UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

GABRIEL TUTIA CORNEJO
LEONARDO CENTENARO RAMOS

Dispositivo de geolocalização para auxílio de idosos com perda de memória

São Paulo
2018

GABRIEL TUTIA CORNEJO
LEONARDO CENTENARO RAMOS

Dispositivo de geolocalização para auxílio de idosos com perda de memória

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia Mecatrônica

Área de Concentração: Engenharia Mecatrônica
Departamento: PMR - Mecatrônica e Sistemas Mecânicos

Autores: Gabriel Tutia Cornejo - 8989085
Leonardo Centenaro Ramos - 8988302
Orientador: Prof. Dr. Rafael Traldi Moura
Coorientador: Prof. Dr. Reginaldo Arakaki

São Paulo
2018

Catalogação-na-publicação

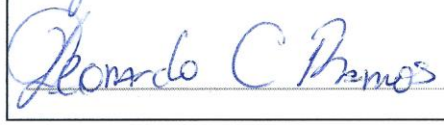
Tutia, Gabriel
Dispositivo de geolocalização para auxílio de idosos com perda de memória / G. Tutia, L. Ramos -- São Paulo, 2018.
84 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Mecatrônica 3.Idosos I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t. III.Ramos, Leonardo

Este relatório é apresentado como requisito parcial para obtenção do grau de Bacharel na Escola Politécnica da Universidade de São Paulo. É o produto do meu próprio trabalho, exceto onde indicado no texto. O relatório pode ser livremente copiado e distribuído desde que a fonte seja citada.

Assinatura dos autores



Agradecimentos

Aos nossos orientadores Rafael e Reginaldo que nos guiaram e auxiliaram ao longo deste último ano.

À EVO System pelo apoio técnico fundamental para a conclusão deste trabalho.

Aos nossos colegas de classe, sempre presentes e dispostos a ajudar uns aos outros, não só na conclusão deste trabalho, mas também ao longo de todo trajeto na Escola Politécnica.

Aos amigos, amigas e familiares pelo suporte nos momentos mais difíceis desta graduação, pelo carinho, motivação e compreensão nos tempos difíceis.

Resumo

Segundo dados da ONU, é reconhecido mundialmente o fenômeno do envelhecimento da população. Dentre os diversos problemas que acompanham esse envelhecimento estão as alterações nas funções cognitivas, que são as principais causas da perda de independência dos idosos. O trabalho aqui apresentado busca uma solução não invasiva para o problema da perda de memória em idosos. Pretende-se obter uma prova de conceito de um dispositivo que seja capaz de localizar o idoso e identificar se ele está seguro através da utilização de sensores GPS. Inicialmente, foi realizada uma pesquisa sobre o uso de tecnologias *wireless* e *mobile* para soluções na área da saúde, conhecida como *mHealth*. Em seguida, através de uma análise mecatrônica, foram identificados os subsistemas existentes na solução proposta. Por fim, são apresentados os requisitos do projeto e as escolhas de cada componente para cada subsistema. Com os componentes definidos, os subsistemas foram integrados e testes foram realizados para medir a precisão do GPS. Além da arquitetura proposta, foi desenvolvido uma ferramenta de monitoramento do paciente a ser utilizada por médicos ou familiares.

Palavras-chave: mHealth. Geolocalização. Idoso.

Abstract

According to United Nations, the phenomenon of population ageing is recognized worldwide. Among several problems that are related to ageing are the changes in the cognitive functions, which are the main causes of the independence loss in elderly. The work presented here seeks a noninvasive solution to the problem of memory loss in elderly. It is intended to obtain, as a proof of concept, a device that is able to locate the patient and identify if he is safe using GPS sensors. Initially, a research was conducted on the use of wireless and mobile technologies for healthcare solutions, known as mHealth. Through a mechatronic analysis, the subsystems of the proposed solution were identified. Finally, the project requirements and the choices of each component for each subsystem are presented. With the components defined, the subsystems were integrated and tests were made to measure the precision of the GPS. Besides the proposed device, it was also developed a monitoring tool to be used by doctors or relatives.

Keywords: mHealth.Geolocation. Elderly.

Lista de Figuras

3.1	Arquitetura proposta	19
3.2	ESP8266 com dimensões de 25mm x 48mm	22
3.3	ESP32 com dimensões 28mm x 58mm	22
4.1	Esquemática da ligação SIM808 e ESP32	27
4.2	Protótipo de testes com SIM808 e ESP32	28
4.3	Diagrama de estados do set-up inicial	30
4.4	Diagrama de estados do loop	31
4.5	Diagrama de casos de uso para o subsistema projetado	34
4.6	Diagrama de atividades de um ciclo do dispositivo	37
4.7	Telas do aplicativo Android	39
4.8	Telas do aplicativo Android	39
4.9	Telas do aplicativo Android	39
5.1	Troca de informações entre o ESP32 e o aplicativo por Bluetooth	41
5.2	Distância do usuário até sua casa medida pelo Google Maps	42
6.1	Testes com aplicativo de monitoramento	45
6.2	Testes com aplicativo de monitoramento	45

Lista de Tabelas

3.1 Comparativo entre ESP32, ESP8266 e Arduino R3 23
5.1 Comparação das medições pelo GPS e do real 43

SUMÁRIO

Agradecimentos 5
Resumo 6
Abstract 7
Lista de Figuras 8
Lista de Tabelas 9
SUMÁRIO 10
Introdução 12
1.1 Contextualização 12
1.2 Revisão bibliográfica 13
1.2.1 Histórico 13
1.2.2 Estado da arte 14
Objetivos 18
Projeto básico 19
3.1 Análise mecatrônica 19
3.2 Parâmetros e requisitos 20
3.3 - Subsistemas 21
3.3.1 - Unidade de Controle e Bluetooth 21
3.3.2 - GPS e GPRS/GSM 24
3.3.3 – Botão do pânico não físico 24
3.3.4 – Aplicativo Android 25
3.3.5 - Carregador e bateria 25
Projeto detalhado 27
4.1 Subsistema 1 e 2 - Unidade de controle, Bluetooth, GPS e GPRS/GSM 27
4.2 Subsistema 3 – Botão do pânico não físico 32
4.3 Subsistema 4 – Aplicativo Android 33
4.4 Subsistema 5 - Carregador e bateria 40
Testes e resultados 41
Discussão 44
6.1 Aplicativo de monitoramento 44
6.2 Projeto modular 45
Conclusões 47

Referências bibliográficas 49
Apêndice A 51
Código SIM808 51
Main 51
SIM808_methods.cpp 55
SIM808_methods.h 68
Apêndice B 69
Código Android - Aplicativo do paciente 69
Main Activity 69
ConectarDispositivo 72
EditarContato 74
EditarEndereco 75
EstadoPaciente 76
PacientesDO 78
ParametrosSeguranca 78
SMSReceiver 80
Android Manifest 80
build.gradle 81
Apêndice C 83
Código Android - Aplicativo de monitoramento 83
MainActivity 83
Android Manifest 84
build.gradle 85

Capítulo 1

Introdução

1.1 Contextualização

De acordo com o relatório da Organização das Nações Unidas (ONU)[1] de 2013 sobre envelhecimento da população, espera-se que o número de pessoas com 60 anos ou mais dobre, de 841 milhões de pessoas em 2013, para mais de 2 bilhões em 2050. Esse fenômeno é observado em praticamente todos os países do mundo como consequência da diminuição da mortalidade, ocasionando a inversão da pirâmide etária mundial.

O mesmo relatório também afirma que, globalmente, 40% das pessoas com idade acima de 60 anos vivem de maneira independente, ou seja, sozinha ou com seu cônjuge.

Acompanhado desse envelhecimento populacional, há também um aumento nos avanços tecnológicos na área da saúde. Um dos exemplos são os serviços de *mHealth* ou "saúde eletrônica", que traz consigo novas possibilidades e soluções para enfrentar diversos problemas.

De acordo com a Organização Mundial da Saúde[2], define-se *mHealth* ou *mobile Health* como práticas médicas e de saúde pública suportadas por dispositivos *mobiles* como telefones portáteis, dispositivos de monitoramento de pacientes, assistente pessoal digital (PDAs), entre outros.

Tais dispositivos podem ser utilizados para a medição de diversas estatísticas fisiológicas como, por exemplo, frequência cardíaca, pressão arterial, temperatura corporal e outros sinais vitais. Essas medições podem tanto ser transmitidas a um médico ou clínica quanto serem mostradas para o próprio paciente.

Segundo a Sociedade Brasileira de Geriatria e Gerontologia (SBGG), a diminuição da capacidade funcional é a maior causa da perda de independência do idoso[3], algo que, na velhice, está intimamente ligada com a qualidade de vida[4].

---

[1] World Population Ageing 2013 .
[2] mHealth: New horizons for health through mobile technologies.
[3] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 1782.
[4] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 2003.

Ainda de acordo com a SBGG, é reconhecido que, com o envelhecimento, ocorram alterações nas funções cognitivas do ser humano. Habilidades como comunicação, orientação topográfica, memória recente e raciocínio prático tendem a entrar em declínio desde os 50 anos, sendo mais comum, porém, ao passar dos 70 anos de idade[5].

O trabalho aqui apresentado se propõe a desenvolver, como prova de conceito, um dispositivo *mobile* que auxilia idosos confusos e/ou perdidos fora de casa. A principal informação que será utilizada para reconhecer esses casos será a própria localização do paciente através da tecnologia GPS.

Tal dispositivo busca promover o autocuidado nos idosos, o que ajuda mantê-los conectados e proativos, com um controle maior sobre suas vidas, além de aliviar a sobrecarga na família e outros cuidadores[6].

1.2 Revisão bibliográfica

1.2.1 Histórico

O trabalho proposto se encontra dentro do escopo de *mHealth* ou *mobile Health*, que se tratam de serviços de assistência médica que utilizam tecnologias mobile e wireless para monitorar a saúde de um paciente.

Com base nisso, foi realizada uma pesquisa no campo científico para entender melhor os avanços já conquistados nessa área. Em 2013, (Chiarini et al. 2013) publicaram um artigo de revisão da literatura sobre tecnologias encontradas nesse segmento entre 2008 e 2012. Nesta pesquisa, foram encontrados 42 trabalhos bastante relevantes ao tema de *mHealth*, com poucos trabalhos relacionados ao tema encontrados no período anterior a 2008.

Segundo a SBGG, a gerontologia, a ciência que estuda o envelhecimento[7], ainda é um campo novo de estudo. As pesquisas nessa área se iniciaram nas universidades apenas na década de 90, enquanto que a graduação em gerontologia surgiu nos anos 2000. É uma área que ainda está em fase de consolidação no Brasil e se encontra distante de atender às necessidades da população[8].

---

[5] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 2025.
[6] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 1781.
[7] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 69.
[8] Tratado de Geriatria e Gerontologia - 3ª Edição, p. 215.

Atualmente, existem figuras conhecidas que ajudam nos cuidados ao idoso, como casas de apoio ou cuidadores, que podem ser tanto familiares quanto um terceiro contratado para tal. Entretanto, tecnologias e serviços focados na terceira idade ainda estão para surgir e serem aprimorados, seja através da especialização de profissionais da área ou do incentivo a novas tecnologias como a proposta nesse trabalho.

1.2.2 Estado da arte

A revisão de literatura feita por (Chiarini et al. 2013) trouxe resultados importantes para entender os avanços obtidos em relação a dispositivos *mHealth*. A metodologia utilizada pelos pesquisadores para a busca de artigos publicados foi a seguinte: inicialmente, o termo "*mobile health*" foi procurado em diversas bases científicas. Os 100 primeiros resultados foram analisados e os termos relacionados a "*mobile health*" foram listados e ranqueados de acordo com a sua relevância. Após identificar os melhores termos, a pesquisa foi refeita adicionando os termos "*chronic*" e "*elderly*" às palavras-chave anteriores, a fim de encontrar resultados relacionados a doenças crônicas e idosos. Ao final da busca, 2055 artigos diferentes foram encontrados. A última pesquisa foi feita nas seguintes bases de dados online: SpringerLink; ScienceDirect; Wiley InterScience; Liebert Online; Journal of Telemedicine and Telecare; Scirus; IEEE Xplore; ACM Digital Library; Cite Seer e Google Scholar.

Os 2055 artigos selecionados foram obtidos utilizando-se apenas das ferramentas de busca e, portanto, (Chiarini et al. 2013) seguiram sua pesquisa filtrando os resultados e analisando se eles atendiam de fato as seguintes condições: publicados entre 2008 e 2012 e já desenvolvidos e implementados; a arquitetura desenvolvida é móvel, ou seja, possui autonomia para funcionar sem estar conectada a uma fonte de energia fixa; e se eram específicos para idosos e pessoas com doenças crônicas. A filtragem foi concebida analisando os artigos na seguinte ordem: título, resumo e texto completo. Ao final da filtragem, 42 artigos relevantes foram encontrados para as premissas definidas. Tal acervo foi então estudado pelos autores desse trabalho para entender melhor as aplicações já disponíveis.

Dentre os trabalhos encontrados na revisão, o que chama mais atenção por similaridade com o tema proposto foi o desenvolvido por (de Jager et al. 2011) na

Universidade de Southampton. Os autores propuseram um sistema chamado DejaView cujo objetivo era utilizar processamento de imagem para ajudar um paciente com problemas de memória a relembrar pessoas, atividades recentes, tarefas programadas, lugares e objetos de acordo com o ambiente ao seu redor e do contexto. Para isso, o sistema operava: através de seus sensores, o dispositivo identificava alguma nova interação com o usuário por meio de um áudio captado pelo microfone, por exemplo, e tirava uma foto que era passada via Bluetooth ao celular. Em seguida, o celular fazia o upload da foto para um serviço de reconhecimento facial online e então enviava um feedback ao celular do usuário. Após a implementação do dispositivo, apenas testes na área de reconhecimento facial foram realizados e o tempo médio total para o usuário receber o feedback foi de 20,1 segundos.

O trabalho de (de Jager et al. 2011) foi o único da revisão que, assim como o proposto nesse trabalho, decidiu atacar o tema da perda de memória. Entretanto, outros focos foram encontrados em maior número no acervo estudado.

A aplicação mais encontrada foi em relação a quedas de idosos, uma das maiores preocupações para uma população que está envelhecendo. Portanto, muitos tentaram elaborar métodos para detecção de quedas, notificação e chamada de ajuda.

No trabalho de (Bourke et al. 2008), por exemplo, foi desenvolvido um colete especial que o idoso utilizaria por baixo da roupa, capaz de detectar uma possível queda. (Chang et al. 2011) também se utilizou de sensores posicionados próximo ao centro de gravidade do corpo (região da cintura) e em ambos os tornozelos. Os sensores inerciais utilizados coletavam os passos do idoso e enviavam os dados através de um transmissor wireless. Posteriormente, o celular utiliza-se de algoritmos para determinar um padrão de movimentação do usuário e do seu centro de massa, buscando também determinar se ele se encontra em um terreno diferente, como uma descida, para diminuir os erros de detecção de queda.

Outra abordagem do problema seria aproveitar os sensores inerciais já presentes em aparelhos celulares para a coleta de dados. Com isso, pode-se aproveitar muitas outras funcionalidades do celular, como GPS e a possibilidade de mandar mensagens e realizar ligações, para contatar um sistema emergencial ou uma pessoa de confiança. Foram criados protótipos de sistemas, como o PerFallD de (Jiangpeng Dai et al. 2010), que utiliza uma interface simples para o idoso ao mesmo tempo que detecta quedas. Apesar de funcionar apenas para aparelhos Android G1 e poucos testes terem sido realizados, a pesquisa aborda como a localização do

telefone celular no idoso, seja no bolso da camiseta, na calça ou na cintura, afeta a detecção de queda e a adaptação do algoritmo.

O SensorFall de (Lopes, Vaidya, and Rodrigues 2013) possui a mesma ideia do PerFallD, mas valendo-se de um sistema de calibração dos sensores para melhorar a obtenção de dados. Já (Yavuz et al. 2010) implementou um sensor de quedas no celular, diferenciando-se dos demais pela utilização de algoritmos robustos para determinar uma queda, uma vez que geralmente define-se um limiar para o acelerador que, quando ultrapassado, detectaria uma queda.

Além da detecção de quedas, o monitoramento do idoso muitas vezes também é de interesse para sua saúde e bem-estar. Com a utilização de um bracelete inteligente, (Postolache et al. 2011) conseguia medir sinais vitais importantes para o monitoramento de pacientes e idosos. Utilizando-se de comunicação Bluetooth com um smartphone, os pesquisadores adquiriam e mostravam em gráficos de fácil interpretação dados como batimentos cardíacos e pressão arterial.

Já em relação aos dispositivos *wearables*, diferentes formatos foram encontrados na pesquisa. (Wu et al. 2009), por exemplo, construíram um dispositivo em formato de anel que media o batimento cardíaco e temperatura corporal. (Bourouis and Feham 2011) propuseram um sistema que funcionava com “wireless body area network” (WBAN), que são sensores sem fio posicionados em áreas específicas do corpo, para medir pulso e oxigenação. Em ambos os casos, a arquitetura também era de três níveis que passava por um dispositivo sensorial, um smartphone e um serviço em nuvem para tratar os dados e realizar as ações necessárias. Nesse trabalho, a mesma arquitetura será utilizada, porém, procura-se realizar um dispositivo que seja o menos invasivo possível ao paciente e utilizando o serviço em nuvem apenas para armazenamento de dados.

Ainda em relação ao dispositivo, o trabalho proposto pretende utilizar principalmente sensores de geolocalização. Assim, será necessário possuir um módulo que tenha serviços de GPS, para localização, e GSM, para comunicação. Neste mesmo caminho, (Bharavi and Sukesh 2017) propõem um dispositivo de rastreamento com base nessas duas tecnologias. No trabalho de (Bharavi and Sukesh 2017) foi utilizado o módulo SIM808 da SIMCom, porém o autor também comenta sobre os módulos SIM900, também da SIMCom, e o M66, da Quectel, que são bastante utilizados no mercado.

Um trabalho bastante interessante e recente que também utiliza tecnologia GPS e GSM na área de *mobile Health* foi o desenvolvido por (Ahmed, Hassan, and Rashid 2017). Os autores propuseram um relógio inteligente movido a energia solar que servia para monitoramento e rastreamento de crianças com autismo. Assim como (Bharavi and Sukesh 2017), o módulo de GPS e GSM utilizado foi o SIM808. O microcontrolador utilizado no projeto foi o Arduino Nano e, além de informar localização, o relógio ainda media batimento cardíaco, temperatura corporal e possuía um botão de emergência.

Capítulo 2

Objetivos

O objetivo do trabalho é, como uma prova de conceito, criar um dispositivo de geolocalização que consiga identificar situações em que o paciente esteja em algum local desconhecido, podendo estar perdido ou confuso, e a partir disso, verificar o seu estado e tomar uma decisão de acordo com a resposta. Tal dispositivo será como um botão do pânico não físico, tentando identificar situações de perigo mesmo quando o paciente não está consciente disso e acionando ajuda.

A forma de comunicação escolhida para verificar se o paciente necessita de socorro é através do *smartphone*. Em vista disso, para facilitar essa comunicação, o trabalho também prevê um aplicativo em Android com uma interface simplificada e intuitiva para o paciente. O aplicativo também poderá ajudar a definir parâmetros de segurança como quais contatos que deverão ser acionados no caso de uma emergência ou a determinação de uma zona de segurança.

Deverá ser entregue também um protótipo com sensores de localização GPS e módulo GSM que irá coletar a posição do idoso e enviar notificações a um smartphone. O seu controle dependerá de um microcontrolador que será responsável pelo processamento dos dados. Como dito anteriormente, o que será entregue não é um produto final, portanto questões como o encapsulamento do protótipo para algo que o idoso possa vestir ficam em segundo plano.

Como objetivo secundário está previsto o armazenamento em nuvem do histórico de localização do usuário para aplicações futuras. Esses dados podem ser importantes tanto para o paciente quanto para seus médicos ou familiares.

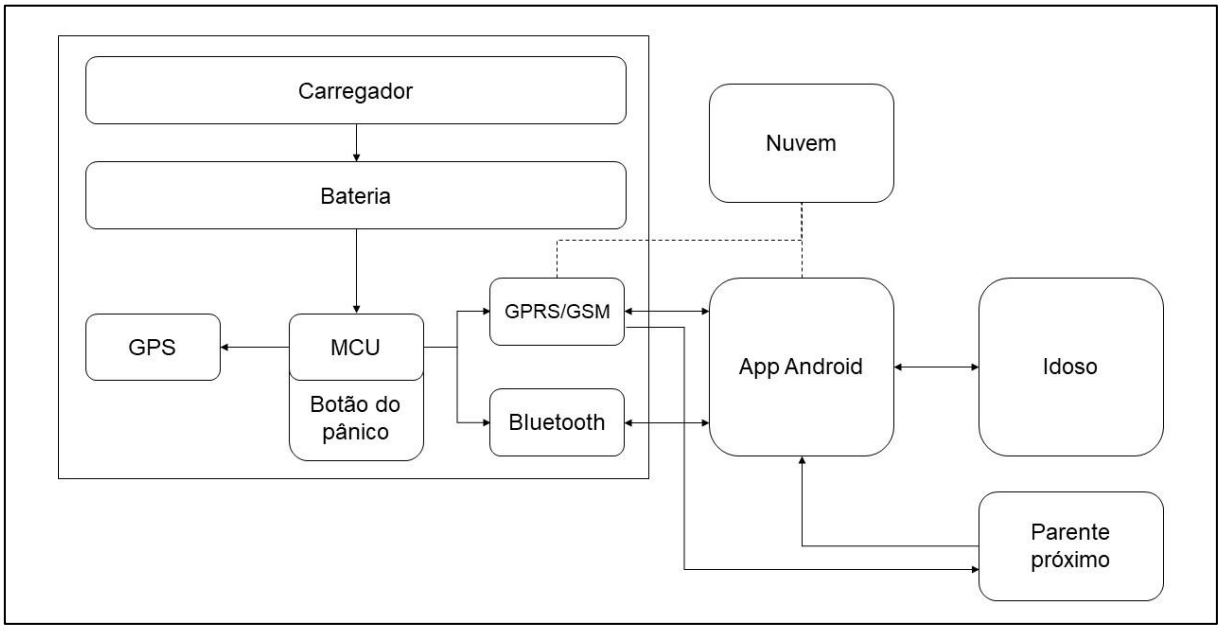
Capítulo 3

Projeto básico

3.1 Análise mecatrônica

A partir do problema proposto, foram definidos os subsistemas do projeto e definido uma metodologia, mostrada na Figura 3.1. São eles: um módulo de localização GPS, um componente com tecnologia GSM para comunicação com o idoso ou um contato emergencial como um parente próximo, um módulo Bluetooth que serve de via secundária de comunicação e um microcontrolador para o controle de cada um desses componentes e a identificação das situações de perigo pelo botão do pânico não físico, além de uma bateria para alimentação dos componentes e o seu devido carregador.

Figura 3.1 - Arquitetura proposta



O módulo GPS recebe as coordenadas do idoso e envia os dados para um microcontrolador que deve fazer o processamento e, no caso de alguma anomalia, utiliza o módulo GSM para enviar mensagens ao celular do paciente ou de um contato emergencial. O Bluetooth funciona como segunda forma de comunicação e é utilizado também para redundância na conexão com o *smartphone* do idoso.

Como dito anteriormente, um aplicativo para celular em Android poderá facilitar a comunicação com o usuário, além de ajudar a estabelecer parâmetros de segurança, como a definição dos contatos confiáveis. Esses parâmetros poderão ser configurados tanto pelo próprio paciente quanto por um familiar caso o usuário não esteja habituado ao uso de *smartphones*.

Por fim, como objetivo secundário, está o armazenamento do histórico de localização em um banco de dados na nuvem, que pode ser realizado tanto pelo microcontrolador quanto pelo aplicativo Android.

3.2 Parâmetros e requisitos

O problema principal que se busca resolver é do idoso que está perdido ou confuso na rua, sem saber o caminho que está tomando. Para resolver o problema é necessário vigiar o idoso para assegurar de que ele não está desnorteado. No Capítulo 1 Subseção 1.2.2 sobre o estado da arte, existem produtos disponíveis no mercado que o idoso carrega e aciona um botão do pânico, que quando apertado entra em contato com algum familiar ou um atendimento de emergência.

O problema dessa solução é que o botão do pânico é algo físico que o idoso deve acionar em caso de alguma emergência. Entretanto, muitas vezes o paciente não reconhece que está perdido. Outro ponto é o fato de que alguns dispositivos são grandes e anormais para ele. Como discutido no Capítulo 1 Seção 1.1, o objetivo do projeto é devolver a autonomia para o idoso, realizando, portanto, as operações da maneira menos perceptível possível. Logo, o ideal seria uma forma de botão do pânico não físico, onde a detecção e acionamento sejam independentes do idoso, acontecendo de forma automática.

Para isso, é preciso saber a localização do idoso e, quando identificado uma situação de problema, tentar se comunicar com o paciente para verificar se ele realmente se encontra em perigo. Caso a tentativa de comunicação não surta efeito, alguém de responsabilidade deve ser acionado para ajudá-lo o mais rápido possível.

Além das funcionalidades acima, como já discutido, o dispositivo deve ser capaz de tomar decisões rapidamente e conseguir acionar de maneira eficaz tanto o idoso quanto um contato de urgência. Ele deve ter também certa autonomia, com o seu funcionamento integral enquanto o idoso está fora de casa, e ser capaz de operar de forma não invasiva para o idoso.

3.3 - Subsistemas

3.3.1 - Unidade de Controle e Bluetooth

Visando a autossuficiência do produto final, a potência consumida pela unidade de controle deve ser a menor possível enquanto mantém o poder de processamento. Um microcontrolador é um circuito integrado que possui núcleo de processamento, memória e periféricos programáveis de entrada e de saída capaz de suprir essas necessidades consumindo menos do que um *Single Board Computer*, por exemplo. Existem numerosos fabricantes e modelos no mercado, como por exemplo o ARM Cortex-M, Atmel AVR ou Intel 8051, além das várias placas e kits como a plataforma Arduino e Microchip PIC Starter Kit. Logo, para facilitar a escolha, definiu-se que o microcontrolador do projeto deveria atender às seguintes principais especificações:

1. Periféricos: é importante que o microcontrolador tenha WiFi e Bluetooth para facilitar a comunicação com um aparelho *smartphone* ao mesmo tempo que provê uma autonomia para o trabalho, com foco para a ideia de “Internet das Coisas”;
2. Velocidade e capacidade de processamento: devido à urgência de algumas situações e a tomada de decisões realizadas pelo próprio programa embutido nele, o microcontrolador deve apresentar um excelente nível de processamento;
3. Memória: o microcontrolador deve ser capaz de armazenar uma série de informações sobre o usuário a fim de tomar as melhores decisões em casos emergenciais;
4. Tamanho e encapsulamento: mesmo buscando apenas uma prova de conceito, o tamanho e o encapsulamento do microcontrolador é algo que também pode ser levado em consideração, pois o dispositivo deve ser o menor possível para que impacte pouco no cotidiano do paciente.
5. Custo: futuramente, o dispositivo pode se tornar um produto comercializável, assim o seu custo também é um fator importante.

Infelizmente, a utilização de um módulo WiFi muitas vezes implica no acoplamento de um sistema com essa funcionalidade com o microcontrolador através

de uma conexão externa. Por exemplo, para se utilizar um Arduino com conexão WiFi é necessário, na maioria das vezes, comprar um adaptador com essa função que deve ser ligado à placa, o que encarece muito o produto.

Logo, os periféricos se mostraram como uma das especificações mais essenciais na escolha do microcontrolador. Durante a pesquisa, o microcontrolador que apresentava tais funcionalidades e o maior número de referências para consulta foi o ESP8266, indicado na Figura 3.2. Porém, em uma pesquisa mais detalhada sobre o microcontrolador, encontrou-se que a sua fabricante (Espessif) produziu uma versão mais recente e com maior capacidade de processamento: ESP32, apresentado na Figura 3.3.

Figura 3.2 - ESP8266 com dimensões de 25mm x 48mm

Imagem extraída de https://bit.ly/2JKB2QN

Figura 3.3 - ESP32 com dimensões 28mm x 58mm

Imagem extraída de https://bit.ly/2MBuh5P

Essa nova versão tem WiFi integrado, mais sinais GPIO e módulo Bluetooth 4.2 (BLE). Assim como seu predecessor, o microcontrolador possui integração com o

Ambiente de Desenvolvimento Integrado (IDE) Arduino e, portanto, suporte a muitas bibliotecas e facilidade de programação.

Fazendo uma comparação simples entre o Arduino UNO R3, o ESP8266 e o novo ESP32, é possível observar na Tabela 3.1 de que se trata de um microcontrolador bem mais potente.

Tabela 3.1 - Comparativo entre ESP32, ESP8266 e Arduino R3

	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16MHz
WiFi	Sim	Sim	Não
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

Tabela extraída de https://bit.ly/2t8qhRf

O ESP32 possui dois cores além de muito mais memória, tanto FLASH comparado ao Arduino R3 quanto RAM comparado ao ESP8266 e Arduino R3. Ele ainda é capaz de controlar muito mais sensores devido a elevada quantidade de portas I/O, o que pode ser interessante caso a pesquisa seja continuada. Possui mais interfaces de comunicação, o que também facilita a sua integração com outros componentes. Por fim, pensando no encapsulamento, o ESP32 é mais compacto que o Arduino R3 e já possui módulos WiFi e Bluetooth inclusos. Por se encaixar nas especificações definidas e ter um custo relativo baixo, o ESP32 foi escolhido para o desenvolvimento do trabalho.

3.3.2 - GPS e GPRS/GSM

No estudo do estado da arte de outros dispositivos, percebeu-se que é bastante comum o uso de módulos que possuem tecnologias GPS e GSM integrados. Tal prática ajuda no encapsulamento e reduz eventuais problemas de comunicação entre módulos separados.

Como discutido no Capítulo 2, os módulos mais utilizados são os da SIMCom (SIM900 e SIM808) e da Quectel (M66). Na pesquisa por componentes, esses se mostraram como as melhores soluções no quesito eficiência e custo, além de suporte técnico e ampla consulta de bibliotecas e outros trabalhos.

Porém, ambos SIM900 e M66 apresentam apenas o módulo GSM/GPRS para comunicação, fazendo com que a localização por GPS seja feita através de outro componente. Por sua vez, o módulo SIM808 combina o módulo quad-band (banda de frequência) GSM/GPRS apresentado pelos outros dois componentes com a tecnologia GPS.

Além de tal facilidade, os módulos da SIMCom foram os mais encontrados no mercado, o que facilitaria a sua compra e a rápida obtenção de tal componente para começar os testes o quanto antes. Logo, por apresentar ambas funcionalidades e ter alta disponibilidade, o SIM808 foi escolhido para localização e envio de SMS.

3.3.3 – Botão do pânico não físico

Para o sucesso do trabalho, define-se que o projeto deveria ser capaz de identificar situações em que o paciente possa estar em perigo para o acionamento de ajuda. Para tal, o protótipo deve ser capaz de delimitar uma zona ao redor da casa do idoso que é considerada segura, onde ele geralmente frequenta.

Inicialmente, a zona é definida como um simples círculo em torno da residência do usuário, delimitando uma região. Ao sair dessa zona, o idoso deve ser contatado para confirmar que ele realmente está perdido. Caso esse contato não surta efeito, deve-se acionar um contato registrado, informando-o de que o idoso deixou a zona de segurança. Porém, essa zona pode ser atualizada conforme o paciente utiliza o dispositivo através de um mapa dinâmico, que levaria em conta o histórico de lugares que o idoso frequenta.

Todos os parâmetros definidos acima deverão ser configurados através de um aplicativo para celular em Android por um parente ou pessoa de confiança do idoso. O aplicativo também funcionaria como um meio de comunicação com os usuários.

3.3.4 – Aplicativo Android

Conforme descrito na análise mecatrônica, quando alguma anomalia for identificada, será realizada uma tentativa de comunicação com o paciente. A forma de comunicação será feita através de um *smartphone* que o idoso deverá levar consigo quando sair de casa. Para facilitar a comunicação, foi proposto um aplicativo para celular, uma vez que nem todos os pacientes estão ambientados com o uso de *smartphones.* O aplicativo utiliza recursos de fácil entendimento, como um layout simples na tela do celular, que pode ser de grande benefício para os usuários.

Em relação ao sistema operacional, o Android foi escolhido por ser o mais comum dentre os disponíveis no mercado[9]. Para a programação do aplicativo, a plataforma Android Studio foi definida por ser a plataforma oficial da Google para a criação de aplicativos no sistema Android. Por fim, os sensores de geolocalização do *smartphone* poderão ser utilizados como redundância para determinar a localização do paciente.

3.3.5 - Carregador e bateria

Como o produto final deve ser algo que o paciente consiga carregar com facilidade e durante longos períodos de tempo, a alimentação dos componentes eletrônicos é algo que deve ser levado em consideração. Como abordado na escolha dos dispositivos nas Subseções anteriores 3.5.1 e 3.5.2, o consumo energético de cada módulo foi um dos fatores utilizados para eleger os aparelhos.

Além de ter baterias capazes de alimentar os componentes, seria ideal que o produto final também possuísse uma forma de carregamento, em que o paciente carregue o aparelho quando estiver em sua residência. Com isso, pode-se manter o aparelho funcional sem a necessidade de pilhas e com o simples carregar das baterias enquanto ele não estiver sendo usufruído.

[9] https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

Porém, como o trabalho proposto é uma prova de conceito, ainda que o gasto energético tenha sido um dos fatores levados em consideração, a preocupação com as baterias e o carregador não serão abordados.

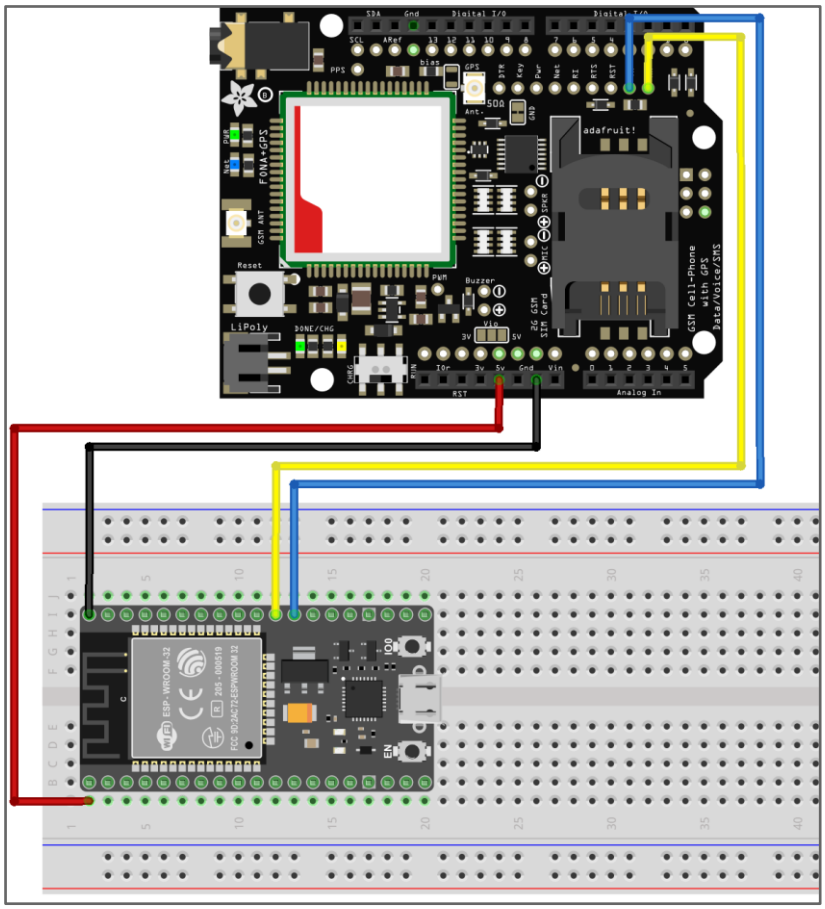
Capítulo 4

Projeto detalhado

4.1 Subsistema 1 e 2 - Unidade de controle, Bluetooth, GPS e GPRS/GSM

Como discutido anteriormente, escolheu-se o módulo SIM808 e o microcontrolador ESP32 para envio e recebimento de mensagens, além da localização GPS e seu controle para identificação de situações de perigo. A comunicação entre o microcontrolador e o módulo de GPS/GPRS é realizada por meio de uma comunicação serial, como apresentado no esquemático da Figura 4.1.

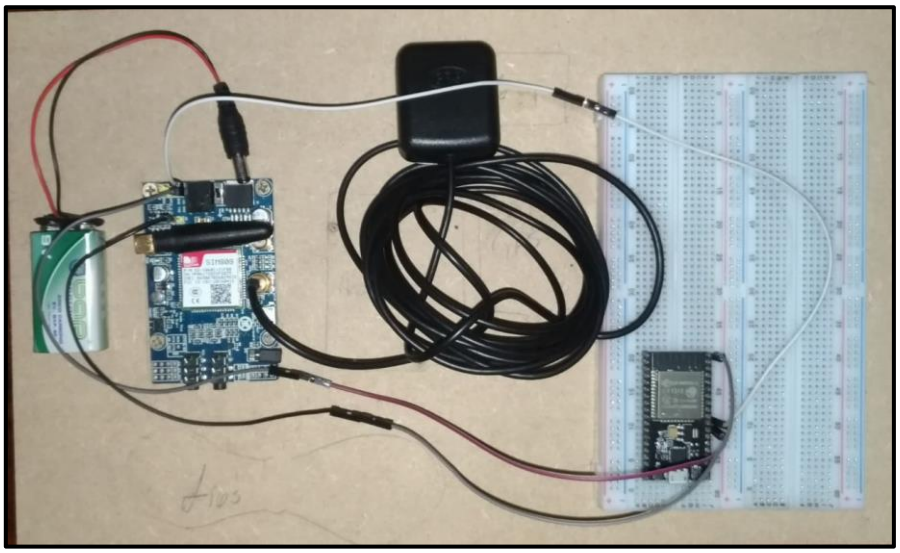
Figura 4.1 - Esquemática da ligação SIM808 e ESP32



Onde os fios azul e amarelo realizam a comunicação serial TX-RX, utilizando os pinos GPIO 16 e 17 do microcontrolador, e o fio preto compartilha o terra entre os componentes. O fio vermelho é facultativo uma vez que é a alimentação de 5V do SIM808 para o microcontrolador, podendo ser substituído por uma alimentação independente na própria protoboard, por exemplo.

Para a integração e facilidade no transporte dos componentes, montou-se uma maleta para carregar o protótipo do trabalho. O esquemático da figura 4.1 foi construído em um pedaço de madeira, colando-se a protoboard com o microcontrolador ESP32 e parafusando o módulo SIM808, como apresentado na figura 4.2.

Figura 4.2 – Protótipo de testes com SIM808 e ESP32



Na figura 4.2, nota-se a pilha de 9V utilizada para ligar o módulo SIM808, além do dispositivo GPS e os três fios de comunicação TX-RX e terra entre o módulo e o microcontrolador ESP32. O microcontrolador é conectado com o computador por um cabo USB-micro para os testes, sendo essa sua alimentação de energia.

Para a comunicabilidade entre os dispositivos, procedeu-se uma busca online por bibliotecas que pudessem fazer a leitura da localização atual pelo SIM808, junto com a possibilidade de envio de SMS controlados pelo ESP32. Apesar de encontradas bibliotecas prontas para a comunicação serial entre o módulo SIM808 e o Arduino com tais funcionalidades separadas, por exemplo, nenhuma biblioteca foi encontrada para essas funções utilizando o ESP32.

Em uma primeira tentativa, empenhou-se em integrar e adaptar as bibliotecas encontradas para Arduino para funcionar com o microcontrolador ESP32. Tais tentativas obtiveram pouco sucesso e, portanto, escreveu-se uma biblioteca nova capaz de implementar as funcionalidades necessárias. Para instalação do microcontrolador ESP32 no IDE do Arduino, seguiu-se o tutorial de configuração da Arduino e Cia.[10] Empregou-se a última versão disponível do software do Arduino IDE 1.8.7 no período da elaboração desse trabalho. O código utilizado, bem como a biblioteca criada pelos autores desse trabalho, podem ser encontrados no anexo A ou pela página o GitHub: https://github.com/LeonardoCRamos/TCC_ESP32_SIM808.

Inicialmente, o microcontrolador verifica se a comunicação serial com o módulo está funcionando e então define as funcionalidades do celular e o formato que o SMS será enviado e recebido. Com isso, ele certifica-se de que o GRPS e o GPS estão funcionando, ou seja, que o chip SIM está inserido e pronto, para então ligar a antena GPS. Com isso, o microcontrolador deleta todos os SMS no chip por uma questão de segurança e facilidade, garantido memória e facilidade na indexação das mensagens.

Uma vez realizada a comunicação serial entre o microcontrolador e o módulo SIM808, deve-se integrar o aplicativo Android para definição de parâmetros como número de emergência e localização da casa do paciente. Definiu-se, portanto, que o microcontrolador iria se conectar via bluetooth com o celular para fazer o cadastro de informações básicas do usuário. Com a conexão estabelecida, o aplicativo enviaria um código UUID (Universally Unique Identifier) já definido no microcontrolador para certificar-se de que ele está conectado com o módulo certo.

Validada a chave, o microcontrolador receberia do aplicativo o número do paciente, um número de emergência e as coordenadas de latitude e longitude da casa do usuário. Em retorno, o aplicativo reconhece o número do chip presente no módulo SIM808.

Se ele estiver fora desse raio de segurança, o microcontrolador deve enviar um SMS perguntando se o usuário está bem a fim de evitar falsos positivos. Se a resposta for negativa ou não houver resposta, é enviada uma mensagem com a localização atual do usuário para o contato de emergência cadastrado. O diagrama de estados do código está representado nas Figuras 4.3 e 4.4.

[10] https://www.arduinoecia.com.br/2017/11/modulo-esp32-wifi-bluetooth-ide-arduino.html

Figura 4.3 - Diagrama de estados do set-up inicial

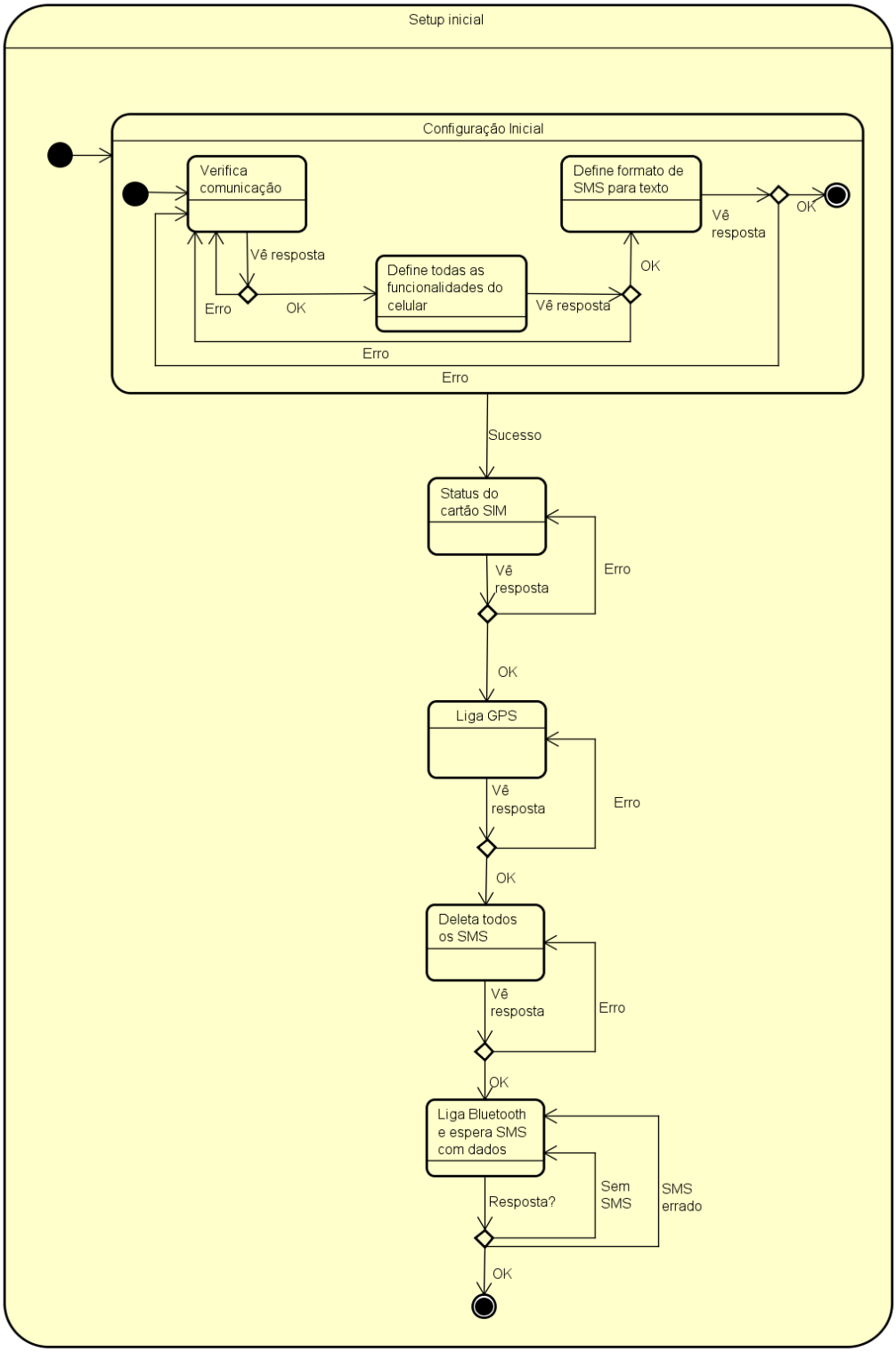
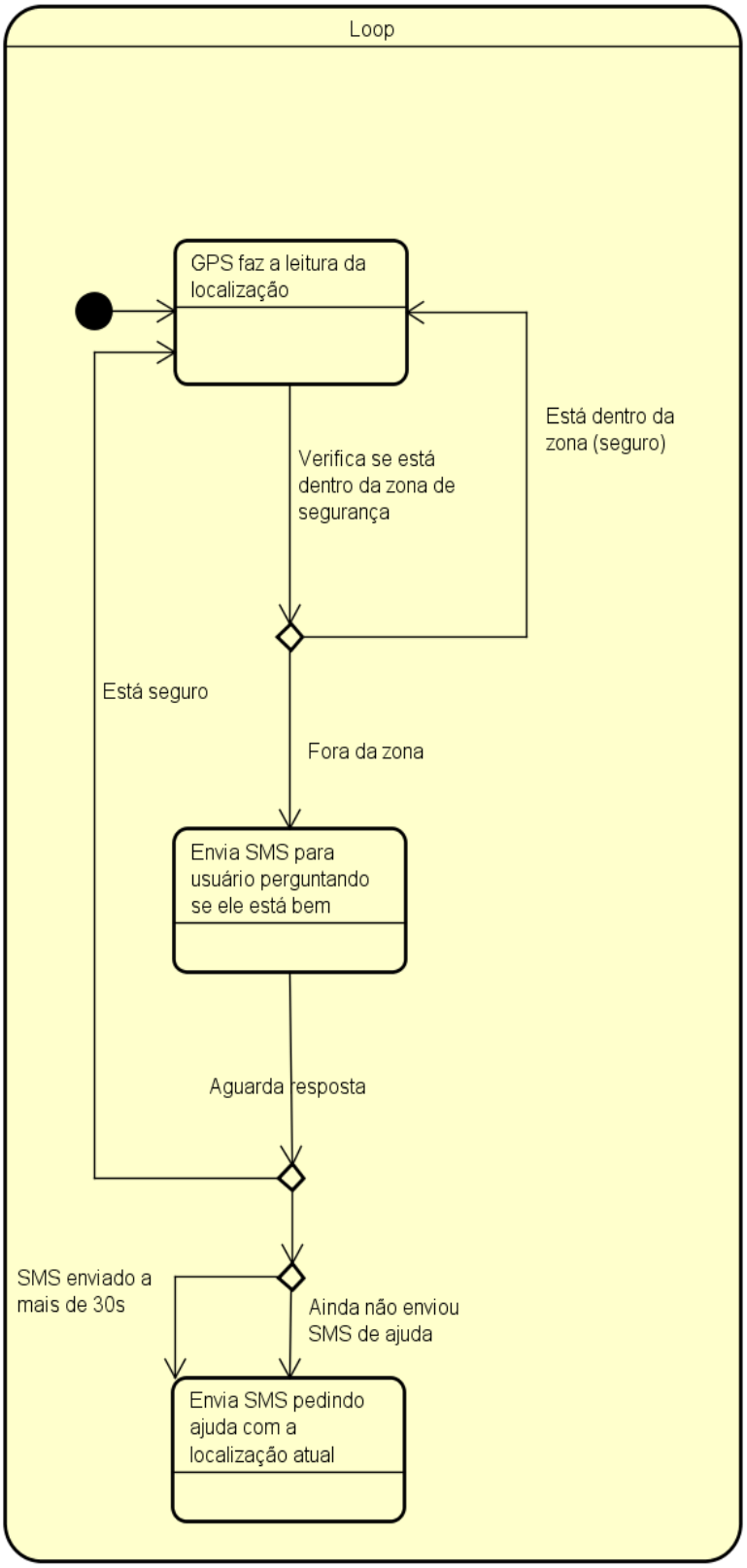


Figura 4.4 - Diagrama de estados do loop



4.2 Subsistema 3 – Botão do pânico não físico

A forma escolhida para delimitar uma zona de segurança para o paciente foi um raio de 0.5km ao redor da residência do paciente. Com essas informações, o módulo faz a leitura da localização atual do paciente e realiza um cálculo para definir a distância até a sua casa. A Comissão Federal de Comunicações dos Estados Unidos (em inglês: Federal Communications Commission - FCC) define a seguinte fórmula[11] para calcular a distância (D) entre dois pontos cujo par (latitude, longitude) é dado por (ϕ_1, λ_1) e (ϕ_2, λ_2):

$$D = \sqrt{(K_1 \Delta\phi)^2 + (K_2 \Delta\lambda)^2}$$

Onde

$$K_1 = 111{,}13209 - 0{,}56605\cos(2\phi_m) + 0{,}0012\cos(4\phi_m)$$

$$K_2 = 111{,}41513\cos(\phi_m) - 0{,}09455\cos(3\phi_m) + 0{,}00012\cos(5\phi_m)$$

$$\Delta\phi = \phi_2 - \phi_1$$

$$\Delta\lambda = \lambda_2 - \lambda_1$$

$$\phi_m = \frac{\phi_1 + \phi_2}{2}$$

Com isso, o microcontrolador calcula a distância do usuário até a sua casa e verifica se ele está fora do raio de segurança pré-definido.

Conforme comentado no Capítulo 3 Subseção 3.3.3, uma forma melhor de delimitar essa zona de segurança seria através de um mapa dinâmico, em que as bordas dessa zona fossem atualizadas de acordo com a utilização do dispositivo pelo paciente. Isso se deve pois o idoso pode se encontrar em um local conhecido, a casa de um parente por exemplo, fora do raio limite definido. O inverso também pode ocorrer, o paciente fica perdido mesmo dentro da área segura. Tal abordagem elevaria a complexidade do projeto, porém foi feita uma base de algumas funções que poderão ser utilizadas para realizar este mapa dinâmico em um trabalho futuro.

[11] https://www.gpo.gov/fdsys/pkg/CFR-2016-title47-vol4/pdf/CFR-2016-title47-vol4-sec73-208.pdf

O mapa dinâmico seria moldado de acordo com um histórico de localizações fornecidas pelo usuário ao longo do tempo. Assim, foi incluída no projeto do aplicativo do *smartphone* uma função que registra periodicamente a localização atual do paciente. Como esse histórico pode crescer de maneira muito rápida e, portanto, prejudicar o armazenamento desses dados no *smartphone*, definiu-se que as informações sobre localização do usuário seriam enviadas a um banco de dados na nuvem.

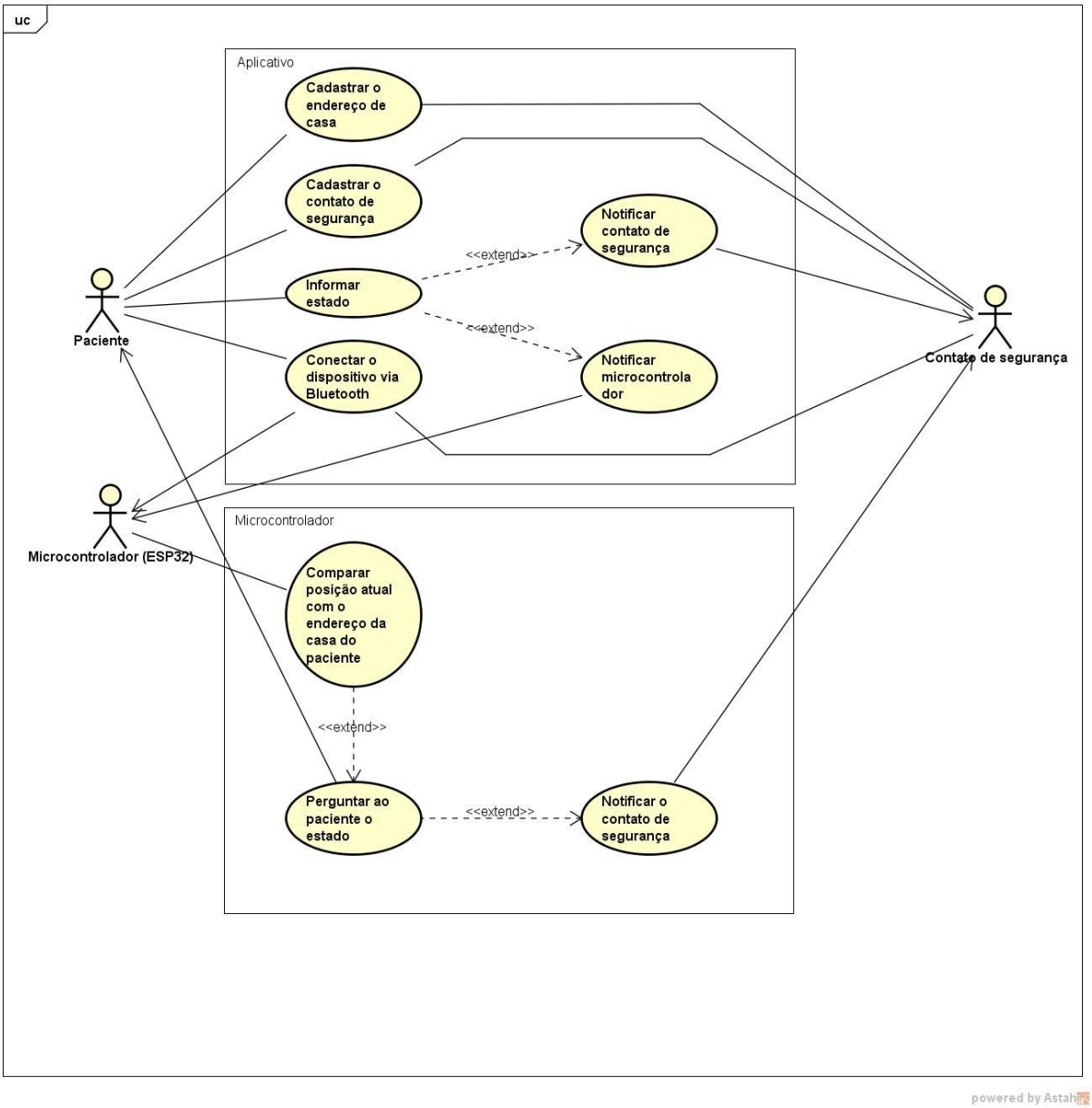
O serviço escolhido para o armazenamento desses dados foi DynamoDB da Amazon Web Services (AWS), pois trata-se de uma solução com alta escalabilidade e que possui bibliotecas para a integração com Android. Além disso, a AWS possui várias ferramentas para análise e tratamento de dados que pode ser útil para o processamento do mapa dinâmico.

De posse do histórico de localização, foi feita uma ferramenta de visualização desse histórico através de um mapa de calor em um segundo aplicativo. Tal ferramenta pode ser útil para a família ou médico do paciente e se encontra descrita com mais detalhes no Capítulo 6.

4.3 Subsistema 4 – Aplicativo Android

Para o detalhamento deste subsistema, foram produzidos dois diagramas para facilitar o entendimento sobre as ações do aplicativo. Na Figura 4.5, encontra-se o diagrama de casos de uso e, em seguida, uma lista que detalha cada caso de uso e os seus atores.

Figura 4.5 - Diagrama de casos de uso para o subsistema projetado



Detalhamento dos casos de uso do aplicativo:

- Cadastrar o endereço de casa:

  Realiza o cadastro do endereço de casa do paciente. Após a inserção do endereço, ele é transformado em coordenadas geográficas através da API Geocode do Google.

- Cadastrar o contato de segurança:

  Realiza o cadastro do número de celular do contato de segurança.

- Informar estado:

O paciente por conta própria ou devido a alguma notificação enviada pelo microcontrolador pode informar o seu estado. Caso ele informe que está bem, uma mensagem de SMS é enviada ao microcontrolador. Caso ele informe que precisa de socorro, uma mensagem de SMS é enviada ao contato de segurança junto com sua localização atual.

- Conectar o dispositivo via Bluetooth:

O aplicativo se conecta via Bluetooth com o microcontrolador e ambos trocam informações. Para ter certeza que o aplicativo está se conectando com o dispositivo correto, é necessário inserir no aplicativo a chave UUID correspondente ao dispositivo. O smartphone envia para o microcontrolador o número do contato de segurança e as coordenadas geográficas da casa do paciente, enquanto o microcontrolador, ao receber essas informações, envia ao aplicativo o número do seu chip de SMS.

- Notificar o contato de segurança:

Envia um SMS ao contato de segurança com um pedido de socorro e a posição atual do paciente.

- Notificar o microcontrolador:

Envia um SMS ao microcontrolador avisando que o paciente está bem.

Detalhamento dos casos de uso do microcontrolador:

- Comparar posição atual com o endereço da casa do paciente:

Na rotina do microcontrolador, ele periodicamente busca no módulo GPS a posição atual e mede a distância até a casa do paciente. Se essa distância for superior a um raio mínimo, ele pergunta o estado do paciente.

- Perguntar o estado do paciente:

Envia uma notificação via SMS para o paciente, para que ele informe se está bem ou não.

- Notificar o contato de segurança:

Em caso de uma resposta negativa do paciente ou de nenhuma resposta em um determinado período pré-estabelecido, o microcontrolador utiliza o módulo SMS para enviar uma mensagem de socorro ao contato de segurança com a posição atual medida pelo GPS.

Detalhamento dos atores:

- Paciente:

Usuário do dispositivo. Um idoso que enfrenta problema de memória, porém que gostaria de manter uma vida ativa.

- Contato de segurança:

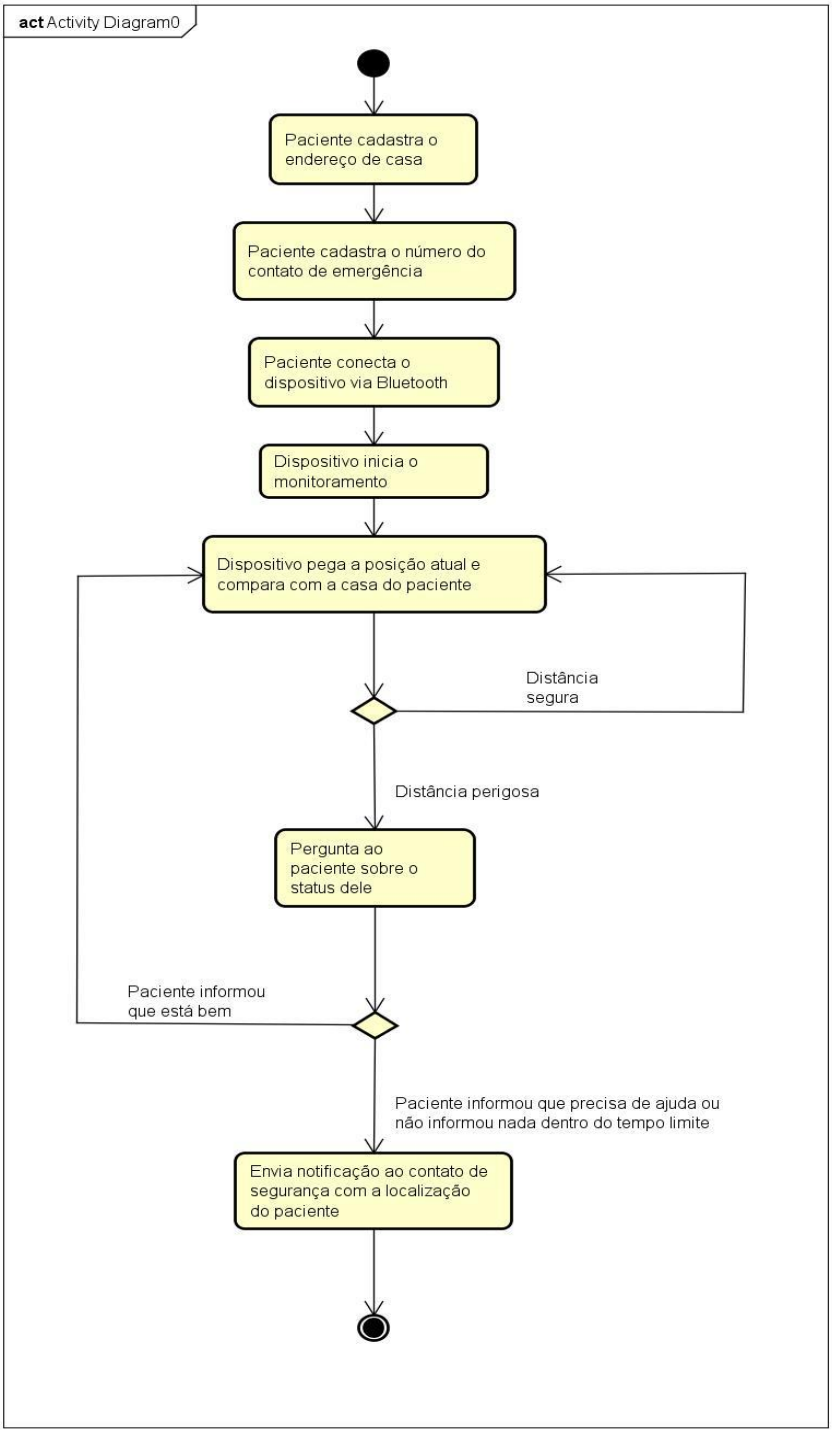
É o contato de confiança do paciente, para quem ele recorrerá no caso de uma emergência. Observe que este ator também tem acesso aos casos de uso de cadastro e de conexão com o dispositivo via Bluetooth. Isso pode ocorrer nos casos em que o paciente não possui familiaridade com o *smartphone*, situação na qual ele poderia pedir auxílio a este contato para configurar corretamente o aplicativo e o dispositivo.

- Microcontrolador:

Microcontrolador ESP32. O microcontrolador possui um módulo de Bluetooth interno. Além disso, ele está conectado ao SIM808, um componente que possui tanto uma antena GPS quanto um módulo para envio de mensagens SMS.

Além do diagrama de casos de uso também foi produzido um diagrama de atividades (Figura 4.6) para explicitar o comportamento do ciclo do dispositivo.

Figura 4.6 - Diagrama de atividades de um ciclo do dispositivo



O ciclo se inicia com o cadastro da casa do paciente e do número do contato de segurança. Como visto anteriormente no diagrama de casos de uso, essas ações podem ser feitas tanto pelo próprio paciente quanto pelo contato de segurança. Em seguida, é feita a conexão do aplicativo com o dispositivo via Bluetooth. Neste

momento ambos os componentes trocam informações para que eles possam se comunicar via SMS, além de enviar as coordenadas geográficas da casa do paciente.

Após essa troca, o dispositivo está configurado e então inicia-se o monitoramento do dispositivo. Periodicamente, o dispositivo mede a distância entre a posição atual indicada pelo módulo GPS e as coordenadas da casa do paciente. Caso essa distância seja superior a uma distância segura pré-determinada, é enviada uma mensagem de SMS ao paciente perguntando sobre seu estado. Caso contrário, o monitoramento segue normalmente. Ao perguntar o estado do paciente existem três possibilidades: se o paciente informou que está bem, o monitoramento volta a sua rotina inicial; se o paciente informou que não está bem ou não respondeu à mensagem dentro de um tempo limite, então é considerada uma situação de perigo e uma mensagem é enviada ao contato de emergência junto com a posição atual do paciente.

Também foi incluído, como dito na seção anterior, uma função que registra a latitude e longitude do paciente e envia para um banco de dados da AWS. Essa função faz parte da rotina do aplicativo independentemente da situação em que o usuário se encontre. Enquanto estiver em uso, a aplicação continuará funcionando e registrando a localização do paciente na nuvem.

Além de todas as funções citadas acima, o aplicativo também foi desenvolvido pensando no seu público alvo. Assim, procurou-se simplificar ao máximo o *layout* do aplicativo para que os usuários tenham mais facilidade ao utilizá-lo. Poucos botões foram incluídos na tela principal do aplicativo e o tamanho da fonte foi escolhida pensando em pacientes que podem ter uma visão mais desgastada. Por fim, assim que o aplicativo é instalado, já são pedidos todos os parâmetros de segurança necessários, o que pode ser bastante útil visto que muitos pacientes começarão a utilizar o dispositivo na presença de um familiar ou conhecido.

O aplicativo foi implementado utilizando o Android Studio, que é o principal *software* de programação em Android atualmente. O nível mínimo de API necessário para que o aplicativo funcione é a API 18, que corresponde a versão do Android 4.3 Jelly Bean. Segundo o Android Studio, o aplicativo funcionará em aproximadamente 95,9% dos *smartphones* e *tablets* com sistema operacional Android.

O código do aplicativo foi incluído no anexo B no trabalho e também pode ser acompanhado no seguinte GitHub: https://github.com/GTutia/TCCV1. Algumas imagens do aplicativo produzido se encontram nas figuras 4.7 a 4.9. Na Figura 4.7 é

mostrado a tela inicial do aplicativo, enquanto que na Figura 4.8 está a tela de cadastro de um contato emergencial. Já na Figura 4.9 encontra-se a tela que é exibida ao usuário quando alguma situação de perigo é identificada. Essa mesma tela pode ser acessada a partir da tela inicial através do botão “INFORMAR ESTADO”, assim o paciente pode acionar ajuda propositalmente caso tenha consciência de que precisa de socorros.

Figuras 4.7 e 4.8 – Telas iniciais e de cadastro do aplicativo Android

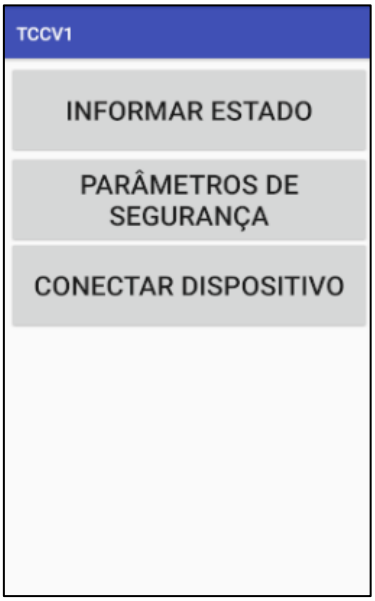
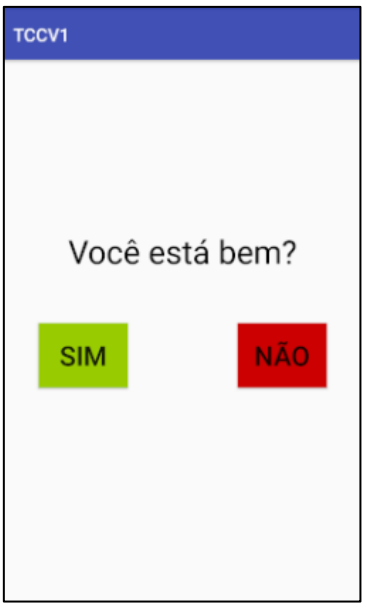


Figura 4.9 – Tela de informar estado do aplicativo Android



4.4 Subsistema 5 - Carregador e bateria

Como dito anteriormente no Capítulo 3 Subseção 3.3.5, tanto o carregador quanto a bateria não serão abordados no trabalho atual, uma vez que o trabalho se trata de uma prova de conceito. Tais assuntos, porém, foram levados em consideração nas escolhas dos componentes do trabalho.

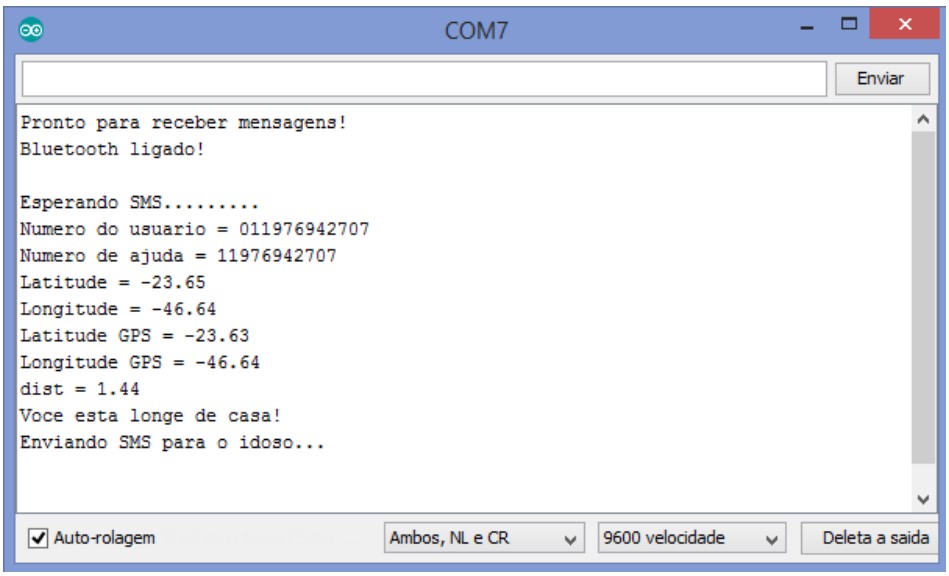
Capítulo 5

Testes e resultados

Após a implementação dos projetos detalhados, foi realizado um teste de integração entre os subsistemas. O aplicativo foi instalado em um *smartphone* e foi pareado via Bluetooth com o microcontrolador ESP32. No aplicativo foi colocado um endereço conhecido como o endereço da casa do paciente e um número também conhecido no campo de contato de emergência. Em seguida, foi feita com sucesso a comunicação Bluetooth entre ambos os dispositivos. O *smartphone* de teste recebeu o número de SMS do microcontrolador, enquanto o ESP32 recebeu as informações cadastradas, bem como o número do *smartphone* utilizado.

Na Figura 5.1 está registrada a mensagem de sucesso emitida pelo microcontrolador com as informações recebidas pelo aplicativo. O microcontrolador recebeu os números do paciente e do contato de emergência, além das coordenadas de sua residência. Tais coordenadas foram validadas, verificando que o endereço inserido no aplicativo foi corretamente convertido em latitude e longitude pelo API Geocode utilizado.

Figura 5.1 - Troca de informações entre o ESP32 e o aplicativo por Bluetooth



Realizada a troca de informações, iniciou-se o monitoramento da posição do sensor de GPS. Na Figura 5.1, está a indicação da distância medida em relação à casa do paciente pelo sensor GPS (dist). Já na Figura 5.2 encontra-se a distância em linha reta da posição do usuário em relação a casa, utilizando o Google Maps. A distância calculada pelo microcontrolador foi de 1,44 quilômetros, enquanto a distância apontada pelo Google Maps foi de 1,54 km.

Figura 5.2 - Distância do usuário até sua casa medida pelo Google Maps

Como a medição indicada foi superior ao raio limite pré-determinado de 500 metros, foi considerado que o paciente estava em situação de perigo e uma mensagem de SMS foi enviada ao *smartphone* do paciente. Passado um tempo limite determinado, uma outra mensagem de SMS foi enviada do microcontrolador ao número do contato de segurança. Assim, realizou-se um teste bem-sucedido de um ciclo completo como projetado para o dispositivo.

Buscando encontrar o erro nas leituras, decidiu-se realizar uma série de levantamento de pontos para determinar o erro entre a distância calculada pelo microcontrolador pela leitura GPS e a distância real. Para isso, a latitude e longitude dos pontos da casa e da posição atual foram mostradas como eram registrados pelo

microcontrolador. A distância calculada por ele e pelo Google Maps, considerada como a “real”, foram anotados e os seus erros e percentuais encontram-se na Tabela 5.1.

Tabela 5.1 - Comparação das medições pelo GPS e do real

Dist. Calculada (km)	Dist. Real (km)	Erro (m)	Erro % do “Real”
0,06	0,10	4,31	42,93
0,12	0,14	1,84	13,59
0,22	0,23	0,74	3,21
0,41	0,45	4,01	8,84
1,50	1,93	42,62	22,08
1,64	2,16	52,45	24,28
6,72	10,15	343,42	33,83

Percebe-se que o erro entre a medição do GPS e a distância “real” tem um valor máximo de aproximadamente 5 metros para regiões menores do que 1 km. Porém, conforme a distância aumenta, o erro se torna cada vez maior e chega a até mais de 300m.

Tanto a leitura do GPS quanto a aplicação da equação podem influenciar no cálculo dos resultados. Muitos fatores interferem na leitura do GPS, como um céu muito nublado, número de edifícios na proximidade e poluição, por exemplo. Desconsiderando-se o primeiro ponto, que se apresenta muito próximo da casa do paciente, o erro aumentou percentualmente com a distância, atingindo valores elevados para distâncias maiores. Portanto, considerando-se um erro esperado de 10%, não é recomendado um raio maior do que 1 km.

Capítulo 6

Discussão

6.1 Aplicativo de monitoramento

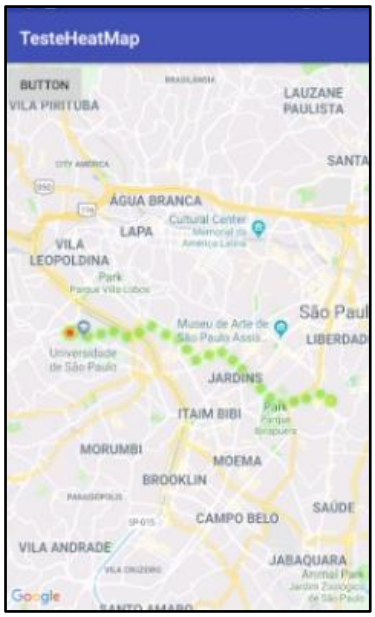
Um aspecto bastante importante do projeto estava relacionado à delimitação de zonas seguras para o paciente. Se ele se encontrasse fora do limite considerado seguro, então o dispositivo verificaria o estado do usuário. A forma mais usual e simples encontrada para delimitar essa zona foi justamente um raio ao redor da residência do paciente. Entretanto, como discutido no Capítulo 4 Seção 4.2, essa delimitação poderia ser feita através de um mapa dinâmico, que seria moldado por um histórico de localização do paciente.

Assim, foi incluída no aplicativo uma função que registra a posição do usuário e a envia para uma base de dados da Amazon Web Services. Esses dados poderão ser a base para uma futura função que altera as bordas do que se considera uma zona segura para o paciente.

De posse dessa base de dados, foi possível idealizar uma ferramenta de visualização dessas informações para identificar as áreas mais frequentadas pelo paciente. Tal informação poderá ser útil para o problema do mapa dinâmico, além de ser uma forma de monitorar o paciente, o que pode ser interessante tanto para a família quanto para um médico.

Essa ferramenta trata-se de um mapa de calor, que conta e agrupa latitudes e longitudes próximas e as identifica como uma zona bastante frequentada ou não pelo paciente. O mapa de calor foi implementado em um segundo aplicativo Android e foram utilizadas bibliotecas sobre mapas do Google para tal. Nas figuras 6.1 e 6.2, estão exemplos da utilização do mapa de calor. Já o código do aplicativo de monitoramento se encontra no Anexo C.

Figuras 6.1 e 6.2 - Testes com aplicativo de monitoramento



6.2 Projeto modular

O trabalho em questão procurou utilizar a tecnologia GPS para tentar identificar alguma situação de perigo a um idoso. A forma como é feita essa identificação pode ser melhorada como discutido na seção anterior. Entretanto, outras informações também influenciam na formação de um botão do pânico não físico, ou seja, um dispositivo capaz de identificar situações de perigo mesmo quando o paciente não tem consciência de que algo grave possa estar acontecendo.

Conforme visto na literatura, diferentes tipos de sensores são utilizados para medir atributos fisiológicos dos pacientes como batimento cardíaco, pressão sanguínea e oxigenação no sangue, além de sensores inerciais presos ao corpo do paciente ou através do *smartphone* para perceber alguma anomalia como uma queda brusca.

Assim, esse trabalho pode ser continuado com a inclusão de novos sensores ou do aprimoramento do algoritmo já existente. A criação de bibliotecas de comunicação via Bluetooth e SMS entre o ESP32, o aplicativo Android e o SIM808 pode facilitar a integração de novos componentes. Fatores como memória e armazenamento de dados também estão facilitados uma vez que foi feita uma função que envia informações para um banco de dados na nuvem.

Por fim, a escolha pelo microcontrolador ESP32 possibilita o controle de vários outros componentes ao mesmo tempo devido ao elevado número de entradas GPIO, além de possuir uma capacidade de processamento superior aos seus antecessores, o que permite ao dispositivo realizar funções mais complexas.

Capítulo 7

Conclusões

O presente trabalho propôs um dispositivo que busca auxiliar idosos com perda de memória a terem mais autonomia, uma vez que, ao saírem de casa, os pacientes têm a segurança de que poderão pedir ajuda caso ocorra algum problema. O trabalho também se apresenta como uma forma de tranquilizar os familiares do usuário, pois eles receberão notificações em situações de perigo, mesmo se o idoso não estiver ciente disso.

O foco do trabalho são pessoas que, ao entrar em idade mais avançada, apresentam declínio em funções cognitivas como comunicação, orientação topográfica e memória recente. Tal problema se inicia desde os 50 anos, sendo mais comum ao chegar nos 70 anos de idade.

A tentativa de resgatar a autonomia em pacientes com idade avançada é bastante relevante, uma vez que a perda da autonomia está intimamente ligada com a queda da qualidade de vida na terceira idade. Além disso, é reconhecido que a população brasileira e mundial está envelhecendo, portanto, problemas ligados ao envelhecimento serão cada vez mais relevantes.

O trabalho também se encaixa na categoria de *mobile Health*, práticas médicas e de saúde pública suportadas por dispositivos *mobiles*, e utiliza sensores GPS para identificar possíveis situações de perigo. De acordo com a literatura, é possível encontrar outros dispositivos *mHealth* que utilizam sensores diferentes com o mesmo propósito. Estatísticas fisiológicas como batimento cardíaco, pressão sanguínea e oxigenação no sangue podem ser medidas com sensores específicos. Também é bastante comum o uso de sensores inerciais na tentativa de detecção de quedas, problema importante e bastante comprometedor para quem está em idade avançada.

Assim, o trabalho pode ser visto como um projeto modular, que pode ser complementado e aprimorado com a inclusão de outros sensores. O objetivo final desse aprimoramento é a construção de um botão do pânico não físico cada vez mais completo, ou seja, um dispositivo que consiga identificar situações de perigo mesmo que o paciente não tenha a consciência disso. Para tanto, a escolha pelo microcontrolador ESP32 é bastante positiva, uma vez que se trata de um

microcontrolador com muitas entradas GPIO e um alto poder de processamento se comparado aos seus precessores.

Bibliotecas importantes também foram construídas para que houvesse comunicação via Bluetooth e SMS entre ESP32, SIM808 e o aplciativo Android. Por fim, o trabalho inclui também uma rotina que envia informações a um banco de dados na AWS, que pode ser aproveitada para armazenar outras informações importantes sobre os pacientes. Essas informações poderão ser valiosas para ajudar a melhorar os algoritmos de identificação de situações de perigo, como também podem servir de base para outras aplicações e pesquisas, como é o caso do aplicativo de monitoramento apresentado.

Outro ponto bastante positivo do trabalho está em relação ao seu público alvo. Apesar de ter sido projetado para pessoas em idade avançada, o mesmo dispositivo pode auxiliar outros perfis que buscam uma autonomia maior para ter uma vida mais ativa. O dispositivo poderia ser utilizado a princípio por pessoas com demência mental leve, autistas e até por crianças.

Os resultados obtidos foram satisfatórios, uma vez que se cumpriu o cadastro dos parâmetros de segurança e o GPS conseguiu realizar uma leitura da sua localização atual, com o cálculo da distância em relação a sua casa pelo microcontrolador. Nos casos em que a distância era superior ao raio de segurança, o dispositivo enviou uma mensagem de SMS para o idoso e, posteriormente, para o contato de emergência cadastrado.

Como discutido, a precisão no cálculo da distância pelo GPS é dependente de uma série de fatores e poderia ser aperfeiçoada pela utilização de um sensor GPS mais preciso. Porém, como uma prova de conceito, o trabalho atendeu aos requisitos propostos e disponibilizou-se para uma série de modulações que podem aumentar ainda mais a abrangência do projeto.

Referências bibliográficas

World Population Ageing 2013 - http://www.un.org/en/development/desa/population/publications/pdf/ageing/WorldPopulationAgeing2013.pdf

mHealth: New horizons for health through mobile technologies - http://www.who.int/goe/publications/goe_mhealth_web.pdf

Tratado de Geriatria e Gerontologia - 3ª Edição.

Ahmed, Iftekhar Uddin, Nazia Hassan, and Humayun Rashid. 2017. “Solar Powered Smart Wearable Health Monitoring and Tracking Device Based on GPS and GSM Technology for Children with Autism.” 111–16.

Bharavi, U. and Rao M. Sukesh. 2017. “Design and Development of GSM and GPS Tracking Module.” Pp. 283–88 in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*. Retrieved (http://ieeexplore.ieee.org/document/8256602/).

Bourke, Alan K., Pepijn W. J. van de Ven, Amy E. Chaya, Gearoid M. OLaighin, and John Nelson. 2008. “Testing of a Long-Term Fall Detection System Incorporated into a Custom Vest for the Elderly.” *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* 2844–47. Retrieved (http://ieeexplore.ieee.org/document/4649795/).

Bourouis, Abderrahim and Mohamed Feham. 2011. “U Biquitous M Obile H Ealth M Onitoring.” *International Journal of Computer Science & Information Technology (IJCSIT)* 3(3):74–82.

Chang, Sung Yen, Chin Feng Lai, Han Chieh Josh Chao, Jong Hyuk Park, and Yueh Min Huang. 2011. “An Environmental-Adaptive Fall Detection System on Mobile Device.” *Journal of Medical Systems* 35(5):1299–1312.

Chiarini, Giovanni, Pradeep Ray, Shahriar Akter, Cristina Masella, and Aura Ganz. 2013. “MHealth Technologies for Chronic Diseases and Elders: A Systematic Review.” *IEEE Journal on Selected Areas in Communications* 31(9):6–18.

de Jager, Dirk et al. 2011. “A Low-Power, Distributed, Pervasive Healthcare System for Supporting Memory.” *Proceedings of the First ACM MobiHoc Workshop on Pervasive Wireless Healthcare - MobileHealth ’11* 1. Retrieved (http://portal.acm.org/citation.cfm?doid=2007036.2007043).

Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. 2010. “PerFallD: A Pervasive Fall Detection System Using Mobile Phones.” *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)* 292–97. Retrieved (http://ieeexplore.ieee.org/document/5470652/).

Lopes, Ivo C., Binod Vaidya, and Joel J. P. C. Rodrigues. 2013. “Towards an Autonomous Fall Detection and Alerting System on a Mobile and Pervasive Environment.” *Telecommunication Systems* 52(4):2299–2310.

Postolache, Octavian et al. 2011. “Enabling Telecare Assessment with Pervasive Sensing and Android OS Smartphone.” *MeMeA 2011 - 2011 IEEE International Symposium on Medical Measurements and Applications, Proceedings*.

Wu, Yu-Chi et al. 2009. “A Mobile Health Monitoring System Using RFID Ring-Type Pulse Sensor.” *2009 Eighth IEEE International Conference on*

Dependable, Autonomic and Secure Computing 317–22. Retrieved (http://ieeexplore.ieee.org/document/5380551/).

Yavuz, Gokhan Remzi et al. 2010. “A Smartphone Based Fall Detector with Online Location Support.” *International Workshop on Sensing for App Phones* (October 2014):31–35.

Apêndice A

Código SIM808

Main

```
/*

  ### GPS SMS loc
  Funcionalidades:
  1. Estabelecer comunicacao serial entre ESP32 e SIM808
  2. Realizar leitura da localizacao utilizando o GPS do SIM808
  3. Enviar uma mensagem de perigo para o paciente
  4. Ler o SMS de resposta do numero
  5. Acionar um contato de emergencia a um numero pre-cadastrado

*/

#include "SIM808_methods.h"
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <math.h>

#define SERVICE_UUID        "a49e4056-1260-4b64-a5a9-8b0ae5ba5126"
#define CHARACTERISTIC_UUID "42fcbe5f-ecb8-4577-a739-6ea6aa54615e"

//Definicao de pi
#define PI 3.14159265359

//Variaveis auxiliares
char MESSAGE[300]; //mensagem de SMS que sera enviada
char lat[12]; //latitude
char lon[12]; //longitude

float del_lat; //diferenca medida na latitude em radianos
float del_lon; //diferenca medida na latitude em radianos
float avg_lat; //latitude media em radianos
float K1, K2; //constantes para calculo da distancia
double dist; //distancia do usuario para sua casa

int state; //estado do usuario
int sms; //indentificador do SMS recebido

bool send_help = false;
bool help_sent = false;

unsigned long time_begin;
unsigned long time_now;
unsigned long time_help;

//Comunicacao serial com o SIM808
//PIN RX 16
//PIN TX 17
HardwareSerial Serial2(2);

//Inicia classe utilizando a biblioteca criada SIM808_methods
```

```
SIM808_methods class_sim808(&Serial2);

void setup() {
  Serial.begin(9600);
  Serial2.begin(9600);

  //Iniciando comunicacao com o SIM808
  while(!(class_sim808.init())) {
    delay(1000);
    Serial.print("\r\nProblema na comunicacao\r\n");
  }
  Serial.println("Comunicacao estabelecida!");

  //Verificando se existe problema com o cartao SIM
  while(!(class_sim808.SIM_status())) {
    delay(1000);
    Serial.println("Problema com cartao SIM\r\n");
  }
  Serial.println("SIM conectado!");

  //Ligando GPS
  while(!(class_sim808.turnon_GPS())) {
    delay(1000);
    Serial.print("Problema ao ligar GPS...\r\n");
  }
  Serial.println("GPS ligado com sucesso!");

  //Deleta todos os SMS (seguranca)
  while(!(class_sim808.deleteall_SMS())) {
    delay(1000);
    Serial.println("Problema ao deletar as mensagens\r\n");
  }
  Serial.println("Pronto para receber mensagens!");

  //Inicia comunicacao Bluetooth (BT)
  BLEDevice::init("MyESP32"); //nome do BT para pareamento
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ
                                       );
  pCharacteristic->setValue("11987345829"); //envia numero do SIM do ESP32
  pService->start();
  BLEAdvertising *pAdvertising = pServer->getAdvertising();
  pAdvertising->start();
  Serial.println("Bluetooth ligado!");

  //Verifica se recebeu uma resposta
  sms = class_sim808.new_SMS();
  if(!sms)
    Serial.print("\r\nEsperando SMS");
  //Enquanto nao recebe um SMS de resposta e esperou menos que 10s
  while(!sms) {
    Serial.print(".");
    delay(2000); //espera 2s
    sms = class_sim808.new_SMS(); //verifica se recebeu algo
  }

  while(!class_sim808.read_SMS_data(sms,CHARACTERISTIC_UUID,36)) {
```

```
    Serial.println("SMS com formatacao errada!");
    Serial.println("Espera-se um SMS no formato:");
    Serial.println("UUID,numero do contato de emergencia, latitude, longitude");
    //Verifica se recebeu uma resposta
    sms = class_sim808.new_SMS();
    if(!sms)
      Serial.print("\r\nAinda esperando SMS");
    //Enquanto nao recebe um SMS de resposta e esperou menos que 10s
    while(!sms) {
      Serial.print(".");
      delay(2000); //espera 2s
      sms = class_sim808.new_SMS(); //verifica se recebeu algo
    }
  }

  class_sim808.USERdata.safe_radius = 0.05; //raio de seguranca em km

  Serial.println();
  Serial.print("Numero do usuario = ");
  Serial.println(class_sim808.USERdata.user_phone);
  Serial.print("Numero de ajuda = ");
  Serial.println(class_sim808.USERdata.help_phone);
  Serial.print("Latitude = ");
  Serial.println(class_sim808.USERdata.lat_home);
  Serial.print("Longitude = ");
  Serial.println(class_sim808.USERdata.lon_home);
}

void loop() {
  /*if (Serial2.available()) {
    Serial.write(Serial2.read());
  }
  if (Serial.available()) {
    while(Serial.available()) {
      Serial2.write(Serial.read());
    }
  }*/

  delay(5000); //faz uma leitura do GPS a cada 5s

  //Se existe leitura no GPS do SIM808
  if(class_sim808.status_GPS()) {

    //Realiza a leitura do GPS e salva em 'GPSdata'
    class_sim808.read_GPS();
    Serial.print("Latitude GPS = ");
    Serial.println(class_sim808.GPSdata.lat);
    Serial.print("Longitude GPS = ");
    Serial.println(class_sim808.GPSdata.lon);

    //Verifica a distancia do usuario para sua casa usando a formula
    //distancia = sqrt((K1*del_lat)^2 + (K2*del_lon)^2); onde
    del_lat = (class_sim808.USERdata.lat_home - class_sim808.GPSdata.lat);
    del_lon = (class_sim808.USERdata.lon_home - class_sim808.GPSdata.lon);
    //De graus pra radianos
    avg_lat = 0.5*(class_sim808.USERdata.lat_home + class_sim808.GPSdata.lat);
    K1 = 111.13209 - 0.56605*cos(2*avg_lat) + 0.0012*cos(4*avg_lat);
    K2 = 111.41513*cos(avg_lat) - 0.09455*cos(3*avg_lat) + 0.00012*cos(5*avg_lat);
    dist = sqrt(sq(K1*del_lat) + sq(K2*del_lon));
```

```
    Serial.print("dist = ");
    Serial.println(dist);

    //Se o usuario saiu do raio de seguranca
    if(dist > class_sim808.USERdata.safe_radius) {
      Serial.println("Voce esta longe de casa!");
      Serial.println("Enviando SMS para o idoso...");
      //Envia mensagem  SMS para verificar se ele esta bem
      while(!class_sim808.send_SMS(class_sim808.USERdata.user_phone,"ALERTA"))
        delay(1000);

      Serial.println("Esperando uma resposta");
      //Verifica se recebeu uma resposta
      sms = class_sim808.new_SMS();

      time_begin = millis();
      time_now = millis();

      //Enquanto nao recebe um SMS de resposta e esperou menos que 10s
      while((!sms) && (time_now - time_begin < 10000)) {
        Serial.print(".");
        delay(1000); //espera 1s
        sms = class_sim808.new_SMS(); //verifica se recebeu algo
        time_now = millis();
      }

      //se nao obteve resposta em 10s
      if(time_now - time_begin >= 10000) {
        if(time_now - time_help > 300000)
          help_sent = false;
        if(!help_sent) {
          time_help = millis();
          send_help = true;
        }
      }

      //se recebeu uma resposta
      else {
        //verifica se e' do usuario e qual resposta

switch(class_sim808.read_SMS_safe(sms,class_sim808.USERdata.user_phone_size,class_sim808.U
SERdata.user_phone)) {
          case 0: //nao e' do paciente
            break;
          case 1: //usuario em seguranca
            break;
          case 2: //mensagem nao reconhecida
            break;
        }
      }
    } //fim do 'if fora da zona de seguranca'

    //Desligando GPS
    /*delay(500);
    while(!(class_sim808.turnoff_GPS())) {
      delay(1000);
      Serial.print("Problema ao desligar GPS...\r\n");
    }
    Serial.println("GPS desligado com sucesso!");*/
  }
```

```
  if(send_help) {

    time_help = millis();

    //Salva o  valor em float da latitude/longitude na string lat/lon
    //4 = casas decimais antes da virgula
    //7 = casas decimais depois da virgula
    dtostrf(class_sim808.GPSdata.lat, 4, 7, lat);
    dtostrf(class_sim808.GPSdata.lon, 4, 7, lon);

    sprintf(MESSAGE, "Socorro! Estou em: http://maps.google.com/maps?q=%s,%s\n", lat, lon);

    class_sim808.send_SMS(class_sim808.USERdata.help_phone,MESSAGE); //envia mensagem de
socorro

    help_sent = true;
    send_help = false;
  }

}
```

SIM808_methods.cpp

```
#include "SIM808_methods.h"

/*************
 * Variaveis *
 *************/

//Ponteiro para o serial sendo utilizado
HardwareSerial *serialGPS = NULL;

//Array para salvar resposta do SIM808
char serial_data[100]; //array
int pos = 0; //posicao do ultimo caracter lido

/***********
 * Metodos *
 ***********/

/*
 * SIM808_methods(HardwareSerial *mySerial)
 * Construtor da classe 'SIM808_methods', identifica
 * o ponteiro *serialGPS para o HardwareSerial que esta
 * sendo utilizado pelo arquivo principal
 */
SIM808_methods::SIM808_methods(HardwareSerial *mySerial){
  serialGPS = mySerial;
}

/*
 * bool init( )
 * Realiza setup inicial, verificando se a comunicacao esta OK
 * e definindo as funcoes do telefone para maxima
 */
bool SIM808_methods::init(void) {
  delay(500);

  //Envia comando de verificacao AT
```

```
  serialGPS->write("AT\r\n");
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT", 2), "OK") == 0)
    return false; //caso contrario, sai da funcao

  //Envia comando para definir as funcionalidades do celular
  serialGPS->write("AT+CFUN=1\r\n"); //Phone Functionality (1 = Full Functionality)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CFUN=1", 9), "OK") == 0) {
    ERROR_code('E'); //mensagem de erro do tipo CME
    return false; //sai da funcao
  }

  //Envia comando para selecionar o formato da mensagem
  serialGPS->write("AT+CMGF=1\r\n"); //SMS Message Format (1 = Text Mode)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CMGF=1", 9), "OK") == 0)
    return false; //caso contrario, sai da funcao

  return true; //fim da funcao
}

/*
 * bool SIM_status( )
 * Verifica se alguma senha e' requerida ou nao para o
 * cartao SIM inserido
 */
bool SIM808_methods::SIM_status(void) {

  delay(500);

  //Envia comando de verificacao de senha
  serialGPS->write("AT+CPIN?\r\n");
  delay(500);

  //Aguarda retorno de que nao existe senha pendente
  if(!strcmp(read_buffer("AT+CPIN?", 8), "READYOK") == 0) {
    if(strcmp(serial_data, "SIMPINOK") == 0) //resposta 'SIM PIN'
      Serial.println("\r\nAguardando SIM PIN ser fornecido");
    if(strcmp(serial_data, "SIMPUKOK") == 0) //resposta 'SIM PUK'
      Serial.println("\r\nAguardando SIM PUK ser fornecido");
    if(strcmp(serial_data, "PH_SIMPINOK") == 0) //resposta 'PH_SIM PIN'
      Serial.println("\r\nAguardando telefone para cartao SIM (antifurto)");
    if(strcmp(serial_data, "PH_SIMPUKOK") == 0) //resposta 'PH_SIM PUK'
      Serial.println("\r\nAguardando por SIM PUK (antifurto)");
    return false; //resposta nao esperada, sai da funcao
  }

  return true; //fim da funcao
}

/*
 * bool turnon_GPS( )
 * Liga o GPS
```

```
 */
bool SIM808_methods::turnon_GPS(void) {

  delay(500);

  //Envia comando para ligar o GPS
  serialGPS->write("AT+CGPSPWR=1\r\n"); //GPS Power Controle (1 = turn on GPS power supply)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CGPSPWR=1", 12), "OK") == 0) {
    //serial_data = '89x', com x = [0,1,2]
    if(serial_data[2] == '0') //erro de codigo 890 (GPS not running)
      Serial.println("\r\nGPS nao esta rodando");
    if(serial_data[2] == '1') //erro de codigo 891 (GPS is running)
      Serial.println("\r\nGPS esta rodando, aguarde...");
    if(serial_data[2] == '2') //erro de codigo 892 (GPS is fixing)
      Serial.println("\r\nGPS esta fixando posicao, aguarde...");
    return false; //resposta inesperada 'ERROR', sai da funcao
  }

  return true; //fim da funcao
}

/*
 * bool turnoff_GPS( )
 * Desliga o GPS
 */
bool SIM808_methods::turnoff_GPS(void) {

  delay(500);

  //Envia comando para desligar o GPS
  serialGPS->write("AT+CGPSPWR=0\r\n"); //GPS Power Controle (0 = turn off GPS power supply)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CGPSPWR=0", 12), "OK") == 0) {
    //serial_data = '89x', com x = [0,1,2]
    if(serial_data[2] == '0') //erro de codigo 890 (GPS not running)
      Serial.println("\r\nGPS nao esta rodando");
    if(serial_data[2] == '1') //erro de codigo 891 (GPS is running)
      Serial.println("\r\nGPS esta rodando, aguarde...");
    if(serial_data[2] == '2') //erro de codigo 892 (GPS is fixing)
      Serial.println("\r\nGPS esta fixando posicao, aguarde...");
    return false; //resposta inesperada 'ERROR', sai da funcao
  }

  return true; //fim da funcao
}

/*
 * bool GPS_status( )
 * Verifica se o GPS leu alguma possicao
 */
bool SIM808_methods::status_GPS(void) {

  delay(500);

  //Envia comando para verificar o status do GPS
```

```
  serialGPS->write("AT+CGPSSTATUS?\r\n");
  delay(500);

  //Aguarda retorno de resposta com localizacao fixada
  if((!strcmp(read_buffer("AT+CGPSSTATUS?", 14),"Location3DFixOK") ==
0)&&(!strcmp(serial_data,"Location2DFixOK") == 0)) {
    if(strcmp(serial_data, "LocationUnknownOK") == 0) //resposta 'Location Unknown'
      Serial.println("\r\nGPS nao esta ligado");
    if(strcmp(serial_data, "LocationNotFixOK") == 0) //resposta 'Location Not Fix'
      Serial.println("\r\nGPS esta ligado, mas nao tem leitura fixa");
    return false; //resposta nao esperada, sai da funcao
  }

  return true; //fim da funcao
}

/*
 * bool read_GPS( )
 * Salva no struct 'GPSdata' as informacoes lidas
 * na ultima leitura de posicao GPS valida
 */
void SIM808_methods::read_GPS(void) {

  //Variaveis auxiliares
  int comma = 0; //contador de virgulas
  char aux[20]; //string auxiliar que salva os dados momentaneamente
  int aux_pos = 0; //indicador de posicao auxiliar da string
  char NSEW; //char posicao Norte Sul Leste Oeste
  float temp; //salva os valores float por um tempo

  delay(500);

  //Envia comando de leitura da localizacao GPS
  serialGPS->write("AT+CGPSINF=32\r\n"); //'32' indentifica dados $GPRMC
  delay(500);

  //Salva no 'serial_data' a resposta do comando
  read_buffer("AT+CGPSINF=32", 13);

  //serial_data = 'ID,horarioUTC,status,latitude,N/S,longitude,L/O,velocidade,direcao,data,<etc>
  //Loop para percorrer serial_data ignorando os dois ultimos caracteres ('OK')
  for(int i = 0; i < pos-2; i++) {

    //se encontra uma virgula
    if(serial_data[i] == ',') {
      if(comma == 1) { //se aux tiver o horarioUTC
        temp = floor(atof(aux)); //temp = hhmmss
        GPSdata.hour = temp/10000; //hour = hh
        temp -= 10000*GPSdata.hour; //temp = mmss
        GPSdata.minute = temp/100; //minute = mm
        temp -= 100*GPSdata.minute; //temp = ss
        GPSdata.second = temp; //second = ss
      }

      else if(comma == 4) { //se aux tiver latitude e NSEW a direcao
        temp = atof(aux); //temp = ddmm.mmmmm
        GPSdata.lat = floor(temp/100); //lat = dd (so graus)
        temp -= (100*GPSdata.lat); //temp = mm.mmmm (so os minutos)
        GPSdata.lat += temp/60; //soma os minutos/60
        if (NSEW == 'S') GPSdata.lat *= -1; //Sul e' indicado por -
```

```
    }

    else if(comma == 6) { //se aux tiver longitude e NSEW a direcao
      temp = atof(aux); //temp = ddmm.mmmm
      GPSdata.lon = floor(temp/100); //lon = dd (so graus)
      temp -= (100*GPSdata.lon); //temp = mm.mmmm (so os minutos)
      GPSdata.lon += temp/60; //soma os minutos/60
      if (NSEW == 'W') GPSdata.lon *= -1; //Oeste(West) e' indicado por -
    }

    else if(comma == 7) //se aux tiver a velocidade em nos
      GPSdata.speed_kph = atof(aux) * 1.852; //conversao de nos para km/h

    else if(comma == 8) //se aux tiver a direcao em graus
      GPSdata.heading = atof(aux);

    else if(comma == 9) { //se aux tiver a dataUTC
      temp = floor(atof(aux)); //temp = ddmmyy
      GPSdata.day = temp/10000; //day = dd
      temp -= 10000*GPSdata.day; //temp = mmyy
      GPSdata.month = temp/100; //month = mm
      temp -= 100*GPSdata.month; //temp = yy
      GPSdata.year = temp; //year = yy
    }

    comma++; //acrescentar contador de virgulas
    if((comma != 4)&&(comma != 6)) //nao reinicia o aux se ele tiver a lat ou lon
      memset(aux, 0, sizeof(aux)); //reiniciar string auxiliar
    aux_pos = 0; //reiniciar auxiliar de posicao
  }

  //caso contrario, caracter e' de interesse (ignora-se o ID, status e dados depois da 'data')
  else {
    if(comma == 1) //horario UTC em formato hhmmss.sss
      aux[aux_pos++] = serial_data[i];
    else if(comma == 3) //latitude em formato ddmm.mmmmmm
      aux[aux_pos++] = serial_data[i];
    else if(comma == 4) //direcao Norte (+1) ou Sul (-1)
      NSEW = serial_data[i];
    else if(comma == 5) //longitude em formato ddmm.mmmmmm
      aux[aux_pos++] = serial_data[i];
    else if(comma == 6) //direcao East/Leste (+1) ou West/Oeste (-1)
      NSEW = serial_data[i];
    else if(comma == 7) //velocidade em nos
      aux[aux_pos++] = serial_data[i];
    else if(comma == 8) //direcao em graus
      aux[aux_pos++] = serial_data[i];
    else if(comma == 9) //dia UTC em formato ddmmyy
      aux[aux_pos++] = serial_data[i];
  }
 } //fim loop

 reset_buffer();
}

/*
 * bool deleteall_SMS( )
 * Deleta todos os SMS no cartao SIM
 */
bool SIM808_methods::deleteall_SMS(void) {
```

```
  int timeout = 0; //contador de tempo transcorrido

  //Envia comando para deletar todos os SMS
  serialGPS->write("AT+CMGDA=\"DEL ALL\"\r\n");

  //Enquanto nao se passaram 30s
  while(timeout < 6) {
    delay(5000); //demora ate 25s para deletar 150 mensagens
    timeout++; //soma contador a cada 5s
    if(serialGPS->available()) //se o serial do GPS respondeu com algo
      break; //sai do 'while' para ver a resposta
  }

  //Se nao obter resposta no tempo limite
  if(timeout == 6) {
    Serial.println("Esperou-se 30s (tempo maximo) e nao se obteve resposta...");
    Serial.println("Tente de novo");
    return false; //sai da funcao com fracasso
  }

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CMGDA=\"DELALL\"", 17), "OK") == 0) {
    if(serial_data[1] == 'E') //serial_data = ERROR
      Serial.println("\r\nAlgo deu errado");
    else //serial_data = +CMS ERROR: <err>
      ERROR_code('S'); //mensagem de erro do tipo CMS
    return false; //resposta inesperada 'ERROR', sai da funcao
  }

  return true; //fim da funcao
}

/*
 * bool delete_SMS(char *index)
 * Deleta um SMS que esta' indexado
 * com valor 'index'
 */
bool SIM808_methods::delete_SMS(char *index) {

  delay(500);

  //Envia comando para deletar SMS indexidada 'index'
  serialGPS->write("AT+CMGD=");
  serialGPS->write(index);
  serialGPS->write("\r\n");

  delay(5000); //pode demorar ate' 5s para deletar um SMS

  //Salva no 'serial_data' a resposta do comando
  read_buffer("+", 1);

  //Verifica-se entao se os dois ultimos caracteres sao 'OK'
  if(serial_data[pos-2] == 'O') {
    if(serial_data[pos-1] == 'K')
      return true; //sai da funcao com sucesso
  }

  else //caso contrario
    ERROR_code('S'); //mensagem de erro do tipo CMS
```

```
  return false; //sai da funcao com fracasso
}

/*
 * int new_SMS( )
 * Verifica se a comunicacao serial recebeu uma
 * notificacao de que o SIM tem um novo SMS.
 * Caso tenha, a funcao retorna o 'index', ou seja,
 * o indentificador do novo SMS
 */
int SIM808_methods::new_SMS(void) {

  char c; //'char' que fara a leitura da resposta

  reset_buffer(); //por seguranca

  //Se existir algo para ser lido no serial,
  //deve-se verificar se e' uma notificacao de
  //novo SMS, recebido na forma:
  //+CMTI:"SM",i | onde:
  //"SM" = indicando novo SMS
  //i = 'index' da novo mensagem
  if(serialGPS->available()){
    while (serialGPS->available()) { //enquanto existir algo
      c = serialGPS->read(); //salva o 'char' disponivel
      if((c != '\n')&&(c != '\r')&&(c != ' ')) //se nao for um 'whitespace'
        serial_data[pos++] = c; //salva no 'serial_data'
        if (c == ':') { //quando receber o char ':'
                        //verifica se e' um aviso do tipo +CMTI:
          if(!strcmp(serial_data, "+CMTI:") == 0)
            return 0; //caso contrario, sai da funcao com fracasso
          reset_buffer(); //caso positivo, limpa 'serial_data'
        }
        //na virgula, serial_data = "SM"
        if(c == ',')
          reset_buffer(); //portanto, limpa-se o array para guardar so o 'index'
    }
    //converte o 'index' para int e termina a funcao
    return atoi(serial_data);
  }

  return 0; //fim da funcao com fracasso
}

/*
 * bool read_SMS_data(int index, char* UUID, int UUID_size)
 * Faz a leitura do SMS com 'index'
 * e verifica se ele tem o UUID correspondente.
 * Se tiver, faz a leitura dos dados da mensagem
 */
bool SIM808_methods::read_SMS_data(int index, char* UUID, int UUID_size) {

  char c; //'char' que fara a leitura da resposta
  char ind[4]; //array com o 'index'
  char aux[20]; //'string' auxiliar que salva os dados momentaneamente
  char number[20]; //'string' com o numero que enviou a mensagem
  int num_pos = 0; //tamanho do numero lido
  int aux_pos = 0; //indicador de posicao auxiliar da string
  int received_uuid_size = 0; //tamanho do UUID recebido
```

```
int comma = 0; //contador de virgulas
int quotes = 0; //contador de aspas
int j = 0; //auxiliar para percorrer os array

itoa(index,ind,10); //converte o int (index) para o array (ind)

//Envia comando para ler o SMS indexado 'ind'
serialGPS->write("AT+CMGR=");
serialGPS->write(ind);
serialGPS->write("\r\n");
delay(500);

reset_buffer(); //por seguranca

//Enquanto existir algo para ser lido (resposta)
//serial_data = "RECUNREAD","XXXXXXXXXXX","","yy/mm/dd,HH:MM:SS+TZ"dataOK
//"RECUNREAD" = unread record / mensagem nao lida
//"XXXXXXXXXXX" = numero de telefone que mandou a mensagem
//"yy/mm/dd,HH:MM:SS+TZ" = dia e hora com fuso horario (TZ)
//data = o conteudo da mensagem recebida
//OK = indicacao que a leitura foi realizada com sucesso
while (serialGPS->available()) {
  c = serialGPS->read(); //salva o 'char' disponivel
  if (c == '"') //se encontrar um aspas
    quotes++; //soma contador de aspas
  if (c == ',') //se encontrar uma virgula
    comma++; //soma contador de virgulas
  if ((c != '\n')&&(c != '\r')&&(c != ' ')&&(c != '"')) { //se nao for um 'whitespace'
    if((quotes == 3)&&(comma == 1)) //se for o numero que enviou o SMS
      number[num_pos++] = c; //salva na string 'number'
    else if((quotes == 8)&&(pos < 100)) //se for o dataOK
      serial_data[pos++] = c; //salva na string 'data'
  }
}

//Se for o SMS que se espera, o seu conteudo (serial_data) deve ser:
//serial_data = UUID,numero de emergencia,latitude,longitude
//Portanto, verifica-se se o tamanho do UUID (ate encontrar a 1a virgula)
//ou ate acabar a string 'data', nesse caso sendo falso
while ((serial_data[received_uuid_size] != ',')&&(received_uuid_size < pos))
  received_uuid_size++;

//Se o SMS e' o esperado, isso nunca seria possivel
//ja que o 'serial_data' tem muito mais do que so o UUID
if(received_uuid_size == pos) {
  reset_buffer(); //limpa o 'serial_data'
  return false; //sai da funcao com fracasso
}

//Se o UUID nao tiver o tamanho certo
if(received_uuid_size != UUID_size) {
  reset_buffer(); //limpa o 'serial_data'
  return false; //sai da funcao com fracasso
}

//Compara-se cada caracter do UUID para ver se e' igual
while((serial_data[j] == UUID[j])&&(j < UUID_size))
  j++;

//Se nao for igual
```

```
  if(j != UUID_size) {
    delete_SMS(ind); //deleta o SMS
    reset_buffer(); //limpa o 'serial_data'
    return false; //sai da funcao com fracasso
  }

  //Se for o UUID esperado, salva-se o numero que enviou
  //o SMS nos dados do usuario
  for(j = 0; j < num_pos; j++)
    USERdata.user_phone[j] = number[j];

  USERdata.user_phone_size = num_pos;

  comma = 0; //reinicia contador de virgulas

  //Fazer a leitura dos outros dados
  //data = UUID,numero_de_emergencia,latitude,longitude
  //Loop para percorrer data ignorando os dois ultimos caracteres ('OK')
  for(int i = 0; i < pos-2; i++) {
    if(serial_data[i] == ',') { //se encontrar virgula
      if(comma == 1) { //se ja encontrou uma virgula, entao acabou de ler o numero de emergencia
        for(j = 0; j < aux_pos; j++)
          USERdata.help_phone[j] = aux[j]; //salva o numero nos dados
      }
      else if(comma == 2) //se ja encontrou duas virgulas, entao acabou de ler a latitude
        USERdata.lat_home = atof(aux); //salva o numero nos dados

      aux_pos = 0; //reinicia contador auxiliar de posicao da string
      memset(aux, 0, sizeof(aux)); //reiniciar string auxiliar
      comma++; //soma contador de virgulas
    }
    else { //caso nao seja uma virgula
      if(comma == 1) //se ja contou uma virgula
        aux[aux_pos++] = serial_data[i]; //salva numero de emergencia
      else if(comma == 2) //se ja contou duas virgulas
        aux[aux_pos++] = serial_data[i]; //salva latitude
      else if(comma == 3) //se ja contou tres virgulas
        aux[aux_pos++] = serial_data[i]; //salva longitude
    }
  } //fim loop

  USERdata.lon_home = atof(aux); //salva longitude

  delete_SMS(ind); //deleta o SMS

  return true; //fim da funcao
}

/*
 * int read_SMS(int index, char* PHONE)
 * Faz a leitura do SMS com indexado 'index'
 * e verifica se e' do numero de interesse PHONE.
 * Se for, verifica seu conteudo.
 * Caso contrario, deleta a mensagem
 */
int SIM808_methods::read_SMS_safe(int index, int phone_size, char* PHONE) {

  char c; //'char' que fara a leitura da resposta
  char ind[4]; //array com o 'index'
  char aux[20]; //'string' auxiliar que salva os dados momentaneamente
```

```
int aux_pos = 0; //indicador de posicao auxiliar da string
int comma = 0; //contador de virgulas
int quotes = 0; //contador de aspas
int j = 0; //auxiliar para percorrer os array
bool flag_msg = false; //indica se a mensagem ja foi lida

itoa(index,ind,10); //converte o int (index) para o array (ind)

//Envia comando para ler o SMS indexado 'ind'
serialGPS->write("AT+CMGR=");
serialGPS->write(ind);
serialGPS->write("\r\n");
delay(500);

reset_buffer(); //por seguranca

//Enquanto existir algo para ser lido (resposta)
while (serialGPS->available()) {
  c = serialGPS->read(); //salva o 'char' disponivel
  //ignora a resposta "AT+CMGR=ind\r\n+CMGR:"
  //quando abrir aspas, salva a resposta de interesse no 'serial_data'
  if (c == '"')
    flag_msg = true;
  if (((c != '\n')&&(c != '\r')&&(c != ' '))&&flag_msg) //se nao for um 'whitespace'
    serial_data[pos++] = c; //salva no 'serial_data'
}

//serial_data = "RECUNREAD","XXXXXXXXXXXX","","yy/mm/dd,HH:MM:SS+TZ"dataOK
//"RECUNREAD" = unread record / mensagem nao lida
//"XXXXXXXXXXX" = numero de telefone que mandou a mensagem
//"yy/mm/dd,HH:MM:SS+TZ" = dia e hora com fuso horario (TZ)
//data = o conteudo da mensagem recebida
//OK = indicacao que a leitura foi realizada com sucesso
//Loop para percorrer serial_data ignorando os dois ultimos caracteres ('OK')
for(int i = 0; i < pos-2; i++) {
  //se encontra uma virgula
  if(serial_data[i] == ',') {
    //se encontrou uma virgula e ja contou outra
    //significa que acabou de salvar o numero do contato
    if(comma == 1) {
      if(aux_pos == phone_size) { //se o numero tem o tamanho salvo
        while(aux[j] = PHONE[j]) //enquanto os numeros foram iguais
          j++; //percorre o array com o numero salvo e o numero que envio o SMS
        if(j == phone_size) { //se os numeros forem exatamente iguais
          memset(aux, 0, sizeof(aux)); //reiniciar string auxiliar
          aux_pos = 0; //reiniciar auxiliar de posicao
        }
        else { //se forem diferentes
          delete_SMS(ind); //deleta o SMS
          return 0; //sai da funcao com fracasso
        }
      } //fim do 'if aux_pos==phone_size'
      else { //se nao tiver o tamanho salvo
        delete_SMS(ind); //deleta o SMS
        return 0; //sai da funcao com fracasso
      }
    } //fim do 'if comma == 1'
    comma++; //acrescentar contador de virgulas
  } //fim do 'if virgula'
```

```
    //se encontrar uma aspas
    else if(serial_data[i] == '"')
      quotes++; //acrescentar contador de aspas

    //caso contrario, caracter e' de interesse (ignora-se o RECUNREAD, dia e hora)
    else {
      if(comma == 1) //depois da primeira virgula, salva-se o numero de telefone
        aux[aux_pos++] = serial_data[i];
      if(quotes == 8) //depois da oitava aspas, salva-se o conteudo do SMS
        aux[aux_pos++] = serial_data[i];
    } //fim do 'else'
  } //fim loop

  reset_buffer(); //limpa o 'serial_data'
  delete_SMS(ind); //deleta o SMS

  if(strcmp(aux, "Estou bem!") == 0) //se o conteudo era 'seguro'
    return 1; //retorna valor '1'

  return 2; //se nao era nenhum dos dois, retorna valor '2'
}

/*
 * bool send_SMS(char *PHONE, char *MESSAGE)
 * Envia um SMS com a mensagem MESSAGE para
 * o numero de celular PHONE
 */
bool SIM808_methods::send_SMS(char *PHONE, char *MESSAGE) {

  delay(500);

  //Envia comando para selecionar o formato da mensagem
  serialGPS->write("AT+CMGF=1\r\n"); //SMS Message Format (1 = Text Mode)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CMGF=1", 9), "OK") == 0)
    return false; //caso contrario, sai da funcao
  delay(500);

  //Envia comando para selecionar o formato dos caracteres
  serialGPS->write("AT+CSCS=\"GSM\"\r\n"); //TE Character Set (GSM = 7 bit default alphabet)
  delay(500);

  //Aguarda retorno de resposta 'OK'
  if(!strcmp(read_buffer("AT+CSCS=\"GSM\"", 13), "OK") == 0)
    return false; //caso contrario, sai da funcao

  //Envia comando para enviar SMS
  //Comando deve ter forma +GSM=<da><text><ctrlz>, onde
  serialGPS->write("AT+CMGS=\""); //<da> = destination-adress
  serialGPS->write(PHONE); //no caso, o telefone que recebera a mensagem
  serialGPS->write("\"\r\n");
  delay(1000);

  //Aguarda '>' indicando que a mensagem pode ser escrita
  if(!wait_response('>'))
    return false; //caso contrario, sai da fucnao

  serialGPS->write(MESSAGE); //<text> com o corpo da mensagem
```

```
  serialGPS->write("\r\n");
  delay(500);

  serialGPS->write((char)26); //<ctrlz>, no caso, o caracter ASCII numero (decimal) 26
  serialGPS->write("\r\n");
  delay(5000);

  //Salva no 'serial_data' a resposta do comando
  read_buffer("+", 1);

  //A resposta tem formato +CMGS:<mr>OK, onde
  //<mr> e' um inteiro que indentifica a mensagem (ID)
  //Verifica-se entao se os dois ultimos caracteres sao 'OK'
  if(serial_data[pos-2] == 'O') {
    if(serial_data[pos-1] == 'K')
      return true;
  }

  else //caso contrario
    ERROR_code('S'); //mensagem de erro do tipo CMS

  return false; //sai da funcao
}

/*
 * void reset_buffer( )
 * Limpa o vetor de caracteres 'serial_data'
 * e o contador de posicao 'pos'
 */
void SIM808_methods::reset_buffer(void) {
  memset(serial_data, 0, sizeof(serial_data));
  pos = 0;
}

/*
 * char* read_buffer(char* msg_sent, int msg_size)
 * Salva em 'serial_data' a reposta do modulo a um
 * comando 'msg_sent' pelo serial.
 * O ESP32 recebe o comando enviado e uma resposta,
 * portanto, deve-se ignorar a confirmacao do que foi
 * enviado e salvar apenas a resposta.
 * A resposta tem formato +CXXX: <data>, onde apenas
 * <data> e' de interesse.
 */
char* SIM808_methods::read_buffer(char* msg_sent, int msg_size) {

  char c; //'char' que fara a leitura da resposta

  reset_buffer(); //por seguranca

  //Enquanto existir algo para ser lido
  while (serialGPS->available()) {
    c = serialGPS->read(); //salva o 'char' disponivel
    if ((c != '\n')&&(c != '\r')&&(c != ' ')) { //se nao for um 'whitespace'
      serial_data[pos++] = c; //salva no 'serial_data'
      if(c == ':') //se for um indicador de resposta
        reset_buffer(); //limpa 'serial_data'
      if((pos == msg_size)&&(strcmp(serial_data, msg_sent) == 0)) //se for a confirmacao de comando
        reset_buffer(); //limpa 'serial_data'
    }
```

```
  }

  //E' de interesse retornar o array para comparacao com 'strcmp'
  return serial_data; //sai da funcao
}

/*
 * bool wait_response(char r)
 * Aguarda o retorno de um 'char' especifico
 * ou ate o tempo acabar
 */
bool SIM808_methods::wait_response(char r) {

  char c; //'char' que fara a leitura da resposta
  unsigned long time_begin = millis();
  unsigned long time_now = millis();

  //Enquanto existir algo para ser lido e tempo esperado < 20s
  while (time_now - time_begin < 20000) {
    while (serialGPS->available()) {
      c = serialGPS->read(); //salva o 'char' disponivel
      if (c == r) //verifica se ele e' o esperado
        return true; //sai da funcao
    }
    delay(1000);
    time_now = millis();
  }

  return false; //sai da funcao
}

/*
 * void ERROR_code(char x)
 * Mostra no 'Monitor serial' a resposta de erro
 * que o ESP32 recebeu
 */
void SIM808_methods::ERROR_code(char x) {

  //char x indentifica se o erro e' do tipo CME ou CMS
  if(x == 'E') {
    Serial.println("\r\nErro do tipo CME ERROR");
    Serial.println("Erro relacionado com equipamento movel ou network");
    Serial.println("Codigo do erro: ");
    Serial.print(serial_data);
    Serial.println("\r\nVerificar manual SIM808 Series_AT_Command Manual_V1.02");
    Serial.println("Paginas 320 a 323");
  }

  else if (x == 'S') {
    Serial.println("\r\nErro do tipo CMS ERROR");
    Serial.println("Erro relacionado com serviço de mensagem ou network");
    Serial.println("Codigo do erro: ");
    Serial.print(serial_data);
    Serial.println("\r\nVerificar manual SIM808 Series_AT_Command Manual_V1.02");
    Serial.println("Paginas 323 a 326");
  }

  else //erro estranho ao programa
    Serial.println("Erro indevido");
}
```

SIM808_methods.h

```
#ifndef SIM808_H
#define SIM808_H

#if (ARDUINO >= 100)
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

class SIM808_methods {
  public:
    SIM808_methods(HardwareSerial *mySerial);
    bool init(void);
    bool SIM_status(void);
    bool turnon_GPS(void);
    bool turnoff_GPS(void);
    bool status_GPS(void);
    void read_GPS(void);
    bool deleteall_SMS(void);
    bool delete_SMS(char* index);
    int new_SMS(void);
    bool read_SMS_data(int index, char* UUID, int UUID_size);
    int read_SMS_safe(int index, int phone_size, char* PHONE);
    bool send_SMS(char* PHONE, char* MESSAGE);

    struct userdata{
      char user_phone[20];
      int user_phone_size;
      char help_phone[20];
      float lat_home;
      float lon_home;
      float safe_radius;
    }USERdata;

    struct gspdata{
      int year;
      int month;
      int day;
      int hour;
      int minute;
      int second;
      float lat;
      float lon;
      float speed_kph;
      float heading;
    }GPSdata;

  private:
    void reset_buffer(void);
    char* read_buffer(char* msg_sent, int msg_size);
    bool wait_response(char r);
    void ERROR_code(char x);
};


#endif
```

Apêndice B

Código Android - Aplicativo do paciente

Main Activity

```
public class MainActivity extends AppCompatActivity {

  int PERMISSION_ALL = 1;
  String[] PERMISSIONS = {
        Manifest.permission.BLUETOOTH,
        Manifest.permission.BLUETOOTH_ADMIN,
        Manifest.permission.READ_SMS,
        Manifest.permission.SEND_SMS,
        Manifest.permission.ACCESS_COARSE_LOCATION,
        Manifest.permission.ACCESS_FINE_LOCATION
  };
  DynamoDBMapper dynamoDBMapper;
  int id;
  int intervalo = 1000*60*2;

  @RequiresApi(api = Build.VERSION_CODES.M)
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     ActivityCompat.requestPermissions(this, PERMISSIONS, PERMISSION_ALL);

     Button b_estado = (Button) findViewById(R.id.b_estado); // Informar estado
     b_estado.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
           Intent i = new Intent(v.getContext(), EstadoPaciente.class);
           startActivity(i);// Vai para Activity EstadoPaciente
        }
     });

     Button b_parametros = (Button) findViewById(R.id.b_parametros); // Parâmetros de Segurança
     b_parametros.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
           Intent i = new Intent(v.getContext(), ParametrosSeguranca.class);
           startActivity(i);// Vai para Activity ParâmetrosSeguranca
        }
     });

     Button b_conectar = (Button) findViewById(R.id.b_conectar);
     b_conectar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           Intent i = new Intent(v.getContext(), ConectarDispositivo.class);
           startActivity(i);// Vai para Activity ConectarDispositivo

        }
     });

     SharedPreferences sharedPref = getSharedPreferences("endereco_casa",
Context.MODE_PRIVATE);
     String str_endereco = sharedPref.getString("endereco_casa", null);
```

```
        if (str_endereco == null) {
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setMessage("Por favor adicione o endereço de sua casa").setTitle("Endereço de casa
não cadastrado");
            builder.setPositiveButton("ok", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Intent i = new Intent(getBaseContext(), EditarEndereco.class);
                    startActivity(i);// Vai para Activity EstadoPaciente
                }
            });
            AlertDialog dialog = builder.create();
            dialog.show();
        }

        sharedPref = getSharedPreferences("telefone_contato", Context.MODE_PRIVATE);
        String str_telefone = sharedPref.getString("telefone_contato", null);
        if (str_telefone == null) {
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setMessage("Por favor adicione o telefone do contato de
emregência").setTitle("Telefone de emergência não cadastrado");
            builder.setPositiveButton("ok", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Intent i = new Intent(getBaseContext(), EditarContato.class);
                    startActivity(i);// Vai para Activity EstadoPaciente
                }
            });
            AlertDialog dialog = builder.create();
            dialog.show();
        }

        //Monitoramento

        AWSMobileClient.getInstance().initialize(this, new AWSStartupHandler() {
            @Override
            public void onComplete(AWSStartupResult awsStartupResult) {
                Log.d("YourMainActivity", "AWSMobileClient is instantiated and you are connected to
AWS!");
            }
        }).execute();


        AWSCredentialsProvider credentialsProvider =
AWSMobileClient.getInstance().getCredentialsProvider();
        AWSConfiguration configuration = AWSMobileClient.getInstance().getConfiguration();

        AmazonDynamoDBClient dynamoDBClient = new AmazonDynamoDBClient(credentialsProvider);

        this.dynamoDBMapper = DynamoDBMapper.builder()
                .dynamoDBClient(dynamoDBClient)
                .awsConfiguration(configuration)
                .build();

        final LocationManager locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);
        final LocationListener[] locationListener = new LocationListener[1];
        id = recupera_ultimo_id();
```

```
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        //    ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                          int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, intervalo, 10,
locationListener[0] = new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
           registrarLocal(location.getLatitude(), location.getLongitude(), id);
           id++;
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {

        }

        @Override
        public void onProviderEnabled(String provider) {

        }

        @Override
        public void onProviderDisabled(String provider) {

        }
    });

  }

  @Override
  protected void onStop() {
    super.onStop();
    SharedPreferences sharedPreferences =
getSharedPreferences("ultimo_id",Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt("ultimo_id",id);
    editor.commit();
  }

  public void registrarLocal(Double lat, Double lng, double id) {

    final PacientesDO newLoc = new PacientesDO();
    newLoc.setUserId("0");
    newLoc.setLatitude(lat);
    newLoc.setLongitude(lng);
    newLoc.setLocation(id);

    Log.d("Location", "Location updated!");

    new Thread(new Runnable() {
```

```
        @Override
        public void run() {
            dynamoDBMapper.save(newLoc);
            // Item saved
        }
    }).start();
  }
  public int recupera_ultimo_id(){
    int id;
    SharedPreferences sharedPreferences =
getSharedPreferences("ultimo_id",Context.MODE_PRIVATE);
    id = sharedPreferences.getInt("ultimo_id",0);
    return  id;
  }

}
```

ConectarDispositivo

```
public class ConectarDispositivo extends AppCompatActivity {

  BluetoothDevice mDevice;
  BluetoothManager btManager;
  BluetoothAdapter btAdapter;
  BluetoothGatt bluetoothGatt;
  String chave, endereco_lat, endereco_long,numero_emerg;
  TextView tv_chave;
  String numero_ESP32;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_conectar_dispositivo);

    SharedPreferences sharedPref = getSharedPreferences("telefone_contato",
Context.MODE_PRIVATE);
    numero_emerg = sharedPref.getString("telefone_contato",null);

    sharedPref = getSharedPreferences("endereco_casa_lat", Context.MODE_PRIVATE);
    endereco_lat = sharedPref.getString("endereco_casa_lat",null);

    sharedPref = getSharedPreferences("endereco_casa_long",Context.MODE_PRIVATE);
    endereco_long = sharedPref.getString("endereco_casa_long",null);

    tv_chave = findViewById(R.id.et_chave);
    Button b_bluetooth = findViewById(R.id.b_bluetooth);

    b_bluetooth.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            chave = tv_chave.getText().toString();
            mDevice = encontrar_ESP();
            bluetoothGatt = mDevice.connectGatt(getBaseContext(),false,btleGattCallback);



        }
    });
  }
```

```
  BluetoothGattCallback btleGattCallback = new BluetoothGattCallback() {
      @Override
      public void onServicesDiscovered(BluetoothGatt gatt, int status) {
          super.onServicesDiscovered(gatt, status);
          encontrar_service(gatt.getServices(), gatt);
          Log.d("OnServicesDiscovered","Entrei aqui");
      }

      @Override
      public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic,
int status) {
          super.onCharacteristicRead(gatt, characteristic, status);
          Log.d("OnCharacteristicRead","Entrei aqui também");

          byte[] value_bt = null;
          String value = null;
          value_bt = characteristic.getValue();
          if(value_bt == null){
              Log.d("VALOR","valor nulo");
          }else{
              try {
                  value = new String(value_bt,"UTF-8");
              } catch (UnsupportedEncodingException e) {
                  e.printStackTrace();
              }
              Log.d("VALOR",value);
          }
          numero_ESP32 = value;

          SharedPreferences sharedPreferences = getSharedPreferences("ESP32",
Context.MODE_PRIVATE);
          SharedPreferences.Editor editor = sharedPreferences.edit();
          editor.putString("ESP32",numero_ESP32);
          editor.commit();

          SmsManager sms = SmsManager.getDefault();
          ArrayList<String> msgArray = sms.divideMessage(chave + "," + numero_emerg + "," +
endereco_lat + "," + endereco_long);
          sms.sendMultipartTextMessage(numero_ESP32,null,msgArray,null,null);

          runOnUiThread(new Runnable() {
              public void run() {
                  Toast.makeText(getBaseContext(), "Informações enviadas ao dispositivo",
Toast.LENGTH_SHORT).show();
              }
          });

          gatt.disconnect();
      }

      @Override
      public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
          super.onConnectionStateChange(gatt, status, newState);
          switch (newState){
              case 0:
                  Log.d("CONEXAO","Desconectado");

                  Intent intent = new Intent(getBaseContext(), MainActivity.class);// New activity
                  intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                  startActivity(intent);
```

```
                finish();
                break;
            case 2:
                Log.d("CONEXAO","Conectado");
                gatt.discoverServices();
                break;
            default:
                Log.d("CONEXAO","Estranho");
                break;
        }
    }
};

private BluetoothDevice encontrar_ESP() {
    BluetoothDevice device = null;
    btAdapter = BluetoothAdapter.getDefaultAdapter();
    Set<BluetoothDevice> pairedDevices = btAdapter.getBondedDevices();
    for(BluetoothDevice btd:pairedDevices){
        if(btd.getName().equals("MyESP32")){
            device = btd;
            Log.d("ENCONTRAR ESP","Encontrei");
            return device;
        }
    }
        return device;
}

private void encontrar_service(List<BluetoothGattService> services_list, BluetoothGatt gatt) {
    if(services_list == null) return ;

    for(BluetoothGattService service:services_list){
        List<BluetoothGattCharacteristic> characteristic_list = service.getCharacteristics();
        for(BluetoothGattCharacteristic characteristic:characteristic_list){
            if(characteristic.getUuid().toString().equals(chave)){
                gatt.readCharacteristic(characteristic);
                Log.d("ENCONTRAR SERVICE","Encontrei Service");
                return;
            }
        }
    }
    return;
}
}
```

EditarContato

```
public class EditarContato extends AppCompatActivity {

    EditText et_nome, et_telefone;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_editar_contato);

        et_nome = findViewById(R.id.nome_cadastro);
        et_telefone = findViewById(R.id.telefone_cadastro);

        Button b_cadastro = (Button) findViewById(R.id.b_cadastro); // Informar estado
```

```
        b_cadastro.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                //SharedPreferences sharedPref = getPreferences(MODE_PRIVATE);
                SharedPreferences sharedPreferences = getSharedPreferences("nome_contato",
Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("nome_contato",et_nome.getText().toString());
                editor.commit();

                sharedPreferences = getSharedPreferences("telefone_contato", Context.MODE_PRIVATE);
                editor = sharedPreferences.edit();
                editor.putString("telefone_contato",et_telefone.getText().toString());
                editor.commit();

                Toast.makeText(v.getContext(), "Contato atualizado!", Toast.LENGTH_LONG).show();

                Intent intent = new Intent(getBaseContext(), MainActivity.class);// New activity
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                finish();
            }
        });
    }
}
```

EditarEndereco

```
public class EditarEndereco extends AppCompatActivity {

    EditText et_endereco;
    Geocoder geocoder;
    List<Address> lista_endereco;
    Address endereco;
    double lat,lng;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_editar_endereco);

        et_endereco = findViewById(R.id.cadastro_endereco_casa);

        Button b_cadastro_endereco = (Button) findViewById(R.id.b_cadastro_endereco); // Informar
estado
        b_cadastro_endereco.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {

                //Salvando no shared preferences
                SharedPreferences sharedPreferences = getSharedPreferences("endereco_casa",
Context.MODE_PRIVATE);
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString("endereco_casa",et_endereco.getText().toString());
                editor.apply();

                //avisando ao usuário que foi cadastrado com sucesso
                Toast.makeText(v.getContext(), "Endereço atualizado!", Toast.LENGTH_LONG).show();

                //enviando endereço atualizado ao ESP32
```

```
            geocoder = new Geocoder(getBaseContext());
            try {
                lista_endereco = geocoder.getFromLocationName(et_endereco.getText().toString(),1);
            } catch (IOException e) {
                e.printStackTrace();
            }
            endereco = lista_endereco.get(0);
            lat = endereco.getLatitude();
            lng = endereco.getLongitude();

            sharedPreferences = getSharedPreferences("endereco_casa_lat",
Context.MODE_PRIVATE);
            editor = sharedPreferences.edit();
            editor.putString("endereco_casa_lat", String.valueOf(lat));
            editor.apply();

            sharedPreferences = getSharedPreferences("endereco_casa_long",
Context.MODE_PRIVATE);
            editor = sharedPreferences.edit();
            editor.putString("endereco_casa_long", String.valueOf(lng));
            editor.apply();

            Intent intent = new Intent(getBaseContext(), MainActivity.class);// New activity
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            finish();


        }
    });
  }
}
```

EstadoPaciente

```
public class EstadoPaciente extends AppCompatActivity {

  String str_telefone_emerg,str_ESP32, longitude, latitude; //localizacao_atual;
  double d_latitude,d_longitude;
  private FusedLocationProviderClient mFusedLocationClient;


  @SuppressLint("MissingPermission")
  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_estado_paciente);

      SharedPreferences sharedPref = getSharedPreferences("telefone_contato",
Context.MODE_PRIVATE);
      str_telefone_emerg = sharedPref.getString("telefone_contato", "sem_cadastro");

      sharedPref = getSharedPreferences("ESP32",Context.MODE_PRIVATE);
      str_ESP32 = sharedPref.getString("ESP32","ESP32_nao_conectado");

      Button b_sim = (Button) findViewById(R.id.b_sim); // Botão de sim
      b_sim.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
              if(str_ESP32.equals("ESP32_nao_conectado")){
```

```
                Toast.makeText(v.getContext(), "Conecte ao seu dispositivo!",
Toast.LENGTH_LONG).show();
                Intent i = new Intent(v.getContext(), ConectarDispositivo.class);
                startActivity(i);// Vai para Activity Editar Contato
            }
            sendSMS(str_ESP32,"Estou bem!");
            Toast.makeText(getBaseContext(),"Estado enviado!",Toast.LENGTH_LONG).show();
            Intent intent = new Intent(getBaseContext(), MainActivity.class);// New activity
            intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            finish();
        }
    });

    Button b_nao = (Button) findViewById(R.id.b_nao); // Botão de não
    b_nao.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            if(str_telefone_emerg.equals("sem_cadatro")){
                Toast.makeText(v.getContext(), "Cadastre um contato de emergência",
Toast.LENGTH_LONG).show();
                Intent i = new Intent(v.getContext(), EditarContato.class);
                startActivity(i);// Vai para Activity Editar Contato
            }else{
                mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(getBaseContext());
                mFusedLocationClient.getLastLocation().addOnSuccessListener(new
OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        if(location!=null){
                            longitude = " Longitude: " + String.valueOf(location.getLongitude());
                            latitude = " Latitude: " + String.valueOf(location.getLatitude());
                            d_latitude = location.getLatitude();
                            d_longitude = location.getLongitude();
                            Log.d("LOCATION", String.valueOf(d_latitude));
                            Log.d("LOCATION", String.valueOf(d_longitude));
                            sendSMS(str_telefone_emerg, "Socorro! Estou em:
http://maps.google.com/maps?q=" + d_latitude+ "," + d_longitude);
                        }
                    }
                });
                Toast.makeText(getBaseContext(),"Pedido de socorro
enviado!",Toast.LENGTH_LONG).show();
                Intent intent = new Intent(getBaseContext(), MainActivity.class);// New activity
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                finish();
            }
        }
    });
  }

  private void sendSMS(String phoneNumber, String message) {
    SmsManager sms = SmsManager.getDefault();
    ArrayList<String> msgArray = sms.divideMessage(message);
    sms.sendMultipartTextMessage(phoneNumber,null,msgArray,null,null);
  }

}
```

PacientesDO

```
public class PacientesDO {
   private String _userId;
   private Double _location;
   private Double _latitude;
   private Double _longitude;

   @DynamoDBHashKey(attributeName = "userId")
   @DynamoDBAttribute(attributeName = "userId")
   public String getUserId() {
      return _userId;
   }

   public void setUserId(final String _userId) {
      this._userId = _userId;
   }
   @DynamoDBRangeKey(attributeName = "Location")
   @DynamoDBAttribute(attributeName = "Location")
   public Double getLocation() {
      return _location;
   }

   public void setLocation(final Double _location) {
      this._location = _location;
   }
   @DynamoDBAttribute(attributeName = "Latitude")
   public Double getLatitude() {
      return _latitude;
   }

   public void setLatitude(final Double _latitude) {
      this._latitude = _latitude;
   }
   @DynamoDBAttribute(attributeName = "Longitude")
   public Double getLongitude() {
      return _longitude;
   }

   public void setLongitude(final Double _longitude) {
      this._longitude = _longitude;
   }

}
```

ParametrosSeguranca

```
public class ParametrosSeguranca extends AppCompatActivity {

   TextView tv_nome, tv_telefone, tv_endereco;
   String str_nome,str_telefone,str_endereco;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_parametros_seguranca);

      tv_nome = (TextView)findViewById(R.id.nome_contato);
```

```
        tv_telefone = (TextView)findViewById(R.id.telefone_contato);
        tv_endereco = (TextView)findViewById(R.id.endereco_casa);

        SharedPreferences sharedPref = getSharedPreferences("nome_contato",
Context.MODE_PRIVATE);
        str_nome = sharedPref.getString("nome_contato",null);

        sharedPref = getSharedPreferences("telefone_contato", Context.MODE_PRIVATE);
        str_telefone = sharedPref.getString("telefone_contato",null);

        sharedPref = getSharedPreferences("endereco_casa", Context.MODE_PRIVATE);
        str_endereco = sharedPref.getString("endereco_casa",null);

        if(str_nome == null){
            tv_nome.setText("Contato não cadastrado");
            tv_telefone.setText("");
        }else{
            tv_nome.setText(str_nome);
            tv_telefone.setText(str_telefone);
        }

        if(str_endereco == null){
            tv_endereco.setText("Endereço não cadastrado");
        }else{
            tv_endereco.setText(str_endereco);
        }

        Button b_editar_contato = (Button) findViewById(R.id.b_editar_contato); // Informar estado
        b_editar_contato.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent i = new Intent(v.getContext(), EditarContato.class);
                startActivity(i);// Vai para Activity EstadoPaciente
            }
        });

        Button b_editar_endereco = (Button) findViewById(R.id.b_editar_endereco); // Parâmetros de
Segurança
        b_editar_endereco.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent i = new Intent(v.getContext(), EditarEndereco.class);
                startActivity(i);// Vai para Activity ParâmetrosSeguranca
            }
        });
    }

    @Override
    protected void onResume() {
        super.onResume();
        tv_nome = (TextView)findViewById(R.id.nome_contato);
        tv_telefone = (TextView)findViewById(R.id.telefone_contato);
        tv_endereco = (TextView)findViewById(R.id.endereco_casa);

        SharedPreferences sharedPref = getSharedPreferences("nome_contato",
Context.MODE_PRIVATE);
        str_nome = sharedPref.getString("nome_contato",null);

        sharedPref = getSharedPreferences("telefone_contato", Context.MODE_PRIVATE);
        str_telefone = sharedPref.getString("telefone_contato",null);

        sharedPref = getSharedPreferences("endereco_casa", Context.MODE_PRIVATE);
```

```
        str_endereco = sharedPref.getString("endereco_casa",null);

        if(str_nome == null){
            tv_nome.setText("Contato não cadastrado");
            tv_telefone.setText("");
        }else{
            tv_nome.setText(str_nome);
            tv_telefone.setText(str_telefone);
        }

        if(str_endereco == null){
            tv_endereco.setText("Endereço não cadastrado");
        }else{
            tv_endereco.setText(str_endereco);
        }
    }
}
```

SMSReceiver

```
public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "";
        if (bundle != null)
        {
            //---retrieve the SMS message received---
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];
            for (int i=0; i<msgs.length; i++){
                msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                str = msgs[i].getMessageBody().toString();
            }
        }
        if(str.equals("ALERTA")){
            Intent i = new Intent(context, EstadoPaciente.class);
            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(i);
        }
    }
}
```

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.gabri.tccv1">

    <uses-permission android:name="android.permission.RECEIVE_SMS" /> <!-- permissão para
utilizar o Receiver de SMS -->
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"></uses-permission>
```

```
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

  <uses-feature android:name="android.hardware.location.gps" />

  <application
     android:allowBackup="true"
     android:icon="@mipmap/ic_launcher"
     android:label="@string/app_name"
     android:roundIcon="@mipmap/ic_launcher_round"
     android:supportsRtl="true"
     android:theme="@style/AppTheme">
     <activity android:name=".MainActivity">
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />

           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
     </activity>
     <activity android:name=".EstadoPaciente" />

     <receiver android:name=".SMSReceiver">
        <intent-filter>
           <action android:name="android.provider.Telephony.SMS_RECEIVED" />
        </intent-filter>
     </receiver>

     <activity android:name=".ParametrosSeguranca" />
     <activity android:name=".EditarContato" />
     <activity android:name=".EditarEndereco" />

     <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="@string/google_maps_key" />

     <activity android:name=".ConectarDispositivo"></activity>
  </application>

</manifest>
```

build.gradle

```
apply plugin: 'com.android.application'

android {
  compileSdkVersion 27
  defaultConfig {
     applicationId "com.example.gabri.tccv1"
     minSdkVersion 18
     targetSdkVersion 27
     versionCode 1
     versionName "1.0"
     testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }
  buildTypes {
     release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
     }
  }
```

```
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    compile 'com.google.android.gms:play-services-maps:15.0.1'
    compile 'com.google.android.gms:play-services-location:15.0.1'
    implementation ('com.amazonaws:aws-android-sdk-mobile-client:2.6.+@aar') { transitive = true }
    implementation 'com.amazonaws:aws-android-sdk-pinpoint:2.6.+'
    implementation 'com.amazonaws:aws-android-sdk-ddb-mapper:2.6.+'
}
```

Apêndice C

Código Android - Aplicativo de monitoramento

MainActivity

```
public class MainActivity extends AppCompatActivity implements OnMapReadyCallback {
  DynamoDBMapper dynamoDBMapper;
  List<LatLng> listaCoordenadas = null;
  GoogleMap mMap;
  HeatmapTileProvider mProvider;
  TileOverlay mOverlay;
  AmazonDynamoDBClient dynamoDBClient;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
           .findFragmentById(R.id.map);
     mapFragment.getMapAsync(this);

     AWSMobileClient.getInstance().initialize(this).execute();

     AWSCredentialsProvider credentialsProvider =
AWSMobileClient.getInstance().getCredentialsProvider();
     AWSConfiguration configuration = AWSMobileClient.getInstance().getConfiguration();

     // Add code to instantiate a AmazonDynamoDBClient
     dynamoDBClient = new AmazonDynamoDBClient(credentialsProvider);

     this.dynamoDBMapper = DynamoDBMapper.builder()
           .dynamoDBClient(dynamoDBClient)
           .awsConfiguration(configuration)
           .build();

     listaCoordenadas =  queryLocation();

     Button bt = findViewById(R.id.bt_atualizar);
     bt.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           adicionaHeatMap(listaCoordenadas);
        }
     });
  }

  @Override
  public void onMapReady(GoogleMap googleMap) {
     mMap = googleMap;
     mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(-23.5580,-
46.7361),12));
  }

  private void adicionaHeatMap(List<LatLng> lista){
     mProvider = new HeatmapTileProvider.Builder().data(lista).build();
     mOverlay = mMap.addTileOverlay(new TileOverlayOptions().tileProvider(mProvider));
```

```
    Log.d("Problema", "Entrei aqui");
  }

  public ArrayList<LatLng> queryLocation() {
    final ArrayList<LatLng> lista = new ArrayList<LatLng>();
    new Thread(new Runnable() {
      @Override
      public void run() {
        PacientesDO news = new PacientesDO();
        news.setUserId("0");

        Condition rangeKeyCondition = new Condition()
            .withComparisonOperator(ComparisonOperator.GT)
            .withAttributeValueList(new AttributeValue().withN("0"));

        DynamoDBQueryExpression queryExpression = new
DynamoDBQueryExpression().withHashKeyValues(news)
            .withRangeKeyCondition("Location", rangeKeyCondition)
            .withConsistentRead(false);

        PaginatedList<PacientesDO> result = dynamoDBMapper.query(PacientesDO.class,
queryExpression);

        // Loop through query results
        for (int i = 0; i < result.size(); i++) {
          double latitude = result.get(i).getLatitude();
          double longitude = result.get(i).getLongitude();
          lista.add(new LatLng(latitude,longitude));
        }
        if (result.isEmpty()) {
          // There were no items matching your query.
        }
      }
    }).start();
    Log.d("MENSAGEM","Terminei");
    return lista;
  }
}
```

Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.gabri.tccfamilia">

  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="AIzaSyC5K-0jLFpc3JNuxFseCVnKiK-oG1n6gto" />
```

```
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

build.gradle

```
apply plugin: 'com.android.application'

android {
  compileSdkVersion 27
  defaultConfig {
    applicationId "com.example.gabri.tccfamilia"
    minSdkVersion 18
    targetSdkVersion 27
    versionCode 1
    versionName "1.0"
    testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
  }
  buildTypes {
    release {
      minifyEnabled false
      proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
  }
}

dependencies {
  implementation fileTree(dir: 'libs', include: ['*.jar'])
  implementation 'com.android.support:appcompat-v7:27.1.1'
  implementation 'com.android.support.constraint:constraint-layout:1.1.3'
  testImplementation 'junit:junit:4.12'
  androidTestImplementation 'com.android.support.test:runner:1.0.2'
  androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
  implementation 'com.google.android.gms:play-services-maps:16.0.0'
  implementation 'com.amazonaws:aws-android-sdk-ddb-mapper:2.6.+'
  implementation ('com.amazonaws:aws-android-sdk-mobile-client:2.6.+@aar') { transitive = true }
  compile 'com.google.maps.android:android-maps-utils:0.5+'
}
```