



**Guilherme António
Gomes Vicente
Salgueiro**

**Proposta de um sistema automático de
rastreamento - Renault 4.0**



**Guilherme António
Gomes Vicente
Salgueiro**

Proposta de um sistema automático de rastreamento - Renault 4.0

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizado sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro, e de Armando Pinto, Professor Associado com Agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Este trabalho teve o apoio financeiro dos projetos UIDB/00481/2020 e UIDP/00481/2020 - FCT - Fundação para Ciência e Tecnologia; e CENTRO-01-0145-FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), no âmbito do Acordo de Parceria Portugal 2020, através do Fundo Europeu de Desenvolvimento Regional.

O júri / The jury

Presidente / President

Prof. Doutor Rui Moreira

Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Ricardo Chaves

Professor Associado do Instituto Superior Técnico de Lisboa

Prof. Doutor Armando Pinto

Professor Associado com Agregação da Universidade de Aveiro (coorientador)

Agradecimentos / Acknowledgements

Agradeço aos Prof. Doutor José Paulo Santos e ao Prof. Doutor Armando Nolasco Pinto todas as suas orientações, disponibilidade e apoio ao longo do desenrolar da minha dissertação. Ao Engenheiro Miguel Teixeira da Renault CACIA, agradeço toda a sua disponibilidade, partilha de conhecimento e entejuda que foi fundamental para a minha evolução enquanto engenheiro e pessoa. Pretendo deixar um agradecimento a todos os meus amigos de faculdade pelo espírito de companheirismo ao longo destes cinco anos e que certamente continuará para o resto da vida. Quero ainda deixar um agradecimento a toda a minha família, especialmente ao Urs e a Barbara Holzer sem os quais o meu percurso não teria tido o mesmo desenrolar. Por fim, gostaria de enaltecer as duas mulheres da minha vida, a minha mãe Teresa e a minha namorada Cristiana que sempre estiveram ao meu lado para me apoiar e sem as quais não seria a pessoa que sou hoje.

Palavras-chave

AES, Blockchain, base de dados, conectividade, encryption, ESP, IoT, HMI

Resumo

A crescente necessidade de obter informação sobre os processos de fabrico leva a que as empresas procurem criar uma cadeia de informação segura, em que possam englobar não só os seus fornecedores, como também o comprador. Deste modo a Renault Cacia procura um sistema de baixo custo baseado em hardware, como os ESP e Raspeberry Pi, para proceder à recolha de dados das suas linhas de produção, para a programação dos equipamentos é esperado que esta tenha por base software open source. Procura ainda que a solução proposta possa ser aplicada às mais variadas linhas sempre de forma rápida e simples. Principalmente é pretendido que toda a comunicação seja executada de forma comprovadamente segura. Como tal, nesta dissertação, executam-se testes à segurança da encriptação gerada pelos dispositivos IoT, melhor dizendo, os testes executados analisam a aleatoriedade dos números gerados por estes equipamentos, sobre os quais é processada a encriptação da informação recorrendo a AES CBC. Tendo ainda em conta a crescente expansão de cripto moedas levou a que a sua forma partilhada de armazenamento de informação fosse adaptada a bases de dados empresariais. Contudo, a falta de desenvolvimento em grande escala de cadeias de informação, baseadas em dispositivos IoT, bem como as suas inúmeras falhas relativas à segurança na transmissão de informação, serviram de motivação para o desenvolvimento deste projeto baseado, também ele, em Blockchain. Esta cadeia serve assim para fazer o armazenamento de dados, tendo sempre em vista um leque variado de dados a recolher bem como a implementação física na Renault Cacia.

Keywords

AES, Blockchain, connectivity, database, encryption, ESP, IoT, HMI

Abstract

The growing need to obtain information about manufacturing processes motivates companies to seek and create a secure information chain, in which they can encompass not only their suppliers, but also the buyer. Thus, Renault Cacia is looking for a low-cost system based on hardware, such as the ESP and Raspberry Pi, to collect data from the production lines. For the equipment programming, it is expected that this will be based on open source software. The implementation also seeks to propose a solution that can be applied to the most varied lines, always quickly and simply. It is primarily intended that all communication will be performed in a demonstrably secure manner. As such, in this dissertation, tests are carried out on the security of the encryption generated by the IoT devices, in other words, the tests performed analyze the randomness of the numbers generated by these equipments, on which the encryption of information is processed using AES CBC. Also taking into account the growing expansion of cryptocurrencies, its shared form of information storage was adapted to incorporate databases. However, the lack of large-scale development of information chains, based on IoT devices, as well as their numerous failures related to security in the transmission of information, served as motivation for the development of this project, also based on Blockchain. This chain thus serves to store data, always bearing in mind a wide range of data to be collected as well as the physical implementation at Renault Cacia.

Índice

Lista de Tabelas	v
Lista de Figuras	vii
1 Introdução	1
1.1 Motivação	1
1.2 Objetivos e Resultados esperados	2
1.3 Guia de leitura	3
2 Estado de Arte	5
2.1 Introdução à Renault Cacia	5
2.1.1 Bombas de óleo de caudal variável	5
2.1.2 Sistema de rastreabilidade Implementado	6
2.1.3 Base de dados	8
2.2 Conceitos e tecnologias de suporte ao projeto	9
2.2.1 Base de dados	9
2.2.2 Cadeia de fornecedores	9
2.2.3 Rastreabilidade	9
2.2.4 Computadores de pequena escala	10
2.2.5 Gateways	10
2.3 Conceitos criptográficos	11
2.3.1 Hash Value	11
2.3.2 SHA-2	12
2.3.3 Encriptação Simétrica vs Assimétrica	12
2.3.4 AES CBC	13
2.4 Segurança de dados	14
2.4.1 Ataques que visam a confidencialidade	14
2.4.2 Ataques que visam a integridade	14
2.4.3 Ataques que visam a disponibilidade da informação	15
2.4.4 Message Integrity Code - MIC	15
2.4.5 Identificadores Únicos	15
2.4.6 Assinatura digital	15
2.5 Capacidade dos equipamentos garantirem aleatoriedade	16
2.6 BlockChain	18
2.6.1 «Miners»	19
2.6.2 Características da BlockChain	20
2.6.3 Tipos de BlockChain	21

2.7	Prototipagem do HMI	21
2.8	Modelos de ataque	21
2.9	Soluções de redes de Blockchain existentes no Mercado	22
2.9.1	Hyperledger	22
2.9.2	Ethereum	23
2.9.3	Corda	23
2.10	Trabalho já realizado por outros	23
2.10.1	«Proposal of an automatic traceability system, in Industry 4.0»	23
2.10.2	«An application of blockchain "smart contracts" and IoT in logistics»	25
2.10.3	«Security techniques for the Internet of Things»	26
2.10.4	«Authentication in interactions with IoT devices»	27
2.10.5	«Proposal of a Traceability 4.0 System in Renault CACIA Using Blockchain Technologies»	28
2.11	Conclusões retiradas dos trabalhos anteriores	28
2.12	Tipologia escolhida e pontos a melhorar	29
3	Solução Proposta	31
3.1	Atacante e tipos de ataques	31
3.2	Recolha de informação e IoT	31
3.3	Gateway	33
3.4	Blockchain	33
3.4.1	Nó central	34
3.4.2	Nós «miners»	34
3.4.3	Tipologia de blocos	34
3.5	Base de dados	36
3.6	Segurança de informação	36
3.7	HMI	36
3.8	Análise aos gerados do ESP e Raspberry Pi	36
3.9	Pontos a reter	38
4	Implementação	39
4.1	Testes executados aos números aleatórios	40
4.1.1	Processo de recolha de dados	40
4.1.2	Solução A para renovação de chaves e IV	41
4.1.3	Solução B para a renovação de chaves e IV	41
4.2	Patamar Iot	43
4.2.1	Business Unit 01	44
4.2.2	Leitura Data Matrix	46
4.2.3	Business Unit 02	46
4.2.4	Programação e software patamar IoT	48
4.3	Patamar de Gateway's	49
4.3.1	Processos em execução na Gateway	51
4.3.2	Base de dados Gateway	52
4.4	Segurança de rede	53
4.5	Certificados de autoridade	54
4.5.1	Criação de certificado de autoridade	54
4.5.2	Criação de certificados para cada entidade	55

4.5.3	Assinar o certificado pelo CA principal	55
4.6	Configuração dos certificados	56
4.6.1	Ligações encriptadas e tipos de utilizadores	56
4.7	Blockchain e Base de dados empresarial	57
4.7.1	TLS aplicado à MongoDB	57
4.7.2	Nós da Blockchain	58
4.7.3	Nó central da Blockchain	58
4.8	HMI	59
4.8.1	Prototipagem	59
4.8.2	Conceção	60
4.8.3	Experiência de utilização	60
5	Testes e Conclusões	65
5.1	Desempenho dos geradores de números aleatórios	65
5.1.1	Números aleatórios do esp	65
5.1.2	Números aleatórios Raspberry Pi	66
5.1.3	Números aleatórios ESP pós processado	66
5.1.4	Custo de operação	67
5.1.5	Verificação de encriptação	67
5.2	Análise à base de dados de cada Gateway	69
5.2.1	Leitura de Dados	70
5.3	Conclusões	72
5.4	Sugestões futuras	73
6	Bibliografia	75
	Anexos	79
A	Conceitos alargados	81
A.1	IoT	81
A.1.1	Industria 4.0	81
A.1.2	«Big Data»	82
A.1.3	«Cloud Computing»	82
B	RFC 4122	83
C	Resultados dos testes aos números aleatórios	85
C.1	ESP	85
C.2	Raspberry Pi	85
C.3	ESP pós processado	85
C.4	Raspberry Pi pós processado	85
D	MicroPython	91
D.1	O que é MicroPython	91
D.2	Ambientes de trabalho	91
D.3	Instalação	91
E	Configuração Raspberry Pi para comunicação RS232	95

F	Instalação Wireshark no Raspberry Pi	99
G	MongoDB	101
G.1	Em Linux	101
G.1.1	Configuração de mongod.conf	101
G.1.2	Leitura de dados - Mongo Shell	101
G.2	Windows	102
H	Base de dados parcial da gateway	107
I	Tecnologia implementada e alternativas	111
I.1	Manual de utilizador ZD5800	111
I.2	Peças para leitura	111
I.3	Tecnologias alternativas	111
I.3.1	Bar Codes	111
I.3.2	Recolha de dados	114

Lista de Tabelas

5.1	Entropia e desvio padrão ESP8266	65
5.2	Dados de outros geradores	66
5.3	Entropia média e desvio padrão Raspberry Pi	66
5.4	Entropia média e desvio padrão ESP pós processado	66
5.5	Armazenamento necessário a cada base de dados intermédia	69

Página em branco intencional.

Lista de Figuras

2.1	Planta Aérea Renault Cacia, [1]	5
2.2	Montagem bomba de óleo	6
2.3	Operações linha de produção, [2]	7
2.4	Chão de fábrica	7
2.5	Base de dados Renault Cacia, [2]	8
2.6	Tipos de funções hash, [8]	11
2.7	Exemplificação do processo de gerar um valor hash	12
2.8	Encriptação através de AES CBC, [11]	14
2.9	Desencriptação através de AES CBC, [11]	14
2.10	Assinatura digital	16
2.11	Exemplificação de troca de dados numa blockchain privada	18
2.12	Organização entre blocos	19
2.13	Hyperledger Green House	22
2.14	Ethereum	23
2.15	Corda	24
2.16	Implementação de sistema de rastreabilidade proposto por Diogo Rocha	25
2.17	Explicação da implementação proposta por Leonor Augusto	26
2.18	Security techniques for the Internet of Things - Performance de algoritmos estudados por José Mendes, [27]	27
2.19	Proposal of a Traceability 4.0 System in Renault CA-CIA Using Blockchain Technologies», por Diogo Costa	30
3.1	Esquema da arquitetura proposta	32
3.2	Sequência de trabalho dos nós	35
3.3	Estrutura dos blocos de dados	35
3.4	Processo de execução de testes	38
4.1	Esquema da Arquitetura proposta	39
4.2	Renovação de chaves	42
4.3	Pós processamento de chaves	43
4.4	Recolha de dados IoT e Gateway	43
4.5	ESP8266, [33]	44
4.6	ZD5800	45
4.7	Esquema de ligação	45
4.8	String lida pelo código Data Matrix	46
4.9	ESP32, [34]	47
4.10	GY-91 pinout, [35]	47
4.11	Ligação ESP32-GY-91	48

4.12	Processo de encriptação nos ESP's	49
4.13	Exemplificação de codificação JSON	49
4.14	Raspberry Pi 3 Model B+, [41]	50
4.15	Raspberry Pi 4 Model B, [41]	50
4.16	Certificados na gateway	51
4.17	config.json implementado na Gateway	52
4.18	Processos em execução na Gateway	53
4.19	Exemplificação de funcionamento de encriptação na gateway	54
4.20	Instalação do certificado - passo 1	56
4.21	Instalação do certificado - passo 2	57
4.22	Verificação de certificado ativo	58
4.23	mongod.conf	59
4.24	Prototipagem com Invision Studio	60
4.25	Login	61
4.26	Index	62
4.27	Controlo de elementos da rede parte 1 - Core trust Network	62
4.28	Controlo de elementos da rede parte 2 - Core trust Network	63
4.29	Dashboard	63
4.30	Consulta de informação - query	64
4.31	Control Cockpit	64
5.1	Teste à geração de IV por Diogo Costa	67
5.2	Teste à geração de IV com pós processamento	68
5.3	Verificação de encriptação ao ESP recorrendo ao WireShark	68
5.4	Processo de leitura nas gateway's através da linha de comandos	70
5.5	Processo de leitura nas gateway's por timestamp através da linha de comandos	71
5.6	Tempos de resposta consoante o número de linhas pedido	71
C.1	Gerador de números aleatórios ESP	86
C.2	Entropia ESP8266 10 Mb por ficheiro	87
C.3	Entropia ESP8266 20 Mb por ficheiro	87
C.4	Entropia Raspberry Pi 10 Mb por ficheiro	87
C.5	Entropia Raspberry Pi 3 model B 20 Mb por ficheiro	88
C.6	Entropia ESP8266 10 Mb por ficheiro pós processado	88
C.7	Entropia ESP8266 20 Mb por ficheiro pós processado	89
C.8	Entropia Raspberry Pi 10 Mb por ficheiro pós processado	89
C.9	EntropiaRaspberry Pi 20 Mb por ficheiro pós processado	90
D.1	Porta COM	92
D.2	Processo de instalação MicroPython	93
D.3	Processo de instalação MicroPython - parte dois	93
E.1	Programação de Raspberry Pi para comunicação RS232 - parte 1	95
E.2	Programação de Raspberry Pi para comunicação RS232 - parte 2	96
E.3	Programação de Raspberry Pi para comunicação RS232 - parte 3	96
E.4	Programação de Raspberry Pi para comunicação RS232 - parte 4	97
E.5	Programação de Raspberry Pi para comunicação RS232 - parte 5	97

G.1	Ficheiro Hosts	102
G.2	Instalação MongoDB diretório	103
G.3	Instalação MongoDB - variáveis	104
G.4	Instalação MongoDB - Variáveis - path	104
G.5	Instalação MongoDB - nova variável	105
G.6	Instalação MongoDB - inicialização	105
G.7	Instalação MongoDB - estabelecer ligação ao server	105
G.8	Instalação MongoDB - Compass	106
G.9	MongoDB bases de dados	106
I.1	Manual de utilizador - Sensor ZD5800	112
I.2	Exemplo de peça a ler na linha de produção	113
I.3	Código Gália, [2]	114
I.4	Barcode 1d	114
I.5	Desenvolvimento de Data Matrix	115
I.6	Organização Data Matrix	115
I.7	Estrutura QR code	116
I.8	ESP32	116
I.9	PLC	116

Página em branco intencional.

Listings

4.1	Criação de CA	55
4.2	Certificado para client	55
4.3	Assinatura do certificado	55
4.4	Compactação das credenciais	55
4.5	Criação de certificado TLS dedicado ao navegador	56
G.1	Acesso à base de dados com recurso a TLS	102
G.2	Base de dados - MongoDB	102
G.3	dbcollection - MongoDB	102
H.1	Exemplo ilustrativo de processo da descriptação da base de dados da gateway	107
H.2	Exemplo ilustrativo de processo da encriptação da base de dados da gateway	108

Página em branco intencional.

Glossário

Assinatura digital Código que permite determinar se a informação contida numa mensagem é segurança e proveniente do emissor esperado. xi

Big Data A big data, pode ser traduzida como grande quantidade de informação, e baseia-se na recolha, tratamento e análise de quantidades massivas de informação com o intuito de melhorar a produção prevendo problemas e facilitando a análise de erros. xi, 10

Blockchain Rede que funciona como base de dados de registo distribuído que garante segurança da informação através da descentralização como principal medida de segurança. xi, 12

Cadeia de fornecedores Rede de contactos que é capaz de fornecer a uma dada empresa bens e serviços que esta necessita. xi, 9

Cipher algoritmo de encriptação e/ou desencriptação de informação. xi

Ciphertext Texto obtido após a encriptação encriptado que apenas é possível ler a sua real informação após desencriptação. xi

Data Matrix Data Matrix é um código bidimensional subdividido em blocos pretos e brancas ou pontos dispostos num determinado padrão, formando uma matriz. Este código é utilizado para codificar informação sendo que pode ser texto ou dados numéricos. Este código é bastante semelhante ao código QR. xi, 10

Indústria 4.0 Nome dado à quarta grande revolução industrial e que é devida à crescente automação da indústria bem como da integração de Big Data nas tecnologias e processos de fabrico. xi

Internet of Things «Internet of Things» traduzida como Internet das coisas, é um conceito que acompanha a Indústria 4.0 e refere-se à possibilidade dos equipamentos comunicarem entre si e trocarem informação sobre o ambiente que os rodeia com recurso aos sensores que os incorporam. xi, 10

JSON Formato de informação que permite a fácil compreensão de pessoas e fácil processamento por parte dos equipamentos. xi, 48

Plaintext Informação contida numa mensagem e que é possível ler sem recurso a nenhuma operação. O plaintext refere-se à informação antes ou depois da encriptação da mensagem. xi

Rastreabilidade Capacidade de saber a origem/percurso de um equipamento desde a sua origem até ao consumidor final, dos elementos que o compõem bem como dos processos de fabrico. xi, 1

RFID A tecnologia RFID (radio frequency identification – identificação por radiofrequência) refere-se a tecnologias que utilizam a frequência de rádio para transmissão de dados e/ou intuito de identificação do dispositivo. xi, 10

RSA Algoritmo de criptografia assimétrica mais comum na transmissão de informação segura. xi, 16

Smart Contract É um programa aceite por mais do que um participante e que é executado a cada transacção de informação, sendo que o seu código é determinado previamente e não pode ser alterado. xi, 20

Nomenclatura

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
API	Application Programming Interface
BBC	British Broadcasting Company
BFT	Byzantine Fault-Tolerant
BU	Business Unit
CA	Certification Authority
CBC	Cipher Block Chaining
CBM	Cloud-Based Manufacturing
CSS	Cascading Style Sheets
DES	Data Encryption Standard
DNS	Domain Name System
DPoS	Delegated Proof of Stake
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic Codebook
EEPROM	Electrically Erasable Programmable Read-Only Memory
GPIO	General Purpose Input/Output
HMAC	Hashed Message Authentication Code
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secured
I2C Inter-Integrated Circuit
IoT Internet of things
ISO International Organization for Standardization
IT QRNG Instituto de Telecomunicações - quantum random number generator
ITU International Telecommunications Union
IV Initialization Vector
LDR Light Dependent Resistor
MAC Message Authentication Code
MSP Membership Service Provider
NIST National Institute of Standards and Technology
NSA National Security Agency
PLC Programmable Logic Controller
PoA Proof of Authority
PoS Proof of Stake
PoW Proof of Work
RAM Random Access Memory
RC4 Rivest Cipher 4
RFID Radio-Frequency Identification
ROM Read-Only Memory
RSA Rivest–Shamir–Adleman
SHA Secure Hash Algorithm
SPI Serial Peripheral Interface
SQL Structured Query Language
SSL Secure Sockets Layer
TCP Transmission Control Protocol
TCP/IP Transmission Control Protocol/Internet Protocol
TLS Transport Layer Security
UART Universal Asynchronous Receiver/Transmitter

URL Uniform Resource Locator
USB Universal Serial Bus
UUID Universally Unique Identifier
VDOP Variable Displacement Oil Pump
YAML YAML Ain't Markup Language

Página em branco intencional.

Capítulo 1

Introdução

1.1 Motivação

A produção em larga escala obrigou as empresas a eliminar erros de forma contínua e reduzir tempos de operação, podendo assim reduzir custos de produção, mantendo-se no mercado de forma competitiva. Assim, a Renault Cacia pretende desenvolver um sistema de Rastreabilidade baseado em Blockchain e dispositivos IoT capaz de realizar o controlo e a supervisão interna dos seus produtos e processos de fabrico, tendo o intuito de interagir com os fornecedores, distribuidores e em última instância com o consumidor, aumentando a relação de confiança para com estes. Na implementação de um sistema deste tipo surgem problemas devido a existirem várias frequências de funcionamento, bem como, diferentes protocolos de comunicação entre os equipamentos, sejam estes sensores, os ESP, os Raspberry Pi, o computador, as bases de dados das empresas ou mesmo entre os software's disponíveis em cada equipamento, o que levanta inúmeras questões na sua utilização e implementação. A integração de sistemas de rastreabilidade com as empresas que participam na cadeia de valor, é muito importante, pois o ideal seria obter uma cadeia desde o fornecedor inicial da matéria prima até aos compradores. Desta forma, esta dissertação de mestrado procura solucionar estes problemas, desenvolvendo uma arquitetura que se apresente competitiva no mercado, tendo sempre em vista uma tipologia modular que interaja com o sistema da empresa. Ao nível da segurança é necessário solucionar pontos críticos de comunicação e armazenamento, recorrendo a métodos criptográficos. Com o intuito de analisar a qualidade dos números aleatórios, sobre os quais assenta a encriptação, foram executados testes propostos pelo NIST. Estes avaliam se os números são gerados de modo a que possam ser considerados de facto aleatórios, ou, se por outro lado, são gerados de forma repetitiva, o que poderá levar à transmissão de informação sobre o próprio gerador. Estes números ao serem de baixa qualidade colocam em causa a segurança da informação, levando a que ocorram ataques que visam a leitura da mesma, ou personificação por parte do atacante, fazendo com que os dados armazenados não sejam os esperados. Além disso, procurou-se ainda desenvolver soluções para a renovação de chaves privadas da comunicação, bem como o armazenamento de informação em bases de dados intermédias. Quanto à base de dados considerada principal, o armazenamento faz-se recorrendo a uma rede de Blockchain descentralizada, evitando pontos críticos de falha, criando confiança pelos participantes através da partilha de informação por mais do que um nó desta mesma cadeia.

1.2 Objetivos e Resultados esperados

O objetivo deste projeto, passa pelo desenvolvimento e teste de um sistema automático de rastreabilidade e armazenamento de dados, bem como a sua amostragem, ou seja, o HMI («Human Machine Interface»). Este sistema terá por base a recolha de dados que ajudem a caracterizar as condições do processo em determinados instantes conferindo a possibilidade de rastrear as condições de uma determinada peça. Tendo principal enfoque na segurança das comunicações. A necessidade de rastreamento surge por diversas questões como:

- Produção Just-In-Time;
- Detecção de produtos com defeito em menor tempo e mais fácil remoção destes;
- Big - Data permitindo estatísticas mais precisas;
- Maior transparência da cadeia de fornecimento aumenta a confiança do cliente;

Todas estas questões levam a que seja estabelecida uma cadeia de informação segura e modular, permitindo a expansibilidade. Assim, com o intuito de desenvolver um sistema que possa ser competitivo com o mercado, a plataforma a desenvolver deve de garantir rápido acesso, não só a membros da empresa mas também a utilizadores externos. Garantir que a informação se mantém inacessível a utilizadores externos, inalterada mas sempre disponível para os utilizadores desejados. Para tal, deverá de se basear em equipamentos como ESP32, Raspberry Pi 3 Modelo B+, Raspberry Pi 4 Modelo B, em linguagens de programação usadas em larga escala, como Python ou JavaScript para produzir as interfaces web. Desta forma, podemos apontar os seguintes objetivos para a implementação:

- Proteção da informação armazenada;
- Canais de comunicação seguros;
- Recolha e tratamento de dados de forma fluida;
- Código baseado em software open-source com processamento de baixa complexidade e desenvolvido sempre em vista a modularidade para que assim possa ser expansível;
- Hardware de baixo custo bem como baixa necessidade de manutenção;
- Fiabilidade dos processos a desenvolver;

Referente aos pré-requisitos desejados por parte da Renault Cacia, existe a necessidade de desenvolver uma base de dados baseada em Blockchain, sendo composta por três nós, responsáveis por processar a informação, bem como por um nó admin responsável por colocar em funcionamento os nós e os equipamentos responsáveis por processar as trocas de informação. Outro pré-requisito passa pela utilização de equipamentos de baixo custo para realizar a recolha e partilha de informação, mais especificamente equipamentos IoT como ESP8266 e Raspberry Pi's 3. Por fim, a tipologia de troca de mensagens entre estes equipamentos deveria ainda de se basear em web services, ou seja, pedidos

HTML. Este pedidos HTML devem de respeitar determinados standart's, sejam eles de segurança nos dispositivos de baixa capacidade onde deverá de ser implementado Secure Hash Algorithm (SHA) e Advanced Encryption Standard (AES), ou alternativas. Para dispositivos de maior complexidade surge como necessário a implementação de protocolos como o SSL/TLS (Secure Sockets Layer e Transport Layer Security).

1.3 Guia de leitura

Este documento encontra-se estruturado em cinco partes. A primeira parte, em que nos encontramos, o capítulo 1, tem um aspeto introdutório. O segundo capítulo, Estado de Arte, retrata as noções e conhecimentos que serão necessários à compreensão do trabalho, mas também, as tecnologias existentes e alguns trabalhos que já foram desenvolvidos nesta área, que de certa forma influenciou a forma como o trabalho foi desenvolvido. Em seguida, é apresentado o terceiro capítulo, onde se irá abordar temas como o ambiente esperado em chão de fábrica de seguida a arquitetura idealizada, bem como esta se encontra estruturada. O quarto capítulo, Implementação, aborda-se aspetos referentes à sua implementação física e soluções desenvolvidas. Por fim, o quinto capítulo, a Conclusão, faz uma síntese do trabalho realizado, o seu desempenho e aspetos que poderão ser alterados em trabalhos futuros.

Página em branco intencional.

Capítulo 2

Estado de Arte

Ao longo deste capítulo iremos abordar assuntos como a empresa a que se destina a implementação a ser desenvolvida bem como conceitos importante para a correta compreensão do trabalho, tecnologias existentes no mercado, bases de dados e trabalhos já realizados por terceiros.

2.1 Introdução à Renault Cacia

A Renault Cacia é uma empresa pertencente ao grupo Renault. O grupo é composto por 40 unidades industriais disposta por 16 países, sendo que a «Companhia Aveirense de Componentes para a Indústria Automóvel (C.A.C.I.A)», Renault Cacia, Figura 2.1, tenha sido fundada em setembro de 1981. Aos dias de hoje, produz componentes mecânicos com maior foco nas caixas de velocidades, bombas de óleo e diferenciais exclusivamente para o grupo Renault-Nissan-Mitsubishi Alliance. Atualmente a empresa encontra-se dividida internamente por nove departamentos, nomeadamente: Produção, Engenharia, Logística, Garantia de qualidade, Finanças, Técnico, Tecnologias de informação, Sistemas de produção e Recursos Humanos, [1].



Figura 2.1: Planta Aérea Renault Cacia, [1]

2.1.1 Bombas de óleo de caudal variável

Esta dissertação incide sobre o desenvolvimento de um sistema de rastreabilidade para a linha de montagem de bombas de óleo de caudal variável. Estas bombas, são responsáveis pelo correto fornecimento de óleo ao motor dos carros que equipam, [1].

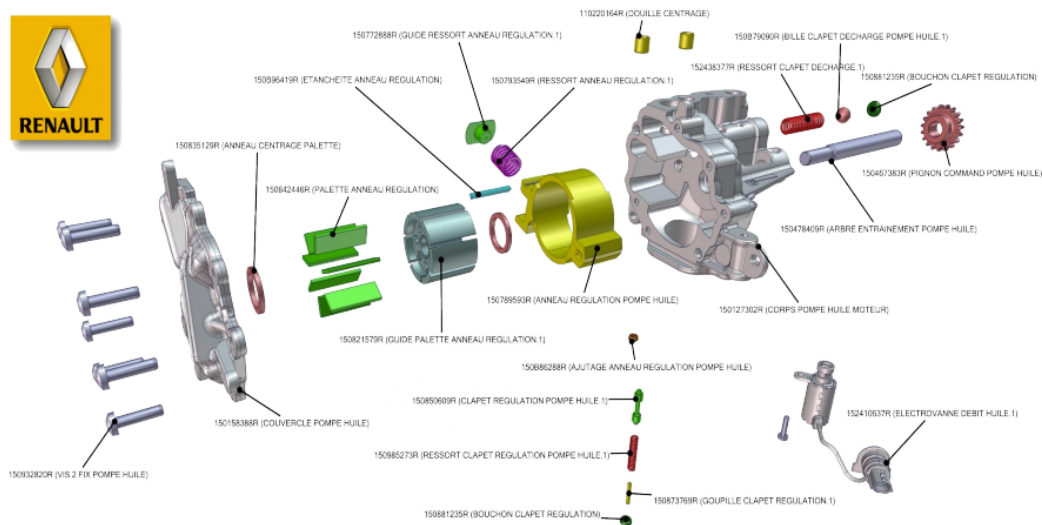


Figura 2.2: Montagem bomba de óleo

As bombas de óleo são compostas por diversos elementos, visível na Figura 2.2, os quais são montados ao longo de duas linhas, exemplificadas no capítulo seguinte. As duas linhas apresentam diferenças pouco significativas e em conjunto produzem anualmente mais de 1 616 000 unidades, [1].

Como a implementação a desenvolver se destina a implementação na linha de bombas de óleo de caudal variável da Renault Cacia será importante iniciar o trabalho por perceber como funciona esta a montagem destas bombas e da própria linha. A linha representada a amarelo na Figura 2.4 é a linha 1, e a verde a linha 2. Os componentes que compõem a estrutura da bomba chegam à linha nas áreas apontadas como Corpo e tampa. Na Figura 2.3, podemos observar a descrição das operações desenvolvidas na segunda linha. Esta linha apresenta-se mais automatizada do que a primeira linha, contudo o diagrama continua a ser válido, sendo que esta apenas é complementada com algumas atividades manuais. Na Figura 2.4, é ainda visível a castanho, a operação 91. Esta secção, é responsável por reparar defeitos que tenham sido encontrados nas bombas quando o é possível.

2.1.2 Sistema de rastreabilidade Implementado

Quando uma bomba defeituosa é encontrada, os departamento de qualidade e logística têm assegurar a capacidade de rastrear o problema e identificar as bombas que possuam o mesmo defeito, tal como, os exatos motores em que foram instaladas. Desta forma a empresa possui o sistema Gália. Assim, cada peça externa que entra na linha é marcada com uma tag Gália, permitindo identificar o seu número de série, o código de barras que tem associado, o fornecedor bem como o lote a que pertence.

O sistema permite identificar os parâmetros de produção de cada peça. Desta forma, é possível identificar os parâmetros de produção de uma peça em específico ou de várias. Assim, quando uma peça dá entrada pela estação 110 é registada no sistema pelo operador que também lhe atribui um código de barras para que esta possa ser identificada. À data, a empresa está a implementar um sistema em que todas as peças dão entrada nas linhas

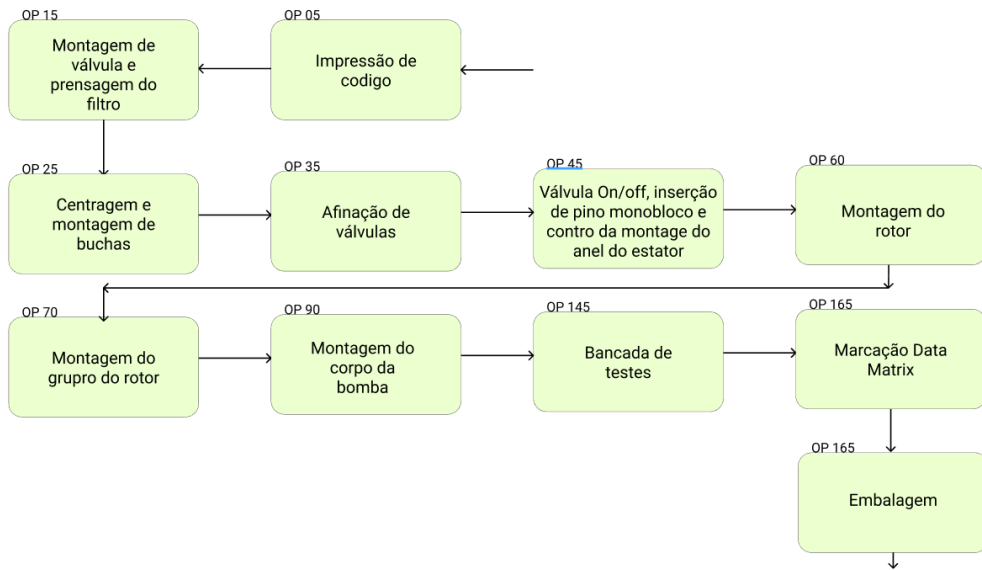


Figura 2.3: Operações linha de produção, [2]

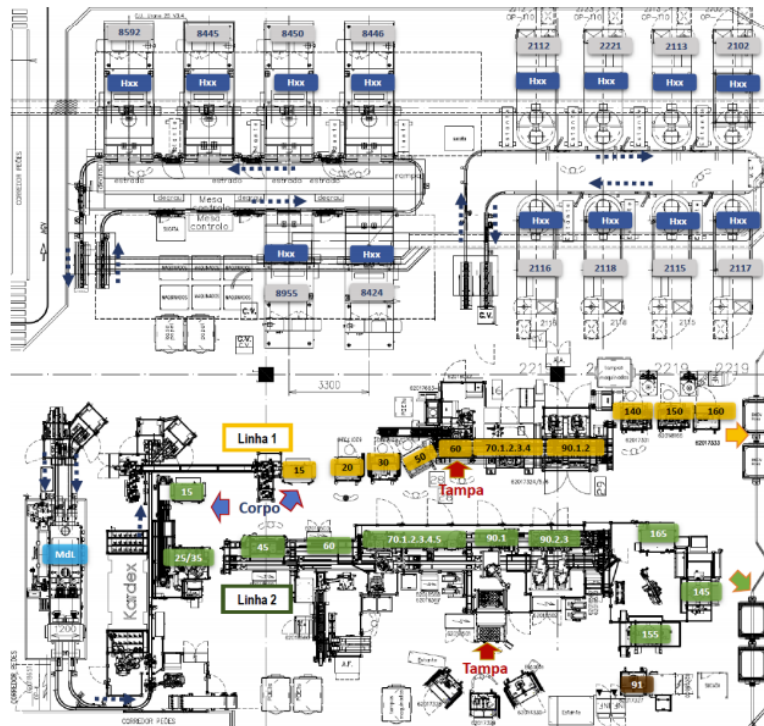


Figura 2.4: Chão de fábrica

com um código DATA MATRIX, código esse que será sobre o qual a nossa implementação irá trabalhar.

No sistema ainda implementado, ao longo da linha de produção sempre que uma peça chega a um posto de trabalho novo procede-se à sua identificação, através do código de barras, sendo no fim adicionado um código Data Matrix após passar todos os testes de fiabilidade. Este código tem embebido informação como o número da peça, que se pertencer à linha um será de 0001 até 4999, ou caso pertença à linha dois, de 5000 a 9999, bem como o ano de fabricação. Toda a informação é gravada num server central e por segurança, redundantemente, num disco duro responsável por uma back-up. Como principais limitações existentes no sistema podemos apontar a dificuldade de afirmar que a peça sai na mesma palete que entrou na linha e como segunda principal dificuldade ainda o rastreio não ser feito a peças individuais.

2.1.3 Base de dados

A informação guardada depende principalmente do processo que estamos a analisar, como visível pelas entradas na Figura 2.5. Se analisarmos o processo de fabrico da estação OP140 iremos encontrar informação como quando se iniciou a produção (data e hora), o tempo que levou para ser concluído, o resultado do teste de qualidade, se a peça foi reparada, bem como parâmetros mecânicos sobre os quais a peça foi avaliada. Toda esta informação é associada ao código de barras da peça, que virá a dar lugar ao código Data Matrix, que, como já foi referido passará a vir previamente marcado na peça. Desta forma, podemos concluir que o sistema desejado tem de ser facilmente adaptado aos dados necessários a recolher em cada estação bem como a possível transição de peças entre estações dado que a peça pode saltar estações ou repeti-las, bem como ser removida da linha. Assim a recolha de dados deve de proceder de forma automática e alocar os dados recolhidos diretamente à peça que dê entrada.

OP140 Barcode	OP140 DateAndTime	OP140 Good	OP140 Recipe	OP140 ScrapCode	OP140 Reworked	OP140 FreeRotMaxTrq	OP140 SafeVlvOpPress	OP140 SafeVlvPres	OP140 RegVlv1PreStOp	OP140 RegVlv1MaxOp	OP140 RegVlv1PrEndCl	OP140 VacTstPrEndTst	OP140 VacTstTstSpeed	OP140 Cycletime
						Nm	Bar	Bar	Bar	Nm	Bar	mBar	Rpm	s
SpaperinoS	10-20-12 10:36:32	0	8224	8224	0	0	0	0	0	0	0	0	8224	
SB1344120875S	12-10-12 15:58:09	1	59	0	0	0.04	0	0	5	1.29	3.5	-27.6	-400	
SB1344120866S	12-10-12 15:59:21	1	59	0	0	0.03	0	0	5.1	1.25	3.4	-27.1	-400	
SB1344120893S	12-10-12 16:00:38	1	59	0	0	0.04	0	0	5	1.29	3.9	-27.8	-400	
SB1344120879S	12-10-12 16:04:18	1	59	0	0	0.05	0	0	5.1	1.29	3.4	-31.6	-400	
SB1344120877S	12-10-12 16:05:13	1	59	0	0	0.04	0	0	5	1.27	1.7	-39.7	-400	
SB1344120880S	12-10-12 16:06:13	1	59	0	0	0.04	0	0	5.2	1.26	3.5	-26.6	-400	
SB1344120872S	12-10-12 16:08:30	1	59	0	0	0.04	0	0	5.1	1.29	3	-28.3	-400	
SB1344120870S	12-10-12 16:09:50	1	59	0	0	0.04	0	0	5.2	1.25	2.8	-30.7	-400	
SB1344120887S	12-10-12 16:10:45	1	59	0	0	0.03	0	0	5.2	1.15	3.1	-31	-400	
SB1344120882S	12-10-12 16:25:13	1	59	0	0	0.04	0	0	4.9	1.28	3.2	-28.3	-400	
SB1344120895S	12-10-12 16:45:05	1	59	0	0	0.04	0	0	5	1.23	2.7	-32.2	-400	
SB1344120901S	12-10-12 17:05:14	1	59	0	0	0.04	0	0	5.2	1.22	0.2	-32	-400	
SB1344120900S	12-10-12 17:06:20	1	59	0	0	0.04	0	0	4.9	1.28	3.1	-36.9	-400	
SB1344120903S	12-10-12 17:08:58	1	59	0	0	0.03	0	0	5	1.29	3.6	-31.4	-400	
SB1344120896S	12-10-12 17:12:49	1	59	0	0	0.03	0	0	5	1.27	3.4	-25.8	-400	
SB1344120897S	12-10-12 17:14:55	1	59	0	0	0.04	0	0	5.1	1.27	3.5	-25	-400	
SB1344120899S	12-10-12 17:19:24	1	59	0	0	0.04	0	0	5	1.26	3.6	-34	-400	
SB1344120865S	12-10-12 17:20:42	1	59	0	0	0.04	0	0	4.7	1.28	3.7	-32.6	-400	
SB1344120905S	12-10-12 17:21:59	1	59	0	0	0.04	0	0	5	1.27	3.3	-30.3	-400	
SB1344120906S	12-10-12 17:25:28	1	59	0	0	0.04	0	0	5	1.28	2	-31.4	-400	
SB1344120904S	12-10-12 17:26:27	1	59	0	0	0.04	0	0	5.2	1.28	3.7	-27.4	-400	
SB1344120898S	12-10-12 17:34:25	1	59	0	0	0.03	0	0	5	1.28	3.1	-32	-400	
SB1344120908S	12-10-12 17:55:37	1	59	0	0	0.05	0	0	5.1	1.28	3.6	-29.9	-400	
SB1344120909S	12-10-12 17:58:06	1	59	0	0	0.07	0	0	4.9	1.29	3	-33.7	-400	

Figura 2.5: Base de dados Renault Cacia, [2]

2.2 Conceitos e tecnologias de suporte ao projeto

2.2.1 Base de dados

Um sistema de Base de dados é uma tipologia de armazenamento de informação normalmente relativa a pessoas, equipamentos ou informação sobre sistemas, com o intuito de poder ser analisada mais tarde para proceder à toma de decisões de forma mais segura. No campo do armazenamento de dados, surgem dois sistemas opostos, bases de dados *Relacionais* e bases de dados *Não Relacionais*. As bases de dados relacionais, apresentam em cada linha da tabela uma única chave primária. As colunas contêm atributos dos dados, e regra geral, é facilmente encontrada uma relação entre os dados, o que facilita a pesquisa de informação. O sistema mais conhecido deste tipo de base de dados é o MySQL, contudo existem outros sistemas, como por exemplo, Oracle Database ou IBM DB2. Todos os referidos possuem características muito semelhantes. [3]

Por outro lado, as bases de dados não relacionais, também conhecidas como NoSQL, fornecem uma chave única, id, gerado pelo próprio sistema, que permite recuperar os dados em caso de perda dos mesmos. Este tipo de sistemas permite introduzir informação em colunas que não foram fornecidas anteriormente, ao contrário das bases de dados relacionais, em que todo o tipo de informação tem de ser previsto. Desta forma para aceder à informação é apenas necessário um «ponteiro». Como tal, quando pretendemos aceder a mais do que uma coluna de informação é necessário trabalhar a informação existente e realizar mais do que uma consulta. Este tipo de base de dados confere maior liberdade de leitura, mas necessita de uma maior capacidade de processamento, [3].

2.2.2 Cadeia de fornecedores

A Cadeia de fornecedores é uma rede global de organizações, pessoas, informação, recursos e atividades que fornecem uma empresa tanto a nível de recursos, como de serviços, com o objetivo de responder aos pedidos dos seus clientes [4]. O objetivo principal de uma cadeia de fornecedores é garantir o funcionamento eficiente da rede respondendo o mais rápido e objetivo quanto possível aos pedidos dos clientes. Desta forma fazem parte de uma cadeia de fornecedores elementos como:

- Produtores
- Distribuidores
- Retalhistas
- Clientes
- Fornecedores de serviços

Assim sendo, cada empresa tem a possibilidade de avaliar se lhe é mais oportuno contratar um serviço ou adquirir um equipamento, parte de um produto ou investir por ela própria na criação do produto/serviço [5].

2.2.3 Rastreabilidade

A rastreabilidade é a capacidade de cada participante saber em que estado se encontra um produto ou cada componente deste, assim como, por onde passou e que operações foi

submetido ou ainda para onde irá de seguida. Em suma, terá de ser capaz de fornecer toda a história de um produto e dos seus componentes. A rastreabilidade é um aspeto largamente adotado pelas empresas, visto ser bastante importante para o planeamento da produção e assim evitar falhas na produção, ou por outro lado, produção em excesso que significaria despender recursos de forma incorreta. Por outro lado, a rastreabilidade permite detetar peças com defeito, uma vez que, se uma peça tiver um defeito é possível que, numa produção em série, tenham sido produzidas mais peças da mesma forma, como tal, poderá ser necessário saber que peças foram produzidas que possam ter o mesmo defeito e substituí-las. Assim para além de saber onde e quando foram produzidas, é importante saber qual foi o passo seguinte na cadeia de fornecimento. Assim a rastreabilidade é complementada por tecnologias como RFID, códigos de barra, Data Matrix e bases de dados. De facto, a rastreabilidade tem como principais pontos positivos [6]:

- Monitorização de stocks permitindo planeamento de produção de forma eficiente;
- Possibilidade de produção «Just-in-Time», JiT;
- Detecção atempada de defeitos e causa deste;
- Otimização de produto e método de produção com base em Big Data;
- Aumento da confiança do consumidor por larga quantidade de informação disponível;

2.2.4 Computadores de pequena escala

Ao longo dos últimos anos tem vindo a crescer a utilização de micro processadores, sejam eles para computadores ou para micro controladores utilizados em sistemas de pequena escala. Estes conseguem facilmente interagir com sensores ou atuadores, enquanto se conectam à Internet. Tudo isto deve-se ao facto de permitirem a sua programação através de diversas plataformas ou linguagens, como micro controladores mais conhecidos temos o Arduino ou os ESP, [7].

2.2.5 Gateways

Uma gateway é uma plataforma física ou de software, que permite a troca de informação entre dispositivos, como por exemplo, módulos pertencentes à «Internet of Things», também denominados de módulos IoT, bem como outros equipamentos que possam estar na mesma rede ou em redes externas, tendo a possibilidade de ter em si próprias código que, normalmente, se apresenta escrito em Python ou Java, sendo que por vezes, ainda em Node.js, o que confere a possibilidade, de para além de realizar a troca de informação, realizar ainda o tratamento desta bem como o seu armazenamento. Em suma, são o pilar da troca e tratamento de informação, podendo conferir ainda segurança e fiabilidade à rede. Exemplo de uma gateway seria um RaspberryPi a trocar informação entre vários módulos IoT e a reenviar para um computador remoto, [7].

2.3 Conceitos criográficos

2.3.1 Hash Value

A função **hash**, é um método «computacionalmente eficiente de binarizar strings de tamanho variável para strings de tamanho fixo, chamada hash value». Os valores hash são normalmente utilizados para verificar se uma mensagem foi alterada, dado que uma mensagem tem apenas um valor hash, caso, algo se altere, por muito pouco que seja essa diferença, o valor gerado será diferente. Se dois computadores diferentes tiverem de calcular o valor hash de um ficheiro igual obtêm o mesmo valor. Comparar o valor hash de dois documentos apresenta-se drasticamente mais rápido do que comparar a integridade total dos dois ficheiros, enquanto que, se recorrer a chaves simétricas disponibiliza um elevado nível de proteção de dados, integridade e autenticação. Assim as funções hash podem ser divididas em dois grupos, um em que não utiliza uma chave privada e por norma é destinado a detetar alterações em documentos ou blocos de informação e outro destinado a autenticar mensagens, como exemplificado na Figura 2.6. Já na imagem 2.7 é ilustrado de forma simplificada o processamento destas funções, [8].

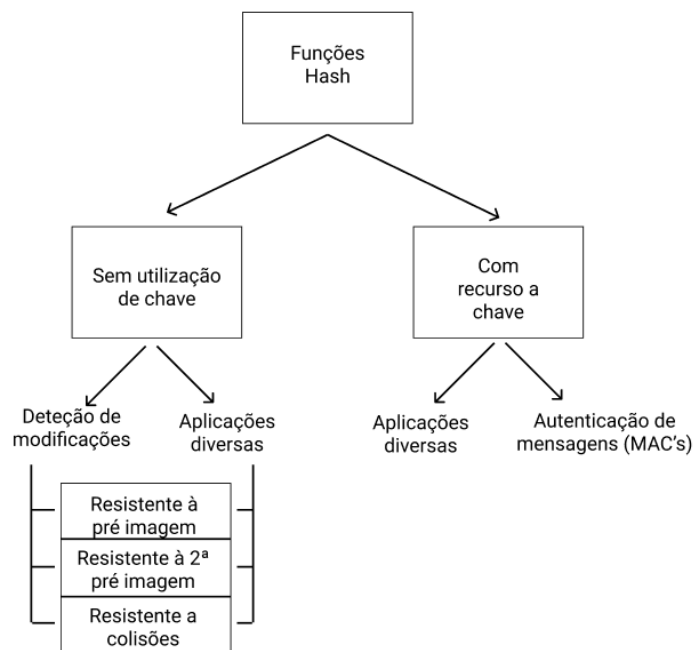


Figura 2.6: Tipos de funções hash, [8]

As principais características das funções hash apontadas por Menezes passam por:

- **Eficiência computacional:** o processamento matemático destas funções é realizado em curtos períodos de tempo, mesmo quando aplicadas a documentos extensos;
- **Algoritmo determinístico:** se for dado a mesma entrada de texto à função será obtido o mesmo valor de saída (código), mesmo em equipamentos distintos;
- **Resistente à pré imagem:** dado um valor específico de hash y , deve ser computacionalmente inviável encontrar uma pré-imagem x tal que $h(x) = y$, chegando a ser

mais fácil obter colisões para o mesmo hash do que o próprio documento;

- Resistente à colisão: dois documentos diferentes, quando processados pela mesma função, obtêm códigos diferentes, por muito semelhantes que sejam.

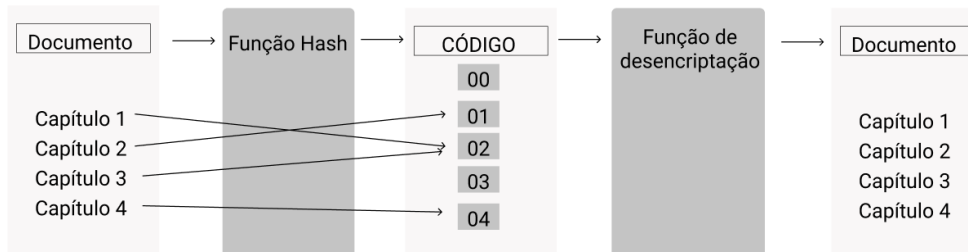


Figura 2.7: Exemplificação do processo de gerar um valor hash

2.3.2 SHA-2

Nos dias de hoje a função de encriptação mais utilizada nas redes de Blockchain é a SHA-2, SHA-256. Sendo que o 256 refere-se ao número de bits de saída da própria função, ou seja, SHA-256 gera 256 bits, o que equivale a 32 bytes, sendo assim visível 32 caracteres alfa numéricos no código de saída da função. Estas funções são utilizadas para gerar um bloco de dados menor que o comprimento total do texto, necessitando assim de menor espaço em disco para fazer o seu armazenamento, ou menor capacidade para fazer o seu envio, sendo usual que apenas este código seja enviado e não o conjunto total da mensagem.

SHA-2 é um conjunto de funções desenvolvido pela NSA (Agência de Segurança Nacional), ao qual procederam à sua publicação no ano de 2001, sendo composto por SHA-224, SHA-256, SHA-384, SHA-512, estas funções apesar de a sua aplicação e funcionamento serem bastante similares têm diferenças marcantes quanto ao seu processo interno e valor de saída final. A principal diferença visível passa pelo número de bits que compõem a resposta final da função, contudo, o tamanho do bloco utilizado por cada algoritmo desta família é sempre de 512 bits. Como cada nome indica, e já referido, SHA-224 por exemplo fornece uma resposta de 224 bits, já a SHA-512 gera 512 bits. Existe também diferenças no processamento dos ficheiros, sendo que para além de variar o número de bits de saída varia também o número de caracteres em que o texto é repartido bem como o número de vezes que este é processado, a título de exemplo, a função SHA-256 processa 64 vezes, [9].

2.3.3 Encriptação Simétrica vs Assimétrica

Cada função pode ainda ser classificada como dois tipos de encriptação, simétrica e assimétrica.

Encriptação simétrica utiliza apenas uma única chave privada partilhada pelos dois equipamentos para realizar a cifra/decifra da mensagem. Os algoritmos mais comuns passam pelo AES, DES 3DES e RC4. Estes algoritmos apresentam-se como extremamente rápidos e de baixa complexidade, contudo o principal ponto negativo passa pela

necessidade de que todos os elementos envolvidos saibam previamente qual é a chave para proceder ao tratamento da informação. A maior parte dos algoritmos é baseado em blocos, ou seja, o algoritmo reparte a mensagem em várias strings de tamanho fixo, sendo que cada bloco é encriptado à vez. Na maioria dos casos, a mensagem a cifrar não é múltipla de 16^D , o que cria a necessidade de juntar à mensagem informação não útil para satisfazer os requisitos de encriptação, esta ação é conhecida por padding, consistindo normalmente em adicionar espaços em branco à mensagem, sendo assim possível cifrar a mensagem composta por blocos de 16 bytes, [10].

A **encriptação assimétrica** utiliza duas chaves diferentes que são criadas por um só elemento, uma para proceder à encriptação e outra para proceder à desencriptação. Para conseguir estabelecer a troca de informação, procede-se à partilha de uma chave pública visível por todos, que apenas pode ser utilizada para encriptar a informação. Tal chave é complementada por uma privada que nunca é partilhada e que possibilita a desencriptação da informação por parte do elemento que procedeu à partilha da chave pública. O algoritmo mais comum é o RSA. Quando comparado com a encriptação simétrica, este tipo de encriptação apresenta-se muito mais dispendioso computacionalmente, o que trás como consequência maiores tempos de processamento. Como tal, não é habitualmente utilizado para enviar informação, mas sim para estabelecer canais de comunicação seguros através de meios não seguros, a título de exemplo, a Internet. Como ponto positivo apresenta ainda o fator de não ser necessário realizar uma distribuição de chaves previamente, [10].

Soluções robustas do ponto de vista de encriptação utilizam os dois meios. A assimétrica para estabelecer comunicação e a simétrica para realizar a troca de informação, [10].

2.3.4 AES CBC

O modo CBC é dos modos mais utilizados aos dias de hoje, contudo assenta no princípio que todos os intervenientes sabem qual é a chave de desencriptação e que esta é segura o suficiente. A encriptação AES é composto por um vetor de iniciação, normalmente denominado de IV («initialization vector»), por uma chave secreta, já o modo CBC é a forma como o algoritmo processa a encriptação, sendo que neste modo cada bloco encriptado depende dos blocos anteriores, por outras palavras, uma encriptação em cadeia, contudo, o primeiro bloco é o IV e atua como um bloco «falso», sendo que em cada bloco enviado o seu tamanho é o mesmo, mas sempre diferente. Assim o IV e a chave privada conferem aleatoriedade ao processo, o que torna extremamente difícil o roubo de informação por ataques através de «força bruta», entenda-se, tentativa erro, [11].

Todo o processamento da informação de forma segura estará sempre dependente de quão boa é a chave de encriptação e de quão aleatório ou sequencial é o IV, se o equipamento emissor produzir IV's de forma repetitivos todo o processo estará comprometido. O processo de encriptação pode ser mais facilmente percebido pela imagem 2.8, onde é visível como parâmetros de entrada o IV como bloco inicial à esquerda e a chave secreta para iniciar o processo de encriptação, sendo a saída desta função a entrada do próximo bloco a processar. Já o processo inverso de desencriptação da informação é baseado no fator de que o recetor conhece a chave e sabe o tamanho do IV inicial, visível na Figura 2.9

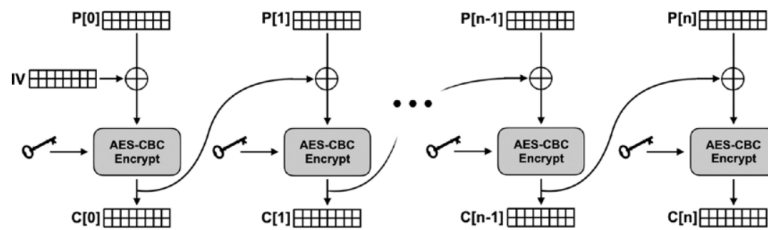


Figura 2.8: Encriptação através de AES CBC, [11]

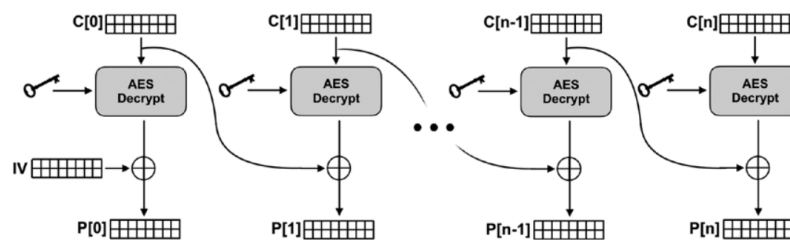


Figura 2.9: Descriptação através de AES CBC, [11]

2.4 Segurança de dados

Uma rede para poder ser considerada como completamente segura tem de cumprir três requisitos. O primeiro, **confidencialidade** apresentando-se como sendo a proteção da informação, tanto durante a transferência desta como durante o seu armazenamento. A **integridade** significa que apesar de a informação sofrer mudanças, estas apenas podem ser realizadas por uma entidade credenciada para tal, bem como através de meios adequados. O último requisito é a **disponibilidade**. A informação apenas é útil quando pode ser acedida em tempo correto e por entidades autorizadas. Sendo ainda importante reter que **uma rede é tão segura quanto o seu ponto mais frágil**, [10]. De acordo com o autor Forouzan, [10], os ataques a uma rede podem ser divididos em três grupos apresentados nas subsecções seguintes.

2.4.1 Ataques que visam a confidencialidade

Os ataques à confidencialidade são principalmente de dois tipos. Um deles, o «**Snooping**», em que o atacante procura aceder ou intercetar informação durante a transmissão desta. Para os contrariar devem de ser implementadas técnicas de encriptação de forma a que a informação não possa ser lida. O segundo tipo de ataque passa por executar **análises de tráfego**, dado que ao analisar o tráfego on-line existe a possibilidade de recolher alguma informação devido à existência de repetibilidade de ações, mesmo quando a informação se encontra encriptada.

2.4.2 Ataques que visam a integridade

Quanto aos ataques à integridade da informação estes são essencialmente de quatro tipos, sendo o primeiro à **Modificação**, procurando alterar ou eliminar informação depois de esta ser intercetada. No segundo tipo de ataque, **Personificação**, o atacante faz-se passar

por outra pessoa enganando a vítima após ter obtido as credenciais do emissor. O terceiro método passa pelo **Reenvio**, onde existe um reenvio de mensagem por parte do atacante, sendo que por norma engloba a personificação e a modificação da mensagem. Por fim, a **Repúdio**, em que tanto o emissor como o recetor podem alegar que a informação não cumpre os parâmetros habituais.

2.4.3 Ataques que visam a disponibilidade da informação

Estes ataques, **Negar o acesso à informação**, passam por colocar o sistema com tempos de resposta extremamente elevados ou mesmo impossibilitar a resposta do sistema, tal é conseguido por sobre carga no sistema.

2.4.4 Message Integrity Code - MIC

Em português «código de integridade de mensagem», vulgarmente conhecido por MIC, e por vezes denominado de MAC quando existe renovação de chaves secretas, é um código que possibilita verificar a integridade de uma mensagem. Como é explicado no livro [10], por exemplo, a Alice cria um código MAC através de uma função hash (poderia ser outro processo de encriptação), utilizando uma chave secreta. Envia a mensagem ao Bob e o código obtido. Após ler a mensagem, o Bob cria um novo MIC proveniente da mensagem e da chave secreta que já possuía. Compara o novo MIC com o que recebeu, se estes forem iguais, a mensagem é validada, [11].

2.4.5 Identificadores Únicos

Aos dias de hoje, existem dois métodos largamente utilizados para proceder à nomeação de equipamentos numa rede, entenda-se o MAC Address («media access control address») e o UUID («universally unique identifier»). O MAC Address é um endereço físico normalmente implementado no hardware, sendo composto por 48 bites e utilizado para identificar cada equipamento que compõem uma rede que se reja pelo protocolo Ethernet, Wi-Fi ou Bluetooth, [10].

O UUID, em português, identificador universal único, é um número gerado por fontes ditas aleatórias e que tem o intuito de criar números únicos para que não exista conflito na identificação de equipamentos. Este número pode realizar as mesmas funções que o MAC address, estando ainda de acordo com a norma RFC 4122. A biblioteca existente em Python contempla várias versões de UUID's, todavia todas geram um identificador de 16 bytes mudando apenas a forma como este é conseguido. Para efeitos de criptografia o UUID mais aconselhado é a versão 4, UUID4, que se baseia em fontes aleatórias do equipamento. Contudo, é necessário analisar também a aleatoriedade dos números obtidos e analisar se os números obtidos podem ser considerados únicos, devido à forma como o equipamento gera entropia, assunto que será abordado na secção 2.5. Mais informação poderá ser encontrada na documentação da biblioteca em questão [12].

2.4.6 Assinatura digital

A assinatura digital é outro método que fornece a possibilidade de conferir integridade e autenticação à mensagem. Enquanto que o MAC utiliza uma chave secreta para codificar a informação, este método utiliza um par de chaves públicas/privadas.

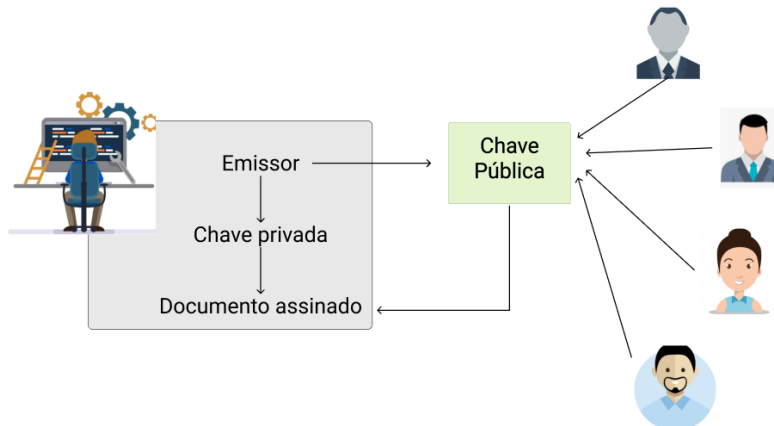


Figura 2.10: Assinatura digital

Uma assinatura em documentos comuns, significa que uma dada pessoa aprovou um documento, por outras palavras, que o documento provém daquela pessoa e que é autêntico. A assinatura digital tem o mesmo efeito que uma pessoa assinar um documento físico. Para proceder a uma assinatura digital, o emissor utiliza um algoritmo de assinatura específico e envia juntamente com a mensagem que pretende verificar. Por sua vez, o recetor, executa o algoritmo de verificação. É de notar que o documento é assinado com uma chave e qualquer pessoa pode verificar se este pertence ao emissor em questão, contudo, a forma como este é verificado, ou seja, assinado, é privada, e só o emissor em questão é que possui forma de o fazer, não podendo partilhar assim a sua chave privada, [10]. Este processo encontra-se exemplificado pela imagem 2.10. Importa salientar que o algoritmo utilizado, entenda-se a chave, tem de estar certificado por uma entidade.

Entre os vários algoritmos de encriptação encontra-se o RSA. Este algoritmo utiliza o esquema de chave pública e privada, sendo que ambos concordam com a função hash. É de notar que para este método ambos necessitam de possuir um par de chaves para poder realizar comunicações, Este método tem como maior vantagem a possibilidade de ambos os intervenientes poderem ler e escrever mensagens, visto que ambas as operações utilizam o mesmo método. Em alternativa surge o DSS, apresentando-se como mais seguro, mas mais dispendioso, contudo tanto o DSS como RSA cumprem os requisitos para aplicar em protocolos como TLS, «Transfer Layer Security», [11].

2.5 Capacidade dos equipamentos garantirem aleatoriedade

Os métodos referidos acima de assinatura de mensagem ou própria identificação de mensagem tem como alicerce o princípio de que a chave que estão a utilizar é única e que provém de um gerador de chaves, entenda-se números aleatórios, ou seja, o equipamento que gerou esta chave baseou-se em variáveis aleatórias e que será praticamente impossível obter outro número igual ou até parecido de forma aleatória. Contudo isto nem sempre acontece. Todos os geradores de números tem por base fontes de entropia para gerar os seus números. A entropia é definida como a medida de desordem ou aleatoriedade. Outro conceito importante será o de entropia mínima, sendo a medida usada na recomendação

do NIST, consistindo na medida da imprevisibilidade de uma variável aleatória, [13].

Dito isto, a entropia mínima, em bits, de uma variável aleatória X é o maior valor m , tendo a propriedade de que cada número gerado de X contenha pelo menos m bits de informação do próprio X , [14]. De forma mais precisa, a entropia mínima de uma variável aleatória discreta independente X que assume valores do conjunto

$$A = x_1, x_2, \dots, x_k \quad (2.1)$$

com probabilidade

$$P_r X = x_i = p_i \quad (2.2)$$

quando $i = 1, \dots, k$ é definido como

$$H = \min_{1 < i < k} -\log_2 P_i = -\log_2 \max_{1 < i < k} P_i. \quad (2.3)$$

Se X tem entropia mínima H , então a probabilidade de observar qualquer valor particular para X não é maior que 2^{-H} . O valor máximo possível para a entropia mínima de uma variável aleatória com k valores distintos é:

$$\log_2 k \quad (2.4)$$

Este valor é obtido quando a variável aleatória tem uma distribuição de probabilidade uniforme, [14]. Os estudos que possam ser executados, por norma, baseiam-se em «Independent and Identically Distributed data set (IID)», ou seja, cada amostra é independente de outra, e deverá de ter a mesma distribuição. Por outras palavras, um elemento pertencente a esta sequência é independente das variáveis aleatórias que existiram antes deste. O oposto portanto, refere-se a uma sequência em que a distribuição de probabilidade para n variáveis aleatórias consiste numa função de variáveis aleatórias presente numa dada sequência, [14].

Um exemplo ilustrativo será o lançamento de um dado. Antes de existir qualquer lançamento a probabilidade de sair qualquer face do 1 ao 6 é de $1/6$, contudo se na primeira amostragem sair o valor 1, a probabilidade considerada passa a ser de 0.5 para o valor 1 e de $1/10$ para os restantes cinco valores, partindo do pressuposto que não é considerado que o dado não se encontra viciado. Se na primeira amostragem sair o valor 6, a probabilidade de sair qualquer valor de 1 a 5 será de $1/10$ e de $5/10$ para o valor 6.

Em suma, um gerador de números aleatório de n bits terá a sua entropia média mínima entre zero e N , sendo N o número de bits que compõem os números testados. Por sua vez, nunca deverá de alcançar os extremos. Entropia próxima de zero é um valor extremamente fraco e próximo de N de elevada qualidade.

Os geradores de números podem ainda ser divididos em dois grupos, o primeiro o grupo «True», a que pertence os quânticos e os físicos ou por outro lado os pseudoaleatórios apresentam-se como o segundo grupo e são baseados em algoritmos, diferindo na forma como obtêm entropia. Os geradores não quânticos são os mais convencionais, dado que geram entropia através de leitura das condições do meio que os rodeia, seja ela temperatura, luminosidade, humidade entre outras leituras que possam efetuar [15].

Já os geradores de números quânticos geram a sua entropia com base em física quântica. Analisando geradores de elevada capacidade, estes apresentam valores de entropia com média igual ou superior 90%, ou seja, se produzirem números de 8 bits, a sua entropia média deverá de rondar pelo menos 7,5 valores, como tal, pode-se apontar este valor

como sendo o valor que se pretende alcançar quando se trabalha com geradores mais comuns, dado ser o valor obtido em geradores de maior complexidade. A problemática inerente à comparação destes dois tipos de geradores passa pelas fontes de entropia, que nos geradores comuns podem ser controladas ou pelo menos simular os valores pretendidos. Por exemplo se um gerador tiver como fonte de entropia a temperatura que o rodeia, se controlarmos o valor de temperatura lido, mesmo que não seja o real, controlamos a entropia que este gera. Assim os geradores de números quânticos conferem uma maior segurança, dado que geram a sua entropia baseada em física quântica, fontes que podem ser consideradas extremamente difíceis de alterar [15].

2.6 Blockchain

A Blockchain é uma combinação de blocos ligados por funções «hash» tendo por base a distribuição de informação baseada em encriptação assimétrica. É um sistema que integra tecnologia como encriptação, distribuição de informação, segurança de rede entre outros elementos. A um nível técnico, a Blockchain pode ser definida como uma tecnologia baseada em algoritmos de encriptação, tecnologias de armazenamento, rede ponto a ponto com definições como a impossibilidade de alterar dados armazenados, mecanismos aceites por todos os participantes e descentralização, [16].

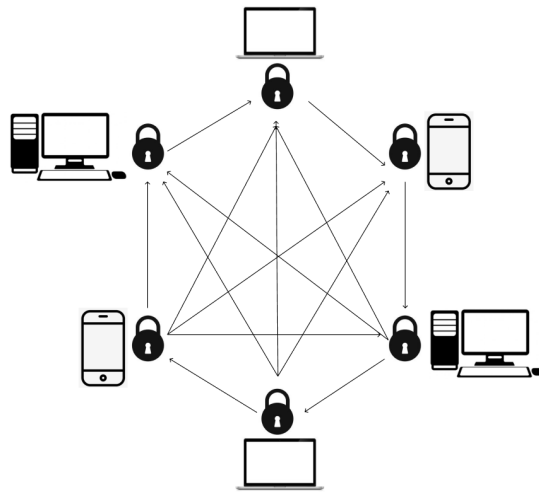


Figura 2.11: Exemplificação de troca de dados numa blockchain privada

O elemento central da Blockchain deverá de ser incapaz de impossibilitar os restantes elementos de realizar transações de informação, sendo que cada transação é verificada por mais de metade dos participantes presentes na rede, ou seja, a rede é baseada num sistema ponto a ponto, em que a informação é replicada por todos os nós e guardada por estes de forma a garantir uma transação, [16].

De um ponto de vista que visa a programação da rede, é possível ver a blockchain como uma base de dados em que as entradas de dados, blocos, são armazenados de forma sequencial, sendo que o bloco anterior ao bloco atual é chamado de «parent block», que poderá ser traduzido como bloco pai. Todos os blocos da cadeia, excetuando o inicial, possuem o respetivo «parent block». A ligação entre os blocos é assegurada por um «hash

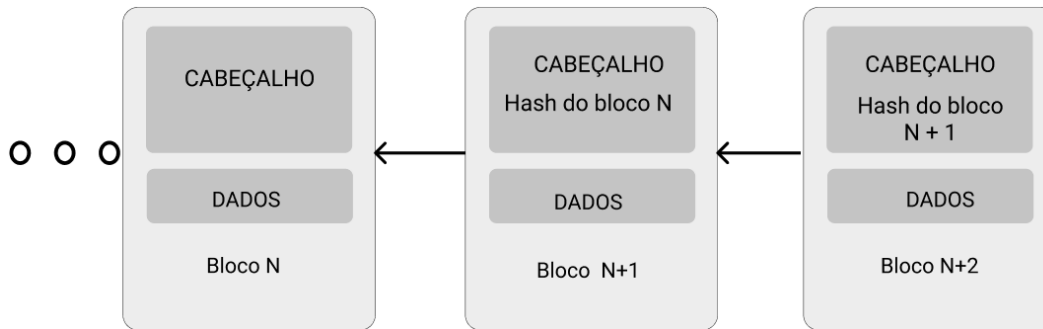


Figura 2.12: Organização entre blocos

value» do bloco anterior, sendo necessário a existência de um protocolo que defina como a rede irá operar conferindo direitos e deveres a cada nó. Deste modo é assegurado a comunicação, verificação e armazenamento de dados de forma segura e consensual por todos os intervenientes [17]. O formato dos blocos, bem como a sua ligação encontra-se exemplificado de maneira simplista na Figura 2.12. Cada bloco é composto por duas partes, a cabeça e o corpo do bloco. A cabeça, ou cabeçalho, contém informação sobre os dados contidos no corpo que nos casos mais comuns, como a Bitcoin assume dados como [18]:

- Versão do bloco: indica as regras que o bloco segue;
- Código hash do bloco pai: indica qual é o bloco anterior com recurso a encriptação 256-bit SHA256;
- Hash de origem: indica o código hash de todas as transações no contidas no bloco;
- «Timestamp»: data a que foi gerado em segundos desde 1970-01-01T00:00 UTC;
- nBits: código hash alvo representado num formato compacto que todos os novos blocos têm de cumprir;
- Nonce: é um espaço de 4 bits geralmente iniciado por 0 e que aumenta com o cálculo de cada novo valor hash.

2.6.1 «Miners»

Para assegurar que apenas transações válidas são autenticadas, e que esta ação não é realizada por uma entidade central, o que criaria um ponto central de falha, existe a necessidade de criar algoritmos que mantenham a integridade e segurança do sistema, dado que uma das dificuldades, passa pelo consenso entre todos os intervenientes, dado ser necessário a implementação em todos os processos [19].

- «Proof of Work» -ou «PoW», foi o primeiro algoritmo a ser desenvolvido por Nakamoto destinado a ser implementado na Bitcoin. Este algoritmo requer que cada nó da rede calcule o novo valor hash que terá de ser gerado para um novo bloco. Tal é conseguido através de sucessivas iterações até atingir um determinado patamar. Após isto, todos os restantes nós da rede devem confirmar o valor obtido. Os

nós que procederam ao cálculo do hash são denominados de «miners». Contudo, este algoritmo é bastante ineficiente devido a todos os nós processarem a mesma informação levando ao aparecimento de alternativas como o «Proof of Stake», [18].

- «Proof of Stake» - é um algoritmo que valida os blocos com base no princípio da fiabilidade de cada participante, por outras palavras, quem possuir controlo sobre a maior parte de informação é o menos provável de atacar a rede. A segurança da rede é obtida através de escolhas aleatórias de nós para a proceder à verificação da integridade e idade da informação. Contudo, este método possibilita que exista um nó na rede que seja dominante, todavia, existem cripto moedas que se encontram a analisar a possibilidade de trocar para este método, como o caso da Ethereum [18].
- «Delegated Proof of Stake» - é um método que se diferencia dos expostos em virtude que os nós presentes na rede procedem à criação do hash em conjunto, em vez de competirem por esta. A validação dos blocos é obtida por uma entidade escolhida pelos intervenientes da rede [20].
- «Proof of Authority» - «PoA», garante a validação das transações através de contas aprovadas previamente. Estas contas são as fornecedoras de informação aos nós. Este método centraliza a decisão num só nó, o que cria um ponto central de falha, o que também aproxima a rede a uma rede centralizada. Desta forma, este método apresenta-se como ideal para uma rede privada, [20].

2.6.2 Características da Blockchain

Numa rede de Blockchain há necessidade de cumprir vários aspetos característicos deste tipo de cadeia, os aspetos de maior importância serão discutidos de seguida. O primeiro destes aspetos é o **Anonimato**, sendo que numa cadeia de Blockchain isto significa que cada entidade participante possui uma entidade virtual, ou seja, um nome de utilizador, isto acontece para que a entidade real de cada participante não seja conhecida o que para alguns intervenientes poderia levar a problemas de privacidade. Exemplo deste aspeto é a Bitcoin, em que cada utilizador possui uma chave pública (entenda-se nome de utilizador) para realizar transações.

O aspeto que poderá ser apontado como um dos mais importantes é a **Descentralização**. Tal significa que não existe um nó central, instituição, e todos os nós têm poder equivalente. Este princípio é o pilar de cripto moedas como Bitcoin [16]. De seguida aborda-se a **Resistência à adulteração de dados**. Toda e qualquer transação de informação armazenada na blockchain não pode ser alterada durante a operação de transação ou após esta. A estrutura de armazenamento da informação é baseada na ligação de blocos que se encontram ligados entre os diferentes nós.

O seguinte ponto a analisar é a **Rastreabilidade**, numa rede de blockchain, rastreabilidade significa existir a possibilidade de encontrar informação sobre as fontes/elementos envolvidos na transação de informação na base de dados [16].

Por fim, os "*Smart Contract's*", em português poderá ser traduzido como contratos inteligentes, é uma versão melhorada de um contrato eletrónico, sendo uma parte de código que, quando vai de encontro às condições, é executado automaticamente, enquanto que em paralelo poderá receber e armazenar informação. Estes contratos foram

introduzidos nas redes de BlockChain para evitar que exista alteração das condições previamente acordadas no que toca ao processamento da informação bem como no caso das cripto moedas a recompensa de cada interveniente, por outras palavras, estes contratos declaram as obrigações, os benefícios e as penalidades dos envolvidos. Contudo a generalidade de cripto moedas não utiliza estes contratos, apontando como principal obstáculo a privacidade dos intervenientes na rede, que seria colocada em causa com o uso destes, tornando os dados dos intervenientes públicos [16].

2.6.3 Tipos de BlockChain

Apesar de existirem diferentes definições sobre a organização dos vários tipos de cadeia, a mais comum passa por dividir em três grupos, Pública, Privada ou Híbrida.

A implementação Pública, refere-se a cadeias de dados em que qualquer pessoa em qualquer parte do mundo, em qualquer altura pode aceder ao sistema, ler informação e realizar transações fidedignas. A tipologia Privada, passa por uma rede de dados em que a permissão para poder escrever é controlada por uma organização central. Esta organização é responsável por atribuir aos nós a permissão para participar na cadeia de dados. Este tipo de rede é geralmente utilizada em redes internas de aplicação prática. Por fim, a estrutura Híbrida apresenta-se como o meio termo das duas disposições apresentadas acima, dado que existe mais do que uma instituição com a capacidade de adicionar elementos à rede. A informação, contudo, pode ser escrita, lida ou acedida por todos os participantes e armazenada por todos estes, [18].

2.7 Prototipagem do HMI

Com o intuito de desenvolver interfaces gráficas que possam ser testadas e visualizadas mais rapidamente poderá ser encontrado no mercado softwares open-source que permitem de forma simples e rápida desenhar e testar a sua interação entre diversos dispositivos, permitindo assim prever problemas antes da conceção do código em si. Como tal, poderá ser encontrado softwares como o «Fidgma», ou «Invision studio», ou mesmo «BootStrap Studio», que apesar de não ser gratuito fornecerá acesso a estudantes.

2.8 Modelos de ataque

Um modelo de ataque é um conjunto de suposições sobre como os invasores podem interagir com uma função criptográfica, ou seja o que podem ou não podem fazer para alterar as condições de encriptação. Os objetivos de um modelo de ataque segundo o autor, [9], são os seguintes:

- Definir requisitos criptográficos que projetam as funções, para prever que atacantes e quais tipos de ataques se deve defender o sistema.
- Informar o utilizador se a função criptográfica está a ser utilizada de forma segura num determinado ambiente.
- Um ataque só pode ser considerado válido se for possível de executar no modelo considerado. Os modelos de ataque não necessitam de corresponder exatamente à

realidade, podem ser uma aproximação, desde que o modelo super estime as capacidades dos invasores, porque desta forma estará a antecipar técnicas de ataques. Um modelo de baixa qualidade subestima os invasores e fornece falsa confiança sobre uma função criptográfica, parecendo segura em teoria quando não é na realidade. Para ser útil em criptografia, os modelos de ataque devem pelo menos englobar o que os invasores podem realmente controlar e fazer para atacar uma encriptação.

2.9 Soluções de redes de Blockchain existentes no Mercado

Com a crescente procura por cripto moedas e redes de blockchain têm surgido no mercado cada vez mais soluções que procuram responder às necessidades dos utilizadores. Apresenta-se de seguida as três soluções com mais impacto neste mercado.

2.9.1 Hyperledger

A Hyperledger é uma comunidade pública de código aberto que se foca em desenvolver ferramentas e bibliotecas para implementação em redes de blockchain, representada na Figura 2.13. Possui ambientes de trabalho como o Hiperledger Fabric, Sawtooth, Indy e Caliper. A hyperledger tem uma abordagem modular aos seus projetos sendo que utiliza cada área de trabalho para contribuir, quando necessário, para o desenvolvimento de uma dada plataforma, ajudando assim no desenvolvimento da indústria sempre com base em «smart contracts» [21]. Assim, a hyperledger caracteriza-se por ser uma rede em que os participantes procuram a partilha de informação entre si com garantia de segurança. Esta rede não utiliza mecanismos como o proof of work, visto que a rede trabalha em conjunto para solucionar os problemas. A informação é partilhada por todos os nós da rede.

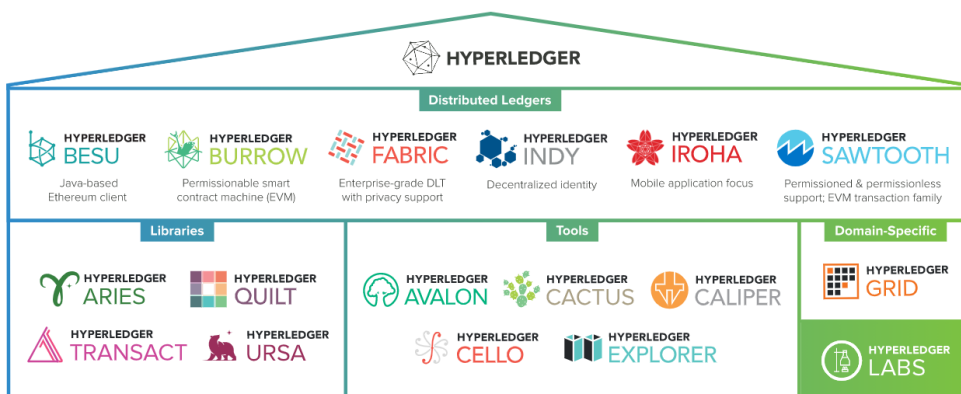


Figura 2.13: Hyperledger Green House [21]

Uma das vertentes é o Hiperledger Fabric que se apresenta como uma rede privada em que todos os membros têm de ser associados por uma entidade credenciada para tal. O Hiperledger Fabric oferece ainda a possibilidade de manter alguma da sua informação como privada, criando assim uma rede dentro da rede principal, ao que a Hiperledger chama de «canais». Tal permite, por exemplo, um revendedor colocar no mercado o

mesmo produto para diferentes consumidores sem estes terem acesso ao vendedor principal, [22].

2.9.2 Ethereum

Ethereum é um plataforma descentralizada com origem na Bitcoin, contudo, baseada em smart contract's que são executadas sem tempo de inatividade (menores que 15 segundos), censura, fraude ou intervenção por parte de terceiros. O principal objetivo desta plataforma é executar código de redes descentralizadas sem necessidade de saber o seu proprietário, esquema exemplificativo na imagem 2.14. Ao contrário da Hyperledger, a Ethereum tem por base o método Proof of Work [23].

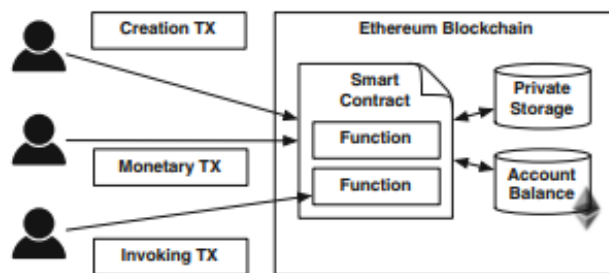


Figura 2.14: Ethereum [24]

No que toca aos Smart Contract's estes são programados em linguagem de alto nível, Solidity, e depois decompostos em código mais simples que é executado por uma máquina virtual. Ao executar um dado código, este tem um custo associado que posteriormente pode ser convertido no seu formato de cripto moeda, [24].

2.9.3 Corda

Corda apresenta-se uma vez mais como uma plataforma open source similar à Hyperledger, sendo que também partilha informação por determinados grupos. Este método uma vez mais, foca a privacidade e a escalabilidade da rede, uma vez que, a informação não está replicada por toda a rede. Assim sendo, apesar de uma plataforma distribuída, não aplica uma rede global a todos os nós [24]. Tem por princípio partilhar informação de uma transação apenas com os participantes que requerem informação sobre esta. Esta rede está mais focada no mercado financeiro, uma vez mais, com base em «smart contracts», exemplificados pela imagem 2.15.

2.10 Trabalho já realizado por outros

2.10.1 «Proposal of an automatic traceability system, in Industry 4.0»

O sistema de rastreabilidade proposto por Diogo Rocha tem por base a implementação de 4 leitores RFID para leitura de «smart objects». Deste modo, cada pallet contém uma tag RFID. Os leitores estariam em permanente funcionamento e à procura de uma tag. Assim que encontrem uma, irão encaminhar a informação para um ESP32 que a

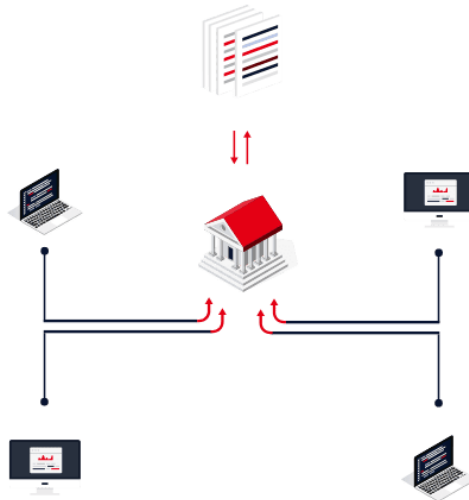


Figura 2.15: Corda
[25]

reencaminha para um computador com recurso à sua capacidade Wi-Fi e a páginas php. Por sua vez esta informação será armazenada em diferentes tabelas de uma base de dados e poderá ser acedida em qualquer dispositivo com recurso a uma interface gráfica. Esta interface permite ainda enviar informação de volta podendo esta ser armazenada nas tags RFID, [2].

Desta forma, começando pelo nível inferior, quando a unidade de processamento detetava um chip RFID, procedia a três testes para verificar se a mensagem lida era realmente verdadeira. Estes testes baseavam-se em analisar o tamanho das mensagens bem como se continha o bite de final de mensagem, de seguida era aplicado um método de controlo de mensagens, ou seja, calculando a soma de todos os bytes se respeitava verificação cíclica de redundância (CRC-8-CCITT). Ao passar aos três testes a tag irá ser enviada para o server. O ESP recorre aos seus dois cores para o processamento da informação, sendo que um é responsável pela leitura e o outro pelo envio das mensagens, estando em permanente troca de informação entre eles. A informação a chegar ao server tem 5 páginas php à sua espera, sendo responsáveis por tratar a informação que é suposto armazenar permanentemente da base de dados, [2]. A implementação desenvolvida para recolha de informação e transmissão é exemplificado pela imagem 2.16.

Relativamente à base de dados, foi utilizado phpMyAdmin para a sua construção existindo uma tabela principal que inclui as chaves primárias de todas as outras sete tabelas bem como informação sobre chegada e partida de produtos. Para desenvolver a interface gráfica recorreram ao Bootstrap, que inclui HTML e CSS. Para a criação dos gráficos utilizou Highcharts que se baseia em JavaScript.

O que ficou a faltar

As limitações apontadas no final da dissertação passam por:

- Não foi testado com um grande número de equipamentos;
- Aplicação real – dificuldade do RFID funcionar com o meio metálico;

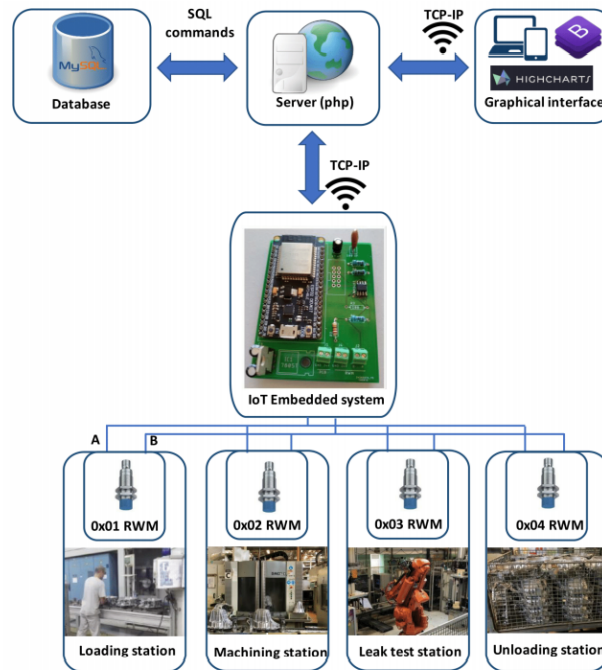


Figura 2.16: Implementação de sistema de rastreabilidade proposto por Diogo Rocha [2]

- Extensão de dispositivos de leitura, apenas suporta 32 em barramento;
- Tempo de leitura de tag's RFID muito extenso, visto analisar bit a bit e não os bits totais de uma vez;
- Custo muito elevado do RFID para a implementação procurada.

2.10.2 «An application of blockchain "smart contracts" and IoT in logistics»

O sistema de blockchain desenvolvido pela Leonor Augusto nesta implementação tem por base um sistema da Ethereum, representado pela Figura 2.17. Para esta escolha são apontadas razões como a disponibilidade de ferramentas, documentos e uma comunidade extensa; o suporte de «smart contract's» de elevada complexidade; ferramentas de desenvolvimento para aplicações de front-end;

No que toca ao patamar dos «smart contract's», nesta implementação foi utilizado uma tipologia do tipo «inheritance», disponibilizada pela Ethereum e que assenta quase num princípio de hierarquia, ou seja, cada elemento da blockchain só poderá realizar leitura do seu sucessor. Este método veio ainda com o ponto negativo de que alguns elementos, dado a estrutura de informação que acarretam, possuem uma maior complexidade no seu código levando a problemas de armazenamento, [26].

No que toca ao armazenamento de informação foi utilizado uma tipologia de mapa em vez de array, que apesar de ser mais complexo no seu armazenamento, futuramente permite tratar a informação de maneira mais conveniente, podendo assim, ser criado o

percurso de um produto de maneira mais acessível. Para elaborar o patamar de front end foram utilizadas ferramentas de «open-source» como http-server, web3.js, Bootstrap,JS e Morris.js bem como Metamask para realizar as credenciais de acesso ao sistema. No patamar inferior da implementação foi utilizado um arduino Uno com recurso a um sensor de temperatura SHT31. A implementação chegou a um funcionamento correto, com o

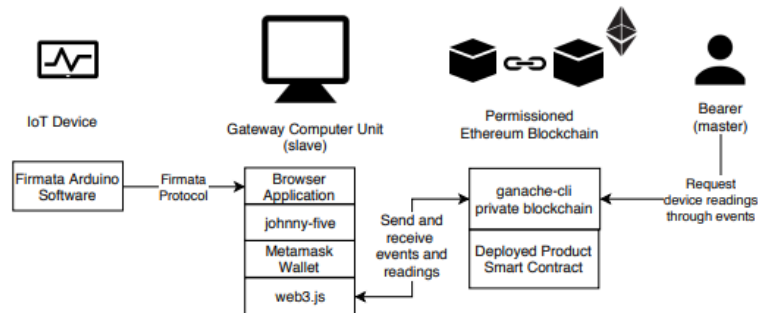


Figura 2.17: Explicação da implementação proposta por Leonor Augusto [26]

armazenamento a aumentar também de forma linear como seria de esperar. Contudo, o armazenamento de informação ocorreu de forma ineficiente devido a uma incorreta utilização dos blocos da blockchain, estando assim a criar um bloco para cada transação e não a armazenar consecutivas transações num só bloco. Outro inconveniente passa pelo uso da Metamask, que levou à necessidade de confirmar cada transação efetuada, [26]. Como pontos em falta podemos apontar o número de equipamentos utilizados, bem como a falta de diversidade destes, bem como o problema de código de nós específicos.

2.10.3 «Security techniques for the Internet of Things»

A análise executada pelo José Mendes nesta dissertação procura analisar o desempenho de técnicas de encriptação em dispositivos IoT, nomeadamente ESP32, sobre os quais esta dissertação também irá trabalhar. Mais especificamente consistiu no estudo e discussão da segurança em dispositivos IoT com foco na segurança criptográfica e avaliação da coexistência de mensagens e recursos. Para isso, recorreu a uma biblioteca criptográfica desenvolvida em C consistindo em 20 implementações diferentes e, adequado para uso em cenários heterogêneos de dispositivos IoT. O estudo teve por base uma implementação em C++ CoAP minimalista, permitindo a implantação em vários cenários com diversas camadas de comunicação não sendo restritas a um cenário particular. Este trabalho estudo assim o desempenho de métodos criptográficos comparando-os com o método AES. Pode assim concluir que algoritmos como SOSEMANUK, HC-128 e Rabbit obtiveram desempenho ligeiramente superior, contudo, estes métodos apresentam-se como menos conhecidos pela comunidade. Em contra partido algoritmos como o RC5 e XTEA apresentam desempenho inferior. Através da imagem 2.18 é possível analisar em mais detalhe os resultados obtidos por cada algoritmo. Este trabalho procurou ainda analisar o desempenho de funções HMAC, apontando estas funções como obsoletas quando aplicadas aos dispositivos IoT, recomendando ainda que a autenticação deve ser oferecida por outras técnicas, [27].

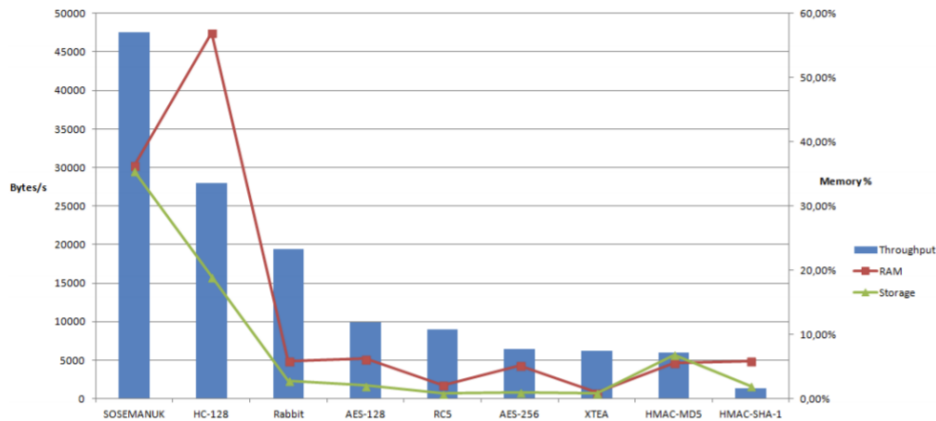


Figura 2.18: Security techniques for the Internet of Things - Performance de algoritmos estudados por José Mendes, [27]

2.10.4 «Authentication in interactions with IoT devices»

A dissertação realizada por João Ramos começa por fornecer uma visão geral de conceitos bem como descrevendo alguns dos principais problemas que estão inerentes a equipamentos IoT, abordando assim problemas inerentes às comunicações mas também as vulnerabilidades de ter o equipamento exposto a entidades não desejadas, bem como algumas sugestões analisadas para combater estes problemas. Nesta área começa por apontar como principal problema a possibilidade de leitura de dados através da porta UART de cada dispositivo, ou seja, através de uma leitura física. Com acesso ao meio aponta ainda como principal problema, dado que através de leituras eletromagnéticas e de consumo de energia dos equipamentos poderá levar a maior controlo sobre a informação que o dispositivo está a processar. Como principal solução para este problema físico, aponta o controlo do meio como a principal solução, ou seja, restringir o acesso ao meio apenas por entidades credenciadas para tal. Para combater o acesso por equipamentos não desejados, utiliza uma gateway para realizar a ponta de comunicação entre dispositivos internos de recolha de dados e equipamentos externos destinados a visualizar e armazenar estes dados. A gateway realiza assim uma filtragem à recessão de dados a equipamentos que contenham apenas um ip interno e já conhecido previamente. É de notar que os equipamentos utilizados foram programados em C++. Quanto a métodos criptográficos na solução desenvolvida recorre a criptografia assimétrica, contudo aponta como defeito a este mesmo método ser computacionalmente dispendioso e possivelmente não suportado por todos os equipamentos. Para tal, procurou utilizar este formato de encriptação apenas para estabelecer sessões seguras e após isto recorre a métodos criptográficos menos dispendiosos como AES-CBC. Por fim, aponta como principal melhoria a implementação de um algoritmo mais eficiente que o RSA, bem como a implementação de um protocolo que permita renovar sessões de comunicação, [28].

2.10.5 «Proposal of a Traceability 4.0 System in Renault CACIA Using Blockchain Technologies»

A implementação proposta por Diogo Costa assenta em plataformas «open source», e procura desenvolver um sistema maioritariamente de raiz recorrendo a tecnologias e linguagens de programação que, de momento, se apresentam com grande crescimento e utilização. Procura ainda ser possível realizar uma verificação de dados entre a base de dados empresarial e a blockchain, sendo que a blockchain é do tipo privada. A implementação aqui apresentada, representada na Figura 2.19, encontra-se muito bem estratificada pelo que assim iremos descrever cada patamar desta:

Sensing layer - a função principal passa por proceder à recolha de dados, existem três estações, compostas respetivamente por um Arduino Uno, um ESP32 e um esp8266. Ao longo das estações, sempre que chega uma peça nova, é recolhido informação como humidade, temperatura e luz. Dado que a comunicação da estação 2 e 3 é por wireless, utilizou AES CBC para realizar a encriptação das mensagens. Desta forma foi criado um UUID para cada elemento da comunicação sendo que este UUID é encriptado e posteriormente é gerado a hash-value a partir dele, sendo posteriormente transmitido nas comunicações.

Network and Security Layer - é importante referir que neste campo foi ainda introduzido um «Certification Authority» para validar todos os participantes da rede, para uma vez mais, garantir que não existe elementos estranhos à rede. Foi ainda utilizado o típico método de autenticação, através de um User e de uma password.

Gateway - esta secção é responsável por recolher os dados vindos da IOT e transmiti-los de forma normalizada para os níveis superiores, necessitando de uma maior capacidade de processamento, como tal, recorreu a um Raspberry PI 3 em ambiente de Ubuntu Server.

Blockchain and Storage Layer - referente ao armazenamento da informação, recorreu ao MongoDB, uma base de dados não relacional atribuindo assim mais liberdade no armazenamento bem como à estruturada da informação e ao seu acesso. Para garantir a integridade desta, foi ainda introduzido um certificado de autoridade bem como comunicações com recurso ao protocolo TLS.

User Layer - para a criação do HMI recorreu a HTML como base para das páginas Web, sendo que recorreu a JavaScript/Ajax para proceder ao display da informação. A disposição gráfica foi auxiliada por CSS sendo a sua amostragem baseada no Google Charts API.

Para proceder à implementação real, é sugerido a utilização de um servidor do tipo comercial, contudo, deverá de ser também implementado um servidor assíncrono para gerir múltiplas conexões. Outra sugestão deixada, é a implementação de tecnologia quântica, para tornar o projeto seguro a longo prazo, [29]. Poderá ainda ser apontado o baixo número de equipamentos utilizado, bem como a incorporação dos equipamentos todos na mesma rede, ou ainda a utilização de um baixo número de gateways.

2.11 Conclusões retiradas dos trabalhos anteriores

Ao analisar o conjunto dos trabalhos desenvolvidos pode-se apontar algumas áreas de melhoria comuns a todas as implementações, tais como, a performance, em que todos os trabalhos anteriores sofriam de problemas de performance, sejam eles no campo do

armazenamento de dados, de leitura de informação ou troca desta. Por outro lado o estudo sobre os algoritmos de encriptação mostrou-se bastante interessante para a implementação a desenvolver.

Como segundo ponto de falha pode se apontar a diversidade e falta de testes. Os projetos apresentados sofriam todos da pouca diversidade de equipamentos bem como de que nenhum foi testado em ambiente empresarial por longos períodos de tempo.

As plataformas «open source» demonstram que possuem capacidade para acarretar o desenvolvimento do projeto pretendido. Por fim, é clara a necessidade de flexibilidade e escalabilidade do sistema a desenvolver, para que este seja facilmente adaptável às mais diversas situações.

2.12 Tipologia escolhida e pontos a melhorar

Dentro das soluções disponíveis no mercado e através da análise de trabalhos já desenvolvidos, a implementação que mais se aproxima ao trabalho que se pretende desenvolver é a proposta por Diogo Costa, como tal será esta que tomaremos como ponto de partida. Ao analisar com mais atenção esta implementação, é de notar que não existe qualquer armazenamento de dados intermediário, ou seja, caso exista algum problema de comunicação entre as gateways e os nós estes dados simplesmente são perdidos.

Outra questão que não foi avaliada, consiste na forma como os dados estão a ser encriptados entre o patamar de recolha de informação e a gateway. A capacidade dos ESP para gerar números aleatórios que realmente possuam um grau de aleatoriedade é completamente desconhecida o que torna a implementação suscetível a ataques de «Snooping» que poderão levar a ataques de modificação de informação ou mesmo personificação, como explicado na secção 2.4.1. Este patamar torna-se ainda mais vulnerável pela não renovação das chaves privadas. Ou seja, se um interveniente indesejado conseguir aceder à informação uma vez não existe mecanismo nenhum que faça com que este perca o controle sobre a comunicação, tornando assim todo o processo de armazenamento de dados na Blockchain pouco credível.

No que toca à interface gráfica desenvolvida nesta implementação, esta encontra-se bastante completa, contudo, a sua navegação poderá ser simplificada com o intuito de a tornar mais intuitiva. Outro ponto a analisar é o seu formato estático, ou seja, foi desenhada apenas a pensar num ecrã de computador portátil, não tendo adaptação a ecrãs de outras dimensões, sejam ele um LCD de maior dimensão ou aos vários tipos de smartphone onde se encontram diversos tamanhos de ecrã. Ao analisar os outros trabalhos aqui descritos importa salientar o uso do algoritmo AES CBC dado ser um bom compromisso de desempenho e de memória RAM utilizada, como descrito na dissertação "Security techniques for the Internet of Things", por outro lado, a divisão entre comunicações de equipamentos internos e externos bem como a tipologia de comunicações, implementada na dissertação "Authentication in interactions with IoT devices", mostra-se como ponto de interesse ao projeto a desenvolver.

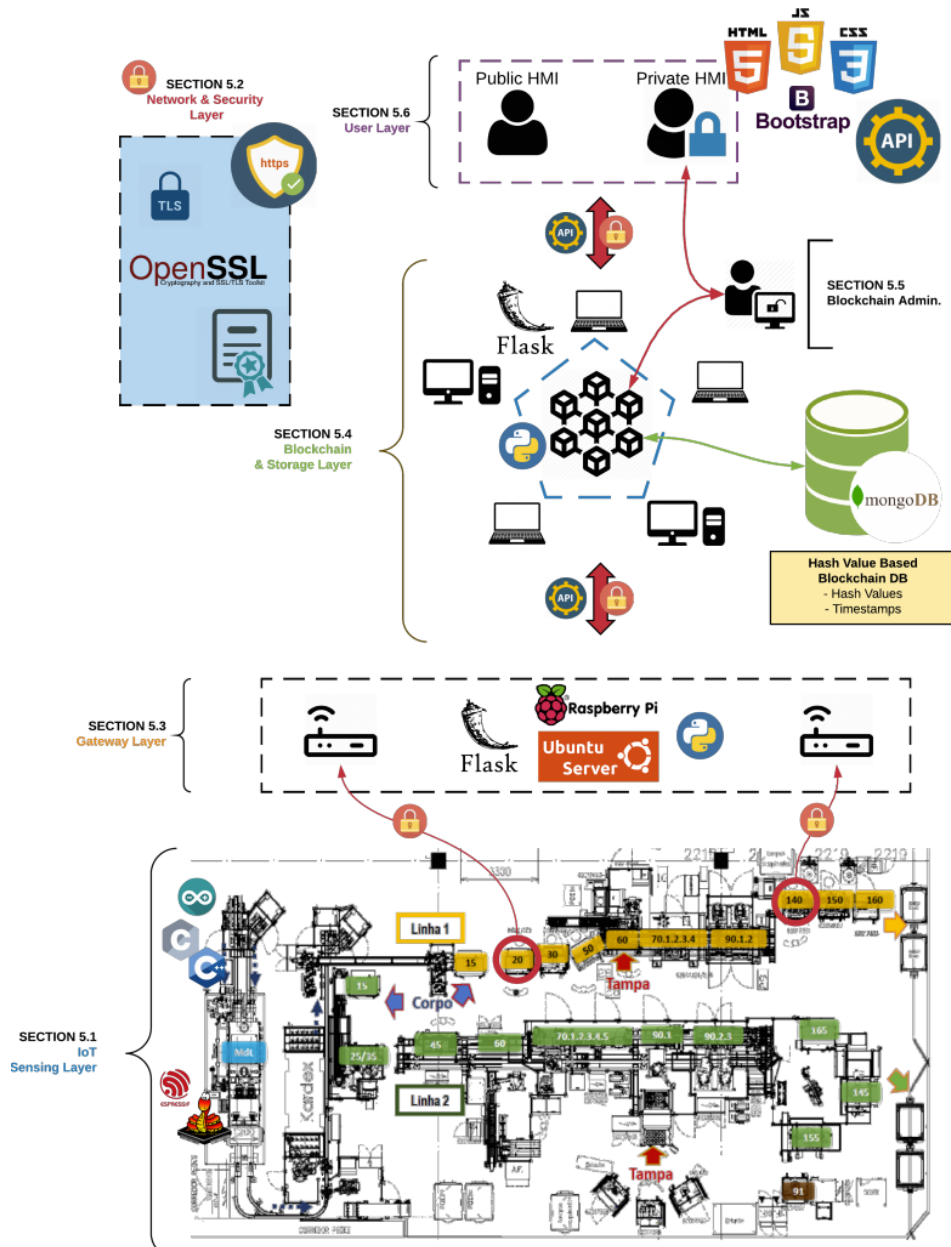


Figura 2.19: Proposal of a Traceability 4.0 System in Renault CA-CIA Using Blockchain Technologies», por Diogo Costa

[29]

Capítulo 3

Solução Proposta

No seguimento do que foi encontrado em trabalhos anteriores, para realizar uma implementação que seja modular e facilmente escalável tanto verticalmente, como horizontalmente, tendo sempre em vista a implementação direta à linha de bombas de óleo, a arquitetura proposta foi pensada de forma estratificada em vários níveis, como se pode ver na Figura 3.1. Deste modo iremos analisar cada patamar sugerido nas subsecções seguintes partindo do nível inferior.

3.1 Atacante e tipos de ataques

Com o intuito de desenvolver um sistema que possa ser considerado seguro, é necessário perceber quais são os ataques que a implementação poderá estar mais susceptível. Deste modo, o modelo de ataque poderá ser dividido em dois tipos, físicos e computacionais. Quanto aos físicos, percebe-se ataques em que existe direto acesso aos equipamentos, tem de se ter em conta que esta implementação se destina a uma implementação fabril em que apenas existe pessoal autorizado, bem como a possibilidade de limitar o acesso a todos os funcionários, foi considerado que não existe a possibilidade de acesso direto aos equipamentos bem como aos sensores, seja, danificação dos sensores ou dos equipamentos responsáveis pela transmissão da informação ou ainda leitura do código que cada equipamento possua. Quanto a ataques computacionais considerou-se que os mais prováveis seriam do tipo "ciphertext only", aplicados à transmissão das mensagens, em que o atacante executa ataques por métodos de força bruta não conseguindo alterar, ou saber partes do texto que está a ser encriptado. Os ataques podem ainda ser do tipo de personificação destinados à base de dados, sejam eles de armazenamento de dados indevidos ou alteração dos existentes, como tal será necessário garantir que as comunicações são executadas apenas com os equipamentos desejados. Como os dispositivos IoT são o ponto de maior debilidade, estes apenas poderão comunicar com a gateway e esta terá de ter também as suas comunicações restritas, criando uma clara divisão entre comunicação numa rede local e equipamentos exteriores.

3.2 Recolha de informação e IoT

O patamar mais baixo desta implementação, representado na Figura 3.1, deverá ser capaz de recolher informação sobre as condições de cada peça aquando da sua entrada na linha,

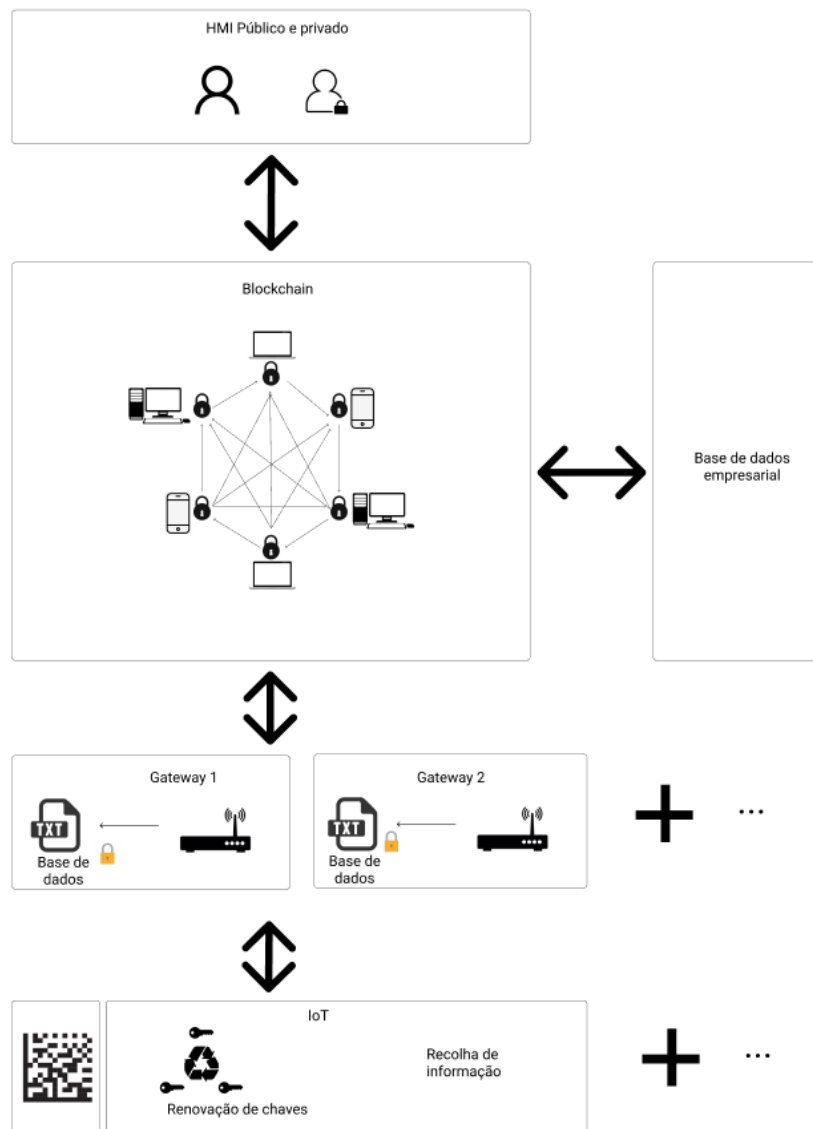


Figura 3.1: Esquema da arquitetura proposta

quando passa por uma dada estação ou ainda em que condições o trabalho sobre a peça está a ser realizado. Todo este processo tem de ser realizado de forma ininterrupta, possibilitando a caracterização do processo o mais completo possível.

Relativamente aos módulos IoT estes deveram de tratar e enviar a informação recolhida pelos sensores para um dispositivo remoto através de um método seguro, juntamente à identificação do próprio dispositivo, recorrendo a um identificador único (ID). Dado que se pretende recolher informação variada ao longo do processo, será importante que estes dispositivos sejam de reduzidas dimensões para que possa ser adaptado a qualquer espaço existente, possuir ligações suficientes para permitir a conexão a diversos sensores e ainda ser portadores comunicação Wi-Fi com o intuito de reduzir a complexidade da instalação física. Para além do tipo de ligação, estes dispositivos deverão ainda possuir memória interna programável para o caso de troca de informação em períodos de tempo

diferentes bem como a possibilidade de implementar protocolos de segurança.

Como **primeiras melhorias** à implementação proposta por Diogo Costa, pretende-se solucionar a questão da renovação das chaves privadas utilizadas na encriptação da informação por parte dos ESP bem como analisar se o IV produzido por estes equipamentos possui um grau de segurança elevado, assunto que será abordado mais a frente na seção 3.8 e posteriormente no capítulo seguinte. Pretende-se ainda alargar a informação recolhida, entenda-se também diversidade desta, bem como o números de equipamentos utilizados.

3.3 Gateway

Este patamar intermediário à IoT e à Blockchain, representado na Figura 3.1 como o segundo patamar, terá de ser capaz de estabelecer a ponte entre os módulos IoT e a Blockchain. Este sistema terá de estabelecer comunicações de forma segura uma vez mais, bem como de rapidez elevada pois poderá ser um ponto gargalo da implementação. Este patamar da implementação ao estabelecer ligação com equipamentos exteriores à própria rede fica mais exposto a ataques de terceiros, desta forma, para proceder à transmissão de informação deverá-se recorrer a protocolos HTTP complementados com protocolos SSL/TLS, ou seja, HTTP's, permitindo assim, comunicação segura e rápida. Desta forma, será necessário que estes equipamentos apresentem uma vez mais comunicação Wireless mas desta vez que sejam também eles portadores de uma interface que permita executar mais do que um processo simultaneamente bem como monitorizar o seu desenvolvimento.

Base de dados intermédia

A gateway deverá ainda ter capacidade suficiente para armazenar a principal informação que lhe seja enviada pelo patamar da Iot, de forma a assegurar uma base de dados extra, que apenas servirá, caso seja necessário, ser possível analisar de forma manual a existência de alguma adulteração de informação entre a gateway e a própria blockchain.

A informação contida nesta base de dados deverá de estar encriptada, e esta encriptação deverá de recorrer a um método de renovação de chaves privadas. Outro ponto a ter em conta é apenas conceder acesso a utilizadores credenciados para tal através de uma palavra passe. O acesso a esta informação em primeira instância deverá de ser feito de forma física.

3.4 Blockchain

Analisando a implementação desejada será de extrema importância fazer o armazenamento de informação para que seja possível consultá-la posteriormente. Para além disto, é necessário garantir que esta informação não foi adulterada desde o seu armazenamento por terceiros, ou em último caso, quando tal aconteça, que seja possível saber que a informação não é a correta. Portanto, com o intuito de manter o espaço em disco utilizado reduzido, na blockchain, representada na Figura 3.1 como o terceiro patamar da implementação, apenas será armazenado o hash-value de cada transacção, permitindo posteriormente validar uma dada informação. É de lembrar que toda a informação é armazenada por mais do que um nó, sendo de extrema importância manter o número de bytes reduzido por cada transacção. Em suma, este patamar tem de ser responsável por

validar a informação e armazená-la. Poderá ainda ser necessário adicionar sensores, que, sendo um equipamento físico, teriam de ser adicionados à gateway de forma inteiramente manual aquando de uma expansão. Por outro lado, é contemplado a hipótese de através da interface com o utilizador, ser permitido a um utilizador gestor adicionar ou remover gateways, nós e ainda contas de clientes. Esta ação será processada por um nó central, explicado na secção abaixo.

3.4.1 Nó central

Apesar da tipologia blockchain usualmente não recorrer a um nó central, como o intuito é cooperação de igual forma para obter a melhor performance comum, a implementação contempla um nó central responsável apenas por controlar o fluxo de mensagens entre todos os nós, os elementos físicos pertencentes à rede, entenda-se as gateways, bem como os nós que processam a informação ou os utilizadores que têm acesso à interface gráfica. Apesar de existir um nó central, este não possui nenhum poder extra sobre a informação que circula na cadeia de dados, tendo acesso apenas aos blocos de dados.

3.4.2 Nós «miners»

Os nós que se apresentam como responsáveis por criar blocos de informação novos, «miners», não deveram requerer manutenção, sendo que devem de ser modulares, de forma a poder ser feita uma expansibilidade ou contração da rede de forma simples. No que toca à sincronização, terá de ser feita por estes de forma completamente autónoma. A informação a tratar é armazenada de duas formas, uma na RAM para que seja possível aceder de forma mais rápida, e outra, na base de dados auxiliar para backup, garantindo maior rapidez ao sistema prevenindo ainda a perda de informação por falha física do mesmo.

O processo de adição de novos blocos deverá de se basear no «proof of authority», e o processo de tratamento de informação tem por base uma ordem pré acordada entre os nós, ou seja, nenhum nó consegue tratar dois blocos de informação seguidos, e ainda o nó que está em descanso à mais tempo é o nó que vai trabalhar de seguida. Esquema exemplificativo na Figura 3.2.

Com o intuito de garantir que a informação é tratada de forma rápida e fluída, ou seja, sem conflitos entre os nós, aquando da adição de um nó à rede, todos os nós existente tem de acordar uma nova ordem pela qual vão tratar a informação. Ou seja, pela qual vai acontecer o tratamento dos blocos. Desta forma garante-se que os dados são tratados de forma mais fluída ao longo da cadeia.

3.4.3 Tipologia de blocos

Cada bloco de informação será dividido em duas partes. A primeira, o cabeçalho, será responsável por conter o código hash da sua transação e o «timestamp», ou seja, data e hora de quando foi criado. Por sua vez, a segunda parte, o corpo do bloco, será responsável por conter o número do bloco, o código hash da transação correspondente ao bloco anterior, assegurando a continuidade na cadeia, o id do nó que tratou esta mesma informação e por fim a informação em si. Esta informação passa pelos dados recolhidos, dito de outra forma, a que peça correspondem, por quem foram recolhidos e quando foram recolhidos. Tal é exemplificado na Figura 3.3.

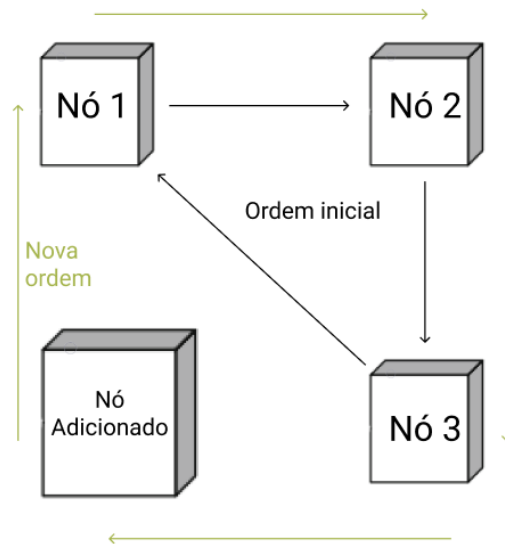


Figura 3.2: Sequência de trabalho dos nós

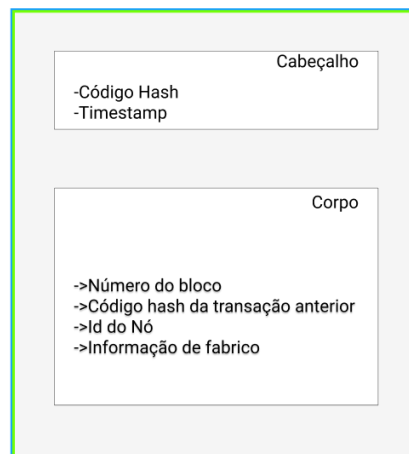


Figura 3.3: Estrutura dos blocos de dados

3.5 Base de dados

Com o intuito de manter a flexibilidade da implementação, será utilizada uma base de dados não relacional que, devido à sua não utilização de relações entre dados, confere uma maior flexibilidade à procura de informação, contudo acarreta problemas aquando da sua pesquisa, pois poderá ser obtida mais informação do que a necessária. Idealmente serão necessárias três coleções de dados, uma responsável pela informação da blockchain, outra pela backup da informação referente aos utilizadores e a terceira responsável pelos dados dos intervenientes na rede, entenda-se nós e gateways.

A implementação deverá contemplar ainda o envio de dados para a base de dados empresarial, ou seja, em conjunto com o código referente à peça que tenha dado entrada na linha, é enviado para a base de dados empresarial um conjunto de códigos hash correspondente aos dados armazenados também na blockchain.

3.6 Segurança de informação

No que toca à segurança da informação, apesar de ser de extrema importância, este não se apresenta como um patamar sugerido, devido a ser comum a todos os níveis. Primeiramente, com o intuito de contemplar a troca de dados de forma segura, serão implementadas ligações do tipo HTTP's, que por sua vez se baseiam em SSL/TLS, sendo necessário a criação de uma entidade certificada destinada à criação dos protocolos para todos os elementos envolvidos na comunicação. Este tipo de certificados apenas é válido por um dado período de tempo, podendo ser necessário proceder a uma actualização à posteriori.

No que toca à base de dados presente na gateway, esta deverá de contemplar formas de encriptação de informação bem como mecanismos que bloqueiem outros utilizadores, não desejados, de a adulterar, seja alterar a informação ou eliminá-la. Para prevenir ataques por força bruta deverá ainda contemplar mecanismos fidedignos de renovação das chaves utilizadas na encriptação.

3.7 HMI

Para fazer o controlo instantâneo de todo o processo será necessário desenvolver um HMI capaz de fornecer todos os dados que sejam considerados importante ao processo. Como tal, por decisão proveniente da empresa, deverá existir dois tipos de utilizadores, o primeiro da tipologia privada, em que apenas podem aceder desde a rede interna, podendo visualizar toda a informação bem como alterar os nós em trabalho ou as gateways. O segundo tipo de utilizador será público, tendo apenas acesso a uma quantidade restrita de informação, no entanto, poderão aceder a partir de qualquer localização. Assim, os utilizadores privados, poderão ser departamentos da empresa ou funcionários, e os públicos fornecedores ou consumidor final por exemplo.

3.8 Análise aos gerados do ESP e Raspberry Pi

Com o intuito de analisar se a comunicação entre o Raspberry Pi e os módulos IoT do patamar inferior era executada de forma segura, surgiu a questão sobre a capacidades

destes módulos fornecerem números que são de facto aleatórios. Esta questão surge dado que um número para poder ser gerado de forma aleatória o equipamento tem de ter fontes de entropia de qualidade e utilizá-las de forma correta, dado que existem normas pelas quais os geradores têm de se regular, como por exemplo a NIST SP800-90B, [13], que fornece métodos para regular as fontes de entropia de forma correta. A escolha sobre os testes propostos pelo NIST advém de ser um instituto de renome internacional com provas dadas na área de análise de números.

A entropia que o gerador utiliza depende também do sistema operativo. Em Linux o sistema baseia-se apenas nos tempos de interação e de dados que o computador recolhe por movimentos do rato e do teclado, [30]. Já no Windows existe maior variedade no que toca a fontes de entropia sendo que estas derivam de micro processadores que alguns computadores têm ou de processos integrados nos próprios processadores, [31], [32].

Contudo desconhece-se a fonte de entropia do ESP quando está a funcionar com firmware baseado em Micropython, bem como a qualidade dos números gerados pelo Raspberry Pi. Propõem-se assim realizar os seguintes testes:

- The Collision Estimate
- The Markov Estimate
- The Compression Estimate
- The Most Common Value Estimate
- The t-Tuple Estimate
- The Longest Repeated Substring (LRS) Estimate
- The Multi Most Common in Window Prediction Estimate
- The Lag Prediction Estimate
- The MultiMMC Prediction Estimate
- The LZ78Y Prediction Estimate

O método de execução dos testes encontra-se descrito na Figura 3.4. Em suma, é suposto um valor médio de entropia e a partir da execução do teste IID verifica-se se o tipo de dados é de facto IID e elabora-se os testes do tipo Restart obtendo a entropia estimada atualizada e de seguida a entropia mínima. Estes testes foram formulados para garantir que:

- As fontes de entropia fornecem sempre dados com a mesma distribuição após reiniciar o processo;
- A distribuição de amostras numa sequência sucessivas operações de reiniciar é independente de quando acontece o reiniciar;
- A informação sobre reiniciar uma sequência não interfere de forma positiva com a sequência seguinte.

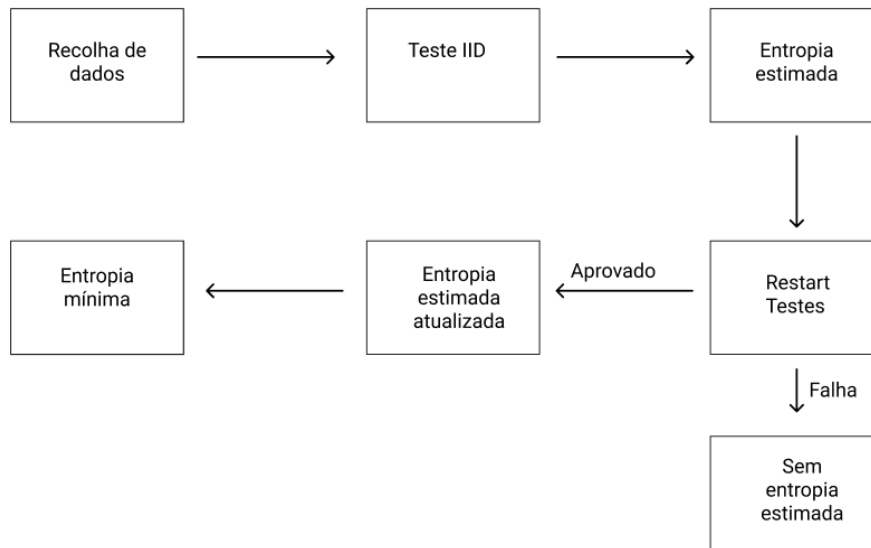


Figura 3.4: Processo de execução de testes

3.9 Pontos a reter

Ao projetar toda a implementação é visível a necessidade de assegurar o correto funcionamento dos testes para que toda a recolha de informação e respetiva transmissão ocorra baseada em comunicações verdadeiramente seguras. É ainda possível, apontar que todo o desenvolvimento se deve basear, uma vez mais, em softwares open source e de larga utilização, com o intuito de utilizar tecnologia já testada e sem custo acrescido. Por outro lado, a importância da estratificação da implementação facilita tanto o pensamento crítico sobre o trabalho a desenvolver, bem como a identificação da tecnologia a utilizar e as ações que esta acarretará.

Capítulo 4

Implementação

Partindo da proposta anterior procurou-se realizar a implementação visível na Figura 4.1. A implementação desenvolvida mantém a estratificação inicial, sendo que o patamar mais baixo da IoT é analisado na secção 4.2, onde serão abordados todos os elementos que constituem este patamar, bem como aspetos da sua programação. Já na secção 4.3 aborda-se aspetos referentes de hardware e programação segundo patamar referente às gateways. De seguida é abordado o terceiro patamar da implementação com recurso à secção 4.7 e por fim a amostragem dos dados pela secção 4.8.

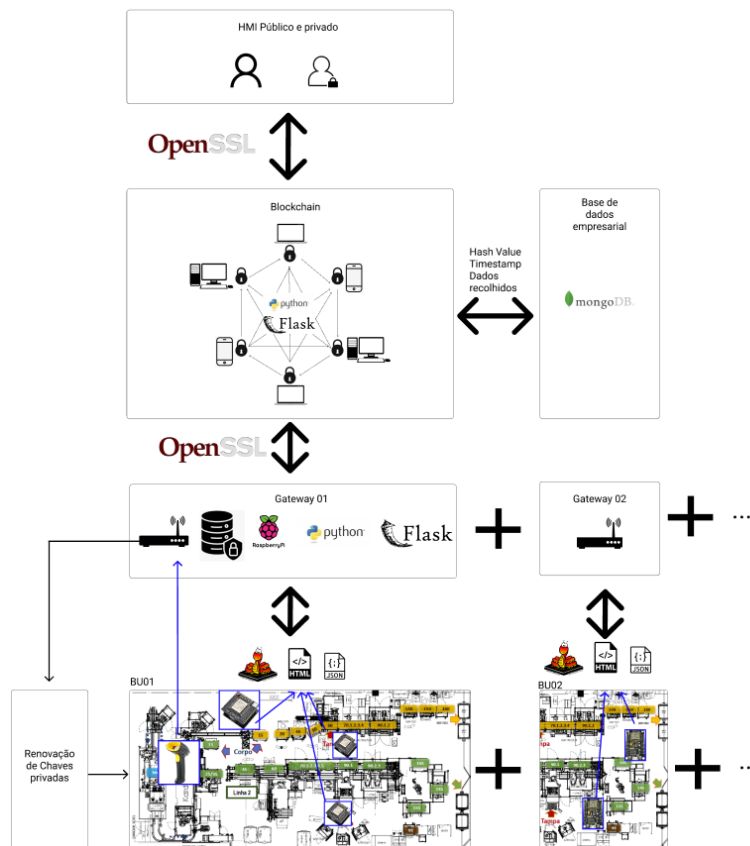


Figura 4.1: Esquema da Arquitetura proposta

4.1 Testes executados aos números aleatórios

Com a finalidade de desenvolver a arquitetura já apresentada, o trabalho desenvolvido parte do patamar mais baixo da implementação, a IoT, e segue o percurso que a própria informação percorre ao longo de toda a cadeia. Como tal, em primeira instância pretende-se avaliar a aleatoriedade dos números gerados pelos dispositivos IoT, recorreu-se, então, aos testes propostos pelo NIST, sendo que será explicado abaixo os que foram considerados mais importantes.

O teste «The Most Common Value Estimate» encontra a percentagem p do valor mais comum de entrada do dataset e constrói um intervalo de confiança para esta percentagem. Após isto, calcula o limite superior da probabilidade do valor de p_u mais comum, sendo que após isto estima o valor mínimo de entropia através de $\log_2(p_u)$, [14].

O teste «The Collision Estimate» mede o número médio de amostras até à primeira colisão, ou seja, até que encontra um valor repetido. Estimando a probabilidade do valor seguinte baseado no número de vezes que obteve colisões. Após estas operações, calcula a média da amostra e o seu desvio, com o intuito de obter o limite inferior do intervalo de confiança. O estudo é feito através de uma procura binária de $\log_2(p)$, [14].

O valor seguinte de uma amostra depende apenas dos valores anteriores n . A análise The Markov Estimate executa esta estimativa, sendo que quantifica a dependência entre consecutivos valores de entrada, baseados na entropia presente nos valores de saída. Desta forma é possível determinar a probabilidade inicial para um valor de saída inicial baseado na maior probabilidade dos valores de saída. Assim a entropia mínima é calculada por $\log_2 p_{max}$, sendo p_{max} o valor máximo de probabilidade calculado, [14].

O último teste a referir é o «The Compression Estimate», sendo responsável por calcular o rácio de entropia, baseado no quanto os dados obtidos podem ser comprimidos. A estimativa é calculada gerando um dicionário de valores e calculando o número médio de vezes que é necessário calcular para obter um valor de saída. A amostra, primeiro é dividida em dois grupos: o primeiro serve como dicionário e o segundo é utilizado como grupo de teste. Os valores de compressão são calculados sobre o grupo de testes, de forma a determinar a média da amostra e o desvio padrão. De seguida para calcular a entropia mínima $\log_2(p)/b$ sendo b o número de blocos de bit sobrepostos, [14].

4.1.1 Processo de recolha de dados

Para proceder aos testes, foram retiradas vários tipos de amostra. A primeira amostragem consiste em mil bytes, ou seja mil números aleatórios de oito bits cada, obtidos sem tempo de espera entre eles. Esta amostra serve para executar testes de verificação, ou seja verificar se os dados recolhidos são válidos e a sua recolha se procedeu da forma correta e se será possível executar a análise por completo.

A segunda amostragem baseou-se em mil ficheiros, cada um constituído por mil números, ou seja mil bytes e cada ficheiro obtido com um tempo de inatividade de quinze segundos entre este e o ficheiro criado previamente. Esta amostragem permite executar testes do tipo «restart tests», ou seja, testes em que as fontes de entropia do sistema são reiniciadas.

Por fim a terceira amostragem consiste em pelo menos cinquenta ficheiros de dez mega-bytes cada, uma vez mais, com um intervalo de inatividade de quinze segundos entre cada um. A quantidade de ficheiros a trabalhar foi considerada por recomendação

uma vez mais do NIST, [14].

Como intuito desta dissertação é avaliar a segurança da informação o enfoque será dado aos resultados dos testes sobre os geradores de números aleatórios. Em suma, os números gerados são de oito bits, como tal, a entropia deverá de variar entre zero e oito, sendo que nunca deverá de alcançar o zero e o oito, sendo tanto melhor, ou seja, mais aleatória, quanto mais próximo do valor oito for, como já foi referido na secção 3.8. Para mais informação consultar a secção 2.5 ou a web grafia [14].

4.1.2 Solução A para renovação de chaves e IV

Para colmatar a possível geração de IV de baixa qualidade, bem como a não renovação das chaves, desenvolveu-se uma primeira solução em que tanto as chaves como o IV são gerados na gateway e enviados para cada ESP que esteja conectado a esta através de uma ligação encriptada. Para garantir que a comunicação é segura foram incluídos na programação de cada ESP um IV e duas chaves geradas previamente pelo computador, entenda-se a chave A e a chave B, assegurando que o primeiro pedido é realizado com máxima segurança. Cada IV apenas será utilizado uma única vez, para realizar o pedido de novos vetores de iniciação à gateway. No entanto, a chave simétrica tem uma duração de dez mil envios de dados. Após processar todos estes envios cada ESP deve de proceder a um pedido de uma nova chave (A) para envio de mensagens, contudo este pedido é realizado com recurso à chave B. Esta nova chave é gerada na gateway e armazenada numa base de dados interna. Aquando de um segundo pedido de chave a gateway deverá de eliminar a segunda chave mas manter sempre a chave inicial. Tal acontece para garantir que em caso de falha completa de energia a comunicação possa ser restabelecida sem necessidade de nova programação dos equipamentos, bem como para prevenir que as mensagens que já tenham sido trocadas sejam decifradas por um atacante que consiga aceder à gateway. O pedido de renovação de chaves é realizado com recurso a uma segunda chave com o intuito de dificultar a decifra da informação por parte de um atacante. Para melhor compreensão visualizar a imagem 4.2. Importa referir, que a chave utilizada para fazer o pedido de renovação de chaves não é a mesma que é utilizada para fazer a encriptação dos dados a enviar.

4.1.3 Solução B para a renovação de chaves e IV

Durante o desenvolvimento foi ainda pedido por parte da Renault Cacia que fosse apresentada uma segunda opção de geração de chaves. Assim, estudou-se as alternativas de pós processamento que pudessem aumentar a entropia dos números gerados.

Como melhor solução surgiu a hipótese de gerar um código hash sobre os números gerados pelo próprio ESP. Assim, a placa deverá de gerar um número com 512 bits, ou seja 64 bytes, o equivalente a dizer que gera 64 números. Após isto a placa deverá de calcular o respetivo código hash deste número por meio da função sha256, obtendo assim a nova chave de 32 bytes. O processo deverá de ser repetido para criar um novo IV como visível na Figura 4.3. Para as chaves privadas cada ESP mantém a totalidade da informação gerada, contudo para cada IV o ESP deverá de reter apenas 16 bytes, dado ser o tamanho do bloco de dados utilizado ao longo de todas as encriptações. À posteriori, o ESP envia para a sua gateway a nova chave encriptada com recurso a uma segunda chave destinada à renovação de chaves, para que a gateway possa atualizar a sua base de dados sobre as chaves destinadas à troca de informação. Ou seja, a chave

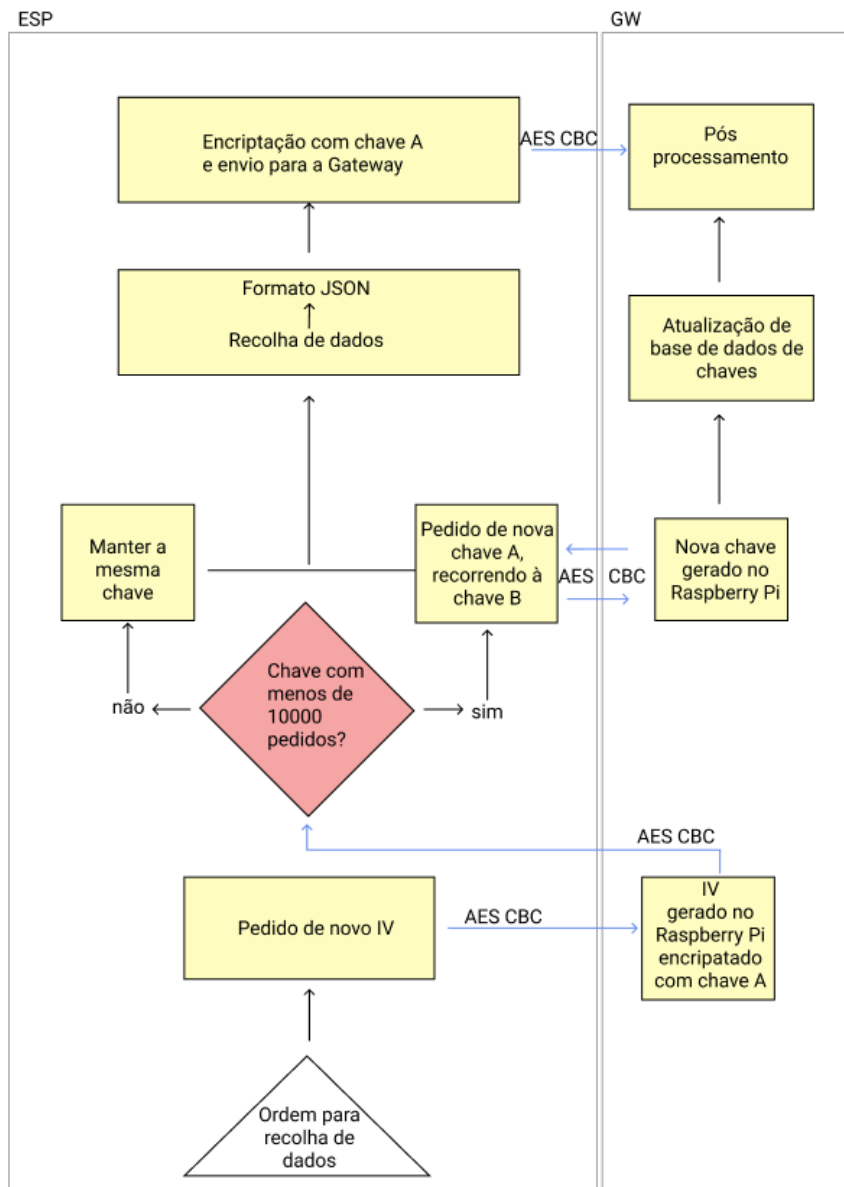


Figura 4.2: Renovação de chaves

que o ESP utiliza para realizar a renovação de chaves não é a mesma que utiliza para realizar a encriptação.

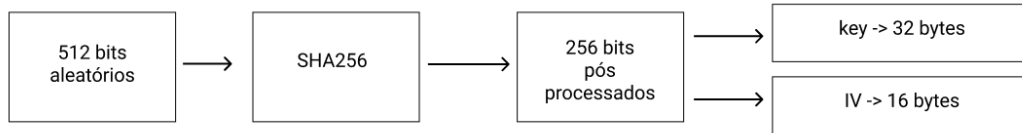


Figura 4.3: Pós processamento de chaves

Com o intuito de manter as duas soluções, como pedido pela Renault Cacia, quando cada ESP procede ao pedido de troca de chaves, recebe também a informação se deverá alterar para ser ele próprio a gerar as chaves privadas e o IV em vez de os pedir à gateway.

4.2 Patamar Iot

Esta camada é constituída por módulos IoT, nomeadamente ESP8266, parte integrante da estação Business Unit 01 (BU01) e módulos ESP8266 da estação Business Unit 02 (BU02). Esta divisão em «Business Units» é comum ao patamar IoT e ao patamar superior das gateways, tendo sido elaborada para referir os sensores que se encontram agregados a cada gateway, ou seja, a BU01, é composta pelo sensor data matrix, os três ESP32 já referidos e o Raspberry Pi 3 model B+.

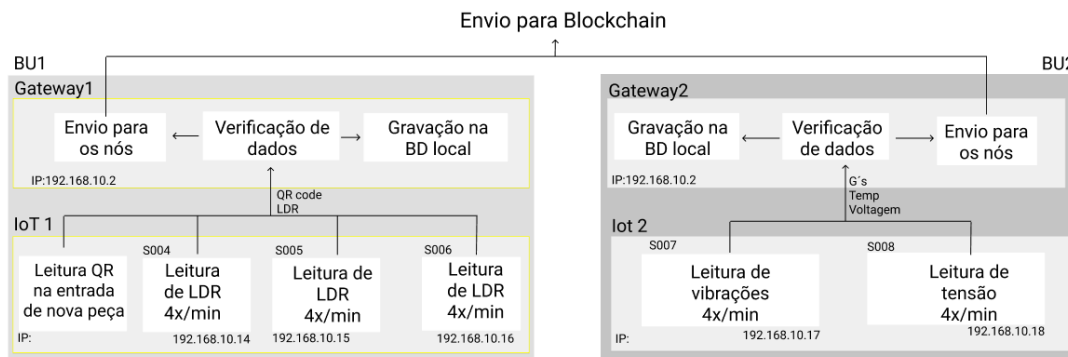


Figura 4.4: Recolha de dados IoT e Gateway

No que toca aos ESP32 e ESP8266, estes dois dispositivos são bastante semelhantes, suportando nos dois casos, ligações físicas e ligações Wi-Fi. Desta forma, estes dois equipamentos conseguem recolher os dados lidos pelos sensores e através de uma ligação Wi-Fi estabelecida com o router local, ligam-se a um servidor, presente na sua respetiva Gateway. Desta forma, através do protocolo HTML, é possível enviar a informação pretendida recorrendo a um POST. Assim, os ESP, enviam os dados necessários recorrendo a encriptação para que esta informação não possa ser lida e recorrendo ainda a autenticação da informação, para que se possa confirmar o emissor da mensagem por comparação da chave privada com a chave armazenada e correspondência ao respetivo nome de cada emissor. Para melhor verificação cada equipamento possui ainda um contador de mensagens que envia o seu valor numérico para a gateway de forma a fazer

uma verificação de que a mensagem é originária do destinatário que realmente se espera que seja. Para melhor compreensão a implementação encontra-se dividida em Business Unit's, denominadas de BU01 e BU02. Como apresentado na Figura 4.4 e explicado de seguida.

4.2.1 Business Unit 01

Esta secção para além da gateway que foi implementada fisicamente com recurso a um Raspberry Pi 3 model b+ é constituída três ESP8266, Figura 4.5, que possuem as mesmas características. Este módulos são responsáveis por recolher dados através do sensor de luminosidade PG5506 que têm embutidos na própria placa. Para efeitos de simulação cada ESP interveniente nesta mesma Business Unit deve de proceder à leitura de dados a cada 15 segundos. As características de cada ESP8266 são as seguintes:

- 1 UART
- 1 SPI
- 1 I2C
- 10-bit ADC
- 17 GPIOs
- Wi-Fi
- custo aproximado: 5€

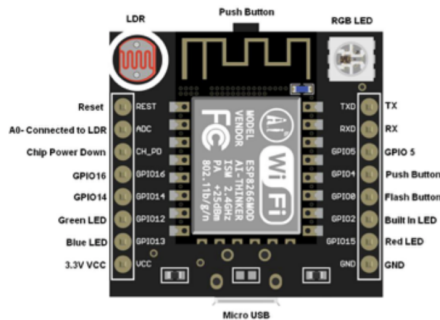


Figura 4.5: ESP8266, [33]

4.2.1.1 Leitura de peças ZD5800

Para além dos três ESP8266 na BU01 foi ainda implementado um leitor de Data Matrix. Este leitor, representado na Figura 4.6, é responsável por comunicar diretamente à gateway sempre que uma nova peça dá entrada na linha.

O leitor laser funciona de fábrica com uma ficha de 10 pinos. Contudo, para realizar a ligação direta ao Raspberry Pi, elaboramos uma ligação com um conector RJ45, em que o pino 1, será o TX, o pino 2 o Ground e o pino 6 a alimentação a 5V, como se



Figura 4.6: ZD5800

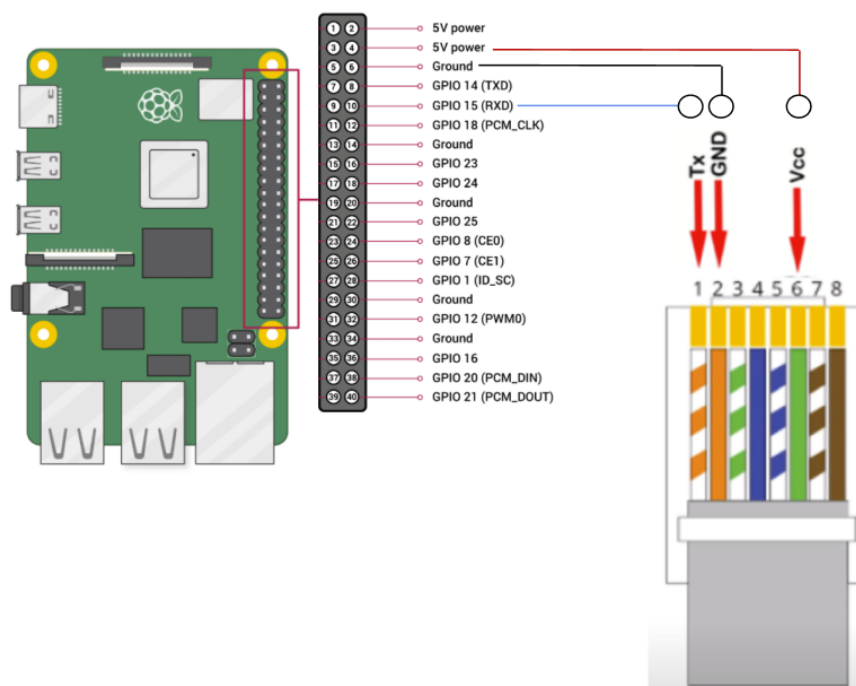


Figura 4.7: Esquema de ligação

pode ver na Figura 4.7. Para completar uma correta configuração do protocolo RS232, faltaria atribuir uma ligação ao pino RX. No entanto, como não existe a necessidade de enviar dados do módulo IoT para o leitor, este pino não foi atribuído. Assim, ao pino RX do Raspberry Pi é ligado ao cabo TX do sensor, o Ground de cada equipamento estará ligado um ao outro, e por último o pino Vcc será alimentado com 5V.

A configuração do sensor é feita com recurso ao manual de utilizador disponibilizado pelo fabricante. Este manual permite configurar o sensor, como por exemplo na implementação realizada, enviar a mensagem sem os caracteres «Carriage Return» e «Line Feed», bem como estando em constante procura do código, assim quando a peça da entrada na linha, a leitura será feita de modo automático sem necessidade de estar um operador presente. Para visualizar o manual de utilizador consultar o anexo I.1.

4.2.2 Leitura Data Matrix

Ao executar a leitura do código Data Matrix presente em cada peça será obtido uma string de elevado comprimento, contudo apenas a informação referente à numeração da peça é gravada na cadeia, como tal, do código lido na Figura 4.8, apenas parte desta é armazenada dos nós da blockchain, tendo em vista reduzir o espaço necessário. Esta informação é processada e concatenada à restante enviada pelos módulos IoT anexados a esta mesma BU, permitindo que cada peça tenha anexado a informação que diz respeito ao seu processo de fabrico.

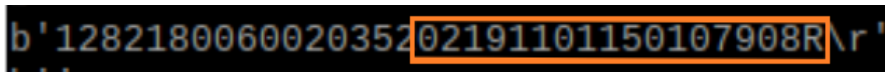


Figura 4.8: String lida pelo código Data Matrix

4.2.3 Business Unit 02

A segunda Business Unit ao nível dos dispositivos de recolha de dados é constituída por dois ESP32 que possuem ambos as mesmas características, representados na Figura 4.9. Estes módulos encontram-se equipados com Wi-Fi e um chip Bluetooth, desenvolvidos pela Espressif Systems, possuem ainda amplificadores de tensão e filtros de ruído. Para além destas características contem ainda outras especificações como:

- 18 pinos Analógico/Digital
- 3 interfaces SPI
- 3 interfaces UART
- 2 I2C interfaces
- 16 pinos PWM
- 2 interfaces I2S
- 10 GPIOs
- 128 kB Ram interna
- 4 MBs de memória flash
- custo aproximado: 5€

O primeiro módulo é responsável por recolher dados de vibrações e temperatura através do sensor GY-91, representado na Figura 4.10, já o segundo é responsável por simular uma variável de estado.

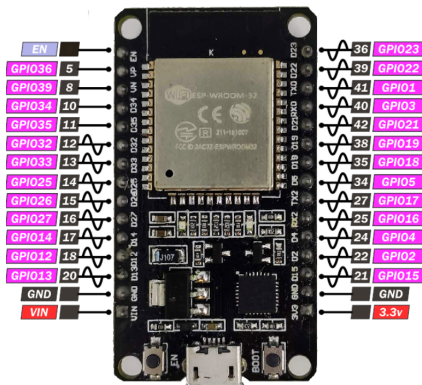


Figura 4.9: ESP32, [34]

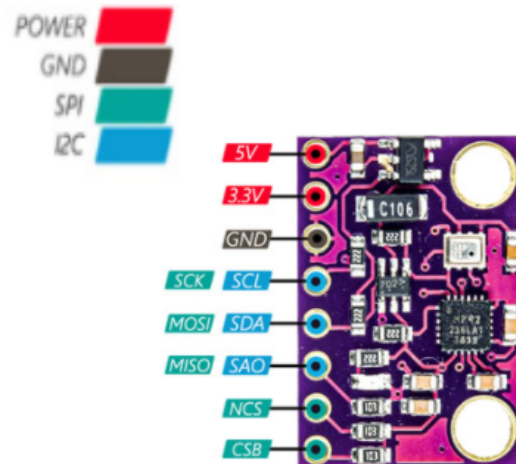


Figura 4.10: GY-91 pinout, [35]

4.2.3.1 GY-91 Sensor de vibração e temperatura

A placa GY-91 tem embutida em si dois sensores, o BMP280, que é capaz de fornecer a pressão barométrica e o MPU-9250, capaz de ler três eixos de aceleração e 3 eixos de giroscópio, usualmente denominados de X, Y e Z. Sendo esta placa composta por mais do que um sensor, na estrutura da mensagem enviada, para recolher a informação dos sensores, é necessário seleccionar o endereço do sensor, e após isto seleccionar o banco de memória pretendido. Para realizar esta ação, de modo a facilitar a programação do chip, recorreu-se à biblioteca «Wire». Posto isto, a comunicação com o chip foi estabelecida recorrendo ao protocolo I2C. Para saber que endereços deveríamos de ler bem como a estrutura da mensagem a enviar, e ainda como processar os dados recolhidos recorreu-se ao datasheet e aos manuais disponibilizados pelo comerciante, [36], [37], [38], [39] e [40].

- VIN: tensão de alimentação de 5v
- GND: terra
- SCL: I2C
- SDA: I2C data
- Configuração de endereço para I2C
- NCS: selecção do chip (apenas para o leitura no sensor MPU-9250)
- CSB: selecção de chip (apenas para o leitura no sensor BMP280)

4.2.3.2 Protocolo I2C e Biblioteca Wire

Como previsto no protocolo I2C, a biblioteca «Wire» permite ao ESP32 comunicar com dispositivos periféricos, só que reúne um conjunto de funções que facilita o envio e tratamento da informação que se pretende aceder. Este protocolo permite comunicar com dispositivos como acelerómetros, EEPROM'S entre outros, recorrendo ao pino SDA para

enviar os sinais de dados, e ao pino SCL para gerar o sinal de relógio, que permite temporizar o envio dos dados. Este protocolo prevê ainda a possibilidade de ligar a mais do que um sensor, slave, recorrendo a endereços de sete bits. O protocolo prevê ainda a possibilidade de controlo de fluxo [34]. Deste modo, para proceder à comunicação estabeleceu-se a seguinte ligação explicada a baixo e representada pela Figura 4.11.

- ESP32 3.3V = 3.3V GY-91
- ESP32 GND = GND GY-91
- ESP32 D23 = SCL GY-91
- ESP32 D21 = SDA GY-91

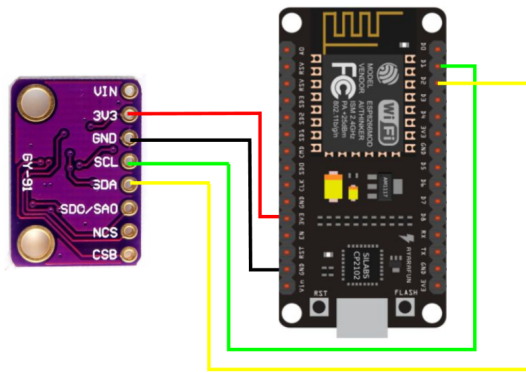


Figura 4.11: Ligação ESP32-GY-91

Para efeitos de simulação cada ESP interveniente nesta mesma Business Unit deve de proceder à leitura de dados a cada 15 segundos.

4.2.4 Programação e software patamar IoT

Os equipamentos utilizados no patamar IoT apesar de serem nativos em C/C++ foram programados em MicroPython, em suma, é uma variante mais leve do Python e que permite a sua utilização em dispositivos com menor capacidade de processamento como os que estão a ser utilizados. Para mais informação consultar o apêndice D.

Para proceder à encriptação recorreu-se a AES CBC. A encriptação, como já explicado, é realizada por primeiro obter um IV e uma chave privada, também conhecida pela gateway e que pode ser obtida por um dos dois métodos implementados. Sendo que cada chave apresenta-se como um UUID, «Universally Unique Identifier». Aquando da sua geração na gateway esta é obtida pela função UUID4 e está de acordo com a norma RFC 4122. Aquando da sua geração no ESP, esta é realizada pelo urandom e pós processado. A partir deste ID é calculado o seu valor hash com recurso à função SHA256 uma vez mais. A imagem 4.12 descreve de forma explicativa este processo.

Com o intuito de facilitar o tratamento de dados os dados recolhidos foram convertidos para um formato JSON, visível na Figura 4.13. Como os dados podem variar de tamanho, e o bloco usado tem um tamanho de 16 bytes, a informação é analisada e quando não é múltiplo de 16 são somados espaços em branco para que este requisito seja cumprido.

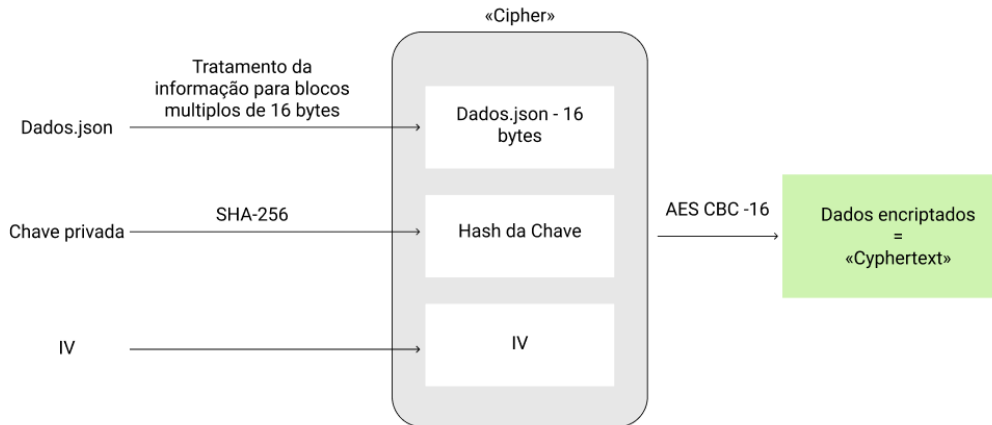


Figura 4.12: Processo de encriptação nos ESP's

```

180 def check_LDR_vals():
181     analog=adc.read()
182     # Data structure to send to Gateway Node.
183     if analog > 500:
184         LED.on()
185     else:
186         LED.off()
187
188     data = {'5005': counter,      # Sensing Unit No.
189            'LDR': analog,       # LDR read value
190            'LED': LED.value()}  # Value of debug LED
191     r=requests.post('http://192.168.10.2:8090/esp/vals05', data=(encryption(data)))

```

Figura 4.13: Exemplificação de codificação JSON

4.3 Patamar de Gateway's

A informação após ser recolhida pelo patamar IoT é enviada para o nível superior. O patamar da gateway apresenta mais processos em funcionamento bem como maior necessidade de processamento dado as grandes quantidades de informação e a maior complexidade dos programas desenvolvidos, como tal recorreu-se a um Raspberry Pi 3 Model B+, Figura 4.14, para desempenhar o papel de primeira gateway, sendo que possui as seguintes características:

- Broadcom BCM2837B0 Cortex-A53 1.4 GHz ;
- 1GB RAM;
- 1 porta Ethernet 100 Mb;
- Bluetooth Low Energy (BLE);
- 4 portas USB;
- 40 pinos GPIO;
- dimensões: 85.60 mm × 56.5 mm;

- custo aproximado: 35€

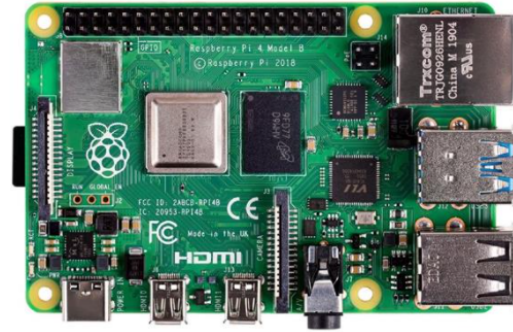
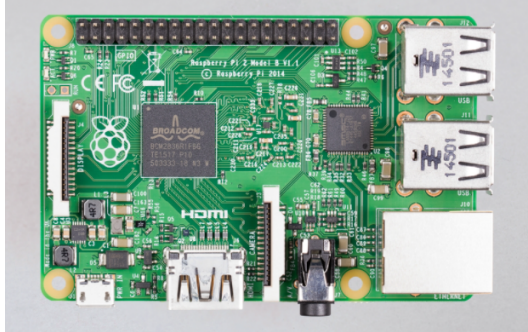


Figura 4.14: Raspberry Pi 3 Model B+, [41] Figura 4.15: Raspberry Pi 4 Model B, [41]

Para desempenhar o papel da segunda gateway recorreu-se a um Raspberry Pi 4 Model B, Figura 4.15. As características do equipamento são apresentadas a baixo.

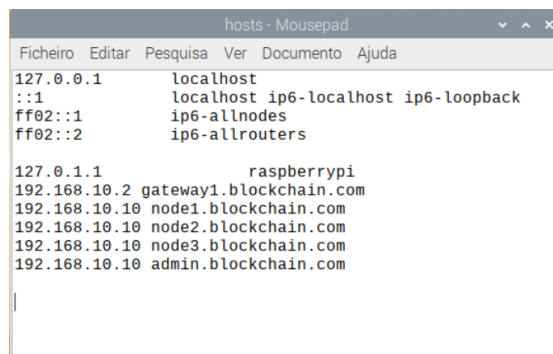
- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB LPDDR4-3200 SDRAM
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
- Bluetooth 5.0
- 1 porta Gigabit Ethernet;
- 2 USB 3.0 ports;
- 2 USB 2.0 ports
- 40 pinos GPIO;
- 2 portas micro-HDMI
- custo aproximado: 45€

Apesar de possuírem características de hardware diferentes, ambos foram programados em ambiente ubuntu, nomeadamente Raspberry Pi OS. Assim desenvolveu-se um servidor Flask com o intuito de poder receber a informação proveniente do patamar inferior bem como verificar se esta é fidedigna aquando da sua descriptação, pela verificação do ID de cada equipamento, do seu contador, bem com da verificação dos dados que se espera obter, como já explicado.

Este patamar necessita de segurança adicional devido à sua possível ligação a redes exteriores foi implementado a firewall presente neste sistema operativo, Ubuntu firewall - ufw - com o intuito de restringir as conexões com IP's que se encontrem fora la lista de IP conhecidos da mesma rede. Desta forma, foram restritas as ligações a equipamentos que se conheça os IP's, neste caso, dos ESP que cada gateway deverá de receber informação, bem como dos nós para onde deverá enviar a informação. Com o intuito de dificultar a decriptação de informação por parte de um atacante, através da análise

dos tempos de processamento por parte da gateway, bem como para otimizar o processamento, foi implementado um URL específico para cada ESP, não existindo assim partilha do mesmo URL por mais do que um ESP. Foram ainda implementadas ligações recorrendo ao protocolo TLS versão 1,3 para realizar as comunicações da gateway para os nós da blockchain. Através desta restrição de IP's em cada gateway criou-se uma clara divisão entre que equipamentos é que a gateway poderá ou não receber pedidos, restringindo assim equipamentos externos que poderiam executar algum tipo de ataque.

Para que o protocolo possa funcionar de forma correta e reencaminhar a informação para os nós da blockchain, é necessário adicionar a correta correspondência de IP's aos respetivos certificados com os quais se pretende estabelecer comunicação, para tal, é necessário aceder ao ficheiro `/etc/hosts` como visível na Figura 4.16.



```

hosts - Mousepad
Ficheiro Editar Pesquisa Ver Documento Ajuda
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

127.0.1.1 raspberrypi
192.168.10.2 gateway1.blockchain.com
192.168.10.10 node1.blockchain.com
192.168.10.10 node2.blockchain.com
192.168.10.10 node3.blockchain.com
192.168.10.10 admin.blockchain.com

```

Figura 4.16: Certificados na gateway

Para facilitar a escalabilidade foi ainda criado um ficheiro que contem toda a informação que a gateway necessita, como o endereço do nó principal, o seu próprio nome ou por exemplo a localização dos certificados que necessita de utilizar ou ainda as portas por onde tem de comunicar, representado pela Figura 4.17. É de notar que ao longo da implementação, toda a lista de IP segue uma sequência, ou seja, os IP que terminam entre .02 e .09 são destinados às gateways, já os IP .10 a .13 são destinados aos nós, sendo que a partir de .14 são destinados aos dispositivos IoT de recolha de dados. Esta terminologia pode voltar-se a repetir por cada novo router que seja implementado. Esta terminologia é também exemplificada pela imagem 4.4.

4.3.1 Processos em execução na Gateway

Em cada gateway física existem 2 programas que podem ser divididos em 4 processos. O primeiro processo é responsável por enviar, de forma encriptada, dois números aleatórios para o patamar inferior, de forma a que o ESP recetor possa atualizar a sua chave privada e guardar o segundo número como IV. Assim no próximo envio por parte do ESP para a gateway a informação será encriptada de forma segura.

O segundo processo exemplificado na imagem 4.18 à direita, é responsável por receber essa mesma informação através de um POST e proceder à sua descriptação. Após isto, procede ao tratamento de dados e envio para o script principal através de um POST HTTP com segurança do tipo TLS. Este script é então responsável por reenviar para os nós da blockchain. Por outro lado é também responsável por receber informação por parte do nó central sobre os nós que se encontram ativos, para que desta forma a gateway

```

() config.json ×
D: > OneDrive - Universidade de Aveiro > Desktop > cenas > UA > tese > desenvolvimento > ficheiros finais > BD_COMUNICACAO > () config.json > ...
1  {
2    "ADMIN_HOST": "admin.blockchain.com",
3    "ADMIN_PORT": 5000,
4    "B_U": "RBU01",
5    "CA_CERT": "certs/ca.cert",
6    "CA_PEM": "certs/ca.pem",
7    "EDGE_CERT": "certs/gateway_edge.cert",
8    "EDGE_HOST": "0.0.0.0",
9    "EDGE_KEY": "certs/gateway_edge.key",
10   "EDGE_PORT": 8090,
11   "GATE_CERT": "certs/gateway1.cert",
12   "GATE_HOST": "gateway1.blockchain.com",
13   "GATE_KEY": "certs/gateway1.key",
14   "GATE_PEM": "certs/gateway1.pem",
15   "GATE_PORT": 5000,
16   "SERIAL_DVC": "/dev/ttyACM0",
17   "sensing_units": {
18     "S003": "c86a301a-d227-4ea1-96a0-40f96de728cf",
19     "S003_V2": "c86a301a-d227-4ea1-96a0-40f96de728cf",
20     "S004": "c86a301a-d227-4ea1-96a0-40f96de728cf",
21     "S004_V2": "dcfad640-5f49-402e-8cf9-fb5ee9c5",
22     "S005": "c2642757-21e6-469a-81fe-99f0ef53be8b",
23     "S005_V2": "CHAVE_V04.0...",
24     "S006": "3cbc45f3-8f1a-4a42-acb0-d8671cc1ab5c",
25     "S006_V2": "3cbc45f3-8f1a-4a42-acb0-d8671cc1ab5c"
26   }
27 }

```

Figura 4.17: config.json implementado na Gateway

saiba quais são os nós disponíveis a receber a informação.

Este processo é ainda responsável por proceder ao armazenamento da informação num ficheiro texto como será explicado na secção a seguir, 4.3.2. É de notar que sempre que um pedido é feito este fica à espera de uma resposta, para evitar congestionamento da gateway ou tempos de espera infinitos, foram implementados tempos de resposta restritos, usualmente denominados de «timeouts».

4.3.2 Base de dados Gateway

Por fim a gateway contempla ainda mais dois processos. O primeiro pertence ao script que comunica com o patamar IoT e é responsável por armazenar toda a informação num ficheiro de texto, contudo, esta informação encontra-se encriptada, através do método AES CBC, tendo sido implementado renovação das chaves privadas, através da leitura de um ficheiro pré-escrito e que apenas este processo consegue aceder através das permissões de utilizador. Apesar disto, as chaves encontram-se ainda encriptadas uma vez mais. Cada chave necessita de 32 bytes, sendo que tem uma vida útil de dez mil mensagens, como tal para garantir a funcionalidade por 20 anos foram previamente geradas e armazenadas doze mil e quinhentas chaves, o que corresponde a 0.6Mb. Por sua vez este ficheiro foi duplicado para possibilitar a leitura através do segundo processo que é responsável por aceder à informação armazenada na base de dados e descriptá-la. Este acesso apenas é possível por um utilizador credenciado para tal, tendo sido implementados mecanismos para controlo de acesso como um nome de utilizador e uma chave com controlo sobre o número de vezes que se tenta aceder sem sucesso, para evitar ataques por força bruta. Uma vez mais este ficheiro encontra-se bloqueado apenas a este processo, o que evita que um terceiro utilizador lhe consiga aceder e descriptar as chaves, que permitiria aceder à totalidade da informação. Para uma melhor compreensão do processo visualizar o esquema 4.19. Do lado esquerdo é possível ver a informação a chegar

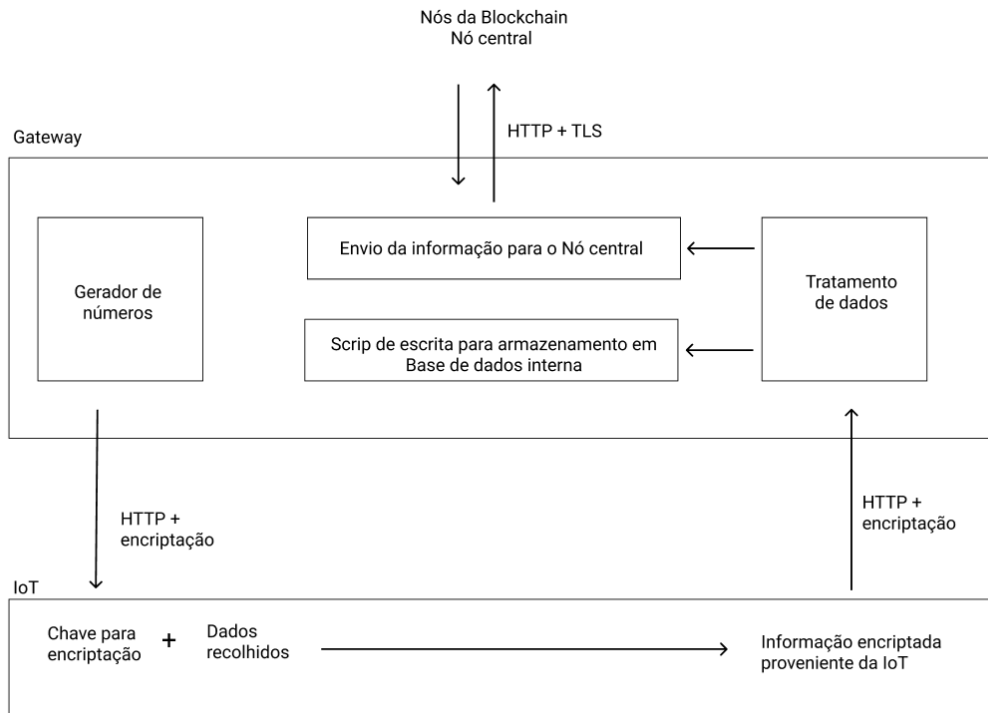


Figura 4.18: Processos em execução na Gateway

ao processo de escrita, bem como apenas este mesmo processo a ler as chaves privadas. À direita do ficheiro que contem as chaves privadas é possível ver esse mesmo ficheiro duplicado, contudo apenas o script de leitura o pode ler. Ao centro da imagem é visível um ficheiro de texto em que apenas o script de escrita pode armazenar informação e o de leitura ler, como já referido.

4.4 Segurança de rede

Garantir a segurança da implementação é uma tarefa de elevada complexidade uma vez que é necessário prever todos os tipos de ataques a que a rede pode ser submetida e de seguida elaborar métodos para que estes não sejam bem sucedidos, ou em último recurso, se um atacante conseguir transpor as defesas da rede que não o consiga fazer de forma repetitiva ou permanente. No que toca à implementação de chaves privadas, é necessário abordar dois aspetos que poderão ser alvo de falhas, a implementação de uma chave nos dois elementos da comunicação e o transporte desta, [8]. Para contrariar assim possíveis ataques foi criado um certificado de autoridade de forma a criar e assinar chaves públicas e privadas nos patamares superiores à IoT. Para além de certificados de autoridade foram ainda implementados outros métodos de encriptação e de chaves privadas como já referido ao nível da gateway e ainda a implementação de credenciais de utilizadores e password no que diz respeito à interação com o utilizador. Para além de garantir que não ocorrem falhas na segurança é necessário saber que elas não ocorreram, como tal foram implementados métodos garantir a verificação de toda a informação na rede bem como em casos de falha de hardware.

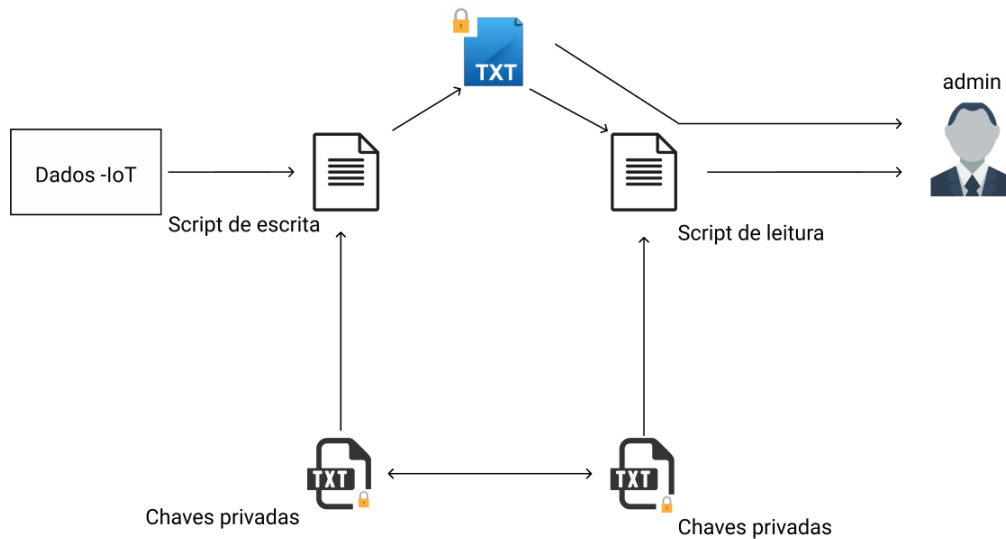


Figura 4.19: Exemplificação de funcionamento de encriptação na gateway

4.5 Certificados de autoridade

Dado que existe a possibilidade dos nós da cadeia se encontrarem em diversos locais fora da mesma rede, sejam eles departamentos da mesma empresa, outras fábricas, ou fornecedores, existe a necessidade de verificar que esses mesmos nós são de facto quem estes dizem ser. Para isto, foram implementados certificados de autoridade. Estes certificados são implementados vulgarmente nos sites que visitamos, sendo credenciados por uma entidade que o nosso navegador confia, partindo do pressuposto que se confia no navegador pode-se então aceder de forma segura.

Para proceder à certificação de documentos a entidade certificadora, fornece uma chave privada e uma pública assinadas por esta para cada elemento da rede. Para saber que um documento pertence a uma entidade a própria assina-o com a sua chave privada que nunca pode ser divulgada. Para verificar se o emissor é o desejado o recetor recorre à chave pública, visível por todos na rede, podendo assim verificar quem assinou.

Com o intuito de diminuir custos e de forma a adquirir mais conhecido desenvolveu-se um CA de base, ao que são chamados de «self signed certificate» e que têm o intuito de ser utilizados em implementações pessoais ou de menor escala. Recorreu-se ao software OpenSSL⁴(1.1.1j). Este software pode ser utilizado em Windows, Linux ou MacOs, na implementação a seguir recorreu-se a Linux.

4.5.1 Criação de certificado de autoridade

No código 4.5.1 é visível o primeiro comando utilizado. A instrução `req` produz e processa o pedido de criação de certificado, sendo que à frente constam instruções para este mesmo pedido. A instrução `-newkey` indica a criação de uma chave privada e outra pública utilizando RSA de 2048 bits. Para ser possível gravar e aceder em disco ao próprio certificado, é utilizado o comando `-x509`. Para fornecer a informação necessária segue-se o comando `-subj` que informa que será fornecido uma string, string essa em que consta

o nome do país, apenas pode ser referido com duas letras maiúsculas (C), o distrito (ST), a localidade (L), a organização (O) e por fim o nome do server em que atua o certificado (CN). É importante referir que estes nomes têm de ser únicos, apesar de poderem partilhar o mesmo IP. Por fim, o comando `-nodes` refere que não é necessário uma palavra chave cada vez que se pretenda trabalhar com o certificado.

```
1 openssl req -newkey rsa:2048 -new -x509 -sha256 -extensions v3_ca -
  out ca.cert -keyout ca.key -subj "/C=PT/ST= Aveiro/L=OAZ/O=UA-DEM/CN=
  blockchain.com" -nodes
```

Listing 4.1: Criação de CA

4.5.2 Criação de certificados para cada entidade

Dado que existem várias entidades é necessário um certificado para cada elemento. Assim procedeu-se à criação de uma pasta no domínio `/etc/hosts` em que constam os IP's de cada elemento a que foi atribuído um certificado seguido do seu respetivo DNS («Domain Name System»). As operações apresentadas à frente tem de ser repetidas para cada elemento da rede, como tal apenas será demonstrado para um elemento. Desta forma, criou-se uma chave encriptada `-.key` e um certificado `-.csr` pertencente ao elemento `popos.blockchain.com`.

```
1 openssl req -newkey rsa:2048 -new -sha256 -out popos.csr -keyout
  popos.key -subj "/C=PT/ST= Aveiro/L=Aveiro/O=UA-DEM/OU=servers/CN=
  popos.blockchain.com" -nodes
```

Listing 4.2: Certificado para client

4.5.3 Assinar o certificado pelo CA principal

Após a criação do certificado individual este tem de ser assinado pela entidade principal. Para tal foi criado o ficheiro `./demoCA/serial` e introduzido uma constante inicial para que o sistema passe a gravar o número de certificados em cima desta. Após isto recorreu-se ao comando 4.3, dando como parâmetros de entrada os ficheiros `popos.csr`, `ca.key` e `ca.cert` para obter como parâmetro de saída o certificado assinado `popos.cert`.

```
1 openssl ca -in popos.csr -out popos.cert .keyfile ca.key -cert ca.cert -
  outdir.
```

Listing 4.3: Assinatura do certificado

Por fim, falta apenas proceder à junção da chave o certificado, através do comando presente no exemplo de código 4.4. É agora possível distribuir e implementar os certificados pelos elementos pretendidos.

```
1 cat popos.cert popos.key > popos.pem
```

Listing 4.4: Compactação das credenciais

4.5.3.1 Certificados para o navegador

Para ser possível aceder às páginas pretendidas da interface é necessário proceder à implementação do certificado. Contudo, para implementação no navegador a criação do certificado tem de ser no formato PKCS12. Para proceder à criação destes certificados

é necessário o `cert.pem` e a chave respetiva para obter o `cert_firefox.p12` com o qual poderemos certificar a ligação através do navegador. É de relembrar que os ficheiros utilizados foram criados pelos passos explicados nas secções 4.5.2 e 4.5.3. O código a utilizar encontra-se visível no exemplo de código 4.5.

```
1 openssl pkcs12 -export -out cert_firefox.p12 -in cert.pem -inkey key.pem
```

Listing 4.5: Criação de certificado TLS dedicado ao navegador

Na imagem 4.20 e 4.21 é exemplificado o processo de instalação dos certificados no navegador Chrome (89.0.4)(64 b-bit), a implementação nos restantes navegadores é bastante similar.

4.6 Configuração dos certificados

Para realizar a configuração do navegador, será necessário seguir os seguintes passos, através da página exemplificada na Figura 4.21, **Definições** -> **Privacidade e Segurança** -> **Segurança** -> **Gerir Certificados** irá abrir uma nova janela, Figura 4.21, no qual deverá seleccionar **Outras pessoas** e adicionar a chave pública do certificado criado.

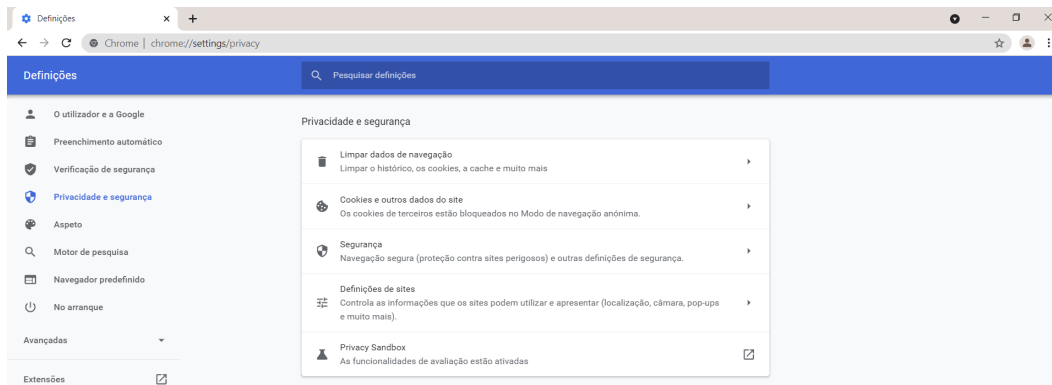


Figura 4.20: Instalação do certificado - passo 1

4.6.1 Ligações encriptadas e tipos de utilizadores

As ligações desde a gateway para os patamares superiores da implementação são asseguradas por meio de HTTPS, recorrendo ao modo de criptografia RSA, sendo que as chaves foram geradas e instaladas como mostrado acima. Todas as ligações de «front-end» podem ser confirmadas como seguras e encriptadas como por exemplo na Figura 4.22, onde também é visível o método que foi utilizado e o emissor do certificado.

Para além destes métodos de segurança, manteve-se o habitual método de nome e chave de utilizador, de forma a garantir que apenas utilizadores conhecidos têm acesso, como visível mais à frente na Figura 4.25. O acesso dos utilizadores foi ainda restrito a uma tentativa de ligação por segundo, o que evita ataques de login repetitivos que tornariam o server extremamente lento. Desta forma foi ainda contemplado a amostragem de informação consoante o tipo de utilizador que tenta estabelecer conexão, ou seja, a informação que é disponibilizada a um utilizador externo não é a total que se encontra armazenada e que é disponibilizada ao administrador da empresa por exemplo. Todo

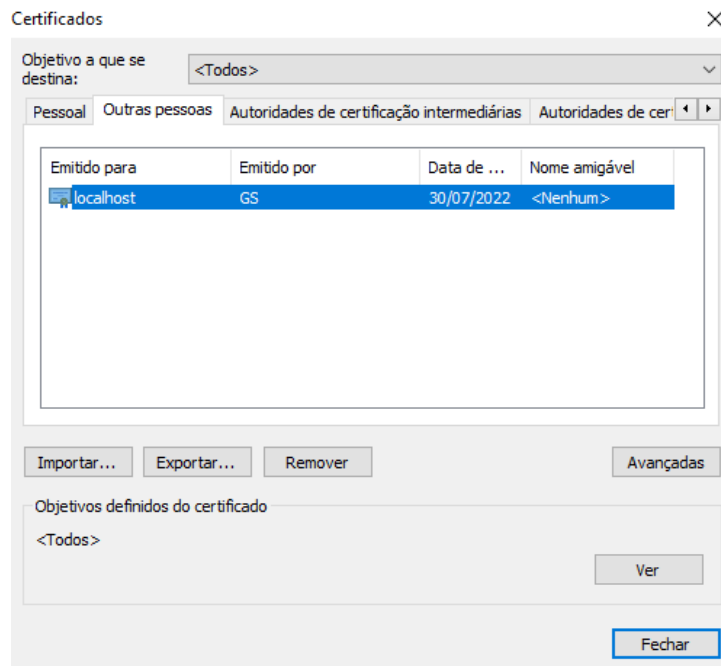


Figura 4.21: Instalação do certificado - passo 2

o armazenamento de informação de cada utilizador foi feito ainda de forma encriptada, para que não existam problemas de fugas de informação pessoal.

4.7 Blockchain e Base de dados empresarial

A base de dados empresarial contém toda a informação recolhida, seja ela proveniente do patamar mais baixo da implementação ou a gerada com o acesso ao front-end. Desta forma, foram mantidas 3 coleções de informação. A primeira responsável por armazenar todos os dados, a segunda responsável por armazenar os dados dos utilizadores de forma encriptada e a sua atividade no front-end e a terceira responsável por assegurar uma backup dos dados armazenados em cada nó. Esta terceira coleção é redundante e serve para testar a implementação em vigor, [29].

4.7.1 TLS aplicado à MongoDB

Para assegurar uma ligação segura com a MongoDB também aqui foi implementado o protocolo TLS, contudo foram necessárias algumas alterações às configurações iniciais. Antes de proceder à execução do mongod é necessário configurar o ficheiro `mongod.conf` que se encontra dentro da pasta `«/etc»`, para tal é necessário aceder ao ficheiro com permissões de administrador e alterá-lo de acordo com a Figura 4.23.

Contudo importa referir que para executar qualquer leitura é necessário aceder ao server com um protocolo TLS previamente conhecido e que o server da MongoDB foi predefinido como `mongodb.blockchain.com`. Para melhor compreensão da programação deste ficheiro ver o anexo G.1.1.

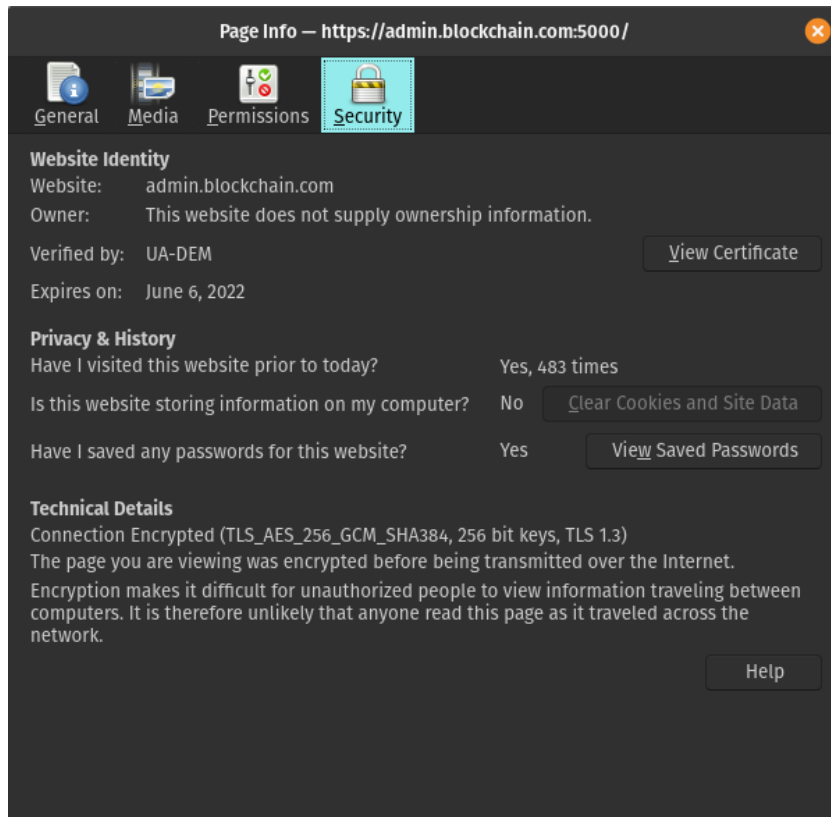


Figura 4.22: Verificação de certificado ativo

4.7.2 Nós da Blockchain

Para a realização do armazenamento de informação na Blockchain é necessário que existam nós «miners» para proceder à criação dos blocos como explicado na secção 3.4.2. Para além de criar blocos de informação, os nós são também capazes de verificar se a informação contida nos blocos é verdadeira através do cálculo do valor hash e comparação com o valor hash armazenado na cadeia. Como os nós atuam como servidores baseados em Flask, respondendo a pedidos HTTP em caso de falha de um nó a lista é atualizada. Os nós ao se encontrarem a funcionar por mais do que um equipamento torna-se mais difícil que exista uma falha completa da rede, dado que é improvável que todos os equipamentos falhem em simultâneo. Desta forma, a quantidade de informação perdida é reduzida, visto que se precede a uma atualização da lista de nós, [29]. Para a implementação realizada recorreu-se à implementação de três nós, responsáveis por processar toda a informação.

4.7.3 Nó central da Blockchain

O nó central, apesar de ter permissões superiores aos restantes, não é intrusivo na troca de informação. Este nó é responsável essencialmente por fazer a gestão dos participantes da rede, para tal necessita de acesso à MongoDB onde armazena as credenciais dos utilizadores. É ainda responsável por interagir com o front-end e aceder aos pedidos do utilizador, podendo acrescentar ou remover gateways, nós ou alterar dados do próprio


```
mongod.conf •
etc > mongod.conf
1 # mongod.conf
2 # for documentation of all options, see:
3 # http://docs.mongodb.org/manual/reference/configuration-options/
4
5 # Where and how to store data.
6 storage:
7   dbPath: /data/db
8   journal:
9     enabled: true
10  engine: wiredTiger
11 # mmapv1:
12 # wiredTiger:
13
14 # where to write logging data.
15 #systemLog:
16 # destination: file
17 # logAppend: true
18 # path: /var/log/mongodb/mongod.log
19
20 # network interfaces
21 net:
22   port: 27017
23   bindIp: mongodb.blockchain.com
24 net:
25   tls:|
26     mode: requireTLS
27     certificateKeyFile: /home/guilhermesalgueiro/Desktop/certificados/mongodb.pem
28     CAFile: /home/guilhermesalgueiro/Desktop/certificados/ca.pem
29     allowConnectionsWithoutCertificates: false
30
31 # how the process runs
32 processManagement:
33   timeZoneInfo: /usr/share/zoneinfo
```

Figura 4.23: mongod.conf

utilizador ou de terceiros. Sempre que um nó seja adicionado o nó central tem de proceder a uma sincronização com os restantes nós bem como uma atualização da lista de nós às gateways. No que toca ao front-end foi mantida a implementação sendo utilizado a extensão Flask-Login para limitar as tentativas de login a um utilizador por IP por segundo respetivamente, [29].

4.8 HMI

O último patamar pela qual a informação passa é a sua disponibilização ao utilizador. Nesta área, o trabalho desenvolvido pode ser separado na sua prototipagem e na sua conceção real de código.

4.8.1 Prototipagem

Como já referido com o intuito de prever possíveis problemas no desenvolvimento de um HMI responsivo recorreu-se a um software de prototipagem, nomeadamente, Invision Studio, visível na imagem 4.24, o que permitiu desenvolver a ideia de como seria apresentada a interface final e o que esta deveria de contemplar. Na imagem ao centro encontra-se representado as páginas a desenvolver. As ligações entre cada botão e a respetiva página encontram-se representadas pelas setas a azul. Do lado esquerdo da imagem é ainda visível todas as paginas desenvolvidas.

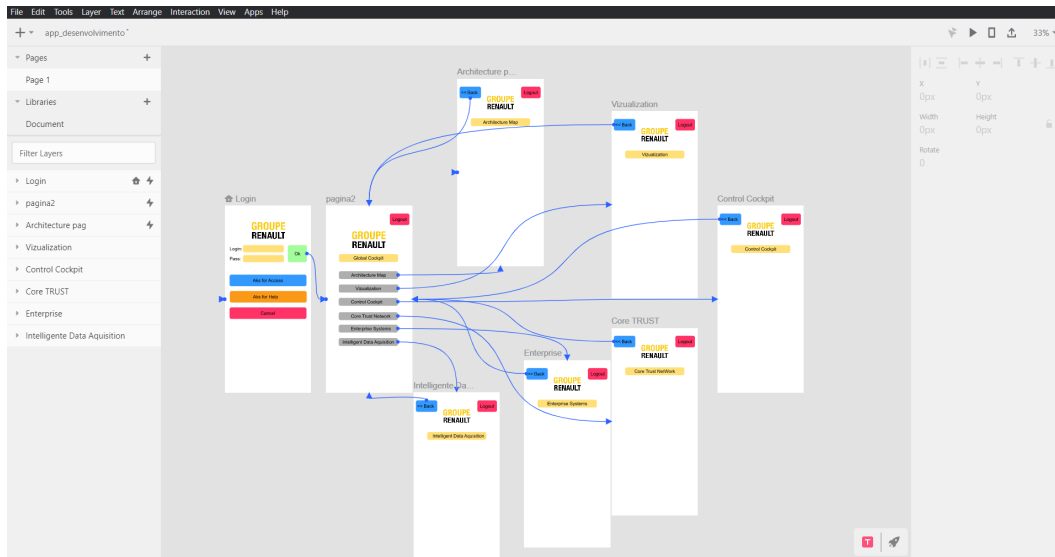


Figura 4.24: Prototipagem com Invision Studio

4.8.2 Conceção

A informação disponibilizada ao utilizador depende das suas credenciais. Sendo que ao utilizador admin é disponibilizada toda a informação, mas a utilizadores convidados apenas a que se achou adequada e que não compromete a cadeia a que é fornecida, como já referido. Para realizar o desenvolvimento das plataformas manteve-se a opção de linguagem baseada em Hypertext Markup Language (HTML), contudo, foi também combinado com JavaScript/AJAX para fazer a troca de informação com o server. Para criar a interface com um aspeto mais fluido manteve-se a opção de Cascading Style Sheets (CSS), acompanhado da livreria Bootstrap. A implementação foi ainda concebida de forma a se adaptar a vários formatos, ou seja, independente do tamanho ou orientação do ecrã a interface deverá de se manter com um aspeto fluido e padrão, não alterando a experiência do utilizador.

4.8.3 Experiência de utilização

A interação entre o utilizador e o software inicia-se pela página de login. Ao aceder a esta página o utilizador deverá de inserir as suas credenciais de acesso. Caso não possua ou exista algum problema é lhe possível fazer um pedido de ajuda através dos botões centrais, como visível na Figura 4.25.

Após proceder ao login é disponibilizado ao utilizador uma página onde existem todas as abas de trabalho desta rede. Aos utilizadores comuns, sem credenciais de administrador não será disponibilizado as páginas `/coretrustnetwork`, `/dashboard` nem ainda a página `/controlcockpit`. Nas figuras 4.27 e 4.28 é visível a página `/settings` onde é possível alterar as credencias dos utilizadores, adicionar ou remover gateways e nós da rede. Para fazer qualquer tipo de adição o utilizador não deverá de incluir o tipo de ligação, entenda-se, `https://`, mas sim iniciar pelo nome do elemento seguido da porta em que está em funcionamento, como por exemplo `node1.blockchain.com:5000`.

Na página «Dashboard», Figura 4.29, é ainda possível verificar informação sobre o

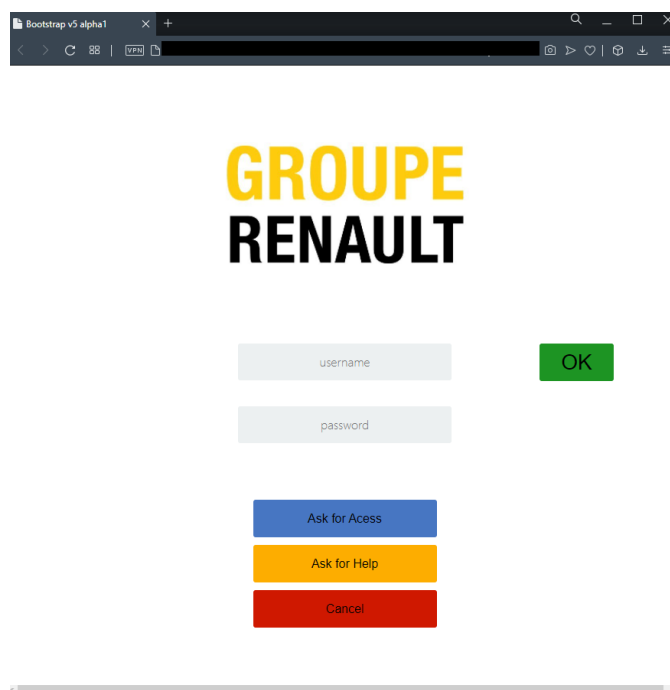


Figura 4.25: Login

último bloco criado, o número de visitas ou o número de utilizadores que se encontram a aceder ao front-end. A informação é mostrada de forma dinâmica. No que toca à procura de informação na base de dados sobre as transações esta por ser feita com recurso ao próprio valor hash, pelo ID da peça que se pretende obter informação ou ainda sobre a Business Unit ou o bloco em que foi escrito, [29].

Para verificar se a informação existe na cadeia, podem ocorrer dois processos. O primeiro em que é dado diretamente o valor hash da transação e apenas é verificado se este se encontra gravado nos nós. A segunda opção passa por fornecer apenas o ID da peça, o bloco de dados ou a Business Unit que pretendemos analisar, desta forma, primeiro procede-se à análise de possíveis valores hash na coleção Blockchain da MongoDB e só depois estes valores são cruzados com os existentes nos nós. Aqueles que obtenham correspondência são apresentados ao utilizador, [29].

Por fim foi ainda desenvolvido uma página de Control Cockpit onde será possível apenas aos utilizadores com total acesso alterar a forma como as chaves privadas e o IV são obtidos como foi explicado nas secções 4.1.2 e 4.1.2.

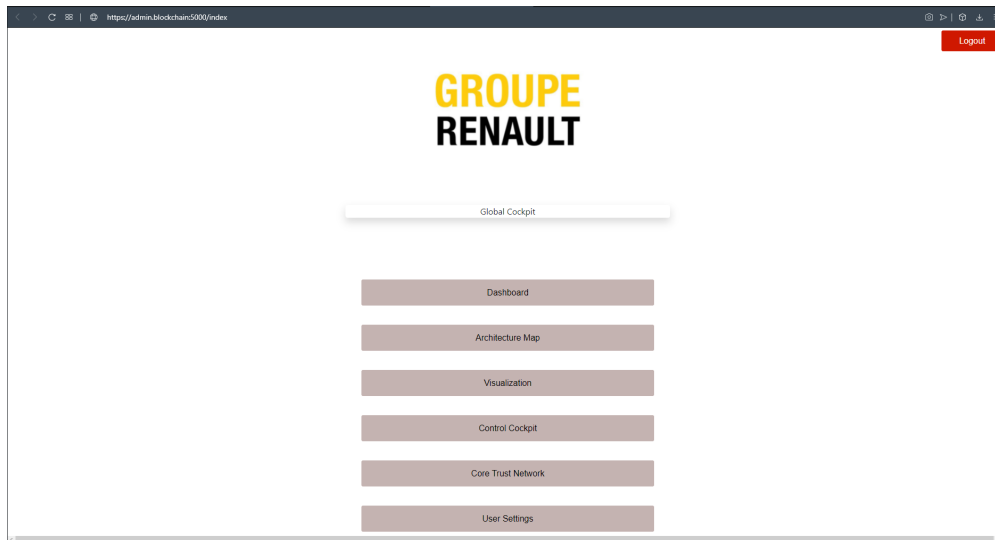


Figura 4.26: Index

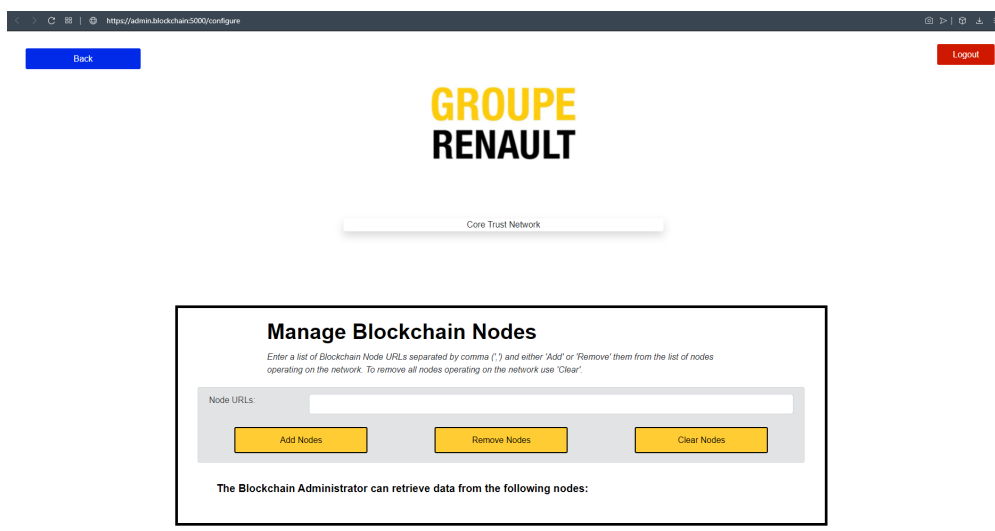


Figura 4.27: Controlo de elementos da rede parte 1 - Core trust Network

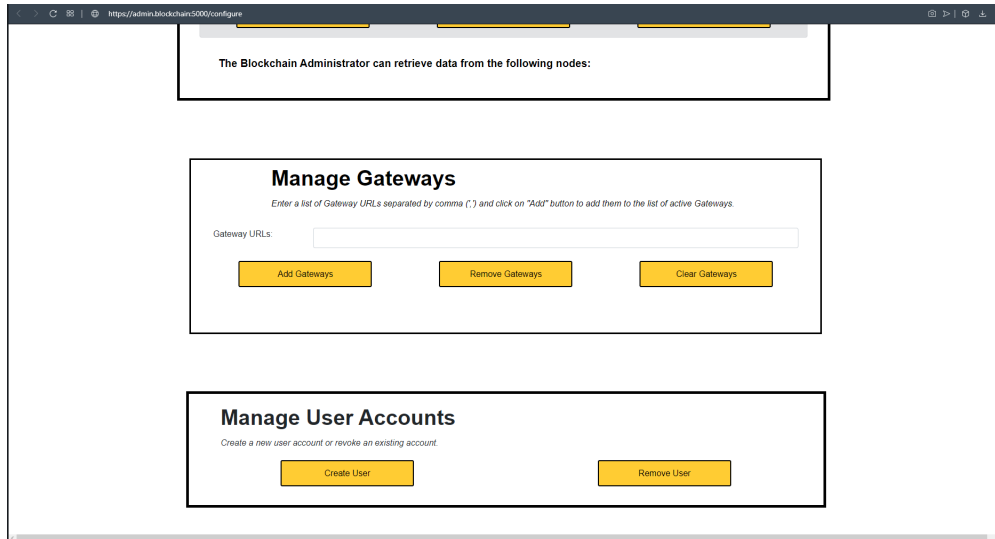


Figura 4.28: Controlo de elementos da rede parte 2 - Core trust Network

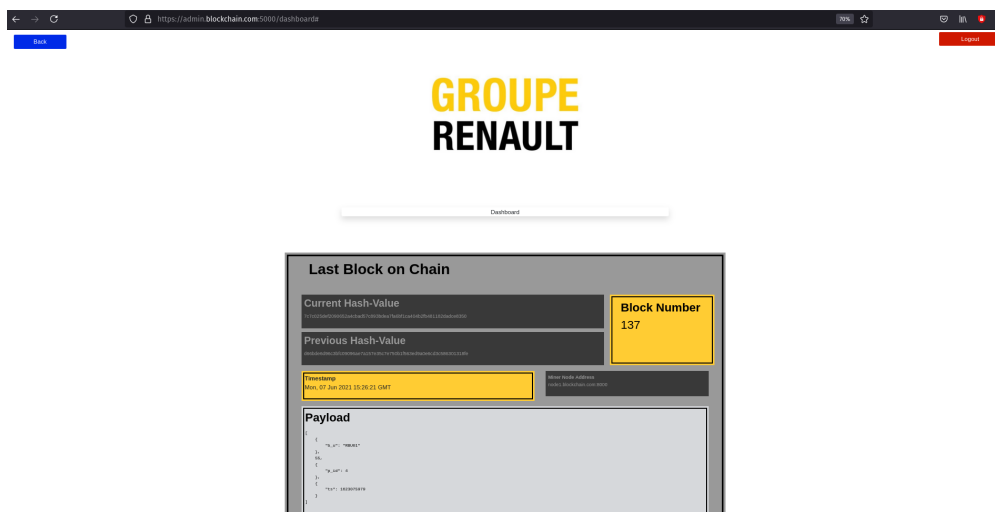


Figura 4.29: Dashboard

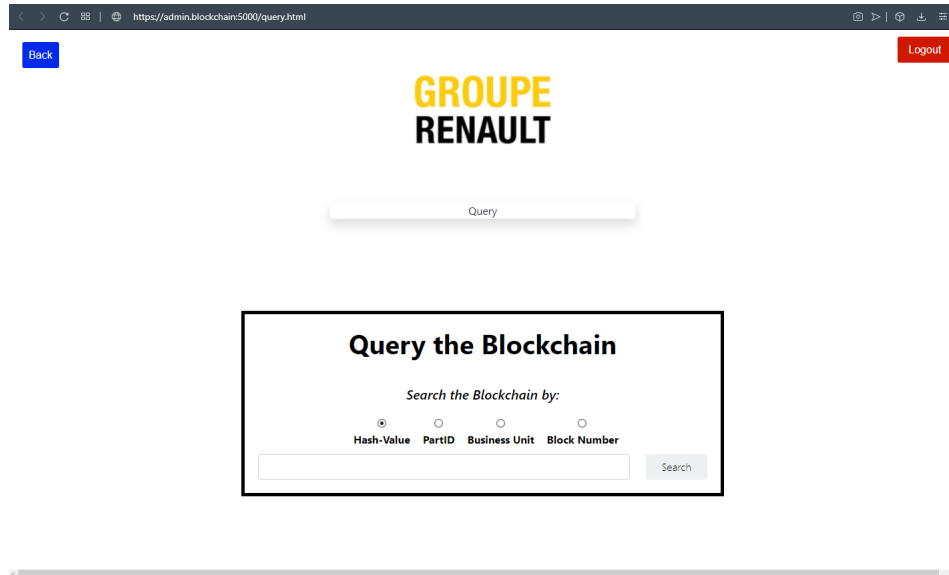


Figura 4.30: Consulta de informação - query

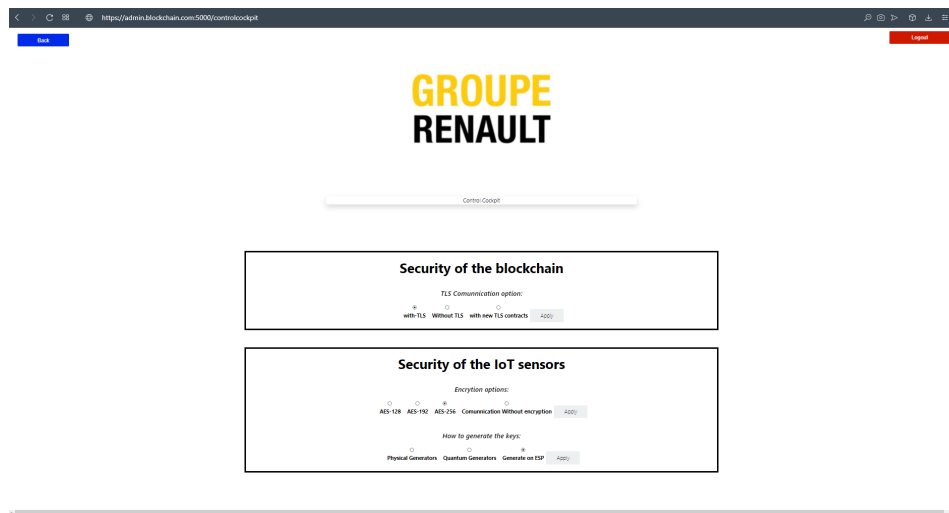


Figura 4.31: Control Cockpit

Capítulo 5

Testes e Conclusões

5.1 Desempenho dos geradores de números aleatórios

Para proceder ao teste da implementação recorreu-se a dois métodos, o primeiro quantitativo no qual foram executados testes de entropia aos geradores utilizados, bem como aos tempos de execução do programa ou ainda a cálculos para verificar a vida útil de cada gateway. Por outro lado, foram também realizadas avaliações qualitativas para verificar se a encriptação se encontrava a ser realizada e sobre o desempenho da implementação.

5.1.1 Números aleatórios do esp

Ao realizar os testes descritos na seção 4.1 sobre o ESP8266 e ESP32 obteve-se resultados semelhantes, deste modo, apenas são apresentados os resultados do ESP8266.

Tabela 5.1: Entropia e desvio padrão ESP8266

ESP8266	10 Mb	20 Mb
Média	1,084	1,071
Desvio padrão	0,0003	0,080

Ao analisar os histogramas C.2 e C.3, presentes em anexo, mas principalmente a tabela resumo 5.1, podemos retirar que o valor de entropia é aproximadamente um bit em cada oito, se executarmos a mesma análise em termos de percentagem seria obtida uma performance de 13,55 valores percentuais. Ao comparar este valor com outro tipo de geradores apresentados na tabela 5.2, conclui-se que este é bastante baixo. Outros geradores de maior qualidade apresentam valores de entropia média de cerca de 7,5. Pode-se concluir que a encriptação inicial apresentada pelo ESP8266 e ESP32 apresenta um grau de segurança extremamente baixo, dado que a forma como gera os números aleatórios torna possível prever qual será o número seguinte, comprometendo a encriptação dado estar a comprometer o Vetor de Inicialização da função AES. Como tal, ao analisar grandes quantidades de informação leva a que seja possível descobrir qual é a chave privada, e desta forma conseguir ler toda a informação podendo até fazer-se passar pelo próprio emissor.

Tabela 5.2: Dados de outros geradores

	Entropia média	Desvio padrão	Tamanho dos ficheiros (Mb)	N. de ficheiros
random.org	7,46	0,043	10	60
IDQuantique Quantis	7,46	0,11	10	199
IT QRNG-processado	7,46	0,1	10	200
IT QRNG - bruto	1,49	0,96	0,5	150

5.1.2 Números aleatórios Raspberry Pi

De forma a analisar se a opção de gerar os números aleatórios no Raspberry Pi confere um maior grau de segurança voltou-se a realizar os testes aos números gerados por este equipamento.

Tabela 5.3: Entropia média e desvio padrão Raspberry Pi

Rasp	10 Mb	20 Mb
Média	7,468	7,549
Desvio padrão	0,054	0,022

Ao analisar os histogramas C.4 e C.5 e a tabela resumo 5.3 podemos concluir que ao contrário dos resultados obtidos por parte do ESP o Raspberry Pi 3 model B apresenta média de 7,46, em termos percentuais 93,25 pontos percentuais. Após pesquisa junto do fabricante podemos apontar estes bons resultados a um firmware que o Raspberry Pi 3 model B tem incluído, denominado de BCM2835, que é responsável apenas por gerar números aleatórios, [42].

5.1.3 Números aleatórios ESP pós processado

Para avaliar a alternativa de pós processamento dos números gerados pelo ESP voltou-se a realizar nova recolha de dados e os respetivos testes tendo se obtido o valor de 7,49 para ficheiros de 20 Mb. Como tal conclui-se que esta opção também se apresenta como válida no que toca à segurança da comunicação.

Tabela 5.4: Entropia média e desvio padrão ESP pós processado

ESP pós processado	10 Mb	20 Mb
Média	7,438	7,494
Desvio padrão	0,172	0,2


Dado a melhoria obtida com o pós processamento, elaborou-se novo estudo para avaliar se seria possível aumentar a entropia do Raspberry com o mesmo método, contudo, a média obtida foi de 7,47, ligeiramente inferior à média obtida com os dados originais. Desta forma, não se apresenta como um pós processamento válido a este equipamento. Para mais informações consultar o anexo C.4.

5.1.4 Custo de operação

Apesar da já comprovada melhoria na geração das chaves privadas, os métodos implementados poderão acarretar um custo adicional de computação. Como tal, iniciou-se o estudo ao tempo de cada uma das opções disponíveis. Para realizar a recolha de tempos recorremos à biblioteca do MicroPython `utime` que deriva da biblioteca `time` original do Python, [43]. Esta biblioteca permite recolher tempos com precisão de um mili segundo. Assim para realizar os testes, foi recolhido o tempo imediatamente antes de executar o pedido e imediatamente após o pedido ser finalizado.

Para o método inicial, sugerido na implementação realizada por Diogo Costa, esta utilizava apenas o comando `uos.urandom(16)`, esta opção, como já referido, apresentava a segurança de pior qualidade, contudo quando analisado o tempo de operação este é menor do que 0,5 mili segundos, acabando por não ser detetável pelo micro controlador e retornando o valor 0, Figura 5.1.

```
66 import utime
67
68 t = utime.ticks_ms()
69 number=uos.urandom(16)
70 delta= utime.ticks_ms()
71
72 print((delta-t))
73
```



```
Shell x
>>> %Run -c $EDITOR_CONTENT
0
```

Figura 5.1: Teste à geração de IV por Diogo Costa

Ao analisar a implementação menos complexa, entenda-se a sugerida na secção 4.1.3, em que a geração do IV ocorre no ESP seguido de um pós processamento, o sistema aponta para um tempo de resposta médio de 1 mili segundo, Figura 5.2.

Por fim, realizar as medições sobre a implementação aquando de pedir um novo IV, processa-la por parte do Raspberry Pi, responder e voltar a descriptar por parte do ESP, obteve-se um valor médio de 0.294 segundos. Este valor é drasticamente maior do que as outras duas implementações, contudo, ao analisar o processo, este valor apresenta se legítimo dada a sua complexidade. Apesar do seu tempo de processamento ser quase **300 vezes mais lento** que a solução de pós processamento. É ainda de notar que caso o equipamento gerador de números se encontra-se noutra rede, pela qual fosse necessário estabelecer uma ponte de comunicação seria de esperar valores ainda maiores.

5.1.5 Verificação de encriptação

Com o intuito de executar uma verificação adicional procedeu-se à instalação do software Wireshark nas gateways bem como no computador onde se encontrava o programa a ser executado. Relativamente às gateways é visível na imagem 5.3 o endereço que foi utilizado para enviar a informação, `</esp/vals>`, sendo também visível, no canto inferior direito da imagem, marcado com uma seta a informação da mensagem, contudo, como o esperado, sem ser possível ler qualquer dado que seja.

```

64 import utime
65 t = utime.ticks_ms()
66 number_to_hash=uos.urandom(64)
67 iv=uhashlib.sha256(number_to_hash).digest()
68 iv=iv[:15]
69 delta = utime.ticks_ms()
70
71 print((delta-t)/1000)
72

```

Shell x

```

>>> %Run -c $EDITOR_CONTENT
0.001

```

Figura 5.2: Teste à geração de IV com pós processamento

The screenshot shows the Wireshark interface with a network capture of an HTTP POST request. The packet list pane shows a POST request to /esp/vals. The packet details pane shows the request structure, and the packet bytes pane shows the raw data. A red arrow points to the data field in the details pane, and another red arrow points to the hex data in the bytes pane.

No.	Time	Source	Destination	Protocol	Length	Info
24	18.845091816	192.168.10.16	192.168.10.2	HTTP	104	POST /esp/valsiv04 HTTP/1.0
25	18.845221087	192.168.10.2	192.168.10.16	TCP	54	8090 → 19795 [ACK] Seq=1 Ack=51 Win=64190 Len=0
26	18.858052702	192.168.10.2	192.168.10.16	TCP	71	8090 → 19795 [PSH, ACK] Seq=1 Ack=51 Win=64190 Len=0
27	18.858997910	192.168.10.2	192.168.10.16	HTTP	254	HTTP/1.0 200 OK (text/html)
28	18.874845670	192.168.10.16	192.168.10.2	TCP	54	19795 → 8090 [ACK] Seq=51 Ack=219 Win=1926 Len=0
29	18.878904212	192.168.10.16	192.168.10.2	TCP	54	19795 → 8090 [FIN, ACK] Seq=51 Ack=219 Win=1926 Len=0
30	18.879016243	192.168.10.2	192.168.10.16	TCP	54	8090 → 19795 [ACK] Seq=219 Ack=52 Win=64190 Len=0
31	18.896471504	192.168.10.16	192.168.10.2	TCP	58	24289 → 8090 [SYN] Seq=0 Win=2144 Len=0 MSS=536
32	18.896598691	192.168.10.2	192.168.10.16	TCP	58	8090 → 24289 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
33	19.00441816	192.168.10.16	192.168.10.2	TCP	54	24289 → 8090 [ACK] Seq=1 Ack=1 Win=2144 Len=0
34	19.145480097	192.168.10.16	192.168.10.2	HTTP	185	POST /esp/vals HTTP/1.0

Packet 34 details:

- Ethernet II, Src: Espressi_38:fa:6d (3c:71:bf:38:fa:6d), Dst: Raspberr_88:11:71 (b8:27:eb:88:11:71)
- Internet Protocol Version 4, Src: 192.168.10.16, Dst: 192.168.10.2
- Transmission Control Protocol, Src Port: 24289, Dst Port: 8090, Seq: 1, Ack: 1, Len: 131
- Hypertext Transfer Protocol
 - POST /esp/vals HTTP/1.0\r\n
 - Host: 192.168.10.2\r\n
 - Content-Length: 64\r\n
 - \r\n
 - [Full request URI: http://192.168.10.2/esp/vals]
 - [HTTP request 1/1]
 - [Response in frame: 37]
 - File Data: 64 bytes
 - Data (64 bytes)
 - Data: db9939f557818f8de655ae9b36a97a51654aad4a33493819...
 - [Length: 64]

Packet 34 bytes:

```

0000 b8 27 eb 88 11 71 3c 71 bf 38 fa 6d 08 00 45 00  ....q q 8 m . E .
0010 00 ab 00 13 00 00 ff 06 25 d7 c0 a8 0a 10 c0 a8  ....%.....
0020 0a 02 5e e1 1f 9a 00 00 19 9c d0 fc 3b e3 50 18  ..A.....; .P .
0030 08 60 af b9 00 00 50 4f 53 54 20 2f 65 73 70 2f  .. .PO ST /esp/
0040 76 61 6c 73 20 48 54 54 50 2f 31 2e 30 0d 0a 48  vals HTT P/1.0..H
0050 6f 73 74 3a 20 31 39 32 2e 31 36 38 2e 31 30 2e  ost: 192 .168.10.
0060 32 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74  2..Conte nt-Lengt
0070 68 3a 20 36 34 0d 0a 0d 0a db 99 39 f5 57 81 8f  h: 64... .9.W.
0080 8d e6 55 ae 9b 36 a9 7a 51 65 4a ad 4a 33 49 38  ..U .6.2 QeJ.J3I8
0090 19 c0 ef 87 e1 c5 00 a7 49 f4 30 8b cb ab cf c8  ....I. ....
00a0 57 03 8d 13 16 9d 85 36 20 3a 14 a8 83 17 c5 00  W.....6 :.....
00b0 c9 38 0a 7c 19 a8 19 64 49  ....8|...d I

```

Figura 5.3: Verificação de encriptação ao ESP recorrendo ao WireShark

5.2 Análise à base de dados de cada Gateway

A implementação de uma base de dados intermédia apesar de ter uma programação acessível tem um custo no que toca ao seu armazenamento. Para estudar qual seria a vida útil da memória interna de cada Raspberry Pi, sendo que cada gateway se encontra equipado com uma memória de 16 Gb e que destes 2 Gb foram reservados para a imagem do sistema sobrando assim 14 Gb que possam ser utilizados na base de dados. Importa assim saber o número de mensagens enviada por ano (N° men. / ano), para tal será necessário saber o número de ESP's ligados à mesma gateway (n° de ESP), as mensagens que são enviada por minuto por cada um (n° de men./minuto). Temos então:

$$\frac{N^{\circ} \text{ men.}}{\text{ano}} = n^{\circ} \text{ de Esp} \times \frac{n^{\circ} \text{ de men.}}{\text{minuto}} \times n^{\circ} \text{ de minutos num ano} \quad (5.1)$$

$$N. \text{ mens. ano} = 3 \times 4 \times (60 \times 24 \times 365) = 6307200 \quad (5.2)$$

Contudo importa saber quantos bytes são ocupados por ano (Bytes/ano), sendo que poderá ser armazenado até 16 bytes por mensagem

$$\frac{\text{Bytes}}{\text{ano}} = N. \text{ mens. ano} \times 16 = 100915200 \quad (5.3)$$

Vale referir que por cada dez mil mensagens recebidas (N° men. por chave), será necessário criar uma nova chave de 36 bytes, contudo, esta chave tem de ser duplicada pelo ficheiro de encriptação e desencriptação obtendo o total de chaves necessária por ano (N° de chaves / ano).

$$\frac{N^{\circ} \text{ chaves}}{\text{ano}} = \frac{N. \text{ mens. ano}}{N^{\circ} \text{ men. / chave}} \times 2 = 1262 \quad (5.4)$$

$$\frac{\text{Número de bytes em chaves}}{\text{ano}} = \frac{\text{Número de chaves}}{\text{ano}} \times \frac{\text{n. de bytes}}{\text{chave}} \quad (5.5)$$

$$\frac{\text{Número de bytes em chaves}}{\text{ano}} = 1262 \times 36 = 45412 \quad (5.6)$$

Na tabela 5.5 é visível uma síntese da memória utilizada com diferentes configurações de utilização, sendo que a utilização do espaço aumenta sempre de forma linear.

Tabela 5.5: Armazenamento necessário a cada base de dados intermédia

N. de esp	Mens./ (ESP x minuto)	Mens./ ano	Gb em cha- ves/ano	Chaves/ ano	Gb/ ano	Vida útil [anos]
1	4	2102400	0,03	420	0,03	416,0
3		6307200	0,10	1261	0,10	138,7
6		12614400	0,20	2523	0,20	69,3
10		21024000	0,34	4205	0,34	41,6
10	8	42048000	0,67	8410	0,67	20,8
20		84096000	1,35	16819	1,35	10,4

Desta forma podemos concluir que com três ESP ligados a uma gateway os quatorze Gb de memória disponíveis teriam uma vida útil de 138,7 anos. A implementação poderá ser expandida para 10 ESP a recolher dados e aumentar a cadência para oito mensagens por minuto por cada equipamento que ainda teria uma vida superior a 20 anos sem necessidade de manutenção.

5.2.1 Leitura de Dados

A utilização do ficheiro de leitura ocorre de forma simples e intuitiva, sendo que o utilizador pode escolher descriptar por linhas ou pelo timestamp. Quando opta por linhas pode ainda escolher se pretende todas as linhas, apenas uma ou um determinado número de linhas, como exemplificado na Figura 5.4, em que se seleccionou a opção de ler mais do que uma linha, many, a iniciar na posição 0 e ler 3 linhas no total.

```

WELCOME!!!
Insert what you want to read, type:
-> lines to read depending on the number of the entry
-> date to read depending on the date
lines
Insert what you want to read, type:
-> all to read everything;
-> 1 to read only one line
-> many to read more than one
many
Type your First line
0

Type the total number of lines to read
3

Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'33' sensor= b'02' variavel 1= b'459' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'33' sensor= b'02' variavel 1= b'400' variavel2= b'1'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'33' sensor= b'02' variavel 1= b'741' variavel2= b'1'

```

Figura 5.4: Processo de leitura nas gateway's através da linha de comandos

Ao descriptar toda a informação de uma só vez, para 100 linhas obtêm-se um tempo médio de resposta de 0.02 segundos. Se pretendemos ler por exemplo três linhas obtemos tempos de resposta de 0.015. Quando se pretende uma linha específica este tempo diminui para cerca de 0.001 segundos em média. Para avaliar como se comporta o tempo de resposta, retiramos mais do que uma amostragem, visível na Figura 5.6, onde se pode retirar que o tempo de resposta aumenta de forma linear consoante o número de linhas pedido como seria de esperar. É ainda de notar uma declive diferente no início da reta, esta diferença é devida a tempos de operação muito reduzidos que dificultaram a medição exata, obtendo o mesmo tempo de leitura para o pedido de uma e duas linhas.

Quando o utilizador pretende que apenas leia uma parte da informação dependendo no timestamp que esta tem agregado deve de seleccionar, date, seguido da data pela qual pretende iniciar a leitura, utilizando dois dígitos para cada componente da data, tendo em conta que deve de referir em primeiro lugar o ano, seguido do mês, do dia, da hora e por fim do minuto, como visível na Figura 5.5. Esta leitura apresenta se com tempos semelhantes ao processo de leitura de todas as linhas, dado ser necessário descriptar toda a informação para que se possa ter acesso à data em que a informação foi gravada em sistema.

```

WELCOME!!!
Insert what you want to read, type:
-> lines to read depending on the number of the entry
-> date to read depending on the date
date
Insert the initial date that you want to read with 2 numbers for year, 2 numbers for month, 2 numbers for day, 2 numbers for hour
2105251005
Insert the final date that you want to read with 2 numbers for year, 2 numbers for month, 2 numbers for day, 2 numbers for hour
2105251007
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'515' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'896' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'608' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'994' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'625' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'968' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'489' variavel2= b'1'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'751' variavel2= b'1'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'138' variavel2= b'0'
Info Guardada== Ano= b'21' mês= b'05' dia= b'25' hora= b'10' minuto= b'06' sensor= b'02' variavel 1= b'523' variavel2= b'0'

```

Figura 5.5: Processo de leitura nas gateway's por timestamp através da linha de comandos

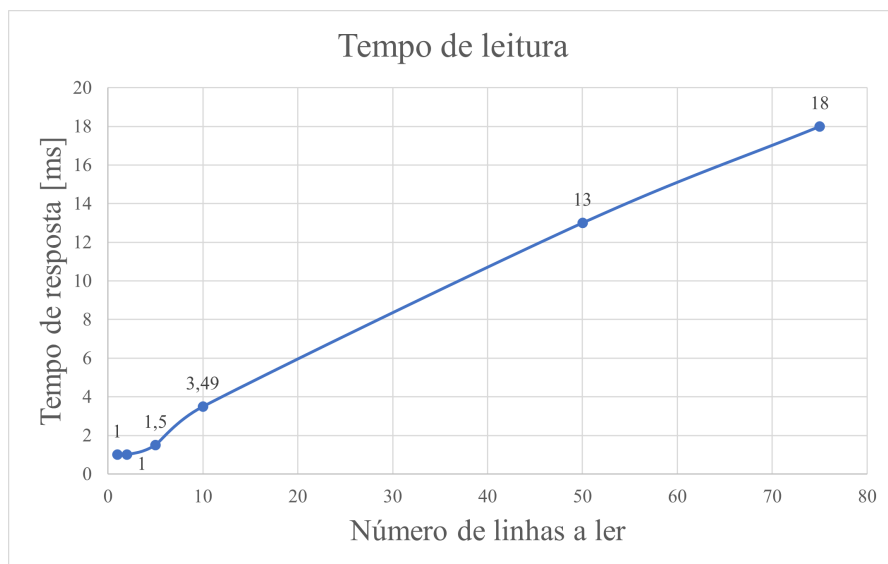


Figura 5.6: Tempos de resposta consoante o número de linhas pedido

5.3 Conclusões

Ao analisar os métodos propostos para a renovação de chaves e geração de IV, estes apresentam-se com diferenças marcantes no que toca ao método de processamento, contudo o pedido de chaves ao Raspberry não se apresenta como gargalo à implementação dado que cada ESP se encontra a fazer pedidos com intervalos de 15 segundos de «descanso», podendo ser ajustado o tempo em que está inativo de forma a que não comprometa os ciclos de amostragem de dados. Este intervalo de tempo também não se encontra comprometido pelos tempos de escrita na blockchain que se apresentam inferiores a dois segundos. Apesar diferença de entropia dos números obtidos pelo ESP e pelo Raspberry Pi ser pouco significativa, cerca de 0.06 para ficheiros de 20 Mb, este último possui um gerador de números próprio. Como tal, esta opção pode ser considerada de maior qualidade, pois como possui um maior número de fontes de entropia, a dificuldade de um possível atacante controlar estes sensores e, assim, alterar a entropia dos números é consideravelmente superior. É de extrema importância referir que, a qualidade da segurança da transmissão da informação, bem como o armazenamento desta de forma encriptada, é assegurada pelos testes executados aos geradores de números, que comprovaram que as soluções propostas são de facto seguras.

Em suma podemos concluir que as duas implementações são consideradas válidas e de boa qualidade não comprometendo o funcionamento da implementação e conferindo-lhe um grau de segurança muito maior do que a implementação inicial. Esta melhoria também foi conseguida em parte pela linguagem de programação escolhida que fornece um elevado número de bibliotecas e capacidade de programação e pós processamento da informação de forma rápida e simples. Estes dois métodos permitem ainda alargar a implementação a outros micro controladores ou outro tipo de gateways dada a sua larga usabilidade. Ao analisar a utilização das bases de dados implementadas às gateways, estas apresentam um tempo de vida útil bastante adequado até à sua primeira manutenção. A sua interação com o utilizador também ocorre de forma bastante simples e intuitiva bem como garantindo um rápido acesso à informação sem colocar em causa a segurança da informação.

De forma geral, as melhorias implementadas melhoraram a segurança da implementação bem como a experiência de utilização no que toca ao HMI dado que passamos a ter compatibilidade com todos os dispositivos sem comprometer os tempos de resposta. Outro ponto em questão é a modularidade da implementação que foi sempre tido em conta garantindo, de modo geral, a fácil escalabilidade para mais sensores, mais linhas ou mesmo mais fábricas. Este ponto foi provado pela expansibilidade da implementação para duas gateways. É ainda importante referir o aumento de dados a ser recolhido pelos equipamentos do patamar IoT bem como o tratamento interrupto da informação por parte dos nós, comprovando assim que a implementação pode ser aplicada a um ambiente fabril para proceder a testes. Por último, esta opção apresenta-se como um bom custo benefício para uma cadeia de rastreabilidade baseada em blockchain, sendo o seu único custo associado ao hardware para a implementação dado que em todo o percurso foram utilizadas soluções de software «open-source» não tendo custo associado. A implementação desenvolvida foi pensada de modo a que no seu normal funcionamento esta não necessite de manutenção, não estando assim associado um custo direto, sendo apenas necessário intervenção física aquando de um possível erro no sistema, sendo então necessário reiniciar as comunicações. Como este será um acontecimento que apenas ocorrerá

de forma esporádica não lhe foi associado um custo.

5.4 Sugestões futuras

O desenvolvimento de computação quântica tem conquistado cada vez mais enfoque por parte das empresas. Esta tecnologia irá trazer a possibilidade de resolver problemas por métodos de tentativa erro, o método chamado na gíria de força bruta, colocando em causa a encriptação estabelecida nas comunicações. Como tal será necessário acompanhar a evolução desta tecnologia bem como as soluções que serão apresentadas para a contrariar, soluções essas que já se encontram a ser pensadas por empresas como a Google ou a OpenSSL. Com o intuito de perceber ainda mais a geração de números aleatórios seria interessante executar novos testes alterando de forma premeditada as condições em que cada equipamento se encontra, analisando as alterações resultantes à entropia destes.

Outro ponto a analisar, passa por realizar uma terceira opção de implementação de chaves em que o Raspberry Pi estabelecerá uma comunicação segura com um gerador de números quânticos, através de API's, possibilitando assim o uso de chaves provenientes de geradores quânticos. Quanto à geração de números aleatórios, referente aos ESP's utilizados, sugere-se o estudo sobre a possibilidade de implementar como fontes de entropia dados provenientes da linha de produção, dados esses como vibrações, temperaturas, consumo de energia, entre outros. Desta forma, a entropia seria gerada por mais fontes, com maior dificuldade de manipulação por um atacante, sendo que um maior número de fontes também significaria um sistema mais complexo de atacar. Ainda em termos de segurança, sugere-se analisar a implementação de chaves simétricas de maior comprimento, sendo para isso necessário analisar se esta alteração seria válida dada a baixa capacidade de processamento dos ESP poder comprometer os tempos de recolha de dados, ou a sua baixa memória interna e baixa memória RAM poderem comprometerem o funcionamento da implementação. Outro ponto importante a analisar seria a implementação de TLS nos ESP, este ponto foi abordado nesta dissertação, mas dada a falta de bibliotecas disponíveis e o curto tempo disponível não foi possível a sua implementação. Surge ainda como ponto de interesse a renovação de certificados digitais bem como a criação de infraestruturas de distribuição de chaves públicas. Esta implementação iria aumentar a segurança das comunicações bem como a sua maior facilidade em expandir para mais equipamentos.

Sobre a base de dados desenvolvida sugere-se, como melhoria, o desenvolvimento de uma função que permita prever um intervalo de dados a desencriptar aquando do pedido de leitura numa determinada data, entenda-se, realizar a leitura entre determinados timestamp's, de modo a que este processo ocorra de forma mais rápida e eficiente em vez de realizar a desencriptação total.

Quanto à blockchain esta apenas foi testada com recurso a três nós, seria ainda importante realizar mais testes, incorporando mais nós, bem como dispersá-los por diferentes redes, analisando os tempos de resposta bem como a validação do processo de partilha de informação.

No que toca à troca de pedidos HTML seria interessante analisar a possibilidade de uma metodologia «Load Balance» para que a implementação se pudesse ajustar por si só à necessidade de processamento, por outras palavras, quando a linha de produção estiver com uma maior cadência de peças, fosse alocado mais recursos para tratar esta

informação de forma rápida, contudo, por exemplo, se uma linha estivesse em manutenção os recursos que são normalmente alocados ao processamento desta linha fossem alocados a outros processos.

Capítulo 6

Bibliografia

- [1] “Renault Cacia,” accessed: 2021-03-10. [Online]. Available: <https://www.renault.pt/renault-cacia.html>
- [2] D. F. Julião and D. Rocha, “Proposal of an automatic traceability system, in Industry 4.0,” Ph.D. dissertation, Aveiro, 2018.
- [3] “Base de dados SQLVS NoSQL,” accessed: 2021-01-15. [Online]. Available: <https://debugeverything.com/diferenca-base-de-dados-relacional-e-nao-relacional/>
- [4] S. Martin and A. Lembke, *Moderation Management*. Pearson, 2015.
- [5] J. Ali, “Issues in supply chain management in Indian agriculture,” *Food for Policy*, vol. 83, pp. 195–225, 2008.
- [6] P. Stasa, F. Benes, J. Svub, Y. S. Kang, J. Unucka, L. Vojtech, V. Kebo, and J. T. Rhee, “Ensuring the visibility and traceability of items through logistics chain of automotive industry based on AutoEPCNet usage,” *Advances in Electrical and Electronic Engineering*, vol. 14, no. 4Special Issue, pp. 378–388, 2016.
- [7] K. Storchmann, “Introduction to the Issue,” *Journal of Wine Economics*, vol. 14, no. 2, pp. 131–132, 2019.
- [8] S. A. Por Alfred J. Menezes, Paul C. van Oorschot, *Handbook of Applied Cryptography*. CRC Press, 1997.
- [9] J.-P. Aumasson, *Serious Cryptography*. No Starch Press, Inc., 2017.
- [10] B. A. Forouzan, *TCP/IP Protocol Suite*, 4th ed. Higher Education, 2010.
- [11] S. J. Nielson and C. K. Monson, *Practical Cryptography in Python*. Austin, TX, USA: APress, 2019. [Online]. Available: <http://www.allitebooks.org/>
- [12] “UUID4,” 2021, accessed: 2021-04-28. [Online]. Available: <https://docs.python.org/3/library/uuid.html>
- [13] “NIST Laboratory,” 2020, accessed: 2021-05-30. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/>

- [14] E. Barker and J. Kelsey, "Recommendation for the Entropy Sources Used for Random Bit Generation, NIST Special Publication 800-90B," 2012.
- [15] S. A. Wilber, "Pure Quantum True Random Number Generators," , pp. 1–28, 2014. [Online]. Available: <https://comscire.com/>
- [16] Y. Zou, T. Meng, P. Zhang, W. Zhang, and H. Li, "Focus on Blockchain: A Comprehensive Survey on Academic and Application," *IEEE Access*, vol. 8, pp. 187 182–187 201, 2020.
- [17] F. Restuccia, S. D'Oro, S. S. Kanhere, T. Melodia, and S. K. Das, "Blockchain for the internet of things: Present and future," *arXiv*, no. November, 2019.
- [18] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, p. 352, 10 2018.
- [19] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain," *CEUR Workshop Proceedings*, vol. 2058, no. January 2018, 2018.
- [20] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.
- [21] Hyperledger, "About Hyperledger," p. 1, 2020. [Online]. Available: <https://www.hyperledger.org/about>
- [22] Hyperledger Project, "Fabric whitepaper," ., p. 2, 2020. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_{_}fabric_{_}whitepaper.pdf
- [23] "Ethereum," accessed: 2021-03-10. [Online]. Available: <https://ethereum.org/en/about/>
- [24] X. Xu, I. Weber, M. Staples, X. Xu, I. Weber, and M. Staples, *Existing Blockchain Platforms*. Springer, 2019.
- [25] "Corda," accessed: 2021-03-10. [Online]. Available: <https://www.corda.net>
- [26] L. Maria, "An application of blockchain smart contracts and IoT in logistics," Ph.D. dissertation, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 2019. [Online]. Available: <https://github.com/joaomlourengo/novathesis>
- [27] J. M. M. L. Mendes, "Security techniques for the Internet of Things," 2013.
- [28] J. P. R. R. de Amaral, "Autenticação em interações com dispositivos IoT Authentication in interactions with IoT devices," Ph.D. dissertation, Universidade de Aveiro, 2019.
- [29] D. F. Duarte and Costa, "Proposal of a Traceability 4.0 System in Renault CACIA Using Blockchain Technologies," Ph.D. dissertation, Aveiro, 2020.

- [30] “random linux.” [Online]. Available: <https://linux.die.net/man/4/random>
- [31] “UEFI entropy gathering protocol.” [Online]. Available: <https://docs.microsoft.com/en-us/windows-hardware/drivers/bringup/uefi-entropy-gathering-protocol>
- [32] “Windows CryptoApi.” [Online]. Available: <https://www.mail-archive.com/openssl-dev@openssl.org/msg21829.html>
- [33] “ESP32Pinout,” accessed: 2021-04-01. [Online]. Available: <https://www.botnroll.com/pt/esp/2149-modulo-wifi-esp-12e-esp8266.html>
- [34] “ESP32.” [Online]. Available: <https://www.studiopieters.nl/esp32-pinout/>
- [35] “No Title,” accessed: 2021-04-01. [Online]. Available: <https://www.botnroll.com/pt/esp/2149-modulo-wifi-esp-12e-esp8266.html>
- [36] R. Chokshi, “Datasheet RS-MPU-6000A-00,” p. 52, 2015. [Online]. Available: www.invensense.com
- [37] Invensense, “MPU-9250 Register Map and Descriptions Revision 1.4,” pp. 1–55, 2014.
- [38] —, “MPU-9250 Register Map and Descriptions Revision 1.4,” pp. 1–55, 2014.
- [39] N. O. T. Under and R. Control, “MPU Hardware Offset Registers Application Note,” pp. 1–8, 2014.
- [40] “GY-91.” [Online]. Available: <https://electropeak.com/learn/interfacing-gy-91-9-axis-mpu9250-bmp280-module-with-arduino/>
- [41] “Raspberrypi,” accessed: 2021-05-29. [Online]. Available: <https://www.raspberrypi.org>
- [42] “BCM2835,” accessed: 2021-05-15. [Online]. Available: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/README.md>
- [43] “utime,” accessed: 2021-05-20. [Online]. Available: <https://docs.micropython.org/en/latest/library/utime.html>
- [44] R. T. Sataloff, M. M. Johns, and K. M. Kost, *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*, O. V. Friess and Peter, Eds. Riber Publishers, 2016.
- [45] P. V. C. Pinheiro, “Indústria 4.0,” 2016. [Online]. Available: https://www.compete2020.gov.pt/destaques/detalhe/Industria{}_4ponto0
- [46] A. Gilchrist, *The industrial internet of things*. Apress, 2019, vol. 33, no. 5.
- [47] “RFC 4122,” accessed: 2021-05-30. [Online]. Available: <https://joinup.ec.europa.eu/collection/ict-standards-procurement/solution/uuid-rfc-4122-universally-unique-identifier-uuid-urn-namespace/about>
- [48] M. Akmalayah, *Distributed Network Systems From Concepts to Implementations*. Springer, 2004, vol. 15, no. 9.

- [49] D. George, “MicroPython,” accessed: 2021-05-15. [Online]. Available: <https://micropython.org/download/>
- [50] “MongoDB linux.” [Online]. Available: <https://docs.mongodb.com/manual/administration/install-on-linux/>
- [51] GS1, “GS1 DataMatrix Guideline Overview and technical introduction to the use of GS1 DataMatrix,” pp. 8–12, 2018. [Online]. Available: <https://www.gs1.org/docs/barcodes/GS1{ }DataMatrix{ }Guideline.pdf>
- [52] S. L., *RFID Sourcebook By Sandip*. Prentice Hall PTR, 2005.

Anexos

Apêndice A

Conceitos alargados

A.1 IoT

«Internet of Things», vulgarmente referida como IoT, ou em português, «Internet das Coisas», teve o seu início em meados de 1973 quando o primeiro dispositivo RFID foi patenteado. Contudo apenas foi utilizado o seu conceito em meados de 1999, e formalmente apresentado em 2005 pela 'International Telecommunication Union (ITU), tendo o seu conceito e definição sofrido algumas alterações ao longo dos anos, tendo-se chegado ao consenso de que não é apenas algo tecnológico. Desta forma, cobre áreas como software, aplicações e serviços, modelos de negócios e de impacto social. [7]

No livro [44], pode se encontrar a definição para IoT como «a rede de objetos físicos que contêm tecnologia de comunicação e de sensores ou que interagem com o ambiente ao seu redor». Em suma, é um conceito que tem como objetivo interligar dispositivos de uso diário, com equipamentos de transportes, eletrodomésticos e a Internet de forma a ler todo o ambiente à sua volta com o intuito de utilizar essa informação de forma inteligente e benéfica [45].

A.1.1 Indústria 4.0

Indústria 4.0 surge com a capacidade de conectar os meios operacionais existentes nas empresas com as tecnologias de informação emergentes. Desta forma, existe a possibilidade de processar largas quantidades de informação em tempo real recorrendo a algoritmos. Esta inovação permite desenvolver uma indústria mais eficiente no que toca ao processo, com maior conexão à sua cadeia de fornecedores, maior manutenção preventiva, bem como alterar a produção de mais facilmente, consoante o pedido de cada cliente.

Indústria 4.0 é um conceito que foi sugerido pela Academia de Ciência e Engenharia alemã e apresentada como a integração de sistemas de produção com tecnologia de informação, baseada em sistemas ciber-físicos, IoT , bem como em produção baseada em informação sediada em nuvem, (do inglês, Cloud-Based Manufacturing, CBM) [46]. Um dos maiores objetivos da IoT é possibilitar a descentralização de decisões, criando sistemas de produção que se adaptam sozinhos. Como tal a indústria 4.0 tem por base alguns opções tecnológicas que serão referidas a seguir.

A.1.2 «Big Data»

Os sistemas existentes aos dias de hoje possibilitam a aquisição de quantidades extremas de informação, que posteriormente será tratada e analisada, fornecendo estatísticas importantes, possibilitando a tomada de decisões com mais precisão e confiança [45].

A.1.3 «Cloud Computing»

Estas plataformas são o nível mais alto das implementações, sendo que tem de possuir a capacidade de armazenar a «Big Data» criada por todos os dispositivos IoT e que chega a este nível por meio das gateways. Este armazenamento é efetuado em servidores cujo acesso à informação é feito remotamente através da Internet [45]. A este nível, poderá ser incluído ainda uma interface gráfica que permite ao utilizador aceder a dados predefinidos ou ainda navegar através dos dados existentes.

Todo o software e hardware utilizado na implementação deve de ser modular e independente, conferindo assim escalabilidade da implementação verticalmente ou horizontalmente. Isto é, caso seja necessário alargar o número de dispositivos a recolher dados, ou expandir a informação disponibilizada no patamar superior, tal deve de ser possível sem grande dificuldade.

Apêndice B

RFC 4122

O RFC4122 é uma norma europeia que define requisitos para que se possa obter um identificador único de 128 bits no espaço e através do tempo. Neste identificador cada bit utilizado pode ser definido por qualquer uma das variáveis disponíveis para o caracterizar. Esta especificação tem origem no DCE, «Distributed Computing Environment» sendo que aos dias de hoje é largamente utilizada pela Microsoft, [47]. Os UUID quando gerados de acordo com esta norma podem ser considerados únicos para fins práticos sem ter de depender a sua singularidade de uma entidade central que a verifique. É importante referir que a probabilidade de existir um UUID repetido não é zero, contudo, é próxima o suficiente do valor zero para que não seja significativo, [47].

Este UUID tem por base codificação de 256 símbolos, ou seja, têm por base «Extended-Ascii», como os símbolos com índice maior que 128 poderão não ter representação gráfica é comum encontrar estes UUID's representados em formato hexadecimal acompanhados de hífen's a separar blocos de texto, obtendo um total de 36 caracteres dos quais 32 alfanuméricos, [48].

Página em branco intencional.

Apêndice C

Resultados dos testes aos números aleatórios

Para proceder à recolha de dados a partir do ESP, recorreu-se a um broker, Mosquito, dado permitir publicação de dados sem necessidade de tratamento e ter tempos de resposta mais rápidos que pedidos HTML, como tal procedeu-se à execução do seguinte código, visível na figura C.1.

C.1 ESP

OS resultados obtidos nos testes executados ao ESP8266 são visíveis nas figuras C.2 C.3. Para ficheiros de 10 Mb existe um maior número de ficheiros com entropia média registado no valore de 1.0836, para 16 ficheiros. Já para ficheiros de 20 Mb a entropia mais registada acontece a 1.0822 com um total de 24 ficheiros.

C.2 Raspberry Pi

Resultados obtidos nos testes executados ao Raspberry são visíveis nas figuras C.4 e C.5. Através dos histogramas podemos afirmar que o valor mais plausível de entropia para ficheiros de 10Mb é de 7,5 valores, dado que este mesmo valor foi obtido em 35 dos 60 ficheiros. Já nos ficheiros de 20Mb o valor mais provável divide-se em 7,55 e 7,57, dado que os dois valores tiveram o mesmo número de ficheiros, nomeadamente 10.

C.3 ESP pós processado

Resultados obtidos nos testes executados aos valores do ESP após processamento com recurso à função SHA256 são visíveis nas figuras C.6 e C.7. Ao analisar estes dois histogramas pode-se afirmar que o valor mais provável de entropia é de 7,56 valores, tendo sido obtido através de 12 ficheiros em ambas as amostragens.

C.4 Raspberry Pi pós processado

Após os excelentes resultados obtidos com o pós processamento ao ESP procurou-se avaliar se ao realizar o mesmo tratamento aos dados se a entropia dos números deste

```
1 import network
2 import esp
3 from time import sleep
4 import gc
5 import os
6 from mqtt import MQTTClient
7
8 programa=1
9 contador=0
10 contador1=0
11
12 # Disable on bootloader mode OS debug
13 # Enable garbage collector (free unused mem.)
14 esp.osdebug(None)
15 gc.collect()
16
17 SSID = "Vodafone-0E3E5C"
18 PASSW = "-----"
19
20 #-----
21 def do_connect():
22     STATION = network.WLAN(network.STA_IF)
23     STATION.active(True)
24     STATION.ifconfig()
25     STATION.ifconfig(('192.168.1.136', #ip do esp
26                       '255.255.255.0', #netmask
27                       '192.168.1.1', #ip do router - gateway
28                       '0.0.0.0'))
29
30     if not STATION.isconnected():
31         print('connecting to network...')
32
33         STATION.connect(SSID,PASSW)
34         while not STATION.isconnected():
35             pass
36         print('network config:', STATION.ifconfig())
37
38 def random_numbers():
39     while contador1<=1000
40         cc=str(uos.urandom(1))
41         c=c+cc
42         client.publish("numeros", c)
43         sleep(0.01)
44         gc.collect()
45
46 #-----
47 do_connect()
48 client= MQTTClient("ESP8266","192.168.1.79")
49 client.connect
50 client.connect()
51
52 while programa == 1:
53     random_numbers()
54     gc.collect()
55     sleep(15)
```

Figura C.1: Gerador de números aleatórios ESP

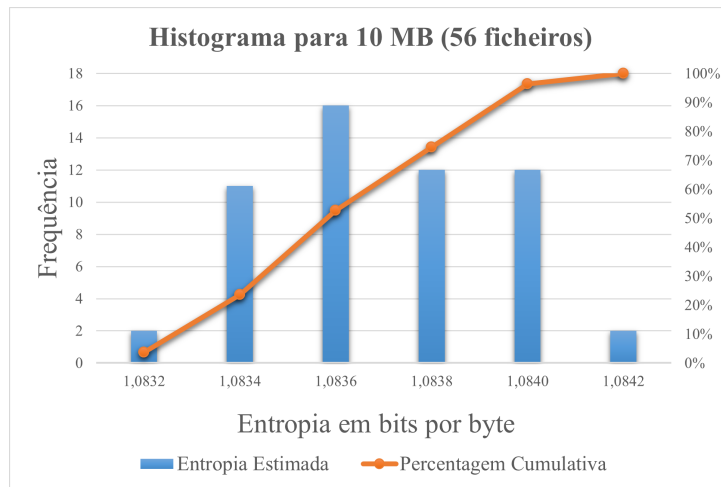


Figura C.2: Entropia ESP8266 10 Mb por ficheiro

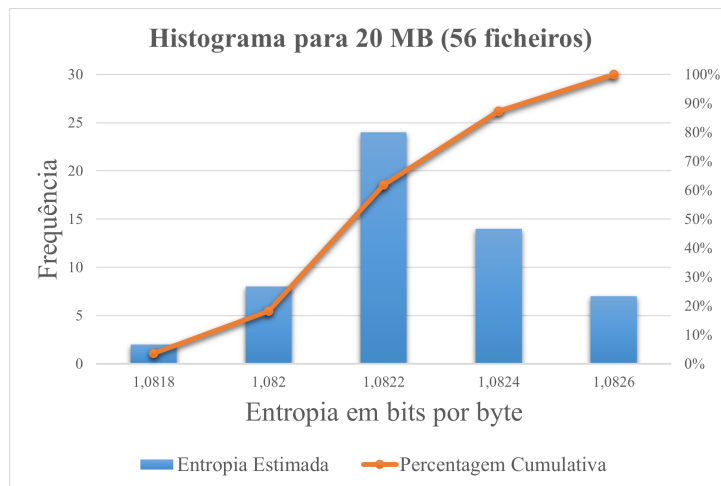


Figura C.3: Entropia ESP8266 20 Mb por ficheiro

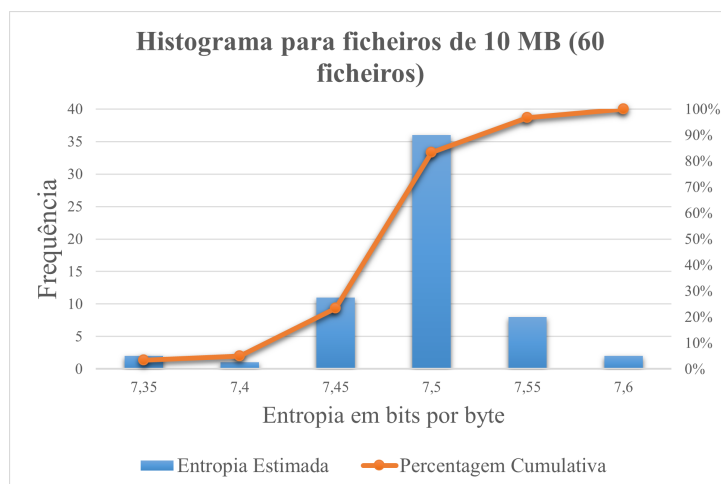


Figura C.4: Entropia Raspberry Pi 10 Mb por ficheiro

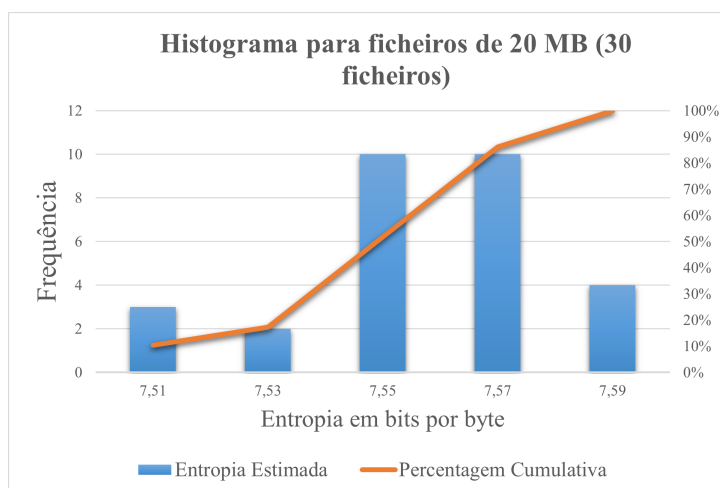


Figura C.5: Entropia Raspberry Pi 3 model B 20 Mb por ficheiro

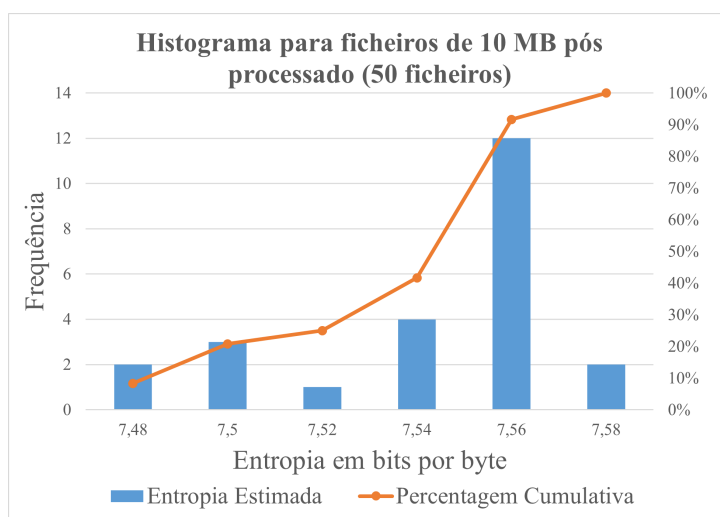


Figura C.6: Entropia ESP8266 10 Mb por ficheiro pós processado

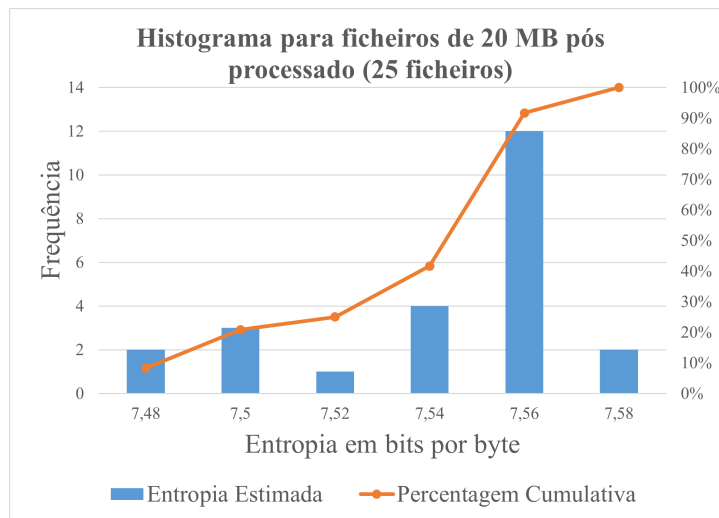


Figura C.7: Entropia ESP8266 20 Mb por ficheiro pós processado

gerador aumentaria. Os dados são visíveis nos histogramas C.8 e C.9. Através destes histogramas podemos concluir que a entropia média obtida é de 7,47, não acrescentando valor à implementação sugerida.

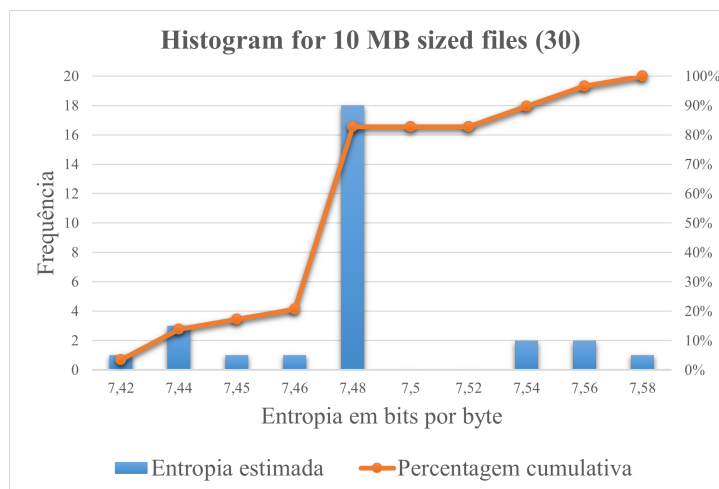


Figura C.8: Entropia Raspberry Pi 10 Mb por ficheiro pós processado

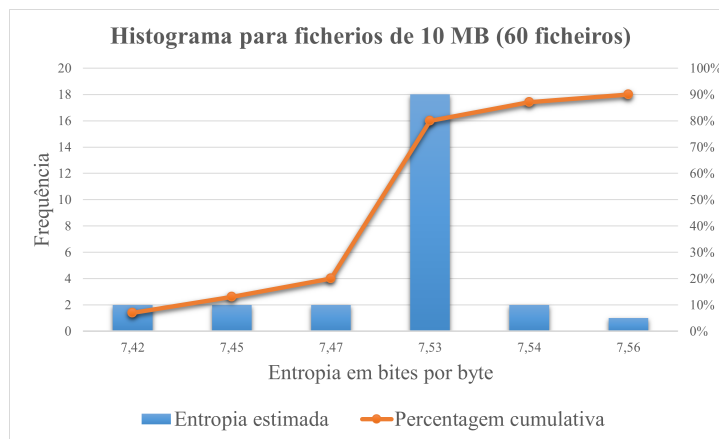


Figura C.9: EntropiaRaspberry Pi 20 Mb por ficheiro pós processado

Apêndice D

MicroPython

D.1 O que é MicroPython

MicroPython é uma implementação simples e eficiente da linguagem de programação Python 3 que inclui algumas das bibliotecas encontradas em Python tendo sido otimizado para ser executado em micro controladores e em ambientes restritos.

Esta linguagem está repleta de recursos avançados, como interface interativa, normalmente denominada de «debug», geradores de números de precisão arbitrária, variáveis do tipo lista, tratamento de exceções e muito mais. No entanto, é compacto o suficiente para caber e executar em apenas 256k de memória interna e 16k de RAM. O MicroPython pretende ser o mais compatível possível com o Python normal para permitir comuta de código com facilidade do desktop para um micro controlador.

D.2 Ambientes de trabalho

Os ambientes de trabalho mais conhecidos para trabalho diretamente com MicroPython passam pelo **Visual Studio Code**, **Thonny IDE** e **PyCharm-Community**. Todos os sistemas possibilitam criar o código, transferir e executar em «collaborative browser», ou seja, executar o programa e conseguir ler valores processados através de uma interface no desktop. Através de uma pesquisa, e por ser também utilizado nos Raspberry Pi OS, a escolha do ambiente de programação recaiu sobre o Thonny IDE.

D.3 Instalação

Para proceder à instalação será necessário em primeiro lugar fazer o download do arquivo que corresponde ao equipamento em questão. Tal pode ser feito na página do software, [49]. A instalação pode ser feita diretamente pela linha de comandos ou com recurso ao Thonny IDE. Para proceder à instalação através do Thonny é necessário conectar o dispositivo ao computador e saber qual é a porta COM a que este se encontra ligado. Para tal podemos ir a **Menu Iniciar - > Gestor de dispositivos -> Portas**, como visível na figura D.1.

Após saber qual é a COM desejada, é necessário abrir o Thonny e na barra superior selecionar **Run ->Text Interpreter**, irá abrir a janela visível na figura D.2. De seguida é necessário selecionar em que dispositivo será instalado, exemplificado pela seta marcada

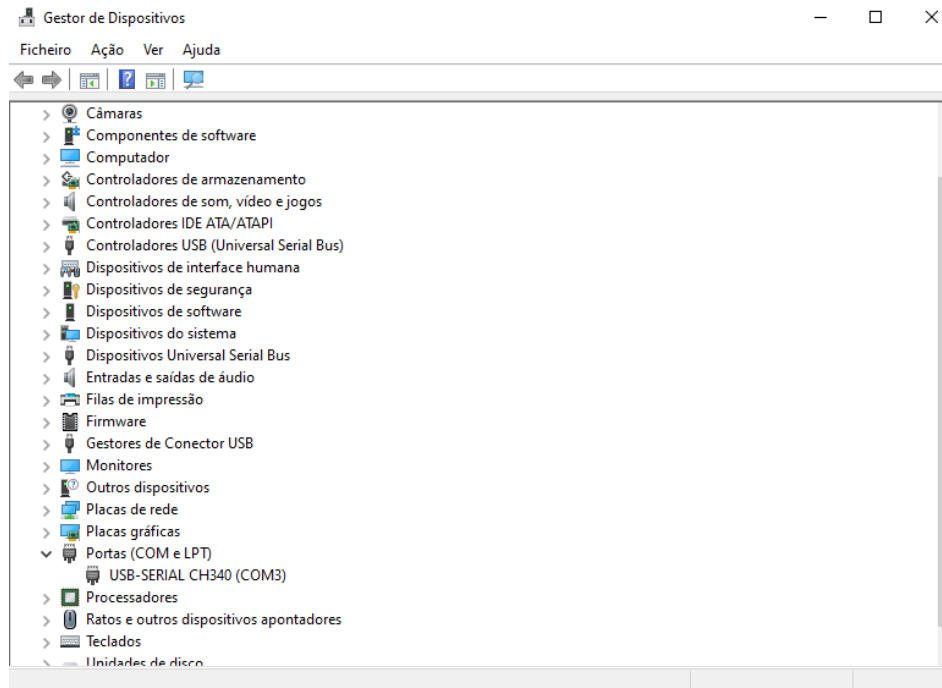


Figura D.1: Porta COM

a 2, depois será necessário selecionar a porta COM a que o dispositivo está conectado e por fim selecionar «Install or update Firmware». Irá abrir uma nova janela onde deverá de selecionar uma vez mais a porta COM caso esta não se encontre já selecionada e de seguida carregar em **Browse...** para selecionar o ficheiro que tenha descarregado. Nas opções **Flash mode** deveram de coincidir com as exemplificadas na figura D.3.

Após a instalação, se tudo tiver corrido da forma correta não serão fornecidas mensagens de erros. Para testar o módulo, na linha de comandos deverá de escrever `help('modules')`, se tudo tiver corrido da forma correta será amostrado uma lista com todos os módulos existentes. Caso contrário deverá de consultar a webgrafia do software, [49].

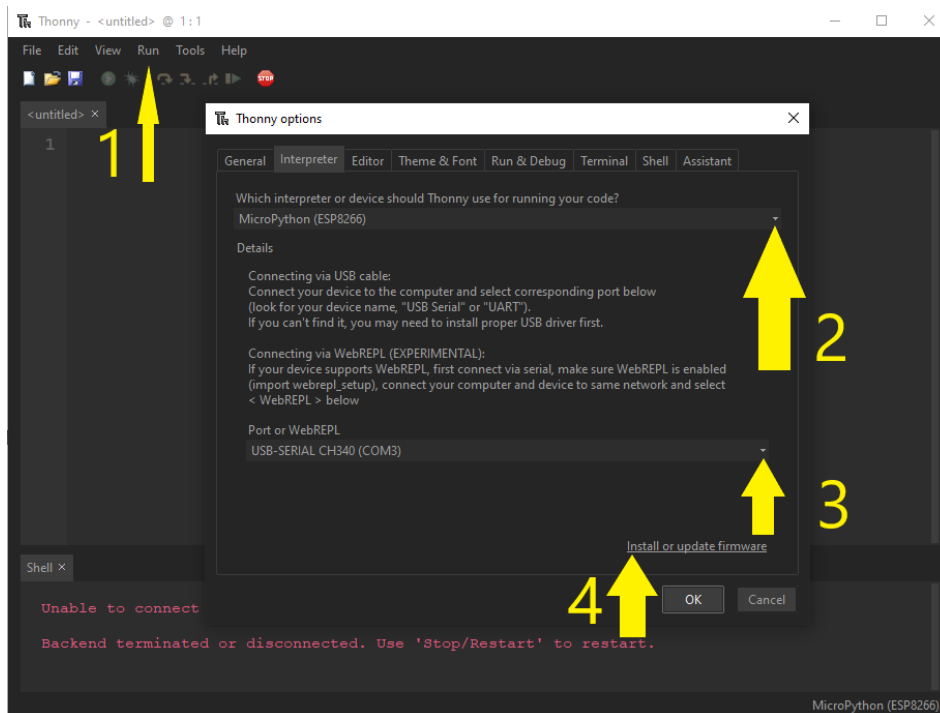


Figura D.2: Processo de instalação MicroPython

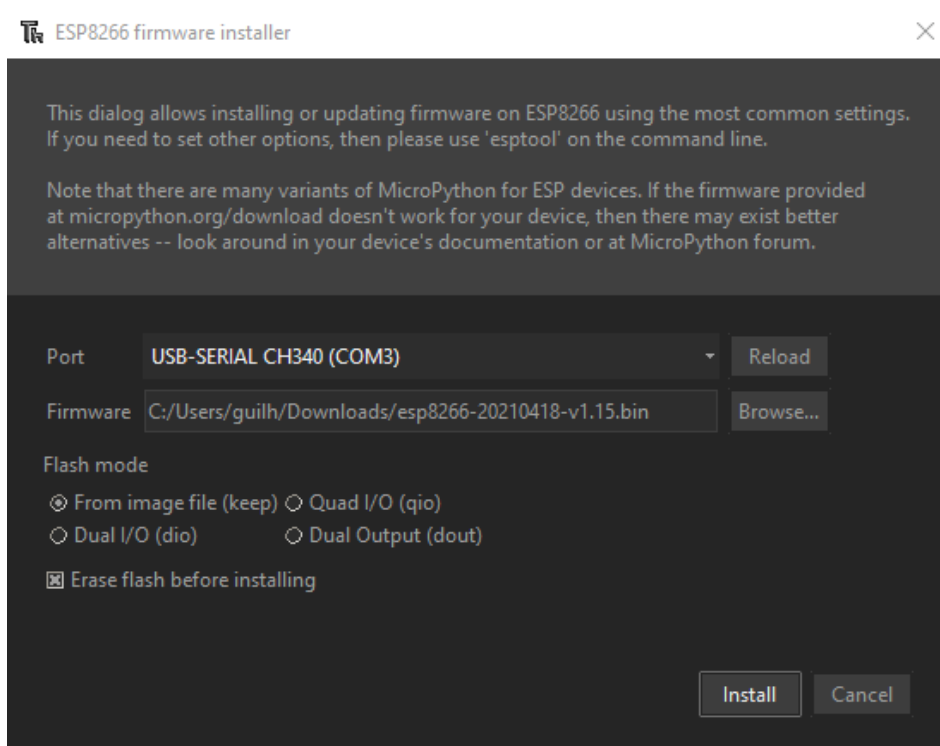


Figura D.3: Processo de instalação MicroPython - parte dois

Página em branco intencional.

Apêndice E

Configuração Raspberry Pi para comunicação RS232

Em primeiro lugar deverá de garantir que o sistema se encontra atualizado. Após isto, na linha de comandos deverá de digitar **sudo raspi-config** e deverá de obter a janela presente na figura E.1.

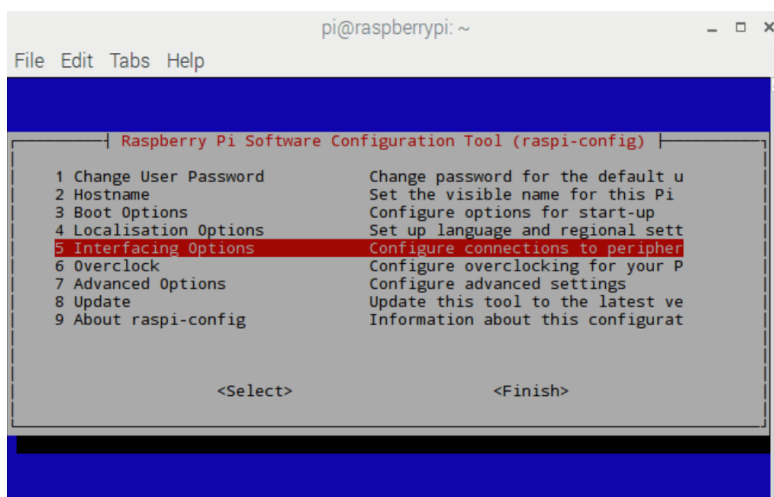


Figura E.1: Programação de Raspberry Pi para comunicação RS232 - parte 1

De seguida deverá de seleccionar a opção **5 Interfacing Options**. Na nova janela, figura E.2, deverá de seleccionar a opção **P6 Serial**. Na janela seguinte aparecerá a pergunta visível na figura E.3 à qual deverá de responder **<No>**. De seguida será apresentado nova questão, visível na figura E.4 à qual deverá de responder **<Yes>**. Após isto deverá de reiniciar o equipamento. Para verificar se a configuração foi bem executada poderá ligar o equipamento às respetivas portas, figura 4.7, e executar o comando **dmesg | grep tty** que deverá de apresentar a mensagem **ch341-uart converter now attached to ttyUSB0**. Para realizar a confirmação da porta que deverá de utilizar na programação poderá executar o comando **ls -l /dev** e procurar a linha respetiva do serial0, como exemplificado na imagem E.5

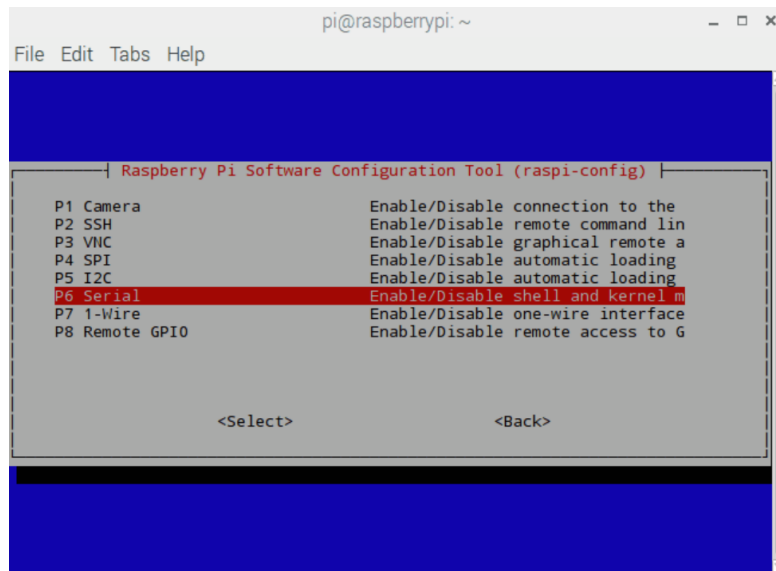


Figura E.2: Programação de Raspberry Pi para comunicação RS232 - parte 2

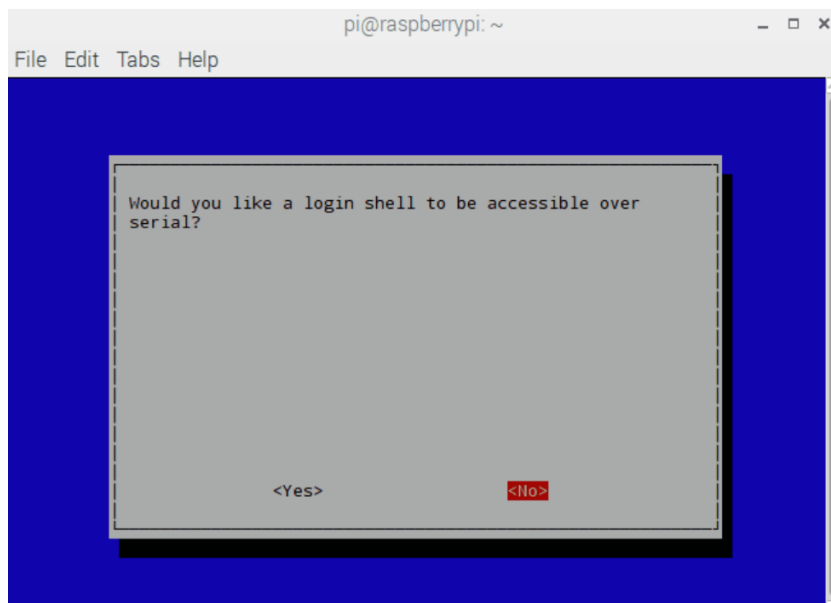


Figura E.3: Programação de Raspberry Pi para comunicação RS232 - parte 3

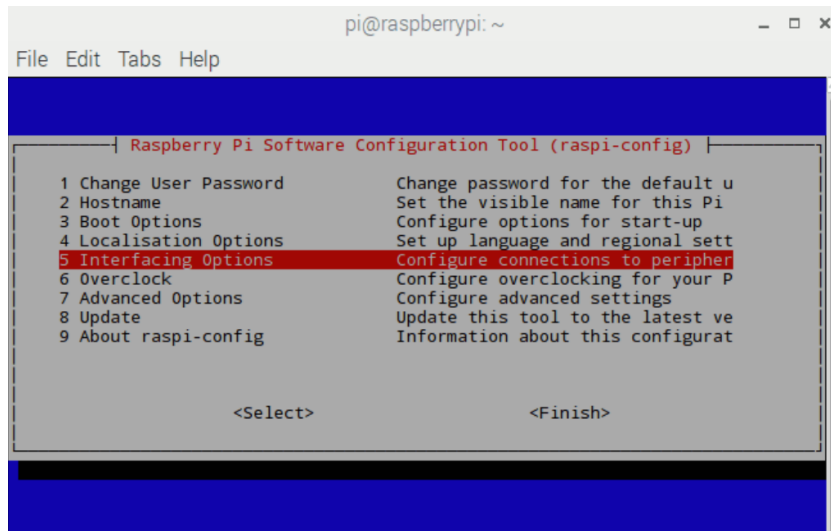


Figura E.4: Programação de Raspberry Pi para comunicação RS232 - parte 4

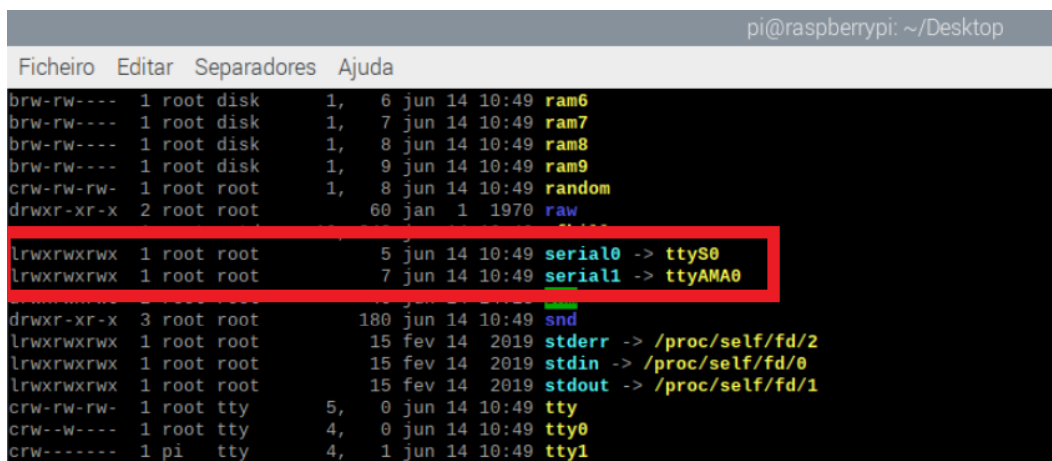


Figura E.5: Programação de Raspberry Pi para comunicação RS232 - parte 5

Página em branco intencional.

Apêndice F

Instalação Wireshark no Raspberry Pi

Para iniciar a instalação deverá de garantir em primeiro lugar que o sistema se encontra atualizado. Após realizar esta verificação deverá de instalar o pacote respetivo ao software com o comando «**sudo apt-get install wireshark**».

Na janela que surgirá no final da instalação deverá de selecionar «**No**», esta escolha é devido a existir a possibilidade de atribuir permissões indesejadas ao software. De seguida deverá de criar um novo grupo denominado wireshark, com o comando «**sudo groupadd wireshark**», seguido do comando «**sudo usermod -a -G wireshark pi**», sendo que **pi** é o respetivo nome de utilizador, que deverá de alterar caso este seja diferente. O próximo comando é «**sudo chgrp wireshark /usr/bin/dumpcap**» que altera o proprietário do grupo já criado. Após isto, é necessário atribuir permissões de escrita e leitura a este mesmo grupo com o comando «**sudo chmod 750 /usr/bin/dumpcap**». Por fim falta apenas configurar as definições deste mesmo ficheiro com o comando «**sudo setcap cap_net_raw,cap_net_admin=eip /usr/bin/dumpcap**». Para verificar se a instalação foi bem executada poderá introduzir o comando «**sudo getcap /usr/bin/dumpcap**» e deverá de retornar “**/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip**”.

Página em branco intencional.

Apêndice G

MongoDB

G.1 Em Linux

Para proceder à instalação do software MongoDB, aconselha-se a visita à página deste software onde se poderá encontrar todas as informações necessárias, contudo por detecção de falta de informação na secção seguinte será explicado a como proceder à configuração deste software, [50]. Para colocar o software a funcionar será necessário executar o processo `mongod`, bem como definir para executar o ficheiro «`mongod.conf`», como já explicado.

G.1.1 Configuração de `mongod.conf`

Para proceder à configuração deste ficheiro é de notar que este encontra-se escrito em YAMAL Ain't Markup Language (YAML), pelo que não suporta indentações. Nas alterações será necessário especificar uma pasta para guardar informação, neste caso `/data/db` bem como definir os parâmetros de ligação, neste caso a porta e o IP de lançamento que tem de ser adequado a cada uso, nesta implementação desenvolvida, `27017` e `mongodb.blockchain.com` respetivamente. Posteriormente é necessário alterar as permissões do diretório `/dat/db` para que o programa possua permissões de leitura e escrita. É ainda importante definir o diretório para o ficheiro `mongodb.log`. Por fim, como a implementação tem contemplado a utilização de certificados será importante definir a localização dos certificados a utilizar, bem como colocar como necessário o uso de certificados para estabelecer uma ligação válida, impedindo que aconteça ligações não seguras.

Por fim, será necessário adicionar no ficheiro `hosts` do Linux os IP com que cada certificado irá funcionar para que o sistema os possa reconhecer como válidos.

G.1.2 Leitura de dados - Mongo Shell

Após executar o comando de lançamento do `mongod` é possível aceder à base de dados de forma a verificar a informação presente nesta, bem como alterar, ou adicionar manualmente. Para tal, é necessário recorrer a uma vez mais a um certificado TLS, e executar o comando visível no exemplo de código G.1 para estabelecer ligação através da Mongo Shell. Para poder ler uma dada informação é necessário saber em que Base de dados se encontra e seleciona-la respetivamente, exemplificado no exemplo de código G.2, e de seguida aceder à respetiva «coleção», exemplo de código G.3. Para realizar outro tipo de

```

1 127.0.0.1 localhost
2 ::1 localhost
3 127.0.1.1 pop-os.localdomain pop-os
4 192.168.0.14 popos.blockchain.com
5 192.168.10.2 raspberry.com
6 192.168.0.14 popos.blockchain.com
7 192.168.10.10 popos.blockchain.com
8 192.168.10.2 gateway1.blockchain.com
9 192.168.10.10 gateway.blockchain.com
10 192.168.10.10 gateway11.blockchain.com
11 192.168.10.10 node1.blockchain.com
12 192.168.10.10 node2.blockchain.com
13 192.168.10.10 node3.blockchain.com
14 192.168.10.10 mongodb.blockchain.com
15 192.168.10.10 admin.blockchain.com
16 192.168.10.10 client1.blockchain.com
17 192.168.10.10 client2.blockchain.com
18 192.168.10.11 client3.blockchain.com
19 192.168.10.10 client2.blockchain.com

```

Figura G.1: Ficheiro Hosts

operações o processo é semelhante e os comandos poderão ser encontrados no site do fornecedor do software.

```

1 mongo --tls --tlsCertificateKeyFile ../certificados/cliente.pem --
  tlsCAFILE ../certificados/ca.pem --hosts mongodb.blockchain.com

```

Listing G.1: Acesso à base de dados com recurso a TLS

```

1 use BlockchainDB

```

Listing G.2: Base de dados - MongoDB

```

1 use BlockchainCLT

```

Listing G.3: dbcollection - MongoDB

G.2 Windows

Primeiramente é necessário instalar os softwares necessários, o que inclui o MongoDB. Como tal, será necessário instalar dois componentes. O primeiro «MongoDB Compas». Este software permite aceder, adicionar ou editar os dados de forma mais simples. Após descarregar o instalador do site do fornecedor deverá de proceder aos passos indicados por este, contudo deverá de manter a check box visível na figura G.2 verificada, pois pretendemos gerar o servidor no nosso computador.

Será necessário instalar também o «Mongo Shell», denominado usualmente como «Mongosh». De notar, que deverá de tomar nota do diretório de instalação. Após proceder à instalação direta através do ficheiro disponibilizado pelo fornecedor, deverá de adicionar esta variável às variáveis do sistema. Para tal, na barra de procuras do windows deverá de digitar «Variáveis de ambiente» abrir a opção referente a propriedades do

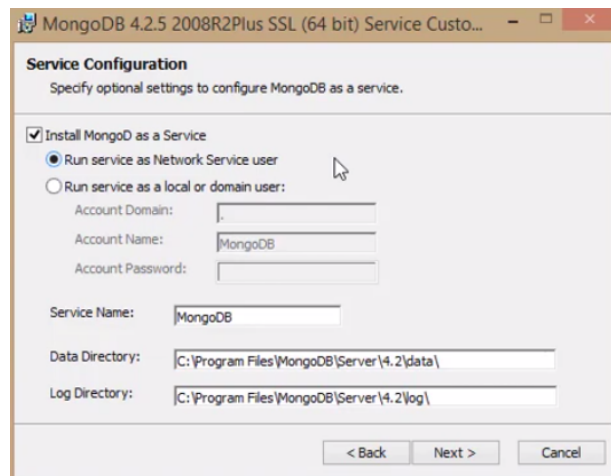


Figura G.2: Instalação MongoDB diretório

sistema, e carregar no botão presente no canto inferior direito como mostra a imagem seguinte, figura G.3. Na próxima janela, na segunda caixa de texto, deverá de carregar na linha com a indicação «path» e de seguida em editar, figura G.4. Para concluir a instalação, deverá carregar em novo e adicionar uma nova linha com o diretório da pasta bin presente na pasta de instalação, figura G.5. Para iniciar corretamente o programa, deveremos primeiro de iniciar como administrador as aplicações «mongo.exe» e «mongosh.exe» presentes na pasta bin, figura G.6. Após abrir as duas aplicações deverá de conseguir visualizar as seguintes janelas, figura G.7. Desta forma, será então possível inicializar o MongoCompass bem como criar servidores e geri-los sem problemas. Na primeira inicialização terá de criar uma nova conexão e uma nova base de dados, através do menu presente à esquerda. Nas seguintes já terá acesso à sua ligação local, facilitando o acesso, figura G.8. No fim de estabelecer a conexão terá acesso a todas as bases de dados criadas como na imagem G.9.

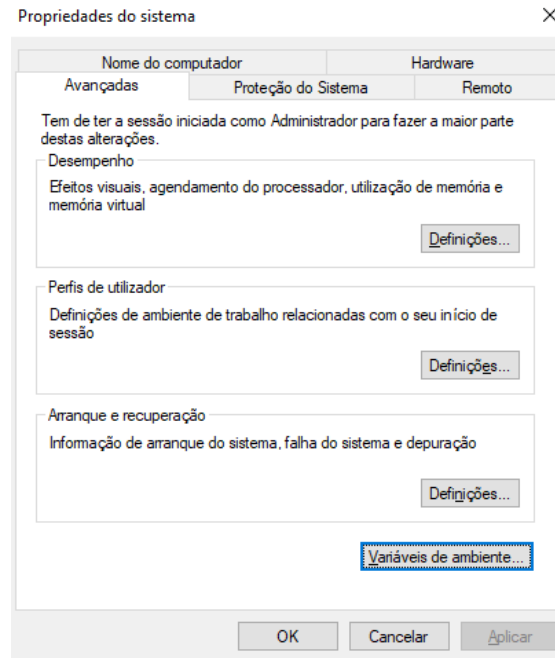


Figura G.3: Instalação MongoDB - variáveis

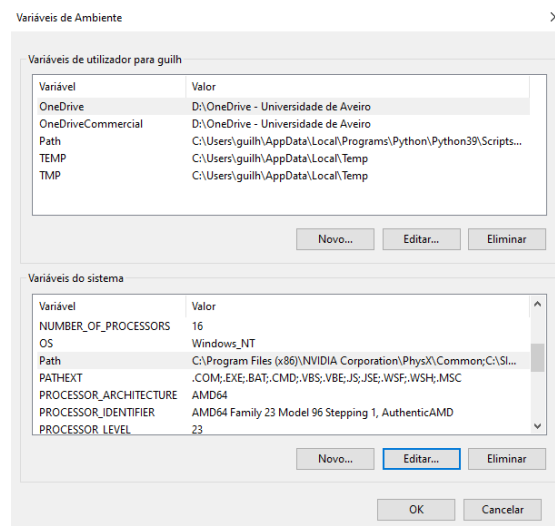


Figura G.4: Instalação MongoDB - Variáveis - path

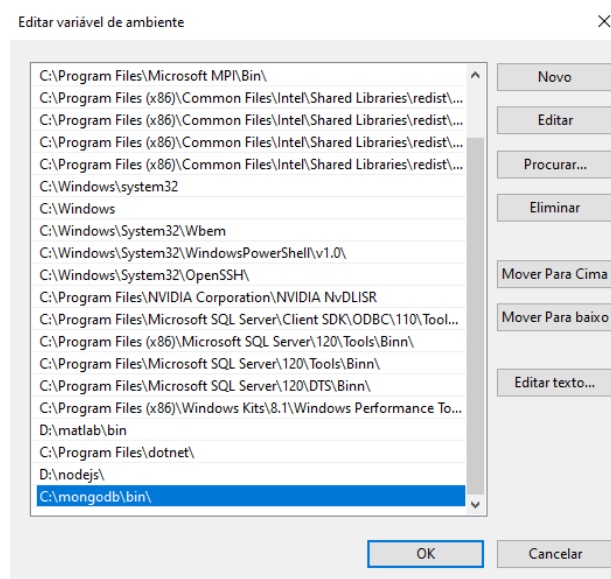


Figura G.5: Instalação MongoDB - nova variável

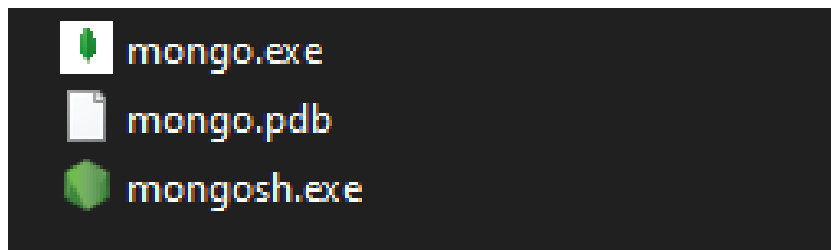


Figura G.6: Instalação MongoDB - inicialização

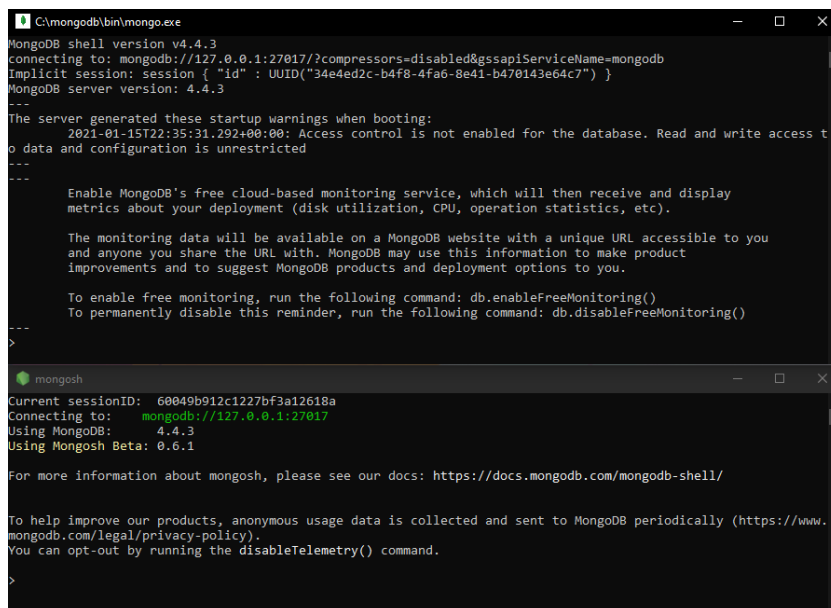


Figura G.7: Instalação MongoDB - estabelecer ligação ao server

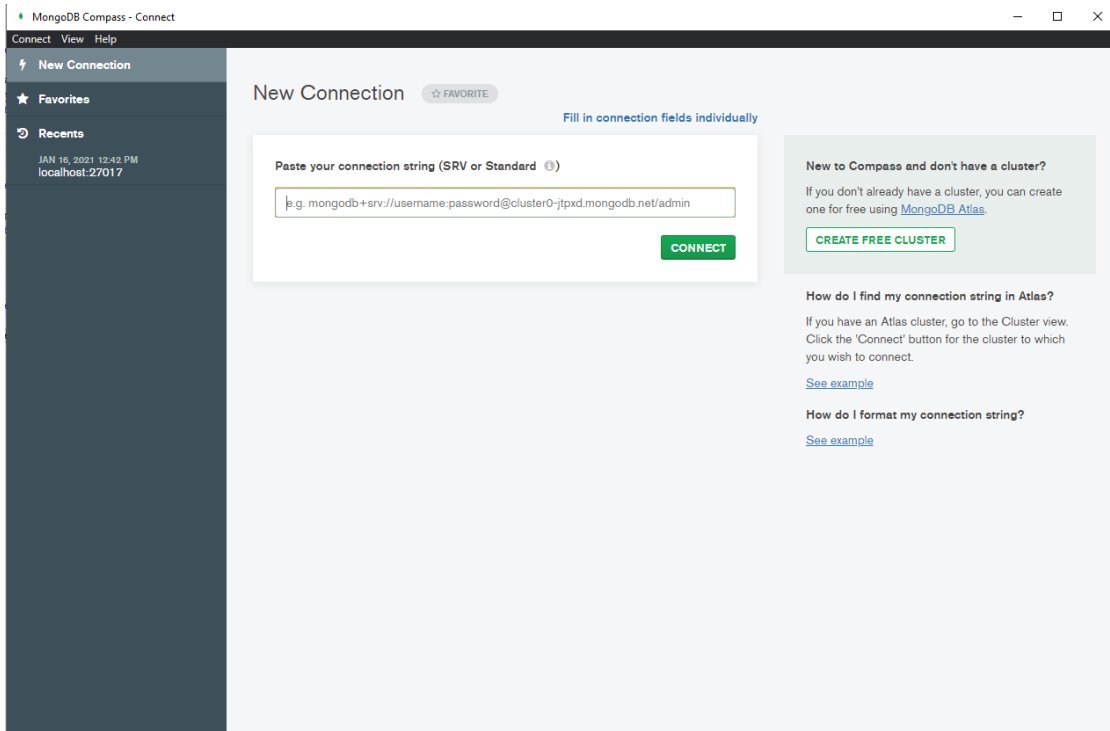


Figura G.8: Instalação MongoDB - Compass

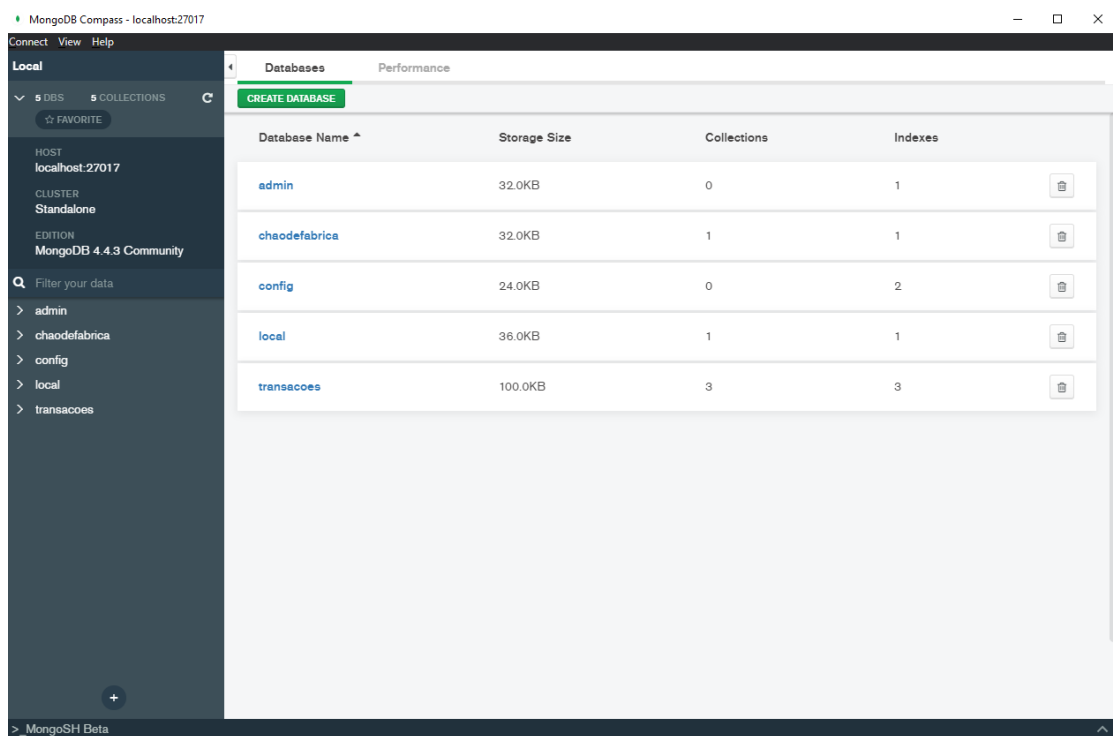


Figura G.9: MongoDB bases de dados

Apêndice H

Base de dados parcial da gateway

Nos exemplos de código H.1 e H.2 são exemplificadas parte do código desenvolvido para que se possa ler e escrever na base de dados de cada gateway respetivamente. No código de leitura é visível o uso da mesma chave para proceder à leitura das primeiras 10000 entradas, bem como o Modo CBC na descriptação. É de notar que o ficheiro se encontra a ler as chaves armazenadas. Já no ficheiro de escrita pela função `check_LDR_vals()` é visível como se procede à concatenação da informação e na função `save_to_db()` é visível o processo de gravação da informação.

```
1 class Getkeys():
2     '''
3     Class used to define and decrypt info in txt file.
4     '''
5     def __init__(self):
6         self.sensors = dict()
7
8
9     def decrypt(self, enc):
10        '''
11        decrypt Decrypt data using AES CBC.
12        Returns:
13        [bytes] -- Decrypted data using a given key.
14        '''
15        cc=0
16        while cc<10000:
17            line=enc[cc]
18            key = "c86a301a-d227-4eal-96a0-40f96de728cf".encode('ISO
-8859-1') #key used to encrypt information
19            key=hashlib.sha256(key).digest()
20            iv= line[:BLOCK_SIZE]
21            cipher = AES.new(key, AES.MODE_CBC, iv)
22            chave=cipher.decrypt(line)[BLOCK_SIZE:]
23            chaves.append(chave)
24
25            cc+=1
26            print(chaves)
27            return chaves
28
29
30
31 def read_info():
32     """
```

```

33     This part of the scrip read the stored keys to encrypt or decrypt the
34     information
35     """
36     with open("chaves_de_escrita", "rb") as file:
37         linhas=file.read()
38     exit
39
40     codificacao=[]
41     posicaooinicial=0
42     while linhas[posicaooinicial:].find(b'\n\nCRLF') > 0:
43         posicaooden=linhas[posicaooinicial:].find(b'\n\nCRLF')
44         codificacao.append(linhas[posicaooinicial:posicaooden+
45         posicaooinicial])
46         posicaooinicial=posicaooden+5+posicaooinicial
47         #+5 because its \n\nCRLF and \n acts like 1 character
48     return codificacao

```

Listing H.1: Exemplo ilustrativo de processo da descriptação da base de dados da gateway

```

1 def check_LDR_vals():
2     # Data structure
3
4     data = {'sensing_unit': 'S002', # Sensing Unit No.
5            'LDR': str(random.randint(0, 999)), # LDR read
6            'LED': str(random.randint(0, 1))} # Value of debug LED
7     global data_to_save
8     data_to_save = data["sensing_unit"]+ ',' + data["LDR"] + ',' + data["
9     LED"] # MAX 16 bytes, for more change the mode of AES
10    data_to_save = data_to_save.encode('ISO-8859-1')
11    save_to_db(data_to_save)
12
13    return data_to_save
14
15 def save_to_db(data_to_save):
16     global n_entrada #number of the
17     message to save #2 operations to
18     pad = BLOCK_SIZE - len(data_to_save) % BLOCK_SIZE #2 operations to
19     pad data do 16 bytes
20     padded_data = data_to_save + ((" ")*pad).encode('ISO-8859-1')
21
22     n_entrada+=1 #increment the
23     number of the message #work with 1000
24     n_pass=math.floor(n_entrada/1000)
25     message
26
27     key=vals_uc[n_pass]
28
29     linefeed="\n\nCRLF"
30     iv = os.urandom(BLOCK_SIZE_IV) #Random iv of 16
31     bytes
32     key=hashlib.sha256(key).digest()
33     aes =AES.new(key, AES.MODE_CBC, iv)
34     encd=iv + aes.encrypt(padded_data) + linefeed.encode('ISO-8859-1')
35
36     ficheiro="db_encrypted"

```

```
32     with open(ficheiro, 'ab') as f:  
33         f.write(encd)  
34     exit  
35  
36     return encd
```

Listing H.2: Exemplo ilustrativo de processo da encriptação da base de dados da gateway

Página em branco intencional.

Apêndice I

Tecnologia implementada e alternativas

I.1 Manual de utilizador ZD5800

O manual de utilizador do equipamento ZD5800 é visível na figura I.1. No primeiro terço da imagem é possível configurar o método de trabalho do equipamento, no caso da implementação desenvolvida será o último código, entenda-se RS232. A parte central é responsável por fornecer configurações de transmissão de informação, bem como se funcionará em modo automático de deteção. Já no último terço é visível um código que pode ser executado para testar o sensor bem como alguma informação sobre o funcionamento do equipamento.

I.2 Peças para leitura

[h] Na figura I.2, é visível um exemplar da peça que é trabalhada na linha em estudo e sobre a qual se processa toda a problemática desta implementação. É ainda visível o código Data Matrix na parte inferior da peça.

Antes de optar por esta implementação a Renault Cacia recorria a códigos Gália visíveis na figura 2.5.

I.3 Tecnologias alternativas

Para o tipo de trabalho que se pretende desenvolver, realizou-se a uma pesquisa das tecnologias existente no mercado, possibilitando avaliar se a tecnologia escolhida seria a indicada para avançar no projeto. Como tal, as principais alternativas ao código Gália, para proceder à identificação da peça, apresentam-se as alternativas a baixo.

I.3.1 Bar Codes

I.3.1.1 Códigos 1D

1D: barras verticais que conseguem conter até 48 caracteres. Estes são lidos com recurso a um laser ou a um leitor de imagens (só lê 1D). Estes códigos são bastante simples contudo estão a cair em desuso dado a pouca quantidade que conseguem armazenar.

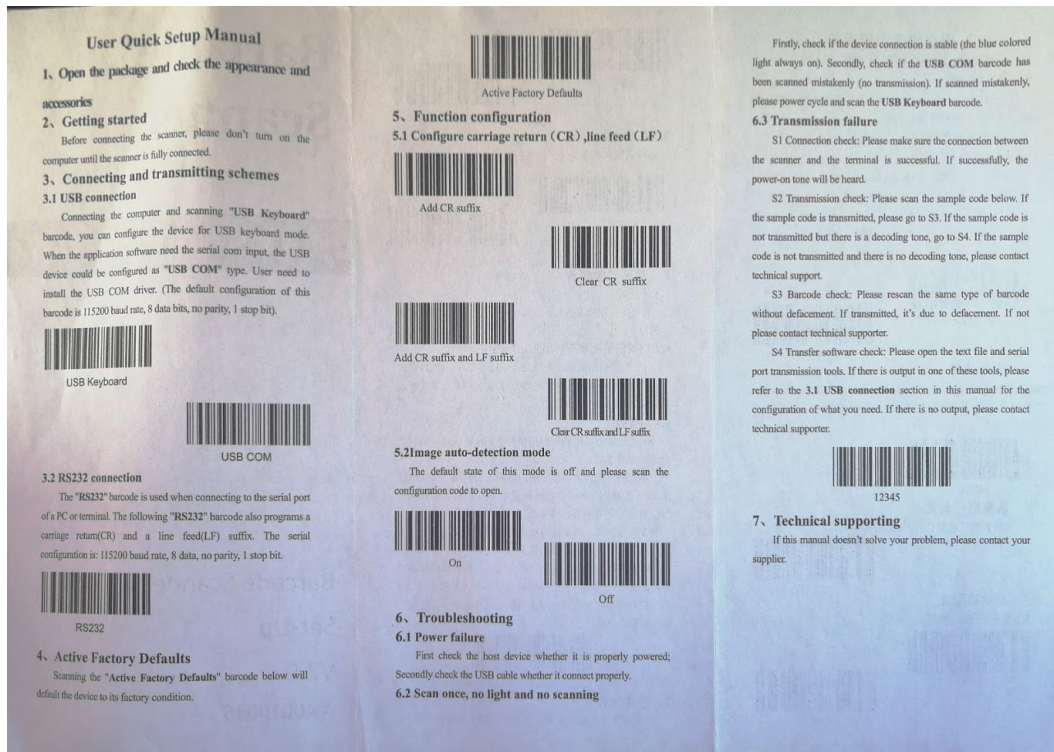


Figura I.1: Manual de utilizador - Sensor ZD5800

I.3.1.2 Data Matrix

A empresa em estudo está a proceder à implementação de gravação prévia de códigos Data Matrix na peça. Como tal, será esta tecnologia que iremos aprofundar. Como já foi referido, Data Matrix é um código de duas dimensões que consegue abranger muito mais informação que os códigos de uma dimensão. A tipologia Data Matrix baseia o seu alinhamento numa estrutura em L, auxiliada por duas linhas a tracejado, denominadas de «Clock Pattern». Esta organização permite que o código seja lido em qualquer direção, sendo assim, preferido em detrimento do QR code no que toca a implementações fabris.

Os códigos Data Matrix são compostos por sete módulos, sendo que cada módulo assume um valor binário de 1 ou 0. Cada grupo de sete módulos representa um número binário de sete dígitos, sendo que ao número resultante é subtraído uma unidade, resultando assim o respetivo caractere ASCII. Este método permite armazenar até 2335 caracteres alfa numéricos, considerando um tamanho de 144x144 módulos. Normalmente, recorre-se a papel, contudo, impressão a laser ou o Dot Peen Marking são métodos mais seguros para garantir a perpetuação da informação. Neste ponto, caso algo aconteça ao código é mais fácil recuperar a informação quando se trata de um código 2D do que 1D [51].

Outro benefício dos códigos 2D, em detrimento dos códigos 1D, é que como este possui maior capacidade, é possível utilizar a informação para corrigir erros de leitura, pelo método, «Reed-Solomon code».



Figura I.2: Exemplo de peça a ler na linha de produção

I.3.1.3 QRCode

Apesar do QRCode ser similar ao Data Matrix, chegando até muitas vezes a serem confundidos pelas pessoas, apresentam-se com diferenças bastante significativas. Os QR Code's apresentam a possibilidade de serem lidos em mais do que uma posição visto possuírem marcadores quadrados em três dos seus extremos. A organização da informação é bastante diferente neste método. Cada caractere é definido por oito módulos em vez de sete. O que permite armazenar mais caracteres. Como nos códigos Data Matrix, grande parte da informação armazenada é utilizada para correção de erros [51].

I.3.1.4 RFID

- RFID: sistema composto por uma tag e um leitor. A tag tem incluída uma antena e um micro-chip que emite um sinal podendo ser lido pelo leitor. Existem 3 tipos de tag's [52]:
 - Passivas: não tem fonte de energia própria (bateria), em vez disso, ficam à espera da onda vinda do leitor, que depois será convertida e enviada com a informação contida na tag;
 - Ativas: para além da antena e do micro-chip, tem uma bateria ou um painel solar como fonte de energia. Podendo ainda ser divididas em dois grupos:
 - * «Transponder», estes permitem que haja uma maior distância entre a tag e o leitor, contudo só enviam informação no fim de receber sinal do leitor.
 - * «Transmitters», estão constantemente a emitir informação podendo ou não ser lida;
 - Semi passivas: também contêm uma fonte de energia, contudo, esta não é forte o suficiente para emitir sinal através da antena, desta forma, só emite após receber uma onda vinda do leitor. Não atuam assim como «transmitters», tendo um alcance menor que estas, contudo, maior que as tag's passivas.
 - Fatores a favor:

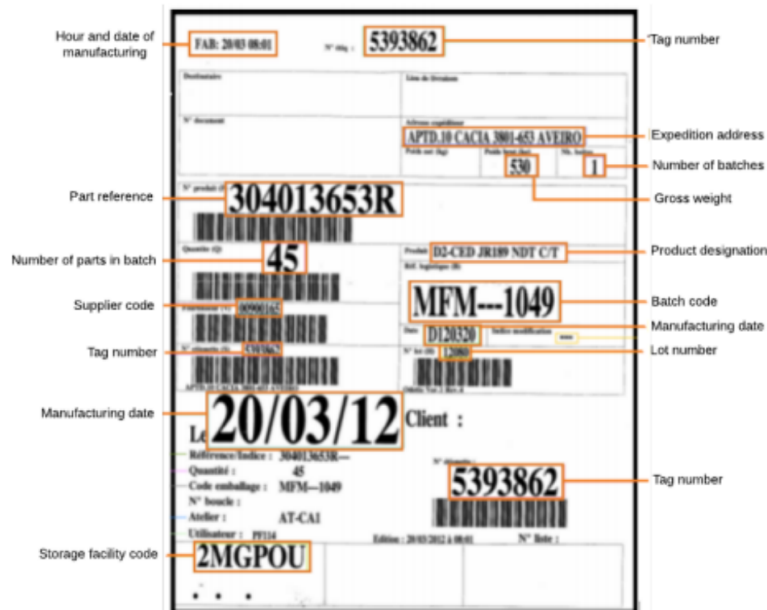


Figura I.3: Código Gália, [2]



Figura I.4: Barcode 1d

- * Informação dinâmica, pode ser alterada em qualquer estação;
- * Possibilita utilizar a informação da tag para automatização;
- * Flexibilidade de leitura;
- * Elevada memória;
- * Elevada resistência a condições extremas;
- * Possibilidade de ler várias tag's ao mesmo tempo.

– Fatores contra:

- * Custo de aquisição maior;
- * Dificuldade de seleção caso seja destinado a ambientes metálicos;

I.3.2 Recolha de dados

Para proceder à recolha de dados existe duas principais alternativas. A primeira passa por módulos IoT e a segunda, por PLC. Neste aspecto, os módulos IoT têm como principal vantagem serem de baixo custo, terem capacidade de processamento adequada e de dimensões reduzidas bem como, terem comunicações Wireless embutidas. Por outro lado, apresentam-se como sendo mais frágeis no que toca a uma implementação em condições adversas, bem como menor número de saídas.

A segunda alternativa, apresenta-se de maior tamanho e de muito maior custo, valores que podem ascender a cinquenta vezes o valor dos módulos IoT referidos, não tendo ainda



Figura I.5: Desenvolvimento de Data Matrix

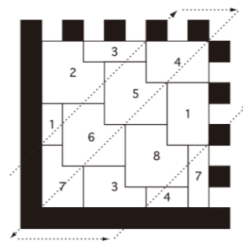


Figura I.6: Organização Data Matrix

incluído comunicações do tipo wireless. Por outro lado apresentam maior robustez física. Contudo é possível concluir que as tecnologias utilizadas são as mais indicadas.

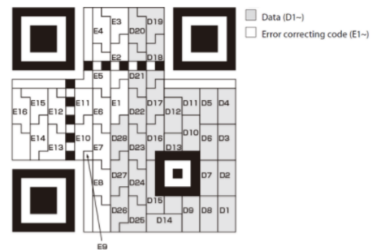


Figura I.7: Estrutura QR code



Figura I.8: ESP32



Figura I.9: PLC