Universidade de São Paulo

Escola de Engenharia de São Carlos

Departamento de Engenharia Elétrica

# Dispositivo microcontrolado para auxílio na orientação de deficientes visuais

**Autor:** Murilo Augusto Gallani

**Orientador:** Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos

2015

**Murilo Augusto Gallani**

# Dispositivo microcontrolado para auxílio na orientação de deficientes visuais

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Sistemas de Energia e Automação

ORIENTADOR: Prof. Dr. Evandro Luis Linhari Rodrigues

Gallani, Murilo Augusto
G163d Dispositivo microcontrolado para auxílio na orientação de deficientes visuais / Murilo Augusto Gallani; orientador Evandro Luís Linhari Rodrigues. São Carlos, 2015.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. mbed. 2. localização sonora. 3. HRIR. 4. unidade de medição inercial. 5. I2C. 6. sensor de distância. I. Título.

# FOLHA DE APROVAÇÃO

Nome: Murilo Augusto Gallani

Título: "Dispositivo microcontrolado para auxílio na orientação de deficientes visuais"

Trabalho de Conclusão de Curso defendido e aprovado em 23/06/2015,

com NOTA 10,0 ( DEZ , ZERO ), pela Comissão Julgadora:

*Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)*

*Prof. Dr. Marcelo Andrade da Costa Vieira - (SEL/EESC/USP)*

*Mestre Thales Eugenio Portes de Almeida - (Doutorando - SEL/EESC/USP)*

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

# Resumo

Esse trabalho apresenta o dispositivo desenvolvido para aprimorar o desempenho de bengalas na orientação e detecção de obstáculos por deficientes visuais durante sua locomoção. O aparelho, anexo ao extremo do instrumento, se utiliza de seu movimento lateral, já naturalmente realizado pelo seu usuário, para efetuar uma varredura da área à sua frente e informá-lo de eventuais obstáculos e impedimentos ali presentes. Essa função, que já é realizada pela bengala, é melhorada através da utilização de sensores de distância para aumentar significantemente seu alcance, enquanto também aumentando sua efetiva área de atuação. Além da distância, com o auxílio de uma unidade de medição inercial obstáculos encontrados têm sua posição relativa ao usuário calculada e a ele transmitida via áudio. Fones de ouvido *stereo* são utilizados para simular sons cuja fonte sonora se localiza no ponto onde tal objeto foi detectado, sendo possível, portanto, a identificação de sua posição pelo ouvinte. Utilizando uma placa de desenvolvimento mbed para o controle e processamento das informações o dispositivo permite, através das funções descritas, que seus usuários se orientem e se locomovam com maior confiança e segurança.



Palavras chave: mbed, localização sonora, HRIR, unidade de medição inercial, I2C, sensor de distância.

# Abstract

This essay presents the device developed to improve the performance of white canes in the orientation and obstacle detection by visually impaired people while they walk. The equipment, attached to the extremity of the cane, uses its lateral movement, already naturally made by its user, to perform a scan of the area ahead and inform him of any obstacles found. This function, already performed by the cane, is improved by the use of distance sensors to notably enhance its range, while also enlarging the area scanned. Besides the distance, with the help of an inertial measurement unit obstacles found have their relative position calculated and sent to him via audio feedback. Stereo earphones are used to simulate sounds that have their source at the location where the object was found, being possible, then, the identification of its position by the listener. Using an mbed development board for the control and data processing the device allows, with the use the functions before mentioned, its users to orientate themselves and move with greater confidence and safety.



Keywords: mbed, sound localization, HRIR, inertial measurement unit, I2C, distance sensor

## Lista de Siglas

HRIR – Head Related Impulse Response

HRTF – Head Related Transfer Function

ITD – Interaural Time Difference

ITL – Interaural Level Difference

PWM – Pulse Width Modulation

I2C – Inter Integrated Circuit

IMU – Inertial Measurement Unit

## Lista de Figuras

## Lista de Tabelas

# Sumário

# 1) Introdução

A deficiência visual, cuja pior condição é a cegueira, falta de visão, afeta milhões de pessoas no mundo (WHO). Devido ao fato de a visão ser um sentido que nos dá uma quantidade enorme de informações sobre o ambiente à nossa volta, o ser humano é naturalmente dependente desse sentido para a realização de muitas de suas atividades diárias. Portanto, indivíduos que sofrem de cegueira total ou parcial enfrentam diversas dificuldades em seu dia a dia.

A locomoção, tanto em locais fechados como em abertos, é um problema para os cegos uma vez que a informação visual do ambiente ao redor é essencial para a identificação de obstáculos e rotas. A ferramenta mais utilizada para o auxílio na orientação e locomoção dos deficientes visuais é a bengala, que serve diversas funções, sendo a principal delas encontrar obstáculos que possam obstruir o caminho do usuário. Outras técnicas também existem, por exemplo a audição treinada para identificar o tamanho e o tipo de ambiente em que se encontra, porém normalmente são utilizadas em conjunto com a bengala para um melhor resultado. (VISIONAWARE)

Todavia encontrar obstáculos não é a única informação que um deficiente visual pode obter através do uso desse aparelho. O cego pode tocar a bengala no chão ao caminhar para perceber a textura da onde está caminhando, podendo identificar as condições do solo, início e fim de caminhos pavimentados, encontrar e evitar imperfeições que possam causar tropeços e acidentes, encontrar degraus, demarcações nas calçadas especificamente criadas para ajudar deficientes visuais, e inclusive utilizar o barulho da bengala ao tocar o solo e objetos como informação sobre o tipo de ambiente em que se encontra (NFB).

Isso mostra que a bengala é um objeto muito versátil, sendo muito difícil criar algo que a substitua. Porém é possível criar um dispositivo que melhore a realização de suas funções, ajudando ainda mais na orientação e locomoção dos deficientes visuais.

Uma das características mais críticas quando se usa uma bengala é seu comprimento. A maioria dos cegos, quando começa a utilizar a bengala, começa com um modelo mais curto por ser mais fácil de utilizar e carregar, mas acaba trocando por um mais longo após dominar seu uso. Um maior comprimento implica em um mais longo alcance na sua busca por informações do ambiente, e o usuário se sentirá mais confortável e confiante ao caminhar, identificando obstáculos com bastante antecedência e tendo um tempo maior para tomar decisões. Infelizmente as bengalas longas possuem algumas desvantagens em relação às curtas. São mais difíceis de carregar e armazenar quando rígidas, e as dobráveis ou telescópicas são mais

frágeis, estando susceptíveis a quebras (NFB). Aumentar o alcance das bengalas, portanto, se mostra uma oportunidade interessante para ajudar e melhorar a orientação e locomoção de deficientes visuais.

A utilização da bengala ocorre posicionando a mão que a segura em frente ao corpo, à altura da cintura. Ela é movimentada lateralmente ao se locomover, posicionando sua extremidade sempre do mesmo lado do pé que está atrás, visando assim proteger e se certificar de que o caminho para o próximo passo não está obstruído (NFB). É recomendado que esse movimento abranja a área à frente estendendo-se lateralmente até 5cm além dos ombros. Deste modo, a cada passo que se dá é então varrida essa área, e suas características identificadas pela bengala (VISIONAWARE).

## 1.1) Objetivos

Esse projeto tem como objetivo a criação de um aparelho que aumente o alcance e a área de atuação das bengalas. Aproveitando o movimento lateral da bengala, realiza leituras do ambiente e colhe informações de distância e posição relativa de obstáculos e objetos em relação ao usuário. Feito isso essas informações são transmitidas ao usuário com o auxílio de fones de ouvido, que através de um sistema *stereo* indicam a direção de cada obstáculo e através de variações no volume do som indicam a distância em que se encontra. O dispositivo pode ser visto na Figura 1.1.

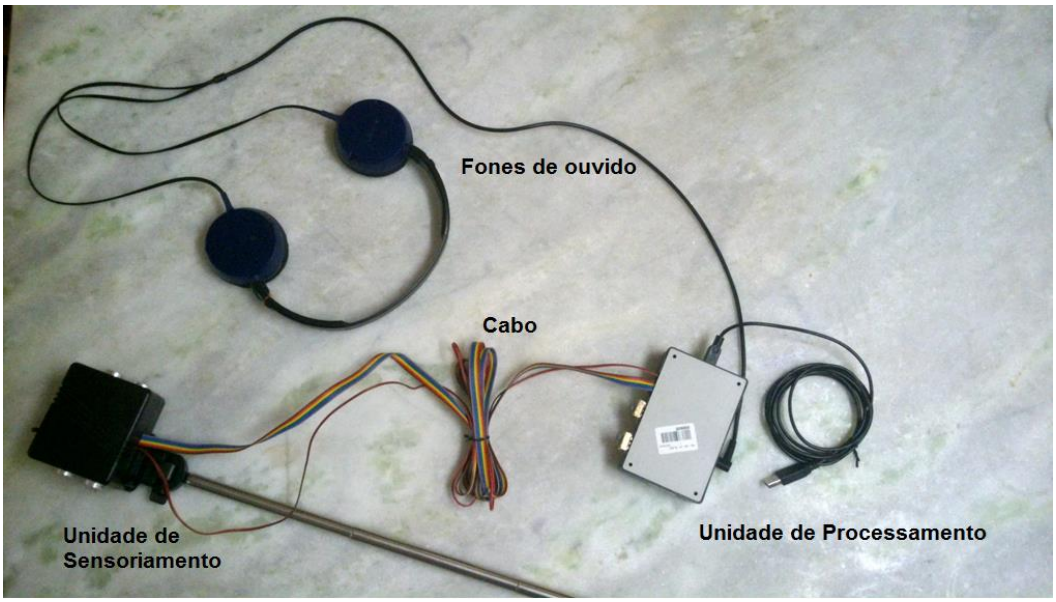



Figura 1.1 – Dispositivo e seus componentes

Foi construído utilizando diferentes sensores posicionados em sua extremidade, na chamada unidade de sensoriamento. Sensores de distância ultrassônicos realizam diversas medidas enquanto que informações de uma unidade de medição inercial são utilizados para identificar a direção para onde a bengala está sendo direcionada. O controle e as leituras dos sensores são realizados por um microcontrolador que se encontra em uma segunda unidade, de processamento, que não está na ponta da bengala como a primeira. As medidas dos dois tipos de sensores são transmitidas através de um cabo e utilizadas em conjunto para calcular as posições relativas dos diversos obstáculos ao redor do usuário.

Sistemas de som *stereo* possuem dois canais independentes e permitem, com sua utilização correta, a emulação de sons proveniente de diferentes localizações em volta do ouvinte. Reproduz-se em cada canal, direito e esquerdo, sinais com pequenas diferenças em suas amplitudes e defasagens no tempo, bem como algumas outras que são causadas pela influência da anatomia humana nas ondas sonoras que atingem os ouvidos. O cérebro humano interpreta esses sinais como sons provenientes de diferentes direções dependendo das diferenças implementadas nos canais, apesar de estarem sendo reproduzidos por fones de ouvido, que são fontes sonoras estáticas.

Essa técnica é utilizada para indicar ao usuário do dispositivo a direção onde cada obstáculo se encontra, e sua distância é transmitida alterando o volume do som de maneira inversamente proporcional à mesma. Ou seja, obstáculos que estão longe resultam em um som baixo, enquanto que objetos mais próximos produzem sons mais altos. O microcontrolador é utilizado para gerar esse som utilizando sinais PWM enviando-os aos dois canais do fone de ouvido, que se encontra conectado à unidade de processamento.

Os sons são reproduzidos iniciando pelas posições mais à esquerda, passando pela frente do ouvinte e finalizando com as posições mais à direita, simulando um movimento de varredura do ambiente, porém em uma área mais ampla e com um alcance maior que a varrida pela bengala física. O usuário pode então interpretar as informações e utilizá-las para a criação de um “mapa” mental e se orientar com maior facilidade.

Atualmente existem no mercado algumas opções similares, como por exemplo a *UltraCane* (http://www.ultracane.com/index.php?route=common/home) e a *SmartCane* (http://smartcane.saksham.org/). Ambas utilizam também sensores ultrassônicos de distância para melhorar o alcance do instrumento, porém detectam apenas objetos imediatamente à frente e na direção que a bengala está apontada, falhando então em atingir uma área mais abrangente. O método que utilizam para transmitir informação ao usuário também é mais limitado, utilizando vibrações. Esse projeto explora uma abordagem diferente para essa função, utilizando o som,

que permite informar não só distância, mas também direção, como já foi mencionado anteriormente.

# 2) Embasamento teórico

Nessa seção serão introduzidos e explicados conceitos utilizados na realização do projeto e cujo entendimento é de suma importância para a compreensão deste trabalho.

## 2.1) Localização Sonora

Localização sonora é a capacidade de um indivíduo, ao ouvir um som, identificar a localização da fonte sonora em relação à sua própria. Ou seja, se o que está gerando esse som está localizado à sua frente, atrás, ao lado. Um ouvinte é capaz de identificar a localização da fonte em qualquer ponto do espaço tridimensional ao seu redor, no entanto algumas posições se mostram mais fáceis de identificar que outras. É mais fácil, por exemplo, indicar a origem de estímulos sonoros provenientes de fontes localizadas em posições frontais indivíduo do que atrás.

Uma posição no espaço em relação à cabeça de um indivíduo pode ser definida por duas coordenadas: azimute e elevação. Consideremos o centro da cabeça como a origem de um sistema de coordenadas e uma reta que conecta essa origem a um dado ponto no espaço. A elevação desse ponto é o ângulo formado entre essa reta e o plano horizontal que contém os ouvidos, e o azimute é o ângulo entre a reta e o plano vertical que contem a origem e o nariz, como representado pela Figura 2.1. (UCDAVIS)

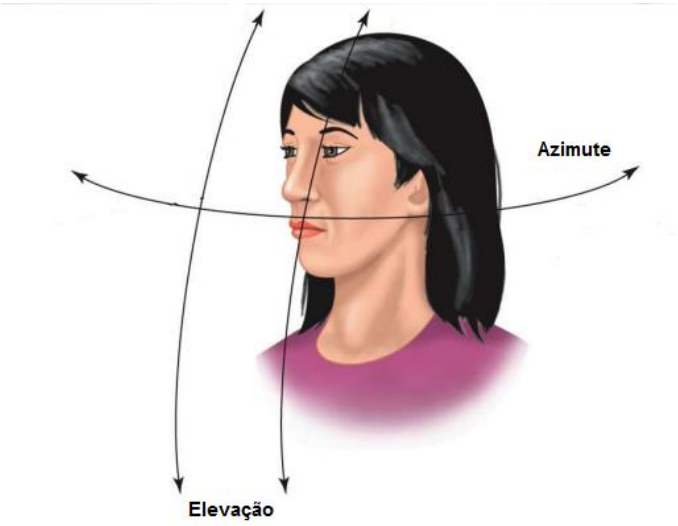



Figura 2.1 – Azimute e elevação identificados por um ouvinte (UW)

O cérebro humano utiliza diferentes informações para encontrar a origem de um som. O azimute pode ser identificado principalmente através de duas medidas:

- *Interaural Time Difference* (ITD) - É a diferença no tempo de chegada do som aos ouvidos. Um estímulo proveniente de uma fonte à frente do ouvinte chega a ambos ouvidos ao mesmo tempo. Porém se essa fonte é deslocada para a esquerda, por exemplo, o som chega primeiro ao ouvido esquerdo e após um pequeno atraso ao ouvido direito, pois tem que percorrer uma distância maior. Esse atraso é medido pelo cérebro é utilizado para calcular a direção do som.
- *Interaural Level Difference* (ILD) - É a diferença na amplitude do som que atinge os ouvidos. Como no caso anterior, um som cuja origem está na frente do indivíduo atinge os dois ouvidos com a mesma intensidade. Ao mover a fonte para a esquerda, a amplitude do estímulo no ouvido direito se torna menor, uma vez que o som sofre uma atenuação ao ter que atravessar ou circundar a cabeça do ouvinte. (UW)

A Figura 2.2 mostra como a distância l percorrida por um som varia com a mudança do posicionamento de sua origem.

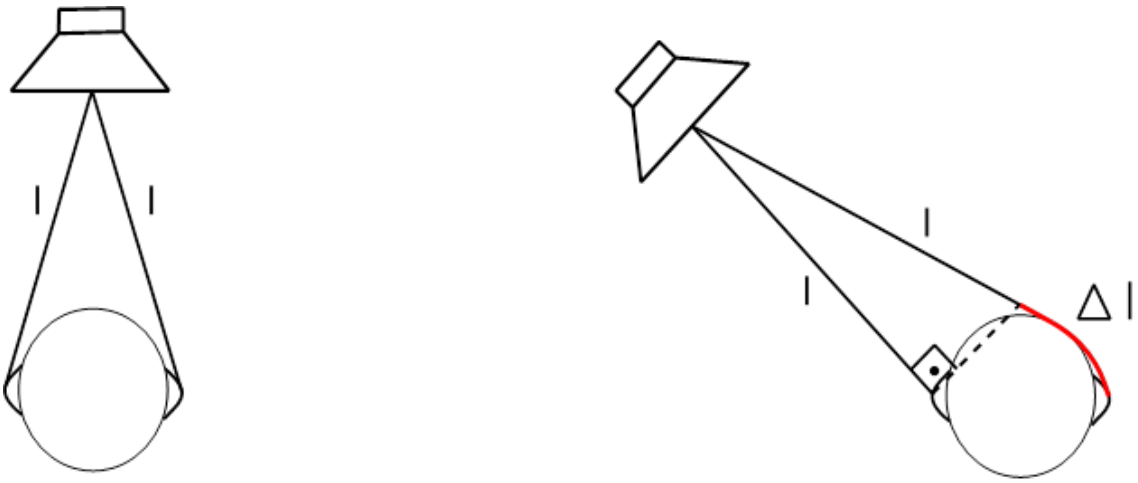


Figura 2.2 – Diferença na distância percorrida pelo som para os dois ouvidos

Somente essas duas informações são insuficientes para identificar com exatidão a localização da fonte sonora, e não oferecem nenhuma informação em relação à elevação. Um som que vem da frente do usuário, por exemplo, possui os mesmos níveis de ITD e ILD que um proveniente da posição oposta, atrás do usuário. Para melhorar o desempenho da localização

sonora, o cérebro então utiliza de diversas Funções de Transferência Relacionada à Cabeça (HRTF - *Head Related Transfer Function*). (NCSU)

Diferente da ITD e ILD, que comparam os estímulos recebidos pelos dois ouvidos, as HRTF refletem a interação do impulso sonoro com a anatomia humana. O formato das orelhas, altura dos ombros, circunferência da cabeça e diversos outros fatores influenciam e modificam os estímulos que atingem os ouvidos internos. Essas modificações são diferentes para sons provenientes de cada ponto do espaço, e podem ser identificadas por diferentes respostas ao impulso, chamadas de Respostas ao Impulso Relacionadas à Cabeça (HRIR - *Head Related Impulse Response*). A convolução de um sinal sonoro puro com a HRIR de um dado ponto no espaço resulta no som ouvido pelo indivíduo como se a fonte estivesse ali localizada. (LISTEN)

## 2.2) Orientação Espacial

Para a descrição de objetos rígidos em um espaço tridimensional uma das informações necessárias é sua orientação no espaço. Na aviação, área do conhecimento onde esse tipo de orientação é crítica, são definidos três ângulos, caracterizados na Figura 2.3, para definir tal orientação:

- Ângulo de *roll* (rolagem em português): É o ângulo de rotação em torno do eixo longitudinal da aeronave, que a atravessa do nariz à cauda. Um ângulo de *roll* positivo eleva a asa esquerda e abaixa a esquerda, e um negativo realiza o oposto.
- Ângulo de *pitch* (arfagem): Ângulo de rotação em torno do eixo lateral da aeronave, que vai de uma ponta de sua asa à outra. Um ângulo de *pitch* positivo eleva o nariz da aeronave.
- Ângulo de *yaw* (guinada): Rotação em torno de um eixo vertical que atravessa o centro de massa da aeronave. Um ângulo de *yaw* positivo move o nariz para a direita. (NASA, 2015)

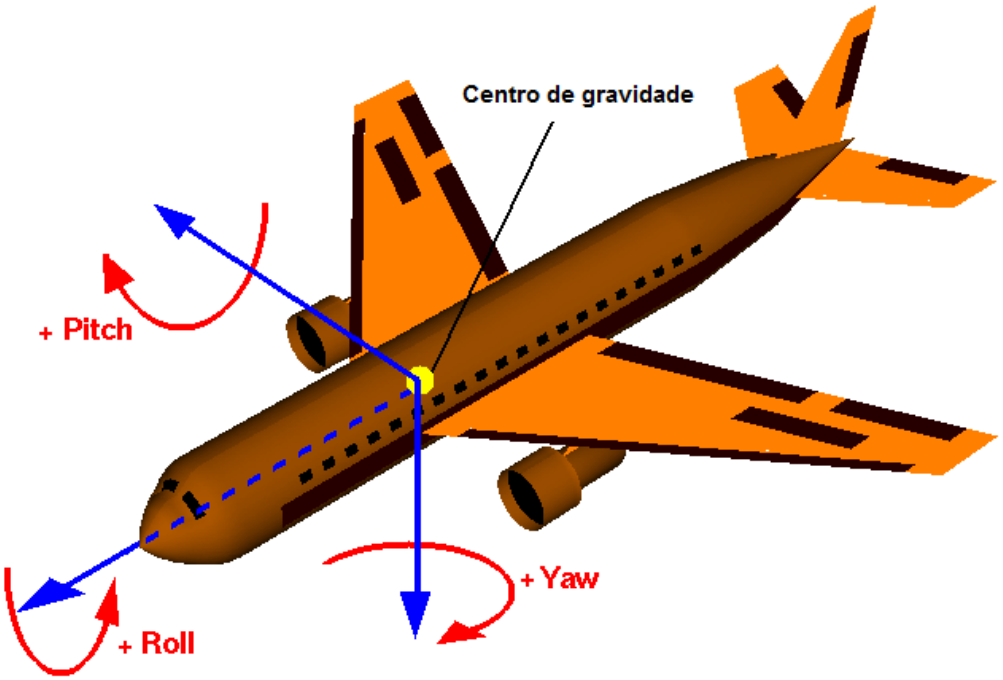



Figura 2.3 – Ângulos de *roll, pitch* e *yaw* em uma aeronave (NASA, 2015)

É possível o cálculo desses três ângulos com dados obtidos através da IMU. A biblioteca IMU (BERK, Aaron) se baseia no trabalho realizado por MADGWICK, S. O. H., para calcular através de um filtro os ângulos de *pitch*, *roll* e *yaw*, fornecidas as leituras dos eixos x, y e z de um acelerômetro e de um giroscópio, bem como a taxa de aquisição dos dados. As principais funções da biblioteca se encontram na Tabela 2.1. A não utilização do magnetômetro pela biblioteca faz com que o cálculo do ângulo de yaw não seja muito exato. Testes mostrando essa inexatidão e uma discussão sobre sua influência no desempenho do dispositivo podem ser vistas na seção 5.2).

| **Função** | **Descrição** |
|---|---|
| IMUfilter(**double** rate, **double** gyroscopeMeasurementError) | Construtor, cria objeto da classe IMUfilter. Esse filtro deve ser atualizado com uma taxa dada por “rate”. “gyroscopeMeasurementError” é um valor de calibração que deve ser ajustado para a obtenção de melhores resultados. |
| updateFilter(**double** w_x, **double** w_y, **double** w_z, **double** a_x, **double** a_y, **double** a_z) | Atualiza o filtro com os valores de velocidade angular nos três eixos em radianos por segundo e aceleração nos três eixos em metros por segundo ao quadrado. |
| **void** computeEuler() | Realiza o cálculo de ângulos de Euler com as variáveis alocadas no filtro. |
| **double** getRoll() | Retorna o ângulo de *roll* |
| **double** getPitch() | Retorna o ângulo de *pitch* |
| **double** getYaw() | Retorna o ângulo de *yaw* |
| **double** reset() | Reinicia o filtro |

Tabela 2.1 – Funções da classe IMUfilter (BERK, Aaron)

# 3) Materiais

Nessa seção os principais componentes e padrões utilizados no desenvolvimento e construção do dispositivo são introduzidos, juntamente com uma descrição de suas características e funcionamento básico.

## 3.1) Placa de desenvolvimento

A placa de desenvolvimento mbed LPC1768 (Figura 3.1) é baseada no processador ARM Cortex-M3 e é ideal para projetos embarcados e de baixo consumo. Possui diversas ferramentas úteis para a elaboração dos mais variados projetos, e seu formato com 40 pinos, compatível com placas *protoboard*, além da possibilidade de alimentação com níveis de tensão que variam de 4,5 a 9 Volts, torna a construção e teste de protótipos fácil e rápida. (MBED)

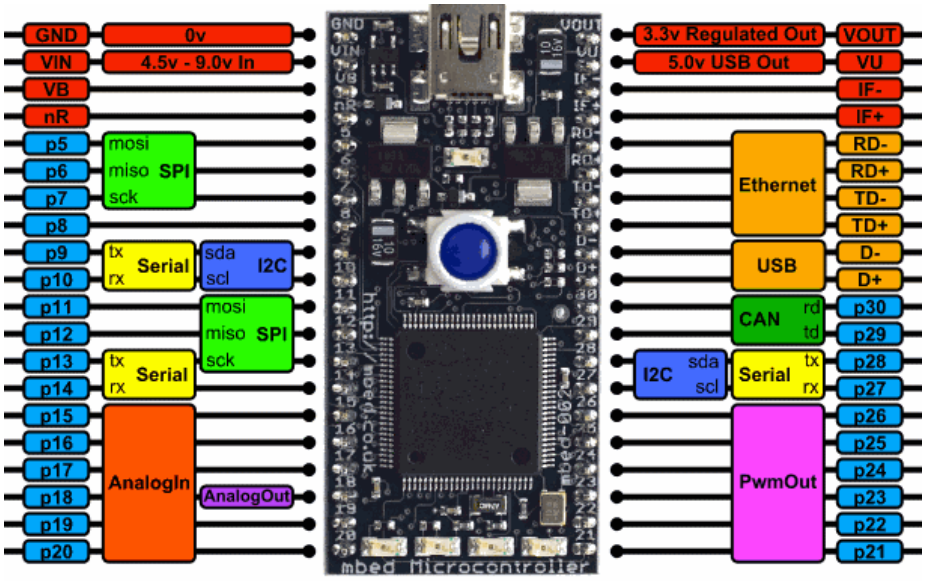



Figura 3.1 – Pinagem da placa mbed LPC1768 (MBED)

Sua programação é simples e baseada em bibliotecas com funções úteis e abrangentes, que evitam que seja necessário programar em baixo nível. Existem bibliotecas para todos os periféricos e portas de comunicação da placa, com códigos funcionais disponíveis *on-line* como exemplos, ensinando o funcionamento básico das mais diversas funções. Isso torna o aprendizado rápido e permite que o novo programador construa seus códigos facilmente e identifique e corrija seus erros com agilidade.

Além das bibliotecas padrão, o site do produto (https://developer.mbed.org/) conta com uma comunidade ativa de programadores e desenvolvedores que divulgam seus códigos e bibliotecas para os mais diversos dispositivos, além de tutoriais e dicas, sendo possível então estudar, aproveitar e reciclar código para adaptá-los às necessidades dos mais diferentes projetos, bem como tirar dúvidas e adquirir informações com outros membros da comunidade.

No site também está disponível um compilador C/C++ on-line, como mostra a Figura 3.2, para os microcontroladores da linha mbed, sendo possível assim facilmente desenvolver códigos sem a necessidade de instalação de nenhum software extra. Só é necessária a criação de um usuário e senha, e todos os seus programas serão salvos na nuvem, podendo ser acessados, modificados e compilados em qualquer computador conectado à internet. Todas as bibliotecas, inclusive as desenvolvidas por outros membros da comunidade, estão integradas a esse sistema, sendo facilmente importadas e utilizadas também sem a necessidade de download ou instalação.

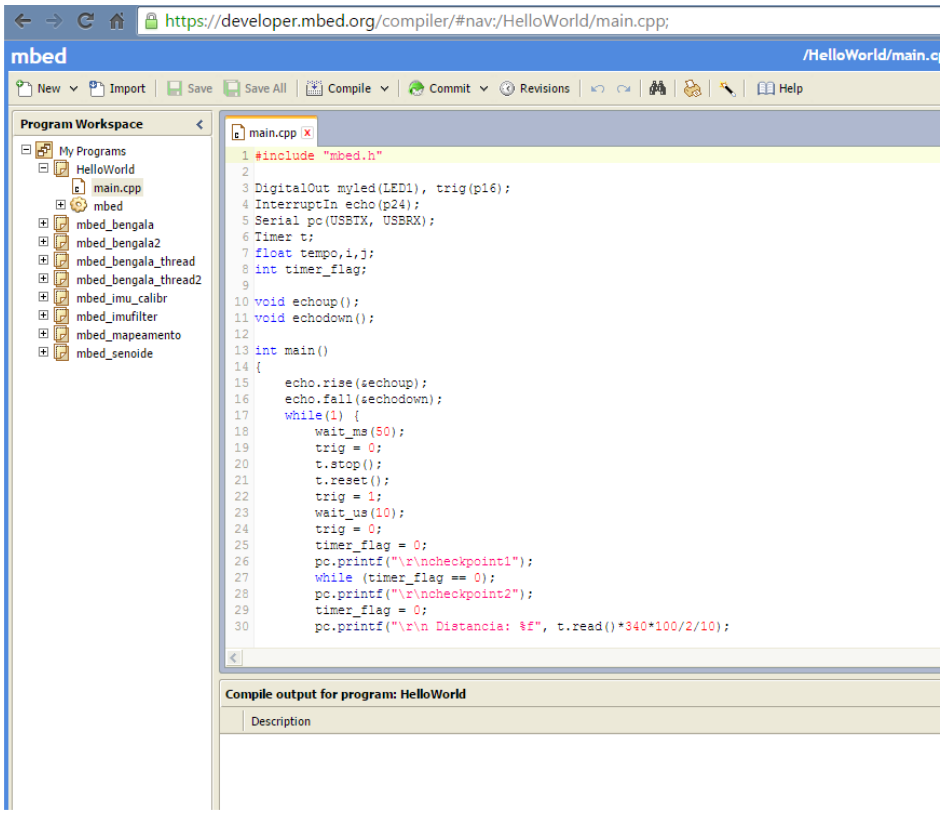



Figura 3.2 – Compilador online no site do dessenvolvedor

Quando compilado, o download do código é então feito para o computador e este pode ser programado no microcontrolador. Este processo também não depende de nenhum software ou hardware extra, com exceção de uma porta USB para a comunicação com a placa de desenvolvimento.

Ao conectar o microcontrolador ao computador via USB, este é identificado como um simples dispositivo *flash*. Basta copiar o arquivo gerado pelo compilador e colar no diretório correspondente ao dispositivo que a programação estará completa e o novo código passará a ser executado após um reset do microcontrolador.

### 3.1.1) Interfaces e bibliotecas utilizadas

São listadas nessa sub-seção as principais bibliotecas e classes utilizadas durante o desenvolvimento do código embarcado no microcontrolador para o funcionamento do dispositivo, juntamente com breves descrições de suas utilidades e funções.

#### 3.1.1.1) DigitalOut

Interface para configurar um determinado pino como saída digital, sendo possível então modificar seu nível lógico como alto (1) ou baixo (0). Qualquer um dos pinos numerados pode ser utilizado como saída digital, além dos 4 LEDs existentes na placa. As principais funções se encontram na Tabela 3.1.

| **Função** | **Descrição** |
|---|---|
| DigitalOut(PinName pin) | Configura um pino como uma saída digital. |
| **void** write(**int** value) | Escreve valor zero ou um no pino especificado. Pode ser substituída pelo caractere “**=**”. |

Tabela 3.1 – Funções importantes da biblioteca *DigitalOut* (MBED)

#### 3.1.1.2) PwmOut

Habilita um pino para gerar uma saída PWM. Os pinos disponíveis para essa função são os pinos p21 ao p 26, podendo ser configurados individualmente para diferentes valores de período, ciclo de trabalho, etc. As principais funções se encontram na Tabela 3.2.

| Função | Descrição |
|---|---|
| PwmOut (PinName pin) | Configura um pino para gerar uma saída PWM. |
| **void** write(**float** value) | Especifica o ciclo de trabalho da saída como uma porcentagem. |
| **float** read() | Lê o valor atual do ciclo de trabalho como uma porcentagem. |
| **void** period(**float** seconds) | Especifica o período da saída em segundos, mantendo o ciclo de trabalho constante. |
| **void** period_ms(**int** ms) | Especifica o período da saída em milissegundos, mantendo o ciclo de trabalho constante. |
| **void** period_us(**int** us) | Especifica o período da saída em microssegundos, mantendo o ciclo de trabalho constante. |
| **void** pulsewidth(**float** seconds) | Especifica a largura do pulso em segundos, mantendo o período constante. |
| **void** pulsewidth_ms(**int** ms) | Especifica a largura do pulso em microssegundos, mantendo o período constante. |
| **void** pulsewidth_us(**int** us) | Especifica a largura do pulso em microsegundos, mantendo o período constante. |

Tabela 3.2 – Funções importantes da biblioteca *PwmOut*. (MBED)

#### 3.1.1.3) InterruptIn

Configura um dado pino como entrada digital e aciona uma função como rotina de interrupção quando seu valor de entrada se altera de acordo com as configurações. Qualquer um dos pinos numerados, com exceção dos pinos p19 e p20 podem ser utilizados. As principais funções se encontram na Tabela 3.3.

| Função | Descrição |
|---|---|
| InterruptIn(PinName pin) | Configura um pino como entrada digital e o torna disponível para acionar uma rotina de interrupção. |
| **void** rise((*fptr)(**void**)) | Configura uma função para ser chamada como rotina de interrupção quando o valor lógico do pino se altera de zero para um. |
| **void** fall((*fptr)(**void**)) | Configura uma função para ser chamada como rotina de interrupção quando o valor lógico do pino se altera de zero para um. |

Tabela 3.3 – Funções importantes da biblioteca *InterruptIn*. (MBED)

#### 3.1.1.4) I2C

Permite a configuração de um par de pinos para operar como mestre em uma comunicação I2C, com frequência padrão de 100kHz, que pode ser alterada. Os pinos disponíveis são p9 e p28 como SDA e p10 e p27 como SCL. É importante destacar que para essa interface é necessária a utilização de resistores de *pull-up* em todos os pinos utilizados. As principais funções se encontram na Tabela 3.4.

| Função | Descrição |
|---|---|
| I2C(PinName sda, PinName scl) | Configura dois pinos, especificados por “sda” e “scl” para caracterizar um dispositivo mestre em um barramento I2C. |
| **void** frequency(**int** hz) | Define a frequência de *clock* da interface I2C. |
| **void** read(**int** address, **char** *data, **int** length, **bool** repeated=false) | Lê do escravo de endereço “address” uma quantidade de bytes apontados por “*data” especificada por “length”. A variável “repeated” especifica se é neccessário ou não o envio de um uma condição de parada ao fim da transmissão (false para envio da condição, true para o não envio). A função retorna o valor zero se foi recebido um bit de reconhecimento, indicando o sucesso da transmissão, ou um valor diferente de zero caso esse bit não seja recebido. |
| **void** write(**int** address, **char** *data, **int** length, **bool** repeated=false) | Análogo à função anterior, mas para o envio de dados. |

Tabela 3.4 – Funções importantes da classe I2C (MBED)

#### 3.1.1.5) Timer

Utilizado para criar e utilizar um contador para intervalos de tempo que variam de microssegundos a segundos. Diversos temporizadores podem ser criados e manipulados individualmente. As principais funções se encontram na Tabela 3.5.

| **Função** | **Descrição** |
|---|---|
| Timer t | Cria um objeto da classe Timer. |
| **void** start() | Inicia contagem de tempo. |
| **void** stop() | Para contagem. |
| **void** reset() | Zera a contagem. |
| **float** read() | Lê o valor armazenado no contador em segundos. |
| **int** read_ms() | Lê o valor armazenado no contador em milissegundos. |
| **int** read_us() | Lê o valor armazenado no contador em milissegundos. |

Tabela 3.5 – Funções importantes da classe *Timer*. (MBED)

#### 3.1.1.6) Timeout

Cria um objeto da classe *Timeout*, que possibilita a execução de uma rotina de interrupção após um dado intervalo de tempo. As principais funções se encontram na Tabela 3.6.

| **Função** | **Descrição** |
|---|---|
| Timeout t | Cria um objeto da classe *Timeout*. |
| **void** attach(**void**(*fptr)(**void**), **float** t) | Especifica que uma dada função seja chamada como rotina de interrupção após um intervalo de tempo “t”, dado em segundos. |
| **void** attach_us(**void**(*fptr)(**void**), timestamp_t) | Análogo à anterior, mas o intervalo é dado em microssegundos |
| **void** detach() | Faz com que a função não seja mais chamada. |

Tabela 3.6 – Funções importantes da classe *Timeout*. (MBED)

#### 3.1.1.7) Ticker

Cria um objeto da classe *Ticker*, que possibilita a execução de uma rotina de interrupção repetidamente com um período especificado. As principais funções se encontram na Tabela 3.7.

| **Função** | **Descrição** |
|---|---|
| Ticker t | Cria um objeto da classe *Ticker*. |
| **void** attach(**void**(*fptr)(**void**), **float** t) | Especifica que uma dada função seja chamada como rotina de interrupção a cada intervalo de tempo t, dado em segundos. |
| **void** attach_us(**void**(*fptr)(**void**), timestamp_t) | Análogo à anterior, mas o intervalo é dado em microssegundos. |
| **void** detach() | Faz com que a função não seja mais chamada. |

Tabela 3.7 – Funções importantes da classe *Ticker*. (MBED)

#### 3.1.1.8) RTOS - Real Time Operating System

Essa biblioteca permite a criação de diferentes tarefas a serem executadas pelo programa. A função “main” por definição é uma das tarefas, e a primeira a ser executada. Nela devem ser estabelecidas quais funções serão as outras tarefas do sistema. Uma tarefa pode estar em um de quatro possíveis estados:

- Executando - A tarefa está sendo realizada e está realizando as devidas funções. Apenas uma tarefa por vez pode estar nesse estado.
- Pronta - A tarefa está pronta para ser executada, e mudara para o estado “Executando” assim que a tarefa que está no momento sendo executada se encerre ou mude para o estado “Esperando”.
- Esperando - A tarefa está esperando que algum evento ocorra para que ela possa então ser executada.
- Inativa - Quando uma tarefa foi encerrada ou ainda não foi criada, seu estado é inativo.

#### 3.1.1.9) Wait

Função que faz com que o programa aguarde um período de tempo sem executar nenhum comando. Suas variações são:

**void** wait ( **float** s ) - espera um tempo “s” em segundos

**void** wait_ms (**int** ms) - espera um tempo “ms” em milissegundos

**void** wait_us(**int** us) - espera um tempo “us” em microssegundos

### 3.2) PWM

PWM (*Pulse Width Modulation*, Modulação por Largura de Pulso) é uma técnica largamente utilizada em sistemas eletrônicos para a criação de sinais analógicos a partir de um sistema digital. É caracterizada por um sinal de onda quadrada cujo ciclo de trabalho, a razão entre o tempo que o sinal permanece em nível lógico alto e seu período, é variado a fim de produzir um sinal cujo valor médio seja controlado de acordo com as necessidades da aplicação. (ADAM).

Essa técnica é extremamente útil para o controle de dispositivos através de variação de tensão, uma vez que esse valor pode ser facilmente alterado. Uma das aplicações mais simples é o controle de luminosidade de um LED. O brilho é diretamente proporcional à tensão aplicada aos seus terminais, logo, variando-se a tensão média de alimentação a luminosidade também se altera.

A característica mais importante de um sinal de PWM é sua frequência. Ela deve ser alta o suficiente para que o dispositivo onde esse sinal é aplicado não responda aos diversos pulsos individuais, mas sim ao valor médio geral da onda. A aplicação de um sinal PWM com frequência baixa ao LED mencionado anteriormente, por exemplo, resultaria no LED visualmente acendendo e apagando ao invés de um brilho constante.

Sua implementação pode ser feita de diferentes maneiras. A mais comum é o PWM com frequência fixa. Neste modo a onda quadrada mantém sua frequência constante e o ciclo de trabalho é variado alterando-se o tempo em que a onda se mantém em nível lógico alto, este nunca ultrapassando o período total da onda. Outros métodos de implementação utilizam uma onda quadrada de frequência variável, fixando o tempo em nível alto ou em nível baixo, e alterando a duração do estado complementar, juntamente com o período do sinal. (VASCA, F. & IANELLI, L., 2012)

A utilização de PWM se encaixa perfeitamente em sistemas eletrônicos devido à sua natureza liga/desliga, facilmente implementada com semicondutores, microcontroladores, etc.

Permite um controle preciso de tensão sem a necessidade de componentes mecânicos ou analógicos, e reduz significantemente as perdas ôhmicas no sistema se comparados à métodos alternativos como reostatos e bancos de resistores, uma vez que as perdas em semicondutores nesse tipo de aplicação são muito pequenas. (VASCA, F. & IANELLI, L., 2012)

No âmbito desse projeto, o PWM foi utilizado para a geração de sinais, variando seu ciclo de trabalho de acordo com valores previamente amostrados. Com isso, ao filtrar as altas frequências da onda quadrada o resultado é o sinal que se deseja reproduzir. A Figura 3.3 mostra a geração de uma onda senoidal a partir desse princípio.

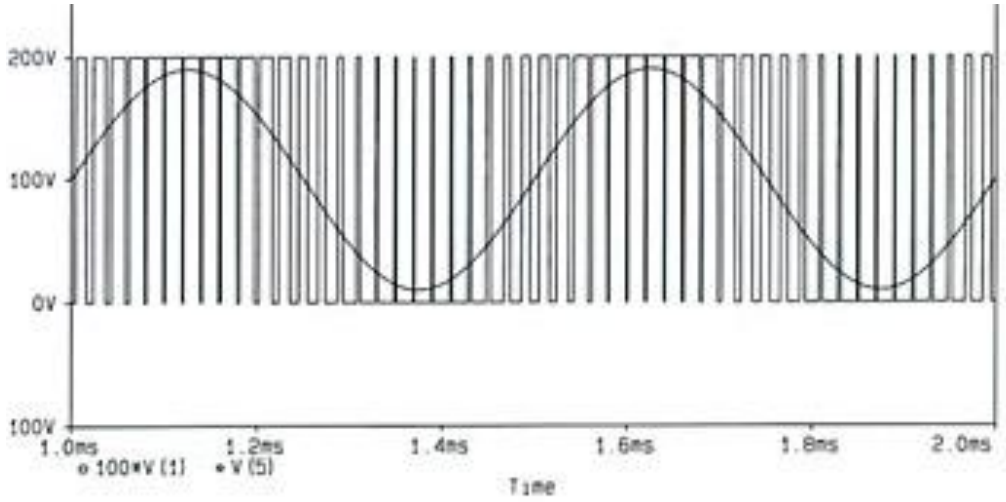



Figura 3.3 - Exemplo de PWM utilizado na geração de uma senóide (ADAM)

## 3.3) I2C

O barramento I2C (*Inter Integrated Circuit*) foi desenvolvido pela *Phillips Semiconductors*, hoje *NXP Semiconductors*, como um método simples e eficaz de controle e comunicação entre circuitos integrados. Necessitando de apenas dois fios, o barramento I2C utiliza comunicação serial de 8 bits para transmitir bidirecionalmente até 100kbits/s em seu modo padrão, mas podendo atingir até 3,4Mbits/s em seus modos mais velozes. (NXP, 2014a)

Os dois fios, SDA (*serial data*) e SCL (*serial clock*), são utilizados para transmitir informações entre dispositivos, cada um identificado por um endereço único. Qualquer dispositivo pode transmitir ou receber dados, incluindo microcontroladores, displays, sensores, circuitos de memória, entre outros. A Tabela 3.8 define importantes terminologias utilizadas na descrição deste barramento

| Termo | Descrição |
|---|---|
| Transmissor | Dispositivo que envia dados ao barramento |
| Receptor | Dispositivo que recebe dados do barramento |
| Mestre | Dispositivo que inicia uma transmissão, gera o sinal de clock e encerra uma transmissão |
| Escravo | Dispositivo endereçado pelo mestre |

Tabela 3.8 – Terminologia do barramento I2C (NXP, 2014a)

Quando uma transmissão de dados é desejada, o dispositivo que a controlará (esteja ele enviando ou recebendo dados) inicia a transferência e se torna o dispositivo mestre, controlando o sinal de *clock* e sendo responsável pelo início e término da transmissão. O dispositivo que ele endereça será o escravo.

O barramento I2C é multi-mestre, ou seja, aceita que mais de um dispositivo tome o controle das transmissões, porém não ao mesmo tempo. Como existe a possibilidade de mais de um dispositivo tentar iniciar uma transmissão de dados ao mesmo tempo medidas são tomadas para que isso não ocorra, sendo elas a sincronização e a arbitragem. Essas medidas não se fazem necessárias em sistemas com apenas um dispositivo mestre, como é o caso deste projeto. (NXP, 2014a)

### 3.3.1) Linhas SDA e SCL

Ambas as linhas do barramento são bidirecionais e conectadas à alimentação do circuito através de resistores de *pull-up*, o que faz com que o nível lógico quando não estão sendo utilizadas são sempre alto. Como diferentes tecnologias utilizam I2C, os níveis de tensão referentes aos níveis lógicos alto e baixo não são fixos, e variam de acordo com a tensão de alimentação do circuito. Nível baixo é caracterizado como menor que 30% e nível alto como maior que 70% da tensão de alimentação. (NXP, 2014a)

### 3.3.2) Transmissão de dados

Qualquer transmissão começa com uma condição de início e termina com uma condição de parada. Uma variação do nível lógico da linha de dados SDA de alto para baixo, durante um

período de nível alto na linha de *clock* SCL caracteriza a condição de início, enquanto que o inverso indica uma condição de parada (Figura 3.5). Ambas as condições são sempre geradas por um mestre, e a linha é considerada ocupada imediatamente após a condição de início até algum tempo após a condição de parada. Caso ao invés desta seja gerada outra condição de início, o barramento continua ocupado (NXP, 2014a).

Durante a transmissão os dados são considerados válidos caso, diferentemente das condições previamente mencionadas, o nível lógico na SDA permaneça constante durante todo o período alto do *clock*. Ou seja, o sinal na SDA só pode se alterar durante o período baixo na SCL, como visto na Figura 3.4.

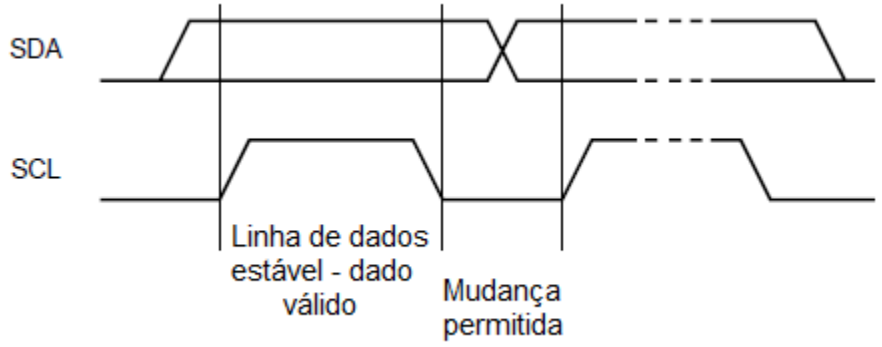



Figura 3.4 – Validação dos dados (NXP, 2014a)

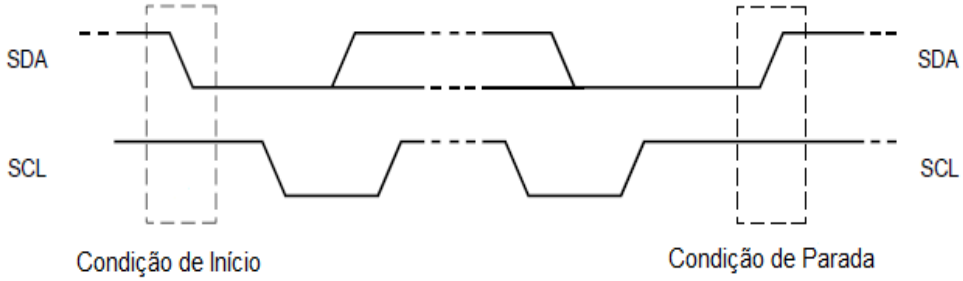



Figura 3.5 – Condições de início e parada no barramento I2C

Bytes transmitidos pela SDA devem conter exatamente 8 bits e ser seguidos por um bit de reconhecimento, sendo transmitidos a partir do bit mais significativo (MSB). O número de bytes transmitidos, porém, é ilimitado.

Caso um escravo não seja capaz de transmitir ou receber outro bit até que uma operação interna seja executada, este pode manter o nível lógico da SCL para fazer com que o mestre entre em um estado de espera.

O bit de reconhecimento é responsável por sinalizar ao transmissor que a transmissão do byte obteve sucesso e um novo pode ser enviado. O mestre libera a linha SDA para que o escravo a possa colocar em nível baixo mantendo-a estável durante o próximo período de *clock*, o 9o a partir do início da transmissão do byte. Caso a SDA permaneça em nível alto durante este período é caracterizado um bit de não-reconhecimento, e o mestre pode então enviar uma condição de parada, abortando a transmissão, ou uma condição de início para começar uma nova transferência de dados. (NXP, 2014a)

O endereçamento do dispositivo escravo é feito logo após o envio da condição de início. O primeiro byte após essa condição deve conter 7 bits com o endereço correspondente ao dispositivo que se quer alcançar, seguido de um oitavo bit que sinaliza se a operação será de leitura (nível alto) ou escrita (nível baixo). O fim de uma transmissão sempre deve ser dado através de uma condição de parada, porém o mestre pode se comunicar com diferentes escravos, ou realizar diversas operações de leitura e escrita enviando novas condições de início e novos endereços e bits de leitura/escrita. Possíveis formatos de transferência de dados são:

- O mestre transmissor envia dados para um escravo receptor. Neste modo o fluxo de dados não muda de direção e o escravo manda bits de reconhecimento após cada byte.
- O mestre lê dados do escravo imediatamente após endereça-lo. Neste caso, o mestre é o transmissor dos primeiros bytes contendo o endereço do escravo e do registrador que se deseja acessar, e recebe bits de reconhecimento do escravo após cada um. Depois disso, ele passa a ser um mestre receptor e envia apenas bits de reconhecimento ao escravo, agora transmissor. Ao fim da transmissão, o mestre envia um sinal de não reconhecimento e uma condição de parada.
- Formato misto, onde o mestre ora envia, ora recebe dados. Para a mudança é necessária uma nova condição de início, um novo endereço do escravo (repetido ou não) e um novo bit de leitura/escrita. Caso o mestre deseje mudar de receptor para transmissor, um bit de não reconhecimento é enviado antes da nova condição de início. (NXP, 2014a)

### 3.4) Sensor HC-SR04

O sensor HC-SR04 é um medidor de distância que utiliza ondas de frequência ultrassônica para encontrar e medir a distância que objetos se encontra, utilizado em aplicações diversas devido ao seu baixo custo, pequenas dimensões e facilidade de utilização. Tem alcance de até 4 metros e sua precisão chega a 3 mm, com um ângulo de detecção de aproximadamente 30 graus, porém seu desempenho varia com o tipo de superfície do objeto detectado. (CYTRON)

Seu princípio de funcionamento é simples, baseado em um emissor e um receptor de ondas sonoras. Ao ser acionado, o sensor emite um trem de pulsos à uma frequência de 40 kHz. Entra, então, em um estado de espera até que seu receptor detecte esse mesmo pulso após ele ter refletido em alguma superfície, como mostra a Figura 3.6. A diferença entre o tempo de partida e o tempo de chegada desse sinal é utilizado para calcular a distância que superfície refletora se encontra em relação ao sensor. Sabendo-se a velocidade do som, o cálculo da distância é feito segundo a seguinte equação:

$$L = \frac{\Delta t}{2v}$$

$L =$ Distância medida, em metros

$\Delta t$ = Tempo medido, em segundos

$v =$ Velocidade do som no ar, em metros por segundo (aprox. 343 m/s)

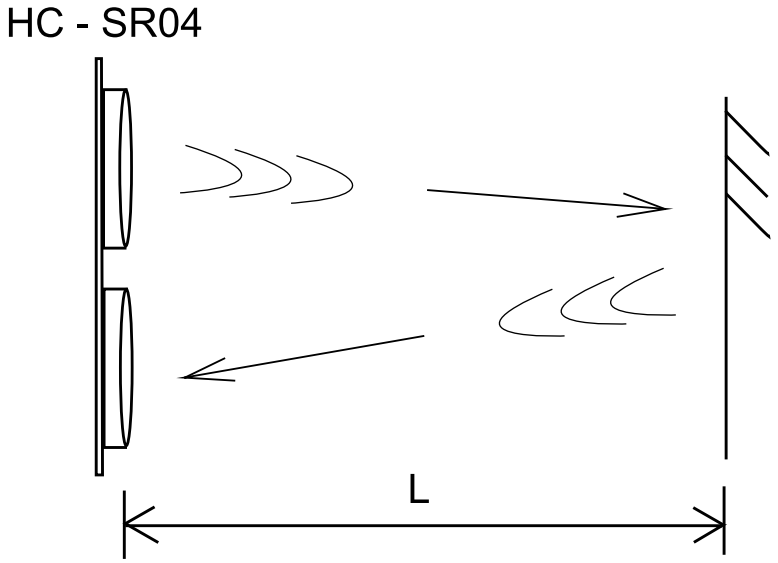



Figura 3.6 – Operação do sensor de distância

Possui quatro pinos que podem ser vistos na Figura 3.7, e são descritos na Tabela 3.9. Seu funcionamento se inicia com os pinos *Trig* e *Echo* em nível baixo. Para se iniciar a medida deve-se aplicar um pulso de nível alto no pino *Trig* com duração de pelo menos 10 microssegundos, quando então o sensor iniciará o processo de emissão dos pulsos sonoros. Ao fim da emissão, o pino *Echo* é levado a nível lógico alto, e assim permanece até que o pulso refletido seja detectado pelo receptor ou após um tempo de 38ms caso não haja detecção. Medindo-se o tempo que o pino *Echo* permanece em nível alto têm-se então o tempo de viagem do pulso.

Características elétricas e mecânicas:

- Tensão de operação: 5 Vcc
- Corrente quiescente: < 2 mA
- Corrente de operação: 15 mA
- Alcance de medida: 2 - 400 cm
- Precisão: 3 mm
- Dimensões: 45mm x 20mm x 15 mm


Figura 3.7 – Sensor HC-SR04 (CYTRON)

| **Nome** | **Função** |
|---|---|
| Vcc | Pino de alimentação do sensor, deve ser conectado à 5V. |
| *Trig* | Disparo do sensor, para iniciar a medida |
| *Echo* | Após a medida, retorna pulso com largura proporcional à distância encontrada |
| GND | Pino terra |

Tabela 3.9 – Descrição dos pinos do sensor HC-SR04 (CYTRON)

### 3.5) MinIMU-9 v3

A placa MinIMU-9 v3 (Figura 3.8) é uma IMU (*Inertial Measurement Unit* - Unidade de Medição Inercial) composta pelo giroscópio L3GD20H e o acelerômetro/magnetômetro LSM303D. Ela combina os dois componentes, de difícil utilização em protótipos e pequenos projetos devido às suas dimensões pequenas não compatíveis com placas *protoboard* e nível de tensão de trabalho restrita, em uma única unidade que abrange, além dos dois circuitos integrados um circuito regulador de tensão e demais componentes eletrônicos para realizar a interface com diferentes níveis de tensão. (POLOLU)

Os dois componentes, que possuem interface de comunicação I2C são também conectados à um mesmo barramento para fácil acesso e leitura de medidas individuais através dos pinos da IMU, esses sim compatíveis com *protoboards* para fácil utilização e teste. Além disso, um pino extra permite, se utilizado, que os endereços I2C de cada componente seja alterado, sendo assim possível a utilização de até duas IMU em um mesmo barramento. Possui 6 pinos, descritos na Tabela 3.10.

O componente L3GD20H é um giroscópio digital de três eixos, com leituras de 16 bits (complemento de 2) por eixo e fundo de escala mínimo de 245 graus por segundo, atinge precisões de até $8{,}75 \times 10^{-3}$ graus por segundo por LSB (bit menos significativo). Seu endereço é 110101xb, sendo possível ser alterado através do pino SDO da IMU. Caso o pino seja conectado à Vcc, o LSB tem valor 1, sendo então o endereço 1101011b. Caso não seja conectado, o pino é aterrado e o LSB é 0, mudando o endereço para 1101010b. (ST-1)

LSM303D é um circuito integrado que inclui um acelerômetro linear e um magnetômetro de três dimensões. Com leituras de 16 bits e fundos de escala mínimos de ±2 g para o acelerômetro e ±2 gauss para o magnetômetro, possui precisão de até 0.061 mg/LSB e 0.080 mgauss/LSB. As duas partes podem ser ligadas/desligadas e configuradas individualmente sem interferência uma na outra. (ST-2)

Características físicas e elétricas da IMU:

- Tensão de operação: 2,5 - 5,5 Vcc
- Corrente: 6 mA
- Formato dos dados: Uma palavra de 16 bits para cada eixo, cada medida.
- Fundo de escala:
    - Giroscópio: ±245, ±500 ou ±2000 graus por segundo
    - Acelerômetro: ±2, ±4, ±6, ±8 ou ±16 g
    - Magnetômetro: ±2, ±4, ±8 ou ±12 gauss (POLOLU)

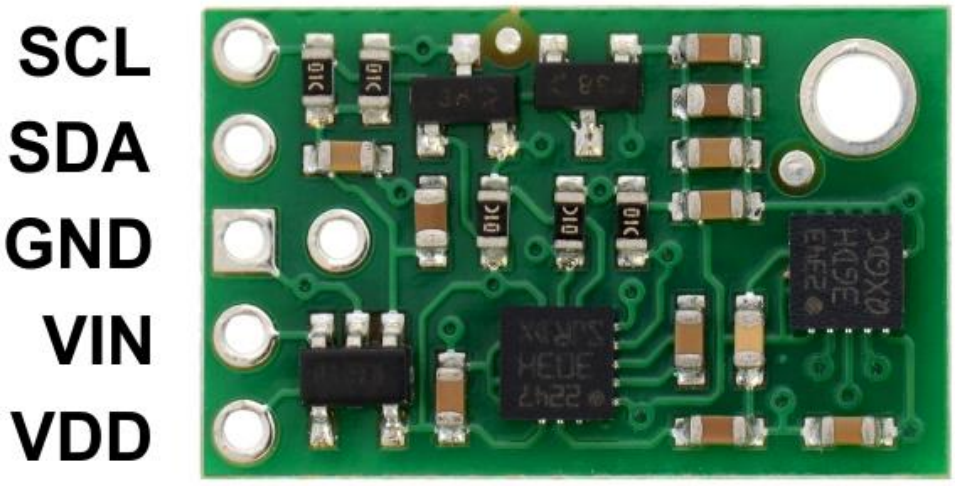



Figura 3.8 – minIMU-9 v3 (POLOLU)

| Nome | Função |
|---|---|
| SCL | Linha de *clock* do barramento I2C |
| SDA | Linha de dados do barramento I2C |
| GND | Pino terra |
| VIN | Alimentação da placa |
| VDD | Fornece uma tensão de 3,3 V de saída. Pode ser utilizado como pino de alimentação se a tensão disponível for 3,3 V, porém nunca deve ser conectada a níveis de tensão maiores. |
| SD0 | Pode ser utilizado para alterar o endereço associado ao dispositivo. |

Tabela 3.10 – Descrição dos pinos da IMU (POLOLU)

A configuração dos dispositivos deve ser utilizando a interface I2C. Valores de configuração devem ser escritos em registradores específicos, modificando as características do sensor, ativando e desativando funções, etc. Alguns registradores importantes são apresentados e descritos nas tabelas Tabela 3.11 e Tabela 3.12.

| Nome | Endereço | Função |
|---|---|---|
| CTRL1 | 20h | Responsável pela seleção da taxa de aquisição de medidas, ligar/desligar o sensor e habilitar/desabilitar os eixos individualmente. |
| CTRL4 | 23h | Seleção do fundo de escala e sensibilidade do dispositivo. |
| OUT_X_L | 28h | Velocidade angular lida no eixo X, em complemento de dois. |
| OUT_X_H | 29h | |
| OUT_Y_L | 2Ah | Velocidade angular lida no eixo Y, em complemento de dois. |
| OUT_Y_H | 2Bh | |
| OUT_Z_L | 2Ch | Velocidade angular lida no eixo Z, em complemento de dois. |
| OUT_Z_H | 2Dh | |

Tabela 3.11 – Registradores importantes do L3GD20H. (ST-2)

| Nome | Endereço | Função |
|---|---|---|
| OUT_X_L_M | 08h | Leitura magnética no eixo X, em complemento de dois. |
| OUT_X_H_M | 09h | |
| OUT_Y_L_M | 0Ah | Leitura magnética no eixo Y, em complemento de dois. |
| OUT_Y_H_M | 0Bh | |
| OUT_Z_L_M | 0Ch | Leitura magnética no eixo Z, em complemento de dois. |
| OUT_Z_H_M | 0Dh | |
| CTRL1 | 20h | Responsável pela seleção da taxa de aquisição de medidas e ligar/desligar o acelerômetro. |
| CTRL2 | 21h | Seleção de fundo de escala e sensibilidade do acelerômetro. |
| CTRL5 | 24h | Seleção da sensibilidade e taxa de leitura do magnetômetro. |
| CTRL6 | 25h | Seleção do fundo de escala do magnetômetro. |
| OUT_X_L_A | 28h | Leitura da aceleração no eixo X, em complemento de dois. |
| OUT_X_H_A | 29h | |
| OUT_Y_L_A | 2Ah | Leitura da aceleração no eixo Y, em complemento de dois. |
| OUT_Y_H_A | 2Bh | |
| OUT_Z_L_A | 2Ch | Leitura da aceleração no eixo Z, em complemento de dois. |
| OUT_Z_H_A | 2Dh | |

Tabela 3.12 – Registradores importantes do LSM303D (ST-1)

### 3.5.1) Leitura e escrita de dados

Tanto para leitura quanto para escrita, os componentes funcionam sempre como um dispositivo escravo no barramento.

A leitura e escrita nos registradores deve ser feita seguindo uma sequência correta. O primeiro byte enviado pelo mestre deve conter o endereço do dispositivo, seguido de um bit de escrita. Em seguida é enviado um sub-endereço de 8 bits: 7 LSB correspondentes ao endereço do registrador que se deseja alcançar e 1 MSB que permite ou não o auto-incremento desse endereço (1 para auto-incremento, 0 para não auto-incremento). Esse último permite a leitura/escrita de diferentes registradores em sequência sem a necessidade de repetir o processo de endereçamento diversas vezes.

Para escrita, osbytes subsequentes serão escritos no registrador endereçado pelo sub-endereço e nos registradores seguintes caso a opção de auto-incremento seja selecionada. Para leitura, mais um passo é necessário. Após a escrita do sub-endereço, uma nova condição de início é enviada, e é necessário endereçar o escravo novamente, mas desta vez seguido de um bit de leitura. A leitura então é feita do registrador endereçado pelo sub-endereço e também dos seguintes caso o bit de auto incremento esteja em nível alto. (ST-2)

## 3.6) Base de dados de HRTF e HRIR Listen

A base de dados do projeto LISTEN contempla o resultado de diversas medidas de HRTF e HRIR para diversos cobaias e posições de fonte sonora. Tais informações foram adquiridas pelos pareceiros do projeto, *AKG Acoustics* e IRCAM – *Institut de Recherche et Coordination Acoustique/Musique*, posicionando voluntários um a um dentro de uma câmara anecóica e um alto-falante em diversas posições pré-estabelecidas em seu interior. Um pulso sonoro então era gerado pelo alto falante em cada uma das posições e gravado por microfones inseridos dentro de cada ouvido do indivíduo ao centro da sala. (LISTEN)

Essas gravações estão disponíveis para uso no site do projeto (http://recherche.ircam.fr/equipes/salles/listen/index.html) e é possível, ao reproduzi-las em um fone de ouvido *stereo*, identificar com facilidade a localização correta da fonte sonora. Foi decidida a utilização desta base de dados para a emulação de diferentes azimutes no projeto. As gravações correspondentes ao primeiro voluntário foram adquiridas e analisadas, e apesar de estarem disponíveis para diversos valores de azimute e elevação, apenas as respostas

pertinentes ao projeto foram utilizadas, ou seja, valores de elevação zero e azimutes que variam de -75 a 75 graus (ver seção 4.2), tomando como zero a região frontal do ouvinte.

Um exemplo pode ser visto na Figura 3.9, onde primeiros 512 pontos das gravações das HRIR do azimute 45 graus à esquerda, elevação zero graus dos ouvidos esquerdo e direito estão representados. Pode-se perceber as sutis diferenças entre os dois canais que são interpretadas pelo cérebro para localizar a fonte sonora.

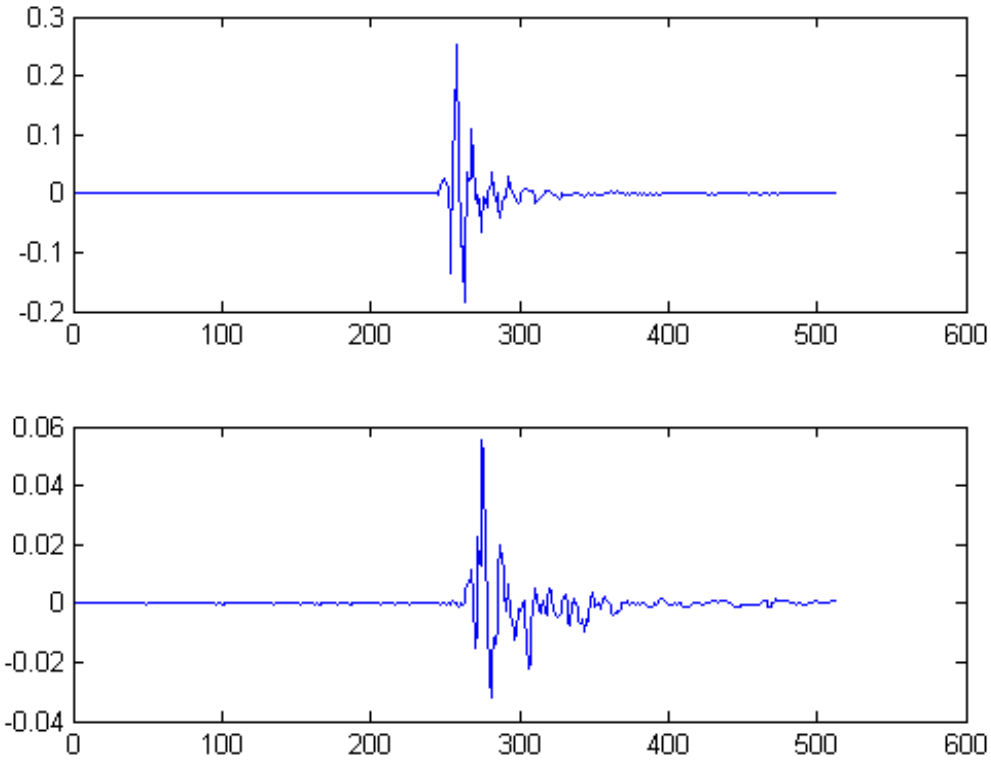



Figura 3.9 – Canais da HRIR de azimute 45 graus, esquerda, elevação zero

## 3.7) Hardware

O projeto foi dividido em duas partes bem definidas: Sensoriamento e processamento. Essa divisão foi feita devido à necessidade de os sensores estarem fisicamente presos à bengala, mais especificamente próximo à sua extremidade. Devido à própria natureza da utilização da bengala e devido ao fato de estar afastada do corpo do usuário, essa é uma posição bastante exposta à impactos e choques, deixando então qualquer componente ali posicionado

bastante vulnerável. Por esse motivo foi decidido que no extremo do instrumento seriam instalados somente os componentes essenciais para as medições, ou seja, os três sensores de distância e a unidade de medição inercial. Deste modo também se evita o excesso de peso na bengala, o que poderia causar desconforto ao usuário durante a utilização do aparelho.

O microcontrolador, por ser o componente mais caro do projeto e mais sensível à impactos, fica localizado junto ao corpo do usuário com os demais elementos que não estão na bengala.

O hardware foi separado em dois compartimentos plásticos: A unidade de sensoriamento e a unidade de processamento. A comunicação entre as duas se faz a partir de dois cabos: um cabo do tipo *flatwire* de duas vias que leva alimentação para a unidade de sensoriamento, e um cabo tipo *flatwire* de 8 vias para a comunicação dos sensores com a unidade de processamento. A escolha da utilização de um cabo foi também resultado dos critérios previamente mencionados. A não necessidade de hardware extra evita colocar em risco eventuais componentes necessários para algum tipo de comunicação sem fio, além de manter o custo geral do aparelho mais baixo. Existem problemas com a utilização de cabos para comunicação digital, existindo a possibilidade de ruídos e comprometimento de dados devido à impedância dos fios, mas testes realizados (Ver seção 5.1)) mostraram que a escolha do cabo neste caso não interfere no desempenho do projeto.

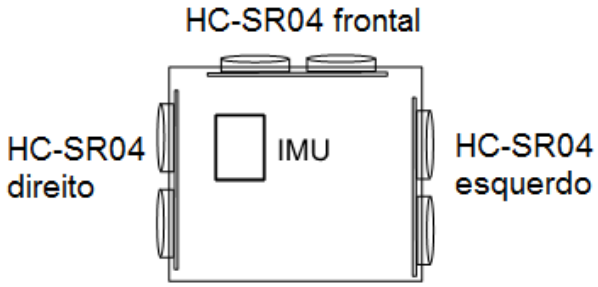



Figura 3.10 – Esboço da unidade de sensoriamento

Na unidade de sensoriamento foram posicionados os três sensores HC-SR04, posicionados segundo o esboço na Figura 3.10, e também a unidade de medição inercial minIMU-9 v3. No compartimento plástico também está localizada uma simples placa de circuito que realiza a interface entre todos os sensores, o cabo de comunicação, alimentação e terra. Por

último, estão presentes dois resistores de *pull-up* necessários para a comunicação I2C da IMU. O posicionamento dos sensores de distância foi definido segundo o formato da caixa. Verificou-se que com a distribuição escolhida os resultados foram satisfatórios (ver seção 5.4), portanto esta foi mantida. A unidade de sensoriamento posicionada em um monopod, que foi utilizado para imitar uma bengala, pode ser vista na Figura 3.11, e suas vistas lateral e frontal na Figura 3.12.

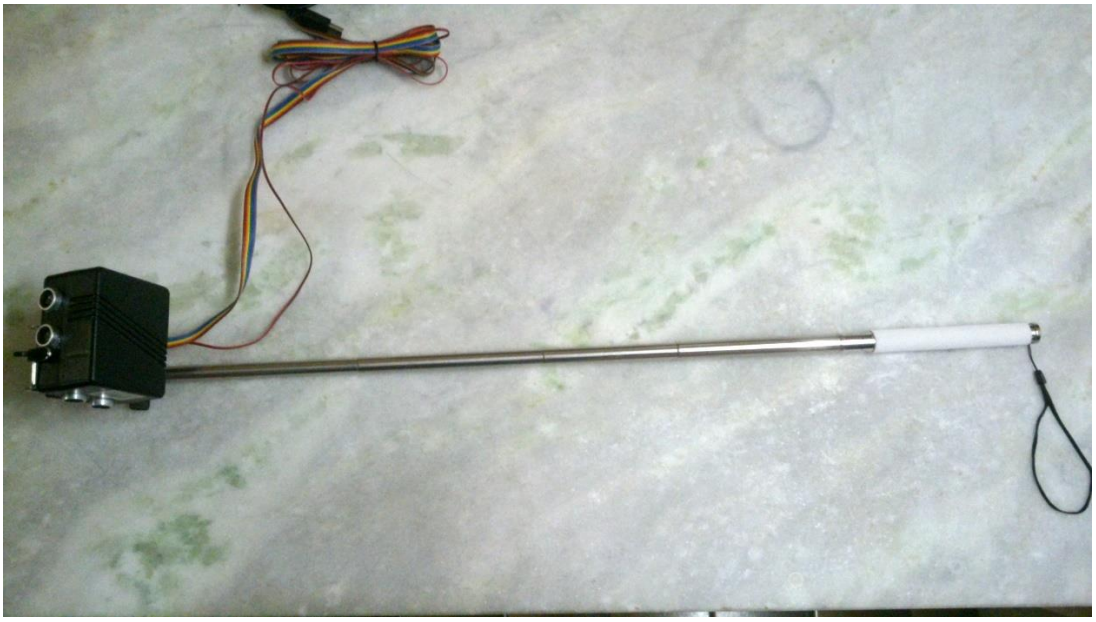


Figura 3.11 – Unidade de sensoriamento posicionada em um *monopod*

Figura 3.12 – Vista lateral e frontal da unidade de sensoriamento

Na unidade de processamento (Figura 3.13) está localizado o microcontrolador, alimentado diretamente por uma bateria de 9 volts (DURACELL), intermediado por uma chave liga/desliga. Essa bateria também alimenta um regulador de tensão LM7805 (FAIRCHILD), que fornece 5 volts para a unidade de sensoriamento. Oito portas do microcontrolador são ligadas à

um conector para o cabo de 8 vias, realizando a comunicação com a outra unidade. A lista de pinos utilizados e suas funções pode ser vista na Tabela 3.13.

| **Pino** | **Função** |
|---|---|
| p9 | Linha de dados do barramento I2C |
| p10 | Linha de *clock* do barramento I2C |
| p11 | Pino *Echo* do sensor de distância esquerdo |
| p12 | Pino *Echo* do sensor de distância central |
| p13 | Pino *Echo* do sensor de distância direito |
| p14 | Pino *Trig* do sensor de distância esquerdo |
| p15 | Pino *Trig* do sensor de distância central |
| p16 | Pino *Trig* do sensor de distância direito |
| p21 | Saída PWM, canal *stereo* direito |
| p22 | Saída PWM, canal *stereo* esquerdo |

Tabela 3.13 – Utilização dos pinos do microcontrolador

Figura 3.13 – Unidade de processamento

A geração de som para este projeto deve ser *stereo*, portanto são utilizados dois canais diferentes e independentes, que enviam sinais para os fones de ouvido. Os pinos responsáveis pelos canais são utilizados como saídas PWM, com frequência de operação de 250 kHz. Para

filtrar essa alta frequência antes de enviar os sinais aos fones eles passam por um simples filtro passa baixa, com frequência de corte de aproximadamente 32 kHz. Esse valor foi escolhido pois as frequências audíveis vão de algumas dezenas de hertz até frequências próximas a 20 kHz. Deste modo, o filtro deve possuir frequência de corte maior que esse valor para não interferir no som, mas baixa o suficiente para eliminar as frequências do PWM. Após a filtragem é feita a conexão com um *jack* para fone de ouvido *stereo* de 3.5 mm, compatível com a grande maioria dos fones de ouvido no mercado.

O esquema elétrico completo pode ser visto no Apêndice A – Esquema Elétrico do dispositivo.

# 4) Métodos

Na seção anterior foram apresentados e explicados os conceitos mais importantes para a execução do projeto. Feito isso, essa seção apresentará e explicará os métodos utilizados, o processo de pensamento e decisões tomadas durante a realização das diversas partes do trabalho.

## 4.1) Geração de som

Para simular diferentes posições de fonte sonora a princípio foram testados sinais compostos por ondas senoidais puras, nas quais foram aplicados os princípios de ITD e ILD. As senóides foram criadas nos dois canais do sistema *stereo* e, de acordo com o ângulo que se desejava emular, atrasos de tempo e diferenças nas amplitudes nos canais foram introduzidos e os resultados foram analisados de maneira qualitativa.

É possível calcular com certa facilidade o atraso de tempo necessário entre os canais para simular o correto ITD para diversos ângulos, e se encontram estudos que mostram quais os níveis de atenuação do sinal dos canais para as mesmas situações, porém após gerar as ondas com as características desejadas os resultados não foram satisfatórios. Foram testados diversos valores de atraso e atenuação, e apesar de algumas combinações apresentarem resultados um pouco melhores ainda não era possível identificar com clareza a posição da fonte sonora emulada.

Foram então utilizadas as HRIRs disponíveis na base de dados LISTEN, que apresentaram resultados melhores tornando a simulação das diferentes posições bastante eficiente.

A geração de som para ambos os canais do sistema *stereo* é feita configurando os pinos a eles associados como saídas PWM. O ciclo de trabalho do sinal pode variar entre 0 e 1, representando valores de 0 a 100%. Ao calibrá-lo em 100% se obtém o máximo nível de tensão possível na saída, e o mínimo é atingido ao se configurar o PWM para 0%.

Um sinal previamente capturado pode, portanto, ser reproduzido atualizando os valores de ciclo de trabalho de acordo com os valores amostrados da onda. Deve-se somente estar atento ao fato de que não é possível a reprodução de valores negativos, portanto qualquer sinal que se deseje reproduzir deve ser ajustado para que sua amplitude se encaixe entre os valores 0 e 1. As HRIR adquiridas anteriormente serão então reproduzidas utilizando esse método, mas

necessitam um pré-tratamento para que os valores das amostras se encaixem dentro dos limites do microcontrolador.

Esse pré-tratamento se deu início com a redução do número de amostras de todas as HRIR. Muitos dos 8192 pontos existentes em cada gravação possuem apenas ruído e não trazem nenhuma informação útil, com medidas de amplitude de valor desprezível quando comparadas com outros pontos. Essa quantidade de pontos foi então reduzida para os 512 primeiros pontos, onde visualmente foi verificado que é onde está concentrada a maior parte da informação útil da resposta. As formas de onda antes e depois desse processo, para um azimute de zero graus, canais esquerdo e direito sobrepostos, podem ser vistas nas figuras Figura 4.1 e Figura 4.2.

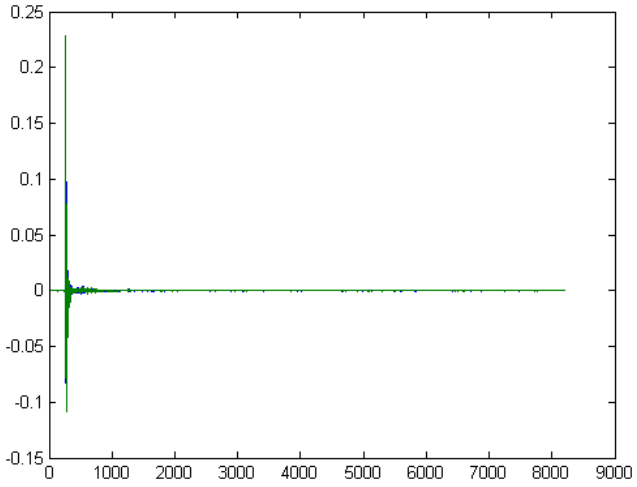



Figura 4.1 – HRIR de azimute 0 graus, canais sobrepostos, 8192 pontos

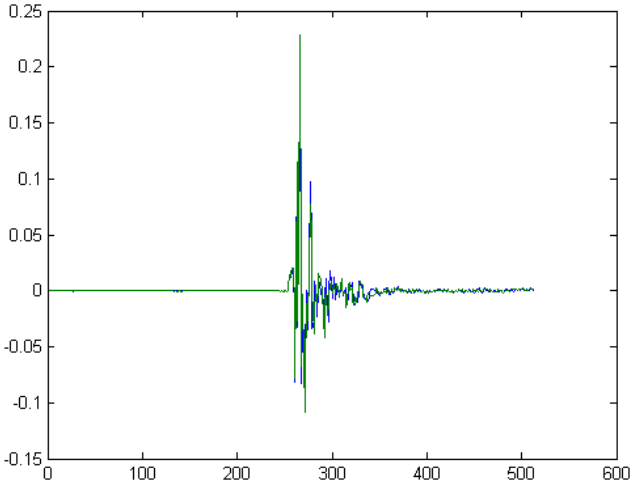



Figura 4.2 - HRIR de azimute 0 graus, canais sobrepostos, 512 pontos

Voltando aos limites do microcontrolador, é necessário que os valores que serão utilizados como ciclo de trabalho do PWM estejam entre zero e um. As figuras mostram que, como era de se esperar, as HRIR possuem valores negativos. Como não é possível a reprodução destes pelo PWM, é preciso realizar um deslocamento de zero, ou seja, somar uma constante à todas as amostras. Considerando um ciclo de trabalho de 50% como nível de zero, valores menores são considerados negativos, enquanto que os mais altos serão considerados positivos. Com isso, podem ser representados quaisquer sinais cujas amostras estejam compreendidas entre os valores ½ e -½. Para se certificar de que estejam dentro desse limite, para cada par (canais esquerdo e direito de um mesmo azimute e elevação) o valor máximo em módulo foi encontrado e todas as amostras foram divididas pelo seu dobro. O resultado é um sinal que está compreendido entre ½ e -½, como desejado.

O resultado desse processo no mesmo sinal apresentado na Figura 3.9 pode ser visto na Figura 4.3.

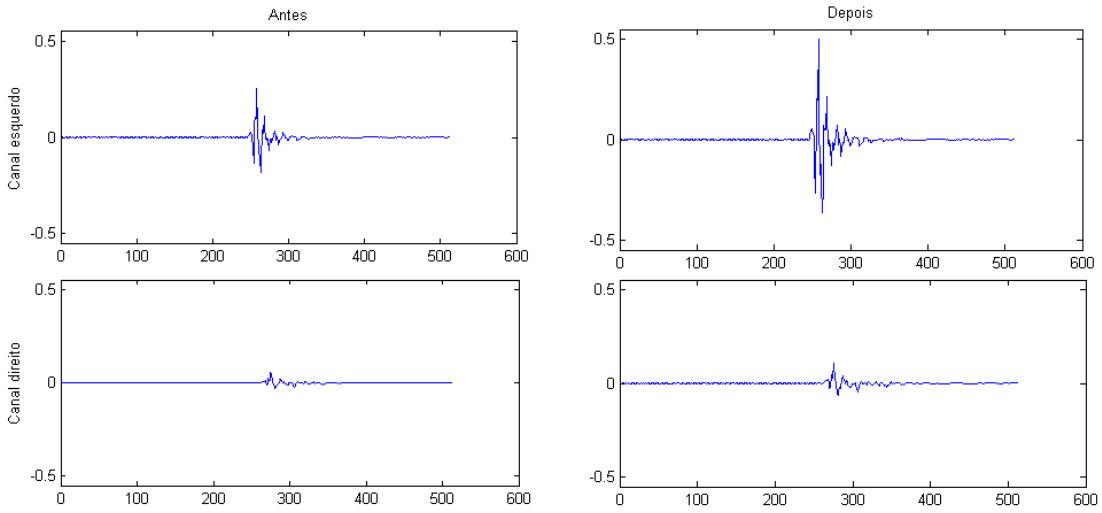



Figura 4.3 – Tratamento da amplitude de um sinal

O deslocamento de zero não é realizado nesse pré-tratamento pois a amplitude dos sinais ainda poderá ser modificada pelo microcontrolador de acordo com as necessidades do programa. Mantendo-se o zero original torna esse processo mais simples, portanto a constante de 0,5 somente é somada às amostras na fase final de atualização do ciclo de trabalho do PWM.

Finalmente, os canais serão filtrados via hardware para excluir as altas frequências do PWM e o resultado é a HRIR amostrada sendo reproduzida. A taxa de reprodução depende da frequência com que se atualiza o valor do ciclo de trabalho do PWM. Amostradas à 44100 Hz, a reprodução mais fiel dos sinais deve atualizar os valores aproximadamente a cada 22 microssegundos

## 4.2) Software

Nessa seção será descrito como o programa, desenvolvido em linguagem C++ utilizando o compilador on-line disponível no site do fabricante do microcontrolador (https://developer.mbed.org/), utiliza os sensores e realiza as funções previstas para o aparelho. Primeiro será dada uma visão geral do software, para depois entrar em mais detalhes quanto às funções essenciais para seu funcionamento.

O usuário, quando caminha com sua bengala, naturalmente a move de um lado para o outro para colher informações sobre os seus arredores. Enquanto faz esse movimento, os sensores HC-SR04 e a IMU constantemente colhem medidas de distância e ângulo de *yaw* da unidade de sensoriamento e utilizam esses dados para calcular, caso seja encontrado um obstáculo, qual sua posição em relação ao usuário, que, para os fins desse projeto, será definida pelo seu ângulo azimutal e sua distância em relação ao operador do aparelho, essa sendo denominada “módulo” para evitar confusão com a distância medida pelos sensores.

Obviamente existem infinitas possibilidades de azimute e módulo para qualquer obstáculo encontrado, portanto é interessante limitá-las a um número finito para que seja mais fácil de trabalhar. Com isso em mente, a área de 360 graus ao redor do usuário foi dividida e numerada em 24 seções de 15 graus cada, segundo a Tabela 4.1. Foi criada uma função que faz a conversão de qualquer ângulo entre +180 e -180 graus para os valores correspondentes indicados pela tabela, e o número da seção é encontrado simplesmente dividindo seu ângulo por 15.

Qualquer obstáculo terá seu azimute considerado como o mesmo da seção onde se encontra. Limitar a quantidade possível de ângulos azimutais em múltiplos de 15 graus é necessário pois esse dado é informado ao usuário através das HRIR, colhidas com resolução de 15 graus. Isso torna inviável utilizar uma resolução melhor, visto que seria preciso a gravação de novas HRIR experimentalmente para qualquer valor diferente dos já existentes no banco de dados.

Porém nem todas as 24 seções são pertinentes a todo momento. As que se localizam atrás do usuário em um dado momento, por exemplo, não são interessantes pois dificilmente serão atingidas pelos sensores. São selecionadas então dentre essas 24, onze cujas informações de azimute e módulo serão utilizadas, destacadas na Figura 4.4. Considerando a seção frontal do usuário, para onde ele está olhando e se deslocando, como zero graus, por exemplo, este receberá informações das seções correspondentes aos azimutes de -75 graus (75 graus à esquerda) até os 75 graus (75 graus à direita). Serão apresentados de maneira sequencial, começando da seção mais à esquerda e terminando mais à direita, para depois começar novamente. Para cada seção será reproduzida nos fones de ouvido sua HRIR correspondente com a amplitude modificada dependendo do módulo do obstáculo existente em tal setor.

| **Ângulo real (α) [graus]** | **Número da seção** | **Ângulo da seção [graus]** | **Ângulo real (α) [graus]** | **Número da seção** | **Ângulo da seção [graus]** |
|---|---|---|---|---|---|
| -7,5 < α < 7,5 | 0 | 0 | -180 < α < - 172,5; 180 < α < 172,5 | 12 | 180 |
| 7,5 < α < 22,5 | 1 | 15 | -157,5 < α < -172,5 | 13 | 195 |
| 22,5 < α < 37,5 | 2 | 30 | -142,5 < α < -157,5 | 14 | 210 |
| 37,5 < α < 52,5 | 3 | 45 | -127,5 < α < -142,5 | 15 | 225 |
| 52,5 < α < 67,5 | 4 | 60 | -112,5 < α < -127,5 | 16 | 240 |
| 67,5 < α < 82,5 | 5 | 75 | -97,5 < α < -112,5 | 17 | 255 |
| 82,5 < α < 97,5 | 6 | 90 | -82,5 < α < -97,5 | 18 | 270 |
| 97,5 < α < 112,5 | 7 | 105 | -67,5 < α < -82,5 | 19 | 285 |
| 112,5 < α < 127,5 | 8 | 120 | -52,5 < α < -67,5 | 20 | 300 |
| 127,5 < α < 142,5 | 9 | 135 | -37,5 < α < -52,5 | 21 | 315 |
| 142,5 < α < 157,5 | 10 | 150 | -22,5 < α < -37,5 | 22 | 330 |
| 157,5 < α < 172,5 | 11 | 165 | -7,5 < α < -22,5 | 23 | 345 |

Tabela 4.1 – Ângulos de *yaw* compreendidos por cada seção

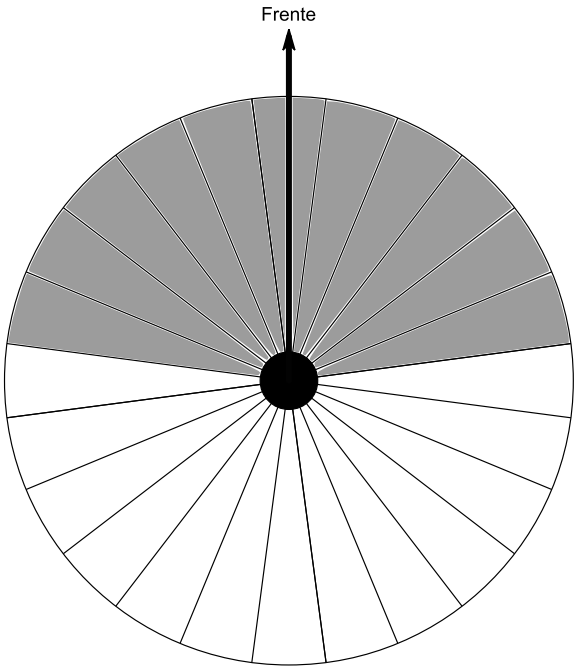



Figura 4.4 – Onze seções selecionadas para a reprodução de som

Os valores de módulo são transferidos ao usuário através da amplitude do sinal sonoro: objetos mais próximos produzem sons mais altos, enquanto que objetos mais distantes sons mais baixos. Essa grandeza é limitada pelo volume máximo que se pode gerar com o microcontrolador (e também um volume que não seja desconfortável aos ouvidos), e o mais baixo que ainda seja audível e capaz de transmitir informação de azimute ao usuário. Os diversos volumes possíveis para o som de saída foram então divididos em 4 faixas (Figura 4.5 e Tabela 4.2):

- Faixa sem medida: Quando a seção que está sendo avaliada não foi alcançada pelos sensores será tocada a HRIR com a amplitude mais baixa, para que o usuário saiba que para aquela região não foi coletada nenhuma informação.
- Faixa distante: Quando a seção em questão foi alcançada pelos sensores, mas ou não foi encontrado obstáculo ou o obstáculo está muito distante, além de um limite de módulo pré-estabelecido. Para o usuário não há diferença entre a presença de um obstáculo além desse limite ou um caminho livre, até que ele se aproxime mais. O volume dessa faixa é fixo e um pouco mais alto que o da faixa sem medida.
- Faixa próxima: Para objetos que estão muito próximos, aquém de um limite de módulo pré-estabelecido, o volume será máximo.

- Faixa ativa: Para objetos entre os limites da faixa distante e da faixa próxima, o volume irá variar linearmente com o módulo do obstáculo, tendo como volume máximo o mesmo da faixa próxima e valor mínimo o mesmo da faixa distante

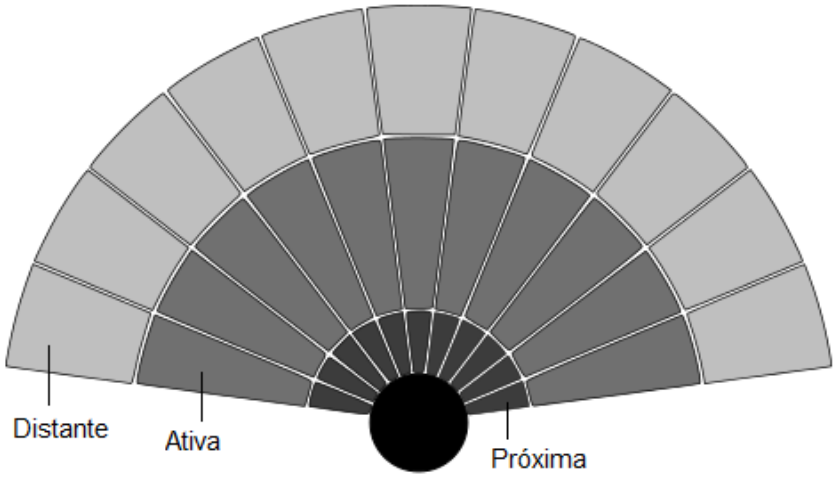



Figura 4.5 – Representação visual das faixas

| Faixa | Distância (d) | Multiplicador de volume (vol) |
|---|---|---|
| Não medida | - | 0,1 |
| Distante | d > 335 cm | 0,3 |
| Ativa | 135 cm < d < 335 cm | 0,3 < vol < 1,0 |
| Próxima | d < 135 cm | 1,0 |

Tabela 4.2 – Características das diferentes faixas

Após os testes realizados, que podem ser vistos com mais detalhes nas seções 5.3) e 5.4), o limite entre a faixa ativa e a faixa próxima foi definido 135 cm, e entre a faixa distante 335 cm. O primeiro representa uma faixa de 50 cm a partir da posição da unidade sensorial em relação ao usuário. O segundo representa 250 cm a partir desse mesmo ponto, distância máxima para qual o sensor foi considerado confiável baseado nos testes.

É importante destacar que as 24 seções azimutais são fixas, e não variam com a rotação do usuário. O que muda com esse fator são as onze seções cujas informações serão transmitidas via som. Essas são selecionadas utilizando os ângulos de *yaw* medidos pela IMU para calcular qual é o ângulo frontal, para onde o indivíduo está olhando, e usar essa informação para selecionar as seções corretas para a reprodução sonora, processo que será explicado com maior

detalhe posteriormente. As seções utilizadas mudam, portanto, com uma frequência relativamente alta, e são sempre relativas ao ângulo frontal, o que torna o aparelho bastante independente de outros sistemas de referência, geográficos por exemplo. Em suma, a seção zero, por exemplo, não precisa estar sempre posicionada no mesmo lugar geográfico, desde que não varie significantemente dentro do intervalo de tempo entre dois cálculos subsequentes de ângulo frontal, que deve levar aproximadamente um segundo, dependendo da velocidade de movimentação da bengala (ver seção 4.2.2.1).

### 4.2.1) Calibração da IMU

Antes de o código principal funcionar, é preciso fazer a calibração da IMU. Essa calibração é necessária pois o acelerômetro e giroscópio, mesmo quando estão em um estado estacionário, apresentam uma medida residual diferente de zero, resultado muitas vezes de stress mecânico sofrido pelo componente desviando o zero da escala. Para corrigir, basta ler as medidas feitas pelo dispositivo quando nesse estado estacionário (em uma posição perfeitamente horizontal para limitar a ação da gravidade à apenas um dos eixos) e então subtrair os valores encontrados de todas as demais medidas feitas pelos mesmos. Para melhores resultados é interessante ler um número elevado de medidas e utilizar a média aritmética de todas como fator de correção.

O código de calibração se encontra no Apêndice B – Código C++ de calibração da IMU

### 4.2.2) Código principal

No código principal são inicializadas todas as variáveis, sensores e funções pertinentes, bem como a criação de duas tarefas que realizam funções distintas: uma relacionada com a leitura dos sensores e a segunda com o processamento das coordenadas dos obstáculos, configuração e controle da geração de som.

O código completo se encontra no Apêndice C – Código C++ Principal

#### 4.2.2.1) Tarefa 1

A primeira função dessa tarefa é realizar a leitura dos valores de aceleração e velocidade angular dos três eixos da IMU, compensando os erros de leitura com os valores da calibração realizada anteriormente. A leitura de cada eixo é formada por dois bytes, lidos separadamente,

portanto é preciso primeiramente juntá-los em uma única variável. Esses valores necessitam ainda ser convertidos para seus valores reais físicos, multiplicando o número contido nessa variável pela sensibilidade do sensor.

Os valores finais são então atualizados no filtro IMU que retorna o ângulo de *yaw* da unidade de sensoriamento, número que será utilizado em conjunto com as leituras dos sensores de distância para encontrar as coordenadas (par azimute, módulo) dos obstáculos detectados.

O ângulo de *yaw* também é utilizado para calcular o ângulo frontal do usuário. Considerando que a bengala estará em constante movimento ora para um lado, ora para outro, têm-se sempre dois extremos desse movimento: um extremo à esquerda e um à direita, sempre quando o sentido da movimentação se altera. É plausível assumir que esse movimento é bastante simétrico e o ângulo central é, portanto, a média aritmética desses dois extremos. O sentido da movimentação é encontrado observando a variação do ângulo de *yaw* entre duas medidas consecutivas. Variações positivas indicam movimento para a direita enquanto que negativas para a esquerda. Seguindo essa linha de pensamento, os extremos são encontrados cada vez que o sinal dessa variação se altera, e é assim que eles são detectados pelo programa.

Por precaução, cada vez que essa mudança de sinal ocorre é iniciada uma função do tipo *Timeout* que, se após um 1,5s não seja encontrado um novo extremo, ela encerra a espera e o ângulo de *yaw* atual é considerado extremo. Com isso o ângulo frontal é calculado mesmo que a bengala esteja parada.

É preciso se atentar a um caso especial, caso os ângulos de *yaw* estejam próximos à ± 180 graus. Isso pode causar que a variação de *yaw* calculada seja muito alta, sendo que na verdade os extremos reais estão próximos um do outro. Para evitar esse erro, a variação é sempre monitorada e, caso seja maior que 180 graus, medidas são tomadas para corrigir a leitura. Caso o usuário movimente sua bengala em um arco maior que 180 graus haverá erro na medida de um ângulo frontal, porém é esperado que em condições normais de uso isso não ocorra. Contanto que essa movimentação anormal não seja constante, medidas subsequentes não serão afetadas pela falha.

Nessa tarefa é também chamada uma rotina de leitura dos sensores de distância sendo que a cada vez que é executada, apenas um dos sensores é lido. Desta maneira, a cada 3 execuções da tarefa todos os sensores terão sido utilizados. Para iniciar a leitura, durante a rotina é enviado um pulso alto de 10 microssegundos no pino *Trigger* do sensor em questão, e em seguida o programa entra em um estado de espera enquanto aguarda a manifestação do pino *Echo*.

O pino *Echo* de todos os sensores está associado com duas rotinas de interrupção, uma na borda de subida e uma na borda de descida. Na borda de subida, é iniciado um contador de tempo, responsável pela medição da largura desse pulso alto no pino. Associada à borda de descida está uma rotina que para esse contador e sinaliza que a medida está completa. Ao receber esse sinal, o programa sai de seu estado de espera e faz a leitura do contador, calculando e retornando a distância encontrada.

Após o pulso no pino trigger, antes de entrar em estado de espera, também é iniciada uma função do tipo *Timeout* que após 25 milissegundos (tempo suficiente para uma medida de mais de 4 metros) irá encerrar a espera e dar sequência ao programa. Essa é uma medida de segurança que impede que o programa fique preso caso não seja encontrado nenhum obstáculo ou em caso de mal funcionamento de algum sensor.

Devido ao ângulo físico dos sensores em relação ao eixo da bengala e a distância entre a unidade de sensoriamento e o corpo do usuário a distância medida pelo sensor não corresponde ao módulo do obstáculo, e o ângulo de *yaw* não corresponde ao azimute. Portanto é necessário o cálculo dessas grandezas através do uso de trigonometria. A Figura 4.6 mostra os ângulos e medidas existentes, e o azimute e módulo são dados pelas equações a seguir. Vale notar que este azimute encontrado não tem como referência a frente do usuário, mas sim o ângulo de *yaw* zero da unidade de sensoriamento.

$$M = \sqrt{[L + d\cos(alpha)]^2 + [d\sin(alpha)]^2}$$

$$Az = Y + \sin^{-1}[d\sin(alpha)]$$

$M =$ Módulo calculado do obstáculo

$Az =$ Azimute calculado do obstáculo

$L =$ Distância entre a unidade de sensoriamento e o usuário

$d =$ Distância medida pelo sensor

$alpha =$ Ângulo físico entre o sensor e o eixo da bengala (± 90 para os laterais, zero para o frontal

$Y =$ Ângulo de yaw medido da IMU

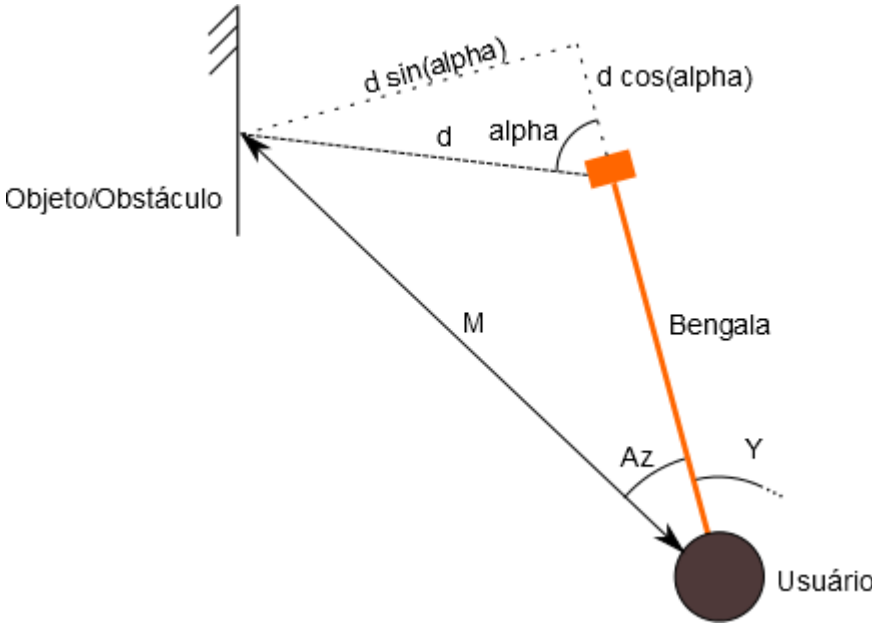



Figura 4.6 – Cálculo de azimute e módulo de obstáculos

O armazenamento das informações é finalmente feito em uma matriz registro bidimensional de duas linhas e 24 colunas, onde cada uma das colunas corresponde a uma das seções azimutais previamente determinadas. Para cada coluna, as duas linhas contêm o módulo do obstáculo encontrado naquela seção e um número, um ou zero, que indica se aquela seção foi ou não medida recentemente, informação importante para caso seja selecionada para reprodução sonora uma seção que não foi atingida recentemente pelos sensores, identifique-se que suas informações devem ser encaixadas na faixa não medida.

Quando calculados o azimute e módulo de um obstáculo, o primeiro deve ser convertido para um dos ângulos pré-estabelecidos dos setores, utilizando a Tabela 4.1, e então dividido por 15. O valor resultante dessa operação resultará em um índice entre zero e 24 que é utilizado para indexar uma coluna da matriz registro. Nessa coluna então é armazenado o valor encontrado do módulo em uma linha, e o valor 1 para sinalizar a ocorrência da medida na outra.

Ao fim da tarefa, deve-se estabelecer o tempo de intervalo até que esta seja executada novamente. Como os cálculos realizados pelo filtro IMU dependem da frequência com que são feitos, é importante que essa execução ocorra em intervalos constantes. No entanto, ela não possui um tempo de execução constante devido à rotina de leitura dos sensores de distância, que pode variar de alguns microssegundos até 25 milissegundos caso não seja encontrado nenhum obstáculo. Medições utilizando os próprios contadores do microcontrolador mostraram que a tarefa chega a levar cerca de 30 milissegundos para se completar. Portanto foi utilizado

um contador para realizar essa medição de tempo toda vez que a tarefa é executada, e o tempo de intervalo entre duas execuções consecutivas é alterado baseado nessa medida para que o tempo entre os cálculos do filtro IMU sejam constantes e iguais a 40 milissegundos.

Vale notar que os sensores de distância possuem uma recomendação de não ser reutilizados antes de 50 milissegundos após o término de sua última medida. Como existem três, o tempo entre duas medidas consecutivas de um mesmo sensor nessa configuração é de 80 ms, o que respeita essa condição.

#### 4.2.2.2) Tarefa 2

Essa tarefa é responsável por coletar as coordenadas e dados colhidos pela Tarefa 1, utilizá-las para selecionar as seções corretas para a reprodução do som, e por fim configurar e reproduzir as informações nelas contidas por meio do fone de ouvido *stereo*.

Primeiramente, as informações contidas na matriz registro são copiadas para uma matriz registro definitiva, que permanecerá imutada enquanto seus dados são utilizados na reprodução do som. A primeira então é completamente limpa para que a Tarefa 1 possa preenchê-la com novas medidas.

O ângulo frontal já calculado é convertido para o índice de seu setor correspondente, através do método da Tabela 4.1 e da divisão por 15, como já feito anteriormente. Obtido esse valor, as cinco seções imediatamente adjacentes à direita e as cinco à esquerda são as que irão ter seus dados utilizados para a reprodução sonora. Isso resulta nas onze seções necessárias (cinco à esquerda, cinco à direita e a frontal). Como já mencionado, a reprodução tem início com a seção mais à esquerda e termina com a seção mais à direita.

A cada seção a ser reproduzida a HRIR correspondente ao seu ângulo azimutal em relação à central é selecionada, e sua amplitude modulada segundo a faixa que o módulo contido em sua coluna da matriz registro permanente indica. Os valores de azimute utilizados para cada seção na ordem que são reproduzidas se encontram na Tabela 4.3.

| Ordem de execução | Azimute da HRIR |
|---|---|
| 1 | 75 graus à esquerda |
| 2 | 60 graus à esquerda |
| 3 | 45 graus à esquerda |
| 4 | 30 graus à esquerda |
| 5 | 15 graus à esquerda |
| 6 | 0 graus |
| 7 | 15 graus à direita |
| 8 | 30 graus à direita |
| 9 | 45 graus à direita |
| 10 | 60 graus à direita |
| 11 | 75 graus à direita |

Tabela 4.3 – Setores reproduzidos e seus ângulos de azimute

Cada seção é reproduzida por um tempo pré-determinado, e a tarefa se encerra após a reprodução de todas as onze. Foi escolhido um valor de 100 milissegundos por seção, resultando em 1,1 segundos no total. Esse valor é baseado em preferência pessoal e pode ser alterado sem problemas caso o usuário do aparelho deseje ou sinta necessidade. Após o encerramento, é aguardado um intervalo de 20 milissegundos e a tarefa se repete, utilizando dados de uma nova matriz registro que foi preenchida pela Tarefa 1 enquanto a reprodução era realizada. Vale destacar que a realização de medidas é completamente independente, ocorrendo concomitantemente à execução dessa tarefa.

A reprodução de som é realizada por uma rotina de interrupção do tipo *Ticker*, que é chamada em intervalos de tempo dependentes da taxa de reprodução configurada (22 microssegundos para 44100 Hz). Essa rotina não realiza nenhuma operação se não habilitada por um sinalizador, ação de responsabilidade da Tarefa 2, quando se deseja que som seja reproduzido nos canais *stereo*. Quando habilitada, ela atualiza repetidamente os valores do ciclo de trabalho das saídas PWM de acordo com os valores amostrados da HRIR e variações de volume configuradas previamente na tarefa.

# 5) Resultados e Discussões

## 5.1) Cabo

Para justificar a utilização do cabo de 8 vias que conecta as duas unidades principais do dispositivo, e verificar que a transmissão de informações não é comprometida, foi utilizado um osciloscópio para medir as informações que nele circulam e verificar a qualidade destas. Essas medidas foram então analisadas e comparadas com outras feitas em um cabo significantemente mais curto, utilizado como base de comparação para os resultados. O primeiro cabo, longo, possui comprimento de 1,5m, enquanto que o segundo, curto, 13cm.

Para cada cabo, os dois canais do osciloscópio representam as medidas feitas na extremidade conectada à unidade de processamento (medida superior no osciloscópio) e a extremidade conectada à unidade de sensoriamento (medida inferior).

### 5.1.1) Sinal de trigger do sensor de distância

O pulso de aproximadamente 10 µs que comanda que o sensor inicie a medição foi monitorado para verificar o desempenho dos cabos na transmissão dessa informação. Os valores mais importantes que se deve observar nesse teste são o nível de tensão do pulso e seu tempo de duração.

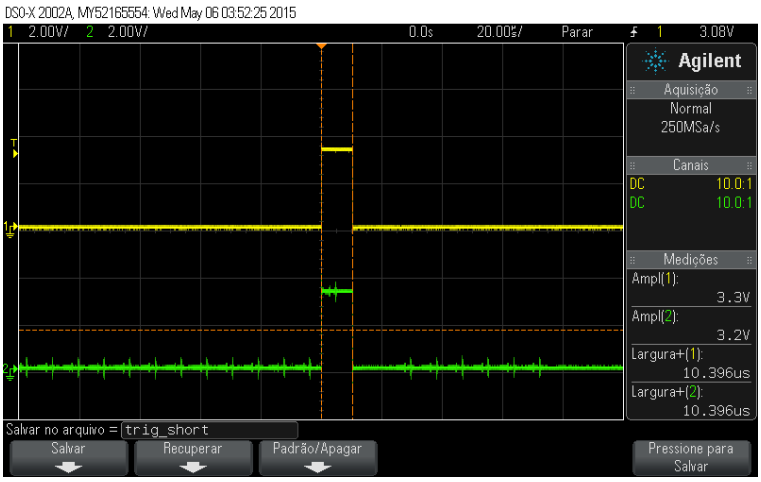



Figura 5.1 – Pulso de disparo, cabo curto

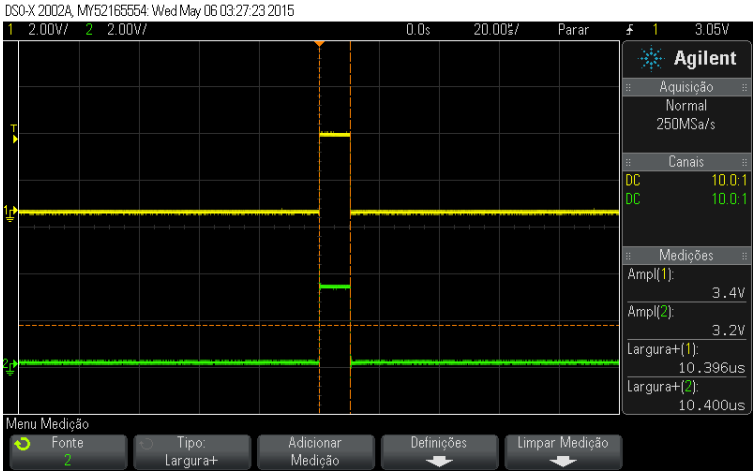



Figura 5.2 – Pulso de disparo, cabo longo

Pode-se ver nas figuras Figura 5.1 e Figura 5.2 que a diferença na medida das duas extremidades do cabo é desprezível tanto para o cabo longo quanto para o cabo curto. A amplitude e a largura de pulso pouco mudaram, bem como seu tempo de subida e descida.

### 5.1.2) Sinal no pino *Echo* do sensor de distância

Nesse teste o mais importante é a largura do pulso, que é utilizada no cálculo da distância do obstáculo.

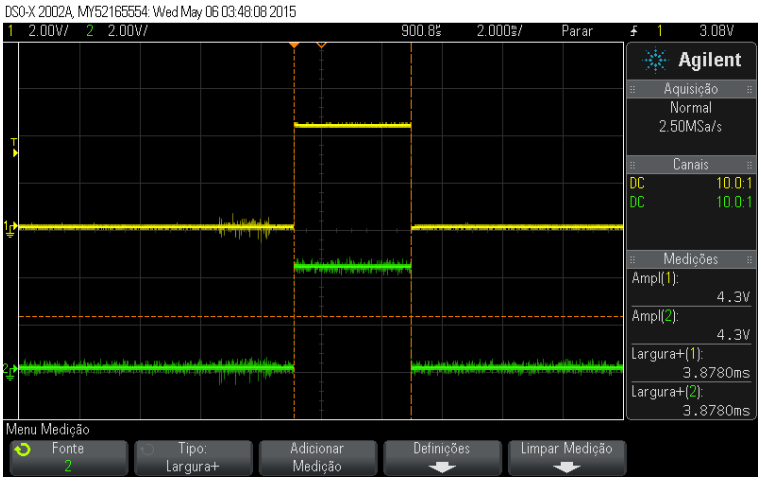



Figura 5.3 – Sinal no pino *Echo*, cabo curto

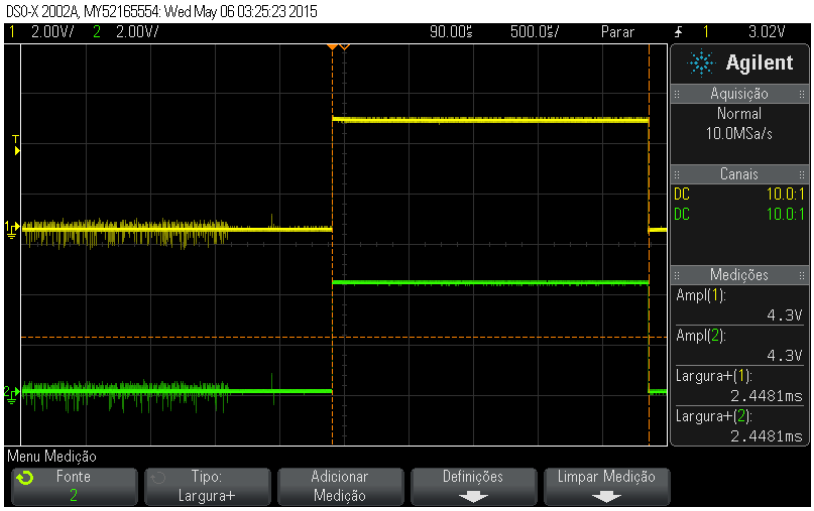



Figura 5.4 – Sinal no pino *Echo*, cabo longo

Apesar da grande diferença entre as larguras dos dois cabos que se observa nas figuras Figura 5.3 e Figura 5.4, isso se dá ao fato de os sensores estarem medindo distâncias diferentes em cada ocasião. Pode-se observar, porém, que em ambos os casos a largura do pulso nas duas extremidades é exatamente o mesmo, mostrando qualidade na transmissão da informação.

### 5.1.3) Sinal SCL

Nesse teste, o pino SCL, que dá o sinal de *clock* para a comunicação I2C, foi monitorado. As informações que se deve observar são a largura de pulso, os níveis de tensão do sinal e o tempo de subida e descida.

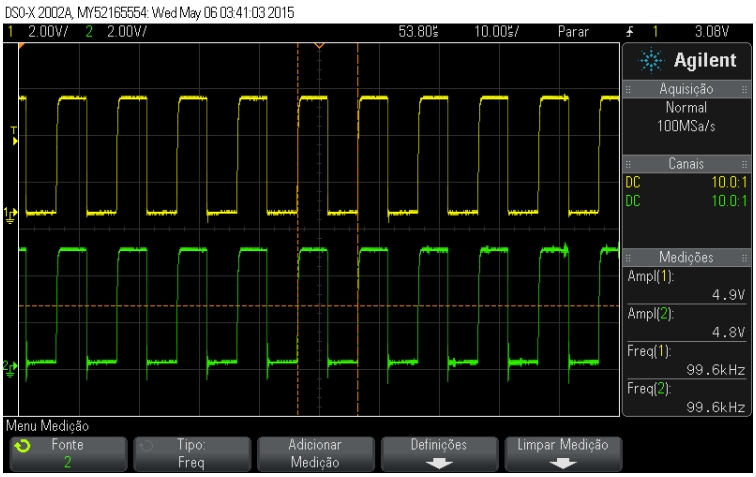



Figura 5.5 – Sinal de *clock* do barramento I2C, cabo curto

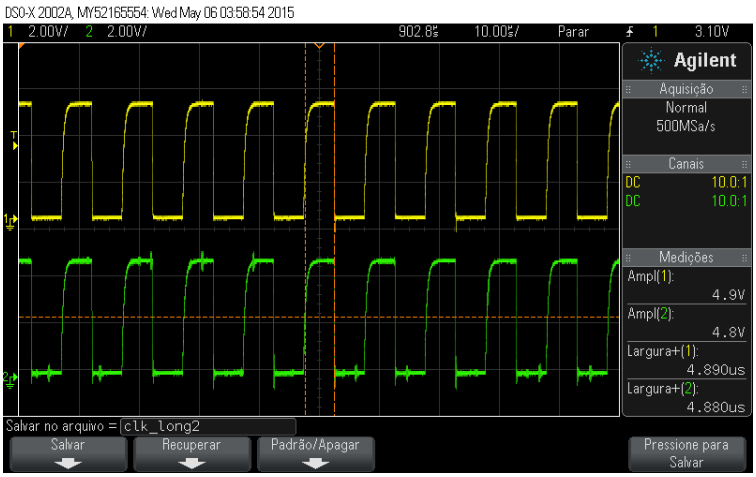



Figura 5.6 – Sinal de *clock* do barramento I2C, cabo longo

Como mostram as figuras Figura 5.5 e Figura 5.6, os valores de largura de pulso e valores máximos e mínimos do sinal são bastante constantes nos dois casos. Porém é bastante perceptível a diferença entre o tempo de subida nos dois casos. O cabo curto apresenta uma forma de onda mais quadrada, com uma subida mais abrupta do nível de tensão, enquanto que no cabo longo, como era de se esperar, essa subida é mais amortecida. Deve-se lembrar, porém, que a comunicação I2C considera qualquer valor de tensão acima de 70% do valor de alimentação como nível lógico alto (3,4 V para o valor máximo medido de 4,9V). Portanto, apesar de demorar mais para atingir o valor máximo, a diferença para atingir o patamar de nível alto não é muito expressiva, sendo de apenas 0,83us, o que corresponde à 8,3% do ciclo de *clock.* A Figura 5.7 mostra o patamar citado e o tempo de subida.

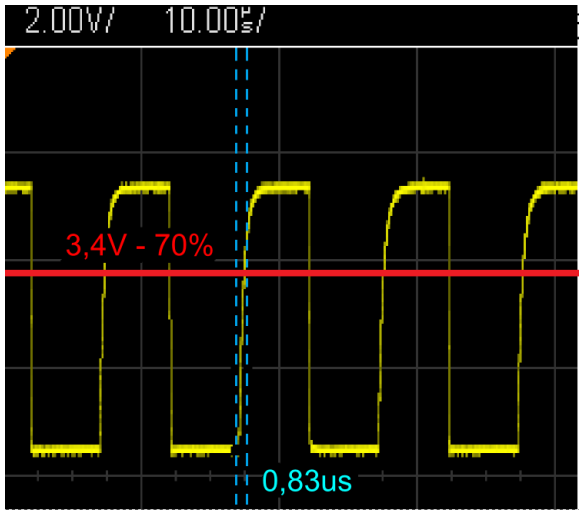



Figura 5.7 – Tempo de subida do sinal de *clock*

### 5.1.4) Sinal SDA

Para esse teste, o importante é a estabilidade dos níveis de tensão durante a transmissão, lembrando que para ser considerado válido o nível de tensão deve ser constante durante um período alto do *clock*.

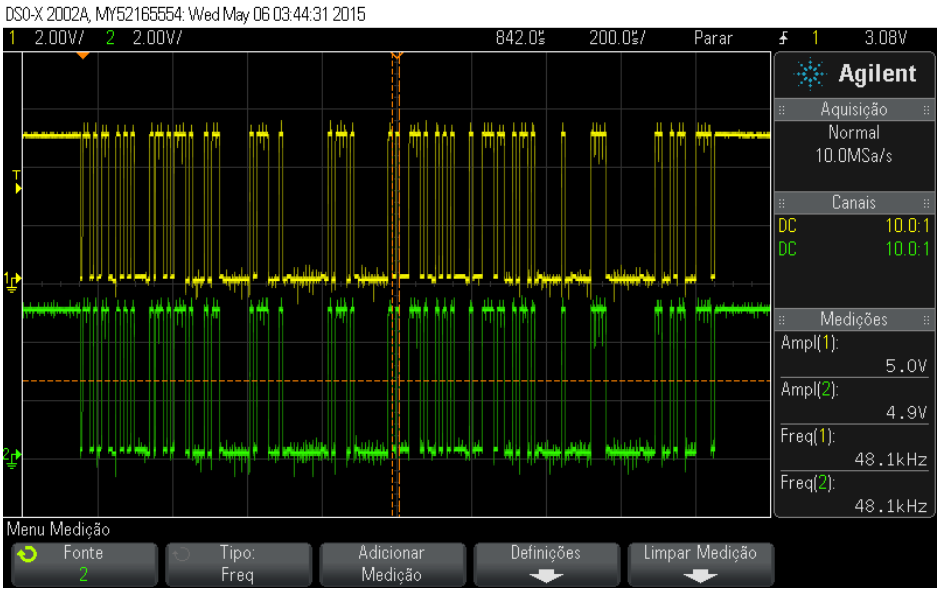



Figura 5.8 – Transmissão de dados, cabo curto

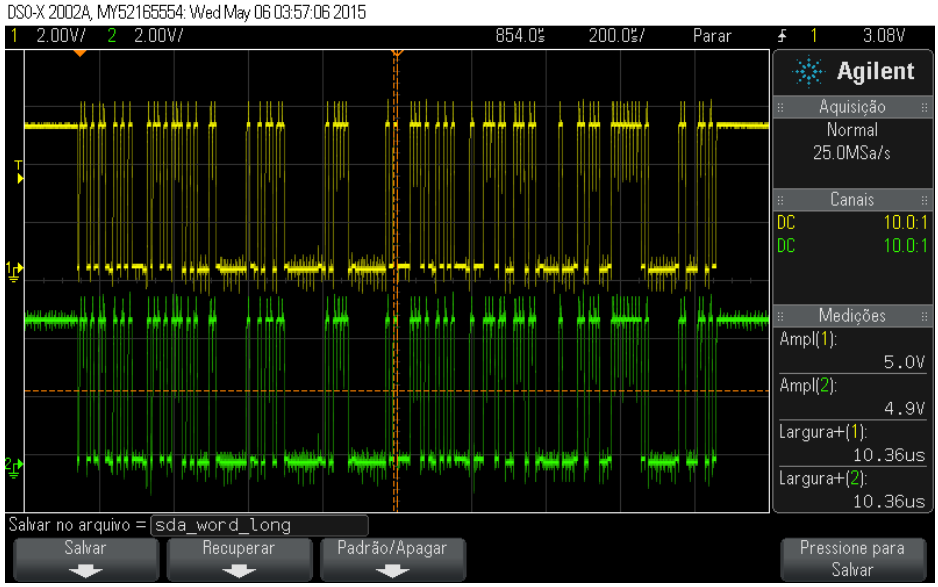



Figura 5.9 – Transmissão de dados, cabo longo

Nas figuras Figura 5.8 e Figura 5.9 mais uma vez é notável a diferença entre os cabos curto e longo. O sinal do cabo longo apresenta diversas quedas no nível de tensão e picos durante seu período em nível alto, enquanto que as mesmas características no cabo curto são mais atenuadas. A impressão geral que se têm é que o cabo longo é extremamente ruidoso e os dados transmitidos são comprometidos. Foi feita então uma análise mais detalhada com a Figura 5.10.

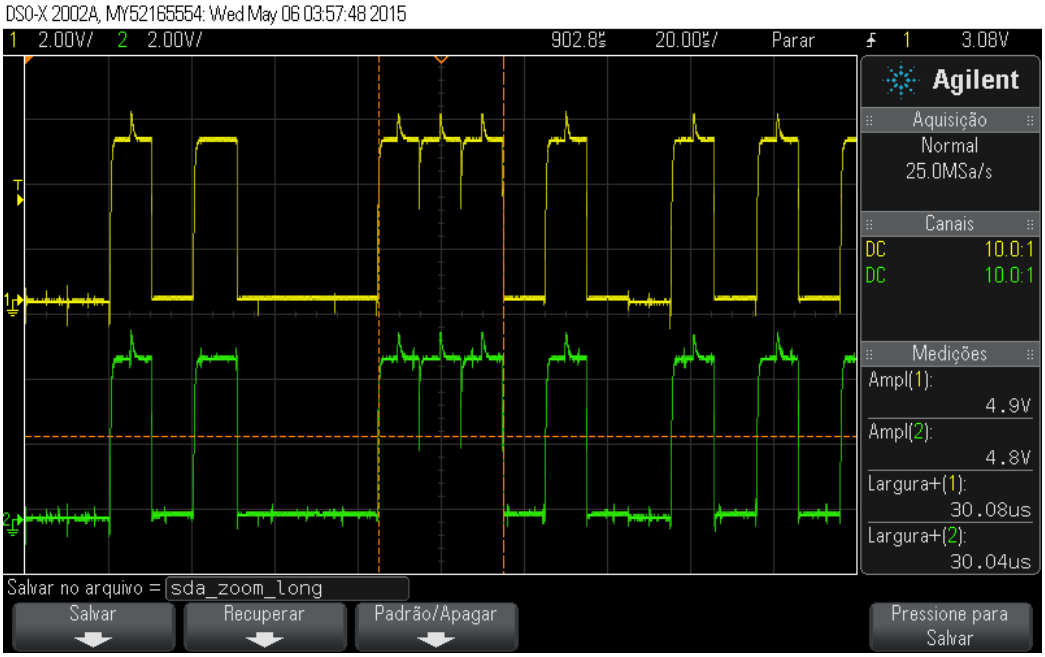



Figura 5.10 – Detalhe, transmissão de dados, cabo longo

As quedas de tensão são bastante acentuadas, inclusive caindo além do limite de 70% para ser consideradas em nível alto. Porém deve-se observar o momento em que elas ocorrem. Analisando o período central em destaque vemos que ele tem uma largura de aproximadamente 30 microssegundos. Para uma transmissão à 100 kHz, como é o caso, isso representa a transmissão de três bits. Logo, estão associados a esses três bits três pulsos de *clock*, que nessa frequência também tem duração de aproximadamente 10 microssegundos, cinco em nível alto e cinco em nível baixo.

Considerando o que se sabe sobre a comunicação I2C, é correto assumir que durante a borda de subida do primeiro bit o nível da linha SCL se encontra em nível baixo. Se tomarmos ainda essa borda como referência de tempo t = 0, vemos que as quedas de tensão observadas

ocorrem nos instantes t = 10 µs e t = 20 µs, um e dois períodos de *clock* após a primeira borda. Portanto, essas quedas de tensão ocorrem justamente em períodos de nível baixo do *clock* (Figura 5.11), não comprometendo a informação.

Quanto aos picos, estes sim ocorrem durante o período de nível alto do clock, porém não alteram o nível logico associado ao sinal, não comprometendo o dado.

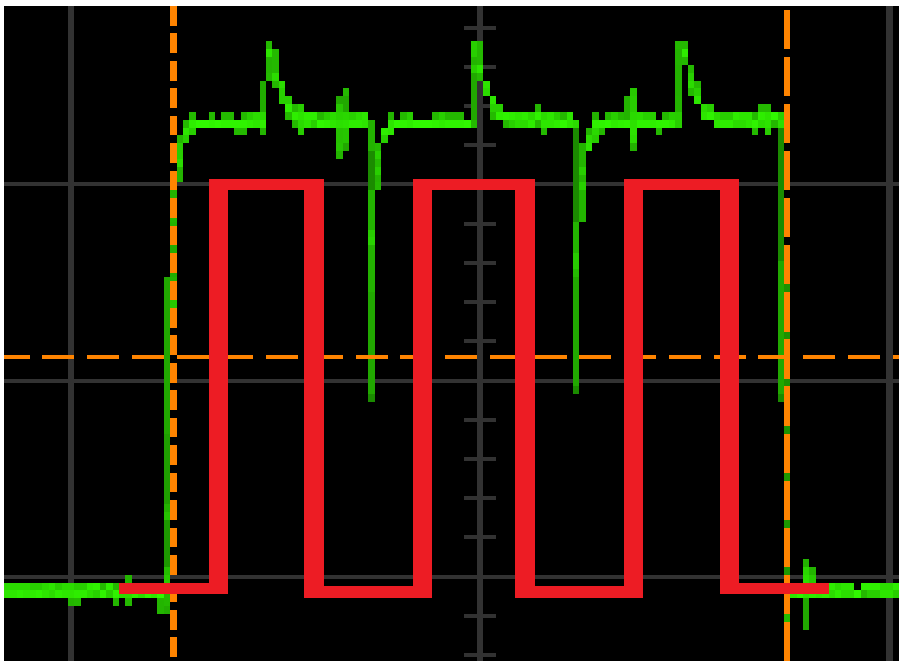


Figura 5.11 – Detalhe, 3 bits com *clock* sobreposto

Após estes testes, percebe-se que para a transmissão dos dados relacionado aos sensores de distância é boa tanto para o cabo longo quanto para o curto. Já os dados da comunicação I2C apresentam diferenças bastante significativas entre os dois, sendo notável a melhor qualidade com um cabo mais curto. Apesar disso, têm-se que as distorções causadas pelo cabo mais longo não afetam os dados transmitidos e, apesar de não ser ideal, este cabo se mostrou capaz de realizar as tarefas a ele designadas.

## 5.2) Medições de ângulo

Para validar as medidas de ângulo de *yaw* utilizadas durante o funcionamento do dispositivo foi realizado um teste onde a unidade de sensoriamento foi posicionada sobre uma folha de papel com marcações reais de ângulos, em incrementos de 15 graus, e estes foram medidos pelo dispositivo. Para cada ângulo foram utilizadas 100 medidas e calculados o valor médio e também o desvio padrão. Os resultados se encontram na Tabela 5.1.

Durante o teste foi percebido que o ângulo de *yaw* medido sofre um pequeno desvio com o tempo, da ordem de -0,075 graus por segundo. Como o teste levou alguns minutos para ser concluído esse desvio influenciou nas medidas. Para contornar esse problema as medidas foram cronometradas e foi feita uma compensação nos valores adquiridos baseado no tempo transcorrido durante a realização do teste.

| **Ângulo Real [graus]** | **Ângulo medido [graus]** | **Desvio Padrão** | **Tempo da medida [s]** | **Desvio devido ao tempo [graus]** | **Medida corrigida [graus]** | **Erro [%]** |
|---|---|---|---|---|---|---|
| 0 | -0,22 | 0,127 | 0 | 0 | -0,22 | - |
| 15 | 15,16 | 0,129 | 14,7 | -1,1025 | 16,26 | 8,4 |
| 30 | 29,24 | 0,126 | 33,7 | -2,5275 | 31,77 | 5,4 |
| 45 | 39,46 | 0,130 | 60,1 | -4,5075 | 43,97 | 2,3 |
| 60 | 52,33 | 0,146 | 87,7 | -6,5775 | 58,88 | 1,9 |
| 75 | 62,21 | 0,147 | 107,1 | -8,0325 | 70,24 | 6,3 |
| 90 | 73,5 | 0,152 | 126,2 | -9,4650 | 83,00 | 7,7 |

Tabela 5.1 – Resultados do teste de medida angular

Analisando os resultados, vemos que as medidas apresentam uma certa inexatidão, com um erro médio de 5,43%, sendo o maior deles 8,4% na medida de 15 graus. Esses valores não são muito expressivos, principalmente se levado em consideração que a resolução das seções azimutais para este projeto é de 15 graus, e um erro de algumas unidades não afeta significantemente o resultado. A natureza do trabalho não exige um alto grau de exatidão, porém esse é um aspecto que poderia ser melhorado para aumentar qualidade das medidas buscando um procedimento de cálculo de ângulos de *yaw* com a utilização do magnetômetro.

Quanto ao desvio no tempo, bem como um possível acúmulo dos erros comentados anteriormente, não interferem no desempenho do aparelho. Como já mencionado nesse trabalho, os ângulos não possuem uma referência fixa a ser respeitada, sendo referenciado apenas ao ângulo frontal que é recalculado frequentemente. Portanto os ângulos podem se alterar livremente desde que não se alterem significantemente durante um período de cálculo de ângulo frontal, feito ao se mover a bengala de um lado ao outro. Esse movimento leva um segundo ou menos sendo, portanto, desprezível o desvio nesse tempo.

## 5.3) Medições de distância

Para a realização de dois testes, a unidade sensorial foi fixada à uma altura de 25 centímetros do chão, em uma garagem vazia. No primeiro teste, uma caixa de papelão de 89 cm de altura e 51 cm de largura foi posicionada em frente ao sensor, inicialmente à 30cm de distância, e então afastada gradualmente. As medidas do sensor em diferentes momentos foram anotadas e se encontram na Tabela 5.2. Assim como no teste de medida angular, 100 medidas foram utilizadas e os respectivos valores médios e desvios padrões foram calculados.

No segundo teste, a caixa foi substituída por um galão de água de 20 litros, com diâmetro da base de 28 cm e altura do corpo 36 cm. O mesmo procedimento do primeiro teste foi executado, e os resultados se encontram na mesma tabela. Um desenho mostrando a montagem do teste pode ser visto na Figura 5.12

| | **Caixa** | | | **Galão** | | |
|---|---|---|---|---|---|---|
| **Distância real [cm]** | **Distância medida [cm]** | **Desvio padrão [cm]** | **Erro [%]** | **Distância medida [cm]** | **Desvio padrão [cm]** | **Erro [%]** |
| 30 | 30,65 | 0,059 | 2,1 | 30,63 | 0,245 | 2,1 |
| 60 | 61,48 | 0,235 | 2,4 | 61,65 | 0,439 | 2,7 |
| 100 | 100,02 | 0,322 | 0,0 | 102,62 | 0,893 | 2,6 |
| 150 | 149,87 | 0,555 | 0,0 | 148,47 | 1,011 | 0,8 |
| 200 | 199,98 | 0,480 | 0,0 | 198,94 | 0,896 | 0,5 |
| 250 | 254,33 | 1,091 | 2,1 | 253,41 | 1,399 | 1,3 |
| 300 | 300,21 | 0,585 | 0,0 | 411,99 | 144,270 | 37,33 |
| 340 | 342,52 | 1,037 | 0,7 | - | - | - |

Tabela 5.2 – Resultados dos testes de medição de distância

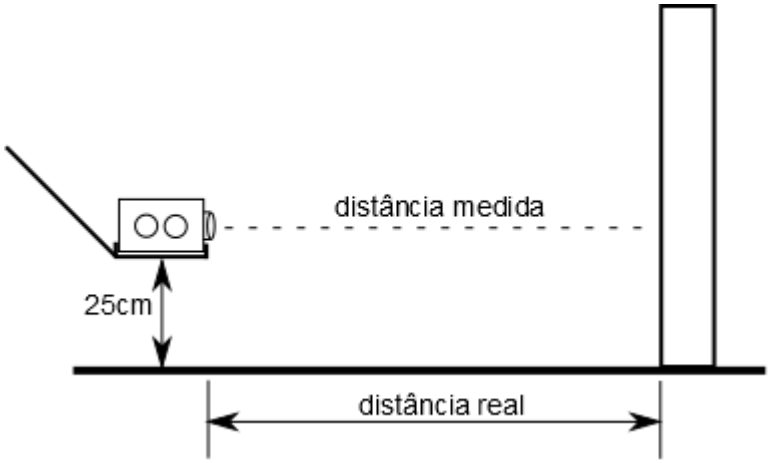



Figura 5.12 – Teste de medição de distância

Os resultados se mostraram bastante exatos em ambos os testes. Com a caixa, devido às suas dimensões maiores, as medidas foram precisas em toda a faixa analisada, e os baixos valores calculados de desvio padrão indicam uma boa consistência nas 100 medições realizadas em cada caso. Com o galão os resultados foram bastante satisfatórios dentro da faixa de 250 cm, porém a medida de distância 300 cm se mostrou bastante errada. Além do valor médio calculado ser bem divergente do valor real, o desvio padrão possui uma magnitude preocupante. Devido às dimensões reduzidas do galão, quando muito distante o sensor encontra dificuldade em detectá-lo. Logo dentre as 100 medições realizadas algumas apresentam valores corretos de distância, mas muitas não apresentam valor algum pois não há sinal de retorno refletido pelo objeto. Isso resulta no valor médio e desvio padrão elevados que estão representados na tabela.

A principal utilidade desse teste, além é claro de verificar que os sensores estão apresentando medidas coerentes, é analisar seu funcionamento e assim definir o limite inferior da faixa distante, explicada na seção 4.2.2.1). Para assegurar a qualidade das medidas e estabelecer esse limite o sensor foi considerado confiável para medidas de até 250 cm.

## 5.4) Teste geral de funcionamento

Após testar os componentes individuais do projeto, esse teste geral analisa o funcionamento do dispositivo como um todo. Na mesma garagem utilizada para o teste de medições de distância foram posicionadas a mesma caixa já mencionada, e uma segunda com 74 cm de altura e 39 cm de largura. Serão chamadas de caixa 1 e caixa 2, respectivamente, para facilitar suas identificações.

Como mostra a Figura 5.14, a garagem possui 2,3m de largura e 2,8m de comprimento. Ao fundo, porém, não há uma parede, mas sim um degrau de 25cm de altura. A caixa 1 foi posicionada junto à parede esquerda, à uma distância de 2m da frente da garagem, e a caixa dois à uma distância de 1,8m junto à parede da direita.

A unidade de sensoriamento foi posicionada na extremidade de um *monopod*, utilizado para simular uma bengala real. O *monopod* foi estendido de modo que a unidade se mantivesse a uma altura aproximada de 25 cm quando sendo manuseado por um usuário. Vale notar que desta maneira foi constatado que os sensores se encontram a uma distância de aproximadamente 85cm do manuseador durante a operação. Em seguida o instrumento foi movimentado lateralmente diversas vezes, simulando a utilização da bengala por um deficiente visual, e os resultados dessa simulação foram coletados e analisados, e podem ser vistos na Figura 5.13.

```
playing: 17              distancia: 0.00          vol: 0.10
front: 10
playing: 5               distancia: 110.00              vol: 1.00
playing: 6               distancia: 120.00              vol: 1.00
playing: 7               distancia: 140.00              vol: 0.98
playing: 8               distancia: 185.00              vol: 0.82
playing: 9               distancia: 215.00              vol: 0.72
playing: 10              distancia: 275.00              vol: 0.51
playing: 11              distancia: 275.00              vol: 0.51
playing: 12              distancia: 165.00              vol: 0.89
playing: 13              distancia: 165.00              vol: 0.89
playing: 14              distancia: 130.00              vol: 1.00
playing: 15              distancia: 115.00              vol: 1.00
front: 11
```

Figura 5.13 – Resultados do teste geral

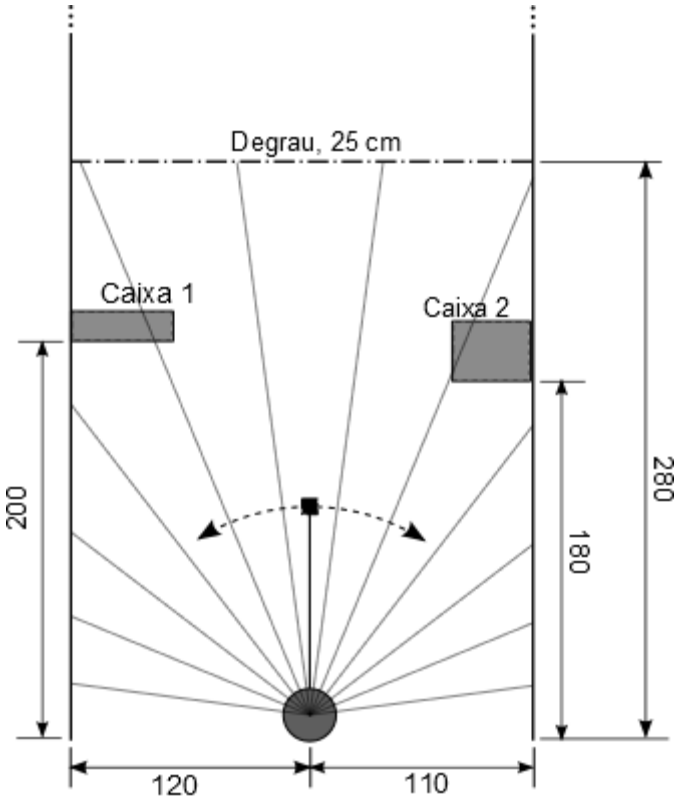



Figura 5.14 – *Layout* do teste geral, com seções azimutais sobrepostas

A primeira informação que se tem é o dado indicado por *front*, na primeira linha da figura. É o ângulo frontal calculado representado pelo número de sua seção azimutal correspondente, no caso seção 10 nessa simulação. Em seguida se têm 11 linhas que indicam cada uma das seções azimutais que foram reproduzidas nos fones de ouvido, juntamente com a distância nelas armazenadas e o multiplicador de volume correspondente. Nota-se que medidas de distância inferiores a 135 cm resultam em um multiplicador de volume igual à um, enquanto que medidas dentro da faixa ativa resultam em diversos valores diferentes, sendo os maiores correspondentes a distâncias mais próximas e os menores a distâncias maiores. Nenhuma das medidas foi posicionada dentro da faixa distante.

Retira-se também da imagem que nenhuma das 11 seções apresentou uma não-medida, ou seja, todas foram atingidas com sucesso pelo dispositivo. Isso é um ótimo resultado, pois mostra que a área de atuação prevista foi efetivamente varrida pelos sensores, sem nenhuma brecha.

Quanto às distâncias medidas é mais fácil a análise com a representação da Figura 5.15.

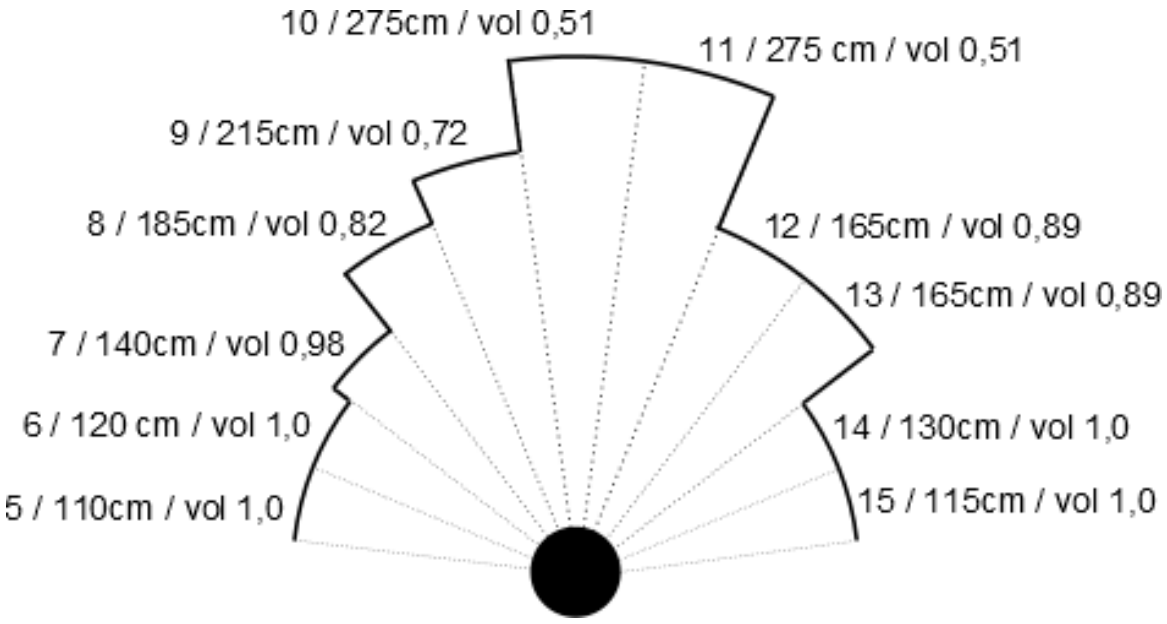



Figura 5.15 – Representação gráfica dos resultados, com os números das seções

Esse diagrama mostra com maior fidelidade a informação recebida pelo usuário. Ele não possui acesso à números e medidas, apenas a noções de distâncias e ângulos azimutais recebidos via som. As onze seções são representadas e numeradas, e seus tamanhos são ajustados baseados nas distâncias medidas e multiplicadores de volume encontrados.

Vê-se com essa imagem que as paredes laterais da garagem são bem caracterizadas pelas seções 5, 6, 14 e 15, e com o som de volume máximo vindo dessas direções o usuário teria facilidade de identifica-las.

As seções 10 e 11 representam o degrau mais à fundo no recinto, atrás das caixas, que apesar de possuir apenas 25 centímetros de altura, ele é detectado pelos sensores. Isso pode ter repercussões positivas e negativas. O usuário consegue identificar que à sua frente o caminho possui uma obstrução à distância, porém é incapaz de dizer que é apenas um pequeno degrau. Caso esse não seja um ambiente familiar, pode pensar que essa informação corresponde à uma parede que não se pode atravessar, podendo confundi-lo.

As seções 8, 9, 12 e 13 mostram a influência das caixas de papelão na varredura. Apesar de não ser claro o formato das caixas, ou mesmo seu início e fim, fica bastante claro, principalmente nas seções 9 e 12, que ali existe algum tipo de objeto, pois sem sua presença essas seções apresentariam volumes de áudio muito menores.

Destaca-se que todas essas informações são exclusivamente adquiridas pelo dispositivo, já que durante o teste a bengala não tocou nenhum dos objetos nem paredes, e o manipulador

do instrumento não se deslocou em nenhuma direção. Sem o aparelho, portanto, o usuário não teria nenhuma informação sobre a garagem em questão.

## 5.5) Limites do projeto

Uma das questões levantadas durante a fase final do projeto foi quais seriam suas limitações devido ao hardware, no caso qual a velocidade máxima que se consegue captar medidas e transmiti-las ao usuário levando em conta os componentes utilizados. Esse projeto possui dois grandes gargalos de velocidade: O sensor de distância e a velocidade de reprodução do som.

Tomando como frequência de reprodução do som a frequência com que foi amostrado, 44100 Hz, a reprodução de uma HRIR de 512 pontos leva 11,61 milissegundos. A reprodução de onze delas, completando uma varredura das seções azimutais, leva, portanto aproximadamente 128 ms. Taxas de reprodução maiores reduzem esse tempo, porém ao alterará-las deve-se levar em consideração a qualidade da informação. É possível reproduzir um som com velocidade extremamente alta, mas se isso altera a percepção da localização da fonte sonora que se quer emular, o aparelho perde seu propósito.

Já os sensores de distância levam 25 ms no pior dos casos para retornar uma medida, e a tarefa que a realiza é executada a cada 40 ms. Nos 128 ms que se leva para realizar a reprodução sonora de todos os setores é possível então realizar três medidas, uma para cada sensor. Esse é, portanto, um limite plausível para o sistema, porém ainda deve-se analisar se o usuário é capaz de assimilar informações transmitidas tão rapidamente aos seus ouvidos, e se apenas três medidas por varredura é o suficiente para se orientar com segurança. Essa questão só será respondida com testes mais extensos envolvendo voluntários, que infelizmente não foram contemplados nesse trabalho.

# 6) Conclusão

Analisando os resultados encontrados o desempenho do dispositivo atendeu aos requisitos, realizando as funções propostas e atingindo os objetivos do projeto. O alcance do aparelho construído é de aproximadamente 250 centímetros, valor três vezes maior do que o apresentado pela bengala física utilizada (*monopod*), que foi constatado ser 85 centímetros. Considerando que esses dois números se somam, têm-se um alcance total extremamente longo nas regiões atendidas por ambos os instrumentos.

A área de atuação resultante também aumenta drasticamente, com a varredura realizada adquirindo informações de uma área bastante ampla, chegando a varrer uma seção de 150 graus, número próximo do ângulo de visão de pessoas sem deficiência. Algumas regiões, no entanto, não são atendidas pela bengala física, somente pelo aparelho o que causa a existência de alguns pontos cegos imediatamente à esquerda e à direita do usuário, que deve se atentar a eles ao se locomover para evitar acidentes.

Deve-se destacar, porém, que a qualidade de informação não se compara com nenhum tipo de informação visual. Como o método escolhido para notificar o usuário da leitura realizada foi o som, os dados recebidos são bastante qualitativos, e servem apenas para auxiliar em sua orientação geral, não providenciando conhecimento muito detalhado do ambiente ao redor. Não é possível à distância, por exemplo, descobrir a altura de obstáculos, determinar seu tamanho com precisão ou até mesmo diferenciar um degrau ou uma pequena mureta de uma parede, ou uma cadeira de um pilar, por exemplo. Todavia, levando-se em conta que sem o aparelho qualquer tipo de informação estaria indisponível, essas simples adições podem ser consideradas uma boa melhoria.

Não se pode esquecer de que o desempenho do aparelho só pode ser julgado com mais autoridade pelo seu público alvo, que são capazes de dar um veredito mais concreto sobre sua viabilidade após um tempo de treinamento para aprender a utilizá-lo. Apesar desse passo não ter sido contemplado pelo trabalho, os resultados se mostraram bastante promissores e inspiram confiança na aprovação dos usuários.

Quanto à utilização do microcontrolador, este não apresentou nenhum problema para a realização das funções a ele designadas. Se mostrou um produto de ótima qualidade e desempenho, além de sua fácil utilização sem a necessidade de software ou hardware extra, e alta quantidade de bibliotecas disponíveis para rápido desenvolvimento e teste de código.

A placa de desenvolvimento, já com todo o hardware necessário para comunicação USB, I2C e saídas PWM se mostrou ideal para o projeto, sendo possível implementar todas as funcionalidades necessárias sem problemas.

No entanto, algumas partes do projeto mostraram que podem ser melhoradas. A começar pela medição angular, que apresentou um desvio no tempo e uma baixa exatidão na medida. Apesar de não ser um problema para este projeto em particular, é algo que se deve observar caso haja a necessidade de reutilizar esse tipo de medição posteriormente, bem como caso se deseje melhorar esse mesmo trabalho e implementar funções que requerem medições mais precisas. Tendo dito isso, a unidade de medição inercial se mostrou bastante prática e de fácil utilização, sendo possível a comunicação rápida através da interface I2C.

Os sensores de distância excederam expectativas quanto ao seu alcance e poder de detecção. Em particular no teste com o galão de água, um objeto relativamente pequeno que foi detectado a distâncias de mais de dois metros com bastante exatidão. Com isso o sensor se mostrou também uma boa escolha para o projeto. Todavia, há espaço para melhoras. O fato desse sensor utilizar ondas ultrassônicas para seu funcionamento o torna um tanto lento, limitando o dispositivo como um todo.

## 6.1) Trabalhos Futuros

Pode-se sugerir algumas melhorias no projeto a fim de aprimorar seu desempenho. Sensores de distância mais rápidos podem aumentar a velocidade de aquisição de dados, que por consequência resulta em uma maior quantidade de medições durante o tempo de uma varredura. Como a velocidade da varredura mecânica (movimentação lateral da bengala) não pode ser aumentada, uma vez que o usuário ainda precisa se sentir confortável ao utilizá-la, essa maior velocidade de medição não implica diretamente em uma varredura mais rápida do ambiente, mas sim em quantidades maiores de dados sobre uma mesma região. Com isso uma análise mais crítica dos dados pode ser realizada, aumentando assim a qualidade das medidas reduzindo-se ruídos e rejeitando leituras falsas.

Sensores de distância com uma faixa de atuação mais estreita beneficiariam bastante o dispositivo, permitindo medidas mais precisas e evitando que um mesmo obstáculo seja identificado repetidamente e posicionado em diferentes seções azimutais, ocupando um espaço maior do que suas dimensões reais. Também se tornaria viável a melhoria da resolução das seções,  e providenciando maiores detalhes nas informações fornecidas ao usuário.

Sensores de distância com operação infravermelha possuem ambas as qualidades mencionadas acima sendo, portanto, uma opção a ser explorada. Deve-se atentar que esse tipo de sensor possui em geral um alcance menor que os utilizados, sendo preciso avaliar a viabilidade da troca, bem como o aumento de custo que eles apresentariam. Para a questão do aumento da resolução das seções azimutais, seriam necessárias novas medidas de HRIR, além de medidas de ângulo mais confiáveis. No entanto, esse tipo de sensor possui limitações quanto sua utilização como ruídos provenientes da luz solar. Uma abordagem interessante seria, então, a utilização de infravermelhos em locais fechados, enquanto que os atuais sensores seriam responsáveis por medidas em locais abertos. (ACRONAME)

Outro aspecto a ser explorado é a implementação de diferentes formas de conectividade. Apesar dos riscos já mencionados nesse documento, um estudo de melhorias e *design* mecânicos adequados pode tornar viável a utilização de hardware sensível na unidade de sensoriamento, permitindo comunicação sem fio entre as duas unidades, ou até mesmo sua integração em uma única, posicionada no extremo da bengala.

Seguindo esse pensamento, pode-se aproveitar o poder de processamento da placa de desenvolvimento mbed para criar uma linha de comunicação com dispositivos inteligentes, como um smartphone. Isso abriria a possibilidade de controle do dispositivo, bem como o desenvolvimento de funções extras caso necessário, integradas com outros tipos de dispositivos e tecnologias. Devido à natureza móvel do aparelho, um método de comunicação interessante para este fim seria o Bluetooth, que não depende de conexão com a internet, nem sempre disponível em lugares públicos ou recintos não familiares ao usuário.

## Referências bibliográficas


ADAM. **PWM.** Disponível em:
<http://www.adam-audio.com/en/technology/pwm>
Acesso em 26 Junho 2015.

ACRONAME. **Sharp infrared ranger comparison.** Disponível em:
< https://acroname.com/articles/sharp-infrared-ranger-comparison >
Acesso em 28 Junho 2015

BERK, Aaron. **Biblioteca para filtro IMU.** 6 Setembro 2010. Disponível em:
<https://developer.mbed.org/cookbook/IMU>
Acesso em 25 Maio 2015.

CYTRON. **Product User's Manual – HC-SR04 Ultrassonic Sensor.** Disponível em:
<https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit>
Acesso em 27 Maio 2015.

DURACELL. **Datasheet da bateria MN1604 6LR61.** Disponível em
<http://ww2.duracell.com/media/en-US/pdf/gtcl/Product_Data_Sheet/NA_DATASHEETS/MN1604_6LR61_US_CT.pdf>
Acesso em 30 Maio 2015

FAIRCHILD, 2014. **LM78XX / LM78XXA 3 Terminal 1A Positive Voltage Regulator**. Disponível em:
<https://www.fairchildsemi.com/datasheets/LM/LM7805.pdf>
Acesso em 30 Maio 2015.

LISTEN. **Listen HRTF Database.** Disponível em:
<http://recherche.ircam.fr/equipes/salles/listen/index.html>
Acesso em 20 Maio 2015

MADGWICK, S. O. H. **An efficient orientation filter for inertial and inertial/magnetic sensor arrays.** 30 Abril, 2010.

MBED. **ARM mbed Developer Site.** Disponível em:
<https://developer.mbed.org/>
Acesso em 21 Maio 2015.

NASA, 2015. **Aircraft Rotations – Body axes.** Disponível em:
<https://www.grc.nasa.gov/www/k-12/airplane/rotations.html>
Acesso em 27 Maio 2015

NCSU. **Sound Localization**. Disponível em:
< http://www.ise.ncsu.edu/kay/msf/sound.htm>
Acesso em 26 Junho 2015

NFB. **Future Reflections, volume 27, number 2 – Cane Travel and Independence.** Disponível em:
<https://nfb.org/Images/nfb/Publications/fr/fr27/2/fr2702tc.htm>
Acesso em 26 Junho 2015

NXP, 2014a. **I2C-bus specification and user manual.** Disponível em:
<http://www.nxp.com/documents/user_manual/UM10204.pdf>
Acesso em 27 Maio 2015.

POLOLU. **MinIMU-9 v3 Gyro, Accelerometer, and Compass (L3GD20H and LSM303D Carrier)**. Disponível em:
<https://www.pololu.com/product/2468>
Acesso em 23 Maio 2013.

ST-1, 2013. **LSM303D Datasheet.** Disponível em:
<http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00057547.pdf>
Acesso em 25 Maio 2015.

ST-2, 2013 **L3GD20H Datasheet.** Disponível em:
<http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00060659.pdf>
Acesso em 25 Maio 2015.

UCDAVIS. **Psychoacoustics of Spatial Hearing.** Disponível em:
<http://interface.idav.ucdavis.edu/sound/tutorial/psych.html>
Acesso em 20 Maio 2015.

UW. **Sound Localization and the Auditory Scene.** Disponível em:
<http://courses.washington.edu/psy333/lecture_pdfs/Week9_Day2.pdf>
Acesso em 20 Maio 2015.

VASCA, F. & IANELLI, L. **Dynamics and Control of Switched Electronic Systems:** Advanced Perspectives for Modeling, Simulation and Control of Power Converters. Springer-Verlag London Limited, 2012.

VISIONAWARE. **Essential skills for everyday living with vision loss.** Disponível em:
<http://www.visionaware.org/info/everyday-living/essential-skills/12>
Acesso em 28 Junho 2015

WHO. **Visual impairment and blindness.** Disponível em:
<http://www.who.int/mediacentre/factsheets/fs282/en/>
Acesso em 28 Junho 2015

## Bibliografia complementar

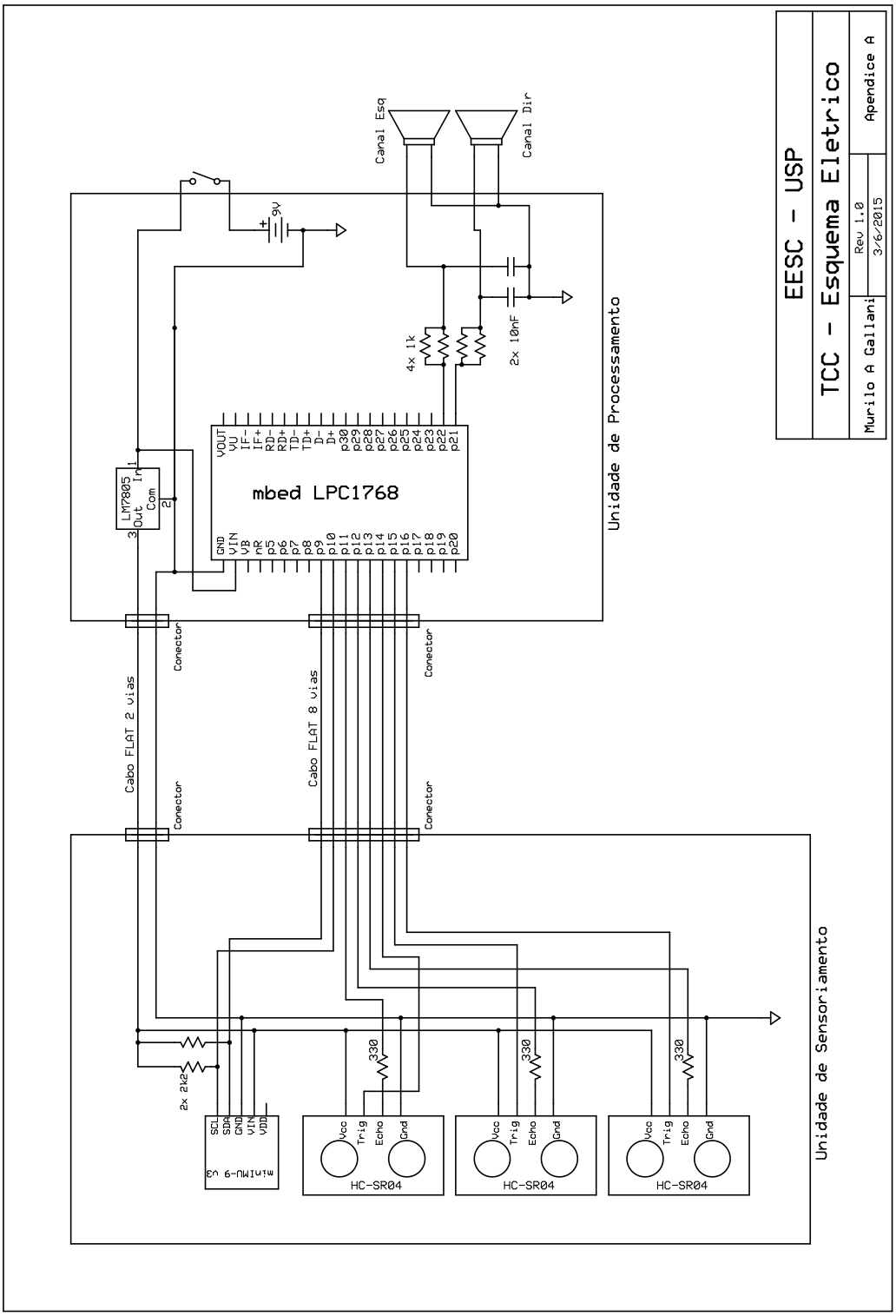
ANALOG. **Fundamentals of Direct Digital Synthesis (DDS).** Disponível em: <http://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf> Acesso em 20 Maio 2015.

NXP, 2009. **mbed NXP LPC1768 Microcontroller Flyer.** Disponível em: <http://www.nxp.com/documents/leaflet/LPC1768.pdf> Acesso em 22 Maio 2015

NXP, 2014b. **NXP LPC176x User Manual.** Disponível em: <http://www.nxp.com/documents/user_manual/UM10360.pdf> Acesso em 22 Maio 2015

NXP, 2014c. **NXP LPC176x Datasheet.** Disponível em: <http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf> Acesso em 22 Maio 2015.

# Apêndice A – Esquema Elétrico do dispositivo



## Apêndice B – Código C++ de calibração da IMU

```
#include "mbed.h"

#define accel_add 0x3A
#define gyro_add 0xD6

I2C i2c(p9, p10);
Serial pc(USBTX, USBRX);


char data[6];
int data_aux;
float accel_acc[3], accel_offset[3], gyro_acc[3], gyro_offset[3];
signed char* signed_data = reinterpret_cast<signed char*> (data);
int i,j;

int main(){
    data[0] = 0x20;
    data[1] = 0x7F; // liga giroscopio 200hz
    i2c.write(gyro_add, data, 2);
    data[0] = 0x20;
    data[1] = 0x77; // liga acelerômetro 200hz
    i2c.write(accel_add, data, 2);

    for (i = 0; i < 3; i++){
        accel_acc[i] = 0;
        gyro_acc[i] = 0;
    }

    wait_ms(500);
    for (i = 0; i <200; i++) {
        data[0] = 0xA8;
        i2c.write(gyro_add,  data, 1, true);
        i2c.read(gyro_add, data, 6);
        for (j = 0; j < 3; j++) {
            data_aux = signed_data[2*j+1];
            data_aux <<=8;
            data_aux += data[2*j];
            gyro_acc[j] += data_aux;
        }

        data[0] = 0xA8;
        i2c.write(accel_add,  data, 1, true);
        i2c.read(accel_add, data, 6);
        for (j = 0; j < 3; j++) {
            data_aux = signed_data[2*j+1];
            data_aux <<=8;
            data_aux += data[2*j];
            accel_acc[j] += data_aux;
        }
    wait_ms(5);
    }

    for ( i = 0; i < 3; i++){
```

```
            gyro_offset[i] = gyro_acc[i]/200;
            accel_offset[i] = accel_acc[i]/200;
        }
        accel_offset[2] -= 1/0.000061;
        pc.printf("\r\nOffset Acelerometro:\r\nx: %3.2f - %3.2f\ty: %3.2f -
%3.2f\tz: %3.2f - %3.2f\r\nOffset Giroscopio:\r\nx: %3.2f - %3.2f\ty: %3.2f -
%3.2f\tz: %3.2f - %3.2f",
                    accel_offset[0], accel_offset[0]*0.000061*9.80665,
accel_offset[1], accel_offset[1]*0.000061*9.80665, accel_offset[2],
accel_offset[2]*0.000061*9.80665,
                    gyro_offset[0], gyro_offset[0]*0.0175, gyro_offset[1],
gyro_offset[1]*0.0175, gyro_offset[2], gyro_offset[2]*0.0175);


    }
```

## Apêndice C – Código C++ Principal

```
include "mbed.h"
#include <math.h>
#include <algorithm>
#include "IMUfilter.h"
#include "rtos.h"

// DEFINIÇÃO DAS CONSTANTES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#define pi 3.14159265359

#define accel_add 0x3A      // endereço do aceletometro para
comunicação i2c
#define gyro_add 0xD6       // endereço do gyroscopio para comunicação
12c
#define gyro_gain 0.00875   // ganho do gyroscópio ( 1 unidade
corresponde a 0.00875 dps )
#define accel_gain 0.000061*9.80665 // ganho do acelerômetro já
incluindo gravidade
#define cane_length 85      // Distancia do usuário à unidade de
sensoriamento
#define range 250           // Alcance máximo do sensor de distância
#define limite_prox (cane_length + 50)           // Limite entre faixa
ativa e faixa proxima
#define limite_dist (cane_length + range)        // Limite entre faixa
ativa e faixa distante
#define vol_prox 1          // volume da faixa prox
#define vol_dist 0.3        // volume da faixa dist
#define vol_n 0.1           // volume da faixa nao medida
#define duration 0.1        // Tempo total que será tocado o trem de
"beeps"

const float accel_offset[] = {290, 266, 34749};         // valores de
offset para os tres eixos do acelerômetro
const float gyro_offset[] = {379.14, 382.52, 95.24 };   // valores de
offset para os tres eixos do giroscopio
const int ang_sensores[] = {-90, 0, 90};                // angulo
físico, em relação à frente do usuario, da direçao onde os sensores de
distância estão apontados

const float pulse[11][2][512]; // Preencher com as HRIR desejadas


Serial pc(USBTX, USBRX);    // comunicação serial
I2C i2c(p9, p10);          // comunicação i2c
DigitalOut trig0(p14),trig1(p15),trig2(p16); // Pinos de trigger dos
sensores de distância
InterruptIn echo0(p11),echo1(p12),echo2(p13);



Timer thrd_timer;               // Contador do tempo de execução da
Tarefa 1

// Função para leitura dos sensores de distância
```

```
      float getdist();
      Timer t;                          // temporizador para a medição da
distância
      int sensor_en = 0;                // Variável que indica qual sensor de
distância será utilizado
      volatile int timer_flag = 0;      // flag que indica fim da contagem de
tempo
      void start_t();                   // rotina de interrupção para iniciar
contagem
      void stop_t();                    // rotina de interrupção para parar
contagem


      // Função para leitura da unidade de medição inercial
      void read_imu(void const *args);
      IMUfilter imuFilter(0.04, 10);      // Inicialização do filtro que
calcula roll, pitch e yaw
      Timeout front_tout, dist_tout;      // Inicialição dos timeouts do
sensoriamento
      void tout_front();                  // Rotina de interrupção de timeout
do cálculo de ângulo frontal
      void tout_dist();                   // Rotina de interrupção de timeout
da medida de distância
      volatile int front_tout_flag, dist_tout_flag;   // Flags que indicam a
chamada da rotina de interrupção
      char data[6];                       // Variável que armazena os 6 bits
correspondentes às mediçoes dos 3 eixos do acelerometro e giroscopio
      int data_aux;                       // Variável auxiliar para
tratamendo de dados
      signed char* signed_data = reinterpret_cast<signed char*> (data);   //
Variável auxiliar para conversão do complemento de dois
      float accel_out[3],gyro_out[3];                 // Variaveis que
armazenam os valores finais lidos do acelerometro e giroscopio
      float imuAng[2], delta_imuAng[2], extremo[2];   // Variaveis que
armazenam os ultimos angulos medidos, as variaçoes do angulo e os extremos
      float registro_buffer[24][2],registro[24][2];   // Matrizes que
armazenam a distância medida para cada ângulo ao redor do usuário
      float dist_med, ang_med, a, b;                  // Variáveis auxiliares
para o cálculo final da distância e ângulo medidos

      // Função que classifica qualquer ângulo medido em um das faixas uteis
para o projeto
      int classifAng(int ang);

      // Função que toca um trem de beeps para um certo ângulo durante um
tempo predeterminado
      void play();
      void stop_play();       // rotina de interrupção que para o som
      volatile int keep_play; // Variavel que habilita ou não tocar o som
      Timeout stop_p;         // Objeto timeout para a parada do som



      // Função que atualiza os valores de saída nos dois canais para a
reprodução do som
      void earphones();
```

```
      Ticker output;          // Objeto para chamar a função repetidamente
      PwmOut rchannel(p22);   // Canal stereo direito no pino p22
      PwmOut lchannel(p21);   // Canal stereo esquerdo no pino p21
      volatile int pulse_index;   // Indexador da matriz que contem as HRIR



      float front, active_angles[13];
      int i,j,k, extremo_convert[2], angle_span, current_index;
      int front_index;    // Número da seção frontal
      int angle_index;    // Índice para a reprodução da HRIR correta
      float vol;          // Multiplicador da amplitude da HRIR


      //****************************** MAIN
**********************************
      int main()
      {
          pc.printf("\r\nStart...");

          // Inicialização do PWM em 250kHz
          rchannel.period_us(4);
          lchannel.period_us(4);

          // Inicialização do IMU
          data[0] = 0x20; // Endereço do registrador de configuração
          data[1] = 0x7F; // liga giroscopio 200hz
          i2c.write(gyro_add, data, 2);
          data[0] = 0x20; // Endereço do registrador de configuração
          data[1] = 0x77; // liga acelerômetro 200hz
          i2c.write(accel_add, data, 2);
          sensor_en = 0;

          // Inicialização das interrupções para os pinos Echo
          echo0.rise(&start_t);
          echo0.fall(&stop_t);
          echo1.rise(&start_t);
          echo1.fall(&stop_t);
          echo2.rise(&start_t);
          echo2.fall(&stop_t);
          output.attach_us(&earphones, 22);   // Configura reprodução do som
em 44100 Hz
          Thread thread(read_imu);            // Tarefa 1



      // ****************************** TAREFA 2
**********************************

          while(true) {
              front_index = classifAng((int)front)/15;        // Indice da
seção do angulo frontal
              for(i = 0; i < 24; i++) {                       // Faz uma
copia do registro para utilizar durante a reprodução do som
                  registro[i][0] = registro_buffer[i][0];
                  registro[i][1] = registro_buffer[i][1];
```

```
            }
            for(i = 0; i < 24; i++) { // Limpa matriz registro para novas
medidas
                registro_buffer[i][1] = 0;
                registro_buffer[i][0] = 0;
            }

            pc.printf("\r\nfront: %d", front_index);
            for(k = front_index - 5; k < front_index + 6; k++) {
// Loop para selecionar as onze seções para reprodução
                current_index = k;
                angle_index = current_index - front_index + 5;
                if (current_index < 0) current_index = 24+current_index;
// Correção para indices negativos
                else if (current_index > 23) current_index = current_index
- 24;    // Correção para índices maiores que 23
                if (registro[current_index][1] == 1) {
// Verifica se seção foi medida recentemente
                    if (registro[current_index][0] < limite_prox) vol =
vol_prox;           // Verifica se modulo esta na faixa proxima
                    else if (registro[current_index][0] > limite_dist) vol
= vol_dist;           // Verifica se modulo esta na faixa distante
                    else vol = 1 - (registro[current_index][0] -
limite_prox)/((limite_dist-limite_prox)*(vol_prox - vol_dist));    //
Verifica se modulo esta na faixa ativa
                } else {
                    vol = vol_n;
                }
                pc.printf("\r\nplaying: %d\t\tdistancia: %3.2f\t\tvol:
%3.2f", current_index, registro[current_index][0],vol);
                play();
            }

            Thread::wait(20);

        }
    }



    // Função que inicia a máquina de estados, tocando um trem de "beeps"
por um tempo previamente determinado, utilizando as configurações
selecionadas posteriormente à chamada da função
    void play()
    {
        keep_play = 1;                      // Flag que permite que o som
seja tocado
        stop_p.attach(stop_play, duration); // Após um atraso, é chamada a
função que faz com que o som pare
        while(keep_play);                   // espera
        pulse_index = 0;
    }

    // Função que para o som
    void stop_play()
    {
        keep_play = 0;
```

```
    }


    // Função que percorre a senoide, caso habilitada, alterando o valor do
pwm e consequentemente o valor da saída
    void earphones()
    {
        if (keep_play) {
            rchannel.write(0.5 + vol*pulse[angle_index][1][pulse_index]);
            lchannel.write(0.5 + vol*pulse[angle_index][0][pulse_index]);
            pulse_index++;
            if (pulse_index >= 512) pulse_index = 0;
        }
    }

    // ************************ TAREFA 1
****************************************

    void read_imu(void const *args)
    {
        while(true) {
            // Inicia contagem do tempo de execução da tarefa
            thrd_timer.reset();
            thrd_timer.start();

            // Processo de leitura do Giroscopio
            data[0] = 0xA8;                          // Endereço do primeiro
registrador a se ler, auto-incremento do endereço habilitado
            i2c.write(gyro_add,  data, 1, true);    // Escreve-se o
endereço no sensor, sem sinal de stop
            i2c.read(gyro_add, data, 6);            // Lê-se 6
registradores em sequência
            for (j = 0; j < 3; j++) {
                data_aux = signed_data[2*j+1];  // Move-se os HSB das
medidas, com sinal, para uma variavel auxiliar
                data_aux <<=8;                  // Rotaciona-se 8 8
posições para a esquerda
                data_aux += data[2*j];          // Soma-se com o LSB
                gyro_out[j] = (data_aux - gyro_offset[j])*gyro_gain; //
Remove-se o offset e multiplica o resultado pelo ganho do giroscópio
            }

            // Processo de leitura do acelerômetro, igual ao do giroscópio
            data[0] = 0xA8;
            i2c.write(accel_add,  data, 1, true);
            i2c.read(accel_add, data, 6);
            for (j = 0; j < 3; j++) {
                data_aux = signed_data[2*j+1];
                data_aux <<=8;
                data_aux += data[2*j];
                accel_out[j] = (data_aux - accel_offset[j])*accel_gain;
            }

            // Atualização do filtro com os novos valores e cálculo da nova
posição
```

```
            imuFilter.updateFilter(gyro_out[0]/57.2957795131,
gyro_out[1]/57.2957795131, gyro_out[2]/57.2957795131, accel_out[0],
accel_out[1], accel_out[2]);
            imuFilter.computeEuler();

            dist_med = getdist(); // Leitura do sensor de distância

            // Se medida maior que alcance max ou ocorre timeout, poe
modulo na faixa distante
            if (dist_med > range) dist_med = 500;
            if (dist_tout_flag == 1) {
                dist_med = 500;
                dist_tout_flag = 0;
            }


            // Calculo da distância e ângulo relativo ao usuario, levando
em consideraçao tamanho da bengala e angulo do sensor
            a = cane_length + dist_med*cos(ang_sensores[sensor_en]*pi/180);
            b = dist_med*sin(ang_sensores[sensor_en]*pi/180);
            dist_med = sqrt(a*a + b*b);
            dist_med = ((int)(dist_med)/5)*5;
            ang_med = asin(b/dist_med)*180/pi;

registro_buffer[classifAng(imuFilter.getYaw()*57.2957795131+ang_med)/15][0] =
dist_med;

registro_buffer[classifAng(imuFilter.getYaw()*57.2957795131+ang_med)/15][1] =
1;


            // Armazenamento da posição anterior e da nova posição

            imuAng[1] = imuAng[0];
            imuAng[0] = imuFilter.getYaw()*57.2957795131;

            // Cálculo da nova variação de ângulo, armazenamento da antiga
posição
            delta_imuAng[1] = delta_imuAng[0];
            delta_imuAng[0] = imuAng[0] - imuAng[1];

            // Casos especiais para quando os ângulos estão próximos à +-
180 graus
            if  (delta_imuAng[0] > 300) {
                delta_imuAng[0] -= 360;
            }
            if (delta_imuAng[0] < -300) {
                delta_imuAng[0] = 360 - delta_imuAng[0];
            }


            // Se um extremo foi encontrado, flag vai pra 2 e inicia
contagem do timeout
            if (front_tout_flag == 0) {
                front_tout_flag = 2;
                front_tout.attach(&tout_front, 1.5);
            }
```

```
            // Se a variação anterior e a nova têm sinais diferentes,
significa que é um extremo do movimento da bengala
            // Timeout também indica extremo para dar continuidade ao
programa
            if (delta_imuAng[0]*delta_imuAng[1] < 0  || front_tout_flag ==
1) {
                front_tout_flag = 0;    // Se novo extremo foi encontrado,
flag vai pra zero
                front_tout.detach();
                extremo[1] = extremo[0];
                extremo[0] = imuAng[0];

                // Para se calcular o ângulo frontal, tira-se a média dos
dois extremos,
                //levando em consideração casos especiais onde os extremos
estão no 2o e 3o quadrante
                if (extremo[0] - extremo[1] > 180) {
                    front = -180 + (extremo[0] + extremo[1])/2;
                } else if (extremo[0] - extremo[1] < -180) {
                    front = 180 + (extremo[0] + extremo[1])/2;
                } else front = (extremo[0] + extremo[1])/2;
            }

            thrd_timer.stop();
            Thread::wait(35 - thrd_timer.read_ms());
        }
    }

    // Rotina para coletar dados dos sensores de distância
    float getdist()
    {
        sensor_en ++;       // Troca sensor
        if (sensor_en > 2) sensor_en = 0;

        // Rotina para um sensor (0), demais rotinas são iguais, trocando
apenas as portas do microcontrolador
        if (sensor_en == 0) {
            trig0 = 0;  // Trigger em nivel baixo para iniciar o processo
            t.stop();   // Para e zera o contador
            t.reset();
            trig0 = 1;  // Leva trigger à nivel alto por 10us
            wait_us(10);
            trig0 = 0;
            timer_flag = 0;         // Limpa flags pertinentes
            dist_tout_flag = 0;
            dist_tout.attach(&tout_dist, 0.025);    // Inicia contagem de
timeout
            while(timer_flag == 0 && dist_tout_flag == 0);  // Aguarda
final da contagem ou timeout
            dist_tout.detach(); // cancela timeout
            t.stop();           // Para contador e retorna o valor da
distância
            return t.read()*340*100/2;
        } else if (sensor_en == 1) {
            trig1 = 0;
            t.stop();
```

```
        t.reset();
        trig1 = 1;
        wait_us(10);
        trig1 = 0;
        timer_flag = 0;
        dist_tout_flag = 0;
        dist_tout.attach(&tout_dist, 0.025);
        while(timer_flag == 0 && dist_tout_flag == 0);
        dist_tout.detach();
        t.stop();
        return t.read()*340*100/2;
    } else if (sensor_en == 2) {
        trig2 = 0;
        t.stop();
        t.reset();
        trig2 = 1;
        wait_us(10);
        trig2 = 0;
        timer_flag = 0;
        dist_tout_flag = 0;
        dist_tout.attach(&tout_dist, 0.025);
        while(timer_flag == 0 && dist_tout_flag == 0);
        dist_tout.detach();
        t.stop();
        return (t.read()*340*100/2);
    }
    return 0;
}

// Rotina de interrupção que inicia timer
void start_t()
{
    t.start();
}

// Interrupção que para Timer
void stop_t()
{
    t.stop();
    timer_flag = 1;
}

// Timeout da medida de distância
void tout_dist()
{
    dist_tout_flag = 1;
}

// rotina de timeout de angulo extremo
void tout_front()
{
    front_tout_flag = 1;
}

// Classificação de ângulos
int classifAng(int ang)
{
```

```
          if ((ang >= -7.5)&&(ang < 7.5)) return 0;
          else if ((ang >= 7.5)&&(ang < 22.5)) return 15;
          else if ((ang >= 22.5)&&(ang < 37.5)) return 30;
          else if ((ang >= 37.5)&&(ang < 52.5)) return 45;
          else if ((ang >= 52.5)&&(ang < 67.5)) return 60;
          else if ((ang >= 67.5)&&(ang < 82.5)) return 75;
          else if ((ang >= 82.5)&&(ang < 97.5)) return 90;
          else if ((ang >= 97.5)&&(ang < 112.5)) return 105;
          else if ((ang >= 112.5)&&(ang < 127.5)) return 120;
          else if ((ang >= 127.5)&&(ang < 142.5)) return 135;
          else if ((ang >= 142.5)&&(ang < 157.5)) return 150;
          else if ((ang >= 157.5)&&(ang < 172.5)) return 165;
          else if ((ang < -172.5)||((ang >= 172.5)&&(ang < 187.5))) return
180;
          else if ((ang <= -157.5 && ang > -172.5)||(ang >= 187.5 && ang <
202.5)) return 195;
          else if ((ang <= -142.5 && ang > -157.5)||(ang >= 202.5 && ang <
217.5)) return 210;
          else if ((ang <= -127.5 && ang > -142.5)||(ang >= 217.5 && ang <
232.5)) return 225;
          else if ((ang <= -112.5 && ang > -127.5)||(ang >= 232.5 && ang <
247.5)) return 240;
          else if ((ang <= -97.5 && ang > -112.5)||(ang >= 247.5 && ang <
262.5)) return 255;
          else if ((ang <= -82.5 && ang > -97.5)||(ang >= 262.5 && ang <
277.5)) return 270;
          else if ((ang <= -67.5 && ang > -82.5)||(ang >= 277.5 && ang <
292.5)) return 285;
          else if ((ang <= -52.5 && ang > -67.5)||(ang >= 292.5 && ang <
307.5)) return 300;
          else if ((ang <= -37.5 && ang > -52.5)||(ang >= 307.5 && ang <
322.5)) return 315;
          else if ((ang <= -22.5 && ang > -37.5)||(ang >= 322.5 && ang <
337.5)) return 330;
          else if ((ang <= -7.5 && ang > -22.5)||(ang >= 337.5 && ang <
352.5)) return 345;
          return 0;
      } // classifAng
```