UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GIOVANI CHRESTANI - 00262697

PROJETO DE COMPARTIMENTO INTELIGENTE PARA COMPRIMIDOS COM CONFIGURAÇÃO E MONITORAMENTO POR PROFISSIONAL DE SAÚDE

Porto Alegre
2021

GIOVANI CHRESTANI - 00262697

PROJETO DE COMPARTIMENTO INTELIGENTE PARA COMPRIMIDOS COM CONFIGURAÇÃO E MONITORAMENTO POR PROFISSIONAL DE SAÚDE

Projeto de Diplomação apresentado como requisito parcial para obtenção do título de Engenheiro Eletricista.

ORIENTADOR:

Prof. Dr. Tiago Balen

Porto Alegre
2021

GIOVANI CHRESTANI - 00262697

PROJETO DE COMPARTIMENTO INTELIGENTE PARA COMPRIMIDOS COM CONFIGURAÇÃO E MONITORAMENTO POR PROFISSIONAL DE SAÚDE

Este Projeto de Diplomação foi julgado adequado para a obtenção dos créditos da disciplina de PD2 do curso e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Tiago Balen, UFRGS
Doutor pela UFRGS – Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Tiago Balen, UFRGS
Doutor pela UFRGS – Porto Alegre, Brasil

Prof. Dr. Raphael Martins Brum, UFRGS
Doutor pela UFRGS – Porto Alegre, Brasil

Prof. Dr. Ivan Muller, UFRGS
Doutor pela UFRGS – Porto Alegre, Brasil

_____
Raphael Martins Brum
Coordenador de Curso

Porto Alegre, dezembro de 2021.

RESUMO

Com o crescente uso de soluções eletrônicas em controle e monitoramento, é natural que sistemas de saúde que envolvam acompanhamento de paciente sejam desenvolvidos. Este trabalho propõe uma solução para o gerenciamento e monitoramento do uso de remédios de forma periódica. A solução é composta de um compartimento que aloca remédios, uma placa de circuito impresso que gerencia os dados e atua em indicadores visuais e sonoros, uma plataforma *web* e um meio de comunicação USB para trocar informações entre o produto e a plataforma. O foco principal deste trabalho está no projeto em *hardware*, mostrando detalhes de como os requisitos de um produto se desdobram para os níveis mais básicos do projeto, incluindo esquemático. De início, apresenta-se conceitos importantes no entendimento do projeto e motivações para o desenvolvimento de tal produto. Após, propõe-se a solução em nível sistêmico, define-se arquitetura de *hardware* e suas soluções em nível de esquemático e *layout*. É apresentada também a solução de programa, implementada em um microcontrolador ATmega328p, da família AVR. Com a montagem da placa finalizada, testes são conduzidos, mostrando concordância entre o projeto e o resultado final.

Palavras-chave: Monitoramento de paciente, Gerenciamento de remédios, Caixa de remédios, Projeto de hardware.

ABSTRACT

With the increasing use of electronic solutions for controling and monitoring, it is natural that health systems that encompass patient follow-up will be developed. This work proposes a solution for managing and monitoring the use of medicines taken periodically. This solution is composed by a box that contains medicines, a printed circuit board that manages the data and acts on visual and audio indicators, a web platform and a USB communication scheme to exchange informations between the product and the platform. The main target of this work is the hardware design, going through the details of how the product requisites unfolds to the design of the basic levels of abstraction, including schematics. At the beginning, important concepts for understanding the design and the motivations for the development of the product are presented. After that, a solution at the schematic and layout level is proposed. Additionally, it is shown a solution for the program implemented on the microcontroller ATmega328p, of the AVR family. After performing the assembly of the components on the board, tests were conducted, proving that the design agrees with the final resuts.

Keywords: Patient monitoring, medicine management, pill box, hardware design.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

WHO	World Health Organization
PCB	Printed Circuit Board
USB	Universal Serial Bus
DTR	Data Terminal Ready
FET	Field Effect Transistor
LED	Light Emmiting Diode
NTC	Negative Temperature Coefficient
EHR	Electronic Health Records
CPU	Central Processing Unit
USART	Universal Synchronous Asynchronous Receiver-Transmiter
PWM	Pulse Width Modulation

1 INTRODUÇÃO

Com a tendência nos últimos anos de uma migração em massa de soluções eletrônicas para sistemas cada vez mais interconectados, a exemplo da IoT (*Internet of Things*), diversas aplicações, que antes eram de difícil controle e atuação, hoje são possíveis, bem como desejadas. No caso de sistemas de saúde, muitas das práticas antes associadas a médicos e pacientes tem, agora, a possibilidade de serem movidas para uma plataforma integrada e de fácil atuação à distância. Além disso, no caso de pacientes cujo tratamento fora receitado com base em medicamentos, uma série de más práticas podem ser evitadas com a implementação de sistemas que impeçam a atuação de pessoas não autorizadas no gerenciamento e uso de remédios.

O presente trabalho de conclusão de curso tem como objetivo apresentar tal solução, com base em um dispositivo que se encarregue de gerenciar e monitorar o uso de medicamentos por um usuário, bem como estabelecer um caminho para sua configuração, abrangendo o médico, que receita o medicamento, o farmacêutico, que preenche a caixa com o medicamento e, por fim, o usuário do produto, que faz uso dos remédios prescritos. Em cada uma destas três interações, há trocas de informação com a caixa, seja para configurar a receita médica virtualmente, informar que os compartimentos da caixa foram de fato preenchidos e informar a caixa de que o respectivo remédio foi tomado. Com esta proposta inicial, tem-se também como objetivo demonstrar como uma estrutura de planejamento em projeto de engenharia pode ser concebida e executada, apresentando um pouco do que é recorrente na área de desenvolvimento na indústria.

Dada a abrangência de possíveis escopos em um projeto que envolva o produto proposto, decidiu-se focar este documento na área de projeto em eletrônica para placas de circuito impresso, acreditando-se ser este foco o mais aderente ao que fora estudado ao longo do curso.

O Capítulo 2 trata de uma breve revisão teórica de conceitos que serão necessários para o entendimento do restante do documento. Neste mesmo capítulo é, também, apresentado um breve panorama do que já existe no mercado em termos de produtos com proposta similar. O Capítulo 3 leva em conta as necessidades apresentadas e descreve o que o produto proposto deve satisfazer e, com isso, determina requisitos para seu desenvolvimento. Estes

requisitos serão, então, utilizados para adentrar um nível ainda mais baixo no *design*, no Capítulo 4. Neste capítulo, é apresentada a solução em nível de *hardware* considerando esquemáticos e algumas simulações. De maneira similar, o Capítulo 5 trata da passagem dos requisitos de produto para o nível de *firmware* e demonstra a solução proposta para tal problema. Após a montagem da placa, testes são executados e sua apresentação é feita no Capítulo 6. Os resultados destes testes são postos à prova comparando-se com as simulações e comportamentos esperados, segundo informações dos fabricantes dos circuitos integrados.

Os blocos de circuito propostos, ao final, corresponderam aos requisitos impostos, permitindo o funcionamento do programa executado no microcontrolador e tornando o protótipo funcional.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas eletrônicos de auxílio a saúde

Segundo a Organização Mundial da Saúde (2011), o termo "mHealth"pode ser definido como: “...a prática de saúde médica e pública suportada por dispositivos móveis, dispositivos de monitoramento de pacientes, assistentes pessoais digitais (PDAs), e outros dispositivos sem fio”. Com essa definição, é possível enquadrar este projeto no contexto de mHealth.

Pesquisas mostram uma crescente aderência das tecnologias mHealth em sistemas de saúde, possibilitando assim integrar as tecnologias atuais na otimização de procedimentos nessa área (MEETOO; RYLANCE; ABUHAIMID, 2018). O que possibilita tal crescimento é, em grande parte, o desenvolvimento de aplicativos para smartphone e a acessibilidade de tais dispositivos a uma porção cada vez maior da população mundial. Para suportar essa mudança, uma variedade de tecnologias de informação estão sendo adotadas e implementadas para fornecedores, administradores e consumidores na indústria de cuidados com a saúde. Gravadores eletrônicos de saúde (EHRs) são parte importante disso. Os EHRs podem ser considerados como sistemas de armazenamento de informação digitalizada sobre os cuidados de saúde de um paciente, que pode ser compartilhada entre as várias partes. Não há dúvidas quanto a necessidade futura dos EHRs. Com computadores coletando dados sobre a doença, o tratamento e os resultados de um paciente, é possível, automaticamente, obter-se informação válida sobre a efetividade destes tratamentos, ou relações entre efeitos colaterais e características populacionais (THIMBLEBY, 2013). Os benefícios para pesquisadores da área médica (até mesmo epidemiologistas) são imensos e, com isso, os mesmos podem extrair conclusões que afetam diretamente os indivíduos.

Alguns produtos já foram lançados visando a integração dos cuidados de saúde através do uso de medicamentos a um sistema inteligente. Produtos em forma de dispensadores são mais comuns. O princípio de funcionamento destes dispensadores é simples: um cuidador responsável ou o próprio usuário dos medicamentos faz a alocação dos remédios em compartimentos, de acordo com os dias da semana (ou outro intervalo de tempo) em que os mesmos devem ser tomados. No momento correto do uso dos medicamentos, sinalizações

são enviadas para o usuário, seja através de mensagens por *e-mail* ou por sinalizações no próprio produto. Dois exemplos de tais produtos são Thales Group (2021) e Memo Box (2021). Uma ideia similar, com sensores de aproximação está em Guerrero Ulloa et al. (2020).

Outras propostas de produtos são voltadas à possibilidade de configuração da caixa inteligente, programando-a para que medicamentos individualmente identificados sejam tomados em seus respectivos horários específicos, necessitando que se faça apenas uma vez suas alocações nos compartimentos. Um destes exemplos está em Abdul Minaam e Abd-ELfattah (2018). Algo que não foi integrado nestes exemplos é a infraestrutura de dados entre médicos e pacientes em uma única plataforma. Dados estes que dizem respeito à receita de medicamentos, acompanhamento de tratamentos e suas eficácias.

A construção destes sistemas demanda a integração de diferentes tecnologias, notadamente, microcontroladores, dispositivos de potência e baterias, além de circuitos de comunicação e interfaces de software. A próxima seção descreve os elementos básicos de hardware necessários para implementação de um protótipo de placa utilizada para o gerenciamento do cronograma de uso de remédios, que é o foco deste trabalho.

2.2 Fundamentos dos elementos de *hardware* do protótipo

2.2.1 Conversores Boost

Com a finalidade de suprir alimentação para o circuito de microcontrolador deste projeto, há a necessidade de conversão de um nível de tensão mais baixo para um nível de tensão mais alto. Para tanto, serão projetados circuitos de conversores *boost*.

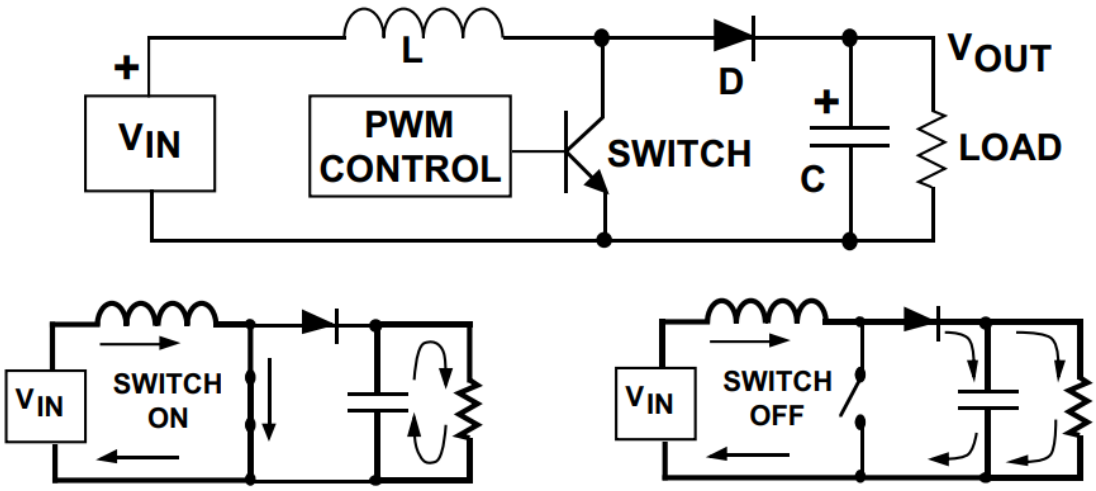
De acordo com o artigo da empresa Texas Instruments Inc. (2019b) os conversores *boost* são destinados a produzirem uma tensão contínua maior do que a tensão de entrada e de mesma polaridade. A Figura 1 mostra o diagrama esquemático de um conversor boost.

Como se observa, o transistor atua como uma chave, por vezes conectando um dos terminais do indutor ao terra e em outras abrindo esta conexão. Os fluxos de corrente para ambos os casos estão descritos na imagem. Para o caso do transistor atuar como uma chave em posição fechada, a corrente faz uma rampa de subida e estabelece uma tensão sobre o indutor. No momento em que esta chave é aberta, o decréscimo da corrente no indutor provoca uma tensão de polaridade oposta à anterior, respeitando a relação entre tensão e corrente no indutor da equação:

$$V_L = L\frac{di}{dt} \tag{1}$$

Com este fenômeno, a tensão sobre os terminais é somada à tensão de entrada, ativando o diodo, carregando o capacitor de saída e estabelecendo uma tensão de saída maior do que a tensão de entrada. Outro fato importante de se notar é que, como a tensão de saída

Figura 1: *Conversor Boost.*



Fonte: Texas Instruments Inc. (2019b)

é sempre maior do que a tensão de entrada, a corrente de saída será sempre menor que a corrente de entrada.

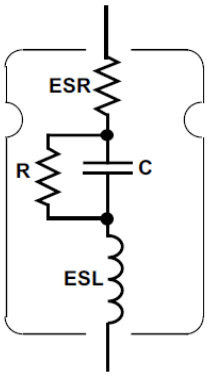
O controle do transistor atuando como chave é feito através de modulação por largura de pulso (PWM). Posto de forma simples, o laço de realimentação ajusta a tensão de saída através do tempo dos estados de *ON* e *OFF* do elemento de chaveamento do conversor (transistor).

Alguns conceitos no projeto de conversores chaveados são muito importantes para que se obtenha um projeto estável, eficiente e funcional. Dentre eles, serão discutidos os elementos parasitas de capacitores e cuidados com aterramento.

2.2.1.1 Parasitas no capacitor

Os capacitores possuem elementos parasitas indutivos e resistivos em série com sua capacitância. A Figura 2 ilustra como se dá a disposição da resistência equivalente série (ESR) e indutância equivalente série (ESL) neste componente.

Figura 2: *Parasitas no capacitor.*



Fonte: Texas Instruments Inc. (2019b)

A ESR é responsável pelo aquecimento do capacitor devido à dissipação de potência causada pela corrente de *ripple*. A ESL é a característica que limita o capacitor para uso em altas frequências. Esta é a razão principal para que capacitores eletrolíticos sejam colocados em paralelo com capacitores cerâmicos. O capacitor, a ESR e a ESL formam um circuito ressonante cuja frequência de ressonância deve ser a maior possível. Reguladores chaveados produzem tensões de *ripple* que, se próximas desta frequência de ressonância, podem provocar oscilações.

Na entrada de reguladores chaveados, a corrente é drenada em janelas de tempo periódicas, de acordo com a frequência de chaveamento e do *duty cycle*. Neste momento, altos valores em $\frac{di}{dt}$ são requeridos e, portanto, é interessante diminuir ao máximo as indutâncias parasitas nos capacitores de entrada e na trilha, obtendo-se, assim, menores picos de tensão.

Na saída destes reguladores uma tensão de *ripple* sempre estará presente e esta depende muito da ESR dos capacitores de saída. Quanto maior a ESR, maior o *ripple* de tensão. Além disso, uma alta ESR dissipa mais potência e a eficiência do regulador cai.

2.2.1.2 Aterramento apropriado

A referência em terra (ou GND) idealmente deve ser constante ao longo da placa. Na realidade isto não ocorre devido à impedância não nula das trilhas de GND da placa. Em baixas frequências, isto é um problema simples, o único parâmetro importante a se tomar cuidado é a resistência DC das trilhas do caminho de GND. Em frequências mais altas, as indutâncias do caminho de GND começam a ser mais representativas e, seguindo o princípio da Equação 1, altos picos de tensão podem aparecer. A corrente de chaveamento em frequências altas tende a fluir na beira do condutor, em um comportamento chamado efeito *skin*. Com isso, é interessante que as trilhas de GND sejam bastante largas e, quando possível, camadas inteiras de placas devem ser dedicadas a GND.

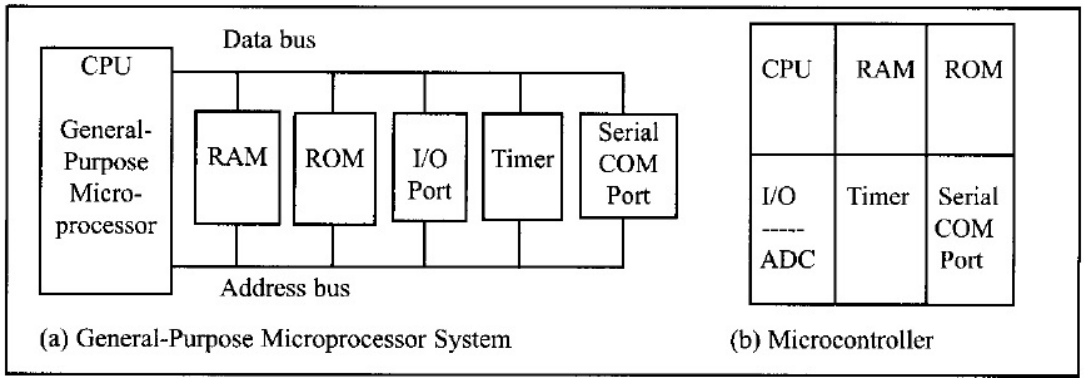
2.2.2 Microcontroladores e família AVR

Para que seja implementado o controle, armazenamento e gerenciamento de dados na placa de circuito utilizada, é necessário o uso de algum elemento de circuito integrado que proveja tais funcionalidades. Os microcontroladores são, hoje, amplamente utilizados e muito acessíveis no mercado, com uma demanda de consumo crescente, como mostra o relatório de Grand View Research (2021).

Segundo Mazidi (2011), um microcontrolador tem um microprocessador (CPU) junto a uma quantidade fixa de RAM, ROM, portas I/O, e temporizadores. Assim sendo, todas estas atribuições anexadas ao microprocessador estão embarcadas em um mesmo chip, ao contrário de chips de microprocessadores em si. Com isso, o projetista não tem a possibilidade de adicionar suas próprias opções de memórias e I/O diretamente no microprocessador. Isto ocorre com a vantagem de otimização de espaço e, quando as neces-

sidades de projeto estão de acordo com as especificações do microcontrolador, um pedaço significativo do projeto é simplificado, diminuindo, também, o tempo e trazendo segurança para a funcionalidade do mesmo. Em geral, casos em que o poder computacional é mais tolerante do que o espaço físico de ocupação do componente na placa, microcontroladores são uma boa alternativa. A Figura 3 mostra as diferenças citadas entre o microcontrolador e o microprocessador.

Figura 3: *Diferenças entre microprocessador e microcontrolador.*



Fonte: Mazidi (2011)

Outra grande facilidade nos dias atuais é a presença dos chamados *single board computers* (SBC), constituídos basicamente de uma unidade processadora, como um microcontrolador, e periféricos que se comunicam ou se conectam com esta unidade central. A ideia do SBC é prover uma maneira acessível para que um protótipo seja feito com custo baixo e funcionalidades suficientes. Dentro deste escopo e, pensando em otimizar o tempo de projeto e testes, um dos mais conhecidos SBC foi utilizado para a implementação do protótipo inicial deste projeto: o Arduino Uno. Este SBC faz uso do microcontrolador Atmega 328p e, portanto, será este o principal elemento dentro do projeto da placa que envolve este projeto. Mais um fator de grande importância na escolha do microcontrolador é a disponibilidade de software para programá-lo. O software Microchip Studio abarca as necessidades de programação do Atmega328p, já que este é o software oficial do fabricante do chip. Além disso, é de fácil utilização e foi anteriormente testado em outras oportunidades, garantindo rápido desenvolvimento para o projeto.

Algumas considerações são importantes no que diz respeito à família de microcontroladores a qual o Atmega328p pertence. Este componente é da família AVR que, com algumas exceções, trabalha em 8 bits. Isto quer dizer que dados são transmitidos em blocos de 8 bits. Além disso, o Atmega328p contém ROM, RAM, EEPROM, temporizadores, portas I/O, ADC, PWM, USART, SPI, dentre outras funcionalidades. Como grande parte das instruções são executadas em um ou dois ciclos de *clock*, é uma boa estimativa afirmar que este dispositivo opera em uma capacidade de ao menos 8 MIPS, considerando um *clock* de 16 MHz.

A fim de poder executar os comandos e carregar as instruções, o microcontrolador deve ser programado pelo usuário. A linguagem de programação mais utilizada para a família AVR é C. Outras alternativas existem, porém esta otimiza tempo com bibliotecas e também permite o fácil acesso em baixo nível.

2.2.3 Comunicação UART

Outra necessidade especial para o projeto é a presença de um mecanismo de comunicação cabeada. Para tanto, é preciso escolher qual a forma mais eficiente e disponível para o projeto. Este projeto é fortemente baseado em requisitos de testabilidade e mitigação de riscos, com o protótipo SBC "Arduino Uno". O Arduino Uno trabalha com interface USB e comunicação serial USART. Sendo assim, será esta a tecnologia utilizada, que a seguir será detalhada.

A sigla USART significa "*Universal synchronous asynchronous receiver-transmiter*". Como neste projeto estará sendo utilizado apenas o modo de operação assíncrono de comunicação, será abordada aqui apenas a UART. O termo assíncrono se refere à não existência de um *clock* em comum entre o transmissor e receptor. A informação necessária para se fazer a captura dos dados na amostragem correta por parte do receptor é o *baud rate*, que é a taxa de símbolos por segundo. Este protocolo de comunicação atua de modo serial e, com isso, necessita apenas dois fios: o de transmissão (TX) e o de recepção (RX). Cada um destes comunica um conjunto de bits sequencialmente. Na UART a comunicação se da em forma de pacotes (ANALOG DEVICES INC., 2020).

Figura 4: *Pacote de UART.*

Fonte: Analog Devices Inc. (2020)

A Figura 4 mostra os bits que compõe um pacote de UART. São estes:

- *Start Bit*: O canal de comunicação UART está em nível elétrico alto quando não utilizado. O bit que da início à transmissão de um pacote UART é colocado em nível baixo para sinalizar o início deste procedimento.

- *Data Frame*: Este conjunto de bits são os que correspondem aos dados em si, cuja transmissão se deseja.

- *Parity*: O bit de paridade descreve se a soma de valores altos na corrente de bits é par ou ímpar. Este bit, então, serve como um ponto de checagem na recepção. Verifica-se, então, se o pacote foi recebido na sua integridade ou se houveram falhas que possam ter causado uma troca de nível elétrico em algum dos bits de dado.

- *Stop Bits*: Estes bits acusam o final do pacote UART recebido. Para fazer esta sinalização, o transmissor muda de nível elétrico baixo para alto durante o período de um a dois bits.

O bloco de circuito dedicado à UART, portanto, se encarrega de (NANDA; PATTNAIK, 2016):

- Converter *bytes* provenientes de blocos paralelos para uma única sequência de bits em série, na transmissão;
- Na recepção, fazer a conversão da sequência de bits em série para a forma paralela em que o sistema opera;
- Adicionar bit de paridade para a sequência de bits de saída e fazer a checagem da paridade dos bits recebidos;
- Adicionar bits de *start* e *stop* na transmissão e fazer a interpretação dos mesmos na recepção;
- Manipular interrupções.

3 O PRODUTO

3.1 Proposta de produto

Além da literatura apresentada na Seção 2.1, é comum na vida cotidiana encontrar pessoas que fazem uso de medicamentos em grande quantidade. Seja um idoso ou alguém que sofra de doença crônica, são várias as causas que levam a esta prática. Também muito comum nestes casos, é a existência de alguém encarregado de gerenciar o uso destes medicamentos da maneira correta. Esta pessoa, muitas vezes, não é a mesma que é medicada. Alguns dos motivos por trás disso são a falta de organização e negligência do paciente ou uma falta de capacidade cognitiva para executar essas tarefas.

Fatos expostos, é possível verificar alguns problemas que decorrem desta forma de gerenciar os medicamentos. Com o aumento no número de unidades a serem consumidas, a organização pode ficar um tanto complicada e de difícil controle se o responsável não tomar os devidos cuidados. Com isso, problemas decorrentes de erros humanos são naturais a aparecerem. Possíveis erros são: a troca de medicamentos, a quantidade incorreta ingerida, o esquecimento do uso no horário correto, etc. Ressalta-se que a execução do tratamento da forma como foi prescrito pelo profissional de saúde é essencial para a sua eficácia, sendo de extrema relevância o desenvolvimento de um dispositivo que permita que isto aconteça da melhor maneira possível e, inclusive, minimize a possibilidade de alteração na forma de ministrar a medicação pela argumentação de qualquer pessoa que não seja o especialista responsável.

Já que estes problemas estão associados, essencialmente, ao erro humano, nada mais natural que encontrar a solução em uma máquina, tornando mais restritas as ações das pessoas envolvidas e, ao mesmo tempo, contribuindo para o bom funcionamento do tratamento, através do monitoramento do uso dos remédios, o gerenciamento do calendário com alertas e, também, armazenando as unidades. Além do benefício individual trazido para cada paciente, com um tratamento mais preciso e seguro, seriam de grande valor os dados adquiridos pelo sistema. Seria, assim, possível associar tratamentos com seus índices de eficácia e também avaliar os próprios profissionais da saúde quanto às suas medidas e escolhas de tratamentos.

A solução proposta consiste em um produto que provê o armazenamento de medicamentos, avisos visuais e sonoros no momento do uso do medicamento específico, alertas de tempo para uso de remédio expirado, configuração de calendário de uso e preenchimento dos compartimentos feito através de uma aplicação por web *browser*. Dentro do escopo deste trabalho, que faz a afirmação dos conhecimentos aprendidos ao longo do curso de engenharia elétrica, a ideia é montar uma solução para este problema envolvendo um protótipo em PCB, que faz comunicação em USB com uma HMI no computador, por onde o médico e o farmacêutico irão interagir. O médico será o responsável por configurar o calendário de uso dos remédios, enquanto que o farmacêutico irá preencher os *slots* devidos, após a consulta médica.

A fim de tornar mais clara a abordagem de solução do problema, é possível dividir o uso do produto em três momentos, enumerados e esclarecidos abaixo.

1. Consulta médica: Este é o momento em que a caixa é configurada. Consideremos que o usuário da caixa de remédios inteligente seja o paciente e que foi identificado no mesmo algum sintoma passível de prescrição de remédio por parte do seu médico. A partir daí, o médico faz o *login* na plataforma online, conecta a caixa através de USB e escreve uma receita, alocando virtualmente o respectivo remédio na caixa e determinando a periodicidade do uso do mesmo.

2. Alocação dos remédios na farmácia: Após a receita de medicamentos por parte do médico, o usuário se dirige à farmácia para a compra dos medicamentos. O farmacêutico responsável então conecta a caixa através do USB e obtém as informações de quais são os medicamentos, quantas unidades e onde os mesmos devem ser alocados na caixa. Estas informações serão obtidas apenas por meio da plataforma, que é onde as são alocadas e gerenciadas através de um banco de dados.

3. Uso em casa: Já em casa, o paciente, então, está com a caixa configurada e preenchida devidamente. Com isso, no momento correto de uso do remédio, o produto irá comunicar o usuário através de sinais sonoros e visuais.

A partir do momento em que a concepção da solução é formalizada, é necessário traduzir as informações para requisitos do produto. Desta forma, o projeto irá descer um nível de abstração e adentrará o domínio de sistema.

3.2 Requisitos do produto

Uma necessidade muito clara da PCB que irá compor o produto é sua alimentação. O uso do remédio pode estar habilitado a acontecer a qualquer momento, já que isto depende do tratamento receitado e da rotina do paciente. Com isso, a placa deve estar alerta a todo instante. A fim de atender esta especificação e flexibilizar o uso do produto em locais onde

não haja disponibilidade para alimentação externa, o uso de uma bateria recarregável é necessário.

Sinalizações visuais e sonoras são, também, importantes para que o usuário possa ter a indicação clara sobre qual o medicamento correto a ser tomado. É necessário que as sinalizações visuais estejam em um local de fácil identificação e que o sinal sonoro provenha de um local que permita a passagem do som sem grandes obstáculos. Sendo assim, é interessante que estes componentes estejam desacoplados da placa. Esta prática também permite um projeto mais flexível com relação à carcaça mecânica, que foge ao escopo deste trabalho.

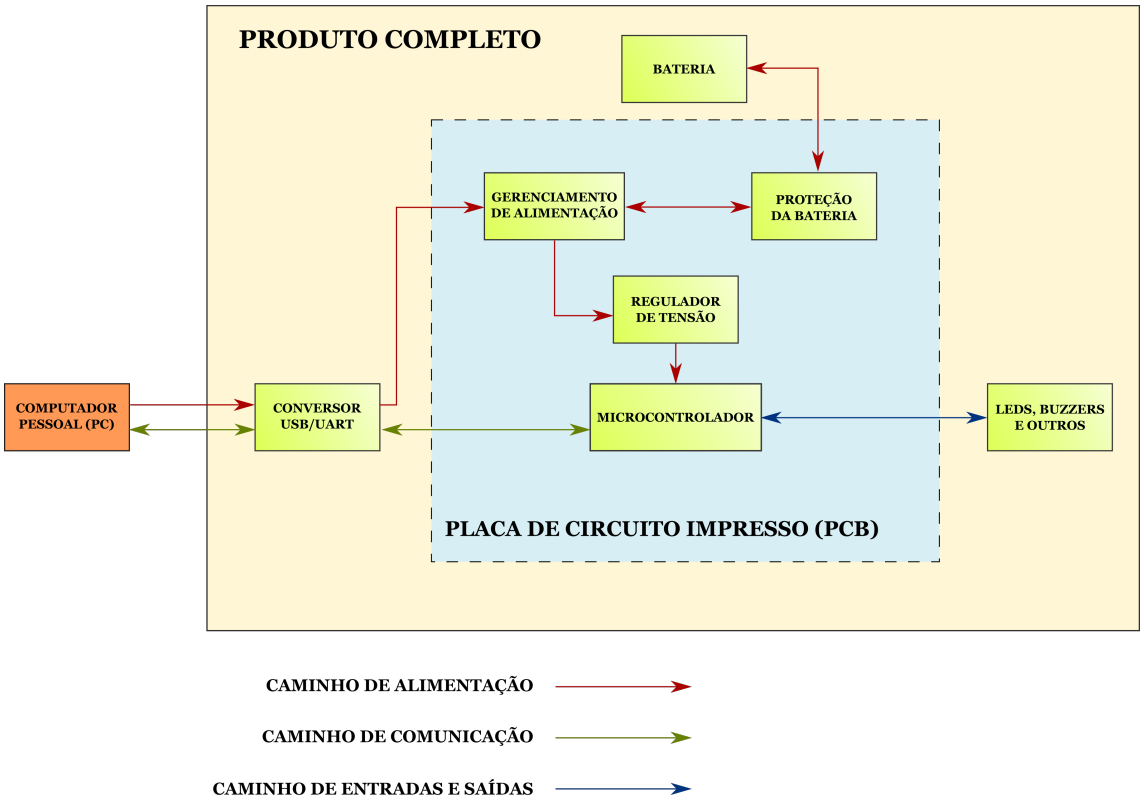
Por fim, o produto requer que configurações sejam carregadas para o dispositivo, seja no momento da consulta do paciente ou na farmácia alocando os medicamentos. Assim, um canal de comunicação deve ser provido entre um computador e a placa. Escolheu-se o uso da porta USB, pela mesma ser comum atualmente, trazendo compatibilidade com dispositivos diversos. Além disso, para que estas informações sejam salvas e as sinalizações sejam executadas de maneira correta, uma unidade lógica se faz necessária. Dada a facilidade no uso e a disponibilidade no mercado, microcontroladores são uma escolha natural para este caso.

4 PROJETO DO HARDWARE

4.1 Arquitetura proposta

Com os requisitos abordados na seção 3.2 é possível estabelecer uma solução a nível de sistema. A Figura 5 mostra um diagrama de blocos resultante.

Figura 5: *Diagrama de blocos da arquitetura de hardware proposta.*



Fonte: O autor (2021)

Observa-se que a placa em si não contém todo sistema, alguns blocos não ficam alocados nela. Para manter uma flexibilidade e minimizar os riscos em um primeiro protótipo, optou-se por implementar a parte de conversão de USB para UART do lado externo da placa, com um módulo dedicado para isso. Sendo assim, é necessário prover a conexão do módulo conversor através de pinos na placa.

A alimentação por bateria também tem de ser externa à placa, já que a sua alocação deve estar em algum compartimento mecânico compatível. Com o uso popular de baterias de polímero de lítio, circuitos integrados (CI) dedicados ao seu uso são cada vez mais frequentes. Isso torna a escolha dessa bateria natural e otimizada para o tempo de projeto. Sua proteção deve ser fornecida quando ela não se faz presente no produto do fabricante. Sendo assim, é interessante prover uma proteção de bateria na placa para uso em casos de baterias com ausência desta funcionalidade. Adicionalmente, um circuito de gerenciamento de tensões e carregamento da bateria deve ser previsto, já que no momento em que a bateria está sendo carregada a alimentação externa provinda do módulo USB/UART deve estar fornecendo a energia ao restante do circuito.

Circuitos de *reset* e *clock* devem estar alocados na placa. O *reset* deve ser uma alternativa para testes e, portanto, um botão será previsto em um local de fácil acesso. O *clock* deve ser estável para a contagem de tempo real.

Como exposto na seção 3.2, os componentes responsáveis pelas sinalizações sonoras e visuais são externos à placa, permitindo suas alocações físicas em locais otimizados.

4.2 Soluções encontradas

A seguir serão detalhadas as escolhas para os blocos de circuito apresentados na Figura 5 que estão dentro do escopo do projeto. O *design* em nível de esquemático é apresentado individualmente para cada bloco e, ao final, estes são unidos para mostrar suas conexões de maneira mais clara.

4.2.1 Microcontrolador

Uma breve pesquisa foi conduzida para avaliar as opções de microcontroladores disponíveis para a aplicação em questão. Dando prioridade ao conhecimento já adquirido previamente no uso de microcontroladores e sua disponibilidade no mercado, optou-se pelo microcontrolador Atmega328p, da família AVR. Detalhes sobre seu funcionamento estão na Seção 2.2.2.

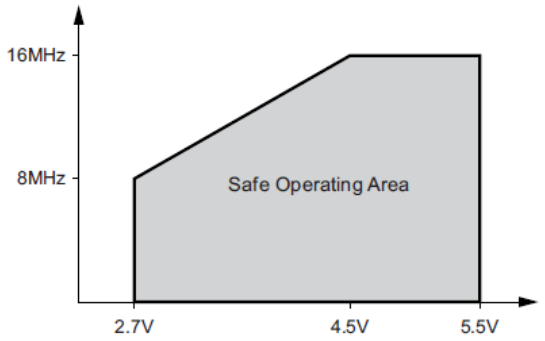
Para dar início a qualquer desenho de esquema elétrico que envolva um circuito integrado, é importante ter em mente os detalhes de funcionamento e recomendações dados pelo fabricante. Estas informações se encontram no *datasheet* do componente. Como mostrado na Figura 5, o microcontrolador, neste projeto, apenas recebe a alimentação, o sinal de *reset* e *clock* e envia sinais para fora da placa, além de realizar a comunicação com o exterior da placa.

Com relação à alimentação, deve-se cuidar para que a tensão esteja dentro da faixa de operação do componente, estipulada no *datasheet* de Microchip Technology Inc (2015). Segundo este documento, o microcontrolador deve estar alimentado entre 2.7V e 5.5V. No caso da escolha de tensão para a alimentação do microcontrolador, é necessário também

atentar para os outros elementos do circuito na placa. A bateria de polímero de lítio tem uma faixa de operação entre 2.4V e 4.2V e esta tensão deve ser regulada em um valor fixo para alimentar o microcontrolador, mantendo seu modo de operação sempre igual. As alimentações em 3.3V e 5V são compatíveis com o módulo de comunicação e com componentes que poderiam ser escolhidos como gerenciadores de alimentação e carregamento da bateria. No entanto, pela tensão da bateria exceder, por vezes, o valor de 3.3V, a escolha deste valor de tensão iria pressupor a utilização de um regulador que gerasse tensões acima e abaixo da sua tensão de alimentação (topologia buck-boost), o que iria adicionar complexidade ao projeto sem nenhum objetivo claro. Sendo assim, esta não seria uma escolha favorável, razão pela qual optou-se pela tensão de 5V. A regulação da tensão é explicada mais a fundo nas Seções 4.2.3 e 4.2.4.

Os *clocks* também são parte importante nesta escolha. Segundo o mesmo *datasheet*, o microcontrolador tem uma relação direta entre tensão de alimentação e frequência de *clock*. A Figura 6, extraída do *datasheet* do componente, ilustra esta relação.

Figura 6: *Relação entre frequência de clock (eixo vertical) e tensão de alimentação (eixo horizontal)*



Fonte: Microchip Technology Inc (2015)

Como o código para o microcontrolador foi desenvolvido na placa Arduino Mega, com um *clock* de 16MHz, utilizou-se esta mesma frequência para o projeto de *hardware*, podendo-se manter a mesma programação com relação aos temporizadores. Com isso, torna-se estritamente necessária a utilização de uma tensão de alimentação em 5V para o microcontrolador.

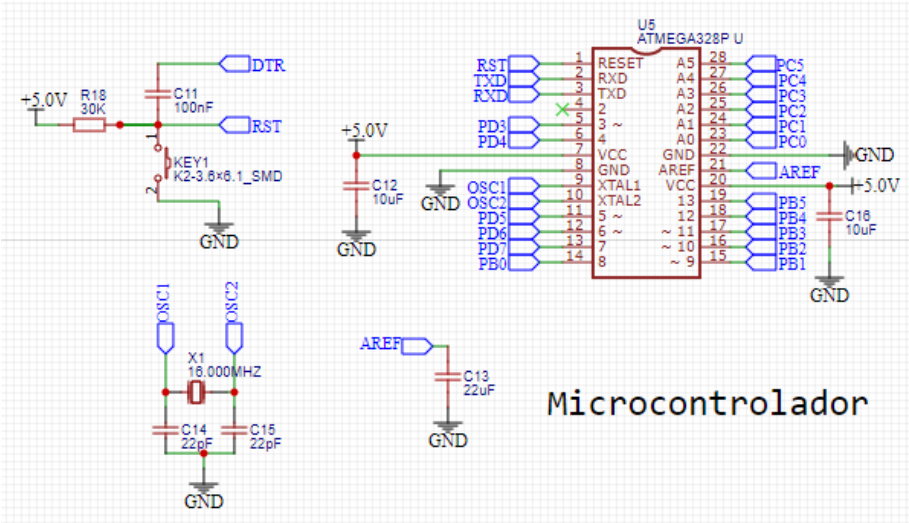
Para o circuito de *reset* foi escolhido um resistor de *pull-up* de 30 kΩ, já que o *datasheet* recomenda um valor de resistência entre 30 kΩ e 60 kΩ. É importante notar que, sendo assim, a lógica do *reset* é negada, ou seja, o mesmo é ativado com um sinal baixo. Além disso, o pino de *Data Terminal Ready* (DTR) que provém do módulo conversor USB/UART é o responsável por ativar o modo de programação do microcontrolador. Um capacitor conectado entre este pino e o pino de *reset* do microcontrolador garante que um nível baixo

no DTR provoque um nível baixo no *reset* rapidamente, até que o resistor de *pull-up* se encarregue de restaurar a tensão neste pino em valor alto.

Outro detalhe no circuito do microcontrolador é a presença de capacitores de desacoplamento. Segundo Microchip Technology Inc (2018) o microcontrolador consome corrente em picos curtos nas bordas do *clock* em utilização. Os capacitores de desacoplamento, então, devem suprir estes picos de corrente de maneira rápida. Posto de outra forma, o capacitor de desacoplamento na alimentação do microcontrolador deve ter capacitância suficiente para suprir a carga necessária, sem criar variações grandes em tensão, o que se traduz em uma indutância parasita baixa. Os valores de capacitância presentes nos circuitos de referência de Microchip Technology Inc (2015) são da ordem de, no máximo, unidades de μF. Colocando uma margem de carga, é razoável a utilização de capacitores de 10 μF no pino de entrada do microcontrolador.

O circuito de *clock* segue as recomendações de Microchip Technology Inc (2015) para os capacitores, e os cuidados de *layout* são detalhados na Seção 4.3.2. Com as informações descritas, o esquema elétrico final é o mostrado na Figura 7.

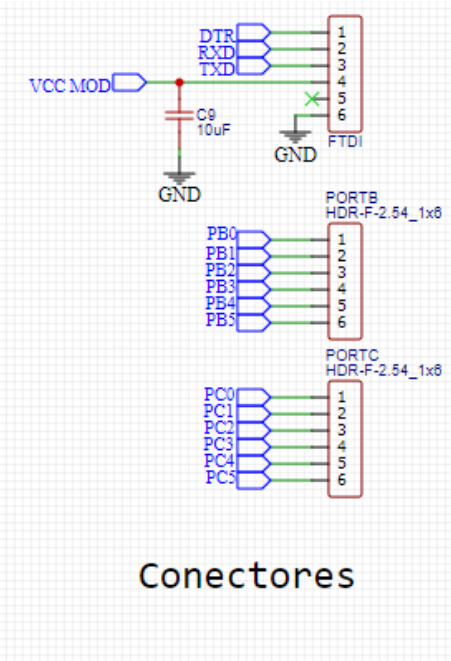
Figura 7: *Esquema elétrico do microcontrolador*



Fonte: O autor (2021)

Os pinos de sinais dados pelas portas do microcontrolador vão para conectores na placa, como mostrado anteriormente na Figura 5. Os pinos que são responsáveis pela comunicação com o módulo conversor USB/UART são TX (transmissão), RX (recepção) e DTR. Estes irão para o conector destinado a esta comunicação, juntamente com a alimentação externa e o seu terra. O circuito de conectores é ilustrado na Figura 8.

Figura 8: *Esquema elétrico dos conectores*



Fonte: O autor (2021)

4.2.2 Escolha da bateria e proteção

Com a escolha da bateria de polímero de lítio, cuidados com sua proteção devem ser tomados para que a operação do produto seja confiável, garantindo a segurança do usuário. A bateria escolhida segue a especificação de operação entre 4,2V (máxima tensão no carregamento) e 2,4V (mínima tensão em descarregamento).

Conforme mostrado em Data Power Technology Limited (2017) esta bateria contém um pequeno circuito de proteção contra sobretensão, subtensão e sobrecorrente. Para o caso de baterias em que não haja esta funcionalidade embarcada, decidiu-se projetar um circuito de proteção como provisão. As especificações do circuito de proteção serão as mesmas descritas em Data Power Technology Limited (2017), já que os valores são padronizados para o uso de baterias de polímero de lítio.

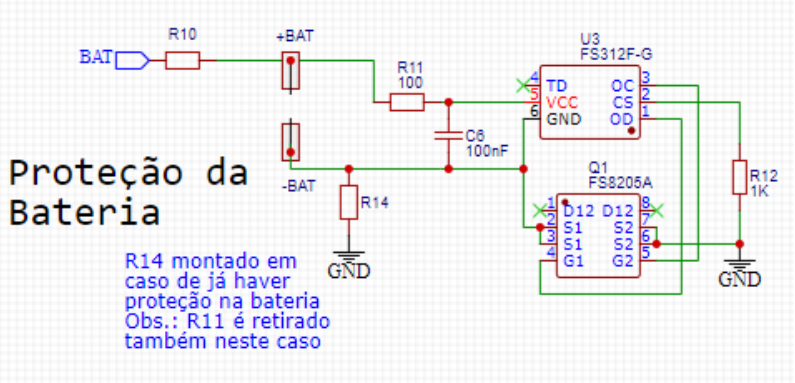
Com base em uma breve pesquisa conduzida visando funcionalidade e otimização de preço, chegou-se à escolha do componente FS312F-G da empresa Fortune Semiconductor Corporation. Em Fortune Semicoductor Corporation (2014) são descritos os detalhes de funcionamento do componente para o projeto deste bloco de circuito.

De maneira sucinta, este CI controla um par de transistores de efeito de campo (FET) utilizados como chaves, que atuam liberando ou cortando o fluxo de energia da bateria para o sistema, bem como na direção contrária. As tolerâncias deste circuito são, basicamente, as mesmas das descritas no *datasheet* da própria bateria. Em modo de carregamento da bateria, este circuito de proteção corta o fluxo de energia em caso de a tensão na mesma

exceder 4,25V. Após este corte, o CI apenas libera o carregamento se a tensão cair abaixo de 4,145V. Já no modo de descarregamento da bateria, o mesmo atua cortando a corrente de saída da bateria para tensões abaixo de 2,9V e libera novamente a partir do momento em que a tensão sobe para acima de 3V. Estas condições de tensão de descarregamento são mais restritivas do que o circuito montado na bateria e não acarretam perigo ou agravantes à funcionalidade do circuito. Adicionalmente, este circuito controla sobrecorrente na saída da bateria através do monitoramento da tensão no pino de *current sense* (CS). Se a tensão em CS estiver excedendo 0,15V por mais de 10ms, o circuito é cortado por sobrecorrente na bateria. Estas informações são as mesmas do circuito montado na bateria já mencionada. Por fim, um filtro passivo RC é conectado à entrada deste CI para suprimir *ripple* de tensão e distúrbios do circuito de carregamento.

Com estas informações, o circuito de proteção da bateria é o mostrado na Figura 9.

Figura 9: *Esquema elétrico do circuito de proteção da bateria*



Fonte: O autor (2021)

Observa-se que o par de transistores está contido em apenas um circuito integrado, designado na imagem como Q1. Os valores de R11, C6 e R12 seguem as recomendações do fabricante em Fortune Semicoductor Corporation (2014). Por fim, de maneira a prover um circuito versátil, que tenha a possibilidade de ser desabilitado em caso de baterias com a proteção já embarcada e também separar o circuito do resto do sistema para fim de testes, os resistores de 0Ω R10 e R14 foram previstos. Por exemplo, para teste, retira-se os resistores R10 e R14 e se faz possível a inserção de sinais e medidas neste bloco de circuito sem a interferência do restante do mesmo. Já para desabilitar a função de proteção, o resistor R11 não é montado, cortando a conexão com U3, e o negativo da bateria vai direto para o GND, através da montagem de R14. Em operação normal, com a proteção da bateria habilitada, monta-se R10 e R11 e não monta-se R14.

4.2.3 Gerenciamento das alimentações

Conforme discutido na Seção 4.1, existe uma necessidade de prover um caminho de energia, chamado *power path*, que varia entre ir da bateria para o circuito e ir da alimentação externa para a bateria, em modo de carregamento, ao mesmo tempo alimentando o resto do circuito da placa. O componente encontrado que faz o carregamento da bateria e tem a funcionalidade de *power path* foi o BQ24074, da empresa Texas Instruments. Segundo o *datasheet* do componente (TEXAS INSTRUMENTS INC., 2019a) ele está de acordo com os padrões USB de corrente e tensão, além de ser ideal para baterias na faixa de tensão usada neste projeto. Sua tensão de saída é regulada em 4.4V quando há uma alimentação externa (originária do USB) e, quando esta alimentação não está conectada, a saída é igual à tensão proveniente da bateria.

O pino de *power good* com lógica negada ($\overline{PGOOD}$) é colocado em nível baixo para o caso de haver uma fonte externa alimentando este CI. O pino de estado de carga ($\overline{CHG}$), por sua vez, é colocado em nível baixo caso a bateria esteja carregando. Com isso, é possível montar diodos emissores de luz (LED) para sinalizar estes possíveis estados. Os LEDs serão colocados em um local exterior à placa de modo que na placa apenas se fixarão seus conectores. Com os LEDs populares de 20 mA de corrente máxima e em torno de 1,8V de tensão (dependendo da cor este valor muda), é possível fazer o cálculo da resistência a ser colocada em série para definir a corrente. Considerando o *pull-up* dos LEDs para a tensão de saída do carregador, que é máxima em 4.4V, e uma corrente de operação em torno da metade da máxima de um LED, que é de 20 mA, resistores de valor comercial em 330 Ω são uma boa escolha para ambos os LEDs.

Os pinos EN1 e EN2 (*enable*) são utilizados para configurar o modo de operação da corrente limite de alimentação do CI. Os modos de operação na corrente máxima de entrada estão descritos da Tabela 1.

Tabela 1: *Configurações dos pinos EN1 e EN2.*

EN2	EN1	Corrente máxima no pino de alimentação (IN)
0	0	100 mA. Modo USB100
0	1	500 mA. Modo USB500
1	0	Configurado por resistor externo em ILIM
1	1	Modo de *Standby*

Fonte: Texas Instruments Inc. (2019a)

Para haver um certo controle sobre estes modos de operação, foi decidido utilizar os pinos EN1 e EN2 como saídas de pinos do microcontrolador. A ideia inicial é que o circuito opere no modo 2 (EN2=1 e EN1=0), permitindo uma corrente maior e suficiente para o consumo no resto do circuito. Com os dados de *datasheet* (TEXAS INSTRUMENTS INC., 2019a) um resistor de 1,5 kΩ conectado no pino de ILIM resulta em uma corrente limite de 1 A, sendo suficiente para suprir a potência máxima (com máximos 200 mA) de

consumo do microcontrolador e com uma folga de provisão para outros componentes que possam vir a ser utilizados no projeto futuramente.

A corrente de carregamento da bateria é estipulada em sua documentação (DATA POWER TECHNOLOGY LIMITED, 2017) como sendo 200mA. É necessário, com esta informação, configurar o carregador para a aplicação desta corrente de carregamento. Esta configuração é feita através do pino ISET, que, com um resistor de 4,87 kΩ, entrega a corrente estipulada. A tensão através deste resistor pode ser utilizada para medir a corrente de carregamento com a aplicação da lei de Ohm.

Outra corrente que deve ser configurada é a que determina o fim da carga na bateria, através de um resistor no pino ITERM. Novamente com base em Data Power Technology Limited (2017), é determinado que esta corrente de terminação de carregamento seja de 10 mA. O resistor que deve ser colocado resulta em torno de 1,5 kΩ. É possível, também, manter o pino de ITERM sem conexão. Neste caso, a corrente que indica o fim do carregamento é 10% da corrente nominal que, portanto, seria em torno de 20 mA, trazendo uma pequena margem de segurança. Foi decidido, com isso, manter este pino em aberto.

Por fim, o pino de TS faz a medição de temperatura através de um termistor de coeficiente de temperatura negativa (NTC). No caso desta aplicação, será colocado um resistor de 10 kΩ, já que a proteção da bateria se encarrega de protegê-la em casos de sobrecorrente e, considerando o uso do produto, as temperaturas típicas são muito menores do que os limites dos componentes em geral, que ficam, no pior dos casos, em torno de 85°C.

Com as observações e análises de funcionamento deste circuito fundamentadas, o esquemático resultante é o da Figura 10. O pino de CE# apenas habilita, com um sinal baixo, o carregamento da bateria. Com isso, é interessante controlá-lo através do microcontrolador. Os valores de capacitância de desacoplamento para os pinos deste CI estão de acordo com especificado em Texas Instruments Inc. (2019a).

4.2.4 Projeto dos conversores boost

Como detalhado na Seção 1, os reguladores *boost* são os responsáveis por elevar a tensão de uma faixa de 2.4V a 4.4V para os 5V necessários à operação do resto do circuito. A fim de suprir as necessidades desta parte do projeto, três possibilidades se mostraram atraentes no mercado. Estas são descritas abaixo.

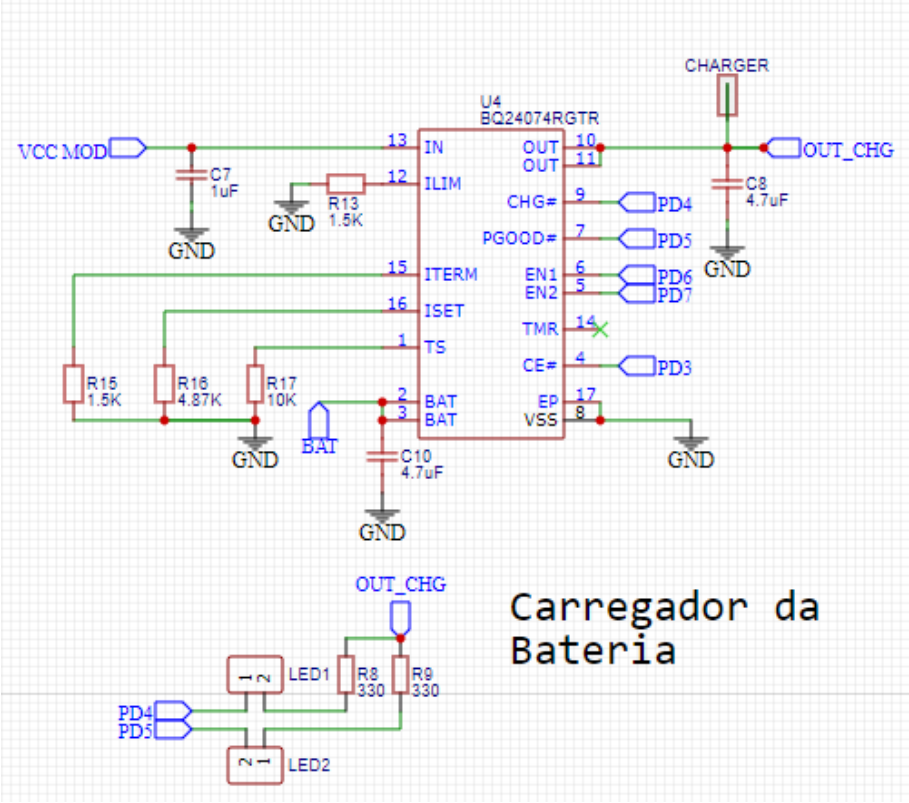
1. Controlador: Esta opção mantém os transistores de chaveamento do circuito não integrados ao componente, bem como o indutor e os capacitores de saída. Sendo assim, o CI se encarrega apenas de controlar o chaveamento dos transistores, permitindo uma grande liberdade para o projetista otimizar seu *design* de regulador.

2. Conversor: O conversor integra a si apenas os transistores de chaveamento, mantendo ainda o indutor à escolha do projetista. Sendo assim, algumas limitações do regulador

são impostas e sua eficiência é otimizada apenas dentro de um certo intervalo de cargas, dado que os transistores de chaveamento estão fixados e possuem restrições de dissipação no *package* do conversor. Por outro lado, o esforço de projeto é diminuído, bem como os riscos associados ao *design*.

3. Módulo: Neste caso, o indutor também é integrado ao componente, fazendo com que poucas escolhas restem ao projetista, e o *design*, muitas vezes, fique bastante restrito e pouco flexível. Uma vantagem nesta opção é a otimização do espaço, que pode ter sua ocupação diminuída muito por conta da integração do indutor, componente que geralmente tem um tamanho considerável. Nesta opção de projeto as chances de falha são menores, já que estas estão quase que inteiramente atribuídas ao próprio fabricante do CI.

Figura 10: *Esquema elétrico do circuito de carregamento da bateria e gerenciamento de tensões e corrente*



Fonte: O autor (2021)

Levando em consideração os fatores descritos e as necessidades do projeto, escolheu-se trabalhar com a solução em conversores. A empresa Texas Instruments possui uma linha extensa de reguladores e, por conta da experiência positiva com esta empresa, a boa documentação acessível ao projetista e a possibilidade de simulação com seu *software*

(Webench Power Designer) que leva em conta o modelo do componente, escolheu-se trabalhar com dois reguladores desta empresa: o TPS61022 e o TPS61222.

O projeto de dois reguladores permite uma certa flexibilidade. Optou-se trabalhar desta forma, pois é possível deixar como provisão um regulador com capacidade mais alta de corrente, em caso de futuro uso de energia por outros componentes alocados na placa ou nas portas do microcontrolador. Além disso, dois reguladores garantem uma segurança maior ao projeto em caso de eventual falha de um deles. O espaço na placa também não é muito afetado por esta escolha. Vale salientar, porém, que para fins de uso em mercado, este *design* possivelmente seria otimizado, em algum momento, para apenas um regulador, diminuindo custos e tornando o projeto mais enxuto.

A seguir serão descritos os projetos individuais de cada um dos dois reguladores escolhidos.

4.2.4.1 Regulador de baixa corrente

Para o projeto do regulador de corrente menor, o CI utilizado é o TPS61222. Através de uma consulta ao *datasheet* do microcontrolador, constata-se que seu consumo com um *clock* de 16 MHz é de máximos 14 mA. Considerando uma folga, o projeto do regulador na ferramenta de simulação Webench utilizou como corrente de saída um valor de 50 mA. A Figura 6 mostra que a alimentação do microcontrolador deve estar dentro do intervalo de 4,5 V e 5,5 V, ou seja, há uma tolerância de 10% para variações em relação à tensão nominal. A Tabela 2 mostra os parâmetros utilizados no projeto deste regulador.

Tabela 2: *Parâmetros de projeto do regulador de baixa corrente*

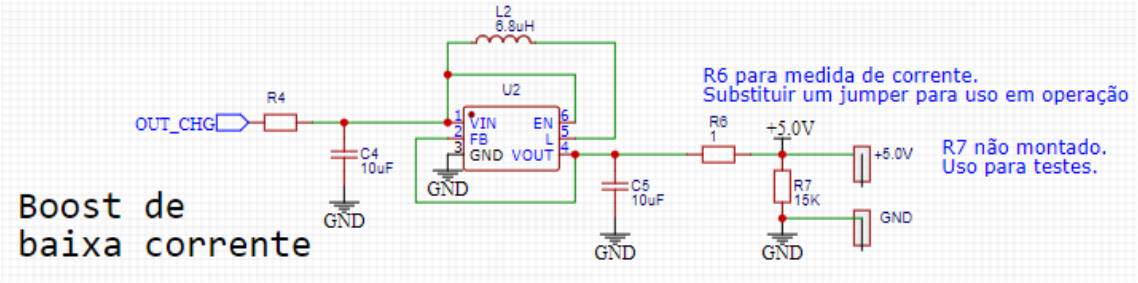
Tensão de entrada	2,5V a 4,5V
Tensão de saída	5,0V
Corrente de saída	50mA
Tolerância	10%

Fonte: O autor (2021)

Com estas entradas, o projeto foi concebido na ferramenta Webench e o esquema elétrico final é o mostrado na Figura 11.

A Tabela 3 mostra os principais componentes utilizados no projeto do regulador e suas especificações. Observa-se a presença dos componentes R4, R5 e R7 no esquemático do regulador. Os resistores R4 e R5 tem a função de prover uma maneira fácil de desacoplar esta parte do circuito do resto. Com isso, este regulador pode ser testado sem sofrer a interferência do restante do circuito na placa. Além disso, como há dois reguladores projetados na placa, apenas um deles terá seus resistores de entrada e saída soldados para fornecer a alimentação do restante do circuito. O resistor R5, no esquemático, possui uma resistência pequena de 1 Ω, permitindo a medição de corrente diretamente pela sua tensão. Com isso, é possível saber o consumo de corrente do microcontrolador e avaliar

Figura 11: *Esquema elétrico do regulador de tensão de baixa corrente*



Fonte: O autor (2021)

a necessidade do regulador de baixa corrente ou o de alta corrente. Com a placa em funcionamento normal, este resistor deve ser de 0 Ω. O resistor R7 serve de carga resistiva para testes e seu valor pode ser mudado de acordo com as necessidades de teste. Este mesmo resistor, portanto, não é montado no caso de a placa estar em funcionamento normal. Maiores informações a respeito dos componentes podem ser encontradas nos *websites* dos fabricantes.

Tabela 3: *Principais componentes presentes no projetos do regulador de baixa corrente*

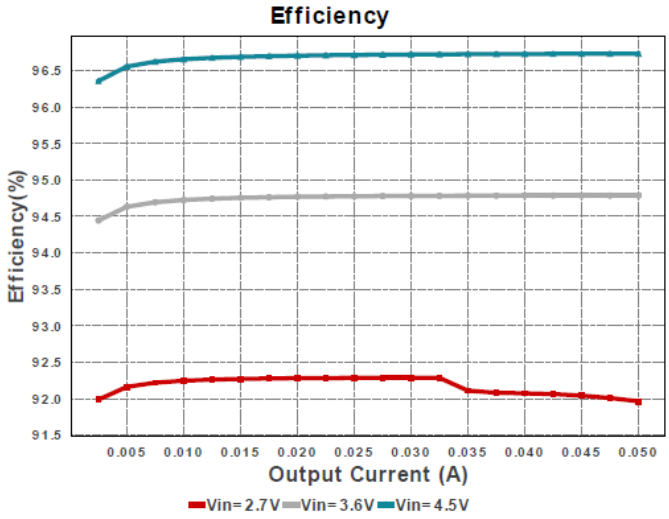
Fabricante	***Part Number***	**Designador**	**Breve descrição**
Texas Instruments Inc.	TPS61222DCKT	U2	Conversor *boost* tensão de 5V e corrente máxima de 200mA
TDK Electronics	VLCF5020T-6R8N1R1	L2	Indutor de 6.8 µH e corrente máxima de 1,11 A para uma queda em 30% da indutância nominal
Murata Manufacturing Co., Ltd.	GRM21BR61H106KE43L	C4 e C5	Capacitor 0805 de 10 µF e tensão máxima DC de 50 V

Fonte: O autor (2021)

Com o auxílio da ferramenta Webench foi levantada a curva de eficiência do regulador, mostrada na Figura 12.

Nota-se que três curvas foram geradas, cada uma diz respeito a condições diferentes de tensão na entrada do regulador. A eficiência do regulador é acima de 90% e isto representa um valor satisfatório, já que o sistema opera com uma bateria, ou seja, o critério de eficiência é de grande importância. Uma boa eficiência também contribui para o não aquecimento do componente por conta das baixas perdas de energia em dissipação térmica.

Figura 12: *Curva de eficiência do regulador de baixa corrente para um intervalo de correntes de carga*



Fonte: *Report* da ferramenta Webench (2021)

O relatório gerado pela ferramenta Webench também inclui uma série de informações relevantes, algumas delas estão descritas na Tabela 4.

Tabela 4: *Parâmetros resultantes da simulação do regulador de baixa corrente*

Tensão de saída	5,0V
Tensão de *ripple* pico a pico	5,38mV
Tolerância	3,6%

Fonte: O autor (2021)

Percebe-se que os parâmetros estão dentro do estipulado como requisito. Por exemplo, a saída deve estar entre ±10% da tensão nominal de 5V e, considerando os três itens descritos na Tabela 4, o projeto se mostra adequado à especificação.

4.2.4.2 Regulador de alta corrente

Os requisitos de projeto para o regulador de alta corrente seguem, basicamente, as mesmas especificações do regulador já descrito na Seção 4.2.4.1. O único parâmetro de exceção, naturalmente, é a corrente. Considerando que componentes poderão ser dispostos nas portas do microcontrolador, aumentando sua necessidade de corrente no pino de alimentação, é interessante que o projeto deste regulador disponha tal corrente máxima de consumo. Segundo o *datasheet* do microcontrolador (MICROCHIP TECHNOLOGY INC, 2015), o valor absoluto máximo de corrente de alimentação, considerando a alocação de componentes de consumo em suas portas, é de 200mA. Agora, considerando que,

futuramente, componentes de maior consumo podem ser alimentados diretamente por este regulador, é mais prudente ainda que uma estimativa em torno de 1A seja razoável para o projeto. É necessário, também, atentar-se ao limite de descarga da corrente da bateria, que pode regular normalmente em torno de 1A sem danos. Dadas estas necessidades, o regulador escolhido foi o TPS61022, da empresa Texas Instruments. Este componente supre até 8 A de corrente, um valor bastante alto e com folga para o caso em questão. A Tabela 5 mostra os parâmetros utilizados no projeto deste regulador.

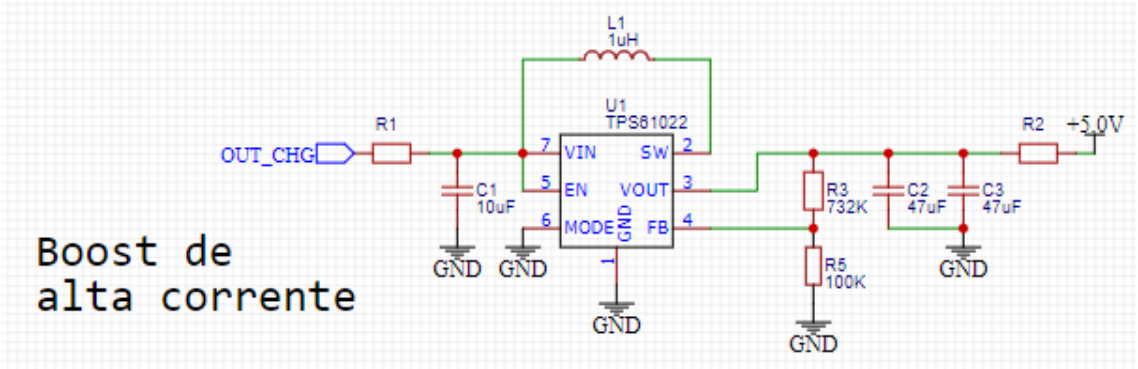
Tabela 5: *Parâmetros de projeto do regulador de alta corrente*

Tensão de entrada	2,5V a 4,5V
Tensão de saída	5,0V
Corrente de saída	1A
Tolerância	10%

Fonte: O autor (2021)

Com estes requisitos, foi desenhado o projeto na ferramenta Webench, resultando no esquema elétrico final mostrado na Figura 13.

Figura 13: *Esquema elétrico do regulador de tensão de alta corrente*



Fonte: O autor (2021)

A Tabela 6 mostra os principais componentes utilizados no projeto do regulador e suas especificações.

Os resistores R3 e R5 formam a malha de realimentação do circuito, mantendo a saída em 5.0V, já que a tensão no pino de FB é de 0,6V. Da mesma forma como no projeto do regulador para corrente baixa, os resistores R1 e R3 proveem uma maneira de desacoplar este regulador do restante do circuito, permitindo testes e a escolha entre um dos reguladores na alimentação do microcontrolador. O resistor R2 pode ser substituído por um resistor de 1 Ω para fins de medida de corrente de saída.

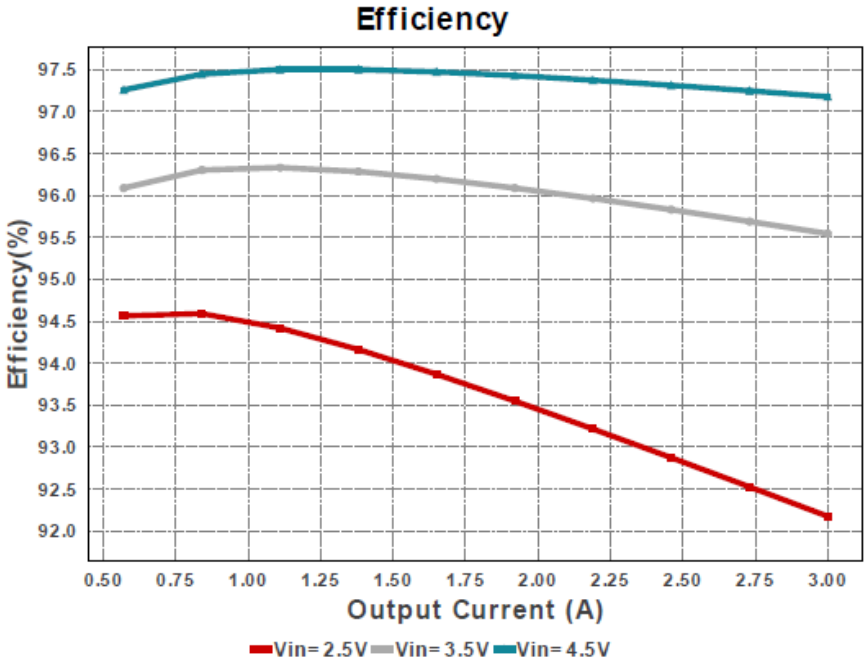
Tabela 6: *Principais componentes presentes no projetos do regulador de alta corrente*

Fabricante	***Part Number***	**Designador**	**Breve descrição**
Texas Instruments Inc.	TPS61022	U1	Conversor *boost* tensão de 5V e corrente máxima de 8A
TDK Electronics	SPM6530T-1R0M120	L1	Indutor de 1 µH e corrente de 13 A para um aumento de temperatura de 40 °C
Murata Manufacturing Co., Ltd.	GRM21BR61H106KE43L	C2 e C3	Capacitor 1206 de 47 µF e tensão máxima DC de 10 V

Fonte: O autor (2021)

A curva de eficiência é detalhada na Figura 14 para as correntes estipuladas de projeto, considerando uma folga para o caso de componentes de maior consumo serem alocados nesta tensão futuramente. Novamente, para todo o intervalo de possíveis tensões de entrada, a eficiência está acima de 90%.

Figura 14: *Curva de eficiência do regulador de alta corrente para um intervalo de correntes de carga*



Fonte: *Report* da ferramenta Webench (2021)

Na Tabela 7 estão as informações geradas pelo Webench para o projeto deste regulador.

Tabela 7: *Parâmetros resultantes da simulação do regulador de alta corrente*

Tensão de saída	4,99V
Tensão de *ripple* pico a pico	46,8mV
Tolerância	3,6%

Fonte: O autor (2021)

Apesar da tensão de *ripple* estar quase 10 vezes mais alta do que a simulada para o outro regulador, esta ainda se mantém dentro do tolerável do microcontrolador.

4.2.5 Módulo para comunicação UART/USB

A solução para a conversão entre UART e USB será feita através do uso de um módulo separado à placa, diminuindo o esforço de desenvolvimento neste momento e garantindo funcionalidade. O custo desta opção sempre deve ser levado em conta em projetos de larga produção, porém, no caso deste estudo, assume-se justificável tal escolha por permitir agilidade e segurança ao projeto como um todo. O módulo utilizado será o do fabricante Future Technology Devices International Ltd.(FTDI), que integra o CI FT232R.

4.3 Layout e cuidados

4.3.1 Conversores boost

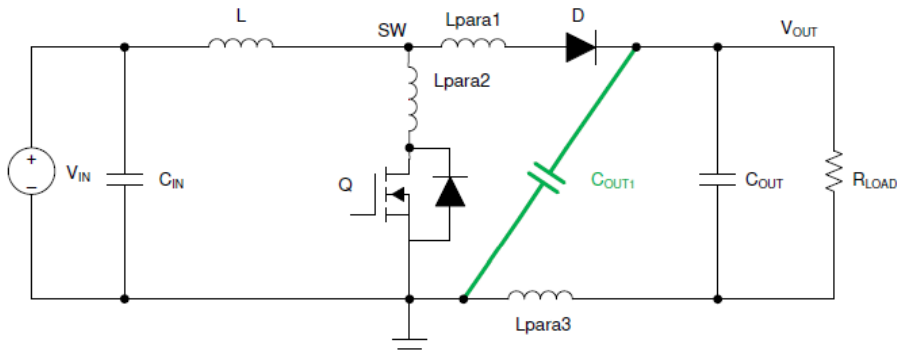
Utilizando Texas Instruments Inc. (2020) como referência e os *datasheets* dos circuitos integrados TPS61022 e TPS61222 da empresa Texas Instruments, foi dado início ao processo de *layout*.

Muitas das imperfeições e não idealidades de um circuito *boost* podem ser explicadas através dos componentes parasitas presentes no circuito, que podem ser provenientes de diversas fontes. A trilha da PCB é um elemento que traz, comumente, impedâncias parasitas na forma de indutância, fazendo com que o circuito do conversor seja aproximado pelo diagrama que mostra a Figura 15.

De acordo com a mesma lei da tensão no indutor, descrita na Equação 1, estes parasitas provocam picos de tensão, podendo causar danos de sobretensão no MOSFET. Além disso, o circuito sujeito a estas imperfeições emite mais radiação eletromagnética, podendo ser fonte de interferência (EMI). Uma saída para contornar este problema é simplesmente posicionar os componentes próximos uns dos outros, diminuindo o comprimento das trilhas de menor largura e, com isso, reduzindo os efeitos decorrentes de indutâncias parasitas. Outra boa prática neste tipo de *layout* é tornar as trilhas de cobre o mais espessas possíveis.

A malha de realimentação dos circuitos *boost* é bastante suscetível a ruído. Diminuir ao máximo a distância entre o pino de realimentação e os divisores resistivos fará com que

Figura 15: *Componentes parasitas presentes em circuitos elevadores de tensão*



Fonte: Texas Instruments Inc. (2020)

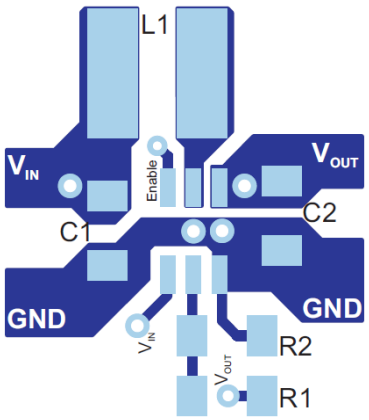
este pino não tenha ruído acoplado pelo pino de SW.

As subseções 4.3.1.1 e 4.3.1.2 a seguir fazem um comparativo entre o que é recomendado pelo fabricante em *datasheet* e o que foi implementado, com base nos conceitos descritos nesta Seção e no projeto resultante.

4.3.1.1 Conversor boost de baixa corrente

A imagem da Figura 16 ilustra a recomendação do fabricante com respeito ao layout do conversor *boost* de baixa corrente. Verifica-se a total aderência dos conceitos de proximidade de componentes com pinos e largura de trilha, reduzindo os efeitos decorrentes da presença de parasitas.

Figura 16: *Layout recomendado pelo fabricante para o conversor boost de corrente baixa*

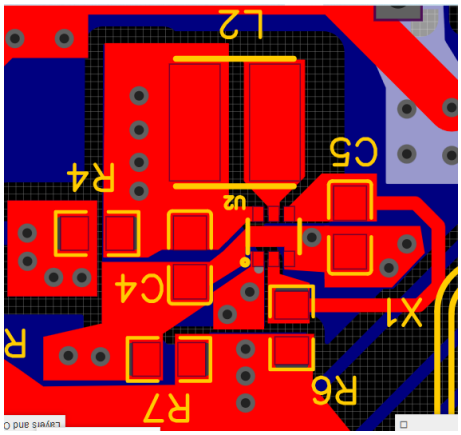


Fonte: Texas Instruments Inc. (2014)

A Figura 17 mostra o *layout*, de fato, implementado no projeto. As anotações de

serigrafia que dizem respeito às designações dos componentes no layout são de menor importância, já que estão de acordo com o esquemático como um todo. Optou-se por apresentar a ilustração de modo que o conversor *boost* esteja na mesma orientação que o próprio se encontra na Figura 16, permitindo uma comparação visual mais direta.

Figura 17: *Layout do conversor boost de corrente baixa implementado no projeto*



Fonte: O autor (2021)

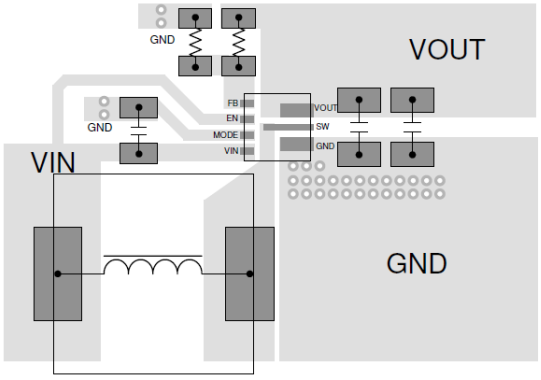
É necessário salientar as diferenças em nível de esquemático. Enquanto que na Figura 16 os resistores de realimentação estão presentes, os mesmos não foram necessários no projeto implementado, já que o componente TPS61222 é, por definição, configurado para 5V se o pino de realimentação for conectado diretamente à saída. Além disso, o pino de *enable* está, no caso do projeto implementado, conectado ao VCC a todo momento.

4.3.1.2 Conversor boost de alta corrente

A Figura 18 mostra um *layout* de referência dado pelo fabricante do conversor. Nota-se que o mesmo decidiu por um *footprint* um tanto diferente ao colocar o *pad* do pino SW mais alongado, permitindo fazer o roteamento deste sinal por dentro do próprio *footprint*. Dessa forma, é possível manter o indutor próximo ao CI, mesmo com um *package* muito pequeno, de máximos 2.1 mm de largura e comprimento.

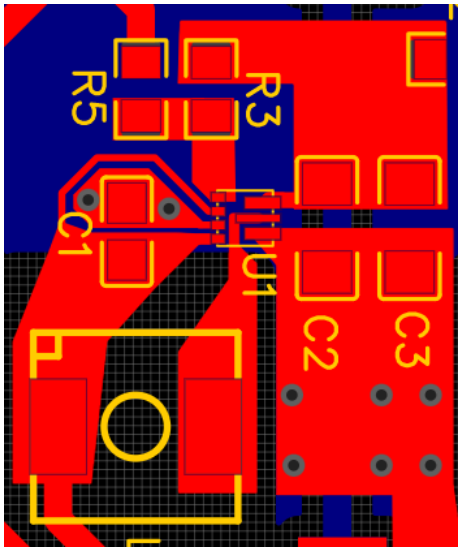
Na Figura 19 está o *layout* implementado no circuito *boost* de corrente mais alta. Nota-se que foi possível manter o *layout* deste *boost* em uma forma muito semelhante à recomendada, tornando o projeto mais confiável com relação às imperfeições e efeitos indesejáveis explicados anteriormente, decorrente de parasitas. Há diferenças no tamanho dos capacitores de saída e no indutor, que no circuito implementado são levemente maiores, em decorrência de necessidades de projeto no nível de esquemático.

Figura 18: *Layout recomendado pelo fabricante para o conversor boost de corrente alta*



Fonte: Texas Instruments Inc. (2021)

Figura 19: *Layout do conversor boost de corrente alta implementado no projeto*



Fonte: O autor (2021)

4.3.2 Detalhamentos gerais

Para o *layout* da placa como um todo, alguns cuidados foram tomados. Considerando as trilhas de sinais e de alimentação de forma diferente, foram estabelecidas algumas regras para garantir um bom fluxo de corrente, sem o aparecimento de parasitas indutivos ou quedas de tensão por resistência de trilha.

Com base no padrão IPC (2012), a distância mínima entre trilhas (*clearance*) com menos de 15V de diferença deve ser de 5.1 mils (equivalente a aproximadamente 0.13 mm) (PETERSON, 2020). Este foi o valor estipulado como regra de *clearance*. Sendo assim, trilhas presentes na placa não devem aproximar-se entre si de um valor menor do que 5.1 mils.

Outro parâmetro importante é a largura das trilhas. Para uma densidade de cobre de 1 oz/ft^2 (equivalente a aproximadamente 305 g/mm^2), um comprimento máximo de 3.1 in e uma corrente máxima de 40 mA (para trilhas de sinal), a largura mínima da trilha resulta em 0.14 mil, causando uma queda máxima de tensão de 0.4V no comprimento máximo de trilha indicado. Para as trilhas de alimentação, considerou-se a corrente máxima em 1.2A, resultando em uma largura de trilha mínima de 15.6 mil. Estes valores foram extraídos de Bittele Eletronics (2020). Na prática, se há espaço para aumentar o tamanho das trilhas, é uma boa opção fazê-lo, já que, com isso, está se otimizando o projeto em termos de mitigação de imperfeições e parasitas. Sendo assim, todo o cuidado foi tomado para que as larguras das trilhas não ficassem menores do que as estipuladas por estes cálculos.

Observando o projeto em nível de esquemático, nota-se um fluxo do sinal desde sua entrada no conversor USB/UART até seu processamento no microcontrolador. Os blocos de circuitos que compõem este caminho foram alocados na placa de forma próxima, minimizando longas trilhas e tornando o *layout* mais intuitivo visualmente. Com isso, a colocação dos componentes na placa foi feita e o roteamento dos sinais e das alimentações foi iniciado. Dada a dificuldade de roteamento proveniente do alto número de trilhas de sinais em relação ao tamanho da placa, este parâmetro, juntamente com o número de camadas de circuito, foi escolhido. Optou-se por escolher o melhor balanço entre custo e densidade da placa, na forma de tamanho final e número de camadas. Por fim, decidiu-se por uma placa de duas camadas (*top* e *bottom*) em um comprimento de 90 mm e largura de 50 mm. Com este tamanho, a placa permite a alocação de todos os componentes em apenas uma face sendo, além disso, pequena suficiente para ser funcional enquanto produto, avaliando sua futura anexação em uma plataforma mecânica.

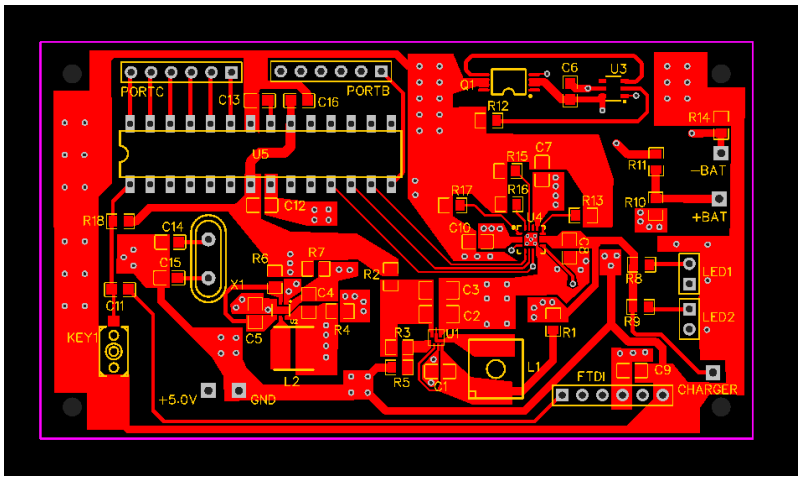
Outro cuidado importante tomado no momento de execução do *layout*, foi o estabelecimento de planos grandes de GND, já que o fluxo de corrente nesta rede não deve criar quedas de tensão. O GND é um ponto muito sensível do circuito, pois todos componentes estão ligados a ele e o tem como referência de tensão. Para que esta referência estivesse sempre próxima a todos os componentes da placa, a camada de *bottom* foi utilizada majoritariamente como plano de GND, permitindo que boa parte da área da placa em *top*

pudesse ser conectado ao GND através de vias. Além do GND, as trilhas de alimentação são, também, de extrema importância. Com a finalidade de tornar largos os caminhos de corrente nas alimentações, foram colocados polígonos nas ligações entre componentes, compartilhando a mesma *net* de alimentação.

Sinais com frequência um pouco mais alta (na casa dos MHz) mercem um cuidado mínimo para que não interfiram nas trilhas vizinhas. Decidiu-se eliminar o GND de *bottom* nas regiões próximas aos pinos de SW dos conversores boost, de modo a mitigar interferências.

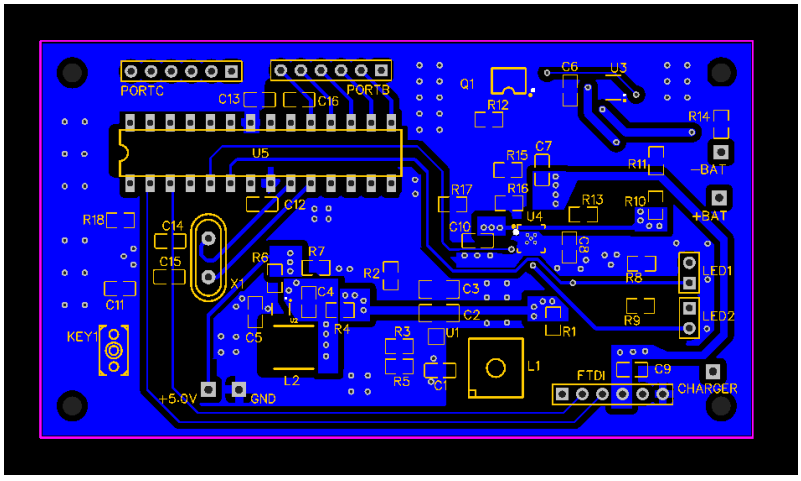
Ao final do processo de *layout* o resultado foi obtido como mostram as Figuras 20 e 21.

Figura 20: *Layout final da placa, camada de cima (top)*



Fonte: O autor (2021)

Figura 21: *Layout final da placa, camada de baixo (bottom)*



Fonte: O autor (2021)

5 PROGRAMAÇÃO DO MICROCONTROLADOR

5.1 Requisitos de programa

Utilizando-se as informações presentes na Seção 3.2, é possível mapear as necessidades do produto em requisitos de *firmware*. Uma observação importante de se fazer é que o objetivo do microcontrolador na placa não é o de salvar todos os dados de consultas do paciente, ou mesmo ter um histórico extenso de uso de medicamentos para os tratamentos. Este tipo de informação é mais adequada ao servidor que irá se comunicar com a caixa inteligente, já que o mesmo dispõe de um banco de dados, com tamanho suficiente, dedicado à esta tarefa. Foi com esse viés que anteriormente, em nível de projeto em *hardware*, decidiu-se por manter apenas as memórias não-voláteis e voláteis já disponíveis do *chip*.

Feita esta observação, a seguir são dispostas as informações que deverão estar obrigatoriamente alocadas na caixa e suas justificativas.

Nomes dos remédios

No momento em que a caixa é conectada, tanto na consulta médica quanto na farmácia, o profissional que a está utilizando deve dispor rapidamente de quais são os remédios que estão alocados na caixa (no caso do médico) ou quais os remédios que devem ser alocados na caixa (no caso do farmacêutico).

Horários de uso de cada medicamento

Esta informação é crucial para que a própria caixa dispare, no momento correto, o alarme avisando o usuário de que deve tomar seu medicamento.

Número de unidades do medicamento em cada compartimento

Com esta informação sempre atualizada, é possível manter o controle de quantas unidades foram consumidas desde a ida à farmácia e quantas ainda existem dentro da caixa, sem a necessidade de contar manualmente.

Nome do usuário

Para tornar confiável ao médico que o paciente está com a sua caixa e que ela não foi trocada, ou mesmo pela segurança do paciente, é interessante ter esta informação

alocada na caixa, permitindo, assim, uma checagem com a informação presente também no sistema.

Número de série do produto

Este número seria programado durante a fabricação e nunca trocado, garantindo o fator único de cada unidade produzida.

Quais compartimentos devem ser preenchidos pelo farmacêutico

No momento da consulta ao médico, a caixa será configurada para receber medicamentos nos compartimentos corretos, estejam estes vazios ou semi preenchidos. Neste último caso, naturalmente, o remédio a ser preenchido deve ser o mesmo que já possui unidades alocadas no compartimento em questão. Para que o farmacêutico saiba onde dispensar os remédios, a caixa irá, então, guardar esta informação e mostrá-la no momento em que o farmacêutico conecta a caixa ao seu computador.

Data e hora

A hora é claramente necessária para que o funcionamento da caixa esteja alinhado com a hora oficial do local e emita o alarme de uso de medicamento no momento correto. A data é necessária apenas para manter controle da passagem dos dias do tratamento, checando-as com os dados já existentes no servidor, externo ao produto.

Todas estas informações devem ser comunicáveis ao servidor, ou seja, deve sempre haver uma maneira de extrair estas informações, atualizadas, da caixa de remédios.

Uma situação importante de ser levada em conta é o caso em que a caixa se desligue, seja por falta de carga na bateria ou por alguma situação de mal contato entre os pinos de alimentação e o circuito. Neste caso, as informações presentes na caixa não devem ser perdidas. Sendo assim, se faz necessário o uso da memória não-volátil presente no chip (EPROM).

5.2 Solução proposta

A solução pensada é resultado de uma avaliação do que é necessário na comunicação entre a placa e o sistema. Por sistema compreende-se a plataforma em que ocorre a navegação na rede. Com os requisitos levantados na Seção 5.1 foi montada uma sequência de regras para codificar o que cada bloco de informação significa na transmissão de um dispositivo para o outro.

A conexão é checada pela caixa enviando, a cada 15 segundos, o símbolo # para o sistema. Se o sistema recebe este símbolo ele deve responder com o próprio símbolo novamente, confirmando que há conexão entre as duas partes. Com exceção desta regra, a caixa de remédios é reativa, ou seja, apenas envia informações para o sistema se for requisitada para tal ação.

A Tabela 8 mostra o conjunto de regras criadas para interpretação das informações passadas entre o sistema e a caixa. Cada quadrado da coluna de blocos de informação corresponde a um conjunto de 8 bits (1 *byte*). O primeiro item de cada conjunto de blocos sempre será uma letra do alfabeto, codificada em ASCII. Esta letra é um cabeçalho que permite tanto ao sistema quanto à caixa prepararem-se para o recebimento dos próximos blocos.

Tabela 8: *Regras definidas para a comunicação entre a caixa e o sistema*

Blocos de informação	Resposta da caixa	Definição
c	c	Estabelecimento da conexão
#	Nenhuma	Manutenção da conexão
h n_1 n_2 $\cdots$ n_5 n_6	h	Sincronização de data e hora
r n_1 $\cdots$ n_6 x_1 $\cdots$ x_{n_6}	r	Configuração de novo remédio
m	Matriz de horários	Requisição de matriz de horários
n	Nomes dos medicamentos	Requisição de nomes dos medicamentos
a n_1	a	Apagar coluna da matriz de horários
e n_1 n_2 n_3	e	Configurar horários de uso manualmente
o n_1 x_1 $\cdots$ x_{n_1}	o	Configurar ID de usuário
q	ID de usuário	Requisição de ID de usuário

Continua na próxima página

Tabela 8 – Continuação da página anterior

Blocos de informação	Resposta da caixa	Definição
f	Informações de farmacêutico	Requisição de informações para farmacêutico
g n_1 n_2	g	Alocação física de remédio
d	Informações de médico	Requisição de informações para médico
j	Quantidades ainda disponíveis nos compartimentos	Requisição de número de unidades disponíveis em cada compartimento
p	Número de unidades que deve ser ingerida até o final do período	Requisição de número de unidades a ser ingerida até o final do período
s	Número de série do produto	Requisição de número de série do produto
z	Número da última consulta	Requisição de número da última consulta
b n_1 $\cdots$ n_{20}	b	Configuração de número da última consulta

Fonte: O autor (2021)

Estabelecimento da conexão Ao receber a letra c minúscula, a caixa entende que a conexão foi estabelecida pela primeira vez. Com isto, a letra minúscula c é enviada de volta ao sistema, indicando que a caixa está pronta para iniciar a comunicação.

Manutenção da conexão Se, em um intervalo maior do que 20 segundos, este símbolo não for recebido pela caixa, a mesma irá encerrar a comunicação, assumindo que o

sistema não está conectado ou não está em operação. Este símbolo serve para checar a manutenção da conexão entre sistema e caixa.

Sincronização de data e hora Toda vez que a caixa é conectada existe a necessidade de sincronização por data e hora. Esse procedimento é importante, já que, após um certo período de tempo, é possível que o relógio da caixa fuja de sincronismo com a hora oficial, causando erros nos momentos em que os alertas de uso de medicamentos devem ser feitos. Os *bytes* de n_1 a n_6 significam, respectivamente: hora, minuto, segundo, dia, mês e ano.

Configuração de novo remédio Este bloco de *bytes* determina o *slot* em que o remédio será alocado (n_1), a hora e minuto em que o remédio deve ser ingerido (n_2 e n_3 respectivamente), quantas vezes ao dia o paciente deve medicar-se deste remédio (n_4), quantas unidades do remédio devem ser ingeridas até o final do tratamento (n_5) e qual o nome do remédio que estará sendo alocado na caixa (de x_1 a x_{n_6}, sendo n_6 o número de caracteres no nome do remédio). Com estas informações, o programa calcula quais os horários em que devem ser ingeridos no período do dia. Estes horários são, então, configurados na matriz de horários, explicada no item "Requisição de matriz de horários".

Requisição de matriz de horários Limitando o uso dos medicamentos a horários fechados de trinta em trinta minutos, resulta que apenas 48 horários são possíveis para tal atividade. Com isso, um vetor de tamanho 48 pode ser definido para cada remédio e, considerando 8 espaços para armazenamento de remédios na caixa, uma matriz 48 x 8 determina totalmente quando cada remédio deve ser ingerido. Acorda-se, neste protocolo, que o valor 1 em um componente desta matriz determina que o remédio respectivo deve ser ingerido naquele momento, enquanto que o valor 0 significa que o mesmo não deve ser ingerido naquele momento. Por exemplo, se o componente da linha 45 e coluna 7 está como 1, isto significa que o remédio alocado no sétimo slot da caixa deve ser ingerido às 22h.

Requisição de nomes dos medicamentos Com o recebimento deste símbolo, os nomes dos medicamentos armazenados na caixa são enviados pela mesma. Cada *string* com os nomes dos medicamentos possui 20 caracteres, sendo que os caracteres que sobram são preenchidos com espaços em branco até completarem-se os 20. Cada nome de remédio é enviado em sequência, um após o outro, em ordem numérica de armazenamento, do 1 ao 8.

Apagar a coluna da matriz de horários Nesta requisição, o valor de n_1 determina qual a coluna da matriz de horários, deve ser apagada (preenchida por zeros). Após esta função, portanto, o respectivo remédio é configurado para não ser ingerido em nenhum horário possível.

Configurar horários de uso manualmente Como descrito no item "Alocação de configuração de remédio", os horários de uso de remédio são, em um primeiro momento, inferidos através do horário obrigatório de tomá-lo e sua periodicidade ao dia. Para que os horários sejam editados manualmente, é necessário que o sistema apague a coluna da matriz de horários do respectivo remédio (através do envio do símbolo "a"explicado no item anterior) e, após, faça o envio deste símbolo referido. Determina-se que n_1 diz respeito ao *slot* do remédio cujos horários se deseja editar manualmente, e n_2 e n_3 determinam, respectivamente, hora e minuto de medicação.

Configurar ID de usuário Cada usuário deve ter sua ID (pode ser seu nome, seu CPF ou qualquer outra convenção que a identifique) e esta informação também deve estar alocada na caixa, para que a comparação entre as informações do sistema e da caixa esteja sempre de acordo, prevenindo troca de caixas e garantindo maior segurança no tratamento do paciente. O caractere n_1 informa quantos caracteres haverá na informação de ID que segue, que vai de x_1 até o x_{n_1}. A informação pura de ID de usuário deve ser de, no máximo, 20 bytes.

Requisição e ID de usuário Sob requisição do sistema, esta função envia para o mesmo a ID do usuário que está salva na caixa.

Requisição de informações para farmacêutico Este símbolo faz o requerimento de informações pertinentes ao farmacêutico, que será o responsável por preencher os remédios nos corretos compartimentos. A caixa envia estas informações na ordem e forma que segue:

1. Número de série da caixa em uma sequência de 6 bytes;
2. ID do usuário em uma sequência de 20 bytes;
3. Número de identificação da última consulta médica com configuração da caixa, em uma sequência de 20 bytes, descrito no item "Configurar ID da consulta médica";
4. Envia nome ou ID dos remédios configurados na caixa em ordem crescente de *slots*. Cada ID possui 20 bytes;
5. Envia quais *slots* devem ser preenchidos pelo farmacêutico. Esta informação está em uma sequência de 8 bytes.

Alocação física de remédio Neste item, o sistema, através da ação do farmacêutico, informa quais os remédios que foram fisicamente alocados nos compartimentos (através do valor correspondente ao compartimento em n_1), bem como o número de unidades (determinada em n_2).

Requisição de informações para médico Nesta requisição do sistema, a caixa irá enviar informações pertinentes ao médico em um pacote único. Estas estão descritas a seguir, em ordem:

1. Número de série da caixa em uma sequência de 6 bytes;
2. ID do usuário em uma sequência de 20 bytes;
3. Número de identificação da última consulta médica com configuração da caixa, em uma sequência de 20 bytes, descrito no item "Configurar ID da consulta médica";
4. Envia nome ou ID dos remédios configurados na caixa em ordem crescente de *slots*. Cada ID possui 20 bytes;
5. Envia quantas unidadesde cada medicamento faltam para finalizar o período de tratamento. Sequência de 8 bytes;
6. Quantidade de remédios ainda presentes nos compartimentos da caixa. Sequência de 8 bytes;
7. Matriz (48x8 bytes) de horários de uso dos remédios;
8. Quais compartimentos devem ser preenchidos com remédios, caso o paciente ainda não tenha ido ao farmacêutico, ou mesmo para checagem de configuração feita.

Requisição de número de unidades disponíveis em cada compartimento Esta requisição é respondida com o envio de um vetor de 8 bytes, cada um deles com o número de unidades ainda restantes no compartimento respectivo.

Requisição de número de unidades a ser ingerida até o final do período A placa responde a esta requisição da mesma forma como no item anterior, utilizando um vetor de 8 bytes em que cada um destes se refere à quantidade de medicamentos, em unidades, que deve ser ingerida pelo paciente até o final do período estipulado.

Requisição de número de série do produto Este item pede pelo número de série da caixa, que é passada através de 6 bytes em sequência. Este número é fixo para cada produto individualmente e é interessante haver esta opção para que o dono da caixa seja checado contra o número de série da mesma. Esta informação também é importante para fins de rastreamento caso algum extravio do produto aconteça ou mesmo algum defeito exista nesta unidade.

Requisição de número da última consulta O número da última consulta serve apenas para assegurar que as informações contidas na caixa estão de acordo com as últimas mudanças. É importante frisar que informações deste tipo, juntamente com o resto do

histórico de consultas, devem ser salvas no sistema externo à placa. Esta informação se dá em uma sequência de 20 *bytes*.

Configurar número da última consulta Este comando configura um número correspondente à última consulta do paciente e usuário da caixa com o seu médico. Uma sequência de 20 *bytes* é enviada entre n_1 e n_{20} referindo-se ao número da consulta.

6 VALIDAÇÃO DE PROJETO

Neste capítulo serão abordados os experimentos e testes referentes ao produto em seus blocos internos. Os equipamentos externos utilizados para os testes foram *protoboard*, fonte de tensão modelo 1902b da BK Precision, multímetro modelo 15B+ da Fluke e osciloscópio modelo DSOX1204 da Keysight.

6.1 Conversores boost

6.1.1 Conversor boost de baixa corrente

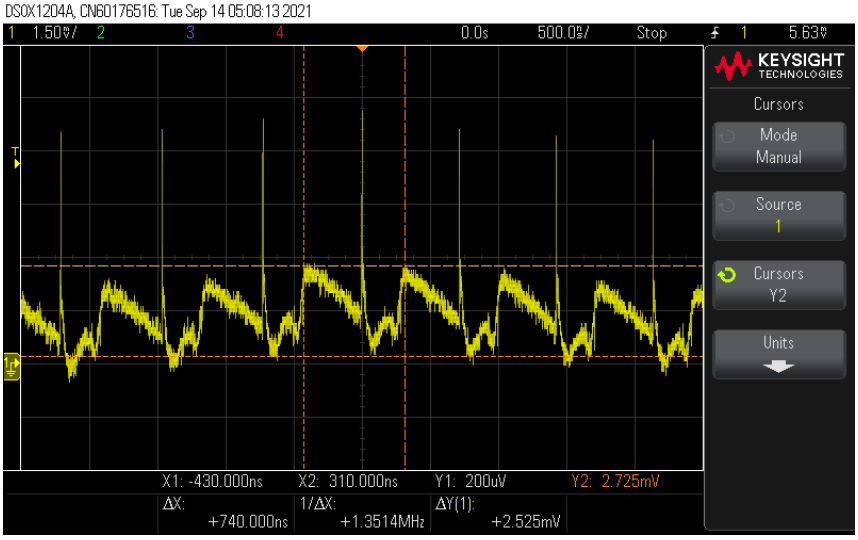
Em um primeiro momento, o bloco de circuito correspondente ao conversor *boost* foi isolado do restante da placa não fazendo a solda dos resistores R4 e R6, como mostra na Figura 11. A fonte de tensão foi variada entre 2,5 V e 4,5 V, não havendo diferenças notáveis nos resultados. Através da solda de um fio no local onde iria soldado o resistor R6, foi possível fixar esta tensão em um ponto da *protoboard* e colocar um resistor de 231 Ω para descarga. Com o multímetro, mediu-se uma tensão de saída DC em 5,04 V. Os 40 mV de diferença, correspondentes a 0,8%, estão dentro do valor de tolerância estipulado como requisito na simulação, de 10%, e o resultante da simulação, de 3,6%. A Figura 22 mostra o resultado do osciloscópio na medida de *ripple* de tensão.

Conforme se verifica na tela, foi medida uma tensão pico a pico de 2,5 mV. Este valor é, portanto, menor do que o valor de 5,38 mV estipulado por simulação e pode ser considerado plenamente satisfatório para esta aplicação. Os grandes picos de curta duração mostrados na imagem não parecem ter relação com a eletrônica do conversor *boost*, mas sim com uma indutância associada à ponteira de prova do osciloscópio, como mostra a Figura 23.

6.1.2 Conversor boost de alta corrente

Da mesma maneira com que fora testado o conversor *boost* de baixa corrente, este conversor foi isolado do restante do circuito através da não montagem dos resistores R1 e R2 da Figura 13, e os pontos de medida e entrada em tensão foram levados à *protoboard*

Figura 22: *Medida de ripple de tensão para o regulador boost de baixa corrente*



Fonte: O autor (2021)

Figura 23: *Ponteira de prova do osciloscópio utilizada*

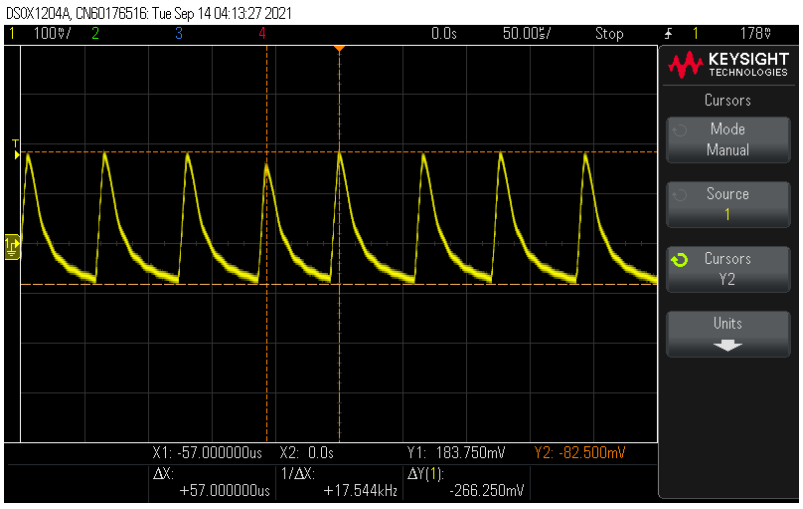
Fonte: O autor (2021)

para fazer as ligações. O valor DC medido em 5,09 V corresponde a um desvio de 1,8 % em relação ao nominal de 5 V. Com isso, este parâmetro está dentro do estipulado em projeto, de 10 %, e do resultante em simulação, de 3,6 %.

A Figura 24 mostra os valores referentes à tensão pico a pico. Observa-se uma alta divergência deste parâmetro em relação ao valor de 46,8 mV simulado. Esta divergência, provavelmente, é consequência do ponto de operação do regulador. O mesmo está atuando com uma corrente bastante abaixo daquela para a qual foi projetado (em torno de 22 mA) e, portanto, os resultados de simulação para este caso talvez não sejam tão representativos da realidade nestas condições. Sendo assim, recomenda-se o uso deste regulador para casos em que a corrente ultrapasse ao menos 1 A, situação para a qual há possibilidade de aplicação futuramente, se houverem outros componentes alimentados por esta tensão. De qualquer forma, ainda há um certo conforto pelo fato de que este *ripple* não representa

risco ao microcontrolador, cuja tolerância na alimentação é ainda maior, na ordem de Volts. Observa-se, neste momento, a importância de executar um projeto com uma certa folga no dimensionamento de parâmetros. Situações como esta podem ocorrer e, muitas vezes, o tempo disponível para corrigi-las é escasso.

Figura 24: *Medida de ripple de tensão para o regulador boost de alta corrente*



Fonte: O autor (2021)

6.2 Consumo do microcontrolador

Utilizando um resistor de 1 Ω em R2 (como mostra a Figura 13), foi possível, através da medida em tensão neste resistor, determinar a corrente consumida pelo microcontrolador. O valor medido foi de 11 mV, que representa apenas 11 mA de corrente. Este é um valor baixo e desejável, já que sistemas que operam com baterias devem prover alta duração com a carga disponível.

6.3 Carregamento da bateria

Os testes para este CI foram conduzidos de três formas, como mostram os itens a seguir.

Apenas bateria como fonte de alimentação Neste caso, uma tensão de 3.9 V foi medida na saída do CI, no local indicado pelo pino OUT_CHG da Figura 10. Esta tensão corresponde à tensão de saída da bateria, comprovando que o caminho de alimentação está corretamente sendo provido pelo CI para a situação em que não há conexão USB da caixa com o computador. Além disso, os LEDS não acendem, já que, em situações de uso da bateria como fonte de energia, não é prioridade que haja

indicadores visuais de conexão com a bateria em favor de um gasto de energia associado a tal atividade.

Apenas fonte externa conectada Nesta situação, o LED de PGOOD# (mostrado na Figura 10 com o designador LED2) é ativado e uma tensão DC de 4,36 V é apresentada na saída.

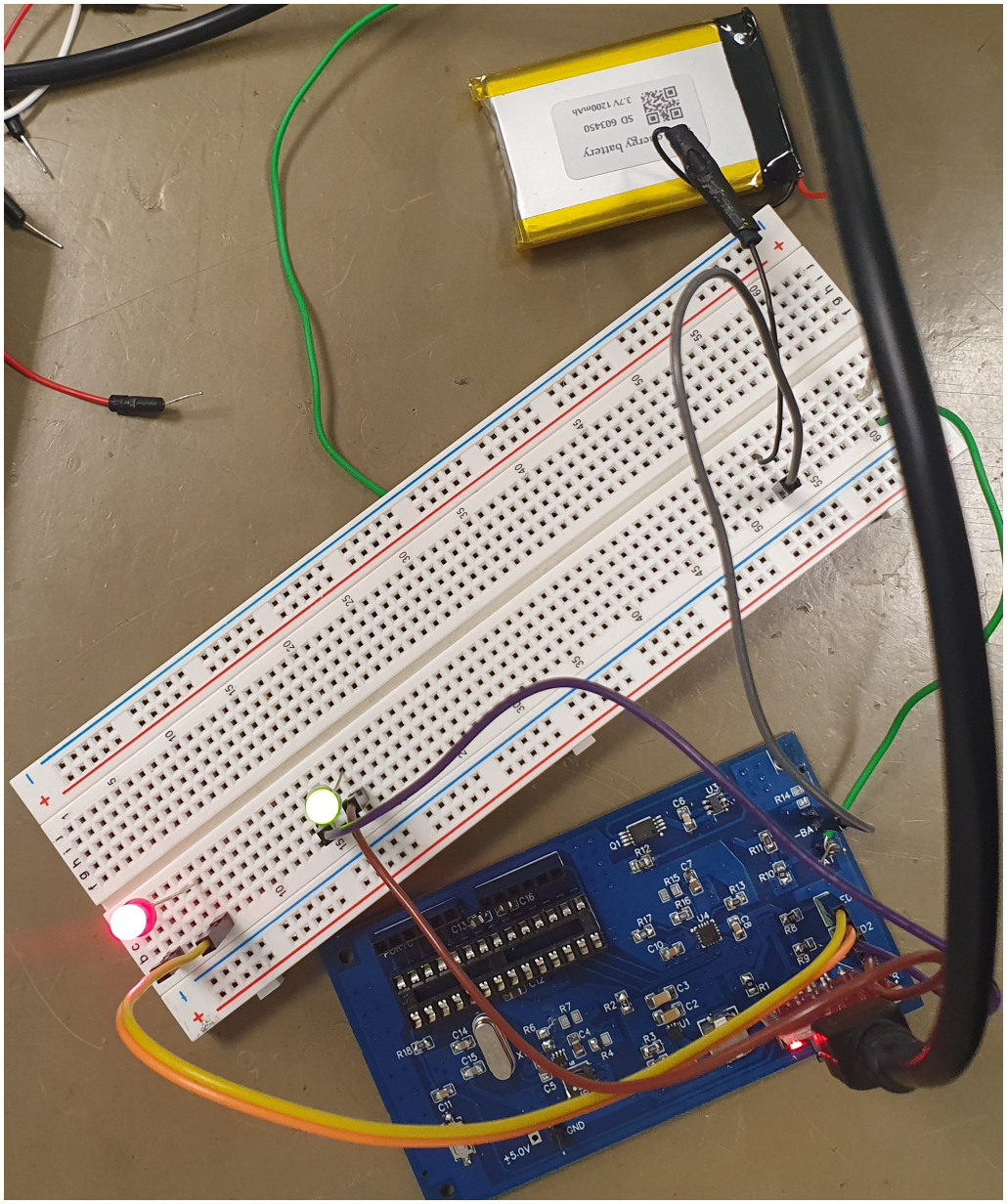
Fonte externa e bateria conectados Por fim, nesta situação, em que ambas fontes de energia são conectadas, o LED1, designador associado ao LED que indica carregamento da bateria (Figura 10), é ativado juntamente com o já mencionado LED2. Após a carga da bateria, este mesmo LED oscila sua luminosidade brevemente e apaga. Neste caso, a tensão medida na saída é a mesma da citada para o caso em que haja apenas a fonte externa conectada.

Uma foto correspondente ao caso de conexão da fonte externa (USB) e bateria é mostrada na Figura 25. Observa-se, na imagem, a placa, o conector de USB e seu cabo, a bateria, ao topo, e os LEDS ativos.

6.4 Proteção da bateria

A proteção da bateria foi testada nas situações de sobretensão e subtensão nos terminais da bateria. Para tanto, uma fonte de tensão foi conectada aos terminais do resistor R10 da Figura 9 e GND para o caso dos testes de sobretensão e, para o caso dos testes de subtensão, a fonte foi conectada onde iria a bateria e as medidas foram feitas nos terminais do resistor R10 e GND. Os testes foram conduzidos variando-se a tensão de entrada da fonte externa e medindo-se nos terminais da bateria para o caso de sobretensão ou nos terminais de R10 e GND para o caso de subtensão. Esperava-se que o circuito de proteção ativasse o corte de sobretensão em 4,25 V e retomasse o carregamento em 4,15 V. Em subtensão, esperava-se que o circuito cortasse a conexão para tensões abaixo de 2,9V e formasse a conexão novamente para tensões acima de 3V. Como a fonte externa apenas conseguia variar sua tensão em degraus de 100 mV, a precisão do teste está restrita a esta ordem de tensão. No caso de sobretensão, obteve-se os valores de 4,3 V e 4,1 V para corte e conexão, respectivamente. No caso de subtensão, obteve-se 2,9 V e 3,1 V para corte e conexão, respectivamente. É possível, assim, afirmar que, com as restrições de precisão impostas, o circuito se comporta como esperado e não apresentará risco no momento de utilização, já que cumpre com os requisitos estipulados em *datasheet* pelo fabricante da bateria.

Figura 25: *Fotografia das conexões para medidas do carregador da bateria*



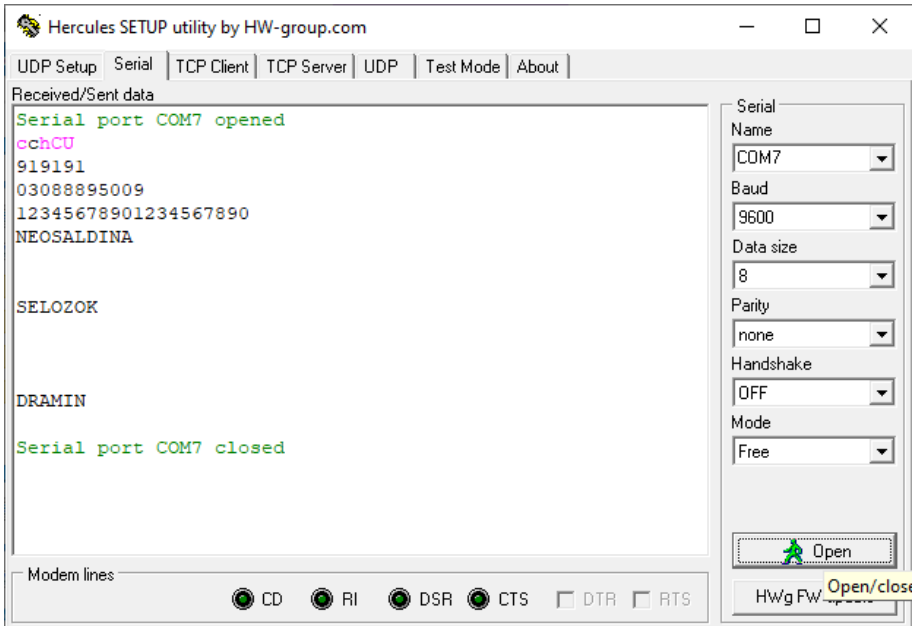
Fonte: O autor (2021)

6.5 Programa

Em termos de *hardware*, o funcionamento do programa depende simplesmente do recebimento e transmissão dos *bytes* no microcontrolador. Isto pode ser observado na Figura 26. Esta imagem mostra, através do programa monitor de serial Hercules, o resultado de uma série de *bytes* trocados entre o microcontrolador e o computador. Neste teste uma conexão é estabelecida, a hora é enviada e, como resposta, a placa envia dados de número de série, identificação do paciente, identificação da última consulta e, por fim, os

nomes dos remédios alocados nos respectivos compartimentos. Neste exemplo, o primeiro, o quarto e o oitavo compartimento estão preenchidos com os remédios NEOSALDINA, SELOZOK e DRAMIN, respectivamente.

Figura 26: *Resposta da conexão serial com a placa*



Fonte: O autor (2021)

Cada uma das funcionalidades descritas na Seção 5.2 está em pleno funcionamento e, considerando o objetivo deste documento, será apenas deixado o código ao final, no Apêndice A.

7 CONCLUSÃO

De forma geral, é possível afirmar que o projeto foi executado com sucesso considerando as soluções pensadas e a maneira com que o mesmo foi desenvolvido. Alguns desvios em relação às práticas na indústria são claros e devem ser pontuados. Em um projeto industrial, dependendo da sua área de atuação e volumes finais, há uma dependência muito forte com o seu custo, seja em tempo de desenvolvimento, alocação de engenheiros, compra de materiais (componentes eletrônicos e PCB) e a própria montagem do produto. Este projeto focou primariamente em aspectos técnicos, desconsiderando questões relativas a custos, o que, em certos casos pontuais, como na indústria de baixo volume de produtos, pode ser uma aproximação razoável. Seria inviável, por exemplo, do ponto de vista de custo, manter os dois reguladores *boost* na placa, considerando que para as situações invariáveis de uso da mesma apenas um deles se faz necessário.

Outro ponto importante em um produto é a compatibilidade entre o projeto mecânico e o projeto elétrico. Além dos furos para parafusamento colocados na placa, o próprio fator de forma da mesma deve estar em comum acordo entre engenheiros dos setores de mecânica e eletrônica. Sendo assim, em uma situação mais comum na indústria, haveriam requisitos com relação ao tamanho da placa previamente estabelecidos e estudados, considerando o produto como um todo.

Há algumas propostas de melhorias que surgem ao se avaliar as outras soluções presentes no mercado. Em possibilidades futuras, talvez fosse interessante o uso de um sistema *WiFi* com mensagens para os *smartphones* dos responsáveis, ou mesmo algum sistema de reconhecimento biométrico para configurar o produto, como descrito também na Seção 2.1. Estas e outras ideias são mais palpáveis com este primeiro protótipo em mãos e, se elas forem implementadas futuramente, muito reuso de esquemático poderia ser feito, diminuindo enormemente o tempo de projeto e o esforço.

Um aspecto que chama a atenção é que projetos como este demonstram claramente as vantagens de se fazer um trabalho cauteloso e prevenido. O trabalho de conclusão de curso permitiu que se tomasse um tempo adequado para trabalhar os riscos, estudando e detalhando cada aspecto duvidoso no projeto e pesquisando por soluções confiáveis. Novamente em um caso concreto presente neste projeto, traz-se à tona a discussão do uso

de provisões de resistores de 0 Ω e o incremento de um regulador *boost* de sobra para o caso de utilização de maior corrente. Ambas estratégias diminuem os riscos associados ao projeto e o flexibilizam para as diferentes situações em que esteja submetido.

Por fim, julga-se importante ressaltar que, durante as fases de planejamento e execução deste projeto, muitos conceitos sutis se mostraram extremamente relevantes. Cita-se como exemplo a discrepância entre o modelo ideal e real do capacitor e como é de crítica importância aplicar a sua modelagem correta quando um projeto de regulador está sendo estudado. No mesmo caso de componentes parasitas está o cuidado em *layout* com trilhas que possam resultar em indutâncias. Sem estes fatores serem levados em conta, o projeto da placa correria altos riscos, podendo perfeitamente inviabilizar a sua utilização ao final.

APÊNDICE A - CÓDIGO EM LINGUAGEM C IMPLEMENTADO

O código apresentado a seguir está como se encontra no momento de escrita este documento. Ele contém ainda algumas funções e implementações que dizem respeito à depuração e, portanto, possui algumas funcionalidades a mais em relação ao que foi explanado na Seção 5.2.

A.1 Main

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <stdio.h>
#include <string.h>
#include "timers.h"
#include "serial.h"
#include "geral.h"
#include "global_variables.h"
#include "eeprom.h"

char letter;
uint8_t word[27];
uint8_t character=0, connection=0, time_connection=0;
char time_string[6];
uint8_t current_hour, current_minute, current_second, time_medicine[8][3],i
    ,j,period_medicine[8],number_of_letters,possible_times[48][2];
uint8_t matrix[48][8];
uint8_t EEMEM matrix_eeprom[48][8];
uint8_t state=0,medicine,jump,ind;
char medicine_name[8][20];
```

```
char EEMEM medicine_name_eeprom[8][20];
char personal_name[20];
char EEMEM personal_name_eeprom[20];
uint8_t which_slots_to_be_filled[8];
uint8_t EEMEM which_slots_to_be_filled_eeprom[8];
uint8_t medicine_quantity[8];
uint8_t EEMEM medicine_quantity_eprom[8];
uint8_t units_to_finish_period[8];
uint8_t EEMEM units_to_finish_period_eeprom[8];
uint8_t identification_number[6]={1,2,3,4,5,6};
uint8_t EEMEM identification_number_eeprom[6]={1,2,3,4,5,6};
uint8_t day, month, year;
uint8_t app_number[20];
uint8_t EEMEM app_number_eeprom[20];

ISR(USART_RX_vect)
{
        letter=receive();
        if (letter=='#')
        {
                connection=1;
                time_connection=0;
                return;
        }
        word[character]=letter;
        number_of_letters=find_number_of_letters(character,letter,
   number_of_letters);
        character++;

        if (character==number_of_letters)
        {
                character=0;
                if (error_message(word[0], state)==0)
                {
                        receive_data();
                        //save_alteration(word);
                        zero_word(word);
                }
        }
}
```

```
ISR(TIMER1_COMPA_vect)
{
        count_time();
        check_connection();

}

int main(void)
{
        timer1_real_time_init();
        init_serial();

        DDRD = 0xFF; //Set port D as output

        PORTD &= ~(1<<3); // CE# = 0
        PORTD &= ~(1<<6); // EN1 = 0
        PORTD |= (1<<7); // EN2=1;

        DDRD &= ~ (1<<5); // Power good as input
        DDRD &= ~ (1<<4); // CHG# charging status as input

        DDRB |= (1<<5); //Set bit 5 of port B as output
        PORTB &= ~ (1<<5);

        sei(); //Global interrupt enabled
        possible_times_init();
        read_from_eeprom();
        while (1)
        {
```

A.2 Timer

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include "timers.h"
#include "serial.h"
#include "geral.h"
```

```
void timer1_real_time_init(void)
{
        TCCR1A=0; //CTC activated
        TCCR1B=0x0D; //CTC activated and prescaler of 1024
        OCR1AH=0x3D; //Compare register high value
        OCR1AL=0x08; //Compare register low value
        TIMSK1=0x02; //Interrupt by OCR1A enabled
}
```

A.3 Serial

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include "timers.h"
#include "serial.h"
#include "geral.h"

void init_serial(void)
{
        UBRR0H=0;
        UBRR0L=103; //select baud rate (9600)
        UCSR0B=(1<<4)|(1<<3); //activate transmission and reception
        UCSR0C=(1<<1)|(1<<2); //8 bits per frame
        UCSR0B |= (1 << 7); //enable interruption by complete reception
}

void send(unsigned char caractere)
{
        while(1) //write byte when it's clean
        {
                if(((UCSR0A>>5) & 0x01)==1)
                {
                        UDR0=caractere;
                        return;
                }
        }
}
```

```
void send_number_to_char(uint8_t number)
{
        while(1) //write byte when it's clean
        {
                if(((UCSR0A>>5) & 0x01)==1)
                {
                        UDR0=number+0x30;
                        return;
                }
        }
}

void send_char_to_number(char number)
{
        while(1) //write byte when it's clean
        {
                if(((UCSR0A>>5) & 0x01)==1)
                {
                        UDR0=number-0x30;
                        return;
                }
        }
}

char receive(void)
{
        while(1)
        {
                if(((UCSR0A>>7) & 0x01)==1)
                {
                        return UDR0;
                }
        }
}
```

A.4 EEPROM

```
#include <avr/io.h>
#include <avr/interrupt.h>
```

```
#include <stdio.h>
#include <string.h>
#include "timers.h"
#include "serial.h"
#include "geral.h"
#include "global_variables.h"
#include "eeprom.h"

void read_from_eeprom()
{
        matrix_read_from_eeprom();
        medicine_names_read_from_eeprom();
        units_to_finish_period_read_from_eeprom();
        personal_name_read_from_eeprom();
        medicine_quantity_in_slot_read_from_eeprom();
        serial_number_read_from_eeprom();
        which_slots_to_be_filled_read_from_eeprom();
        app_number_read_from_eeprom();
}

void which_slots_to_be_filled_read_from_eeprom()
{
        eeprom_read_block((void*) &which_slots_to_be_filled, (const void*)&
    which_slots_to_be_filled_eeprom, 8);
}

void which_slots_to_be_filled_write_eeprom()
{
        eeprom_write_block((const void*)&which_slots_to_be_filled,(void*)&
    which_slots_to_be_filled_eeprom,8);
}

void serial_number_read_from_eeprom()
{
        eeprom_read_block((void*) &identification_number, (const void*) &
    identification_number_eeprom, 6);
}

void serial_number_write_eeprom()
{
```

```
        eeprom_write_block((const void*)&identification_number,(void*)&
    identification_number_eeprom,6);
}

void medicine_quantity_in_slot_read_from_eeprom()
{
        eeprom_read_block((void*) &medicine_quantity, (const void*) &
    medicine_quantity_eprom, 8);
}

void medicine_quantity_in_slot_write_eeprom()
{
        eeprom_write_block((const void*)&medicine_quantity,(void*)&
    medicine_quantity_eprom,8);
}

void personal_name_read_from_eeprom()
{
        eeprom_read_block((void*) &personal_name, (const void*) &
    personal_name_eeprom, 20);
}

void personal_name_write_eeprom()
{
        eeprom_write_block((const void*)&personal_name,(void*)&
    personal_name_eeprom,20);
}

void units_to_finish_period_read_from_eeprom()
{
        eeprom_read_block((void*) &units_to_finish_period, (const void*) &
    units_to_finish_period_eeprom, 8);
}

void medicines_to_take_write_eeprom()
{
        eeprom_write_block((const void*)&units_to_finish_period,(void*)&
    units_to_finish_period_eeprom,8);
}

void app_number_write_eeprom()
```

```
{
        eeprom_write_block((const void*)&app_number,(void*)&
    app_number_eeprom,20);
}

void app_number_read_from_eeprom()
{
        eeprom_read_block((void*) &app_number, (const void*) &
    app_number_eeprom, 20);
}

void identification_number_write_eeprom()
{
        eeprom_write_block((const void*)&identification_number,(void*)&
    identification_number_eeprom,6);
}

void identification_number_read_from_eeprom()
{
        eeprom_read_block((void*) &identification_number, (const void*) &
    identification_number_eeprom, 6);
}

void medicine_names_read_from_eeprom()
{
        for (i=0;i<8;i++)
        {
                for (j=0;j<20;j++)
                {
                        eeprom_read_block((void*) &medicine_name[i][j], (
    const void*) &medicine_name_eeprom[i][j], 1);
                        //medicine_name[i][j]=eeprom_read_byte((const
    uint_8*)&medicine_name_eeprom[i][j]);
                }
        }

}

void medicine_names_write_eeprom()
{
        for (i=0;i<8;i++)
```

```
        {
                for (j=0;j<20;j++)
                {
                        eeprom_write_byte((uint8_t*)&medicine_name_eeprom[i
   ][j], medicine_name[i][j]);
                }
        }
}

void matrix_write_eeprom()
{
        for (j=0;j<48;j++)
        {
                for (i=0;i<8;i++)
                {
                        eeprom_write_byte((uint8_t*)&matrix_eeprom[j][i],
   matrix[j][i]);
                }
        }
}

void matrix_read_from_eeprom()
{
        for (i=0;i<48;i++)
        {
                for (j=0;j<8;j++)
                {
                        matrix[i][j]=eeprom_read_byte((const uint8_t*)&
   matrix_eeprom[i][j]);
                }
        }
}
```

A.5 Funções genéricas

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include "timers.h"
#include "serial.h"
```

```
#include "geral.h"
#include "global_variables.h"
#include "eeprom.h"


void send_doctor_status()
{
        send_identification_number();
        send_personal_name();
        send_app_number();
        send_medicine_names();
        send_units_to_finish_period();
        send_medicine_quantity();
        send_matrix();
        send_which_slots_to_be_filled();
}

void send_pharm_status()
{
        send_identification_number();
        send_personal_name();
        send_app_number();
        send_medicine_names();
        send_which_slots_to_be_filled();
}

void send_app_number()
{
        for (i=0;i<20;i++)
        {
                send(app_number[i]);
        }
}

void send_medicine_quantity()
{
        for (i=0;i<8;i++)
        {
                send(medicine_quantity[i]);
        }
}
```

```
void send_units_to_finish_period()
{
        for (i=0;i<8;i++)
        {
                send(units_to_finish_period[i]);
        }
}

void string_serial(char string[])
{
        for(i=0;i<strlen(string);i++)
        {
                send(string[i]);
        }
}

void send_which_slots_to_be_filled()
{
        for (i=0;i<8;i++)
        {
                send(which_slots_to_be_filled[i]);
        }
}

void receive_data()
{
        switch(word[0])
        {
                case 'c':
                //send(0x0d);
                //send(0x0a);
                //show_medicine_names();
                //read_from_eeprom();
                connection=1;
                state=1;
                time_connection=0;
                send('c');
                break;

                case 'h':
```

```
                current_hour=word[1];
                current_minute=word[2];
                current_second=word[3];
                day=word[4];
                month=word[5];
                year=word[6];
                state=3;
                send_doctor_status();
                break;


                case 'r':
                for (i=0;i<2;i++)
                {
                        time_medicine[word[1]][i]=word[i+2];
                }
                medicine=word[1];
                zero_name(medicine);
                period_medicine[medicine]=word[4];
                ind=find_index(time_medicine[medicine][0],time_medicine[
medicine][1]);
                if (ind=='I')
                {
                        send('I');
                        return;
                }
                zero_column(medicine);
                write_matrix(medicine, period_medicine[medicine], ind);
                which_slots_to_be_filled[medicine]=1;
                which_slots_to_be_filled_write_eeprom();
                units_to_finish_period[medicine]=word[5];
                medicines_to_take_write_eeprom();
                for (i=0;i<number_of_letters-7;i++)
                {
                        medicine_name[medicine][i]=word[i+7];
                }
                medicine_names_write_eeprom();
                send('r');
                break;

                case 'm':
```

```
send_matrix();
break;

case 'n':
send_medicine_names();
break;

case 'e':
matrix[find_index(word[2],word[3])][word[1]]=1;
matrix_write_eeprom();
send('e');
break;

case 'a':
zero_column(word[1]);
send('a');
break;

case 'o':
clean_name();
for (i=0;i<(number_of_letters-2);i++)
{
        personal_name[i]=word[i+2];
}
personal_name_write_eeprom();

send('o');
break;

case 'q':
send_personal_name();
break;

case 'f':
send_pharm_status();
state=2;
break;

case 'g':
which_slots_to_be_filled[word[1]]=0;
which_slots_to_be_filled_write_eeprom();
```

```
                medicine_quantity[word[1]]=word[2];
                medicine_quantity_in_slot_write_eeprom();
                send('g');
                break;

                case 'd':
                state=3;
                send_doctor_status();
                break;

                case 'j':
                for (i=0;i<8;i++)
                {
                        send(medicine_quantity[i]);
                }
                break;

                case 'p':
                send_units_to_finish_period();
                break;

                case 's':
                send_identification_number();
                break;

                case 'z':
                send_app_number();
                break;

                case 'b':
                for (i=0;i<20;i++)
                {
                        app_number[i]=word[i+1];
                }
                app_number_write_eeprom();
                send('b');
                break;

                case 'i':
                for (i=0;i<6;i++)
                {
```

```
                        identification_number[i]=word[i+1];
                }
                identification_number_write_eeprom();
                send('i');
                break;
                //case 'k':
                //send_alterations();
                //break;
        }

}


void zero_word(uint8_t word[27])
{
        for (i=0;i<40;i++)
        {
                word[i]='␣';
        }
}

void clean_name()
{
        for (i=0;i<20;i++)
        {
                personal_name[i]='␣';
        }
        personal_name_write_eeprom();
}

void paragraph()
{
        send(0x0d);
        send(0x0a);
}

void send_personal_name()
{
        for (i=0;i<20;i++)
        {
                send(personal_name[i]);
```

```
        }
}

void zero_name(uint8_t row)
{
        for (i=0;i<20;i++)
        {
                medicine_name[row][i]='␣';
        }
        medicine_names_write_eeprom();
}

void send_medicine_names()
{
        for (i=0;i<8;i++)
        {
                for (j=0;j<20;j++)
                {
                        send(medicine_name[i][j]);
                }
                paragraph();
        }
}

void write_matrix(uint8_t remedy_number, uint8_t times_per_day, uint8_t
    indice)
{
        if (times_per_day==0)
        {
                return;
        }
        else
        {
                jump=48/times_per_day;
                matrix[indice][remedy_number]=1;
                i=jump+indice;
                while (i!=indice)
                {
                        if (i>47)
                        {
                                i=i%48;
```

```
                        }
                        matrix[i][remedy_number]=1;
                        i=i+jump;
                        if (i>47)
                        {
                                i=i%48;
                        }
                }

        }
        matrix_write_eeprom();
}


uint8_t error_message(char letter, uint8_t state)
{

        switch(state)
        {
                case 0:
                if (letter==’c’)
                {
                        return 0;
                }
                else
                {
                        send(’C’);
                }
                break;

                case 1:
                if (letter==’h’)
                {
                        return 0;
                }
                else
                {
                        send(’H’);
                }
                break;
```

```
                case 2:
                if (letter=='g' || letter=='d' || letter=='p' || letter=='n
    ' || letter=='z' || letter=='i')
                {
                        return 0;
                }
                else
                {
                        send('F');
                }
                break;

                case 3:
                if (letter=='o' || letter=='r' || letter=='a' || letter=='m
    ' || letter=='e' || letter=='f' || letter=='q' || letter=='j' || letter
    =='n' || letter=='s' || letter=='p' || letter=='z' || letter=='b' ||
    letter=='i' || letter=='g')
                {
                        return 0;
                }
                else
                {
                        send('D');
                }
        }

        return 1;
}

uint8_t find_index(uint8_t hour, uint8_t minute)
{
        for (i=0;i<48;i++)
        {
                if (hour==possible_times[i][0] && minute==possible_times[i
    ][1])
                {
                        return i;
                }
        }
        return 'F';
}
```

```
void send_matrix()
{
        for (i=0;i<48;i++)
        {
                for (j=0;j<8;j++)
                {
                        send_number_to_char(matrix[i][j]);
                }
        }
}

void send_matrix_debug()
{
        send(0x0d);
        send(0x0a);
        for (i=0;i<48;i++)
        {
                send_number_to_char(i/10);
                send_number_to_char(i%10);
                send(':');
                send('␣');
                for (j=0;j<2;j++)
                {
                        send_number_to_char((possible_times[i][j])/10);
                        send_number_to_char((possible_times[i][j])%10);
                        if (j==0)
                        {
                                send('h');
                        }
                }
                for (j=0;j<8;j++)
                {
                        send('␣');
                        send_number_to_char(matrix[i][j]);
                }
                send(0x0d);
                send(0x0a);
        }
}
```

```
void zero_column(uint8_t column)
{
        for (i=0;i<48;i++)
        {
                matrix[i][column]=0;
        }
        matrix_write_eeprom();
}

void possible_times_init()
{
        for(i=0;i<48;i++)
        {
                possible_times[i][0]=i/2;
                if (i%2==0)
                {
                        possible_times[i][1]=0;
                }
                else
                {
                        possible_times[i][1]=30;
                }
        }
}

void matrix_init()
{
        for (i=0;i<48;i++)
        {
                for (j=0;j<8;j++)
                {
                        matrix[i][j]=eeprom_read_byte((const uint8_t*)&
   matrix_eeprom[i][j]);
                        matrix[i][j]=0;
                }
        }
}


void send_time()
{
```

```
        send('%');
        send_number_to_char(current_hour/10);
        send_number_to_char(current_hour%10);
        send(':');
        send_number_to_char(current_minute/10);
        send_number_to_char(current_minute%10);
        send(':');
        send_number_to_char(current_second/10);
        send_number_to_char(current_second%10);
        send('%');
}

void count_time()
{
        current_second++;
        if(current_second>59)
        {
                current_second=0;
                current_minute++;
                if (current_minute>59)
                {
                        current_minute=0;
                        current_hour++;
                        if (current_hour>23)
                        {
                                current_hour=0;
                        }
                }

        }
}


uint8_t sum(uint8_t vector[], uint8_t n)
{
        uint8_t sum_value=0;
        for(i=0;i<n;i++)
        {
                sum_value=sum_value+vector[i];
        }
        return sum_value;
```

```
}


uint8_t find_number_of_letters(uint8_t character, char letter, uint8_t
    number_of_letters)
{
                if (character==0)
                {
                        switch(letter)
                        {

                                case 'h':
                                return 7;

                                case 'r':
                                return 7;

                                case 'o':
                                return 2;

                                case 'e':
                                return 4;

                                case 'a':
                                return 2;

                                case 'g':
                                return 3;

                                case 'b':
                                return 21;

                                case 'i':
                                return 7;

                                default:
                                return 1;
                        }
                }

                if (character==1)
```

```
                {
                        if (word[0]=='o')
                        {
                                uint8_t number= (uint8_t)word[1];
                                return number+2;
                        }
                }

                if (character==6)
                {
                        if (word[0]=='r')
                        {
                                return word[6]+7;
                        }
                }
                return number_of_letters;
}

void check_connection()
{
        time_connection++;
        if((time_connection==15) && (state!=0))
        {
                send('#');
        }
        if (time_connection>19)
        {
                state=0;
        }
}

void send_identification_number()
{
        for (i=0;i<6;i++)
        {
                send(identification_number[i]);
        }
}
```

REFERÊNCIAS

ABDUL MINAAM, D. S.; ABD-ELFATTAH, M. Smart drugs:Improving healthcare using Smart Pill Box for Medicine Reminder and Monitoring System. *Future Computing and Informatics Journal*, v. 3, n. 2, p. 443–456, 2018. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2314728818300230>. Acesso em: 25 ago. 2021.

ANALOG DEVICES INC. *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter*. [S.l.], 2020. p. 5. Disponível em: <https://www.analog.com/media/en/analog-dialogue/volume-54/number-4/uart-a-hardware-communication-protocol.pdf>. Acesso em: 29 ago. 2021.

BITTELE ELETRONICS. *PCB Trace Width Calculator*. [S.l.]. Disponível em: <https://www.7pcb.com/trace-width-calculator.php>. Acesso em: 11 set. 2020.

DATA POWER TECHNOLOGY LIMITED. *Polymer Li-ion Rechargeable Battery*. [S.l.], 2017. p. 10. Disponível em: <https://cdn.sparkfun.com/assets/5/6/e/1/5/SPE-DTP603450-1000mah-3.7V-En-1.0V.pdf>. Acesso em: 10 ago. 2021.

FORTUNE SEMICODUCTOR CORPORATION. *FS312F-G Datasheet: One Cell Lithium-Ion/Polymer Battery Protection IC*. [S.l.], 2014. p. 13. Disponível em: <https://www.ic-fortune.com/upload/Download/FS312F-G-DS-12_EN.pdf>. Acesso em: 10 ago. 2020.

GRAND VIEW RESEARCH. *Microcontroller Market Size, Share Trends Analysis Report By Product (8-bit, 16-bit, 32-bit), By Application (Automotive, Consumer Electronics Telecommunication, Medical Devices), And Segment Forecasts, 2021 - 2028*. [S.l.], 2021. p. 120. Disponível em: <https://www.grandviewresearch.com/industry-analysis/microcontroller-market>. Acesso em: 23 ago. 2021.

GUERRERO ULLOA, G. et al. IoT-Based Smart Medicine Dispenser to Control and Supervise Medication Intake. In:

IPC. *IPC 2221: Generic Standard on Printed Circuit Board Design*. [S.l.], 2012. p. 170.

MAZIDI, M. E. A. *The AVR microcontroller and embedded systems : using Assembly and C*. London: Prentice Hall, 2011. p. 776. ISBN 0138003319.

MEETOO, D.; RYLANCE, R.; ABUHAIMID, H. Health Care in a Technological World. *British Journal of Nursing*, v. 27, p. 1172–1177, nov. 2018.

MEMO BOX. Memo Box Mini Bluetooth Electronic Pill Organizer and APP, Visual Audio Smart Pill Reminder Alarm, Automatic Medication Records Tracking and Family Meds Notifications. In: MEMO BOX. Disponível em: <https://www.amazon.com/Bluetooth-Electronic-Organizer-Medication-Notification/dp/B071J5PV84>. Acesso em: 25 ago. 2021.

MICROCHIP TECHNOLOGY INC. *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*. [S.l.], 2015. p. 294. Disponível em: <https://www.microchip.com/en-us/product/ATmega328P#document-table>. Acesso em: 4 ago. 2021.

MICROCHIP TECHNOLOGY INC. *AN2519: AVR Microcontroller Hardware Design Consideations*. [S.l.], 2018. p. 26. Disponível em: <https://ww1.microchip.com/downloads/en/AppNotes/AN2519-AVR-Microcontroller-Hardware-Design-Considerations-00002519B.pdf>. Acesso em: 10 ago. 2021.

NANDA, U.; PATTNAIK, S. Universal Asynchronous Receiver and Transmitter (UART). In: p. 1–5.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. *mHealth: New horizons for health through mobile technologies*. [S.l.], 2011. p. 102. Disponível em: <https://www.who.int/goe/publications/goe_mhealth_web.pdf>. Acesso em: 5 set. 2021.

PETERSON, Z. *PCB Trace and Pad Clearance: Low vs. High Voltage*. [S.l.], 2020. Disponível em: <https://resources.altium.com/p/pcb-trace-and-pad-clearance-low-vs-high-voltage>. Acesso em: 11 set. 2020.

TEXAS INSTRUMENTS INC. *BQ2407x Standalone 1-Cell 1.5-A Linear Battery Charger with PowerPath*. [S.l.], 2019a. p. 50. Disponível em: <https://www.ti.com/lit/ds/symlink/bq24074.pdf?ts=1628732329732&ref_url=https%253A%252F%252Fwww.ti.com%252Fsitesearch%252Fdocs%252Funiversalsearch.tsp%253FlangPref%253Den-US%2526searchTerm%253Dbq24074%2526nr%253D181>. Acesso em: 11 ago. 2020.

TEXAS INSTRUMENTS INC. *Switching Regulator Fudamentals*. [S.l.], 2019b. p. 29. Disponível em: <https://www.ti.com/lit/an/snva559c/snva559c.pdf?ts=1629743214125>. Acesso em: 23 ago. 2021.

TEXAS INSTRUMENTS INC. *TPS61022 8-A Boost Converter with 0.5-V Ultra-low Input Voltage*. [S.l.], 2021. p. 31. Disponível em: <https://www.ti.com/lit/ds/symlink/tps61022.pdf?ts=1631393739492&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS61022>. Acesso em: 11 set. 2020.

TEXAS INSTRUMENTS INC. *TPS61022 and TPS61023 Boost Converters Layout Guidelines*. [S.l.], 2020. p. 08. Disponível em: <https://www.ti.com/lit/an/slvaes4/slvaes4.pdf?ts=1631384997476&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTPS61022>. Acesso em: 11 set. 2020.

TEXAS INSTRUMENTS INC. *TPS6122x Low Input Voltage, 0.7V Boost Converter With 5.5μA Quiescent Current*. [S.l.], 2014. p. 27. Disponível em: <https://www.ti.com/lit/ds/symlink/tps61222.pdf?ts=1631393907731&ref_url=https%253A%252F%252Fwww.ti.com%252Fstore%252Fti%252Fen%252Fp%252Fproduct%252F%253Fp%253DTPS61222DCKT%2526utm_source%253Dgoogle%2526utm_medium%253Dcpc%2526utm_campaign%253Dapp-null-null-OPN_EN-cpc-store-google-wwe%2526utm_content%253DDevice%2526ds_k%253DTPS61222DCKT%2526DCM%253Dyes%2526gclid%253DCjwKCAjwp_GJBhBmEiwALWBQkwP5SQrQDkDJ_0cD9YLEGGgokVUqhn-Zoa4aN-rKpmmwK82ghtOz2hoCGVMQAvD_BwE%2526gclsrc%253Daw.ds>. Acesso em: 11 set. 2020.

THALES GROUP. MedMinder makes Medication Smarter with IoT Connected Pill Box. In: THALES GROUP. Disponível em: <https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/customer-cases/smart-pill-dispenser>. Acesso em: 25 ago. 2021.

THIMBLEBY, H. Technology and the Future of Healthcare. *Journal of public health research*, v. 2, e28, dez. 2013.