geralmente fontes energéticas com base em energia hidráulica, solar, eólica, biomassa ou cogeração qualificada (VASQUEZ et al., 2010).

Os sistemas de geração distribuída, têm como principais benefícios, proporcionar ao sistema elétrico a postergação de investimentos em expansão dos sistemas de distribuição e transmissão, redução no impacto ambiental (exceto em casos em que a geração de energia se dá a partir de fontes não-renováveis, redução no carregamento das redes), redução de perdas com transmissão e distribuição e diversificação da matriz energética (ANEEL, 2017). Estes efeitos contribuem para a minimização das crises energéticas, que por exemplo costumam acontecer em períodos de seca nas regiões de usinas hidrelétricas.

2.2 MICRORREDES

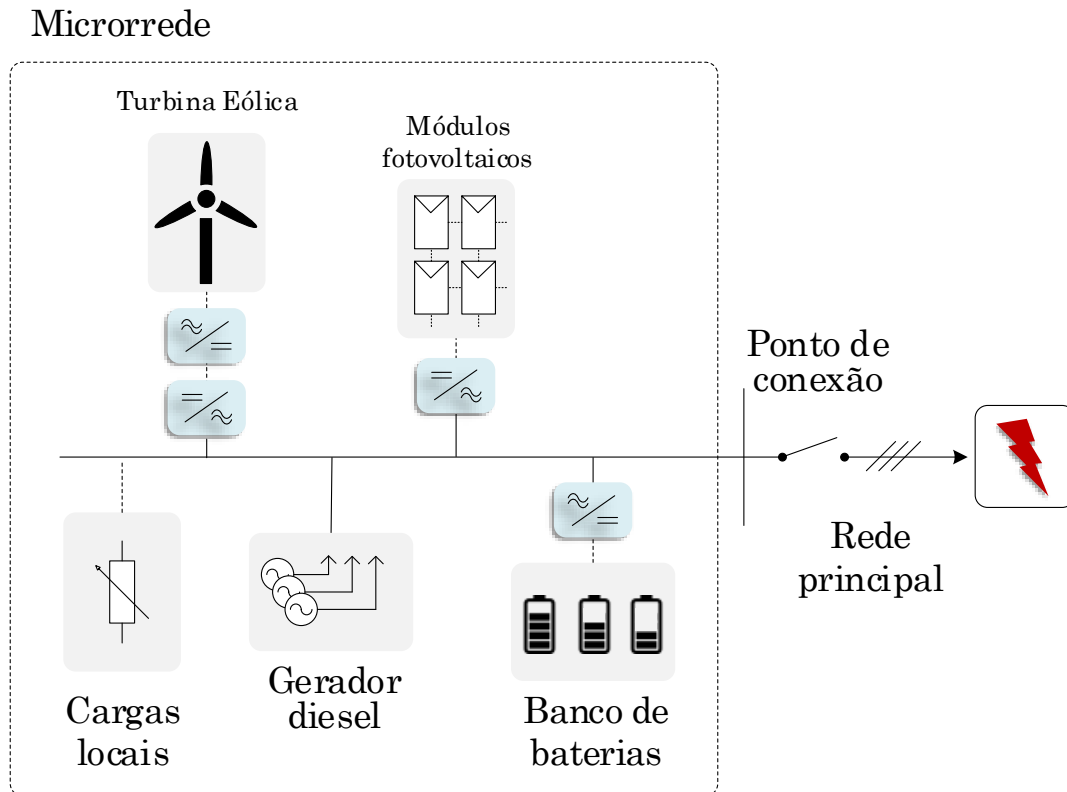
Hoje a maior parte da energia gerada é produzida em usinas de geração centralizadas e distribuída através de linhas de transmissão e distribuição. Entretanto, sabe-se que a produção e a distribuição de eletricidade de forma integrada, confiável e sustentável são apontadas como uma necessidade para o futuro, com o intuito de automatizar o monitoramento de redes de energia e aumentar a qualidade da energia consumida. As microrredes constituem em sistemas que proporcionam a interação entre as camadas de geração distribuída e variadas cargas, como proposto por Guerrero et al. (2010), ou seja são centros de geração menores e geralmente próximos à carga.

Vandoorn et al. (2013), definem que uma microrrede pode ser identificada por três principais características. Primeiramente, em comparação às redes convencionais, pode-se apontar que os elementos de uma microrrede são principalmente compostos por dispositivos eletrônicos, sendo assim, ao ocorrer um desbalanceamento da rede, o balanceamento não ocorre pela inércia da rotação na geração, mas sim pela ação dos conversores eletrônicos. Como segunda característica destaca-se que em redes de distribuição de baixa tensão a impedância da linha é predominantemente resistiva, já nas redes de transmissões a impedância da linha é predominantemente indutiva. Por fim, temos que grande parte das microrredes podem ser alimentadas por energias renováveis que necessitam de sistemas de controles atualizados devido ao uso de novas tecnologias.

A Fig. 2.1 mostra um exemplo de microrrede formada por fontes de energia e cargas. Nesta ilustração é possível observar o papel efetuado pelos inversores e conversores de potência, permitindo que a energia das fontes sejam inseridas na rede de distribuição local, alimentando cargas pertencentes à microrrede.

Existem dois modos de operação das microrredes, modo ilhado e modo conectado à rede principal. O modo ilhado a microrrede trabalha de modo independente da rede e de fontes geradoras fora da mesma. Por definição, a microrrede pode transitar entre os

Figura 2.1 – Exemplo de estrutura de uma micro rede de energia.

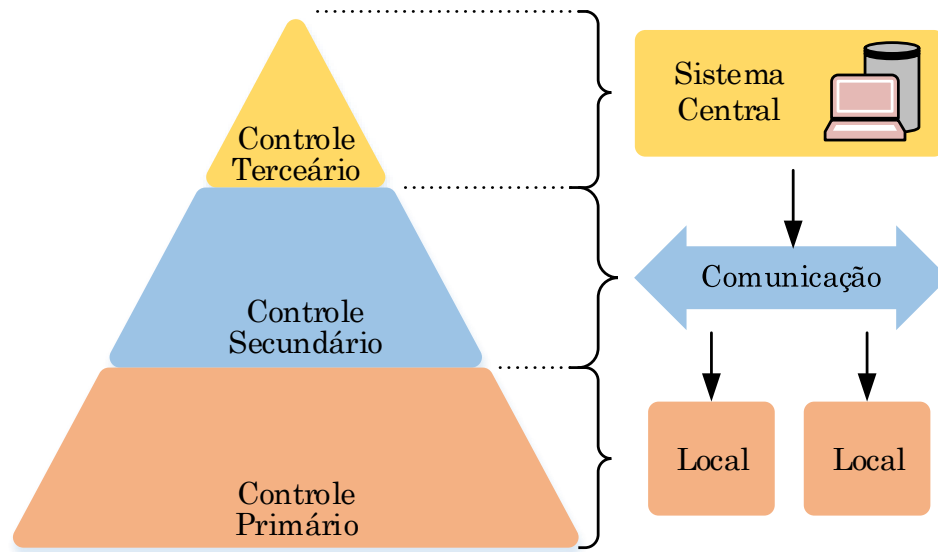


Fonte: Adaptado de (VASQUEZ et al., 2010)

dois modos conforme estado da rede, como por exemplo, no caso de faltas nas rede de distribuição. (VASQUEZ et al., 2010).

A Fig. 2.2 mostra a divisão em três principais controles de uma microrrede definida por Vasquez et al. (2010). O controle primário é responsável pelo controle da tensão local, assegurando o compartilhamento de carga entre todas as unidades geradoras além de ser responsável pela reabilitação do sistema em caso de falhas. O controle secundário é responsável pelo monitoramento e gerenciamento da microrrede, sendo responsável por garantir as especificações de tensão e frequência. O controle terciário relaciona a economia e otimização, é o qual que controla a qualidade do acoplamento. Analisando a divisão em camadas proposta e aplicando para este trabalho, observa-se que o sistema proposto visa aplicação na segunda camada onde há interações entre os dados, através do CCS, buscando a melhor operação do sistema como um todo.

Figura 2.2 – Divisão dos controles de uma microrrede.



Fonte: Adaptado de (VASQUEZ et al., 2010)

2.3 REDES INTELIGENTES

A geração de energia de modo “inteligente” refere-se a fazer uso mais eficiente da energia, utilizando a integração de tecnologias avançadas. Yu et al. (2011) defende que as redes inteligentes são necessárias em razão da disponibilidade limitada de fontes não renováveis.

As redes inteligentes, ou *smart grids*, podem ser definidas como redes elétricas que suportam a geração de energia provendo o controle e acompanhamento da mesma, integrando em seu controle o comportamento e ação do usuário permitindo uma distribuição de forma sustentável. Estas formam um sistema de distribuição e de transmissão de energia elétrica, capazes de operar uma composição de várias microrredes, bem como plantas com variadas fontes de geração de energia e com diferentes tipos de carga. Com isso, as smart grids proporcionam não só maior aproveitamento da energia, mas também um controle de fluxo de energia eficaz, que garante a segurança e operabilidade das linhas de distribuição (VASQUEZ et al., 2010).

Como o termo de uso inteligente da energia depende de diversos fatores, sua definição pode mudar conforme mostrado por Yu et al. (2011). Observou-se que nos Estados Unidos tem-se o termo rede inteligente como os meios de descentralizar a indústria de eletricidade, ou seja, depender mais dos consumidores gerarem sua própria energia. Já na China, o conceito de rede inteligente, refere-se a geração de energia mais segura, confiável, responsável e economicamente viável de uma forma sustentável. Na Europa o conceito

remete a maior participação dos cidadãos em energias renováveis. O conceito no Brasil, ainda em desenvolvimento, pode ser observado em Herzog (2013), onde indica que implantar uma rede inteligente é baseado na troca dos medidores eletromecânicos de consumo dos clientes por medidores digitais inteligentes, uma visão ainda um pouco limitada visto a complexidade deste conceito.

O sistema proposto neste projeto não oferece uma *smart grid* completa, mas fornece parte das ferramentas que podem ser utilizadas para o seu desenvolvimento, que são o hardware e software de comunicação entre os dispositivos para ocorrer uma interação entre os mesmos.

2.4 PROTOCOLOS E MODELOS DE COMUNICAÇÃO

Obter uma comunicação com êxito entre dispositivos sem uma regra seria difícil. Em uma comunicação entre duas pessoas, por exemplo, se ambas falarem ao mesmo tempo e em idiomas diferentes não haveria um entendimento e a comunicação falharia. Para sanar este e outros problemas foram criados alguns dos modelos de comunicação, normas e protocolos dos quais são apresentados a seguir.

Nesta seção, inicialmente há uma exposição dos padrões seriais RS que são fortemente utilizados nos dispositivos frequentemente utilizados em microrredes, em seguida é apresentada a estrutura Modbus que serve como base para a criação e desenvolvimento do software e protocolos propostos por este trabalho.

2.4.1 Padrão serial RS

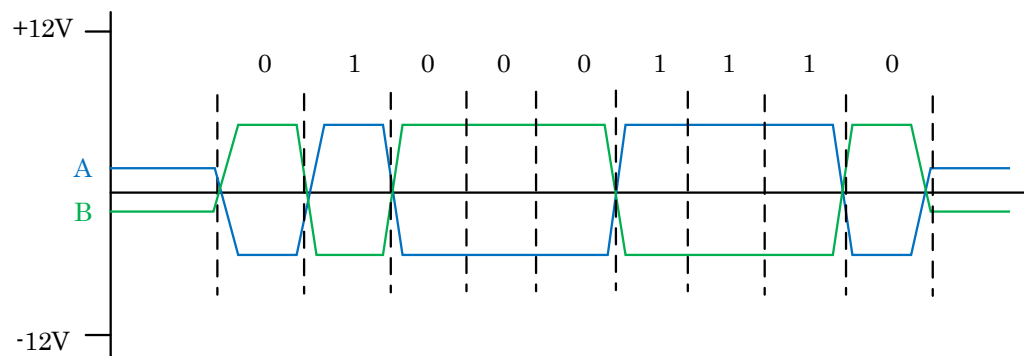
Os padrões de interface serial RS¹ surgiram para a uniformização de interface de comunicação de dados entre equipamentos, definidos pela *Electronic Industries Alliance*, uma organização composta para definição de padrões associados a eletrônicos dos Estados Unidos. Estes, não definem qual o protocolo a ser utilizado para a comunicação dos dados, mas a camada física da comunicação. Neste capítulo são apresentados dois tipos de padrões o RS-485 e RS-232. Ambos os tipos estão presentes no sistema constituído, o padrão RS-485 no inversor solar e RS-232 no DSC.

¹A sigla RS significa *Recommended Standard*, ou seja padrão recomendado.

2.4.1.1 Padrão RS-485

O protocolo de comunicação RS-485 utiliza dois pares trançados de cabos que transmitem o sinal e possibilitam a obtenção de um diferencial de tensão que determina o valor em binário do sinal. A Fig. 2.3 exemplifica a transmissão de uma mensagem, onde através da diferença entre os dois níveis de tensão é identificado.

Figura 2.3 – Exemplo de transmissão de dados utilizando o padrão RS-485.



Fonte: Elaborado pelo autor (2018).

É importante salientar que o padrão não define níveis de tensão máximos ou mínimos a serem utilizados pois a identificação do nível lógico ocorre conforme a diferença dos sinais de tensão padronizados conforme descrito na Tabela 2.1. Porém, MARAIS (2008), indica que geralmente os valores máximos e mínimos ficam entre valores de tensão de -12 e +12V.

Tabela 2.1 – Níveis de tensão de diferença entre sinais padronizados para o padrão RS-485.

Valor da tensão	Valor em binário
$A-B < 200\text{mV}$	0
$A-B > 200\text{mV}$	1

Fonte: Elaborado pelo autor (2018).

Uma desvantagem deste modelo é o uso de mais cabos, devido a comparação entre dois sinais, porém ao mesmo tempo isso permite o aumento da máxima taxa de transmissão. Ainda, neste tipo de conexão o dispositivo aciona o seu transmissor apenas no instante que necessita transmitir um dado, mantendo desligado no resto do tempo, o que permite a conexão de vários dispositivos em uma mesmo barramento (INSTRUMENTS, 2013).

2.4.1.2 Padrão RS-232

A principal diferença entre RS-485 e RS-232 é a relação de tensão. Diferente da RS-485 onde há um diferencial de tensão, no padrão RS-232 o sinal é comparado com a referência (GND). A Fig. 2.4 exemplifica o uso do padrão transmitindo o valor 47 em hexadecimal utilizando 7 bits, com start bit de nível lógico “0”, paridade par, e dois stop bit com nível lógico “1”, configuração usualmente utilizada para este tipo de comunicação.

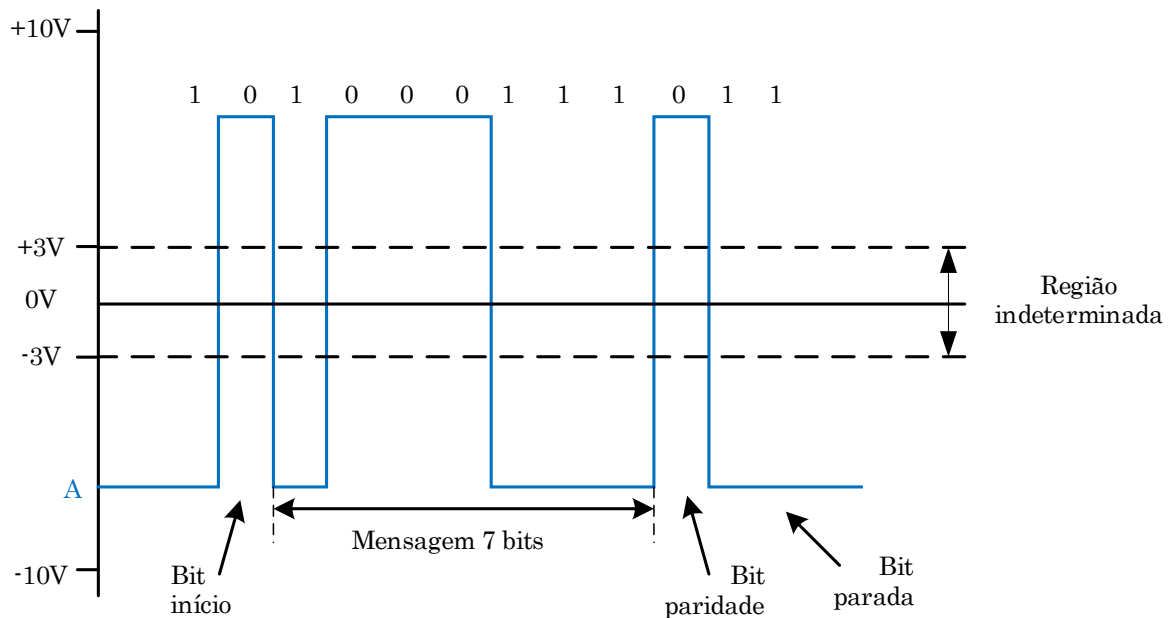
Neste padrão de comunicação, considerando um sinal com valor de tensão “A”, a identificação do valor binário do sinal segue por padrão os valores indicados na Tabela 2.2, onde os valores máximos não ultrapassam de -10V e +10V. (SMARTWORX, 2014).

Tabela 2.2 – Níveis de tensão padronizados pelo padrão RS-232.

Valor da tensão	Valor em binário
$A > +3V$	0
$-3V \text{ a } +3V$	Indefinido
$A < -3V$	1

Fonte: Elaborado pelo autor (2018).

Figura 2.4 – Exemplo de transmissão de dados em uma comunicação utilizando RS-232.



Fonte: Elaborado pelo autor (2018).

2.4.2 Modbus

O protocolo Modbus foi desenvolvido em 1979 para possibilitar a comunicação entre dispositivos de automação. Atualmente este tipo de comunicação é utilizado em diversos dispositivos por ter como principal vantagem a capacidade de prover uma comunicação simplificada de forma confiável (INSTRUMENTS, 2014).

A comunicação em Modbus é efetuada através da arquitetura mestre-escravo, na qual o equipamento mestre transmite uma requisição ao equipamento denominado escravo que processa a solicitação, respondendo ou não ao mestre. A comunicação sempre inicia com o mestre sendo que os escravos não possuem uma comunicação direta entre eles e somente enviam respostas as solicitações do mestre.

Uma requisição realizada pelo mestre, pode ser realizada de duas maneiras. O primeiro modo é o *unicast* onde o mestre envia a mensagem diretamente para o escravo. O segundo modo é chamado de *broadcast*, onde o mestre envia a mensagem para todos os escravos.

A rede estabelecida por este protocolo pode ter um máximo de 247 dispositivos conectados, sendo que cada um, deles deve ter um endereço de “1” à “247”, onde o valor “0” é identificado como endereço de broadcast que todo dispositivo deve reconhecer caso seja requisitado (MODBUS, 2006).

A definição dos bytes contidos no protocolo é apresentada na Tabela 2.3. O primeiro conjunto de bytes indica os endereços do dispositivos, seguido dos campos de *function code* e *data* que podem ser variados conforme aplicação do usuário. Para a detecção de erro deve ser utilizado o método CRC² ou LRC³.

Tabela 2.3 – Estrutura dos bytes do protocolo Modbus.

Endereço	Function code	Data	CRC (or LRC)
----------	---------------	------	--------------

Fonte: Adaptado de Modbus (2006)

Para a transmissão dos dados dois modos de transmissão são definidos: RTU (*Remote Terminal Unit*) e ASCII (*American Standard Code for Information Interchange*), que devem ser únicos em uma única rede. Todos os dispositivos obrigatoriamente devem ter a transmissão RTU, fazendo com que a ASCII seja opcional. As principais diferenças dos modos de transmissão são apresentadas na Tabela 2.4.

²CRC, do inglês *Cyclic Redundancy Check* ou verificação cíclica de redundância.

³LRC, do inglês *Longitudinal Redundancy Check* ou checagem longitudinal de redundância.

Tabela 2.4 – Principais diferenças entre modo de transmissão para Modbus RTU e Modbus ASCII.

Característica	Modo RTU	Modo ASCII
Sistema de codificação	Binário	Hexadecimal (ASCII)
Início de mensagem	Intervalos de silêncio	“:” em ASCII
Bits de dados	8 bits	7 bits
Verificação de erros	CRC	LRC
Fim de mensagem	Intervalos de silêncio	“Carriage return – Line feed” em ASCII

Fonte: Elaborado pelo autor (2018).

A Tabela 2.5 apresenta a estrutura típica das mensagens nos modos ASCII e RTU. Em ambos os modos o protocolo Modbus usa um indicador de início e fim da mensagem, que permite que o dispositivo organize a transmissão de dados. Para o modo ASCII uma mensagem inicia com o símbolo (:) que representa “3A hex”. No fim de mensagem o modo utiliza o “Carriage return e Line feed” que são representados por “0D e 0A hex”. Já no modo RTU uma mensagem é iniciada e finalizada com um intervalo de silêncio pré-determinado de 3,5 caracteres de tempo (na Tabela 2.5 exemplificado como T1–T2–T3–T4).

Tabela 2.5 – Estrutura típica para comunicação Modbus em modo ASCII e RTU.

Modo ASCII					
Start	Address	Function	Data	LRC Check	End
1 Char (:))	2 Chars	2 Chars	$n \times Chars$	2 Chars	2 Chars (CRLF)
Modo RTU					
Start	Address	Function	Data	CRC Check	End
T1–T2–T3–T4	8 Bits	8 Bits	$n \times 8$ Bits	16 Bits	T1–T2–T3–T4

Fonte: Adaptado de Modicon INC. (1996).

Apesar de ser um protocolo com vantagens e geralmente utilizado na indústria, a aplicação do modelo exato desse protocolo neste projeto causaria uma complexidade demasiada. Essa dificuldade se dá principalmente devido a as regras de sincronia de tempos definidos pelo fabricante. Desta forma, foram criados protocolos baseados neste para a utilização, como é apresentado no Capítulo 4. A opção de criar um protocolo próprio baseado no Modbus, não é incomum, como encontrado em diversas fabricantes e que é discutido também no Capítulo 4.

2.5 INTERNET DAS COISAS (IOT)

Com o crescimento de novos tipos de dispositivos cresceu também a necessidade do ser humano de procurar formas de facilitar seu trabalho e ter mais controle sobre os processos. O paradigma da internet das coisas (*Internet of Things* - IoT) promete fazer “coisas”, como dispositivos eletrônicos ou aplicações caseiras, parte da rede de internet, com o objetivo de melhorar a qualidade de vida dos usuários e melhor utilização dos recursos. Devido ao desenvolvimento da tecnologia, o processo de expansão deste tipo de sistema vem crescendo. Estima-se que as IoTs poderão ter um impacto econômico de 11 trilhões de dólares em 2025, o que representa a cerca de 11% da economia mundial, esperando que nesse mesmo período já estejam desenvolvidos cerca de 1 trilhão de dispositivos, como apresentado por Buyya e Dastjerdi (2016).

Um sistema IoT pode trazer diversos benefícios como: facilitar a interação entre homem-máquina e próprios dispositivos, tornando o acesso mais amplo; Permite expandir os canais de comunicação facilitando o acesso a sensores e dispositivos em geral, melhorando a obtenção de diversos tipos de dados; Pode diminuir custos de manutenção em casos em que o sistema faça uma checagem do status dos componentes de maneira remota.

O conceito de IoT pode ser diretamente ligado a este projeto e principalmente a sua expansão. Os dados coletados pelo software atualmente tem acesso local, porém para serem adquiridos é utilizado uma placa de comunicação que obtém os dados via rede local, que pode ser expandida em futuros trabalhos para acesso remoto.

2.6 REDE SEM FIO - WIFI

As redes de dispositivos foram criadas com a característica de facilitar a execução de operações entre os equipamentos que a ela compõem. A conexão não precisa ser feita necessariamente por cabos, podem ser utilizados outros meios como microondas, fibra óptica, ondas infravermelhas, entre outros.

No início do desenvolvimento das redes de comunicação o IEEE (*Institute of Electrical and Electronics Engineers*) procurou as padronizar. É o caso dos padrões 802.3 (Ethernet) e o 802.11 (LAN sem fio). A rede 802.11 foi definida para operar em dois modos: O primeiro ocorre na presença de uma estação-base, como por exemplo, quando um computador é conectado a um roteador, que conecta-se a outros dispositivos; O segundo caso é na ausência de uma estação base, onde é utilizada uma comunicação direta entre os dispositivos, como ocorre no caso do controle de alguns drones em curta distância, por exemplo (FRANCISCATTO; CRISTO; PERLIN., 2014).

A rede Wi-fi[®] é um padrão baseado em 802.11, se tornou muito popular na utilização em dispositivos para comunicação devido a permitir mobilidade, ter baixo custo e grande aceitação em diversos produtos. Uma marca registrada da Wi-Fi Alliance, que certifica os produtos e define padrões quanto a transmissão de dados.

A vantagem da utilização de uma rede sem fio para a aplicação proposta se dá justamente por permitir mais mobilidade devido a não conter fiação entre dispositivo e roteador. Em contraponto ruídos eletromagnéticos podem causar perdas na qualidade de sinal.

3 HARDWARE

Um dos primeiros componentes a ser definido para a rede proposta é o hardware a ser utilizado, pois através deste que são baseados os softwares e protocolos utilizados. Para o uso neste trabalho, dois principais tipos de dispositivos são o foco: um inversor solar e DSC. O inversor solar vem ganhando cada vez mais uso no mercado e utilização nas microrredes de energia devido principalmente a facilidade de instalação deste sistema e estar provendo uma geração de forma mais sustentável. O DSC, um controlador de sinais digitais, vem ganhando espaço devido a facilidade de adaptação para diversos tipos de aplicações em processamento de energia.

Para a comunicação entre os hardwares mencionados e o software da comunicação, algumas conversões são necessárias e estas ocorrem através das placas de comunicação. Este trabalho apresenta dois tipos de placas de comunicação para dispositivos do tipo RS-485: A primeira placa consiste em uma conversão da comunicação RS-485 para RS-232. A segunda placa estabelece uma comunicação sem fio utilizando o microcontrolador ESP8266. Para os dispositivos que utilizam a comunicação do tipo RS-232 foi desenvolvida uma placa que também transmite os dados via conexão sem fio.

3.1 ESP8266

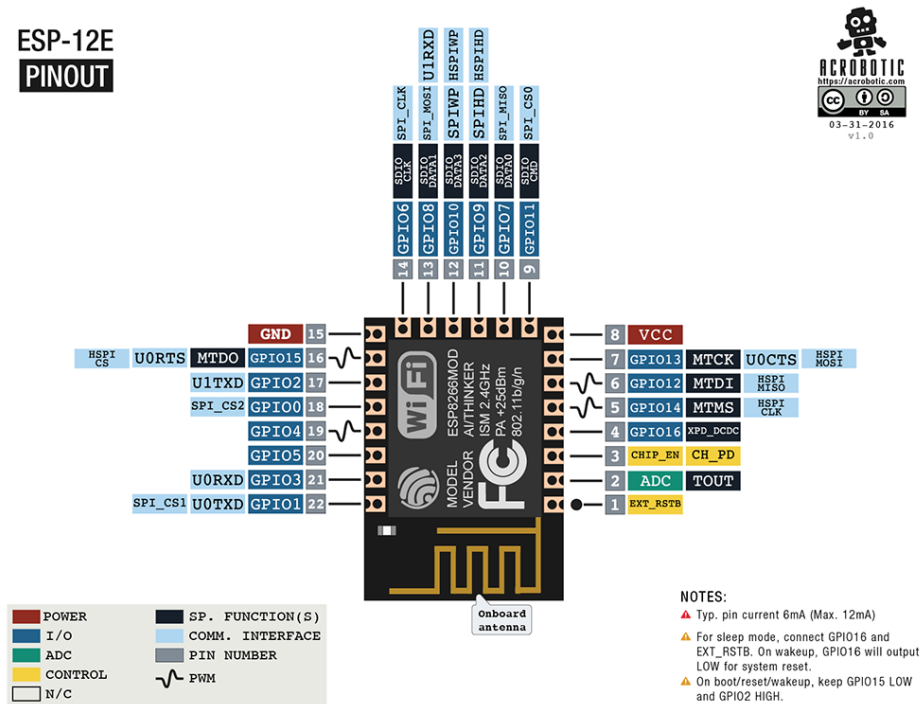
O ESP8266 é um microcontrolador fabricado pela Espressif[®], e muito utilizado em sistemas IoTs, devido sua simplicidade de configuração para realizar uma comunicação sem fio e baixo custo. De acordo com a própria fabricante, ele entrega uma solução em conexão Wifi procurando eficiência em relação ao uso da energia em um compacto design, proporcionando uma performance confiável para o uso em aplicações de Internet das coisas (IoT) (ESPRESSIF, 2017).

A principal vantagem deste microcontrolador, se comparados a outros produtos similares do mercado, é o baixo custo. Kodali e Naikoti (2016), comparam os valores de vários dispositivos com funções semelhantes. Enquanto o ESP8266-12E tem o valor de aproximadamente \$7, uma placa Zigbee custa em torno de \$25 e um shield Ethernet para Arduino \$60, o que apresenta uma diferença considerável.

A Fig. 3.1 mostra o esquema da versão “12E” do microcontrolador. Esta versão, tem o tamanho de 24mm×16mm contendo uma memória flash de 4MB, baixo consumo

de energia e 22 pinos que proporcionam entradas e saídas digitais, comunicação serial e uma entrada analógica.

Figura 3.1 – Pinagem do módulo ESP8266-12E.



Fonte: Acrobotic (2015).

Para a programação do firmware do microcontrolador é utilizada uma biblioteca externa que permite operar com o *Arduino Software IDE*¹, proporcionando um ambiente amigável e utilização da linguagem de programação C.

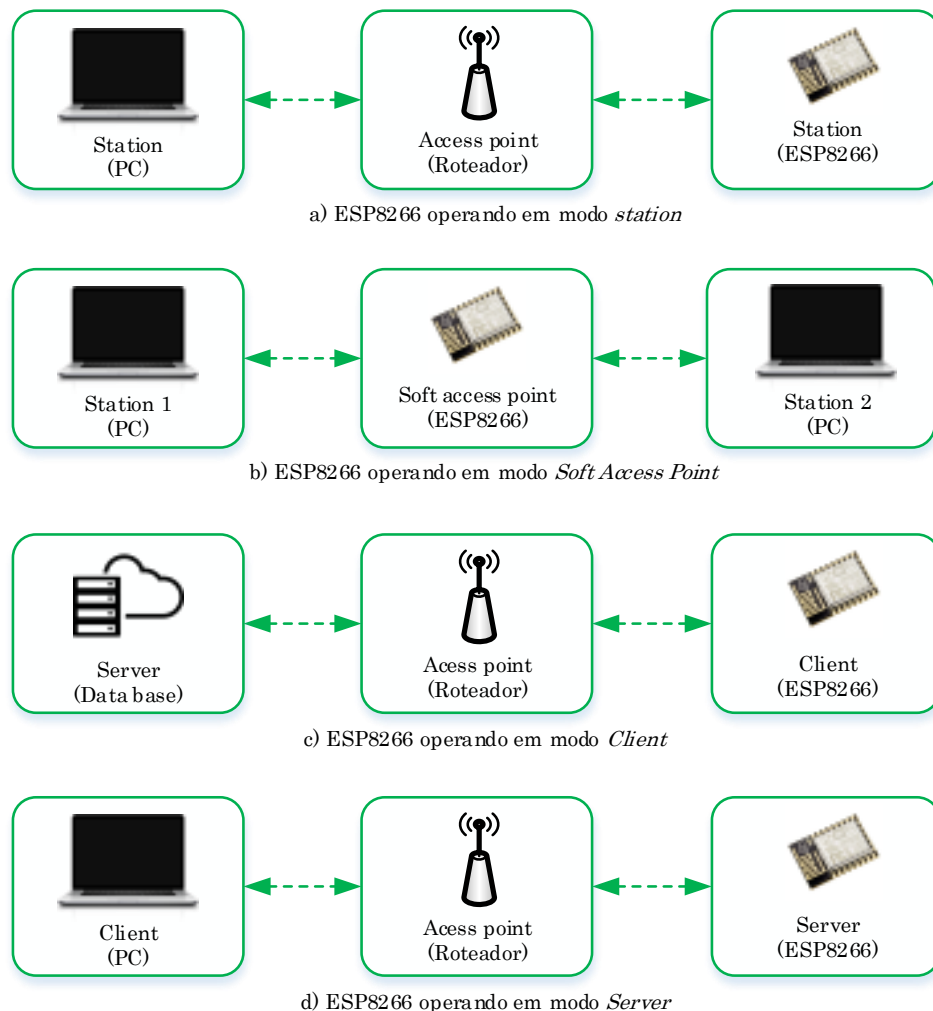
3.1.1 Modos de operação do ESP8266

O microcontrolador pode operar de diferentes modos dependendo da função que o dispositivo executa na rede. Github (2016b) classifica os dispositivos com uma nomenclatura específica conforme sua função. Os dispositivos que são conectados a rede Wi-fi são chamados de *station*, e os dispositivos que provém a conexão recebem o nome de chamado *access point*. Já os dispositivos que conectam-se a servidores, recebem a nomenclatura de *client*. A Fig. 3.2 ilustra os diferentes modos em que o ESP8266 pode operar conforme os termos:

¹Open-source Arduino Software (IDE) é uma plataforma utilizada para fazer upload de códigos em placas desenvolvida pela Arduino[®]

- **Modo *station*** - O ESP8266 está conectado a rede provida de um *access point* e a utiliza para comunicar com os demais dispositivos;
- **Modo *soft access point*** - O ESP8266 fornece uma conexão Wi-fi, conectando-se com no máximo cinco dispositivos;
- **Modo *client*** - O ESP8266 está conectado a um servidor conectado a rede e acessa dados deste;
- **Modo *server*** - O ESP8266 é o servidor e provém aos outros dispositivos as informações armazenadas no mesmo.

Figura 3.2 – Modos de operação do ESP8266 conforme tipo de utilização.



Fonte: Adaptado pelo autor de Github (2016b).

Para este projeto foi selecionado o modo de operação *station*, pois na rede proposta o ESP8266 opera como uma ponte para os dados, transferindo o que foi recebido via serial para o software de monitoramento e o que foi recebido do software de monitoramento para a serial. Este modo ainda permite que caso a conexão seja perdida o microcontrolador

irá automaticamente reconectar ao último *access point* conectado e o mesmo ocorre se o microcontrolador é reiniciado.

3.2 PHB3000-SS

O PHB3000-SS é um inversor solar produzido pela PHB Solar que tem como parâmetros 3200VA/500V/18A, contendo funções como anti-ilhamento e monitoramento de corrente de fuga. O inversor é dimensionado para conexão com uma rede elétrica de 220V@60Hz, entretanto sua alimentação é proveniente da tensão CC e sua operação se inicia quando a tensão atinge o valor de 125 V. O equipamento proporciona duas saídas RS485 que são utilizadas na comunicação com demais dispositivos conectados, com a taxa de transmissão de dados a 9600bps, através de um protocolo próprio, que será discutido em detalhes na seção 4.5. O sistema de comunicação opera no modelo de comunicação mestre-escravo com uma estrutura de protocolo similar ao protocolo Modbus. Através de um aplicativo supervisor, pode-se efetuar a leitura de parâmetros tais como: tensão e corrente nos painéis, tensão, corrente e potência nas fases; potência fornecida; temperatura do inversor; indicador de erros; potência total de alimentação na rede; total de horas de alimentação; potência fornecida para rede no dia.

Conforme mencionado anteriormente, para a comunicação do inversor solar com o computador duas placas de circuito impresso foram criadas. Uma placa que converte a comunicação do padrão RS-485 para RS-232 e outra que realiza a comunicação com um microcontrolador que envia os dados via rede Wifi e estas serão discutidas a seguir.

3.2.1 Placa de conversão RS-485/RS-232

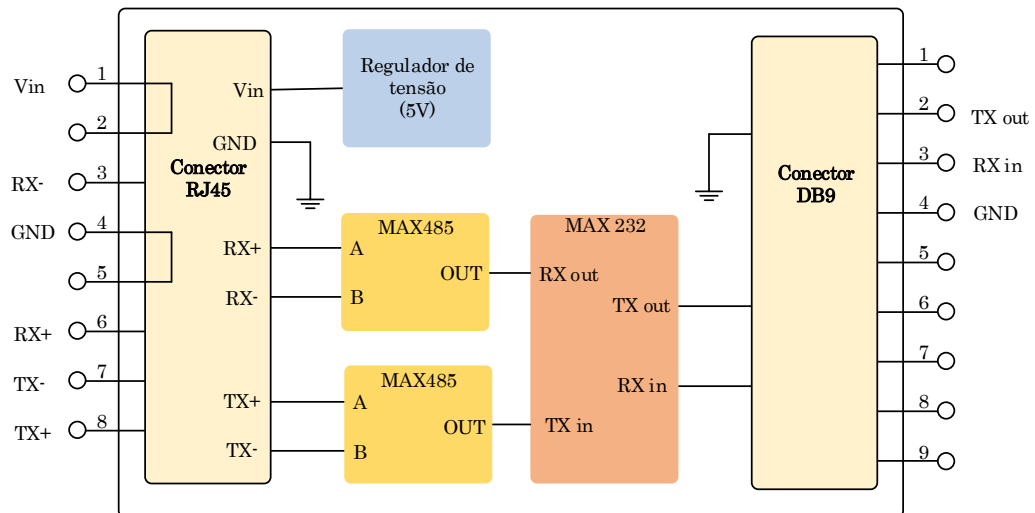
Os inversores PHB, assim como alguns outros dispositivos, utilizam uma comunicação RS-485 que muitas vezes não pode ser lida diretamente através de um computador. Para resolver este problema, foi desenvolvida uma placa de circuito impresso que proporciona a conversão de RS-485 para RS-232 que através um conversor comercial RS-232/USB pode ser conectado aos computadores e notebooks atuais.

Na Fig. 3.3 é apresentado o diagrama de blocos para a criação do conversor RS-232/RS-485. O circuito proposto basicamente contém três componentes principais:

- **Regulador de tensão** - Assegura que a tensão permaneça regulada em 5V para demais componentes;
- **MAX485** - É um transceptor para comunicação RS-485 que converte o sinal recebido por um par diferencial, entradas “A” e “B”, em uma única saída, “OUT”.

- **MAX232** - É um transceptor que converte níveis de tensão focado em uso de portas seriais. O receptor converte a tensão de entrada, que pode variar $\pm 30V$, para $\pm 5V$.

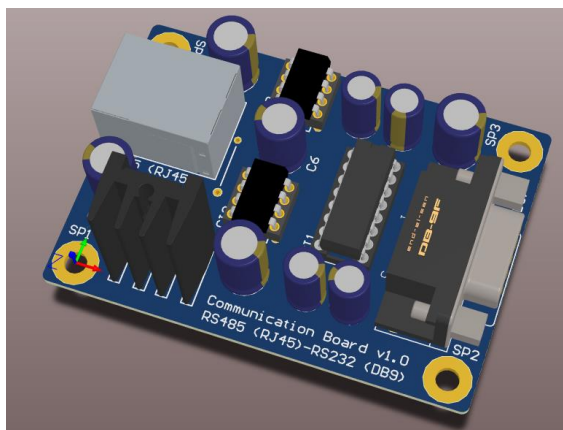
Figura 3.3 – Diagrama de blocos do conversor RS-232/RS-485.



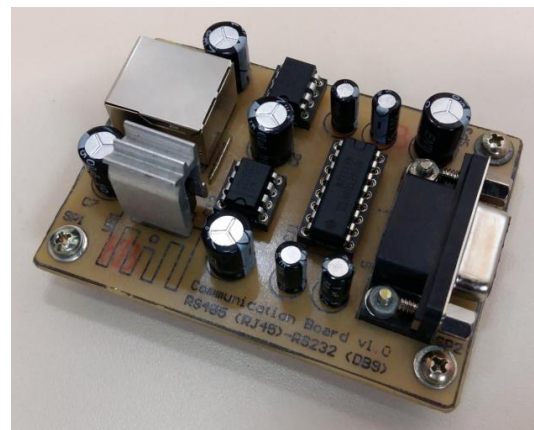
Fonte: Elaborado pelo autor (2018).

Com o diagrama apresentado, foi criada a placa mostrada na Fig. 3.4. A figura “a” apresenta o projeto da placa e a figura “b” a montagem.

Figura 3.4 – Projeto em 3D (a) e placa de circuito impresso produzida (b) da placa de conversão RS485/RS232.



a)



b)

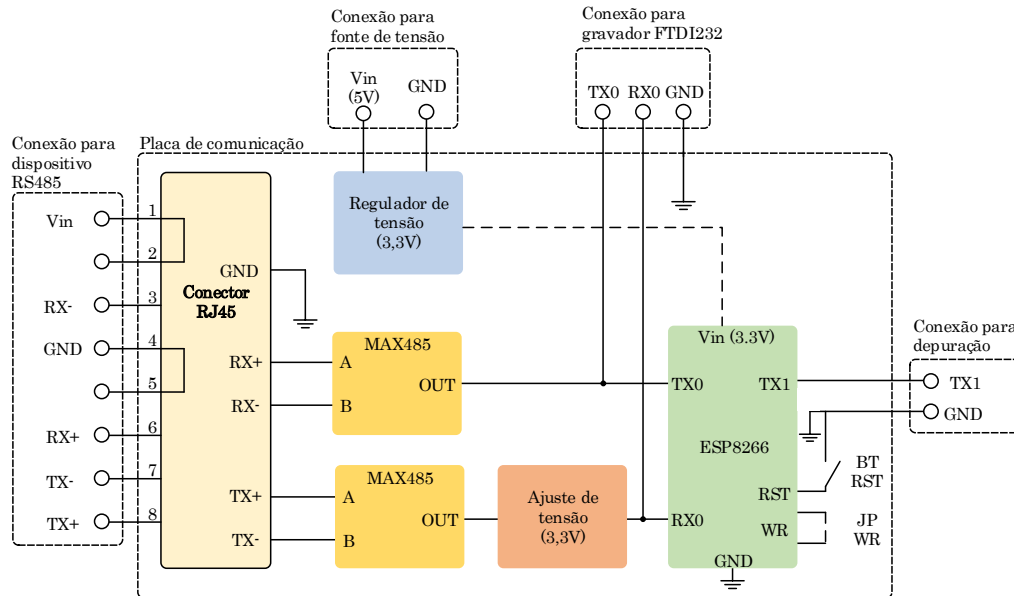
Fonte: Elaborado pelo autor (2018).

3.2.2 Placa de comunicação Wi-fi para dispositivos RS-485

Através da placa apresentada anteriormente, foi desenvolvida uma placa que permite uma comunicação sem fio entre dispositivos com conexão RS-485 utilizando o microcontro-

lador ESP8266 integrado a mesma. Como o inversor solar utilizado neste projeto contém a característica desta conexão, esta placa foi utilizada como meio para monitorar os dados do mesmo.

Figura 3.5 – Diagrama de blocos da placa de comunicação para dispositivos RS485.



Fonte: Elaborado pelo autor (2018).

Na Fig. 3.5 é apresentado o diagrama de blocos da placa de comunicação que contém os seguintes componentes:

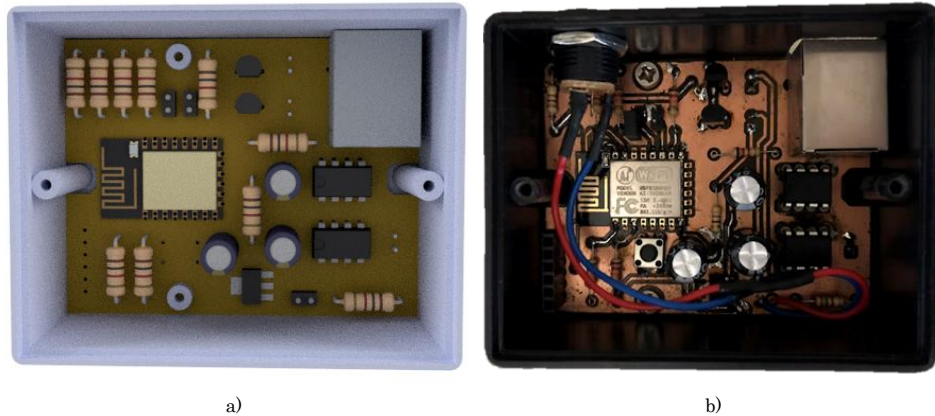
- **ESP8266** - Microcontrolador utilizado para comunicação sem fio. Neste há conectado, além da ligação de comunicação e alimentação, um botão (BT RST) que reinicia o dispositivo ao ser pressionado e um jumper (JP WR) que é utilizado para gravar o microcontrolador.
- **Regulador de tensão** - Através da entrada de 5V regula a tensão de 3,3V na saída que é utilizada para alimentação do microcontrolador e demais CIs;
- **Ajuste de tensão (3,3V)** - Através de um circuito composto por transistores e resistores reduz a tensão recebida do MAX485 para 3.3V. Similar a função realizada na placa apresentada anteriormente pelo MAX232.

O dispositivo a ser monitorado é conectado através da saída via conector RJ45. A alimentação da placa ocorre com uma alimentação de 5V e corrente de 1A que é fornecida através de uma conexão para fonte externa AC/DC.

As portas RX0 e TX0 são utilizadas para a gravação do firmware no microcontrolador, utilizando um gravador FTDI232. Neste processo, deve-se realizar a conexão do jumper, *JPWR*, indicada na figura Fig. 3.5. A depuração do firmware do ESP8266 pode ser realizada pela conexão serial através da conexão “TX1”.

Através do circuito proposto foi desenvolvida uma placa de circuito impresso, que pode ser vista na Fig. 3.6 onde é comparado o modelo 3D e montagem na prática.

Figura 3.6 – Comparação entre projeto 3D (a) e placa montada (b) da placa de comunicação para dispositivos RS485



Fonte: Elaborado pelo autor (2018).

3.3 DIGITAL SIGNAL CONTROLLER - DSC

O DSC, da sigla *Digital Signal Controller*, é um controlador de sinais digitais que processa e executa ações com sinais digitais amostrados em suas entradas analógicas e digitais. É amplamente utilizado em projetos que necessitam ações de controle, como no caso de conversores e outros dispositivos.

Para simular o uso deste dispositivo na rede, foi utilizado o hardware de desenvolvimento “LAUNCHXL-F28377S” da fabricante Texas[®], que traz as vantagens de simular a utilização de um DSC F2837xS, também da fabricante. A ferramenta permite uma fácil conexão via USB com o software de programação² utilizado para depuração do firmware.

Para comunicar este hardware com o software de monitoramento proposto é necessário de um meio entre placa e computador. Foi optado utilizar uma de suas saídas digitais conectada a uma placa de comunicação criada que permite a conexão de um dispositivo utilizando o padrão de comunicação RS-232 e será discutida a seguir.

3.3.1 Placa de comunicação Wi-fi para dispositivos RS-232

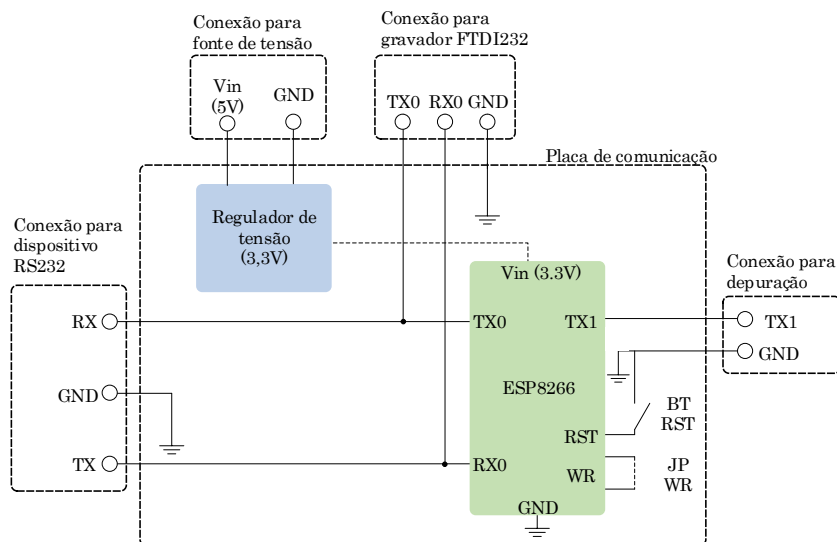
Da mesma forma que foi elaborada para a comunicação com os dispositivos RS-485, uma placa de comunicação foi criada para dispositivos do tipo RS-232. Com esta, o DSC escolhido pode-se conectar a rede de monitoramento. A diferença entre esta placa e as

²A fabricante indica o uso do software *Code Composer Studio* para a programação do firmware.

apresentadas anteriormente é que o padrão RS-232 tem uma comunicação através apenas de um canal e não um par de sinais, assim não sendo necessária a conversão de sinal para enviar ao microcontrolador, tornando a placa simplificada.

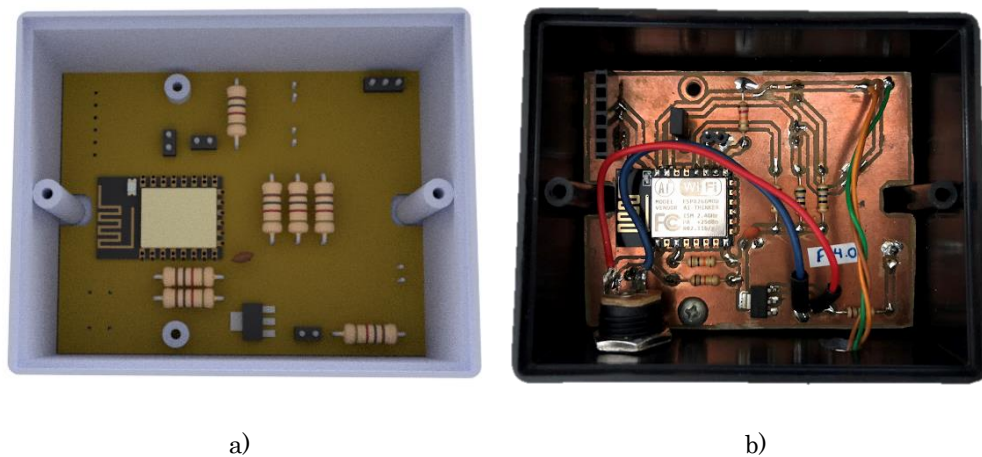
A Fig. 3.7 mostra o diagrama de blocos do funcionamento básico da placa de comunicação para os dispositivos RS-232, que contém apenas o ESP8266 e componentes para seu funcionamento. Esta placa ainda conta com um botão para o reset do microcontrolador e um jumper para a gravação do firmware. A placa desenvolvida é apresentada na Fig. 3.8.

Figura 3.7 – Diagrama de blocos da placa de comunicação para dispositivos RS-232.



Fonte: Elaborado pelo autor (2018).

Figura 3.8 – Comparação entre projeto 3D (a) e placa montada (b) da placa de comunicação para dispositivos RS-232.



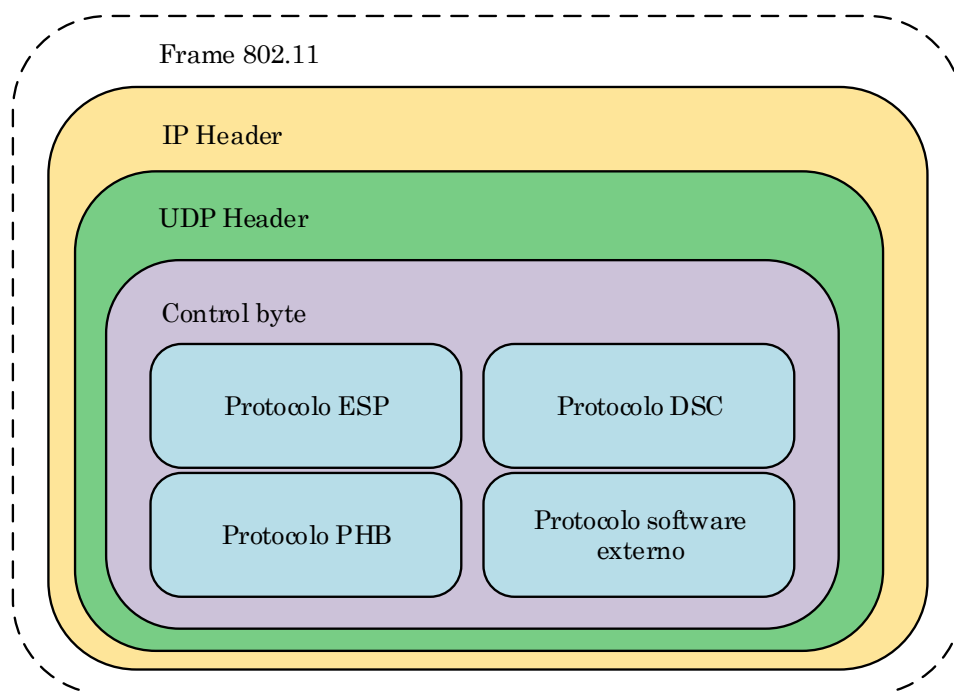
Fonte: Elaborado pelo autor (2018).

4 PROTOCOLOS DE COMUNICAÇÃO

Para a comunicação ocorrer com êxito entre dois ou mais dispositivos em uma rede, os mesmos devem seguir modelos, normas, padrões e protocolos, proporcionando uma conexão íntegra, com qualidade, confiabilidade e segurança. Diversos protocolos podem ser implementados nos dispositivos selecionados para integrar a rede proposta, como o protocolo Modbus. Neste capítulo são apresentados os protocolos desenvolvidos e utilizados na comunicação do sistema proposto.

Na Fig. 4.1 são mostradas as camadas que formam a estrutura de comunicação sem fio, que é utilizada durante a comunicação dos dispositivos propostos para uso na microrrede deste trabalho. Baseado no modelo OSI¹, que separa a comunicação em camadas, divide-se o sistema utilizado de modo similar. Na camada externa há o padrão de comunicação da rede Wifi. Em seguida, a camada de comunicação IP e UDP. Na camada interna encontra-se o *control byte*, finalizando com a camada do protocolo.

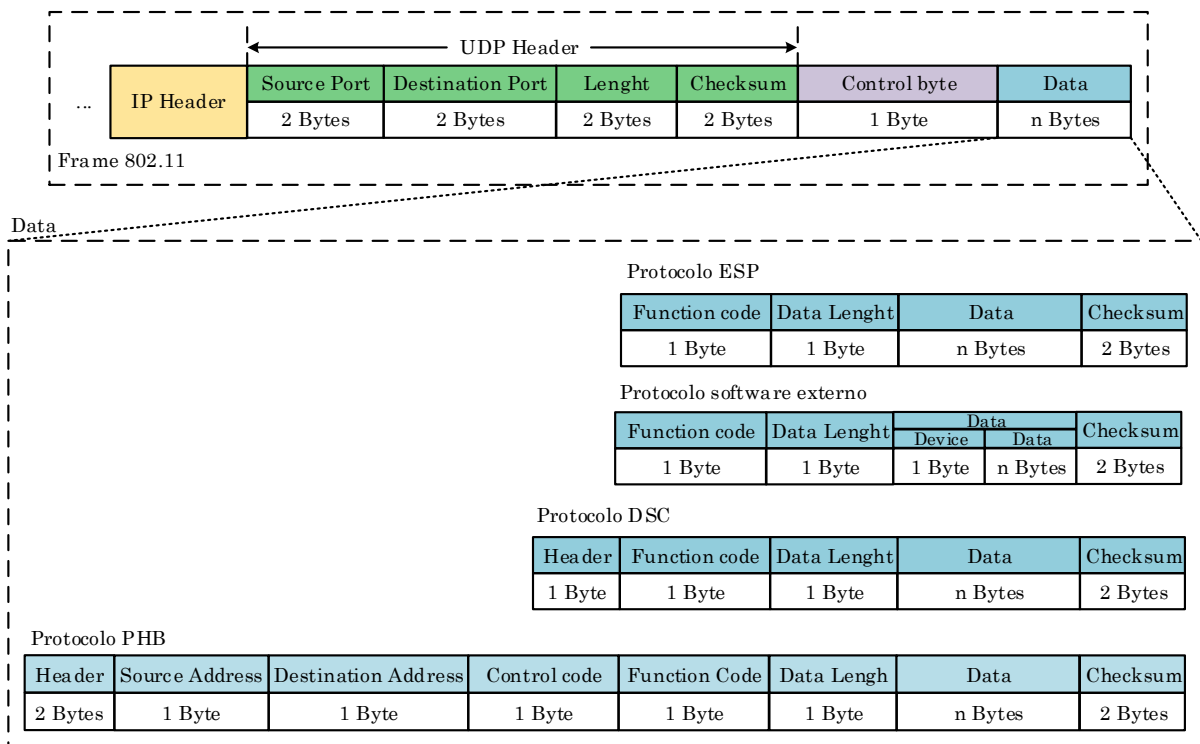
Figura 4.1 – Visão geral das camadas de protocolos utilizadas na comunicação entre dispositivos.



Fonte: Próprio autor.

¹O modelo OSI (*Open System Interconnection*), é um modelo de comunicação que foi introduzido pela primeira vez no final de 1970 que tem como principal característica a divisão da rede por camadas (FOROUZAN; MOSHARRAF, 2013).

Figura 4.2 – Visão detalhada das camadas de protocolos utilizadas na comunicação entre dispositivos.



Fonte: Elaborado pelo autor (2018).

Na Fig. 4.2 são mostrados os bytes que compõem o protocolo de comunicação proposto. Este capítulo visa abordar a camada “*control byte*” e “*Data*”, que é utilizada para alocar diferentes protocolos, dependendo do dispositivo que está sendo utilizado:

- Protocolo ESP;
- Protocolo software externo;
- Protocolo DSC;
- Protocolo PHB.

4.1 CAMADA DE COMUNICAÇÃO CONTROL BYTE

As mensagens podem ter várias origens e diversos destinatários durante a comunicação dos componentes que integram a microrrede. Uma mensagem que é enviada do CCS para a placa de comunicação transita de forma direta, porém, a comunicação entre um DSC e o CCS, por exemplo, passa pela placa de comunicação para então chegar ao destino. Para esta identificação, foi criada uma camada de comunicação chamada de *control byte*, composta de 1 Byte. A criação de uma camada baseia-se no modelo OSI que define a comunicação por camadas conforme a função a ela atribuída.

Foram criados dois modos de identificação das mensagens: o modo *bridge* e modo *configuração do microcontrolador*. No modo *bridge* o microcontrolador apenas encaminha os dados recebidos pelas portas UDP e serial, operando de forma similar a uma ponte, dessa forma nenhuma análise da mensagem é feita pelo microcontrolador. No modo de configuração o microcontrolador processa os dados contidos na mensagem e verifica se alguma ação deve ser tomada. Porém, além do uso do microcontrolador sem fio, os mesmos protocolos podem ter origem de um software externo e não se enquadrariam em nenhuma destas duas classificações. Desta forma, o *control byte* recebe uma terceira definição de *conexão com software externo*. A Tabela 4.1 mostra todos os valores configurados para o *control byte*.

Tabela 4.1 – Valores configurados para o *control byte* e respectivos modos de operação.

Control byte	Modos de operação
00	<i>Bridge</i>
01	Configuração do microcontrolador
02	Conexão com software externo

Fonte: Elaborado pelo autor (2018).

4.2 PROTOCOLO ESP

Para receber e enviar as configurações gerais de ajuste e funções especiais do ESP8266, foi criado um protocolo de comunicação utilizado entre o CCS e o microcontrolador sem fio que é chamado de “Protocolo ESP”. Este protocolo é baseado no protocolo Modbus e tem sua estrutura de bytes mostrada na Tabela 4.2.

Tabela 4.2 – Estrutura dos Bytes do protocolo de comunicação.

Function code	Data length	Data	Checksum
1 byte	1 byte	<i>n</i> bytes	2 bytes

Fonte: Elaborado pelo autor (2018).

O protocolo ESP contém os seguintes campos:

- **Function code:** composto de 1 Byte, indica qual o tipo de função que está sendo solicitada ou executada pelo microcontrolador.
- **Data Length:** composto de 1 Byte, indica o tamanho total dos dados da mensagem;
- **Data:** composto de *n* Bytes, que varia conforme o tipo de mensagem;

- **Checksum:** composto de 2 Bytes, soma o valor de todos os bytes da mensagem (Function code + Data Length + Data).

A Tabela 4.3 apresenta as mensagens implementadas conforme o valor do campo *function code*. O campo *Data* possui as informações de IP, porta e MAC dos dispositivos em que a comunicação ocorre.

Tabela 4.3 – Tabela com valores de *function code* e respectivas funções no protocolo ESP.

Function code	Descrição	Data
00	Conectar o ESP à rede	$IP^1 + Port\ in^1$
01	Configuração do ESP na rede	$IP^2 + Port^2 + MAC\ Address^2$
02	Confirmação de conexão	$IP^1 + Port\ in^1$
03	Reiniciar o ESP	0

Informação do dispositivo: ¹CCS ²ESP

Fonte: Elaborado pelo autor (2018).

Para exemplificar o uso do protocolo dois cenários são tomados. No primeiro caso, ocorre o registro da placa de comunicação ao software de comunicação, processo que deve ser feito no início de toda a comunicação com algum dispositivo que utilizará a placa de comunicação. No segundo caso é feito o pedido de reinicialização do microcontrolador via protocolo e são apresentados nas seções 4.2.1 e 4.2.2 respectivamente.

4.2.1 Registro da placa de comunicação

Antes de iniciar a comunicação de um dispositivo é necessário efetuar o cadastro do respectivo ESP8266 no CCS como um cliente. Cada dispositivo tem uma respectiva placa de comunicação com um ESP, assim é concedido a cada cliente um número de identificação. Este processo permite verificar quantos e quais dispositivos estão conectados ao CCS. O cadastro ocorre com o uso de três mensagens, os *function codes* “00”, “01” e “02” da Tabela 4.3.

A Fig. 4.3 apresenta a troca de mensagens que ocorre ao registrar a placa de comunicação no CCS, com as características dos dispositivos apresentadas na Tabela 4.4. As mensagens enviadas neste exemplo são descritas abaixo e detalhadas na Tabela 4.5:

- **Function code: 00 – Conectar o ESP à rede** – Para que os dispositivos sejam detectados na rede é necessário que o usuário clique em “Busca por dispositivos” dentro da interface do CCS. Quando isso ocorre, o software envia a mensagem de

Tabela 4.4 – Parâmetros contidos no campo *data* no exemplo de comunicação entre o CCS e o ESP8266 para o registro da placa de comunicação.

Parâmetros	Dispositivos	
	CCS	ESP
Endereço IP	192.168.137.01	192.168.137.212
Porta de recebimento	9001	9001
Porta de envio	-	9001
Endereço MAC	-	60:01:94:0E:7E:20

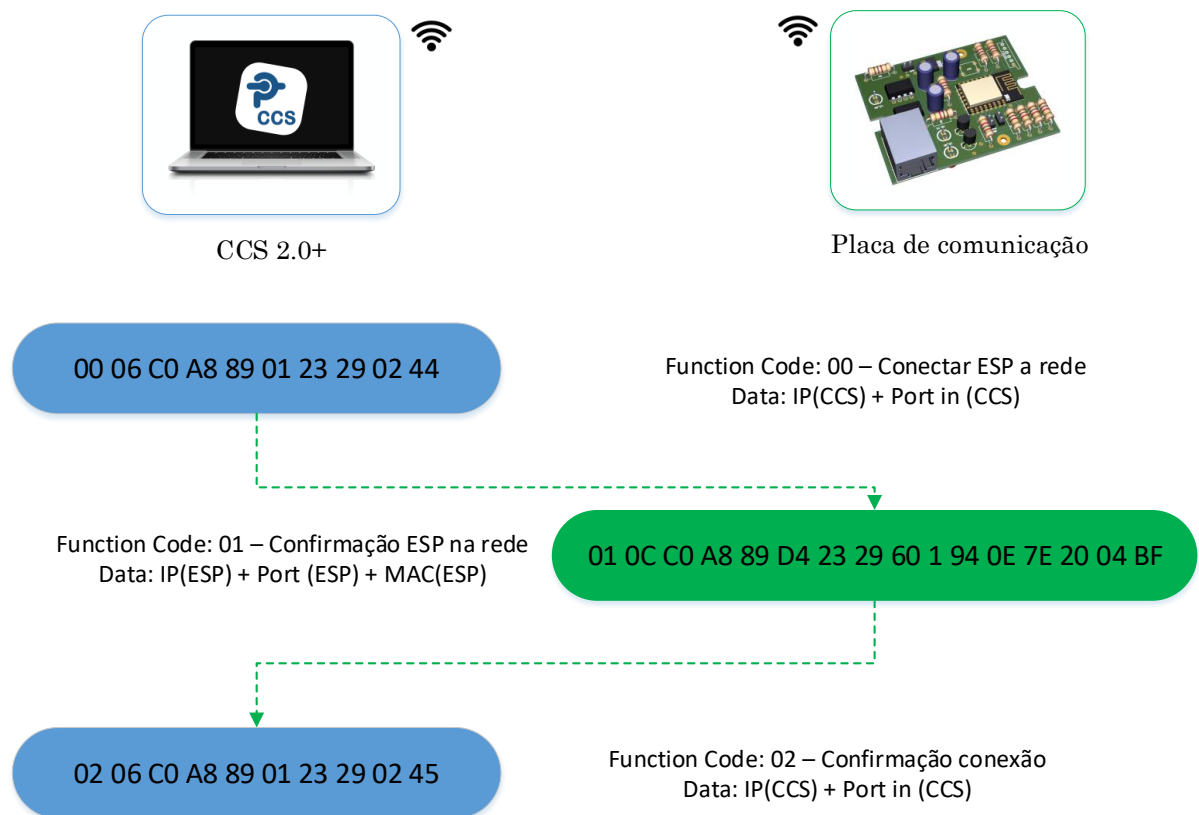
Fonte: Elaborado pelo autor (2018).

“Conectar ESP à rede” via broadcast pela rede, para que todos os equipamentos se identifiquem respondendo a mensagem. Somente os dispositivos programados para interpretar corretamente esta mensagem irão responder. A mensagem contém no campo *data* o endereço IP e a porta de recebimento de mensagens do CCS.

- **Function code: 01 – Configuração do ESP na rede** – Recebida a mensagem anterior pelo ESP o mesmo responde ao CCS com a mensagem “Confirmação ESP na rede”. Como os dados de IP e porta do CCS foram recebidos pela mensagem anterior, esta mensagem é enviada diretamente para o CCS, e não em broadcast como anteriormente. Assim, o CCS terá cadastrado a placa de comunicação para conexão. A mensagem contém no campo *data* o endereço IP, porta de recebimento de mensagens e endereço MAC do ESP.
- **Function code: 02 – Confirmação de conexão** – Para finalizar a etapa de registro o usuário, é selecionado o dispositivo em que deseja realizar a conexão e confirmada a informação enviando a mensagem “Confirmação de conexão”. Desta forma o CCS e ESP estão conectados e prontos para realizar o envio e recebimento das mensagens do dispositivo, finalizando assim a série de mensagens necessárias para o registro da placa de comunicação. A mensagem contém no campo *data* o endereço IP e a porta de recebimento de mensagens do CCS, da mesma maneira que na primeira mensagem.

A integridade do sinal é verificada utilizando o valor recebido no campo *checksum*. Para o cálculo deste somatório são utilizados os valores dos campos: Function code, Data length e Data. O campo *control byte* é considerado como um byte de uma camada superior, portanto, não é considerado no cálculo do checksum e estão apresentados nas tabelas e exemplos apenas para indicação de que em todos estes casos o *control byte* sinaliza configuração.

Figura 4.3 – Exemplo de mensagens enviadas entre o software de monitoramento e placa de comunicação com ESP8266.



Fonte: Elaborado pelo autor (2018).

Tabela 4.5 – Análise das mensagens enviadas no processo de registro da placa de comunicação para o protocolo ESP.

Function code: 00 – Conectar ESP a rede

Control byte	Function code	Data length	Data		Checksum
01	00	06	C0 A8 89 01	23 29	02 44
	a		b	c	d
a	00 – Identificador da função de conectar ESP a rede.				
b	C0 A8 89 01 – Endereço IP 192.168.137.1 do CCS.				
c	23 29 – Porta 9001 em que o CCS recebe as mensagens.				
d	02 44 – Somatório dos bytes Function code, Data length e Data				

Function code: 01 – Confirmação ESP na rede

Control byte	Function code	Data length	Data				Checksum
01	01	0C	C0 A8 89 D4	23 29	60 01 94 0E 7E 20		04 BF
	a		b	c	d		e
a	01 – Identificador da função de solicitação de registro do ESP na rede.						
b	C0 A8 89 D4 – Endereço IP 192.168.137.212 do ESP.						
c	23 29 – Porta 9001 que o ESP recebe as mensagens.						
d	60 01 94 0E 7E 20 – Endereço MAC 60:01:94:0E:7E:20 do ESP.						
e	02 44 – Somatório dos bytes Function code, Data length e Data						

Function code: 02 – Confirmação conexão

Control byte	Function code	Data length	Data		Checksum
01	02	06	C0 A8 89 01	23 29	02 45
	a		b	c	d
a	02 – Identificador da função da confirmação de conexão do ESP na rede.				
b	C0 A8 89 01 – Endereço IP 192.168.137.1 do CCS.				
c	23 29 – Porta 9001 que o CCS recebe as mensagens.				
d	02 45 – Somatório dos bytes Function code, Data length e Data)				

Fonte: Elaborado pelo autor (2018).

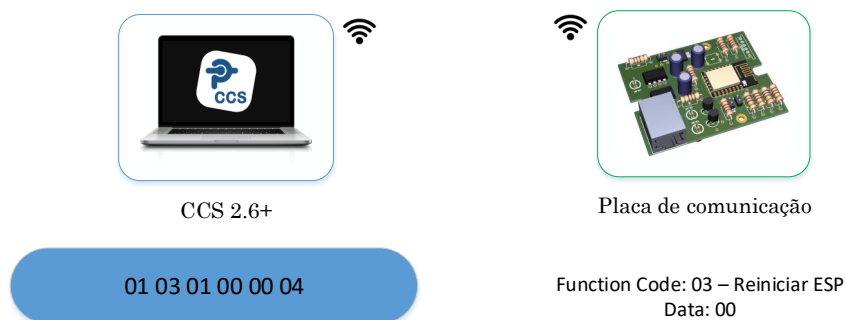
4.2.2 Reiniciar ESP

Caso o usuário precisar reiniciar o ESP existem duas formas de proceder. A primeira e mais comum, é manualmente através da própria porta *reset* do microcontrolador, outra forma é via firmware. Para o último caso uma mensagem foi criada.

A Fig. 4.4 apresenta a mensagem enviada no processo de reset. Como a mensagem solicita a reinicialização do dispositivo nenhuma mensagem de confirmação é recebida pelo software. A Tabela 4.6 apresenta os bytes da mensagem que utilizam o *function code* “03”:

- **Function code: 03 – Reiniciar o ESP** – A mensagem é enviada pelo CCS ao clicar no botão “Reset ESP”. Ao receber a mensagem o ESP reinicia via firmware. Como microcontrolador já estava configurado na rede Wi-Fi, ele conectará automaticamente a última rede conectada após reinicialização.

Figura 4.4 – Exemplo de mensagens enviadas pelo CCS para placa de comunicação com ESP8266 solicitando reinicialização.



Fonte: Elaborado pelo autor (2018).

Tabela 4.6 – Mensagem enviada pelo CCS solicitando a reinicialização do ESP.

Function code: 03 – Reiniciar ESP

Control byte	Function code	Data length	Data	Checksum
01	03	01	00	00 04
	a		b	
a	03 – Identificador da função de reiniciar o ESP.			
b	00 – Sem conteúdo de mensagem.			

Fonte: Elaborado pelo autor (2018).

4.3 PROTOCOLO SOFTWARE EXTERNO

Uma importante funcionalidade do CCS é a possibilidade de conecta-lo a um software externo para receber as informações que estão sendo monitoradas. Isso permite que um software, através de uma conexão UDP, efetue ajustes de parâmetros e controle das variáveis monitoradas pelo CCS. Para que esta comunicação seja realizada é necessário que ambos os softwares utilizem o mesmo protocolo. Para esta finalidade, foi criado o protocolo que é chamado de “protocolo software externo”.

Como diferentes dispositivos podem estar conectados ao CCS e os dados desses diversos dispositivos devem ser enviados ou recebidos pelo software externo é necessário identificar de qual dispositivo a mensagem pertence. Para diferenciar as mensagens dos dispositivo monitorado, foi utilizado um byte dentro do campo *data* deste protocolo. A Tabela 4.7 apresenta a estrutura básica do protocolo.

Tabela 4.7 – Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com software externo.

Function code	Data length	Data		Checksum
		Device	Data	
1 Byte	1 Byte	1 Byte	<i>n</i> Byte(s)	2 Bytes

Fonte: Elaborado pelo autor (2018).

O protocolo contém os campos de:

- **Function code:** composto de 1 Byte, indica a função que está sendo solicitada ou executada pelo software externo.
- **Data Length:** composto de 1 Byte, indica o tamanho total dos dados da mensagem;
- **Data:** composto de 1 Byte, que indica o número do dispositivo a ser monitorado, chamado de *device*, mais *n* Bytes variam conforme o tipo de mensagem.
- **Checksum:** composto de 2 Bytes, soma o valor de todos os bytes da mensagem (Function code + Data Length + Data).

O byte *device* recebe o valor do dispositivo (“01” à “10”) deseja-se receber os dados do respectivo dispositivo. Caso o valor do mesmo seja “00” indica que foi requisitado as informações de todos os dispositivos cadastrados. Esta solicitação é respondida com uma mensagem de resposta para cada dispositivo. A Tabela 4.8 apresenta as mensagens existentes no protocolo de comunicação software externo e mostra o conteúdo da campo *data*.

Tabela 4.8 – Tabela de funções para comunicação com software externo.

Function code	Descrição	Data
00	Requisição nome do dispositivo	Device
01	Resposta com o nome do dispositivo	Device + nome do dispositivo
02	Requisição dos nomes das variáveis	Device
03	Resposta com o nomes das variáveis	Device + nomes das variáveis
04	Requisição das unidades das variáveis	Device
05	Resposta com as unidades das variáveis	Device + unidades das variáveis
06	Requisição dos valores das variáveis	Device
07	Resposta com o valor das variáveis	Device + valor das variáveis ¹
08	Requisição de atualização dos valores das variáveis	Device + valor das variáveis ¹
09	Confirmação da atualização dos valores das variáveis	Device + valor das variáveis ¹

¹ Padrão de envio dos valores de variáveis seguindo codificação IEEE-754.

Fonte: Elaborado pelo autor (2018).

As variáveis a serem monitoradas são números reais. Se apenas 1 Byte fosse utilizado para a transmissão destes dados, somente valores naturais poderiam ser transmitidos, além do problema de limitação de valor da variável. Por este motivo, foi definido utilizar o padrão de codificação IEEE-754 (*Single-precision floating-point format*), que estabelece a utilização de 4 bytes para cada variável do tipo single (IEEE, 2008). Com este padrão é possível que a variável tenha o seu valor máximo de aproximadamente $3,402823 \times 10^{38}$.

4.3.1 Aquisição de parâmetros

Três requisições independentes são feitas para a aquisição dos parâmetros. Desta forma uma mensagem não tem relação com a outra mensagem e podem ser enviadas separadamente a qualquer momento da comunicação. Para exemplificar o processo da obtenção desses dados pelo CCS a Fig. 4.5 apresenta a troca de mensagens. Os dados a serem monitorados apresentados são na Tabela 4.9.

Tabela 4.9 – Parâmetros no exemplo de comunicação entre o CCS e o ESP8266.

Parâmetro	Valor
Números de dispositivos conectados	1
Nome do dispositivo	GUI
Nome da variável a ser monitorada	Var1
Unidade da variável	V

Fonte: Elaborado pelo autor (2018).

As mensagens trocadas no exemplo são apresentadas em detalhes na Tabela 4.10 e contém:

- **Function code: 00 – Requisição nome do dispositivo** – Mensagem enviada do software externo para o CCS e solicita ao CCS o nome do dispositivo desejado. No exemplo são solicitadas as informações do dispositivo de número “1”. Ao receber esta solicitação o CCS responderá com a próxima mensagem.
- **Function code: 01 – Resposta com o nome do dispositivo** – Mensagem enviada do CCS para o software externo contendo o nome do dispositivo. No exemplo, o dispositivo de número "01", tem o nome de "GUI" e essa informação é enviada na mensagem de resposta.
- **Function code: 02 – Requisição dos nomes das variáveis** – Esta mensagem, enviada do software externo para o CCS solicita os nomes das variáveis que serão monitoradas referente ao dispositivo indicado.
- **Function code: 03 – Resposta com o nomes das variáveis** – Após o recebimento da mensagem pelo CCS, o mesmo responde com a solicitação, onde envia do nomes das variáveis do dispositivo solicitado. Por padrão do protocolo o nome das variáveis virão separadas por espaçamento simples². No exemplo apenas uma variável é monitorada.
- **Function code: 04 – Requisição das unidades das variáveis** – Mensagem de solicitação do nome das unidades das variáveis enviada pelo software externo para o CCS.
- **Function code: 05 – Resposta com as unidades das variáveis** – Ao CCS receber a requisição das unidades do software externo responde com as informações requisitadas, caso tenha as informações cadastradas. Como ocorre na solicitação dos nomes as unidades das variáveis também serão enviadas separadas por um espaçamento simples.

²Em ASCII representado por “20” hex

Figura 4.5 – Exemplo de mensagens enviadas entre CCS e software externo para aquisição de parâmetros.



Fonte: Elaborado pelo autor (2018).

Tabela 4.10 – Análise das mensagens enviadas no processo de aquisição dos parâmetros no protocolo software externo (Continua).

Function code: 00 – Requisição nome do dispositivo

Function code	Data Length	Data		Checksum
		Device	Data	
00	01	01	–	00 02
a		b	c	d
a	00 – Identificador da função de requisição nome do dispositivo.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	- – Sem conteúdo de dados.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Function code: 01 – Resposta com o nome do dispositivo

Function code	Data Length	Data		Checksum
		Device	Data	
01	04	01	47 55 49 00	00 EA
a		b	c	d
a	01 – Identificador da função de resposta com o nome do dispositivo.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	47 55 49 00 – Representa “GUI” em ASCII.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Function code: 02 – Requisição dos nomes das variáveis

Function code	Data Length	Data		Checksum
		Device	Data	
02	01	01	–	00 04
a		b	c	d
a	02 – Identificador da função de requisição dos nomes das variáveis.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	- – Sem conteúdo de dados.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Tabela 4.10 – Análise das mensagens enviadas no processo de aquisição dos parâmetros no protocolo software externo (Conclusão).

Function code: 03 – Resposta com o nomes das variáveis

Function code	Data Lenght	Data		Checksum
		Device	Data	
03	05	01	56 61 72 31	01 63
a		b	c	d
a	03 – Identificador da função de resposta com o nome das variáveis.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	56 61 72 31 – Representa “Var1”, nome da variável, em ASCII.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Function code: 04 – Requisição das unidades das variáveis

Function code	Data Lenght	Data		Checksum
		Device	Data	
04	01	01	–	00 06
a		b	c	d
a	02 – Identificador da função de requisição das unidades das variáveis.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	- – Sem conteúdo de dados.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Function code: 05 – Resposta com as unidades das variáveis

Function code	Data Lenght	Data		Checksum
		Device	Data	
05	02	01	56	00 5E
a		b	c	d
a	05 – Identificador da função de resposta com as unidades das variáveis.			
b	01 – Requisição dos dados do dispositivo de número 1.			
c	56 – Representa “V”, unidade da variável monitorada, em ASCII.			
d	00 02 – Somatório dos bytes Function code, Data length e Data			

Fonte: Elaborado pelo autor (2018).

4.3.2 Requisição dos valores das variáveis

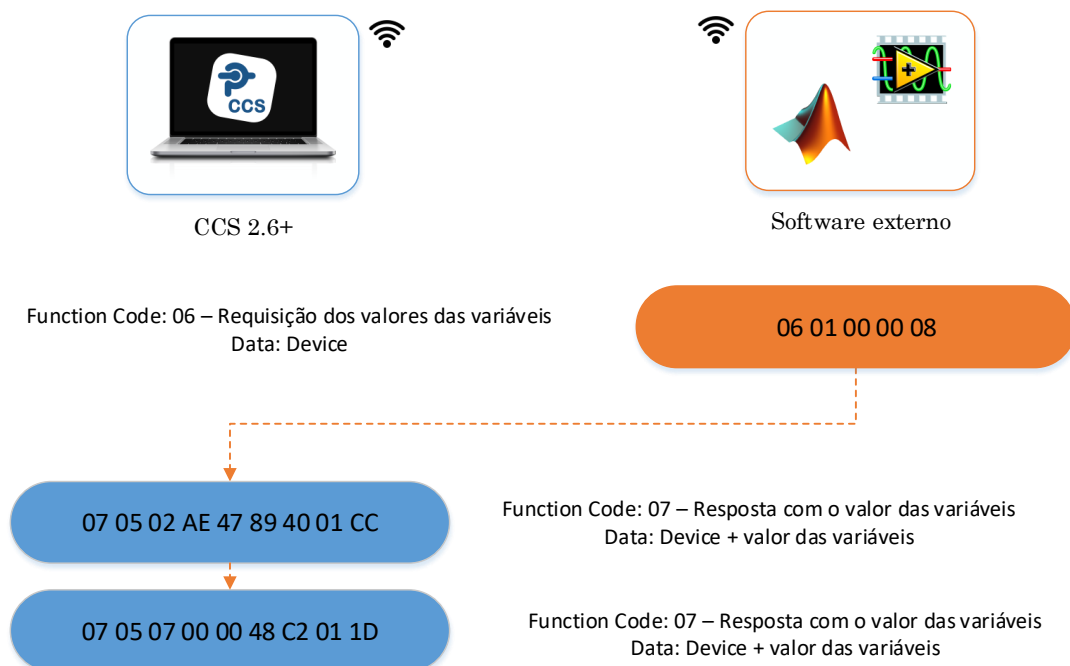
A sequência de mensagens trocadas para o software externo receber os valores das variáveis monitoradas pelo CCS baseia-se em duas mensagens: a requisição do valor seguida da resposta. Porém, esta resposta pode ser composta de uma série de mensagens se as informações mais de um dispositivo estiverem sendo requisitadas. Para indicar a solicitação das informações de todos os dispositivos conectados ao software é utilizado o valor “00” no campo *device*. Para exemplificar esta solicitação foi criado o exemplo apresentado na Fig. 4.6 que contém as características apresentadas na Tabela 4.11.

Tabela 4.11 – Parâmetros contidos no campo *data* no exemplo de de comunicação entre o CCS e o software externo para a requisição dos valores das variáveis.

Parâmetro	Valor
Número dos dispositivos conectado	2 e 7
Valor da variável monitorada do dispositivo 2:	4,29
Valor da variável monitorada do dispositivo 7:	-50,0

Fonte: Elaborado pelo autor (2018).

Figura 4.6 – Exemplo de mensagens enviadas entre CCS e software externo solicitando os valores das variáveis monitoradas em todos os dispositivos conectados ao CCS.



Fonte: Elaborado pelo autor (2018).

As mensagens trocadas são detalhas na Tabela 4.12 e descritas a seguir:

- **Function code: 06 – Requisição dos valores das variáveis** – Mensagem enviada do software externo para o CCS requisitando o valor das variáveis monitoradas por todos os dispositivos. Neste exemplo estão cadastrados no total 2 de 10 dispositivos no CCS, desta forma, após o recebimento da requisição do valor de todos os dispositivos duas mensagens independentes serão enviadas pelo CCS. Caso mais dispositivos estiverem conectados a quantidade de mensagens recebidas é igual a de dispositivos cadastrados.
- **Function code: 07 – Resposta com o valor das variáveis, referente ao dispositivo 2.** – Mensagem enviada do CCS para o software externo com o valor de todas as variáveis cadastradas no dispositivo “2”.
- **Function code: 07 – Resposta com o valor das variáveis, referente ao dispositivo 7.** – Mensagem enviada do CCS para o software externo com o valor de todas as variáveis cadastradas no dispositivo “7”.

Tabela 4.12 – Análise das mensagens enviadas no processo de aquisição dos valores das variáveis no protocolo software externo.

Function code: 06 – Requisição dos valores das variáveis

Function code	Data Length	Data		Checksum
		Device	Data	
06	01	00	–	00 07
a		b	c	d
a	06 – Identificador da função de requisição dos valores das variáveis.			
b	00 – Requisição dos dados de todos dispositivos cadastrados no CCS.			
c	– – Sem conteúdo de dados.			
d	00 07 – Somatório dos bytes Function code, Data length e Data			

Function code: 07 – Resposta com o valor das variáveis, referente ao dispositivo 2.

Function code	Data Length	Data		Checksum
		Device	Data	
07	05	02	AE 47 89 40	01 CC
a		b	c	d
a	07 – Identificador da função de resposta com o valor das variáveis.			
b	02 – Resposta com os dados do dispositivo de número 2.			
c	AE 47 89 40 – Representa valor de 4,29 em conversão do IEEE-754.			
d	01 CC – Somatório dos bytes Function code, Data length e Data			

Function code: 07 – Resposta com o valor das variáveis, referente ao dispositivo 7.

Function code	Data Length	Data		Checksum
		Device	Data	
07	05	07	00 00 48 C2	01 1D
a		b	c	d
a	07 – Identificador da função de resposta com o valor das variáveis.			
b	07 – Resposta com os dados do dispositivo de número 7.			
c	AE 47 89 40 – Representa valor de 50,0 em conversão do IEEE-754 .			
d	01 CC – Somatório dos bytes Function code, Data length e Data.			

Fonte: Elaborado pelo autor (2018).

4.3.3 Requisição de atualização dos valores das variáveis

Através do software externo o usuário pode solicitar a mudança do valor das variáveis de um dispositivo. Ao solicitar uma mensagem de confirmação será recebida. Caso o valor enviado seja realmente atualizado o mesmo valor será encontrado na mensagem de confirmação. Se por alguma lógica do dispositivo, como proteção ou ação de controle, a variável não foi modificada, o valor atual da mesma será enviado na mensagem de confirmação. O exemplo deste pedido de atualização é mostrado na Fig. 4.7 e utiliza as características apresentadas na Tabela 4.13.

Tabela 4.13 – Parâmetros contidos no campo *data* no exemplo de comunicação entre o CCS e o software externo para a requisição de atualização dos valores das variáveis.

Parâmetro	Valor
Número do dispositivo para atualizar dados:	4
Valor da variável para ser atualizada:	4,29
Valor da variável após atualização:	4,30

Fonte: Elaborado pelo autor (2018).

Figura 4.7 – Exemplo de mensagens enviadas entre software criado (CCS) e software externo solicitando atualização dos valores das variáveis.



Fonte: Elaborado pelo autor (2018).

As mensagens trocadas no exemplo são apresentadas em detalhes na Tabela 4.14 e descritas a seguir:

- **Function code: 08 – Requisição de atualização dos valores das variáveis**
 - Mensagem enviada do software externo para o CCS solicitando a atualização dos valores das variáveis. A mensagem inclui o valor que deseja que as variáveis recebam. Para o caso de mais de uma variável a ser monitorada, os respectivos valores são enviados dentro do campo *data*, um em seguida do outro, onde cada variável ocupa o espaço de 4 bytes devido a seguir o padrão de codificação IEEE-754. Para o exemplo apresentado apenas um valor é enviado devido ao dispositivo apenas ter este cadastrado.
- **Function code: 09 – Confirmação da atualização dos valores das variáveis**
 - Mensagem enviada do CCS para o software externo. Ao receber a mensagem anterior o CCS processa a informação e a envia para o dispositivo conectado. O dispositivo efetua a mudança e responde ao CCS que por sua vez confirma a atualização para o software externo.

No exemplo apresentado na Fig. 4.7 é solicitado que o valor da variável “1” seja alterado para “4,29”, porém considerando que dentro do software do dispositivo há um processo que indique que este valor não possa ser recebido e devido a sua lógica o valor deve ser de “4,30”. Desta forma, na mensagem de confirmação será o valor de “4,30” que será encontrado. Com essa dinâmica de confirmação o usuário pode verificar se o valor da variável realmente foi alterado e evitar que erros ocorram durante as requisições.

Tabela 4.14 – Análise das mensagens enviadas no processo de atualização dos valores das variáveis no protocolo software externo.

Function code: 08 – Requisição de atualização dos valores das variáveis.

Function code	Data Length	Data		Checksum
		Device	Data	
08	05	04	AE 47 89 40	01 CE
a		b	c	d
a	08 – Função de requisição de atualização dos valores das variáveis.			
b	04 – Requisição dos dados do dispositivo de número 4.			
c	AE 47 89 40 – Valor de 4,29 em conversão do IEEE-754.			
d	00 07 – Somatório dos bytes Function code, Data length e Data			

Function code: 09 – Confirmação da atualização dos valores das variáveis.

Function code	Data Length	Data		Checksum
		Device	Data	
09	05	04	9A 99 89 40	02 0E
a		b	c	d
a	09 – Função de confirmação da atualização dos valores das variáveis.			
b	04 – Resposta com os dados do dispositivo de número 4.			
c	9A 99 89 40 – Representa valor de 4,30 em conversão do IEEE-754.			
d	02 0E – Somatório dos bytes Function code, Data length e Data			

Fonte: Elaborado pelo autor (2018).

4.4 PROTOCOLO DSC

Devido alto uso de DSCs para aplicações, desenvolvimento de pesquisas e também na indústria, foi criado um protocolo para a comunicação de tais dispositivos com o CCS. O protocolo criado para esta aplicação é similar ao protocolo de comunicação com softwares externos. A principal modificação é o uso de um byte *header* no início da mensagem, utilizado para indicar o início da mensagem caso seja utilizado através de comunicação serial. A estrutura básica do protocolo é vista na Tabela 4.15. Neste caso, o *control byte* recebe o valor “02”, ou seja, o ESP apenas transmite os dados que trafegam por ele.

O protocolo contém os campos de:

- **Header:** composto de 1 byte, indica início da mensagem, no caso de comunicação via serial facilita a identificação;

- **Function code:** composto de 1 byte, indica qual o tipo de função que está sendo solicitada/executada pelo DSC.
- **Data Length:** composto de 1 byte, indica o tamanho total dos dados da mensagem;
- **Data:** composto de n Bytes, que variam conforme o tipo de mensagem;
- **Checksum:** composto de 2 bytes, soma o valor de todos os bytes da mensagem (Header + Function code + Data Length + Data).

Tabela 4.15 – Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com DSC.

Header	Function code	Data length	Data	Checksum
1 byte	1 byte	1 byte	n bytes	2 bytes

Fonte: Elaborado pelo autor (2018).

A Tabela 4.16 mostra as funções definidas para o “*function code*” conforme descrição e quais as informações que o campo “Data” recebe.

Tabela 4.16 – Tabela de funções para comunicação com DSC.

Function code	Descrição	Data
00	Requisição do nome do dispositivo	0
01	Resposta com o nome do dispositivo	Nome do dispositivo
02	Requisição dos nomes das variáveis	0
03	Resposta com o nomes das variáveis	Nomes das variáveis
04	Requisição das unidades das variáveis	0
05	Resposta com as unidades das variáveis	Unidades das variáveis
06	Requisição dos valores das variáveis	0
07	Resposta com o valor das variáveis	Valor das variáveis ¹
08	Requisição de atualização dos valores das variáveis	Valor das variáveis ¹
09	Confirmação da atualização dos valores das variáveis	Valor das variáveis ¹

¹ Padrão de envio dos valores de variáveis seguindo codificação IEEE-754.

Fonte: Elaborado pelo autor (2018).

4.4.1 Aquisição de parâmetros

Para a requisição dos parâmetros três requisições independentes podem ser feitas. A mensagem de requisição do nome do dispositivo, *function code* "00", a mensagem da requisição do nome das variáveis, *function code* "02", e a mensagem de requisição das

unidades *function code* "04". A ordem das mensagens não interfere no funcionamento do software. O exemplo na Fig. 4.8 mostra a aquisição dos parâmetros, onde o dispositivo conectado ao CCS teria as características apresentadas na Tabela 4.17.

Tabela 4.17 – Parâmetros contidos no campo *data* no exemplo de de comunicação entre o CCS e o DSC para pedidos de parâmetros.

Parâmetro	Valor
Nome do dispositivo:	"GUI"
Nome das variáveis a serem monitoradas:	"Var1" e "Var2"
Unidade das variáveis:	"V" e "A"

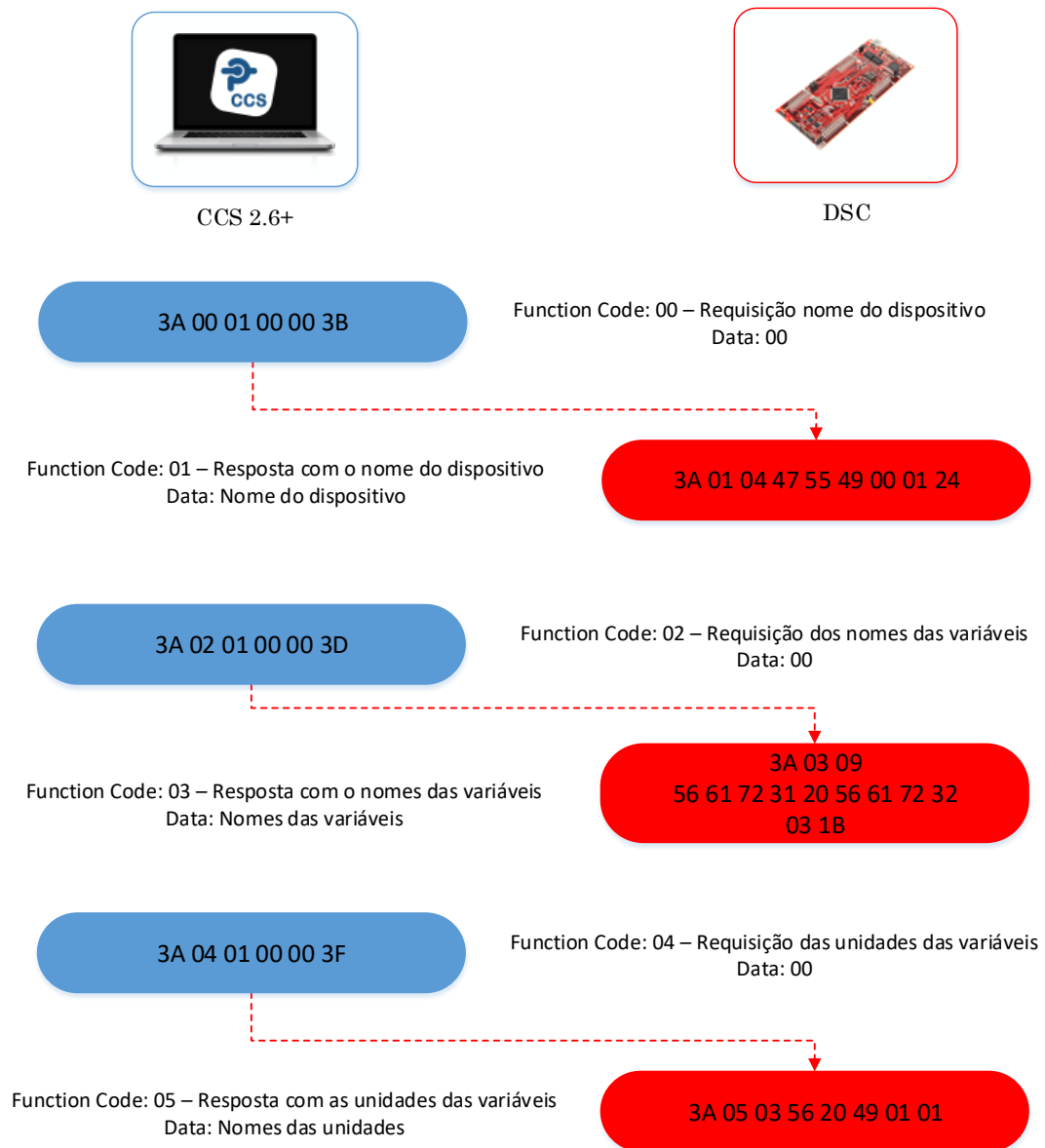
Fonte: Elaborado pelo autor (2018).

Os detalhes de cada mensagem enviada no exemplo, são mostradas na Tabela 4.18 e são compostos por:

- **Function code: 00 – Requisição do nome do dispositivo** – Mensagem enviada do CCS para o DSC solicitando o nome do dispositivo. Esta informação é utilizada para a identificação, pelo usuário, do dispositivo no software.
- **Function code: 01 – Resposta com o nome do dispositivo** – Após receber a solicitação do nome do dispositivo, o DSC responderá com a informação. No exemplo o dispositivo tem o nome de "GUI" e essa informação é enviada via protocolo no pacote *data*.
- **Function code: 02 – Requisição dos nomes das variáveis** – Mensagem que solicita o nome das variáveis cadastradas no DSC enviada pelo CCS.
- **Function code: 03 – Resposta com o nomes das variáveis** – Mensagem de resposta que contém os nomes das variáveis solicitadas. Caso houver mais de uma variável elas são separadas pelo caractere de espaçamento³. No exemplo utilizado duas variáveis estão cadastrada no DSC com nomes "Var1" e "Var2".
- **Function code: 04 – Requisição das unidades das variáveis** – Enviada do CCS para o DSC, esta mensagem solicita a unidade das variáveis monitoradas.
- **Function code: 05 – Resposta com as unidades das variáveis** – Mensagem que contém o valor das unidades cadastradas. No exemplo utilizado duas variáveis estão sendo monitoradas com as respectivas unidades "V" e "A".

³Em ASCII representado por 20 hex

Figura 4.8 – Exemplo de mensagens enviadas entre CCS e DSC solicitando parâmetros.



Fonte: Elaborado pelo autor (2018).

Tabela 4.18 – Análise das mensagens enviadas no processo de solicitação de parâmetros
(Continua).

Function code: 00 – Requisição do nome do dispositivo

Header	Function code	Data length	Data	Checksum
3A	00	01	00	00 3B
	a		b	c
a	00 – Identificador da função de requisição nome do dispositivo.			
b	00 – Valor “00” que indica sem conteúdo de dados.			
c	00 02 – Somatório de Header, Function code, Data length e Data			

Function code: 01 – Resposta com o nome do dispositivo

Header	Function code	Data length	Data	Checksum
3A	01	04	47 55 49 00	01 24
	a		b	c
a	01 – Identificador da função da resposta com o nome do dispositivo.			
b	47 55 49 00 – Representa “GUI”, nome do dispositivo, em ASCII.			
c	01 24 – Somatório de Header, Function code, Data length e Data			

Function code: 02 – Requisição dos nomes das variáveis

Header	Function code	Data length	Data	Checksum
3A	02	01	00	00 3D
	a		b	c
a	02 – Identificador da função da requisição dos nomes das variáveis.			
b	00 – Valor “00” que indica sem conteúdo de dados.			
c	00 3D – Somatório de Header, Function code, Data length e Data			

Tabela 4.18 – Análise das mensagens enviadas no processo de solicitação de parâmetros (Conclusão).

Function code: 03 – Resposta com o nomes das variáveis

Header	Function code	Data length	Data	Checksum
3A	03	0A	56 61 72 31 20 56 61 72 32	03 1B
	a		b	c
a	03 – Identificador da função da resposta com o nomes das variáveis.			
b	56 61 72 31 – Representa “Var1”, nome da variável 1, em ASCII.			
	20 – Espaçamento simples, indicando divisão de variável.			
	56 61 72 32 – Representa “Var2”, nome da variável 2, em ASCII.			
c	03 1B – Somatório de Header, Function code, Data length e Data			

Function code: 04 – Requisição das unidades das variáveis

Header	Function code	Data length	Data	Checksum
3A	04	01	00	00 3F
	a		b	c
a	04 – Identificador da função da requisição das unidades das variáveis.			
b	00 – Valor “00” que indica sem conteúdo de dados.			
c	00 3F – Somatório de Header, Function code, Data length e Data			

Function code: 05 –Resposta com as unidades das variáveis

Header	Function code	Data length	Data	Checksum
3A	05	03	56 20 49	01 01
	a		b	c
a	05 – Identificador da função com as unidades das variáveis.			
	56 – Representa “V”, unidade da variável 1, em ASCII.			
b	20 – Espaçamento simples, indicando divisão de variável.			
	49 – Representa “A”, unidade da variável 2, em ASCII.			
c	01 01 – Somatório de Header, Function code, Data length e Data			

Fonte: Elaborado pelo autor (2018).

4.4.2 Requisição do valor das variáveis

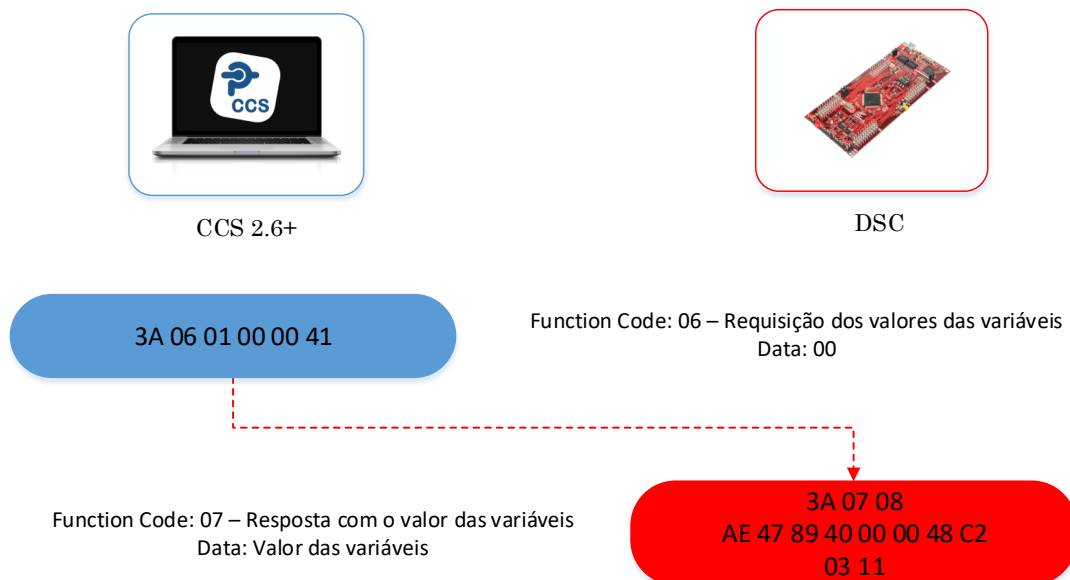
Para o recebimento do valor das variáveis apenas uma mensagem é necessária. Em resposta, uma mensagem com o valor de todas as variáveis cadastradas no dispositivo é recebida. Nesta, cada variável ocupa o espaço de 4 bytes devido ao uso do padrão IEEE-754 Floating Point. Um exemplo desta aplicação é apresentado na Fig. 4.9 onde a troca de mensagens tem as características apresentadas na Tabela 4.19.

Tabela 4.19 – Parâmetros contidos no campo *data* no exemplo de de comunicação entre o CCS e o DSC para requisição do valor das variáveis.

Parâmetro	Valor
Valor da variável 1:	4,29
Valor da variável 2:	-50,0

Fonte: Elaborado pelo autor (2018).

Figura 4.9 – Exemplo de mensagens enviadas entre DSC e CCS para monitoramento dos valores das variáveis.



Fonte: Elaborado pelo autor (2018).

As mensagens trocadas no exemplo contém as mensagens que são apresentadas em detalhes na Tabela 4.20 e são descritas a seguir:

- **Function code: 06 – Requisição dos valores das variáveis** – Mensagem enviada do CCS para o DSC que solicita o valor das variáveis monitoradas.

- **Function code: 07 – Requisição dos valores das variáveis** – Resposta à mensagem recebida e contém o valor das variáveis monitoradas. Dentro do campo *data* cada variável ocupa sempre o tamanho de 4 Bytes, seguindo o padrão de conversão IEEE-754. No exemplo apresentado é recebido o valor de “4,29” para a variável “1” e “-50,0” para a variável “2”.

Tabela 4.20 – Análise das mensagens enviadas no processo requisição do valor das variáveis.

Function code: 06 – Requisição dos valores das variáveis

Header	Function code	Data length	Data	Checksum
3A	06	01	00	00 41
	a		b	c
a	06 – Identificador da função de requisição dos valores das variáveis .			
b	00 – Valor “00” que indica sem conteúdo de dados.			
c	00 41 – Somatório dos bytes Header, Function code, Data length e Data			

Function code: 07 – Resposta com o valor das variáveis

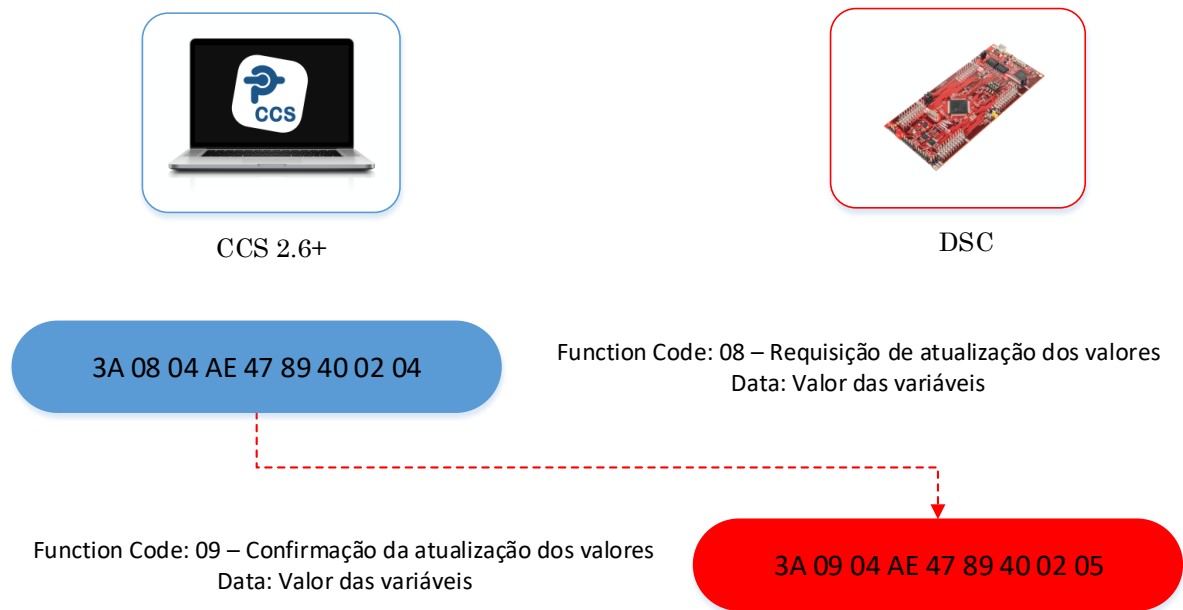
Header	Function code	Data length	Data	Checksum
3A	07	01	AE 47 89 40 00 00 48 C2	00 41
	a		b	c
a	07 – Identificador da função de resposta com o valor das variáveis.			
b	AE 47 89 40 – Representa valor de 4,29 em conversão do IEEE-754.			
	00 00 48 C2 – Representa valor de -50,0 em conversão do IEEE-754.			
c	00 02 – Somatório dos bytes Header, Function code, Data length e Data			

Fonte: Elaborado pelo autor (2018).

4.4.3 Requisição de atualização dos valores das variáveis

Através da requisição de atualização dos valores das variáveis é possível modificar o valor das variáveis do DSC via CCS. Ao receber a mensagem o DSC processa a modificação e envia uma mensagem com o valor da variável após a mudança, desta forma indicando se houve ou não a mudança da variável. Sendo assim, resta ao DSC aceitar, ou não, a troca da variável conforme lógica interna. Para exemplificar o funcionamento dessa dinâmica, no exemplo mostrado na Fig. 4.7 é solicitada a modificação do valor da variável para “4,29”. Após receber o valor o DSC efetua a mudança e envia a mensagem de resposta.

Figura 4.10 – Exemplo de mensagens enviadas entre CCS e DSC solicitando atualização dos valores das variáveis.



Fonte: Elaborado pelo autor (2018).

As mensagens trocadas no exemplo apresentado na Fig. 4.10 contém as mensagens que são apresentadas em detalhes na Tabela 4.21 e são descritas a seguir:

- **Function code: 08 – Requisição de atualização dos valores das variáveis**
– Mensagem enviada do CCS para o DSC solicitando a atualização das variáveis para o valor contido no campo *data*. No exemplo apresentado o valor de “4,29” é enviado.
- **Function code: 09 – Confirmação da atualização dos valores das variáveis**
– Mensagem com o valor das variáveis após o processamento do DSC que confirma, ou não, a alteração da variável. No exemplo a mensagem de resposta contém o mesmo valor que enviado, desta forma a atualização do valor foi realizada com sucesso.

Tabela 4.21 – Análise das mensagens enviadas no processo de requisição de atualização dos valores das variáveis.

Function code: 08 – Requisição de atualização dos valores das variáveis

Header	Function code	Data length	Data	Checksum
3A	08	04	AE 47 89 40	02 04
	a		b	c
a	08 – Identificador da função de requisição de atualização dos valores das variáveis.			
b	AE 47 89 40 – Representa valor de 4,29 em conversão do IEEE-754.			
c	02 04 – Somatório dos bytes Header, Function code, Data length e Data			

Function code: 09 – Confirmação da atualização dos valores das variáveis

Header	Function code	Data length	Data	Checksum
3A	09	04	AE 47 89 40	02 05
	a		b	c
a	09 – Identificador da função de confirmação da atualização dos valores das variáveis.			
b	AE 47 89 40 – Representa valor de 4,29 em conversão do IEEE-754.			
c	02 05 – Somatório dos bytes Header, Function code, Data length e Data			

Fonte: Elaborado pelo autor (2018).

4.5 PROTOCOLO PHB

Para a comunicação de seus dispositivos a empresa PHB utiliza um protocolo próprio, porém muito similar ao Modbus. A Tabela 4.22 mostra a estrutura e tamanho dos campos.

Tabela 4.22 – Estrutura dos Bytes do protocolo de comunicação utilizado para comunicação com inversor PHB.

Header	Source Address	Destination Address	Control Code	Function code	Data length	Data	Checksum
2 Bytes	1 Bytes	1 Bytes	1 Bytes	1 Bytes	1 Bytes	<i>n</i> Bytes	2 Bytes

Fonte: Elaborado pelo autor (2018).

No protocolo desta comunicação é utilizado os parâmetros:

- **Header:** composto de 2 Byte, indica início do pacote e recebe valor de 0xAA e 0x55;
- **Source Address:** composto de 1 Byte, determina o endereço do remetente;

- **Destination Address:** composto de 1 Byte, determina o endereço do destinatário (Endereço inicial: 0x7F);
- **Control code:** composto de 1 byte, está dividido nos tipos: Registro - 0x00, Leitura - 0x01, Escrever - 0x02, Executar - 0x03;
- **Function code:** composto de 1 Byte, contém o código da função a ser utilizada, uma subdivisão do control code;
- **Data length:** composto de 1 Byte, determina o comprimento dos dados (0 se não tem dados);
- **Data:** composto de n Bytes, contém os dados a serem enviados/recebidos;
- **Checksum:** composto de 2 bytes, soma todos os valores para checagem (Header + Source Address + Destination Address + Control Code + Function code + Data length + Data).

Durante toda a comunicação os dispositivos são identificados por um endereço único, desta forma ao enviar uma mensagem essa se destina exatamente para o dispositivo desejado sendo identificado através do campo *destination address*. Os Bytes *control code* e *function code* recebem diversos valores que estão citados nas definições do protocolo fornecidos pelo próprio fabricante.

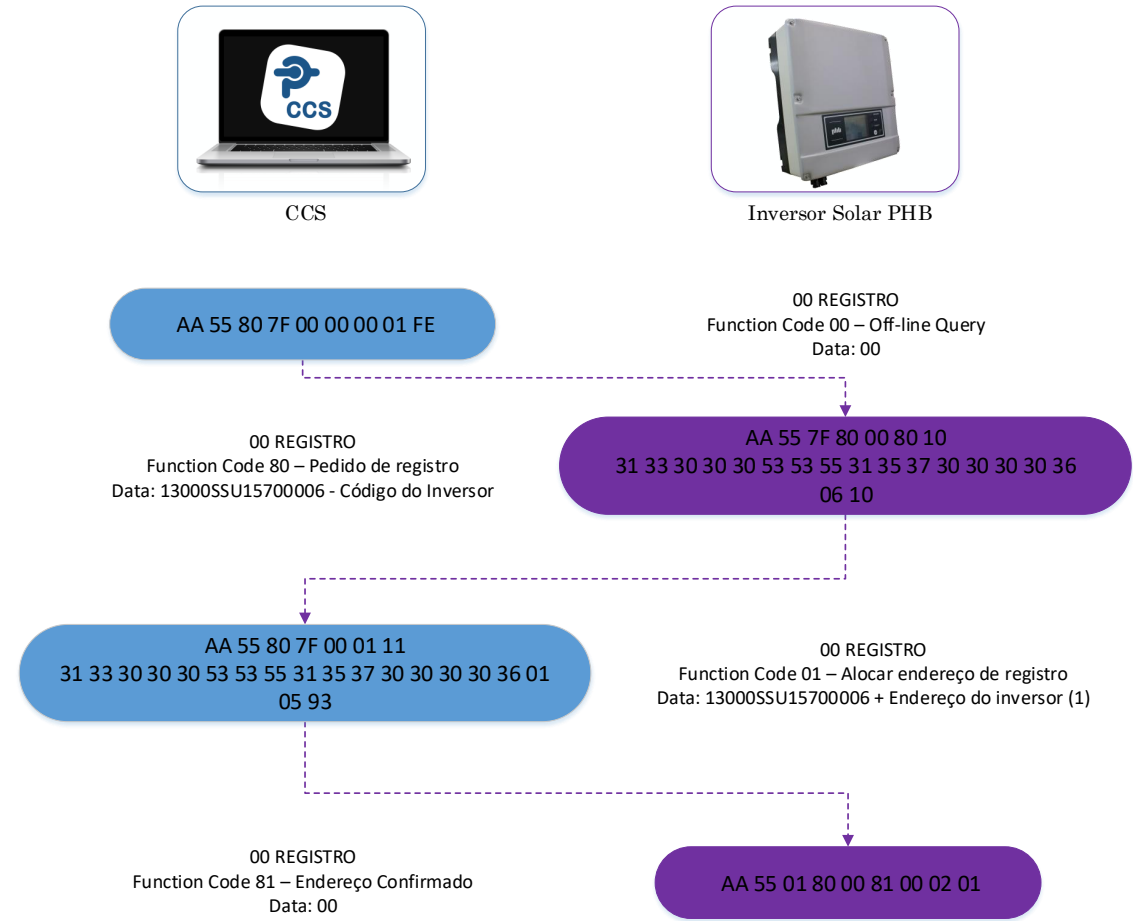
4.5.1 Registro de um inversor na rede

Antes de iniciar a comunicação deve-se registrar o dispositivo. Como regra de comunicação, caso um inversor PHB não receba nenhuma instrução dentro de 10 min o estado do inversor será descadastrado automaticamente, e o processo de registrar o dispositivo deve ocorrer novamente. Para o cadastro são utilizadas as mensagens de *control code* “00”, classificadas como registro. A Fig. 4.11 ilustra o procedimento composto por quatro mensagens. As mensagens de registro do inversor são descritas abaixo e analisadas na Tabela 4.23:

- **Function code: 00 – Off-line query** – Mensagem enviada do CCS para o inversor PHB solicitando o envio dos dados para executar o registro do inversor.
- **Function code: 80 – Pedido de registro** – O inversor solar responde a solicitação com o código do inversor que inclui o modelo e número de série do produto.
- **Function code: 01 – Alocar endereço de registro** – Ao receber os dados do inversor o CCS responde a solicitação que confirma o código recebido e determina qual será o número do dispositivo. No exemplo utilizado é cadastrado como endereço número 1.

- **Function code: 81 – Endereço confirmado** – Finalizando o processo de registro, o inversor confirma com a mensagem.

Figura 4.11 – Comunicação do inversor solar com AP para registro de um inversor.



Fonte: Elaborado pelo autor (2018).

Tabela 4.23 – Mensagens enviadas no processo de registro de um inversor a rede.

Control Code: 00 – Registro
Function code: 00 – Off-line Query

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	80	7F	00	00	00	-	01 FE
			a	b		c	d
a	00 – Identificador do código de controle do tipo registro.						
b	00 – Identificador da função de Off-line Query.						
c	- – Sem conteúdo de dados.						
d	02 04 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Function code: 80 – Pedido de registro

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	7F	80	00	80	10	[...]	06 10
			a	b		c	d
a	00 – Identificador do código de controle do tipo registro.						
b	80 – Identificador da função de pedido de registro.						
c	[...] – Conteúdo dos dados analisado na Tabela 4.24						
d	02 04 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Function code: 01 – Alocar endereço de registro

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	80	7F	00	01	11	[...]	05 93
			a	b		c	d
a	00 – Identificador do código de controle do tipo registro.						
b	80 – Identificador da função de alocar endereço de registro.						
c	[...] – Conteúdo dos dados analisado na Tabela 4.25						
d	02 04 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Function code: 81 – Confirmação de endereço

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	01	80	00	81	00	-	02 01
			a	b		c	d
a	00 – Identificador do código de controle do tipo registro.						
b	80 – Identificador da função de confirmação de endereço.						
c	- – Sem conteúdo de dados.						
d	02 04 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Tabela 4.24 – Parâmetros contidos no campo *data* no exemplo de comunicação entre o CCS e o DSC para mensagem de pedido de registro.

Byte	Data	ASCII	Byte	Data	ASCII	Byte	Data	ASCII	Byte	Data	ASCII
1	31	1	5	30	0	9	31	1	13	30	0
2	33	3	6	53	S	10	35	5	14	30	0
3	30	0	7	53	S	11	37	7	15	30	0
4	30	0	8	55	U	12	30	0	16	36	6

Fonte: Elaborado pelo autor (2018).

Tabela 4.25 – Parâmetros contidos no campo *data* no exemplo de comunicação entre o CCS e o DSC para mensagem para alocar endereço de registro.

Byte	Data	ASCII	Byte	Data	ASCII	Byte	Data	ASCII	Byte	Data	ASCII
1	31	1	6	53	S	11	37	7	16	36	6
2	33	3	7	53	S	12	30	0	17	01	Address
3	30	0	8	55	U	13	30	0			
4	30	0	9	31	1	14	30	0			
5	30	0	10	35	5	15	30	0			

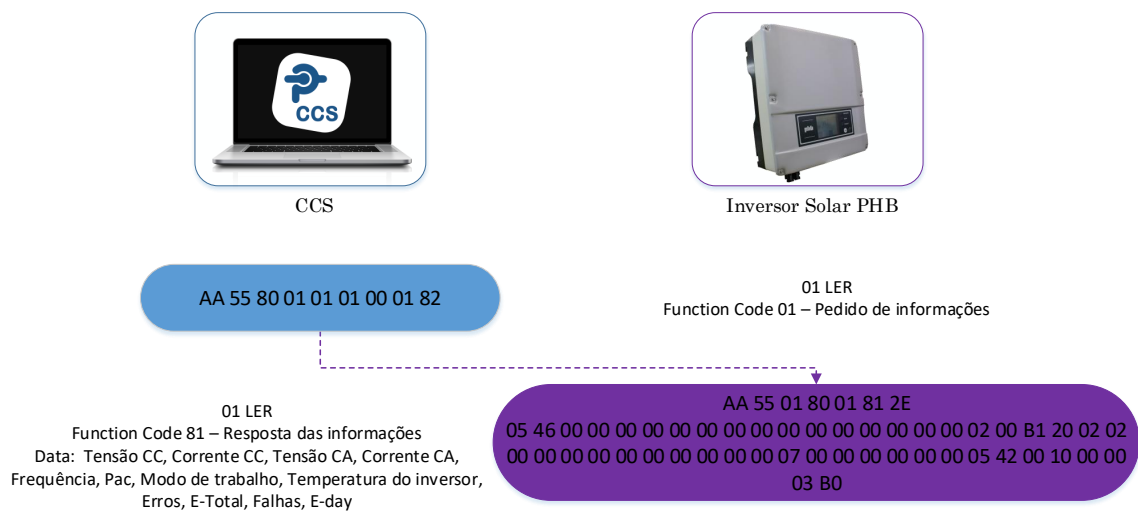
Fonte: Elaborado pelo autor (2018).

4.5.2 Requisição dos valores das variáveis

O envio de apenas uma mensagem é suficiente para monitorar os dados do inversor solar PHB sendo que a resposta enviada pelo inversor contém todos os dados de leitura. O processo é apresentado na Fig. 4.12 com suas mensagens apresentadas na Tabela 4.26 e descritas a seguir:

- **Function code: 01 – Pedido de informações** – Mensagem enviada do CCS para o inversor PHB solicitando o envio dos dados das variáveis.
- **Function code: 81 – Resposta das informações** – Mensagem enviada do inversor PHB para o CCS com todas as variáveis monitoradas. Todas as variáveis virão na mesma mensagem e são detalhadas na Tabela 4.27.

Figura 4.12 – Comunicação do inversor solar com CCS para verificação de informações.



Fonte: Elaborado pelo autor (2018).

Tabela 4.26 – Análise das mensagens enviadas no processo de requisição de dados.

Control Code: 01 – Leitura
Function code: 01 – Pedido de informações

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	80	01	01	01	00	-	01 82
			a	b		c	d
a	01 – Identificador do código de controle do tipo leitura.						
b	01 – Identificador da função de pedido de informações.						
c	- - Sem conteúdo de dados.						
d	01 82 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Function code: 81 – Resposta das informações

Header	Source	Destination	Control	Function	Data length	Data	Checksum
AA 55	01	80	01	81	2E	[...]	03 80
			a	b		c	d
a	01 – Identificador do código de controle do tipo leitura.						
b	81 – Identificador da função de respostas das informações.						
c	[...] – Conteúdo dos dados analisado na Tabela 4.27						
d	03 80 – Somatório dos bytes Header, Source, Dest., Contr., Func., D.Length e Data						

Fonte: Elaborado pelo autor (2018).

Tabela 4.27 – Parâmetros contidos no campo *data* no exemplo de de comunicação entre o CCS e o DSC para requisição do valor das variáveis e mensagem de confirmação de endereço.

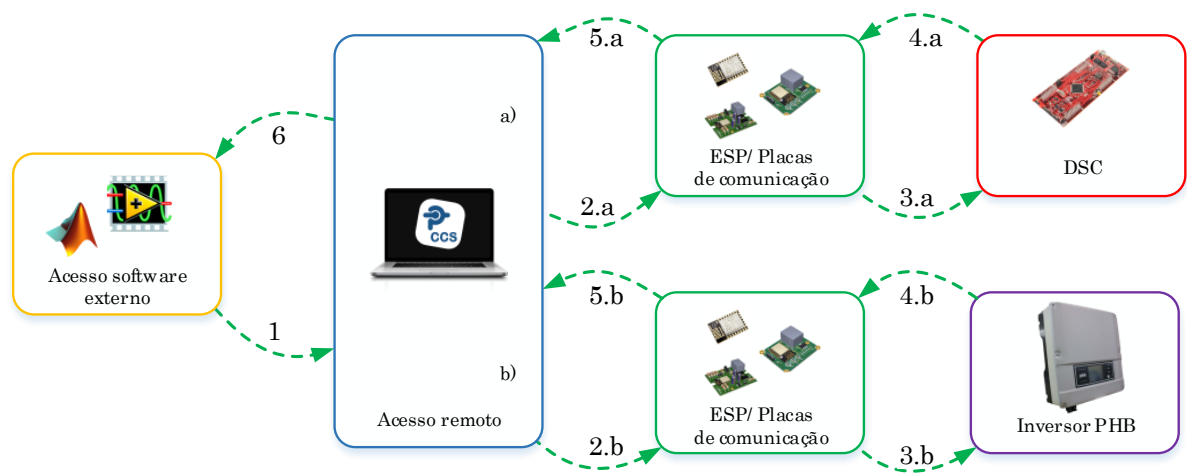
Byte	Index	Data	Descrição	Byte	Index	Data	Descrição
1	0x00	05	Vpv1	25	0x0C	00	e-total 00
2	0x00	46	135,0V	26	0x0C	00	
3	0x01	00	Vpv2	27	0x0D	00	
4	0x01	00	0 V	27	0x0D	00	
5	0x02	00	Ipv1	29	0x0E	00	h-total 00
6	0x02	00	0 A	30	0x0E	00	
7	0x03	00	Ipv2	31	0x0F	00	
8	0x03	00	0 A	32	0x0F	00	
9	0x04	00	Vac1	33	0x10	00	Temp. Fault 0,7°C
10	0x04	00	0 V	34	0x10	07	
11	0x05	00	Iac1	35	0x11	00	Pv1 Fault 00
12	0x05	00	0 A	36	0x11	00	
13	0x06	00	Fac1	37	0x12	00	Pv2 Fault 00
14	0x06	00	0 Hz	38	0x12	00	
15	0x07	00	PAC	39	0x13	00	Line1 Fault 00
16	0x07	00	0 W	40	0x13	00	
17	0x08	00	Work Mode	41	0x16	05	Line2 Fault 13,46Hz
18	0x08	02	02	42	0x16	42	
19	0x09	00	Temp.	43	0x19	00	GFCI Fault 10mA
20	0x0A	81	17,7°C	44	0x19	10	
21	0x0A	20	Erro 20 02 02 00	45	0x1A	00	Eday 0
22	0x0A	02		46	0x1A	00	
23	0x0B	02					
24	0x0B	00					

Fonte: Elaborado pelo autor (2018).

4.6 INTEGRAÇÃO ENTRE DIFERENTES DISPOSITIVOS E PROTOCOLOS

Para analisar a integração dos protocolos apresentados foi utilizado um cenário onde na microrrede estão conectados um DSC e um inversor PHB. Estes dispositivos estão conectados ao software de monitoramento através de uma placa de comunicação. Espera-se ainda que os dados sejam monitorados através de um software externo. A Fig. 4.13 mostra a rede proposta e indica o fluxo de mensagens que ocorre.

Figura 4.13 – Esquema do fluxo de comunicação com protocolos apresentados.



Fonte: Elaborado pelo autor (2018).

Considerando que as mensagens de configurações de conexões dos dispositivos já foram feitas e ambos dispositivos já estão corretamente cadastrados no CCS, será analisado quais os caminhos que as mensagens de requisição e respostas de dados tomam.

Para a requisição dos dados pelo software externo dois caminhos podem ser tomados conforme o tipo de dispositivo selecionado. Na Fig. 4.13 a comunicação com DSC é representados pela letra “a” e para a comunicação com o inversor é utilizada a letra “b”.

A requisição de dados do software externo ocorre pelo caminho “1” e utiliza o protocolo de software externo associado ao *function code* “08”. Neste caso, o *control byte* tem o valor de “02”, que indica um software externo está conectado. Ao ser recebida pelo CCS a mensagem é processada e o mesmo tomará as ações conforme o número do dispositivo monitorado, tomando assim, neste caso o caminho “a” ou “b” mostrado na Fig. 4.13.

No caminho “2.a” a mensagem é enviada para o DSC, porém, antes passa pelo ESP ou placa de comunicação, desta forma a mensagem já está com o protocolo do DSC, com o *control byte* igual a “0”, ou seja, o ESP trabalha como ponte e só transmite a mensagem recebida para o dispositivo, através do caminho “3.a”. Nesta fase a mensagem

para a requisição dos parâmetros está com o *function code* "08". Após o processamento da mensagem pelo DSC a mensagem de resposta é transmitida para o ESP, pelo caminho de retorno "4.a", novamente com o *control byte* com valor "00", que será transmitida para o CCS com o mesmo protocolo via caminho "5.a".

A comunicação com o inversor solar, via caminho "b", ocorre de forma similar ao apresentado anteriormente para o DSC. A mensagem é enviada do CCS para o ESP, via caminho "2.b", com o protocolo da PHB e *control byte* com valor "00", onde o ESP recebe a mensagem e encaminha para o inversor, apresentado por "3.b". Após a análise da mensagem o inversor responde para o ESP utilizando o caminho de retorno indicado por "4.b". O ESP recebe o valor e encaminha para o CCS novamente com o *control byte* com valor "00".

Após o CCS receber tanto a mensagem pelo caminho "5.a" ou "5.b" é realizada a resposta da solicitação do software externo com o protocolo do software externo e *function code* com valor "09".

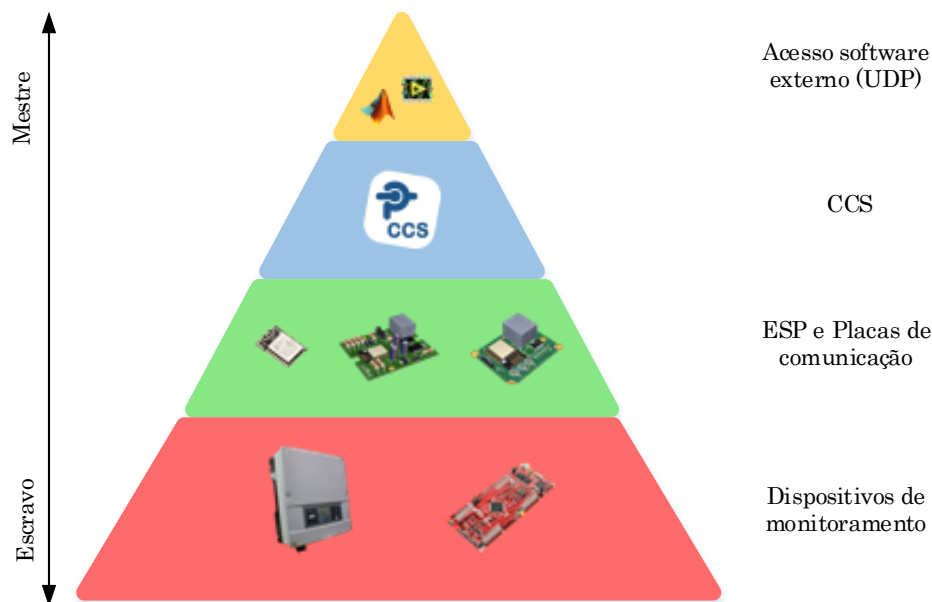
O protocolo ESP não foi utilizado nessa dinâmica, pois neste exemplo todas as configurações referentes ao microcontrolador já haviam sido realizadas, portanto, a única interação que ocorre com o ESP é utilizá-lo como ponte das informações sem alterar suas configurações.

5 SOFTWARES E FIRMWARES

Nos capítulos anteriores foram apresentados os hardwares que compõem o sistema proposto seguidos dos protocolos utilizados na comunicação. Porém, para integrar estes dois sistemas são necessários softwares ou firmwares¹.

Foi desenvolvido um software central chamado de *Central Control Software - CCS*, que é responsável pelo gerenciamento da comunicação, como a requisição de dados e o envio de configurações. Este software pode se conectar com dispositivos do tipo inversor PHB, ou com dispositivos que utilizam os protocolos criados. Caso seja necessário efetuar um controle das variáveis de uma forma mais efetiva, como o uso de técnicas de controle, é possível conectar o CCS a um software externo utilizando a conexão UDP para esta conexão. O sistema pode ser integrado com outros softwares como por exemplo, Matlab/Simulink ou LabVIEW. Para o caso de exemplo mostrado neste trabalho foi selecionado o software Matlab/Simulink.

Figura 5.1 – Esquema hierárquico entre componentes conectados à microrrede.



Fonte: Elaborado pelo autor (2018).

A Fig. 5.1 mostra a estrutura de níveis hierárquicos entre os componentes utilizados na rede proposta. A comunicação estabelecida entre os componentes dessa rede segue os

¹Na literatura há uma discussão entre a definição de software e firmware e como identificar cada um. Neste trabalho chama-se de firmware o programa que é executado em um hardware, enquanto o software é um programa que tem uma interface e é executado no computador

conceitos da comunicação mestre-escravo porém de forma “hierárquica”. O CCS é mestre perante aos dispositivos conectados nele, ou seja, neste caso a solicitação vem do mesmo e os dispositivos respondem a isso. Porém, quando um software externo é conectado, este está acima do CCS, e neste caso, o CCS que responde as solicitações do software externo.

5.1 CENTRAL CONTROL SOFTWARE

O *Central Control Software* foi criado em linguagem C# para adquirir, processar e executar os dados dos dispositivos conectados a microrrede de energia. O objetivo é estabelecer a comunicação entre o software e os dispositivos permitindo o envio de mensagens pré-estabelecidas e análise das mensagens recebidas. O software recebeu este nome pois trata-se do elemento central de comunicação.

A plataforma de desenvolvimento selecionada para o projeto foi o *Visual Studio Community* 2017, distribuído de forma gratuita pela Microsoft®. A vantagem do uso deste software, se destaca pela flexibilidade de criar aplicativos compatíveis com vários sistemas, pode abranger diversas linguagens de programação, contando com um amplo espaço para discutir melhorias e funções através da comunidade existente.

Inicialmente, o software foi desenvolvido no *command prompt*² do Windows®, com função básicas de comunicação e interação com o usuário. Esta versão viabilizou os primeiros testes validando a ideia de comunicação. Neste desenvolvimento, foram criadas as seguintes versões:

- 1.0 – Envio de mensagens padrão e recebimento pela serial;
- 2.0 – Análise mensagem recebida;
- 3.0 – Criação arquivo de dados;
- 4.0 – Análise de erros e criação de arquivo falha;
- 5.0 – Apenas a análise automática de dados;
- 5.1 – Correção de erros, integração requisição de dados automática com versões anteriores;

Até a versão 5.1 a comunicação ocorria com 1 dispositivo e conexão apenas via serial. Após a versão 5.1 foram desenvolvidas versões com interface gráfica e com maior interação com o usuário. As novas versões são chamadas de PRO e contam com todas as funcionalidades das versões anteriores e com recursos antes não implementados como a comunicação via UDP e monitoramento de até 10 dispositivos simultaneamente. A lista de versões é mostrada a seguir:

²*Command prompt* ou *prompt* de comando é um interpretador de linhas de comando nativo em sistemas baseados em Windows®.

- PRO 1.0 – Atualização da versão 5.1 com nova interface;
- PRO 1.1 – Teste de visualização de gráficos (Função removida);
- PRO 2.0 – Permite conexão UDP de um dispositivo do tipo PHB;
- PRO 2.1 – Criação de abas de dispositivo;
- PRO 2.2 – Compatibilidade com protocolo DSC e exibição dos dados em tabela;
- PRO 2.3 – Habilitado requisição automática de dados;
- PRO 2.4 – Criar arquivos do tipo .txt para dispositivos do tipo PHB e DSC;
- PRO 2.5 – Permissão de conexão com até 10 dispositivos simultaneamente;
- PRO 2.6 – Permissão de conexão com softwares externos via UDP;
- PRO 2.7 – Correção bugs provendo maior estabilidade.

Para executar as versões do programa PRO, devido as bibliotecas utilizadas, é necessário fazer a instalação do pacote de complemento .Net Framework 4.6.1³. Os dados monitorados são salvos individualmente para cada dispositivo em um arquivos de dados de texto do tipo .txt.

A última versão do software, conta com a possibilidade de conexão de até 10 dispositivos simultaneamente sendo todos através da comunicação UDP ou um deles via conexão serial se utilizado com o protocolo do tipo PHB.

Devido a diversa gama de tipos de dispositivos no mercado e permitir com a compatibilidade destes no software, foi permitido a leitura de arquivos do tipo .xml que contém a estrutura das mensagens de resposta dos dispositivos permitindo a adição de novos tipos de protocolos.

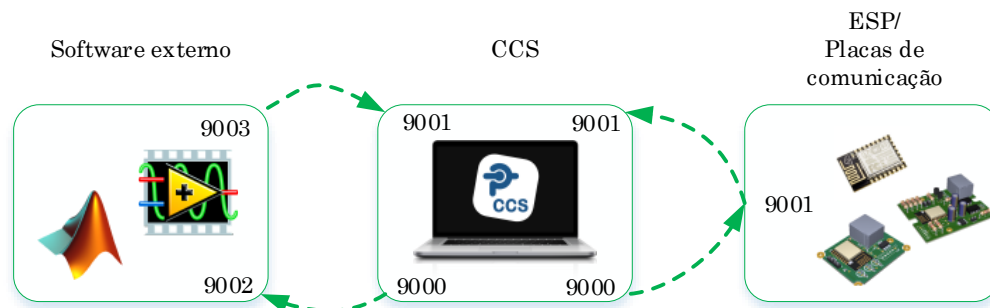
Utilizando a conexão UDP é possível conectar um software externo. Para que o software externo possa ser executado tanto no mesmo computador que o CCS esta sendo executado ou em outro computador na rede local é necessário padronizar as portas de comunicação. Mostrada na Fig. 5.2, tem-se um esquema das portas utilizadas para garantir a conexão. A saída dos dados do CCS ocorre pela porta 9000 e entrada via porta 9001, enquanto que para o software externo a saída de dados ocorre pela porta 9002 e entrada pela porta 9003. Já o ESP utiliza a mesma porta para a entrada e saída de dados, sendo esta a 9001. Portas diferentes foram utilizadas no software externo e no CCS, pois o próprio sistema operacional não permite que a mesma porta seja utilizada para envio e recebimento.

As versões PRO contam com uma interação via interface gráfica com o usuário. Através desta interface gráfica o usuário pode verificar e selecionar o dispositivo desejado para efetuar a comunicação, verificar as informações obtidas dos dispositivos e enviar

³Complemento disponível para Windows[®] superiores a versão XP de forma gratuita

requisições. As operações foram divididas em janelas que são discutidas neste capítulo, seguidas da descrição da programação e funcionamento do software desenvolvido.

Figura 5.2 – Definição das portas de comunicação UDP.



Fonte: Elaborado pelo autor (2018).

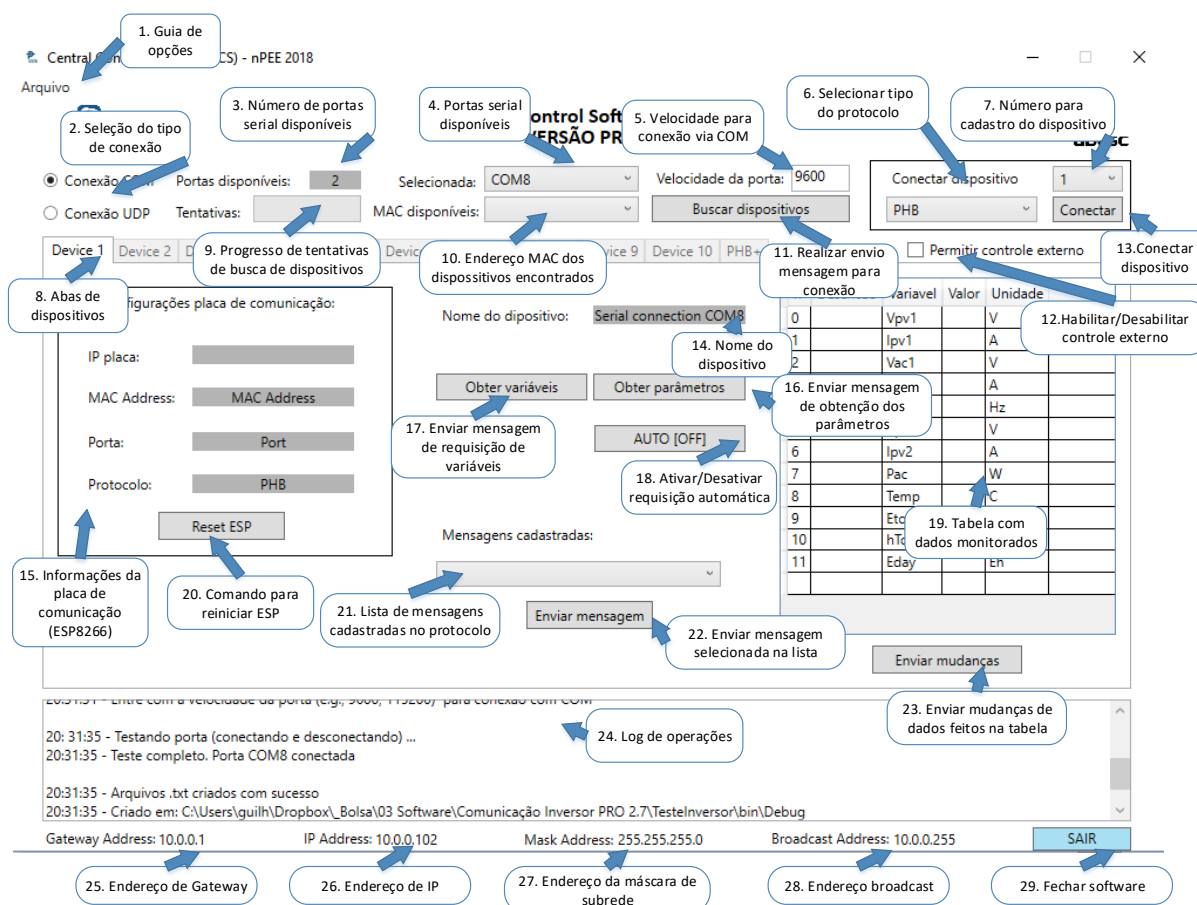
5.1.1 Janelas versão PRO 2.7

O CCS é composto de uma janela principal e mais duas auxiliares. A “janela principal” contém 11 abas, onde são cadastrados os dispositivos para monitoramento. Cada aba é nomeada com um número de dispositivo de 1 a 10, sendo que a 11ª aba recebe o nome de “PHB+”. Na aba “PHB+” pode ser acessada uma segunda janela, “janela parâmetros extras”, onde são mostrados os parâmetros extras em relação ao inversor do tipo PHB. Por fim, na “janela sobre” são mostrados dados sobre o software e as informações adicionais. As janelas são descritas nas subseções abaixo, onde são encontrados os modos de funcionamento de cada uma das funções.

5.1.1.1 Janela principal

Na Fig. 5.3 é apresentada a “janela principal”. Na parte superior da janela encontram-se as funções responsáveis pela conexão do dispositivo, o tipo de conexão parâmetros e tipo de dispositivo. Na parte central da janela há as informações referentes ao dispositivo monitorado como dados da placa de comunicação, mensagens pré-cadastradas e dados das variáveis. Na parte inferior é localizada o log de dados onde as ações do usuário e software são registradas. Na barra inferior são mostrados dados da conexão Ethernet, que são úteis ao usuário para configurar a conexão de um software externo ao CCS. Abaixo são descritas todas as funções acessadas através da janela principal com a aba de um dispositivo selecionado:

Figura 5.3 – Janela principal do CCS na versão PRO 2.7 com a aba “Device 1” selecionada.



Fonte: Elaborado pelo autor (2018).

- Guia de opções** – Apresenta atalho para a pasta onde é salvo o log de dados dos dispositivos, acesso a “janela sobre” e sair do programa;
- Seleção do tipo de conexão** – O usuário seleciona qual o tipo de conexão que o dispositivo vai conectar via UDP ou serial;
- Número de portas serial disponíveis** — Neste bloco é indicada a quantidade de portas seriais (COM) disponíveis para conexão. Esta verificação ocorre na inicialização do software;
- Portas serial disponíveis** — São listadas as portas seriais disponíveis para conexão, sendo que o usuário pode selecionar entre uma delas para conexão;
- Velocidade para conexão via COM** – Na conexão serial é necessário informar a velocidade de *baud rate* com que a transmissão é efetuada;
- Selecionar tipo do protocolo** – Ao iniciar o programa os arquivos de protocolos .xml são carregados e listados para seleção;

7. **Número para cadastro do dispositivo** – Ao efetuar a conexão do dispositivo deve-se selecionar por qual número o dispositivo será identificado, liberando assim a respectiva aba (item 8);
8. **Abas de dispositivos** – Cada dispositivo obtém uma aba para dispor os dados do mesmo;
9. **Progresso de tentativas de busca de dispositivos** – Barra que indica o progresso da busca de dispositivos (se estiver ativa);
10. **Endereço MAC dos dispositivos encontrados** – Ao executar a busca de dispositivos e receber a confirmação o endereço do MAC é adicionado a lista de para permitir a conexão;
11. **Realizar envio mensagem para conexão** – Ao pressionar o botão é executada a busca por dispositivos gerando a lista apresentada no item 10;
12. **Habilitar/Desabilitar controle externo** – A caixa de seleção permite habilitar e desabilitar o recebimento e envio de dados de um software externo;
13. **Conectar dispositivo** — Para confirmar as configurações apresentadas nos itens anteriores o usuário pressiona este botão, confirmado as informações. O software irá efetuar o teste de comunicação (para tipo COM) e registra-rá o dispositivo no software;
14. **Nome do dispositivo** – Mostra o nome do dispositivo cadastrado, se requisitado via protocolo;
15. **Informações da placa de comunicação (ESP8266)** – Esta janela contém as informações gerais de conexão em que o ESP foi configurado, sendo ela o endereço IP, endereço MAC, porta de recebimento de mensagem e protocolo cadastrado (selecionado na conexão pelo item 6);
16. **Enviar mensagem de obtenção dos parâmetros** – Envia a mensagem que requisita a atualização dos valores tabelados (no item 19). Esta mensagem varia conforme tipo de protocolo e segue arquivo .xml externo;
17. **Enviar mensagem de requisição de variáveis** – Envia a sequência de mensagens que requisitam os valores das variáveis e suas unidades para o preenchimento da tabela;
18. **Ativar/Desativar requisição automática** — Ativa a função de verificação de dados automática, sendo mostrado em seu interior o texto “[AUTO] ON”. Ao pressionar novamente o usuário desabilita a função, voltando o conteúdo para “[AUTO] OFF”;
19. **Tabela com dados monitorados** – Este espaço contém os dados das variáveis sendo os campos de descrição, nome da variável, valor e unidade;

20. **Comando para reiniciar ESP** – Mensagem para reinicialização do ESP é enviada;
21. **Lista de mensagens cadastradas no protocolo** – Conforme o protocolo cadastrado a lista de mensagens pré-cadastradas para envio são carregadas;
22. **Enviar mensagem selecionada na lista** – Envia a mensagem de envio selecionada no item 21;
23. **Enviar mudanças de dados feitos na tabela** – Botão que envia a mensagem de atualização de variáveis com os valores contidos na tabela.
24. **Log de operações** — Apresenta os principais eventos causados pelo usuário e software, auxiliando a tomar decisões e sugerindo as ações que devem ser tomadas;
25. **Endereço de Gateway** – Mostra o valor do endereço de gateway da rede;
26. **Endereço de IP** – Mostra o endereço IP do software de monitoramento;
27. **Endereço de máscara de subrede** – Mostra o valor do endereço da máscara de subrede da rede conectada;
28. **Endereço broadcast** – Mostra o valor do endereço de broadcast;
29. **Botão de sair do programa** — Permite encerrar o programa com segurança, finalizando todos processos e conexões. Este tem a mesma função que o tradicional botão de fechar “X” do Windows®.

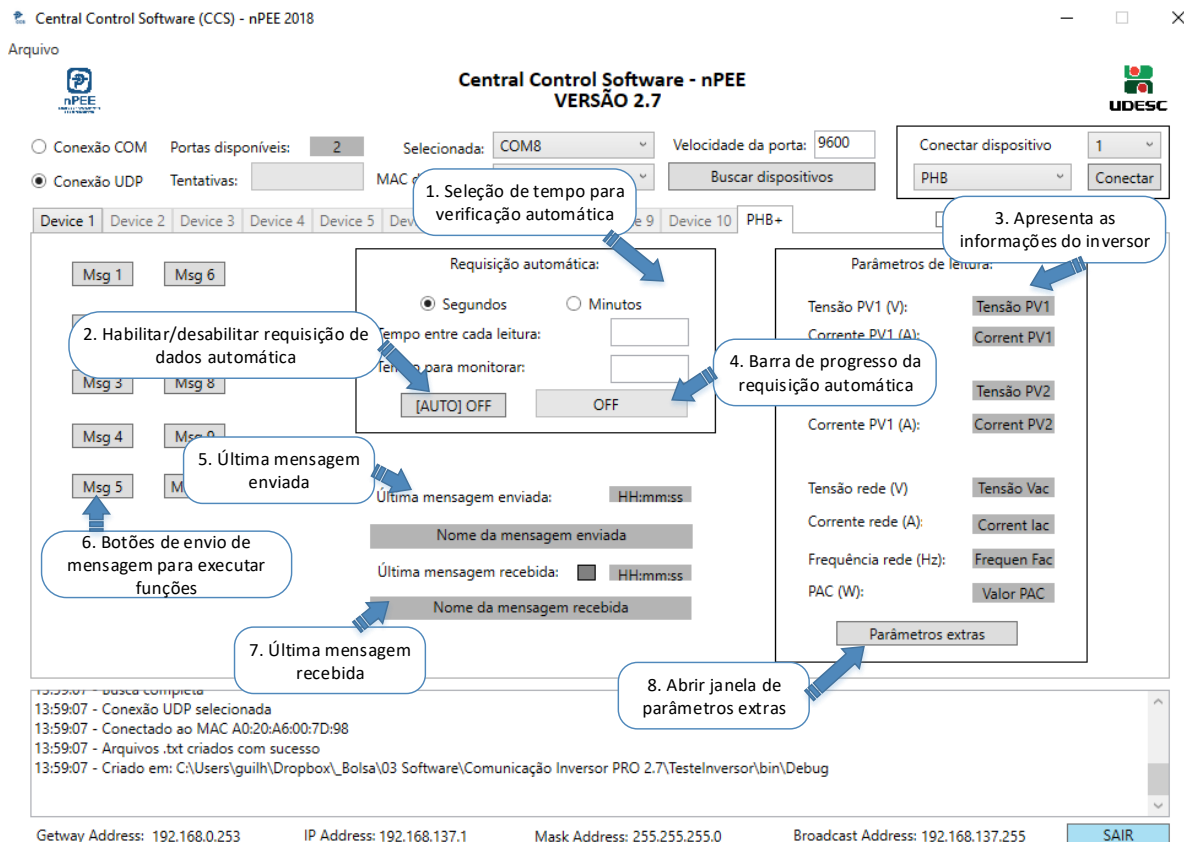
5.1.1.2 Aba PHB+

Para o inversor solar da PHB há uma aba com parâmetros dispostos de uma forma diferenciada, nomeada de “PHB+”, que é apresentada na Fig. 5.4. Esta aba é liberada para acesso caso o inversor esteja cadastrado como dispositivo de número “1”. Desta forma, os dados adquiridos pelo dispositivo de número “1” são exibidos em duas abas ao mesmo tempo.

Conforme mostrado na Fig. 5.4 as mensagens de envio estão numeradas e localizadas no canto esquerdo. Na parte central superior é possível ativar a opção de envio de requisição de dados automática, o tempo de envio entre as mensagens e tempo que deseja monitorar. O progresso total da operação pode ser verificado através da barra de status. Ao lado esquerdo pode ser visto as variáveis a serem monitoradas. Os elementos são descritos a seguir:

1. **Seleção de tempo para verificação automática** – Através desta área pode ser selecionado o intervalo de tempo e a duração total das requisições sendo possível escolher se o tempo inserido é em minutos ou segundos;

Figura 5.4 – Janela principal do CCS na versão PRO 2.7 com a aba “PHB+” selecionada.



Fonte: Elaborado pelo autor (2018).

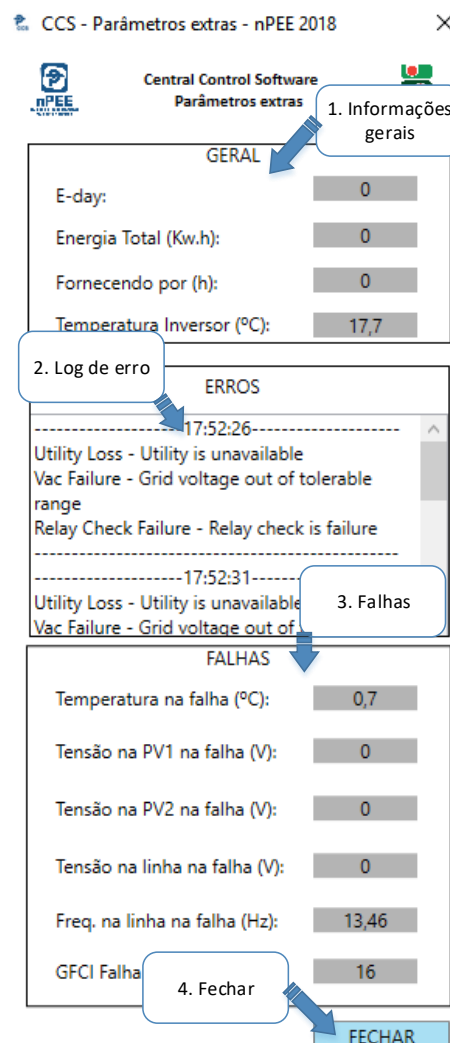
2. **Habilitar/Desabilitar requisição automática de dados** – Botão que inicia ou interrompe a requisição automática de dados com os valores de tempo digitados no item 1;
3. **Apresenta as informações do inversor** – Ao contrário das demais abas onde os dados são mostrados em uma tabela e totalmente configuráveis, nesta os dados são mostrados em campos já estabelecidos;
4. **Barra de progresso da requisição automática** – Mostra o status da requisição automática de dados;
5. **Última mensagem enviada** – Local onde é indicado o horário e nome da última mensagem enviada para o dispositivo;
6. **Botões de envio de mensagem para executar funções** – Envia as mensagens pré configuradas do arquivo .xml;
7. **Última mensagem recebida** – Similar ao item 5, este mostra os dados da última mensagem recebida.

8. **Abrir janela de parâmetros extras** – Abre a janela de parâmetros extras que é discutida na subseção 5.1.1.3.

5.1.1.3 Janela de parâmetros extras

A janela de parâmetros extras pode ser acessada clicando no botão “Parâmetros extras” da aba “PHB+”, indicado pelo número 8 na Fig. 5.4. Esta janela apresenta dados adicionais sobre o inversor solar além de erros e dados das falhas ocorridas. As falhas apresentadas no software são as últimas registradas pelo inversor solar e podem não ter ocorrido durante a execução do software. A interface da janela é apresentada na Fig. 5.5 e suas funções são descritas abaixo:

Figura 5.5 – Janela de parâmetros extras do CCS na versão PRO 2.7.



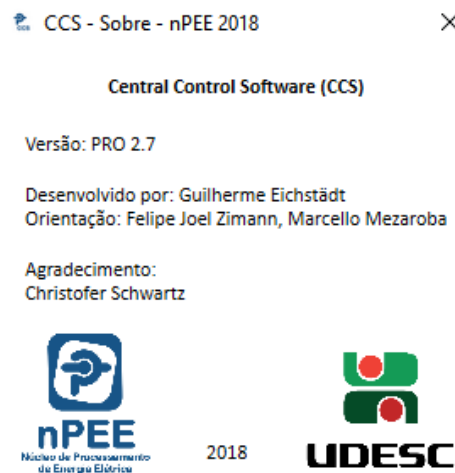
Fonte: Elaborado pelo autor (2018).

1. **Informações gerais** — São apresentados quatro parâmetros de leitura que não estão apresentados na interface principal;
2. **Log de erros** — Similar ao arquivo do tipo .txt criado para informar os erros que foram ocasionados durante a leitura dos dados, o log de erros, apresenta os erros detectados separados por horário;
3. **Falhas** — São apresentados parâmetros de leitura e os dados das últimas falhas ocorridas com o inversor;
4. **Fechar** — Utilizado para fechar a janela de parâmetros extras.

5.1.1.4 Janela sobre

Mostrada na Fig. 5.6 a “janela sobre” não apresenta função sobre as operações executadas com os dispositivos e apenas tem a função de informar ao usuário sobre o desenvolvimento do software. Diferentemente da janela de parâmetros, esta janela impede a utilização do software se estiver aberta, contendo a atenção do usuário para as informações da mesma.

Figura 5.6 – Janela sobre o CCS na versão PRO 2.7.



Fonte: Elaborado pelo autor (2018).

5.1.2 Programação e funcionamento

Para a criação do software em linguagem C# buscou-se manter uma padronização entre elementos utilizados facilitando a modificação e adaptação do software em aplicações futuras. O Apêndice A apresenta a descrição dos objetos de cada janela e abas. Esta subseção apresenta a descrição das funções criadas procurando trazer de forma mais clara

o funcionamento do software atrelado a forma com que ele foi programado. Ao final é apresentado os arquivos de dados gerados pelo software.

5.1.2.1 Funções

O software foi dividido em funções, o que permite organizar as rotinas e identificar os processos. A interação de uma função com a outra ocorre conforme a ação do usuário, porém um fluxo esperado de funcionamento pode ser traçado e será seguido para a explicação do funcionamento do software. A aplicação é composta pelas funções abaixo, que serão descritos a seguir:

- **MainWindow** – Inicializa componentes;
- **ConfigTimers** – Configura timers;
- **GetProtocols** – Obtém protocolos cadastrados nos arquivos .xml;
- **Start** – Desabilita botões, e verifica portas conectadas;
- **GetBroadcastIP** – Obtém endereço de broadcast, IP e mascara do computador;
- **GetHostMask** – Obtém mascara de subrede;
- **GetHostIP** – Obtém endereço IP do computador;
- **GetDefaultGateway** – Obtém endereço de gateway;
- **StartListening** – Habilita inicio da conexão UDP;
- **Button_Click** – Clique no botão “Buscar dispositivos” - Envia mensagem de busca por dispositivos;
- **ReceiveDataUDP** – Evento que ocorre quando recebe mensagem via UDP e o *control byte* é analisado;
- **Button_Click_OK** – Clique no botão “Conectar” - Encaminha teste da porta, inicia recebimento de mensagens;
- **CheckSerialPort** – Teste da COM selecionada, Habilita botões;
- **ReceiveData** – Recebimento das mensagens na porta serial;
- **CreateFile** – Cria arquivos do tipo .txt;
- **UdpSender** – Envia mensagem via UDP;
- **ConfigIP** – Analisa mensagem do protocolo ESP;
- **ExternalRequest** – Analisa mensagem recebida de um software externo;
- **Msg_AnalysisPHB** – Analisa a mensagem recebida para o protocolo PHB;
- **Msg_AnalysisDSP** – Analisa a mensagem recebida para o protocolo DSC;
- **Exter_send** – Envia a mensagem para o software externo;
- **SendMsg** – Envia mensagem para serial;
- **LoadValueDSP** – Carrega valor na tabela do dispositivo;

- `Write_txt_DSP_value` – Escreve no arquivo do tipo .txt os valores das variáveis;
- `Write_txt_DSP_name` – Escreve no arquivo do tipo .txt os valores dos nomes das variáveis e unidades;
- `LoadCollectionDSP` – Carrega nomes ou unidades na lista do dispositivo;
- `LoadValuePHB` – Carrega valor na tabela do dispositivo do tipo PHB;
- `Write_txt_PHB` – Escreve no arquivo do tipo .txt de dados do dispositivo PHB;
- `Write_txt_Error` – Escreve no arquivo do tipo .txt de erros do dispositivo PHB;
- `Button_Click_Auto` – Botão para verificação automática de dados na aba PHB+;
- `EnableAuto` – Habilitada função automática de verificação de informações na aba PHB+;
- `DisableAuto` – Desabilita função automática de verificação de informações na aba PHB+;
- `OnTimedEvent` – Evento para envio de mensagem automática na aba PHB+;
- `SendMsg` – Analisa tipo de mensagem para envio para COM ou via UDP;
- `LoadCollectionPHB` – Carrega variáveis do tipo PHB na lista;
- `MainWindow_Closed` – Evento clique no botão de fechar;
- `Button_Click_Exit` – Botão de fechar programa;
- `Button_Click_Parametros` – Abre janela de parâmetros extras;
- `Button_Click_Sobre` – Abre janela sobre software;
- `Enabl_extern_Checked` – Habilita/Desabilita requisição de dados automática devido a requisição externa;
- `Button_Click_X_param` – Seleciona mensagem para atualização de parâmetros
- `Button_Click_X_names` – Seleciona mensagem para atualizar nomes das variáveis;
- `Button_Click_X_chang` – Enviar variáveis para mudar valores;
- `Button_Click_nº` – Clique nos botões para enviar mensagens.

O fluxo de funcionamento do software pode ser visto no fluxograma apresentado no Anexo B, o qual segue a ordem lógica e o modo recomendável da utilização do software. Primeiramente para a inicialização dos componentes do software são executadas as funções: `MainWindow`, `ConfigTimers` e `GetProtocol`.

A função `Start` desabilita os botões de envio de mensagem e define a lista de portas seriais disponíveis para conexão. Se nenhuma porta é detectada o usuário é informado que necessita conectar antes de iniciar o software. Caso haja apenas uma porta esta é selecionada. Entretanto, se houver mais de uma opção, as mesmas são listadas e o usuário pode selecionar a porta desejada. Em seguida, os dados de conexão do software são carregados, como endereços de IP e de broadcast através da função `GetBroadcastIP` e `StarListening` que habilita o recebimento de mensagens via UDP. Finalizando estes

processos, o software ficará aguardando a ação do usuário. A continuação lógica do software é o usuário efetuar a busca de dispositivos que ocorre através do botão “Buscar dispositivos”. Ao clicar no botão é enviada a mensagem de “conectar ESP a rede”, via broadcast, cinco vezes com intervalos de dois segundos. Uma barra de progresso mostra o status da busca destes dispositivos.

Para o funcionamento do software foram criadas duas listas de dados a lista *pre-client* e *client*. A lista *pre-client* contempla os dados dos possíveis dispositivos a serem monitorados, enquanto a lista *client* contempla os dados dos dispositivos já conectados. Desta forma, quando o software busca os dispositivos e os encontra na rede, os dados do dispositivo são salvos na lista *pre-clients*. Apenas na confirmação de conexão, através do botão “Conectar”, o valor dos dados é transferido para a lista *client*, fazendo com que este dispositivo agora está conectado e pronto para enviar e receber as informações.

A confirmação de conexão feita através do botão “Conectar”, faz iniciar a função `Button_Click_Ok`. Nesta função dois tipos de conexão podem ser selecionadas, via serial ou UDP:

- **Conexão serial** – Este caso é iniciado verificando se os dados de conexão da porta serial estão corretos. Após a verificação, ocorre a conexão com a respectiva porta e é iniciado o recebimento das mensagens. O recebimento ocorre através de uma *thread*, ou seja, em paralelo com o funcionamento do programa principal, através da função `ReceiveData`. No fim, a função é encaminhada para a criação do arquivo de log e de erros;
- **Conexão UDP** – Neste cenário é analisado o endereço MAC do dispositivo para identificar o número do dispositivo. Dessa forma, é possível habilitar a respectiva aba do dispositivo e registrar os dados do mesmo na lista *client*. No final da função é criado o arquivo de log e de erros. Da mesma forma que a conexão via serial os dados recebidos via UDP também são verificados através de uma *thread* pela função `ReceiveDataUDP`.

Os dados recebidos via conexão UDP na função `ReceiveDataUDP` são analisados quanto a tipo de mensagem através do *control byte*:

- Se *control byte* tem o valor “0” a mensagem é do tipo dispositivo externo que pode ser do tipo PHB, tratado em `Msg_AnalysisPHB` ou se for do tipo DSC tratado em `Msg_AnalysisDSP`;
- Se *control byte* tem o valor “1” a mensagem é do tipo configuração do ESP e seu tratamento é feito em `ConfigIP`;

- Se *control byte* tem o valor “2” a mensagem é do tipo de requisição de dados externos que é tratado em `ExternalRequest`.

A verificação da porta serial é feita na função `CheckSerialPort`. Através da porta serial selecionada é efetuado o teste, com o software que solicita a conexão e desconexão com a porta. Caso o mesmo não tenha problema é indicado que o teste foi concluído com sucesso e habilitado os demais botões para a interação do usuário.

Para a criação do arquivo de texto, onde são salvos os dados e erros dos dispositivos, é utilizada a função `CreateFile`. Nesta função inicialmente é verificado se o arquivo não é existente, para a criação do mesmo. Em seguida ocorre a escrita dos cabeçalhos dos arquivos. Como o inversor solar da PHB tem as variáveis de monitoramento fixas, caso o dispositivo conectado seja deste tipo, o nome das variáveis de monitoramento serão escritas no arquivo.

A função `LoadCollectionDSP` é responsável por atualizar os valores da lista *client* indicado pelo dispositivo e atualizar as tabelas onde são mostrados os valores das variáveis.

A requisição automática de dados ocorre através do botão “AUTO[ON]”, que faz com que interrupções por tempo sejam ativadas. Para o caso da aba PHB+ a função `Button_Click_Auto` é chamada, enquanto que nas demais abas a função `Button_Click_XX_par_auto`⁴ é chamada, sendo que os eventos que ocorrem a cada intervalo de tempo é nomeado por `OnTimeEventXX`⁴.

5.1.2.2 Arquivos de textos gerados via CCS

Todos os dados adquiridos são armazenados em arquivos de texto criados após a conexão com o dispositivo. Para cada dispositivo é criado um arquivo que contém os dados das leituras efetuados, porém, para o dispositivo do tipo PHB um arquivo extra é criado com o log de erros. Todos arquivos são salvos em uma pasta de nome “*Log*” localizada no diretório em que o programa (.exe) está sendo executado. O software deve ser executado em modo administrador para que a criação e modificação dos arquivos sejam efetuadas com êxito.

O arquivo que contém os dados de leitura do inversor é nomeado de “CCS hh.mm.ss - Dados DV XX”, já o de erros “CCS hh.mm.ss - Erros DV XX”, onde “hh:mm:ss” são as horas, minutos e segundos em que o programa foi executado e “XX” representa o número em que o dispositivo foi cadastrado.

Um exemplo do arquivo de dados adquiridos de um dispositivo é apresentado na Fig. 5.7. Este dispositivo teve como variáveis recebidas o valor de *LoopCount*, *-LoopCount*,

⁴Onde “XX” é substituído pelo número do dispositivo.

$vc1$, vg , $iL1$, ig , vbc , ia e ib . As unidades das variáveis também são registradas no arquivo e estão entre parênteses ao lado da respectiva variável.

Figura 5.7 – Modelo de arquivo do tipo .txt de dados criado pelo software.

```
#####
#          CENTRAL CONTROL SOFTWARE - NPEE 2018          #
#          PRO VERSION                                     #
#####
Log de dados      19/02/2018 14:20:14
19-02-2018      14:20:29      LoopCount      -LoopCount      vc1(V)  vg(V)  iL1(A)  ig(A)  vbc(V)  ia(A)      ib(A)
19-02-2018      14:20:30      14441      -14441      0      0      0      0      -428,52 -29,17571 -30,20958
19-02-2018      14:20:31      59196      -59196      0      0      0      0      -428,52 -29,17571 -30,20958
19-02-2018      14:20:31      13111      -13111      0      0      0      0      -428,52 -29,17571 -30,20958
19-02-2018      14:20:32      9661      -9661      0      0      0      0      -428,52 -29,17571 -30,20958
19-02-2018      14:20:33      50572      -50572      0      0      0      0      -428,52 -29,17571 -30,20958
19-02-2018      14:20:33      48822      -48822      0      0      0      0      -428,52 -29,17571 -30,20958
```

Fonte: Elaborado pelo autor (2018).

A Fig. 5.8 mostra um exemplo de arquivo de erros que apresenta os erros detectados na execução do software juntamente das últimas informações sobre as falhas ocorridas, criado apenas para dispositivos do tipo PHB.

Figura 5.8 – Modelo de arquivo do tipo .txt de erros criado pelo software.

```
#####
#          TESTE DE COMUNICACAO INVERSOR PHB - NPEE 2017          #
#          PRO VERSION                                     #
#####
Log de erros      16/03/2017 17:52:21
----- ERROS -----
16-03-2017      17:52:26      Utility Loss - Utility is unavailable
16-03-2017      17:52:26      Vac Failure - Grid voltage out of tolerable range
16-03-2017      17:52:26      Relay Check Failure - Relay check is failure
----- FALHAS -----
FTemp  FVpv1  FVpv2  FVLin1  FFLin1  FGFCI
0,7    0      0      0      13,46  16
----- ERROS -----
```

Fonte: Elaborado pelo autor (2018).

Ao serem registradas as variáveis elas são dispostas no arquivo de texto separadas por tabulação simples. Desta forma, o arquivo criado é facilmente exportado para programas de processamento de dados, o que possibilita a criação de gráficos e análise mais detalhada dos dados.

5.2 FIRMWARE ESP8266

As placas que contém o microcontrolador ESP8266 são utilizadas para realizar a comunicação sem fio entre os dispositivos e o CCS. Este capítulo apresenta as funções e modo de funcionamento do firmware necessário para o funcionamento do microcontrolador.

A programação do firmware para o ESP8266 foi utilizada a interface Arduino IDE, que permite uma programação com uso de linguagem C. Porém, a compatibilidade da placa ESP8266 não vem integrada à IDE do Arduino. Para isso deve ser instalada uma

biblioteca adicional ao software, como apresentado em Kruz (2017), com a documentação acessada em Github (2016a) e Github (2016b).

O firmware tem dois objetivos. O primeiro deles é estabelecer e configurar os requisitos para a conexão à rede sem fio, o segundo é identificar a mensagem recebida, e tomar a ação requerida.

5.2.1 Funcionamento

O firmware pode ser simplificado em duas partes principais: a função de configuração (*setup*) e repetição (*loop*). O fluxograma que descreve o funcionamento do *firmware* pode ser visto no Apêndice C e suas principais funções serão discutidas a seguir.

Inicialmente as definições de rede e conexão serial são configuradas, como parâmetros de *baud rate*, nome e senha da rede sem fio a ser conectada. Neste processo também são identificados alguns parâmetros como endereços de IP, broadcast e máscara de subrede.

Após a configuração inicial, o firmware entra um *loop* de verificação onde é confirmada a conexão com a rede sem fio. Caso esteja conectado a rede, o software permanece verificando as portas UDP e serial, aguardando o recebimento de alguma mensagem. Se a mensagem for recebida via conexão serial, a função `Send_to_UDP` é executada, onde são enviadas as mensagens recebidas para o CCS via UDP. Se uma mensagem via UDP for recebida, a função `Check_if_bridge` é chamada, e através dela ocorre a verificação do *control byte* para identificar o tipo de mensagem:

- *Control byte* com o valor de “00” indica que a mensagem recebida deve ser encaminhada para a serial do dispositivo ocorrendo através da função `Send_to_COM`;
- *Control byte* com o valor de “01” indica que a mensagem é de configuração do ESP, portanto será analisada na função `ESPFunctions`, onde estão cadastradas as mensagens do protocolo criado para executar ações com o próprio ESP.

5.3 FIRMWARE DSC

Para a programação do firmware do DSC, foi utilizado o *Code Composer Studio*, juntamente da adaptação de um exemplo de comunicação SCI⁵ que o próprio fabricante fornece.

⁵*Serial Communications Interface* - Usualmente chamada de UART, interface de comunicação entre dispositivos. Este tipo de comunicação é utilizado para curtas distâncias e de forma assíncrona.

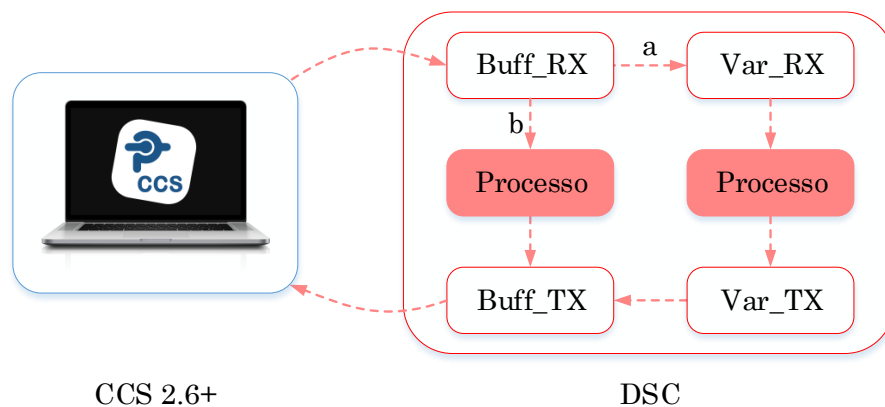
5.3.1 Funcionamento

O funcionamento deste firmware baseia-se apenas em executar as funções contidas no protocolo desenvolvido para o DSC. O Apêndice D contém o fluxograma que apresenta as funções que compõem este software.

Inicialmente, devem ser definidas as variáveis de escrita: nome do dispositivo, nome das variáveis e nome das unidades das variáveis que são utilizadas ao verificar as respectivas informações no CCS e arquivo de texto utilizado.

O firmware criado baseia-se na interação entre quatro principais variáveis. A variável “Buff_RX” contém o valor na íntegra da mensagem recebida, já a variável “Buff_TX” contém o valor na íntegra da mensagem à enviar para o CCS. As variáveis monitoradas antes e depois dos “Var_RX” e “Var_TX”⁶ contém apenas o valor das variáveis monitoradas antes e depois do processamento, ou seja, em comparação com as duas primeiras variáveis apresentadas, estas não incluem o header, *control byte* e já tem o valor do dados convertido do padrão IEEE-754 para float.

Figura 5.9 – Variáveis utilizadas para recebimento de mensagem pelo CCS no DSC.



Fonte: Elaborado pelo autor (2018).

Ao DSC receber uma mensagem completa, é efetuado o processamento da mensagem na função `msg_analysis` e pode trazer dois casos mostrados na Fig. 5.9:

- **Caminho “a”** - Neste caso a mensagem contém valores das variáveis monitoradas, desta forma os valores são salvos na variável “Var_RX”. O processamento dos valores monitorados é feito pela função `change_var`, que salva os valores modificados na variável “Var_TX”. Logo a mensagem de confirmação deve ser criada e isto ocorre através da variável “Buff_TX” na função `confirm_changes` finalmente encaminhando a mensagem para o CCS via função `scia_xmit`;

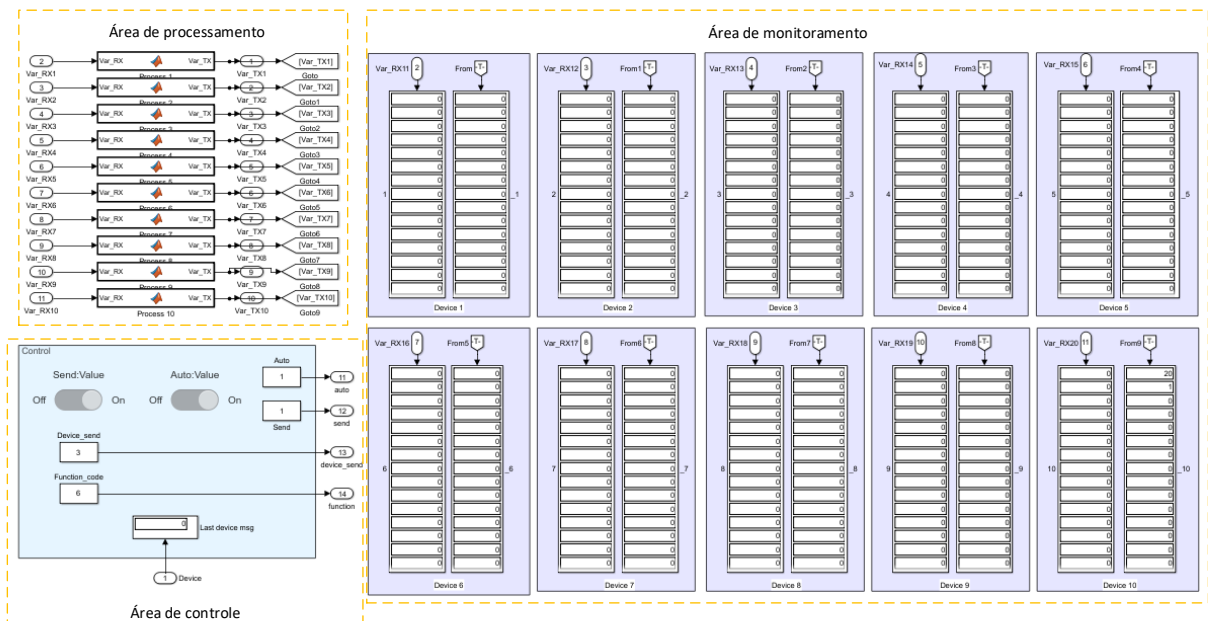
⁶O uso da nomenclatura “RX” remete ao valor recebido e “TX” ao transmitido

- **Caminho “b”** - Neste caso a mensagem não contém valores das variáveis monitoradas. Desta forma, uma mensagem de resposta, conforme protocolo DSC, será enviada diretamente para o CCS, não modificando as variáveis “Var_RX” e “Var_TX”. Portanto, a mensagem de envio é salva na variável “Buff_TX” e encaminhada para o CCS via função `scia_xmit`.

5.4 SOFTWARE EXTERNO

O CCS em sua última versão permite a conexão de um software externo para monitoramento dos dados. Neste caso, a requisição de dados ocorre pelo software externo que solicita os dados aos CCS, que por sua vez, requisita aos dispositivos. Esta inclusão permite um tratamento das variáveis recebidas, realizando operações com as variáveis que não eram possíveis apenas com o CCS.

Figura 5.10 – Tela no Simulink em que o usuário efetua interações com dispositivos e requisição de mensagens separada em três áreas.



Fonte: Elaborado pelo autor (2018).

Para estudo de caso foi selecionado o software Matlab com o uso da plataforma Simulink que envia e processa as mensagens recebidas conforme protocolo criado. A Fig. 5.10 apresenta a interface criada no Simulink que é dividida em três áreas:

- **Área de processamento** - Funções para processamento das variáveis, no qual é recebido como entrada a variável “Var_RX” e tem como saída o valor da variável “Var_RX”;

- **Área de monitoramento** - Contém as tabelas que mostram o valor variáveis monitoradas dos dispositivos. Nestas tabelas, além do valor da variável recebida, via mensagem, é mostrado o valor após o processamento;
- **Área de controle** - Encontra-se um painel de controle onde o usuário seleciona, conforme protocolo, qual a mensagem que será enviada para o CCS e para qual dispositivo que deseja ser requisitada a mensagem. O *slider Send* controla quando é permitido o envio das mensagens e o *slider Auto* é responsável por habilitar o envio das mensagens de forma automática conforme o tempo de execução do Simulink, similar a função de enviar informações automaticamente no CCS.

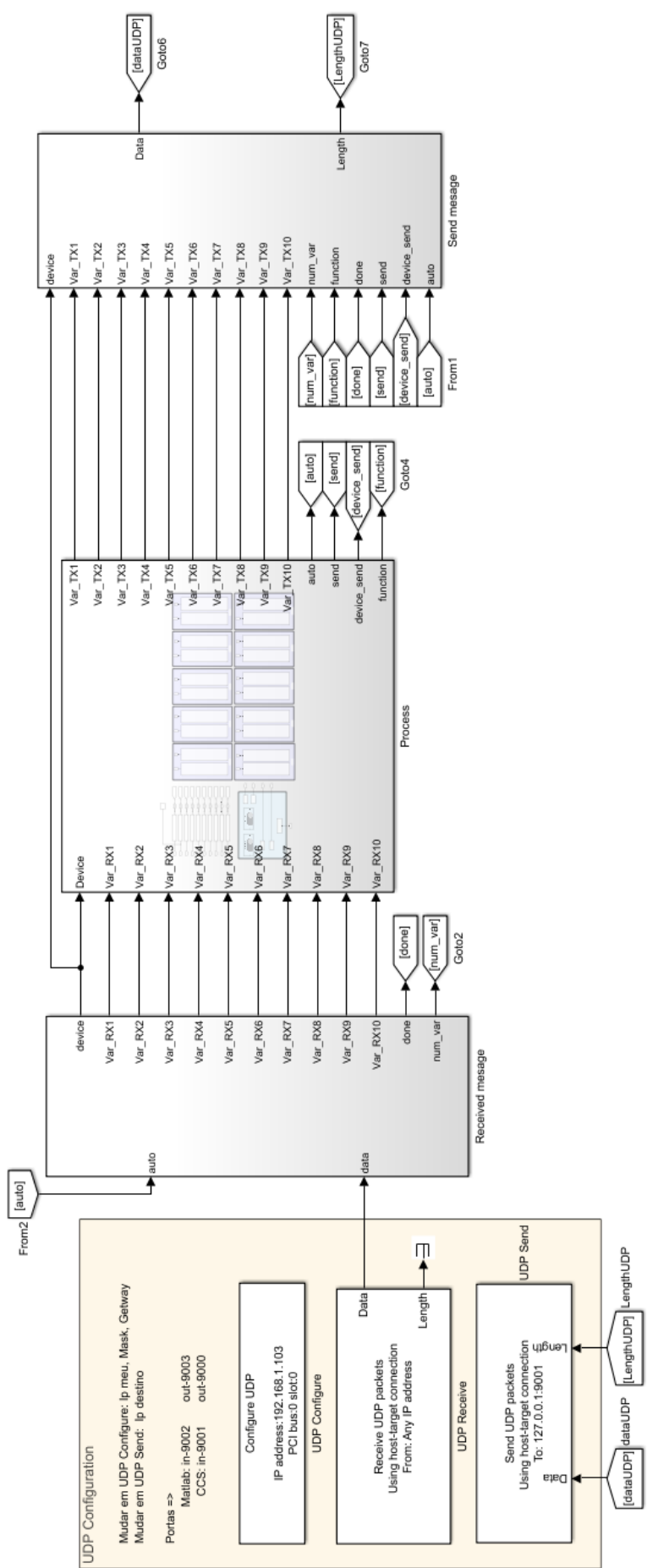
5.4.1 Funcionamento

De forma análoga do funcionamento dos firmware do DSC, este software, também utiliza variáveis nomeadas de “Var_RX”, para o valor das variáveis recebidas e “Var_TX” para armazenar os valores após processamento. Porém neste caso, como podem ser recebidos os dados de até dez dispositivos, um número é acrescido no final da variável conforme respectivo dispositivo. Assim, para as variáveis do dispositivos “1” temos “Var_RX1” e “Var_TX1”, para o dispositivo “2” temos “Var_RX2” e “Var_TX2”, e assim por diante, até o dispositivo “10” que tem as variáveis “Var_RX10” e “Var_TX10”.

O fluxograma de funcionamento do software externo é visto no Apêndice E e o funcionamento baseia-se em quatro processos que foram separados na tela do Simulink apresentado na Fig. 5.10:

- **Configuração UDP** - O bloco **UDP Configuration** contém todas as configurações que devem ser feitas pelo usuário antes de iniciar o software;
- **Mensagens recebidas** - Na função **Received message**, ocorre a análise das mensagens seguindo o protocolo criado, verificando *control byte* e *function code*;
- **Processo** - No bloco **Process** que se encontra a interface mostrada anteriormente na Fig. 5.10 e é responsável por conter os dados em que o usuário mais interage durante um monitoramento;
- **Envio de mensagem** - A função **Send message** contém a lógica do envio de mensagens, respeitando os dados inseridos na seção de controle ou recebidos a partir do CCS.

Figura 5.11 – Blocos de interação no Simulink, via Matlab, que executam as funções programadas.



Fonte: Elaborado pelo autor (2018).

6.1 ESTUDO DE CASO I - DSC

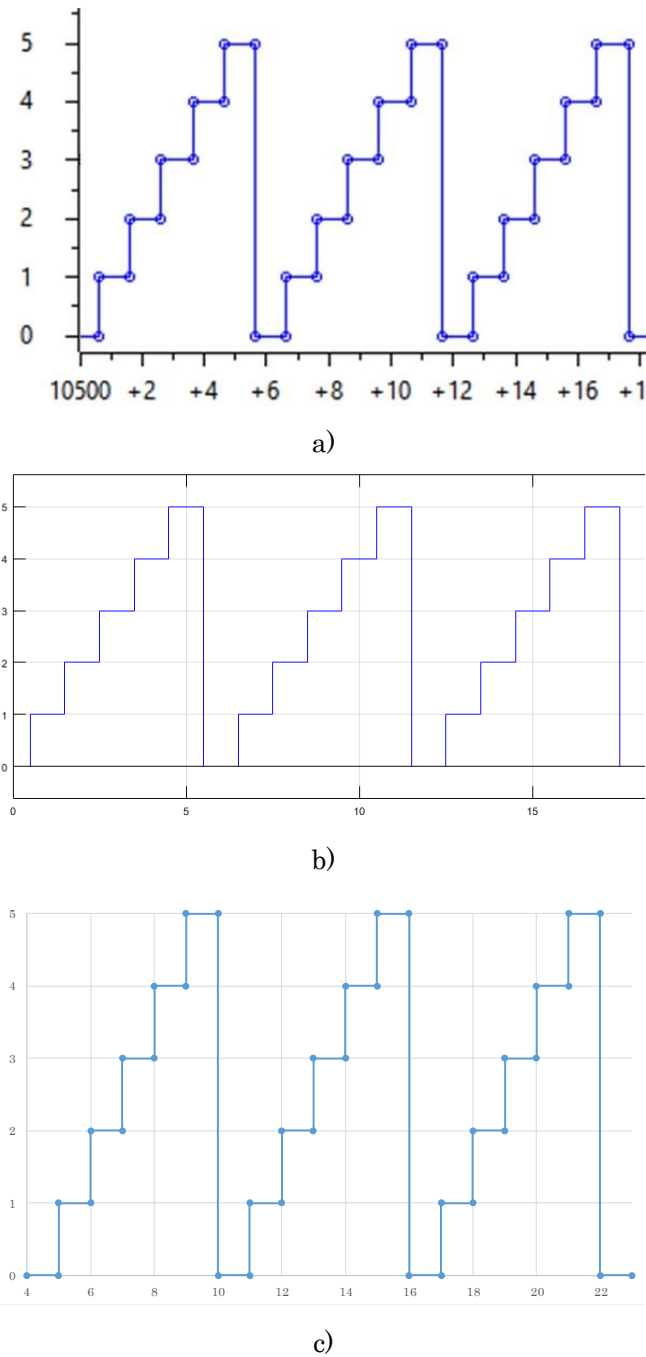
Neste cenário o DSC é conectado, via placa de comunicação, ao CCS para monitoramento dos dados juntamente ao software externo Matlab. Foi considerado que o DSC obtém uma variável nomeada de “VarPi” que contém o valor de “3.141593” e o CCS já efetuou a requisição de nome e unidade das variáveis e deseja-se que esse valor seja monitorado pelo CCS e software externo. A Fig. 6.3 apresenta as capturas das telas do software externo, CCS, depuração do ESP8266, depuração do DSC e log de dados apresentados em ordem cronológica:

1. **Software externo** – Inicialmente os valores das variáveis apresentam o valor nulo pois nenhuma requisição foi feita. Os valores das variáveis são requisitados através desta interface;
2. **CCS** – O valor da variável é nulo antes de efetuar a requisição do valor, da mesma forma que o software externo. Uma mensagem de requisição será enviada para o IP do dispositivo requisitado após recebido a requisição via software externo;
3. **Placa de comunicação** – A placa de comunicação recebe a mensagem de solicitação do valor das variáveis e requisita ao DSC o valor, enviando a mensagem recebida para a porta serial;
4. **DSC** – O valor “3.141593” é mostrado na variável de saída “Var_TX” que foi solicitado via mensagem;
5. **Placa de comunicação** – A mensagem com o valor é enviado do DSC pela porta serial e recebida pela placa de comunicação que envia a mensagem recebida para o CCS;
6. **CCS** – Ao receber a mensagem o valor é atualizado na tela para a visualização do usuário e é salvo o valor no log de dados;
7. **Log de dados** – Os dados monitorados são salvos no arquivo de texto contendo o horário em que foram registrados;
8. **Software externo** – Valor das variáveis é recebido pelo software externo e mostrado no respectivo display.

Para realizar uma simulação com uma interação entre os dados recebidos e enviados via CCS, foi criado um código no DSC que analisa a variável recebida, e a cada requisição aumenta o valor da mesma em uma unidade, chegando no máximo ao valor de 5 e reiniciando o processo. O resultado desta simulação é mostrado nos gráficos apresentados na Fig. 6.2. Nesta imagem são comparados três gráficos, que mostram como os dados transmitidos do DSC (gráfico “6.2.a”), chegam ao CCS que gera o log de dados, produzindo

o gráfico “6.2.c”, e são transmitidos ao software externo gerando o gráfico no Simulink e mostrado no gráfico em “6.2.b”.

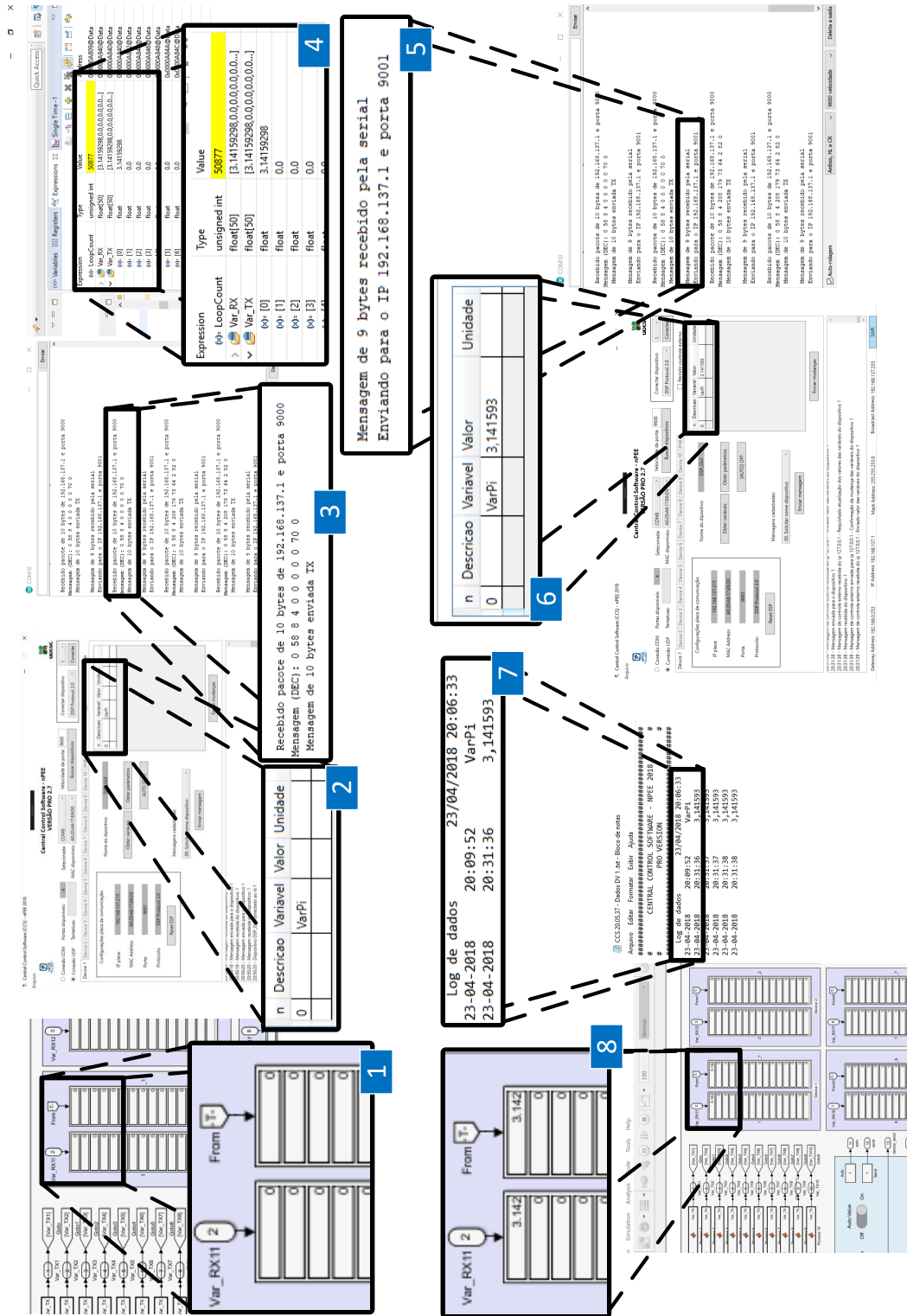
Figura 6.2 – Gráficos gerados através da simulação da interação da requisição do software externo com CCS.



- a) Gráfico gerado a partir do *Code Composer Studio* efetuando a leitura dos dados enviados pelo DSC.
- b) Gráfico gerado a partir do *Matlab/Simulink* através da leitura dos dados recebidos.
- c) Gráfico gerado a partir dos dados registrados no log de dados.

Fonte: Elaborado pelo autor (2018).

Figura 6.3 – Telas de softwares mostrando requisição de atualização solicitada pelo software externo para o DSC.



Fonte: Elaborado pelo autor (2018).

6.2 ESTUDO DE CASO II - INVERSOR SOLAR

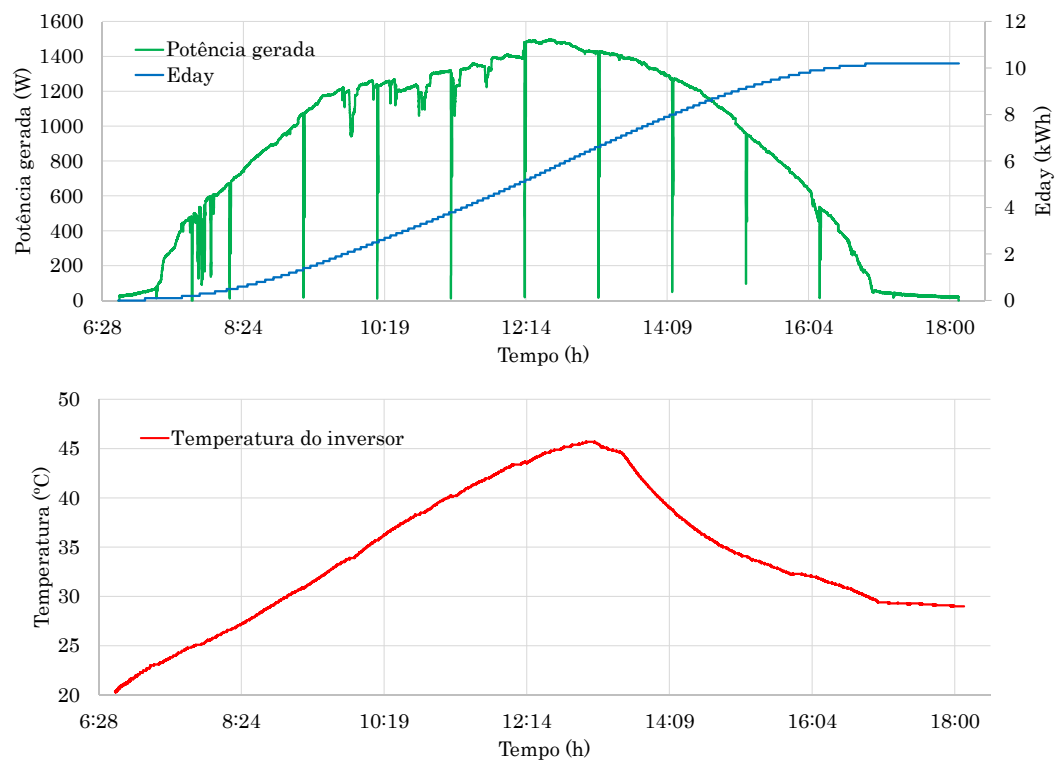
Para a validação da comunicação entre inversor solar e software de monitoramento, foi montada uma rede composta pelo inversor solar conectado ao CCS via placa de comunicação com o software externo requisitando os dados. A Fig. 6.5 apresenta as capturas das telas do software externo, CCS, depuração do ESP8266 e log de dados apresentados em ordem cronológica:

1. **Software externo** – Inicialmente a tela do software externo não contém nenhuma informação quanto as dados pois os mesmos ainda não foram requisitados;
2. **CCS** – O CCS contém as informações de nomes e unidades das variáveis que já estão pré-cadastradas no software, porém os valores estão nulos devido a ainda não ter recebido resposta da solicitação dos valores;
3. **ESP** – A mensagem de solicitação dos valores é enviada pelo CCS e recebida pelo ESP8266 que encaminha os bytes para a porta serial, onde está conectado o inversor solar;
4. **ESP** – Após o processamento da mensagem o inversor responde a solicitação com a mensagem encaminhada via serial para a placa de comunicação. Desta vez ao receber a informação a placa encaminha a mensagem para o CCS via rede UDP;
5. **CCS** - Os dados são recebidos pelo CCS e processados completando os valores na tabela;
6. **Log de dados** – Ao obter os dados o CCS também cria o arquivo de texto com os valores recebidos;
7. **Software externo** – Finalmente os dados são enviados via UDP do CCS para software externo onde os valores podem ser processados e visualizados em tabela.

Para validar o uso deste sistema em uma rede real foi instalada a placa de comunicação, conectada ao CCS, no inversor solar PHB instalado no laboratório do Núcleo de Processamento de Energia Elétrica (nPEE). Foram coletados dados em um dia ensolarado e de temperatura amena onde é possível analisar diversos parâmetros quanto a geração solar. A Fig. 6.4 apresenta o gráfico da potência gerada durante o dia e a temperatura em que o inversor solar operava.

Através do monitoramento, pode-se observar que nos dados de potência há uma queda da potencia gerada cada hora. Este comportamento é o resultado de um algoritmo interno ao inversor solar. Segundo o fabricante, este algoritmo faz a busca pelo ponto de máxima potência em situações onde possam existir sombreamentos parciais sobre o conjunto de painéis solares.

Figura 6.4 – Gráficos de potência e temperatura do inversor solar gerados a partir dos dados salvos no log de dados gerados pelo CCS.



Fonte: Elaborado pelo autor (2018).

6.3 DISPOSIÇÕES FINAIS

Através deste trabalho pode-se obter um sistema funcional com uma interface amigável de comunicação baseado em dispositivos conectados em uma microrrede de energia. Este trabalho trouxe diversos avanços quanto a uso de hardwares e softwares.

Na fundamentação teórica, conceitos importantes puderam ser visto quanto ao funcionamento de uma microrrede de energia e os componentes que a ela compõem. Um estudo sobre padrões de comunicação proporcionou compreender como a comunicação deve ocorrer e auxiliou para criação de três placas de circuito impresso. A primeira delas de conversão de dados RS-485/RS-232 que pode ser utilizada para diversas aplicações em muitos produtos comerciais e duas placas de comunicação que permitem a conexão de um hardware RS-232 ou RS-485 a uma rede Wifi para transmissão dos dados.

Como software foi obtido o *Central Control Software* (CCS), que proporciona através de uma interface amigável o monitoramento dos dispositivos conectados salvando os dados em arquivos de texto. Como extensão foi permitido a conexão do software de monitoramento a um software externo que fez com que ampliasse o uso do software de diversas maneiras, permitindo o uso de outros softwares para o processamento dos dados e controle.

A criação de protocolos de comunicação foi realizada para permitir que um padrão fosse utilizado entre a comunicação dos dispositivos monitorados e CCS.

Os estudos de caso realizados permitiram a validação do funcionamento do software interagindo com dispositivos, fornecendo assim exemplos de aplicações futuras. Além de validar o uso do conjunto em uma aplicação real.

Para futuros trabalhos, sugere-se a atualização do software para a prevenção de problemas, adição de mais dispositivos a microrrede habilitando a conexão com o CCS, estudar e adicionar ao CCS a compatibilidade com protocolos mais utilizados na indústria e permitir que os dados não sejam mais apenas acessados através de uma rede local e que possa ser armazenado em um servidor.

REFERÊNCIAS BIBLIOGRÁFICAS

- ACROBOTIC. **Micro e minigeração distribuídas**. 2015. <<http://learn.acrobotic.com/tutorials/post/esp8266-getting-started>>. Acesso em 22 de ago de 2017.
- ANEEL. **Micro e minigeração distribuídas**. 2017. <<http://www.aneel.gov.br/geracao-distribuida>>. Acesso em 03 de jul de 2017.
- BUYAYA, R.; DASTJERDI, A. V. **Internet of Things: Principles and paradigms**. [S.l.]: Elsevier, 2016.
- ESPRESSIF. **ESP8266EX Datasheet, 5.4 Version**. Curitiba, 2017.
- FOROUZAN, B. A.; MOSHARRAF, F. **Redes de Computadores**. Porto Alegre: AMGH, 2013.
- FRANCISCATTO, R.; CRISTO, F.; PERLIN., T. **Redes de Computadores**. Frederico Westphalen : Universidade Federal de Santa Maria, Colégio Agrícola de Frederico Westphalen: Campus, 2014. ISBN 978-85-63573-46-9.
- GITHUB. **Arduino core for ESP8266 WiFi chip**. 2016. <<https://github.com/esp8266/Arduino>>. Acesso em 15 de mar de 2018.
- GITHUB. **ESP8266 WiFi library**. 2016. <<https://github.com/esp8266/Arduino/tree/master/doc/esp8266wifi>>. Acesso em 15 de mar de 2018.
- GUERRERO, J. M.; BLAABJERG, F.; ZHELEV, T.; HEMMES, K.; MONMASSON, E.; JEMEI, S.; COMECH, M. P.; GRANADINO, R.; FRAU, J. I. Distributed generation: Toward a new energy paradigm. **IEEE Industrial Electronics Magazine**, 2010. v. 4, p. 52–64, 2010 2010.
- HERZOG, A. L. **O Brasil na Onda das smart grids - Exame**. 2013. <<http://exame.abril.com.br/revista-exame/edicoes/1040/noticias/o-brasil-na-onda-das-smart-grids>>. Acesso em 03 de jul de 2017.
- IEEE. Ieee standard for floating-point arithmetic - redline. **IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - Redline**, 2008. v. 754, p. 1–82, Aug 2008.
- INSTRUMENTS, N. **Uma rápida comparação das Interfaces de Comunicação Serial RS-232, RS-422, e RS-485**. 2013. <<http://digital.ni.com/public.nsf/allkb/DE153F74C4BF3AD8862576AB006C1AAF>>. Acesso em 02 de maio de 2017.
- INSTRUMENTS, N. **Introdução ao Modbus**. 2014. <<http://www.ni.com/white-paper/7675/pt/>>. Acesso em 03 de jul de 2017.
- KODALI, R. K.; NAIKOTI, A. Ecdh based security model for iot using esp8266. In: **2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICT)**. [S.l.: s.n.], 2016. p. 629–633.

KRUZ, C. d. H. V. **Tutorial ESP8266 - Primeiros passos ESP12 + IDE Arduíno**. 2017. <<https://www.clubedohardware.com.br/forums/topic/1206440-tutorial-esp8266-primeiros-passos-esp12-ide-arduino/>>. Acesso em 15 de mar de 2018.

MARAIS, H. **RS-485/RS-422 Circuit Implementation Guide - Analog Devices**. 2008. <<http://www.analog.com/media/en/technical-documentation/application-notes/AN-960.pdf>>. Acesso em 03 de jul de 2017.

MODBUS. **MODBUS over Serial Line - Specification and Implementation Guide**. 2006. <http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf>. Acesso em 03 de jul de 2017.

MODICON INC., I. A. S. **Modicon Modbus Protocol Reference Guide**. [S.l.], 1996.

SMARTWORX, B. **Whats is RS-232 anyway**. 2014. <<http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/What-Is-RS-232/What-Is-RS-232.pdf>>. Acesso em 02 de maio de 2018.

TRETER, M. E.; PIETTA, L. P.; XAVIER, P.; MICHELS, L. Data acquisition and cloud storage system applied photovoltaic systems. In: **2015 IEEE 13th Brazilian Power Electronics Conference and 1st Southern Power Electronics Conference (COBEP/SPEC)**. [S.l.: s.n.], 2015. p. 1–6.

TSAGARIS, A.; TRIANTAFYLLIDIS, D. G. Data monitoring system for supervising the performance assessment of a photovoltaic park. In: **2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI)**. [S.l.: s.n.], 2012. p. 385–389.

VANDOORN, T. L.; VÁSQUEZ, J. C.; KOONING, J. d.; GUERRERO, J. M.; VANDEVELDE, L. Microgrids - hierarchical control and an overview of the control and reserve management strategies. **IEEE industrial electronics magazine**, 2013. p. 42 –53, 2013 2013.

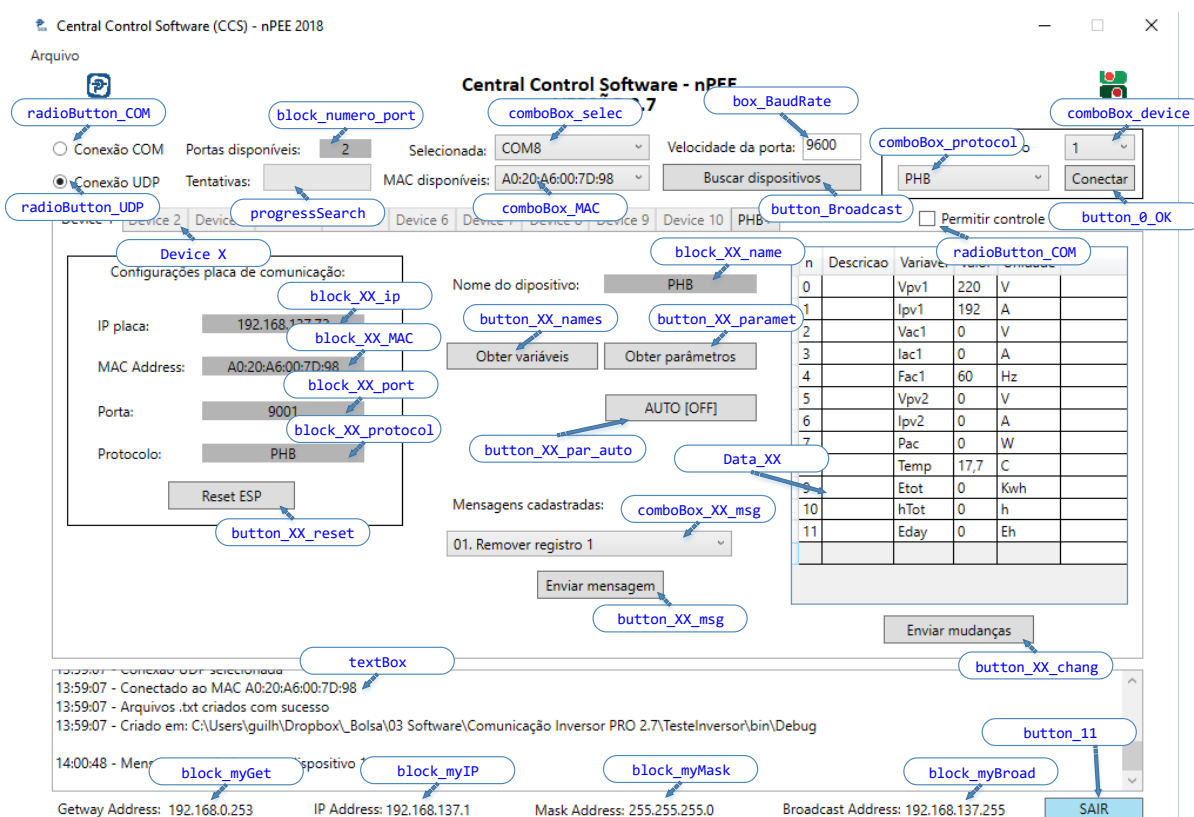
VASQUEZ, J. C.; GUERRERO, J. M.; MIRET, J.; CASTILLA, M.; VICUNA, L. G. de. Hierarchical control of intelligent microgrids. **IEEE Industrial Electronics Magazine**, 2010. v. 4, p. 23–29, 2010 2010.

YU, X.; CECATI, C.; DILLON, T.; SIMOES, M. The new frontier of smart grids. **Industrial Electronics Magazine, IEEE**, 2011. v. 5, p. 49–63, 2011 2011.

APÊNDICE A – ESTRUTURA DOS OBJETOS

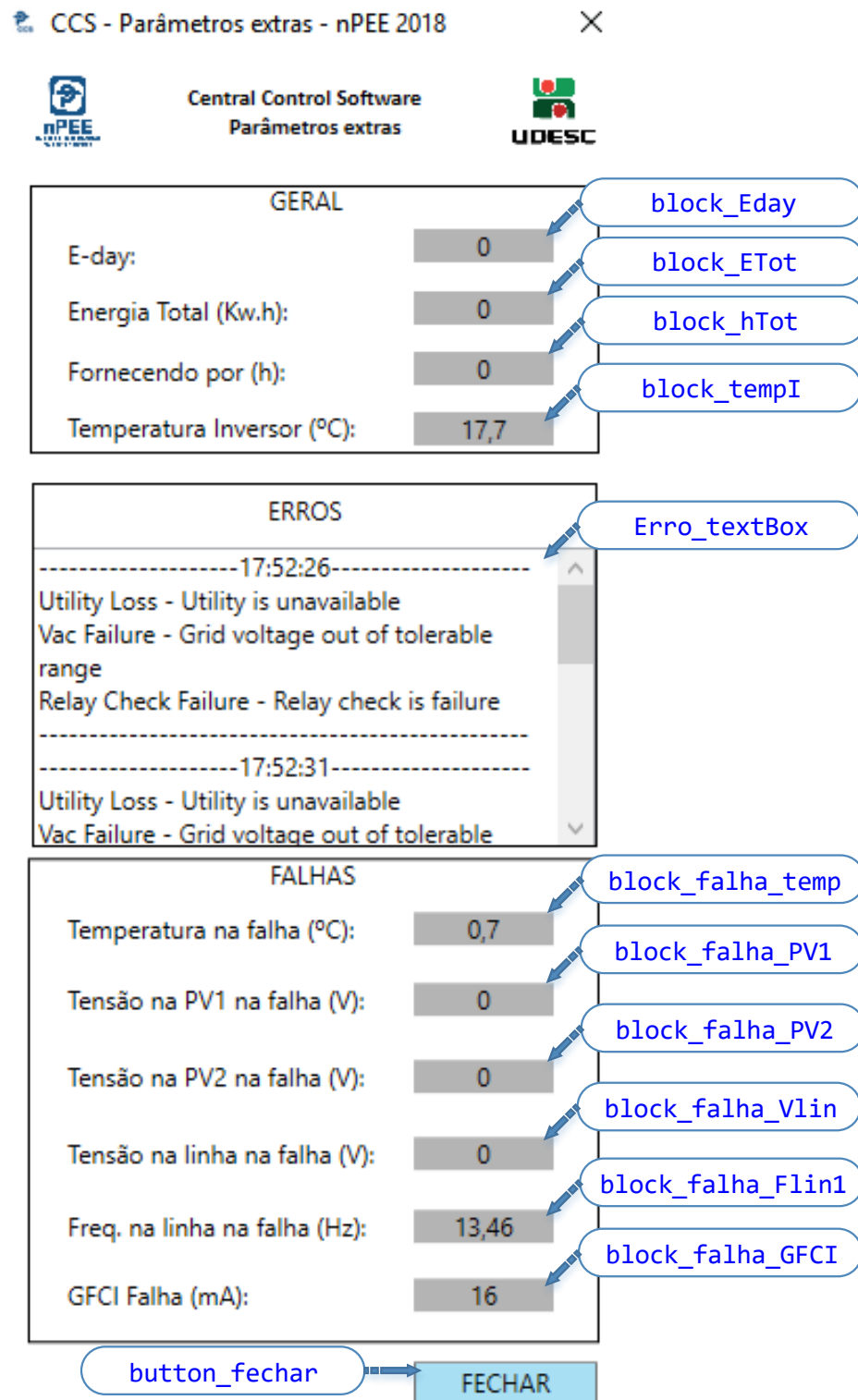
As Fig. A.1, Fig. A.2 e Fig. A.3 mostram o nome de cada objeto utilizado nas janelas do software criado e em suas abas. Para criar um padrão os objetos foram nomeados de acordo com seu tipo seguido de sua principal função. Como os elementos utilizados por dispositivos diferentes se repetem entre as abas, na figura A.1 foi re-nomeado o nome destes com o valor “XX”, que na programação recebem o número do dispositivo.

Figura A.1 – Nomes dos objetos utilizados na janela principal na aba de dispositivos para um dispositivo genérico de número “XX”.



Fonte: Elaborado pelo autor (2018).

Figura A.3 – Nomes dos objetos utilizados na janela de parâmetros extras

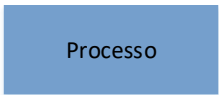


Fonte: Elaborado pelo autor (2018).

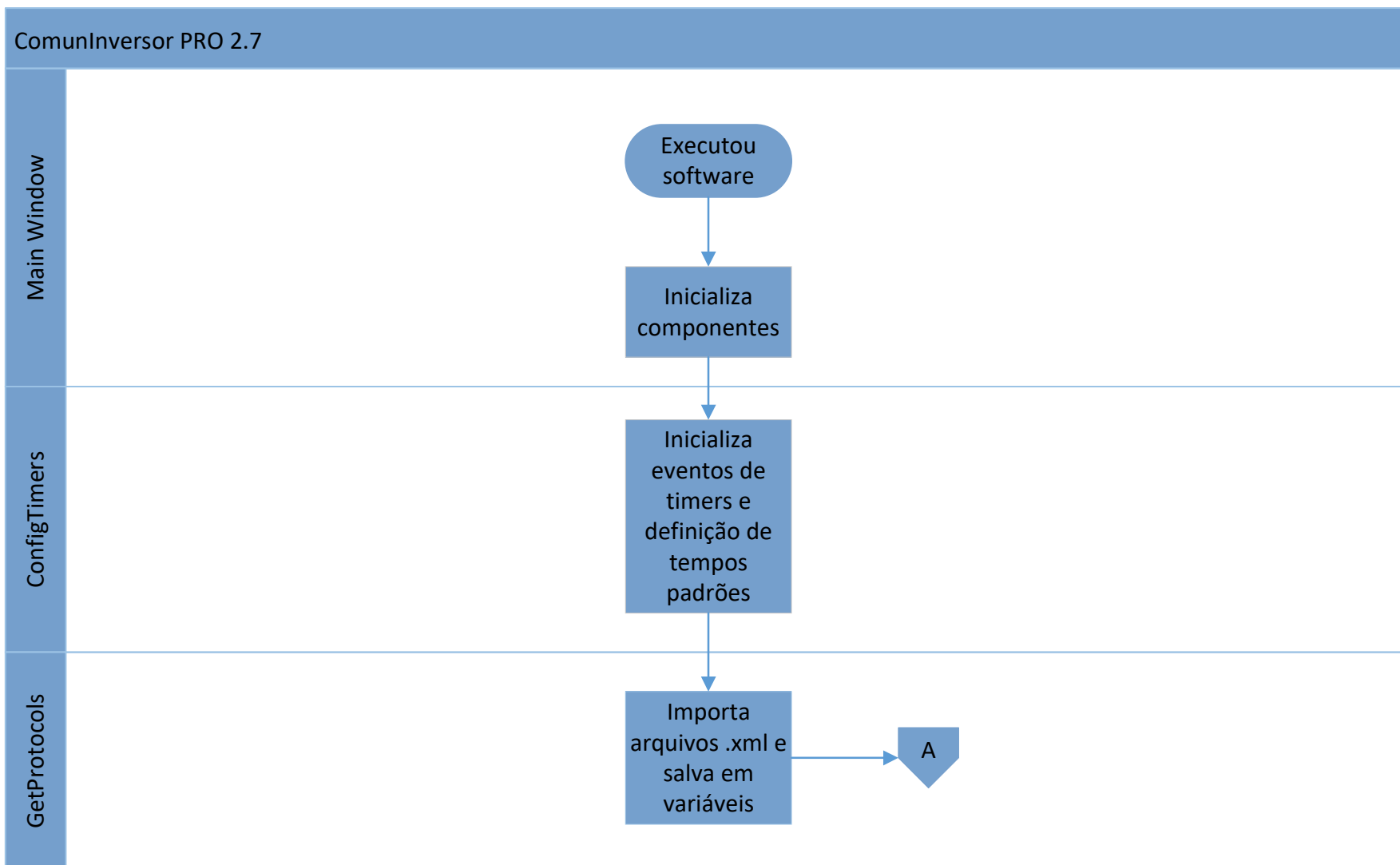
APÊNDICE B – FLUXOGRAMA DE FUNCIONAMENTO DO CCS VERSÃO PRO 2.7

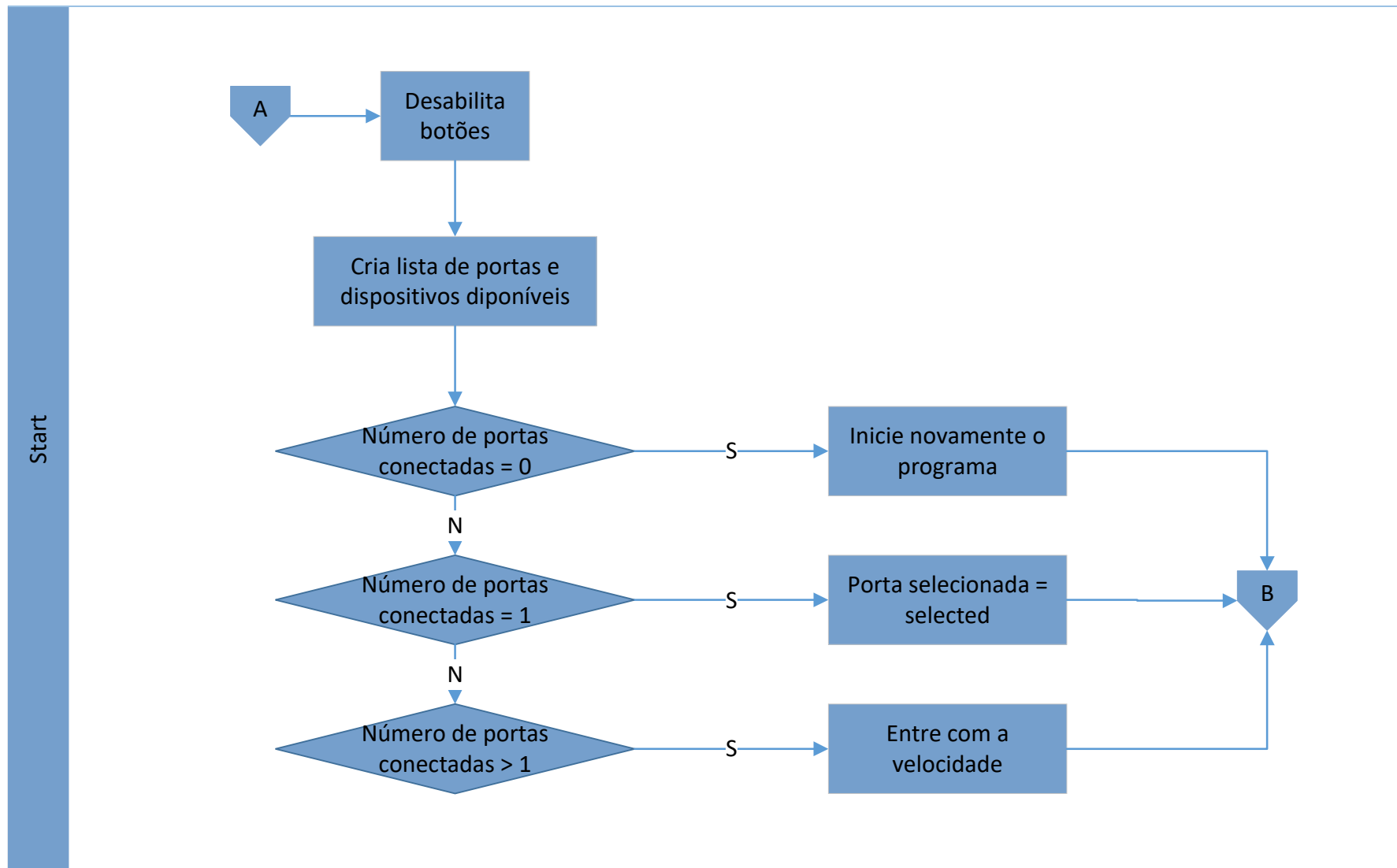
Este apêndice traz o fluxograma de funcionamento do software CCS versão 2.7 criado em linguagem C# via interface de programação *Visual Studio* 2017. O fluxograma segue a definição de objetos mostrado na Fig. B.1.

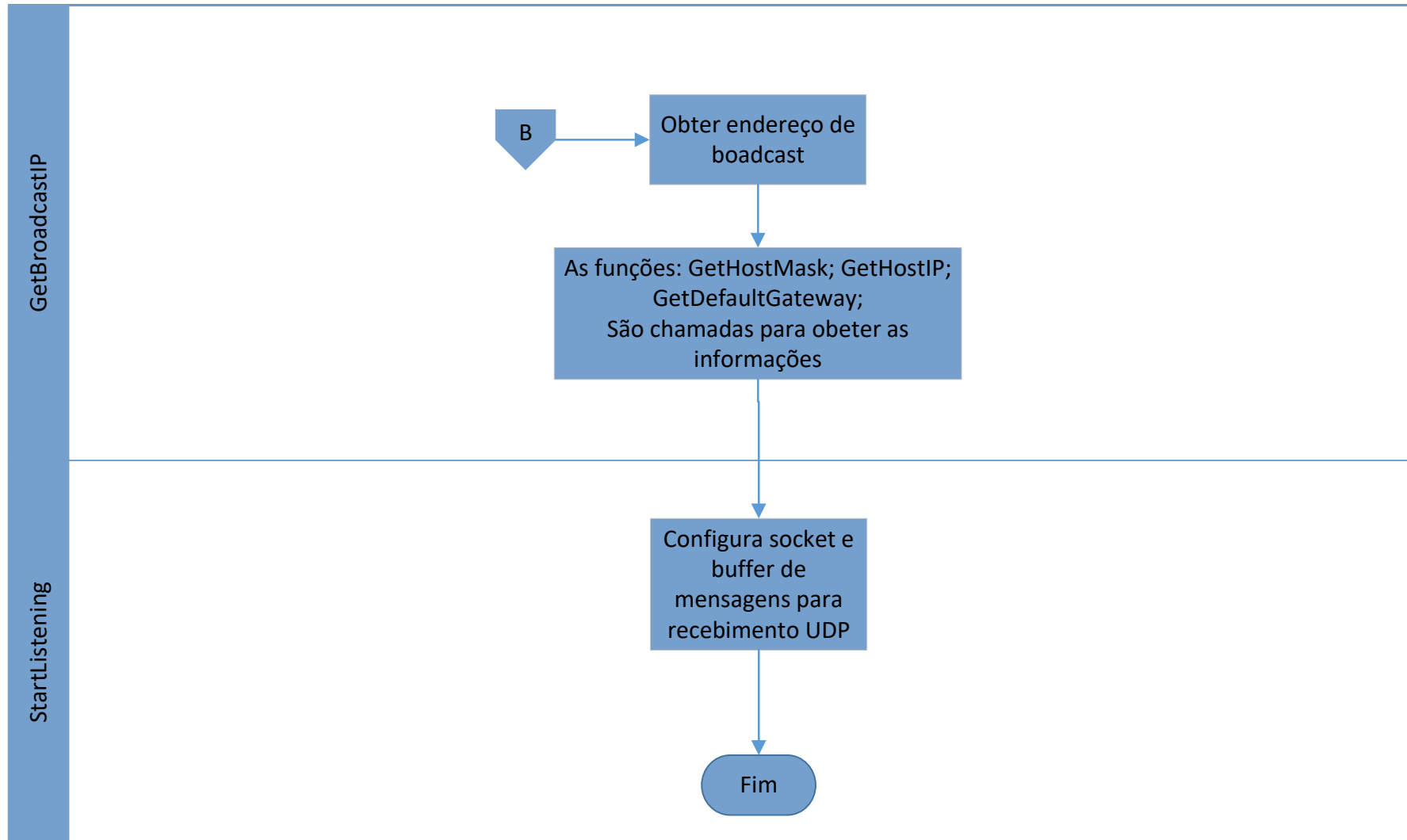
Figura B.1 – Descrição de objetos utilizados no fluxograma apresentado

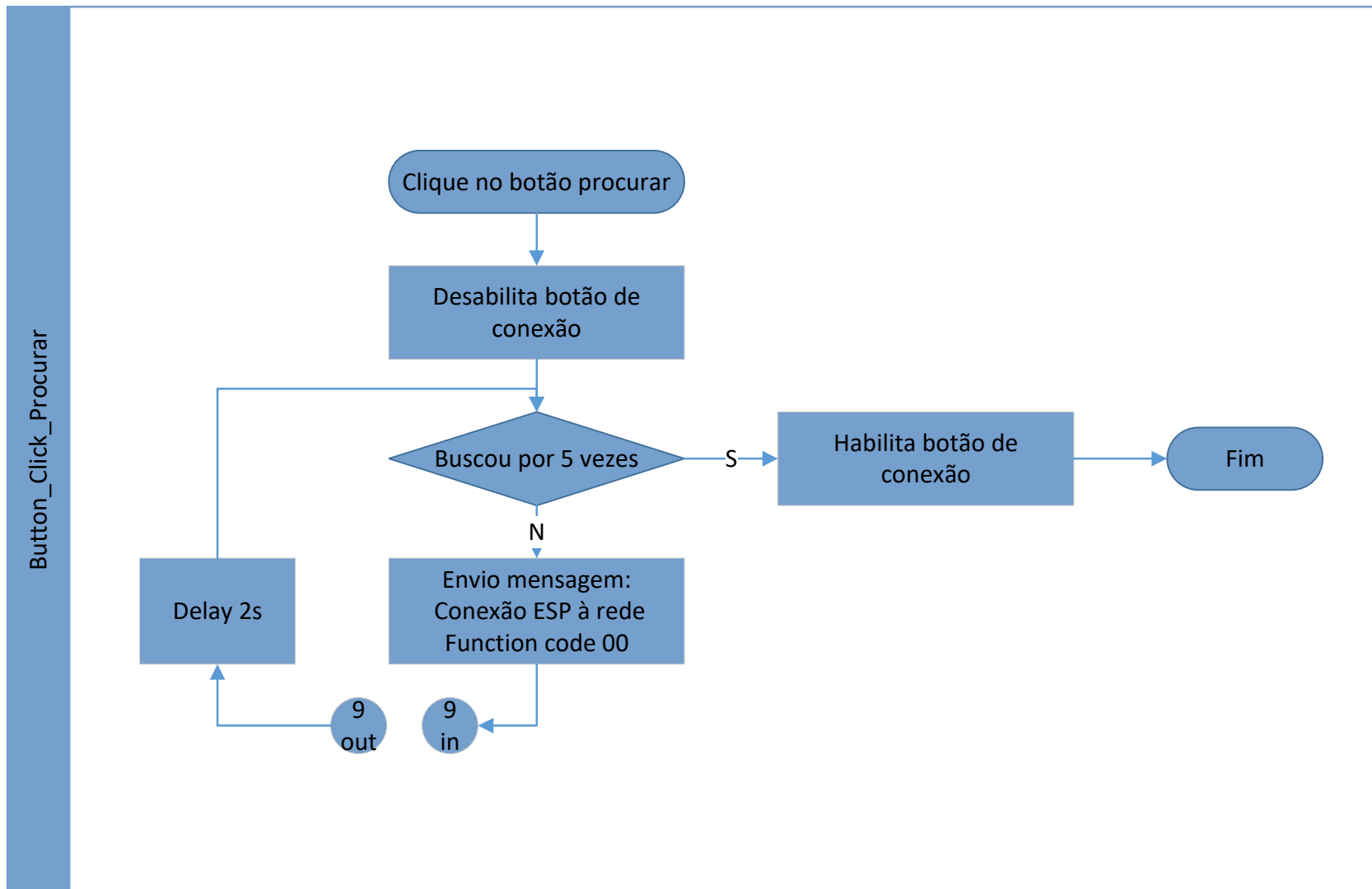
Forma	Descrição
	Início de fluxograma
	Decisão para tomada de ação
	Execução de ação
	Entrada em função de número 'X'
	Saída da função de número 'X'
	Continuação fora da página Z
	Fim de fluxograma

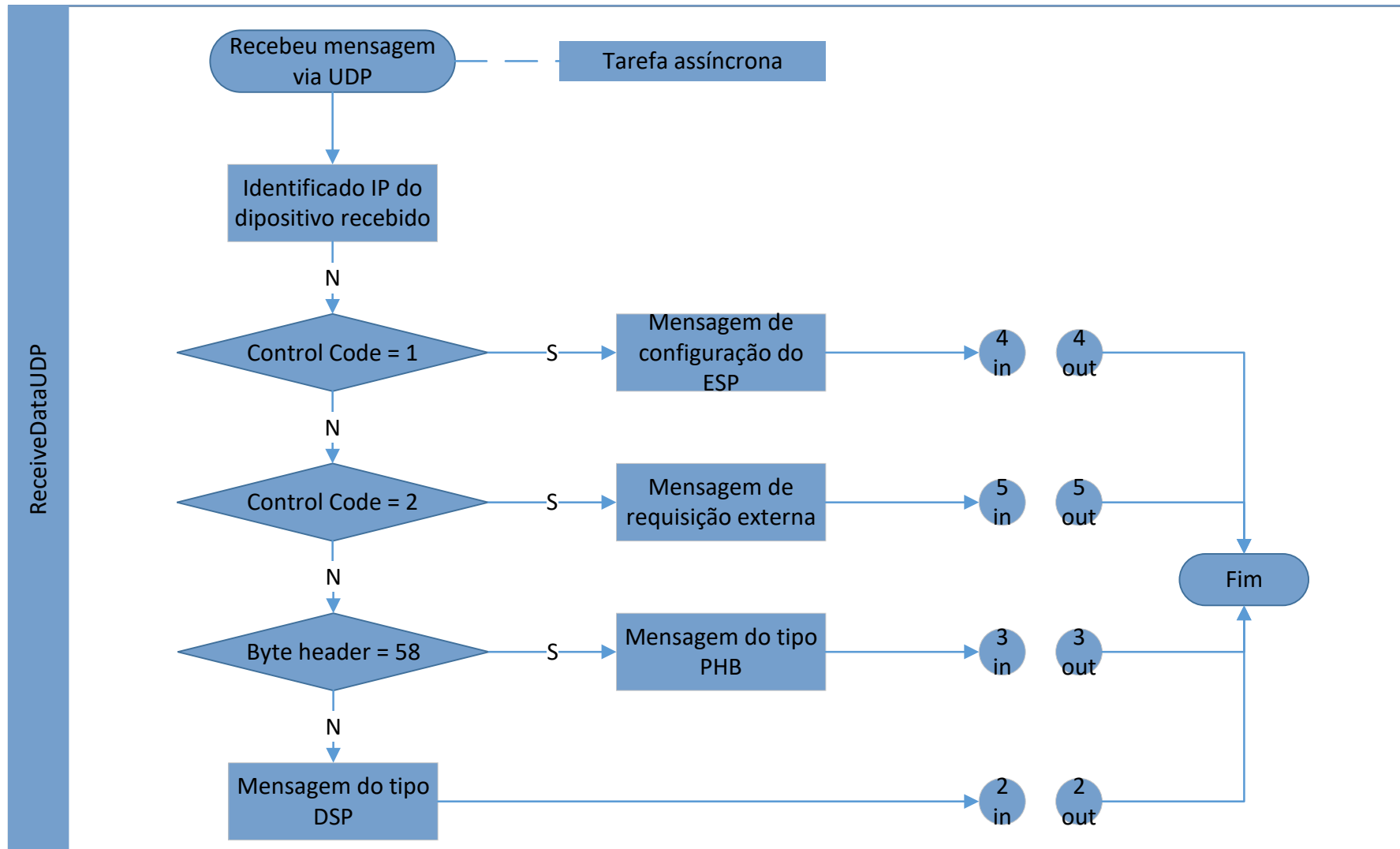
Fonte: Elaborado pelo autor (2018).

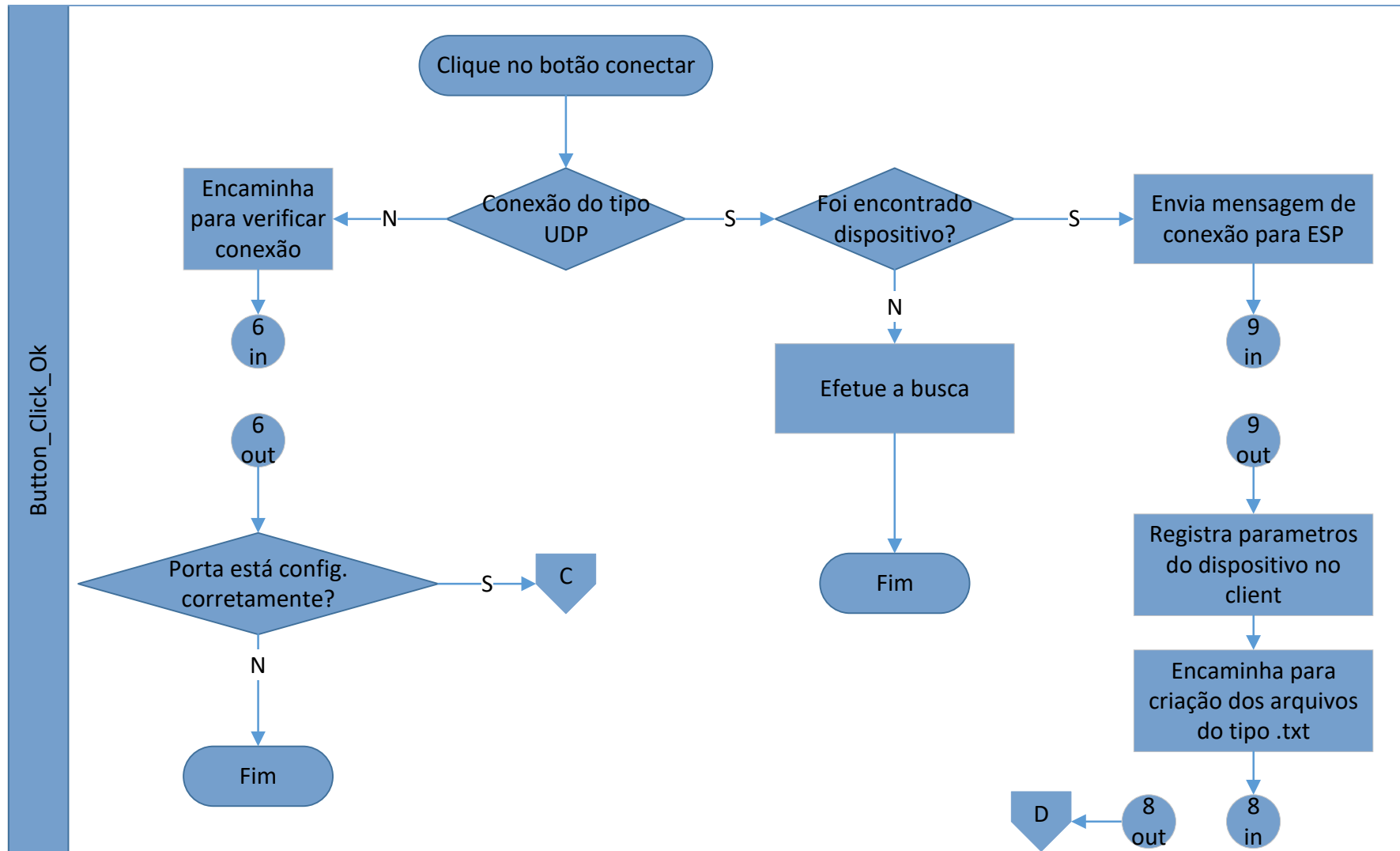


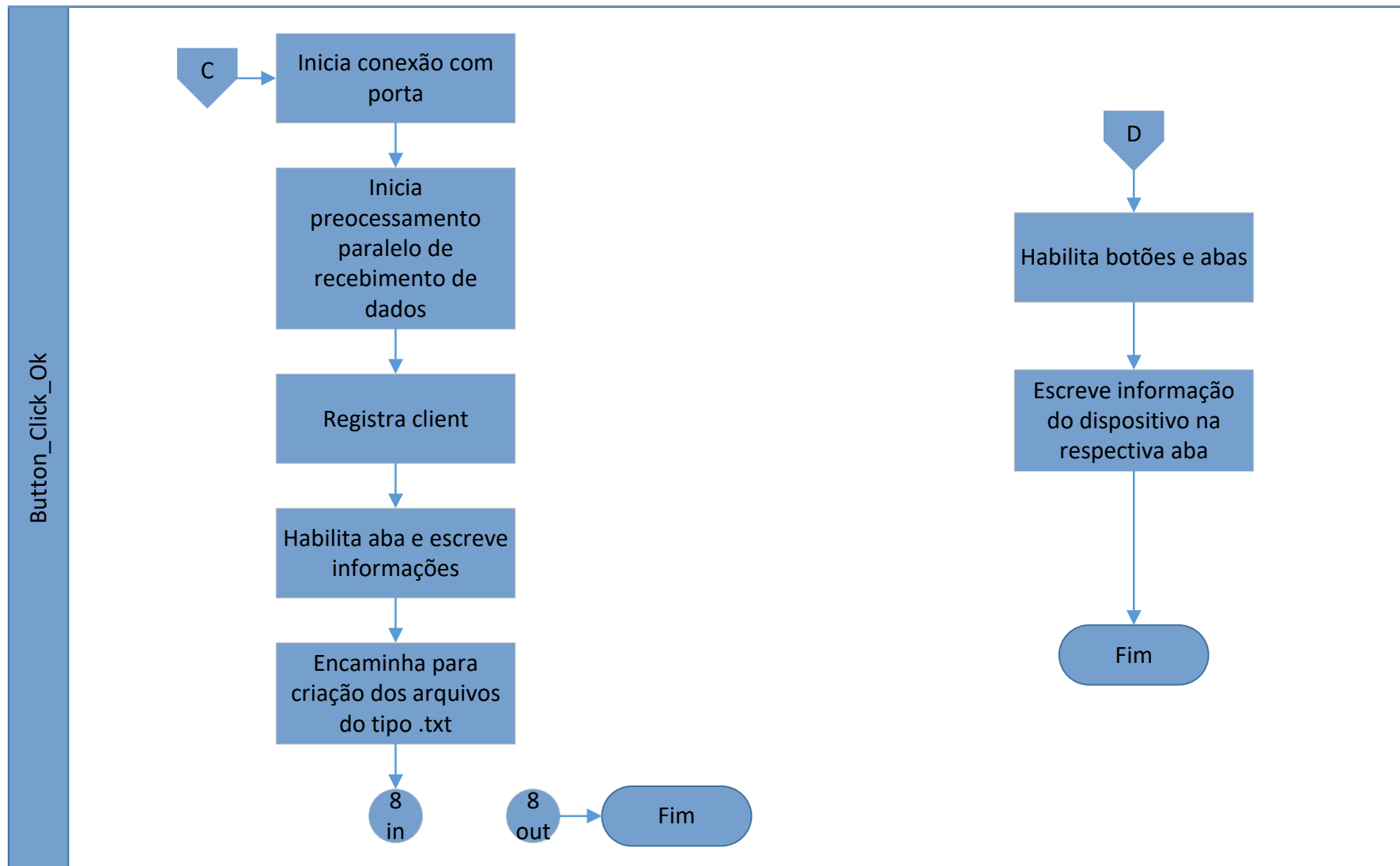


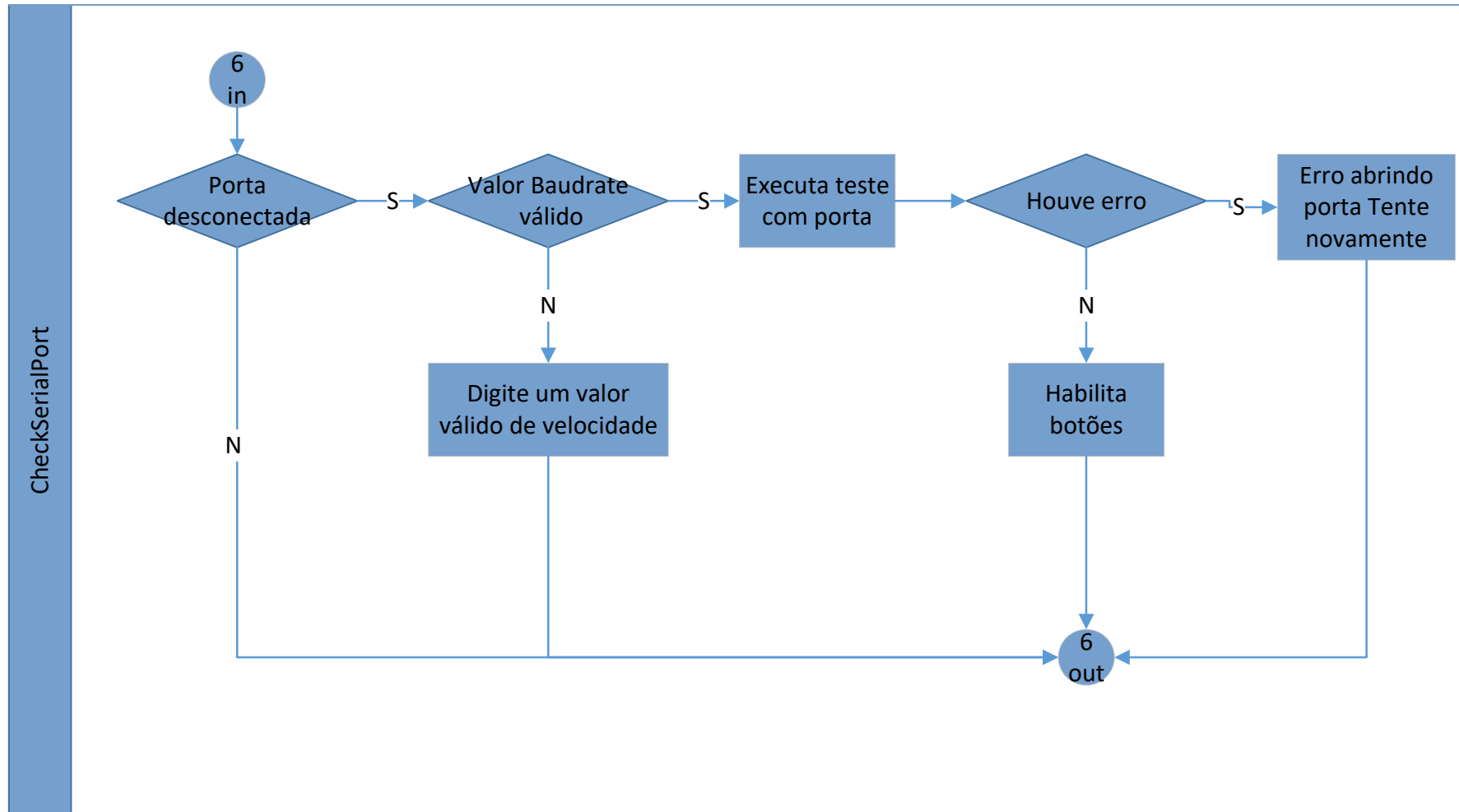


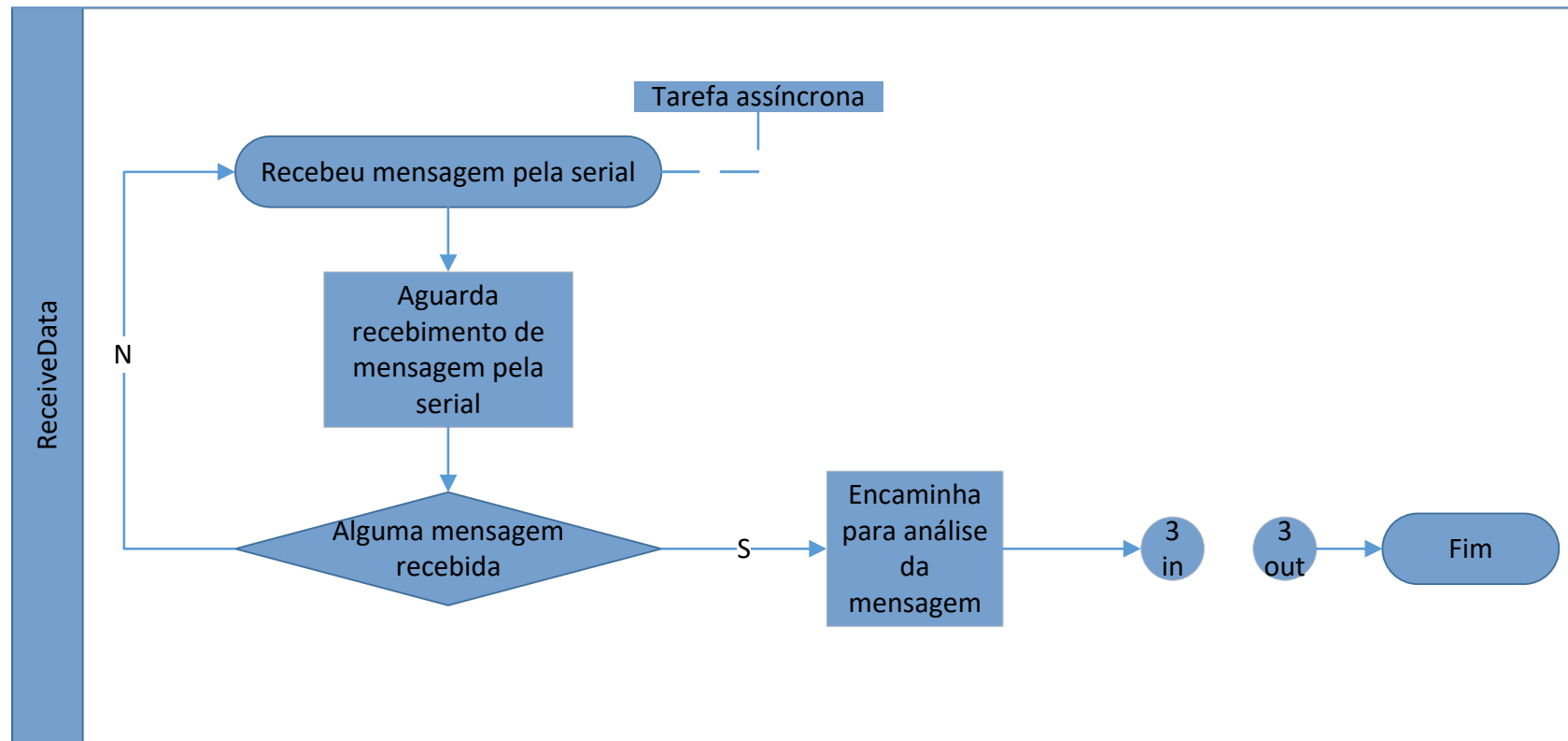


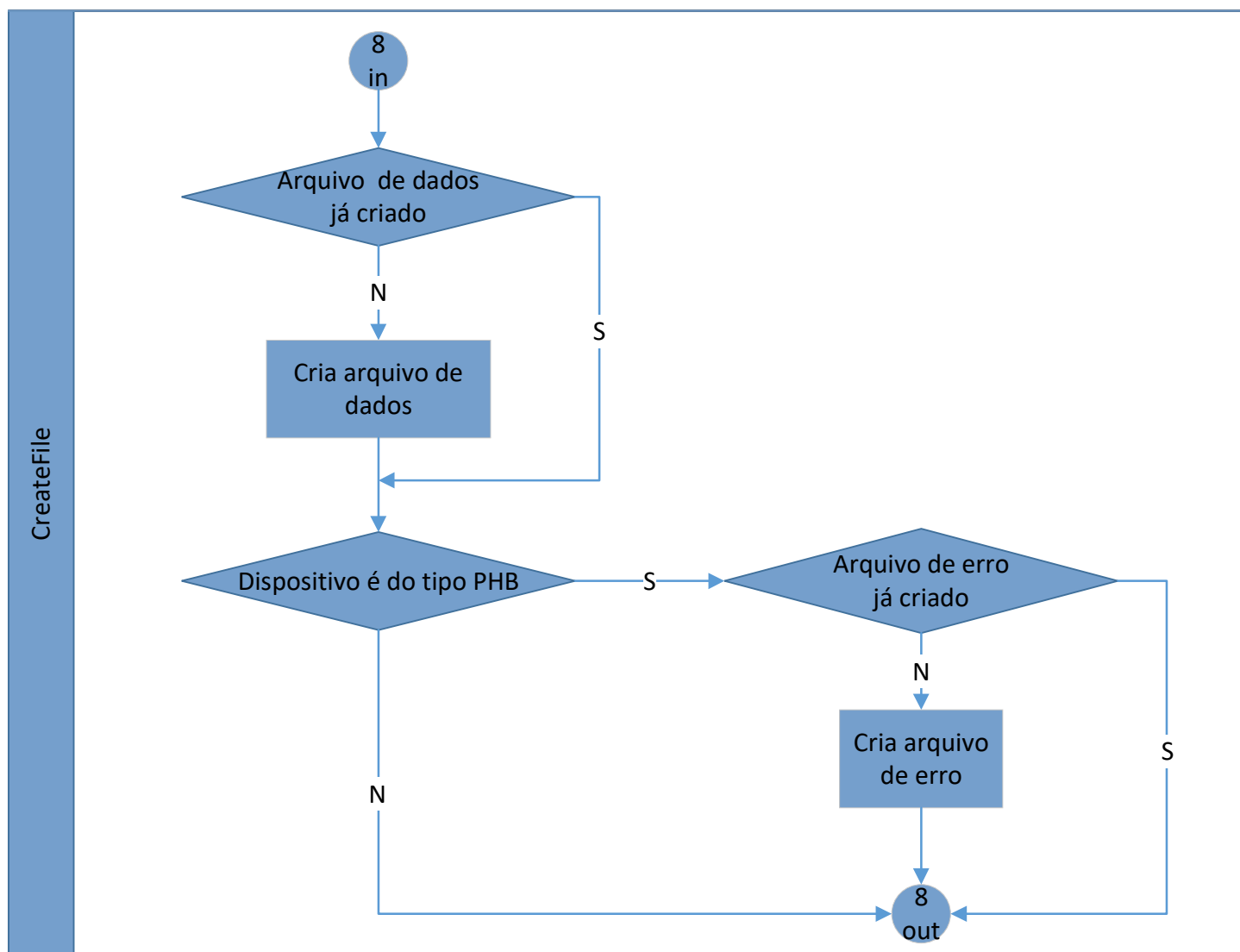


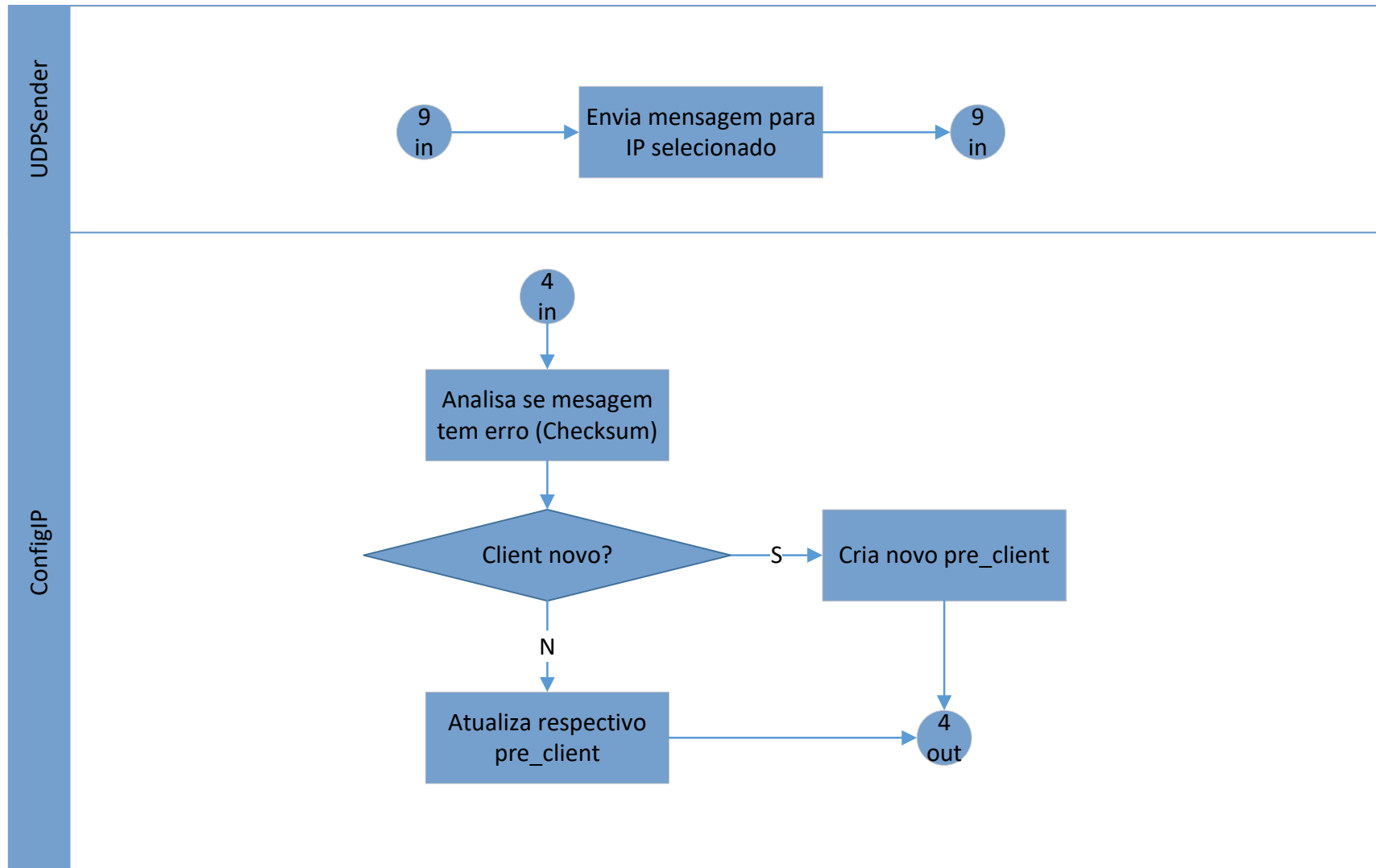


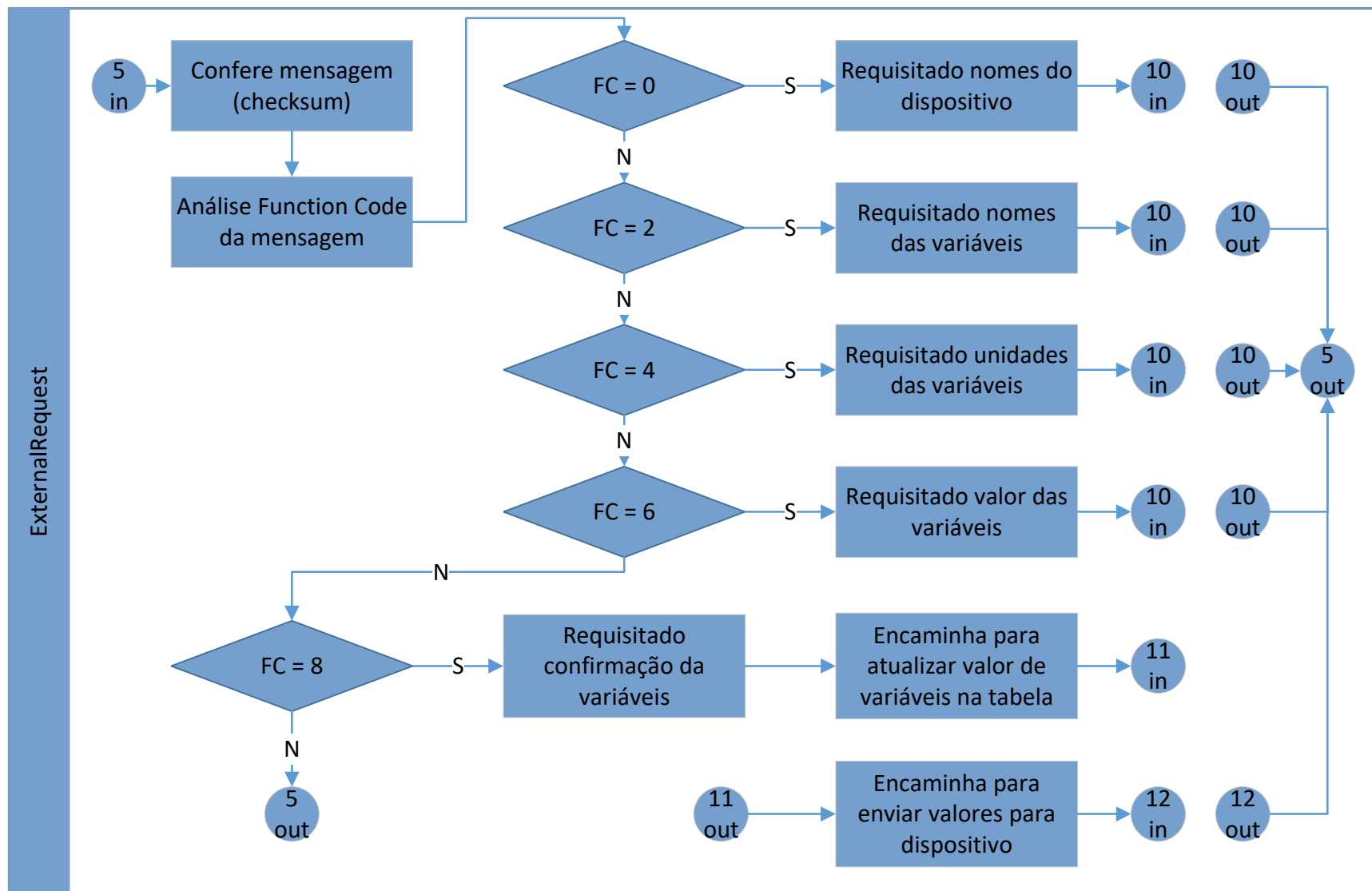


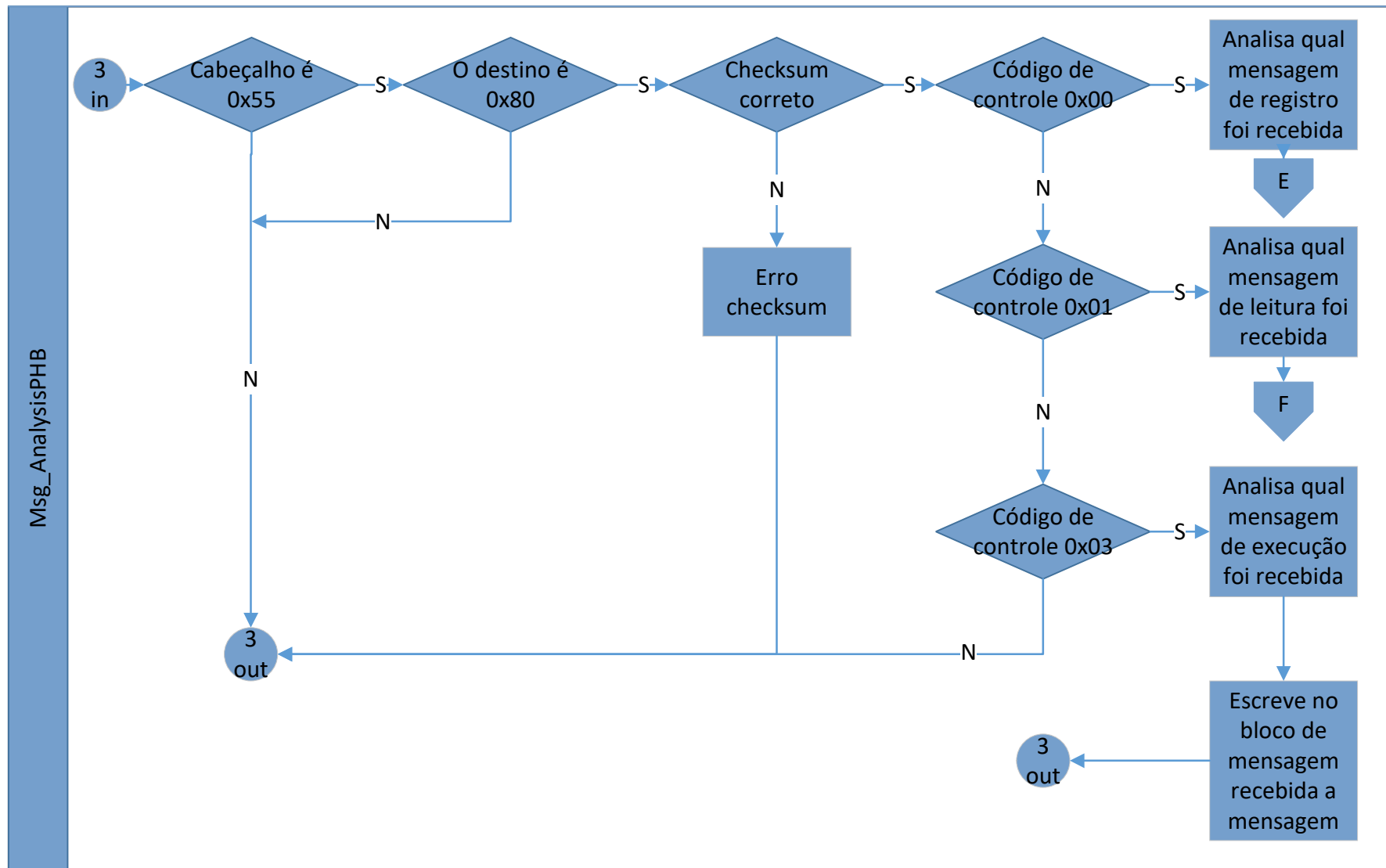




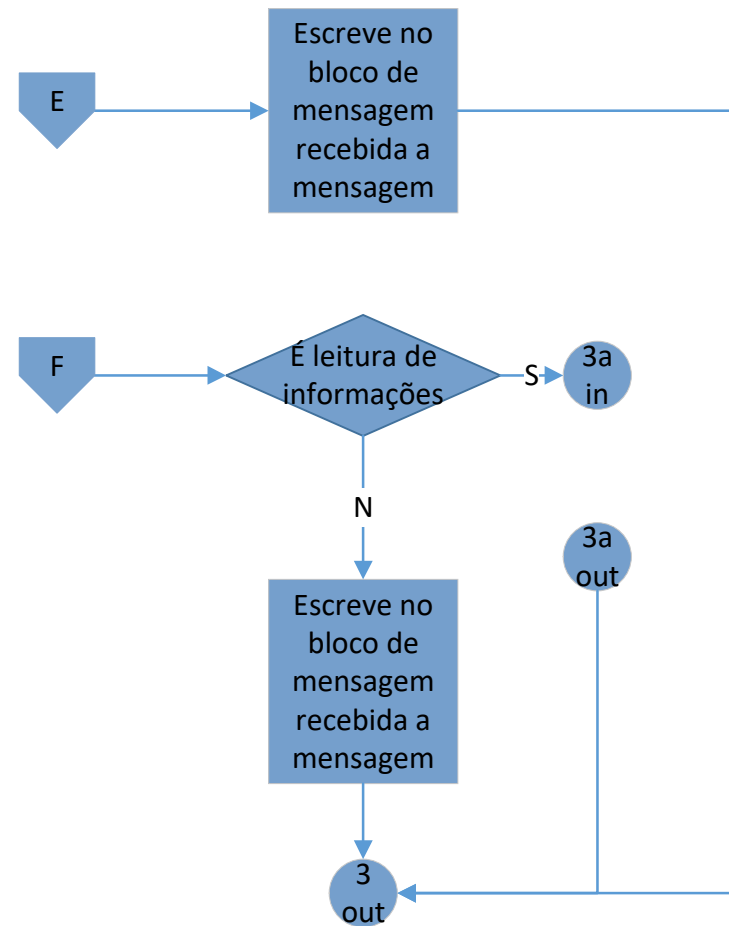


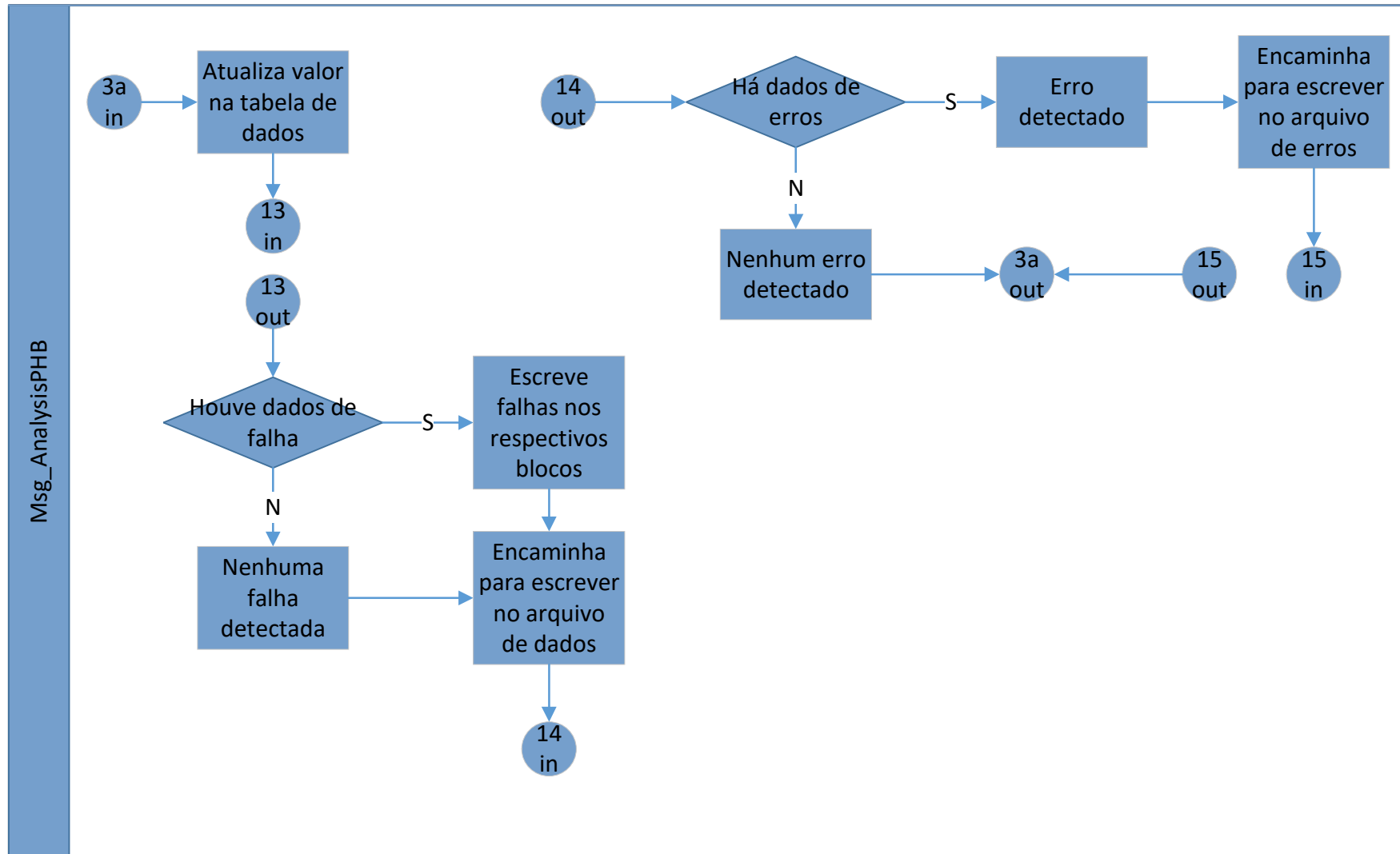


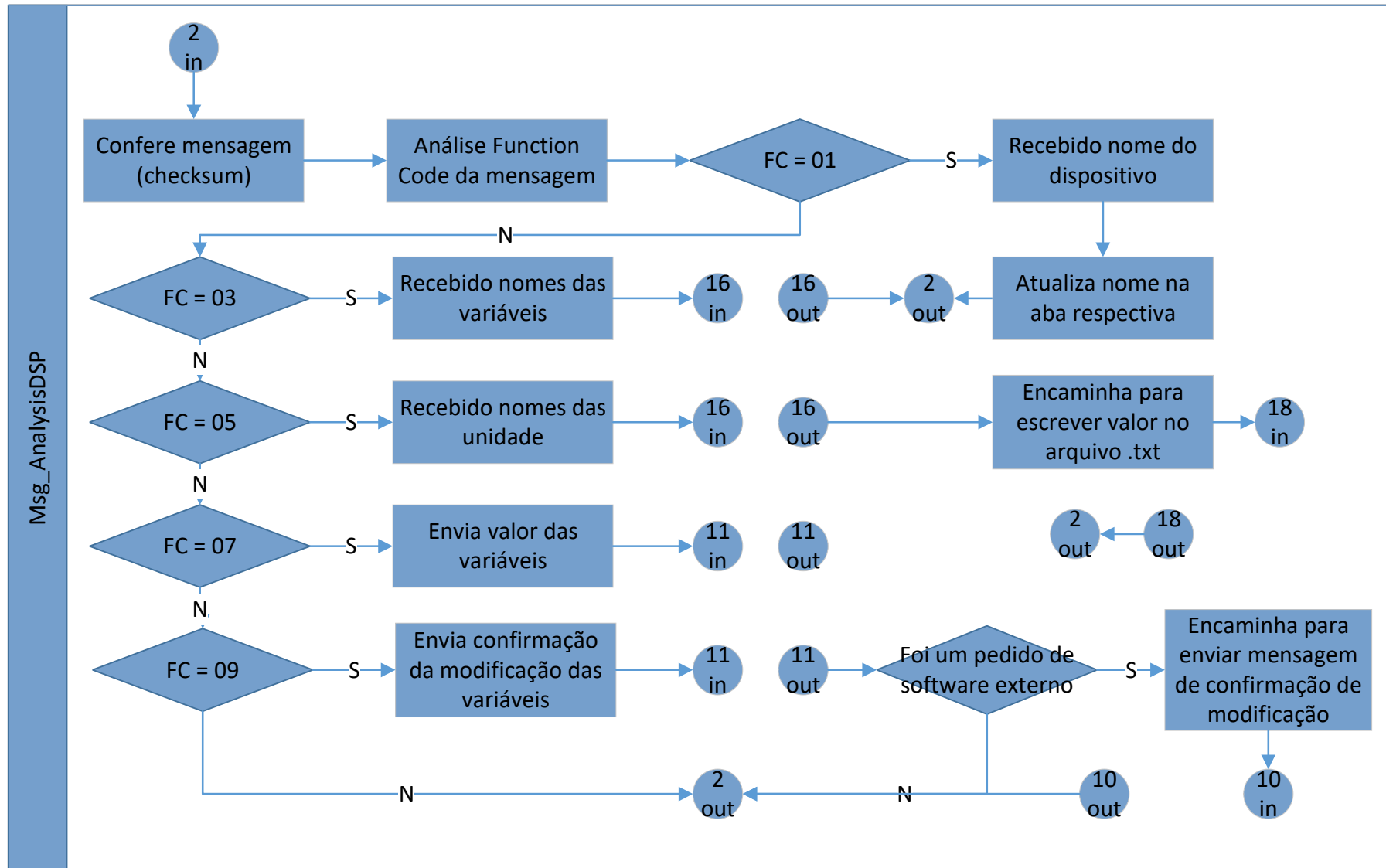


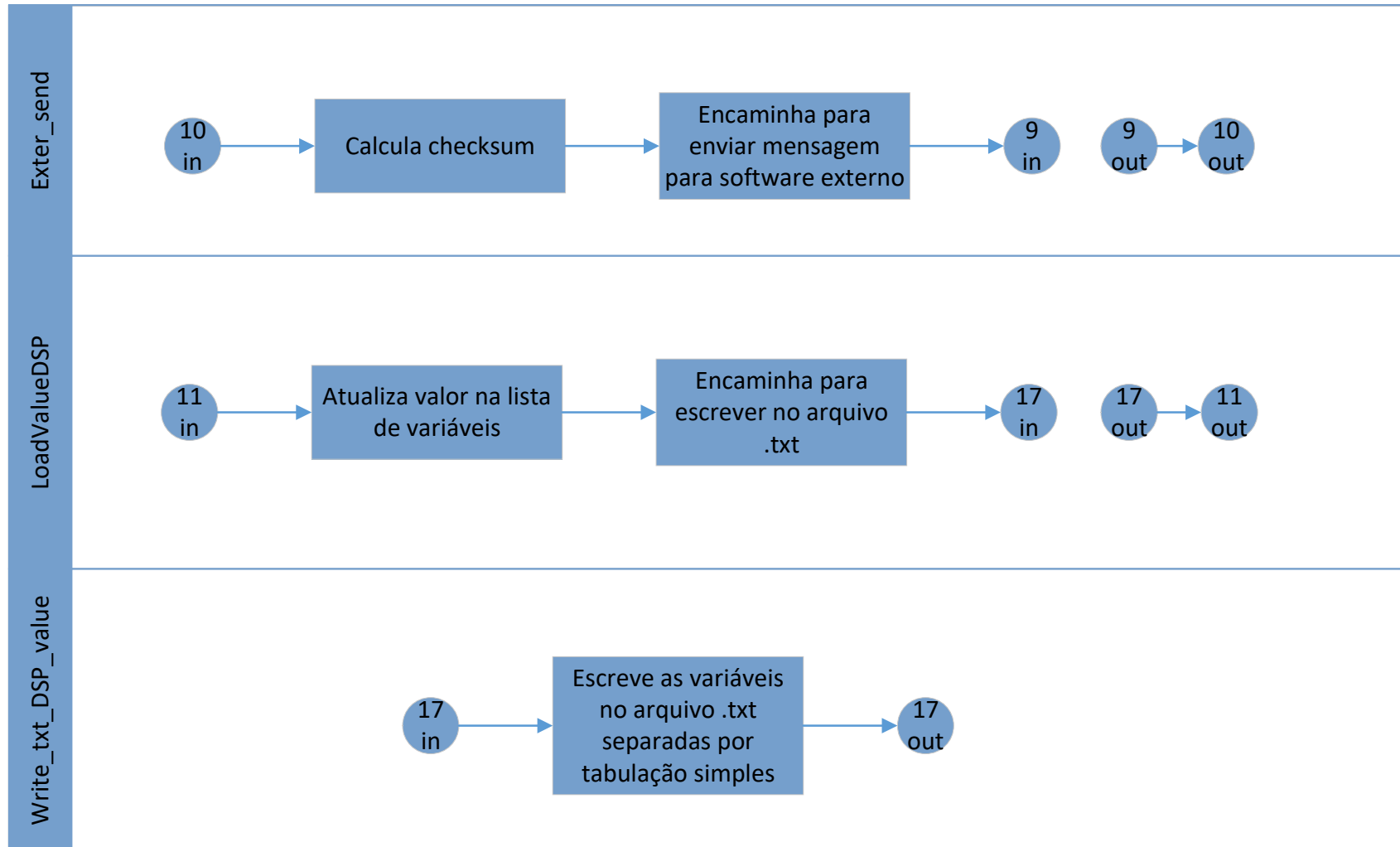


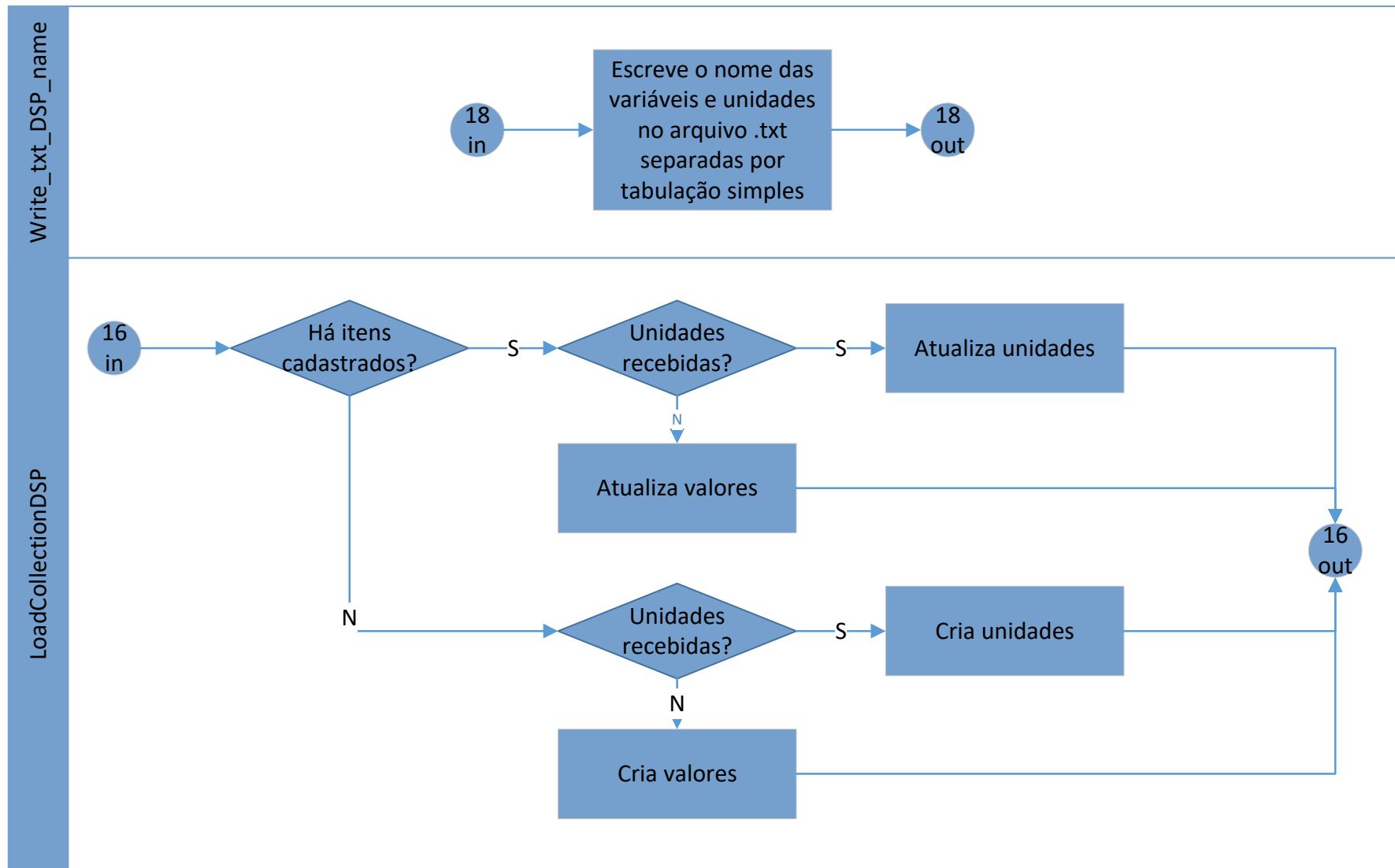
Msg_AnalysisPHB

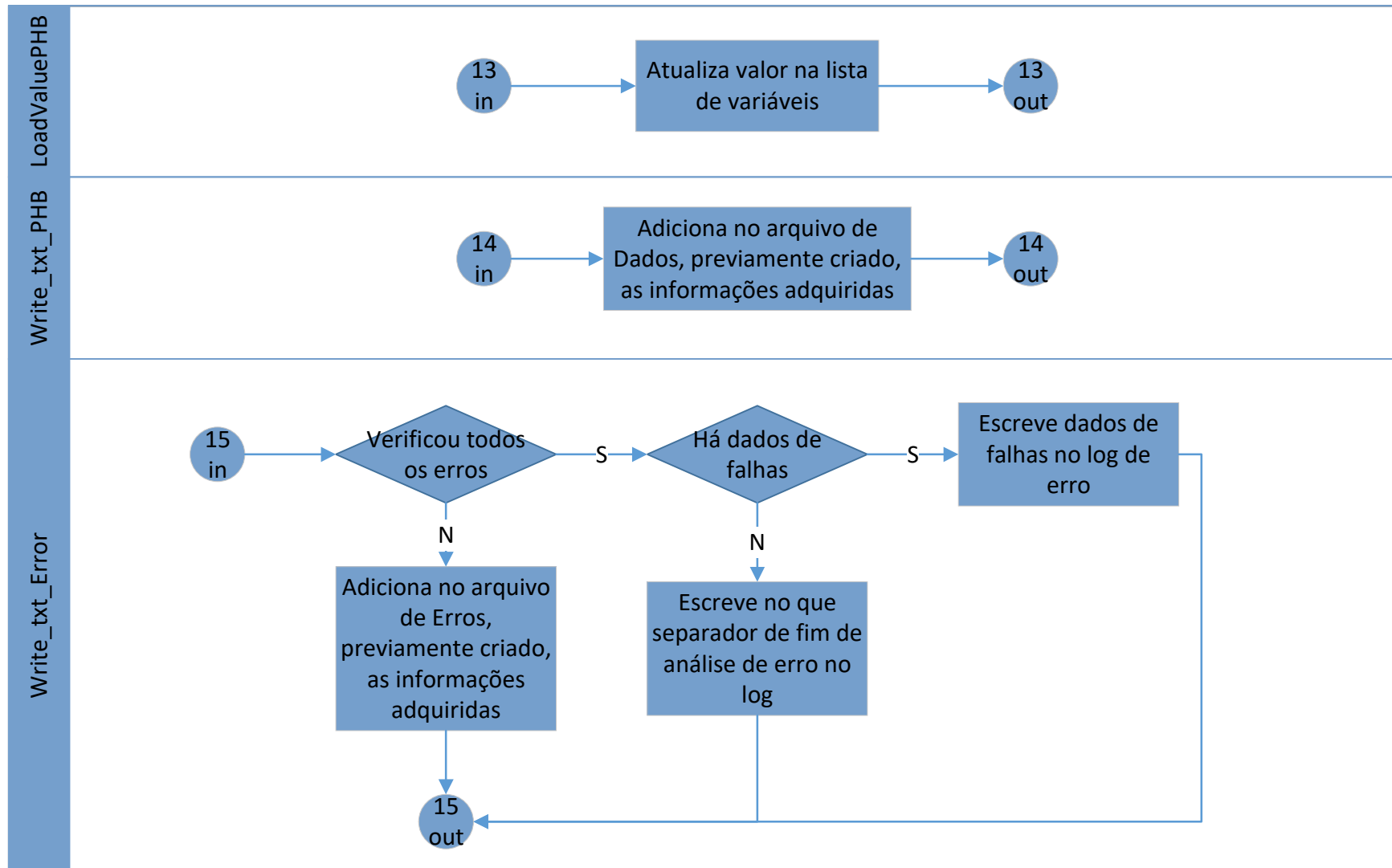




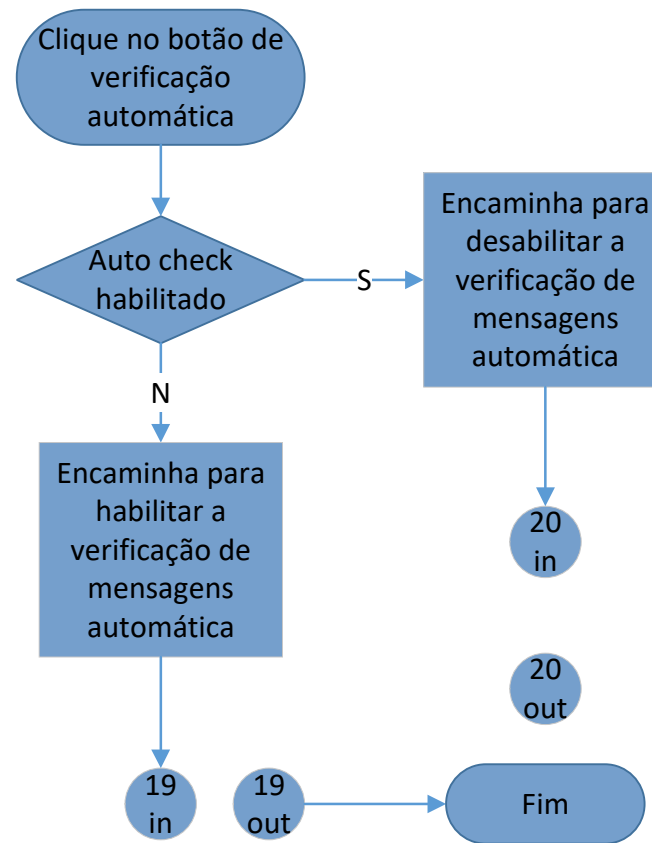


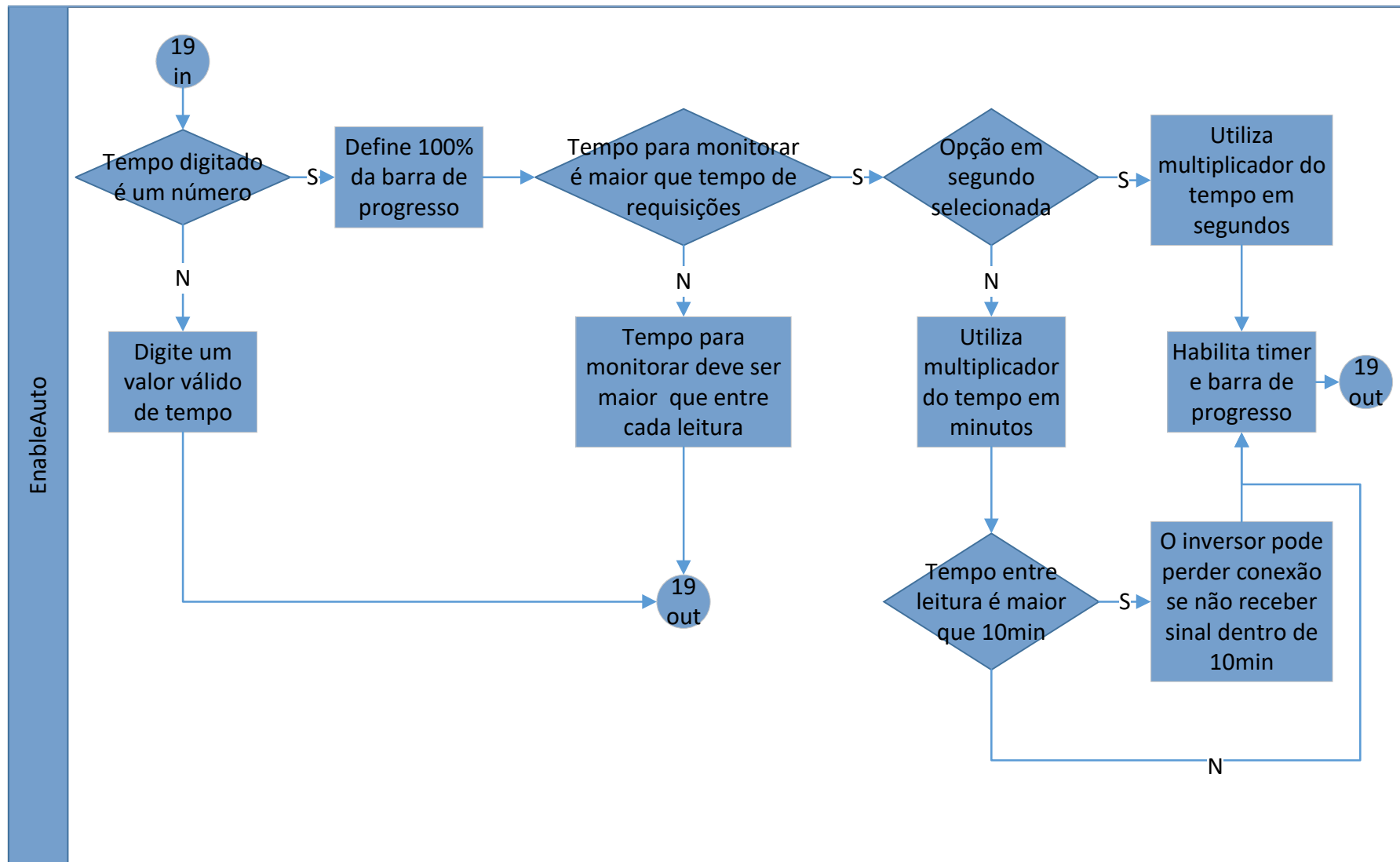


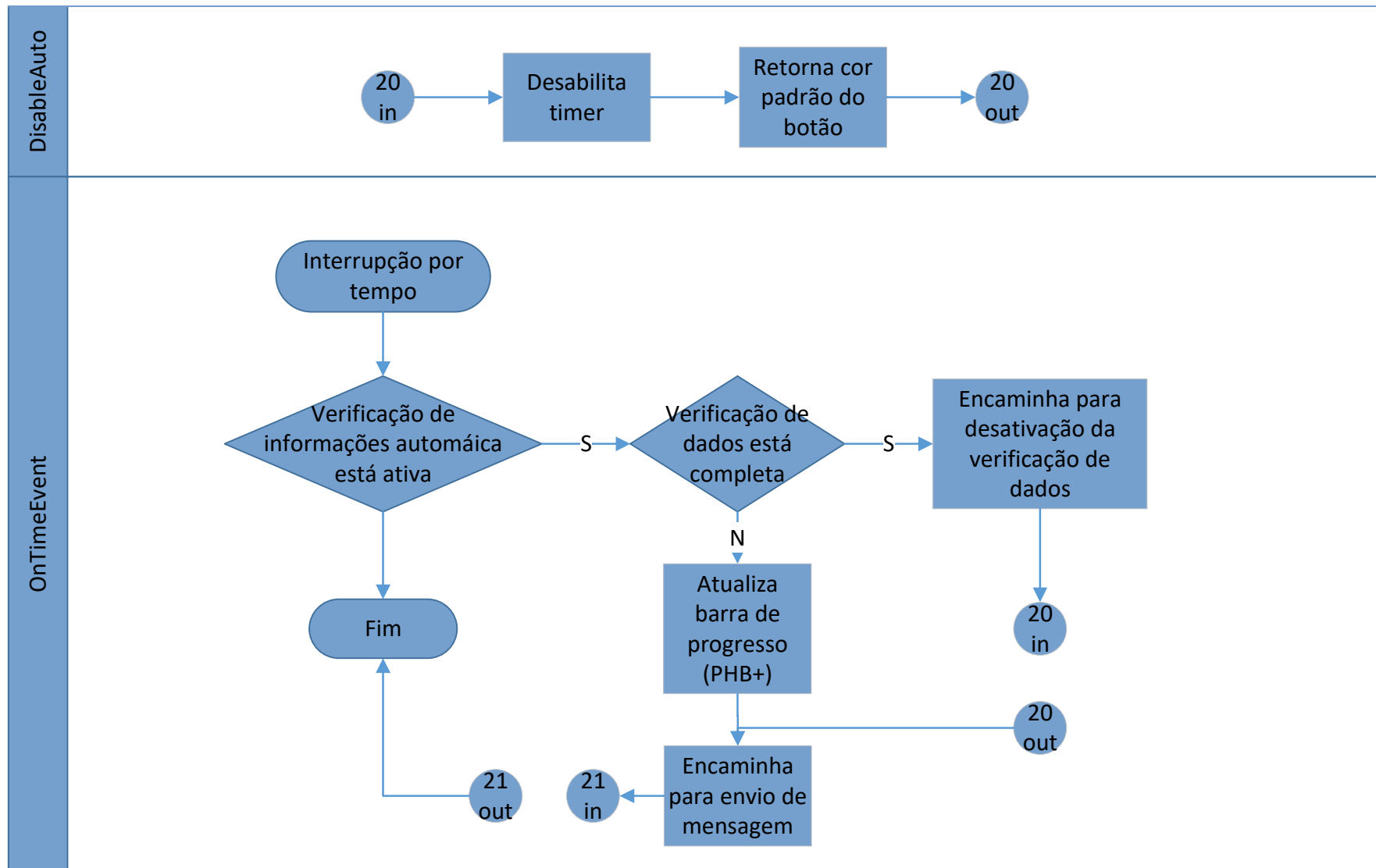


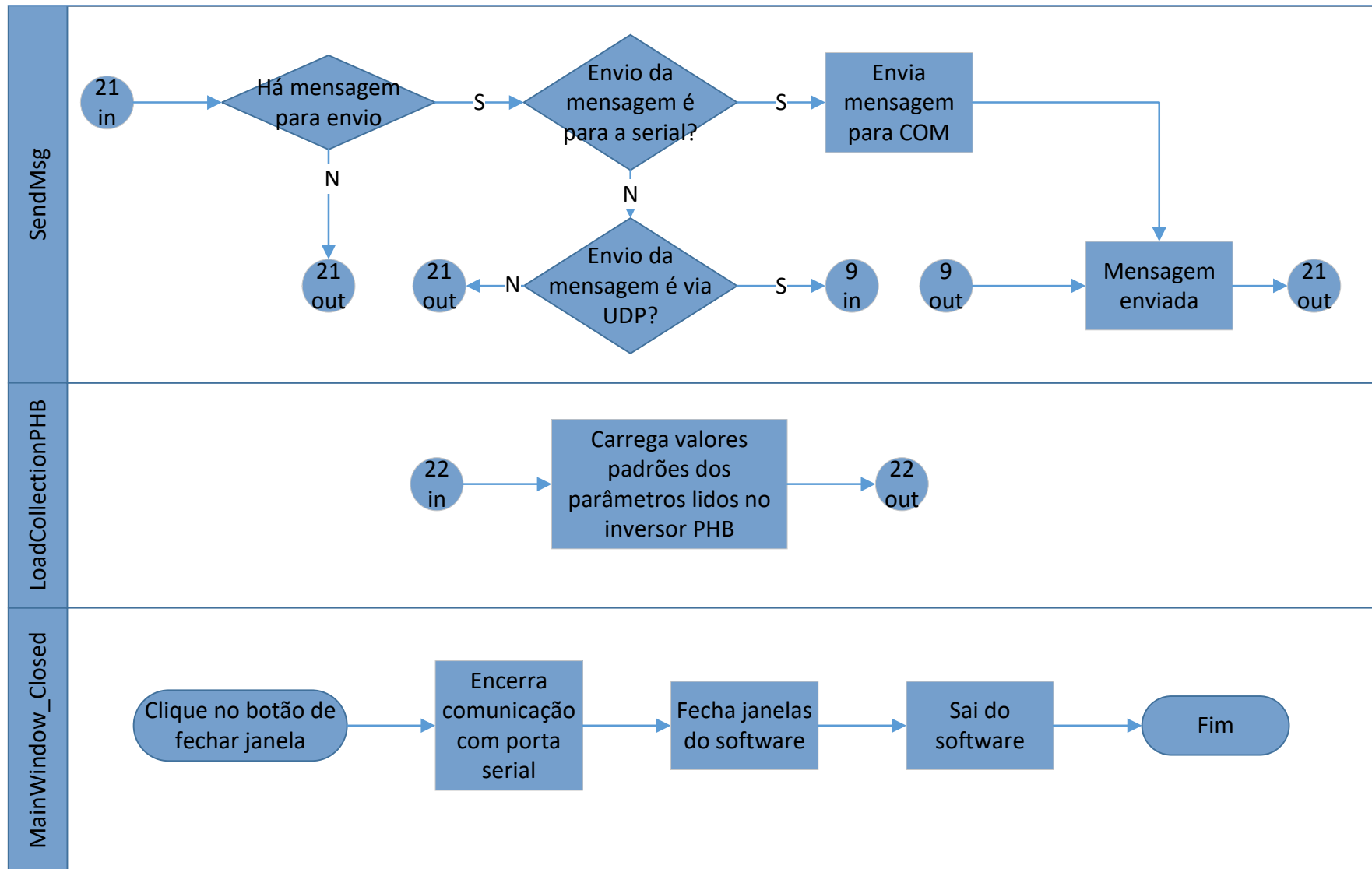


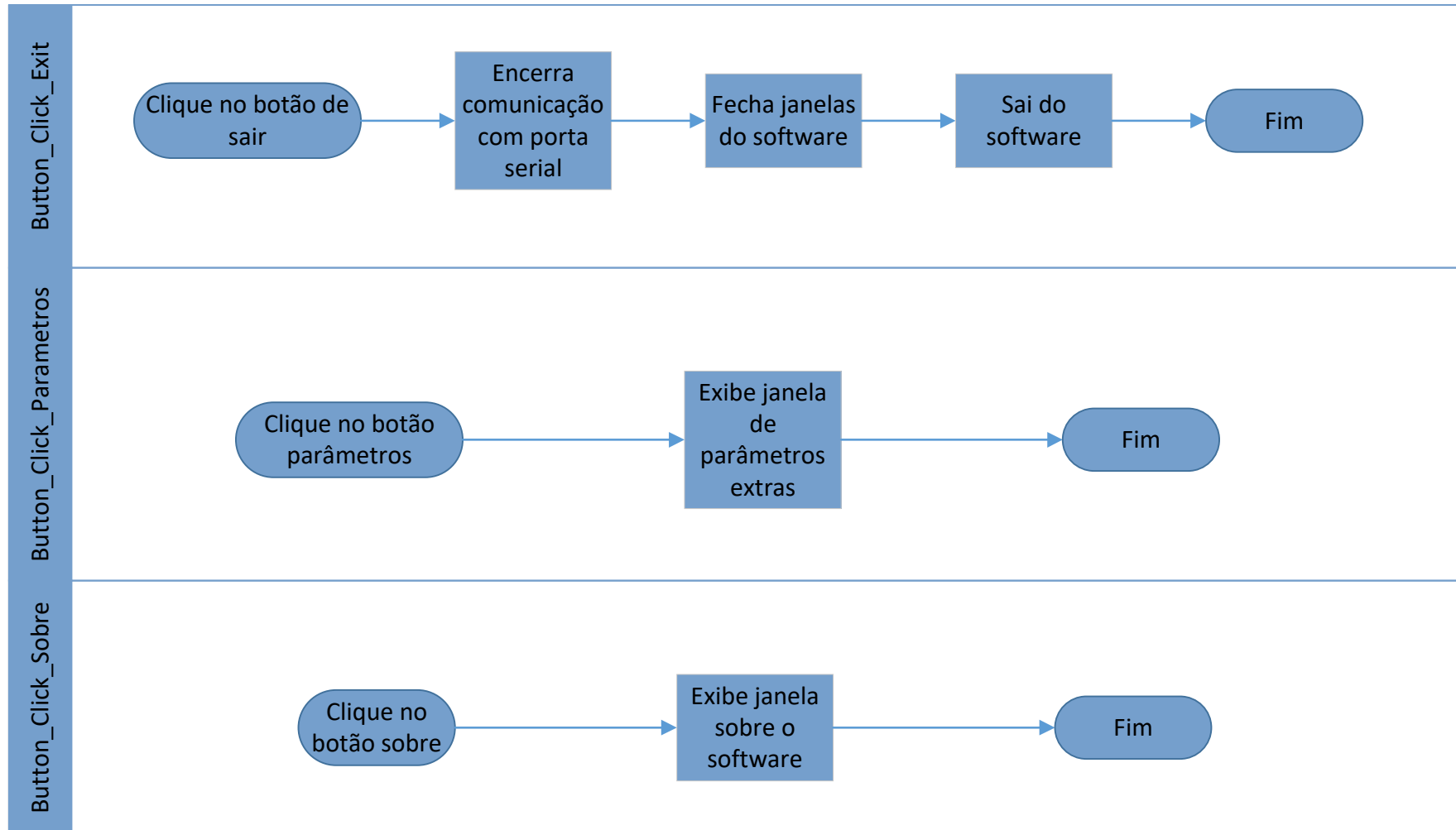
Button_Click_Auto

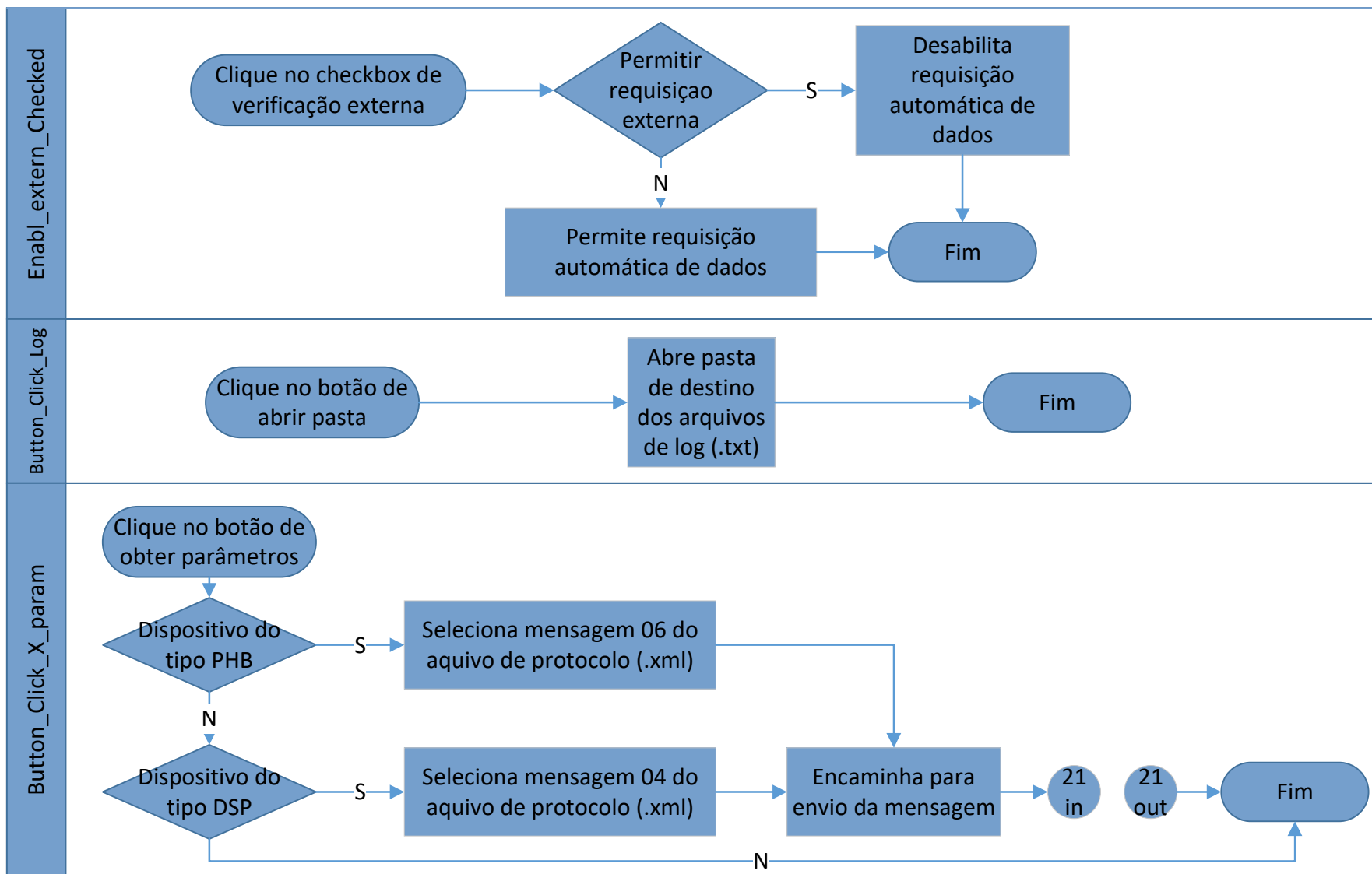


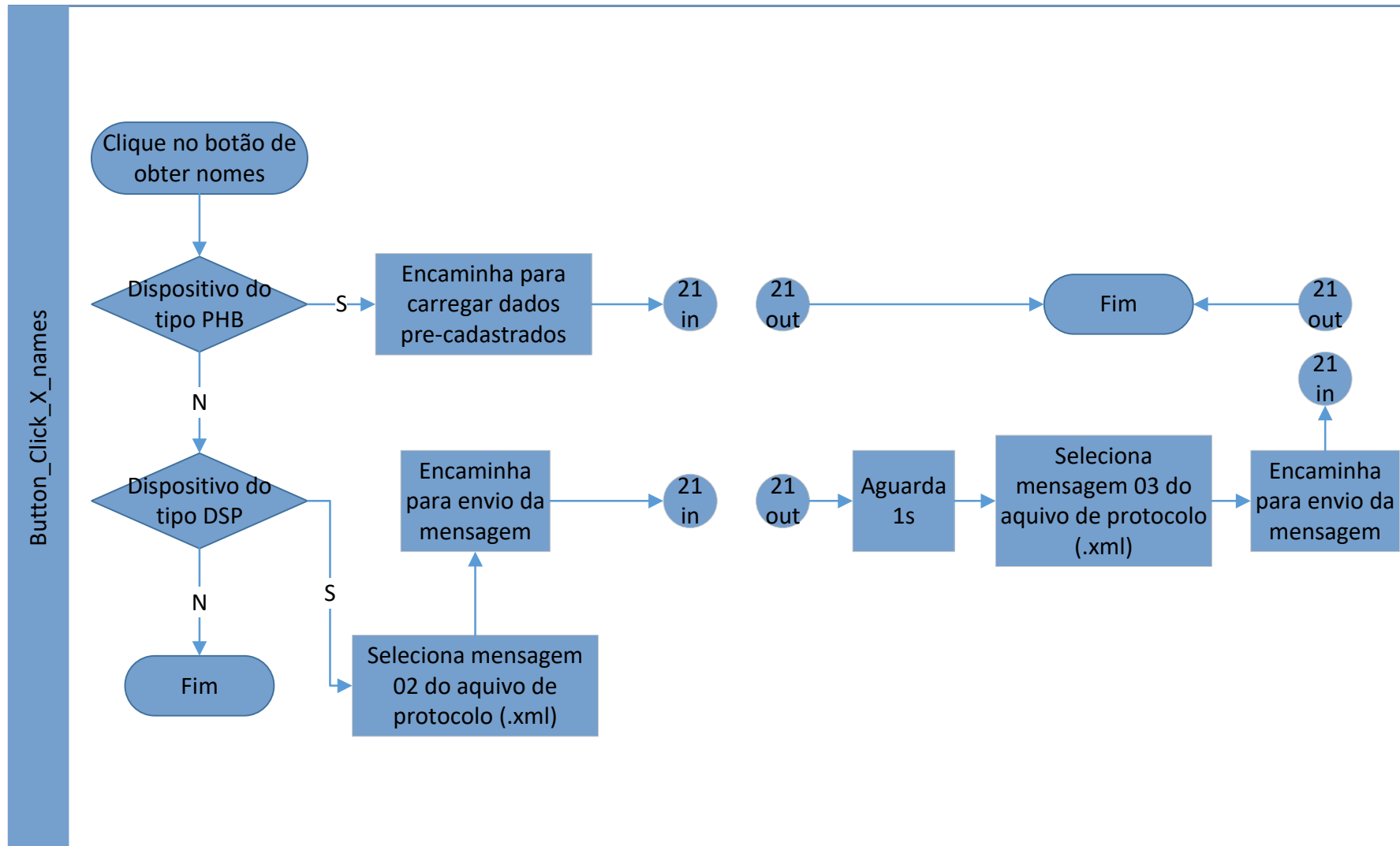


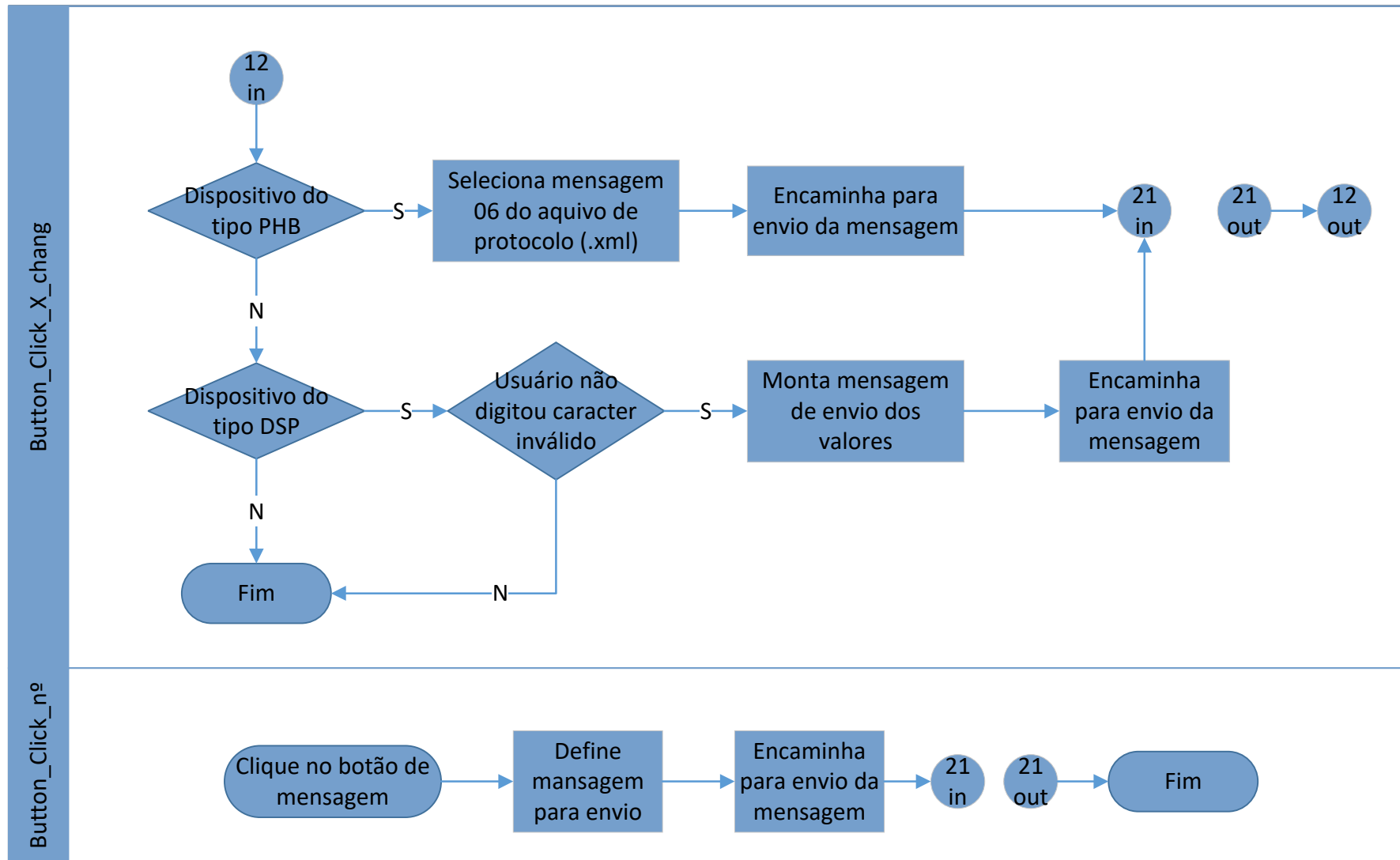








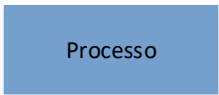




APÊNDICE C – FLUXOGRAMA DE FUNCIONAMENTO DO FIRMWARE DO ESP8266

Este apêndice traz o fluxograma de funcionamento do firmware para o microcontrolador ESP8266, na versão 4. Este foi criado em linguagem C, via interface de programação *IDE Arduino*. O fluxograma segue a definição de objetos mostrado na Fig. C.1.

Figura C.1 – Descrição de objetos utilizados no fluxograma apresentado

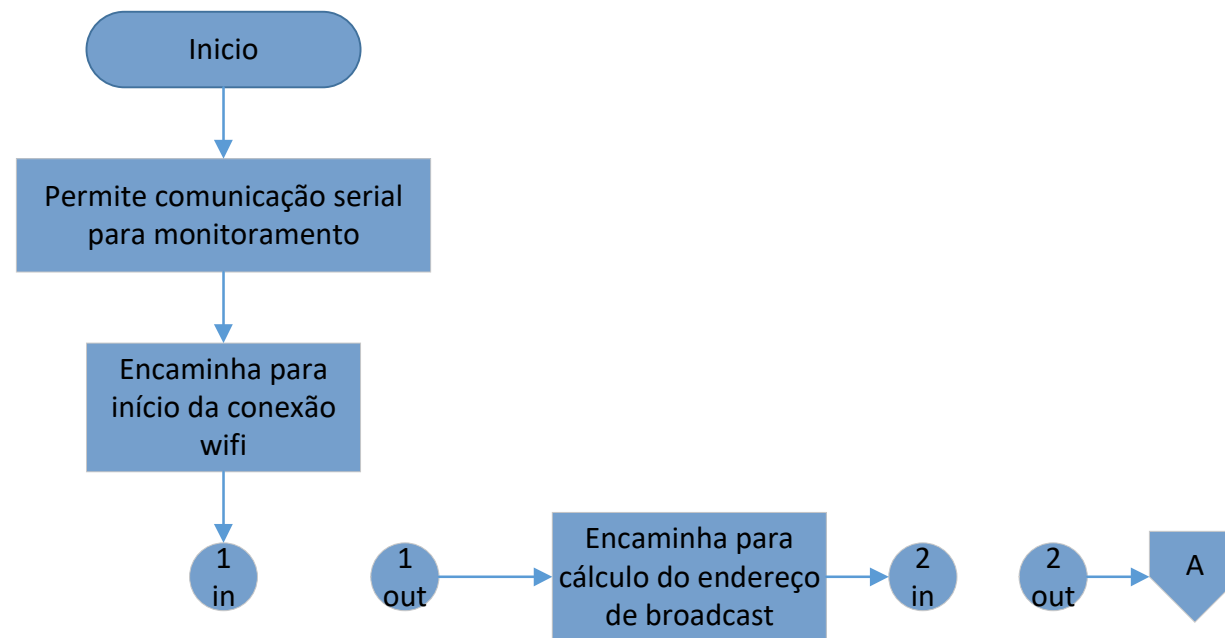
Forma	Descrição
	Início de fluxograma
	Decisão para tomada de ação
	Execução de ação
	Entrada em função de número 'X'
	Saída da função de número 'X'
	Continuação fora da página Z
	Fim de fluxograma

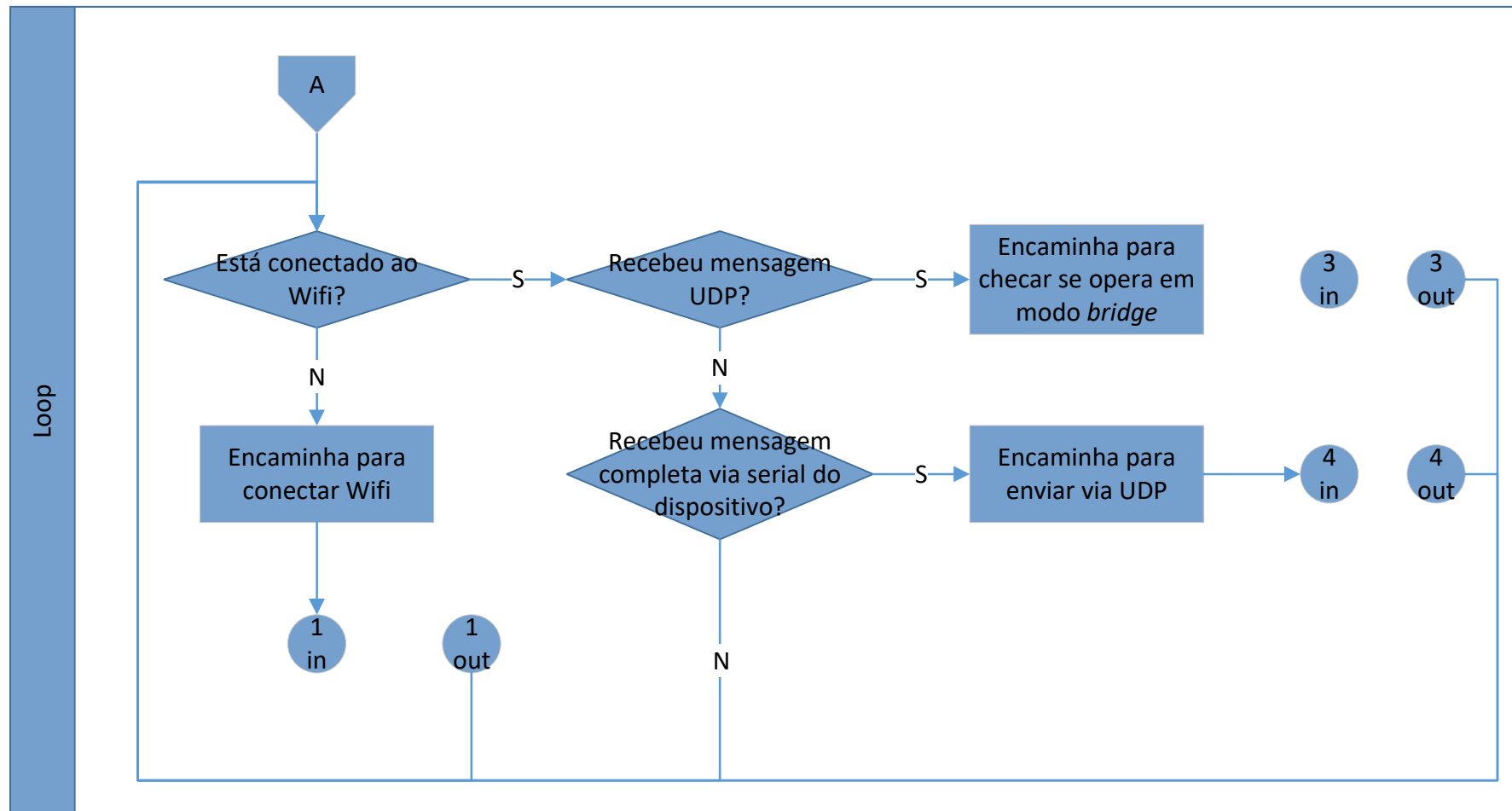
Fonte: Elaborado pelo autor (2018).

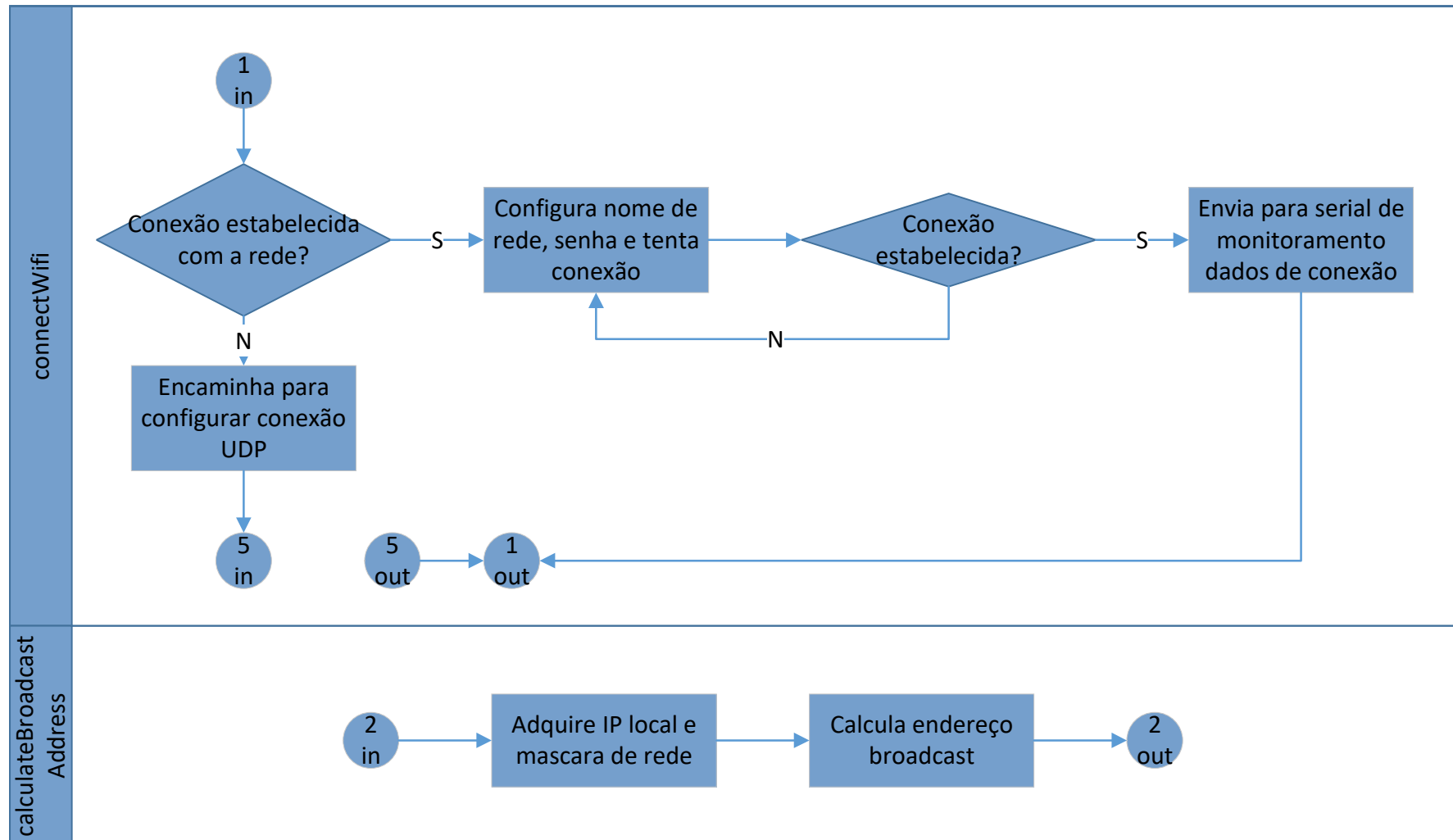
ESP8266 Firmware – (Arduino IDE: Inclui ESP8266WiFi library - <https://github.com/esp8266/Arduino/tree/master/doc/esp8266wifi>)

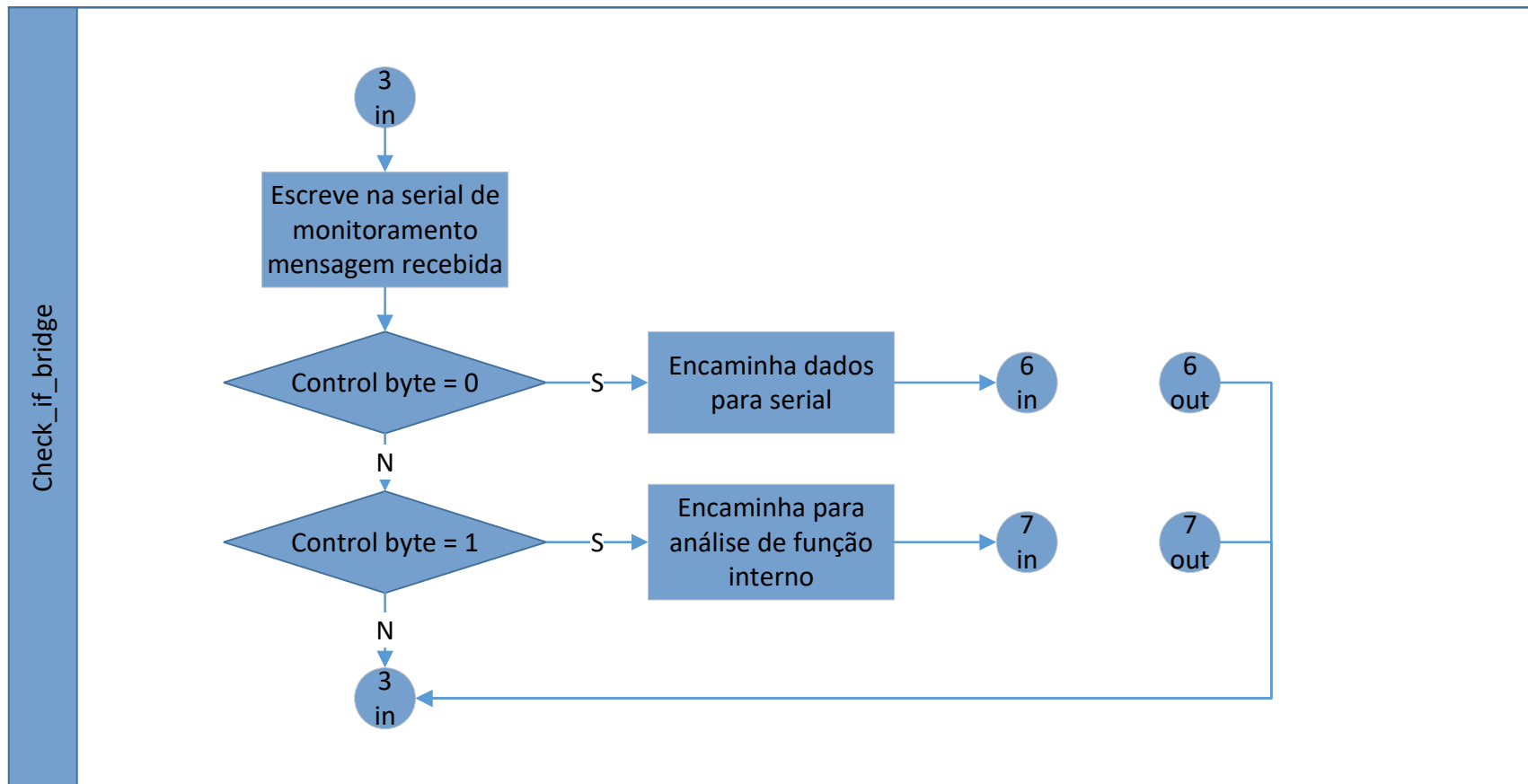
ESPGUI - V4

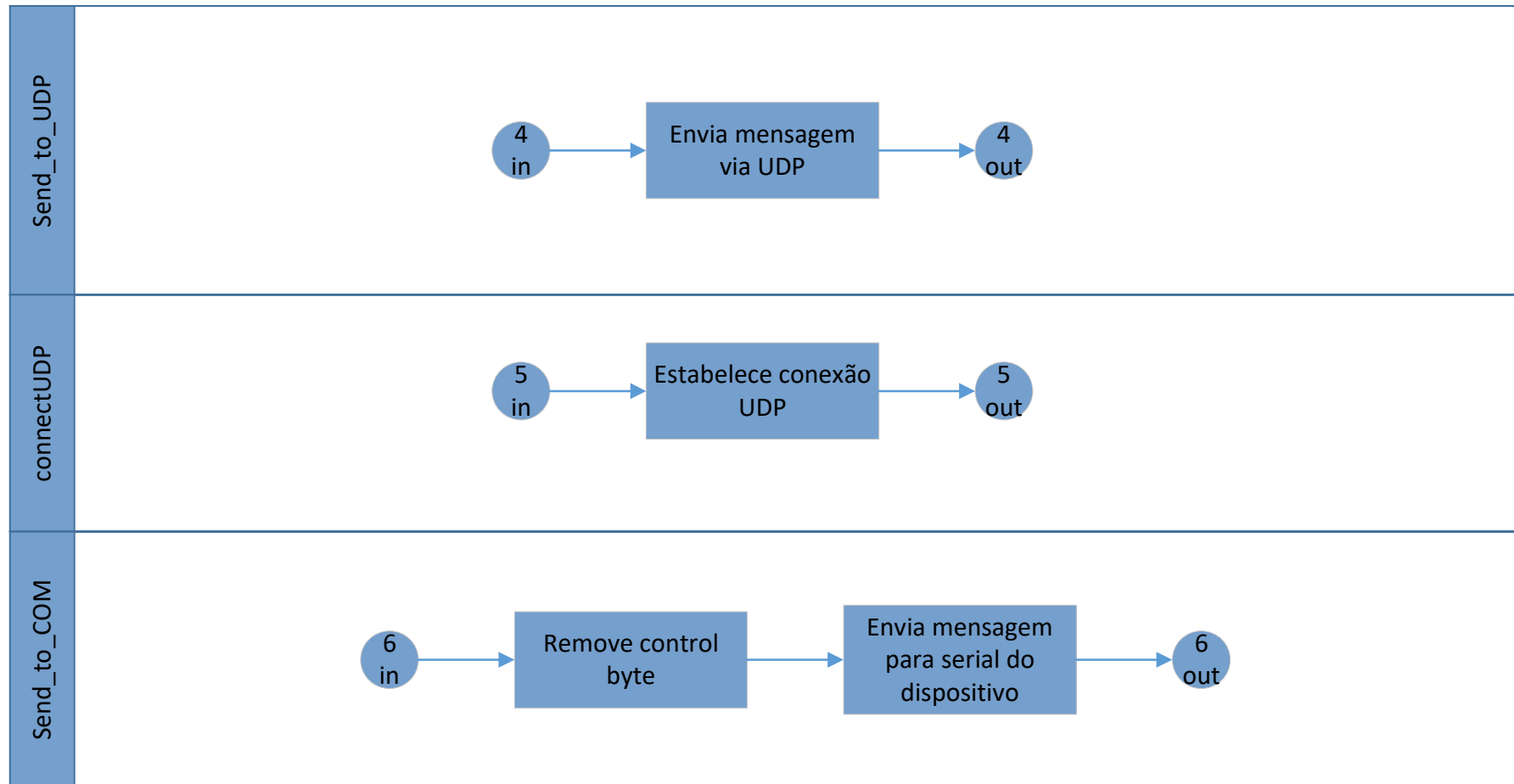
Setup

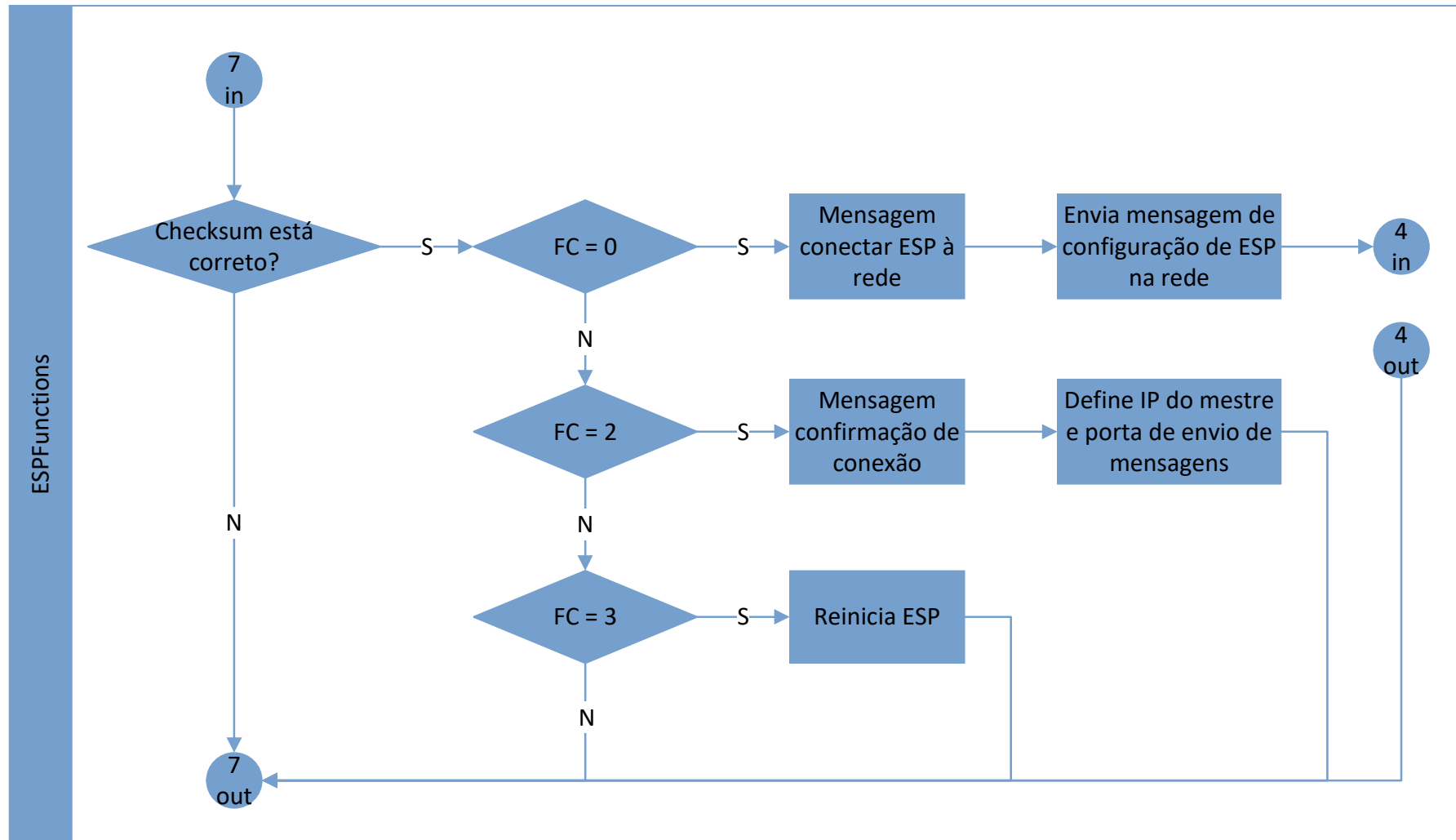








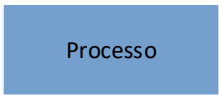




APÊNDICE D – FLUXOGRAMA DE FUNCIONAMENTO DO FIRMWARE DO DSC

Este apêndice traz o fluxograma de funcionamento do firmware para o DSC *LAUNCHXL -F28377S*. Este foi criado em linguagem C, via interface de programação *Code Composer Studio*. O fluxograma segue a definição de objetos mostrado na Fig. D.1.

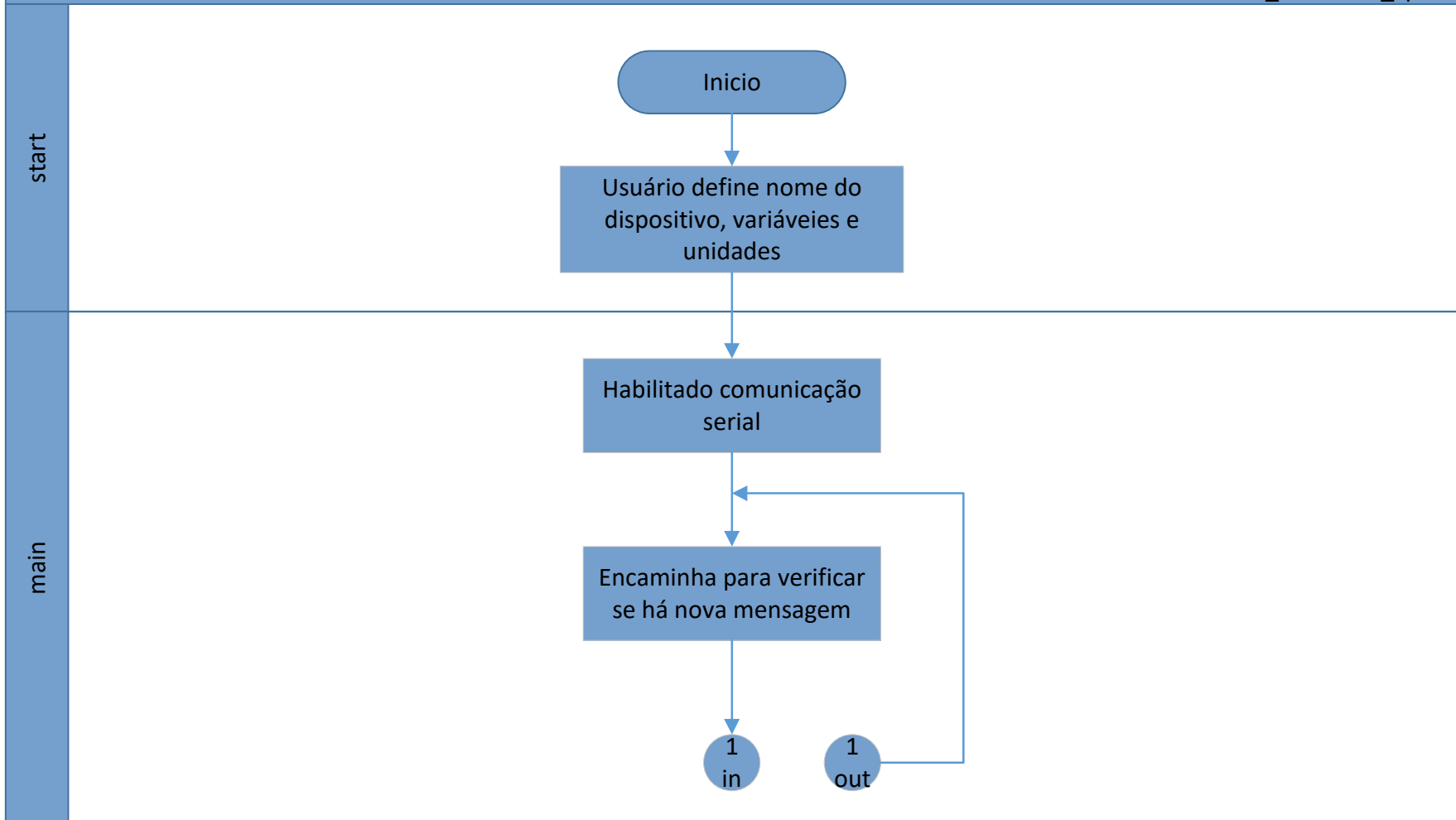
Figura D.1 – Descrição de objetos utilizados no fluxograma apresentado

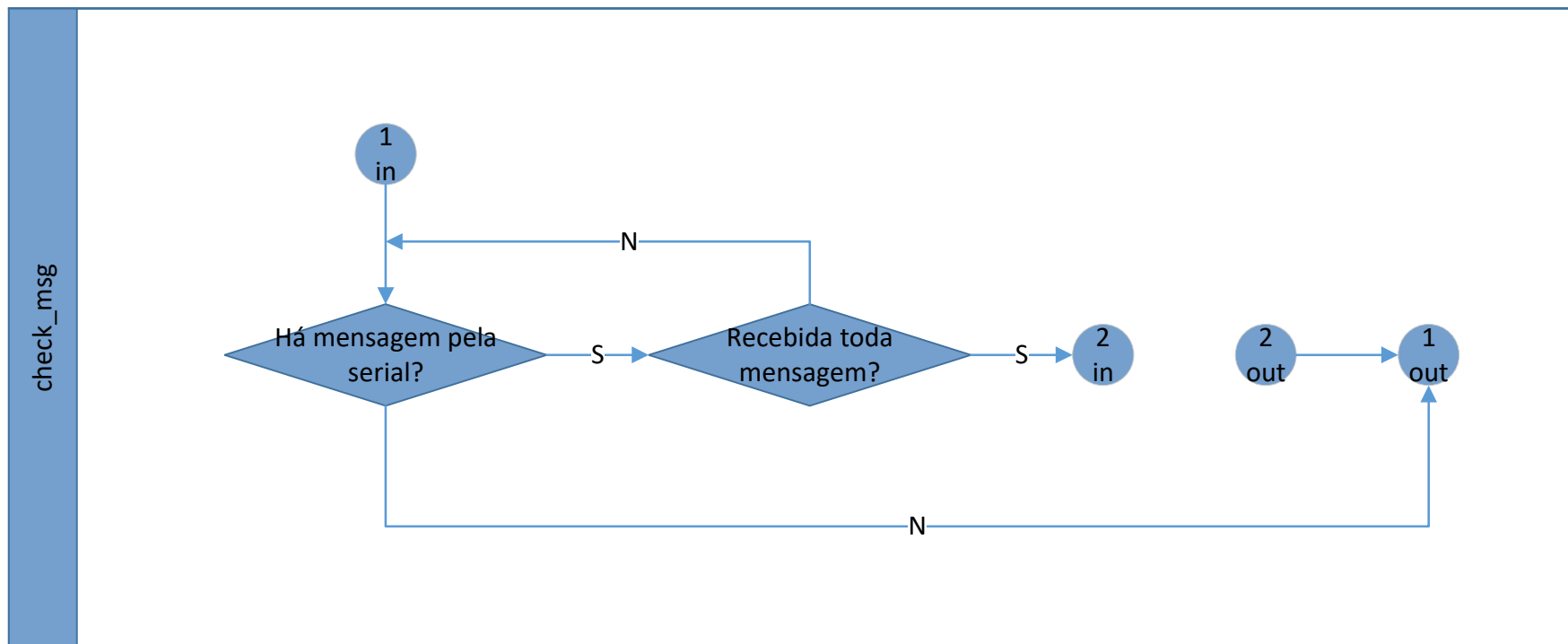
Forma	Descrição
	Início de fluxograma
	Decisão para tomada de ação
	Execução de ação
	Entrada em função de número 'X'
	Saída da função de número 'X'
	Continuação fora da página Z
	Fim de fluxograma

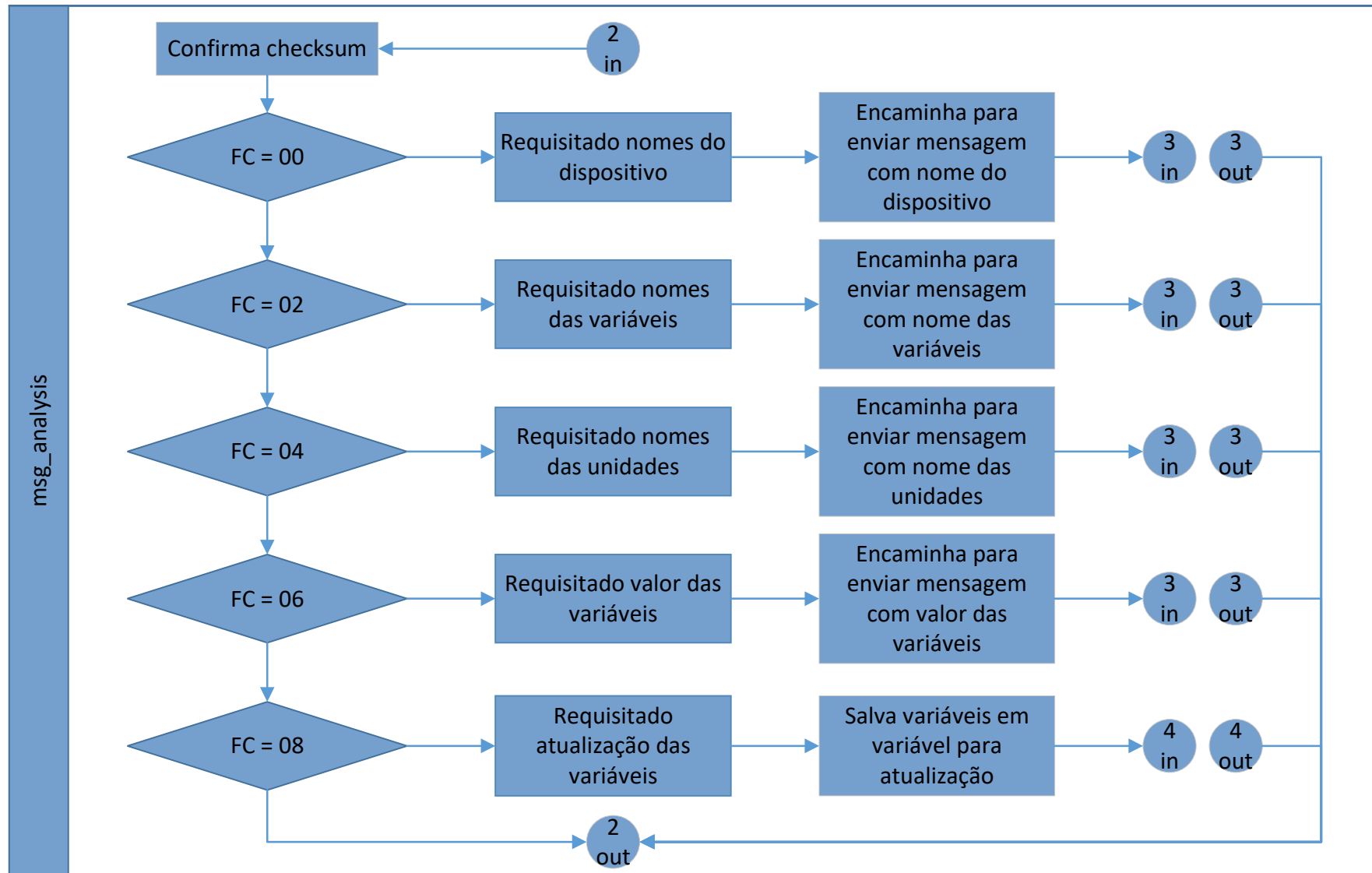
Fonte: Elaborado pelo autor (2018).

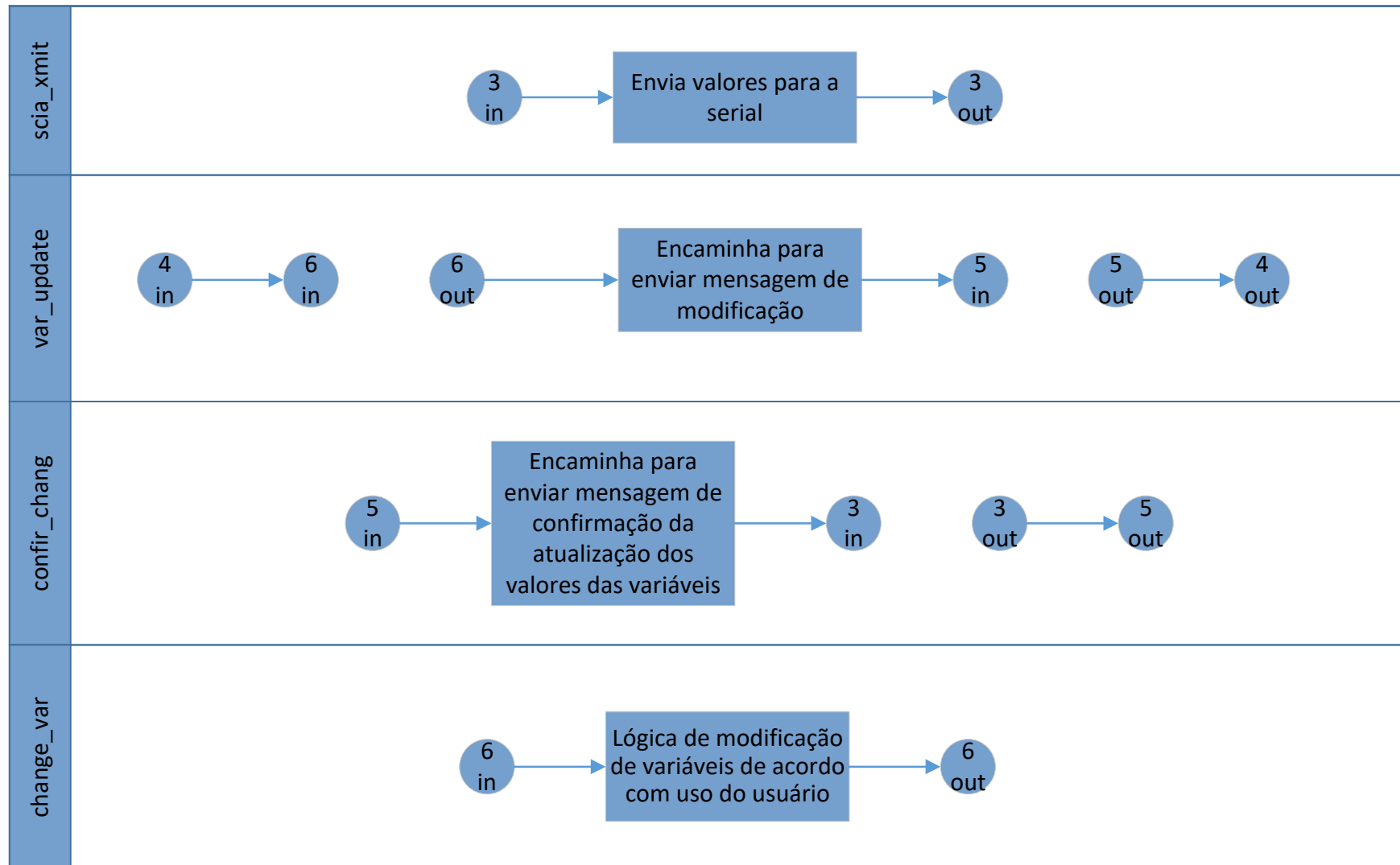
DSC - F28377S – Code Composer Studio

Sci_echoback_cpu01





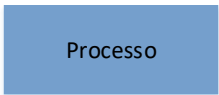




APÊNDICE E – FLUXOGRAMA DE FUNCIONAMENTO DO SOFTWARE EXTERNO (MATLAB)

Este apêndice traz o fluxograma de funcionamento de um software externo implementado em Matlab através do Simulink. O fluxograma segue a definição de objetos mostrado na Fig. E.1.

Figura E.1 – Descrição de objetos utilizados no fluxograma apresentado

Forma	Descrição
	Início de fluxograma
	Decisão para tomada de ação
	Execução de ação
	Entrada em função de número 'X'
	Saída da função de número 'X'
	Continuação fora da página Z
	Fim de fluxograma

Fonte: Elaborado pelo autor (2018).

Matlab como software externo – Via Simulink

Matlab_UDP – V4

