Universidade do Minho
Escola de Engenharia

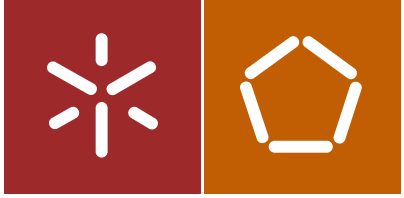
José Evaristo Lopes Lima

# Sistema de Comunicação e Controlo para Hidroponia

outubro de 2014

José Evaristo Lopes Lima Sistema de Comunicação e Controlo para Hidroponia

UMinho | 2014



Universidade do Minho
Escola de Engenharia

José Evaristo Lopes Lima

# Sistema de Comunicação e Controlo para Hidroponia

Dissertação de Mestrado
Engenharia Mecatrónica

Trabalho efetuado sob a orientação do
Professor Doutor António A. Caetano Monteiro
Professor Doutor Paulo José Guimarães Garrido

outubro de 2014

DECLARAÇÃO

**Nome:**
José Evaristo Lopes Lima

**Endereço eletrónico:** pg23183@alunos.uminho.pt

**Número do cartão de cidadão:** 13765927

**Título dissertação:**
Sistema de comunicações e controlo para hidroponia

**Orientadores:**
Professor António A. Caetano Monteiro

Professor Paulo José Guimarães Garrido

**Ano de conclusão:** 2014

Designação do Mestrado:

Engenharia Mecatrónica



Universidade do Minho, 31 de Outubro de 2014
Assinatura: _____

“Se não puderes destacar-te pelo talento, vence pelo esforço.”

Dave Weinbaum

**Agradecimentos**

Sem dúvida que o caminho foi longo para chegar até aqui. Ao longo de todos estes anos, amizades incríveis permitiram criar uma ponte sobre todos os obstáculos, pessoas que na retaguarda ajudaram, incentivaram e corrigiram. Quero agradecer, em primeiro lugar, aos meus pais por todos os valores transmitidos e pela forma como me educaram. Aos meus amigos e amigas, por toda a partilha de experiências, vivências, pela paciência e pela motivação que me transmitiram ao longo deste ano. Agradeço aos meus professores, atuais e de outros tempos por toda a partilha de conhecimentos. Por fim, agradeço a todas as pessoas que fazem parte da minha vida e me completam de alguma forma.

**Resumo**

A hidroponia é um método de cultivo onde as plantas crescem sem solo, que usa apenas uma solução baseada em nutrientes para alimentar as plantas. (Saaid, et al., 2013) As produções agrícolas são afetadas por diversos fenómenos meteorológicos, em particular, no caso da hidroponia para além desses fatores existem outros ligados à solução nutritiva que interferem no crescimento das plantas. Neste trabalho, é feita a monitorização da solução nutritiva e também da temperatura, humidade e luminosidade através de uma rede de sensores sem fios. O sistema é destinado ao interior de uma estufa e disponibiliza informação em tempo real, sobre as variáveis recolhidas, ao agricultor a partir de um computador. A metodologia utilizada neste trabalho consiste em fazer um levantamento das necessidades na cultura por hidroponia, seguidamente fazer um estudo de mercado sobre componentes eletrónicos – sensores e por fim desenvolver todo o sistema, software e hardware.

**Palavras-chave**

Redes de Sensores Sem Fios (RSSF), nós sensores, *ZigBee*, Hidroponia, temperatura, humidade, luminosidade, pH, electro-condutividade, CO2, painel solar.

**Abstract**

Hydroponic method of growing plants is based on a mineral nutrient solution (in water) without soil. There are several weather phenomena which affect the crop yields and, on this particular case of hydroponics crop, there are other factors that interfere with the plants' growth such as nutrient solution. (Saaid, et al., 2013) Therefore, in this work is made the nutrient solution monitoring and also the temperature, humidity and luminosity monitoring by a wireless sensor network. The system is conceived for the insides of a greenhouse and it provides to the farmer real time information on the collected variables from a computer. The methodology in this project consists in doing a statement of requirements in hydroponics culture, a research and market study about electronic components (like sensors) and, in the end, develop a system.

**Keyword's**

Wireless Sensor Networks, sensor mote, ZigBee, Hydroponics, temperature, humidity, luminosity, pH, electro-conductivity, CO2, solar panel.

# ÍNDICE

# ÍNDICE DE FIGURAS

# ÍNDICE DE GRÁFICOS

# ÍNDICE DE TABELAS

# 1. INTRODUÇÃO

A produção agrícola é muito dependente dos fatores climatéricos, uma estufa permite aos produtores criarem um clima propício ao desenvolvimento das plantas, protegendo as culturas dos fatores climáticos exteriores adversos. A hidroponia é uma técnica de cultivo agrícola sem solo, onde as raízes das plantas recebem uma solução nutritiva equilibrada que contém água e nutrientes essenciais ao desenvolvimento das plantas. Esta técnica de cultivo permite que as plantas cresçam num local onde o solo seja infértil.

As culturas hidropónicas são influenciadas por fatores ambientais como é o caso da temperatura, humidade, intensidade de luz e concentração de CO2, e também pela qualidade da solução nutritiva, pH, temperatura, oxigenação electro-condutividade.

A agricultura tecnológica é um meio que tem sido desenvolvido para apoiar as culturas agrícolas permitindo uma maior produtividade, qualidade, redução de custos, melhoria nos processos produtivos e atenuação dos efeitos provocados pelas condições atmosféricas adversas. As diversas investigações que se tem realizado na área da automação facilitaram o desenvolvimento das comunicações sem fios. A automação em conjunto com as Redes de Sensores Sem Fios substitui os sistemas manuais tradicionais e tem ganho bastante popularidade na indústria, nas casas e também no setor agrícola. (Baviskar, et al., 2014)

Apesar da grande evolução dos meios de apoio à agricultura, em concreto à hidroponia, os sistemas de monitorização e controlo apresentam um custo ainda muito elevado para os pequenos e médios agricultores. Com este trabalho pretende-se desenvolver um sistema de baixo custo, para estufas com culturas hidropónicas, que possibilite o controlo e monitorização dos fatores ambientais e da própria solução nutritiva. Com este sistema pretende-se aumentar a eficiência da produção agrícola facultando informações relevantes ao agricultor.

## 1.1 Objetivos

Este trabalho tem como objetivo criar um sistema de aquisição de dados através de sensores, no interior de uma estufa, para monitorizar variáveis físicas uteis na cultura hidropónica. Pretende-se para isso criar uma rede de sensores sem fios que abranja o espaço da cultura, recolha dados e disponibilize a informação num computador central tornando-a acessível ao agricultor e também criar um elemento para medições nos tanques que contém a solução nutritiva. A rede de sensores sem fios deverá ser autónoma energeticamente e aplicar os princípios de fiabilidade, robustez e acessibilidade para o utilizador. O custo do sistema desenvolvido deverá ser o menor possível, cumprindo no entanto com os requisitos. Será feito um estudo de mercado sobre componentes eletrónicos e sensores, assim como uma contextualização do problema na atual situação da hidroponia.

# 2. ESTADO DA ARTE

## 2.1 Redes de Sensores Sem Fios na Agricultura

Nos últimos anos desenvolveram-se diversos sistemas inteligentes aplicados na agricultura para a produção de alimentos com qualidade a baixo custo. O crescimento das plantas, para além de outros fatores, baseia-se nos níveis de humidade, temperatura, concentração de CO2 e intensidade de luz que o ambiente envolvente propicia. Estes fatores precisam de ser monitorizados e preservados para se criar um sistema autónomo. (Ijaz, et al., 2012)

Nas estufas mais recentes, para se monitorizar os parâmetros climáticos relevantes são necessários diversos pontos de medida em locais diferentes ao longo da estufa para que o sistema autónomo funcione corretamente. Um sistema de monitorização utilizando cabos torna-se vulnerável e tem um custo de instalação elevado. Para além disso, as alterações feitas depois de concluída a instalação são muito complicadas e difíceis de executar. (Ahonen, et al., 2008)

Uma Rede de Sensores Sem Fios (RSSF) é uma rede constituída por um certo número de pequenos nós sensores de baixo custo, de fácil desenvolvimento e baixo consumo de energia. Um nó sensor é constituído essencialmente por 4 unidades distintas, são elas a unidade de aquisição de dados, a unidade de memória e processamento, a unidade de comunicações e a unidade de energia. (Kalaivani & P, 2011)

As RSSF nas culturas em estufa podem ser consideradas como uma solução que emula o ecossistema para que as colheitas cresçam rapidamente. (Li, et al., 2011) A monitorização contínua das variáveis ambientais proporciona ao agricultor informações sobre os fatores que afetam o crescimento das plantas e também como poderá ter uma maior produtividade. Assim as RSSF hoje em dia começam a ser um instrumento importante para o agricultor.

Dependendo do tipo de produção ou de colheita o sistema tem de ser ajustado e adaptado. *Shining Li et al.* desenvolveram um sistema de aquisição de dados baseado numa RSSF para ser aplicado na agricultura de precisão, referem que o sistema tem de ser robusto para resistir aos fatores ambientais porque é aplicado em campo aberto. O

objetivo do sistema é monitorizar as variáveis ambientais através dos nós sensores, enviando a informação recolhida para um computador, telemóvel tornando-a acessível ao agricultor. (Li, et al., 2011)

Segundo *Jaypal Baviskar et al.* a aplicação de uma RSSF numa estufa apresenta um custo-benefício bom, o que torna rentável o desenvolvimento de um sistema de controlo. Desenvolveram assim um sistema para a monitorização em tempo real da temperatura e humidade e também do consumo de energia do sistema (nó sensor). A informação recolhida é depois apresentada num computador através de uma aplicação GUI. O sistema de comunicações é baseado na norma IEEE 802.15.4 com o *standard* ZigBee. (Baviskar, et al., 2014)

Os sistemas de monitorização inteligentes também podem ser aplicados no cultivo por hidroponia, *M.F. Saaid et al.* apresenta o desenvolvimento de um sistema monitorização de pH para a técnica de cultivo *Deep Water Culture* baseado num microcontrolador. (Saaid, et al., 2013)

*Lenord Melvix J.S.M e Sridevi C*. apresentam um sistema autónomo para a hidroponia que monitoriza e controla a solução nutritiva, nomeadamente os valores de pH e electro condutividade para a técnica de cultivo hidropónica *Ebb and Flow*. (J.S.M & C., 2014)

As RSSF potenciam o desenvolvimento dos diversos sistemas hidropónicos ajudando no controlo e otimização das culturas, uma área que está em desenvolvimento.

## 2.2 Hidroponia e Técnicas de Cultivo

A hidroponia não é um conceito recente, historiadores encontraram hieróglifos que descrevem a cultura de plantas em água há milhares de anos atrás, antes de Cristo. Em 1936 Dr. *W.F. Gericke* batizou com o nome hidroponia o cultivo de plantas comestíveis e ornamentais a partir de uma solução com água e nutrientes dissolvidos. (Roberto, 2005)

Atualmente o termo hidroponia é definido como uma técnica para cultivar plantas sem solo, onde as raízes recebem uma solução nutritiva com água e nutrientes essenciais ao seu desenvolvimento. Desta forma plantas terrestres podem ser cultivadas, sustentadas apenas por um meio inerte como exemplo a perlite, gravilha, argila

expandida ou fibra de coco. Um sistema hidropónico deve ser projetado de forma a cumprir com as necessidades básicas das plantas. Segundo *Keith Roberto* assenta sobre três princípios: fornecer uma fonte fresca e equilibrada de água e nutrientes às raízes da planta; manter um nível elevado de troca de gases entre as raízes e a solução nutritiva; proteger as raízes contra a desidratação e uma possível destruição da colheita caso haja uma falha na bomba que transporta água para a cultura ou uma falha de energia.

Os sistemas hidropónicos podem ser ativos ou passivos. Nos sistemas ativos existe um meio mecânico que transporta a solução nutritiva para as raízes das plantas, ao passo que no sistema passivo a solução nutritiva serve-se da ação capilar, a absorção e/ou gravidade para abastecer as raízes das plantas com nutrientes. As principais técnicas de cultivo em hidroponia são: *Nutrient Film Technique, Drip-Irrigation ou Micro-Irrigation, Aeroponics / Deep Water Culture, Flood & Drain (EBB and Flow), Water Flows*. (Saaid, et al., 2013)

### 2.2.1 Nutrient Film Technique - NFT

A técnica de cultivo com um fluxo contínuo de nutrientes, NFT (*Nutrient Film Technique*) é uma das mais populares e eficazes em hidroponia. Neste sistema existe um tanque que contém a solução nutritiva que é bombeada para as raízes das plantas que estão assentes em tubos, onde se desenvolvem. Estes tubos em forma de V invertido são feitos com um plástico especial (PVC), que permite a passagem contínua de nutrientes ao mesmo tempo que as raízes estão em contacto com a camada de ar. O perfil em V invertido para além de permitir uma boa oxigenação às raízes das plantas, por ser fechado mantém 100% de humidade evitando a desidratação. Por vezes as raízes alargam muito junto à base da calha o que provoca congestionamentos na circulação de água para as restantes plantas. O excesso de nutrientes, que não foram absorvidos, circula por gravidade até ao final do tubo, entrando depois num esgoto para serem reciclados.

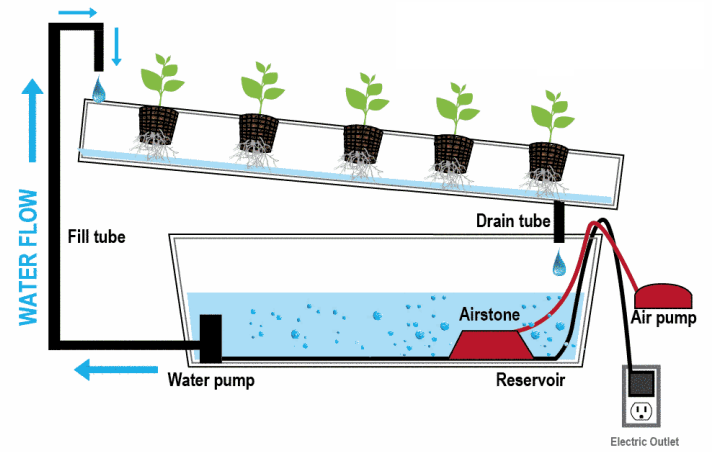



**Figura 1:**Esquema de funcionamento da cultura hidropónica por NFT.

Fonte: *http://hydroponicsinfomation.wordpress.com/2013/04/21/nfthydroponics/*

### 2.2.2 Water Flow – Fluxo de Água

A cultura baseada no fluxo de água pode ser de dois tipos, estática ou contínua. Na cultura estática, as plantas crescem num recipiente que contém uma solução nutritiva que geralmente é oxigenada. Este recipiente pode ser um frasco de vidro, um balde de plástico, um tanque ou uma banheira. Na tampa do reservatório é aberto um orifício onde é colocada a planta. É importante escolher um meio sólido com grande capacidade de retenção e uma boa qualidade de drenagem de água. Esta combinação permite que as raízes das plantas libertem CO2 e absorvam oxigénio conforme necessitam.

**Figura 2:** Esquema representativo da técnica de cultivo hidropónica com fluxo de água.

Na cultura com um fluxo contínuo de nutrientes (Figura 2 - direita), as raízes das plantas recebem continuamente a solução nutritiva. Neste sistema existe um tanque que contém a solução nutritiva que é bombeada para as raízes das plantas. Desta forma é

mais fácil obter-se um controlo de temperatura e nutrientes à medida que vão circulando pelas plantas, suprindo as suas necessidades.

### 2.2.3 Drip-Irrigation

*Drip method* ou conta-gotas é a solução que poupa mais água nas culturas hidropónicas. As plantas são colocadas num vaso com um determinado meio. Esse vaso é colocado dentro de um tabuleiro com outros vasos e plantas. Existe um tanque onde contém água e nutrientes, a solução nutritiva é bombeada para as plantas em certos períodos de tempo. Cada planta recebe individualmente a solução. Os excessos que não foram absorvidos retornam ao tanque para serem reaproveitados.

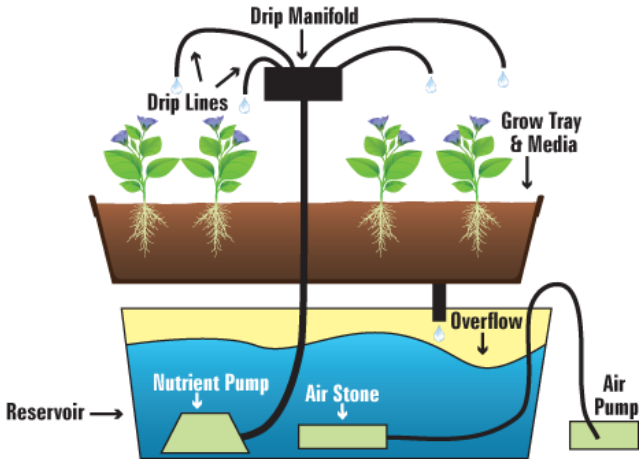



**Figura 3:** Esquema representativo da cultura hidropónica *Drip-Irrigation*.

Fonte: *http://sdhydroponics.com/resources/articles/gardening/how-to-grow-hydroponically-%E2%80%93-overview-of-grow-systems*

### 2.2.4 Aeroponics / Deep Water Culture

*Deep Water Culture* ou cultura de águas profundas baseia-se na suspensão das plantas sobre um tanque com água, onde as raízes ficam submersas numa solução rica em nutrientes e altamente oxigenada. Devido à grande abundância de oxigénio, esta técnica de cultivo permite que as plantas cresçam muito rapidamente. No caso do *Aeroponics* o sistema é semelhante, ou seja, as plantas, em particular as raízes, não possuem um meio físico de sustentação apenas são colocadas num cesto tipo de basquete.

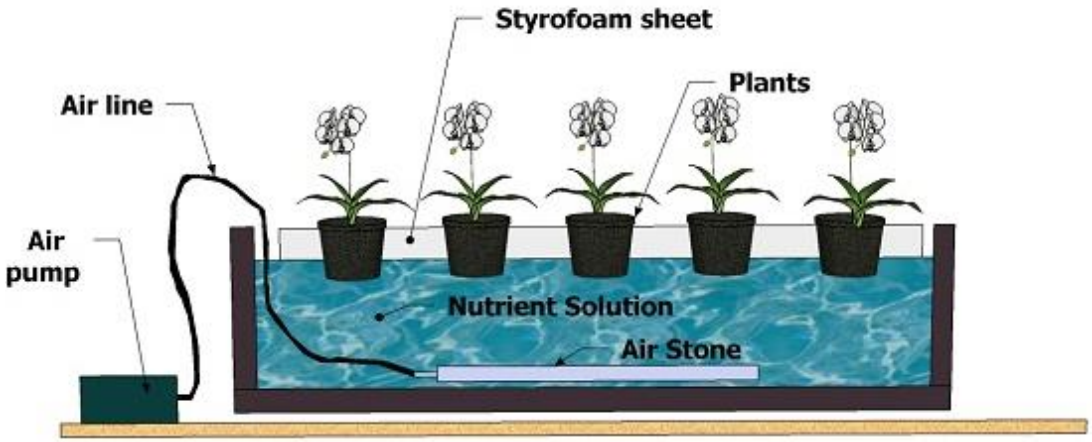



**Figura 4:** Esquema representativo da cultura hidropónica DWC.

Fonte: *http://www.medicalmarijuanaadvisor.net/hydroponic-systems/reminders-when-growing-your-weed-using-deep-water-culture-method/*

### 2.2.5 Flood & Drain (Ebb and Flow)

*Ebb and Flow* ou sistema de fluxo e refluxo é outra técnica de cultivo em hidroponia. As plantas são colocadas em vasos que normalmente contém argila e suportados num tabuleiro. No reservatório que contém água e nutrientes uma bomba é acionada em certos períodos tempo para bombear a solução nutritiva para as raízes das plantas até ficarem submersas. Após isso, a solução é drenada para o reservatório. Isto permite que haja uma troca regular de oxigénio e nutrientes nas raízes das plantas.

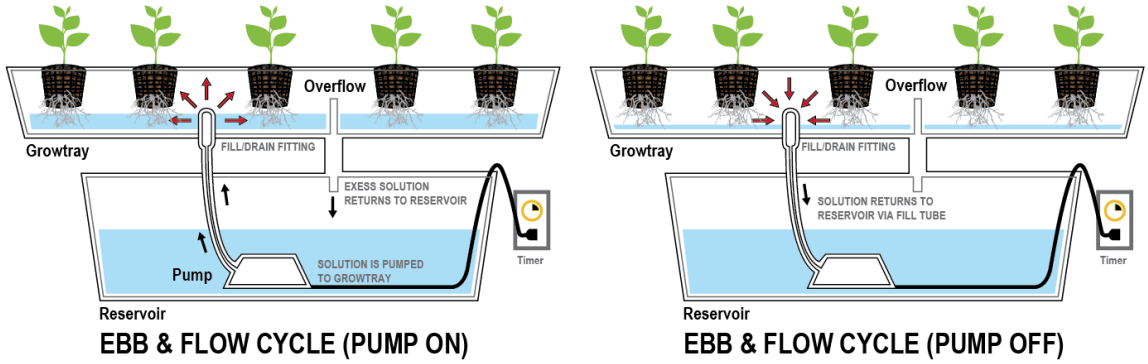



**Figura 5:** Esquema representativo da técnica de cultivo hidropónica *Ebb and Flow* - Fluxo e Refluxo.

Fonte: *http://www.medicalmarijuanablog.com/benefits/grow-box-growing.html*

## 2.3 Hidroponia em Portugal, Perspetiva e Evolução

Numa entrevista ao jornal Expresso, *Wolfgang Kemper*, da *Filkemp* (empresa líder mundial no sector das linhas de pesca), viveu toda a sua vida profissional no universo químico do gigante alemão *Hoescht*, mas a sua grande paixão sempre foi a

agricultura. Aos 83 anos admite que a solução para o êxito do sector primário português está ao alcance de todos. A solução diz, está na "hidroponia", que é o cultivo de plantas sem solo, apenas com nutrientes líquidos. Assim, que culturas terão bons resultados? "Façam como um amigo meu que cultiva frutos silvestres, amoras e morangos apenas em um hectare e está feliz da vida", diz. O segredo deste mercado é verdadeiramente milionário. "Só a Alemanha importa 70 mil milhões de euros anuais em produtos agrícolas e Portugal apenas pesa 0.2% nessas compras", remata. Querem melhor? Concluiu assim a breve reflexão. (Kemper, 2013)

### 2.3.1 História do grupo Hubel

O grupo Hubel e a Fertirrega são empresas que foram criadas por Humberto Teixeira, engenheiro eletrotécnico de profissão, no início dos anos 90 com objetivos muito específicos na agricultura tecnológica. Nesta altura a hidroponia em Portugal era um mundo para descobrir. Em 1992, *Thierry Roussel* propôs um projeto à *Fertirrega* para criação de uma infraestrutura em Odemira para a agricultura com sistemas de nutrição vegetal. Pouco depois foi lançado o desafio da hidroponia, nessa altura o conhecimento sobre estes sistemas era muito pouco. O autoconhecimento aliado a parcerias com grupos desenvolvidos tecnologicamente neste setor foi a forma encontrada para ultrapassar o desconhecido. Após 5 anos de testes Humberto Teixeira relata que se consideravam aptos para produzir e criaram assim uma plantação de meio hectare de meloa em substrato (semi-hidroponia). Mais tarde, no ano 2000, por causa das dificuldades em escoar o produto formou-se a Organização de Produtores Madre Fruta numa tentativa de mudar e melhorar. Chegou-se assim aos frutos vermelhos, em particular o morango que já teria história no Algarve de ser produzido e exportado. Entretanto, no ano de 2004 o grupo *Hubel* instalou o sistema de rega para a empresa *Driscoll's* ao mesmo tempo que tomava conhecimento que a produção deles seria de framboesa em solo. Decidiram experimentar a framboesa em hidroponia e tiveram uma agradável surpresa pois é mais rentável que o morango. Humberto Teixeira pretende aumentar a produção para os 40 hectares em 5 anos, tendo 1 porção de morangos para 3 de framboesa. Salienta o facto de o clima ser favorável para a produção destes frutos no local onde se encontra, pois um desvio de 10 ou 15 quilómetros é o bastante para perder qualidade e produção. A *Hubel* representa uma das maiores cotas de mercados nos projetos de assistência e chave na mão em hidroponia em Portugal. A título exemplar,

dos 14 mil hectares de tomate de indústria que se fazem em Portugal, a grande parte serve-se da assessoria dos técnicos da *Hubel*. Apresenta ainda um modelo estilo gótico de estufas, com um preço de 15 euros o metro quadrado, com tendência a baixar.

### 2.3.2 Visita à Hortivolátil – Produção de morangos em Hidroponia

A Hortivolátil é uma empresa situada em Vila Nova de Famalicão que se dedica ao cultivo de morango por hidroponia, seguindo elevados padrões de produção, procurando fornecer um produto de qualidade superior, aos mercados onde se insere. Em contactos com a empresa foi-nos propiciada uma visita às suas instalações, nomeadamente às estufas onde é produzido e colhido o morango que é também onde está o sistema de hidropónico. A estrutura de produção baseia-se no sistema NGS, ou seja, a solução nutritiva circula no interior de mangas de polietileno. A distribuição das mangas faz-se de tal forma que a solução nutritiva, depois de percorrer as mesmas, é descarregada através de um tubo de drenagem permitindo a recirculação da água e dos nutrientes. Trata-se de um sistema hidropónico suspenso, que trabalha em circuito fechado, otimizando a água e os fertilizantes destinados ao cultivo através da solução nutritiva. O desenho do sistema origina um fluxo em cascata, que permite que as raízes se estendam sem restrições, conseguindo-se uma maior oxigenação do sistema radicular.

Em jeito de resumo pode-se analisar abaixo os diversos constituintes da cultura hidropónica, quais as variáveis controladas, as que não são controladas, isto para o caso do ambiente da estufa e também do tanque que contém a solução nutritiva.

- Parâmetros controlados
    - Interior da estufa – medidos no topo superior e no topo inferior
        - Temperatura
        - Humidade
    - Exterior da estufa
        - Velocidade do vento
        - Temperatura
        - Luz
- Solução nutritiva
    - pH
    - Electro Condutividade (1.6 referência)

A variação do valor das variáveis acima indica se faltam nutrientes na solução. No entanto não é possível determinar qual o nutriente específico em falta. Existem dois grandes tipos de nutrientes, macronutrientes (Nitrogênio, Fósforo, Potássio, Cálcio, Magnésio e Enxofre) e micronutrientes (Cloro, Boro, Ferro, Cobre, Manganês, Molibdênio e Zinco).

- Plantas - Morangueiro
    - São naturais, não necessitam de preparação para este sistema.
    - Utilizam fibra de coco para firmar as raízes. A fibra absorve a solução nutritiva e assim retém os nutrientes junto das raízes durante mais tempo para serem absorvidos.
    - A nutrição das plantas foi melhorada com a experiência, mas ainda existem opções por descobrir e melhorar a produção, salientam os produtores.
- Estufa
    - No interior da estufa as calhas que contém as plantas são movíveis até a 1,20m de altura facilitando assim a apanha dos morangos.
    - No verão pintam o exterior das estufas com cal para reduzir a temperatura e incidência dos raios solares.
    - Para o próximo inverno irão colocar aquecimento por biomassa para criar condições de cultivo durante mais tempo.
- Produção
    - Neste momento estão com uma produção anual de 130 toneladas, esperam atingir este ano as 150 toneladas. Para este efeito o aquecimento será fundamental.
    - O quilograma de morango estava a ser vendido a 1,60€
    - O pico da produção situa-se entre março e julho para o morango.

Um dos grandes entraves ao aumento da tecnologia nesta empresa de produção de morangos tem a ver com os sistemas apresentados pelas empresas serem muito caros e não apresentarem um custo-benefício que compense o investimento. Segundo um dos gerentes da empresa, existe um monopólio na venda dos sistemas hidropónicos e fala também da questão das patentes que eleva os custos na compra de equipamento, nomeadamente as calhas NGS.

# 3. SISTEMA DE COMUNICAÇÃO E CONTROLO HIDROPONIA

## 3.1 Motivação

A cultura em estufas assim como a cultura por hidroponia são formas de cultivo desenvolvidas principalmente por razões de ordem económica e financeira. Estas formas de cultivo especializadas permitem um aumento da produção ao mesmo tempo que a tornam mais rentável. No caso da hidroponia num local onde os solos são inférteis ou até mesmo onde não existe solo puro, como por exemplo uma varanda de uma casa, é possível fazer-se uma cultura.

Num ambiente controlado, caso de uma estufa com cultivo por hidroponia ou não, é possível cultivar espécies agrícolas fora da sua época produtiva normal. Isto pressupõe, para além de uma valorização económica grande, que o ambiente da estufa seja controlado e propício ao seu desenvolvimento.

Para que o controlo do clima dentro de uma estufa seja eficaz do ponto de vista fisiológico, é necessário saber quais os elementos que caracterizam as condições ideias para o cultivo de uma espécie agrícola. O controlo destes fatores não quer dizer que haja um lucro iminente, por vezes é necessário despender energia sem que isso aumente a produtividade ou dê rendimento biológico.

A definição das condições ideais para o cultivo de uma espécie ao longo do seu ciclo de vida, que vai desde a plantação até à recolha do fruto, é praticamente impossível, no entanto existem fatores climatéricos que devem situar-se dentro de certos limites. Neste caso, na hidroponia é necessário atender aos aspetos climatéricos e também aos aspetos da solução nutritiva.

Recorrendo à utilização de sensores, atuadores, microcontroladores aliados a um *software* que possibilite a gestão, é possível controlar estes fatores a que se propõe neste documento.

## 3.2 Fatores Climáticos Relevantes

Os sistemas ambientais, em particular os sistemas biológicos associados à produção agrícola são extremamente complexos, assim como qualquer outro sistema na natureza. Na verdade, a sua complexidade é tão grande que descrever e prever o seu

comportamento matematicamente e fisicamente é quase impossível. Desta forma, para gerir o sistema deve-se simplifica-lo e descrevê-lo como um conjunto de valores mensuráveis que são conhecidos por terem um maior impacto sobre a produção agrícola. (Zazueta, et al., 1991)

Na cultura por hidroponia a produção depende dos fatores ambientais da estufa e dos fatores da solução nutritiva. No caso do ambiente da estufa, os fatores mais importantes são a temperatura, a humidade, a intensidade de luz e a concentração de CO2. Na solução nutritiva os fatores mais importantes são o oxigênio, o pH e a condutividade elétrica. (Roberto, 2005)

### 3.2.1 Temperatura

A temperatura é uma das variáveis climáticas que envolve as plantas mais importantes a monitorizar. Ela afeta todos os processos de crescimentos ou funções metabólicas das plantas, tais como a fotossíntese, respiração, transpiração, queda de sementes, germinação das sementes, síntese de proteínas e a translocação (movimento da água e solutos no interior da planta). A altas temperaturas a translocação é mais rápida o que leva a um amadurecimento precoce. (Bareja, 2011)

A temperatura ótima para a cultura em estufa situa-se entre os 15°C e os 30°C dependendo do produto agrícola cultivado. (Santos, 2008) Temperaturas elevadas provocam danos permanentes nas plantas, queimando-as. Por outro lado valores muito baixos de temperatura, sensivelmente abaixo dos 5°C provocam deformações estruturais e limitações no crescimento. Quando a temperatura é inferior ao ponto de congelação da água, o processo de solidificação em células vivas provoca a rutura das paredes das células.

No inverno, as culturas no interior da estufa podem ser protegidas utilizando aquecimento externo para manter a temperatura em valores adequados. No verão, quando as temperaturas são muito elevadas, para arrefecer a estufa recorre-se a ventilação natural ou artificial, sistemas de arrefecimento (ar condicionado), sombreamento e também caiar o exterior da estufa.

### 3.2.2 Humidade

A quantidade de vapor água que o ar pode conter depende da temperatura. O ar quente consegue reter mais água que o ar frio, a cada 10°C de descida de temperatura existe uma redução para metade dessa quantidade. (Bareja, 2011)

A humidade relativa é a quantidade de vapor de água, expressa em proporção (percentagem), que o ar pode conter a uma certa temperatura. Por exemplo, o valor de humidade relativa de 60% a 27°C significa que cada quilograma de ar contém 60% da quantidade máxima de água.

A humidade relativa do ar afeta a abertura e o fecho dos estomas que regulam a perda de água nas plantas por transpiração, e também na fotossíntese. Os valores ótimos para o crescimento das plantas situam-se entre os 70% e os 80%. Valores elevados de humidade relativa levam ao aparecimento de fungos e limitam os mecanismos de regulação de temperatura das plantas por transpiração. Por outro lado, quando o ar é muito seco revela uma baixa humidade relativa provocando um atraso no crescimento das plantas. (Santos, 2008)

### 3.2.3 Radiação Solar

A luz visível é apenas uma parte da radiação solar ou do espectro eletromagnético. Trata-se de uma forma de energia cinética que vem do sol em partículas minúsculas chamadas fotões ou quanta, que viajam em forma de onda. A luz é um fator essencial na produção de clorofila e na fotossíntese das plantas. É neste processo que as plantas fabricam o seu próprio alimento em forma de açúcar (hidratos de carbono). (Bareja, 2011)

Existem três características próprias deste fator climático que afetam o crescimento e o desenvolvimento das plantas, são elas a qualidade da luz, intensidade da luz e a duração do dia ou fotoperíodo. A qualidade da luz tem interferência devido ao comprimento de onda, a intensidade dita o grau de brilho que a planta recebe e por fim o fotoperíodo refere-se à duração do período do dia em relação ao período da noite.

A radiação solar total recebida à superfície da terra pode ser dividida em radiação direta, aquela originada pela posição momentânea do sol, e radiação difusa, aquela que é dispersada pela atmosfera e pelas nuvens. O fluxo energético recebido no planeta tem um comprimento de onda entre os 300nm e os 2500nm, no entanto, para

as plantas os comprimentos de onda relevantes situam-se entre os $400nm$ e os $700nm$, a chamada região da luz visível.

Uma cultura deve receber em média $1{,}4\mathrm{MJ/m^2}$, valores inferiores estagnam o desenvolvimento das plantas. Na instalação de uma estufa ou área para o cultivo em hidroponia deve ter-se em conta o nível de radiação que o local oferece, o seja, se não existem sombras provocadas por árvores ou outros obstáculos. Em locais onde a radiação é muito intensa poderá recorrer-se a ecrãs de sombreamento. (Santos, 2008)

### 3.2.4 Dióxido de Carbono – CO2

O ar é uma mistura de gases na atmosfera. O dióxido de carbono (CO2) é um desses gases que em conjunto com a água, a luz e o calor, é assimilado pelas plantas por forma a sintetizarem matéria orgânica. A concentração de CO2 no ar livre é de aproximadamente $330\mathrm{ppm}$ (partes por milhão). Num local fechado onde existe uma cultura, como é o caso das estufas, este valor altera-se ao longo do dia. Durante a noite pode atingir as $500\mathrm{ppm}$ e durante o dia devido à fotossíntese as $200$ ppm.

O aumento da concentração de CO2 no ar, em conjunto com condições favoráveis de temperatura e luminosidade leva a um maior desenvolvimento das plantas. Em determinadas alturas existe a inserção de CO2 no interior das estufas, até um valor de $2000\mathrm{ppm}$ para aumentar e velocidade do crescimento das plantas, ou faze-las amadurecer mais rapidamente.

### 3.2.5 Outros Fatores

Para além dos fatores climatéricos descritos anteriormente, dependendo da situação, poderá ser necessário monitorizar outros, como é o caso da velocidade do vento ou a precipitação. No caso de uma estufa, estes fatores são medidos no exterior para prevenir essencialmente danos na própria estrutura. Um sistema de controlo com estas variáveis poderá programar a abertura e fecho das janelas da estufa reduzindo o efeito do vento.

## 3.3 Solução Nutritiva

### 3.3.1 pH

O pH representa o potencial de hidrogénio presente numa solução aquosa indicando se ela é ácida, alcalina ou neutra. Para que haja um bom crescimento e desenvolvimento das plantas, a concentração de nutrientes e o pH devem ser equilibrados de tal forma que as necessidades das plantas sejam asseguradas com o que necessitam e no momento em que necessitam. (Roberto, 2005)

O valor do pH da solução nutritiva que circula pelas raízes das plantas é um fator importante que afeta a captação de muitos dos nutrientes. Os valores ótimos para diversas culturas situam-se entre os 4.5 pH e os 6.5pH. Se as raízes das plantas entrarem em contacto, mesmo que seja por alguns segundos, com um valor de pH entre 2 a 3 são causados danos nas raízes irrecuperáveis. (Saaid, et al., 2013)

Monitorizando o pH no tanque que contém a solução nutritiva num sistema hidropónico, é possível detetar quando existe a falta de algum nutriente evitando assim que as plantas sejam afetadas.

### 3.3.2 Electro-condutividade

A condutividade elétrica trata-se da medida da capacidade que uma solução tem para conduzir corrente. Nos metais a corrente flui através dos eletrões, no caso dos líquidos são os iões com cargas opostas que se formam quando é dissolvido um sólido que fazem circular a corrente. (Controls, 2005)

Num sistema hidropónico, à medida que a solução nutritiva circula pelas raízes das plantas existem trocas e alguns nutrientes são absorvidos. Com o passar do tempo a solução nutritiva perde nutrientes e muda os valores de concentração. A monitorização da quantidade de TDS ou PPM, Total de Sólidos Dissolvidos ou Partes Por Milhão, é uma forma fácil de controlar os níveis de nutrientes que a solução tem. A condutividade elétrica permite saber o valor, de forma indireta, da quantidade de sólidos ou nutrientes existentes na solução. Contudo não é possível saber independentemente a quantidade de cada sólido.

## 3.4 Análise dos Requisitos

### 3.4.1 Listagem de requisitos em controlo / monitorização

O sistema de controlo e monitorização em primeiro lugar deverá ser capaz de adquirir valores sobre as variáveis físicas descritas acima. Neste caso são:

- Ambiente da estufa - Monitorização
    - Temperatura
    - Humidade
    - Concentração de CO2
    - Radiação solar
- Tanque com Solução Nutritiva - Monitorização
    - pH
    - Electro Condutividade
    - Temperatura
    - Nível de água

### 3.4.2 Listagem de requisitos de armazenamento de dados

Posteriormente, com os valores adquiridos deverá fazer-se um registo numa base de dados num computador central e mostrar a evolução ao longo do tempo das variáveis escolhidas pelo sistema de monitorização.

### 3.4.3 Listagem de requisitos de comunicação

O sistema de monitorização deverá possuir também um sistema de avisos e alertas climatéricos. O agricultor poderá aceder através da internet a todo o sistema de monitorização e verificar o seu estado. Aplicação de uma Rede de Sensores Sem Fios para a aquisição de dados.

### 3.4.4 Análise financeira

A instalação de todo o sistema para cultivar plantas por hidroponia custa no mercado 40 mil euros, valor do primeiro financiamento do jovem agricultor. O sistema de monitorização deverá ser de baixo custo, autónomo e facilmente implementável.

# 4. LEVANTAMENTO DE COMPONENTES ELETRÓNICOS

Neste capítulo é feito um levantamento dos sensores úteis que podem auxiliar a cultura em hidroponia medindo variáveis físicas tais como a temperatura, humidade, luminosidade, condutividade elétrica, pH, entre outros. Apresentam-se as principais características, modo de funcionamento, fiabilidade, implementação e custo. No final do capítulo serão descritos os componentes que também integram o nó sensor, é o caso do módulo de processamento e memória, módulo de comunicações e o módulo de energia.

## 4.1 Sensores de Temperatura

A temperatura ambiente é um dos fatores que tem grande influência nos processos fisiológicos das plantas, no seu crescimento e desenvolvimento. Na cultura por hidroponia assim como na cultura tradicional, a monitorização e controlo da temperatura ambiente previne danos nas plantas ou até mesmo a sua destruição.

A medição de temperatura usualmente é feita a partir de resistências dependentes da temperatura (RTDs), termopares, termístores e sensores monolíticos. Cada aplicação tem uma necessidade específica que vai ao encontro das características dos sensores, tais como a linearidade, precisão, gama, custo e implementação.

### 4.1.1 Termopar

Um termopar consiste na junção de dois metais diferentes. Quando a junção dos metais é aquecida gera-se aos terminais uma tensão termiónica proporcional à temperatura. Chama-se a isto efeito *Seeback* e a tensão gerada é a tensão de *Seeback* ($e_{AB}$).

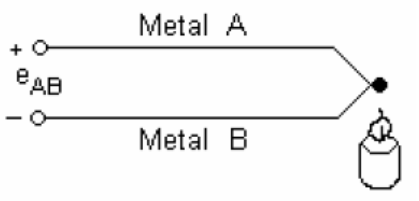



**Figura 6:** Tensão *Seeback* ($e_{AB}$).

Para se medir a temperatura na junção dos metais é necessário uma temperatura de referência, assim conhecendo a temperatura na junção $J_1$ ($T_1 = \text{ref}$), que

normalmente é a temperatura de fusão do gelo, a corrente térmica pode ser calibrada em termos de temperatura na junção quente ($J_2$) $T_2$.

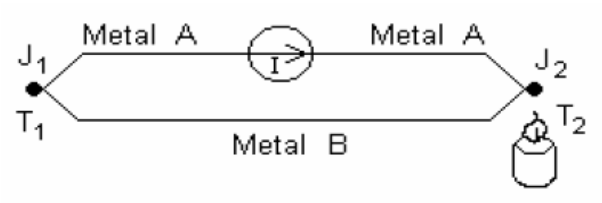



**Figura 7:** Efeito de *Seeback*.

O processo de medição da tensão não é direto, este fato deve-se ao mesmo princípio com que é feita medição de temperatura, ou seja, à junção de dois metais diferentes originar um novo circuito termoelétrico. Assim para se colocar um voltímetro e medir a tensão gerada pelo aquecimento na junção quente, em relação à junção fria (de referência) utiliza-se um bloco isotérmico que anula as f.e.m. originadas pela nova junção de metais.

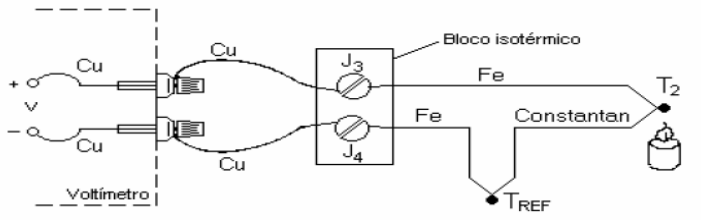



**Figura 8:** Cancelamento do erro devido à ligação de um voltímetro.

A Tabela 1 apresenta as principais composições de metais que originam termopares e as suas características.

| Tipo | Composição | Gama de temperatura | Tensão de saída | Precisão |
|---|---|---|---|---|
| K | Níquel + Cromo<br>Níquel + Alumínio | -270°C a 1260°C | -6.458 a 54.886mV | ± 2.2°C ou ± 0.75% |
| J | Ferro + Constantan | -210°C a 760°C | -8.095 a 69.553mV | ± 2.2°C ou ± 0.75% |
| T | Cobre + Constantan | -270°C a 370°C | -6.258 a 20.872mV | ± 1.0°C ou ± 0.75% |
| E | Cromo + Níquel<br>Constantan | -270°C a 870°C | -9.835 a 76.373mV | ± 1.7°C ou ± 0.5% |
| S | Platina<br>Platina + 10% Ródio | -50°C a 1480°C | -0.236 a 18.693mV | ± 1.5°C ou ± 0.25% |
| B | Platina + 30% Ródio<br>Platina + 6% Ródio | 0°C a 1700°C | 0 a 13.82mV | ± 0.5% |
| R | Platina + 13% Ródio | -50°C a 1480°C | -6.258 a 20.872mV | ± 1.5°C ou ± 0.25% |

**Tabela 1:** Comparação de alguns termopares de referência.

Fonte: *http://www.thermocoupleinfo.com/*

Um dos grandes inconvenientes na utilização de termopares é a necessidade de uma temperatura de referência para efetuar qualquer medição. Toda a incerteza associada ao valor da temperatura de referência ($T_{Ref}$) irá provocar erros de leitura ($T_x$) com a mesma incerteza. Este tipo de sensor tem como vantagem o baixo custo - 5€ (Farnell, s.d.) ou 4,768€ (RS, s.d.), a sua robustez e uma ampla gama de temperatura e uma resposta rápida. Não é linear e tem baixa precisão.

### 4.1.2 Sensores Resistivos

A resistência elétrica de vários materiais altera-se de modo reprodutível com a temperatura. Estes materiais dividem-se em duas categorias: Materiais condutores (metais) e materiais semicondutores. Os materiais condutores são chamados detetores de temperatura resistivos – RTD, enquanto os materiais semicondutores são denominados termístores. (Santos, 2008)

Neste ponto aborda-se os detetores de temperatura resistivos (RTD). Este tipo de sensores de temperatura apoia-se no princípio da proporcionalidade que existe entre a variação da temperatura com o valor da resistência, aplicada aos metais. Os metais que formam este tipo de sensores devem possuir características estáveis para que a sua resistência à temperatura de referência não se altere após vários ciclos de aquecimento e arrefecimento. Além disso, a sua resistência específica deve permitir o fabrico de sensores de tamanho prático. A Tabela 2 mostra alguns dos metais ou ligas metálicas utilizados na construção destes sensores.

| Material | Gama de temperatura | Linearidade | Coeficiente de temperatura |
|---|---|---|---|
| Platina | -200°C a 850°C | Alta | 0.39 %/°C |
| Níquel | -80°C a 320°C | Baixa | 0.67 %/°C |
| Cobre | -200°C a 260°C | Alta | 0.38 %/°C |
| Tungsténio | -70°C a 2700°C | Média | 0.45 %/°C |
| Níquel/Ferro | -200°C a 260°C | Média | 0.46 %/°C |

**Tabela 2:** Coeficiente de temperatura de alguns metais.

O material que possui as melhores características como sensor é a platina, no estado puro consegue ter ótimas condições de repetibilidade e também uma grande precisão, servindo até como referência para calibração. Nos extremos possui uma não linearidade que pode ser corrigida consultando tabelas. Estes dispositivos necessitam de energia para serem alimentados, por vezes a corrente aplicada pelos circuitos de

acondicionamento provoca um auto aquecimento do metal e consequentemente erros na leitura.

Os RTD têm como principais vantagens a sua precisão, linearidade, tamanho, estabilidade ao longo do tempo. Em termos de limitações ou desvantagem está o seu custo – PT100 de cobre 20€ (RS, s.d.) ou PT100 de teflon 26€ (Farnell Components SL, 2014), fonte de alimentação, compensação dos fios de ligação/erros de auto aquecimento.

### 4.1.3 Termístor

O termístor é outro tipo de sensor resistivo com uma sensibilidade 10 vezes superior à das resistências metálicas. Por outro lado, o seu coeficiente de temperatura é geralmente negativo e fortemente dependente da temperatura. Existem dois grandes tipos de termístores, os NTC com um elevado coeficiente de temperatura negativo, ou seja a sua resistência diminui com o aumento da temperatura e os PTC com um coeficiente de temperatura positivo, chamados também de termístores de comutação. Neste último, à medida que a temperatura aumenta a resistência mantém-se até ao ponto de comutação ou ponto de *Curie*. A temperatura de comutação poderá estar entre os -20°C e os +125°C. Usualmente os termístores PTC são utilizados como termostatos para regulação de temperatura assim como em dispositivos de segurança térmica.

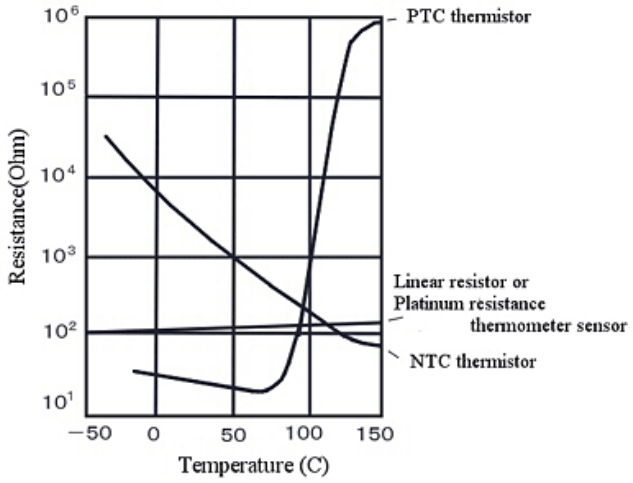



**Figura 9:** Gráfico com dependência com a temperatura de termístores NTC e PTC.

### 4.1.4 Semicondutores

Um sensor de temperatura é facilmente construído recorrendo à característica tensão-corrente da junção *pn* nos semicondutores. Assim, quando num transístor bipolar

ou num díodo é percorrida uma corrente constante na junção p-n, a tensão resultante torna-se medida de temperatura.

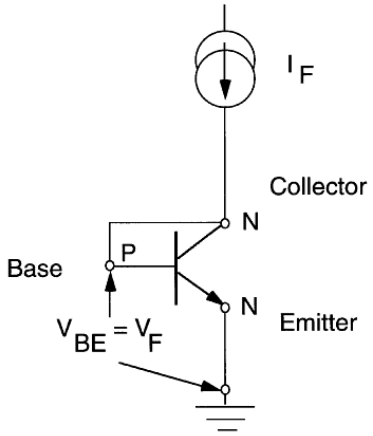



**Figura 10:** Configuração de um transístor bipolar como sensor de temperatura.

Uma das principais características dos sensores semicondutores é a sua grande linearidade que permite determinar a sensibilidade conhecendo apenas dois pontos de medida. Quando comparado com um termopar ou um RTD a gama de medição é larga mas comparativamente torna-se pequena. Por outro lado a relação voltagem-temperatura e muito mais linear. Por fim o circuito de acondicionamento é muito simples, a sua saída pode ligar-se facilmente a um microcontrolador e obter valores de tensão uteis para determinadas aplicações. Para além a grande linearidade, uma elevada sensibilidade e facilidade de implementação, os sensores semicondutores também possuem um custo reduzido – LM35DZ 1,45€ (RS, s.d.) ou MCP9701 0,21€ (Farnell Components SL, 2014) e existe uma enorme variedade de modelos. Em termos de limitações existe a baixa gama de medição, requer excitação (corrente) e está propício a erros de auto aquecimento.

Através da análise dos tipos de sensores descritos neste ponto, tendo em conta os fatores como custo, gama de medição, facilidade de implementação, os sensores semicondutores são a melhor opção para medir temperatura na estufa.

## 4.2 Sensores de Humidade

A par da temperatura, a humidade e, mais em concreto a humidade relativa é também um dos fatores climatéricos que influência os processos fisiológicos das plantas, devendo por isso ser monitorizada e controlada. A humidade afeta a taxa de

transpiração das plantas, quando a humidade relativa é baixa (cerca de 20%) provoca desidratação nas plantas assim como quando é elevada promove o aparecimento doenças por fungos e até apodrecimento.

Os sensores de humidade eletrónicos podem ser divididos em duas categorias, sensores com efeito capacitivo e sensores com efeito resistivo. Nesta secção aborda-se os dois modelos, que medem a humidade relativa do ambiente em que se encontram, analisando a melhor solução para integrar o módulo sensor.

### 4.2.1 Capacitivo

Os sensores de humidade capacitivos são formados por um material dielétrico higroscópico colocado no seio de elétrodos metalizados formando assim um condensador. Em funcionamento normal, o vapor de água (humidade) presente no dielétrico ficará em equilíbrio com o ar que o rodeia, ou seja com a mesma quantidade de água, desta forma a impedância elétrica do condensador indicará a humidade do ar.

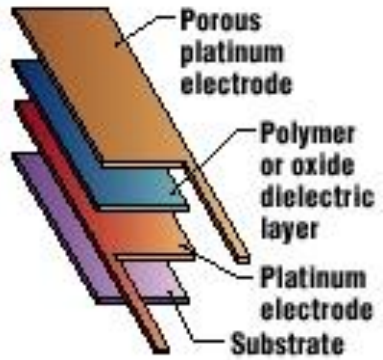



**Figura 11:** Esquema de um sensor RH capacitivo.

Fonte: *http://www.engineersgarage.com/articles/humidity-sensor*

O sensor capacitivo de humidade relativa HCH-1000-séries da *Honeywell* apresenta uma variação linear com sensibilidade de $0.6\,pF/\%RH$, uma capacidade de 330pF e gama testada de 10%RH a 98%RH. O preço deste sensor ronda os 10,13€ (Farnell Components SL, 2014).

### 4.2.2 Resistivo

Os sensores de humidade do tipo resistivo baseiam-se na variação da resistência elétrica entre dois elétrodos para se obter valores da humidade relativa. Os elétrodos constituídos por um metal nobre são envoltos num substrato dielétrico, como por exemplo o gesso. O vapor de água é absorvido pelo substrato do sensor, isso provoca

um dissociação dos grupos funcionais iónicos e consequentemente um aumento da condutividade elétrica.

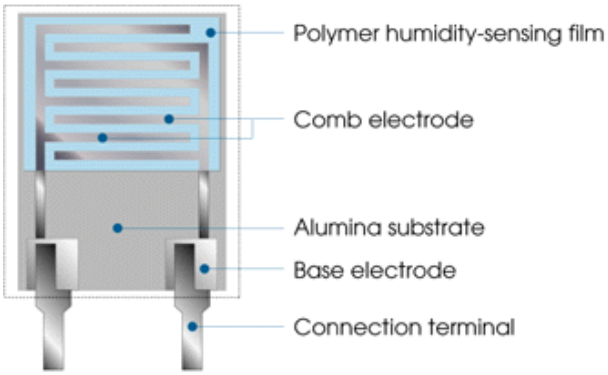



**Figura 12:** Esquema de um sensor RH resistivo.

O sensor HIH-4010 da *Honeywell* apresenta uma variação praticamente linear, numa temperatura funcional de entre os 0°C e os 50°C, uma sensibilidade de 3,5% e um preço 18,83€ (Farnell Components SL, 2014). Em comparação com os sensores capacitivos, os sensores resistivos apresentam uma saída em voltagem, assim não necessitam de um circuito de acondicionamento e podem ser ligados diretamente a um microcontrolador.

Tendo em conta as características funcionais, apesar do preço ser mais elevado o sensor de humidade resistivo não necessita de um circuito de acondicionamento e adequa-se às necessidades do módulo sensor sendo a escolha realizada.

## 4.3 Sensores de Luminosidade

A radiação solar a que as plantas estão sujeitas influência diretamente o seu processo de desenvolvimento pelo que deve ser monitorizada. Níveis baixos de radiação levam a um fraco desenvolvimento e em contrapartida níveis elevados levam a queimaduras. (Santos, 2008)

A banda de radiação ótica de acordo com o *standard* DIN5130, parte 7, está divida em 3 sub-bandas, a ultravioleta (UV), a radiação visível (luz normal) e os infravermelhos (IR). As bandas UV e IR encontram-se ainda subdivididas em três subgrupos A,B e C e a gama visível nas cores principais. Para as plantas importa medir a radiação que contribui para a fotossíntese, que neste caso situa-se na faixa dos 400nm aos 700nm, ou seja, na gama da luz visível.


**Figura 13:** Subdivisão do espetro da radiação ótica.

A sensibilidade espetral ou resposta relativa é uma das características mais importantes que se deve ter em conta na escolha de um sensor de radiação solar. A sensibilidade está dividida em dois tipos. A sensibilidade espetral S(λ), que é a sensibilidade do transdutor face ao comprimento de onda supondo o raio incidente monocromático e a sensibilidade total St, que é a sensibilidade do transdutor que recebe um sinal não monocromático.

As grandezas referentes à radiação solar dividem-se em grandezas fotométricas que são relativas à radiação visível e grandezas energéticas relativas ao conteúdo energético da radiação. Importa saber quais as unidades utilizadas no sistema S.I. e a sua definição tendo em conta as gradezas referidas acima.

A **potência radiada** (Φ) e **fluxo luminoso** (Φ**v**) descrevem a potência total radiada para o espaço por uma fonte luminosa. Os valores apresentam-se em Watt para a potência radiada e lúmen para o fluxo luminoso.

A **energia** e **energia luminosa** são dadas pelo produto do fluxo em função do tempo **t**. As unidades são **Joule** (J) para a energia e o **lm segundo** para a energia luminosa.

A **intensidade energética** (**I**) e **intensidade luminosa** (**Iv**) traduzem a potência radiada numa determinada direção sobre um ângulo sólido unitário (em esterradiano, sr) e também se relacionam com a fonte luminosa. As unidades são o $\mathbf{W/sr}$ para a intensidade energética e o $\frac{\mathbf{lm}}{\mathbf{sr}} = \mathbf{candela}$ (cd) para a intensidade luminosa.

A **radiância** (**L**) e a **luminância** (**Lv**) são medidas da superfície da fonte de luz. Tal com a intensidade energética e luminosa, estes valores são características da fonte de luz e também estão relacionadas com a área unitária da fonte. As unidades são o $\mathbf{W/m^2 sr}$ para a radiância e $\mathbf{cd/m^2}$ para a luminância.

A **irradiação** (**E**) e **iluminação** (**Ev**) são medidas relativas ao detetor e as unidades são o $\mathbf{W/m^2}$ para a irradiação e $\frac{\mathbf{lm}}{\mathbf{m^2}} = \mathbf{lux}$ para a iluminação. (Cruz, et al., 2012)

Os sensores fotelétricos são utilizados para monitorizar a intensidade luminosa que incide na estufa e consequentemente nas plantas. Existem dois tipos, os sensores fotocondutores resistivos e os fotocondutores semicondutores. A seguir apresenta-se uma descrição para cada um.

### 4.3.1 Foto-Resistência (LDR)

W. Smith em 1873 descobriu que o valor de uma resistência de selênio dependia da intensidade luminosa. Desde essa altura que a foto condução é uma ferramenta importante para a caracterização das propriedades dos materiais. (Webster, 1999)

As foto-resistências, vulgarmente conhecidas pelo acrónimo LDR (Light Dependent Resistor), variam o valor resistivo de acordo com o fluxo de luz incidente. Quando o material fotocondutor está sob a influência da radiação incidente, existe uma libertação de cargas elétricas que origina um aumento da condutividade.

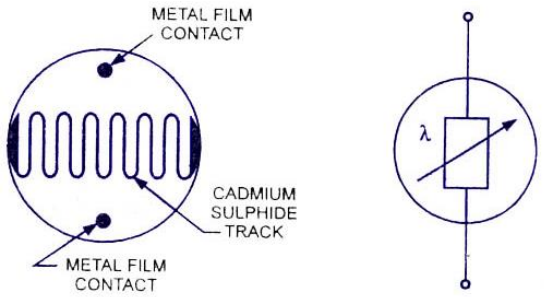



**Figura 14:** Esquema de um LDR e símbolo elétrico.

O material mais utilizado na construção de um LDR é o sulfito de cádmio (CdS) que exibe uma forte resposta fotocondutiva e cuja sensibilidade centra-se entre os $300nm$ e os $880nm$ com um máximo nos $550nm$, sendo similar à do olho humano.

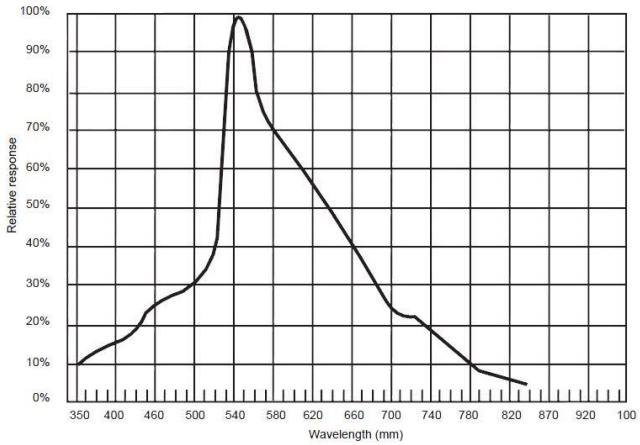



**Figura 15:** Relação entre a resposta fotocondutiva e o comprimento de onda.

Um LDR apresenta uma variação praticamente linear entre condutividade e radiação. Como se pode verificar na seguinte Figura 16, a relação entre a resistência e a radiação incidente é uma exponencial decrescente.

O tempo de resposta é elevado e apresenta variações com a temperatura.

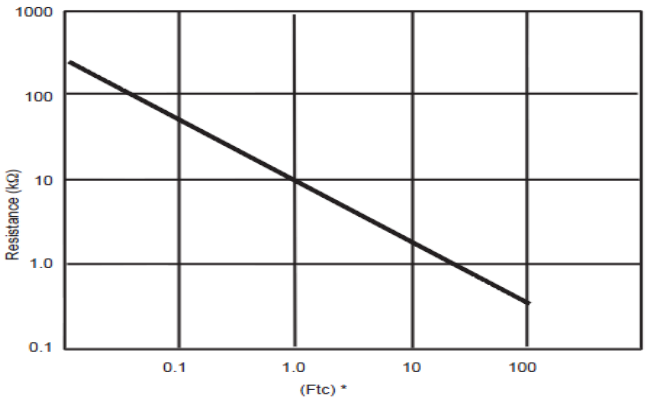



**Figura 16:** Relação entre o valor da resistência e a iluminação de um LDR.

O NORP12 da RS é um LDR constituído por duas células fotocondutoras de cadmio, custa 3,85€ e pode ser facilmente ligado a um ADC de um microcontrolador usando um divisor de tensão.

### 4.3.2 Foto-Díodo

Os foto-díodos e os foto-transístores são dispositivos semicondutores que traduzem os eletrões gerados pelo efeito fotelétrico num sinal elétrico. O foto-díodo pode funcionar em dois modos, fotocondutivo e fotovoltaico. Quando a junção p-n de um foto-díodo é polarizada inversamente, o foto-díodo opera no modo condutivo

comportando-se como uma fonte de corrente controlado pelo fluxo incidente na junção. Apresenta uma relação bastante linear entre o fluxo incidente e a foto-corrente gerada.

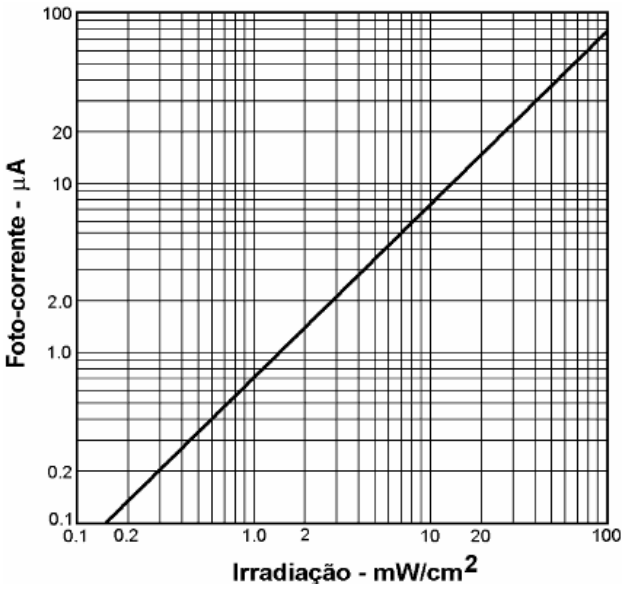



**Figura 17:** Característica corrente versus irradiação típica de um foto-díodo.

Quando não existe uma polarização externa, o foto-díodo funciona no modo fotovoltaico onde apresenta, aos seus terminais, uma corrente proporcional ao fluxo de luz incidente na junção p-n. Este efeito é utilizado nos painéis solares para converter luz em tensão elétrica para alimentações a energia solar. Os foto-díodos possuem pequenas dimensões, são robustos e apresentam uma resposta linear em função do fluxo incidente sendo o seu custo bastante reduzido.

Atualmente a maior parte dos sensores de radiação são baseados em foto-díodos fabricados com materiais que oferecem uma resposta relativa desejada face aos comprimentos de onda de interesse. Grande parte das máquinas fotográficas atualmente utilizam este tipo de sensor monocromático para captar a energia da luz e através de filtros transforma-la numa imagem visível.

## 4.4 Sensores de CO2

O dióxido de carbono (CO2) presente no ar em conjunto com a água, luz e calor, é assimilado pelas plantas de forma a sintetizarem matéria orgânica. Em condições favoráveis, o aumento de CO2 no ar leva a um maior desenvolvimento das plantas.

Os sensores de dióxido carbono dividem-se em dois tipos tendo como ponto de partida o método de mediação, são eles os sensores Infravermelhos Não-Dispersivos (NDIR Non-Dispersive InfraRed) e os sensores eletroquímicos. (Kwon, et al., 2009)

### 4.4.1 Eletroquímico

Os sensores de CO2 químicos utilizam polímeros que são colocados em camadas para medir a concentração de dióxido de carbono. O conteúdo do sensor consiste num reservatório de bicarbonato coberto por uma membrana permeável ao gás. A tampa contém um ácido fluorescente sensível ao pH, comparando a fluorescência do corante em dois comprimentos de onda diferentes consegue-se determinar a concentração de CO2 da amostra. (Quan, et al., 2011)

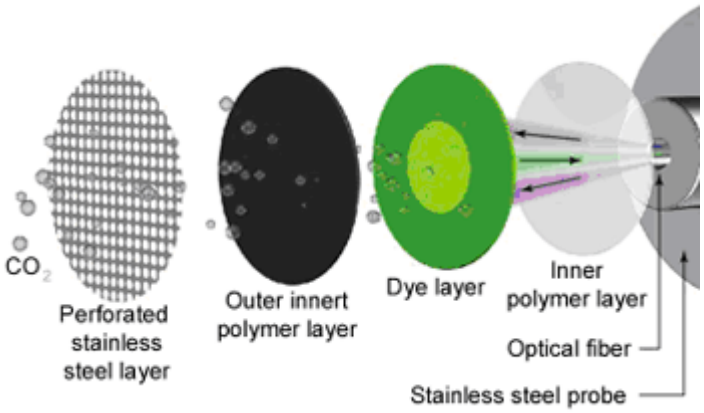



**Figura 18:** Esquema de um sensor de CO2 eletroquímico.

As principais vantagens dos sensores de CO2 químicos são o baixo consumo de energia e as reduzidas dimensões podendo ser aplicados na microeletrónica. Em contra partida a sua durabilidade e tempo de vida são curtos, necessitam de calibrações e degradam-se muito.

### 4.4.2 NDIR

Os gases em geral, e neste caso o CO2, absorvem um determinado comprimento de onda. Os sensores infravermelhos não-dispersivos usam esta particularidade que os gases têm, de absorverem um determinado comprimento de onda para efetuar a medição da concentração no ar.

É utilizado um emissor infravermelho de banda larga que cobre todos os comprimentos de onda para a medição de um conjunto de gases, depois utiliza-se um filtro ótico passa-banda que apenas permite a passagem do comprimento de onda referente ao gás que se pretende medir. O sinal resultante produzido pelo detetor é proporcional à energia infravermelha absorvida pelo gás. Isso origina um sinal elétrico e posteriormente gera informações sobre a concentração do gás no ar.

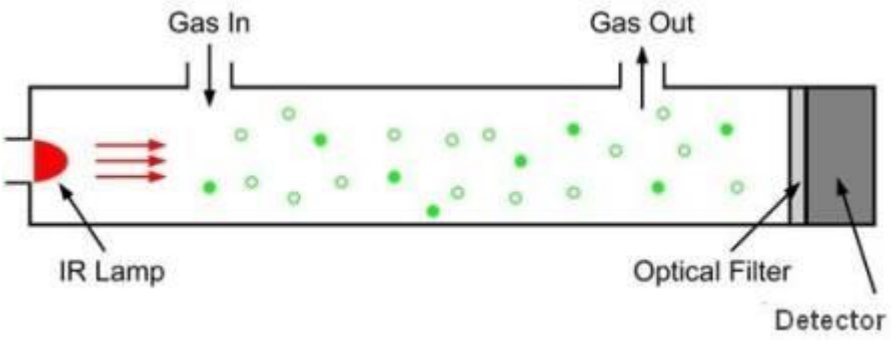



**Figura 19:** Esquema de um sensor NDIR.

Fonte: http://www.lumasenseinc.com/BR/solutions/techoverview/ndir/

Este tipo de sensores em relação aos sensores químicos tem como vantagem a elevada precisão, grande durabilidade, estabilidade e seletividade em relação ao gás a medir. O sensor T6615 da *Telaire* é um sensor do tipo NDIR que permite a medição da concentração de CO2 com uma gama entre 0 ppm e 50 000 ppm, uma precisão de 75 ppm ou 10%. O seu preço é de 135,96€ (Farnell Components SL, 2014) que comparativamente com outros sensores é elevado. Alternativamente existe o sensor MG-811 da Olimex com uma gama de mediação entre 400 ppm e 10 000 ppm, e um preço aproximado de 40€.

## 4.5 Sensores de pH

O pH da solução nutritiva é muito importante no cultivo em hidroponia pois as plantas não conseguem sobreviver a valores abaixo dos 3.5 e o seu desenvolvimento máximo situa-se para valores entre os 5.5 e os 6.5.

### 4.5.1 Elétrodos com Membrana

O método mais comummente utilizado na medição de pH é através de um elétrodo de membrana de vidro. Este sensor é composto por dois elétrodos, ambos são colocados no líquido que se pretende medir, sendo um deles o elétrodo de referência e o outro o elétrodo indicador. A medição é feita através da diferença de potencial (voltagem) entre os dois elétrodos. Os elétrodos são combinados unicamente numa ponta de prova, a membrana de vidro do elétrodo indicador desenvolve um potencial elétrico depende do valor do pH, em resultado da troca entre iões de hidrogénio na solução e catiões univalentes na membrana de vidro.

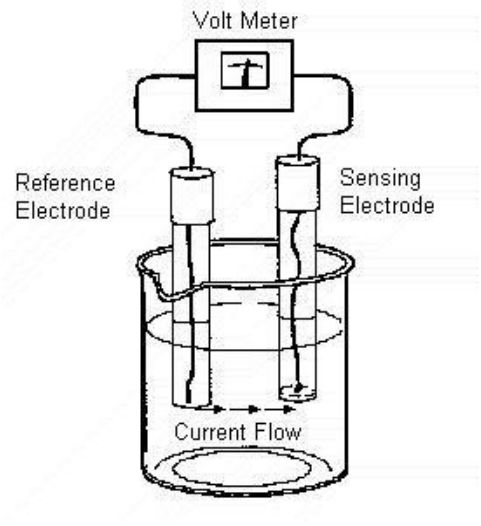



**Figura 20:** Esquema de um sensor de pH baseado em elétrodos.

O sensor da JUMO com revestimento a plástico PPO apresenta uma gama de medição de pH entre 0 e os 12. Tem um preço 67,58€ (RS, s.d.).

### 4.5.2 pH FET

O pH também pode ser medido recorrendo a transístores de efeito de campo sensíveis a iões (ISFETs - Ion-Sensitive Field-Effect Transístor) neste caso de hidrogénio. Resultam da derivação de MOSFETs (Metal-Oxide-Semiconductor FET). Estes circuitos integrados de silício conseguem ter uma resposta à variação do pH muito semelhante aos elétrodos de vidro e ao mesmo tempo propiciam uma amplificação de um transístor com efeito de campo. A tensão de limiar do ISFET (ponto 3 da Figura 21) varia linearmente com o pH da solução. (Scaff, 2008)

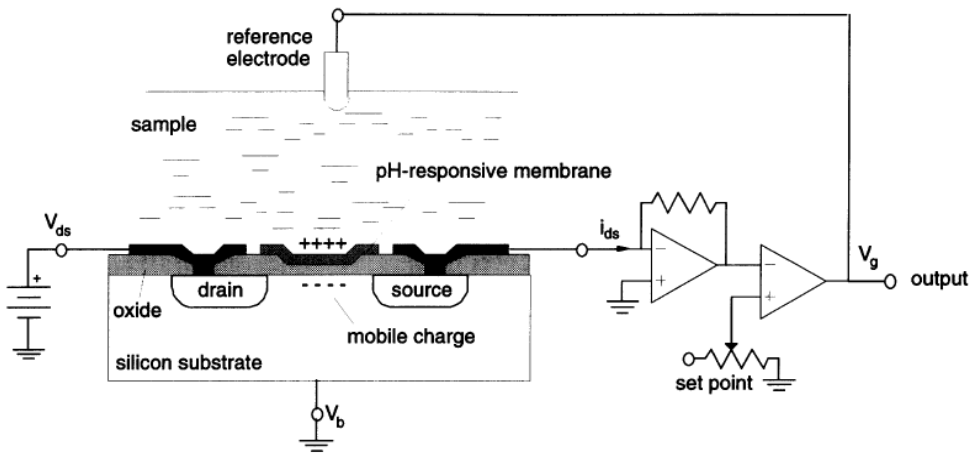



**Figura 21:** Esquema de um sensor pH baseado em ISFET.

Fonte: *http://www.jumo.co.uk/products/liquid%20analysis/ph/electrodes/201050/isfet-ph-single-rod-electrode-201050.html*

## 4.6 Sensores Condutividade Elétrica

O controlo da condutividade elétrica da solução nutritiva fornece-nos a informação de quantos iões estão presentes na solução, isso representa a quantidade de adubo existente. O crescimento das plantas é afetado por esta variável, pois uma elevada condutividade elétrica pode levá-las à morte ou cessar o seu crescimento. Por outro lado valores baixos indicam a falta de algum elemento na solução nutritiva que deverá ser reposto (no entanto não é possível saber-se qual sem uma análise química laboratorial).

### 4.6.1 Sensores por Elétrodos

A condutividade elétrica é a medida da capacidade de uma solução transportar corrente. O fluxo de corrente nos metais é feito pela deslocação dos eletrões para as lacunas, nos líquidos é diferente, o fluxo de corrente é efetuado pelos iões. Os iões são formados quando existe um sólido, como por exemplo o sal, dissolvido num líquido formando assim cargas elétricas opostas. No caso do cloreto de sódio separa-se para formar iões Na+ e Cl-. Todos os iões presentes na solução contribuem para o fluxo de corrente que passa através do sensor, e por conseguinte para medir a condutividade.

A medição é feita mergulhando dois elétrodos no líquido ao mesmo tempo que é mantida uma corrente constante e moderada a circular por eles. O volume do líquido entre os dois elétrodos deve ser exato. Este efeito é conhecido como *cell constant.* Qualquer variação no volume faz variar também o valor da *cell constant* e da corrente.

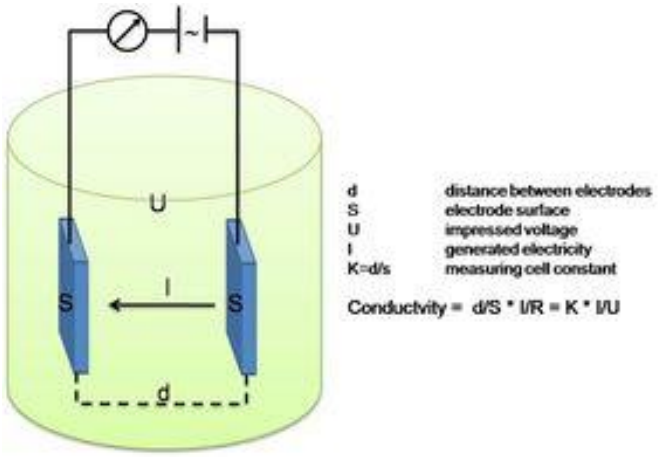



**Figura 22:** Esquema de mediação de um sensor de condutividade por elétrodos.

O LFS 155 da *Innovative Sensor Technology* é um sensor que mede a condutividade em eletrólitos. Integra também no mesmo package um sensor de temperatura PT1000. Apresenta uma gama de medição de condutividade entre os

$5\mu s/cm$ e os $50ms/cm$ e de temperatura entre os $-20\,°C$ e os $+80\,°C$. A frequência de medição aplica-se entre os $300Hz$ e os $3000Hz$ com uma corrente de $1.6\,Vpp$. O preço é de $55{,}23€$ (Farnell Components SL, 2014).

### 4.6.2 Sensores Indutivos

O sensor de condutividade indutivo é constituído por duas bobinas que são incorporadas num polímetro, uma ao lado da outra. Estas bobinas formam um transformador de corrente. O sensor é projetado para quando for colocado num meio líquido se forme um caminho condutor de corrente entre as duas bobinas. A corrente é aplicada na bobina primária (bobine indutora), o qual induz uma tensão alternada na solução aquosa. Os líquidos que conduzem melhor a energia elétrica geram um fluxo de corrente que é capturado pela segunda bobina (bobina recetora). Esse fluxo de corrente é proporcional à condutividade da solução.

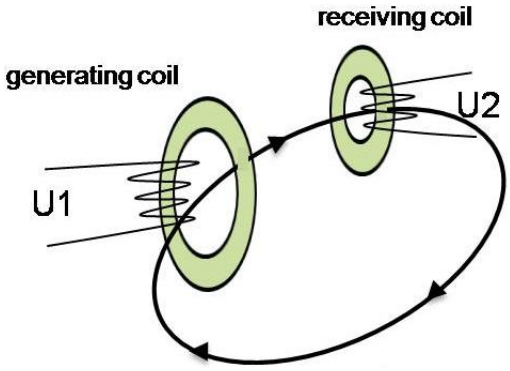



**Figura 23:** Esquema de medição e um sensor de condutividade por indução.

## 4.7 Sensores de Nível

O sensor de nível é utilizado no tanque que contém a solução nutritiva para indicar o limite máximo ou noutras aplicações para indicar os níveis de algum nutriente. Existem diversos tipos de sensores de nível, dependendo do tipo de aplicação uns são mais apropriados do que outros, são eles: a boia de ação magnética ou mecânica, os sensores capacitivos, por ultrassons, micro-ondas e magnéticos.

O sensor magnético possui um íman no seio de uma boia que ao mover-se ao longo de um cilindro aciona um sensor magnético que por sua vez fecha o contacto entre dois fios que saem do sensor. Conforme de desloca a boia, liga ou desliga o circuito, indicando se o líquido está na posição máxima ou abaixo. É o sensor mais de nível mais simples, assemelhando-se ao sensor de boia. Pela facilidade de

implementação, ao se colocar vários sensores deste tipo num só tanque pode-se monitorizar vários níveis do líquido.

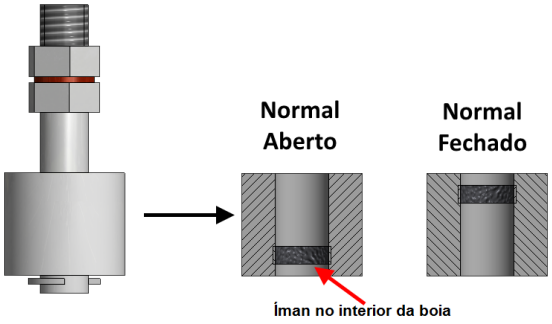



**Figura 24:** Esquema de um sensor magnético.

O sensor magnético 519-242 da RS é feito a partir do plástico PP adequado para água e ácidos diluídos, composto por uma boia de material flutuante. O preço é de 6,42€ (RS, s.d.).

## 4.8 Microcontrolador

Um microcontrolador é um computador embutido num único chip, possui periféricos de entrada e saída, memória e um processador. Especificamente é um microprocessador que pode ser programado com funções específicas. Na eletrónica, os microcontroladores são utilizados essencialmente no controlo de pequenos sistemas, como por exemplo uma máquina de lavar roupa.

O sistema de monitorização utiliza diversos sensores para medir as variáveis físicas estipuladas (temperatura, humidade, etc.). A esmagadora maioria dos microcontroladores possui como periférico uma ADC. A ADC – Conversor Analógico Digital, como o próprio nome indica, tem como função converter um valor analógico variável no tempo, i.e. tensão elétrica, num valor digital-número. Assim, uma ADC com uma resolução de $10bits$ que recebe um sinal analógico entre $0\ V$ e $5\ V$ pode assumir valores entre $0\ (0000000000)$ e $1023\ (1111111111)$, ou seja, é capaz de capturar $1024$ níveis discretos de um determinado sinal. Através da ADC interliga-se os sensores com o microcontrolador, assim o sinal físico é adquirido pelo sensor convertido pela ADC e transmitido ao microcontrolador onde pode armazenar o valor em memória ou efetuar outra ação.

O sistema de monitorização que será desenvolvido ao longo deste trabalho assenta sobre uma rede de sensores sem fios, cada módulo é uma unidade de aquisição

de dados físicos. Deste modo, o microcontrolador que integra o módulo de memória e processamento deve ter como característica principal o baixo consumo de energia e também garantir que a memória, processamento e os periféricos sejam suficientes para a ligação dos sensores e do módulo de comunicações.

| Tipo | Característica | Número |
|---|---|---|
| I2C | Comunicação | 1 |
| SPI | Comunicação Série | 1 |
| ADC | Conversor Analógico-Digital | 5 |
| USART | Comunicação Série | 1 |

**Tabela 3:** Principais características que o microcontrolador deve ter.

Hoje em dia a indústria dos semicondutores está largamente difundida, em particular existem diversas empresas que produzem microcontroladores tecnologicamente muito evoluídos. Para este sistema de monitorização a característica para ponto de partida é o baixo consumo de energia dos equipamentos. A *Atmel* há mais de 10 anos que aposta no baixo consumo de energia para os seus produtos AVR e microcontroladores baseados em ARM. (Atmel, 2014)

O modelo ATxmega16A4 da *Atmel* é de baixo consumo de energia, apresenta um consumo entre 30 µA e 11.4 mA dependendo da velocidade de clock em funcionamento. É um exemplo de um microcontrolador que disponibiliza também as interfaces SPI, I2C, USART e ADC.

A *Texas Instruments* utiliza métodos para a fabricação de microcontroladores que desperdiçam o mínimo de energia possível. O MSP430x é um exemplar com um consumo médio em modo ativo de 280 µA. ( Texas Instruments , 2014)

A *Microchip* aposta no mais baixo consumo de energia para os seus produtos, hoje em dia as redes de sensores sem fios estão ligadas à internet das coisas (IoT) onde a conservação de energia é um fator muito importante. Segundo eles, existem aplicações de baixo consumo de energia que, em casos extremos, devem durar entre 15 a 20 anos só com uma única bateria. Apresenta assim os seus sistemas de XLP (eXtrem Low Power) com consumos em modo *sleep* de 9 nA e em funcionamento abaixo dos 30 µA. A PIC16F1784 obtém um consumo em funcionamento de 32 µA/MHz. A Tabela 4 apresenta um resumo das principais características dos microcontroladores apresentados destes 3 fabricantes.

| Fabricante | Modelo | Memória | Especificações | Consumo Energético | Preço |
|---|---|---|---|---|---|
| Microchip | PIC16F1784 | 512 byte | I2C, SPI, USART, ADC12 | 32µA/MHz | 2,35 € |
| Atmel | ATxmega16A4 | 2Kb | I2C, SPI, USART, ADC12 | 260µA/MHz | 2,52 € |
| Ti | MSP430F133 | 256 byte | SPI, USART, ADC12 | 280µA/MHz | 5,41 € |

**Tabela 4:** Comparação entre alguns microcontroladores de baixo consumo de energia.

## 4.9 Módulo de Comunicações

As redes de sensores sem fios são bastante utilizadas para monitorizar um fenómeno ou variável física. Na maioria dos casos, os locais onde se coloca um nó sensor são de difícil acesso e requerem que o sistema seja o mais fiável e autónomo possível. As comunicações utilizadas na rede de sensores, assim como os restantes componentes que compõem o nó sensor, obedecem ao critério do baixo consumo de energia. Neste caso é necessário encontrar um equipamento que possibilite a transmissão do valor das variáveis medidas de um nó para outro até chegar ao ponto central da rede.

A norma IEEE 802.15.4 é utilizada nas redes de sensores sem fios e descreve as especificações da camada PHY (física) e a subcamada MAC (enlace) para este tipo de redes. (Mouftah, et al., 2014) As restantes camadas do modelo OSI são utilizadas por *standards* como é o caso do ZigBee® e do 6LoWPAN. (Xu, et al., 2008) (Khattak, et al., 2014)

A Tabela 5 sugere um exemplar de um módulo de comunicações *wireless* baseado na norma IEEE 802.15.4 de diferentes fabricantes. A *Microchip* apresenta um custo-benefício bom, e também um nível de consumo de energia equivalente aos restantes.

| Fabricante | Modelo | Protocolos | Consumo Tx/Rx | Preço |
|---|---|---|---|---|
| TI | CC2520 Emk | 6LoWPAN, ZigBee | 25.8/18.5mA | (1) 101,58 € |
| Atmel | AT86RF231 | ZigBee, ISM applications | 29/24mA | (2) - |
| Microchip | MRF24J40MA | ZigBee, MiWi, MiWi P2P | 23/19mA | 7,44 € |

**Tabela 5:** Principais características dos módulos de transmissão *wireless* com a norma IEEE 802.15.4.

Nota: (1) O preço corresponde a dois dispositivos;
(2) Não existe fornecedor na Europa para este equipamento.

## 4.10 Módulo de Alimentação

As redes de sensores sem fios, como foi descrito no ponto anterior, por vezes integram os nós sensores em locais remotos onde não há uma fonte de energia. Para além deste fator, torna-se muito dispendioso criar uma linha de energia para alimentar cada nó sensor e ao mesmo tempo deixaria de fazer sentido a própria rede de comunicações sem fios. Os nós sensores são caracterizados pelo baixo consumo de energia, assim normalmente são alimentados por pilhas ou baterias. As pilhas ou baterias se forem recarregáveis e existir um sistema que as possa carregar, o caso de um painel solar, aumenta a autonomia do sistema sem a necessidade de substituição da fonte de energia.

Um painel solar tem uma dimensão proporcional a quantidade de energia que produz. Sabe-se que a energia produzida depende de fatores como o ângulo de incidência dos raios solares e a luminância. A *Multicomp* apresenta um painel solar de baixo custo MC-SP0.8-NF-GCS, 7.79€ (Farnell Components SL, 2014) com uma potência $800mW$, uma voltagem máxima de $3.85v$ e uma intensidade de corrente máxima de $210mA$, que poderá ser suficiente para alimentar todo o sistema.

Em conjunto com o painel solar é necessário uma bateria ou pilha recarregável que permita alimentar o sistema quando não haja sol (durante a noite) ou mesmo quando a produção de energia não seja suficiente para alimentar todo o sistema.

Existem diversas baterias e pilhas recarregáveis com diversos níveis de tensão e corrente. No sistema que se pretende implementar o consumo de energia é muito baixo, colocando o exemplo teórico do microcontrolador que consome $32\mu A$ em funcionamento e o módulo de comunicação que necessita de $23mA$ para transmissão e $19mA$ para a receção. A *Varta* apresenta uma bateria recarregável NI-MH com uma capacidade de $150mAh$ ou com $700mAh$ e uma voltagem de $3.6v$. O preço aproximado de ambas é de 6,60€ (Farnell Components SL, 2014).

# 5. ARQUITETURA DO SISTEMA

O sistema de controlo e monitorização para culturas por hidroponia requer um enorme número de sensores e atuadores de forma a garantir que o processo produtivo seja cumprido sem danificar a cultura. Como já foi referido, um sistema hidropónico divide-se essencialmente em dois campos, o tanque que contém a solução nutritiva ou seja o alimento, e o suporte ou sustentação das plantas (onde elas crescem). Ambos estão interligados sendo que um é dependente do outro e vice-versa. O sistema de controlo que se pretende desenvolver, para além deste dois campos essenciais à cultura por hidroponia acrescenta o ambiente envolvente como um elemento fulcral. Numa estufa o controlo de temperatura, luminosidade, humidade e outros fatores, propícia ao agricultor informação importante sobre o *stress* (no sentido das temperaturas serem muito altas ou muito baixas, a humidade muito alta ou muito baixa, etc.) das plantas.

Este sistema pode ser assim dividido em 3 partes: sistema de monitorização no tanque que contém a solução nutritiva, sistema de monitorização do ambiente em diversos locais da estufa e por fim o sistema de controlo que recebe informação dos sistemas anteriores e toma uma decisão. A Figura 25 apresenta um esquema que contempla todos os constituintes do sistema de controlo e monitorização.

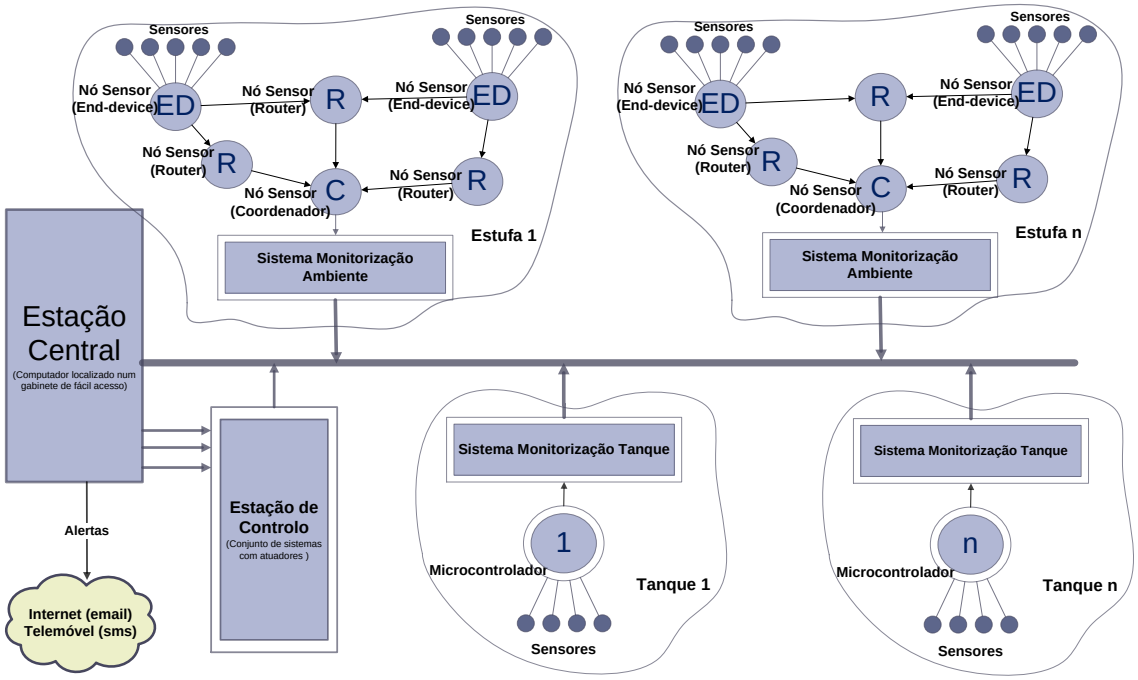



**Figura 25:** Esquema com a estrutura de controlo e monitorização para uma unidade de cultura por hidroponia em estufa.

A estrutura de forma hierárquica permite que cada elemento individual execute uma tarefa e transmita ao seu superior o resultado. Este processo repete-se até ao cimo da hierarquia. Na estação central processa-se toda a informação recolhida, caso seja necessário efetuar alguma ação, como por exemplo ligar a ventilação, é gerado um alerta e posteriormente atuado o sistema de ventilação. Outro exemplo, se o nível de água no tanque for baixo, o sistema de monitorização do tanque comunica com a estação central, que gera um alerta para o agricultor e ao mesmo tempo manda atuar o motor que transporta a água.

O encaminhamento de toda a informação para a estação central permite que os dados estejam acessíveis de forma transparente para o agricultor a qualquer momento, por outro lado possibilita também que haja um controlo estatístico ao longo do tempo de cada variável física monitorizada.

No caso de existirem diversas estufas ou exista a necessidade de descentralizar o computador central numa estufa por algum motivo, deve-se ter em conta o sistema de ligação utilizado. A comunicação por rádio frequência ou *wireless* é bastante limitadora em termos de distância e perde-se muita força de sinal ao atravessar zonas que contenham obstáculos. Uma das soluções seria ciar uma rede LAN utilizando cabo de rede (RJ45) entre os pontos de ligação, no entanto existe a necessidade de repetidores a cada 100 metros. A melhor solução é a utilização de fibra ótica porque oferece uma alta taxa de transmissão de dados superiores 160 Gbit/s, baixa atenuação - até 120 quilómetros sem repetidor e neste momento um cabo de fibra ótica monomodo com 100 metros custa cerca de 75,04€ (RS, s.d.), cerca de 0,75€ por metro.

O projeto de uma unidade de aquisição de dados, como referido acima, requer uma especial atenção à forma como será operada ao longo do tempo pelo utilizador, neste caso o agricultor hidropónico. O sistema deverá ser simples e de utilização intuitiva, ao mesmo tempo deverá ser robusto e fiável de forma a garantir a sobrevivência das culturas em caso de imprevistos.

## 5.1 Protocolo ZigBee

O *ZigBee* é uma tecnologia de comunicação sem fios destinada a sistemas com necessidades de baixo consumo de energia, pouco alcance, baixa velocidade de transmissão de dados e custo reduzido. É baseado no *standard* IEEE 802.15.4 com a

capacidade de coordenar a comunicação entre milhares de pequenos sensores. (LI, et al., 2011)

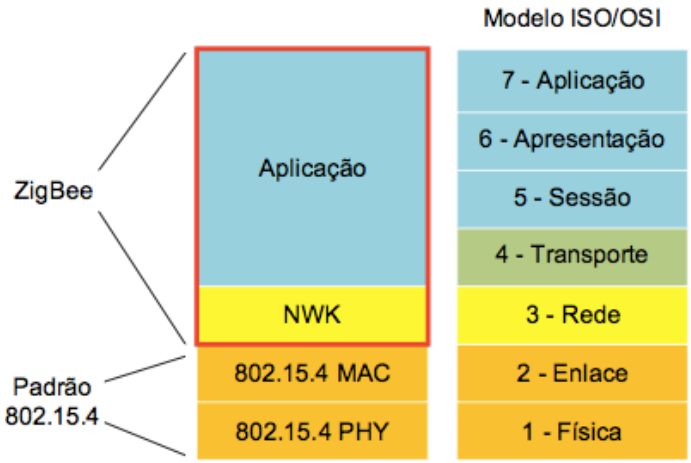



**Figura 26:** Modelo OSI comparado com o *standard* IEEE 802.15.4 e o ZigBee.

Aplica as camadas superiores do modelo OSI (Figura 26) agrupando-as em apenas duas, Rede e Aplicação. Através das ondas rádio, os nós sensores conseguem transmitir informação para outro nó com um baixo consumo de energia e uma elevada eficiência. Comparando com outros sistemas de comunicação *wireless*, a tecnologia utilizada no ZigBee é a que tem um menor consumo de energia e custo. As principais características deste sistema são:

- Frequência de operação de 2.4 GHz e taxa de dados de 250 Kbps;
- Permite que o mesmo nó funcione com funções diferentes;
- Possibilita configurações em diversas topologias de rede;
- Consegue auto organizar-se e auto reestruturar-se: self-organizing e self-healing;
- Permite um número elevado de dispositivos conectados à rede (máximo de 65535 dispositivos por cada dispositivo coordenador);
- Alta durabilidade da bateria dos dispositivos;
- Interoperabilidade, ou seja, a capacidade de se comunicar de forma transparente com outros sistemas;
- Mantém a integridade dos dados e utiliza o algoritmo de encriptação com a função AES-128.

### 5.1.1 Tipos de Dispositivos

O padrão IEEE 803.15.4 define para as redes ZigBee dois tipos de dispositivos (Li, et al., 2011):

- RFD - *Reduced Function Device*: dispositivos de função reduzida
- FFD - *Full Function Device*: dispositivos de função completa

Os dispositivos FFD podem funcionar em qualquer um dos modos de operação do padrão, ou seja, como *coordinator*, *router* ou *end-device*. Podem também comunicar com outros FFD e com dispositivos RFD.

Os dispositivos RFD são dispositivos mais simples e com um custo menor que só podem comunicar com dispositivos FFD. Desta forma estes dispositivos apenas poderão funcionar como *end-device* na rede.

O dispositivo *coordinator* (coordenador) é o nó inicial da rede. Tem como função inicializar e atualizar a topologia da rede, transmitir o símbolo de rede, gerir os nós da rede, armazenar informações dos nós, fornecer informações de rotas entre nós e armazenar dados dos nós sensores. Ao ser ligado pela primeira vez cria uma rede selecionando um identificador PAN único no seu raio de atividade. Opera no estado ativo para efetuar o controlo da rede e costuma ser alimentado diretamente reduzindo o risco de falha.

Os dispositivos que funcionam como *routers* são usados nas topologias malha (mesh) e árvore (tree) para dar maior robustez à rede. Eles possuem tabelas de encaminhamento e, por serem FFD, permitem encontrar o menor caminho para se chegar ao destino. Caso o *router* não possua o endereço de destino requisitado, este fará um *broadcast* de uma requisição de rota (route request) e receberá do destino a rota mais eficaz atualizando depois a sua tabela. Este mecanismo dá à rede a característica de autorregeneração caso ocorra a queda das funcionalidades de outros nós com função de *router* na rede.

Por fim, os dipositivos *end-device* são os nós folhas das topologias estrela e árvore. Por serem dispositivos RFD, não tem função de encaminhamento nem coordenam a rede. Eles comunicam diretamente com o *router* pai e podem ser implementados com microcontroladores menos potentes (em termos de memória e processamento), passando quase todo o tempo em estado inativo. Os sensores, atuadores e sistemas de controlo são normalmente aplicados num dispositivo RFD.

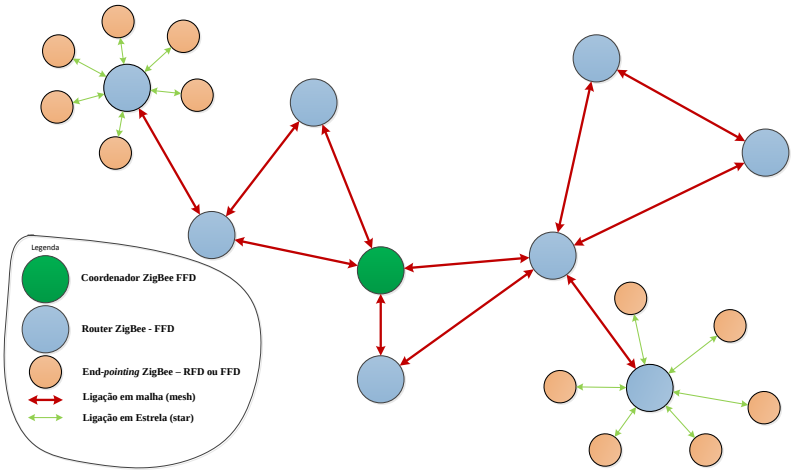



**Figura 27:** Exemplo de uma rede baseada em ZigBee.

### 5.1.2 Topologias de Rede

As diferentes funcionalidades dos dispositivos permitem também uma variedade na forma como se instala a rede. De acordo com a necessidade da aplicação, uma rede pode ser mais robusta, mais econômica, centralizada ou distribuída. As topologias apresentadas são estrela, malha e árvore.

Na topologia estrela (star) os dispositivos conectam-se a um único nó central. Cada rede funciona com um identificador PAN diferente permitindo que cada rede opere individualmente mesmo estando sob influência das ondas rádio de outra rede.

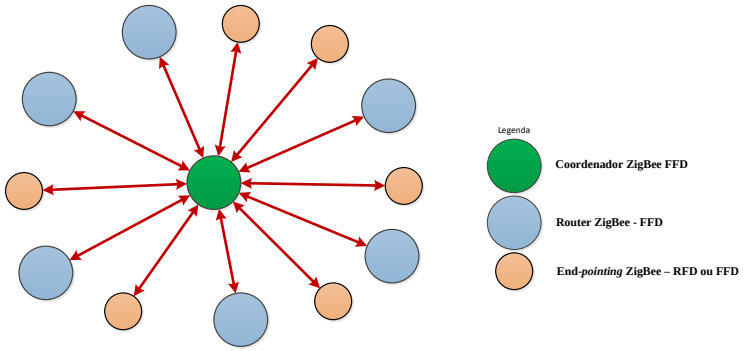



**Figura 28:** Topologia de rede em estrela.

Na topologia malha (mesh) existe apenas um nó central coordenador da PAN. Todos os dispositivos comunicam entre si desde que estejam dentro do alcance da rede. Esta topologia de rede permite reorganizar-se e estruturar-se sozinha mediante as necessidades, o que a torna muito robusta.

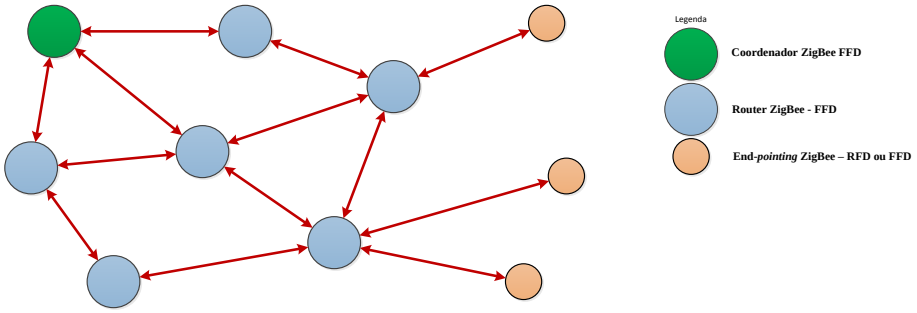



**Figura 29:** Topologia de rede em malha.

Por fim, exista a topologia em árvore (tree) que pode ser descrita como um aglomerado de redes em malha mas contendo apenas um nó central responsável pela rede. Este nó central coordenador da PAN (CLH-Cluster Head) define a identificação para a rede (CID – Cluster Identifier) através da escolha de um identificador PAN ocioso. Seguidamente o dispositivo faz *broadcast* do *beacon frame* para anunciar a criação da rede. Se algum dispositivo receber este sinal e pretender entrar na rede, ele faz a requisição ao CLH e se o coordenador PAN autorizar será adicionado o novo dispositivo filho. A principal vantagem dessa estrutura em árvore é aumentar a área de cobertura. O custo a pagar por este aumento é o atraso na receção da mensagem. (Vasques, et al., 2010)

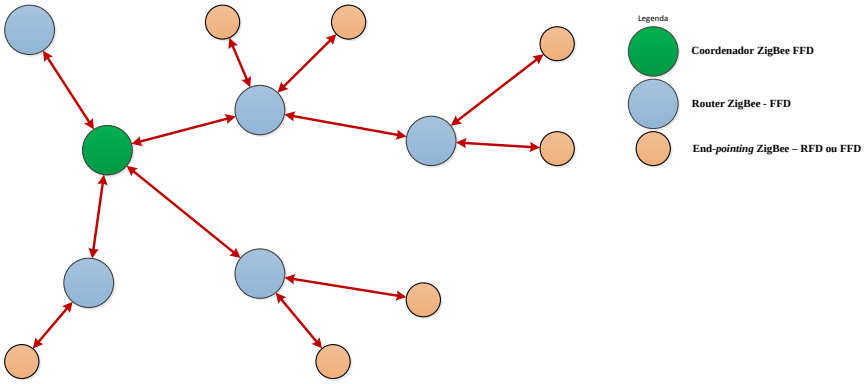



**Figura 30:** Topologia de rede em árvore.

## 5.2 Protocolo 6LoWPAN

O 6LoWPAN, acrônimo de *IPv6 over Low power Wireless Personal Area Networks*, é um grupo responsável por desenvolver meios que permitem a transferência de pacotes do tipo IPv6 sobre redes do tipo 802.15.4. O 6LoWPAN é um dos primeiros protocolos que permite ligações em IPv6 a dispositivos com baixa capacidade de processamento e memoria. O uso do IP nas redes *wireless* PAN traz uma série de

vantagens porque assim é possível aproveitar toda a estrutura já existente em que se baseia a internet para comunicar, eliminando a necessidade de *gateways* complexos.

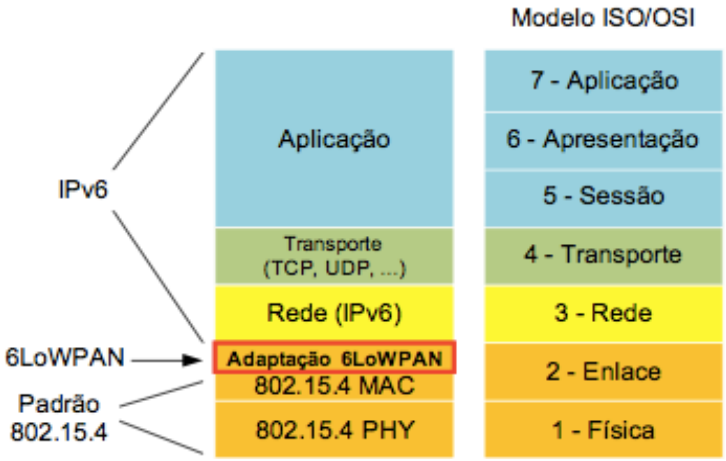



**Figura 31:** Modelo OSI em comparação com o 6LoWPAN.

O 6LoWPAN caracteriza-se por ser uma camada de aplicação, um nível dois e meio que permite aplicar endereçamento IP, neste caso IPv6, em redes sem fios pessoais de baixo consumo de energia, processamento e alcance. O funcionamento em termos de topologias de rede e tipos de dispositivos é igual ao protocolo descrito anteriormente, o ZigBee. (Pediredla, et al., 2013)

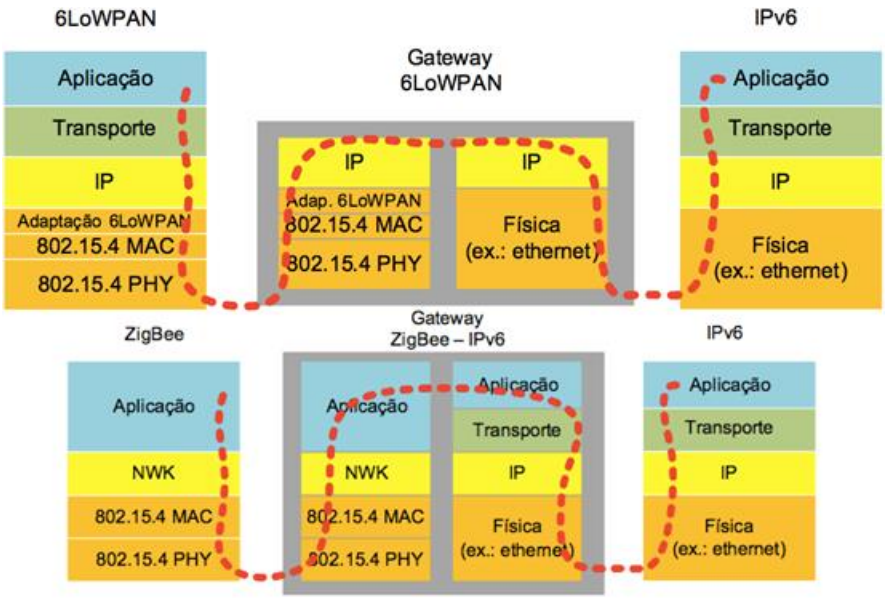



**Figura 32:** Comparação entre o protocolo 6LoWPAN e ZigBee na transmissão de dados para uma rede IPv6.

A Figura 32 apresenta o percurso que um pacote numa rede IEEE 802.15.4 com o protocolo ZigBee e numa rede IEEE 802.15.4 com protocolo 6LoWPAN faz desde a camada da aplicação até chegar à camada da aplicação de um ponto de rede IPv6. A principal diferença está no *gateway* utilizado em cada um dos sistemas, no 6LoWPAN a

transmissão entre as redes é praticamente direta, apenas necessita da conversão de alguns cabeçalhos. (Keng, et al., 2009)

# 6. DESCRIÇÃO DOS NÓS DE REDE

Neste capítulo descreve-se o sistema de monitorização desenvolvido, que pretende satisfazer as necessidades encontradas na cultura hidropónica referidas no ponto 3.4 Análise dos Requisitos, em particular para a recolha de dados sobre o ambiente da estufa. A solução proposta para o desenvolvimento do sistema de monitorização consiste na criação de uma rede de sensores sem fios baseada no protocolo IEEE 802.15.4 e ZigBee com o objetivo de transmitir os dados recolhidos pelos sensores para um computador central.

Na RSSF existirão dois tipos de nós, os *routers* e os *end-device*. Cada nó desempenha uma tarefa específica, é autónomo energeticamente e comunica com outros nós.

## 6.1 Nó *Router*

O nó *router* tem como função receber os dados recolhidos pelo nó *end-device* e transmiti-los a outro nó da mesma categoria ou então envia os dados diretamente para o computador central. A dimensão de uma área agrícola normalmente é grande (na ordem dos hectares), isto faz com exista a necessidade de percorrer longas distâncias entre o local onde se encontra o nó *end-device* e a estação central. A topologia de rede mais indicada para esta aplicação é a árvore por permitir um maior alcance com um menor número de nós. O nó *router* é composto por 3 módulos, são eles o módulo de alimentação, módulo de processamento e o módulo de comunicações, como apresenta a seguinte Figura 33.

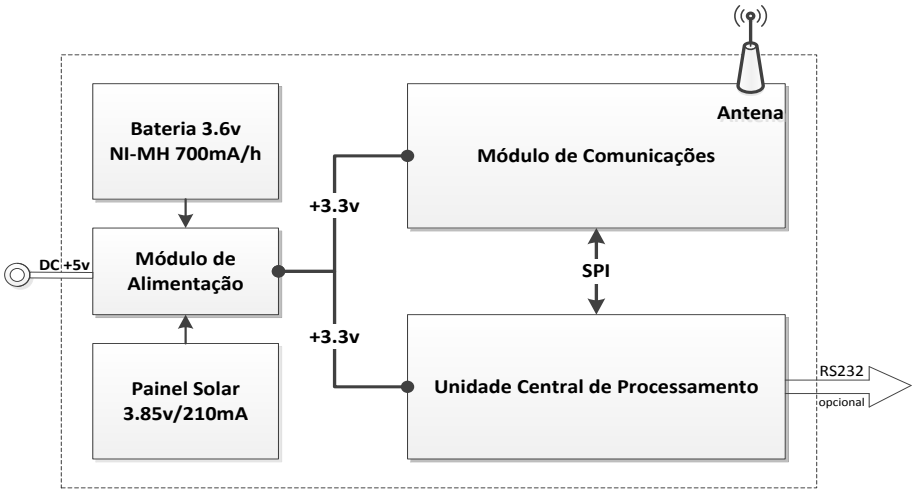



**Figura 33:** Diagrama funcional do nó *router*.

### 6.1.1 Módulo de Alimentação

Um dos requisitos para o nó sensor funcionar é ser autossuficiente energeticamente, pois num ambiente de uma estufa não é fácil disponibilizar uma fonte de energia externa para alimentar cada nó de rede. Assim sendo, utilizou-se um painel solar em conjunto com uma bateria recarregável para alimentar o sistema. A bateria recarregável Varta é constituída por 3 células de *Nickel Metal Hydride*, disponibiliza uma tensão de $3{,}6\,\mathrm{V}$ e tem uma capacidade de $700\,\mathrm{mAh}$. O painel solar tem uma capacidade de produção de $800\,\mathrm{mW}$, uma intensidade de corrente de $210\,\mathrm{mA}$ e $3.85\,\mathrm{V}$ de tensão de saída com carga.

O painel solar fornece energia a todo o sistema durante o dia ao mesmo tempo que carrega a bateria. Durante a noite, como não existe energia solar o nó sensor é alimentado com a energia contida na bateria. Por vezes, mesmo durante o dia poderá ser necessário utilizar-se a energia da bateria para alimentar o nó devido à baixa produção do painel solar.

As baterias do tipo NI-MH são de "carga lenta", ou seja, quando o painel solar está a produzir energia ela não pode ser canalizada diretamente para a bateria, pois existe um fator chamado C/10 que indica qual a corrente máxima com que deve ser carregada a bateria. Neste caso, corrente para a carga da bateria adequada será de $\frac{1}{10} * 700\mathrm{mAh} = 70\ \mathrm{mA}$ ou menos. Esta limitação deve-se ao ciclo de carga que este tipo de baterias tem, uma corrente superior cria aquecimento e provoca danos. O controlo de carga é feito pelo regulador de tensão LM317.

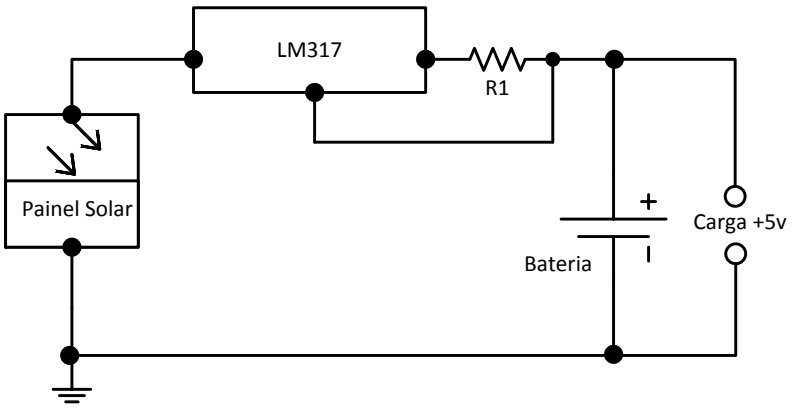



**Figura 34:** Circuito de alimentação com bateria sem carga acoplada.

O valor da resistência R1 é ajustável consoante o valor da tensão e da corrente que se pretende à saída do regulador de tensão (LM317). Em particular, nesta caso o indicado é de $R1 = 18\ \Omega$ que fornece uma corrente de $I = 69.4$ mA.

Inicialmente utilizou-se uma bateria com a capacidade de 150 mAh, no entanto como o consumo total do sistema é de $CT = 23$ mA, a autonomia seria apenas de $\frac{150\text{mAh}}{23\text{mA}} = 6.5\text{horas}$. Este valor torna-se insuficiente, pois o período da noite é sempre superior a 6 horas e meia. Assim, uma bateria 700 mAh já permite alimentar o sistema durante 30.4 horas teoricamente.

### 6.1.2 Módulo de Comunicação

A comunicação com o exterior do nó é feita com um transmissor de radiofrequência MRF24J40 da *Microchip*. Este módulo opera na banda ISM dos 2.4GHz e aplica a camada física e ligação do *standard* IEEE 802.15.4. Permite comunicação com um alcance sem antena até 200 metros no exterior, a uma velocidade de 250kbps. Suporta diversos protocolos de rede, ZigBee, MiWi, MiWi P2P, 6LoWPAN, sendo utilizado o ZigBee nesta aplicação. A comunicação com o microcontrolador é feita por 4 fios utilizando o protocolo SPI. As principais características que levaram à sua utilização são:

- Transmissor e recetor no mesmo dispositivo
- Baixo consumo de energia
    - Receção RX: 19 mA
    - Transmissão TX: 23 mA
    - *Sleep mode*: 2 µA
- É alimentado a 3.3 V
- Entrada e saída diferencial
- Gama de transmissão 36 dB
- Comunicação por SPI

Nas ligações ponto a ponto, os transmissores estão normalmente com a parte do recetor RX ligada à espera de receber uma emissão de um dispositivo. No caso do nó *router* quando recebe uma transmissão ele processa-a e depois envia para outro nó *router*. Existe continuamente um consumo efetivo de energia que varia entre os 23 mA os 19 mA conforme o sistema transmite ou recebe, respetivamente, pois a qualquer

momento poderá existir uma transmissão e assim o sistema não pode entrar no modo *Sleep.*

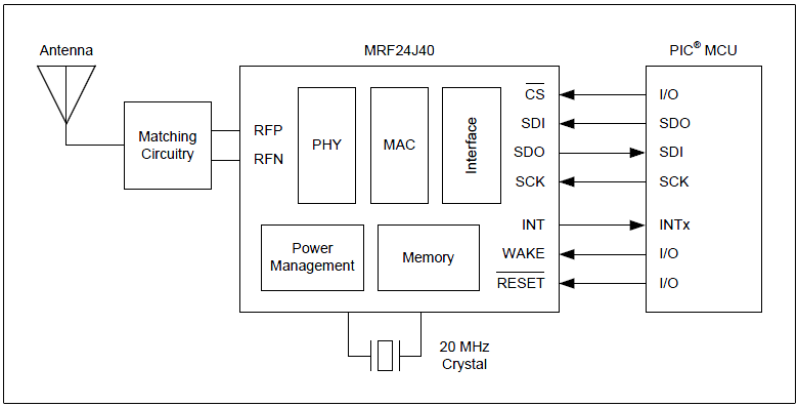



**Figura 35:** Diagrama de blocos do módulo MRF24J40 com ligação ao microcontrolador.

### 6.1.3 Módulo de Processamento

A evolução dos microcontroladores permite hoje em dia ter dispositivos com uma dimensão muito reduzida mas com grandes capacidades de memória e processamento. A escolha do microcontrolador recaiu sobre a relação qualidade preço, memória, processamento, consumo de energia, pelas ferramentas de apoio e facilidade no desenvolvimento. O microcontrolador escolhido para integrar o nó do sistema de monitorização é da família PIC o 16F1784 da *Microchip*. As principais características são:

- CPU de alto rendimento com arquitetura RISC
- 32 MHz de velocidade
- 125ns por ciclo de instrução
- Memória de 4096 *Words*
- Baixo consumo de energia
    - *Sleep mode* 50 *nA*
    - 1.3mA @ 3,3 V
- ADC com resolução de 12 bit e DAC de 8 bits
- 36 Entradas ou Saídas
- I2C, SPI
- EUSART
- 8/16bit Timers
- PWM

- 40 pinos

O esquema da Figura 36 ilustra o nó *router* implementado com o módulo de alimentação e o módulo de comunicações. O esquema representa todas as ligações físicas existentes. A comunicação entre o microcontrolador e o transmissor é feita através do protocolo SPI.

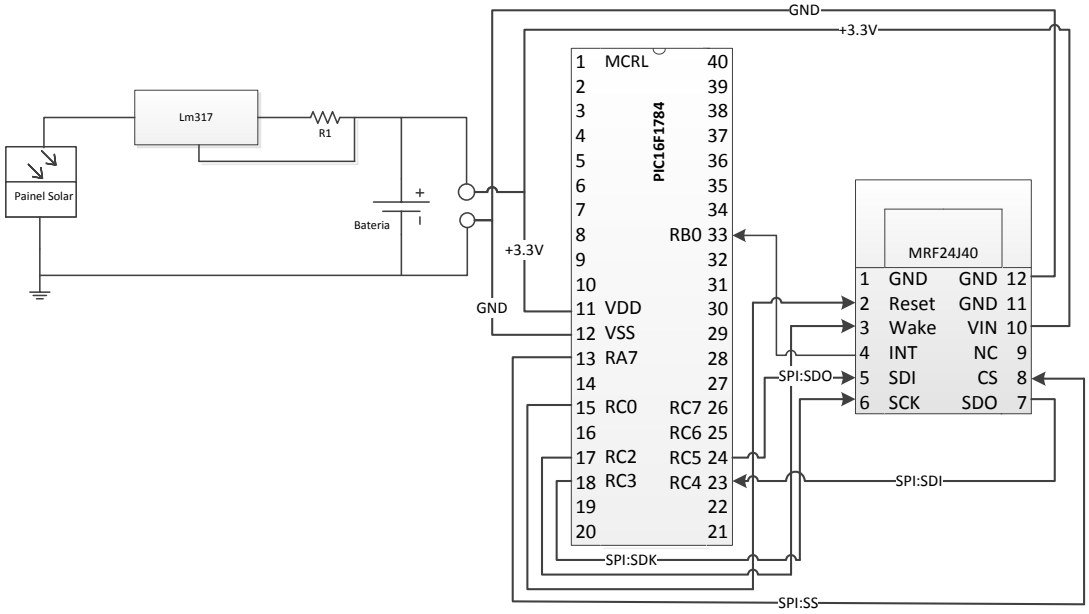



**Figura 36:** Esquemático do nó *router* em conjunto com o módulo de alimentação e comunicação.

O microcontrolador tem a função de gerir todo o processo de receção e transmissão dos dados. Sendo um nó *router* é responsável também pela criação de um ID de rede único no seu alcance onde se integram os nós clientes. Cada nó *end-device* ao enviar um bloco de dados especifica qual o ID de rede e também endereço para onde envia os dados. Existem 3 modos diferentes para a receção de dados, são eles: o modo Normal, *Error* e *Promiscuous*. Diferem entre si na forma como aceitam os pacotes enviados, ou seja, se recebem pacotes com bom ou mau CRC (Cyclic Redundancy Check). Os passos para a receção de um pacote são os seguintes:

1. Recebe a interrupção RXIF
2. Limpar a interrupção no microcontrolador.
3. Colocar RXDECINV = 1; Desativa a receção de pacotes.
4. Ler o endereço, 0x300; Receber o tamanho dos dados RXFIFO.
5. Ler o endereço RXFIFO, 0x301 através (0x300 + Frame Length + 2);
6. Limpar RXDECINV = 0; Ativa a receção de pacotes

7. Ativa a interrupção no microcontrolador.

Na transmissão para além de se especificar qual o ID de rede e endereço para onde irão os dados, é necessário formatar o pacote que será enviado como mostra a Figura 37.

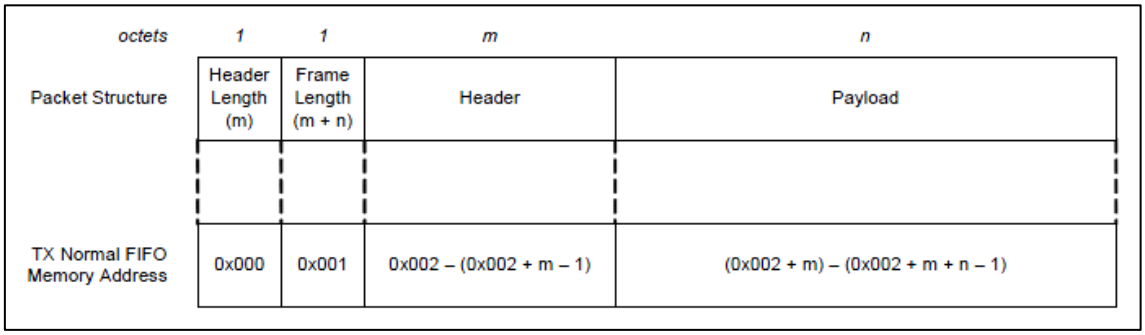



**Figura 37:** Formato da *frame* com os campos de dados a enviar.

A confirmação para o envio dos dados ter ocorrido com sucesso é feita através de uma função interna do transmissor MRF24J40, *Acknowledgement*, que envia um sinal do recetor para o transmissor a indicar a receção num certo período de tempo. Caso o transmissor não receba a confirmação, reenvia novamente os dados.

## 6.2 Nó *end-device*

O nó *end-device* tem como função adquirir valor sobre determinadas variáveis físicas através de sensores, processar a informação recolhida e enviar os dados para um nó *router* que esteja ao seu alcance. Este dispositivo passa grande parte do tempo em modo *sleep* (a dormir), só acorda em determinados intervalos de tempo predefinidos para recolher informação dos sensores, envia-la e depois volta novamente ao modo *sleep*. Dependendo do tipo de variáveis que esteja a monitorizar, em particular neste caso a temperatura, humidade, luminosidade são variáveis que não alteram o seu valor minuto a minuto, ou de cinco em cinco minutos. Otimizando o intervalo de tempo em que o nó tem de acordar faz com que exista uma grande poupança de energia.

O nó *end-device* comparativamente com o nó *router* na sua estrutura base possui mais um módulo de interface de sensores, os restantes alimentação, comunicação e processamento mantêm-se.

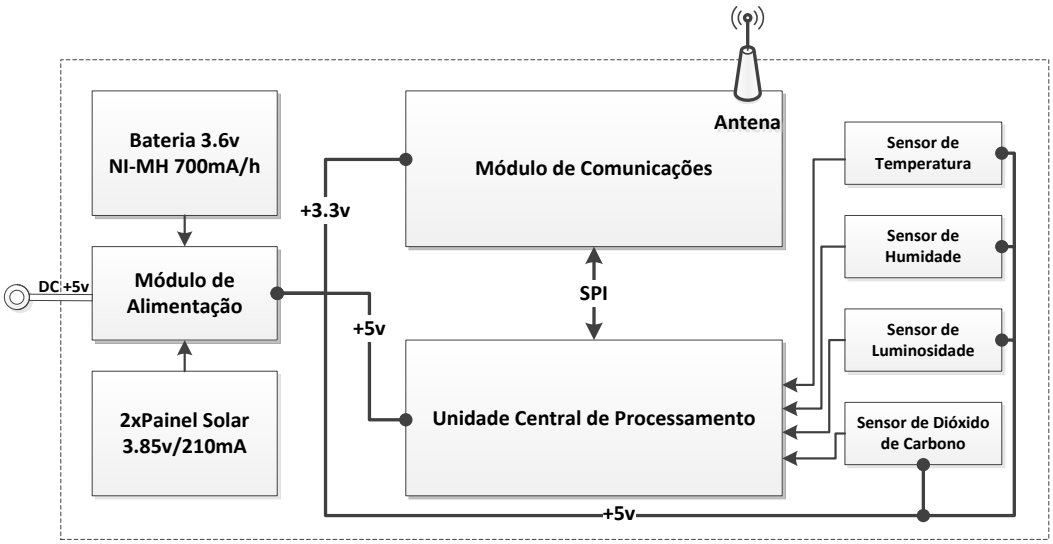



**Figura 38:** Diagrama funcional do nó *end-device*.

### 6.2.1 Módulo de Alimentação

A alimentação do nó *end-device* é ligeiramente diferente da representada no ponto 6.1.1 Módulo de Alimentação para no *router*. Os sensores adotados necessitam de uma alimentação a $5v$, a entrada da ADC do microcontrolador tem de ser regulada entre $0v$ e $5v$ para conseguir captar o sinal corretamente, dessa forma o microcontrolador é alimentado também a $5v$. No caso do módulo de comunicações, MRF24J40, a voltagem máxima em funcionamento permitida é de $3.6v$, sendo o ideal $3.3v$.

O esquema de energia mantém-se o mesmo, ou seja, o painel solar é colocado em série com outro da mesma capacidade ($3.85v$) para se obter uma voltagem superior aos $5v$. Da mesma forma colocou-se duas baterias de $3.6v$ em série para alimentar o circuito quando não há produção de energia solar.

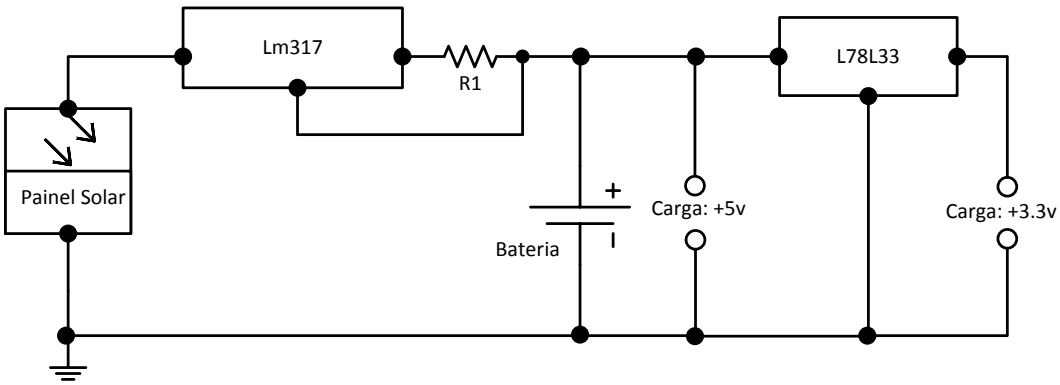



**Figura 39:** Circuito de alimentação com bateria sem cargas acopladas.

A associação de baterias assim como de painéis solar em série duplica a voltagens e mantém a corrente. Neste caso, as duas baterias em série produzem uma voltagem de $7.2v$ e a corrente mantém-se nos $700mAh$. O valor de $R1$ será de $18\Omega$ de forma a obedecer às regras de carga. O circuito integrado L78L33 é um regulador de tensão para $3.3v$ a $100mA$, aplica-se nesta situação para conectar o módulo de comunicações. O circuito integrado LM317 também é um regulador de tensão e corrente, neste caso aplicar uma saída de $5v$.

### 6.2.2 Módulo de Comunicações

O módulo de comunicações utilizado no nó *end-device* é o mesmo que foi utilizado no nó *router* descrito no ponto 6.1.2, estamos a falar do transmissor da *Microchip* MRF24J40MA. O nó *end-device*, como já foi descrito, tem períodos em que está inativo porque não necessita de estar continuamente a adquirir valores dos sensores. Este fator permite que haja uma poupança de energia aumentando a durabilidade da bateria. O sistema ao estar em modo inativo, chamado *Sleep*, consome muito pouca energia e permite ficar ativo a qualquer momento, sem que seja necessário inicializar todo o sistema.

O transmissor utilizado contempla nas suas características dois modos *Sleep*, são eles *Timed Sleep Mode* e *Immediate Sleep and Wake Mode*. O primeiro modo utiliza um contador de tempo (timer) para acordar o sistema passado um certo período de tempo. Ou seja, se for configurado ficar ativo de 5 em 5 minutos, a cada 5 minutos o sistema fica ativo e depois volta ao modo inativo durante mais 5 minutos, e assim sucessivamente. O segundo modo, que foi o implementado, transfere a decisão de colocar o sistema em modo ativo ou inativo para o microcontrolador. No dispositivo existe um pino chamado WAKE que em conjunto com algumas instruções coloca o módulo em modo ativo e posteriormente em modo inativo. A lista de passos é a seguinte:

Configuração do pino WAKE:

1. Colocar o pino WAKE = low
2. Enviar a configuração para ativar o pino WAKE e colocar a polaridade a High RXFLUSH (0x0D) = 0x60
3. Enviar a configuração para ativar o modo *Immediate Wake-up mode* WAKECON (0x22) = 0x80

Colocar o dispositivo em modo *Sleep*-inativo:

1. SOFTRST (0x2A) = 0x04 – Perform a Power Management Reset
2. SLPACK (0x35) = 0x80 – Coloca o MRF24J40 em modo inativo

Colocar o dispositivo em modo *Wake*-ativo:

1. WAKE pin = high – Wake-up
2. RFCTL (0x36) = 0x04 - RF State Machine reset
3. RFCTL (0x36) = 0x00
4. Colocar um atraso (delay) de 2 ms para permitir estabilizar o oscilador de 20 MHz antes de receber ou transmitir dados.

### 6.2.3 Módulo de Processamento

O microcontrolador PIC 16F1784 da *Microchip* foi adotado para o nó *router* e também para o nó *end-device*. O modo de funcionamento entre os dois tipos de nós é diferente, contudo as características do microcontrolador suportam ambos. O nó *end-device*, como já foi dito funciona por certos períodos de tempo, apenas quando é necessário fazer uma leitura dos sensores e transmitir a informação recolhida. Semelhante ao transmissor, o microcontrolador possui um modo inativo ou *Sleep* para poupança de energia. A função que coloca o microcontrolador em modo *Sleep* chama-se *Watchdog Timer*. Esta opção desativa as funções do microcontrolador por um certo período de tempo máximo de 256s. Quando atinge o tempo o *Watchdog* para cumprir com as tarefas e depois volta ao modo inativo. Antes do microcontrolador entrar em modo *Sleep* é colocado o transmissor neste modo.

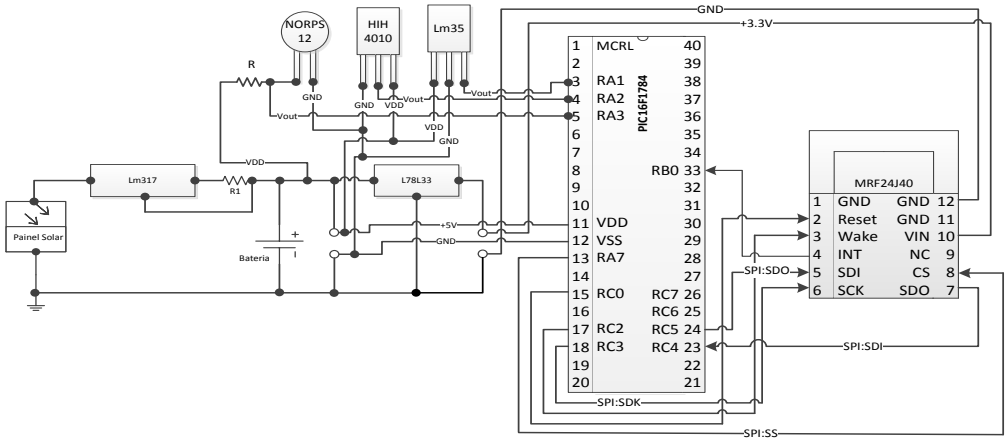



**Figura 40:** Esquemático do nó ***end-device*** com módulo de alimentação, comunicação processamento e sensores.

O sensor LM35 da *Texas Instruments* mede temperatura em graus Celsius, com uma linearidade de $+10\,\mathrm{mV/°C}$. Possui uma saída em tensão para uma gama de temperatura entre $-55\,°\mathrm{C}$ e $+150\,°\mathrm{C}$ com uma precisão de $\pm 0.5\,°\mathrm{C}$. Opera entre 4 V e 30 V, consome cerca de 60 µA. Pode ser ligado diretamente à ADC do microcontrolador sem a necessidade de circuito de condicionamento de sinal adicional.

O sensor de humidade HIH-4010 da *Honeywell* apresenta uma gama de medição para a humidade relativa à temperatura ambiente ($25°C$) entre os 0% e os 100% desde que o ambiente não esteja condensado. Opera entre os 4 V e os 5,5 V, com um consumo em funcionamento aproximado de 200 µA. A precisão é de ±3.5%. Ligando o sensor diretamente à ADC do microcontrolador obtém-se o valor da humidade relativa através da expressão (1), sendo $Vout$ o valor em volts à saída do sensor, $Vfonte$ a tensão da fonte que alimenta o sensor e $sensor\ RH$ o valor da humidade relativa em %.

$$V_{OUT} = \left(V_{fonte}\right)(0.0062(sensor\ RH) + 0.16) \quad (1)$$

O sensor de luminosidade utilizado trata-se do LDR NORPS12 da RS. É composto por duas células fotocondutivas de sulfito de cadmio. A gama de medição vai desde 10 Lux a 1000 Lux, sendo que cada Lux corresponde à incidência perpendicular de um *Lumen*. Por se tratar de uma resistência, para se ligar o LDR à ADC do microcontrolador é necessário utilizar um divisor de tensão para que a variação da resistência seja traduzida na variação da tensão. A resposta do sensor à intensidade luminosa é dada pela expressão (2), em que a variável independente $x$ representa o valor da resistência e a variável depende $y$ representa o valor da luminosidade em Lux.

$$y = 44687x^{-0,822} \quad (2)$$

## 6.3 Computador Central

O nó de rede do tipo *coordenador* está mais perto fisicamente do computador central que apresenta os dados ao agricultor, deve receber as comunicações dos dispositivos *end-device* ou *routers* e transmiti-las ao computador numa ligação série.

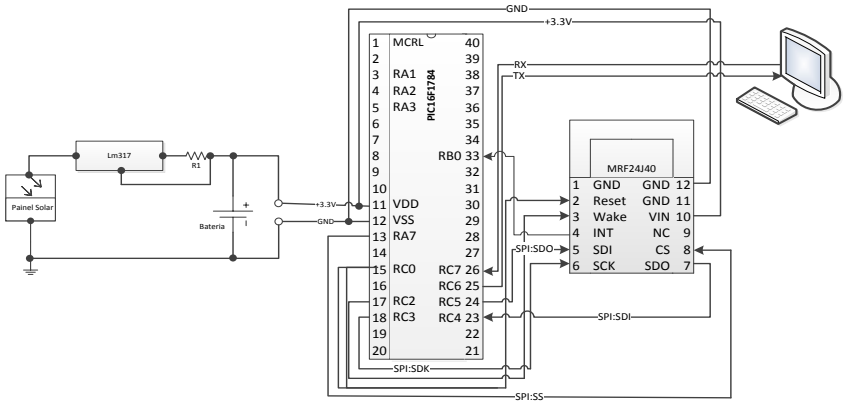



**Figura 41:** Esquemático de nó *router* com ligação ao computador central.

O computador recebe os dados, identifica qual o tipo de variável física que recebeu, se é temperatura, se é humidade, ou se é luminosidade, e guarda-a numa base de dados. Uma aplicação vai à base de dados, lê os valores registados e apresenta o valor da última leitura para cada variável. O *software* apresenta um gráfico com o histórico de todas as leituras registadas para cada variável, relacionando a hora a que foi recebida a informação com o valor registado. Neste caso particular só dispomos apenas de um nó *end-device*. Numa aplicação real, existirão diversos dispositivos e cada um deles é identificável por um ID que é único na rede. Cada nó *end-device* ao enviar os dados recolhidos referentes às variáveis físicas, envia também o seu ID de rede. Sabendo o local onde está colocado o nó sensor referente a determinado ID, é possível organizar a informação por zonas.

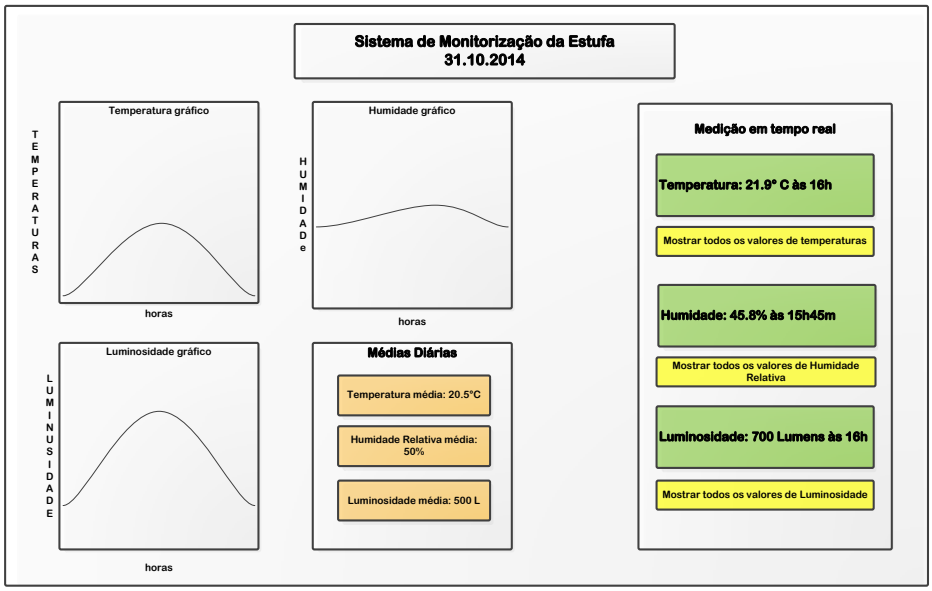



**Figura 42:** Esquema com aspeto gráfico do *software* de monitorização. Conteúdo meramente exemplificativo.

# 7. DESCRIÇÃO SISTEMA MONITORIZAÇÃO NO TANQUE

Neste capítulo descreve-se o sistema de monitorização utilizado para adquirir dados sobre as variáveis pH, temperatura, electro-condutividade e nível presentes no tanque que contém a solução nutritiva.

Num sistema hidropónico, os nutrientes necessários ao crescimento das plantas são quantificados e colocados num tanque juntamente com água. Nesse tanque gera-se assim uma solução de água e nutrientes que é depois canalizada para as raízes das plantas. Nalguns sistemas hidropónicos a solução que foi canalizada para as plantas e que não foi absorvida é reutilizada, ou seja, os excessos são canalizados novamente para o tanque principal de onde saíram. A parametrização dos valores de pH e electro condutividade indicam ao agricultor se a solução contida no tanque está em condições de ser canalizada para as plantas ou se necessita de ser ajustada.

O sistema proposto neste ponto tem como objetivo recolher valores de pH e electro condutividade ao longo do tempo e disponibiliza-los ao agricultor. O âmbito destas variáveis físicas não possibilita o conhecimento de qual nutriente está em falta na solução, para isso seria necessário colocar um sensor para cada nutriente utilizado na dieta das plantas. O controlo da temperatura da solução e dos níveis de água no tanque também são contemplados nesta estação de controlo.

## 7.1 Sensor de Nível

O sensor de nível 519-242 da RS permite controlar o nível de água no tanque gerando um sinal do tipo ON ou OFF conforme é atingido o limite máximo estipulado. Este sensor é constituído por uma boia flutuante que contém um íman no seu seio que ao deslocar-se funciona para um interruptor ligando ou desligando um circuito. Quando a boia flutua até atingir o topo do cilindro o circuito é ligado, existe a condução do sinal elétrico de uma extremidade para a outra. Um dos terminais do sensor é ligado a um microcontrolador que gera uma interrupção quando deteta no pino de entrada um sinal a HIGH representativo do circuito em modo ON. Seguidamente gera-se uma mensagem de alerta enviada para o agricultor ao mesmo tempo que se liga um LED. Quando o nível de água volta ao normal, a boia desce e consequentemente desliga o circuito (OFF). O esquema da Figura 43 apresentação o diagrama de ligações do sensor.

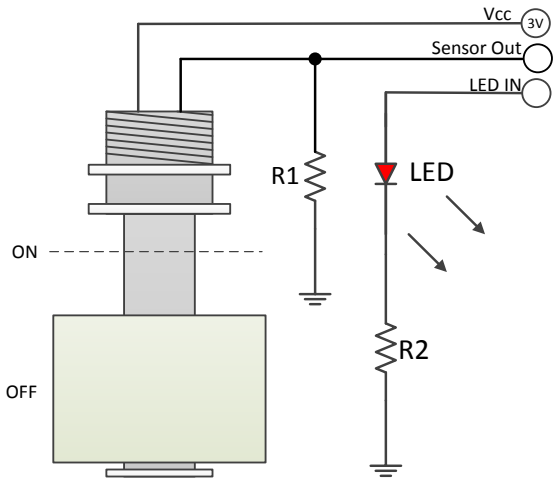



**Figura 43:** Diagrama de ligações sensor de nível magnético.

A colocação de vários sensores no tanque ou tanques a níveis correspondentes a diferentes alturas permite determinar quantidades de água que poderão ser úteis nomeadamente para a criação da solução nutritiva.

## 7.2 Sensor de Condutividade Elétrica

O sensor de condutividade elétrica LFS1K0.155.6W.B.010-6 da IST incorpora no seu seio um sensor de temperatura resistivo de platina $PT1000$. O sensor de temperatura embutido no mesmo package que os elétrodos $Al_2O_3$, sensíveis à condução em meios aquosos, permite que os valores adquiridos pelo sensor sejam ajustados mediante a temperatura. Nos eletrólitos, a condução elétrica não acontece pelo movimento dos eletrões para as lacunas mas sim pela viagem de espécies atómicas chamadas iões. Cada ião transporta uma carga elétrica. A resistividade dos líquidos iónicos varia muito com a concentração, a água destilada funciona quase como um isolante e em contra partida, a água salgada é um condutor elétrico muito eficiente. No caso da hidroponia pretende-se que a solução nutritiva seja supervisionada pelos valores recolhidos de condutividade elétrica.

Um dos principais desafios na incorporação de um sensor de condutividade elétrica com elétrodos num projeto é a necessidade de utilizar uma fonte de corrente alternada (AC) para efetuar medições. Outro desafio encontrado é a necessidade de amplificação do sinal resultante da diferença de potencial (V) entre os dois elétrodos, por ser na ordem das dezenas de milivolts ou até menor.

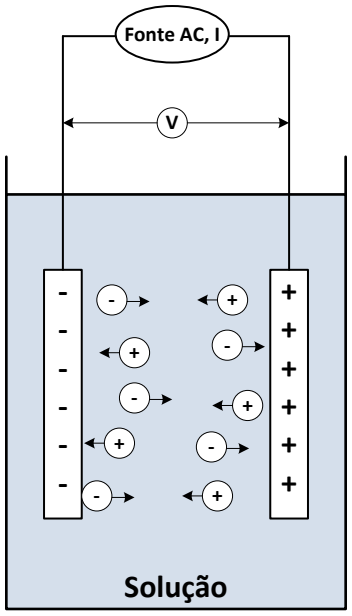



**Figura 44:** Esquema de medição do sensor de condutividade elétrica.

A interligação deste tipo sensor com um microcontrolador, responsável pela recolha dos dados e posterior envio para o computador central, não é direta e divide-se em 3 passos são eles, a geração do sinal AC, amplificação e medição da diferença de potencial, neste último caso já em corrente continua DC. Em primeiro lugar é necessário gerar um sinal elétrico que varia ao longo do tempo, AC. Para isso, necessita-se um gerador de sinais ou uma fonte de corrente alternada.

A plataforma utilizada para a ligação dos sensores no sistema de monitorização no tanque é o *kit* de desenvolvimento da *STMicroelectronics* o STM32F4 *Discovery* (STMicroelectronics, 2014). O microcontrolador embutido neste *kit* permite a geração de um sinal AC, variante no tempo de duas formas, utilizando a função PWM ou DAC, *Pulse Width Modulation* ou *Digital Analogue Converter*, respetivamente. O método utilizado foi o segundo, ou seja, através da DAC - Conversor Digital para Analógico do microcontrolador gerou-se uma sinusoide com frequência de 500 Hz. A frequência de medição do sensor situa-se entre os 300 Hz e os 3000 Hz. Para a construção da onda de saída, antes de se programar o microcontrolador, utilizou-se o *Matlab* para construir a função com um certo número de pontos de amostragem e obter-se os respetivos valores num intervalo definido, neste caso $2\pi$. Com os valores do *plot* do gráfico da sinusoide no *maltlab* (código descrito abaixo), obtém-se o sinal AC conjugando um Timer e a DAC do microcontrolador. (Ostrikov, 2014)

```
%% Generating an n-bit sine wave
% Modifiable parameters: step, bits, offset
```

```
close; clear; clc;

points = 128;           % number of points between sin(0) to sin(2*pi)
bits   = 12;            % 12-bit sine wave for 12-bit DAC
offset = 75;            % limiting DAC output voltage

t = 0:((2*pi/(points-1))):(2*pi);  % creating a vector from 0 to 2*pi
y = sin(t);                        % getting the sine values
y = y + 1;          % getting rid of negative values (shifting up by 1)
y = y*((2^bits-1)-2*offset)/2+offset; % limiting the range (0+offset)
to (2^bits-offset)
y = round(y);           % rounding the values
plot(t, y); grid        % plotting for visual confirmation

fprintf('%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, \n', y);
```

Posto isto, é necessário converter o sinal de saída do µC numa gama abaixo dos 1.6 Vpp, dado que é o limite máximo definido pelo fabricante do sensor. A integração do sensor é feita a partir do circuito apresentado na Figura 46. A primeira parte (superior) trata-se de um filtro passa-baixo de 2ª ordem (Rauch) que converte o sinal de entrada ($XFG1$) numa onda $\pm$ 200 mV ($XMM1$). Seguidamente, onde se insere o sensor, o sinal é canalizado para um amplificador de alta impedância ($XMM2$). Por fim, utiliza-se um detetor de pico para transformar o sinal sinusoidal num sinal contínuo ($XMM3$). A saída do detetor de pico é ligada diretamente à ADC do microcontrolador, onde ocorrerá a conversão do sinal num valor digital e posteriormente em siemens (unidade SI para condutividade elétrica).

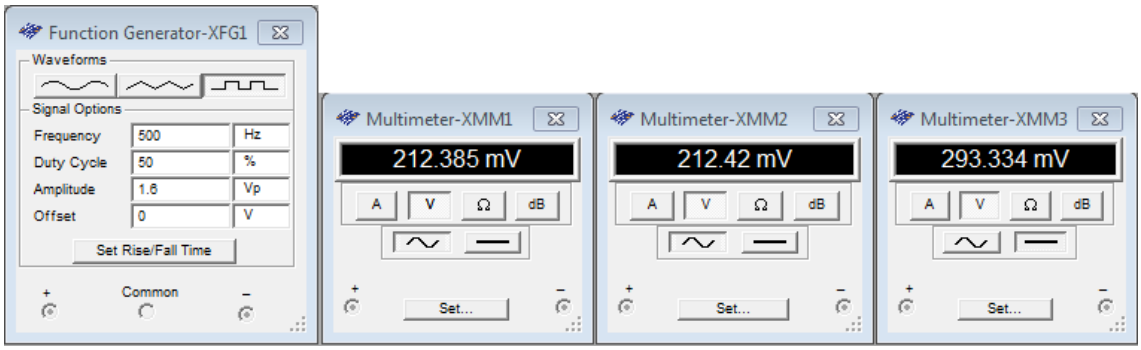



**Figura 45:** Gerador de Funções e multímetros com resultado da simulação em *Multisim* 10.

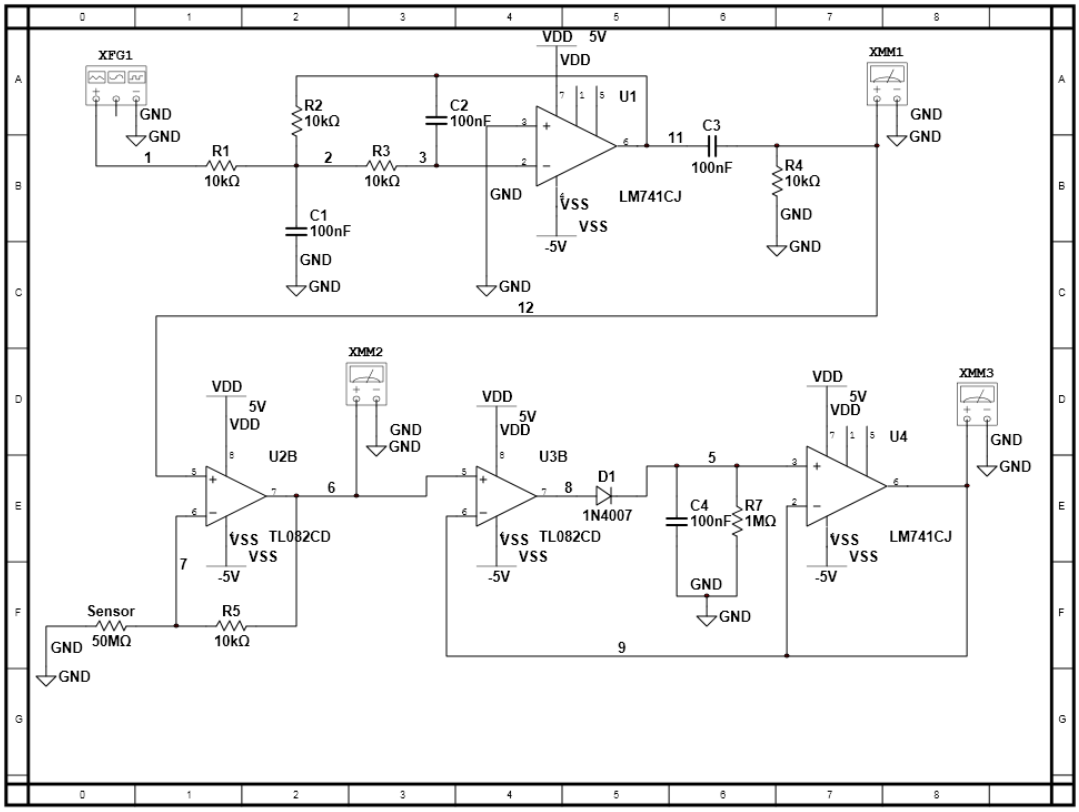



**Figura 46:** Circuito para integração do sensor de condutividade elétrica composto por um filtro passa baixo de 2ª ordem, um amplificador de alta impedância e um detetor de pico (simulação em Multisim 10).

As medições feitas a partir do sensor necessitam de ser calibradas antes de se proceder à utilização do sensor. Para se efetuar a calibração do sensor utilizam-se soluções com um valor de condutividade elétrica conhecido e depois faz-se algoritmicamente o ajuste. Neste caso a calibração não foi efetuada, apenas se fizeram medições para duas soluções diferentes, uma constituída por água natural engarrafada e outra contendo água e sal. Os resultados comparativos apresentam-se na seção 8.

## 7.3 Sensor de Temperatura – PT1000

Como já foi descrito no ponto anterior, 7.2 Sensor de Condutividade Elétrica, o sensor de condutividade elétrica incorpora um RTD do tipo PT1000. Foram separados em dois tópicos, pois apesar do sensor de temperatura ser utilizado para fazer compensações nas medições de condutividade elétrica, ele também pode ser usado independentemente para medir a temperatura do líquido onde está inserido. Os detetores de temperatura resistivos, neste caso o PT1000 apresenta à temperatura de 0 °C o valor resistivo de 1000 Ω ou 1 kΩ.

A medição de temperatura com o $PT1000$ é feita através da ADC do microcontrolador, bastando para isso inserir o sensor num divisor de tensão e determinar a expressão que representa a resposta em função da temperatura.

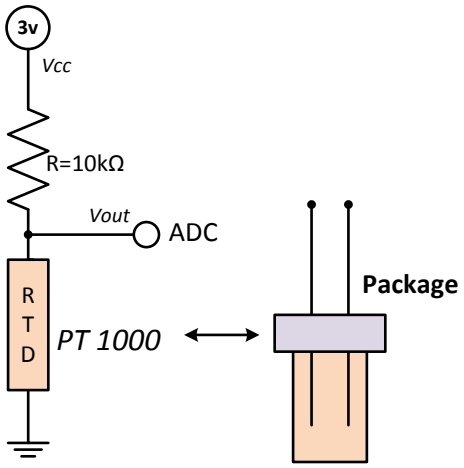



**Figura 47:** Esquema do circuito para ligação do sensor de temperatura resistivo *PT1000* à ADC do µC.

A expressão (3) apresenta o valor em graus celsius da temperatura em função da resistência do sensor.

$$T(°C) = (0.2592 * RTD) - 259.26 \quad (3)$$

A relação entre o valor atribuído pela ADC ao nível de voltagem que recebe à saída do divisor de tensão e o valor da resistência do RTD é dado pela expressão.

$$RTD(\Omega) = \frac{\frac{Vout}{Vcc} * 10000}{1 - \frac{Vout}{Vcc}} \quad (4)$$

## 7.4 Sensor de pH

A Jumo Instruments apresenta sensores de pH com uma boa relação qualidade preço. O sensor da linha *Black* possui um revestimento fixo em plástico PPO (polyphenylene oxide) e permite medições de pH entre os 0 e os 12. A resposta do sensor aos valores de pH, segundo o fabricante, é de cerca de $-59\ mV/pH$. A gama de medição vai desde os $-400\ mV$ aos $400\ mV$, o valor $0\ mV$ representa o nível $7pH$.

A interligação deste sensor com a ADC do microcontrolador descrito no ponto **7.2 Sensor de Condutividade Elétrica** não é direta, ou seja, como a gama de valores à saída da ponta de prova varia entre os $-400\ mV$ e os $400\ mV$ é necessário fazer um

*offset* do sinal para valores positivos e também amplifica-lo de forma a transformar o sinal em valores ideais para aquisição na ADC. Assim sendo, o sinal de saída varia entre os $0\,V$ e os $5\,V$, sendo que para o valor $7\,pH$ a saída é de $2{,}5\,V$. Na Figura 48 o multímetro $XMM1$ representa o valor em tensão (simulado) que saí do sensor de pH, neste caso 0v que representa $pH7$ e o multímetro $XMM3$ o valor de saída para entrar na ADC, $\sim 2.5v$.

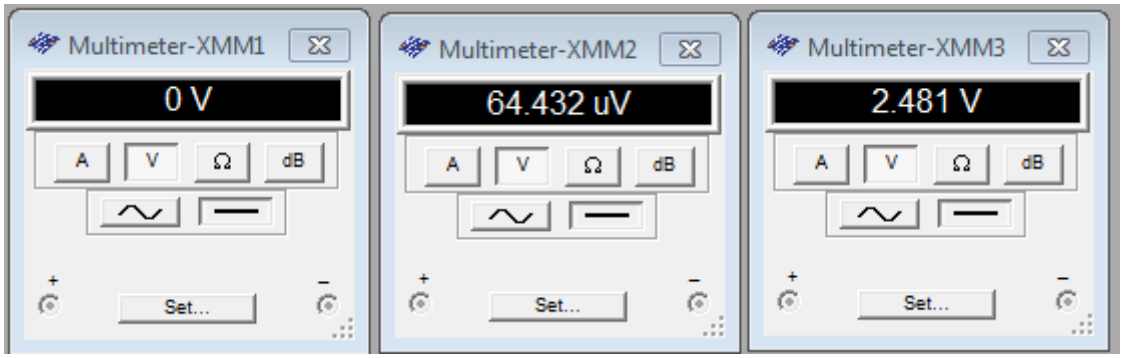



**Figura 48:** Tensão de saída em vários pontos do circuito de acondicionamento de sinal para o sensor pH, simulado no *Multisim 10*.

Na Figura 49 descreve-se o circuito para ligação do sensor de pH à ADC do microcontrolador. O sensor no caso real é substituído pela fonte de tensão no ponto 1 do esquema abaixo bem como a saída para a ADC é conectada no ponto 6.

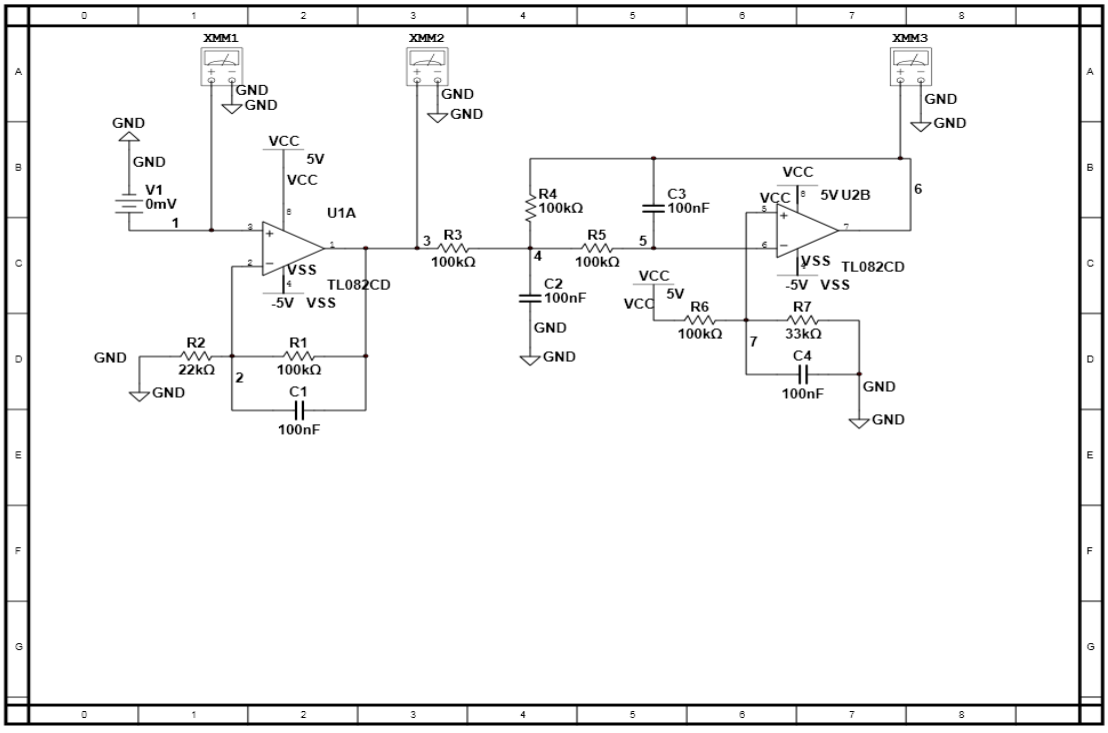



**Figura 49:** Circuito para integração do sensor de pH composto por um filtro passa baixo de 2ª ordem e um amplificador (simulação em Multisim 10).

O sinal resultante à saída do sensor de pH varia ao longo do tempo, pois as características dos elétrodos vão-se alterando. O caso de uma bateria química é uma boa analogia para este efeito, sabemos ao longo do tempo a tensão disponibilizada pela bateria é menor, vai enfraquecendo até eventualmente falhar. No caso dos elétrodos acontece o mesmo, uma vez que o sinal de saída varia para as mesmas mudanças de pH. Por isto é necessário recorrer a calibrações periódicas, de cada vez que o módulo que mede pH é calibrado atribui um novo conjunto de valores para os sinais de entrada.

Nesta utilização do sensor de pH não foi efetuada uma calibração, existirá um erro associado às medições experimentais desconhecido. Na secção 8.2 Sistema de monitorização no tanque apresentam-se os resultados obtidos.

# 8. RESULTADOS

Neste capítulo apresentam-se os principais resultados obtidos para a validação do sistema de monitorização do ambiente da estufa e do sistema de monitorização no tanque. Em relação aos nós de rede, serão descritos a nível energético, os consumos de cada nó, a produção do painel solar quando exposto ao sol e a autonomia do sistema em bateria. Noutro ponto, descreve-se os resultados obtidos com o sistema em funcionamento durante 24 horas apresentando os dados recolhidos e por fim o custo total do sistema.

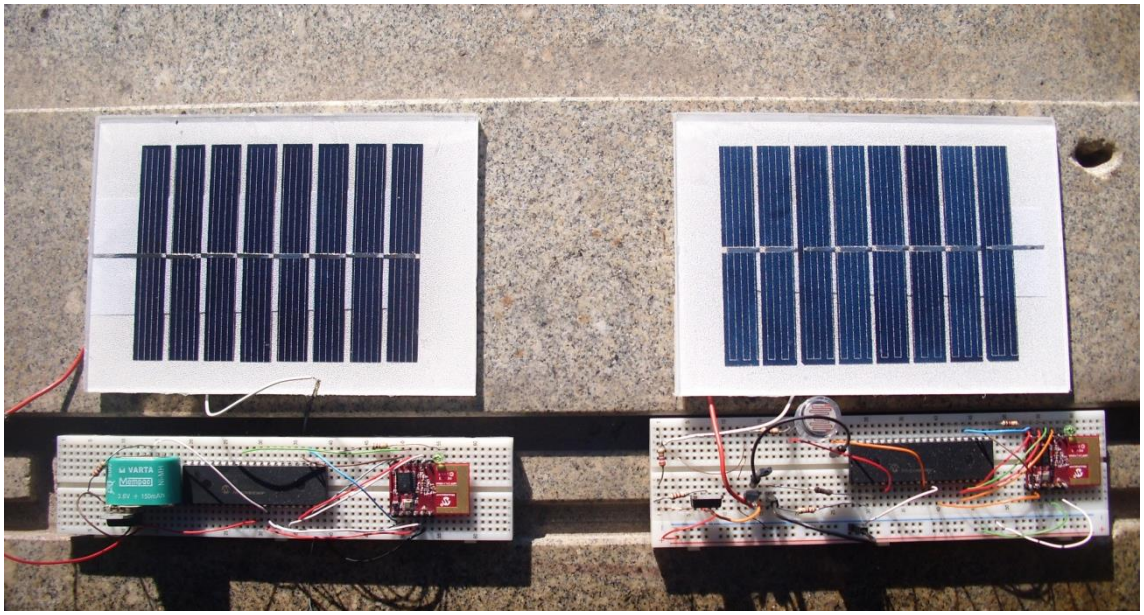



**Figura 50:** Nó *router e nó end-device.*

Em relação ao sistema de monitorização no tanque, apresenta-se no ponto 8.2 os dados recolhidos para a validação do sistema de aquisição, nomeadamente para o sensor de temperatura, electro-condutividade, pH e no final o custo da estação.

## 8.1 Rede de Sensores

### 8.1.1 Consumo energético

O consumo de energia de cada nó é condicionado por dois fatores, são eles o tempo em que passa no modo ativo e o consumo dos periféricos tal como os sensores. Para se medir o consumo de energia, colocou-se uma fonte de alimentação externa ligada ao circuito e aos seus terminais com um multímetro recolheram-se os valores de corrente e tensão, em série e em paralelo respetivamente. Para cada variável corrente e tensão, fizeram-se 10 medições independentes. Os resultados obtidos são os seguintes:

- Nó *end-device*

  O nó *end-device* funciona em dois modos, ativo e inativo para poupar energia quando não necessita de recolher dados. Para além do módulo de comunicações, tem também os sensores a consumir energia.

  - Modo ativo – w*ake up*
    - Tensão: 5,17 V
    - Corrente: 40 mA
  - Modo inativo – *sleep*
    - Tensão: 5,11 V
    - Corrente: 1,8 mA
  - Autonomia estimada só com bateria (700 mAh)
    - *Sleep mode*: 388 horas
    - *Wake up mode*: 17,5 horas

- Nó *Router*

  O nó *router* está continuamente em funcionamento à espera de ligações de outros nós. Isto leva a um gasto constante de energia para manter o sistema pronto para receber dados. Na transmissão existe um gasto maior de energia devido à comunicação série.

  - Modo ativo
    - Tensão média: 3,40 V
    - Corrente: 30 mA em transmissão e 22,1 mA em receção.
  - Autonomia aproximada só com bateria (700 mAh)
    - 23 horas

- Painel Solar

  Para determinar a potência máxima produzida pelo painel solar, colocou-se o mesmo no exterior em contacto direto com a luz solar sobre um ângulo aproximado de 30°. Com um multímetro obteve-se os valores para tensão e corrente.

    - Tensão: 4,37 $V$
    - Corrente: 70,1 $mA$

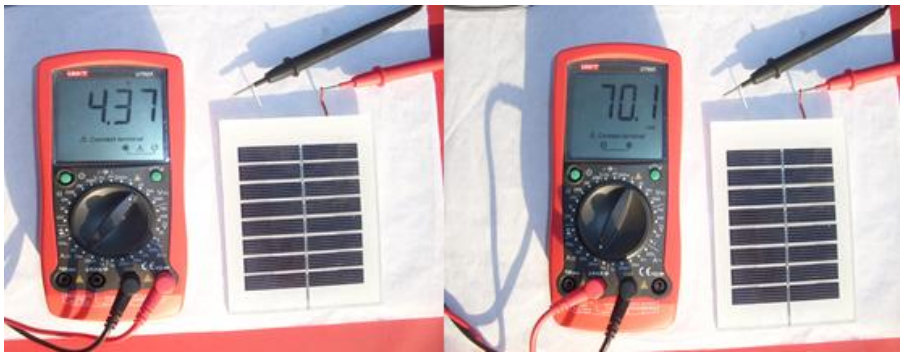



**Figura 51:** Potência disponibilizada pelo painel solar tensão (esquerda) e corrente (direita).

### 8.1.2 Monitorização de grandezas ambientais

O sistema de monitorização foi instalado no interior de uma varanda de uma casa onde se fizeram medições de temperatura, humidade relativa e luminosidade durante 24 horas. Os resultados obtidos são descritos na Tabela 6.

| Hora | Temperatura (°C) | Humidade (%) | Luminosidade (L) |
|---|---|---|---|
| **29-10-14 13:00** | 23,5 | 38,9 | 500 |
| **29-10-14 14:00** | 22,9 | 40,1 | 600 |
| **29-10-14 15:00** | 23,4 | 42,5 | 540 |
| **29-10-14 16:00** | 23,1 | 37,6 | 517 |
| **29-10-14 17:00** | 20,0 | 45,6 | 400 |
| **29-10-14 18:00** | 19,6 | 48,3 | 200 |
| **29-10-14 19:00** | 22,3 | 49,1 | 90 |
| **29-10-14 20:00** | 20,4 | 46,7 | 96 |
| **29-10-14 21:00** | 18,6 | 44,3 | 105 |
| **29-10-14 22:00** | 18,3 | 52,3 | 110 |
| **29-10-14 23:00** | 18,4 | 51,7 | 87 |
| **30-10-14 0:00** | 17,8 | 50,0 | 90 |
| **30-10-14 1:00** | 18,0 | 48,5 | 106 |
| **30-10-14 2:00** | 18,7 | 45,7 | 95 |
| **30-10-14 3:00** | 17,2 | 42,1 | 98 |
| **30-10-14 4:00** | 17,5 | 39,6 | 97 |
| **30-10-14 5:00** | 16,9 | 45,7 | 150 |
| **30-10-14 6:00** | 16,4 | 45,8 | 230 |
| **30-10-14 7:00** | 16,3 | 43,4 | 290 |
| **30-10-14 8:00** | 17,8 | 42,7 | 315 |
| **30-10-14 9:00** | 19,0 | 42,8 | 400 |
| **30-10-14 10:00** | 19,8 | 41,6 | 500 |
| **30-10-14 11:00** | 21,4 | 40,0 | 680 |
| **30-10-14 12:00** | 20,5 | 43,0 | 710 |
| **30-10-14 13:00** | 21,2 | 42,5 | 700 |

**Tabela 6:** Monitorização de temperatura, humidade e luminosidade numa varanda.

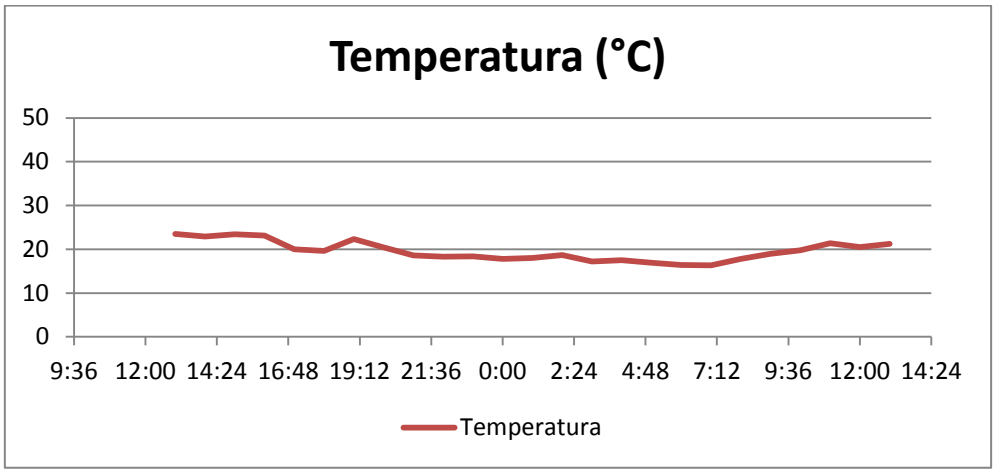



**Gráfico 1:** Medição temperatura em varanda.

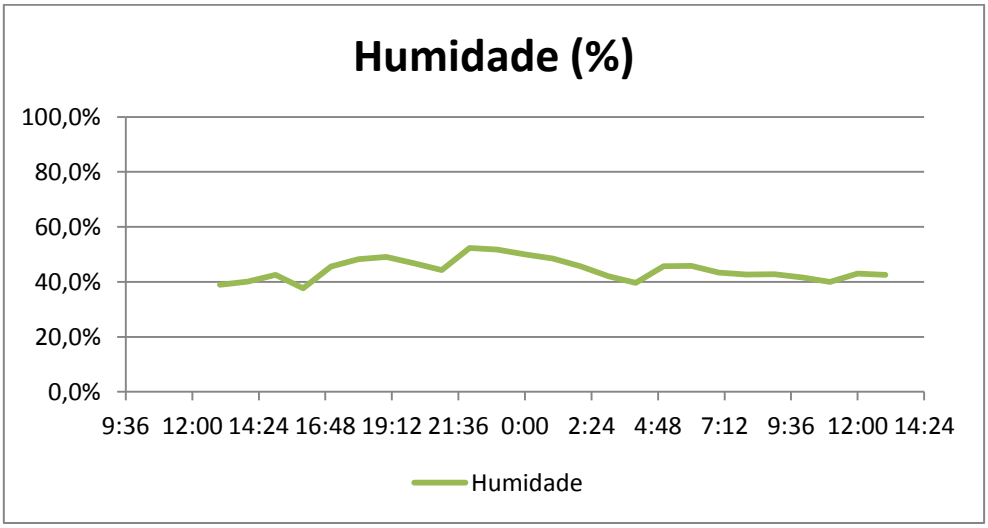



**Gráfico 2:** Medição de humidade relativa em varanda.

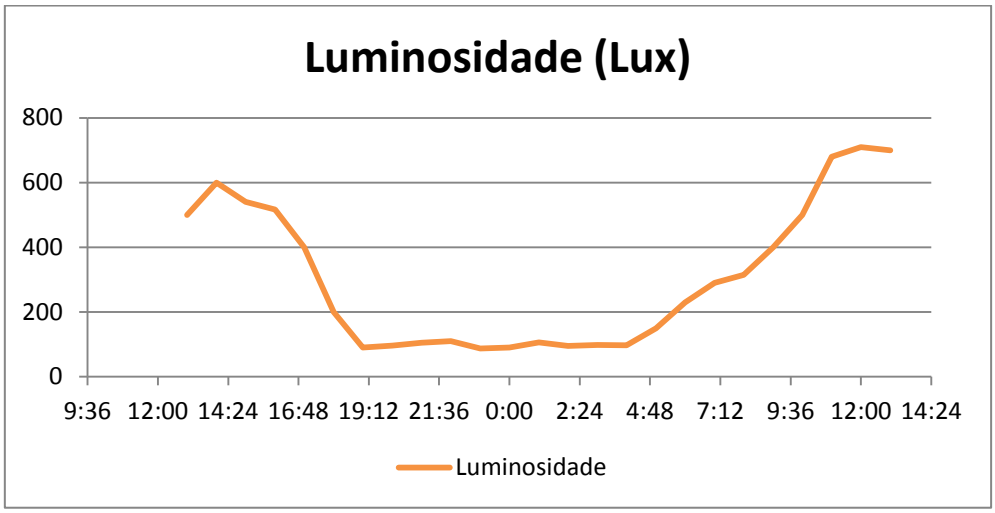



**Gráfico 3:** Medição da luminosidade em varanda.

Pode verificar no Gráfico 3 o período noturno onde a intensidade luminosa baixa abruptamente. No caso da temperatura e humidade as variações não foram tão acentuadas, em primeiro lugar porque nos encontramos num mês de outubro muito quente ao mesmo tempo que no interior da varanda é conservado o calor acumulado durante o dia, pela noite. Os valores de luminosidade para além de permitirem saber se existe muita ou pouca energia solar, quantificados e tabelados permitem fazer um gráfico com a estimativa da potência que o painel solar debita ao longo do dia.

A Figura 52 apresenta o aspeto gráfico do *software* de monitorização onde se verificar o registo ao longo do tempo das diversas variáveis monitorizadas.

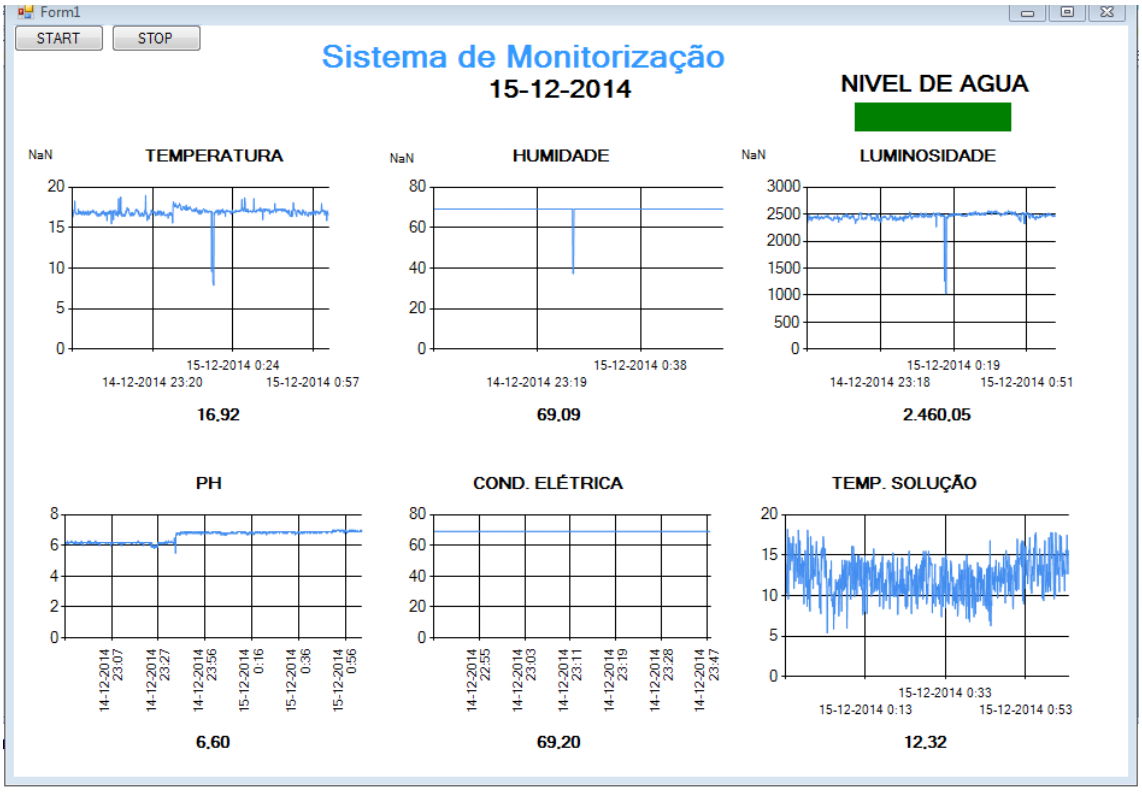



**Figura 52:** *Print-Screen* do *software* de monitorização.

### 8.1.3 Custo do sistema

Um dos requisitos do sistema de monitorização é o seu custo quem deve ser o menor quanto possível. Os diversos componentes adquiridos para a construção dos nós sensores se forem comprados em grandes quantidades aferem valores mais baixos. A seguinte lista apresenta os componentes e respetivo custo para os dois nós com sensores.

| Referência | Designação | Fabricante | Quantidade | Custo |
|---|---|---|---|---|
| **Lm35** | Sensor de temperatura | TI | 1 | 4,41 € |
| **HIH-4010** | Sensor de Humidade | HONEYWELL | 1 | 18,83 € |
| **NORPS-12** | Sensor de Luminosidade | RS | 1 | 3,85 € |
| **CO2** | TGS4161 | Fígaro | nc | - € |
| **PIC 16F1784** | Microcontrolador | Microchip | 2 | 2,35 € |
| **MRF24J40MA** | *Transceiver Wireless* | Microchip | 2 | 7,44 € |
| **SP0.9-NF-GCS** | Painel Solar | Multicomp | 2 | 7,79 € |
| **3/V 150 H SK SC TP** | Bateria | Varta | 2 | 6,60 € |
| **LM317** | Regulador Tensão | Fairchild | 2 | 0,46€ |
| **YR1B10KCC** | Resistência | TE Connectivity | 5 | 0,05€ |
| | | | **TOTAL** | 76,62 € |

**Tabela 7:** Lista de componentes utilizados na construção dos nós sensores com respetiva descrição e custo.

## 8.2 Sistema de monitorização no tanque

### 8.2.1 Sensor de Temperatura

O sensor de temperatura PT1000 utilizado no sistema de monitorização no tanque serve diretamente como referência para quantificar o valor da temperatura da solução nutritiva no tanque e indiretamente para compensação nas medidas de pH e condutividade-elétrica.

Para a validação do seu funcionamento, o sensor foi colocado num recipiente contendo água no estado líquido e ligado à ADC do microcontrolador. Os valores obtidos descritos no Gráfico 4 representam a temperatura da solução, que neste caso é constituída apenas por água.

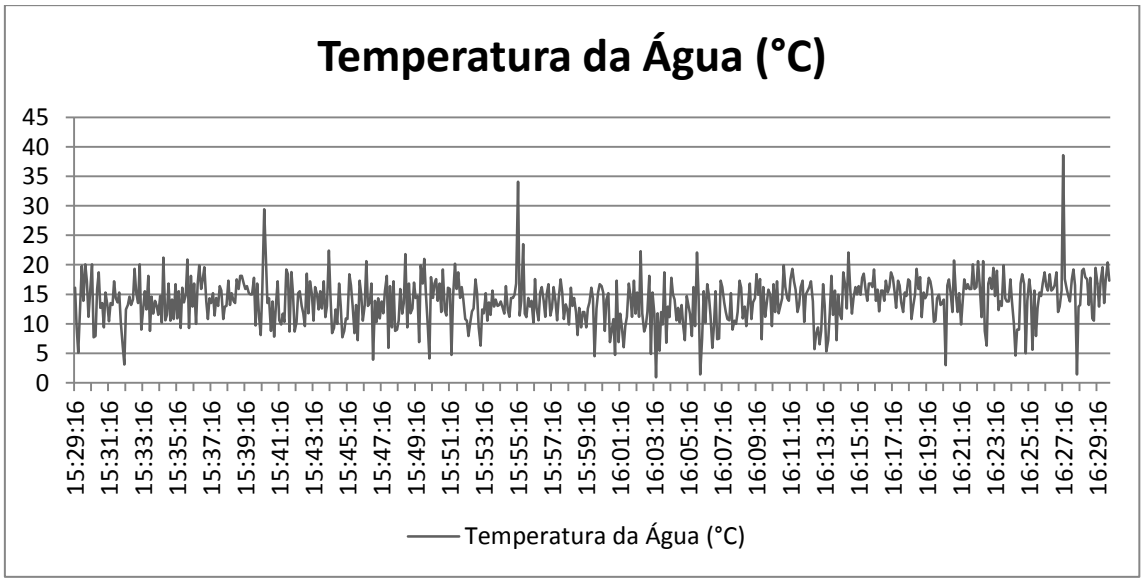



**Gráfico 4:** Valores obtidos na medição da temperatura da água utilizando o sensor PT1000.

Realizaram-se 609 medições num espaço temporal de 1 hora. A média obtida foi 13,83 °C.

### 8.2.2 Sensor de Condutividade Elétrica

O sensor de condutividade elétrica necessita de ser calibrado por forma apresentar, valores concisos sobre a capacidade que uma solução tem para transmitir corrente elétrica. Neste caso os resultados apresentados mostram apenas resposta do sensor a uma solução composta por água engarrafa. A calibração deste sensor só é possível utilizando uma solução com um valor conhecido, muitas vezes é o usado o valor $1413\mu S$. O sensor de temperatura usado no ponto anterior está inserido no mesmo package que este sensor, pois as medições de condutividade elétrica necessitam de ser compensadas pelas variações de temperatura da solução relativamente aos valores de referência previamente calibrados.

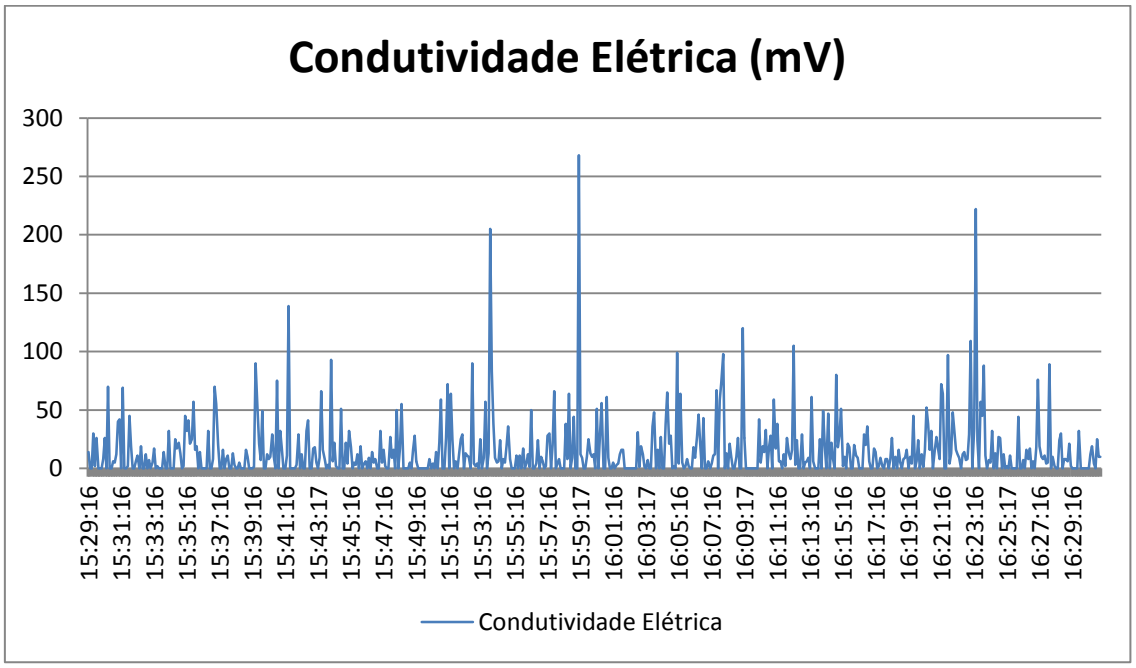



**Gráfico 5:** Valores obtidos na medição da Condutividade-Elétrica da água engarrafada.

Realizaram-se 609 medições num espaço temporal de 1 hora. A média obtida foi 15,11 mV. Os valores de condutividade elétrica são compensados devido à temperatura através da expressão (5), onde $C_t$ representa o valor obtido pelo sensor, $C_{25^\circ}$o valor de referência, t a temperatura da solução medida e $\propto$ o coeficiente de temperatura linear que varia entre 0% e 5%. Ou seja, para uma solução diluída $\propto$=2%, para soluções ácidas é valor é menor e para as soluções básicas maior.

$$C_{25°C} = \frac{C_t}{1+\propto(t-25)} \quad (5)$$

### 8.2.3 Sensor de pH

O sensor de pH foi utilizado conforme chegou do produtor, não foi realizada qualquer ação de calibração. Como já foi referido anteriormente, após várias utilizações os elétrodos sofrem alterações e necessitam de ser calibrados. Neste caso, considerando que o sensor está calibrado fizeram-se 5 medições de pH para $50ml$ de água engarrafada. Na composição característica da água refere-se que o valor de pH a $18°C$ é $6pH$.

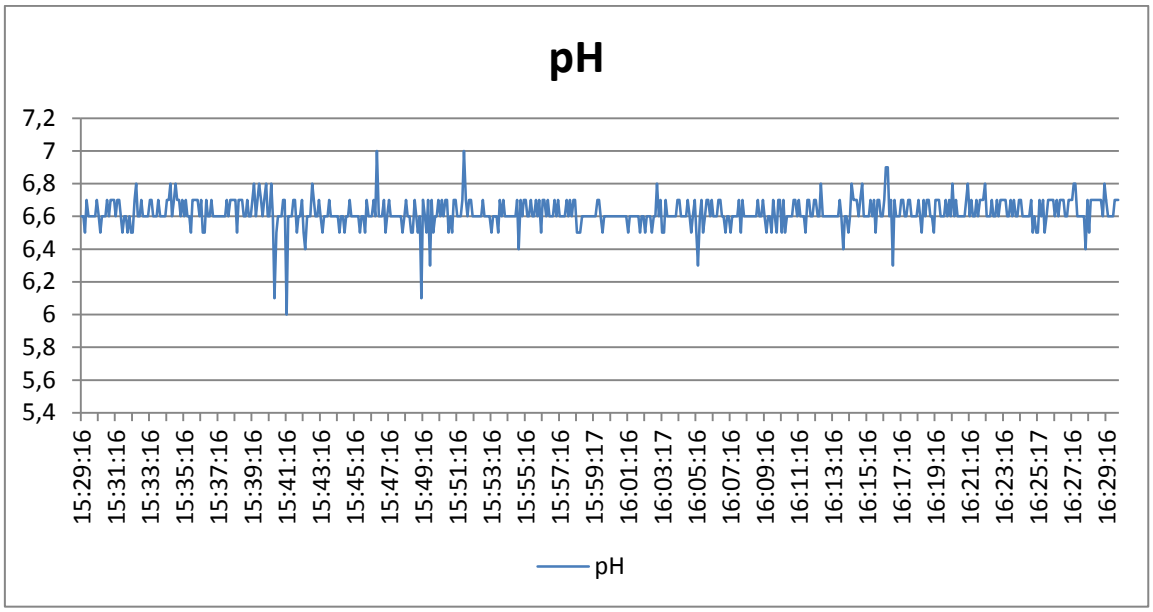



**Gráfico 6:** Valores obtidos na medição do pH de água engarrafada.

Realizaram-se 609 medições num espaço temporal de 1 hora. A média obtida foi 6,62 pH.

Neste caso, para os valores obtidos de pH é necessário fazer compensação da temperatura. Existem duas formas para fazer esta compensação. Uma delas é utilizando no programa de aquisição os valores do sensor de temperatura para fazer a compensação diretamente na aquisição, outra é recorrendo à equação (6).

$$pH = 7.0 + (pH - 7.0) * \frac{T}{T0} \quad (6)$$

Os valores de temperatura T e T0 são expressos em Kelvin, neste caso a temperatura de referência $T0_{20°C} = 293$ K e $T0_{13,42°C} = 286$ K. Como a diferença de

temperatura não é grande e estando o valor de pH medido próximo de 7, o valor de pH já com a compensação é igual.

### 8.2.5 Custo do sistema de monitorização no tanque

O custo do sistema de monitorização no tanque torna-se avolumado devido ao tipo de sensores utilizados, pois as variáveis que se pretendem medir requerem sensores de difícil construção, elevado preço e com características próprias. Normalmente são utilizados equipamentos específicos para fazer a leitura destes sensores. Um dos principais objetivos do trabalho é a criação de todo um sistema de baixo custo. Como a utilização destes sensores é imprescindível, o custo do sistema é menor porque foi desenvolvido o dispositivo de aquisição de dados. O custo do sistema diminui apresenta-se na Tabela 8.

| Referência | Designação | Fabricante | Quantidade | Custo |
|---|---|---|---|---|
| **LFS1K0.155.6W.B.010-6** | Electro Condutividade | IST INNOVATIVE | 1 | 55,23 € |
| **201005 series** | Sensor pH | Jumo | 1 | 67,28 € |
| **519-242** | Sensor de Nível | RS | 1 | 6,42 € |
| **STM32F4Discovery** | Microcontrolador | STMicroelectronics | 1 | 13,72 € |
| **LM741CN/NOPB** | Amplificador operacional | Texas Instruments | 2 | 0,58 € |
| **TL082CP** | Amplificador operacional | Texas Instruments | 4 | 0,64 € |
| **SR215C104KAR** | Condensador 100nF | AVX | 8 | 0,16 € |
| **YR1B10KCC** | Resistência 10kΩ | TE Connectivity | 5 | 0,05 € |
| **YR1B1M0CC** | Resistência 1MΩ | TE Connectivity | 1 | 0,10 € |
| **YR1B100KCC** | Resistência 100kΩ | TE Connectivity | 5 | 0,05 € |
| **YR1B33KCC** | Resistência 33kΩ | TE Connectivity | 1 | 0,05 € |
| | | | **Total** | 148,30 € |

**Tabela 8:** Lista de componentes utilizados no sistema de monitorização no tanque com respetiva descrição e custo.

# 9. CONCLUSÃO E TRABALHO FUTURO

O objetivo deste trabalho consistiu em desenvolver um sistema que permitisse apoiar as culturas agrícolas por hidroponia disponibilizando essa informação ao agricultor. A solução encontrada baseou-se na criação de uma rede de sensores sem fios para monitorizar o ambiente da estufa e uma estação para aquisição de dados sobre as variáveis inerentes à solução nutritiva.

A rede de sensores sem fios é autónoma energeticamente e transmite a informação para um computador central, assenta sobre o protocolo de comunicações *IEEE* 802.15.4 e *ZigBee* que lhe fornece robustez e baixo consumo de energia. Um dos objetivos futuros é a integração de mais nós sensores na rede para permitir uma abrangência maior, neste momento apenas dispõe-se de dois. Também seria pertinente aplicar algoritmos de *routing* e criar funções inteligentes tais como autoconfiguração, deteção e correção de falhas ou erros. Em suma, os nós desenvolvidos cumprem com os requisitos propostos inicialmente e possuem um baixo custo de aquisição.

O sistema de monitorização no tanque cumpre também com os requisitos definidos na aquisição de dados sobre as variáveis físicas, pH, condutividade-elétrica e temperatura. O desenvolvimento e utilização deste tipo de sensores é mais sensível, necessitaram de circuitos de acondicionamento de sinal e um estudo sobre o funcionamento inerente a cada sensor. A grande dificuldade encontrada e que surge como proposta para trabalho futuro tem a ver com a calibração dos sensores de forma a obter-se valores corretos para as medidas efetuadas. É necessário utilizar soluções padrão para calibrar os sensores e também disponibilizá-las ao longo do tempo, pois estes sensor (pH e condutividade elétrica) necessitam e calibrações periódicas. Finalmente o custo do sistema de monitorização no tanque é aceitável para o tipo de sensores utilizados. O custo apenas do sensor com a integração que foi realizada é bem menor comparando com o equipamento de recolha de dados que é normalmente utilizado.

Por fim, todo o sistema cumpre com os requisitos para ser implementado num ambiente real, necessitando apenas de alguns ajustes próprios à integração deste tipo de sistemas. Este trabalho possibilitou o autoconhecimento de diversas matérias a fim de ser concluído da melhor forma. Na bagagem fica a prática e a capacidade de solucionar questões deste tipo.

# 10. BIBLIOGRAFIA


[1] Texas Instruments , 2014. *Low-power Microcontrollers (MCUs).* [Online] Available at: http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/getting-started.page
[Acedido em 05 10 2014].

[2] Ahonen, T., Virrankoski, R. & Elmusrati, M., 2008. *Greenhouse Monitoring with Wireless Sensor Network.* Beijing, s.n., pp. 403-408.

[3] Atmel, 2014. *Atmel - Low Power.* [Online]
Available at: http://www.atmel.com/technologies/lowpower/default.aspx
[Acedido em 05 10 2014].

[4] Bareja, B. G., 2011. Climatic Factors Promote or Inhibit Plant Growth and Development.. *Crop Farming Review.*

[5] Baviskar, J. et al., 2014. *Real time Monitoring and Control System for Green House Based On 802.15.4 Wireless Sensor Network.* Navi Mumbai - India, IEEE.

[6] Controls, I., 2005. *Conductivity Theory and Measurement.* s.l.:s.n.

[7] Cruz, A., Filipe, E. & Pellegrino, O., 2012. *Vocabulário Internacional de Metrologia (VIM).* s.l.:Instituto Português da Qualidade.

[8] Farnell Components SL, 2014. *Farnell element14.* [Online]
Available at: http://pt.farnell.com/
[Acedido em 5 10 2014].

[9] Farnell, s.d. *Farnell element 14.* [Online]
Available at: http://pt.farnell.com
[Acedido em Agosto 2014].

[10] Ijaz, F., Siddiqui, A. A., Im, B. K. & Lee, C., 2012. *Remote management and control system for LED based plant factory using ZigBee and Internet.* s.l., IEEE, pp. 942-946.

[11] J.S.M , L. M. & C., S., 2014. *Design of Efficient Hydroponic Nutrient Solution Control System using Soft Computing based Solution Grading.* s.l., s.n., pp. 148-154.

[12] Kalaivani, T. & P, A. A. P., 2011. *A Survey on Zigbee Based Wireless Sensor Networks.* s.l., IEEE, pp. 85-89.

[13] Kemper, W., 2013. *Guru da Hidroponia.* [Online]
Available at: http://expresso.sapo.pt/guru-da-hidroponia=f816158#ixzz3Gt5Mq49S
[Acedido em 22 10 2014].

[14] Keng, G., Ng, C. K., Noordin, N. K. & Ali, B. M., 2009. A Review of 6LoWPAN Routing Protocols. *Proceedings of the Asia Pacific Advanced Network.*

[15] Khattak, H. A., Ruta, M. & Di Sciascio, E., 2014. *CoAP-based Healthcare Sensor Networks: a survey.* s.l., s.n.

[16] Kwon, J. et al., 2009. *A study on NDIR-based CO2 sensor to apply remote air quality monitoring system.* s.l., s.n., pp. 1683-1687.

[17] LI, L.-l., YANG, S.-f., WANG, L.-y. & GAO, X.-m., 2011. The greenhouse environment monitoring system based on wireless sensor network technology. *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems,* Mar..pp. 265-268.

[18] Li, S., Cui, J. & Li, Z., 2011. Wireless Sensor Network for Precise Agriculture Monitoring. *Intelligent Computation Technology and Automation, International Conference on,* Volume 1, pp. 307-310.

[19] Mouftah, H. T., Khanafer, M. & Guennoun, M., 2014. A Survey of Beacon-Enabled IEEE 802.15.4 MAC Protocols in Wireless. *IEEE Communications Surveys and Tutorials,* 16(2), pp. 856-876.

[20] Ostrikov, S., 2014. *STM32 F4 DAC DMA Waveform Generator.* [Online] Available at: http://00xnor.blogspot.pt/2014/01/6-stm32-f4-dac-dma-waveform-generator.html
[Acedido em 13 10 2014].

[21] Pediredla, B., Ivoghlian, A., Wang, K. I.-K. & Salcic, Z., 2013. A 6LoWPAN implementation for memory constrained and power efficient wireless sensor nodes. *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE.*

[22] Quan, V. M., Gupta, G. S. & S.Mukhopadhyay, 2011. Review of Sensors for Greenhouse Climate Monitoring. *Sensors Applications Symposium (SAS), 2011 IEEE.*

[21] Roberto, K., 2005. *How-to Hydroponics.* s.l.:Futuregarden Press.

[22] RS, s.d. *RS Components.* [Online]
Available at: pt.rs-online.com
[Acedido em Agosto 2014].

[23] Saaid, M., Yahya, N., MZH, N. & Megat, A. M., 2013. *A Development of an Automatic Microcontroller System for Deep Water Culture (DWC).* s.l., s.n., pp. 328-332.

[24] Santos, R. M. P. M., 2008. *Estação Multisensorial para Estufas Agrícolas,* Braga: s.n.

[25] Scaff, R., 2008. *Caracterização eléctrica de dispositivos tipo ISFET com estrutura Si/SiO2/Si3N4 para medição de pH utilizando pseudoeletrodos de Pt, Ag e Au.,* s.l.: s.n.

[26] STMicroelectronics, 2014. *Discovery kit for STM32F407/417 lines - with STM32F407VG MCU.* [Online]
Available at: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419
[Acedido em 13 10 2014].

[27] Vasques, B. L. R. P., Coutinho, I. B. d. A., Lima, M. . F. & Carneval, V. P. d. O., 2010. *ZigBee - Topologias.* [Online]
Available at: http://www.gta.ufrj.br/grad/10_1/zigbee/topologias.html
[Acedido em 05 10 2014].

[28] Webster, J., 1999. *The Measurement, Instrumentation, and Sensors: Handbook.* s.l.:CRC Press.

[29] Xu, X., Yuan, D. & Wan, J., 2008. *An Enhanced Routing Protocol for ZigBee/IEEE 802.15.4 Wireless Networks.* Washington, DC, USA, IEEE Computer Society, pp. 294-298.

[30] Zazueta, F., Bucklin, R. & Jones, P., 1991. *Basic Concepts in Environmental Computer Control of Agricultural Systems.* s.l.:University of Florida.

# 11. ANEXOS

**Anexo A:** *Software* do nó *end-device*.

```
/*
 * File:   main.c
 * Author: Jose Evaristo
 * Emissor (verde)
 * Created on 16 de Julho de 2014, 17:58
 */
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <pic16f1784.h>
#include "registers.h"
#include <math.h>

// set Config bits
#pragma config FOSC=INTOSC, PLLEN=OFF, MCLRE=ON, WDTE=0x01,
#pragma config LVP=OFF, CLKOUTEN=OFF,

// Definitions
#define _XTAL_FREQ  16000000    // this is used by the __delay_ms(xx) and __delay_us(xx)
functions

#define CS LATAbits.LATA7      // CS pin direction
#define RST LATCbits.LATC0     // RST pin direction
#define INT LATCbits.LATC1     // INT pin direction
#define WAKE LATCbits.LATC2    // WAKE pin direction

#define Vref 5.09 // Tensão de referência da ADC

int address_RX_FIFO, address_TX_normal_FIFO;
short int lost_data;

short int ADDRESS_short_1[2], ADDRESS_long_1[8];        // Source address
short int ADDRESS_short_2[2], ADDRESS_long_2[8];        // Destination address
short int PAN_ID_1[2];               // Source PAN ID
short int PAN_ID_2[2];               // Destination PAN ID
short int DATA_RX[1], DATA_TX[5], data_TX_normal_FIFO[18];
short int LQI, RSSI2, SEQ_NUMBER;

// Protótipos de Funções
void PortConfig();
// Adc
void AdcConfig ();
short int AdcRead(unsigned char channel);
// SPI
```

```
void SpiConfig();
//WSN Init
void WsnInit ();

//Outras
void set_not_ACK();
void set_ACK();
void set_not_encrypt();
void start_transmit();
void RF_reset();

void pin_reset();                               // Activate reset from pin
void software_reset();                          // Activate software reset
void nonbeacon_PAN_coordinator_device();

short int read_ZIGBEE_short(short int address);
void set_frame_format_filter(short int fff_mode);
void set_short_address(short int * address);
void set_long_address(short int * address);
void set_PAN_ID(short int * address);
void set_wake_from_pin();                       // Set wake from pin
void pin_wake();
void set_TX_power(int power);
void init_ZIGBEE_basic();
void init_ZIGBEE_nonbeacon();
void write_TX_normal_FIFO();
void PWR_reset();
void pin_wake_up();

/*
 * SPI - Read and Write
 */
void WriteSPI(unsigned int databyte)
{
    //unsigned int buffer;
    //buffer = SSPBUF;          // Read the buffer to clear any remaining data and clear
the buffer full bit
    PIR1bits.SSP1IF=0;          // clear SSP interrupt bit
    SSPBUF  =  databyte;                // //  Write  data  byte  to  the  buffer  to  initiate
transmission
    while(!PIR1bits.SSP1IF);       //  Wait  for  interrupt  flag  to  go  high  indicating
transmission is complete
}

short int ReadSPI(void)
{
    short int databyte;

    PIR1bits.SSP1IF=0;          // Clear SSP interrupt bit
```

```
    SSPBUF = 0x00;                 // Write dummy data byte to the buffer to initiate
transmission
    while(!SSPSTATbits.BF);        // Wait for interrupt flag to go high indicating
transmission is complete
    databyte = SSPBUF;          // Read the incoming data byte
    return (databyte);
}
/*
 * ADC - data acquisition
 */
short int AdcRead(unsigned char channel)
{
    short int adc = 0;
    // Seleciona o canal
    ADCON0bits.CHS = channel; //AN1 0x01;
    // Wait some arbitrary acquisiton time
    __delay_us(5);
    // Inicia a conversão
    ADCON0bits.GO_nDONE = 1;
    // Espera que a conversão seja feita
    while(ADCON0bits.GO_nDONE);
    // Coloca o valor na variavel adc
    adc = (ADRESH<<8) + ADRESL;
    // Envia o valor adquirido
    return(adc);
}
void temperatura()
{
    unsigned int valorT = 0, temperatura = 0;
    // ADC acquiring value
    temperatura = AdcRead(0x01);    // Channel AN1->RA1
    // Conversion
    valorT = ((float)((Vref*temperatura)/(float)(4096)))*1000;     // Vref+=5.02V
    // Send temperature value
    DATA_TX[4] = (valorT%10);
    DATA_TX[3] = '.';
    valorT/=10;
    DATA_TX[2] = (valorT%10);
    valorT/=10;
    DATA_TX[1] = (valorT%10);
    DATA_TX[0] = 0x54; // T
}
void humidade ()
{
    unsigned int valorH = 0, valorHR = 0, humidade = 0;
    // ADC acquiring value
    humidade = AdcRead(0x02);           // Channel AN4->RA5
    // Conversion
```

```
    valorH  =  ((float)((Vref*humidade)/(float)(4096)))*1000;    //  Retira  o  valor  da
voltagem da ADC
    valorHR    =    (float)(((float)((float)(valorH/1000)/Vref)-0.16)/(float)0.0062)*10;
// Valor da Humidade Relativa -> valorH = 5.0*(0.0062*(valorHR)+0.16);
    // Send humidity value
    DATA_TX[4] = (valorHR%10);
    DATA_TX[3] = '.';
    valorHR/=10;
    DATA_TX[2] = (valorHR%10);
    valorHR/=10;
    DATA_TX[1] = (valorHR%10);
    DATA_TX[0] = 0x48; // H
}
void luminosidade ()
{
    unsigned int valorL = 0, luminosidade = 0, valorRLDR=0, valorLux = 0;
    // ADC acquiring value
    luminosidade = AdcRead(0x03);       // Channel AN3->RA3
    // Conversion
    valorL = ((float)((Vref*luminosidade)/(float)(4096)))*1000;    // Retira o valor da
voltagem da adc

    valorRLDR  =  ((float)((10000  *  Vref)/(valorL/1000))-10000);      //  Valor  do  LDR:
R1=((R2*Vin)/vout)-R2

    valorLux  =  452025  *  valorRLDR^(-152/125);         //  Conversão  dos  valores  de
resistência em 1 Lux = 1 Lumen/metro^2

    // Send temperature value
    DATA_TX[4] = (valorLux%10);
    valorLux/=10;
    DATA_TX[3] = (valorLux%10);
    valorLux/=10;
    DATA_TX[2] = (valorLux%10);
    valorLux/=10;
    DATA_TX[1] = (valorLux%10);
    DATA_TX[0] = 'L';
}
int main()
{
    WDTCONbits.WDTPS = 0x0C; //Prescaller para 256segundos
    WDTCONbits.SWDTEN = 0x01; //WDT ligado

    // set up oscillator control register for internal clock running at 16Mhz
    OSCCONbits.SCS = 0x02; //set the SCS bits to select internal oscillator block
    OSCCONbits.IRCF = 0x0F; //set OSCCON IRCF bits to select OSC frequency=16Mhz

    PortConfig();
    AdcConfig ();
```

```
    SpiConfig();
    WsnInit ();

    while(1)
    {
        SLEEP();                    // SLEEP mode for microcontroller

        pin_wake_up();              // Wake up for wireless module

        PORTDbits.RD4 = 1;

        temperatura();
        write_TX_normal_FIFO();     // Transmiting data
        __delay_ms(100);

        humidade();
        write_TX_normal_FIFO();     // Transmiting data
        __delay_ms(100);

        luminosidade();
        write_TX_normal_FIFO();     // Transmiting data
        __delay_ms(100);

        PORTDbits.RD4 = 0;
        PWR_reset();                // SLEEP mode for wireless module
    }
}
/*
 * Configuração das entradas e saídas
 */
void PortConfig (void)
{
    // PORT A Assignments
    TRISAbits.TRISA0 = 0; // RA0 = nc
    TRISAbits.TRISA1 = 1; // RA1 = adc temperatura
    TRISAbits.TRISA2 = 1; // RA2 = adc humidade
    TRISAbits.TRISA3 = 1; // RA3 = adc luminusidade
    TRISAbits.TRISA4 = 0; // RA4 = nc
    TRISAbits.TRISA5 = 0; // RA5 = nc
    TRISAbits.TRISA6 = 0; // RA6 = nc
    TRISAbits.TRISA7 = 0; // RA7 = CS

    // all port A pins are digital I/O com excepção do RA1, RA2 e RA3
    ANSELAbits.ANSA1 = 1;
    ANSELAbits.ANSA2 = 1;
    ANSELAbits.ANSA3 = 1;

    // PORT B Assignments
    TRISBbits.TRISB0 = 0; // RB0 = nc
```

```
    TRISBbits.TRISB1 = 0; // RB1 = nc
    TRISBbits.TRISB2 = 0; // RB2 = nc
    TRISBbits.TRISB3 = 0; // RB3 = nc
    TRISBbits.TRISB4 = 0; // RB4 = nc
    TRISBbits.TRISB5 = 0; // RB5 = nc
    TRISBbits.TRISB6 = 0; // RB6 = nc
    TRISBbits.TRISB7 = 0; // RB7 = nc

    ANSELB = 0x00; // all port B pins are digital I/O

    // PORT C Assignments
    TRISCbits.TRISC0 = 0; // RC0 = RST
    TRISCbits.TRISC1 = 1; // RC1 = INT
    TRISCbits.TRISC2 = 0; // RC2 = WAKE
    TRISCbits.TRISC3 = 0; // RC3 = SCK output SPI
    TRISCbits.TRISC4 = 1; // RC4 = SDI input from SPI
    TRISCbits.TRISC5 = 0; // RC5 = SDO outrput to SPI
    TRISCbits.TRISC6 = 0; // RC6 = nc
    TRISCbits.TRISC7 = 0; // RC7 = nc

    // PORT D Assignments
    TRISDbits.TRISD0 = 0; // RD0 = nc
    TRISDbits.TRISD1 = 0; // RD1 = nc
    TRISDbits.TRISD2 = 0; // RD2 = nc
    TRISDbits.TRISD3 = 0; // RD3 = nc
    TRISDbits.TRISD4 = 0; // RD4 = LED
    TRISDbits.TRISD5 = 0; // RD5 = nc
    TRISDbits.TRISD6 = 0; // RD6 = nc
    TRISDbits.TRISD7 = 0; // RD7 = nc

    ANSELD=0x00;    // all port D pins are digital I/O

    // PORT E Assignments
    TRISEbits.TRISE0 = 0; // RE0 = nc
    TRISEbits.TRISE1 = 0; // RE1 = nc
    TRISEbits.TRISE2 = 0; // RE2 = nc

    ANSELE=0x00;    // all port E pins are digital I/O
}
/*
 * Configuração da ADC
 */
void AdcConfig (void)
{
    ADCON0bits.ADRMD = 0;       // Right Justification
    ADCON1bits.ADFM = 1;
    ADCON0bits.GO_nDONE = 0;    // Stop Conversion
    ADCON0bits.ADON = 1;        // Turn ON ADC
```

```
    ADCON1bits.ADCS = 0x06;     //Fosc/64 -> 4us
    ADCON1bits.ADNREF = 0;      //Vref- ligada ao VSS
    ADCON1bits.ADPREF0 = 0;     //Vref+ ligado ao VDD
    ADCON1bits.ADPREF1 = 0;
}
/*
 * Configuração do SPI para ligação ao módulo wireless
 */
void SpiConfig(void)
{
    SSPCONbits.SSPM=0x01;       // SPI Master mode, clock = Fosc/16 (1 Mhz)
    SSPCONbits.CKP=0;           // Idle state for clock is low
    SSPSTATbits.CKE=1;            // Transmit occurs on transition from active to idle
clock state
    SSPSTATbits.SMP=0;          // Data is sampled at middle of data output time
    SSPCONbits.SSPEN=0x01;      // Enable SPI Port
}
/*
 * Incialização do módulo wireless
 */
void WsnInit (void)
{
    short int i = 0;

    //variable initialization
    LQI = 0;
    RSSI2 = 0;
    SEQ_NUMBER = 0x23;
    lost_data = 0;
    address_RX_FIFO = 0x300;
    address_TX_normal_FIFO = 0;

    for (i = 0; i < 2; i++)
    {
        ADDRESS_short_1[i] = 1;
        ADDRESS_short_2[i] = 2;
        PAN_ID_1[i] = 3;
        PAN_ID_2[i] = 3;
        }

    for (i = 0; i < 8; i++)
    {
      ADDRESS_long_1[i] = 1;
      ADDRESS_long_2[i] = 2;
      }

  DATA_TX[0] = 0;        // Initialize first byte

  __delay_ms(5);
```

```
  pin_reset();                              // Activate reset from pin
  software_reset();                         // Activate software reset
  RF_reset();                               // RF reset
  set_wake_from_pin();                      // Set wake from pin

  set_long_address(ADDRESS_long_1);         // Set long address
  set_short_address(ADDRESS_short_1);       // Set short address
  set_PAN_ID(PAN_ID_1);                     // Set PAN_ID

  init_ZIGBEE_nonbeacon();                  // Initialize ZigBee module
  nonbeacon_PAN_coordinator_device();
  set_TX_power(5);                          // Set max TX power
  set_frame_format_filter(1);               // 1 all frames, 3 data frame only

  pin_wake();
}
/*
 * Functions for reading and writing registers in short address memory space
 */
// write data in short address register
void write_ZIGBEE_short(short int address, short int data_r)
{
  CS = 0;

  address = ((address << 1) & 0b01111111) | 0x01; // calculating addressing mode
  WriteSPI(address);       // addressing register
  WriteSPI(data_r);        // write data in register

  CS = 1;
}

// read data from short address register
short int read_ZIGBEE_short(short int address)
{
  short int data_r = 0;

  CS = 0;

  address = (address << 1) & 0b01111110;      // calculating addressing mode
  WriteSPI(address);                          // addressing register
  data_r = ReadSPI();           // read data from register

  CS = 1;
  return data_r;
}

/*
 * Functions for reading and writing registers in long address memory space
```

```
 */
// Write data in long address register
void write_ZIGBEE_long(int address, short int data_r)
{
  short int address_high = 0, address_low = 0;

  CS = 0;

  address_high = (((short int)(address >> 3)) & 0b01111111) | 0x80;   // calculating
addressing mode
  address_low   = (((short int)(address << 5)) & 0b11100000) | 0x10;   // calculating
addressing mode
  WriteSPI(address_high);           // addressing register
  WriteSPI(address_low);            // addressing register
  WriteSPI(data_r);                 // write data in registerr

  CS = 1;
}

// Read data from long address register
short int read_ZIGBEE_long(int address)
{
  short int data_r = 0;
  short int address_high = 0, address_low = 0;

  CS = 0;

  address_high = ((short int)(address >> 3) & 0b01111111) | 0x80;    //calculating
addressing mode
  address_low   = ((short int)(address << 5) & 0b11100000);             //calculating
addressing mode
  WriteSPI(address_high);            // addressing register
  WriteSPI(address_low);             // addressing register
  data_r = ReadSPI();    // read data from register

  CS = 1;
  return data_r;
}

/*
 * Transmit packet
 */
void start_transmit()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x01;                 // mask for start transmit
  write_ZIGBEE_short(TXNCON, temp);
```

```
}

/*
 * FIFO
 */
void write_TX_normal_FIFO()
{
  int i = 0;

  data_TX_normal_FIFO[0]  = 0x0B;
  data_TX_normal_FIFO[1]  = 0x10;
  data_TX_normal_FIFO[2]  = 0x01;                        // control frame
  data_TX_normal_FIFO[3]  = 0x88;
  data_TX_normal_FIFO[4]  = SEQ_NUMBER;                  // sequence number
  data_TX_normal_FIFO[5]  = PAN_ID_2[1];                 // destinatoin pan
  data_TX_normal_FIFO[6]  = PAN_ID_2[0];
  data_TX_normal_FIFO[7]  = ADDRESS_short_2[0];          // destination address
  data_TX_normal_FIFO[8]  = ADDRESS_short_2[1];
  data_TX_normal_FIFO[9]  = PAN_ID_1[0];                 // source pan
  data_TX_normal_FIFO[10] = PAN_ID_1[1];
  data_TX_normal_FIFO[11] = ADDRESS_short_1[0];          // source address
  data_TX_normal_FIFO[12] = ADDRESS_short_1[1];

  data_TX_normal_FIFO[13] = DATA_TX[0];                  // data
  data_TX_normal_FIFO[14] = DATA_TX[1];                  // data
  data_TX_normal_FIFO[15] = DATA_TX[2];                  // data
  data_TX_normal_FIFO[16] = DATA_TX[3];                  // data
  data_TX_normal_FIFO[17] = DATA_TX[4];                  // data

  for(i = 0; i < (18); i++) {
    write_ZIGBEE_long(address_TX_normal_FIFO  +  i,  data_TX_normal_FIFO[i]);  //  write
frame into normal FIFO
  }

  set_ACK();
  set_not_encrypt();
  start_transmit();
}
/*
 *  Interrupt
 */
void enable_interrupt()
{
 write_ZIGBEE_short(INTCON_M, 0x00);   // 0x00  all interrupts are enable
}

/*
 *  Set channel
 */
```

```
void set_channel(short int channel_number) // 11-26 possible channels
{
  if((channel_number > 26) || (channel_number < 11)) channel_number = 11;
  switch(channel_number) {
    case 11:
      write_ZIGBEE_long(RFCON0, 0x02);  // 0x02 for 11. channel
      break;
    case 12:
      write_ZIGBEE_long(RFCON0, 0x12);  // 0x12 for 12. channel
      break;
    case 13:
      write_ZIGBEE_long(RFCON0, 0x22);  // 0x22 for 13. channel
      break;
    case 14:
      write_ZIGBEE_long(RFCON0, 0x32);  // 0x32 for 14. channel
      break;
    case 15:
      write_ZIGBEE_long(RFCON0, 0x42);  // 0x42 for 15. channel
      break;
    case 16:
      write_ZIGBEE_long(RFCON0, 0x52);  // 0x52 for 16. channel
      break;
    case 17:
      write_ZIGBEE_long(RFCON0, 0x62);  // 0x62 for 17. channel
      break;
    case 18:
      write_ZIGBEE_long(RFCON0, 0x72);  // 0x72 for 18. channel
      break;
    case 19:
      write_ZIGBEE_long(RFCON0, 0x82);  // 0x82 for 19. channel
      break;
    case 20:
      write_ZIGBEE_long(RFCON0, 0x92);  // 0x92 for 20. channel
      break;
    case 21:
      write_ZIGBEE_long(RFCON0, 0xA2);  // 0xA2 for 21. channel
      break;
    case 22:
      write_ZIGBEE_long(RFCON0, 0xB2);  // 0xB2 for 22. channel
      break;
    case 23:
      write_ZIGBEE_long(RFCON0, 0xC2);  // 0xC2 for 23. channel
      break;
    case 24:
      write_ZIGBEE_long(RFCON0, 0xD2);  // 0xD2 for 24. channel
      break;
    case 25:
      write_ZIGBEE_long(RFCON0, 0xE2);  // 0xE2 for 25. channel
      break;
```

```
    case 26:
      write_ZIGBEE_long(RFCON0, 0xF2);  // 0xF2 for 26. channel
      break;
  }
  RF_reset();
}

/*
 *  Set CCA mode
 */
void set_CCA_mode(short int CCA_mode)
{
  short int temp = 0;
  switch(CCA_mode) {
        case 1: {                               // ENERGY ABOVE THRESHOLD
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0x80;                   // 0x80 mask
          temp = temp & 0xDF;                   // 0xDF mask
          write_ZIGBEE_short(BBREG2, temp);
          write_ZIGBEE_short(CCAEDTH, 0x60);    // Set CCA ED threshold to -69 dBm
        }
        break;

        case 2: {                               // CARRIER SENSE ONLY
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0x40;                   // 0x40 mask
          temp = temp & 0x7F;                   // 0x7F mask
          write_ZIGBEE_short(BBREG2, temp);

          temp = read_ZIGBEE_short(BBREG2);     // carrier sense threshold
          temp = temp | 0x38;
          temp = temp & 0xFB;
          write_ZIGBEE_short(BBREG2, temp);
        }
        break;

        case 3: {                                       // CARRIER SENSE AND ENERGY ABOVE
THRESHOLD
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0xC0;                   // 0xC0 mask
          write_ZIGBEE_short(BBREG2, temp);

          temp = read_ZIGBEE_short(BBREG2);     // carrier sense threshold
          temp = temp | 0x38;                   // 0x38 mask
          temp = temp & 0xFB;                   // 0xFB mask
          write_ZIGBEE_short(BBREG2, temp);

          write_ZIGBEE_short(CCAEDTH, 0x60);    // Set CCA ED threshold to -69 dBm
        }
```

```
        break;
    }
 }

/*
 *  Set RSSI mode
 */
void set_RSSI_mode(short int RSSI_mode) {       // 1 for RSSI1, 2 for RSSI2 mode
  short int temp = 0;

  switch(RSSI_mode) {
    case 1: {
      temp = read_ZIGBEE_short(BBREG6);
      temp = temp | 0x80;                       // 0x80 mask for RSSI1 mode
      write_ZIGBEE_short(BBREG6, temp);
    }
    break;

    case 2:
      write_ZIGBEE_short(BBREG6, 0x40);         // 0x40 data for RSSI2 mode
      break;
  }
}

/*
 * Set type of device
 */
void nonbeacon_PAN_coordinator_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp | 0x08;                 // 0x08 mask for PAN coordinator
  write_ZIGBEE_short(RXMCR, temp);

  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);

  write_ZIGBEE_short(ORDER, 0xFF);    // BO, SO are 15
}

void nonbeacon_coordinator_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp | 0x04;                 // 0x04 mask for coordinator
  write_ZIGBEE_short(RXMCR, temp);
```

```
  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);

  write_ZIGBEE_short(ORDER, 0xFF);    // BO, SO  are 15
}

void nonbeacon_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp & 0xF3;                 // 0xF3 mask for PAN coordinator and coordinator
  write_ZIGBEE_short(RXMCR, temp);

  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);
}

/*
 * ACK request
 */
void set_ACK(void)
{
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x04;                   // 0x04 mask for set ACK
  write_ZIGBEE_short(TXNCON, temp);
}

void set_not_ACK(void)
{
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp & (!0x04);                // 0x04 mask for set not ACK
  write_ZIGBEE_short(TXNCON, temp);
}

/*
 *  Encrypt
 */
void set_encrypt(void)
{
  short int temp = 0;
```

```
  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x02;                  // mask for set encrypt
  write_ZIGBEE_short(TXNCON, temp);
}

void set_not_encrypt(void){
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp & (!0x02);               // mask for set not encrypt
  write_ZIGBEE_short(TXNCON, temp);
}

/*
 * Interframe spacing
 */
void set_IFS_recomended()
{
  short int temp = 0;

  write_ZIGBEE_short(RXMCR, 0x93);    // Min SIFS Period

  temp = read_ZIGBEE_short(TXPEND);
  temp = temp | 0x7C;                 // MinLIFSPeriod
  write_ZIGBEE_short(TXPEND, temp);

  temp = read_ZIGBEE_short(TXSTBL);
  temp = temp | 0x90;                 // MinLIFSPeriod
  write_ZIGBEE_short(TXSTBL, temp);

  temp = read_ZIGBEE_short(TXTIME);
  temp = temp | 0x31;                 // TurnaroundTime
  write_ZIGBEE_short(TXTIME, temp);
}

void set_IFS_default()
{
  short int temp = 0;

  write_ZIGBEE_short(RXMCR, 0x75);    // Min SIFS Period

  temp = read_ZIGBEE_short(TXPEND);
  temp = temp | 0x84;                 // Min LIFS Period
  write_ZIGBEE_short(TXPEND, temp);

  temp = read_ZIGBEE_short(TXSTBL);
  temp = temp | 0x50;                 // Min LIFS Period
  write_ZIGBEE_short(TXSTBL, temp);
```

```
  temp = read_ZIGBEE_short(TXTIME);
  temp = temp | 0x41;                 // Turnaround Time
  write_ZIGBEE_short(TXTIME, temp);
}
/*
 *  Frame format filter
 */
void set_frame_format_filter(short int fff_mode) {    // 1 all frames, 2 command only, 3
data only, 4 beacon only
  short int temp = 0;

  switch(fff_mode) {
   case 1: {
     temp = read_ZIGBEE_short(RXFLUSH);      // all frames
     temp = temp & (!0x0E);                  // mask for all frames
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 2: {
     temp = read_ZIGBEE_short(RXFLUSH);      // command only
     temp = temp & (!0x06);                  // mask for command only
     temp = temp | 0x08;                     // mask for command only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 3: {
     temp = read_ZIGBEE_short(RXFLUSH);      // data only
     temp = temp & (!0x0A);                  // mask for data only
     temp = temp | 0x04;                     // mask for data only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 4: {
     temp = read_ZIGBEE_short(RXFLUSH);      // beacon only
     temp = temp & (!0x0C);                  // mask for beacon only
     temp = temp | 0x02;                     // mask for beacon only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;
  }
}

/*
 *  Flush RX FIFO pointer
 */
void flush_RX_FIFO_pointer()
```

```
{
  short int temp;

  temp = read_ZIGBEE_short(RXFLUSH);
  temp = temp | 0x01;                        // mask for flush RX FIFO
  write_ZIGBEE_short(RXFLUSH, temp);
}

/* Address
 * configura o endereço
 */
void set_short_address(short int * address)
{
  write_ZIGBEE_short(SADRL, address[0]);
  write_ZIGBEE_short(SADRH, address[1]);
}

void set_long_address(short int * address)
{
  short int i = 0;

  for(i = 0; i < 8; i++) {
    write_ZIGBEE_short(EADR0 + i, address[i]);   // 0x05 address of EADR0
  }
}

void set_PAN_ID(short int * address)
{
  write_ZIGBEE_short(PANIDL, address[0]);
  write_ZIGBEE_short(PANIDH, address[1]);
}

/*
 * Wake
 */
void set_wake_from_pin()
{
  short int temp = 0;

  WAKE = 0;
  temp = read_ZIGBEE_short(RXFLUSH);
  temp = temp | 0x60;                      // mask
  write_ZIGBEE_short(RXFLUSH, temp);

  temp = read_ZIGBEE_short(WAKECON);
  temp = temp | 0x80;
  write_ZIGBEE_short(WAKECON, temp);
}
void pin_wake_up()
```

```
{
    WAKE = 1;
    write_ZIGBEE_short(RFCTL, 0x04); // RF state machine reset
    write_ZIGBEE_short(RFCTL, 0x00);
    __delay_ms(5);
}

void pin_wake()
{
  WAKE = 1;
  __delay_ms(5);
}

/*
 * PLL
 */
void enable_PLL()
{
  write_ZIGBEE_long(RFCON2, 0x80);       // mask for PLL enable
}

void disable_PLL()
{
  write_ZIGBEE_long(RFCON2, 0x00);       // mask for PLL disable
}

/*
 * Tx power
 */
void set_TX_power(int power) // 0-31 possible variants
{
  if((power < 0) || (power > 31))
    power = 31;
  power = 31 - power;                                    // 0 max, 31 min -> 31 max, 0
min
  power = ((power & 0b00011111) << 3) & 0b11111000;       // calculating power
  write_ZIGBEE_long(RFCON3, power);
}

/*
* Init ZIGBEE module
*/
void init_ZIGBEE_basic()
{
  write_ZIGBEE_short(PACON2, 0x98);   // Initialize FIFOEN = 1 and TXONTS = 0x6
  write_ZIGBEE_short(TXSTBL, 0x95);   // Initialize RFSTBL = 0x9
  write_ZIGBEE_long(RFCON1, 0x01);    // Initialize VCOOPT = 0x01
  enable_PLL();                       // Enable PLL (PLLEN = 1)
  write_ZIGBEE_long(RFCON6, 0x90);    // Initialize TXFIL = 1 and 20MRECVR = 1
```

```
  write_ZIGBEE_long(RFCON7, 0x80);     // Initialize SLPCLKSEL = 0x2 (100 kHz Internal
oscillator)
  write_ZIGBEE_long(RFCON8, 0x10);    // Initialize RFVCO = 1
  write_ZIGBEE_long(SLPCON1, 0x21);   // Initialize CLKOUTEN = 1 and SLPCLKDIV = 0x01
}

void init_ZIGBEE_nonbeacon()
{
  init_ZIGBEE_basic();
  set_CCA_mode(1);     // Set CCA mode to ED and set threshold
  set_RSSI_mode(2);    // RSSI2 mode
  enable_interrupt();  // Enables all interrupts
  set_channel(11);     // Channel 11
  RF_reset();
}
/*
 * Reset functions
 */
void pin_reset() // Reset from pin
{
  RST = 0;  // activate reset
  __delay_ms(5);
  RST = 1;  // deactivate reset
  __delay_ms(5);
}

void PWR_reset()
{
  write_ZIGBEE_short(SOFTRST, 0x04);    // 0x04  mask for RSTPWR bit
  write_ZIGBEE_short(SLPACK, 0x80);     // 0x80 mask for put sleep immediately
}

void BB_reset()
{
  write_ZIGBEE_short(SOFTRST, 0x02);    // 0x02 mask for RSTBB bit
}

void MAC_reset()
{
  write_ZIGBEE_short(SOFTRST, 0x01);    // 0x01 mask for RSTMAC bit
}

void software_reset() // PWR_reset,BB_reset and MAC_reset at once
{
  write_ZIGBEE_short(SOFTRST, 0x07);
}

void RF_reset()
{
```

```
  short int temp = 0;
  temp = read_ZIGBEE_short(RFCTL);
  temp = temp | 0x04;                   // mask for RFRST bit
  write_ZIGBEE_short(RFCTL, temp);
  temp = temp & (!0x04);                // mask for RFRST bit
  write_ZIGBEE_short(RFCTL, temp);
  __delay_ms(1);
}
```

**Anexo B:** *Software* do nó *router*.

```
/*
 * File:   main.c
 * Author: Jose Evaristo
 * Receptor (amarelo)
 * Created on 16 de Julho de 2014, 17:58
 */
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <pic16f1784.h>
#include "registers.h"

// set Config bits
#pragma config FOSC=INTOSC, PLLEN=OFF, MCLRE=ON, WDTE=0x01,
#pragma config LVP=OFF, CLKOUTEN=OFF,

// Definitions
#define _XTAL_FREQ  16000000    // this is used by the __delay_ms(xx) and __delay_us(xx)
functions

#define CS LATAbits.LATA7      // CS pin direction
#define RST LATCbits.LATC0     // RST pin direction
#define INT LATBbits.LATB0     // INT pin direction
#define WAKE LATCbits.LATC2    // WAKE pin direction

/*****************************************************************************************
*********************
 * Variaveis Globais
 */
volatile unsigned int temp1=0; // Received data

int address_RX_FIFO, address_TX_normal_FIFO;
short int data_RX_FIFO[17], lost_data;

short int ADDRESS_short_1[2], ADDRESS_long_1[8];        // Source address
short int ADDRESS_short_2[2], ADDRESS_long_2[8];        // Destination address
short int PAN_ID_1[2];               // Source PAN ID
short int PAN_ID_2[2];               // Destination PAN ID
short int DATA_RX[5], DATA_TX[1], data_TX_normal_FIFO[14];
short int LQI, RSSI2, SEQ_NUMBER;

/*****************************************************************************************
 * Protótipos de Funções
 */
void PortConfig();
// Usart
void UsartConfig();
//SPI
void SpiConfig();
//Modulo wireless
void WsnInit();
//Outras
void set_not_ACK();
void set_not_encrypt();
void start_transmit();
void RF_reset();

void pin_reset();                              // Activate reset from pin
```

```
void software_reset();                          // Activate software reset
void nonbeacon_PAN_coordinator_device();

char Debounce_INT();
short int read_ZIGBEE_short(short int address);
void read_RX_FIFO();
void set_reception_mode(short int r_mode);
void set_frame_format_filter(short int fff_mode);
void set_short_address(short int * address);
void set_long_address(short int * address);
void set_PAN_ID(short int * address);
void set_wake_from_pin();                       // Set wake from pin
void pin_wake();
void set_TX_power(int power);
void init_ZIGBEE_basic();
void init_ZIGBEE_nonbeacon();
void PWR_reset();
void pin_wake_up();

void usart_send();

/*******************************************************************************
 * SPI - Read and Write
 */
void WriteSPI(unsigned int databyte)
{
    PIR1bits.SSP1IF=0;        // clear SSP interrupt bit
    SSPBUF = databyte;        // Write data byte to the buffer to initiate transmission
    while(!PIR1bits.SSP1IF);  //  Wait  for  interrupt  flag  to  go  high  indicating
transmission is complete
}

unsigned int ReadSPI(void)
{
    unsigned int databyte;

    PIR1bits.SSP1IF=0;   // Clear SSP interrupt bit
    SSPBUF = 0x00;       // Write dummy data byte to the buffer to initiate transmission
    while(!SSPSTATbits.BF);// Wait for interrupt flag to go high indicating transmission
is complete
    databyte = SSPBUF;    // Read the incoming data byte
    return (databyte);
}
/*
 * Interrupt Service Routine
 */
void interrupt ISR()
{
    if (INTCONbits.INTF) // External interrupt INT-> RB0
    {
        LATDbits.LATD3 = 1;
        temp1 = read_ZIGBEE_short(INTSTAT);  // Read and flush register INTSTAT
        read_RX_FIFO();                      // Read receive data
        usart_send();                        // Send received data from usart
        LATDbits.LATD3 = 0;
        INTCONbits.INTF = 0;
    }
}
/*
 * USART Send Data
 */
void usart_send()
{
    int i=0;

    while(!TXSTAbits.TRMT);    // make sure buffer full bit is high before transmitting
    for(i=0;i<5;i++){
        TXREG = DATA_RX[i];
        __delay_ms(1);
        }
    TXREG = '\n';
    __delay_ms(1);
}
```

```
/*****************************************************************************
 * MAIN
 *
*****************************************************************************
 */
int main()
{
    WDTCONbits.WDTPS = 0x0C; //Prescaller para 256segundos
    WDTCONbits.SWDTEN = 0x01; //WDT ligado
    // set up oscillator control register for internal clock running at 16Mhz
    OSCCONbits.SCS = 0x02; //set the SCS bits to select internal oscillator block
    OSCCONbits.IRCF = 0x0F; //set OSCCON IRCF bits to select OSC frequency=16Mhz

    PortConfig();
    UsartConfig();
    SpiConfig();
    WsnInit();
    usart_send();

    while(1)
    {
    }
}
/*
 * Configuração das entradas e saídas
 */
void PortConfig (void)
{
    // PORT A Assignments
    TRISAbits.TRISA0 = 0; // RA0 = nc
    TRISAbits.TRISA1 = 0; // RA1 = nc
    TRISAbits.TRISA2 = 0; // RA2 = nc
    TRISAbits.TRISA3 = 0; // RA3 = nc
    TRISAbits.TRISA4 = 0; // RA4 = nc
    TRISAbits.TRISA5 = 0; // RA5 = nc
    TRISAbits.TRISA6 = 0; // RA6 = nc
    TRISAbits.TRISA7 = 0; // RA7 = CS

    ANSELA = 0x00; // all port A pins are digital I/O
    LATDbits.LATD = 0x00;

    // PORT B Assignments
    TRISBbits.TRISB0 = 1; // RB0 = INT
    TRISBbits.TRISB1 = 0; // RB1 = nc
    TRISBbits.TRISB2 = 0; // RB2 = nc
    TRISBbits.TRISB3 = 0; // RB3 = nc
    TRISBbits.TRISB4 = 0; // RB4 = nc
    TRISBbits.TRISB5 = 0; // RB5 = nc
    TRISBbits.TRISB6 = 0; // RB6 = nc
    TRISBbits.TRISB7 = 0; // RB7 = nc

    ANSELB = 0x00;        // all port B pins are digital I/O
    INTCONbits.INTE = 1;  // Enable external interrupt
    INTCONbits.GIE = 1;   // Enable Global Interrupt
    INTCONbits.PEIE = 0;  // Disable all unmasked peripheral interrupt
    OPTION_REGbits.INTEDG = 0; //Interrupt on rising edge

    //  PORT C Assignments
    TRISCbits.TRISC0 = 0; // RC0 = RST
    TRISCbits.TRISC1 = 0; // RC1 = nc
    TRISCbits.TRISC2 = 0; // RC2 = WAKE
    TRISCbits.TRISC3 = 0; // RC3 = SCK output SPI
    TRISCbits.TRISC4 = 1; // RC4 = SDI input from SPI
    TRISCbits.TRISC5 = 0; // RC5 = SDO output to SPI
    TRISCbits.TRISC6 = 0; // RC6 = TX out
    TRISCbits.TRISC7 = 1; // RC7 = RX in

    //  PORT D Assignments
    TRISDbits.TRISD0 = 0; // RD0 = nc
    TRISDbits.TRISD1 = 0; // RD1 = nc
    TRISDbits.TRISD2 = 0; // RD2 = nc
    TRISDbits.TRISD3 = 0; // RD3 = LED
    TRISDbits.TRISD4 = 0; // RD4 = nc
```

```
    TRISDbits.TRISD5 = 0; // RD5 = nc
    TRISDbits.TRISD6 = 0; // RD6 = nc
    TRISDbits.TRISD7 = 0; // RD7 = nc

    ANSELD=0x00;    // all port D pins are digital I/O

    //  PORT E Assignments
    TRISEbits.TRISE0 = 0; // RE0 = nc
    TRISEbits.TRISE1 = 0; // RE1 = nc
    TRISEbits.TRISE2 = 0; // RE2 = nc

    ANSELE=0x00;    // all port E pins are digital I/O
}
/*
 * Configuração da USART
 */
void UsartConfig(void)
{
    TXSTAbits.BRGH=0;       // select low speed Baud Rate (see baud rate calcs below)
    TXSTAbits.TX9=0;        // select 8 data bits
    TXSTAbits.TXEN = 1;     // enable transmit


    RCSTAbits.SPEN=1;       // serial port is enabled
    RCSTAbits.RX9=0;        // select 8 data bits
    RCSTAbits.CREN=1;       // receive enabled

    // calculate values of SPBRGL and SPBRGH based on the desired baud rate
    //
    // For 8 bit Async mode with BRGH=0: Desired Baud rate = Fosc/64([SPBRGH:SPBRGL]+1)
    // For 8 bit Async mode with BRGH=1: Desired Baud rate = Fosc/16([SPBRGH:SPBRGL]+1)

    // For our example,we will use BRGH=0,Fosc=16Mhz and we want baud rate=9600
    //
    //  9600 = Fosc/64([SPBRGH:SPBRGL]+1)
    //  9600 = Fosc/64(X+1)
    //  9600 = Fosc/64X + 64
    //  9600(64X + 64) = Fosc
    //  X = [Fosc/(9600)(64)]-1
    //  X = [16000000/(9600)(64)] -1
    //  X = SPBRGH:SPBRGL = 25.01 (round to 25)

    SPBRGL=25;  // here is calculated value of SPBRGH and SPBRGL
    SPBRGH=0;
}
/*
 * Configuração do SPI para ligação ao módulo wireless
 */
void SpiConfig(void)
{
    SSPCONbits.SSPM=0x01;       // SPI Master mode, clock = Fosc/16 (1 Mhz)
    SSPCONbits.CKP=0;           // Idle state for clock is low
    SSPSTATbits.CKE=1;            // Transmit occurs on transition from active to idle
clock state
    SSPSTATbits.SMP=0;          // Data is sampled at middle of data output time
    SSPCONbits.SSPEN=0x01;      // Enable SPI Port
}
/*
 * Inicialização do módulo wireless
 */
void WsnInit(void)
{
    short int i = 0;
    // variable initialization
    LQI = 0;
    RSSI2 = 0;
    SEQ_NUMBER = 0x23;
    lost_data = 0;
    address_RX_FIFO = 0x300;
    address_TX_normal_FIFO = 0;

    LATDbits.LATD=0x00;
```

```
    for (i = 0; i < 2; i++) {
        ADDRESS_short_1[i] = 1;
        ADDRESS_short_2[i] = 2;
        PAN_ID_1[i] = 3;
        PAN_ID_2[i] = 3;
    }

    for (i = 0; i < 8; i++) {
        ADDRESS_long_1[i] = 1;
        ADDRESS_long_2[i] = 2;
    }
    __delay_ms(5);

    pin_reset();                            // Activate reset from pin
    software_reset();                       // Activate software reset
    RF_reset();                             // RF reset
    set_wake_from_pin();                    // Set wake from pin

    set_long_address(ADDRESS_long_2);       // Set long address
    set_short_address(ADDRESS_short_2);     // Set short address
    set_PAN_ID(PAN_ID_2);                   // Set PAN_ID

    init_ZIGBEE_nonbeacon();                // Initialize ZigBee module
    nonbeacon_PAN_coordinator_device();
    set_TX_power(31);                       // Set max TX power
    set_frame_format_filter(1);             // 1 all frames, 3 data frame only
    set_reception_mode(1);                  // 1 normal mode

    pin_wake();                             // Wake from pin
}

/*
 * Functions for reading and writing registers in short address memory space
 */
// write data in short address register
void write_ZIGBEE_short(short int address, short int data_r) {
  CS = 0;

  address = ((address << 1) & 0b01111111) | 0x01; // calculating addressing mode
  WriteSPI(address);       // addressing register
  WriteSPI(data_r);        // write data in register

  CS = 1;
}

// read data from short address register
short int read_ZIGBEE_short(short int address)
{
    short int data_r = 0;

    CS = 0;

    address = (address << 1) & 0b01111110;      // calculating addressing mode
    WriteSPI(address);                          // addressing register
    data_r = ReadSPI();           // read data from register

    CS = 1;
    return data_r;
}

/*
 * Functions for reading and writing registers in long address memory space
 */
// Write data in long address register
void write_ZIGBEE_long(int address, short int data_r) {
  short int address_high = 0, address_low = 0;

  CS = 0;

  address_high = (((short int)(address >> 3)) & 0b01111111) | 0x80;   // calculating
addressing mode
  address_low   = (((short int)(address << 5)) & 0b11100000) | 0x10;   // calculating
addressing mode
```

```
  WriteSPI(address_high);           // addressing register
  WriteSPI(address_low);            // addressing register
  WriteSPI(data_r);                 // write data in registerr

  CS = 1;
}

// Read data from long address register
short int read_ZIGBEE_long(int address) {
  short int data_r = 0;
  short int address_high = 0, address_low = 0;

  CS = 0;

  address_high = ((short int)(address >> 3) & 0b01111111) | 0x80;    //calculating
addressing mode
  address_low  = ((short int)(address << 5) & 0b11100000);            //calculating
addressing mode
  WriteSPI(address_high);            // addressing register
  WriteSPI(address_low);             // addressing register
  data_r = ReadSPI();    // read data from register

  CS = 1;
  return data_r;
}

/*
 * Transmit packet
 */
void start_transmit() {
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x01;                 // mask for start transmit
  write_ZIGBEE_short(TXNCON, temp);
}

/*
 * FIFO
 */
void read_RX_FIFO() {
  unsigned short int temp = 0;
  int i = 0;

  temp = read_ZIGBEE_short(BBREG1);      // disable receiving packets off air.
  temp = temp | 0x04;                    // mask for disable receiving packets
  write_ZIGBEE_short(BBREG1, temp);

  for(i=0; i<128; i++)
  {
      if(i <  (18))
          data_RX_FIFO[i] = read_ZIGBEE_long(address_RX_FIFO + i);   // reading valid
data from RX FIFO

      if(i >= (18))
          lost_data = read_ZIGBEE_long(address_RX_FIFO + i);         // reading invalid
data from RX FIFO
  }

  DATA_RX[0] = data_RX_FIFO[12];              // coping valid data
  DATA_RX[1] = data_RX_FIFO[13];              // coping valid data
  DATA_RX[2] = data_RX_FIFO[14];              // coping valid data
  DATA_RX[3] = data_RX_FIFO[15];              // coping valid data
  DATA_RX[4] = data_RX_FIFO[16];              // coping valid data

  LQI   = data_RX_FIFO[17];  // coping valid data
  RSSI2 = data_RX_FIFO[18];  // coping valid data

  temp = read_ZIGBEE_short(BBREG1);      // enable receiving packets off air.
  temp = temp & (!0x04);                 // mask for enable receiving
  write_ZIGBEE_short(BBREG1, temp);
}
```

```
void write_TX_normal_FIFO() {
  int i = 0;

  data_TX_normal_FIFO[0]  = 0x0B;
  data_TX_normal_FIFO[1]  = 0x0C;
  data_TX_normal_FIFO[2]  = 0x01;                          // control frame
  data_TX_normal_FIFO[3]  = 0x88;
  data_TX_normal_FIFO[4]  = SEQ_NUMBER;                    // sequence number
  data_TX_normal_FIFO[5]  = PAN_ID_2[1];                   // destinatoin pan
  data_TX_normal_FIFO[6]  = PAN_ID_2[0];
  data_TX_normal_FIFO[7]  = ADDRESS_short_2[0];            // destination address
  data_TX_normal_FIFO[8]  = ADDRESS_short_2[1];
  data_TX_normal_FIFO[9]  = PAN_ID_1[0];                   // source pan
  data_TX_normal_FIFO[10] = PAN_ID_1[1];
  data_TX_normal_FIFO[11] = ADDRESS_short_1[0];            // source address
  data_TX_normal_FIFO[12] = ADDRESS_short_1[1];
  data_TX_normal_FIFO[13] = DATA_TX[0];                    // data

  for(i = 0; i < (14); i++) {
    write_ZIGBEE_long(address_TX_normal_FIFO  +  i,  data_TX_normal_FIFO[i]);  //  write
frame into normal FIFO
  }

  set_not_ACK();
  set_not_encrypt();
  start_transmit();
}

/*
 *  Interrupt
 */
void enable_interrupt() {
 write_ZIGBEE_short(INTCON_M, 0x00);   //0x00  all interrupts are enable
}

/*
 *  Set channel
 */
void set_channel(short int channel_number) {               // 11-26 possible channels
  if((channel_number > 26) || (channel_number < 11)) channel_number = 11;
  switch(channel_number) {
    case 11:
      write_ZIGBEE_long(RFCON0, 0x02);  // 0x02 for 11. channel
      break;
    case 12:
      write_ZIGBEE_long(RFCON0, 0x12);  // 0x12 for 12. channel
      break;
    case 13:
      write_ZIGBEE_long(RFCON0, 0x22);  // 0x22 for 13. channel
      break;
    case 14:
      write_ZIGBEE_long(RFCON0, 0x32);  // 0x32 for 14. channel
      break;
    case 15:
      write_ZIGBEE_long(RFCON0, 0x42);  // 0x42 for 15. channel
      break;
    case 16:
      write_ZIGBEE_long(RFCON0, 0x52);  // 0x52 for 16. channel
      break;
    case 17:
      write_ZIGBEE_long(RFCON0, 0x62);  // 0x62 for 17. channel
      break;
    case 18:
      write_ZIGBEE_long(RFCON0, 0x72);  // 0x72 for 18. channel
      break;
    case 19:
      write_ZIGBEE_long(RFCON0, 0x82);  // 0x82 for 19. channel
      break;
    case 20:
      write_ZIGBEE_long(RFCON0, 0x92);  // 0x92 for 20. channel
      break;
    case 21:
      write_ZIGBEE_long(RFCON0, 0xA2);  // 0xA2 for 21. channel
```

```
      break;
    case 22:
      write_ZIGBEE_long(RFCON0, 0xB2);  // 0xB2 for 22. channel
      break;
    case 23:
      write_ZIGBEE_long(RFCON0, 0xC2);  // 0xC2 for 23. channel
      break;
    case 24:
      write_ZIGBEE_long(RFCON0, 0xD2);  // 0xD2 for 24. channel
      break;
    case 25:
      write_ZIGBEE_long(RFCON0, 0xE2);  // 0xE2 for 25. channel
      break;
    case 26:
      write_ZIGBEE_long(RFCON0, 0xF2);  // 0xF2 for 26. channel
      break;
  }
  RF_reset();
}

/*
 *  Set CCA mode
 */
void set_CCA_mode(short int CCA_mode) {
  short int temp = 0;
  switch(CCA_mode) {
        case 1: {                               // ENERGY ABOVE THRESHOLD
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0x80;                   // 0x80 mask
          temp = temp & 0xDF;                   // 0xDF mask
          write_ZIGBEE_short(BBREG2, temp);
          write_ZIGBEE_short(CCAEDTH, 0x60);    // Set CCA ED threshold to -69 dBm
        }
        break;

        case 2: {                               // CARRIER SENSE ONLY
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0x40;                   // 0x40 mask
          temp = temp & 0x7F;                   // 0x7F mask
          write_ZIGBEE_short(BBREG2, temp);

          temp = read_ZIGBEE_short(BBREG2);     // carrier sense threshold
          temp = temp | 0x38;
          temp = temp & 0xFB;
          write_ZIGBEE_short(BBREG2, temp);
        }
        break;

        case 3: {                                     // CARRIER SENSE AND ENERGY ABOVE
THRESHOLD
          temp = read_ZIGBEE_short(BBREG2);
          temp = temp | 0xC0;                   // 0xC0 mask
          write_ZIGBEE_short(BBREG2, temp);

          temp = read_ZIGBEE_short(BBREG2);     // carrier sense threshold
          temp = temp | 0x38;                   // 0x38 mask
          temp = temp & 0xFB;                   // 0xFB mask
          write_ZIGBEE_short(BBREG2, temp);

          write_ZIGBEE_short(CCAEDTH, 0x60);    // Set CCA ED threshold to -69 dBm
        }
        break;
    }
 }

/*
 *  Set RSSI mode
 */
void set_RSSI_mode(short int RSSI_mode) // 1 for RSSI1, 2 for RSSI2 mode
{
  short int temp = 0;

  switch(RSSI_mode) {
```

```
    case 1: {
      temp = read_ZIGBEE_short(BBREG6);
      temp = temp | 0x80;                       // 0x80 mask for RSSI1 mode
      write_ZIGBEE_short(BBREG6, temp);
    }
    break;

    case 2:
      write_ZIGBEE_short(BBREG6, 0x40);         // 0x40 data for RSSI2 mode
      break;
  }
}

/*
 * Set type of device
 */
void nonbeacon_PAN_coordinator_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp | 0x08;                 // 0x08 mask for PAN coordinator
  write_ZIGBEE_short(RXMCR, temp);

  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);

  write_ZIGBEE_short(ORDER, 0xFF);    // BO, SO are 15
}

void nonbeacon_coordinator_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp | 0x04;                 // 0x04 mask for coordinator
  write_ZIGBEE_short(RXMCR, temp);

  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);

  write_ZIGBEE_short(ORDER, 0xFF);    // BO, SO  are 15
}

void nonbeacon_device()
{
  short int temp = 0;

  temp = read_ZIGBEE_short(RXMCR);
  temp = temp & 0xF3;                 // 0xF3 mask for PAN coordinator and coordinator
  write_ZIGBEE_short(RXMCR, temp);

  temp = read_ZIGBEE_short(TXMCR);
  temp = temp & 0xDF;                 // 0xDF mask for CSMA-CA mode
  write_ZIGBEE_short(TXMCR, temp);
}

/*
 * ACK request
 */
void set_ACK() {
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x04;                 // 0x04 mask for set ACK
  write_ZIGBEE_short(TXNCON, temp);
}

void set_not_ACK() {
  short int temp = 0;
```

```
  temp = read_ZIGBEE_short(TXNCON);
  temp = temp & (!0x04);              // 0x04 mask for set not ACK
  write_ZIGBEE_short(TXNCON, temp);
}

/*
 *  Encrypt
 */
void set_encrypt() {
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp | 0x02;                 // mask for set encrypt
  write_ZIGBEE_short(TXNCON, temp);
}

void set_not_encrypt(void){
  short int temp = 0;

  temp = read_ZIGBEE_short(TXNCON);
  temp = temp & (!0x02);              // mask for set not encrypt
  write_ZIGBEE_short(TXNCON, temp);
}

/*
 * Interframe spacing
 */
void set_IFS_recomended() {
  short int temp = 0;

  write_ZIGBEE_short(RXMCR, 0x93);    // Min SIFS Period

  temp = read_ZIGBEE_short(TXPEND);
  temp = temp | 0x7C;                 // MinLIFSPeriod
  write_ZIGBEE_short(TXPEND, temp);

  temp = read_ZIGBEE_short(TXSTBL);
  temp = temp | 0x90;                 // MinLIFSPeriod
  write_ZIGBEE_short(TXSTBL, temp);

  temp = read_ZIGBEE_short(TXTIME);
  temp = temp | 0x31;                 // TurnaroundTime
  write_ZIGBEE_short(TXTIME, temp);
}

void set_IFS_default() {
  short int temp = 0;

  write_ZIGBEE_short(RXMCR, 0x75);    // Min SIFS Period

  temp = read_ZIGBEE_short(TXPEND);
  temp = temp | 0x84;                 // Min LIFS Period
  write_ZIGBEE_short(TXPEND, temp);

  temp = read_ZIGBEE_short(TXSTBL);
  temp = temp | 0x50;                 // Min LIFS Period
  write_ZIGBEE_short(TXSTBL, temp);

  temp = read_ZIGBEE_short(TXTIME);
  temp = temp | 0x41;                 // Turnaround Time
  write_ZIGBEE_short(TXTIME, temp);
}

/*
 * Reception mode
 */
void set_reception_mode(short int r_mode) { // 1 normal, 2 error, 3 promiscuous mode
  short int temp = 0;

  switch(r_mode) {
   case 1: {
     temp = read_ZIGBEE_short(RXMCR);      // normal mode
     temp = temp & (!0x03);                // mask for normal mode
```

```
    write_ZIGBEE_short(RXMCR, temp);
   }
   break;

   case 2: {
     temp = read_ZIGBEE_short(RXMCR);      // error mode
     temp = temp & (!0x01);                // mask for error mode
     temp = temp | 0x02;                   // mask for error mode
     write_ZIGBEE_short(RXMCR, temp);
   }
   break;

   case 3: {
     temp = read_ZIGBEE_short(RXMCR);      // promiscuous mode
     temp = temp & (!0x02);                // mask for promiscuous mode
     temp = temp | 0x01;                   // mask for promiscuous mode
     write_ZIGBEE_short(RXMCR, temp);
   }
   break;
  }
}

/*
 *  Frame format filter
 */
void set_frame_format_filter(short int fff_mode) {   // 1 all frames, 2 command only, 3
data only, 4 beacon only
  short int temp = 0;

  switch(fff_mode) {
   case 1: {
     temp = read_ZIGBEE_short(RXFLUSH);      // all frames
     temp = temp & (!0x0E);                  // mask for all frames
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 2: {
     temp = read_ZIGBEE_short(RXFLUSH);      // command only
     temp = temp & (!0x06);                  // mask for command only
     temp = temp | 0x08;                     // mask for command only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 3: {
     temp = read_ZIGBEE_short(RXFLUSH);      // data only
     temp = temp & (!0x0A);                  // mask for data only
     temp = temp | 0x04;                     // mask for data only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;

   case 4: {
     temp = read_ZIGBEE_short(RXFLUSH);      // beacon only
     temp = temp & (!0x0C);                  // mask for beacon only
     temp = temp | 0x02;                     // mask for beacon only
     write_ZIGBEE_short(RXFLUSH, temp);
   }
   break;
  }
}

/*
 *  Flush RX FIFO pointer
 */
void flush_RX_FIFO_pointer() {
  short int temp;

  temp = read_ZIGBEE_short(RXFLUSH);
  temp = temp | 0x01;                        // mask for flush RX FIFO
  write_ZIGBEE_short(RXFLUSH, temp);
}
```

```
/*
 * Address
 */
void set_short_address(short int * address) {
  write_ZIGBEE_short(SADRL, address[0]);
  write_ZIGBEE_short(SADRH, address[1]);
}

void set_long_address(short int * address) {
  short int i = 0;

  for(i = 0; i < 8; i++) {
    write_ZIGBEE_short(EADR0 + i, address[i]);   // 0x05 address of EADR0
  }
}

void set_PAN_ID(short int * address)
{
    write_ZIGBEE_short(PANIDL, address[0]);
    write_ZIGBEE_short(PANIDH, address[1]);
}

/*
 * Wake
 */
void set_wake_from_pin()
{
  short int temp = 0;

  WAKE = 0;

  temp = read_ZIGBEE_short(RXFLUSH);
  temp = temp | 0x60;                      // mask
  write_ZIGBEE_short(RXFLUSH, temp);

  temp = read_ZIGBEE_short(WAKECON);
  temp = temp | 0x80;
  write_ZIGBEE_short(WAKECON, temp);
}
void pin_wake()
{
  WAKE = 1;
  __delay_ms(5);
}
void pin_wake_up()
{
    WAKE = 1;
    write_ZIGBEE_short(RFCTL, 0x04); // RF state machine reset
    write_ZIGBEE_short(RFCTL, 0x00);
    __delay_ms(5);
}
/*
 * PLL
 */
void enable_PLL() {
  write_ZIGBEE_long(RFCON2, 0x80);       // mask for PLL enable
}

void disable_PLL() {
  write_ZIGBEE_long(RFCON2, 0x00);       // mask for PLL disable
}

/*
 * Tx power
 */
void set_TX_power(int power)  // 0-31 possible variants
{
    if((power < 0) || (power > 31))
    {
        power = 31;
        }
```

```
  power = 31 - power;                                     // 0 max, 31 min -> 31 max, 0
min
  power = ((power & 0b00011111) << 3) & 0b11111000;       // calculating power
  write_ZIGBEE_long(RFCON3, power);
}

/*
 * Init ZIGBEE module
 */
void init_ZIGBEE_basic() {
  write_ZIGBEE_short(PACON2, 0x98);   // Initialize FIFOEN = 1 and TXONTS = 0x6
  write_ZIGBEE_short(TXSTBL, 0x95);   // Initialize RFSTBL = 0x9
  write_ZIGBEE_long(RFCON1, 0x01);    // Initialize VCOOPT = 0x01
  enable_PLL();                       // Enable PLL (PLLEN = 1)
  write_ZIGBEE_long(RFCON6, 0x90);    // Initialize TXFIL = 1 and 20MRECVR = 1
  write_ZIGBEE_long(RFCON7, 0x80);     // Initialize SLPCLKSEL = 0x2 (100 kHz Internal
oscillator)
  write_ZIGBEE_long(RFCON8, 0x10);    // Initialize RFVCO = 1
  write_ZIGBEE_long(SLPCON1, 0x21);   // Initialize CLKOUTEN = 1 and SLPCLKDIV = 0x01
}

void init_ZIGBEE_nonbeacon() {
  init_ZIGBEE_basic();
  set_CCA_mode(1);     // Set CCA mode to ED and set threshold
  set_RSSI_mode(2);    // RSSI2 mode
  enable_interrupt();  // Enables all interrupts
  set_channel(11);     // Channel 11
  RF_reset();
}
/*
 * Reset functions
 */
void pin_reset()        // Reset from pin
{
    RST = 0;            // activate reset
    __delay_ms(5);
    RST = 1;            // deactivate reset
    __delay_ms(5);
}

void PWR_reset()
{
    write_ZIGBEE_short(SOFTRST, 0x04);   // 0x04  mask for RSTPWR bit
    write_ZIGBEE_short(SLPACK, 0x80);    // 0x80 mask for put sleep immediately
}

void BB_reset()
{
    write_ZIGBEE_short(SOFTRST, 0x02);   // 0x02 mask for RSTBB bit
}

void MAC_reset()
{
    write_ZIGBEE_short(SOFTRST, 0x01);   // 0x01 mask for RSTMAC bit
}

void software_reset() // PWR_reset,BB_reset and MAC_reset at once
{
    write_ZIGBEE_short(SOFTRST, 0x07);
}

void RF_reset()
{
    short int temp = 0;
    temp = read_ZIGBEE_short(RFCTL);
    temp = temp | 0x04;                  // mask for RFRST bit
    write_ZIGBEE_short(RFCTL, temp);
    temp = temp & (!0x04);               // mask for RFRST bit
    write_ZIGBEE_short(RFCTL, temp);
    __delay_ms(1);
}
```

**Anexo C:** *Software* do sistema de monitorização no tanque.

```
/*****************************************************************
*Autor: José Evaristo
* Data: 07.11.2014
*Descrição: Sensor EC, Temp, pH and Sensor Nivel
* Versão: 1.0.0
*****************************************************************/

/* Includes ******************************************************/
#include "stm32f4xx.h"
#include "stdio.h"

/* Variaveis Globais *********************************************/
# define VCC 2.94 // Voltagem de referência
// Wave DAC definitions
#define   OUT_FREQ           3000                                    // Output waveform
frequency
#define   SINE_RES          128                                    // Waveform resolution
#define   DAC_DHR12R1_ADDR  0x40007408                             // DMA writes into this
reg on every request
#define   CNT_FREQ           42000000                                // TIM6 counter clock
(prescaled APB1)
#define   TIM_PERIOD        ((CNT_FREQ)/((SINE_RES)*(OUT_FREQ))) // Autoreload reg value

const uint16_t function[SINE_RES] = { 2048, 2145, 2242, 2339, 2435, 2530, 2624, 2717,
2808, 2897,
                                      2984, 3069, 3151, 3230, 3307, 3381, 3451, 3518,
3581, 3640,
                                      3696, 3748, 3795, 3838, 3877, 3911, 3941, 3966,
3986, 4002,
                                      4013, 4019, 4020, 4016, 4008, 3995, 3977, 3954,
3926, 3894,
                                      3858, 3817, 3772, 3722, 3669, 3611, 3550, 3485,
3416, 3344,
                                      3269, 3191, 3110, 3027, 2941, 2853, 2763, 2671,
2578, 2483,
                                      2387, 2291, 2194, 2096, 1999, 1901, 1804, 1708,
1612, 1517,
                                      1424, 1332, 1242, 1154, 1068, 985, 904, 826, 751,
679,
                                      610, 545, 484, 426, 373, 323, 278, 237, 201, 169,
                                      141, 118, 100, 87, 79, 75, 76, 82, 93, 109,
                                      129, 154, 184, 218, 257, 300, 347, 399, 455, 514,
                                      577, 644, 714, 788, 865, 944, 1026, 1111, 1198,
1287,
                                      1378, 1471, 1565, 1660, 1756, 1853, 1950, 2047 };

/* Function prototype ********************************************/
void GPIO_Config(void);        // Input/Output pin configuration
void ADC_Config(void);                 // ADC configuration
void USART_Config(void);               // USART configuration
void TIM2_Config (void);               // Timer configuration
void Temperature_Sensor(void); // Temperature sensor
void Conductivity_Sensor(void);// Electro Conductivity Sensor
void pH_Sensor(void);                                  // pH Sensor
void DAC_Pin_Config(void);            // GAPIO A4 DAC Output configuration
static void TIM6_Config(void); // Timer 6 configuration for DAC
static void DAC1_Config(void); // DAC configuration

/*****************************************************************************/
/*            STM32F4xx Peripherals Interrupt Handlers                       */
/*****************************************************************************/
       /**
  * @brief  Nivel sensor Interrupt
  * @param  None
  * @retval None
  */
void TIM2_IRQHandler (void)
       {
              if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
              {
```

```
                        // Limpa a Flag correspondente à interrupção
                        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);

                        Temperature_Sensor();

                        Conductivity_Sensor();

                        pH_Sensor();
            }
}
/**
  * @brief  This function handles External line 0 interrupt request.
  * @param  None
  * @retval None
  */
void EXTI0_IRQHandler(void)
{
       char buffer[30]="Atencao: Nivel Maximo de Agua\n", i=0;
  if (EXTI_GetITStatus(EXTI_Line0) != RESET)
  {
    /* Clear the EXTI line 0 pending bit */
    EXTI_ClearITPendingBit(EXTI_Line0);

    if (GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0))
            {
                    GPIO_SetBits(GPIOD, GPIO_Pin_13);

                    while(buffer[i]!='\0')
                            {
                                    USART_SendData(USART2,buffer[i]);
                                    while(!USART_GetFlagStatus(USART2,USART_FLAG_TXE));
                                    i++;
                                    }
            }
    else
      GPIO_ResetBits(GPIOD, GPIO_Pin_13);
  }
}

/*******************************************************************************
**********************
  * @brief  Temperature Sensor PT1000
  * @param  None
  * @retval None
  */
void Temperature_Sensor(void)
{
                    unsigned int valor=0;
                    float Temperatura=0, Vout=0, RTD=0;
                    char buffer_Temp[10],i=0;

                    // Inicia a conversão do valor de tensão
          ADC_SoftwareStartConv(ADC2); // Pin PC1

                    // Espera que a conversão seja feita
               while(!ADC_GetFlagStatus(ADC2, ADC_FLAG_EOC));

                    // Atribui à variavel valor a tensao recolhida pela adc respetiva
                    valor = ADC_GetConversionValue(ADC2);

                    Vout = ((VCC * valor)/4095);    // Voltagem à saída do divisor de
tensão

                    RTD  =  (((Vout/VCC)*1000)/(1-(Vout/VCC)));     //  Conversao  da
voltagem em resistência RTD=(Vout/Vin*R1)/(1-Vout/Vin)

                    Temperatura = (0.2592 * RTD) - 259.26;            // Conversao de
Resistência em Temperatura ºC

                    // Concatenação do valor
                    sprintf(buffer_Temp,"Temperatura da Agua: %.1f *C\n",Temperatura);

                    //Envia valor da Temperatura
```

```
                        while(buffer_Temp[i]!='\0')
                                        {
                                                USART_SendData(USART2,buffer_Temp[i]);

        while(!USART_GetFlagStatus(USART2,USART_FLAG_TXE));
                                                i++;
                                                }
}
/*******************************************************************************
**********************
  * @brief  Electrical resistivity and conductivity Sensor
  * @param  None
  * @retval None
  */
void Conductivity_Sensor(void)
{
                unsigned int valor=0;
                float Vout=0;
                char buffer_EC[12], i=0;

                // Inicia a conversão do valor de tensão
                ADC_SoftwareStartConv(ADC3);  // Pin PC2

                // Espera que a conversão seja feita
                while(!ADC_GetFlagStatus(ADC3, ADC_FLAG_EOC));

                // Atribui à variavel valor a tensao recolhida pela adc respetiva
                valor = ADC_GetConversionValue(ADC3);

                Vout = ((VCC * valor)/4095); // Conversão do valor adquirido pela ADC em
voltagem

                // Concatenação do valor
                        sprintf(buffer_EC,"Condutividade da Solucao: %d V\n",valor);

                        //Envia valor da Temperatura
                        while(buffer_EC[i]!='\0')
                                        {
                                                USART_SendData(USART2,buffer_EC[i]);

        while(!USART_GetFlagStatus(USART2,USART_FLAG_TXE));
                                                i++;
                                                }
}
/*******************************************************************************
**********************
  * @brief  pH sensor aquiring data
  * @param  None
  * @retval None
  */
void pH_Sensor(void)
{
                unsigned int valor=0;
                float Vout=0;
                char buffer_pH[12], i=0;

                // Inicia a conversão do valor de tensão
                ADC_SoftwareStartConv(ADC1);  // Pin PC0

                // Espera que a conversão seja feita
                while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));

                // Atribui à variavel valor a tensao recolhida pela adc respetiva
                valor = ADC_GetConversionValue(ADC1);

                Vout = ((VCC * valor)/4095); // Conversão do valor adquirido pela ADC em
voltagem

                // Concatenação do valor
                        sprintf(buffer_pH,"O pH da Solucao: %d V\n",valor);

                        //Envia valor da Temperatura
                        while(buffer_pH[i]!='\0')
```

```
                                        {
                                                USART_SendData(USART2,buffer_pH[i]);

        while(!USART_GetFlagStatus(USART2,USART_FLAG_TXE));
                                                i++;
                                                }
}
/* Main ********************************************************/
int main ()
{
        // Function Inialization
        GPIO_Config();
        ADC_Config();
        USART_Config();
        TIM2_Config();
  DAC_Pin_Config();
        TIM6_Config();
  DAC1_Config();

        while(1)
        {
}
}

/* Funções *****************************************************************/
void GPIO_Config(void)
{
        // GPIO Structure declaration
        GPIO_InitTypeDef GPIO_InitStruct;
        // NVIC structure declaration
        NVIC_InitTypeDef NVIC_InitStructure;
        // EXTI structure declaration
        EXTI_InitTypeDef EXTI_InitStructure;

        //Enabling GPIO peripheral clock
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
        // Enable SYSCFG clock
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

        // Setting peripheral properties specification
        GPIO_InitStruct.GPIO_Pin = GPIO_Pin_13; //LED3 GPIO pin
        GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF; //output
        GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz; // clock speed
        GPIO_InitStruct.GPIO_OType = GPIO_OType_PP; //push/pull
        GPIO_InitStruct.GPIO_PuPd   =   GPIO_PuPd_NOPULL;   //pullup/pulldown   resistor
inactive
        // Setting GPIO peripheral corresponding bits
        GPIO_Init(GPIOD, &GPIO_InitStruct);

        // GPIO peripheral properties specification for B1 (Pin0 GPIOA)
        GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
  GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOA, &GPIO_InitStruct);
        // Setting GPIO peripheral corresponding bits for B1
        GPIO_Init(GPIOA, &GPIO_InitStruct);

        // Connect EXTI Line0 to PA0 pin
  SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);

        // Configure EXTI Line0, Interrupt Mode, Enable on rising edge
  EXTI_InitStructure.EXTI_Line = EXTI_Line0;
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        // Stting EXTI Line 0
  EXTI_Init(&EXTI_InitStructure);

        // Enable the EXTI0 gloabal Interrupt
        NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x00;
```

```
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_Init(&NVIC_InitStructure);
}
/*******************************************************************************
***********
  * @brief  ADC configuration
  * @param  None
  * @retval None
  */
void ADC_Config(void)
{
	/* Inicialização das estruturas GPIO e ADC */
		GPIO_InitTypeDef GPIO_InitStruct;
		ADC_InitTypeDef ADC_InitStructure;
		ADC_CommonInitTypeDef ADC_CommonInitStruct;

	/* Ativar os relogios dos periféricos */
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC2, ENABLE);
		RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC3, ENABLE);
		RCC_AHB1PeriphClockCmd(RCC_AHB1ENR_GPIOCEN,ENABLE);


	/* Configuração do Pinos do porto C da estrutura GPIO */
		GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2;
		GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AN;
		GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
	  GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
	  GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
	  // Inicializa a estrutura do GPIO
	  GPIO_Init(GPIOC, &GPIO_InitStruct);

	/* Configuração da estrutura da ADC */
		ADC_InitStructure.ADC_Resolution = ADC_Resolution_12b;
		ADC_InitStructure.ADC_ScanConvMode = DISABLE;
		ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
		ADC_InitStructure.ADC_ExternalTrigConvEdge                                    =
ADC_ExternalTrigConvEdge_None;
		ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T1_CC1;
		ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
		ADC_InitStructure.ADC_NbrOfConversion = 1; //Usa um canal
		// Inicializa a estrutura ADC1
		ADC_Init(ADC1,&ADC_InitStructure);
		ADC_Init(ADC2,&ADC_InitStructure);
		ADC_Init(ADC3,&ADC_InitStructure);

	/* Configuração dos periféricos da ADC */
		ADC_CommonInitStruct.ADC_Mode = ADC_Mode_Independent;
		ADC_CommonInitStruct.ADC_Prescaler = ADC_Prescaler_Div2;
		ADC_CommonInitStruct.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
		ADC_CommonInitStruct.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_5Cycles;
		// Inicializa a estrutura
		ADC_CommonInit(&ADC_CommonInitStruct);

	/* Ativar a ADC */
		ADC_Cmd(ADC1, ENABLE);
		ADC_Cmd(ADC2, ENABLE);
		ADC_Cmd(ADC3, ENABLE);
	/* Seleção do canal que vai ser lido */
		ADC_RegularChannelConfig(ADC1,ADC_Channel_10,1,ADC_SampleTime_144Cycles);
		ADC_RegularChannelConfig(ADC2,ADC_Channel_11,1,ADC_SampleTime_144Cycles);
		ADC_RegularChannelConfig(ADC3,ADC_Channel_12,1,ADC_SampleTime_144Cycles);

}
/*******************************************************************************
******************
  * @brief  USART configuration
  * @param  None
  * @retval None
  */
void USART_Config(void)
{
	/* Inicialização das estruturas GPIO, USART */
```

```
	GPIO_InitTypeDef GPIO_InitStruct;
	USART_InitTypeDef USART_InitStruct;

	/* Ativar o relogio dos periféricos */
		RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
		RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

	/* Configuração do Pinos do porto A da estrutura GPIO */
		GPIO_InitStruct.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3; // Pin 2 (TX) Pin3
(RX)
		GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF; // alternate function
		GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz; // clock speed
		GPIO_InitStruct.GPIO_OType = GPIO_OType_PP; // push/pull
		GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_UP; // pullup/pulldown resistors
active
		// Inicializa a estrutura do GPIO
		GPIO_Init(GPIOA, &GPIO_InitStruct);

	/* Associar o periférico USART ao pinos do GPIO */
		GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_USART2);
		GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_USART2);

	/* Configuração da estrutura da USART2 */
		USART_InitStruct.USART_BaudRate = 9600; // baudrate
		USART_InitStruct.USART_WordLength = USART_WordLength_8b;
// frame size 8 bits (standard)
		USART_InitStruct.USART_StopBits = USART_StopBits_1;
// 1 stop bit (standard)
		USART_InitStruct.USART_Parity = USART_Parity_No;
// no parity bit (standard)
		USART_InitStruct.USART_HardwareFlowControl =
USART_HardwareFlowControl_None; // no flow control (standard)
		USART_InitStruct.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
// enable transmitter
		// Inicializa a estrutura da USART2
		USART_Init(USART2, &USART_InitStruct);

	/* Ativar a USART */
		USART_Cmd(USART2, ENABLE);
		USART_SendData(USART2,'S');
}
/*******************************************************************************
  * @brief  Timer Configuration
  * @param  None
  * @retval None
  */
void TIM2_Config (void)
{
	/* Inicialização das estruturas TIM e Interrupção */
		TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
		NVIC_InitTypeDef NVIC_InitStructure;

	/* Ativar o relogio dos periféricos */
		RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,ENABLE);

	/* Configuração da estrutura do TIMER2 */
		TIM_TimeBaseInitStruct.TIM_Prescaler = 42000-1;
// (F_clock_tim = F_clock_mcu/Prescaler= 84MHz/42kHz = 2kHz)
		TIM_TimeBaseInitStruct.TIM_Period = 18000-1;
// Timer_trigger = Period / F_clock_tim = 2000 cycles / 2 kHz = 1 sec
		TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
// Divide clock by 1
		TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
		// Inicializa a estrutura da TIMER 2
		TIM_TimeBaseInit(TIM2,&TIM_TimeBaseInitStruct);

	/* Configuração da estrutura dalinha de interrupção do TIMER2 */
		NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
		NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
		NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x01;
		NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
		// Inicializa a estrutura de interrupção
		NVIC_Init(&NVIC_InitStructure);
```

```
        /* Ativar a interrupção do TIMER2 */
                TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);

        /* Ativar o TIMER2 */
                TIM_Cmd(TIM2,ENABLE);
}
/*******************************************************************************
  * @brief  GAPIO A4 DAC Output Pin configuration
  * @param  None
  * @retval None
  */
void DAC_Pin_Config(void)
{
        GPIO_InitTypeDef gpio_A;

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA1, ENABLE);

  gpio_A.GPIO_Pin  = GPIO_Pin_4;
  gpio_A.GPIO_Mode = GPIO_Mode_AN;
  gpio_A.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &gpio_A);
}
/*
  * @brief  Timer 6 for DAC configuration
  * @param  None
  * @retval None
  */
static void TIM6_Config(void)
{
  TIM_TimeBaseInitTypeDef TIM6_TimeBase;

  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM6, ENABLE);

  TIM_TimeBaseStructInit(&TIM6_TimeBase);
  TIM6_TimeBase.TIM_Period        = (uint16_t)TIM_PERIOD;
  TIM6_TimeBase.TIM_Prescaler     = 0;
  TIM6_TimeBase.TIM_ClockDivision = 0;
  TIM6_TimeBase.TIM_CounterMode   = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM6, &TIM6_TimeBase);
  TIM_SelectOutputTrigger(TIM6, TIM_TRGOSource_Update);

  TIM_Cmd(TIM6, ENABLE);
}
/*
  * @brief  DAC configuration
  * @param  None
  * @retval None
  */
static void DAC1_Config(void)
{
  DAC_InitTypeDef DAC_INIT;
  DMA_InitTypeDef DMA_INIT;

  DAC_INIT.DAC_Trigger        = DAC_Trigger_T6_TRGO;
  DAC_INIT.DAC_WaveGeneration = DAC_WaveGeneration_None;
  DAC_INIT.DAC_OutputBuffer   = DAC_OutputBuffer_Enable;
  DAC_Init(DAC_Channel_1, &DAC_INIT);

  DMA_DeInit(DMA1_Stream5);
  DMA_INIT.DMA_Channel            = DMA_Channel_7;
  DMA_INIT.DMA_PeripheralBaseAddr = (uint32_t)DAC_DHR12R1_ADDR;
  DMA_INIT.DMA_Memory0BaseAddr    = (uint32_t)&function;
  DMA_INIT.DMA_DIR                = DMA_DIR_MemoryToPeripheral;
  DMA_INIT.DMA_BufferSize         = SINE_RES;
  DMA_INIT.DMA_PeripheralInc      = DMA_PeripheralInc_Disable;
  DMA_INIT.DMA_MemoryInc          = DMA_MemoryInc_Enable;
  DMA_INIT.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
  DMA_INIT.DMA_MemoryDataSize     = DMA_MemoryDataSize_HalfWord;
  DMA_INIT.DMA_Mode               = DMA_Mode_Circular;
  DMA_INIT.DMA_Priority           = DMA_Priority_High;
```

```
  DMA_INIT.DMA_FIFOMode           = DMA_FIFOMode_Disable;
  DMA_INIT.DMA_FIFOThreshold      = DMA_FIFOThreshold_HalfFull;
  DMA_INIT.DMA_MemoryBurst        = DMA_MemoryBurst_Single;
  DMA_INIT.DMA_PeripheralBurst    = DMA_PeripheralBurst_Single;
  DMA_Init(DMA1_Stream5, &DMA_INIT);

  DMA_Cmd(DMA1_Stream5, ENABLE);
  DAC_Cmd(DAC_Channel_1, ENABLE);
  DAC_DMACmd(DAC_Channel_1, ENABLE);
}
```