INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Arquitetura de Suporte ao Desenvolvimento de Sistemas Auto-configuráveis para Redes de Sensores sem Fios

César Miguel da Silva Lúcio Penha

2019

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Arquitetura de Suporte ao Desenvolvimento de Sistemas Auto-configuráveis para Redes de Sensores sem Fios

Elaborado por:

César Miguel da Silva Lúcio Penha

Orientado por:

Professor Doutor Rogério Campos-Rebelo, IPBeja

Professor Doutor João Paulo Barros, IPBeja

Dissertação de Mestrado
Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

2019

Agradecimentos

Institucionalmente, os meus agradecimentos a toda a comissão cientifica do Mestrado em Internet das Coisas, pelas facilidades e soluções concedidas para a realização deste trabalho. Meus sinceros agradecimentos ao Professor Doutor Rogério Campos Rebelo, orientador deste trabalho, pela preciosa ajuda pela sua disponibilidade e orientações e pelos seus valiosos contributos enquanto professor da unidade curricular de Infraestruturas e Comunicações para a Internet das Coisas. Obrigado por nunca dizer não a qualquer uma das minhas dúvidas. Ao coordenador do Mestrado em Internet das Coisas e coorientador Professor Doutor João Paulo Barros pelo apoio incondicional e conhecimentos valiosos facultados, que tanto ajudaram durante esta fase. Aos meus colegas de curso pelo seu companheirismo, preocupação e alto profissionalismo que em muito contribuiu para que me fosse possível conseguir encarar as dificuldades inerentes ao Mestrado, mesmo quando tudo parecia estar contra.

À minha mãe, pelo exemplo que sempre foi, pela educação que me deu e pela possibilidade que me ofereceu para fazer uma formação superior - Eternamente grato.

À minha namorada e companheira Lara, pela paciência, pelo tempo que lhe deixei de dedicar, pelas noites mal dormidas, mas sobretudo por acreditar!

Aos meus amigos e colegas que estudaram e trabalharam comigo durante a minha formação, em especial ao Engenheiro e colega de mestrado Rui Mesquita - *"The sky is the limit..."*, ao Professor José Carlos Gil pelos seus preciosos e exímios conhecimentos de inglês - *"Thank you very much !"* e ao meu amigo, Engenheiro João Vieira, pela paciência demonstrada aquando das minhas dúvidas e pelo vasto conhecimento, muitas vezes sem o parecer - *"DJ, spin that music!*.

A todos, bem hajam!

Resumo

Arquitetura de Suporte ao Desenvolvimento de Sistemas Auto-configuráveis para Redes de Sensores sem Fios

A agricultura em geral e os espaços verdes em particular permanecem como atividades em que a quantidade de água desperdiçada persiste em níveis muito altos. Apesar de toda a tecnologia disponível atualmente, ainda é comum encontrar parques e espaços verdes onde os sistemas de irrigação aplicados são frequentemente acionados com uma necessidade duvidosa. Atualmente, a maioria dos sistemas de irrigação são operados manualmente ou baseados em sistemas equipados com controladores horários. A Internet das coisas, pode ajudar na recolha de informações, na tomada de decisões, na obtenção de dados através de sensores e no seu feedback. Este trabalho apresenta uma abordagem para desenvolver sistemas IoT reconfiguráveis usando tecnologias sem fios como, por exemplo, um sistema de irrigação automático. Três principais contribuições são propostas: (1) Uma abordagem de automatização de projeto de sistemas IoT suportada no uso de um formalismo de modelação gráfica para a definição de regras que especificam o comportamento do sistema, permitindo a geração automática de código para microcontroladores, bem como a comunicação através de protocolo sem fios. (2) Uma ferramenta de automatização de projeto para o desenvolvimento de redes de sensores sem fios em LoRa modeladas com redes de Petri IOPT; (3) o desenvolvimento de um protótipo de um sistema de irrigação automático sem fios, suportando a verificação e seleção automático de dispositivos sensores, com base no tipo de sensor requerido e o respetivo indicador de intensidade de sinal recebido.

Palavras-chave: *Sistemas de Rega, Internet das Coisas, Ferramentas de Automatização, Desenvolvimento Baseado em Modelos, Redes de Petri.*

Abstract

Architecture for Development Suport of Auto-configurable Systems for Wireless Sensor Networks

Agriculture in general and landscape in particular remain activities where the amount of wasted water persists at very high levels. Despite all the technology available today, it is still common to find parks and green spaces where applied irrigation systems are often triggered with a dubious need. Currently, most irrigation systems are either manually operated or based on systems equipped with time controllers. The IoT can help us gather information, make decisions, get data through sensors, and get your feedback. This work presents an approach to developing reconfigurable IoT systems using wireless technologies such as an automatic irrigation system. Three main contributions are proposed: (1) An IoT systems design automation approach supported by the use of a graphical modeling formalism to define rules that specify system behavior, allowing automatic code generation for microcontrollers, as well as communication over wireless protocol. (2) A design automation tool for the development of LoRa wireless sensor networks modeled with IOPT Petri nets; (3) the development of a prototype wireless automatic irrigation system, supporting automatic verification and selection of sensor devices, based on the type of sensor required and the respective signal strength indicator received.

Keywords: *Irrigation Systems, Internet of Things, Design Automation, Model-Driven Development, Petri Nets.*

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Abreviaturas e Siglas

ADC	Analog Digital Converter
ADR	Adaptative Data Rate
ANSI	American National Standards Institute
AVR	Advanced Virtual RISC
BD	Base de Dados
CCS	Chirp Spread Spectrum
dB	Décibel
dBi	Décibel Isotrópico
dBm	Décibel miliwatt
DC	Direct current
DHT	Digital Humidity Temperature
EEPROM	Electrically-Erasable Programmable Read-Only Memory
FCC	Federal Communications Commission
FSK	Frequency-shift keying
FSPL	Free Space Path Loss
FTP	File Transfer Protocol
FZ	Fresnel Zone
IDE	Integrated development environment
IOPT	Input Output Place-Transitions Tool
IoT	Internet of Things
IPBeja	Instituto Politécnico de Beja
ISM	Industrial Scientific and Medical
JSON	JavaScript Object Notation
KB	quilobyte
KWh	Kilowatt hour
LED	Light Emiting Diode
Li-Po	Lithium Polymer Battery
LoRaWAN	Long Range Wireless Area Network

LoS	Line of Sight
LPWAN	Low Power Wireless Area Network
mA	miliamper
MCU	Microcontroller Unit
MCU	Microcontroller Unit
METNO	Norwegian Meteorological Institute
MHz	MegaHertz
MIPS	Millions of Instructions Per Second
MPa	Megapascal
NCC	National Certification Corporation
NOAA	National Oceanic and Atmospheric Administration
OSC	Oscillator
PCB	Printed Circuit Board
PIC	Programmable Interrupt Controller
PWM	Pulse With Modulation
RdP	Rede de Petri
RF	Radio Frequency
RGB	Red Green Blue
RISC	Reduced Instruction Set Compute
RSSI	Received Signal Strength Indicator
RTC	Real Time clock
SF	Spreading Factor
SMPS	Switched-mode power supply
SNR	Signal-To-Noise Ratio
SRAM	Static Random Access Memory
SWAIS	Self-Configurable Wireless Automatic Irrigation System
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VCC	Common Collector Voltage
VHDL	VHSIC Hardware Description Language
WSN	Wireless Sensor Networks

Capítulo 1

Introdução

1.1 Motivação

Os sistemas de rega são infraestruturas que garantem a irrigação em condições específicas de pressão e caudal em toda a zona a irrigar. A irrigação não se cinge unicamente à distribuição de água; a avaliação das necessidades em relação à quantidade, qualidade e transporte deste recurso, fazem parte do processo de irrigação.

A irrigação adequada de uma determinada área verde, pressupõe uma boa utilização do recurso hídrico e também de quatro aspetos fundamentais:

1. as necessidades hídricas específicas de cada espécie, tendo em conta os valores da evapotranspiração da zona a irrigar;

2. o desenho e implementação de sistemas de rega eficientes de modo a obter uma distribuição uniforme do recurso hídrico ao longo da zona a irrigar;

3. uma programação apropriada do próprio sistema de rega;

4. a análise dos próprios indicadores de qualidade da irrigação[1].

Diferentes sistemas de irrigação possuem diferentes eficiências de aplicação. Entender as eficiências de aplicação de diferentes sistemas de irrigação pode ajudar a tomar decisões de utilização da irrigação de forma a obter o melhor rendimento e qualidade da rega.

Em espaços verdes tais como jardins e parques públicos ou privados, comumente verifica-se irrigação em excesso, provocando escorrência superficial, ou por defeito, provocando mau desenvolvimento das plantas ou, em casos extremos, a ausência de vegetação. Este panorama pode e deve ser tido em conta aquando da conceção dos espaços verdes e, posteriormente, na sua manutenção e conservação, implementando técnicas e práticas sustentáveis, beneficiando deste modo o valor paisagístico desses mesmos espaços [2, 3]. As eficiências de aplicação do sistema de irrigação têm melhorado constantemente desde a introdução da irrigação por aspersão. As eficiências de aplicação

aumentaram de menos de 30% em 1965 para mais de 70% em 2005 [4]. Atualmente, o setor económico que mais consome água é a agricultura: Cerca 70% dos recursos totais, contra os 20% utilizados pela indústria e os 10% para uso doméstico.

A incrementar ao consumo, temos o desperdício no uso da água. A nível nacional, o setor com mais desperdício é a agricultura. Cerca de 35% de desperdício, seguido pelo setor urbano com 20% e do industrial com 15% [5]. A incrementar a este problema e considerando o aumento esperado na população mundial, urge encontrar soluções para assegurar o fornecimento suficiente da água a toda a população e evitar ao máximo desperdícios. Além do desperdício deste recurso tão escasso, outro aspeto relevante consiste no consumo de energia dos sistemas de rega. Os sistemas de rega são, na maior parte dos casos, abastecidos por estações de bombagem. Essas estações são responsáveis por cerca de 50% da fatura energética.

Considerando estes dados, a estimativa fiável das necessidades hídricas dos espaços verdes, desempenha um papel fundamental não só no projeto de um sistema de rega mais adequado, mas também no apoio para a gestão da rega ao longo do ano. Por outro lado, o desenvolvimento de modelos matemáticos da rede permite ao responsável pela rega um melhor conhecimento do comportamento hidráulico da mesma, possibilitando uma utilização energeticamente mais eficiente, através da simulação de diversas condições de funcionamento[6].

Este tema surge da necessidade urgente e atual da utilização racional dos recursos naturais, nomeadamente solo, vegetação e principalmente da água. Pretende-se que haja um equilíbrio quer a nível económico quer a nível ambiental que permita a gestão sustentável destes recursos sendo provavelmente a água o recurso com maior relevância.

A irrigação inteligente é um componente fundamental da agricultura de precisão; Ajuda a evitar o desperdício de água e melhorar a qualidade do cultivo nos campos ou espaços verdes:

1. Irrigando nos momentos corretos;

2. Minimizando os escoamentos e outros desperdícios;

3. Determinando os níveis de humidade do solo com precisão, encontrando os requisitos de irrigação em qualquer lugar.

A substituição da irrigação manual por válvulas e sistemas automáticos também elimina o elemento de erro humano (por exemplo, o esquecimento de desligar uma válvula depois de regar o espaço verde) e torna-se fundamental para economizar energia, tempo e recursos valiosos.

Os objetos interligados, também referidos como a Internet das Coisas ou *Internet of Things* (IoT), continuam evoluindo, oferecendo mais controlo sobre nosso ambiente de vida e facilitando a realização de determinadas tarefas. Muitos consideram isso como o

próximo grande horizonte na evolução da Internet. A capacidade de recolher, armazenar, analisar e distribuir dados entre diversas interfaces, aplicativos e dispositivos, levou à facilidade de desenvolvimento de aplicações que disponibilizam dados em tempo real.

A irrigação inteligente pode e deve beneficiar destas novas tecnologias IoT, obtendo informações em tempo real sobre vários aspectos relacionados com a irrigação. Por exemplo, existem sensores que podem oferecer leituras remotamente, fornecer dados em tempo real sobre o nível de humidade ou temperatura do solo de um determinado espaço. Todos esses dados fornecidos por sensores de alta tecnologia, podem ajudar-nos a gerir melhor os recursos de irrigação e a própria irrigação.

Este trabalho apresenta uma abordagem para desenvolver sistemas IoT reconfiguráveis, usando tecnologias sem fios, nomeadamente, um sistema de irrigação automático.

É proposta uma abordagem de automatização de projeto de sistemas IoT, suportada no uso de um formalismo de modelação gráfica, que definem as regras que especificam o comportamento do sistema, permitindo a geração automática de código para o sistema, bem como a comunicação através de protocolos sem fios.

Foi desenvolvida uma ferramenta de automatização de projeto, que permite a construção de redes de sensores sem fios usando a tecnologia LoRa, modeladas com Redes de Petri (RdP) IOPT;

Foi ainda implementado um protótipo de um sistema de irrigação auto-configurável sem fios, suportando a verificação e seleção automática de dispositivos sensores, com base no tipo de sensor requerido e o respetivo indicador de intensidade de sinal recebido.

1.2 Objetivos

Esta dissertação tem como principal objectivo propor/desenvolver uma arquitetura de suporte ao desenvolvimento de sistemas IoT, como por exemplo sistemas de rega automáticos baseado em modelos.

De forma a atingir esse objetivo, pretende-se aproveitar as vantagens das ferramentas de modelação gráfica e dos formalismos de automação no desenvolvimento deste trabalho.

O sistema proposto deverá ser auto-configurável, permitindo a utilização de diversos dispositivos sensores, selecionando os mais adequados.

Foi desenvolvida uma ferramenta que automatiza todo o processo de integração do sistema, tendo em contas as características do hardware, da comunicação e a modelação gráfica, tirando partido das vantagens que as *Design Automation Tools* proporcionam. A ferramenta proposta pretende também validar a arquitetura do sistema.

Por último foi implementado um protótipo funcional de um sistema de rega automatizado e auto-configurável. Com recurso a este protótipo, pretende-se validar o fun-

cionamento da ferramenta de automatização de projeto e, consequentemente, a própria arquitetura proposta.

1.3 Contribuições

As contribuições visam desenvolver um conjunto de soluções as quais se passam a enumerar:

1. Uma arquitetura de suporte ao desenvolvimento de sistemas auto-configuráveis para redes de sensores sem fios.

2. Uma ferramenta para automatização de projeto de sistemas IoT, que permita a integração do código referente às características do hardware e do protocolo de comunicação, com o código gerado pelo modelo gráfico desenvolvido pelo utilizador.

3. Desenvolvimento e apresentação de um sistema de rega auto-configurável com sensores ambientais e comunicação LoRa baptizado como SWAIS *(Self-Configurable Wireless Automatic Irrigation System)*. O SWAIS é um protótipo funcional de um sistema de rega reconfigurável assente numa rede LPWAN *(Low Power Wireless Area Network)*.

O SWAIS foi descrito e apresentado num artigo na conferência internacional YEF-ECE *(3rd International Young Engineers Forum on Electrical and Computer Engineering)*, realizada em 10 de maio de 2019 na Costa da Caparica - Portugal.[7].

1.4 Estrutura do Documento

A dissertação apresentada está dividida em seis capítulos principais. O presente capítulo **"INTRODUÇÃO"**, tem como objetivo fazer uma breve apresentação do tema, expor as motivações que levaram à realização do estudo e justificar a sua relevância. Neste capítulo são ainda enunciados os objetivos definidos para a dissertação e as suas contribuições. O segundo capítulo **"ENQUADRAMENTO"**, pretende fazer uma introdução ao tema da água e um *overview* da cronológico de evolução dos controladores de rega, desde a sua invenção até aos sistemas utilizados atualmente. Neste capítulo, são também apresentados os conceitos dos formalismos de modelação e das ferramentas de automatização de projeto.

No terceiro capítulo **"CONTRIBUIÇÕES"**, pretende-se apresentar a arquitetura proposta para o sistema, bem como a ferramenta de automatização do projeto que foi desenvolvida e implementada. Neste capítulo faz-se uma breve explicação do protótipo funcional desenvolvido e as suas características.

No quarto capítulo **"EXEMPLO DE VALIDAÇÃO"**, pretende-se explicar detalhadamente quais os componentes utilizados para o desenvolvimento do protótipo funcional, nomeadamente os microcontroladores utilizados, os sensores e os módulos de comunicação LoRa. Neste capítulo são também explicadas as fontes de energia utilizadas para alimentar o sistema e a razão da sua escolha. Por fim é demonstrada a integração de todos os componentes que compõem o sistema numa placa de circuito impresso.

O quinto capítulo **"RESULTADOS"**, apresenta os *outputs* referentes ás comunicações nas portas de comunicação COM, referentes aos dispositivos sensores e atuadores, bem como os testes realizados em relação à cobertura evidenciada pelos dispositivo, os fatores a ter em conta e que podem ser preponderantes quando se implementa uma rede LPWAN. Por último apresentam-se os resultados dos consumos de energia dos dispositivos, e quais foram as preocupações tidas em conta de forma a poder reduzir estes ao mínimo.

Por fim, no capítulo **"CONCLUSÕES E TRABALHOS FUTUROS"**, apresenta-se uma síntese das ilações retiradas ao longo da dissertação, bem como o trabalho futuro a realizar.

Capítulo 2

Enquadramento

Este capítulo apresenta uma visão global sobre os objectivos dos sistemas de rega e a sua evolução ao longo dos tempos. Neste capítulo é também abordada a importância da automação no incremento da produção e na melhoria da qualidade de vida que lhe está associada. É abordado o futuro da gestão da água e quais os benefícios da sua gestão otimizada. Por último é também explicado o conceito dos formalismos de modelação e a sua importância no processo de análise de sistemas automatizados, dando alguns exemplos de ferramentas de automatização de projeto.

2.1 Sistemas de Rega

A água é um recurso cada vez mais escasso [8], e o seu uso racional é mais do que nunca, vital. Os problemas relacionados com a falta de água aumentarão se as previsões a longo prazo sobre as mudanças climáticas globais se confirmarem. Os registos meteorológicos sugerem aumentos significativos da temperatura média e decréscimos na precipitação anual, o que implicará a redução dos recursos hídricos disponíveis no século XXI [9]. A indústria e o turismo, entre outras atividades produtivas, competem por esse recurso de forma a aumentarem a sua rentabilidade e produtividade. Atualmente, o setor económico que mais consome água é a agricultura: Cerca 70% dos recursos totais, contra os 20% utilizados pela indústria e os 10% para uso doméstico [5]. A Figura 2.1 apresenta o consumo de água a nível global deste o ano 1900.

Muitas vezes encontramos sistemas de irrigação que exigem que um operador intervenha, nomeadamente, a abrir e fechar as válvulas de forma manual. Outras vezes encontramos sistemas de irrigação baseados em programação com ciclos horários o que significa que, mesmo em condições de pluviosidade, estes são ativados por um determinado período de tempo, muitas vezes levando ao desperdício de água.

Os sistemas de irrigação convencionais que muitas vezes encontramos nos espaços verdes, nomeadamente em jardins domésticos ou públicos, são compostos principalmente por aspersores ou *sprinklers* de irrigação. Estes sistemas controlados de forma automática

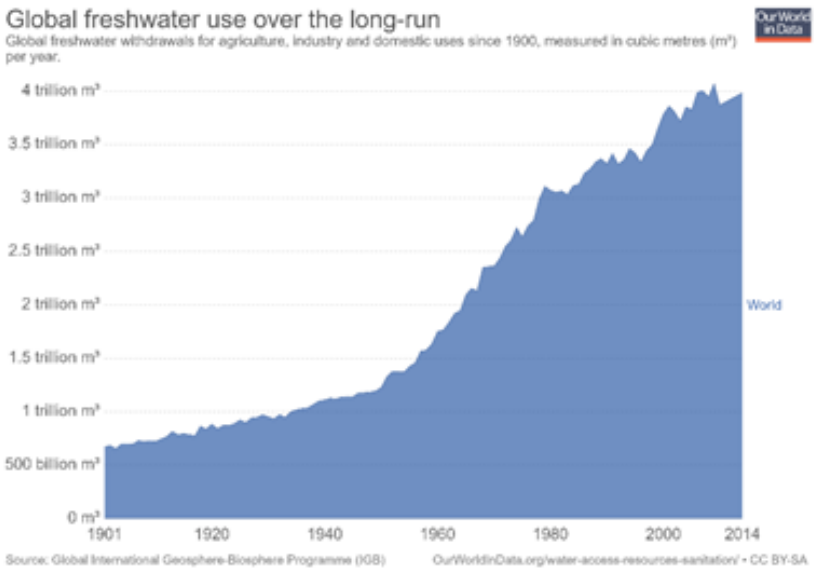

Figura 2.1: Consumo de água a nível global. (*in* http://en.wikipedia.org/wiki/Class_diagram)

ou manual, não tem qualquer *feedback* sobre as condições do solo, condições ambientais ou mesmo o tipo de planta ou cultura numa dada localização. A água é fornecida às plantas ou culturas de uma forma quase aleatória com parâmetros baseados no tempo ou na vontade própria do jardineiro.

No caso de sistemas manuais de irrigação, as plantas são colocadas frequentemente sob stress devido à grande variação na humidade do solo. Esse stress muitas vezes leva ao desenvolvimento de plantas pobres e, por vezes, à morte em no caso de plantas ou culturas mais sensíveis [10]. A falta de um automatismo leva a sub-irrigação ou sob-irrigação do solo. Como já referido, atualmente o problema do desperdício da água é um tema de grande importância e objeto de diversos estudos e trabalhos.

Na rega automática ou automatizada constituídos por sistemas mais simples, este problema é parcialmente resolvido. Esses sistemas são principalmente baseados em controladores de rega que são programados em função do tempo, ou seja, aplicam sempre a mesma quantidade de água num determinado espaço. A automação liberta o trabalho manual de abertura e fecho de válvulas de irrigação, e fornece uma rega mais ou menos constante. No entanto, não resolve todos os problemas [11]. A maioria desses sistemas de irrigação não possui qualquer *feedback* sobre parâmetros ambientais ou do solo. A ideia principal por detrás do controle automático é o uso de *feedback*. No campo de irrigação autónoma, medições de solo, plantas e variáveis climáticas relacionadas às necessidades de água, podem fornecer informações sobre as consequências das ações anteriores para calcular a próxima irrigação.

O principal objetivo dos sistemas de rega é fornecer água ao solo de forma a melhorar as condições em que as plantas cultivadas se desenvolvem. Apesar do objetivo ser sempre

o mesmo a rega pode ser utilizada com diversas finalidades, tais como:

- Humedecimento;
- Proteção;
- Fertilização;
- Distribuição de pesticidas e de herbicidas;
- Lavagem dos sais;

As regas de humedecimento são conhecidas apenas por regas e têm como finalidade compensar a insuficiência da chuva, fornecendo ao terreno a quantidade de água adequada para repor os níveis de humidade do solo dentro dos valores mais indicados para o bom desenvolvimento da cultura. As regas de proteção são normalmente utilizadas para combater os efeitos das condições climáticas desfavoráveis, especialmente as baixas temperaturas e as geadas. As regas de fertilização denominam-se fertirregas e consistem na incorporação dos adubos na água de rega. São especialmente eficientes para a distribuição de adubos azotados e potássicos, nutrientes secundários (magnésio e enxofre) e alguns micronutrientes (boro, zinco e ferro). As regas de distribuição de pesticidas e herbicidas baseiam-se no mesmo principio que as regas de fertilização. Quando as plantas absorvem água do solo existem sais que se vão acumulando na proximidade das raízes, com o tempo os níveis de salinidade aumentam, podendo mesmo tornar-se tóxicos para a planta. Para combater este problema, são aplicadas regas para a lavagem dos sais em excesso do solo.

Uma das abordagens de forma a otimizar a irrigação de campos agrícolas e espaços verdes é o uso de controladores automáticos de rega. A automatização de processos foi aplicada praticamente em todos os campos da engenharia com enorme sucesso, embora a sua implementação na agricultura e de uma forma particular na rega de precisão, seja algo limitada [12]. A ideia-chave por detrás do controlo automático é o uso de *feedback*. No campo da rega autónoma, as medições das variáveis do solo, das plantas e das condições climatéricas, relacionadas com as necessidades hídricas das culturas, podem fornecer a informação das consequências das ações anteriores para calcular a próxima irrigação.

2.1.1 História

Irrigar culturas é trazer água de canos, canais, *sprinklers* ou outros meios artificiais, em vez de depender apenas da chuva. Locais com chuvas escassas ou sazonais não poderiam sustentar a agricultura sem irrigação. Em áreas com precipitação irregular, a irrigação melhora o crescimento e a qualidade das culturas. Desta forma é possível ter plantas não nativas em zonas por vezes com climas bastante diferenciados. Civilizações antigas em muitas partes do mundo praticavam a irrigação. De fato, a civilização não seria possível

sem alguma forma de irrigação. A forma mais antiga de irrigação provavelmente envolvia pessoas carregando baldes de água de poços ou rios para despejar nas suas plantações.

6000 AC

A irrigação começou mais ou menos na mesma época no Egipto e na Mesopotâmia, correspondendo ao atual Iraque e Irão usando a água das enchentes do rio Nilo ou dos rios Tigre / Eufrates. As águas das enchentes, ocorridas de julho a dezembro, eram desviadas para os campos agricolas, sendo drenada de volta para o rio no momento certo do ciclo de crescimento [13].

3100 AC

O primeiro grande projeto de irrigação foi criado pelo rei Menes durante a Primeira Dinastia do Egipto. Ele e seus sucessores usaram represas e canais para usar as águas das inundações desviadas do Nilo, num novo lago chamado lago Moeris [13].

500 AC

Roda de água persa - O primeiro uso do que é hoje chamado de bomba. Este dispositivo era uma série interminável de vasos numa corda que corria sobre duas polis. O dispositivo movido por animais acionava uma roda dentada permitindo que os vasos entrassem na água, enchessem e depois fossem levantados e esvaziados [13].

À medida que as técnicas foram sendo desenvolvidas, as sociedades no Egipto e na China construíram canais de irrigação, represas, diques e instalações de armazenamento de água. Roma antiga construiu estruturas chamadas aquedutos para transportar a água do degelo nos Alpes para as cidades e vilas nos vales abaixo. Esta água foi usada para beber, lavar e irrigar [14].

Os sistemas modernos de irrigação usam reservatórios, tanques e poços para fornecer água para as plantações. Os reservatórios incluem aquíferos, bacias que recolhem neve derretida, lagos e bacias criadas por barragens, canais ou aquedutos que transportam a água dos reservatórios para os campos. Os canais e aquedutos, assim como os antigos aquedutos romanos, geralmente dependem da força da gravidade. As bombas também podem mover a água dos reservatórios para os campos. Com a evolução dos sistemas de irrigação, houve uma necessidade crescente de automatizar o processo da rega.

2.1.2 Automação

Desde tempos remotos, o homem tem procurado uma melhoria da sua qualidade de vida. Sempre que possível preocupou-se em facilitar o seu trabalho, desenvolvendo artefactos que ajudassem nas sua tarefas e inventando e descobrindo outros tais como a roda ou o fogo. Com o advento da automação, o homem conseguiu desenvolver técnicas e equipamentos que o fez produzir mais e melhor, o que permitiu ter condições

de vida melhores. O processo de automação não se resume ao âmbito industrial, tem desempenhado um papel fundamental no avanço da engenharia e da ciência, sendo extremamente importante na corrida espacial, aviação comercial, indústria militar, entre outras aplicações.

Um dos primeiros dispositivos automáticos da humanidade foi o relógio de água, desenvolvido em meados do século II AC., facilitando a medição do tempo. O homem nunca se deu por vencido e sempre procurou novos caminhos para o seu desenvolvimento. A revolução industrial do século XVIII trouxe a máquina a vapor e, com ela, James Watt[15], desenvolvendo o primeiro controlador automático para um processo industrial conforme ilustra a Figura 2.2. Este controlador baseia-se na força centrífuga exercida em duas esferas, de acordo com o aumento da velocidade do motor: Quanto maior era a força centrífuga que deslocava as esferas para fora, o mecanismo de válvulas fechava-se, deixando passar uma quantidade menor de vapor. À medida que a velocidade de rotação diminuía, a força centrífuga era menor, levando a que as esferas ficassem mais próximas do centro e, por consequência, à abertura das válvulas, levando a uma maior passagem do vapor. Deste modo era possível controlar a velocidade da máquina a vapor.

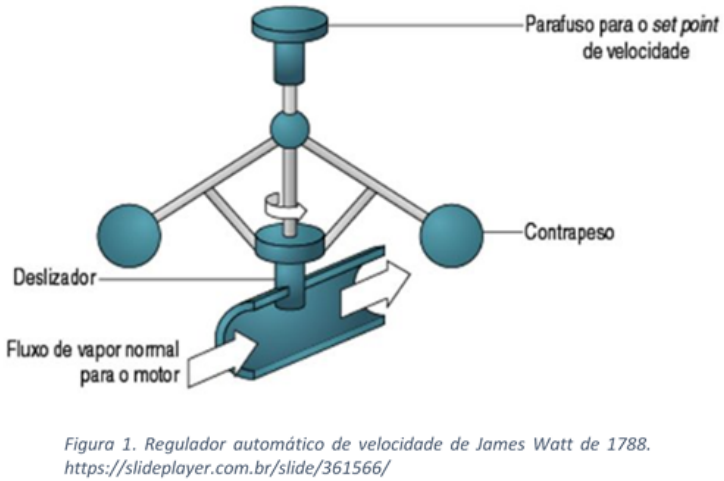

Figura 2.2: Regulador automático de velocidade de James Watt de 1788. (*in* https://slideplayer.com.br/slide/361566/)

A história da automação tem muitos colaboradores, desde Isaac Newton [16], que lançou os fundamentos da modelação matemática e da análise, até os cientistas de hoje, que conseguem enviar um míssil, controlado por satélites, a milhares de quilómetros de distância acertando num alvo, com um uma precisão de poucos centímetros. A indústria militar é o maior patrocinador do desenvolvimento de tecnologias em todas as áreas, e a automação não fica de fora. O grande desenvolvimento das indústrias trazem para si a implementação da automação, aumentado a qualidade e a produção, deixando os processos menos complexos e menos onerosos, baixando os custos de produção e

aumentando a eficiência. Em conjunto com a automação, a eletrónica que invadiu as indústrias, proporcionou a automação destas.

O grande papel da automação industrial, não é eliminar o homem do processo industrial, mas sim, integrar diferentes processos, seguindo um projeto e uma gestão administrativa e financeira. Podemos então definir automação industrial como "a técnica, método ou sistema de operar ou controlar um processo por meios altamente automáticos, como por dispositivos eletrónicos, reduzindo a intervenção humana ao mínimo" [17].

2.1.3 Evolução dos controladores de rega e sensores

Com a evolução da automação e a necessidade crescente de automatização dos processos de rega, surgiram na década de 50 os primeiros controladores de rega e os primeiros sensores de solo. Com o decorrer dos anos paralelamente ao avanço na automatização existiu uma evolução dos controladores de rega. Na Figura 2.3 podemos observar um exemplo de um controlador de rega hidráulico modelo "K" fabricado pela Moody Sprinkler Co., Los Angeles, Califórnia, EUA em 1950. O seu funcionamento baseava-se num relógio com batentes de metal inseridos em buracos num mostrador que fazia atuar válvulas hidráulicas.

Figura 1. Controlador de rega hidráulico de 1950.
Fonte: http://www.irrigationmuseum.org.

Figura 2.3: Controlador de rega hidráulico de 1950. (*in* http://www.irrigationmuseum.org.)

Na Figura 2.4 pode ser observado um exemplo de um tensiómetro fabricado pela Irrometer Co., Riverside Califórnia, EUA em 1951. Este sensor permite indicar o esforço que devem realizar as raízes para extrair do solo a humidade que necessita a cultura. A terra seca extrai líquido do tensiómetro produzindo um vazio parcial no instrumento que fica indicado no vacuómetro. Quanto mais seca a terra, mais alto o valor registado

no mostrador do vacuómetro. Ao humedecer-se a terra, como consequência da chuva ou de uma rega, o Irrometer volta a absorver humidade do solo, com isso se reduz a tensão e o vacuómetro indica um valor inferior até chegar a zero, isso indica que a terra alcançou outra vez a sua capacidade máxima de retenção de humidade que denominamos "capacidade de campo".

Figura 2.4: Tensiómetro de solo de 1951. (*in* http://www.irrigationmuseum.org)

Na Figura 2.5 podemos observar o primeiro controlador de rega hidráulico de seis estações "Moist-O-Matic", modelo C-6 para uso comercial e residencial desenvolvido por Ed Hunter em 1958 nos EUA.

Figura 2.5: Controlador de rega Moist-O-Matic de 1958. (*in* http://www.irrigationmuseum.org)

Na Figura 2.6 podemos observar o primeiro controlador de rega eletrónico. Este controlador de rega, era controlado por um microcontrolador e é o primeiro protótipo para a série KCS de controladores fabricados por Johns Manville nos EUA em 1976. Um dos aspetos únicos deste controlador é o painel de exposição feito em fibra de vidro. Os modelos de produção, eram de plástico e entraram em produção em 1978. Este controlador permitia a programação por 14 dias consecutivos e a programação individual

de cada estação. Permitia uma operação manual ou automática e possuía um botão *"rain"* para interromper o funcionamento caso estivesse a chover.

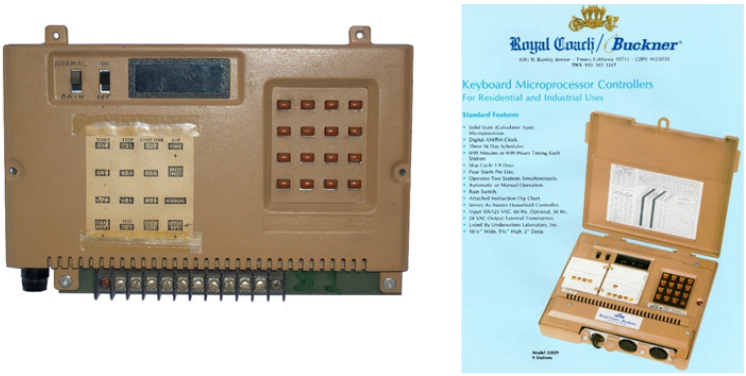

Figura 2.6: Controlador de rega eletrónico KCS de 1980. (*in* http://www.irrigationmuseum.org)

A Figura 2.7 representa um sensor de humidade do solo fabricado pela Companhia Irrometer em 1985.

Figura 2.7: Sensor de humidade do solo de 1985. (*in* http://www.irrigationmuseum.org)

O primeiro controlador de rega computorizado pode ser observado na Figura 2.8. Foi desenvolvido pela Griswold Solar Wind em 1986 nos EUA. Este controlador opera com um cartão perfurado de computador que o cliente preenche. Em seguida, o cartão é inserido no *slot* no painel frontal, e o controlador fica programado.

Na Figura 2.9 podemos observar o sensor de humidade do solo Watermark, desenvolvido pela companhia Irrometer em 2008.

Na Figura 2.10 podemos observar um tensímetro de solo fabricado pela Irrometer em 2008, 57 anos após o primeiro modelo ter surgido no mercado. Como podemos constatar pela história, os controladores de rega e os sensores que se podem acoplar a estes, têm vindo a evoluir constantemente. Hoje em dia temos controladores de rega com as mais diversas funcionalidades. Na Figura 2.11 podemos observar um exemplo de um controlador de rega atual fabricado pela Rain Machine onde as funcionalidades disponíveis são imensas.

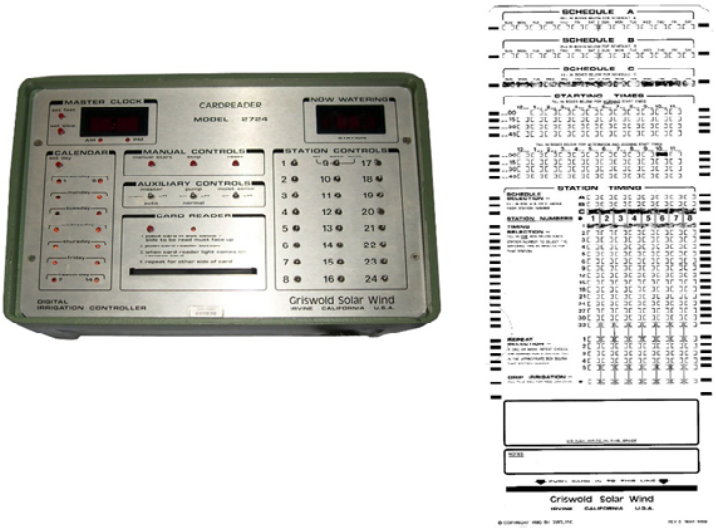

Figura 2.8: Controlador de Rega computorizado de 1986 e o cartão perfurado que permite a sua programação. (*in* http://www.irrigationmuseum.org)

Figura 2.9: Sensor de humidade do solo Watermark de 2008. (*in* http://www.irrigationmuseum.org)

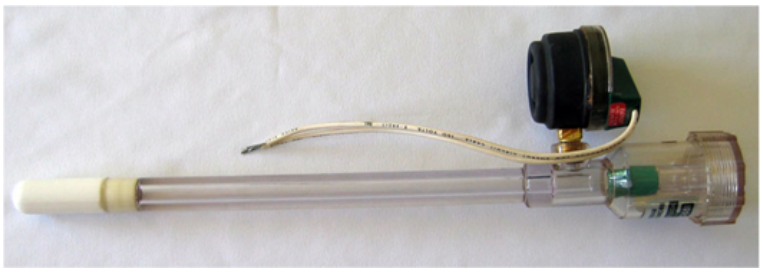

Figura 2.10: Tensímetro de solo de 2008. (*in* http://www.irrigationmuseum.org)

O Rainmachine HD-12 é um controlador inteligente de *sprinklers* WiFi de 12 zonas com ecrã táctil a cores. Utiliza previsões meteorológicas e dados de estações meteorológicas para ajustar dinamicamente os níveis de rega do seu relvado e jardim. O Rainmachine é capaz de aumentar a eficiência da irrigação, reunindo e processando informações sobre o tempo para a sua localização, com vários dias de antecedência. Não é necessário estar vinculado a nenhum terceiro provedor de serviços em *Cloud*. O Rainmachine liga-se di-

Figura 2.11: Controlador de rega Rainmachine Touch HD-12 fabricado em 2018. (*in* https://www.rainmachine.com)

retamente a fontes confiáveis de dados meteorológicos nacionais e internacionais e ajusta os valores diários de irrigação. Pode ser acedido de qualquer parte do mundo através do navegador Web para iPhone, Android e PC. Este controlador permite usar fontes de dados meteorológicos gratuitos da NOAA, METNO, Netatmo ou Wunderground simultaneamente para criar uma rede meteorológica redundante e precisa. Permite ainda ligação WiFi 802.11n de forma a poder ligar-se em qualquer zona residencial. Possui 12 zonas que podem ser controladas individualmente e pode ser usado com o Amazon Echo ou o Google Home.

2.1.4 O futuro da gestão da água

Com a tecnologia IoT em pleno desenvolvimento e as condições severas de seca afetando a maioria dos países, devido principalmente às alterações climáticas, a irrigação inteligente tornou-se mais do que apenas um programa de irrigação temporizado. Quer sejam ferramentas de orçamentação de custos com a água, aplicativos de pesquisa de *sites*, aplicativos para *smartphones* ou monitorização da qualidade da água por parte das empresas de distribuição de água, o objetivo é o mesmo. Trata-se de usar os dados e o conhecimento que os dados fornecem, de forma a ajudar as organizações a tornar os seus negócios mais inteligentes. Na indústria de irrigação inteligente, esta utilização de transformar dados em conhecimento fornece uma poderosa combinação de precisão, visibilidade e eficiência. Acredito que o próximo passo evolucionário para alcançar todo o potencial da irrigação verdadeiramente "inteligente" é, numa palavra, fluxo. A visibilidade e o controlo do fluxo garantem que, em última instância, a gestão inteligente da água está a ser mantida 24 horas por dia, 7 dias por semana. Existem muitos *sites* e aplicações que disponibilizam a informação dos controladores inteligentes de rega, mas que não possuem controlo de fluxo, no entanto, permitem de uma ou outra forma economizar água.

Desvalorizar a monitorização do uso da água em tempo real, irá mais tarde ou mais

cedo ter custos associados. A demanda por soluções com medição de fluxo precisas a um custo competitivo nunca foi tão elevada. Com a medição do fluxo de água em tempo real, os controladores inteligentes de irrigação ficam ainda mais inteligentes, tornando possível identificar fugas de água, interrupções no sistema de rega, erros de configuração do controlador e até furtos de água. Contudo, muitos sensores de fluxo perdem a precisão em baixas taxas de fluxo, o que pode levar a leituras imprecisas em sistemas de baixo fluxo, como por exemplo a rega gota a gota. Soluções acessíveis de medição de fluxo *retrofit* equipadas com sensores de fluxo de alta resolução, que medem com precisão os sistemas de rega gota a gota de alta eficiência, estão agora disponíveis para ajudar neste problema. Apesar da maioria os *sites* e aplicações disponibilizarem informação da rutura da linha de água num ponto ou outro, mais de 90% não tem nenhum tipo de gestão de fluxo. A adição de tecnologia de medição de fluxo a um controlador de irrigação inteligente com comunicação bidirecional fornece dados em tempo real do uso da água, mas também o controlo remoto imediato, de forma a efetuar mudanças imediatas nos sistemas de rega. Essa tecnologia facilita a resposta a ruturas e vazamentos catastróficos, protege os ativos imobiliários e gere os recursos hídricos de forma eficaz. Então, qual é o próximo horizonte para novas tecnologias inteligentes de gestão da água? Acredito que as próximas áreas onde se verificarão melhorias tecnológicas que terão um impacto significativo na indústria, serão soluções de comunicação sem fios *infield* mais confiáveis e acessíveis, além de gestão aprimorada do fluxo da água baseado em *Cloud*, apresentando soluções para sites de elevada complexidade, que fornecem ao utilizador um ambiente gráfico intuitivo, interfaces que permitam balancear demandas complexas de fluxo, mas que ao mesmo tempo possuam uma curva de aprendizagem rápida. É aqui que o SWAIS se encaixa, um sistema *wireless*, confiável, energéticamente eficiente e que pode possibilitar a monitorização de diversos parâmetros em tempo real. Encontra-mo-nos numa era verdadeiramente nova, onde o tremendo poder e eficiência da tecnologia da informação, fez o seu percurso para as mãos de profissionais da agricultura e dos espaços verdes, e que utilizam todo o potencial do mundo do IoT no desenvolvimento de novos produtos e soluções. Porquê? Porque permite que os seus clientes poupem tempo, dinheiro e água. Afinal, esse é o principal objetivo evolutivo de gestão inteligente da água: menos ineficiente, mais inteligente.

2.2 Formalismos de Modelação

A fase inicial de um projeto de um dado sistema, baseado na modelação e análise de requisitos é considerada a mais complexa e a mais estratégica para a obtenção de bons resultados. Por conseguinte, os modelos gráficos permitem a deteção precoce de erros, antes de iniciar a implementação, evitando assim o desperdício de tempo e recursos. Neste contexto, novos métodos de análise, modelação, desenvolvimento e implementação

foram sugeridos e testados em várias fases do processo de um projeto. Especificamente, a fase inicial de projeto, baseada na obtenção, especificação e análise de requisitos, é considerada a fase estrategicamente mais sensível, dado que uma formalização completa do processo é inviável nesta fase. Por outro lado, as pesquisas mais recentes vão na direção da antecipação da formalização, aproximando-a cada vez mais da fase inicial de elicitação de requisitos [18]. A importância da modelação no processo de análise, especialmente quando tratamos de sistemas automatizados, isto é, que possuem uma dinâmica, e esta dinâmica é uma das principais propriedades do sistema a ser desenvolvido, é deveras relevante. Um formalismo esquemático baseado em grafos, as RdP, é apresentado como base para a análise e a verificação de formal de requisitos, minorando a possibilidade de erros e omissões que afetam a descrição do sistema a ser desenvolvido.

Os modelos discretos têm sido aplicados com sucesso na representação de uma grande variedade de sistemas, tais como sistemas de produção, redes de computadores e tráfego automóvel. Alguns sistemas podem ser mais facilmente representados por modelos capazes de alterar a sua própria estrutura.

Os formalismos de modelação oferecem várias vantagens sobre os métodos não formais na representação de modelos de simulação. Estas vantagens incluem um sistema compacto e rigoroso de notação, além de que os modelos especificados num qualquer formalismo de modelação são também independentes da linguagem de simulação. Como exemplo de formalismos de modelação, podemos olhar para os fluxogramas; bastante utilizados na descrição um processo, sistema ou algoritmo de computador. São amplamente utilizados em várias áreas para documentar, estudar, planear, melhorar e comunicar processos complexos por meio de diagramas claros e de fácil compreensão. Outro exemplo de um formalismo de modelação, as máquinas de estado finitos (*Finite State Machines - FSM*): As máquinas de estados finitos são máquinas abstratas que capturam as partes essenciais de algumas máquinas concretas. Essas últimas vão desde máquinas de vender jornais e de vender refrigerantes, passando por relógios digitais e elevadores, até programas de computador, como alguns procedimentos de editores de textos e de compiladores.

2.2.1 Redes de Petri

As Redes de Petri (RdP) são uma linguagem para modelação e desenvolvimento de sistemas usando uma representação gráfica e precisa do sistema, muitas vezes considerada uma extensão das máquinas de estado finito. A análise de um modelo de uma RdP permite a avaliação do comportamento dinâmico do sistema modelado. O resultado dessa avaliação pode levar a melhorias ou mudanças no sistema [19]. O método foi introduzido em 1962 na tese de doutoramento de Carl Adam Petri [20], cujo objetivo principal era a modelação de processos com troca de mensagens. A base formal do método foi sempre a teoria de Grafos e a base conceptual a representação de relações entre um conjunto de elementos que pode ser associada ao paradigma estado / transição, usado no desenho de

sistemas dinâmicos discretos. Desta forma, as RdP passaram a compor a base teórica dos sistemas automatizados, que na sua grande maioria são sistemas discretos, dinâmicos e concorrentes. Uma rede de Petri é um grafo composto por dois tipos de nós (lugares e transições) ligados por arcos, como ilustrado na Figura 2.12

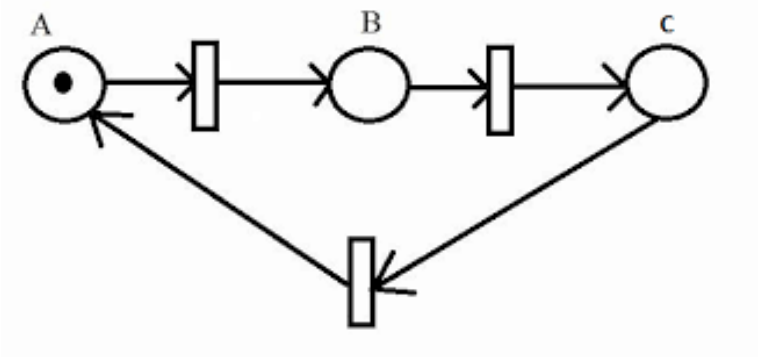

Figura 2.12: Exemplo de uma Rede de Petri. Os círculos representam os lugares e os retângulos representam as transições. O lugar "A"encontra-se marcado com uma marca.

Durante o período da sua existência, as passaram por um ciclo evolutivo que acompanha de perto a sofisticação dos sistemas automatizados e a exigência cada vez maior de sistemas com baixo índice de falhas. O aumento da aplicação e a abrangência das RdP, levou a que passassem a ser estudadas como uma representação formal independente do domínio de aplicação.

2.2.2 IOPT-nets

As IOPT-Nets [21, 22] são uma classe de RdP não autónomas, que estendem a RdP lugar-transição pois permitem a modelação explícita de entradas e saídas no modelo de rede, permitindo por exemplo a leitura de sensores através de sinais de entrada na rede ou, a geração de sinais de saída, atuando noutros dispositivos externos à rede. As IOPT-Nets permitem ainda a comunicação com outros sistemas externos enviando sinais ou gerando eventos de saída. As novas extensões desenvolvidas proporcionam as seguintes características:

- As entradas (sinais e eventos) do sistema podem ser associadas às transições;
- As saídas (sinais e eventos) do sistema podem ser associadas às transições e aos lugares;
- Definição das prioridades nas transições;
- Os lugares possuem um atributo que indica o número máximo de marcas;
- Dois tipos de valores para os sinais de entrada e de saída (inteiro e booleano);

- Uma especificação explicita para um conjunto de transições com conflitos;
- Uma especificação explicita para um conjunto de transições síncronas;
- Arcos de teste.

As IOPT-nets permitem a sua especificação em expressões algébricas, variáveis, e funções nas transições (denominadas guardas), bem como condições nas ações de entrada associadas a cada lugar da RdP.

2.3 Ferramentas de Automatização de Projeto

2.3.1 IOPT Tools

Na sua essência, as IOPT Tools [21] são um conjunto de ferramentas interactivas que consistem num editor gráfico2.13 e interativo de RdP [23, 24] de forma a projetar os modelos IOPT, um simulador [25, 26] para testar o comportamento do sistema, um gerador de espaço de estados [22] e um sistema de consulta para automatizar a verificação de modelos e propriedades [27, 28].

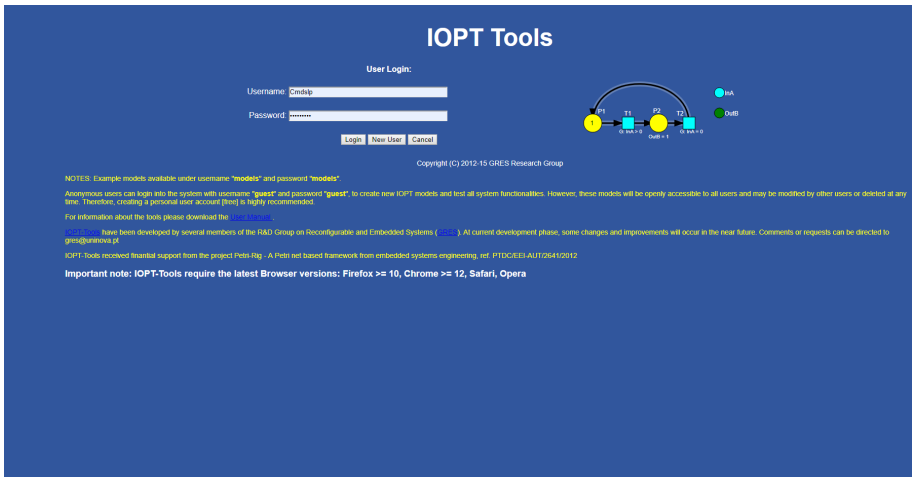

Figura 2.13: Website da ferramenta IOPT Tools desenvolvida pelo *Group on Reconfigurable and Embedded Systems* (GRES).

As IOPT-Tools são uma classe de RdP não autónomas, pois permitem a modelação explícita de entradas e saídas no modelo de rede, permitindo por exemplo a leitura de sensores através de sinais de entrada na rede ou, a geração de sinais de saída, atuando noutros dispositivos externos à rede.

AS IOPT Tools dispõem de geradores automáticos de código necessários para a implementação de controladores em sistemas embutidos, possibilitando a geração de código C, de código de descrição de hardware em VHDL, JavaScript entre outros. As

IOPT Tools possuem também uma ferramenta de *debugger* com acesso remoto, via HTTP, ao hardware do controlador onde se encontra em execução o código, previamente, gerado automaticamente, capaz de afetar e verificar as saídas / entradas / variáveis / registos do sistema em tempo real.

As características de maior importância definidas para a classe IOPT, são a introdução de sinais e eventos de entrada e saída, a definição de transições com prioridades, a definição de um limite da quantidade de marcadores em cada lugar e a possibilidade de definição de arcos de teste. A possibilidade de definir transições com prioridades e arcos de teste permite resolver conflitos inerentes às RdP, de forma injusta ou justa, respetivamente. A definição do limite de 8 marcas nos lugares permite que as ferramentas de geração de código automático criam os recursos de memória necessários à descrição dos lugares.

Além das características supracitadas, podemos adicionar a interface gráfica da edição de modelos RdP IOPT, a validação / simulação de modelos, a monitorização remota do sistema em tempo de execução e a geração automática de código. Todas estas ferramentas incluindo o respetivo manual [29], podem ser encontradas no website pertencente ao GRES: *http://gres.uninova.pt/IOPT-tools/login.php*.

Editor Gráfico IOPT Tools

Como extensão da classe de redes de Petri Lugar-Transição, IOPT-nets, o ambiente gráfico de edição da ferramenta IOPT Tools contém um painel adaptado para o desenvolvimento de modelos IOPT e um caixa de ferramentas contendo o design não só para lugares, marcas, transições e arcos, mas também blocos para os sinais de entrada e saída, blocos para os eventos de entrada e saída e os arcos de teste, entre outros, como ilustrado na Figura 2.14.

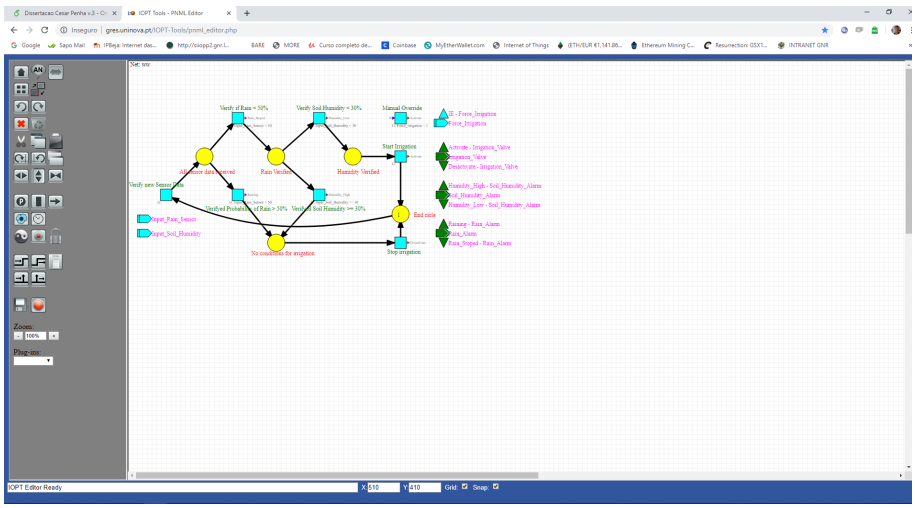

Figura 2.14: Editor gráfico da ferramenta IOPT Tools.

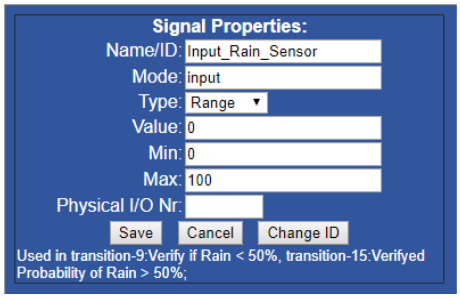

Figura 2.15: Editor propriedades dos sinais.

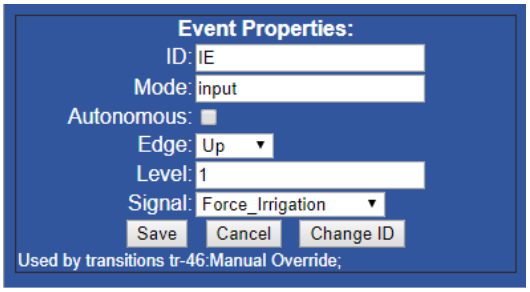

Figura 2.16: Editor propriedades dos eventos.

Ao adicionar um sinal ao modelo IOPT são apresentadas as propriedades inerentes a esse mesmo sinal, nomeadamente, o nome do sinal, o tipo de valor (booleano ou inteiro), o valor mínimo e máximo e por último o número da porta física da qual será realizada a leitura ou afetação do sinal na plataforma de desenvolvimento conforme pode ser verificado na Figura 2.15. O mesmo acontece em relação ao evento; Um evento encontra-se diretamente relacionado com um sinal e este ocorre quando ocorre uma determinada alteração ao sinal, conforme pode ser observado na Figura 2.16.

Validação e Simulação de Modelos

Tal como já referenciado, as IOPT Tools possuem uma ferramenta de simulação e uma ferramenta de geração de espaço de estados de forma a verificar / confirmar o funcionamento do controlador, não havendo necessidade de usar plataformas de teste físicas.

A ferramenta de simulação da Figura 2.17 é composta por uma coluna de controlos, que contêm as seguintes funcionalidades:

- Iniciar;
- Parar;
- Verificar passo a passo a execução do controlador;
- Extração do histórico de execução do controlador;
- Visualização da execução nos instantes de tempo anteriores.

Na zona central da ferramenta é apresentado o modelo IOPT, onde é utilizado a cor vermelha para os lugares e laranja para as transições, facilitando a visualização das transições habilitadas e dos lugares que as habilitam. Por fim, na zona direita da Figura 2.17 são apresentados todos os lugares, sinais e eventos bem como os respetivos valores para cada instante de tempo durante a execução da simulação do modelo.

A ferramenta de geração de espaço de estados observada na Figura 2.18, permite aos utilizadores verificarem o funcionamento do modelo IOPT para todas as possíveis

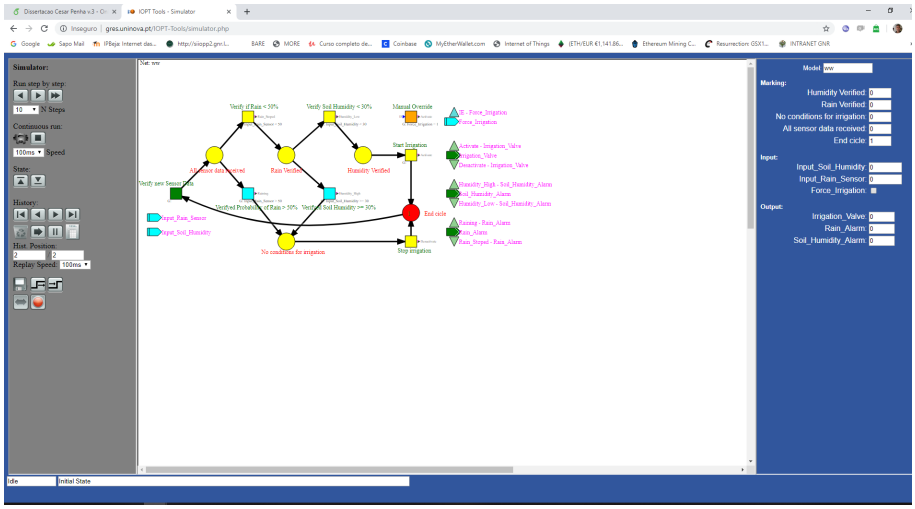

Figura 2.17: Editor gráfico da ferramenta IOPT Tools.

de sequências de estados do controlador. A partir das marcas iniciais em cada lugar é validada a possibilidade de existência de *deadlocks* comummente designado por um estado de bloqueio do controlador, conflitos e estados inválidos, ilustrando em forma de texto ou graficamente.

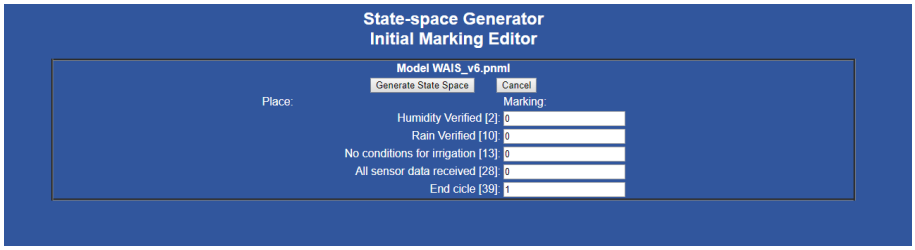

Figura 2.18: Editor gráfico da ferramenta IOPT Tools.

Geração automática de código

Os geradores de código permitem que os modelos desenvolvidos com as RdP IOPT sejam implementados em diversas plataformas possibilitando ao desenvolvedor uma vasta gama de plataformas de teste/execução. Em particular, o gerador de código C gera um determinado numero de ficheiros que permitem a implementação do código em diversas plataformas, tais como plataformas da família Arduino. Seguidamente são apresentadas as descrições de cada ficheiro gerado pela ferramenta IOPT Tools:

- **net_main.c**: Este ficheiro contém a definição do ciclo de execução do modelo IOPT;
- **net_function.c**: Neste ficheiro são declaradas as funções para implementação das semânticas e regras da rede de Petri IOPT;

- **net_types.c**: O ficheiro contém a definição dos tipos de dados para a marcação de cada lugar, sinais de entrada ou saída e eventos. Declaração dos métodos associados ao comportamento da rede;
- **net_server.c**: O ficheiro possuí funções para a associação do modelo IOPT com o *Ethernet Shield* do Arduino para a implementação do protocolo HTTP;
- **net_server.h**: O ficheiro define as funções da biblioteca para o documento net_server.c, onde se encontram declaradas as funções para implementação do servidor HTTP apenas para plataformas Arduino;
- **net_io.c**: O ficheiro contém a definição de quais as entradas e saídas que irão ser utilizadas pelo controlador;
- **net_dbginfo**: O ficheiro permite a realização de debug da rede, forçar valores nas entradas e obter informação sobre os sinais de entrada e saída, marcas e disparos de transições;
- **net_exec_step.c**: Neste ficheiro encontram-se todas as transições utilizadas no desenvolvimento do modelo IOPT, bem como o local da realização do disparo da transição;
- **http_server.c**: O ficheiro é constituído pela implementação de uma comunicação, monitorização do servidor e HTTP debug.
- **linux_sys_gpio.c**: O ficheiro permite a utilização das portas físicas associadas ao controlador em sistemas Linux. Sendo implementadas as funções para definição das portas de entrada ou saída e respetivamente a escrita e leitura de sinais;
- **raspi_mmap_gpio.c**: O ficheiro permite a leitura e escrita nas portas de física nas plataformas Raspberry Pi de qualquer geração, sendo mais rápida a utilização deste em relação ao uso do documento linux_sys_gpio.c;
- **dummy_gpio.c**: O ficheiro permite que os sinais de entrada sejam gerados a partir de um documento de texto (inputs.txt), e o registo das saídas sejam guardados para outro documento de texto (outputs.txt);
- **Makefile**: A utilização deste ficheiro é opcional. Construtor de projeto Gnu Make/Unix com instruções para construção do projeto.

Os ficheiros gerados possuem diversas características que permitem a comunicação HTTP e a definição das portas de entrada e saída, bem como funções de leitura e escrita nas mesmas e que serão utilizadas pelo modelo. Estas características são essenciais para o funcionamento do sistema, permitindo a gerir a execução e os valores dos sinais de entrada e de saída do sistema.

2.3.2 IOPT Flow

As RdP têm sido usadas tradicionalmente para modelar comportamento de sistemas reativos, cujo estado evolui dependendo da interação com eventos externos. Pelo contrário, os formalismos de fluxo de dados foram usados predominantemente para descrever sistemas baseados em dados que produzem dados de saída transformações matemáticas aplicadas a sinais de entrada. A estrutura de modelação IOPT-Flow é um conjunto de ferramentas baseadas na Web para o desenvolvimento de controladores de sistemas embutidos e ciber-físicos, compreendendo um editor gráfico, um simulador e ferramentas automáticas de geração de código. Os modelos são projetados usando um formalismo híbrido combinando RdP e fluxos de dados para especificar as partes reativas e acionadas por dados dos controladores. A composição de modelos baseada em componentes simplifica o desenvolvimento de sistemas complexos, que podem ser montados com a ajuda de uma biblioteca de componentes previamente projetados. A sinergia da interação entre fluxo de dados e elementos de RdP traz muitas vantagens ao projeto de sistemas mistos que devem realizar simultaneamente operações de processamento de dados em sinais de entrada e gerir máquinas de estado complexas acionadas por eventos. Os controladores resultantes podem ser implementados em software ou dispositivos de hardware reconfiguráveis, como as ferramentas de geração de código [30, 31].

2.3.3 TOMSPIN

O objetivo básico do TOMSPIN (Ferramenta para Modelação com Redes de Petri Estocásticas) é fornecer aos utilizadores industriais uma ferramenta que pode analisar sistemas razoavelmente grandes usando modelos GSPN *(Generalized Stochastic Petri Nets)* com algumas extensões. O *front-end* do TOMSPIN consiste numa linguagem de descrição de rede estruturada hierarquicamente, com um editor gráfico opcional. Depois de transformar a rede numa cadeia de Markov, ela pode ser avaliada por análise de estado estacionário ou transiente. Para a análise de estado estacionário, vários algoritmos podem ser usados tais como o Gau, spl beta ou o Seidel. O TOMSPIN tem sido usado para avaliar uma ampla gama de sistemas complexos de computação comercial, particularmente considerando aspectos de desempenho e confiabilidade [32].

2.3.4 Spin

O verificador do modelo Spin é um sistema que foi usado para modelar e analisar um grande número de aplicações em vários domínios, incluindo a indústria aeroespacial. Uma das novidades do Spin é sua linguagem de especificação relativamente simples, Promela, assim como as poderosas habilidades do verificador de modelos. A notação de rede de Petri é uma ferramenta matemática para modelar várias classes de sistemas, especialmente aqueles que envolvem simultaneidade e paralelismo. O Ambiente de

modelação de Domínios da Honeywell (DOME) é uma ferramenta que suporta o projeto de sistemas usando uma ampla variedade de notações de modelação, incluindo diagramas UML e RdP. A ferramenta suporta o uso do verificador do modelo Spin para analisar e verificar as especificações da RdP que foram construídas usando a ferramenta DOME. Além de discutir a tradução de redes de Petri para Promela, a ferramenta apresenta várias especificações de RdP, bem como sua análise usando Spin[33].

Capítulo 3

Arquitetura e Protótipo

Neste capítulo a são descritas as contribuições deste trabalho. É proposta uma arquitectura do sistema onde são descritas as características do hardware e do protocolo de comunicação. É apresentado o modelo gráfico que serve de base ao sistema, bem como a ferramenta de automatização de projeto e um modelo exemplificativo. São descritas as alterações e as funcionalidades referentes aos códigos provenientes das características do hardware e do protocolo de comunicação e a sua integração. Por fim, é apresentado o protótipo funcional SWAIS como forma de validação da ferramenta de automatização de projeto e da própria arquitetura do sistema.

3.1 Arquitetura Proposta

A arquitetura de um sistema inclui as principais propriedades físicas, estilo, estrutura, interações e finalidade de um dado sistema. A arquitetura é composta na sua essência por diversos elementos que se relacionam entre si, e cuja finalidade é obter um código pronto a implementar num microcontrolador e a própria programação do mesmo. A "espinha dorsal"do sistema é composta pelas características, tanto do hardware como do protocolo de comunicação, bem como o modelo gráfico. A arquitetura a desenvolver, deverá contemplar nas suas características a escolha dos sensores e a forma como são escolhidos. Essa escolha poderá ser determinada por diversos parâmetros tais como a distância, a força do sinal recebido ou outras achadas convenientes. O sistema deverá reconfigurar-se automaticamente de forma a que a lista de sensores esteja sempre optimizada e atualizada. Seguidamente será descrita a funcionalidade de cada um dos elementos constituintes e o seu comportamento dentro do sistema.

3.1.1 Características de Hardware

As características do hardware contemplam o tipo de hardware que será usado na construção do sistema. Essas características são programadas através de código de-

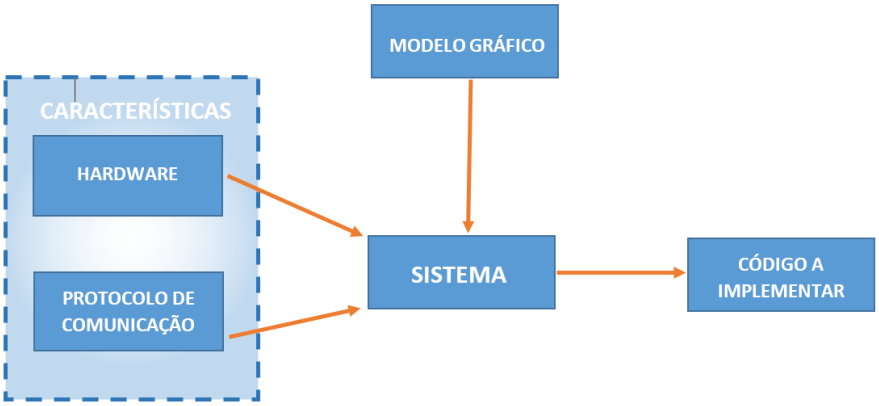

Figura 3.1: Arquitetura do sistema.

senvolvido de forma manual e podem contemplar o tipo de plataforma a ser utilizado como por exemplo ATMEL, MSP430, Raspberry Pi, PC ou outros. Elas definem uma série de propriedades relativas à plataforma a utilizar e o comportamento desta no sistema.

3.1.2 Características do Protocolo de Comunicação

As características do protocolo de comunicação definem o tipo de comunicação sem fios a ser usada no sistema entre os dispositivos constituintes. Esta comunicação sem fios poderá ser de diversos tipos tais como LoRa, ZigBee, BLE ou outros. O desenvolvimento das características de cada protocolo de comunicação é realizado de forma manual e guardado em ficheiros de configuração. Primeiramente o sistema adquire todos os sensores que estejam ao seu alcance, descartando os sensores que não interessam ao sistema. Seguidamente os sensores guardados são ordenados por tipo, excluindo os tipos não necessários. Por último, dos sensores restantes do mesmo tipo, são escolhidos os que apresentam as melhores "condições de escolha". As condições de escolha poderão ser de diversa ordem tais como, a potência do sinal recebido, a média dos valores dos sensores encontrados que sejam do mesmo tipo ou outros. O sistema poderá também classificar outros sensores vizinhos que não possuam qualquer identificação, mediante os valores recebidos, comparando os mesmos com os sensores do sistema, e atribuindo-lhe um "tipo"ou uma classificação.

3.1.3 Modelo Gráfico

As regras que determinam a ativação ou inativação do atuador, são definidas recorrendo a uma ferramenta de modelação gráfica. O modelo disponibilizado contém todos os sinais de entrada e saída associados, permitindo ao utilizador modelar uma rede à medida

das necessidades do sistema. Será disponibilizado um modelo base ao utilizador que, seguidamente, irá modelar a parte constante às regras que interagem com o atuador.

3.2 Ferramenta de Automatização de Projeto

O código necessário ao funcionamento do SWAIS tem duas origens distintas: A primeira é desenvolvida de forma manual e contem as especificidades do hardware a ser utilizado, nomeadamente a plataforma de desenvolvimento onde assenta o projeto e o tipo de protocolo de comunicação entre os diversos dispositivos. A segunda é automaticamente gerada pela *framework* IOPT Tools, mediante o modelo de rede modelado no editor.

3.2.1 Modelo Base

O sistema utilizará a modelação gráfica como forma de desenvolver as regras que determinam o comportamento do sistema. Essas regras serão desenhadas pelo utilizador, recorrendo a um ambiente gráfico de modelação (IOPT Editor) que permitirá a geração automática de código e a sua posterior integração no sistema.

Na Figura 3.2 é apresentado o interface do modelo base do sistema. Essa interface é constituída por dois esqueletos / grafos: o primeiro é constituído por dois lugares (*"Conditions to STOP Verified"e "Verify Conditions to START"*) e uma transição (*Stop*); a segunda é constituida por uma transição (*"Start"*) e por um lugar (*"Verify conditions to STOP"*).

O primeiro grafo apresenta no primeiro lugar (*Conditions to STOP Verified* uma marca. Essa marca irá ativar a transição "STOP"cuja mesma tem associado um evento de saída "OE_Desactivate"que está conetado a um sinal de saída do tipo booleano. Esse sinal será interpretado pelo MCU e o mesmo irá desativar um atuador.

Seguidamente a marca será colocada no lugar *"Verify Conditions start"*. A partir deste ponto, o utilizador irá modelar as regras que irão inverter a situação, ou seja, as regras que irão ativar o atuador. As regras que irão ser modeladas estão representadas pelo retângulo verde da Figura 3.2.

Após as condições serem verificadas e caso sejam validadas, a marca ativará a transição *"Start"* que tem associado o um evento de saída *"OE_Activate"*. Por sua vez, este evento de saída têm associado um sinal de saída do tipo boleano. Esse sinal será interpretado pelo MCU que ativará o atuador. Seguidamente o lugar *"Verify Conditions do STOP"* será marcado. As condições que definem a mudança do estado do atuador de "Ativado"para Desativado, ou seja de 1 para 0, estão representadas pelo retângulo vermelho. Caso estas condições se verifiquem, o lugar *"Conditions to stop Verified"*é marcado e o todo o processo se repetirá.

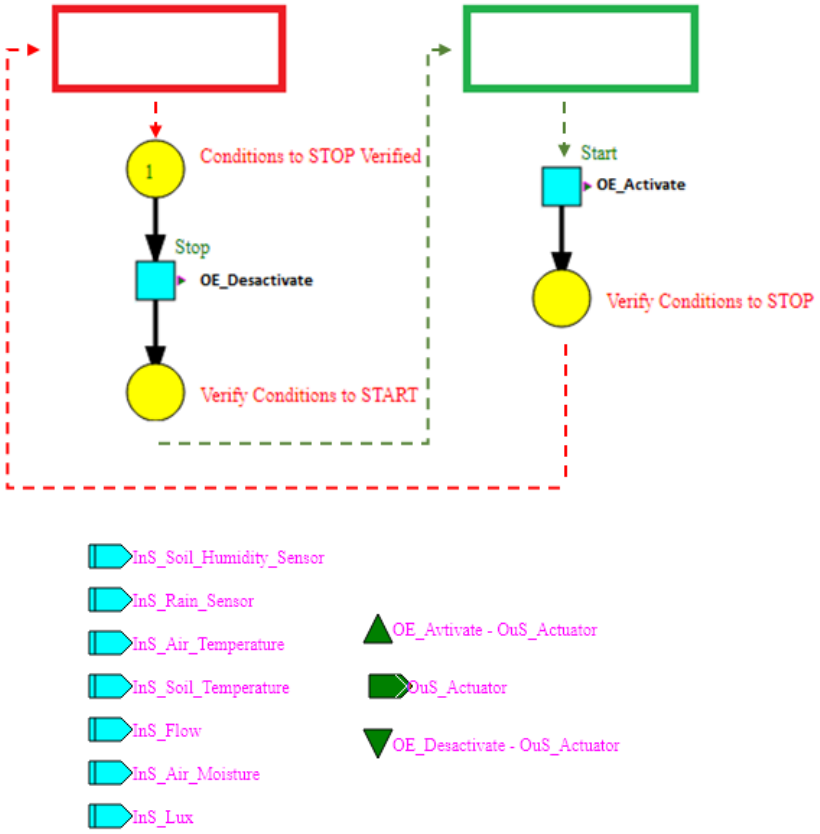

Figura 3.2: O modelo base disponibilizado ao utilizador.

Modelo Exemplificativo

Na Figura 3.3 é apresentado um modelo de rede a titulo exemplificativo. Essa interface é constituída pelo modelo base da Figura 3.2 e pelos dois esqueletos / grafos referentes ás condições de inicio e paragem da irrigação: Esses grafos foram adicionados à rede anterior na parte onde estava representado o retângulo verde e vermelho respetivamente.

A simulação da rede inicia no lugar *"Verify Conditions to Start"* com uma marca. A primeira verificação é constituída por duas transições: (*"Verify Conditions 1"* e *"Verify Conditions 2"*). Estas duas transições representam dois caminhos possíveis para que o lugar *"Conditions Verifyed"* seja marcado e se dê inicio à rega na transição seguinte (*"Start Irrigation"*).

Na primeira hipótese, a rede verifica as condições impostas pela guarda da transição *"Verify Conditions 2"*. Esta guarda contém os valores dos sensores referentes aos sinais de entrada *"InS_Lux"*,*"InS_Soil_Humidity_Sensor"* e *"InS_Manual_Override"*. Caso a condição verificada seja válida, o lugar *"Conditions Verified"* é marcado, ativando a transição seguinte *"Start Irrigation"*. A essa transição está associada a um evento de saída *"OE_ACTIVATE"* que, consequentemente, colocará o sinal de saída *"OuS_Actuator"* no

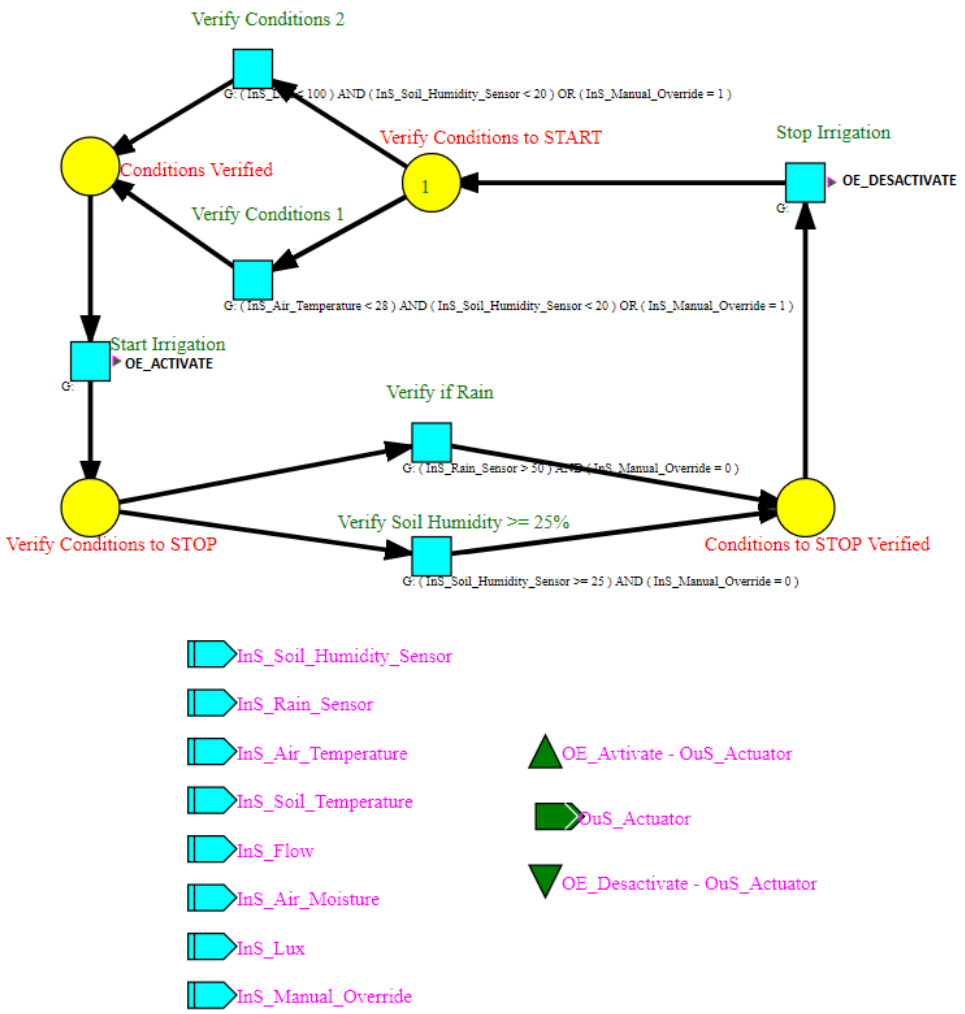

Figura 3.3: O segundo modelo da RdP , desenvolvida com a framework IOPT Tools. Neste caso foram utilizados quatro sensores (luminosidade, temperatura ambiente, chuva e humidade do solo).

valor 1, ativando a válvula selonoide da rega.

Na segunda hipótese, a rede verifica as condições impostas pela guarda da transição *"Verify Conditions 1"*. Esta guarda contém os valores dos sensores referentes aos sinais de entrada *"InS_Air_Temperature"*, *"InS_Soil_Humidity_Sensor"* e *"Ins_Manual_Override"*. Caso a condição verificada seja válida, o lugar *"Conditions Verified"* é marcado, ativando a transição seguinte *"Start Irrigation"*. A essa transição está associado o evento de saída *"OE_ACTIVATE"* que, consequentemente, colocará o sinal de saída *"OuS_Actuator"* no valor 1, ativando a válvula selonoide da rega.

Desta forma, existe uma concorrência entre estas duas transições, podendo a rega ser ativada pela primeira transição ou pela segunda transição. No entanto, as duas transições podem ser ativadas, caso as condições de ambas as guardas se verifiquem. De forma

a resolver esta última situação, e tendo em conta que o sistema deve possuir sempre o mesmo comportamento, foram definidas prioridades nas duas transições. Assim, à transição *"Verify Conditions 2"* foi dada uma prioridade de "2", e à transição *"Verify Conditions 1"* foi dada uma prioridade de "1", ou seja, mesmo que as duas transições sejam ativadas, a marca passará sempre pela transição com maior prioridade (*"Verify Conditions 1"*). Por último, caso nenhuma das transições seja ativada devido à não validação de nenhuma das guardas, a irrigação não é iniciada, ficando o sistema a aguardar por novos sinais de entrada provenientes dos sensores.

Caso haja validação de uma das guardas e a irrigação seja ativada, existe necessidade de parar a mesma quando se atinge determinados valores nos sinais de entrada. Essa verificação é logo iniciada após o começo da irrigação com a marcação do lugar *"Verify Conditions to STOP"*. Neste caso, existem duas possibilidades de parar a irrigação:

A primeira encontra-se na guarda da transição *"Verify if Rain"*, que verifica se o valor do sinal de entrada *"InS_Rain_Sensor"* é superior a 50%, e se o valor do sinal de entrada *"InS_Manual_Override"* é igual a "0". Caso a condição se verifique, o lugar *"Conditions to STOP Verified"* é marcado, o que ativará a transição *"Stop Irrigation"*. Por sua vez, esta transição tem associado um evento de saída *"OE_DESACTIVATE"* que, consequentemente, colocará o sinal de saída *"OuS_Actuator"* no valor 0, desativando a válvula selonoide da rega.

A segunda encontra-se na guarda da transição *"Verify Soil Humidity >25%"*, que verifica se o valor do sinal de entrada *"InS_Soil_Humidity_Sensor"* é igual ou superior a 25%, e se o valor do sinal de entrada *"InS_Manual_Override"* é igual a "0". Caso a condição se verifique, o lugar *"Conditions to STOP Verified"* é marcado, o que ativará a transição *"Stop Irrigation"*. Por sua vez, esta transição tem associado um evento de saída *"OE_DESACTIVATE"* que, consequentemente, colocará o sinal de saída *"OuS_Actuator"* no valor 0, desativando a válvula selonoide da rega.

Também neste caso, existe uma concorrência entre estas duas transições, podendo a rega ser desativada pela primeira transição ou pela segunda transição. No entanto, as duas transições podem ser desativadas, caso as condições de ambas as guardas se verifiquem. Tal como na situação anterior, foram definidas prioridades nas duas transições. Assim, à transição *"Verify if Rain"* foi dada uma prioridade de "1", e à transição *"Verify Soil Humidity >= 25%"* foi dada uma prioridade de "2", ou seja, mesmo que as duas transições sejam ativadas, a marca passará sempre pela transição com maior prioridade (*"Verify if Rain"*).

3.2.2 Código Específico das Características de Hardware

O código desenvolvido manualmente contempla as características do hardware, e está presente tanto nos dispositivos sensores como nos dispositivos atuadores. Nos dispositivos sensores, todo o código é desenvolvido de forma manual e é responsável pelas seguintes ações:

- Criar o *payload* a enviar que contenha o tipo e o valor do sensor;
- Criar um objeto JSON que contem o *payload* a enviar para o dispositivo atuador;
- Recolher os dados do sensor acoplado e realizar os cálculos necessários para obtenção do valor pretendido caso seja necessário;
- Configurar o módulo de comunicação LoRa;
- Enviar o *payload* para o dispositivo atuador;
- Gerir o modo de baixo consumo do microcontrolador Amtel ATMega 328P utilizado nos dispositivos sensores.

Nos dispositivos atuadores, também foi desenvolvido código de forma manual que é responsável pelas seguintes ações:

- Configuração do módulo de comunicação LoRa;
- Recolha do *payload* enviado pelos dispositivos sensores;
- Decomposição do objeto JSON recebido de forma a retirar todos os valores nele contidos;
- Guardar todos os sensores dos diferentes tipos num array;
- Escolher os sensores com melhor RSSI de entre aqueles do mesmo tipo;
- Determinar quando realizar um *reset* ao dispositivo atuador.

3.2.3 Código Gerado pela *Framework*

Como já referenciado, a *framework* IOPT Tools suporta a geração automática de código ANSI C para uma ampla gama de plataformas [19]. Esse código é responsável pelas regras modeladas na *framework*, no entanto, para o que o sistema funcione corretamente é necessário integrar o código desenvolvido manualmente com o código gerado de forma automática. Além da integração dos dois códigos, é necessário proceder a alterações cirúrgicas no código e nos ficheiros gerados de forma automática. Essas alterações visam adaptar o código à plataforma de desenvolvimento utilizada no SWAIS, a plataforma Arduíno. As alterações resumem-se basicamente à renomeação de arquivos gerados com a extensão .c para arquivos com a extensão .ino, de modo que possam ser editáveis e compiladas pelo IDE da plataforma Arduíno, à definição e associação de algumas variáveis globais com os sinais de entrada e saída da rede e à inicialização de determinadas funções necessárias.

3.2.4 Alterações aos Códigos

As alterações efetuadas aos códigos referentes às propriedades do hardware e da comunicação, têm como finalidade adaptar os mesmos às especificidades da plataforma que irá ser usada como base do sistema, bem como ao hardware utilizado na comunicação entre dispositivos. Estas alterações são necessárias, de forma a que o programa possa ser compilado e executado corretamente.

Na listagem 3.1 pode-se observar a função *"WAIS_v_7_GetInputSignals"* gerada de forma automática no ficheiro *"net_io"*. Os sinais de entrada da rede *"InS_Rain_Sensor"* e *"InS_Soil_Humidity_Sensor"*são associados às variáveis globais *"inputIOPTrainSensorValue"* e *"inputIOPTSoilHumiditySensorValue"*. Estas variáveis provêm do código desenvolvido manualmente referente às propriedades do hardware, e contêm os valores dos sensores.

Listagem 3.1: Função que atribui as variáveis globais que contêm os valores dos sensores aos sinais de entrada da rede modelada.

```
1  /* Read all hardware input signals and fill data-structure */
2
3  void WAIS_v_7_GetInputSignals(WAIS_v_7_InputSignals* inputs, WAIS_v_7_InputSignalEvents*
       events ){
4
5      inputs->InS_Rain_Sensor = inputIOPTrainSensorValue;
6      inputs->InS_Soil_Humidity_Sensor = inputIOPTSoilHumiditySensorValue;
7  }
```

Na listagem 3.2 podemos observar a função *"WAIS_v_7_PutOutputSignals"* gerada de forma automática no ficheiro *"net_io"*. A esta função foram adicionadas as linhas nº9 a nº13 e têm como finalidade atribuir os sinais de saída da rede modelada ao hardware do sistema. Na linha nº 9 o sinal de saída *"OuS_Valve"*, do tipo booleano, associado ao evento *"event_out"* é verificado. Se o *output* for igual a 1, a válvula de irrigação associada ao pino nº 10 do MCU é ativada dando inicio à irrigação (linha nº 10); caso contrário, a válvula é desativada (linha nº 12).

Listagem 3.2: Função que atribui os sinais de saída da rede ao hardware físico do dispositivo atuador.

```
1  /* Write all output values to physical hardware outputs */
2
3  void WAIS_v_7_PutOutputSignals(
4              WAIS_v_7_PlaceOutputSignals* place_out,
5              WAIS_v_7_EventOutputSignals* event_out,
6              WAIS_v_7_OutputSignalEvents* events )
7  {
8
9      if (event_out->OuS_Valve == 1){
```

```
10          digitalWrite (10, HIGH);
11      }else{
12          digitalWrite (10, LOW);
13      }
14  }
```

No ficheiro *"swais_main"*, desenvolvido de forma manual, procedeu-se também a alterações de pormenor conforme o apresentado nas listagens 3.3, 3.4 e 3.5, onde foram acrescentadas linhas de código provenientes do ficheiro *"main"* do código gerado de forma automática. A opção de copiar estas linhas do ficheiro gerado de forma automática para o ficheiro desenvolvido manualmente, e não o contrário, teve por base a quantidade de código a mover ser inferior neste sentido.

Listagem 3.3: Função setup no ficheiro *swais_main* que inicializa funções geradas automaticamente necessárias ao bom funcionamento do programa.

```
1
2  void setup(){
3
4      createInitial_WAIS_v_7_NetMarking( &marking );
5      init_WAIS_v_7_OutputSignals( &place_out, &ev_out );
6      WAIS_v_7_InitializeIO();
7      WAIS_v_7_GetInputSignals( &prev_inputs, NULL );
8        ...
9  }
```

Listagem 3.4: Função loop no ficheiro *swais_main* regula a execução dos passos da rede modelada.

```
1
2  void loop(){
3
4      if ( trace_control != TRACE_PAUSE )
5        WAIS_v_7_ExecutionStep( &marking, &inputs, &prev_inputs, &place_out, &ev_out );
6      else WAIS_v_7_GetInputSignals( &inputs, NULL );
7      if ( trace_control > TRACE_PAUSE ) --trace_control;
8       ...
9  }
```

Listagem 3.5: Declaração de variáveis globais no ficheiro *swais_main* necessárias ao bom funcionamento do programa.

```
1      int trace_control = TRACE_CONT_RUN;
2      static WAIS_v_7_NetMarking marking;
3      static WAIS_v_7_InputSignals inputs, prev_inputs;
4      static WAIS_v_7_PlaceOutputSignals place_out;
```

```
5    static WAIS_v_7_EventOutputSignals ev_out;
```

3.2.5 Aplicação swaisApp

Numa primeira fase de desenvolvimento do projeto, a integração dos dois códigos, automático gerado pela *framework* IOPT Tools e o código proveniente das caracteristicas do *hardware* e das características da comunicação, foi realizada de forma manual, ou seja, foram alterados os ficheiros copiando linhas de código necessárias no IDE do Arduíno. Após este passo, recorreu-se ao IDE do Arduíno para compilar e programar o microcontrolador, validando as alterações efetuadas. Apesar de ser uma possível solução, depressa chegou-se à conclusão que esta solução apresenta um elevado grau de erro na cópia de linhas de código entre ficheiros. Além da possibilidade de erro, é um processo moroso e é necessário ao utilizador final ter conhecimentos de programação que o permitam saber quais as linhas necessárias a copiar. Outra dificuldade que esta solução apresenta, assenta na necessidade de recorrer ao IDE do Arduíno de forma a poder compilar o código e envia-lo para o microcontrolador. Tendo em consideração os pontos apresentados anteriormente, estavam lançados os dados para uma solução que visava automatizar todos estes processos, levando ao desenvolvimento da aplicação "swaisApp". A aplicação swaisApp permite que o utilizador consiga de uma forma automática e intuitiva, integrar ambos os códigos acima referidos.

Esta aplicação foi desenvolvida utilizando a linguagem Pyton, e contém um ambiente gráfico *user friendly* onde o utilizador ao premir um botão, desencadeia um conjunto de operações que levam ao objetivo final de ter a plataforma programada, pronta a implementar no terreno. Desta forma, e com o intuito de melhor compreender o processo de automatização, podemos enumerar quais os passos e as operações que a aplicação desenvolve, de forma a atingir os objetivos propostos:

1. Extrai o ficheiro com extensão .zip gerado automaticamente pela ferramenta IOPT Tools que contem os ficheiros com extensão .c e .h para as várias plataformas, para uma nova pasta chamada *"automaticCode"*;

2. Apaga os ficheiros desnecessários à plataforma utilizada criados automaticamente pela framework IOPT Tools;

3. Renomeia as extensão dos ficheiros .c para a extensão .ino de forma a poder ser compilado pelo compilador do IDE do Arduíno;

4. Copia os ficheiros com o código desenvolvido manualmente já com a nova extensão para a pasta *"automaticCode"*;

5. Escreve no ficheiro *"tmp_iopt.ino"* a nova função *setup* que contém as novas linhas de código provenientes da geração automática de código;

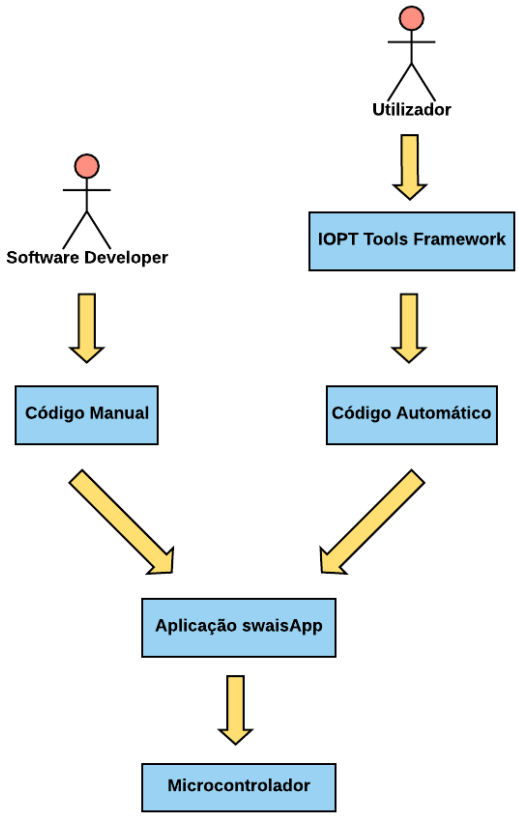

Figura 3.4: Integração dos códigos relativos ao hardware e à *framework* IOPT Tools

6. Abre o ficheiro *"main"* desenvolvido manualmente e lê as linhas da função *"loop"*. Seguidamente abre o ficheiro *"tmp_iopt.ino"* e escreve essas linhas lidas;

7. Lê as linhas relativas aos sinais de entrada do ficheiro net_io.ino e substitui as mesmas pelas variáveis globais que contêm o valor dos sensores provenientes do código desenvolvido manualmente;

8. Lê as linhas relativas aos sinais de saída do ficheiro *"net_io.ino"* e substitui as mesmas pelas linhas onde são definidos o pino do microcontrolador a ativar em caso do sinal de saída ser 0 ou ser 1;

9. Renomeia o ficheiro temporário *"tmp_iopt.ino"*, criado para efetuar todas as alterações, para o ficheiro definitivo *"swais_main.ino"* e copia a mesmo para uma nova pasta de projeto *"swais_main"*;

10. Remove todos os ficheiros denecessários contendo o código desenvolvido manualmente da pasta *"swais_main"*;

11. Verifica as portas de comunicação COM disponíveis;

12. Envia o código final para o microcontrolador utilizando o compilador *"avrdude"*;
13. Disponibiliza uma mensagem ao utilizador a informar que todo o processo foi concluído com êxito.

Na Figura 3.5 podemos observar uma vista geral da aplicação *"swaisApp"*, onde na janela central é visivel o Log da aplicação com a execução dos respetivos passos de execução suprareferenciados.

Apesar da aplicação estar preparada para programar integrados Amtel Atmega 328P, existe possibilidade de a mesma programar outros microcontroladores. Para que isso seja possível a função *"sendToMCU"* do ficheiro *"swaisGenerator2"* terá que ser revista e adaptada, nomeadamente na linha nº 10 da listagem 3.6 no parâmetro referente ao tipo de MCU *"patmega328p"*.

Listagem 3.6: Função "sendToMCU"pertencente à aplicação swaisApp

```
1
2  def sendToMCU():
3      print('Compile code to MCU...')
4      # Change to project folder
5      os.chdir("/")
6      os.chdir("C:/Users/Cesar Penha/AppData/Local/Temp/arduino_build_752414")
7      path_avrdude = 'C:/Program Files (x86)/Arduino/hardware/tools/avr/etc/avrdude.conf'
8      path_avrdude = os.path.normpath(path_avrdude);
9      pathFile = ('avrdude -v -patmega328p -carduino -PCOM9 -b57600 -D
           -Uflash:w:IOPT_20032019.ino.hex -D -C "'+path_avrdude+'"')
10     os.system(pathFile)
11     print('OK!\n\n')
```

Existem ainda outros dois aspetos a ter em conta, aquando da eventual alteração da máquina em que a aplicação está a correr: A primeira prende-se com o fato de que o caminho onde se encontra o compilador (linha nº 7) terá que ser ajustado consoante o PC ou, o mesmo terá que ser incluído na "PATH"do sistema operativo, de forma a correr independentemente do caminho que nos encontramos. A segunda refere-se ao caminho onde se encontra o ficheiro a compilar (linha nº 6); o mesmo poderá diferir caso se mude de máquina.

Estes procedimentos serão objeto de revisão num trabalho futuro de aprimoramento da aplicação swaisAPP, de forma a que o utilizador insira o caminho onde se encontra a ficheiro a tratar, ou que a aplicação construa o caminho com base no ficheiro selecionado no ambiente gráfico da aplicação.

3.2.6 Verificação Automática do Sinal

Como já referenciado, este trabalho apresenta uma abordagem para desenvolver sistemas IoT reconfiguráveis usando tecnologias sem fios. O sistema a implementar poderá con-

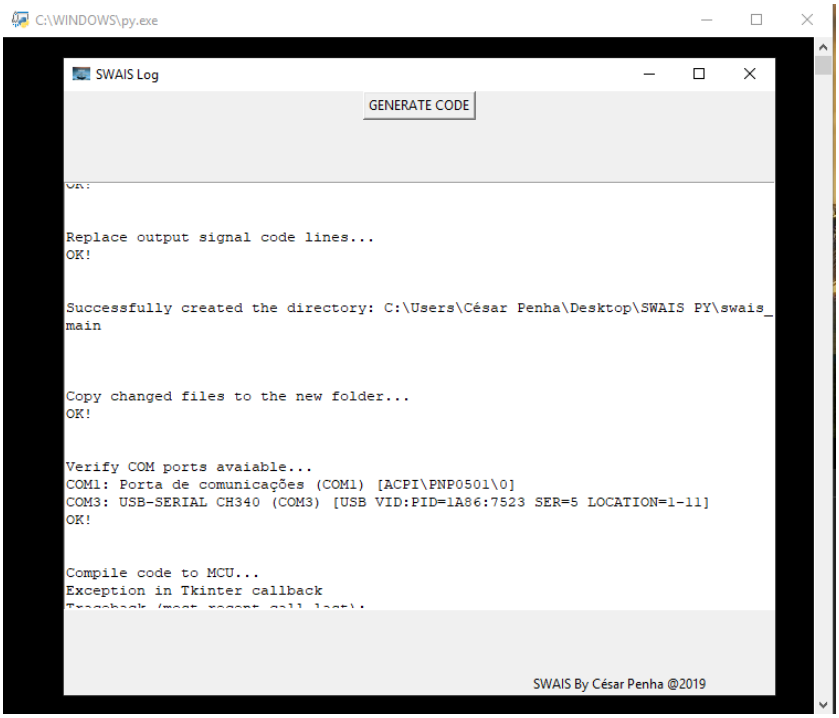

Figura 3.5: Aplicação swaisApp desenvolvida em linguagem Python.

sistir em vários dispositivos sensores e um dispositivo atuador. No caso da existência de diversos dispositivos sensores, o sistema permite a reconfiguração dos mesmos baseando a sua escolha na potência do sinal recebido.

Na Figura 3.6 pode ser observado o dispositivo atuador ao centro (AD) e os diversos dispositivos sensores (SD). Os dispositivos que se encontram conetados ao dispositivo atuador por um qualquer tipo de comunicação sem fios, estão representados pelas setas de côr verde, enquanto os que não se encontram ligados estão representados pelas setas laranjas. Dentro da área representada pelo círculo azul, o dispositivo atuador selecionará quais sensores necessários para alimentar o *feedback* para o qual o dispositivo atuador está programado. Dentro destes, quais os que apresentam o melhor RSSI que, em princípio, serão os mais próximos do dispositivo atuador. Existe no entanto outra especificidade em relação à verificação do sinal: Caso existam dois dispositivos sensores do mesmo tipo (humidade, temperatura, etc) dentro do alcance de um determinado dispositivo atuador, este irá "escolher"qual o que apresenta melhor potência de sinal. Isso mesmo está representado na Figura 3.6 no caso dos dispositivos sensores SD5 e SD2, e também nos dispositivos SD4 e SD1. Apesar do sistema poder funcionar autonomamente, o dispositivo atuador poderá permitir a ligação externa como por exemplo um *GateWay* ou a uma ligação WiFi, dependendo do protocolo de comunicação utilizado.

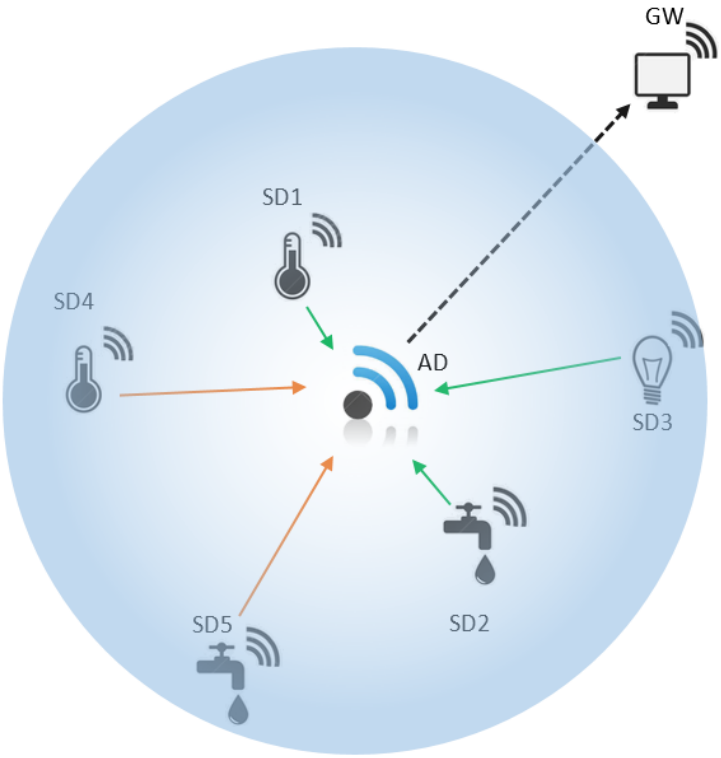

Figura 3.6: Exemplo de um nó do sistema. O círculo azul representa o alcance do dispositivo atuador (AD) e dentro dele os diversos dispositivos sensores (SD).

3.3 Protótipo Funcional

Outra das contribuições desta dissertação, consiste no desenvolvimento e implementação do protótipo funcional SWAIS *(Self-Configurable Wireless Automatic Irrigation System)*[7]. Além da sua funcionalidade, este protótipo pretende validar as contribuições descritas nas secções 3.1 e 3.2.

O protótipo SWAIS desenvolvido e apresentado nesta dissertação refere-se a um sistema de rega inteligente constituído por dispositivos sensores a dispositivos atuadores. Este sistema permite o controlo e monitorização da quantidade de água a irrigar, dependendo de uma diversidade de fatores. Esses fatores referem-se à sensorização ambiental e do solo proveniente de um conjunto diverso de dispositivos sensores que podem ser instalados nas proximidades do local a irrigar. Deste modo, pretende-se que o protótipo SWAIS possua as seguintes características / funcionalidades e que seguidamente serão explicadas individualmente:

- Desenvolvimento de um sistema redundante ao nível de sensores instalados, de modo que, em caso de falha de um dos sensores, o sistema permite o uso de outro sensor do mesmo tipo que esteja ao alcance;

- O sistema permite o uso de sensores externos, desde que o formato do *payload* transmitido respeite certos requisitos do próprio sistema;
- Desenvolvimento e implementação de todo o hardware e software necessários para que o consumo de energia seja reduzido ao mínimo, aumentando a vida útil das baterias;
- Uso de tecnologia *wireless* de longo alcance e baixa potência na comunicação entre dispositivos, nomeadamente LoRaWAN;
- O sistema ser autónomo ao nível do consumo de energia, utilizando baterias e painéis solares, aumentando sua portabilidade e a facilidade de adaptação às infra-estruturas existentes;
- A configuração simplificada por parte do utilizador, das regras pretendidas para a irrigação;[34]
- A utilização por parte do sistema das regras definidas pelo utilizador, recorrendo à geração automática do código C.[19]
- As regras que definem a irrigação serão desenvolvidas usando Redes de Petri, nomeadamente as IOPT Tools [34], usando o gerador automático de código C [19] em conjunto com um outro código adicional.
- O sistema contará com uma aplicação desenvolvida em linguagem Python que ficará encarregue de compilar todo o código e programar de forma automática o microcontrolador.

Para que seja possível os dispositivos comunicarem entre si, foi necessário definir um protocolo de comunicação a utilizar. Para este caso, e após uma ponderação sobre diversas hipóteses, foi escolhida a solução LoRa, nomeadamente LoRaWAN. Esta solução está inserida nas redes LPWAN e consiste num tipo de rede de comunicação sem fios, projetada para permitir comunicações de longo alcance com uma baixa taxa de transferência de bits entre dispositivos, tais como sensores alimentados por baterias. Uma LPWAN pode ser usada para criar uma rede privada de sensores sem fios, mas também pode ser um serviço ou infraestrutura oferecida por terceiros, permitindo que os proprietários de sensores os implementem em campo, sem investir em gateways. O SWAIS pode ser utilizado numa rede privada, sem recurso a gateways externos ao sistema, mas pode também ser integrado numa infraestrutura existente e mais ampla. Além dos motivos supracitados, o local de testes do protótipo (Lagoa, Algarve) está a ser equipado com uma infraestrutura de rede LoRa constituída por diversos gateways. Essa infraestrutura irá permitir a implementação de projetos piloto, onde o protótipo SWAIS irá ser inserido.

Outra especificidade do protótipo consiste na auto-configuração do sistema. Esta auto-configuração seleciona os dispositivos sensores com o melhor sinal dentro do seu raio de alcance. A potencia do sinal recebido no caso do protocolo LoRa é baseado no RSSI [35] de cada comunicação. Esse valor recebido é analisado pelo sistema que descarta os sinais mais fracos, caso existam dois ou mais sensores do mesmo tipo dentro do raio de alcance. Além dos dispositivos pertencentes ao sistema, o SWAIS permite o uso de sensores externos tais como, estações meteorológicas, sensores de rua ou outros, desde que o formato do *payload* transmitido respeite certos requisitos do próprio sistema; Isto permite um sistema redundante ao nível de sensores instalados, de modo que, em caso de falha de um dos sensores, o sistema permita o uso de outro sensor do mesmo tipo que esteja ao alcance;

Além do controlo da irrigação de forma local, mediante um botão que permite forçar o inicio ou a paragem da irrigação, o sistema foi desenvolvido para funcionar de uma forma totalmente autónoma, ou seja, ativando ou desativando a irrigação mediante o *feedback* recebido pelos sensores. No entanto, é necessário disponibilizar as regras que definem o inicio ou a paragem da irrigação. Para tal foram utilizadas as RdP, nomeadamente a *framework* IOPT Tools, que são uma classe das RdP não autónomas pois permitem a modelação das entradas e saídas no modelo, além de gerar código C de forma automática. Desta forma, o utilizador poderá modelar as regras de irrigação num ambiente gráfico e *userfriendly*, tendo a possibilidade de testar o modelo, antes de programar a plataforma que serve de base ao protótipo.

De modo a facilitar a integração do código respeitante às características do hardware utilizado, das características do protocolo de comunicação e do código gerado pela *framework* IOPT Tools, utilizou-se a aplicação swaisApp desenvolvida para o efeito. Esta aplicação permite que o utilizador consiga selecionar o hardware que irá ser utilizado, bem como a programação direta do MCU.

Capítulo 4

Exemplo de Validação

Este capítulo apresenta um exemplo de um sistema de rega desenvolvido com base na arquitetura proposta. O exemplo apresentado pretende validar a ferramenta e a arquitetura desenvolvida. O protótipo foi desenvolvido de raiz a nível de hardware e software, tendo em conta a arquitetura propostas. Neste capítulo primeiramente é utilizada swaisApp responsável pela integração dos códigos referentes ás características do hardware e do protocolo de comunicação. Seguidamente é explicada a arquitetura do protótipo e a forma como é realizada a escolha dos sensores. É apresentado um modelo desenvolvido com recurso à *framework* IOPT Tools, o qual foi utilizado no desenvolvimento e na implementação do protótipo SWAIS. É também realizada uma revisão de todo o hardware utilizado e construído especificamente para o protótipo incluindo os circuitos eletrónicos dos mesmos. Todo o código referente ao protótipo e à ferramenta encontra-se no repositório GitHub, disponível em https://github.com/Cmdslp/SWAIS.git.

4.1 Ferramenta de Automatização de Projeto

A acrescentar a essa funcionalidade, a ferramenta pode prever a seleção por parte do utilizador do tipo de plataforma a ser usada para programar, bem como o tipo de protocolo de comunicação a ser usado. Como o modelo pode variar consoante as necessidades do sistema, o código gerado de forma automática também varia.

Pelos motivos acima expostos, urge a necessidade de validação da ferramenta de automatização de projeto.

Essa validação foi efetuada utilizando diversos modelos, Dois desses são apresentados neste trabalho: O primeiro modelo ilustrado na Figura 3.3 onde se testou a execução de passos da rede e os sinais de entrada e saída do modelo.

Após esses testes, foi utilizada a ferramenta de automatização de projeto, de forma a automatizar o processo de integração dos códigos referentes às características do hardware e do protocolo de comunicação, com o código gerado pela ferramenta IOPT Tools do modelo apresentado.

Por fim, foi implementado o hardware necessário ao funcionamento do protótipo, utilizando uma *breadboard*, de modo a que a ferramenta pudesse programar o MCU que, neste caso, foi um Arduino na sua versão NANO. Foram desenvolvidas diversas funções de forma a que os sensores referenciados nesta dissertação, pudessem ser validados.

Após a verificação de que o código tinha sido bem compilado, observou-se os resultados tanto dos dispositivos sensores, como do dispositivo atuador, recorrendo à janela referente à porta de comunicação do IDE do Arduino.

Constatou-se que a ferramenta funcionava de forma correta, tanto na parte respeitante à junção dos diversos códigos, como na parte correspondente à programação do MCU.

Não obstante do acima exposto, desenvolveu-se um segundo modelo ilustrado na Figura 4.5. Todos os passos acima descritos, foram realizados uma segunda vez, de forma a validar a ferramenta com um segundo modelo.

Os resultados obtidos foram novamente os esperados, concluindo-se assim a validação da ferramenta de automatização de projeto.

4.2 Arquitetura do Protótipo

A arquitetura de um sistema inclui as principais propriedades físicas, estilo, estrutura, interações e finalidade de um dado sistema. O SWAIS, na sua forma mais compacta, consiste num ou vários dispositivos sensores e um dispositivo atuador. Um dispositivo sensor é constituído por um microcontrolador, um sensor e um módulo de comunicação.

Os sensores podem ser de diversos tipos tais como:

- Sensor Humidade do Solo;
- Sensor Temperatura do solo;
- Sensor Chuva;
- Sensor Temperatura do Ar;
- Sensor Humidade do Ar;
- Sensor Luminosidade;
- Sensor de Caudal.

O dispositivo atuador consiste num módulo de comunicação, um microcontrolador e um atuador (válvula de irrigação).

A arquitetura do protótipo é ilustrada da Figura 4.1. O sistema está a receber *feedback* contínuo do(s) dispositivo(s) sensor(es) espalhados pelo espaço verde. O dispositivo atuador recebe os dados do (s) dispositivo (s) de sensores e verifica a potência do sinal recebido de forma individual, selecionando o que possui melhor RSSI de entre os

dispositivos do mesmo tipo ao seu alcance. O utilizador desenvolve graficamente um modelo usando a *framework* IOPT Tools [36], onde as regras de irrigação são especificadas. O código gerado automaticamente pela ferramenta IOPT Tools[19] é combinado pela ferramenta de automatização de projeto "swaisApp", com o código desenvolvido manualmente, proveniente das características do hardware e do protocolo de comunicação a ser utilizado. A ferramenta de automatização de projeto swaisApp, programa a plataforma (microcontrolador). Por fim, o microcontrolador atua na válvula solenoide responsável pela irrigação.

Como alternativa, o sistema pode receber dados de outros sensores não pertencentes ao SWAIS, tais como sensores de rua ou estações meteorológicas. De forma a tornar isto possível, certos parâmetros da comunicação entre esses sensores e o SWAIS devem ser levados em conta. A *string* referente ao *payload* transmitido deve ter o mesmo formato para que possa ser reconhecida pelo módulo atuador. Além deste requisito, os sensores externos devem comunicar na mesma banda de frequência e para a mesma rede interna.

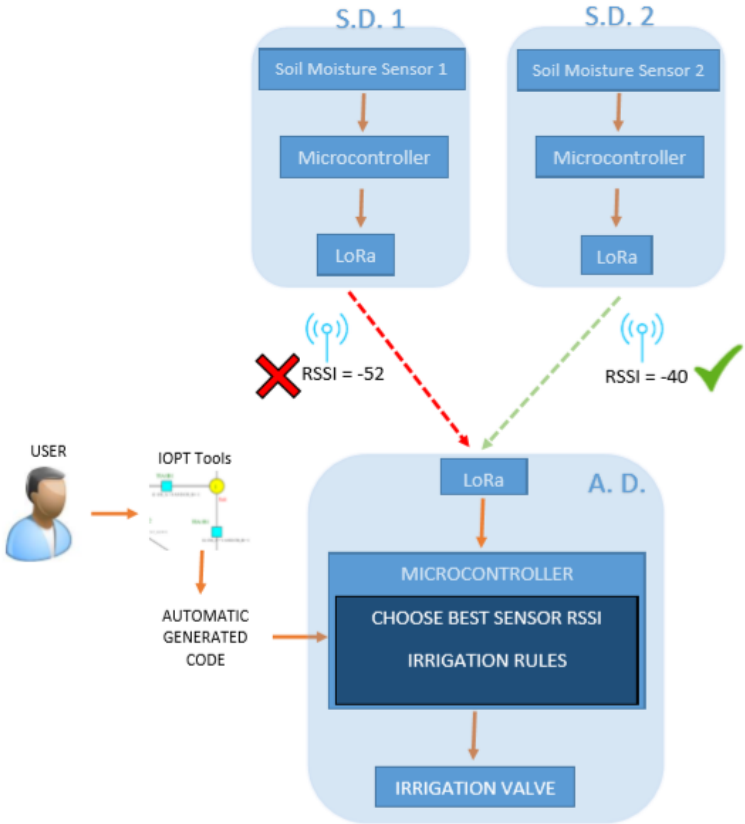

Figura 4.1: Arquitetura do protótipo SWAIS. Podemos observar dois dispositivos sensores (S.D.) do mesmo tipo (humidade do solo) e um dispositivo atuador (A.D.). O dispositivo sensor do mesmo tipo com o melhor RSSI é o escolhido.

No SWAIS, estes e outros parâmetros são definidos na comunicação LoRa dos módulos, e pode ser alterado conforme o necessário no código desenvolvido manualmente. No

entanto, existe a possibilidade de ligar os dispositivos sensores diretamente a um *gateway* LoRa. Desta forma, os dispositivos sensores comunicarão com o *gateway* e este com o dispositivo atuador, ativando ou desativando a irrigação. Com esta configuração, as preocupações com os parâmetros da comunicação anteriormente descritos deixarão de fazer sentido visto que, um concentrador LoRa recebe quaisquer *payloads* desde que o dispositivo se encontre na tabela de dispositivos autorizados do *gateway*.

4.3 Verificação da Potência do Sinal Recebido

Como já mencionado, o SWAIS consiste em vários sensores dispositivos e um dispositivo atuador. Na Figura 4.2 podemos observar um plano de implementação do SWAIS no parque Carlos Boto na cidade de Lagoa (Algarve), onde o dispositivo actuador está zona central do parque, contíguo a um ponto de água existente. Neste caso foi considerado um alcance na ordem de 250 metros, o que é muito abaixo do intervalo de referência anunciado pelo fabricante dos módulos de comunicação.

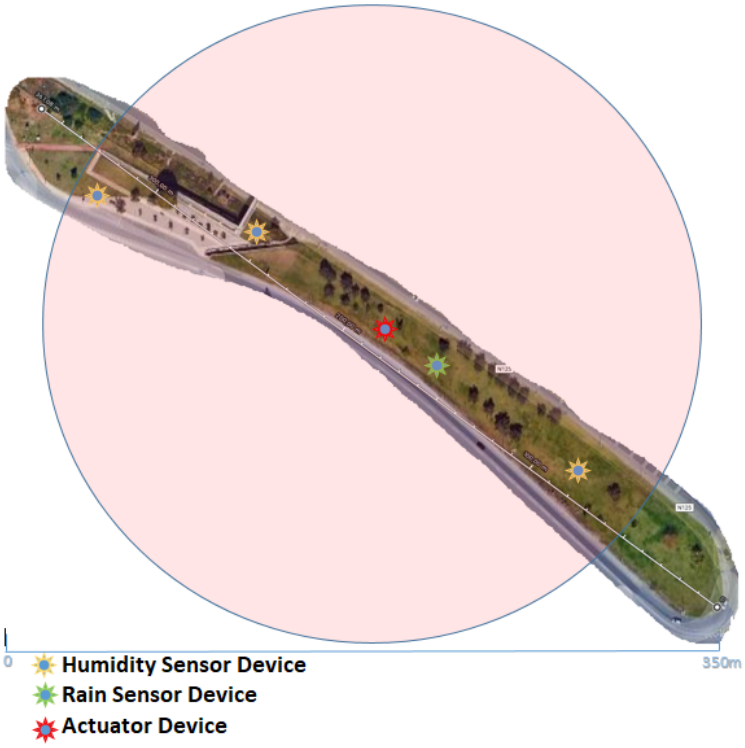

Figura 4.2: Exemplo de implementação do Protótipo SWAIS no parque Carlos Boto em Lagoa. O círculo vermelho representa o alcance do dispositivo atuador e dentro dele os diversos dispositivos sensores.

Dentro da área representada pelo círculo vermelho, o dispositivo atuador selecionará quais sensores necessários para alimentar o *feedback* para o qual o dispositivo atuador está programado. Dentro destes, quais os que apresentam o melhor RSSI que, em princípio,

estará mais próximo do dispositivo atuador. Podemos observar que, dentro do escopo, existem três dispositivos sensores equipados com um sensor de humidade, e um dispositivo sensor equipado com um sensor de chuva, mas na verdade apenas dois serão usados; um de humidade do solo e um de chuva. À primeira vista, parece desnecessário ter mais de um sensor do mesmo tipo no mesmo espaço, mas com um olhar mais atento, podemos constatar algumas vantagens:

- No caso de plantas ou culturas de diferentes tipos e consequentemente, com necessidades de água variadas, podem ser necessários usar sensores do mesmo tipo localizados nas diferentes zonas do parque onde essas culturas florescem, segmentando a irrigação área de acordo com as necessidades;

- Em caso de falha de um dos sensores, o dispositivo atuador pode usar outro sensor do mesmo tipo desde que esteja ao seu alcance; isto permite redundância no sistema pois, é melhor ter um sensor um pouco mais distante, ou até mesmo um que não pertence ao mesmo parque, do que não ter nenhum parâmetro de *feedback*.

Quando extrapolamos o SWAIS para uma cidade pequena, podemos observar na Figura 4.3 que a sobreposição do alcance dos módulos de comunicação é considerável. Como exemplo, vários parques foram selecionados na cidade de Lagoa, onde existe necessidade de irrigação (automática ou manual). Um alcance médio de 250 metros foi considerado para cada dispositivo. Segundo o fabricante do módulo LoRa usado neste protótipo, o mesmo possui um alcance de cerca de 15Km em LoS, dependendo na potência de emissão programada no próprio módulo. Com isto, é fácil ver a redundância de dispositivos e a cobertura da cidade pelos dispositivos sensores e atuadores.

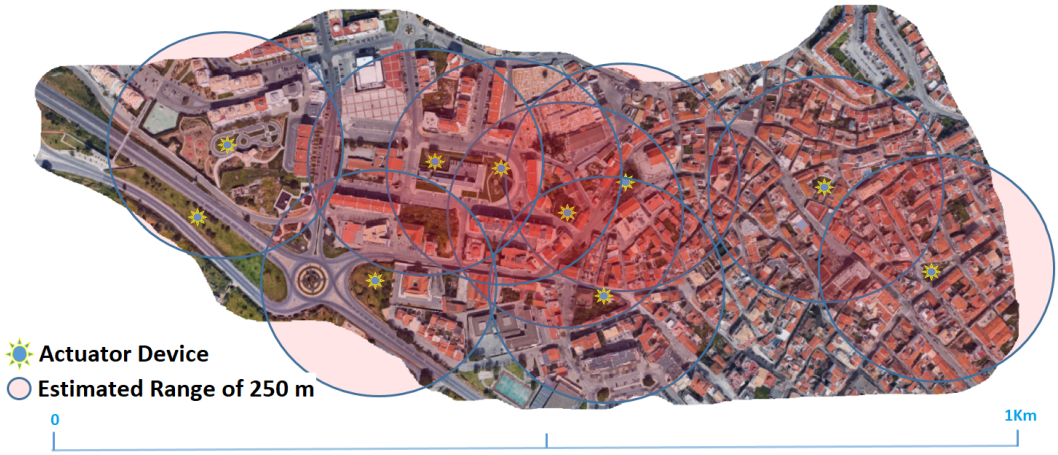

Figura 4.3: Exemplo de cobertura SWAIS da cidade de Lagoa (Algarve): O centro de cada círculo é um dispositivo de atuador.

4.3.1 Programação e Funcionamento

A programação e funcionamento da verificação automática do sinal, revelou-se um desafio devido principalmente às limitações do hardware em termos de memória e capacidade de processamento. Não se pode deixar de considerar que os microcontroladores usados neste protótipo apenas possuem 32 KB de memória flash, dos quais 2 KB são usados pelo *bootloader*. Se não for feita uma boa gestão dos recursos disponíveis, depressa os mesmos são esgotados. O primeiro passo é a recolha do valor do sensor pelo dispositivo sensor. Este valor juntamente com o tipo de dispositivo é colocando dentro de um objeto JSON. JSON é a abreviação de *JavaScript Object Notation* e é um formato de intercâmbio de dados, que possui algumas características vantajosas na transmissão de dados: É um formato independente da linguagem de programação usada, leve e baseado em texto. Em suma, dá-nos uma coleção de dados legível por humanos na forma de uma *string*, que podemos aceder de uma forma lógica. Pelas razões supracitadas, e pelo fato de existir uma biblioteca JSON disponível para a programação do microcontrolador utilizado no protótipo, foi escolhido este formato para a transmissão do *payload* entre dispositivos sensores e o dispositivo atuador.

O microcontrolador envia o *payload* para a porta *serial* do módulo de comunicação LoRa através do recurso à função "LoraSerial.println(stringToSend)". O *payload* a enviar ilustrado na listagem 4.1, não é mais do que uma variável (stringToSend) do tipo *string*, constituída pelas seguintes partes onde:

- O cabeçalho da mensagem a enviar (AT+SEND=);
- O endereço do dispositivo que envia a mensagem (0);
- O tamanho da mensagem em número de bits (31);
- A string com o tipo de sensor (RS) e valor do mesmo (59) contidos no objeto JSON.

Listagem 4.1: String correspondente ao *payload* a enviar pelo dispositivo sensor.

```
1   AT+SEND=0,31,{"s_type":"RS","s_value":"59"}
```

Após a receção do *payload* por parte do dispositivo atuador, conforme o apresentado na listagem 4.2, o módulo de comunicação LoRa disponibiliza o *payload* recebido os seguintes parâmetros:

- O cabeçalho da mensagem recebida (+RCV=);
- O endereço do dispositivo que enviou a mensagem (2);
- O tamanho da mensagem em número de bits (28);
- A string com o tipo de sensor e o seu valor que, neste caso, é um objecto JSON;

- A potência do sinal (-54);
- A relação Sinal Ruido (56).

Listagem 4.2: String correspondente *payload* recebido pelo dispositivo atuador.

```
1   +RCV=2,28,{"s_type":"RS","s_value":"59"},-54,56
```

Todos os valores recebidos além do objeto JSON são acrescentados ao *payload* pelo módulo de comunicação LoRa. Assim, quando a *string* com o *payload* é recebida, a mesma tem que ser decomposta em várias partes para poderem ser guardados os diferentes valores. Essa decomposição é realizada por diversas funções. Além do objeto JSON que é recebido, existem funções que retiram o endereço do dispositivo que enviou o *payload*, o tamanho, o valor da potência do sinal (RSSI) e da relação sinal ruido (SNR). O primeiro passo após a receção da mensagem pelo dispositivo atuador, é verificar se o *array* que contem os sensores com melhor recepção não se encontra cheio. Caso se encontre cheio, o programa ignora a verificação do sinal e segue para o próximo passo que retira os valores do *payload*.

Caso o *array* não se encontre cheio o programa entra na função "deserializeIncomingString". Nesta função, o programa verifica no *array* "collection"se o endereço do dispositivo que enviou a mensagem já existe. Se existir sai da função, se não existir copia o novo dispositivo sensor para o *array* juntamente com os seus dados. Após copiar o novo dispositivo sensor para o *array*, a variável "sensorCount"é incrementada. Este segundo passo repete-se a cada mensagem recebida, desde que o *array* não esteja cheio. Na listagem 4.3 em código C++, podemos observar a função supracitada.

Listagem 4.3: Função que verifica a existência do endereço da mensagem recebida no array "collection". Se não existir adiciona o novo endereço.

```
1  void deserializeIncomingString(String finalStrings) {
2
3    StaticJsonBuffer<200> jsonBuffer;
4
5    //Get the root object in the jsonBuffer
6    JsonObject& root = jsonBuffer.parseObject(finalStrings);
7
8    //Verify possibly deserilazation errors
9    if (root == JsonObject::invalid()) {
10     return;
11   }
12
13   //Verify each unique sensor address from the incomingString (JSON Object) and compares if
         exists in the collection
14   // If incoming sensor address exists in the collection, ignores and exit the function
15   for (int j = 0; j < sensorCount; j++) {
```

```
16      if (collection[j].s_address == incomingSensorAdress(incomingString))
17        return;
18    }
19
20    // If not, adds the new sensor to the collection
21    collection[sensorCount].s_address = incomingSensorAdress(incomingString);
22    collection[sensorCount].s_type = (char*)root["s_type"];
23    collection[sensorCount].s_value = root["s_value"];
24    collection[sensorCount].s_rssi = incomingRSSIsignal(incomingString);
25
26    printStruct();
27    // Increment the number of sensors saved in the collection
28    sensorCount++;
29    jsonBuffer.clear();
30  }
```

Seguidamente o programa verifica para cada tipo de sensor qual o que possui melhor receção. Essa verificação é realizada pelas funções "betterRSsensors", "betterSHsensors", "betterATsensors". Cada função respeita a um tipo individual de sensor: RS significa *Rain Sensor*, SH signifiva *Soil Humidity* e AT significa *Ambient Temperature*. A estas funções poderão ser adicionadas muitas outras, conforme os tipos de sensores instalados, no entanto neste caso foram considerados estes três tipos de sensores como demonstração. Na listagem 4.4 podemos observar um exemplo da função que verifica a potencia do sinal de um dispositivo sensor.

Listagem 4.4: Função que verifica qual o dispositivo sensor com melhor receção. Neste caso particular, um sensor de chuva.

```
1  //Check Rain Sensor
2  void betterRSsensors(){
3
4    for (int j = 0; j < sensorCount; j++) {
5
6      //Check if is RS
7      if (collection[j].s_type == "RS") {
8
9        //Check if current has better signal
10       if (collection[j]. s_rssi > betterRSSignalStregth) {
11         betterRSSignalId = j;
12         betterRSSignalStregth = collection[j]. s_rssi;
13         bestRSsensorAddress = collection[j].s_address;
14
15       }
16     }
17   }
18   Serial.println("Best Rain Sensor Sensor located on index " + (String)betterRSSignalId + " as a
       RSSI of " + (String)betterRSSignalStregth);
```

```
19
20 }
```

Por último, e após ser guardado o endereço do dispositivo sensor com melhor receção na variável "bestRSsensorAddress", o programa entra na função "parseCurrentValues". Esta função tem como finalidade comparar o endereço do dispositivo sensor da mensagem que chegou em último lugar com o endereço do melhor dispositivo sensor guardado na variável "bestRSsensorAddress". Caso esta comparação seja verdadeira, ou seja, o endereço seja o mesmo, os dados relativos à mensagem recebida (tipo de sensor, valor do sensor) são guardados em variáveis globais que serão usadas pelo código gerado automáticamente pela *framework* IOPT Tools. Esses dados servirão de *input* na rede de Petri, nomeadamente o tipo de sensor e o seu valor. Nesta função, tal como na listagem 4.4, para cada tipo de sensor terá que ser adicionado um ciclo *"if"* de forma a realizar a verificação. Na listagem 4.5 podemos observar a função acima descrita a qual dispõe de três tipos de sensores diferentes (RS, SH, e AT).

Listagem 4.5: Função que verifica se o endereço do dispositivo sensor que enviou a última mensagem é igual ao endereço do melhor dispositivo sensor guardado.

```
1
2  void parseCurrentValues(String finalStrings) {
3    StaticJsonBuffer<200> jsonBuffer;
4
5    //Get the root object in the jsonBuffer
6    JsonObject& root = jsonBuffer.parseObject(finalStrings);
7    Serial.println(finalStrings);
8
9    //Verify possibly deserilazation errors
10   if (root == JsonObject::invalid()) {
11     return;
12   }
13
14   // Verify if the address of the incomming sensor its the same
15   // of to the best sensor saved on the collection. If yes, saves
16   // IOPT variables and print the type and the value of that sensor
17
18   if (root["s_add"] == bestRSsensorAddress) {
19     inputIOPTrainSensorValue = root["s_value"];
20     inputIOPTrainSensorType = root["s_type"];
21     Serial.print(inputIOPTrainSensorType);
22     Serial.print(" IOPT Value: ");
23     Serial.println(inputIOPTrainSensorValue);
24   }
25
26   if (root["s_add"] == bestSHsensorAddress) {
27     inputIOPTSoilHumiditySensorValue = root["s_value"];
```

```
28      inputIOPTSoilHumiditySensorType = root["s_type"];
29      Serial.print(inputIOPTSoilHumiditySensorType);
30      Serial.print(" IOPT Value: ");
31      Serial.println(inputIOPTSoilHumiditySensorValue);
32    }
33
34    if (root["s_add"] == bestATsensorAddress) {
35      inputIOPTambientTemperatureSensorValue = root["s_value"];
36      inputIOPTambientTemperatureSensorType = root["s_type"];
37      Serial.print(inputIOPTambientTemperatureSensorType);
38      Serial.print(" IOPT Value: ");
39      Serial.println(inputIOPTSoilHumiditySensorValue);
40    }
41    jsonBuffer.clear();
42  }
```

Na figura 4.4 podemos observar um fluxograma que representa as operações necessárias para que se possa obter o endereço do dispositivo sensor com melhor valor de RSSI.

Além dos dispositivos sensores pertencentes ao SWAIS, o protótipo poderá usar outros sensores disponíveis ao alcance do dispositivo atuador tais como, estações meteorológicas, sensores de rua ou outros. Para que o dispositivo atuador receba esses *payloads*, os mesmos deverão conter obrigatoriamente o cabeçalho constante na listagem 4.2, de forma a ser reconhecida por este módulo de comunicação específico LoRa. No entanto, o sistema poderá ser adaptado para funcionar com um *gateway*, e nesse caso, poderão existir duas formas diferentes de implementação:

1. A primeira forma seria utilizar um concentrador externo ao sistema - poderá por exemplo ser um concentrador pertencente a uma infraestrutura de um município. Assim, todos os dispositivos sensores comunicariam com este concentrador, em vez de comunicarem diretamente com o dispositivo atuador. Por sua vez, esse concentrador iria transmitir para o dispositivo atuador os dados recebidos pelos diversos dispositivos sensores, e este por sua vez iria ativar ou desativar a rega.

2. A segunda forma seria implementar em cada dispositivo atuador um pequeno concentrador. Este concentrador ficaria responsável pela recepção dos *payloads* referentes aos dispositivos sensores associados ao mesmo, e pelo envio para o dispositivo atuador dos *payloads* recebidos. Além dessa funcionalidade, esse pequeno concentrador ficaria responsável pela comunicação desses dados a um concentrador de maior que conseguisse acoplar mais dispositivos.

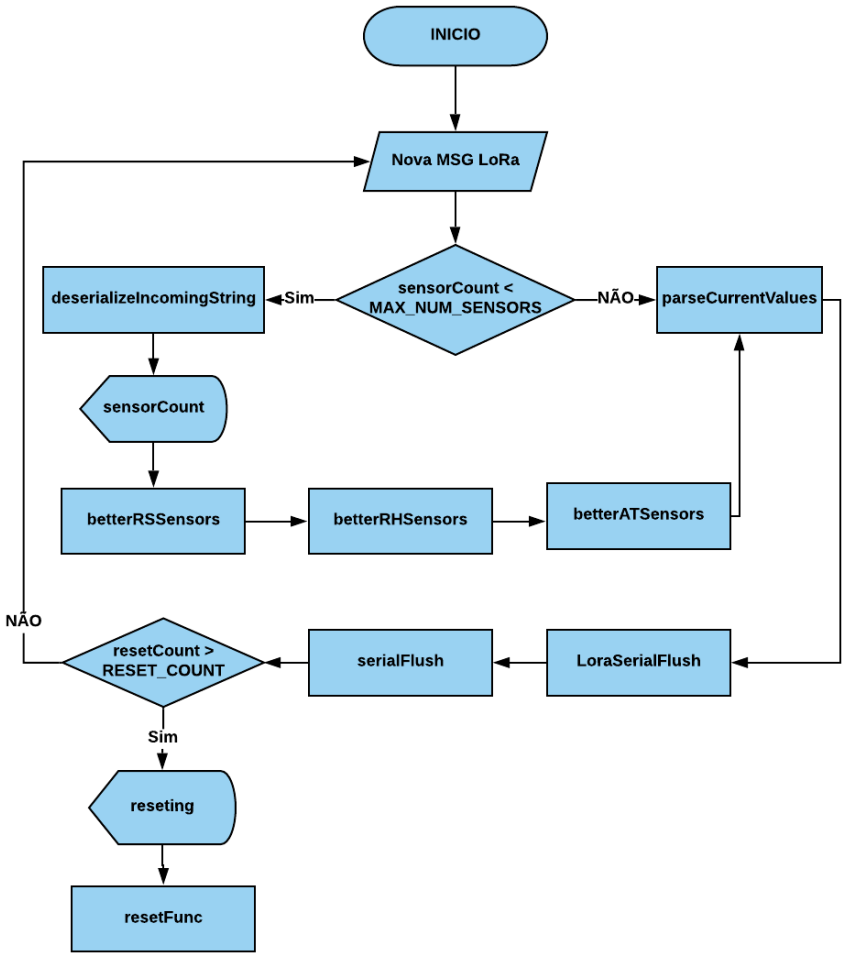

Figura 4.4: Fluxograma representativo das operações a realizar de forma a obter o melhor RSSI. (ISO 5807-1985 (E)).

4.4 Modelação e Verificação das Regras de Irrigação

Com o intuito de facilitar o desenvolvimento da parte do projeto referente aos procedimentos que enquadram a ativação ou desativação da irrigação, utilizou-se uma classe de redes de Petri baseada nas redes de Petri Lugar / Transição - As IOPT Tools.

Nas Figuras 4.5 e 3.3 são apresentados o design de dois modelos exemplificativos. No primeiro caso foram usados dois sensores, mas poderão ser quaisquer outros da lista de sinais de entrada. A verificação das regras definidas começa com a entrada dos valores recolhidos pelos sensores na rede.

Esses valores estão guardados nas variáveis globais do código desenvolvido referente às características do hardware e são as constantes nas linhas 19, 27 e 35 da listagem 4.5. Estes *inputs* poderão ser usados conjuntamente ou separadamente, consoante sejam

utilizados sensores de um ou mais tipos. Para que sejam utilizados outros tipos de sensores, basta replicar as funções constantes na listagem 4.4 e 4.5, alterando o campo *"s_type"* por outro à escolha que identifique de forma inequívoca o novo tipo de sensor.

Seguidamente, o programa verifica se o sensor de chuva apresenta um valor maior ou menor do que 30% de probabilidade de chuva (intervalo 0% a 100%). Se um valor inferior a 30% for confirmado, o programa realizará a próxima verificação, que neste caso será um sensor de humidade do solo. Se o valor do sensor de humidade do solo for menor que 30% (faixa de 0% a 100%), o sistema começará a regar, fechando assim o *loop* do programa. Caso o valor do sensor de chuva seja superior a 30%, o sistema emitirá um alerta e irá parar a rega caso esta esteja ativa, não realizando a segunda verificação (sensor de humidade do solo) fechando assim o programa.

O mesmo acontece se o primeiro sensor (sensor de chuva) passar na verificação, isto é, o valor apresentado pelo sensor é inferior a 30%: se o valor do sensor de humidade do solo for inferior a 30%, a irrigação será ativada, caso contrário, a mesma será desativada. Entende-se assim que, neste caso, o sistema possui uma verificação sequencial caso exista mais do que um tipo de sensor. É importante ter em atenção aquando do desenho do modelo que define as regras de irrigação, que o sensor de maior importância seja o primeiro a ser verificado. No entanto, poderão ser adotados outros tipos de verificações não sequenciais, criando expressões lógicas para cada caso utilizando os valores dos sensores mais relevantes. Por fim, há também uma funcionalidade chamada *"Manual Override"* que permite ao utilizador iniciar ou terminar a irrigação, independentemente dos valores do(s) sensor(s). Esta funcionalidade é ativada por um botão localizado no dispositivo atuador. As verificações feitas pele rede da Figura 4.5, podem ser observadas no fluxograma da Figura 4.6.

É importante esclarecer que o modelo apresentado é apenas um exemplo de aplicabilidade desenvolvido para o protótipo de forma a validar o conceito. A rede disponibilizada ao utilizador é a constante da Figura 3.2, ou seja, a parte da rede correspondente a transição *"Start Irrigation"* e à transição *"Stop Irrigation"*. Estas duas transições têm associadas a elas eventos de saída que ativarão ou desativarão o sinal de saída *"OuS_Valve"*, responsável pela abertura ou fecho da válvula solenoide de irrigação. O utilizador poderá modelar a rede consoante as necessidades, alterando a parte correspondente às regras que determinam o inicio da irrigação e as regras que determinam a paragem da irrigação.

4.5 Microcontroladores

O ATmega 328 é basicamente um microcontrolador AVR possuindo vários recursos que o tornam atualmente o dispositivo mais popular no mercado. Esses recursos consistem em arquitetura RISC avançada, bom desempenho, baixo consumo de energia, RTC com oscilador separado, 6 pinos PWM, *serial* USART programável, bloqueio de programação para

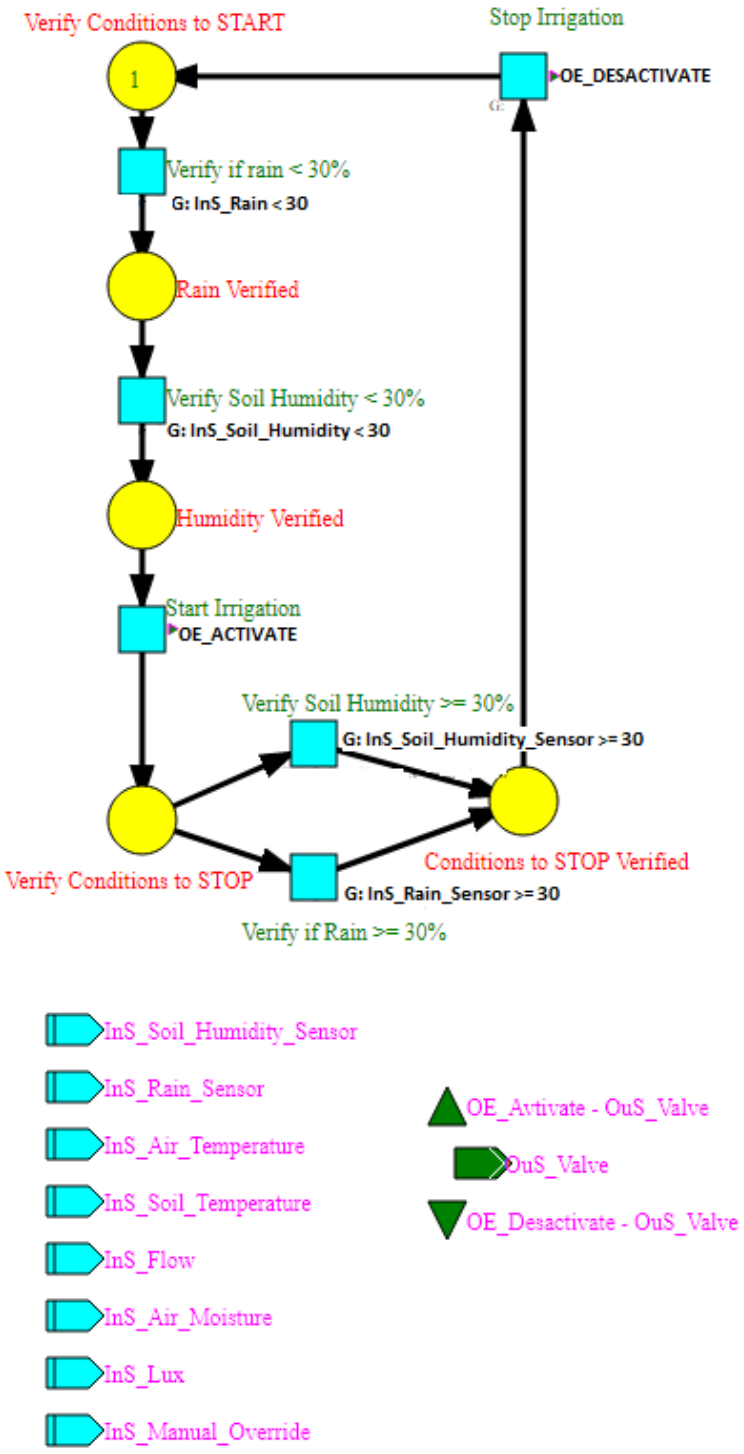

Figura 4.5: A rede de Petri, desenvolvida com a framework IOPT Tools e que representa as regras para a irrigação. Neste caso foram utilizados dois sensores (chuva e humidade do solo).

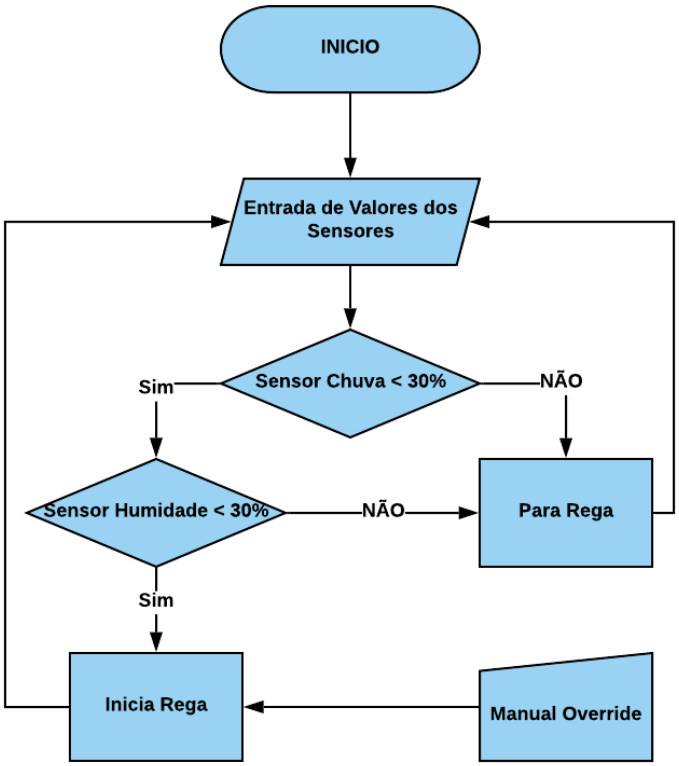

Figura 4.6: Fluxograma ilustrativo da verificação das condições de irrigação usando dois dispositivos sensores (ISO 5807-1985 (E)).

segurança de software e taxa de transferência de até 20 MIPS entre outros especificações. ATmega 328 é usado principalmente em plataformas de prototipagem nomeadamente a plataforma Arduino. Na figura 4.7 podemos observar o *pinout* do MCU.

O ATmega 328 é um microcontrolador AVR de 8 bits e 28 pinos fabricado pela empresa Microchip, possuindo uma memória flash programável de 32KB. Possui também uma memória EEPROM de 1KB e sua memória SRAM é de 2KB. Esta propriedade permite que, se a alimentação eléctrica fornecida ao microcontrolador for retirada, o integrado pode armazenar o código programado e voltar fornecer resultados após ser alimentado novamente. Este microcontrolador possui 8 Pinos para operações de ADC, e 3 temporizadores internos, dois deles são temporizadores de 8 bits, enquanto o terceiro é o temporizador de 16 bits.

O ATmega 328 opera com uma tensão entre 1.9 Volts a 5,5 Volts, mas normalmente é usado nas plataformas de prototipagem a uma tensão de 5 Volts. O seu consumo pode variar entre 15.15 mA no modo *"Active Mode"* e os 0.36 mA no modo *"Power Down"*, utilizando apenas o microcontrolador na suaconfiguração *"Bare Bone"*. As suas excelentes características fazem com que o mesmo seja bastante usado em sistemas embutidos e, por

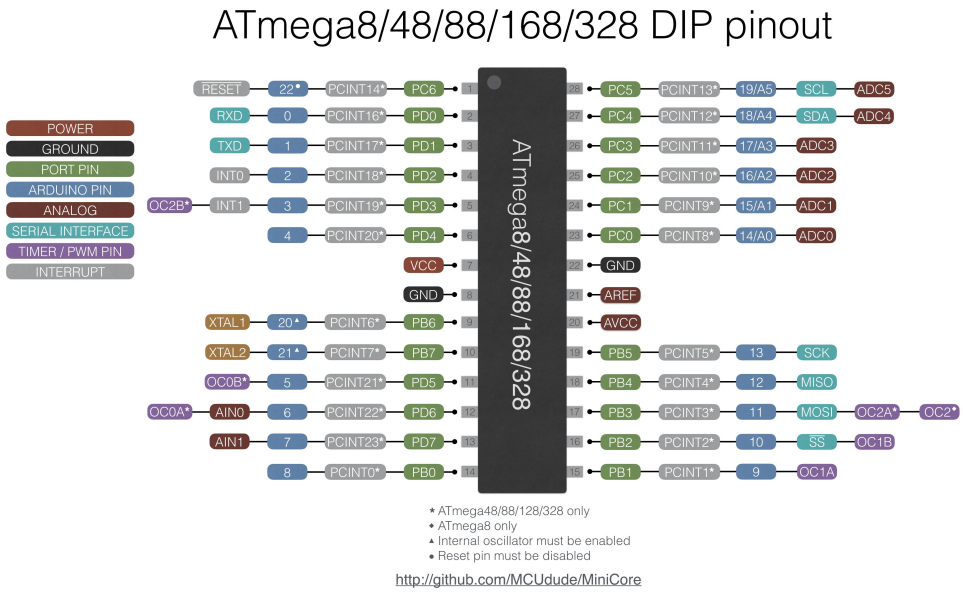

Figura 4.7: Microcontrolador ATMega 328P usado no dispositivo sensor. (*in* https://github.com/MCUdude/MiniCore)

fim, também a escolha para o desenvolvimento deste trabalho.

Os dispositivos sensores recorrem a este MCU na sua forma mais básica *"bare bone"*, ou seja, na sua forma elementar, apenas com os componentes necessários ao seu funcionamento, sem recorrer a plataforma de prototipagem Arduino, conforme pode ser observado na figura 4.8. Isto permite *downsizing* da PCB, bem como a redução de consumo pela não existência de componentes periféricos normalmente existentes nas plataformas de prototipagem.

Os dispositivos atuadores recorrem à plataforma Arduíno na sua versão NANO que utiliza o mesmo MCU dos dispositivos sensores. A figura 4.9 apresenta esta plataforma de prototipagem, onde são visíveis a porta mini USB usada para programar e alimentar o MCU, o botão de reset, e os 4 LEDs de notificação PWR, TX, RX, e o LED ligado ao pino nº 13. A razão da escolha de uma plataforma de prototipagem para os dispositivos atuadores em vez de um MCU na configuração *"bare bone"*, recai sobretudo por razões de simplicidade do projeto. O fator consumo nos dispositivos atuadores não é tão importante como nos dispositivos sensores, pois a produção de energia é bastante superior nestes últimos.

O foco do protótipo SWAIS é mostrar a potencialidade do projeto em si, e não tanto a escolha óptima de todos os componentes de hardware. Existirá sempre possibilidade de melhoramentos a nível da escolha de componentes do hardware, possibilitando a optimização do consumo e redução do custo do projeto.

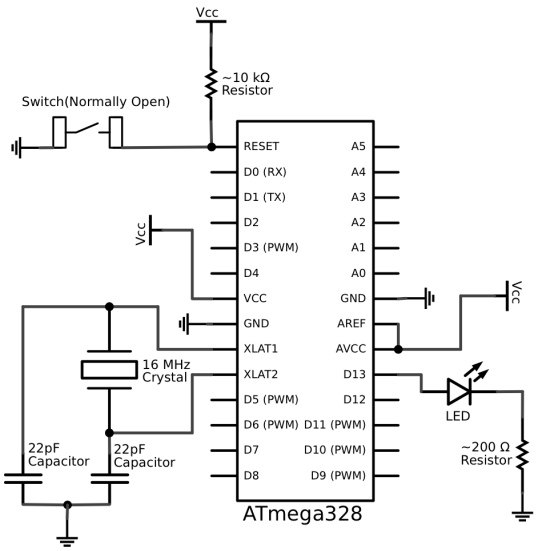

Figura 4.8: Microcontrolador ATmega328 na sua configuração *barebone*.

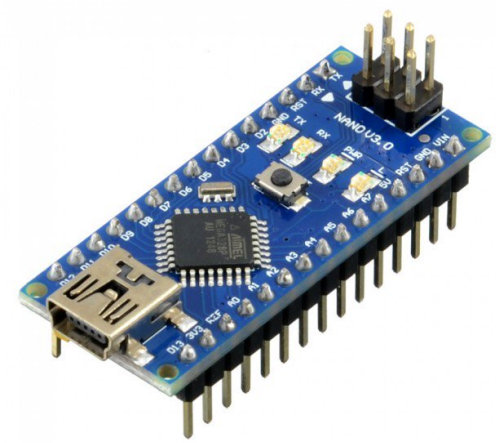

Figura 4.9: Microcontrolador Arduino Nano.

4.6 Sensores

Neste capítulo iremos fazer um *overview* sobre os sensores utilizados no protótipo SWAIS. A razão pela qual foram escolhidos estes sensores, foi devido ao fato de estes, talvez sejam os mais recorrentemente utilizados em sistemas de rega com *feedback*. No entanto as possibilidades são quase infinitas a nível de sensorização. Poderá ser utilizado qualquer sensor cujas características encaixem no perfil dos MCUs utilizados, nomeadamente a nível do sinal de *output* e alimentação energética do próprio sensor.

Tal como já referenciado anteriormente, o SWAIS permite a utilização de sensores ex-

ternos ao sistema: Poderão ser utilizados sensores de outros sistemas tais como estações meteorológicas, ou sensores isolados como por exemplo sensores de rua. Caso sejam utilizados sensores externos ao sistema conjuntamente com os mesmos módulos de comunicação LoRa utilizados neste projeto, terá que se respeitar o formato do *payload* de modo a que os módulos de comunicação do dispositivo atuador reconheçam a mensagem recebida. O modulo de comunicação utilizado neste protótipo admite o envio e recepção de/para vários módulos simultaneamente, mas não são *gateways*. Este requisito torna-se importante somente se for utilizado este modulo de comunicação LoRa; se no dispositivo atuador o módulo LoRa utilizado for substituído por um *gateway LoRa*, este requisito deixará de ser obrigatório, pois um *gateway* recebe *payloads* de diferentes formatos.

4.6.1 Sensor de Humidade do Solo

Um dos principais sensores que o protótipo possui é o sensor de humidade do solo exibido na Ffigura 4.10. Este sensor é responsável pela medição da quantidade de água presente no solo. A escolha deste sensor para o protótipo residiu no facto de o mesmo ser um sensor capacitativo e não resistivo. O principal problema com os sensores resistivos de humidade do solo é a corrosão das sondas do sensor, não apenas porque está em contacto com o solo, mas também porque há um fluxo de corrente contínua que causa a electrólise dos sensores.

A medição capacitiva tem algumas vantagens, não apenas evita a corrosão da sonda, mas também fornece uma leitura mais precisa do teor de humidade do solo, do que a medição da resistência.Na realidade não mede a humidade (como a água é um mau condutor de corrente), em vez disso, mede os iões que estão dissolvidos na humidade, ou seja, a adição de fertilizante, por exemplo, diminuirá a resistência do solo, mesmo que não seja adicionada água. A medição capacitiva mede basicamente o dieléctrico que é formado pelo solo e a água é o fator mais importante que forma esse dieléctrico. Este sensor opera com uma tensão entre 3.3 volts e 5.5 volts, visto que possui um regulador de tensão interno. O seu consumo é de cerca de 5mA o que o torna ideal para projetos em que se requer baixos consumos.

4.6.2 Sensor de Temperatura e Humidade do Ar

Outros parâmetros de elevada importância quando se trata de irrigação é, sem duvida, a temperatura ambiente e a humidade relativa do ar. Desta forma, foi selecionado o sensor de humidade e temperatura DHT22 (*Digital Humidity and Temperature*) exibido na figura 4.11. Este sensor é utilizado para medir a temperatura nas escalas de - 40 a + 80 graus Celsius e a humidade do ar nas faixas de 0 a 100%, contando com uma precisão que varia entre 2 a 5%. A tensão de trabalho situa-se no intervalo entre 3.3 volts e 6 volts e possui um consumo máximo de 2.5mA durante a conversão. A taxa de amostragem é de 0,5Hz (uma

Figura 4.10: Sensor de humidade do solo.

vez a cada 2 segundos). É muito utilizado no desenvolvimento de projetos eletrónicos e robóticos através de "boards"de desenvolvimento, entre elas, Arduino, Raspberry, ARM, AVR, PIC, etc. pois possui apenas 1 pino de *output* com saída digital. A principal diferença do sensor de humidade Temperatura DHT22 para o seu irmão DHT11 é o que o primeiro (DHT22) apresenta maior margem de trabalho, alcançando maiores temperaturas e maior nível de humidade, além de possuir maior precisão. Este sensor AM2302 é compatível com os modelos DHT22/AM2303 e é formado por um sensor de humidade capacitivo e um termístor para medir o ar ao redor, enviando um sinal digital para o microcontrolador. Foi escolhido este sensor devido à sua relação qualidade preço e à sua fiabilidade.

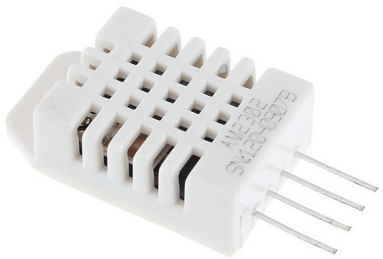

Figura 4.11: Sensor de humidade e temperatura do ar DHT22.

4.6.3 Sensor Chuva

Um dos fatores determinantes para a rega é se está a chover ou se choveu à relativamente pouco tempo. Assim, julgou-se importante adicionar um sensor de chuva ao protótipo de forma a possibilitar a medição deste parâmetro meteorológico. Foi selecionado um sensor de chuva igual ao da figura 4.12; este sensor é alimentado por uma tensão de 5 volts V ou 3.3 Volts no pino VCC. O pino "S"dará-nos um valor analógico entre os pinos VCC e GND. Iremos então usar o pino "S"como entrada analógica ligando o mesmo a um pino analógico do microcontrolador definido como "input". O valor lido será maior dependendo da superfície do sensor é coberto com água. Isso ocorre porque a água age como um condutor, diminuindo a resistência entre os pinos VCC e GNG. Este sensor consome um valor de cerca de 20mA a 5V e possui um sensibilidade entre 10 e 90%. Para detectar se está a chover, este sensor tem que ser posicionado horizontalmente para que as gotas de chuva caiam sobre o sensor. À medida que os pingos de chuva caem e se vão acumulando na superfície do sensor, a diferença de potencial no pino S irá incrementar e , desta forma podemos deduzir que está a chover.

Figura 4.12: Sensor de Chuva Analógico.

4.7 Atuador

O dispositivo atuador é responsável por receber os dados do (s) dispositivo (s) sensor (es), escolhendo os dispositivos idênticos com o melhor RSSI, verificando as regras de irrigação e determinando se é necessário irrigar abrindo ou fechando a válvula de irrigação. Para ativar ou desativar a rega, foi utilizada uma válvula solenoide semelhante à exibida na figura 4.13 com uma tensão de operação de 4.5 volts e com um diâmetro de 1/2". Esta

válvula possui uma resistência na bobine de 15 Ohm e uma pressão de trabalho de 0.02 a 1 MPa.

Figura 4.13: Válvula solenoide de irrigação.

4.8 Módulo de Comunicação LoRa

LoRa é a camada física ou a modulação sem fios utilizada para criar o *link* de comunicação de longo alcance. Muitos sistemas sem fios legados usam deslocamento de frequência de modulação (FSK) como a camada física, isto porque é muito eficiente na modulação de forma a alcançar um baixo consumo de energia. A tecnologia LoRa é baseada na alteração de frequência de modulação CSS, que mantém as mesmas características de baixa potência que a modulação FSK mas aumenta significativamente o alcance de comunicação. O *Chirp Spread Spectrum* tem sido usado há décadas nas comunicações a nível militar e espacial especialmente devido às longas distâncias de comunicação que podem ser alcançadas e à robustez às interferências, mas o LoRa é a primeira implementação de baixo custo para uso comercial.

A vantagem da tecnologia LoRa está na capacidade de se alcançarem comunicações de longo alcance. Um único *gateway* pode cobrir cidades inteiras ou dezenas de quilómetros quadrados em áreas rurais. No entanto o alcance dependerá sempre do meio ambiente ou das obstruções num determinado local tais como edificações e relevo do terreno. No entanto a tecnologia LoRa e LoRaWAN possui um *link budget* superior a qualquer outra tecnologia de comunicação padronizada. O *link budget* é normalmente dado em decibéis (dB), e é o principal fator na determinação do alcance num determinado ambiente.

Por todos os motivos acima descritos, e pelo fato de que o local escolhido para implementar o protótipo estar a ser equipado com uma rede LoRa, a escolha da rede LPWAN

a usar no SWAIS, recaiu sobre esta tecnologia.

Depois de uma pesquisa *online* acerca de módulos e chips de comunicação LoRa, a escolha foi de um módulo*transceiver* da marca Reyax, modelo RYLR896 [37].

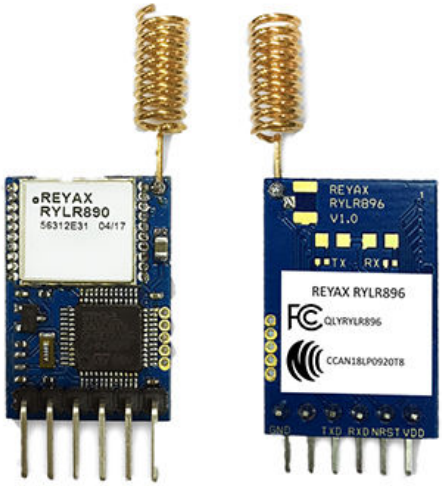

Figura 4.14: Modulo de comunicação LoRa.

Este *transceiver* possui diversas vantagens comparativamente a outros modulos entre as quais destacam-se:

1. Certificação NCC e FCC;
2. Encriptação de dados AES128;
3. Amplificador de potência de alta eficiência;
4. Programação através de comandos AT;
5. Antena integrada na PCB;
6. Alcance até 15km.

Além das vantagens supracitadas, este módulo possui algumas particularidades interessantes; Cada módulo pode ser programado com um endereço único (até ao máximo de 65536), ou seja, cada dispositivo sensor ou atuador poderá ser identificado numa qualquer rede; Outra particularidade deste módulo é que podem ser configuradas diversas redes distintas (até ao máximo de 17 redes). Isto significa que podemos ter vários módulos com IDs semelhantes a trabalhar em redes diferentes, sem que os mesmos entrem em conflito de comunicação; além das particularidades anteriores, a frequência de transmissão/recepção de cada módulo pode ser programada de forma individual. Todas estas

particularidades permitem combinações quase infinitas, que podem ser usadas sem necessidade de *gateways*. Em relação aos consumos, este modulo de comunicação consome cerca de 43mA em transmissão, quando a potência máxima do sinal é definida. Em modo de recepção, este módulo consome cerca de 16.5mA e cerca de 0.5uA em *sleep mode*.

4.9 Sistema de Alimentação

4.9.1 Fundamentação

A energia é um elemento essencial à vida das sociedades modernas e é uma das chaves para o desenvolvimento destas. Hoje em dia a poluição e o crescente consumo de recursos para a produção de energia elétrica são problemas que afetam toda a humanidade. Algumas consequências são, por exemplo, o efeito estufa e alterações climatéricas que estão a afetar todo o planeta. Diante desse cenário, a geração de energia elétrica através de fontes renováveis de energia apresenta-se como uma alternativa atraente, pois além de não poluírem o ambiente, ainda têm a vantagem de serem inesgotáveis. Observando o gráfico da figura 4.15, torna-se imprescindível encontrar alternativas em outros tipos de recursos energéticos. Existe a necessidade de se identificar soluções locais, viáveis sob o ponto de vista técnico e económico, que permitam a redução do atual consumo energético. Estas soluções podem passar, por exemplo, por incentivos à adoção de procedimentos e equipamentos que visem a eficiência energética, alternativas de geração de energia que diminuam a necessidade do consumo de combustíveis fósseis. Reforçar a importância da investigação e do desenvolvimento de novas tecnologias, além da transferência e adaptação de tecnologias já existentes à realidade. No que diz respeito a Portugal, o país não possui recursos fósseis suficientes para atender às suas necessidades de energia.

É neste contexto que o sistema de alimentação do SWAIS foi projetado. O sistema requerer-se autónomo em relação à rede pública de abastecimento elétrico. Esse motivo prende-se com o fato de o SWAIS poder ser instalado em qualquer lugar, mesmo afastado de uma qualquer rede elétrica. Isso permite não só uma independência elétrica, uma portabilidade superior, bem como uma redução dos custos de implementação e manutenção. Desta forma, não será necessário tomar em atenção esta limitação aquando da escolha dos lugares de implementação, devendo apenas tomar em consideração os princípios de instalação de um qualquer painel fotovoltaico.

4.9.2 Painéis Fotovoltaicos

Sabendo que Portugal é um dos países europeus com maior número de horas de sol anuais, foi considerada a utilização de painéis fotovoltaicos como principal fonte energética, aproveitando um recurso abundante e limpo. No mapa da figura 4.16, podemos observar a produção de energia em KWh/m2 em toda a Europa utilizando painéis foto voltaicos com uma inclinação ótima e orientação a Sul. É fácil constatar que Portugal é dos países

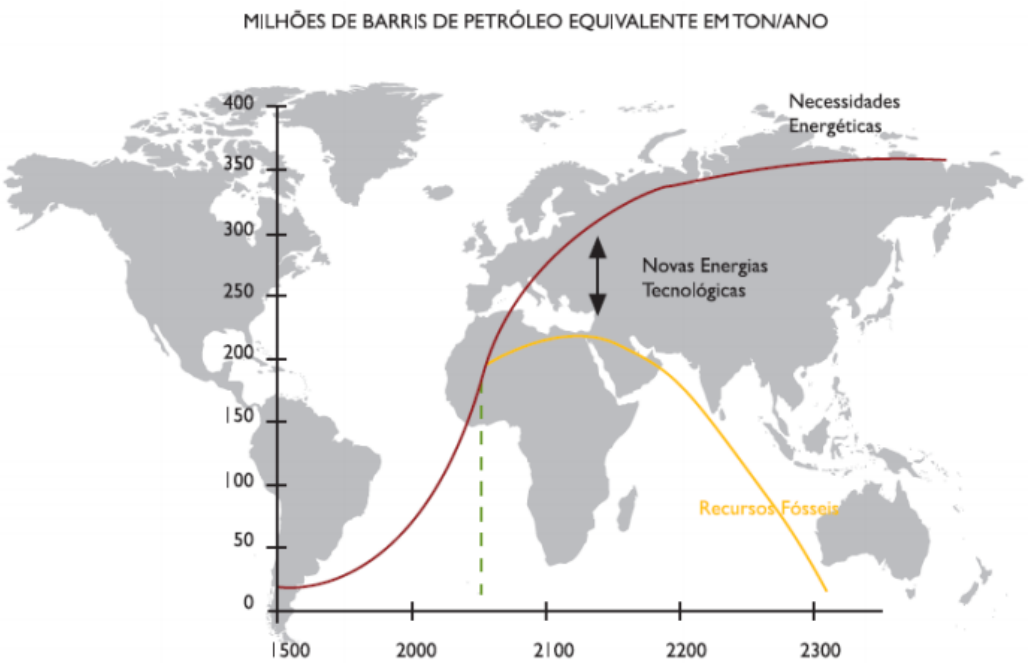

Figura 4.15: Necessidades energéticas *versus* recursos fósseis.(*in* ???)

europeus que mais beneficia desta fonte de energia, podendo obter produções superiores a 1900KWh/m2 no sul do país.

Desta forma, podemos concluir antemão que é deveras vantajoso aproveitar toda esta potencialidade fazendo uma correta gestão entre a produção de energia elétrica, o consumo e o carregamento das baterias, que irão servir de suporte ao sistema para quando não houver lugar à produção de energia elétrica, ou quando essa produção seja inferior às solicitações do sistema. Com vista a tornar o SWAIS independente, serão utilizados painéis solares tanto nos dispositivos sensores como nos dispositivos atuadores. Depois de efetuados os cálculos necessários, chegou-se a conclusão de que irão ser necessários dois tipos de painéis fotovoltaicos policristalinos:

- Dispositivos sensores: Um painel fotovoltaico de 5 volts com uma produção máxima de 60mA e uma potência de 0.3 Watts conforme o observado na figura 4.17.

- Dispositivos Atuadores: Um painel fotovoltaico de 6 volts com uma produção máxima de 866mA e uma potência de 5.2 watts, conforme o observado na figura 4.18.

Os paineis fotovoltaicos foram dimensionados tendo em conta os consumos dos dispositivos sensores e dispositivos atuadores, medidos com recurso a multímetro digital após a montagem de todo o hardware e efetuados diversos testes. No entanto, é importante ressalvar que o consumo médio pode variar consoante o número de transmissões entre

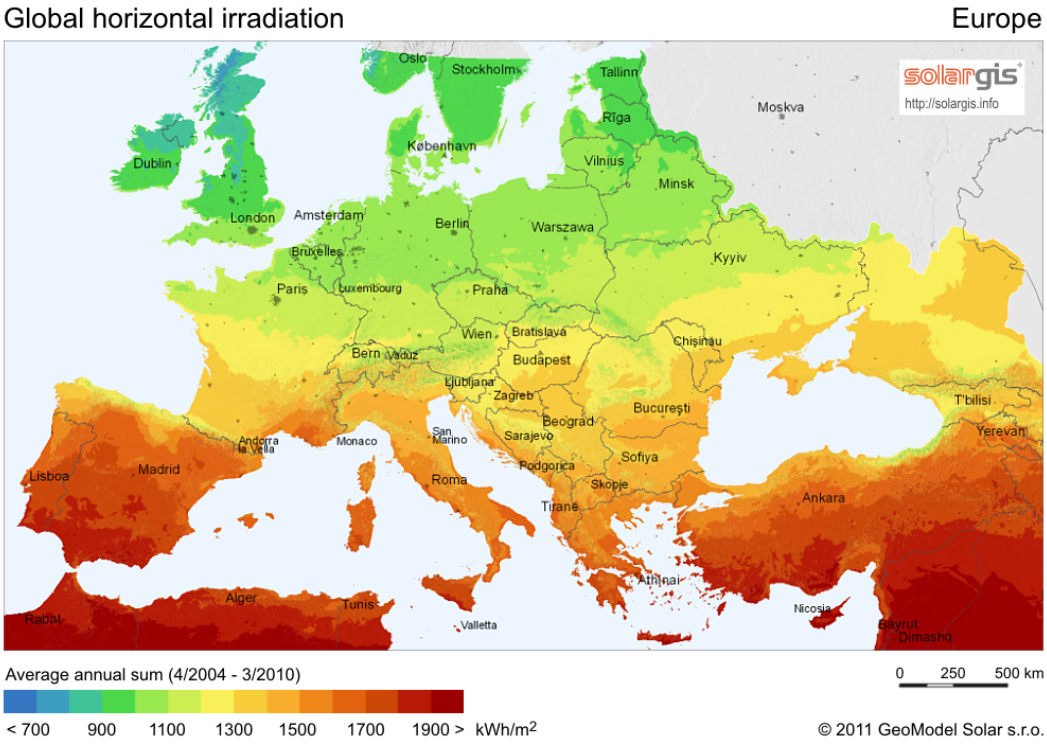

Figura 4.16: Potencial de produção fotovoltaica na Europa. (*in* https://en.wikipedia.org/wiki/Solar_energy_in_the_European_Union)

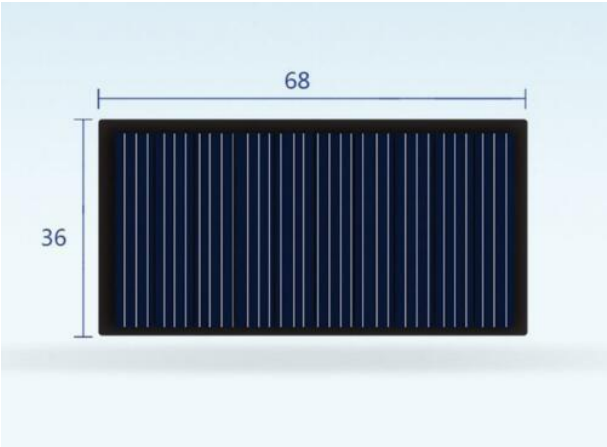

Figura 4.17: Painel fotovoltaico de 60mA usado nos dispositivos sensores.

dispositivos, nomeadamente nos dispositivos sensores. A maior parte do consumo de corrente é efetuado durante a transmissão de dados, quando o módulo de comunicação LoRa é ativado, tendo sido medidos picos de consumo na ordem dos 40mA durante a transmissão de dados, baixando para cerca de 4mA durante o período que se encontra em *sleep mode*.

Figura 4.18: Painel fotovoltaico de 866mA usado nos dispositivos atuadores.

Nos dispositivos atuadores, o maior consumidor de corrente é a válvula solenoide. Este consumo aumenta bastante consoante o tempo em que a válvula solenoide está ativada, ou seja, quando está a ser realizada irrigação. O consumo dos restantes componentes do dispositivo atuador nomeadamente o módulo de comunicação LoRa são em tudo semelhantes aos dispositivos sensores.

Para isso foi necessário instalar um circuito de carga das baterias em ambos os tipos dispositivos. De uma forma simplificada podemos esquematizar a alimentação do dispositivos observando o circuito da figura 4.19.

O problema seguinte consistiu exatamente no ponto anterior – gestão da produção da energia elétrica – pois requerer-se que o consumo esteja ligeiramente inferior à produção bem como o balanceamento entre este e o carregamento da bateria.

4.9.3 Carregador Solar do Dispositivo Atuador

Foi com este princípio que, depois de consultada diversa documentação sobre esta temática, que se decidiu implementar um carregador/gestor solar para baterias Li-Po conforme o apresentado na figura 4.20 [38]. Apesar de haver no mercado muitos carregadores de baterias, nem todos servem os propósitos do sistema, pois tal como existem baterias de vários tipos, também os carregadores têm várias características diferentes. A escolha final recaiu sobre um carregador específico da Adafruit para sistemas solares que faz a gestão automática entre o estado da bateria, a produção energética e o consumo do sistema.

Este carregador foi projetado especificamente para carregamento solar e automatica-

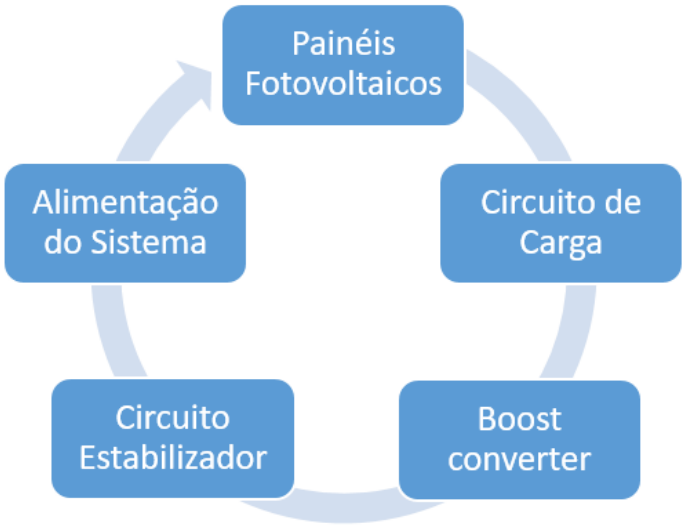

Figura 4.19: Circuito de alimentação dos dispositivos sensores e atuadores.

Figura 4.20: Carregador de baterias LiPo da marca Adafruit

mente retirará o máximo de corrente possível a partir do painel em quaisquer condições de luminosidade. Mesmo não sendo um verdadeiro MPPT, tem um desempenho quase idêntico sem o adicional de custo de um *buck-converter*. O carregador vem montado numa PCB com duas portas JST, uma para a bateria *(BATT)* e outra para a carga *(LOAD)*. Possui um grande condensador de estabilização, bem como um Jack de 2.1mm para ligação ao painel solar e uma porta mini USB que serve igualmente para carregar a bateria.Existem três LEDs de estado no carregador:

- O LED *"PWR"* vermelho indica que há uma boa fonte de alimentação conectada ao carregador. Se este LED não estiver aceso, algo está errado com a fonte de alimentação.

- O LED laranja *"CHRG"* indica o estado de carga atual. Quando este LED está aceso, o carregador está a carregar a bateria. Este LED também atua como um indicador de bateria fraca (fixado em 3.1V) quando o carregador não está ligado a qualquer fonte de energia (USB ou Painel Solar). Quando a tensão da bateria cai abaixo de 3.1V, o LED laranja acende-se.
- O LED verde *"DONE"* ilumina-se quando a bateria está carregada.

Se for necessário ligar LEDs de maiores dimensões ou ligar um microcontrolador a estes pinos de status, podemos usar as furações existentes na *board* na zona inferior da PCB, tal como podemos observar nas figura4.21 e 4.22.

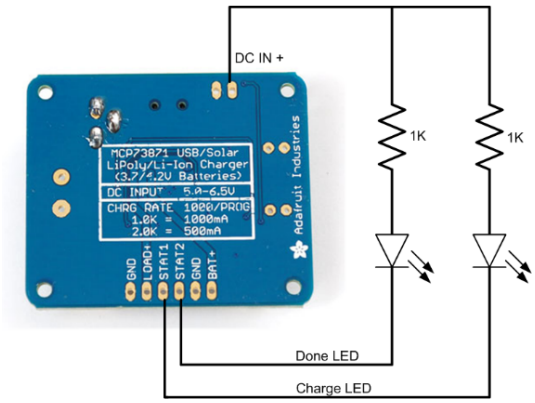

Figura 4.21: Carregador de baterias LiPo da marca Adafruit

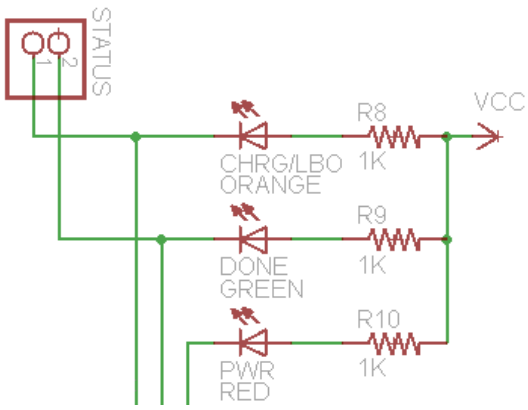

Figura 4.22: Carregador de baterias LiPo da marca Adafruit

O chip MCP73871 usado no carregador solar tem uma característica muito interessante chamada "compartilhamento de carga."Tomemos como exemplo a seguinte situação: Queremos usar a bateria enquanto ela está a carregar. Para isso teríamos que ter um carregador de baterias dito "normal"ligado à bateria e ligar uma carga diretamente aos pólos

da bateria. Isto significa, no entanto, que o carregador está a proceder ao carregamento da bateria, enquanto alimenta uma carga. Desta forma o carregador está a trabalhar num esforço extra e a bateria está a ser carregada e descarregada constantemente. Como característica, este carregador tem um transístor de passagem dentro do chip ligado à carga de saída da tensão de entrada de modo a não perder a eficiência de carga / descarga da bateria. Quando o carregador solar é alimentado a partir de uma porta USB ou por um painel fotovoltaico, a corrente de carga vai diretamente a partir da tensão de entrada para a porta de saída. Se a corrente exigida for maior do que o que a porta do painel fotovoltaico ou da porta USB podem fornecer, a corrente é complementada pela bateria Li-Po até um máximo de 1.8 Amp. Na fig 4.23 podemos observar o circuito de controlo do carregador solar.

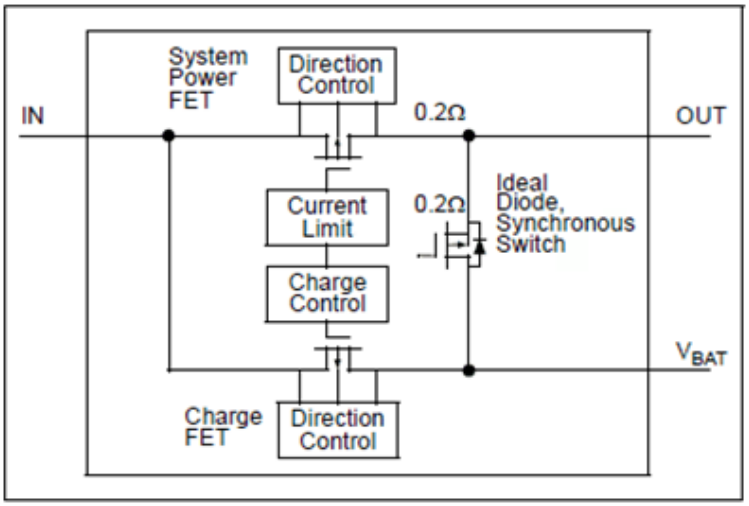

Figura 4.23: Diagrama de controlo do carregador solar Adafruit

O compartilhamento de carga inteligente significa que a tensão de saída de carga pode ser tão alta quanto 6 volts (máximo permitido de input) se os painéis estiverem expostos ao sol de forma direta, porque o carregador vai chamar a corrente diretamente do painel de 6V, em vez chamar a partir da bateria. Outra característica importante é o facto de este carregador poder monitorizar a temperatura da bateria. Para isso basta remover a resistência *"THERM"* do local onde se encontra soldada e soldar um termómetro NTC de 10k tal como pode ser observado na Fig 4.24. Desta forma se a temperatura da bateria subir acima de 50ºC ou baixar até temperaturas próximas do ponto de congelamento, o carregador para o processo de carga para evitar danificar a bateria.

Além das características acima mencionadas, este carregador solar vem com uma potência de carga predefinida de 500mA, o que se torna ideal para as portas USB, adaptadores USB de parede e painéis solares até 3 Watts. No entanto, a produção do painel fotovoltaico do dispositivo atuador é de cerca de 866 mA , ou seja, é superior ao máximo permitido pela resistência (RPROG) cujo seu valor é de 2000 Ω, levando a um valor

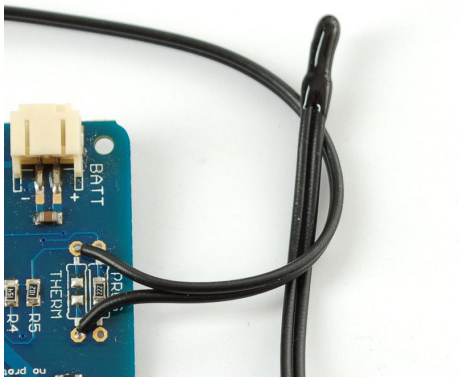

Figura 4.24: Sensor de temperatura acoplado no carregador de bateria

máximo de carregamento de 500 mA. Recorrendo à lei de Ohm, a corrente é definida por:

$$\frac{1000mA}{RPROG} = I, \tag{4.1}$$

onde RPROG é a resistência programável.

Assim, com uma resistência de 2000 Ω teríamos:

$$\frac{1000mA}{2000\,\Omega} = I = 500mA. \tag{4.2}$$

No caso do SWAIS, o painel fotovoltaico utilizado no dispositivo atuador possui uma potência superior (5.2 W), o que permite configurar a resistência delimitadora (PROG) para o valor máximo de 1000 Ω o que permite uma corrente máxima de 1000 mAh.

4.9.4 Conversor DC para DC

O SWAIS alimenta-se dos painéis solares ou da bateria em caso dos mesmos não estarem a produzir energia elétrica. Como já mencionado as baterias de Li-Po ou Li-Ion monocelulares possuem uma tensão que vai aproximadamente dos 4.2 volts quando carregada, até aos 3.1 volts quando descarregada. Na figura 4.25 podemos observar um gráfico que representa o ciclo de carga de uma bateria de Li-Ion.

Os painéis solares também não produzem uma tensão constante, pois a mesma varia consoante a luminosidade que incide nos mesmos. Em dias nublados podemos ter uma tensão de alimentação inferior às necessidades do sistema requerendo um dispositivo que nivele a tensão de alimentação do projeto. É com base nesta necessidade que foi instalado um conversor de impulso. Um conversor de impulso ou também chamado de *step-up converter* ou *step-down converter* é um conversor de energia DC para DC com uma tensão (Volt) de saída maior ou menor do que a sua tensão de entrada. É um tipo de fonte de alimentação de modo comutada (SMPS), contendo pelo menos dois semicondutores (um

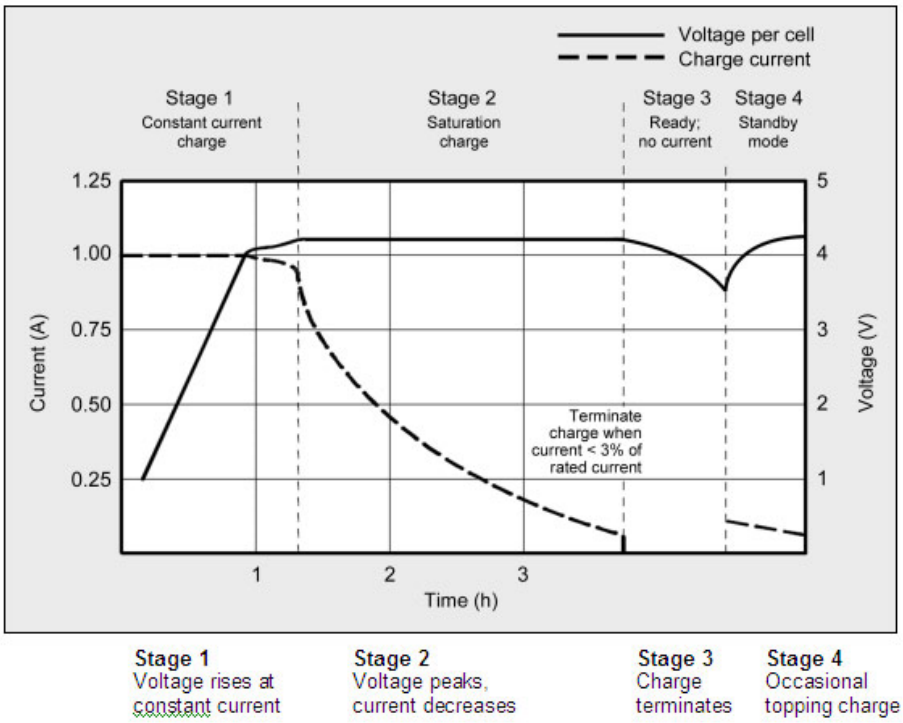

Figura 4.25: Ciclo de carga de uma bateria Li-Ion.

díodo e um transístor) e pelo menos um elemento de armazenamento de energia (um condensador indutor, ou os dois em combinação). Filtros feitos de condensadores (por vezes em combinação com indutores) são normalmente adicionados à saída do conversor para reduzir a saída de tensão de ondulação. A alimentação para o conversor pode vir de qualquer fonte de DC adequada tais como baterias, painéis solares, retificadores e geradores de corrente contínua. Um processo que altera uma tensão DC para uma tensão DC diferente é chamado de conversor DC to DC. Um conversor de impulso é um conversor DC para DC com uma tensão de saída maior do que a tensão da fonte. Um conversor de impulso é por vezes chamado um conversor *step-up*, uma vez que eleva em "degraus "a tensão da fonte. Dado que a energia (Potência = Tensão x Intensidade) deve ser conservada, a corrente de saída é menor do que a fonte de corrente. O contrário também é válido, ou seja, um conversor *step-down*, diminui a tensão de saída em relação à sua fonte.

Desta forma, foi instalado no hardware dos dispositivos atuadores um *Auto Step-UP and Step-Down DC DC Converter* da empresa Canton-Power Lda. semelhante ao ilustrado na Fig 4.26, de forma a regular a tensão de saída da bateria para 5 volts, esteja ela totalmente carregada ou descarregada, de forma a poder alimentar o circuito com uma tensão constante. Este conversor converte tensões de entrada entre 0.9 volts e 6 volts, para uma saída constante de 5 volts com uma eficiência que varia entre os 61% e os 85%

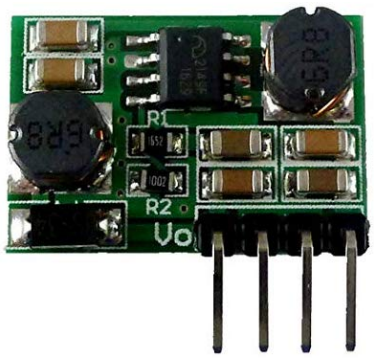

Figura 4.26: Conversor DC para DC com saída de 5 volt.

4.9.5 Carregador bateria dos dispositivos sensores

De forma a poder gerir a carga da bateria LiPo dos dispositivos sensores, foi utilizado um pequeno carregador linear que pudesse ser integrado no circuito eletrónico e soldado na PCB. Este carregador admite uma tensão de entrada entre 4.5 e 8 volts o que se revela ajustado ao painel solar utilizado. A corrente máxima de carga situa-se nos 1000 mA o que se torna mais do que suficiente para um painel solar que apenas fornece 60 mA. A tensão de carga situa-se nos 4.2 volts com uma margem de erro de cerca de 2%. Este carregador possui dois pequenos LEDs identificados como "OK"e "CR"de cor verde e vermelha respetivamente. Estes LEds permitem saber se a bateria está totalmente carregada (caso o LED verde esteja aceso), ou em se a mesma se encontra em carga (LED vermelho aceso). Além disso, este módulo vem programado com uma função específica de carga, que permite entrar no modo de espera quando a bateria se encontra totalmente carregada. Quando a capacidade da bateria for inferior a 80%, o módulo inicia novamente o carregamento.

Figura 4.27: Carregador de baterias LiPo dos dispositivos sensores

4.9.6 Baterias

Para que os dispositivos atuadores e sensores funcionem durante o período noturno, ou em períodos diurnos em que os painéis fotovoltaicos não forneçam energia suficiente, os mesmos são alimentados por baterias Li-Po. A bateria dos dispositivos sensores possui uma capacidade de 1000 mA, enquanto a bateria dos sistemas atuadores possui uma capacidade de 4000 mA conforme ilustrado na fig 4.28. A mesma possui ainda um sistema de proteção de sobrecarga e sobredescarga. A carga e descarga será monitorizada pelos respetivos carregadores solares pelo que não é necessário um controlador de carga.

Figura 4.28: Bateria de Li-Po de 3.7 volts

4.10 Desenho do Circuito

Os circuitos electrónicos ilustrados nas figura 4.30 e figura 4.29 foram desenhados recorrendo ao software Fritzing e permitem observar a ligação entre todos os componentes e módulos electrónicos.

Na figura 4.29 representa o circuito utilizado no dispositivo sensor. Nesta imagem é possível observar o microcontrolador Atmega 328-PU ao centro. Os pinos conetados estão representados a cor verde e os desconetados a cor vermelha. Do lado esquerdo da imagem pode ser observado o oscilador (OSC) externo de 16 Mhz ligado a dois condensadores de 22pF.

Na parte superior da imagem podemos observar o sensor de humidade conetado ao pino nº 28 do microcontrolador, o que equivale a pino analógico A5, a bateria de Li-Po de 3.7 volts responsável por alimentar o circuito quando o painel solar não fornece a energia

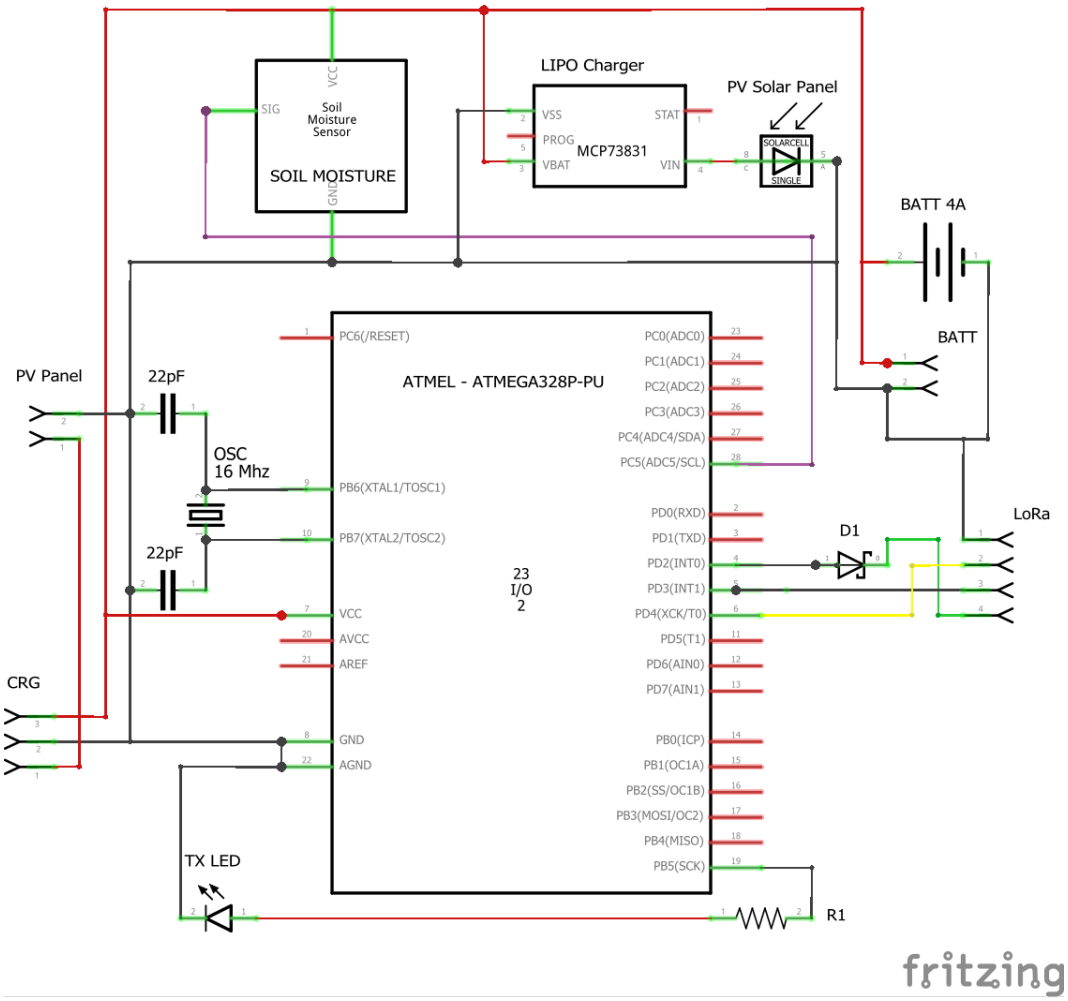

Figura 4.29: Circuito electrónico do dispositivo sensor.

suficiente, e o carregador de bateria LiPo. Do lado direito da imagem podemos observar o díodo *zener* D1 que está ligado ao pino nº xxx do microcontrolador e que serve de alimentação ao módulo de comunicação LoRa. O díodo *zener*faz baixar a tensão de saída para um limite máximo de 3.3 volts. O módulo de comunicação está ainda ligado ao pino nº xxx e nº xxx responsáveis pelo envio (Tx) e recepção (RX) das mensagens rádio. No canto inferior direito temos a resistência R1 limitadora de corrente ao LED1. Este LED acende cada vez que é enviado uma mensagem para o dispositivo atuador. Na tabela 4.1 podemos observar a funcionalidade dos pinos do microcontrolador usados neste circuito, de forma a melhor compreender as ligações entre o microcontrolador e os componentes integrantes.

A figura 4.30 representa o circuito utilizado no dispositivo atuador. Nesta imagem é possível observar o microcontrolador Atmega 328-PU do lado esquerdo da imagem. Os pinos conetados estão representados a cor verde e os desconetados a cor vermelha.

Tabela 4.1: Pinout do dispositivo sensor.

COMPONENTE	PINO	A/D	TIPO
Sensor	5	Analógico	Input
LoRa Rx	2	Digital	Input
LoRa Tx	3	Digital	Output
LED1	12	Digital	Output
OSC	3	Digital	XT1
OSC	4	Digital	XT2

Na parte superior da imagem encontra-se representado a válvula solenoide responsável pela irrigação. A mesma encontra-se ligada a um transístor de potencia. Este transístor é ativado mediante o pino nº xxx do microcontrolador. Existe ainda um super condensador que estabiliza a tensão do circuito quando a válvula é ativada ou desativada, eliminando os picos de tensão. Do lado direito da imagem podemos observar a ligação do painel solar de 6 volts ao carregador solar da empresa Adafruit. este carregador possui 2 LEDs de *status* da bateria. Na zona inferior da imagem está também representado um LED RGB referente ao *staus* das comunicações.

Na tabela 4.2 podemos observar a funcionalidade dos pinos do microcontrolador usados neste circuito, de forma a melhor compreender as ligações entre o microcontrolador e os componentes integrantes.

Após uma primeira fase de implementação dos componentes numa *breadboard* de teste, passou-se para a fase de implementação do protótipo a titulo definitivo, fabricando a PCB.

4.11 Placa Circuito Impresso

A placa de circuito impresso (PCB) é uma das maravilhas na produção moderna de componentes eletrónicos. É feita de fibra de vidro com pistas de cobre que atuam como fios para que os componentes possam ser ligados. Para que os componentes sejam devidamente fixados na placa, a produção requer furos a serem realizados na PCB. A construção de PCBs trás uma serie de vantagens na realização de projetos quer industriais, quer particulares. De entre as vantagens podemos destacar algumas de maior importância:

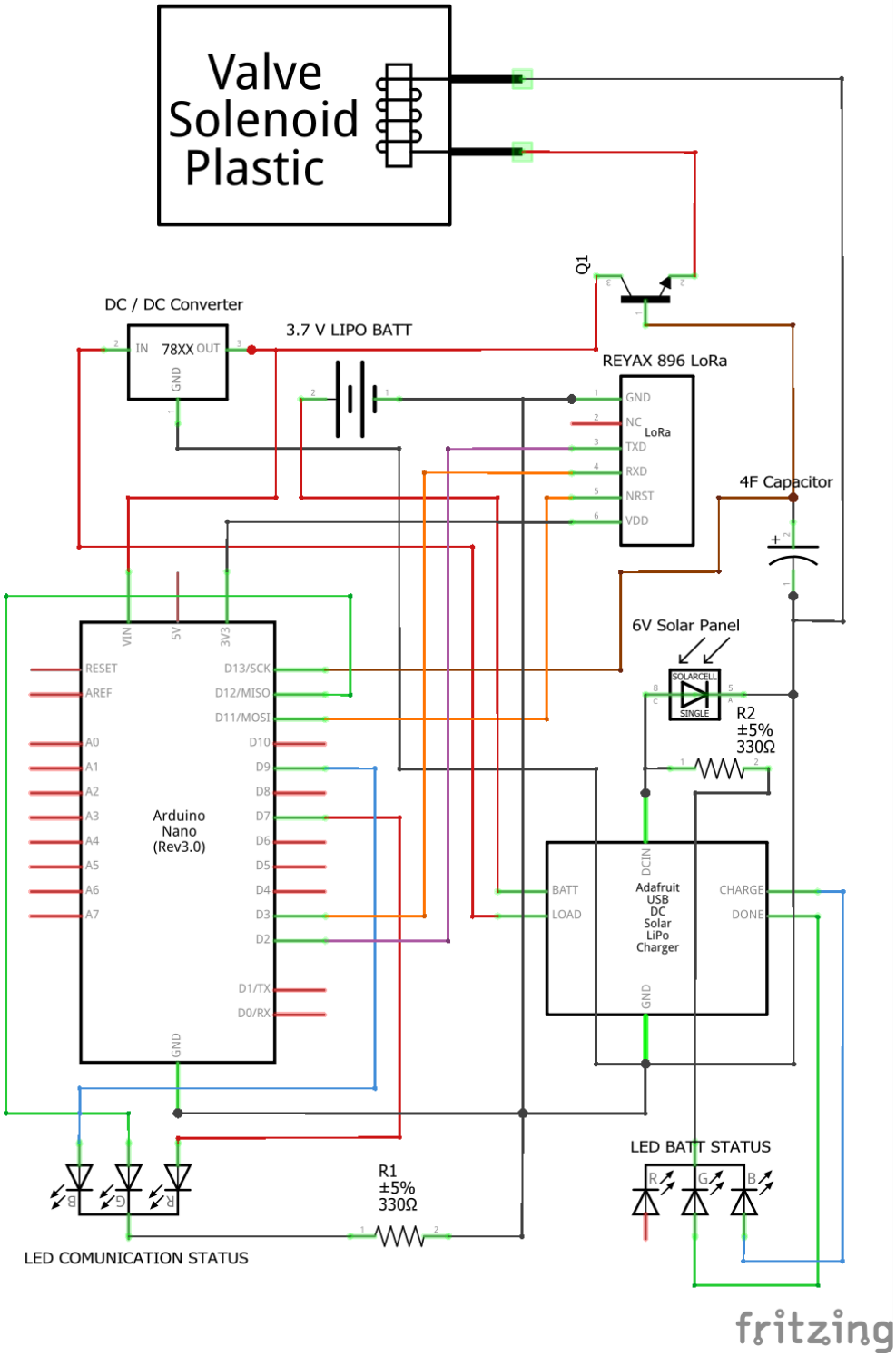

Figura 4.30: Circuito electrónico do dispositivo atuador

Tabela 4.2: Pinout do dispositivo atuador.

COMPONENTE	PINO	A/D	TIPO
Transistor	1	Digital	Output
LoRa Rx	2	Digital	Input
LoRa Tx	3	Digital	Output
LED1 Vermelho	6	Digital	Output
LED1 Verde	7	Digital	Output
LED1 Azul	8	Digital	Output

1. Todos os componentes são fixos PCB:

   A placa de circuito impresso não tem fios com ligações complexas. Desta forma, faz da sua utilização um processo simples. Uma vez que os componentes da placa são fixos e fáceis de identificar, torna-se fácil a sua manutenção ou mesmo a substituição de componentes.

2. Preocupação mínima sobre curtos-circuitos e ligações erradas:

   As chance de curtos-circuitos nas PCBs e ligações erradas na sua produção são mínimas, uma vez que as faixas de cobre são incorporadas na placa. Outro motivo pelo qual as PCBs tem poucas hipóteses de criar curto-circuitos é que são projetadas usando um computador, portanto, quaisquer erros podem ser corrigidos antes da produção.

3. Não há necessidade de uma inspeções mais aprofundadas:

   Após a fase de desenvolvimento e testes, utilizando para o efeito aplicações informáticas, podemos eliminar uma inspeção mais aprofundada antes de usar o produto. Como não há necessidade de nos preocuparmos com circuitos interrompidos, existe uma maior probabilidade de funcionar dos que os circuitos montados manualmente nas *breadboards*.

4. Produções em massa com custo reduzido:

   A criação de múltiplas placas de circuito a partir de um desenho é fácil de realizar, uma vez que só precisamos de imprimir o desenho e grava-lo numa uma placa de cobre. Isto permite a produção em massa mais económica do que placas criadas

manualmente. Podemos guardar o desenho esquemático e reproduzi-lo a qualquer momento que a placa seja necessária.

5. Perfeito para fazer a reprodução:

   Como acima mencionado, as placas de circuito impresso são ideais para a reprodução de múltiplas placas. Uma vez que é gerada por computador, podemos criar placas de circuito impresso uniformes usando a mesma disposição esquemática. Assim, a criação de placas que são idênticas é fácil de concretizar.

Foi pensando nas vantagens acima descritas que se decidiu desenvolver uma placa de circuito impresso (PCB) onde todos os módulos constituintes do projeto fossem montados definitivamente e que facilitasse as ligações entre os diversos componentes. As PCBs foram desenvolvidas no software Fritzing e foram produzidas com recurso a uma impressora Laser, papel transfer e utilizará o método de gravação com reagente ao cobre. As PCBs ficarão instaladas dentro de uma caixa construída especificamente para o efeito impressas numa impressora 3D.

Na PCB foram soldados diversos componentes necessários ao bom funcionamento do circuito, bem como os circuitos de carga da bateria, o conversor DC/DC e o microcontrolador.

Na figura 4.31 pode ser observado o aspeto geral da PCB dos dispositivos sensores e, neste caso especifico, conetado a um sensor de humidade do solo. Do lado direito da figura, está representado o painel solar e o carregador da da bateria. Como são módulos externos, foram deixados *pin headers* para que possam ser soldados os cabos provenientes dos mesmos. Quanto ao sensor de humidade aqui representado, o mesmo também não será soldado diretamente na PCB visto que o mesmo será colocado numa zona especifica da caixa de forma a que fique em contacto com o solo, isolando o mesmo da parte eletrónica do dispositivo. Para tal, recorreu-se à mesma técnica dos componentes exteriores ao circuito; aliás, todos os sensores que necessitem de variáveis ambientais externas à caixa onde fica alojada a PCB, serão soldados recorrendo a *pin headers*. Na parte superior da PCB, foi também deixado um pino digital D12 disponível, caso o sensor a instalar seja um sensor com um sinal de output digital.

Os componentes instalados possuem as seguintes funcionalidades:

- R1: Resistência limitadora da corrente do TX LED;
- TX LED: LED verde que acende quando é realizada uma trânsmissão de dados;
- 22pF : Condensadores cerâmicos de apoio ao oscilador;
- OSC: Oscilador de 16MHz
- LoRa: Pinos de ligação do mnódulo de comunicação LoRa;

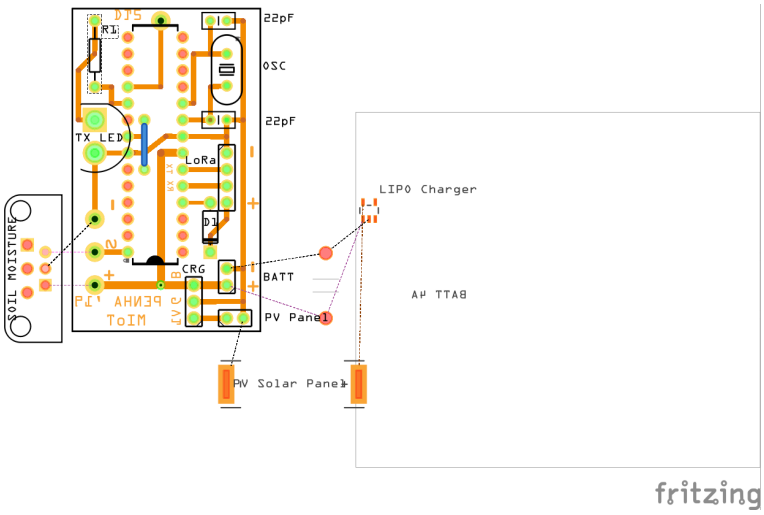

Figura 4.31: Desenho da PCB do dispositivo sensor

- BATT: Pinos de coneção da bateria

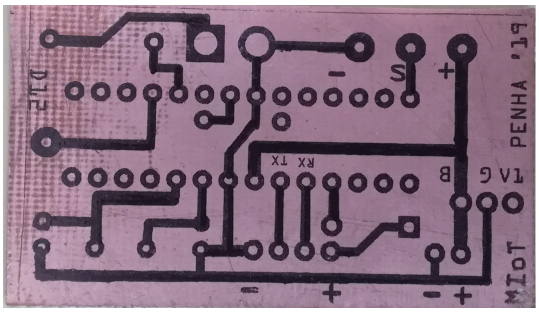

Figura 4.32: Impressão da PCB do dispositivo sensor em placa de cobre.

Na figura 4.34 pode ser observado o aspeto geral da PCB do dispositivo atuador. Neste dispositivo recorreu-se à plataforma de prototipagem Arduino, na sua versão NANO, visível na parte central do desenho da PCB. Na parte superior da imagem é também visível a bateria, o módulo carregador solar, e o painel solar. Foram instalados diversos componentes eletrónicos de forma a solucionar diversos aspetos inerentes ao funcionamento do circuito, nomeadamente:

- **TIP 120:** Transístor responsável pela abertura e e fecho da válvula solenoide.
- **R4:** Resistência limitadora de corrente do LED "VALVE ON LED".
- **4F:** Condensador de estabilização do circuito.
- **R5:** Resistência limitadora de carga do condensador.

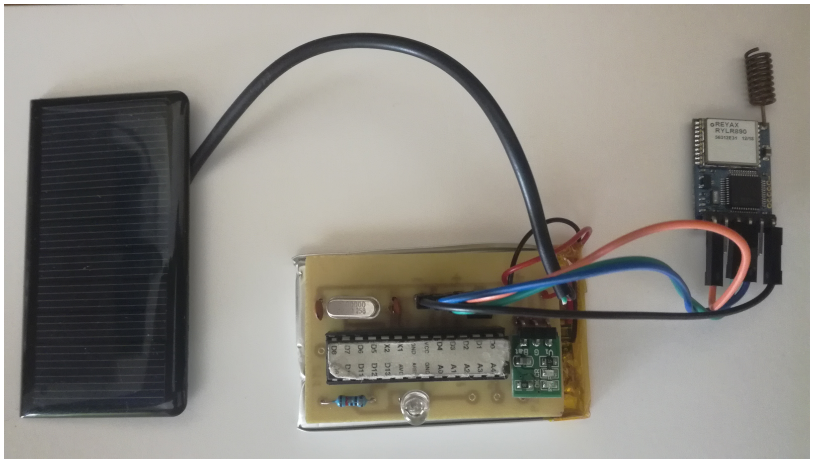

Figura 4.33: Dispositivo sensor finalizado. Na figura, pode ser observado o microcontrolador, o painel solar e o módulo de comunicação LoRa.

- **5V Regulador:** Conversor de tensão. Estabiliza a tensão em 5 volts quer a tensão fornecida pela bateria/carregador seja superior ou inferior a este valor.
- **LED COM STAT:** LED de status da transmissão.
- **R1:** Resistência limitadora de corrente do LED "COM STAT".
- **VALVE:** Zona de ligação da válvula de irrigação.
- **LoRa:** *Pin header* de ligação do módulo de comunicação.
- **PWR IN:** Zona de ligação da alimentação de energia proveniente do carregador/bateria.

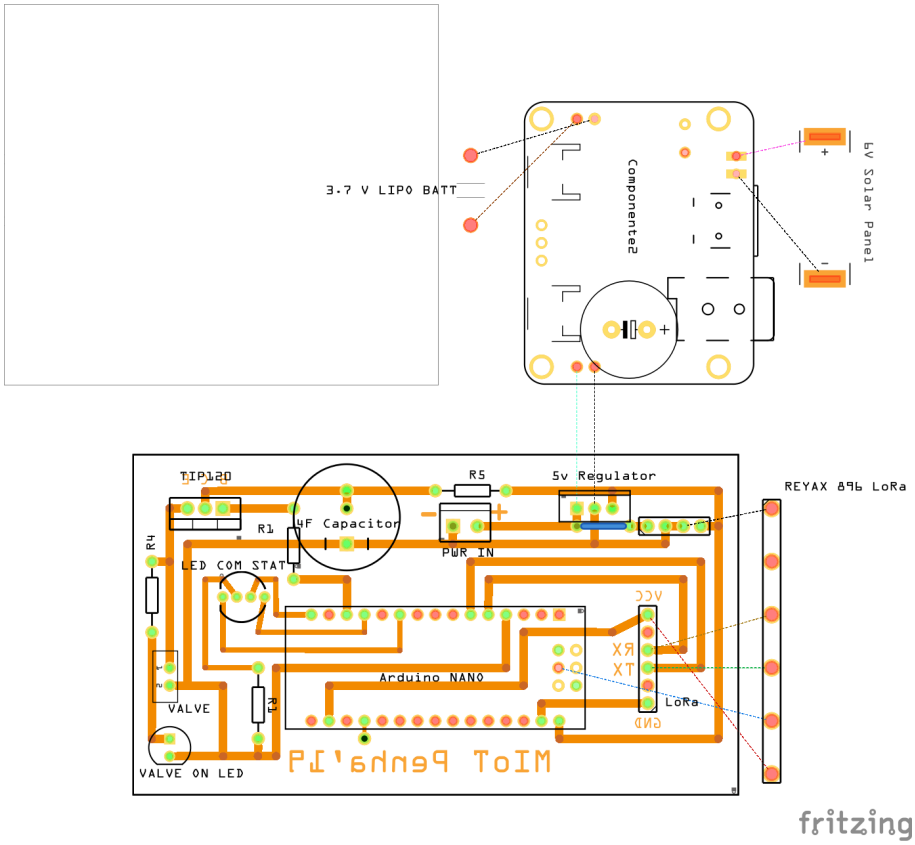

Figura 4.34: Desenho da PCB do dispositivo atuador

Capítulo 5

Resultados

Neste capítulo serão apresentados os resultados obtidos durante a fase de implementação do protótipo funcional. Numa perspectiva de complementaridade entre a teoria e o conhecimento empírico é minha intenção apresentar os dados recolhidos durante os testes realizados *"in field"*. No sentido de proceder a uma análise mais objetiva, optou-se por separar os campos de análise em três grandes eixos complementares.

O primeiro eixo de análise visa conhecer a cobertura dos dispositivos e quais os fatores que a podem influenciar. São apresentados resultados referentes ás distancias de comunicação e ás diferentes parametrizações dos módulos de comunicação.

O segundo eixo de análise recai sobre a comunicação entre dispositivos, nomeadamente os resultados observados na diferentes portas de comunicação (COM), quer nos dispositivos sensores como nos dispositivos atuadores e a configuração dos módulos de comunicação.

O terceiro eixo de análise refere-se aos consumos dos dispositivos. Um dos fatores com mais importância em redes LPWAN, são sem dúvida a otimização do consumo energético dos dispositivos que integram a rede. São apresentados os resultados das medições de consumo dos componentes e dos dispositivos.

5.1 Cobertura

Na tecnologia de rádio, existem essencialmente três características que podem ser usadas para caracterizar uma rede de rádio:

1. alcance / distância;
2. A velocidade de transmissão de dados;
3. O consumo de energia.

É difícil considerar todos os três critérios com a mesma ênfase, já que as leis físicas estabelecem limites claros neste caso: Por exemplo, o LoRaWAN pode transmitir dados

por longas distâncias e requer relativamente pouca energia, mas tem uma baixa taxa de transferência de dados. O WiFi e o Bluetooth, por exemplo, alcançam taxas de transmissão de dados muito altas, mas o consumo de energia é comparativamente alto e seu alcance é muito baixo[39]. Todos os utilizadores de *smartphones* estão bem cientes dessa "sede"de energia. As estações de base dos grandes provedores de telecomunicações oferecem taxas de transmissão de dados elevadas e alcances relativamente altos, mas uma grande quantidade de energia deve estar disponível de forma a permitir alimentar estas tecnologias. Desta forma, a fonte de alimentação é sempre um fator de planeamento bastante importante para as referidas instalações.

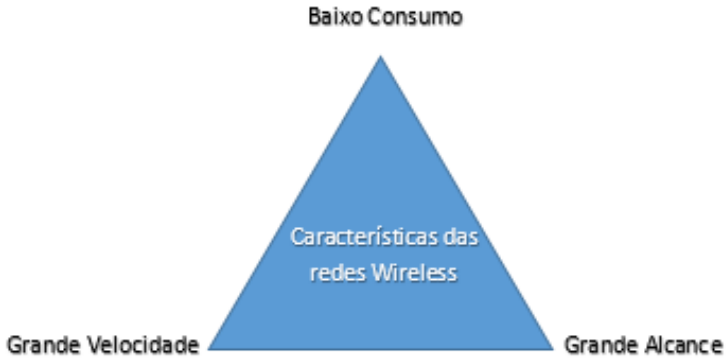

Figura 5.1: Características das WSN.

Em geral, é possível uma otimização para um máximo de 2 dos critérios mencionados na figura 5.1. Portanto, é necessário tomar a decisão de quais propriedades recebem prioridade. Foi principalmente com base nos critérios de baixo consumo e com necessidade de enviar pequenas quantidades de dados, descartando por completo a necessidade de velocidade nas transmissões de dados, que a escolha recaiu sobre a tecnologia LoRa.

Existem vários aspetos que influenciam a cobertura / alcance de uma transmissão LoRa:

Link Budget

O *link budget* indica a qualidade de um canal de transmissão de rádio. Usando um modelo simples, o *link budget* pode ser calculado adicionando a potência de transmissão (*Transmmitter Power*, Tx), sensibilidade do receptor (*Receiver Power*, Rx), ganho da antena e perda de energia entre o transmissor e o recetor também chamado *free-space path loss* ou FSPL.

Fator 1: Free Space Path Loss

Ao duplicar a distância, o FSPL para LoRaWAN aumenta em 6 dB, de modo que as perdas do sinal de rádio estão sujeitas a uma função logarítmica.

$$FSPL = \left(\frac{4\pi d}{\lambda}\right)^2 = \left(\frac{4\pi d f}{c}\right)^2 \tag{5.1}$$

Em que:

- FSPL = Free Space Path Loss
- d = distância entre Tx e Rx em metros
- f = frequência em Hertz

Além da perda de energia em função da distância, fatores como reflexão e refração de ondas de rádio em objetos, podem levar à sobreposição de ondas de rádio, o que também pode ter um efeito negativo no alcance.

Fator 2: Perda de sinal devido a elementos estruturais

A perda de sinal causada por estruturas, isto é, a absorção de sinais de rádio ao penetrar diferentes obstáculos, tais como edifícios, influencia a recepção de sinais transmitidos e pode encurtar consideravelmente o alcance. Por exemplo, o vidro reduz o sinal em apenas 2dB. Isso afeta muito menos o alcance do que uma parede de betão de 30 cm[40]. A tabela 5.1 lista vários materiais e seus efeitos típicos nos sinais de rádio.

Tabela 5.1: Tipo de material e a sua influência na perda de sinal (Path Loss)

MATERIAL	PATH LOSS (dB)
Vidro (6mm)	0.8
Vidro (13mm)	2
Madeira (76mm)	2.8
Tijolo (89mm)	3.5
Tijolo (178mm)	5
Tijolo (267mm)	7
Betão (102mm)	12
Parede Pedra (203mm)	12
Tijolo de Cimento (192mm)	14
Parede de Pedra (406mm)	17
Betão (203mm)	23
Betão Reforçado (89mm)	27
Parede de Pedra (610mm)	28
Betão (305mm)	35

Fator 3: *Fresnel Zone*

Para cobrir efetivamente as longas distâncias e obter um bom *Link Budget*, também é importante estabelecer, sempre que possível, uma linha de visão direta (LoS) entre o transmissor e o recetor. Na transmissão de rádio, áreas espaciais específicas entre a linha de visão são referidas como *Fresnel Zones*. Se houver objetos nessas zonas, elas podem ter uma influência negativa na propagação de ondas, embora haja geralmente contato visual entre as antenas de transmissão e recepção. Para cada objeto localizado na FZ, o nível do

sinal é reduzido e o alcance é reduzido. Na figura 5.2 podemos observar um exemplo de uma *Fresnel zone*.

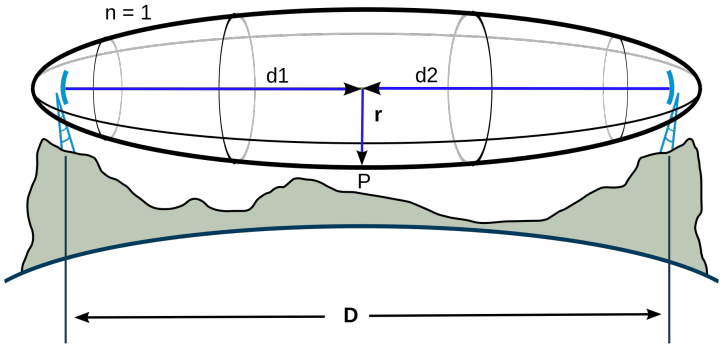

Figura 5.2: Fresnel Zone: D é a distância entre o transmissor e o receptor; r é o raio da primeira zona de Fresnel (n=1) no ponto P. P está a uma distância d1 do transmissor, e d2 do receptor.

Em redes LoRaWAN normalmente são usadas antenas omnidirecionais. Isso faz com que a energia emitida se espalhe no plano horizontal, onde os nós da rede e os gateways estão localizados. Na Europa, o limite de potência de uma banda ISM é definido para 14 dBm para uma frequência de 868 MHz. Além disso, o ganho máximo da antena é limitado a 2,15 dBi.

Factor 4: Spreading Factor

Uma rede LoRaWAN usa fatores de propagação (SF) para definir especificamente a taxa de transferência de dados relativa ao alcance. Nas redes LoRaWAN, inclusive no módulo de comunicação usado no SWAIS, são usados SF entre 7 e 12. Devido à sua modulação *Chirp Spread Spectrum (CCS)*, assim como várias frequências de mudança de fase usadas para nos *"chirps"*, insensíveis à interferência, propagação de múltiplos caminhos e desvanecimento. *"Chirps"* são usados para codificar dados em redes LoRaWAN no lado Tx, enquanto *"chirps"* inversos são usados no lado Rx para decodificação de sinais. Os SFs acima indicam quantos *"chirps"* são usados por segundo, e definem taxas de bits, por potência irradiada e alcance espectável.

O SF9, por exemplo, é 4 vezes mais lento que o SF7 em termos de taxas de bits. A escalabilidade do LoRaWAN é alcançada pelos FSs. Quanto mais lenta a taxa de bits, maior a energia por conjunto de dados e maior o alcance. O LoRaWAN suporta uma adaptação automática dos fatores SF, dependendo da configuração da rede, a chamada Taxa de Dados Adaptativa (ADR).

Os módulos de comunicação LoRa utilizados no SWAIS, permitem a configuração de diferentes parâmetros, nomeadamente:

1. *Spreading Factor* - Quanto maior o SF, melhor será a sensibilidade, no entanto, a transmissão levará mais tempo ($7 \simeq 12$);

2. *Bandwith* - quanto menor for a a largura de banda, melhor será a sensibilidade mas, a transmissão demorará mais tempo($0 \simeq 12$).

3. *Coding Rate* - Será mais rápida se programada para o menor valor ($1 \simeq 4$);

4. *Programmed Preamble* - Código do Preâmbulo. Se o código preambular for maior, resultará na menor oportunidade de perder dados ($4 \simeq 7$).

Por tudo o que foi exposto anteriormente, a configuração dos módulos de comunicação deve ter em conta, a distancia entre os dispositivos sensores e os dispositivos atuadores, a morfologia do terreno e a sua localização. Uma otimização dos parâmetros configuráveis, pode levar a uma considerável redução no consumo dos módulos de comunicação sem prejudicar a performance. O fabricante recomenda uma configuração padrão para comunicações num raio inferior a 3Km com os parâmetros 10,7,1,7, correspondendo respetivamente ao *Spreading Factor*, *Bandwith*, *Coding Rate* e *Programmed Preamble*, e outra para comunicações superiores a 3Km com os parâmetros definidos para 12,3,1,7 respetivamente.

Numa primeira fase do projeto, a cobertura por parte dos dispositivos foi uma cobertura estimada conforme o observável nas figuras 4.2 e 4.3 do capítulo 4. No entanto, foi sentida a necessidade de confirmar estas coberturas. Para tal, e após a construção de protótipos funcionais tanto dos dispositivos sensores e dos dispositivos atuadores, foram efetuados testes *in field* com os protótipos. Na tabela 5.2 podem ser observados os alcances verificados em três situações distintas, bem como com duas parametrizações diferentes sugeridas pelo fabricante.

Tabela 5.2: Alcance verificado dos Módulos de comunicação LoRa.

SITUAÇÃO	CONFIGURAÇÃO	ALCANCE (Km)
LoS	12,3,1,7	8
LoS	10,7,1,7	5
No LoS	12,3,1,7	4
No LoS	10,7,1,7	1
Urbano	12,3,1,7	3
Urbano	10,7,1,7	0,5

Os resultados obtidos revelaram-se bastante satisfatórios tendo em conta o custo dos módulos LoRa e do próprio protótipo em si. De salientar que as comunicações entre dispositivos não recorrem a gateway LoRa, ou seja ,não necessitam de qualquer infraestrutura adicional.

5.2 Comunicação entre dispositivos

Quando os dispositivos sensores ou atuadores são ligados, o primeiro passo é a configuração do módulo de comunicação LoRa. Essa configuração é composta seguintes parâmetros:

- ADDRESS - O endereço é considerado como a identificação do transmissor ou receptor especificado.
- NETWORKID - Define o ID da rede Lora. Esta é uma função de grupo. Apenas configurando o mesmo NETWORKID os módulos podem comunicar entre si. Se o endereço do módulo recetor especificado pertencer a um grupo diferente, este não será capaz de comunicar com os outros módulos.
- BAND - para definir a frequência central da banda sem fios. O transmissor e o receptor são obrigados a usar a mesma frequência para comunicar entre si.
- PARAMETER - para definir os parâmetros de RF sem fios. O transmissor e o receptor são necessário para definir os mesmos parâmetros de forma a comunicar uns com os outros. Os parâmetros são os seguintes:
    1. Spreading Factor: quanto maior o SF, melhor a sensibilidade, mas a duração da transmissão irá aumentar.
    2. Bandwidth: Quanto menor a largura de banda, melhor a sensibilidade, mas a duração da transmissão irá aumentar.
    3. Coding Rate: a taxa de codificação será a mais rápida se for definida como 1.
    4. Programmed Preamble: Se o código de preâmbulo for maior, isso resultará na redução da possibilidade de perda de dados. Geralmente código de preâmbulo pode ser definido acima de 10 se não houver grandes preocupações com a duração da transmissão.
- MODE - Se configurado para 0, o módulo encontra-se no modo de transmissão e receção; se configurado para 1, o módulo encontra-se em *sleep mode*, diminuindo o consumo energético.
- IPR - estabelece a velocidade da porta UART.

Existem outros parâmetros configuráveis que estão listados no manual do fabricante, mas os supracitados são os que foram utilizados na configuração dos módulos deste projeto. Os parâmetros são configuráveis através de comandos AT. A cada comando enviado, o módulo deverá responder com "OK"caso o mesmo tenha sido aceite. Caso o utilizador queira saber qual o valor de determinado parâmetro, pode enviar um comando AT seguido do sinal "?"ex: AT+MODE?. A resposta deverá ser o valor que está

programado no módulo antecedido do sinal "+"ex: +MODE=1. Na figura 5.3 podemos observar a configuração inicial do módulo LoRa do dispositivo atuador. As primeiros seis linhas "+OK"representam a resposta aos comandos enviados para a configuração do módulo LoRa. As últimas seis linhas referem-se à resposta por parte do módulo após "questionado"acerca dos valores dos parâmetros configurados anteriormente. A cada reinicialização dos dispositivos sensores e atuadores, é realizada uma nova configuração.

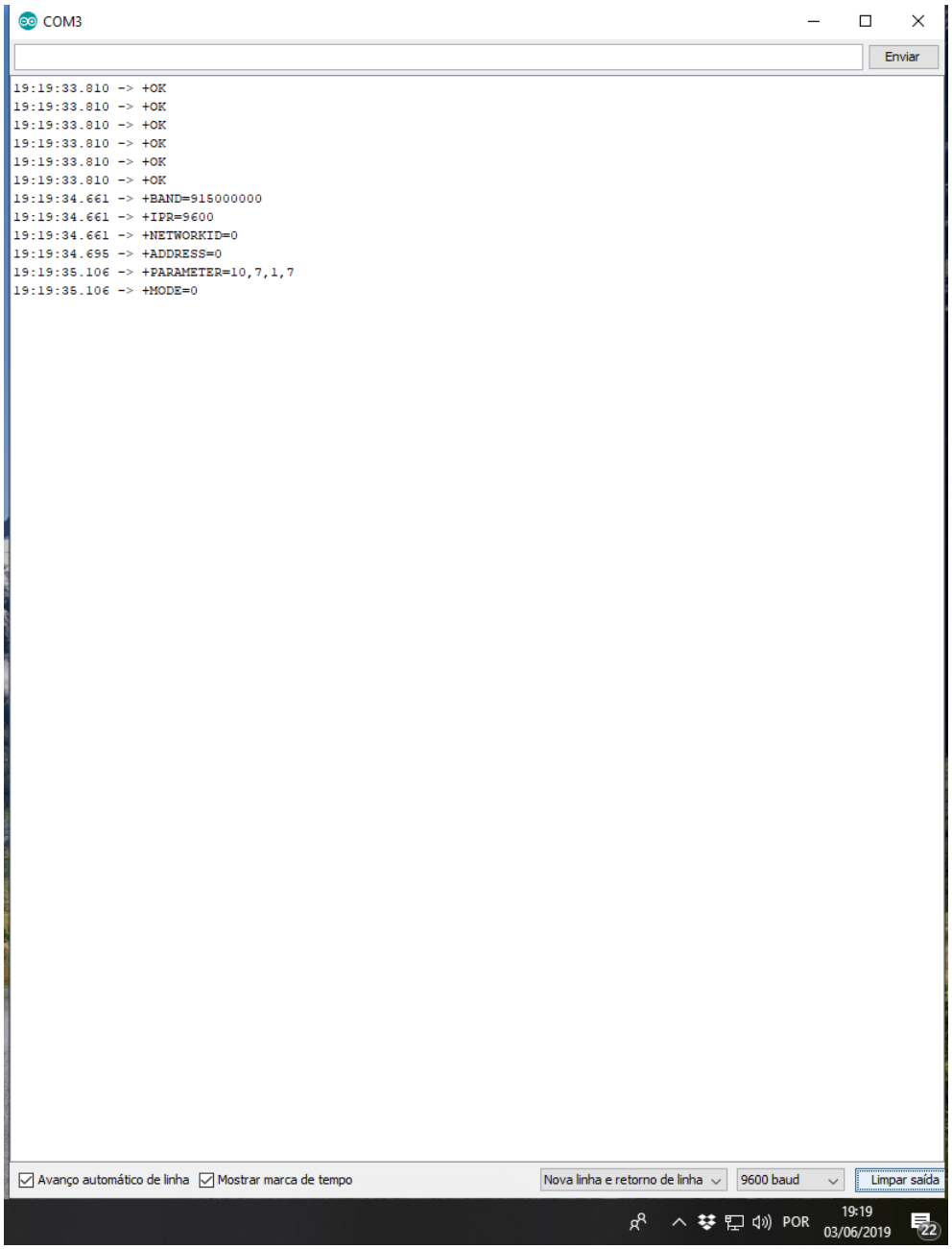

Figura 5.3: Configuração inicial do módulo LoRa.

Como já explicado no capítulo 4 subsecção 4.3.1, o envio do payload por parte dos

dispositivos sensores é realizado através de um comando AT seguido do endereço do módulo e os dados a enviar (AT+SEND=0). O payload do SWAIS não é mais do que uma string contendo o número de bytes do *payload* e um objeto JSON contido na *string* "tempString". Esse objeto JSON contem o ID do sensor e o seu valor. A concatenação destas *strings* cria uma nova string chamada "stringToSend". Por fim, é realizado um *"print"* da *string* "stringToSend"para o módulo de comunicação LoRa através da função "LoraSerial.print".

Listagem 5.1: Linhas de código responsáveis pelo envio do *payload* dos dispositivos sensores

```
1    stringToSend = "AT+SEND=0," + String(tempString.length()) + "," + tempString;
2    LoraSerial.println(stringToSend);
```

Na figura 5.4 podemos verificar os *payloads* a serem enviados pelo dispositivo sensor através da porta de comunicação serie da janela do IDE do Arduino. É também observável uma mensagem *"Awake"* e *"Sleep"* referente ao módulo de comunicação LoRa, dependendo do estado de funcionamento em que se encontra. Além da informação acima descrita, o *output* informa-nos também da tensão da bateria dos dispositivo sensor, apesar de este valor não ser transmitido para o dispositivo atuador.

A figura 5.5 apresenta uma imagem da janela referente à porta de comunicação do IDE do Arduino, pertencente ao dispositivo atuador. Na imagem pode ser visualizado a configuração inicial do módulo de comunicação LoRa e, seguidamente, a recepção de dois *payloads* entre as linhas picotadas. O *payload* é constituído pelo tipo de sensor "s_type"e pelo seu valor "s_value". O módulo LoRa ao receber a comunicação disponibiliza automaticamente o valor SNR e o RSSI. Posteriormente o valor do RSSI e o endereço do sensor que enviou os dados (s_add) são acrescentados à *string* de chegada e é criada uma nova *string* a qual é apresentada como o *payload* recebido. Na figura podemos observar que após receber o primeiro *payload*, o dispositivo atuador irá verificar se existe algum sensor do mesmo tipo já guardado no *array* de sensores. Como não existe, pois o sistema foi reiniciado, o dispositivo atuador guarda o primeiro dispositivo sensor juntamente com a *struct* de dados que compõe o *payload*, na posição 0 do *array* de sensores. Podemos verificar que o sistema disponibiliza a informação de qual o melhor RSSI de cada sensor do mesmo tipo. No caso do primeiro *payload*, o mesmo é referente a um sensor do tipo "RS"(*Rain Sensor*) com o endereço "3". Quando é recebido o segundo *payload*, o sistema irá fazer uma pesquisa no *array* de sensores para confirmar ou não a sua existência. Na Fig 5.5 pode ser observado que o segundo *payload* é referente a um sensor do tipo "AT"(*Ambient Temperature*) e como tal, não existe no *array* de sensores. Desta forma, o segundo dispositivo sensor é adicionado ao *array* de sensores na posição 1, juntamente com os dados que o compõe. Assim, o *array* de sensores passa a contar com um total

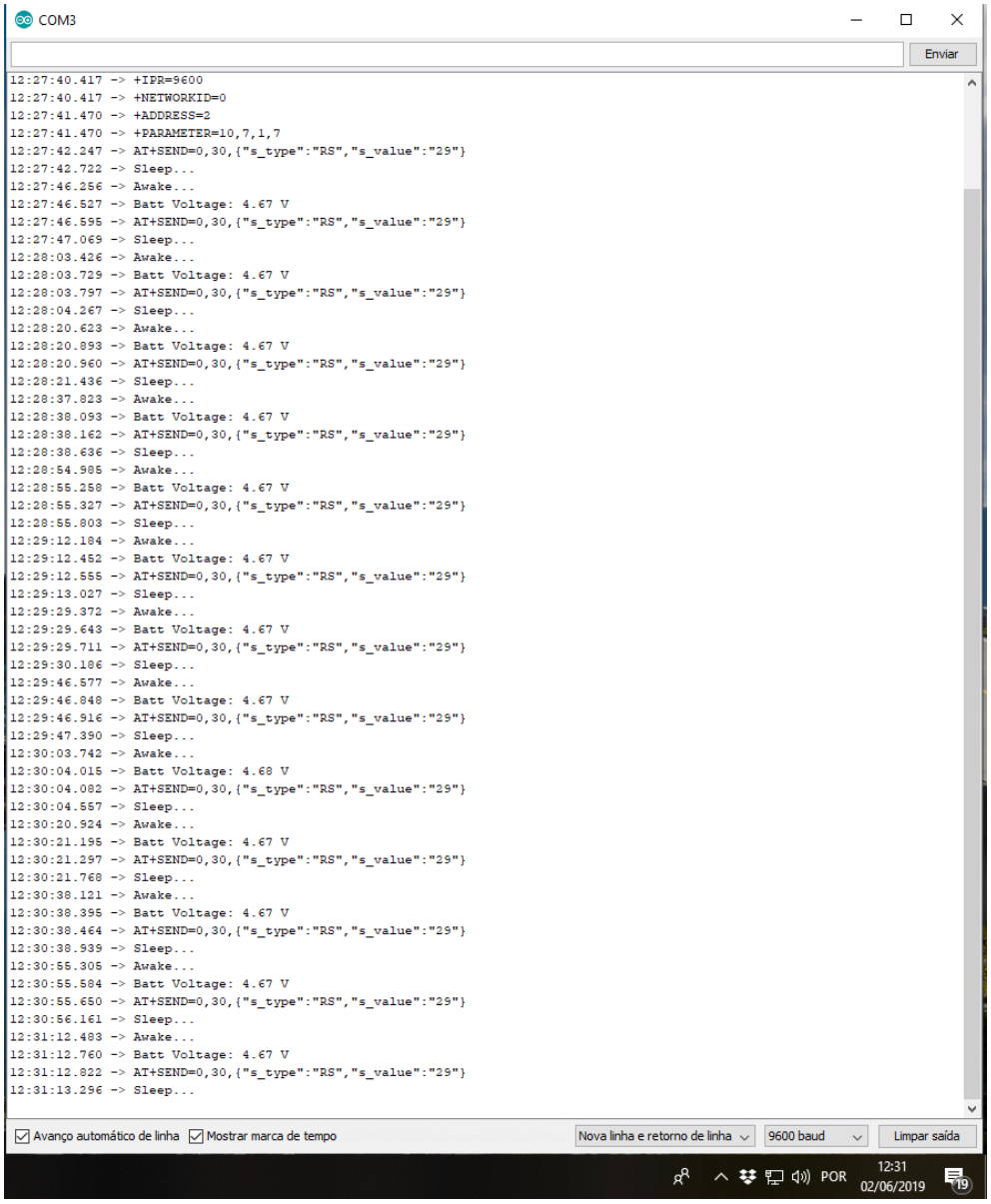

Figura 5.4: Envio do *payload* por parte do dispositivo sensor.

de 2 sensores, um de cada tipo. Caso os dois sensores fossem do mesmo tipo, o sistema iria recorrer ao valor do RSSI que se encontra no *payload* recebido, selecionando o que apresentasse a melhor receção. O melhor valor de RSSI referente a cada tipo de sensor é apresentado no fim de cada mensagem recebida.

5.3 Consumos

De forma a validar os consumos, foram efetuados testes de consumo individuais a cada componente. Os testes foram realizados com recurso a um multímetro FLUKE 87 III, cujo

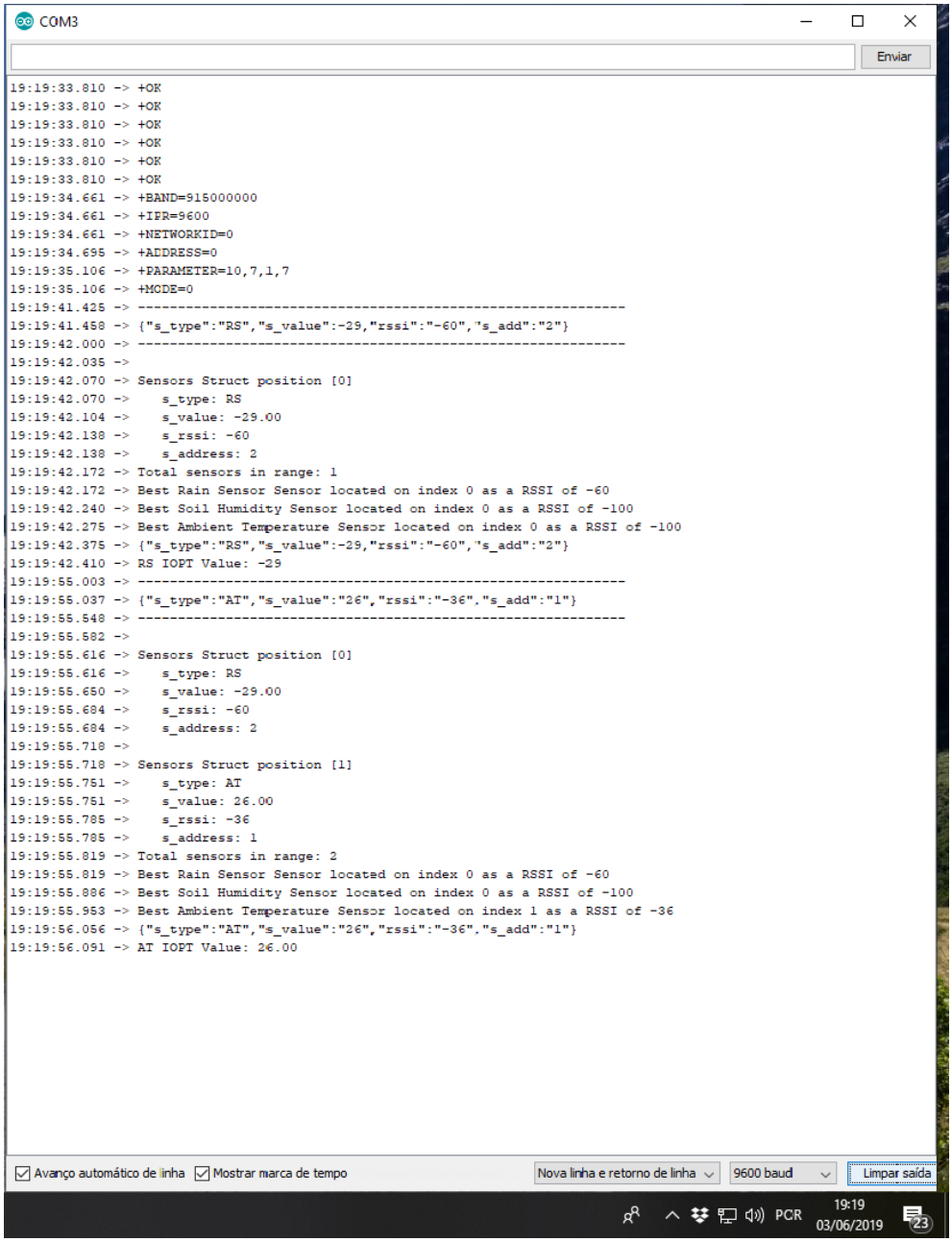

Figura 5.5: *Output* da porta de comunicação do dispositivo atuador.

o mesmo encontra-se equipado com uma funcionalidade que permite observar os valores mínimos, máximos e a média durante um determinado período de tempo.

Nos dispositivos sensores recorreu-se à biblioteca "LowPower.h"de forma a colocar o microcontrolador em *"Power Down Mode"* entre os intervalos de transmissão. O módulo de comunicação LoRa também é colocado em *"Sleep Mode"* entre transmissões. O modo 0 coloca o módulo em modo *"active mode"*, enquanto o modo 1 coloca o módulo em *"sleep mode"*, reduzindo o consumo do mesmo.

Na tabela 5.3 são apresentados os consumos de cada componente de forma individual respeitantes ao dispositivo sensor, e no final, o consumo total do dispositivo. A estes valores terá que ser adicionado o valor do consumo do sensor utilizado no dispositivo que poderá variar consoante o sensor utilizado.

Tabela 5.3: Consumo medido do dispositivo sensor.

COMPONENTE	MAX	MIN
Microcontrolador	15 mA	0.36 mA
Módulo LoRa	43 mA	0.5 uA
Carregador Bateria	5 mA	5 mA
LED Comunicação	15 mA	0 mA
TOTAL	78 mA	5.365 mA

A produção de energia através dos painéis fotovoltaicos instalados tanto nos dispositivos sensores como nos dispositivos atuadores, prevê-se suficiente para as necessidades do sistema. No entanto, além do consumo instantâneo do próprio sistema, a produção tem que ser capaz de carregar as baterias instaladas nos dispositivos sensores e atuadores. Estas irão fornecer a energia para o período nocturno e quando a produção dos painéis fotovoltaicos se revelar insuficiente ou inexistente. Na tabela 5.4 podemos observar os consumos dos componentes instalados no dispositivo atuador, e o total do sistema.

É importante referir que os consumos apresentados, são consumos instantâneos. A variação do consumo tem a ver com outros fatores como por exemplo o estado de carga da bateria. Além desse fator, a média do consumo varia consoante o intervalo de tempo entre transmissões - quanto maior for o intervalo entre as transmissões, menor será a média de consumo.

Tabela 5.4: Consumo medido do dispositivo Atuador.

COMPONENTE	MAX	MIN
Arduino NANO	33 mA	15.5 mA
Módulo LoRa	43 mA	0.5 uA
Carregador Bateria	5 mA	5 mA
LED Comunicação	15 mA	0 mA
Válvuva Solenoide	300 mA	0 mA
TOTAL	396 mA	20.505 mA

Capítulo 6

Conclusão e Trabalhos Futuros

Neste capítulo são apresentadas as conclusões e o trabalho futuro desta dissertação. Serão apresentadas as conclusões relativamente à arquitetura do sistema, à ferramenta de automatização de projeto, bem como ao protótipo funcional implementado. Neste capítulo são também abordados os trabalhos futuros considerados relevantes para o melhoramento do sistema.

6.1 Conclusão

Este trabalho propõe uma arquitetura de automatização de projeto para o desenvolvimento de sistemas auto-configuráveis, baseados em redes de sensores sem fios. A arquitetura é validada a partir da aplicação swaisApp, uma ferramenta de automatização de projeto. Por fim foi implementado o protótipo funcional SWAIS como forma de validação da aplicação.

A configuração modular do sistema automatizado permite que ele seja dimensionado para qualquer sistema. Além disso, outras funcionalidades como monitorização de temperatura, humidade entre os parâmetros, podem fornecer dados relevantes à tomada de decisão.

Tanto em termos de consumo como em termos qualidade da comunicação, provou-se que os sistemas desenvolvidos com base na arquitetura proposta, conseguem obter bons resultados de consumo e de QoS da comunicação concluindo-se que, tanto o uso da arquitetura proposta como o uso da geração automática de código são opções válidas.

Em relação às ferramentas de modelação concluiu-se que a utilização destas ferramentas tais como as RdP e, nomeadamente as IOPT Tools que permitem a geração automática de código, facilitam o desenvolvimento de sistemas complexos, permitindo implementar um variedade de regras de forma fácil e intuitiva, diminuindo a possibilidade de erros de programação, tendo sido possível constatar o bom funcionamento tanto em sistemas sequenciais, mas também em concorrência. Verificou-se que a ferramenta swaisApp con-

segue implementar qualquer modelo desenvolvido na *framework* IOPT Tools, bem como a programação do hardware.

Nesta dissertação foi também apresentada implementação do protótipo SWAIS. Foi demonstrado uma possível cobertura do protótipo. Foi fácil observar a redundância que pode ser obtida em caso de falha de comunicação ou falha de um ou vários sensores pertencentes a um determinado jardim. Seguidamente, foi demonstrada uma implementação *in loco* do protótipo funcional, e realizados diversos testes de alcance entre dispositivos e testes de consumos energéticos. Tendo em conta os resultados obtidos, podemos afirmar que os mesmos foram bastante satisfatórios.

Em relação aos consumos energéticos, o uso de energia solar neste sistema de irrigação é pertinente e significativamente importante para a irrigação de espaços que são geograficamente isolados, onde o investimento em fornecimento de energia elétrica seria dispendiosa.

Foi realizado um trabalho de optimização do hardware e do software tanto nos dispositivos sensores como nos dispositivos atuadores. O objetivo principal foi que não houvesse necessidade de recorrer à rede elétrica para alimentar o sistema, de forma a torna-lo mais versátil, incrementando a sua portabilidade e de facilidade de implementação em locais que não disponham de abastecimento de energia da rede. Para ajudar a atingir esse objetivo foram colocadas baterias e painéis fotovoltaicos em todos os dispositivos. Os consumos dos dispositivos sensores cifraram-se na ordem dos 4 mA quando em hibernação e dos 30 mA quando em modo de transmissão de dados, o que se encontram em linha com o consumo dos sistemas IoT. Nos dispositivos atuadores os consumos observados cifraram-se na ordem das dezenas de mA. No entanto, estes consumos variam bastante consoante o dispositivo atuador esteja com a válvula de irrigação ativada ou não, e do número de transmissões recebidas; quanto mais sensores estiverem "acoplados"a um dispositivo atuador, mais elevado será o seu consumo. Apesar do esforço aplicado neste projeto de forma a reduzir os consumos, ficou claro que a otimização a este nível pode melhorar bastante. Para tal é necessário um estudo mais aprofundado do hardware a utilizar e conhecimentos mais aprofundados de eletrónica e do seu comportamento.

Em relação aos testes de cobertura, o alcance dos módulos LoRa utilizados dependerá sempre do meio envolvente, nomeadamente do relevo e das edificações urbanas. No entanto foram realizados testes com os dispositivos ao nível do solo onde puderam ser verificados coberturas entre dispositivos superiores a 1 km em ambiente urbano e cerca de 8 Km em LoS.

Por fim, podemos concluir que a integração da Internet das Coisas com os sistemas de irrigação pode permitir a automação inteligente destes sistemas, permitindo reduzir o consumo de água e o desperdício. A adopção dessas tecnologias incorpora novos benefícios em sistemas de apoio à decisão, monitorização e gestão dos recursos hídricos, além das preocupações a nível energético.

6.2 Trabalhos futuros

Para maior valorização da aplicação swaisApp reconhecem-se algumas melhorias e novas funcionalidades ou subsistemas que poderiam ser agregados ao projeto.

As melhorias identificadas para a aplicação são a possibilidade de programar diversas plataformas de prototipagem, possibilitando ao utilizador escolher na aplicação swaisApp qual a desejada.

Outro trabalho futuro a realizar será a disponibilidade de utilização de vários protocolos de comunicação. O utilizador poderá escolher qual o protocolo desejado entre a comunicação de dispositivos.

Estas extensões podem ser desenvolvidas recorrendo a diversos ficheiros de configuração tanto para as características do hardware como as características do protocolo de comunicação. Esse ficheiros de configuração de hardware podem facilmente ser integrados na aplicação, pois a mesma encontra-se preparada para o efeito.

Outra potencialidade do sistema assenta na possibilidade futura de analisar sensores que não são conhecidos. O cálculo e comparação destes sensores, através de médias ou intervalos de valores, com sensores vizinhos conhecidos, pode vir a identificar padrões semelhantes e definir um tipo ou uma classificação para estes sensores, podendo ser usados pelo sistema.

Em relação ao protótipo, o trabalho futuro levará em conta a implementação do SWAIS em parques públicos e outros terrenos, a fim de realizar testes de implementação em larga escala e resolver possíveis problemas na comunicação e instalação dos dispositivos. Os dispositivos atuadores terão uma funcionalidade adicional de forma a transmitir os dados recebidos pelos diversos dispositivos sensores para um gateway LoRa. Desta forma, os dados recebidos pelo gateway serão enviados por FTP para uma base de dados central. Após os dados serem guardados em BD, os mesmos poderão ser tratados, analisados e disponibilizados na Internet.

Bibliografia

[1] M. Valím, R. Castro, C. Pedras, e L. Pereira, "Uso del agua en espacios verdes: cálculo y evaluación de estrategias de riego." Talavera de la Reina, Espanha: VII Congreso Ibérico sobre Gestión y Planificación del Agua., 2011. (citado na pág. 1)

[2] C. Pedras, H. Fernandez, F. Martins, R. Lança, e M. Valín, "Estratégias para a gestão da água em espaços verdes: Jardins complexos vs relvados." Ponte de Lima, Portugal: World Urban Parks Congress and 9º Congreso Iberoamericano de Parques y Jardines Públicos., 2015. (citado na pág. 1)

[3] C. Pedras, M. Farrajota, M. Valín, e L. Pereira, "A rega nos espaços verdes públicos. caso de estudo: Campus Gambelas da Universidade do Algarve." Alvôr, Portugal: 10º Congresso da Água, 2010. (citado na pág. 1)

[4] J. Berbel, C. Gutiérrez-Martín, e A. Expósito, "Impacts of irrigation efficiency improvement on water use, water consumption and response to water price at field level," *Agricultural Water Management*, vol. 203, pp. 423 – 429, 2018. (citado na pág. 2)

[5] Agência Portuguesa do Ambiente, I.P. e Governo de Portugal. Programa Nacional para o uso eficiente da água. Accessed on 2019-02-08. [*Online*]. Disponível: http://www.apambiente.pt/_zdata/CONSULTA_PUBLICA/2012/PNUEA/Implementacao-PNUEA_2012-2020_JUNHO.pdf (citado nas págs. 2 e 7)

[6] A. Lourenço, "Redes de rega de campos de golf," Dissertação de mestrado, Instituto Superior de Engenharia de Lisboa, 2013. (citado na pág. 2)

[7] C. Penha, R. Campos-Rebelo, e J. Paulo Barros, "Self-Configurable Wireless Automatic Irrigation System," in *2019 International Young Engineers Forum (YEF-ECE)*, May 2019, pp. 52–58. (citado nas págs. 4 e 40)

[8] K. Bellware. Global Water Shortage Risk Is Worse Than Scientists Thought. Accessed on 2019-02-08. [*Online*]. Disponível: https://www.huffingtonpost.com/entry/water-scarcity-study_us_56c1ebc5e4b0b40245c72f5e?guccounter=1 (citado na pág. 7)

[9] J. B. Hugh Turral e J.-M. Faurès, *Climate change, water and food security*. Food and Agriculture Organization of the United Nations, 2016, http://www.fao.org/3/i2096e/i2096e00.htm, accessed 2019-02-08. (citado na pág. 7)

[10] A. Berg, J. Byrne, e R. Rogerson, "An urban water balance study, lethbridge, alberta: Estimation of urban lawn overwatering and potential effects on local water tables," *Canadian Water Resources Journal / Revue canadienne des ressources hydriques*, vol. 21, n. 4, pp. 355–365, 1996. [*Online*]. Disponível: https://doi.org/10.4296/cwrj2104355 (citado na pág. 8)

[11] N. Durga e M. Ramakrishna, "Smart irrigation system based on soil moisture using iot," *International Research Journal of Engineering and Technology*, vol. 05, n. 06, pp. 2003–2007, 2018. (citado na pág. 8)

[12] S. Bennett, "Brief history of automatic control," *Control Systems, IEEE*, vol. 16, pp. 17 – 25, 07 1996. (citado na pág. 9)

[13] (2019) Gres research group. IOPT tools editor. Consultado em 2019/04/24. [*Online*]. Disponível: http://gres.uninova.pt/IOPT-Tools/ssinitmarking.php (citado na pág. 10)

[14] (2019) National geographic encyclopedia. Consultado em 2019/04/25. [*Online*]. Disponível: https://www.nationalgeographic.org/encyclopedia/irrigation (citado na pág. 10)

[15] B. Spear, "James Watt: The steam engine and the commercialization of patents," *World Patent Information*, vol. 30, n. 1, pp. 53 – 58, 2008. (citado na pág. 11)

[16] R. S. Westfall. (2019) Sir Isaac Newton. Consultado em 2019/06/12. [*Online*]. Disponível: https://www.britannica.com/biography/Isaac-Newton (citado na pág. 11)

[17] A. Mccarthy, N. Hancock, e S. Raine, "Advanced process control of irrigation: The current state and an analysis to aid future development," *Irrigation Science*, vol. 31, 05 2011. (citado na pág. 12)

[18] L. Liu, E. Yu, e J. Mylopoulos, "Analyzing security requirements as relationships among strategic actors," in *Proceedings of the 3rd Symposium on Requirements Engineering for Information Security*, ser. CERIAS '02. West Lafayette, IN: CERIAS - Purdue University, 2002, pp. 4:1–4:14. [*Online*]. Disponível: http://dl.acm.org/citation.cfm?id=2835417.2835421 (citado na pág. 18)

[19] R. Campos-Rebelo, F. Pereira, F. Moutinho, e L. Gomes, "From IOPT Petri nets to C: An automatic code generator tool," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2011, pp. 390–395. (citado nas págs. 18, 33, 41 e 45)

[20] C. A. Petri, "Kommunikation mit automaten," Tese de doutoramento, Universität Hamburg, 1962. (citado na pág. 18)

[21] L. Gomes e J. P. Barros, "Refining IOPT petri nets class for embedded system controller modeling," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Oct 2018, pp. 4720–4725. (citado nas págs. 19 e 20)

[22] F. Pereira, F. Moutinho, L. Gomes, J. Ribeiro, e R. Campos-Rebelo, "An IOPT-net state-space generator tool," in *2011 9th IEEE International Conference on Industrial Informatics*, July 2011, pp. 383–389. (citado nas págs. 19 e 20)

[23] (2019) Gres research group. IOPT Tools Editor. Consultado em 2019/04/24. [*Online*]. Disponível: http://gres.uninova.pt/IOPT-Tools/ssinitmarking.php (citado na pág. 20)

[24] F. Pereira, F. Moutinho, J. Ribeiro, e L. Gomes, "Web based IOPT Petri net editor with an extensible plugin architecture to support generic net operations," in *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, Oct 2012, pp. 6151–6156. (citado na pág. 20)

[25] GRES Research Group. IOPT Tools Simulator. Accessed on 2019-03-12. [*Online*]. Disponível: http://gres.uninova.pt/IOPT-Tools/simulator.php (citado na pág. 20)

[26] F. Pereira e L. Gomes, "Cloud based IOPT Petri net simulator to test and debug embedded system controllers," in *Technological Innovation for Cloud-Based Engineering Systems*, L. M. Camarinha-Matos, T. A. Baldissera, G. Di Orio, e F. Marques, Eds. Cham: Springer International Publishing, 2015, pp. 165–175. (citado na pág. 20)

[27] R. Hofestädt, "Advantages of Petri-Net modeling and simulation for biological networks," vol. 7, n. 4, pp. 221–229, 10 2017. (citado na pág. 20)

[28] L. Gomes, J. P. Barros, A. Costa, e R. Nunes, "The input-output place-transition petri net class and associated tools," in *2007 5th IEEE International Conference on Industrial Informatics*, vol. 1, June 2007, pp. 509–514. (citado na pág. 20)

[29] F. Moutinho, F. Pereira, e L. Gomes, "Analyzing security requirements as relationships among strategic actors," in *IOPT Tools User Manual*, 2014, pp. 4:1–4:50. [*Online*]. Disponível: http://gres.uninova.pt/iopt_usermanual.pdf (citado na pág. 21)

[30] F. Pereira e L. Gomes, "The iopt-flow framework pairing petri nets and data-flows for embedded controller development," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2016, pp. 4832–4837. (citado na pág. 25)

[31] ——, "The iopt-flow modeling framework applied to power electronics controllers," *IEEE Transactions on Industrial Electronics*, vol. 64, n. 3, pp. 2363–2372, March 2017. (citado na pág. 25)

[32] E. M. Thurner, "Tomspin-a tool for modelling with stochastic petri nets," in *Proceedings 6th International Workshop on Petri Nets and Performance Models*, Oct 1995, pp. 218–219. (citado na pág. 25)

[33] G. C. Gannod e S. Gupta, "An automated tool for analyzing petri nets using spin," in *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, Nov 2001, pp. 404–407. (citado na pág. 26)

[34] GRES Research Group. IOPT Tools Editor. Accessed on 2019-03-12. [*Online*]. Disponível: http://gres.uninova.pt/IOPT-Tools/pnml_editor.php (citado na pág. 41)

[35] K. Heurtefeux e F. Valois, "Is rssi a good choice for localization in wireless sensor network?" *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp. 732–739, 03 2012. (citado na pág. 42)

[36] L. Gomes, F. Moutinho, e F. Pereira, "IOPT-tools: A web based tool framework for embedded systems controller development using Petri nets," in *2013 23rd International Conference on Field programmable Logic and Applications*, Sep. 2013, pp. 1–1. (citado na pág. 45)

[37] (2019) Reyax tecnology corporation lda. Consultado em 2019/21/05. [*Online*]. Disponível: http://reyax.com/products/rylr896/ (citado na pág. 63)

[38] (2019) Adafruit solar lithium ion / polymer charger. Consultado em 2019/20/05. [*Online*]. Disponível: https://www.adafruit.com/product/390 (citado na pág. 67)

[39] P. Sérgio Rangel Garcia e J. Kleinschmidt, "Tecnologias emergentes de conectividade na IoT: Estudo de redes LPWAN," 09 2017. (citado na pág. 84)

[40] H. Linka, M. Rademacher, K. Jonas, e O. Aliu, "Path loss models for low-power wide-area networks: Experimental results using LoRa," 05 2018. (citado na pág. 85)