OSMAR DA CUNHA FILHO

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO APONTADOR TRIDIMENSIONAL COM INTERFACE USB

FLORIANÓPOLIS, 2012

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – IF-SC
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE PÓS-GRADUAÇÃO EM DESENVOLVIMENTO DE PRODUTOS ELETRÔNICOS

OSMAR DA CUNHA FILHO

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO APONTADOR TRIDIMENSIONAL COM INTERFACE USB

Monografia apresentada ao Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos como requisito parcial à obtenção do título de Especialista em Desenvolvimento de Produtos Eletrônicos

Orientador:
Prof. Fernando S. Pacheco, Dr. Eng.

FLORIANÓPOLIS, 2012

DESENVOLVIMENTO DE UM PROTÓTIPO DE DISPOSITIVO APONTADOR TRIDIMENSIONAL COM INTERFACE USB

OSMAR DA CUNHA FILHO

Este trabalho foi julgado adequado para obtenção do Título de Especialista em Desenvolvimento de Produtos Eletrônicos e aprovado na sua forma final pela banca examinadora do Curso de Pós-Graduação em Desenvolvimento de Produtos Eletrônicos do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 16 de Abril de 2012

Banca Examinadora:

_____

Prof. Fernando S. Pacheco, Dr. Eng.
Orientador

_____

Prof. Charles Borges de Lima, Dr. Eng.

_____

Prof. Joel Lacerda, Dr. Eng.

RESUMO

Os usuários de computadores com interface gráfica utilizam dispositivos de entrada para movimentar o cursor na tela, os chamados dispositivos apontadores. O mais comum deles é o *mouse*, dispositivo que geralmente necessita de uma superfície de apoio. Este trabalho descreve o desenvolvimento de *hardware* e *firmware* de um apontador que não necessita de uma superfície de apoio. O desenvolvimento do trabalho foi baseado na plataforma Arduino e separado em unidade fixa e unidade móvel. A unidade móvel é o dispositivo que será manuseado e possui um *firmware* que interpreta os sinais vindos de um acelerômetro, analisa e transmite-os sem fio à unidade fixa. A unidade fixa recebe o sinal e gera comandos de movimentação do cursor na tela repetindo o movimento do usuário. Os testes iniciais foram feitos com a unidade móvel ligada diretamente à unidade fixa para validar o algoritmo de cálculo e o protocolo de comunicação entre as placas. Em um segundo momento, acrescentou-se a comunicação via radiofrequência (RF) para validar a transmissão à distância. Sobre os resultados obtidos, o alcance foi de aproximadamente 6 metros e a duração da bateria da unidade móvel em transmissão ininterrupta foi de 2 horas. Foi feito um levantamento dos custos do protótipo e da fabricação do produto final além de algumas sugestões para implementações futuras.
Palavras-chave: Acelerômetro. Apontador. Arduino. Movimento.

ABSTRACT

Graphical interface computer users make use of input devices to move the cursor at the screen. These devices are called pointers. The most common of them is the mouse, device that usually requires a supporting surface to operate. This work describes the development of hardware and firmware of a pointer that doesn't require a supporting surface. The development was based on the Arduino platform and divided in a fixed unit and a mobile unit. The mobile unit is the device that will be handled by a person. It has a firmware that reads the signals coming from an accelerometer, analyze them and send them wirelessly to the fixed unit. The fixed unit receives the signal and generate commands to move the screen cursor repeating the user movement. Initially, the tests were done with the fixed unit connected directly to the mobile unit. This way it is able to validate the calculation algorithm and the communication protocol between units. At a second moment, the radio frequency was added to validate the data transmission at a distance. About the results obtained, the distance range was 6 meters and the battery duration was about 2 hours (on an uninterrupted transmission). By he end, there is a spreadsheet presenting the project costs for prototype and product and some suggestions for future works.

Keywords: Accelerometer. Arduino. Movement. Pointer.

ÍNDICE DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Os computadores são utilizados por usuários que operam dispositivos de entrada. É pelos dispositivos de entrada que o usuário insere informações no sistema. Nesta classificação de dispositivos estão os teclados, *joysticks*, *mouses*, *touchpads, scanners*, entre outros. O computador processa os dados de entrada e pode apresentar informações referentes ao que foi recebido (como teclas em um teclado e a posição de um dedo em um *touchpad).* Os dispositivos que movimentam o cursor também podem ser chamados de dispositivos apontadores e seu uso é comum em computadores pessoais. Um dos mais conhecidos dispositivos apontadores é o *mouse*, que foi inventado por Douglas Engelbart, em meados da década de 1960. Ele criou a patente americana 3541541 ("*X-Y position indicator for a display system*") registrada em 17 de Novembro de 1970 (UNITED STATES PATENT OFFICE, 1970).

Há modelos de *mouses* disponíveis no mercado que possuem características diferenciais como:

- mais botões (além dos dois botões de ação e do *scroll*), principalmente nas laterais;
- sensores mais precisos para captar o movimento utilizando *lasers* na unidade óptica e maior resolução em pontos por polegada *(dpi);*
- operação sem fio via Bluetooth;
- formato ergonômico, proporcionando um melhor ajuste à posição dos dedos; uso de lastros para regulagem do peso.

Há poucos modelos mais avançados de *mouses* que não necessitam de uma superfície de apoio. Eles geralmente funcionam captando o movimento com acelerômetros ou giroscópios dentro do produto. O movimento é captado por um acelerômetro que está ligado a um microcontrolador que lê e interpreta os sinais. Estes sinais são processados para transformar o valor lido em coordenadas na tela. A Figura 1 apresenta um exemplo de como é o sistema.

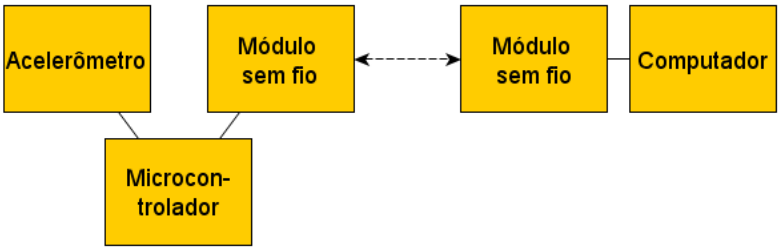

Figura 1 - Princípio de funcionamento do sistema. Fonte: autoria própria

Um levantamento de apontadores tridimensionais identificou produtos já disponíveis no mercado:

– Air Mouse Ellite (Gyration): design ambi-destro, conectividade wireless 2,4 GHz, alcance de até 30 m sem obstáculos (Figura 2).

Figura 2 - Air Mouse Ellite, da empresa Gyration.
Fonte: GYRATION, 2012.

– Mobile Mouse Pro (RPA Tech): aplicativo para plataformas móveis que roda em celulares Android ou iPhone, tornando o celular um *mouse* (Figura 3).

Figura 3 - Mobile Mouse Pro da empresa RPA Tech. Fonte: RPA TECH, 2012.

Este trabalho apresenta o desenvolvimento do *hardware* e *firmware* de um apontador tridimensional, isto é, um dispositivo alternativo ao *mouse* de um computador e que não necessita de uma superfície para ser utilizado.

A organização do trabalho foi feita nos tópicos de captura do movimento, transmissão de dados, microcontroladores e o desenvolvimento de *hardware* e *firmware*. A captura do movimento leva em questão as formas possíveis de captar o movimento utilizando componentes eletrônicos voltados para mensurar deslocamentos (angulares ou lineares) no espaço: giroscópios e acelerômetros. As formas de transmissão de dados sem fio englobam alternativas para a comunicação entre o dispositivo captador do movimento e sua respectiva unidade receptora conectada ao computador. O desenvolvimento destes elementos de captação do movimento e o posterior envio ao computador serão feitos por microcontroladores. Ao final da apresentação destes temas, discorre-se sobre o

desenvolvimento do *hardware* e *firmware*. Na sequência, faz-se uma análise dos testes, problemas encontrados, usabilidade, código-fonte e custos do projeto.

1.1 Justificativa

Em palestras e apresentações, faz-se necessário efetuar operações básicas de avançar e retroceder *slides*. Há dispositivos no mercado que efetuam esta tarefa. O objeto deste trabalho propõe uma ferramenta para apresentadores que substitui completamente um *mouse* e pode ser operado à distância, sem fios e sem uma superfície de apoio.

Este trabalho pode ser utilizado para auxiliar pessoas com necessidades especiais ou em tratamentos fisioterapêuticos. O movimento do corpo pode ser considerado um exercício que interage com um software ou jogo eletrônico que visa melhorar ou desenvolver habilidades motoras da pessoa que está usando o dispositivo (CÔRREA et al., 2008).

1.2 Definição do problema

Este trabalho procura responder à questão: como controlar um cursor de um computador à distância utilizando um dispositivo que não necessite de uma superfície de apoio?

1.3 Objetivos

O presente trabalho tem como objetivo geral o desenvolvimento de um circuito eletrônico (*hardware*) e *software* embarcado *(firmware)* capazes de realizar a captura de movimentos de forma sem fio e transformá-la em coordenadas para o cursor do computador.

Como objetivos específicos, têm-se:

- Desenvolver *firmware* de captação do movimento;
- Desenvolver *firmware* de movimentação do cursor;
- Montar *hardware* para comunicação;
- Testar funcionamento do conjunto de *hardware* e *firmware* com a comunicação sem fio.

2 CAPTAÇÃO DE MOVIMENTO

Este capítulo apresenta como o movimento pode ser utilizado como dado de entrada e alguns dispositivos que realizam esta operação de interpretar o movimento. Ao final, apresentam-se as alternativas para sensor de movimento e o sensor escolhido.

2.1 Movimento

O movimento como forma de entrada de dados é alvo de estudos desde que Paul Fitts publicou em 1954 um artigo que relaciona o movimento com a transmissão de informações. Fitts faz uma análise matemática e prevê que o tempo para mover um cursor até um alvo depende da distância do alvo e do tamanho deste alvo, considerando algumas variáveis, entre elas a velocidade de arranque e parada do dispositivo apontador, velocidade “em curso” do dispositivo e distância linear até o centro do alvo. Esta análise matemática originou o termo *Lei de Fitts*. A Lei de Fitts pode ser usada em várias áreas (cinemática, psicologia, ergonomia, interface homem-computador) para modelar o ato de apontar (MACKENZIE, 1992).

Na área de interface homem-computador há muito interesse na aplicação dos estudos de Fitts. Pode-se exemplificar isso pelo acesso a itens de menu de *pop-up*, onde um menu de *pop-up* aparece próximo ao cursor, de forma a minimizar o tempo de deslocamento do cursor aos itens de menu. Outro uso cabível dos estudos de Fitts são as interfaces de dispositivos móveis, onde botões e elementos de controle devem ser de tamanho adequado para que não dificultem o acesso. (HOOBER & BERKMAN, 2011).

2.2 Dispositivos apontadores

Dispositivos apontadores servem como uma interface humano-computador que permite a entrada de dados espaciais. Alguns exemplos de dispositivos apontadores:

- *mouse*: captura movimentos sobre uma superfície plana;
- *touchscreen*: tela sensível ao toque, que o usuário pressiona para efetuar as interações com o sistema;
- *joystick:* altera a posição do cursor conforme a posição ou força aplicados ao dispositivo.

Os dispositivos apontadores possuem sensores que captam o movimento. No caso do *mouse*, o sensor pode ser óptico por LEDs ou lasers, captando diferenças minuciosas na superfície sobre o qual o mouse está apoiado. Em certos modelos de *touchscreen,* utiliza-se um sensor capacitivo ou resistivo para determinar a posição do toque na tela.

2.3 Sensores de movimento

2.3.1 Giroscópio

Um giroscópio mede a orientação do eixo conforme o torque aplicado externamente. A operação de um giroscópio baseia-se no princípio da conservação do momento angular, onde o momento angular de um sistema relativo a um ponto fixo é constante se não houver forças externas sobre o sistema.

O giroscópio com rotor como é conhecido hoje (Figura 4) foi inventado em 1852 por Leon Foucault. O eixo de giro sempre possui uma referência e na maior parte das vezes é a superfície da Terra.

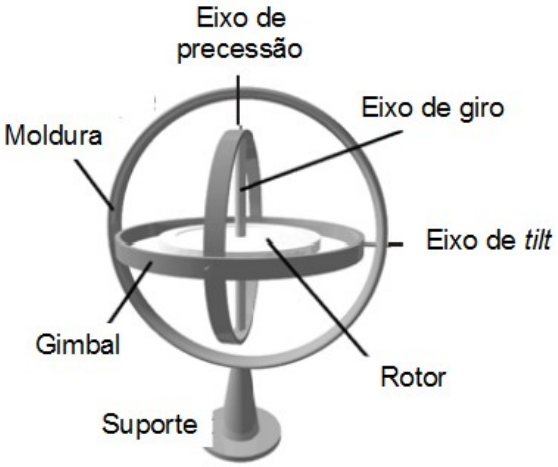

Figura 4 - Giroscópio com rotor. Fonte: STEFANESCU (2011)

O giroscópio com rotor é composto por um disco que gira livremente (rotor) em um eixo de giro (*spin axis*) presos em uma moldura (*frame*) que o deixa livre para girar em dois sentidos. A quantidade de sentidos (eixos) possíveis de giro determinam os graus de liberdade de um giroscópio, portanto o giroscópio da Figura 4 possui dois graus de liberdade. (FRADEN, 2010).

As tecnologias MEMS (*Micro Eletro-Mechanical System* – Sistemas microeletromecânicos) permitem a fabricação de giroscópios miniaturizados na forma de circuitos integrados. Um dos métodos para medir mudanças nos ângulos é substituir o disco giratório por elementos vibratórios. A Figura 5 apresenta como uma das implementações miniaturizadas de giroscópios pode ser feita.

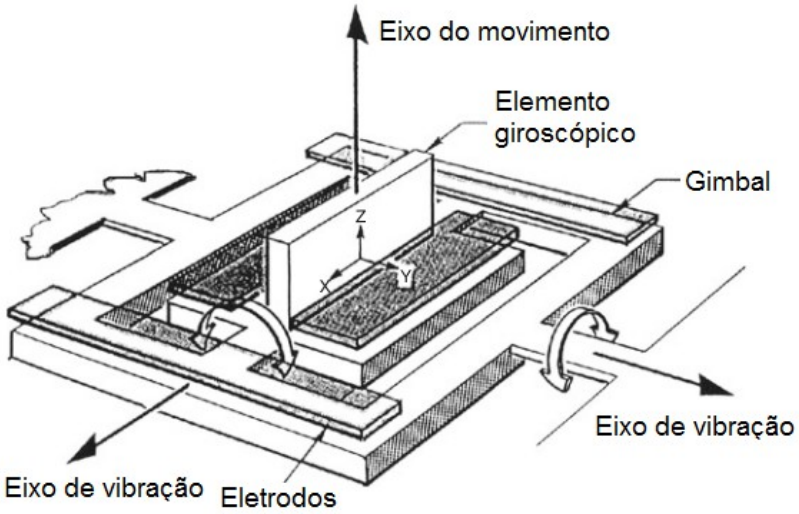

Figura 5 - Giroscópio por variação de vibração. Fonte: FRADEN, 2010.

A leitura dos movimentos é feita pela diferença na capacitância no par de eletrodos. Isso possibilita a miniaturização dos elementos mecânicos e também a fabricação em pastilhas de silício, resultando em giroscópios implementados em circuitos integrados.

2.3.2 Acelerômetro

Um acelerômetro é um transdutor que transforma variações de aceleração em sinais elétricos. A medida da aceleração é feita com uma parte móvel que exerce uma força em um elemento que está ligado a um sensor.

Para acelerômetros eletromecânicos há um núcleo magnético móvel e um sensor capacitivo que monitora o corpo do transdutor. Ao movê-lo, a massa móvel gera uma diferença na capacitância e o sensor capacitivo transforma essa diferença de capacitância em uma variação de tensão (WERNECK, 1996). A Figura 6 apresenta um diagrama de um acelerômetro

eletromecânico.

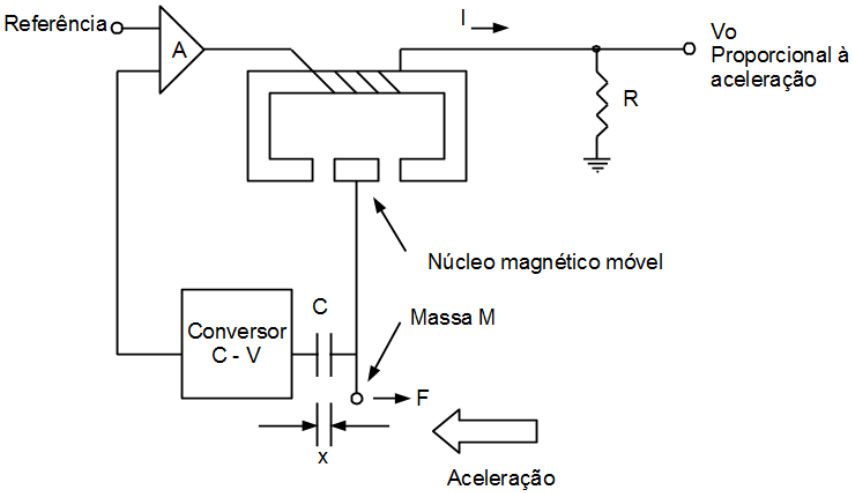

Figura 6 - Acelerômetro eletromecânico. Fonte: WERNECK, 1996

A aceleração é uma característica dinâmica dos corpos, pois conforme a segunda Lei de Newton, para apresentar uma aceleração, deve-se aplicar uma força ao corpo. Mesmo um corpo em um movimento com velocidade constante não necessariamente apresenta forças sendo aplicadas sobre ele. A aceleração, velocidade e posição estão relacionadas (velocidade é a primeira integral da aceleração e a posição é a segunda integral da aceleração). (FRADEN, 2010).

Acelerômetros podem ser fabricados utilizando processos MEMS. Uma das possibilidades de fabricação de circuitos miniaturizados consistem em eletrodos alternados dispostos em forma de pente. A Figura 7 ilustra este aspecto. Um conjunto de eletrodos é completamente fixo ao substrato do circuito integrado, enquanto outro conjunto fica suspenso, mas fixo por um único ponto ao substrato.

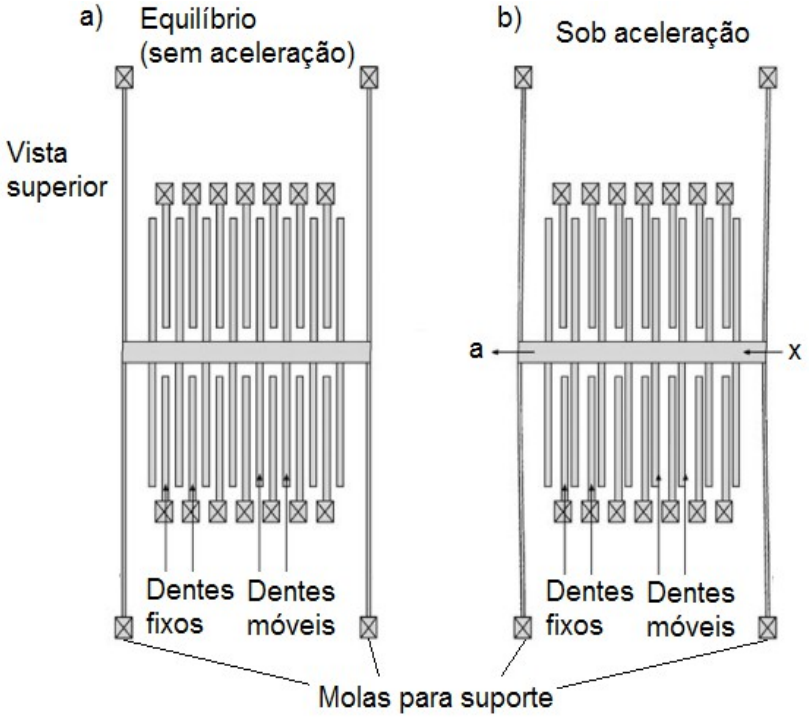

Figura 7 - Acelerômetro micromecânico de superfície (a) em repouso e (b) em aceleração. Fonte: BISHOP, 2008.

A Figura 7 a) mostra o acelerômetro em estado de repouso, onde a distância entre os dentes móveis ficam em um ponto equilibrado entre os dentes fixos. Ao aplicar uma aceleração (Figura 7 b), o sistema de molas move proporcionalmente o conjunto dos dentes móveis e aproxima os dentes móveis dos dentes fixos. Isso altera a capacitância entre os dentes, possibilitando a detecção do movimento.

3 COMUNICAÇÃO

Os dados captados e processados pelo dispositivo apontador devem ser transmitidos ao computador para que a posição do cursor na tela seja atualizada. Neste capítulo serão apresentadas as tecnologias Bluetooth e rádio-frequência para efetuar a transmissão sem fio e a tecnologia USB para repassar os dados ao computador.

3.1 Transmissão sem fio

3.1.1 Bluetooth

Bluetooth é uma implementação do padrão IEEE 802.15 de redes de baixo alcance (*PAN – Personal Area Network*) para transmissão de dados, voltada para uso por diversas aplicações como áudio, gráficos e vídeo. Nessas aplicações encontram-se diversos dispositivos: fones de ouvido, teclados, *mouses,* impressoras, etc. (STALLINGS, 2005).

O principal conceito do Bluetooth é criar uma rede universal sem fio de curto alcance: utilizando a frequência não licenciada de 2,4 GHz a distâncias de até 10 m pode-se chegar a velocidades de até 720 kbps. As principais aplicações podem ser classificadas em três grandes áreas:

- Pontos de acesso de dados e voz: transmissões em tempo real com comunicação entre dispositivos móveis e fixos;
- Substituição de cabos: conexões feitas sem necessariamente estar em linha de visão (como infravermelho) em um raio de 10m (podendo se estender a 100m com amplificadores);
- Rede ad hoc: um dispositivo pode conectar a outro para formar uma rede entre si.

A menor unidade básica de funcionamento chama-se *piconet* e é composta de um mestre e até sete escravos. O

dispositivo mestre define o canal em que os escravos irão se comunicar. Os dispositivos escravos somente podem comunicar-se com o mestre quando este permitir. Um dispositivo pertencente a uma *piconet* pode ser mestre ou escravo de uma outra *piconet*, originando uma *scatternet*. A Figura 8 mostra um exemplo de *piconet* e *scatternet*.

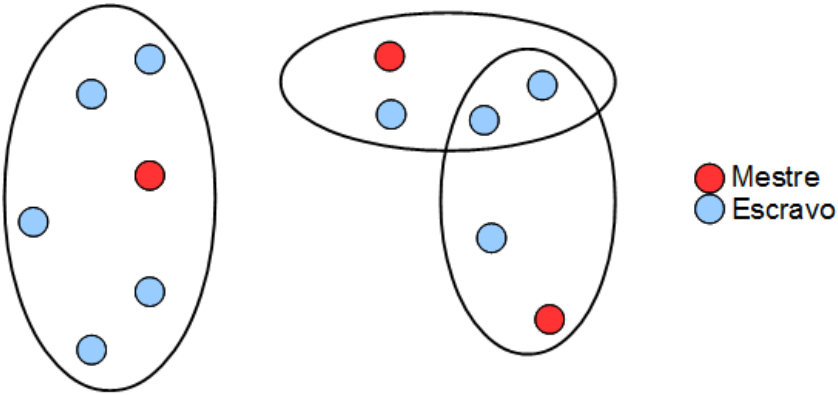

Figura 8 - Exemplo de piconet *(a)* e scatternet *(b)*. Fonte: autoria própria

O Bluetooth usa a banda de 2,4 GHz, que é a faixa aceita para aplicações de uso geral (ISM – *Industrial, Scientific and Medical*). A potência do sinal pode ser de três classes: 100 mW (Classe 1); 2,4 mW (Classe 2) ou 1 mW (Classe 3).

Como as aplicações de Bluetooth são variadas, no mercado comercializam-se módulos específicos para cada aplicação. A seguir alguns exemplos de módulos oferecidos:

– RN-41 da Roving Networks (Figura 9): implementa até a versão 2.1 da especificação Bluetooth, Classe 1 (alcance de 100 m), permite comunicação criptografada, possui comunicação via UART ou USB;

Figura 9 - Módulo RN-41. Fonte: ROVING NETWORKS, 2011

– LMX9838 da National Semiconductor (Figura 10): implementa apenas a versão 2.0 da especificação, Classe 2, possui interface de áudio;

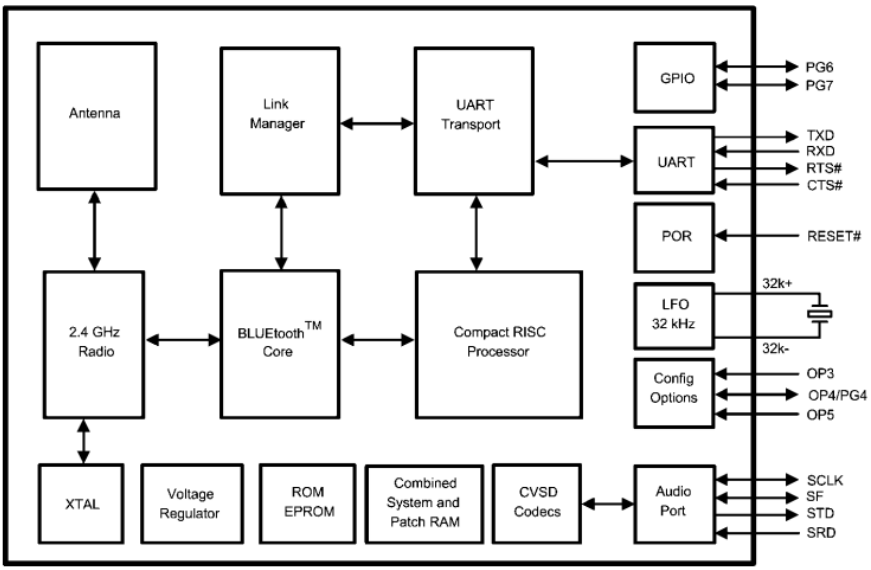

Figura 10 - Módulo LMX9838. Fonte: NATIONAL SEMICONDUCTOR, 2007

– WT12 da Bluegiga (Figura 11): implementa a versão 2.1 da especificação, Classe 2, possui comunicação via UART ou USB, permite atualização de firmware por SPI, possui interface de áudio;

Figura 11 - Módulo WT12. Fonte: BLUEGIGA TECHNOLOGIES OY, 2009

3.1.2 Rádio-frequência

As frequências para a comunicação via rádio-frequência (RF) são autorizadas por normas internacionais (*International Telecommunication Union* - ITU), mas não possuem um padrão ou norma específico que limite, restrinja ou defina quais são os parâmetros e protocolos na comunicação entre os dispositivos mestre e escravo. Existem padrões como o Bluetooth, Wi-Fi, ZigBee, mas o fabricante dos módulos receptor e transmissor pode implementar seu protocolo de comunicação.

Os fabricantes de módulos de comunicação sem fio que não implementam um padrão como ZigBee, Bluetooth ou Wi-Fi geralmente tentam facilitar a utilização do módulo com o modo "transparente". O modo transparente de alguns módulos apenas necessita que o usuário o configure uma vez e a partir daí o módulo estará pronto para enviar e receber dados de forma serial sem a necessidade de acrescentar cabeçalhos à carga útil ou acionar pinos para transmissão/recepção.

Há disponível no mercado diversos modelos de módulos RF que funcionam de modo transparente à aplicação, entre eles:

– RC1140-232 da Radiocrafts (Figura 12): baixo consumo (35 mA a 3,3 V), necessita apenas de antena externa, mínima interface UART com apenas 2 fios (RX e TX)

para facilitar uso como substituto de cabos, taxas de transferência de até 100 kbps;

Figura 12 - Módulo RC1140. Fonte: RADIOCRAFTS AS, 2010

– RDL2-433-32 da Radiometrix (Figura 13) consumo de 28 mA a 5 V, necessita de antena externa, taxas de transferência de até 32 kbps;

Figura 13 - Módulo RDL2. Fonte: RADIOMETRIX LTD, 2007

– TRM-433-LT da Linx (Figura 14): necessita taxas de transferência de até 10 kbps, interface direta para serial UART para uso como substituo de cabos:

Figura 14 - Módulo TRM. Fonte: LINX TECHNOLOGIES, INC, 2010

Esses módulos possuem um pino de recepção (RX) e um pino de transmissão (TX) que são responsáveis por receber um sinal serial TTL e transmiti-lo pelo meio (ar) e vice-versa. Os módulos citados não possuem antena embutida, sendo necessário acrescentá-la ao circuito do módulo.

3.2 USB – Universal Serial Bus

USB é um padrão para comunicação entre PC *(host)* e periféricos (AXELSON, 2001). Começou em 1996 com diversas motivações:

- diminuir a quantidade de cabos e pinagens para cada aparelho existente;
- ser rápida para que não seja um gargalo na comunicação;
- garantir que erros sejam raros e que possam ser corrigidos automaticamente;
- consumir pouca energia.

Depois de várias versões da especificação do padrão, há portas USB na maioria dos computadores pessoais e em diversos aparelhos como impressoras, dispositivos de armazenamento em massa (*pendrives*), *mouses,* entre outros.

A topologia da USB é organizada na forma de um barramento composto de elementos com papéis de dispositivo (*device)* e de *host*. Um dispositivo é um periférico no barramento que apenas responde requisições do *host* (exceto na funcionalidade de acordar remotamente, onde o dispositivo toma iniciativa e solicita resposta do *host*). O dispositivo deve detectar

dados direcionados a si, responder às requisições, gerenciar a própria energia, efetuar checagens de erros e trocar dados com o *host*.

Os *hosts* são elementos que se comunicam com os dispositivos no barramento, gerenciam as comunicações no barramento e formatam dados a serem transmitidos e recebidos de forma que os componentes do sistema operacional entendam. As responsabilidades do *host* são: detectar dispositivos, gerenciar o fluxo de dados, prover energia aos dispositivos, efetuar checagem de erros e trocar dados com dispositivos. (AXELSON, 2001).

O padrão USB possui uma classe de dispositivos específica para dispositivos de interface humana (*Human Interface Device* - HID). A classe HID do padrão USB não quer dizer unicamente que será um usuário humano que irá operar o dispositivo, mas para ser considerado um dispositivo USB de classe HID, deve-se atender a alguns requisitos (AXELSON, 2001):

- dados são trocados na forma de relatórios (*reports*), onde o *host* envia e recebe dados solicitando e recebendo relatórios;
- cada transação pode carregar 8, 64 ou 1024 bytes, dependendo da velocidade da comunicação USB;
- o dispositivo pode transmitir dados imprevisivelmente: o *host* não sabe exatamente quando o usuário pressiona uma tecla, por exemplo;
- a velocidade das transferências é limitada, mas as taxas de transferência não são garantidas: embora as transferências possam chegar a escalas de MB/s, as transações podem ocorrer a intervalos irregulares.

Este trabalho caracteriza-se como um dispositivo de classe HID pois atende aos requisitos para ser considerado um dispositivo desta classe, além de ser utilizado por um usuário humano. Os sistemas operacionais Microsoft Windows já incluem todos os *drivers* necessários para comunicar com dispositivos de classe HID

Como a interface USB está disponível em computadores, passa-se a considerar viável o uso da interface USB em projetos

microcontrolados. Utilizar portas USB do computador depende do dispositivo ser visto como um *device* para o computador. Alguns microcontroladores já possuem módulos dedicados para comunicação USB, atuando tanto como *host* ou como *device* podendo ser configurado dependendo da necessidade (chamado de USB OTG – *On The Go*). Além de microcontroladores com o módulo USB existem circuitos integrados que atuam apenas na conversão de protocolos de transmissão serial para USB. A seguir há uma lista de exemplos de microcontroladores e circuitos integrados que operam com USB:

– FT232R da FTDI (Figura 15): interface de transferência serial assíncrona para USB, gerador de clock e resistores da USB integrados ao CI, cria uma porta serial virtual no computador:

Figura 15 - Circuito integrado conversor UART - USB.
Fonte: FTDI, 2011.

– PIC18F4550 da Microchip (Figura 16): microcontrolador com USB compatível com versão 2.0 da especificação da USB, possui 32 KB de memória Flash, 2 KB de RAM, 4 *timers*, 2 comparadores, 13 canais de conversores A/D, comunicação via UART, SPI, I2C, LIN;

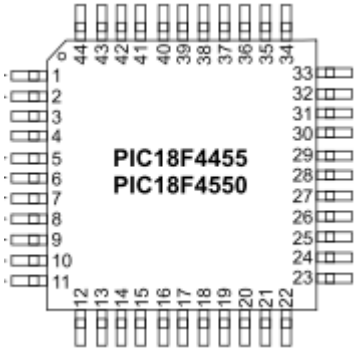

Figura 16 - Microcontrolador da família 18F da Microchip com suporte à USB. Fonte: MICROCHIP TECHNOLOGY, 2009.

– AT90USB da Atmel (Figura 17): microcontrolador com USB compatível com a versão 2.0 da especificação, 16 KB de Flash, 512 bytes de RAM, 2 timers, 1 comparador, comunicação via UART, SPI;

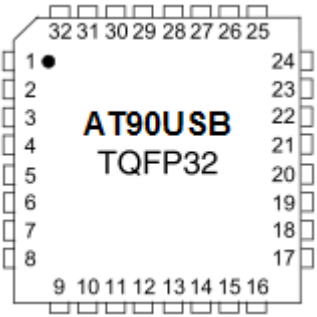

Figura 17 - Microcontrolador da família AT90USB da Atmel. Fonte: ATMEL CORPORATION, 2010.

Há circuitos integrados (como os citados nos exemplos) que possuem módulos de *hardwares* específicos para operar com a USB. Mesmo microcontroladores sem estes módulos de *hardware* podem ser capazes de implementar comunicação USB, como é o caso da implementação da USB na plataforma Arduino pela biblioteca V-USB (OBDEV, 2011).

4 MICROCONTROLADOR

Neste capítulo serão apresentadas duas plataformas de *hardware* que foram utilizadas para acelerar o desenvolvimento do *firmware*. Partindo de uma plataforma pronta, pode-se chegar mais rapidamente à fase de testes.

4.1 Plataforma Arduino

Arduino é uma plataforma para desenvolvimento e prototipagem de projetos eletrônicos criada em 2005 na Itália. Os criadores Massimo Banzi e David Cuartelles elaboraram um dispositivos para seus alunos testarem a interação com o ambiente de forma mais barata que prototipar cada placa desenvolvida. Esta plataforma oferece uma forma mais fácil de interagir com objetos, trazendo a computação para o mundo físico. Também possui uma interface gráfica e uma linguagem de programação diferenciadas de ambientes de desenvolvimento específicos para microcontroladores (como WinAVR, MPLAB), voltada para facilitar a entrada no mundo da computação física (ARDUINO, 2011).

A plataforma Arduino é baseada em um microcontrolador AVR. Aliado a esse microcontrolador, cada fabricante do *hardware* acrescenta itens para auxiliar o desenvolvimento de aplicações. A seguir alguns exemplos das diversas placas disponíveis no mercado:

- Arduino Uno (Figura 18): revisão da versão básica do Arduino, lançada em 2005. Dispõe de conectores nas laterais para conectar os *"shields",* que são placas para adicionar funcionalidades (Ethernet, ZigBee, controle de motores, etc) à placa original;

Figura 18 - Placa Arduino Uno. Fonte:
ARDUINO, 2011

- Arduino Duemilanove (Figura 19): lançada em 2009 e similar ao Arduino Uno. A diferença principal está no circuito de comunicação via USB: no modelo Uno, é realizada por um microcontrolador ATmega; Duemilanove emprega um *chip* conversor USB-serial da FTDI*;*

Figura 19 - Placa Arduino Duemilanove. Fonte:
ARDUINO, 2011

- LilyPad Arduino (Figura 20): placa desenvolvida para ser costurada em roupas e utilizada em conjunto com outras placas similares conectadas com fios condutivos.

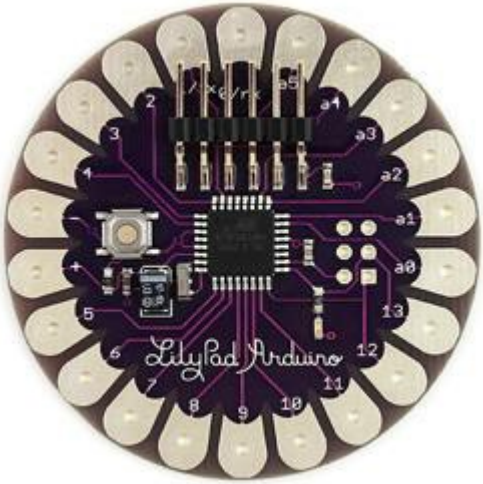

Figura 20 - Placa lilyPad Arduino. Fonte: ARDUINO, 2011

4.1.1 Biblioteca V-USB

A biblioteca V-USB foi desenvolvida para que microcontroladores da linha AVR pudessem realizar comunicação via interface USB sem o uso de nenhum circuito integrado adicional.

A principal vantagem da biblioteca V-USB é a redução de custos de hardware, tornando desnecessário o uso de circuitos integrados adicionais para efetuar a comunicação USB no projeto. Por exemplo, evita-se o uso de um chip para conversão de serial para USB (da empresa FTDI, por exemplo). Outra alternativa que passa a ser desnecessária é o uso de microcontroladores com USB integrada, como os da família AT90 da Atmel.

Esta biblioteca implementa a comunicação compatível com a versão 1.1 do protocolo USB para microcontroladores de baixa velocidade da linha AVR. Os requisitos mínimos para o

microcontrolador suportar a V-USB são 2 kB de memória Flash, 128 bytes de RAM e frequência *clock* de 12 MHz. Não é necessário o uso de portas seriais, *timers*, ou módulo de *hardware* especial, exceto por uma interrupção por borda (OBDEV, 2011).

As desvantagens no uso da V-USB são as próprias limitações que ela impõe: versão da especificação da USB é 1.1 (e não 2.0, que aceita velocidades e formas de transferência mais rápidas) e deve-se utilizar microcontroladores da família AVR que atendam aos requisitos mínimos de memória e periféricos.

4.2 Plataforma ArduIMU

ArduIMU é uma plataforma de *hardware* (Figura 21) dotada de sensores para efetuar medidas inerciais que podem ser utilizadas em controle de atitude. Este *hardware* inclui um processador compatível com Arduino (ARDUIMU, 2011).

O ArduIMU possui um acelerômetro de três eixos e dois giroscópios (para efetuar medidas nos três eixos). Em conjunto ao acelerômetro e giroscópio, pode-se conectar um aparelho GPS ao ArduIMU para manter uma orientação da placa no espaço, considerando que a placa pode ser utilizada em aeromodelos. Além disso, a placa possui duas saídas PWM para acionamento de motores possibilitando realizar o controle dos flapes ou hélices de um aeroplano, por exemplo.

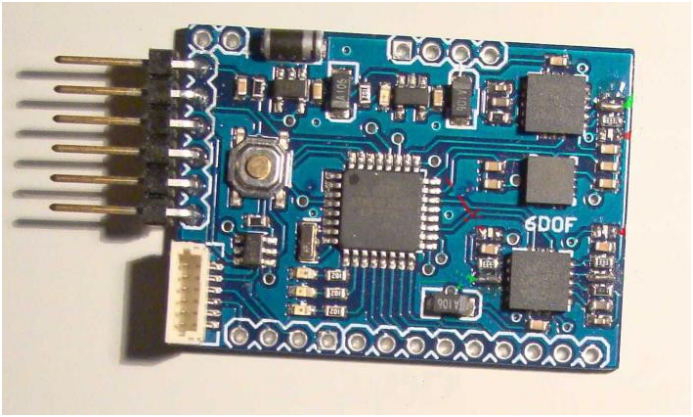

Figura 21 - Placa ArduIMU. Fonte: ARDUIMU, 2011

O acelerômetro disponível na placa ArduIMU é do fabricante Analog Devices, modelo ADXL335. Na Figura 22 consta um diagrama funcional do circuito integrado.

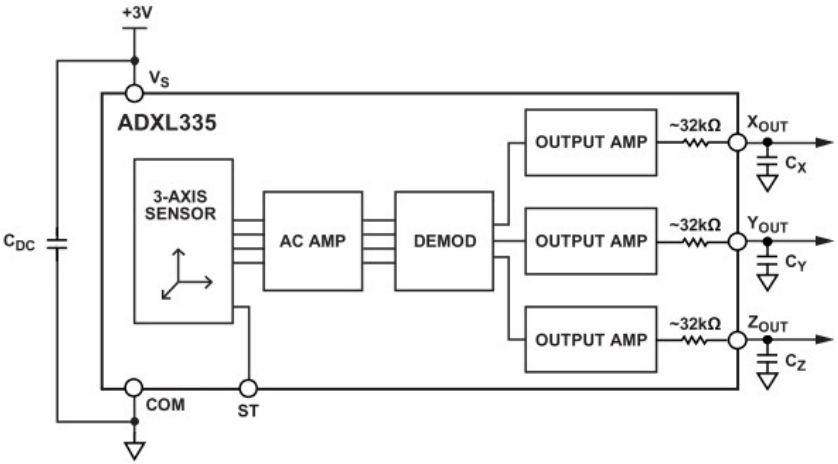

Figura 22 - Diagrama funcional do ADXL335. Fonte: ANALOG DEVICES, 2010

Os sinais dos sensores passam por etapas de amplificação e filtragem antes de serem disponibilizados nos pinos de saída. Os eixos estão definidos pelo fabricante e seguem a orientação

apresentada na Figura 23.

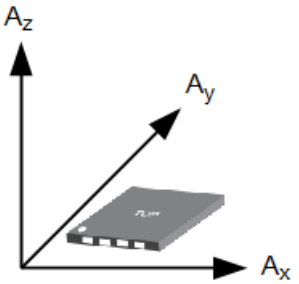

Figura 23 - Orientação dos eixos do ADXL335. Fonte: ANALOG DEVICES, 2010

A aceleração aplicada em cada eixo fica disponível em um pino específico do componente.

5 DESENVOLVIMENTO

O objeto que será utilizado pelo usuário e sobre o qual será aplicado o movimento será chamado de Unidade Móvel. A unidade que fica conectada ao computador será chamada de Unidade Fixa. Para cada uma delas serão apresentados detalhes do desenvolvimento do *firmware* e do *hardware.*

A Figura 24 demonstra de forma geral os componentes e como é a sequência de funcionamento do sistema.

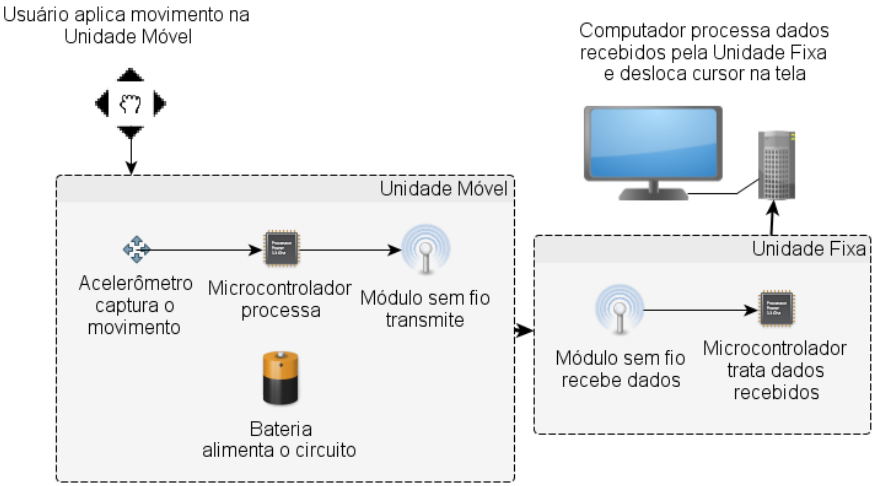

Figura 24 - Componentes do sistema e fluxo de dados. Fonte: autoria própria

A Unidade Móvel é a responsável pela captação, processamento e transmissão do movimento. Quando o usuário executa um movimento sobre esta unidade, o acelerômetro capta este movimento e atribui valores aos seus pinos de saída. O microcontrolador lê estes pinos, calcula o deslocamento feito, prepara os dados do deslocamento e transmite de forma sem fio à Unidade Fixa.

A Unidade Fixa tem por objetivo receber os dados do deslocamento pelo módulo sem fio e comunicar (via USB) ao sistema operacional do computador que houve uma alteração na

posição do cursor ou algum botão pressionado.

5.1 Unidade Móvel

5.1.1 Firmware

O desenvolvimento do *firmware* foi feito na interface de programação Arduino, que utiliza a linguagem Wiring. A sintaxe da linguagem é similar à programação C++.

O acelerômetro tem como saída pinos analógicos com o valor da aceleração instantânea. O interesse do trabalho é obter o deslocamento a partir de um movimento aplicado ao acelerômetro. Para isso, aplica-se uma integral dupla sobre o sinal: a primeira integral resulta na velocidade e a segunda, na posição (FREESCALE SEMICONDUCTOR, 2007). A integral foi calculada utilizando a aproximação trapezoidal para cálculo de integrais definidas conforme a fórmula a seguir:

$$\int_a^b f(x)\,dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

Onde:

(b - a): representa a diferença de tempo entre cada amostra. Como este é um sistema síncrono e todas a medidas serão tomadas a intervalos regulares, considera-se este valor como 1.

f(a): representa a base menor do trapézio; considerado como o valor lido da iteração anterior;

f(b): representa a base maior do trapézio; considerado como o valor lido da iteração atual;

A Figura 25 apresenta de forma gráfica como é a interpretação da aproximação trapezoidal para o cálculo da integral.

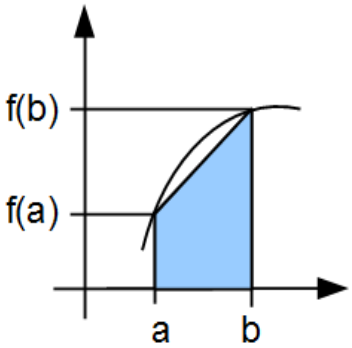

Figura 25 - Cálculo da integral por aproximação trapezoidal.
Fonte: autoria própria.

O código que é executado na Unidade Móvel lê os sensores e envia pela porta serial (que está conectada a um transceptor RF). A Figura 26 apresenta um fluxograma do processo de leitura.

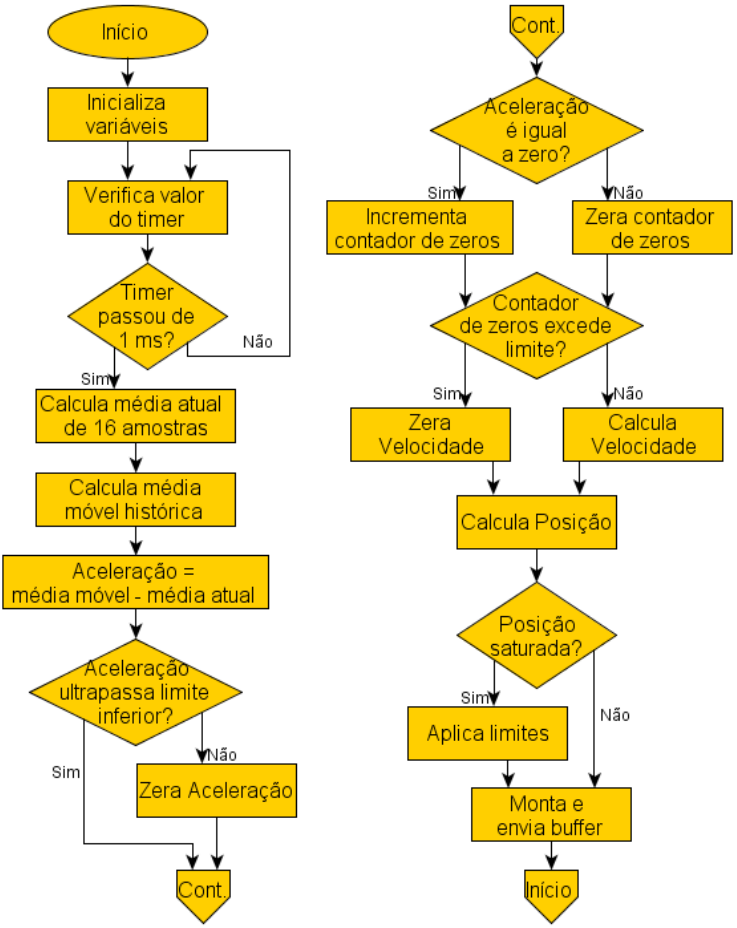

Figura 26 - Fluxograma do firmware da Unidade Móvel. Fonte: autoria própria.

Após inicializar, o *firmware* verifica o estouro de um *timer.* Quando estoura, começa a etapa de cálculo. O cálculo começa realizando 16 leituras e calculando a média delas. Este valor será utilizado como o *offset* para descontar da média móvel que é armazenada a cada etapa de cálculo. O valor da aceleração é obtido diminuindo a média atual (*offset*) da média móvel.

Se a aceleração calculada ultrapassar um limite mínimo, ela passa a ser considerada, caso contrário ela é zerada. Esta etapa é mais um filtro para eliminar pequenas movimentações. Cada valor zerado incrementa um contador. Quando este contador excede um valor fixo, considera-se que a velocidade é zero. Isto garante que quando uma quantidade determinada de sinais de aceleração forem zero, considera-se que o movimento cessou, portanto a velocidade também é zero. Se o contador de zeros não for excedido, pode-se calcular a velocidade (pela fórmula da integral).

Para calcular a posição, aplica-se a mesma fórmula que foi aplicada anteriormente. A posição calculada ainda é subtraída da posição calculada na etapa passada, para enviar somente a variação do movimento entre o instante anterior e o presente. Depois de calcular a posição, faz-se um ajuste de escala para evitar que o valor extrapole os limites do tipo de dado que será armazenado e transmitido via RF.

A Figura 27 apresenta os sinais obtidos para cada passo do fluxo aplicando um movimento à Unidade Móvel. O movimento aplicado consiste basicamente em um quadrado:

- Movimento de baixo para cima entre as amostras 76 e 121 (aproximadamente): deslocamento no eixo X;
- Movimento da esquerda para direita entre as amostras 136 e 181 (aproximadamente): deslocamento no eixo Y;
- Movimento de cima para baixo entre as amostras 196 e 226 (aproximadamente): deslocamento no eixo X;
- Movimento da direita para esquerda entre as amostras 241 e 286 (aproximadamente): deslocamento no eixo Y;

A Figura 27 ilustra a cada gráfico duas curvas, uma com os valores do eixo X e outra com os valores do eixo Y:

a) a aceleração após o cálculo da média móvel;

b) velocidade calculada pela integral da aceleração, se o contador de zeros não exceder o limite;

c) posição calculada pela integral da velocidade;

d) deslocamento: é a variação do valor da posição que será armazenado no *buffer* para o posterior envio.

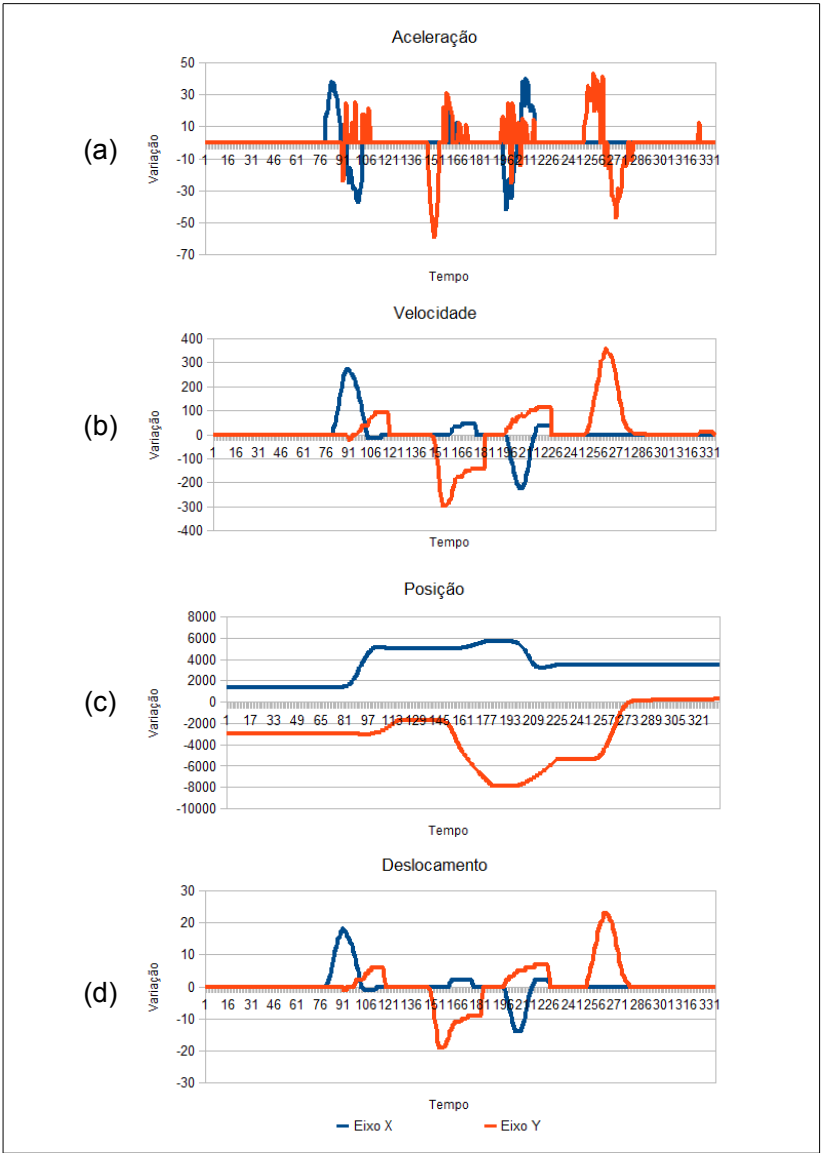

Figura 27 - Gráficos obtidos a partir da aplicação do cálculo. Fonte: autoria própria

A Figura 28 apresenta setas indicando o sentido do movimento no gráfico do deslocamento (gráfico (d) da Figura 27).

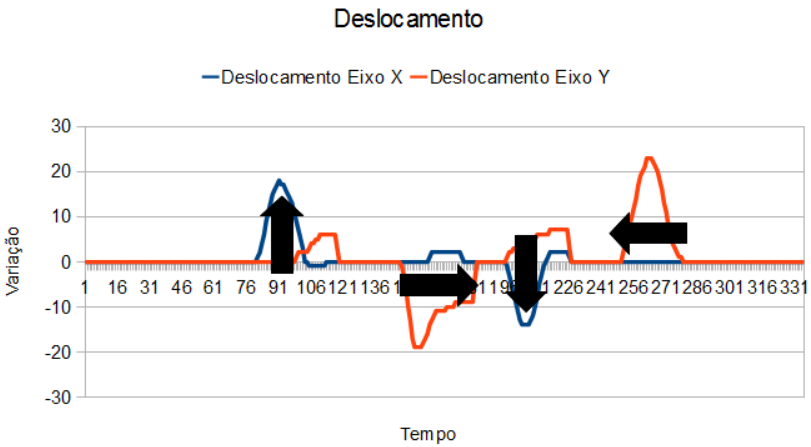

Figura 28 - Gráfico do deslocamento com setas indicando o movimento. Fonte: autoria própria.

Nota-se que mesmo os movimentos aplicados em apenas um eixo geram um pequeno sinal no outro eixo. Por este sinal não ter amplitude nem duração superior a do sinal do movimento em questão, pode-se desconsiderá-lo.

Após o tratamento aplicado, monta-se o *buffer* de dados para a transmissão à Unidade Fixa. O *buffer* é composto dos seguintes elementos:

- Início: 1 *byte* com o valor 'A' (ASCII);
- Posição X: 1 *byte* com o valor do deslocamento no eixo horizontal variando de -127 a 127;
- Posição Y: 1 *byte* com o valor do deslocamento no eixo vertical variando de -127 a 127;
- Botão 1: 1 *byte* com o status do botão 1 ('N' para pressionado ou 'F' para não pressionado, em ASCII);
- Botão 2: 1 *byte* com o status do botão 2 ('N' para pressionado ou 'F' para não pressionado, em ASCII);

– Fim: 1 *byte* com o valor 'B' (ASCII);
– *Checksum:* 1 *byte* com um cálculo de soma de verificação aplicando uma operação de ou-exclusivo *byte-a-byte* com os *bytes* das posições 0 a 5 do *buffer.*

A Tabela 1 mostra a organização do *buffer.*

Tabela 1 - *Buffer* dos dados. Fonte: autoria própria.

Posição no *buffer*	**Descrição**	**Valores**	**Tamanho**
0	Início	'A'	1 *byte*
1	Posição X	-127 a 127	1 *byte*
2	Posição Y	-127 a 127	1 *byte*
3	Botão 1	'N' ou 'P'	1 *byte*
4	Botão 2	'N' ou 'P'	1 *byte*
5	Fim	'B'	1 *byte*
6	*Checksum*	Valor calculado	1 *byte*

Os limites de -127 a 127 para o deslocamento foram estabelecidos porque a posição calculada é uma diferença entre o valor calculado nesta iteração e o valor da iteração anterior.

Com o *buffer* montado, este é enviado pelo pino TX da porta serial da placa ArduIMU. A placa transceptor envia o sinal via RF à Unidade Fixa.

O código-fonte originou um código executável de 4248 bytes.

5.1.2 Hardware

O *hardware* da Unidade Móvel é composto dos seguintes itens (a Figura 29 exibe um diagrama de blocos dos componentes do *hardware*):

– placa ArduIMU (que possui o microcontrolador principal e

o acelerômetro);
- placa transceptora que faz a comunicação via RF;
- três botões: dois para simular os botões do *mouse* e mais um para indicar que a Unidade Móvel deve transmitir o movimento;
- baterias: quatro pilhas AA recarregáveis para alimentação.

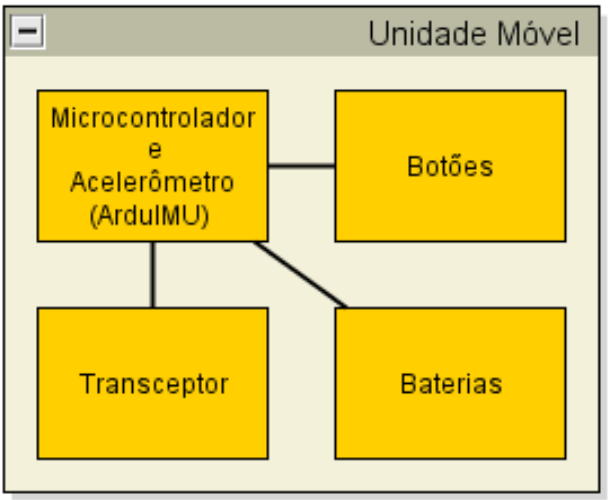

Figura 29 - Diagrama de blocos do hardware da Unidade Móvel.
Fonte: autoria própria.

Os três botões foram ligados aos pinos PWM1, MOSI e MISO da placa ArduIMU. Estes pinos representam respectivamente os pinos PB2, PB3 e PB4 do microcontrolador ATmega da placa. Os três pinos tiveram o *pull-up* interno do microcontrolador ativado para evitar que o pino tenha um sinal flutuando e para que não seja necessário montar um circuito de *pull-up* externo. A Figura 30 ilustra a conexão dos botões ao circuito.

O terceiro botão da Unidade Móvel tem o intuito de garantir

que movimentações indesejadas não sejam transmitidas à Unidade Fixa. Ou seja, quando o usuário desejar mover o cursor, deve pressionar e segurar o botão do movimento.

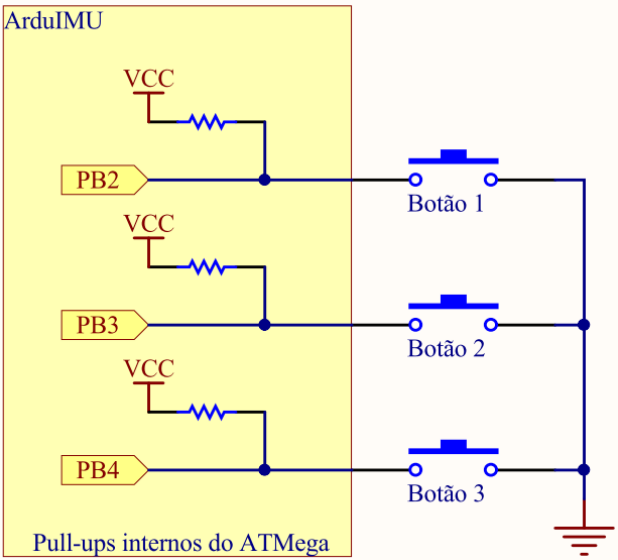

Figura 30 - Conexão dos botões à Unidade Móvel. Fonte: autoria própria.

As quatro pilhas recarregáveis totalizam um fornecimento de 4,8 V (1,2 V cada pilha) para o circuito. Como os circuitos foram dimensionados para funcionarem com 5 V, a tensão de 4,8 V permite a operação do circuito.

Após uma avaliação das opções de módulos Bluetooth e RF apresentadas, optou-se pelos seguintes critérios para utilizar o módulo RC1140 da Radiocrafts, que faz comunicação via RF:

- menor custo comparado a outros módulos;
- maior disponibilidade no mercado;
- facilidade de montagem e integração ao projeto.

A placa transceptora possui um circuito adicional de *driver* de sinal (MAX3232) e um regulador de 3,3 V na placa. O elemento mais importante da placa é o módulo transceptor RC1140-RC232. Ele faz a decodificação do sinal RF e

transforma-o em sinal TTL. O módulo transceptor RC1140-RC232 possui as seguintes características (RADIOCRAFTS AS, 2010):

- alimentação de 3,3 V;
- opera na faixa de 433 MHz;
- funciona de forma transparente: pode enviar e receber caracteres apenas utilizando os pinos TX e RX do módulo;
- aceita várias taxas de transmissão para comunicação.

Como qualquer sistema de comunicação sem fio, esse transceptor precisa de uma antena conectada a ele. O fabricante sugere uma antena fio de um quarto de onda, resultando em uma antena de 14 cm (calculados a partir da frequência de 433 MHz).

A Figura 31 apresenta o diagrama esquemático do circuito completo proposto da Unidade Móvel. Apenas acrescentou-se a interface de gravação e um LED de operação.

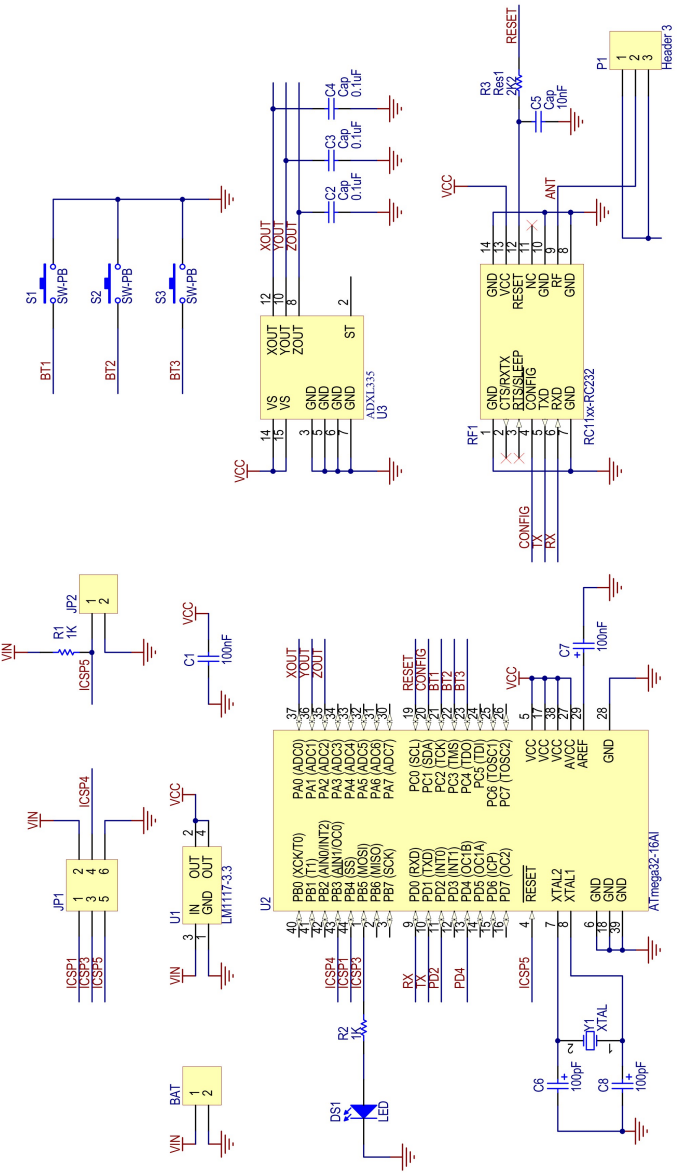

Figura 31 - Diagrama esquemático da Unidade Móvel. Fonte: autoria própria

O diagrama esquemático da Figura 31 dá origem ao circuito traçado na Figura 32.

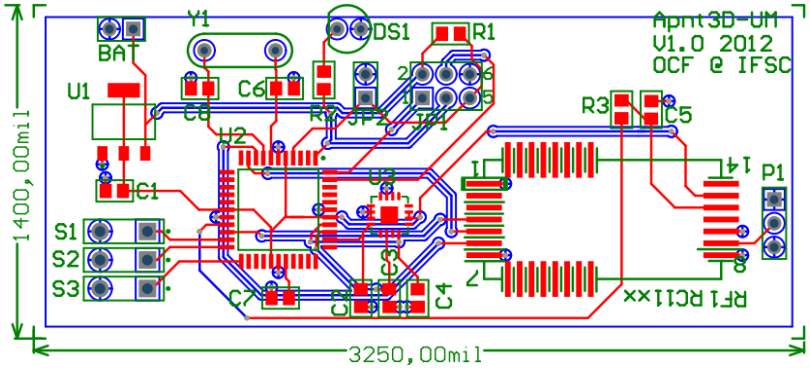

Figura 32 - Placa de circuito impresso da Unidade Móvel. Fonte: autoria própria.

As medidas da placa estão em milésimos de polegadas (mils) e convertendo estes valores para milímetros resulta em um tamanho de 82,55 mm por 35,56 mm. O circuito da Figura 32 possui um plano de terra na camada inferior da placa. Na figura, apenas os contornos do plano são exibidos.

5.1.3 Aspectos ergonômicos

O formato do gabinete da Unidade Móvel deve levar em conta a facilidade de ser segurado com uma mão, manuseado e ter os botões pressionados. No gabinete deve haver espaço para o circuito, antena e suporte para as quatro pilhas AA. Levando em conta essas características, o gabinete deve ser leve o suficiente para que o usuário não canse com pouco tempo de uso. O formato do gabinete deve prover uma forma ergonômica de segurar a unidade mantendo a mão o menos contraída possível. Os pressionamentos de botões devem ser similares aos pressionamentos em um *mouse* para aproveitar a experiência do usuário na ação de movimentar um cursor e clicar.

A Figura 33 apresenta algumas sugestões de modelos para o gabinete da Unidade Móvel.

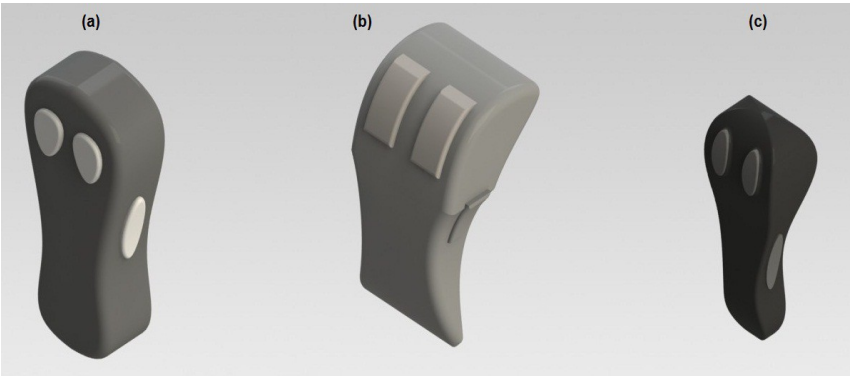

Figura 33 - Exemplos de desenhos mecânicos para a Unidade Móvel. Fonte: autoria própria

O modelo (a) da Figura 33 baseia-se em um conceito equilibrado entre *mouse* e apontador. Sua face superior plana assegura que somente a ponta dos dedos toquem os botões superiores.

O modelo (b) aproxima-se de um *mouse*, em que os botões possuem uma extensão maior para contato com os dedos. O corpo apresenta uma curva para deixar a palma mão levemente curvada para pousá-la sobre a face superior do dispositivo. As faces laterais são curvadas para apoiar o polegar.

O modelo (c) tem traços influenciados por um dispositivo apontador para apresentações, em que seu principal uso não é sobre uma superfície. O corpo, por ser mais estreito permite que os dedos envolvam o corpo do dispositivo para segurá-lo.

Os botões laterais são os botões que indicam o movimento. Os botões frontais operam com os botões de ação de um *mouse*.

5.2 Unidade Fixa

5.2.1 Firmware

O ambiente de desenvolvimento do *firmware* é o mesmo utilizado na Unidade Móvel porque esta placa também é baseada na plataforma Arduino.

Este *firmware* utiliza a biblioteca V-USB para a comunicação USB com o computador. O fluxograma apresentado na Figura 34 contém o fluxo do *firmware* que está sendo executado no microcontrolador da placa Arduino.

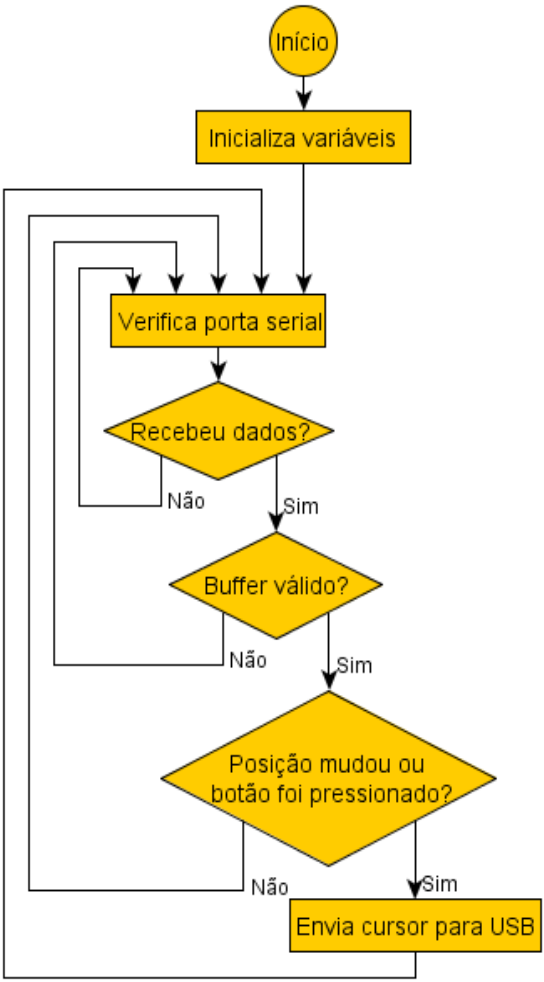

Figura 34 - Fluxograma do firmware da Unidade Fixa. Fonte: autoria própria.

O uso da biblioteca V-USB de forma alternativa à porta USB já presente na placa Arduino deve-se à redução de custos de *hardware* que essa biblioteca oferece, eliminando circuitos integrados adicionais para comunicação USB. Isso faz com que o

circuito final tenha menos componentes e consequentemente permita a diminuição do tamanho da placa.

O *buffer* é considerado válido se os caracteres de início e fim do *buffer* estão nas posições corretas e são os caracteres definidos para este fim ('A' e 'B', respectivamente) conforme a Tabela 1 que detalha os elementos do *buffer.*

Se a posição em qualquer um dos eixos mudou ou se algum botão do *mouse* foi pressionado, a Unidade Fixa envia o relatório (*report*) à USB, que por sua vez movimenta o cursor na tela.

O código fonte da unidade fixa não ultrapassou 4 KB (3786 bytes).

5.2.2 Hardware

O *hardware* que serve de base é a placa Arduino Duemilanove (ilustrado na Figura 19). A Figura 35 exibe um diagrama de blocos do circuito implementado neste trabalho. Os itens que compõem o *hardware* são:

- placa Arduino: onde está o microcontrolador principal;
- placa transceptora: responsável pela comunicação via RF;
- circuito da interface USB: circuito adicional necessário para a comunicação USB entre a placa Arduino e o computador.

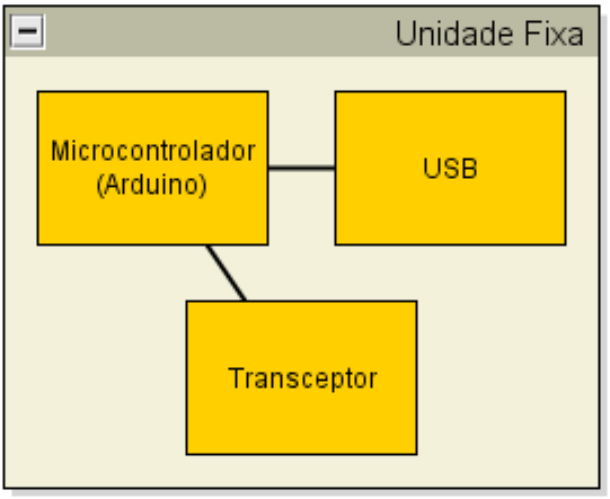

Figura 35 - Diagrama de blocos do hardware da Unidade Fixa. Fonte: autoria própria.

O circuito adicional da interface USB é composto de apenas dois diodos Zener e três resistores. Esse circuito garante os níveis adequados de sinal na USB e faz o *pull-up* na linha do sinal “D-”. Os pinos da placa Arduino que foram utilizados são os pinos padrão da implementação da biblioteca V-USB (PD2 e PD4). O circuito está apresentado na Figura 36.

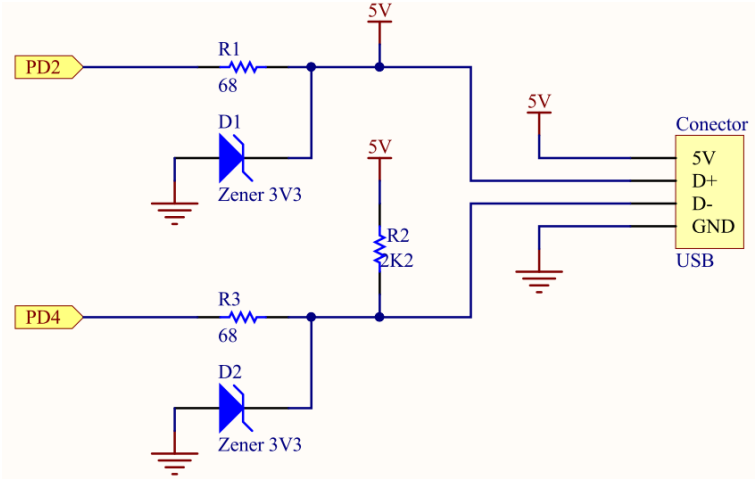

Figura 36 - Diagrama esquemático da USB. Fonte: PRACTICAL ARDUINO, 2011.

A placa Arduino é alimentada pelo conector USB do circuito adicional, não sendo necessário ocupar duas portas USB para a alimentação. A placa transceptora também utiliza a alimentação de 5V que vem da USB.

O diagrama esquemático do *hardware* proposto por esse trabalho está apresentado na Figura 37. Foram acrescentados ao diagrama: a interface de programação do microcontrolador, um regulador para 3,3 V, um botão com a função de *reset* e um LED para indicação de operação.

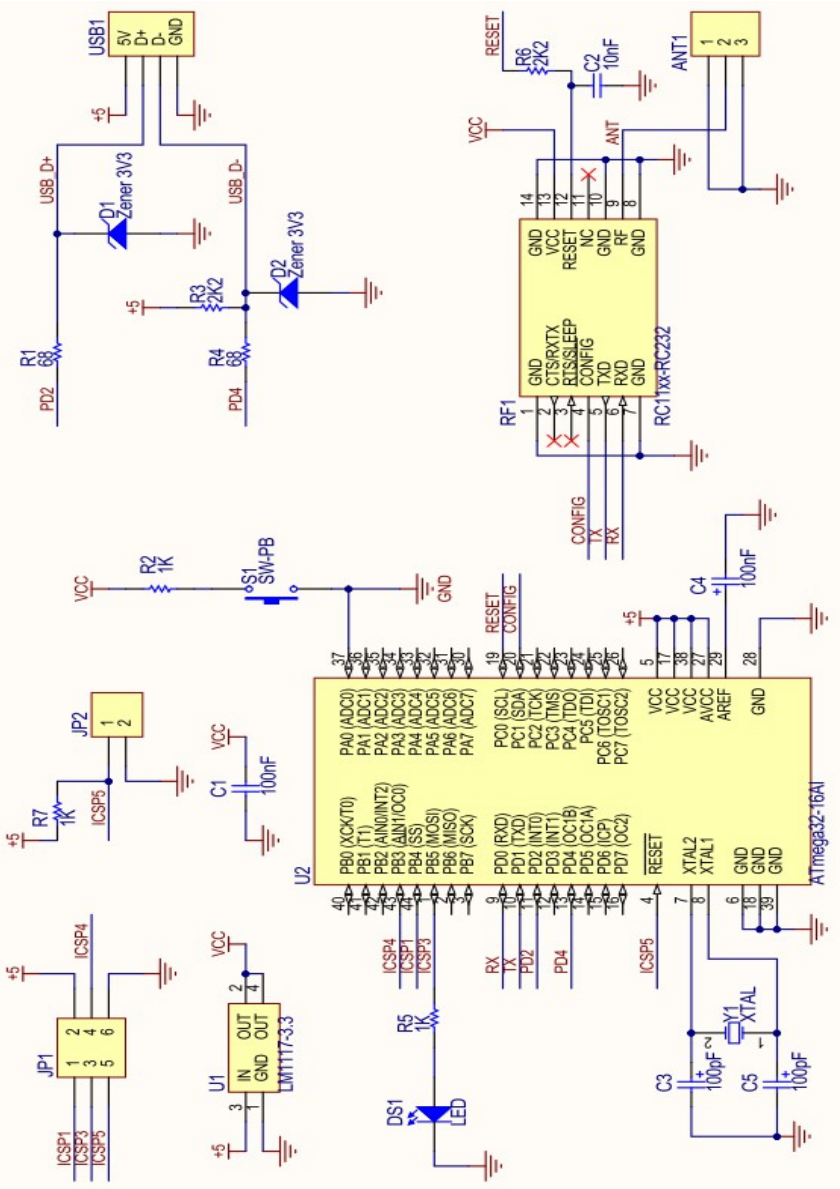

Figura 37 - Diagrama esquemático da Unidade Fixa. Fonte: autoria própria

O esquemático serve de origem à placa de circuito impresso apresentada na Figura 38.

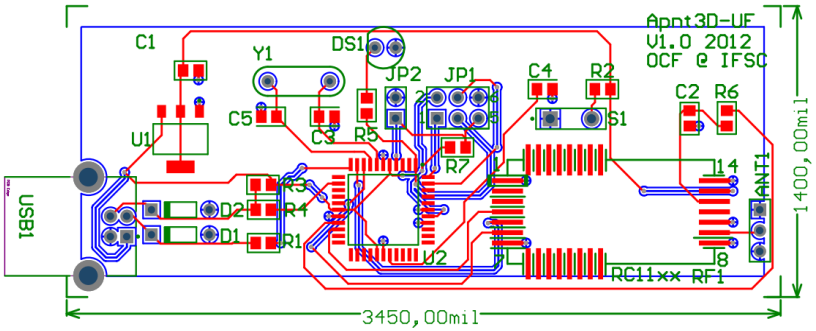

Figura 38 - Placa de circuito impresso da Unidade Fixa. Fonte: autoria própria.

As unidades de medida são em mils (milésimos de polegadas). Em milímetros a placa mede 87,63 mm por 35,56 mm. Há um plano de terra na camada inferior da placa exibido apenas pelos contornos para facilitar a visualização neste documento. No momento da fabricação este plano seria preenchido por cobre.

5.2.3 Aspectos ergonômicos

O gabinete que envolve a Unidade Fixa deve ser o menor possível, desde que haja uma abertura para o conector USB e espaço para a antena. O gabinete deve ser de um material leve para que não exerça muito peso sobre o conector USB que estará conectado ao computador.

A Figura 39 apresenta uma sugestão de gabinete para a Unidade Fixa.

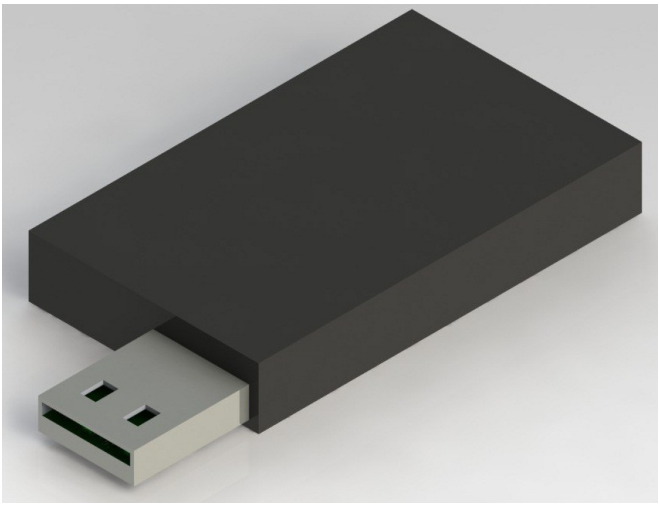

Figura 39 - Exemplo de desenho mecânico para Unidade Fixa.
Fonte: autoria própria.

A Unidade Fixa pode ter um gabinete mais simples, apenas para acomodar a placa, conector e antena. Não há muitas preocupações com ergonomia porque esta unidade não será manuseada a todo instante pelo usuário.

6 TESTES E RESULTADOS OBTIDOS

Neste capítulo serão apresentados testes e resultados obtidos com o desenvolvimento do dispositivo. Os testes feitos avaliaram o uso, funcionamento, alcance, duração da bateria e custos.

6.1 Funcionamento e uso

Primeiramente os testes capturaram os dados lidos dos conversores analógico-digital do microcontrolador. Estes dados capturados foram analisados e efetuadas as operações de integral dupla (conforme Seção 5.1.1) para ajustar o algoritmo e os limites de corte.

Para validar o funcionamento do protocolo de comunicação, o sistema foi montado sem utilizar os transceptores RF. Dessa forma, a Unidade Móvel transmite os dados via serial diretamente à Unidade Fixa. Assim validou-se:

- as etapas de cálculo da Unidade Móvel: o movimento foi interpretado, calculado e transmitido à Unidade Fixa;
- funcionamento da identificação dos botões da Unidade Móvel: um botão para sinalizar que há movimento e outros dois botões de ação de um *mouse*;
- o protocolo de comunicação: certifica-se o envio e recebimento do *buffer* que contém o incremento do cursor e os botões pressionados. A Unidade Fixa recebe o *buffer* e envia pela USB o deslocamento e botões pressionados;
- comunicação via USB: a Unidade Fixa foi reconhecida pelo sistema operacional (Windows 7) como um dispositivo "*HID-compliant mouse*" (conforme mostrado na Figura 40). Além de reconhecida pelo sistema operacional, a Unidade Fixa comporta-se adequadamente como um *mouse*, deslocando o cursor e pressionando os botões conforme determinado no *buffer*.

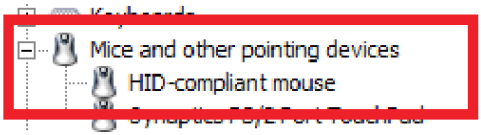

Figura 40 - Unidade Fixa reconhecida como HID pelo sistema operacional Windows 7.
Fonte: autoria própria

Com o funcionamento das unidades conectadas entre si garantido, acrescentam-se os transceptores ao sistema. À Unidade Móvel também acrescenta-se o conjunto de pilhas para alimentação. Neste ponto o sistema está montado de acordo como planejado. Como os módulos operam de forma transparente, bastou conectar as unidades aos transceptores, ou seja, não foi necessária nenhuma alteração no firmware para que a comunicação funcionasse. A Unidade Móvel continuou enviando o *buffer* e a Unidade Fixa recebeu-o, traduziu e enviou o comando de deslocamento à USB.

6.2 Alcance

O apontador tridimensional será utilizado para controlar um cursor de um computador que estará afastado do usuário. Nesse cenário o computador pode estar em uma sala ou ambiente diferente. Os testes efetuados revelaram que o alcance da transmissão via RF, em ambiente doméstico foi de aproximadamente 6 metros, levando-se em conta paredes e obstáculos presentes.

6.3 Consumo de bateria da Unidade Móvel

A Unidade Móvel foi alimentada utilizando quatro pilhas recarregáveis AA de 1,2 V cada. Isto totaliza 4,8 V para alimentar um circuito que foi dimensionado para 5 V. O circuito funcionou

perfeitamente mesmo com essa diferença de 0,2 V. A bateria alimentou o circuito (placas ArduIMU e transceptora) em modo de transmissão constante (transmitindo movimentos o tempo todo) por aproximadamente 2 horas.

6.4 Custos

Este trabalho foi implementado com as placas Arduino Duemilanove e ArduIMU, com o uso adicional de uma placa transceptora RF. O hardware utilizado foi custeado pelo próprio autor. Sugeriu-se um *hardware* diferente para redução de componentes e de tamanho de placa. A Tabela 2 apresenta os custos aproximados da implementação discutida no trabalho.

Tabela 2 - Custos do hardware utilizado (em 04/2012). Fonte: autoria própria.

Componente	**Quantidade**	**Valor Unitário (R$)**	**Valor Total (R$)**
Placa Arduino Duemilanove (fornecedor Multilógica)	1	108,00	108,00
Placa ArduIMU V2 (fornecedor Sparkfun)	1	168,00	168,00
Módulo Transceptor RC1140-232 (fornecedor RS do Brasil)	2	50,00	100,00
Regulador LM1117-3.3	2	0,73	1,46
Cabo USB	1	3,00	3,00
Conjunto de 4 Pilhas recarregáveis	1	20,00	20,00
Suporte para pilhas	1	2,00	2,00
Outros (resistores, diodos, botões, conectores)	1	10,00	10,00
Total			**412,46**

A Tabela 3 apresenta os custos aproximados do *hardware* proposto já contemplando a Unidade Fixa e a Unidade Móvel.

Tabela 3 - Custos do hardware proposto (em 04/2012). Fonte: autoria própria.

Componente	**Quantidade**	**Valor Unitário (R$)**	**Valor Total (R$)**
Microcontrolador ATMEGA328 (fornecedor Digikey)	2	8,28	16,56
Acelerômetro ADXL 335 (fornecedor Digikey)	1	5,47	5,47
Regulador LM1117-3.3 (fornecedor Digikey)	2	1,97	3,94
Módulo Transceptor RC1140-232 (fornecedor RS do Brasil)	2	50,00	100,00
Placa da Unidade Fixa (fornecedor Digicart)	1	16,80	16,80
Placa da Unidade Móvel (fornecedor Digicart)	1	16,00	16,00
Conjunto de 4 Pilhas recarregáveis	1	20,00	20,00
Suporte para pilhas	1	2,00	2,00
Outros (resistores, diodos, botões, conectores)	1	20,00	20,00
Total			**200,77**

Os valores consultados em fornecedores internacionais tomaram como base a cotação do dólar valendo R$ 1,70. A cotação das placas para o *hardware* proposto foi feita considerando uma quantidade de 10 placas. Ao aumentar a quantidade, o preço por placa também diminui: para 20 placas, o preço por placa seria de R$ 12,50 e R$ 12,05 para as placas da Unidade Fixa e da Unidade Móvel, respectivamente.

7 CONSIDERAÇÕES ADICIONAIS

Este capítulo apresenta alguns pontos avaliados no decorrer do trabalho e uma breve discussão com considerações e apresentação de resultados.

7.1 Deslocamento do cursor

O gráfico (d) da Figura 27 apresenta o deslocamento depois de feito o tratamento dos sinais lidos no sensor. Devido à sensibilidade do sensor, qualquer pequeno movimento resulta em um deslocamento no cursor. O ajuste fino dos valores de limite mínimo para aceleração e do contador de zeros para considerar que o movimento chegou ao fim foram estabelecidos com base no uso do dispositivo.

7.2 Segurança do sistema operacional sobre a Unidade Fixa

A Unidade Fixa é vista pelo sistema operacional como um dispositivo HID. Por isso, não é necessária instalação de *drivers* específicos ou de atribuir permissão de acesso para o dispositivo. Isso pode ser considerado uma falha de segurança em que o dispositivo poderia aproveitar-se disso e ao detectar um período de inatividade do computador, enviar cliques e pressionamentos de tecla para pontos específicos da tela.

Não há como o sistema operacional interpretar que cliques e pressionamentos de teclas podem ser algo intrusivo ou indesejado pelo usuário. O dispositivo desenvolvido não tem esse foco, mas levanta-se a questão de até que ponto a segurança de um sistema depende do usuário ou dos equipamentos conectados ao computador.

8 CONCLUSÕES

O objetivo de desenvolver *firmware* e *hardware* de um apontador tridimensional sem fio foi concluído com sucesso. O *hardware* pôde ser montado e o dispositivo funcionou.

As etapas de obtenção do sinal, filtragem e processamento foram necessárias e embasadas na mecânica newtoniana de deslocamento, velocidade e aceleração. O *firmware* foi desenvolvido sobre a plataforma Arduino, possibilitando um menor *time-to-prototype* do produto.

O *hardware* também foi desenvolvido sobre a mesma plataforma com a placa Arduino Duemilanove para a comunicação USB e a placa ArduIMU para captação do movimento pelo acelerômetro ADXL 335. Outro motivo para selecionar este *hardware* é o fato de ser baseado em um microcontrolador ATmega, que encontra-se disponível no mercado nacional e possui um encapsulamento adequado para aplicações de pequenas dimensões. A transmissão foi feita utilizando os módulos RF modelo RC1140-232 da Radiocrafts.

Como assunto para trabalhos futuros, sugere-se a fabricação de *hardware* (placas de circuito impresso e gabinetes) específicos para a aplicação. Com um *hardware* específico, pode-se melhorar o circuito, otimizar o consumo de bateria e reduzir a quantidade e tamanho das pilhas recarregáveis de forma a fornecer uma tensão mais adequada ao circuito. Além disso, outras funcionalidades podem ser agregadas ao dispositivos, como *scroll*, configuração de gestos e um modo de calibração.

Para trabalhos futuros, também pode-se refinar a seleção dos microcontroladores utilizados. Pode-se optar por modelos mais adequados à função exercida, visto que o código fonte consumiu pouca memória de programa.

Revisando o *hardware* e o *firmware* desenvolvidos, pode-se:

- reduzir a quantidade e o tamanho (de AA para AAA) de pilhas melhorando o circuito de alimentação;
- explorar modos de operação *sleep* ou *idle* que levam o microcontrolador e módulo transceptor a consumir

menos.

Com relação às funcionalidades, pode-se acrescentar mais recursos ao apontador, tais como:

- *scroll*: presente em quase todos os *mouses* encontrados no mercado;
- desenhar um gabinete para a Unidade Fixa ser armazenada junto ou dentro da Unidade Móvel, já que esta possui um volume maior;
- permitir calibrar o dispositivo, adaptando a velocidade dos movimentos ao deslocamento do cursor na tela;
- permitir a configuração de gestos: o usuário cria um gesto e o associa a uma ação. Por exemplo: movimentar a Unidade para baixo e para a esquerda significa retroceder uma página no aplicativo em uso;
- sinalizar para o usuário quando as pilhas estiverem fracas.

Outras implementações podem focar na garantia que este dispositivo (com os recursos citados) seja multiplataforma, rodando em Linux, Windows ou MAC.

Durante a elaboração deste trabalho, foi encontrado o circuito integrado CC2511 da Texas Instruments que possui um microcontrolador com arquitetura baseada em 8051, realiza a comunicação via RF e via USB. Ele atende aos requisitos da Unidade Fixa, possibilitando um futuro desenvolvimento que utilizasse o CC2511 substituindo o microcontrolador e o transceptor. Nessa mesma família de CIs há o CC2510, que não possui comunicação via USB, mas poderia substituir o microcontrolador da Unidade Móvel. Ou seja, em ambas unidades, poder-se-ia substituir o conjunto “microcontrolador ATMega e módulo transceptor RC1140” pelos CIs CC251x.

REFERÊNCIAS BIBLIOGRÁFICAS

ANALOG DEVICES, Inc. **Datasheet ADXL335: Small low power, 3-axis +- 3g Accelerometer. Rev. B.** PDF. Estados Unidos, 2010.

ATMEL CORPORATION. **Datasheet AT90USB82, AT90USB162: 8-bit AVR Microcontroller with 8/16K Bytes of ISP Flash and USB controller**. PDF. Estados Unidos, 2010.

ARDUIMU. **Introduction to ArduIMU**. Disponível em: http://code.google.com/p/ardu-imu/wiki/IntroductionPage. Acessado em 04/2011.

ARDUINO. **Arduino**: Introduction. Disponível em: http://arduino.cc/en/Guide/Introduction. Acessado em 04/2011.

AXELSON, Jan. **USB Complete**: Everything You Need to Develop Custom USB Peripherals. 2. ed. Madison, Estados Unidos: Lakeview Research. 2001. 519 p.

BISHOP, Robert H. **Mechatronic systems, sensors and actuators**: fundamentals and modeling. Boca Raton, Florida, Estados Unidos: CRC Press - Taylor & Francis Group. 2008. 656 p.

BLUEGIGA TECHNOLOGIES OY. **WT12 Bluetooth Module**. PDF. Espoo, Finlândia. 2009.

BLUETOOTH SIG. **Bluetooth basics**: A Look at the Basics of Bluetooth Wireless Technology. Disponível em: http://www.bluetooth.com/Pages/Basics.aspx. Acessado em 04/2011.

CÔRREA, Ana Grasielle Dionísio; ASSIS, Gilda Aparecida de; NASCIMENTO, Marilena do; LOPES, Roseli de Deus. **GenVirtual: Um jogo musical para reabilitação de indivíduos com necessidades especiais**. Revista brasileira de informática na educação, Porto Alegre, Volume 16. Ano 1. p.10-17. 2008.

FITTS, Paul M. **The information capacity of the human motor system in controlling the amplitude of movement**. Journal of Experimental Psychology, Volume 47, Número 6. p.381-3911954.

FREESCALE SEMICONDUCTOR. **Application note AN3397: Implementing positioning algorithms using accelerometers**. PDF. Estados Unidos, 2007.

FRADEN, Jacob. **Handbook of Modern Sensors**: Physics, Designs and Applications. 4a ed. Springer-Verlag Berlin Heidelberg. New York, Estados Unidos, 2010.

FTDI – Future Technology Devices International, Ltd. **Datasheet FT232R: UART USB IC**. PDF. Reino Unido, 2011.

GYRATION. **Air Mouse Ellite**. Disponível em: www.gyration.com/index.php/us/products/in-air-micekeyboards/air-mouse-elite.html. Acessado em 03/2012.

HOOBER, Steven; BERKMAN, Eric. **Designing mobile interfaces**. O'Reilly Media Inc. Canada, 2011.

MACKENZIE, I. Scott. **Human-Computer Interaction**: Fitt's Law as a research and design tool in human-computer interaction. Lawrence Erlbaum Associates, Inc. Toronto, Canadá. 1992.

MICROCHIP TECHNOLOGY Inc. **Datasheet PIC18F2455/2550/4455/4550: 28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology**. PDF. Estados Unidos, 2009.

NATIONAL SEMICONDUCTOR. **Datasheet LMX9838**: **Bluetooth Serial Module**. PDF. Estados Unidos, 2007.

OBDEV, Objective Development Software GmbH. **V-USB**: Virtual USB port for AVR microcontrollers. Disponível em: http://www.obdev.at/products/vusb/index.html. Acessado em 04/2011.

PRACTICAL ARDUINO. **Virtual USB Keyboard**. Disponível em: http://www.practicalarduino.com/projects/virtual-usb-keyboard. Acessado em: 04/2011.

RADIOCRAFTS AS. **Datasheet RC11xx-RC232: Low cost multi-channel RF transceiver module. Rev 1.43**. PDF. Noruega, 2010.

RADIOMETRIX LTD. **Datasheet RDL2**: UHF Multi Channel Wide Band FM Transceiver. PDF. Reino Unido. 2007.

ROVING NETWORKS. **Datasheet RN-41**: Class 1 Bluetooth Module. PDF. Estados Unidos. 2011.

STALLINGS, William. **Wireless Communications and networks**. New Jersey, Estados Unidos: Pearson Education, Inc., 2005. 569 p.

STEFANESCU, Dan Mihai. **Handbook of Force Tranducers**: Principles and components. Alemanha: Springer-Verlag Berlin Heidelberg, 2011. 643 p.

UNITED STATES PATENT OFFICE. **X-Y position indicator for a display system**. 3541541. 1970. 7 p.

WERNECK, Marcelo Martins. **Transdutores e interfaces**. Rio de Janeiro, RJ: LTC – Livros Técnicos e Científicos Editora S.A. 1996.

ANEXOS

Código-fonte do *firmware* da Unidade Fixa (para Arduino)

```
// Includes
#include "UsbMultiHID.h"

// Defines
#define TAM_CARGA_UTIL     6
#define TAM_BUFFER         7
#define TAM_MAX_BUFFER     64

#define IND_INICIO         0
#define IND_POS_X          1
#define IND_POS_Y          2
#define IND_BT_ESQUERDO    3
#define IND_BT_DIREITO     4
#define IND_FIM            5
#define IND_CHECKSUM       6

#define INICIO_BUFFER      'A'
#define FIM_BUFFER         'B'
#define BT_ATIVADO         'N'
#define BT_DESATIVADO      'F'

#define PINO_LED           13

// Variaveis
int byteLido = 0;
unsigned int indice = 0, indiceInicio, indiceFim;
char buffer[TAM_MAX_BUFFER];
unsigned char ucPodeLer;
char botaoAnterior[2];


// Funcoes
void delayMs(unsigned int ms) {
    for (int i = 0; i < ms; i++) {
        delayMicroseconds(1000);
    }
}

void setup() {
    unsigned int i;

    pinMode(PINO_LED, OUTPUT);
```

```
    Serial.begin(38400);

    for (i = 0; i < TAM_MAX_BUFFER; i++) {
        buffer[i] = 0;
    }
    indice = 0;
    ucPodeLer = 0;
    indiceInicio = 0;
    indiceFim = 0;

    botaoAnterior[0] = false;
    botaoAnterior[1] = false;

    // disable timer 0 overflow interrupt (used for millis)
    TIMSK0 &= !(1 << TOIE0);
}


void loop() {
    boolean bt_esquerdo, bt_direito;
    int nQuantidade, nContador;
    unsigned char i, checksum;

    usbMultiHID.update();

    if (Serial.available()) {
        nQuantidade = Serial.available();

        while (nQuantidade > 0) {
            byteLido = Serial.read();
            buffer[indiceFim] = byteLido;
            indiceFim = (indiceFim + 1) % TAM_MAX_BUFFER;
            nQuantidade--;

            ucPodeLer = 1;
        }
    }

    if (ucPodeLer) {
        ucPodeLer = 0;

        if (indiceFim > indiceInicio) {
            nContador = indiceFim - indiceInicio;
        }
        else {
            nContador = (TAM_MAX_BUFFER - indiceInicio) +
indiceFim;
        }
```

```
        while ((nContador > 0) && (buffer[indiceInicio] !=
INICIO_BUFFER)) {
            indiceInicio = (indiceInicio + 1) % TAM_MAX_BUFFER;
            nContador--;
        }


        if ((buffer[indiceInicio] == INICIO_BUFFER) &&
(buffer[indiceInicio + IND_FIM] == FIM_BUFFER)) {

            checksum = 0;
            for (i = indiceInicio; i < (indiceInicio +
TAM_CARGA_UTIL); i++) {
                checksum ^= buffer[i];
            }

            if (buffer[indiceInicio + IND_BT_ESQUERDO] ==
BT_ATIVADO) {
                bt_esquerdo = true;
            }
            else if (buffer[indiceInicio + IND_BT_ESQUERDO] ==
BT_DESATIVADO) {
                bt_esquerdo = false;
            }


            if (buffer[indiceInicio + IND_BT_DIREITO] ==
BT_ATIVADO) {
                bt_direito = true;
            }
            else if (buffer[indiceInicio + IND_BT_DIREITO] ==
BT_DESATIVADO) {
                bt_direito = false;
            }

            if ( (buffer[indiceInicio + IND_POS_X] != 0) ||
                 (buffer[indiceInicio + IND_POS_Y] != 0) ||
                 (buffer[indiceInicio + IND_BT_ESQUERDO] !=
botaoAnterior[0]) ||
                 (buffer[indiceInicio + IND_BT_DIREITO] !=
botaoAnterior[1]) )
            {

                digitalWrite(PINO_LED, HIGH);
                movimentaCursor(buffer[indiceInicio +
IND_POS_X], buffer[indiceInicio + IND_POS_Y], bt_esquerdo,
bt_direito);
                digitalWrite(PINO_LED, LOW);
            }

            botaoAnterior[0] = buffer[indiceInicio +
```

```
IND_BT_ESQUERDO];
            botaoAnterior[1] = buffer[indiceInicio +
IND_BT_DIREITO];

            buffer[indiceInicio] = 'X';
            indiceInicio = (indiceInicio + TAM_BUFFER) %
TAM_MAX_BUFFER;

        }
    }

}

void movimentaCursor(short posicao_x, short posicao_y, boolean
botao_esquerdo, boolean botao_direito)
{
    byte mouse_buttons;
    short mouse_wheel;

    mouse_wheel = 0;

    mouse_buttons = 0;
    if (botao_esquerdo) mouse_buttons |= 0x01;
    if (botao_direito)  mouse_buttons |= 0x02;

    usbMultiHID.send_mouse_report(mouse_buttons, posicao_x,
posicao_y, mouse_wheel);

}
```

Código-fonte do *firmware* da Unidade Móvel (para ArduIMU)

```
// Defines
#define PINO_LED_VERMELHO 5
#define PINO_LED_AZUL     6
#define PINO_LED_AMARELO  7
#define LIGA_LED_VERMELHO()
{ digitalWrite(PINO_LED_VERMELHO,HIGH); };
#define LIGA_LED_AZUL()     { digitalWrite(PINO_LED_AZUL,HIGH);
};
#define LIGA_LED_AMARELO()
{ digitalWrite(PINO_LED_AMARELO,HIGH);  };
#define DESLIGA_LED_VERMELHO()
{ digitalWrite(PINO_LED_VERMELHO,LOW); };
#define DESLIGA_LED_AZUL()
{ digitalWrite(PINO_LED_AZUL,LOW);     };
#define DESLIGA_LED_AMARELO()
{ digitalWrite(PINO_LED_AMARELO,LOW);  };

#define PINO_BOTAO_1      11
#define PINO_BOTAO_2      12
#define PINO_BOTAO_3      10

// Variaveis
long timer_atual;
long timer;

boolean iniciado;

// Funcoes
void setup()
{
    iniciado = false;
    timer_atual = 0;
    timer = 0;

    pinMode(PINO_LED_VERMELHO,OUTPUT);
    pinMode(PINO_LED_AZUL,OUTPUT);
    pinMode(PINO_LED_AMARELO,OUTPUT);

    pinMode(PINO_BOTAO_1, INPUT);
    pinMode(PINO_BOTAO_2, INPUT);
    pinMode(PINO_BOTAO_3, INPUT);

    digitalWrite(PINO_BOTAO_1, HIGH);
    digitalWrite(PINO_BOTAO_2, HIGH);
```

```
    digitalWrite(PINO_BOTAO_3, HIGH);

    pinMode(8, INPUT);

    LIGA_LED_AMARELO();

    Analog_Init();
    Calculo_Init();

    timer = millis();
    delay(100);

    DESLIGA_LED_AMARELO();

    iniciado = true;
}


void loop()
{
    if (iniciado) {

        timer_atual = millis();

        if ((timer_atual - timer) >= 1) // 1
        {
            timer = timer_atual;

             CalculaPosicao();

            if (digitalRead(PINO_BOTAO_3) == LOW) {
                LIGA_LED_AMARELO();

                EnviaBuffer();
            }
            else {
                DESLIGA_LED_AMARELO();
            }
        }
    }

}
```

```
// Defines
#define QTDE_SENSORES      2

#define TAM_CARGA_UTIL     6
#define TAM_BUFFER         7

#define IND_INICIO         0
#define IND_POS_X          1
#define IND_POS_Y          2
#define IND_BT_ESQUERDO    3
#define IND_BT_DIREITO     4
#define IND_FIM            5
#define IND_CHECKSUM       6

#define INICIO_BUFFER      'A'
#define FIM_BUFFER         'B'
#define BT_ATIVADO         'N'
#define BT_DESATIVADO      'F'

#define QTDE_LEITURAS_MEDIA_MOVEL    64
#define QTDE_LEITURAS_ANALOGICAS     16
#define LIMITE_MINIMO_ACELERACAO     10
#define CONTAGEM_ACELERACAO_ZERO     10
#define DIVISOR_ESCALA    40 //15

#define ESTADO_ANTERIOR   0
#define ESTADO_ATUAL      1
#define QTDE_ESTADOS      2

// Variaveis
unsigned char sensores[QTDE_SENSORES] = { 0, 1 }; // {0 , 1 ,
2}; Accel // {3, 6, 7}; Gyro

unsigned int indice_leitura;
signed int leituras[QTDE_SENSORES][QTDE_LEITURAS_MEDIA_MOVEL];
signed int contador_aceleracao_zero[QTDE_SENSORES];
signed int aceleracao[QTDE_SENSORES][2];
signed int velocidade[QTDE_SENSORES][2];
signed long posicao[QTDE_SENSORES][2];
signed char pos[QTDE_SENSORES];

// Funcoes
void Analog_Init (void) {
    int i, j;

    analogReference(EXTERNAL);

    for (i = 0; i < QTDE_SENSORES; i++) {
        pinMode(sensores[i], INPUT);
    }
```

```
}

void Calculo_Init() {
    int i, j;
    long acumulador;

    Serial.begin(38400);
    indice_leitura = 0;

    for (i = 0; i < QTDE_SENSORES; i++) {
        contador_aceleracao_zero[i] = 0;

        for (j = 0; j < QTDE_ESTADOS; j++) {
            aceleracao[i][j] = 0;
            velocidade[i][j] = 0;
            posicao[i][j] = 0;
        }

        acumulador = 0;
        for (j = 0; j < QTDE_LEITURAS_MEDIA_MOVEL; j++) {
            leituras[i][j] = analogRead(sensores[i]);
            acumulador += leituras[i][j];
        }
    }
}

void CalculaPosicao(void)
{
    signed int leitura_auxiliar, media_atual, auxiliar;
    signed long acumulador;
    unsigned char i, indice_sensor;
    signed int diferenca;

    for (indice_sensor = 0; indice_sensor < QTDE_SENSORES;
indice_sensor++) {
        acumulador = 0;
        for (i = 0; i < QTDE_LEITURAS_ANALOGICAS; i++) {
            acumulador += analogRead(sensores[indice_sensor]);
        }
        leitura_auxiliar = acumulador /
QTDE_LEITURAS_ANALOGICAS;

        leituras[indice_sensor][indice_leitura] =
leitura_auxiliar;

        acumulador = 0;
        for (i = 0; i < QTDE_LEITURAS_MEDIA_MOVEL; i++) {
            acumulador += leituras[indice_sensor][i];
        }
        media_atual = acumulador / QTDE_LEITURAS_MEDIA_MOVEL;
```

```
        aceleracao[indice_sensor][ESTADO_ATUAL] =
leitura_auxiliar - media_atual;

        if (abs(aceleracao[indice_sensor][ESTADO_ATUAL]) <=
LIMITE_MINIMO_ACELERACAO) {
            aceleracao[indice_sensor][ESTADO_ATUAL] = 0;
        }

        if (aceleracao[indice_sensor][ESTADO_ATUAL] == 0) {
            contador_aceleracao_zero[indice_sensor]++;
        }
        else {
            contador_aceleracao_zero[indice_sensor] = 0;
        }

        if (contador_aceleracao_zero[indice_sensor] >=
CONTAGEM_ACELERACAO_ZERO) {
            velocidade[indice_sensor][ESTADO_ATUAL] = 0;
        }
        else {
            auxiliar = (aceleracao[indice_sensor][ESTADO_ATUAL]
- aceleracao[indice_sensor][ESTADO_ANTERIOR]) / 2;
            velocidade[indice_sensor][ESTADO_ATUAL] =
velocidade[indice_sensor][ESTADO_ANTERIOR] +
aceleracao[indice_sensor][ESTADO_ANTERIOR] + auxiliar;
        }

        auxiliar = (velocidade[indice_sensor][ESTADO_ATUAL] -
velocidade[indice_sensor][ESTADO_ANTERIOR]) / 2;
        posicao[indice_sensor][ESTADO_ATUAL] =
posicao[indice_sensor][ESTADO_ANTERIOR] +
velocidade[indice_sensor][ESTADO_ANTERIOR] + auxiliar;

        diferenca = posicao[indice_sensor][ESTADO_ATUAL] -
posicao[indice_sensor][ESTADO_ANTERIOR];
        if (abs(diferenca / DIVISOR_ESCALA) > 127) {
            if (diferenca > 0) {
                pos[indice_sensor] = 127;
            }
            else {
                pos[indice_sensor] = -127;
            }
        }
        else {
            pos[indice_sensor] = diferenca / DIVISOR_ESCALA;
        }

        aceleracao[indice_sensor][ESTADO_ANTERIOR] =
aceleracao[indice_sensor][ESTADO_ATUAL];
        velocidade[indice_sensor][ESTADO_ANTERIOR] =
```

```
velocidade[indice_sensor][ESTADO_ATUAL];
        posicao[indice_sensor][ESTADO_ANTERIOR] =
posicao[indice_sensor][ESTADO_ATUAL];
    }

    indice_leitura = (indice_leitura + 1) %
QTDE_LEITURAS_MEDIA_MOVEL;
}

void EnviaBuffer() {
    char buffer[TAM_BUFFER];
    unsigned char i, checksum;

    buffer[IND_INICIO] = INICIO_BUFFER;
    buffer[IND_POS_X] = pos[1];
    buffer[IND_POS_Y] = pos[0];

    if (digitalRead(PINO_BOTAO_1) == LOW) {
        buffer[IND_BT_DIREITO] = BT_ATIVADO;
        LIGA_LED_AZUL();
    }
    else {
        buffer[IND_BT_DIREITO] = BT_DESATIVADO;
        DESLIGA_LED_AZUL();
    }

    if (digitalRead(PINO_BOTAO_2) == LOW) {
        buffer[IND_BT_ESQUERDO] = BT_ATIVADO;
        LIGA_LED_VERMELHO();
    }
    else {
        buffer[IND_BT_ESQUERDO] = BT_DESATIVADO;
        DESLIGA_LED_VERMELHO();
    }


    buffer[IND_FIM] = FIM_BUFFER;

    checksum = 0;
    for (i = 0; i < TAM_CARGA_UTIL; i++) {
        checksum ^= buffer[i];
    }

    buffer[IND_CHECKSUM] = checksum;

    for (i = 0; i < TAM_BUFFER; i++) {
        Serial.print(buffer[i]);
    }
}
```