ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

JHENNERSON BARBOSA DA SILVA

JOCIEL LOPES DAS NEVES

MARCELO GONÇALVES DE CARVALHO

MARIO SÉRGIO COSTA DE MATOS

RAFAEL APARECIDO AMADO MORENO

DETECTOR DE CHORO PARA PAIS DEFICIENTES AUDITIVOS

São Paulo

2016

ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

JHENNERSON BARBOSA DA SILVA
JOCIEL LOPES DAS NEVES
MARCELO GONÇALVES DE CARVALHO
MARIO SÉRGIO COSTA DE MATOS
RAFAEL APARECIDO AMADO MORENO

DETECTOR DE CHORO PARA PAIS DEFICIENTES AUDITIVOS

Trabalho de Conclusão de Curso apresentado ao Ensino Técnico da ETEC Horácio Augusto da Silveira, como requisito parcial para conclusão do curso de Técnico em Eletrônica.

ORIENTADOR: Prof. Ulisses Nogueira

CO-ORIENTADOR: Prof. Jonatas Xavier

São Paulo

2016

Dedicamos este trabalho as nossas famílias por toda a ajuda e incentivo, aos nossos professores que nos auxiliaram no desenvolvimento e pesquisa, enfim, a todas as pessoas que fizeram parte desta etapa decisiva das nossas vidas.

AGRADECIMENTOS

À Professora Roseli Cshunderlick pela colaboração extremada na fase inicial deste trabalho.

Ao Professor Ulisses Nogueira por compartilhar conosco seus conhecimentos.

Aos Professores Jonatas Xavier e Carlos A. Zoboli por sua dedicação e colaboração no decorrer desta pesquisa, apresentando observações importantes em seus comentários.

“Apesar dos nossos defeitos, precisamos enxergar que somos pérolas únicas no teatro da vida e entender que não existem pessoas de sucesso ou pessoas fracassadas.”

(Augusto Cury)

RESUMO

Este trabalho tem por objetivo apresentar um equipamento eletrônico capaz de auxiliar os pais, em especial, os portadores de deficiência auditiva, na criação de seus filhos desde recém-nascidos até os 02 anos de idade que é o período em que a criança necessita de maiores cuidados, tudo isso por monitoramento através da comunicação Wi-Fi entre dois módulos ESP8266 contidos nas placas de desenvolvimento e prototipagem WeMos D1 e WeMos D1 Mini além de uma placa de desenvolvimento e prototipagem Arduino Uno embarcados em uma pulseira e uma central de monitoramento. A central de monitoramento tem como função a detecção do choro da criança e a transmissão do sinal de ativação da pulseira, já a pulseira que será usada pelos pais e terá como função alertar com o acender de um LED e por meio de vibração com um motor Vibracall.

Palavras-chave: Monitoramento, Choro, Pulseira,Vibração, ESP8266, Arduino, Wi-Fi.

CRYING DETECTOR POR HEARING DISABLED PARENTS

ABSTRACT

This paper aims to present an electronic device able to help parents, particularly those with hearing impairment in raising their children from newborns up to 02 years old which is the period in which the child needs more care all this by monitoring through the Wi-Fi communication between two ESP8266 modules and a development board and prototyping Arduino Uno embedded in a wristband and a central monitoring. The central monitoring function is to the child's crying detection and transmission of the bracelet activation signal, since the bracelet that will be used by parents and will to alert with the lighting of a LED and by vibrating with an engine vibrate.

Keywords: Monitoring, cry, Bracelet, ESP8266, Arduino, Wi-Fi.

LISTA DE FIGURAS

Figura 1 – Rádio Nurse e Guardian Ear 2
Figura 2 – Módulo ESP8266EX 10
Figura 3 – Diagrama Funcional do Módulo ESP8266EX 14
Figura 4 – PCB de Aquisição de Áudio 15
Figura 5 – Placa de Acoplamento 15
Figura 6 – Placa WeMos D1 17
Figura 7 – Placa WeMos D1 Mini 18
Figura 8 – Placa Arduino Uno 20
Figura 9 – Diagrama elétrico da Central de Monitoramento 21
Figura 10 – Diagrama elétrico da Pulseira 22
Figura 11 – Fluxograma do programa principal da placa Arduino Uno 50
Figura 12 – Fluxograma do programa principal da placa WeMos D1 51
Figura 13 – Fluxograma da interrupção da placa WeMos D1 52
Figura 14 – Fluxograma do programa principal da placa WeMos D1 Mini 53
Figura 15 – Fluxograma da interrupção do programa da placa WeMos D1 Mini 54
Figura 16 – Circuito da Central de Monitoramento 56
Figura 17 – Circuito da Pulseira 57
Figura 18 – Projeto Final 57

LISTA DE TABELAS

Tabela 1 – Principais Especificações Técnicas do Módulo ESP8266EX 12
Tabela 2 – Principais especificações técnicas da placa Arduino Uno 20
Tabela 3 – Cronograma de Desenvolvimento do Projeto 55

LISTA DE ABREVIATURA E SIGLAS

LED	Diodo Emissor de Luz (Light Emitting Diode)
Wi-Fi	Fidelidade Sem Fio (Wireless Fidelity)
VDC	Tensão em Corrente Contínua (Voltage Direct Current)
JFET	Transistor de Junção com Efeito de Campo (Junction Field Effect Transistor)
TX	Canal de Transmissão
RX	Canal de Recepção
I/O	Input/Output
UDP	User Datagram Protocol
OSI	Open Systems Interconnection
Hz	Hertz
mA	Mili Ampère
UART	Transmissão e Recepção Assíncrona Universal
CPU	Unidade de Processamento Central (Central Processing Unit)
GPIO	Entrada/Saída de Propósito Geral (General Purpose Input/Output)
PCB	Placa de Circuito Impresso (Printed Circuit Board)
IoT	Internet das Coisas (Internet of Things)
SDK	Kit de Desenvolvimento de Software (Software Development Kit)
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
AHB	Barramento Avançado de Alto Desempenho (Advanced High-performance Bus)

SUMÁRIO

1. INTRODUÇÃO 1

1.1. Contextualização 1

1.2. Objetivo 3

1.3. Justificativa 3

1.4. Objetivos Específicos 4

2. ESPECIFICAÇÃO DO PROJETO 5

2.1. Características Do Equipamento 5

2.2. Requisitos 6

3. MÓDULOS DO PROJETO 7

3.1. Central de Monitoramento 7

3.2. Pulseira 8

4. MÓDULO ESP8266 9

4.1. Introdução ao Módulo ESP8266 9

4.1.1. Aplicações do módulo ESP8266EX 13

5. PCBs DO PROJETO 14

5.1. PCB de Aquisição Áudio 14

5.1.1. Captação do Áudio 14

5.1.2. Pré-amplificação do Sinal de Áudio 15

5.1.3. Filtragem do Sinal de Áudio 15

5.1.4. Amplificação do Sinal de Áudio 15

5.2 Placa de Acoplamento 16

5.3. Placas de Desenvolvimento e Prototipagem WeMos D1 e D1 Mini 17

5.4. Placa de Desenvolvimento e Prototipagem Arduino 19

5.4.1. Introdução à placa Arduino 19

5.4.2. Especificações Técnicas da placa Arduino Uno 20

6. DIAGRAMAS ELÉTRICOS 21

6.1. Diagrama Elétrico da Central de Monitoramento 21

6.2. Diagrama Elétrico da Pulseira 22

7. FIRMWARES DO SISTEMA 23

7.1. Programas do sistema 23

7.1.1. Código do Firmware da Placa Arduino Uno (Central de Monitoramento). 23

7.1.2. Código do Firmware da Placa WeMos D1 (Central de Monitoramento) 28

7.1.3. Código do Firmware da Placa WeMos D1 Mini (Pulseira) 35

7.2. Fluxogramas do Sistema 48

7.2.1. Fluxograma da placa Arduino Uno (Central de Monitoramento) 48

7.2.2. Fluxograma da placa WeMos D1 (Central de Monitoramento) 51

7.2.3. Fluxograma da placa WeMos D1 Mini (Pulseira) 52

8. CRONOGRAMA 55

9. VALIDAÇÃO DO PROJETO 56

10. REFERÊNCIAS BIBLIOGRÁFICAS 58

11. ANEXOS 59

1. INTRODUÇÃO

1.1 Contextualização

A eletrônica nos tempos atuais é muito explorada nas áreas da saúde e monitoramento, já se tornou comum o uso de aparelhos para monitoramento de crianças como as “babás eletrônicas” que têm diversos modelos encontrados no mercado, estes dispositivos têm como finalidade captar uma informação seja ela o choro, movimento ou imagem da criança a ser monitorada através de um dispositivo de monitoramento e transmitir esta informação para um dispositivo receptor encarregado de alertar de alguma maneira à pessoa que está cuidando desta criança que a mesma está necessitando de cuidados.

O primeiro equipamento eletrônico de monitoramento de bebês foi o “Radio Nurse” que era um receptor de radiofrequência e alto falante moldado em baquelite projetado pelo escultor americano Isomu Noguchi (1904 – 1988) e fabricado pela Zenith Radio Corporation (Illinois - Chicaco, EUA, fundada em 1918). O “Radio Nurse” era usado em conjunto com o transmissor “Guardian Ear” e tinha como finalidade transmitir os sons do quarto onde ficava o transmissor para o receptor que podia ser colocado em qualquer cômodo da casa. Um fato interessante é que o “Radio Nurse” foi esteticamente projetado para sugerir a cabeça de uma enfermeira.

Figura 1: Rádio Nurse e Guardian Ear.

Fonte: Radio Nurse and Guardian Ear disponível em: <http://thenoguchimuseum.tumblr.com/image/98247372608> Acesso em 23 maio 2016.

1.2 Objetivo

O objetivo deste trabalho é desenvolver um equipamento eletrônico de baixo custo capaz de monitorar bebês com uma adaptação para ser usado por pais que possuem deficiência auditiva e que por este motivo se limitam ao uso dos convencionais aparelhos deste segmento que fazem uso do alarme sonoro para comunicar que o bebê está chorando, tudo isso a partir dos sentidos tátil e visual, por meio de vibração e luz proveniente de LEDs. Também tem como objetivo o uso da tecnologia Wi-Fi que tem sido usada cada vez com maior frequência na atualidade como meio de comunicação wireless, através do protocolo de rede UDP que é um protocolo simples da camada de transporte do modelo de protocolos de comunicação OSI.

1.3 Justificativa

Cuidar de um bebê é uma tarefa muito difícil, e é ainda mais difícil se o pai ou a mãe possuir alguma limitação auditiva, pensando em como seria árdua esta missão decidimos estudar o assunto e foi em depoimentos dados por pais em blogs destinados às pessoas com deficiência auditiva que vimos que a parte mais difícil é na hora de dormir, onde a preocupação em saber se a criança está bem é de tirar o sono. Muitos pais e mães não conseguem dormir de tanta que é a preocupação com a saúde da criança, uma das medidas adotadas por estes pais é dormir junto com a criança, mas, isso acaba trazendo um risco ainda maior que é o risco de machucar a criança ou até mesmo no pior dos casos leva-la à morte por sufocamento, além dos riscos aos próprios pais como os transtornos causados pela falta de um sono adequado tais como: fadiga, stress, sonolência diurna, falta de energia, irritabilidade, ansiedade, falta de concentração e memória, dores musculares ou até mesmo a depressão que é um dos piores problemas da atualidade (Coletiva de Imprensa sobre Insônia, São Paulo, 22 de junho de 2012).

São estes os motivos que nos levaram a desenvolver este projeto que é acessível e que possibilita auxiliar os pais deficientes auditivos na criação de seus filhos no período em que eles mais precisam de cuidados que é quando ainda são bebês.

1.4 Objetivos Específicos

Estudar e desenvolver um sistema embarcado baseado no módulo ESP8266 e na placa de desenvolvimento e prototipagem Arduino.

Estudar e desenvolver firmwares baseados na linguagem de programação Arduino.

Estudar e desenvolver uma estrutura de comunicação wireless baseada em uma rede UDP.

Estudar e projetar um circuito de sensoriamento eletroacústico utilizando amplificadores operacionais LM324.

2. ESPECIFICAÇÃO DO PROJETO

O Detector de Choro de Bebês Para Pais Deficientes Auditivos é um projeto que tem como finalidade alertar aos pais portadores de deficiência auditiva quando seu bebê está chorando e assim, necessitando de cuidados.

O equipamento possui uma central de monitoramento que deve ficar perto da criança, este é encarregado de identificar o choro da mesma e enviar um sinal via Wi-Fi para a pulseira que deverá ficar no pulso do pai ou da mãe e que irá vibrar e acender um LED indicando que a criança está chorando, além de possuir outro LED sinalizador que indica o status da conexão Wi-Fi.

Este equipamento visa à utilização pelos pais portadores de deficiência auditiva principalmente durante a noite na hora de dormir que é o momento em que há maior preocupação em saber do estado da criança para garantir sua saúde. O mesmo destina-se unicamente ao uso doméstico.

2.1. Características Do Equipamento

A. Para crianças de até 02 anos de idade;
B. Alcance da captação do choro de até 1m sem obstáculos;
C. Alcance da comunicação entre a central de monitoramento e a pulseira de até 50m sem obstáculos;
D. Alertas visual e tátil através da pulseira conforme a detecção do choro.

2.2. Requisitos

Para a central de monitoramento:

A. Alimentação interna: 2 baterias alcalinas 6LF22 9V$_{DC}$;
B. Proximidade do bebê (mínimo 1m);

Para a pulseira:

A. Alimentação interna: bateria CR123A 3V$_{DC}$ 700 mAh;
B. Proximidade da central de monitoramento (mínimo 50m sem obstáculos);
C. Contato com a pele para a percepção da vibração.

3. MÓDULOS DO PROJETO

O projeto é dividido em dois módulos a central de monitoramento e a pulseira.

3.1. Central de Monitoramento

O circuito eletrônico da central de monitoramento é dividido em 5 partes, são elas:

- Fonte de alimentação (2 baterias alcalinas 6LF22 9V$_{DC}$ com regulador de tensão de 5 V$_{DC}$);
- PCB de aquisição de áudio;
- Processamento matemático (placa Arduino Uno);
- Placa de acoplamento;
- Ponto de acesso e transmissão de sinal Wi-Fi (placa de desenvolvimento e prototipagem WeMos D1).

O circuito de alimentação é simplesmente constituído de duas baterias alcalinas 6LF22 9V$_{DC}$ com regulador de tensão L7805CV estabilizado com capacitores de eliminação de ruído que regula a tensão de 9 V$_{DC}$ da bateria para 5 V$_{DC}$ e alimentando todo o circuito da placa de aquisição de áudio.

A placa de aquisição de áudio é uma placa de circuito impresso que tem como finalidade a aquisição dos sinais das ondas sonoras provenientes do ambiente onde se encontra a central de monitoramento, a pré-amplificação destes sinais, a filtragem das faixas de áudio captadas a fim de eliminar sinais de frequências muito altas e muito baixas que implicam negativamente no tratamento matemático do projeto e a amplificação do sinal a fim de deixa-lo em condições aceitáveis ao processamento pela placa Arduino Uno.

Simplificando o processo de tratamento do sinal amplificado, ao receber o sinal analógico inserido no pino digital 08 da placa Arduino Uno, o mesmo realiza equações matemáticas e gera dados que são usados para medir a frequência aproximada deste sinal e caso o valor gerado por esta frequência esteja dentro da

faixa de frequência semelhante à do choro dos bebês 7 vezes a cada 10 valores gerados, o microcontrolador envia para a placa WeMos D1 um sinal de interrupção que fará com que a mesma envie para a pulseira o comando necessário para sua ativação. Para maiores informações sobre o sistema vide capítulo 7.

A placa de acoplamento tem como função transmitir o sinal elétrico de interrupção que é necessário para a placa WeMos D1 enviar o comando de ativação da pulseira.

A configuração do ponto de acesso Wi-Fi e a transmissão do comando de ativação utilizando o protocolo de rede UDP são feitas pela placa de desenvolvimento e prototipagem WeMos D1, nela está alocado o módulo ESP8266-SP12E que contem um chip Wi-Fi com um núcleo de processamento Tensilica L106, nele foi gravado um firmware que contem as instruções para tratar a informação recebida e fazer a transmissão do comando para a ativação da pulseira através da tecnologia Wi-Fi esta informação é um sinal de radiofrequência do padrão IEEE 802.11g e está codificada no formato de uma *string de caracteres.* Vide capítulo 4 para informações sobre o módulo ESP8266.

3.2 Pulseira

O circuito eletrônico da pulseira é constituído de uma placa de desenvolvimento e prototipagem WeMos D1 Mini, uma bateria CR123A, dois LEDs (um na cor verde e um na cor vermelha), um botão para cessar o alarme e um motor *Vibracall*.

A placa WeMos D1 Mini presente na pulseira recebe a informação enviada da central de monitoramento através de comunicação Wi-Fi usando o protocolo de rede UDP, um determinado código é enviado pela Central de Monitoramento, este é um código especifico que determina a ação a ser adotada pelo sistema.

O núcleo de processamento do módulo ESP8266 contido na placa WeMos D1 Mini é o Tensilica L106 este juntamente com o programa nele gravado é encarregado de tratar a informação recebida e tornar o nível lógico de seus terminais

de I/O para alto ou baixo, assim acionando os atuadores de alarme, ou seja, o LED vermelho e o motor *Vibracall*.

4. MÓDULO ESP8266

4.1. Introdução ao Módulo ESP8266

O módulo ESP8266 é uma solução integrada Wi-Fi que tem como principais características o uso eficiente de energia, design compacto e desempenho confiável na indústria.

Com recursos completos de redes Wi-Fi, ele pode trabalhar tanto como um *shield* de um *host* microcontrolado, quanto como uma plataforma independente. Quando está desempenhando o papel de processador de aplicação independente no dispositivo em que é que é vinculado, ele é capaz de inicializar diretamente a partir de um *flash* externo além de possuir um rápido *sleep/Wake* e *cache* integrado para melhorar o desempenho do sistema e para minimizar os requisitos de memória.

Pode ser integrado em qualquer projeto baseado em microcontrolador com conectividade simples através de interface UART ou CPU AHB Bridge, servindo como um adaptador Wi-Fi permitindo acesso a redes *wireless*.

Com sua capacidade de processamento *on-board* ele permite interação a sensores e outros dispositivos de aplicações especificas, através de conexão em seus GPIOs com o mínimo de desenvolvimento *up-front* e carregamento durante a execução. Com seu elevado grau de interação *on-chip*, que inclui o interruptor de antena balun, conversores de gestão de energia e exigência mínima de circuitos externos incluindo módulo *front-end* ele é designado para ocupar a área mínima na PCB em que é aplicado (ESPRESSIF SYSTEMS INC., 2013).

Inicialmente o módulo ESP8266 era utilizado apenas como escravo de um *host* MCU, mas a fabricante Espressif Systems Inc. lançou em sua comunidade *open-source* de desenvolvedores, o firmware NodeMCU que inicialmente permitia a programação diretamente no microcontrolador presente na plataforma utilizando

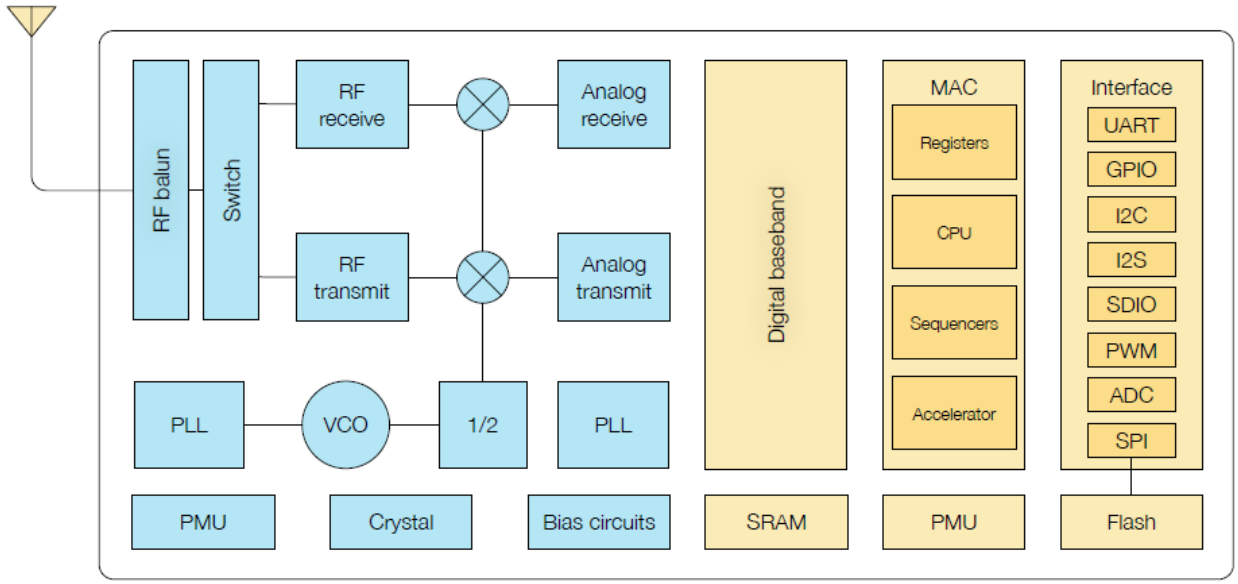
linguagem de programação Lua e a plataforma de programação ESPlorer. Para gravar o firmware NodeMCU é necessário o uso de um utilitário como o ESPtool que é facilmente encontrado nas comunidades de desenvolvedores ESP8266.

Em consequência ao avanço do constante uso do módulo pelos desenvolvedores e hobbystas IoT a Espressif lançou também a SDK que permitia programar o módulo usando a linguagem de programação Arduino e o software Arduino IDE.

Figura 2: Módulo ESP8266EX.

Fonte: Elaborada pelo autor.

Figura 3: Diagrama Funcional do Módulo ESP8266EX.



Fonte: ESP8266EX Datasheet Versão 1.0, p 7, Figura: 3-1. Functional Block Diagram, Espressif Inc., 2016.

Tabela 1: Principais Especificações Técnicas do Módulo ESP8266EX.

Categorias	Itens	Parameters
Wi-Fi	Padrões	FCC/CE/TELEC/SRRC
	Protocolos	802.11 b/g/n/e/i
	Ranjo de Frequência	2.4 G □ 2.5 G (2400M □ 2483.5M)
	Tx	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antena	PCB Trace, Externa, Conector IPEX, Chip Cerâmico
Hardware	CPU	Tensilica L106 32-bit micro controller
	Interface de Periféricos	UART/SDIO/SPI/I2C/I2S/IR Controle Remoto
		GPIO/ADC/PWM
	Tensão de Trabalho	3.0 V ~ 3.6 V
	Corrente de Trabalho	Valor Médio: 80 mA
	Limite de Temperatora de operação	-40°C ~ 125°C
	Limite de Temperatura de Armazenamento	-40°C ~ 125°C
	Dimensões	QFN32-pin (5 mm x 5 mm)
	Interface Externa	-
Software	Modos Wi-Fi	station/softAP/SoftAP+station
	Segurança	WPA/WPA2
	Criptografia	WEP/TKIP/AES
	Upgrade de Firmware	UART Download / OTA (via network)
	Desenvolvimento de Software	Supporta Cloud Server Development / Firmware e SDK para programação on-chip rápida
	Protocolos de Rede	IPv4, TCP/UDP/HTTP/FTP
	Configuração do Usuário	Comandos AT, Cloud Server, Android/iOS App

Fonte: ESP8266EX Datasheet Versão 1.0, p 3, Tabela: 1-1 Main Technical Specifications Espressif Inc., 2016.

4.1.1. Aplicações do módulo ESP8266EX

- Eletrodomésticos;
- Automação Residencial;
- Controle Industrial Wireless;
- Babás Eletrônicas;
- Câmeras IP;
- Redes Sensoriais;
- Plugues e Lâmpadas Inteligentes;
- Redes Mesh;
- Eletrônicos Wearable;
- Dispositivos de Posicionamento Wi-Fi;
- Crachás de Segurança Eletrônica.
- Sistemas de Sinalização Wi-Fi;

5. PCBs DO PROJETO

5.1 PCB de Aquisição Áudio

Para fins de explicação a placa foi dividida em blocos que representam os estágios pelos quais o sinal de áudio deve passar até chegar à placa Arduino Uno, Estes blocos são:

- Captação do áudio;
- Pré-amplificação do sinal de áudio;
- Filtragem do sinal de áudio;
- Amplificação do sinal de áudio.

5.1.1 Captação do Áudio

O elemento captador de áudio é o microfone tipo cápsula de eletreto que é constituído basicamente de um capacitor com membrana de eletreto e um transistor de efeito de campo de junção (JFET), o eletreto é um material que apresenta cargas elétricas permanentes, que se alteram quando sofre deformação mecânica esta alteração exerce sobre o capacitor uma alteração na indução elétrica de sua armadura fixa que está conectada ao *gate* do JFET este que amplifica o sinal e distribui em seu terminal de saída, este tipo de microfone já tem uma pré-amplificação própria, mas está apenas não é suficiente para gerar um sinal com amplitude suficiente para nossa aplicação, então, foi necessária a aplicação de um circuito amplificador de pré-amplificação de áudio semelhante aos encontrados nos microfones condensadores.

5.1.2 Pré-amplificação do Sinal de Áudio

A pré-amplificação do sinal de áudio é realizada a partir de um circuito de pré-amplificação simples. O microfone de eletreto aplica à base de um transistor de uso geral BC549 polarizado para amplificação, um sinal de áudio através de um capacitor de acoplamento, este transistor irá pré-amplificar o sinal à um valor de amplitude elevada o suficiente para atender a necessidades do amplificador operacional LM324 mesmo com a redução da amplitude do sinal de entrada devido a distancia entre a fonte do som e o microfone, além disso este circuito torna possível a calibração da sensibilidade do microfone a partir da variação da resistência de um trimpot.

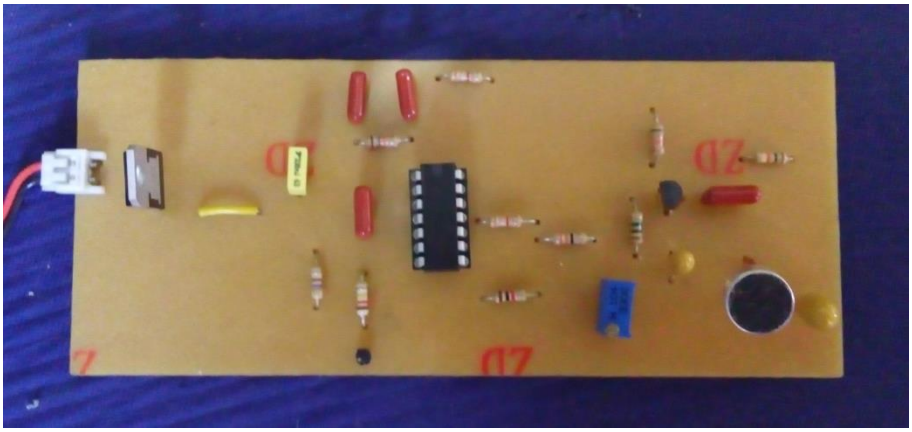
5.1.3 Filtragem do Sinal de Áudio

Após a captação e pré-amplificação do sinal de áudio o mesmo é aplicado a um filtro de frequência do tipo passa-banda ativo de topologia *Sallen-Key*. Este circuito de filtragem é basicamente composto por um amplificador operacional de alimentação simples LM324, resistores e capacitores para sua polarização. Esta filtragem tem como objetivo filtrar o sinal de entrada eliminando ruídos externos provenientes do ambiente além dos sinais de áudio de frequências muito altas e muito baixas, (maiores que 5kHz e menores que 300Hz) que poderiam interferir negativamente na leitura realizada pela placa Arduíno Uno.

5.1.4 Amplificação do Sinal de Áudio

Posterior ao filtro de frequência está o circuito de amplificação do sinal que tem como finalidade aumentar a amplitude do sinal que será inserido na entrada da placa de chaveamento e posteriormente no pino digital 8 da placa de desenvolvimento e prototipagem Arduino Uno a níveis de trabalho seguros para a mesma. O circuito de amplificação do sinal consiste em um amplificador operacional LM324 usando a configuração não inversora de ganho máximo (aproximadamente120 vezes).

Figura 4: PCB de Aquisição de Áudio.

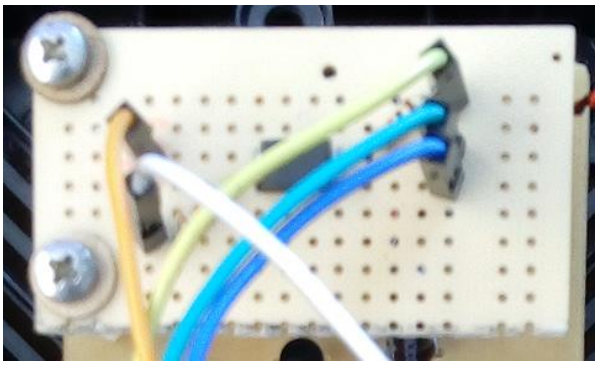


Fonte: Elaborada pelo autor.

5.2 Placa de Acoplamento

Como a placa Arduino Uno trabalha com um valor de tensão de 5 VDC em seus pinos de I/O e a placa WeMos D1 só suporta até 3,6 V DC de entrada, foi necessário o uso de um opto-acoplador PC123 para fazer esta ligação. O opto-acoplador utiliza luz de LED para chavear um fototransistor que permitirá a ligação entre seu coletor e emissor mudando então o nível lógico inserido no pino D3/D15 da placa WeMos D1 de alto para baixo, gerando, portanto uma interrupção externa do tipo *RISING* necessária para o envio do comando de ativação da pulseira.

Figura 5: Placa de Acoplamento.

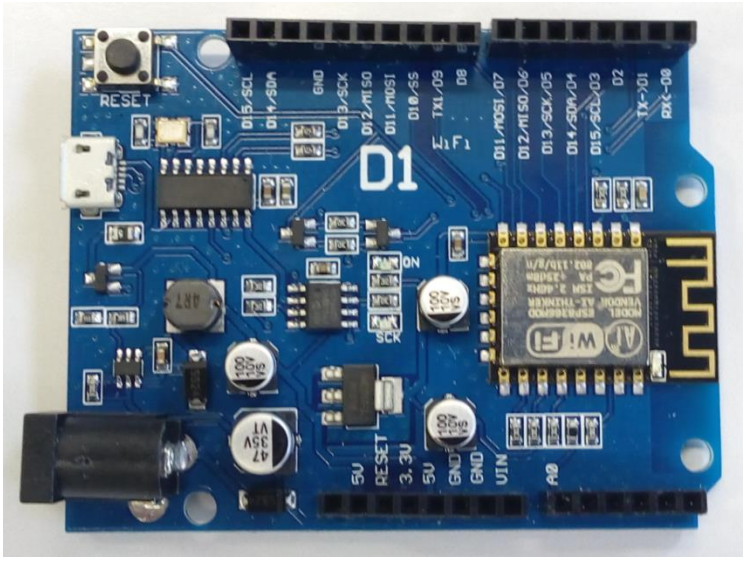


Fonte: Elaborada pelo autor

5.3 Placas de Desenvolvimento e Prototipagem WeMos D1 e D1 Mini

As placas de desenvolvimento e prototipagem WeMos D1 e D1 Mini são plataformas de programação de sistemas embarcados desenvolvidas para adaptar o módulo ESP8266 a um ambiente amigável a ser usado por estudantes, hobbystas e desenvolvedores iniciantes em *IoT*. A placa WeMos D1 traz ao módulo esp8266 uma solução em conexão ao computador através de conexão USB além do ambiente muito similar a placa Arduino Uno em termos de aparência, alimentação e *pinout*.

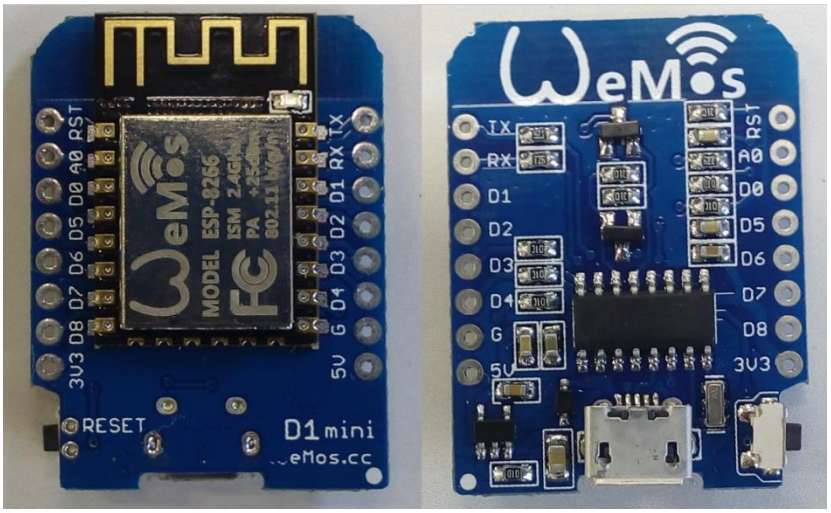
Figura 6: Placa WeMos D1.



Fonte: Elaborada pelo autor.

Já a placa WeMos D1 Mini é uma versão da placa WeMos D1 com tamanho reduzido e adaptação ao acoplamento em *proto-boards* o que soluciona a dificuldade em montagem de protótipos.

Figura: 7: Placa WeMos D1 Mini.



Fonte: Elaborada pelo autor.

Ambas as placas possuem o *firmware* NodeMCU e podem ter seus softwares desenvolvidos utilizando a linguagem Arduino e o software Arduino IDE.

O único empecilho destas placas é sua característica de tensão de trabalho que é de 3,3 VDC diferente das placas Arduino que trabalham com tensão de 5 VDC o que não permite a ligação direta aos *shields* usados pelas mesmas.

A escolha das placas foi decidida devido a acessibilidade por se tratar de um produto de baixo custo, a possibilidade de programação em linguagem Arduino esta que foi aprendida durante o curso e ao acesso a informação disponível em comunidades *open-source*.

Uma das dificuldades era a necessidade de um sistema de comunicação *wireless* microcontrolado compacto o suficiente para ser implantado dentro de uma pulseira, este problema foi solucionado com o módulo ESP8266 e mais além com a placa WeMos D1 Mini que às todas nossas expectativas.

5.4. Placa de Desenvolvimento e Prototipagem Arduino

5.4.1 Introdução à placa Arduino

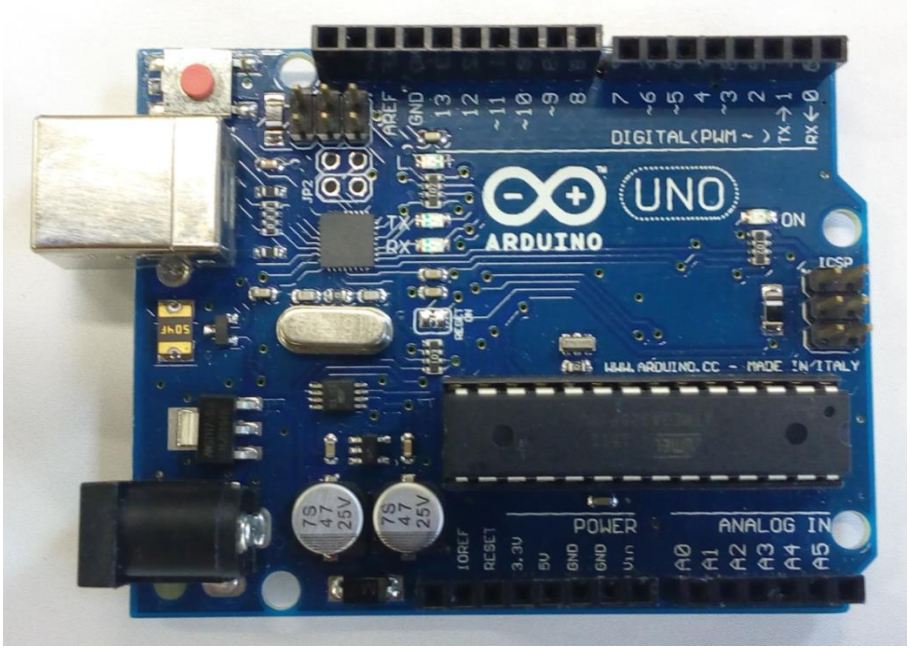
Criada no Ivrea Interaction Design Institute a placa Arduino é uma plataforma de prototipagem de código aberto baseada em *easy-to-use hardware* e *software* desenvolvida com o principal intuito de criar uma ferramenta de prototipagem de fácil programação destinada à estudantes sem alto conhecimento de engenharia de software ou eletrônica.

Após sua grande aceitação nas comunidades de programadores e desenvolvedores *IoT*, Arduino tem se tornado cada vez mais usada em projetos acadêmicos de todos os segmentos como por exemplo: automação residencial e industrial, segurança, *wereable*, impressão 3D, telecomunicações, entretenimento, lazer, etc.

Com seu microcontrolador ATmega, a plataforma Arduino atua como um computador com capacidade de processamento reduzida, realizando cálculos e processamento de dados a fim de ler informações advindas de sensores e ativar atuadores conectados em seus pinos de I/O. “Para fazer isso é usada a linguagem de programação Arduino (baseada em *Wiring*), e o software Arduino (Arduino IDE), baseado em *Processing*.”

“Todas as placas Arduino são completamente *open-source*, capacitando os usuários para construí-los de forma independente e, eventualmente, adaptá-los às suas necessidades específicas. O *software* também é *open-source*, e está crescendo através das contribuições dos usuários em todo o mundo” (ARDUINO, [2016]).

Figura 8: Placa Arduino Uno.



Fonte: Elaborada pelo autor.

5.4.2 Especificações Técnicas da placa Arduino Uno

Tabela 2: Principais especificações técnicas da placa Arduino Uno.

Microcontrolador	ATmega328P
Tensão de Trabalho	5 VDC
Alimentação (limite)	6-20 VDC
Pinos Digitais de I/O	14
de Pinos Digitais de I/O	6
Pinos Analógicos de I/O	6
Pinos de Interrupção Externa	2
Corrente Máxima fornecida por I/O	20 mA DC
Memória Flash	32 KB (0,5 usados para o bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidade de Clock	Velocidade de Clock
Comprimento	68,6 mm
Largura	53,4 mm

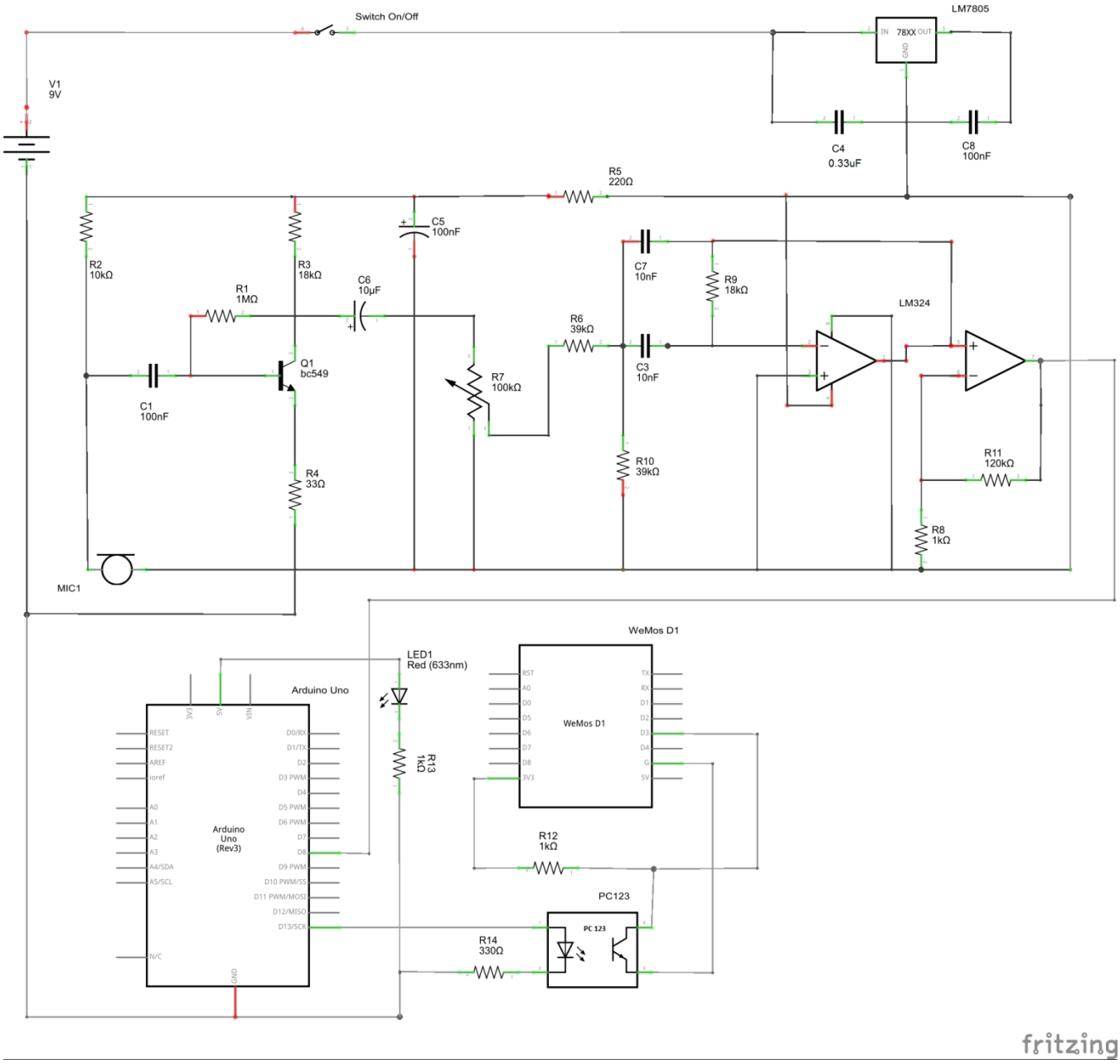
Fonte: ARDUINO, disponível em: < https://www.arduino.cc/en/Main/ArduinoBoardUno>, acesso em 01/06/2016.

6. DIAGRAMAS ELÉTRICOS

Os diagramas elétricos são representações esquemáticas das ligações dos circuitos elétricos do sistema.

6.1. Diagrama Elétrico da Central de Monitoramento

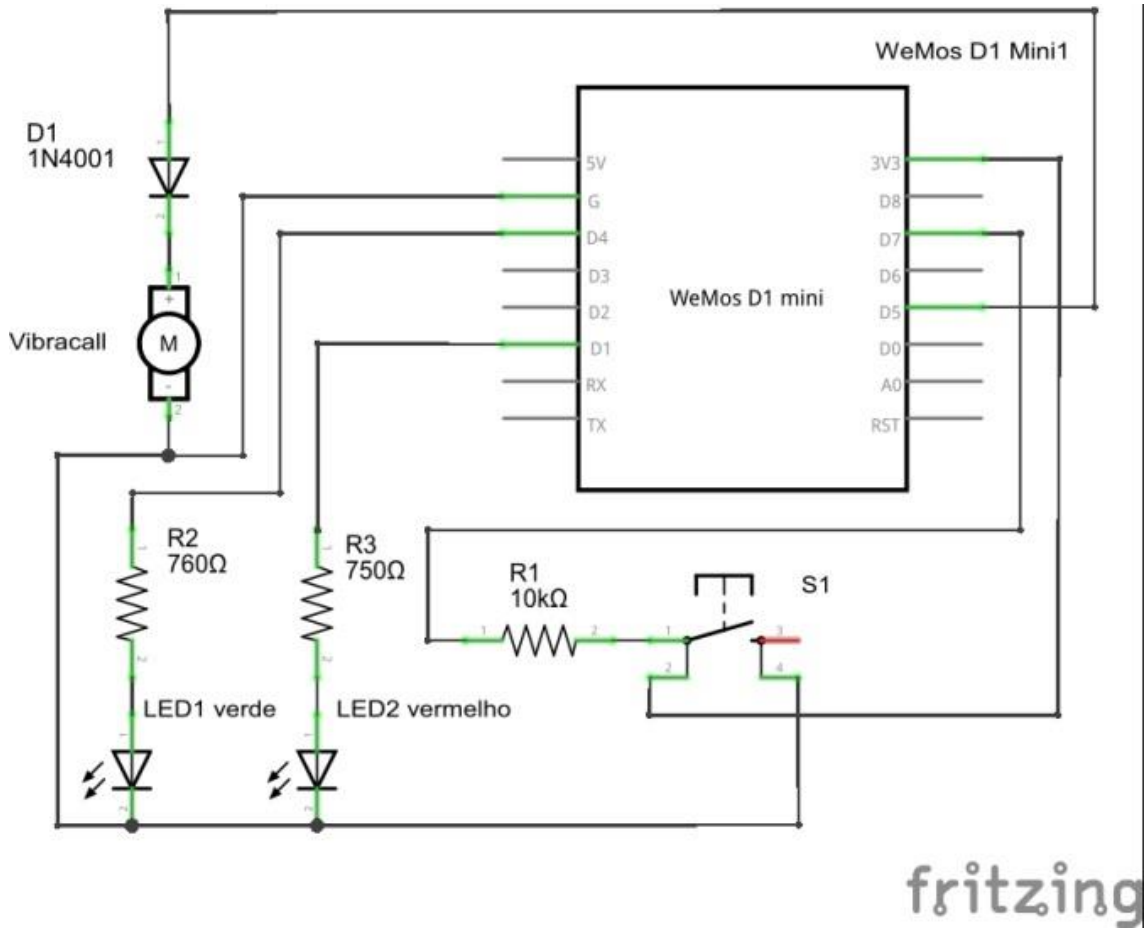
Figura 9: Diagrama elétrico da Central de Monitoramento.



Fonte: Elaborada pelo autor.

6.2. Diagrama Elétrico da Pulseira

Figura 10: Diagrama elétrico da Pulseira.



Fonte: Elaborada pelo autor.

OBS: As placas de desenvolvimento e prototipagem estão meramente representadas pelos seus microcontroladores discriminando o circuito completo da plataforma. Suas alimentações também não estão representadas, pois não há simbolização dos terminais disponível para esta representação no software utilizado para a criação destes diagramas elétricos.

7. FIRMWARES DO SISTEMA

Os firmwares de todas as placas de desenvolvimento e prototipagem do projeto foram desenvolvidos em linguagem Arduíno, utilizando o software Arduino IDE versão 1.6.8 para criação e compilação. Para a programação das placas WeMos D1 e D1 Mini, foi implementado ao gerenciador de placas do software Arduino IDE o pacote esp8266 por ESP8266 Community versão 2.1.0 que contém uma série de configurações de placas de desenvolvimento e prototipagem construídas a partir do módulo ESP8266.

As bibliotecas utilizadas para o desenvolvimento são todas de domínio público e disponíveis livremente para utilização, são elas:

<WiFiUDP.h>

<ESP8266WiFi.h>

<FreqMeasure.h>

7.1. Programas do sistema

7.1.1. Código do Firmware da Placa Arduino Uno (Central de Monitoramento)

```
/*
  Firmware para a central do projeto "Detector de Choro Para Pais Deficientes Auditivos"
  utilizando a biblioteca licenciada abaixo:


  http://www.pjrc.com/teensy/td_libs_FreqMeasure.html
```

```
  This example code is in the public domain.

*/


//Colocar valor em 1 para permitir debug pela porta serial, colocar qualquer outro valor para desabilitar

#define __debug  1


#include <FreqMeasure.h>

byte inBand = 0;                // Variável a guardar a quantidade de vezes em que o valor de frequência medido estava entre 1.2kHz e 5kHz

byte resetCountTime = 0;        // Variável utilizada para o reset de contagem da variável inBand

float frequencyRes = 0;         // Variável a guardar o valor resultante para a freqência inserida

const byte pinInterrupt = 13;   // Variável a guardar o valor do pino de interrupção


// Declaração de functions

void sendInterruptRequest();


// Configuração de I/Os, inicialização da medição de freqência e de comunicação serial

void setup()

{

  #if (__debug == 1)

  Serial.begin(57600);

  #endif
```

```
  FreqMeasure.begin();
  pinMode(pinInterrupt, OUTPUT);
  digitalWrite(pinInterrupt, LOW);
}

double sum = 0; // Variável a guardar o valor da soma dos intervalos de tempos dos períodos de borda de subida e borda de descida
int count = 0;  // Variável usada para fazer a divisão para retornar a média dos valores gerados

void loop()
{
  // Se houver sinal de frequência disponível lê o valor resultante, o guarda na variável sum, soma este com o próximo valor lido e adiciona 1 a variável count
  if (FreqMeasure.available())
  {
    sum = sum + FreqMeasure.read();
    count ++;
    // Se count for maior que 30 então, transforma o valor de 32 bits resultante da soma de sum e o divide pelo valor de count para obter uma média depois zera os valores de sum, count e soma 1 a variavel resetCountTime
    if (count > 30)
    {
      frequencyRes = FreqMeasure.countToFrequency(sum / count);
      #if (__debug == 1)
      Serial.println(frequencyRes);
      #endif
```

```
      sum = 0;
      count = 0;
      resetCountTime++;

      #if (__debug == 1)
      Serial.print("Time to reset the cumulative frequency results: ");
      Serial.print(resetCountTime);
      Serial.println("");
      #endif

      if (resetCountTime >= 10)
      {
        resetCountTime = 0;
        inBand = 0;
      }
      else
      {
        sendInterruptRequest();
      }
    }
  }
}
// Loop de envio de sinal de interrupção
```

```
void sendInterruptRequest()
{
  digitalWrite(13, LOW);

  if (frequencyRes >= 1200 && frequencyRes <= 5000)
  {
    inBand++;
    #if (__debug == 1)
    Serial.print(inBand);
    #endif

    if (inBand > 7)
    {
      digitalWrite(pinInterrupt, HIGH);
      delay(50);
      digitalWrite(pinInterrupt, LOW);
      delay(5000);
      inBand = 0;
    }
  }

}
```

7.1.2. Código do Firmware da Placa WeMos D1 (Central de Monitoramento)

```
/*
  Firmware para a central do projeto "Detector de Choro Para Pais Deficientes Auditivos"
  utilizando as bibliotecas licenciadas abaixo:

  Copyright (c) 2015, Majenko Technologies
  All rights reserved.

  Redistribution and use in source and binary forms, with or without modification,
  are permitted provided that the following conditions are met:

* * Redistributions of source code must retain the above copyright notice, this
    list of conditions and the following disclaimer.

* * Redistributions in binary form must reproduce the above copyright notice, this
    list of conditions and the following disclaimer in the documentation and/or
    other materials provided with the distribution.

* * Neither the name of Majenko Technologies nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

      THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
      "AS IS" AND
```

*/

```
#include <ESP8266WiFi.h>
#include <WiFiUDP.h>



    //Colocar valor em 1 para permitir debug pela porta serial, colocar qualquer outro valor para desabilitar
#define __debug  1
```

```
// Variáveis de IO

byte pinIntImput  = 5;  // pino conectado ao pino de interrupção do arduíno


// Variáveis WiFi

const char *ssid = "Central";         // ID da rede

const char *password = "Warframe";   // Senha da rede


// Variáveis UDP

WiFiUDP UDP;


const char *IpPulseira = "192.168.4.2";

unsigned int localPort = 8888;


boolean udpConnected = false;

        char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  // Buffer para guardar pacote
        encaminhado

// Variáveis de comandos

char warning[3] = "on"; // String de comando enviado para ativar o alarme da pulseira

boolean sendState = false;


// Declaração de functions

void interrupt_ISR();

void sendRequest();

void sendCommand();

void DecodePacket();
```

```
void setup()
{
  #if (__debug == 1)
    Serial.begin(115200);
    Serial.println();
    Serial.print("Configuring access point...");
    Serial.println();
  #endif



  // Configura o pino de interrupção
  pinMode(pinIntImput, INPUT);


        WiFi.softAP(ssid, password); // Se quiser pode-se remover a senha e deixar a rede aberta
        apagando "password"


  IPAddress myIP = WiFi.softAPIP();


  #if (__debug == 1)
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  #endif


  // Define parâmetros UDP
```

```
  udpConnected = connectUDP();

  // Inicializa UDP
  UDP.begin(localPort);

  // Interrupção para envio do comando de acionamento da pulseira
  attachInterrupt(pinIntImput, interrupt_ISR, FALLING);
}

void loop()
{
  // Solicita envio de comando de ativação da pulseira
  sendRequest();
}

// Loop de interrupção para envio do comando de alarme da pulseira
void interrupt_ISR()
{
  sendState = true;

  #if (__debug == 1)
  Serial.print("Button pressed");
  #endif
}
```

```
// Loop de requisição de envio de comando de ativação da pulseira
void sendRequest()
{
  if (sendState == true)
  {
    sendCommand();
    sendState = false;
  }
}

// Loop para enviar comandos de ativação do alarme da pulseira.
void sendCommand()
{
  UDP.beginPacket(IpPulseira, localPort);
  UDP.write(warning);
  UDP.endPacket();

  #if (__debug == 1)
  Serial.println("");
  Serial.println("Command has been sent");
  #endif

  delay(5000);
```

```
}

        // Estabelece uma conexão UDP e retorna "true" se for bem sucedida ou "false" se não for
        bem sucedida
boolean connectUDP()
{
  boolean state = false;

  #if (__debug == 1)
  Serial.println("");
  Serial.println("Connecting to UDP");
  #endif

  if (UDP.begin(localPort) == 1)
  {
    #if (__debug == 1)
    Serial.println("Connection successful");
    state = true;
    #endif
  }
  else
  {
    #if (__debug == 1)
    Serial.println("Connection failed");
    return state;
```

```
    #endif
  }
}
```

7.1.3. Código do Firmware da Placa WeMos D1 Mini (Pulseira)

```
/*
Firmware para a central do projeto "Detector de Choro Para Pais Deficientes Auditivos"
utilizando as bibliotecas licenciadas abaixo:

   Copyright (c) 2015, Majenko Technologies
   All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
   are permitted provided that the following conditions are met:

 * * Redistributions of source code must retain the above copyright notice, this
     list of conditions and the following disclaimer.

* * Redistributions in binary form must reproduce the above copyright notice, this
     list of conditions and the following disclaimer in the documentation and/or
     other materials provided with the distribution.
```

```
#include <ESP8266WiFi.h>

#include <WiFiUDP.h>
```

```
//Colocar valor em 1 para permitir debug pela porta serial, colocar qualquer outro valor para desabilitar

#define __debug  1


// Variáveis de IO

byte pinIntButton   = 13; // D7 pino de interrupção conectado ao botão de interrupção do alarme

byte pinAlarmMotor  = 14; // D5 pino conectado ao motor vibracall

byte pinAlarmLed    = 5;  // D1 pino conectado ao LED de alarme

byte pinWifiLed     = 2;  // D4 pino conectado ao LED de indicação do Wi-Fi


// Variáveis e constantes Wi-Fi

const char* ssid = "Central";

const char* password = "Warframe";

boolean wifiConnected = false;


// Variáveis de comandos

        char warning[3] = "on";             // comando para ativação dos atuadores de alarme


#if (__debug == 1)

char alarmOn[10] = "LED is On";         // String para indicar que o arlarme está ativado

char alarmOff[11] = "LED is Off";       // String para indicar que o alarme está desativado

#endif


boolean activateQuery = false;
```

```
// variáveis e constantes UDP

WiFiUDP UDP;

const char *IpCentral = "192.168.4.1";

unsigned int localPort = 8888;

boolean udpConnected = false;

char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; // Buffer para guardar o pacote encaminhado



// Declaração de functions

void BlinkWiFiConnected();

void BlinkWiFiDisconnected();

void DecodePacket();

void intStopAlarm_ISR();

void activateRequest();

void activateAlarm();


void setup()

{

  // Inicializa comunicação serial


  #if (__debug == 1)

  Serial.begin(115200);

  #endif
```

```
  // Inicializa os pinos de IO
  pinMode(pinAlarmMotor, OUTPUT);
  pinMode(pinWifiLed, OUTPUT);
  pinMode(pinAlarmLed, OUTPUT);
  pinMode(pinIntButton, INPUT);

  // Inicializa conexão WiFi
  wifiConnected = connectWifi();

  // Inicializa UDP
  UDP.begin(localPort);

  // Configuração da interrupção de parada do alarme
  attachInterrupt(pinIntButton, intStopAlarm_ISR, FALLING);

  // Procede apenas se a conexão for bem sucedida
  if (wifiConnected)
  {
    udpConnected = connectUDP();
  }
}

// Loop principal de repetição contínua
void loop()
```

```
{
  activateRequest();

  if (WiFi.status() == 3)
  {
    BlinkWiFiConnected();
  }
  else
  {
    BlinkWiFiDisconnected();
    connectWifi();
  }
  // Checa se a conexão Wi-Fi foi bem sucedida
  if (wifiConnected)
  {
    if (udpConnected)
    {
      activateRequest();
      // Se tiver algum pacote disponivel ele é lido
      int packetSize = UDP.parsePacket();
      if (packetSize)
      {

        #if (__debug == 1)
```

```
Serial.println("");
Serial.print("Received packet of size ");
Serial.println(packetSize);
Serial.print("From ");
IPAddress remote = UDP.remoteIP();
for (int i = 0; i < 4; i++)
{
  Serial.print(remote[i], DEC);
  if (i < 3)
  {
    Serial.print(".");
  }
}
Serial.print(", port ");
Serial.println(UDP.remotePort());
#endif

// Lê o pacote recebido
int len = UDP.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
if (len > 0)
{
  packetBuffer[len] = 0;
}
```

```
        #if (__debug == 1)
        Serial.println("Contents: ");
        Serial.println(packetBuffer);
        #endif



        delay(30);
 // Alterna o estado do pino de ativação dos atuadores da pulseira proporcionalmente ao comando do pacote recebido
        DecodePacket();
      }
    }
  }
}

// Loop para piscar o LED de status do WiFi de maneira a representar que há conexão
void BlinkWiFiConnected()
{
  digitalWrite(pinWifiLed, HIGH);
  delay(100);
  digitalWrite(pinWifiLed, LOW);
  delay(3000);
}

// Loop para piscar o LED de status do WiFi de maneira a representar que não há conexão
```

```
void BlinkWiFiDisconnected()
{
  digitalWrite(pinWifiLed, HIGH);
  delay(250);
  digitalWrite(pinWifiLed, LOW);
  delay(250);
}

// Loop para decodificar o pacote de dados recebido e tomar a ação referente a ele
void DecodePacket()
{
  if (packetBuffer[0] == warning[0])
  {
    if (packetBuffer[1] == warning[1])
    {
      activateQuery = true;
    }
  }
}

// Loop de interrupção para parar o alarme
void intStopAlarm_ISR()
{
  digitalWrite(pinAlarmMotor, LOW);
```

```
  digitalWrite(pinAlarmLed, LOW);


  #if (__debug == 1)
  Serial.print(ledStatusOff);
  #endif


  packetBuffer[UDP_TX_PACKET_MAX_SIZE] = UDP.read();
}



// Loop de requisição de ativação dos atuadores de alarme da pulseira
void activateRequest()
{
  if (activateQuery == true)
  {
    activateAlarm();
    activateQuery = false;
  }
}


// Loop de ativação dos atuadores de alarme da pulseira
void activateAlarm()
{
  digitalWrite(pinAlarmMotor, HIGH);
```

```
  digitalWrite(pinAlarmLed, HIGH);


  #if (__debug == 1)
  Serial.print(ledStatusOn);
  #endif
}

// Estabelece uma conexão UDP e retorna "true" se for bem sucedida ou "false" se não for bem sucedida
boolean connectUDP()
{
  boolean state = false;


  #if (__debug == 1)
  Serial.println("");
  Serial.println("Connecting to UDP");
  #endif


  if (UDP.begin(localPort) == 1)
  {


    #if (__debug == 1)
    Serial.println("Connection successful");
    state = true;
    #endif
```

```
  }
  else
  {

    #if (__debug == 1)
    Serial.println("Connection failed");
    #endif
  }

  return state;
}
// Estabelece uma conexão WiFi e retorna "true" se for bem sucedida ou "false" se não for bem sucedida
boolean connectWifi()
{
  boolean state = true;

  WiFi.begin(ssid, password);

  #if (__debug == 1)
  Serial.println("");
  Serial.println("Connecting to WiFi");
  #endif

  // Aguarda pela conexão
```

```
  while (WiFi.status() != WL_CONNECTED)
  {
    BlinkWiFiDisconnected();
    break;
  }

// Quando a conexão WiFi for bem sucedida escreve na serial "Connected to " e o endereço IP que foi estabelecido
  if (WiFi.status() == WL_CONNECTED)
  {

    #if (__debug == 1)
    Serial.println("");
    Serial.print("Connected to ");
    Serial.println(ssid);
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    #endif
  }
  return state;
}
```

7.2. Fluxogramas do Sistema

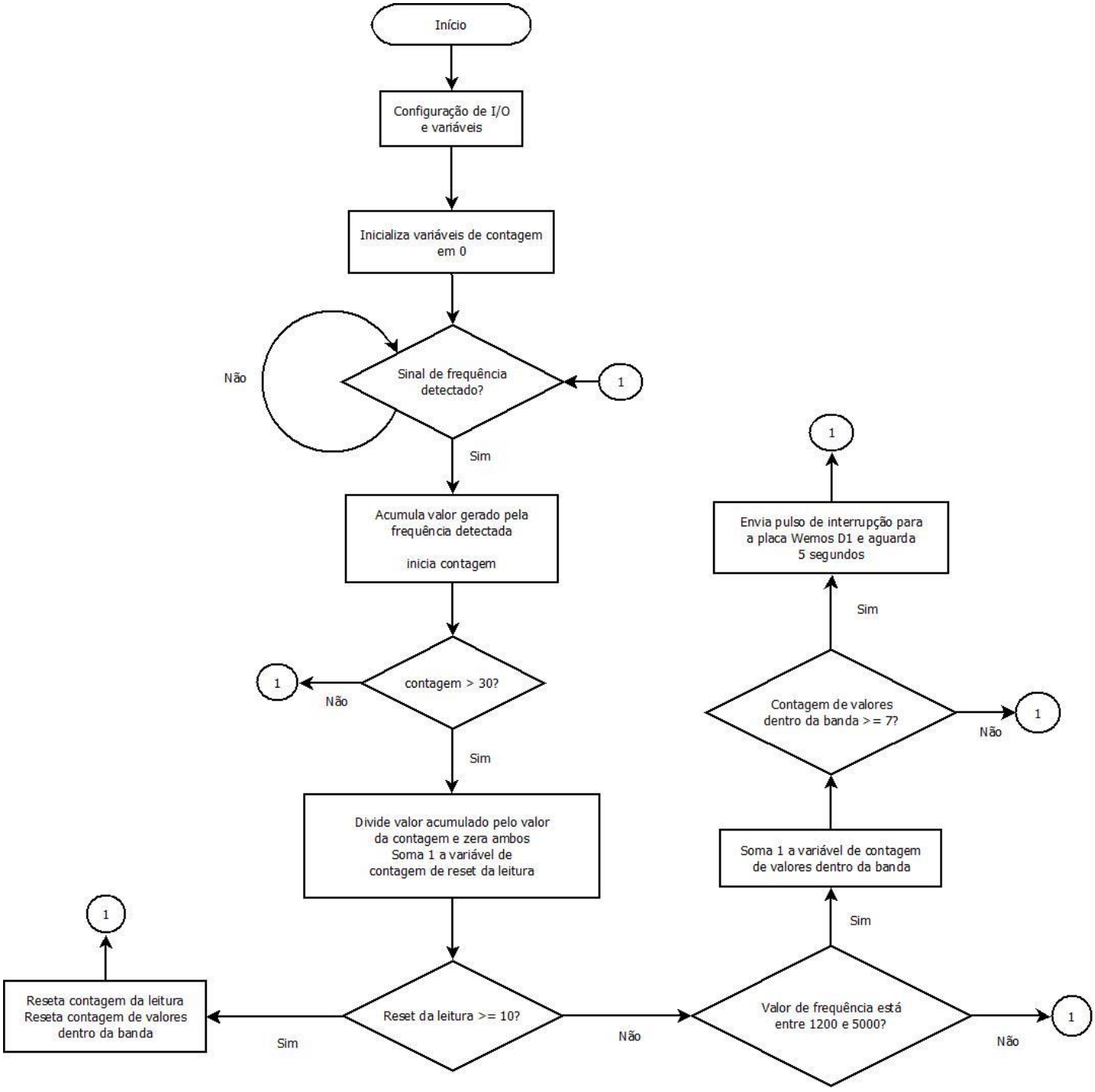
Os fluxogramas representam simbolicamente o funcionamento do sistema, estes foram divididos em partes representando os firmwares das placas de desenvolvimento e prototipagem.

7.2.1. Fluxograma da placa Arduino Uno (Central de Monitoramento)

O fluxograma da placa Arduino Uno representa o funcionamento do programa de cálculo matemático do sistema, ele gera valores proporcionais ao nível de frequência proveniente dos sinais elétricos advindos da PCB de aquisição de áudio, estes valores são usados para diferenciar os níveis de frequência a fim de distinguir sons com faixa de frequência muito distintas das faixas de frequência em que está o choro de um bebê para então enviar um pulso elétrico de nível lógico alto que irá gerar uma interrupção na placa WeMos D1 através da placa de acoplamento.

Através de testes em laboratório, foi constatado que os sons de choros de bebês geram neste programa valores de 32 bits que estão entre 1200 e 5000, então foi implementada uma limitação onde a geração do sinal de interrupção somente aconteceria se dentre 10 valores obtidos pelo menos 7 estejam entre 1200 e 5000.

Figura 11: Fluxograma do programa principal da placa Arduino Uno.

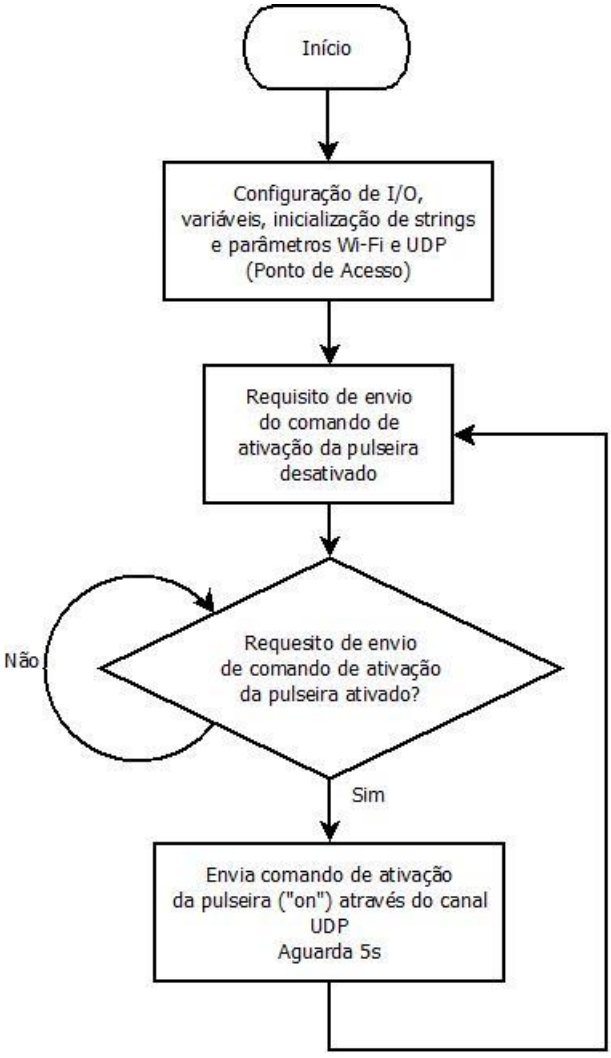


Fonte: Elaborada pelo autor.

7.2.2. Fluxograma da placa WeMos D1 (Central de Monitoramento)

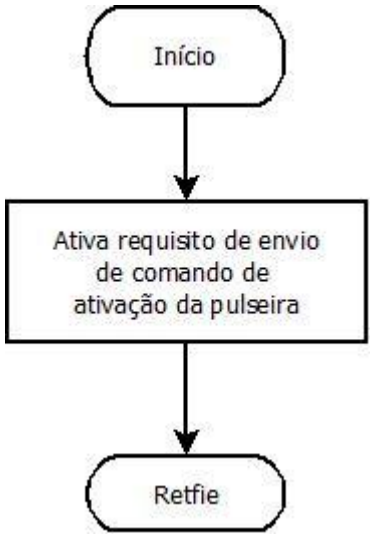
O fluxograma da placa WeMos D1 representa simbolicamente o funcionamento do programa de ponto de acesso Wi-Fi, este é encarregado de configurar um ponto de acesso com segurança WPA2 que envia uma *string* de caracteres utilizando o protocolo de rede UDP quando recebe uma interrupção externa proveniente da placa de acoplamento ligada à placa Arduino, após o recebimento e decodificação dessa mensagem a pulseira ativa seu alarme avisando ao usuário que a criança está chorando.

Figura 12: Fluxograma do programa principal da placa WeMos D1.



Fonte: Elaborada pelo autor.

Figura 13: Fluxograma da interrupção da placa WeMos D1.



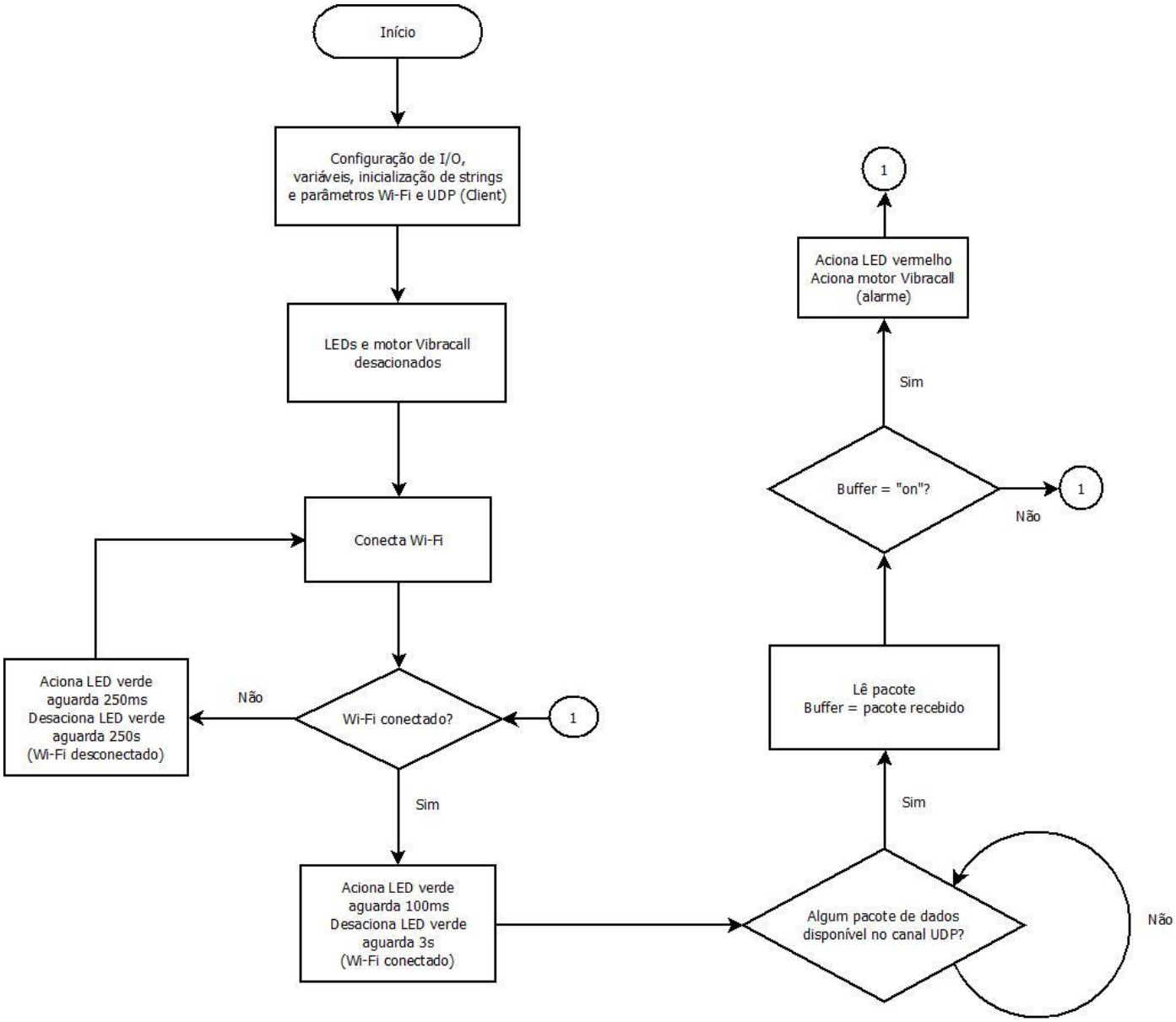
Fonte: Elaborada pelo autor

7.2.3. Fluxograma da placa WeMos D1 Mini (Pulseira)

O fluxograma da placa WeMos D1 Mini representa simbolicamente o funcionamento do programa de ativação da pulseira, este tem como função tomar a decisão de conectar-se ao ponto de acesso da Central de Monitoramento e aguardar pelo comando de alarme. Quando a pulseira está desconectada ela pisca seu LED verde rapidamente, já quando está conectada, a mesma pisca seu LED verde devagar. Sempre que a pulseira está desconectada ela dá inicio a tentativa de conexão com a central e indica a falta de conexão através do piscar do LED verde. Quando conectada, ao receber o comando de ativação, a pulseira ativa seu LED vermelho e o motor Vibracall, avisando ao usuário que o bebê está chorando.

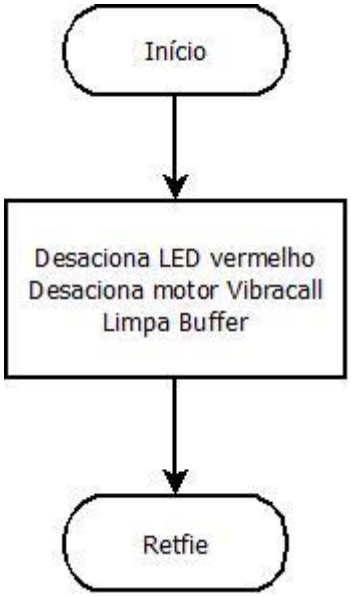
Ao pressionar o botão da pulseira é gerada uma interrupção externa que faz com que o LED vermelho e o motor Vibracall sejam desacionados, fazendo assim com que o alarme seja interrompido.

Figura 14: Fluxograma do programa principal da placa WeMos D1 Mini.



Fonte: Elaborada pelo autor.

Figura 15: Fluxograma da interrupção do programa da placa WeMos D1 Mini.



Fonte: Elaborada pelo autor.

8. CRONOGRAMA

Tabela 3: Cronograma de Desenvolvimento do Projeto.

Tópico	Ago	Set	Out	Nov	Dez	Fev	MAr	Abr	MAi	Jun	Jul
Escolha do Projeto											
Documentação											
Pesquisa de Componentes											
Pesquisa de Software											
Desenvolvimento da Monografia											
Desenvolvimento da Apresentação Visual											
Hardware											
Montagem de Protótipos											
Montagem da PCB de Aquisição de Áudio											
Montagem da Pulseira											
Montagem da Central de Monitoramento											
Software											
Desenvolvimento do Software da Pulseira											
Desenvolvimento do Software da Central de Monitoramento											
Testes											
Testes de Validação de Desenvolvimento											
Testes de Validação											
Apresentação Final											

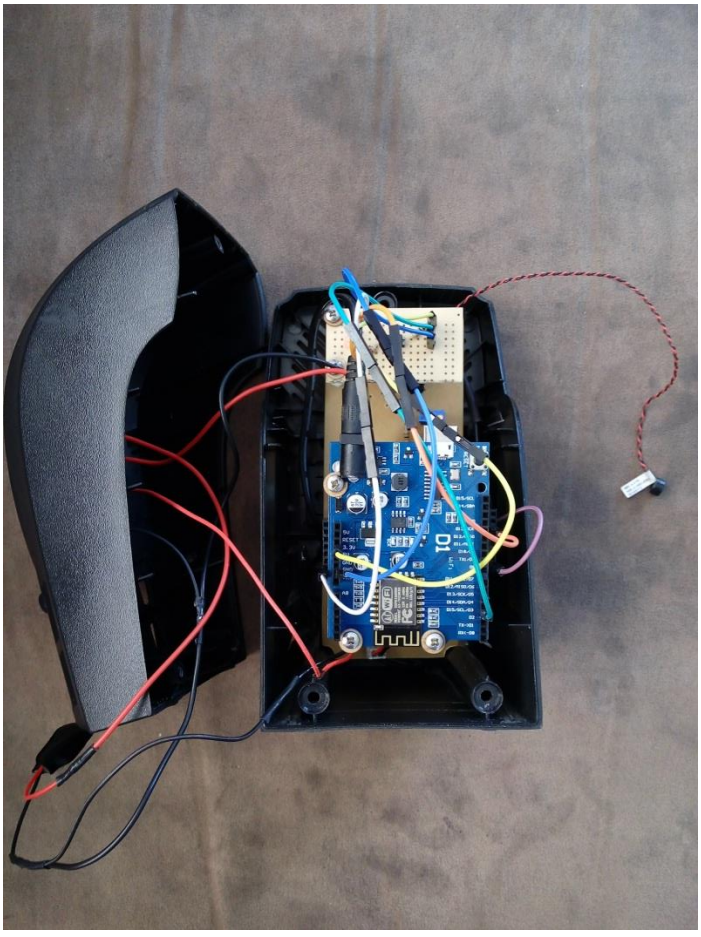
Fonte: Elaborada pelo autor.

9. VALIDAÇÃO DO PROJETO

Durante os testes realizados foram coletados os seguintes dados:

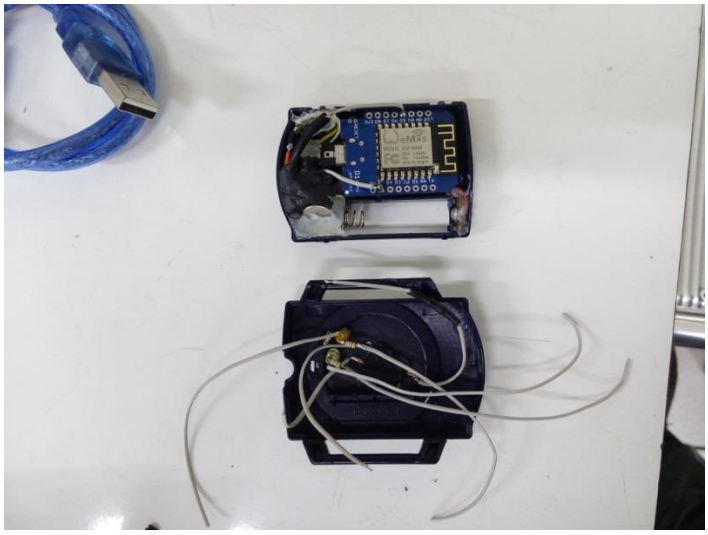
- A taxa de envio e recebimento do sinal de ativação da pulseira é de 100% se respeitado o limite de distancia;
- Sons com ranjo de frequência próximo ao do choro ativam o aparelho;
- Equipamento pode sofrer interferência se próximo de aparelhos elétricos de alta potência;

Figura 16: Circuito da Central de Monitoramento.



Fonte: Elaborada pelo autor.

Figura 17: Circuito da Pulseira.



Fonte: Elaborada pelo autor

Figura 18: Projeto Final.

Fonte: Elaborada pelo autor.

10. REFERÊNCIAS BIBLIOGRÁFICAS

MOREIRA, Paula Pfeifer. **SER MÃE E SURDA.** 2010. Disponível em: <http://cronicasdasurdez.com/ser-mae-e-surda/>. Acesso em: 20 abr. 2016.

THE METROPOLITAN MUSEUM OF ART (New York). **Radio Nurse.** 2000–2016. Disponível em: <http://www.metmuseum.org/toah/works-of-art/2000.600.14/>. Acesso em: 21 abr. 2016.

ESPRESSIF INC. **ESP8266 Datasheet.** 2016. Disponível em < https://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 09 mai. 2016.

ARDUINO. **Arduino UNO & Genuino UNO:** Technical specs. 2016. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 20 jun. 2016.

GITHUB INC. (San Francisco). **Arduino core for ESP8266 WiFi chip.** Disponível em: <https://github.com/esp8266/Arduino>. Acesso em: 16 maio 2016.

WEMOS. **D1 mini:** Technical specs. Disponível em: <http://www.wemos.cc/Products/d1_mini.html>. Acesso em: 22 maio 2016.

WEMOS. **D1 mini:** Technical specs. Disponível em: <http://www.wemos.cc/Products/d1.html>. Acesso em: 22 maio 2016.

BRAGA, Newton C.. **Como funcionam os microfones (ART616).** Disponível em: <http://www.newtoncbraga.com.br/index.php/como-funciona/4446-art616>. Acesso em: 26 maio 2016.

PJRC. **FreqMeasure Library.** 2015. Disponível em: <http://www.pjrc.com/teensy/td_libs_FreqMeasure.html>. Acesso em: 21 maio 2016.

ZESKIND, Philip Sanford. **Impacto do choro do bebê de risco no desenvolvimento psicossocial.** 2007. Disponível em: <http://www.enciclopedia-crianca.com/choro/segundo-especialistas/impacto-do-choro-do-bebe-de-risco-no-desenvolvimento-psicossocial>. Acesso em: 25 maio 2016.

11. ANEXOS

ANEXO A - LAB 7 Filtros ativos – FILTRO PASSA-FAIXA ATIVO DE SEGUNDA ORDEM

Montagem 3

FILTRO PASSA-FAIXA ATIVO DE SEGUNDA ORDEM

O objetivo desta montagem é verificar a funcionalidade de um filtro ativo passa faixa de segunda ordem observando a sua freqüência central, freqüências de corte inferior e superior, ganho na faixa de passagem e fator de qualidade Q.

A figura 5 mostra uma configuração de circuito usando amplificador operacional com múltipla realimentação para implementação de um filtro ativo passa faixa de segunda ordem.

Monte o circuito da figura 5. Aplique um sinal senoidal na entrada e observe os sinais de entrada e saída simultaneamente com o osciloscópio. Inicie o experimento aplicando um sinal na entrada com amplitude constante e com freqüência inferior a freqüência central do filtro (calculada teoricamente). Varie apenas a freqüência do sinal de entrada, mantendo a amplitude constante, e observe em que freqüência ocorre o ganho máximo correspondente a freqüência central do filtro. Diminua a freqüência e observe a freqüência de corte inferior (-3dB). Aumente a freqüência e observe a freqüência de corte superior (-3dB). A partir destes valores determine o fator de qualidade Q experimental. Compare os valores medidos com os valores teóricos esperados. Considere R1= 10 kΩ, R2=680 kΩ e C=1nF.

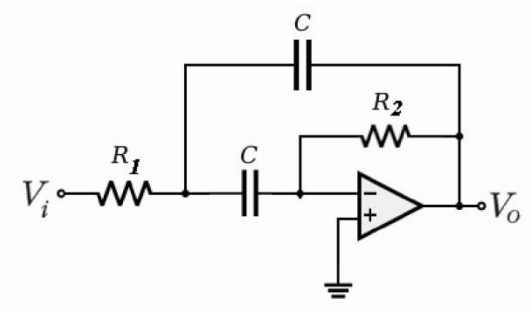


Figura 5 – Filtro Ativo Passa Faixa

Função de transferência H(s) e equações para o Filtro Passa Faixa Topologia Multipla Realimentação (Figura 5)

$$H(s) = -\frac{s\frac{1}{R_1 C}}{s^2 + \frac{2}{R_2 C}s + \frac{1}{R_1 R_2 C^2}}$$

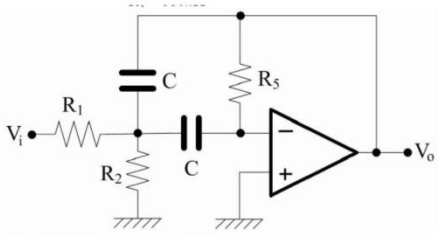
ANEXO B - LAB 7 Filtros ativos – Equações para o projeto de um filtro passa faixa

$$Y_1 = \frac{1}{R_1} = \frac{1}{4k7} \qquad Y_3 = sC_3$$

$$Y_2 = \frac{1}{R_2} = \frac{1}{270\Omega} \qquad Y_4 = sC_4$$

$$Y_5 = \frac{1}{R_5} = \frac{1}{680k\Omega} \qquad C_3 = C_4 = 10nF$$

Equações para o projeto de um filtro passa faixa



$f_0 = \frac{1}{2\pi C}\sqrt{\left[\frac{1}{R_5}\frac{(R_1+R_2)}{R_1R_2}\right]}$	$\frac{\omega_0}{Q} = \frac{2}{R_5C}$
$Q = \frac{1}{2}\sqrt{\frac{R_5(R_1+R_2)}{R_1R_2}}$	$Q = \frac{\omega_0 R_5 C}{2}$
$\omega_0^2 = \frac{(R_1+R_2)}{R_1R_2R_5C^2}$	$\Delta\omega = \frac{\omega_0}{Q} = \frac{2}{R_5C}$

Figura 7 – *FILTRO ATIVO PASSA-FAIXA*

Substituindo-se as respectivas admitancias na expressão geral encontra-se a função de transferência H(s) para o filtro passa faixa de segunda ordem em função de seus componentes (resistores e capacitores).

$$\frac{V_o}{V_i} = -\frac{\frac{1}{R_1C}s}{s^2 + \frac{2}{R_5C}s + \frac{(R_1+R_2)}{R_1R_2R_5C^2}}$$

$$H(s) = -\frac{R_5}{2R_1}\frac{s\frac{2}{R_5C}}{s^2 + \frac{2}{R_5C}s + \frac{R_1+R_2}{R_1R_2R_5C^2}}$$

Associando-se os coeficientes da equação geral com os respectivos coeficientes da função de transferência H(s) para o filtro passa faixa de segunda ordem mostrado na figura 7 obtem-se os parâmetros do respectivo filtro, conforme apresentados a seguir.

seguir.

$$H(s) = H_o\frac{s\frac{\omega_o}{Q}}{s^2 + s\frac{\omega_o}{Q} + \omega_0^2} \qquad H_0 = -\frac{R_5}{2R_1}$$