Título

Subtítulo

Relatório Final de
pesquisa de Iniciação Científica

Aluno: Leonardo José Held
Orientador: Raimes Moraes

Departamento de Engenharia Elétrica e Eletrônica
Curso de Graduação em Engenharia Eletrônica
Universidade Federal de Santa Catarina
Brasil

Conteúdo

Capítulo 1

Resumo

Nesse projeto foi realizado uma revisão bibliográfica acerca das tecnologias como *Bluetooth Low Energy* e Sistemas Operacionais e protocolos de comunicação entre periféricos eletrônicos, a seguir, um aprendizado diante das ferramentas disponíveis, como ambientes de programação de *software* e *firmware* integrados e diversos softwares gráficos e de linha de comando. Após essa asserção, foram iniciados trabalhos para a ampliação de um sistema de captação de sinais respiratórios, originalmente construído por (MATTOS, 2018). A ampliação do sistema contava, mas não somente, com a correção de bugs e aprimoramentos do sistema, inserção de interface para sensores de posicionamento espacial, introdução de um sistema de inicialização externa da aquisição via botão físico e melhoramento do sistema de coleta de dados.

Palavras-chave: Instrumentação médica. Sistema respiratório. Sons respiratórios. Estetoscópio Eletrônico. Processamento Digital de Sinais.

Capítulo 2

Introdução

2.1 Revisão bibliográfica

Os principais conceitos a serem revisados são os de Sistemas Operacionais em Tempo Real, Protocolos de Comunicação, familizarização com os manuais e *datasheets* dos equipamentos utilizados e leitura da pesquisa original realizada por (MATTOS, 2018). Diversos manuais e documentos de software foram consultados durante essa e todas as etapas do projeto.

(TANENBAUM; BOS, 2014) define um Sistema Operacional (SO) como software que rode em modo privilegiado. Ou seja, é o software que roda mais perto do hardware. Normalmente esse software é usado como barreira entre o hardware e o software de alto nível, sendo também chamado de *Middleware*. O SO define um claro ponto de entrada e saída para programas que rodem em alto nível. Também pode conter uma lógica de organização de qual programa vai ser executado em qual momento afim de obter o melhor uso de tempo possível de um dado processador.

(BARRY, s.d.) define um Sistema Operacional em Tempo Real como um sistema que precisa saber o tempo de término quão mais exato de uma tarefa. Ou seja, um sistema em tempo real se carateriza pelo caráter determínistico. É ideal

para aplicações onde ações externas precisam ser induzidas em intervalos de tempo regulares e extremamente corretos, como aquisição de som, no nosso caso, ou o disparo de um air-bag.

Protocolos de Comunicação são, como definidos por (POPOVIC, 2006) como um conjunto de regras que governam como mensagens deverão ser trocadas por computadores. É preciso entender, em especial, como funcionam os protocolos de comunicação *serial*, i^2c e *SPI*, sendo que os dois últimos são utilizados para a obter os dados dos sensores inseridos no sistema.

(HOROWITZ; HILL, 1989) define os dois protocolos de maneira extremamente semelhante, mas cita diferenças claras como a velocidade de transmissão e a comunicação totalmente bidirecional (full-duplex) na *SPI* e a simplicidade e fácil adesão de novos dispositivos no *bus* que é essencialmente a comunicação i^2c, permitindo fácil extensão de mais dispositivos num mesmo *bus*. Assim, possuímos dois tipos de comunicação bem suportada com parâmetros bem definidos e amplamente utilizadas para todo tipo de comunicação eletrônica.

Capítulo 3

MATERIAL E MÉTODOS

3.1 Características de módulos microcontrolados com capacidades *BLE 5.0*

De acordo com (AH, 2010), os principais pontos que fazem com que *Bluetooth* sejam uma boa escolha meios médicos são: interoperabilidade, baixo consumo, customização de formato de transmissão (pacote), compatibilidade eletromagnética (equipamentos médicos coexistem, não podendo inteferir negativamente uns com os outros), segurança na transmissão para proteger confidencialidade da informação e hub de sensores (diversos equipamentos médicos precisam ser atualizados em tempo real via internet para fins de telemedicina).

Também é um protocolo aberto para implementação, realizado pelas principais empresas de tecnologia mundiais, fomentando os pontos de interoperabilidade de equipamentos e segurança.

O sistema original foi criado usando uma plataforma embarcada da *Texas Instruments*® codenome `LAUCHPAD-XL` contendo o chip *CC2640R2*, e suportando a tecnologia *BLE 5.0*, *Bluetooth Low Energy 5.0*. Essa tecnologia possibilita uma alta taxa de transmissão, fazendo dela ideal para aplicações médicas que requerem grande *throughput* de da-

dos.

Observando o esquemático de blocos do *SoC*, vemos que o mesmo conta com um processador ARM Cortex-M3 e ARM Cortex-M0. O primeiro serve de processador de propósito geral, e é extendido para os pinos de entrada e saída (I/O) de cada placa. O segundo é um processador de propósito específico que contém código que controla o circuito de radiofrequência (RF) responsável pela implementação do protocolo *Bluetooth* da placa.

Fazemos uso, então, do *Cortex-M3* para acessar e obter dados dos sensores ou periféricos. Estes, por sua vez, são enviados via para a segunda *CC2640R2* usando a API da *TI*.

O protocolo de transmissão usado é uma tecnologia que define os requerimentos de *Bluetooth* necessários em dois dispositivos quaisquer. Nesse caso, é utilizado o *RFCOMM*. Esse protocolo essencialmente emula uma porta serial, e define uma série de procedimentos necessários para interoperabilidade que são coletivamente chamados de *Serial Port Profile*.

A placa possuí um bom suporte de software e firmware, possibilitado e mantido pela empresa que a produz. Algumas limitações de hardware ainda são realidade para essa tecnologia, fato que, por extensão, limita a escolha de placas de desenvolvimento no mercado. A tecnologia *BLE 5.0*, por mais que o protocolo da presente versão tenha sido homologado em 2016, ainda não é totalmente adotada e nem amplamente utilizada por vários dispositivos. Visto que o ecossistema do projeto já estava baseado na `LAUCHPAD-XL`, o projeto ali se deu prosseguimento.

3.2 Estudo de ambientes de programação, Sistema Operacional e exemplos de fabricante

A *TI* disponibiliza, em termos, o TIRTOS, um Sistema Operacional em Tempo Real (a partir daqui denominado como RTOS, ou *Real Time Operating System*, *Sistema Operacional em Tempo Real*, em tradução livre). Tal RTOS é uma peça de software capaz de gerenciar de forma inteligente, levando em consideração questões de tempo, rotinas e procedimentos escritos em linguagem de programação(INSTRUMENTS, s.d.). A empresa, no entanto, não disponibiliza o código aberto de tal RTOS, mas sim uma *API - Application Programming Interface*, que permite seu uso. Essa *API* contém *drivers*, ou camadas de abstração entre o código e o uso do hardware. Por exemplo, um driver de timer para controlar ciclos de tempo, ou um driver de periférico como *SPI*.

O princípio de funcionamento básico de um RTOS é fazer com que várias rotinas (também chamadas de funções) possam ser executadas em *quasi*-paralelismo. Ou seja, eficientemente alocando tempo de execução de código em um único processador para concluir várias tarefas diferentes, criando uma ilusão de paralelismo entre as mesmas. Além, a parte de "Tempo Real"se refere aos padrões rígidos que o sistema impõe para execução de tarefas.

Tais sistemas são de suma importância para diversas aplicações na indústria e na ciência, sendo exaustivamente utilizado em diversos setores, como robótica, equipamentos biomédicos e quaisquer sistemas sensíveis a requerimentos de tempo.

A empresa também disponibiliza exemplos de aplicação que podem ser extendidos e modificados livremente, sob licença aberta. Um desses é o *Simple Port Profile*, de nome homônimo ao padrão apresentado na seção 3.1, que é um

simples firmware que conecta duas plataformas *CC2640* e transfere texto numa configuração half-duplex (bidirecional, não simultâneo) entre as duas.

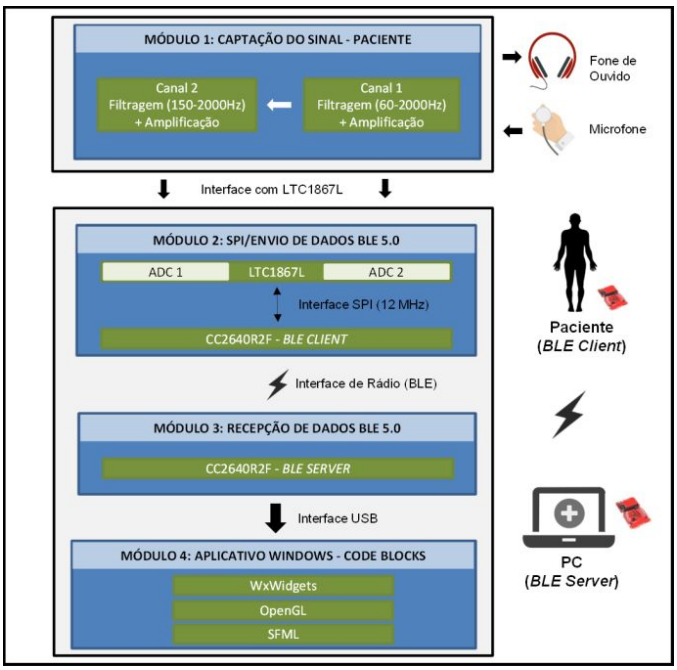
Esse exemplo foi compilado, estudado e analisado, visando compreendimento base do Sistema Operacional. O projeto também usa esse exemplo de aplicação como projeto-base.

3.3 Estudo do Sistema de transmissão por (MATTOS, 2018)

A partir disso, foram iniciados estudos sobre como o sistema de aquisição e transmissão criado por (MATTOS, 2018). Durante esse passo, notamos as limitações promovidas pela construção e configuração padrão do hardware da plataforma *CC2640R2*.

O principal esforço se deu a trabalhar dentro de limitações extremas de memória volátil. Com apenas 8`kb` de memória do tipo volátil, sendo que o Sistema Operacional em uso pode ocupar cerca de 25% desse valor, fazendo com que inserção de rotinas novas necessitem de cuidado para evitar possíveis *overflows*, ou estouro de pilha.

O sistema conta com um *ADC* de 16 *bits* de dois canais modelo *LTC1867LCGN* da *Linear Devices®*. Este se conecta a uma front-end desenvolvida por [trabalho que desenvolveu a frontend]



O projeto pode ser utilizado para se obter sinais de com pouca interferência e distúrbio para pacientes. Os sinais podem, *a posteriori* ser processados e analisados por técnicas específicas de processamento de sinal. A interface também pode ser vista como um estetoscópio eletrônico de baixo custo e performance comparável aos dispositivos existentes no mercado, que são de código fechado e valor proibitivo para uma grande parcela de possíveis aplicações.

A comunicação $ADC \rightarrow CC2640$ é realizada pelo padrão *SPI*, tal como citado na seção 2.1 com as seguintes configurações:

Foram corrigidos avisos gerados por conversões digitais implícitas que existiam nos valores amostrados do ADC quando levadas ao buffer de transmissão utilizado pelo projeto.

O projeto original também conta com um aplicativo com Windows NT como sistema host, que capturava os pacotes transmitidos via Bluetooth pelo *Client* do sistema por

emulação de porta serial via USB. O primeiro problema foi a falta de documentação sobre o projeto, que não provinha de instruções de compilação. As bibliotecas utilizadas no projeto também eram de versões antigas, com diversos problemas de interoperabilidade.

Ultrapassando o problema de compilação do projeto, percebemos que o mesmo utilizada a *API* do próprio sistema Windows NT. O código utilizado para acesso ao periférico da porta serial emulada sobre USB era datado, não se comportando bem em novas versões do sistema Windows NT. Decidimos então criar um novo programa que capture as informações de quaisquer porta serial, com a necessidade de fácil manutenção e pouca modificação entre sistemas operacionais host. Dessa forma, foi escrito um programa na linguagem de programação Python que obtém os dados da porta serial e os salva para pós-processamento, que mostrou bom desempenho.

Nessa programa também foi realizado um algoritmo simples de *poooling* ou espera para inicialização remota da gravação. Se necessário, métodos de pós-processamento podem ser programados de maneira não intrusiva no programa.

3.4 Extensão e aprimoramento do Sistema Desenvolvido

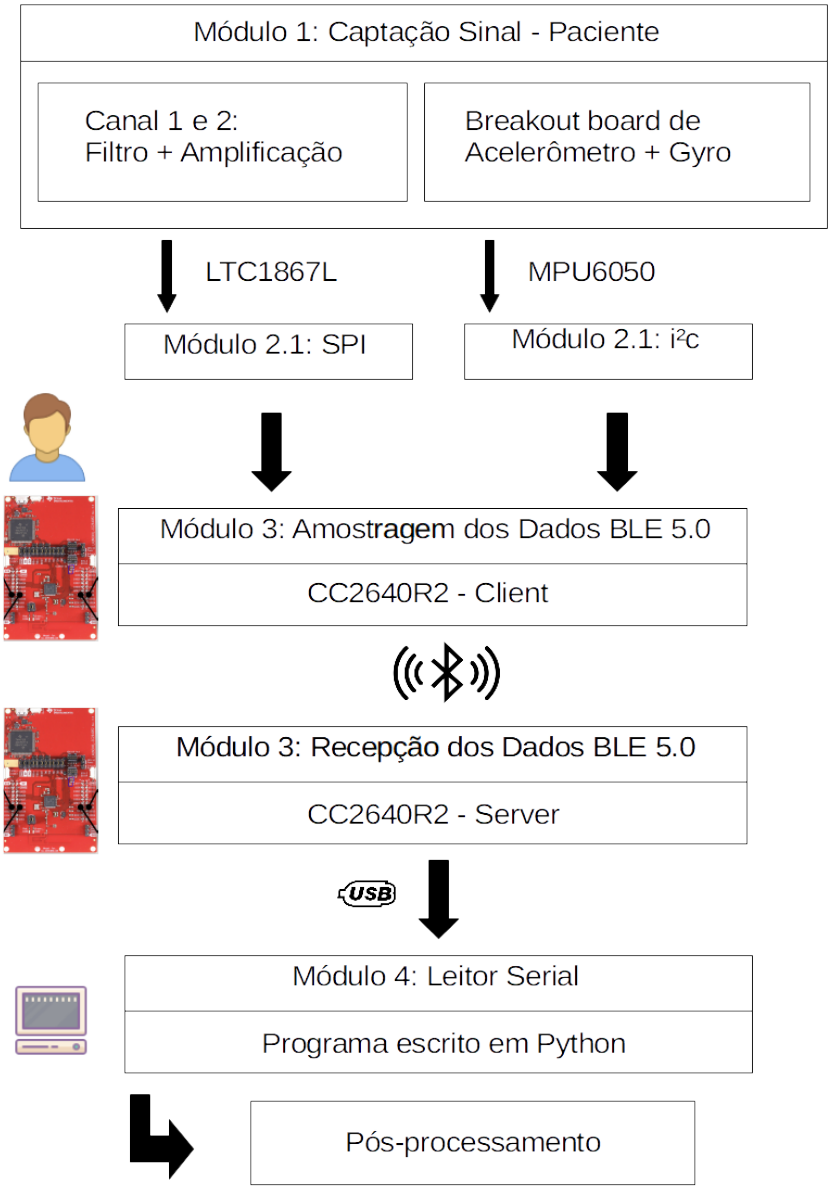
Uma das propostas concebidas pelo orientador foi a integração de um acelerômetro e giroscópio, de forma a obter dados de posição espacial do tórax do paciente. A partir dessa proposta foi elaborado uma interface parar aquisição de dados de pequenos sensores que contém tanto acelerômetro quando giroscópio.

O periférico de giroscópio e acelerômetro escolhido foi o

MPU6050. Seu baixo custo, alta precisão e plena disponibilidade oferecem um bom dispositivo. O *MPU6050* é conta com 6 *ADCs*, cada um de 16 *bits*. Cada um desses *ADCs* é responsável pela digitalização de um eixo do acelerômetro ou do giroscópio embarcados na *breakout board*. Também oferece um filtro passa-baixas digitalmente programável, descartando a necessidade de pós-processamento e baixo consumo (em torno de $3.6mA$ de acordo com a fabricante).

Nossa expansão incluiu o módulo de acelerômetro e giroscópio *MPU6050*, o que possibilita a captação de sinais de posição em três dimensões com n-graus de liberdade, onde n é o número de sensores acelerômetro e giroscópio disponíveis no sistema. Claramente existe um limite no número de sensores dado as limitações de memória e hardware disponíveis no sistema. Em específico, o MPU6050 possui um sistema simples de endereçamento, que permite endereçar até dois chips usando apenas um periférico i^2c. Como a plataforma atual possui dois desses periférico, é possível o endereçamento de até 4 combos de sensores de posição espacial.

A metodologia de programação desse periférico se deu no desenvolvimento de um *driver* do mesmo, que abstraí rotinas de baixo nível do periférico via uma *API*. Como no programa escrito em Python para recebimento dos dados pela serial, optamos por uma forte filosofia de portabilidade, escrevendo a biblioteca de uma maneira portável e agnóstica à plataforma, podendo ser utilizada em outros microcontroladores e até outros projetos.


Módulo 1: Captação Sinal - Paciente
Canal 1 e 2:
Filtro + Amplificação
Breakout board de
Acelerômetro + Gyro
LTC1867L
MPU6050
Módulo 2.1: SPI
Módulo 2.1: i²c
Módulo 3: Amostragem dos Dados BLE 5.0
CC2640R2 - Client
Módulo 3: Recepção dos Dados BLE 5.0
CC2640R2 - Server
USB
Módulo 4: Leitor Serial
Programa escrito em Python
Pós-processamento

Capítulo 4

RESULTADOS E DISCUSSÃO

As condições para realização dos testes não foi propícia devido aos eventos externos ocorridos no ano de 2020, impossibilitando tanto contato quanto acesso aos equipamentos e profissionais necessários para continuar os trabalhos normalmente, dadas as questões sanitárias. De qualquer forma, houveram testes regulares em todas as partes do projeto, o que garente o funcionamento em ambiente de desenvolvimento.

Ademais, a natureza de dispositivos embarcados é de baixo consumo e baixo processamento. Logo, a maior dificuldade em todas as etapas do projeto foi fazer uma alta manutenção de RAM (memória volátil). Ainda sim, a plataforma escolhida pode permitir comutações de código entre processadores e famílias de dispositivos, permitindo fácil portabilidade para microcontroladores com maior capacidade que venham a ser lançados suportando BLE 5.0.

Referências

AH, Omre. Bluetooth low energy: wireless connectivity for medical monitoring. **J Diabetes Sci Technol**, v. 4,2, n. 457-63, 2010.

BARRY, Richard. **Mastering the FreeRTOS™ Real Time Kernel**. [S.l.].

HOROWITZ, P.; HILL, W. **The Art of Electronics**. USA: Cambridge University Press, 1989. ISBN 0521370957.

INSTRUMENTS, Texas. **TI-RTOS Kernel (SYS/BIOS) User's Guide**. [S.l.].

MATTOS, Willian Daniel de. **Dissertação (mestrado)**. [S.l.: s.n.], 2018.

POPOVIC, Miroslav. **Communication Protocol Engineering**. USA: CRC Press, Inc., 2006. ISBN 0849398142.

TANENBAUM, Andrew S.; BOS, Herbert. **Modern Operating Systems**. 4th. USA: Prentice Hall Press, 2014. ISBN 013359162X.

VAN ROSSUM, Guido; DRAKE, Fred L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

Interface do Driver MPU6050

```
bool        mpu6050_test         ();
bool        mpu6050_writeByte    (char reg, char value);
int8_t      mpu6050_readByte     (char reg);
bool        mpu6050_reset        ();
bool        mpu6050_configAccel  (mpu6050_range_t range);
bool        mpu6050_configGyro   (mpu6050_dps_t range);
uint16_t    getAccel             (mpu6050_axis axis);
uint16_t    getGyro              (mpu6050_axis axis);
```