UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROPOSTA DE SISTEMA EMBARCADO PARA AUXÍLIO E MONITORAMENTO DO IDOSO

Autor: Alexandre Moretti Bernardo

Orientador: Prof. Dr. Evandro Luís Linhari Rodrigues

São Carlos

2015

Universidade de São Paulo
Escola de Engenharia de São Carlos

ALEXANDRE MORETTI BERNARDO

PROPOSTA DE SISTEMA EMBARCADO PARA AUXÍLIO E MONITORAMENTO DO IDOSO

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
eletrônica

ORIENTADOR: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos
2015

Bernardo, Alexandre Moretti
B518p Proposta de sistema embarcado para auxilio e monitoramento do idoso / Alexandre Moretti Bernardo; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2015.

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

1. Auxilio ao idoso. 2. Sistemas embarcados. 3. Linux. 4. Transdutores. 5. Processamento de imagens. 6. Intel Galileo. I. Titulo.

FOLHA DE APROVAÇÃO

Nome: Alexandre Moretti Bernardo

Título:"Proposta de sistema embarcado para auxílio e monitoramento do idoso"

Trabalho de Conclusão de Curso defendido e aprovado em 17 / 11 / 2015,

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

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

Prof. Associado Ivan Nunes da Silva - (SEL/EESC/USP)

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

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Dr. José Carlos de Melo Vieira Júnior

Dedicatória

Dedico o trabalho aqui apresentado a toda minha família, que se empenharam em me auxiliar em todas as etapas de minha vida, especialmente na minha permanência em São Carlos, me incentivando a buscar e lutar por meus sonhos e objetivos.

Alexandre Moretti Bernardo

Agradecimentos

Agradeço primeiramente a Deus por ser essencial em minha vida, estar presente em todos os momentos, auxiliado e concedido forças para superar todos os obstáculos e desafios.

Ao meu pai Luiz Donizeti Bernardo, minha mãe Vitalina Moretti Bernardo e meu irmão Samuel Moretti Bernardo pelo apoio em todos os momentos de dificuldade e pela compreensão sobre as privações que a dedicação à Universidade requer.

Ao professor Orientador Evandro, pelo apoio ao desenvolvimento do projeto e pelo seu empenho em extrair o melhor de seus alunos, incentivando e aconselhando-nos durante toda a graduação.

Aos meus amigos que foram fundamentais durante todo esse período, estudamos para muitas provas, fizemos muitos trabalhos, enfrentamos grandes desafios e nos divertimos juntos. Passamos por momentos de dificuldade e de alegria que fortaleceu nossa amizade.

Aos amigos do *Warthog Robotics*, por todas as madrugadas passadas em claro nas vésperas de campeonatos, por ensinar o verdadeiro significado de engenharia, determinação e paixão por fazer os projetos funcionarem. Também agradeço a este grupo e a seus integrantes pela grande experiência em liderança da divisão de eletrônica, contribuindo para minha formação.

Aos professores que se importaram com o aprendizado de seus alunos durante a graduação. Agradeço também aos meus professores da Escola Estadual Júlia Calhau Rodrigues que apesar de todas as limitações e dificuldades de uma escola pública, se emprenharam em incentivar os sonhos de seus alunos e deram o seu melhor para permitir que atingíssemos nossos objetivos.

Agradeço também a todos os funcionários do Departamento de Engenharia Elétrica e Computação (SEL/EESC/USP), em especial às secretárias de graduação, Jussara Ramos Zoia e Aura Aparecido Brizolar, aos técnicos Rosenberg Júlio da Silva, Petrússio Gonçalves da Silva.

À Safira e ao José por contribuírem pelos necessários momentos de descontração.

Por fim, a todos que contribuíram, direta ou indiretamente, para a realização deste trabalho.

“E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria” (I Coríntios 13:2)

“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê” (Arthur Schopenhauer)

“Ter problemas na vida é inevitável, ser derrotado por eles é opcional”.
(Roger Crawford)

Resumo

O envelhecimento com saúde é o triunfo do desenvolvimento de um país. Garantir que essa parcela da população tenha os devidos cuidados sem que sejam privados de sua liberdade e independência torna-se um desafio. Para satisfazer essa necessidade, este trabalho aborda a criação de três mecanismos para identificar eventuais riscos ou problemas de saúde do idoso. O primeiro sistema é uma pulseira munida de um acelerômetro e barômetro capaz de detectar queda. O segundo sistema é composto por um sensor que monitora possíveis vazamentos de gás na residência do idoso. O terceiro sistema é composto por uma caixa de remédios com alarme para informar os horários da ingestão dos medicamentos com compartimentos separados e identificados por LEDs (*Light Emitting Diode*). Os três sistemas propostos são gerenciados pela placa de desenvolvimento Intel Galileo que em caso de detecção de riscos como queda ou vazamentos de gás envia uma mensagem para o celular do responsável. Neste trabalho é apresentado as características dos sistemas propostos bem como suas vantagens e limitações. Para a proposta, a placa Intel Galileo acentuada limitação de recursos como processamento e memória RAM para aplicações em visão computacional, mas mostrou-se como uma solução bastante versátil, sendo possível encontrar soluções que não necessitem de alto poder de processamento e memória. Todos os transdutores utilizados para aquisição dos dados dos sistemas mencionados, foram caracterizados. O módulo barômetro, utilizado para medir variação de altura apresentou erros acima dos valores indicados pelo fabricante, mas que foram minimizados por meio de algoritmos de filtro. Os módulos acelerômetros apresentaram precisão acima do necessário para detectar impacto mecânico. O módulo sensor de gás apresentou funcionamento conforme especificado pelo fabricante quando submetido a incidência de gás, simulando um possível vazamento. Apesar das limitações os protótipos foram concluídos e apresentaram as características requeridas, isto é, gerenciar a ingestão de medicamentos, indicar situações de risco como vazamento de gás ou queda do idoso, bem como envio de mensagens ao responsável e transmissão de vídeo do local do acidente.

Palavras chave: Auxílio ao idoso, sistemas embarcados, Intel Galileo, processamento de imagens, Linux, transdutores.

Abstract

Healthy ageing serves as a reference of how developed a country is. Guaranteeing the wellbeing of the elderly while maintaining their independence is a challenge. In order to address this issue, this work presents the development of three devices that aim to prevent accidents that are most common among older people; a bracelet which uses an accelerometer and a barometer to identify if the user is falling down; a system to be installed at home to detect gas leakages; and a medicine dispenser with separate compartments identified with LEDs (Light Emitting Diode) that informs the user about the time to take the pills with the use of an alarm. These three devices have the feature of sending messages to mobile phones if there is a risk situation so that the elder is helped as soon as possible. They are implemented using the Intel Galileo microcontroller board, which has low processing speed for computational vision applications, but is versatile enough to meet the requirements to make this project possible. This text describes the development of the mentioned equipments along with their limitations, it also includes the characteristics of the transducers used for acquiring data. The barometer used, which measures changes in height, did not meet its specifications, showing larger variations, but filter algorithms were used to solve this problem. The accelerometer is more precise than necessary to detect mechanic impact, and, as such, is the most suitable sensor for this application. The gas sensor behaved as specified by the manufacturer in the presence of gas, simulating a leakage. In spite of its limitations, the prototypes do what they are supposed to: managing medicine intake, detecting gas leakage and falls, sending messages and video transmission in case of risk situations.

Keywords: Assistance to the elderly, Embedded systems, Intel Galileo, IP cameras, Linux, transducers.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de abreviações

3D – Três dimensões

ADC – *Analog-to-digital converter* (Conversor analógico para digital)

ARM – Família de microcontroladores (*Advanced RISC Machines*)

CAD - *Computer-Aided Design*

CGI - *Common Gateway Interface*

CI – Circuito Integrado

CSD - *Circuit Switched Data*

DLL - *Dynamic Linked Library*

DNS - *Domain Name System*

EEPROM – *Electrically Erasable Programmable Read-Only Memory*

fps – frames por segundo

GND – *Ground*

GPIO - *General-Purpose Input/Output*

GPRS - *General packet radio service*

Grub - *GRand Unified Bootloader*

HLR - *Home Location Register*

I/O – *Input/Output*

I²C – Protocolo de comunicação (*Inter-Integrated Circuit*)

IBGE – Instituto Brasileiro de Geografia e Estatística

ICSP - *In Circuit Serial Programming*

IDE - *Integrated Development Environment*

IoT – *Internet of Things* (Internet das coisas)

IP - *Internet Protocol*

IPL - *Image Processing Library*

LAN - *local area network*

LCD – *Liquid crystal display*

LED - *Light Emitting Diode* (Diodo Emissor de Luz)

MEMS - *Micro-Electro-Mechanical Systems*

Mjpeg - *Motion JPEG*

MS - *Mobile Station*

MT - *Mobile Terminal*

OpenCV - *Open Source Computer Vision*

Opkg - *Open Package Management*

PC – *Personal Computer*

PCIe - *Peripheral Component Interconnect Express*

ppm – partes por milhão

PWM - *Pulse Width Modulation*

RAM - *Random-Access Memory*

RF – Radio Frequência

RISC - *Reduced Instruction Set Computing*

RTC - *Real-Time Clock*

RTOS - *Real-Time Operating System*

Rx - Recebimento

SDIO - *Secure Digital Input Output*

SDK - *Software Development Kit*

SMS - *Short Message Service*

SO – Sistema Operacional

SPI - *Serial Peripheral Interface*

SRAM - *Static Random-Access Memory*

SSH - *Secure Shell*

SUS – Sistema Único de Saúde

TCP - *Transmission Control Protocol*

Tx - Transmissão

UART - *Universal Asynchronous Receiver/Transmitter*

UDP - *User Datagram Protocol*

UEFI - *Unified Extensible Firmware Interface*

URL - *Uniform Resource Identifier*

USB - *Universal Serial Bus*

USSD - *Unstructured Supplementary Service Data*

VGS – Tensão entre Gate e Source do transistor

VLR - *Visitor Location Register*

WLAN - *Wireless Local Area Network*

XLP – *Extreme Low Power*

Sumário

1. Introdução

A industrialização trouxe consigo diversos avanços tecnológicos na área da saúde. A introdução da informática e o surgimento de aparelhos modernos e sofisticados trouxeram benefícios no tratamento e diagnósticos de doenças [1].

O termo designado para tecnologias que contribuem para a área da saúde e bem-estar é “E-saúde” (do inglês *E-Health*). Este termo é relativamente recente, datando de 1999 [2] e apresenta cerca de 51 definições “exclusivas” [3]. De forma genérica o *E-Health* inclui dispositivos que registram o estado de saúde por meios eletrônicos, telemonitorização, dispositivos que rastreiam as condições do paciente remotamente, aplicativos móveis de saúde e sistemas de informação hospitalar [4].

Nota-se uma crescente busca tecnológica neste setor, incentivada pela possibilidade de desenvolver sistemas que auxiliem a saúde da população [5]. O gráfico da Figura 1 apresenta os investimentos em tecnologias voltadas para saúde no mercado norte americano de 2010 a 2014.

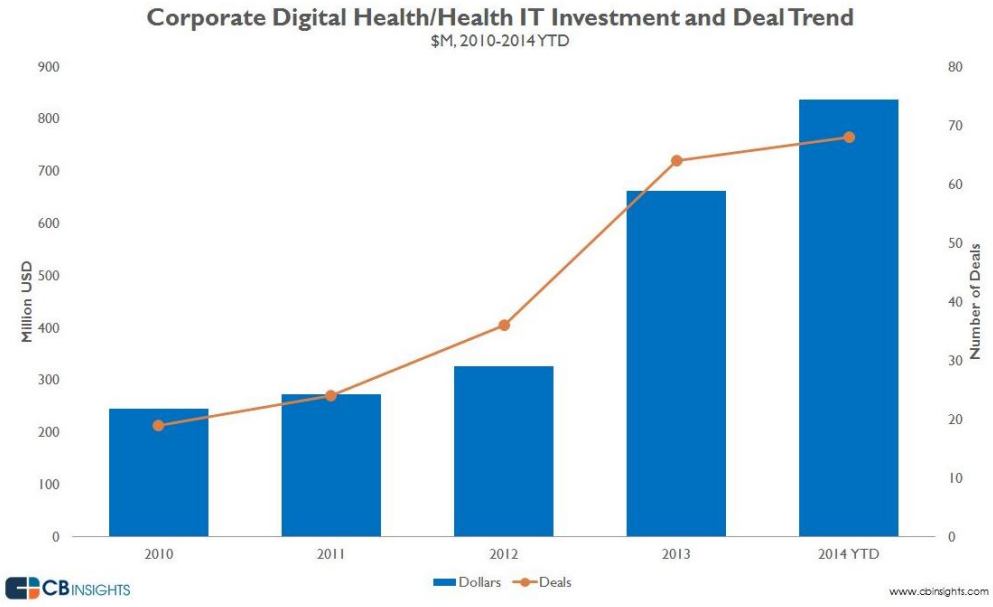


Figura 1: Gráfico do investimento em tecnologias voltadas a área da saúde [5].

Essa recente busca tecnológica tem motivado *startups* e empresas de tecnologia voltadas a área de saúde e *E-Health* que levantaram US$ 2,2 bilhões de dólares no primeiro semestre de 2014, e os investimentos corporativos têm grande parte nesse feito [5]. Além de *startups*, grandes empresas de diversos setores têm investido no desenvolvimento de

tecnologias relacionadas a saúde. A Figura 2 apresenta a classificação de empresas com maior investimento no setor. Quantitativamente se destaca a empresa Google, que em 2015 investiu cerca de U$32,5 milhões em apenas uma empresa (*startup Oscar*) que desenvolveu um aplicativo que permite aos usuários entrem em contato com médicos especialistas a qualquer momento e sem custos [6].

The Most Active Strategic Investors in Digital Health/Health IT Since 2010

Rank	Investor
1	Qualcomm Ventures
2	Merck Global Health Innovation Fund
3	BlueCross BlueShield Venture Partners
3	Google Ventures
5	Intel Capital
6	Ascension Ventures
7	MemorialCare Innovation Fund
8	Johnson & Johnson Development Corporation
8	Dignity Health
8	Kaiser Permanente Ventures
8	Reed Elsevier Ventures
8	GE Ventures
8	TELUS Ventures

CB INSIGHTS

Figura 2: Classificação das empresas que apresentam maior investimento em tecnologia voltada à saúde [5].

Como forma de contribuição tecnológica para o setor, este trabalho apresentará um sistema "*E-Health*" utilizando sistemas operacionais embarcados e transdutores no monitoramento e auxílio à manutenção da saúde do idoso.

O uso de sistemas embarcados permite o processamento de informações de forma dedicada, reduzindo o custo quando comparado a sistemas de proposito mais genérico, ao mesmo tempo em que permite alterações e implementações de forma remota. O uso de transdutores associados ao sistema operacional no sistema embarcado, expande a aplicabilidade da plataforma utilizada. Gera-se, portanto, um projeto facilmente mutável em função das informações adquiridas e inclui o monitoramento de acidentes ao conceito de internet das coisas (IoT).

1.1. Motivação

O aumento da longevidade é um indicador de que todos os setores da sociedade estão se desenvolvendo e melhorando a qualidade de vida da população. A população idosa do Brasil é composta de 26 milhões de pessoas, o que corresponde a 13% da população total. A expectativa de vida média da população brasileira é de 74,6 anos segundo IBGE [7].

Apesar dos avanços, esta parcela da população carece de atenção especial. No Brasil, 70% dos idosos possuem pelo menos uma patologia crônica [8], ou seja, necessitam de tratamento farmacológico e uso regular de medicamentos. A necessidade de grande ingestão de medicamentos, acrescida das limitações ocasionadas pela idade, gera um campo de risco extremamente alto onde os idosos podem ingerir medicamentos errados. Além do grande número de medicamentos, segundo o Sistema Único de Saúde, um terço dos atendimentos por lesões traumáticas nos hospitais do país ocorre com pessoas com mais de 60 anos e cerca de 75% dessas lesões acontecem dentro da própria casa, onde muitas vezes os primeiros atendimentos tardam a ocorrer e agravam a situação do idoso.

Para evitar acidentes como a ingestão indevida de medicamentos, ocasionados por confusões de nomes ou horários, ou para a monitoração de quedas do idoso, o uso de tecnologias apropriadas pode gerar resultados importantes e melhorar a qualidade de vida e a longevidade da população.

1.2. Propostas

Para identificar possíveis quedas é proposto a utilização de uma pulseira com um acelerômetro e barômetro capaz de detectar quedas, nessa situação, aciona-se um alarme por alguns segundos, caso o idoso não desative o aparelho, este enviará a mensagem via SMS ou mensagem de texto por meio do aplicativo WhatsApp ao responsável, evitando assim falsas identificações de quedas. O sistema faz uso de câmeras com algoritmos de detecção e identificação da localização do idoso. Caso seja identificado, por meio dos transdutores, que o idoso caiu, uma central de processamento de dados envia uma mensagem ao celular de um familiar ou responsável. Este por sua vez pode acessar as câmeras e verificar se realmente ocorreu uma situação de emergência ou se houve apenas um falso alarme. Este mesmo sistema pode ser utilizado para fazer toda a vigilância e segurança da casa, monitorando ações suspeitas.

Para auxiliar o idoso a ingerir o medicamento de maneira adequada, é proposto a criação de um sistema de gerenciamento de medicamentos. Esse sistema indica o

momento da ingestão do medicamento e possibilita o envio de mensagem de celular ao farmacêutico ou ao responsável da família quando o medicamento estiver acabando, evitando assim, problemas decorrentes da não ingestão do medicamento.

Para detectar possíveis vazamentos de gás foi utilizado um módulo sensor de gás que deve ser instalado próximo ao fogão do idoso. Caso a válvula de gás seja deixada aberta, o módulo é capaz de enviar um comando a central de processamento (placa Intel Galileo) e este envia uma mensagem para o celular do responsável indicando o possível vazamento de gás.

A placa de desenvolvimento Intel Galileo juntamente com o sistema operacional proporciona a integração de todos os sistemas por meio de conexões via web e acesso remoto. Dessa forma, o familiar pode acompanhar qualquer problema ou suspeita de problema pela internet.

O diagrama do projeto com seus respectivos módulos é apresentado na Figura 3. Para a central de processamento foi utilizado a placa de desenvolvimento Intel Galileo. O sistema detector de queda, as câmeras IPs e o sistema de monitoramento de vazamento de gás utilizam comunicação via *Wireless*. O sistema de gerenciamento de medicamentos é conectado diretamente a central de processamento.

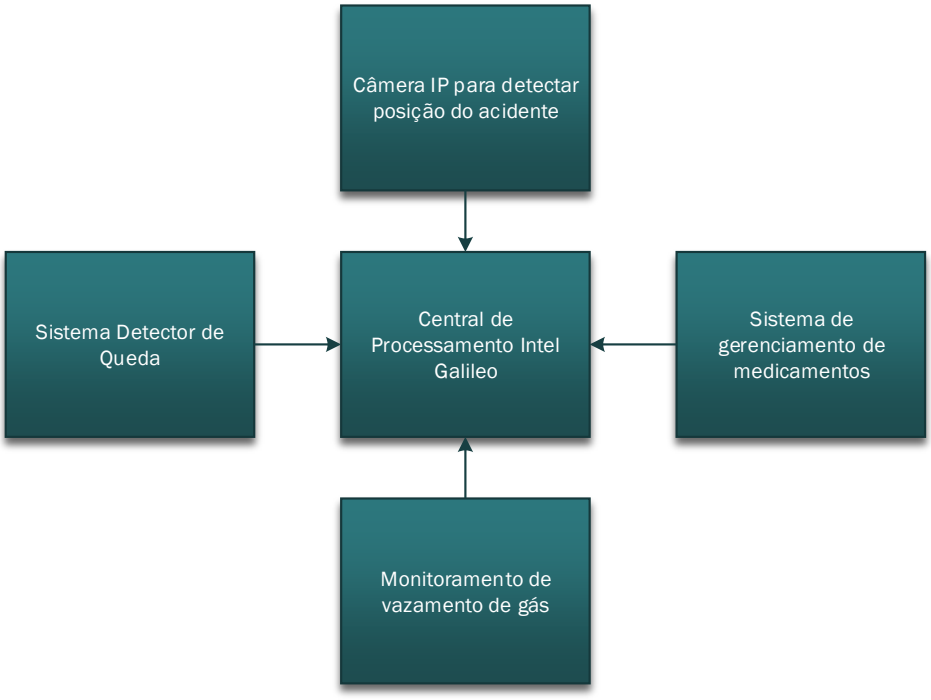


Figura 3: Diagrama de alto nível do projeto

1.3. Objetivos

Este trabalho foi idealizado de forma a cumprir três objetivos principais:

Fomentar o uso tecnologias distintas para o desenvolvimento de um sistema de auxílio a saúde do idoso, apresentando todas as etapas e configurações realizadas oferecendo a base para futuros desenvolvedores desta temática.

Apresentar um sistema tecnológico atual que possa contribuir para a supervisão da qualidade de vida das pessoas, em especial idosos, permitindo que eventuais problemas possam ser detectados automaticamente além de auxiliar em tarefas como a ingestão de remédios e do controle de reserva destes.

Para alcançar os objetivos principais, foram estipulados objetivos secundários para servir de base e direcionamento do projeto principal:

1 – Desenvolver aplicações utilizando a placa de desenvolvimento Intel Galileo;

2 – Apresentar funcionalidades e configurações utilizadas no projeto em forma de tutorial, com a finalidade de contribuir para futuros projetos que se utilizem desse sistema;

3 – Estudar e apresentar uma forma de comunicação entre o Arduino e o sistema operacional embarcado;

5 – Utilizar diferentes plataformas, linguagens de programação, sistemas de comunicação e sensores, bem como a comunicação entre estes;

4 – Apresentar e utilizar dispositivos transdutores, assim como princípios de funcionamento e aplicabilidade destes;

5 – Implementar formas de comunicação entre os módulos e envio de mensagem de texto de alerta ao celular do responsável;

6 – Apresentar a viabilidade técnica de um sistema de indicação do medicamento a ser ingerido;

7 – Realizar testes individuais de funcionamento e posteriormente a integração de todo o projeto culminando em um protótipo funcional;

1.4. Justificativas

O desenvolvimento deste projeto apresenta-se como uma solução tecnológica para auxiliar e monitorar a saúde do idoso. Atualmente existem alguns projetos similares [9, 10], mas que se destinam a um público alvo diferente, ou que não apresentam todas as funções que o sistema proposto oferece, em função do uso de sistemas embarcados, bem como aplicação de conceitos de Internet das Coisas (IoT) [11].

O projeto apresentado em Sistema Detector de Quedas (SDQ) [9] e monitoramento de quedas para pessoas com deficiência motora [10] propõe uma solução de detecção de quedas utilizando acelerômetro e em [9] a implementação com o envio de mensagem por SMS. Diferentemente das soluções apresentadas, este projeto, utiliza além do acelerômetro, um barômetro que permite redundância (por sistemas distintos) na detecção de quedas. Este trabalho também permite o acesso remoto a câmeras IP, agilizando a identificação da gravidade do ocorrido. Além desse sistema, este projeto também abrange a detecção de possíveis vazamento de gás e de gerenciamento da ingestão de medicamentos. Outro diferencial é apresentado por utilizar mensagens de celular por meio do aplicativo WhatsApp, que garante redução de custos, quando comparado ao envio de SMS.

Devido a longevidade alcançada, principalmente nos países desenvolvidos, um sistema de auxílio ao idoso torna-se cada vez mais útil. A solução tecnológica proposta permite que o idoso tenha liberdade em manter seus afazeres em sua casa ao mesmo tempo em que garante tranquilidade ao responsável que será avisado caso o idoso sofra algum acidente ou haja algum potencial risco como vazamento de gás ou falta de medicamento.

A viabilidade deste projeto decorre do atual cenário tecnológico disponível, com sistemas embarcados e sensores que fornecem um campo de desenvolvimento ágil e versátil. Permitem, portanto, a elaboração de protótipos dedicados a soluções de problemas observados no cotidiano.

A proposta deste trabalho é de grande importância, pois apresenta uma aplicação tecnológica original que busca contribuir para saúde da população mais idosa. Este trabalho poderá ser utilizado como base para elaboração de um produto que contribuirá para a manutenção da saúde e integridade física do idoso.

1.5. Organização do Trabalho

Este projeto contempla o uso de diversas ferramentas e componentes detalhados ao longo deste trabalho. Dessa forma, inclui-se neste trabalho todo o processo de aprendizagem, desenvolvimento, escolhas e configurações dos dispositivos e ferramentas utilizados. Para contemplar todas as etapas de projeto, este trabalho apresenta o embasamento teórico, desenvolvimento e resultados de cada módulo. Posteriormente é apresentado o resultado da integração do sistema e a conclusão. Para atingir esse objetivo esse trabalho está disposto em 5 capítulos, incluindo essa introdução, disposto conforme a descrição que segue:

Capítulo 2: Introduz os conceitos e fundamentos teóricos que serviram de base para o desenvolvimento do projeto. Divide-se este capitulo em cada módulo e ferramenta utilizada.

Capítulo 3: Em materiais são apresentadas as definições de projeto, isto é, as ferramentas utilizadas para o desenvolvimento do projeto. Em método é apresentado todo o desenvolvimento do projeto, incluindo testes individuais, elaboração de algoritmos e procedimentos. Neste capitulo, descreve-se a implementação de cada módulo, preparando-o para compor o projeto final.

Capítulo 4: São apresentados os resultados e considerações de cada módulo abordado no Capítulo 3.

Capítulo 5: Conclusão do projeto, é o capitulo em que são detalhados os conceitos e lições aprendidos no desenvolvimento do projeto. Neste capitulo também são apresentadas as possibilidades tanto da evolução do projeto como também de derivações deste.

2. Embasamento teórico

Nesta seção serão apresentados os tema e linhas de pesquisa e desenvolvimento que foram utilizados como base para o desenvolvimento do projeto.

2.1. Sistemas embarcados

Sistemas embarcados consistem em uma combinação de Hardware, Software e possíveis componentes adicionais (mecânicos), desenvolvidos para a execução de uma função dedicada [12], [13]. Esses sistemas computacionais aplicados diferem de sistemas computacionais de propósito geral, como computadores pessoais (PC – *Personal Computer*) ou supercomputadores, apresentando maiores limitações de funcionalidade de hardware e de software [14], [15].

O adjetivo embarcado reflete o fato de esses sistemas serem usualmente parte integrante de um sistema maior. No entanto, apesar de muitos sistemas embarcados poderem coexistir em um sistema, eles podem, por si só, representar o sistema completo e operarem individualmente [12].

O uso de sistemas embarcados vem aumentando drasticamente em nosso dia-a-dia de forma não imaginada nas décadas passadas, sistemas embarcados estão transformando o modo de vida, trabalho e diversão das pessoas [12], [15].

Atualmente, é difícil encontrar na vida diária segmentos que não envolvam sistemas embarcados de alguma forma. Eles estão espalhados em diferentes áreas como indústria automotiva, eletrônica de consumo, aviônica, controle industrial, instrumentos médicos e dispositivos de rede (hubs, gateways, roteadores etc.) [15].

2.1.1. Arduino

Os Arduinos são módulos de desenvolvimento de código aberto que se baseiam em hardware e software flexíveis. O Arduino possui uma vasta comunidade na internet com bibliotecas prontas para diversas aplicações, o que agiliza e facilita a criação de projetos.

Existem diversas placas Arduino que matem um padrão de conexão compatível com shields. Shield é Hardware específico e dedicado que seguem o padrão Arduino de conexão e podem ser conectados a placa base Arduino.

Para a programação, a plataforma Arduino fornece um ambiente de desenvolvimento integrado (IDE) baseado no processamento do projeto que inclui suporte para as linguagens de programação C, C++ e Java.

O uso das bibliotecas prontas pode reduzir o desempenho do sistema uma vez que gera um nível de abstração elevado. Em alguns projetos essa velocidade pode ser crítica e nesse caso é necessário programar em baixo nível criando rotinas dedicadas e otimizadas a aplicação.

2.1.2. Sistemas embarcados para instrumentação

Na década de 90 o pesquisador Mark Weiser, do centro de pesquisa da Xeroz em Palo Alto, lançou o conceito de Computação Ubíqua (também conhecido como Computação Pervasiva). Neste conceito ele prevê uma nova tendência nos sistemas computacionais, inicialmente baseados em *mainframes* e posteriormente nos computadores pessoais, na qual os computadores estarão de tal forma embutidos no ambiente que os usuários terão uma interação com estes sistemas de uma maneira muito mais natural do que a atual. Uma das metas da Computação Ubíqua é habilitar dispositivos que permitam perceber as mudanças do ambiente e automaticamente se adaptarem a atuarem baseados nestas mudanças e nas necessidades e preferências do usuário. Esta tendência tem se tornado realidade graças ao avanço tecnológico nas mais diversas áreas como: computação distribuída, redes de sensores, interface homem-máquina, computação móvel, entre outras [16].

2.2. Sistemas operacionais

Os sistemas operacionais surgiram com o desenvolvimento da computação, como base para sistemas computacionais, promovendo desenvolvimentos mais modulares e a abstração entre hardware e código de aplicação [12].

Segundo Noergaard [14], sistemas operacionais (OS's) são um conjunto de bibliotecas de software que atendem dois propósitos principais:

- Prover uma camada de abstração, tornando o software mais independente do hardware, facilitando o desenvolvimento do software de aplicação.
- Gerenciar os vários recursos de software e hardware garantindo confiabilidade e eficiência na operação do sistema.

2.3. Sistema operacional embarcado – Linux (Software)

Os sistemas operacionais embarcados estão se popularizando devido ao avanço tecnológico que permite grande poder de processamento e dimensões reduzidas. Segundo os dados de 2006, 71% dos sistemas embarcados utilizam um SO, RTOS (*Real-Time Operating System*) ou *kernel*. Este fato é compreensível, visto que, de acordo com a

pesquisa, a maioria dos participantes utiliza microprocessadores de 32 bits, os quais se adequam melhor a sistemas operacionais (54%) [15].

Entre os principais motivos para o uso de sistemas operacionais estão a necessidade de gerenciamento de multitarefas, exigências de processos em tempo real, velocidade no desenvolvimento, modularidade, reuso e robustez são apontados como fatores decisivos na escolha pela utilização de um sistema operacional [15].

Nesse aspecto o Linux tem se destacado por tratar-se de um sistema aberto permitindo alterações e desenvolvimento de projetos com o auxílio do mesmo. O Linux também possui grande número de adeptos, proporcionando maior segurança no desenvolvimento de novos projetos à medida em que possíveis empecilhos ou desafios podem já ter sido enfrentados e solucionados por outros usuários.

2.4. Internet das coisas (IoT)

A Internet das Coisas ou Internet of Things (IoT), em inglês, é um termo utilizado para descrever um paradigma tecnológico no qual os objetos físicos estão conectados em rede e são acessados por meio da Internet.

Uma “Coisa”, no contexto da Internet das Coisas, é um objeto conectado que pode ser, por exemplo, uma pessoa com um monitor cardíaco, um animal rastreado em uma fazenda, um tanque industrial com sensores de nível, um carro com sensores que avisam a pressão dos pneus, uma lâmpada de iluminação pública de uma cidade, uma tomada em sua casa ou qualquer outro objeto natural ou construído pelo homem.

Atualmente as tecnologias de Internet das Coisas envolvem dispositivos conectados à aplicações por meio da Internet.

A Internet das Coisas possui várias tecnologias em comum com as tecnologias de comunicação *machine-to-machine* (M2M), usual em produtos industriais, de medição de energia, água, gás e óleo. Entretanto, o conceito de Internet das Coisas vai além da comunicação M2M pois propõe um futuro no qual todos os objetos estão conectados e comunicando-se de forma inteligente. Em outras palavras, o mundo físico com a Internet das Coisas dá origem a um grande sistema de informações [17].

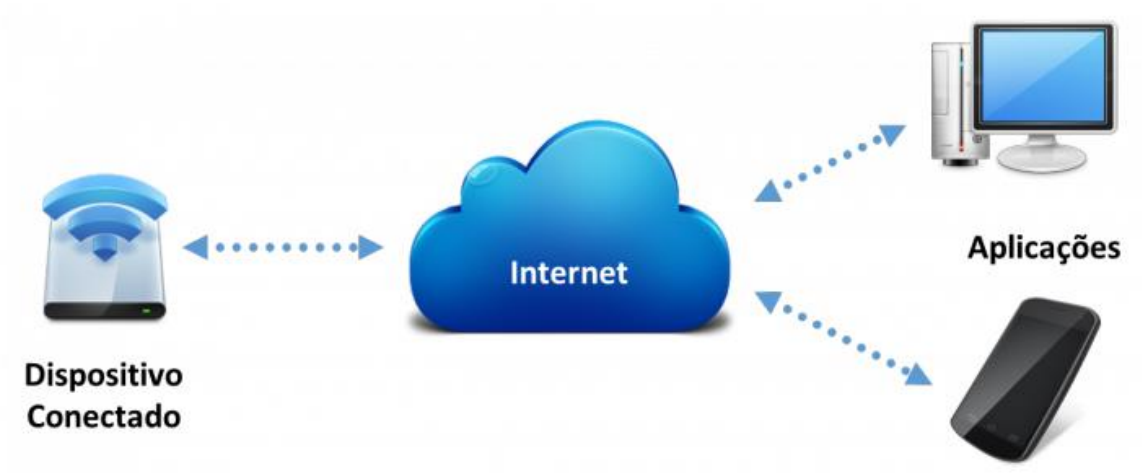


Figura 4: Representação da aplicação do conceito de internet das coisas [17]

2.5. Instrumentação

O conceito de internet das coisas tem por objetivo integrar objetos físicos em uma rede e disponibilizar acesso a estes por meio da internet. Nesse aspecto o uso de transdutor é fundamental para a conversão da grandeza de interesse em sinais mensuráveis e amostráveis. Atualmente existem diversos tipos de transdutores como os transdutores mecânicos (relógios comparadores e extensômetros), elétricos (resistivos, indutivos, etc.), acústicos (transdutores de corda vibrante), ópticos (mira telescópica, interferometria a laser, etc.), entre outros.

2.5.1. Transdutor – Acelerômetro

Os acelerômetros são sensores ou transdutores que medem acelerações. A aceleração é uma medida da variação da velocidade no tempo e pode ser obtida segundo uma, duas ou três direções, utilizando acelerômetros uni, bi ou triaxiais, respectivamente. Tipicamente, os acelerómetros são constituídos por uma massa de reação suspensa por uma estrutura estacionária. Este aparelho pode ser visto como um transdutor massa-mola, que se encontra no interior de um sensor. Sempre que este acelera, a inércia faz com que a massa resista. A força exercida pela massa é equilibrada pela mola e, como o deslocamento permitido pela mola é proporcional à força aplicada, a aceleração do corpo é proporcional ao deslocamento da massa. A magnitude da aceleração aplicada é vista, por instrumentos ou circuitos, como um impulso elétrico. O impulso eléctrico é depois processado por circuitos externos, podendo ser usado em inúmeras aplicações. Pode-se utilizar acelerômetros para medir não apenas acelerações (dinâmicas), como também

inclinação, rotação, vibração, colisão e gravidade (acelerações estáticas), constituindo assim um aparelho de elevada utilidade para projetos na área da eletrônica e robótica [18].

2.5.2. Transdutor - Pressão barométrica

Um sensor de pressão piezoresistivo consiste de uma fina membrana de silício monocristalino suportado por uma espessa borda de silício como mostra a Figura 5.

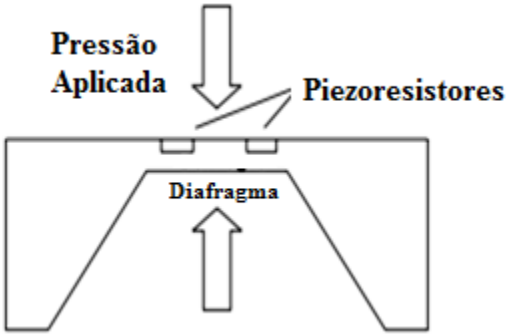


Figura 5: Desenho esquemático de um Sensor de Pressão Piezoresistivo.

No sensor o diafragma atua como um amplificador de stress mecânico. O Silício não é usado apenas como um substrato para difusão de piezoresistores, é também um material elástico e, por isso, utiliza-se a Lei de Hooke para relacionar as tensões e as deformações no diafragma. Quando uma pressão é aplicada sobre o dispositivo, o fino diafragma se curvará para baixo ou para cima, indicando tração ou compressão nos piezoresistores. A mudança de resistência causada por este stress pode ser facilmente medida [19].

2.5.3. Transdutor de Gás (GLP)

Os sensores de gás GLP comumente utilizados são sensor semicondutor de dioxido de estanho (SnO2). Este sensor apresenta menor condutividade em ar puro. Quando exposto a um local com gás GLP sua resistência aumenta. Dessa forma, um simples circuito eletrônico é capaz de detectar as variações de tensão provenientes da variação da resistividade do material

2.5.4. Instrumentação via web

Um sistema de instrumentação e medida por aquisição de dados pode ser definida como o conjunto de dispositivos que compõe uma cadeia pela qual os sinais físicos (de acordo com as suas características no tempo e na frequência) podem ser medidos, graças à sua conversão para sinais elétricos e, posteriormente para o formato digital adequado à apresentação e processamento em um sistema computacional [20] [21]. Obviamente, esta é uma definição moderna deste tipo de sistema, decorrente de uma série de avanços tecnológicos que diretamente ou indiretamente mudaram o conceito de instrumentação. Desses avanços pode-se destacar as tecnologias de conversão analógico-digitais, a popularização da presença dos computadores pessoais em sistemas de instrumentação. Nesse sentido, a Internet surge como interface de acesso remoto e disseminação de sistema embarcados dedicados à instrumentação, cada vez mais populares devido à redução dos custos dos dispositivos microeletrônicos e ao emprego de novas tecnologias de comunicação sem fio.

Dentro desse contexto surge um novo paradigma de sistemas de instrumentação baseado no conceito de Instrumentação Virtual, que pode ser definido como um sistema composto por um computador pessoal qualquer, um software dedicado à instrumentação e placas de aquisição de dados com seus respectivos drivers [22].

2.6. Visão computacional

O processamento digital de imagens envolve processos cujas entradas e saídas são imagens e, além disso, envolve processos de extração de atributos de imagens até e, inclusive, o reconhecimento de objetos individuais [23]. O processamento de imagens é composto por diversas fases. Estas fases estão apresentadas no diagrama de Figura 6.

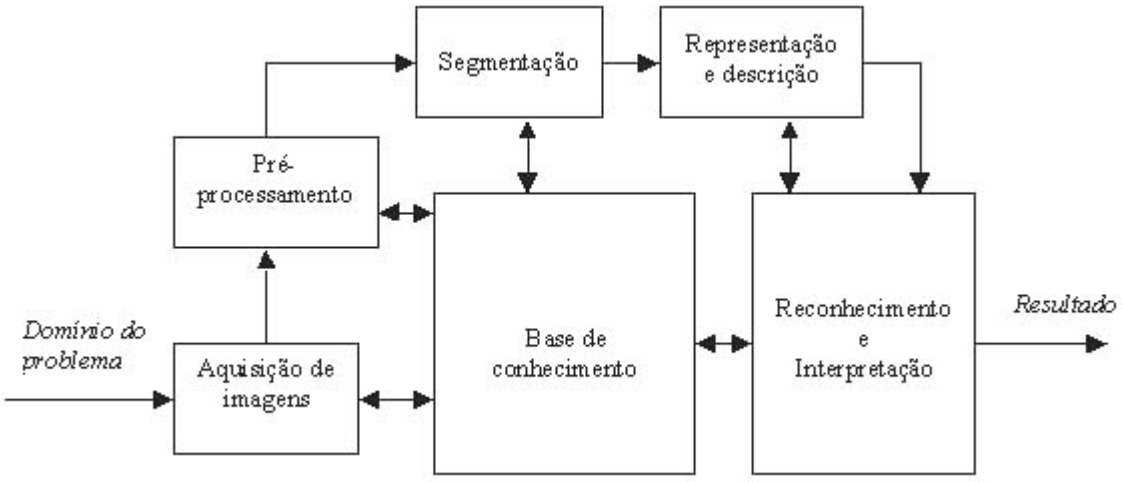


Figura 6: Diagrama de fases do processamento de imagens [23].

A primeira fase, aquisição da Imagem Digital, é caracterizada pelo uso de dispositivos físicos dotados de sensores sensíveis a certos espectros de energia eletromagnética e, posteriormente, digitalizar o sinal elétrico obtido. O Pré-processamento é a fase onde são realizadas transformações sobre a imagem visando um melhoramento na imagem para o sucesso das etapas posteriores. Este melhoramento pode ser obtido, por exemplos, realçando contrastes ou reduzindo ruídos. A segmentação é responsável por dividir a imagem em regiões disjuntas com algum significado para a aplicação, esta fase é totalmente dependente do domínio do problema. Na etapa da caracterização (ou descrição), procura-se extrair características das regiões segmentadas que tenham relevância para o processo. Por fim, a classificação, é o processo que identifica a imagem.

2.7. Comunicação Wireless

Torna-se cada vez mais popular e aplicado o conceito de conectividade dos equipamentos. Sua conectividade deve permitir a integração de equipamentos sem o uso de cabos que os limitam e dificultam sua instalação. Nesse sentido, a comunicação sem fio (*wireless*) é almejada na maioria dos projetos.

A LAN sem fio (WLAN ou Wi-Fi) é um sistema de transmissão de dados projetado para fornecer acesso à rede independente de localização entre dispositivos de computação, utilizando ondas de rádio em vez de uma infraestrutura de cabo.

A aceitação generalizada de WLANs depende da padronização da indústria para garantir a compatibilidade e confiabilidade do produto entre os vários fabricantes.

A especificação 802.11 [24] como um padrão para redes sem fio foi ratificada pelo Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) no ano de 1997. Esta versão de 802.11 prevê 1 Mbps e 2 Mbps taxas de dados e um conjunto de métodos de comunicação fundamentais e outros serviços. Como todos os padrões IEEE 802, os padrões 802.11 foca os dois níveis inferiores do modelo ISO, a camada física e da camada de enlace (Figura 7). Qualquer aplicativo LAN, sistema operacional de rede, protocolo, incluindo TCP / IP e Novell NetWare, será compatível com a WLAN 802.11.

A principal motivação e beneficiar de LANs sem fio é o aumento da mobilidade dos usuário e dispositivos conectados à rede.

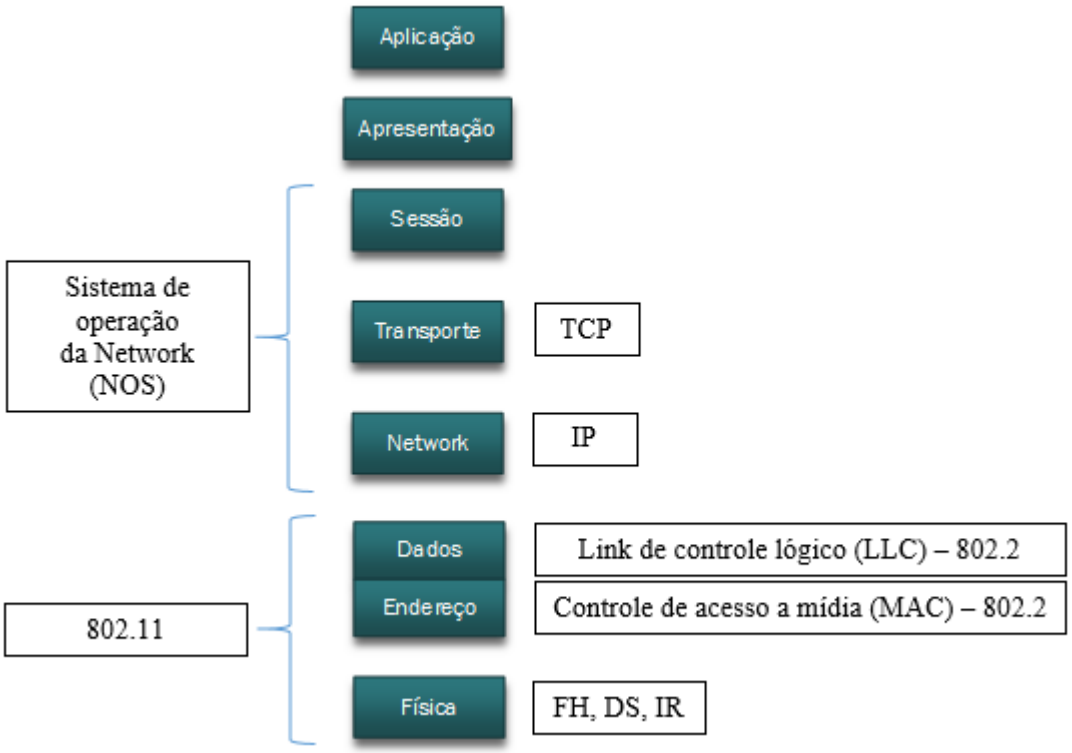


Figura 7: Camadas da Comunicação *Wireless*

As outras vantagens para WLAN incluem configuração de rede de baixo custo para locais de difícil fio tais como edifícios mais antigos e estruturas de parede sólida e reduzido custo de propriedade, especialmente em ambientes dinâmicos, exigindo modificações frequentes, graças a cablagem e instalação de custos mínimos por dispositivo e usuário. WLANs liberta os usuários da dependência de acesso *hard-wired* para o *backbone* da rede, dando-lhes a qualquer hora, em qualquer lugar de acesso à rede. Esta liberdade para

vaguear oferece inúmeros benefícios para o usuário para uma variedade de ambientes de trabalho.

2.8. Comunicação GSM

O GSM (sistema global de comunicações móveis) é o sistema de telecomunicações móveis de maior sucesso no mundo digital de hoje. Ele é usado por mais de 800 milhões de pessoas em mais de 190 países [25].

O GSM especifica diferentes mecanismos de transmissão de dados, o GSM original permite a comunicação utilizando taxas de dados de até 9600 bits/s para serviços que não incluem áudio.

Uma característica fundamental do sistema GSM é a localização automática, mundial de usuários. O sistema detecta a localização do atual do usuário e o mesmo número de telefone é válido em todo o mundo. Para fornecer este serviço, GSM realiza localização periódica e atualiza mesmo se um usuário não usa a estação móvel (desde que o MS *(Mobile Station)* ainda esteja conectado à rede GSM e não esteja completamente desligado). O HLR (*Home Location Register*) sempre contém informações sobre a localização atual (apenas a área de localização, não a localização geográfica precisa), e o VLR (*Visitor Location Register*), atualmente responsável pelo MS, informa ao HLR sobre mudanças de local. Assim que uma MS se move para o intervalo de uma nova VLR (uma nova área de localização), o HLR envia todos os dados necessários para a nova VLR [25].

3. Materiais e métodos

A organização deste trabalho bem como o método de abordagem do estudo dos blocos constituintes do trabalho serão apresentados e detalhados individualmente. Posteriormente será apresentado os resultados da integração dos módulos.

3.1. Materiais

Nessa seção serão apresentados os materiais utilizados e estudados no desenvolvimento do projeto, evidenciando suas características.

3.1.1. Placa de Desenvolvimento Intel Galileo (Hardware)

A placa de desenvolvimento Intel Galileo é uma placa de microcontrolador baseado no *"Intel Quark SoC Processor X1000".* Contém um processador *Intel Pentium* de 32 bits e é a primeira placa baseada na arquitetura Intel projetada para ser compatível com hardware, software e *shields* Arduino Uno R3. Possui pinos digitais 0-13, entradas analógicas 0-5, conexões ICSP, e os pinos da porta UART (0 e 1). Todos os pinos são posicionados de acordo com o padrão Arduino 1.0 de conexão.

O Galileo foi projetado para suportar *shields* que operam em 3.3V ou 5V. A tensão de funcionamento do núcleo do Galileo é 3.3V. No entanto, um *jumper* na placa permite a alteração de tensão para 5V nos pinos de I/O, isso fornece suporte para *shields* Uno de 5V. Ao mudar a posição do *jumper*, pode-se operar com 3.3V nos pinos de I/O.

O Galileo também é compatível com o *Arduino Software Development Environment* (IDE), o que torna seu uso um processo fácil e dinâmico. Além de hardware Arduino e compatibilidade de software, o Galileo tem vários padrões da indústria PC, portas I/O e recursos para expandir o uso nativo e capacidades para além da base Arduino. O Galileo possui *slot mini-PCI Express* de tamanho completo, porta 100Mb *Ethernet*, *slot Micro-SD*, porta serial RS-232, porta *USB Host*, porta *USB device*, e 8Mbyte de memória *flash* de fábrica na placa[26]. As Figura 8a e 4b apresentam as vistas superiores e inferiores da placa de desenvolvimento Intel Galileo. A Figura 9 apresenta um diagrama esquemático da placa, identificando seus periféricos.

a) Vista superior b) Vista inferior

Figura 8: a) e b) são imagens da placa Intel Galileo (Gen1) [27].

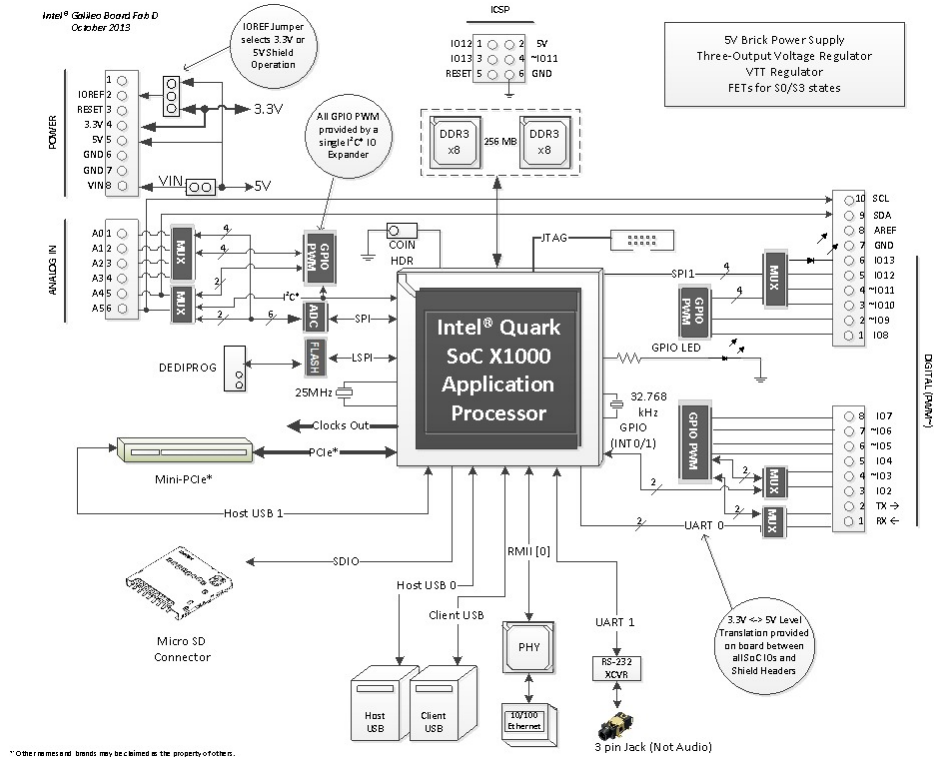


Figura 9: Esquemático da placa [6]

Na seção seguinte (Conexões) será apresentada uma visão geral dos conectores e peças que cercam o processador Quark para formar o módulo de desenvolvimento Intel Galileo.

➢ Conexões:

A Figura 10 apresenta em destaque todas as conexões disponíveis na parte superior da placa Intel Galileo. Na parte inferior, há slot para Mini PCIe.

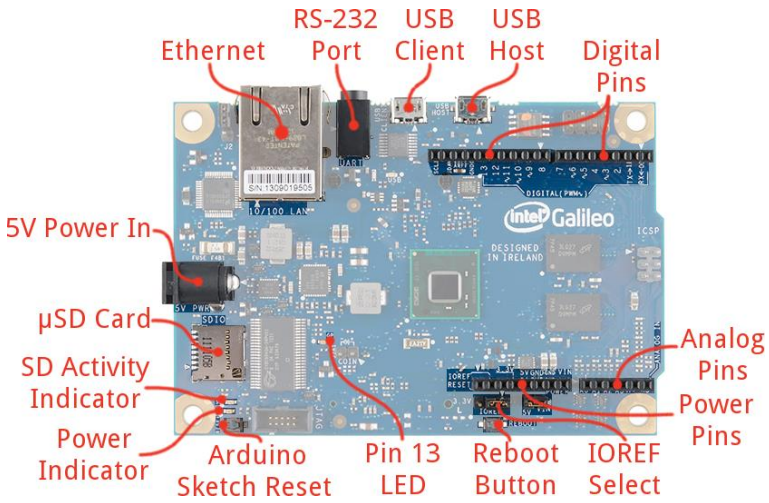


Figura 10: Visão geral dos conectores e peças que cercam o processador Quark para formar o Galileo [28].

A parte superior da Galileo é o lugar onde estão a maior parte das conexões:

- **Ethernet** - Conecta o Galileo a LAN 10/100 Mb/s.
- **RS-232** – Utiliza um conector de 3,5 milímetros, sendo necessário um cabo com conexão P2 (comumente utilizado para áudio)
- **USB *Device*** – Utilizado para conexão com computador para programação d**o Galileo como Arduino**.
- **USB 2.0 Host** - Este suporta uma interface com dispositivos USB, como teclados, dispositivos de armazenamento, etc. Com um hub USB, até 128 dispositivos podem ser conectados a esta porta.
- **Possui os conectores padrão do Arduino**:
  - 8 pinos de alimentação (3.3V, 5V, GND, Reset, etc.)
  - 6 pinos de entrada analógica (A0-A6)
  - 8 pinos I/O digital header (D0-D7), incluindo UART nos pinos 0/1, PWM nos pinos 3, 5 e 6
  - 10 pinos I / O digital header (D8-SCL), incluindo os pinos I^2C e PWM.
  - 2x3 pinos header ICSP e SPI.

- **Botão de reinicialização** - Pressionar este botão irá reiniciar todo o Galileo (incluindo o Linux). Tempo de boot é cerca de 30 segundos.
- **Pin 13 LED** - Tal como acontece com a maioria dos Arduinos, o Galileo dispõe de um pequeno LED (on-board) conectado ao pino 13.
- **Botão Reset Arduino** - Isto irá reiniciar apenas o Arduino em execução no Galileo.

- **Cartão µSD** - O Galileo suporta até 32GB cartões SD. Este cartão é utilizado para iniciar a imagem completa do sistema operacional.
- **5V In** - Este é um conector de 2,1 milímetros jack de centro-positivo para uma fonte de 5V regulados**.**
- **Memória DRAM**: Possui 256 MBytes de memória DRAM, habilitada pelo firmware.
- **Memória EEPROM:** Possui 11 kByte de EEPROM, que pode ser programado por meio da biblioteca de EEPROM

O Intel® Quark SoC X1000

O Galileo possui um processador Quark SoC X1000. Este é um processador de 32-bit construído sobre a arquitetura x86. O Quark X1000 apresenta as seguintes características:

- Velocidade máxima de *clock* de 400 MHz;
- 16KB de cache L1;
- 512KB SRAM;
- Núcleo único;
- SDIO, UART, SPI, USB, I^2C, Ethernet, RTC integrados;

O foco da Quark X1000 é a computação embarcada. É destinado a ter um baixo consumo de energia (15W), detém, portanto, um processador com arquitetura baseada em x86 para atender a uma grande variedade de projetos como por exemplo, dispositivos portáteis, aplicação do conceito de Internet das Coisas (*IoT*) nos projetos, veículos autónomos, entre outros.

Arquitetura: x86 *vs* ARM

A diferença fundamental entre as arquiteturas ARM e x86 é o tamanho do conjunto de instruções. A arquitetura ARM apresente o conjunto de instruções RISC (*Reduced Instruction Set Computing*). Este conjunto de instruções é mais simples e restrito, é necessário, portanto, maior conhecimento em programação para o desenvolvimento de aplicações sofisticadas. Os processadores com arquitetura x86 apresentam o conjunto de instruções CISC (*Complex Instruction Set Computing*) que por sua vez apresenta maior número de instruções o que facilita a implementação dos projetos. Em termos de processamento, o desempenho dos processadores x86 é melhor, pois um processo que

leva um ciclo de *clock* em um x86, pode levar três em um processador ARM. Entretanto, um grande conjunto de instruções requer mais hardware e consequentemente há um maior consumo de energia.

Produtos da Intel *Atom* e *Quark* tentar combater o alto consumo de energia. Eles estão direcionados para o mercado móvel. Mas eles ainda são de arquiteturas x86, consumindo, portanto, mais energia.

- **Resumo elétrico da placa:**

A Tabela 1 apresenta o resumo elétrico da placa Intel Galileo. Essas informações são base para a construção e adequação do projeto.

Tabela 1: Resumo elétrico da placa Intel Galileo

Tensão de entrada	5V
Pinos Digitais I/O	14 (dos quais 6 oferecem saída PWM)
Pinos de entrada analógica	6
Total de Corrente de saída DC em todas as linhas de I/O	80 mA
Corrente DC 3.3V por porta	800 mA
Corrente DC 5V por porta	800 mA

- **Medição de consumo de corrente elétrica dos módulos**

Para aferir o valor da corrente dos módulos, foi utilizado o multímetro digital Minipa – ET-2940.

- **Fonte de alimentação utilizada**

A fonte de alimentação utilizada, foi uma fonte chaveada genérica modelo ODL-0520, capaz de fornecer 5V de saída e 2A.

3.1.2. Projeto Yocto

O projeto Yocto é um projeto de colaboração de software livre que fornece modelos, ferramentas e métodos que suportam sistemas customizados baseados em Linux para produtos integrados, independentemente da arquitetura de hardware [29].

O Yocto inclui em sua arquitetura o *build system Poky*, que, por sua vez, é derivado do *build system OpenEmbedded*. O *OpenEmbedded* é composto de dois elementos principais, como mostrado na **Erro! Fonte de referência não encontrada.**: *BitBake* e *Metadata*. O *BitBake* é uma ferramenta de *build* muito flexível mantida pelos projetos Yocto e *OpenEmbedded*, comandada pelas instruções presentes no *Metadata* e com a finalidade de gerar, entre outros, as imagens finais do sistema de arquivos, *kernel*, *bootloader* e *SDKs* [30].

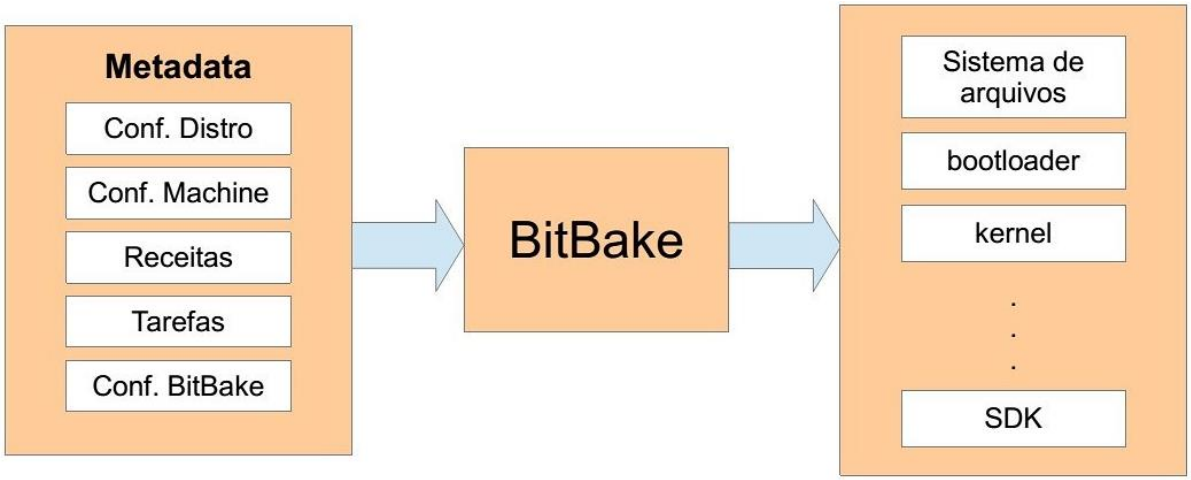


Figura 11: BitBake, ferramenta de build mantida pelos projetos Yocto e OpenEmbedded. [30].

3.1.3. Linguagem de programação Python

Python é uma linguagem de altíssimo nível (em inglês*, Very High Level Language*) orientada a objetos, de tipagem dinâmica e forte, interpretada (híbrida) e interativa. Entre as linguagens dinâmicas, Python se destaca como uma das mais populares e poderosas [31].

A linguagem interpretada requer outro programa, o interpretador. Esse método tem uma séria desvantagem em relação ao tempo de execução, que é de 10 a 100 vezes mais lento que nos sistemas compilados [32]. Isso ocorre devido a decodificação das sentenças em linguagem de máquina. Além disso, independentemente de quantas vezes uma sentença for executada, ela deve ser decodificada a cada vez.

A linguagem híbrida traduz os programas de linguagem em alto nível para uma linguagem intermediária projetada para facilitar a interpretação. Sistemas de implementação híbridos são mais rápidos do que a interpretação pura porque as sentenças da linguagem fonte são decodificadas apenas uma vez [32].

Deve-se considerar, contudo, que as linguagens interpretadas ou híbridas apresentam desempenho inferior a uma linguagem compilada. Isto é consequência do processo de tradução previamente realizado para a linguagem de máquina. O compilador é capaz de fazer otimizações significativas nesse processo de tradução para linguagem de máquina.

No projeto apresentado optou-se pela utilização de Python por ser uma linguagem hibrida e por apresentar grande facilidade de programação, permitindo a rápida implementação do sistema proposto. Esta linguagem de programação também possui grande número de fóruns e bibliotecas que auxiliam no desenvolvimento das aplicações. O processamento requerido para a aplicação será avaliado no desenvolvimento do projeto.

3.1.4. Biblioteca dedicada à visão computacional – OpenCV

A biblioteca OpenCV foi desenvolvida pela Intel e possui mais de 500 funções [33]. Foi idealizada com o objetivo de tornar a visão computacional acessível a usuários e programadores em áreas tais como a interação homem-máquina em tempo real e a robótica. A biblioteca está disponível com o código fonte e os executáveis (binários) otimizados para os processadores Intel. Um programa OpenCV, ao ser executado, invoca automaticamente uma DLL (*Dynamic Linked Library*) que detecta o tipo de processador e carrega, por sua vez, a DLL otimizada para este. Juntamente com o pacote OpenCV é oferecida a biblioteca IPL (*Image Processing Library*), da qual a OpenCV depende parcialmente, além de documentação e um conjunto de códigos exemplos.

A biblioteca está dividida em cinco grupos de funções: Processamento de imagens, análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões, calibração de câmera e reconstrução 3D.

3.1.5. Câmeras IPs

As câmeras IPs estão se popularizando devido a sua facilidade de uso. Seu funcionamento independe de um computador local, basta que haja um roteador Wi-Fi no ambiente a ser instalada. Se este roteador possuir acesso à internet, então, será possível utilizar de recursos e configurações para visualizar a imagem da câmera IP de qualquer

lugar por meio de um computador, smartphone, Iphone, Tablet ou outros que utilizem navegadores de internet e possuam acesso a esta. Além da facilidade apresenta, as câmeras IPs ainda permitem rotação de 320º na horizontal e de 120º na vertical [34]. Também apresentam LEDs infravermelho que permitem a captura de imagens em ambientes escuros. A Figura 12 apresenta a câmera utilizada no projeto proposto.

Figura 12: Câmera IP utilizada no projeto proposto

3.1.6. Transdutores – Acelerômetros ADXL335 e MPU-6050

Para o desenvolvimento do projeto forma analisado dois acelerômetros. A partir de testes foi constatado o acelerômetro mais indicado para aplicação e este foi integrado ao protótipo final. As características desejáveis são: alta precisão, alta velocidade de comunicação, imunidade a ruído.

A primeira das opções de um acelerômetro foi o CI ADXL335 que é um pequeno acelerômetro, de baixa potência, e capaz de efetuar medidas de 3 eixos com sinal condicionado a saídas de tensão. A aceleração medidas permite variação de amplitude mínima em grande escala de ± 3g.

Ele pode medir a aceleração da gravidade estática em aplicações de detecção de inclinação, bem como aceleração dinâmica resultante do movimento, vibração, ou para o caso desta aplicação, a medida de choque mecânico. O diagrama de blocos de funcionamento é apresentado na Figura 13, permitindo observar a saída referente a cada eixo coordenado [35]. A Figura 14 apresenta o módulo ADXL335 utilizado no projeto.

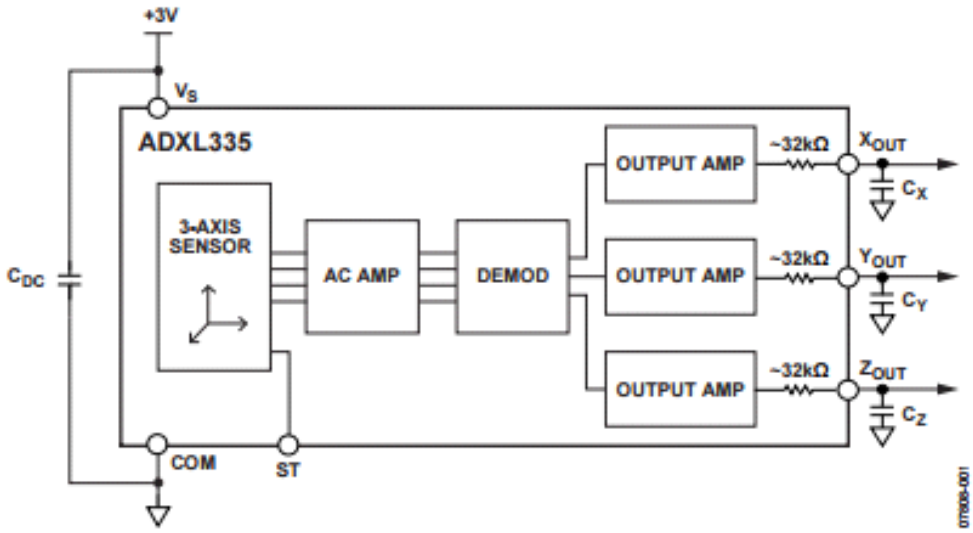


Figura 13: Diagrama de blocos do acelerômetro [35]

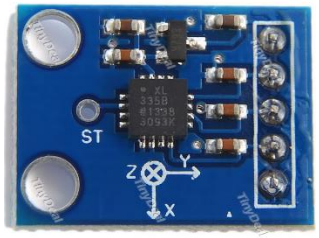


Figura 14: Módulo ADXL335 [36]

A segunda opção foi o CI MPU6050 que apresenta saída processada, com comunicação I²C como mostra a Figura 15.

O sensor de MPU-6050 contém um acelerômetro e um giroscópio MEMS (*Micro-Electro-Mechanical Systems*) em um único chip. Ele é muito preciso, pois ele contém 16-bits de conversão análgico/digital para cada pino. Captura variações nos eixos x, y, e z de canal ao mesmo tempo [37]. O módulo Arduino oferece apenas a comunicação I²C-Bus para fazer a interface.

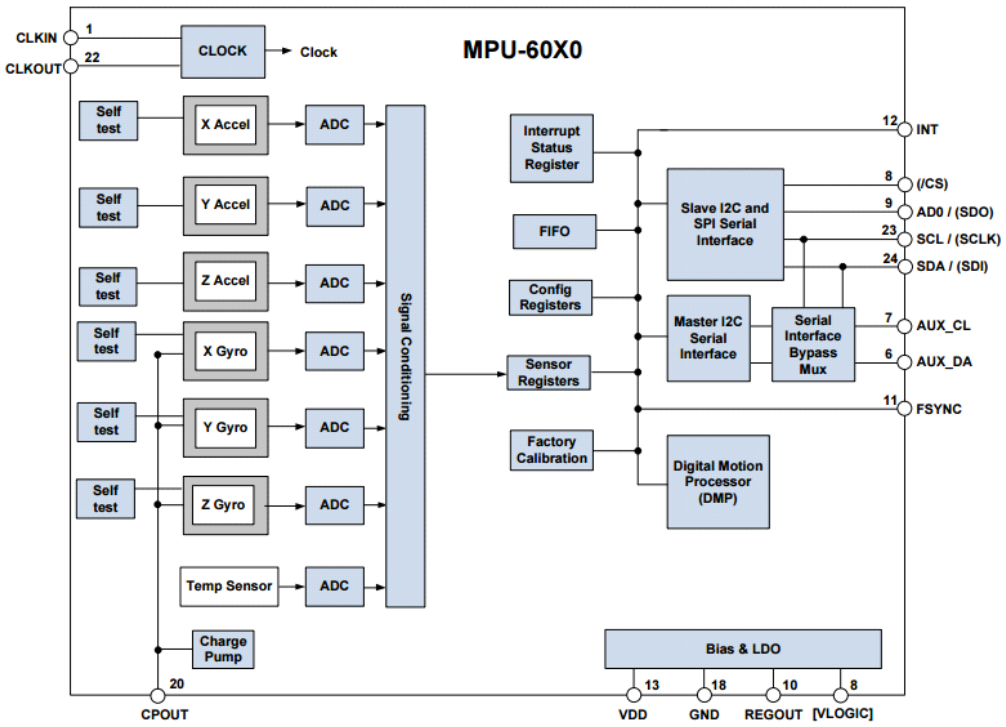


Figura 15: Diagrama de blocos do transdutor MPU6050 [37]

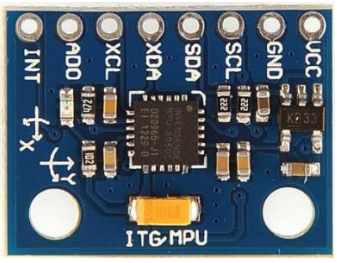


Figura 16: Módulo MPU6050 [38]

Características do Giroscópio do CI MPU6050 [37]

O giroscópio MEMS de eixo triplo no MPU-60X0 inclui uma ampla gama de recursos:

- Saída digital para os eixos X, Y, e Z.
- Sensores de frequência angular (giroscópios) com escalas programáveis pelo usuário na faixa de ± 250, ± 500, ± 1000, e ± 2,000 °/seg.
- Sinal de sincronismo externo ligado ao pino Fsync suporta imagem, vídeo e GPS sincronização
- ADCs integradas de 16 bits permitir amostragem simultânea de giroscópios

- Viés aprimorada e estabilidade de temperatura.
- Reduz a necessidade de calibração do usuário
- Melhor desempenho de ruído de baixa frequência
- Filtro Passa-Baixa Digital programável
- Giroscópio operacional atual: 3.6mA
- Atual de espera: 5μA
- Fator de escala de sensibilidade calibrado de fábrica
- Auto-teste do Usuário

Características do acelerômetro do CI MPU6050 [37]

O acelerômetro MEMS de eixo triplo em MPU-60X0 inclui uma ampla gama de recursos:

- Saída Digital do acelerômetro de eixo triplo, com um intervalo de escala programável de ± 2 g, ± 4 g, ± 8g e ± 16g
- ADCs integradas de 16 bits permitir amostragem simultânea de acelerômetros e não exige multiplexadores externos.
- Acelerômetro atual de funcionamento normal: 500μA
- O modo de baixa energia do acelerômetro apresenta: 10μA em 1.25Hz, 20μA em 5 Hz, 60μA em 20Hz, 110μA em 40Hz.
- Orientação de detecção e sinalização
- Interrupções programáveis pelo usuário
- Auto-teste do Usuário

3.1.7. Transdutor – Barômetro BMP180

Para o projeto apresentado optou-se pela utilização do barômetro BMP180. Este circuito integrado consiste de um sensor piezo-resistivo. O BMP180 é uma nova geração de sensores altímetros de alta resolução munidos de interface I²C otimizado para altímetros e variômetros com uma resolução de 17 cm de altitude. O módulo inclui um sensor de pressão de alta linearidade, baixa potência e grande estabilidade térmica por possuir sensor de temperatura para a compensação desta [39]. A Figura 17 apresenta o diagrama de funcionamento do sensor e sua respectiva comunicação. A Figura 18 apresenta o módulo BMP180 utilizado no projeto.

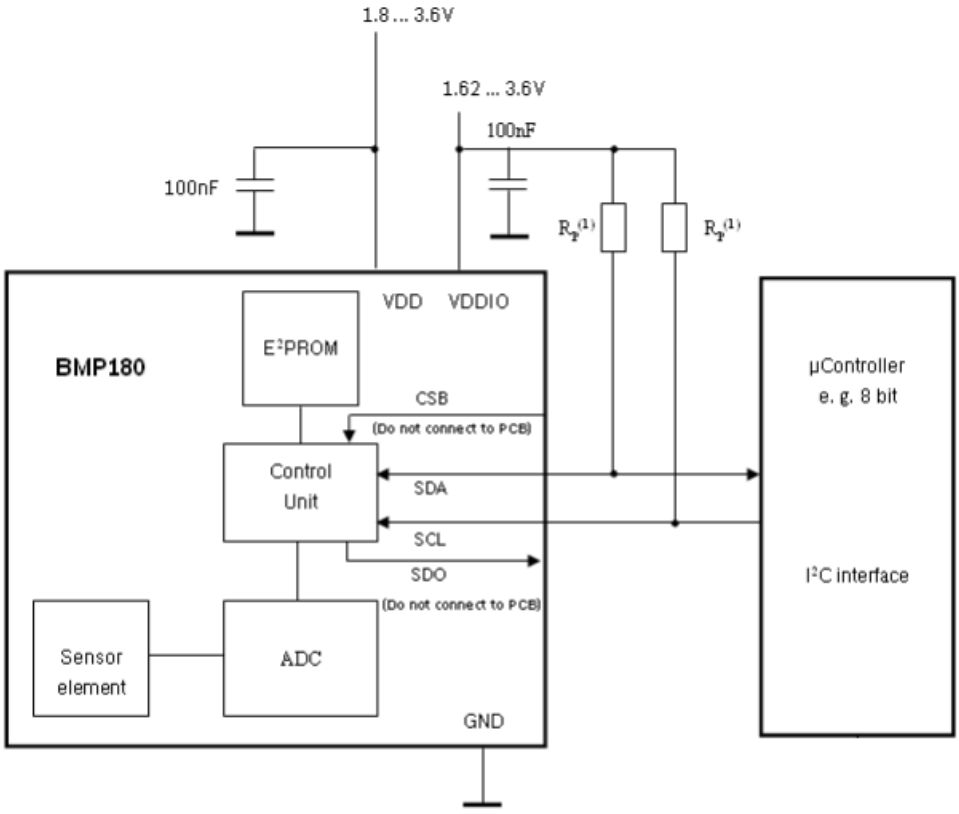


Figura 17: Diagrama de conexão do módulo BMP180 [39]

Figura 18: Módulo BMP180 [40]

- **Considerações:**

Tempo: Mudanças de pressão devido ao mau tempo irá afetar suas leituras de altitude.

Altitude máxima: O BMP180 apresenta altitude máxima de medida de 3000 m.

Altitude mínima: Da mesma forma, este sensor não é adequado para grandes pressões. O limite superior é de 1100 hPa, que é aproximadamente 152 metros abaixo do nível do mar.

3.1.8. Módulo GSM SIM900

O módulo GSM SIM900 apresenta-se como solução para o envio de mensagem de texto para o celular do responsável. O uso de um módulo GSM facilita o projeto por possuir o correto casamento de impedância do circuito, principalmente da parte de transmissão e recepção de sinais [41]. A Tabela 2 sintetiza as características do módulo Sim900.

Tabela 2: Síntese das características do CI Sim900

Característica	Descrição
Alimentação	3,2V ~ 4,8V
Frequências de transmissão	Sim900 *Quad-band*: GSM 850MHz, EGSM 900MHz, DCS 1800MHz, PCS 1900MHz. Sim900 utiliza a busca pelas frequências automaticamente.
Potência de transmissão	Classe 4 (2W) para GSM 850 e EGSM 900 Classe 1 (1W) para DCS 1800 e PCS 1900
Dados GPRS	Transferência de dados máxima (GPRS *downlink*) 85,6 kbps Transferência de dados máxima (GPRS *uplink*) 42,8 kbps
CSD	Suporte a transmissão CSD
USSD	Suporte a *Unstructured Supplementary Services Data* (USSD)
SMS	MT, MO, CB, Texto e modo PDU Armazenamento de SMS: cartão SIM
Interface SIM	Suporte ao cartão SIM: 1,8V e 3V
Antena Externa	Sim
Comunicação	Porta serial: Interface de controle completa com linhas de comando assíncronas Frequência de comunicação: 1200bps a 115200bps Suporte a comandos AT Porta de Debug Interface de comunicação DBG_TXD e DBG_RXD Esta porta pode ser utilizada para debug e atualização do firmware.
RTC	Suporte a RTC

A Figura 19 apresenta o módulo de desenvolvimento a ser utilizada no projeto, munido do CI SIM900.

Figura 19: Módulo GSM

3.1.9. Transdutor - Sensor de gás MQ-2

O sensor utilizado foi o MQ-2. Trata-se de um sensor semicondutor de dióxido de estanho (SnO2) que apresenta menor condutividade em ar puro. Quando este sensor é exposto a um local com gás GLP sua resistência aumenta. Dessa forma, um simples circuito eletrônico é capaz de detectar as variações de tensão provenientes da variação da resistência do material. O sensor de gás MQ-2 tem alta sensibilidade para GLP, propano e hidrogênio, também poderia ser usado para metano e outros vapores combustível [42]. A variação da resistência $\left(\frac{R_S}{R_O}\right)$ em função da concentração em partes por milhão das substâncias sensibilizadoras são apresentados no gráfico da Figura 20.

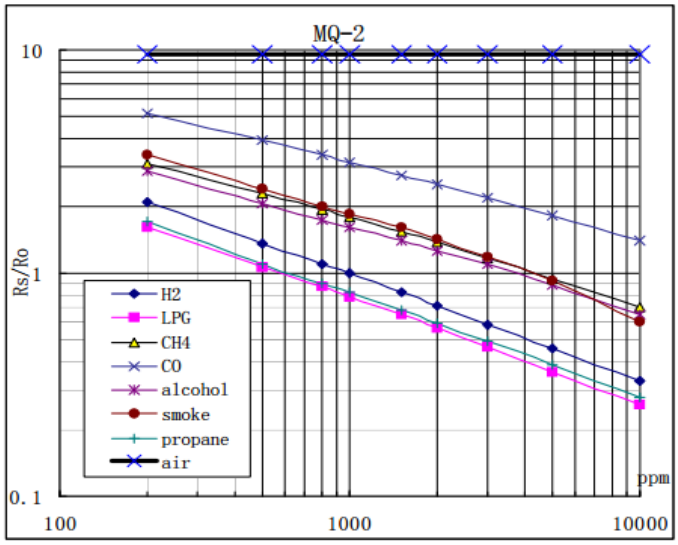


Figura 20: Variação da resistência em função da concentração de gases [42]

Seu nível de detecção abrange de 300 a 10.000 ppm (partes por milhão), ajustáveis por um potenciômetro na parte de trás do módulo. Um chip comparador LM393 é responsável por ler as informações do sensor e converter essas informações em sinais para o microcontrolador.

A tensão de alimentação do módulo é de 5V e a comunicação com o microcontrolador pode ser feita de duas maneiras: pela saída digital D0 ou pela saída analógica A0, conforme indicado na Figura 21. Pode-se utilizar a saída digital para atuações binárias como acionamentos de mecanismos. Já a saída analógica informa o nível de concentração de gases detectados pelo sensor. Quanto maior a concentração, maior o nível de sinal na saída analógica A0.

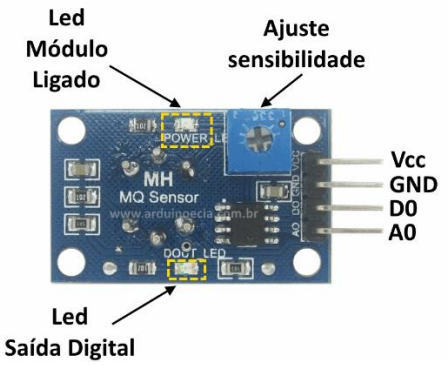


Figura 21: Módulo sensor de gás [43]

3.1.10. Módulo Wi-Fi ESP8266

A seguir será apresentado algumas de suas principais características do módulo ESP8266 [44]:

- É um *System-On-Chip* com Wi-Fi embutido;
- Possui conectores GPIO, barramentos I²C, SPI, UART, entrada ADC, saída PWM e sensor interno de temperatura;
- CPU que opera em 80MHz, com possibilidade de operar em 160MHz;
- Arquitetura RISC de 32 bits;
- 32KBytes de RAM para instruções;
- 96KBytes de RAM para dados;
- 64KBytes de ROM para boot;
- Possui uma memória Flash SPI *Winbond* W25Q40BVNIG de 512KBytes;
- O núcleo é baseado no IP *Diamand Standard* LX3
- Fabricado pela *Espressif*;
- Existem módulos de diferentes tamanhos e fabricantes.

Os módulos ESP8266 estão disponíveis atualmente em 12 modelos distintos, sendo nomeados de ESP8266-01 a ESP8266-12. Os módulos ESP8266-01 a ESP8266-10 tem por objetivo transmitir dados seriais UART para *wireless* (Wi-Fi) por meio de conexões TCP/UDP. O módulo ESP8266-12 utilizado no projeto proposto também é capaz de realizar esta função, porém, ainda é disponibilizado uma porta com conversor AD utilizado para realizar a aquisição de dados do sensor de gás.

O modelo ESP8266-01 é o mais comumente utilizado e mais amplamente comentado até o momento [44]. A principal aplicação deste modelo é de utiliza-lo como ponte Serial-Wi-Fi, seja com o Arduino, propriamente, seja com qualquer outro microcontrolador com porta de comunicação serial.

A Figura 22 apresenta o modelo ESP8266-01 e posteriormente é apresentado uma breve descrição de seus pinos.

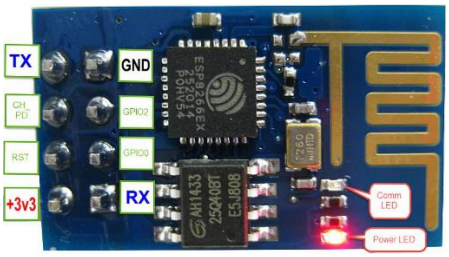


Figura 22: Módulo ESP8266 – 01 [45]

- Vcc: Tensão de alimentação 3,3V. Este módulo consome até 300 mA;
- GND: Sinal de Terra GND;
- Tx: Sinal de Tx do módulo, a ser conectado no Rx do microcontrolador
- Rx: Sinal de Rx do módulo, a ser conectado no Tx do microcontrolador
- RST: Sinal de *Reset/Restart* acionado em nível baixo (GND);
- CH_PD: Sinal de habilitação do chip (*chip enable*), usado para gravação de firmware ou atualização. Deve ser mantido em nível ALTO para operação normal;
- GPIO0: Pode ser controlado pelo firmware, e deve ser colocado em nível baixo (GND) para modo *update*, ou em nível alto para operação normal;
- GPIO2: I/O que pode ser controlada pelo firmware;
- LED: Quando está ligado, fica aceso em cor Vermelha, e aciona a cor Azul para indicar atividade. Pisca uma vez para indicar momento de boot.

Uma questão relevante sobre o funcionamento deste modulo é o fato de apresentar um considerável consumo de energia, o que pode ser prejudicial em sistema com alimentação restrita. A Tabela 2 [46] quantifica o consumo de corrente para cada modo de operação do módulo.

Tabela 2: Consumo do módulo ESP8266-01 para cada modo de operação.

Modo	Típico	Unidades
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215	mA
Transmit 802.11b, CCK 1Mbps, POUT=+19.5dBm	215	mA
Transmit 802.11b, CCK 11Mbps, POUT=+18.5dBm	197	mA
Transmit 802.11g, OFDM 54Mbps, POUT =+16dBm	145	mA
Transmit 802.11n, MCS7, POUT=+14dBm	135	mA
Receive 802.11b, packet length=1024 byte, -80dBm	60	mA
Receive 802.11g, packet length=1024 byte, -70dBm	60	mA
Receive 802.11n, packet length=1024 byte, -65dBm	62	mA
Standby	0.9	mA
Deep sleep	10	uA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.86	mA
Total shutdown	0.5	uA

A Figura 23 apresenta o módulo ESP8266 -12, este módulo disponibiliza 9 portas de GPIO e um conversor ADC além das funções do Módulo ESP866 – 01 já apresentadas.

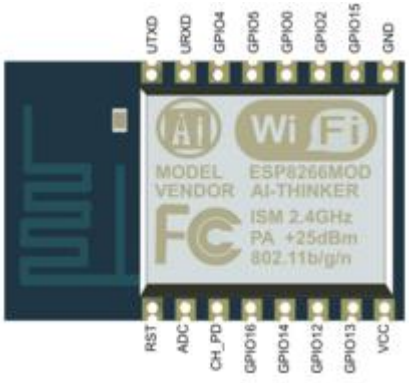


Figura 23: Módulo ESP8266 – 12 [47]

3.1.11. Arduino Mega2560

O Arduino Mega2560 é uma placa de desenvolvimento com microcontrolador ATmega2560. Possui 54 canais de entrada/saída, desses, 15 de PWM, 16 entradas analógicas, 4 portas de comunicação UARTs, oscilador de 16MHz, conexão USB, alimentação externa, regular de tensão integrado e *ICSP header*. Esta placa propicia o rápido desenvolvimento e teste de sensores e demais módulos, pois, é disponibilizado distintas formas de comunicação (UART, SPI, I²C). Todavia, apresenta como desvantagem um desempenho reduzido ao se utilizar a programação Arduino pois o uso de bibliotecas prontas agiliza o processo de desenvolvimento, porém, o maior nível de abstração implica numa redução de desempenho.

A Tabela 3 apresenta algumas especificações técnicas relevantes para o projeto [48]. A Figura 24 apresenta a visualização da placa Arduino Mega2560.

Tabela 3: Especificações Técnicas do Arduino Mega2560

Microcontrolador	ATmega2560
Tensão operacional	5V
Tensão de entrada (recomendada)	7-12V
Memória Flash	256 kB
SRAM	8 kB
EEPROM	4k
Frequência de clock	16 MHz
Comprimento	101,52 mm
Largura	53,3 mm
Peso	37 g

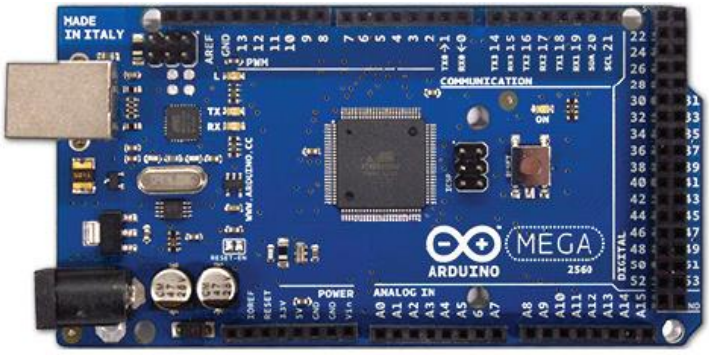


Figura 24: Arduino Mega2560 [49]

- **Medição de consumo de corrente elétrica dos módulos**

Para aferir o valor da corrente dos módulos, foi utilizado o multímetro digital Minipa – ET-2940.

- **Fonte de alimentação utilizada**

A fonte de alimentação utilizada, foi uma fonte chaveada genérica capaz de fornecer 12V de saída e 1A. O módulo Arduino Mega2560 possui regulador de tensão que ajusta a tensão de entrada para a tensão de 5V, o que justifica a utilização de uma fonte de 12V.

3.1.12. Conversor de Tensão

O módulo Arduino Mega2560 apresenta comunicação com tensão de 5V para nível lógico alto. Os módulos de barômetro, acelerômetro e ESP8266 requerem comunicação de 3,3V para nível lógico alto, desta forma, fez-se necessário o uso de um sistema de acondicionamento de sinal. Seu funcionamento se baseia em um divisor resistivo para aplicações unidirecionais e uso de transistores para aplicações bidirecionais. Seu circuito esquemático é apresentado no anexo deste trabalho.

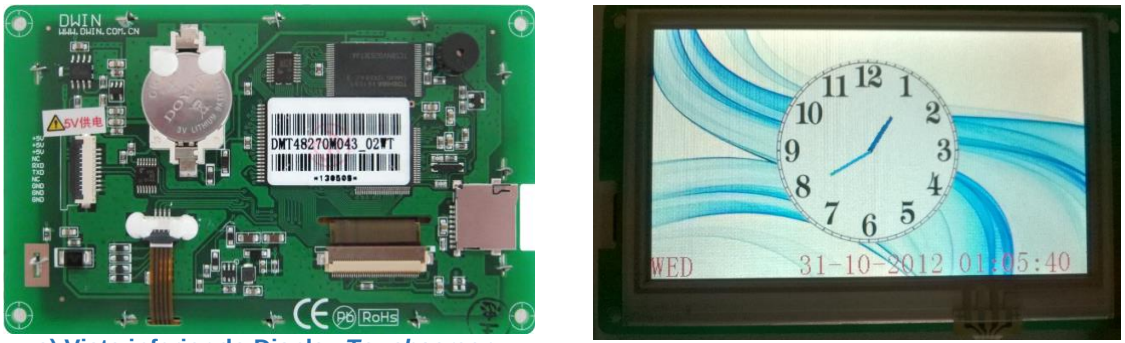
Figura 25: Conversor de tensão

3.1.13. Display *Touhscreen* DWIN

Para permitir futuro desenvolvimento do projeto foi utilizado um display *touchscreen* DMT4820M043_02WT da empresa DWIN. Sua escolha ocorreu devido a sua facilidade de uso e baixo custo. Suas características [50] estão apresentadas na Tabela 9.

Tabela 4: Características do display *touchscreen* DMT4820M043_02WT

Característica	Valor
Tamanho do display	4,3 polegadas (109 mm)
Cores	16 bits (65536 cores)
Resolução	480 x 272 pixels
Área *touchscreen*	95,0 mm x 53,9 mm
Área display	96,0 mm x 55,8 mm
Tensão	5V
Corrente (display ligado)	210 mA
Corrente (display desligado)	100 mA
Periféricos	Buzzer, RTC



a) Vista inferior do Display *Touchscreen*.

b) Display *Touchscreen* utilizado no projeto

Figura 26: Vista superior e inferior do Display *Touchscreen* utilizado no projeto

3.2. Métodos

Nesta seção serão descritos os métodos empregados no projeto até o desenvolvimento do protótipo. Cada módulo citado anteriormente foi testado individualmente, considerando suas particularidades. Os softwares foram implementados tendo como base técnicas de engenharia de software até sua etapa de teste.

3.2.1. Desenvolvimento utilizando a placa Intel Galileo

A primeira etapa desenvolvida para aplicação da placa Intel Galileo foi garantir o correto funcionamento do módulo. Inicialmente deve-se atualizar o *firmware* da placa para corrigir eventuais problemas de fábrica e garantir que o módulo opere com o melhor desempenho e integridade. A versão da IDE Arduino recomendada pelo fabricante é Arduino Software 1.5.3 ou posterior. Entretanto, a versão 1.5.3 (para Microsoft Windows) apresentou incompatibilidade com o idioma do sistema operacional, sendo necessário alterá-lo para Inglês. A versão Arduino Software 1.6 corrigiu esse problema e funciona em qualquer idioma de sistema operacional. Nesta nova IDE, deve-se instalar a placa acessando o menu “Ferramentas->Placa->BoardManager...” e instalar “intel i586 Boards”.

Com a placa em suas perfeitas condições e atualizada, foi realizado um estudo sobre as distribuições e versões de Linux disponíveis, atentando-se para aquelas que dispõem dos programas propostos para o funcionamento do projeto como Python, OpenCV e que permitem o funcionamento concomitante dos programas Arduino.

Definido a distribuição Linux a ser utilizada, foi realizado e apresentado todos os procedimentos de configuração da placa como, configuração de rede com IP estático, Servidor DNS, acesso SSH, Nameserver e apresentada as formas de acesso ao Linux instalado na placa Intel Galileo.

Para aferir o funcionamento da placa bem como sua interação com câmeras IPs, foi realizada a implementação de uma página Web (*Web Server*), utilizando programação Arduino (S*ketch*), com a finalidade de se analisar a velocidade e qualidade de exibição das imagens da câmera por meio de um link PHP.

- **Distribuição das imagens Linux disponíveis para Galileo:**

Existem várias imagens baseadas em Yocto compiladas para a placa de desenvolvimento Intel Galileo. A compilação Yocto pode ser ou à base de uclibc ou eglibc. Ambos são implementações da biblioteca libc fundacional encontrados em sistemas baseados em Linux embarcado. A uClibc é menor (razão que é padrão - de modo que toda

a imagem com firmware UEFI e Grub são instalados em 8 MB de memória flash SPI da Galileo), enquanto eglibc é maior, com recurso completo e tem maior grau de compatibilidade com libc.

Geralmente, apenas imagens baseadas em uClibc trabalha com Arduino IDE, com algumas exceções que serão mencionadas. Imagens baseadas em EGLIBC, por outro lado, em geral, são mais fáceis de usar quando você quer compilar software Linux, devido a uma maior compatibilidade com o padrão libc.

Neste projeto foram testados quatro sistemas operacionais (distribuições Linux) distintos, sendo a versão do Linux Yocto de fábrica, a “Bigger Linux Image” fornecida pela Intel e definida como a imagem completa, uma versão conhecida como IoTDev Kit Intel e uma versão alternativa compilada por um usuário com base Debian.

Dentre as opções testadas a que apresentou maior estabilidade e facilidade de implementação foi a IoTDev Kit. As demais imagens apresentaram algumas limitações ou dificuldade principalmente na execução de programas com o uso da biblioteca OpenCV. A imagem IoTDev Kit apresentou o Linux Yocto de forma mais completa, por apresentar os comandos básicos do Linux e funcionalidades úteis como SSH, Python, Node.js e OpenCV [28]. Esta imagem também permite o funcionamento da programação Arduino em conjunto com o Linux.

- **Acessando a Galileo**

Existem diversas formas de se acessar o sistema de arquivos Linux da Galileo. Pode acessar por comunicação serial, ou acessar pela porta USB ou via SSH com IP dinâmico.

O acesso por comunicação serial requer o uso do cabo P2/DB9 e de algum programa de comunicação serial instalado no computador a ser conectado.

Para acessar o Linux utilizando a plataforma Arduino e a comunicação USB, pode-se criar um programa (sketch) de interface e posteriormente conectar-se utilizando programas de comunicação serial. O programa utilizado como exemplo é apresentado no Apêndice deste trabalho (Algoritmo A - 1).

Outra forma de acesso ao Linux é por meio de uma conexão SSH, neste caso, deve-se encontrar o IP dinâmico da placa, para isso, pode-se implementar um programa Arduino (sketch) para apresentar as configurações de rede, dentre elas o IP dinâmico. De posse do IP dinâmico, será possível realizar conexão via SSH. No primeiro acesso o usuário

é “root” e não há senha. O referido programa é apresentado Apêndice deste trabalho (Algoritmo A - 2).

A opção utilizada no projeto foi realizar a primeira conexão por meio da comunicação utilizando o cabo serial (P2/DB9). Com acesso ao Linux, foi realizado toda a configuração de rede, alterando o acesso para IP estático. Configurado como IP estático, todos os demais acessos foram realizados utilizando SSH.

- **Configurações de Rede**

A configuração de rede é um pouco diferente da realizada no padrão Linux Debian. Não existe o arquivo “interfaces”, neste caso é necessário configurar o sistema Connman conforme se segue.

Acesse o diretório “/usr/lib/connman/test” e execute “./get-services”. Esse comando irá informar as configurações da rede, mas a informação que desejamos é o nome do serviço ethernet descrito nas informações geradas (ethernet_XXXXXXX_cable).

```
cd /usr/lib/connman/test
./get-services
```

- **Configurando Nameservers**

No diretório “/usr/lib/connman/test” execute o comando “./set-nameservers” dessa forma será informado a sintaxe do comando para configuração (/set-nameservers <service> [nameserver*]). Foi inserido, portanto, os respectivos dados, por exemplo:

```
cd /usr/lib/connman/test
./set-nameservers ethernet_c8a030ab323a_cable 143.107.225.6 143.107.182.2 8.8.8.8
```

- **Configurando IP estático**

No diretório “/usr/lib/connman/test” execute o comando “./set-ipv4-method”, dessa forma será informado a sintaxe do comando (./set-ipv4-method <service> [off|dhcp|manual <address> [netmask] [gateway]]).

Deve-se executar o comando de acordo com suas configurações, por exemplo:

```
cd /usr/lib/connman/test
./set-ipv4-method ethernet_c8a030ab323a_cable manual 10.235.10.XX 255.255.255.0 10.235.10.1
```

- **Configuração Servidor DNS**

Para utilizar corretamente o opkg (repositórios) é necessário corrigir o servidor DNS, para isso acesse o arquivo resolv.conf

```
vi /etc/resolv.conf
```

E inclua o endereço dos servidores, para o exemplo foram utilizados os seguintes endereços:

```
nameserver 143.107.225.6
 nameserver 143.107.182.2
 nameserver 8.8.8.8
```

Todavia, a cada reinicialização este arquivo é reescrito, dessa forma para solucionar essa limitação criei um script de inicialização para configurar automaticamente o servidor DNS.

- **Script de inicialização**

O script de inicialização segue um padrão diferente do utilizado no Linux de distribuição Debian.

Deve-se criar um script de acordo com a função desejada. Para o caso da configuração do servidor DNS foi criado um arquivo denominado resolv1.conf com a configuração correta do Nameserver. Ao ser executado esse script exclui o arquivo resolv.conf e copia o arquivo resolv1.conf alterando o nome para resolv.conf, dessa forma, como o Linux possui seu sistema de configuração baseado em arquivos, a substituição dos arquivos resultará na configuração correta do Nameserver.

O script de inicialização foi denominado de copy_ns e salvo em /home/root. A primeira linha é o cabeçalho indicativo de script. A Linha 2 exclui o arquivo resolv.conf. A terceira linha copia o arquivo resolv1.conf para o local “/etc/” com o nome de resolv.conf.

Arquivo resolv1.conf

```
nameserver 143.107.225.6
nameserver 143.107.182.2
nameserver 8.8.8.8
```

Script de atualização do nameserver denominado de copy_ns:

```
#!/bin/sh
rm /etc/resolv.conf
cp /etc/resolv1.conf /etc/resolv.conf
```

Após a criação do é necessário transformá-lo em um executável, para isso, pode-se utilizar o comando Linux “chmod +x”:

Posteriormente deve-se executar o comando “chmod 777” para garantir completa permissão ao arquivo

```
chmod +x copy_ns
chmod 777 copy_ns
```

Nesta etapa o script já está funcionando e para executá-lo e testá-lo basta utilizar um terminal com o seguinte comando:

```
./copy_ns
```

Com o *script* funcionando foi necessário incluí-lo na inicialização do sistema. Para isso, foi necessário criar um *script* de inicialização que executa o *script* copy_ns quando o sistema é inicializado. Este script de inicialização deve ser criado no diretório “/lib/systemd/system”. Para o caso exemplificado, o script de inicialização foi denominado myscript.service e seu conteúdo é apresentado a seguir:

Script de inicialização denominado myscript.service e salvo no diretório “/lib/systemd/system”.

```
[Unit]
Description=Script de inicializacao
After=getty.target

[Service]
ExecStart=/home/root/copy_ns

[Install]
WantedBy=multi-user.target
```

Neste arquivo é identificado a descrição da inicialização, sua ordem de execução (neste caso o script deve ser executado depois de getty.target). Também é apresentado o que esse script deve executar, que para o caso exemplificado é o script copy_ns. Por fim é indicado que o script deve ser executado para todos os usuários.

Após salvar as alterações no script “myscript.service” deve-se executar os seguintes comandos para inclui-lo na lista de inicializações, iniciá-lo e habilitá-lo respectivamente:

```
systemctl daemon-reload
systemctl start myscript.service
systemctl enable myscript.service
```

- **Configuração da porta para acesso SSH**

Para alterar a porta de conexão padrão 22 é necessário editar o arquivo “/lib/systemd/system/sshd.socket” modificando o parâmetro 22 para o número desejado na linha ListenStream.

Para reiniciar o ssh basta utilizar o seguinte comando “systemctl daemon-reload” ou reiniciar a placa.

3.2.2. Medidas de corrente consumidas pelos módulos

Para realizar a medida da corrente consumida pelos módulos, foi aberto o circuito de alimentação e medido a corrente na saída da fonte chaveada, utilizando o multímetro Minipa ET-2940, na escala de Ampere.

3.2.3. Uso de Câmeras IPs.

Para realizar testes com a câmera foi necessário realizar pesquisas sobre comandos CGI (Common Gateway Interface) [51]. Estes comandos permitem acessar e controlar as câmeras IPs utilizando comandos por URL. Este conhecimento é fundamental para realizar o acesso e a aquisição de imagens por meio da linguagem de programação escolhida.

As funcionalidades e acesso a câmera IP disponível foram estudadas e suas funcionalidades básicas foram testadas e executadas utilizando comandos por URL. Essas funcionalidades estarão presentes no código Arduino que controlará a câmera.

3.2.4. Estudo e aplicação do OpenCV

Inicialmente os programas foram implementados em um computador para permitir que os resultados gráficos fossem analisados. Posteriormente os códigos foram portados para a placa Intel Galileo, porém funções de exibição gráfica foram removidas (funções como “*imshow()”* foram removidas da execução do código).

O primeiro programa, implementado em computador, aplicou um filtro detector de borda e exibiu uma imagem original e outra apenas com os contornos. Este processamento e apresentação de vídeo foi realizado em tempo de execução.

O segundo programa apresentou subtração de imagens e mecanismos para detecção de movimento e indicação gráfica das regiões que se moveram, permitindo assim, identificar possíveis ruídos que posteriormente seriam tratados. Este programa serviu de base para identificação do posicionamento do idoso.

O terceiro programa realizou a mesma função do anterior, porém utilizando a câmera IP e atualizando a imagem de referência para compensar alterações ambientais que não se configuravam como movimento.

Com o correto funcionamento do programa em um computador este pode ser migrado para a placa de desenvolvimento Intel Galileo.

Estes desenvolvimentos foram acompanhados de testes de desempenho para aferir as limitações da placa.

3.2.5. Sistema de detecção de movimento

Para o projeto proposto foi necessário o reconhecimento de movimentos utilizando câmeras. Para esta finalidade o kit de desenvolvimento Intel Galileo com Linux IoT instalado apresentou um ambiente favorável ao desenvolvimento por apresenta Python e OpenCV já instalados. Fora necessário, entretanto, a instalação de outras bibliotecas como a biblioteca numpy, urlib2, imutils e argparse.

O primeiro desafio dessa implementação foi a aquisição de vídeo de câmeras IPs utilizando a biblioteca OpenCV. Inicialmente, exibir e processar vídeo de uma câmera USB local era simples, bastava utilizar o comando:

```
camera = cv2.VideoCapture(0)
```

Em que “camera” era a variável que receberia o vídeo.

Entretanto, contrariando algumas informações disponíveis na internet, não fora possível obter o vídeo diretamente da câmera IP com o comando citado. Para solucionar essa limitação foi utilizado um algoritmo para analisar o fluxo de frames sem depender da biblioteca OpenCV.

A solução foi inicialmente implementada em um código mais simples apenas para abrir e exibir o vídeo da câmera IP em um computador, conforme é apresentado na Figura 27 e no Algoritmo 1.

```
import cv2
import urllib2
import numpy as np

stream=urllib2.urlopen('http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha
&url=video.mjpeg')
bytes=''
while True:
    bytes+=stream.read(1024)
    a = bytes.find('\xff\xd8')
    b = bytes.find('\xff\xd9')
    if a!=-1 and b!=-1:
        jpg = bytes[a:b+2]
        bytes= bytes[b+2:]
        i = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
        cv2.imshow('i',i)
        if cv2.waitKey(1) ==27:
            exit(0)
```

Algoritmo 1: Aquisição de imagens da câmera IP

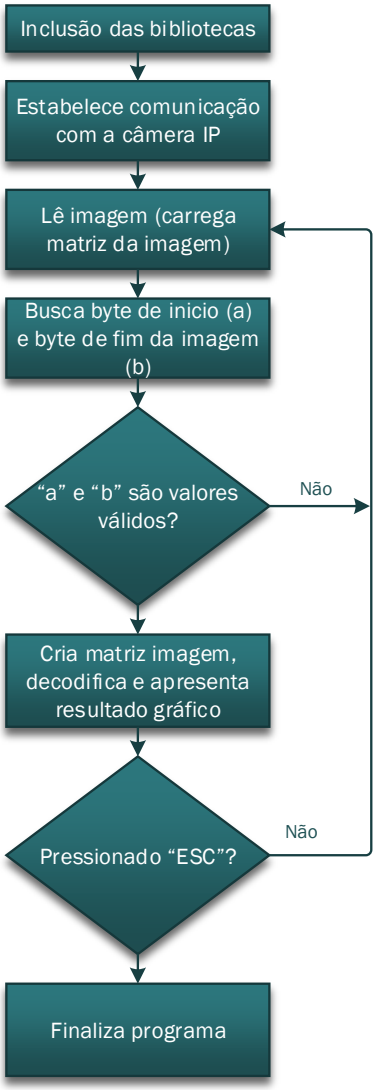


Figura 27: Fluxograma referente ao Algoritmo 1 - aquisição de imagens da camera IP

Todos os quadros de jpeg começam com o marcador "0xff 0xd8" e terminam com "0xff 0xd9" [52]. Assim, o código apresentado em Algoritmo 1 extrai tais quadros do fluxo de http e decodifica-os individualmente. Após utilizar esse método, foi possível realizar normalmente o processamento de imagens com a biblioteca OpenCV.

Para estimar o tempo de processamento de construção dos frames foi utilizado o algoritmo apresentado no Apêndice deste trabalho (Algoritmo A - 3), que apresenta o tempo decorrido a cada 100 frames processados.

Com o uso da biblioteca OpenCV, foi possível realizar a subtração de frames e determinado um limiar de alterações nas imagens para identificar que uma pessoa passou pela câmera. Inicialmente o teste foi realizado em um computador com plataforma gráfica

para visualizar o resultado gráfico do processamento da imagem, posteriormente o código foi portado para a placa de desenvolvimento Intel Galileo. A Figura 29 apresenta o Fluxograma completo do código implementado. O Algoritmo 4 apresenta o código para processamento de imagem e detecção de movimento, sendo este, parte do código disponibilizado no Apêndice deste trabalho (Algoritmo A - 4).

Este código apresentou o resultado da aquisição de uma webcam instalada em um computador, porém, este pode ser portado para permitir detecção de movimento utilizando câmeras IP.

Para realizar a aquisição de imagens da câmera IP utilizando a linguagem de programação Python, é necessário a inclusão da biblioteca “urlib2” e a implementação do código para estabelecer comunicação com a câmera IP (Algoritmo 2). O Algoritmo 2 apresenta o código de acesso utilizado para aquisição de frames no formato jpeg, neste, é necessário fornecer o nome de usuário e a senha de acesso a câmera IP.

A detecção de movimento realizada até esta etapa, utilizou subtração de imagens e análise do resultado dessa subtração. Para tornar o sistema mais eficaz foi necessária a atualização da figura de referência (“*fisrtFrame*”). Para suprir essa necessidade, foi incluído no código a atualização da figura de referência a cada 100 imagens processadas (Algoritmo 3). O Algoritmo A - 5 apresentado no Apêndice deste trabalho realiza o processamento de imagens utilizando acesso a câmera IP e detecção de movimentos atualizando a imagem de referência a cada 100 imagens processadas.

```
stream=urllib2.urlopen('http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha
&url=video.mjpeg')
bytes=''
```

Algoritmo 2: Trecho do código utilizado para aquisição dos dados (frames) da câmera IP

```
    if cont > 100:
        firstFrame = None
        cont = 0

    if firstFrame is None:
        firstFrame = gray
        continue

    cont = cont+1
```

Algoritmo 3: Trecho do código incluído no Algoritmo A - 4 utilizado para atualização da imagem de referência a cada 100 frames

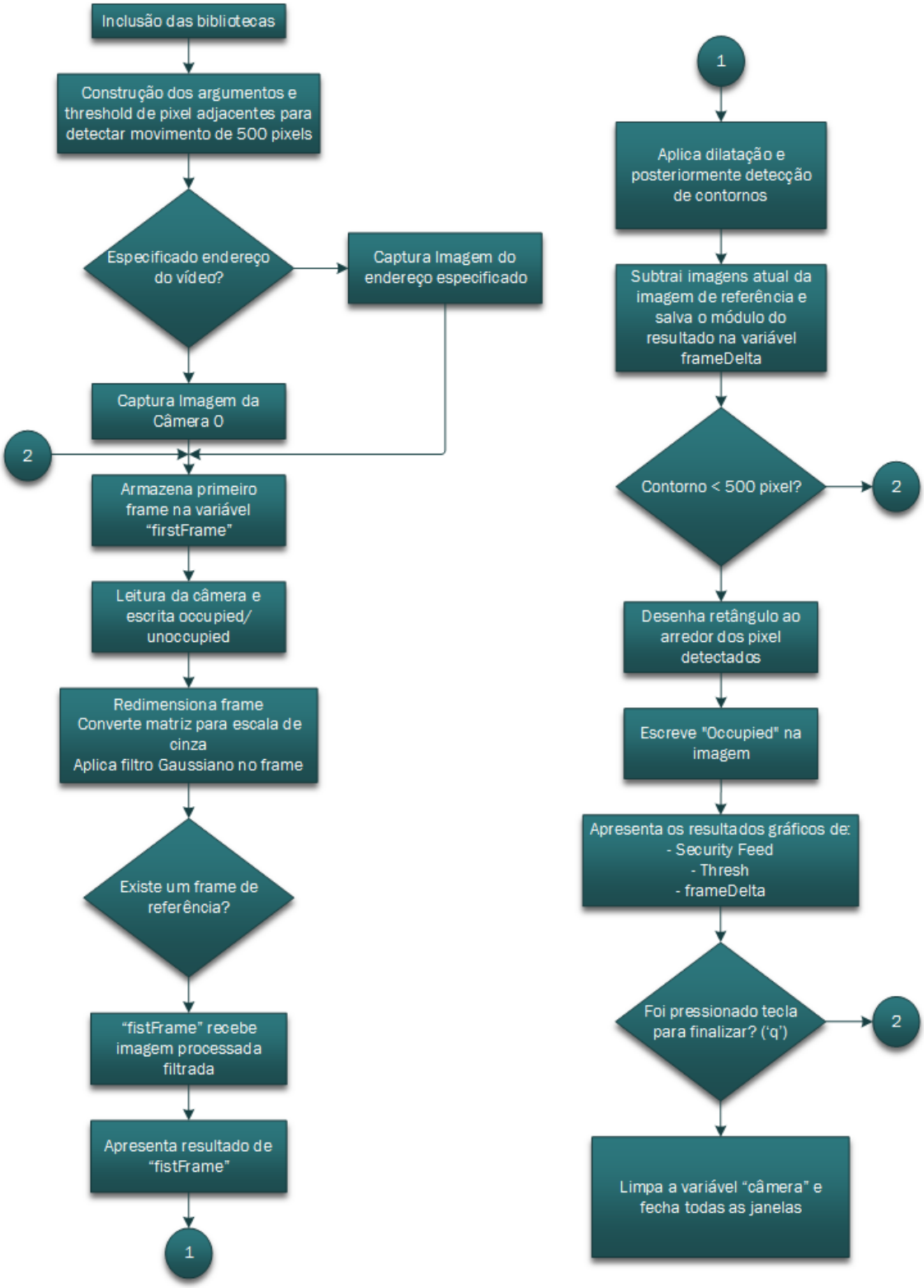


Figura 28: Fluxograma para do algoritmo para detecção de movimento

Os trechos de códigos utilizado para processamento de imagem estão em destaque no Algoritmo 4:

```
# loop de verificação
while True:
        # Leitura da camera e escrita occupied/unoccupied
        (grabbed, frame) = camera.read()
        text = "Unoccupied"

        # Redimensionamento do quadro, conversão para tons de cinza, aplicado filtro
        frame = imutils.resize(frame, width=500)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (21, 21), 0)

        # Inicializa a primeira imagem caso esta já não tenha inicializado
        if firstFrame is None:
                firstFrame = gray
                continue
        cv2.imshow("firstFrame", firstFrame)

      # Compara a imagem atual e a imagem salva em firstFrame
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)

        # Aplica dilatação e posteriormente detecção de contornos
        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                cv2.CHAIN_APPROX_SIMPLE)

        # Verificação dos contornos
        for c in cnts:
                # Se o contorno for pequeno, ignore-o
                if cv2.contourArea(c) < args["min_area"]:
                        continue
# Calcular a caixa delimitadora do movimento, desenhá-la no frame e atualizar texto
                (x, y, w, h) = cv2.boundingRect(c)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                text = "Occupied"

        # Desenha o texto e as informações de data e hora na imagem
        cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
                (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0,
255), 1)
        # Apresenta os resultados gráficos
        cv2.imshow("Security Feed", frame)
        cv2.imshow("Thresh", thresh)
        cv2.imshow("Frame Delta", frameDelta)
```

Algoritmo 4: Trecho de algoritmo implementado para detecção de movimento.

Após aferir o funcionamento do algoritmo, este pôde ser portado para a placa Intel Galileo. Neste caso, todas as operações gráficas utilizadas para analisar o resultado foram excluídas do processo de compilação (comentados).

Após esta etapa, o código foi otimizado com a redução de funções de processamento de imagem e utilização de classes. Para a comunicação entre o algoritmo de detecção de movimento e o *WebServer*, foi necessário realizar escrita em arquivos utilizando a linguagem de programação Python.

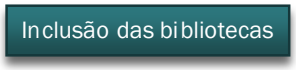
A velocidade de comunicação entre o Linux e o Arduino não foi significativo para atrasos menores que 1 segundo para a aplicação proposta, dessa forma, optou-se pela utilização desse método de comunicação por registrar o último evento e não depender de interrupções para ocorrer a comunicação.

O Fluxograma e o algoritmo são apresentados na Figura 29 e no Apêndice (Algoritmo A - 7) respectivamente. O Fluxograma exibido na Figura 23 é apresentado em blocos, pois o módulo *Threading* permite execução concorrente de parte do código. Essa execução "simultânea" é uma abstração do módulo e da linguagem Python, sabe-se que se analisado em um nível mais próximo do hardware as execuções são sequenciais devido a arquitetura do processador.

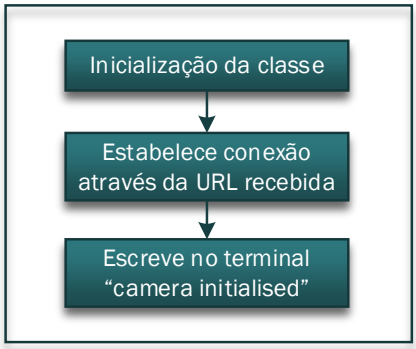
O Algoritmo 5 apresenta a escrita em um arquivo de texto e armazenamento deste no cartão SD. O trecho apresentado escreve "Cam01" em um arquivo de texto disponível no diretório "/media/mmcblk0p1/".

```
for c in cnts:
  if cv2.contourArea(c) < args["min_area"]:
     continue
  arq = open("/media/mmcblk0p1/test.txt", 'w')
  texto = []
  texto.append('Cam01\n')
  arq.writelines(texto)
  arq.close()
```

Algoritmo 5: Escrita em arquivo indicando a câmera que detectou movimento

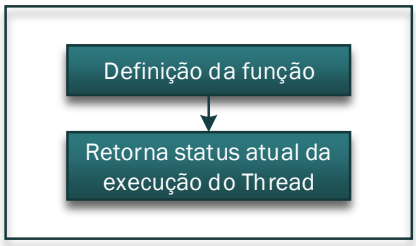


def __init__(self, url):

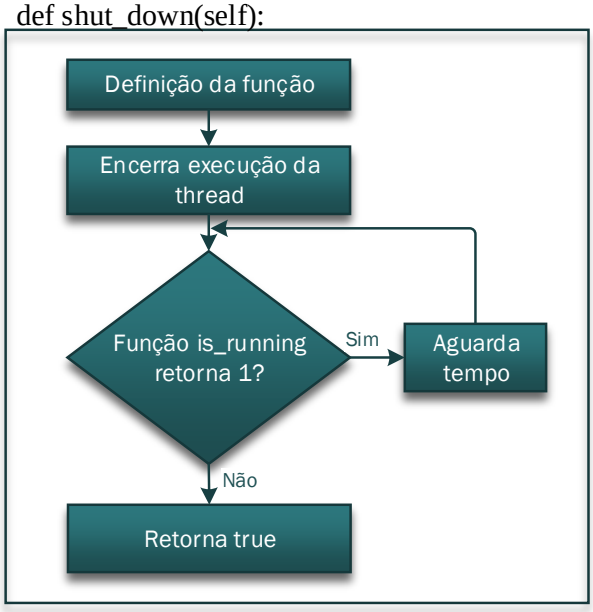


def start(self):

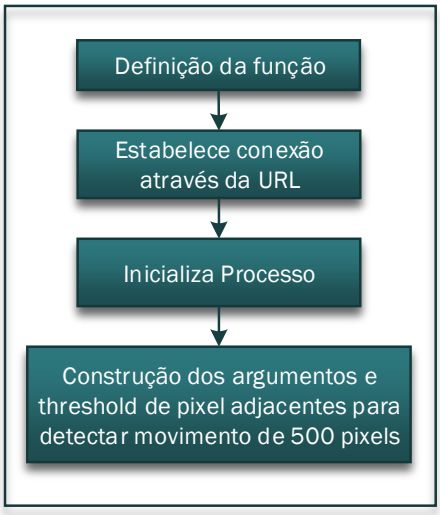
def is_running(self):



def shut_down(self):



if __name__ == "__main__"



def run(self):

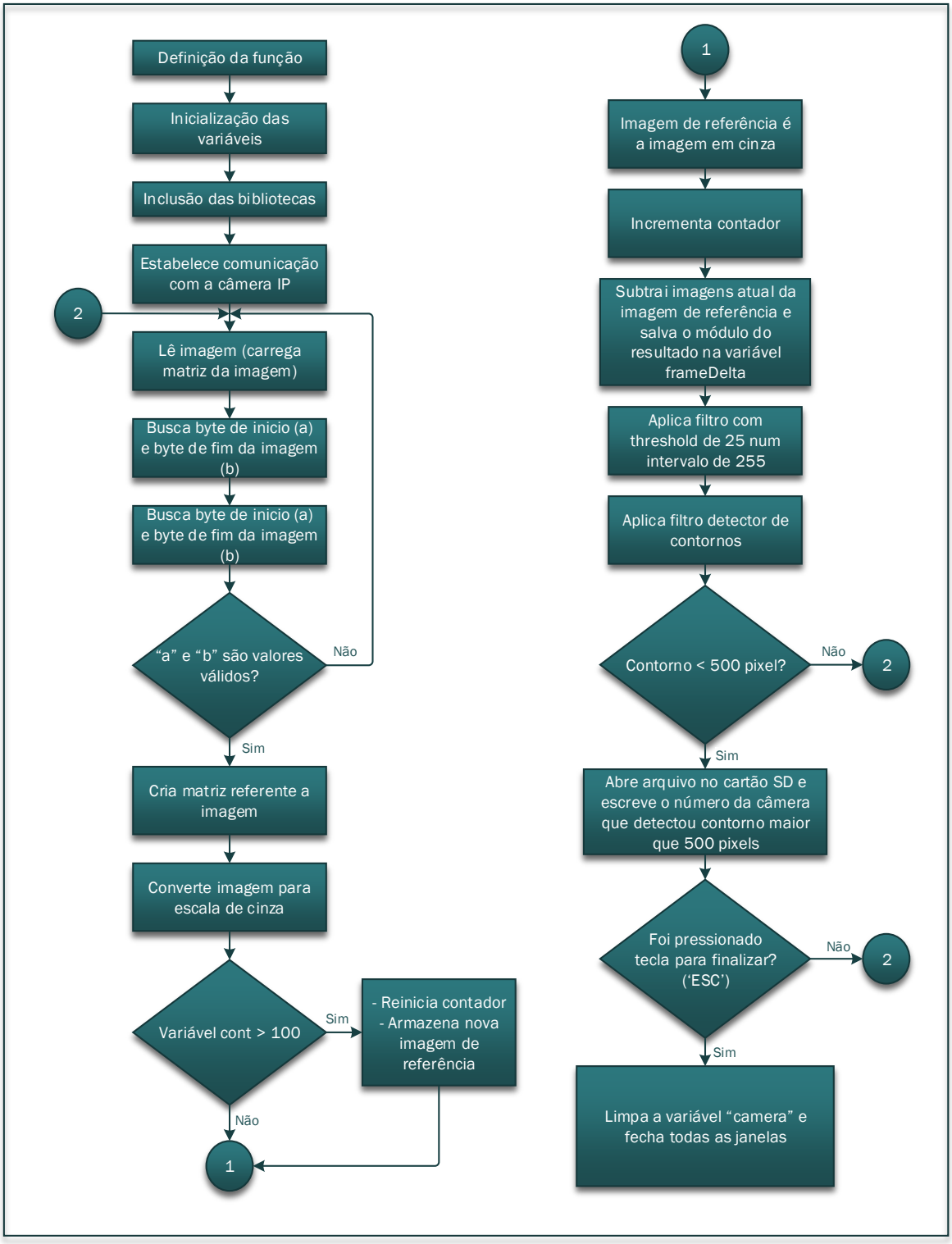


Figura 29: Fluxograma completo de detecção de movimento e escrita em arquivo.

O algoritmo apresentado no Apêndice (Algoritmo A - 7) executa o processamento de imagem e identificação de movimento (posição da pessoa). Para esse exemplo foram utilizadas duas câmeras IPs disponíveis no laboratório, porém, o sistema pode ser aplicado em um número maior de câmeras.

De forma análoga ao anterior, esse código armazena uma imagem (frame) e compara com os frames subsequentes para verificar se houve alteração, para isso, utiliza subtração de imagem. Essa imagem de referência é atualizada quando o contador atinge o número de 100 frames de aquisição. Após esse número uma nova imagem é utilizada como referência. Essa atualização da imagem de referência é necessária para considerar variações de luminosidade ou mesmo alguma alteração no ambiente, como por exemplo, a abertura de uma porta. Para evitar que ruídos de imagem sejam detectados como movimento, foi implementado um limiar de pixels alterados na imagem que determinarão a identificação de movimento. Neste código esse limiar foi de 500 pixels. Ao detectar movimento, o código registra a ação armazenando o respectivo número da câmera em um arquivo de texto.

3.2.6. Envio de mensagem ao responsável

Foram analisadas e implementadas duas formas de comunicação para envio de mensagem de texto. A primeira foi utilizando o módulo GSM que permite o envio de informações independentemente da internet. A segunda foi utilizando a biblioteca Yowsup da linguagem de programação Python.

- **Uso do módulo GSM para envio de mensagem de texto para celular do responsável**

O módulo GSM apresenta como vantagem sua independência da internet para o envio de mensagem. Seu sistema de comunicação não é limitado a uma rede Wi-Fi, mas abrange todo local com sinal com alcance da empresa telefônica em questão. Sua conexão a placa de desenvolvimento Arduino Mega2560 é apresentada na Tabela 4. Esta conexão é necessária além da conexão direta do acoplamento entre o Arduino Mega2560 e o *Shield*.

Tabela 5: Conexão entre Arduino Mega2560 e *Shield* GSM

Arduino Mega2560	Shield GSM
Rx1 (Pino 18) Arduino	Tx Shield
Tx1 (Pino 19) Arduino	Rx Shield

Para enviar a mensagem “Arduino SMS” para o número "+55199927779XX" por SMS, foi implementado o código apresentado em Algoritmo A - 9 (Apêndice).

- **Uso da biblioteca Python (yowsup) para envio de mensagem por WhatsApp [54]**

Inicialmente deve-se instalar a biblioteca pip do python. Esta biblioteca é um sistema de gerenciamento de pacotes Python. Sua usabilidade e funcionalidade é bem semelhante aos comandos apt-get em sistemas Debian, ou o opkg no Linux Yocto.

Para instalar a biblioteca pip foi realizado o download do mesmo no repositório (GitHub) e copiado para o cartão SD da Galileo. Posteriormente foi acessado a pasta com o arquivo setup.py e executado o comando de instalação:

```
python setup.py install
```

Após a instalação do gerenciado pip, o pacote Yowsup pode ser instalado pelo seguinte comando, usando agora o recém-instalado pip:

```
pip install yowsup2
```

O sistema do WhatsApp é todo via *WebServices*, ou seja, não faz uso de ligações ou mensagens SMS, usa pura e simplesmente a Internet. Mas a título de criar uma chave de registro única e vinculada a um número de telefone celular válido, o sistema do WhatsApp envia uma mensagem com um código de cadastro para o número de telefone a ser utilizado no serviço.

Foi criado um arquivo com as configurações, dados de códigos, chaves e denominado de whatsapp.config. Este arquivo contém as seguintes configurações:

```
cc=55 #Brasil coloca 55
mcc=724 # Mobile Country Code do Brasil
mnc=05 # Mobile Network Code da Claro no Brasil
phone=551999514xxxx #Codigo país + DDD + Numero de telefone
id=0000000000 #Deixa assim, sem problema.
password= ##########################.
```

O pass*word é ob*tido enviando a requisição ao serviço do WhatsApp, dessa forma deve-se executar o seguinte comando:

```
yowsup-cli registration -c whatsapp.config --requestcode sms
```

No comando acima, usamos o yowsup-cli, que é um utilitário em linha de comando da aplicação yowsup. Neste utilitário, são utilizadas algumas funções como a *registration*, que dispara as chamadas necessárias para requisitar o código. Juntamente com essa função, é preciso passar o arquivo de configuração, por meio dos parâmetros -c whatsapp.config e depois especificar que queremos um código via SMS com os parâmetros --requestcode sms.

Ao receber o código por SMS, basta registrá-lo no sistema com o comando:

```
yowsup-cli registration -R XXX-XXX -c whatsapp.config
```

Após executado o comando acima, obteremos como resposta algumas informações do status do registro e o *password* de acesso ao serviço WhatsApp. Este *password* pode ser copiado para o campo *password* no arquivo whatsapp.config.

Nesta etapa o programa enviou corretamente mensagem ao celular ao executar o comando:

```
yowsup-cli demos -s 551999277xxxx "Teste Galileo" -c whatsapp.config
```

Esta funcionalidade será utilizada para enviar uma mensagem de WhatsApp ao celular do responsável da família sobre um potencial acidente.

3.2.7. Implementação dos acelerômetros

Os acelerômetros ADXL335 e MPU-6050 foram implementados individualmente e seus resultados analisados afim de identificar o mais indicado para o projeto, isto é, que apresentasse resposta rápida e que fosse pouco susceptível à ruídos.

- **Acelerômetro ADXL335**

O Acelerômetro ADXL335 apresentado é pequeno e de simples implementação pois sua resposta, conforme apresentado no tópico 3.1.6, é analógica. Desta forma, basta analisar o nível de tensão e definir o limiar de aceleração que indica impacto mecânico.

O código utilizado foi adaptado dos exemplos disponíveis para comunicação entre o Arduino e este módulo. Sua distribuição é de domínio público. Neste código o *threshold* foi de 1.0 V para os eixos x, ou y ou z. Se qualquer destes apresentarem aceleração maior que este valor, um alarme é acionado. Para o teste em banca foi utilizado um LED

sinalizador. A sensibilidade do sensor é de 300mV/g, logo o limiar para acionamento do alarme foi de 32,7 m/s² o que empiricamente se mostrou um valor aceitável, correspondente a desaceleração um choque mecânico. Nota que neste código utilizou-se a variação da aceleração para determinar se houve um choque mecânico. O tempo de análise da iteração do código foi de 200 milissegundos. Não fora necessário nenhuma biblioteca especifica, bastou analisar os valores obtidos no conversor AD. O código completo é apresentado no Algoritmo A - 10 (Apêndice). O Algoritmo 6 apresenta a lógica principal para obtenção dos valores analógicos e cálculo da variação destes.

A Tabela 4 apresenta as respectivas conexões entre o Arduino Mega2560 e o acelerômetro ADXL335.

Tabela 6: Respectivas conexões entre Arduino Mega2560 e acelerometro ADXL335

Pinos Arduino Mega2560	Pinos ADXL335
A3	X-OUT
A4	Y-OUT
A5	Z-OUT

```
void loop()
{
  if(i == 0){
     X0 = analogRead(xpin); // Valor inicial do eixo “x”
     Y0 = analogRead(ypin); // Valor inicial do eixo “y”
     Z0 = analogRead(zpin); // Valor inicial do eixo “z”
     i++;
  }
  else{
    delay(dt);
     X1 = analogRead(xpin); // Valor final do eixo “x”
     Y1 = analogRead(ypin); // Valor final do eixo “y”
     Z1 = analogRead(zpin); // Valor final do eixo “z”
     i = 0;
// Cálculo da variação da aceleração
    dx = abs(X1-X0)/const_conv;
    dy = abs(Y1-Y0)/const_conv;
    dz = abs(Z1-Z0)/const_conv;

// alarme
     if(dx > 1.0 || dy > 1.0 || dz > 1.0){
       set_alarm = 1;
     }
     if (set_alarm == 1){
       digitalWrite(52, HIGH);
     }
  }
}
```

Algoritmo 6: Trecho para cálculo da variação da aceleração ADXL335

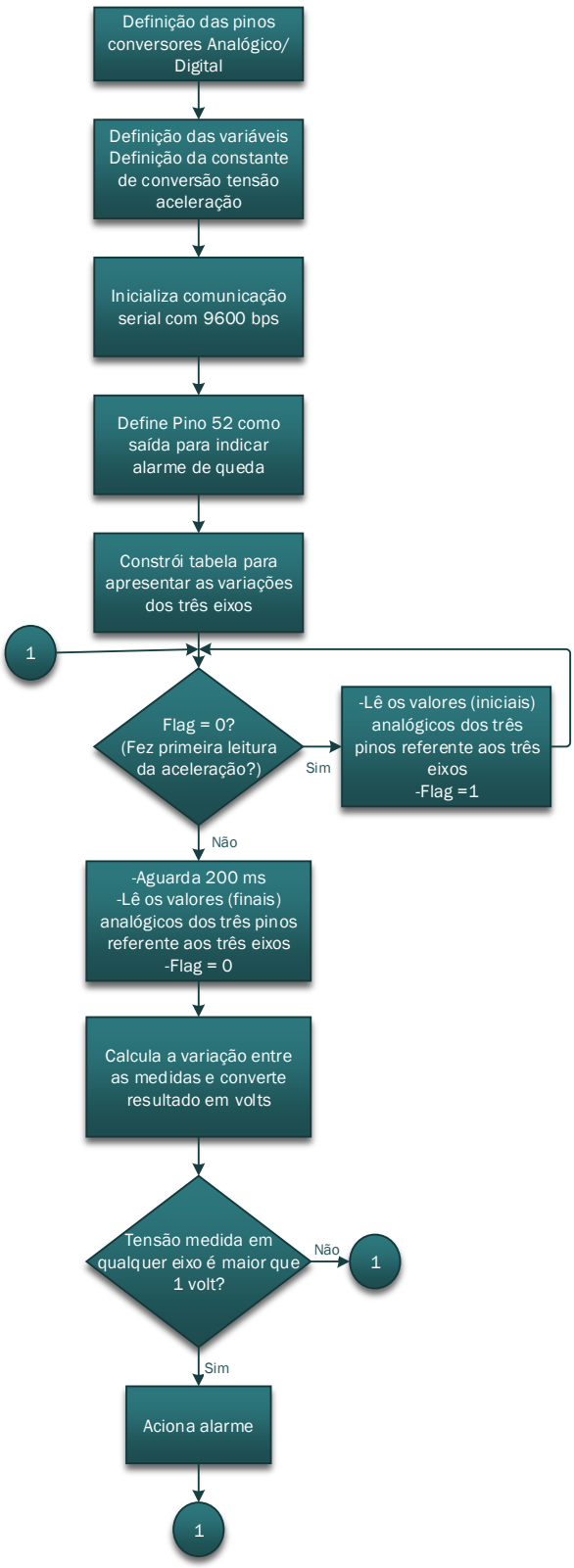


Figura 30: Fluxograma de leitura e atuação do acelerômetro ADXL335

- **Acelerômetro MPU-6050**

O acelerômetro MPU-6050, conforme descrito na seção 3.1.6 apresenta saída processada, com comunicação I²C. O programa apresentado abaixo (Algoritmo 7) também avalia a variação da aceleração. O limiar de sensibilidade foi avaliado empiricamente (em testes de bancada) e o resultado obtido foi o valor apresentado no Algoritmo 7. A implementação completa do código está apresentada no Algoritmo A - 11 (Apêndice).

Tabela 7: Respectivas conexões entre Arduino Mega2560 e acelerômetro MPU-6050

Pinos Arduino Mega2560	Pinos MPU-6050
Arduino Ground	MPU GND
Arduino A4 (SDA)	MPU SDA
Arduino A5(SCL)	MPU SCL
Arduino 3V3	MPU VCC

```
const int MPU=0x68; // I2C address of the MPU-6050

int AcSensitivity = 10000;
boolean moved = false;

void loop(){
//Estabelecendo comunicação I²C com o dispositivo
Wire.beginTransmission(MPU);
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU,14,true);

// Obtendo dados da aceleração
AcX=Wire.read()<<8|Wire.read();
AcY=Wire.read()<<8|Wire.read();
AcZ=Wire.read()<<8|Wire.read();

//Calculando variação da aceleração
dx = abs(AcX - OldAcX);
dy = abs(AcY - OldAcY);
dz = abs(AcZ - OldAcZ);

//Verificando se houve queda ou movimento
if (dx > AcSensitivity || dy > AcSensitivity || dz > AcSensitivity) {
moved = true;
}
if (moved == true) {
Serial.println("MOVED");
}
OldAcX = AcX;
```

```
OldAcY = AcY;
OldAcZ = AcZ;
moved = false;
delay(100);
}
```

Algoritmo 7: Leitura e cálculo da variação de aceleração

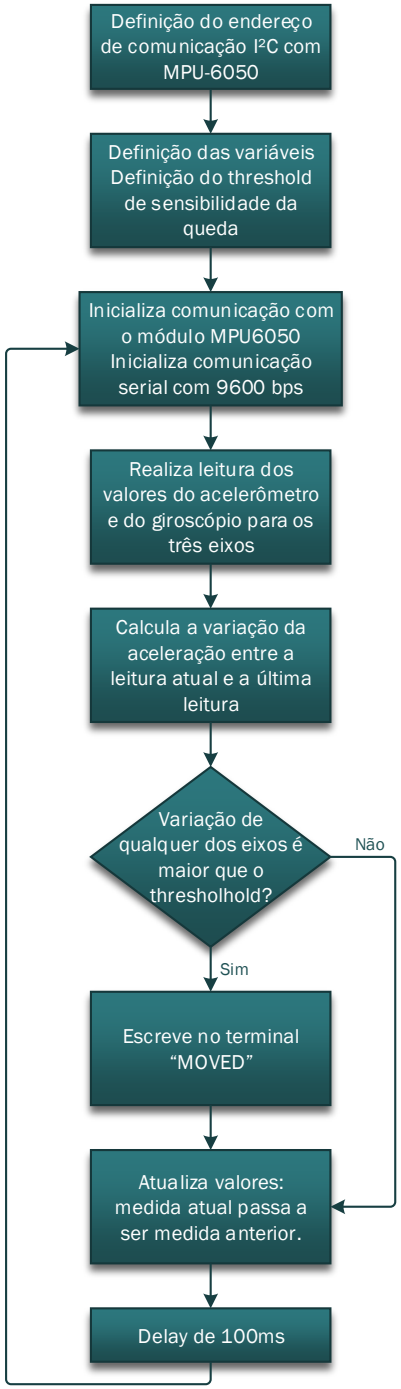


Figura 31: Fluxograma de leitura e atuação do acelerômetro MPU-6050

3.2.8. Implementação do Barômetro

O barômetro BMP180 foi implementado individualmente e seus resultados foram avaliados afim de comprovar a precisão indicada pelo fabricante, bem como sua aplicabilidade para o projeto proposto. Suas limitações e susceptibilidade a interferências externas também foram analisadas.

O módulo BMP180 apresenta como saída a pressão absoluta em pascais (Pa). Um Pascal é uma quantidade muito pequena de pressão, aproximadamente, a quantidade que uma folha de papel vai exercer apoiado sobre uma mesa. Dessa forma, a biblioteca Arduino fornece valores de saída já convertidos com ponto flutuante em hPa.

- **Compensação Térmica**

A temperatura afeta diretamente a densidade de um gás. A densidade afeta a massa do gás e este por sua vez altera drasticamente a pressão atmosférica.

Para compensar a temperatura, o BMP180 inclui um sensor de temperatura bastante preciso, assim como um sensor de pressão. Para obter o correto valor de altitude, primeiro deve-se tomar uma leitura de temperatura, em seguida, combiná-la com a leitura de pressão.

- **Determinando Altitude**

Como a pressão varia com a altitude, este sensor pode usar utilizado para medir a altitude (com algumas ressalvas).

A pressão média da atmosfera ao nível do mar é 1013,25 hPa (ou mbar). Este cai para zero à medida que sobe em direção ao vácuo do espaço. Porque a curva deste drop-off é bem compreendida, você pode calcular a diferença de altitude entre duas medições de pressão (p e p_0), usando a seguinte equação:

$$altitude = 44330 * \left(1 - \left(\frac{p}{p_0}\right)^{\frac{1}{5,255}}\right)$$

Em que p é a pressão medida e p_0 é a pressão de referência.

Há dois modos de operação:

1. Utilizando a pressão do nível do mar (hPa 1.013,25) como a pressão basal p_0 a saída da equação será a sua altitude atual acima do nível do mar.

2. Utilizando uma única leitura de pressão em sua posição atual, e definir este valor como referencial (p_0) todas as leituras de pressão subsequentes resultarão em relação mudanças de altitude a partir da linha de base.

Neste projeto, como será analisado queda, o modo de operação 2 é mais indicado por detectar variações a partir de um referencial inicial, compensando algumas variações naturais da pressão atmosférica que alteram o cálculo da altitude.

A Figura 32 apresenta o fluxograma utilizado para a implementação do programa disponibilizado no Algoritmo A - 12 (Apêndice).

Tabela 8: Respectivas conexões entre Arduino Mega2560 e barômetro BMP180

Pinos Arduino Mega2560	Pinos BMP180
Arduino Ground	BMP180 GND
Arduino A4 (SDA)	BMP SDA
Arduino A5(SCL)	BMP SCL
Arduino 3V3	BMP Vin

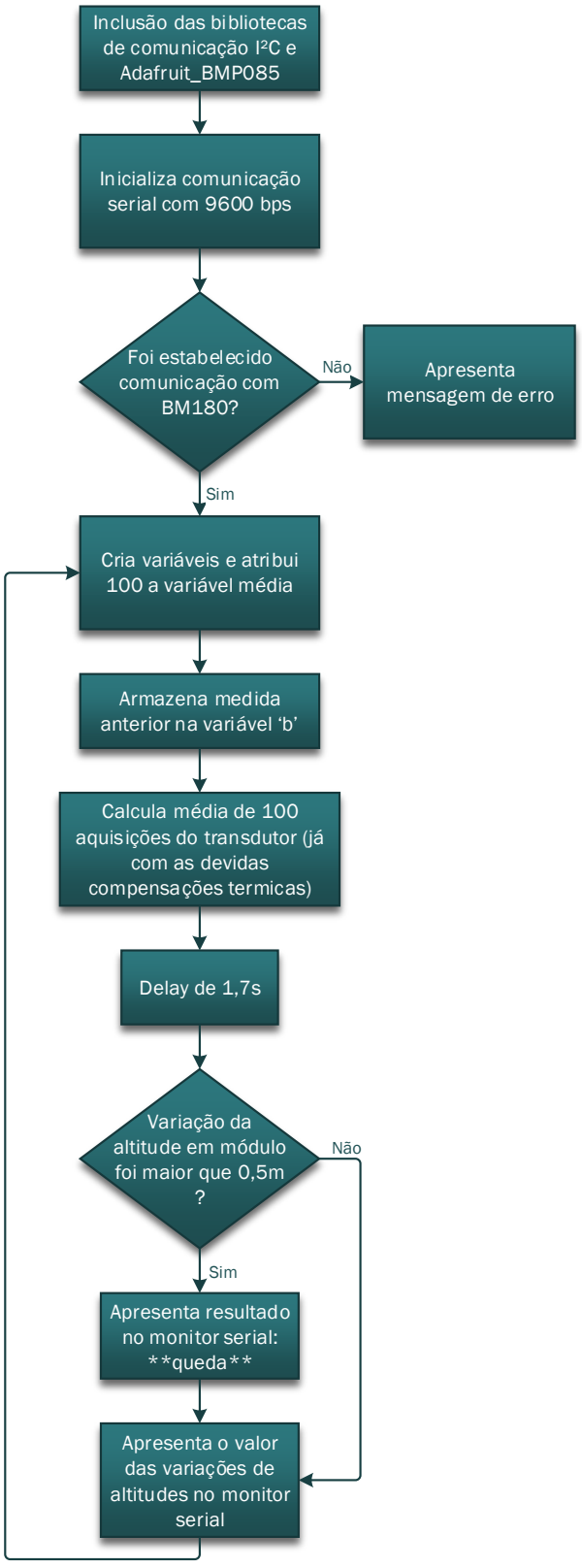


Figura 32: Fluxograma utilizado para elaboração do Algoritmo 7 - Cálculo da variação da altitude em função da pressão atmosférica

- **Observações gerais:**

Precisão: O nível de ruído teórico da resolução mais alta é de 0,25 m, embora, na prática, vemos ruído na ordem de 0,5 m. Esta precisão pode ser melhorada tomando um grande número de leituras e analisando suas médias, entretanto, isso vai abrandar a sua taxa de amostra e tempo de resposta. Para o projeto proposto um número muito grande de amostras irá prejudicar a detecção de uma queda, que se caracteriza pela variação abruta de altura.

Ambiente aberto: O BMP180 precisa de acesso ao ar ambiente para medir a sua pressão, desta forma, o local onde ele ficará não deve ser vedado do ambiente externo. Para o protótipo proposto o BMP180 ficou em ambiente aberto.

Não se deve incidir ventilação diretamente sobre o BMP180: A exposição ao ar ou vento em movimento rápido pode causar variações de pressão momentânea que vão afetar suas leituras. Deve-se, portanto, proteger o dispositivo a partir de correntes de ar fortes.

Mudanças bruscas de temperatura: O componente apesar de possuir compensação térmica pode apresentar erro de medida se houver mudanças bruscas de temperatura.

Evitar umidade: O BMP180 é sensível à umidade.

Influência da iluminação: O silício dentro do BMP180 é sensível à luz, que pode entrar no dispositivo através do orifício no topo do chip. Para máxima precisão, deve-se proteger o chip da luz ambiente [39].

3.2.9. Módulo ESP8266

Com a finalidade de testar a comunicação entre o módulo ESP8266 e a placa Intel Galileo, foi implementado um site (Web Server) para acionar um *buzzer* conectado a placa Intel Galileo. Como o teste se refere ao funcionamento da comunicação entre o módulo ESP8266 e a placa Intel Galileo, o sistema com *buzzer* foi testado com o auxílio de um botão criado no site que, quando pressionado, enviava a seguinte requisição URL e acionava o *buzzer*:

```
143.107.235.59:8152/?LED2;
```

Buscando simular este feito, foi implementado no módulo ESP8266 – 01 um sistema de requisição por meio da URL e testado com o acionamento do *buzzer*.

Neste trabalho foi utilizado o Arduino IDE para gravação dos programas no módulo ESP8266. Para gravar os programas e atualizar o firmware do módulo é necessário que o pino GPIO0 esteja previamente aterrado. Este pino pode ser terrado antes da placa ser energizada ou deve-se utilizar o pino de reset para reiniciar a placa com o pino GPIO0 aterrado conforme Tabela 9.

Para a gravação, utilizou-se um módulo conversor USB->UART.

Tabela 9: Conexão do módulo ESP-01 para gravação

Pino do Módulo ESP-01	Conexão USB UART
VCC	3V3
CH_PD	3V3
GPIO0	GND
GND	GND
Tx	Rx
Rx	Tx

A IDE que oferece suporte a programação do módulo ESP8266 é a versão 1.6.5. Após abrir esta versão do IDE deve-se acessar o menu arquivo/preferência e inserir a seguinte URL no campo "*Additional Boards Manager URLs*"

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Após adicionar a URL deve-se baixar a biblioteca referente ao modulo ESP8266. Para isso deve-se acessar o menu "Sketch/Include->Library/Manage->Libraries" e pesquisar e instalar a biblioteca ESP8266.

Após realizar esse processo a biblioteca já estará instalada. Deve-se então, selecionar o modelo do módulo.

A Figura 33 exibe o fluxograma do Algoritmo A - 13 (Apêndice) implementado no módulo ESP8266 -01 utilizado para comunicação dos transdutores à placa de desenvolvimento Intel Galileo

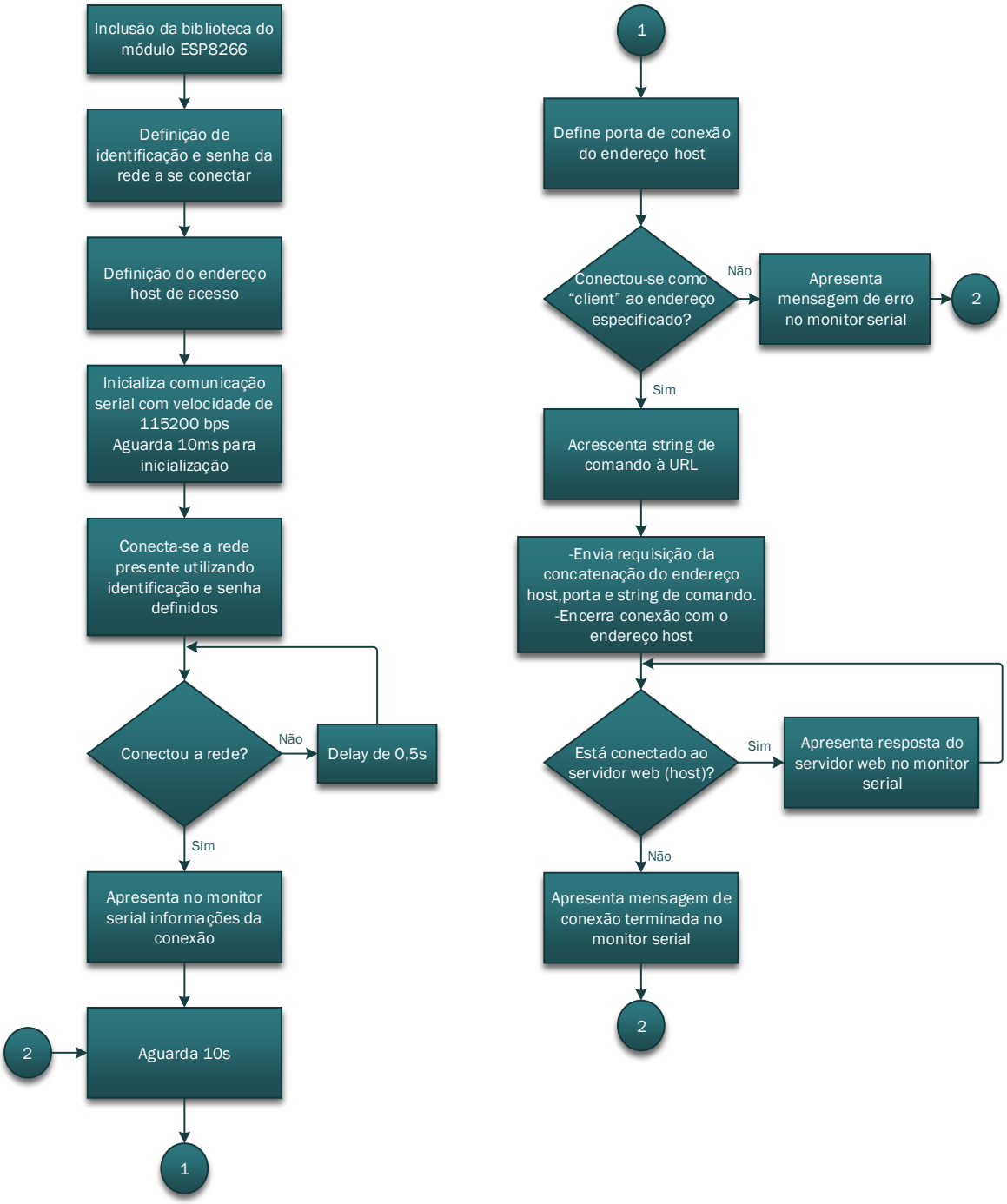


Figura 33: Fluxograma do programa do módulo ESP8266 utilizado para comunicação dos transdutores à placa de desenvolvimento Intel Galileo.

Com o programa apresentado no Algoritmo A - 13 do Apêndice foi possível acionar um *buzzer* conectado na placa Intel Galileo, confirmando a correta transmissão de informações.

3.2.10. Módulo ESP8266 para sensor de gás

Neste módulo do projeto foi estabelecida a comunicação entre o módulo Wi-Fi ESP8266-12 e o Arduino Mega2560. O módulo ESP8255-12 foi utilizado para monitorar o sensor de gás por meio da leitura de resposta (analógica) do sensor, neste caso, a saída do sensor será lida pino ADC do ESP8266-12. De forma análoga ao descrito no item 3.2.8, este sistema acionará o *buzzer* conectado a placa Intel Galileo por meio de requisição URL.

Para gravar o programa na memória flash, deve-se configurar os pinos de acordo com a Tabela 9.

O sistema de monitor de vazamento de gás funcionará com o mesmo princípio de comunicação anteriormente descrito. A diferença é apresentada logo no início do laço principal do programa. Neste trecho é avaliado se o sensor de gás detectou vazamento por meio da análise da saída analógica do transdutor. Foi definido um limiar para detecção do gás. Quando o mesmo é detectado, é enviado um comando (requisição) ao servidor web com o acréscimo "/?GAS;" na URL. Ao receber essa requisição a placa Intel Galileo se incumbirá de enviar uma mensagem de texto ao celular do responsável. O programa é análogo ao apresentado no Algoritmo A - 13 (Apêndice), difere apenas na verificação do valor analógico e na requisição do comando à URL, as diferenças estão apresentadas no trecho do Algoritmo 8. O algoritmo completo está apresentado no Algoritmo A - 14 (Apêndice).

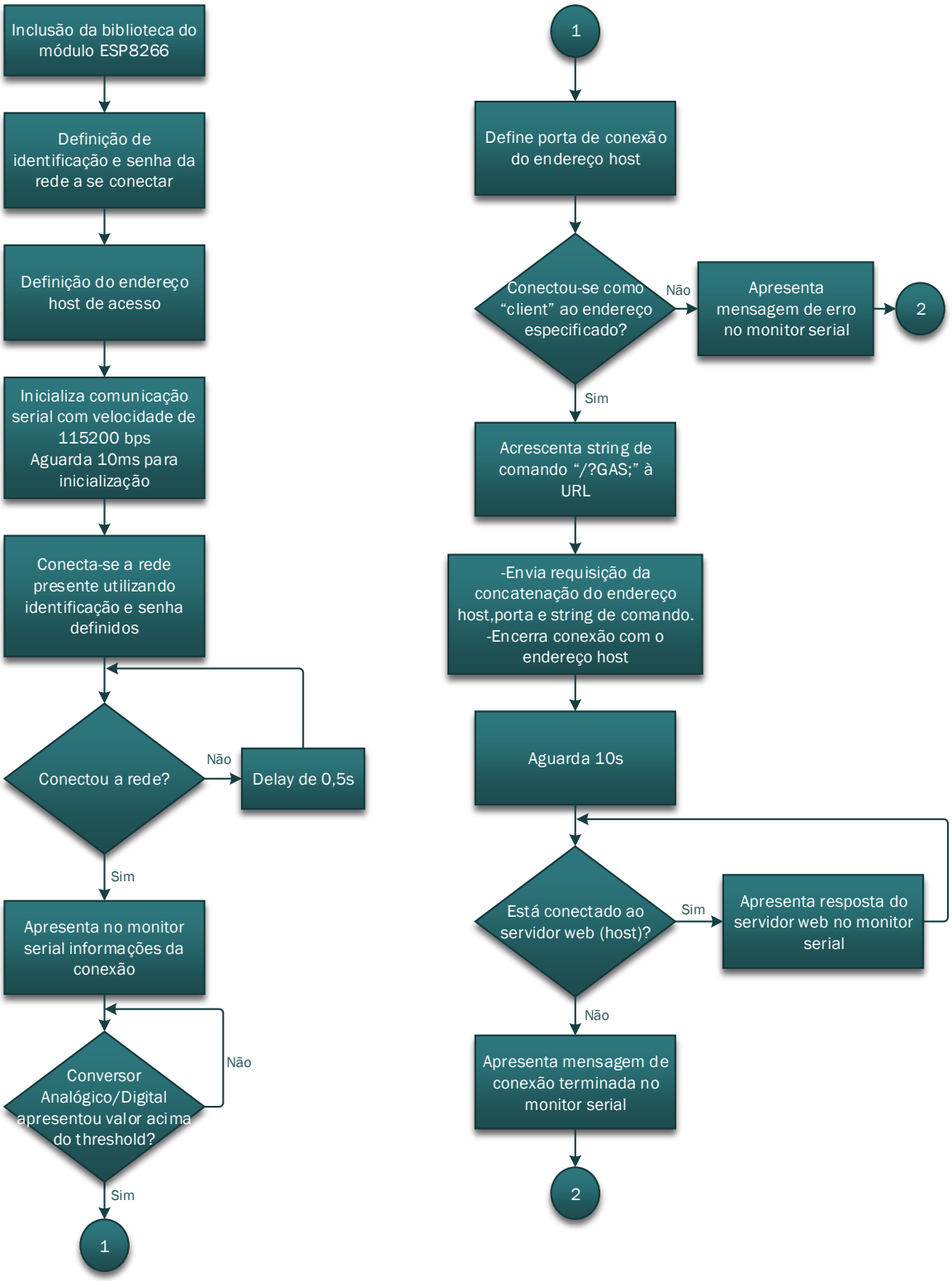


Figura 34: Fluxograma do módulo detector de gás

```
void loop() {
  if(analogRead(A0) > 600){
  Serial.print("connecting to ");
  Serial.println(host);

  // Conexão WiFiClient para conexão TCP
  WiFiClient client;
  const int httpPort = 8152;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // Concatenação da URL
  String url = "/?GAS;";
  Serial.print("Requesting URL: ");
  Serial.println(url);

  // Envio de requisição ao servidor
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10000);

  // Leitura da resposta do servidor
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
  }
}
```

Algoritmo 8: Trecho do algoritmo utilizado para comunicação entre o sensor de gás e a central de processamento Intel Galileo

O código apresentado no Algoritmo A - 14 (Apêndice) estabelece comunicação com uma rede Wi-Fi presente no ambiente em que se encontra. Para isso são informados a rede a qual se deve conectar e a respectiva senha (*ssid* e *password*). No programa, o módulo ESP8266 se conecta à rede e executa a função em um site cujo endereço é armazenado na variável “Host”.

3.2.11. Sistema de alarme e indicação de medicamento

Para este projeto foi utilizada as funcionalidades de programação por meio do Arduino. A parte de programação é simples por se basear em um sistema de relógio com alarmes e acionamentos de LEDs nos organizadores específicos.

Além do LED para indicar o compartimento do medicamento, também foi utilizado um display gráfico para a indicação da pessoa que deverá tomar o medicamento tornando mais intuitivo a ingestão do remédio e possibilitando a configuração do compartimento por uma interface *touchscreen*. Neste protótipo, foi executada inicialmente apenas as funcionalidades básicas de indicação do compartimento do medicamento, sem a criação de uma interface amigável para o usuário devido ao tempo restrito de execução do projeto. Entretanto, esta condição permitirá observar a funcionalidade técnica do dispositivo antes da aplicação de uma evolução.

A transferência dos arquivos e imagens ao display pode ser realizada por meio de um cartão SD. Para o controle das imagens foi utilizada a comunicação serial entre a placa Intel Galileo e o Display. As instruções de comando podem ser obtidas no manual de usuário do display [55].

O Algoritmo 9, utilizado para validação da viabilidade tecnológica verifica se o horário e data apresentados são iguais a “05/29/15 22:53”, caso seja, o LED 13 da placa Intel Galileo Pisca e o comando para acesso a primeira imagem do display é executado. A Figura 35 apresenta o fluxograma do sistema proposto.

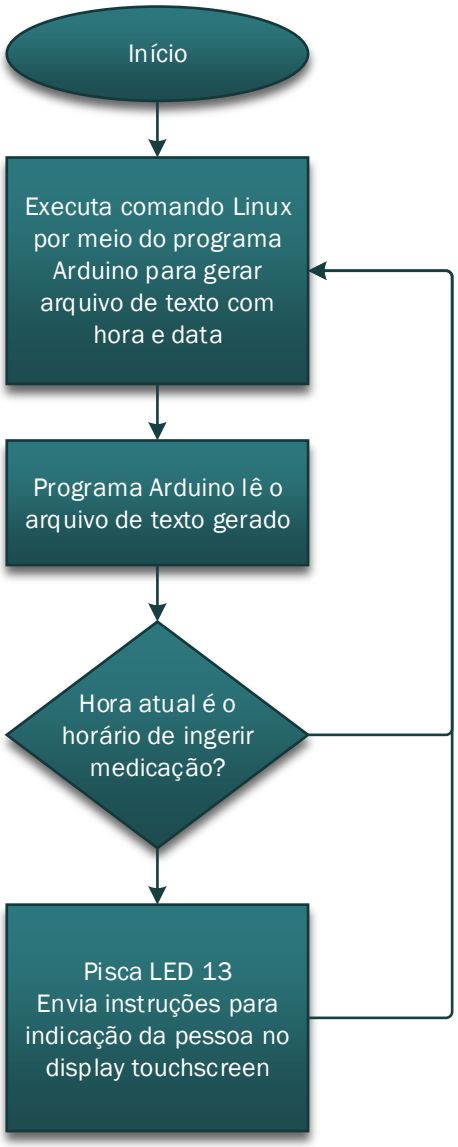


Figura 35: Diagrama do programa teste para gerenciamento da medicação

```
char buf[15];
void setup() {
  pinMode (13, OUTPUT);
  // initialize both serial ports:
  Serial.begin(115200);
  Serial1.begin(115200);
}

void loop() {
  // read from port 1, send to port 0:
  if (Serial1.available()) {
    char inByte = Serial1.read();
    Serial.print(inByte);
    Serial.write(inByte);
    Serial.print(inByte);
  }

  // read from port 0, send to port 1:
  if (Serial.available()) {
    char inByte = Serial.read();
  system("date '+%D %H:%M:%S' > /home/root/time.txt");
  FILE * fp;
  fp = fopen ("/home/root/time.txt", "r");
  fgets (buf, 15, fp);
  fclose (fp);
  Serial.print ("O momento atual eh ");
  Serial.println(buf);

  String stringOne = String(buf);
  String stringTwo = String("05/29/15 22:53");
  if(stringOne == stringTwo){
digitalWrite (13, HIGH); // set o LED
delay (1000); // esperar por um segundo
digitalWrite (13, LOW); // definir o off LED
delay (1000); // esperar por um segundo
    }
    Serial1.write(0x5A);
    Serial1.write(0xA5);
    Serial1.write(0x04);
    Serial1.write(0x80);
    Serial1.write(0x03);
    Serial1.write((byte)0x00);
    Serial1.write((byte)0x00);
  }
}
```

Algoritmo 9: Algoritmo utilizado para indicação do medicamento a ser ingerido

3.2.12. Integração Placa de desenvolvimento Intel Galileo e Linux

A placa de desenvolvimento Intel Galileo, por possuir todas as características de processamento citadas na seção 3.1.1., permite o uso de sistema operacional embarcado.

O sistema operacional gerencia os programas Arduinos como um processo Linux, sendo executado em paralelo aos processos Linux habituais [56]. A Figura 36 exemplifica o que fora explicado apresentando o resultado do gerenciador de processos do Linux. Neste, é possível observar a execução do programa Arduino como um de seus processos.

Vale ressaltar, no entanto, que existem diferentes imagens Linux compiladas para placa Intel Galileo. Também é possível criar a própria distribuição utilizando as ferramentas apropriadas do Linux Yocto, disponíveis e indicados em seu site [57].

```
Mem: 118048K used, 118824K free, 0K shrd, 35660K buff, 50552K cached
CPU:  56% usr  43% sys   0% nic   0% idle   0% io   0% irq   0% sirq
Load average: 1.16 1.11 1.07 2/70 364
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
  127   125 root     R    26668  11%  96% /sketch/sketch.elf /dev/pts/0
  206     1 root     S    20060   8%   2% /usr/bin/redis-server /etc/redis/redis.conf
  364   359 root     R     2764   1%   1% top
  357     1 root     S     5516   2%   0% sshd: root@pts/1
  164     1 root     S     4920   2%   0% /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
   14     2 root     SW       0   0%   0% [kworker/0:1]
    3     2 root     SW       0   0%   0% [ksoftirqd/0]
  215   204 root     S    71344  30%   0% /usr/bin/node /opt/xdk-daemon/current/appDaemon.js
  204   191 root     S    35272  15%   0% /usr/bin/node /opt/xdk-daemon/main.js
    1     0 root     S    22100   9%   0% {systemd} /sbin/init
  115     1 systemd- S    12212   5%   0% /lib/systemd/systemd-timesyncd
   84     1 root     S    10004   4%   0% /lib/systemd/systemd-udevd
  168     1 root     S     7272   3%   0% /usr/sbin/connmand -n
  214     1 root     S     6992   3%   0% /usr/sbin/wpa_supplicant -u
   58     1 root     S     6724   3%   0% /lib/systemd/systemd-journald
  131     1 root     S     5840   2%   0% /usr/sbin/tcf-agent -d -L- -l0
  136     1 root     S     5652   2%   0% /usr/sbin/ofonod -n
  190     1 root     S     4708   2%   0% /usr/lib/bluez5/bluetooth/bluetoothd
  359   357 root     S     3476   1%   0% -sh
  132     1 avahi    S     3436   1%   0% avahi-daemon: running [galileo.local]
  191     1 root     S     3348   1%   0% {xdk-daemon} /bin/sh /opt/xdk-daemon/xdk-daemon
  123     1 root     S     3344   1%   0% /bin/sh /opt/cln/galileo/launcher.sh
  154   132 avahi    S     3332   1%   0% avahi-daemon: chroot helper
  149     1 messageb S     3088   1%   0% /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
  179     1 nobody   S     2852   1%   0% /usr/sbin/mdnsd
  146     1 root     S     2768   1%   0% /lib/systemd/systemd-logind
  171     1 root     S     2388   1%   0% /usr/sbin/lighttpd -D -f /etc/lighttpd.conf
  125   123 root     S     2268   1%   0% /opt/cln/galileo/clloader --escape --binary --zmodem --disable-timeouts
  177     1 avahi-au S     2268   1%   0% avahi-autoipd: [enp0s20f6] sleeping
  194     1 root     S     2244   1%   0% /sbin/agetty -8 --keep-baud ttyS1 115200 xterm
  195     1 root     S     2244   1%   0% /sbin/agetty --noclear tty1 linux
  178   177 root     S     2012   1%   0% avahi-autoipd: [enp0s20f6] callout dispatcher
```

Figura 36: Programa Arduino funcionando como um processo Linux

3.2.13. Comunicação entre os programas Arduino e Linux

Atualmente não é possível alterar os programas compilados pela IDE Arduino por meio do Linux, todavia, é possível prover a comunicação entre os dois sistemas, permitindo a integração dos mesmos.

A forma mais eficiente de prover a comunicação entre os dois sistemas é por meio de comunicação padrão entre processos (IPC) para gerenciar a comunicação entre o Arduino e os processos nativos. Esta abordagem é uma solução funcional, pois os

processos Arduino e Linux compartilham a mesma memória o que permite que qualquer alteração em um processo seja imediatamente visível a todos os outros processos.

Outra forma de prover a comunicação é utilizando leitura e escrita em arquivos, pois tanto o Linux quanto os processos Arduinos são capazes de acessar o cartão de memória SD. Neste processo, basta escrever realizar a leitura e escrita em um mesmo arquivo (tanto com o processo Arduino quanto utilizando o Linux) para estabelecer a comunicação.

A comunicação padrão entre processos (IPC) é o método mais eficiente dos citados pois o tempo de acesso a memória RAM é menor que o tempo de acesso a memória Flash.

A Figura 37 apresenta o fluxograma completo do programa implementado na placa Intel Galileo para recebimento das requisições e atuação do envio das mensagens de texto ao celular do responsável. O código implementado é apresentado no Algoritmo A - 15 (Apêndice).

Os Scripts executados são apresentados a seguir:

Script “./gas” executa yowsup para envio de mensagem de texto com a mensagem “Possivel vazamento de gas detectado”.

```
#!/bin/sh
yowsup-cli demos -s 55199927779XX " Possivel vazamento de gas detectado" -c whatsapp.config
```

Script “./acelerometro” executa yowsup para envio de mensagem de texto com a mensagem “Choque mecanico detectado”.

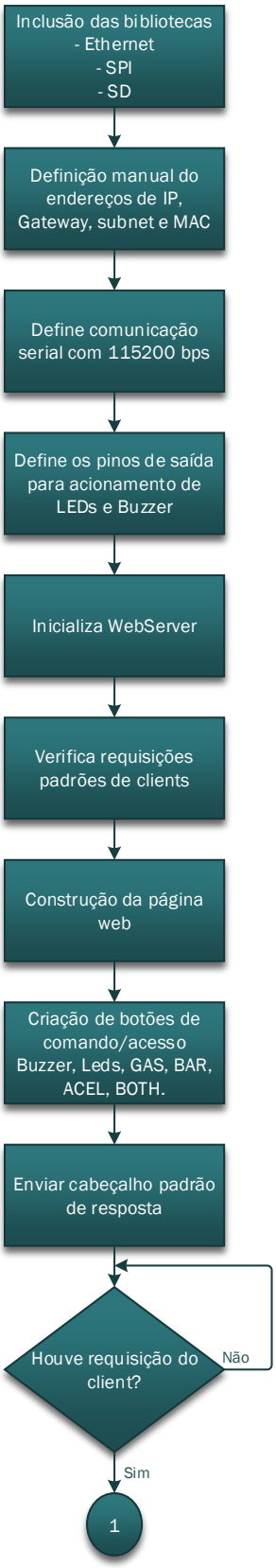
```
#!/bin/sh
yowsup-cli demos -s 55199927779XX " Choque mecanico detectado "  -c whatsapp.config
```

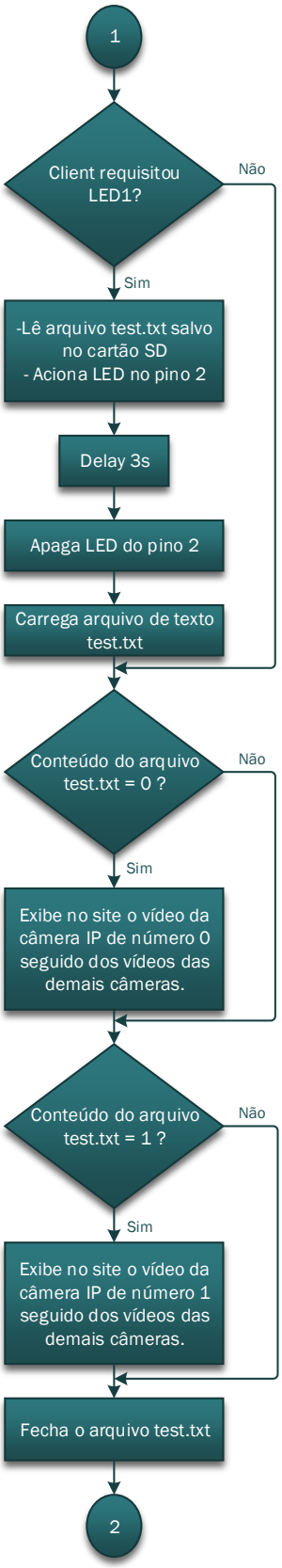
Script “./barometro” executa yowsup para envio de mensagem de texto com a mensagem “Reducao brusca de altitude detectada”.

```
#!/bin/sh
yowsup-cli demos -s 55199927779XX " Reducao brusca de altitude detectada " -c whatsapp.config
```

Script “./acel_and_bar.sh” executa yowsup para envio de mensagem de texto com a mensagem “Perigo possivel queda, verifique o sistema de cameras”.

```
#!/bin/sh
yowsup-cli demos -s 55199927779XX " Perigo possivel queda, verifique o sistema de cameras"  -c whatsapp.config
```


Inclusão das bibliotecas
- Ethernet
- SPI
- SD
Definição manual do endereços de IP, Gateway, subnet e MAC
Define comunicação serial com 115200 bps
Define os pinos de saída para acionamento de LEDs e Buzzer
Inicializa WebServer
Verifica requisições padrões de clients
Construção da página web
Criação de botões de comando/acesso Buzzer, Leds, GAS, BAR, ACEL, BOTH.
Enviar cabeçalho padrão de resposta
Houve requisição do client?
Não
Sim
1


1
Client requisitou LED1?
Não
Sim
-Lê arquivo test.txt salvo no cartão SD
- Aciona LED no pino 2
Delay 3s
Apaga LED do pino 2
Carrega arquivo de texto test.txt
Conteúdo do arquivo test.txt = 0 ?
Não
Sim
Exibe no site o vídeo da câmera IP de número 0 seguido dos vídeos das demais câmeras.
Conteúdo do arquivo test.txt = 1 ?
Não
Sim
Exibe no site o vídeo da câmera IP de número 1 seguido dos vídeos das demais câmeras.
Fecha o arquivo test.txt
2

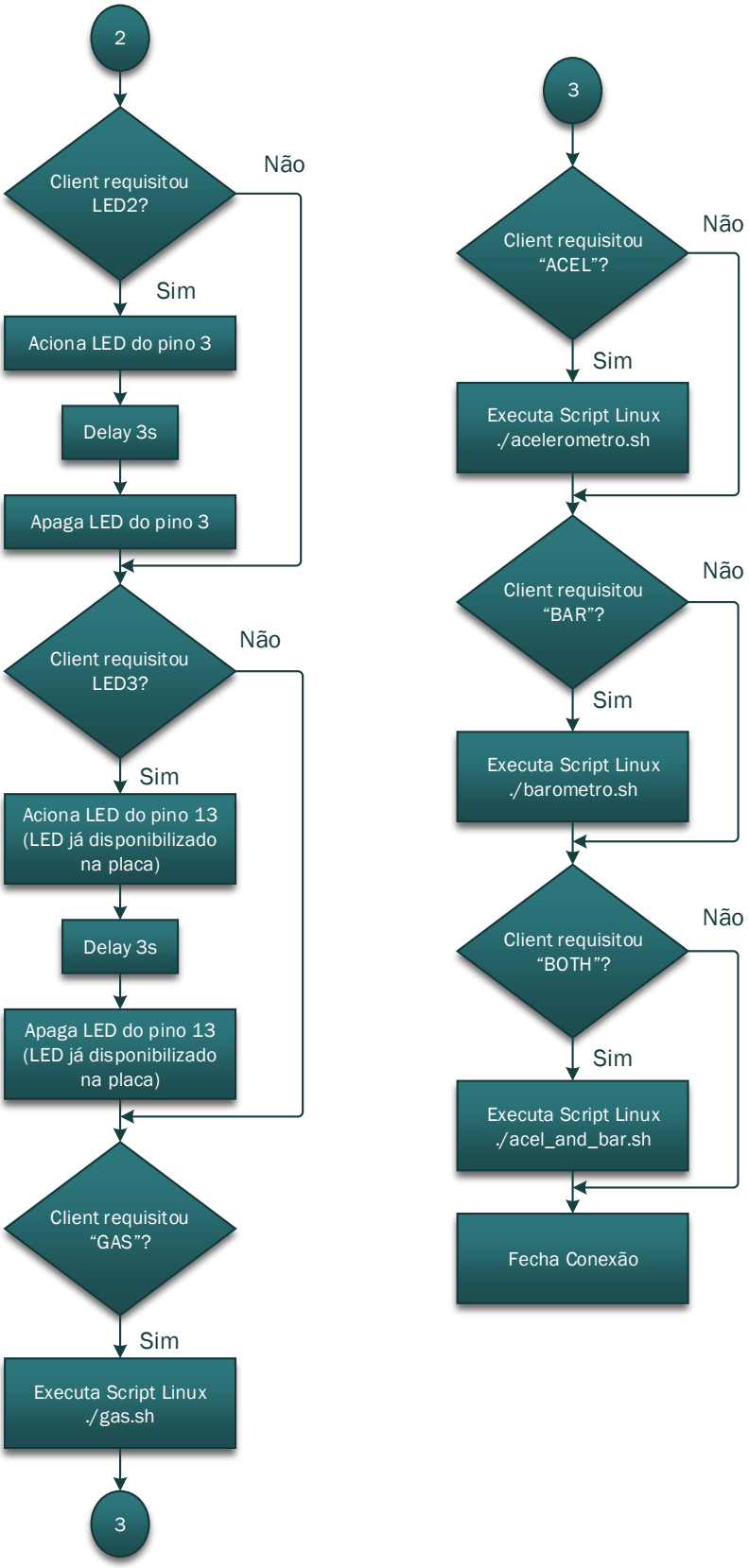


Figura 37: Fluxograma completo da placa de desenvolvimento Intel Galileo, incluindo comunicação entre o processo Arduino e o Linux

4. Resultados e Discussões

Nesta seção serão descritos os resultados do desenvolvimento do projeto até a criação de um protótipo. Cada módulo citado anteriormente foi testado individualmente. Na seção 4.2. (Discussões) são apresentadas as observações obtidas a partir dos resultados, com foco nas aplicações propostas. A visão geral do projeto, com seus componentes (Diagrama de alto nível) é apresentado no diagrama de blocos da Figura 38. Na seção 4.12.13 é apresentado o diagrama funcional de cada módulo.

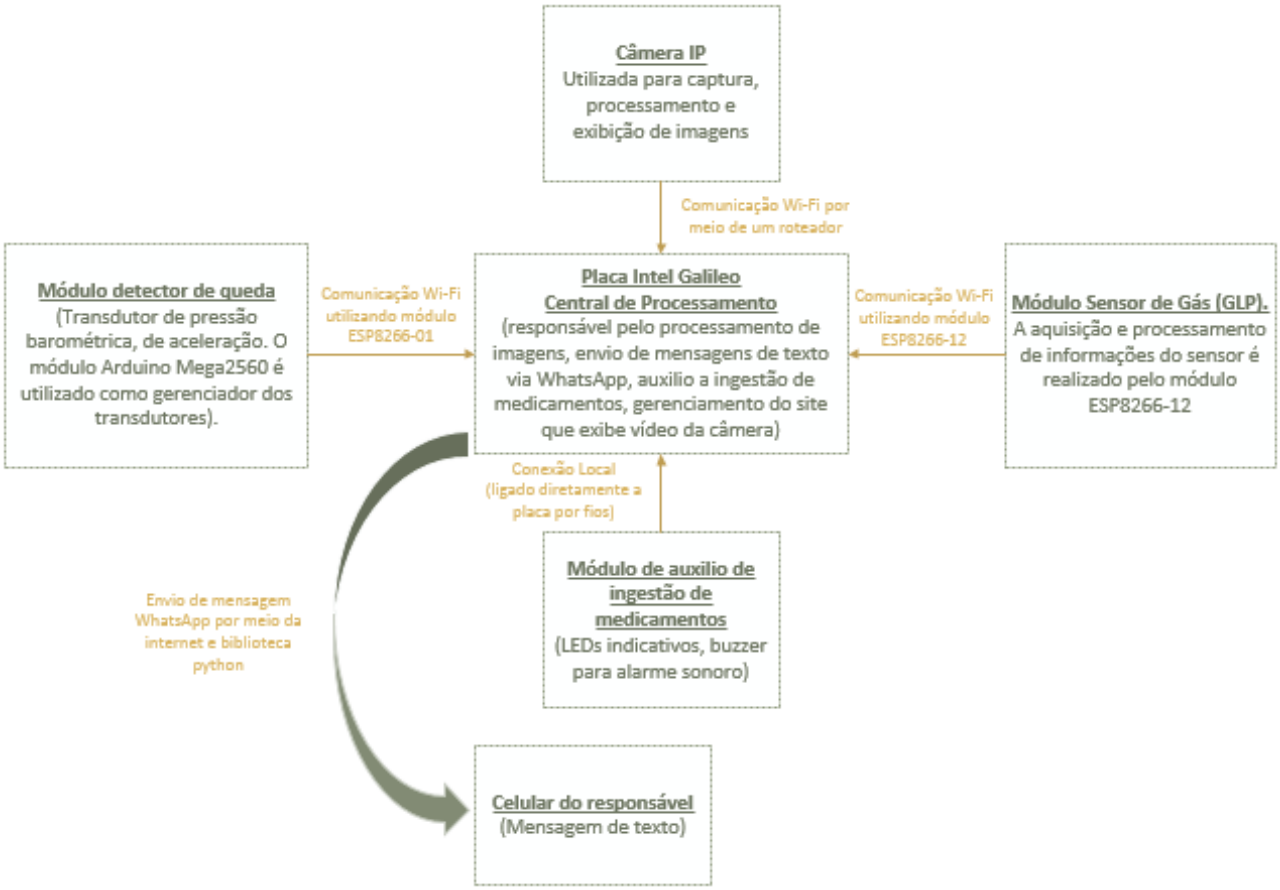


Figura 38: Diagrama de blocos do funcionamento completo do sistema

4.1. Resultados

Nesta seção serão apresentados os resultados baseados nos materiais e métodos utilizados.

4.1.1. Atuação das funcionalidades de Arduino

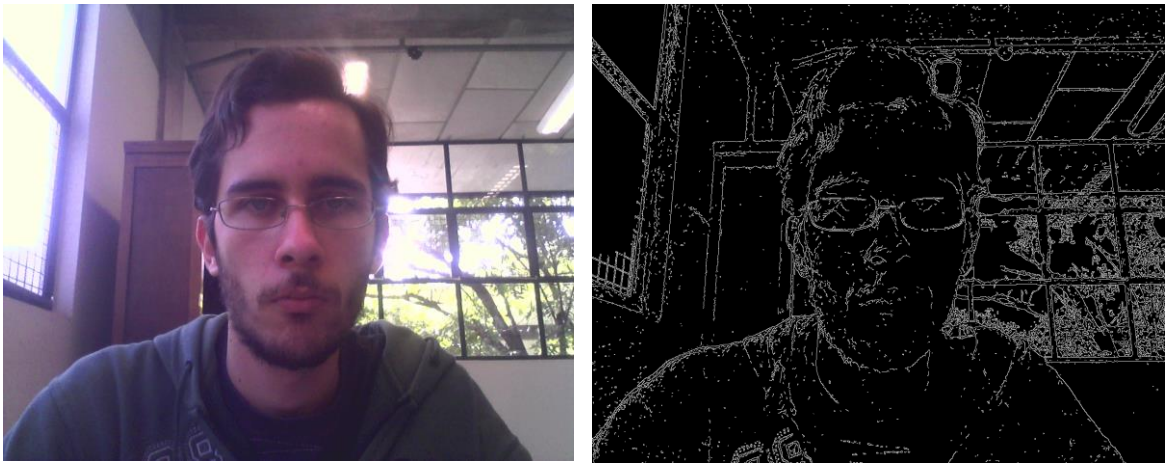
A placa Intel Galileo apresenta funcionalidade de programação como Arduino adicionadas as operações com sistema operacional. Para efeito de teste foi criado um programa utilizando a IDE Arduino funcionando como um Servidor de páginas Web. Nesta página foi possível controlar funcionalidades da placa por meio desta interface, desde acionamento de um buzzer ou LED ao acesso e exibição de imagens de uma câmera IP instaladas no laboratório conforme apresentado na Figura 39. Parte deste teste foi utilizado no projeto, pois o sistema de visão computacional identifica a posição em que a pessoa caiu e esta imagem deverá ser a primeira a ser apresentada na página web, agilizando assim, os primeiros socorros caso seja necessário. A Figura 39 é resultado de uma requisição de vídeo do Arduino para a câmera IP.

Figura 39: Exibição do site gerenciado pela placa Intel Galileo

4.1.2. Uso da biblioteca OpenCV

Conforme apresentado na seção 3.2.4 (Estudo e aplicação da biblioteca OpenCV), os testes iniciais com o uso do programa OpenCV foram realizados em um computador com Linux (Ubuntu) e uma webcam conectada a este.

Como resultado simples da utilização da biblioteca OpenCV pode-se observar a Figura 40 a) e b). Na Figura 40 a) é apresentado a imagem capturada pela câmera utilizando comandos da biblioteca OpenCV. Na Figura 40 b) é apresentado o resultado da aplicação de um filtro de borda que detecta alta frequência espacial (variação abrupta dos pixels).



a) Imagem original

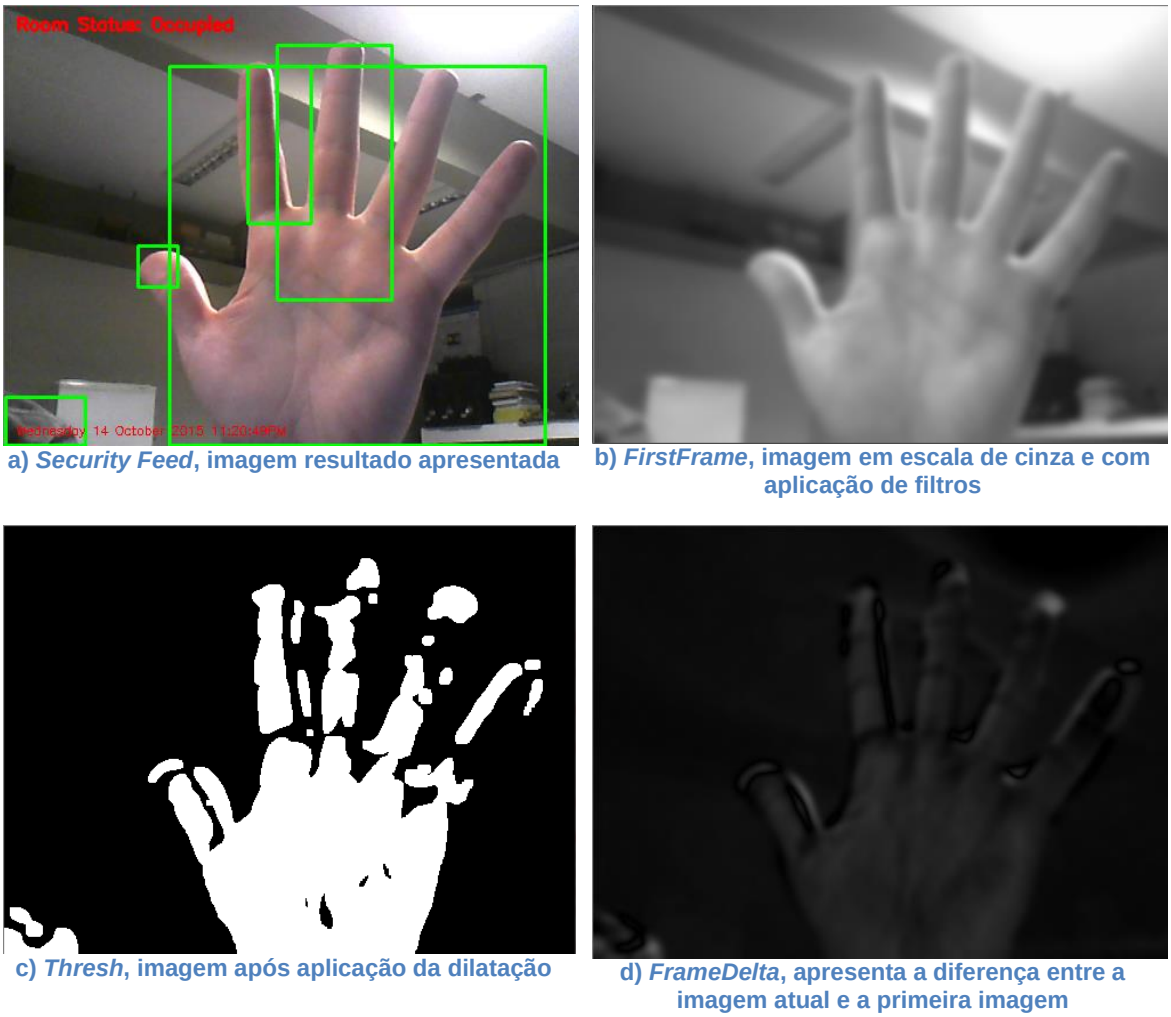
b) Resultado da aplicação de um filtro detector de borda

Figura 40: Imagem antes e após a aplicação do processamento de imagem

A versão de Linux que apresentou melhor funcionalidade com o uso de OpenCV e Python foi a versão IoT dev Kit da Intel, motivo pelo qual esta foi escolhida. As demais apresentaram erros ao executar o programa ou não possuíam Python já compilado.

4.1.3. Sistema de detecção de movimentos

Conforme indicado na seção 3.2.5 (Sistema de detecção de movimentos) os algoritmos propostos para identificação de movimento foram implementados. A Figura 41 apresenta a resposta gráfica para o algoritmo implementado em um computador (Algoritmo A - 4), sendo obtido a imagem por meio de uma webcam. Este resultado foi utilizado como base para verificar o funcionamento do algoritmo. Posteriormente este código foi portado para a placa Intel Galileo.



a) *Security Feed*, imagem resultado apresentada

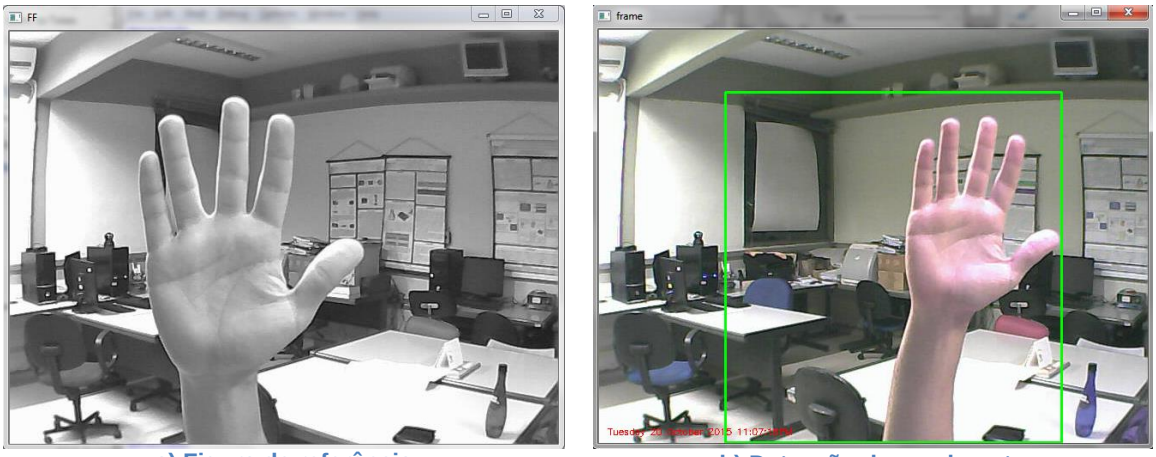
b) *FirstFrame*, imagem em escala de cinza e com aplicação de filtros

c) *Thresh*, imagem após aplicação da dilatação

d) *FrameDelta*, apresenta a diferença entre a imagem atual e a primeira imagem

Figura 41: Resultado do processamento de imagem com o algoritmo apresentado em Algoritmo A – 4 (Apêndice)

A Figura 42 apresenta o resultado do Algoritmo A - 5 (Apêndice) de detecção de movimento por meio da câmera IP disponível no laboratório. Na “Figura 42 a)” é exibida a imagem de referência para detecção de movimento (subtração de imagens) e na “Figura 42 b)” é exibida a imagem atual com a inserção do retângulo indicando a região de movimento.



a) Figura de referência b) Detecção do movimento

Figura 42: Resultado do Algoritmo 2

4.1.4. Desempenho da placa Intel Galileo

Para calcular a velocidade média do processamento das imagens na placa de desenvolvimento Intel Galileo, foi utilizado inicialmente o código apresentado no Apêndice (Algoritmo A - 6) e descrito na seção 3.2.5 (Sistema de detecção de movimento). Como resultado a variação média do tempo retornado foi de 105,4 segundos o que resulta numa taxa de processamento de aproximadamente 1 fps na placa Intel Galileo.

O programa utilizado para análise de desempenho final está apresentado em Algoritmo A - 8 (Apêndice) e descrito na seção 3.2.5 (Sistema de detecção de movimento), este apresentou a variação entre os tempos exibidos de 141,3 segundos, o que equivale a 1,4 fps. O aumento do tempo anteriormente analisado ocorreu devido ao aumento no número de câmeras, logo, a velocidade por câmera foi de 0,7 fps. A redução no tempo por câmera foi possível devido a otimização do código Python, utilizando módulos *Threading* que gerencia processamento “simultâneo” (em um nível alto de abstração).

A placa Intel Galileo não apresentou poder de processamento suficiente para aplicações em visão computacional complexas como detecção de aceleração de objetos na imagem. Observa-se que o sistema em pleno funcionamento para detecção de movimento utilizando o Algoritmo A - 8 (Apêndice) apresentou uso de 96% da CPU (Figura 43). Parte do programa Arduino é apresentado como um processo Linux e é responsável pelo consumo de 32% da CPU enquanto que a aplicação de processamento de imagens é responsável por 64% do uso da CPU, totalizando todo o consumo significativo de processos. O *Sketch* utilizado na implementação completa do sistema utilizou 72.971 bytes.

Em termos de consumo de corrente elétrica, fora medido o valor de 550 mA, apenas para execução do sistema, sem qualquer periférico conectado à placa.

```
Mem: 141884K used, 94988K free, 0K shrd, 7504K buff, 82864K cached
CPU:  96% usr   3% sys   0% nic   0% idle   0% io   0% irq   0% sirq
Load average: 2.85 2.04 1.45 4/75 326
  PID  PPID USER     STAT   VSZ %VSZ %CPU COMMAND
  311   309 root     S    78936  33%  64% python V4 para galileo.py
  124   122 root     R    26668  11%  32% /sketch/sketch.elf /dev/pts/0
  205     1 root     S    20060   8%   2% /usr/bin/redis-server /etc/redis/redis.conf
  324   322 root     R     2764   1%   1% top
  320     1 root     S     5516   2%   1% sshd: root@pts/2
   24     2 root     SW       0   0%   0% [mmcqd/0]
  212   197 root     S    71344  30%   0% /usr/bin/node /opt/xdk-daemon/current/appDaemon.js
  197   185 root     S    35272  15%   0% /usr/bin/node /opt/xdk-daemon/main.js
    1     0 root     S    22048   9%   0% {systemd} /sbin/init
  112     1 systemd- S    12212   5%   0% /lib/systemd/systemd-timesyncd
   81     1 root     S    10004   4%   0% /lib/systemd/systemd-udevd
  163     1 root     S     7264   3%   0% /usr/sbin/connmand -n
  140     1 root     S     6992   3%   0% /usr/sbin/wpa_supplicant -u
   55     1 root     S     6724   3%   0% /lib/systemd/systemd-journald
  133     1 root     S     5840   2%   0% /usr/sbin/tcf-agent -d -L- -l0
  128     1 root     S     5652   2%   0% /usr/sbin/ofonod -n
  307     1 root     S     5516   2%   0% sshd: root@pts/1
  160     1 root     S     4920   2%   0% /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
  184     1 root     S     4708   2%   0% /usr/lib/bluez5/bluetooth/bluetoothd
  309   307 root     S     3460   1%   0% -sh
  322   320 root     S     3460   1%   0% -sh
  126     1 avahi    S     3448   1%   0% avahi-daemon: running [galileo.local]
  185     1 root     S     3348   1%   0% {xdk-daemon} /bin/sh /opt/xdk-daemon/xdk-daemon
  120     1 root     S     3344   1%   0% /bin/sh /opt/cln/galileo/launcher.sh
  143   126 avahi    S     3332   1%   0% avahi-daemon: chroot helper
  146     1 messageb S     3092   1%   0% /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
  144     1 root     S     2768   1%   0% /lib/systemd/systemd-logind
  177     1 nobody   S     2708   1%   0% /usr/sbin/mdnsd
  166     1 root     S     2388   1%   0% /usr/sbin/lighttpd -D -f /etc/lighttpd.conf
  122   120 root     S     2268   1%   0% /opt/cln/galileo/clloader --escape --binary --zmodem --disable-timeouts
  174     1 avahi-au S     2264   1%   0% avahi-autoipd: [enp0s20f6] sleeping
  190     1 root     S     2244   1%   0% /sbin/agetty -8 --keep-baud ttyS1 115200 xterm
  191     1 root     S     2244   1%   0% /sbin/agetty --noclear tty1 linux
  176   174 root     S     2012   1%   0% avahi-autoipd: [enp0s20f6] callout dispatcher
  123     1 root     S     1956   1%   0% /opt/cln/galileo/galileo_sketch_reset
```

Figura 43: Comando Linux (TOP) para exibir características do desempenho

A página Web criada para realizar testes é apresentada na Figura 44. Quando acionado o botão LED1, um LED se acende na placa e também é apresentado a imagem da última câmera que detectou movimento (registrado em arquivo pelo programa em Python detector de movimento). Os botões GAS, BAR, ACEL, BOTH são respectivamente utilizados para simular o envio de mensagem por WhatsApp de vazamento de gás, redução de altura detectado pelo barômetro, choque mecânico detectado pelo acelerômetro e ambos (redução de altura e choque mecânico detectados). Cada módulo monitor envia uma requisição para o site apresentado que tem por efeito o mesmo que o acionamento manual desses botões. Com isso pode-se verificar isoladamente o funcionamento do envio de mensagem e posteriormente verificar o mesmo automaticamente quando os sensores foram sensibilizados.

Figura 44: Página Web criada para exibir câmera que capturou último movimento e também para interface de comunicação entre os módulos monitores e o envio de mensagens por WhatsApp

4.1.5. Sistema detector de quedas

A Figura 45 apresenta o resultado do desenvolvimento do sistema detector de quedas abordado na seção métodos.

Em 1 na Figura 45 é apresentado a placa de desenvolvimento Arduino Mega2560. Em 2 é apresentado o módulo condicionar de sinais para adequar os níveis lógicos de comunicação entre os módulos. Em 3 é apresentado o módulo barômetro, responsável pelas leituras das variações de alturas. Em 4 é apresentado o módulo acelerômetro,

responsável por detectar os choques mecânicos característicos de uma queda. Em 5 é apresentado um LED utilizado para verificar os momentos em que as mensagens de WhatsApp eram enviadas. Em 6 é apresentado o módulo ESP8266-01 responsável pela comunicação Wi-Fi e consequentemente pelo envio dos comandos à placa Intel Galileo.

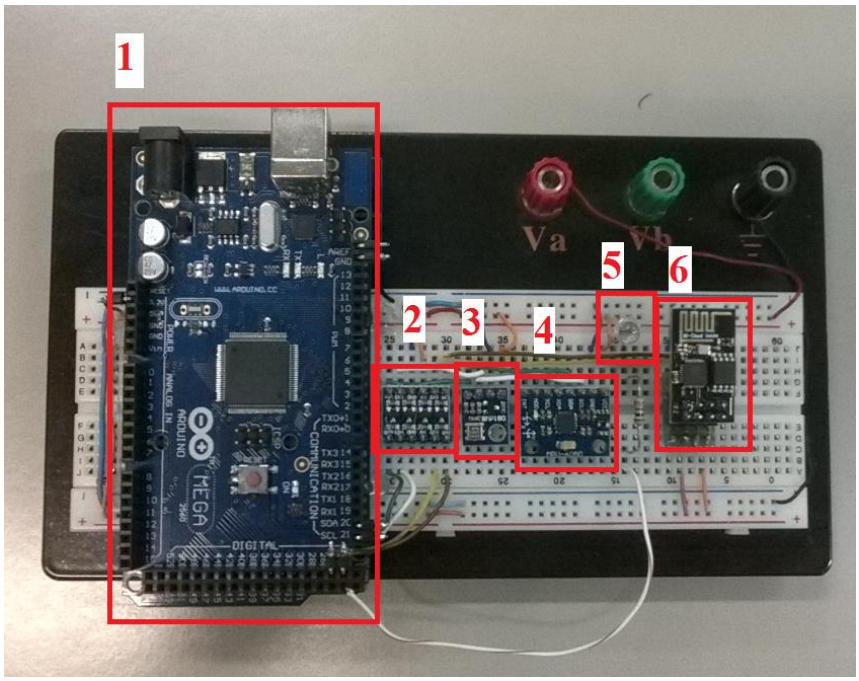


Figura 45: Sistema detector de quedas

- **Módulo ESP8266**

Para o último programa implementado no módulo ESP8266-01, o sketch usa 294.336 bytes (67%) do espaço de armazenamento para programas. O valor máximo é de 434.160 bytes.

Variáveis globais utilizaram 48.124 bytes (58%) da memória dinâmica, deixando 33.796 bytes para variáveis locais. O valor máximo é de 81.920 bytes.

- **Barômetro**

Para a aplicação de detecção de queda de uma pessoa, fora constatado que aumentar o número de aquisições para o cálculo da altura média não apresentou o melhor resultado, pois, a variação se mantem mesmo para 1000 aquisições e a finalidade do projeto requer a detecção abrupta dos valores. O melhor resultado empírico, ocorreu para uma média de 100 amostras e um intervalo de 1,7 após cada resultado da média. Com isso, a detecção gera um atraso de cerca de 4 segundos. Apesar de ser um tempo considerável, ainda assim, é aceitável para prestar socorro em caso de acidente. Foi utilizado um

threshold de 0,5 metros para indicar uma possível queda. O gráfico apresentado na Figura 46 evidencia a variância obtida para 400 pontos, com o módulo estático. Vale ressaltar que o ambiente em que o teste foi realizado apresentou variação de temperatura, pois estava sofrendo ação de ar condicionado, com ventilação e iluminação também variáveis. Apesar do ambiente não ser controlado, este se assemelha com o ambiente em que se propõe a utilização do mesmo.

Outro fator a ser considerado do uso do barômetro para essa aplicação é a existência de degraus ou declives nos ambientes onde o idoso se movimenta. Essa limitação possui sua ação atenuada pois o sistema detecta a variação da altitude em relação à última medida e não em relação à altura absoluta. A detecção da variação relativa atenua problemas em relação a rampas ou declives suaves. Quanto ao uso em ambientes com degraus ou escada, o sistema pode ser utilizado conjuntamente com acelerômetro, permitindo detectar choques mecânicos. A mensagem enviada ao responsável indicará qual sistema foi acionado e caso ambos tenham sido acionados, há uma maior probabilidade de ter ocorrido um acidente e não ser um falso alarme.

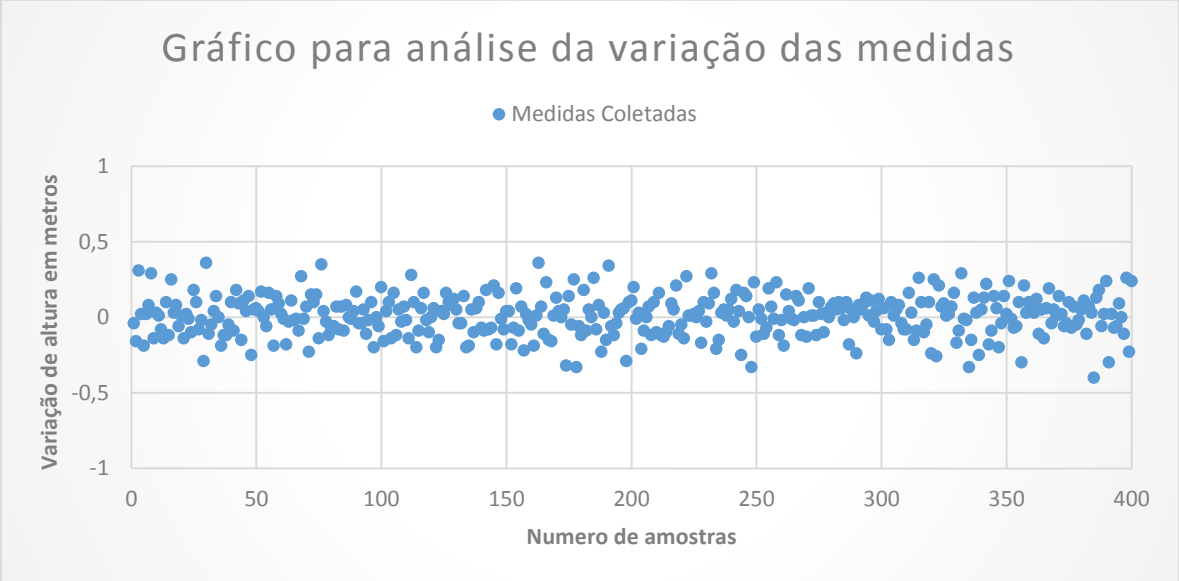


Figura 46: Gráfico da variação da pressão atmosférica e consequentemente altitude para o sensor posicionado de forma estática

Observações gerais:

Precisão: O nível de ruído teórico da resolução mais alta é de 0,25 m, embora, na prática, vemos ruído na ordem de 0,5 m. Esta precisão pode ser melhorada tomando um grande número de leituras e analisando suas médias, entretanto, isso vai abrandar a sua taxa de amostra e tempo de resposta. Para o projeto proposto um número muito grande de

amostras irá prejudicar a detecção de uma queda, que se caracteriza pela variação abruta de altura.

Ambiente aberto: O BMP180 precisa de acesso ao ar ambiente para medir a sua pressão, desta forma, o local onde ele ficará não deve ser vedado do ambiente externo. Para o protótipo proposto o BMP180 ficou em ambiente aberto.

Não se deve incidir ventilação diretamente sobre o BMP180: A exposição ao ar ou vento em movimento rápido pode causar variações de pressão momentânea que vão afetar suas leituras. Deve-se, portanto, proteger o dispositivo a partir de correntes de ar fortes.

Mudanças bruscas de temperatura: O componente apesar de possuir compensação térmica pode apresentar erro de medida se houver mudanças bruscas de temperatura.

- **Consumo elétrico do módulo**

A medida de corrente elétrica foi de 218,3 mA, efetuada conforme apresentado em materiais e métodos.

4.1.6. Sistema detector de vazamento de gás

O módulo do sistema detector de vazamento de gás é apresentado na Figura 47. Em 1 é apresentado o sensor de gás GLP, este deve ser alimentado com 5V. Para a aplicação proposta, foi utilizado a saída analógica do sensor, permitindo assim, maior controle por meio da programação do valor lido no módulo 3. Em 3 é apresentado o modulo ESP8266-12 que apresenta conversor analógico-digital integrado. O módulo 3 deve ser alimentado com 3,3V. Para a adequação das diferentes tensões foi utilizado um divisor resistivo na interface entre os módulos. O consumo elétrico medido do módulo foi de 200mA.

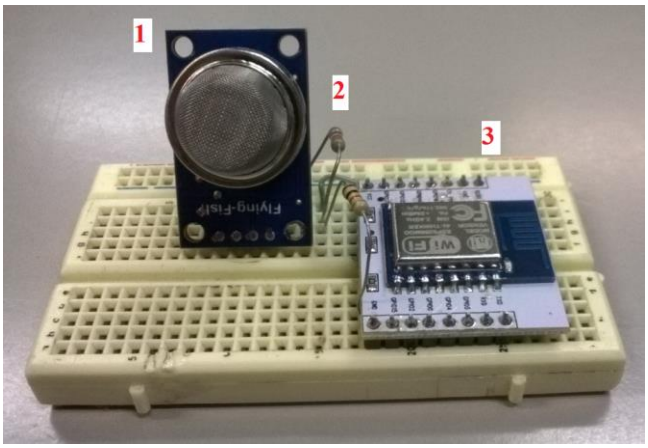


Figura 47: Módulo detector de gás

O sketch usou 294.168 bytes (67%) do espaço de armazenamento para programas. O valor máximo é de 434.160 bytes. Variáveis globais usaram 48.092 bytes (58%) de memória dinâmica, deixando 33.828 bytes para variáveis locais. O máximo são 81.920 bytes.

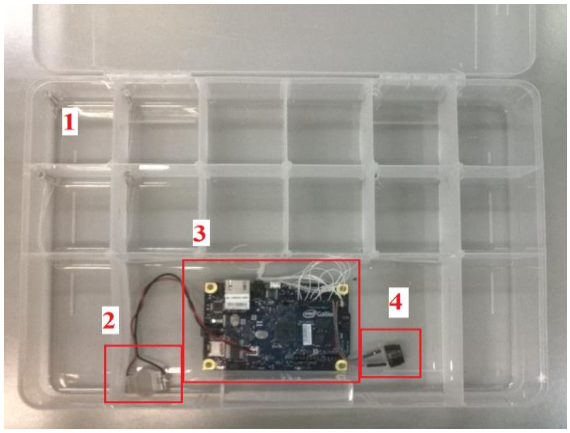
4.1.7. Sistema de auxílio a ingestão de medicamentos

O sistema de auxílio a ingestão de medicamentos apresentou sua viabilidade técnica, conforme descrito na seção métodos 3.2.11 (Sistema de alarme e indicação de medicamento). A Figura 48 apresenta a implementação do protótipo do módulo de auxílio a ingestão de medicamentos. Observa-se que a indicação dos compartimentos pode ser melhorada utilizando meios que apresentam espalhamento da luz como acrílico ou com a utilização de mais LEDs indicativos para um mesmo compartimento. Para efeito de protótipo, bastou observar a correta indicação do compartimento.

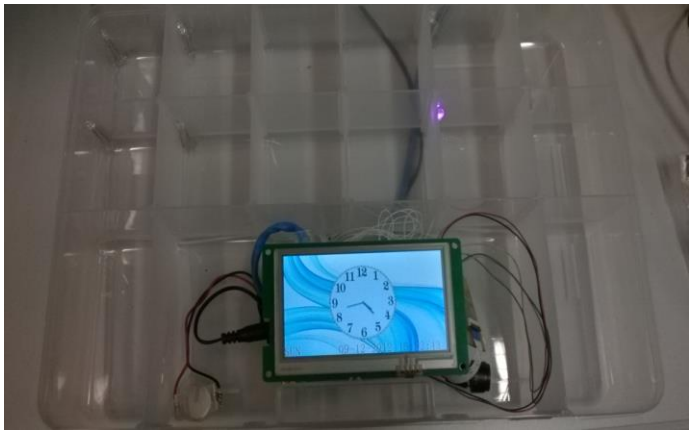
Na Figura 48 a) é possível observar em 1 a caixa, exemplo de proposta mecânica para o projeto. Em 2 é indicado a bateria CR2032 utilizada para manter o correto funcionamento da data e horário da placa (RTC) cruciais para a aplicação em questão. Em 3 é apresentado a placa Intel Galileo já conectada aos LEDs da caixa. Em 4 é apresentado um *buzzer* para acionar o alarme sonoro para indicar que o medicamento deve ser ingerido (este para maior eficácia deve ser instalado na parte externa da caixa).

Na Figura 48 b) é apresentado a caixa aberta exibindo imagens no display touchscreen. Esta imagem, para o projeto proposto, exibe a foto da referida pessoa que deve ingerir o medicamento quando for o horário da ingestão do medicamento, auxiliando assim, confusões caso haja mais de uma pessoa utilizando o sistema.

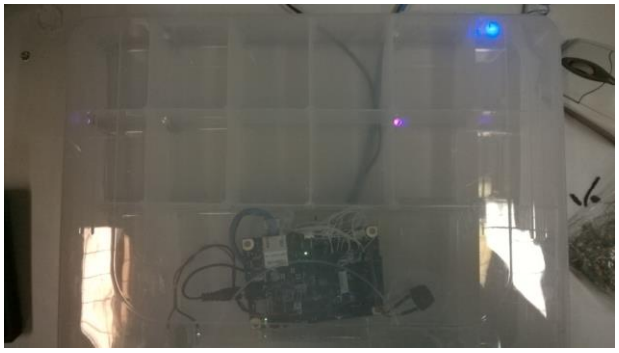
Na Figura 48 c) é apresentado o acionamento dos LEDs, sendo possível visualizá-los mesmo com a caixa fechada.



a) Intel Galileo posicionada no módulo de auxílio a ingestão de medicamentos.



b) Display LCD posicionado acima da placa Intel Galileo



c) Visualização do compartimento indicado pelo LED mesmo com a caixa fechada

Figura 48: Módulo de auxílio a ingestão de medicamentos

- **Consumo elétrico**

O Consumo elétrico da placa Intel Galileo, aferido conforme indicado em materiais e métodos, foi de aproximadamente 530mA. Ao instalar o display *touchscreen* o consumo total aumentou para 730mA. Deve-se considerar ainda que a corrente mínima dimensionada para os LEDs é de 5mA, neste caso, se os 12 compartimentos forem ligados simultaneamente o consumo eletrico se aproxima do valor máximo fornecido pelo fabricante.

4.1.8. Integração

Realizados os testes individuais conforme discutido e apresentado nas seções anteriores, houve a necessidade da integração dos módulos. Seu funcionamento completo é apresentado no diagrama de blocos da Figura 49, Figura 50, Figura 51 e Figura 52.

Foi observado incompatibilidade entre os níveis de tensão de comunicação do módulo Arduino Mega2560 e os módulos de barômetro, acelerômetro e ESP8266. Para solucionar essa limitação foi utilizado um conversor de sinais que pode ser construído com divisor resistivo e transistores. Neste protótipo, foi utilizado um módulo conversor de nível para esta tarefa.

Ao integrar os módulos testados individualmente, houve uma redução no tempo de envio das mensagens de texto por WhatsApp de 30 segundos até o recebimento da mensagem (nos piores casos). Isto ocorre devido ao acumulo de processamentos de dados. Também se observou que o vídeo exibido no site utilizando o link com a câmera IP apresenta pequenos atrasos de exibição (atraso menor que 1 segundo) nos momentos em que a figura de referência é atualizada. Apesar dessas alterações do sistema na integração, o projeto apresentou todas as funcionalidades exigidas como detectar vazamento de gás, identificar possíveis quedas (variação de aceleração e altitude), realizar processamento de imagens de câmeras IPs e implementar base para um sistema de gerenciamento de medicamentos.

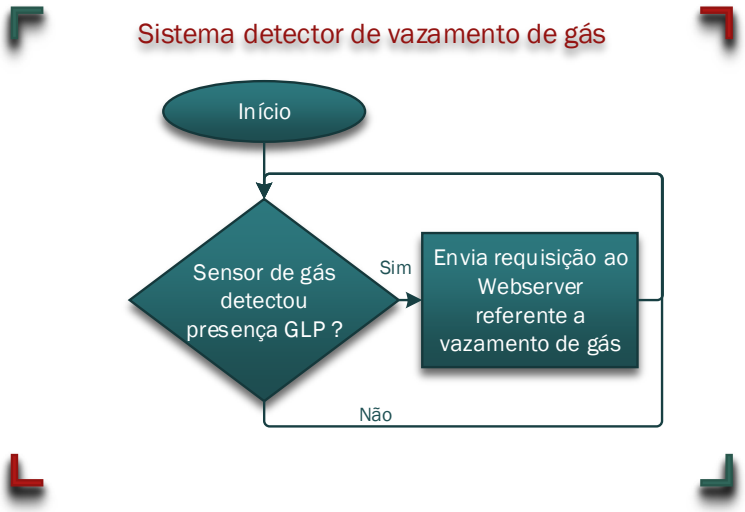


Figura 49: Diagrama de blocos do sistema detector de vazamento de gás

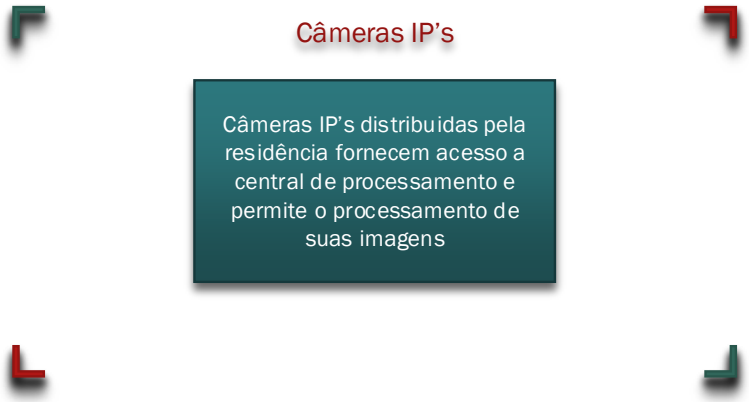


Figura 50: Diagrama representativo da câmera IP, componente deste trabalho

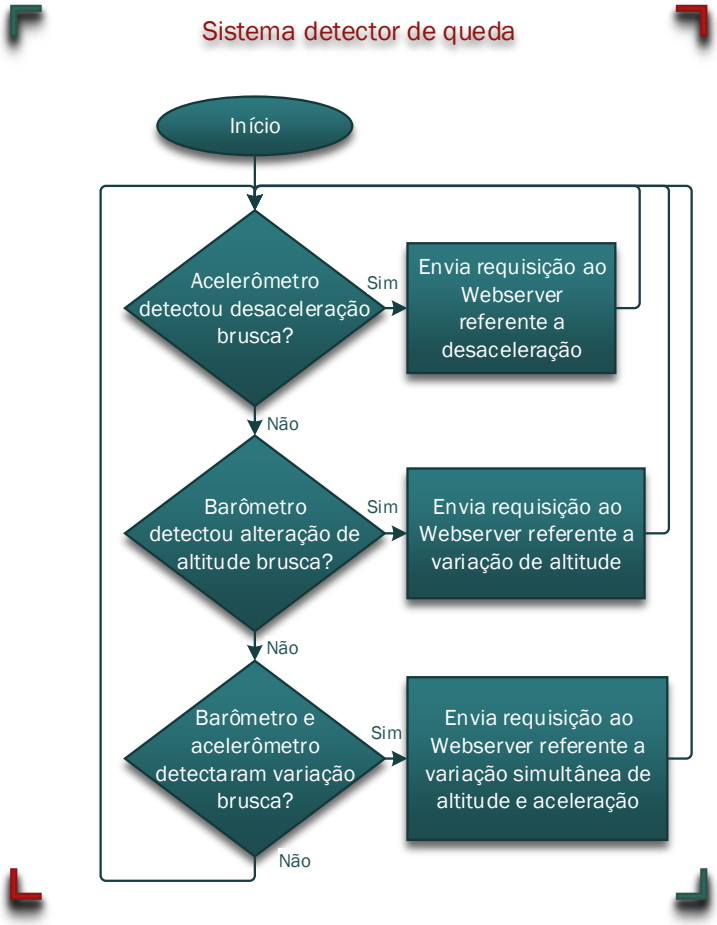


Figura 51: Diagrama de blocos do módulo detector de quedas.

Central de Processamento Intel Galileo

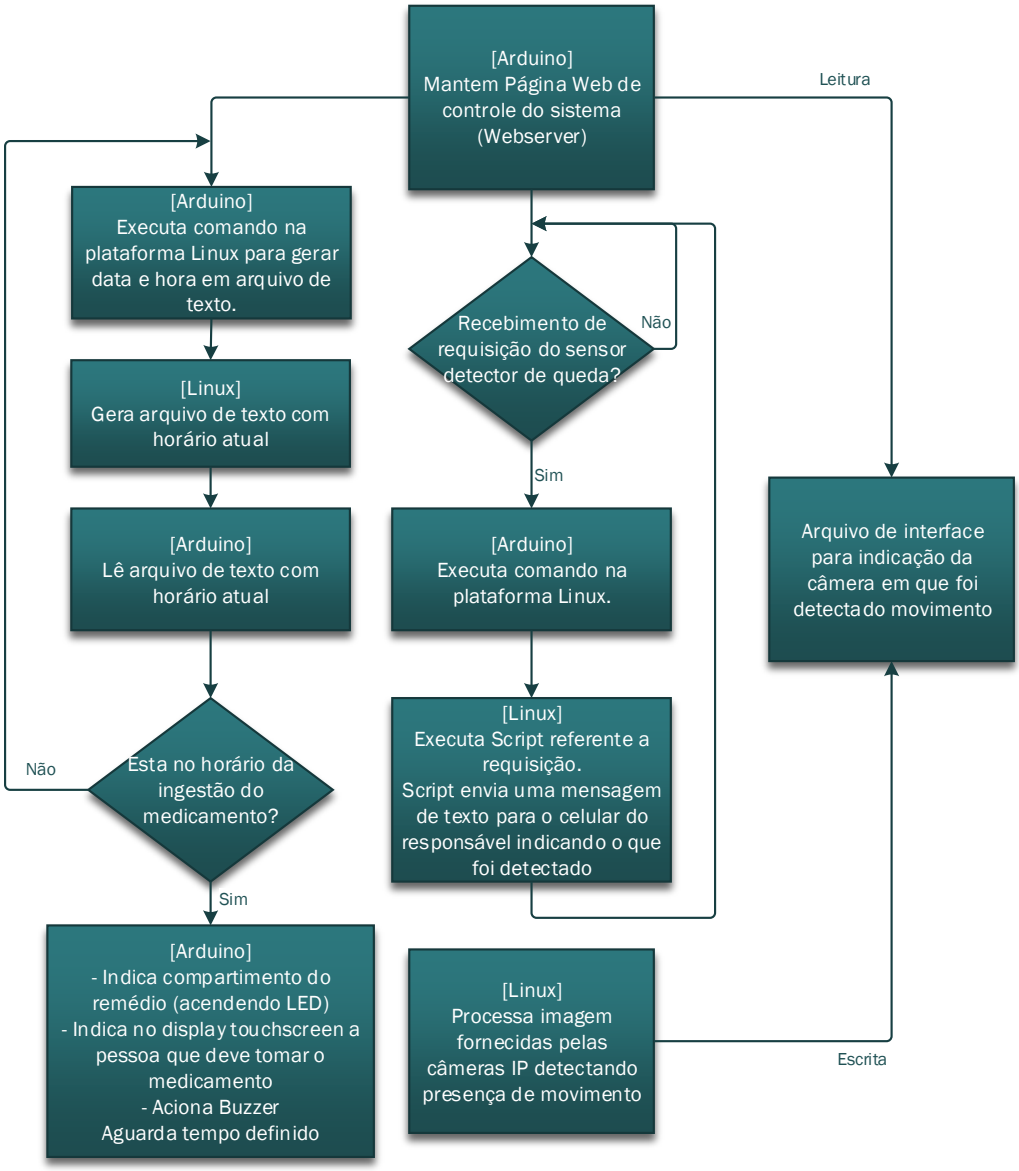


Figura 52: Diagrama de blocos do módulo de desenvolvimento Intel Galileo

4.2. Discussões

Nesta seção serão apresentadas observações e características constatadas com os testes realizados.

4.2.1. Placa Intel Galileo (Hardware)

A Placa Intel Galileo apresenta um processador de 32bits que permite o uso de software embarcado. Também possui sistema de conexão e pinos semelhantes ao padrão Arduino UNO oferecendo compatibilidade com *shields* Arduino e permitindo a integração com o Linux embarcado. Essas caraterísticas tornam o desenvolvimento e a integração entre software e Hardware fáceis de implementar, conforme abordado neste trabalho. A placa apresenta uma estrutura para desenvolvimento bastante abrangente na medida em que oferece funcionalidades de um sistema embarcado e as facilidades dos módulos de desenvolvimento Arduino.

Apesar das grandes facilidades apresentadas, também foram constatadas as limitações da placa, como o processamento de imagens de aproximadamente 1 fps, inviabilizando várias aplicações em visão computacional. Conforme apresentado na seção 4.2.4 sua velocidade final para processamento de imagem é de cerca de 15 vezes mais lento que um computador convencional [53].

4.2.2. Placa Intel Galileo (software - Arduino)

A IDE Arduino permite compatibilidade com as bibliotecas do Arduino UNO, possibilitando migração dos usuários para uma placa de desenvolvimento com mais recursos como comunicação Ethernet, USB host, Cartão de Memória e relativamente alto poder de processamento.

Inicialmente a IDE Arduino 1.5.3 apresentou compatibilidade apenas para o sistema operacional Windows 7 em inglês. Esse problema foi solucionado meses depois com o lançamento da versão Arduino IDE 1.6.

Conforme apresentado em Desenvolvimento os programas Arduinos compilados (chamados Sketch) são salvos no diretório “/sketch” e a partir desse local são executados como um processo do Linux.

4.2.3. Placa Intel Galileo (Software - Linux).

O grande diferencial dessa placa é possui Linux embarcado. Sua compatibilidade e suporte oferecido em comunidade Yocto presente na internet permitem a construção de uma distribuição personalizada para determinada aplicação. O uso do *bitBaker* agiliza e facilita o processo de escolha e compilação do Linux personalizado.

A distribuição utilizada neste projeto (Intel IoT DevKit) apresentou grandes diferenças de configurações e usabilidade do padrão de distribuição Debian de Linux. Algumas configurações mesmo após executadas não surtiram efeito sendo necessário buscar formas alternativas para solucionar tais problemas. Neste quesito, a Intel apresentou suporte ao usuário que respondeu prontamente as dúvidas, porém, mesmo o suporte indicou procedimento que não surtiram efeito, sendo necessário realizar pesquisas em conjunto com o suporte para solucionar esses problemas.

Neste trabalho foi apresentado 3 formas de comunicação entre os programas Arduino e o Linux, sendo eles por meio do cartão SD (memória flash compartilhada), comunicação IPC (Comunicação Inter-Processos) e por envio de comandos do programa Arduino para o Linux utilizando o comando “system()”. Apesar de conseguir estabelecer a comunicação corretamente com os meios propostos, uma limitação da integração se refere a impossibilidade de compilar programas Arduino no próprio Linux Embarcado, essa funcionalidade permitiria programar o Arduino remotamente.

4.2.4. Linguagem de programação Python

A linguagem de programação Python apresentou grande facilidade de aprendizado com vários fóruns disponíveis na internet e disponibilidade de várias bibliotecas que agilizam o desenvolvimento do projeto. O uso desta linguagem permitiu a rápida implementação do sistema proposto. Apesar do desempenho ser inferior a linguagem puramente compilada observou-se que o atraso decorre do poder de processamento da placa, o uso de uma linguagem compilada não contribuiria significativamente para a velocidade de processamento de imagens desejado. Isso se evidenciou ao comparar o tempo de processamento de um computador convencional com o tempo da placa Intel Galileo para o processamento do mesmo código.

O uso da biblioteca OpenCV permitiu a rápida implementação das funções de processamento de imagem, pois apresenta funções dedicadas e otimizadas para esta finalidade

4.2.5. Módulo (Shield) GSM

O funcionamento do módulo apresentado ocorreu corretamente na placa Arduino Mega2560, entretanto, o mesmo programa e biblioteca não foi compatível com a placa Intel Galileo. Seu funcionamento ocorreu na placa Arduino Mega2560. Nessas condições o dispositivo detector de queda envia uma mensagem diretamente ao celular do responsável evitando um ponto de falha (enviar comando para central de processamento e esta transmitir a mensagem). Todavia, recomenda-se o fornecimento de alimentação de até 2A o que dificulta a aplicação em sistemas móveis, pois o consumo de bateria torne-se significativo. O módulo GSM também apresenta como desvantagem o custo da operadora para envio da mensagem. Por esses motivos e pela possibilidade de implementar a solução de envio de mensagem de texto utilizando o aplicativo WhatsApp sem qualquer custo, a solução com módulo GSM não foi implementada no projeto final.

4.2.6. Câmeras IPs

As câmeras IPs apresentaram resolução satisfatória para o projeto, permitindo a visualização de ambientes com clareza. O uso de LEDs infravermelho permitiu que o sistema funcione mesmo em ambientes sem iluminação.

A apresentação da imagem no site ocorre diretamente da câmera como link de vídeo, sem aplicar conversão ou filtro de visão computacional. Esse fato permitiu que o vídeo apresentasse velocidade de exibição de 20 fps.

4.2.7. Transdutores – Acelerômetros

Ambos os acelerômetros testados apresentaram resultados satisfatórios de funcionalidade, isto é, precisão e imunidade a ruído requeridos para o projeto. O MPU-6050 foi escolhido para integrar o projeto final por apresentar os quesitos anteriormente citados com melhores resultados teóricos. O uso de comunicação I²C evita a influência de possíveis erros de leitura e permite a detecção de erros de conexão de forma mais efetiva.

4.2.8. Transdutor – Barômetro

O manual do fabricante do barômetro indica erro teórico de medida de 17 a 25 cm de altitude. Entretanto, a análise empírica dos resultados varia cerca de 0,5 m. Como solução o código apresentado utiliza média de 100 aquisições. Essas 100 aquisições são realizadas a cada 1,7 segundos, este tempo foi obtido empiricamente e é necessário para que o cálculo da média não influenciasse a detecção da variação abrupta de altitude. Esse

sistema permitiu detectar variações de altitudes de aproximadamente 35 cm. Neste caso o sistema de detecção apresenta atrasos, porém, para a aplicação desejada, atrasos de 2 segundos não são significativos.

4.2.9. Transdutor – Sensor de Gás

A detecção de vazamento de gás apesar de ser influenciada de forma significativa pela temperatura apresentou-se bastante efetiva para determinar presença de gás em uma simulação realizada onde uma válvula de gás foi aberta, simulando um possível vazamento, e em aproximadamente 7 segundos o sensor detectou vazamento, estando posicionado a 1 metro de distância da saída da válvula.

4.2.10. Módulo ESP8266

O módulo ESP8266 apresentou funcionalidades de Arduino, como sua programação utilizando a IDE Arduino, a utilização de periféricos e comunicação Wi-Fi. Sua principal desvantagem está no que se refere a consumo de energia conforme apresentado na Tabela 2. Se consumo não for limitante no projeto, este modulo apresentará grande destaque na escolha de módulos Wi-Fi.

4.2.11. Módulo Arduino Mega2560

O Arduino Mega2560 ganha destaque por apresentar vários periféricos disponíveis como portas de comunicação serial, pinos ADC, Pinos de PWM e outros. Por esse motivo foi utilizado para a elaboração do protótipo proposto. Entretanto, para um futuro produto este pode ser substituído por um PIC permitindo a redução do tamanho do circuito. Devido as suas dimensões e consumo não é viável que este seja utilizado num produto final, mas que seja implementada uma solução dedicada, baseada na funcionalidade comprovada e apresentada pelo protótipo.

4.2.12. Display *Touchscreen*

Para a validação do protótipo e apresentação da viabilidade técnica do uso do Display não foi necessário criar uma interface amigável entre o usuário e o sistema de medicamentos. Entretanto, para um possível produto, esta implementação é indispensável.

4.2.13. Integração

O resultado final do protótipo apresentou todas as funcionalidades descritas nesse trabalho. Entretanto, vale ressaltar que o tempo de processamento de imagem é de 0,7 fps para cada câmera. Como o processamento de imagem é utilizado apenas para detectar alterações dos frames, esse valor não é crítico para até 14 câmeras instaladas (aproximadamente 10 segundos), pois o envio da mensagem de texto apresentou tempo de resposta que variou entre 10 a 20 segundos. Dessa forma, garante-se que ao acessar o site, este apresente o monitoramento da última câmera que detectou algum evento. Para otimização da aplicação, ou utilização de um número maior de câmeras pode-se utilizar um algoritmo para restringir as câmeras de análise conforme descrito no item 5.2 (Trabalhos futuros) desta monografia.

O sistema também apresenta como limitação o alcance do Wi-Fi, entretanto, esta limitação pode ser superada com o uso de repetidores de sinais instalados na residência. Estes repetidores permitem a expansão da área de alcance do sinal sem a necessidade de alterações de projeto.

5. Conclusão e trabalhos futuros

5.1. Conclusão

Este trabalho apresentou-se como um dos mais complexos desenvolvidos durante o curso de graduação por abordar diversas especialidades e o aprendizado do funcionamento de diferentes dispositivos e ferramentas.

Por meio do estudo de dispositivos comerciais, como acelerômetro e barômetro, foi possível aprofundar os conhecimentos em transdutores. Também foram aplicados diversos conceitos de protocolos de comunicação como *wireless*, I²C e UART. Foi necessário o aprendizado de tecnologias atuais de comunicação como GSM, placas de desenvolvimento com microprocessadores de 32 bits com sistema operacional embarcado, linguagem de programação Python, bibliotecas de processamento de imagens (OpenCV), protocolos de comunicação *wireless* (Wi-Fi), uso de bibliotecas e programação do padrão e IDE Arduino, sistemas de arquivos Linux, uso de bibliotecas para envio de mensagem por WhatsApp e utilização de um display *touchscreen*.

Conforme discutido nos itens 4.2.1, 4.2.2 e 4.2.3 a placa Intel Galileo apresentou alguns problemas e inconsistências de software. Para solucionar esses problemas foi necessário entrar em contato com o suporte da Intel constantemente. Isso propiciou grande experiência na medida em que muitos dos defeitos detectados ainda não haviam sido encontrados por outras pessoas, devido à recente comercialização da placa. Atualmente, novas versões dos softwares foram lançados com a correção de parte dos problemas relatados. A busca por informações para solucionar os desafios enfrentados exigiu um tempo maior que o esperado, acarretando um atraso no desenvolvimento do projeto.

As limitações da placa de desenvolvimento Intel Galileo com foco em sua velocidade de processamento de imagens foram analisadas. O melhor desempenho para o algoritmo implementado apresentou tempo de processamento de 0,7 fps por câmera para a detecção de movimento. Este desempenho impossibilitou o uso de algoritmos de visão computacional mais sofisticados de detecção de velocidade ou aceleração típicos de queda livre.

Como solução implementou-se um sistema de detecção de queda utilizando transdutores e comunicação Wi-Fi. Esta solução foi uma proposta para viabilizar o projeto, mantendo o uso do kit de desenvolvimento Intel Galileo. Para garantir um bom desempenho, o uso dos módulos com processamento local foi essencial, pois possibilitou a distribuição do processamento de parte dos dados.

Este projeto contemplou o uso e intercomunicação de diferentes linguagens de programação como Python, scripts de inicialização e comandos Linux, linguagem de

programação dedicada a Arduino, e intercomunicação entre os programas Arduino e Linux. Por meio do desenvolvimento do projeto, além do aprendizado de novas ferramentas, foi possível observar como sistemas distintos se comunicam e se complementam para compor o sistema final.

Por sua organização em forma de tutorial, o presente trabalho poderá auxiliar no desenvolvimento de ferramentas e soluções apresentadas tanto para sistemas operacionais embarcados quanto para utilização e integração destes com transdutores e módulos apresentados. Dessa forma, o trabalho abrange uma contribuição tecnológica por apresentar os passos dos desenvolvimentos do projeto e de seus respectivos resultados.

O trabalho contemplou todas as etapas de um projeto, desde a identificação de uma dificuldade cotidiana de uma parcela de pessoas, a proposta de uma solução, identificação das limitações dessa solução e implementação de sistemas de contorno para as limitações encontradas. Exigiu, portanto, criatividade, aplicação de conhecimentos adquiridos durante a graduação e busca e pesquisa por soluções e tecnologias não constituintes da grade curricular da universidade.

Este trabalho também contribuiu para minha formação, pois exigiu a solução de diversos desafios que precisaram ser enfrentados. Estes foram superados graças a solida e diversificada base teórica obtida no curso de graduação que forneceu ferramentas para o aprofundamento em determinadas áreas e a possibilidade de buscar formas de contornar as limitações do sistema utilizado.

5.2. Trabalhos futuros

Este trabalho contemplou a construção de protótipos, porém, para sua aplicação como produto será necessário um estudo da eficácia e análise da adaptação do usuário com os dispositivos, realizando alterações para tornar sua interface e uso mais amigáveis ao usuário final.

Por se tratar de um protótipo este trabalho não abordou a implementação de sistemas eficientes de energia, como fontes chaveadas, necessários para os sistemas portáteis, ou de sistema de nobreak para manter o sistema funcionando mesmo com ausência de energia elétrica, entretanto, tais tecnologias devem ser estudadas e implementadas para um possível produto.

Para residências onde é necessário um grande número de câmeras, pode-se criar um algoritmo que restringe a análise e processamento de imagens em câmeras próximas dos locais onde são identificados movimentos. Dessa forma, o processamento de imagens

pode se restringe a até três câmeras simultâneas independentemente do número de câmeras disponíveis na residência.

Utilizando o sistema de processamento de imagens com câmeras IPs, pode-se implementar um algoritmo para ignorar movimentos em determinada região da câmera. Esse sistema pode ser útil quando se deseja verificar se apenas a pessoa acamada tenta sair da cama sozinha, neste caso, a região de movimento a ser ignorada seria a cama.

6. Bibliografia

[1] UFG – REE – [2006] - Revista Eletrônica de Enfermagem - EVOLUÇÂO HISTÓRICA E IMPACTO DA TECNOLOGIA NA ÁREA DA SAÚDE E DA ENFERMAGEM - v. 08, n. 03, p. 422 - 430.
Disponível em <http://www.fen.ufg.br/revista/revista8_3/v8n3a13.htm>
Acesso em 01 de novembro de 2015

[2] JMIR , Journal of Medical Internet Research, Della Mea, Vincenzo (2001). "What is e-Health: The death of telemedicine?" Retrieved 2012-04-15.
Disponível em <http://www.jmir.org/2001/2/e22/>
Acesso em 01 de novembro de 2015

[3] JMIR, Journal of Medical Internet Research "What Is eHealth: A Systematic Review of Published Definitions". Retrieved 5 February 2012.
Disponível em <http://www.jmir.org/2005/1/e1/>
Acesso em 01 de novembro de 2015

[4] INFOBRASIL, Mobile Health: Utilização de tecnologias móveis na saúde será discutida durante a InfoBrasil 2015.
Disponível em <http://www.infobrasil.inf.br/noticia/mobile-health-utilizacao-de-tecnologias-moveis-na-saude-sera-discutida-durante-infobrasil-20>
Acesso em 01 de novembro de 2015.

[5] SAUDEBUSINESS - Investimentos corporativos na tecnologia em saúde batem recorde – Gordilho, Rafael – 07/11/2014
Disponível em <http://saudebusiness.com/noticias/investimentos-corporativos-em-saude-digital-e-tecnologia-em-saude-batem-recorde/>
Acesso em 01 de novembro de 2015

[6] THE WALL STREET JOURNAL, Google Backs Startup Oscar Health Insurance, 15/09/2015.
Disponível em: <http://www.wsj.com/articles/google-backs-startup-oscar-health-insurance-1442374756>
Acesso em 01 de novembro de 2015

[7] IBGE - Tábua Completa de Mortalidade para o Brasil de 2012.
Disponível em <http://www.ibge.gov.br/home/estatistica/populacao/tabuadevida/2012/>
Acesso em 12 de julho de 2015

[8] Gomes, H. O. e Caldas, C. P.; Envelhecimento Humano. Revista Hospital Universitário Pedro Ernesto Vol 7, Número 1 – Jan/Jun – 2008.
Disponível em <http://revista.hupe.uerj.br/detalhe_artigo.asp?id=195>
Acesso em 01 de novembro de 2015

[9] Adriano de, Peter MarikSistema (Pontifícia universidade católica do paraná) - Detector de Quedas (SDQ)
Disponível em <http://www.ppgia.pucpr.br/~laplima/ensino/pfec/concluidos/1sem2008/sdq.pdf>
Acesso em 19 de julho de 2015

[10] Joao Paulo Dupinska de Oliveira, Kaya Sumire Abe, Leandro Vinicius Silva Forneck, Matheus Rezende Oliveira, (Universidade Tecnológica Federal do Paraná) - Sistema de Monitoramento de quedas para pessoas com deficiência motora
Disponível em <http://www.dainf.ct.utfpr.edu.br/~fabro/IF66J/Relatorios_Finais/2012_2/Sistema%20Monitor%20de%20Quedas/MonografiaFinal_Sistema_Monitor_Quedas_2012_2.pdf>
Acesso em 19 de julho de 2015

[11] INTERNET OF THINGS, Rolf H. Weber, Romana Weber, Editora: Springer Berlin Heidelberg

[12] Real-Time Concepts for Embedded Systems - Qing Li and Carolyn Yao - CMP Books, 2003

[13] Operating systems – internal and design principles – William Stallings - 7ed - Prentice Hall

[14] Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers - Tammy Noergaard - Newnes, 2005

[15] Borges, Rodrigo Weissmann – Aplicabilidade de Sistemas Operacionais de Tempo Real (RTOS) para sistemas embarcados de baixo custo e pequeno porte. 2011. Dissertação (Mestrado em Ciências, Programa de Engenharia Elétrica) – Escola de Engenharia de São Carlos – Universidade de São Paulo.

[16] Weiser, Mark (1991). "The Computer for the 21st Century"
Disponível em <http://dl.acm.org/citation.cfm?id=329126>
Acesso em 18 de novembro de 2015

[17] Internet das Coisas - IoT/M2M - DEV Tecnologia
Disponível em <http://devtecnologia.com.br/internet-das-coisas-iot/>
Acesso em 18 de novembro de 2015

[18] Lígia J. Figueiredo, Ana R. Gafaniz, Gustavo S. Lopes e Rúben Pereira. Aplicações de Acelerómetros. IAS 2007 – Instrumentação e Aquisição de Sinais. Monografia. Lisboa, Portugal, 19 Dezembro 2007

[19] Ranjit Singh, Low Lee Ngo, Ho Soon Seng, “A Silicon Piezoresistive Pressure Sensor”, IEEE, pp. 1-4, 2002

[20] Richard D. Schneeman, "Implementing a Standards-based Distributed Measurement and Control Application on the Internet "
Disponível em <http://ieee1451.nist.gov/framework.pdf>
Acesso em 18 de novembro de 2015

[21] WUNNAYA, S. V.; HOO, P.. Remote Instrumentation Access and Control.
(RIAC) Through Inter-network, 1999Proceeding IEEE

[22] BORGES, A. P.. Instrumentação Virtual Aplicada a uma Laboratório com Acesso pela Internet, 2002. Dessertação de Mestrado, Escola Politécnica da universidade de São Paulo - Departamento de Sistemas Eletrônicos, São Paulo.

[23] Gonzalez, R.C. e Woods, R.E. e Eddins, S.L., Digital Image Processing, Pearson, 2010.

[24] IEEE Std 802.11 (ISO / IEC 8802-11: 1999)

[25] Schiller,Jochen; - Mobile Communications – Segunda Edição, Pearson Education Limited, 2003.

[26] Ramon, M. C.; Intel Galileo and Intel Galileo Gen 2. API Features and Arduino Projects for Linux Programmers. Apress open.

[27] ARDUINO. <https://www.arduino.cc/en/ArduinoCertified/IntelGalileo>

[28] SPARKFUN. - Galileo Getting Started Guide
Disponível em <https://learn.sparkfun.com/tutorials/galileo-getting-started-guide>
Acesso em 25 de junho de 2015

[29] IBM. Desenvolva Distribuições Integradas e Customizadas do Linux com o Projeto Yocto.
Disponível em <http://www.ibm.com/developerworks/br/library/l-yocto-linux/>
Acesso em 05 de setembro de 2015

[30] Rossi, Henrique Persico - BeagleBone Black + Yocto
Disponível em <http://www.embarcados.com.br/beaglebone-black-yocto/>
Acesso em 05 de setembro de 2015

[31] Borges, L. E.; Python para Desenvolvedores. Rio de Janeiro, Edição do Autor, Editora Novatec, 2010.

[32] Robert W. Sebesta. Conceitos de Linguagem de Programação, 9ª edição, Porto Alegre: Bookman, 2011.

[33] Marengoni, M; Stringhini, S., 2009, Revista de Informática Teórica e Aplicada, Volume 16, Edição 1

[34] Informações fornecidas pelo fabricante da Camera IP Showtec
Disponível em <http://www.showtec.com.br/produtos/cameras-ip>
Acesso em 13 de dezembro de 2014

[35] Analog Devices, Manual de dados do fabricante ADXL335
Disponível em <http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf>
Acesso em 19 de setembro de 2015

[36] TINYDEAL, GY-61 ADXL335 Triple Axis Accelerometer / Analog Sensor for Arduino
Disponível em <http://www.tinydeal.com/gy-61-adxl335-triple-axis-accelerometer-analog-sensor-blue-p-128418.html>
Acesso em 04 de novembro de 2015

[37] INVENSENSR, Manual de dados do fabricante família MPU-6000
Disponível em <http://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>
Acessado em 19 de setembro de 2015

[38] BITIFY, blog, Imagem MPU6050
Disponível em <http://blog.bitify.co.uk/2013/11/interfacing-raspberry-pi-and-mpu-6050.html>
Acesso em 04 de novembro de 2015

[39] BOSH, Manual de dados do fabricante BMP180.
Disponível em <https://www.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>
Acesso em 26 de setembro de 2015

[40] INSTRUCTABLES. Imagem BMP180 .
Disponível em <http://www.instructables.com/id/TFT-Environment-Monitor-using-BMP180-DHT11/>
Acesso em 04 de novembro de 2015

[41] Tutorial – Arduino and SIM900 GSM Modules
Disponível em <http://tronixstuff.com/2014/01/08/tutorial-arduino-and-sim900-gsm-modules/>
Acesso em 05 de novembro de 2015

[42] Pololu Robotics & Eletronics - Sensor MQ-2 Manual de dados do fabricante
Disponível em <https://www.pololu.com/file/download/MQ2.pdf?file_id=0J309>
Acesso em 17 de outubro de 2015

[43] Arduino e Cia - Alarme sensor de gás com o módulo MQ-2
Disponível em <http://www.arduinoecia.com.br/2015/01/alarme-sensor-de-gas-modulo-mq-2.html>
Acesso em 22 de agosto de 2015

[44] Curvello, André - Apresentando o módulo ESP8266
Disponível em <http://www.embarcados.com.br/modulo-esp8266/>
Acesso em 29 de agosto de 2015

[45] ELECTROSCHEMATICS. Imagem ESP8266 – 01
Disponível em <http://www.electroschematics.com/11276/esp8266-datasheet/>
Acesso em 04 de novembro de 2015

[46] ITEADSTUDIO - ESP8266 Serial WIFI Module
Disponível em <http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module>
Acesso em 29 de agosto de 2015

[47] MAKEHACKVOID. Imagem ESP8266 – 12
Disponível em https://wiki.makehackvoid.com/projects:group_projects:esp2866
Acesso em 04 de novembro de 2015

[48] ARDUINO. Arduino MEGA 2560
Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>
Acesso em 12 de setembro de 2015.

[49] MATHWORKS. Imagem Arduino Mega2560
Disponível em <http://www.mathworks.com/company/newsletters/articles/teaching-and-learning-resources-project-based-learning.html>
Acesso em 04 de novembro de 2015

[50] DWIN – DMT48270_M043_02_W Product Specification
Disponível em <http://www.dwin.com.cn/uploads/English%20Documents/DMT48270M043_02W_DATASHEET.pdf>
Acesso em 02 de setembro de 2015.

[51] IP Camera CGI MANUAL
Disponível em:
http://corz.org/windows/software/oodlecam/files/IP%20Camera%20CGI%20Manual%20[from%20Tenvis%203815%20SDK].pdf
Acessado em 05 de novembro de 2015

[52] Axis, Axis JPEG Format – Comment Fields
Disponível em <http://www.axis.com/files/tech_notes/JPEG_format_1_02.pdf>
Acesso em 26 de setembro de 2015

[53] Dell Inspiron 14 (N4050)
Disponível em <www.pcworld.com/product/1278065/dell-inspiron-14-n4050-notebook.html>
Acesso em 28 de outubro 2015.

[54] Curvello, André – Sistemas embarcados - Intel Edison com WhatsApp
Disponível em <http://www.embarcados.com.br/intel-edison-com-whatsapp/>
Acesso em 04 de setembro de 2015

[55] VICTORVISION, Especificação de comandos DGUS
Disponível em :
<http://victorvision.com.br/images/DocumentosAtualizados/DGUS_Commands_Specification.pdf>
Acesso em 02 de setembro de 2015

[56] Hahn, Matthias - Efficient Communication Between Arduino and Linux Native Processes
Disponível em <https://software.intel.com/pt-br/blogs/2014/09/22/efficient-communication-between-arduino-and-linux-native-processes>
Acesso em 29 de agosto de 2015

[57] Yocto Project, It's not an embedded Linux distribution - it creates a custom one for you
Disponível em <https://www.yoctoproject.org/>
Acesso em 25 de abril de 2015

Apêndice

1 - Programa utilizado para acessar o Linux por meio da interface Arduino

```
void setup()
{
 system("cp /etc/inittab /etc/inittab.bak");  // Back up inittab
 // Replace all "S:2345" with "S0:2345"'s (switching serial ports):
 system("sed -i 's/S:2345/S0:2345/g' /etc/inittab");
 // Replace all "ttyS1" with "ttyGS0"'s (switching serial ports):
 system("sed -i 's/ttyS1/ttyGS0/g' /etc/inittab");
 // Replace all "grst" with "#grst"'s to comment that line out:
 system("sed -i 's/grst/#grst/g' /etc/inittab");
 // Replace all "clld" with "#clld"'s to comment that line out:
 system("sed -i 's/clld/#clld/g' /etc/inittab");
 system("kill -SIGHUP 1");
}
void loop()
{
}
```

Algoritmo A - 1: Programa Arduino utilizado para acessar linux utilizando comunicação USB

2 - Programa utilizado para obter endereço IP dinâmico do Linux por meio da interface Arduino

```
void setup() {
 //Ativar o servidor Telnet
 system("telnetd -l /bin/sh");
}

void loop() {
 system("ifconfig eth0 > /dev/ttyGS0");
}
```

Algoritmo A - 2: Programa Arduino utilizado para obter endereço IP dinâmico

3 - Algoritmo implementado para calcular o tempo de processamento da formação da imagem

```
import cv2
import urllib2
import numpy as np
import datetime

stream=urllib2.urlopen('http://143.107.235.59:8086/video.cgi?user=galileo&pwd=la
bytes=''
while True:
    aux = 0
    print datetime.datetime.now().time()
    while (aux<100):
        bytes+=stream.read(1024)
        a = bytes.find('\xff\xd8')
        b = bytes.find('\xff\xd9')
        if a!=-1 and b!=-1:
            jpg = bytes[a:b+2]
            bytes= bytes[b+2:]
            i = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8),cv2.CV_LOAD_IMAG
        aux += 1
```

Algoritmo A - 3: Programa utilizado para calcular o tempo de processamento de uma imagem obtida por uma câmera IP

4 - Algoritmo utilizado para detecção de movimento

```
# Inclusão de bibliotecas
import argparse
import datetime
import imutils
import time
import cv2

# Construção do argumentos, threshold de pixel adjacentes para detectar movimento
#de 500 pixels
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help= "path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

# Caso o endereço não seja encontrado, a câmera utilizada será a default (câmera 0)
if args.get("video", None) is None:
        camera = cv2.VideoCapture(0)
        time.sleep(0.25)

# Caso o Endereço seja encontrado
else:
        camera = cv2.VideoCapture(args["video"])

# Inicialização do primeiro frame
firstFrame = None

# loop de verificação
while True:
        # Leitura da camera e escrita occupied/unoccupied
        (grabbed, frame) = camera.read()
        text = "Unoccupied"

        # Se o frame não for encontrado então o processo é parado
        if not grabbed:
                break

        # Redimensionamento do quadro, conversão para tons de cinza, aplicado filtro
        frame = imutils.resize(frame, width=500)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (21, 21), 0)

        # Inicializa a primeira imagem caso esta já não tenha inicializado
        if firstFrame is None:
                firstFrame = gray
                continue
        cv2.imshow("firstFrame", firstFrame)

      # Compara a imagem atual e a imagem salva em firstFrame
```

```
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

        # Aplica dilatação e posteriormente detecção de contornos
        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
                cv2.CHAIN_APPROX_SIMPLE)

        # Verificação dos contornos
        for c in cnts:
                # Se o contorno for pequeno, ignore-o
                if cv2.contourArea(c) < args["min_area"]:
                        continue

# Calcular a caixa delimitadora do movimento, desenhá-la no frame e atualizar texto
                (x, y, w, h) = cv2.boundingRect(c)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                text = "Occupied"

        # Desenha o texto e as informações de data e hora na imagem
        cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
        cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
                (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0,
255), 1)

        # Apresenta os resultados gráficos
        cv2.imshow("Security Feed", frame)
        cv2.imshow("Thresh", thresh)
        cv2.imshow("Frame Delta", frameDelta)
        key = cv2.waitKey(1) & 0xFF

        # Inclui tecla para sair da execução do progama
        if key == ord("q"):
                break

# Limpa a variável camera e fecha todas as janelas
camera.release()
cv2.destroyAllWindows()
```

Algoritmo A - 4: Algoritmo implementado para detecção de movimento da *webcam*

5 - Algoritmo utilizado para processamento de imagens de uma câmera IP e detecção de movimentos. A figura de referência é atualizada a cada 100 frames.

```
import cv2
import urllib2
import numpy as np
import imutils
import time
import argparse
import datetime

stream=urllib2.urlopen('http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha
&url=video.mjpeg')
bytes=''

ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

firstFrame = None
cont = 0
while True:
    bytes+=stream.read(1024)
    a = bytes.find('\xff\xd8')
    b = bytes.find('\xff\xd9')
    if a!=-1 and b!=-1:
        jpg = bytes[a:b+2]
        bytes= bytes[b+2:]
        frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (21, 21), 0)

        if cont > 100:
            firstFrame = None
            cont = 0

        if firstFrame is None:
            firstFrame = gray
            continue

        cont = cont+1
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)
```

```
        for c in cnts:
            # verifica se o contorno for muito pequeno, este será ignorado
            if cv2.contourArea(c) < args["min_area"]:
                continue
            (x, y, w, h) = cv2.boundingRect(c)
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            print 'detectado'

        cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
            (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

        cv2.imshow('FF',firstFrame)
        cv2.imshow('frame',frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break
stream.close()
cv2.destroyAllWindows()
```

Algoritmo A - 5: Algoritmo implementado para detecção de movimento em imagens capturadas pela câmera IP

6 - Código implementado para obter a velocidade de processamento de imagens na placa Intel Galileo. Este código apresenta o tempo de processamento da detecção de movimento registrada por uma câmera IP em tempo de execução. A cada 100 ciclos do programa, isto é, 100 aquisições e processamento de imagem é apresentado o tempo resultante.

```
import cv2
import urllib2
import numpy as np
import imutils
import time
import argparse
import datetime

stream=urllib2.urlopen('http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha
&url=video.mjpeg')
bytes=''

ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
args = vars(ap.parse_args())

firstFrame = None
cont = 0
while True:
    bytes+=stream.read(1024)
    a = bytes.find('\xff\xd8')
    b = bytes.find('\xff\xd9')
    if a!=-1 and b!=-1:
        jpg = bytes[a:b+2]
        bytes= bytes[b+2:]
        frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
#         gray = cv2.GaussianBlur(gray, (21, 21), 0)

        if cont > 100:
            print datetime.datetime.now().time()
            firstFrame = None
            cont = 0

        if firstFrame is None:
            firstFrame = gray
            continue

        cont = cont+1
        frameDelta = cv2.absdiff(firstFrame, gray)
```

```
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]

        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)

        for c in cnts:
            # verifica se o contorno for muito pequeno, este será ignorado
            if cv2.contourArea(c) < args["min_area"]:
                continue
 #           (x, y, w, h) = cv2.boundingRect(c)
 #           cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
 #           print 'detectado'

#       cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
#           cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
#       cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
#           (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
1)



        key = cv2.waitKey(1) & 0xFF

        if key == ord("q"):
            break
stream.close()
cv2.destroyAllWindows()
```

Algoritmo A - 6: Código implementado para calcular tempo de aquisição e processamento de imagens de câmera IP

7 - Algoritmo otimizado para identificação de movimento, com atualização da figura de referência. Ao detectar movimento o programa escreve o número da referida câmera IP em um arquivo.

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('/usr/lib/python2.7/site-packages')
import cv2
import numpy as np
import time
import requests
import threading
import urllib2
import imutils
import argparse
import datetime
from threading import Thread, Event, ThreadError

class Cam0():

  def __init__(self, url):

    self.stream = requests.get(url, stream=True)
    self.thread_cancelled = False
    self.thread = Thread(target=self.run)
    print "camera initialised"

  def start(self):
    self.thread.start()
    print "camera stream started"

  def run(self):
    bytes=''
    firstFrame = None
    cont = 0
    while not self.thread_cancelled:
      try:
        bytes+=self.stream.raw.read(1024)
        a = bytes.find('\xff\xd8')
        b = bytes.find('\xff\xd9')
        if a!=-1 and b!=-1:
          jpg = bytes[a:b+2]
          bytes= bytes[b+2:]
          frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
          gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

          if cont > 100:
            firstFrame = None
```

```
          cont = 0

        if firstFrame is None:
          firstFrame = gray
#           cv2.imwrite("teste referenceframe.png",firstFrame);
          continue

        cont = cont+1
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
#         cv2.imwrite("framedelta.png",firstFrame);
        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
           cv2.CHAIN_APPROX_SIMPLE)

        for c in cnts:
          if cv2.contourArea(c) < args["min_area"]:
             continue
          arq = open("/media/mmcblk0p1/test.txt", 'w')
          texto = []
          texto.append('Cam00\n')
          arq.writelines(texto)
          arq.close()

#         cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
#           (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
1)

#         cv2.imshow('FF',firstFrame)
#         cv2.imshow('frame',frame)
        if cv2.waitKey(1) ==27:
          exit(0)
      except ThreadError:
        self.thread_cancelled = True


  def is_running(self):
    return self.thread.isAlive()


  def shut_down(self):
    self.thread_cancelled = True
    #block while waiting for thread to terminate
    while self.thread.isAlive():
      time.sleep(1)
    return True

class Cam1():

  def __init__(self, url):
```

```
    self.stream = requests.get(url, stream=True)
    self.thread_cancelled = False
    self.thread = Thread(target=self.run)
    print "camera initialised"



  def start(self):
    self.thread.start()
    print "camera stream started"

  def run(self):
    bytes=''
    firstFrame = None
    cont = 0
    while not self.thread_cancelled:
      try:
        bytes+=self.stream.raw.read(1024)
        a = bytes.find('\xff\xd8')
        b = bytes.find('\xff\xd9')
        if a!=-1 and b!=-1:
          jpg = bytes[a:b+2]
          bytes= bytes[b+2:]
          frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
          gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

          if cont > 100:
            firstFrame = None
            cont = 0

          if firstFrame is None:
            firstFrame = gray
#             cv2.imwrite("teste referenceframe.png",firstFrame);
            continue

          cont = cont+1
          frameDelta = cv2.absdiff(firstFrame, gray)
          thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
#           cv2.imwrite("framedelta.png",firstFrame);
          thresh = cv2.dilate(thresh, None, iterations=2)
          (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
             cv2.CHAIN_APPROX_SIMPLE)

          for c in cnts:
            if cv2.contourArea(c) < args["min_area"]:
               continue
            arq = open("/media/mmcblk0p1/test.txt", 'w')
            texto = []
            texto.append('Cam01\n')
```

```
            arq.writelines(texto)
            arq.close()

#           cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
#             (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255),
1)

#           cv2.imshow('FF1',firstFrame)
#           cv2.imshow('frame1',frame)
          if cv2.waitKey(1) ==27:
            exit(0)
      except ThreadError:
        self.thread_cancelled = True


  def is_running(self):
    return self.thread.isAlive()


  def shut_down(self):
    self.thread_cancelled = True
    #block while waiting for thread to terminate
    while self.thread.isAlive():
      time.sleep(1)
    return True

if __name__ == "__main__":
  url =
'http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha&url=video.mjpeg'
  cam = Cam0(url)
  cam.start()
  ap = argparse.ArgumentParser()
  ap.add_argument("-v", "--video", help="path to the video file")
  ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size")
  args = vars(ap.parse_args())
  url1 =
'http://143.107.235.59:8085/video.cgi?user=usuário&pwd=senha&url=video.mjpeg'
  cam1 = Cam1(url1)
  cam1.start()
```

Algoritmo A - 7: Algoritmo implementado e otimizado para detecção de movimento e escrita em arquivos.

8 - Código implementado para obter a velocidade de processamento de imagens na placa Intel Galileo. Este código apresenta o tempo de processamento da detecção de movimento registrada por duas câmeras IPs em tempo de execução e escrita em arquivos. A cada 10 ciclos do programa, isto é, 10 aquisições e processamento de imagem é apresentado o tempo resultante.

```
# -*- coding: utf-8 -*-
import sys
sys.path.append('/usr/lib/python2.7/site-packages')
import cv2
import numpy as np
import time
import requests
import threading
import urllib2
import imutils
import argparse
import datetime
import os
from threading import Thread, Event, ThreadError

class Cam0():

  def __init__(self, url):

    self.stream = requests.get(url, stream=True)
    self.thread_cancelled = False
    self.thread = Thread(target=self.run)
    print "camera initialised"



  def start(self):
    self.thread.start()
    print "camera stream started"

  def run(self):
    bytes=''
    firstFrame = None
    cont = 0
    while not self.thread_cancelled:
      try:
        bytes+=self.stream.raw.read(1024)
        a = bytes.find('\xff\xd8')
        b = bytes.find('\xff\xd9')
        if a!=-1 and b!=-1:
          jpg = bytes[a:b+2]
          bytes= bytes[b+2:]
```

```
        frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        if cont > 100:
          print datetime.datetime.now().time()
          firstFrame = None
          cont = 0

        if firstFrame is None:
          firstFrame = gray
#           cv2.imwrite("teste referenceframe.png",firstFrame);
          continue

        cont = cont+1
        frameDelta = cv2.absdiff(firstFrame, gray)
        thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
#         cv2.imwrite("framedelta.png",firstFrame);
        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
           cv2.CHAIN_APPROX_SIMPLE)

        for c in cnts:
          if cv2.contourArea(c) < args["min_area"]:
             continue
#           (x, y, w, h) = cv2.boundingRect(c)
#           cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
          arq = open("/media/mmcblk0p1/test.txt", 'w')
          texto = []
          texto.append('Cam00\n')
          arq.writelines(texto)
          arq.close()

#         cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
#           (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0,
255), 1)

#         cv2.imshow('FF',firstFrame)
#         cv2.imshow('frame',frame)
        if cv2.waitKey(1) ==27:
          exit(0)
    except ThreadError:
      self.thread_cancelled = True


  def is_running(self):
    return self.thread.isAlive()


  def shut_down(self):
```

```
    self.thread_cancelled = True
    #block while waiting for thread to terminate
    while self.thread.isAlive():
      time.sleep(1)
    return True

class Cam1():

  def __init__(self, url):

    self.stream = requests.get(url, stream=True)
    self.thread_cancelled = False
    self.thread = Thread(target=self.run)
    print "camera initialised"



  def start(self):
    self.thread.start()
    print "camera stream started"

  def run(self):
    bytes=''
    firstFrame = None
    cont = 0
    while not self.thread_cancelled:
      try:
        bytes+=self.stream.raw.read(1024)
        a = bytes.find('\xff\xd8')
        b = bytes.find('\xff\xd9')
        if a!=-1 and b!=-1:
          jpg = bytes[a:b+2]
          bytes= bytes[b+2:]
          frame = cv2.imdecode(np.fromstring(jpg,
dtype=np.uint8),cv2.CV_LOAD_IMAGE_COLOR)
          gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

          if cont > 100:
            print datetime.datetime.now().time()
            firstFrame = None
            cont = 0

          if firstFrame is None:
            firstFrame = gray
#             cv2.imwrite("teste referenceframe.png",firstFrame);
            continue

          cont = cont+1
          frameDelta = cv2.absdiff(firstFrame, gray)
          thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]
#           cv2.imwrite("framedelta.png",firstFrame);
```

```
        thresh = cv2.dilate(thresh, None, iterations=2)
        (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
          cv2.CHAIN_APPROX_SIMPLE)

        for c in cnts:
          if cv2.contourArea(c) < args["min_area"]:
            continue
#           (x, y, w, h) = cv2.boundingRect(c)
#           cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
          arq = open("/media/mmcblk0p1/test.txt", 'w')
          texto = []
          texto.append('Cam01\n')
          arq.writelines(texto)
          arq.close()
#         cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y
%I:%M:%S%p"),
#           (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0,
255), 1)
        if cv2.waitKey(1) ==27:
          exit(0)
      except ThreadError:
        self.thread_cancelled = True

  def is_running(self):
    return self.thread.isAlive()

  def shut_down(self):
    self.thread_cancelled = True
    #block while waiting for thread to terminate
    while self.thread.isAlive():
      time.sleep(1)
    return True

if __name__ == "__main__":
  url =
'http://143.107.235.59:8086/video.cgi?user=usuário&pwd=senha&url=video.mjpeg'
  cam = Cam0(url)
  cam.start()
  ap = argparse.ArgumentParser()
  ap.add_argument("-v", "--video", help="path to the video file")
  ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area
size")
  args = vars(ap.parse_args())
  url1 =
'http://143.107.235.59:8085/video.cgi?user=usuário&pwd=senha&url=video.mjpeg'
  cam1 = Cam1(url1)
  cam1.start()
```

Algoritmo A - 8: Código implementado para calcular velocidade de processamento do Algoritmo A - 7.

9 - Programa Arduino implementado para envio de mensagem SMS utilizando o módulo GSM.

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "sms.h"
SMSGSM sms;

int numdata;
boolean started=false;
char smsbuffer[160];
char n[33];

char sms_position;
char phone_number[33];
char sms_text[100];
int i;

void setup()
{
   Serial.begin(9600);

   if (gsm.begin(9600))
   {
      Serial.println("\nstatus=READY");
      started=true;
   }
   else
      Serial.println("\nstatus=IDLE");

   if(started)
   {
      if (sms.SendSMS("+55199927779XX", "Arduino SMS"))
      {
        Serial.println("\nSMS sent OK.");
      }
      else
      {
        Serial.println("\nError sending SMS.");
      }
   }
};

void loop()
{
   if(started)
   {
      sms_position=sms.IsSMSPresent(SMS_UNREAD);
      if (sms_position)
      {
         Serial.print("SMS postion:");
         Serial.println(sms_position,DEC);
         sms.GetSMS(sms_position, phone_number, sms_text, 100);
```

```
            Serial.println(phone_number);
            Serial.println(sms_text);
        }
        delay(2000);
    }
};
```

Algoritmo A - 9: Código implementado para envio de mensagem SMS

10 - Leitura e atuação do acelerômetro ADXL335

```
const int xpin = A3;                  // x-axis of the accelerometer
const int ypin = A4;                  // y-axis
const int zpin = A5;                  // z-axis (only on 3-axis models)
const float const_conv = 204.6;
int set_alarm=0;
float X0,X1;
float Y0,Y1;
float Z0,Z1;
int i=0;
float dt = 200,dx,dy,dz;

void setup()
{
 // initialize the serial communications:
 Serial.begin(9600);

 //analogReference(EXTERNAL);
 pinMode(52, OUTPUT);
 Serial.println("  ++++ ADXL335 Sensor ++++");
 Serial.println();
 Serial.print("X Sensor");

 Serial.print("\t");
 Serial.print("Y Sensor");

 Serial.print("\t");
 Serial.println("Z Sensor");
}

void loop()
{
  if(i == 0){
     X0 = analogRead(xpin);
     Y0 = analogRead(ypin);
     Z0 = analogRead(zpin);
     i++;
  }
  else{
    delay(dt);
     X1 = analogRead(xpin);
     Y1 = analogRead(ypin);
     Z1 = analogRead(zpin);
     i = 0;
    dx = abs(X1-X0)/const_conv;
    dy = abs(Y1-Y0)/const_conv;
    dz = abs(Z1-Z0)/const_conv;

     Serial.print(dx);
```

```
    Serial.print("\t");
    Serial.print("\t");

    Serial.print(dy);
    Serial.print("\t");
    Serial.print("\t");

    Serial.print(dz);
    Serial.println();

    if(dx > 1.0 || dy > 1.0 || dz > 1.0){
      set_alarm = 1;
    }
    if (set_alarm == 1){
      digitalWrite(52, HIGH);
    }
  }
}
```

Algoritmo A - 10: Código implementado para leitura e atuação do acelerômetro ADXL335

11 - Leitura e atuação do acelerômetro MPU6050

```
#include<Wire.h>
const int MPU=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
int16_t dx,dy,dz;
int OldAcX,OldAcY,OldAcZ,OldTmp,OldGyX,OldGyY,OldGyZ;

int AcSensitivity = 10000;
boolean moved = false;

void setup(){
Wire.begin();
Wire.beginTransmission(MPU);
Wire.write(0x6B); // PWR_MGMT_1 register
Wire.write(0); // set to zero (wakes up the MPU-6050)
Wire.endTransmission(true);
Serial.begin(9600);
}

void loop(){
Wire.beginTransmission(MPU);
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);
Wire.requestFrom(MPU,14,true);
AcX=Wire.read()<<8|Wire.read();
AcY=Wire.read()<<8|Wire.read();
AcZ=Wire.read()<<8|Wire.read();
Tmp=Wire.read()<<8|Wire.read();
GyX=Wire.read()<<8|Wire.read();
GyY=Wire.read()<<8|Wire.read();
GyZ=Wire.read()<<8|Wire.read();

dx = abs(AcX - OldAcX);
dy = abs(AcY - OldAcY);
dz = abs(AcZ - OldAcZ);

if (dx > AcSensitivity || dy > AcSensitivity || dz > AcSensitivity) {
moved = true;
}

if (moved == true) {
Serial.println("MOVED");
}
OldAcX = AcX;
OldAcY = AcY;
OldAcZ = AcZ;
moved = false;
delay(100);
}
```

Algoritmo A - 11: Código implementado para leitura e atuação do acelerômetro MPU6050

12 - Algoritmo implementado para o cálculo da variação da altitude em função da pressão atmosférica

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
Adafruit_BMP085 bmp;
void setup() {
  Serial.begin(9600);
  if (!bmp.begin()) {
        Serial.println("Could not find a valid BMP085 sensor, check wiring!");
        while (1) {}
  }
}
void loop() {
  int i,media = 100;
  double temp[media],a=0,b,c;
//    Serial.print("Temperature = ");
//    Serial.print(bmp.readTemperature());
//    Serial.println(" *C");
//
//    Serial.print("Pressure = ");
//    Serial.print(bmp.readPressure());
//    Serial.println(" Pa");
//
//    // Calculate altitude assuming 'standard' barometric
//    // pressure of 1013.25 millibar = 101325 Pascal
//    Serial.print("Altitude = ");
//    Serial.print(bmp.readAltitude());
//    Serial.println(" meters");
//
//    Serial.print("Pressure at sealevel (calculated) = ");
//    Serial.print(bmp.readSealevelPressure());
//    Serial.println(" Pa");

while(1){
  b = a;
  a=0;
  for(i=0;i<media;i++){

  temp[i] = (bmp.readAltitude(101500))/media;
  a += temp[i];
  }

  delay(1700);
  //delay(1);
 //   Serial.print("Real altitude = ");
```

```
  //   Serial.print(a);
  //   Serial.println(" meters");
       c = a-b;
  if (c > 0.5 || c < -0.5 ){
//     Serial.print(a);
//     Serial.println(" ");
     Serial.println("**queda**");
  }
      Serial.print(c);
     Serial.println();
}
}
```

Algoritmo A - 12: Algoritmo implementado calcular altitude com base na informação da pressão atmosférica fornecida pelo sensor BMP180

13 - Programa do módulo ESP8266 utilizado para comunicação dos transdutores à placa de desenvolvimento Intel Galileo.

```
#include <ESP8266WiFi.h>

const char* ssid     = "REDE";
const char* password = "SENHA";

const char* host = "143.107.235.59";

void setup() {
  Serial.begin(115200);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  delay(10000);

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 8152;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URL for the request
  String url = "/?LED2;";
```

```
  Serial.print("Requesting URL: ");
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}
```

Algoritmo A - 13: Código implementado no módulo ESP8266 e utilizado para conexão e comunicação com a placa Intel Galileo

14 - Programa utilizado no módulo detector de vazamento de gás

```
#include <ESP8266WiFi.h>

const char* ssid     = "REDE";
const char* password = "SENHA";

const char* host = "143.107.235.59";

void setup() {
  Serial.begin(115200);
  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if(analogRead(A0) > 600){

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 8152;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url = "/?GAS;";

  Serial.print("Requesting URL: ");
```

```
  Serial.println(url);

  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(10000);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
  }
}
```

Algoritmo A - 14: Programa implementado para o módulo detectar vazamento de gás

15 - Algoritmo completo da placa de desenvolvimento Intel Galileo, incluindo comunicação entre o processo Arduino e o Linux.

```
#include <Ethernet.h>
#include <SPI.h>
#include <SD.h>

boolean reading = false;
String readString;
//const int chipSelect = 4;

// http request termina com uma linha em branco
boolean currentLineIsBlank = true;
boolean sentHeader = false;
String header;

////////////////////////////////////////////////////////////////////
//CONFIGURACAO
////////////////////////////////////////////////////////////////////
  byte ip[] = { 192, 168, 1, 151 };   //Para setup manual
  byte gateway[] = { 192, 168, 0, 1 }; //Para setup manual
  byte subnet[] = { 255, 255, 255, 0 }; //Para setup manual

  // Caso precise alterar o MAC address
  byte mac[] = { 0x98, 0x4F, 0xEE, 0x01, 0x13, 0x30 };

  EthernetServer server = EthernetServer(8152); //port 80
////////////////////////////////////////////////////////////////////

void setup(){
  Serial.begin(115200);

  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(SS, OUTPUT);

//  if (!SD.begin(chipSelect)) {
//    Serial.println("initialization failed!");
//    return;
//  }

  Ethernet.begin(mac);

  server.begin();
  Serial.println(Ethernet.localIP());
}

void loop(){
```

```
// listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        header += c;

        if (c == '\n' && currentLineIsBlank) {

          //parse headers
          //bWluaDp0ZXN0 = 'minh:test' (user:password) base64 encode

          Serial.print(header);

          // Simpler just to find the credential string
          // send a standard http response header
          if(header.indexOf("dGhvbWFzOnZpYWRv") >= 0) {
            //successful login
            while(1)
            checkForClient();
            client.println("HTTP/1.1 200 OK");
            client.println("Content-Type: text/html");
            client.println("Connection: close");  // the connection will be closed after
completion of the response
            //client.println("Refresh: 5");  // refresh the page automatically every 5 sec
            client.println();
              if(header.indexOf("GET / HTTP/1.1") >= 0) {
                client.println("<!DOCTYPE HTML>");
                client.println("<html>");
                client.println("index");
                client.println("</html>");
              } else {
                client.println("<!DOCTYPE HTML>");
                client.println("<html>");
                client.println("hello world!");
                client.println("</html>");
              }

          } else {

            // wrong user/pass
            //client.println("HTTP/1.0 401 Authorization Required");
            client.println("HTTP/1.1 401 Unauthorized");
            client.println("WWW-Authenticate: Basic realm=\"Secure\"");
            client.println("Content-Type: text/html");
            client.println();
```

```
          client.println("<html>Text to send if user hits Cancel button</html>"); // really
need this for the popup!

        }

        header = "";
        break;
      }
      if (c == '\n') {
        // you're starting a new line
        currentLineIsBlank = true;
      }
      else if (c != '\r') {
        // you've gotten a character on the current line
        currentLineIsBlank = false;
      }
    }
  }
  // give the web browser time to receive the data
  delay(1);
  // close the connection:
  client.stop();
  Serial.println("client disconnected");
 }
}

void printPage(EthernetClient client){

 client.println("<!DOCTYPE html>");
  client.println("<h1>Galileo na web!</h1>");
  client.println("<p>Página web para acender os leds da Galileo.</p>");
  client.println("<p>Selecione o led que deseja acender:</p>");
//   client.println("<input type=button value=LED1
onmousedown=location.href='http://143.107.235.59:8086/snapshot.cgi?user=usuário&p
wd=senha&next_url='>");
  client.println("<input type=button value=LED1
onmousedown=location.href='/?LED1;'>");
  client.println("<input type=button value=Buzzer
onmousedown=location.href='/?LED2;'>");
  client.println("<input type=button value=LED3
onmousedown=location.href='/?LED3;'>");
  client.println("<input type=button value=LED13
onmousedown=location.href='/?LED4;'>");
  client.println("<input type=button value=GAS
onmousedown=location.href='/?GAS;'>");
  client.println("<input type=button value=BAR
onmousedown=location.href='/?BAR;'>");
  client.println("<input type=button value=ACEL
onmousedown=location.href='/?ACEL;'>");
  client.println("<input type=button value=BOTH
onmousedown=location.href='/?BOTH;'>");
```

```
    client.println("</body>");
    client.println("</html>");
    client.println();

}
void sendHeader(EthernetClient client, char *title)
{
  // envia o http response header padrao
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html ; charset=UTF-8");
  client.println ("Connection: close");
  client.println();
  client.print("<html><head><title>");
  client.print(title);
  client.println("</title><body>");
}

void checkForClient(){
  EthernetClient client = server.available();
  int pin = 0;
  if (client) {

    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        //read char by char HTTP request
        while (readString.length() < 100) {
          char c = client.read();
          //store characters to string
          readString += c;
          //Serial.print(c);
          if(c == '\n' )
            break;
        }

          if(!sentHeader) {
              sendHeader(client,"Web Page Galileo");
              printPage(client);
              sentHeader = true;
           }

           //controle no led 1  (pino 2)
          if(readString.indexOf("LED1") >= 0 ){
            File myFile;
            system("touch /media/mmcblk0p1/test.txt");
            myFile = SD.open("test.txt",FILE_READ);

              // envia o http response header padrao
            digitalWrite(2, HIGH);
            Serial.println("Led 1 On");
```

```
            delay(3000);
            digitalWrite(2, LOW);
            readString = "";
            client.println("<br/>");
            if (myFile) {
 //   Serial.println("test.txt:");
              char aux[5];
              unsigned int i;
    // read from the file until there's nothing else in it:
 //             while (myFile.available()) {
              for(i = 0; i < 5; i++)
                aux[i] = myFile.read();
              aux[5] = aux[4];
              myFile.close();
              if (aux[3] == '0'){
                if (aux[4] == '1'){
//                    Serial.write("0");
//                    Serial.write("1");
                  client.println("<img src
='http://143.107.235.59:8085/videostream.cgi?user=usuário&pwd=senha&resolution=32
&rate=0'>");
//                  client.println("<img src
='http://143.107.235.59:8086/videostream.cgi?user=usuário&pwd=senha&resolution=32
&rate=0'>");
                 }
                 if (aux[4] == '0'){
//                    Serial.write("0");
//                    Serial.write("0");
                  client.println("<img src
='http://143.107.235.59:8086/videostream.cgi?user=usuário&pwd=senha&resolution=32
&rate=0'>");
                 }
               }
 //            }
    // close the file:
          }
        }

        //controle no led 2  (pino 3)
        if(readString.indexOf("LED2") >= 0 ){
          digitalWrite(3, HIGH);
          Serial.println("Led 2 On");
          delay(3000);
          digitalWrite(3, LOW);
          readString = "";
        }

        //controle no led 3  (pino 4)
        if(readString.indexOf("LED3") >= 0 ){
          digitalWrite(4, HIGH);
          Serial.println("Led 3 On");
```

```
          delay(3000);
          digitalWrite(4, LOW);
          readString = "";
        }

        //controle no led 13  (pino 13)
        if(readString.indexOf("LED4") >= 0 ){
          digitalWrite(13, HIGH);
          Serial.println("Led 13 On");
          delay(3000);
          digitalWrite(13, LOW);
          readString = "";
          client.println("<br/>");
        }

        //Recebimento de comando pelo sistema de detecção de vazamento de gás
        if(readString.indexOf("GAS") >= 0 ){
          system("./gas.sh");
          readString = "";
          client.println("<br/>");
        }

        //Recebimento de comando pelo sistema de detecção de variação da velocidade
        if(readString.indexOf("ACEL") >= 0 ){
          system("./acelerometro.sh");
          readString = "";
          client.println("<br/>");
        }

        //Recebimento de comando pelo sistema de detecção de variação de altura
(altitude)
        if(readString.indexOf("BAR") >= 0 ){
          system("./barometro.sh");
          readString = "";
          client.println("<br/>");
        }

        //Recebimento de comando pelo sistema de detecção de variação da velocidade
e altitude
        if(readString.indexOf("BOTH") >= 0 ){
          system("./acel_and_bar.sh");
          readString = "";
          client.println("<br/>");
        }

       if (c == '\n' && currentLineIsBlank)
         break;

       if (c == '\n') {
         currentLineIsBlank = true;
       }
```

```
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }

    readString="";
    sentHeader = false;
    delay(1); // tempo para o browser receber os dados
    client.stop(); // fecha a conexão
  }
}
```

Algoritmo A - 15: Algoritmo completo implementado na placa Intel Galileo, incluindo comunicação entre o processo Arduino e Linux

Anexo

A Figura 53 apresenta o diagrama esquemático do conversor de tensão. Para aplicações unidirecionais o sistema funciona como um divisor resistivo, isto é, com os resistores devidamente calculados pode-se obter a tensão de 3,3V a partir de 5V. Para aplicações bidirecionais, utiliza-se um transistor (BSS138).

Se a conexão TX_LV apresentar nível lógico alto (3V3), o transistor entra em corte, pois seu V_{GS} tende a 0. Neste caso, a saída TX_LH apresentará o mesmo nível de HV, isto é, de 5V.

Se a conexão TX_LV apresentar nível lógico baixo (0V), o transistor apresenta V_{GS} suficiente para conduzir e TX_LH passa a nível lógico baixo.

De forma análoga, se TX_HV apresentar nível lógico alto (3V3), o diodo intrínseco ao transistor não conduz, pois apresenta-se reversamente polarizado. Neste caso, a saída TX_LV apresentará o mesmo nível de LV, isto é, de 3V3.

Se a conexão TX_HV apresentar nível lógico baixo (0V), o diodo intrínseco ao transistor entrará em condução e a saída TX_LV passa a nível lógico baixo.

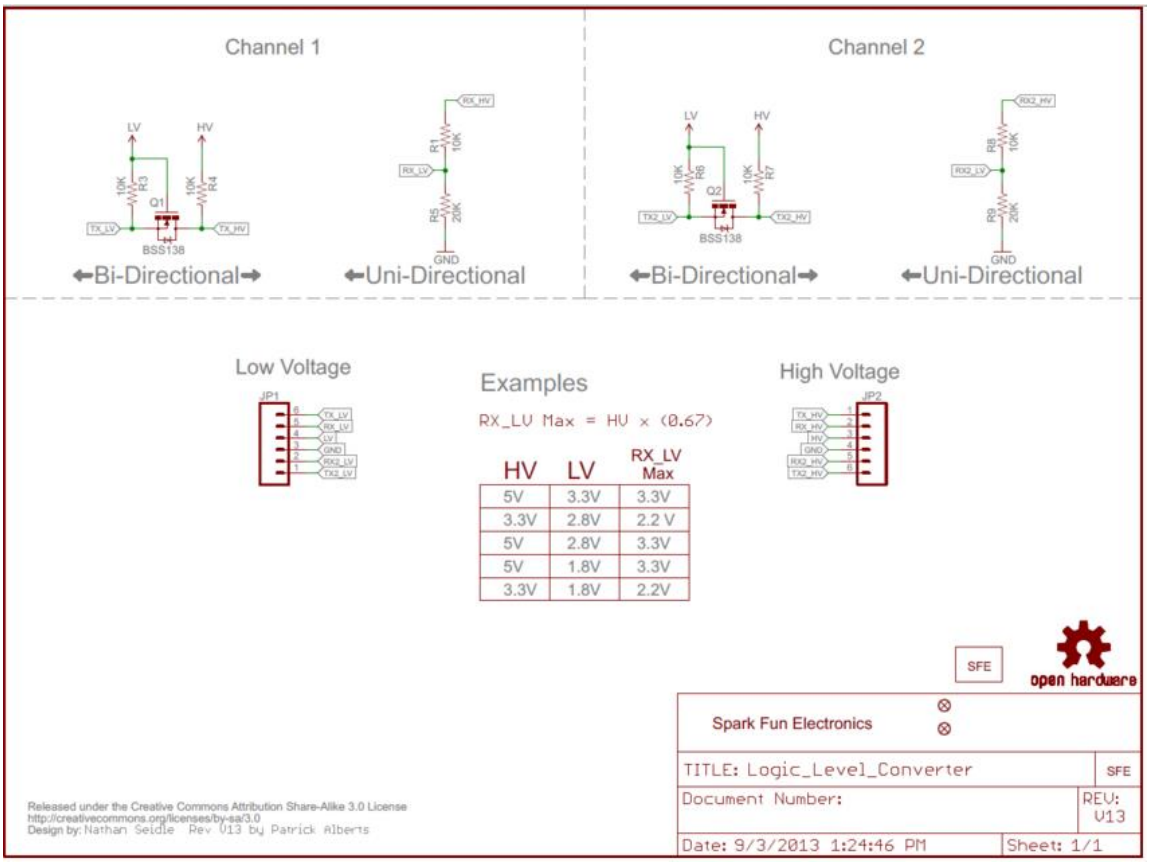


Figura 53: Esquemático do circuito conversor de tensão. Fonte: https://cdn.sparkfun.com/assets/b/0/e/1/0/522637c6757b7f2b228b4568.png