INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Lisímetro Inteligente com Monitorização da Cultura e Ambiente

Uma Aproximação IoT

Carlos Manuel dos Santos Almeida

2024

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Lisímetro Inteligente com Monitorização da Cultura e Ambiente

Uma Aproximação IoT

Elaborado por:

Carlos Manuel dos Santos Almeida

Orientado por:

Professor Doutor José Jasnau Caeiro, IPBeja

Professor Doutor João Carlos Martins, IPBeja

Dissertação de Mestrado

Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

2024

Agradecimentos

A realização desta dissertação de mestrado contou com diversos apoios e incentivos, sem os quais não seria possível a sua concretização.

Em primeiro lugar quero agradecer aos meus orientadores pela excelente orientação, incentivo, apoio e total disponibilidade para a realização deste trabalho.

Agradeço em particular ao Professor Doutor José Jasnau Caeiro, pelos seus ensinamentos e pelos desafios que me colocou, os quais me permitiram adquirir novas competências. Entre os diversos desafios, destaco o desafio de realizar um artigo científico para apresentar numa conferência de IoT.

Agradeço em particular ao Professor Doutor João Carlos Martins, a preciosa ajuda nas revisões da dissertação, assim como na realização do artigo científico, nomeadamente na sua tradução e revisão.

Agradeço também ao Dr. João Santos pelo seu contributo na realização do artigo científico.

Agradeço a todos os professores do curso de mestrado em Internet das Coisas, os conhecimentos que transmitiram ao longo do curso, esses conhecimentos foram fundamentais para conseguir concluir este trabalho.

Agradeço a todos os meus colegas do curso de mestrado a troca de conhecimentos nas diversas áreas, em particular àqueles com quem fiz trabalhos de grupo.

Agradeço ao IP Beja por ter um curso de mestrado em Internet das Coisas em regime Pós-laboral e *blended-learning*, só deste modo foi possível o acompanhamento das aulas, e a concretização deste trabalho.

Para finalizar quero agradecer à minha esposa Rita Pereira e ao meu filho Tiago Almeida, o apoio, e a compreensão no decorrer deste trabalho.

Resumo

Lisímetro Inteligente com Monitorização da Cultura e Ambiente

Uma Aproximação IoT

Adotando uma abordagem IoT apresentamos um modelo de lisímetro inteligente, melhorado com análise de pragas e o estado da cultura. Além da obtenção do balanço tradicional de evaporação-transpiração, o lisímetro mede parâmetros adicionais tais como a temperatura e humidade do solo, a diferentes profundidades; temperatura e humidade do ar; exposição à luz solar (visível e infravermelho). Além disso, o sistema faz a captura imagens de alta resolução da cultura alvo. Estas imagens são processadas localmente, para redução de dados que são armazenados posteriormente numa plataforma remota. O objetivo principal é a monitorização e com vista ao aumento da produtividade global da cultura. Este lisímetro também fornece dados para um sistema global de monitorização de recursos hídricos que integra informações de várias fontes: outros lisímetros, estações meteorológicas, sistemas de monitorização da qualidade da água, *etc.*. O resultado deste trabalho culminou no desenvolvimento e teste de um protótipo funcional.

Palavras-chave: Lisímetros Inteligentes, Agricultura de Precisão, IdC, Gestão da Água, Evapotranspiração.

Abstract

Smart Lysimeter with Crop and Environment Monitoring

An IoT Approach

A model of a smart lysimeter, adopting an IoT approach, enhanced with pest and crop state analysis is presented. Besides the measurement of the traditional evaporation-transpiration balance, the lysimeter senses additional parameters like the soil temperature and humidity at different depths; air temperature and humidity; sunlight exposition (visible and infrared). Additionally, the system can capture high-resolution images of the target culture. These images are locally processed for data reduction and the main features are stored in a remote platform afterwards. The main goal is the monitoring and enhancement of the global crop yield. This lysimeter also provides data for a global water resources system that integrates information from several sources: lysimeters, weather stations, water quality monitoring systems, *etc.*. The result of this work culminated in the development and testing a functional prototype.

Keywords: Smart Lysimeter, Precision Agriculture, IoT, Water Management, Evapotranspiration.

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Abreviaturas e Siglas

ADC	*Analog to Digital Converter*
AI	*Artificial Intelligence*
ANN	*Artificial Neural Network*
ARM	*Advanced RISC Machine*
CBB	*Cassava Bacterial Blight*
CBSD	*Cassava Brown Streak Disease*
CCV	*Color Coherence Vectors*
CCS	*Code Composer Studio*
CCV	*Color Coherence Vectors*
CGM	*Cassava green mite*
CMD	*Cassava Mosaic Disease*
CNC	*Computer Numerical Control*
CNN	*Convolutional Neural Network*
CoAP	*Constrained Application Protocol*
CPU	*Central Processing Unit*
CSI	*Camera Serial Interface*
DCNN	*Deep Convolutional Neural Network*
DL	*Deep Learning*
DT	*Decision Tree*
EDA	*Electronics Design Automation*
ERT	*Extremely Randomized Trees*
ET	Evapotranspiração

FCM	*Fuzzy C-means Clustering*
FNR	*False Negative Rate*
GCH	*Global Color Histogram*
GMM	*Gaussian Mixture Modelling*
GPIO	*General Purpose Input/Output*
GPRS	*General Packet Radio Services*
HOG	*Histograms of Oriented Gradients*
HSV	*Hue, Saturation, Value*
I2C	*Inter-Integrated Circuit*
IC	*Integrated Circuit*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
KNN	*K-Nearest Neighbour*
LBP	*Local Binary Pattern*
LPM	*Low Power Mode*
MCU	*Microcontroller Unit*
MQTT	*Message Queuing Telemetry Transport*
NB	*Naive Bayes Algorithm*
MIA	*Make Intelligent Applications*
ML	*Machine Learning*
NoSQL	*Not only SQL*
NTP	*Network Time Protocol*
ORB	*Oriented FAST and Rotated BRIEF*
PCA	*Principal Component Analysis*
PCB	*Printed Circuit Board*
PWA	*Progressive Web App*
RF	*Random Forest*

RFID *Radio Frequency Identification*

RGB *Red Green Blue*

RMSP *Root Mean Square Propagation*

RPi Raspberry Pi

RSA *Rivest-Shamir-Adleman*

SBC *Singe Board Computer*

SFTP *Secure File Transfer Protocol*

SGD *Stochastic Gradient Descent*

SGLDM *Spatial Gray Level Dependence Matrices*

SIFT *Scale-Invariant Feature Transform*

SMS *Short Message Service*

SO Sistema Operativo

SoC *System on Chip*

SSH *Secure Shell*

SURF *Speed-ed Up Robust Features*

SVC *Support Vector Classifier*

SVM emphSupport Vectors Machines

TPR *True Positive Rate)*

UART *Universal Asynchronous Receiver-Transmitter*

URL *Uniform Resource Locator*

VM *Virtual Machine*

WLAN *Wireless Local Area Network*

WSN *Wireless Sensor Network*

ZB *ZigBee*

Capítulo 1

Introdução

O tema da dissertação é o projeto e o desenvolvimento de um lisímetro tendo por base o paradigma da Internet das Coisas (*Internet of Things* (IoT)).

O lisímetro é um instrumento muito utilizado na agricultura e tem como função base a medição da Evapotranspiração (ET) de uma cultura.

Os lisímetros medem a evapotranspiração (ET) do solo e das plantas. São usados na agricultura há alguns séculos e têm como objetivo principal quantificar a necessidade hídrica das plantas de forma a fornecer apenas a quantidade de água necessária ao seu crescimento saudável. Sendo a água fundamental para o desenvolvimento socioeconómico, para a produção de energia e alimentos, para a construção de ecossistemas saudáveis e para a sobrevivência da espécie humana. O seu uso criterioso é essencial para fazer frente às alterações climáticas, servindo como elo crucial entre a sociedade e o meio ambiente [1].

1.1 Enquadramento da Dissertação

A evapotranspiração é a medida da perda de água do solo por evaporação e a perda de água da planta por transpiração. Numa cultura, temos um equilíbrio hídrico entre a entrada e saída de água. Na entrada do sistema temos a água da chuva, da rega e devida à condensação. Na saída temos a água drenada, a água perdida por evaporação e transpiração. A Figura 1.1 apresenta o balanço dos ciclos de água numa cultura.

Usando uma abordagem IoT, é descrito a estrutura e o desenvolvimento de um lisímetro acrescentado de um conjunto de sensores que permitem medir outros parâmetros relevantes para a agricultura, para além da evapotranspiração. Os parâmetros adicionais medidos pelo lisímetro são: a temperatura e humidade do solo a diferentes profundidades; temperatura e humidade do ar; exposição à luz solar (visível e infravermelho). O sistema, permite também

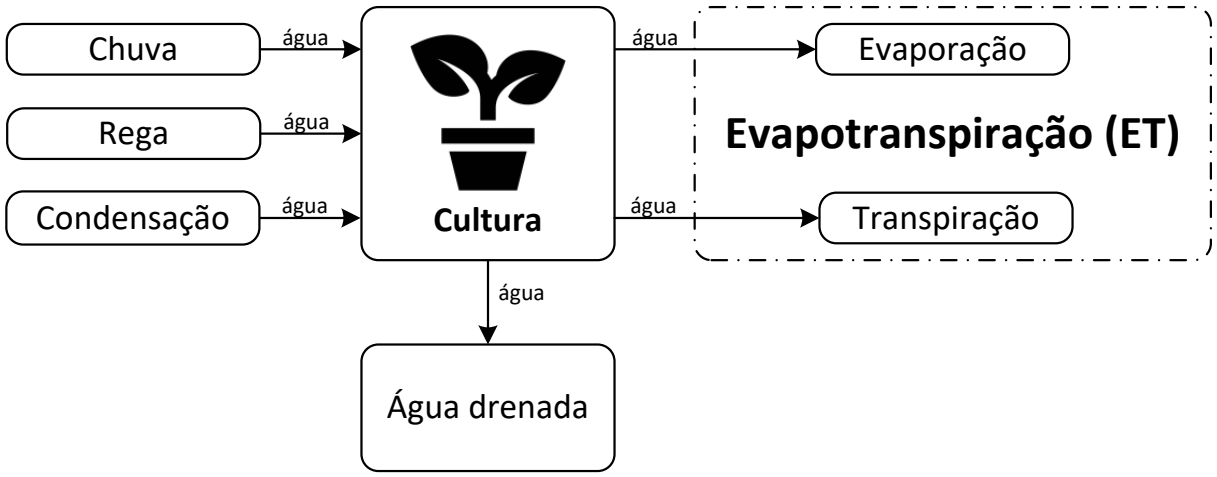


Figura 1.1: Diagrama da Evapotranspiração.

capturar imagens de alta resolução da cultura alvo. Essas são processadas localmente para redução da quantidade de dados, que são armazenados posteriormente numa plataforma remota.

O objetivo principal do lisímetro é a recolha de dados com vista ao aumento da produtividade global da cultura, e a poupança de água. Este lisímetro também fornece dados que são integrados num sistema global de gestão e monitorização de recursos hídricos, que integra informação proveniente de diversas fontes: outros lisímetros, estações meteorológicas, sistemas de monitorização da qualidade da água, sistemas de informação geográfica.

A Monitorização de Culturas com IoT

O sistema descrito recorre a sensores de imagem que, juntamente com microcomputadores (*Singe Board Computer* (SBC)), e sensores ambientais (temperatura, humidade e luminosidade) permitem monitorizar o desenvolvimento de culturas agrícolas e/ou eventualmente prevenir e detetar o surgimento de doenças e pragas. O processamento destes dados será feito na *Cloud* recorrendo a técnicas de *Machine Learnig* e *Deep Learning*. Prevê-se que após uma prévia aprendizagem dos modelos, seja possível detetar atempadamente problemas com a cultura de modo a gerar e enviar avisos necessários ao agricultor. Este sistema de monitorização de culturas têm uma componente de processamento digital de imagem e outra de aprendizagem automática.

1.2 Contributos da Dissertação

A dissertação apresenta como ponto principal a concepção dum sistema de monitorização de colheitas baseado numa arquitectura IoT. Este sistema foi desenvolvido a partir do trabalho de análise da documentação publicada recentemente sobre sistemas similares na sua natureza. Os contributos com este trabalho, são os seguintes:

1. Reunião e análise de informação sobre: a arquitectura dos sistemas; grandezas monitorizadas; sensores e demais componentes electrónicos, microcomputadores e microcontroladores.

2. Projeto e desenvolvimento de uma arquitetura que combina a obtenção do valor da ET, com a aquisição de dados adicionais da cultura e do ambiente, e ainda a avaliação do estado de saúde da cultura através da aquisição e processamento de imagem, tudo num sistema único. Esta abordagem tem vantagens relativamente a sistemas não integrados, nomeadamente:

   a) Maior eficiência energética;

   b) Menor custo económico;

   c) Processamento de dados centralizado;

   d) Correlação de dados dos diversos sensores de cada lisímetro;

   e) Correlação de dados entre os diversos lisímetros;

   f) A aprendizagem obtida através de técnicas de *Deep Learning* (DL) num determinado lisímetro é partilhada com os restantes. Todos os lisímetros contribuem com os seus dados para uma melhor precisão global do sistema.

3. Concepção de um protótipo experimental de elevada precisão recorrendo às mais recentes tecnologias IoT, e com custos controlados. A realização experimental, permite:

   a) Comprovar a eficácia da arquitetura desenvolvida;

   b) Detetar e corrigir falhas na arquitetura do sistema;

   c) Detetar e corrigir falhas na arquitetura de hardware;

   d) Avaliar a robustez do sistema;

   e) Obter dados reais para análise posterior;

   f) Quantificar o custo real de implementação do sistema.

4. Analise do desempenho do protótipo experimental implementado, a qual permitiu:

    a) Comprovar a precisão dos sensores usados nos lisímetros;

    b) Avaliar a performance da gestão de energia, nomeadamente a autonomia das baterias;

    c) Avaliar a facilidade de acesso à informação gerada pelo sistema, através das interfaces criados.

Do trabalho de investigação realizado resultou a publicação e apresentação do artigo científico: "*Smart Lysimeter with Crop and Environment Monitoring, Enhanced with Pest and Crop Control*", na *4th IFIP International Internet of Things (IoT) Conference*, em Novembro de 2021.

Algumas figuras possuem texto em inglês pelo facto de terem sido incluídas no artigo apresentado em conferência internacional [2].

1.3 Estrutura do Documento

A dissertação está organizada em cinco capítulos, incluindo o capítulo inicial de introdução e o capítulo final das conclusões.

Capítulo 1 - Introdução

Neste capítulo é feito o enquadramento da dissertação, descreve como é feita a monitorização de culturas recorrendo ao paradigma IoT, apresenta os principais contributos da dissertação e por último a estrutura do documento.

Capítulo 2 - Estudo da Arte

É feito o levantamento do estado da arte seguindo-se uma análise dos mais recentes trabalhos científicos sobre lisímetros, sistemas de aquisição de imagem para deteção de doenças e pragas em plantas e sistemas para análise de colheitas.

Capítulo 3 - Arquitetura do Sistema Lisímetro

Apresenta a arquitetura proposta para o lisímetro baseada em IoT, detalhando a estrutura geral do lisímetro nos seus diversos níveis e, por fim, a arquitetura de hardware do sistema.

Capítulo 4 - Implementação Experimental

Descreve a implementação experimental do lisímetro e apresenta as especificações requeridas para o protótipo. Depois é descrita a implementação física do protótipo experimental. Esta implementarão é dividida em 4 partes: Módulo Lisímetro, Módulo Câmara, Módulo de Computação na Nuvem, a estrutura de suporte e, por último, são apresentados os aspetos experimentais.

Capítulo 5 - Conclusões

Apresenta as conclusões do trabalho desenvolvido e propostas para trabalhos futuros.

Capítulo 2

Estado da Arte

2.1 Introdução

O capítulo incide sobre o levantamento de lisímetros inteligentes, sistemas de aquisição de imagens de plantações e sistemas de processamento de análise de colheitas.

Foram analisados diversos artigos publicados em revistas científicas em conferências, e outros documentos provenientes de locais da Internet. A pesquisa de artigos científicos foi dividida em três partes:

1. Pesquisa de artigos sobre lisímetros com abordagem IoT;
2. Pesquisa de artigos sobre sistemas de aquisição e processamento de imagem usados na agricultura;
3. Pesquisa de artigos sobre sistema de análise de colheitas.

Quando é necessária uma estimativa precisa da ET, deve-se usar um lisímetro de pesagem. Estes sistemas são usados para calibrar outros sistemas baseados em *Machine Learning* (ML) para estimar a ET.

Num período de 10 anos verifica-se que houve grandes avanços nas técnicas de processamento de imagem, nomeadamente a adoção de técnicas de DL *Deep Learning* do tipo *Convolutional Neural Network* (CNN) em vez dos métodos convencionais, nomeadamente de ML.

A evolução das redes neuronais, nomeadamente a *Deep Convolutional Neural Network* (DCNN) , nos últimos cinco anos está a permitir criar sistemas mais precisos e exatos, e com menor necessidade de ajustes. Prevemos que os métodos convencionais terão tendência a desaparecer com o tempo e com a inovação tecnológica.

2.2 Lisímetros

Existem vários tipos de lisímetros, tendo em conta o objetivo em concreto a que se destinam, dependendo essencialmente do clima, disponibilidade de materiais, tecnologia envolvida, dimensões e custos envolvidos na construção. Os lisímetros podem ser essencialmente divididos em duas grandes categorias: pesagem (mecânica, electrónica, hidráulica e de flutuação), e não pesagem (volumétricos: drenagem e lençol freático). Geralmente são classificados em quatro grupos: drenagem, pesagem, flutuante, e hidráulico [3].

Desde o primeiro lisímetro até há cerca de uma década atrás, os lisímetros eram na maioria dos casos, usados por instituições de ensino e investigação, devido ao seu elevado custo. No entanto, o aparecimento de novas tecnologias, assim como novos materiais, veio permitir a construção de lisímetros de pesagem de alta tecnologia e de baixo custo. A tecnologia IoT veio permitir uma melhor gestão das culturas devido à redução de custos e aumento da eficiência, assim como o seu controlo automatizado. A IoT permite o acesso à monitorização das culturas em tempo real e com os dados recolhidos é possível gerar alertas precoces e atempados, e fornecer informação de apoio à decisão ao agricultor [4].

Um lisímetro de pesagem é um dispositivo normalmente constituído por dois vasos, um para o solo com as plantas, outro para a recolha da água drenada. O cálculo da evapotranspiração é obtido pela relação entre a entrada de água (chuva, rega, e condensação) e a água drenada (Figura 1.1).

Em [5] a ET das plantas no deserto é quantificada usando sensores de humidade simples colocados a várias profundidades e sensores de pressão hídrica associados a um microcontrolador (*Microcontroller Unit* (MCU)) [6] para controlar a rega e para manter a humidade do solo num determinado nível. Este lisímetro não recorre à pesagem de água.

Para quantificar a ET de grandes campos em [7], o sistema descrito combina dados de sensorização remota, juntamente com dados meteorológicos e um lisímetro tradicional com sensores de peso, tornando possível estimar a ET de grandes áreas de cultura.

Em [8] é apresentado o cálculo de uma estimativa de ET inteligente. Os autores empregam uma rede neuronal difusa para estimar a ET de uma plantação em estufa a partir dos dados de temperatura, humidade e pressão atmosférica. Após o treino da rede neuronal e calibração do sistema com um lisímetro de pesagem local, a ET é estimada pela rede neuronal.

Quando é necessário uma estimativa precisa da ET, deve-se recorrer a um lisímetro de pesagem. Normalmente este é constituído por dois vasos: um contendo o solo com a plantação e um segundo vaso que recolhe a água drenada a partir do primeiro vaso. Ao medir e comparar o peso do vaso com o solo com o peso do vaso de drenagem, é feita uma

estimativa da ET com precisão. Os solos possuem taxas de infiltração de água variáveis consoante o seu tipo. No início, a taxa de infiltração é alta, mas à medida que o solo absorve a água, esta taxa torna-se estável, e o uso de um lisímetro de pesagem permite abordar ambas as situações com maior precisão. A utilização de vários sensores de humidade do solo a diferentes profundidades permite a caracterização do perfil do movimento da água em termos de profundidade [9]. No artigo [9], um balanço preciso da infiltração de água é apresentado recorrendo a um lisímetro de pesagem de precisão. Da literatura conclui-se que os lisímetros de pesagem apresentam uma melhor estimativa para a ET em termos de precisão, comparado com as técnicas alternativas.

2.3 Aquisição de Imagem

Um levantamento do estado-da-arte, muito amplo, relativamente à utilização do processamento de imagem na área da agricultura é apresentado em [10]. Onde não é abordado apenas uma área especifica, mas uma abordagem completa e detalhada das possibilidades do uso do processamento de imagem na agricultura, onde podemos destacar:

1. Deteção de plantas e frutos;
2. Apoio às colheitas, incluindo classificação de frutos, deteção de maturação, contagem de frutas e previsão de rendimento;
3. Proteção da saúde de plantas e frutos e deteção de doenças, incluindo ervas daninhas, insectos, deteção de deficiência;
4. Tipos de câmaras usadas no processamento de imagem na agricultura;
5. Sistema de orientação por visão por computador para veículos agrícolas;
6. Robôs agrícolas autónomos usando visão por computador.

No nosso estudo vamos apenas focar o ponto (3) Protecção da saúde de plantas e frutas e deteção de doenças, incluindo ervas daninhas, insectos, deteção de deficiência.

Deteção de ervas daninhas Na deteção de ervas daninhas foram referenciadas diversas abordagens, nomeadamente:

- *Speed-ed Up Robust Features* (SURF) [11] + *Gaussian Mixture Modelling* (GMM) [12];
- *Make Intelligent Applications* (MIA) [13];

- *Principal Component Analysis* (PCA) [14] + *Artificial Neural Network* (ANN) [15];
- emphSupport Vectors Machines (SVM) [16];
- DL com uma CNN.

Todos os métodos anteriores apresentam índices de precisão superiores a 60%, destacando-se a precisão da CNN nos campos de cenouras com índices de precisão superiores a 99%.

Deteção de insectos Na deteção de insetos foram referenciadas apenas duas culturas (uvas e morangos). Nas uvas foi possível detectar a *"Lobesia botrana"* e o bicho da videira com índices de precisão de 95.10% e 86.00% respectivamente. Nos morangos foi possível detectar uma praga de insetos com um erro quadrático médio de MSE=0,471.

Deteção de doenças e deficiências Na deteção de doenças e deficiências nas plantas foram referenciadas doze abordagens relativamente às culturas de café, trigo, soja, batata, citrinos, pepino, maça e estufas com verduras. Recorrendo a métodos de processamento de imagem mais recentes, o pior índice de precisão obtido (64.90%) é referente à deteção de deficiência de nutrientes nas folhas de café.

Conclui-se que a deteção de doenças e deficiências não é uma tarefa fácil, a precisão dos métodos é limitada, com valores inferiores a 90%. A maior dificuldade reside na necessidade de descriminar a zona doente da saudável, e onde, por vezes a sobreposição das folhas dificulta a tarefa.

A publicação [10] é um excelente guia para futuros trabalhos relacionados com processamento de imagem na agricultura, ele aborda todas a áreas possíveis de intervenção. Relativamente aos estudos referentes à utilização de métodos de ML e DL, são apresentadas tabelas com os resultados obtidos. Com base nos resultados é possível saber qual a melhor opção a tomar no caso da implementação de um sistema idêntico, e com os mesmos objetivos dos apresentados.

Na referência [17] é feito um levantamento das técnicas atualmente usadas na área de processamento de imagem, verificando quais são as técnicas mais adequadas a usar na deteção de doenças em folhas ou frutos de plantas, assim como o modelo usado para classificar doenças. O estudo tem como objectivo mostrar as várias etapas necessárias ao processamento da imagem, nomeadamente as vantagens e desvantagens de cada técnica usada, visando mostrar o actual estado da arte no campo do processamento de imagens para a deteção e classificação de doenças.

De acordo com este estudo, o processamento de imagens segue dois tipos de abordagens: primeiro usando uma rede neural convolucional CNN [18], e o segundo usando métodos convencionais.

CNN - *Convolutional Neural Network*

Para classificar as doenças nas plantas são geralmente usadas duas abordagens diferentes:

- Aprendizagem a partir do zero (0);
- Transferência de aprendizagem.

A aprendizagem a partir do zero envia as imagens para uma série de camadas *"Convolution"* e de *"Pooling"* sucessivamente até à camada *"Full connected""*, esta última tem o resultado da classificação.

A transferência de aprendizagem usa o conjunto de dados obtidos da aprendizagem a partir do zero. O uso desta abordagem reduz o tempo de classificação e melhora o desempenho do sistema.

As principais vantagens reportadas de uma CNN, comparado com métodos convencionais, são:

- Aprende directamente com as imagens introduzidas;
- Não necessita fazer ajustes;
- O sistema pode ser usado para detectar várias doenças;
- Permite detectar doenças em diversas culturas.

A Figura 2.1 mostra os passos durante o processamento de uma imagem usando uma CNN.

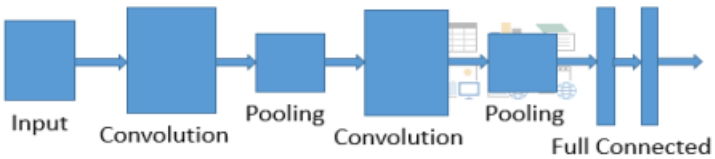


Figura 2.1: Passos usados numa CNN [17].

Devido às suas características esta abordagem é a mais usada pela maioria dos investigadores actualmente.

Métodos convencionais

O método convencional usa uma sequência de seis(6) passos para obter as características das imagens das plantas:

1. Aquisição de imagem
2. Processamento de imagem (Redução de ruído, Redimensionamento, Remoção de partes indesejada, Suavização, Ajuste do contraste e brilho);
3. Segmentação de imagem (Divisão da imagem em várias partes com as mesmas características: OTSU [19], K-means [20], conversão de *Red Green Blue* (RGB) [21] para HIV, e conversão de RGB para *Hue, Saturation, Value* (HSV) [22]);
4. Extracção de características (Cores, Formas, Contornos, Texturas, Técnicas: *Global Color Histogram* (GCH) [23], *Local Binary Pattern* (LBP) [24], *Color Coherence Vectors* (CCV) [25], *Spatial Gray Level Dependence Matrices* (SGLDM) [26])
5. Classificação (Análise de características - ANN, SVM, *Naive Bayes Algorithm* (NB) [27], *Decision Tree* (DT) [28], *Random Forest* (RF) [29]);
6. Teste (Teste com uma amostra, Verificação de precisão).

A Figura 2.2 apresenta os diversos passos usados nos métodos convencionais de processamento de imagem.

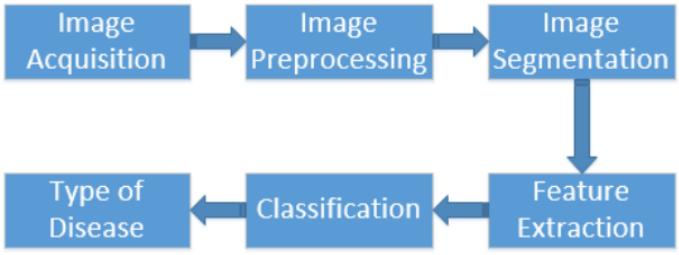


Figura 2.2: Passos usados no método convencional [17].

Conclui-se que a utilização de técnicas de processamento de imagem, para detectar e classificar doenças em frutos e folhas de plantas, são uma boa solução para os agricultores. É possível criar sistemas que fazem a deteção e classificação das doenças em tempo real e podem enviar mensagens de alerta aos agricultores, e desta forma evitar perdas de rendimento da colheita causadas pela propagação de doenças. O método convencional tem um bom desempenho para lidar com um determinado tipo de planta ou fruto, mas precisa de

muitos ajustes até obter uma boa precisão. Por outro lado a rede neuronal convolucional com uma aprendizagem a partir do zero, apresenta uma melhor classificação mas necessita de um maior numero de amostras, sobretudo quando há muitos dados.

Em [30] é usado o processamento de imagem para detectar automaticamente doenças nas plantações de milho. A partir da imagem original captada pela câmara, no formato RGB [21], foram convertidas noutros formatos, nomeadamente:

- *Scale-Invariant Feature Transform* (SIFT) [31];
- SURF;
- *Histograms of Oriented Gradients* (HOG) [32];
- *Oriented FAST and Rotated BRIEF* (ORB) [33].

Desta forma foi possível comparar o desempenho da combinação dos diversos formatos de imagem com diferentes métodos de ML, nomeadamente: SVM; DT; RF; e NB.

Concluíram que o formato RGB com o método SVM obtém o melhor desempenho (Figura 2.3), e que o método RF pode ser melhorado com um número de amostras maior (Figura 2.4).

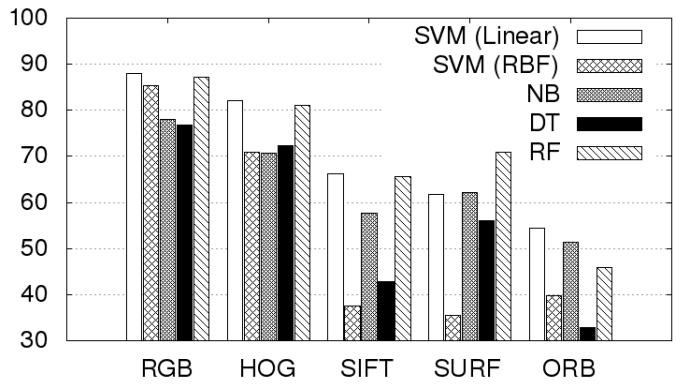


Figura 2.3: Comparações de desempenho dos métodos ML avaliados [30].

Em [34] é descrito um sistema para identificar plantas através do processamento de imagem das nervuras foliares. Á semelhança de [30], foi usado os mesmos métodos de ML. O sistema conseguiu reconhecer plantas com um a taxa de verdadeiros positivos (*True Positive Rate)* (TPR)) de 84,29% e uma taxa de falsos negativos (*False Negative Rate* (FNR)) de 15,71% [35]. É de referir a importância das condições de luz, assim como a distância da câmara à folha, para fazer uma boa aprendizagem.

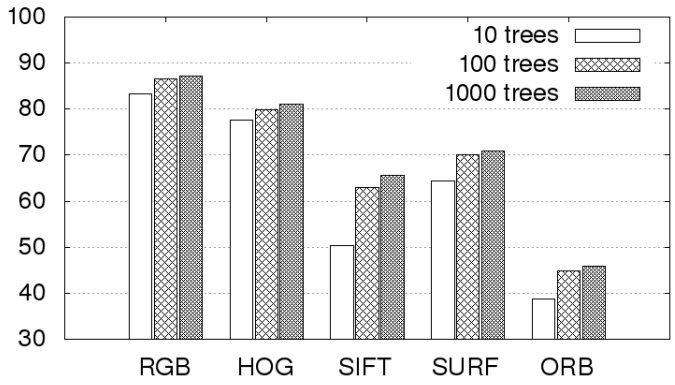


Figura 2.4: Comparações de desempenho de RF com diversos n.º de árvores [30].

Concluíram que para obter 70% de precisão, esta depende do pré-processamento da imagem, e por sua vez este pré-processamento depende do hardware e software utilizado.

Embora este sistema não se destine a detectar doenças, ao ser implementado num sistema que faça a deteção de doenças em plantas, permite verificar em tempo real se estamos a analisar a planta correcta e não alguma planta invasora que poderá surgir na plantação.

Em [36] é analisado um sistema para detectar doenças nas plantações de mandioca, porque o rendimento dessas culturas é muito afectado por quatro doenças bem conhecidas:

- *Cassava Mosaic Disease* (CMD);
- *Cassava Brown Streak Disease* (CBSD);
- *Cassava Bacterial Blight* (CBB);
- *Cassava green mite* (CGM).

Das quatro doenças a CMD e CBSD aquelas que mais degradam o rendimento dos agricultores na África Oriental e Central.

O aparecimento de doenças virais manifesta-se principalmente na deformação da cor e forma da folhas. Para extrair essas características da imagem foi usado os métodos HOG, SIFT, e SURF, tendo sido obtidos os melhores resultados com os métodos HOG e SIFT.

Para fazer a classificação das doenças foram usados os seguintes classificadores:

- Linear *Support Vector Classifier* (SVC) [37];
- *K-Nearest Neighbour* (KNN) [38];

- *Extremely Randomized Trees* (ERT) [39].

Os testes de desempenho, mostraram que a precisão com método ORB obteve a melhor precisão, superior a 99% (Figura 2.5).

	LinearSVC	ExtraTrees	*k*-NN
Color	80	48.94	44.68
ORB	99.98	99.88	100

Figura 2.5: Tabela comparativa de métodos e classificadores [36].

A Figura 2.6 apresenta a comparação dos histogramas das imagens de uma planta saudável e de uma planta doente. A Figura 2.7 mostra os pontos de intercepção através do método ORB.

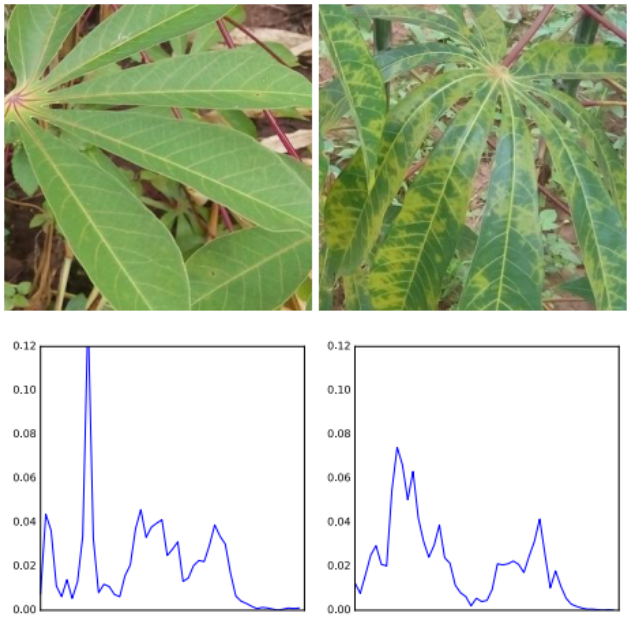


Figura 2.6: Exemplo do histograma da imagem - Planta saudável vs Planta doente[36].

O sistema apresentado foi desenvolvido para fazer um diagnóstico das doenças das plantações de mandioca baseado numa aplicação para *Smartphone*, de modo a que os agricultores possam fazer uma análise rápida das suas plantações em locais remotos. Contudo, este sistema pode ser implementado com um microcomputador do tipo Raspberry Pi[40] munido de uma câmara de 8 MPx, com processamento de imagem local.

Figura 2.7: Pontos de intercepção com ORB[36].

Em [41] foi usada uma rede neuronal convolucional profunda (DCNN) [42] baseada na arquitetura VGGnet16 [43] para detectar automaticamente doenças em folhas de plantas, os modelos foram testados num PC e num Raspbery Pi[40].

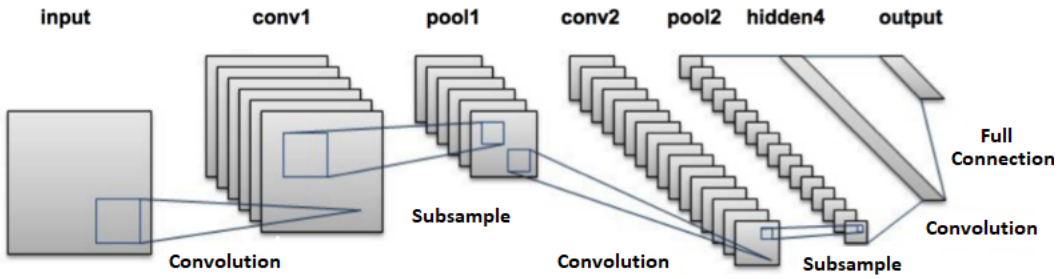


Figura 2.8: Modelo da arquitetura de uma DCNN [41].

Para obter o melhor desempenho de uma DCNN é fundamental usar um grande conjunto de dados. Este estudo recorreu a diferentes fontes de dados, nomeadamente, Internet, *dataset's* de acesso livre e imagens de telemóvel. Foram redimensionadas as resoluções das imagens para 100×100 pixel, de modo a reduzir o tempo de aprendizagem.

Foram usados os seguintes optimizadores:

- *Stochastic Gradient Descent* (SGD) [44];
- AdaGrad [45];
- *Root Mean Square Propagation* (RMSP) [46];

- Adadelta [47];

Os testes de validação no PC obtiveram uma precisão de 60% a 90%, com tempos de execução de 0,27s a 0,30s. No Raspberry Pi a precisão variou entre os 70% a 90% com tempos de execução 0,774s a 0,941s.

Deste modo conclui-se que o uso do modelo DCNN na agricultura é uma inovação que melhora o desempenho do sistemas, possibilitando uma melhoria na qualidade dos alimentos e, uma redução dos factores risco que afectam as colheitas. A parte mais critica do modelo DCNN é o processo de aprendizagem e o ajuste do optimizador.

Conclui-se que o modelo funciona bem nos diferentes hardwares, com precisões idênticas, apenas com tempos de execução diferentes, devido à capacidade computacional de cada um.

Em [48] é feito um estudo sobre a deteção e reconhecimento precoce da doença da ferrugem nas culturas de trigo. Para a deteção da doença foi usado o algoritmo *Fuzzy C-means Clustering* (FCM) [49], o reconhecimento foi feito com base numa rede neuronal artificial ANN [15], em 85% dos casos a identificação da doença foi bem sucedida.

O trigo pode ser infectado por 4 tipo de doenças consideradas ferrugem:

1. Oídio;
2. Mancha de fungos *Septoria* na folha;
3. Mancha bronzeada ou folha amarela;
4. Bolor de neve.

O processo de deteção de doença é feito em três fases:

1. Análise de imagens;
2. Extrator de características;
3. Classificador

Numa primeira fase é necessário fazer uma aprendizagem (*OFFLINE PHASE*). Com base num conjunto de imagens de folhas de trigo doentes e normais, através do **"Analisador de imagens"** algumas das características são obtidos pelo **"Extractor"**. Depois o **"Classificador"** é treinado, informando quais são as folhas normais e quais as doentes, assim como o nível de infecção.

Na segunda fase (*ONLINE PHASE*) pode-se identificar uma doença através das características obtidas pelo **"Extractor"** e testadas pelo **"Classificador"** se a folha é normal ou doente, de acordo com as informações guardadas na fase de aprendizagem.

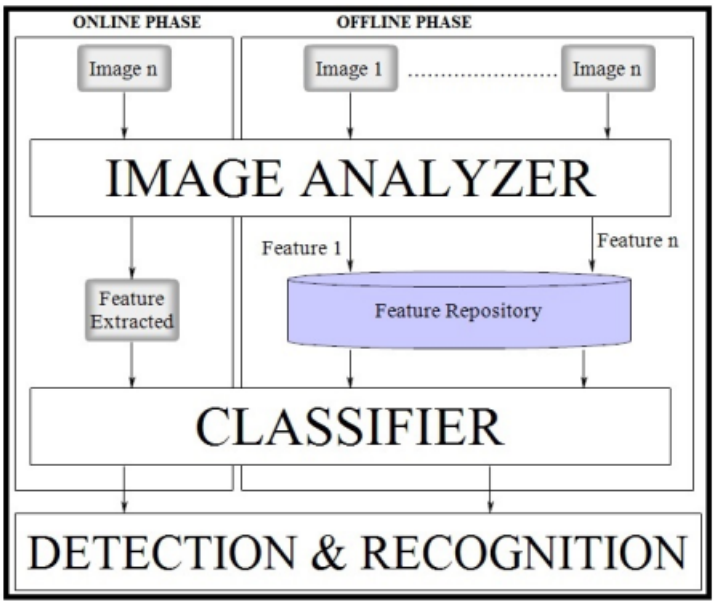


Figura 2.9: Arquitectura do sistema com ANN [48].

No estudo descrito foram recolhidas mais de 300 imagens de folhas de trigo numa plantação, depois as mesmas foram classificadas de acordo com o nível de infecção. De seguida foi treinada a rede neural artificial (ANN) com 20 imagens, quatro livres de doenças, as restantes com doença com diferentes níveis de infecção. Depois da fase de treino concluída, foram introduzidas 342 imagens na entrada da rede das quais 290 foram reconhecidas, tendo uma precisão de 84.8%.

Concluiu-se que o objectivo proposto de criar um sistema capaz de identificar as doenças nas folhas de trigo foi conseguido.

Assim, podemos concluir que, à semelhança das CNN, as ANN também produzem resultados bastantes bons na deteção de doenças em plantas, como é o exemplo do trigo.

Em [50] é proposto um sistema para identificar plantas através do processamento de imagem, designado por "WTPlant- (*What's That Plant?*), contudo recorreu ao DL pelo facto de ter demonstrado ter excelentes resultados com processamento de imagem naturais. Através de uma rede neural convolucional (CNN) é feito a aprendizagem por intermédio do método *MIT Scene Parsing*. Os primeiros resultados experimentais permitiram detetar a presença de plantas em 99.3% das imagens (17000 imagens naturais). Depois de um

processo de segmentação e aprendizagem das CNNs foi obtido uma precisão de 61.87% com a presença de plantas e flores misturadas.

Em [51] é proposto um sistema muito idêntico ao descrito em [50], baseado em ML. No entanto, faz a aprendizagem da sua rede neuronal com quatro *dataset's* diferentes:

1. Folio [52];
2. Swedish [53];
3. Flavia [54];
4. Leaf12 (tempo real).

A aprendizagem foi realizada recorrendo ao método convencional de DL, usando o conjunto de dados referidos.

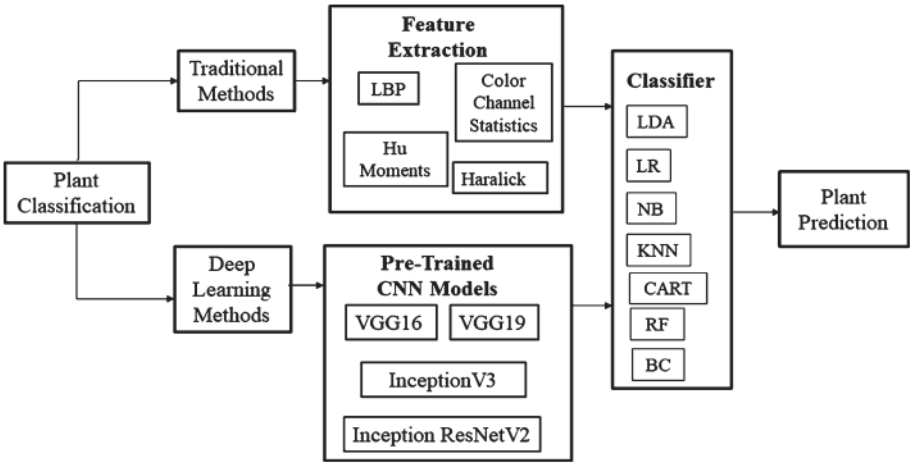


Figura 2.10: Métodos usados no estudo de [51].

Para o método convencional, a implementação foi feita em linguagem Python [1] com ajuda do pacote OpenCV [2]. A rede neuronal artificial foi implementada com ajuda da biblioteca Keras [3] juntamente com o pacote Theano [4] no *Backend*.

Tendo-se concluido que relativamente aos dois métodos utilizados, a partir dos testes experimentais, o método DL tem uma precisão mais alta em comparação com os métodos

[1] Python - Linguagem de programação de alto nível [55]
[2] OpenCV - Pacote de livrarias para processamento de imagem *Open Source* [56]
[3] Keras - Biblioteca *Deep Learning* para Python [57]
[4] Theano - Biblioteca Python para operações matemáticas multi-dimensionais [58]

convencionais em todos os conjuntos de dados. Em todos os casos, as previsões podem ser melhoradas através dos aumento da quantidade e representatividade dos dados.

Embora o sistema não se destine a detectar nenhuma doença, à semelhança do sistema implementado em [34], caso se trate de um sistema implementado com abordagem DL permite verificar em tempo real se estamos a analisar a planta correcta e não alguma planta invasora.

2.4 Análise de Colheitas

Um estudo muito completo sobre trabalhos já realizados relativamente à monitorização de culturas usando a IoT é apresentado em [59]. Além de uma introdução sobre a IoT, são analisados cerca de 38 artigos relacionados com o tema. Na maior parte dos trabalhos foram usadas redes de sensores sem fios *Wireless Sensor Network* (WSN) [60], nomeadamente com comunicações *ZigBee* (ZB) [61]. Estas redes de sensores sem fios permitiram monitorizar temperaturas, humidades do solo e do ar entre outros parâmetros, e controlar a rega das culturas. Noutros trabalhos foram usadas outras tecnologias, nomeadamente a *Radio Frequency Identification* (RFID) [62], o *General Packet Radio Services* (GPRS) [63], e o Wi-Fi [64].

É referenciado um artigo onde é usado uma câmara para deteção de pragas de insectos [65], embora a arquitectura seja explanada, não é dada informação sobre as técnicas de processamento de imagem usadas.

Em [66] é proposto um sistema com uma câmara de baixo custo, como alternativa na observação do crescimento das culturas. Usa-se uma câmara para avaliar os índices vegetativos (ev-NDVI [67], ev-SR, e ev-CIgreen) das culturas. O estudo conclui que ev-VARI funcionou melhor com o milho e o ev-CIgreen para a soja. O ev-VARI também foi o melhor para estimar a biomassa das folhas verdes no milho e o NBRINIR na soja. Os indicies vegetativos baseados em câmaras têm a possibilidade de estimar diversos parâmetros biofísicos, sendo uma boa opção para uma elevada frequência de observações e em muitos locais da plantação. Em todos os casos referenciados, a monitorização de culturas usando IoT mostra ser uma forma fácil e eficiente de aumentar a produtividade das culturas. Contudo, neste estudo não é mencionado como é feito o tratamento dos dados dos diversos sensores, nomeadamente se com a utilização de algoritmos do tipo ML - *Machine Learning* [68] e/ou DL - *Deep Learnig* [69].

2.5 Conclusão

A análise das publicações técnicas e científicas sobre o assunto, permite retirar algumas conclusões.

Quando necessitamos de uma estimativa precisa da ET, devemos usar um lisímetro de pesagem. Os sistemas que usam ML para estimar a ET recorrem a lisímetros de pesagem para a sua calibração.

O conjunção de dados proveniente dos sensores de lisímetros inteligentes com processamento de imagem da câmara numa única rede neuronal convolucional irá permitir mais precisão e fiabilidade na deteção e prevenção de doenças e pragas em plantas, assim como um melhor estudo das plantas.

Nos diversos estudos apresentados a precisão dos sistemas baseados no método de redes neuronais convolucionais CNN geralmente apresentam uma maior precisão.

A evolução das redes neuronais, nomeadamente a DCNN , nos últimos cinco anos permitem a criação de sistemas mais precisos, e com menor necessidade de ajustes. Assim, prevemos que os métodos convencionais terão tendência a desaparecer no futuro.

Alguns estudos permitem concluir que é possível desenvolver sistemas para deteção de doenças e pragas em plantas, recorrendo a hardware de baixo custo.

É muito difícil criar um sistema genérico para deteção de doenças em plantas independente do seu tipo. Para se obter bons níveis de precisão e rapidez devemos ajustar os sistemas para cada planta e doença especifica.

Capítulo 3

Arquitetura do Lisímetro

Para entender melhor a arquitetura proposta começamos por apresentar a estrutura física do lisímetro, que está ilustrada na Figura 3.1. O lisímetro é constituído por três recipientes, ou vasos, distintos. O vaso superior contém o solo com as plantas e é dotado de diversos sensores: temperatura e humidade do solo, em três profundidades distintas, assim como um conjunto de células de carga para medir o seu peso. O recipiente intermédio, que vai conter a água drenada do vaso superior, é dotado de uma célula de carga para quantificar o peso da água drenada, e uma válvula para drenar a água de em intervalos de tempo pré-definido para o vaso inferior após a avaliação do seu peso. O vaso inferior serve para reter a água drenada para eventuais análises químicas e físicas adicionais, como por exemplo para o estudo da percolação. A relação dos pesos dos vasos superior e intermédio permitem o cálculo da ET.

3.1 Estrutura Geral do Lisímetro

A arquitetura do lisímetro é projetada com recurso a tecnologias associadas à Internet das Coisas (IoT) atualmente disponíveis, nomeadamente: sensores de baixo custo, micro-controladores de baixo consumo de energia e comunicações sem fios. Um dos objetivos do projeto consiste em construir um sistema de baixo custo e elevado desempenho seguindo o paradigma IoT.

A Figura 3.2 apresenta um diagrama em camadas da arquitetura do sistema, composto por três partes:

- **Camada física** - responsável pela aquisição e processamento inicial dos dados provenientes dos sensores e posterior envio para a camada nuvem;

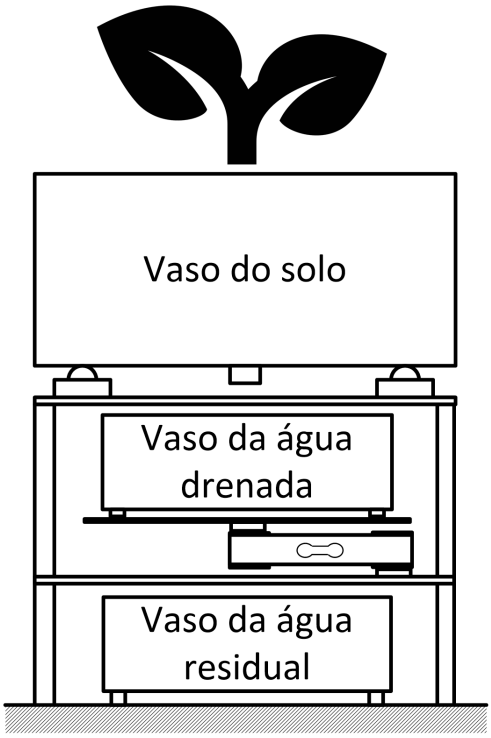


Figura 3.1: Estrutura física do lisímetro [2].

- **Camada nuvem** - responsável pela receção, o armazenamento e a análise dos dados, gerando avisos relacionados com a cultura e com o controle de pragas;
- **Camada aplicação** - disponibiliza as aplicações para os diferentes utilizadores de acordo com suas funções.

3.1.1 Camada Física

Na parte inferior da **camada física** estão os sensores que medem os parâmetros do solo e do ambiente, assim como a câmara que captura imagens da planta. No solo da cultura é medida a temperatura e a humidade em três profundidades diferentes tal como o peso do vaso. Toda a água drenada do solo da cultura que fica retida no segundo vaso é pesada, e depois é trespassada para um terceiro vaso através de uma válvula. A pesagem periódica do solo e da água drenada permitem o cálculo da ET. A temperatura, a humidade e a luminosidade ambiente são medidas para obter informação do meio onde está a cultura. Os sensores estão ligados a um MCU [6], de baixo consumo de energia, que recebe os dados, faz um pré-processamento dos mesmos, formatando-os para serem enviados para a **camada**

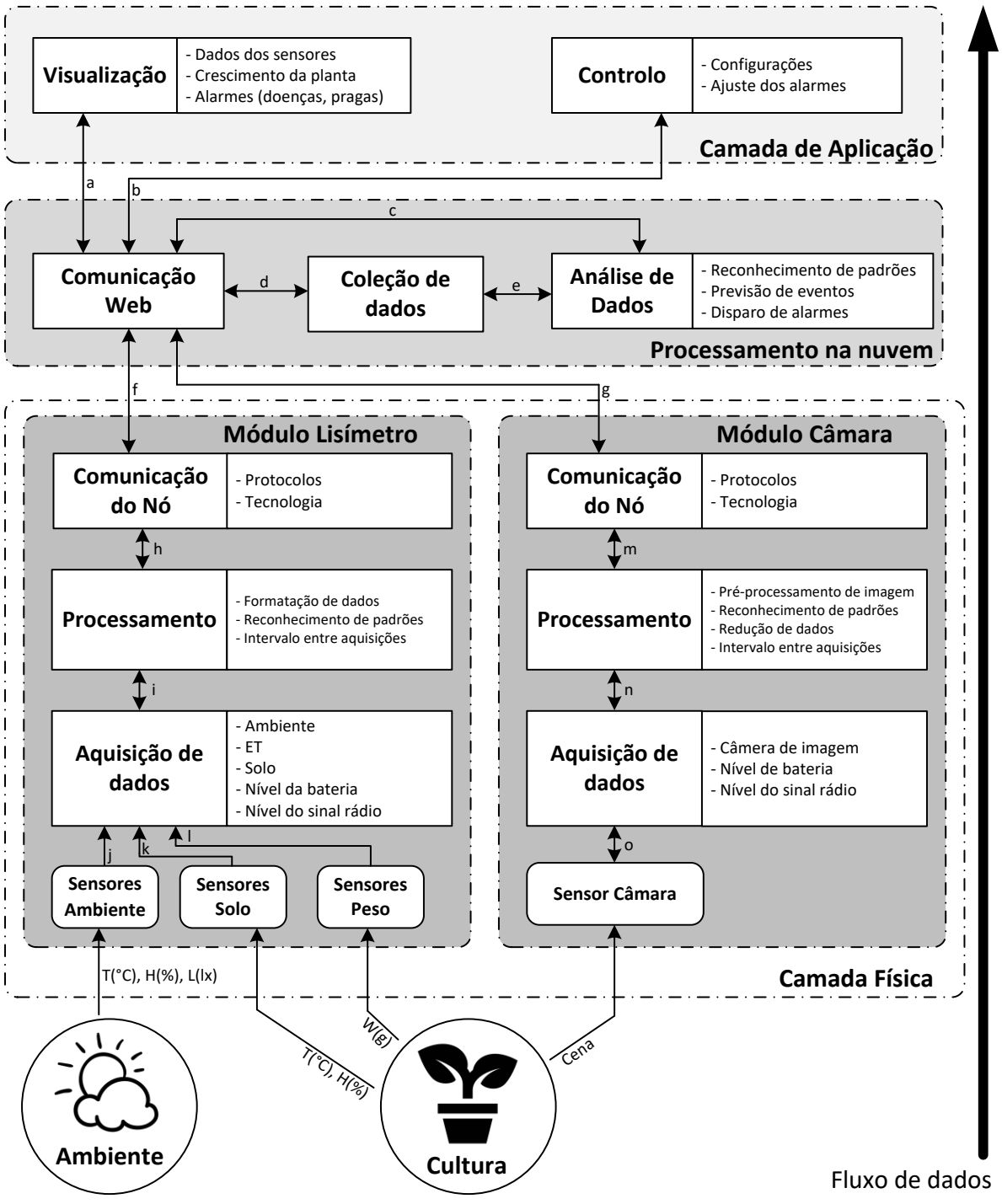


Figura 3.2: Arquitetura do sistema de um lisímetro com aquisição de imagens [2].

nuvem através de um *System on Chip* (SoC) [70] com comunicações sem fios integrado. Os dados são enviados para a **camada nuvem** em intervalos de tempo pré-definidos que podem ser programados.

Após o envio dos dados o MCU entra em modo de poupança de energia (*Low Power Mode* (LPM) [71]). Todos os sub-circuitos (SoC, Sensores, Servo-motor) são desligados e neste modo o consumo de energia é muito baixo, o que permite aumentar significativamente a autonomia da bateria. Em LPM o *timer* do MCU continua a funcionar, e decorrido o intervalo de tempo pré-definido para nova leitura dos sensores, o *timer* vai ativar todo o sistema e um novo ciclo de leituras é iniciado. Juntamente com os dados dos sensores é enviado ainda: a tensão da bateria, para avaliar o seu estado de carga; o nível de sinal rádio das telecomunicações, para obter a qualidade de ligação de dados; um *timestamp*, com a informação de data e hora da leitura dos dados, que é obtido através de servidor de tempo na Internet (*Network Time Protocol* (NTP) [72]).

A **camada física** usa uma câmara de alta resolução para a análise da saúde e crescimento da planta e controlo de possíveis pragas. As imagens adquiridas são processadas localmente, para redução de dados, e são posteriormente enviadas para a **camada nuvem**. A câmara está ligada a um micro computador (SBC [73]), que faz o pré-processamento das imagens e envia os dados através dos seus recursos de comunicações para a **camada nuvem**.

Não sendo necessário capturar imagens com a mesma frequência dos restantes sensores, dado que o crescimento das plantas é lento, é suficiente a aquisição de uma a três imagens por dia. Um SBC consome em média cerca de $5Wh$ e se estiver constantemente ligado há um consumo de energia diário muito elevado para um sistema IoT, que apenas trabalha alguns minutos por dia. O problema é solucionado com o uso de um MCU com LPM, que desliga todos os sub-circuitos (a câmara e o SBC) quando não está a ser feita a aquisição, processamento ou envio dos dados. O *timer* do MCU é programado para alimentar os sub-circuitos de acordo com o horário escolhido para aquisição das imagens.

Juntamente com as imagens também são enviados para a **camada nuvem** outros dados importantes para saber o estado do sistema, nomeadamente: a tensão da bateria, o nível de sinal rádio, e um *timestamp*, à semelhança do que acontece no módulo lisímetro.

3.1.2 Camada na Nuvem

No nível intermédio situa-se a **camada nuvem**, que é responsável pela receção e processamento dos dados provenientes dos diversos módulos: módulos lisímetros e módulos câmaras da camada física. A camada nuvem disponibiliza os seguintes serviços:

a) Protocolo de comunicações - estabelece a comunicação dos dispositivos com a camada nuvem.

b) Sistema de gestão de base de dados - armazenar todos os dados provenientes dos dispositivos;

c) Serviços de processamento de dados - reconhecimento de padrões através de *Machine Learning* ML, para a predição de eventos e geração de alarmes;

d) Servidor *WEB* [74] - disponibiliza à **camada aplicação** o acesso aos dados, configuração e gestão do sistema.

Como já referido, os dispositivos da camada física devem ser eficientes do ponto de vista energético, sendo que uma parte da energia é despendida para a transmissão de dados. A transmissão de dados sem fios aumenta o consumo de energia comparativamente a outros tipos de comunicações. Uma forma de reduzir o consumo de energia, será reduzir o tempo necessário para a transmissão dos dados, e para isso devem ser escolhidos os protocolos mais adequados. Para reduzir energia, os dados dos sensores de um dispositivo devem ser enviados de uma só vez, usando um protocolo de comunicações leve, tal como o *Constrained Application Protocol* (CoAP) [75] ou o *Message Queuing Telemetry Transport* (MQTT) [76].

Após a receção dos dados, estes são guardados numa base de dados para posterior consulta e análise. Os dispositivos IoT podem gerar um enorme volume de dados. O módulo lisímetro está programado para obter dados dos sensores de 10 em 10 minutos, o que corresponde a 6 leituras por hora, perfazendo um total de 144 leituras diárias e 52560 leituras anuais. Para armazenar este grande volume de informação, é recomendado o uso de base de dados não relacional, nomeadamente do tipo *Not only SQL* (NoSQL) [77]. Este tipo de bases de dados orientadas a documentos estão preparadas para gerir um grande volume de dados, com uma grande variedade, velocidade de acesso elevada, fiabilidade e valor. Para garantir a escalabilidade é necessário definir desde o início uma base de dados com as características referidas.

Os dados guardados na base de dados, juntamente com as imagens capturadas, são processados com recurso a técnicas de *Machine Learning* ML. Este processamento permite a predição de eventos e gerar avisos sempre que se preveja que algo não esteja bem com a cultura. Através da análise dos dados dos sensores do módulo lisímetro é possível prever o aparecimento de determinadas doenças, tais como por exemplo o míldio e o oídio. A análise das imagens provenientes do módulo câmara permitem avaliar o crescimento saudável da planta e ainda detetar doenças e possíveis pragas, gerando avisos para o agricultor.

O servidor WEB disponibiliza um serviço de acesso à informação dos lisímetros de uma forma transversal. O acesso pode ser feito das mais diversas plataformas independentemente do seu sistema operativo. A configuração e gestão do sistema também é feita através desta plataforma WEB, nomeadamente: gestão de utilizadores, parametrização do reconhecimento de padrões, configuração de alarmes e avisos, etc..

3.1.3 Camada de Aplicação

No topo do diagrama temos a **camada aplicação** que fornece diversos serviços aos utilizadores finais, tais como a configuração do sistema, a visualização dos dados, os resultados das análises de dados, configuração e exibição de alarmes, etc. Um dado utilizador acede aos dados em função das suas funções/competências, sendo criados grupos de utilizadores com as permissões bem definidas. Podendo ter nomeadamente, os seguintes grupos: Agricultores, Técnicos Agrícolas, Técnicos de hardware e Técnicos de ML.

3.2 Arquitetura do Hardware

A Figura 3.3 representa a arquitetura de hardware do sistema com base no modelo descrito atrás. Neste diagrama pode-se ver a interligação entre os diversos componentes físicos, tal como o fluxo e tipo de sinal. A Tabela 3.1 lista os acrónimos utilizados no diagrama da Figura 3.3.

Tabela 3.1: Acrónimos do hardware do sistema

	Description		Description
MCU	Micro-controlador	SBC	Micro-computador
SoC	Sistema num C.I.	PS	Fonte de alimentação
ATS	Sensor Temperatura Ambiente	AHS	Sensor Humidade Ambiente
ALS	Sensor Luz Ambiente	STS	Sensor Temperatura Solo
SMS	Sensor Humidade Solo	SWS	Sensor Peso Solo
DWS	Sensor Água Drenada	BVS	Sensor Tensão Bateria
CS	Sensor de Imagem	SM	Servo Motor válvula
Data	Dados	PW	Energia
Ctrl	Sinal de Controlo		

Com base no diagrama da Figura: 3.3 é possível escolher os componentes eletrónicos no mercado, de forma a atender às especificações pretendidas para o lisímetro.

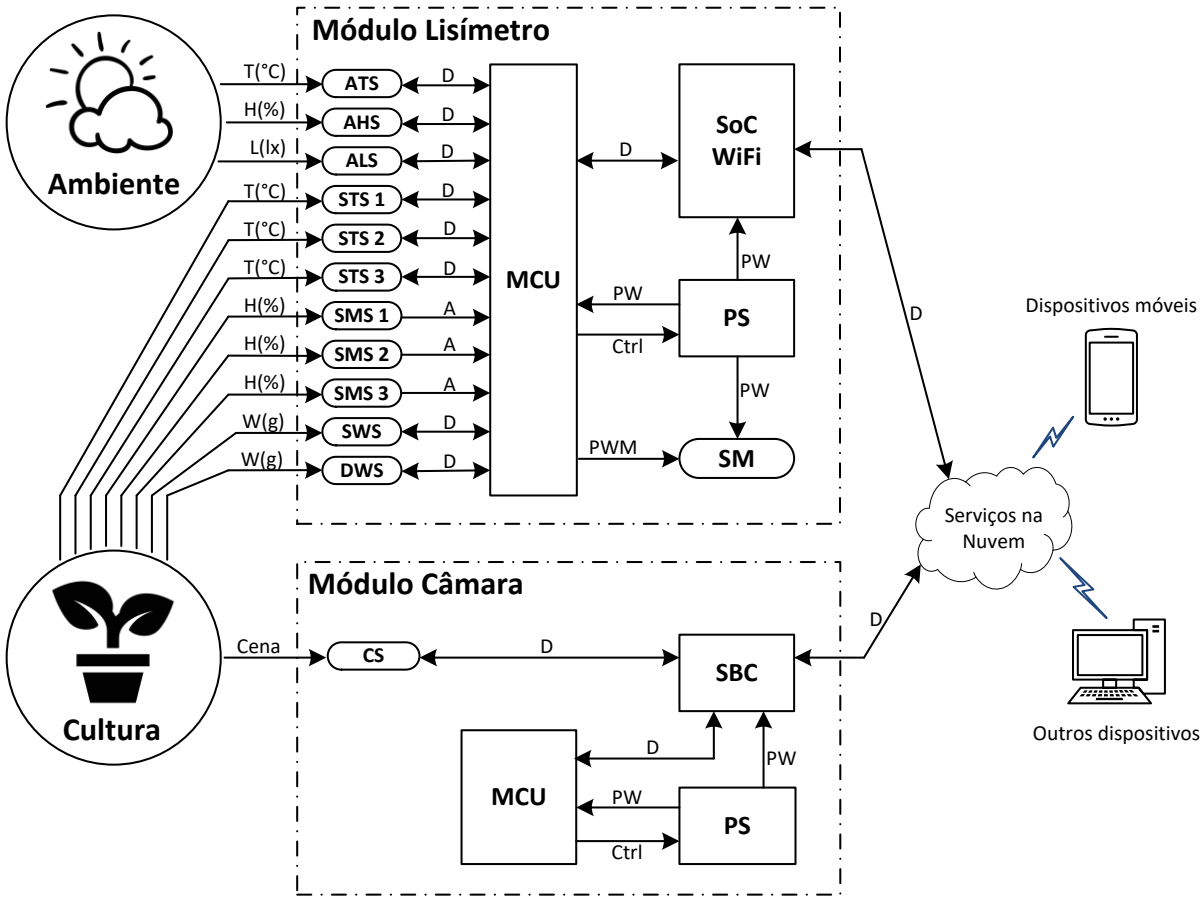


Figura 3.3: Arquitetura do hardware do sistema [2].

3.3 Conclusão

A arquitetura proposta é o ponto de partida para o desenvolvimento e construção de um protótipo funcional de um lisímetro. A abordagem IoT permite uma redução de custos, elevado desempenho, escalabilidade, eficiência energética, disponibilidade em tempo real dos dados, bem como uma análise autónoma e automática dos mesmos.

O facto dos módulos lisímetro e câmara estarem grande parte do tempo em LPM, torna-os eficientes do ponto de vista energético, e podem trabalhar em condições de temperaturas extremas.

A análise de dados através de técnicas de DL necessita de grandes quantidades de amostras para terem um bom desempenho. A centralização dos dados provenientes de diversos lisímetros, pode gerar de uma forma rápida um grande conjunto de dados, o que

permite obter uma boa precisão comparado com outros sistemas.

Para obter bons resultados é importante que os agricultores possam interagir com o sistema, registando na plataforma a sua análise da cultura sempre que sejam feitas visitas. Este procedimento permite a calibração do sistema e aumento do seu constante desempenho.

Esta arquitetura permite a monitorização permanente com o consequente aumento da produtividade das culturas, e a redução do consumo de água, porque é possível determinar apenas a quantidade necessária de água que determinada plantação precisa.

Capítulo 4

Implementação Experimental

Neste capitulo é descrita a implementação física do protótipo experimental do lisímetro. Com base nas especificações pretendidas, é feita a escolha de componentes para o protótipo.

4.1 Introdução

Para implementar a arquitetura descrita no capitulo anterior, desenvolvemos um protótipo funcional o qual é composto por 4 partes:

1. Módulo Lisímetro;
2. Módulo Câmara;
3. Módulo Computacional;
4. Estrutura de Suporte.

Os componentes que compõem o sistema foram escolhidos atendendo às especificações pretendidas para o lisímetro. A Tabela 4.1 apresenta a especificações pretendidas para o nosso protótipo.

4.2 Módulo Lisímetro

A estrutura de hardware do módulo do lisímetro é apresentada na Figura 4.1. A Tabela 4.2 apresenta a legenda dos componentes usados para implementação, incluindo as especificações mais importantes. O diagrama contem representadas as interligações entre os dispositivos.

Tabela 4.1: Especificações/requisitos pretendidas para o protótipo.

Descrição	**Especificação/Requisito**
Módulo Lisímetro	
Microcontrolador	Baixo consumo de energia
Comunicações	Wi-Fi
Sensor de Temperatura Ambiente	-10ºC:+50ºC, ±1ºC
Sensor de Humidade Ambiente	0–100% RH, ±5%
Sensor de Luz Ambiente	Luz visível e IR
Sensor de Temperatura do Solo	-10ºC:+50ºC, ±1ºC
Sensor de Humidade do Solo	0–100% RH, ±10%
Sensor de Peso do Solo	Erro < ±100gr
Sensor de Peso da Água Drenada	Erro < ±10gr
Servomotor	Torque > 2,5Kg/cm
Alimentação	Bateria + Painel solar
Módulo Câmara	
Sensor de Imagem	Resolução de imagem > 4Mpx
Computador numa Placa	CPU ARM, 2MB RAM, CSI, GPIO
Microcontrolador	Baixo consumo de energia
Alimentação	Bateria + Painel solar
Módulo Computação na Nuvem	
Servidor	Virtual e *Open Source*
Instalação de serviços	*Containers*
Protocolo de comunicação	Publicação / Subscrição
Base de dados	Não relacional
Processamento de dados	*Machine Learning*
Envio de imagens	*Open* SFTP
Servidor WEB	*Open Source*
Estrutura de Suporte	
Estrutura	Metálica, pintura anti-corrosão

4.2.1 Escolha de componentes

Para a realização do protótipo procurou-se no mercado componentes com as especificações pretendidas, com o objetivo de criar um protótipo recorrendo às mais recentes tecnologias, e de elevada precisão e com custos controlados. A seguir apresentamos a descrição e justificação para a escolha de cada componente.

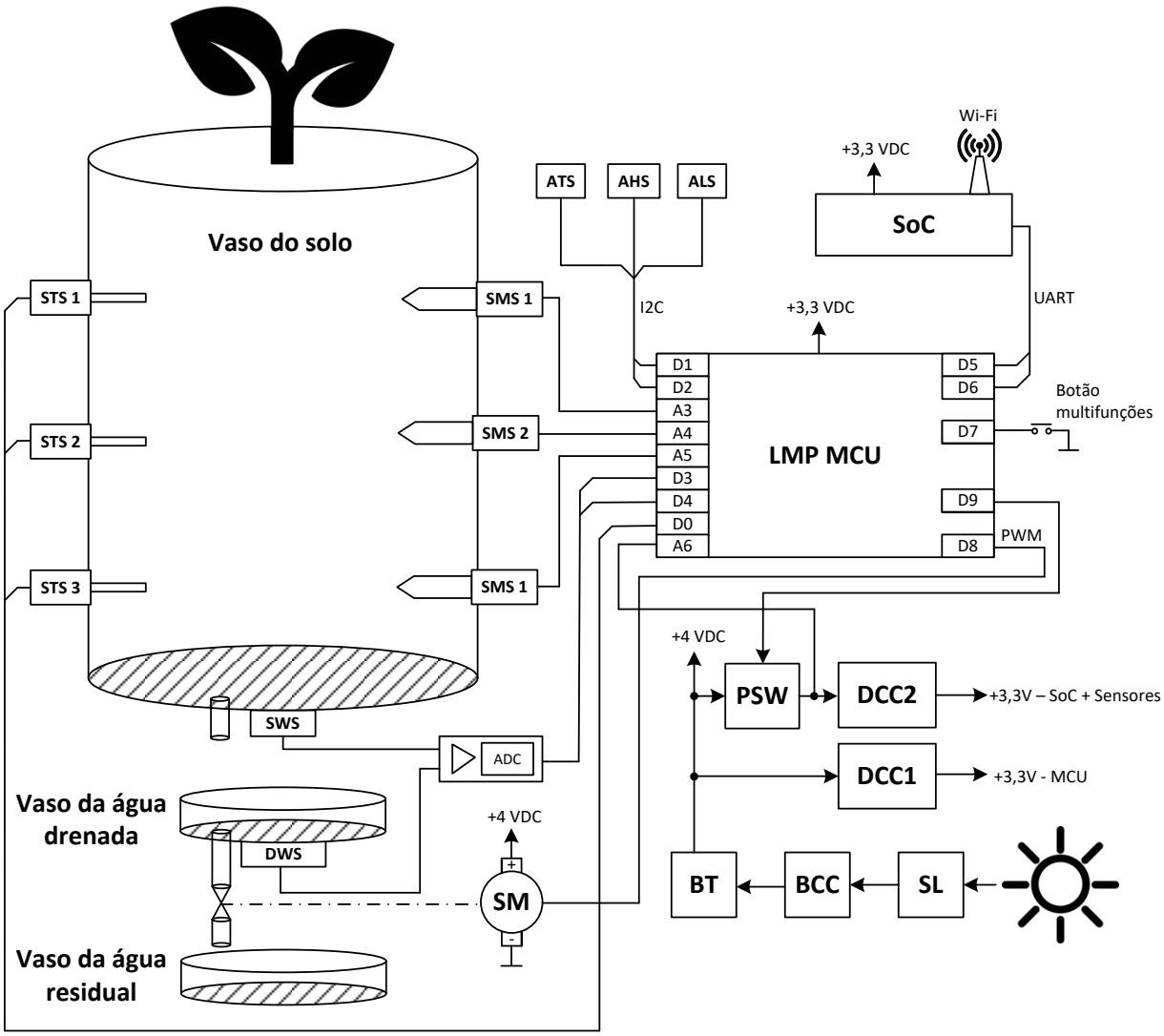


Figura 4.1: Diagrama de blocos do hardware do módulo lisímetro [2].

MCU - Microcontrolador

Para controlar o módulo foi escolhido o MCU MSP430G2553 [1] devido ao seu baixo consumo de energia e custo. Este MCU dispõe de portas analógicas e digitais necessárias para ligar os diversos sensores pretendidos. Possui um barramento (*Inter-Integrated Circuit* (I2C)) que permite interligar os sensores com essa tecnologia de comunicação, e uma interface série (*Universal Asynchronous Receiver-Transmitter* (UART)) necessária para comunicar com o SoC. No diagrama da Figura 4.1 o MCU está identificado com as portas usadas

[1] MSP430G2553 - Especificações: https://www.ti.com/product/MSP430G2553

Tabela 4.2: Componentes de hardware do módulo lisímetro.

	Descrição	**Modelo**	**Especificações**
MCU	Microcontrolador	MSP430G2553	16 MHz, 16KB/512B, LPM
SoC	Sistema num *Chip*	ESP8266-01	Wi-Fi, 3.3VDC
ATS	Sensor Temperatura Ambiente	SHT30	(I2C) -40ºC:+80ºC, ±0.3ºC
AHS	Sensor Humidade Ambiente	SHT30	(I2C) 0–100% RH, ±2.0%
ALS	Sensor Luz Ambiente	TSL2561	(I2C) Visível IR
STS	Sensor Temperatura do Solo	DS18B20	(1Wire) -10ºC:+85ºC, ±0.5ºC
SMS	Sensor Humidade do Solo	CSMS	Sensor Capacitivo Grove
SWS	Sensor Peso do Solo	HX711	4 células de 1/2 ponte
DWS	Sensor Peso da Água Drenada	HX711	1 célula ponte *Wheatstone*
BVS	Sensor Tensão da Bateria	Resistivo	Divisor tensão resistivo
SM	Servomotor	HS422	(PWM) 180º, 3.3Kg/cm
SL	Painel Solar Fotovoltaico	FAL09004	5VDC, 1W, 110×60mm
BCC	Controlador Carga da Bateria	TP4056	1A, 1×BAT Li-Ion
BT	Bateria Li-Ion	INR18650-35E	3.6V, 3500mAh, Samsung
PSW	Interruptor de potência	DMP2022LSS	-10A, P-MOSFET
DCC1	Conversor DC/DC MCU	MCP1700-3302	3.3V, 250mA, LDO
DCC2	Conversor DC/DC	MCP1700-3302	3.3V, 250mA, LDO

para interligação com os diversos tipos de sensores. A Figura 4.2 apresenta a estrutura interna do MSP430G2553, com os diversos módulos periféricos realçados.

SoC - Sistema num *Chip*

Como a rede sem fios escolhida para o protótipo foi a tecnologia *Wireless Local Area Network* (WLAN) [79], o módulo WLAN ESP-01 [2] com o SoC ESP8266 [3] foi o selecionado, devido às suas especificações técnicas. É um módulo de reduzidas dimensões, compatível com as normas de redes sem fios IEEE 802.11b/g/n, com um *Central Processing Unit* (CPU) 32bits com uma frequência de 160MHz, uma porta de comunicação UART, e um consumo médio de 71mA, suportando uma tensão de alimentação entre 3,0V e 3,6V. A Figura 4.3, mostra uma imagem do módulo ESP-01.

[2]ESP-01 - Especificações: https://docs.ai-thinker.com/_media/esp8266/docs/esp-01_product_specification_en.pdf

[3]ESP8266 - Especificações: https://www.espressif.com/en/products/socs/esp8266

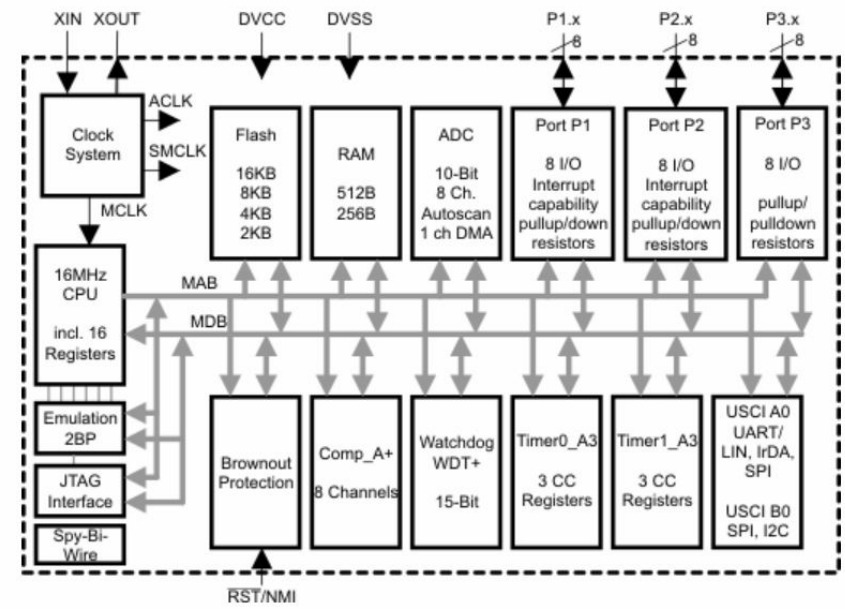


Figura 4.2: Estrutura interna do MCU MSP430G2553 [78].

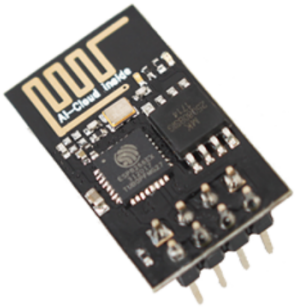


Figura 4.3: Módulo WLAN ESP-01 com SoC ESP8266 [80].

ATS/AHS - Sensor de Temperatura e Humidade Ambiente

Para medição da temperatura e humidade do ar ambiente foi usado um módulo com o sensor SHT30 [4], desenvolvido pela *Sensirion*. A versão usada recorre à interface I2C para as comunicações com o MCU. Este sensor é uma referência na indústria devido à sua elevada precisão e baixo custo. Este sensor tem uma precisão de ±0.3ºC para temperaturas de -40ºC a +80ºC, e uma uma precisão de ±2.0% para valores de humidade relativa entre 0 e 100%. A Figura 4.4 apresenta o módulo com o sensor SHT30.

[4]SHT30 - Especificações no URL: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Datasheets/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf

Figura 4.4: Módulo com sensor de temperatura e humidade ambiente SHT30 [81].

Para a sua instalação no exterior, o módulo foi instalado numa caixa plástica, a qual foi perfurada de forma permitir a circulação o ar ambiente, e ao mesmo tempo, não permitir a entrada da água da chuva.

ALS - Sensor de Luminosidade

Para medir a intensidade da luminosidade ambiente é usado um módulo com o sensor TSL2561 [5], desenvolvido pela *AMS* e usa a interface I2C. Este dispositivo possui internamente dois sensores, o primeiro mede a intensidade da luz visível com a luz infravermelha (IR) e o segundo apenas mede a intensidade da luz IR. A saída digital ligada a um MCU e usando uma função de conversão permite obter o valor da iluminação em lux idêntica à resposta do olho humano. A Figura 4.5 apresenta o módulo usado no protótipo.

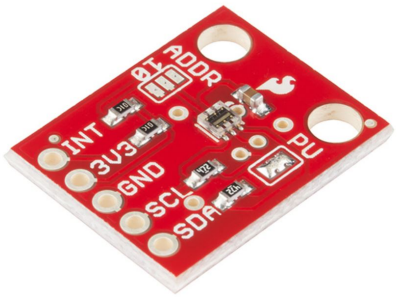


Figura 4.5: Módulo com Sensor de Luminosidade TSL2561 [82].

Este módulo está instalado na mesma caixa com o módulo sensor SHT30. Ambos os sensores usam o protocolo I2C permitindo, assim, usar a mesma cablagem para ligação ao MCU.

[5] TSL5261 - Especificações: https://ams.com/documents/20143/36005/TSL2561_DS000110_3-00.pdf/18a41097-2035-4333-c70e-bfa544c0a98b

STS - Sensor de Temperatura do Solo

O vaso do solo está equipado com um conjunto de sensores de humidade e temperatura a diferentes profundidades. A temperatura do solo é medida recorrendo a sensores DS18B20 [6], da *Maxim Integrated*, com tecnologia de comunicação *OneWire* [83], que permitem medir temperaturas entre -10ºC e +85ºC com uma precisão de ±0,5ºC. Estes 3 sensores estão ligados em paralelo a uma porta digital bi-direcional do MCU. A Figura 4.6 apresenta uma imagem do sensor de temperatura, que é selado de modo a ser à prova de água, usado na realização experimental.

Figura 4.6: Sensor de temperatura DS18B20 á prova de água [84].

A instalação destes sensores no vaso é efetuada através de bucins PG7, de modo a garantir a estanquidade do vaso.

SMS - Sensor de Humidade do Solo

A humidade do solo é medida com sensores capacitivos resistentes à corrosão, produzidos pela *Seeed Grove* [7]. O nível de humidade é medida em função da alteração da capacidade do dialétrico da sonda capacitiva do sensor. Como a sonda do sensor não tem partes metálicas expostas, é mais resistente à corrosão, e mantendo a estabilidade das leituras por muito mais tempo.

As saídas dos sensores são ligadas a portas analógicas do MCU, cujo conversor *Analog to Digital Converter* (ADC) tem uma resolução de 10 bits. Estes sensores necessitam de uma calibração prévia para ajustar a percentagem de humidade medida ao tipo de solo.

A electrónica do sensor foi encapsulado numa caixa com certificação IP65 de modo a assegurar o correto funcionamento em ambiente exterior e manter a estanquidade do vaso.

[6] DS18B20 - Especificações: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[7] CSMS - Especificações: https://files.seeedstudio.com/wiki/Grove-Capacitive_Moisture_Sensor_Corrosion_Resistant/res/soil_sensor.pdf

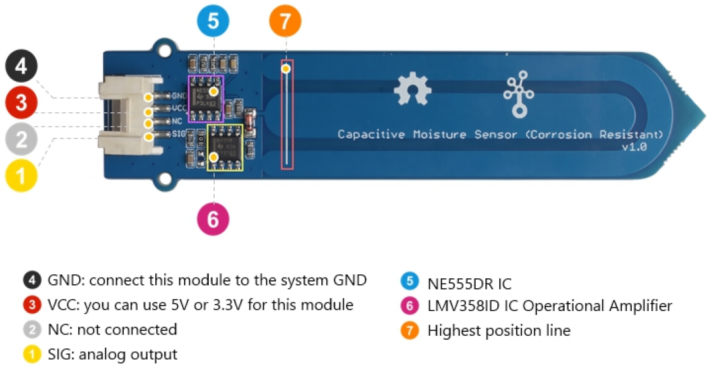


Figura 4.7: Sensor Capacitivo para Medição da Humidade do Solo [85].

SWS - Sensor de Peso do Solo

O vaso do solo é pesado recorrendo a 4 células de carga de 50Kg. Este conjunto de células permite usar vasos com solo e plantas até 200Kg de peso com um erro inferior a ±50g . A Figura 4.8 mostra a imagem de uma das células de carga usadas.

Figura 4.8: Célula de carga de 50Kg com 1/2 ponte de *Wheatstone* [86].

Cada célula dispõe de 1/2 ponte de *Wheatstone*. Recorrendo a um circuito elétrico com o esquema da Figura 4.9 é possível obter o equivalente eléctrico de uma ponte completa *Wheatstone* através de 4 células de 1/2 ponte.

Este conjunto de células está ligado a um circuito integrado (*Integrated Circuit* (IC)) que contém internamente um amplificador e um conversor analógico-digital(ADC) com a designação HX711 [8]. Este IC comunica com o MCU recorrendo a um protocolo proprietário, usando 2 portas (*Data* e *Clock*). Este IC HX711 tem 2 canais, o que permite ligar 2 células

[8]HX711 - Especificações: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

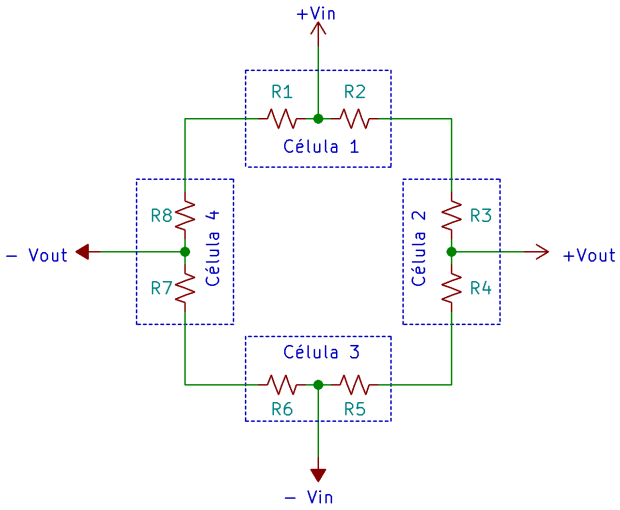


Figura 4.9: Esquema de ligação de 4 células de 1/2 ponte de *Wheatstone*.

de carga, o ADC tem uma resolução de 24bits. A Figura 4.10 apresenta o diagrama de blocos interno do HX711. A Figura 4.11 apresenta o módulo HX711 utilizado.

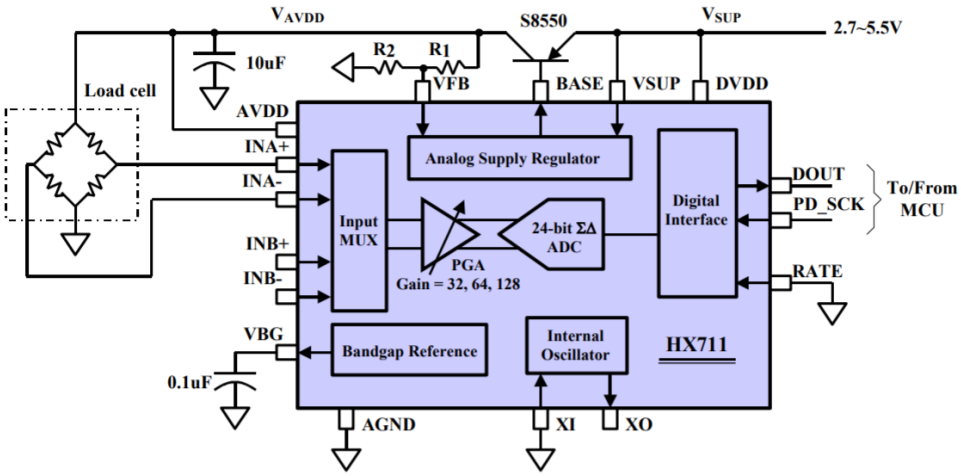


Figura 4.10: Diagrama de blocos do CI HX711 [87].

O módulo HX711 foi encapsulado numa caixa de plástico estanque com certificação IP65, para assegurar o seu correto funcionamento em ambiente exterior. A caixa com o módulo HX711 foi instalada o mais próximo das células de modo a reduzir o máximo possível interferências eletromagnéticas.

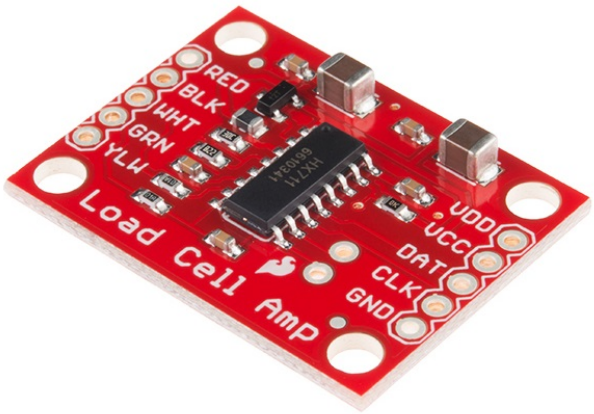


Figura 4.11: Módulo com CI HX711 [88].

DWS - Célula de Carga da Água Drenada

O vaso da água drenada é pesado com uma célula de carga de 10Kg [9] com ponte de *Wheatstone*. O erro da célula de carga é inferior a 5g. A Figura 4.12 apresenta uma imagem da célula utilizada. Esta célula está ligada a um outro canal do módulo HX711 já descrito.

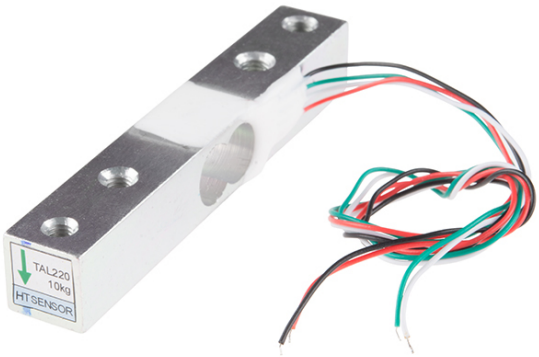


Figura 4.12: Célula de carga de 10Kg [89].

[9] Célula de carga 10Kg - Especificações: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/TAL220M4M5Update.pdf

BVS - Sensor da Tensão da Bateria

O estado da carga da bateria é proporcional à tensão dos seus terminais. Para fazer a leitura da tensão da bateria é usado o ADC interno do MCU. O ADC foi programado para usar como referência a referência interna de 2500mV, o que implica que a tensão máxima de entrada na porta analógica não pode exceder esse valor. Para obter uma amostra proporcional da tensão da bateria é usado um divisor resistivo por intermédio de duas resistências (R4 e R5). A Figura 4.13 apresenta o divisor de tensão implementado, com um condensador para estabilizar a tensão medida.

A tensão máxima da bateria totalmente carregada é de 4200mV, com um fator de divisão de 2, a tensão à entrada da porta do ADC é de 2100mV. Analisando a malha resistiva, a tensão à saída do divisor resistivo é dada por: $V_{sens} = \frac{R5}{R4+R5} \times V_{bateria}$. Recorrendo a duas resistências de $10k\Omega$, obtemos o divisor pretendido com uma corrente na malha inferior a $210\mu A$.

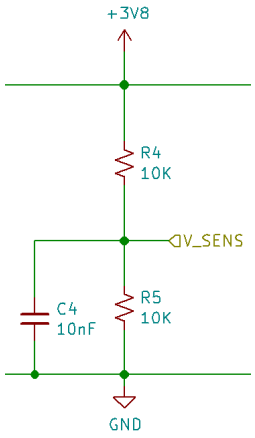


Figura 4.13: Medição da Tensão da Bateria do Lisímetro [II.1].

SM - Servomotor

Para fazer a abertura da passagem da água drenada após a sua pesagem, é usado um servomotor de radio-modelismo, nomeadamente foi escolhido o modelo HS-422 [10] da *HiTec*. Este servomotor possui um torque mínimo de 3.3Kg/cm, e uma rotação de 180º que são suficientes para acionar a válvula de descarga do vaso da água drenada. A Figura 4.14 apresenta a imagem deste servomotor.

[10]HS-422 - Especificações: https://hitecrcd.com/products/servos/analog/sport-2/hs-422/product

Figura 4.14: Servomotor HS-422 para abertura de válvula do vaso da água drenada [90].

SL - Painel Solar Fotovoltaico

Foi escolhido o sol como fonte de energia do sistema. Atendendo ao consumo de energia do nosso módulo foi escolhido um painel solar mono-cristalino de 5V, com uma potência máxima de 1W, com capacidade para fornecer a energia necessária para carregar a bateria. A Figura 4.15 apresenta a imagem do painel solar.

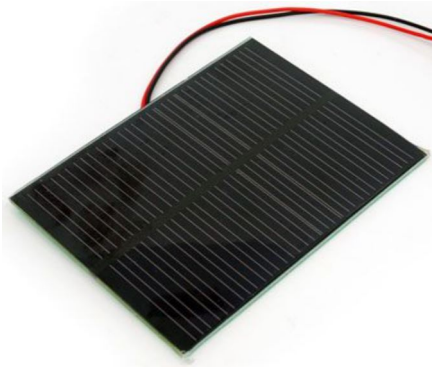


Figura 4.15: Painel solar de 5V 1W [91].

BCC - Controlador de Carga da Bateria

De modo a carregar adequadamente a bateria, é necessário um controlador de carga, tendo sido escolhido um módulo com um controlador dedicado para carregar baterias de Li-Ion de apenas uma célula, o módulo CI TP4056 [11]. Este módulo permite carregar com uma corrente máxima de 1A, com tensões de entrada de $4,5VDC$ a $5,5VDC$. A Figura 4.16 apresenta a imagem do módulo escolhido.

[11] TP4056 - Especificações: http://www.tp4056.com/d/tp4056.pdf

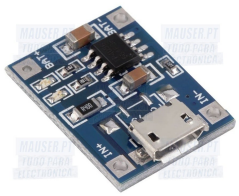


Figura 4.16: Módulo Controlador de Carga de Bateria com TP4056 [92].

BT - Bateria

Para que o sistema funcione independente das condições meteorológicas, é importante que nunca falte energia ao sistema. Para assegurar o fornecimento ininterrupto de energia é usada uma bateria de iões de lítio (Li-Ion)[93] com capacidade suficiente para manter o sistema em funcionamento mesmo com vários dias sem presença de sol. A Figura 4.17 apresenta a imagem da bateria *Samsung* INR18650-35E[12], com uma tensão nominal de $3,6V$ e uma capacidade de $3500mAh$, usada no protótipo.

Figura 4.17: Bateria de iões de lítio INR18650-35E da *Samsung SDI* [94].

PSW - Interruptor de potência

Pretende-se fazer leituras dos sensores com intervalos de 10 minutos, pelo que não é necessário que todo o sistema esteja a ser alimentado em continuo, apenas o MCU necessita de ser alimentado sem interrupções. Para se poupar a energia da bateria os sensores e o servomotor apenas são alimentados quando se pretende obter a sua ou o seu acionamento, respetivamente. Assim, é usada uma porta do MCU para controlar o interruptor de potência de forma a alimentar os sensores e o SoC apenas quando necessário. Para isso,

[12]INR18650-35E - Especificações: https://www.orbtronic.com/content/samsung-35e-datasheet-inr18650-35e.pdf

foi escolhido o transístor DMP2022LS [13] que é usado como interruptor. É um transístor MOSFET do tipo P, que permite controlar uma corrente até $-8A$ @ $+75^{o}$C, e possuiu uma baixa resistência de $25m\Omega$ @ $V_{GS} = 2,5V$.

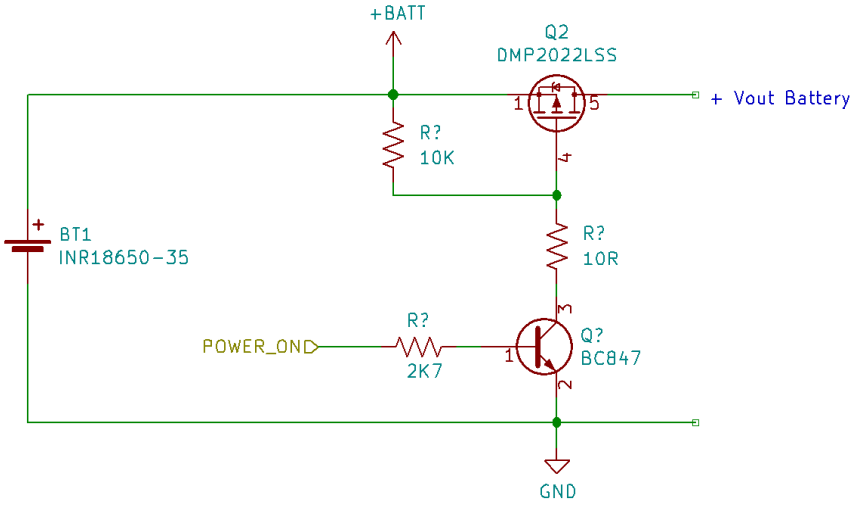


Figura 4.18: Circuito do interruptor de potência com MOSFET [Apêndice II.1].

O circuito elétrico do interruptor de potência está apresentado na Figura 4.18. A resistência R1 garante a zona de corte quando o transístor Q1 também está ao corte, evitando que o transístor entre na zona de tríodo. A resistência R2 garante a saturação de Q1 quando a porta do MCU está com nível lógico 1 $(3,3V)$. Quando o transístor Q1 está saturado obriga também à saturação de Q2. Quando o transístor Q2 está saturado permite a circulação de corrente que alimenta os sensores, o servomotor e o SoC.

DCC1 - Conversor DC/DC MCU

Os MCU são dispositivos sensíveis a sobretensões, pelo que devem ser alimentados com a tensão indicada pelo fabricante $(1,8V$ a $3,6V)$ no caso do MSP430G2553. A tensão da bateria varia entre $3,3V$ (descarregada) a $4,2V$ (carregada). Para alimentar o MCU adequadamente com a tensão recomendada de $3,3V$, é usado um regulador de tensão MCP1700-330 [14], que fornece uma tensão de saída fixa de $3,3V$ para uma corrente máxima de $250mA$. Tem um baixo consumo de energia em repouso $(1,6\mu A)$, e apresenta uma baixa queda de tensão (LDO $178mV$), o que o torna adequado para o nosso sistema.

[13] DMP2022LS - Especificações: https://www.diodes.com/assets/Datasheets/ds31373.pdf

[14] MCP1700-330 - Especificações: https://ww1.microchip.com/downloads/en/DeviceDoc/MCP1700-Low-Quiescent-Current-LDO-20001826E.pdf

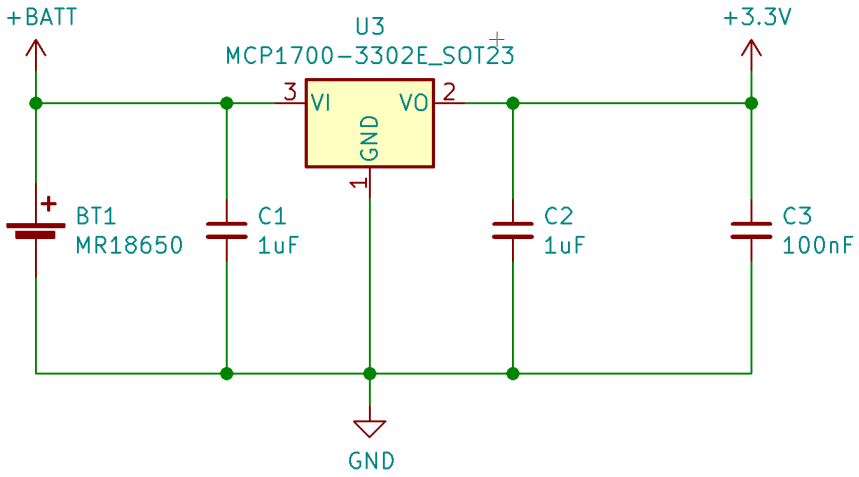


Figura 4.19: Circuito do conversor DC/DC para alimentar o MCU [Apêndice II.1].

A Figura 4.19 apresenta o circuito conversor DC/DC para alimentar o MCU. Os valores dos condensadores C1 e C2 são os recomendados pelo fabricante.

DCC2 - Conversor DC/DC - Sensores e SoC

A tensão de alimentação dos sensores e do SoC é igual ao do MCU sendo o consumo médio do conjunto é inferior a 250mA, pelo que foi escolhido o regulador de tensão MC1700-330, pelas características já apresentadas usando-se o mesmo circuito da Figura 4.19.

4.2.2 Implementação do módulo do lisímetro

A implementação do módulo foi dividida em três partes. Uma parte diz respeito ao software desenvolvido para o MCU MSP430G2553, outra para o software desenvolvido para o SoC ESP8266, e a última corresponde à concepção do hardware: placa de circuito impresso, instalação dos componentes e respectiva caixa.

Software do MCU

Para o desenvolvimento e teste do código para o MCU, foi usada uma placa de desenvolvimento MSP-EXP430G2ET [15] da *Texas Instruments*, apresentada na Figura 4.20.

A programação do MCU foi realizada recorrendo ao ambiente de desenvolvimento integrado (*Integrated Development Environment* (IDE)) *Code Composer Studio* (CCS) [96]. O

[15] EXP430G2ET - Especificações: https://www.ti.com/lit/ug/slau772a/slau772a.pdf?ts=1638397419840

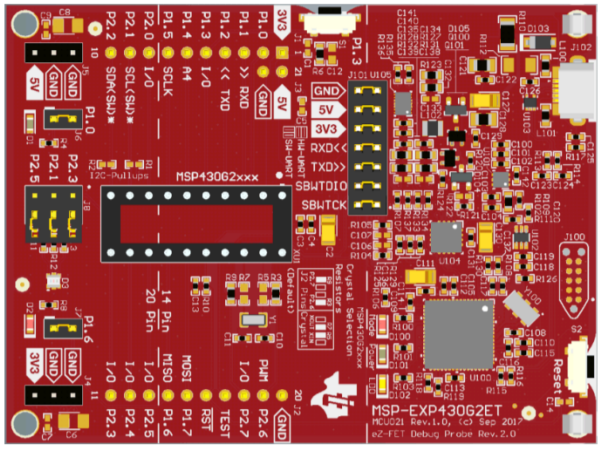


Figura 4.20: Placa de desenvolvimento EXP430G2ET [95].

código foi escrito em linguagem de programação C. A Tabela 4.3 apresenta os ficheiros de código desenvolvidos para o MCU, listando as diversas bibliotecas usadas para comunicação e interface com os sensores.

Tabela 4.3: Código desenvolvido para o MCU MSP430G2553.

Nome	Descrição	Loc.
main.c	Código principal	I.1
CDC.c	Biblioteca com funções de comunicação e conversão de dados	I.2
delay.h	Biblioteca com funções de atraso de ms e μs segundos	I.3
ds18b20.h	Biblioteca com funções dos sensores DS18B20	I.5
ds18b20.c	Biblioteca com funções dos sensores DS18B20	I.4
hx711.h	Biblioteca com funções do HX711	I.6
servo.h	Biblioteca com funções de controlo do servomotor	I.7
sht3x.h	Biblioteca com funções do sensor SHT30	I.8
TSL2561.h	Biblioteca com funções do sensor TSL2561	I.9
swi2c_master.h	Biblioteca com funções do protocolo I2C por software	I.11
swi2c_master.c	Biblioteca com funções do protocolo I2C por software	I.10

O MCU para além de fazer a leitura dos dados dos sensores, faz a gestão de energia do módulo. Pretende-se fazer a leitura dos sensores a intervalos de tempo de 10 minutos, sendo necessário menos de 60 segundos para fazer a leitura de todos os sensores, drenar a água do vaso intermédio, e enviar os dados para o SoC, que são posteriormente enviados para a camada nuvem. Como durante cerca de 90% do tempo o sistema não está a produzir trabalho, o MCU é usado para para ligar e desligar os sensores e o SoC. Após a leitura dos

sensores e a comunicação com o SoC terminar, o MCU desliga os subcircuitos entrando em modo de baixo consumo LPM. Este ciclo de leitura de dados volta a repetir-se quando é atingido o intervalo de tempo pré-definido para leitura dos dados ou caso se pressione o botão multifunções. A Figura 4.21 apresenta um diagrama com o funcionamento do software do MCU.

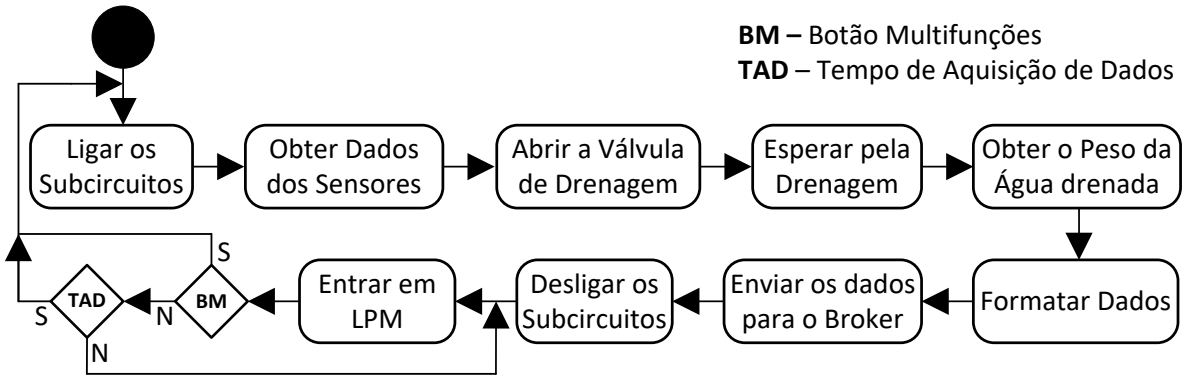


Figura 4.21: Diagrama de funcionamento do Módulo Lisímetro.

O ficheiro *main.c* contém o código principal do MCU, este foi implementada uma máquina de estados com as funcionalidades atrás descritas. Este programa faz inclusão de bibliotecas dos sensores e utilitários, necessárias para o correto funcionamento. O cabeçalho de cada biblioteca possui a documentação sobre o autor e histórico do seu desenvolvimento.

Software do SoC

A programação do módulo SoC ESP8266 foi feita recorrendo ao IDE Arduino [16]. Para desenvolver e testar o código, usamos a placa de programação ESP8266PROG [17], específica para programar módulos ESP-01, desenvolvida pela *JOY-it*. A Figura 4.22 apresenta a placa de programação do módulo ESP-01 com um módulo ligado.

O SoC ESP8266 recebe os dados formatados vindos do MCU através da sua porta série e, depois de estabelecer uma ligação de internet, obtém a data e hora atual através de um servidor NTP [72] para colocar uma marca temporal nos dados. A seguir esses dados são empacotados num objeto *JavaScript Object Notation* (JSON) [98], e enviados para camada de nuvem usando o protocolo MQTT [76] através de uma publicação. Adicionalmente

[16] Arduino IDE 1.8.16 - Especificações: https://www.arduino.cc/en/software
[17] ESP8266PROG - Manual: https://joy-it.net/files/files/Produkte/SBC-ESp8266-Prog/SBC-ESP8266-Prog-Manual.pdf

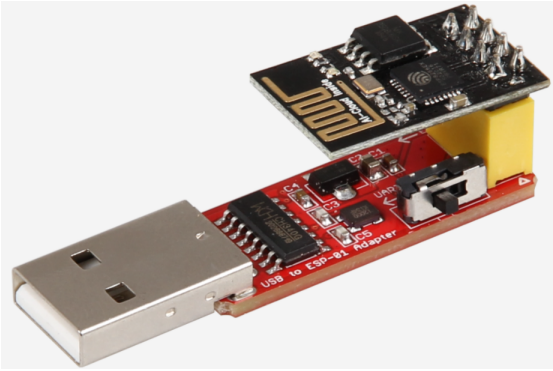


Figura 4.22: Placa de programação de módulos ESP-01 [97].

também é enviado o nível de sinal da rede WiFi, para aferir a qualidade da ligação com a *internet*.

O código da função *loop()* com a implementação das funcionalidades pretendidas, pode ser visualizado no Apêndice I.12. A Tabela 4.4 lista os ficheiros de código usado no SoC ESP8266.

Tabela 4.4: Código usado no SoC ESP8266.

Nome	Descrição
MQTT_Lysimeter.ino	Código principal I.12.
NTPClient.h	Cliente NTP, para obter a data e a hora da *internet*.
EspMQTTClient.h	Cliente MQTT para ESP8266.
WiFiManager.h	Gestor de rede Wi-Fi.
EasyButton.h	Utilitário Wi-Fi para ligação à rede pela 1ª vez.

Apenas foi desenvolvido de raiz o código principal **"MQTT_Lisimetro.ino"** sendo que todas as outras bibliotecas foram obtidas pela partilha de código desenvolvido por terceiros. A origem das bibliotecas é apresentada sobre a forma de um comentário junto da diretiva de importação das mesmas no código principal. Estas bibliotecas foram previamente instaladas no IDE *Arduino*.

Hardware

O esquema elétrico completo do módulo lisímetro é apresentado na Figura 4.23. O esquema do circuito elétrico foi realizado com o software (*Electronics Design Automation* (EDA))

KiCad [18], que também permite o desenho das placas de circuito impresso (*Printed Circuit Board* (PCB)).

Este circuito foi realizado com uma placa de circuito impresso perfurada ponto a ponto, instalada dentro de uma caixa IP65 com dimensões 82.1x158.5x55mm. A estanquidade da passagem dos diversos cabos dos sensores é assegurada por meio da utilização de bucins. A Figura 4.24 mostra imagens da caixa do módulo lisímetro e do seu interior.

O Apêndice III.1 apresenta um orçamento detalhado dos custos com material para construir o módulo, tendo sido despendida a importância de cerca de €240 para aquisição do material necessário.

[18] KiCad - Características: https://www.kicad.org/

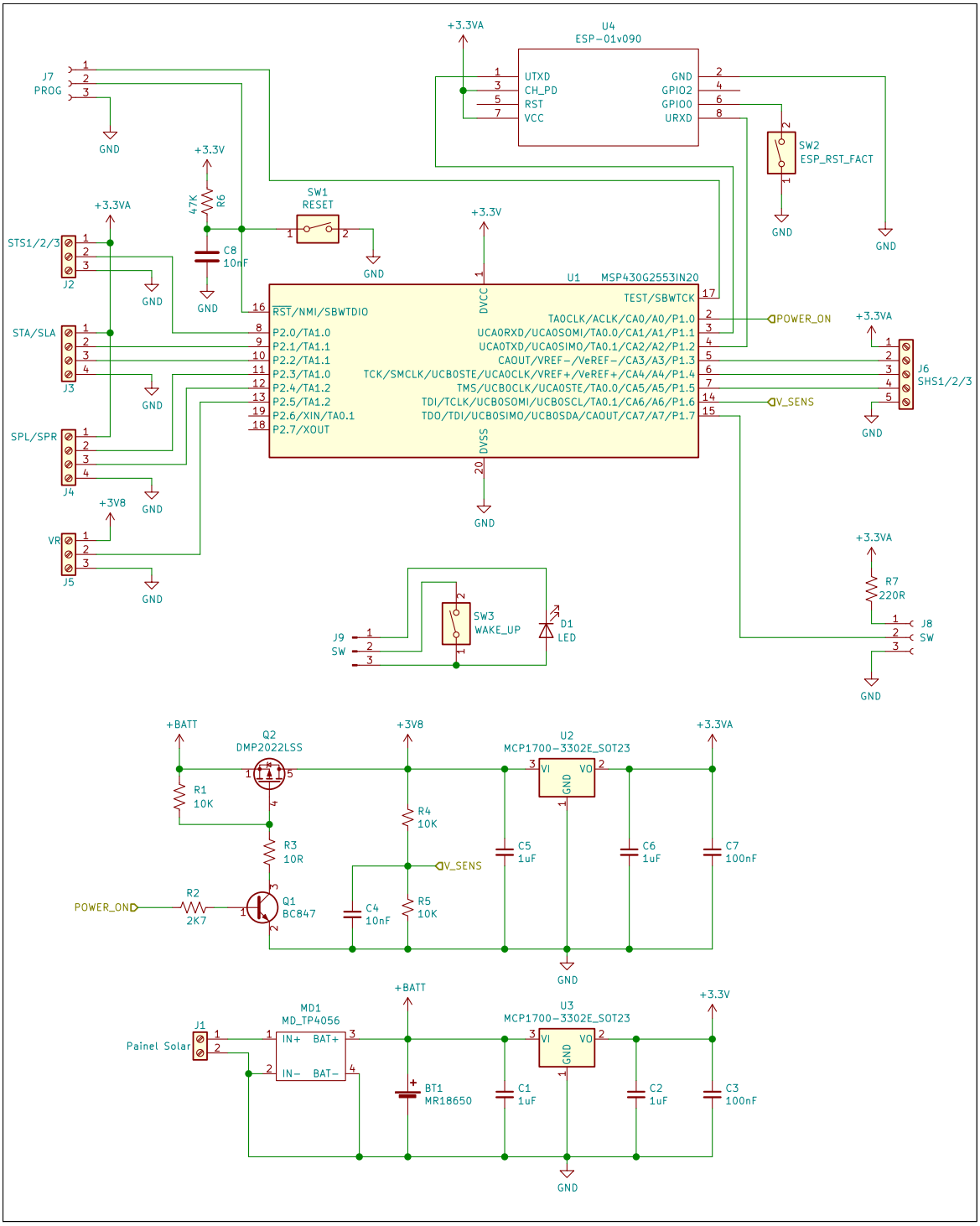


Figura 4.23: Esquema Elétrico Completo do Módulo Lisímetro

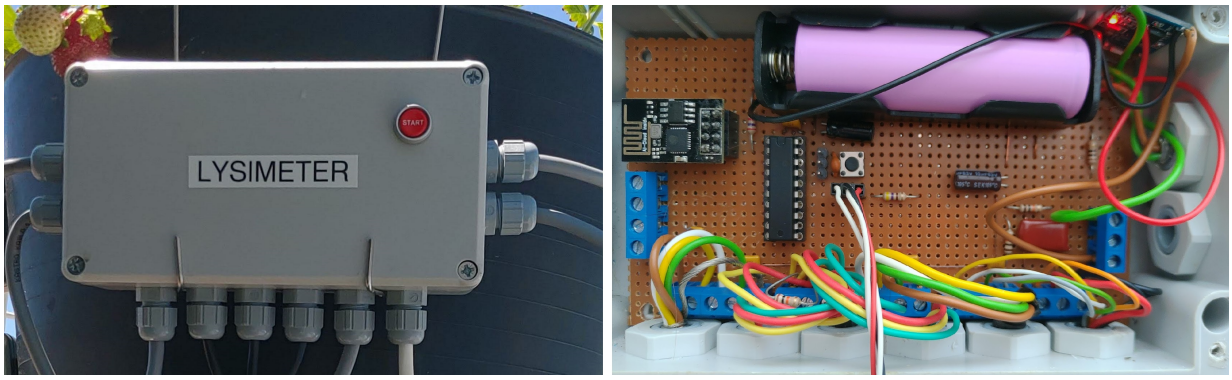


Figura 4.24: Caixa do módulo lisímetro

Figura 4.25: Visão geral do lisímetro.

4.3 Módulo Câmara

A estrutura de hardware do módulo da câmara é apresentada na Figura 4.26. A Tabela 4.5 apresenta a listagem e legenda dos componentes usados para implementação, incluindo as especificações mais importantes. O diagrama ilustra as interligações entre os diversos

dispositivos.

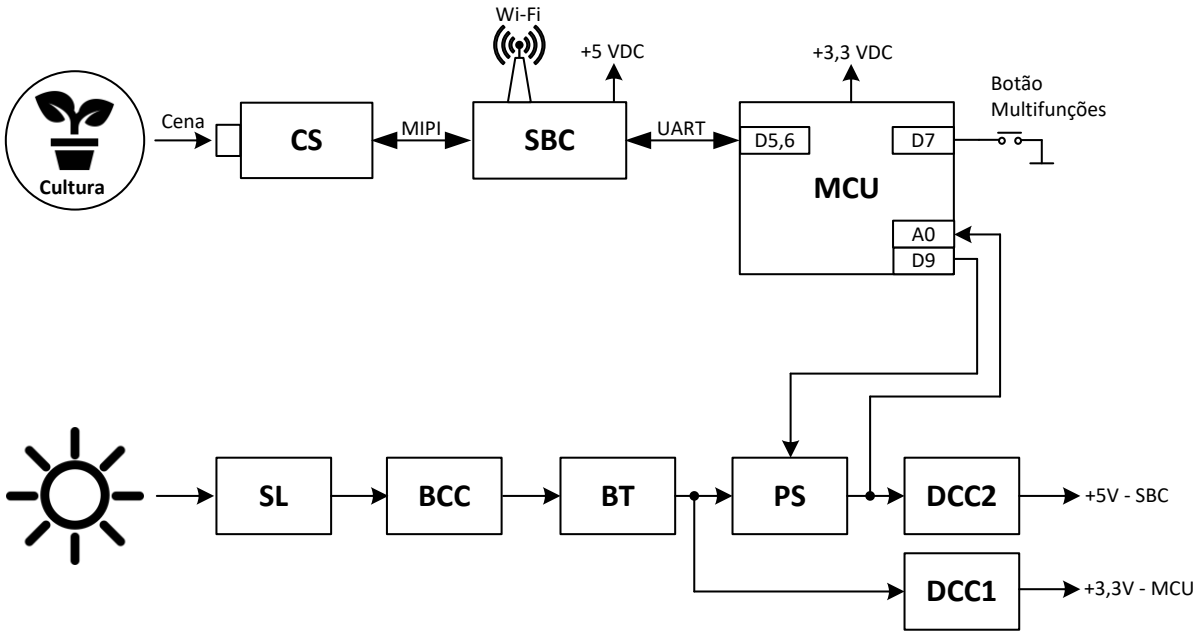


Figura 4.26: Diagrama de blocos do módulo câmara [2].

Tabela 4.5: Componentes de hardware do módulo câmara.

	Descrição	**Modelo**	**Especificações**
CS	Sensor de Imagem	RPi Cam. Mod. 2	IMX219, 8Mpx sensor
SBC	Computador numa Placa	RPi 3B+	1.4 GHz, BCM2837, 4GB
MCU	Microcontrolador	MSP430G2553	16 MHz, 16KB/512B
SL	Painel Solar Fotovoltaico	FAL09004	5VDC, 1W, 100x80mm
BCC	Controlador Carga de Bateria	TP4056 Module	1A, 1xBAT Li-Ion
BT	Bateria Li-Ion	INR18650-35E	3.6V, 3500mAh Samsung
PS	Interruptor de Potência	DMP2022LSS	-10A, P-MOSFET
DCC1	Conversor DC/DC	MCP1700-3302E	3.3V, 250mA, LDO
DCC2	Conversor DC/DC	VMA402 Module	5V, 2A, Step-Up LM2577

4.3.1 Escolha de componentes de hardware

Este módulo que partilha alguns dos componentes com as mesmas especificações do módulo lisímetro, sendo, a seguir apresentada a escolha, descrição e justificação dos novos

componentes usados neste módulo.

Componentes apresentados e descritos no módulo lisímetro na Secção [4.2.1]: MCU, SL, BCC, BT, PSW, BVS e DCC1.

CS - Sensor de Imagem

Para capturar a cena foi escolhido a 2ª versão do módulo de câmara Raspberry Pi [19]. Este módulo usa o sensor de imagem IMX219 da *Sony*, com uma resolução de 8Mpx. Este sensor apresenta uma excelente qualidade de imagem, em condições de iluminação exterior diurna. A Figura 4.27 apresenta a imagem deste módulo.

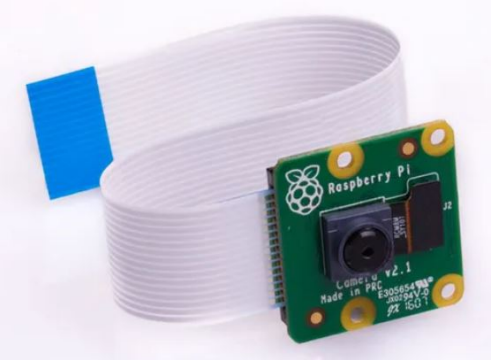


Figura 4.27: Módulo câmara Raspberry Pi V2 [99].

SBC - Computador numa Placa

Para processar as imagens capturadas é necessário um dispositivo com capacidade de processamento elevado e com uma interface para ligação ao módulo de câmara atrás descrito. A escolha recaiu no SBC Raspberry Pi (RPi) 3B+ [20] . Este SBC tem internamente o SoC Broadcom BCM2837B0 com um CPU *Advanced RISC Machine* (ARM) [100] *quad-core*, de 64bits a funcionar a uma frequência de 1,4GHz. Tem enumeras funcionalidades, contudo destaca-se a porta *Camera Serial Interface* (CSI) [101] para ligação da câmara, rede WLAN integrada, e a exposição de 40 pinos *General Purpose Input/Output* (GPIO), necessárias para a implementação deste módulo. A Figura 4.28 apresenta uma imagem do SBC Raspberry Pi 3B+.

[19] Módulo de Câmara RPi V2 - Especificações: https://www.raspberrypi.com/products/camera-module-v2/

[20] RPi 3B+ - Especificações: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

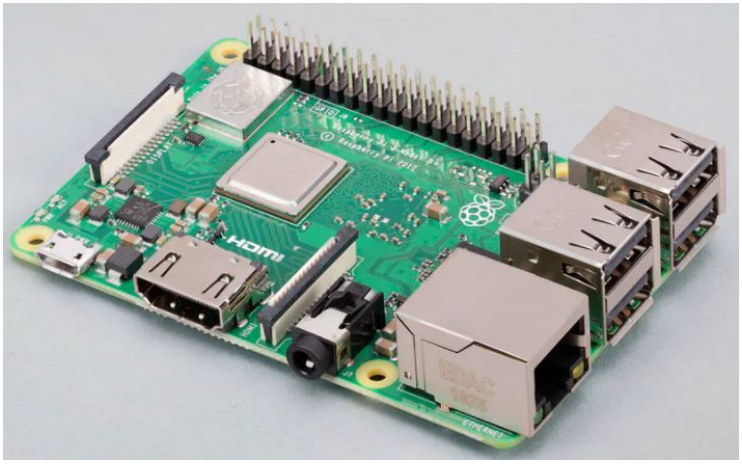


Figura 4.28: SBC Raspberry Pi 3B+ [102].

DCC2 - Conversor DC/DC SBC

A bateria escolhida tem uma tensão nominal de 3.6V, variando entre os 3.3V, quando descarregada, e 4.2V quando está completamente carregada. No entanto, o SBC Raspberry Pi 3B+ necessita de uma tensão estabilizada de 5V e uma corrente máxima de 2.5A, para isso utilizou-se um conversor DC/DC elevador de tensão para o alimentar. Este conversor é baseado no CI LM2577 [21] é produzido pela *Texas Instruments*. O módulo com a referência VMA402 [22] produzido pela *Velleman* permite o ajuste da tensão de saída e fornecer a corrente necessária ao correto funcionamento do SBC. A Figura 4.29 apresenta a imagem deste módulo.

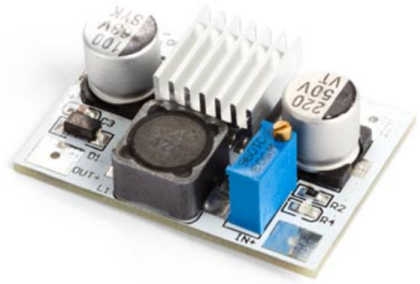


Figura 4.29: Módulo VMA402 - Conversor DC/DC [103].

[21] LM2577 - Especificações:https://www.ti.com/lit/ds/symlink/lm2577.pdf
[22] VMA402 - Especificações:https://www.velleman.eu/products/view?id=435562

4.3.2 Implementação do módulo câmara

A implementação do módulo câmara foi dividida em três partes. Uma parte diz respeito ao software desenvolvido para o MCU MSP430G2553, outra para o software desenvolvido para o SBC *Raspberry Pi*, e por último a concepção do hardware, nomeadamente: a placa de circuito impresso, a instalação dos componentes e respetiva caixa.

Software do MCU

Para o desenvolvimento e teste do código para o MCU, foram usadas a mesmas ferramentas enumeradas no módulo lisímetro, nomeadamente a placa de desenvolvimento MSP-EXP430G2ET [95], e o IDE CCS [96]. A Tabela 4.6 apresenta a listagem dos ficheiros de código implementado no MCU do módulo da câmara.

Tabela 4.6: Código desenvolvido para o MCU do módulo da câmara.

Nome	Descrição	Loc.
main.c	Código principal	I.13
CDC.c	Biblioteca com funções de comunicação e conversão de dados	I.2
delay.h	Biblioteca com funções de atraso de ms e μs segundos	I.3

Neste módulo, o MCU é responsável por fazer a gestão de energia do módulo e monitorizar a tensão da bateria. O circuito para a medição da tensão da bateria é idêntico ao do módulo lisímetro já descrito anteriormente apresentado na Figura 4.13.

Quando o módulo é ligado pela primeira vez, o MCU ativa a porta que controla a alimentação do SBC, sendo programado no MCU o tempo necessário para o SBC fazer o arranque do Sistema Operativo (SO) *Raspberry Pi OS*. Após esse tempo o MCU usa o seu ADC interno fazer a leitura da tensão da bateria e envia essa informação através da porta série para o SBC. Depois disso, o MCU aguarda que o SBC execute e complete as operações que tem predefinidas e faça o encerramento do SO, tornando possível desligar a energia do SBC em segurança. De seguida o MCU calcula o tempo necessário para a próxima leitura de imagem, com base na tabela de leituras, e afeta a variável que guarda o intervalo de tempo entre leituras, entrando de seguida em modo de poupança de energia (LPM). O MCU fica em LPM até ter decorrido o intervalo de tempo predefinido para uma nova aquisição de imagem, ou quando é pressionado o botão multifunções.

O ficheiro *main.c* contém o código principal do MCU, onde está implementada uma máquina de estados com as funcionalidades descritas, que pode ser consultado no Apêndice I.

Software do SBC

Depois de instalar o sistema operativo *Raspberry Pi OS* [104] no SBC, na primeira vez que o SO arranca, é configurada a rede Wi-Fi, é ativada a porta série, a porta CSI da câmara, o serviço *Secure Shell* (SSH), e o auto *login*. Posteriormente foi criada a pasta de trabalho *"lysimeter"* para guardar o código desenvolvido e uma sub-pasta *"image"* para guardar as fotos adquiridas. Depois de atualizar o SO, foi instalado um conjunto de bibliotecas para se ter as funcionalidades pretendidas. A Tabela 4.7 apresenta as bibliotecas instadas e as respetivas funcionalidades.

Tabela 4.7: Bibliotecas instaladas no SBC RPi

Nome	Versão	Descrição	Ref.:
paho-mqtt	1.6.1	Cliente MQTT	[105]
picamera	1.13	Captura de imagem da câmara	[106]
pyserial	3.5	Comunicação com a porta série	[107]

O código para o SBC RPi foi desenvolvido usando a linguagem de programação Python[23], decomposto em cinco *scripts*:

- **start.py** - Código principal [I.14];
- **shutd.py** - Encerra o sistema operativo [I.18];
- **mqtt.py** - Aguarda dados na porta série e envia para a *cloud* via MQTT [I.17];
- **camera.py** - Tira uma fotografia e guarda na pasta *Image* [I.15];
- **copyimage.py** - Envia as imagens da pasta *Image* e envia para o servidor SFTP [I.16] ;

O código *start.py* é executado no arranque do SO com a adição de uma linha para a execução do código *"python /home/pi/lysimeter/start.py"* no ficheiro *.bashrc*. Este *script* começa por tirar uma fotografia executando a função *takePhoto()*, posteriormente envia essa fotografia para o servidor *Secure File Transfer Protocol* (SFTP) ao executar a função *copyImage()*, e envia os dados do estado da câmara num objeto JSON via protocolo MQTT, para a *cloud* com a execução da função *mqtt()*. Terminadas as tarefas é chamada a função *shutd()* para encerrar o sistema operativo.

[23]Python - Linguagem de programação de alto nível [55]

Para enviar as imagens para o servidor SFTP sem necessidade de fazer *login* foi gerado um par de chaves de autenticação pública/privada no SBC e posteriormente guardada no servidor remoto [108].

Hardware

O esquema do circuito elétrico completo do módulo câmara II.2 foi realizado com o software EDA Kicad à semelhança do módulo lisímetro, tendo sido igualmente usado para desenvolver o desenho de uma placa de circuito impresso PCB da Figura II.4 II.3.

A PCB foi produzida pelo processo de fresagem *Computer Numerical Control* (CNC) [109]. Através da conversão do desenho da PCB em código compatível com a CNC (G-Code [24]) para o isolamento das pistas, a furação e o corte foi efetuado com a aplicação FlatCAM [25].

O esquema elétrico do módulo câmara é apresentado no Apêndice II.2, é possível ver o aspecto da parte frontal da PCB simulada no apêndice II.3 e a parte traseira no apêndice II.4

A placa de circuito impresso produzida, juntamente com o SBC RPi, foi instalada dentro de uma caixa IP65 com dimensões de 82.1x158.5x55mm. No seu exterior foi acoplado o painel solar. A Figura 4.30 apresenta o aspeto exterior do módulo câmara.

Figura 4.30: Imagem do exterior do módulo câmara

Para realização deste módulo foi despendido cerca de €120 na aquisição de material e componentes electrónicos necessários. O Apêndice III.2 apresenta um orçamento detalhado dos custos com material para construção do o módulo câmara.

[24] G-Code - Código G, conjunto de instruções compatível com CNC [110]
[25] FlatCAM - Aplicação para conversão de desenhos em formato compatível com CNC [111].

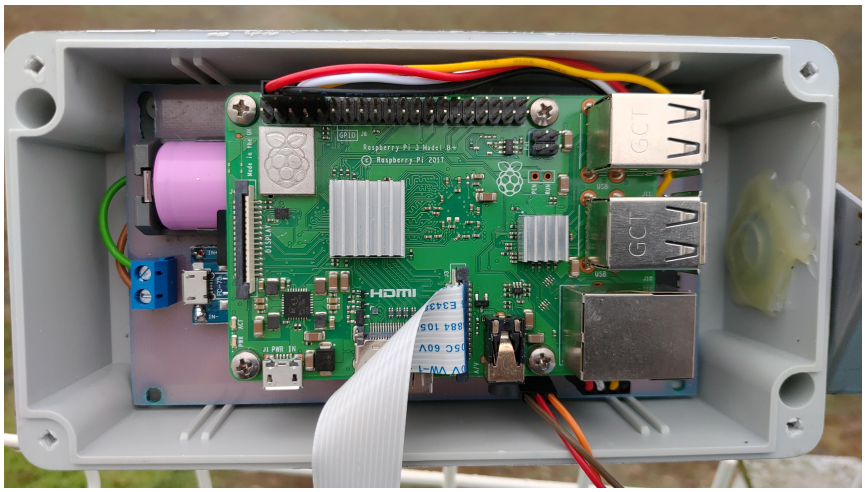


Figura 4.31: Imagem do interior do módulo câmara

4.4 Módulo de Computação na Nuvem

A estrutura de software do módulo computação na nuvem é apresentado na Figura 4.32.

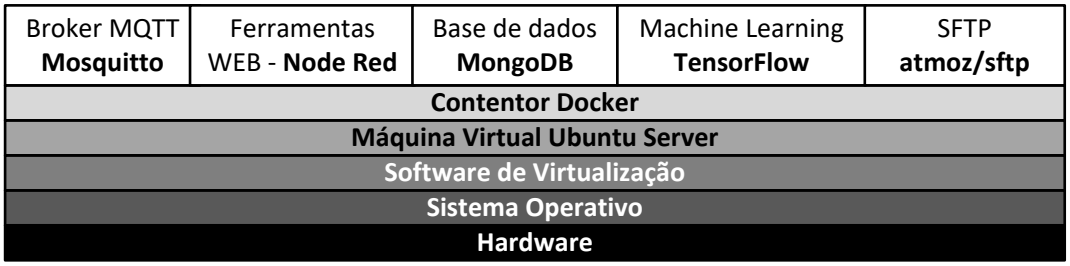


Figura 4.32: Virtualização dos serviços na nuvem [2].

O módulo de computação na nuvem foi implementado numa máquina virtual (*Virtual Machine* (VM) [112]) com o sistema operativo *Ubuntu Server* 20.04 LTS [113]. A utilização de máquinas virtuais permite portabilidade entre sistemas e máquinas físicas de uma maneira rápida e eficiente.

Todos os serviços de software são executados em contentores *Docker* [114], que apresentam as seguintes características:

- Portabilidade de aplicações;
- Isolamento de processos;

- Prevenção de violação externa;
- Gestão do consumo de recursos porque requerem menos recursos que as máquinas virtuais tradicionais usadas na implantação de aplicações isoladas.

A Figura 4.33 apresenta à esquerda a estrutura de um sistema baseado em *dockers* e à direita outro baseado em máquinas virtuais.

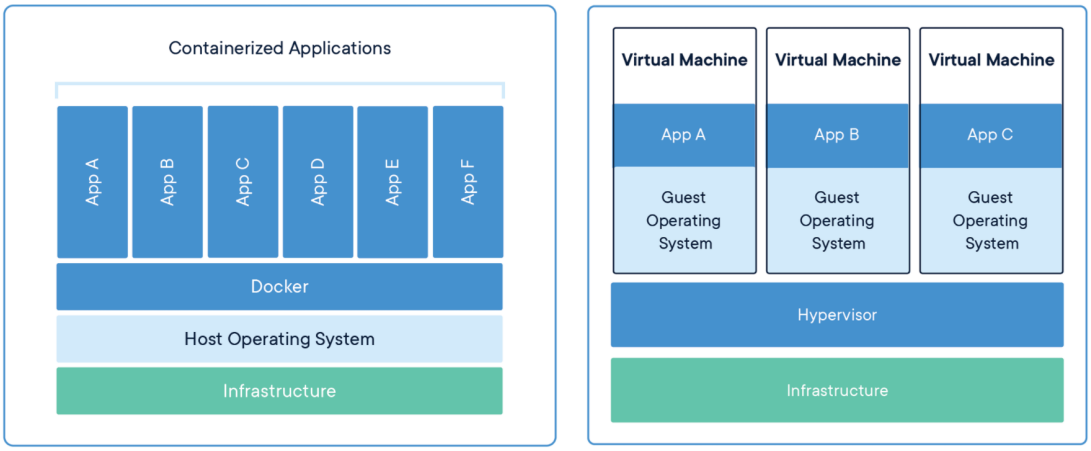


Figura 4.33: Comparativo entre contentores *docker* e máquinas virtuais [114].

Os serviços que estão a correr em *dockers* são os seguintes:

- **Mosquitto** [115] - Broker [26] MQTT.
- **Node-RED** [116] - Ferramenta de programação visual baseada em *Node.js* [117].
- **MongoDB** [118] - Servidor de base de dados não relacional;
- **AT-MOZ** [119] - Servidor de transferência de ficheiros seguro (SFTP);
- **TensorFlow** [120] - Plataforma para *Machine Learning* (ainda não implementado).

O MQTT é um protocolo para troca de mensagens caracterizado por usar pouca largura de banda e recursos de hardware reduzidos. Foi desenvolvido pela IBM e Eurotech na década dos anos 90, seguindo um padrão de publicação/subscrição. Quando um elemento deseja receber informação relacionada com um determinado tema faz uma subscrição ao elemento que gere as publicações e subscrições associadas a esse tema, designado por *Broker*, da mesma forma os elementos que desejam publicar informações podem fazê-lo por

[26] Broker - Elemento responsável por gerir as subscrições/publicações MQTT

intermédio do *Broker*. A Figura 4.34 apresenta o diagrama simplificado do funcionamento do protocolo MQTT.

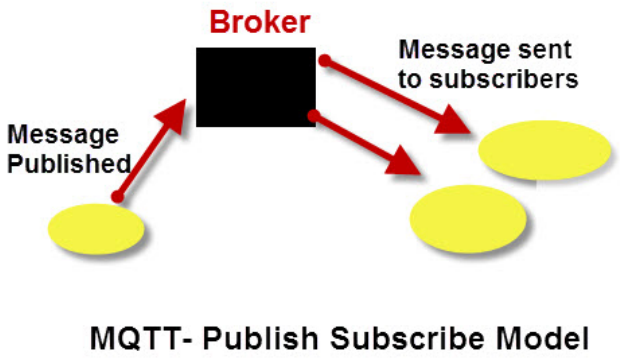


Figura 4.34: Diagrama de funcionamento do protocolo MQTT [121].

No nosso caso prático, os diversos lisímetros fazem publicações dos dados no *Broker* com o tópico "LISIMETRO", o Node-Red faz a subscrição no mesmo tópico para receber os dados. Os lisímetros fazem ligação ao *Broker* através do *Uniform Resource Locator* (URL): `lysimeter.ddnsfree.com` usando a porta 1883.

O *Node-RED* [116] é uma ferramenta de programação visual de código aberto, possui várias APIs e serviços online, tendo sido criado pela *IBM Emerging Technology*. Possui um editor visual que permite arrastar e soltar elementos que trabalha diretamente no navegador (*Browser*).O Node-RED possui vários nós que podem ser arrastados e colocados no ecrã, permitindo programar de forma nativa em *JavaScript* [122], ou outras linguagens de programação através da instalação de pacotes. A Figura 4.35 apresenta o diagrama de fluxo Node-RED do protótipo.

O nó *MQTT* faz a subscrição do tópico "LISIMETRO". Sempre que um lisímetro faz uma publicação no *Broker* esse nó recebe como *payload* um objeto JSON, com os dados do referido lisímetro. Depois o nó *Data* decompõe o objeto JSON nos diversos campos dos sensores através do código em *JavaScript* listado em I.19. Os dados já unificados são enviados para nós do tipo *Dashboard* para posteriormente serem apresentados no painel de visualização. Foram criadas duas páginas para visualização dos dados do lisímetro, que são:

- Indicadores tipo manómetro, com indicação da última leitura enviada pelo lisímetro (Fig. 4.36), com o URL: `http://lysimeter.ddnsfree.com:1880/ui/#!/0`;

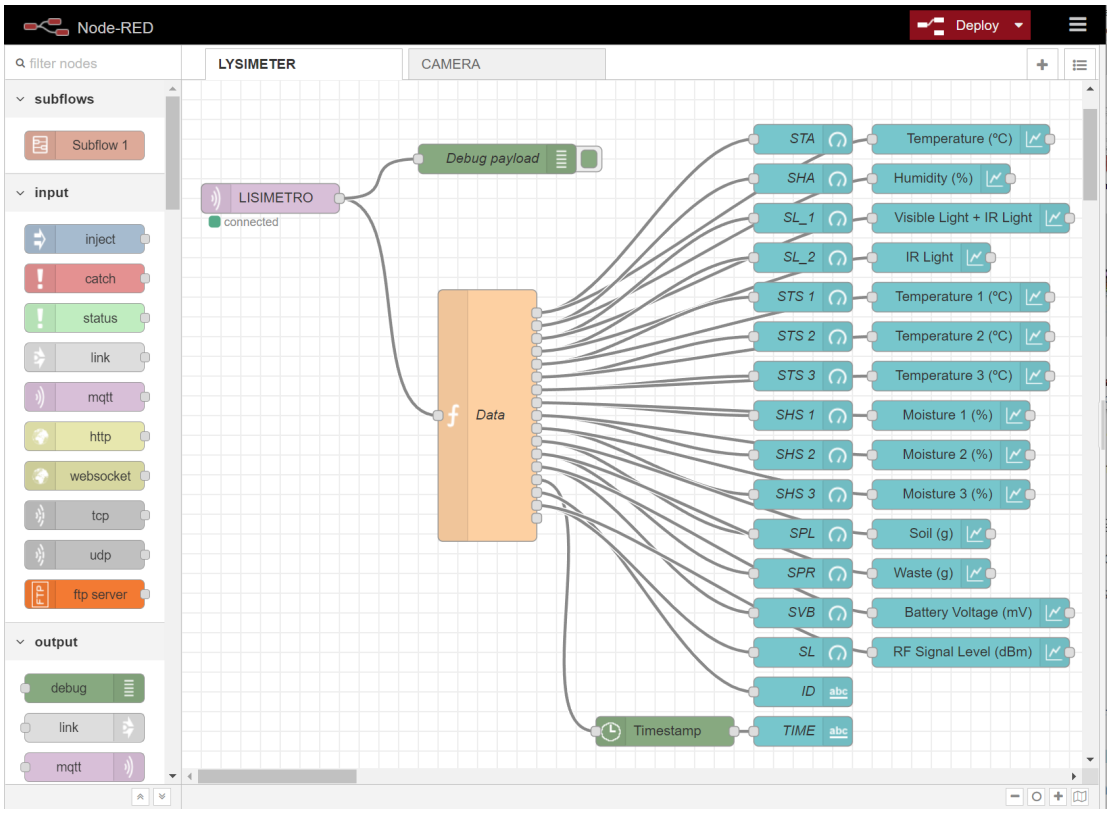


Figura 4.35: Diagrama de fluxo do Node-Red.

- Gráficos com os dados recebidos do lisímetro relativo às ultimas 48H (Figura 4.37), com o URL: `http://lysimeter.ddnsfree.com:1880/ui/#!/1`.

O *mongoDB* [118] é um sistema de gestão de base de dados não relacional, do tipo NoSQL [77], orientado a documentos, de código aberto e multi-plataforma. As bases de dados de documentos ampliam o conceito da base de dados de chave-valor ao organizarem documentos inteiros em coleções. Suportam pares chave-valor alinhados e permitem consultas de qualquer atributo num documento [77]. As principais características do *mongoDB* são: código aberto, elevado desempenho, alta disponibilidade, e escalabilidade automática.

No desenvolvimento do protótipo do lísimetro, o *mongoDB* foi instalado num contentor *docker* para fins de teste, contudo, para explorar ao máximo as características atrás mencionadas, nomeadamente o facto de ser uma base de dados distribuída, é necessário termos um *cluster* [27] com várias máquinas espalhadas numa rede de computadores.

O *AT-MOPS* é um servidor de ficheiros seguro SFTP, com *Open* SSH [124] disponível para instalação num contentor (*container*). Este servidor é compatível com acesso através

[27] Cluster - Conjunto de máquinas a trabalhar na mesma tarefa [123]

Figura 4.36: Painel de indicadores dos sensores do lisímetro.

de chave pública/privada (*Rivest-Shamir-Adleman* (RSA) [125]). O acesso ao servidor SFTP sem necessidade de autenticação por utilizador e palavra-passe é necessário para que o módulo câmara possa enviar as imagens capturadas para o servidor. No SBC do módulo câmara são geradas as chaves RSA (pública e privada), e depois a chave pública é copiada para o servidor SFTP. Lista de comandos usados para configurar o acesso ao servidor através de chaves RSA:

```
# Verifica se existe alguma chave
ls -al ~/.ssh/id_*.pub
# Gera um par de chaves RSA
ssh-keygen -t rsa -b 4096 -C "18510@stu.ipbeja.pt"
# Copia chave publica para o servidor
ssh-copy-id lisimetro@lysimeter.ddnsfree.com
# Testar a ligação SSH sem password
ssh lisimetro@lysimeter.ddnsfree.com
# Sair do SSH
exit
```

Figura 4.37: Painel com gráficos das últimas 48h dos sensores do lisímetro.

4.5 Estrutura de Suporte

De modo albergar os 3 vasos do lisímetro como representado na Figura 3.1, foi necessário dimensionar uma estrutura metálica de suporte. A dimensão desta estrutura pode ser ajustada em função da cultura pretendida e das dimensões o lisímetro. No caso, a estrutura foi dimensionada para uma cultura de morangueiros em vaso.

Para a colocação das plantas com o respetivo solo, foi usado um vaso cónico com as dimensões de 350x300mm (DxA) de polietileno. A colocação dos sensores de temperatura e humidade no vaso do solo foram feitas através de perfurações nas profundidades pretendidas. Para a retenção da água drenada foi utilizado um recipiente plástico cilíndrico com as dimensões de 200x120mm (DxA). Na fase de testes do protótipo não foram realizadas análises químicas à água drenada, como tal não foi usado o vaso inferior.

A partir das medidas dos vasos, foi dimensionado as medidas da estrutura. A Figura 4.38 apresenta as medidas da estrutura usada no protótipo, com os respectivos vasos.

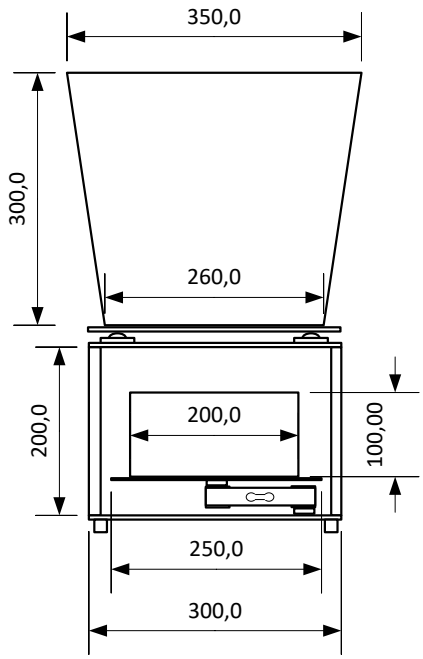


Figura 4.38: Medidas da Estrutura do Lisímetro [mm].

A estrutura foi produzida com tubos e chapa de aço galvanizado, unidas por meio de soldadura de arco elétrico, acabado com pintura. Para a construção de futuras estruturas é desejável a utilização de aço inox, para evitar a oxidação, nomeadamente junto aos pontos de soldadura, o que permite aumentar a longevidade da estrutura.

4.6 Aspetos Experimentais

Ao longo da implementação do protótipo procederam-se a diversas alterações do projeto de forma a corrigir alguns problemas e também a melhorar as características do lisímetro.

4.6.1 Construção do protótipo

Os circuitos eléctricos dos módulos foram desenhados no KiCad [126] e concretizados em placas de circuito impresso. Estas placas de circuito impresso, juntamente com os restantes componentes/módulos, foram alojados em caixas com certificação IP65 [28]. A passagem dos cabos foi efetuada através de bucins que asseguram o mesmo índice de proteção das caixas.

[28] IP65 - Índice de proteção de elementos sólidos e líquidos [127]

As figuras 4.24, 4.25, 4.30, 4.31 apresentadas anteriormente evidenciam os pormenores de construção descritos.

4.6.2 Gestão de energia

Desde o início existiu uma preocupação com a gestão de energia. Sendo o protótipo alimentado por energia solar, pretende-se o seu funcionamento mesmo após vários dias sem sol.

O módulo lisímetro em funcionamento possui um consumo de corrente máximo de $320mA$ em quando está a transmitir dados, a bateria de Li-Ion tem uma tensão nominal de $3.6V$, e são necessários cerca de $20s$ para obter os dados dos sensores e fazer a sua os transmissão para a *cloud*. Os valores de corrente, foram obtidos através de medições no protótipo. A Figura 4.39 apresenta o cálculo da energia diária do módulo lisímetro.

Consumo de energia diário	
Intervalo entre leituras	10 min
Leituras por hora	6
Leituras por dia	144
Tempo em func. por leitura	20 s
Tempo em func. por dia	2880 s
	48 min
	0,8 h
Consumo corrente em LPM	0,3 mA
Consumo corrente em funcionamento	320 mA
Tensão da bateria média	3,6 V
Potência consumida em funcionamento	1,152 W
Potência consumida em LPM	0,0011 W
Energia consumida em funcionamento	0,9216 Wh
Energia consumida em LPM	0,0251 Wh
Consumo de Energia total por dia	**0,9467 Wh**

Painel Solar	
Energia	1 Wh
Tempo exposição solar	4 h/d
Energia produzida por dia	**4 Wh**

Bateria 18650	
Tensão média	3,6 V
Capacidade	3500 mAh
Energia	**12,6 Wh**

Tempo funcionamneto sem painel solar	**13,3 dias**

Figura 4.39: Cálculo da energia diária do módulo lisímetro.

O módulo câmara em funcionamento tem um consumo de corrente máximo de $1580mA$ em operação. Possui uma bateria com uma tensão média de $3.6V$, e necessita de $60s$ para iniciar o SO, tirar a fotografia, e enviar para o servidor SFTP. A Figura 4.40 apresenta o cálculo da energia diária do módulo lisímetro.

Esta folha de cálculo permite, de uma forma simples estimar a autonomia dos módulos na ausência de luz solar. Como o cálculo é feito com base no consumo de corrente máximo dos módulos, os valores das autonomias reais serão sempre maiores.

A folha de cálculo também apresenta uma estimativa da energia produzida por dia pelo painel solar, considerando um tempo de exposição de 4h diárias (Inverno), desta forma é

Consumo de energia diário	
Leituras por dia	3
Tempo em func. por leitura	60 s
Tempo em func. por dia	180 s
	3 m
	0,05 h
Consumo corrente em LPM	0,3 mA
Consumo corrente em funcionamento	1580 mA
Tensão da bateria média	3,6 V
Potência consumida em funcionamento	5,688 W
Potência consumida em LPM	0,0011 W
Energia consumida em funcionamento	0,2844 Wh
Energia consumida em LPM	0,0259 Wh
Consumo de Energia total por dia	**0,3103 Wh**

Painel Solar	
Energia	1 Wh
Tempo exposição solar	4 h/d
Energia produzida por dia	**4 Wh**

Bateria 18650	
Tensão média	3,6 V
Capacidade	3500 mAh
Energia	**12,6 Wh**

Tempo funcionamneto sem painel solar	**40,6 dias**

Figura 4.40: Cálculo da energia diária do módulo câmara.

possível inferir se a produção de energia no inverno é suficiente para manter o módulo em funcionamento. Em qualquer dos casos a energia produzida diária é quatro vezes superior à necessária.

O tempo de funcionamento sem painel solar é calculado em função do consumo de energia diário e da capacidade da bateria utilizada. Os cálculos apresentados nas figuras anteriores usam uma bateria com uma capacidade de $3500mAh$, sendo possível alterar a capacidade da bateria e obter novos valores.

Para comprovar os cálculos apresentados foi desligado o módulo lisímetro em 25/09/2021 pelas 20h00, em que a tensão da bateria era de $4.2V$, apresentando-se completamente carregada. O sistema ficou a funcionar até a carga da bateria rondar os 30%, no dia 12/10/2021 pelas 9h15 verificou-se que a tensão da bateria era de $3.42V$, tendo sido reposto a ligação do painel solar. Assim, foi possível comprovar uma autonomia de cerca de 16,5 dias. A bateria recuperou a carga máxima passado 3 dias, de acordo com o valor também expectável considerando o painel e a bateria usados.

4.6.3 Visualização dos dados

Os valores dos dados enviados pelo lisímetro são visualizados num painel usando um cliente com navegador WEB (Fig.: 4.36). Os valores recolhidos durante um período de 2 dias são disponibilizados em forma de gráfico, para poder inferir a evolução e tendência das grandezas (Fig.: 4.37).

O serviço WEB é compatível com a moderna tecnologia *Progressive Web App* (PWA) [128], sendo possível aceder aos dados do lisímetro a partir de qualquer dispositivo móvel através

de uma interface similar de aplicativo para *Smartphone*. A Figura 4.41 apresenta a imagem dos dados dos sensores num *Smartphone*.

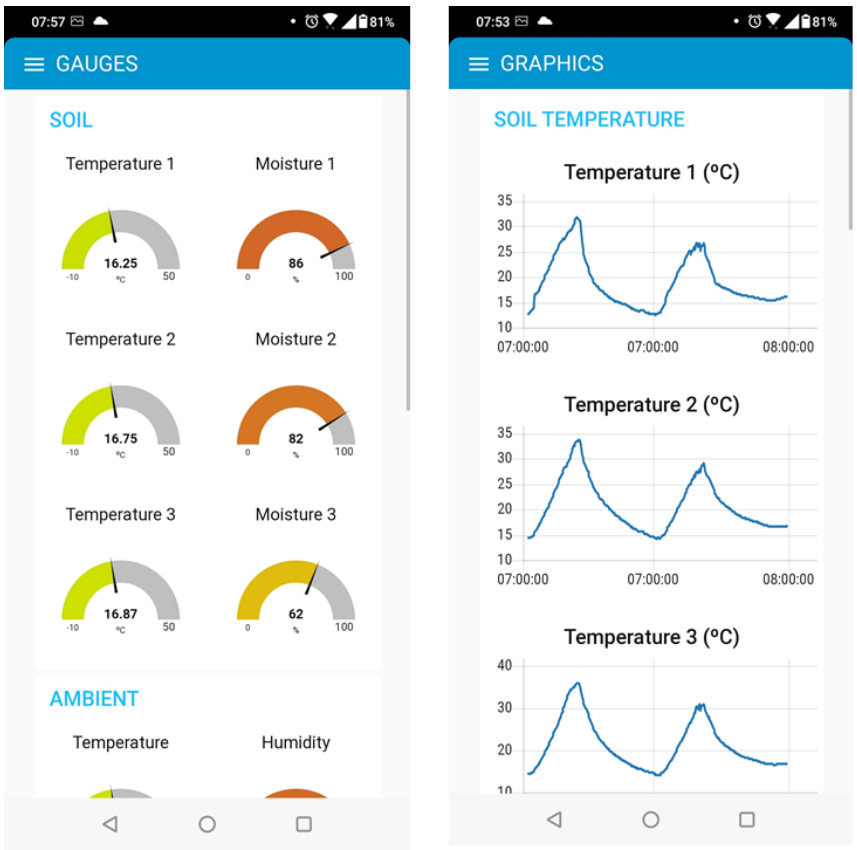


Figura 4.41: Dados dos sensores apresentados num *Smartphone*.

As imagens capturadas e enviadas para o servidor SFTP, podem ser acedidas através de um cliente SFTP, nomeadamente a aplicação *FileZilla* [129], através do URL: `lysimeter.ddnsfree.com`.

A Figura 4.42 apresenta uma imagem de alta resolução das plantas, adquirida pelo módulo câmara, à direita é apresentada uma imagem detalhada do fruto onde é possível detectar o estado das plantas a olho nu. No entanto, o objetivo futuro é ter um sistema baseado em ML a funcionar na nuvem, capaz de detetar o estado dos frutos, as possíveis doenças das plantas, e disparar automaticamente um aviso e enviar um alerta aos utilizadores pré-definidos através de e-mail ou *Short Message Service* (SMS).

Figura 4.42: Direita: Exemplo de uma imagem capturada, Esquerda: Detalhe da imagem ampliada.

4.7 Conclusão

O protótipo de lisímetro apresentado exemplifica a aplicação do paradigma IoT na resolução de um problema comum e fundamental na agricultura, a medição do equilíbrio evaporação-transpiração.

Vários outros parâmetros físicos do solo e ambientes são medidos, assim como um conjunto de imagens para monitorizar a evolução das plantas. Estes dados podem ser usados para detectar doenças e pragas usando um sistema baseado em *Machine Learning*.

Os módulos lisímetro e câmara superaram a autonomia calculada, o que indica que o consumo de energia é inferior ao estimado.

O sistema de lisímetro proposto oferece uma solução integrada de baixo custo e eficiência energética para a medição da evapotranspiração, que pode ser usada em diferentes locais e cenários, nomeadamente diferentes culturas.

O servidor WEB com tecnologia PWA permite um acesso ao sistema de uma forma transversal, sendo possível aceder aos dados do lisímetro a partir de qualquer dispositivo móvel através de uma interface similar de aplicativo para *Smartphone*.

A ferramenta de programação visual Node-RED, além de criar um *Dashboard* num página WEB com a visualização dos dados lisímetro, permite fazer *Debug* das comunicações, nomeadamente ver o *Payload* das mensagens enviados pelo lisímetro.

Capítulo 5

Conclusões

Neste capítulo são apresentadas as conclusões gerais e as perspetivas de desenvolvimento futuro.

5.1 Conclusões gerais

O protótipo de lisímetro apresentado exemplifica a aplicação do paradigma IoT na resolução de um problema comum e fundamental na agricultura: a medição do equilíbrio hídrico evaporação-transpiração. Adicionalmente, são medidos parâmetros físicos do solo e do meio ambiente, sendo feita a aquisição de imagens das plantas com o objetivo de monitorizar a evolução da cultura. Estes dados podem ser usados para alimentar um sistema, baseado em ML, para deteção de doenças e pragas e predição do rendimento. O estudo realizado permite concluir que é possível desenvolver sistemas para detecção de doenças e pragas em plantas recorrendo a hardware de baixo custo.

A arquitetura proposta foi o ponto de partida para a construção de um protótipo funcional para recolha de dados da ET, do solo e do meio ambiente. A abordagem IoT permite redução de custos, elevado desempenho, escalabilidade e eficiência energética.

A arquitetura é composta por dois módulos independentes: o módulo lisímetro e o módulo câmara que, por estarem na maior parte do tempo em modo de poupança de energia LPM, são eficientes do ponto de vista energético, e podem trabalhar em condições ambientais de temperatura e humidade extremas.

O objetivo desta arquitetura é permitir a monitorização contínua da cultura com o consequente aumento da produtividade das culturas, redução do consumo de água, dado que é possível calcular a quantidade exata de água que uma determinada planta necessita.

Este lísimetro permite de forma autónoma, e automática, a recolha em contínuo de dados para alimentar sistemas de ML/*Artificial Intelligence* (AI), que são caracterizados por necessitarem de grandes quantidades de dados de treino de modo a alcançarem uma precisão elevada. Para otimizar os resultados é importante que os agricultores possam interagir com o sistema através de uma página WEB, registando na plataforma a sua análise, porventura mais experiente, da cultura sempre que sejam feitas visitas.

A conjugação de dados proveniente dos sensores de lisímetros inteligentes, do tipo proposto e desenvolvido, com o processamento das imagens adquiridas pela câmara numa rede neural convolucional irá permitir maior precisão e fiabilidade na detecção e prevenção de doenças e pragas em plantas, assim como aprofundar o estudo da evolução das plantas em função dos parâmetros ambientais.

Assim, o desenvolvimento e construção de um protótipo funcional permitiu comprovar a arquitetura proposta, sendo o primeiro passo para a construção de um sistema completo para monitorização das culturas.

O sistema de lisímetro proposto compreende os sistemas completos de aquisição de dados, comunicação para a *Cloud*, armazenamento em base de dados e visualização remota dos dados, consistindo numa solução integrada de baixo custo e eficiência energética para a medição da evapotranspiração, que pode ser usado em diferentes locais e cenários, e com diferentes culturas. É um sistema escalável em duas vertentes: o lisímetro é escalável em termos da sua dimensão e tipo de cultura, e todo o sistema computacional pode receber, armazenar e tratar dados de um grande número de lisímetros.

O servidor WEB com tecnologia PWA permite um acesso ao sistema de uma forma transversal, sendo possível aceder aos dados do lisímetro a partir de qualquer dispositivo móvel através de uma interface similar de aplicativo para *Smartphone*.

Os módulos lisímetro e câmara superaram a autonomia energética dimensionada, o que indica que o consumo de energia é inferior ao estimado, tendo cumprido todos os requisitos de projeto e superado todos os testes realizados, estando em modo de produção já há alguns meses.

5.2 Desenvolvimento Futuro

Um próximo estágio de desenvolvimento deste projeto será a colocação de diversos lisímetros em modo de produção com vista à recolha de um grande conjunto de dados: imagens e dados dos sensores, de monitorização da cultura e do solo para desenvolver e treinar um modelo de ML para inferir do estado de desenvolvimento das plantas e se existem problemas

relacionados com o desenvolvimento da cultura, nomeadamente deteção do aparecimento de doenças e pragas.

As pesquisas realizadas permitem inferir que quando necessitamos de uma estimativa precisa da ET, devemos recorrer a um lisímetro de pesagem dado ser preciso, robusto e fácil de construir, tal como o desenvolvido. Inclusive, os sistemas que recorrem a ML para estimar a ET usam lisímetros de pesagem para a sua calibração. Relativamente ao estudo de sistemas de aquisição e processamento de imagens na agricultura, os sistemas com maior precisão são baseados no método de redes neuronais convolucionais CNN. A evolução das redes neuronais, nomeadamente as DCNN, nos últimos cinco anos permitem criar sistemas mais precisos, e com menor necessidade de ajustes. No entanto, a bibliografia sobre o assunto mostra que é muito difícil criar um sistema universal para detecção de doenças em plantas. Para se obter bons níveis de precisão e rapidez devemos ajustar os sistemas para cada tipo de cultura e planta específica.

Também está planeado a integração do lisímetro num sistema geral de monitorização de recursos hídricos, que reúne informação de diversas fontes, incluindo estações meteorológicas, sistemas de monitorização da qualidade da água, sistemas de monitorização da evaporação da água, sistemas de monitorização do nível da água de lagos e rios, etc., o que permitirá ter um sistema completo para monitorizar a gestão da água. Está previsto que este sistema integrado também forneça dados a um modelo de dados baseado em sistemas de reconhecimento de padrões ML e/ou algoritmos de inteligência artificial que permitam relacionar todos estes dados a fim de prever eventos e prevenir ameaças de longo prazo associados com os recursos hídricos.

Outra vertente a ser desenvolvida consiste na implementação de mecanismos de segurança ao nível do acesso, transmissão, e armazenamento dos dados, nomeadamente, recorrendo a mecanismos de encriptação do tipo *blockchain*.

Bibliografia

[1] Água. [*Online*]. Disponível: https://unric.org/pt/agua/

[2] Carlos Almeida, João Miguel Santos, João C. Martins, e José Jasnau Caeiro, "Smart Lysimeter with Crop and Environment Monitoring: Enhanced with Pest and Crop Control," in *4th IFIP International Internet of Things (IoT) Conference*, Novembro 2021.

[3] "Lisímetro," Junho 2019, page Version ID: 55578825. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=Lis%C3%ADmetro&oldid=55578825

[4] G. Vitali, M. Francia, M. Golfarelli, e M. Canavari, "Crop management with the IoT: An interdisciplinary survey," *Agronomy*, vol. 11, n. 1, p. 181, 2021, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [*Online*]. Disponível: https://www.mdpi.com/2073-4395/11/1/181

[5] W. Zhu, Y. Tian, e S. Wang, "Design of non-weighing type desert plant lysimeter observation system based on PIC18," in *2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 3, 2013, pp. 42–44, ISSN: 2155-1472.

[6] "Microcontroller," Junho 2021, page Version ID: 1027668018. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=Microcontroller&oldid=1027668018

[7] G. Yang, C. Zhao, e Q. Xu, "Spatial-temporal analysis of field evapotranspiration based on complementary relationship model and IKONOS data," in *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*, 2013, pp. 2836–2839, ISSN: 2153-7003.

[8] J. A. Hernández-Salazar, D. Hernández-Rodríguez, R. A. Hernández-Cruz, J. C. Ramos-Fernández, M. A. Márquez-Vera, e F. R. Trejo-Macotela, "Estimation of

the evapotranspiration using ANFIS algorithm for agricultural production in greenhouse," in *2019 IEEE International Conference on Applied Science and Advanced Technology (iCASAT)*, 2019, pp. 1–5.

[9] L. Ávila Dávila, M. Soler-Méndez, C. F. Bautista-Capetillo, J. González-Trinidad, H. E. Júnez-Ferreira, C. O. Robles Rovelo, e J. M. Molina-Martínez, "A compact weighing lysimeter to estimate the water infiltration rate in agricultural soils," *Agronomy 2021*, vol. 11, n. 1, p. 180, 2021, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [*Online*]. Disponível: https://www.mdpi.com/2073-4395/11/1/180

[10] E. Mavridou, E. Vrochidou, G. A. Papakostas, T. Pachidis, e V. G. Kaburlasos, "Machine vision systems in precision agriculture for crop farming," *Journal of Imaging*, vol. 5, n. 12, p. 89, 2019.

[11] A. Agarwal e B. Triggs, "Hyperfeatures – multilevel local coding for visual recognition," in *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 30–43.

[12] D. Foley, "Gaussian Mixture Modelling (GMM)," Janeiro 2021. [*Online*]. Disponível: https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f

[13] "Mia Tutorial: Machine Learning Model Deployment on Mia," Junho 2021. [*Online*]. Disponível: https://omdena.com/blog/mia-tutorial/

[14] "PCA (Principal Component Analysis) Machine Learning Tutorial." [*Online*]. Disponível: https://www.projectpro.io/data-science-in-python-tutorial/principal-component-analysis-tutorial

[15] Wikipédia, "Artificial neural network," Website:, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Artificial_neural_network

[16] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," Julho 2018. [*Online*]. Disponível: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[17] G. G. Hungilo, G. Emmanuel, e A. W. R. Emanuel, "Image processing techniques for detecting and classification of plant disease," in *Proceedings of the 2019 International Conference on Intelligent Medicine and Image Processing - IMIP '19*. ACM Press, 2019.

[18] Wikipédia, "Convolutional neural network," Website:, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Convolutional_neural_network

[19] "Otsu's method," Outubro 2021, page Version ID: 1051475285. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=Otsu%27s_method&oldid=1051475285

[20] "K Means Clustering | K Means Clustering Algorithm in Python," Agosto 2019. [*Online*]. Disponível: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[21] "RGB color model," Dezembro 2021, page Version ID: 1061131512. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=RGB_color_model&oldid=1061131512

[22] "HSL and HSV," Dezembro 2021, page Version ID: 1059079850. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=HSL_and_HSV&oldid=1059079850

[23] B. Malinga, D. Raicu, e J. Furst, "Local vs. Global Histogram-Based Color Image Clustering."

[24] "Local Binary Pattern - an overview | ScienceDirect Topics." [*Online*]. Disponível: https://www.sciencedirect.com/topics/engineering/local-binary-pattern

[25] T. Mamdouh, "Image Retrieval: Color Coherence Vector." [*Online*]. Disponível: https://owlcation.com/stem/Image-Retrieval-Color-Coherence-Vector

[26] O. B. Sassi, "Improved Spatial Gray Level Dependence Matrices for Texture Analysis," *International Journal of Computer Science and Information Technology*, vol. 4, n. 6, pp. 209–219, Dezembro 2012. [*Online*]. Disponível: http://www.airccse.org/journal/jcsit/4612ijcsit15.pdf

[27] "Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples," Setembro 2017. [*Online*]. Disponível: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[28] G. says, "Decision Tree Algorithm, Explained," section: 2020 Jan Tutorials, Overviews. [*Online*]. Disponível: https://www.kdnuggets.com/decision-tree-algorithm-explained.html/

[29] "Random Forest Algorithms: A Complete Guide | Built In." [*Online*]. Disponível: https://builtin.com/data-science/random-forest-algorithm

[30] B. S. Kusumo, A. Heryana, O. Mahendra, e H. F. Pardede, "Machine learning-based for automatic detection of corn-plant diseases using image processing," in *Proc. Informatics and its Applications (IC3INA) 2018 Int. Conf. Computer, Control*, Novembro 2018, pp. 93–97.

[31] D. G. Lowe, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image," US patentus US6 711 293B1, 2000.

[32] C. Tomasi, "Histograms of oriented gradients," Computer Vision Sampler, pp. 1–6, 2012.

[33] E. Rublee, V. Rabaud, K. Konolige, e G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*. IEEE, 11 2011.

[34] J. D. S. Selda, R. M. R. Ellera, L. C. Cajayon, e N. B. Linsangan, "Plant identification by image processing of leaf veins," in *Proceedings of the International Conference on Imaging, Signal Processing and Communication - ICISPC 2017*. ACM Press, 2017.

[35] Wikipedia, "Sensitivity and specificity," Website, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Sensitivity_and_specificity

[36] E. Mwebaze e G. Owomugisha, "Machine learning for plant disease incidence and severity measurements from leaf images," in *Proc. 15th IEEE Int. Conf. Machine Learning and Applications (ICMLA)*, Dezembro 2016, pp. 158–163.

[37] "1.4. Support Vector Machines." [*Online*]. Disponível: https://scikit-learn/stable/modules/svm.html

[38] "KNN Algorithm | What is KNN Algorithm | How does KNN Function," Abril 2021. [*Online*]. Disponível: https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/

[39] J. Brownlee, "How to Develop an Extra Trees Ensemble with Python," Abril 2020. [*Online*]. Disponível: https://machinelearningmastery.com/extra-trees-ensemble-with-python/

[40] R. P. Foundation, "Raspberry pi - about us," Website, Janeiro 2020. [*Online*]. Disponível: https://www.raspberrypi.org/about/

[41] F. Jakjoud, A. Hatim, e A. Bouaaddi, "Deep learning application for plant diseases detection," in *Proceedings of the 4th International Conference on Big Data and Internet of Things*. ACM, 10 2019.

[42] A. Kamilaris e F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 4 2018.

[43] "Vgg16 – convolutional network for classification and detection," Website:, Janeiro 2020. [*Online*]. Disponível: https://neurohive.io/en/popular-networks/vgg16/

[44] "1.5. Stochastic Gradient Descent." [*Online*]. Disponível: https://scikit-learn/stable/modules/sgd.html

[45] R. Gylberth, "An Introduction to AdaGrad," Maio 2018. [*Online*]. Disponível: https://medium.com/konvergen/an-introduction-to-adagrad-f130ae871827

[46] "Root Mean Square Propagation - Andrea Perlato." [*Online*]. Disponível: https://www.andreaperlato.com/aipost/root-mean-square-propagation/

[47] neuralthreads, "Adadelta — Optimizer which was developed to eliminate the need for the learning rate," Novembro 2021. [*Online*]. Disponível: https://medium.com/@neuralthreads/adadelta-optimizer-which-was-developed-to-eliminate-the-need-for-the-learning-rate-93e8f295ab

[48] D. Majumdar, D. K. Kole, A. Chakraborty, e D. D. Majumder, "An integrated digital image analysis system for detection, recognition and diagnosis of disease in wheat leaves," in *Proceedings of the Third International Symposium on Women in Computing and Informatics - WCI '15*. ACM Press, 2015.

[49] A. Gupta, "Fuzzy C-Means Clustering (FCM) Algorithm in Machine Learning," Junho 2021. [*Online*]. Disponível: https://medium.com/geekculture/fuzzy-c-means-clustering-fcm-algorithm-in-machine-learning-c2e51e586fff

[50] J. Krause, G. Sugita, K. Baek, e L. Lim, "WTPlant(what's that plant?)," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval - ICMR '18*. ACM Press, 2018.

[51] S. A. Pearline, V. S. Kumar, e S. Harini, "A study on plant recognition using conventional image processing and deep learning approaches," *Journal of Intelligent & Fuzzy Systems*, vol. 36, n. 3, pp. 1997–2004, 3 2019.

[52] "UCI Machine Learning Repository: Folio Data Set." [*Online*]. Disponível: https://archive.ics.uci.edu/ml/datasets/Folio

[53] O. J. O. Söderkvist, "Computer vision classifcation of leaves from swedish trees," Website, Janeiro 2020. [*Online*]. Disponível: https://www.cvl.isy.liu.se/en/research/datasets/swedish-leaf/

[54] "Flavia, A Leaf Recognition Algorithm for Plant Classification using PNN." [*Online*]. Disponível: http://flavia.sourceforge.net/

[55] "Python.org." [*Online*]. Disponível: https://www.python.org/about/

[56] "Opencv (open source computer vision library)," Website, Janeiro 2020. [*Online*]. Disponível: https://opencv.org/about/

[57] "Keras: The python deep learning library," Website:, Janeiro 2020. [*Online*]. Disponível: https://keras.io/

[58] "Theano is a python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently," Website:, Janeiro 2020. [*Online*]. Disponível: http://deeplearning.net/software/theano/

[59] D. K. Sreekantha e Kavya A. M. , "Agricultural crop monitoring using iot - a study," in *Proc. 11th Int. Conf. Intelligent Systems and Control (ISCO)*, Janeiro 2017, pp. 134–139.

[60] Wikipédia, "Wireless sensor network - wsn," website, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Wireless_sensor_network

[61] zigbee alliance, "What is zigbee?" website, Janeiro 2020. [*Online*]. Disponível: https://zigbeealliance.org/solution/zigbee/

[62] "Radio Frequency Identification (RFID)." [*Online*]. Disponível: https://www.investopedia.com/terms/r/radio-frequency-identification-rfid.asp

[63] "General Packet Radio Service," Novembro 2021, page Version ID: 1055785300. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=General_Packet_Radio_Service&oldid=1055785300

[64] Wikipédia, "Wi-fi," Website, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/General_Packet_Radio_Service

[65] P. Tirelli, N. A. Borghese, F. Pedersini, G. Galassi, e R. Oberti, "Automatic monitoring of pest insects traps by zigbee-based wireless networking of image sensors," in *Proc. IEEE Int. Instrumentation and Measurement Technology Conf*, Maio 2011, pp. 1–5.

[66] T. Sakamoto, A. A. Gitelson, A. L. Nguy-Robertson, T. J. Arkebauer, B. D. Wardlow, A. E. Suyker, S. B. Verma, e M. Shibayama, "An alternative method using digital cameras for continuous monitoring of crop status," *Agricultural and Forest Meteorology*, vol. 154-155, pp. 113–126, 2012. [*Online*]. Disponível: https://www.sciencedirect.com/science/article/pii/S0168192311003133

[67] GISGeography, "What is NDVI (Normalized Difference Vegetation Index)?" Maio 2017. [*Online*]. Disponível: https://gisgeography.com/ndvi-normalized-difference-vegetation-index/

[68] Wikipédia, "Machine learning," Website, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Machine_learning

[69] ——, "Deep learning," Website, Janeiro 2020. [*Online*]. Disponível: https://en.wikipedia.org/wiki/Deep_learning

[70] "System-on-a-chip," Maio 2020, page Version ID: 58370125. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=System-on-a-chip&oldid=58370125

[71] C. Hernitscheck e F. Europe, "Designing for Ultra-Low-Power with MSP430," *MSP430 Advanced Tehnical Conference 2006*, p. 37, 2006. [*Online*]. Disponível: https://www.ti.com/lit/ml/slap124/slap124.pdf

[72] "Network Time Protocol," Abril 2020, page Version ID: 58071828. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=Network_Time_Protocol&oldid=58071828

[73] "Single-board computer," Abril 2021, page Version ID: 1018888075. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=Single-board_computer&oldid=1018888075

[74] "World Wide Web," Outubro 2021, page Version ID: 1048690604. [*Online*]. Disponível: https://en.wikipedia.org/w/index.php?title=World_Wide_Web&oldid=1048690604

[75] "CoAP — Constrained Application Protocol | Overview." [*Online*]. Disponível: https://coap.technology/

[76] "MQTT - The Standard for IoT Messaging." [*Online*]. Disponível: https://mqtt.org/

[77] "Base de Dados NoSQL - O Que É NoSQL? | Microsoft Azure." [*Online*]. Disponível: https://azure.microsoft.com/pt-pt/overview/nosql-database/

[78] "MSP430G2553 data sheet, product information and support | TI.com." [*Online*]. Disponível: https://www.ti.com/product/MSP430G2553

[79] Wikipédia, "Wi-fi," published: Website. [*Online*]. Disponível: https://en.wikipedia.org/wiki/General_Packet_Radio_Service

[80] "ESP-01." [*Online*]. Disponível: http://www.ai-thinker.com/pro_view-60.html

[81] "SHT30 Sensor de Temperatura e Humidade Digital i2c." [*Online*]. Disponível: https://www.botnroll.com/pt/temperatura/4067-sht30-sensor-de-temperatura-e-humidade-digital-i2c.html

[82] "TSL2561 Luminosity Sensor Hookup Guide - learn.sparkfun.com." [*Online*]. Disponível: https://learn.sparkfun.com/tutorials/tsl2561-luminosity-sensor-hookup-guide

[83] "One wire," Julho 2019, page Version ID: 55719179. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=One_wire&oldid=55719179

[84] "DS18B20 Waterproof Temperature Sensor." [*Online*]. Disponível: https://www.seeedstudio.com/DS18B20-Temperature-Sensor-Waterproof-Probe-p-4283.html

[85] "Grove - Capacitive Moisture Sensor (Corrosion-Resistant) - Seeed Wiki." [*Online*]. Disponível: https://wiki.seeedstudio.com/Grove-Capacitive_Moisture_Sensor-Corrosion-Resistant/

[86] "Load Sensor - 50kg (Generic) - SEN-10245 - SparkFun Electronics." [*Online*]. Disponível: https://www.sparkfun.com/products/10245

[87] "Avia Semiconductor (Xiamen) Ltd.-HX711." [*Online*]. Disponível: http://en.aviaic.com/detail/730856.html

[88] "SparkFun Load Cell Amplifier - HX711 - SEN-13879 - SparkFun Electronics." [*Online*]. Disponível: https://www.sparkfun.com/products/13879

[89] "Load Cell - 10kg, Straight Bar (TAL220) - SEN-13329 - SparkFun Electronics." [*Online*]. Disponível: https://www.sparkfun.com/products/13329

[90] "HS-422 Deluxe Standard Servo | HITEC RCD USA." [*Online*]. Disponível: https://hitecrcd.com/products/servos/analog/sport-2/hs-422/product

[91] "Small Solar Panel 80x100mm 1W." [*Online*]. Disponível: https://www.seeedstudio.com/1W-Solar-Panel-80X100.html

[92] "Placa de carregamento para bateria de lítio 1A (TP4056) - Micro-USB." [*Online*]. Disponível: https://mauser.pt/catalog/product_info.php?products_id=096-8206

[93] "BU-204: How do Lithium Batteries Work?" Setembro 2010. [*Online*]. Disponível: https://batteryuniversity.com/article/bu-204-how-do-lithium-batteries-work

[94] "Samsung SDI Small-Sized Li-ion Battery - Index | Samsung SDI." [*Online*]. Disponível: https://www.samsungsdi.com/lithium-ion-battery/index.html

[95] "MSP-EXP430G2ET Development kit | TI.com." [*Online*]. Disponível: https://www.ti.com/tool/MSP-EXP430G2ET

[96] "CCSTUDIO IDE, configuration, compiler or debugger | TI.com." [*Online*]. Disponível: https://www.ti.com/tool/CCSTUDIO

[97] "ESP8266 PROG | Joy-IT." [*Online*]. Disponível: https://joy-it.net/en/products/SBC-ESP8266-PROG

[98] "JSON." [*Online*]. Disponível: https://www.json.org/json-en.html

[99] R. P. Ltd, "Buy a Raspberry Pi Camera Module 2." [*Online*]. Disponível: https://www.raspberrypi.com/products/camera-module-v2/

[100] "Arquitetura ARM," Novembro 2020, page Version ID: 59822910. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=Arquitetura_ARM&oldid=59822910

[101] "MIPI Camera Serial Interface 2 (MIPI CSI-2)," Janeiro 2017. [*Online*]. Disponível: https://www.mipi.org/specifications/csi-2

[102] R. P. Ltd, "Buy a Raspberry Pi 3 Model B+." [*Online*]. Disponível: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

[103] "Lm2577 dc-dc voltage step-up (boost) module." [*Online*]. Disponível: http://www.velleman.eu/products/view/?id=435562&country=pt&lang=pt

[104] R. P. Ltd, "Raspberry Pi OS." [*Online*]. Disponível: https://www.raspberrypi.com/software/

[105] R. Light, "paho-mqtt: MQTT version 5.0/3.1.1 client class." [*Online*]. Disponível: http://eclipse.org/paho

[106] D. Jones, "picamera: A pure Python interface for the Raspberry Pi camera module." [*Online*]. Disponível: http://picamera.readthedocs.io/

[107] C. Liechti, "pyserial: Python Serial Port Extension." [*Online*]. Disponível: https://github.com/pyserial/pyserial

[108] "How to Setup Passwordless SSH Login," Junho 2018, section: post. [*Online*]. Disponível: https://linuxize.com/post/how-to-setup-passwordless-ssh-login/

[109] "Controle Numérico Computadorizado," Julho 2021, page Version ID: 61695318. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=Controle_Num%C3%A9rico_Computadorizado&oldid=61695318

[110] "Código G," Julho 2021, page Version ID: 61687211. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=C%C3%B3digo_G&oldid=61687211

[111] "FlatCAM: PCB Prototyping CAD/CAM." [*Online*]. Disponível: http://flatcam.org/

[112] "Virtual Machine." [*Online*]. Disponível: https://www.vmware.com/topics/glossary/content/virtual-machine

[113] "Ubuntu Server." [*Online*]. Disponível: https://ubuntu.com/download/server

[114] "What is a Container? | App Containerization | Docker." [*Online*]. Disponível: https://www.docker.com/resources/what-container

[115] "Eclipse Mosquitto," Janeiro 2018. [*Online*]. Disponível: https://mosquitto.org/

[116] "Node-RED." [*Online*]. Disponível: https://nodered.org/

[117] Node.js, "nodejs." [*Online*]. Disponível: https://nodejs.org/en/about/

[118] "mongoDB - The most popular database for modern apps." [*Online*]. Disponível: https://www.mongodb.com

[119] "atmoz/sftp Dockerfile." [*Online*]. Disponível: https://hub.docker.com/r/atmoz/sftp/dockerfile

[120] "TensorFlow." [*Online*]. Disponível: https://www.tensorflow.org/?hl=pt

[121] steve, "How MQTT Works -Beginners Guide," Junho 2018. [*Online*]. Disponível: http://www.steves-internet-guide.com/mqtt-works/

[122] "Learn JavaScript basics with our free JavaScript tutorials for programmers." [*Online*]. Disponível: https://www.javascript.com/about

[123] "Cluster: conceito e características." [*Online*]. Disponível: https://www.infowester.com/cluster.php

[124] "SSH Secure Shell home page, maintained by SSH protocol inventor Tatu Ylonen. SSH clients, servers, tutorials, how-tos." [*Online*]. Disponível: https://www.ssh.com/academy/ssh

[125] "RSA (sistema criptográfico)," Maio 2021, page Version ID: 61145079. [*Online*]. Disponível: https://pt.wikipedia.org/w/index.php?title=RSA_(sistema_criptogr%C3%A1fico)&oldid=61145079

[126] "KiCad EDA." [*Online*]. Disponível: https://www.kicad.org/

[127] "IP ratings | IEC." [*Online*]. Disponível: https://www.iec.ch/ip-ratings

[128] "What are Progressive Web Apps?" [*Online*]. Disponível: https://web.dev/what-are-pwas/

[129] "FileZilla - The free FTP solution." [*Online*]. Disponível: https://filezilla-project.org/

Apêndices

Apêndice I

Código desenvolvido

I.1 MCU Módulo lisímetro

```
/*  LISIMETRO - MIOT1819

 *
 *                    MSP430G2553
 *                  ---------------
 *              /|\|              XIN|-
 *               | |                 |
 *               --|RST          XOUT|-
 *                 |                 |
 *    <P ON>---|P1.0           P2.0|--<ONE WIRE>   STSx
 *                 |                 |
 *  <UART RX>-|P1.1           P2.1|--<I2C-SCL>  |
 *                 |                 |                | STA / SHA / SLA
 *  <UART TX>-|P1.2           P2.2|--<I2C-SDL>  |
 *                 |                 |
 *    <SHS 1>--|P1.3           P2.3|--<HX711-DATA>  |
 *                 |                 |                  | SPL / SPR
 *    <SHS 2>--|P1.4           P2.4|--<HX711-CLK>   |
 *                 |                 |
 *    <SHS 3>--|P1.5           P2.5|--<PWM> VR
 *                 |                 |
 *       <SVB>--|P1.6           P2.6|--<XTAL>
 *                 |                 |
 *        <SW>--|P1.7           P2.7|--<XTAL>
 *                 |                 |
 *                  -----------------
```

```
27  */
28
29
30 #include <msp430.h>
31 #include <stdint.h>
32 #include "CDC.h"
33 #include "delay.h"
34 #include "ds18b20.h"
35 #include "hx711.h"
36 #include "swi2c_master.h"
37 #include "TSL2561.h"
38 #include "SHT3X.H"
39 #include "SERVO.h"
40
41 const unsigned int pooling_time = 600;  // Pooling time in seconds
42
43 const unsigned int t_startup = 3;       // Time to start ESP in seconds
44 const unsigned int t_drain = 15;        // Drain Time in seconds
45
46 unsigned int seconds;                   // Seconds counter
47
48 unsigned int OPEN_VAL = 135;
49 unsigned int CLOSE_VAL = 45;
50
51
52 uint8_t addr1[8]={0x28,0xC0,0xB3,0xE2,0x08,0x00,0x00,0xEB};      //ID
       SHS1
53 uint8_t addr2[8]={0x28,0x10,0xFA,0xE3,0x08,0x00,0x00,0xBA};      //ID
       SHS2
54 uint8_t addr3[8]={0x28,0xCF,0x86,0xE2,0x08,0x00,0x00,0x99};      //ID
       SHS3
55
56
57 static const unsigned int ID = 1901;     // ID do Lisimetro
58 float STA = 0;          // Sensor Temperatura Ambiente [ÂºC]
59 float SHA = 0;          // Sensor Humidade Ambiente [%]
60 unsigned int SL_1 = 0;  // Sensor Luz visivel (V) + Infra Vermerlho (IR
       )
61 unsigned int SL_2 = 0;  // Sensor Luz Infra Vermerlho (IR)
62 float STS_1 = 0;        // Sensor Temperatura Solo 1 [ºC]
63 float STS_2 = 0;        // Sensor Temperatura Solo 2 [ºC]
64 float STS_3 = 0;        // Sensor Temperatura Solo 3 [ºC]
65 unsigned int SH_1 = 0;  // Sensor Humidity Solo 1 [%]
```

```
66 unsigned int SH_2 = 0;  // Sensor Humidity Solo 2 [%]
67 unsigned int SH_3 = 0;  // Sensor Humidade Solo 3 [%]
68 unsigned int SPL = 0;    // Sensor Peso Lisimetro [g]
69 unsigned int SPR_I = 0;   // Sensor Peso Residuos [g]
70 unsigned int SPR_F = 0;   // Sensor Peso Residuos [g]
71 unsigned int SPR = 0;    // Sensor Peso Residuos [g]
72 unsigned int SVB = 0;    // Sensor Voltagem Bateria [mV]
73
74 unsigned int arraySamples[4];    // ADC samples array
75
76
77 void GPIO_Init(void);
78 void BCM_Init(void);
79 void TIMER_Init(void);
80 void USCI_A0_Init(void);
81 void ADC10_Init(void);
82 void ADC10_StartSampling(void);
83 void sendData(void);
84
85
86 void main(void)
87 {
88     WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
89
90     seconds = pooling_time - 1;
91
92     GPIO_Init();
93     BCM_Init();
94     TIMER_Init();
95     USCI_A0_Init();
96     ADC10_Init();
97     ADC10_StartSampling();
98
99     _enable_interrupts();
100
101    while(1)
102    {
103        if (seconds == pooling_time) //
104        {
105            P1OUT |= BIT0;  // Power ON
106
107            seconds = 0;
108            while (seconds < t_startup);
```

```
109
110             SWI2C_initI2C();
111
112             //**HX711**********************************
113             HX711_init();
114             SPL = get_Weight(2);
115             SPR_I = get_Weight(1);
116
117             //*SHT3x**********************************
118             init_SHT3X();
119             if(read_SHT3X())
120             {
121                 STA = get_Temp();
122                 SHA = get_Hum();
123             }
124
125             //*TSL2561*********************************
126             init_TSL2561();
127             SL_2 = read_TSL2561_CH1();
128             SL_1 = read_TSL2561_CH0();
129
130             //**DB18B20********************************
131             STS_1 = get_temp_ad(addr1);
132             STS_2 = get_temp_ad(addr2);
133             STS_3 = get_temp_ad(addr3);
134
135             //*ADC*************************************
136             ADC10_StartSampling();
137             SH_1 = arraySamples[1];
138             SH_2 = arraySamples[2];
139             SH_3 = arraySamples[3];
140             SVB  = arraySamples[0];
141
142             init_SERVO();
143             set_SERVO(OPEN_VAL);
144
145             seconds = 0;
146             while (seconds < t_drain);
147
148             set_SERVO(CLOSE_VAL);
149
150             delay_ms(1000); // Time to close valve
151
```

```
152                 HX711_init();
153                 SPR_F = get_Weight(1);
154
155                 if( SPR_I > SPR_F ) SPR = SPR_I - SPR_F;
156                 else SPR = 0;
157
158                 delay_ms(500);
159
160                 sendData();
161
162                 delay_ms(5000); // Time to send data
163
164                 P1OUT &= ~BIT0; // Power OFF
165             }
166             __low_power_mode_3();
167         }
168 }
169
170
171
172
173 void  sendData(void)
174 {
175     p_ui(ID); ec(" ");
176     printfloat(STA); ec(" ");
177     printfloat(SHA); ec(" ");
178     p_ui(SL_1); ec(" ");
179     p_ui(SL_2); ec(" ");
180     printfloat(STS_1); ec(" ");
181     printfloat(STS_2); ec(" ");
182     printfloat(STS_3); ec(" ");
183     p_ui(SH_1); ec(" ");
184     p_ui(SH_2); ec(" ");
185     p_ui(SH_3); ec(" ");
186     p_ui(SVB); ec(" ");
187     p_ui(SPL); ec(" ");
188     p_ui(SPR); ec(" ");
189     LF_();
190     CR_();
191 }
192
193 void GPIO_Init()
194 {
```

```
195     P1OUT = (0x00);
196     // P1DIR = (0xFF);
197     P1DIR = ( BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0); // P1DIR
            = (0xF7); BIT7->IN
198     // USCI:
199     P1SEL  |= (BIT2 |BIT1); // UCA0TXD , UCA0RXD
200     P1SEL2 |= (BIT2 |BIT1); // UCA0TXD , UCA0RXD
201
202     P1REN |= BIT7;  // enable pull-up/pull-down resistor
203     P1OUT |= BIT7;  // select pull-up resistor
204
205     P1IES |= BIT7;  // select edge interrupt high->low transition
206     P1IFG = 0x00;   // clear P1IFG since writing to P1OUT, P1DIR, P2OUT
            , or P2DIR can result in setting the corresponding P1IFG
207                         // or P2IFG flags. (slau144j pg.331)
208     P1IE |= BIT7;   // enable interrupt on P1.7
209 }
210
211
212 void BCM_Init()
213 {
214     DCOCTL = CALDCO_8MHZ;    // DCO frequency select and Modulator
            selection:
215                              // are set with factory calibrated value
216     BCSCTL1 = XT2OFF         // XT2 off
217             | CALBC1_8MHZ    // Range select: factory value for 8MHz
218             | DIVA_0;        // ACLK Divider: 0->/1, 1_>/2, 2->/4, 3->/8
219
220     BCSCTL2 = SELM_0         // Select the MCLK source: 0->DCOCLK, 1->
            DCOCLK, 2-> XT2CLK when XT2, 3->LFXT1CLK or VLOCLK
221             | DIVM_0         // MCLK Divider: 0->/1, 1_>/2, 2->/4, 3->/8
222             | DIVS_0;        // SMCLK Divider: 0->/1, 1_>/2, 2->/4,
                    3->/8:
223
224     BCSCTL3 = LFXT1S_2;      // Low-frequency clock select and LFXT1
            range: 0->Crystal 32768Hz, 2->VLOCLK, 3->Ext CLK
225 }
226
227
228 void USCI_A0_Init(void)
229 {
230     // UART Mode: 9600 8N1
231     UCA0CTL1 = UCSSEL_2; // USCI clock source select: SMCLK
```

```
232     // From Table 36.4 in slau144j:
233     // BRCLK Frequency (Hz)     Baud Rate   UCBRx   UCBRSx  UCBRFx
234     // 8 000 000                9600        833     2       0
235     UCA0BR0 = 0x41;      // UCBRx = 833 = 0x341
236     UCA0BR1 = 0x03;
237     UCA0MCTL = UCBRS_2;
238     // USCI reset released for operation:
239     UCA0CTL1 &= ~ UCSWRST;
240     IE2 |= UCA0RXIE;     // USCI_A0 receive interrupt enable
241 }
242
243 #pragma vector = USCIAB0RX_VECTOR
244 __interrupt void USCIA0RX_ISR(void)
245 {
246     while(!(IFG2 & UCA0TXIFG));
247     UCA0TXBUF = UCA0RXBUF;
248 }
249
250
251 void TIMER_Init()
252 {
253     // Timer A0 settings for 1s counter
254     TA0CCTL0 = CCIE;     // Capture/Compare Interrupt Enable
255     TA0CTL = TASSEL_1    // Clock source = 0->TACLK, 1->ACLK, 2->SMCLK,
            3->INCLK
256             | MC_1       // Mode control = 1->UP / 2->CONTINUOS / 3->UP/
                DOWN
257             | ID_0;      // Divider = 0->/1, 1->/2, 2->/4, 3->/8
258     TA0CCR0 = 10350;     // Compare value: 32768 for 32768Hz crystal,
            10000 for 10KHz VLOCLK (10350 after 1s calibration)
259     // Timer A1 settings for PWM servo control
260     TA1CTL = TASSEL_2    // Clock source = 0->TACLK, 1->ACLK, 2->SMCLK,
            3->INCLK
261             | TACLR      // Timer_A1 clear
262             | MC_1       // Mode control = 1->UP / 2->CONTINUOS / 3->UP/
                DOWN
263             | ID_3;      // Divider = 0->/1, 1->/2, 2->/4, 3->/8
264     TA1CCR0 = 20000;      // Capture and Compare: 20000=> F=50Hz, T=20ms
265     TA1CCTL2 |= OUTMOD_7;    // Output Mode: 7-> Reset/Set
266 }
267
268
269 #pragma vector = TIMER0_A0_VECTOR
```

```
270 __interrupt void TIMER0_A0_ISR(void)
271 {
272     ++ seconds;
273     __low_power_mode_off_on_exit();
274 }
275
276
277 #pragma vector = PORT1_VECTOR
278 __interrupt void Port1_ISR(void)
279 {
280     if(P1IFG & BIT7) // switch pressed
281     {
282         seconds = pooling_time - 1;
283     }
284     P1IFG &= ~BIT7;  // clear interrupt flag of P1.3
285 }
286
287
288 void ADC10_Init(void)
289 {
290     // ADC10 Control Register 0:
291     ADC10CTL0 = SREF_1       // Select reference: VR+ = VREF+ and VR- =
        VSS
292               | ADC10SHT_0   // ADC10 sample-and-hold time: 4 ï¿½
                    ADC10CLKs
293               | ADC10SR      // ADC10 sampling rate: Reference buffer
                    supports up
294                              // to ~50 ksps
295               | REFBURST     // Reference buffer on only during
296                              // sample-and-conversion
297               | MSC          // Multiple sample and conversion
298               | REF2_5V      // Reference-generator voltage 2.5V
299               | REFON        // Reference generator on: V_REF+ takes 30
                    us to
300                              // settle (slau144j pg. 38)
301               | ADC10ON      // ADC10 on
302               | ADC10IE;      // ADC10 interrupt enable
303
304     // ADC10 Control Register 1:
305     ADC10CTL1 = INCH_6       //  Input channel selection: A6-A5-A4-A3
306               | SHS_0        //  Sample-and hold source select:
307               | ADC10DIV_3   // ADC10 clock divider: /4
308               | ADC10SSEL_0  // ADC10 clock source select: ADC10OSC - 0
```

```
309             | CONSEQ_1;       // Conversion sequence mode select:
310                               //     Sequence-of-channels (A7 - A6 - A5 -
                                      A4)
311
312
313     // Analog (Input) Enable Control Register 0:
314     ADC10AE0 |= BIT3
315             | BIT4          // ADC10 analog enable: A7 on P1.7
316             | BIT5            // ADC10 analog enable: A6 on P1.6
317             | BIT6;           //
318     // ADC10 data transfer control register 1:
319     ADC10DTC1 = 4;          // Define the number of transfers: 4 values
320 }
321
322
323 void ADC10_StartSampling(void)
324 {
325 //     while (ADC10CTL1 & ADC10BUSY);  // Wait if ADC10 core is active
326     // ADC10 data transfer start address:
327     ADC10SA = (unsigned int) arraySamples; // ADC10 start address
328     ADC10CTL0 |= ENC        // Enable conversion
329                | ADC10SC;
330     delay_ms(1);
331 }
332
333 #pragma vector = ADC10_VECTOR
334 __interrupt void ADC10_ISR(void)
335 {
336    ADC10CTL0 &= ~ENC;
337    __low_power_mode_off_on_exit();
338 }
```

Listagem I.1: Código do programa principal main.c (MCU Módulo lisímetro).

```
1 /*
2  * TSL2561.h
3  *
4  *  Created on: 07/10/2019
5  *      Author: Carlos Almeida
6  */
7
8 #ifndef TSL2561_H_
9 #define TSL2561_H_
```

```
10
11 #include <msp430.h>
12 #include <stdbool.h>
13 #include <stdint.h>
14
15 void init_TSL2561();
16 unsigned int read_TSL2561_CH0();
17 unsigned int read_TSL2561_CH1();
18
19 SWI2C_I2CTransaction TSL2561;
20
21 uint8_t WR_Buffer[2];
22 uint8_t RD_Buffer[6];
23
24 void init_TSL2561()
25 {
26     TSL2561.address = 0x39;
27     WR_Buffer[0] = 0x80;    // CMD
28     WR_Buffer[1] = 0x03;     // DATA
29     TSL2561.writeBuffer = WR_Buffer;
30     TSL2561.numWriteBytes = 2;
31     SWI2C_performI2CTransaction(&TSL2561);
32     delay_ms(150);
33     SWI2C_performI2CTransaction(&TSL2561);
34     delay_ms(150);
35
36     TSL2561.address = 0x39;
37     WR_Buffer[0] = 0x81;     // CMD - Set up Timing Register
38     WR_Buffer[1] = 0x02;     // DATA set gain 1x integration time 402ms
           (pág. 20)
39     TSL2561.writeBuffer = WR_Buffer;
40     TSL2561.numWriteBytes = 2;
41     SWI2C_performI2CTransaction(&TSL2561);
42     delay_ms(150);
43     SWI2C_performI2CTransaction(&TSL2561);
44     delay_ms(150);
45 }
46
47 unsigned int read_TSL2561_CH0()
48 {
49     TSL2561.address = 0x39;
50     WR_Buffer[0] = 0xAC;     // CMD Read Word
51     TSL2561.writeBuffer = WR_Buffer;
```

```
52      TSL2561.numWriteBytes = 1;
53      TSL2561.numReadBytes = 2;
54      TSL2561.readBuffer = RD_Buffer;
55      SWI2C_performI2CTransaction(&TSL2561);
56      delay_ms(150);
57      SWI2C_performI2CTransaction(&TSL2561);
58      delay_ms(150);
59      return (256*RD_Buffer[1]+RD_Buffer[0]);
60  }
61
62
63  unsigned int read_TSL2561_CH1()
64  {
65      TSL2561.address = 0x39;
66      WR_Buffer[0] = 0xAE;      // CMD Read Word
67      TSL2561.writeBuffer = WR_Buffer;
68      TSL2561.numWriteBytes = 1;
69      TSL2561.numReadBytes = 2;
70      TSL2561.readBuffer = RD_Buffer;
71      SWI2C_performI2CTransaction(&TSL2561);
72      delay_ms(150);
73      SWI2C_performI2CTransaction(&TSL2561);
74      delay_ms(150);
75      return (256*RD_Buffer[1]+RD_Buffer[0]);
76  }
77
78  #endif /* TSL2561_H_ */
```

Listagem I.2: Código CDC.c (MCU Módulo lisímetro).

```
1  #ifndef DELAY_H_
2  #define DELAY_H_
3
4  void delay_ms(unsigned int ms);
5  void delay_us(unsigned int us);
6
7
8  void delay_us(unsigned int us)
9  {
10     while (us)
11     {
12         __delay_cycles(8); // 1 for 1 Mhz set 16 for 16 MHz
13         us--;
```

```
14      }
15  }
16
17  void delay_ms(unsigned int ms)
18  {
19      while (ms)
20      {
21          __delay_cycles(8000);// 1000 for 1MHz and 16000 for 16MHz
22          ms--;
23      }
24  }
25
26  #endif /* #endif  DELAY_H_ */
```

Listagem I.3: Código delay.h (MCU Módulo lisímetro).

```
1  /*****************************************************************************
2  // https://www.maximintegrated.com/en/app-notes/index.mvp/id/187
3
4  #include <msp430g2553.h>
5  #include <stdio.h>
6  #include <stdint.h>
7
8  #include "ds18b20.h"
9
10
11
12 float celsius [4];
13 uint8_t data[12];
14 uint8_t i;
15
16 //****************************************************************************
17 void ow_portsetup() {
18      OWPORTDIR |= OWPORTPIN;
19      OWPORTOUT |= OWPORTPIN;
20      OWPORTREN |= OWPORTPIN;
21 }
22
23
24 //****************************************************************************
25 void select(uint8_t id[8]){
26      uint8_t i;
27      ow_write_byte(0x55);
```

```
28      for (i=0; i<8; i++) ow_write_byte(id[i]);
29  }
30
31  //#########################################################################
32
33  /// @return: 0 if ok
34  int ow_reset()
35  {
36   OW_LO
37    delay_us(480); // 480us minimum
38   OW_RLS
39     delay_us(40); // slave waits 15-60us
40     if (OWPORTIN & OWPORTPIN) return 1; // line should be pulled down by
           slave
41     delay_us(300); // slave TX presence pulse 60-240us
42     if (!(OWPORTIN & OWPORTPIN)) return 2; // line should be "released"
           by slave
43     return 0;
44  }
45
46  //#########################################################################
47  void ow_write_bit(int bit)
48  {
49  //  delay_us(1); // recovery, min 1us
50    OW_HI
51     if (bit) {
52   OW_LO
53        delay_us(5); // max 15us
54   OW_RLS // input
55        delay_us(56);
56     }
57     else {
58      OW_LO
59       delay_us(60); // min 60us
60      OW_RLS // input
61       delay_us(1);
62     }
63    }
64
65  //#########################################################################
66  int ow_read_bit()
67  {
68     int bit=0;
```

```
69 //  delay_us(1); // recovery, min 1us
70   OW_LO
71   delay_us(5); // hold min 1us
72   OW_RLS
73   delay_us(10); // 15us window
74   if (OWPORTIN & OWPORTPIN) {
75    bit = 1;
76   }
77   delay_us(46); // rest of the read slot
78   return bit;
79 }
80
81 //#########################################################################
82 void ow_write_byte(uint8_t byte)
83 {
84   int i;
85   for(i = 0; i < 8; i++)
86   {
87     ow_write_bit(byte & 1);
88     byte >>= 1;
89   }
90 }
91
92 //#########################################################################
93 uint8_t ow_read_byte()
94 {
95  unsigned int i;
96   uint8_t byte = 0;
97   for(i = 0; i < 8; i++)
98   {
99     byte >>= 1;
100    if (ow_read_bit()) byte |= 0x80;
101  }
102  return byte;
103 }
104
105 //#########################################################################
106
107 float get_temp()
108 {
109     uint8_t i;
110     uint8_t type_s=0;
111 //    uint8_t data[12];
```

```
112
113     ow_reset();
114     ow_write_byte(DS1820_SKIP_ROM); // skip ROM command
115     ow_write_byte(DS1820_CONVERT_T); // convert T command
116     OW_HI
117     delay_ms(75); // at least 750 ms for the default 12-bit resolution
118     ow_reset();
119     ow_write_byte(DS1820_SKIP_ROM); // skip ROM command
120     ow_write_byte(DS1820_READ_SCRATCHPAD); // read scratchpad command
121
122     for (i = 0; i < 9; i++) {
123         data[i] = ow_read_byte();
124     }
125
126     int16_t raw = (data[1] << 8) | data[0];
127     if (type_s) {
128         raw = raw << 3; // 9 bit resolution default
129         if (data[7] == 0x10) {
130             // "count remain" gives full 12 bit resolution
131             raw = (raw & 0xFFF0) + 12 - data[6];
132         }
133     } else {
134         uint8_t cfg = (data[4] & 0x60);
135         // at lower res, the low bits are undefined, so let's zero them
136         if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
137         else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
138         else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
139         //// default is 12 bit resolution, 750 ms conversion time
140     }
141     return (float)raw / 16.0;
142 }
143
144 //#########################################################################
145
146 float get_temp_ad(uint8_t addr[8])
147 {
148
149     ow_reset();
150     select(addr);
151     ow_write_byte(DS1820_CONVERT_T); // convert T command
152     OW_HI
153     delay_ms(750); // at least 750 ms for the default 12-bit resolution
154     ow_reset();
```

```
155     select(addr);
156     ow_write_byte(DS1820_READ_SCRATCHPAD); // read scratchpad command
157
158     for (i = 0; i < 9; i++) {
159         data[i] = ow_read_byte();
160     }
161
162     int16_t raw = (data[1] << 8) | data[0];
163     uint8_t cfg = (data[4] & 0x60);
164     // at lower res, the low bits are undefined, so let's zero them
165     if (cfg == 0x00) raw = raw & ~7;  // 9 bit resolution, 93.75 ms
166     else if (cfg == 0x20) raw = raw & ~3; // 10 bit res, 187.5 ms
167     else if (cfg == 0x40) raw = raw & ~1; // 11 bit res, 375 ms
168     //// default is 12 bit resolution, 750 ms conversion time
169   //  }
170     return (float)raw / 16.0;
171
172 }
```

Listagem I.4: Código ds18b20.c (MCU Módulo lisímetro).

```
1 //////////////////////////////////////////////////////////////////////////////
2 // https://www.maximintegrated.com/en/app-notes/index.mvp/id/187
3
4 #ifndef DS18B20_H_
5 #define DS18B20_H_
6 #include <stdint.h>
7
8
9 // Port and pins definition:
10 #define OWPORTDIR P2DIR
11 #define OWPORTOUT P2OUT
12 #define OWPORTIN P2IN
13 #define OWPORTREN P2REN
14 #define OWPORTPIN BIT0
15 #define OW_LO { OWPORTDIR |= OWPORTPIN; OWPORTREN &= ~OWPORTPIN;
       OWPORTOUT &= ~OWPORTPIN; }
16 #define OW_HI { OWPORTDIR |= OWPORTPIN; OWPORTREN &= ~OWPORTPIN;
       OWPORTOUT |= OWPORTPIN; }
17 #define OW_RLS { OWPORTDIR &= ~OWPORTPIN; OWPORTREN |= OWPORTPIN;
       OWPORTOUT |= OWPORTPIN; }
18
19 // list of commands DS18B20:
```

```
20
21 #define DS1820_WRITE_SCRATCHPAD   0x4E
22 #define DS1820_READ_SCRATCHPAD    0xBE
23 #define DS1820_COPY_SCRATCHPAD    0x48
24 #define DS1820_READ_EEPROM        0xB8
25 #define DS1820_READ_PWRSUPPLY     0xB4
26 #define DS1820_SEARCHROM          0xF0
27 #define DS1820_SKIP_ROM           0xCC
28 #define DS1820_READROM            0x33
29 #define DS1820_MATCHROM           0x55
30 #define DS1820_ALARMSEARCH        0xEC
31 #define DS1820_CONVERT_T          0x44
32
33 // Function definitions:
34 void ow_portsetup();
35 int ow_reset();
36 void ow_write_bit(int bit);
37 int ow_read_bit();
38 void ow_write_byte(uint8_t byte);
39 uint8_t ow_read_byte();
40 void onewire_line_low();
41 void onewire_line_high();
42 void onewire_line_release();
43 float get_temp_ad(uint8_t addr[8]);
44 float get_temp();
45
46 #endif /* ONEWIRE_H_ */
```

Listagem I.5: Código ds18b20.h (MCU Módulo lisímetro).

```
1 /*
2  * HX711 driver
3  *
4  *  Created on: 19/05/2019
5  *      Author: Carlos Almeida
6  *      Develop by Lysimeter MIOT1819
7  */
8
9
10 #ifndef HX711_H_
11 #define HX711_H_
12
13 #include "delay.h"
```

```
14 #include <stdint.h>
15
16 #define ADSK BIT4    // Clock
17 #define ADDO BIT3    // Data
18
19 // Functions
20 //
    ********************************************************************************

21 unsigned long read_HX711(void);          // Read ADC Converter
22 void calibration(uint8_t ch);             // Calibration channel
23 int get_Weight(uint8_t ch);               // Get weight channel in grams
24 void set_CH(unsigned char);               // Set channel - 0->Ch A, 1->Ch
       B, 2->Ch A
25 void power_off();
26 void HX711_init();
27
28 //unsigned long count;
29 // Constants
30 //
    ********************************************************************************

31 const float F_CAL[3] = {1.15, 8.88, 0.575};          // Calibration
    factor - Ch A, Ch B, Ch A
32
33 //Variables
34 //
    ********************************************************************************

35 uint8_t clk_pulses;
36 long buffer = 0;
37 long weight = 0;
38
39 //#pragma LOCATION(calibration_value, 0x1000);         // Save in Flash
    Memory Address 0x1000
40 unsigned long calibration_value[3] = {8460000, 8543011, 8555100}; //
    Calibration Value - Ch A, Ch B, Ch A
41
42
43 //
    ********************************************************************************

44 int get_Weight(uint8_t ch)
```

```
45 {
46     set_CH(ch);
47     read_HX711();
48     buffer = read_HX711();
49     weight = (buffer - calibration_value[ch-1]) /100;
50     weight = (int)((float)weight*F_CAL[ch-1]);  // calibration factor /
           need adjust to compare
51
52     return weight;
53 }
54
55 //
    ************************************************************************

56 void calibration(uint8_t ch)
57 {
58     set_CH(ch);
59     read_HX711();
60     buffer = read_HX711();
61     calibration_value[ch-1] = buffer;
62 }
63
64
65 //
    ************************************************************************

66 void power_off()
67 {
68     P2OUT |= ADSK;
69     delay_us(60);
70 }
71
72 //
    ************************************************************************

73 void set_CH(uint8_t ch)
74 {
75     switch (ch){
76
77         case 1:
78             clk_pulses = 25;     // Channel A - gain of 128
79             break;
80         case 2:
```

```
81              clk_pulses = 26;     // Channel B - gain of 32
82              break;
83          case 3:
84              clk_pulses = 27;     // Channel A - gain of 64
85              break;
86
87          default: clk_pulses = 25;
88      }
89  }
90
91
92
93  //
        ***************************************************************************
94  unsigned long read_HX711 (){
95      unsigned long count;
96      unsigned char i;
97      P2OUT |= ADDO;
98  //      delay_us(1);
99      P2OUT &= ~ADSK;
100     delay_us(1);
101     count = 0;
102     while(P2IN == ADDO);
103     for(i=1; i<clk_pulses; i++){
104         P2OUT |= ADSK;
105         delay_us(10);
106         count = count << 1;
107         P2OUT &= ~ADSK;
108         delay_us(10);
109         if(P2IN == ADDO) count ++;
110     }
111     P2OUT |= ADSK;
112     delay_us(10);
113     count = count^0x800000;
114     P2OUT &= ~ADSK;
115     delay_us(10);
116
117     return (count);
118
119 }
120
121 void HX711_init()   //  &= 0    / |= 1
```

```
122 {
123     P2OUT = 0x00;
124     P2DIR = 0xFF;
125     P2DIR &= ~ADDO;
126     P2SEL = 0x00;
127 }
128
129 #endif
```

Listagem I.6: Código hx711.h (MCU Módulo lisímetro).

```
1  /*
2   * servo.h
3   *
4   *  Created on: 01/12/2019
5   *      Author: Carlos Almeida
6   */
7
8  #ifndef SERVO_H_
9  #define SERVO_H_
10
11
12 #include <msp430g2553.h>
13 #include <stdio.h>
14 #include <stdint.h>
15
16
17 void init_SERVO();
18
19 void set_SERVO(unsigned int ang );
20
21
22
23 void init_SERVO()
24 {
25
26     P2SEL |= BIT5;
27 //    P2SEL2 &= ~BIT5;
28 }
29
30 void set_SERVO(unsigned int ang )
31 {
32     TA1CCR2 = 530 + ang*10; // TA1CCR2: 500 => T=0.5ms, 2500 => T=2.5ms
```

```
33 }
34
35
36 #endif /* SERVO_H_ */
```

Listagem I.7: Código servo.h (MCU Módulo lisímetro).

```
1  *
2   * sht3x.h
3   *
4   *  Created on: 03/11/2019
5   *      Author: Carlos Almeida
6   */
7
8  #ifndef SHT3X_H_
9  #define SHT3X_H_
10
11 #include <msp430.h>
12 #include <stdbool.h>
13 #include <stdint.h>
14
15 void init_SHT3X();
16 bool read_SHT3X();
17 float get_Temp();
18 float get_Hum();
19
20 SWI2C_I2CTransaction SHT3X;
21 uint8_t WR_Buffer[2];
22 uint8_t RD_Buffer[6];
23
24 void init_SHT3X()
25 {
26     SHT3X.address = 0x44;
27     WR_Buffer[0] = 0x30;      // CMD
28     WR_Buffer[1] = 0xA2;      // DATA
29     SHT3X.writeBuffer = WR_Buffer;
30     SHT3X.numWriteBytes = 2;
31     SWI2C_performI2CTransaction(&SHT3X);
32     delay_ms(5);
33     SWI2C_performI2CTransaction(&SHT3X);
34     delay_ms(5);
35 }
36
```

```
37
38 bool read_SHT3X()
39 {
40     SHT3X.address = 0x44;
41     WR_Buffer[0] = 0x24;     // CMD Read Word
42     WR_Buffer[1] = 0x0b;
43     SHT3X.writeBuffer = WR_Buffer;
44     SHT3X.numWriteBytes = 2;
45     SHT3X.numReadBytes = 6;
46     SHT3X.readBuffer = RD_Buffer;
47     SWI2C_performI2CTransaction(&SHT3X);
48     delay_ms(5);
49     SWI2C_performI2CTransaction(&SHT3X);
50     delay_ms(5);
51     // check CRC for both RH and T
52     if (crc8(&RD_Buffer[0], 2) != RD_Buffer[2] || crc8(&RD_Buffer[3],
           2) != RD_Buffer[5]) {
53       return false;
54     }
55     return true;
56 }
57
58 float get_Temp()
59 {
60     uint32_t stemp = ((uint32_t)RD_Buffer[0] << 8) | RD_Buffer[1];
61     stemp = ((4375 * stemp) >> 14) - 4500;
62     return  (float) stemp / 100.0;
63
64 }
65
66 float get_Temp1()
67 {
68     uint16_t stemp = ((uint16_t)RD_Buffer[0] << 8) | RD_Buffer[1];
69     float temp = -45 + 175 * ((float)stemp/65535);
70     return  temp;
71 }
72
73
74
75
76 float get_Hum()
77 {
78     uint32_t shum = ((uint32_t)RD_Buffer[3] << 8) | RD_Buffer[4];
```

```
79      shum = (625 * shum) >> 12;
80      return  (float) shum / 100.0f;
81    }
82
83
84  crc8(const uint8_t *data)
85  {
86    // adapted from SHT21 sample code from
87    // http://www.sensirion.com/en/products/humidity-temperature/download
          -center/
88
89    uint8_t crc = 0xff;
90    uint8_t byteCtr;
91    uint8_t bit;
92    for (byteCtr = 0; byteCtr < 2; ++byteCtr) {
93      crc ^= data[byteCtr];
94      for (bit = 8; bit > 0; --bit) {
95        if (crc & 0x80) {
96          crc = (crc << 1) ^ 0x31;
97        } else {
98          crc = (crc << 1);
99        }
100     }
101   }
102   return crc;
103 }
104
105 #endif /* SHT3X_H_ */
```

Listagem I.8: Código sht3x.h (MCU Módulo lisímetro).

```
1  /*
2   * TSL2561.h
3   *
4   *  Created on: 07/10/2019
5   *      Author: Carlos Almeida
6   */
7
8  #ifndef TSL2561_H_
9  #define TSL2561_H_
10
11 #include <msp430.h>
12 #include <stdbool.h>
```

```
13 #include <stdint.h>
14
15 void init_TSL2561();
16 unsigned int read_TSL2561_CH0();
17 unsigned int read_TSL2561_CH1();
18
19 SWI2C_I2CTransaction TSL2561;
20
21 uint8_t WR_Buffer[2];
22 uint8_t RD_Buffer[6];
23
24 void init_TSL2561()
25 {
26     TSL2561.address = 0x39;
27     WR_Buffer[0] = 0x80;    // CMD
28     WR_Buffer[1] = 0x03;     // DATA
29     TSL2561.writeBuffer = WR_Buffer;
30     TSL2561.numWriteBytes = 2;
31     SWI2C_performI2CTransaction(&TSL2561);
32     delay_ms(150);
33     SWI2C_performI2CTransaction(&TSL2561);
34     delay_ms(150);
35
36     TSL2561.address = 0x39;
37     WR_Buffer[0] = 0x81;     // CMD - Set up Timing Register
38     WR_Buffer[1] = 0x02;     // DATA set gain 1x integration time 402ms
           (pág. 20)
39     TSL2561.writeBuffer = WR_Buffer;
40     TSL2561.numWriteBytes = 2;
41     SWI2C_performI2CTransaction(&TSL2561);
42     delay_ms(150);
43     SWI2C_performI2CTransaction(&TSL2561);
44     delay_ms(150);
45 }
46
47 unsigned int read_TSL2561_CH0()
48 {
49     TSL2561.address = 0x39;
50     WR_Buffer[0] = 0xAC;     // CMD Read Word
51     TSL2561.writeBuffer = WR_Buffer;
52     TSL2561.numWriteBytes = 1;
53     TSL2561.numReadBytes = 2;
54     TSL2561.readBuffer = RD_Buffer;
```

```
55      SWI2C_performI2CTransaction(&TSL2561);
56      delay_ms(150);
57      SWI2C_performI2CTransaction(&TSL2561);
58      delay_ms(150);
59      return (256*RD_Buffer[1]+RD_Buffer[0]);
60  }
61
62
63  unsigned int read_TSL2561_CH1()
64  {
65      TSL2561.address = 0x39;
66      WR_Buffer[0] = 0xAE;      // CMD Read Word
67      TSL2561.writeBuffer = WR_Buffer;
68      TSL2561.numWriteBytes = 1;
69      TSL2561.numReadBytes = 2;
70      TSL2561.readBuffer = RD_Buffer;
71      SWI2C_performI2CTransaction(&TSL2561);
72      delay_ms(150);
73      SWI2C_performI2CTransaction(&TSL2561);
74      delay_ms(150);
75      return (256*RD_Buffer[1]+RD_Buffer[0]);
76  }
77
78  #endif /* TSL2561_H_ */
```

Listagem I.9: Código TSL2561.h (MCU Módulo lisímetro).

```
1  /* --COPYRIGHT--,BSD
2   * Copyright (c) 2016, Texas Instruments Incorporated
3   * All rights reserved.
4   *
5   * Redistribution and use in source and binary forms, with or without
6   * modification, are permitted provided that the following conditions
7   * are met:
8   *
9   * *  Redistributions of source code must retain the above copyright
10  *    notice, this list of conditions and the following disclaimer.
11  *
12  * *  Redistributions in binary form must reproduce the above copyright
13  *    notice, this list of conditions and the following disclaimer in
        the
14  *    documentation and/or other materials provided with the
        distribution.
```

```
15  *
16  * *  Neither the name of Texas Instruments Incorporated nor the names
        of
17  *    its contributors may be used to endorse or promote products
        derived
18  *    from this software without specific prior written permission.
19  *
20  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
        "AS IS"
21  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO
        ,
22  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
        PARTICULAR
23  * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
24  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL
        ,
25  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
26  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
        PROFITS;
27  * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
        LIABILITY,
28  * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
         OR
29  * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
30  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
31  * --/COPYRIGHT--*/
32 //
     ***************************************************************************


33
34 #include "swi2c_master.h"
35
36 //#include "delay.h"
37
38 int dl = 5;
39
40 /* Static Functions */
41 static bool SWI2C_readData(uint8_t addr,  uint8_t *inputArray,
       uint_fast16_t size);
42 static bool SWI2C_writeData(uint8_t addr, uint8_t *outputArray,
       uint_fast16_t size, bool sendStop);
43
44 void SWI2C_initI2C(void)
```

```
45 {
46     /* Using the direction pin to control the output. When set as an
           input, the
47         hardware pull-ups will take over and cause the pin to go high.
               When
48         set as an output, the MSP430 will drive the lines low */
49     SWI2C_PxOUT &= ~(SWI2C_SCL | SWI2C_SDA);
50     SWI2C_SCL_HIGH;
51     SWI2C_SDA_HIGH;
52
53 //    PM5CTL0 &= ~LOCKLPM5;                    // Disable the GPIO power
       -on default high-impedance mode
54                                                // to activate previously
                                                      configured port
                                                      settings
55
56     /* Timer is initialized to run off SMCLK(8MHz) with frequency 200
           KHz */
57 //    TB0CCR0 = SWI2C_TIMER_PERIOD;
58 }
59
60 bool SWI2C_performI2CTransaction(SWI2C_I2CTransaction *i2cTransaction)
61 {
62     if(i2cTransaction->numWriteBytes > 0)
63     {
64         /* Only skipping the stop if we have a repeated start to send
               */
65         if(i2cTransaction->repeatedStart && i2cTransaction->
               numReadBytes > 0)
66         {
67             if (!SWI2C_writeData(i2cTransaction->address,
68                     i2cTransaction->writeBuffer, i2cTransaction->
                         numWriteBytes,
69                     false))
70             {
71                 return false;
72             }
73         }
74         else
75         {
76             if (!SWI2C_writeData(i2cTransaction->address,
77                     i2cTransaction->writeBuffer, i2cTransaction->
                         numWriteBytes,
```

```
78                      true))
79              {
80                  return false;
81              }
82          }
83      }
84
85      /* Next doing the read */
86      if (i2cTransaction->numReadBytes > 0)
87      {
88          if (!SWI2C_readData(i2cTransaction->address, i2cTransaction->
                readBuffer,
89                  i2cTransaction->numReadBytes))
90          {
91              return false;
92          }
93      }
94
95      return true;
96  }
97
98  static bool SWI2C_writeData(uint8_t addr, uint8_t *outputArray,
        uint_fast16_t size, bool sendStop)
99  {
100     uint_fast8_t bits, temp;
101     uint16_t ii = 0;
102
103     /* Starting the timer */
104 //    TB0CTL = TBSSEL_2 + MC_1 + TBCLR;
105
106     /* Sending the START */
107     SWI2C_SDA_LOW;
108     __no_operation();
109     SWI2C_SCL_LOW;
110 //    TIMER_ITERATION();
111     delay_us(dl);
112
113     /* Next doing the control byte */
114     temp = (addr << 1);
115     bits = 8;
116
117     /* Loop until all bits of the address byte are sent out */
118     do
```

```
119     {
120         /* Deciding if we want to send a high or low out of the line */
121         if (temp & BIT7)
122         {
123             SWI2C_SDA_HIGH;
124         }
125         else
126         {
127             SWI2C_SDA_LOW;
128         }
129 
130         /* Now that we set the SDA line, we have to send out a clock
                pulse */
131         SWI2C_SCL_HIGH;
132 //          TIMER_ITERATION();
133         delay_us(dl);
134 
135         /* Incrementing to the next bit and waiting for the next clock
                cycle */
136         temp = (temp << 1);
137         bits = (bits - 1);
138 
139         SWI2C_SCL_LOW;
140 //          TIMER_ITERATION();
141         delay_us(dl);
142 
143 
144     } while (bits > 0);
145 
146     /* Detecting if we have a NAK on the bus. If the slave device NAKed
            the
147         control byte, it probably isn't there on the bus so we should
                send
148          an I2C stop and return false */
149     SWI2C_SDA_HIGH;
150     SWI2C_SCL_HIGH;
151     /*
152     * Waiting for our clock line to go high if the slave is stretching
153     */
154     while(!(SWI2C_PxIN & SWI2C_SCL));
155 
156 //    TIMER_ITERATION();
157     delay_us(dl);
```

```
158
159     if(SWI2C_PxIN & SWI2C_SDA)
160     {
161         goto I2CWriteTransactionCleanUp;
162     }
163
164     /* Sending out another clock cycle */
165     SWI2C_SCL_LOW;
166 //    TIMER_ITERATION();
167     delay_us(dl);
168
169     /* Next, let us send out all bytes in the user buffer */
170     for(ii=0;ii<size;ii++)
171     {
172         temp = outputArray[ii];
173         bits = 8;
174
175         /* Loop until all bits of the current byte are sent out */
176         do
177         {
178             /* Deciding if we want to send a high or low out of the
                    line */
179             if (temp & BIT7)
180             {
181                 SWI2C_SDA_HIGH;
182             }
183             else
184             {
185                 SWI2C_SDA_LOW;
186             }
187
188             /* Now that we set the SDA line, we send out a clock pulse
                    */
189             SWI2C_SCL_HIGH;
190 //              TIMER_ITERATION();
191             delay_us(dl);
192
193             /* Incrementing to the next bit and waiting for next clock
                    cycle */
194             temp = (temp << 1);
195             bits = (bits - 1);
196             SWI2C_SCL_LOW;
197 //              TIMER_ITERATION();
```

```
198             delay_us(dl);
199
200         } while (bits > 0);
201
202         /* Detecting the NAK. We should break out of the send loop */
203         SWI2C_SDA_HIGH;
204         SWI2C_SCL_HIGH;
205         /*
206          * Waiting for our clock line to go high if the slave is
                stretching
207          */
208         while(!(SWI2C_PxIN & SWI2C_SCL));
209
210 //          TIMER_ITERATION();
211         delay_us(dl);
212
213         if(SWI2C_PxIN & SWI2C_SDA)
214         {
215             goto I2CWriteTransactionCleanUp;
216         }
217
218         /* Sending out another clock cycle */
219         SWI2C_SCL_LOW;
220 //          TIMER_ITERATION();
221         delay_us(dl);
222     }
223
224 I2CWriteTransactionCleanUp:
225     /* If the user did not request to skip, we send out the stop bit */
226     if((sendStop && ii == size) || (ii != size))
227     {
228         SWI2C_SCL_LOW;
229 //          TIMER_ITERATION();
230         delay_us(dl);
231         SWI2C_SDA_LOW;
232 //          TIMER_ITERATION();
233         delay_us(dl);
234         SWI2C_SCL_HIGH;
235         __no_operation();
236         SWI2C_SDA_HIGH;
237     }
238     else
239     {
```

```
240             SWI2C_SCL_HIGH;
241             SWI2C_SDA_HIGH;
242         }
243
244         /* Stop the timer */
245 //      TB0CTL = MC_0;
246
247         /* If all bytes were sent, return true- otherwise false. */
248         if(ii == size)
249             return true;
250         else
251             return false;
252 }
253
254 static bool SWI2C_readData(uint8_t addr, uint8_t *inputArray,
        uint_fast16_t size)
255 {
256         uint_fast8_t bits, temp;
257         uint16_t ii = 0;
258
259         /* Starting the timer */
260 //      TB0CTL = TBSSEL_2 + MC_1 + TBCLR;
261
262         /* Sending the START */
263         SWI2C_SDA_LOW;
264         __no_operation();
265         SWI2C_SCL_LOW;
266 //      TIMER_ITERATION();
267         delay_us(dl);
268
269         /* Next doing the control byte */
270         temp = (addr << 1) | BIT0;
271         bits = 8;
272
273         /* Loop until all bits of the address byte are sent out */
274         do
275         {
276             /* Deciding if we want to send a high or low out of the line */
277             if (temp & BIT7)
278             {
279                 SWI2C_SDA_HIGH;
280             }
281             else
```

```
282             {
283                 SWI2C_SDA_LOW;
284             }
285
286             /* Now that we set the SDA line, we have to send out a clock
                    pulse */
287             SWI2C_SCL_HIGH;
288 //              TIMER_ITERATION();
289             delay_us(dl);
290
291             /* Incrementing to the next bit and waiting for the next clock
                    cycle */
292             temp = (temp << 1);
293             bits = (bits - 1);
294             SWI2C_SCL_LOW;
295 //              TIMER_ITERATION();
296             delay_us(dl);
297
298         } while (bits > 0);
299
300         /* Detecting if we have a NAK on the bus. If the slave device NAKed
                the
301             control byte, it probably isn't there on the bus so we should
                    send
302              an I2C stop and return false */
303         SWI2C_SDA_HIGH;
304         SWI2C_SCL_HIGH;
305 //      TIMER_ITERATION();
306         delay_us(dl);
307
308         if(SWI2C_PxIN & SWI2C_SDA)
309         {
310             goto I2CReadTransactionCleanUp;
311         }
312
313         /* Next, we want to read out all of the data requested */
314         for(ii=0;ii<size;ii++)
315         {
316             /*
317             * Waiting for our clock line to go high if the slave is
                    stretching
318             */
319             while(!(SWI2C_PxIN & SWI2C_SCL));
```

```
320
321         /* Setup the read variables */
322         temp = 0;
323         bits = 0x08;
324
325          /* Sending out another clock cycle */
326          SWI2C_SCL_LOW;
327 //          TIMER_ITERATION();
328          delay_us(dl);
329          SWI2C_SDA_HIGH;
330
331         /* Loop to read until all bits have been read */
332         do
333         {
334             /* Priming our temporary variable and sending a clock pulse
                    */
335             temp = (temp << 1);
336             SWI2C_SCL_HIGH;
337 //              TIMER_ITERATION();
338             delay_us(dl);
339
340             /* If the data line is high, recording that */
341             if (SWI2C_PxIN & SWI2C_SDA)
342             {
343               temp += 1;
344             }
345
346             /* Send out another clock cycle and decrement our counter
                   */
347             bits = (bits - 1);
348             SWI2C_SCL_LOW;
349 //              TIMER_ITERATION();
350             delay_us(dl);
351         }
352         while (bits > 0);
353
354         /* Storing the data off */
355         inputArray[ii] = temp;
356
357         /* Now the master needs to send out the ACK */
358         if(ii == size - 1)
359             SWI2C_SDA_HIGH;
360         else
```

```
361             SWI2C_SDA_LOW;
362             SWI2C_SCL_HIGH;
363
364             /*
365             * Waiting for our clock line to go high if the slave is
                    stretching
366             */
367             while(!(SWI2C_PxIN & SWI2C_SCL));
368
369 //              TIMER_ITERATION();
370             delay_us(dl);
371         }
372
373 I2CReadTransactionCleanUp:
374
375         /* Sending out the stop bit */
376         SWI2C_SCL_LOW;
377         SWI2C_SDA_LOW;
378 //          TIMER_ITERATION();
379         delay_us(dl);
380         SWI2C_SCL_HIGH;
381         __no_operation();
382         SWI2C_SDA_HIGH;
383 //          TIMER_ITERATION();
384         delay_us(dl);
385
386         /* Stop the timer */
387 //          TB0CTL = MC_0;
388
389         /* If all bytes were read, return true- otherwise false. */
390         if(ii == size)
391             return true;
392         else
393             return false;
394 }
```

Listagem I.10: Código swi2c_master.c (MCU Módulo lisímetro).

```
1 /* --COPYRIGHT--,BSD
2  * Copyright (c) 2016, Texas Instruments Incorporated
3  * All rights reserved.
4  *
5  * Redistribution and use in source and binary forms, with or without
```

```
6   * modification, are permitted provided that the following conditions
7   * are met:
8   *
9   * * Redistributions of source code must retain the above copyright
10  *   notice, this list of conditions and the following disclaimer.
11  *
12  * * Redistributions in binary form must reproduce the above copyright
13  *   notice, this list of conditions and the following disclaimer in
        the
14  *   documentation and/or other materials provided with the
        distribution.
15  *
16  * * Neither the name of Texas Instruments Incorporated nor the names
        of
17  *   its contributors may be used to endorse or promote products
        derived
18  *   from this software without specific prior written permission.
19  *
20  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
        "AS IS"
21  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO
        ,
22  * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
        PARTICULAR
23  * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
24  * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL
        ,
25  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
26  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
        PROFITS;
27  * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
        LIABILITY,
28  * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
        OR
29  * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
30  * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
31  * --/COPYRIGHT--*/
32 //
    ******************************************************************************

33
34 #include <msp430.h>
35 #include <stdbool.h>
```

```
36 #include <stdint.h>
37
38 /* Pin Definitions. These  should be changed depending on the device
       that
39  * you are using.
40  */
41 #define SWI2C_SCL           BIT1
42 #define SWI2C_SDA           BIT2
43 #define SWI2C_PxDIR         P2DIR
44 #define SWI2C_PxOUT         P2OUT
45 #define SWI2C_PxIN          P2IN
46 #define SWI2C_SDA_LOW       SWI2C_PxDIR |= SWI2C_SDA
47 #define SWI2C_SDA_HIGH      SWI2C_PxDIR &= ~SWI2C_SDA
48 #define SWI2C_SCL_LOW       SWI2C_PxDIR |= SWI2C_SCL
49 #define SWI2C_SCL_HIGH      SWI2C_PxDIR &= ~SWI2C_SCL
50
51 /* Defines the buffer size to be used. This will change depending on
       your
52  * application and the size requirements for the transfer.
53  */
54
55 /* Configuration structure for performing an I2C transaction */
56 typedef struct _SWI2C_I2CTransaction
57 {
58     uint8_t             address;
59     uint_fast16_t       numWriteBytes;
60     uint8_t*            writeBuffer;
61     uint_fast16_t       numReadBytes;
62     uint8_t*            readBuffer;
63     bool                repeatedStart;
64 } SWI2C_I2CTransaction;
65
66 /* Timer period for determining the clock rate of the I2C data clock.
       Note that
67  * the timer is sourced from SMCLK and that this number is equal to the
       duration
68  * of roughly HALF a clock period. For example, if SMCLK is set to 3MHz
       and the
69  * period below is set to 15, we would end up with an I2C data rate of
70  * approximately 100Khz.
71  *
72  * In short, the I2C data rate frequency can be calculated by:
73  *
```

```
74   * I2C Data Rate =       SMCLK Frequency
75   *                      ------------------
76   *
77   *                        2 * TimerPeriod
78   */
79 #define SWI2C_TIMER_PERIOD   15
80
81 /* Macro for a timer iteration */
82 /*
83 #define TIMER_ITERATION()              TB0CCTL0 &= ~(CCIFG);              \
84                                        while(!(TB0CCTL0 & CCIFG));
85 */
86 /* Function Prototypes */
87
88 //
       ***************************************************************************
89 //
90 //! Initializes the software I2C master. This function takes the port
91 //! definitions that are given above and configures the device for
       software
92 //! I2C operation.
93 //!
94 //! \return None
95 //
96 //
       ***************************************************************************
97 extern void SWI2C_initI2C(void);
98
99 //
       ***************************************************************************
100 //
101 //! Starts an I2C transaction over the configured I2C master device.
       Note that
102 //! this function is blocking until the transaction is completed. If a
       timeout
103 //! feature is required, the user should use the watchdog module of
       their MCU
104 //! in tandem with this function. Since the I2C slave has the ability
       to
105 //! "clock stretch" the module, care has to be taken to manage the real
```

```
     time
106 //!  behavior of the main application.
107 //!
108 //! <hr>
109 //! <b>Configuration options for \link SWI2C_I2CTransaction \endlink
    structure.</b>
110 //! <hr>
111 //!
112 //! \param address I2C Slave Address to communicate with.
113 //! \param numWriteBytes Number of bytes for the master to write
114 //! \param writeBuffer Pointer to the buffer to write
115 //! \param numReadBytes Number of bytes to read
116 //! \param readBuffer Pointer to the buffer to read values into
117 //! \param repeatedStart In the event that both a read and write
    operation are
118 //!         requested, this bool value determines if a repeated start
    coniditon
119 //!         is sent out over the bus. If set to true, no I2C STOP is
    sent out
120 //!         after the write transaction. If set to false. After the
    write
121 //!         transaction completes an I2C STOP condition is sent out,
    and then
122 //!         the I2C read transaction is treated as a completely
    separate
123 //!         transaction
124 //!
125 //! Note that any order of combinations can be passed into the
    transaction
126 //! structure. If the user wants to only perform an I2C read, then only
     the
127 //! read parameters should be populated and the write parameters should
     be
128 //! set to 0 (or vice versa for write). The user can also specify both
    read
129 //! and write bytes and use the repeatedStart parameter to specify if
    there
130 //! is an I2C STOP between the transactions. Note that the write
    transaction
131 //! is always first when using this function.
132 //!
133 //! \return true if the operation passed, false otherwise. A false
    return
```

```
134 //!      value means that the device received a NAK where one was not
        expected.
135 //
136 //
        ***************************************************************************

137 extern bool SWI2C_performI2CTransaction(SWI2C_I2CTransaction *
        i2cTransaction);
```

Listagem I.11: Código swi2c_master.h (MCU Módulo lisímetro).

I.2 SoC ESP8266

```
1  // https://github.com/arduino-libraries/NTPClient
2  #include <NTPClient.h>
3  // https://github.com/plapointe6/EspMQTTClient
4  #include <EspMQTTClient.h>
5  // https://github.com/evert-arias/EasyButton/
6  #include <EasyButton.h>
7  // https://github.com/tzapu/WiFiManager
8  #include <WiFiManager.h>
9
10 #define BUTTON_PIN 2     // Botão RST
11
12 EasyButton button(BUTTON_PIN);
13 WiFiManager wifiManager;
14 WiFiUDP ntpUDP;
15 NTPClient timeClient(ntpUDP, "pt.pool.ntp.org");
16
17 EspMQTTClient client(
18   "lysimeter.ddnsfree.com", // Servidor MQTT
19   1883,                     // Porta MQTT
20   "",                       // Nome de utilizador MQTT
21   "",                       // Senha MQTT
22   "ESP8266"                 // Nome que identifica o dispositivo
23 );
24
25 void setup()
26 {
27   Serial.begin(9600); // inicia a comunicação serie
28
```

```
29    wifiManager.autoConnect("AutoConnectAP");
30
31    // Inicializa o botão RST
32    button.begin();
33
34    // Ativa o evento para responder ao pressionar o botão RST
35    button.onPressed(onPressed);
36
37    // Inicia o cliente NTP
38    timeClient.begin();
39
40 }
41
42 // Função que é chamada quando o botão RST for pressionado
43 // para limpar a memoria
44 void onPressed()
45 {
46      Serial.println("Botão RST foi pressionado");
47      WiFi.disconnect(true);
48      delay(2000);
49      ESP.reset();
50 }
51
52 // Função que é chamada quando as ligações forem estabelecidas
53 // (Wifi and MQTT).
54 void onConnectionEstablished()
55 {
56    // atualiza a data logo que seja estabelecida a ligação
57    timeClient.update();
58 }
59
60 // Função que devolve uma substring de uma string com base no
61 // separador e no índice. Serve para separar os dados de cada
62 // sensor
63 String getValue(String data, char separator, int index)
64 {
65    int found = 0;
66    int strIndex[] = {0, -1};
67    int maxIndex = data.length()-1;
68
69    for(int i=0; i<=maxIndex && found<=index; i++){
70      if(data.charAt(i)==separator || i==maxIndex){
71          found++;
```

```
72          strIndex[0] = strIndex[1]+1;
73          strIndex[1] = (i == maxIndex) ? i+1 : i;
74        }
75    }
76    return found>index ? data.substring(strIndex[0], strIndex[1]) : "";
77  }
78
79  // função para converter os dados recebidos pela porta série
80  // do microcontrolador msp430 para o formato JSON para enviar
81  // por mqtt
82  String convertToJSON(String input)
83  {
84    String json = "{\"ID\":\"" + getValue(input, ' ', 0) + "\",";
85    json += "\"TIME\":\"" + String(timeClient.getEpochTime()) + "\",";
86    json += "\"STA\":\"" + getValue(input, ' ', 1) + "\",";
87    json += "\"SHA\":\"" + getValue(input, ' ', 2) + "\",";
88    json += "\"SL1\":\"" + getValue(input, ' ', 3) + "\",";
89    json += "\"SL2\":\"" + getValue(input, ' ', 4) + "\",";
90    json += "\"STS1\":\"" + getValue(input, ' ', 5) + "\",";
91    json += "\"STS2\":\"" + getValue(input, ' ', 6) + "\",";
92    json += "\"STS3\":\"" + getValue(input, ' ', 7) + "\",";
93    json += "\"SHS1\":\"" + getValue(input, ' ', 8) + "\",";
94    json += "\"SHS2\":\"" + getValue(input, ' ', 9) + "\",";
95    json += "\"SHS3\":\"" + getValue(input, ' ', 10) + "\",";
96    json += "\"SVB\":\"" + getValue(input, ' ', 11) + "\",";
97    json += "\"SPL\":\"" + getValue(input, ' ', 12) + "\",";
98    json += "\"SPR\":\"" + getValue(input, ' ', 13) + "\",";
99    json += "\"QOS\":\"" + String(signal_quality()) + "\"}";
100
101   return json;
102 }
103
104
105 int signal_quality()
106 {
107   if (WiFi.status() != WL_CONNECTED)
108     return -1;
109   int dBm = WiFi.RSSI();
110   if (dBm <= -100)
111     return 0;
112   if (dBm >= -50)
113     return 100;
114   return 2 * (dBm + 100);
```

```
115 }
116
117
118 void loop()
119 {
120   button.read(); // verifica o estado do botão RST
121
122   // verifica se as conexões foram feitas
123   if (client.isConnected())
124   {
125     // Verifica se existe alguma coisa no buffer
126     if(Serial.available()>0)
127     {
128       // lê os dados do buffer serial
129       String msg = Serial.readString();
130
131       timeClient.update(); // atualiza a data
132
133       // publica os dados no broker mqtt
134       String payload = convertToJSON(msg);
135       client.publish("LISIMETRO", payload);
136
137     }
138   }
139
140   client.loop();
141
142 }
```

Listagem I.12: Código do programa principal do ESP8266 (SoC ESP8266).

I.3 MCU do módulo da câmara

```
1 /*  IOT CAMERA - MIOT1819
2     Autor: Carlos Almeida
3  Data: 15/10/2021
4 */
5
6 #include <msp430.h>
7 #include <stdint.h>
8 #include "CDC.h"
```

```
9  #include "delay.h"
10
11
12 const unsigned long pooling_time = 14400;   // Pooling time in seconds
13
14 const unsigned int t_startup = 65 ;     // Time to startup RPi (seconds)
15 const unsigned int t_shutdown = 10 ;  // Time to startup RPi (seconds)
16
17
18 unsigned long seconds;                      // Seconds counter
19
20
21 static const unsigned int ID = C1901;     // ID da Camera
22
23
24 unsigned int SVB = 0;      // Sensor Voltagem Bateria [mV]
25
26 unsigned int arraySamples[1];    // ADC samples array
27
28
29 void GPIO_Init(void);
30 void BCM_Init(void);
31 void TIMER_Init(void);
32 void USCI_A0_Init(void);
33 void ADC10_Init(void);
34 void ADC10_StartSampling(void);
35 void sendData(void);
36
37
38 void main(void)
39 {
40      WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
41
42     seconds = pooling_time - 1;
43
44     GPIO_Init();
45     BCM_Init();
46     TIMER_Init();
47     USCI_A0_Init();
48     ADC10_Init();
49     ADC10_StartSampling();
50
51     __enable_interrupts();
```

```
52
53
54      while(1)
55      {
56          if (seconds == pooling_time) //
57          {
58              P1OUT |= BIT0;   // Power ON
59
60              seconds = 0;
61              while (seconds < t_startup);
62
63
64              //*ADC*********************************
65              ADC10_StartSampling();
66              SVB = arraySamples[0];
67              delay_ms(100); //
68              sendData();
69              seconds = 0;
70              while (seconds < t_shutdown);
71              P1OUT &= ~BIT0; // Power OFF
72          }
73
74          ___low_power_mode_3();
75      }
76  }
77
78
79  void   sendData(void)
80  {
81      p_ui(ID); ec(" ");
82      p_ui(SVB); ec(" ");
83      p_ui(SCB); ec(" ");
84      LF_();
85      CR_();
86  }
87
88
89  void GPIO_Init()
90  {
91      P1OUT = (0x00);
92      // P1DIR = (0xFF);
93      P1DIR = ( BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0); // P1DIR
            = (0xF7); BIT7->IN
```

```
94
95      // USCI:
96      P1SEL  |= (BIT2 |BIT1); // UCA0TXD , UCA0RXD
97      P1SEL2 |= (BIT2 |BIT1); // UCA0TXD , UCA0RXD
98
99
100     P1REN |= BIT7;  // enable pull-up/pull-down resistor
101     P1OUT |= BIT7;  // select pull-up resistor
102
103     P1IES |= BIT7;  // select edge interrupt high->low transition
104     P1IFG = 0x00;   // clear P1IFG since writing to P1OUT, P1DIR, P2OUT
            , or P2DIR can result in setting the corresponding P1IFG
105                         // or P2IFG flags. (slau144j pg.331)
106     P1IE |= BIT7;   // enable interrupt on P1.7
107
108 }
109
110
111
112 void BCM_Init()
113 {
114     DCOCTL = CALDCO_8MHZ;    // DCO frequency select and Modulator
            selection:
115                              // are set with factory calibrated value
116     BCSCTL1 = XT2OFF         // XT2 off
117             | CALBC1_8MHZ    // Range select: factory value for 8MHz
118             | DIVA_0;        // ACLK Divider: 0->/1, 1_>/2, 2->/4, 3->/8
119
120     BCSCTL2 = SELM_0         // Select the MCLK source: 0->DCOCLK, 1->
            DCOCLK, 2-> XT2CLK when XT2, 3->LFXT1CLK or VLOCLK
121             | DIVM_0         // MCLK Divider: 0->/1, 1_>/2, 2->/4, 3->/8
122             | DIVS_0;        // SMCLK Divider: 0->/1, 1_>/2, 2->/4,
                    3->/8:
123
124     BCSCTL3 = LFXT1S_2;      // Low-frequency clock select and LFXT1
            range: 0->Crystal 32768Hz, 2->VLOCLK, 3->Ext CLK
125 }
126
127
128 void USCI_A0_Init(void)
129 {
130     // UART Mode: 9600 8N1
131
```

```
132     UCA0CTL1 = UCSSEL_2; // USCI clock source select: SMCLK
133     // From Table 36.4 in slau144j:
134     // BRCLK Frequency (Hz)      Baud Rate   UCBRx   UCBRSx  UCBRFx
135     // 8 000 000                 9600        833     2       0
136     UCA0BR0 = 0x41;       // UCBRx = 833 = 0x341
137     UCA0BR1 = 0x03;
138     UCA0MCTL = UCBRS_2;
139     // USCI reset released for operation:
140     UCA0CTL1 &= ~ UCSWRST;
141
142     IE2 |= UCA0RXIE;      // USCI_A0 receive interrupt enable
143 }
144
145 #pragma vector = USCIAB0RX_VECTOR
146 __interrupt void USCIA0RX_ISR(void)
147 {
148     while(!(IFG2 & UCA0TXIFG));
149     UCA0TXBUF = UCA0RXBUF;
150 }
151
152
153 void TIMER_Init()
154 {
155     // Timer A0 settings for 1s counter
156     TA0CCTL0 = CCIE;      // Capture/Compare Interrupt Enable
157
158     TA0CTL = TASSEL_1     // Clock source = 0->TACLK, 1->ACLK, 2->SMCLK,
            3->INCLK
159             | MC_1        // Mode control = 1->UP / 2->CONTINUOS / 3->UP/
                DOWN
160             | ID_0;       // Divider = 0->/1, 1->/2, 2->/4, 3->/8
161     TA0CCR0 = 10350;      // Compare value: 32768 for 32768Hz crystal,
            10000 for 10KHz VLOCLK (10350 after 1s calibration)
162
163     // Timer A1 settings for PWM servo control
164     TA1CTL = TASSEL_2     // Clock source = 0->TACLK, 1->ACLK, 2->SMCLK,
            3->INCLK
165             | TACLR       // Timer_A1 clear
166             | MC_1        // Mode control = 1->UP / 2->CONTINUOS / 3->UP/
                DOWN
167             | ID_3;       // Divider = 0->/1, 1->/2, 2->/4, 3->/8
168     TA1CCR0 = 20000;       // Capture and Compare: 20000=> F=50Hz, T=20ms
169     TA1CCTL2 |= OUTMOD_7;    // Output Mode: 7-> Reset/Set
```

```
170 }
171
172
173 #pragma vector = TIMER0_A0_VECTOR
174 __interrupt void TIMER0_A0_ISR(void)
175 {
176     ++ seconds;
177     __low_power_mode_off_on_exit();
178 }
179
180
181 #pragma vector = PORT1_VECTOR
182 __interrupt void Port1_ISR(void)
183 {
184     if(P1IFG & BIT7) // switch pressed
185     {
186         seconds = pooling_time - 1;
187     }
188     P1IFG &= ~BIT7;          // clear interrupt flag of P1.3
189 }
190
191
192 void ADC10_Init(void)
193 {
194     // ADC10 Control Register 0:
195     ADC10CTL0 = SREF_1       // Select reference: VR+ = VREF+ and VR- =
            VSS
196               | ADC10SHT_0   // ADC10 sample-and-hold time: 4 ï¿½
                  ADC10CLKs
197               | ADC10SR      // ADC10 sampling rate: Reference buffer
                  supports up
198                              // to ~50 ksps
199               | REFBURST     // Reference buffer on only during
200                              // sample-and-conversion
201               | MSC          // Multiple sample and conversion
202               | REF2_5V      // Reference-generator voltage 2.5V
203               | REFON        // Reference generator on: V_REF+ takes 30
                  us to
204                              // settle (slau144j pg. 38)
205               | ADC10ON      // ADC10 on
206               | ADC10IE;      // ADC10 interrupt enable
207
208     // ADC10 Control Register 1:
```

```
209     ADC10CTL1 = INCH_6       //  Input channel selection: A6–A5–A4–A3
210               | SHS_0        //  Sample–and hold source select:
211               | ADC10DIV_3   // ADC10 clock divider: /4
212               | ADC10SSEL_0  // ADC10 clock source select: ADC10OSC – 0
213               | CONSEQ_1;    // Conversion sequence mode select:
214                              //    Sequence–of–channels (A7 – A6 – A5 –
                                   A4)
215
216
217     // Analog (Input) Enable Control Register 0:
218     ADC10AE0 |= BIT3
219               | BIT4         // ADC10 analog enable: A7 on P1.7
220               | BIT5         // ADC10 analog enable: A6 on P1.6
221               | BIT6;        //
222
223
224     // ADC10 data transfer control register 1:
225     ADC10DTC1 = 4;        // Define the number of transfers: 4 values
226 }
227
228
229 void ADC10_StartSampling(void)
230 {
231 //    while (ADC10CTL1 & ADC10BUSY);  // Wait if ADC10 core is active
232     // ADC10 data transfer start address:
233
234     ADC10SA = (unsigned int) arraySamples; // ADC10 start address
235     ADC10CTL0 |= ENC       // Enable conversion
236               | ADC10SC;
237     delay_ms(1);
238 }
239
240 #pragma vector = ADC10_VECTOR
241 __interrupt void ADC10_ISR(void)
242 {
243
244    ADC10CTL0 &= ~ENC;
245    __low_power_mode_off_on_exit();
246 }
```

Listagem I.13: Código main.c MCU (Módulo câmara).

I.4 SBC RPi

```
1
2 import time
3 import os
4 from camera import takePhoto
5 from copyimage import copyImage
6 from time import sleep
7 from shut_down  import shut_d
8 from mqtt import mqtt
9
10
11
12 def main():
13
14  try:
15   takePhoto()
16  except Exception as e:
17   print("type error: " + str(e) + " Problema com a câmara!")
18  try:
19   copyImage()
20  except Exception as f:
21   print("type error: " + str(f) + " Problema com SSH!")
22
23     try:
24   mqtt()
25  except Exception as e:
26   print("type error: " + str(e) + " Problema com MQTT!")
27
28     shut_d()
29
30 if __name__ == "__main__":
31    main()
```

Listagem I.14: Código start.py SBC

```
1 #!/usr/bin/env python
2 from picamera import PiCamera
3 from time import sleep
4 import datetime
5 from copyimage import copyImage
6
7 PATH = "/home/pi/lysimeter/image/"
```

```
8
9  def takePhoto():
10     camera = PiCamera()
11     name_photo = datetime.datetime.now().strftime('%Y%m%d-%H%M%S.jpg')
12     print("Start image capture, wait")
13     sleep(5) # Time to adjust image
14     try:
15 #        print("Iniciar captura")
16         camera.resolution = (3280, 2464)
17         camera.capture(PATH + " " + name_photo)
18 #        print("Fim de captura")
19  #       sleep(1)
20     except Exception as e:
21         print("type error: " + str(e))
22     print("End capture")
23
24 def main():
25     takePhoto()
26
27
28 if __name__ == "__main__":
29     main()
```

Listagem I.15: Código camera.py SBC.

```
1  #!/usr/bin/python
2  import os
3
4  HOST = "lisimetro@lysimeter.ddnsfree.com:"
5  PORT = "22"
6  LOCAL_PATH = "/home/pi/lysimeter/image/"
7  SERVER_PATH = "/home/lisimetro/image"
8
9  def copyImage():
10     print("scp -P "+ PORT + " " + LOCAL_PATH + "*.jpg " + HOST +
       SERVER_PATH)
11     os.system("scp -P " + PORT + " " + LOCAL_PATH + "*.jpg " + HOST +
       SERVER_PATH)
12     os.system("rm " + LOCAL_PATH + "*.jpg")
13     print("Done - File in server!")
14
15 def main():
16     copyImage()
```

```
17
18 if __name__ == "__main__":
19     main()
```

Listagem I.16: Código copyimage.py SBC.

```
1  import time
2  import paho.mqtt.client as paho
3  import serial
4  import re
5  import json
6  import os
7  from datetime import datetime
8  from camera import takePhoto
9
10 # this port address is for the serial tx/rx pins on the GPIO header
11 SERIAL_PORT = "/dev/ttyS0"
12 # be sure to set this to the same rate used on the Arduino
13 SERIAL_RATE = 9600
14
15
16 def mqtt(data):
17
18     keys = ["ID", "SVB"]
19     values = re.split("\s+", data)
20     values = [x for x in values if x]  # remove null items
21     new_dict = dict(zip(keys, values))
22     new_dict["TIME"] = str(datetime.now())
23     print("received message =", new_dict)
24     new_dict["SL"] = str(get_signal_level(wlan0))
25     topic = "CAMARA"
26     print("TOPIC: ", topic)
27     data = {
28         "ID": new_dict["ID"],
29         "TIME": new_dict["TIME"],
30         "SVB": new_dict["SVB"],
31         "SL": new_dict["SL"],
32     }
33
34     data = json.dumps(data, ensure_ascii=True)
35     print("JSON DUMP =", data)
36     broker = "lysimeter.ddnsfree.com"
37     # reading is a string...do whatever you want from here
```

```
38      client = paho.Client("client-001")
39      print("connecting to broker ", broker)
40      client.connect(broker, 1883, 60)  # connect
41      client.loop_start()  # start loop to process received messages
42      client.subscribe(topic)  # subscribe
43      client.publish(topic, data)  # publish
44
45
46  def main():
47      print("A espera de dados porta série")
48      ser = serial.Serial(SERIAL_PORT, SERIAL_RATE)
49      while True:
50          # using ser.readline() assumes each line contains a single
                reading
51          # sent using Serial.println()
52          reading = ser.readline().decode("utf-8")
53          mqtt(reading)
54
55
56  if __name__ == "__main__":
57      main()
```

Listagem I.17: Código mqtt.py SBC.

```
1  import time
2  import os
3
4  def shut_d():
5   print("The system will shutdown in 10s (Ctrl + C to cancel)")
6   for i in xrange(10, 0, -1):
7    time.sleep(1)
8    print(i)
9   os.system("sudo shutdown now")
10
11 def main():
12  shut_d()
13
14 if __name__ == "__main__":
15     main()
```

Listagem I.18: Código shutd.py SBC.

I.5 Node-Red

```
var ID = {"payload": msg.payload.ID};

var TA = {"payload": msg.payload.TA};
var HA = {"payload": msg.payload.HA};
var L1 = {"payload": msg.payload.L1};
var L2 = {"payload": msg.payload.L2};

var TS1 = {"payload": msg.payload.TS1};
var TS2 = {"payload": msg.payload.TS2};
var TS3 = {"payload": msg.payload.TS3};


var HS1 = SHS_convert(parseInt(msg.payload.HS1))
var HS2 = SHS_convert(parseInt(msg.payload.HS2))
var HS3 = SHS_convert(parseInt(msg.payload.HS3))


var PL = {"payload": msg.payload.PL};
var PR = {"payload": msg.payload.PR};

var VB = SVB_convert(parseFloat(msg.payload.VB));

var T = {"payload": msg.payload.T};


function SHS_convert(SHS){  // Converte a saida ADC em percentagem de
    humidade
    var aux = parseInt((-0.3052*SHS)+ 258.29);     //(SHS*(-1.3585)+809)
    if (aux > 100 ) aux = 100;
    if (aux < 0) aux = 0;
    return {"payload": aux.toString()};
}

function SVB_convert(SVB){ // Converte saida ADC na tensao da bateria
    em milivolts
    return {"payload": parseInt((SVB * 5.0)).toString()};
}

var NS = {"payload": msg.payload.NS};


```

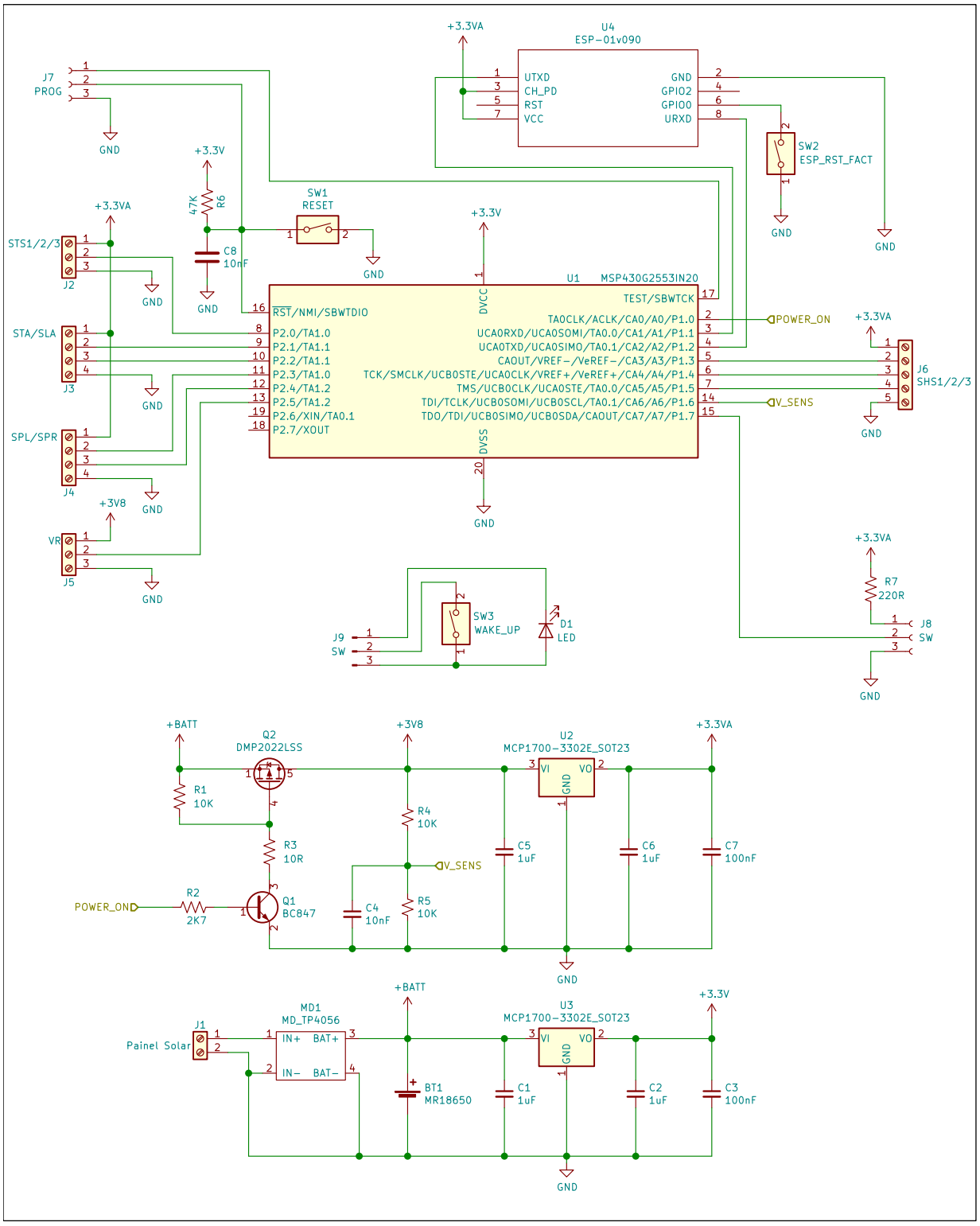
```
40 return [TA, HA, L1, L2, TS1, TS2, TS3, HS1, HS2, HS3, PL, PR, VB, T, ID
       , NS];
```

Listagem I.19: Código Data.js

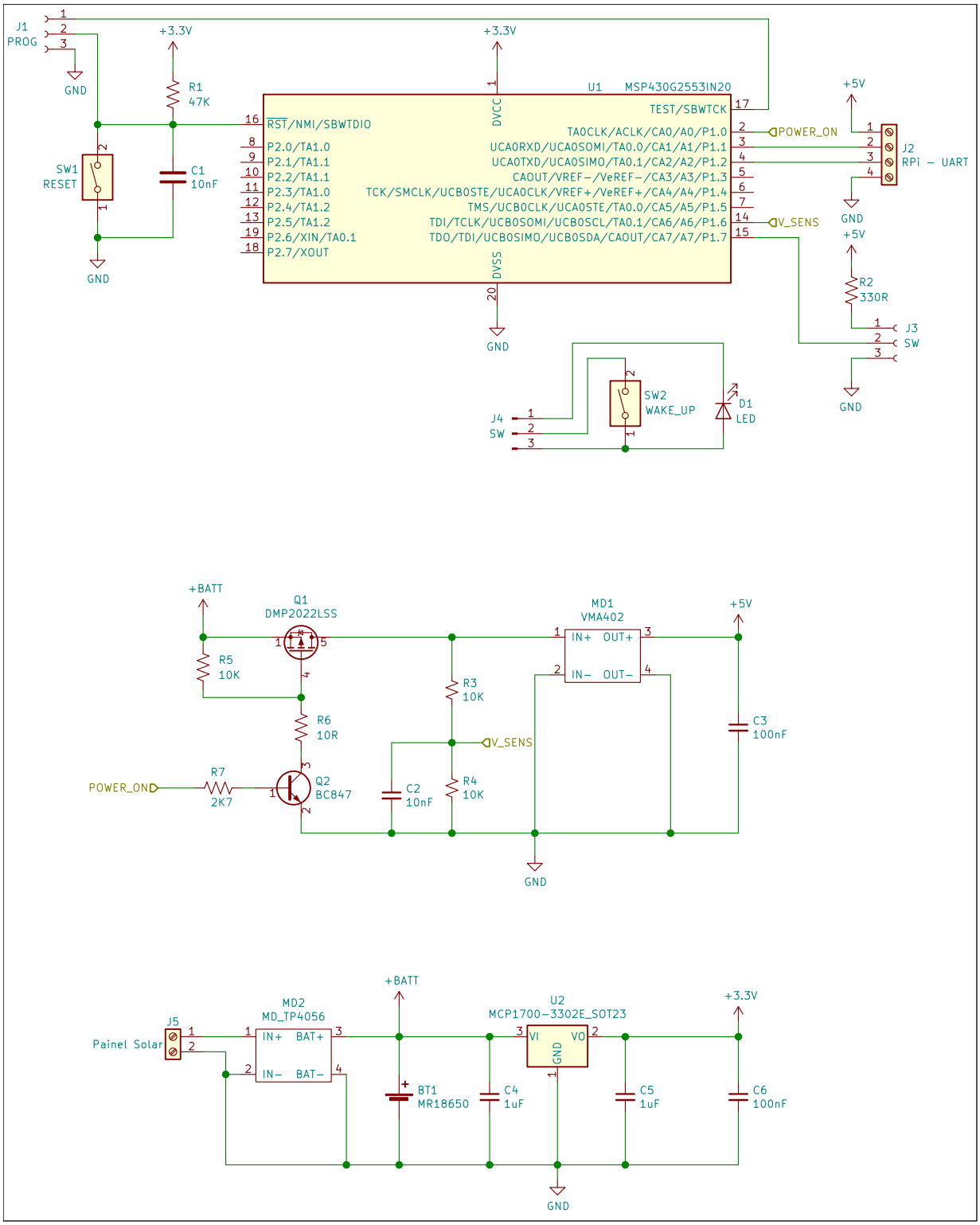
Apêndice II

Esquemas elétricos / Desenhos PCB

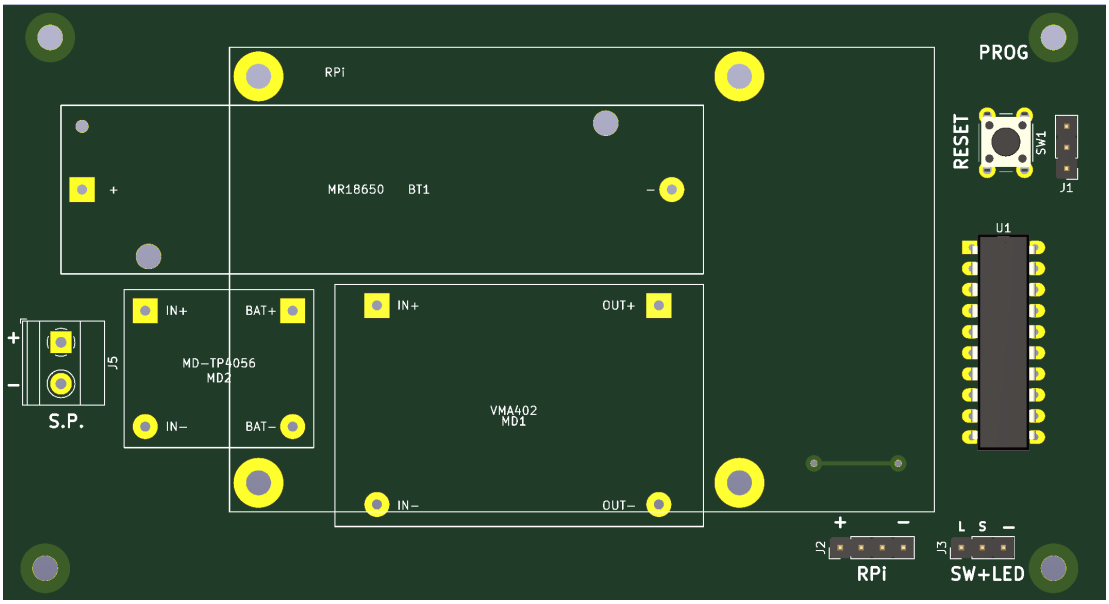
II.1 Esquema Elétrico do Módulo Lisímetro



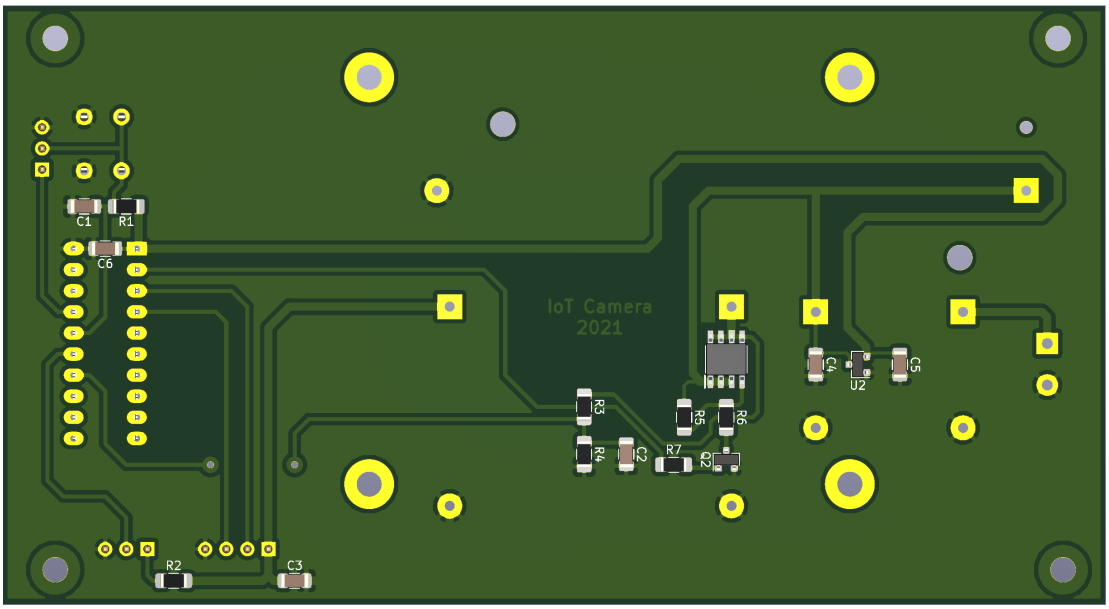
II.2 Esquema Elétrico do Módulo Câmara



II.3 Parte frontal da PCB câmara (KiCad)



II.4 Parte traseira da PCB câmara (KiCad)



Apêndice III

Custo de implementação

Orçamento de componentes para construção do prototipo

Tabela III.1: Cotação de material para o módulo lisímetro.

Qt.	Descrição	P.Unit.	Valor
1	MCU MSP430G2553 - DIP20	2.35€	2.35€
1	Módulo ESP8266 - ESP01	4.55€	4.55€
1	Placa Circuito Impresso Perfurada	7.47€	7.47€
1	Transistor DMP2022	1.20€	1.20€
1	Transistor BC847	0.10€	0.10€
2	Regulador de Tensão MCP1700-3302E	0.90€	1.80€
8	Resistências de carvão 1/4W	0.15€	1.20€
6	Condensadores 10nF 50V	0.15€	0.90€
2	Condensadores 100nF 50V	0.18€	0.36€
4	Condensadores 1uF 50V	0.22€	0.88€
2	Interrutor de pressão PCB	0.17€	0.34€
15	CABO LiYCY-DIN 4x0,50	0.97€	14.55€
4	Blocos 3 terminais	0.25€	1.00€
5	Blocos 2 terminais	0.18€	0.90 €
1	Painel solar 5V 1W	4.75€	4.75€
1	Bateria Li-Ion MR18650 3,6V 3500mAh	8.75€	8.75€
1	Módulo controlador de carga TP4056	1.96€	1.96€
1	Suporte bateria MR18650 c/ fios	1.75€	1.75€
1	Módulo sensor Temp/Humidade SHT30	19.25€	19.25€
1	Módulo sensor luminosidade TLS2561	4.80€	4.80€
3	Sensor temperatura DS18B20 - IP67	9.30€	27.90€
3	Sensor de humidade solo capacitivo	8.20€	24.60€
1	Módulo HX711	4.95€	4.95€
4	Células de carga 1/2 Ponte 0-50Kg	9.50€	38.00€
1	Células de carga 0-10Kg	13.50€	13.50€
1	Servo motor HiTEC HS-422	11.81€	11.81€
1	Caixa IP65 82.1x158.5x55mm	4.02€	4.02€
1	Caixa IP65 50x70x36mm	2.03€	2.03€
4	Caixa IP65 40x64x30mm	1.89€	7.56€
16	Bucins PG7 - IP65	1.02€	16.32€
1	Vaso Polietileno 25L	6.80€	6.80€
1	Recipiente plástico 1,5L	1.95€	1.95€
		Total	**238.30€**

Tabela III.2: Cotação de material para o módulo câmara.

Qt.	Descrição	P.Unit.	Valor
1	SBC Raspberry Pi 3B+	39.90€	39.90€
1	Cartão SD 16GB	6.00€	6.00€
1	Módulo câmara Raspberry Pi V2	27.57€	27.57€
1	MCU MSP430G2553 - DIP20	2.35€	2.35€
1	Placa de circuito impresso 1 face	4.90€	4.90€
1	Transistor DMP2022	1.20€	1.20€
1	Transistor BC847	0.10€	0.10€
1	Regulador de Tensão MCP1700-3302E	0.90€	1.80€
1	Módulo conversor DC/DC VMA402	9.90€	9.90€
7	Resistências SMD	0.10€	0.70€
8	Condensadores SMD	0.26€	2.08€
1	Interruptor de pressão PCB	0.17€	0.34€
1	Blocos 2 terminais	0.18€	0.90€
1	Painel solar 5V 1W	4.75€	4.75€
1	Bateria Li-Ion MR18650 3,6V 3500mAh	8.75€	8.75€
1	Módulo controlador de carga TP4056	1.96€	1.96€
1	Suporte bateria MR18650 PCB	3.26€	3.26€
1	Caixa IP65 82.1x158.5x55mm	4.02€	4.02€
		Total	**118.69€**

Anexos

Anexo I

Datasheets dos componentes

1. MCU: 01-MSP430g2550.pdf
2. SoC: 02-ESP-01.pdf
3. Sensor de temperatura e humidade: 03-SHT3x.pdf
4. Sensor de luminosidade: 04-TSL2561.pdf
5. Sensor de temperatura: 05-DS18B20.pdf
6. Sensor de humidade do solo: 06-CMSC.pdf
7. Sensor de peso 1/2 ponte: 07-Loadsensor.pdf
8. Conversor ADC p/ sensores de peso: 08-HX711.pdf
9. Sensor peso ponte completa: 09-TAL220M4M5.pdf
10. Servo-motor: 10-HS422.pdf
11. Controlador de carga da bateria: 11-TP4096.pdf
12. Bateria Li-Ion: 12-INR8650-35e.pdf
13. Transístor P-Mosfet : 13-DMP2022LSS.pdf
14. Transístor NPN: 14-BC847-D.pdf
15. Regulador de tensão: 15-MCP1700-330.pdf
16. Placa de desenvolvimento: 16-MSP430-EXP430G2ET.pdf

17. Placa programação SoC: 17-SBC-ESP8266-Prog.pdf
18. SBC: 18-RPi3b+.pdf
19. Conversor DC/DC : 19-LM2577.pdf