Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

Relatório de Estágio

Dispositivo IoT de Monitorização de DataCenter

Cláudio Branco de Vasconcelos

Orientado por:

Professora Ana Lopes

Trabalho final de mestrado apresentado ao Instituto Politécnico de Tomar para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática – Internet das Coisas

Resumo

Este trabalho de estágio surge da necessidade de monitorizar em permanência as condições ambientais do *Datacenter* da Câmara Municipal de Abrantes (CMA), assim como da disponibilidade e conetividade dos seus equipamentos.

As quebras elétricas relativamente frequentes, na Câmara Municipal de Abrantes, requerem o constante funcionamento do equipamento de energia de reserva. Estas solicitações frequentes causam um elevado stress do equipamento, dada a necessidade de alimentar não só todos os equipamentos passivos e ativos, assim como o ar condicionado que se encontra na sala.

A ideia de desenvolver um sistema de monitorização ambiental e alimentação elétrica surgiu após uma quebra prolongada de energia que forçou a equipa a desligar todos os equipamentos manualmente, após um funcionário ter verificado que o local se encontrava extremamente quente. Até à existência deste dispositivo de monitorização, não existia nenhum mecanismo activo que nos alerte de tal facto, a não ser pela monitorização através de uma solução que foi desenvolvida através da integração da UPS (Uninterruptible Power Supply) existente no *DataCenter*, recorrendo ao software disponibilizado pelo fabricante que permite o envio de dados simples dos valores de carga elétrica instantânea. Estes dados são registados numa base de dados InfluxDB e que serve de fonte de dados para um sistema simples de monitorização através da amostragem e representação gráfica dos valores registados, através do *software de* observação e métricas Grafana[1].

Este estágio tem como objetivo o desenvolvimento de um dispositivo IoT que permita monitorizar através de um *heartbeat* a disponibilidade e conectividade do equipamento, temperatura e humidade presente na sala e em caso de alarme, envie uma mensagem de alerta SMS (*Short Message Service*) para um número pré-determinado.

Através do aviso de alerta, é possível aos técnicos tomarem as medidas necessárias de forma atempada.

Abstract

This internship work arises from the need to permanently monitor the environmental conditions of the *Datacenter* of the Municipality of Abrantes (CMA), as well as the availability and connectivity of its equipment.

The relatively frequent electrical outages at the Abrantes City Council require the constant operation of the reserve energy equipment. These frequent requests cause high equipment stress, given the need to supply not only all passive and active equipment, as well as the air conditioning in the room.

The idea of developing an environmental monitoring system and power supply came after a prolonged power outage that forced the team to manually turn off all equipment, after an employee verified that the place was extremely hot. Until the existence of this monitoring device, there was no active mechanism that would alert us to such fact, except for monitoring through a solution that was developed through the integration of UPS (Uninterruptible Power Supply) existing in the *DataCenter*, using the software provided by the manufacturer that allows simple data to be sent from the instantaneous electrical charge values. These data are recorded in an InfluxDB database and serve as a data source for a simple monitoring system through the sampling and graphical representation of the recorded values, using the Grafana[1] observation and metrics software.

This stage aims at the development of an IoT device that allows to monitor through a heartbeat the availability and connectivity of the equipment, temperature and humidity present in the room and in case of alarm, send an SMS (Short Message Service) alert message to a predetermined number.

Through the warning notice, it is possible for technicians to take the necessary measures in a timely manner.

Agradecimentos

Quero agradecer aos meus pais, pela paciência e dedicação que sempre demonstraram e que me permitiu chegar ao que sou hoje.

Ao meu irmão Daniel de Vasconcelos, pelo apoio que me tem sempre dado e por estar sempre disponível para me ouvir e dar a sua opinião.

À Sandra, a minha esposa e as minhas duas filhas magníficas, meigas e sempre presentes, que me têm apoiado incondicionalmente ao longo dos tempos.

Aos meus amigos e conhecidos, que fazem parte do leque social para mantermos um convívio, troca de experiências, conselhos e momentos de diversão.

À Professora e Orientadora de Mestrado, Ana Cristina Lopes, pelo seu apoio na realização deste trabalho.

Ao Município de Abrantes, por ter aceite a concretização do estágio.

Índice

Índice de figuras

Índice de tabelas

Acrónimos

DC – Data Center (Centro de dados)

API - Application Programming Interface (Interface de programação de aplicações)

REST - Representational State Transfer (Transferência de estado representacional)

IDE – Integrated Development Enviornment (Ambiente de desenvolvimento integrado)

OOP - Object-Oriented Programming (Programação orientada a objetos)

RGPD – Regulamento Geral de Proteção de Dados

RaspPI – Raspberry PI (mini computador de secretária)

wPI – Wiring Pi

MOSFET – Metal-Oxide-Semiconductor Field-Effect Transístor

ARPA - Advanced Research Projects Agency

ARM – Advanced RISC Machine

RISC – Reduced Instruction Set Computing

HAT – placa de expansão para Raspberry Pi

HAT's – placas de expansão para Raspberry Pi

HDMI – Standard para ligação vídeo de alta-definição

GSM – Global System for Mobile Communications

GPRS – General Packet Radio Service

SMS – Short Message Service

IEEE 802.15.4e – Standard que define operação de redes pessoais sem fios de baixa taxa de transferência de dados

LoRa – Protocolo de rede de área ampla e baixo consumo

CSV – Comma-Separated Values (ficheiro de valores separados por vírgula)

JSON – Javascript Object Notation

SPI – Serial Peripheral Interface

I2C – Inter-Integrated Circuit

GPIO – General Purpose Input/Output

UART – Universal Asyncronous Receiver/Transmitter

PCM – Pulse Code Modulation

SCL – Relógio I2C

CGI – Common Gateway Interface

AT – Abreviatura para ATtention

UPS - Uninterruptible Power Supply

HTTP - Hypertext Transfer Protocol

CRUD – Create, Read, Update and Delete

TCP – Transmission Control Protocol

1 Introdução

1.1 Enquadramento tecnológico atual

A tecnologia, que na sua definição mais teórica pode ser definida como "*uma entidade material criada pela aplicação de esforço físico e mental à natureza, a fim de alcançar algum valor*" [2], desempenha cada vez mais, um papel fundamental na vida das pessoas, assim como nas organizações. A procura incessante por melhorar, otimizar e baixar custos de processos através da tecnologia, a necessidade de automação em ambientes hostis ou onde o erro humano pode ser um fator crítico, tem permitido observar um desenvolvimento acelerado, no campo do software e hardware. Desta constante evolução, resulta também o aparecimento de micro computadores, sensores e placas de expansão, que permitem desenhar e desenvolver toda uma nova indústria baseada nestes dispositivos, que através da integração e combinação destes, é possível obter dados em tempo real. Através destes sensores, que podem ser de monitorização das condições ambientais, som, luminosidade, condutividade elétrica, para referir alguns.

Com o aparecimento e desenvolvimento da Internet[3][4][5][6], uma rede de computadores interligados derivada do projeto *ARPANET*[7], criado pela Agência de Investigação de Projetos Avançados (ARPA) do Departamento de Defesa dos Estados Unidos, a globalização de pessoas, equipamentos e dispositivos tornou-se uma realidade, ao permitir em qualquer lugar e altura, aceder remotamente, controlar, monitorizar e tomar ações, tal como se estivéssemos fisicamente presentes no local.

A miniaturização de circuitos integrados e microprocessadores e as possibilidades oferecidas por uma rede de computadores interligada ao nível global, são os percursores do conceito de Internet das Coisas[8], que trouxe consigo, uma geração de dispositivos com capacidade computacional e

interligados, autónomos ao não necessitarem da intervenção humana, pela convergência e utilização de diferentes tecnologias como a analítica em tempo real[9][10], sensores diversificados[11], automação[12][13] e inteligência artificial[13].

1.2 Motivação e Objetivos

A necessidade de monitorizar permanentemente uma área vital ao funcionamento do Município de Abrantes, quer ao nível das condições ambientais quer à alimentação elétrica, que representa o *DataCenter*, implicaria uma equipa dedicada com rotatividade de horários em permanência no local, situação que obrigaria a custos e contratação de recursos humanos adicionais. Ao utilizar um dispositivo baseado e construído com um micro computador e sensores, é possível monitorizar, alertar e executar ações específicas por forma a minimizar ou eliminar o risco de sobre-aquecimento do espaço confinado do *DataCenter*, a quebra abrupta do funcionamento dos equipamentos ativos por falta de energia elétrica por longos períodos e em simultâneo, alertar de forma autónoma independentemente da quebra das ligações de rede para o exterior, o administrador de sistemas ou a equipa responsável pelo mesmo.

1.3 Descrição da solução

A solução passa por usar um micro computador, neste caso, a utilização de um *Raspberry Pi* que apesar de construído sob uma arquitectura ARM, permite a utilização de um Sistema Operativo completo baseado em Linux,

como o *raspbian*. A vantagem por uma solução baseada num micro computador ao invés de um microcontrolador, é a disponibilidade de recursos e bibliotecas, que aceleram o desenvolvimento da solução.

Uma das vantagens de utilização do RaspPi é a capacidade de adicionar placas de expansão, vulgo *HAT's (Hardware attached on top)*, neste caso um módulo GSM/GPRS para permitir o envio de mensagens SMS (*Short Message Service*) e estabelecer uma ligação à internet independente da ligação disponibilizada pelo equipamento de rede do *DataCenter*. Deste modo, mesmo com o *DataCenter* num estado de *offline*, é possível avisar o Administrador de Sistemas, estabelecer uma ligação remota e executar comandos na linha de comandos.

O dispositivo tem também a capacidade de monitorizar as condições ambientais presentes no *DataCenter*, avisando de igual forma, caso a temperatura suba acima de um nível pré-definido, pela utilização de um sensor do tipo DHT11[14].

A solução passa também pelo desenvolvimento de uma API (*Application Programming Interface*) que regista os valores dos sensores para análise por uma solução de analítica disponível no Município.

1.4 Contributos deste trabalho

Com este projeto, foi possível introduzir uma nova camada de proteção a eventos provenientes de terceiros, que possam pôr em risco o correto funcionamento do *DataCenter* do Município de Abrantes, quer a nível de alimentação elétrica, quer a nível de condições ambientais no espaço físico.

Essas novas medidas foram obtidas através da implementação das seguintes contribuições:

1 - Projeto e desenvolvimento de um sistema IoT baseado em Raspberry PI com sistema operativo Linux Debian, complementado com uma placa de expansão GSM/GPRS SIM800, sensor de condições ambientais DHT11 e sensor de continuidade elétrica SCT-013-000 com conversão de leituras analógicas para digitais através de um conversor ADC MCP3008.

2 – Desenvolvimento do software de monitorização recorrendo ao Python e com implementação de módulos dos fabricantes ou de terceiros, para obter leituras dos sensores.

3 – Desenvolvimento de um módulo de alarmística em Python, responsável pelo envio de mensagens SMS (*Short Message Service*) para um terminal móvel, com o estado atual das condições ambientais e elétricas, quando os valores obtidos dos sensores ultrapassem parâmetros pré-definidos.

4 – Desenvolvimento de uma API (*Application Programming Interface*) em PHP sobre servidor *web* NGINX e protocolo HTTP (*Hypertext Transfer Protocol*), com implementação do método POST derivado das funções básicas de persistência de dados (CRUD – *Create, Read, Update and Delete*) para registo em base de dados dos valores dos sensores.

5 – Desenvolvimento de um relatório em IBM Cognos Analytics para amostragem dos valores dos sensores em intervalos temporais e com a finalidade de analítica e apoio à decisão.

2 Entidade de Acolhimento

2.1 Introdução

Abrantes é uma cidade portuguesa pertencente ao distrito de Santarém, na província do Ribatejo, região do Centro e sub-região do Médio Tejo, com cerca de 18 500 habitantes. É sede de um município com 714,69 km² de área. O município, cujo brasão pode ser observado na Fig. 1, é limitado a norte pelos municípios de Vila de Rei, Sardoal e Mação, a leste por Gavião, a sul por Ponte de Sor e a Oeste por Chamusca, Constância, Vila Nova da Barquinha e Tomar.

Figura 1 - Brasão da Câmara Municipal de Abrantes, retirado de [46]

2.2 Missão

Desenvolvimento Económico e Social do concelho de forma a proporcionar a melhoria das condições gerais de vida, de trabalho e de lazer dos seus habitantes, no respeito pelo ambiente, património edificado e no legítimo interesse pelas minorias.

2.3 Visão

O Município orienta a sua ação no sentido de promover e dinamizar o concelho a nível económico, social e ambiental, primando a sua atuação por modelos de gestão de excelência que permitam a otimização sustentável dos seus recursos.

2.4 Valores

No processo de governação autárquica, a CMA, no seu todo, deve pautar a sua atuação de acordo com os princípios éticos que regem a prática de um serviço público de excelência e que consagrem, cumulativamente, uma identidade própria em defesa do interesse público. Neste sentido, o Executivo define como Valores Organizacionais da CMA: respeito, integridade, serviço público, excelência e responsabilidade.

2.5 Organograma

A câmara de Abrantes é atualmente constituída pro 13 divisões (ver Fig. 2), cada uma deles com os seus respetivos serviços de acordo com a sua área de intervenção. Estando a proteção civil, o gabinete de apoio à presidência, o serviço de auditoria interna e a equipa para a gestão inteligente dos territórios sob a alçada do executivo. O serviço de Administração de Sistemas e o serviço de Desenvolvimento Aplicacional fazem parte da Divisão de Gestão das Pessoas e dos Sistemas de Informação. Estes dois serviços tiveram um papel fundamental no desenvolvimento deste projeto.

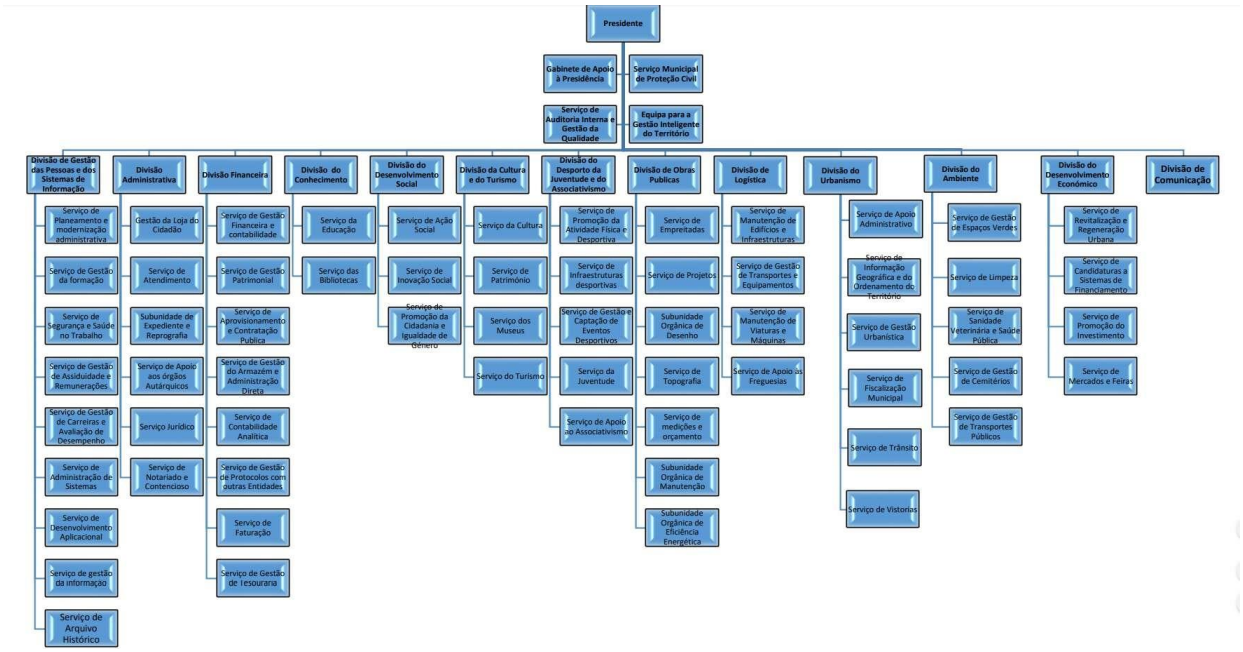

Figura 2 - Organograma Câmara de Abrantes, retirado de [47]

3 Estado da arte em dispositivos de monitorização

"In a few decades time, computers will be interwoven into almost every industrial product"
Karl Steinbuch, German computer science pioneer, 1966

Motivado por uma solução muito específica de monitorização, procurou-se em primeira mão, perceber e fazer um levantamento das atuais soluções disponíveis no mercado. O resultado desta pesquisa, permitiu não só contribuir para aperfeiçoar a solução em desenvolvimento, assim como trazer um fator de inovação ao produto.

3.1 Projetos enquadrados no âmbito deste trabalho

Este capítulo pretende fazer uma análise das várias soluções disponíveis no mercado e que se enquadrem nos objetivos propostos deste projeto, quer a nível de funcionalidade quer a nível tecnológico utilizada. Será efetuada uma descrição de cada solução encontrada bem como a tecnologia utilizada.

3.1.1. Advantech Smart IoT

A Advantech[15], uma empresa Norte Americana e com presença em Taiwan e Irlanda, dispõe de uma solução de monitorização recorrendo a pequenos dispositivos com sensores de temperatura e controlo de corrente elétrica. A solução passa pela comunicação entre cada dispositivo através de uma rede WiFi IEEE 802.15.4e, através da qual comunicam entre si os seus dados, até chegar ao dispositivo principal que funciona como um *gateway* ou *edge router*.

Algumas das funcionalidades disponibilizadas por esta solução são:

- Modularidade de sensores
- Controlo de corrente eléctrica e de temperatura
- Comunicação dos dados por rede WiFi
- Disponibilização dos dados para aplicações de terceiros
- Solução de hardware e software proprietárias

Na Figura 3 é possível ver uma implementação típica da solução *Smart IoT Wzzard*[16]

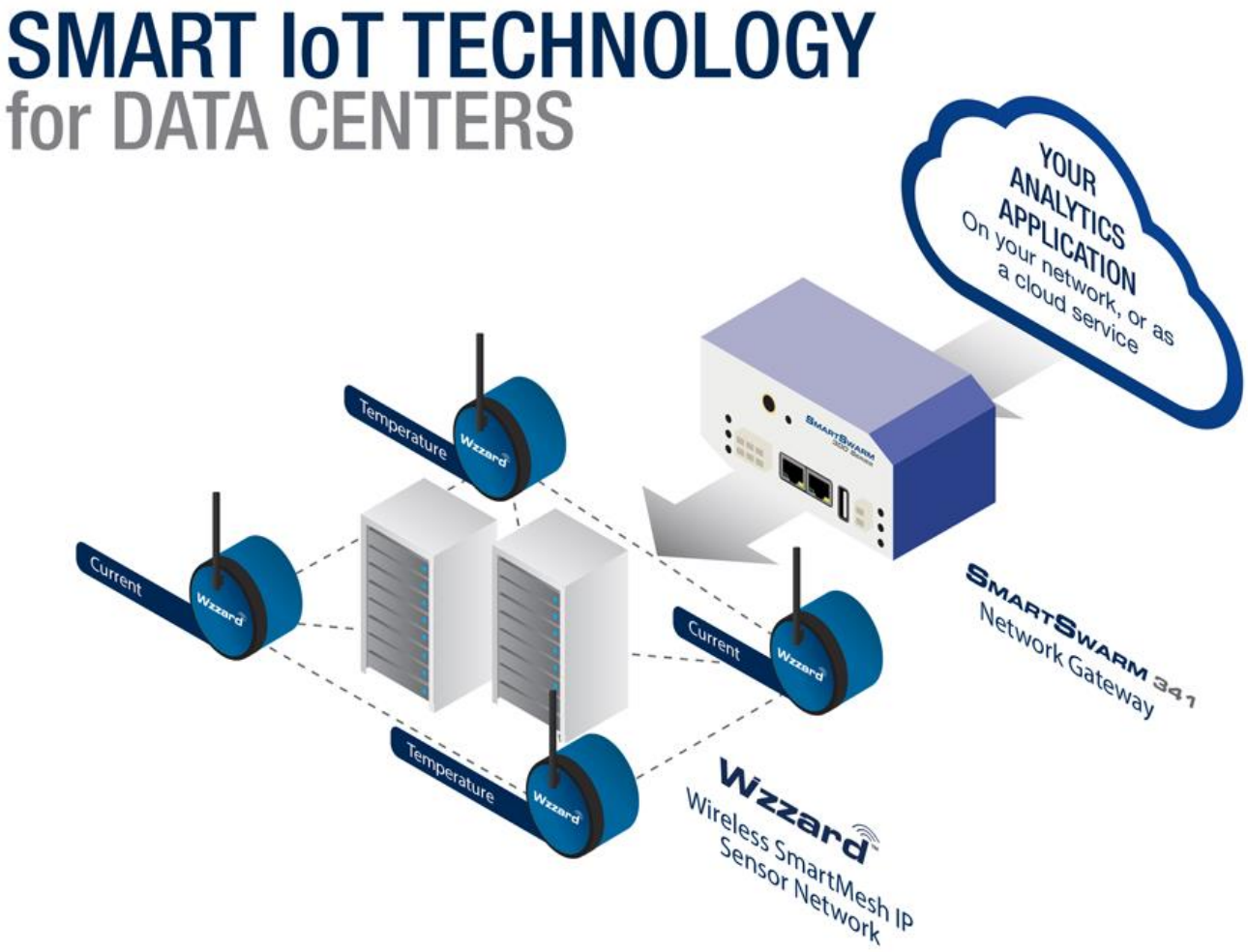

Figura 3 - Solução Smart IoT Wzzard, retirado de [48]

3.1.2. AKCP

A empresa Norte Americana AKCP[17] apresenta uma solução de monitorização de temperatura e humidade, fugas de água e diferencial de pressão de ar, através da instalação de um equipamento de montagem em bastidor. Esta solução aposta também numa forte componente de software, ao disponibilizar um portal web, que permite a parametrização da monitorização através do arrasto de controlos para um painel que pretende representar o bastidor físico e fazer uma reprodução animada do mapa térmico que ocorre dentro do mesmo, como se pode observar na Fig. 4, através da solução AKCP Pro Server[18].

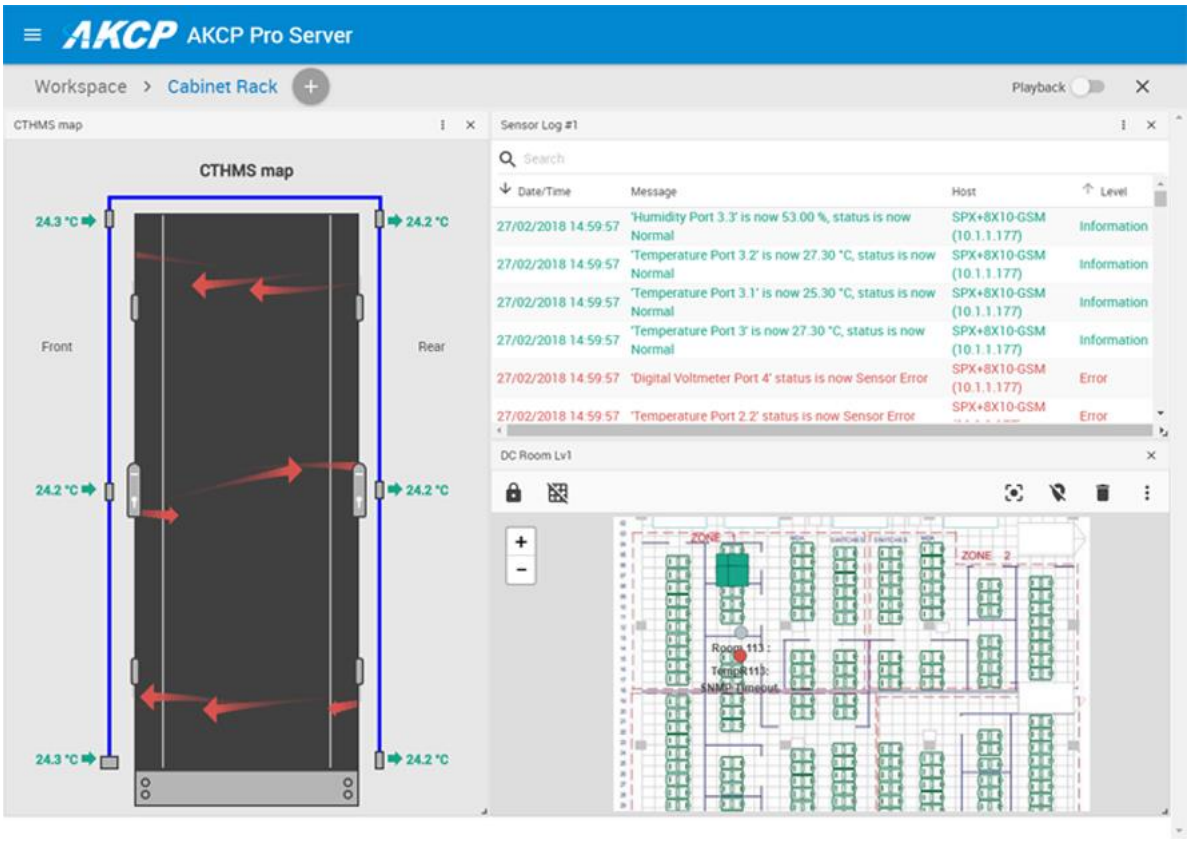

Figura 4 - Solução de software AKCP Pro Server, retirado de [49]

As soluções apresentadas são baseadas em hardware e software proprietárias.

Funcionalidades e características disponíveis nesta solução:

- Sensores para temperatura e humidade, fugas de água e diferencial de pressão de ar
- Portal apelativo com animações e intuitivo
- Mapeamento térmico
- Solução proprietária

Na Figura 5 é possível ver alguns dos sensores de controlo de condições ambientais[19] do fabricante.

Figura 5 - Sensores disponíveis para instalação em Data Center (da esquerda para a direita: sensor de temperatura e humidade, sensor com mapeamento térmico, sensor de fugas de água), retirados de [50]

3.1.3. Xtralis Vesda-E

Com sede na Suiça e espalhada por vários continentes, a empresa Xtralis[20] desenvolveu um dispositivo de deteção preventiva de incêndios em Data Centers. A particularidade deste dispositivo é a capacidade de detectar que tipo de partículas se encontram no fumo, de modo a identificar se se trata de poluição proveniente do ambiente onde o ar fresco é extraído e evitar falsos

alarmes. O ponto forte deste produto e no que o fabricante se apoia, é manter um detetor de incêndios preciso, por comparação a outros produtos similares, onde se reduz a sensibilidade para evitar falsos alarmes.
O fabricante também aposta na componente de software, ao disponibilizar em cada dispositivo, um servidor de páginas web embebido, para monitorização e configuração e interfaces de ligação como *ethernet*, *Wi-Fi* e USB (Universal Serial Bus). Na Fig. 6 podemos ver os dispositivos VESDA-E[21].

Principais funcionalidades e características desta solução:

- Deteção preventiva e precisa de incêndios
- Redução ou eliminação de falsos alarmes
- Servidor de páginas web embebido no dispositivo
- Interfaces de ligação *Ethernet*, *Wi-Fi* e USB
- Solução proprietária

Figura 6 - Dispositivos VESDA-E, retirado de [51]

3.1.4. SensorPush

Orientado mais para o mercado doméstico, a empresa SensorPush com base em Nova Iorque nos Estados Unidos, desenvolveu sensores e um router para medir temperaturas e humidades nos locais onde os sensores são colocados.
Dotados de tecnologia Bluetooth, cada dispositivo de medição pode comunicar os dados com um smartphone, através da aplicação desenvolvida para o efeito pela empresa. Ao adicionar um router *SensorPush G1 WiFi Gateway*, é possível monitorizar todos os sensores através da comunicação entre dispositivos com *WiFi* e upload dos dados das leituras, para portal na internet.
De acordo com a SensorPush[22], o seu sistema consiste num produto fácil de configurar e fácil de usar, para o utilizador comum. Em baixo, na Fig. 7 podemos ver os dispositivos comercializados pela SensorPush.

Principais funcionalidades e características desta solução:

- Orientado apenas para o mercado doméstico
- Apenas leitura de temperatura e humidade
- *Gateway WiFi* que permite interligar vários dispositivos
- Aplicação *Smartphone* e ligação *Bluetooth* V4 aos dispositivos
- Envio de dados para portal na internet
- Tecnologia proprietária

Figura 7 – Gateway[23] e sensor[24] da empresa SensorPush

3.1.5. Fludia

Apresentando algumas soluções em IoT (Internet of Things), destaca-se o FM432ir – IoT da empresa francesa Fludia[25]. Este pequeno dispositivo com apenas capacidade para leituras óticas, acoplado diretamente ao mostrador de

um contador de eletricidade físico, consegue de uma forma não intrusiva e de fácil instalação, medir o consumo em tempo real e comunicar através de LoRaWAN 1.0. Tem a capacidade de produzir ficheiros simples de CSV ou JSON para posterior utilização em soluções de terceiros.

Na Figura 8 é possível ver uma solução baseada no leitor ótico FM432ir[26] e dispositivo de comunicação FM432p[27] LoRaWAN comercializado pela Fludia.

Figura 8 - Imagem do leitor ótico FM432ir instalado num contador, retirado de [26] e [27]

Principais funcionalidades e características desta solução:

- Apenas leitura de consumo do contador de eletricidade
- Comunicação por protocolo LoRaWAN
- Exportação de dados em CSV ou JSON
- Solução proprietária

3.1.6. Conclusão

Nas várias propostas que foram apresentadas, verifica-se que todas elas são baseadas em tecnologia proprietária, que se traduz geralmente numa solução mais onerosa, assim como o problema de ficar dependente da tecnologia do fabricante e da estratégia comercial deste. Verifica-se também que à exceção da solução da empresa Advantech, as soluções apresentadas apenas funcionam numa das áreas que este projeto propõe, seja o consumo instantâneo elétrico, temperatura e humidade e comunicação dos dados ou exportação dos mesmos para posterior tratamento.

Em baixo apresenta-se a Tabela I com uma comparação entre tecnologias e dispositivos feitos neste capítulo.

	Advantech Smart IoT	*AKCP*	*Xtralis Vesda-E*	*SensorPush*	*Fludia*
Controlo Temperatura	Sim	Sim	Não	Sim	Não
Controlo Humidade	Não	Sim	Não	Sim	Não
Controlo Eléctrico	Sim	Não	Não	Não	Sim
Comunicação GSM/GPRS/WiFi	Sim – Wifi com edge router	Não	Sim	Sim – Wifi com gateway	Sim - LoRaWAN 1.0
Analítica / API / Web Interface	Sim - API	Sim – Web Interface embebido	Sim – Web Interface embebido	Sim - Portal	Sim – Ficheiros CSV
Outros sensores	Não	Fugas de Água e diferencial de pressão de ar	Prevenção de incêndios por análise de partículas no ar	Não	Leitura ótica

Tabela I – Comparação de tecnologias e dispositivos

4 Arquitetura do dispositivo IoT de Monitorização

4.1 Tecnologias utilizadas

Este projeto recorre a várias tecnologias seja em termos de hardware, como ao nível do software. A utilização de um computador de placa-simples garante uma plataforma coesa, de baixo custo e facilmente acessível, que apresenta um baixo consumo ao usar processadores eficientes energeticamente e de arrefecimento passivo. A expansão de funcionalidades através de placas de expansão, possibilita um quase número infinito de combinações, limitadas apenas pela sobreposição de comunicação resultante da utilização dos mesmos pinos GPIO do Raspberry Pi.

Ao permitir o desenvolvimento do software com linguagens mais acessíveis como o Python, é possível uma prototipagem, testes e produção da componente lógica, mais rápida e propensa a menos erros de desenvolvimento.

4.2 Hardware

4.2.1 Raspberry Pi 2 Model B

O Raspberry Pi 2 Model B[28] apresenta-se como um computador de placa-simples, do tamanho de um cartão de débito/crédito bancário, de baixo custo e consumo energético. Tem como principais características as seguintes:

- Sistema Operativo oficial baseado em Linux Debian
- Processador ARM Cortex-A7 de 900MHz com possibilidade de *overclock* a 1000MHz
- Memória RAM de 1GB
- 4 portas USB geração 2
- Placa *ethernet* integrada 100Base-T

- Saída de vídeo HDMI

Na Figura 9 é possível ver o Raspberry PI 2 modelo B.

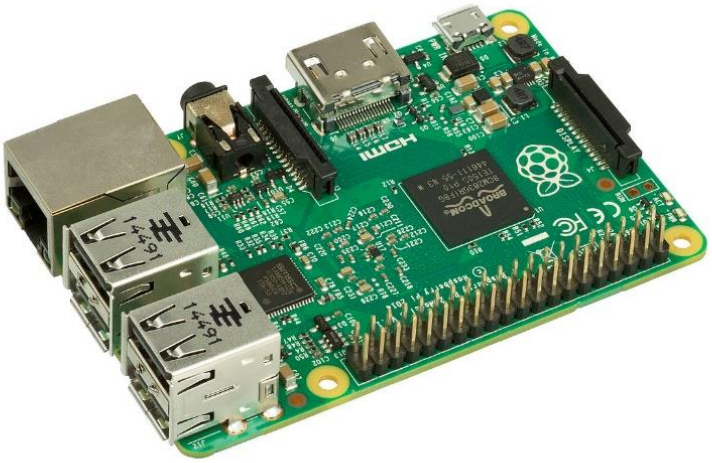

Figura 9 - Computador de placa-simples Raspberry Pi 2 Model B

4.2.2 Placa de expansão SIM800 GSM/GPRS V2.0

A placa de expansão GSM V2.0[29] é desenvolvida especificamente para a interface do Raspberry Pi, baseada no módulo SIM800 de quatro bandas GSM / GPRS / BT. Os comandos AT podem ser enviados através da porta de séria no Raspberry Pi, permitindo deste modo, funções como marcar e atender chamadas, enviar e receber mensagens SMS (*Short Message Service*) assim como estabelecer ligações de internet. Principais características:

- Desenvolvido para o Raspberry Pi
- Quatro bandas de funcionamento – 850 / 900 / 1800 / 1900 MHz
- GPRS multislot classe 12
- Capacidade de empilhar com outros dispositivos

Na Figura 10 podemos ver a placa de expansão SIM800.

Figura 10 - Placa de expansão SIM800 GSM/GPRS

4.2.3 Sensor DHT11

O sensor DHT11[30] permite expandir as funcionalidades do Raspberry Pi com a capacidade de fazer medições de condições ambientais no espaço onde se encontra. De baixo custo e simples utilização, este sensor consegue leituras de razoável precisão. Na Fig. 11 é mostrado o sensor DHT11. Principais características do sensor:

- Baixo custo
- Funcionamento de 3V a 5V
- Leitura de humidade entre 20-80% com precisão de 5%
- Leitura de temperatura entre 0-50ºC com precisão de +/- 2ºC
- Compacto e leve

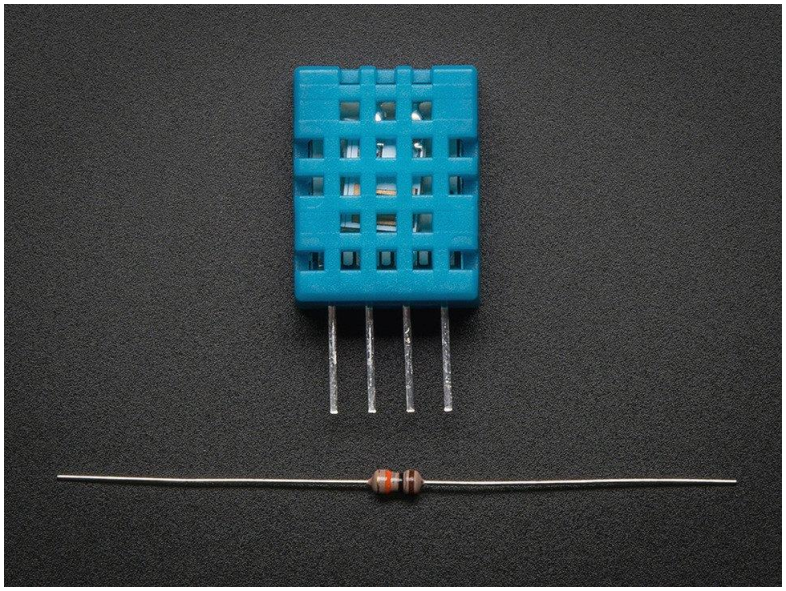

Figura 11 - Sensor de temperatura / humidade DHT11

4.2.4 Sensor não-intrusivo de 100A SCT-013-000

O sensor não-intrusivo SCT-013-000[31] funciona pela indução magnética gerada num dos cabos condutores do fio elétrico. O fio elétrico tem uma pequena secção descarnada a fim de separar os fios condutores e passar um dos fios pelo sensor. Na Fig. 12 é possível ver o sensor. Principais características do sensor:

- Baixo custo
- Simplicidade de utilização

Figura 12 - Sensor não-intrusivo SCT-013-000

4.3 Software – Linguagens de Programação

4.3.1 Python

Python[32] é uma linguagem de programação orientada a objetos, interpretada e interativa. Esta linguagem de programação combina poder com uma sintaxe muito clara. Possui módulos, classes, exceções, tipos de dados dinâmicos de alto nível e tipos dinâmicos. O desenvolvimento de novos módulos integrados escritos em C ou C ++ permitem expandir a capacidade base do software. O *Python* também pode ser usado como uma linguagem de extensão para aplicações escritos noutras linguagens que precisam de scripts fáceis de usar ou interfaces de automação.

4.3.2 C/C++

Consideradas linguagens de baixo nível, o C e C++ permitem um maior controlo sobre o hardware, motivo pelo qual, são usadas para expandir também o *Python* através de módulos. Existe uma diferença fundamental entre o C e o C++, sendo que a primeira é processual, ou seja, contém uma série de etapas computacionais a serem realizadas, enquanto que a última, é uma linguagem de programação orientada a objectos, considerada como uma extensão do C, sendo que C++ poderá ser chamada de "*C with Classes*".

São linguagens de programação mais complexas, que permitem a execução diretamente no hardware do dispositivo, permitindo também um maior controlo da memória, processamento e dos recursos existentes.

4.3.3 PHP

PHP[33] (*Hypertext Preprocessor*) é uma linguagem de scripting generalizada, especialmente adequada para desenvolvimento *web*. O código PHP é processado num servidor *web*, através de um interpretador que pode

ser implementado como um módulo, *daemon* ou um executável CGI (*Common Gateway Interface*). O código PHP é interpretado e executado no servidor web, que pode ser qualquer tipo de dados, como HTML gerado ou imagem e que se tornam no resultado da reposta do protocolo HTTP. Desde 2014 que foi criada uma especificação formal do PHP ao nível de sintaxe e semântica que se enquadra num domínio permitindo reutilização de componentes.

4.3.4 Conclusão

Das linguagens referidas, neste projeto existe uma utilização para componentes específicos, em que de um modo geral, é usado o *Python* pela sua simplicidade e disponibilidade de módulos dos fabricantes do hardware usado neste projeto, e que permite uma rápida prototipagem e testes das funcionalidades. O C/C++ quando existe a necessidade de interagir mais a nível de hardware e quando os módulos disponíveis em *Python* não contemplem ou não correspondam às necessidades pontuais de uma determinada funcionalidade. Finalmente o PHP usado no desenvolvimento de uma API para registo dos valores dos sensores em base de dados.

4.4 IDE utilizadas para o desenvolvimento

Das ferramentas mais importantes no desenvolvimento aplicacional são as IDE (*Integrated Development Environment*). Existem opções no mercado, cuja finalidade pode ser mais generalizada, ou mais orientada para uma tecnologia em específico. Os IDE permitem uma coesão e rapidez no desenvolvimento através de funções e automatismos, preenchimento

automático com escrita inteligente e identificação de erros na sintaxe. Nas próximas subsecções, é realizada uma análise de algumas IDE e, no final, procura-se identificar as suas principais diferenças.

4.4.1 Eclipse

O Eclipse é uma das IDE mais populares. Suporta vários tipos de linguagem, como PHP, C e C++ entre outras. Além disso, com recurso ao Eclipse Marketplace é possível outro tipo de linguagens e extensões.
Algumas das funcionalidades suportadas são:

- Suporte a vários tipos de linguagens de programação
- Versões mais recentes do Java
- Suporte a temas personalizados (layout / cores)
- Uma enorme variedade de plug-ins de plataforma
- Uma grande comunidade

4.4.2 Visual Studio

O Visual Studio da Microsoft é outra proposta popular. Composta por várias versões, dispõe também de um conjunto de funcionalidades a quem pretende fazer desenvolvimento. Suporta várias linguagens de programação, sendo o .NET a preferencial. O Visual Studio da Microsoft tem um interface bastante apelativo e que oferece um grande conjunto de extensões. Estas são algumas das funcionalidades presentes no Visual Studio:

- Correção de código rápida
- Depuração e diagnóstico de erros

- Grande número de extensões
- Ferramenta de testes em desenvolvimento
- Layout personalizado
- Editor inteligente

4.4.3 Conclusão

Além das IDE aqui referidas, existem outras que também oferecem algumas das funcionalidades mencionadas. Pelas linguagens escolhidas neste projeto, o Eclipse foi escolha mais acertada, principalmente pelo seu cariz de código aberto e forte comunidade. O Visual Studio seria uma boa opção, caso o projeto fosse desenvolvido inteiramente em C/C++, ao integrar uma forte componente de *debug* e escrita inteligente para este tipo de projetos.

4.5 Base de dados

4.5.1 MariaDB

A base de dados usada no projeto é um derivado do MySQL[34], neste caso a MariaDB[35] e que segue os mesmos comandos e funcionamento do primeiro. Sendo a base de dados MariaDB relacional, integra de forma nativa com a solução de analítica em *IBM Cognos Analytics* que o Município já dispõe.

4.5.2 Conclusão

Dos diferentes motores de base de dados existentes no mercado, a utilização para a MariaDB recaí pela utilização noutros projetos internos do Município.

4.6 Tecnologia de placas de expansão do Raspberry Pi

Nos modelos A e B do Raspberry Pi, o conector GPIO (general purpose input/output) tem 26 pinos o que faz com que seja necessário identificar quais os *drivers* necessários para a placa de expansão e posteriormente editar alguns ficheiros do Sistema Operativo *Linux*, para os fazer iniciar aquando do boot do sistema, para os apresentar e poder utilizá-los. O Raspberry Pi não tem conhecimento das placas ligadas e os drivers quando carregados, simplesmente assumem um acesso em exclusivo à interface GPIO.

O Raspberry Pi modelo B+ foi desenhado tendo em conta as placas de expansão, através da introdução do conceito 'HAT' (Hardware Attached on Top) e que em conformidade com uma série regras, permite facilitar a utilização de várias placas. Esta funcionalidade permite ao modelo B+ identificar o 'HAT' que está ligado e configurar automaticamente os GPIO e *drivers* para a placa.

4.7 Controlo e revisão de código

Em equipas de desenvolvimento compostas por vários elementos, é cada vez mais uma prioridade um sistema de controlo de versões de código. Estes sistemas permitem o versionamento do mesmo ficheiro, sendo possível aos programadores, recuperar uma versão antiga, ou consolidar novas versões com a união de vários ficheiros em que determinados blocos foram desenvolvidos por pessoas diferentes. Nas subsecções a seguir, são apresentados os sistemas mais conhecidos ou comuns.

4.7.1 Github

Tendo como linha de inspiração "GitHub é como as pessoas constroem software" [36], atualmente dispõe de uma comunidade de 50 milhões* de utilizadores.

** Em Agosto de 2019*

As principais funcionalidades apresentadas por esta ferramenta são as seguintes:

- Revisão de código
- Gestão de projeto
- Integrações
- Gestão de equipas
- Documentação
- Alojamento de código
- Repositórios

É possível alojar código em repositórios públicos e privados e partilhar com outras equipas ou pessoas. Uma das grandes funcionalidades é o versionamento de código possibilitando a reversão a qualquer altura. O serviço é apresentado na forma de SaaS (Software as a Service), tendo sido recentemente adquirido pela Microsoft[37].

4.7.2 GitLab

GitLab é uma empresa de código aberto que desenvolve software para o ciclo de vida do desenvolvimento de software. Tendo mais de 100000 organizações, 30 milhões de utilizadores registados e uma comunidade activa de mais de 3000 colaboradores [38].

As suas principais funcionalidades são as seguintes:

- Análise de ciclos
- Gestão do código fonte
- Revisão de código
- Wiki
- Snippets
- IDE Web
- Analise código
- Repositórios

À semelhança do Github, oferece praticamente as mesmas funcionalidades. O quadro da Fig. 19, o qual foi retirado de [39], evidencia algumas diferenças entre estas duas soluções:

Version Control Comparison	GitHub	GitLab
Free Private Repos		✓
Free Public Repos	✓	✓
Merge Request / Issue Templates	✓	✓
Integrated CI	*	✓
Enterprise Plans	✓	✓
Integrated Project Board	✓	✓
Navigation Usability	✓	✓
Open Source		✓
Self-Hosted Option	✓	✓
Discoverability	✓	
AD/LDAP	✓	✓
LFS File Storage	✓	✓
Integrated Time Tracking	*	✓
Integrated Review Apps	*	✓
Free, Public Static Web Pages	✓	✓
Burndown Charts, Project Analytics		✓
Third Party tool integration	✓	✓

Figura 13 - Alguma diferenças entre o Github e GitLab, retirado de [14]

4.7.3 SVN

Fundado em 2000 pela CollabNet, Inc., o projeto e o software *Subversion* é atualmente um projeto Apache de nível de topo, desenvolvida e usada por uma comunidade global de colaboradores.

As suas principais funcionalidades são as seguintes:

- Versionamento
- *Commit´*s atómicos
- Bloqueio de ficheiros

- Resolução de conflitos interativo
- Controlo de fusões
- Repositórios

4.7.4 Conclusão

Dos sistemas apresentados todos eles suportam as funcionalidades básicas controlo de versões. Existem um número de especificações que podem de certa forma influenciar a escolha, que recaem, por exemplo na rapidez dos "commit´s", a capacidade de trabalhar em modo offline, o número de repositórios e a fiabilidade do sistema. O SVN é um dos sistemas de versionamento mais utilizado e mais populares, no entanto a escolha pela plataforma GitLab ocorre simplesmente pelo facto de ter sido o sistema adotado internamente no Município de Abrantes.

5 Desenho da arquitetura da solução

Na Figura 14 é possível ver um diagrama da solução proposta. A sua descrição detalhada é descrita nas próximas subsecções.

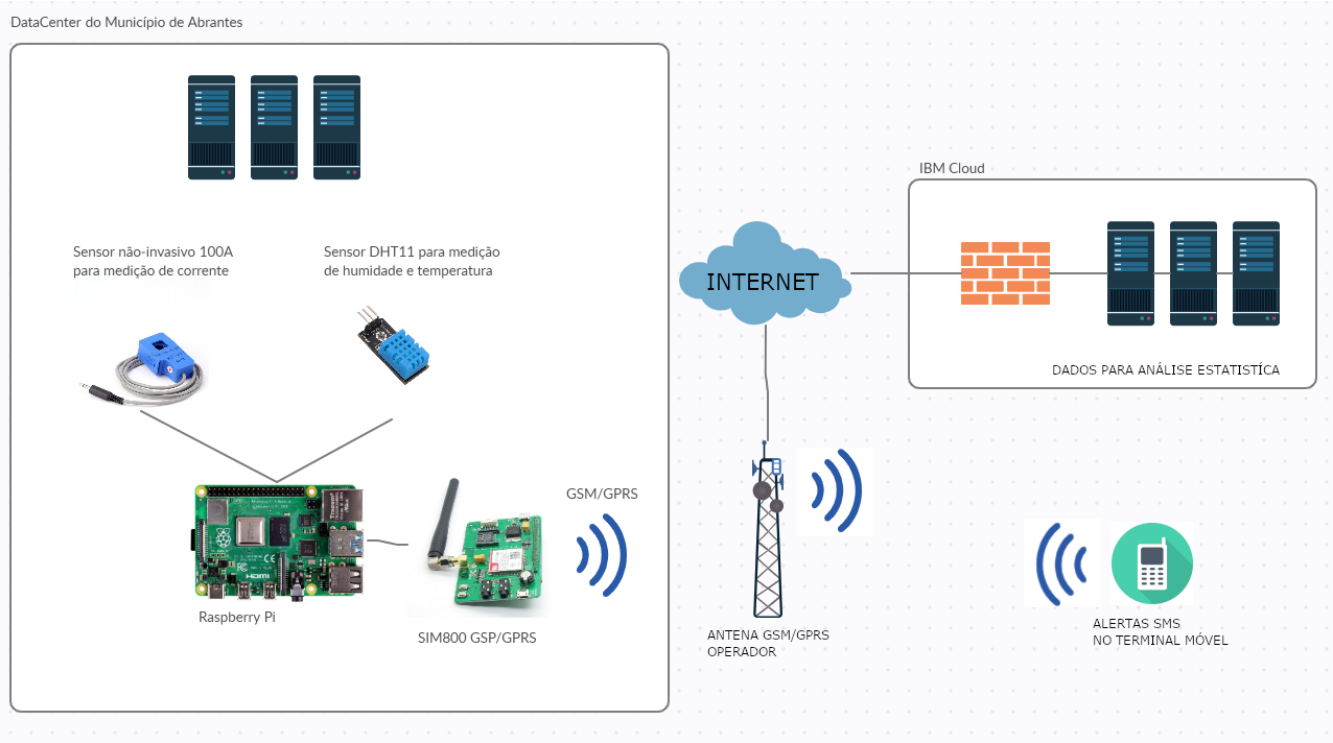

Figura 14 - Diagrama geral da solução

5.1 Camada aplicacional

A camada aplicacional é responsável por monitorizar e enviar os dados dos sensores para a *Cloud* da IBM. Estes dados serão posteriormente visualizados numa solução gráfica.

5.1.1 Raspberry Pi – Sistema Operativo e Programação

A solução desenvolvida, assenta na capacidade do Raspberry Pi, como um micro-pc, disponibilizar uma régua GPIO (General Purpose Input/Output) programável[40], que permite a comunicação com as placas de expansão, a fim de obter os valores dos sensores. Para começar o desenvolvimento, é necessário instalar um Sistema Operativo para arquiteturas ARM, sendo que o fabricante, neste capítulo, disponibiliza várias versões do Sistema Operativo[41] para ARM baseado em *Debian* com funcionalidades mínimas, a uma solução completa com ambiente de trabalho. Neste projeto, foi usada a versão do Sistema Operativo com funcionalidades mínimas, uma vez que o dispositivo funciona num modo *standalone* e comunica os dados através da API (Application Programming Interface).

Para inicar com o Sistema Operativo no Rapsberry Pi, é necessário descarregar uma imagem da página do fabricante e gravar a imagem, que se encontra em formato RAW, num cartão MicroSD. Para tal é executado o comando seguinte a partir de uma *Shell Linux*:

```
# dd if=rasppi-lite-x.img of=/dev/usb1 bs=4096m
```

O comando descrito acima, permite fazer um "*dump*" da imagem para o cartão MicroSD. Ao fazer esta operação, estamos basicamente a permitir que o Raspberry Pi carregue o Sistema Operativo gravado no cartão e após o processo de carregamento, seja apresentada uma *Shell* de comandos.

Nesta fase é necessário a configuração do Sistema Operativo, uma vez que a gravação no cartão, apresenta uma imagem genérica, que é configurada para o dispositivo usado, estado das funcionalidades do dispositivo e expansão do

Sistema Operativo pelo espaço disponível no cartão MicroSD. Na Fig. 15, podemos ver a aplicação de configuração do Sistema Operativo para o Raspberry Pi:

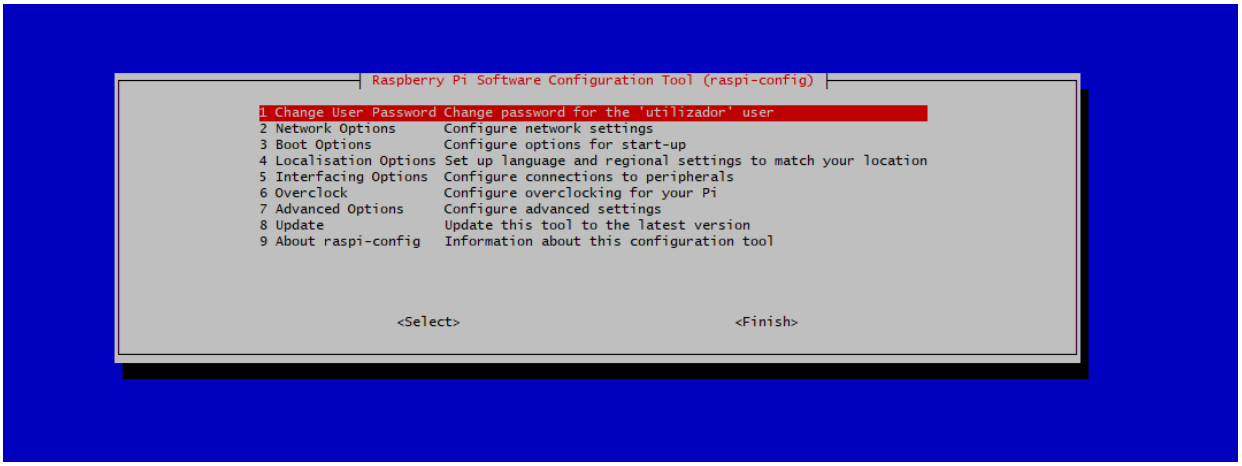

Figura 15 - Aplicação de configuração do Raspberry Pi: raspi-config

De seguida é necessário instalar a linguagem de programação Python versão 3.x, uma vez que a camada aplicacional assenta sobre esta e que pode ser instalada através dos seguintes comandos numa *Shell Linux*:

```
# apt update && apt install python3 python3-pip
```

O comando executado, para além de instalar o *Python* versão 3.x, instala também o gestor de pacotes que é utilizado para instalar os módulos necessários à programação das placas de expansão e sensores.

Todo o desenvolvimento das placas de expansão e sensores, apenas é possível devido à capacidade do Raspberry Pi disponibilizar uma régua GPIO programável. Na Fig. 25, é possível ver a disposição lógica dos pinos na régua GPIO de acordo com o fabricante[42].

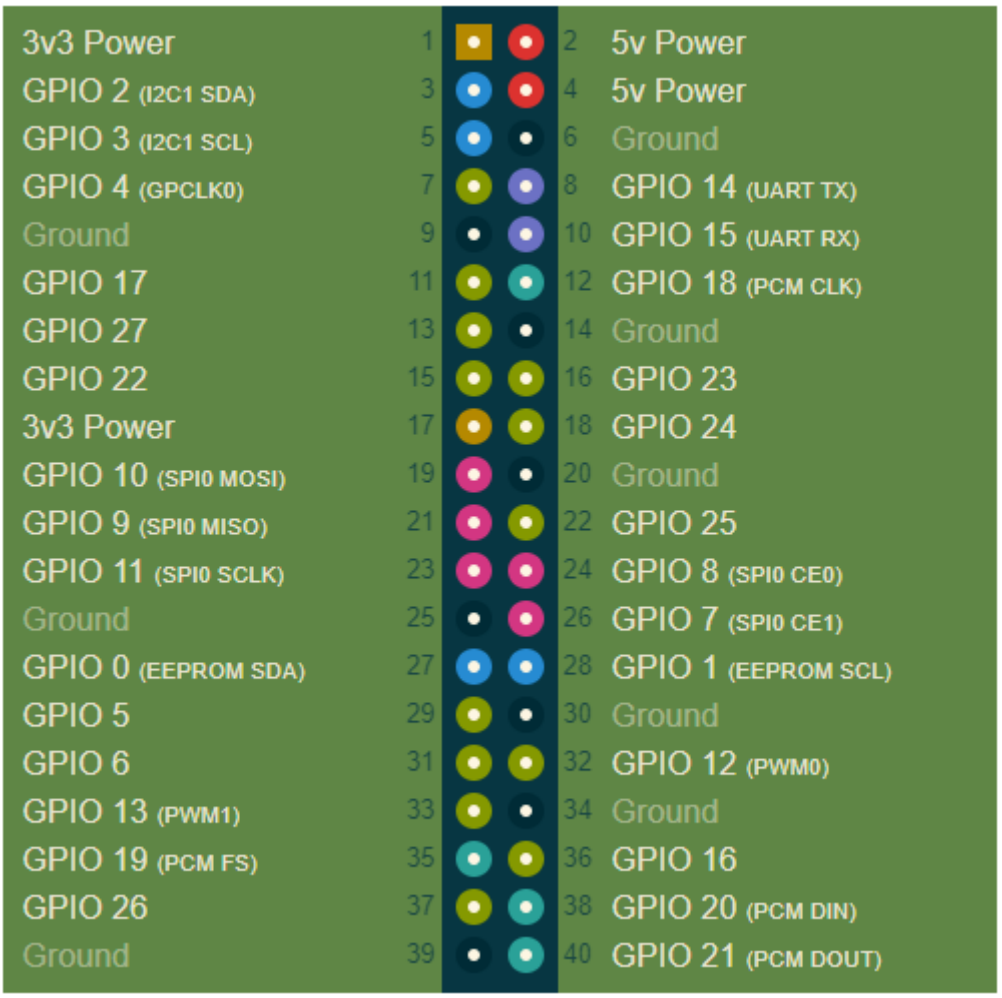

Figura 16 - Disposição geral dos pinos, retirado de [52]

5.2 Sensor de Temperatura e Humidade DHT11

O sensor de condições ambientais DHT11 é um sensor compacto que se liga ao Raspberry Pi através de 3 pinos e que se mostra na Tab. II abaixo. Esta tabela refere-se à ligação elétrica relativamente ao Raspberry PI.

Pino	Descrição
4	Alimentação de 5V
6	Negativo
7	Pino de dados

Tabela II – Pinos utilizados na comunicação com o sensor DHT11

Existem dois tipos de sensores DHT11, em que um deles necessita de uma resistência adicional de 5K e um com circuito integrado onde esta resistência já está incluída. Na Fig. 17 pode-se observar do lado esquerdo a versão simples sem resistência e do lado direito a versão em circuito integrado com a resistência incluída.

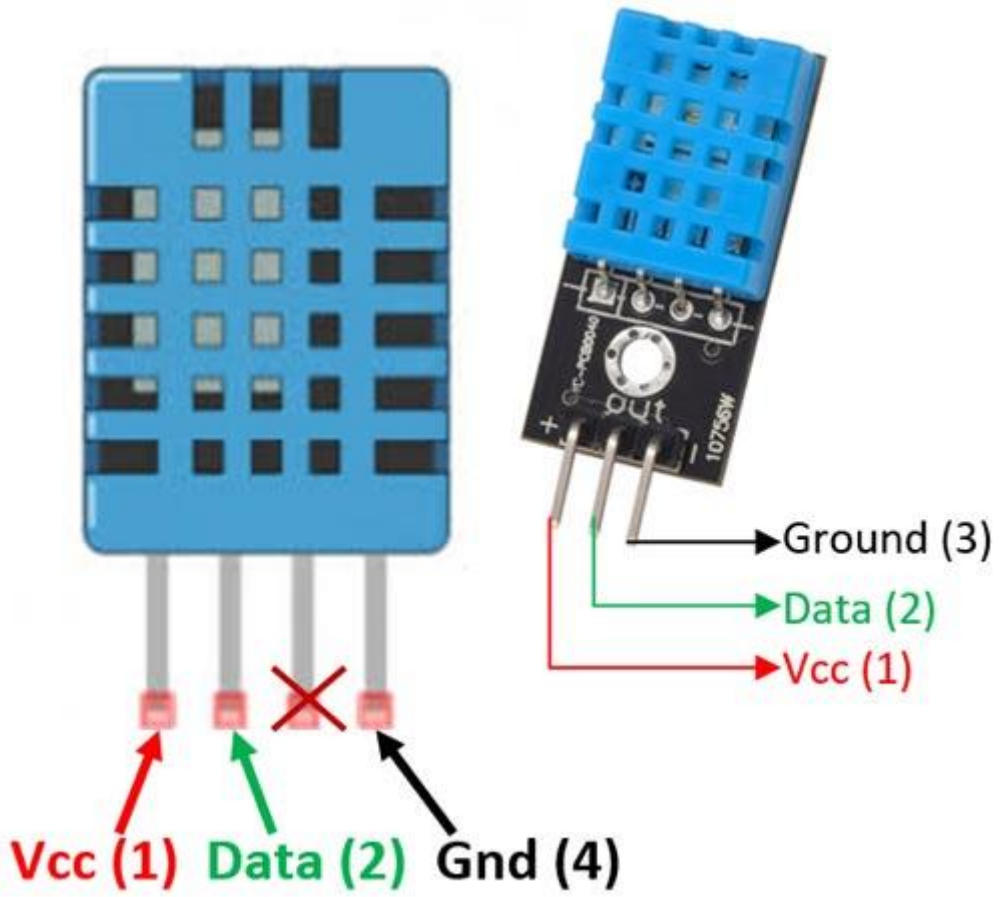

Figura 17- Esquema elétrico para o Sensor DHT11, retirado de [53]

O sensor é calibrado de fabrica[43] e os dados são lidos em ligação série. Na Fig. 18 é apresentado um diagrama geral para o sensor DHT11 usado no projeto.

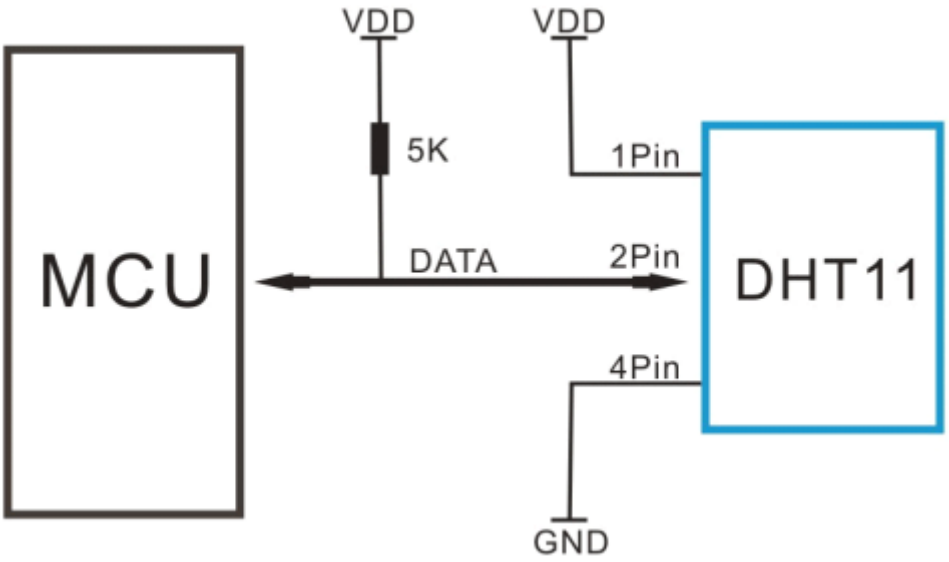

Figura 18 - Diagrama do sensor DHT11, retirado de [53]

Para a utilização do sensor através de código *Python*, é necessário a instalação do módulo para comunicação com este. O módulo é instalado através da descarga de um projeto a partir de um repositório *GIT*. Em baixo deixa-se os comandos para descarregar e instalar o módulo:

```
# git clone
   https://github.com/adafruit/Adafruit_Python_DHT.git
#  sudo python3 setup.py install
```

Com os passos indicados atrás, é possível então programar o sensor através da linguagem de programação *Python.*

Na Figura 19 fica o bloco de código responsável pela leitura de condições ambientais do sensor DHT11. O código usa um módulo do fabricante que é instanciado e é chamado um método para ler os valores para duas variáveis locais. Estes valores são formatados para decimais e apresentados na consola através de um comando de saída.

```
import sys
import Adafruit_DHT

while True:
    humidity, temperature = Adafruit_DHT.read_retry(11, 4)
    print(f'Temp: {temperature:0.1f} Humidity: {humidity:0.1f}')
```

Figura 19 - Código para leitura do sensor DHT11

5.2.1 Valores obtidos

Através do código transcrito anteriormente para a utilização do sensor DHT11, verificaram-se valores de temperatura e humidade expectáveis para o ambiente controlado característico de uma sala de *DataCenter*. Na Fig. 20 em baixo é possível verificar a saída dos valores obtidos do sensor de condições ambientais de forma isolada e posteriormente formatados para uma leitura facilitada.

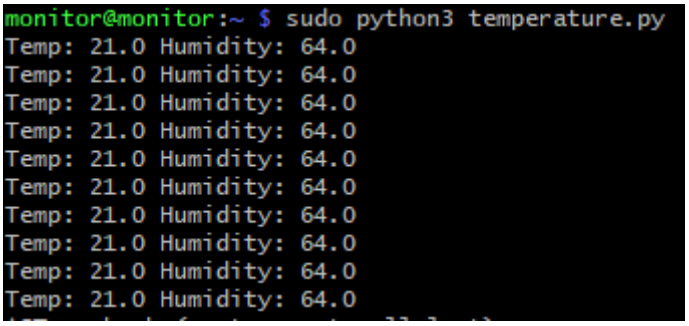

Figura 20 - Leituras do sensor de controlo ambiental

5.2.2 Conclusão

A utilização do sensor DHT11 permite ao Município ter um controlo em tempo real das condições ambientais dentro do *DataCenter*, conseguindo antecipar problemas que possam surgir com aquecimento excessivo dentro do mesmo.

5.3 Placa de expansão GSM/GPRS Sim800

A placa de expansão GSM/GPRS SIM800 permite múltiplas funcionalidades, sendo que para este projeto, é utilizada a capacidade de ligar à rede móvel de dados e envio de mensagens de serviço, vulgo SMS (Short Message Service). A placa de expansão liga-se diretamente sobre a régua de pinos do Raspberry Pi em modo cascata e permite de acordo com o fabricante[44], a ligação de outras placas de expansão em cima desta última, tendo apenas a atenção de os pinos a utilizar, serem sempre diferentes entre as placas de expansão. Na Fig. 21 mostra-se como a placa de expansão é empilhada na régua do Raspberry Pi.

Figura 21 - Placa de expansão GSM/GRPS Sim800 sobre o Raspberry Pi

Na Tabela III seguinte descreve-se os pinos utilizados pela placa.

Pino	Descrição
3	SDA
5	SCL
8	GSM_DIN
10	GSM_DOUT
11	SIM800_POWERKEY
12	SIM800_RESET
19	SPI_MOSI
21	SPI_MISO
23	SPI_SCK
24	SPI_CE0
26	SPI_CE1

Tabela III – Pinos usados para comunicar com a placa de expansão SIM800

A placa de expansão GSM/GPRS Sim800 permite a funcionalidade de ligar, desligar e *reset* através de software. A comunicação com a placa é feita através da porta de série e comandos AT (ATtention). Em baixo transcreve-se os comandos necessários à instalação dos módulos necessários para funcionar com a placa de expansão.

```
# sudo pip3 install usim800 RPi.GPIO
```

Nos blocos seguintes mostra-se algum do código utilizado para comunicar com a placa de expansão.

Na Figura 22 é apresentado um excerto de código Python para ligar o modem. Este pequeno bloco utiliza o módulo de comunicação GPIO (*General Purpose Input/Output*) e define como constantes, o pino 11 para energia e pino 12 para fazer um reset ao modem. De seguida, é definido o modo de operação do pino de energia para saída e a tensão é controlada pelo código para ligar o modem.

```
import RPi.GPIO as GPIO
import time

P_POWER = 11 # Pino de energia
P_RESET = 12 # Pino de reset

GPIO.setmode(GPIO.BOARD)
GPIO.setup(P_POWER, GPIO.OUT)
GPIO.output(P_POWER, GPIO.LOW)
time.sleep(0.5)
GPIO.output(P_POWER, GPIO.HIGH)
time.sleep(3)
GPIO.output(P_POWER, GPIO.LOW)
print("Modem ligado")
```

Figura 22 - Código para ligar o modem

Na Figura 23 é apresentado um pequeno código Python para verificar o estado do modem. É definido um objecto que representa a comunicação de série ao modem, através da serialização de um dispositivo que o representa ao nível do sistema operativo, definindo também a capacidade de comunicação de bits por segundo.
O estado do modem é verificado através do envio de comandos AT (*ATtension*) e verificação do retorno.

```
import serial
import os, time

port = serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=1)

# Enviar comandos AT para o modem
# '\r\n' Indica um "Enter"

port.write(str.encode("AT" + "\r\n"))
rcv = port.read(10)
time.sleep(1)
print(rcv)
```

Figura 23 - Código para verificar o estado do modem

Na Figura 24 é implementado o bloco que trata de enviar as mensagens SMS (*Short Message Service*). São instanciados módulos de comunicação GPIO (*General Purpose Input Output*), comunicação em série e manipulação de data. Novamente é definido o modo de operação do pino 11 para energia da placa de expansão, é instanciado um objecto que representa o dispositivo ao nível do sistema operativo e são enviados comandos AT (*ATtension*) para eliminação de todas as mensagens, limpeza do *buffer* e envio de uma

mensagem de teste. Para efeitos de teste, o resultado é impresso na linha de comandos.

```
import RPi.GPIO as GPIO
import serial
import time, sys
import datetime

P_BUTTON = 11

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(P_BUTTON, GPIO.IN, GPIO.PUD_UP)

SERIAL_PORT = "/dev/ttyAMA0"  # Raspberry Pi 2
#SERIAL_PORT = "/dev/ttyS0"    # Raspberry Pi 3

ser = serial.Serial(SERIAL_PORT, baudrate = 9600, timeout = 5)
setup()
ser.write("AT+CMGF=1\r") # set to text mode
time.sleep(3)
ser.write('AT+CMGDA="DEL ALL"\r') # delete all SMS
time.sleep(3)
reply = ser.read(ser.inWaiting()) # Clean buf


ser.write('AT+CMGS="911848816"\r')
time.sleep(3)
print("Sending test SMS")
ser.write("Test SMS" + chr(26))
time.sleep(3)
reply = ser.read(ser.inWaiting())
print(reply)
```

Figura 24 - Código para envio de uma mensagem SMS

5.3.1 Valores obtidos

A placa de expansão GSM SIM800 permite o funcionamento como um *modem*, possibilitando o controlo por software. Através do código transcrito anteriormente, é possível ligar e obter o estado do modem, tal como se mostra na Fig. 25.

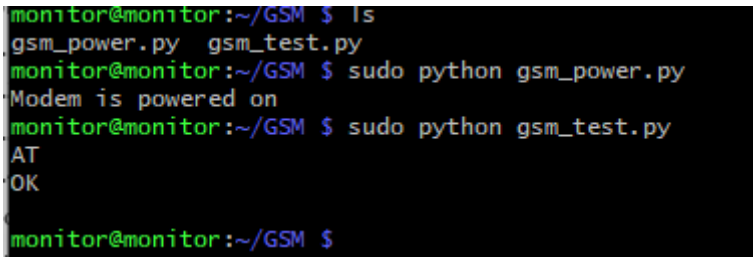

Figura 25 - Estado do modem

O código de envio de SMS (Short Message Service) permite enviar uma mensagem parametrizada para um número pré-determinado. Na Fig. 26 é apresentado o envio de uma mensagem de teste.

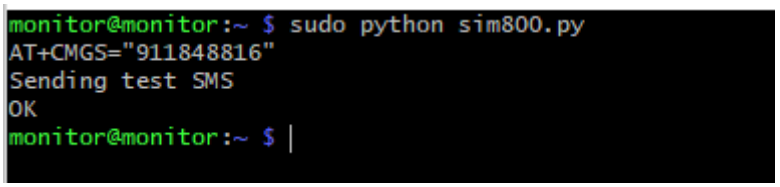

Figura 26 - Envio de um SMS de teste

5.3.2 Conclusão

A utilização da placa de expansão SIM 800 GSM/GPRS neste projecto, permite o envio de alarmística através de SMS (Short Message Service) para o terminal móvel do responsável pela administração do *DataCenter*. Ao combinar os valores de condições ambientais obtidos do sensor DHT11 com a leitura de corrente elétrica através do sensor não-intrusivo SCT-013-000, é possível saber em que condições se encontra o *DataCenter* a nível ambiental e carga elétrica.

5.4 Sensor não-intrusivo de 100A SCT-013-000

O sensor não-intrusivo de 100A SCT-013-000 funciona por indução magnética que excita uma bobina no seu interior e gera uma pequena corrente que pode ser medida no Raspberry Pi através da conversão de sinal analógico para digital. O sensor SCT tem um funcionamento analógico pelo que é necessário converter o sinal para digital. Na Fig. 27 é possível observar o sensor SCT-013-000.

Figura 27 - Sensor SCT-013-000 de 100A

Para isso, utilizou-se um conversor ADC MCP3008 que necessita de um circuito dedicado ligado entre o sensor SCT e o Raspberry Pi. Em baixo na Fig. 28 mostra-se um esquema dos pinos do conversor.

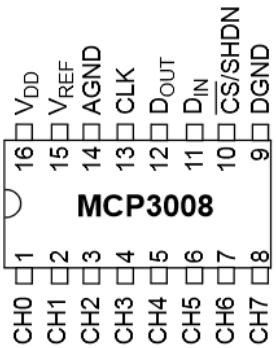

Figura 28 - Conversor ADC MCP3008, retirado de [55]

Sendo que 100V gera uma corrente com uma voltagem acima do que o Raspberry Pi pode receber, é necessário introduzir uma resistência de 10K Ohm. Em baixo na Fig. 29 mostra-se um circuito de teste montado ainda numa placa de testes (*breadboard*), que integra o Raspberry Pi, a resistência de 10K Ohm e o sensor SCT-013-000.

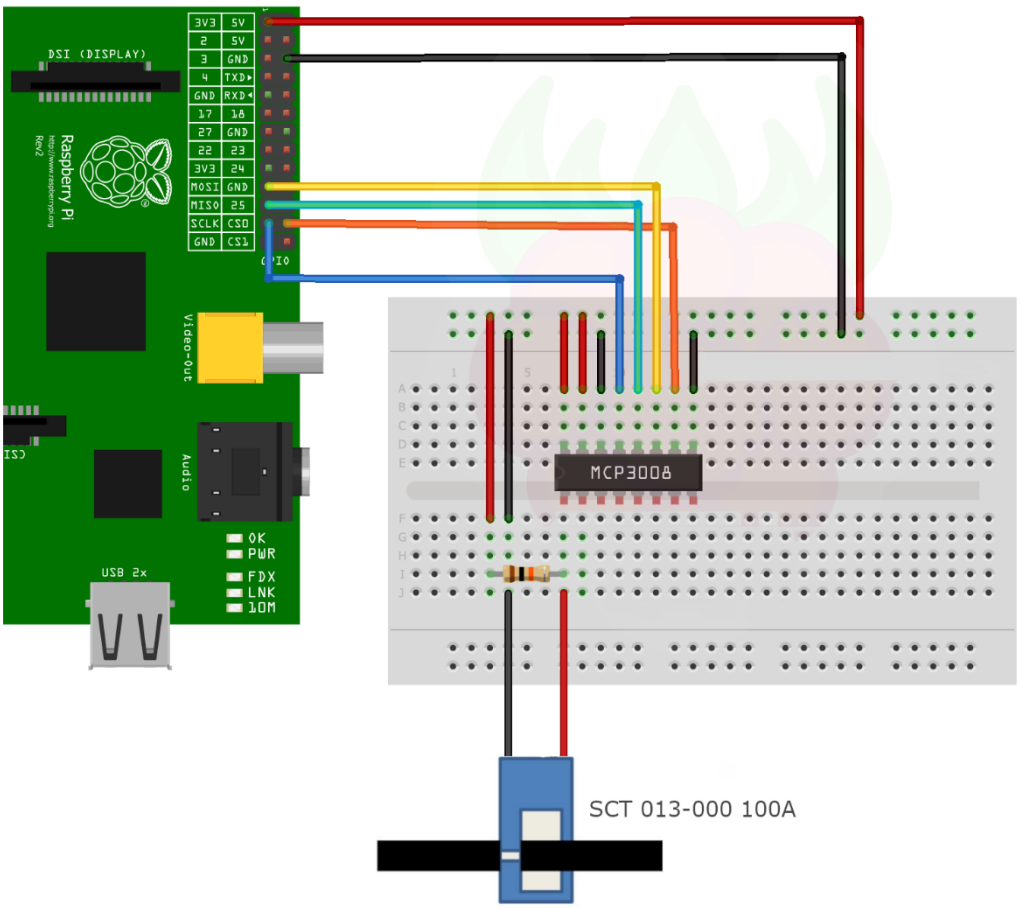

Figura 29 - Esquema para conversão de sinal analógico para digital do sensor SCT-013-000 através do ACD MCP3006

A utilização do sensor SCT 013-000 neste projeto tem como finalidade saber se existe corrente elétrica na cablagem que alimenta o *DataCenter*. Para isso, o código utilizado no Raspberry Pi foi programado apenas com a finalidade de medir numa determinada altura, um valor de tensão entre 0 e 3.3v. Esse valor pode ser calculado pelo valor digital obtido do conversor MCP3008 e normalizado com a seguinte fórmula:

Tensão = (valor_mcp3008 * 3,3) / float(1023)

Na Figura 30 é apresentado um código em Python para testar e obter valores do sensor SCT-013-000. O código usa os módulos do fabricante para aceder

ao sensor SCT-013-000 e ao conversor de sinal ADC MCP3008. No objecto que representa o conversor de sinal, são definidos os pinos para sincronização de relógio SPI (*Serial Peripheral Interface*), dados de saída, dados de entrada e selecção de chip, na ordem de cima para baixo definidos no código. É instanciado um objecto que representa o conversor de sinal e são obtidos em ciclo, os valores do sensor SCT-013-000. É admitido um valor como ligado se a tensão for superior a o.1 e desligado se menor ou igual a 0.1.

```
import Adafruit_GPIO.SPI as SPI
import Adafruit_MCP3008

# Software SPI configuration:
CLK  = 23 #18
MISO = 21 #23
MOSI = 19 #24
CS   = 24 #25
mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

def ConvertVolts():
    volts = (mcp.read_adc(0)*3.3)/float(1023)
    volts = round(volts,9)
    return volts

while True:
    sensor1_volts = ConvertVolts()
    Status = ''
    if sensor1_volts > 0.10:
        Status = 'ON'
    elif sensor1_volts <= 0.1:
        Status = 'OFF'
    print(Status)
    time.sleep(0.5)
```

Figura 30 - Código para obter leituras do sensor SCT-013-000

5.4.1 Valores obtidos

Com o código transcrito anteriormente para a utilização do sensor não-intrusivo SCT-013-000, é possível saber se o *DataCenter* tem alimentação elétrica. Os valores esperados são apenas *ON* e *OFF* mediante o valor de tensão medido no conversor analógico para digital MCP3008.

5.4.2 Conclusão

Através da utilização de um sensor SCT-013-000 filtrado com um conversor de analógico para digital MCP3008 e uma fórmula para normalizar o intervalo de valores obtidos, é possível verificar se o *DataCenter* está a ser alimentado pelo fornecedor de energia ou pela unidade de apoio elétrico.

5.5 Código para o funcionamento geral da solução

Todo o código para funcionamento com a solução foi desenvolvido com *Python* 3, juntamente com a integração de bibliotecas de terceiros para atingir a finalidade pretendida. No código que se segue, as funcionalidades de medição de temperatura, humidade e corrente despoletam alarmes por SMS (Short Message Service) caso surjam leituras fora de intervalos de valores considerados ideais, assim como o registo dos valores dos sensores na base de dados através da API (*Application Programming Interface*).

Na Figura 31 é apresentado todo o código usado no projecto, que integra os exemplos anteriores, para obtenção de valores dos sensores, funções para operar a placa de expansão GSM/GPRS SIM800 através do envio de comandos AT (*ATtension*), sendo elas, fazer um *reset* ao *modem*, ligar e desligar por software o *modem*, verificar o estado do *modem*, estabelecer comunicação GSM (*Global System for Mobile Communications*), estabelecer

comunicação TCP (*Transmission Control Protocol*), envio de um pedido HTTP (*Hypertext Transfer Protocol*), terminar a comunicação TCP, verificar o estado da *internet* e envio de mensagens SMS (*Short Message Service*). O bloco final de código trata da monitorização ao minuto, através da leitura dos valores dos sensores, o envio de alarmes através de SMS e envio dos valores para o *endpoint* da API através do protocolo HTTP.

```
import RPi.GPIO as GPIO
import serial
import time, sys

import Adafruit_DHT
import Adafruit_GPIO.SPI as SP
import Adafruit_MCP3008

DEVICE_ID = "2c408ae0-ff00-4a3e-9217-a465fe5111ba"
DEVICE_LOCALE = "CMA DATACENTER"



# ADC MCP3008
CLK  = 23 #18
MISO = 21 #23
MOSI = 19 #24
CS   = 24 #25
mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)

# SIM800 GSM/GPRS Modem
VERBOSE = False
P_POWER = 11 # Power pin
P_RESET = 12 # Reset pin
APN = "net2.vodafone.pt"
PORT = 5000
```

```
# RASPBERRY PI DEVICE
SERIAL_PORT = "/dev/ttyAMA0"  # Raspberry Pi 2
#SERIAL_PORT = "/dev/ttyS0"    # Raspberry Pi 3
ser = serial.Serial(SERIAL_PORT, baudrate = 9600, timeout = 5)

# FLAGS
ARMED_TEMP_ALERT = True
ARMED_HUM_ALERT = True
ARMED_POWER_ALERT = True

# ENDPOINT
HOST = "monitor.cm-abrantes.pt"
PORT = 443

def debug(text):
    if VERBOSE:
        print("Debug:---", text)

def resetModem():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(P_RESET, GPIO.OUT)
    GPIO.output(P_RESET, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(P_RESET, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(P_RESET, GPIO.LOW)
    time.sleep(3)

def togglePower():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(P_POWER, GPIO.OUT)
    GPIO.output(P_POWER, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(P_POWER, GPIO.HIGH)
```

```
    time.sleep(3)
    GPIO.output(P_POWER, GPIO.LOW)

def isReady(ser):
    # Resetting to defaults
    cmd = 'ATZ\r'
    debug("Cmd: " + cmd)
    ser.write(str.encode(cmd))
    time.sleep(2)
    reply = ser.read(ser.inWaiting())
    time.sleep(8) # Wait until connected to net
    return (reply)

def connectGSM(ser, apn):
    # Login to APN, no userid/password needed
    cmd = 'AT+CSTT="' + apn + '"\r'
    debug("Cmd: " + cmd)
    ser.write(str.encode(cmd))
    time.sleep(3)

    # Bringing up network
    cmd = "AT+CIICR\r"
    debug("Cmd: " + cmd)
    ser.write(str.encode(cmd))
    time.sleep(5)

    # Getting IP address
    cmd = "AT+CIFSR\r"
    debug("Cmd: " + cmd)
    ser.write(str.encode(cmd))
    time.sleep(3)

    # Returning all messages from modem
    reply = ser.read(ser.inWaiting())
```

```
    debug(reply)
    return reply

def connectTCP(ser, host, port):
    cmd = 'AT+CIPSTART="TCP","' + host + '","' + str(port) + '"\r'
    ser.write(str.encode(cmd))
    time.sleep(5)
    reply = ser.read(ser.inWaiting())
    debug(reply)
    return reply

def sendHTTPRequest(ser, host, request):
    ser.write(str.encode("AT+CIPSEND\r"))
    time.sleep(2)
    request = "GET " + request + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n"
    ser.write(str.encode(request + chr(26)))  # data<^Z>
    time.sleep(2)

def closeTCP(ser, showResponse = False):
    ser.write(str.encode("AT+CIPCLOSE=1\r"))
    reply = ser.read(ser.inWaiting())
    debug("closeTCP() retured:\n" + reply)
    if showResponse:
        print("Server reponse:\n" + reply[(reply.index("SEND OK") + 9):])
    time.sleep(2)

def getIPStatus(ser):
    cmd = "AT+CIPSTATUS\n"
    ser.write(str.encode(cmd))
    time.sleep(1)
    reply = ser.read(ser.inWaiting())
    return reply

def sendSMSMessage(ser, msg):
```

```
    setup()
    ser.write(str.encode("AT+CMGF=1\r")) # set to text mode
    time.sleep(3)
    ser.write(str.encode('AT+CMGDA="DEL ALL"\r')) # delete all SMS
    time.sleep(3)
    reply = ser.read(ser.inWaiting()) # Clean buf
    ser.write(str.encode('AT+CMGS="+351111111111"\r')) # send SMS message
    time.sleep(3)
    ser.write(str.encode("Alert: " + msg + chr(26)))
    time.sleep(3)

def ConvertVolts():
    volts = (mcp.read_adc(0)*3.3)/float(1023)
    volts = round(volts,9)
    return volts

# Power on modem and reset to ready state
togglePower()
resetModem()

# Test if modem is ready
if not isReady(ser):
        print("Modem not ready.")
        sys.exit(0)

# Connecto to GSM
connectGSM(ser, APN)

while True:

    # Get temperature and humidity
    humidity, temperature = Adafruit_DHT.read_retry(11, 4)

    # Test if temperature is above permitted
```

```
   if ({temperature:0.1f} > 23 and ARMED_TEMP_ALERT):

      ARMED_TEMP_ALERT = False
      sendSMSMessage("Temperature above permitted 23º")

   # Test if humidity is above permitted
   if ({humidity:0.1f} > 85 and ARMED_HUM_ALERT)

      ARMED_HUM_ALERT = False
      sendSMSMessage("Humidity above permitted 85%")

   # Test if power cable is energized
   if (ConvertVolts() <= 0.1 and ARMED_POWER_ALERT)

      ARMED_POWER_ALERT = False
      sendSMSMessage("No power")



   # Send data to server
   data                                                                    =
{"deviceid":DEVICE_ID,"locale":DEVICE_LOCALE,"temperature":{temperature:0.1f}
,"humidity":{humidity:0.1f},"current": ConvertVolts ()}
   ser.write('AT+HTTPPARA="URL","http://monitor.cm-abrantes.pt/readvalues.php"')
   ser.write('AT+HTTPPARA="CONTENT","application/json"')
   ser.write('AT+HTTPDATA=1500,5000'+'\r\n')
   ser.write(json.dumps(data))
   ser.write('AT+HTTPACTION=1' + '\r\n')

   # Make pause for 1 minute
   sleep(60)
```

Figura 31- Código para funcionamento geral do projecto

5.6 Registo de leituras dos sensores

Para registar os dados da leitura dos sensores, foi criado uma pequena API que recebe os dados e grava em base de dados. A API foi desenvolvida em PHP assente sobre um servidor Web NGINX[45]. Na Fig. 32 é apresentado o código para registar os valores na base de dados. Este começa por verificar qual o método CRUD (*Create, Read, Update and Delete*) usado no envio de dados, que neste caso, só interessa o método POST. Os dados são enviados como um objecto JSON (*Javascript Simple Object Notation*) que têm de ser convertidos para um objecto PHP por forma a aceder aos valores. É instanciado um objecto PDO (*PHP Data Object*) para executar a operação de escrita na base de dados. Para efeitos de teste, é impresso no terminal o objecto recebido pelo pedido HTTP.

```
<?php
    require_once '../includes/database.php';

    $results = null;
    $method = $_SERVER['REQUEST_METHOD'];

    switch ($method) {
        case 'POST':
            parse_str(file_get_contents('php://input'), $input);

            $pdo = Database::connect();
            $pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            $sql = "INSERT INTO sensor_reads
(device_uuid,device_locale,stamp,temperature,humidity,power) values(?, ?, ?, ?, ?,
?)";
            $q = $pdo->prepare($sql);
            $q-
>execute(array($input['device_uuid'],$input['device_locale'],$input['stamp'],$input
['temperature'],$input['humidity'],$input['power']));
            Database::disconnect();
            header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
            header('Content-type: application/json');
            echo json_encode(array('status' => '201', 'payload' => $input));
            break;
        default:
            header("Access-Control-Allow-Origin: {$_SERVER['HTTP_ORIGIN']}");
            header("Access-Control-Allow-Methods: OPTIONS, HEAD, GET, POST,
PUT, DELETE ");
            header('Content-type: application/json');
            echo json_encode(array('status' => '404', 'payload' => ''));
            break;
    }
?>
```

Figura 32 - Código da API (Application Programming Interface)

5.7 Camada de Base de Dados

Os dados são armazenados numa base de dados MariaDB, com a estrutura que se define na Tab. IV.

Campo	Tipo	Descrição
id	inteiro auto incrementado	Chave primária
device_uuid	varchar(36)	UUID Versão 4
device_locale	varchar(200)	Localização do dispositivo
stamp	datetime	Carimbo de tempo
temperature	float	Temperatura
humidity	float	Humidade
power	float	Corrente electrica

Tabela IV – Tabela para armazenar os valores na base de dados

Estrutura em linguagem DDL:

```
CREATE TABLE monitor.sensor_reads (
 id INT auto_increment NULL,
 device_uuid varchar(36) NOT NULL,
 device_locale varchar(200) NULL,
 stamp DATETIME NOT NULL,
 temperature FLOAT NULL,
```

```
 humidity FLOAT NULL,
 power FLOAT NULL,
 CONSTRAINT sensor_reads_pk PRIMARY KEY (id)
)
ENGINE=InnoDB
```

5.8 Camada de Analítica

O dispositivo tem a capacidade de enviar dados da leitura dos sensores a cada minuto, para uma API para posterior processamento. Visualizar os dados diretamente na tabela da base de dados, além de não ser prático, pode tornar-se numa tarefa complexa pela análise da quantidade de valores registados, até se encontrar a informação pretendida. Para complementar a solução, optou-se por integrar uma ferramenta de analítica, neste caso o *IBM Cognos Analytics*, ferramenta que tem sido usada já noutras áreas, para análise de dados e suporte à decisão do Executivo. Em baixo, na Fig. 33, mostra-se um relatório gerado a partir desta ferramenta.

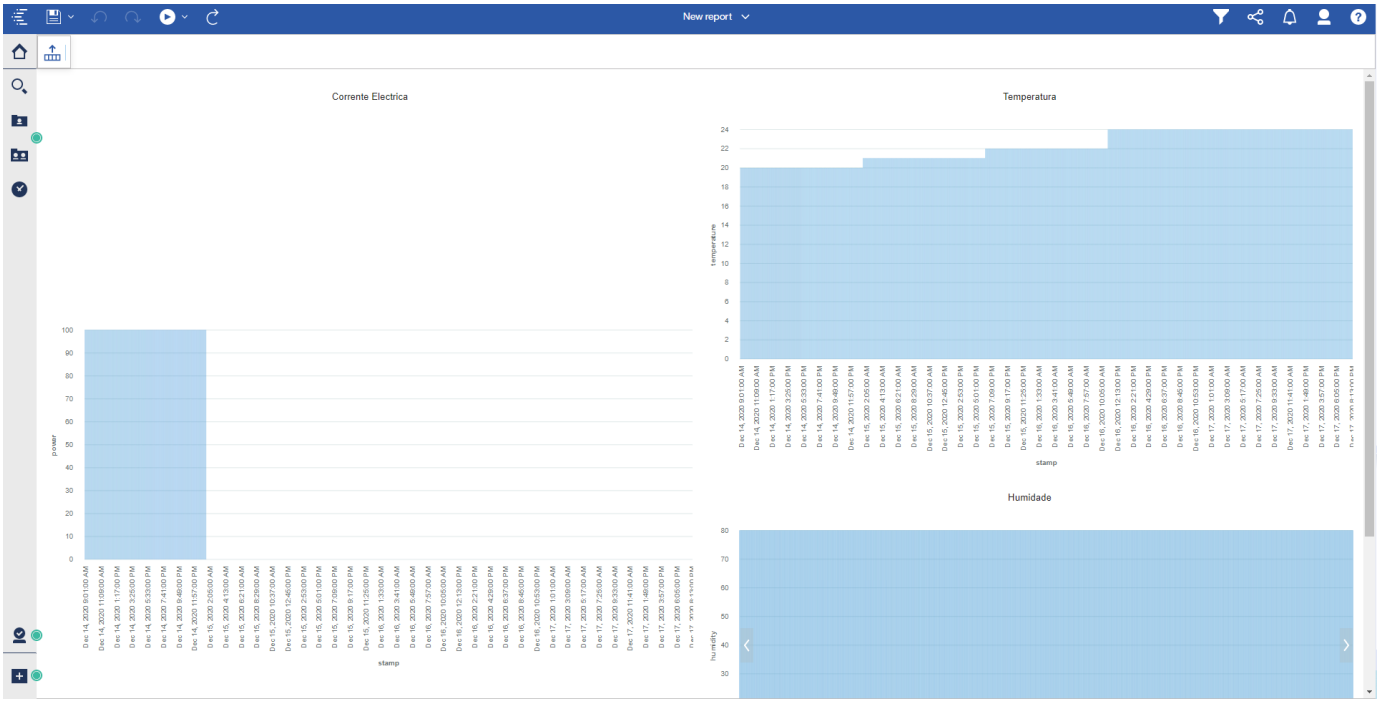

Figura 33 - Relatório gerado no IBM Cognos Analytics

5.9 Análise de uma amostra de dados

Todos os dados gerados pelo dispositivo são registados em base de dados para utilização em analítica. Em baixo mostra-se um conjunto de valores obtidos num determinado espaço temporal, onde houve um corte deliberado da alimentação elétrica por via a simular uma falha real. O expectável neste cenário é encontrar valores a zero para o consumo elétrico no campo da tabela de registo de leituras. De igual modo é também enviado um alarme através de um SMS (Short Message Service) para um terminal móvel pré-determinado.

device_uuid	device_locale	timestamp	temperature	humidity	power
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:12	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:13	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:14	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:15	21	64	3.2

2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:16	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:17	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:18	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:19	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:20	21	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:21	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:23	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:24	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:25	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:26	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:27	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:28	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:29	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:30	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:31	22	64	3.3
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:32	22	64	3.3
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:33	22	64	3.3
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:34	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:35	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:36	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:37	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:38	22	64	3.1
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:39	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:40	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:41	22	64	3.2

2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:42	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:43	22	64	3.2
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:44	22	64	0
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:45	22	64	0
2c408ae0-ff00-4a3e-9217-a465fe5111ba	CMA DATACENTER	9/10/2020 10:46	22	64	0

Nos últimos três registos podemos observar que a tensão do consumo elétrico baixou para 0.0, resultado da bobina do sensor SCT-013-000 já não se encontrar energizada pela falta da passagem de corrente.

Na Figura 34 podemos observar um *screenshot* do *smartphone* que está registado para receber as mensagens de alarmística.

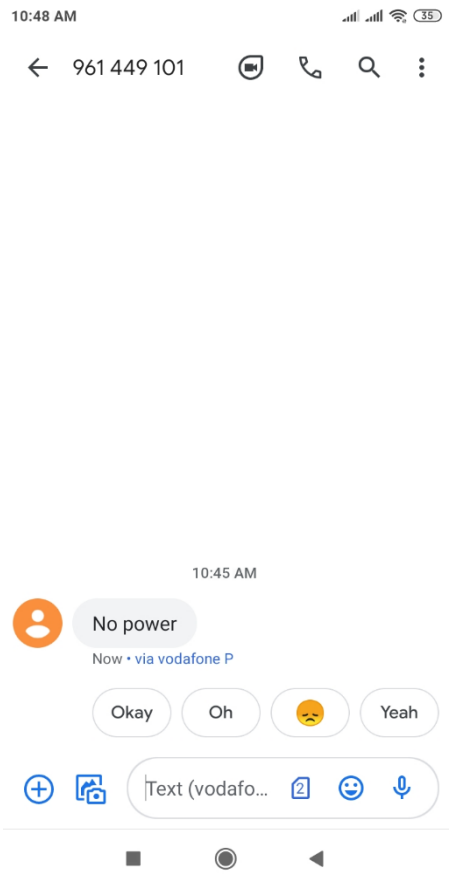

Figura 34- Screenshot do smartphone com SMS da alarmística

6 Conclusões

Este projeto teve como finalidade criar um dispositivo IoT em permanente monitorização de um componente crítico que é o *DataCenter*. Se bem que o *DataCenter* está atualmente protegido de picos e quebras de corrente elétrica, a monitorização da temperatura dentro da sala não era feita de modo autónomo.

A monitorização da humidade dentro do *DataCenter*, tem como vantagem adicional controlar a presença de pessoas no espaço, uma vez que o ambiente é refrigerado por unidades de ar condicionado e é tipicamente seco ou de baixa humidade, pelo que o sensor DHT11 pela sua sensibilidade de relativa alta precisão, consegue detetar uma subida da humidade mesmo que ligeira.

Ao nível da analítica, a integração com o *IBM Cognos Analytics* trouxe um valor acrescentado à solução, uma vez que permite uma análise visual temporal de incidências e duração das mesmas, sendo uma ferramenta importante no suporte à decisão quanto ao dimensionamento das Unidades de Energia de Backup e enquadramento das mesmas, assim como das unidades de ar condicionado.

Em relação a outros sensores e equipamentos apresentados no capítulo do estado da arte, podemos fazer um comparativo das vantagens e desvantagens para cada um.

Vantagens relativamente a Advantech Smart IoT

- Leitura de condições ambientais e elétricas num único dispositivo
- A comunicação de alarmística é feita por GSM/GPRS
- Integração com a API do cliente através da parametrização dos dados a comunicar

- Plataforma aberta

Desvantagens relativamente a Advantech Smart IoT

- Modularidade e leituras isoladas podem ser vantajosas em algumas aplicações
- Dispositivos mais simples e especializados

Vantagens relativamente a AKCP

- Comunicação de alarmística por GSM/GPRS
- Integração com a API do cliente através da parametrização dos dados a comunicar
- Plataforma aberta

Desvantagens relativamente a AKCP

- Capacidade de detetar fugas de água e diferenciais de pressão de ar para além das condições climáticas
- Interface apelativa e modo de simulação de circulação de ar nos bastidores de servidores
- Controlo térmico orientado para instalação nos bastidores de servidores

Vantagens relativamente a Xtralis Vesda-E

- Controlo ambiental e elétrico
- Comunicação de alarmística por GSM/GPRS
- Integração com a API do cliente através da parametrização dos dados a comunicar
- Plataforma aberta

Desvantagens relativamente a Xtralis Vesda-E

- Prevenção de incêndios

Vantagens relativamente a SensorPush

- Para além do controlo ambiental permite o controlo elétrico
- Orientado para qualquer tipo de mercado (doméstico ou empresarial)
- Comunicação de alarmística por GSM/GPRS
- Integração com a API do cliente através da parametrização dos dados a comunicar
- A utilização de uma *Cloud* proprietária pode ser uma desvantagem para clientes mais especializados
- Plataforma aberta

Desvantagens relativamente a SensorPush

- A orientação para o mercado doméstico torna o dispositivo mais acessível para utilizadores comuns
- Aplicação para *Smartphone* e ligação Bluetooth aos dispositivos mostra ser uma proposta apelativa para utilizadores domésticos
- O acesso aos dispositivos através da Cloud proprietária pode ser uma solução amigável para utilizadores domésticos

Vantagens relativamente a Fludia

- Controlo ambiental e elétrico num único dispositivo
- Comunicação de alarmística por GSM/GPRS
- Integração com a API do cliente através da parametrização dos dados a comunicar
- Plataforma aberta

Desvantagens relativamente a Fludia

- Controlo preciso de consumo elétrico pela colocação do dispositivo directamente no contador
- Comunicação de dados por LoRaWAN

A introdução deste dispositivo veio permitir um controlo sobre fatores

como temperatura e humidade da sala e quebras prolongadas de eletricidade, que ao integrar com uma placa de expansão GSM/GPRS, permite o aviso de forma eficaz para o administrador de sistemas, através de uma simples mensagem SMS (*Short Message Service*).

Na sequência da Tabela I, comparativa de dispositivos no capítulo do Estado da Arte, compara-se agora com o dispositivo IoT (*Internet of Things*) desenvolvido neste projeto na Tabela V.

	Dispositivo de Monitorização	*Advantech Smart IoT*	*AKCP*	*Xtralis Vesda-E*	*SensorPush*	*Fludia*
Controlo Temperatura	Sim	Sim	Sim	Não	Sim	Não
Controlo Humidade	Sim	Não	Sim	Não	Sim	Não
Controlo Eléctrico	Sim	Sim	Não	Não	Não	Sim
Comunicação GSM/GPRS/WiFi	Sim	Sim – Wifi com edge router	Não	Sim	Sim – Wifi com gateway	Sim - LoRaWAN 1.0
Analítica / API / Web Interface	Sim – API e Analítica	Sim - API	Sim – Web Interface embebido	Sim – Web Interface embebido	Sim - Portal	Sim – Ficheiros CSV
Outros sensores	Não	Não	Fugas de Água e diferencial de pressão de ar	Prevenção de incêndios por análise de partículas no ar	Não	Leitura ótica

Tabela V – Comparativo entre o dispositivo de monitorização desenvolvido e outros dispositivos

7 Bibliografia

[1] -. Grafana – The Open Observability Platform. Retrieved September 14, 2020. (https://grafana.com/)

[2] -. Radovan Richta "*Mankind In Transition; A View of the Distant Past, the Present and the Far Future*", Masefield Books, 1993.

[3] -. "*The Computer History Museum, SRI International, and BBN Celebrate the 40th Anniversary of First ARPANET Transmission, Precursor to Today's Internet*". SRI International. October 27, 2009. Archived from the original on March 29, 2019. Retrieved September 25, 2017. (https://web.archive.org/web/20190329134941/https://www.sri.com/newsroom/press-releases/computer-history-museum-sri-international-and-bbn-celebrate-40th-anniversary)

[4] -. by Vinton Cerf, as told to Bernard Aboba (1993). "*How the Internet Came to Be*". Retrieved September 25, 2017. (http://elk.informatik.hs-augsburg.de/tmp/cdrom-oss/CerfHowInternetCame2B.html)

[5] -. Hauben, Ronda (May 1, 2004). "*The Internet: On its International Origins and Collaborative Vision A Work In-Progress*". Retrieved September 25, 2017. (http://www.columbia.edu/~rh120/other/misc/haubenpap.rtf)

[6] -. Kim, Byung-Keun (2005). Internationalising the Internet the Co-evolution of Influence and Technology. Edward Elgar. pp. 51–55. ISBN 978-1845426750.

[7] -. "ARPANET, Internet". www.livinginternet.com. Retrieved April 4, 2020.

[8] -. Rouse, Margaret (2019). "internet of things (IoT)". IOT Agenda. Retrieved August 14, 2019. (https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT)

[9] -. Riyadh Arridha *"Design and Implementation of IoT-Big Data Analytic for Smart Environment Monitoring System"*. Retrieved December 14, 2020. (https://thesiscommons.org/6frp3/)

[10] -. ArcGIS Velocity - Analyze real-time and big data. Retrieved December 14, 2020. (https://www.esri.com/en-us/arcgis/products/arcgis-velocity/overview)

[11] -. OMRON Electronic Components – Sensors. Retrieved December 14, 2020.

[12] -. A guide to modern factory automation and Industry 4.0 in manufacturing. Retrieved December 14, 2020. (https://www.information-age.com/guide-modern-factory-automation-

industry-4-0-manufacturing-123490992/#:~:text=Industry%204.0%2C%20also%20known%20as,of%20an%20ever%2Dchanging%20industry.)

[13] -. How AI Is Paving the Way for Autonomous Cars. Retrieved December 14, 2020. (https://www.machinedesign.com/mechanical-motion-systems/article/21838234/how-ai-is-paving-the-way-for-autonomous-cars)

[14] -. Adafruit Industries – DHT11 Sensor. Retrieved December 14, 2020. (https://www.adafruit.com/product/386)

[15] -. Advantech – Enabling an Intelligent Planet. Retrieved December 14, 2020. (https://www.advantech.eu/)

[16] -. Advantech Smart IoT Wzzard. Retrieved December 14, 2020. (https://advantech-bb.com/smart-iot-technology-for-data-centers)

[17] -. AKCP – Remote Sensor Monitoring. Retrieved December 14, 2020. (https://www.akcp.com/)

[18] -. AKCP – AKCP Pro Server. Retrieved December 14, 2020. (https://www.akcp.com/rack-plus)

[19] -. AKCP – AKCP Environmental Sensors. Retrieved December 14, 2020. (https://www.akcp.com/akcp-products/sensors-2/#sensors-2+category:environmental)

[20] -. Xtralis Company. Retrieved December 14, 2020. (https://xtralis.com/)

[21] -. Xtralis VESDA-E. Retrieved December 14, 2020. (https://xtralis.com/product_subcategory/1/VESDA-E-Aspirating-Smoke-Detection)

[22] -. SensorPush Company. Retrieved December 14, 2020. (https://www.sensorpush.com/about)

[23] -. SensorPush WiFi Gateway. Retrieved December 14, 2020. https://www.sensorpush.com/products/p/g1-gateway)

[24] -. SensorPush temperature and humidity sensor. Retrieved December 14, 2020. (https://www.sensorpush.com/products/p/ht1)

[25] -. Fludia Company. Retrieved December 14, 2020. (https://www.fludia.com/-A-propos-de-Fludia-.html?lang=fr)

[26] -. Fludia sensor FM432ir. Retrieved December 14, 2020. ()

[27] -. Fludia LoRaWAN FM432p. Retrieved December 14, 2020. (https://www.fludia.com/-

LORA-192-.html?lang=fr)

[28] -. Raspberry Pi Model B specifications. Retrieved December 14, 2020. (https://www.raspberrypi.org/products/raspberry-pi-2-model-b/)

[29] -. GSM/GPRS SIM800 board. Retrieved December 14, 2020. (https://www.itead.cc/wiki/RPI_SIM800_GSM/GPRS_ADD-ON_V2.0)

[30] -. Adafruit temperature and humidity sensor. Retrieved December 14, 2020. (https://learn.adafruit.com/dht?gclid=CjwKCAiAo5qABhBdEiwAOtGmbggJsark3qtrEOlFdCS2WkMkacCXR5T_TTZsHvo6f3_hiFhYOmW8ChoCetcQAvD_BwE)

[31] -. SCT-013-000 current transformer. Retrieved December 14, 2020. (https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/yhdc-sct-013-000-ct-sensor-report)

[32] -. Python Language. Retrieved December 14, 2020. (https://www.python.org/)

[33] -. PHP Language. Retrieved December 14, 2020. (https://www.php.net/)

[34] -. MySQL Database. Retrieved December 14, 2020. (https://www.mysql.com/)

[35] -. MariaDB Database. Retrieved December 14, 2020. (https://mariadb.org/)

[36] -. GitHub is how people build software. Retrieved 14 September 2020. (https://github.com/about)

[37] -. Microsoft acquires GitHub. Retrieved December 14, 2020. (https://news.microsoft.com/announcement/microsoft-acquires-github/)

[38] -. About GitLab. Retrieved 14 September 2020. (https://about.gitlab.com/company/)

[39] -. Seesparkbox “The Big Three of Git Storage: Comparing GitHub and GitLab”, Catherine Meade. Online: https://seesparkbox.com/foundry/github_vs_gitlab_vs_bitbucket. Retrieved 14 September 2020

[40] -. Raspberry PI programmable GPIO. Retrieved December 14, 2020. (https://www.raspberrypi.org/documentation/usage/gpio/)

[41] -. Raspberry PI Operating Systems. Retrieved December 14, 2020. (https://www.raspberrypi.org/software/operating-systems/)

[42] -. Raspberry PI 2 Model B Pinouts. Retrieved December 14, 2020. (https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md)

[43] -. The DHT11 Sensor. Retrieved December 14, 2020.

(https://components101.com/dht11-temperature-sensor)

[44] -. Itead SIM800 GSM/GPRS Module. Retrieved December 14, 2020. (https://www.itead.cc/wiki/RPI_SIM800_GSM/GPRS_ADD-ON_V2.0)

[45] -. NGINX Web Server. Retrieved December 14, 2020. (https://www.nginx.com/)

[46] -. Brasão do Município de Abrantes. Retrieved December 14, 2020. (http://cm-abrantes.pt/index.php/pt/2014-11-27-18-15-49/2014-11-27-18-17-37/identidade-municipal)

[47] -. Organigrama do Município de Abrantes. Retrieved December 14, 2020 (http://cm-abrantes.pt/images/documentos/municipio/camara-municipal/recursos-humanos/estrutura-organica/Organigrama_2020.pdf)

[48] -. Advantech Wzzard solution. Retrieved December 14, 2020. (https://advantech-bb.com/product-technology/iot-and-network-edge-platforms/industrial-wireless-sensing-solutions/wzzard/)

[49] -. AKCP Pro Server. Retrieved December 14, 2020. (https://www.akcp.com/blog/thermal-maps-in-akcpro-server/)

[50] -. AKCP Sensors. Retrieved December 14, 2020. (https://www.akcp.com/akcp-products/sensors-2/#sensors-2+category:environmental)

[51] -. VESDA-E Solution. Retrieved December 14, 2020. (https://xtralis.com/product_subcategory/1/VESDA-E-Aspirating-Smoke-Detection)

[52] -. Raspberry PI pinouts. Retrieved December 14, 2020. (https://pinout.xyz/)

[53] -. DHT11 electric diagram. Retrieved December 14, 2020. (https://components101.com/dht11-temperature-sensor)

[54] -. SCT013-000 Current Transformer. Retrieved December 14, 2020. (https://www.ptrobotics.com/sensor-de-corrente/2143-non-invasive-ac-current-sensor-100a.html?gclid=EAIaIQobChMI3L7-xPyv7gIVxYTVCh09gACQEAQYBSABEgKYsfD_BwE)

[55] -. MCP3008 electric diagram. Retrieved December 14, 2020. (https://learn.adafruit.com/raspberry-pi-analog-to-digital-converters/mcp3008)