Licenciatura em Engenharia Eletrónica e Telecomunicações e Computadores

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

RELATÓRIO FINAL DO PFC

P24

Rede de monitorização multiparâmetro para contentores de reciclagem

Trabalho realizado por:
Bruno Soares - 51832 | Miguel Rocha – 51404

Orientadores:
Miguel Fernandes, João Cruz

PFC
2025 / 2026 - Verão

Esta secção apresenta alguns termos técnicos utilizados ao longo do projeto, com uma breve explicação do seu significado.

- **ADC** – Conversor analógico-digital responsável por converter sinais analógicos em valores digitais que possam ser processados pelo sistema.
- **Arduino** – Plataforma de eletrónica composta por hardware e software.
- **Arduino IDE** – Ambiente de desenvolvimento utilizado para escrever, compilar e enviar código para placas Arduino.
- **Comandos AT** – Instruções de texto utilizadas para configurar e ajustar parâmetros de módulos de comunicação.
- **Data Mules** – Topologia de recolha de dados que utiliza veículos em movimento para receber informação de telemetria.
- **DRC (Design Rules Check)** – Procedimento de validação utilizado para garantir que o design da PCB está isento de falhas.
- **ESP** – Microcontrolador usado em projetos de eletrónica e Internet das Coisas.
- **Ficheiros Gerber** – Formato de ficheiro normalizado que contém a informação para o fabrico de PCBs.
- **Host** – Sistema central que recebe, organiza e processa a informação enviada pelos restantes dispositivos na rede.
- **HX711** – Módulo conversor analógico-digital com amplificador integrado, utilizado em sistemas de medição de peso.
- **LabView** – Software utilizado para aquisição, visualização e processamento de dados de instrumentação.
- **LiDAR** – Tecnologia de medição de distâncias baseada na emissão e receção de feixes de luz.
- **LoRa** – Tecnologia de comunicação sem fios de longo alcance e baixo consumo.
- **LoRaWAN** – Protocolo de rede para longas distâncias e baixa potência, concebido para redes públicas.
- **OpenSCAD** – Software de modelação 3D baseado em código, utilizado para criar peças de forma paramétrica.

- **PCB** – Placa de circuito impresso que utiliza caminhos de cobre para encaminhar sinais elétricos entre componentes.
- **Ponte de Wheatstone** – Circuito elétrico utilizado para medir com precisão pequenas variações de resistência.
- **RFID** – Sistema de identificação por radiofrequência.
- **Strain Gauge** – Sensor que mede deformações mecânicas para determinar força ou peso.

Índices

Lista de Figuras

Lista de Tabelas

Índice

Introdução

No âmbito do desenvolvimento urbano moderno, o conceito de *Smart Cities* (Cidades Inteligentes) tem impulsionado a transição de serviços municipais tradicionais para sistemas dinâmicos. Entre estes serviços, a gestão de resíduos sólidos urbanos destaca-se como um dos maiores desafios ambientais e financeiros enfrentados pelas empresas de saneamento público. Tradicionalmente, a recolha de resíduos é efetuada com base em rotas e horários estáticos pré-programados. Esta abordagem convencional resulta em assimetrias operacionais: frequentemente, os camiões de recolha visitam contentores que se encontram praticamente vazios, gerando um desperdício desnecessário de combustível, mão de obra e desgaste de frotas, enquanto outros contentores transbordam em períodos de pico, provocando problemas de saúde pública, odores desagradáveis e a degradação estética do espaço urbano. A integração da Internet das Coisas (*Internet of Things* - IoT) surge como o elo tecnológico capaz de enfrentar estas ineficiências. Através do desenvolvimento de nós de computação embebida de baixo custo e baixo consumo energético, torna-se viável monitorizar em tempo real o estado de ocupação.

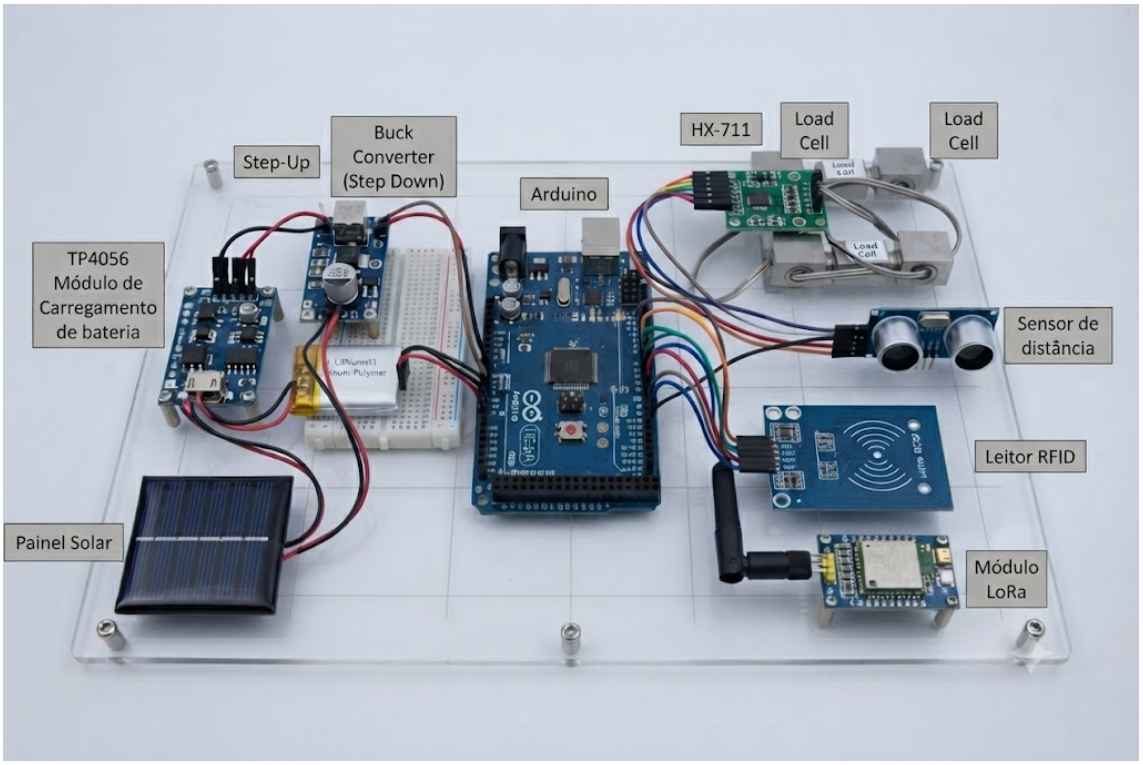


Figura 1: Conecções Projeto

1 Arquitetura Proposta

1.1 Deteção de capacidade

Para o tema principal, a deteção da capacidade, foi realizada uma pesquisa sobre os diferentes sensores, com o objetivo de chegar a um conjunto de sensores ideal para o projeto. As condições definidas passaram pela utilização de sensores com uma boa relação custo-eficiência. Para decidir que sensores utilizar, é necessário perceber o que deve ser medido. Neste projeto, cada contentor será medido de diferentes formas, pois cada um poderá variar tanto no formato como no conteúdo (líquidos, sólidos mais densos e sólidos menos densos).

1.2 Sensores

Com o objetivo de medir todo o tipo de conteúdo, é necessário definir o que qualifica um contentor como cheio. Assim, chegou-se a um consenso de classificar um contentor segundo duas vertentes: repleto e pesado. Para implementar esta classificação, é necessário um sensor de peso e um sensor que consiga confirmar que o interior do contentor está repleto (IR, US, LiDAR, etc.).

1.2.1 Sensores de peso

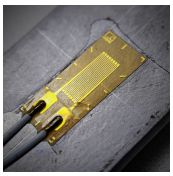
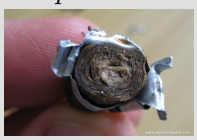
Sensor	Descrição	Vantagens	Desvantagens
Strain Gauge  Extensómetro	Varia a sua resistência elétrica quando sofre deformação mecânica, permitindo medir a força aplicada.	Boa precisão, dimensões reduzidas e baixo peso.	Estrutura frágil e necessidade de circuito de condicionamento mais complexo.
Capacitance  Sensor capacitivo	Mede variações de capacidade elétrica provocadas pela deformação das placas internas do sensor.	Boa precisão, baixo consumo e elevada resistência mecânica.	Insensível a interferências externas; a leitura e o circuito associado são mais complexos.

Tabela 1: Tabela de sensores de peso

1.2.2 Sensores de deteção

Sensor	Descrição	Vantagens	Desvantagens
Ultrassom  HC-SR04	Envia impulsos ultrassónicos e mede o tempo de retorno do eco para determinar a distância até ao objeto.	Boa precisão a curtas distâncias, resolução de cerca de 1 cm e integração simples com Arduino.	Menos eficaz em espaços muito reduzidos, dependente da textura do objeto e do ângulo de incidência.
Infravermelho  HW-201	Emite um feixe de luz infravermelha; quando o feixe é refletido por um objeto, o sensor deteta a sua presença.	Deteção simples de objetos, implementação fácil e baixo custo.	Sensível a objetos transparentes ao infravermelho e a superfícies pouco refletivas.
Feixe  Sensor de feixe	Utiliza um emissor e um recetor; quando o feixe de luz é interrompido, o sensor assinala a presença de um objeto.	Fácil deteção de objetos, montagem simples e resposta rápida.	Vulnerável a objetos transparentes ao infravermelho e requer alinhamento entre emissor e recetor.
LiDAR  LiDAR	Contém um emissor e um recetor, calculando a distância através do tempo que o feixe demora desde a emissão até à reflexão e receção.	Apresenta maior precisão, permitindo criar um mapa 3D do ambiente lido.	Vulnerável a objetos transparentes ao infravermelho e não refletivos.

Tabela 2: Tabela de sensores de deteção

1.3 Comunicação entre dispositivos

A infraestrutura de comunicação do sistema foi projetada segundo uma abordagem de recolha móvel conhecida como topologia baseada em mulas de dados (*Data Mules*). Em vez de depender de uma rede fixa de *gateways*, o sistema tira partido das rotas regulares dos veículos municipais de recolha de resíduos (a carrinha de recolha) para efetuar o escoamento dos dados de telemetria.

O fluxo arquitetural e a dinâmica de comunicação rádio estruturam-se em três fases:

1. **Transmissão e Baixo Consumo:** O nó sensor instalado no contentor de reciclagem opera de forma autónoma e maioritariamente em modo de repouso (*Idle/Sleep Mode*) para preservar a bateria. Periodicamente, ou após a deteção de uma nova deposição via RFID, o Arduino Nano processa os dados de peso e nível de enchimento. O módulo LoRa local faz o *broadcast* rádio da trama de dados contendo o identificador único do contentor, o peso acumulado e o estado de capacidade.

2. **Interceção em Movimento:** A tecnologia LoRa (Long Range) é o elemento fulcral que permite esta arquitetura em movimento. Devido à sua elevada sensibilidade de receção, o sinal emitido pelo contentor consegue ser intercetado com sucesso pelo veículo de recolha assim que este entra no raio de cobertura rádio do contentor (geralmente centenas de metros), mesmo sem a necessidade de uma paragem prolongada.

3. **Centralização Local e Interface de Operação:** No interior do veículo de recolha encontra-se instalado o nó recetor dedicado, composto por outro módulo LoRa. Este módulo está em escuta contínua e, ao captar a transmissão do contentor que se avizinha, desmodula a mensagem e reencaminha-a instantaneamente, através da interface série UART (USB), para o computador de bordo onde corre a aplicação gráfica *Entity Weight Dashboard*.

Esta abordagem confere ao projeto uma vantagem significativa: os operadores na carrinha recebem alertas em tempo real no *dashboard* sobre o estado do contentor, mesmo antes de pararem junto dele. Se o contentor enviar uma mensagem a indicar que o peso ou o volume estão muito abaixo do limite, a equipa pode otimizar a rota e seguir em frente sem perder tempo a inspecioná-lo visualmente, reduzindo diretamente os custos operacionais de combustível e as emissões poluentes.

1.4 Identificação e Autenticação de Utilizadores (RFID)

O controlo de acesso e a rastreabilidade do depósito de lixo no contentor inteligente são assegurados pelo subsistema de identificação por radiofrequência (RFID), implementado através do leitor MFRC522. Na arquitetura global do projeto, este módulo atua como o mecanismo de ativação do ciclo de leitura física e serve como identificador indireto do utilizador.

A integração arquitetural e o comportamento dinâmico deste sistema baseiam-se nos seguintes aspetos:

- **Interface de Comunicação de Alta Velocidade:** Ao contrário de outros periféricos o módulo RFID liga-se ao Arduino Nano através de uma interface *Serial Peripheral Interface* (SPI) síncrona. Esta escolha garante uma taxa de transferência de dados elevada e uma latência de milisegundos na captura do identificador único (*UID - Unique Identifier*) do cartão ou tag de aproximação do cidadão, o que é benéfico e prático para uma pessoa que vá depositar lixo.

- **Gestão de Consumo Energético por Interrupções:** Para cumprir as restrições energéticas, o leitor RFID não se encontra em amostragem contínua acelerada. Em vez disso, o firmware do Arduino utiliza o *Timer1* para acordar o sistema em intervalos regulares. O microcontrolador deteta a presença de um cartão. Caso não seja detetado nenhum utilizador, o Arduino regressa de imediato ao estado de baixo consumo (*Idle Mode*).

- **Sincronismo de Dados:** Quando ocorre uma aproximação bem-sucedida, o módulo descodifica o *UID* do utilizador. Este evento inicia a leitura do peso antes e depois da deposição do resíduo, calculando o peso líquido adicionado.

Desta forma, o pacote final enviado por LoRa para a carrinha de recolha deixa de ser uma métrica isolada e passa a ser uma estrutura de dados enriquecida, contendo o par ordenado $[UID_{utilizador}, Peso_{adicionado}]$, viabilizando no *dashboard* local as estatísticas de utilização por cidadão ou habitação.

1.5 Conclusão da Seleção de Componentes

Dos sensores apresentados, foi decidido utilizar, para as medições de peso, quatro extensómetros, formando uma ponte de *Wheatstone* completa. Nesta montagem, quando não houver peso aplicado, a tensão na ponte será nula. Quando existir deformação provocada pelo peso, a resistência variará em valores inferiores a 1 Ω, o que originará uma variação de tensão na ordem dos milivolts. Por esse motivo, é necessário ligar esta ponte a um amplificador.
Para a amplificação e conversão do sinal analógico em digital, é utilizado o ADC com amplificador integrado HX711, desenvolvido para aplicações com extensómetros e adaptado a variações de milivolts. Também é necessário detetar se o caixote se encontra repleto por outros métodos, quando a medição por peso não é fiável, como acontece, por exemplo, no papelão e no plástico, em que o peso é mais relativo.
Assim, é possível que o caixote esteja cheio e que o peso não seja suficiente para ser considerado cheio. Para resolver este problema, foi decidida a implementação de um sensor capaz de detetar quando o conteúdo do caixote ultrapassa um determinado nível. Para a infraestrutura de comunicação rádio, foi selecionado o módulo LoRa REYAX RYLR998. Esta escolha não representa a solução ideal ou mais escalável para um ecossistema comercial de *Smart Cities*. A sua seleção foi motivada, fundamentalmente, por critérios de restrição orçamental e pela disponibilidade imediata deste material, permitindo a viabilização prática do protótipo dentro do cronograma estipulado.

O módulo RYLR998 apresenta uma limitação arquitetural: opera através de um protocolo proprietário baseado em comandos AT sobre ligação série, o que significa que apenas consegue comunicar diretamente com outros módulos rigorosamente idênticos da mesma fabricante. Esta característica impede o transrecetor de se integrar na pilha de protocolos normalizada do ecossistema LoRaWAN.

Consequentemente, o nó sensor torna-se incapaz de comunicar com *gateways* públicas ou comerciais já existentes na malha urbana, como as redes da *The Things Network* (TTN) ou de operadoras de telecomunicações. Numa solução industrial ideal, a conformidade com o padrão LoRaWAN seria mandatória, uma vez que permitiria ao contentor despachar o seu estado de ocupação e peso através de qualquer antena pública circundante diretamente para um servidor centralizado na nuvem (*Cloud Server*). Isto eliminaria a dependência de uma arquitetura móvel de recolha de dados e expandiria o alcance global do sistema de monitorização, mantendo a topologia de mulas de dados apenas para zonas mais isoladas onde não existem *gateways* públicas.

2 Desenvolvimento do Hardware

2.1 Design do esquema elétrico

O esquema elétrico foi concebido com base no diagrama abaixo e com o objetivo de eliminar qualquer dependência de módulos comerciais pré-fabricados, como carregadores de baterias ou conversores comerciais. Nesta primeira iteração, contudo, a conversão de potência para o módulo LoRa foi negligenciada.

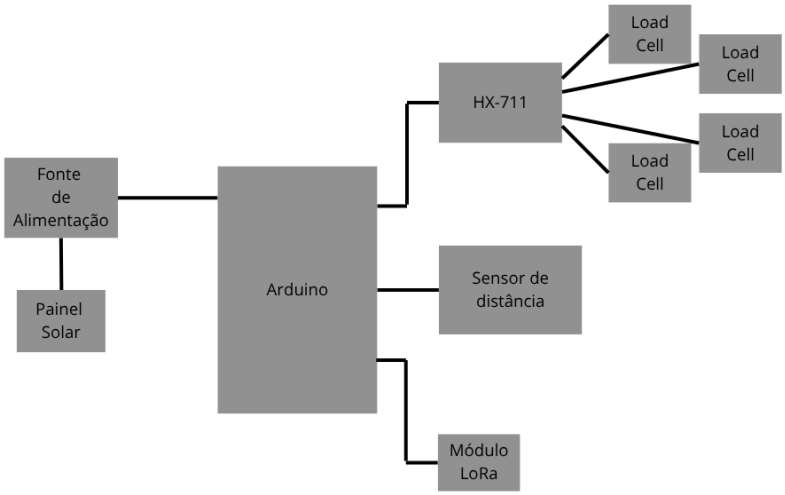


Figura 2: Diagrama inicial para a PCB

Depois ao se revelar a necessidade de alimentar o módulo LoRa diretamente e não pelo arduino foi alterado o esquemático para a PCB e chegou-se a um novo diagrama de blocos que se encontra a baixo. As ligações são feitas diretamente ao *Arduino*, com exceção das *Load Cells*, que necessitam do *HX711* para transmitir a informação necessária, e do painel solar, que serve de recarga à fonte de alimentação, e do módulo LoRa que é alimentado pelo conversor(Step-down).

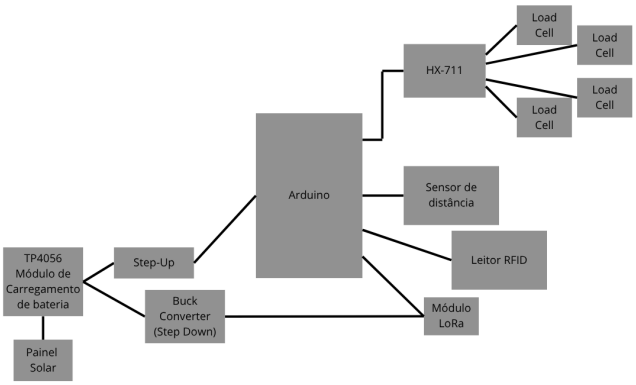


Figura 3: Diagrama atualizado para a PCB

Com o diagrama de blocos, partiu-se para o esquema elétrico onde surgiram duas iterações, a primeira foram implementados os módulos diretamente na placa de acordo com as especificações dos fornecedores,como demonstrado na seguinte figura.

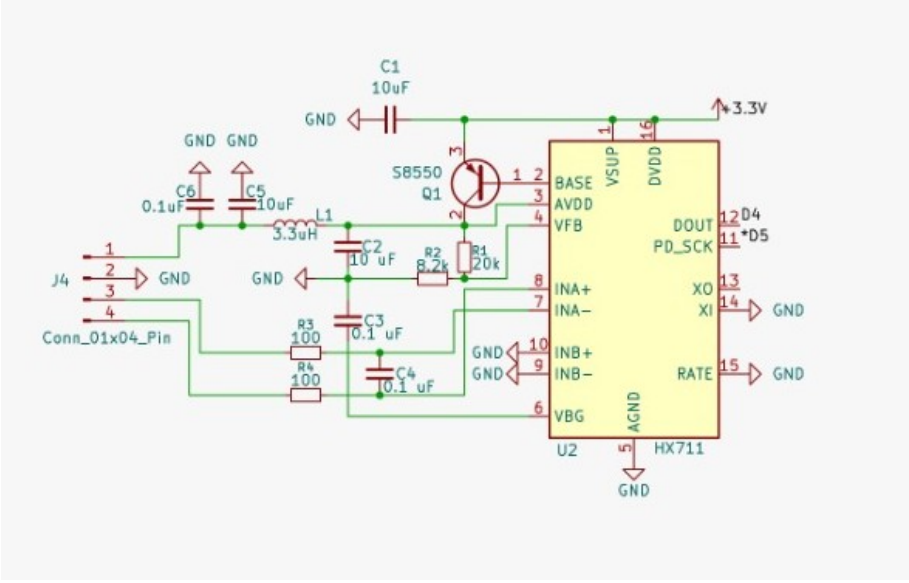


Figura 4: Esquemático do HX711

Para o esquemático final que vem abaixo foi concluido que seria mais benéfico para o projeto,trazer os módulos fabricados e introduzilos diretamente na placa. O que facilita a montagem e troca de componentes, caso seja necessário.

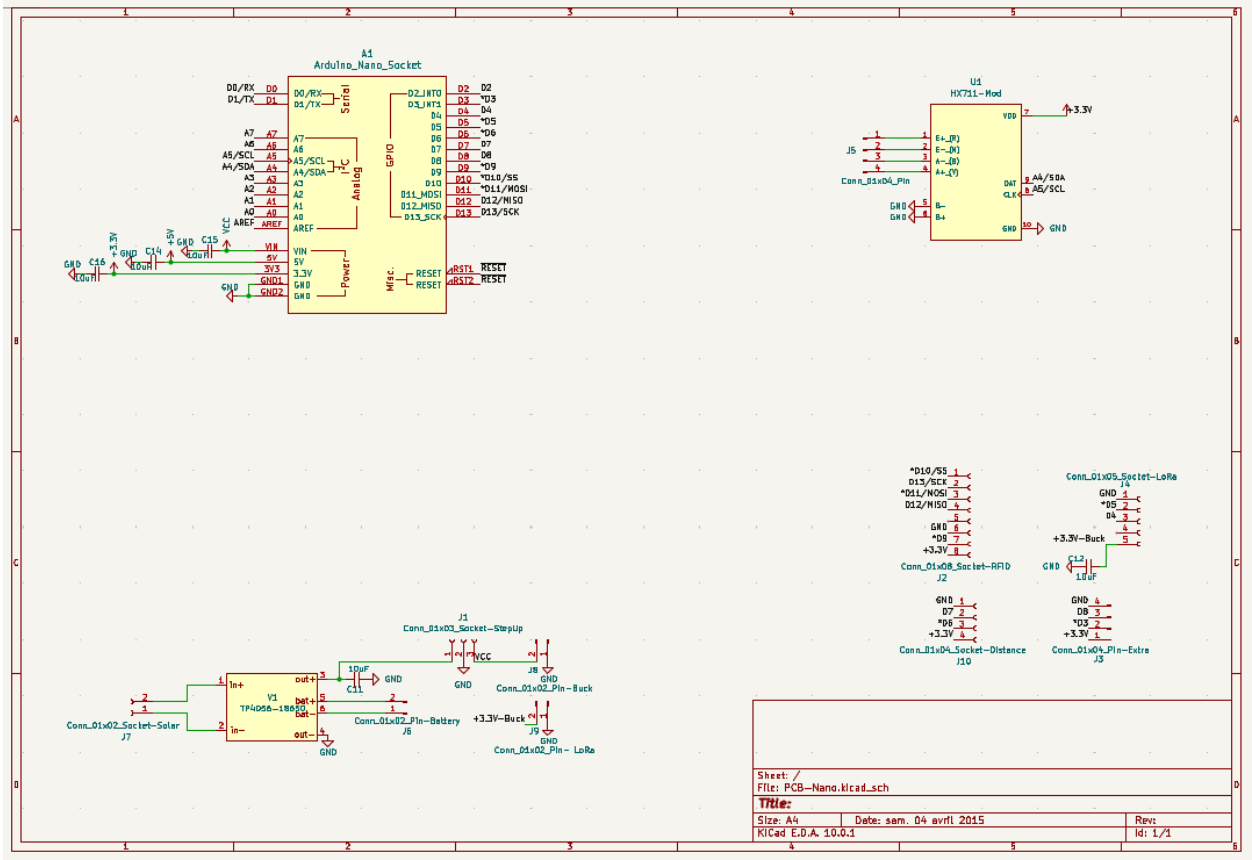


Figura 5: Esquema elétrico completo

2.2 Design da PCB

2.2.1 Introdução às Placas de Circuito Impresso (PCBs)

Uma placa de circuito impresso (PCB, do inglês *Printed Circuit Board*) é a base elétrica de muitos sistemas eletrónicos. Consiste num substrato não condutor — tipicamente fibra de vidro (FR-4) — laminado com uma ou mais camadas finas de folha de cobre condutora. Em vez de depender de cablagem manual ponto a ponto, uma PCB utiliza caminhos de cobre gravados quimicamente ou fresados mecanicamente, conhecidos como *pistas*, para encaminhar sinais elétricos e potência entre os componentes. Os componentes são fixados e ligados à placa através de soldadura em *ilhas* (*pads*) ou por intermédio de *vias* (usadas para fazer ligações entre os lados das placas). Para prototipagem de pequena escala, as placas de face simples são frequentemente preferidas devido à facilidade de fabrico manual, elevada durabilidade mecânica e simplicidade na inspeção visual. Embora as validações iniciais do circuito tenham sido realizadas com sucesso em placas de ensaio (*breadboards*), a transição do protótipo para uma PCB personalizada de face simples foi uma necessidade para este projeto. A decisão foi fundamentada pelos seguintes fatores:

- **Resistência Parasita e Integridade do Sinal:** As placas de ensaio introduzem capacidades parasitas e uma elevada resistência de contacto (0.05 Ω a 0.1 Ω por ligação). Para linhas críticas de dados, como a *interface UART* entre o Arduino Nano e o módulo LoRa REYAX RYLR998, estes elements parasitas podem distorcer os impulsos dos níveis lógicos, resultando na corrupção de dados. A PCB personalizada minimiza o comprimento dos caminhos e otimiza a geometria das pistas para garantir uma elevada integridade do sinal.

- **Distribuição de Potência:** A rede de distribuição de potência deste design deve suportar perfis de corrente variáveis, incluindo correntes de carga de até 1 A na interface do TP4056 e picos severos de corrente transitória de até 140 mA quando o módulo LoRa transmite. As linhas de ligação de uma *breadboard* não conseguem suportar estas cargas a longo prazo. Ao desenhar a PCB à medida, as pistas de potência e de terra (*GND*) foram alargadas para 1.0 mm, diminuindo significativamente a resistência da pista, eliminando riscos térmicos e facilitando a criação da PCB na universidade, uma vez que pistas mais estreitas poderiam originar problemas após o banho químico.

- **Integração de Tensões Mistas:** O sistema necessita de alimentar o Arduino e também o Dispositivo LoRa,sendo que este necessita de mais corrente que o arduino disponibiliza, exigindo um barramento de 5V (Arduino Nano) e outro de 3.3V (módulo LoRa RYLR998). A PCB permite a integração de um módulo de 3 pinos para o conversor elevador (*step-up*) e um regulador dedicado LM1117-3.3 para gerir alimentar diretamente o módulo loRa.

2.3 Regras de Design (DRC) e Preparação para Produção

O design digital no KiCad foi feito seguindo procedimentos habituais. Para garantir que o circuito desenhado é perfeitamente exequível e isento de falhas de fabrico comuns, foram aplicadas as ferramentas normativas de validação integradas no KiCAD.

O processo de verificação e preparação para produção estruturou-se nos seguintes passos fundamentais:

1. **Configuração das Restrições e Design Rules Check (DRC):** No editor de PCB, foram aplicadas as regras físicas com base no padrão de fabrico (como o isolamento mínimo e a largura de pista aceitável por fresagem CNC ou corrosão química). Após o posicionamento e o *routing*, foi corrido o *Design Rules Check* (DRC). O DRC realizou uma verificação geométrica tridimensional na placa para inspecionar:

   - **Folga Mínima (*Clearance*):** Garantia de que a distância entre pistas adjacentes ou ilhas (*pads*) é suficiente para evitar arcos elétricos ou pontes de solda inadvertidas.
   - **Largura de Pista:** Confirmação de que nenhuma linha de potência ou sinal ficou abaixo dos limites estipulados (1.0 mm e 0.8 mm, respetivamente).
   - **Conexões isoladas:** Verificação de que nenhuma malha ficou por fechar (pistas incompletas).

   A obtenção de um relatório com zero erros de DRC validou a viabilidade física do protótipo.

2. **Geração dos Ficheiros de Fabrico Normalizados (Gerber):** Com o *layout* validado pelo DRC, o projeto foi finalizado através da exportação dos ficheiros padronizados para circuitos impressos:

   - **Ficheiros Gerber (RS-274X):** Gerou-se o ficheiro da camada inferior de cobre (*B.Cu*) contendo a geometria exata das pistas, e o ficheiro de contorno físico da placa (*Edge.Cuts*) que define o perímetro do corte.
   - **Ficheiro de Furação Excellon:** Exportou-se o mapa de furação automatizado (*Drill File*), que armazena as coordenadas X-Y exatas e os respetivos diâmetros de broca para todos os componentes THT (*Through-Hole Technology*).

Esta metodologia rigorosa assegura que o projeto digital se encontra totalmente concluído, validado e documentado. Qualquer laboratório ou fabricante de PCBs consegue, a partir destes ficheiros gerados, produzir a placa física sem necessidade de ajustes ou intervenção manual no design.

Como referido anteriormente esta parte foi feita em duas iterações, esta primeira versão do circuito caracterizou-se pelos seguintes fatores técnicos:

- **Montagem em Superfície (SMD):** Para otimizar a área da placa, o circuito integrou circuitos integrados (ICs) dedicados, transístores (Q_1),tal como o uso resistências em encapsulamentos SMD, e foi utilizado como refência os esquemáticos de alguns componentes já utilizados.
- **Alimentação:** Em vez de utilizar blocos funcionais externos, foram desenhados circuitos integrados diretamente na PCB para a gestão de carga da bateria de iões de lítio, regulação linear local e filtragem de ruído de alta frequência.

Como se pode observar nas imagens abaixo, esta abordagem revelou-se excessivamente complexa para a realidade prática. A elevada densidade de pistas finas e a necessidade de soldadura manual de componentes SMD microscópicos aumentavam drasticamente a probabilidade de falhas por pontes de soldadura, pistas partidas durante a fresagem mecânica e, também, a possibilidade de não ser possível adquirir alguns componentes.

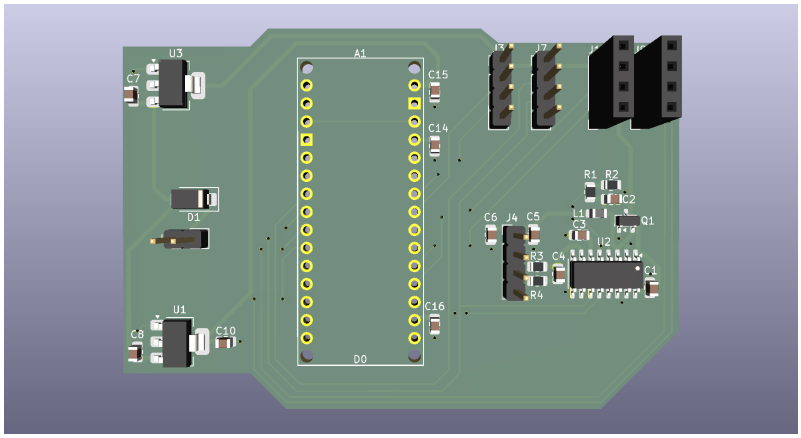


Figura 6: Modelo 3D da placa inicial

2.3.1 Segunda Iteração: Simplificação da PCB

Devido às restrições físicas de fabrico da PCB, o design foi reformulado para uma arquitetura híbrida e simplificada, substituindo os circuitos discretos complexos por módulos comerciais, utilizados posteriormente, bem como outros módulos introduzidos nesta fase. Nesta etapa, foi também introduzida uma secção dedicada à alimentação do módulo LoRa.

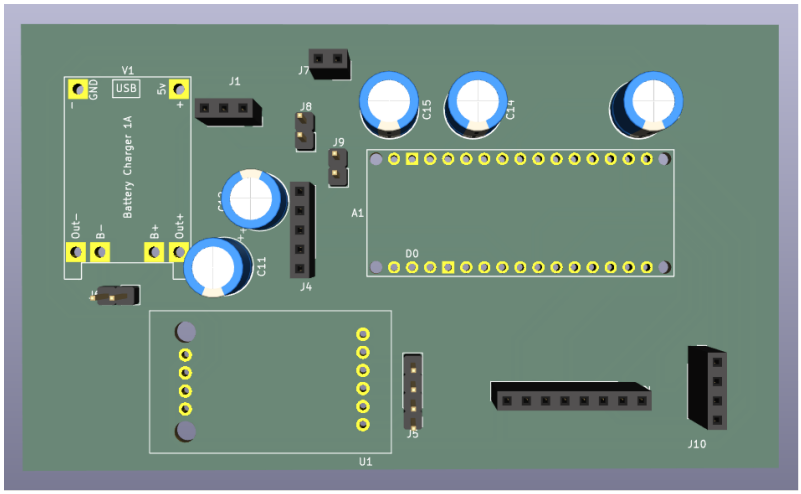


Figura 7: Modelo 3D da placa

Nesta versão final, a complexidade do circuito foi substituida por subsistemas modulares:

- **Carregador Integrado via Módulo:** O circuito de carga foi implementado com um módulo TP4056 dedicado (V_1), posicionado na extremidade esquerda da placa.
- **Interface Modular de Potência:** A regulação e elevação de tensão foram delegadas a um conversor elevador (*step-up*) externo, acoplado à placa principal através de um barramento de 3 pinos (J_1).
- **Migração para Furação (THT):** Toda a placa foi convertida para a tecnologia *Through-Hole*, utilizando ilhas largas e pistas robustas de 1.0 mm na camada inferior (*B.Cu*).

2.4 Escolha de um meio de comunicação

A seleção do módulo de comunicação sem fios baseou-se na necessidade de um sinal que consiga transmitir a Longa Distância e com baixo consumo de energia e facilidade de integração. O módulo REYAX RYLR998 é um bom começo para a demonstração do projeto, pois opera na banda de frequências ISM de 868 MHz (conforme a regulamentação europeia para dispositivos de curto alcance) e pela sua simplicidade, utilizando LoRa, que é uma tecnologia de rádio, utilizada para tranfêrencia de informação a longa distância e com baixo consumo.

O módulo escolhido tem a vantagem de ser programável com commandos *Hayes*(comandos AT) através de uma interface série UART. Esta característica permite a configuração e ajuste de parâmetros críticos como a largura de banda (*Bandwidth* - BW) e a taxa de codificação (*Coding Rate* - CR) através de simples instruções de texto. Deste modo, o microcontrolador central (Arduino Nano) liberta memória de programa (*Flash*) e poder de processamento, uma vez que não necessita de bibliotecas dedicadas ao módulo.

2.4.1 Incompatibilidade e Restrições de Uso com Gateways LoRaWAN

Uma limitação fundamental subjacente ao REYAX RYLR998, que dita a arquitetura de rede deste projeto, prende-se com a sua incapacidade nativa de comunicar com *gateways* normalizados da rede LoRaWAN (como as infraestruturas associadas à *The Things Network* - TTN).

O ecossistema LoRa divide-se em duas camadas distintas: a camada física (LoRa, que define a modulação por variação de frequência chirp) e a camada de rede (LoRaWAN, um protocolo de rede aberta com topologia em estrela gerido por uma aliança internacional). O módulo RYLR998 implementa de raiz um firmware fechado da REYAX focado exclusivamente em comunicações P2P (*Peer-to-Peer*, ou ponto a ponto). Isto significa que o protocolo de encapsulamento de dados nas tramas de rádio obedece a um endereçamento proprietário do fabricante, concebido para que um módulo RYLR998 comunique apenas com outro módulo idêntico (ou compatível com o mesmo protocolo de comandos AT).

Visto que o firmware interno do RYLR998 não possui suporte para a pilha de protocolos (*stack*) oficial do LoRaWAN uma *gateway* LoRaWAN convencional interpretará as transmissões deste módulo como ruído ou pacotes corrompidos. Consequentemente, para viabilizar a topologia de rede do projeto, adota-se uma arquitetura descentralizada onde o nó concentrador (recetor acoplado ao *dashboard*) utiliza outro módulo RYLR998 idêntico para efetuar a desmodulação direta e segura das tramas de dados recebidas.

3 Firmware

3.1 Aplicação de sensores

Por motivos de demonstração e apresentação, o projeto será criado em pequena escala. Por conseguinte, o dispositivo será mais pequeno e utilizará materiais diferentes dos que seriam usados em contentores de grande porte.

3.1.1 Sensor de peso

Para aquele que será o sensor mais importante deste projeto, foi necessário decidir que sensor escolher para a medição do peso. Perante as opções disponíveis, foi escolhido o *Strain Gauge*. Foi também recebido um sensor baseado em *Strain Gauge*.

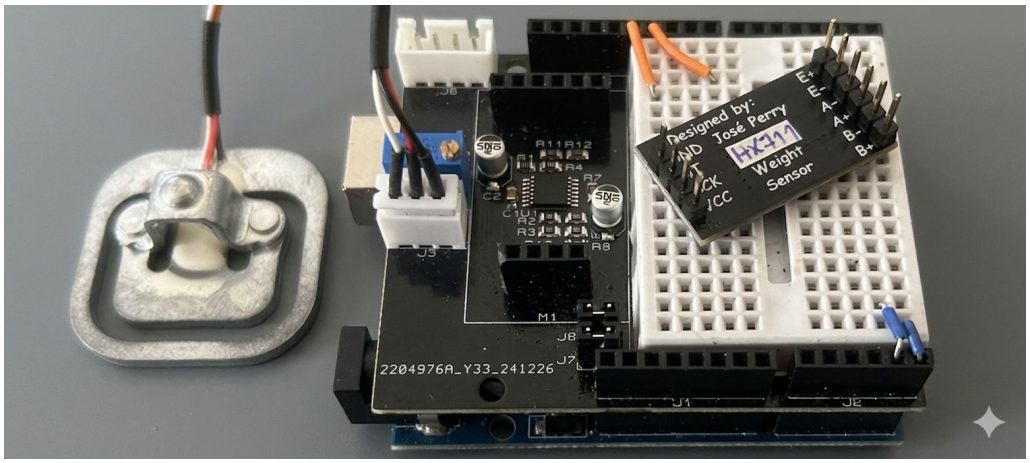


Figura 8: Sensor de peso

Este sensor comunica com uma aplicação utilizada na unidade curricular de Instrumentação e Medição, o *LabView*; no entanto, para simplificar, será utilizada a janela do *ArduinoIDE* para a obtenção de dados. O sensor baseia-se numa diferença de resistência para calcular o peso, contendo uma ponte de *Wheatstone* (um circuito que mede com precisão uma resistência desconhecida, sendo essa resistência a do *strain gauge*), em conjunto com um amplificador, para obter uma maior resolução.

3.1.2 Código do sensor de peso

O código de origem utilizado com este sensor foi desenvolvido pelo colega José Perry, sendo também utilizada uma biblioteca do *GitHub* de autoria desconhecida. Após o estudo desse código, foi desenvolvido um código original, também com recurso à mesma biblioteca do *GitHub*.

```
// HX711 Channels Gain
#define                 CHANNEL_A_GAIN
↪           (hx711_chanGain_t)
↪   CHAN_A_GAIN_128

#define         HX711_AVG_SAMPLES   5
// Zero
#define         ZERO_TRESHOLD (int)4
// Class Variables
Adafruit_HX711 HX711(HX711_DT_PIN,
↪   HX711_SCK_PIN);
Scale scale;
```

Nesta secção do código é definido o ganho da *Load Cell*, neste caso para ±40 mV. É também definido o número de amostras para o cálculo do valor médio. Com o suporte da biblioteca da Adafruit, é inicializado o módulo HX711 com os pinos definidos para *Data* e *System Clock*.

Esta função auxiliar serve para calcular a média dos valores recebidos, de modo a obter um valor mais próximo do real.

```
void TARE_channel (hx711_chanGain_t
↪   gain)
{
        int32_t mean = 0;
        for (uint8_t i = 0; i <
        ↪   MEAN_VALUES; i++)
        {
                mean += HX711.readChan
                ↪   nelRaw(gain);
                delay(10);
        }
        mean /= HX711_AVG_SAMPLES;
        HX711.tareA(mean);
}
```

```
void setup()
{
        Serial.begin(115200);

        // Pin directions
        pinMode(LED_PIN, OUTPUT);
        pinMode(LED_OVER_PIN, OUTPUT);
        pinMode(LED_UNDER_PIN, OUTPUT);
        pinMode(LM0, OUTPUT);
    ...
        pinMode(LM5, OUTPUT);

        pinMode(A0, INPUT);

        HX711.begin();
        delay(50);
        scale.begin();
        TARE_channel(CHANNEL_A_GAIN);
}
```

No *setup* do *Arduino* foi feita a definição dos pinos de entrada e saída, bem como a inicialização do módulo HX711, através das funções da biblioteca *Adafruit*.

Nesta secção de código é implementada a lógica de transformação do valor recebido em escala para percentagem, sendo previamente definido um limite, neste caso 200000. Consoante a percentagem, é acionado um segmento de 6 *LEDs*, tendo sido também implementada uma forma externa de fazer a tara através de um botão.

```
void loop()
{
    ...

    int32_t currentReading = HX711.rea
    ↪  dChannel(CHANNEL_A_GAIN);
    ChannelA_ADC = currentReading;

    if (ChannelA_ADC > MP)ChannelA_ADC
    ↪  = MP;
    int ADC_percent = ChannelA_ADC *
    ↪  100 /MP;

    ...

    if (ADC_percent >= (100/6)*1)
    ↪  digitalWrite(LM0, HIGH);
    ...
    if (ADC_percent >= (100/6)*6)
    ↪  digitalWrite(LM5, HIGH);

    // Set LEDs
    digitalWrite(LED_OVER_PIN,  (rawA
    ↪  >  ZERO_TRESHOLD));
    digitalWrite(LED_UNDER_PIN, (rawA
    ↪  < -ZERO_TRESHOLD));
    digitalWrite(LED_PIN,
    ↪  !digitalRead(LED_PIN));
    int val = digitalRead(A0);
    if(val)TARE_channel(CHANNEL_A_GAIN);
    sprintf(text, ":%d \n", val,
    ↪  ChannelA_ADC);
    // Send data
    if (Serial)
    {
        sprintf(text, ":%d
        ↪  ,A%+08ld\n", ADC_percent,
        ↪  ChannelA_ADC);
        Serial.write(text);
    }
}
```

3.1.3 Optimização do Consumo por Software

Para reduzir ainda mais o consumo do sistema, foi implementada uma solução por *software*, seguindo a estrutura do fluxograma apresentado na Figura 9. Nesta solução, o Timer1 é utilizado para gerar interrupções de segundo a segundo, de modo a verificar se existe um cartão presente. Caso exista, é lido o ID, seguindo-se uma medição do peso adicionado e a verificação de que os objetos colocados no contentor não ultrapassaram um determinado limite. Em seguida, o ID lido, o peso total e o peso adicionado são enviados para o gestor de dados através do transcetor LoRa. No final do ciclo, é verificado se já passou uma hora desde a última medição; caso isso se confirme, é realizada e enviada uma nova medição de peso. Por fim, o Arduino entra em *Idle Mode*.

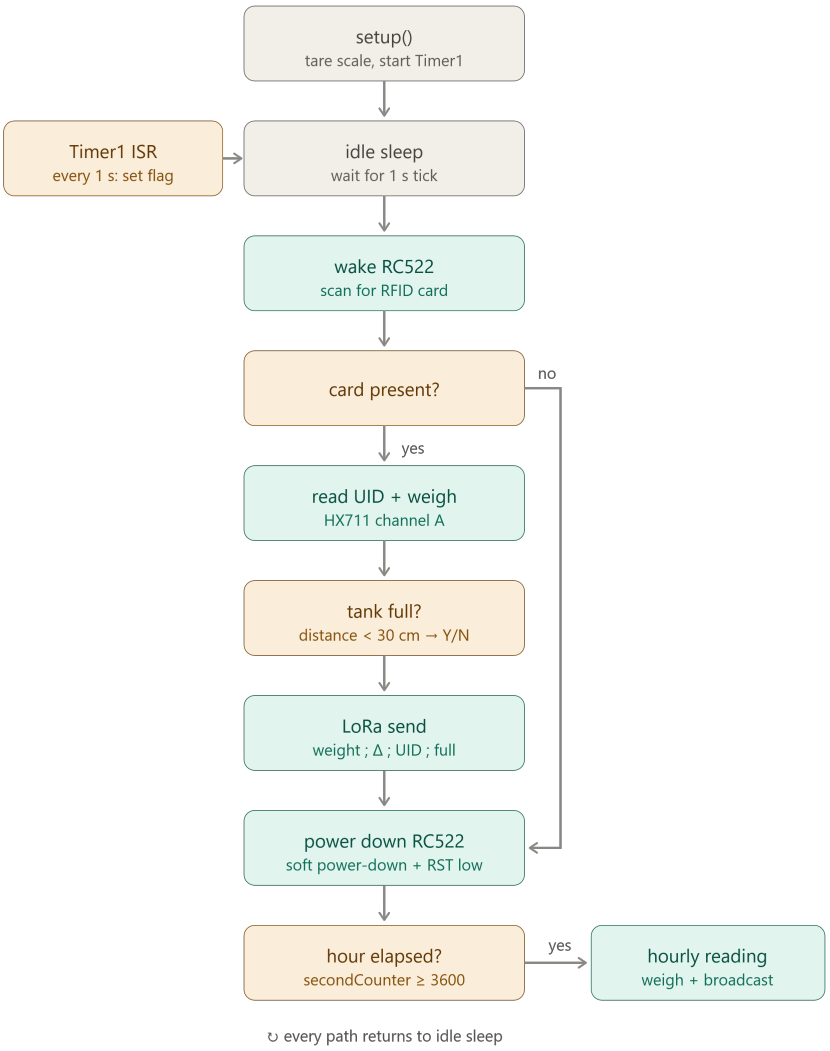


Figura 9: Flowchart do Codigo de Contentor

4 Interface Gráfica de Monitorização (Dashboard)

Para além da infraestrutura de hardware desenvolvida, o sistema integra uma aplicação de supervisão e controlo local, denominada *Entity Weight Dashboard* (Figura 10). Esta interface gráfica de utilizador (GUI) foi desenvolvida recorrendo a ferramentas de assistência baseadas em Inteligência Artificial para a aceleração do *layout* e estruturação do código, resultando num ambiente centralizado de monitorização em tempo real.Optámos por uma interface mais simples e direta, para demonstração.

Figura 10: Interface Gráfica do Utilizador (*Entity Weight Dashboard*).

A interface divide-se em quatro sub-sistemas principais de visualização:

- **Controlo de Conectividade:** Permite a seleção dinâmica da porta série (*Serial Port*) e configuração da taxa de transmissão (*Baud Rate*), gerindo o ciclo de vida da ligação ao hardware.
- **Estado da Capacidade:** Apresenta graficamente, através de um medidor radial (*gauge*), a percentagem de ocupação acumulada com base num limite máximo configurável (*Set Max Capacity*).
- **Histórico de Atividade:** Uma tabela estruturada que regista as métricas em tempo real, associando á identificação de cada contentor, o código do utilizador, o peso medido pelo sensor e a respetiva marca temporal.

- **Gráfico de Tendência:** Um gráfico para a análise do volume acumulado de peso ao longo do tempo.Este gráfico serve maioritariamente para estatisticas, podendo saber as horas de pico.

4.1 Protocolo de Comunicação com o Arduino

A comunicação entre o arduino e o computador e o Arduino Nano baseia-se num protocolo de comunicação por barramento série (UART via USB). O fluxo de dados opera a uma taxa modificável de 9600 bps (padrão de sincronismo estabelecido na GUI). O mecanismo de troca de dados processa-se através do seguinte fluxo operacional:

1. **Aquisição de dados:** O Arduino Nano amostra continuamente os dados de peso recolhidos e a identificação do utilizador. Estes dados são estruturados numa cadeia de caracteres (*string*) contínua, utilizando delimitadores específicos.

2. **Transmissão de Pacotes:** O microcontrolador envia periodicamente a *string* através da porta série nativa recorrendo à instrução `Serial.println()`.

3. **Descodificação de Pacotes(Software):** No lado do *dashboard*, uma *thread* secundária em segundo plano realiza a escuta da porta COM. Ao detetar o caractere de terminação (`\n`), a aplicação efetua a leitura do pacote, valida a integridade dos dados através de descodificadores específicos(delimitadores), e decompõe os campos individuais para atualizar os elementos da interface gráfica (tabela, gráfico e *gauge*).

4.2 Necessidade da Interface de Supervisão (Dashboard)

A integração da Interface na gestão de resíduos gera um registo de dados que, por si só, seriam interpretados de forma enificiente. A necessidade de desenvolver a aplicação *Entity Weight Dashboard* liga o hardware no contentor a um ponto central.

5 Dimensionamento energético

5.1 Alimentação do Projeto

Como este projeto não está a ser desenvolvido para estar ligado à rede elétrica, é necessário encontrar uma solução que permita mantê-lo funcional durante longos períodos de tempo. Assim, a abordagem mais adequada passa pela utilização de um painel solar em conjunto com uma bateria. Desta forma, é possível recarregar a bateria através do painel solar, mantendo o sistema operacional durante períodos prolongados.

5.1.1 Consumo Periféricos

Para caracterizar o painel e a bateria, é necessário saber o consumo do sistema, composto pelo sensor de peso (*load cell*), pelo sensor ultrassónico, pelo transmissor/recetor LoRa e pelo Arduino.

Componente	**Modo de operação**	**Tensão**	**Corrente**
Arduino Nano	Operação normal	5.0 V	19–31 mA
RYLR998	Transmissão (22 dBm)	3.3 V	144.7 mA
	Receção	3.3 V	17.5 mA
AJ-SR04M	Modo de espera	3.3 V	2.5 mA
	Medição	3.3 V	30 mA
HX711	Operação normal	3.3 V	13–15 mA
RFID RC522	Operação normal	3.3 V	13–26 mA
	Modo de espera	3.3 V	10–13 mA

Tabela 3: Resumo do consumo energético dos componentes

Com base nos valores obtidos nas *datasheets*, foi elaborada a tabela anterior. Estes valores foram considerados sem a aplicação de métodos de redução do consumo dos periféricos ou do Arduino. Para reduzir esse consumo, existem abordagens tanto por *hardware* como por *software*.

No caso do RYLR998, a melhor abordagem para reduzir o consumo passa pelo *software*, utilizando o *Sleep Mode*, que reduz a corrente consumida até 10 μA. Também é possível reduzir a potência do sinal, diminuindo assim a corrente de transmissão.

No sensor ultrassónico, a única abordagem é por *hardware*, introduzindo uma resistência na placa do sensor. Através do valor dessa resistência, é possível escolher o modo de funcionamento, reduzindo a corrente até 20 μA em modo de espera e até 2 mA durante a medição.

Como o Arduino consome entre 19 mA e 31 mA, é necessário encontrar uma forma de reduzir o consumo. A solução proposta consiste na implementação de um sistema que coloca o Arduino em *Sleep Mode*.

No caso do sensor de peso, foi identificado um problema: a redução da corrente de 13–15 mA para 1 μA implica manter o Arduino ligado para manter o pino em nível alto (*HIGH*). Isto faz com que o consumo permaneça elevado, ficando, no mínimo, entre 19 mA e 31 mA. Assim, concluiu-se que seria mais simples ligar o sensor apenas quando fosse necessário realizar uma leitura, permitindo que o Arduino alimente diretamente o sensor.

Componente	**Modo de operação**	**Tensão**	**Corrente**
Arduino Nano	Operação normal	5.0 V	19–31 mA
	Idle Mode	5.0 V	3.5–5 mA
RYLR998	Transmissão (14 dBm)	3.3 V	115.5 mA
	Receção	3.3 V	17.5 mA
	Sleep Mode	3.3 V	10 μA
AJ-SR04M R:47k	Modo de espera	3.3 V	20 μA
	Medição	3.3 V	2 mA
HX711	Operação normal	3.3 V	13–15 mA
	Sleep Mode	3.3 V	1 μA
RFID RC522	Operação normal	3.3 V	13–26 mA
	Modo de espera	3.3 V	10–13 mA
	Sleep Mode	3.3 V	80 μA

Tabela 4: Consumo energético dos componentes com técnicas eficientes

A partir dos tempos obtidos para a execução de cada componenete, conseguimos chegar aos valores médios do consumo de cada componente, facilitando assim a escolha da bateria.

Componente	**Modo de operação**	**Tempo de Execução**	**Corrente**
RYLR998	Transmissão (14 dBm)	20–40 ms	115.5 mA
	Receção	200 ms	17.5 mA
	Sleep Mode	–	10 μA
AJ-SR04M R:47k	Modo de espera	–	20 μA
	Medição	2.5 ms	2 mA
HX711	Operação normal	100 ms	13–15 mA
	Sleep Mode	–	1 μA
RFID RC522	Leitura	20–50 ms	13–26 mA
	Sleep Mode	–	80 μA
Arduino Nano	Operação normal	340–400 ms	19–31 mA
	Idle Mode	–	3.5–5 mA

Tabela 5: Tempo e Consumo de cada periférico

A partir destes valores é possivel determinar a consumo médio do sistema ao longo de um dia. Antes de ser definido tem de ser estabelicido que sempre que o sistema ativa-se este irá executar se em tempos diferentes dependendo se o cartão é lido ou não, no caso de não haver uma leitura do cartão o sistema só terá ativo entre 20–50ms(Tabela 5), sendo a grande parte do consumo do sistema nesta iteração o leitor RFID.

Já se um cartão for detetado na leitura, o sistema irá fazer todas as medições, isto já irá distribuir o consumo da corrente total pelos perfiféricos, isto aumentará o consumo da corrente, mas como esta não é a situação mais comum, o aumento não terá um efeito excessivo no consumo total.

Situação	**Consumo**	**Consumo Diário**
Sem Cartão	6.1 mAh	147.5 mAh/dia
Com Cartão	22.8 mAh	852 mAh/dia

Tabela 6: Consumo diário

5.1.2 Painel Solar e Bateria

A partir dos valores adquiridos anteriormente conseguimos procurar a bateria e o painel mais apropriado.

Capacidade da Bateria(mAh)	**Consumo Diário Base(mAh)**	**Duração(Dias)**
1100	147.5	7
2600	147.5	18
3300	147.5	22
5200	147.5	35

Tabela 7: Baterias e Duração

Dado que o modulo de carregamento da bateria precisa de ser alimentado a 5V, a escolha de paineis solares fica restringido aos de 5V para simplicidade do sistema.

Produção Painel Solar(W)	**Corrente Operacional(mA)**
1	200
2.5	500
4.5	900
5	1000

Tabela 8: Especificações do Painel Solar

6 Conclusões e Trabalho Futuro

6.1 Objetivos Concluídos

- Escolha dos sensores (células de carga, sensor de distância e módulo RFID) e da tecnologia de comunicação (LoRa).
- Criação de um código com rotinas de interrupção,e um sistema que mete os periféricos em modos de baixo consumo (Sleep/Idle Mode).
- Desenho, validação por DRC (Design Rules Check) e preparação de uma Placa de Circuito Impresso (PCB).
- Desenvolvimento da aplicação gráfica "Entity Weight Dashboard"para apresentação visual dos dados em tempo real.

6.2 Trabalho futuro

- Integração com a Rede LoRaWAN
- Melhorar a comunicação entre os dispositivos(as mensagens, adicionar reply e command system).

Referências

- **Tabela 1** (Tabela de sensores de peso) — fonte: tabela elaborada pela equipa. Imagens utilizadas:
  - *Strain gauge* — `figures/Strain_gauge.jpg`; fonte: Commons Wikimedia.
  - Sensor capacitivo — `figures/electrolytic-capacitor-inside.jpg`.
- **Tabela 2** (Tabela de sensores de deteção) — fonte: tabela elaborada pela equipa. Imagens utilizadas:
  - Sensor ultrassónico HC-SR04 — `figures/HRS04.jpg`.
  - Sensor infravermelho HW-201 — `figures/HW-201.jpg`.
  - Sensor de feixe — `figures/BreakBeam(1).jpg`.
  - Sensor LiDAR — `figures/LiDAR.jpg`.
- **Tabela 3** (Resumo do consumo energético dos componentes) — fonte: tabela elaborada pela equipa com base nos valores recolhidos nas *datasheets* dos componentes.
- **Tabela 4** (Consumo energético dos componentes com técnicas eficientes) — fonte: tabela elaborada pela equipa com base nos valores recolhidos nas *datasheets* dos componentes.
- **Tabela 5** (Tempo e consumo de cada periférico) — fonte: tabela elaborada pela equipa.
- **Tabela 6** (Consumo diário) — fonte: tabela elaborada pela equipa.
- **Tabela 7** (Baterias e duração) — fonte: tabela elaborada pela equipa.
- **Tabela 8** (Especificações dos Painéis Solares) — fonte: tabela elaborada pela equipa apartir de valores obtidos - fonte:ptrobotics.
- **Figura 1** (Connecções do Projeto) — fonte: Imagem gerada por IA.
- **Figura 2** (Diagrama de Blocos Inicial) — fonte: diagrama desenvolvido no website Canva.
- **Figura 3** (Diagrama de blocos) — fonte: diagrama desenvolvido no website Canva; ficheiro utilizado: `figures/Diagrama_Design_PCB.png`.

- **Figura 4** (Esquemático inicial do HX711) — fonte: Imagem retirada do KiCad.
- **Figura 5** (Esquemático Elétrico) — fonte: imagem retirada do Kicad.
- **Figura 6** (Esquemático inicial simplificado) — fonte: imagem retirada da renderização 3D do KiCad.
- **Figura 7** (Modelo 3D da placa simplificada) — fonte: imagem retirada da renderização 3D do KiCad.
- **Figura 8** (Sensor de peso) — fonte: sensor de peso desenvolvido no ISEL pelo colega José Perry. A fotografia foi tirada pela equipa do projeto, com edição de fundo realizada pelo *Gemini*; ficheiro utilizado: `figures/Sensor_Perry_Banner_mute.png`.
- **Figura 9** (Flowchart do Código de Contentor) — fonte: elaborado pela equipa.
- **Figura 10** (Interface Gráfica do Utilizador) — fonte: imagem retirada da dashboard usada.