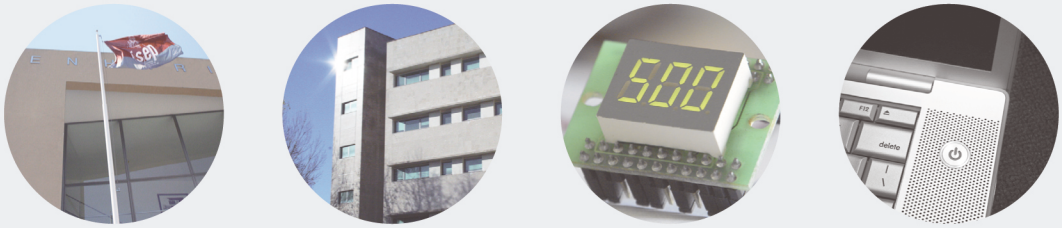
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de domotica sem fios utilizando rede de longa distancia e baixa potencia

RUI PEDRO GONÇALVES DA CRUZ MOREIRA DA SILVA
novembro de 2018

POLITÉCNICO
DO PORTO

SISTEMA DE DOMÓTICA SEM FIOS UTILIZANDO REDE DE LONGA DISTÂNCIA E BAIXA POTÊNCIA (LOW-POWER WIDE-AREA NETWORK)

Rui Pedro Gonçalves da Cruz Moreira Silva

Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

2018

Relatório da Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores Ramo Automação e Sistemas

Candidato: Rui Pedro Gonçalves da Cruz Moreira Silva - 1120728@isep.ipp.pt
Orientação científica: Eng. António Quadros Flores - aqf@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
14 de novembro de 2018

Agradecimentos

Agradeço ao meu orientador o Sr. Engenheiro António Quadros Flores os esclarecimentos prestados assim como todo o apoio dispensado.

Agradeço ainda e uma vez mais, aos meus pais, que sempre me incentivaram e apoiaram ao longo do meu percurso académico. A conclusão deste projeto só foi possível com o seu apoio incondicional.

Por fim, gostaria de agradecer ao Instituto Superior de Engenharia do Porto e a todo o corpo docente que me acompanhou ao longo do meu percurso académico transmitindo-me todo o conhecimento adquirido assim como a todos os colegas que de alguma forma comigo se cruzaram.

Resumo

A popularidade da domótica tem crescido ao longo dos anos devido aos seus numerosos benefícios, e com a inclusão de automações e inteligência artificial esses benefícios são ainda mais evidentes. A automação e a inteligência artificial integradas num sistema de domótica permitem não só o aumento da eficiência e da produtividade mas também um menor gasto de eletricidade e água, e reduzindo o envolvimento humano consequente nas tarefas são também reduzidos os erros derivados. A domótica residencial tem como principal função o aumentar o conforto, gerir a energia e aumentar a segurança dos habitantes.

Com o aparecimento de vários sistemas de domótica no mercado, cada um com o seu protocolo diferente de comunicação, levou a problemas de compatibilidade entre eles. Isto levou à necessidade de desenvolver sistemas de comunicação *standard* para utilizar na domótica como o sistema European Home System, CEBus ou European Installation Bus. No entanto com a evolução das tecnologias de comunicação sem fios estas começaram a apresentar mais vantagens em relação aos sistemas com fios principalmente no que diz respeito à instalação. Mesmo assim, os sistemas de domótica sem fios existentes no mercado são baseados em tecnologias de comunicação que possuem um alcance limitado levando a um aumento de latência ou ao uso de repetidores de sinal elevando o custo da instalação.

Este projeto pretende ser uma solução para esses problemas. O sistema desenvolvido utiliza a tecnologia de longo alcance e baixa potência LoRa para a comunicação entre dispositivos e integra a comunicação Wi-Fi que permite o acesso fácil dos utilizadores ao sistema. A utilização de um *software* de domótica como o Home Assistant permite a inclusão de outros sistemas de domótica com diferentes protocolos de comunicação na mesma habitação. Este projeto pretende ser uma alternativa aos sistemas de domótica sem fios existentes no mercado apresentando baixo consumo de energia, grande área de atuação, possibilidade de integração com outros sistemas de domótica e também com baixo custo monetário.

Palavras-Chave

Domótica, LoRa, LPWAN, Home Assistant

Abstract

The popularity of home automation has grown over the years because of its numerous benefits, and with the inclusion of automation and artificial intelligence these benefits are even more evident. Automation and artificial intelligence integrated in a home automation system allow not only increased efficiency and productivity but also lower electricity and water costs, and reducing the consequent human involvement in tasks are also reduced derived errors. Home automation has as its main function to increase comfort, manage energy and increase the safety of the inhabitants.

With the appearance of several home automation systems on the market, each with its different communication protocol, led to compatibility problems between them. This led to the need to develop standard communication systems for use in home automation systems such as the European Home System, CEBus or European Installation Bus. However with the evolution of wireless technologies these have started to present more advantages compared to wired systems mainly in regards to installation. Even so, the wireless home automation systems on the market are based on communication technologies that have limited reach leading to increased latency or the use of signal repeaters by raising the cost of installation.

This project is intended to be a solution to these problems. The system uses LoRa's low-power technology for communication between devices and integrates Wi-Fi communication that allows users easy access to the system. The use of home automation software such as Home Assistant allows the inclusion of other home automation systems with different communication protocols in the same housing. This project intends to be an alternative to the wireless domotic systems in the market presenting low power consumption, large area of operation, possibility of integration with other home automation systems and also with low monetary costs.

Keywords

Domotic, LoRa, LPWAN, Home Assistant

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AES	–	Advanced Encryption Standard
BLE	–	Bluetooth Low Energy
CA		Corrente Alternada
CC		Corrente Continua
DES	–	Data Encryption Standard
FSK	–	Frequency-Shift Keying
GFSK	–	Gaussian Frequency-Shift Keying
GPIO		General Purpose Input/Output
GPU		Graphics Processing Unit
I/O		Input/Output
I²C		Inter-Integrated Circuit
IOT	–	Internet of Things
IP		Internet Protocol
ISM	–	Industrial, Scientific and Medical
LPWAN	–	Low-Power Wide-Area Network
RSSI		Received Signal Strength Indicator
SMD		Surface Mount Device
SNR	–	Sinal to Noise Ratio
SoC		System On Chip

SPI – Serial Peripheral Interface

UART Universal Asynchronous Receiver-Transmitter

UHF – Ultra High Frequency

VHF – Very High Frequency

1 Introdução

1.1 Enquadramento e Motivação

O avanço da tecnologia vem provocar mudanças nos edifícios e trazer às sociedades mais conforto, segurança e consequentemente mais qualidade de vida.

A domótica permitiu simplificar tarefas no quotidiano das pessoas seja nas suas habitações seja nos seus empregos permitindo que vários sistemas como a comunicação, a segurança, iluminação e climatização estivessem todos integrados no mesmo sistema automático. A integração da eletrónica residencial com a informática permitiu que a gestão de um sistema de domótica seja feito conforme as necessidades dos seus utilizadores e nos sistemas mais evoluídos o sistema pode recorrer também à inteligência artificial para tomar ações totalmente automáticas sem qualquer interferência do utilizador. Um simples exemplo do uso da inteligência artificial é a ativação de uma lâmpada através de um processo de aprendizagem dos hábitos do utilizador [1].

Os sistemas de domótica também tem desvantagens, a sua instalação é por vezes difícil, principalmente se o sistema necessitar de ligação com fios, e requer técnicos especializados para a fazer. Além disso os sistemas são dispendiosos, gastam energia que em certos casos é significativa e levantam questões acerca da privacidade dos seus utilizadores. A maioria dos sistemas são proprietários (licenciado com direitos exclusivos para o produtor), pouco flexíveis e não permitem a integração de vários sistemas ou módulos de domótica a operarem em conjunto, sendo que o utilizador apenas pode usar os equipamentos da mesma marca ou ter um sistema de domótica limitado nas suas capacidades [2]. Deste modo, de forma a combater estes problemas surge a necessidade da elaboração deste projeto. Com a evolução das tecnologias de comunicação sem fios, hoje é possível transmitir dados a um longo alcance sem gastar muita energia, o que é perfeito para equipamento IOT (Internet of Things), domótica entre outras aplicações.

1.2 Cenários de Aplicação

Este projeto pretende ser uma melhor solução aos sistemas de domótica sem fios introduzindo para isso a tecnologia LPWAN (Low-Power Wide-Area Network) permitindo assim um maior alcance de comunicação dos sensores numa grande área e

com um baixo consumo comparativamente a outras tecnologias de comunicação sem fio. Além da inclusão desta tecnologia o sistema terá ligação à rede Wi-Fi e à internet de modo a que o utilizador tenha acesso e controlo do sistema a partir de qualquer dispositivo com acesso Wi-Fi em qualquer lugar. O sistema de domótica integrará uma unidade central de controlo, que estará ligada à rede Wi-Fi e à internet assim como aos dispositivos a controlar e sensores do sistema através da comunicação de longo alcance, formando assim uma LPWAN.

O resultado final deste projeto será destinado a qualquer pessoa que pretenda instalar um sistema de domótica numa habitação, empresa ou numa unidade industrial ou agrícola. Devido às caraterísticas definidas nos objetivos do projeto (subcapítulo 1.3), o sistema integra-se não só em habitações novas como também instalações já antigas devido à comunicação sem fios, sendo este tipo de sistema de alta portabilidade.

1.3 Objetivos

Com este projeto pretende-se desenvolver e apresentar uma melhor solução que as já existentes no mercado para sistemas de domótica sem fios, para isso foram estabelecidos os seguintes objetivos:

- Estudar as soluções existentes no mercado, no âmbito do projeto, e as suas tecnologias, nomeadamente as tecnologias de comunicação sem fios;
- Criar um sistema de fácil instalação e compacto;
- Criar módulos de domótica que permitiram a ligação a vários sensores e equipamentos atuadores;
- Integrar um *software* de automação que permita a inclusão de módulos de domótica de outros fabricantes, de preferência que possam interagir entre si;
- Permitir o acesso ao sistema, incluindo o seu controlo, por comunicação Wi-Fi;

1.4 Estrutura do relatório

A estrutura do presente relatório está relacionada com os objetivos anteriormente apresentados e encontra-se divida em 5 capítulos.

No capítulo, 1, “Introdução”, são referidos diversos aspetos relacionados com o enquadramento e motivação, os cenários de aplicação, objetivos, bem como a estrutura do relatório. A calendarização do trabalho é também apresentada neste capítulo.

No capítulo 2, “Estado de Arte”, apresentam-se os dispositivos existentes no mercado, tecnologias e abordagens semelhantes à deste projeto, assim como aspetos teóricos essenciais à compreensão do projeto.

No capítulo 3, “Projeto”, são enumeradas as necessidades do sistema bem como os requisitos funcionais e não funcionais. São também apresentados aqui a Arquitetura do Sistema, as opções do projeto, e a descrição do *Hardware* e do *Software*. Explicando de uma forma simples mas detalhada como foi projetada a solução. Os vetores de teste são apresentados neste capítulo de modo a testar os vários constituintes do projeto.

No capítulo 4, “Implementação”, revela-se passo a passo o que foi feito durante a execução do projeto, assim como o resultado final deste projeto, o protótipo.

Finalmente no capítulo 5, “Conclusões e trabalho futuro”, apresentam-se as conclusões mais relevantes em relação ao projeto realizado e analisa-se o resultado final, em conjunto com uma reflexão de possíveis ampliações/modificações a efetuar no projeto.

1.5 Calendarização

De modo a planificar o trabalho durante o tempo disponível foi elaborado um plano, a calendarização semanal é apresentada na Figura 1. As principais tarefas, relacionadas com a elaboração do relatório mas também na execução do protótipo, foram incluídas.

A calendarização permite verificar se o trabalho evolui conforme o previsto e caso o plano não seja cumprido, seja por atrasos ou avanços no trabalho, é necessário fazer um ajuste na calendarização ou diminuir o tempo disponível das tarefas seguintes.

	Tarefa	Duração
1	Execução do Projeto	174 dias
2	Elaboração do Relatório	174 dias
3	Formulação do problema	21 dias
4	Introdução	7 dias
5	Estado de arte	28 dias
6	Analise de requisitos	21 dias
7	Definição e projecção do sistema	35 dias
8	Implementação	42 dias
9	Conclusão, trabalho futuro e resumo	10 dias
10	Finalização do relatório e correcções	10 dias
11	Elaboração do Prototipo	119 dias
10	Testes individuais dos equipamentos	14 dias
11	Preparação do SO da Gateway	21 dias
12	Criação dos modulos de domotica	21 dias
13	Desenvolvimento do software	42 dias
14	Montagem e ligação dos componentes	7 dias
15	Testes finais do prototipo	14 dias
16	Apresentação	

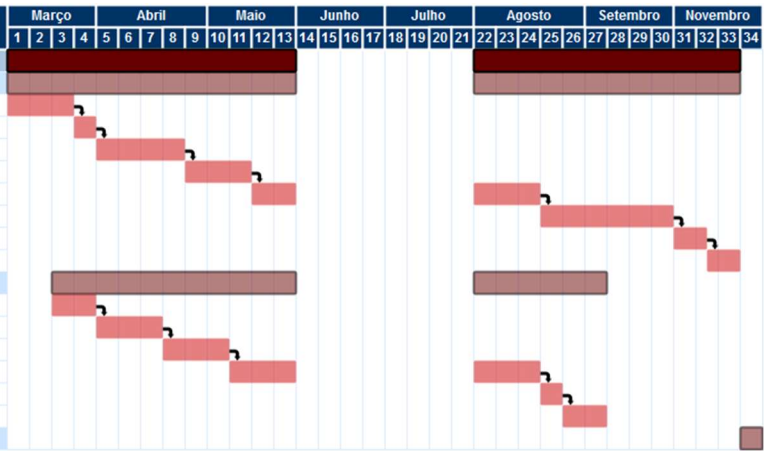


Figura 1 - Plano de trabalho

2 Estado de Arte

Neste capítulo será feito um estudo teórico sobre matérias importantes para a elaboração do projeto. Sendo assim, engloba a comunicação sem fios, a Low-Power Wide-Area Network (LPWAN), e uma pesquisa de produtos semelhantes a este projeto que se encontram já no mercado.

2.1 Comunicação sem fios

Um dos objetivos mais importantes desse projeto é a utilização da comunicação sem fios. Logo, foi feito um estudo das arquiteturas de comunicação sem fios disponíveis.

A comunicação sem fios implica a transferência de dados sem a utilização de cabos. O total de informação a transmitir, a velocidade, a distância e a energia necessária para a transferência depende do tipo de comunicação utilizado. A comunicação sem fios apresenta vantagens em relação à comunicação com fios nomeadamente um maior grau de mobilidade, baixa manutenção e fácil instalação. No entanto este estilo de comunicação também apresenta algumas preocupações em termos de segurança já que qualquer pessoa dentro do alcance da rede pode tentar aceder aos dados a serem transmitidos, sendo assim, é importante garantir que todos os dados são encriptados antes de serem enviados [3].

Apesar de existirem vários tipos de comunicação sem fios como infravermelho, ultrassom ou indução eletromagnética será dada especial atenção à comunicação rádio já que dentro do espetro eletromagnético é na gama de frequências radio que se encontram as principais tecnologias de comunicação sem fios.

A gama de frequências das ondas rádio encontra-se entre 30 Hz e 300 GHz segundo a IEEE divididos para diferentes bandas que correspondem a intervalos de frequências diferentes, como podemos ver na Tabela 1. Nesse espetro, na banda UHF (Ultra high frequency) entre 0.3 GHz e 1 GHz e na banda VHF (Very High Frequency) entre 30 MHz e 300 MHz é onde se inclui a comunicação LoRa, e na banda ISM tipo B (Industrial, Scientific and Medical entre 2.4 GHz e 2.483 GHz que está incluída na banda S) podemos encontrar a comunicação Wi-Fi, Bluetooth, ou BLE (Bluetooth Low Energy) [4].

Tabela 1 - Designações padrão da IEEE para bandas radar [5]

Banda IEEE	Comprimento da onda	Intervalo de frequências
MF	De 1 km a 100 m	300 KHz a 3 MHz
HF	De 100 m a 10 m	3 MHz a 30 MHz
VHF	De 10 m a 1 m	30 MHz a 300 MHz
UHF	De 1 m a 30 cm	300 MHz a 1 GHz
L	De 30 cm a 15 cm	1 GHz a 2 GHz
S	De 15cm a 5 cm	2 GHz a 4 GHz
C	De 5 cm a 3,75 cm	4 GHz a 8 GHz
X	De 3,75 cm a2,5 cm	8 GHz a 12 GHz
Ku	De 2,5 cm a 1,6 cm	12 GHz a 18 GHz
K	De 1,6 cm a 1,2 cm	18 GHz a 26 GHz
Ka	De 1,2 cm a 750 mm	26 GHz a 40 GHz
V	De 750 mm a 40 mm	40 GHz a 75 GHz
W	De 40 mm a 28 mm	75 GHz a 111 GHz
	De 1 km a 100 m	Acima de 111 GHz

A tecnologia Wi-Fi permite a transmissão por rádiofrequência baseada na norma IEEE 802.3 (Ethernet) e permite cobrir grandes áreas e facilmente equipar uma casa com internet. A tecnologia Wi-Fi sofreu várias alterações ao longo dos anos desde a primeira versão 802.11 padronizada pela IEEE em 1997 que permitia uma velocidade de transmissão de 1 Mb/s ou 2 Mb/s no intervalo de frequências entre 2,4 GHz e 2,4835 GHz. O mais recente padrão Wi-Fi a ser aprovado pela IEEE é o 802.11ac, este padrão, também chamado de 5G Wi-Fi pois opera na frequência de 5 GHz (banda C), permite velocidades até 433Mb/s podendo chegar, no modo mais avançado com várias antenas (no máximo 8), até 6 Gb/s. A tecnologia Wi-Fi pode ser instalada com diferentes mecanismos de autenticação que permitem a segurança de toda a rede, entre eles está o WEP, WPA, WPA2 e o WPS que permitem diferentes níveis de segurança na rede [6].

O BLE chamado também de "Bluetooth Smart" foi introduzido como parte do Bluetooth 4.0. A tecnologia BLE foi desenvolvida inicialmente pela Nokia Research Centre num

projeto chamado "Wibree" antes de ser adotado pela Bluetooth SIG (*Bluetooth Special Interest Group*) [7]. Esta tecnologia está presente em vários equipamentos como *smartphones* ou *tablets* sendo suportado pelos principais sistemas operativos e funciona na banda de frequências ISM tipo B. A tecnologia BLE permite o uso de topologias diferentes, como *Mesh*, *Point-to-point* ou *Broadcast* e requer muito pouca energia na transmissão, no entanto tem uma largura de banda e um alcance baixo de 50 m a 100m [8].

O Z-Wave é uma tecnologia de comunicação sem fios desenvolvida pela empresa Zensys especificamente para funcionar para automação residencial. A tecnologia Z-Wave permite fazer o controlo básico de domótica como iluminação e aparelhos ligados à tomada, assim como climatização, persianas elétricas e sensores de segurança. O Z-Wave é suportada por um conjunto de várias empresas que trabalham com esta tecnologia, a Z-Wave aliance. A comunicação Z-Wave tem a vantagem de utilizar pouca energia na transmissão, funciona na banda de frequências UHF (ver Tabela 1) e utiliza a modulação GFSK (Gaussian frequency-shift keying) uma variação da modulação FSK (Frequency-shift keying). O uso de frequências abaixo da banda ISM previne interferências com as redes comuns de uma habitação residencial como o Wi-Fi e o Bluetooth. A comunicação é ainda encriptada com a codificação AES (Advanced Encryption Standard) tornando assim o sistema seguro. Este tipo de codificação foi adotado pelo governo dos Estados Unidos da América e é agora usado em todo o mundo substituindo a encriptação DES (Data Encryption Standard). O Z-Wave tem uma largura de banda definida entre 9600 bit/s e 40 kbit/s e uma largura de banda entre 300 kHz e 400 kHz, esta tecnologia consegue um alcance de apenas 25 m em campo aberto e 10 m em ambiente fechado [9] [10]. No entanto o Z-Wave utiliza uma arquitetura de comunicação *mesh* que permite aumentar o alcance da rede utilizando vários caminhos para a transmissão da informação usando os equipamentos do sistema como retransmissores. Uma rede Z-Wave permite 232 dispositivos a funcionar simultaneamente [10] [11].

Devido a ser uma comunicação sem fios o sistema Z-Wave é fácil de instalar e não necessita de alterações elétricas numa habitação utilizando a instalação elétrica já existente. Por outro lado, esta tecnologia apresenta muito pouco alcance e uma grande latência devido à utilização da arquitetura de comunicação *mesh*. A velocidade de transmissão de dados é muito baixa não sendo possível a integração de câmaras ou microfones no sistema visto que a transmissão de imagem ou som não é permitida. Além

disso, a instalação de um sistema Z-Wave com muitos equipamentos torna o sistema demasiado caro comparando com outros sistemas de domótica [12].

ZigBee (IEEE 802.15.4) é uma comunicação sem fios desenvolvida para ser de baixo custo e de baixa potência. A aplicação principal do ZigBee é a automação residencial (domótica) mas também tem usos em ambientes industriais e agrícolas. O ZigBee utiliza a arquitetura *mesh* para a comunicação o que faz aumentar o alcance e a latência desta tecnologia mas que de outro modo o alcance seria apenas de 10 m a 100 m. A frequência de operação do ZigBee depende da região sendo que pode ser usada a frequência da banda ISM em todo o mundo. No entanto esta banda de frequência já é muito usada nos edifícios podendo gerar interferências na comunicação ZigBee. As taxas de transmissão dentro da rede variam muito dependendo da frequência utilizada, sendo que as frequências mais elevadas permitem velocidades de transmissão mais altas até 250 kbit/s e as mais baixas a partir de 20 kbit/s. Existem três tipos diferentes de dispositivos numa rede ZigBee, o coordenador (ZC), o router (ZR) e o dispositivo final (ZED). Existe apenas um ZC em cada rede, e é este dispositivo que cria a rede e a controla. O ZR, além de executar a função de um ZED também funciona como intermediário de comunicação entre dispositivos. E o ZED apenas comunica com o seu ZR ou ZC poupando assim energia sendo o mais simples dos dispositivos dentro da rede [13]. Na Figura 2 está representado a arquitetura *mesh* e a arquitetura em estrela.

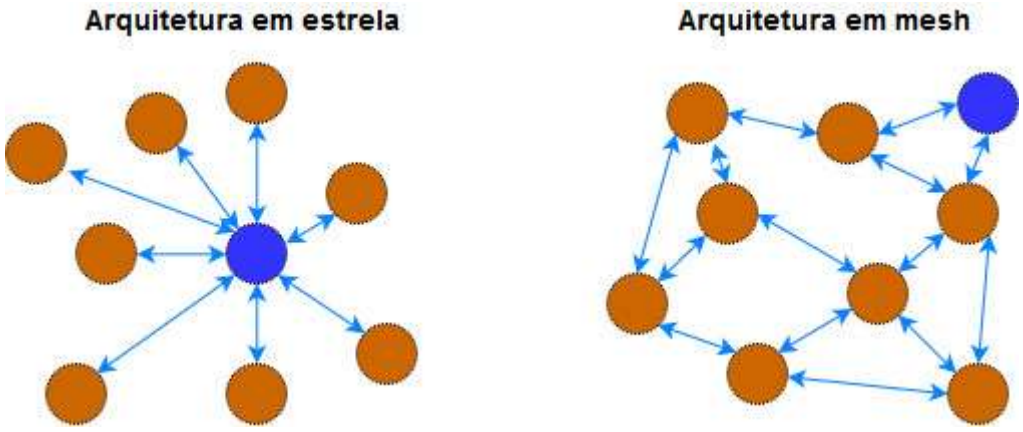


Figura 2 - Arquitetura em estrela e em mesh, *end nodes* representados a laranja e *gateways* representadas a azul

LoRa provem de *Long Range* é uma tecnologia de modulação patenteada pela *Semtech,* é uma tecnologia de transmissão sem fios de dados a longo alcance, de baixo consumo e tem um alto *link budget*. Esta tecnologia é uma derivação do CSS (chirp spread spectrum). Esta tecnologia consegue um longo alcance mas tem uma largura de banda extremamente baixa, normalmente 125 kHz mas pode ser configurada, comparado com outras

tecnologias de comunicação sem fio já mencionadas [14]. Estas caraterísticas fazem com que esta tecnologia não compita com as outras tecnologias sem fios já referidas como o Wi-Fi ou o Bluetooth, reservando um propósito diferente a estas. O Wi-Fi 802.11n a 2.4 GHz tem alta largura de banda e um alcance considerável de 250 m mas necessita de bastante energia na transmissão de dados e o mais recente Wi-Fi 802.11ac a 5 GHz consegue ainda mais largura de banda mas tem menos alcance e gasta mais energia na transmissão que a versão Wi-Fi a 2.4 GHz. Já o BLE consegue operar com pequenas baterias mas tem um alcance de poucos metros e baixa largura de banda de apenas 1 MHz. Por outro lado o Bluetooth Classic consegue um maior alcance e uma maior largura de banda mas necessita de mais energia. O ZigBee tem um alcance e largura de banda semelhante ao Bluetooth mas um maior gasto de energia que o BLE. O Z-Wave consegue um gasto de energia muito baixo limitando a largura de banda mas tem um alcance muito baixo de apenas 50 m. A relação entre a largura de banda e o alcance das várias tecnologias sem fio pode ser analisada na Figura 3.

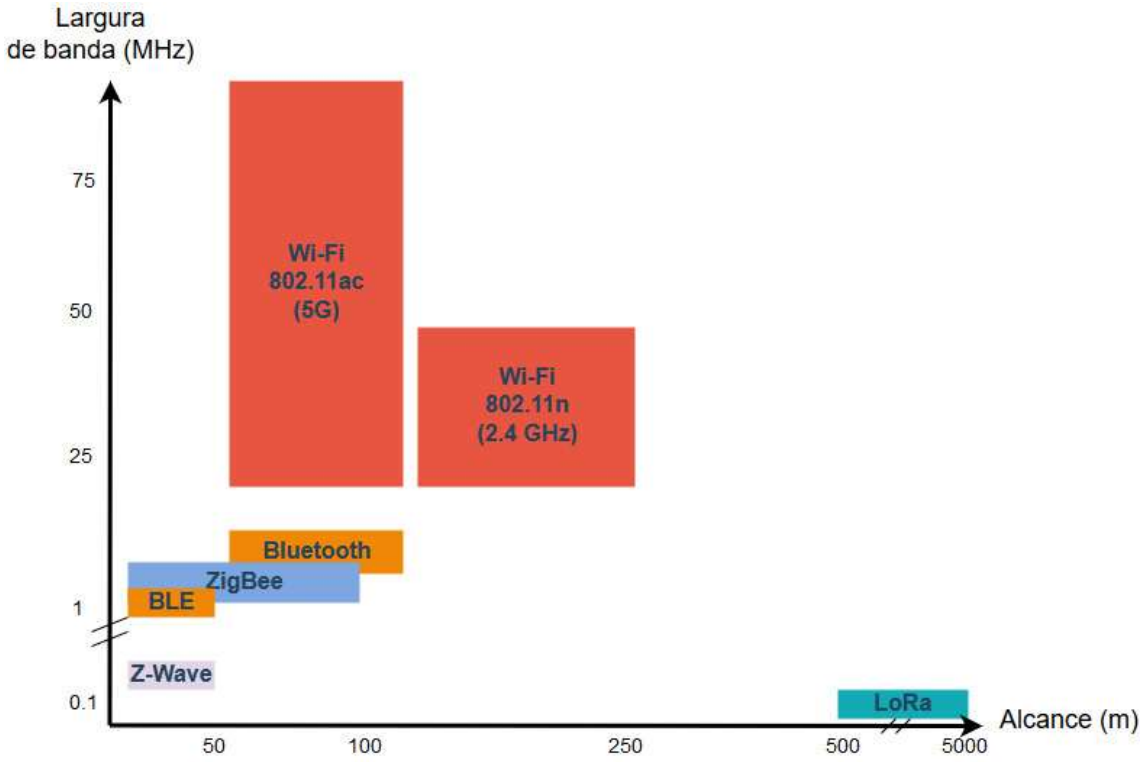


Figura 3 - Relação entre largura de banda e o alcance das várias comunicações sem fios

Outro aspeto importante a analisar nestas tecnologias de comunicação sem fio é o *Link Budget.* O *Link Budget* é a soma de todos os ganhos e perdas na comunicação do transmissor até ao recetor. Diferentes estilos de comunicação assim comos as situações em que a transmissão ocorre varia o *link budget*. Parâmetros como a potência de envio,

os pacotes perdidos, amplificadores, antenas e a sensibilidade do recetor são contabilizados no *link budget* [15]. A equação geral para o cálculo do *link budget* está descrita na Equação 1 [16].

$$P_{Recebida} = P_{Transmitida} + Ganhos - Perdas \quad (dB) \tag{1}$$

O *link budget* está diretamente ligado ao alcance possível de uma determinada tecnologia de comunicação sem fios e quanto maior o *link budget* maior será o alcance dos pacotes transmitidos. A tecnologia LoRa, de acordo com o seu fabricante, um *link budget* disponível de 154 dB o que é considerável se compararmos com outras tecnologias de comunicação sem fios [17]. Uma antena com um ganho pode fazer aumentar o alcance consideravelmente [18].

Uma das razões que levam a tecnologia LoRa a conseguir um elevado *link budget* é a elevada sensibilidade de receção do sinal, o chamado de SNR (Sinal to Noise Ratio). O SNR é o mínimo rácio de potência de sinal para ruído em que é possível dividir o sinal de informação do sinal da portadora (desmodulação) em dB. A técnica de modulação usada na tecnologia LoRa levou a grandes melhorias no SNR. O SNR depende de outra variável, que é definida pelo utilizador chamada de *Spreading Factor*, que é definido como o logaritmo de base 2 dos chips/symbol. Esta variável, o *Spreading Factor,* varia de 6 a 12 tem uma relação direta com a velocidade da comunicação e o alcance que são necessários para a aplicação. Quanto menor o *Spreading Factor* maior a taxa de transmissão de dados (Bit Rate) mas haverá um menor alcance. Pelo contrário um *Spreading Factor* maior levará a um maior alcance mas menos velocidade na transferência de dados. Na Figura 4 é mostrado os *Spreading Factors* mais usados e a sua relação com o SNR e a velocidade de transmissão de dados para uma largura de banda de 125 kHz.

Spreading Factor	Chips/symbol	SNR limit	Time-on-air (10 byte packet)	Bitrate
7	128	-7.5	56 ms	5469 bps
8	256	-10	103 ms	3125 bps
9	512	-12.5	205 ms	1758 bps
10	1024	-15	371 ms	977 bps
11	2048	-17.5	741 ms	537 bps
12	4096	-20	1483 ms	293 bps

Figura 4 - Relação entre o Spreading Factor e o Chips/symbol, SNR tempo de envio de um pacote e a taxa de transferência para uma largura de bande de 125 kHz [19]

Caso seja necessário ainda é possível configurar o *Spreading Factor* 7 para uma largura de banda de 250 kHz que aumenta a taxa de transmissão (Bit Rate) para 11000 bit/s. Ainda é possível usar larguras de banda mais baixas até, 7.8 kHz, apesar da Semtech não recomendar larguras de banda inferiores a 62,5 kHz sem um Temperature Compensated Crystal Oscillator (TCXO) instalado [20].

Consoante a escolha do *Spreading Factor* é possível fazer o cálculo da sensibilidade do recetor. A sensibilidade do recetor mede a habilidade do recetor em realizar a desmodulação e obter a informação de um sinal fraco, quanto maior a sensibilidade maior a capacidade do recetor receber a mensagem a longa distancia do transmissor [21]. A sensibilidade pode ser calculada usando a Equação 2 [19].

$$Sens = 10 \times \log(k \times T \times LB) + Ruido + SNR \quad (dB) \qquad (2)$$

Onde o $k \times T \times LB$ representa o ruido térmico (Ruído Johnson–Nyquist), que é composto por três componentes, o "k" que representa a constante de Boltzmann ($1{,}38 \times 10^{-2}\ J/K$), o T, indicativo da temperatura em Kelvin, e a largura de banda (LB) em Hz. A potencia do ruido pode ser calculada em decibéis (mostrado na equação 2) e para a temperatura ambiente aproximada (27 ºC = 300 K). O calculo do ruido térmico em dBm é feito na Equação 3.

$$\begin{aligned} Ruido\ termico &= 10 \times \log(1{,}38 \times\ 10^{-23} \times 300 \times LB \times 1000) \quad (dBm) \qquad (3) \\ &= 10 \times \log(1{,}38 \times\ 10^{-23} \times 300 \times 1000) + 10\ \times \log(LB) \quad (dBm) \\ &= -174 + 10 \times \log(LB) \quad (dBm) \end{aligned}$$

Desenvolvendo a fórmula podemos obter uma versão simplificada da equação da sensibilidade do recetor estimada para a temperatura de 27 ºC. Essa equação está descrita na Equação 4 [21] [22].

$$Sens = -174 + 10 \times \log(largura\ de\ banda) + Ruido + SNR \quad (dB) \qquad (4)$$

Depois de calculada a sensibilidade é possível saber a potencia mínima aceitável para o recetor receber a mensagem que permite ao utilizador desta tecnologia estimar o alcance do sistema através da fórmula de Friis mostrada na Equação 5 [23].

$$\frac{P_{recebida}}{P_{transmitida}} = \frac{A_{recebida} \times A_{transmitida}}{d^2 \times \lambda^2} \qquad (5)$$

A fórmula de Friis é usada em engenharia de telecomunicações e permite relacionar as potências do recetor e transmissor, as aberturas das antenas usadas na comunicação ($A_{recetor}\ e\ A_{transmissor}$), o comprimento de onda e a distância entre o transmissor e o recetor, em condições ideais. De forma a usar os valores em decibéis é possível utilizar a variação da fórmula de Friis mostrada na Equação 6 [24].

$$P_{recebida} = P_{transmitida} + G_{transmitor} + G_{receptor} + 20 \times \log_{10} \frac{\lambda}{4 \times \pi \times d} \qquad (6)$$

As potências são medidas em dBm e os ganhos das antenas são medidas em dBi (*decibels-isotropic*). As medidas da distância entre o recetor e do transmissor ("d") e o comprimento de onda ("λ") tem de ser da mesma grandeza. Para calcular o comprimento de onda através da frequência utilizada na comunicação é utilizada a Equação 7.

$$\lambda = \frac{c}{f} \qquad (7)$$

A variável "f" corresponde à frequência da onda e "c" à velocidade. Tratando-se as ondas rádios de ondas eletromagnéticas a velocidade é igual a velocidade da luz (299.792.458 m/s).

Utilizando um exemplo prático de modo a conhecer o alcance máximo teórico desta tecnologia em situações ideais são feitos os cálculos tendo por base as configurações seguintes: um *Spreading Factor* de 7, largura de banda de 125 KHz, uma potência de transmissão de +13 dBm e a utilização de antenas que amplificam o sinal em 2 dBi.

O primeiro passo para calcular o alcance estimado da comunicação é calcular a sensibilidade, usando para isso a Equação 2. Considerando um ruido de 5 dB na comunicação e mais perdas da transmissão no valor de 5 dB (devido a cabos, conectores,

etc.). O valor de SNR para um *Spreading Factor* de 7 é de 7,5 (ver Figura 4), assim o cálculo da sensibilidade é mostrado na Equação 8.

$$Sens = -174 + 10 \times \log_{10} 125000 + 10 - 7{,}5 \quad (8)$$

A sensibilidade calculada para um *Spreading Factor* de 7 é igual a -120,5 dBm. Sabido a potencia recebida mínima podemos calcular a distancia estimada em condições ideais reformulando a fórmula da Friir (Equação 6) da seguinte maneira (Equação 9).

$$d = \frac{\lambda}{4 \times \pi \times 10^{\frac{P_{recebida} - P_{transmitida} - G_{recetor} - G_{transmissor}}{20}}} \quad (9)$$

Para calcular o comprimento de onda através da frequência de comunicação (434 MHz) é utilizada a Equação 7 e sabendo os ganhos das antenas utilizadas (2 dbi) e a potência de transmissão (13 dbm) podemos calcular o alcance em campo aberto (Equação 10).

$$d = \frac{\frac{299792458}{434000000}}{4 \times \pi \times 10^{\frac{-120{,}5-13-2-2}{20}}} = 388722\ m = 388{,}7\ Km \quad (10)$$

Realizando os mesmos cálculos mas utilizando um *Spreading Factor* de 12 é obtido a distância máxima de 1738 km.

Tais distanciais são possíveis de atingir utilizando configurações muito específicas e não são esperadas em testes de utilização reais comuns. Por exemplo, em agosto de 2017 foram lançados balões de hélio utilizados em estações metrológicas durante um evento nos Países Baixos. Com esses balões a uma altura de 38,7 Km de altura foi possível estabelecer comunicação a mais de 700 Km de distância utilizando apenas uma potência de 14 dBm. Utilizando também balões de hélio a 15 km de altura uma companhia holandesa conseguiu que um pacote de dados percorresse uma distância de 354 km [25]. No entanto, em zona urbana sem obstáculos entre o transmissor e o recetor o intervalo de alcance desta tecnologia varia geralmente entre 500 metros a 5 Km. Na Figura 5 é mostrado os pacotes recebidos por uma *gateway* LoRa com uma antena de 8 dBi em função da distância do transmissor e a potência recebida pelo recetor, RSSI (received signal strength indicator).

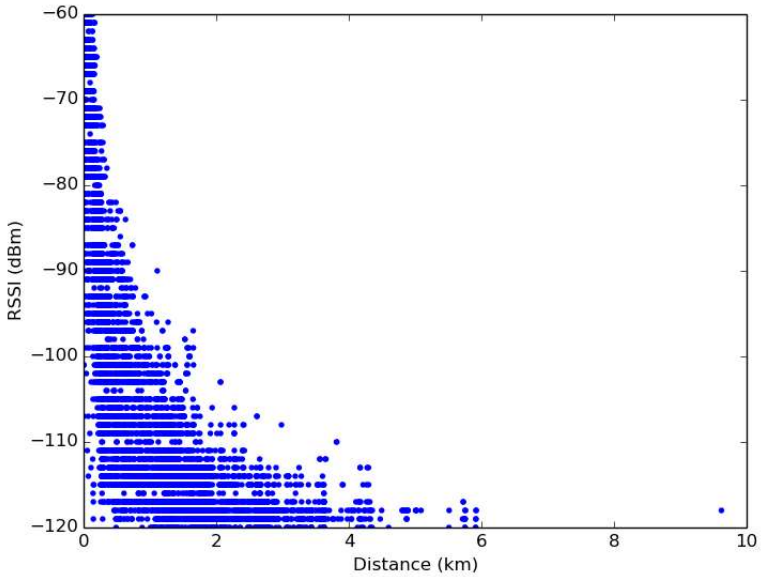


Figura 5 - Pacotes recebidos por uma *gateway* LoRa em função da distância do transmissor e do RSSI [26]

Em suma, a tecnologia LoRa ocupa um lugar entre as tecnologias de comunicação sem fios permitindo um longo alcance e baixa potência mas não substitui outras tecnologias de comunicação já existentes, sendo interessante para algumas aplicações que não exijam muita velocidade e taxas de transferência de dados altas.

2.2 Low-Power Wide-Area Network (LPWAN)

Low power wide area network (LPWAN) é um tipo de comunicação sem fios usada em grandes áreas, desenvolvida para operar a baixa potência e com baixas taxas de transmissão. Este tipo de rede conecta vários equipamentos estabelecendo uma ou várias centrais chamadas *gateways*, apesar de não serem obrigatórias na LPWAN as *gateways* controlam o trafego dos equipamentos na rede e oferecem ligação ao exterior. Existem no mercado vários tipos de redes LPWAN como a LoRaWAN, Sigfox, Nwave entre outras baseando-se em tecnologias diferentes de comunicação mas todas com princípios iguais como o grande alcance, baixa potência e baixa taxa de transferência. A LPWAN oferece melhores caraterísticas que não se enquadram em nenhuma outra tecnologia equivalente como o alcance e a cobertura geográfica como podemos ver na Figura 6.

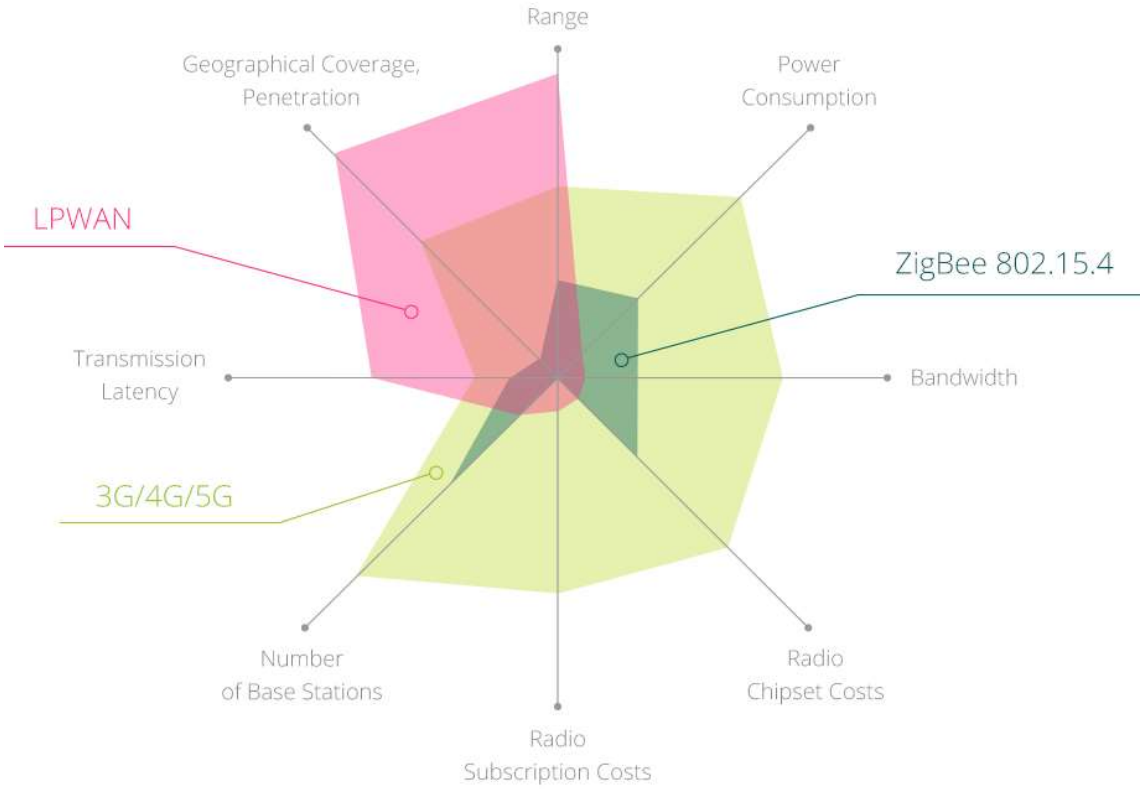


Figura 6 - Comparação das caraterísticas da LPWAN, 3G/4G/5G e ZigBee [27]

A LPWAN apresenta vantagens consideráveis em termos de penetração, pois utiliza frequências de comunicação mais baixas que outras redes, abaixo da banda S (ver Tabela 1) o que é ideal para áreas urbanas com muitos obstáculos à comunicação. Devido ao seu baixo consumo os equipamentos na rede LPWAN podem funcionar durante anos alimentados apenas por pequenas baterias. Além disso, módulos de comunicação das redes LPWAN são relativamente baratos dependendo do tipo de tecnologia de transmissão utilizado [27].

A LPWAN assenta em 3 redes fundamentais, uma rede de longo alcance que utiliza uma tecnologia de comunicação de longo alcance e baixa potência, como a tecnologia LoRa, um servidor exterior ligado à internet e uma rede local para os utilizadores. A rede de longo alcance é constituída por vários equipamentos, como sensores e atuadores chamados de *end devices* que comunicam com *gateways* que estão ligadas à internet. Estas *gateways* tem a função de transmitir os dados recebidos processá-los e transmiti-los ao servidor do sistema na internet. O servidor receberá os pedidos vindos da rede local, feitos pelos utilizadores do sistema, e fornecerá os dados pedidos. Na Figura 7 está uma representação geral da rede LPWAN.

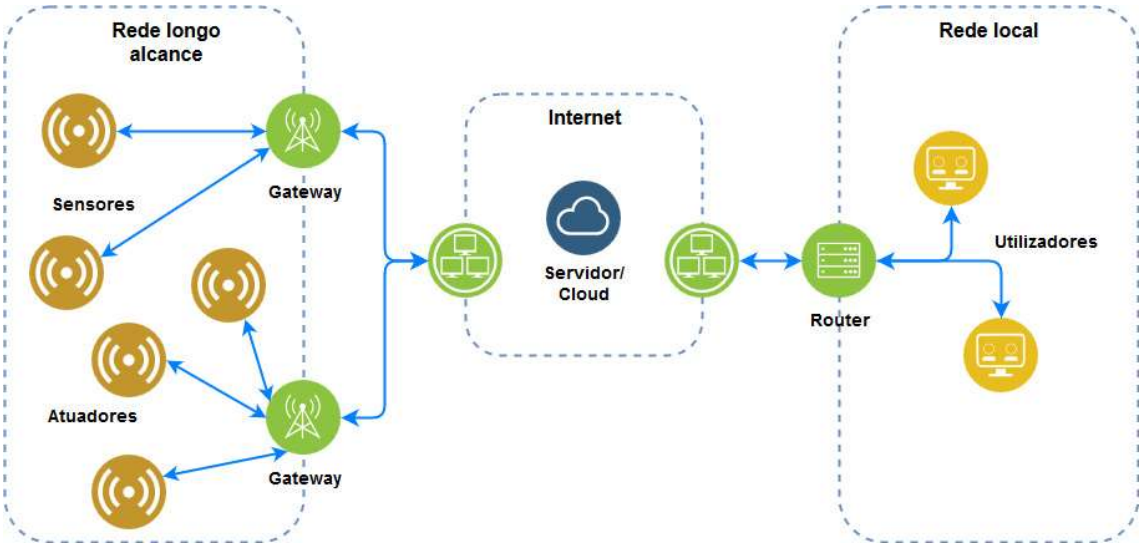


Figura 7 - Arquitetura geral de uma rede LPWAN

Um tipo específico de protocolo LPWAN é o LoraWAN que consiste numa rede LPWAN que utiliza a tecnologia LoRa para a rede de longo alcance. O LoRaWAN é desenvolvido pela LoRa Alliance, fundada em 2015 por um conjunto de centenas de empresas incluindo a dona da tecnologia LoRa a Semtech. Desde a sua criação a LoRaWAN tornou-se no protocolo mais popular de LPWAN juntamente com o Sigfox. Sigfox é uma abordagem diferente à LPWAN e por isso não se pode chamar de concorrente à LoRaWAN.

Existe redes LoraWAN privadas e redes públicas, sendo que as redes privadas, construídas pelas empresas de telecomunicações, necessitam de uma mensalidade para serem utilizadas. As redes públicas apesar de serem livres para serem utilizadas por qualquer pessoa podem não ter muita cobertura geográfica e são suportadas pela construção das *gateways* pelos seus utilizadores [28].

A LoRaWAN utiliza a arquitetura em estrela em vez da *mesh* (Figura 2), isto porque apesar da rede *mesh* permitir um maior alcance fazendo os *end nodes* comunicarem entre si isto reduz também a velocidade de comunicação e o gasto de energia, além de introduzir complexidade à comunicação. Sendo assim cada *end node* comunica diretamente com a *gateway* específica o que faz reduzir a necessidade de energia. As *gateways*, conectadas à internet, comunicam através do protocolo TCP/IP com o servidor do sistema o servidor e por sua vez comunica com o utilizador.

A arquitetura da LoRaWAN garante segurança em toda a rede através da encriptação AES em todos os pacotes transmitidos [17]. Na Figura 8 está representada a arquitetura da rede LoRaWAN.

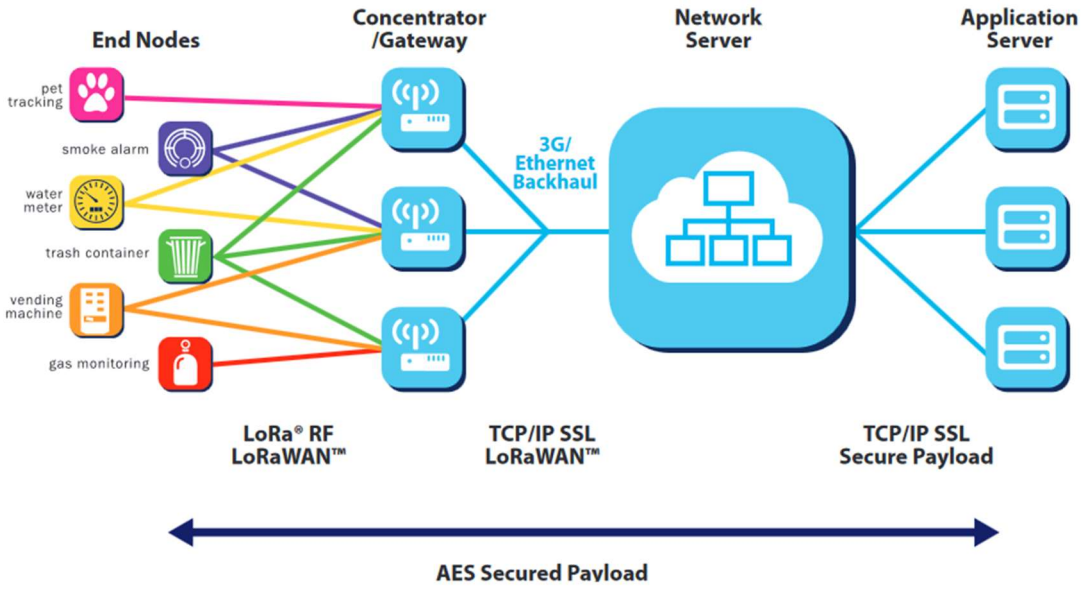


Figura 8 - Arquitetura da rede LoRaWAN [17]

A tecnologia de comunicação sem fios LoRa é a base fundamental da LoRaWAN, a camada física. O *end node* é composto por um microcontrolador ou microprocessador que usa a comunicação SPI (Serial Peripheral Interface) para comunicar com o módulo de transmissão LoRa que por sua vez comunica com a *gateway* com uma configuração semelhante. A *gateway*, que tem ligação ao módulo LoRa e à rede Internet e faz a ligação do *end node* ao servidor do sistema através de Wi-Fi, 2G/3G/5G ou Ethernet. Na Figura 9 está uma representação do protocolo LoRaWAN.

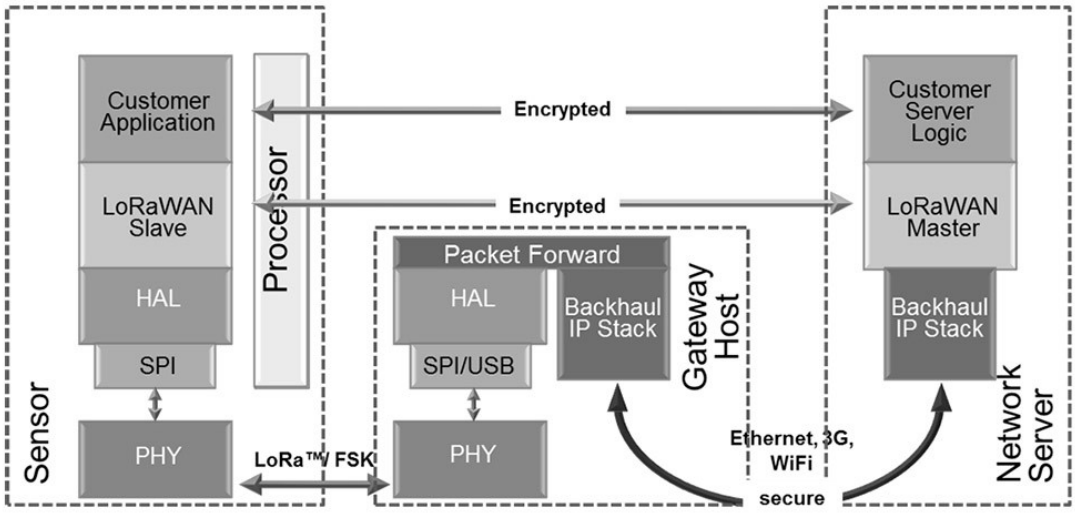


Figura 9 - Protocolo LoRaWAN [28]

2.3 Dispositivos no Mercado

De modo a conhecer as caraterísticas de sistemas de domótica semelhantes a este projeto foi feito um estudo de equipamentos já no mercado. Os dispositivos que se destacaram foram o sistema da Oomi com a tecnologia Z-Wave e o sistema da Wulian que utiliza a tecnologia ZigBee.

Para perceber melhor o funcionamento de um sistema de domótica e as caraterísticas de um sistema com a tecnologia Z-Wave fez-se um estudo aos produtos da Oomi, o Oomi In-Wall Switch e o Oomi Plug.

O Oomi In-Wall Switch é um dispositivo desenvolvido para dar a capacidade de automação a interruptores domésticos. O equipamento tem capacidade para dois interruptores independentes e é alimentado diretamente com a rede doméstica de 220V, nos pinos N e L. A alimentação do aparelho a ser controlado pode ser variada e é colocada nos pinos IN e OUT. Para a ligação aos interruptores é usada a tensão de entrada do equipamento. O esquema de ligação é mostrado na Figura 10.

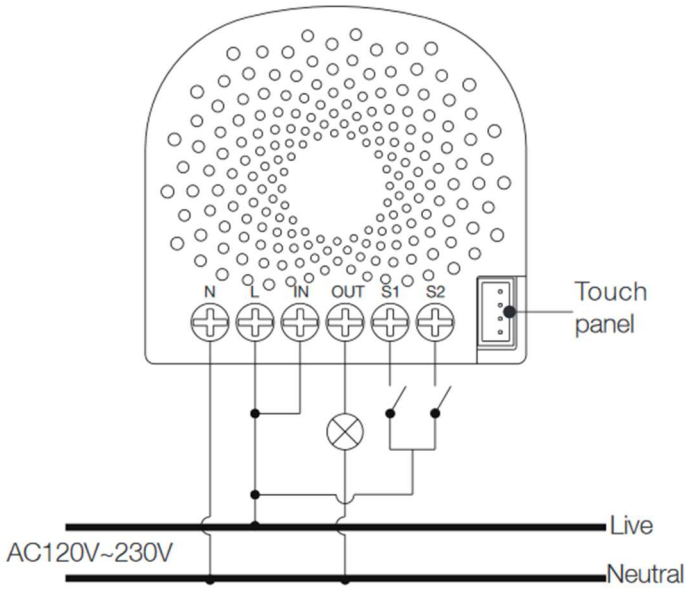


Figura 10 - Esquema de ligação elétrica do equipamento Oomi In-Wall Switch [29]

Os interruptores, geralmente de iluminação, são mantidos durante a instalação e o equipamento é colocado atrás deste, dentro da parede. Conforme mostra a Figura 11.

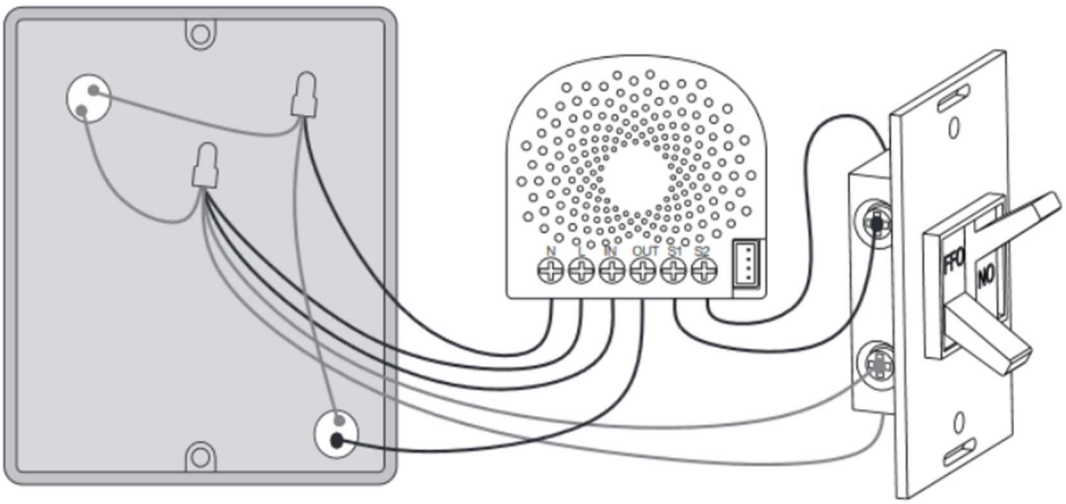


Figura 11 - Esquema de instalação do equipamento Oomi In-Wall Switch [29]

Para o controlo de aparelhos ligados a tomadas domésticas a Oomi desenvolveu um equipamento capaz de conectar a uma tomada convencional e ligar ou desligar o aparelho conectado a esta através da rede Z-Wave. Esse equipamento é o Oomi Plug que para além de ligar e desligar qualquer aparelho pode também ver o seu consumo e mostrá-lo ao utilizador. Ainda inclui um led multicor indicativo que permite mostrar o sinal da rede Z-Wave. Uma representação gráfica do Oomi Plug é mostrada na Figura 12.

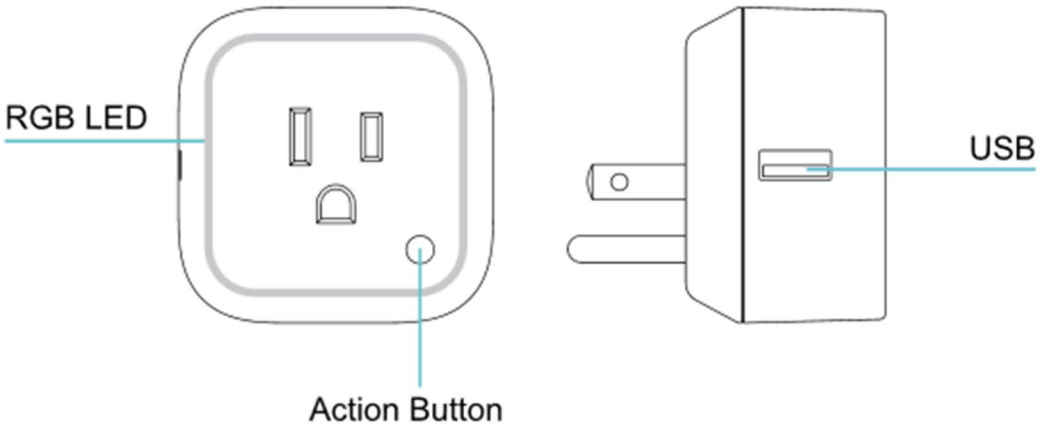


Figura 12- Representação do Oomi Plug [29]

O sistema sem fios da Wulian apresenta uma arquitetura semelhante aos restantes sistemas de domótica com comunicação sem fios. Este sistema é controlado através de uma aplicação para telemóvel chamada de Smart Home, que permite a criação de cenários de automação. Uma instalação básica deste sistema de domótica é composta no mínimo por um módulo controlador principal (Smart Gateway) e um módulo de controlo (atuador), como um interruptor inteligente, ou um sensor, como um sensor volumétrico.

Para perceber melhor o funcionamento do sistema fez-se um estudo do módulo controlador Smart Gateway e do Sensor Smart PIR Motion Sensor.

O módulo controlador é alimentado a 5 V e necessita de ligação à rede domestica LAN ou à internet para funcionar. Depois de ligados os cabos o Ethernet Indicator e o Power Indicator devem estar acessos. A rede ZigBee é criada de seguida pela Gateway e passado dois minutos a rede está pronta e o indicador ZigBee Network Indicator acende. Para adicionar o equipamento à rede pela primeira vez é clicar no botão SET Key e para desconectar o equipamento é usado o botão SYS Key. Na Figura 13 é possível ver uma ilustração da Smart Gateway Wulian.

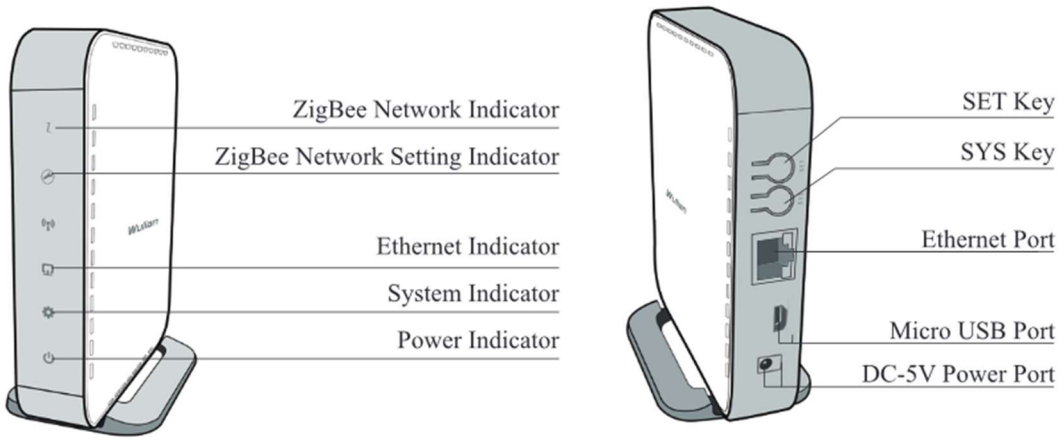


Figura 13 - Smart Gateway Wulian [30]

O equipamento Smart PIR Motion Sensor é um sensor de movimento que faz parte do sistema de domótica da Wulian. Este dispositivo alerta o utilizador sempre que deteta movimento dentro da área do sensor enviando um alerta para a plataforma do sistema. O sensor pode ser colocado em qualquer superfície plana e não requer ligação elétrica à rede doméstica, em vez disso, este equipamento utiliza baterias para o seu funcionamento. Depois de instalado o equipamento tem de ser colocado dentro da rede ZigBee criada pela Smart Gateway, para isso é necessário clicar no botão Multi-Functional Key que serve também para desconectar o equipamento da rede. Na Figura 14 está representado o Smart PIR Motion Sensor Detector assim como alcance do sensor.

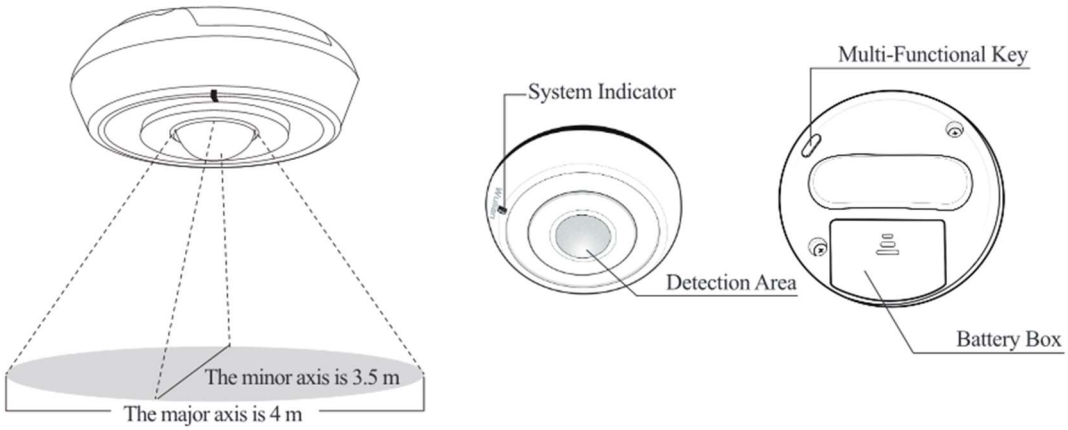


Figura 14 - Smart PIR Motion Detector Wulian [31]

3 Projeto

Neste capítulo serão estruturadas e introduzidas as fases mais relevantes do projeto como a arquitetura, o *hardware*, tecnologias, programas e linguagem de programação utilizados.

3.1 Análise de requisitos

Após a análise realizada no capítulo 2, é necessário definir de forma estruturada os requisitos deste projeto. O projeto será desenvolvido de forma a cumprir determinadas condições previamente estabelecidas neste capítulo que terão por base os objetivos definidos no capitulo 1.3. Os requisitos deste projeto são:

- Criar um sistema de domótica que permita o controlo da iluminação, tomadas e sensores através de uma interface própria;
- Possibilitar o controlo do sistema a partir de uma rede Wi-Fi e permitir acesso à internet;
- Permitir a criação de automações e cenários;
- Permitir uma comunicação, sem fios, ao sistema que seja segura;
- Garantir um longo alcance e baixo consumo de energia ao sistema;
- Conceber uma interface simples e intuitiva que permita a configuração do sistema e a consulta do histórico de dados;
- Permitir a integração de outros sistemas de domótica na mesma interface;
- Possibilitar uma instalação fácil do sistema sem necessidade de alterar a instalação elétrica;
- Tornar o sistema o mais barato e compacto possível.

Para ser possível cumprir com estes requisitos o produto final deverá ser dividido em vários dispositivos que comunicam com uma central base de controlo. A central de controlo terá de gerir a comunicação feita com os equipamentos a controlar (iluminação e tomadas) e os sensores, gerir a interface de controlo e disponibilizar configurações do sistema ao utilizador. A central deve também disponibilizar a interface aos utilizadores através da rede Wi-Fi e o sistema deverá possibilitar o acesso à internet mas não depender deste acesso para o seu funcionamento. Será necessário a criação de uma interface que

permita configurações no sistema relacionadas, por exemplo, com a rede Wi-Fi e também permitir a inclusão ou retirada de equipamentos de domótica do sistema, de forma simples.

O sistema de domótica terá de utilizar uma comunicação sem fios que garanta um longo alcance e necessite de pouca energia para o seu funcionamento. Além disso, a comunicação utilizada deve ser encriptada de modo a garantir a segurança dos utilizadores do sistema. O *software* de domótica, a utilizar, deverá aceitar a inclusão do sistema de domótica criado assim como outras plataformas já disponíveis no mercado permitindo em simultâneo a utilização de vários equipamentos com modos de operação diferentes, interagindo entre si através da mesma plataforma e permitindo também a criação de automações e cenários.

A instalação do sistema deve ser o mais simples possível de modo que seja rápida e não tenha um custo elevado para o utilizador. Para isso terá de ser possível instalar o sistema num edifício que já tenha a instalação elétrica concluída e adapta-lo às necessidades de cada cenário proposto pelo utilizador.

A alimentação dos equipamentos deverá ser feita a partir da rede doméstica (220 VAC) nos equipamentos atuadores (iluminação e tomadas elétricas) e na central de controlo. Os sensores deverão ter a possibilidade de ser alimentados por bateria para além da alimentação a partir da rede doméstica.

Por fim, os dispositivos que compõem o sistema devem ser o mais pequenos possível, além disso a escolha dos materiais e componentes deve ter em conta o seu custo de modo que o produto final seja o mais barato possível cumprindo os restantes requisitos estabelecidos.

3.2 Arquitetura do Sistema

O sistema de domótica a ser criado será chamado de AssistLora e será composto por uma unidade central, chamada de *gateway*, pelos sensores e pelos atuadores. Num sistema existirá sempre apenas uma *gateway* e um sensor ou atuador mas o sistema pode ser adaptado às necessidades do utilizador usando um número maior de sensores e atuadores necessários para cada caso. A comunicação sem fios será feita entre a *gateway* e os sensores e atuadores recorrendo para isso a módulos rádio com capacidade de longo alcance e baixa potência. Dois módulos de rádio são ligados à *gateway* permitindo assim a operação de dois estilos diferentes de comunicação, médio alcance e comunicação

rápida que se torna ideal para atuadores de muito uso, e longo alcance com comunicação lenta, ideal para sensores e alguns atuadores que se encontram a grande distância da *gateway*. A *gateway* criará uma rede Wi-Fi própria, chamada AssistLora, onde os utilizadores se podem ligar para aceder à interface de controlo e às configurações através de um equipamento com ligação Wi-Fi. Caso seja necessário, será ainda possível ligar o sistema à internet através de outra interface de rede permitindo assim ter o controlo do sistema a partir de qualquer lugar. Na Figura 15 é feita uma comparação do tamanho das redes usadas no sistema.

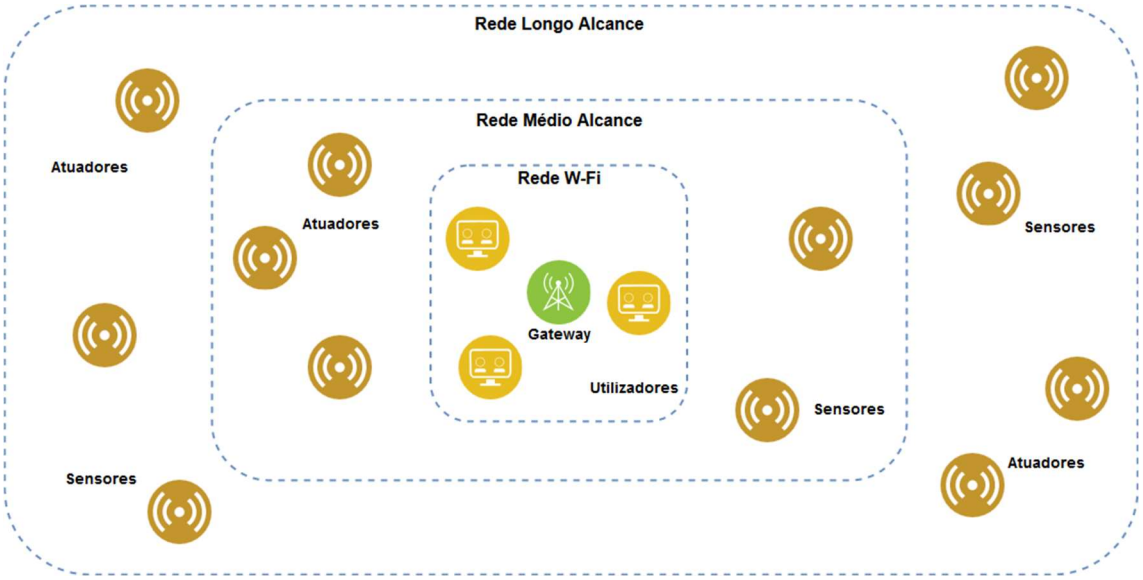


Figura 15 - Comparação da dimensão das redes usadas no sistema AssistLora

As redes de longo alcance e medio utilizam a tecnologia LoRa para a comunicação, a *gateway* faz a gestão dos dispositivos e a sua interligação com os utilizadores do sistema que utilizam a rede Wi-Fi. Sendo assim, a *gateway* tem suporte para as duas tecnologias de comunicação usadas, LoRa e Wi-Fi e permite o acesso da interface de controlo do sistema aos utilizadores através de um servidor incluído. O acesso exterior ao sistema pode ser feito ligando a *gateway* à internet permitindo o controlo em qualquer lugar. A arquitetura da rede do sistema AssistLora é mostrado na Figura 16.

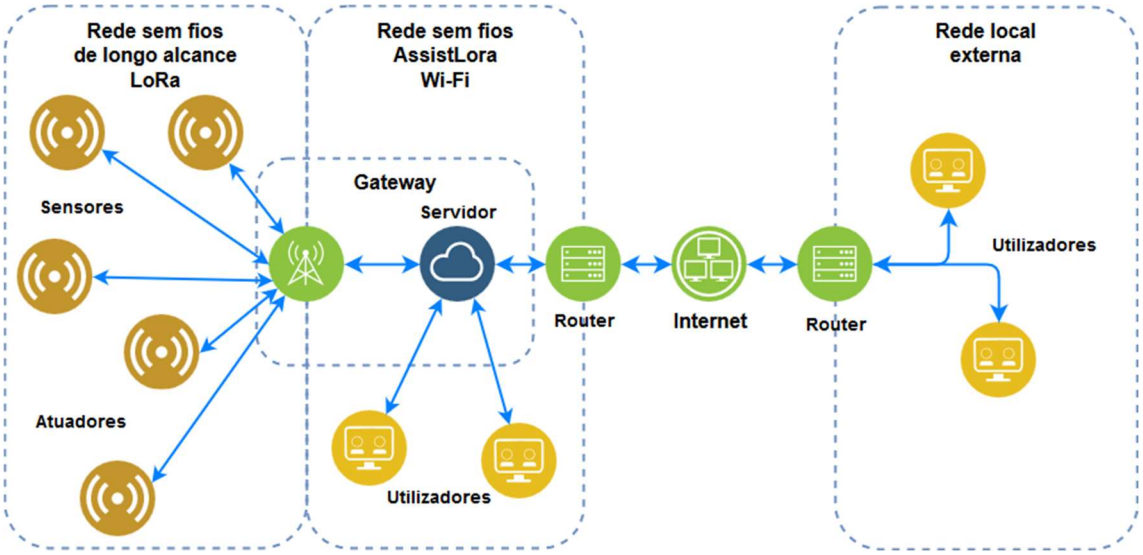


Figura 16 - Arquitetura geral da rede

Existirão quatro equipamentos diferentes de domótica no AssistLora, o sensor binário, o sensor analógico, o controlo de tomada e o controlo de iluminação. A configuração destes equipamentos será semelhante tendo um microcontrolador de controlo, um módulo rádio, configurado conforme as necessidades e uma ligação aos periféricos necessários ao funcionamento do sensor ou atuador. Na Figura 17 está representado o diagrama de blocos de alto nível do sistema AssistLora.

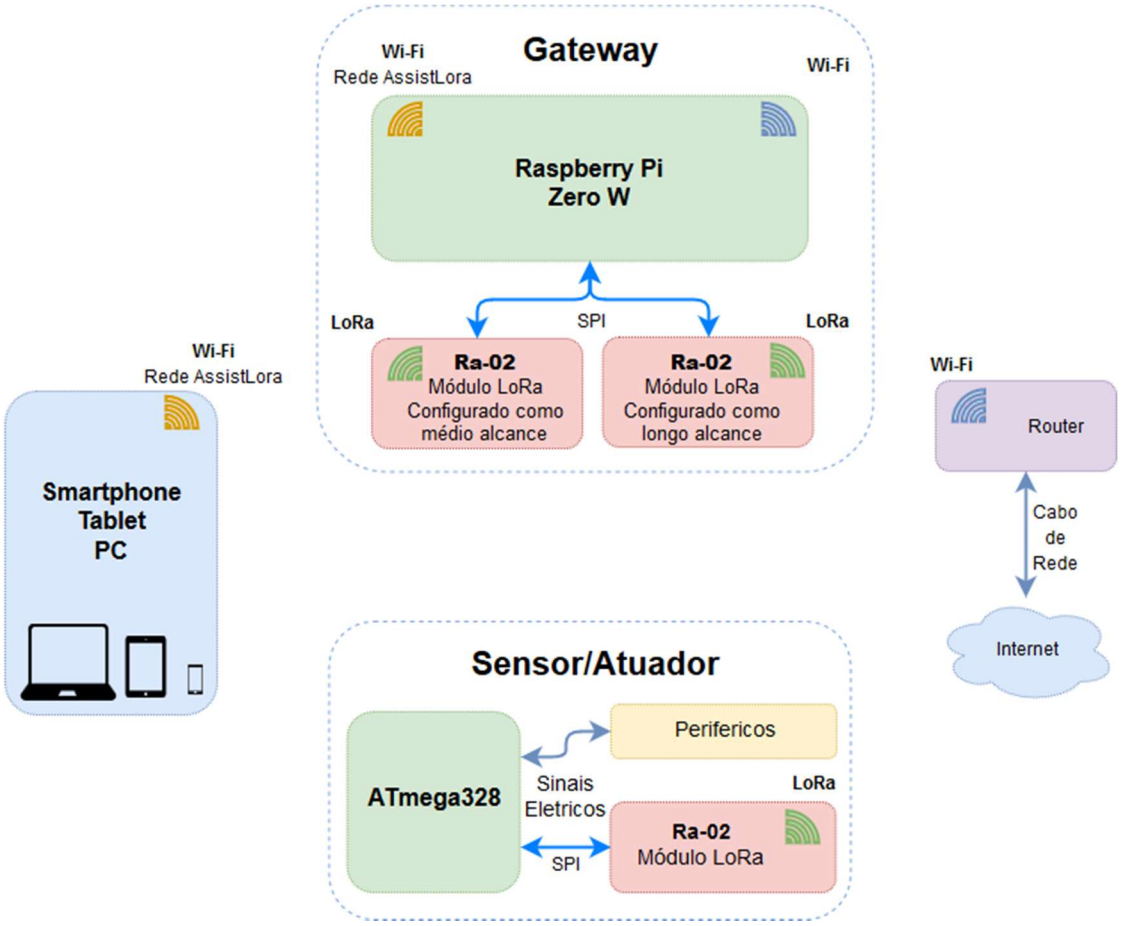


Figura 17 - Diagrama de blocos de alto nível

3.3 Descrição do *Hardware*

Aqui serão apresentados todos os elementos de *Hardware* escolhidos para integrar neste projeto.

3.3.1 Microcomputador

Para a unidade de controlo da *gateway* era necessário um microcomputador, o escolhido foi o Raspberry Pi Zero W. Este *single-board computer* foi desenvolvido no Reino Unido pela Raspberry Pi Foundation, lançado em fevereiro de 2017, é o mais pequeno microcomputador de todos os desenvolvidos pela fundação com suporte Wi-Fi 802.11n e Bluetooth 4.1. O Raspberry Pi Zero W está representado na Figura 18.

Figura 18 - Raspberry Pi Zero W [32]

O Raspberry Pi Zero W apresenta as caraterísticas básicas necessárias para cumprir com os requisitos deste projeto, nomeadamente a conectividade Wi-Fi, a dimensão e o preço. Este microcomputador possui vários pinos I/O (Input/Output) disponíveis e comunicação SPI, I²C (Inter-Integrated Circuit), UART (Universal Asynchronous Receiver-Transmitter), tem disponível também 5 V e 3,3V no seu GPIO. Na Tabela 2 encontram-se mencionadas as caraterísticas principais do Raspberry Pi Zero W.

Tabela 2 - Caraterísticas principais do Raspberry Pi Zero W [33]

CPU	Single-Core ARM1176JZF-S
SoC	Broadcom BCM2835
Frequência do Clock	1 GHz
Memória SDRAM	512 MB (partilhada com o GPU)
Periféricos	1 Micro-USB (USB 2.0 direto do SoC); Saída de vídeo 1080p60 por Mini-HDMI; Saída de vídeo composto; Áudio Stereo; Conector CSI Câmera Wi-Fi 802.11n e Bluetooth 4.1 e BLE
Tensão de alimentação	5 V por Micro USB
Consumo	Mínimo: 0,5 W Máximo 1,75W

Periféricos de baixo nível	17× GPIO (General Purpose Input/Output), UART, I²C, SPI com 2 chip selects, I²S áudio

O Raspberry Pi Zero W é um microcomputador de baixa potência tendo um consumo mínimo de apenas 0,5 W (5 V × 100 mA) podendo chegar aos 1,75 W (5 V × 350 mA) com elevado nível de processamento. Este consumo depende dos periféricos que estão ligados internamente ou externamente ao microcomputador, por exemplo, a porta HDMI usa cerca de 50 mA mas pode ser desligada caso não seja usada, e caso o módulo da câmera seja usado o consumo de corrente é elevado em 250 mA. Já um periférico externo ligado ao USB pode fazer elevar muito o consumo pois o Raspberry Pi Zero W não impõe limites de corrente na porta USB. O GPIO (General Purpose Input/Output) do Raspberry Pi Zero W é igual às restantes versões recentes do Raspberry Pi, possui 17 GPIO pinos que podem ser configurados como entrada ou saída e também para comunicação de dados como UART, I²C, SPI, e I²S áudio, a tensão usada no GPIO é de 3,3 V. Cada pino pode fornecer cerca de 16 mA de corrente com segurança, um consumo superior corre-se o risco de danificar o microcomputador. Ainda estão disponíveis as tensões de 3,3 V e 5 V que podem fornecer até um máximo de 1 A dependendo sempre da fonte de alimentação do microcomputador.

O Raspberry Pi Zero W possui um Broadcom SoC (System On Chip) que inclui o CPU ARM1176JZF-S, GPU (Graphics Processing Unit), as portas I/O, a memória SDRAM de 512 MB, e outras funções num único componente eletrónico. O processador do Raspberry Pi inclui o processador ARM1176JZF-S que apresenta um único núcleo e utiliza arquitetura ARM de 32 bits, sendo assim, qualquer programa para ser executado no Raspberry Pi tem de compilar nesta arquitetura.

Toda a memória física do microcomputador é gravada num micro SD, incluindo o sistema operativo. Existem vários sistemas operativos disponíveis para o Raspberry Pi a maioria deles é baseada em Linux incluindo a distribuição oficial da Raspberry Pi Foundation o Raspbian [33].

3.3.2 Microcontrolador

Para os atuadores e sensores do AssistLora era necessário escolher um microcontrolador, este terá a função de controlo do módulo rádio, leitura de sinais e também o controlo de periféricos. O microcontrolador escolhido, que preenche os requisitos deste projeto, foi o

ATmega328P. O ATmega328P é um microcontrolador de 8 bits criado pela Atmel que pertence agora à Microchip Technology Inc.. De baixo custo e de baixa potência, o ATmega328P reúne as funcionalidades básicas de um microcontrolador, possui 32 kB de memória *flash*, 1 kB de memoria EEPROM, 23 pinos de I/O, vários estilos de comunicação disponíveis como USART, SPI e I²C, e também um A/D de 10 bits com 8 canais.

Para o desenvolvimento do projeto foi escolhido um módulo que contivesse o ATmega328P, que fosse pequeno e de baixo custo e que tivesse todos os componentes necessários para o funcionamento do microcontrolador como o oscilador e o botão Reset num espaço pequeno. O módulo escolhido foi o Arduino Pro Mini, que contem apenas o microcontrolador, um oscilador cristal externo, um botão de Reset e um regulador linear de tensão que como não é necessário será removido da placa. Existem varias versões deste módulo desde que foi introduzido pela Sparkfun, e duas variações, a de 5 V equipado com um regulador linear de tensão para 5 V, um oscilador de 16 MHz e outra versão com 8 MHz que pode ser alimentado a tensões mais baixas como 3,3 V. O módulo usado neste projeto será a versão de 3,3 V com o oscilador de 8 MHz e está representado na Figura 19.

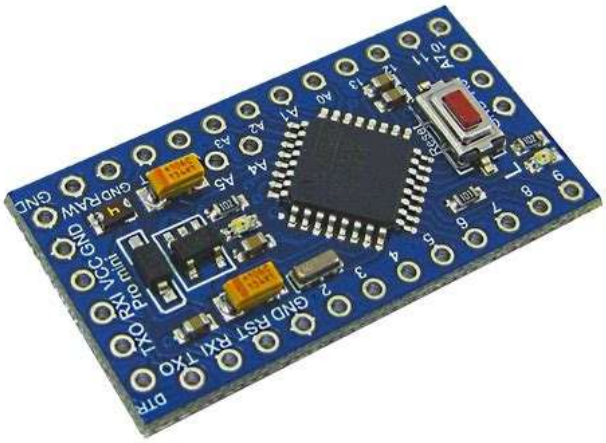


Figura 19 - Módulo Arduino Pro Mini usado no projeto [34]

3.3.3 Módulo Rádio

O módulo rádio a ser escolhido para a comunicação sem fios teria de permitir a comunicação a longo alcance e ter um consumo baixo, por essa razão restringiu-se a procura de módulos rádio por aqueles que permitissem a comunicação através da modulação LoRa. O módulo escolhido foi o Ra-02 LoRa module desenvolvido pela empresa Ai-Tinker. A escolha deste módulo em detrimento de outros com maior alcance

deveu-se ao seu baixo custo e à inclusão do adaptador para antena no módulo permitindo ligar a antena facilmente. Na Figura 20 é mostrado o módulo LoRa Ra-02.

Figura 20 - Modulo Ra-02 LoRa [35]

Este módulo utiliza o chip SX1278 produzido pela SEMTECH e permite uma potência de transmissão até +18 dBm.e permite vários tipos de modulações diferentes na comunicação sem fios, incluindo a tecnologia LoRa, e utiliza a frequência de 434 MHz para a comunicação. O módulo pode ser alimentado com um tensão de 2,5 V a 3,7 V podendo ser alimentado com a tensão típica de 3,3 V. O Ra-02 tem um consumo mínimo de energia sendo que em transmissão consome apenas 93 mA quando alimentado a 3,3 V (0,307 W) e pode ser configurado para modos de menos consumo como o *Standby* com um consumo de 5 mW (3,3 V × 1,6 mA) mantendo operacional o oscilador cristal ou o modo *Sleep* com um consumo de apenas 0,66µW (3,3 V × 0,2 µA) [36] [37].

O módulo Ra-02 ainda possui 6 pinos I/O que podem ser configurados para funcionar como interrupções, como por exemplo a receção de um pacote ou quando a transmissão estiver concluída. Na Tabela 3 estão enumeradas as principais caraterísticas do módulo Ra-02.

Tabela 3 - Principais caraterísticas do módulo Ra-02 Lora [35] [37]

Transmissor chip	SX1278
Modulações de comunicação rádio suportadas	FSK, GFSK, MSK, GMSK, LoRa e OOK
Frequência de comunicação	De 410 MHz até 525 MHz
Interface de comunicação	SPI
Potência máxima de transmissão	18±1 dBm
Tensão de alimentação	2,5 V até 3,7 V

Consumo	A transmitir: 0,307 W À escuta: 0,040 W Em standby: 0,005 W
Periféricos de baixo nível	6 pinos digitais I/O

Para a utilização deste módulo será necessário usar um adaptador já que o formato do módulo é SMD (Surface Mount Device).

As antenas escolhidas amplificam o sinal em 2 dbi e estão preparadas para a transmissão na frequência 434 MHz de modo a funcionar com o módulo Ra-02.

Com o Ra-02 é espectável realizar a comunicação até uma distância de 500 metros que será suficiente para a realização do protótipo e a sua validação, podendo ser substituído por um módulo rádio com maior alcance num projeto futuro.

3.3.4 Alimentação

A alimentação dos dispositivos que fazem parte do AssistLora depende do tipo de equipamento. A *gateway*, o sensor binário, o sensor analógico, o atuador de tomada e o atuador de iluminação têm configurações diferentes por causa da função que desempenham no sistema. A unidade de processamento da *gateway* (microcomputador) é alimentada a partir de uma entrada USB micro a 5 V que o Raspberry Pi possui. Qualquer fonte com um mínimo de 0,5 W de potência pode alimentar a *gateway*. Na placa do Raspberry Pi os 5 V são transformados em 3,3 V através de um conversor estático Buck-boost, essa tensão está disponível no GPIO da placa e é usada para fazer a alimentação de dois módulos rádio Ra-02 LoRa que serão posteriormente configurados de modo diferente. Os módulos rádio comunicam com o Raspberry Pi através de comunicação SPI usando o CE0 e o CE1 de modo a identificar cada módulo. Os pinos I/O dos módulos Ra-02 são ainda ligados ao GPIO do microcomputador. O Raspberry Pi possui uma ligação USB que permite a ligação de um adaptador Wi-Fi e assim ligar o sistema à internet caso seja necessário.

Na Figura 21 está representado o diagrama de blocos da central do sistema, a *gateway*.

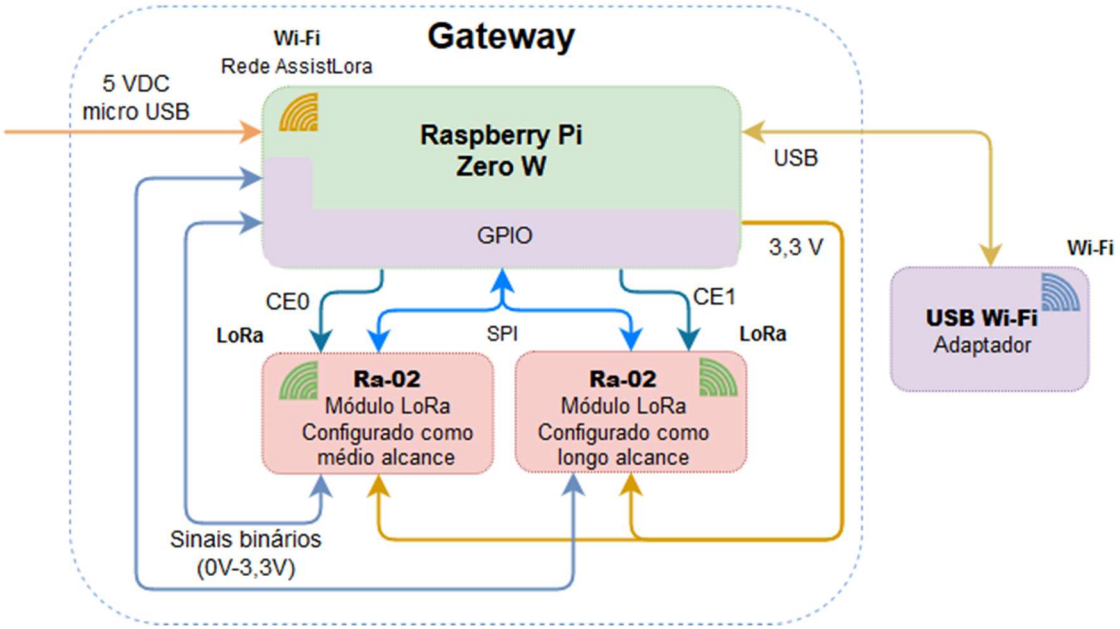


Figura 21 - Diagrama de blocos da *gateway*

Os sensores e os atuadores têm são semelhantes em termos de configuração no entanto diferem em pequenos aspetos. Todos os sensores e atuadores tem como unidade de processamento o microcontrolador ATmega328P e possuem um módulo rádio Ra-02 LoRa. Os sensores estarão equipados com uma bateria de li-ion de modo a garantir o seu funcionamento sem a necessidade de cabos de alimentação. Devido ao seu consumo mínimo, os sensores não precisam necessariamente de ligação constante à rede elétrica. Já os atuadores terão um gasto de energia um pouco superior pois necessitam de estar sempre à escuta de mensagens vindas da *gateway* no entanto não existe a necessidade de equipar os atuadores com baterias já que têm à sua disposição a rede elétrica de 220 VAC. Os atuadores utilizarão os 220 VAC para o seu funcionamento e assim não é necessário proceder à recarga destes dispositivos.

O sensor binário necessita para o seu funcionamento de uma alimentação de 5 V, esta tensão será dirigida para um circuito de carregamento da bateria, que funciona também como circuito protetor de subtensão, sobretensão e curto-circuito. A tensão da bateria seguirá para um conversor CC/CC Buck que transformará a tensão da bateria em 3,3 V. Esses 3,3 V servirão para a alimentação do microcontrolador e também do módulo rádio. A comunicação SPI será usada entre o ATmega328P e o módulo Ra-02. O microcontrolador fará a leitura de um pino configurado como entrada que funcionará como interruptor (*input binário*). O diagrama de blocos do sensor binário é mostrado na Figura 22.

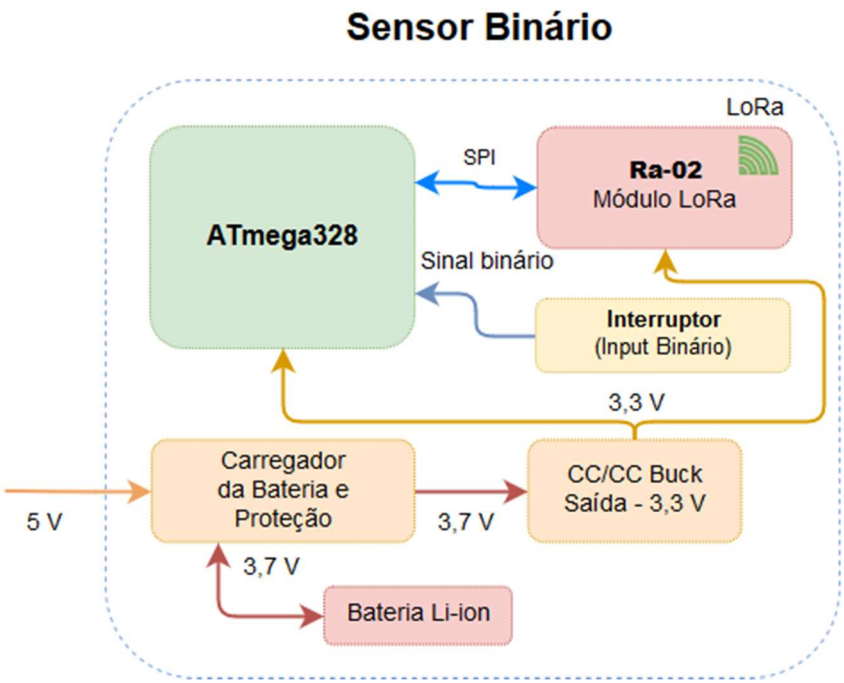


Figura 22 - Diagrama de blocos de um sensor binário

O sensor analógico é bastante semelhante em termos de *hardware* ao sensor binário, deste difere a entrada de leitura do microcontrolador, que em vez de ser de um pino configurado como entrada é de uma entrada analógica. O microcontrolador permite a leitura direta de qualquer sensor que tenha uma saída de 0 V a 3,3 V, caso o sensor tenha uma saída diferente é necessário a utilização de um circuito de condicionamento de sinal que tornará a saída do sensor legível para o microcontrolador. Na Figura 23 é mostrado o diagrama de blocos do sensor analógico.

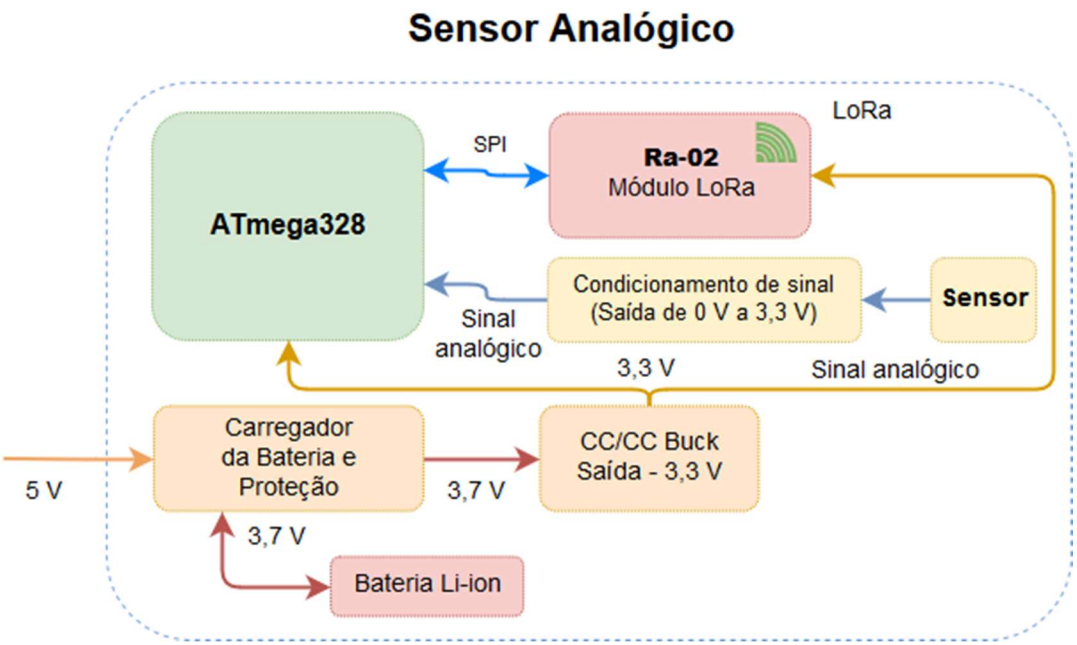


Figura 23 - Diagrama de blocos de um sensor analógico

No caso dos atuadores estes terão acesso à rede de energia elétrica de 220 VAC e desse modo podem utilizar essa tensão, depois de tratada, para a alimentação do equipamento. Sendo assim, uma derivação dos 220 VAC será direcionada para transformação, retificação e estabilização através de um conversor CA/CC que cria 5 V. Os 5 V serão usados na entrada de um conversor CC/CC Buck que os converte para 3,3 V que servirão para alimentar o microcontrolador e o módulo Ra-02 LoRa. Os 5 V servirão também para a alimentação do relé que será controlado a partir de um driver de potência. Alguns periféricos são controlados diretamente pelo ATmega328P como o interruptor e o LED, e o controlo do módulo rádio é feito por comunicação SPI.

No caso do atuador da tomada elétrica o relé, que é controlado pelo microcontrolador através do driver de potência, funciona como interruptor na fase dos 220 VAC cortando e restaurando o fornecimento de energia à tomada elétrica. O *input* binário trata-se de um interruptor que permite ligar ou desligar a tomada no local sem interferência do sistema. O diagrama de blocos do atuador de tomada está representado na Figura 24.

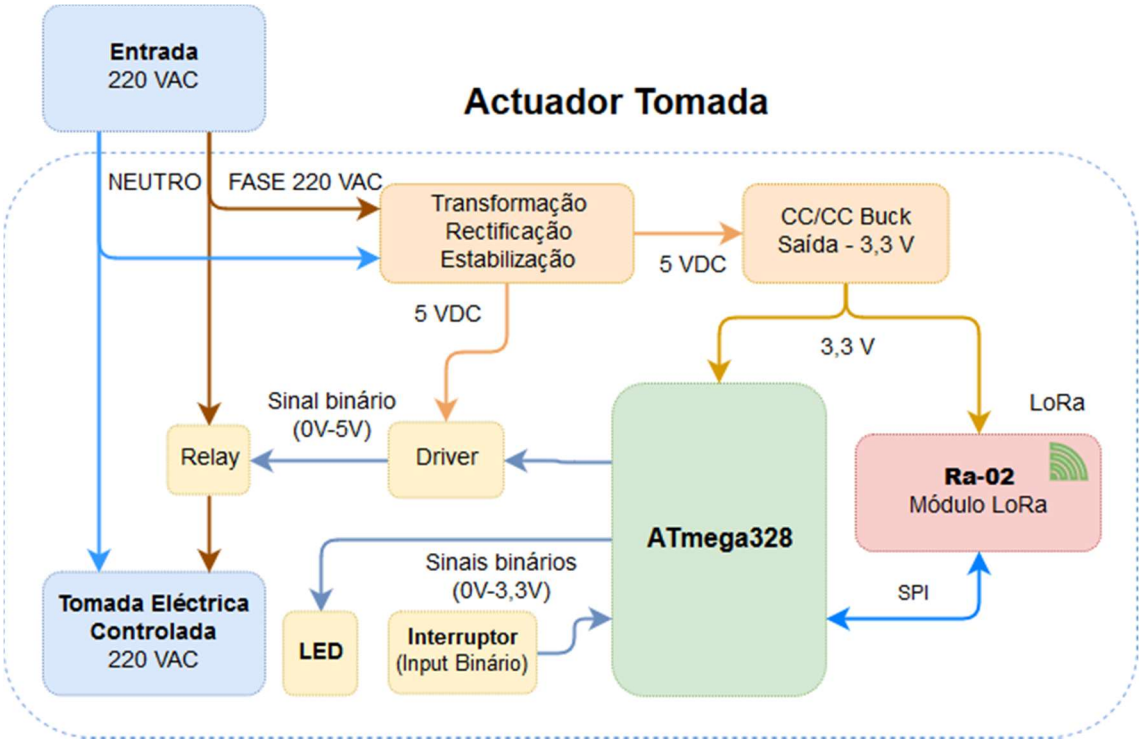


Figura 24 - Diagrama de blocos de um atuador de tomada

No caso do controlo de iluminação é usado o atuador de iluminação que tem um funcionamento muito semelhante ao atuador de tomada. A diferença encontra-se naquilo que o relé controla efetivamente, que neste caso será a iluminação onde o equipamento está instalado. O *input* binário permite ligar a iluminação manualmente sem a interferência do sistema como acontece no caso do atuador de tomada. A Figura 25 mostra o diagrama de blocos do atuador de iluminação.

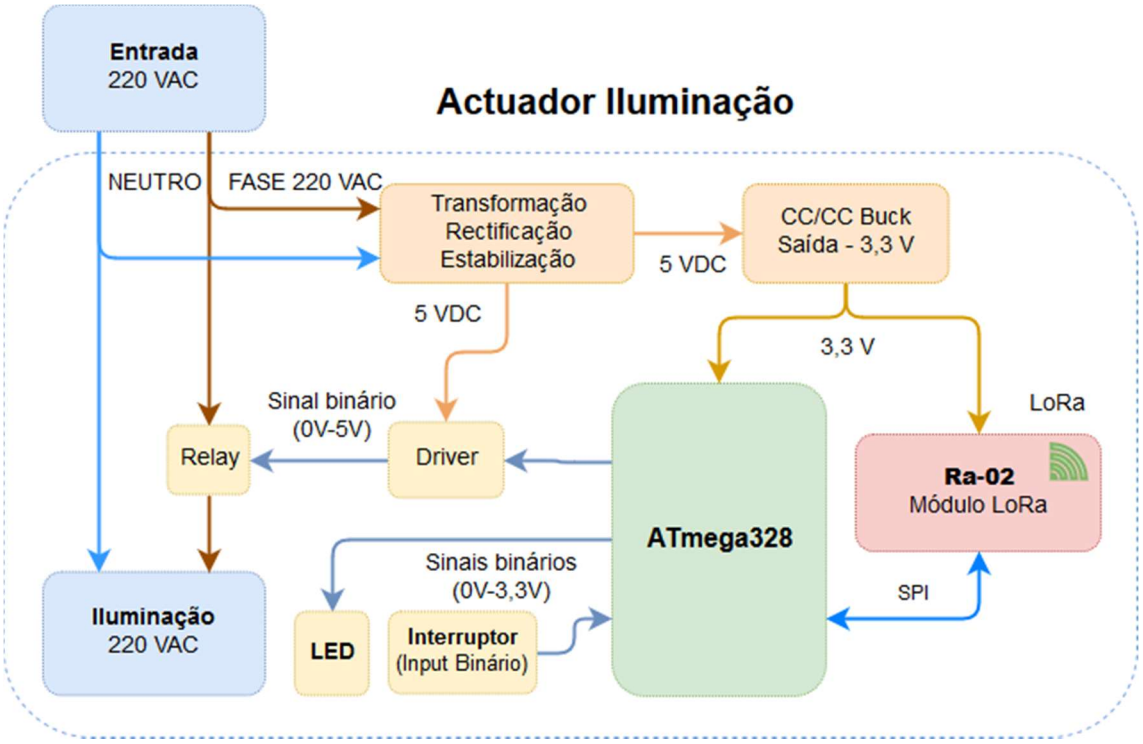


Figura 25 - Diagrama de blocos de um atuador de iluminação

3.4 Descrição do *Software*

Neste subcapítulo é explicada de uma forma geral a estrutura de funções a serem executadas pelo microcontrolador, a estrutura de comunicação, e os *softwares* de apoio ao desenvolvimento do projeto. No decorrer da pesquisa teve-se o cuidado de dar prioridade a *softwares* não proprietários com licenças livres dando preferência a programas *open source* e com licença para uso comercial.

3.4.1 Sistema Operativo

Para a preparação do sistema operativo da *gateway* será necessário começar por preparar um cartão micro SD. Como a Raspberry Pi Foundation aconselha um cartão com um tamanho mínimo de 8 GB esse será o tamanho do cartão. De seguida será necessário a formatação do cartão, para isso pode ser usado o programa SD Formatter criado pela SD Association especialmente para esta tarefa [38]. Depois de formatado o é necessário copiar o sistema operativo base para o cartão para isso foi escolhido o *software* Win32DiskImager que para além de gravar uma imagem do sistema operativo num cartão SD permite também a função inversa, criando uma cópia exata do seu conteúdo que será útil para fazer cópias de segurança (*backups*) durante o desenvolvimento do projeto [39].

O sistema operativo escolhido foi o Raspbian Stretch Lite, este é o sistema operativo oficial da Raspberry Pi Foundation, baseado em Linux, e contém todo o *software* básico necessário, incluindo um pequeno programa de configuração do sistema operativo que ajudará na realização deste projeto. A versão Lite deste sistema operativo é uma versão mais compactada que não inclui ambiente gráfico.

A *gateway* tem de estar preparada para apresentar a interface e as configurações do sistema a partir de uma rede Wi-Fi que será gerada pela mesma interface que está incluída no Raspberry Pi. Para isso ser possível é necessário a instalação de um servidor e uma plataforma de domótica. O servidor web escolhido foi o servidor livre Apache suportado pela Apache Software Foundation. É um dos servidores mais utilizados atualmente e com bastante suporte e documentação levando-o a que seja a escolha para integrar neste projeto [40]. Para o desenvolvimento das páginas Web será usado código HTML (HyperText Markup Language), CSS e Javascript utilizando a *framework open sourse* Bootstrap. O código a executar no lado do servidor, que desempenhará funções mais compostas no sistema será código PHP.

De modo a que a *gateway* possa interagir com os módulos rádio Ra-02 é necessário a utilização de uma biblioteca específica. Devido à escolha da plataforma de domótica (capitulo 3.4.2), será essencial que a biblioteca seja escrita em linguagem Python 3 pois terá de interagir diretamente com o código da plataforma de domótica também escrita em Python 3. Sendo assim, a escolha recaiu sobre a biblioteca pySX127x. Esta biblioteca desenvolvida em Python permite a comunicação do Raspberry Pi com o transmissor Semtech SX1276/7/8/9. Apesar da biblioteca precisar de algumas alterações para que o funcionamento com os módulos Ra-02 seja possível, esta foi a primeira escolha para a comunicação entre o microcomputador e os módulos rádio.

3.4.2 Home Assistant

A plataforma de domótica escolhida foi o Home Assistant. Este *software open sourse* que funciona em Python 3 preenche os requisitos definidos para o projeto, como a possibilidade de integração de outros sistemas de domótica na mesma interface, e possui uma interface intuitiva que permite a gravação, e consulta de atividades e dados históricos. Esta plataforma permite a inclusão de *software* de sistemas de domótica externos não incluídos na plataforma, que será essencial neste projeto pois envolverá a criação de um *software* de domótica específico. A grande desvantagem desta plataforma de domótica é a documentação disponível para programadores que é muito escassa e a

que existe é pouco detalhada, de qualquer forma esta foi a escolha para implementar neste projeto.

O Home Assistant permite o controlo de equipamentos através de uma interface Web que funciona em qualquer dispositivo com ligação à internet. O controlo é feito com ordens diretas do utilizador mas também a partir de automações. As automações são definidas com regras estabelecidas pelo utilizador e ativadas sempre que um evento acontece no sistema. É possível ainda definir condições para que uma determinada automação seja ativada, como por exemplo estado de um equipamento, e a automação pode controlar um ou mais dispositivos. As configurações ao Home Assistant são feitas a partir da modificação do ficheiro chamado "configuration.yaml" incluindo todas as configurações relacionadas com as plataformas e dispositivos, e também aspetos visuais da interface Web, automações e cenários.

O Home Assistant é um sistema baseado em eventos, ou seja, só existem alterações no sistema se ocorrer um evento. O Event Bus é o núcleo do funcionamento do Home Assistant, este pode escutar qualquer evento e também disparar qualquer evento. O State Machine guarda o estado dos dispositivos no sistema (entities), sempre que o estado de um dispositivo muda o State Machine dispara um evento que é escutado pelo Event Bus. O Service Registry permite aos componentes registar serviços para que estes sejam disponibilizados no sistema e permitir ao Event Bus chamar esses serviços, por exemplo o serviço "ligar a lâmpada" é registado pelo componente "light" e permite ao Event Bus chamar o serviço sempre que o utilizador precise de ligar uma lâmpada. Assim que o serviço é executado, o Service Registry dispara um evento que é ouvido pelo Event Bus, que indica que o serviço foi executado com sucesso. O Timer tem a função de disparar um evento a cada segundo, isto permite atualizar o sistema sempre que a hora muda. Os componentes são representações dos dispositivos dentro do Home Assistant, por exemplo o componente "Light" tem a função de ler o estado de uma lâmpada para que de seguida seja passado ao State Machine [41]. O Home Assistant tem já incluído no *software* os componentes para controlo dos equipamentos mais comuns em domótica mas permite a inclusão de componentes específicos. A arquitetura central do Home assistant pode ser consultada na Figura 26.

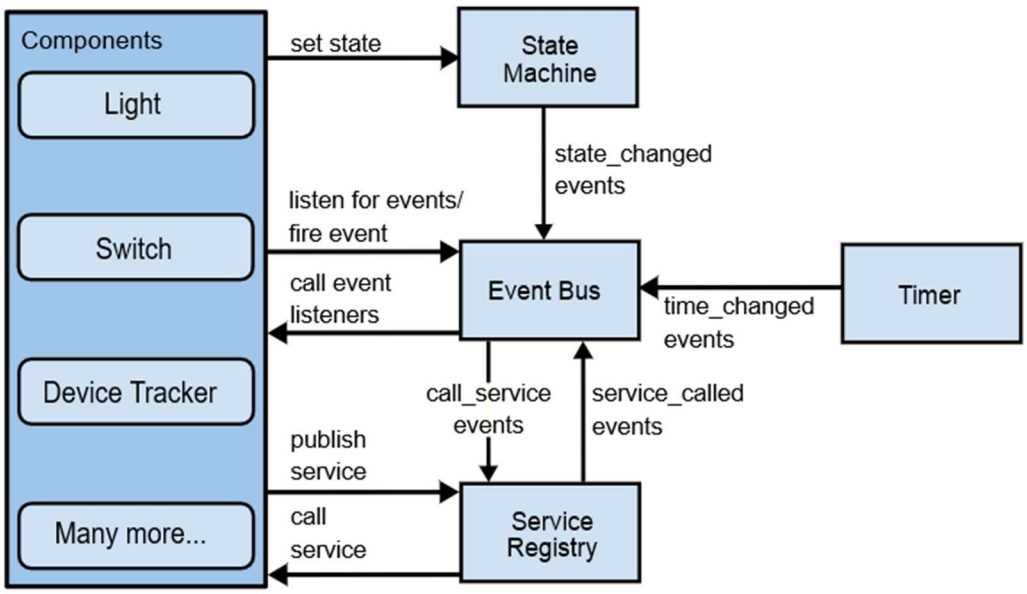


Figura 26 - Arquitetura central do Home Assistant -Imagem modificada [42]

Para que o Home Assistant funcione com os dispositivos do sistema AssistLora é necessário criar uma plataforma específica que possa comunicar com os módulos rádio através da biblioteca Python (pySX127x) e também criar componentes que serão a representação dos atuadores e sensores do sistema AssistLora. O Home Assistant permite que as plataformas e os componentes não tenham qualquer interação direta entre si, deste modo é possível ter no *software* várias plataformas de domótica que utilizam protocolos diferentes a funcionar em conjunto. Isto é possível devido ao funcionamento do Home Assistant que permite que cada componente seja executado em isolamento sem saber da existência de outros componentes no sistema. Um exemplo de funcionamento do Home Assistant é a ativação de uma lâmpada sempre que seja detetado movimento num sensor próprio. A lâmpada pode ser controlada pelo sistema AssistLora através do protocolo sem fios LoRa e o sensor de movimento através da rede sem fios Z-Wave. Assim que é detetado movimento no sensor, a plataforma Z-Wave toma conhecimento e transmite ao componente "Binary Sensor" que dispara um evento que é escutado pelo Event Bus. A automação é ativada de seguida, o Event Bus dispara outro evento que é lido pelo componente Automation que recolhe-se o evento e dispara outro, desta vez um evento que indica que é necessário ativar a lâmpada. O Event Bus escuta esse evento e chama o serviço "ligar lâmpada" ao Service Registry que por sua vez comunica com o componente Light que indica à plataforma AssistLora que tem de ligar a lâmpada. A função da plataforma AssistLora é o controlo absoluto da lâmpada através do protocolo de comunicação LoRa. Na Figura 27 é mostrado o funcionamento da automação no Home Assistant utilizando blocos.

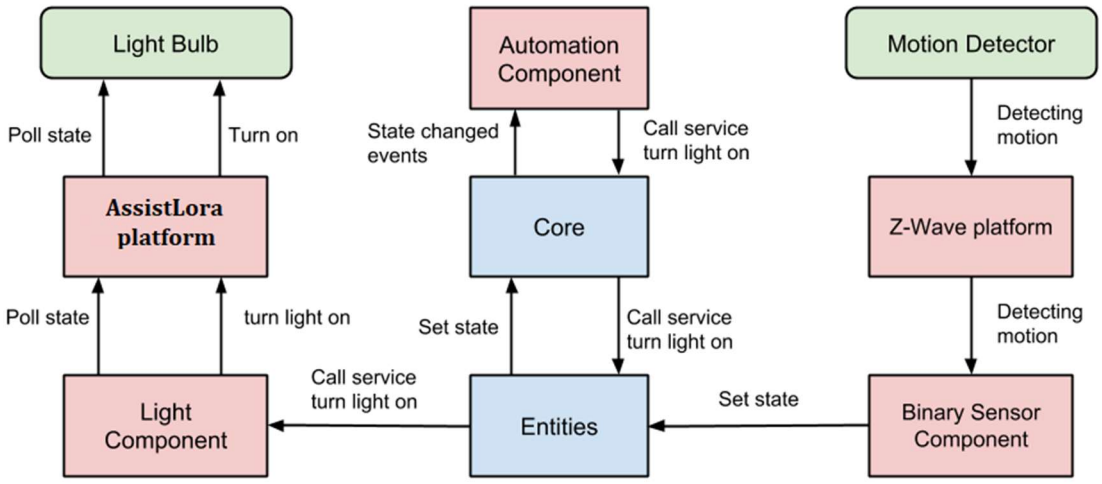


Figura 27 - Interação entre diferentes componentes no Home Assistant num exemplo de automação – Imagem modificada [42]

Cada componente inclui as funções necessárias para o seu funcionamento de modo a executar o que é esperado, no componente "Light", por exemplo, está incluído a função "turn_on" que já não faz parte do componente "Sensor". De modo a utilizar a formatação dos componentes já incluídos no *software* é necessário criar novos componentes utilizando as classes disponíveis pelo Home Assistant, como a classe "Light" ou "Sensor" para ser possível aproveitar as funções respetivas. Assim a representação dos dispositivos AssistLora será baseada nos componentes predefinidos incluídos no Home Assistant, os componentes a utilizar são o "Switch" para o atuador de tomada, o "Light" para o atuador de iluminação, o "Sensor" para o sensor analógico, e o "Binary Sensor" para o sensor binário. A plataforma a ser criada terá a função do controlo dos componentes AssistLora utilizando para isso a biblioteca Python (pySX127x) que fará a interligação entre o módulo Ra-02 através do GPIO do Raspberry Pi e o Home Assistant, possibilitando assim o controlo dos equipamentos a partir do protocolo LoRa. Na Figura 28 está uma representação da integração do Home Assistant com o sistema AssistLora no projeto.

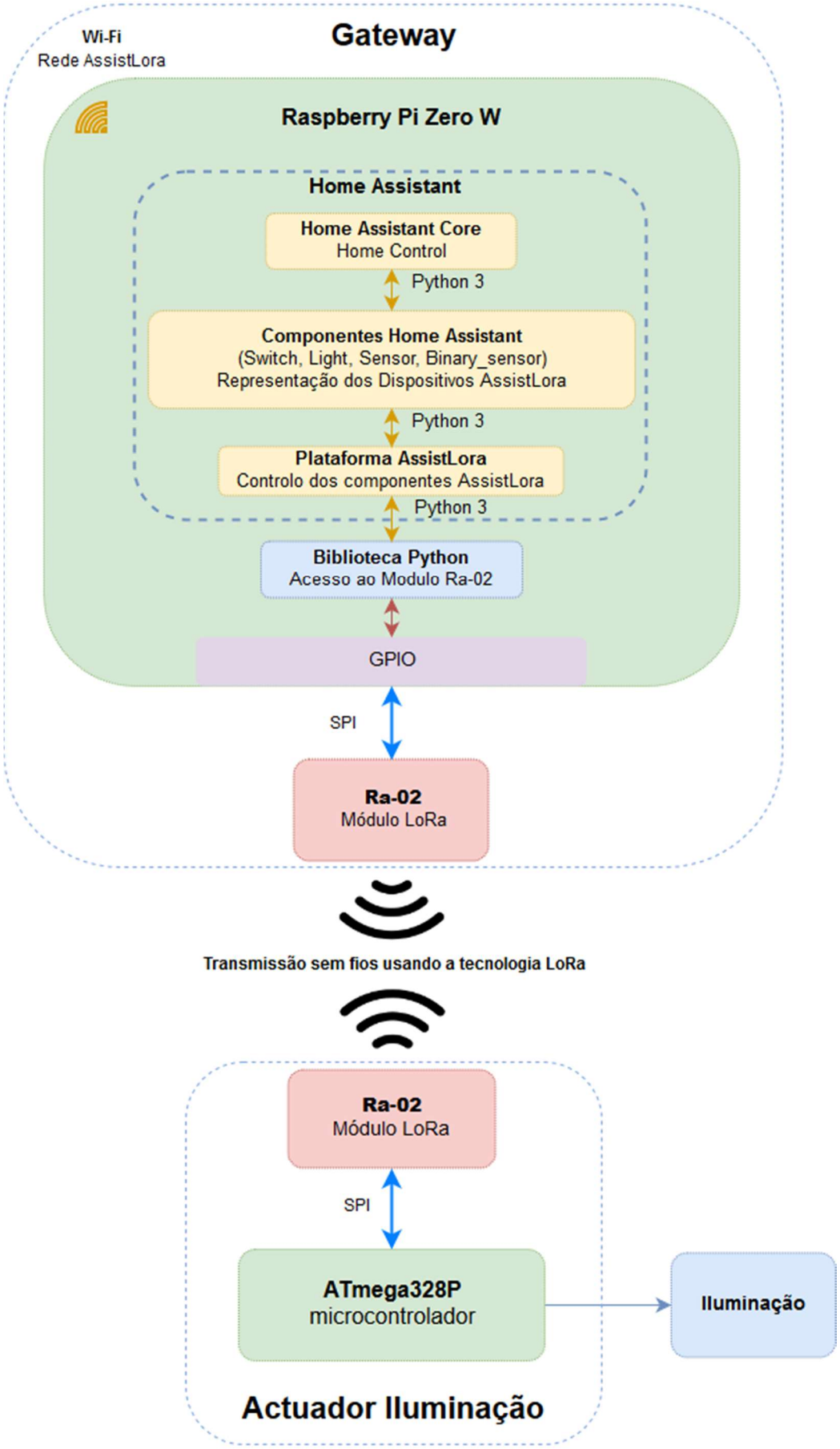


Figura 28 - Integração e interação do Home Assistant no sistema AssistLora

3.4.3 Envio de mensagens

O pacote a ser enviado terá de conter toda a informação necessária para o bom funcionamento do sistema, de modo a que seja possível identificar no pacote o seu transmissor, o recetor, o tamanho da mensagem e a mensagem. Assim definiu-se uma formatação do pacote a enviar que permitisse a identificação de vários parâmetros essenciais, a partir da localização que estes ocupam na mensagem. Estabeleceu-se que um pacote transmitido irá ter 16 bytes com 10 bytes reservados para a mensagem e 6 bytes para a restante informação necessária a transmitir. O primeiro byte do pacote é reservado para o identificador do recetor da mensagem, este byte será um número de 0 a 4 que representa um tipo de dispositivo de domótica do sistema AssistLora. O "0" é o identificador da *gateway*, o "1" é o atuador de tomada, o "2" é o sensor binário, o "3" é o sensor analógico, e o "4" é o identificador do atuador de iluminação. Os três bytes seguintes (segundo, terceiro e quarto bytes) serão o identificador do equipamento em numerário, chamado de ID. Este identificador está dividido em equipamentos de médio alcance (de 000 a 099) e de longo alcance (de 100 a 999) e apenas poderá haver um ID por equipamento. O quinto byte diz respeito ao identificador do transmissor da mensagem, e é dividido da mesma fora que o primeiro byte (identificador do recetor). O sexto byte será um número de 0 a 9 e indicará o tamanho da mensagem enviada em bytes, o valor "0" indica que é usado o tamanho máximo da mensagem (10 bytes). Os dez bytes seguintes estão reservados para a mensagem a enviar. Um exemplo de um pacote enviado poderia ser "104503OFF " este pacote será enviado para um atuador de tomada de médio alcance com o ID 45 pela *gateway*, e enviará uma mensagem com um tamanho de 3 bytes "OFF" indicando ao equipamento para desligar a tomada elétrica. Na Figura 29 está uma representação do pacote transmitido no sistema AssistLora.

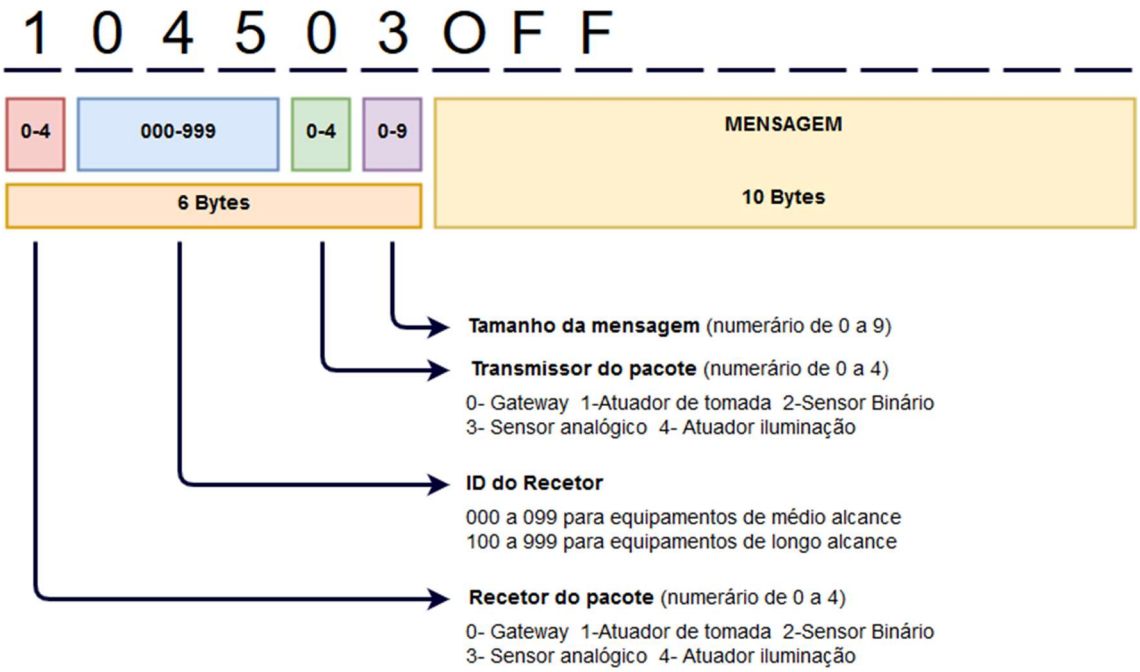


Figura 29 - Formatação de um pacote transmitido no sistema AssistLora

O funcionamento que terá o microcontrolador dependerá do tipo de equipamento, se é uma atuador ou sensor. No caso do atuador o microcontrolador terá de manter o módulo rádio sempre em escuta de modo que possa receber ordens da *gateway*, o atuador apenas enviará mensagens de reconhecimento ACK (*acknowledgement*) quando receber uma mensagem da *gateway* ou enviará para a *gateway* um pacote caso haja alteração de estado no atuador (alteração manual). O esquema de transmissão de dados do atuador está demostrado na Figura 30.

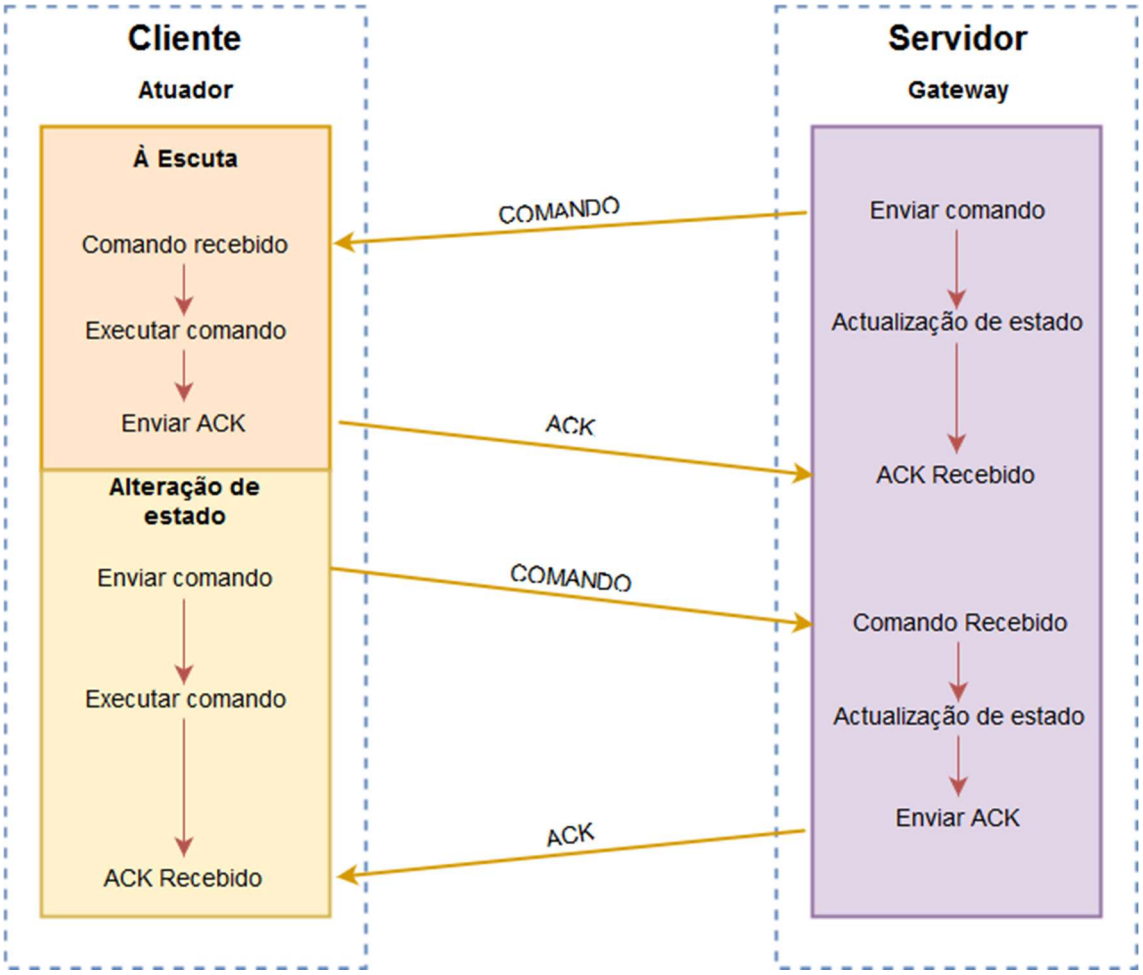


Figura 30 - Esquema de comunicação entre o cliente (atuador) e o servidor (gateway)

Já no caso de um sensor, este não necessita de ficar sempre à escuta pois a sua função é apenas o envio de dados para a *gateway*, esperando de seguida por confirmação da receção da *gateway* desses mesmos dados (ACK). A *gateway* recebe os dados, faz a atualização do sistema e de seguida envia o pacote de confirmação (ACK). O esquema da transmissão de dados do sensor está representado na Figura 31.

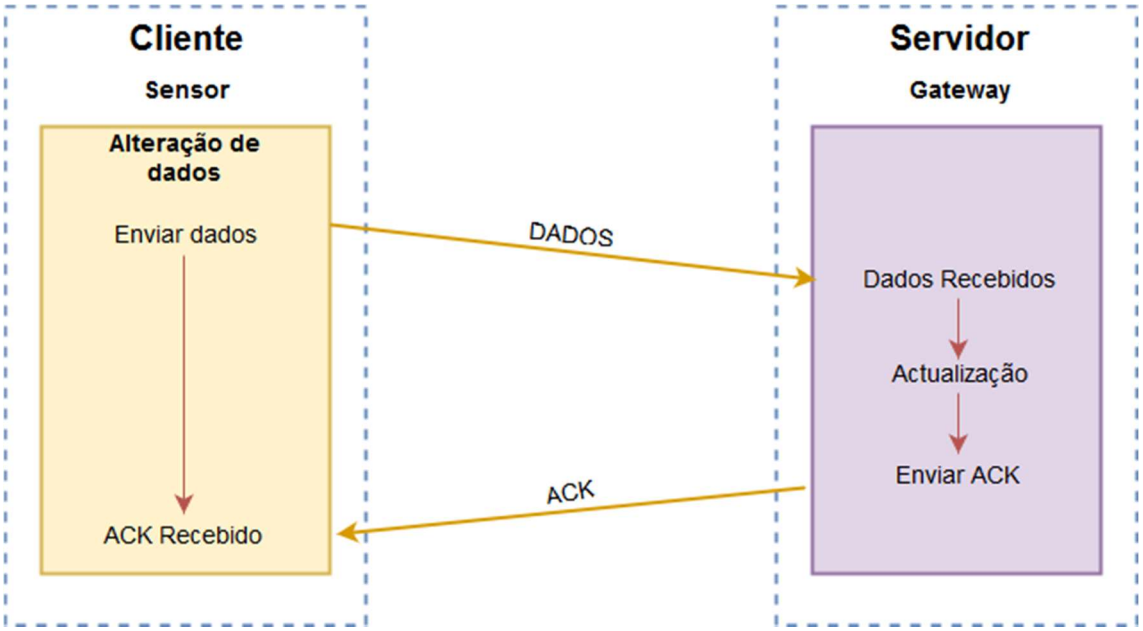


Figura 31 - Esquema de comunicação entre o cliente (sensor) e o servidor (gateway)

A verificação de erros durante a transmissão será realizada através de um temporizador. Caso o envio não seja bem-sucedido e o dispositivo não tenha recebido a mensagem ACK durante um determinado tempo predefinido, então é considerado que houve uma perda de pacote durante a transmissão ou o equipamento está inoperacional, ou fora de alcance ou ocupado. Nesse caso o dispositivo pode enviar novamente a mensagem ou dar informação de erro ao utilizador. Os sensores e atuadores estão preparados para enviar até três pacotes sem receber confirmação por parte da *gateway*. Se no terceiro pacote enviado não houver receção do ACK enviado pela *gateway* o dispositivo mostra o sinal de erro ao utilizador e volta a tentar enviar um novo pacote ao fim de um minuto. O caso mais grave de erro no sistema ocorre quando apenas o pacote ACK não é recebido pelo dispositivo, depois de a *gateway* ter feito a atualização da interface, ou seja, o dispositivo considera que a *gateway* não fez a atualização e mostrará ao utilizador o sinal de erro. No entanto esta situação é muito rara pois só ocorre quando existe perda de pacote durante a transmissão e é reduzida enviando três vezes o pacote para a *gateway*, bastando apenas uma transmissão bem sucedida para que o sistema continue o seu funcionamento normal. Na Figura 32 é mostrado o pior caso de erro que pode acontecer no sistema.

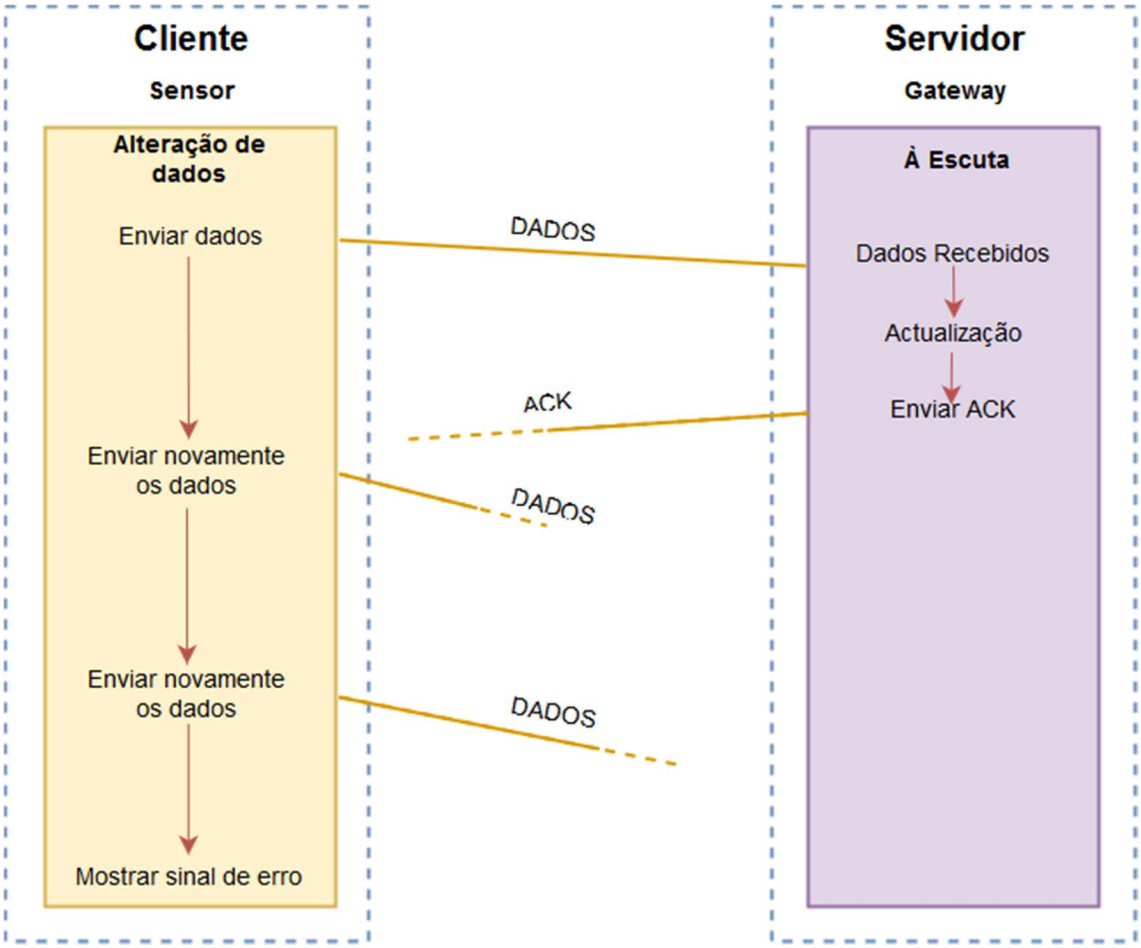


Figura 32 - Situações de erro na transmissão de dados

3.4.4 Microcontrolador

Os sensores e atuadores utilizam o ATmega328P como unidade de comando, este microcontrolador pode ser programado através de várias formas sendo a mais comum a utilização do programador AVR-ISP. No entanto é possível fazer o *upload* de código sem a necessidade de um programador especial caso seja utilizado um *frimware* próprio, chamado de *bootloader*. Essa será a escolha neste projeto, nomeadamente o *bootloader* Arduino, que apesar de reduzir o espaço de memória disponível, permite o envio do código para o ATmega328P a partir de uma porta USB com recurso a um *FTDI programmer* (Future Technology Devices International) que faz a conversão da comunicação entre UART e USB. Além disso a utilização do *bootloader* Arduino permite a utilização de várias bibliotecas que estão disponíveis para o *software* tornando a programação mais simples e rápida. De modo a poder comunicar com o módulo Ra-02 será utilizada a biblioteca RadioHead, uma biblioteca para microcontroladores que reúne vários drivers de controlo para inúmeros módulos rádio. O driver necessário para a comunicação com o módulo Ra-02 é o RH_RF95 que suporta a comunicação LoRa nos

módulos baseados no Semtech SX1276/77/78/79 assim como os módulos Modtronix Air4 e inAir9, e também HopeRF RFM95/96/97/98. Esta biblioteca funciona em várias plataformas desde que seja usado o *bootloader* Arduino [43]. A utilização do driver RH_RF95 não garante a encriptação dos dados transmitidos, sendo esse um dos requisitos do projeto foi feita uma pesquisa de modo a encontrar forma de o fazer por outros meios. A encriptação das mensagens tem de ser feita antes da transmissão utilizando uma encriptação segura, como a encriptação AES (Advanced Encryption Standard) para isso será utilizada a biblioteca AESLib. Esta biblioteca permite a encriptação de dados de 16 bytes utilizando uma chave de 128 bits e foi a escolha para utilizar neste projeto [44].

Os dispositivos do sistema, atuadores e sensores, tem a função da execução de ordens, e o envio de dados para a *gateway*. No caso dos atuadores, o dispositivo executa ordens assim que as recebe da *gateway* ou manualmente através de um botão no dispositivo. No caso dos sensores estes enviam os dados para a *gateway* apenas quando existem dados novos para enviar e não recebem ordens da *gateway*.

Assim que um sensor do sistema é ligado este começa por fazer as inicializações de variáveis e a configuração do módulo rádio. De seguida o microcontrolador entra num ciclo infinito em que apenas sairá se o dispositivo for reiniciado, e faz uma leitura do sensor (leitura analógica ou digital dependendo do tipo de sensor). Se a leitura for diferente da anterior então a *gateway* precisa de ser avisada, para isso o sensor envia um pacote para a *gateway* com os dados atualizados e depois espera pela confirmação de receção (ACK) que será enviada pela *gateway*. Depois do sensor ter recebido o ACK retoma o funcionamento fazendo uma nova leitura. De modo a poupar energia é possível colocar o microcontrolador a dormir durante um tempo predefinido antes de cada leitura. O fluxograma geral do funcionamento do código de um sensor é mostrado na Figura 33.

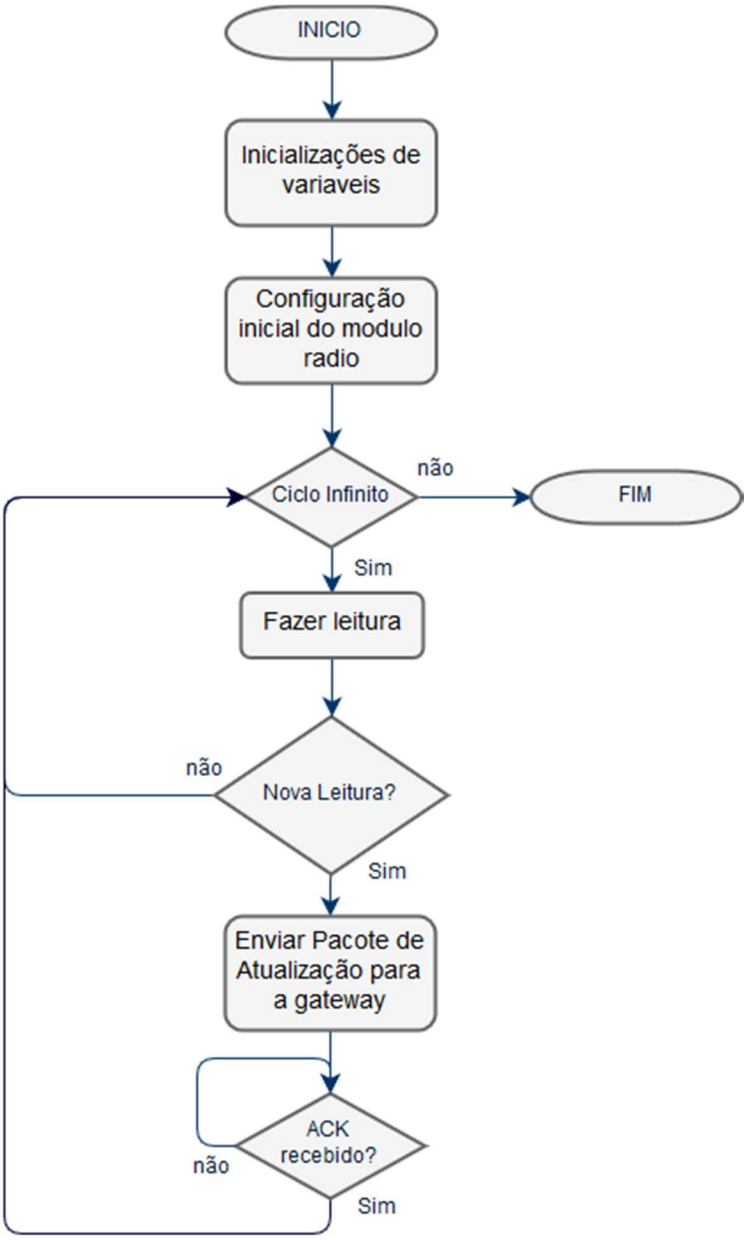


Figura 33 - Fluxograma geral do código a executar no microcontrolador de um sensor

O código executado por um atuador do sistema tem as mesmas inicializações que um sensor e um ciclo infinito, no entanto um atuador tem de estar sempre à escuta por pacotes enviados pela *gateway*, deste modo não é possível colocar o microcontrolador a dormir para poupar energia. Dentro do ciclo, o microcontrolador verificará se o botão manual está a ser premido e caso esteja o atuador irá mudar o seu estado, ou seja, no caso do atuador de iluminação se a lâmpada a controlar estiver desligada este vai liga-la e vice-versa. De seguida é enviado um pacote de atualização para a *gateway* indicando que houve uma atualização manual de estado no atuador esperando de seguida pela confirmação da receção do pacote (ACK) por parte da *gateway*. Como foi já mencionado, o atuador está sempre à escuta de novas ordens enviadas pela *gateway* e caso receba uma

o microcontrolador irá executar a ordem recebida e de seguida enviará um ACK para a *gateway* indicando que recebeu e executou a ordem enviada. O fluxograma do funcionamento de um atuador é mostrado na Figura 34.

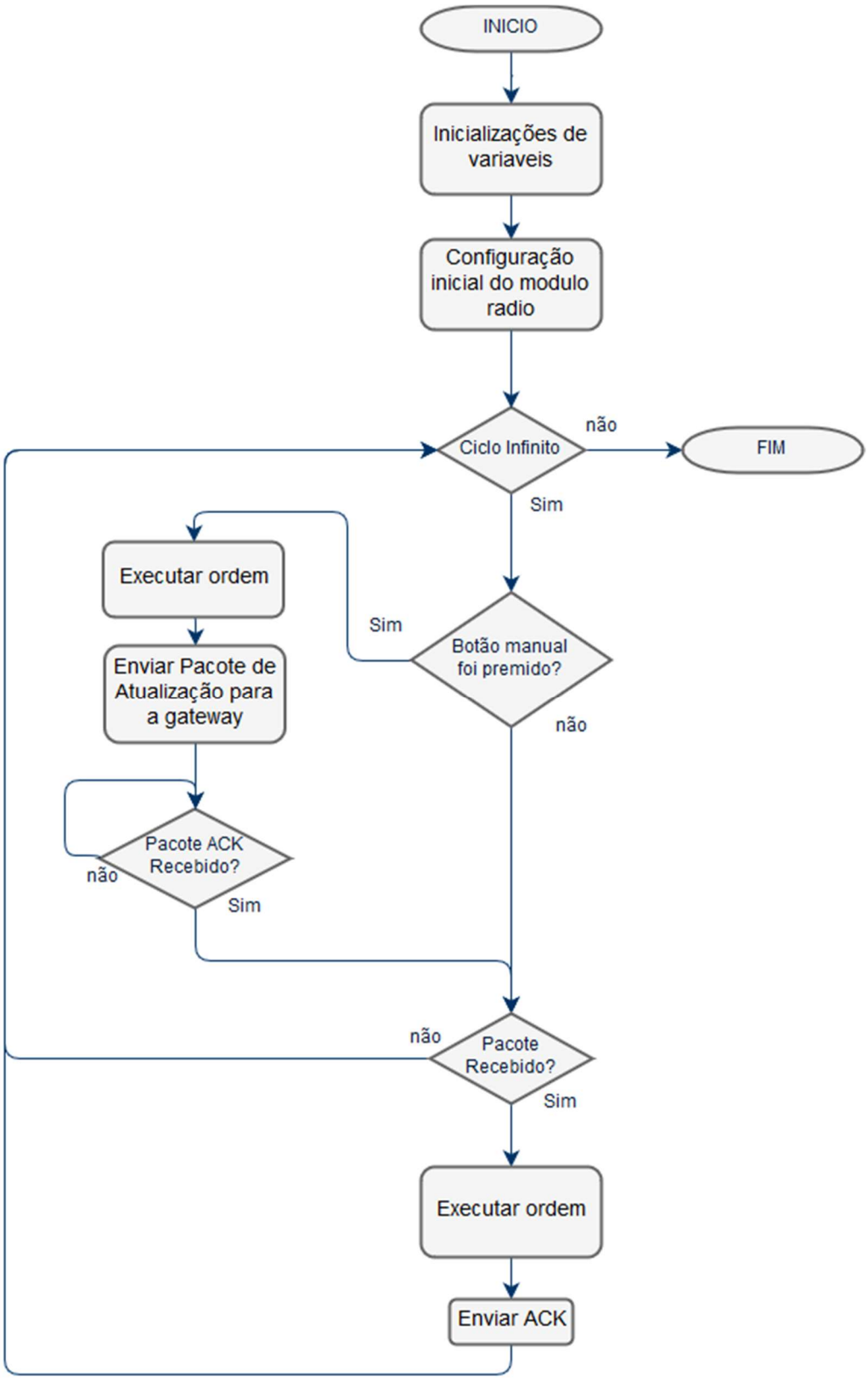


Figura 34 - Fluxograma geral do código a executar no microcontrolador de um atuador

3.5 Vetores de teste

Neste subcapítulo serão enumerado os procedimentos que permitirão testar os vários constituintes do projeto. A primeira tarefa será testar cada componente de *hardware* seguindo-se um teste a cada elemento que faz parte do *software* do sistema.

Para verificar o correto funcionamento do microcomputador e do sistema operativo é possível formatar um cartão micro SD com a imagem Raspbian e de seguida ligar o Raspberry Pi a um ecrã, através de um adaptador micro HDMI, e a uma fonte de alimentação através de um cabo micro USB. É esperado que o microcomputador arranque automaticamente e que não apresente erros na inicialização do sistema operativo, se isso ocorrer considera-se o teste validado.

De seguida, para testar o funcionamento do microcontrolador e do módulo Arduino Pro Mini, é alimentado com uma tensão regulada de 3,3 V. Por predefinição o Arduino Pro Mini é vendido já com o *bootloader* Arduino e testado com um pequeno programa que tem a função de piscar o LED do módulo em intervalos definidos. Logo após a alimentação do módulo o LED deve começar a piscar mas caso isso não aconteça é necessário verificar se o módulo não está danificado, para isso é possível enviar um código para o microcontrolador para ver se este o executa. Presumindo que o módulo já terá o *bootloader* Arduino instalado utiliza-se um FTDI *programmer* e o programa Arduino IDE e seleciona-se o módulo Arduino Pro Mini e a porta USB respetiva. De seguida envia-se um pequeno código de teste para verificar o funcionamento do microcontrolador, caso este não responda ou se ocorrer um erro no envio é provável que não esteja instalado o *bootloader* Arduino no módulo. É necessário, então, instalar o *bootloader* que pode ser instalado a partir de um AVR-ISP *programmer*, caso o microcontrolador não responda à instalação então podemos considerar que o microcontrolador está danificado.

Segue-se o teste do módulo rádio Ra-02. Para testar este módulo é necessário testar dois ao mesmo tempo de modo a verificar se conseguem enviar e receber mensagens. Para o teste preparam-se dois microcontroladores e envia-se um código de teste da biblioteca RadioHead para os microcontroladores, depois de feitas as ligações físicas com os módulos rádio. O teste é considerado validado se os dois módulos conseguirem comunicar entre si.

Testados os principais elementos *hardware* do projeto seguem-se os componentes de software. O funcionamento do sistema operativo da *gateway* é verificado logo no teste do

Raspberry Pi, mas não outros softwares que irão executar no microcomputador. Testar a funcionalidade do servidor depois da instalação é relativamente simples já que o servidor Apache cria uma página de teste automaticamente durante a instalação, assim acedendo ao endereço do Raspberry Pi na porta 80 deverá ser possível ver a página de teste do servidor, caso contrário ocorreu um erro durante a instalação que terá de ser repetida.

Para verificar o funcionamento do Home Assistant depois da instalação é realizado um processo semelhante ao teste do servidor, o Home Assistant funcionará no endereço do microcomputador mas na porta 8123 onde indicará se durante o processo de instalação ocorreu ou não algum erro.

No controlo dos módulos rádio pelo microcontrolador é necessário a utilização da biblioteca pySX127x, de modo a testar esta biblioteca é possível utilizar alguns ficheiros de código disponibilizados por esta. Com esses ficheiros é possível verificar se o módulo é reconhecido pelo microcomputador assim como definir as configurações da transmissão de dados e ainda o seu envio e receção. Depois de verificar as configurações da transmissão e se o módulo rádio está bem ligado e é reconhecido pelo Raspberry Pi é possível testar o envio e recessão de dados, para isso prepara-se inicialmente outro módulo Ra-02 ligado a um microcontrolador com um programa de envio de dados com um intervalo definido e executa-se no microcomputador o programa que irá ficar à escuta de dados. Caso as mensagens sejam recebidas pelo Raspberry Pi pode-se passar ao processo inverso, o microcomputador a enviar dados e o ATmega328P a receber. Se a comunicação funcionar nos dois sentidos o teste está concluído.

4 Implementação

O protótipo criado será apresentado neste capítulo assim como todo o processo que levou ao seu desenvolvimento. De forma a iniciar e implementar o projeto foram definidos 14 passos sequenciais:

1. Pesquisa dos componentes, softwares e materiais a usar;
2. Teste individual dos vários componentes;
3. Preparação do sistema operativo da *gateway*;
4. Preparação das bibliotecas para comunicação entre o microprocessador e o microcomputador;
5. Teste de comunicação entre os módulos rádio utilizando as bibliotecas modificadas;
6. Criação dos esquemas elétricos e implementação em *breadboard*;
7. Programação do código a implementar nos microcontroladores;
8. Programação da plataforma e dos componentes AssistLora para o Home Assistant;
9. Teste da plataforma e dos componentes criados;
10. Desenvolvimento das páginas do *website* utilizando PHP e Javasript;
11. Otimização do funcionamento da *gateway*;
12. Montagem do protótipo;
13. Testes gerais e aperfeiçoamentos;
14. Elaboração do artigo.

O primeiro passo foi a pesquisa de todo o material necessário para a elaboração do projeto, seguiram-se os testes dos vários componentes segundo o descrito no capítulo 3.5. A instalação do sistema operativo Raspbian foi concluída com sucesso, seguiu-se o teste de funcionamento do módulo Arduino pro mini. Para realizar o teste alimentou-se o módulo através de uma fonte de alimentação regulada para 3,3 V no pino VCC, o LED incluído na placa começou de imediato a piscar, validando o teste. Para os testes dos módulos rádio Ra-02 foi feita a ligação elétrica de cada módulo a um microcontrolador e de seguida foi feito o *upload* de um código de teste da biblioteca RadioHead para cada microcontrolador. Estes testes decorreram sem problemas e foi possível verificar que os módulos Ra-02 funcionam como previsto recebendo e enviando mensagens.

Ligando um adaptador Wi-Fi ao Raspberry Pi por USB é possível aceder a este a partir da rede interna por SSH. Deste modo, sabendo o endereço IP do microcomputador acedeu-se a este a partir do comando mostrado na Figura 35.

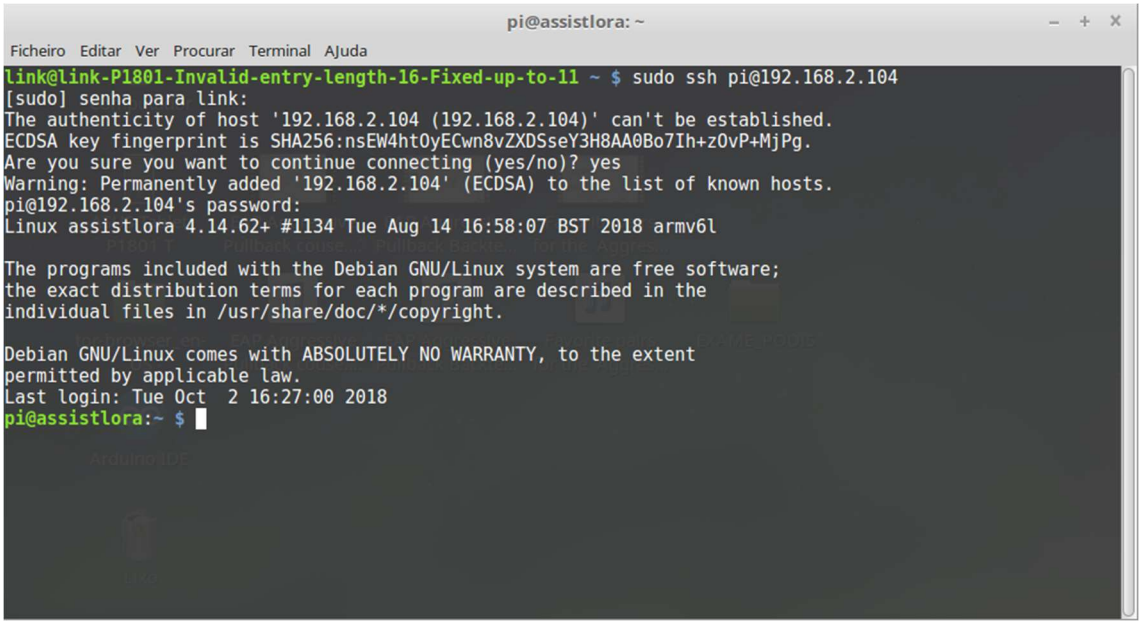


Figura 35 - Acesso por SSH ao Raspberry PI

Depois de instalados todos os *softwares* no microcomputador foram realizados os testes, necessários ao projeto. Para verificar o funcionamento do servidor colocou-se na barra de endereços de um navegador web o IP (Internet Protocol) do microcomputador na porta 80 que apresentou uma página predefinida pelo servidor Apache, considerou-se o teste validado. De seguida verificou-se a porta 8123 (porta predefinida da interface Home Assistant) que apresentou o ecrã inicial de boas vindas do Home Assistant indicando que não houve erros durante a instalação do *software*.

Depois de verificar o funcionamento do *hardware* e software do sistema foi feita a preparação das bibliotecas usadas. Durante a escolha da biblioteca pySX127x verificou-se que esta precisava de modificações para que o funcionamento com os módulos rádio Ra-02 fosse prefeito, no entanto, as alterações não foram necessárias para um teste inicial já que para testes os programadores da biblioteca disponibilizaram ficheiros de códigos especiais para usar nestes casos. Depois de efetuadas as ligações elétricas entre um módulo rádio e o Raspberry Pi foi executado um código que indicou que o microcomputador consegue comunicar com o módulo Ra-02. De seguida colocou-se o ATmega328P ligado ao Ra-02 a transmitir mensagens com um intervalo estabelecido e no microcomputador executou-se um código que coloca o módulo rádio à escuta de mensagens e caso as receba as mostre no terminal. No decorrer do teste percebeu-se que

o microcomputador conseguia detetar as mensagens enviadas pelo microcontrolador mas não apresentava a mensagem enviada no terminal. Presumindo tratar-se de um problema de codificação realizou-se o teste inverso, colocando desta vez o microcontrolador a receber mensagens e o microcomputador a transmitir. Este teste revelou que o microcontrolador não conseguia detetar qualquer mensagem enviada pelo Raspberry Pi. Para tentar resolver o problema foi modificado o código do microcomputador de modo a que mostrasse todos os carateres enviados pelo microcontrolador. Deste modo, verificou-se que o microcontrolador enviava mais carateres do que aqueles incluídos na mensagem a enviar causando uma errada descodificação por parte do microcomputador. Analisando melhor o código da biblioteca RadioHead verifica-se que existe um *header* colocado a cada mensagem de quatro carateres que não são documentados no entanto esses quatro carateres são lidos pelo microcomputador como sendo 0xff, 0xff, 0x00 e 0x00. De modo que a comunicação entre o ATmega328P e o Raspberry Pi seja feita sem problema esses carateres tem de ser incluídos na mensagem a enviar pelo microcomputador, caso isso não seja feito o microcontrolador descarta qualquer mensagem recebida pelo Raspberry Pi. Assim sendo foram feitas várias alterações na biblioteca pySX127x de modo a permitir a comunicação com biblioteca RadioHead e o suporte para o módulo Ra-02.

Até este momento todas as comunicações foram feitas sem qualquer segurança de dados, ou seja, qualquer dispositivo podia aceder aos dados que estavam a ser transmitidos. Sendo um dos requisitos do sistema a encriptação de dados esta tinha se ser implementada. Foi escolhido um método de encriptação em função do ATmega328P pois é este que possui o menor nível de processamento. Foi escolhida a encriptação AES e a biblioteca AESLib para implementar no microcontrolador. Depois da encriptação os dados encriptados tem de ser transformados numa variável *string* de modo a serem enviados a partir da biblioteca RadioHead para isso os dados são codificados pelo método Base64 a partir da biblioteca Arduino-Base64 [45]. No caso do microcomputador a codificação de Base64 já está incluída na programação Python mas não as funções de encriptação AES, desse modo foi utilizado o pacote Crypto.Cipher da biblioteca PyCryptodome. É utilizado o algoritmo de encriptação “Symmetric Ciphers” onde uma única chave é utilizada para encriptar e desencriptar. É utilizado na encriptação o Block Cipher AES sem nenhum modo de operação de forma a tornar compatível com o microcontrolador já que este também não utiliza nenhum modo de operação como o CTR (CounTeR mode), GCM (Galois/Counter Mode) ou CBC (Ciphertext Block Chaining)

[46]. O processo de encriptação e desencriptação de mensagens no microcontrolador e no microcomputador está representado na Figura 36.

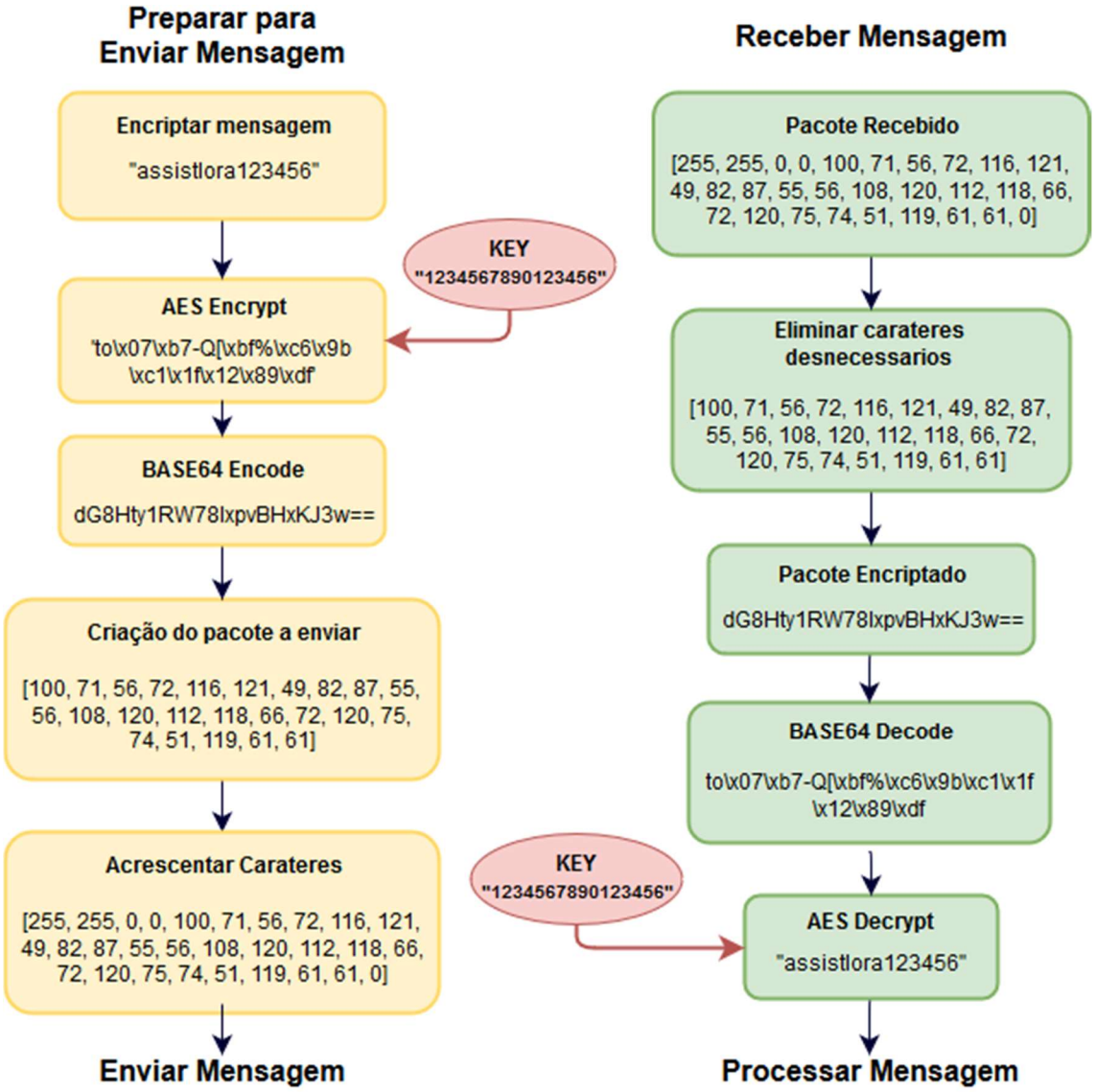


Figura 36 - Representação do processo de encriptação e desencriptação de mensagens

Todo o código do sistema criado e utilizado como o *website*, código dos atuadores e sensores e o código da plataforma e componentes AssistLora para o Home Assistant foram colocados num repositório público no GitHub e está disponível para ser consultado em www.github.com/rpsreal/assistlora.

As várias alterações feitas à biblioteca pySX127x, das quais se inclui a encriptação dos dados transmitidos e a possibilidade da utilização de dois módulos ao mesmo tempo, estão disponíveis noutro repositório público: www.github.com/rpsreal/pySX127x, que inclui também os códigos utilizados nos testes iniciais de comunicação, com e sem encriptação de dados, no microcomputador, assim como uma explicação de utilização da biblioteca

para outros programadores. De modo a utilizar a biblioteca criada no Home Assistant é necessário que a biblioteca esteja disponível no repositório PyPi (Python Package Index) assim a biblioteca foi colocada no repositório com o nome ”pyLoRa” e pode ser consultada em www.pypi.org/project/pyLoRa.

Os códigos de teste utilizados no microcontrolador estão disponíveis em www.github.com/rpsreal/LoRa_Ra-02_Arduino também com uma breve descrição do código.

Depois de realizado o teste de comunicação, entre o microcontrolador e o microcomputador, com sucesso foram criados os esquemas elétricos dos atuadores, sensores e da *gateway* e implementados em definitivo na *breadboard*.

Foi feita a programação em C++ dos sensores e atuadores de acordo com o seu modo de funcionamento utilizando o *software* Visual Studio Code [47]. Este *software* permite também enviar o código para o microcontrolador e a utilização do sistema de gerenciamento de código GIT. Na Figura 37 pode-se ver o envio com sucesso do código para o microcontrolador.

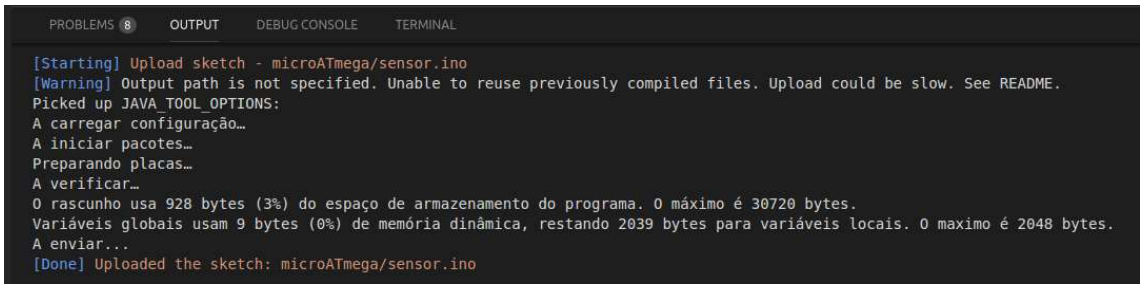


Figura 37 - Envio de código para o ATmega328P

Seguiu-se a programação da plataforma e dos componentes AssistLora para a integração no Home Assintant. Esta programação foi complicada devido à falta da documentação disponível o que levou que fosse demorada. Foi realizada em linguagem Python 3 e foi nestes ficheiros que a biblioteca pySX127x foi usada. Depois da conclusão da programação da plataforma e dos componentes, os ficheiros de código foram enviados para o microcomputador com a ajuda do cliente FTP FileZilla [48] e de seguida foram realizados os testes de funcionamento que se revelaram um sucesso.

Na programação das páginas do *website* a linguagem de código utilizada para executar no lado do servidor foi PHP e do lado do cliente Javascript. Para a formatação das páginas web é usada a *framework* Bootstrap 4. Isto permite que o *website* se adapte a qualquer dispositivo (computador, *tablet* e *smartphone*) e fique mais apresentável esteticamente.

Depois de concluída a programação os ficheiros de código foram enviados para o Raspberry Pi e testados.

Depois de todos os elementos do projeto estarem a funcionar foi feita uma otimização do sistema operativo da *gateway*. O Home Assistan foi colocado a iniciar automaticamente assim que o microcomputador ligue assim como o servidor Apache, também foram feitas alterações ao funcionamento da interface Wi-Fi, incluída no Raspberry Pi Zero W, de modo que se comportasse como um ponto de acesso. Depois destas alterações o utilizador tem acesso à interface de configuração e do Home Assistant assim que ligue a *gateway*.

Depois de se decidir o aspeto e o funcionamento do protótipo do sistema, foi feita uma pesquisa de materiais a utilizar para a sua construção. Tendo em conta as limitações financeiros e de outros recursos, os materiais usados e o acabamento da unidade apresentada estará longe de um protótipo final para uma possível futura comercialização.

O objetivo da criação do protótipo foi mostrar o funcionamento do projeto. Pretendeu-se que o protótipo fosse apresentável, compacto, e sem eletrónica visível ao utilizador mas facilmente acessível. O material escolhido para a elaboração do protótipo foi madeira contraplacado, este material foi escolhido por ser fácil de trabalhar, rígido e de baixo custo. Depois da montagem do protótipo seguiram-se os testes finais e aperfeiçoamentos e, de modo a concluir o projeto, a escrita deste artigo.

4.1 Hardware

O *hardware* do sistema encontra-se distribuído pelos sensores, atuadores e na *gateway*.

Cada sensor do sistema é composto por um microcontrolador e um módulo rádio. O ATmega328P está conectado a um oscilador de 8 MHz e um botão que terá a função de reiniciar o microcontrolador. O módulo Ra-02 comunica com o microcontrolador através de SPI utilizando os pinos SCK, MISO, MOSI e NSS. É feita a ligação do pino RESET do módulo rádio ao microcontrolador e também do pino DIO0 que permite ao ATmega328P saber quando um pacote é recebido e enviado. Dependendo do tipo de sensor (analógico ou digital) é feita a ligação ao respetivo pino do microcontrolador. A alimentação do sistema provem de um conversor CC/CC que transforma a tensão das baterias em 3,3 V de modo a alimentar o microcontrolador e o módulo Ra-02. Apresenta-se na Figura 38 o esquema elétrico de um sensor do sistema.

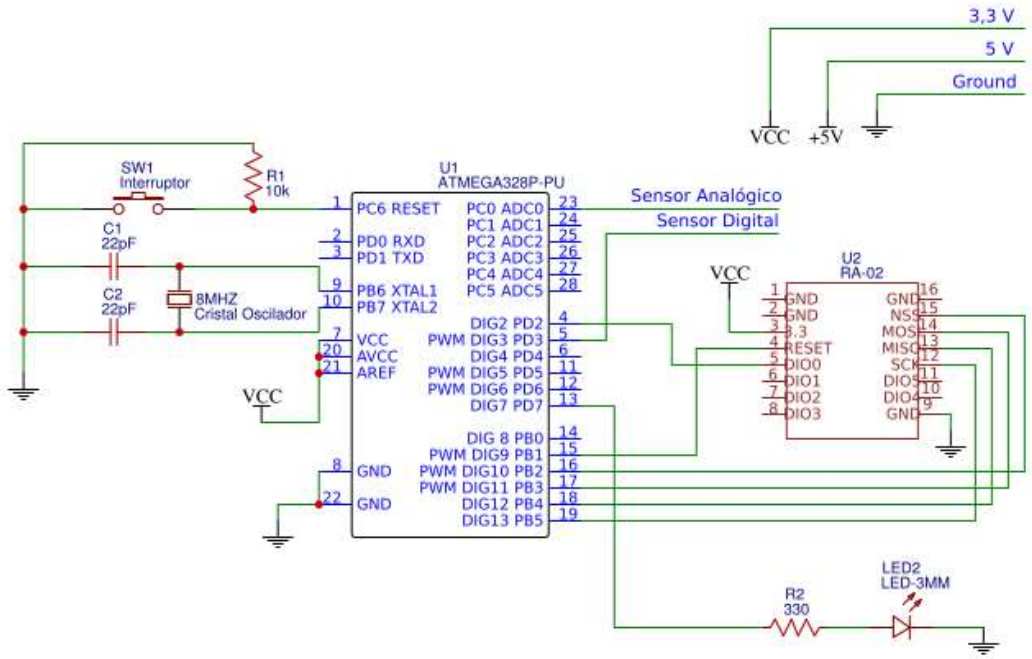


Figura 38 - Esquema elétrico do sensor

A implementação de um sensor analógico em *breadboard* é apresentada na Figura 39. Nesta implementação é utilizado um potenciómetro como sensor analógico sem qualquer circuito de condicionamento de sinal. A alimentação é feita diretamente por uma fonte de alimentação regulada para 5 V que posteriormente é ligada ao conversor abaixador apresentado também na Figura 39 que criará os 3,3 V para alimentação do microcontrolador e módulo rádio.

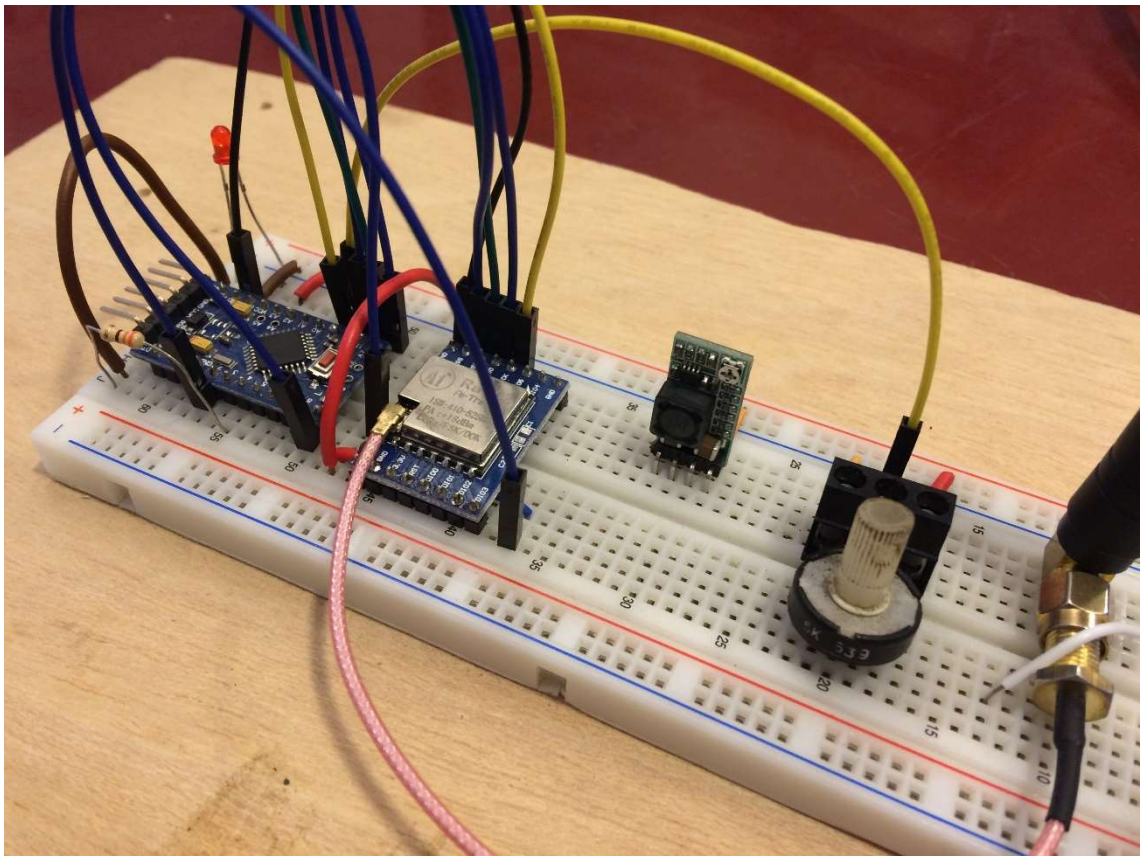


Figura 39 - Implementação do circuito de um sensor em *breadboard*

O sistema elétrico de um atuador do AssistLora é semelhante ao esquema elétrico dos sensores. O ATmega328P é ligado ao oscilador e também ao botão de *reset* assim como ao módulo rádio da mesma maneira que os sensores. Nos atuadores é utilizado um LED indicador e também um botão de alteração de estado manual, esses são ligados ao microcontrolador utilizando, no caso do LED, uma resistência de limitação de tensão de 330 Ω. Para controlo da iluminação ou da tomada é utilizado um relé que é controlado, a partir do microcontrolador, por um simples driver de potência constituído por um transístor NPN, uma resistência e um díodo. O transístor NPN BC548 permite uma corrente no coletor até 100 mA, sendo suficiente para funcionamento do relé [49]. Os terminais do relé são ligados à tomada ou à iluminação a controlar pretendida. A Figura 40 mostra o esquema elétrico dos atuadores.

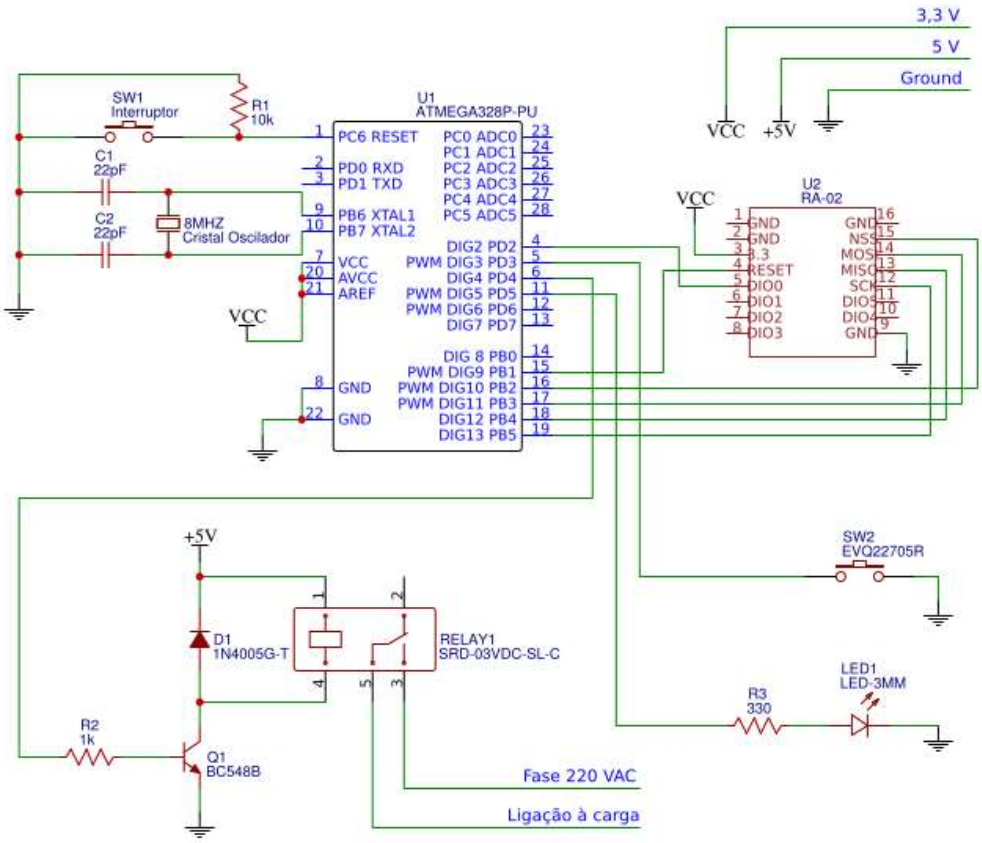


Figura 40 - Esquema elétrico do atuador

A implementação da eletrónica do atuador em *breadboard* é mostrada na Figura 41.

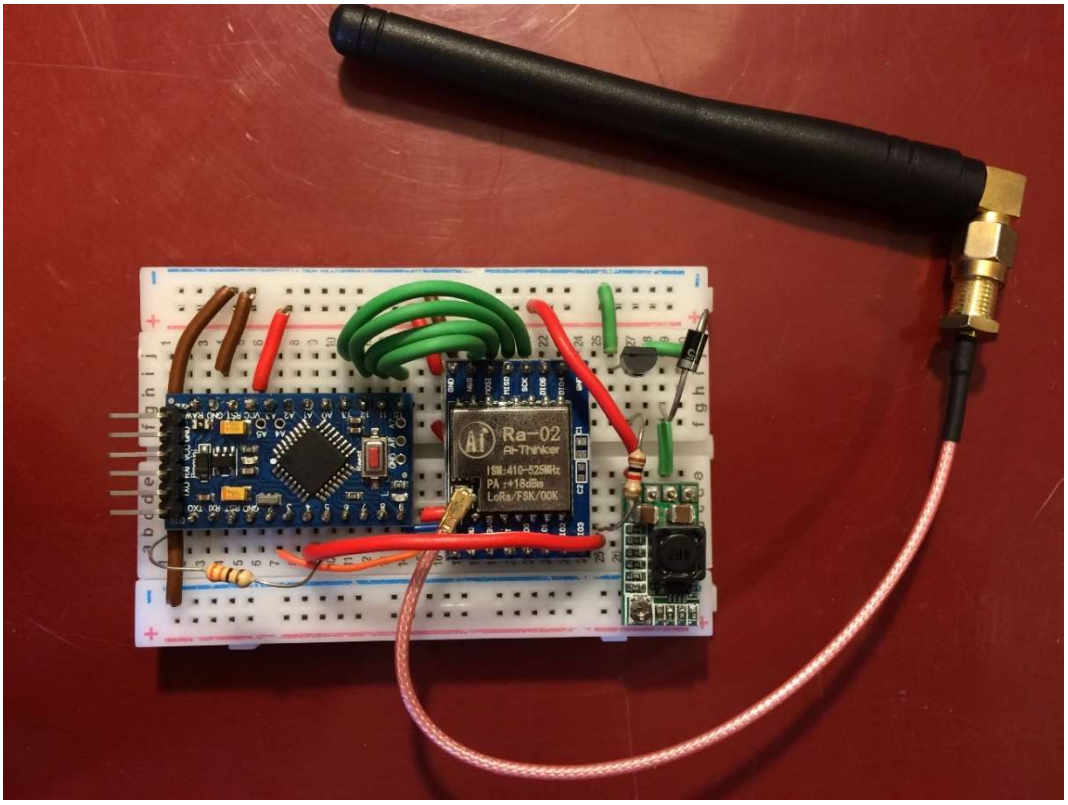


Figura 41 - Implementação do circuito de um atuador em *breadboard*

Este circuito foi implementado em *breadboard* de modo a que os periféricos fossem ligados posteriormente. O relé, a fonte de alimentação, o botão e o LED são todos ligados à *breadboard* através de cabos. A ligação aos periféricos assim como a entrada de 220 VAC pode ser visto na Figura 42.

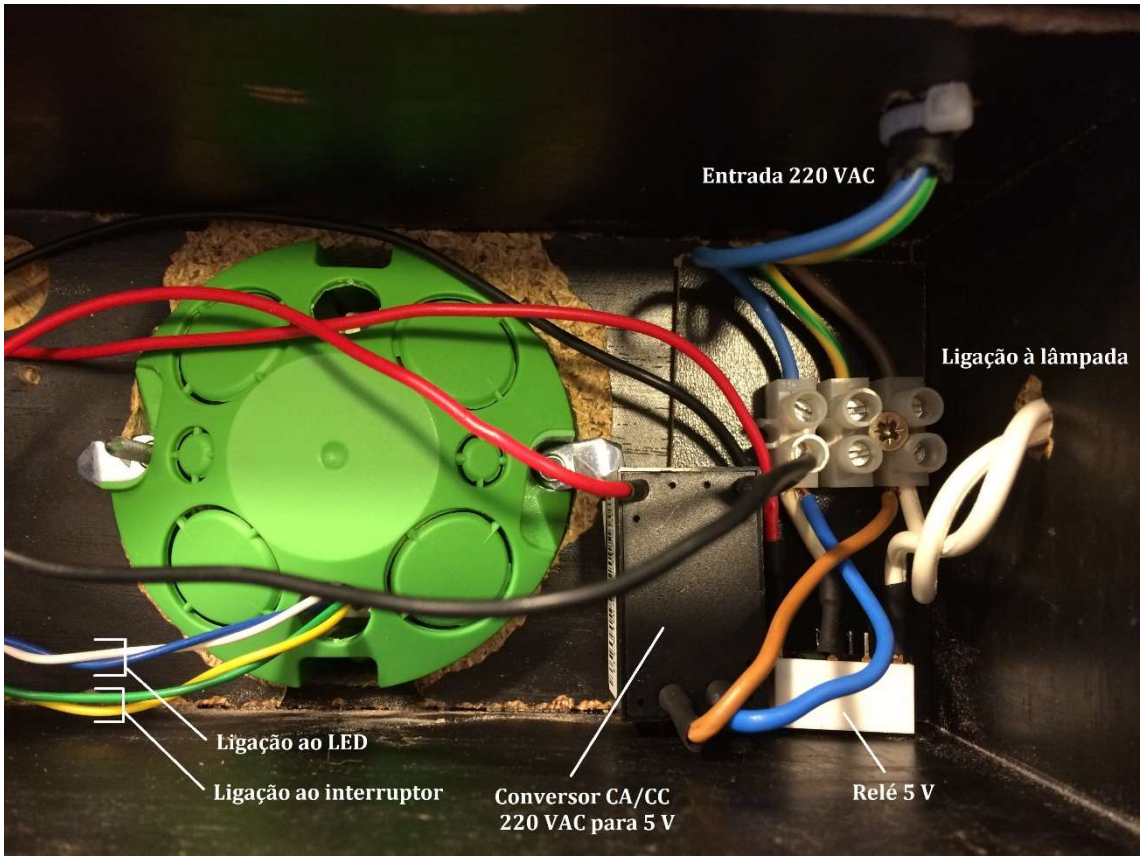


Figura 42 - Ligações dos periféricos do sistema

O esquema elétrico da *gateway* inclui o microcomputador, dois módulos rádio e dois LEDs. O microcomputador comunica com os módulos rádio a partir de comunicação SPI e de modo a identificar cada módulo rádio é utilizado um pino diferente do microcontrolador ligado ao NSS do Ra-02. A ligação aos pinos DIO0, DIO1, DIO2 e DIO3 permite o acesso do microcomputador a mais interrupções criadas pelos módulos Ra-02 que podem ser usadas pela biblioteca pySX127x. O microcontrolador tem acesso também ao pino *reset* de ambos os módulos Ra-02 e controla dois LEDs diferentes. Os módulos Ra-02 são alimentados com 3,3 V gerados pelo conversor de tensão do Raspberry Pi e o Raspberry Pi alimentado com 5 V de uma fonte externa. O circuito elétrico utilizado está representado na Figura 43.

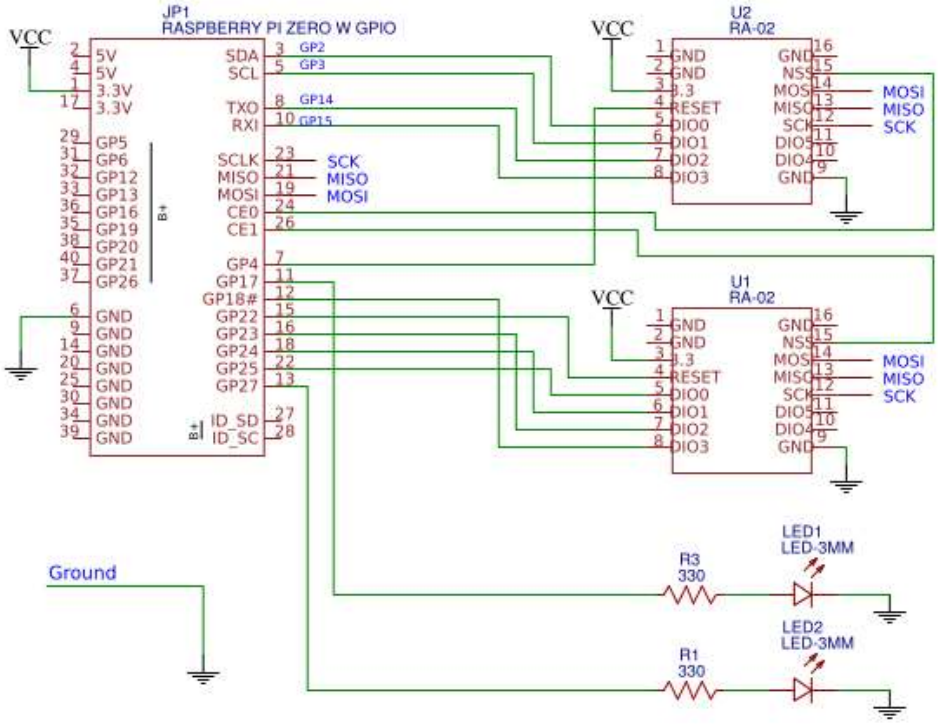


Figura 43 - Esquema elétrico da gateway

Este circuito foi posteriormente implementado numa *breadboard* para testes e para a construção do protótipo. O circuito da *gateway* implementado pode ser visto na Figura 44.

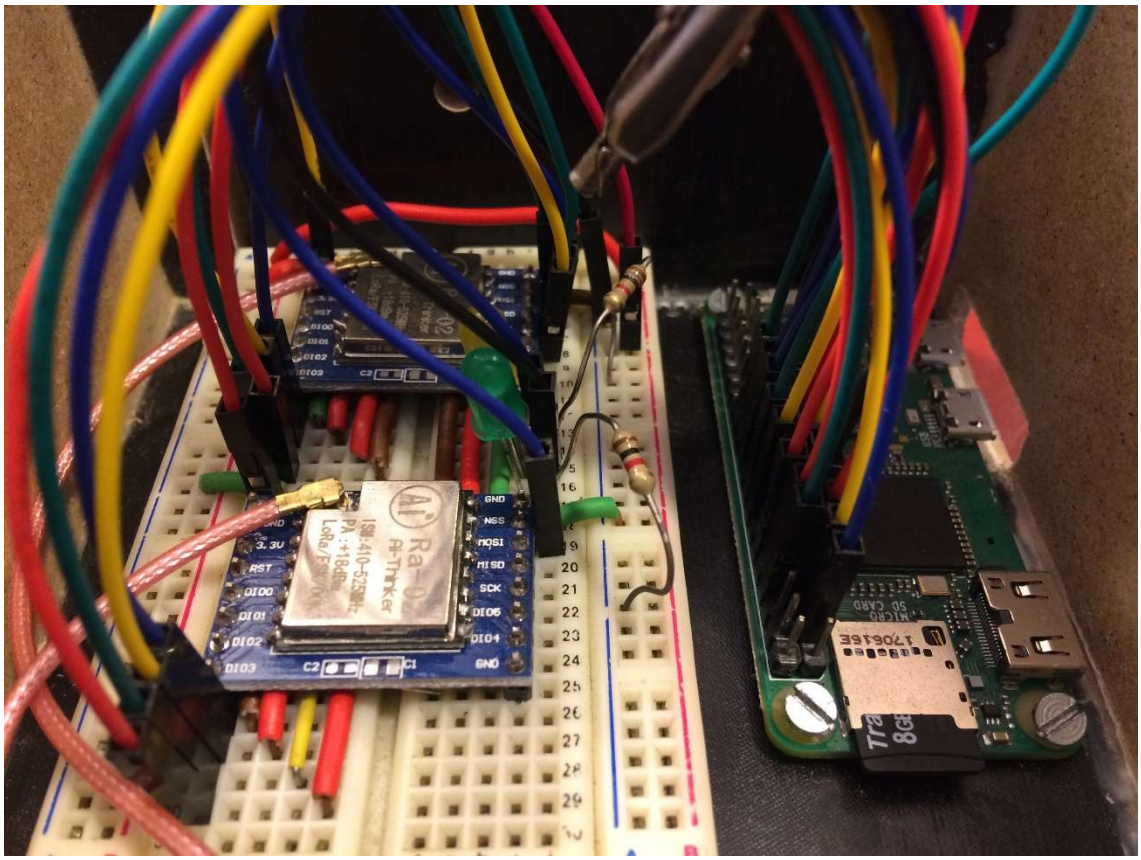


Figura 44 - Implementação do circuito da *gateway*

4.2 Software

Da mesma forma que o *hardware* o software encontra-se distribuído pelos sensores, atuadores e a *gateway*. Nos sensores e atuadores o *software* encontra-se todo no microcontrolador ATmega328P onde é guardado e executado.

Para os sensores e atuadores as definições específicas de cada dispositivo são configuradas nas constantes iniciais, como é o caso da frequência e o modo de operação do módulo rádio, os pinos I/O onde se encontram ligados os periféricos e o tempo máximo de espera pelo pacote de recebimento. No caso de um sensor analógico para além dessas configurações iniciais é também escolhido o intervalo de valores que correspondem à leitura do sensor escolhido, ou seja, o microcontrolador executa uma leitura de dez bits que tem de ser traduzida para um valor real que corresponde à leitura do sensor. Por exemplo o uso de um potenciómetro como sensor onde se pretende obter um valor em percentagem que corresponde ao valor da sua resistência, para isso, o valor máximo lido pelo microcontrolador é de 1023 (2^{10}) e corresponde a 100% do valor do potenciómetro. Esse intervalo é definido nas variáveis iniciais e depende do tipo de sensor que é usado. Além deste parâmetro é também necessário escolher o valor da diferença que é

significativa para atualizar o sistema. Utilizando o mesmo exemplo do potenciómetro como sensor, o valor da diferença pode ser por exemplo 10, isto quer dizer que sempre que existir uma diferença de 10 entre o último valor enviado e o valor atual o microcontrolador enviará o novo valor para a *gateway,* para atualização. Para efeitos práticos um sistema com esta configuração detetará variações de 10% no sensor potenciómetro. A escolha deste parâmetro é fundamental para o bom funcionamento do sistema pois a escolha de um valor muito baixo enviará para a *gateway* alterações irrisórias, que não devem ser consideradas, a uma frequência muito elevada que provocará que a *gateway* não consiga atender pedidos de outros dispositivos. Por outro lado um valor muito alto neste parâmetro fará com que a função do sensor se torne desnecessária.

O código a executar num sensor do sistema começa por incluir bibliotecas e inicializar as variáveis. A Figura 45 mostra o código usado para a inclusão das bibliotecas e definição das constantes iniciais.

```
#include <SPI.h>      // SPI
#include <RH_RF95.h>  //HadioHead http://www.airspayce.com/mikem/arduino/RadioHead/
#include <Base64.h> // https://github.com/adamvr/arduino-base64
#include <AESLib.h> // https://github.com/DavyLandman/AESLib

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

// radio frequency
#define RF95_FREQ 434.0

// ALTERAR AQUI CONFORME O DEVICE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#define LED 4 // error led
#define SENSOR A0 // Pin of SENSOR (ANALOG)
#define LONG_RANGE_MODE 0   // Slow+long range mode =1
#define WAITING_TIME 2000 // Max waiting time for ACK (depende do modo LONG_RANGE_MODE se =0 -> 2000)
// Parametros MAP   map(sensorValue, 0, 1023, 0, 255);
#define MAP1 0
#define MAP2 1023
#define MAP3 0
#define MAP4 100
#define SIG_Difference 10 // Value significant difference to send (send only if there is 100 difference in OUTPUTValue for example)
```

Figura 45 - Inclusão das bibliotecas e definição das contantes iniciais

É feita também a configuração do ID do dispositivo que deve ser único no sistema. A Figura 46 mostra a definições.

```
// ALTERAR AQUI CONFORME O DEVICE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  strcpy(TP, "3"); // device type 3=sensor
  strcpy(ID, "001"); // device ID  (3 char) 001-999     medium_range=[001-099] Long_range=[100-999]
```

Figura 46 - Definição do tipo e do ID do dispositivo

De seguida é feita a configuração do módulo rádio que pode ser configurado para longo alcance ou médio alcance dependendo da finalidade do dispositivo. O código entra depois num ciclo infinito, nesse ciclo são realizadas novas leituras do pino digital ou analógico que se encontra no sensor e espera-se que haja um valor novo para enviar para a *gateway*. Caso exista um novo valor para enviar o microcontrolador envia o pacote contendo a mensagem com o valor, inicia um temporizador e ativa a *flag* “espera_por_ack”. A flag

"espera_por_ack" é testada de seguida, esta *flag* indica se o sensor está à espera de receber um reconhecimento do envio dos dados por parte da *gateway*, se for esse o caso, e se o temporizador for igual ao tempo máximo de espera pelo pacote, são enviados novamente os dados para a gateway. Este procedimento é feito três vezes antes de mostrar um sinal de erro ao utilizador. Se entretanto for recebido um pacote o microcontrolador desencripta-o e verifica se a mensagem lhe é destinada, se o for e contiver o ACK então a *flag* "espera_por_ack" é colocada a 0. Este funcionamento é mostrado no fluxograma da Figura 47.

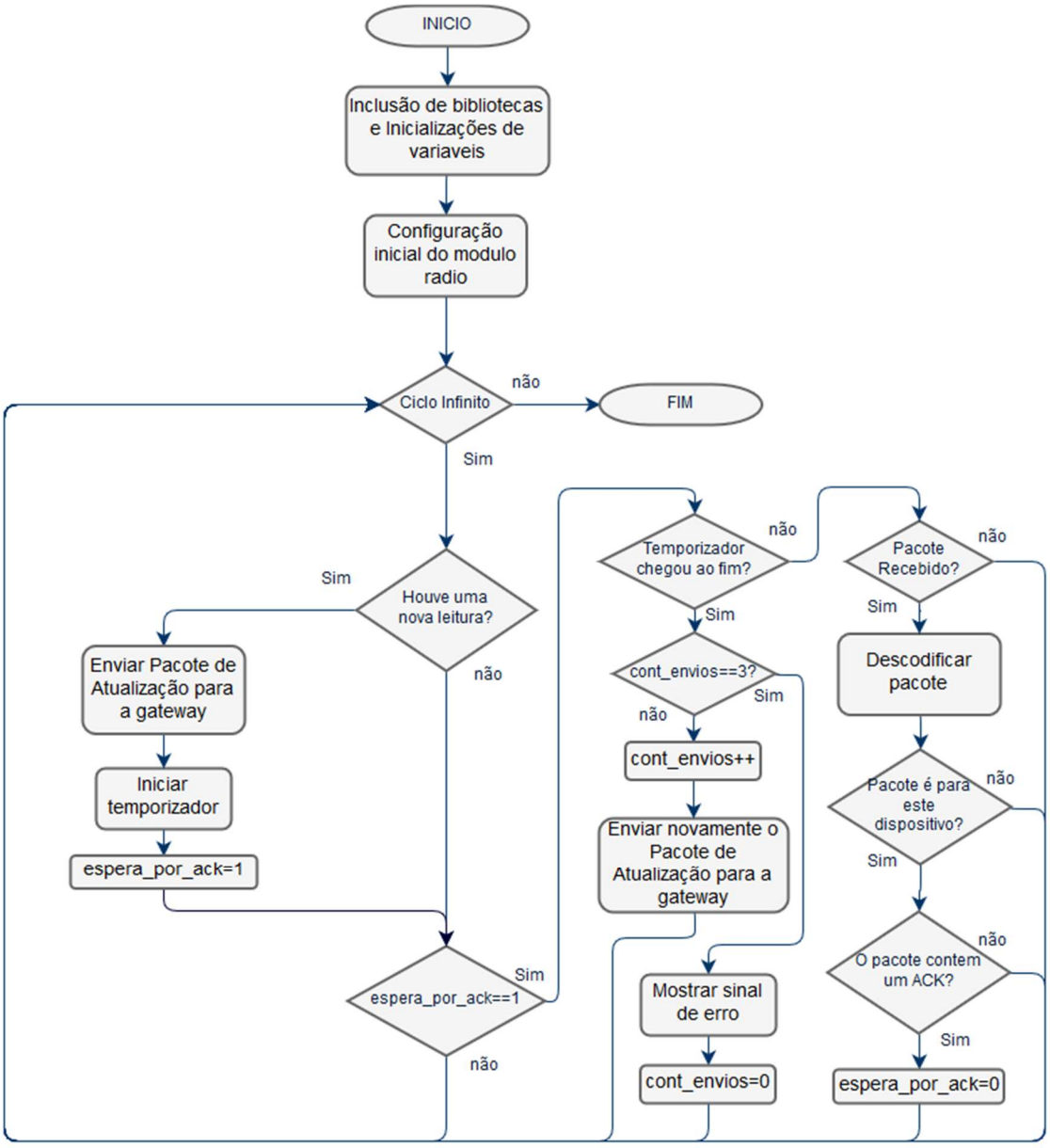


Figura 47 - Fluxograma do código de um sensor

O código executado nos atuadores segue o mesmo princípio do código dos sensores. Despois de definidas as configurações do dispositivo (constantes iniciais) a execução do

código começa por fazer a inclusão das bibliotecas e inicialização de variáveis passando de seguida para a configuração do módulo Ra-02 conforma as constantes iniciais. Dentro do ciclo infinito o módulo rádio é colocado à escuta e caso seja recebido um pacote o microcontrolador irá processá-lo, para isso, o pacote é desencriptado e é verificado se se destina ao atuador. Caso o pacote não se destine ao atuador este é descartado senão é verificada a mensagem e se contiver uma ordem o atuador executa-a colocando de seguida a *flag* "enviar_ack" a 1. Se o pacote contiver a mensagem ACK a *flag* "espera_por_ack" é colocada a 0. Dentro do ciclo se for verificado que a *flag* "enviar_ack" está a 1 é enviado o pacote com a mensagem ACK para a *gateway* e reiniciada a *flag*.

A cada ciclo é verificado o estado do interruptor do atuador, se este for premido é alterada a situação do atuador, por exemplo se a lâmpada do atuador de iluminação estivesse ligada um clique do botão faria desligar a lâmpada. Depois é enviado o pacote de atualização para a *gateway* informando-a que houve uma alteração manual do estado do atuador. Para concluir a alteração de estado o atuador tem de receber a confirmação da *gateway*, por isso é iniciado o temporizador e a *flag* "espera_por_ack" é colocada a 1. Se o atuador estiver à espera da confirmação da *gateway* da mudança de estado e não a receber quando o temporizador exceder o tempo máximo de espera, então é enviado um novo pacote de atualização para a *gateway*. Este procedimento ocorre três vezes antes de mostrar o sinal de erro ao utilizador. A Figura 48 mostra o fluxograma do código executado num atuador.

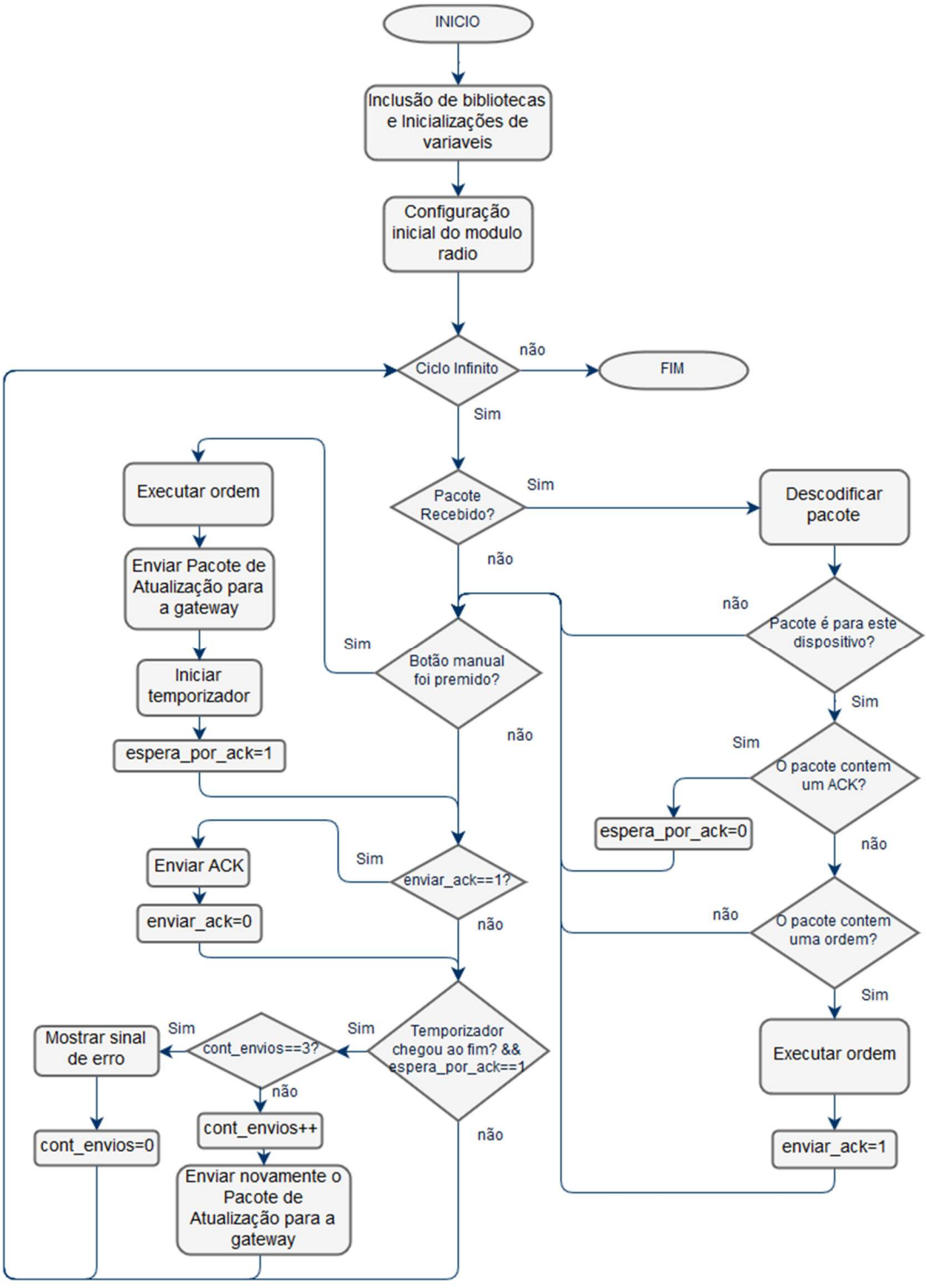


Figura 48 - Fluxograma do código de um atuador

Os sensores e atuadores dependem da *gateway* para o seu funcionamento. A plataforma AssistLora é o controlo do sistema dentro do *software* Home Assistant, esta é também fundamental para os componentes AssistLora que dependem da plataforma para a sua atividade.

A plataforma só contem uma função chamada de “setup” que é executada na inicialização do Home Assistant se estiver incluída no ficheiro de configuração “configuration.yaml”.

Depois da importação das bibliotecas são definidos os pacotes de código essenciais para o funcionamento do sistema AssistLora, incluindo as suas versões. Isto permite que caso alguma biblioteca não esteja atualizada no sistema o Home Assistant a instale antes de a plataforma iniciar.

A função “setup” é executada de seguida e irá configurar os dois módulos Ra-02 um para médio alcance e o outro para longo alcance. Para a configuração é usada sempre uma largura de banda de 125 kHz o que difere nos dois modos de funcionamento é o *Spreading Factor* utilizado. No caso da configuração para a comunicação de médio alcance é utilizado um *Spreading Factor* de 7 e para longo alcance um *Spreading Factor* de 12. De seguida são inicializadas as interrupções de *hardware* para a receção e envio de mensagens. As funções de processamento das mensagens recebidas e as funções de envio também são aqui declaradas. A Figura 49 mostra o processo de execução da plataforma AssistLora.

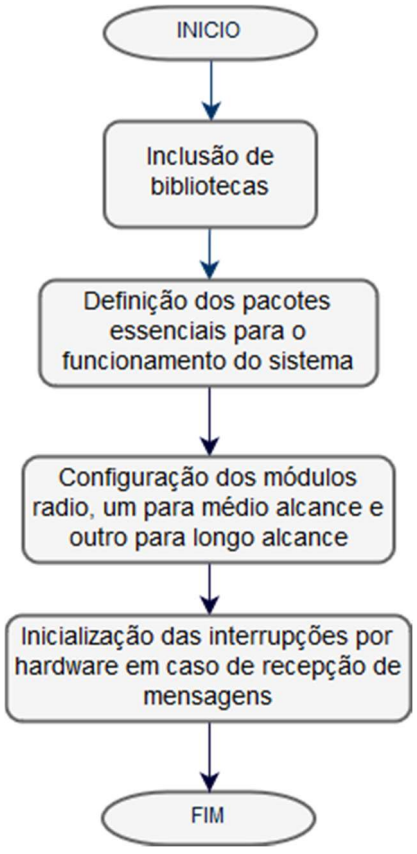


Figura 49 - Fluxograma da plataforma AssistLora integrada no Home Assistant

Os módulos rádio instalados na *gateway* estão configurados para estarem sempre à escuta. Caso seja recebido um pacote o módulo Ra-02 gera um sinal elétrico que é captado pelo Raspberry Pi, isto leva à execução do código da interrupção definida pela plataforma.

O primeiro passo após o sinal de receção de um pacote é a leitura desse mesmo pacote e de seguida a sua desencriptação. É verificado se o pacote se destina à *gateway*, se não for é automaticamente descartado. Caso a *gateway* seja o destino do pacote é feito o processamento da mensagem e depois a atualização da interface do Home Assistant. O envio do ACK para o emissor do pacote é o último passo. O fluxograma do código executado em caso de receção de uma mensagem é mostrado na Figura 50.

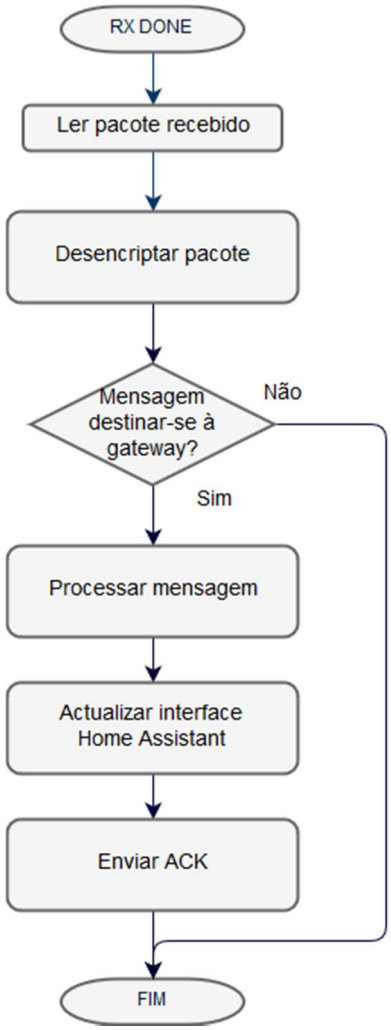


Figura 50 - Fluxograma do código executado em caso de receção de uma mensagem

Os componentes AssistLora só são iniciados depois da execução da função "setup" da plataforma. A função de cada componente é a criação de uma representação de um dispositivo dentro do sistema. Essa representação é executada a partir da função "setup_platform" dentro do código de cada componente. O componente criará tantas

representações como o número de dispositivos configurados no ficheiro "configuration.yaml". Dentro do ficheiro de configuração deve ser escolhido um ID por cada dispositivo instalado no sistema. Na Figura 51 é mostrado um exemplo de configuração de dois atuadores de tomada no ficheiro "configuration.yaml".

```
# AssistLora components
# id (identifier) - Medium Range=[1-99] Long Range=[100-999]
# icons - https://materialdesignicons.com/
# monitored_conditions (only sensor) - Custom like this 'ms' or ºC or Kg etc...
switch:
  - platform: lora
    devices:
      1:
        icon: fan
      100:
        icon: power-socket-eu
```

Figura 51 - Exemplo de configuração de atuadores de tomada no ficheiro "configuration.yaml"

Depois da execução da função "setup_platform" as representações dos dispositivos estão disponíveis na interface do Home Assistant. A partir da interface é possível o controlo de cada dispositivo que dependendo da sua representação apresenta controlos diferentes. Num atuador, por exemplo, é possível ligar ou desligar o controlo que se apresenta ao utilizador na interface como um interruptor. A ativação desse interruptor desperta no sistema a função "Turn_on" desse atuador que, por sua vez, irá enviar um pacote para o dispositivo correspondente com a mensagem para ativar a iluminação ou a tomada, dependendo do seu uso. De seguida irá esperar pela resposta do microcontrolador em que recebeu a mensagem e caso a receba a interface do Home Assistant é atualizada. O fluxograma do código executado num evento "Turn on" é mostrado no Figura 52.

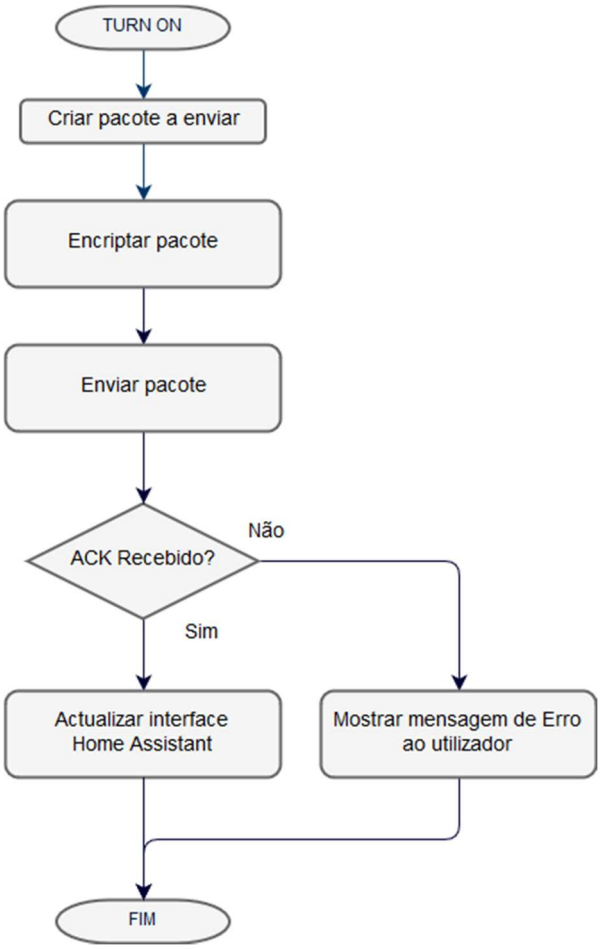


Figura 52 - Fluxograma do código executado num evento Turn on

O *website* criado é o primeiro contacto do sistema AssistLora com o utilizador. O *website* possui três páginas, "página inicial", "configurações do sistema" e "sobre o projeto" e pretende ser simples e direto. Devido à utilização da *framework* Bootstrap 4 o *website* adapta-se a qualquer dispositivo com ligação à internet. Todas as configurações disponíveis estão na página "configurações do sistema" e é também aí que se encontra a maior parte do código PHP usado. Nesta página de configurações é possível ver as redes Wi-Fi disponíveis, o IP do sistema assim como a rede de Wi-Fi a que a *gateway* está ligada. É possível ligar o sistema a outra rede Wi-Fi inserindo os dados da mesma, também é possível alterar o nome da rede criada pela *gateway*. As opções de desligar e reiniciar a *gateway* estão também nesta página.

A alteração do nome da rede criada pela *gateway* passa por editar o ficheiro "hostapd.conf" e a introdução dos dados de uma nova rede Wi-Fi é feito acrescentando os mesmos dados no ficheiro "wpa_supplicant.conf" através de linguagem PHP. O código para acrescentar uma nova rede Wi-Fi ao sistema é mostrado na Figura 53.

```php
if ($_POST["gravar"]=="Gravar"){
  $myfile = fopen("/etc/wpa_supplicant/wpa_supplicant.conf", "a") or die("Unable to open file!");
  $txt = "\nnetwork={\n";
  fwrite($myfile, $txt);
  $txt = "\t".'ssid="'.$_POST["rede"].'"'."\n";
  fwrite($myfile, $txt);
  $txt = "\t".'psk="'.$_POST["psw"].'"'."\n"."}";
  fwrite($myfile, $txt);
  fclose($myfile);
```

Figura 53 - Código PHP usado para acrescentar uma nova rede Wi-Fi ao sistema

O *website* permite também o acesso fácil à central de controlo do Home Assistant e ajuda a alguns problemas comuns.

4.3 Funcionamento do Protótipo

O funcionamento do sistema AssistLora pretende ser o mais simples e fácil para o utilizador, para isso, durante a execução do projeto, foram feitas escolhas nesse sentido. O protótipo construído inclui a *gateway* e dois atuadores, um de iluminação e outro de tomada.

Para o utilizador começar a utilizar o sistema os sensores e atuadores tem de ser instalados em primeiro lugar. Dependendo da função do dispositivo a instalação é diferente, os atuadores necessitam de ser ligados à rede doméstica de 220 VAC já os sensores necessitam de uma alimentação de 5 V que pode ser fornecida a partir de baterias ou pela rede de 220 VAC. Depois da alimentação dos dispositivos é necessário fazer a ligação destes aos periféricos de atuação ou aos sensores, no caso de ser necessário a utilização de um sensor analógico que tenha uma saída de sinal não compreendido no intervalo de aceitação do dispositivo sensor do AssistLora (entre 0 V e 3,3 V) então é fundamental a utilização de um circuito de condicionamento de sinal externo ao sistema.

Depois da instalação dos dispositivos o sistema está pronto a funcionar, para iniciar o funcionamento posiciona-se a *gateway* no centro da área de operação e liga-se à alimentação. O sistema inicializará automaticamente e criará uma rede Wi-Fi chamada "AssistLora" a qual o utilizador se deve ligar a partir de qualquer equipamento com acesso Wi-Fi. Esta rede está protegida pela *password* "assistlora". Na Figura 54 é possível ver a ligação do equipamento à rede "AssistLora".

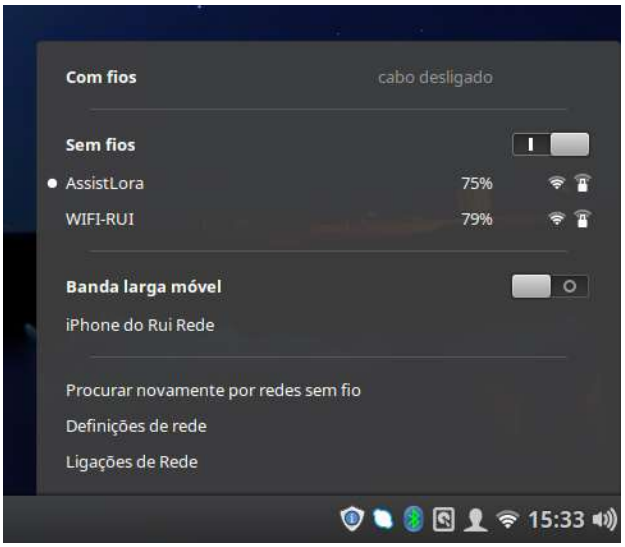


Figura 54 - Ligação à rede criada pela gateway

Depois do equipamento estar ligado à rede AssistLora o utilizador deve dirigir-se ao *website* www.assistlora.com para aceder à interface do sistema. Na Figura 55 é mostrado a página inicial do *website*.

Figura 55 - Página inicial da interface do sistema

O *website* criado fornece a possibilidade de fazer alterações ao sistema de maneira simples através da página "configurações do sistema". A Figura 56 mostra a página de configurações do *website*.

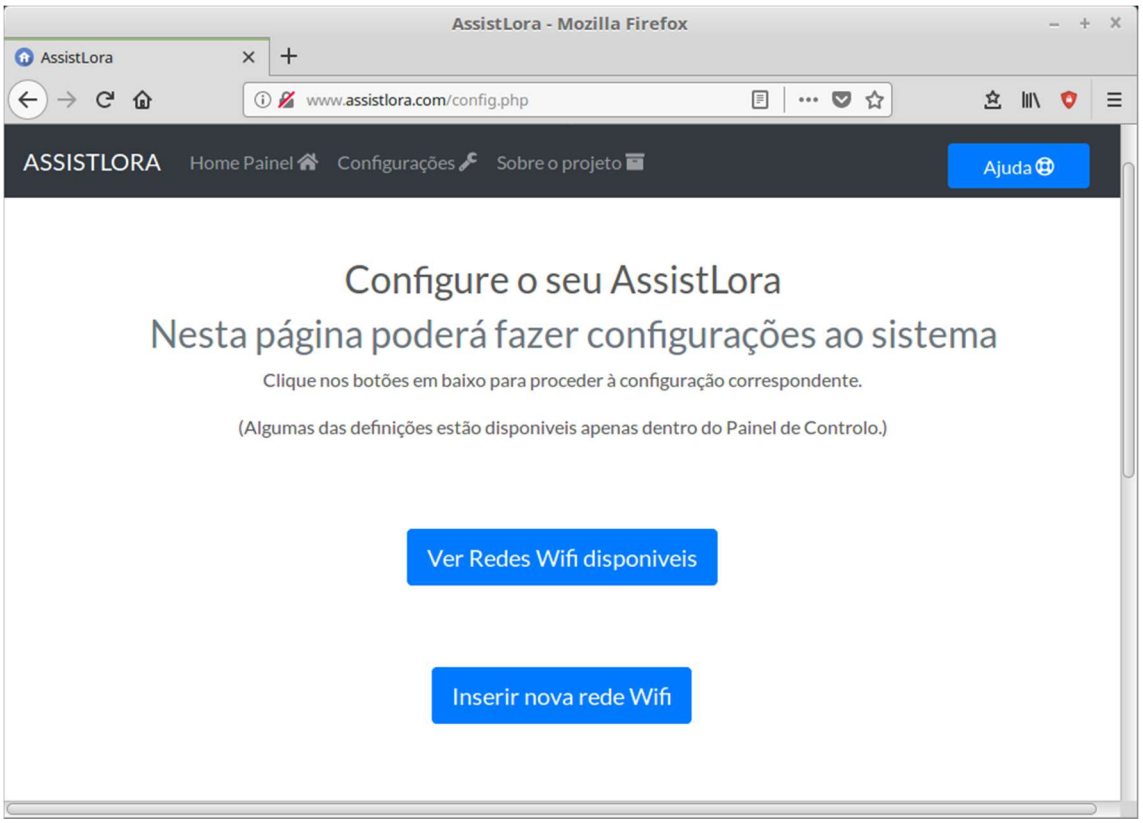


Figura 56 - Página do *website* "configurações do sistema"

O *website* contém também texto de ajuda ao utilizador, uma página com informação sobre o projeto e uma ligação direta à interface de controlo do Home Assitant. Para aceder à interface de controlo o utilizador apenas tem de clicar em "Painel de controlo" no menu do *website* ou no botão da página inicial. Antes da apresentação da interface de controlo o utilizador tem de colocar os dados de acesso, que, inicialmente, estão configuradas para o utilizador "admin" com a palavra-chave "admin". A palavra-chave de acesso pode e deve ser alterada dentro da interface de controlo, onde o utilizador poderá também criar outros usuários da plataforma com configurações diferentes. Na Figura 57 é mostrado o painel de login da interface.

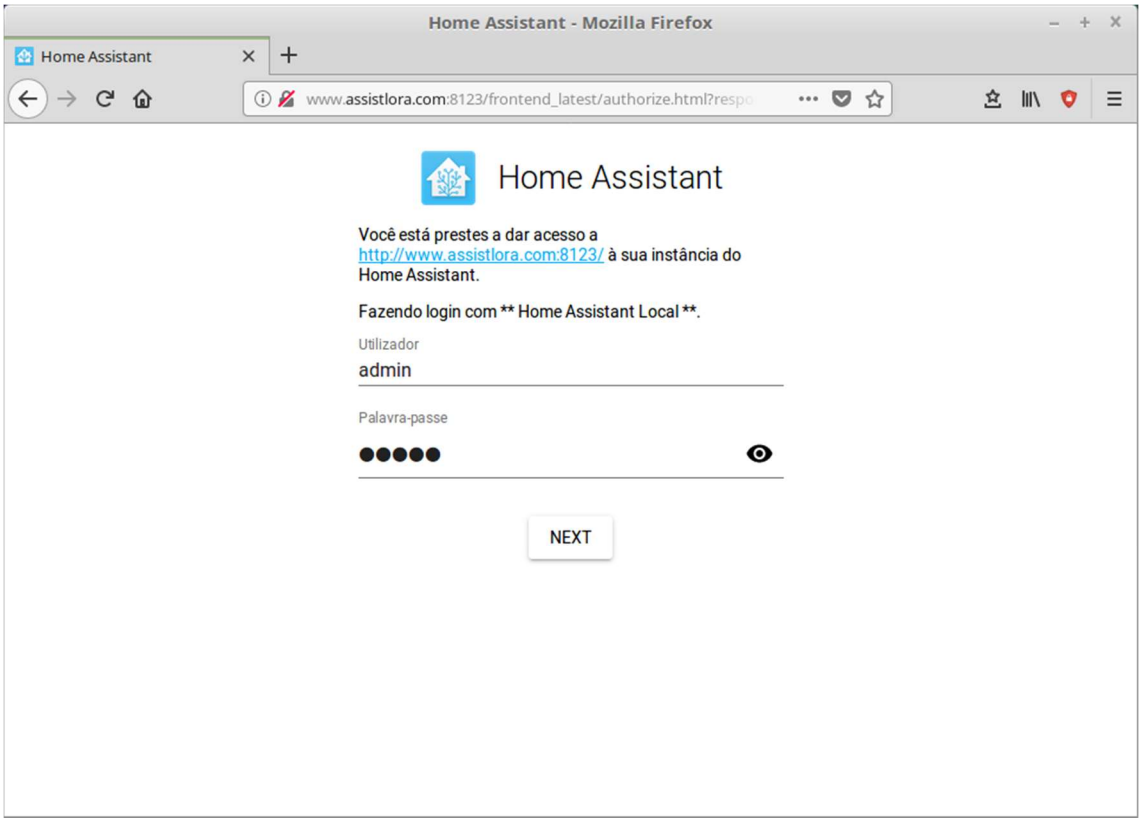


Figura 57 - Painel de login inicial

Após a introdução das credenciais de acesso é mostrado a interface de controlo do Home Assistant ao utilizador. Na Figura 58 é apresentada a interface do Home Assistant.

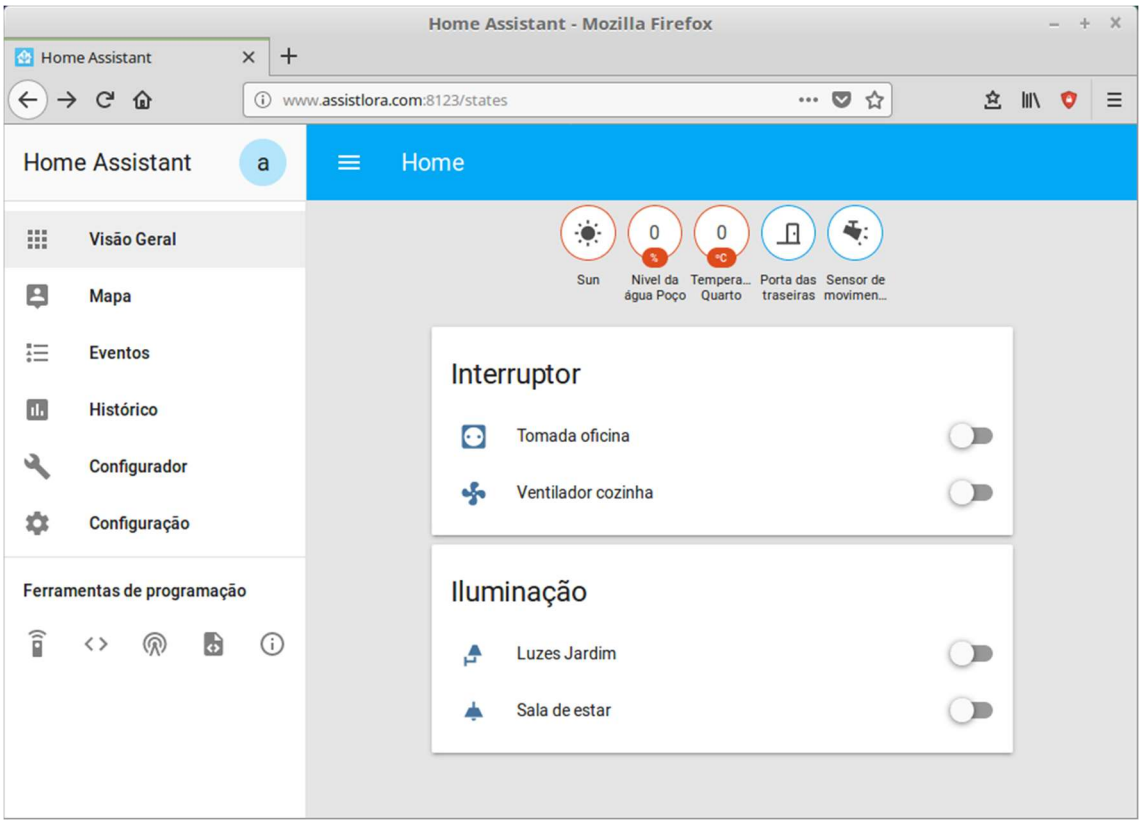


Figura 58 - Interface de controlo do Home Assistant

Inicialmente a interface apresenta oito dispositivos AssistLora configurados, dois dispositivos de cada estilo diferente um com configuração de longo alcance e o outro configurado para médio alcance. No entanto esta configuração não está em conformidade com a instalação inicial dos dispositivos e serve apenas de exemplo. De modo a adaptar o sistema à instalação inicial é necessário alterar o ficheiro "configuration.yaml" com as informações dos dispositivos instalados. Essa alteração pode ser feita a partir do menu do Home Assistant clicando em "Configurador" e de seguida em "configuration.yaml". É fundamental a indicação do ID e de um *icon* para cada dispositivo instalado. No caso do sensor analógico é necessário indicar também a unidade de medição do sensor (por exemplo: Kg, m, ou Cº). É mostrado na Figura 59 o editor do ficheiro "configuration.yaml".

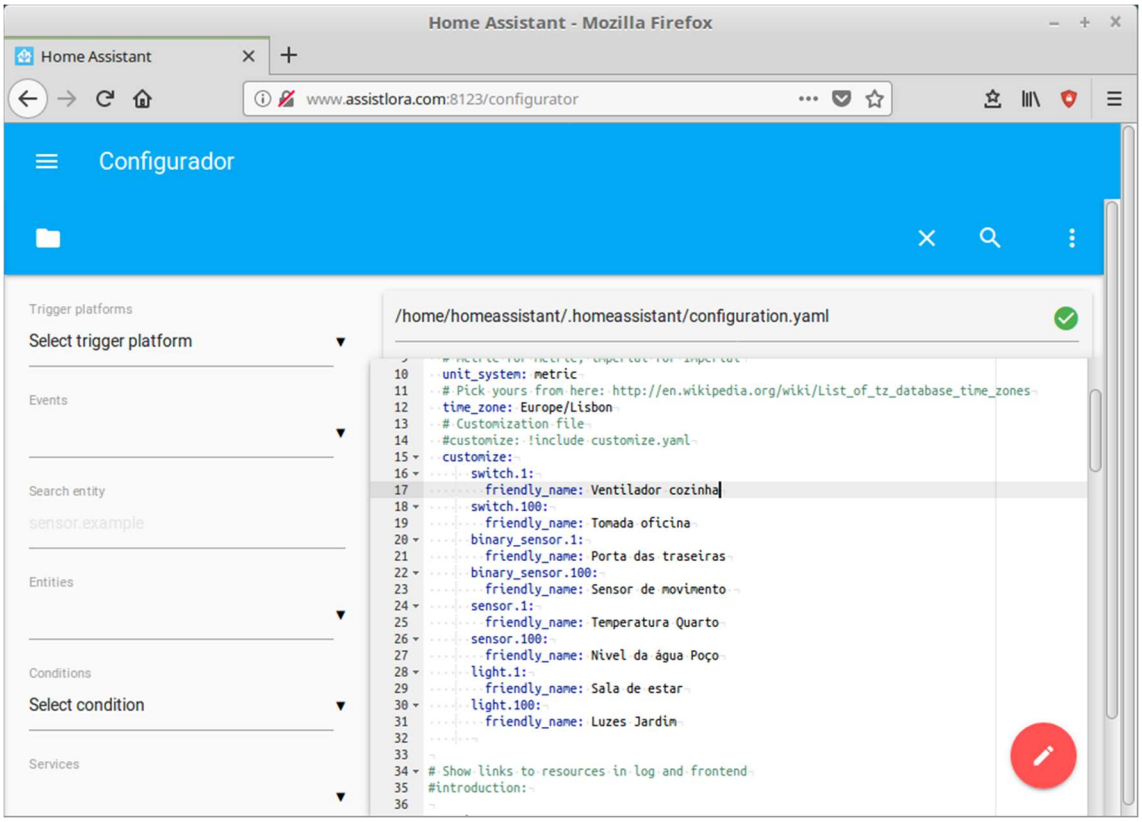


Figura 59 - Configuração dos dispositivos utilizando o "Configurador"

Depois de feita esta configuração o sistema está pronto a funcionar. O sistema pode ser ligado à internet utilizando para isso um adaptador Wi-Fi USB e adicionando o nome e a *password* da rede Wi-Fi a que o sistema se vai ligar na página das "configurações do sistema" do *website*. A Figura 60 mostra a introdução dos dados de uma rede Wi-Fi.

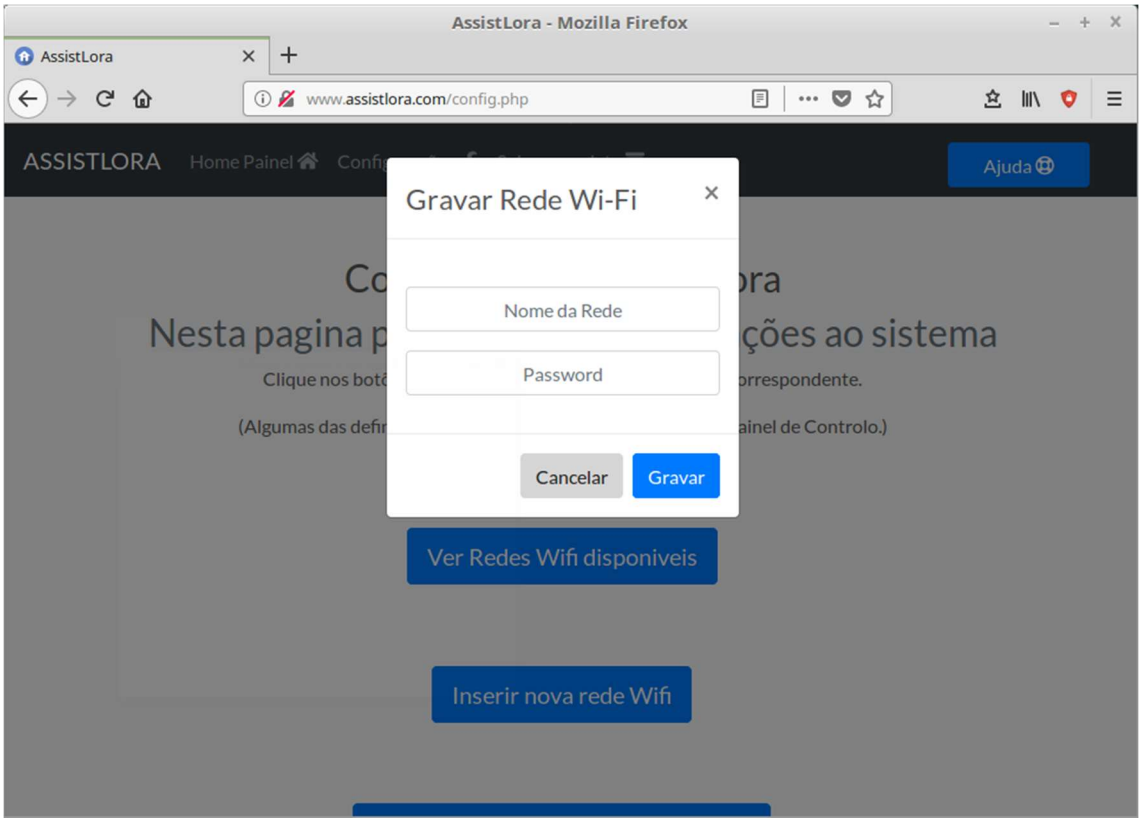


Figura 60 - Introdução de dados de uma rede Wi-Fi no sistema

O acesso ao sistema fora da rede local pode ser feito de várias maneiras mas é aconselhável o acesso a partir de uma VPN instalada dentro da rede, acedendo ao IP da *gateway* que pode ser consultado na página de "configurações do sistema". A partir da opção "Ver rede Wi-Fi" é possível ver várias informações sobre a interface Wi-Fi, Figura 61.

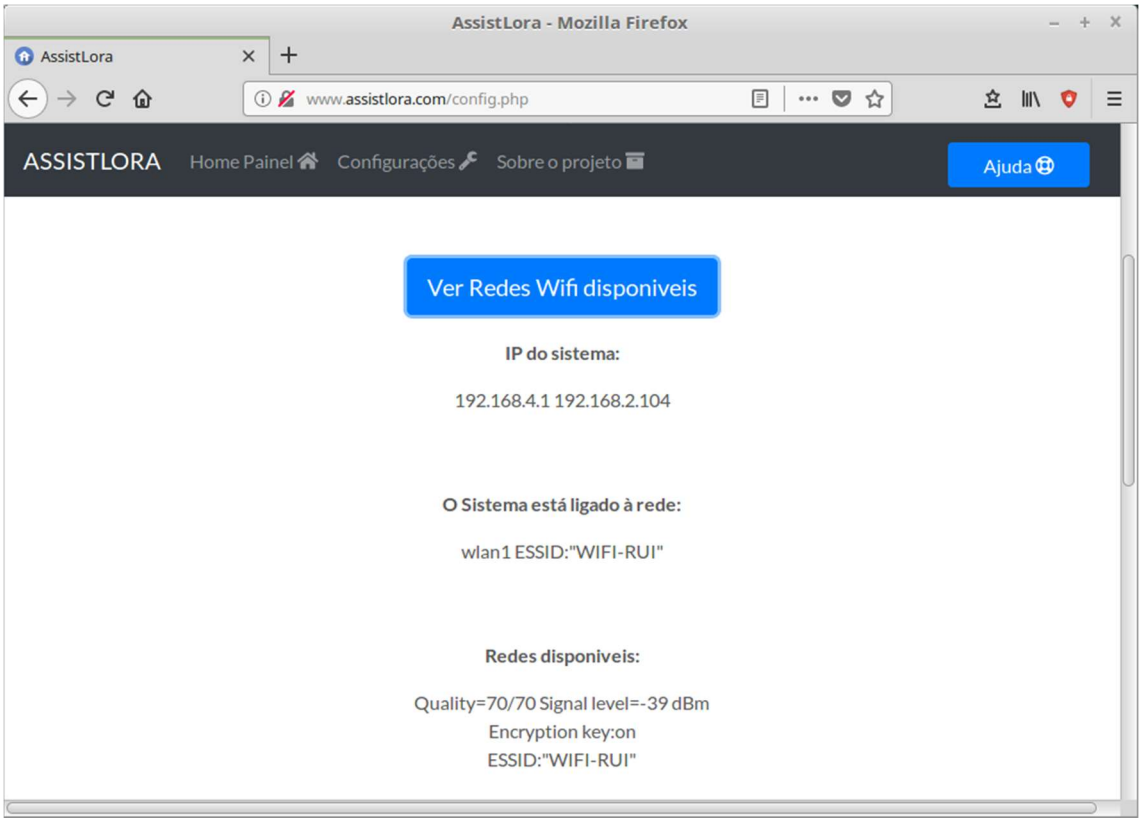


Figura 61 - Informações disponíveis na opção "Ver rede Wi-Fi".

É possível também a partir da página de "configurações do sistema" desligar e reiniciar a *gateway* caso o utilizador deseje. Na Figura 62 pode-se ver a opção de desligar a *gateway* selecionada.

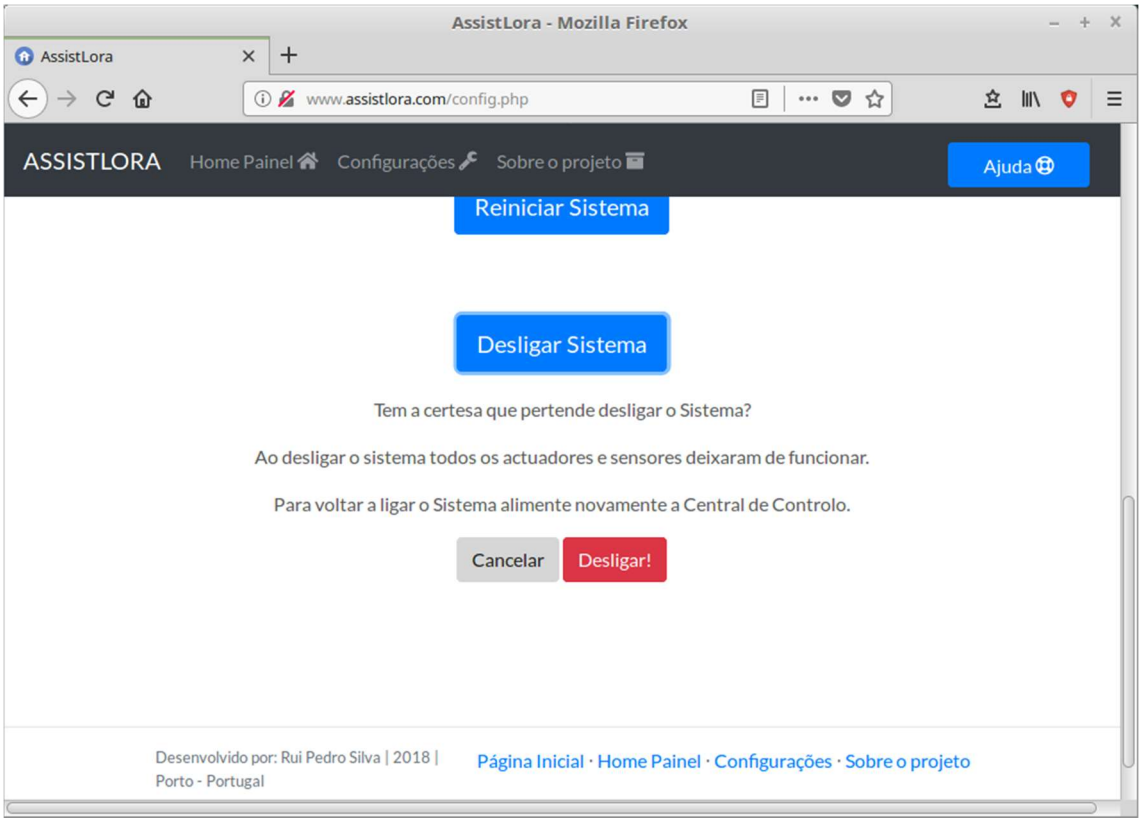


Figura 62 - Opção de desligar a gateway

4.4 Testes realizados

Depois da conclusão do protótipo foram realizados testes, nomeadamente, testes de consumo, tempos de inicialização e reação e testes de alcance de comunicação.

Começou-se por medir os tempos de inicialização dos dispositivos e da *gateway*. Para medir o tempo de inicialização dos dispositivos começou-se a contar o tempo a partir da alimentação até a ativação de um LED (indicando que o dispositivo fez as configurações iniciais e está operacional). O tempo medido foi aproximadamente de 2 segundos nos sensores e ativadores. De seguida mediu-se o tempo de inicialização da *gateway* desde a ativação até à criação da rede AssistLora e de seguida até à disponibilização da interface de controlo do Home Assistant. Os tempos medidos foram de 30 segundos, 1 minuto e 40 segundos respetivamente.

Concluído o teste dos tempos de inicialização foi realizado o teste de consumos energéticos. A *gateway* apresenta um consumo médio de 0,75 W sem a utilização de um adaptador USB Wi-Fi. O maior consumo da *gateway* ocorre na inicialização mas não apresenta picos de corrente superiores a 500 mA, podendo ser alimentada a partir de um conversor CA/CC de apenas 2,5 W, sem problemas. Com um adaptador USB Wi-Fi o consumo médio eleva-se para 1,25 W.

Os consumos dos dispositivos são muito baixos comparativamente ao da *gateway*. No caso dos sensores, com alimentação de 5 V e em funcionamento é utilizada uma corrente de 13,3 mA, sendo que em modo de suspensão é gasta uma corrente de 0,368 mA. Grande parte do consumo do dispositivo provém do conversor CC/CC que transforma os 5 V em 3,3 V gerando um gasto de 0,272 mA, já o microcontrolador e o módulo rádio colocados em modo "power down" e "sleep" respetivamente geram um consumo de apenas 96 µA.

Os atuadores representam um consumo superior aos sensores. O consumo em funcionamento do microcontrolador com o modo rádio à escuta é de 17,3 mA e inclui o consumo de um LED que está sempre ativo. A ativação do relé eleva o consumo do dispositivo em 75 mA, ou seja, um atuador de iluminação que receba o comando para ligar uma lâmpada apresentará um consumo de 92,3 mA (17,3 + 75) enquanto a lâmpada estiver ligada e um consumo de 17,3 mA quando esta estiver desligada. Os atuadores são equipados com um conversor CA/CC com uma eficiência superior a 70% o que aumenta o consumo total do dispositivo conforme a corrente consumida. Na Figura 63 é mostrado o teste de consumo realizado ao atuador de tomada.

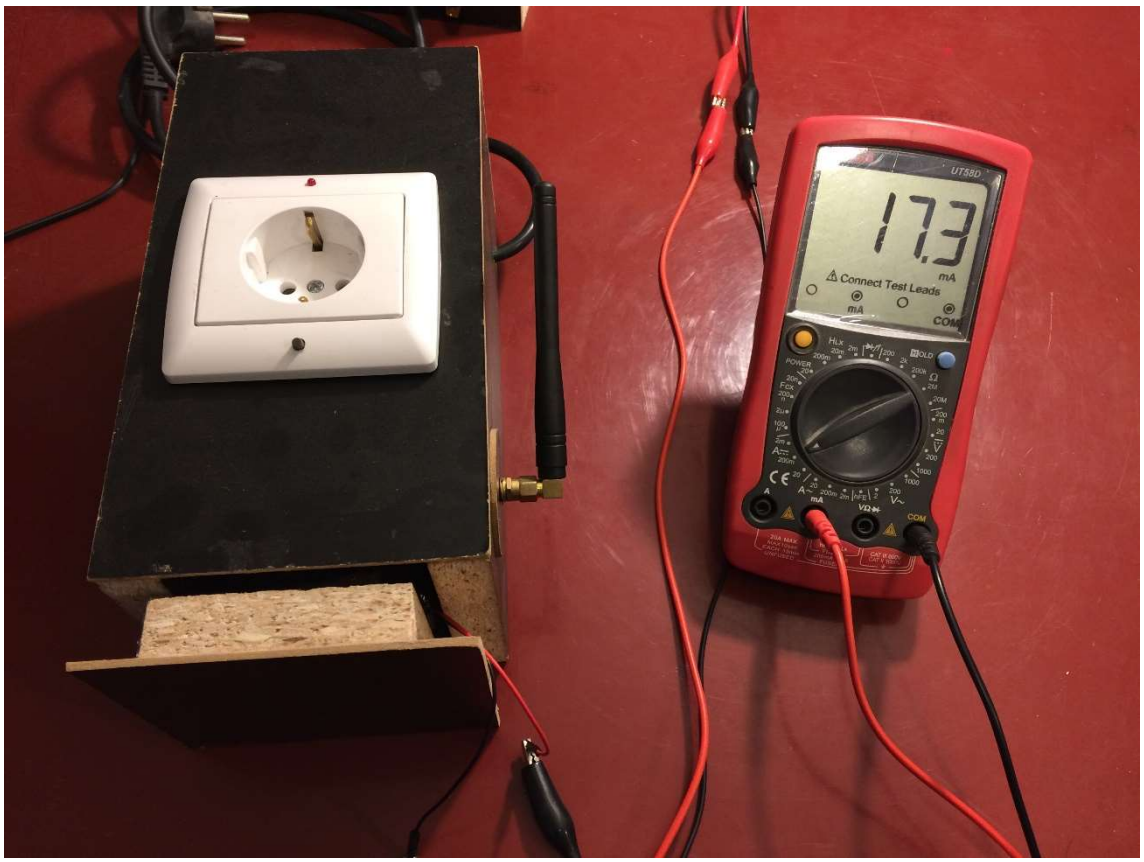


Figura 63 - Teste de consumo realizado ao atuador de tomada

Os sensores e atuadores apresentam picos de consumo durante a transmissão de dados, estes variam caso o dispositivo tenha sido configurado para longo ou médio alcance. No

caso de o dispositivo ter sido configurado para médio alcance o consumo eleva-se em cerca de 10,7 mA, em relação ao consumo total e no caso de ser configurado para transmissão de longo alcance essa subida é de 66,4 mA.

Pode-se concluir que o consumo é maior nos dispositivos com o módulo rádio configurado para longo alcance, não só pelo aumento de corrente comparativamente aos dispositivos de médio alcance, mas também pelo tempo em que o consumo se mantém elevado. A transmissão de longo alcance demora cerca de 1 segundo a ser concluída enquanto que a transmissão de médio alcance dura menos de meio segundo. Em suma, durante todo o tempo de transmissão a corrente mantém-se elevada registando-se assim mais consumo nos dispositivos de longo alcance.

Seguiu-se a medição dos tempos de comunicação. Utilizando a interface de controlo mediu-se o tempo desde que a ordem do utilizador é colocada na interface até à execução da ordem no atuador e a confirmação de que a ordem foi executada. Utilizando um atuador configurado para longo alcance o tempo medido desde a colocação da ordem na interface e a execução da mesma no atuador foi de 2,5 segundos. A confirmação que a ordem foi executada com sucesso demora cerca de 5 segundos desde que a ordem foi dada pelo utilizador. No caso de o atuador ser configurado para médio alcance os tempos são relativamente menores. O tempo de colocação da ordem do utilizador na interface de controlo até que esta seja executada pelo atuador é menos de 0,5 segundos e a confirmação é recebida pelo utilizador em cerca de 1 segundo.

Foram medidos os tempos para que, na eventualidade de o utilizador dar uma ordem para um dispositivo que não exista ou que esteja fora de alcance, o utilizador recebe a informação de erro. Para dispositivos configurados para longo alcance o tempo desde a ordem do utilizador até à mensagem de erro é de 10,5 segundos. Para dispositivos de médio alcance esse tempo é de 3,5 segundos. Se uma ordem for executada no atuador, através do botão manual, ou houver uma alteração num sensor e a *gateway* estiver fora do alcance do dispositivo é mostrado um sinal de erro através do LED incluído. Foram medidos os tempos desde o envio do primeiro sinal de atualização para a *gateway* até a indicação de erro no dispositivo. No caso da utilização da configuração de longo alcance o tempo medido foi de 13 segundos e nos dispositivos de médio alcance foi de 4,5 segundos.

Um dos fatores de melhoria deste projeto comparativamente a outros sistemas de domótica sem fios no mercado é o longo alcance da comunicação, por essa razão os testes de alcance são importantes.

Foram realizados testes de alcance dentro de uma habitação com diferentes configurações e em campo aberto.

Para o primeiro teste foi realizado em dois pisos diferentes de uma habitação colocando a *gateway* a uma distância de 12 metros em comprimento e 3 metros de altura de dois atuadores, um configurado para longo alcance e outro para medio alcance. O teste foi realizado com sucesso mostrado que o sistema era operacional nestas configurações. De seguida alterou-se a disposição dos dois atuadores colocando-os a uma diferença de dois pisos de distância, equivalente a 6 metros de altura e mantendo os 12 metros de comprimento. Nesta configuração apenas o atuador de longo alcance se manteve operacional. Por último, realizou-se um teste no mesmo piso mas utilizando cinco paredes como obstáculos entre a *gateway* e os atuadores. Ambos os atuadores conseguiram comunicar com a *gateway* nesta disposição.

Seguiu-se o teste em campo aberto, utilizando um equipamento configurado para medio alcance foi possível estabelecer comunicação com a *gateway* a 250 metros. Já utilizando um equipamento de longo alcance foi possível controla-lo a 600 metros de distância.

4.5 Resultados

Na execução do protótipo foi contruído, para além da *gateway*, dois atuadores, um de iluminação e outro de tomada. O protótipo permite a validação do funcionamento do projeto para alem de possibilitar a melhor compreensão do sistema.

Os atuadores foram contruídos de modo a ser possível aceder à eletrónica de maneira fácil. Para isso os atuadores possuem uma gaveta que permite aceder aos componentes e ligações elétricas. A energia elétrica é obtida a partir dos 220 VAC de uma tomada elétrica convencional.

Na atuador de iluminação foi acoplado um casquilho de lâmpada no dispositivo para alem do interruptor e do LED indicativo. Na Figura 64 é mostrada uma foto do atuador de iluminação.

Figura 64 - Atuador de iluminação

Na atuador de tomada exista para além da tomada elétrica um botão e um LED indicativo. O atuador de tomada é mostrado na Figura 65.

Figura 65 - Atuador de tomada

A *gateway* não apresenta qualquer periférico disponível sendo o seu controlo todo feito a partir de um equipamento ligado à rede Wi-Fi AssistLora. Para ativar a *gateway* basta alimenta-la a partir da abertura no recipiente reservado para o efeito. A ligação ao adaptador USB Wi-Fi pode também ser feita a partir desta abertura. O acesso à eletrónica

pode ser feito a partir de cima da caixa retirando os dois parafusos. Uma foto da *gateway* é mostrada na Figura 66.

Figura 66 – Central de controlo do sistema (*gateway*)

5 Conclusões e Trabalho Futuro

O sistema de domótica sem fios utilizando rede de longa distância e baixa potência foi concluído com sucesso pois todos os requisitos traçados foram cumpridos. No entanto este projeto deixa espaço para melhorias e desenvolvimentos futuros.

O sistema desenvolvido é capaz de controlar iluminação, tomadas e ainda receber informação de sensores binários e analógicos, além disso devido à integração com a rede Wi-Fi os utilizadores poderão ter acesso à interface de controlo em qualquer dispositivo com acesso a esta rede. Utilizando o microcomputador Raspberry Pi Zero W como unidade de processamento principal é possível diminuir os gastos de energia obtendo ainda assim as capacidades básicas para executar o *software* de domótica escolhido, o Home Assistant. Devido ao seu funcionamento, esta interface de domótica *open sourse* permite a inclusão de plataformas customizadas criadas por terceiros tendo sido assim uma excelente escolha para ser usada neste projeto. Além disso a capacidade de abstração do Home Assistant aos diferentes protocolos de comunicação permite a integração de vários sistemas de domótica a funcionar em conjunto, um dos requisitos deste projeto.

Os atuadores e sensores utilizam a tecnologia de comunicação LoRa de longo alcance e baixa potência tal como foi definido nos requisitos do projeto, e de modo a tornar a comunicação segura foi utilizado o protocolo de encriptação AES. A utilização de um microcontrolador como o ATmega328P permitiu o baixo consumo de cada dispositivo.

Outro requisito estabelecido era a criação de um sistema de baixo custo e compacto. Comparando com os sistemas de domótica estudados este projeto pode facilmente competir por espaço no mercado caso fosse mais desenvolvido no sentido de otimização do circuito eletrónico.

Durante a realização do projeto surgiram diversos problemas, alguns de difícil resolução, como a comunicação LoRa entre o microcomputador e o microcontrolador. A programação da plataforma a executar no Home Assistant foi também uma tarefa difícil que se deveu ao *software* Home Assistant ser um projeto recente e não disponibilizar documentação suficiente para os programadores. Estes foram os principais problemas que necessitaram de muito tempo de pesquisa e análise de código.

Neste projeto desenvolveu-se a base de um sistema de domótica no entanto é deixado em aberto desenvolvimentos futuros que permitirão uma possível comercialização de um produto. O desenvolvimento de placas de circuito impresso para substituir os circuitos

nas *breadboards* é fundamental para um produto escalável. A criação de outros dispositivos de controlo como persianas, portões, alarmes, etc, ficará para um futuro que se pretende próximo. De forma a aumentar a eficiência energética dos dispositivos é possível eliminar o conversor CC/CC, que converte os 5 V para 3,3 V, e alimentar o sistema diretamente com 3,3 V substituindo o relé para um com uma bobine de 3 V.

6 Referencias Documentais

[1] S. Kumar e M. A. Qadeer, "Application of AI in Home Automation," *IACSIT International Journal of Engineering and Technology, Volume 4, Numero 6,* pp. 803-807, 12 2012.

[2] J. Hill, "Best home automation system," *Consumer Reports magazine,* 2014.

[3] T. Agarwal, "Wireless Communication Technologies Types and Advantages," Edgefx Kits & Solutions, 2015. [Online]. Available: https://www.efxkits.us/different-types-of-wireless-communication-technologies/. [Acedido em Maio 2018].

[4] IEEE, "IEEE Std 521-2002 Standard Letter Designations for Radar-Frequency Bands," 2009. [Online]. Available: http://standards.ieee.org/findstds/standard/521-2002.html. [Acedido em Março 2018].

[5] National Academies of Sciences, Engineering, and Medicine, "IEEE Standard Letter Designations for Radar Bands," em *Handbook of Frequency Allocations and Spectrum Protection for Scientific Uses*, Washington, DC, Second Edition. The National Academies Press, 2015, p. Appendix B.

[6] E. Alecrim, "O que é Wi-Fi (IEEE 802.11)?," 2013. [Online]. Available: https://www.infowester.com/wifi.php. [Acedido em Maio 2018].

[7] Adafruit, "Introduction to Bluetooth Low Energy," 2014. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction.

[8] P. Smith, "Comparing Low-Power Wireless Technologies," 2011. [Online]. Available: https://www.digikey.be/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies. [Acedido em Abril 2018].

[9] Z-Wave Alliance, "Z-Wave Transceivers - Specification of Spectrum Related Components," 2014. [Online]. Available: http://z-wavealliance.org/wp-content/uploads/2015/02/ZAD12837-1.pdf. [Acedido em Maio 2018].

[10] Honeywell, “An Introductory Guide to Z-Wave Technology,” 2 2013. [Online]. Available: http://library.ademconet.com/MWT/fs2/L5210/Introductory-Guide-to-Z-Wave-Technology.PDF. [Acedido em Abril 2018].

[11] R. Fritz, “What is Z-Wave?,” 5 2018. [Online]. Available: https://www.lifewire.com/what-is-z-wave-817700. [Acedido em Maio 2018].

[12] RF Wireless World, “Advantages of Z-wave | Disadvantages of Z-wave,” 2012. [Online]. Available: http://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-z-wave.html. [Acedido em Março 2018].

[13] NXP Laboratories UK, “ZigBee Home Automation User Guide,” 8 2016. [Online]. Available: https://www.nxp.com/docs/en/user-guide/JN-UG-3076.pdf. [Acedido em Maio 2018].

[14] Semtech Corporation, “Fractional-n Synthesized Chirp Generator,” 2010. [Online]. Available: https://patentimages.storage.googleapis.com/e7/cb/f0/8c1936044391c6/US7791415.pdf. [Acedido em Maio 2018].

[15] J. Zyren e A. Petrick, “Tutorial on Basic Link Budget Analysis,” 6 1998. [Online]. Available: http://www.sss-mag.com/pdf/an9804.pdf. [Acedido em Maio 2018].

[16] M. Willis, “Link Budgets,” 5 2007. [Online]. Available: http://www.mike-willis.com/Tutorial/PF13.htm. [Acedido em Agosto 2018].

[17] LoRa Alliance, “A technical overview of LoRa and LoRaWAN,” 11 2015. [Online]. Available: https://www.tuv.com/media/corporate/products_1/electronic_components_and_lasers/TUeV_Rheinland_Overview_LoRa_and_LoRaWANtmp.pdf. [Acedido em Abril 2018].

[18] Jirous Antennas, “Radio link budget calculator,” [Online]. Available: http://en.jirous.com/calculation-wifi. [Acedido em Setembro 2018].

[19] Techplayon, “LoRa Link-budget and Sensitivity Calculations,” 12 2017. [Online]. Available: http://www.techplayon.com/lora-link-budget-sensitivity-calculations-example-explained/. [Acedido em Agosto 2018].

[20] Semtech, “Application Note: LoRa Modulation Crystal Oscillator Guidance,” 7 2017. [Online]. Available: https://www.mouser.com/pdfdocs/an120014-xo-guidance-lora-modulation.pdf. [Acedido em Maio 2018].

[21] D. Layne, “Receiver Sensitivity and Equivalent Noise Bandwidth,” High Frequency Electronics, 2014. [Online]. Available: https://highfrequencyelectronics.com/index.php?option=com_content&view=article&id=553:receiver-sensitivity-and-equivalent-noise-bandwidth&catid=94:2014-06-june-articles&Itemid=189. [Acedido em Setembro 2018].

[22] I. Poole, “Radio receiver noise floor,” Adrio Communications Ltd, [Online]. Available: https://www.radio-electronics.com/info/rf-technology-design/rf-noise-sensitivity/noise-floor.php. [Acedido em Setembro 2018].

[23] H. Friis, “A Note on a Simple Transmission Formula,” *Proceedings of the IRE,* vol. 34, nº 5, pp. 254 - 256, 1946.

[24] P. Bevelacqua, “Friis Equation - Decibel Math,” 2015. [Online]. Available: http://www.antenna-theory.com/definitions/decibelMath.php. [Acedido em Setembro 2018].

[25] The Things Network Global Team, “Ground breaking world record! LoRaWAN packet received at 702 km (436 miles) distance,” The Things Network, 9 2017. [Online]. Available: https://www.thethingsnetwork.org/article/ground-breaking-world-record-lorawan-packet-received-at-702-km-436-miles-distance. [Acedido em Agosto 2018].

[26] S. Design, “TTN Mapper,” Stamen, 2018. [Online]. Available: http://ttnmapper.org/?gateway=B827EBFFFEB4B18A&type=radar. [Acedido em Novembro 2018].

[27] WAVIoT, “What is LPWAN?,” 2018. [Online]. Available: https://waviot.com/technology/what-is-lpwan. [Acedido em Agosto 2018].

[28] i-SCOOP, “LoRa and LoRaWAN: the technologies, ecosystems, use cases and market,” 2016. [Online]. Available: https://www.i-scoop.eu/internet-of-things-guide/iot-network-lora-lorawan/. [Acedido em Maio 2018].

[29] Oomi, “Product Manual Oomi In-Wall Switch”.

[30] Wulian, “Smart Gateway Wulian User Manual,” 2016. [Online]. Available: http://www.wuliangroup.com/en/uploads/5833d408/Wulian%20Smart%20Gateway%20(LAN).pdf. [Acedido em Maio 2018].

[31] Wulian, “Smart PIR Motion Detector User Manual,” 2016. [Online]. Available: http://www.wuliangroup.com/en/uploads/36300000610-Wulian&e7&ba&a2&e5&a4&96&e5&85&a5&e4&be&b5&e6&8e&a2&e6&b5&8b&e5&99&a804&e5&9e&8b&e4&bd&bf&e7&94&a8&e8&af&b4&e6&98&8e&e4&b9&a6&ef&bc&88Ember&ef&bc&8c&e8&8b&b1&e6&96&87)(8-6)-2016.11.4.pdf. [Acedido em Maio 2018].

[32] Kiwi electronics, “Raspberry Pi Zero W,” 2018. [Online]. Available: https://www.kiwi-electronics.nl/raspberry-pi/board-and-kits/raspberry-pi-zero-w. [Acedido em Maio 2018].

[33] Raspberry Pi Foundation, “FAQs,” [Online]. Available: https://www.raspberrypi.org/help/faqs/. [Acedido em Agosto 2018].

[34] G. Santiago, “Arduino pro mini + RFID RC522,” 14 12 2012. [Online]. Available: https://simplesoftmx.blogspot.com/2014/12/arduino-pro-mini-rfid-rc522.html. [Acedido em Maio 2018].

[35] electronilab, “Módulo Transceptor LoRa SX1278 Ra-02,” [Online]. Available: https://electronilab.co/tienda/modulo-transceptor-lora-sx1278-ra-02-largo-alcance-433-mhz/. [Acedido em Maio 2018].

[36] Semtech Corporation, “semtech.com,” 8 2016. [Online]. Available: https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V5.pdf. [Acedido em Agosto 2018].

[37] Shenzhen Ai-Thinker Technology Co, Ltd, “Ra-02 LoRa Product Specification V1.1,” 2017. [Online]. Available: http://wiki.ai-thinker.com/_media/lora/docs/c048ps01a1_ra-02_product_specification_v1.1.pdf. [Acedido em Maio 2018].

[38] SD-3C LLC, “SD Memory Card Formatter,” 2018. [Online]. Available: https://www.sdcard.org/downloads/formatter_4/. [Acedido em Agosto 2018].

[39] t. gruemaster, “Win32 Disk Imager,” 6 2018. [Online]. Available: https://sourceforge.net/projects/win32diskimager/. [Acedido em Maio 2018].

[40] The Apache Software Foundation, “Apache Server,” 2018. [Online]. Available: https://www.apache.org/. [Acedido em Maio 2018].

[41] P. Schoutsen, *Awaken your home: Python and the Internet of Things,* Portland, Oregon: PyCon 2016, 2016.

[42] Home Assistant, “Architecture,” 2018. [Online]. Available: https://developers.home-assistant.io/en/. [Acedido em Agosto 2018].

[43] M. McCauley, “RadioHead,” 2018. [Online]. Available: http://www.airspayce.com/mikem/arduino/RadioHead/. [Acedido em Maio 2018].

[44] DavyLandman e AVR-Crypto-Lib, “AESLib,” 2017. [Online]. Available: https://github.com/DavyLandman/AESLib. [Acedido em Agosto 2018].

[45] A. Rudd, “A base64 library for the arduino platform, written in C,” 5 2013. [Online]. Available: https://github.com/adamvr/arduino-base64. [Acedido em Julho 2018].

[46] PyCryptodome, “Docs - API documentation,” 4 2018. [Online]. Available: https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html. [Acedido em Julho 2018].

[47] Microsoft, “Visual Studio Code. Code editing Redefined.,” 2018. [Online]. Available: https://code.visualstudio.com/. [Acedido em Junho 2018].

[48] T. Kosse, “Filezilla project,” 4 2018. [Online]. Available: https://filezilla-project.org/. [Acedido em Julho 2018].

[49] Fairchid, “NPN Epitaxial Silicon Transistor,” 11 2014. [Online]. Available: https://www.fairchildsemi.com/datasheets/BC/BC547.pdf. [Acedido em Julho 2018].

[50] R. F. M. Brandão, “A domótica ao serviço da sociedade,” *ISEP - DEE - Neutro à Terra - Revista Técnico-Científica,* nº 1, pp. 4-6, 2008.

[51] V. S. Gunge e P. S. Yalagi, “Smart Home Automation: A Literature Review,” *International Journal of Computer Applications,* vol. IJCA Proceedings on National Seminar on Recent Trends in Data Mining, nº 1, 2016.

[52] e-Gizmo Mechatronix Central, “LORA Module RA-02 V.1,” 2017. [Online]. Available: https://www.e-gizmo.net/oc/kits%20documents/LORA%20Module%20RA-02%20V.1/LORA%20Module%20RA-02%20V.1.pdf. [Acedido em Maio 2018].

Anexos

Códigos

Código do Website – Pagina inicial – index.php

```
<!--
------ index.php ------
Pagina inicial
 -->

<!DOCTYPE html>
<html lang="pt-pt" xml:lang="pt-pt">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Pagina Pessoal">
    <meta name="author" content="Rui Pedro Silva">
  <link rel="shortcut icon" href="/img/icons/favicon.ico">
  <link rel="icon" type="image/png" href="/img/icons/favicon-48x48.png" sizes="48x48">
  <link rel="apple-touch-icon" sizes="128x128" href="/img/icons/apple-touch-icon-128x128.png">

    <title>AssistLora</title>

    <!-- Bootstrap Core CSS -->
    <link href="./css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="./css/inicial_style.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <script defer src="./js/fontawesome-all.js"></script>
    <link href="./css/fontawesome-all.css" rel="stylesheet">
    <link
href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic"
rel="stylesheet" type="text/css">
</head>

  <body>

<!-- Modal Ajuda-->
<div class="modal fade" id="modalajuda" tabindex="-1" role="dialog" aria-
labelledby="modalajudaLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
    <h4 class="modal-title" id="modalajudaLabel">Precisa de Ajuda?</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
    <p><strong>Bem Vindo ao AssistLora!</strong></p>
    <p>Esta é a pagina inicial do AssistLora, nesta pagina pode seguir para o Painel Controlo
clicando no botão <strong>Home Assistant</strong>. Caso pertenda alterar as definições ao
sistema clique em <strong>Configurações</strong> na barra do menu. Pode ver mais informação
sobre este projeto clicando em <strong>Sobre o Projeto</strong> no Menu.</p>
    <br>
    <p><strong>Problemas comuns:</strong> Não consigo aceder ao Home Assistant!</p>
    <p>  -> O Painel de Controla demora algum tempo a iniciar, se no fim de 5 minutos ainda
nao tiver iniciado pode reiniciar o equipamento nas <strong>Configurações</strong>.</p>
    <p><strong>Problemas comuns:</strong> Não sei o login para entrar no Home Assistant!</p>
    <p>  -> O Login predefinido é Login:<strong>admin</strong>
Password:<strong>admin</strong>. É aconselhável alterar a palavra chave assim que inicie o
equipamento por questões de segurança, para fazer isso clique no símbolo do utilizador dentro
do menu do Home Painel e de seguida em <strong>Change Password</strong>. </p>
      </div>
      <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Fechar</button>
      </div>
    </div>
```

```
  </div>
</div>

 <header>
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" href="index.php">ASSISTLORA</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="/" onclick="javascript:event.target.port=8123">Home Assistant
<i class="fas fa-home"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="config.php">Configurações <i class="fa fa-wrench"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="about.php">Sobre o projeto <i class="fas fa-archive"></i></a>
    </li>
    </ul>
    <ul class="nav navbar-nav navbar-right">
    <li><button type="button" style="width: 120px; margin:0px 10px; margin-top:8px;"
class="btn btn-primary" data-toggle="modal" data-target="#modalajuda"> Ajuda <i class="far fa-
life-ring"></i></button></li>
    </ul>
  </div>
  </nav>
</header>


  <div class="cover-container flex-column text-center">
      <main role="main" class="inner cover">
        <h1 class="cover-heading">Bem-Vindo ao AssistLora!</h1>
        <p class="lead">   </p>
        <br>
        <p class="lead">
          <a type="button" class="btn btn-lg btn-secondary" href="/"
onclick="javascript:event.target.port=8123"> Home Assistant  <i class="fas fa-home"></i></a>
        </p>
      </main>

      <footer class="mastfoot mt-auto">
        <div class="inner">
          <p >Desenvolvido por: Rui Pedro Silva | 2018 | Porto - Portugal </p>
        </div>
      </footer>
    </div>

    <!-- Bootstrap core JavaScript ================================================== -->
    <script src="./js/jquery-3.2.1.slim.min.js"></script>
    <script src="./js/bootstrap.min.js"></script>

  </body>
</html>
```

Código do Website – Pagina Sobre o projeto – about.php

```
<!--
------ about.php ------
Pagina sobre mim
 -->
<!--

<!DOCTYPE html>
<html lang="pt-pt" xml:lang="pt-pt">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Pagina Pessoal">
    <meta name="author" content="Rui Pedro Silva">
    <link rel="shortcut icon" href="/img/icons/favicon.ico">
  <link rel="icon" type="image/png" href="/img/icons/favicon-48x48.png" sizes="48x48">
  <link rel="apple-touch-icon" sizes="128x128" href="/img/icons/apple-touch-icon-128x128.png">

    <title>AssistLora</title>

    <!-- Bootstrap Core CSS -->
    <link href="./css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="./css/projects_style.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <script defer src="./js/fontawesome-all.js"></script>
    <link href="./css/fontawesome-all.css" rel="stylesheet">
    <link
href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic"
rel="stylesheet" type="text/css">
</head>


  <body>

<!-- Modal Ajuda-->
<div class="modal fade" id="modalajuda" tabindex="-1" role="dialog" aria-
labelledby="modalajudaLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
    <h4 class="modal-title" id="modalajudaLabel">Precisa de Ajuda?</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
    <p><strong>Bem Vindo ao AssistLora!</strong></p>
    <p>Esta é a pagina inicial do AssistLora, nesta pagina pode seguir para o Painel Controlo
clicando no botão <strong>Home Painel</strong>. Caso pertenda alterar as definições ao sistema
clique em <strong>Configurações</strong> na barra do menu. Pode ver mais informação sobre este
projeto clicando em <strong>Sobre o Projeto</strong> no Menu.</p>
    <br>
    <p><strong>Problemas comuns:</strong> Não consigo aceder ao Home Painel!</p>
    <p>  -> O Painel de Controla demora algum tempo a iniciar, se no fim de 5 minutos ainda
nao tiver iniciado pode reiniciar o equipamento nas <strong>Configurações</strong>.</p>
    <p><strong>Problemas comuns:</strong> Não sei o login para entrar no Home Painel!</p>
    <p>  -> O Login predefinido é Login:<strong>admin</strong>
Password:<strong>admin</strong>. É aconselhável alterar a palavra chave assim que inicie o
equipamento por questões de segurança, para fazer isso clique no símbolo do utilizador dentro
do menu do Home Painel e de seguida em <strong>Change Password</strong>. </p>
      </div>
      <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Fechar</button>
```

```
      </div>
    </div>
  </div>
</div>

 <header>
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" href="index.php">ASSISTLORA</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="/" onclick="javascript:event.target.port=8123">Home Assistant
<i class="fas fa-home"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="config.php">Configurações <i class="fa fa-wrench"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="about.php">Sobre o projeto <i class="fas fa-archive"></i></a>
    </li>
    </ul>
    <ul class="nav navbar-nav navbar-right">
    <li><button type="button" style="width: 120px; margin:0px 10px; margin-top:8px;"
class="btn btn-primary" data-toggle="modal" data-target="#modalajuda"> Ajuda <i class="far fa-
life-ring"></i></button></li>
    </ul>
  </div>
  </nav>
</header>


    <main role="main">


<!-- ================================================== -->

      <div class="container marketing">
            <br>
            <h2 class="featurette-heading">AssistLora <span class="text-muted">Uma ajuda
preciosa.</span></h2>
            <p class="lead">Um sistema de Domotica sem fios baseado na tecnologia LoRa.


       <hr class="featurette-divider">

       <div class="row featurette">
          <div class="col-md-7">
            <h2 class="featurette-heading">Controlo Total<span class="text-muted"> sobre a sua
casa</span></h2>
            <p class="lead">
            AssistLora é um sistema de Domotica que integra o software Home Assistant
devenvolvido para funcionar com varios equipamentos sem fios.
            Estes equipamentos comunicam atraves de sinais wireless com a tecnologia LoRa que
permite um longo alcance e baixa potencia.
            Este projeto foi desenvolvido para trabalho final de curso de Mestrado.
            </p>

          </div>
          <div class="col-md-5">
            <img class="featurette-image img-fluid mx-auto" src="/img/imagem_projeto.jpg"
alt="imagem_projeto">
```

```
          </div>
        </div>

        <hr class="featurette-divider">

      </div>

    </main>
 <!-- ================================================== -->


  <!-- Footer -->
  <footer class="container">
    <p class="float-right"><a href="index.php">Página Inicial</a> &sdot; <a href="/"
onclick="javascript:event.target.port=8123">Home Assistant </a> &sdot; <a
href="config.php">Configurações</a> &sdot; <a href="about.php">Sobre o projeto</a></p>
    <p class="copyright text-muted small">Desenvolvido por: Rui Pedro Silva | 2018 | Porto -
Portugal </p>
  </footer>

    <!-- Bootstrap core JavaScript ================================================== -->
    <script src="./js/jquery-3.2.1.slim.min.js"></script>
    <script src="./js/bootstrap.min.js"></script>
  </body>
</html>
```

Código do Website – Pagina Configurações – config.php

```
<!--
------ projects.php ------
Paguina projetos
 -->

<!DOCTYPE html>
<html lang="pt-pt" xml:lang="pt-pt">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Pagina Pessoal">
    <meta name="author" content="Rui Pedro Silva">
    <link rel="shortcut icon" href="/img/icons/favicon.ico">
  <link rel="icon" type="image/png" href="/img/icons/favicon-48x48.png" sizes="48x48">
  <link rel="apple-touch-icon" sizes="128x128" href="/img/icons/apple-touch-icon-128x128.png">

    <title>AssistLora</title>

    <!-- Bootstrap Core CSS -->
    <link href="./css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="./css/projects_style.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <script defer src="./js/fontawesome-all.js"></script>
    <link href="./css/fontawesome-all.css" rel="stylesheet">
    <link
href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic"
rel="stylesheet" type="text/css">
</head>


  <body>

<!-- Modal Ajuda-->
<div class="modal fade" id="modalajuda" tabindex="-1" role="dialog" aria-
labelledby="modalajudaLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
    <h4 class="modal-title" id="modalajudaLabel">Precisa de Ajuda?</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
    <p><strong>Bem Vindo ao AssistLora!</strong></p>
    <p>Esta é a pagina inicial do AssistLora, nesta pagina pode seguir para o Painel Controlo
clicando no botão <strong>Home Painel</strong>. Caso pertenda alterar as definições ao sistema
clique em <strong>Configurações</strong> na barra do menu. Pode ver mais informação sobre este
projeto clicando em <strong>Sobre o Projeto</strong> no Menu.</p>
    <br>
    <p><strong>Problemas comuns:</strong> Não consigo aceder ao Home Painel!</p>
    <p>  -> O Painel de Controla demora algum tempo a iniciar, se no fim de 5 minutos ainda
nao tiver iniciado pode reiniciar o equipamento nas <strong>Configurações</strong>.</p>
    <p><strong>Problemas comuns:</strong> Não sei o login para entrar no Home Painel!</p>
    <p>  -> O Login predefinido é Login:<strong>admin</strong>
Password:<strong>admin</strong>. É aconselhável alterar a palavra chave assim que inicie o
equipamento por questões de segurança, para fazer isso clique no símbolo do utilizador dentro
do menu do Home Painel e de seguida em <strong>Change Password</strong>. </p>
      </div>
      <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Fechar</button>
      </div>
```

```
    </div>
  </div>
</div>

<!-- Modal rede-->
<div class="modal fade" id="Modalrede" tabindex="-1" role="dialog" aria-
labelledby="Modalrede_label">
  <div class="modal-dialog modal-sm" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title" id="Modalrede_label">Gravar Rede Wi-Fi</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
        <form id="form_gravar" method="post" action="<?php echo "{$_SERVER['PHP_SELF']}"; ?>">
          <br>
          <div class="form-group">
            <input id="rede_wifi_a_gravar" type="text" class="form-control" style="text-align:
center; width:100%; margin: 0 auto;" name="rede" placeholder="Nome da Rede">
          </div>
          <div class="form-group">
            <input type="password" class="form-control" style="text-align: center; width:100%;
margin: 0 auto;" name="psw" placeholder="Password">
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Cancelar</button>
        <button type="submit" class="btn btn-primary" form="form_gravar" name="gravar"
value="Gravar">Gravar</button>
      </div>
    </div>
  </div>
</div>


<!-- Modal mudar password assistlora-->
<div class="modal fade" id="Modalredeassistlora" tabindex="-1" role="dialog" aria-
labelledby="Modalredeassistlora_label">
  <div class="modal-dialog modal-sm" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title" id="Modalredeassistlora_label">Mudar Password da Rede</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
        <form id="form_gravar2" method="post" action="<?php echo "{$_SERVER['PHP_SELF']}";
?>">
          <br>
          <div class="form-group">
            <input type="password" class="form-control" style="text-align: center; width:100%;
margin: 0 auto;" name="psw" placeholder="Password">
          </div>
        </form>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">Cancelar</button>
        <button type="submit" class="btn btn-primary" form="form_gravar2" name="alterar"
value="Alterar">Alterar</button>
      </div>
    </div>
  </div>
</div>
```

```
<?php
$text=0;
    if ($_POST["gravar"]=="Gravar"){
      $myfile = fopen("/etc/wpa_supplicant/wpa_supplicant.conf", "a") or die("Unable to open
file!");
      $txt = "\nnetwork={\n";
      fwrite($myfile, $txt);
      $txt = "\t".'ssid="'.$_POST["rede"].'"'."\n";
      fwrite($myfile, $txt);
      $txt = "\t".'psk="'.$_POST["psw"].'"'."\n"."}";
      fwrite($myfile, $txt);
      fclose($myfile);
      $text=1;

    }elseif ($_POST["reiniciar"]=="Reiniciar"){
      $text=2;
    }elseif ($_POST["desligar"]=="Desligar"){
      $text=3;
    }elseif ($_POST["alterar"]=="Alterar"){

      $reading = fopen('/etc/hostapd/hostapd.conf', 'r');
      $writing = fopen('/etc/hostapd/hostapd_tmp.conf', 'w');

      $replaced = false;

      while (!feof($reading)) {
        $line = fgets($reading);
        $rest = substr($line, 0, 15);
        if (stristr($rest,"wpa_passphrase=")) {
          $line = "wpa_passphrase=".$_POST["psw"]."\n";
          $replaced = true;
        }
        #echo "<p>".$line."</p>";
        fputs($writing, $line);
      }
      fclose($reading);
      fclose($writing);
      // might as well not overwrite the file if we didn't replace anything
      if ($replaced) {
        rename('/etc/hostapd/hostapd_tmp.conf', '/etc/hostapd/hostapd.conf');
      } else {
        unlink('/etc/hostapd/hostapd_tmp.conf');
      }
      $text=4;

    }


?>

<header>
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" href="index.php">ASSISTLORA</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="/" onclick="javascript:event.target.port=8123">Home Assistant
<i class="fas fa-home"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="config.php">Configurações <i class="fa fa-wrench"></i></a>
```

```
    </li>
    <li class="nav-item">
      <a class="nav-link" href="about.php">Sobre o projeto <i class="fas fa-archive"></i></a>
    </li>
    </ul>
    <ul class="nav navbar-nav navbar-right">
    <li><button type="button" style="width: 120px; margin:0px 10px; margin-top:8px;"
class="btn btn-primary" data-toggle="modal" data-target="#modalajuda"> Ajuda <i class="far fa-
life-ring"></i></button></li>
    </ul>
  </div>
  </nav>
</header>

<!-- ============================================== -->


    <main role="main">
    <br><br><br><br>

          <?php
              if( $text==1 ){
                echo <<<END1
                    <br>
                    <div class="alert alert-success">
                      <a href="#" class="close" data-dismiss="alert">&times;</a>
                      <strong>Feito!</strong> Rede Wifi adicionada com sucesso. Para verificar
as alterações reinicie o Sistema.
                    </div>
END1;
              }elseif( $text==2 ){
                echo <<<END2
                    <br>
                    <div class="alert alert-warning">
                    <a href="#" class="close" data-dismiss="alert">&times;</a>
                      <strong>Feito!</strong> O sistema irá reiniciar dentro de momentos.
                    </div>
END2;
                exec('/var/www/html/phpreboot > /dev/null 2>/dev/null &');
              }elseif( $text==3 ){
                echo <<<END2
                    <br>
                    <div class="alert alert-warning">
                    <a href="#" class="close" data-dismiss="alert">&times;</a>
                      <strong>Feito!</strong> O sistema irá desligar dentro de momentos.
                    </div>
END2;
                exec('/var/www/html/phphalt > /dev/null 2>/dev/null &');
              }elseif( $text==4 ){
                echo <<<END2
                    <br>
                    <div class="alert alert-success">
                      <a href="#" class="close" data-dismiss="alert">&times;</a>
                      <strong>Feito!</strong> Foi alterada a password da Rede Wifi AssistLora.
Para verificar as alterações reinicie o Sistema.
                    </div>
END2;
              }
          ?>

      <br>
      <div style="text-align:center;">
        <h2>Configure o seu AssistLora</h2>
        <h2><span class="text-muted"> Nesta página poderá fazer configurações ao
sistema</span></h2>
        <p>Clique nos botões em baixo para proceder à configuração correspondente.</p>
```

```
        <p>(Algumas das definições estão disponiveis apenas dentro do Painel de Controlo.)</p>

      </div>

     <div class="row">
        <div class="col-sm-1"></div>
        <div class="col-sm-10">

        <br><br>

          <div style="text-align:center;">
            <button style="margin:10px 0px;" class="btn btn-primary btn-lg" type="button"
data-toggle="collapse" data-target="#collapsRedes" aria-expanded="false" aria-
controls="collapsRedes">
              Ver Redes Wifi disponiveis
            </button>
            <div class="collapse" id="collapsRedes" style="margin:10px 0px;">
              <div class="well" style="text-align:center;">
                <p><strong>IP do sistema: </strong></p>
                <p>
                <?php
                  exec('hostname -I', $output0);
                  echo $output0[0];
                ?>
                </p>
                <br><br>
                <p><strong>O Sistema está ligado à rede: </strong></p>
                <p>
                <?php
                  exec('iwgetid', $output1);
                  echo $output1[0];
                ?>
                </p>
                <br><br>
                <p><strong>Redes disponiveis:</strong></p>
                <p>
                <?php
                  exec("iwlist wlan1 scanning | egrep 'Encryption|Quality|ESSID'", $output2);
                  foreach ($output2 as $value) {
                    echo "$value <br>";
                  }
                ?>
                </p>
              </div>
            </div>

          </div>
          <br><br>

        <div class="col-sm-1"></div>


          <div style="text-align:center;">
            <button style="margin:10px 0px;" class="btn btn-primary btn-lg" type="button"
data-toggle="modal" data-target="#Modalrede" aria-expanded="false" aria-controls="Modalrede">
              Inserir nova rede Wifi
            </button>

          </div>
          <br><br>


          <div style="text-align:center;">
            <button style="margin:10px 0px;" class="btn btn-primary btn-lg" type="button"
data-toggle="modal" data-target="#Modalredeassistlora" aria-expanded="false" aria-
controls="Modalredeassistlora">
```

```
              Alterar Password da Rede AssistLora
            </button>

          </div>
          <br><br>

            <div style="text-align:center;">
            <button style="margin:10px 0px;" class="btn btn-primary btn-lg" type="button"
data-toggle="collapse" data-target="#collapsReset" aria-expanded="false" aria-
controls="collapseReset">
              Reiniciar Sistema
            </button>
            <div class="collapse" id="collapsReset" style="margin:10px 0px;">
              <div class="well" style="text-align:center;">
                <p>Tem a certesa que pertende reiniciar o serviço?</p>
                <p>Ao reiniciar o serviço deixarar de poder aceder ao sistema até iniciar
novamente.</p>
                <p>Alguns actuadores e sensores poderam deixar de funcionar até o reiniciar do
sistema esteja concluido.</p>
                <form method="post" action="<?php echo "{$_SERVER['PHP_SELF']}"; ?>">
                  <button class="btn btn-default" type="button" data-toggle="collapse" data-
target="#collapsReset" >Cancelar</button>
                  <button name='reiniciar' type='submit' value="Reiniciar" class="btn btn-
danger">Reiniciar!</button>
                </form>
              </div>
            </div>

          </div>
          <br><br>

          <div style="text-align:center;">
            <button style="margin:10px 0px;" class="btn btn-primary btn-lg" type="button"
data-toggle="collapse" data-target="#collapsDesligar" aria-expanded="false" aria-
controls="collapsDesligar">
              Desligar Sistema
            </button>
            <div class="collapse" id="collapsDesligar" style="margin:10px 0px;">
              <div class="well" style="text-align:center;">
                <p>Tem a certesa que pertende desligar o Sistema?</p>
                <p>Ao desligar o sistema todos os actuadores e sensores deixaram de
funcionar.</p>
                <p>Para voltar a ligar o Sistema alimente novamente a Central de Controlo.</p>
                <form method="post" action="<?php echo "{$_SERVER['PHP_SELF']}"; ?>">
                  <button class="btn btn-default" type="button" data-toggle="collapse" data-
target="#collapsDesligar" >Cancelar</button>
                  <button name='desligar' type='submit' value="Desligar" class="btn btn-
danger">Desligar!</button>
                </form>
              </div>
            </div>

          </div>
          <br><br>

        </div>
    </div>

      <br><br>
      <hr>

    </main>
  <!--============================================== -->


  <!-- Footer -->
```

```
  <footer class="container">
    <p class="float-right"><a href="index.php">Página Inicial</a> &sdot; <a href="/"
onclick="javascript:event.target.port=8123">Home Assistant </a> &sdot; <a
href="config.php">Configurações</a> &sdot; <a href="about.php">Sobre o projeto</a></p>
    <p class="copyright text-muted small">Desenvolvido por: Rui Pedro Silva | 2018 | Porto -
Portugal </p>
  </footer>

    <!-- Bootstrap core JavaScript ================================================== -->
    <script src="./js/jquery-3.2.1.slim.min.js"></script>
    <script src="./js/bootstrap.min.js"></script>
  </body>
</html>
```

Código do Website – Pagina de Erro – error.php

```
<!--
------  error.php --------
Mostra erros
 -->
 <!DOCTYPE html>
<html lang="pt-pt" xml:lang="pt-pt">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Pagina Pessoal">
    <meta name="author" content="Rui Pedro Silva">
    <link rel="shortcut icon" href="/img/icons/favicon.ico">
  <link rel="icon" type="image/png" href="/img/icons/favicon-48x48.png" sizes="48x48">
  <link rel="apple-touch-icon" sizes="128x128" href="/img/icons/apple-touch-icon-128x128.png">

    <title>AssistLora</title>

    <!-- Bootstrap Core CSS -->
    <link href="./css/bootstrap.min.css" rel="stylesheet">
    <!-- Custom CSS -->
    <link href="./css/projects_style.css" rel="stylesheet">

    <!-- Custom Fonts -->
    <script defer src="./js/fontawesome-all.js"></script>
    <link href="./css/fontawesome-all.css" rel="stylesheet">
    <link
href="http://fonts.googleapis.com/css?family=Lato:300,400,700,300italic,400italic,700italic"
rel="stylesheet" type="text/css">
</head>


  <body>

<!-- Modal Ajuda-->
<div class="modal fade" id="modalajuda" tabindex="-1" role="dialog" aria-
labelledby="modalajudaLabel">
  <div class="modal-dialog modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
    <h4 class="modal-title" id="modalajudaLabel">Precisa de Ajuda?</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span
aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body" style="float:center;">
    <p><strong>Bem Vindo ao AssistLora!</strong></p>
    <p>Esta é a pagina inicial do AssistLora, nesta pagina pode seguir para o Painel Controlo
clicando no botão <strong>Home Painel</strong>. Caso pertenda alterar as definições ao sistema
clique em <strong>Configurações</strong> na barra do menu. Pode ver mais informação sobre este
projeto clicando em <strong>Sobre o Projeto</strong> no Menu.</p>
    <br>
    <p><strong>Problemas comuns:</strong> Não consigo aceder ao Home Assistant!</p>
    <p>  -> O Painel de Controlo demora algum tempo a iniciar, se no fim de 5 minutos ainda
nao tiver iniciado pode reiniciar o equipamento nas <strong>Configurações</strong>.</p>
    <p><strong>Problemas comuns:</strong> Não sei o login para entrar no Home Assistant!</p>
    <p>  -> O Login predefinido é Login:<strong>admin</strong>
Password:<strong>admin</strong>. É aconselhável alterar a palavra chave assim que inicie o
equipamento por questões de segurança, para fazer isso clique no símbolo do utilizador dentro
do menu do Home Painel e de seguida em <strong>Change Password</strong>. </p>
      </div>
      <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Fechar</button>
      </div>
    </div>
```

```
  </div>
</div>

 <header>
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" href="index.php">ASSISTLORA</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarCollapse">
    <ul class="navbar-nav mr-auto">
    <li class="nav-item">
      <a class="nav-link" href="/" onclick="javascript:event.target.port=8123">Home Assistant
<i class="fas fa-home"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="config.php">Configurações <i class="fa fa-wrench"></i></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="about.php">Sobre o projeto <i class="fas fa-archive"></i></a>
    </li>
    </ul>
    <ul class="nav navbar-nav navbar-right">
    <li><button type="button" style="width: 120px; margin:0px 10px; margin-top:8px;"
class="btn btn-primary" data-toggle="modal" data-target="#modalajuda"> Ajuda <i class="far fa-
life-ring"></i></button></li>
    </ul>
  </div>
  </nav>
</header>

<!-- ================================================== -->


    <main role="main">
    <br><br>


  <div class="content-section-c">
    <div class="container">

<?php

    if ($_GET["erro"] != "") {
      $title = 'Ocoreu um Erro';
      $message = $_GET["erro"];
    }else{
      $status = $_SERVER['REDIRECT_STATUS'];
      $codes = array(
        403 => array('Erro 403 - Forbidden', 'O servidor recusou-se a satisfazer o seu
pedido.'),
        404 => array('Erro 404 - Not Found', 'O documento ou arquivo solicitado não foi
encontrado.'),
        405 => array('Erro 405 - Method Not Allowed', 'O método especificado no Request-Line
não é permitido para o recurso especificado.'),
        408 => array('Erro 408 - Request Timeout', 'O seu navegador não conseguiu enviar um
pedido no tempo permitido pelo servidor.'),
        500 => array('Erro 500 - Internal Server Error', 'O pedido não foi bem sucedido devido
a uma condição inesperada encontrada pelo servidor.'),
        502 => array('Erro 502 - Bad Gateway', 'O servidor recebeu uma resposta inválida do
server while ao tentar satisfazer o pedido.'),
        504 => array('Erro 504 - Gateway Timeout', 'O upstream server não conseguiu enviar um
pedido no tempo permitido pelo servidor.')
      );
```

```
      $title = $codes[$status][0];
      $message = $codes[$status][1];
      if ($title == false || strlen($status) != 3) {
        $title = 'Erro Desconhecido';
        $message = 'Ocorreu um Erro desconhecido no servidor, se o problema persistir reinicie
o sistema.';
      }
    }


    echo '<br><br><p><h1 style="text-align:center;">' . $title . '</h1></p>' .
      '<p style="text-align:center;">' . $message . '</p><br>' .
      '<p style="text-align:center; cursor:pointer" onclick="goBack()"><b> Voltar para a
página anterior</b></p>';

    ?>
    <br><br>
    </div>
    <div class="row">
      <div class="col-sm-4"></div>
      <div class="col-sm-4"><img class="featurette-image img-fluid mx-auto"
src="/img/imagem_erro.jpg" alt="Erro_imagem"></div>
      <div class="col-sm-4"></div>
    </div>


  </div>

  <script>
    function goBack() {
      window.history.back();
    }
  </script>

  <br><br><br>

    </main>
  <!--================================================= -->


  <!-- Footer -->
  <footer class="container">
    <p class="float-right"><a href="index.php">Página Inicial</a> &sdot; <a href="/"
onclick="javascript:event.target.port=8123">Home Assistant </a> &sdot; <a
href="config.php">Configurações</a> &sdot; <a href="about.php">Sobre o projeto</a></p>
    <p class="copyright text-muted small">Desenvolvido por: Rui Pedro Silva | 2018 | Porto -
Portugal </p>
  </footer>

    <!-- Bootstrap core JavaScript ================================================= -->
    <script src="./js/jquery-3.2.1.slim.min.js"></script>
    <script src="./js/bootstrap.min.js"></script>
  </body>
</html>
```

```
// This program is part of ASSISTLORA
//
// Copyright 2018 Rui Silva.
// This file is part of rpsreal/assistlora

#include <SPI.h>      // SPI
#include <RH_RF95.h>  //HadioHead http://www.airspayce.com/mikem/arduino/RadioHead/
#include <Base64.h> // https://github.com/adamvr/arduino-base64
#include <AESLib.h> // https://github.com/DavyLandman/AESLib

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

#define RF95_FREQ 434.0

RH_RF95 rf95(RFM95_CS, RFM95_INT);

// ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#define LED_ARDUINO 13 // built in led
#define LED 5 // led pwm
#define RELAY 4 // Pin of Relay
#define BUTTON 3 // change state button
#define LONG_RANGE_MODE 0   // Slow+long range mode =1
#define WAITING_TIME 2000 // Max waiting time for ACK (depende do modo LONG_RANGE_MODE se =0 -
> 2000)

char TP[2];
char ID[4];

void setup() {
  pinMode(LED_ARDUINO, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(RELAY, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  digitalWrite(RELAY, LOW);
  analogWrite(LED, 30);
  digitalWrite(RFM95_RST, HIGH);


  // ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  strcpy(TP, "1"); // device type 1=switch
  strcpy(ID, "001"); // device ID  (3 char) 001-999     medium_range=[001-099]
Long_range=[100-999]



  while (!Serial);
  Serial.begin(9600);
  delay(100);

  // manual reset
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
```

```
    while (1);
  }

  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }

  // Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on.
  // Slow+long range.
  if (LONG_RANGE_MODE==1) {
    rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096);
  }
  // Defaults after init are 434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on
  // Medium Range

  rf95.setTxPower(18);
  Serial.println("START");
}

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
uint8_t key[17] = "1234567890123456";
char pck_to_receive_ON[17]="";
char pck_to_receive_OFF[17]="";
char pck_to_receive_ACK[17]="";
char pck_to_send[17]="";

int RELAYState = LOW;
int buttonState;             // the current reading from the input pin
int lastButtonState = HIGH;   // the previous reading from the input pin
unsigned long lastDebounceTime = 0;
int wait_for_ack=LOW;
unsigned long send_ack_time = 0;
uint8_t send_count=0; // contagem de envios sem receber Ack


void loop(){
  uint8_t send_ack=0;
  uint8_t send_inf=0;

  if (rf95.available()){

    if (rf95.recv(buf, &len)){
      digitalWrite(LED_ARDUINO, HIGH);
      //RH_RF95::printBuffer("Got: ", buf, len);
      //Serial.print("RSSI: ");
      //Serial.println(rf95.lastRssi(), DEC);
      Serial.print("== RECEIVED:  ");
      Serial.print((char*)buf);

      uint8_t bufLen = sizeof(buf);
      uint8_t decodedLen = base64_dec_len((char*)buf, bufLen);
      uint8_t data_de[decodedLen];
      base64_decode((char*)data_de, (char*)buf, bufLen);

      aes128_dec_single(key, data_de);
      Serial.print("  |  Decoded: ");
      Serial.println((char*)data_de);

      digitalWrite(LED_ARDUINO, LOW);

      strcpy(pck_to_receive_ON, TP);
      strcat(pck_to_receive_ON, ID);
      strcat(pck_to_receive_ON, "02ON        ");// "111212ON        "
```

```
    strcpy(pck_to_receive_OFF, TP);
    strcat(pck_to_receive_OFF, ID);
    strcat(pck_to_receive_OFF, "03OFF       ");// "111213OFF       "

    strcpy(pck_to_receive_ACK, TP);
    strcat(pck_to_receive_ACK, ID);
    strcat(pck_to_receive_ACK, "03ACK       ");// "100003ACK       "


    if (strcmp(pck_to_receive_ON,((char*)data_de)) == 0){
      Serial.println("Receive ON - send ACK");
      digitalWrite(RELAY, HIGH);
      analogWrite(LED, 255);
      RELAYState=HIGH;
      send_ack=1;

    }else if (strcmp(pck_to_receive_OFF,((char*)data_de)) == 0){
      Serial.println("Receive OFF - send ACK");
      digitalWrite(RELAY, LOW);
      analogWrite(LED, 30);
      RELAYState=LOW;
      send_ack=1;
    }else if (strcmp(pck_to_receive_ACK,((char*)data_de)) == 0){
      Serial.println("Receive ACK");
      wait_for_ack=LOW;
      send_ack_time=0;
      send_count=0;
    }


    if (send_ack==1){
      digitalWrite(LED_ARDUINO, HIGH);

      strcpy(pck_to_send, "0"); // TPS
      strcat(pck_to_send, ID);
      strcat(pck_to_send, TP);
      strcat(pck_to_send, "3"); // LEN of mens
      strcat(pck_to_send, "ACK       "); //mens
      uint8_t input[17];
      memcpy(input, pck_to_send, 17);  // TPS+ID+TP+"3ACK       "

      aes128_enc_single(key, input);
      uint8_t  inputLen = sizeof(input)-1;
      uint8_t  encodedLen = base64_enc_len(inputLen);
      uint8_t  encoded[encodedLen+1];
      base64_encode((char*)encoded, (char*)input, inputLen);

      rf95.send(encoded, sizeof(encoded));
      rf95.waitPacketSent();
      Serial.print("== SEND: ");
      Serial.print(pck_to_send); // TPS+ID+TP+"3ACK       "
      Serial.print("  |  Encoded: ");
      Serial.println((char*)encoded);

      digitalWrite(LED_ARDUINO, LOW);
      send_ack=0;
    }
  }else{
    Serial.println("Receive failed");
  }

}


int reading = digitalRead(BUTTON);
```

```
  if (reading != lastButtonState) {
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > 50) {
    if (reading != buttonState) {
      buttonState = reading;

      //digitalWrite(RELAY, !buttonState);
      if (buttonState == LOW) {
        RELAYState=!RELAYState;
        digitalWrite(RELAY, RELAYState);
        send_inf=1; //enviar mens para a gateway (mudou de estado)
      }

    }
  }
  lastButtonState = reading;


  // ACK time out
  if( (wait_for_ack==HIGH) && ((millis() - send_ack_time) > WAITING_TIME) ){
    Serial.print(" | ACK TIME OUT | - Change again State");
    send_count++;
    if(send_count==2){ // já enviou 2 vezes sem obter resposta - dar sinal se erro
      Serial.print("Terceira tentativa de envio sem sucesso - Inverter estado - Dar sinal de
erro e continuar");
      RELAYState=!RELAYState;
      digitalWrite(RELAY, RELAYState);
      for(int i=0;i<5;i++){ // apenas 5 piscas
        analogWrite(LED, 255);
        delay(500);
        analogWrite(LED, 0);
        delay(500);
      }
      if(RELAYState==HIGH){
        analogWrite(LED, 255);
      }else{
        analogWrite(LED, 30);
      }
      wait_for_ack=LOW;
      send_ack_time=0;
      send_count=0;
    }else{ // enviar de novo
      //delay(2000);
      send_inf=1;
    }
  }


  if(send_inf==1){ //senviar mens para a gateway (mudou de estado)
    // contar o tempo, caso nao receba o ACK deste envio entao envia novamente
    // se mesmo assim nao receber o ACK entao dá erro
    // Enviar para a Gateway - TP=0 - ID deste device + MENS

    strcpy(pck_to_send, "0"); // TPS
    strcat(pck_to_send, ID);
    strcat(pck_to_send, TP);
    if(RELAYState==HIGH){
      analogWrite(LED, 255);
      strcat(pck_to_send, "2"); // LEN of mens
      strcat(pck_to_send, "ON        "); //mens
    }else{
      analogWrite(LED, 30);
      strcat(pck_to_send, "3"); // LEN of mens
      strcat(pck_to_send, "OFF       "); //mens
    }
```

```
    uint8_t input[17];
    memcpy(input, pck_to_send, 17);  // 0+ID+TP"2ON        "
    aes128_enc_single(key, input);
    uint8_t  inputLen = sizeof(input)-1;
    uint8_t  encodedLen = base64_enc_len(inputLen);
    uint8_t  encoded[encodedLen+1];
    base64_encode((char*)encoded, (char*)input, inputLen);

    rf95.send(encoded, sizeof(encoded));
    rf95.waitPacketSent();
    Serial.print("== SEND: ");
    Serial.print(pck_to_send); // 0+ID+TP"2ON        "
    Serial.print("  |  Encoded: ");
    Serial.println((char*)encoded);

    lastDebounceTime = millis();
    wait_for_ack=HIGH;
    send_ack_time=millis();
    send_inf=0;
  }
}
```

Código no microcontrolador – Dispositivo Atuador iluminação – light.ino

```
// This program is part of ASSISTLORA
//
// Copyright 2018 Rui Silva.
// This file is part of rpsreal/assistlora

#include <SPI.h>      // SPI
#include <RH_RF95.h>  //HadioHead http://www.airspayce.com/mikem/arduino/RadioHead/
#include <Base64.h> // https://github.com/adamvr/arduino-base64
#include <AESLib.h> // https://github.com/DavyLandman/AESLib

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

// Change to 434.0 or other frequency, must match RX's freq!
#define RF95_FREQ 434.0

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);

// ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#define LED_ARDUINO 13 // built in led
#define LED 5 // led pwm
#define RELAY 4 // Pin of Relay
#define BUTTON 3 // change state button
#define LONG_RANGE_MODE 0   // Slow+long range mode =1
#define WAITING_TIME 2000 // Max waiting time for ACK (depende do modo LONG_RANGE_MODE se =0 -
> 2000)

char TP[2];
char ID[4];

void setup() {
  pinMode(LED_ARDUINO, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(RELAY, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  digitalWrite(RELAY, LOW);
  analogWrite(LED, 30);
  digitalWrite(RFM95_RST, HIGH);


  // ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  strcpy(TP, "4"); // device type 4=light
  strcpy(ID, "001"); // device ID  (3 char) 001-999     medium_range=[001-099]
Long_range=[100-999]



  while (!Serial);
  Serial.begin(9600);
  delay(100);

  // manual reset
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);
```

```
  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
    while (1);
  }

  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }

  // Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on.
  // Slow+long range.
  if (LONG_RANGE_MODE==1) {
    rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096);
  }
  // Defaults after init are 434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on
  // Medium Range

  rf95.setTxPower(18);

  Serial.println("START");
}

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
uint8_t key[17] = "1234567890123456";
char pck_to_receive_ON[17]="";
char pck_to_receive_OFF[17]="";
char pck_to_receive_ACK[17]="";
char pck_to_send[17]="";

int RELAYState = LOW;
int buttonState;             // the current reading from the input pin
int lastButtonState = HIGH;   // the previous reading from the input pin
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled

int wait_for_ack=LOW;
unsigned long send_ack_time = 0;  // the last time the output pin was toggled
uint8_t send_count=0; // contagem de envios sem receber Ack


void loop(){
  uint8_t send_ack=0;
  uint8_t send_inf=0;

  if (rf95.available()){

    if (rf95.recv(buf, &len)){
      digitalWrite(LED_ARDUINO, HIGH);
      //RH_RF95::printBuffer("Got: ", buf, len);
      //Serial.print("RSSI: ");
      //Serial.println(rf95.lastRssi(), DEC);
      Serial.print("== RECEIVED:  ");
      Serial.print((char*)buf);

      uint8_t bufLen = sizeof(buf);
      uint8_t decodedLen = base64_dec_len((char*)buf, bufLen);
      uint8_t data_de[decodedLen];
      base64_decode((char*)data_de, (char*)buf, bufLen);

      aes128_dec_single(key, data_de);
      Serial.print("  |  Decoded: ");
      Serial.println((char*)data_de);

      digitalWrite(LED_ARDUINO, LOW);
```

```
    strcpy(pck_to_receive_ON, TP);
    strcat(pck_to_receive_ON, ID);
    strcat(pck_to_receive_ON, "02ON        ");// "111212ON        "

    strcpy(pck_to_receive_OFF, TP);
    strcat(pck_to_receive_OFF, ID);
    strcat(pck_to_receive_OFF, "03OFF       ");// "111213OFF       "

    strcpy(pck_to_receive_ACK, TP);
    strcat(pck_to_receive_ACK, ID);
    strcat(pck_to_receive_ACK, "03ACK       ");// "100003ACK       "


    if (strcmp(pck_to_receive_ON,((char*)data_de)) == 0){
      Serial.println("Receive ON - send ACK");
      digitalWrite(RELAY, HIGH);
      analogWrite(LED, 255);
      RELAYState=HIGH;
      send_ack=1;

    }else if (strcmp(pck_to_receive_OFF,((char*)data_de)) == 0){
      Serial.println("Receive OFF - send ACK");
      digitalWrite(RELAY, LOW);
      analogWrite(LED, 30);
      RELAYState=LOW;
      send_ack=1;

    }else if (strcmp(pck_to_receive_ACK,((char*)data_de)) == 0){
      Serial.println("Receive ACK");
      wait_for_ack=LOW;
      send_ack_time=0;
      send_count=0;
    }

  }else{
    Serial.println("Receive failed");
  }

}

if (send_ack==1){
  digitalWrite(LED_ARDUINO, HIGH);

  strcpy(pck_to_send, "0"); // TPS
  strcat(pck_to_send, ID);
  strcat(pck_to_send, TP);
  strcat(pck_to_send, "3"); // LEN of mens
  strcat(pck_to_send, "ACK       "); //mens
  uint8_t input[17];
  memcpy(input, pck_to_send, 17);  // TPS+ID+TP+"3ACK       "
  aes128_enc_single(key, input);
  uint8_t  inputLen = sizeof(input)-1;
  uint8_t  encodedLen = base64_enc_len(inputLen);
  uint8_t  encoded[encodedLen+1];
  base64_encode((char*)encoded, (char*)input, inputLen);

  rf95.send(encoded, sizeof(encoded));
  rf95.waitPacketSent();
  Serial.print("== SEND: ");
  Serial.print(pck_to_send); // TPS+ID+TP+"3ACK       "
  Serial.print("  |  Encoded: ");
  Serial.println((char*)encoded);

  digitalWrite(LED_ARDUINO, LOW);
  send_ack=0;
```

```
  }


  int reading = digitalRead(BUTTON);
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }
  if ((millis() - lastDebounceTime) > 50) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;
      // only toggle the LED if the new button state is HIGH
      if (buttonState == LOW) {
        RELAYState=!RELAYState;
        digitalWrite(RELAY, RELAYState);
        send_inf=1; //enviar mens para a gateway (mudou de estado)
      }
    }
  }
  lastButtonState = reading;


  // ACK time out
  if( (wait_for_ack==HIGH) && ((millis() - send_ack_time) > WAITING_TIME) ){
    Serial.print("Receive ACK TIME OUT (2s) - Change again State");
    send_count++;
    if(send_count==2){ // já enviou 2 vezes sem obter resposta - dar sinal se erro
      Serial.print("Terceira tentativa de envio sem sucesso - Inverter estado - Dar sinal de
erro e continuar");
      RELAYState=!RELAYState;
      digitalWrite(RELAY, RELAYState);
      for(int i=0;i<5;i++){ // apenas 5 piscas
        analogWrite(LED, 255);
        delay(500);
        analogWrite(LED, 0);
        delay(500);
      }
      if(RELAYState==HIGH){
        analogWrite(LED, 255);
      }else{
        analogWrite(LED, 30);
      }
      wait_for_ack=LOW;
      send_ack_time=0;
      send_count=0;
    }else{ // enviar de novo
      //delay(2000);
      send_inf=1;
    }
  }


  if(send_inf==1){ //senviar mens para a gateway (mudou de estado)
    // contar o tempo, caso nao receba o ACK deste envio entao envia novamente
    // se mesmo assim nao receber o ACK entao dá erro

    // Enviar para a Gateway - TP=0 - ID deste device + MENS
    strcpy(pck_to_send, "0"); // TPS
    strcat(pck_to_send, ID);
    strcat(pck_to_send, TP);
    if(RELAYState==HIGH){
      analogWrite(LED, 255);
```

```
      strcat(pck_to_send, "2"); // LEN of mens
      strcat(pck_to_send, "ON        "); //mens
    }else{
      analogWrite(LED, 30);
      strcat(pck_to_send, "3"); // LEN of mens
      strcat(pck_to_send, "OFF       "); //mens
    }
    uint8_t input[17];
    memcpy(input, pck_to_send, 17);  // 0+ID+TP"2ON        "
    aes128_enc_single(key, input);
    uint8_t  inputLen = sizeof(input)-1;
    uint8_t  encodedLen = base64_enc_len(inputLen);
    uint8_t  encoded[encodedLen+1];
    base64_encode((char*)encoded, (char*)input, inputLen);

    rf95.send(encoded, sizeof(encoded));
    rf95.waitPacketSent();
    Serial.print("== SEND: ");
    Serial.print(pck_to_send); // 0+ID+TP"2ON        "
    Serial.print("  |  Encoded: ");
    Serial.println((char*)encoded);
    lastDebounceTime = millis();

    wait_for_ack=HIGH;
    send_ack_time=millis();
    send_inf=0;
  }


}
```

Código no microcontrolador – Dispositivo Sensor analógico – sensor.ino

```
// This program is part of ASSISTLORA
//
// Copyright 2018 Rui Silva.
// This file is part of rpsreal/assistlora
// sensor.ino - Control sensors analog

#include <SPI.h>      // SPI
#include <RH_RF95.h>  //HadioHead http://www.airspayce.com/mikem/arduino/RadioHead/
#include <Base64.h> // https://github.com/adamvr/arduino-base64
#include <AESLib.h> // https://github.com/DavyLandman/AESLib

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

// radio frequency
#define RF95_FREQ 434.0

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);


// ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#define LED 4 // error led
#define SENSOR A0 // Pin of SENSOR (ANALOG)
#define LONG_RANGE_MODE 0   // Slow+long range mode =1
#define WAITING_TIME 2000 // Max waiting time for ACK (depende do modo LONG_RANGE_MODE se =0 -
> 2000)
// Parametros MAP   map(sensorValue, 0, 1023, 0, 255);
#define MAP1 0
#define MAP2 1023
#define MAP3 0
#define MAP4 100
#define SIG_Difference 10 // Value significant difference to send (send only if there is 100
difference in OUTPUTValue for example)


char TP[2];
char ID[4];

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  digitalWrite(RFM95_RST, HIGH);


// ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  strcpy(TP, "3"); // device type 3=sensor
  strcpy(ID, "001"); // device ID  (3 char) 001-999     medium_range=[001-099]
Long_range=[100-999]



  while (!Serial);
  Serial.begin(9600);
  delay(100);

  // manual reset
  digitalWrite(RFM95_RST, LOW);
```

```
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
    while (1);
  }

  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }


  // Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on.
  // Slow+long range.
  if (LONG_RANGE_MODE==1) {
    rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096);
  }
  // Defaults after init are 434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on
  // Medium Range

  rf95.setTxPower(18);
  Serial.println("START");
}

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
uint8_t key[17] = "1234567890123456";
char pck_to_receive_ACK[17]="";
char pck_to_send[17]="";


int SENSORValue = 0; //analogRead(SENSOR); // Value on pin
int OUTPUTValue = map(SENSORValue, MAP1, MAP2, MAP3, MAP4); // Value to send
int LastSend_OUTPUTValue = OUTPUTValue; // Ultimo valor enviado do sensor


unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled

int wait_for_ack=LOW;
unsigned long send_ack_time = 0;  // the last time the output pin was toggled
uint8_t send_count=0; // contagem de envios sem receber Ack



void loop(){
  uint8_t send_ack=0;
  uint8_t send_inf=0;

  if (rf95.available()){

    if (rf95.recv(buf, &len)){
      //RH_RF95::printBuffer("Got: ", buf, len);
      //Serial.print("RSSI: ");
      //Serial.println(rf95.lastRssi(), DEC);
      Serial.print("== RECEIVED:  ");
      Serial.print((char*)buf);

      uint8_t bufLen = sizeof(buf);
      uint8_t decodedLen = base64_dec_len((char*)buf, bufLen);
      uint8_t data_de[decodedLen];
      base64_decode((char*)data_de, (char*)buf, bufLen);
```

```
      aes128_dec_single(key, data_de);
      Serial.print("  |  Decoded: ");
      Serial.println((char*)data_de);

      strcpy(pck_to_receive_ACK, TP);
      strcat(pck_to_receive_ACK, ID);
      strcat(pck_to_receive_ACK, "03ACK       ");// "300503ACK       "


      if (strcmp(pck_to_receive_ACK,((char*)data_de)) == 0){
        Serial.println("Receive ACK");
        wait_for_ack=LOW;
        send_ack_time=0;
        send_count=0;
      }

    }else{
      Serial.println("Receive failed");
    }

  }

  int reading = map(analogRead(SENSOR), MAP1, MAP2, MAP3, MAP4);
  if (abs((reading-LastSend_OUTPUTValue))< SIG_Difference) { // se a leitura tiver uma
diferença menor que a SIG_Difference em relação ao LastSend_OUTPUTValue
    lastDebounceTime = millis(); // reset the debouncing timer
  }
  if ((millis() - lastDebounceTime) > 50) { //O valor lido já está à mais de 50ms com uma
diferença maior que o SIG_Difference
    Serial.print("O valor lido já está à mais de 50ms com uma diferença maior que o
SIG_Difference   ");
    SENSORValue = analogRead(SENSOR);
    OUTPUTValue = map(SENSORValue, MAP1, MAP2, MAP3, MAP4);
    LastSend_OUTPUTValue=OUTPUTValue;

    send_inf=1; //enviar mens para a gateway (mudou de estado)
  }



  if( (wait_for_ack==HIGH) && ((millis() - send_ack_time) > WAITING_TIME) ){ // ACK time out
    Serial.print(" | ACK TIME OUT | - Send mens again");
    send_count++;
    if(send_count==2){ // já enviou 2 vezes sem obter resposta - dar sinal se erro
      Serial.print("Terceira tentativa de envio sem sucesso - Dar sinal de erro");
      for(int i=0;i<120;i++){
        digitalWrite(LED, HIGH);
        delay(500);
        digitalWrite(LED, LOW);
        delay(500);
      }
      //wait_for_ack=LOW;
      //send_ack_time=0;
      // tentar novamente o envio passado 1min
      send_count=0;
      send_inf=1;
    }else{ // enviar de novo
      send_inf=1;
    }
  }


  if(send_inf==1){ //enviar mens para a gateway (mudou de estado)
    // contar o tempo, caso nao receba o ACK deste envio entao envia novamente
    // se mesmo assim nao receber o ACK entao dá erro
    Serial.print("Sensor = ");
```

```
    Serial.print(SENSORValue);
    Serial.print("\t Output = ");
    Serial.println(OUTPUTValue);
    // Enviar para a Gateway - TP=0 - ID deste device + MENS

    strcpy(pck_to_send, "0"); // TPS
    strcat(pck_to_send, ID);
    strcat(pck_to_send, TP);
    char buffer[10];
    itoa(OUTPUTValue,buffer,10);
    int len = strlen(buffer);
    char buffer2[1];
    itoa(len,buffer2,10);
    strcat(pck_to_send, buffer2); // LEN of mens
    strcat(pck_to_send, buffer); //mens
    for(int i=0; i<(10-len);i++){// completar a mens com espaços brancos (10 char)
      strcat(pck_to_send, " ");
    }


    uint8_t input[17];
    memcpy(input, pck_to_send, 17);  // TPS+ID+TP+"3ACK       "  000523ACK       "
    aes128_enc_single(key, input);
    uint8_t  inputLen = sizeof(input)-1;
    uint8_t  encodedLen = base64_enc_len(inputLen);
    uint8_t  encoded[encodedLen+1];
    base64_encode((char*)encoded, (char*)input, inputLen);

    rf95.send(encoded, sizeof(encoded));
    rf95.waitPacketSent();
    Serial.print("== SEND: ");
    Serial.print(pck_to_send); // TPS+ID+TP+"3ACK       "  000523ACK       ""
    Serial.print("  |  Encoded: ");
    Serial.println((char*)encoded);
    lastDebounceTime = millis();

    wait_for_ack=HIGH;
    send_ack_time=millis();
    send_inf=0;
  }
}
```

Código no microcontrolador – Dispositivo Sensor binario – binary_sensor.ino

```
// This program is part of ASSISTLORA
//
// Copyright 2018 Rui Silva.
// This file is part of rpsreal/assistlora
// binary_sensor.ino control binary sensors

#include <SPI.h>      // SPI
#include <RH_RF95.h>  //HadioHead http://www.airspayce.com/mikem/arduino/RadioHead/
#include <Base64.h> // https://github.com/adamvr/arduino-base64
#include <AESLib.h> // https://github.com/DavyLandman/AESLib

#define RFM95_CS 10
#define RFM95_RST 9
#define RFM95_INT 2

// Change to 434.0 or other frequency, must match RX's freq!
#define RF95_FREQ 434.0

// Singleton instance of the radio driver
RH_RF95 rf95(RFM95_CS, RFM95_INT);

// ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
#define LED 4 // error led
#define SENSOR 3 // Pin of SENSOR
#define LONG_RANGE_MODE 0   // Slow+long range mode =1
#define WAITING_TIME 2000 // Max waiting time for ACK (depende do modo LONG_RANGE_MODE se =0 -
> 2000)


char TP[2];
char ID[4];

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(RFM95_RST, OUTPUT);
  pinMode(SENSOR, INPUT_PULLUP);
  digitalWrite(RFM95_RST, HIGH);


  // ALTERAR AQUI CONFORME O DEVICE
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  strcpy(TP, "2"); // device type 2=binary_sensor
  strcpy(ID, "001"); // device ID  (3 char) 000-999



  while (!Serial);
  Serial.begin(9600);
  delay(100);

  // manual reset
  digitalWrite(RFM95_RST, LOW);
  delay(10);
  digitalWrite(RFM95_RST, HIGH);
  delay(10);

  while (!rf95.init()) {
    Serial.println("LoRa radio init failed");
    while (1);
  }
```

```
  // Defaults after init are 434.0MHz, modulation GFSK_Rb250Fd250, +13dbM
  if (!rf95.setFrequency(RF95_FREQ)) {
    Serial.println("setFrequency failed");
    while (1);
  }

  // Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on.
  // Slow+long range.
  if (LONG_RANGE_MODE==1) {
    rf95.setModemConfig(RH_RF95::Bw125Cr48Sf4096);
  }
  // Defaults after init are 434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on
  // Medium Range

  rf95.setTxPower(18);
  Serial.println("START");
}

uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
uint8_t len = sizeof(buf);
uint8_t key[17] = "1234567890123456";
char pck_to_receive_ACK[17]="";
char pck_to_send[17]="";

int SENSORState = LOW; // HIGH State on pin
int buttonState;             // the current reading from the input pin
int lastButtonState = LOW;   // the previous reading from the input pin
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled

int wait_for_ack=LOW;
unsigned long send_ack_time = 0;  // the last time the output pin was toggled
uint8_t send_count=0; // contagem de envios sem receber Ack


void loop(){
  uint8_t send_ack=0;
  uint8_t send_inf=0;

  if (rf95.available()){

    if (rf95.recv(buf, &len)){

      //RH_RF95::printBuffer("Got: ", buf, len);
      //Serial.print("RSSI: ");
      //Serial.println(rf95.lastRssi(), DEC);
      Serial.print("== RECEIVED:  ");
      Serial.print((char*)buf);

      uint8_t bufLen = sizeof(buf);
      uint8_t decodedLen = base64_dec_len((char*)buf, bufLen);
      uint8_t data_de[decodedLen];
      base64_decode((char*)data_de, (char*)buf, bufLen);

      aes128_dec_single(key, data_de);
      Serial.print("  |  Decoded: ");
      Serial.println((char*)data_de);

      strcpy(pck_to_receive_ACK, TP);
      strcat(pck_to_receive_ACK, ID);
      strcat(pck_to_receive_ACK, "03ACK       ");// "200503ACK       "


      if (strcmp(pck_to_receive_ACK,((char*)data_de)) == 0){
        Serial.println("Receive ACK");
        wait_for_ack=LOW;
```

```
      send_ack_time=0;
      send_count=0;
    }

  }else{
    Serial.println("Receive failed");
  }

}


int reading = digitalRead(SENSOR);
if (reading != lastButtonState) {
  // reset the debouncing timer
  lastDebounceTime = millis();
}
if ((millis() - lastDebounceTime) > 50) {
  // whatever the reading is at, it's been there for longer than the debounce
  // delay, so take it as the actual current state:

  // if the button state has changed:
  if (reading != buttonState) {
    buttonState = reading;

    SENSORState=!reading; // contrario da leitura
    send_inf=1; //enviar mens para a gateway (mudou de estado)

  }
}
lastButtonState = reading;


// ACK time out
if( (wait_for_ack==HIGH) && ((millis() - send_ack_time) > WAITING_TIME) ){
  Serial.print("Receive ACK TIME OUT (2s) - Send mens again");
  send_count++;
  if(send_count==2){ // já enviou 2 vezes sem obter resposta - dar sinal se erro
    Serial.print("Terceira tentativa de envio sem sucesso - Dar sinal de erro");
    for(int i=0;i<120;i++){
      digitalWrite(LED, HIGH);
      delay(500);
      digitalWrite(LED, LOW);
      delay(500);
    }
    //wait_for_ack=LOW;
    //send_ack_time=0;
    // tentar novamente o envio passado 1min
    send_count=0;
    send_inf=1;
  }else{ // enviar de novo
    //delay(2000);
    send_inf=1;
  }
}


if(send_inf==1){ //enviar mens para a gateway (mudou de estado)
  // contar o tempo, caso nao receba o ACK deste envio entao envia novamente
  // se mesmo assim nao receber o ACK entao dá erro


  // Enviar para a Gateway - TP=0 - ID deste device + MENS
  strcpy(pck_to_send, "0"); // TPS
  strcat(pck_to_send, ID);
  strcat(pck_to_send, TP);
  strcat(pck_to_send, "1"); // LEN of mens
```

```
    if(SENSORState==HIGH){
      strcat(pck_to_send, "1         "); //mens
    }else{
      strcat(pck_to_send, "0         "); //mens
    }
    uint8_t input[17];
    memcpy(input, pck_to_send, 17);  // TPS+ID+TP+"3ACK       "  000523ACK       "
    aes128_enc_single(key, input);
    uint8_t  inputLen = sizeof(input)-1;
    uint8_t  encodedLen = base64_enc_len(inputLen);
    uint8_t  encoded[encodedLen+1];
    base64_encode((char*)encoded, (char*)input, inputLen);

    rf95.send(encoded, sizeof(encoded));
    rf95.waitPacketSent();
    Serial.print("== SEND: ");
    Serial.print(pck_to_send); // TPS+ID+TP+"3ACK       "  000523ACK       ""
    Serial.print("  |  Encoded: ");
    Serial.println((char*)encoded);
    lastDebounceTime = millis();

    wait_for_ack=HIGH;
    send_ack_time=millis();
    send_inf=0;
  }
}
```

Código no Home Assistant – Plataforma Assistant – assistlora.py

```
"""
 This program is part of ASSISTLORA

 Copyright 2018 Rui Silva.
 This file is part of rpsreal/assistlora
 ASSISTLORA Devices Control
"""
import logging

import voluptuous as vol

from homeassistant.const import (
EVENT_HOMEASSISTANT_START, EVENT_HOMEASSISTANT_STOP)
from homeassistant.helpers import config_validation as cv


import time, base64
import RPi.GPIO as GPIO                                    # RPi.GPIO==0.6.3
from Crypto.Cipher import AES                              # pycrypto==2.6.1 and
pycryptodome==3.6.1

# spidev==3.2 and pyLoRa==0.2.2
from SX127x.constants import add_lookup, MODE, BW, CODING_RATE, GAIN, PA_SELECT, PA_RAMP,
MASK, REG
from SX127x.LoRa import set_bit, getter, setter, LoRa, LoRa2
from SX127x.board_config import BOARD, BOARD2

GPIO.setwarnings(False)


_LOGGER = logging.getLogger(__name__)

REQUIREMENTS =
['pyLoRa==0.2.2','spidev==3.2','RPi.GPIO==0.6.3','pycrypto==2.6.1','pycryptodome==3.6.4']
DOMAIN = 'assistlora'

def setup(hass, config):
    """Set up AssistLora component."""
    log = logging.getLogger(__name__)

    class ra_02_board1(LoRa): # lora_medium_range - IDs =[001-099]
        def __init__(self, verbose=False):
            super().__init__(verbose)
             #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.AES_KEY = '1234567890123456' # Key to ecrypt and decrypt
            self.wait_for_send=0
            self.RX_BUFFER=''

        def start(self):
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT)


        def on_rx_done(self): # activado por hardware atraves do pino dio0 da board
            #recebeu
            print("RxDone=1")
            self.clear_irq_flags(RxDone=1) #limpar flag
            payload = self.read_payload(nocheck=True)
            decrypt_mens=self.decrypt_payload(payload) # descodifica a mens
            self.RX_BUFFER=decrypt_mens
              #meter a receber logo
            self.set_mode(MODE.SLEEP)
```

```
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode

            self.process_received_mens(decrypt_mens) # Processar mens recebida



        def on_tx_done(self):
            #já enviou
            print("TxDone=1")
            self.clear_irq_flags(TxDone=1) #limpar flag
            self.wait_for_send=1
            #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode

        def ecrypt_mens(self, mens):
            """ mens (16 char) --> ecrypt_payload """
            print("ecrypt_mens -", mens)

            cipher = AES.new(self.AES_KEY)
            encoded = base64.b64encode(cipher.encrypt(mens))
            ecrypt_payload=list(encoded)
            ecrypt_payload.insert(0,0)
            ecrypt_payload.insert(0,0)
            ecrypt_payload.insert(0,255)
            ecrypt_payload.insert(0,255)
            ecrypt_payload.append(0)

            return ecrypt_payload

        def decrypt_payload(self, payload):
            """ payload lido  -->  decrypt_mens """
            print("decrypt_payload -", payload)

            mens=payload[4:-1] #to discard \xff\xff\x00\x00 and \x00 at the end
            mens=bytes(mens).decode("utf-8",'ignore')
            cipher = AES.new(self.AES_KEY)
            decodemens=base64.b64decode(mens)
            decrypt_mens = cipher.decrypt(decodemens)
            decrypt_mens = bytes(decrypt_mens).decode("utf-8",'ignore')

            return decrypt_mens

        def send_ack(self, pck_to_send):
            """ pck_to_send (16 char) """
            #meter a enviar
            #erro=0
            #self.set_dio_mapping([1,0,0,0,0,0])
            print("SEND_MENS MEDIUM RANGE")
            BOARD.led_off()
            ecrypt_payload=self.ecrypt_mens(pck_to_send)
            #ecrypt_payload=self.ecrypt_mens('INF             ')
            self.write_payload(ecrypt_payload)
            self.set_mode(MODE.TX)
            BOARD.led_on()
            time.sleep(2) # esperar que envie
            # NAO PERMITE DISPARAR INTERRUPÇÂO on_tx_done dentro da interrupção on_rx_done
            #start_time = time.time()
            #while ((self.wait_for_send==0) and (time.time() - start_time < 10)): # wait until
receive data or 10s
            #    pass;
            #if (time.time() - start_time > 10):
```

```
            #    print("Demorou mais de 2s -" , time.time() - start_time)
            #    erro=1
            #else:
            #    self.wait_for_send=0

            #if (erro==0): #esperar pelo ack
            #    return True
            #else: #houve um erro
            #    return False

            #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode


            return True

        def send_mens(self, pck_to_send, pck_to_receive):
            """ enviar mensagem """
            print("SEND_MENS MEDIUM RANGE")
            BOARD.led_off()
            #meter a enviar
            erro=0
            self.set_dio_mapping([1,0,0,0,0,0])

            ecrypt_payload=self.ecrypt_mens(pck_to_send)
            #ecrypt_payload=self.ecrypt_mens('INF             ')
            self.write_payload(ecrypt_payload)
            self.set_mode(MODE.TX)

            start_time = time.time()
            while ((self.wait_for_send==0) and (time.time() - start_time < 2)): # wait until
receive data or 2s
                pass;
            if (time.time() - start_time > 2):
                print("Demorou mais de 2s -" , time.time() - start_time)
                hass.components.persistent_notification.async_create("""
                Ocorreu um erro ao tentar enviar uma mensagem.
                Tente novamente. Se o problema persistir reinicie o equipamento.
                """, 'AssistLora ERROR')
                erro=1
            else:
                self.wait_for_send=0

            BOARD.led_on()
            if (erro==0): #esperar pelo ack
                start_time = time.time()
                while ((self.RX_BUFFER!=pck_to_receive) and (time.time() - start_time < 3)): #
wait until receive ACK or 3s
                    pass;
                if (time.time() - start_time > 3):
                    print("Demorou mais de 3s a receber -", time.time() - start_time)
                    tipo=pck_to_send[0:1]
                    if(tipo=='1'):
                        tipo='Switch'
                    elif(tipo=='2'):
                        tipo='Binary Sensor'
                    elif(tipo=='3'):
                        tipo='Sensor'
                    elif(tipo=='4'):
                        tipo='Light'

                    hass.components.persistent_notification.async_create("""
                    Não houve comfirmação de recepção da mensagem pelo dispositivo.
```

```
                    O dispositivo """+tipo+""" com ID-"""+pck_to_send[1:4]+""" existe?
                    Verifique se a ligação do dispositivo e se a sua configuração estão
corretas.
                    """, 'AssistLora ERROR')
                    return False
                else: #recebeu o ack
                    print("Demorou a receber o ack -", time.time() - start_time)
                    self.RX_BUFFER='' #limpar buffer
                    return True
            else: #houve um erro
                return False

        def process_received_mens(self, RX_BUFFER):
            TP=RX_BUFFER[0:1]
            ID=RX_BUFFER[1:4] #retirar do buffer o ID
            TPS=RX_BUFFER[4:5]
            LEN=RX_BUFFER[5:6]
            MENS=RX_BUFFER[6:]
            print('Process received mens medium range-> RX=', RX_BUFFER, "; TP=", TP, "; ID=",
ID, "; TPS=", TPS, "; LEN=", LEN,"; MENS=", MENS)
            if(int(ID)>=100 and int(ID)<=999): # Se o ID estiver fora do intervalo
lora_medium_range - IDs =[001-099] dá erro
                print("O elemento "+str(int(ID))+ " nao é medium range")
                hass.components.persistent_notification.async_create("""
                O elemento com o ID-"""+str(int(ID))+" é de longo alcance, no entanto foi
configurado como medio alcance."+"""
                Altere a configuração no dispositivo.
                """, 'AssistLora ERROR')
            elif(TP=='0'): #MENS para a gateway
                if(TPS=='1'): #MENS para Switch
                    if(MENS=='ON        '):
                        print("Mudar State para ON do switch."+str(int(ID)))
                        states = hass.states.get('switch.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to switch
                            if(enviou==True):
                                print("State ON")
                                #print(hass.states.get('switch.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('switch.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para switch."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Switch com
o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento switch."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Switch com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')
                    elif(MENS=='OFF       '):
                        print("Mudar State para OFF do switch."+str(int(ID)))
                        states = hass.states.get('switch.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to switch
                            if(enviou==True):
                                print("State OFF")
                                #print(hass.states.get('switch.'+str(int(ID))))
```

```
                                atts = states.attributes.copy()
                                hass.states.set('switch.'+str(int(ID)), "off", atts)
                            else:
                                print("Erro a enviar ACK para switch."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Switch com
o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento switch."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Switch com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                if(TPS=='2'): #MENS para Binary_Sensor
                    if(MENS=='1         '):
                        print("Mudar State para 1 (HIGH) do binary_sensor."+str(int(ID)))
                        states = hass.states.get('binary_sensor.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -200503ACK
- send ack to binary sensor
                            if(enviou==True):
                                print("State 1")
                                #print(hass.states.get('switch.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('binary_sensor.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para binary_sensor."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Binary
Sensor com o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento binary_sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Binary Sensor com ID-
"""+str(int(ID))+""" no entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                    elif(MENS=='0         '):
                        print("Mudar State para 0 (LOW) do binary_sensor."+str(int(ID)))
                        states = hass.states.get('binary_sensor.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -200503ACK
- ssend ack to binary sensor
                            if(enviou==True):
                                print("State OFF")
                                #print(hass.states.get('switch.'+str(int(ID))))

                                atts = states.attributes.copy()
                                hass.states.set('binary_sensor.'+str(int(ID)), "off", atts)
                            else:
                                print("Erro a enviar ACK para binary_sensor."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Binary
Sensor com o ID-"""+str(int(ID))+"""
```

```
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento binary_sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Binary Sensor com ID-
"""+str(int(ID))+""" no entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                if(TPS=='3'): #MENS para Sensor
                    print("Actualizar do sensor."+str(int(ID)))
                    data=MENS[:int(LEN)]
                    states = hass.states.get('sensor.'+str(int(ID)))
                    if(states!=None):
                        enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -300503ACK        -
send ack to sensor
                        if(enviou==True):
                            print("Update State")
                            atts = states.attributes.copy()
                            hass.states.set('sensor.'+str(int(ID)), data, atts)
                        else:
                            print("Erro a enviar ACK para sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Ocorreu um erro ao tentar enviar uma mensagem para Sensor com o
ID-"""+str(int(ID))+"""
                            Tente novamente. Se o problema persistir reinicie o equipamento.
                            """, 'AssistLora ERROR')
                    else:
                        print("Nao existe o elemento sensor."+str(int(ID)))
                        hass.components.persistent_notification.async_create("""
                        Foi recebida uma mensagem do Sensor com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                        Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                        """, 'AssistLora ERROR')

                if(TPS=='4'): #MENS para Light
                    if(MENS=='ON        '):
                        print("Mudar State para ON do light."+str(int(ID)))
                        states = hass.states.get('light.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to light
                            if(enviou==True):
                                print("State ON")
                                #print(hass.states.get('light.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('light.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para light."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Light com o
ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento light."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Light com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
```

```
                            """, 'AssistLora ERROR')

                    elif(MENS=='OFF       '):
                        print("Mudar State para OFF do light."+str(int(ID)))
                        states = hass.states.get('light.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to light
                            if(enviou==True):
                                print("State OFF")
                                #print(hass.states.get('light.'+str(int(ID))))

                                atts = states.attributes.copy()
                                hass.states.set('light.'+str(int(ID)), "off", atts)
                            else:
                                print("Erro a enviar ACK para light."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Light com o
ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento light."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Light com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')


    BOARD.setup()
    BOARD.reset()
    lora_medium_range = ra_02_board1(verbose=False)
    #  Medium Range   434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf = 128chips/symbol, CRC on 13 dBm
    lora_medium_range.set_pa_config(pa_select=1)
    assert(lora_medium_range.get_agc_auto_on() == 1)
    lora_medium_range.start() # inicia ra-02  para escuta



    class ra_02_board2(LoRa2): # lora_long_range - IDs =[100-999]
        def __init__(self, verbose=False):
            super().__init__(verbose)
             #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.AES_KEY = '1234567890123456' # Key to ecrypt and decrypt
            self.wait_for_send=0
            self.RX_BUFFER=''

        def start(self):
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT)


        def on_rx_done(self): # activado por hardware atraves do pino dio0 da board
            #recebeu
            print("RxDone=1")
            self.clear_irq_flags(RxDone=1) #limpar flag
            payload = self.read_payload(nocheck=True)
            decrypt_mens=self.decrypt_payload(payload) # descodifica a mens
            self.RX_BUFFER=decrypt_mens
              #meter a receber logo
```

```
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode

            self.process_received_mens(decrypt_mens) # Processar mens recebida



        def on_tx_done(self):
            #já enviou
            print("TxDone=1")
            self.clear_irq_flags(TxDone=1) #limpar flag
            self.wait_for_send=1
            #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode

        def ecrypt_mens(self, mens):
            """ mens (16 char) --> ecrypt_payload """
            print("ecrypt_mens -", mens)

            cipher = AES.new(self.AES_KEY)
            encoded = base64.b64encode(cipher.encrypt(mens))
            ecrypt_payload=list(encoded)
            ecrypt_payload.insert(0,0)
            ecrypt_payload.insert(0,0)
            ecrypt_payload.insert(0,255)
            ecrypt_payload.insert(0,255)
            ecrypt_payload.append(0)

            return ecrypt_payload

        def decrypt_payload(self, payload):
            """ payload lido  -->  decrypt_mens """
            print("decrypt_payload -", payload)

            mens=payload[4:-1] #to discard \xff\xff\x00\x00 and \x00 at the end
            mens=bytes(mens).decode("utf-8",'ignore')
            cipher = AES.new(self.AES_KEY)
            decodemens=base64.b64decode(mens)
            decrypt_mens = cipher.decrypt(decodemens)
            decrypt_mens = bytes(decrypt_mens).decode("utf-8",'ignore')

            return decrypt_mens

        def send_ack(self, pck_to_send):
            """ pck_to_send (16 char) """
            #meter a enviar
            #erro=0
            #self.set_dio_mapping([1,0,0,0,0,0])
            print("SEND_ACK LONG RANGE")
            BOARD2.led_off()
            ecrypt_payload=self.ecrypt_mens(pck_to_send)
            #ecrypt_payload=self.ecrypt_mens('INF             ')
            self.write_payload(ecrypt_payload)
            self.set_mode(MODE.TX)
            BOARD2.led_on()
            time.sleep(4) # esperar que envie
            # NAO PERMITE DISPARAR INTERRUPÇÂO on_tx_done dentro da interrupção on_rx_done
            #start_time = time.time()
            #while ((self.wait_for_send==0) and (time.time() - start_time < 10)): # wait until
receive data or 10s
            #    pass;
```

```
            #if (time.time() - start_time > 10):
            #    print("Demorou mais de 2s -" , time.time() - start_time)
            #    erro=1
            #else:
            #    self.wait_for_send=0

            #if (erro==0): #esperar pelo ack
            #    return True
            #else: #houve um erro
            #    return False

            #meter a receber logo
            self.set_mode(MODE.SLEEP)
            self.set_dio_mapping([0,0,0,0,0,0])
            self.reset_ptr_rx()
            self.set_mode(MODE.RXCONT) # Receiver mode

            return True

        def send_mens(self, pck_to_send, pck_to_receive):
            """ pck_to_send (16 char) """
            print("SEND_MENS LONG RANGE")
            BOARD2.led_off()
            #meter a enviar
            erro=0
            self.set_dio_mapping([1,0,0,0,0,0])

            ecrypt_payload=self.ecrypt_mens(pck_to_send)
            #ecrypt_payload=self.ecrypt_mens('INF             ')
            self.write_payload(ecrypt_payload)
            self.set_mode(MODE.TX)

            start_time = time.time()
            while ((self.wait_for_send==0) and (time.time() - start_time < 5)): # wait until
receive data or 5s
                pass;
            if (time.time() - start_time > 5):
                print("Demorou mais de 5s -" , time.time() - start_time)
                hass.components.persistent_notification.async_create("""
                Ocorreu um erro ao tentar enviar uma mensagem.
                Tente novamente. Se o problema persistir reinicie o equipamento.
                """, 'AssistLora ERROR')
                erro=1
            else:
                self.wait_for_send=0

            BOARD2.led_on()
            if (erro==0): #esperar pelo ack
                start_time = time.time()
                while ((self.RX_BUFFER!=pck_to_receive) and (time.time() - start_time < 8)): #
wait until receive ACK or 8s
                    pass;
                if (time.time() - start_time > 8):
                    print("Demorou mais de 8s a receber -", time.time() - start_time)
                    tipo=pck_to_send[0:1]
                    if(tipo=='1'):
                        tipo='Switch'
                    elif(tipo=='2'):
                        tipo='Binary Sensor'
                    elif(tipo=='3'):
                        tipo='Sensor'
                    elif(tipo=='4'):
                        tipo='Light'

                    hass.components.persistent_notification.async_create("""
                    Não houve comfirmação de recepção da mensagem pelo dispositivo.
```

```
                    O dispositivo """+tipo+""" com ID-"""+pck_to_send[1:4]+""" existe?
                    Verifique se a ligação do dispositivo e se a sua configuração estão
corretas.
                    """, 'AssistLora ERROR')
                    return False
                else: #recebeu o ack
                    print("Demorou a receber o ack -", time.time() - start_time)
                    self.RX_BUFFER='' #limpar buffer
                    return True
            else: #houve um erro
                return False

        def process_received_mens(self, RX_BUFFER):
            TP=RX_BUFFER[0:1]
            ID=RX_BUFFER[1:4] #retirar do buffer o ID
            TPS=RX_BUFFER[4:5]
            LEN=RX_BUFFER[5:6]
            MENS=RX_BUFFER[6:]
            print('Process received mens long range-> RX=', RX_BUFFER, "; TP=", TP, "; ID=",
ID, "; TPS=", TPS, "; LEN=", LEN,"; MENS=", MENS)
            if(int(ID)>=1 and int(ID)<=99): # Se o ID estiver fora do intervalo
lora_long_range - IDs =[100-999] dá erro
                print("O elemento "+str(int(ID))+ " nao é long range")
                hass.components.persistent_notification.async_create("""
                O elemento com o ID-"""+str(int(ID))+" é de medio alcance, no entanto foi
configurado como longo alcance."+"""
                Altere a configuração no dispositivo.
                """, 'AssistLora ERROR')
            elif(TP=='0'): #MENS para a gateway
                if(TPS=='1'): #MENS para Switch
                    if(MENS=='ON        '):
                        print("Mudar State para ON do switch."+str(int(ID)))
                        states = hass.states.get('switch.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to switch
                            if(enviou==True):
                                print("State ON")
                                #print(hass.states.get('switch.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('switch.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para switch."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Switch com
o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento switch."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Switch com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')
                    elif(MENS=='OFF       '):
                        print("Mudar State para OFF do switch."+str(int(ID)))
                        states = hass.states.get('switch.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to switch
                            if(enviou==True):
                                print("State OFF")
                                #print(hass.states.get('switch.'+str(int(ID))))
```

```
                                atts = states.attributes.copy()
                                hass.states.set('switch.'+str(int(ID)), "off", atts)
                            else:
                                print("Erro a enviar ACK para switch."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Switch com
o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento switch."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Switch com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                if(TPS=='2'): #MENS para Binary_Sensor
                    if(MENS=='1         '):
                        print("Mudar State para 1 (HIGH) do binary_sensor."+str(int(ID)))
                        states = hass.states.get('binary_sensor.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -200503ACK
- send ack to binary sensor
                            if(enviou==True):
                                print("State 1")
                                #print(hass.states.get('switch.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('binary_sensor.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para binary_sensor."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Binary
Sensor com o ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento binary_sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Binary Sensor com ID-
"""+str(int(ID))+""" no entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                    elif(MENS=='0         '):
                        print("Mudar State para 0 (LOW) do binary_sensor."+str(int(ID)))
                        states = hass.states.get('binary_sensor.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -200503ACK
- ssend ack to binary sensor
                            if(enviou==True):
                                print("State OFF")
                                #print(hass.states.get('switch.'+str(int(ID))))

                                atts = states.attributes.copy()
                                hass.states.set('binary_sensor.'+str(int(ID)), "off", atts)
                            else:
                                print("Erro a enviar ACK para binary_sensor."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Binary
Sensor com o ID-"""+str(int(ID))+"""
```

```
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento binary_sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Binary Sensor com ID-
"""+str(int(ID))+""" no entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                            """, 'AssistLora ERROR')

                if(TPS=='3'): #MENS para Sensor
                    print("Actualizar do sensor."+str(int(ID)))
                    data=MENS[:int(LEN)]
                    states = hass.states.get('sensor.'+str(int(ID)))
                    if(states!=None):
                        enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -300503ACK        -
send ack to sensor
                        if(enviou==True):
                            print("Update State")
                            atts = states.attributes.copy()
                            hass.states.set('sensor.'+str(int(ID)), data, atts)
                        else:
                            print("Erro a enviar ACK para sensor."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Ocorreu um erro ao tentar enviar uma mensagem para Sensor com o
ID-"""+str(int(ID))+"""
                            Tente novamente. Se o problema persistir reinicie o equipamento.
                            """, 'AssistLora ERROR')
                    else:
                        print("Nao existe o elemento sensor."+str(int(ID)))
                        hass.components.persistent_notification.async_create("""
                        Foi recebida uma mensagem do Sensor com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                        Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                        """, 'AssistLora ERROR')

                if(TPS=='4'): #MENS para Light
                    if(MENS=='ON        '):
                        print("Mudar State para ON do light."+str(int(ID)))
                        states = hass.states.get('light.'+str(int(ID)))
                        if(states!=None):
                            enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to light
                            if(enviou==True):
                                print("State ON")
                                #print(hass.states.get('light.'+str(int(ID))))
                                atts = states.attributes.copy()
                                hass.states.set('light.'+str(int(ID)), "on", atts)
                            else:
                                print("Erro a enviar ACK para light."+str(int(ID)))
                                hass.components.persistent_notification.async_create("""
                                Ocorreu um erro ao tentar enviar uma mensagem para Light com o
ID-"""+str(int(ID))+"""
                                Tente novamente. Se o problema persistir reinicie o
equipamento.
                                """, 'AssistLora ERROR')
                        else:
                            print("Nao existe o elemento light."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Foi recebida uma mensagem do Light com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                            Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
```

```
                        """, 'AssistLora ERROR')

                elif(MENS=='OFF       '):
                    print("Mudar State para OFF do light."+str(int(ID)))
                    states = hass.states.get('light.'+str(int(ID)))
                    if(states!=None):
                        enviou=self.send_ack(TPS+ID+TP+'3ACK       ') #  -100503ACK
- send ack to light
                        if(enviou==True):
                            print("State OFF")
                            #print(hass.states.get('light.'+str(int(ID))))

                            atts = states.attributes.copy()
                            hass.states.set('light.'+str(int(ID)), "off", atts)
                        else:
                            print("Erro a enviar ACK para light."+str(int(ID)))
                            hass.components.persistent_notification.async_create("""
                            Ocorreu um erro ao tentar enviar uma mensagem para Light com o
ID-"""+str(int(ID))+"""
                            Tente novamente. Se o problema persistir reinicie o
equipamento.
                            """, 'AssistLora ERROR')
                    else:
                        print("Nao existe o elemento light."+str(int(ID)))
                        hass.components.persistent_notification.async_create("""
                        Foi recebida uma mensagem do Light com ID-"""+str(int(ID))+""" no
entanto este dispositivo nao está configurado.
                        Verifique se a configuração do dispositivo está correta ou inclua
o dispositivo no fichiro de configuração.
                        """, 'AssistLora ERROR')


    BOARD2.setup()
    BOARD2.reset()
    lora_long_range = ra_02_board2(verbose=False)
    #     Slow+long range  Bw = 125 kHz, Cr = 4/8, Sf = 4096chips/symbol, CRC on. 13 dBm
    lora_long_range.set_pa_config(pa_select=1, max_power=21, output_power=15)
    lora_long_range.set_bw(BW.BW125)
    lora_long_range.set_coding_rate(CODING_RATE.CR4_8)
    lora_long_range.set_spreading_factor(12)
    lora_long_range.set_rx_crc(True)
    lora_long_range.set_low_data_rate_optim(True)
    assert(lora_long_range.get_agc_auto_on() == 1)
    lora_long_range.start() # inicia ra-02  para escuta




    def exit_lora(event):
        """Stuff to do before stopping."""
        log.info("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~EXIT~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        lora_long_range.set_mode(MODE.SLEEP)
        lora_medium_range.set_mode(MODE.SLEEP)
        BOARD.led_off()
        BOARD2.led_off()

    def prepare_lora(event):
        """Stuff to do when home assistant starts."""
        log.info("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~PREPARE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
        BOARD.led_on()
        BOARD2.led_on()
        hass.bus.listen_once(EVENT_HOMEASSISTANT_STOP, exit_lora)

    hass.bus.listen_once(EVENT_HOMEASSISTANT_START, prepare_lora)
```

```
    lora_class=[lora_medium_range, lora_long_range]
    hass.data[DOMAIN]=lora_class

    hass.components.persistent_notification.async_create("""
Este é o painel de controlo do sistema, aqui pode controlar todos os dispositivos do
AssistLora.
Pode alterar adicionar/eliminar/configurar dispositivos na tab Configurador alterando o
ficheiro `configuration.yaml`.

 - [Pagina Inicial](http://assistlora.com/index.php)
 - [Configurar o Sistema](http://assistlora.com/config.php)
 - [Sobre o Projeto](http://assistlora.com/about.php)

""", 'Bem Vindo ao AssistLora!')

    return True
```

Código no Home Assistant – Componente Switch – lora.py

```
"""
 This program is part of ASSISTLORA

 Copyright 2018 Rui Silva.
 This file is part of rpsreal/assistlora
 Switch Control
"""
import logging

import voluptuous as vol

from custom_components.assistlora import DOMAIN
from homeassistant.components.switch import SwitchDevice, PLATFORM_SCHEMA
from homeassistant.const import (
EVENT_HOMEASSISTANT_START, EVENT_HOMEASSISTANT_STOP, CONF_NAME, CONF_ICON)
import homeassistant.helpers.config_validation as cv

DEPENDENCIES = ['assistlora']

CONF_PULL_MODE = 'pull_mode'
CONF_DEVICES = 'devices'
CONF_ID = 'id'

SWITCH_SCHEMA = vol.Schema({
    #vol.Optional(CONF_NAME, default='true'): cv.string,
    vol.Optional(CONF_ICON, default='true'): cv.string,
})
PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend({
    vol.Required(CONF_DEVICES): vol.Schema({cv.slug: SWITCH_SCHEMA}),
})



# pylint: disable=no-member
def setup_platform(hass, config, add_devices, discovery_info=None):
    """Find and return switches controlled by a generic RF device via GPIO."""

    lora_class=hass.data[DOMAIN]
    switches = []
    devices = config.get(CONF_DEVICES, {})

    for ID, device_config in devices.items():

        ID_int=int(ID)
        if((ID_int>0) and (ID_int<100)): # Use medium range
            lora=lora_class[0]
        else:
            lora=lora_class[1]

        switches.append(
            AssistloraSwitch(
                lora,
                ID,
                #device_config.get(CONF_NAME),
                'mdi:'+device_config.get(CONF_ICON) #mdi:panda
            )
        )

    if not switches:
        _LOGGER.error("No switches added")
        return False

    add_devices(switches)
```

```
    print("Switches added")



class AssistloraSwitch(SwitchDevice):
    """Representation of AssistloraSwitch"""

    def __init__(self, lora, ID, icon):
        """Initialize"""
        self.lora = lora
        self._name = ID
        self._state = False
        self._icon = icon


    @property
    def name(self):
        """Return the name of the switch."""
        return self._name

    @property
    def should_poll(self):
        """No polling needed."""
        return False

    @property
    def icon(self):
        """Return the icon to use for device if any."""
        return self._icon

    @property
    def is_on(self):
        """Return true if device is on."""
        return self._state

    def turn_on(self, **kwargs):
        print("Turn ON - ", self._name)
        device_ID=self._name
        erro=False

        TPS="0" #switch # TP do Sender
        ID_int=int(device_ID)
        if ((ID_int>=0) and (ID_int<=9)):
            ID='00'+device_ID  # por ex "002"
        elif ((ID_int>=10) and (ID_int<=99)):
            ID='0'+device_ID  # por ex "052"
        elif ((ID_int>=100) and (ID_int<=999)):
            #ID=ID_str  # por ex "345"
            ID=device_ID
        else:
            print("Error ID configuration check configuration.yaml")
            erro=True

        if (erro==False):
            TP="1" # TP do Receptor
            LEN="2"
            MENS="ON        "
            enviou=self.lora.send_mens(TP+ID+TPS+LEN+MENS, TPS+ID+TP+"3"+"ACK       ") #
enviar "100502ON        "      receber tipo "000513ACK       "
            print("Enviou Sinal e recebeu Ack? ", enviou)

        if(enviou==True and erro==False):
            self._state = True        # Só é activado depois de receber o ack
            self.schedule_update_ha_state()
```

```
    def turn_off(self, **kwargs):
        print("Turn OFF - ", self._name)
        device_ID=self._name

        erro=False

        TPS="0" #switch # TP do Sender
        ID_int=int(device_ID)
        if ((ID_int>=0) and (ID_int<=9)):
            ID='00'+device_ID  # por ex "002"
        elif ((ID_int>=10) and (ID_int<=99)):
            ID='0'+device_ID  # por ex "052"
        elif ((ID_int>=100) and (ID_int<=999)):
            #ID=ID_str  # por ex "345"
            ID=device_ID
        else:
            print("Error ID configuration check configuration.yaml")
            erro=True

        if (erro==False):
            TP="1" # TP do Receptor
            LEN="3"
            MENS="OFF       "
            enviou=self.lora.send_mens(TP+ID+TPS+LEN+MENS, TPS+ID+TP+"3"+"ACK       ") #
enviar "100503OFF       "      receber tipo "000513ACK       "
            print("Enviou Sinal e recebeu Ack? ", enviou)

        if(enviou==True and erro==False):
            self._state = False        # Só é activado depois de receber o ack
            self.schedule_update_ha_state()
```

Código no Home Assistant – Componente Light – lora.py

```
"""
 This program is part of ASSISTLORA

 Copyright 2018 Rui Silva.
 This file is part of rpsreal/assistlora
 Light Control
"""
import logging

import voluptuous as vol

from custom_components.assistlora import DOMAIN
from homeassistant.components.light import Light, PLATFORM_SCHEMA
from homeassistant.const import (
EVENT_HOMEASSISTANT_START, EVENT_HOMEASSISTANT_STOP, CONF_NAME, CONF_ICON)
import homeassistant.helpers.config_validation as cv

DEPENDENCIES = ['assistlora']

CONF_PULL_MODE = 'pull_mode'
CONF_DEVICES = 'devices'
CONF_ID = 'id'

LIGHT_SCHEMA = vol.Schema({
    #vol.Optional(CONF_NAME, default='true'): cv.string,
    vol.Optional(CONF_ICON, default='true'): cv.string,
})
PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend({
    vol.Required(CONF_DEVICES): vol.Schema({cv.slug: LIGHT_SCHEMA}),
})



# pylint: disable=no-member
def setup_platform(hass, config, add_devices, discovery_info=None):
    """Find and return switches controlled by a generic RF device via GPIO."""

    lora_class=hass.data[DOMAIN]

    lights = []
    devices = config.get(CONF_DEVICES, {})

    for ID, device_config in devices.items():

        ID_int=int(ID)
        if((ID_int>0) and (ID_int<100)): # Use medium range
            lora=lora_class[0]
        else:
            lora=lora_class[1]

        lights.append(
            AssistloraLight(
                lora,
                ID,
                #device_config.get(CONF_NAME),
                'mdi:'+device_config.get(CONF_ICON) #mdi:panda
            )
        )

    if not lights:
        _LOGGER.error("No lights added")
        return False

    add_devices(lights)
```

```
    print("Light added")


class AssistloraLight(Light):
    """Representation of a  Raspberry Pi GPIO."""

    def __init__(self, lora, ID, icon):
        """Initialize the pin."""
        self.lora = lora
        self._name = ID
        self._state = False
        self._icon = icon


    @property
    def name(self):
        """Return the name of the light."""
        return self._name

    @property
    def should_poll(self):
        """No polling needed."""
        return False

    @property
    def icon(self):
        """Return the icon to use for device if any."""
        return self._icon

    @property
    def is_on(self):
        """Return true if device is on."""
        return self._state

    def turn_on(self, **kwargs):
        print("Turn ON - ", self._name)
        device_ID=self._name
        erro=False

        TPS="0" #switch # TP do Sender
        ID_int=int(device_ID)
        if ((ID_int>=0) and (ID_int<=9)):
            ID='00'+device_ID  # por ex "002"
        elif ((ID_int>=10) and (ID_int<=99)):
            ID='0'+device_ID  # por ex "052"
        elif ((ID_int>=100) and (ID_int<=999)):
            #ID=ID_str  # por ex "345"
            ID=device_ID
        else:
            print("Error ID configuration check configuration.yaml")
            erro=True

        if (erro==False):
            TP="4" # TP do Receptor
            LEN="2"
            MENS="ON        "
            enviou=self.lora.send_mens(TP+ID+TPS+LEN+MENS, TPS+ID+TP+"3"+"ACK       ") #
enviar "100502ON        "      receber tipo "000513ACK       "
            print("Enviou Sinal e recebeu Ack? ", enviou)

        if(enviou==True and erro==False):
            self._state = True        # Só é activado depois de receber o ack
            self.schedule_update_ha_state()
```

```
    def turn_off(self, **kwargs):
        print("Turn OFF - ", self._name)
        device_ID=self._name

        erro=False

        TPS="0" #switch # TP do Sender
        ID_int=int(device_ID)
        if ((ID_int>=0) and (ID_int<=9)):
            ID='00'+device_ID  # por ex "002"
        elif ((ID_int>=10) and (ID_int<=99)):
            ID='0'+device_ID  # por ex "052"
        elif ((ID_int>=100) and (ID_int<=999)):
            #ID=ID_str  # por ex "345"
            ID=device_ID
        else:
            print("Error ID configuration check configuration.yaml")
            erro=True

        if (erro==False):
            TP="4" # TP do Receptor
            LEN="3"
            MENS="OFF       "
            enviou=self.lora.send_mens(TP+ID+TPS+LEN+MENS, TPS+ID+TP+"3"+"ACK       ") #
enviar "100503OFF       "      receber tipo "000513ACK       "
            print("Enviou Sinal e recebeu Ack? ", enviou)

        if(enviou==True and erro==False):
            self._state = False        # Só é activado depois de receber o ack
            self.schedule_update_ha_state()
```

Código no Home Assistant – Componente Sensor Analogico – lora.py

```
"""
This program is part of ASSISTLORA

 Copyright 2018 Rui Silva.
 This file is part of rpsreal/assistlora
 Sensor analogico
"""
import logging
import voluptuous as vol
from custom_components.assistlora import DOMAIN
from homeassistant.components.sensor import PLATFORM_SCHEMA
from homeassistant.helpers.entity import Entity
from homeassistant.const import (
EVENT_HOMEASSISTANT_START, EVENT_HOMEASSISTANT_STOP, CONF_NAME, CONF_ICON,
CONF_MONITORED_CONDITIONS)
import homeassistant.helpers.config_validation as cv

_LOGGER = logging.getLogger(__name__)
DEPENDENCIES = ['assistlora']

CONF_DEVICES = 'devices'
CONF_ID = 'id'

SENSOR_SCHEMA = vol.Schema({
    #vol.Optional(CONF_NAME, default='true'): cv.string,
    vol.Required(CONF_MONITORED_CONDITIONS): cv.string,
    vol.Optional(CONF_ICON): cv.string,
})
PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend({
    vol.Required(CONF_DEVICES): vol.Schema({cv.slug: SENSOR_SCHEMA}),
})



# pylint: disable=no-member
def setup_platform(hass, config, add_devices, discovery_info=None):
    """Find and return sensores controlled by a generic RF device via GPIO."""

    lora_class=hass.data[DOMAIN]

    sensors = []
    devices = config.get(CONF_DEVICES, {})

    for ID, device_config in devices.items():

        ID_int=int(ID)
        if((ID_int>0) and (ID_int<100)): # Use medium range
            lora=lora_class[0]
        else:
            lora=lora_class[1]

        sensors.append(
            AssistloraSensor(
                lora,
                ID,
                'mdi:'+device_config.get(CONF_ICON),
                device_config.get(CONF_MONITORED_CONDITIONS)
            )
        )

    if not sensors:
        _LOGGER.error("No sensors added")
        return False
```

```
    add_devices(sensors, True)
    print("Sensors added")



class AssistloraSensor(Entity):
    """Represent a sensor that uses Raspberry Pi GPIO."""
    def __init__(self, lora, ID, icon, unit):
        """Initialize the pin."""
        self.lora = lora
        self._name = ID
        self._state = 0
        self._icon = icon
        self.unit = unit

    @property
    def name(self):
        #Return the name of the switch.
        return self._name

    @property
    def should_poll(self):
        #No polling needed.
        return False

    @property
    def state(self):
        """Return the state of the sensor."""
        return self._state

    @property
    def icon(self):
        #Return the icon to use for device if any.
        return self._icon

    @property
    def unit_of_measurement(self):
        #Return the unit of measurement.
        #return self._unit
        return self.unit
```

Código no Home Assistant – Componente Sensor binario – lora.py

```
"""
This program is part of ASSISTLORA

 Copyright 2018 Rui Silva.
 This file is part of rpsreal/assistlora
 Sensor binario
"""
import logging

import voluptuous as vol

from custom_components.assistlora import DOMAIN
from homeassistant.components.binary_sensor import (BinarySensorDevice, PLATFORM_SCHEMA)
from homeassistant.const import (
EVENT_HOMEASSISTANT_START, EVENT_HOMEASSISTANT_STOP, CONF_NAME, CONF_ICON)
import homeassistant.helpers.config_validation as cv

DEPENDENCIES = ['assistlora']

CONF_DEVICES = 'devices'
CONF_ID = 'id'

BINARY_SENSOR_SCHEMA = vol.Schema({
    #vol.Optional(CONF_NAME, default='true'): cv.string,
    vol.Optional(CONF_ICON, default='true'): cv.string,
})
PLATFORM_SCHEMA = PLATFORM_SCHEMA.extend({
    vol.Required(CONF_DEVICES): vol.Schema({cv.slug: BINARY_SENSOR_SCHEMA}),
})



# pylint: disable=no-member
def setup_platform(hass, config, add_devices, discovery_info=None):
    """Find and return sensores controlled by a generic RF device via GPIO."""

    lora_class=hass.data[DOMAIN]

    sensores = []
    devices = config.get(CONF_DEVICES, {})

    for ID, device_config in devices.items():

        ID_int=int(ID)
        if((ID_int>0) and (ID_int<100)): # Use medium range
            lora=lora_class[0]
        else:
            lora=lora_class[1]

        sensores.append(
            AssistloraBinarySensor(
                lora,
                ID,
                #device_config.get(CONF_NAME),
                'mdi:'+device_config.get(CONF_ICON) #mdi:panda
            )
        )

    if not sensores:
        _LOGGER.error("No binary sensors added")
        return False

    add_devices(sensores)
    print("Binary_sensors added")
```

```
class AssistloraBinarySensor(BinarySensorDevice):
    """Represent a binary sensor that uses Raspberry Pi GPIO."""

    def __init__(self, lora, ID, icon):
        """Initialize the pin."""
        self.lora = lora
        self._name = ID
        self._state = False
        self._icon = icon

    @property
    def name(self):
        """Return the name of the binary sensor."""
        return self._name

    @property
    def should_poll(self):
        """No polling needed."""
        return False

    @property
    def icon(self):
        """Return the icon to use for device if any."""
        return self._icon

    @property
    def is_on(self):
        """Return true if device is on."""
        return self._state
```