UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUIS HENRIQUE MENDONÇA GRASSI

Sistema de monitoramento de vazamento de gás liquefeito de petróleo (GLP) e gás natural (GN) em tempo real.

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. João Cesar Netto

Porto Alegre
2017

"Que a indiferença não te vença"

- *Autor desconhecido*

RESUMO

Não raramente nos deparamos com notícias sobre graves acidentes provocados por vazamentos de gás. Muito deles provocando o colapso de prédios inteiros devido a explosões. Dado esse problema recorrente, foi desenvolvido um sistema para monitoramento de vazamento de gás LP e gás natural em tempo real. O sistema auxilia os usuários alertando-os quando detectado vazamentos ou uso não usual (sistema de aquecimento ligado por um tempo maior que o estabelecido - útil para detectar esquecimentos de chamas acesas, por exemplo). Desenvolveu-se um hardware para monitoramento de vazamentos com comunicação em nuvem. Ainda, desenvolveu-se um aplicativo para a plataforma iOS, no qual o usuário pode consultar e configurar o sistema.

Palavras-chave: Gás. Vazamento. Acidentes. Sistema de monitoramento em tempo real.

Monitoring system for liquid petroleum gas (LPG) and natural gas (NG) leakage in real-time

ABSTRACT

We often come across news of serious accidents caused by gas leaks, many of them causing the collapse of whole buildings due to explosions. Introduced this recurrent problem, a monitoring system of gas GLP and GN leak in real time was developed. The system aims to assist users by alerting them when leakage or unusual use is detected (heating system switched on for a longer time than intended - useful for detecting forgotten flames burning, for example). We have developed leak monitoring hardware with cloud communication. We have also developed an application for the iOS platform, in which the user can consult and configure the system.

Keywords: Gas. Leakage. Accidents. Real-time monitoring system.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ACID	Atomicity, Consistency, Isolation, Durability
ADC	Analog-to-digital
ADO	ActiveX Data Objects
API	Application Programming Interface
APP	Application Software
BaaS	Backend as a Service
BGA	Ball Grid Array
GLP	Gás Liquefeito de cozinha
GN	Gás Natural
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IHM	Interface homem máquina
IIS	Internet Information Services
I/O	Input/Output
IR	Infravermelho
JDBC	Java Database Connectivity
MB	Megabyte
MCC	Controle de Concorrência Multiversão
MCU	Microcontroller
MVC	Model-view-controller
MVP	Minimum Viable Product
MVVM	Model-view-viewModel
ODBC	Open Database Connectivity
PaaS	Platform as a Service
PHP	Hypertext Preprocessor

ppm	Parte por milhão
PWM	Pulse-Width Modulation
RAM	Random Access Memory
REST	Representational State Transfer
RISC	Reduced Instruction Set Computer
ROM	Read-only memory
RX	Receive
SoC	System-On-Chip
SnO2	Dióxido de estanho
SDK	Software development kit
SGBD	Sistema de Gerenciamento de Banco de Dados
SMS	Short Message Service
SO	Sistema Operacional
SOA	Service-Oriented Architecture
SoftAP	Software enabled access point
TX	Transmit
TCP/IP	Transmission Control Protocol/Internet Protocol
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UDF	User Defined Functions
URI	Identificador Uniforme de Recurso
UI	User Interface
UX	User Experience
WSDL	Web Services Description Language
WCF	Windows Communication Foundation
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O gás GLP, conhecido popularmente como gás de cozinha, tem um papel importante no Brasil. Além de representar 3,2% da nossa matriz energética [1], estima-se que cerca de 195 milhões de brasileiros o utilizam direta ou indiretamente. Essa fonte de energia é vendida em botijões e está dentro de 95% dos domicílios, com cerca de 33 milhões de unidades comercializadas mensalmente em todo o país [1].

Por ser de fácil armazenagem e transporte, tem uma versatilidade que vai além dos isqueiros ou dos fogões à gás: é usado em equipamentos de solda e corte; no ramo industrial como combustível para empilhadeiras; em lavanderias como fonte de calor para secagem de roupas; no ramo agrícola para secagem de grãos, e também para incineração de lixo.

Outro gás muito utilizado, porém, não tão popular, é o gás natural. Este representa 7,2% [1] da matriz energética brasileira. Seus usos vão dos mais simples - como a utilização em fogões domésticos e combustível veicular, substituindo o etanol ou a gasolina - até usos mais complexos, nas indústrias químicas, como na elaboração de compostos como o metanol e a ureia. Por ser uma fonte mais limpa que os derivados do petróleo, como o gás LP, tem recebido grandes investimentos no Brasil para expansão de seu uso.

No entanto, apesar de todos os investimentos, o Brasil aparece apenas na 30ª posição entre os maiores consumidores de GN. Já quando falamos de consumo de GLP, aparecemos logo após da Tailândia, na 10ª posição [5].

Por estar presente no nosso cotidiano, muitas vezes acabamos negligenciando cuidados básicos de segurança na utilização dessas fontes de energia. Apesar de não serem tóxicos, ambos podem ser perigosos quando não as normas de segurança não são seguidas. Por exemplo, se inalados, ambos podem causar asfixia e levar à morte, sendo que o gás LP, por ser mais denso do que ar, é mais perigo. Já o gás natural se dissipa facilmente por ter uma densidade mais baixa do que o ar o deixando menos propício a explosões, [4] outro grande risco.

Abaixo alguma manchetes e fotos demonstrando a gravidade desses acidentes.

Figura 1.1 - Explosão por vazamento de gás deixa uma vítima fatal

Fonte: Band News

Figura 1.2 - Vazamento de gás causa explosão e deixa seis pessoas feridas

Fonte: A Crítica - Jornal Digital

Figura 1.3 - Apartamento explode após vazamento de gás; moradora sofre queimaduras em todo o corpo

Fonte: O Povo

Figura 1.4 - Vazamento de gás causa explosão, prédio desaba, cinco vítimas

Fonte: O Tempo

Figura 1.5 - Vazamento de gás interrompe VLT e esvazia dois prédios no centro do Rio

Fonte: O Globo

Como citado, o problema não se restringe às nossas cozinhas: vazamentos de gás podem ocorrer nos mais variados locais colocando nossas vidas em risco sem nem mesmo percebermos.

1.1 Motivação

Existem dezenas de soluções de monitoramento de vazamento de gases, entretanto não se encontrou nenhuma que explore os recursos disponíveis nos dias de hoje, tais como interações ricas com os usuários (através dos *apps*) e disponibilidade ampla de internet. É de interesse do projetoque se faça uso de todos os recursos disponíveis atualmente, para melhorar a qualidade de vida das pessoas, ainda mais quando a questão é segurança.

O empreendedorismo também atuou como fonte de motivação: por ser uma solução simples e acessível, acredita-se que exista um bom mercado para se comercializar.

1.2 Objetivo

O objetivo desse trabalho é desenvolver um sistema de monitoramento de vazamentos e uso não usual em tempo real. Para ser de amplo acesso, deverá ser de fácil instalação e configuração, não deverá custar mais que algumas dezenas de reais e não poderá ser intrusivo: todo o processo de monitoramento deverá ser passivo, não podendo alterar de forma alguma o funcionamento de outros componentes como válvulas, reguladores de pressão ou mangueiras. Isso exigiria certificação de órgãos reguladores o que encareceria o processo, provavelmente inviabilizando-o.

1.3 Estrutura do texto

No próximo capítulo será apresentado os recursos utilizados e seus conceitos para o desenvolvimento do trabalho. O capítulo de implementação detalha como os recursos foram utilizados para se alcançar o objetivo. No final, tem-se os resultados e as conclusões.

2 RECURSOS

Todos os recursos utilizados no projeto são descritos a seguir com seus respectivos conceitos e definições. Apresenta-se primeiramente uma definição geral do recurso e logo após detalha-se o utilizado.

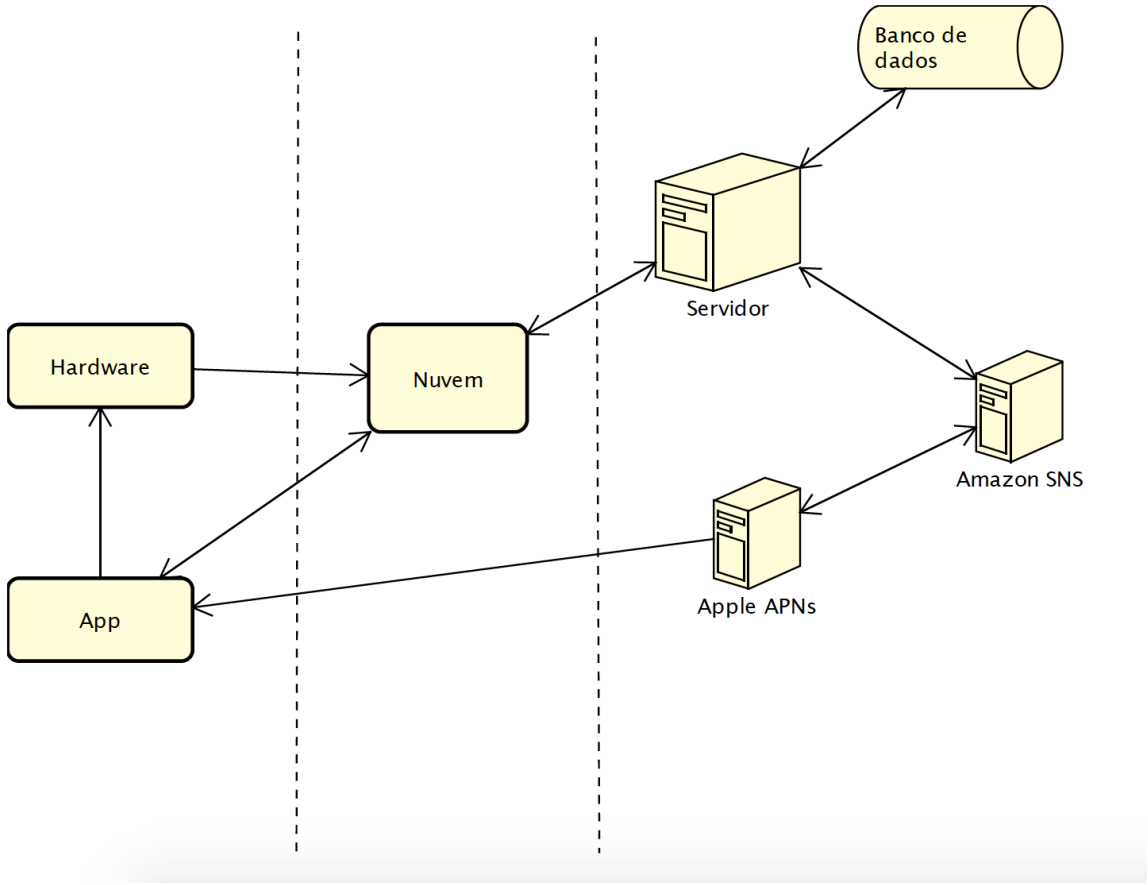

Figura 2.1- visão geral dos recursos

2.1 Microcontroladores

Microcontroladores *system-on-chip* [17] (SoC) são dispositivos, que em adição a um processador, combinam vários outros componentes e elementos periféricos como memória ROM, RAM, módulos wireless (como Wi-Fi e *bluetooth*), *clock*, BUS, GPU, IOs e muito outros dependendo do modelo e fabricante. Basicamente, um SoC contém todos os componentes necessários para o seu funcionamento integrados em um único chip.

Devido ao seu baixo custo e praticidade, são largamente utilizados em sistemas embarcados de controle e automação, dispositivos médicos, eletrodomésticos e controles automotivos. É praticamente impossível acharmos um dispositivo eletrônico que não possua ao menos um SoC, normalmente vários.

2.1.1 Família ESP8266EX

Os microprocessadores da família ESP8266EX [14] da fabricante chinesa Espressif Systems contam com uma MCU 32-bits RISC da fabricante norte-americana Tensilica. A memória RAM e a flash dependem do modelo.

Figura 2.1- ESP-07 com antena de cerâmica

Possuem até 16 GPIO, sendo que alguns modelos possuem IO analógicos, e módulo Wi-Fi integrado com receptor e transmissor de 2.4 GHz, pilha TCP/IP integrado, e com suporte para os padrões 802.11 b/g/n/e/i. O tipo de antena também depende do escolhido. Lançado em 2014, e com quase nenhuma documentação em inglês naquela época, o ESP8266EX vem ganhando cada vez mais espaço, principalmente devido ao seu baixo custo. Houve um grande esforço dos adeptos da cultura *maker culture* [18] para gerar documentação, contudo, hoje a própria fabricante disponibiliza *datasheets* em inglês. Possui SDK de desenvolvimento próprio, porém pode-se utilizar outras IDEs como do Arduino [19] ou até mesmo o xCode [20] da Apple.

Os modelos variam do ESP-01 até o ESP-13, todos com características próprias variando de quantidade de IOs, tipo de encapsulamento da antena e da MCU, tamanho da *flash* a recursos como *watchdog* timer e *deep sleep*.

A família ESP8266EX não é própria para prototipagem, começando pelo seu tamanho (5cm x 5cm) que dificulta o manuseio e não é *breadboard friendly*. Além disso, não possui controlador

USB, apenas controlador serial UART, o que exige um adaptador para fazer o *upload* da aplicação visto que a grande maioria dos computadores atuais não possuem porta serial. Alguns modelos, como o ESP-06 possuem os conectores na parte de baixo, no estilo BGA (*Ball Grid Array*).

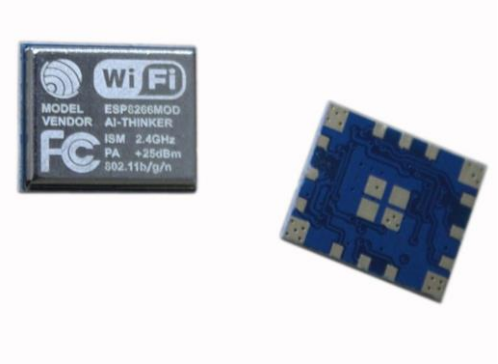

Figura 2.2 - ESP-06 com conectores BGA

2.1.1.1 ESP-07

O modelo ESP-07 [21] é um dos mais difundidos da família ESP8266EX. Versátil e com uma boa relação custo-benefício, ele possui antena de cerâmica e ainda um conector U-FL caso se queira adicionar uma antena externa e MCU RISC de 32 bits rodando a 80 MHz. O chip possui encapsulamento metálico, o que garante uma boa proteção contra interferências do ambiente e já vem com o protocolo TCP/IP integrado, suporte para os padrões 802.11 b/g/n/e/i, 1 MB de memória FLASH; expansível até 16 MB. Possui 9 portas GPIO (sendo 4 com suporte a PWM), 1 ADC com 10 bits de precisão. Com dimensões de 21,2 x 16mm e tensão de operação de 3,3V, possui um consumo de 50 mW em modo *modem-sleep*, porém alcance 560 mW quando transmitindo dados.

2.2 Sensores

Sensores [22] são dispositivos que podem responder a estímulos físicos ou químicos gerando uma saída de dados que pode ser interpretada por outros componentes. Existe uma infinidade de tipos de sensores, tais como: sensores de som, sísmicos, calor, radiação, orientação dentro muitos outros. Normalmente, sensores atuam como gatilhos gerando ações específicas no sistema, desde ativar um dispositivo de segurança (*airbag* em veículos, por exemplo), disparar um sinal sonoro de alerta ou simplesmente atuar para capturar informações para uma posterior análise (por exemplo, estações de coleta de qualidade do ar).

Figura 2.3 - alguns sensores da família MQ

2.2.1 Sensor de gases MQ-05

O sensor MQ-05 faz parte da série MQ [23], uma família de sensores do tipo semicondutores sensíveis a uma grande gama de tipos de gases. Em específico, o modelo MQ-05 tem uma alta sensibilidade ao gás de cozinha (GLP) e ao gás natural (GN). Ainda, mas com menor precisão, é capaz de detectar fumaça e álcool. O MQ-05 possui um fino filamento chamado pérola, feito de dióxido de estanho (SnO2). Quando aplicado a uma fonte de tensão, este filamento mantém uma resistência elétrica estável na presença de ar, porém, na presença dos gases especificados acima, o filamento sofre uma pequena oxidação provocando uma diminuição na resistência elétrica do semicondutor e, consequentemente um aumento na corrente elétrica.

Aplicando-se uma carga resistiva na saída do sensor, é possível calcular a diferença de tensão sobre a carga devido a variação da corrente elétrica. Essa diferença pode ser usada para se saber a concentração de gás presente. O sensor trabalha com tensão de 5V e tem um burn-in (tempo mínimo que precisa ficar ligado para ter uma resposta mais consistente) de 24h. Isso se dá ao fato da pérola precisar estar aquecida para que na presença dos gases, sofra oxidação. O sensor pode medir concentrações de gases inflamáveis na faixa de 300 a 10.000 ppm (quanto maior a concentração menor a resistência elétrica da pérola), opera em temperaturas de -20 a 50ºC e consome menos de 150mA a 5V.

Figura 2.4 - sensor MQ-05

2.2.2 Sensores infravermelhos

Luz infravermelha (IR) é um tipo de radiação eletromagnética, assim como as ondas de rádio e a radiação ultravioleta. A luz IR faz parte do espectro eletromagnético invisível ao olho humano, porém pode ser sentida na forma de calor. Qualquer objeto com temperatura superior a 268 graus Celsius negativos emite luz IR.

Os sensores IR são dispositivos com um fotodiodo semicondutor capaz de converter luz em corrente elétrica através do efeito fotoelétrico. Alterando as propriedades químicas dos fotodiodos, é possível limitar a que faixas de comprimento de onda estes serão sensíveis.

Figura 2.5 - LED IR receptor

Figura 2.6 - circuito IR receptor

2.2.3 Sensores de som

Sensores de som são dispositivos, popularmente conhecidos como microfone, que convertem variação de pressão acústica em sinais elétricos. Existem várias tecnologias para converter a energia mecânica (o som) em sinais elétricos, mas todas compartilham do mesmo princípio: o diafragma. O diafragma é uma peça fina (de papel, plástico ou até mesmo de alumínio) que vibra quando é atingida pelas ondas sonoras. Essa vibração é transmitida para outro dispositivo que vai fazer a conversão. Como essa será feita dependerá da tecnologia, podendo ser por indução eletromagnética (chamado microfone dinâmico), ou por diferença capacitiva (chamado microfone condensador), por exemplo.

2.3 Banco de dados relacional

Bancos de dados relacionais [29] podem ser considerados como coleções de tabelas digitais e suas as relações entre si nas quais os registros são armazenados. Cada linha (tupla) em uma tabela é um registro. Cada tupla consiste em um conjunto de valores de atributos (também chamados propriedades), que são as colunas da tabela.

BOOKING_ID	PERSON_ID	HOSTEL_ID_ORIGIN	HOSTEL_ID_DESTINATION	CAMINO...	TOTAL_DISTANCE	TOTAL_VALUE	INSERT_DATE	UPDATE_DATE
12	13	15	17	1	4,900	7.000	12.04.2016 20:29	<null>
13	2	18	19	1	8,100	7.000	12.04.2016 20:31	<null>
14	13	14	17	1	4,900	7.000	12.04.2016 20:32	19.04.2016 16:00
15	13	17	18	1	2,800	7.000	12.04.2016 20:33	<null>
16	11	15	17	1	4,900	14.000	13.04.2016 02:03	<null>
17	11	18	19	1	8,100	14.000	13.04.2016 02:04	<null>
18	11	14	20	1	19,300	14.000	13.04.2016 02:07	19.04.2016 12:38
19	11	17	19	1	10,900	42.000	13.04.2016 02:08	<null>
20	11	17	20	1	14,400	56.000	13.04.2016 02:10	<null>
21	2	14	17	1	4,900	14.000	13.04.2016 18:59	14.07.2016 10:27
22	2	14	17	1	4,900	7.000	13.04.2016 19:01	19.04.2016 17:00
23	11	15	17	1	4,900	7.000	13.04.2016 20:40	<null>
24	11	15	17	1	4,900	14.000	13.04.2016 20:51	<null>
25	19	14	17	1	4,900	7.000	14.04.2016 13:26	19.04.2016 12:38
26	20	14	17	1	4,900	7.000	14.04.2016 14:16	19.04.2016 12:37
27	21	14	17	1	4,900	7.000	14.04.2016 14:19	20.04.2016 14:15
28	21	14	17	1	4,900	7.000	14.04.2016 16:57	20.04.2016 16:44
29	21	14	17	1	4,900	7.000	15.04.2016 13:31	19.04.2016 17:00
30	13	15	17	1	4,900	0.000	15.04.2016 14:13	<null>
31	13	17	18	1	2,800	0.000	15.04.2016 14:14	<null>
32	13	15	17	1	4,900	7.000	15.04.2016 15:33	<null>
33	21	15	17	1	4,900	14.000	15.04.2016 18:02	<null>
34	13	17	18	1	2,800	21.000	15.04.2016 18:30	<null>
35	13	17	18	1	2,800	14.000	15.04.2016 19:01	<null>

Figura 2.7 - exemplo de uma tabela

Tabelas nada mais são do que simples conjuntos de linhas que compartilham as mesmas colunas. As associações entre tabelas são feitas através de seus atributos, que nesse caso serão chamados de chaves. As chaves podem ser de dois tipos:

- Chave primária (*primary key*): é o identificador exclusivo de cada registro não podendo se repetir na tabela. A chave primária pode ser simples, ou seja, apenas um atributo, ou composta - quando é formada por mais de um atributo.
- Chave estrangeira (*foreign key*): é um atributo usado para relacionar duas tabelas. A chave estrangeira é a chave formada através do relacionamento com a chave primária de outra tabela, e, diferentemente da chave primária, pode ocorrer repetidas vezes.

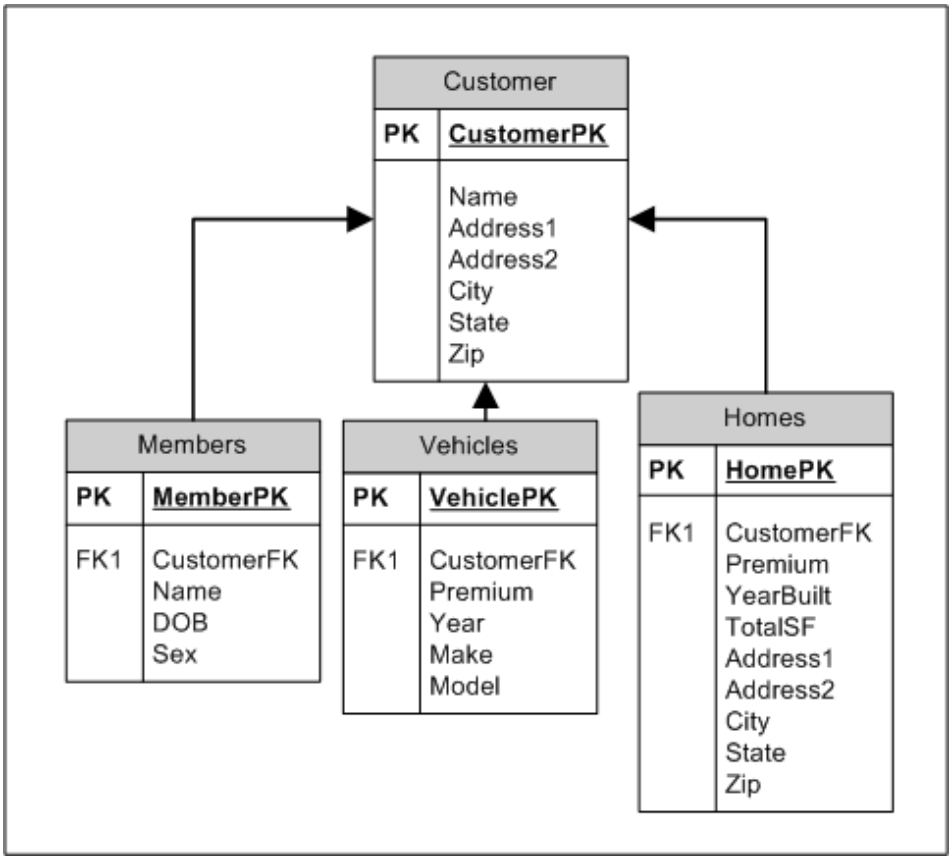

Figura 2.8 - exemplos de FKs e PKs

2.4 Firebird

Firebird é um sistema de gerenciamento banco de dados *opensource* que roda em Linux, Windows, Mac OS e em uma variedade de plataformas Unix. O projeto foi baseado no sistema Interbase da antiga Borland, hoje pertencente à empresa norte-americana Embarcadero Technologies.

A primeira versão do Firebird (1.0) foi lançada em 2000 já com suporte para processadores de 64 bits. A versão 1.0 foi baseada no Interbase 6.0, porém foi totalmente reescrita em C++.

Somente em 2016, já na versão 3.0, que o Firebird ganhou total suporte para processadores com multicores. Desde sua primeira versão, oferece recursos como: compatibilidade ACID, transações MCC, *triggers*, *stored procedures*, *collations*, UDFs e *transactions*.O Firebird possui três versões: Super Server, Classic Server e Embedded Server.

- super server: na versão *Super Server*, cada requisição é uma *thread*, porém existe apenas um processo.

- classic server: na versão *Classic Server*, cada requisição é um processo. Essa versão é indicada para servidores com multicores.
- embedded: na versão *Embedded*, existe apenas um processo. Não é possível fazer acessos paralelos. Essa versão é indicada para aplicações que não requerem acesso multiusuário.

2.5 Web Services

Web services [24][25][26][27] é um solução que possibilita a comunicação entre diferente aplicações utilizando qualquer protocolo de comunicação (normalmente HTTP e/ou HTTPS) através da internet. *Web services* são serviços hospedados em algum computador e podem ser acessados através de seu identificador único de recurso (URI). Cada *web service* possui uma interface (WSDL - *Web Services Description Language*) que define quais e que tipos de dados são esperados, tão como que e quais os tipos de dados são retornados.

Os principais benefícios em usar *web services* são:

- por ser um padrão aberto, evita custos com licenças de aplicações proprietárias.
- por não serem dependentes a nenhum protocolo de comunicação, podem ser utilizados por qualquer tipo de aplicação e cenário.
- é possível que novas aplicações possam interagir com aquelas que já existam e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis.

2.6 Aplicativos móveis

Aplicativos móveis, popularmente conhecidos como *app*s, são um tipo de software desenvolvido especialmente para rodar em dispositivos *smarts*, como telefones, *tablets* e *watches*.Originalmente, os *apps* abordavam os aspectos de produtividade apenas, como *apps* para *email*, calendário e contatos. Porém, com a popularização dos *smartphones*, houve uma rápida expansão na gama de categorias disponíveis como: Notícias, Esportes, Utilidades, Música, Entretenimento, Negócios, Livros, Educação, Jogos dentre várias outras.

O Desenvolvimento desse tipo de software exige algumas considerações especiais. Por serem dispositivos móveis, existe uma limitação no tempo que a bateria dura. Também se deve

levar em consideração que os recursos computacionais - processador, memória, capacidade de armazenamento - são muito inferiores quando comparados a computadores tradicionais.

Outro ponto de extrema importância é com a interface de usuário (UI). Deve se ter um trabalho minucioso de experiência de usuário (UX) devido ao tamanho limitado da tela. O profissional responsável pela criação não deve limitar-se apenas com a disposição da informação, mas também com cores, formas e tipos de interações possíveis com o usuário.

Um erro muito comum das empresas é apenas reescrever seus softwares para que rodem em telefones sem repensar a aplicação. Um *app* não precisa conter todas as funcionalidades e informações disponíveis na aplicação original (muito comum de se ver em aplicativos de bancos e instituições financeiras) ele deve conter apenas as mais importantes e usadas para tornar a experiência do usuário mais agradável. No caso de se querer ter todas as funcionalidades disponíveis em aplicações móveis, pode-se criar mais de um *app* e separar as funcionalidades.

A distribuição dos *apps* é feita pelas lojas virtuais das empresas desenvolvedoras dos sistemas operacionais dos telefones. As duas maiores lojas são a Google Play, da Google e a Play Store, da Apple. Qualquer *app*, para que seja distribuído de forma oficial, precisa ser submetido e aprovado nas lojas. Ambas as lojas têm políticas e regras especificando o que é permitido, sendo a da Apple a mais restritiva.

3 IMPLEMENTAÇÃO

Objetivando desenvolver um sistema de monitoramento de vazamentos de gases e uso não usual em tempo real, optou-se por implementar todo o processo de monitoramento de forma não intrusiva utilizando-se sensores de vazamento de gases e sensores infravermelhos. A comunicação com o *app* é feita através de um servidor na nuvem que recebe eventos do hardware que monitora o ambiente.

Figura 3.1 - visão geral do sistema

3.1 Visão geral

O sistema pode ser dividido em três partes:

- hardware, responsável pelo monitoramento contínuo do ambiente: é o gatilho de eventos (vazamento ou uso não usual);

- serviço IaaS na nuvem - responsável por centralizar a comunicação entre o hardware e o aplicativo móvel;
- *app*, usado como IHM.

Cabe ressaltar, ainda, que o projeto é visto como um mínimo produto viável (MVP). MVP é uma técnica utilizada que visa colocar um produto no mercado o mais rápido possível a fim de colher feedbacks de usuários reais. No MVP, foca-se exclusivamente nos recursos mínimos necessários para o produto desempenhar suas funções. É muito comum, e um grande erro, empresas ficarem meses ou até anos, desenvolvendo um produto para só depois descobrirem que a solução não resolve o problema, ou então que o produto é inviável por inúmeros fatores externos que só podem ser descobertos com o produto já em campo. Por fim, é frequente produtos serem lançados com o principal objetivo de executar uma determinada tarefa para resolver um estipulado problema, e acabarem por serem utilizados de uma forma totalmente diferente pelos usuários finais.

Essas informações são muito importantes para que na próxima versão do produto, se foque nas funcionalidades que os usuários realmente estão usando. Pode-se ainda, chegar à conclusão de que o produto não deve continuar, seja por ser técnica e/ou financeiramente inviável, seja por não ter tido uma aceitação pelo mercado. Nesse caso, soube-se da forma mais rápida possível, gastando-se o mínimo de recursos (dinheiro e tempo).

3.2 Funcionamento do sistema

Tendo em vista que a solução visa atender a uma variada gama de pessoas, com diferentes *backgrounds*, tentou-se desenvolver uma aplicação simples e intuitiva.

Para o funcionamento, o hardware, também denominado módulo monitor, ou apenas módulo, precisa estar conectado em uma rede Wi-Fi com acesso à internet. Quando não configurado, o módulo se comporta como um dispositivo softAP (*Software enabled access point*). SoftAP é quando um dispositivo que originalmente não foi projetado para ser um roteador atua como tal através de um software. No modo softAP, o módulo monitor cria uma rede Wi-Fi chamada "*safeHome*" aberta. Assim, basta conectar o telefone nela, ir no *app* na tela de configuração de módulos, informar o nome e a senha da rede Wi-Fi com acesso à internet e apertar "Salvar". O módulo após receber essa informação irá se conectar na rede informada e irá reiniciar, dessa vez desempenhando suas funções de monitoramento.

A funcionalidade de configuração de módulos só ficará disponível após o usuário realizar a autenticação no *app*. Caso ele já possua cadastro, poderá realizar o *login* informando o usuário e a senha, caso contrário, o cadastro poderá ser feito diretamente no *app*. Uma vez logado, ele tem acesso à tela de ambiente e a função de configuração de módulos. Ambientes são agrupadores de módulos. Estes podem ser de dois tipos: detecção de vazamentos de gás ou alerta de uso não usual. Um exemplo típico de um ambiente seria: "Cozinha". Podem ser cadastrados tantos ambientes quantos forem desejados.

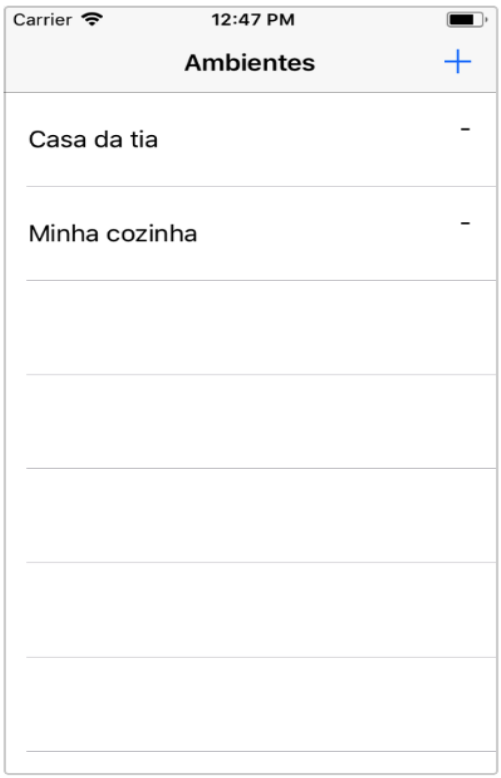

Figura 3.2 - tela com a lista de ambientes

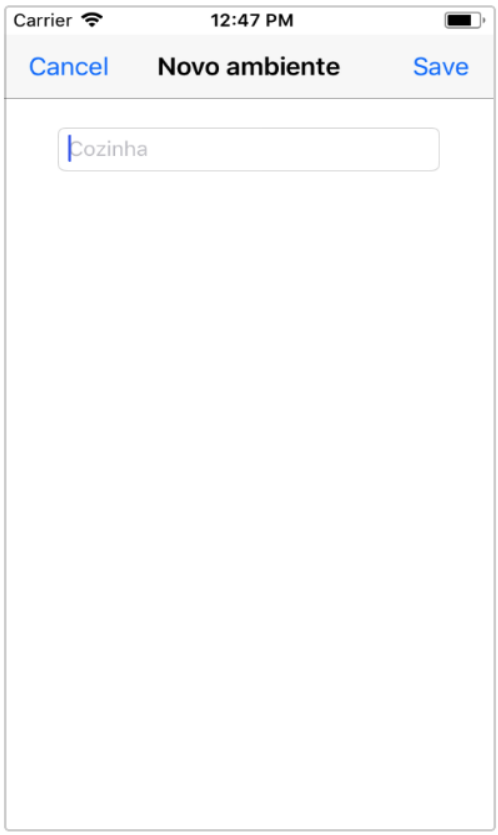

Figura 3.3 - tela de cadastro de ambientes

Ao acessar um ambiente, são listados os módulos presentes nele e o último evento de cada módulo, se existirem. Ainda é possível adicionar mais módulos e/ou excluir os já existentes. Em uma visão mais arquitetural do sistema, módulos não estão relacionados diretamente a usuários, ou seja, um usuário não possui módulos. Todos os módulos existentes já estão pré-cadastrados no banco de dados. Logo, o que de fato ocorre quando um usuário adiciona um módulo a um ambiente é um registro na tabela relacional entre o ambiente e o módulo. Ademais, uma descrição deve ser dada a essa relação, como por exemplo, "sensor do fogão".

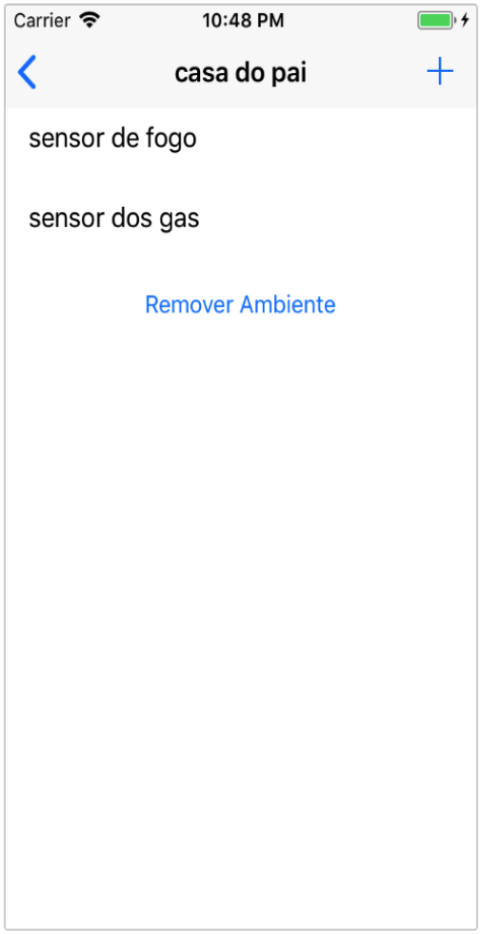

Figura 3.4 - tela com a lista de módulos

Esta estrutura permite que o mesmo módulo seja monitorado por mais de um usuário ao mesmo tempo. Mais que isso, cada usuário pode colocar a descrição que mais lhe convém no módulo. Por exemplo: considere dois módulos: o primeiro de detecção de vazamento com identificador único "1234", o segundo de uso não usual com identificador "1235" e dois usuários, "João" e "Maria". Os módulos estão fisicamente instalados na casa do pai de João, "Gustavo". Gustavo é tio de Maria. Nesse cenário, uma possível configuração seria: João, no seu usuário, cria o ambiente "Casa do pai" e adiciona os dois módulos através de seus identificadores únicos neste ambiente com as seguintes descrições: "Sensor de gás cozinha" e "Sensor do fogão". Maria, por sua vez, cria o ambiente "Casa do tio Gustavo" e adiciona os mesmos módulos com as descrições que lhe convém: "Sensor gás" e "Sensor fogão".

Quando qualquer um dos dois sensores na casa do Gustavo disparar um evento, um vazamento, por exemplo, tanto João quanto Maria receberão uma notificação de alerta com as respectivas mensagens: "Vazamento de gás detectado no ambiente 'Casa do pai', sensor 'Sensor de

gás cozinha'" e "Vazamento de gás detectado no ambiente 'Casa do tio Gustavo', sensor 'Sensor de gás'". O módulo responsável pelo monitoramento envia uma mensagem para o serviço que está rodando na nuvem com o identificador único do módulo. Através desse identificador, o sistema recupera todos os ambientes relacionados a este módulo. Através dos ambientes, é possível identificar os usuários para os quais necessita-se enviar os alertas. Cabe ressaltar, que, somente os usuários que autorizaram receber *push notifications* irão receber os alertas. O sistema operacional iOS tem várias restrições quando o assunto é relacionado à privacidade do usuário. Para mandar notificações, é necessário pedir autorização para o usuário. Se for autorizado, um *token* que identifica o telefone para aquele *app* é gerado. Quando se deseja enviar um *push* para este telefone, é necessário informar o *token* previamente gerado.

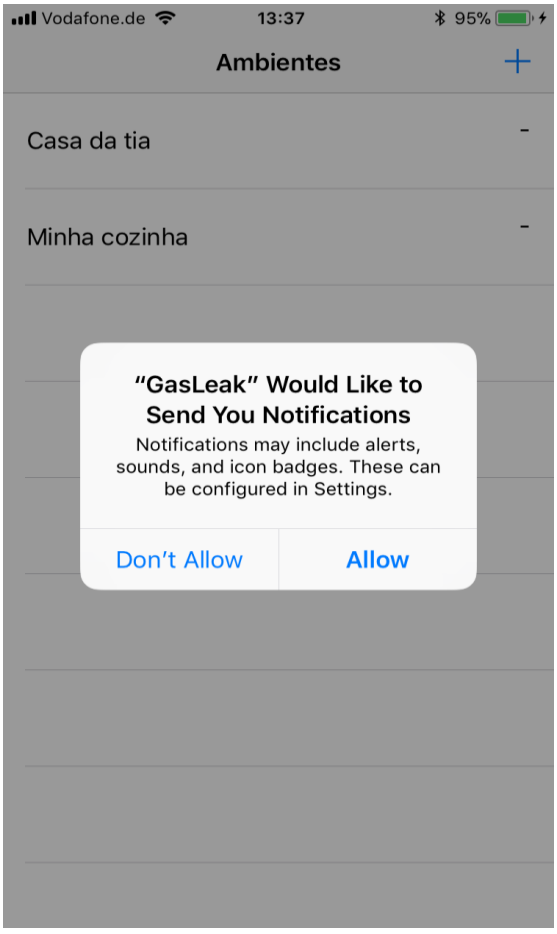

Figura 3.5 - Figura de pedido de autorização para push notifications

Para adicionar um módulo em um ambiente, o usuário precisa informar, além da descrição, o identificador único (ID) presente no *QR Code* que acompanha o módulo (é possível ler o *QR Code* com a câmera do próprio telefone).

O ID é uma sequência alfanumérica única no sistema de 128 caracteres gerada randomicamente que identifica o módulo. Os dois primeiros caracteres do ID são o identificador do tipo de sensor presente no módulo: 'SG' para o sensor de gás e 'SU' para o sensor de uso não usual. Quando adicionando um módulo com sensor de uso não usual, o usuário precisa ainda informarqual o tempo mínimo de detecção para que o sensor dispare o evento. Por exemplo, se for informado 30 minutos, só após este tempo de detecção contínua que será gerado um evento de uso não usual. Neste contexto, entende-se por "detecção contínua" períodos de tempo contínuo ou com intervalos de no máximo um minuto.

Se o sensor ficar mais de um minuto sem detectar eventos, o *timer* é zerado. Esta abordagem funciona como um filtro para evitar que pequenas interferências atrapalhem o correto funcionamento do sistema, como uma pessoa caminhando e cortando momentaneamente o campo de visão do sensor, por exemplo.

Figura 3.6 - tela de cadastro de módulos

3.3 Aplicativo móvel

O app foi desenvolvido nativamente para a plataforma iOS utilizando a linguagem de programação Swift 4.0 e a IDE Xcode 9.2 com suporte para iOS 10+. O código foi estruturado com a arquitetura MVVM [28]. Essa arquitetura permite uma boa organização do código desacoplando a camada de apresentação (*view*) das demais camadas, facilitando, assim, a manutenção do código e adição de novos recursos. Apesar da arquitetura MVC também ser uma boa opção - todo o *framework* provido pela Apple é estruturado em MVC - e o MVVM apresentar um pequeno *overhead* devido à comunicação entre as camadas *view* e *view model*, o MVVM possui vantagens devido ao seu alto grau de desacoplamento, tornando mais fácil, por exemplo, a adição de testes automatizados, porém teste automatizados ficaram fora do escopo, até porque o aplicativo móvel é apenas um MVP não justificando a adição de testes automatizados.

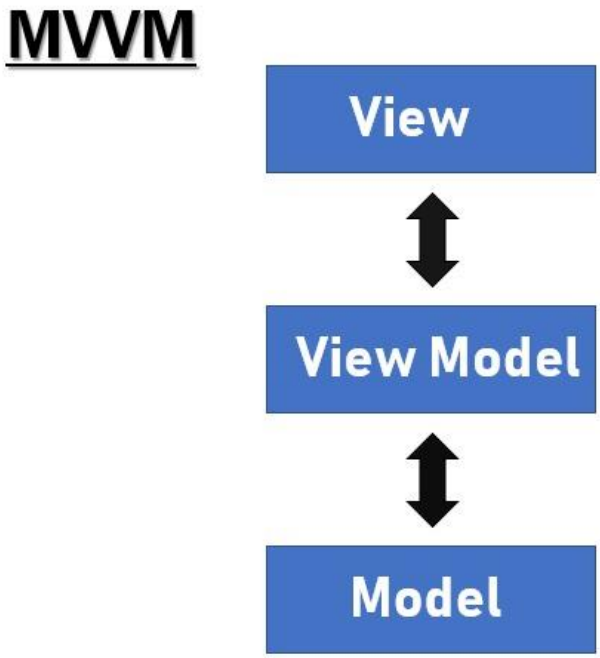

Figura 3.7 - comunicação típica MVVM

O *app* pode ser divido em quatro camadas: *view*, *view model*, *model* e *Networking*. Em uma arquitetura MVVM, a *View* não deve nunca acessar a camada *model* diretamente, e vice-versa. A *view model* é usada como interface entre as duas camadas conforme demonstrado na figura 3.7. Ainda, não existe definição da camada *Networking*. É muito comum se achar projetos onde a todaa

comunicação é feita dentro da *view model*, porém escolheu-se separar em uma nova camada a fins de organização e clareza do código.

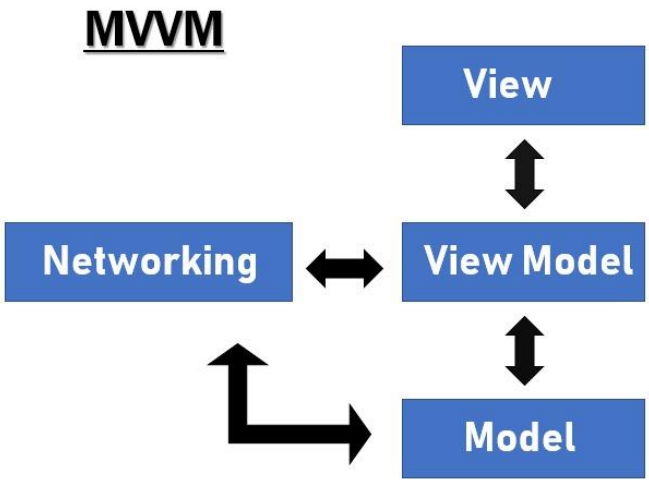

Figura 3.8 - comunicação MVVM com *networking*

Camada *view*

A camada *view* contém apenas código de apresentação, ou seja, somente o que o usuário vê de fato. Essa camada não possui nenhuma regra de negócio, ela é apenas usada como interface como o usuário. Toda a informação que deve ser mostrada, e, como deve ser mostrada, vem da camada *view model*. Todas as ações do usuário são delegadas na forma de eventos para a camada seguinte. Por exemplo, quando o usuário clica no botão salvar, um evento é gerado e enviado para a *view model* fazer a tomada de decisão. No iOS, essa camada é representada pelas *storyboards* e pelas *xibs*, que são arquivos de interface xml. Ainda, estão presentes nessas camadas todas as classes que herdam *UIViewController*. *UIViewControllers* são uma das principais classes do *framework* iOS. Elas são usadas para coordenar todos o ciclo de vida de uma tela e também toda a interação com o usuário.

Figura 3.9 - *delegates*

Camada *view model*

A camada *view model* possui todas as regras de negócio. A *view model* é responsável por receber todos os eventos enviados pela *view*, interpretá-los e fazer a devida tomada de decisão. Por exemplo, quando o usuário clica no botão salvar, é de sua responsabilidade decidir se o *form* pode ser salvo ou não, fazendo todas as validações necessárias e, se válido, fazer as requisições para a camada de *networking*. A camada *view model* orquestra todas as interações entre as outras camadas e trata devidamente exceções que possam ocorrer, como por exemplo, se o servidor estiver fora do ar, é de sua responsabilidade tratar e decidir se deve ser mostrado um alerta para o usuário, ou se deve ser simplesmente ignorado.

Da mesma forma que a *view* não possui regras de negócio, a *view model* não deve possuir elementos de interface. Por exemplo, se uma informação deve ser destacada para o usuário, a *view model* deve apenas informar a *view* para destacar a informação. O tipo de destaque, se vai ser uma cor diferente, ou uma fonte maior, é de responsabilidade da *view*. Desta forma, mantemos o mínimo necessário de acoplamento entre as duas camadas. Imagine a situação em que damos suporte para iPads. Neste caso, podemos ter duas *views*, uma para telas pequenas (iPhone) e outra para telas grandes (iPad). O layout pode ser totalmente diferente, visto que temos muito mais espaço disponível em iPads. Para a *view model*, isso não faz diferença, ela apenas informa que determinada informação deve ser destacada, e cada *view* destaca a informação da forma que melhor lhe convém. Aqui, cabe a ressalva que o MVVM clássico defende que as *views* devem ser totalmente *dummies*, ou seja, sem nenhum tipo de inteligência. Contudo, na prática, *views* devemter tomada de decisão

quanto a como uma informação deve ser apresentada graficamente (cores, fontes, tamanhos e formato).

A principal motivação para a migração do MVC para o MVVM foi o problema conhecido como "massive *view controllers*", onde as *Views Controllers* possuem muito código, pois devem lidar com toda a lógica de apresentação, ciclo de vida, interações com o usuário e ainda, com as regras de negócio. O MVVM foi originalmente desenvolvido para ser a arquitetura da tecnologia WPF da Microsoft e posteriormente adotado por outras tecnologias. O que vemos são muitos programadores seguindo à risca uma arquitetura que originalmente foi projetada para outra tecnologia. Nesse caso, teríamos um novo problema: estaríamos trocando as "*massive view controllers*" por "*massive view models*".

Camada *model*

Das quatro, a camada *model* é a mais simples: nada mais é que as classes que encapsulam os dados. As *models* podem ainda conter funções de cálculo e validações que não dependam de outras entidades, as chamadas *computed properties*. Por exemplo, uma *model* que represente a entidade "Funcionário", sendo que, dentre outras, temos as propriedades "Horas trabalhadas" e "Valor hora trabalhada". O cálculo do "Valor total a ser recebido" pode ser uma função dentro da própria classe. *View models* usam *models* para se comunicar com a camada de *Networking* e vice-versa.

```
struct Funcionario {
    var nome                : String
    var setor               : Int
    var horasTrabalhadas    : Float
    var valorHora           : Float

    func totalAReceber() -> Float {
        return (horasTrabalhadas * valorHora)
    }
}
```

Figura 3.10 - exemplo de um model

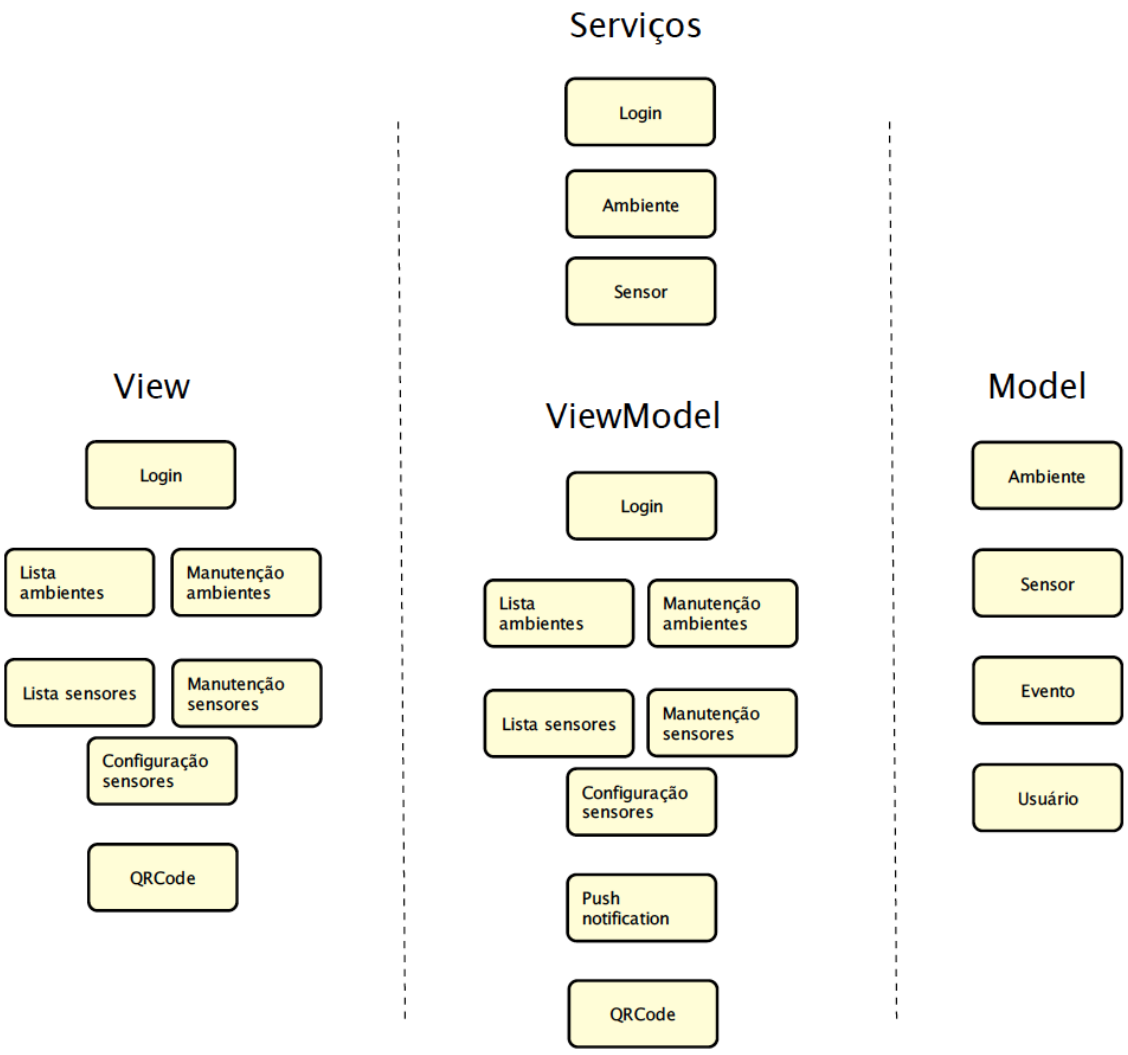

Figura 3.11 – camadas do app

3.4 Servidor

BaaS - Backend as a Service. BaaS é um conjunto de serviços, APIs e ferramentas providas por um provedor, neste caso a Google. A idéia por trás do conceito BaaS é que o desenvolvedor se preocupe apenas com o que de fato tem valor para o usuário final. Por "valor para o usuário final", entende-se as funcionalidades disponíveis para ele. Não há dúvidas de que segurança dos dados é um fator muito importante para o usuário, pois ninguém quer seus dados expostos inadvertidamente. Todavia, no contexto de engenharia de software, segurança de dados não tem valor para o usuário a não ser que a solução seja especificamente para isso, como por exemplo, uma solução de criptografia de dados. Sendo assim, uma solução BaaS deve prover tudo que é necessário para que

uma aplicação rode na nuvem, de hardware (mesmo que virtual), sistema de armazenamento persistente de dados, sistemas de segurança, à funcionalidades secundárias como dados analíticos e mensageiros (*push notifications*, SMS dentre outros).

Originalmente, seria usado o Firebase da Google como uma plataforma BaaS. Apesar de todas as vantagens já apresentadas, o Firebase demonstrou um grau muito grande de acoplamento das aplicações com a estrutura de dados usada no backend. Tanto o *app* quanto o *hardware* necessitam ter conhecimento sobre a estrutura de dados. Qualquer mudança exigiria alterações em todas as pontas. Por exemplo, o simples fato de remover um atributo de uma tabela para outra, mantendo as mesmas informações, implicaria em mudanças em todas as aplicações que consomem o serviço.

Por esse motivo, migrou-se para uma solução IaaS - *Infrastructure as a Service* - da Amazon. Em uma solução IaaS, o provedor fornece apenas os recursos computacionais - um servidor virtual com apenas o Sistema Operacional (SO) de nossa escolha dentre as opções disponíveis. Além do SO, podemos escolher outros aspectos como a capacidade de processamento, a memória RAM disponível, e a capacidade de armazenamento. Temos a nossa disposição um painel de controle online com vários recursos que fazem o IaaS muito mais vantajoso do que ter uma máquina física real provendo serviços. Através desse painel, por exemplo, pode-se migrar o servidor para um com mais recursos computacionais com um simples clique sem fazer que o serviço fique indisponível. Outra grande vantagem é a alta disponibilidade provida pela Amazon com servidores espalhados pelo mundo inteiro.O servidor pode ser configurado através de acesso remoto, como se estivessemos acessando uma máquina real. O servidor escolhido foi o Windows Server 2012, processador Intel Xeon E5-2670 2.6 GHz, 4 GB de memória RAM e 40 GB de armazenamento.

Utilizou-se o framework WCF da Microsoft para criar serviços *RESTful* rodando sobre o protocolo HTTP. WCF é um *framework* que permite a criação de serviços sobre o paradigma SOA (*Service-oriented architecture*). Um dos princípios do SOA é a abordagem *request/reply*. Um serviço pode requisitar e/ou responder para outros serviços, ou apenas responder para aplicações. Um serviço *RESTful* é um *Web Service* com arquitetura REST. Serviços REST sobre HTTP funcionam sobre quatro princípios:

- ***Uniform Resource Identifiers* (URI)**: é o identificador único do recurso. É através do URI que o serviço será identificado e exposto para ser consumido. Um recurso nada mais é o serviço em si que será acessado através de uma mensagem.

- ***Uniform interface***: define as ações que podem ser feitas em um recurso. Essas ações são as definidas no protocolo HTTP: PUT, GET, POST e DELETE. PUT é usado para operações de atualização, enquanto POST para operações de criação. Outra diferença é que PUT deve ser idempotente, já POST não. GET é usado para consultas e DELETE para remover. Nada impede que, por exemplo, façamos tudo usando GET. Essas definições são apenas uma convenção, porém é importante segui-lás para que qualquer aplicação saiba como o serviço vai se comportar.

- ***Self-descriptive messages***: todo o serviço deve fornecer as informações necessárias para que outro cliente o consuma. O WCF implementa *Self-descriptive messages* através da interface WSDL. WSDL é um arquivo de configuração no formato XML que provê toda a configuração do serviço. Através do WSDL, um cliente pode tanto interpretar corretamente as mensagens de resposta do servidor quanto enviar os parâmetros de entrada no formato esperado. Ademais, contém o protocolo de comunicação que deve ser utilizado.***Stateless***: toda mensagem deve conter as informações necessárias para que possa ser interpretada. O servidor não deve manter nenhum estado das mensagens prévias. A principal vantagem de servidores *stateless* é a escalabilidade, pois não ter que armazenar o estado das transações demanda menos recursos, além de permitir que sejam rapidamente liberados.

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsa10="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://tempuri.org/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" name="APPService"
targetNamespace="http://tempuri.org/">
 <wsdl:types>
  <xsd:schema targetNamespace="http://tempuri.org/Imports">
   <xsd:import schemaLocation="http://54.233.97.105/GasLeak/APPService.svc?xsd=xsd0" namespace="http://tempuri.org/"/>
   <xsd:import schemaLocation="http://54.233.97.105/GasLeak/APPService.svc?xsd=xsd1" namespace="http://schemas.microsoft.com/2003/10/Serialization/"/>
   <xsd:import schemaLocation="http://54.233.97.105/GasLeak/APPService.svc?xsd=xsd2" namespace="http://schemas.datacontract.org/2004/07/GasLeak.Entity"/>
   <xsd:import schemaLocation="http://54.233.97.105/GasLeak/APPService.svc?xsd=xsd3" namespace="http://schemas.datacontract.org/2004/07/ApproachFramework"/>
  </xsd:schema>
 </wsdl:types>
 <wsdl:message name="IAPPService_login_InputMessage">
  <wsdl:part name="parameters" element="tns:login"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_login_OutputMessage">
  <wsdl:part name="parameters" element="tns:loginResponse"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_deleteDevice_InputMessage">
  <wsdl:part name="parameters" element="tns:deleteDevice"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_deleteDevice_OutputMessage">
  <wsdl:part name="parameters" element="tns:deleteDeviceResponse"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_deleteRoom_InputMessage">
  <wsdl:part name="parameters" element="tns:deleteRoom"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_deleteRoom_OutputMessage">
  <wsdl:part name="parameters" element="tns:deleteRoomResponse"/>
 </wsdl:message>
 <wsdl:message name="IAPPService_getRooms_InputMessage">
  <wsdl:part name="parameters" element="tns:getRooms"/>
```


Figura 3.11 - arquivo WSDL

Implementou-se o WCF com a linguagem C# versão 7.0 e a IDE Visual Studio 2017. C# é uma linguagem de programação amplamente utilizada em servidores e em sistemas desktop, além de ser umas das principais linguagens da plataforma .NET da Microsoft. Devido à sua simplicidade, a arquitetura escolhida foi o *N-Tier Architecture* (uma vez que não existe muita complexidade na aplicação servidor, não se justifica uma arquitetura mais sofisticada como a usada no *app*). A aplicação servidor possui apenas duas camadas: uma com as regras de negócio e persistência de dados e outra muito similar a *model* do MVVM com as entidades. Por fim, o WCF exige uma interface com a descrição dos serviços que será usada para gerar o arquivo WSDL.

```
[OperationContract]
[WebInvoke(Method = "POST",
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        BodyStyle = WebMessageBodyStyle.Wrapped,
        UriTemplate = "insertDevice")]
1 reference
Response insertDevice(Device device);

[OperationContract]
[WebInvoke(Method = "POST",
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        BodyStyle = WebMessageBodyStyle.Wrapped,
        UriTemplate = "insertRoom")]
1 reference
ResponseString insertRoom(Room room);
```

Figura 3.12 – exemplo de *interface*

A aplicação servidor foi hospedada na solução IIS da Microsoft. O IIS é um *Web Server* que monitora requisições através das portas que foram registradas e as encaminha para a aplicação correspondente processar.

3.5 Banco de dados

Implementou-se um banco de dados relacional com o SGBD Firebird versão 3.0.2. A principal motivação para o uso do Firebird deve-se ao fato dele ser uma solução extremamente leve (processador de um *core* de 2 GHz, 1 GB de memória RAM e 400 MB de espaço disponível em contraste com o MySQL, por exemplo, que necessita processador de 2 *cores*, 2 GB de memória RAM e 700 MB de espaço disponível). Mesmo assim, apresenta um bom desempenho e tem todos os recursos que outros SGBDs mais famosos oferecem, tais como: *stored procedures*, *triggers*, conformidade com ACID, *backups* online, *generators* e *transactions.* Possui diversos *drivers* de acesso (ODBC, JDBC, ADO, PHP dentre outros) possibilitando que seja usado com praticamente qualquer linguagem de programação. Ainda, os dados ficam armazenados em um único arquivo, tornando-o muito conveniente e prático.

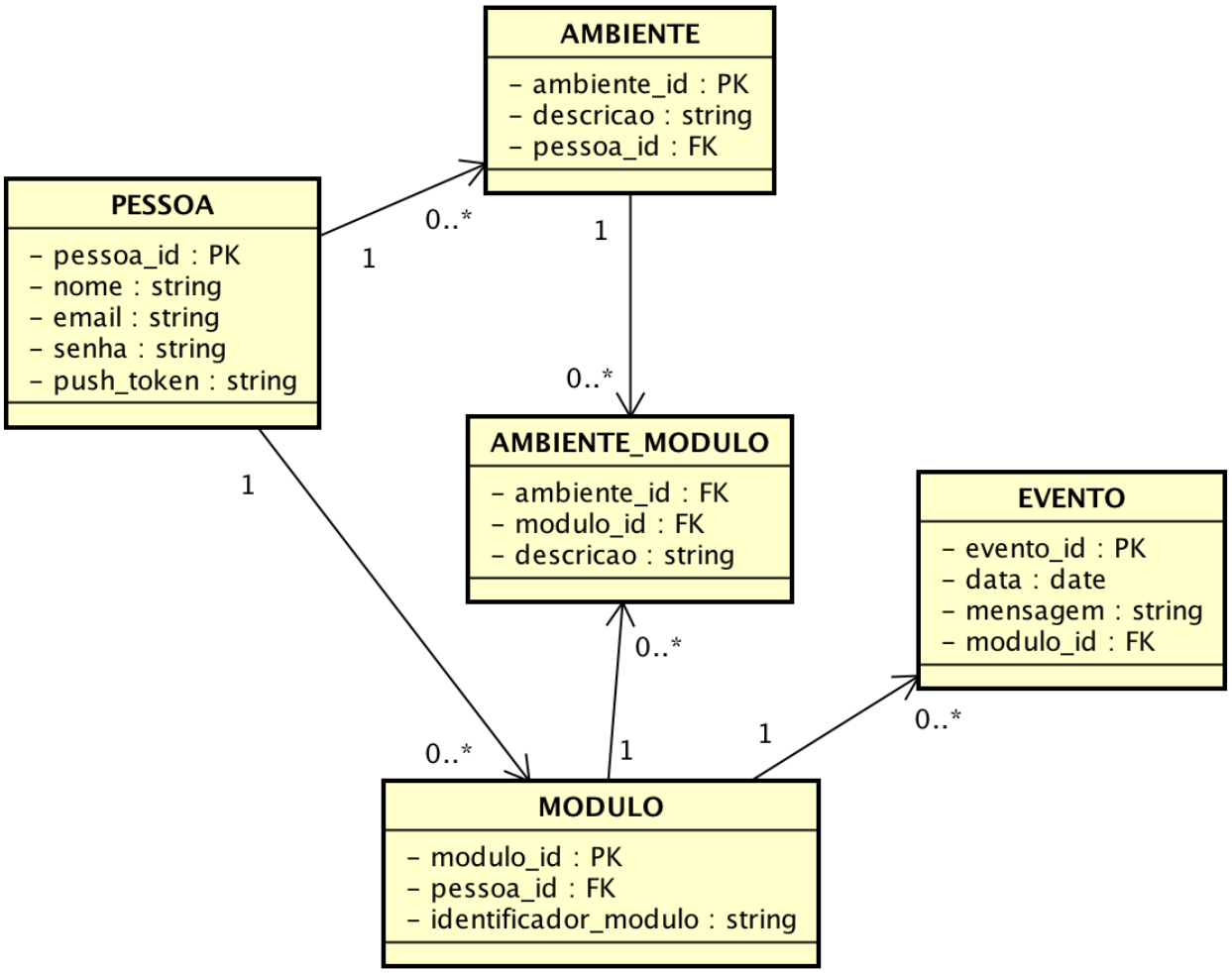

Figura 3.12 - diagrama relacional do banco de dados

3.6 Hardware monitor

Para o monitoramento, empregou-se o microcontrolador ESP-07 e mais dois tipos de módulos: um com o sensor de gás e outro com o sensor infravermelho.

O sensor de gás utilizado foi o MQ-05, já descrito em seções anteriores. O sensor utilizado já conta com um circuito amplificador de sinal e um potenciômetro para ajuste do tempo de resposta (quanto menor o tempo de resposta, menor é a precisão do sensor). O sensor possui uma saída analógica que varia de zero a 5V conforme a quantidade de gás detectada. Visto que a tensão de trabalho do ESP-07 é 3.3V e a do MQ-05 é 5V, precisou-se utilizar um divisor de tensão como

driver para fazer a conversão. A saída do sensor foi ligada diretamente na entrada analógica ADC do microcontrolador.

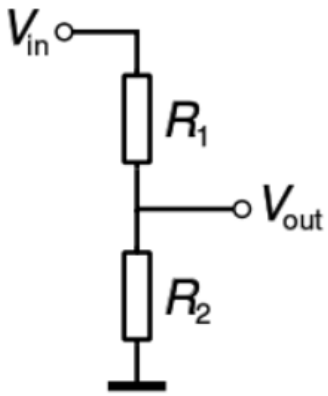

Figura 3.13 – divisor de tensão

Para detectar o uso não usual tentou-se, primeiramente, utilizar um sensor de som - um microfone. Acoplou-se o microfone diretamente na mangueira de gás, porém o microfone não foi capaz de captar nenhum ruído. Utilizou-se então um sensor infravermelho para detectar quando a chama do fogão está acesa. A grande desvantagem desta abordagem em relação à anterior é que o sensor infravermelho precisa ser apontado diretamente para a chama, criando um desafio achar um local para ser instalado. Muito similar ao módulo de vazamento de gás, este módulo possui um amplificador operacional para amplificar o sinal do sensor infravermelho. O módulo foi ligado diretamente na entrada digital número 03 do microcontrolador.

A programação do ESP-07 foi feita utilizando a linguagem de programação C e a IDE Arduino 1.6. O *upload* do programa para o microcontrolador foi feito com o conversor CP2102.

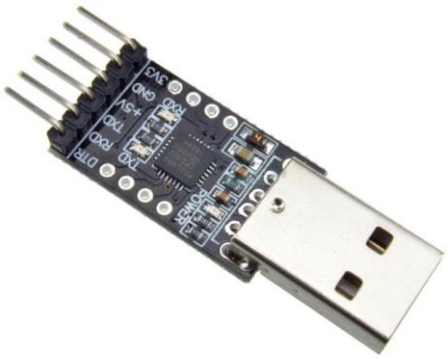

Figura 3.14 - conversor CP2102

O CP2102 é um conversor USB para UART. Com ele podemos usar qualquer porta USB do computador, uma vez que a interface de comunicação do microcontrolador é UART. A própria IDE Arduino pode ser usada para o *upload*.

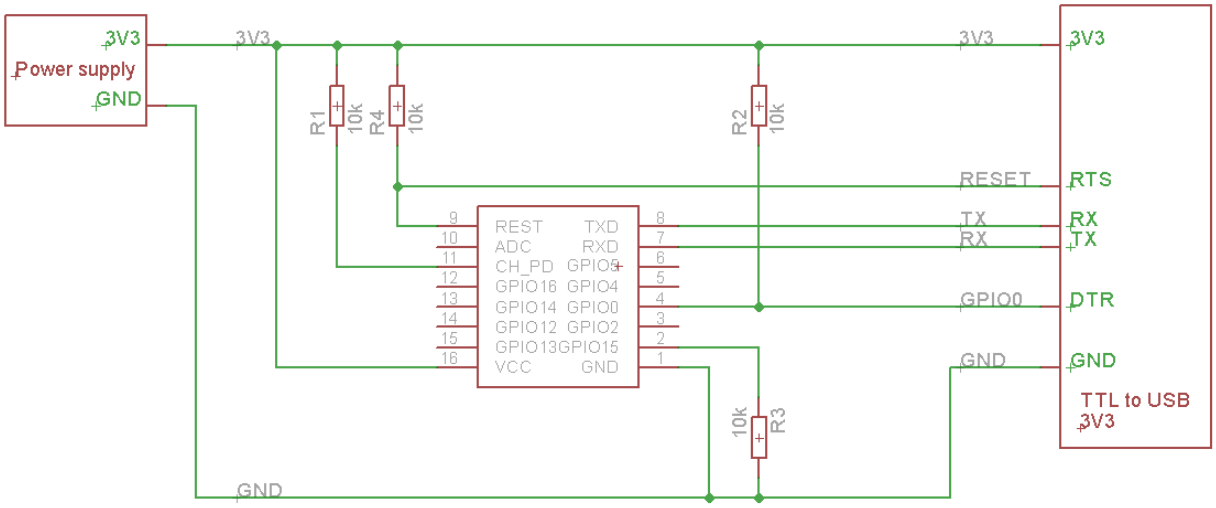

Figura 3.15 - esquemático para *upload*

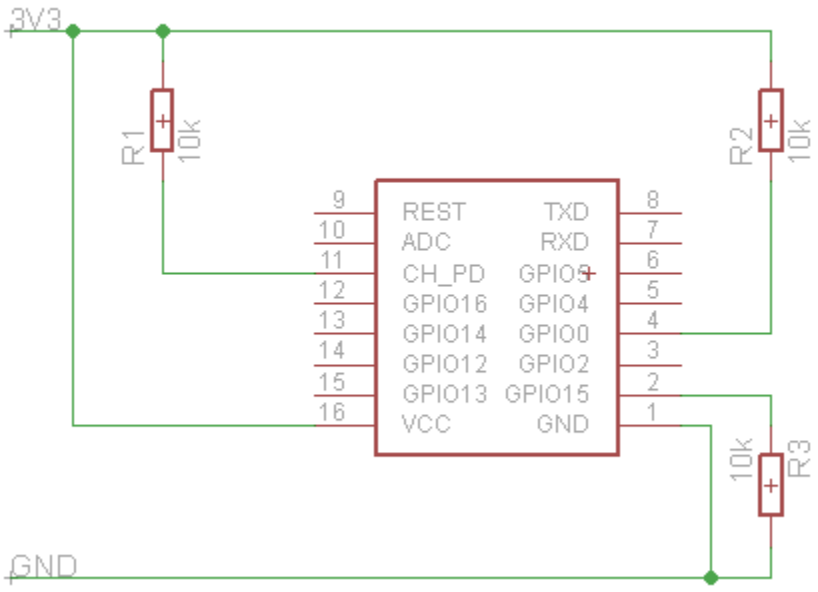

Figura 3.16 - esquemático para modo run

4 RESULTADOS E ANÁLISE

O desfecho deste trabalho é um sistema de monitoramento de vazamento de gás de baixo custo - menos de R$ 30,00. O resultado da implementação do recurso principal, monitoramento de vazamentos de gases, mostrou um resultado excelente: de fácil instalação (pode ser instalado em qualquer ponto do ambiente) e com resposta a vazamentos de gases muito boa.

Por outro lado, o recurso de uso não usual demonstrou vulnerabilidades devido à dificuldade de se achar um bom local para instalação. A necessidade do sensor infravermelho ter que apontar diretamente para a chama pode ser um problema dependendo da disposição do ambiente a ser monitorado.

Outro ponto a ser analisado é o custo com o servidor. Se a solução vier a tornar-se um produto comercial, precisa-se analisar o modelo de negócio. O servidor tem um custo mensal que poderia ser diluído no preço do produto fazendo-se uma projeção de quantos deveriam ser vendidos mensalmente. No entanto, a análise do modelo de negócio ideal foge ao escopo do trabalho.

Quanto à parte de *design* e *user experience* do *app*, considera-se que o resultado não foi bom, apesar de funcional. Não se explorou todos os recursos disponíveis como cores, formatos eanimações. O principal motivo deve-se ao fato de que, apesar de parecer simples, isto é uma tarefa complexa que normalmente é feita por um profissional especializado.

Conclui-se dessa análise, apesar dos pontos negativos levantados, que de fato criou-se um sistema de monitoramento de vazamentos de amplo acesso - fácil instalação e configuração a baixo custo.

5 CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho teve como objetivo o desenvolvimento um sistema de monitoramento de vazamentos de gases e uso não usual em tempo real. Esperava-se que esse sistema fosse de amplo acesso - simples instalação e configuração a baixo custo.

Considerando-se os resultados, pode-se dizer que a solução foi implementada com sucesso.

5.1 Trabalhos futuros

Sugestões de trabalhos futuros são:

Quanto ao *app*:

- Melhoria de usabilidade e *design* do *app*
- Recurso para remover
- Forma de remover o módulo de todos os usuários
- Senha para adicionar o módulo
- Forma de compartilhar o módulo com outros usuários

Quanto ao hardware:

- Módulo com os dois sensores
- Aprimoramento do sensor de uso não usual

6 REFERÊNCIAS

[1] Gás LP. **O Gás do Brasil.** Disponível em: <http://www.sindigas.org.br/uploads/book_sindigas_2013_site.pdf. Acesso em: 23 de maio de 2017.

[2] **Gás natural em diversas aplicações.** Disponível em: <http://www.brasil.gov.br/infraestrutura/2011/12/gas-natural-tem-diversas-aplicacoes. Acesso em: 23 de maio de 2017.

[3] **Gás liquefeito de petróleo.** Disponível em: <www.supragas.com.br/glp.php. Acesso em:23 de maio de 2017.

[4] **Quimicamente, o que é o Gás Natural?.** Disponível em: <www.goiasgas.com.br/quimicamente,_o_que_e_o_gas_natural_.html>. Acesso em: 20 de maio de 2017.

[5] **Mercado de GLP.** Disponível em: <https://www.liquigas.com.br>. Acesso em: 20 de maio de 2017.

[6] NASHIRO, Arnaldo. **Tendências do mercado GLP no mundo e oportunidades para o Brasil.** Disponível em: <http://www.gasescombustiveis.com.br/encontroglp/Palestras_2016/Tendencias_do_mercado_GLP_12_EVENTO_GLP_ULTRA.pdf>. Acesso em: 25 de maio de 2017.

[7] Gás natural: **potencialidades de utilização no Brasil.** Disponível em: <https://periodicos.ufsm.br/reget/article/viewFile/7896/pdf>. Acesso em: 13 de maio de 2017.

[8] **Anuário Estatístico Brasileiro do Petróleo, Gás Natural e Biocombustíveis 2016.** Disponível em: <http://www.anp.gov.br/wwwanp/publicacoes/anuario-estatistico/2441-anuario-estatistico-2016#Se%C3%A7%C3%A3o%201%20%C2%BF%20Panorama%20Internacional>. Acesso em: 23 de maio de 2017.

[9] **O Firebase ajuda você a criar apps melhores e desenvolver seus negócios.** Disponível em: <https://firebase.google.com>. Acesso em: 20 de maio de 2017.[10] **Turn your company into an apps company.** Disponível em: <https://www.heroku.com/>. Acesso em: 20 de maio de 2017.

[11] **Firebase é a nova aposta da Google em uma plataforma móvel unificada.** Disponível em: <https://www.tudocelular.com/android/noticias/n72091/google-lanca-nova-versao-do-firebase.html>. Acesso em: 20 de maio de 2017.

[12] **Princípios de funcionamento de um detector de gás.** Disponível em: <http://acessopercon.com.br/percon/sensores-de-gas-principios-e-tecnologias/>. Acesso em: 20 de maio de 2017.

[13] **Gas leakage module using the MQ-5 gas sensor.** Disponível em: <https://wisense.wordpress.com/2015/07/06/gas-leakage-module-using-the-mq-5-gas-sensor/>. Acesso em: 20 de maio de 2017.

[14] **Site oficial da documentação da Espressif.** Disponível em: <https://espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14>. Acesso em: 12 de junho de 2017.

[15] **Site oficial da Smappee**. Disponível em: http://www.smappee.com
Acesso em 24 de junho de 2017.

[16] **Trabalho de conclusão do Jefferson Silva Santos**. Disponível em: http://www.repositorio.uniceub.br/bitstream/235/3697/2/Monografia%20JEFFERSON%20SANTOS%202-2012.pdf
Acesso em 24 de junho de 2017.

[17] **System-on-chips**
https://www.arm.com/develop/custom-system-on-chips

[18] **Maker Culture**
https://en.wikipedia.org/wiki/Maker_culture

[19] **Site oficial do Arduino**

https://www.arduino.cc/

[20] **Site oficial do Xcode**

https://developer.apple.com/xcode/

[21] **Datasheet ESP-07**

https://www.tinyosshop.com/datasheet/ESP-07S_User_Manual.pdf

[22] **Livro sobre sensores**

Sensores Industriais - Fundamentos e Aplicações (Cód: 185249)

Thomazini,Daniel / Albuquerque,Pedro U. Braga de

[23] **Datasheet da familia de sensores de gás MQ**
https://www.plexishop.it/pdf/Sensori%20Gas%20serie%20MQ%20XX.pdf

[24] **Web services**
https://en.wikipedia.org/wiki/Web_service

[25] **XML Web services**
https://www.w3schools.com/xml/xml_services.asp

[26] **Como funcionam os Web services**
http://www.soawebservices.com.br/como-funciona.aspx

[27] **Web Services Explained**
https://www.service-architecture.com/articles/web-services/web_services_explained.html

[28] **Entendendo o MVVM**
https://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411

[29] **Fundamentals of database systems**
Elmasri, Ramez. Fundamentals of database systems. 6th ed. Reading : Addison-Wesley, c2011. 1172 p. : il.

6.1. Referências das imagens

Figura 1.1
http://noticias.band.uol.com.br/cidades/noticia/100000797300/sp-explosao-por-vazamento-de-gas-deixa-um-morto-e-tres-feridos.html. Acesso em 24 de junho de 2017.

Figura 1.2
http://www.acritica.com/channels/manaus/news/vazamento-de-gas-causa-explosao-e-deixa-seis-pessoas-feridas-na-compensa Acesso em 24 de junho de 2017.

Figura 1.3
http://www.opovo.com.br/noticias/fortaleza/2017/04/apartamento-explode-apos-vazamento-de-gas-moradora-sofre-queimaduras.html Acesso em 24 de junho de 2017.

Figura 1.4
http://www.otempo.com.br/vazamento-de-g%C3%A1s-causa-explos%C3%A3o-pr%C3%A9dio-desaba-e-cinco-morrem-1.1213443 Acesso em 24 de junho de 2017.

Figura 1.5
http://g1.globo.com/rio-de-janeiro/noticia/2015/08/predios-da-av-rio-branco-rio-sao-evacuados-apos-vazamento-de-gas.html Acesso em 24 de junho de 2017.

Figura 2.1
http://www.arduinoecia.com.br/2017/01/upgrade-firmware-esp8266-esp-07.html Acesso em 24 de junho de 2017.

Figura 2.2
https://esp8266.ru/wp-content/images/esp8266-modules/ESP8266_ESP-06.jpg Acesso em 24 de junho de 2017.

Figura 2.4
http://dwmzone.com/546-thickbox_default/mq-5-city-gas-detector-sensor-module.jpg Acesso em 12 de dezembro de 2017.

Figura 2.5
http://i.ebayimg.com/00/s/ODAxWDgwMQ==/z/pDEAAOSw9N1VsZhA/$_35.JPG Acesso em 12 de dezembro de 2017.

Figura 2.6
http://mlb-d2-p.mlstatic.com/sensor-digital-detector-de-fogo-chamas-arduino-pic-836011-MLB20470218657_102015-F.jpg?square=false Acesso em 12 de dezembro de 2017.

Sistema de monitoramento de vazamento de gás liquefeito de petróleo (GLP) e gás natural (GN) em tempo real

Luis Henrique Mendonça Grassi

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
lhmgrassi@gmail.com

Abstract*. We often come across news of serious accidents caused by gas leaks, many of them causing the collapse of whole buildings due to explosions. Introduced this recurrent problem, we will propose a monitoring system of gas GLP and GN leak in real time. The system will aim to assist users by alerting them when leakage or unusual use is detected (heating system switched on for a longer time than intended - useful for detecting forgotten flames burning, for example). We will be developing a hardware to monitor leaks with cloud communication and also an application, initially for Android platform, in which the user can consult and configure the system.*

Resumo*. Não raramente nos deparamos com notícias sobre graves acidentes provocados por vazamentos de gás. Muito deles provocando o colapso de prédios inteiros devido a explosões. Dado esse problema recorrente, apresentaremos uma proposta de sistema para monitoramento de vazamento de gás LP e gás natural em tempo real. O sistema visará auxiliar os usuários alertando-os quando for detectado vazamentos ou uso não usual (sistema de aquecimento ligado por um tempo maior que o estabelecido – útil para detectar esquecimentos de chamas acesas, por exemplo). A ideia inicial será desenvolver um hardware para monitoramento de vazamentos com comunicação em nuvem. Ainda, desenvolveremos um aplicativo, inicialmente para plataforma Android, no qual o usuário poderá consultar e configurar o sistema*

1. Introdução

O gás LP, mais conhecido como gás de cozinha, tem um papel importante na vida de cada um de nós. No Brasil, ele representa 3,2% da nossa matriz energética [1]. São cerca de 195 milhões de brasileiros que usam direta ou indiretamente essa fonte de energia. O famoso botijão está dentro de 95% dos domicílios, com cerca de 33 milhões de unidades vendidas mensalmente em todo o país [1].

De fácil armazenagem e transporte, tem uma versatilidade que vai muito além dos fogões dos nossos lares ou dos isqueiros em nossos bolsos: é usado em equipamentos de solda e corte; no ramo industrial como combustível para empilhadeiras; em lavanderias como fonte de calor para secagem de roupas; no ramo agrícola para secagem de grãos, e também para incineração de lixo.

Outro gás não tão popular, porém muito utilizado, é o gás natural. Este representa 7,2% [1] da matriz energética brasileira. Seus usos vão dos mais simples - como a utilização em fogões domésticos e combustível veicular, substituindo o etanol ou a gasolina - até usos mais complexos, nas indústrias químicas, como na elaboração de compostos como o metanol e a ureia. Por ser uma fonte mais limpa que os derivados do petróleo, como o gás LP, tem recebido grandes investimentos no Brasil para expansão de seu uso.

Apesar de todos os investimentos nos último anos, o Brasil aparece apenas na 30ª posição entre os maiores consumidores de GN. Já quando falamos de consumo de GLP, aparecemos logo após da Tailândia, na 10ª posição [5].

Por estar presente no nosso dia-a-dia, muita vezes acabamos negligenciando cuidados básicos de segurança ao utilizar essas fontes de energia. Uma simples consulta no Google, por exemplo, nos mostra o quão perigoso pode ser seu uso se não seguirmos as normas de segurança. Apesar de não serem tóxicos, ambos podem causar asfixia levando à morte. No caso do gás LP, por ser mais denso do que ar, esse quadro é agravado. Já o gás natural se dissipa facilmente por ter uma densidade mais baixa do que o ar deixando-o menos propício a explosões, [4] outro grande risco.

Abaixo alguma manchetes e fotos demonstrando a gravidade desses acidentes.

Figura 1 - Explosão por vazamento de gás deixa 1 morto

Fonte: Band News

Figura 2 - Vazamento de gás causa explosão e deixa seis pessoas feridas

Fonte: A Crítica - Jornal Digital

Figura 3 - Apartamento explode após vazamento de gás; moradora sofre queimaduras em todo o corpo

Fonte: O Povo

Figura 4 - Vazamento de gás causa explosão, prédio desaba e cinco morrem

Fonte: O Tempo

Vazamento de gás interrompe VLT e esvazia dois prédios no centro do Rio

Fonte: O Globo

É fácil perceber o tamanho do problema que estamos lidando. Diariamente, bilhões de pessoas dependem dessa fonte de energia, direta ou indiretamente. E, como citado, o problema não se restringe às nossas cozinhas: vazamentos de gás podem ocorrer nos mais variados locais colocando nossas vidas em risco sem nem mesmo percebermos.

Porém, nossa proposta é focada no uso residencial, todavia não exclusiva, por entendermos que grande quantidade dos acidentes domésticos poderiam ser facilmente evitados com providências simples e baratas. Quantas vezes nos preocupamos com a validade da mangueira ou da válvula do botijão? Ou ainda, com a distância mínima que o fogão deve estar de tomadas e interruptores?

Propomos um sistema de monitoramento de vazamentos e uso não usual em tempo real. Para ser de amplo acesso, deverá ser de fácil instalação e configuração, não deverá custar mais que algumas dezenas de reais e não poderá ser intrusivo: todo o processo de monitoramento deverá ser passivo, não podendo alterar de forma alguma o funcionamento de outros componentes como válvulas, reguladores de pressão ou mangueiras. Isso exigiria certificação de órgãos reguladores o que encareceria o processo, provavelmente inviabilizando-o.

1.1. Sistema similares

A Smappee [15], uma empresa sediada na Bélgica, possui uma solução para monitoramento de consumo de eletricidade e gás, sendo que umas das *features* é a detecção de vazamentos. Porém, além de não ser o foco do produto, a solução custa mais de mil reais. Achamos também um trabalho [16] datado de 2012, onde o autor apresenta uma idéia similar - o usuário seria avisado do vazamento por uma mensagem SMS - o que nos dias de hoje não seria uma boa opção visto os custos relacionados (plano de dados, compra de chip) e a baixa praticidade (todo mês o usuário tem que lembrar de recarregar o plano de dados). Por fim, achamos dezenas de soluções de monitoramento tradicionais e sem nenhuma inovação tecnológica: sistemas que não possuem nenhum tipo de interação com o usuário, apenas emitem um sinal sonoro quando detectam um vazamento.

2. Visão geral do sistema

Visão geral do sistema

O sistema pode ser dividido em três partes: hardware - responsável pelo monitoramento contínuo do ambiente: é o gatilho de eventos (vazamento ou uso não usual), serviço PaaS na nuvem - responsável por centralizar a comunicação entre o hardware e o aplicativo móvel (app). E o app, usado com IHM. Com o intuito de facilitar a visão, faremos uma abordagem *top down*, começando pelo app.

2.1. Aplicativo móvel

O app, inicialmente desenvolvido para a plataforma Android utilizando a linguagem Java (posteriormente para iOS, porém fora do escopo deste trabalho), será utilizado pelo usuário tanto para configuração quanto para monitoramento do sistema. O usuário poderá cadastrar sensores (entende-se por sensor nesse contexto todo o hardware, que será detalhado posteriormente) e colocar descrições para cada sensor com o intuito de facilitar a identificação. Por exemplo, poderia

ser cadastrado o sensor "Fogão da casa - mãe" e o sensor "Fogão minha casa". A descrição tem semântica apenas para o usuário. O app apenas recebe e solicita informações sobre os sensores nele cadastrados. Logo, o usuário pode monitorar ambientes distintos, como a casa dos pais e a sua própria casa no mesmo app. Todo esse controle é feito através da conta do usuário.

Cada usuário terá uma conta que poderá ser acessada mediante uma senha por ele configurada no primeiro acesso. Cada sensor terá um identificador alfanumérico único anexado a ele que deverá ser informado pelo usuário no momento do cadastro. Isso pode ser feito, por exemplo, com um QR Code colocado no sensor. No momento do cadastro, o app apenas escaneia o QR Code e já faz a associação. O usuário poderá, ainda, excluir e editar os sensores previamente cadastrados. Um mesmo sensor poderá ser monitorado por usuários distintos. A descrição do sensor estará relacionada a tupla (idUsuário, idSensor) - identificador único do usuário e identificador único do sensor, respectivamente. Isso possibilita, por exemplo, que o sensor "Fogão da casa - mãe" cadastrado em um usuário, seja também cadastrado em outro usuário como "Fogão da casa - tia". Ambos os usuários irão consultar o mesmo sensor, o que muda é apenas a descrição. Por fim, o usuário poderá consultar o *status* dos sensores a qualquer momento. Não obstante, o app poderá receber *pushes notifications* com alertas quando for detectado vazamentos - "Atenção! Vazamento detectado: Fogão da casa - mãe".

2.2 Serviço na nuvem

Como seria inviável criar conexões p2p entre cada sensor e o smartphone, teremos um serviço PaaS (plataform as a service) para sincronizar as informações. Isso nos permitirá também ter persistência de dados. Existem, atualmente, vários serviços PaaS: Amazon AWS, Heroku e Google Firebase, Umbler, e Microsoft Azure são exemplos. Apesar de serem menos flexíveis do que um servidor próprio, o uso desse serviços nos possibilita focarmos no produto. Não precisamos perder tempo configurando sistemas operacionais, servidores web, banco de dados e afins; plataformas PaaS seguem o conceito de *Plug & Play* e sem contar que são economicamente mais viáveis. A Google vem fazendo fortes investimentos nessa área e sua plataforma Firebase é focada em aplicações móveis. Ainda, tem um plano gratuito que atende muito bem nossa proposta. Por esses motivos e também pelo fato de ter uma comunidade imensa e ativa, utilizaremos o Firebase.

2.3 Hardware

O sensor, como previamente chamado em seções anteriores, é na verdade um conjunto de módulos e não apenas um dispositivo que responde a estímulos do ambiente. Para não causar confusão, denominaremos o conjunto todo como *endpoint.* O *endpoint* é responsável pelo monitoramento do ambiente no qual estiver inserido e pela comunicação com o serviço em nuvem. Será composto basicamente de um módulo de processamento, um módulo de memória permanente, um módulo de comunicação e um módulo sensor. Para que seja possível detectar vazamentos e/ou

uso não usual, precisaremos de dois tipos de sensores. No primeiro caso, precisaremos de um sensor capaz de detectar partículas de gás no ambiente. Dentre os mais comuns estão os eletro-catalíticos e os semicondutores. Ambos têm o mesmo princípio: quando o gás passa por uma superfície aquecida, ocorre a combustão alterando alguma propriedade do sensor. No primeiro caso, utiliza-se um termômetro. Quanto maior for a quantidade gás, maior será a temperatura. Já no caso dos semicondutores, o catalisador é feito de silício. A diferença de temperatura causa uma diferença na condutividade elétrica. Da mesma forma do anterior, quanto maior for a quantidade de gás, maior será a combustão ocasionando uma variação maior da resistência. O sensores semicondutores são ligeiramente mais baratos e podem detectar uma gama de gases maiores, apesar de não serem tão precisos - o que não é um problema, pois não queremos detectar a quantidade exata de gás presente, e sim apenas identificar o vazamento. No máximo daremos três níveis de vazamento: baixo, médio, alto.

Usaremos o sensor da família MQ, em específico o MQ-5, capaz de detectar gás natural e de cozinha. Essa família de sensores é amplamente utilizada na indústria por ser barato, ter uma curva de resposta rápida e longa durabilidade. Já para detectar uso anormal, precisamos, de alguma forma, detectar quando o fogão estiver aceso. Como um dos objetivos é não ser intrusivo, não podemos mudar as características do fogão, nem dos equipamentos como mangueiras e válvulas. Sendo assim, uma das alternativas seria usar um sensor de som para detectar ruídos na mangueira. O sensor ficaria acoplado na mangueira ou na válvula.

Essa abordagem nos traz alguns desafios, como filtrar os ruídos do ambiente e também a própria detecção do som - talvez seja inviável devida à baixa amplitude do ruído. Outra forma seria um sensor infravermelho de alta sensibilidade. Esses sensores são projetados para capturar o comprimento de onda emitido por uma chama alimentada por gás GN/GLP (dentre outros combustíveis). A grande desvantagem dessa forma é que o sensor tem que ficar apontado para a chama, o que pode ser complicado dependendo da posição onde se encontra o fogão e da quantidade de bocas. Com esse sensor seria inviável detectar o uso do forno. Tentaremos as duas abordagens, e se possível disponibilizaremos ambas as soluções para o usuário. Assim, dependendo do ambiente, pode-se fazer uso de uma, de outra, ou até mesmo de ambas.

Para capturar e enviar os dados dos sensores, utilizaremos um microcontrolador (SoC). Uma das primeiras alternativas que nos vêm à cabeça seria o uso de um Arduino. O Arduino é uma plataforma de prototipagem largamente difundida no mundo, e, com a grande quantidade de *shields* - módulos prontos para desempenhar determinada função no estilo *plug & play* - facilita muito o desenvolvimento, nos dando tempo para focar de fato no produto. Um desses shields são os módulos da família ESP8266EX da fabricante chinesa Espressif Systems utilizados para dar conectividade WI-FI ao Arduino.

Figura 6 - ESP-07 com antena de cerâmica

Porém, o que poucos percebem é que, muito mais que módulos WI-FI, são na verdade SoCs. Contam com uma MCU 32-bits RISC da fabricante norte-americana Tensilica. A memória RAM e a flash dependem do modelo. Possuem até 16 GPIO, sendo que alguns modelos têm IO analógicos, e, é claro, módulo WI-FI integrado. O tipo de antena também depende do sabor escolhido. Lançado em 2014, e com quase nenhuma documentação em inglês naquela época, o ESP8266EX vem ganhando cada vez mais espaço, principalmente devido ao seu baixo custo. Houve um grande esforço dos adeptos da cultura *maker culture* para gerar documentação, porém, hoje a própria fabricante disponibiliza datasheets em inglês. Possui SDK de desenvolvimento próprio, todavia a própria IDE do Arduino pode ser utilizada.

Os modelos variam do ESP-01 até o ESP-13, todos com características próprias variando de quantidade de IOs, tipo de encapsulamento da antena e da MCU, tamanho da flash a recursos como *watchdog timer* e *deep sleep*. Ainda, é possível notar que alguns modelos não são próprios para prototipagem, como o ESP-06 que possui os conectores na parte de baixo, no estilo BGA (Ball Grid Array).

Um dos modelos mais difundidos é o ESP-07. Versátil e como uma boa relação custo-benefício (custa em torno de R$ 30,00), ele possui antena de cerâmica e MCU de 32 bits com encapsulamento metálico, o que garante uma boa proteção contra interferências do ambiente. Esse módulo já vem com o protocolo TCP/IP integrado, possui 1MB de memória FLASH; o que parece ser suficiente, caso contrário é possível adicionar uma memória externa - e operar a uma frequência de 80 MHz. Possui duas IOs analógicas na qual podemos usar como trigger para determinar o nível de vazamento de gás. Outra opção seria o ESP-01. Custa apenas alguns reais a menos que o ESP-07, possui a mesma MCU e a mesma FLASH, mas não possui IOs analógicas (as IOs das MCU não estão ligadas a pinos do chip), a antena é do tipo PCB sem conector externo (é impressa diretamente na placa apenas) o que pode limitar o alcance e aumentar os pontos cegos, e não conta com nenhum encapsulamento metálico.

O ESP8266EX não é tão amigável quanto um Arduino, a começar pelo seu tamanho (5cm x 5cm) que dificulta o manuseio e não é *breadboard friendly*). Além disso, não possui controlador USB, apenas controlador serial UART, o que exige um adaptador para fazer o *upload* da aplicação visto que a grande maioria dos computadores atuais não possuem porta serial. Apesar dos problemas citados, manteremos-o como nossa primeira opção, pois seu baixo custo compensa. Fica em aberto ainda o modelo a ser utilizado

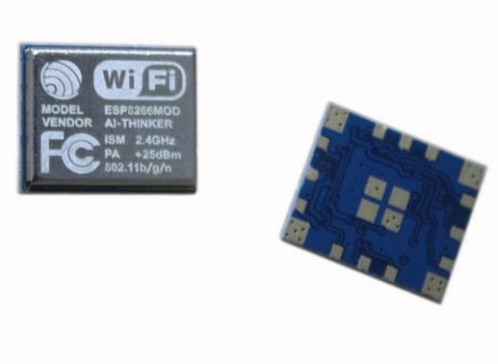

Figura 7 - ESP-06 com conectores BGA

3. Próximos passos

Nosso próximo passo será testar separadamente cada componente. Nesta fase, iremos fazer testes de estresse para avaliar o funcionando dos módulos em situações extremas. Aproveitaramos também para entender melhor as diferenças entre os SoCs apresentados e os serviços em nuvem podendo assim fazer a escolha mais adequada.

4. Cronograma

	Jul	Ago	Set	Out	Nov	Dez
Testar componentes isoladamente	x					
Desenvolvimento do hardware		x	x			
Configuração do ambiente em nuvem			x			
Desenvolvimento do aplicativo móvel				x		
Comunicação entre o hardware e o app				x	x	
Testes					x	
Conclusão e relatório						x

5. Conclusão

Em virtude dos riscos associados ao uso desses combustíveis, faz-se necessário que busquemos soluções modernas e acessíveis. Nosso app, juntamente com o hardware monitor, visa atender a essa necessidade alertando os usuários sobre ocorrências, objetivando assim reduzir a quantidade de acidentes ou, no mínimo, amenizar os danos.

Esperamos como resultado um sistema capaz de detectar automaticamente, sem falsos-positivos e, principalmente, sem falsos-negativos, vazamentos de forma rápida (algumas dezenas de segundos, no máximo) e uso não usual. O custo energético do sistema deve ser baixo, porém não é um ponto crítico, pois o sensores de vazamento funcionam por aquecimento, o que provavelmente exigirá que o sistema fique conectado diretamente na rede de energia. Ainda, a nível de resultado, esperamos que o sistema tenha uma durabilidade compatível com os sensores disponíveis hoje, (2~3 anos).

Quanto às métricas, teremos dificuldades para simular situações reais de vazamentos devido ao grande risco envolvido - não podemos induzir vazamentos de gás, pois não teremos como controlar todas as variáveis envolvidas. Faremos testes simples com isqueiros e refis de gás em campo aberto e também utilizando maquetes. Já quanto ao teste de uso não usual, testaremos na prática configurando o sistema para alertar após x minutos e deixando o fogão ligado por y. O sistema só poderá alertar quando a condição y > x for verdadeira. Sobre a durabilidade do sistema, nos basearemos na vida útil de cada componente (elencando a de menor durabilidade), visto que são componentes altamente difundidos, esperemos encontrar informações relevantes sobre isso.

6. Referências

[1] Gás LP. **O Gás do Brasil.** Disponível em: <http://www.sindigas.org.br/uploads/book_sindigas_2013_site.pdf. Acesso em: 23 de maio de 2017.

[2] **Gás natural em diversas aplicações.** Disponível em: <http://www.brasil.gov.br/infraestrutura/2011/12/gas-natural-tem-diversas-aplicacoes. Acesso em: 23 de maio de 2017.

[3] **Gás liquefeito de petróleo.** Disponível em: <www.supragas.com.br/glp.php. Acesso em:23 de maio de 2017.

[4] **Quimicamente, o que é o Gás Natural?.** Disponível em: <www.goiasgas.com.br/quimicamente,_o_que_e_o_gas_natural_.html>. Acesso em: 20 de maio de 2017.

[5] **Mercado de GLP.** Disponível em: <https://www.liquigas.com.br>. Acesso em: 20 de maio de 2017.

[6] NASHIRO, Arnaldo. **Tendências do mercado GLP no mundo e oportunidades para o Brasil.** Disponível em: <http://www.gasescombustiveis.com.br/encontroglp/Palestras_2016/Tendencias_do_mercado_GLP_12_EVENTO_GLP_ULTRA.pdf>. Acesso em: 25 de maio de 2017.

[7] Gás natural: **potencialidades de utilização no Brasil.** Disponível em: <https://periodicos.ufsm.br/reget/article/viewFile/7896/pdf>. Acesso em: 13 de maio de 2017.

[8] **Anuário Estatístico Brasileiro do Petróleo, Gás Natural e Biocombustíveis 2016.** Disponível em: <http://www.anp.gov.br/wwwanp/publicacoes/anuario-estatistico/2441-anuario-estatistico-2016#Se%C3%A7%C3%A3o%201%20%C2%BF%20Panorama%20Internacional>. Acesso em: 23 de maio de 2017.

[9] **O Firebase ajuda você a criar apps melhores e desenvolver seus negócios.** Disponível em: <https://firebase.google.com>. Acesso em: 20 de maio de 2017.

[10] **Turn your company into an apps company.** Disponível em: <https://www.heroku.com/>. Acesso em: 20 de maio de 2017.

[11] **Firebase é a nova aposta da Google em uma plataforma móvel unificada.** Disponível em: <https://www.tudocelular.com/android/noticias/n72091/google-lanca-nova-versao-do-firebase.html>. Acesso em: 20 de maio de 2017.

[12] **Princípios de funcionamento de um detector de gás.** Disponível em: <http://acessopercon.com.br/percon/sensores-de-gas-principios-e-tecnologias/>. Acesso em: 20 de maio de 2017.

[13] **Gas leakage module using the MQ-5 gas sensor.** Disponível em: <https://wisense.wordpress.com/2015/07/06/gas-leakage-module-using-the-mq-5-gas-sensor/>. Acesso em: 20 de maio de 2017.

[14] **Site oficial da documentação da Espressif.** Disponível em: <https://espressif.com/en/support/download/documents?keys=&field_type_tid%5B%5D=14>. Acesso em: 12 de junho de 2017.

[15] **Site oficial da Smappee**. Disponível em: http://www.smappee.com

Acesso em 24 de junho de 2017.

16] **Trabalho de conclusão do Jefferson Silva Santos**. Disponível em: http://www.repositorio.uniceub.br/bitstream/235/3697/2/Monografia%20JEFFERSON%20SANTOS%202-2012.pdf. Acesso em 24 de junho de 2017.

7. Referências das imagens

Figura 1.
http://noticias.band.uol.com.br/cidades/noticia/100000797300/sp-explosao-por-vazamento-de-gas-deixa-um-morto-e-tres-feridos.html. Acesso em 24 de junho de 2017.

Figura 2.
http://www.acritica.com/channels/manaus/news/vazamento-de-gas-causa-explosao-e-deixa-seis-pessoas-feridas-na-compensa Acesso em 24 de junho de 2017.

Figura 3
http://www.opovo.com.br/noticias/fortaleza/2017/04/apartamento-explode-apos-vazamento-de-gas-moradora-sofre-queimaduras.html Acesso em 24 de junho de 2017.

Figura 4
http://www.otempo.com.br/vazamento-de-g%C3%A1s-causa-explos%C3%A3o-pr%C3%A9dio-desaba-e-cinco-morrem-1.1213443 Acesso em 24 de junho de 2017.

Figura 5
http://g1.globo.com/rio-de-janeiro/noticia/2015/08/predios-da-av-rio-branco-rio-sao-evacuados-apos-vazamento-de-gas.html Acesso em 24 de junho de 2017.

Figura 6
http://www.arduinoecia.com.br/2017/01/upgrade-firmware-esp8266-esp-07.html Acesso em 24 de junho de 2017.

Figura 7
https://portuguese.alibaba.com/product-detail/esp8266-serial-wifi-wireless.html Acesso em 24 de junho de 2017.