
# TRABALHO DE GRADUAÇÃO III

## ENGENHARIA DE INFORMAÇÃO

**Detector de queda com envio de notificação e dados**

**Aluno: Edson Cassius Duarte Lopes**

**Orientador: Prof. Dr. João Henrique Kleinschmidt**

**Santo André, 07 de dezembro de 2016.**

## Sumário

## 1. Introdução

O desenvolvimento de ferramentas para auxiliar em tarefas do dia-a-dia tem sido feito há muito tempo pela humanidade. Desde a invenção da roda, machado, entre outras ferramentas, o homem sempre tentou encontrar soluções para diversos tipos de problemas.

Com a invenção da eletricidade, novas ferramentas ainda mais poderosas foram aparecendo e a evolução da tecnologia ao longo dos últimos anos ampliou o campo para desenvolvimento de novas aplicações, tanto em grande escala no setor empresarial quanto para pequenas ferramentas que podem ser implementadas, facilitando o cotidiano das pessoas ou muitas vezes até salvando vidas.

O termo *Internet of Things* (IoT) ou Internet das Coisas, vem se tornando cada vez mais conhecido e é um tema de pesquisa e nicho de mercado com crescente importância atualmente [1].

A computação durante muito tempo evoluiu baseando-se na utilização do computador pessoal, entretanto, a próxima onda na era da computação estará fora do *desktop* tradicional. No paradigma da Internet das Coisas (IoT), muitos dos objetos que nos rodeiam estarão na rede de uma forma ou de outra. [2] .

Essa inovação será possibilitada pela incorporação de eletrônicos nos objetos físicos do dia-a-dia, tornando-os "inteligentes" e permitindo que eles se integrem perfeitamente dentro da infraestrutura cibernética global. Isto dará lugar a novas oportunidades para o setor das Tecnologias de Informação e Comunicação, abrindo o caminho para novos serviços e aplicações capazes de potencializar a interconexão de domínios físicos e virtuais. [3]

A utilização de objetos conectados à rede pode auxiliar no monitoramento de qualquer tipo de sistema em tempo real. A indústria está cada vez mais aderindo a este tipo de tecnologia, mas a utilização de aparelhos deste tipo dentro de casa traz um grande número de possibilidades de implementação.

Por exemplo, auxiliar na qualidade de vida e segurança das pessoas é um dos possíveis campos de aplicação, no qual o monitoramento de seus

hábitos e de eventuais problemas no dia-a-dia é de extrema importância. Através da análise de informações em tempo real, pode-se ter um sistema que solicita ajuda se detectar alguma atividade incomum, como uma eventual queda, que pode ser fatal para uma pessoa de idade ou debilitada.

Quedas são um grande risco para a saúde dos idosos e um grande obstáculo à vida independente. A incidência estimada de quedas tanto para pessoas institucionalizadas quanto para pessoas independentes acima de 75 anos é de pelo menos 30% ao ano. [4]

Para um monitoramento efetivo de quedas, por exemplo, é necessária a utilização de algum dispositivo para captura e análise das informações que podem indicar essa possível queda. Pode ser utilizado um sensor para obtenção dos dados e um microcontrolador para análise.

Este projeto pretende auxiliar na detecção de possíveis quedas de idosos e pacientes, utilizando de um módulo Wi-Fi para enviar notificações para um aparelho externo. Também são capturadas informações que são armazenados em Nuvem.

Um dos principais objetivos deste projeto é desenvolver uma ferramenta que promova segurança através um bom desempenho na detecção de quedas, diminuindo o número de possíveis erros. Outros objetivos envolvem o envio de diversas notificações para aparelhos externos e e-mails; o sucesso na captura e armazenamento de dados para serem utilizados posteriormente. O projeto foi desenvolvido utilizando a plataforma Arduino que permite a integração com diversos módulos e sensores, auxiliando a atingir os objetivos do mesmo.

Entre os objetivos também está o desenvolvimento de dois algoritmos de queda para uma análise estatística de qual algoritmo desempenha melhor a atividade proposta, levando em consideração para os testes casos de queda reais e casos em que não ocorre queda (a fim de evitar falsos positivos).

A plataforma Arduino é baseada em um microcontrolador central e possui uma linguagem de programação padrão baseada em C/C++. Com esta é possível programar e executar algumas funções utilizando diversos acessórios eletrônicos, como leds, sensores, botões, módulos, entre outros.

As placas dispõem de pinos digitais de entrada e saída, assim como algumas entradas analógicas. A placa Arduino UNO DIP possui o

microcontrolador ATmega328P removível, possibilitando a otimização de projetos.

A estrutura deste trabalho contará com uma fundamentação teórica a respeito dos temas no capítulo 2, seguido de uma descrição detalhada de todos os materiais, implementação do circuito para captura de movimentos e conexão WiFi e o desenvolvimento dos algoritmos de queda no capítulo 3. No capítulo 4 são mostrados os resultados para testes de queda realizados e comparação entre os algoritmos. A conclusão encontra-se no capítulo 5 com algumas considerações do desempenho geral do protótipo e possíveis melhorias.

## 2. Fundamentação Teórica

### 2.1. IoT

O termo Internet das Coisas (*Internet of Things – IoT)* foi primeiramente utilizado por Kevin Ashton em 1999 no contexto de gestão de *supply chain*. Entretanto, na última década essa definição foi mais inclusiva, cobrindo grande variedade de aplicações como cuidados com a saúde, transporte, utilitários, etc. E embora a definição de "coisas" tenha mudado à medida que a tecnologia evoluiu, o principal objetivo de fazer um computador obter informação sem o auxílio de intervenção humana permanece o mesmo. [2]

Diversos dispositivos e aplicações voltadas para o desenvolvimento de ambientes inteligentes tem sido implementadas recentemente. O conceito de casa inteligente com interruptores, lâmpadas e termostatos conectados já está mais difundido em mercados como dos Estados Unidos e Europa. Aplicações na área da saúde estão sendo desenvolvidas principalmente na área de sensoriamento em tempo real de pacientes, identificação e autenticação de pessoas e coleta e armazenamento de dados automaticamente através de sensores. [1]

A adição de diversas tecnologias de comunicação a esses objetos como Bluetooth, ZigBee e Wi-Fi não só proporciona aos mesmos capacidade de se comunicar, como também abre possibilidades de agregar um certo nível de inteligência a esses dispositivos, o que abre um grande leque de novos serviços que podem surgir a partir dessa conectividade. [1]

Há por exemplo, a possibilidade de se utilizar de um aplicativo de *smartphone* para receber notificações de um dispositivo ou até mesmo utilizá-lo para algum tipo de controle. Um exemplo de dispositivo que foi desenvolvido e já está no mercado é a Smart Lamp desenvolvida pela empresa sul-coreana LG, que possibilita o controle da lâmpada através de um aplicativo de celular. O produto é indicado na Figura 1.

Figura 1: Smart Lamp da LG.

A lâmpada é encaixada na base que contém os módulos de bluetooth, Wi-Fi, e outros componentes para exercer algumas funções propostas, como controle de luminosidade ou programar a lâmpada para acender em um determinado momento.

### 2.2. Detecção de queda

A maioria das quedas em idosos acontecem durante atividades do dia-a-dia que envolvem uma pequena perda de equilíbrio, como ficar em pé ou estar andando. Uma menor quantidade de quedas acontece quando a atividade envolve um movimento mais significativo, como sentar-se em uma cadeira ou subir a escada. Para pessoas acima de 65 anos, quedas acidentais são as principais causas de admissão dessas pessoas em hospitais e a sexta em causa de mortes. Para pessoas entre 65 e 75 anos, são a segunda principal causa de morte e para pessoas acima de 75 anos a principal. [4]

O monitoramento de uma queda deve ser baseado em diversos fatores e cenários, pois existem diferentes tipos de quedas e a não consideração de um desses pode ser crucial para com a vida do paciente.

Para avaliar a qualidade de um sistema de detecção de queda, define-se na Tabela I as possíveis respostas do sistema.

Tabela 1: Possíveis respostas de um sistema de detecção de quedas.

| | Queda ocorre | Queda não ocorre |
|---|---|---|
| Queda é detectada | *True* Positivo (TP) | Falso Positivo (FP) |
| Queda não é detectada | Falso Negativo (FN) | *True* Negativo (TN) |

Três índices principais podem ser definidos a partir das possibilidades:

- Sensibilidade, sendo a capacidade do sistema de detectar a queda;

$$\text{Sensibilidade} = \frac{TP}{TP + FN} \quad (1)$$

- Especificidade, sendo a capacidade do sistema de evitar falsos positivos;

$$\text{Especificidade} = \frac{TN}{TN + FP} \quad (2)$$

- Precisão, sendo a habilidade do sistema de distinguir e detectar movimentos que levam a uma queda e movimentos em que não ocorre queda.

$$\text{Precisão} = \frac{TP + TN}{P + N} \quad (3)$$

Sendo P e N o número de movimentos em que ocorrem queda e o número de movimentos que não ocorre queda. [4]

Dependendo dos valores obtidos com os cálculos realizados para cada índice, pode-se avaliar diferentes aspectos do sistema de detecção de queda implementado para aferir acerca da efetividade do mesmo, e também compará-lo com outros sistemas desenvolvidos já presentes na literatura.

Para detecção de um possível movimento de queda, dois sensores são comumente utilizados:

- Acelerômetro, para medir a aceleração;
- Giroscópio, para medir a velocidade angular. [4]

Integrando os sensores é possível detectar o movimento realizado por uma pessoa em três eixos e a intensidade do movimento, auxiliando na detecção efetiva de uma queda.

Através dos dados de aceleração nos 3 eixos obtidos do acelerômetro, é possível calcular o módulo da aceleração, sendo melhor a utilização dessa magnitude para monitoramento ao invés de monitorar cada eixo separadamente.

$$A = \sqrt{a_x^2 + a_y^2 + a_z^2} \qquad (4)$$

O módulo da velocidade angular também pode ser calculado, assim como a mudança de orientação baseada em um vetor de aceleração de referência (P) para o sensor, definido a partir de sua posição [13].

O produto escalar entre os dois módulos pode ser obtido, e isolando o ângulo θ tem-se a mudança de orientação, definida na equação 5.

$$\theta = cos^{-1}\left(\frac{A*P}{|A||P|}\right) \qquad (5)$$

Apesar de depender primordialmente do algoritmo utilizado, a posição dos sensores também é importante para o monitoramento de uma queda, no qual esta posição pode muitas vezes dificultar a detecção.

O pulso não é recomendado como uma boa posição, uma vez que o mesmo está sujeito a muitos movimentos de aceleração alta que aumentaria o número de falsos positivos. A cintura é uma posição mais aceitável do ponto de vista de quem está utilizando o sensor, uma vez que esta opção pode se encaixar em um cinto e está mais próxima do centro de gravidade do corpo. Outras posições já foram também selecionadas por pesquisadores, como a axila, a coxa ou o tronco, citando suas próprias vantagens e desvantagens. [4]

No geral, sistemas de detecção de queda podem ser utilizados para diversas utilidades. Estudos relacionados a essas áreas têm sido tema de pesquisa há alguns anos. Em 2004, Sixsmith e Johnson [5], desenvolveram um projeto para monitoramento de queda em idosos utilizando uma rede de sensores infravermelhos integrados, montados nas paredes de uma casa. Com

o aparecimento de novas tecnologias, ampliou-se ainda mais as possibilidades de monitoramento.

Em 2012, por Suryadevara e Mukhopadhyay, foi criado e desenvolvido um projeto para monitoramento inteligente de uma casa, baseado em uma rede de sensores sem fio ZigBee, com as principais funções de monitorar e avaliar o bem-estar dos idosos vivendo sozinhos em um ambiente doméstico. Os sensores captavam dados diários e poderiam ser utilizados para prever situações não seguras durante o monitoramento de atividades regulares. [6]

### 2.3. Produtos disponíveis no mercado

Aparelhos para auxiliar no monitoramento de idosos e pacientes estão presentes no mercado. Os serviços prestados vão desde o monitoramento em tempo real de atividades cardíacas ou corporais ao monitoramento de localização e solicitação de ajuda.

O sistema *BeClose* é um sistema de monitoramento sem fio especialmente projetado para o ambiente domiciliar, que permite o monitoramento remoto de qualquer cômodo, porta, entre outros. Os sensores se conectam a uma central principal de monitoramento que exibe os dados de forma simplificada para uma eventual consulta. O sistema ainda permite enviar notificações para aparelhos e pode ser programado para enviar uma notificação caso uma porta abra em horário incomum, por exemplo.

Já o produto *BodyGuardian Heart* permite o monitoramento em tempo real através de um pequeno módulo instalado no peito de pacientes. Os dados capturados são de algumas funções corporais, como batimento cardíaco, taxa de respiração, entre outras. O aparelho utiliza tecnologia *wireless* para efetuar o envio dos dados diretamente para uma central de apoio. Apesar do fornecedor não dar detalhes sobre a maneira com que o dispositivo efetua sua conexão, sabe-se que com um aplicativo próprio, é possível visualizar os dados através de alguns dispositivos e pela web. O aplicativo disponibiliza os dados capturados por 30 dias. Na Figura 2, pode-se visualizar o dispositivo desenvolvido e um *smartphone* com o aplicativo.


Figura 2: BodyGuardian Heart da empresa Preventice.

Detectores de queda também estão disponíveis no mercado atualmente. Grande parte requer um contrato mensal para monitoramento e dispõem de uma base para receber e enviar informações do monitoramento e um pequeno módulo que é colocado no corpo do paciente.

Um detector mais bem avaliado por uma análise feita recentemente [7], é o da empresa Medical Guardian, que dispõe de uma base de longo alcance juntamente com um pequeno módulo que deve ser utilizado em forma de colar ou no cinto pela pessoa a qual se deseja monitorar, formado principalmente por sensores que permitem detectar quedas. Ao detectar-se uma queda um alerta é enviado à base fixa, que pode levar alguns segundos para decidir a eventual ocorrência de uma queda ou não, enviando em caso positivo um alerta para a central que toma providências a respeito.

Foi estabelecido um ranking baseado em testes de queda para testar a precisão e a sensibilidade de cada dispositivo que é comercializado no formato base fixa e colar para detecção de quedas. Os três que possuíram melhores desempenhos tiveram médias de detecção de quedas variando entre 75% e 80%. [7]

No Brasil atualmente temos alguns produtos semelhantes sendo oferecidos pela empresa Tecnosenior. São oferecidos acessórios como botão de emergência, sensores de queda e presença, além de um dispensador eletrônico de medicamentos. Para utilização dos acessórios é necessário uma base fixa que se comunica com uma central de atendimento, como um sistema

de emergência pessoal, e se comunica com os acessórios via tecnologia sem fio. O sensor de queda, assim como de algumas empresas americanas é oferecido no formato de pingente e se comunica também com o sistema de emergência pessoal.

## 3. Metodologia

Um protótipo para detecção de queda foi desenvolvido. A placa Arduino UNO DIP foi utilizada para elaboração do mesmo, juntamente com sensores que auxiliam nas tarefas propostas.

São exibidos os materiais utilizados para montagem do protótipo, assim como a implementação do circuito final, o desenvolvimento dos algoritmos de quedas e o envio de notificações e dados.

### 3.1. Materiais

A lista dos materiais que foram utilizados no desenvolvimento do protótipo é dada abaixo.

- Arduino UNO DIP
- Módulo Wi-Fi ESP-01
- Placa GY-521 com acelerômetro e giroscópio integrado.
- Protoboard
- Regulador de Tensão Linear AMS1117
- Jumpers

Na Figura 3, pode-se visualizar a placa Arduino UNO DIP. Pode-se observar o microcontrolador e as entradas analógicas e digitais fornecidas pela placa.


Figura 3: Arduino UNO DIP.

O Arduino UNO DIP possui um microcontrolador ATmega328 removível e sua tensão de operação é de 5V. Algumas outras especificações técnicas são apresentadas:

- Tensão de entrada: 7-12 V
- Entradas e saídas digitais: 20 (of which 6 provide PWM output)
- Entradas analógicas: 6
- Corrente contínua fornecida em 3.3V: 50 mA
- Memória Flash: 32 KB (ATmega328) of which 0.5 KB used by bootloader
- Velocidade do clock: 16 MHz

Conectou-se um módulo na placa Arduino para conexão em redes sem fio (Wi-Fi). Para esta função foi utilizado o módulo Wi-Fi ESP8266. É possível programar o módulo para conectar na rede e habilitar a transferência de dados, possibilitando o envio de dados adquiridos no protótipo à rede.

O módulo Wi-Fi ESP8266 opera em três funções: Access Point (AP) e Station (STA) e ambos. No modo STA, o mesmo é conectado em um AP externo e pode enviar dados para a Web. Pode ser conectado com o microcontrolador ATmega328 por comunicação serial para operação em conjunto.

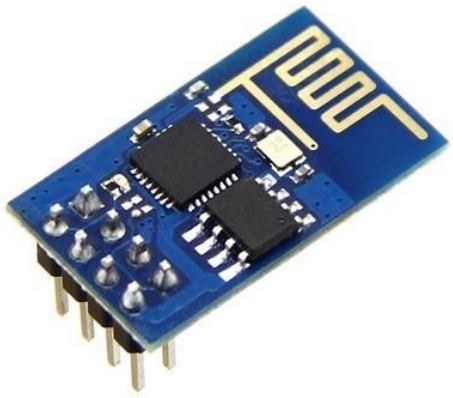


Figura 4: Módulo Wi-Fi ESP8266, modelo ESP-01.

O módulo opera dentro dos canais IEEE 802.11 b/g/n e algumas especificações técnicas são listadas abaixo.

- Máxima taxa de operação serial: 115200 bps
- Tensão de entrada: 3.3V
- Tensão limite para as entradas digitais: 3.6V
- Proteção: WPA, WPA2

Na Tabela 2 é possível visualizar a quantidade de corrente necessária para cada modo de operação do módulo. Em todos os casos a quantidade de corrente utilizada supera os 50mA fornecido pela tensão 3.3V do Arduino UNO, sendo necessária a utilização de um regulador de tensão para esse dispositivo, fornecendo uma maior quantidade de corrente para uma melhor operação do mesmo.

Tabela 2: Corrente utilizada pelo módulo. [8]

| Modo | Corrente | Unidade |
|---|---|---|
| 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 |

É necessária a utilização do acelerômetro e do giroscópio para captura dos dados de aceleração e orientação. Utiliza-se para este fim a placa GY-521, que possui acelerômetro e giroscópio integrados conectados nas entradas da placa.

A placa GY-521 é baseada no CI MPU-6050 e possui um acelerômetro e um giroscópio de alta precisão. É possível a utilização dos três eixos do acelerômetro e dos três eixos do giroscópio para monitoramento dos valores

que podem indicar uma possível queda. O CI também faz o processamento do algoritmo de detecção de movimento, não ocupando o microcontrolador com esta tarefa.

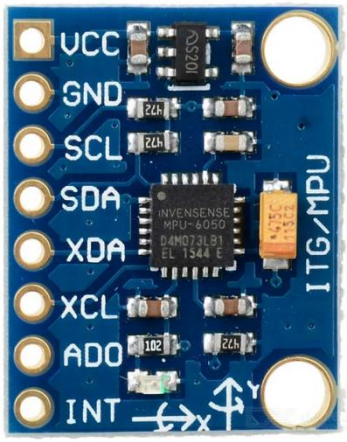



Figura 5: Placa GY-521 com acelerômetro e giroscópio.

Algumas características técnicas obtidas através do fabricante são listadas nos tópicos abaixo.

- Valores de alcance do acelerômetro: ±2, ±4, ±8, ±16g
- Valores de alcance do giroscópio: ± 250, 500, 1000, 2000 °/s
- Alimentação: 3.3V - 5V

O módulo possui um regulador de tensão embutido para ser alimentado na faixa de tensão de operação.

Uma protoboard é utilizada para montagem do protótipo, juntamente com diversos jumpers para as conexões entre os módulos e o microcontrolador. Na montagem do circuito utiliza-se um botão para cancelar uma queda que não ocorreu (falso positivo).

Um regulador de tensão é necessário para alimentar o módulo ESP-01, pois a saída 3.3V fornecendo apenas 50mA não promove um funcionamento ideal do módulo. É utilizado então o regulador de tensão linear AMS1117, convertendo a saída de 5V para 3.3V que fornece até 800mA.

### 3.2. Implementação do Circuito

Os módulos foram conectados no microcontrolador ATmega328 através do Arduino UNO DIP. O botão utiliza o resistor de pull-up interno disponibilizado pelo Arduino e o circuito utilizado pelo algoritmo desenvolvido para captação da queda e envio de notificação e dados é apresentado na Figura 6.

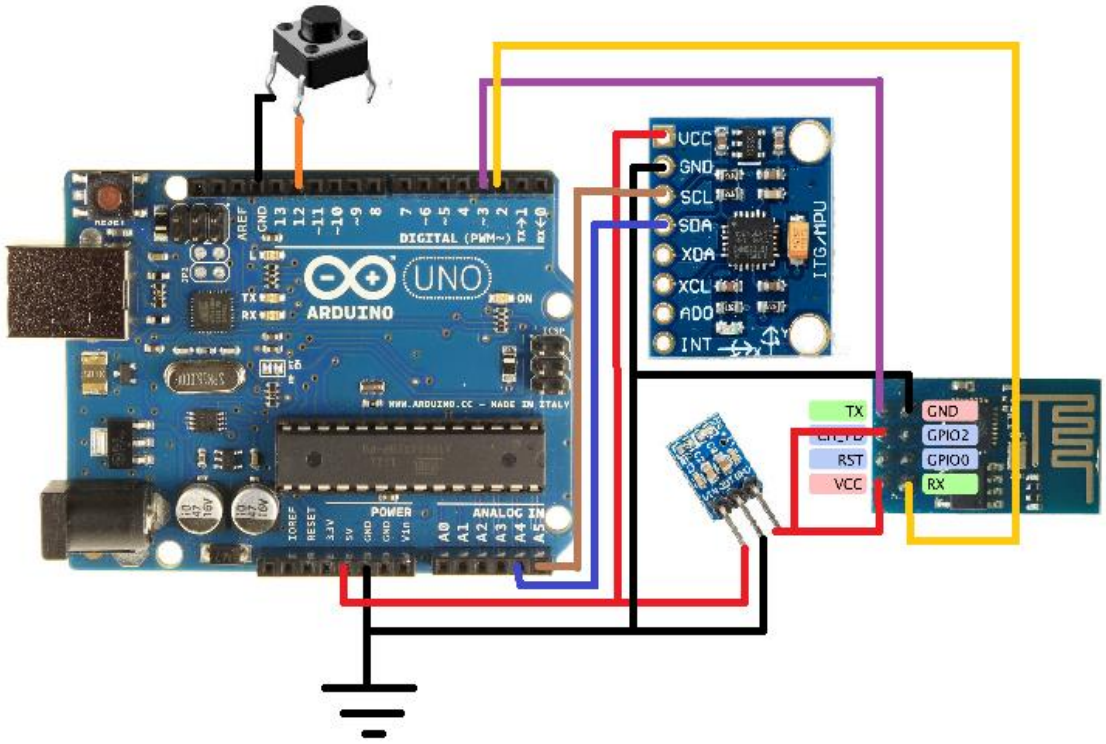



Figura 6: Circuito do algoritmo de detecção de queda.

Foi montado um protótipo para testes em uma protoboard com o circuito da Figura 6. Este protótipo foi utilizado para testes na região da cintura, e o mesmo pode ser visualizado na Figura 7.

Foram feitos testes para adicionar um módulo de monitoramento de atividade cardíaca que necessita ser colocado na ponta do dedo. Devido ao posicionamento do protótipo no usuário ser na região da cintura decidiu-se não utilizá-lo.

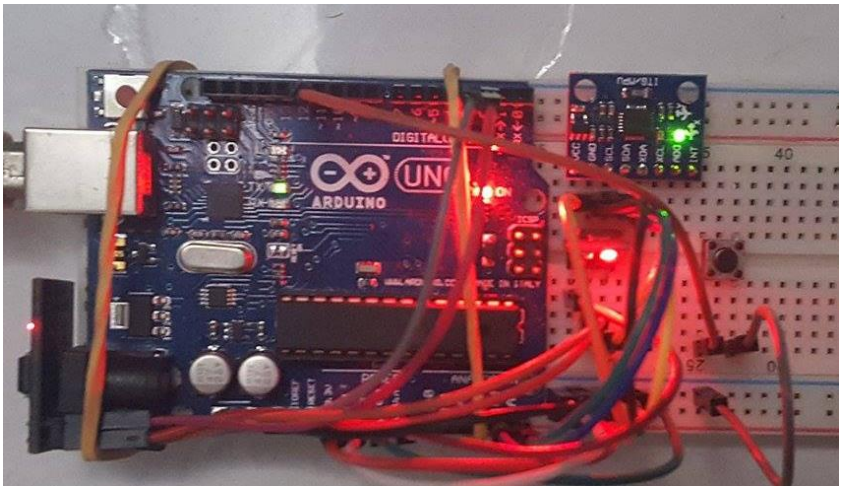



Figura 7: Circuito montado na Protoboard.

### 3.3. Algoritmos para identificação de queda

Com o módulo GY-521 é possível obter os valores de aceleração em cada eixo e a velocidade angular também de cada eixo. A comunicação se dá por meio da interface *I2C*, através dos pinos SCL e SDA, e a leitura é feita por meio da biblioteca *Wire*, sendo realizada pelo microcontrolador recebendo os dados em pinos analógicos.

É necessário que seja realizada a calibração desses valores pois cada módulo apresenta um valor de offset diferente. O módulo deve ser colocado em cada eixo, variando sua posição e anotando-se os valores de mínimo e máximo para cada valor do acelerômetro e do giroscópio, para assim calcular a diferença entre as medidas e estabelecer o offset para cada variável. Um script disponibilizado na biblioteca *I2Cdev* [9] foi utilizado para obter os valores de calibração, e esses valores foram implementados no código para cada respectivo valor. Após a aplicação do offset nas variáveis, as mesmas são divididas pelo fator de escala de sensibilidade de cada sensor, especificados no *datasheet* [10].

Para o monitoramento de queda a literatura recomenda a utilização do módulo da aceleração. [6,11,12,13] Nos algoritmos, a cada frame da função *loop()* é efetuado o cálculo do módulo da aceleração nos 3 eixos e também o cálculo do módulo da velocidade angular nos 3 eixos, assim como o ângulo da mudança de orientação.

Foram desenvolvidos 2 algoritmos independentes para o monitoramento de queda, que utilizam parâmetros diferentes para avaliar a cada frame a chance de uma possível queda.

### 3.3.1. Algoritmo I

Para o primeiro algoritmo, utilizou-se a referência de que quando ocorre uma queda, observa-se que primeiramente o módulo da aceleração tem seu valor reduzido significativamente, seguido de um grande aumento por um breve período de tempo e também uma mudança de orientação [11], detectada por meio do módulo da velocidade angular em relação a uma posição de referência para o protótipo.

Com base nestas informações, foi possível desenvolver um fluxograma (Figura 8) para indicar as decisões necessárias para se obter uma queda. Com a intenção de diminuir o número de falsos positivos, indica-se que o aparelho detectou uma queda através de um LED piscando; o usuário pode então apertar um botão para indicar que não precisa de socorro.

Se o botão não for apertado dentro de 10 segundos após a indicação de queda, uma notificação é enviada através do módulo ESP-01 e recebida no celular, além de armazenar as informações referentes em uma planilha de dados em nuvem.

Utiliza-se de variáveis binárias para controle de cada um dos estados gerados pelas condicionais. Através de contadores é possível contar o intervalo de tempo que o programa permaneceu em cada estado. O código desenvolvido encontra-se disponível no Anexo A.

Para o limiar inferior para o módulo da aceleração que é um *trigger* da queda, utilizou-se os valores de 0,6g respectivamente. O valor de 0,6g é um pouco abaixo do valor normal para o módulo (1,0g), representando uma possível mudança brusca no sensor. Para este valor preferiu-se não utilizar um valor muito baixo pois é apenas a primeira validação do algoritmo.

No valor do limiar superior do módulo da aceleração, optou-se pela utilização do valor de 2,0g, pois o valor do módulo de aceleração médio de uma queda qualquer para uma pessoa é de 2,2g [12]. Após alguns testes de

situações reais de queda, o algoritmo respondeu muito bem para os valores definidos.

A condicional para a verificação do ângulo da mudança de orientação utilizada foi para valores maiores que 1,1 radianos, pois a partir desse valor a variação da posição do sensor é significativa, indicando que possivelmente esta pessoa está fora das posições normais em pé ou sentada, ou seja, fora da posição de referência do protótipo.

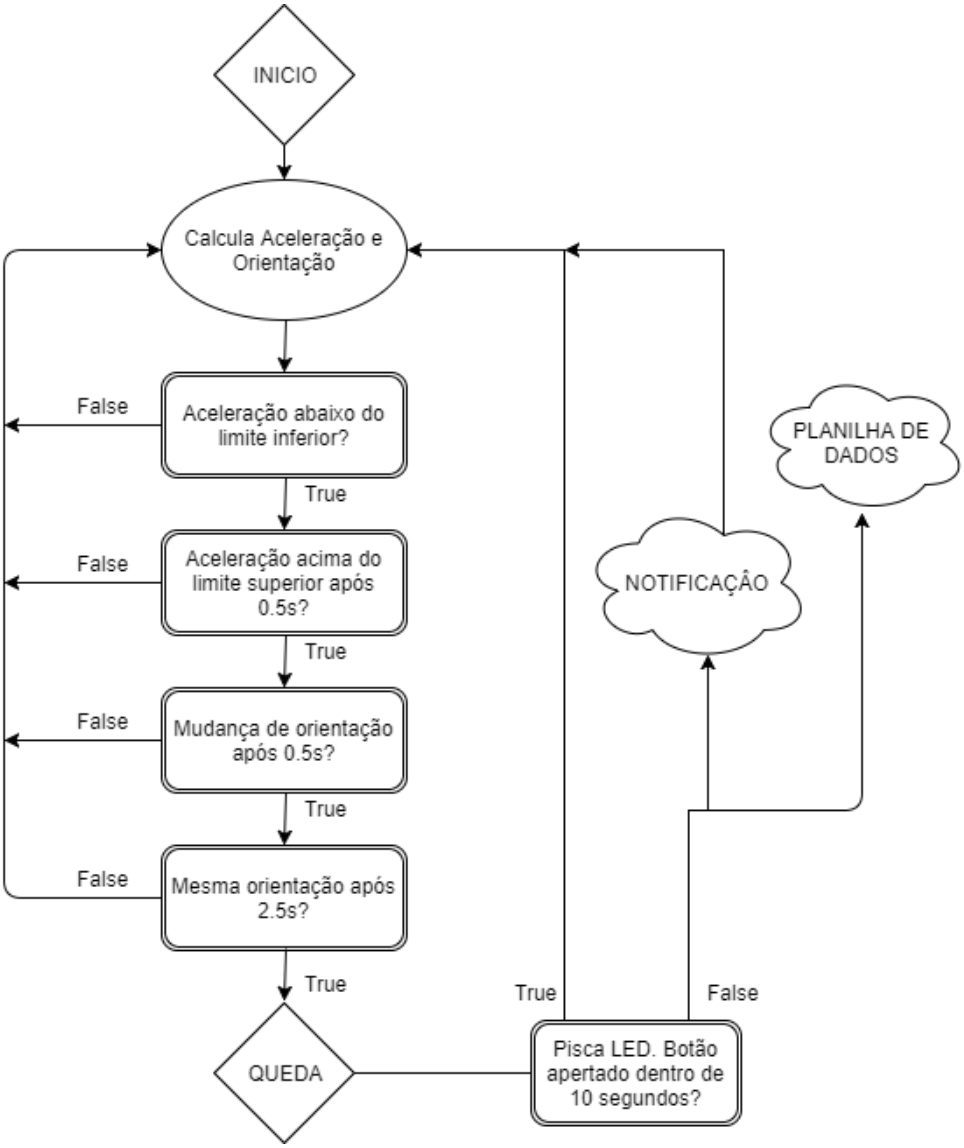



Figura 8: Fluxograma do algoritmo desenvolvido.

### 3.3.2. Algoritmo II

` Este outro método de detecção utilizado, é também baseado em um limiar superior para o módulo de aceleração, sendo essa a primeira condicional do algoritmo para a ativação da respectiva flag.

O algoritmo então verifica se a aceleração está próxima ao de uma pessoa parada. Isto é feito pois se a esta condicional for verdadeira, significa que a pessoa pode estar caída no chão [11]. A terceira verificação realizada, é se o acelerômetro está de volta na posição de referência, utilizando o valor o ângulo da mudança de orientação como limite.

Esta última condicional, como também utilizada no algoritmo anterior, verifica se a pessoa está caída de fato no chão, com um ângulo de referência diferente do que ela teria se estivesse em pé ou sentada, com o sensor na posição de referência.

São utilizadas flags para controle das condicionais, e o valor limiar superior para o módulo da aceleração utilizado foi de 2,0g, novamente confiando em um valor seguro próximo aos 2,2g [12].

Considerou-se como aceleração normal, valores próximos de 1,0g, com uma pequena margem de erro, resultando em uma verificação para valores entre 0,7g e 1,3g. Novamente o ângulo da mudança de orientação utilizado foi de 1,1 radianos, sendo a verificação realizada para quaisquer ângulos maiores que este.

Na Figura 9 é possível visualizar um fluxograma demonstrando o funcionamento do algoritmo. O código desenvolvido para este algoritmo pode ser visualizado no anexo A.

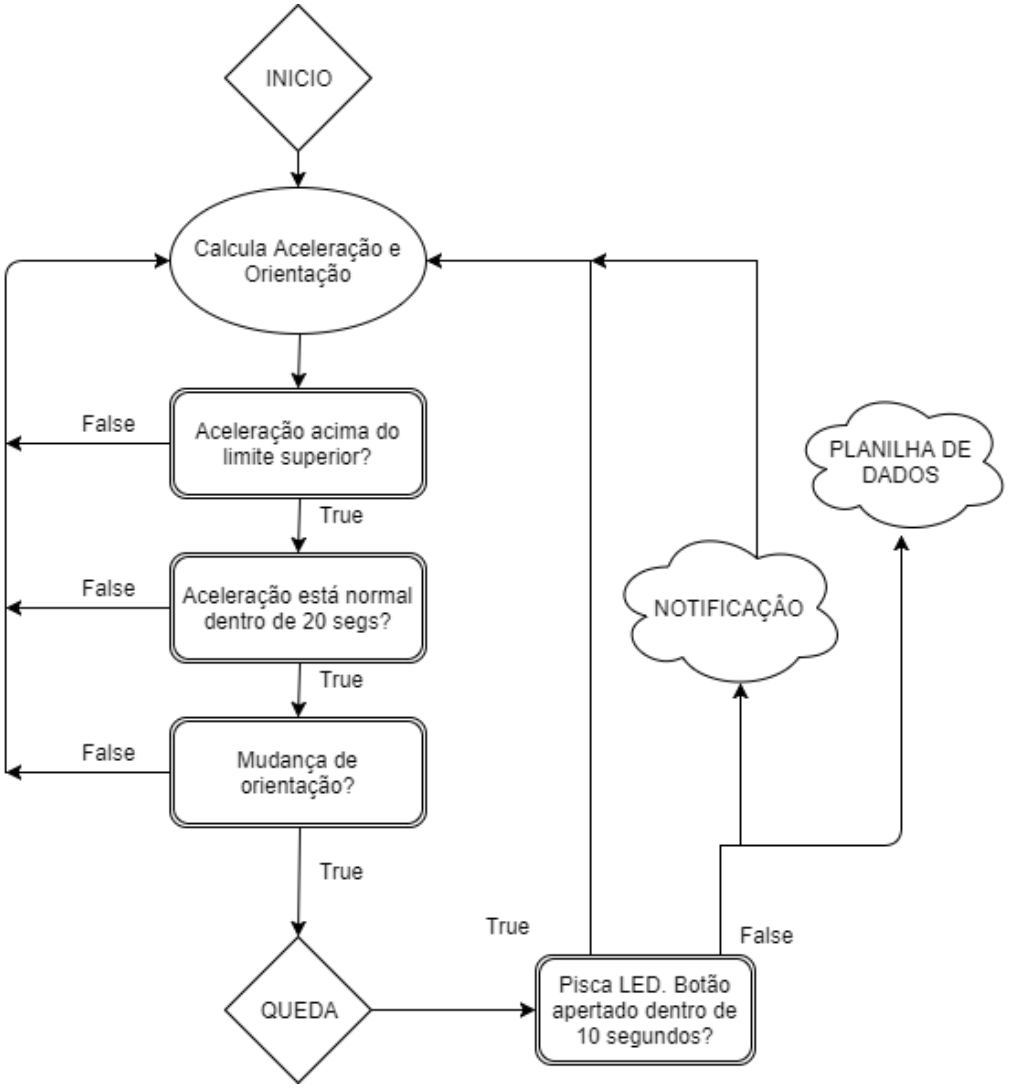



Figura 9: Fluxograma do Algoritmo II.

### 3.4. Notificação e envio de dados

Para utilizar o ESP-01 é preciso conectá-lo a uma entrada serial. É possível usar de um conversor USB-Serial para enviar comandos diretamente do computador, ou então utilizar da biblioteca *SoftwareSerial* para enviar comandos para o módulo através do microcontrolador.

Utiliza-se assim do Arduino UNO DIP com essa biblioteca para enviar comandos de configuração do ESP-01. O fabricante disponibiliza um manual com instruções para cada comando que é possível executar no módulo [14]. Funções como AT+RST para reiniciar e AT+CIOBAUD para definir a taxa de comunicação são exemplos de comandos que o módulo interpreta. Através do comando AT+CWJAP é possível conectar em uma rede.

Para receber notificações, utilizou-se do aplicativo *PushBullet*, disponível para iOS, Android e alguns navegadores. É possível obter um *token* através do aplicativo e utilizá-lo para receber notificações do *PushingBox* [15]. Para enviar uma notificação através do ESP-01, é utilizada uma requisição HTTP para o endereço fornecido pela nuvem *PushingBox*, que foi projetado para facilitar essa integração através de um API único. Na Figura 10 é possível visualizar alguns comandos executados no ESP-01, juntamente com a resposta dada pelo dispositivo e o encerramento com sucesso da requisição HTTP.

Após o envio da requisição é possível visualizar a notificação no dispositivo instantaneamente. Uma conexão com a Internet e ter o aplicativo instalado são necessários para o recebimento de tais notificações. Na Figura 11 pode-se observar como a notificação é recebida.

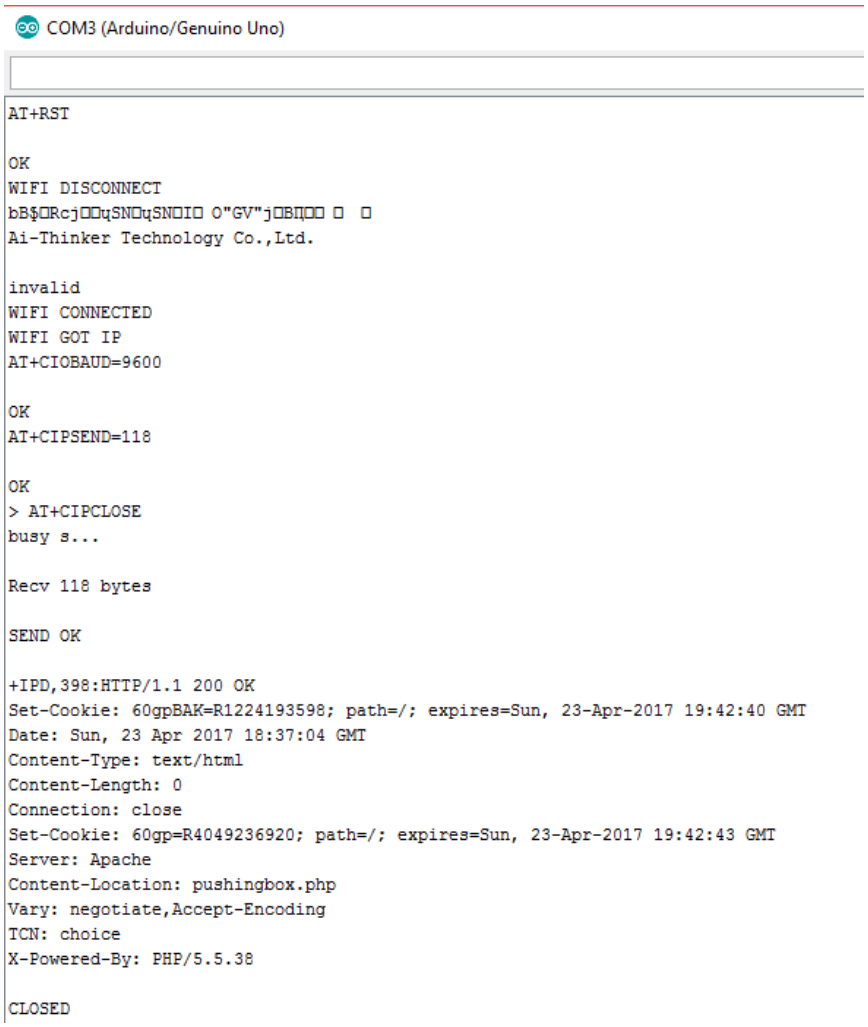



Figura 10: Comunicação serial com o módulo e resposta HTTP.

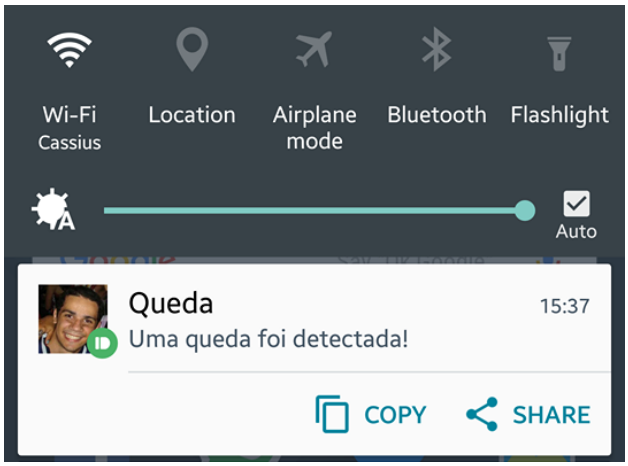



Figura 11: Notificação recebida através do aplicativo *PushBullet*.

Todas as notificações podem ser recebidas por familiares ou uma possível central de monitoramento, alertando pessoas que podem solicitar algum tipo de socorro ou tentar entrar em contato com a vítima. Além de receber na forma de notificação, pode-se enviar um e-mail ou alertar qualquer tipo de serviço web.

Na Figura 12 demonstra-se o fluxo utilizado para se receber as notificações. Faz-se uma requisição GET do protocolo HTTP para a nuvem *PushingBox*. A variável *devID* referencia para o *PushingBox* qual o dispositivo que deve receber a notificação.

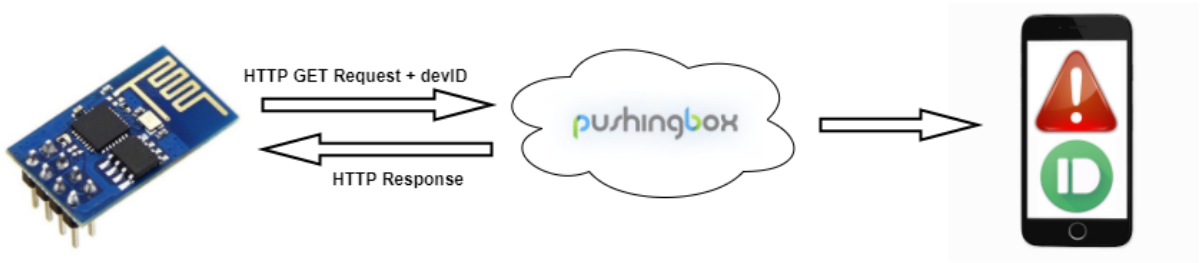



Figura 12: Fluxo para envio de notificação.

O módulo também foi utilizado para o envio de dados referentes a uma possível queda detectada. Para isso, utilizou-se novamente do *PushingBox* como API para enviar as informações para uma planilha hospedada pelo Drive da Google [16].

Após o envio da notificação, envia-se uma nova requisição HTTP com 4 informações relevantes selecionadas do Algoritmo I:

- Data e hora
- Módulo da aceleração no limiar inferior
- Módulo da aceleração no limiar superior
- Valor da mudança de orientação

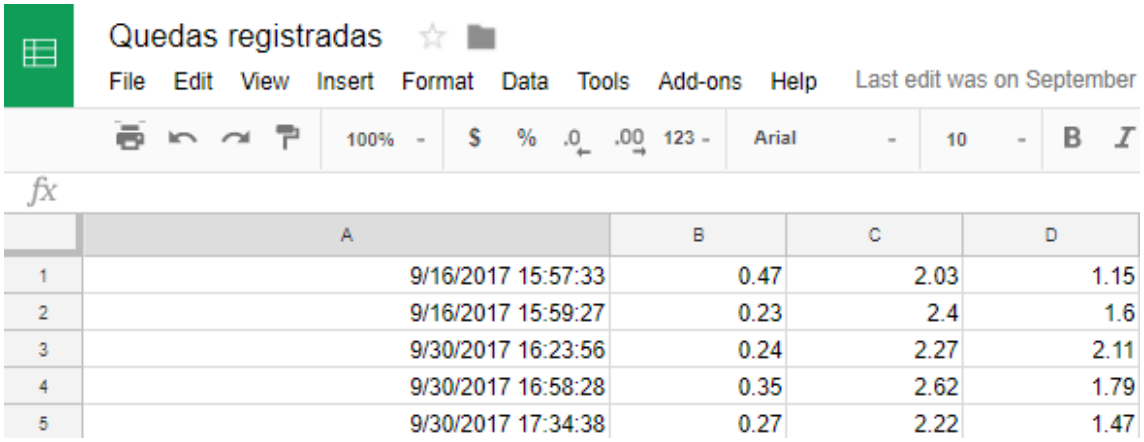
Essas informações são então enviadas através do *PushingBox* e recebidas em um API da própria Google configurado para receber estes dados e armazenar nas respectivas colunas de uma planilha. Isto é somente possível através da utilização do editor de Scripts do Google Planilhas, sendo possível interagir com o API do *PushingBox* e armazenar as informações nas respectivas colunas sem sobrepor as anteriores.

Novamente utiliza-se de uma requisição GET para o API, juntamente com a identificação da planinha de dados e os valores das variáveis utilizadas como condicionais. Este fluxo pode ser visualizado na Figura 13.


Figura 13: Fluxo para envio de dados

Para armazenar as variáveis na planinha um *script* base foi utilizado como exemplo [17], foram feitas as devidas alterações no código para salvar as variáveis corretamente. O algoritmo utilizado no microcontrolador também foi adaptado para o envio dos dados na ordem e com o mesmo identificador das variáveis sendo recebidas pelo API.

Na Figura 10 é possível visualizar como estes dados ficam armazenados no Google Drive.




| | A | B | C | D |
|---|---|---|---|---|
| 1 | 9/16/2017 15:57:33 | 0.47 | 2.03 | 1.15 |
| 2 | 9/16/2017 15:59:27 | 0.23 | 2.4 | 1.6 |
| 3 | 9/30/2017 16:23:56 | 0.24 | 2.27 | 2.11 |
| 4 | 9/30/2017 16:58:28 | 0.35 | 2.62 | 1.79 |
| 5 | 9/30/2017 17:34:38 | 0.27 | 2.22 | 1.47 |

Figura 14: Dados do Algoritmo I de queda em Nuvem.

Tais informações podem ser utilizadas tanto como log das quedas detectadas, quanto para uma otimização dos algoritmos no futuro.

## 4. Resultados

O protótipo foi colocado na região da cintura com o auxílio de um cinto e submetido a alguns testes de queda e também situações em que não há queda mas gera um alto valor de aceleração, tentando avaliar todos os aspectos e eventuais ocorrências com a utilização do mesmo.

As situações escolhidas para teste em que ocorre queda, foram cenários onde a pessoa cai para frente ou para trás, que são os tipos de quedas que ocorrem com maior frequência no cotidiano. As situações mais comuns para estes tipos de queda é a pessoa tropeçar ou escorregar, respectivamente.

Na Figura 15 é possível visualizar dois diagramas com os respectivos fluxos dos dois tipos de teste de quedas realizados. Utilizou-se um colchão para apoio para evitar lesões.

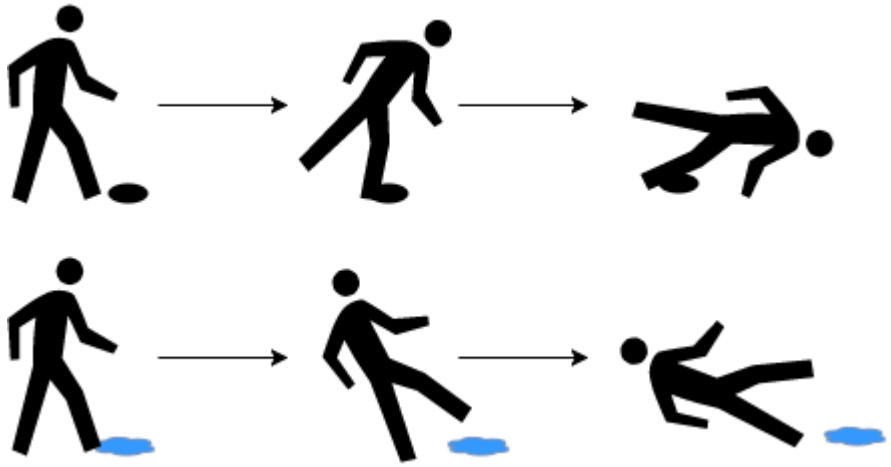


Figura 15: Simulações de queda realizadas.

Já para os testes de falsos positivos, foram definidos os seguintes cenários: sentar, andar rapidamente, dançar, agachar. Essas situações são as mais citadas e que ocorrem com maior frequência segundo a literatura [4].

Para os testes de sentar, utilizou-se de um sofá como apoio e a posição final da pessoa levemente inclinada. Para os testes de dançar, foram realizados movimentos aleatórios de tipos de dança distintos como por exemplo valsa, anos 80 e forró.

Foram simuladas 20 quedas com o protótipo em possíveis situações como tropeçar, causando uma queda para frente e escorregando, que leva a uma queda para trás. O mesmo número de testes foram realizados para as

situações em que não ocorre queda. O processo foi realizado para os 2 algoritmos desenvolvidos e por 2 pessoas distintas.

Os resultados obtidos para o Algoritmo I são apresentados na Tabela 3.

Tabela 3: Resultados dos testes do Algoritmo I

| | Queda ocorre | Queda não ocorre |
|---|---|---|
| Quedas detectada | TP = 31 | FP = 1 |
| Quedas não detectadas | FN = 9 | TN = 39 |

Definidos os valores obtidos experimentalmente, é possível calcular os valores de sensibilidade, especificidade e precisão para esse algoritmo:

- Sensibilidade = 0.775
- Especificidade = 0.975
- Precisão = 0.875

Resultados com informações para cada tipo de situação de queda e de não queda testados podem ser visualizados nas tabelas 4 e 5 para o voluntário I e tabelas 6 e 7 para o voluntário II.

O algoritmo teve um excelente valor de especificidade, conforme observado pelo baixo número de falso positivos obtidos. O algoritmo consegue identificar muito bem situações de movimentos em que não ocorre queda, evitando falsos alarmes. Já a sensibilidade teve um valor abaixo de 80%, indicando que o mesmo falhou na detecção de algumas quedas reais simuladas.

Tabela 4: Resultados dos testes de queda do Algoritmo I para o voluntário I

| | Queda para frente | Queda para trás |
|---|---|---|
| Quedas detectada | TP = 8 | TP = 9 |
| Quedas não detectadas | FN = 2 | FN = 1 |

Tabela 5: Resultados dos testes em que não ocorre queda do Algoritmo I para o voluntário I

| | Andar rapidamente | Sentar | Dançar | Agachar |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Quedas detectada | FP = 0 | FP =1 | FP = 0 | FP = 0 |
| Quedas não detectadas | TN = 5 | TN = 4 | TN = 5 | TN = 5 |

Tabela 6: Resultados dos testes de queda do Algoritmo I para o voluntário II

| | Queda para frente | Queda para trás |
|---|---|---|
| Quedas detectada | TP = 6 | TP = 8 |
| Quedas não detectadas | FN = 4 | FN = 2 |

Tabela 7: Resultados dos testes em que não ocorre queda do Algoritmo I para o voluntário II

| | Andar rapidamente | Sentar | Dançar | Agachar |
|---|---|---|---|---|
| Quedas detectada | FP = 0 | FP = 0 | FP = 0 | FP = 0 |
| Quedas não detectadas | TN = 5 | TN = 5 | TN = 5 | TN = 5 |

O maior número falsos negativos aconteceu em quedas em que o voluntário não gerou aceleração suficiente para atingir o limiar superior de detecção, principalmente no voluntário II, onde as quedas simuladas foram mais leves.

O único falso positivo foi em uma situação onde o voluntário I sentou no sofá de forma mais rápida e o protótipo se deslocou para uma posição fora da posição referência medida para o mesmo.

Os resultados obtidos para o Algoritmo II são apresentados na Tabela 8.

Tabela 8: Resultados dos testes do Algoritmo II

| | Queda ocorre | Queda não ocorre |
|---|---|---|
| Quedas detectada | TP = 33 | FP = 6 |
| Quedas não detectadas | FN = 7 | TN = 34 |

Calcula-se os valores de sensibilidade, especificidade e precisão para o Algoritmo II:

- Sensibilidade = 0.825

- Especificidade = 0.85
- Precisão = 0.8375

Resultados detalhados para os testes e os voluntários I e II podem ser visualizados nas tabelas 9,10 e 11,12, respectivamente.

Este algoritmo teve números mais próximos para os 3 valores calculados, demostrando um bom desempenho e equivalente tanto na detecção de quedas quanto evitando falsos alarmes.

Tabela 9: Resultados dos testes de queda do Algoritmo II para o voluntário I

| | Queda para frente | Queda para trás |
|---|---|---|
| Quedas detectada | TP = 9 | TP = 9 |
| Quedas não detectadas | FN = 1 | FN = 1 |

Tabela 10: Resultados dos testes em que não ocorre queda do Algoritmo II para o voluntário I

| | Andar rapidamente | Sentar | Dançar | Agachar |
|---|---|---|---|---|
| Quedas detectada | FP = 2 | FP = 1 | FP = 1 | FP = 0 |
| Quedas não detectadas | TN = 3 | TN = 4 | TN = 4 | TN = 5 |

Tabela 11: Resultados dos testes de queda do Algoritmo II para o voluntário II

| | Queda para frente | Queda para trás |
|---|---|---|
| Quedas detectada | TP = 7 | TP = 8 |
| Quedas não detectadas | FN = 3 | FN = 2 |

Tabela 12: Resultados dos testes em que não ocorre queda do Algoritmo II para o voluntário II

| | Andar rapidamente | Sentar | Dançar | Agachar |
|---|---|---|---|---|
| Quedas detectada | FP = 0 | FP = 0 | FP = 2 | FP = 0 |
| Quedas não detectadas | TN = 5 | TN = 5 | TN = 3 | TN = 5 |

Este algoritmo teve um maior número de falsos positivos em relação ao anterior, no qual qualquer movimento mais brusco como por exemplo andar rapidamente de forma que o sensor se mova ou dançar foram detectadas como possíveis quedas. Os falsos negativos nesse caso aconteceram novamente em quedas mais leves ou em que o acelerômetro acaba ficando próximo a posição original.

O Algoritmo II teve um melhor desempenho na detecção de quedas do que o Algoritmo I, no qual o valor da sensibilidade foi relativamente maior. Entretanto, pela diferença significativa dos valores de especificidade, pode-se afirmar que o Algoritmo I gerou significativamente menos resultados falsos positivos. A verificação de se há movimentação e se a pessoa está na posição de referência em 2,5 segundos após a detecção de uma possível queda efetuada pelo mesmo ajudou a evitar esse número.

Os valores são próximos para a precisão, mas bem distintos para a sensibilidade e a especificidade. O Algoritmo I atua muito bem em evitar falsos positivos, entretanto falhou em um número maior de detecção de quedas. Já o Algoritmo II conseguiu identificar um melhor número de quedas reais nos testes realizados, entretanto teve um número de falsos positivos relativamente maior.

Para uma situação em que é extremamente relevante a detecção de qualquer tipo de queda, este valor de sensibilidade tem uma maior importância na avaliação dos algoritmos. O número de falsos positivos é significativo para evitar falsos alarmes, no qual o botão para evitar quedas errôneas foi acrescentado com este objetivo.

Uma informação a se considerar é o fato do voluntário II possuir um peso menor que o primeiro, e o número de quedas detectadas em ambos os algoritmos para esta pessoa foi menor. Um ajuste nos valores limiar do módulo da aceleração para detecção da queda pode ser feita nos dois algoritmos, melhorando o desempenho dos mesmos para esta situação.

No geral, o desempenho do protótipo para efetuar as atividades propostas com os respectivos módulos foi muito bom. O protótipo se manteve estável durante a execução dos algoritmos e não houveram falhas graves que necessitaram a reinicialização do mesmo.

O protótipo desenvolvido não teve um custo muito elevado, o custo de cada componente adquirido no ano de 2016 são listados na Tabela 13. A maioria dos produtos oferecidos no mercado cobram valores de mensalidade ultrapassando este custo.

Tabela 13: Gastos com os materiais.

| Material | Valor Pago (R$) |
|---|---|
| Arduino UNO DIP | 45,00 |
| Módulo Wi-Fi ESP-01 | 20,00 |
| Placa GY-521 | 20,00 |
| Protoboard | 20,00 |
| Regulador de Tensão AMS | 8,00 |
| Total | 113,00 |

O valor total gasto de aproximadamente R$113,00 aumentaria significativamente com uma versão do protótipo para testes finais com um circuito impresso, uma bateria recarregável e uma proteção plástica, entretanto seria um custo viável dada baixa manutenção e remuneração mensal para utilização do mesmo.

A memória total ocupada pelo Algoritmo I, que ocupou 2KB a mais de memória, preencheu 18 KB dos 32KB (56%) de memória *flash* disponibilizada pelo microcontrolador. Ainda há memória disponível para eventuais manutenções e adição de novas funcionalidades nos códigos, ainda mais devido a possível utilização da memória do ESP-01, que é de 1MB.

# 5. Conclusão

Foi possível desenvolver e testar um protótipo para monitoramento e detecção instantânea de quedas, alertando um dispositivo móvel através de um API, além do envio de dados da queda para uma planilha em Nuvem. Os módulos de WiFi, acelerômetro e giroscópio são conectados no microcontrolador, que executa um código compilado para realizar as atividades propostas.

A comunicação com os módulos é feita através do próprio microcontrolador, recebendo informações e calculando as variáveis relevantes para o monitoramento de queda em tempo real. É também o este que aciona o módulo WiFi quando necessário.

Dois algoritmos de queda foram desenvolvidos e testados, com o intuito de avaliar o desempenho de cada um em diferentes situações e cenários. Utilizando métodos de detecção distintos, é possível perceber a diferença nos resultados. O primeiro detecta mais quedas enquanto o segundo desempenha melhor à situações de falsos positivos, onde ocorrem um menor número de detecções errôneas.

Ambos os algoritmos tiveram um desempenho próximo ao da literatura, onde os valores de sensibilidade e precisão não ficaram distante dos obtidos experimentalmente, que em média ficam em torno de 85% e 90% [4], respectivamente.

## 5.1. Considerações Finais

O envio de notificações utilizando um único API permite alertas em diversas plataformas diferentes. Pode-se receber a notificação via e-mail, navegador web e diversos aplicativos para smartphone ou tablet. Um aplicativo próprio pode comunicar-se com esse API para receber alertas e dados do Arduino.

A planilha dos dados de queda pode ser utilizado posteriormente para otimizar o desempenho do algoritmo. Pode ser aplicado algum método estatístico ou até mesmo de aprendizado de máquina para melhorar a detecção de quedas baseando-se nessas informações.

Os testes realizados envolveram a simulação de diferentes tipos de quedas e em pessoas com características diferentes. Ainda assim não é possível abranger todas as possibilidades de quedas e diferenças físicas entre as pessoas, sendo necessário um maior número de amostras para uma análise de desempenho mais detalhada.

Ambos os algoritmos utilizam valores limite para o módulo da aceleração, no qual foi utilizado um valor próximo ao recomendado pela literatura. Observou-se que a alteração desses valores influencia diretamente na detecção de quedas e possíveis falsos positivos, podendo ser alterado dependendo das características de uma pessoa, com o intuito de melhorar ainda mais o desempenho dos mesmos.

## 5.2. Trabalhos Futuros

Este protótipo foi desenvolvido através de uma conexão USB como alimentação para o Arduino e os respectivos módulos. Para um funcionamento ideal deste aparelho, é necessário alimentação própria. Uma bateria recarregável poderia ser utilizada para este fim, com um respectivo regulador de tensão.

A remoção do microcontrolador da placa Arduino, possibilita a implementação de um circuito menor, uma vez que os programas padrão para o funcionamento dos dispositivos fica salvo no próprio microcontrolador. O mesmo circuito provido pelo Arduino DIP pode ser implementado com o auxílio de resistores e um LED. Isto facilitaria o desenvolvimento de uma proteção plástica para melhorar o design do protótipo e proteger o circuito interno.

O módulo WiFi deve ser propriamente configurado para funcionamento e necessita da reinicialização se algum problema ocorre com a conexão. Seria ideal a adição de um botão para a reinicialização do circuito para facilitar a manutenção de eventuais problemas do tipo. Um módulo de áudio ou com função de vibração também pode ser adicionado a fim de melhorar a percepção do usuário quando ocorre um falso positivo, além do LED já existente.

## 6. Referências


[1] L. Atzori et al. **The Internet of Things: A survey**, 2010. Computer Networks, vol. 54 p. 2787–2805.

[2] Gubbi et al. **Internet of Things (IoT): A vision, architectural elements, and future directions**, 2013. Future Generation Computer Systems, vol. 29, ed. 7, p. 1645-1660.

[3] D. Miorando et al. **Internet of Things: Vision, applications and research challenges**, 2012. Ad Hoc Networks, vol. 10, p. 1497–1516.

[4] Stefano Abbate, Marco Avvenuti, Paolo Corsini, Janet Light and Alessio Vecchio. **Monitoring of Human Movements for Fall Detection and Activities Recognition in Elderly Care Using Wireless Sensor Network: a Survey**, 2010. Wireless Sensor Networks: Application-Centric Design, ISBN: 978-953-307-321- 7, InTech.

[5] A. Sixsmith e N. Johnson. **A Smart Sensor to Detect The Falls of the Elderly,** 2004. IEEE Pervasive Computing, vol. 3, ed. 2, p. 42-47.

[6] Suryadevara e Mukhopadhyay. **Wireless Sensor Network Based Home Monitoring System for Wellness Determination of Elderly**, 2012. IEEE Sensors Journal, vol. 12, ed. 6, p. 1965-1972.

[7] Top Ten Reviews. Best fall detection sensors of 2016. Disponível em:<http://www.toptenreviews.com/health/senior-care/best-fall-detection-sensors/>. Acesso em 03/12/2016.

[8] Datasheet ESP8266. Disponível em:<http://www.electroschematics.com/wp-content/uploads/2015/02/esp8266-datasheet.pdf>. Acesso em 25/04/2017.

[9] Script de calibração do MPU6050. Disponível em: <https://www.i2cdevlib.com/forums/topic/96-arduino-sketch-to-automatically-calculate-mpu6050-offsets/>. Acesso em 25/04/2017.

[10] Datasheet MPU6050. Disponível em: <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. Acesso em 25/04/2017.

[11] Jonathan Tomkum and Binh Nguyen. **Design of a Fall Detection and Prevention System for the Elderly**, 2010. McMaster University Hamilton, Ontario, Canada.

[12] S. Hwang et al. **Fall detection with three-axis accelerometer and magnetometer in a smartphone**, 2012. Proceedings of the International Conference on Computer Science and Technology. p. 25-27.

[13] J. Luo, B. Zhong, Dinghao LV. **Fall Monitoring Device for Old People based on Tri-Axial Accelerometer.** Int J Adv Comput Sci Appl, v. 6, n. 5, 2015.

[14] Instruction set ESP8266. Disponível em: <https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf>. Acesso em 25/04/2017.

[15] *PushingBox*. Disponível em: <https://www.pushingbox.com>. Acesso em 25/04/2017.

[16] Google Drive. Disponível em: <https://drive.google.com>. Acesso em 25/04/2017.

[17] Script para envio de dados ao Google Drive. Disponível em: <https://www.hackster.io/detox/transmit-esp8266-data-to-google-sheets-8fc617>. Acesso em: 21/09/2017.

## Anexo A

### Código Algoritmo I

```
#include<Wire.h>
#include <SoftwareSerial.h>

#define DEVID "vCD952E5AB68C415" // id do aparelho
#define DEVIDLOG "v0EF18F8569680F0" // id do API criado no Google Sheets

SoftwareSerial ESP01(3, 2); // 3 no TX do ESP, 2 no RX do ESP

int LED13 = 13; // led 13 do arduino para piscar com a queda

float Acel;
int Orie;

bool acel_baixa = false; //se aceleracao baixa detectada
bool acel_alta = false; //se aceleracao alta detectada
bool orientacao_mudou = false; //se houve mudanca de orientacao

float acel1;
float acel2;
float ori;

int cont_baixa = 0;
int cont_alta = 0;
int cont_orientacao = 0;

// valores do acelerometro para uma pessoa na posicao vertical
float pos_x = -0.18;
float pos_y = 1.02;
float pos_z = 0.02;

int pos_Acel = pow(pow(pos_x,2)+pow(pos_y,2)+pow(pos_z,2),0.5); // magnitude na p
osicao vertical
float mudanca_ori;

void setup() {
  ESP01.begin(9600);
  Wire.begin();
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0);     // liga o MPU6050 no endereço 0x6B
  Wire.endTransmission(true);
  Serial.begin(9600);
  pinMode(LED13, OUTPUT);
  pinMode(12, INPUT_PULLUP);

}
void loop() {


  delay(100); // delay para controle
  obterAcelOrie();
  //Serial.println(Acel);
  if (Acel <= 0.6 && acel_alta == false) { //magnitude da aceleracao baixa

    // Dimunuindo-
se esse valor, requer um movimento inicial de queda mais brusco

    acel1 = Acel;
```

```
58.     acel_baixa = true;
59.     Serial.println("LIMIAR INFERIOR");
60.   }
61.
62.   if (cont_baixa >= 5) { // se em 0.5segs a magnitude nao superar os 3g
63.     acel_baixa = false;
64.     cont_baixa = 0;
65.   }
66.
67.   if (acel_baixa == true) {
68.     cont_baixa++;
69.     if (Acel >= 2) { //se a magnitude for maior que 2g
70.
71.       // Aumentado-
    se esse valor, requer um maior impacto para deteccao de queda
72.
73.       acel2 = Acel;
74.       acel_alta = true;
75.       acel_baixa = false;
76.       cont_baixa = 0;
77.
78.       Serial.println("LIMIAR SUPERIOR");
79.     }
80.   }
81.
82.   if (cont_alta >= 5) { // se em 0.5 segs nao houver mudança de direcao
83.     acel_alta = false;
84.     cont_alta = 0;
85.   }
86.
87.   if (acel_alta == true) {
88.     cont_alta++;
89.     if (mudanca_ori >= 1.1) { // pessoa na vertical: mudanca_ori = 0, se nao esta
     na vertical, maior que o limiar de 1.1 e detectada mudanca de orientacao,
90.
91.       // variavel mudanca_ori melhor que a magnitude da orientacao (Orie) para es
    te caso, pois se baseia na posicao base da pessoa.
92.       // Aumentando-
    se esse valor requer uma maior diferenca da posicao do prototipo em relacao a pos
    icao vertical
93.
94.       ori = mudanca_ori;
95.       orientacao_mudou = true;
96.       acel_alta = false;
97.       cont_alta = 0;
98.       Serial.println("MUDANCA DE ORIENTACAO");
99.     }
100.        }
101.
102.        if (orientacao_mudou == true) {
103.          cont_orientacao++;
104.
105.          if (cont_orientacao >= 25) { // 2.5 segundos para se mover (posicao do
     giroscopio nao varia)
106.            if ((Orie >= 0) && (Orie <= 10) && mudanca_ori >= 1.1) {
107.
108.              // Magnitude da orientacao (Orie) é baixa se a pessoa esta parada.

109.
110.              // Outra opcao: Se em 2,5 segundos a pessoa esta em pe novamente.

111.
112.              quedaOcorreu(acel1,acel2,ori); // queda pois pessoa ainda esta sem
     se mover
113.
114.              orientacao_mudou = false;
```

```
115.               cont_orientacao = 0;
116.             }
117.             else {
118.
119.               orientacao_mudou = false;
120.               cont_orientacao = 0;
121.             }
122.           }
123.         }
124.       }
125.
126.       void obterAcelOrie() {
127.         int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
128.         float ax = 0, ay = 0, az = 0, gx = 0, gy = 0, gz = 0;
129.
130.         Wire.beginTransmission(0x68); //transmissao no endereço do acelerometro
131.         Wire.write(0x3B);  //registrador 0x3B
132.         Wire.endTransmission(false);
133.         Wire.requestFrom(0x68, 14, true); // 14 registradores
134.         AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
135.         AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
136.         AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
137.         Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
138.         GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
139.         GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
140.         GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
141.
142.         ax = (AcX - 4879) / 16384.0;
143.         ay = (AcY + 564) / 16384.0;
144.         az = (AcZ + 1055) / 16384.0;
145.         gx = (GyX + 76) / 131.0;
146.         gy = (GyY + 6) / 131.0;
147.         gz = (GyZ - 3) / 131.0;
148.
149.         // valores do acelerometro calibrados e divisao pela resolucao
150.
151.         Acel = (pow(pow(ax, 2) + pow(ay, 2) + pow(az, 2), 0.5));
152.       //  Serial.println(Acel);
153.         Orie = pow(pow(gx, 2) + pow(gy, 2) + pow(gz, 2), 0.5);
154.         mudanca_ori=acos((double(ax)*double(pos_x)+double(ay)*double(pos_y)+double(az)*double(pos_z))/(double(Acel)*double(pos_Acel))); // calculo da orientacao baseada na posicao base (vertical)
155.       //  Serial.println(mudanca_ori);
156.       }
157.
158.       void quedaOcorreu(float acel1, float acel2, float ori) {
159.         Serial.println("QUEDA DETECTADA");
160.
161.         for (int i = 0; i < 100; i++) {
162.           bool botao_desativar = digitalRead(12);
163.           digitalWrite(LED13, HIGH);
164.           delay(50);
165.           if (botao_desativar) { //se nao tiver sido acionado verifica novamente após delay
166.             botao_desativar = digitalRead(12);
167.           }
168.           digitalWrite(LED13, LOW);
169.           delay(50);
```

```
170.           if (botao_desativar == false) {
171.             Serial.println("BOTAO DESATIVAR ACIONADO!!");
172.             return;
173.           }
174.         }
175.         enviarPushingBox(DEVID);
176.         enviarLogPushingBox(DEVIDLOG,acel1,acel2,ori);
177.       }
178.       String enviarComando(String cmd, const int timeout)
179.       {
180.         char c;
181.         String resposta = "";
182.
183.         ESP01.print(cmd);
184.         long int time = millis();
185.         while ((time + timeout) > millis()) // calcula o timeout para sair
186.         {
187.           while (ESP01.available())
188.           {
189.             c = ESP01.read(); //le cada caracter
190.             resposta += c;
191.           }
192.         }
193.         Serial.print(resposta);
194.         return resposta;
195.       }
196.
197.       void enviarPushingBox(String devid) {
198.
199.         String cmd = "AT+CIPSTART=\"TCP\",\"";
200.         cmd += "api.pushingbox.com";
201.         cmd += "\",80";  // conexao http
202.         ESP01.println(cmd);
203.         delay(2000);
204.
205.         // confere se o esp nao retornou ERROR
206.         if (ESP01.find("Error")) {
207.           Serial.println("Erro ao enviar notificação");
208.           return;
209.         }
210.         cmd = "GET /pushingbox?devid=" + devid + "  HTTP/1.1\r\nHost: api.pushin
    gbox.com\r\nUser-Agent: Arduino\r\nConnection: close\r\n\r\n";
211.
212.         ESP01.print("AT+CIPSEND=");
213.         ESP01.println(cmd.length());
214.         delay(1000);
215.
216.         ESP01.print(cmd); // executa o request http para enviar notificacao
217.         enviarComando("AT+CIPCLOSE", 2000); // fecha a conexao, recebendo a resp
    osta http
218.       }
219.
220.       void enviarLogPushingBox(String devid, float acel1, float acel2, float ori
    ) {
221.
222.         String cmd = "AT+CIPSTART=\"TCP\",\"";
223.         cmd += "api.pushingbox.com";
224.         cmd += "\",80";  // conexao http
225.         ESP01.println(cmd);
226.         delay(2000);
227.
228.         // confere se o esp nao retornou ERROR
229.         if (ESP01.find("Error")) {
230.           Serial.println("Erro ao enviar notificação");
231.           return;
232.         }
```

```
233.         cmd = "GET /pushingbox?devid=" + devid + "&acel1=" + (String) acel1
234.              + "&acel2="      + (String) acel2
235.              + "&ori="     + (String) ori
236.              + "  HTTP/1.1\r\nHost: api.pushingbox.com\r\nUser-
    Agent: Arduino\r\nConnection: close\r\n\r\n";
237.
238.         ESP01.print("AT+CIPSEND=");
239.         ESP01.println(cmd.length());
240.         delay(1000);
241.
242.         ESP01.print(cmd); // executa o request http para enviar notificacao
243.         enviarComando("AT+CIPCLOSE", 2000); // fecha a conexao, recebendo a resp
    osta http
244.       }
```

### Código Algoritmo II

```
1.  #include<Wire.h>
2.  #include <SoftwareSerial.h>
3.
4.  #define DEVID "vCD952E5AB68C415" // id do aparelho
5.  #define DEVIDLOG "v0EF18F8569680F0" // id do API criado no Google Sheets
6.
7.  SoftwareSerial ESP01(3, 2); // 3 no TX do ESP, 2 no RX do ESP
8.
9.  int LED13 = 13; // led 13 do arduino para piscar com a queda
10.
11. float Acel;
12. int Orie;
13.
14. bool acel_baixa = false; //se aceleracao baixa detectada
15. bool acel_alta = false; //se aceleracao alta detectada
16. bool orientacao_mudou = false; //se houve mudanca de orientacao
17.
18. float acel1;
19. float acel2;
20. float ori;
21.
22. int cont_baixa = 0;
23. int cont_alta = 0;
24. int cont_orientacao = 0;
25.
26. // valores do acelerometro para uma pessoa na posicao vertical
27. float pos_x = -0.18;
28. float pos_y = 1.02;
29. float pos_z = 0.02;
30.
31. int pos_Acel = pow(pow(pos_x,2)+pow(pos_y,2)+pow(pos_z,2),0.5); // magnitude na p
    osicao vertical
32. float mudanca_ori;
33.
34. void setup() {
35.   ESP01.begin(9600);
36.   Wire.begin();
37.   Wire.beginTransmission(0x68);
38.   Wire.write(0x6B);
39.   Wire.write(0);     // liga o MPU6050 no endereço 0x6B
40.   Wire.endTransmission(true);
41.   Serial.begin(9600);
42.   pinMode(LED13, OUTPUT);
43.   pinMode(12, INPUT_PULLUP);
44.
```

```
45. }
46. void loop() {
47.
48.   delay(100); // delay para controle
49.   obterAcelOrie();
50.
51.     if (Acel >= 2) { //se a magnitude for maior que 2g
52.
53.       // Aumentando-
    se esse valor, requer um maior impacto para detecção de queda
54.
55.       acel2 = Acel;
56.       acel_alta = true;
57.
58.       Serial.println("LIMIAR SUPERIOR");
59.   }
60.
61.   if (cont_alta >= 200) { // 20 segs para aceleracao nao estabilizar
62.     acel_alta = false;
63.     cont_alta = 0;
64.   }
65.
66.   if (acel_alta == true) {
67.     cont_alta++;
68.     if ((Acel < 1.3) && (Acel > 0.7)) {
69.
70.       ori = mudanca_ori;
71.       orientacao_mudou = true;
72.       acel_alta = false;
73.       cont_alta = 0;
74.       Serial.println("ACEL NORMAL");
75.     }
76.   }
77.
78.   if (orientacao_mudou == true) {
79.
80.       if (mudanca_ori >= 1.1) {
81.
82.         quedaOcorreu(acel1,acel2,ori); // queda pois pessoa nao esta na posicao n
    ormal
83.
84.         orientacao_mudou = false;
85.         cont_orientacao = 0;
86.       }
87.       else {
88.
89.         orientacao_mudou = false;
90.         cont_orientacao = 0;
91.       }
92.     }
93.   }
94.
95.
96. void obterAcelOrie() {
97.   int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
98.   float ax = 0, ay = 0, az = 0, gx = 0, gy = 0, gz = 0;
99.
100.         Wire.beginTransmission(0x68); //transmissao no endereço do acelerometro

101.         Wire.write(0x3B);  //registrador 0x3B
102.         Wire.endTransmission(false);
103.         Wire.requestFrom(0x68, 14, true); // 14 registradores
104.         AcX = Wire.read() << 8 | Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACC
    EL_XOUT_L)
105.         AcY = Wire.read() << 8 | Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACC
    EL_YOUT_L)
```

```
106.          AcZ = Wire.read() << 8 | Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACC
    EL_ZOUT_L)
107.          Tmp = Wire.read() << 8 | Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_
    OUT_L)
108.          GyX = Wire.read() << 8 | Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO
    _XOUT_L)
109.          GyY = Wire.read() << 8 | Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO
    _YOUT_L)
110.          GyZ = Wire.read() << 8 | Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO
    _ZOUT_L)
111.
112.          ax = (AcX - 4879) / 16384.0;
113.          ay = (AcY + 564) / 16384.0;
114.          az = (AcZ + 1055) / 16384.0;
115.          gx = (GyX + 76) / 131.0;
116.          gy = (GyY + 6) / 131.0;
117.          gz = (GyZ - 3) / 131.0;
118.
119.          // valores do acelerometro calibrados e divisao pela resolucao
120.
121.          Acel = (pow(pow(ax, 2) + pow(ay, 2) + pow(az, 2), 0.5));
122.        //  Serial.println(Acel);
123.          Orie = pow(pow(gx, 2) + pow(gy, 2) + pow(gz, 2), 0.5);
124.          mudanca_ori=acos((double(ax)*double(pos_x)+double(ay)*double(pos_y)+doub
    le(az)*double(pos_z))/(double(Acel)*double(pos_Acel))); // calculo da orientacao
    baseada na posicao base (vertical)
125.        //  Serial.println(mudanca_ori);
126.        }
127.
128.        void quedaOcorreu(float acel1, float acel2, float ori) {
129.          Serial.println("QUEDA DETECTADA");
130.
131.          for (int i = 0; i < 100; i++) {
132.            bool botao_desativar = digitalRead(12);
133.            digitalWrite(LED13, HIGH);
134.            delay(50);
135.            if (botao_desativar) { //se nao tiver sido acionado verifica novamente
     após delay
136.              botao_desativar = digitalRead(12);
137.            }
138.            digitalWrite(LED13, LOW);
139.            delay(50);
140.            if (botao_desativar == false) {
141.              Serial.println("BOTAO DESATIVAR ACIONADO!!");
142.              return;
143.            }
144.          }
145.          enviarPushingBox(DEVID);
146.          //enviarLogPushingBox(DEVIDLOG,acel1,acel2,ori);
147.        }
148.        String enviarComando(String cmd, const int timeout)
149.        {
150.          char c;
151.          String resposta = "";
152.
153.          ESP01.print(cmd);
154.          long int time = millis();
155.          while ((time + timeout) > millis()) // calcula o timeout para sair
156.          {
157.            while (ESP01.available())
158.            {
159.              c = ESP01.read(); //le cada caracter
160.              resposta += c;
161.            }
162.          }
163.          Serial.print(resposta);
```

```
164.          return resposta;
165.        }
166.
167.        void enviarPushingBox(String devid) {
168.
169.          String cmd = "AT+CIPSTART=\"TCP\",\"";
170.          cmd += "api.pushingbox.com";
171.          cmd += "\",80";  // conexao http
172.          ESP01.println(cmd);
173.          delay(2000);
174.
175.          // confere se o esp nao retornou ERROR
176.          if (ESP01.find("Error")) {
177.            Serial.println("Erro ao enviar notificação");
178.            return;
179.          }
180.          cmd = "GET /pushingbox?devid=" + devid + "  HTTP/1.1\r\nHost: api.pushin
    gbox.com\r\nUser-Agent: Arduino\r\nConnection: close\r\n\r\n";
181.
182.          ESP01.print("AT+CIPSEND=");
183.          ESP01.println(cmd.length());
184.          delay(1000);
185.
186.          ESP01.print(cmd); // executa o request http para enviar notificacao
187.          enviarComando("AT+CIPCLOSE", 2000); // fecha a conexao, recebendo a resp
    osta http
188.        }
189.
190.        void enviarLogPushingBox(String devid, float acel1, float acel2, float ori
    ) {
191.
192.          String cmd = "AT+CIPSTART=\"TCP\",\"";
193.          cmd += "api.pushingbox.com";
194.          cmd += "\",80";  // conexao http
195.          ESP01.println(cmd);
196.          delay(2000);
197.
198.          // confere se o esp nao retornou ERROR
199.          if (ESP01.find("Error")) {
200.            Serial.println("Erro ao enviar notificação");
201.            return;
202.          }
203.          cmd = "GET /pushingbox?devid=" + devid + "&acel1=" + (String) acel1
204.               + "&acel2="      + (String) acel2
205.               + "&ori="     + (String) ori
206.               + "  HTTP/1.1\r\nHost: api.pushingbox.com\r\nUser-
    Agent: Arduino\r\nConnection: close\r\n\r\n";
207.
208.          ESP01.print("AT+CIPSEND=");
209.          ESP01.println(cmd.length());
210.          delay(1000);
211.
212.          ESP01.print(cmd); // executa o request http para enviar notificacao
213.          enviarComando("AT+CIPCLOSE", 2000); // fecha a conexao, recebendo a resp
    osta http
214.        }
```