



Universidade do Minho
Escola de Engenharia

Laboratórios de Telecomunicações e Informática II

ENGENHARIA DE TELECOMUNICAÇÕES E INFORMÁTICA
2020/2021

(Docentes: Bruno Daniel Mestre Viana Ribeiro, Vadym Serhiyovych Hapanchak,
José Augusto Afonso, Sérgio Adriano Fernandes Lopes)

29 de abril de 2021

Relatório

FASE A

Rui Filipe Ribeiro Freitas - a84121@alunos.uminho.pt

Sandro Teixeira Ribeiro – a85316@alunos.uminho.pt

Tiago João Pereira Ferreira - a85392@alunos.uminho.pt

Índice

Índice de figuras.....	4
Índice de tabelas	5
Lista de abreviaturas.....	6
Introdução.....	7
1. Planeamento do projeto	8
1.1. Planeamento temporal	8
1.2. Tecnologias/Ferramentas necessárias	9
1.2.1. Ao nível do <i>hardware</i>	9
1.2.2. Ao nível do <i>software</i>	10
2. Fundamentos	11
2.1. Sistema sensor	11
2.1.1. Sensor de movimento	11
2.1.2. Sensor de luminosidade	12
2.1.3. LED.....	13
2.2. Placa ESP32.....	14
2.3. Concentrador	14
3. Desenvolvimento	15
3.1. Arquitetura do sistema	15
3.1.1. Arquitetura de <i>hardware</i>	15
3.1.2. Arquitetura de <i>software</i>	16
3.2. Obtenção da luminosidade	18
4. Implementação	21
4.1. Sistema sensor	21
4.1.1. Código.....	21
4.1.2. Fluxograma.....	23
4.2. Concentrador	24
4.2.1. Código.....	24
4.2.2. Fluxograma	25
5. Testes e discussão de resultados	26
Conclusão.....	28
Referências	29
Autoavaliação	30

Índice de figuras

Figura 1 - Planeamento temporal tabela.....	8
Figura 2 - Planeamento temporal gráfico.....	8
Figura 3 - Esquema PIR HC-SR501.	11
Figura 4 - Esquema do divisor de tensão.....	12
Figura 5 - LDR (Light Dependent Resistor).....	12
Figura 6 - Circuito do LED.	13
Figura 7 - Esquema placa ESP32.	14
Figura 8 - Circuito do esquema geral.....	15
Figura 9 - Circuito Sistema Sensor.	15
Figura 10 - Trama START.....	16
Figura 11 - Trama STOP.	16
Figura 12 - Trama DATA LIGHT.	16
Figura 13 - Trama ERROR.....	17
Figura 14 - Trama DATA LDR.....	17
Figura 15 - Trama DATA MOV.....	17
Figura 16 - Divisor de tensão.	18
Figura 17 - Gráfico dos luxs em relação à resistência.	19
Figura 18 - Gráfico do log dos luxs em relação ao log das resistências.	20
Figura 19 - Variáveis globais.	21
Figura 20 - Função setup().	22
Figura 21 - Fluxograma ESP32.	23
Figura 22 - Interface do utilizador.	24
Figura 23 - Fluxograma do concentrador.	25
Figura 24 - Valores lidos no concentrador.	26
Figura 25 - Valores enviados pela placa ESP.	26
Figura 26 - Verificação de escrita no ficheiro log.	27

Índice de tabelas

Tabela 1 - Tecnologias ao nível do hardware.	9
Tabela 2 - Tecnologias ao nível do software.	10
Tabela 3 - Valores da resistência vs valores de luminosidade.	19

Lista de abreviaturas

LTI II – Laboratórios de Telecomunicações e Informática II

MQTT – *Message Queuing Telemetry Transport*

LED – Light-Emitting Diode

LDR – Light Dependent Resistor

IDE – Integrated Development Environment

CSV – Comma-Separated Values

PS – Porta *Socket*

FA – Frequência de amostragem

TA – Tempo de amostragem

TC – Tipo de comunicação

IESP – Identificador da placa ESP

TA – *Timestamp* das amostras

NTP – *Network Time Protocol*

Introdução

No âmbito da Unidade Curricular de LTI II (Laboratórios de Telecomunicações e Informática II) foi-nos proposto o desenvolvimento de um sistema de iluminação inteligente que permita monitorizar a ocupação e a luminosidade em diferentes áreas de uma residência, bem como controlar a luminosidade desses ambientes através do acionamento de lâmpadas.

Este projeto está dividido em 3 fases principais e uma fase extra. A fase A passa pelo planeamento, desenvolvimento e teste de todo o *hardware* e *software* que compõem o sistema necessários para a implementação dos sensores reais e a sua comunicação com os concentradores. Na fase B o objetivo é implementar um sistema central básico e o protocolo de comunicação entre este sistema e os concentradores. Em relação à fase C, muito resumidamente passa por completar o sistema central elaborado na fase B através de uma base de dados relacional e acrescentar funcionalidades de monitorização e administração que possamos achar necessárias. Na fase extra do projeto, que se trata de uma fase opcional é nos sugerida a implementação de funcionalidades mais avançadas através do uso do MQTT (*Message Queuing Telemetry Transport*). Cada fase do projeto terá uma entrega de um relatório de especificação passada uma semana do início dessa fase e uma demonstração acompanhada de um relatório de demonstração.

De modo a sermos capazes de cumprir com os objetivos deste projeto semestral devemos pôr em prática conhecimentos adquiridos noutras unidades curriculares, nomeadamente Redes de Computadores, Sistemas Operativos, Sistemas Distribuídos, Eletrónica e Laboratórios de Telecomunicações e Informática I.

1. Planeamento do projeto

1.1. Planeamento temporal

De modo que o grupo se mantenha focado no trabalho e com um compromisso para cumprir horários, resolvemos planejar as tarefas a fazer nesta fase. Na figura 1 observamos em forma de tabela os vários assuntos a ser tratados nesta fase com um período dado por nós para cumprir. Em relação à figura 2 demonstramos em forma de gráfico o tempo despendido nas várias tarefas. Ambas as figuras foram retiradas do programa *Gantt*.

		
Nome	Data de início	Data de fim
☐ Fase A	22-02-2021	07-04-2021
• Planeamento Temporal	22-02-2021	26-02-2021
• Definição da Arquitetura do Sistema	01-03-2021	01-03-2021
• Início da elaboração do relatório	02-03-2021	02-03-2021
• Ligações físicas entre o esp e sensores	04-03-2021	04-03-2021
• Desenvolvimento código esp	05-03-2021	15-03-2021
• Desenvolvimento código concentrador	18-03-2021	29-03-2021
• Realização de Testes	01-04-2021	02-04-2021
• Discussão dos resultados obtidos	05-04-2021	05-04-2021
• Conclusão e revisão do relatório	07-04-2021	07-04-2021

Figura 1 - Planeamento temporal tabela.

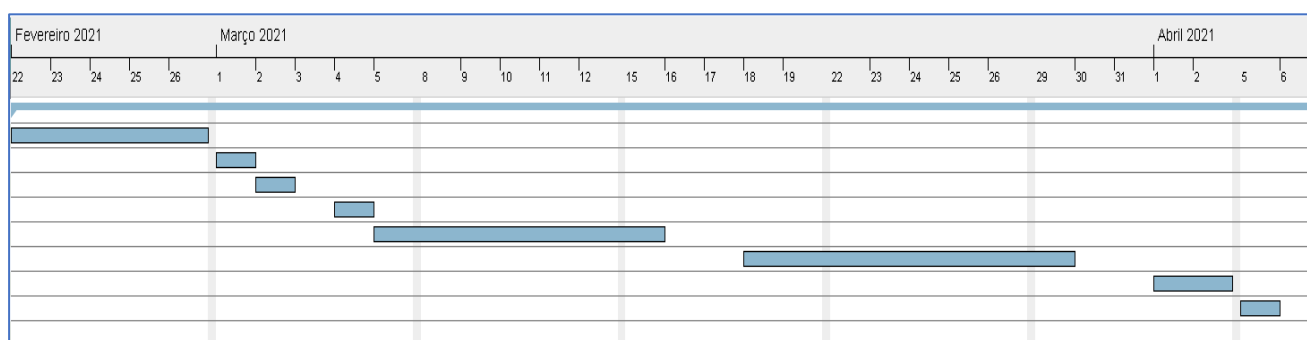


Figura 2 - Planeamento temporal gráfico.

1.2. Tecnologias/Ferramentas necessárias

Em qualquer projeto são necessárias certas tecnologias/ferramentas que nos facilitem o trabalho e ao mesmo tempo que aumentem a nossa produtividade. Para este trabalho em específico teremos de recorrer a tecnologias específicas tanto ao nível do *hardware* como ao nível do *software*, de modo a cumprir com os objetivos propostos.

1.2.1. Ao nível do *hardware*

No que toca à parte do *hardware* deste nosso trabalho as tecnologias por nós utilizadas estão presentes na tabela 1.









Tabela 1 - Tecnologias ao nível do *hardware*.

FERRAMENTA	QUANTIDADE	UTILIZAÇÃO
Computador Portátil	1	Desenvolvimento de código, elaboração do relatório, pesquisa e testes
Placa ESP32	1	Comunicação com os sensores e os LEDs
Sensor de movimento (PIR HC-SR501)	1	Deteção de movimento
Sensor de luminosidade (LDR 5...10kΩ)	1	Medição da luminosidade
LED	3	Simulação de lâmpadas
Cabo Micro USB	1	Ligações físicas entre o PC e a placa ESP32
Fios de ligação	Vários	Efetuar as ligações entre as placas ESP32 e os sensores e LEDs
Resistência	Várias	Limitar a corrente drenada pelos LEDs

1.2.2. Ao nível do *software*

Em relação ao *software* que irá ser utilizado na execução do nosso projeto este passará maioritariamente pelas aplicações apresentadas na tabela 2. Nesta realçamos as mais importantes e fundamentais na elaboração do trabalho.

Tabela 2 - Tecnologias ao nível do *software*.

LOGO	DESCRIÇÃO	UTILIZAÇÃO
	Microsoft Word	Realização e sincronização do relatório do projeto
	GanttProject	Planeamento do projeto através de diagramas de Gantt
	Arduino IDE	Software para a conexão com a placa ESP32 e comunicação com o mesmo
	Visual Studio Code	Execução e compilação de código
	GitHub	Sincronização do código da aplicação
	Visual Paradigm	Implementação de fluxogramas para a elaboração da aplicação e do projeto
	Google Drive	Sincronização de ficheiros essenciais à realização do trabalho
	Discord	Comunicação entre o grupo

2. Fundamentos

2.1. Sistema sensor

O sistema sensor que temos de implementar na elaboração deste projeto é constituído por uma placa ESP32 que comunica através de Wi-Fi ou Bluetooth com um concentrador. Esta ligação é importante para a aquisição de dados por parte do concentrador através de uma conexão sem fios fiável. Ligados ao dispositivo de RF estão conectados sensores de movimento e de luminosidade e LEDs através de cabos devidamente conectados. Todos os elementos que constituem o sistema sensor serão explicados de seguida.

2.1.1. Sensor de movimento

O sensor de movimento utilizado pelo nosso grupo foi o recomendado pelos docentes, o PIR HC-SR501. Este sensor tem como principal função a deteção de movimento. Na placa é possível alterar os valores de tempo e de sensibilidade do sensor através de dois potenciómetros. O valor de tempo altera o tempo que passa até o *output* passar de HIGH para LOW após a deteção de movimento. O mínimo é 3 segundos após deteção de movimento, pelo que este sensor não é o mais eficaz se quisermos um sensor que funcione em tempo real. Em relação à sensibilidade, este serve para aumentar ou diminuir a distância de deteção, que varia entre os 3 e os 7 metros [1].

Na figura 3 podemos observar o *pinout* do sensor de movimento, onde observamos os potenciómetros que nos permitem ajustar os parâmetros enunciados anteriormente, assim como 2 formas de disparo, único ou repetido, em que no modo único o sensor mantém-se HIGH até deixar de detetar movimento e no modo repetido, em que o sensor se mantém HIGH até o tempo definido pelo potenciómetro que ajusta o tempo ser atingido. Na elaboração do nosso projeto iremos utilizar o modo de disparo repetido. Para além destes parâmetros observamos na imagem 3 pinos que vamos ligar à placa ESP32, o 3.3 V de modo a fornecer energia ao sensor, o SIGNAL que liga o sensor à placa e o GND, cujos pontos são identificadas na figura seguinte.

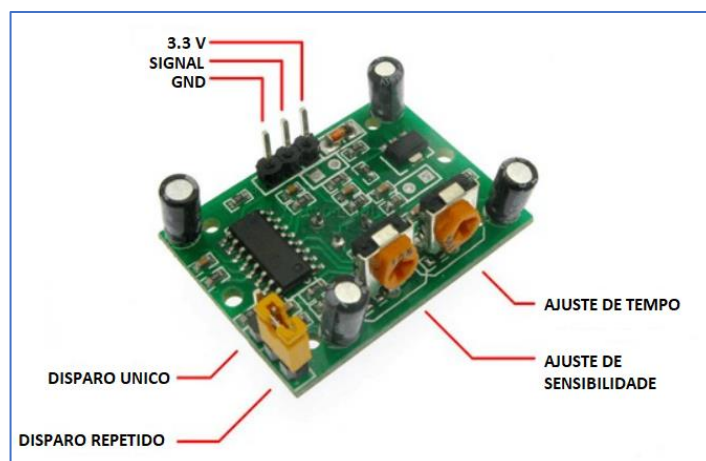


Figura 3 - Esquema PIR HC-SR501.

2.1.2. Sensor de luminosidade

O sensor de luminosidade escolhido pelo grupo foi um LDR (Light Dependent Resistor) com resistência na luz de 5-10k Ω para 10 lux. Um dos materiais presentes é um semicondutor de alta resistência e, como mostra a figura 5, possui apenas dois terminais e não tem polaridade.

Este sensor é uma resistência, a qual varia conforme a intensidade de luz que incide sobre ele. Quando temos pouca luminosidade a resistência terá um valor elevado e quanto maior for a luminosidade menor será a resistência. O sensor LDR pode apresentar um atraso até 10 milissegundos para detectar a mudança entre pouca e muito luminosidade, ou seja, entre um ambiente escuro e um ambiente claro [2].

Na montagem do hardware foi necessária a utilização de uma resistência de 1k Ω junto ao sensor LDR para preservar o bom funcionamento deste e o acesso a uma gama de valores mais ampla pelo ADC. Na figura 4 podemos observar o esquema a que nos baseamos para obter o valor da resistência em que o valor apresentado no V_{out} corresponde à tensão entre a resistência e o LDR.

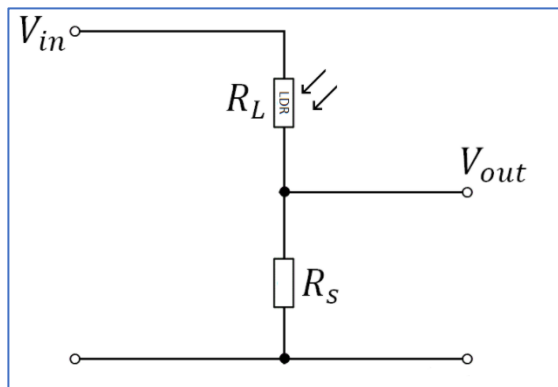


Figura 4 - Esquema do divisor de tensão.

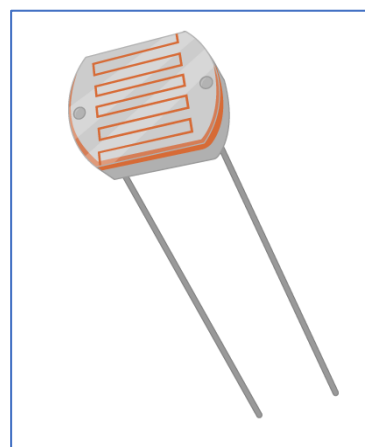


Figura 5 - LDR (Light Dependent Resistor).

2.1.3. LED

Na elaboração deste projeto temos por objetivo utilizar 2 LEDs, no entanto como ainda estamos numa fase inicial poderemos vir a utilizar mais ou menos. A nossa perspectiva é utilizar um LED para servir como lâmpada em que podemos acender sempre que for conveniente, outro é para caso o sensor de movimento detete movimento o LED acende e apaga caso não seja detetado mais movimento, bem como para efeitos de iluminação caso o sensor de luminosidade atinja valores perto de 0 indicando que a área se encontra em escuridão. De modo a acender o LED é necessário que por ele passe corrente. No entanto é necessário controlar a quantidade de corrente e para isso utilizamos uma resistência entre o LED e a fonte de alimentação. O valor desta resistência foi calculado através da equação 1. Através da observação da figura 6 e da equação retiramos vários valores dependendo dos LEDs usados pois a tensão do LED varia com a cor. Pelas contas efetuadas pelo grupo a resistência que devíamos utilizar seria por volta dos 100 Ω (2). Como o recurso a resistências era limitado decidimos escolher uma resistência de 470 Ω das disponíveis visto acharmos não haver grande problema ter uma resistência um pouco superior. Na expressão seguinte vemos o cálculo que usamos no cálculo da resistência com resistência (R) igual à tensão do sistema (SV) menos a tensão do LED (FV) sobre a corrente (I).

$$R = \frac{SV - FV}{I} \quad (1)$$

$$R = \frac{3.3 - 1.8}{15\text{mA}} = 100 \Omega \quad (2)$$

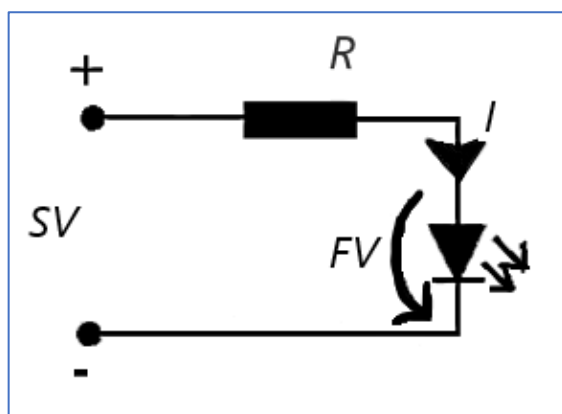


Figura 6 - Circuito do LED.

Com uma resistência de 470 Ω obtivemos uma corrente de aproximadamente 3 mA (3). Não é o desejado, no entanto visto estarmos a utilizar uma resistência superior era de esperar estes valores. Relativamente aos valores lidos pelo multímetro estes deram por volta dos 2.8 mA que era o espectável relativamente aos cálculos.

$$I = \frac{3.3 - 1.8}{470} = 3.1 \text{ mA} \quad (3)$$

2.2. Placa ESP32

A placa ESP32 foi escolhida pelo nosso grupo para fazer a comunicação entre o concentrador e o sistema sensor visto já estarmos habituados com a placa devido ao uso da mesma na cadeira de LTI I. Outras vantagens de usarmos esta placa é o seu custo reduzido e o facto de existir um IDE próprio que nos permita elaborar código para o dispositivo, o Arduino IDE. De seguida apresentamos o *pinout* desta placa em que os pinos que iremos utilizar com maior frequência são os 3.3 V, o GND e os pinos de contacto com os sensores, no nosso caso o 13 para a simulação de uma lâmpada, o 18 para o sensor de movimento, o 25 para o LED do sensor e o 32 para o LDR.

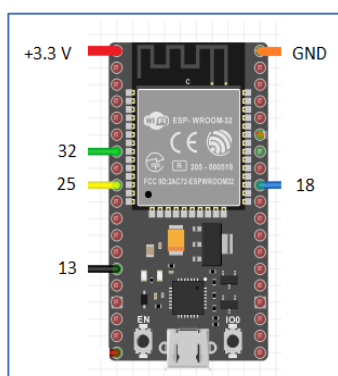


Figura 7 - Esquema placa ESP32.

2.3. Concentrador

O concentrador refere-se a um programa de computador que trata do processo de recolha de informação proveniente das placas ESP32 sobre o estado dos sensores e execução de certas tarefas como ativar lâmpadas a pedido do utilizador. A implementação deste programa será efetuada em linguagem C visto estarmos mais habituados quando comparado com C++. Para o concentrador é importante a configuração deste e a manutenção de dados importantes para o utilizador. Por isso mesmo iremos utilizar ficheiros CSV (Comma-Separated Values) com estes objetivos [3].

No que toca ao ficheiro de configuração do concentrador irá ter campos como a porta a utilizar para o *socket* (PS), frequência de amostragem (FA), tempo de amostragem (TA) e o tipo de comunicação (TC).

Relativamente aos ficheiros *log* vamos utilizar tal como o ficheiro de configuração o formato CSV na sua execução. Iremos utilizar 5 ficheiros distintos, um para guardar o estado do concentrador, ou seja, quando começa e quando acaba o seu funcionamento e o *timestamp* de quando ocorreram. Um segundo ficheiro para guardar as amostras periódicas de luminosidade, um terceiro para as amostras de movimento, um quarto para os erros e outro para os clientes que estão conectados ao concentrador e a área onde se encontram.

3.Desenvolvimento

3.1. Arquitetura do sistema

3.1.1. Arquitetura de *hardware*

A arquitetura de *hardware* do sistema representa a estrutura dos componentes utilizados na elaboração do projeto e a implementação dos mesmos através de ligações físicas. O sistema que vamos organizar tem como principais componentes o PC onde estão implementados o concentrador e o sistema sensor que envia e recebe informações dos vários sensores e LEDs. Na figura 8 apresentamos um esquema que traduz todo o processo elaborado no que diz respeito ao *hardware* por nós utilizado. A ligação entre o concentrador e o sistema sensor será efetuado através de Wi-Fi. O nosso grupo decidiu escolher esta forma de ligação pois achamos ser uma tecnologia recente e em constante desenvolvimento como o *Bluetooth*, mas uma tecnologia mais rápida assim como o facto de na nossa pesquisa termos encontrado mais informações sobre o mesmo.

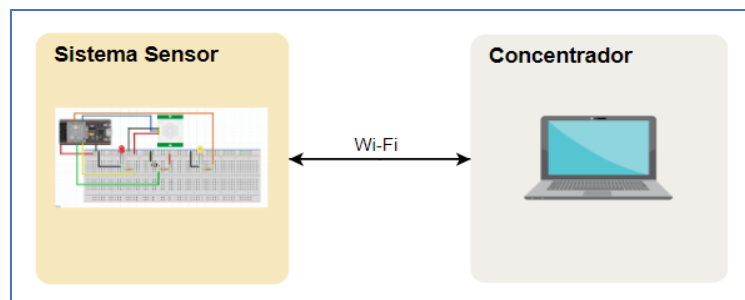


Figura 8 - Circuito do esquema geral.

Na figura 9 apresentamos mais detalhadamente o esquema do sistema sensor composto pela placa ESP32 e pelos componentes que caracterizam o sistema sensor. Salientamos as 2 resistências de 470 Ω para os LEDs e uma de 1 k Ω para o sensor de movimento. Na figura observamos também o sensor de movimento (PIR) conectado à porta 18 do ESP e o sensor de temperatura (LDR) à porta 32.

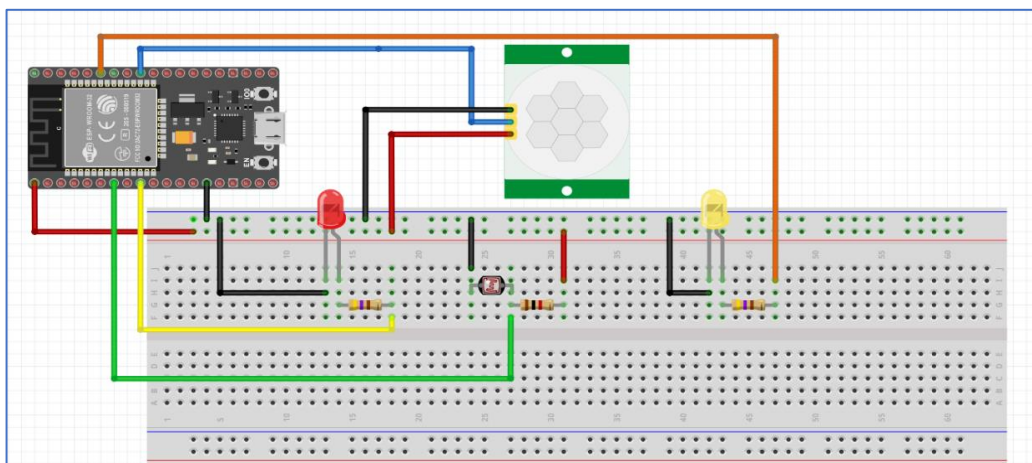


Figura 9 - Circuito Sistema Sensor.

3.1.2. Arquitetura de *software*

No que toca à arquitetura de *software* esta passa pela discussão de certas medidas a tomar relativamente ao protocolo de comunicação entre o concentrador e a placa ESP32. Decidimos utilizar diversas tramas de dados, todas elas partilham os campos tipo, que identifica o tipo da trama, iss, que identifica o sistema sensor e o *timestamp* com o tempo e data exata do envio. Para além disso contêm campos para o funcionamento eficiente do nosso projeto.

- **START** – Na trama *START* é enviada a informação do concentrador para o sistema sensor para este começar o seu funcionamento, ou seja, diz ao sistema sensor que este pode começar a recolher e a enviar amostras para o concentrador. Esta trama contém os campos apresentados na figura 10.

Tipo (1 byte)	ISS (1 byte)	Timestamp (4 bytes)	Sample Time (1 byte)	Sample Frequency (1 byte)
------------------	-----------------	------------------------	----------------------------	---------------------------------

Figura 10 - Trama *START*.

- **STOP** – Relativamente à trama *STOP* é enviada a informação para o sistema sensor cessar o seu funcionamento, ou seja, diz ao sistema sensor para este não enviar mais amostras e para parar de as recolher. Esta trama contém os campos apresentados na figura 11.

Tipo (1 byte)	ISS (1 byte)	Timestamp (4 bytes)	Stop Reason (1 byte)
------------------	-----------------	------------------------	-------------------------

Figura 11 - Trama *STOP*.

- **DATA LIGHT** – Na trama *DATA LIGHT* é enviado do concentrador para a placa ESP32 uma mensagem para que esta acenda/desliga a lâmpada do sistema sensor de acordo com o valor enviado no campo *LED STATE*. Esta trama contém os campos apresentados na figura 12.

Tipo (1 byte)	ISS (1 byte)	Timestamp (4 bytes)	Led State (1 byte)
------------------	-----------------	------------------------	-----------------------

Figura 12 - Trama *DATA LIGHT*.

- **ERROR** – A trama *ERROR* é enviada da placa ESP32 para o concentrador a indicar que houve um erro no sistema. Esta trama contém os campos apresentados na figura 13.

Tipo (1 byte)	ISS (1 byte)	Timestamp (4 bytes)	Error Type (1 byte)
------------------	-----------------	------------------------	------------------------

Figura 13 - Trama *ERROR*.

- **DATA LDR** – A trama *DATA LDR* é enviada pela placa para o concentrador com informações sobre o sensor de luminosidade e da lâmpada que controla a luminosidade e movimento no local onde se encontra. Os campos são os apresentados a seguir.

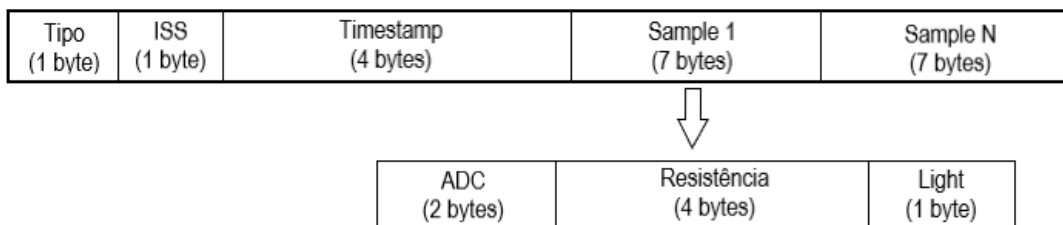


Figura 14 - Trama *DATA LDR*.

- **DATA MOV** – A trama *DATA MOV* é enviada para o concentrador quando há detecção de movimento por parte do sensor. Este aviso é enviado através desta trama com os seguintes campos.

Tipo (1 byte)	ISS (1 byte)	Timestamp (4 bytes)	Mov State (1 byte)
------------------	-----------------	------------------------	-----------------------

Figura 15 - Trama *DATA MOV*.

3.2. Obtenção da luminosidade

De modo a obtermos valores reais de luminosidade no LDR tivemos de arranjar uma forma de calcular a densidade luminosa por unidade de área, luxs. Com o objetivo de chegar a tais valores enviamos o valor medido pelo ADC e a resistência no LDR do Arduino para o concentrador onde passamos para luxs. O valor do ADC é obtido através do método *analogRead()* no Arduino enquanto que o da resistência é através da fórmula do divisor de tensão (4) representado na figura 16 . Para passar o valor obtido para tensão utilizamos uma regra de 3 simples, a equação (5). Colocando a resistência do LDR em evidência obtemos a fórmula que usamos no código (6).

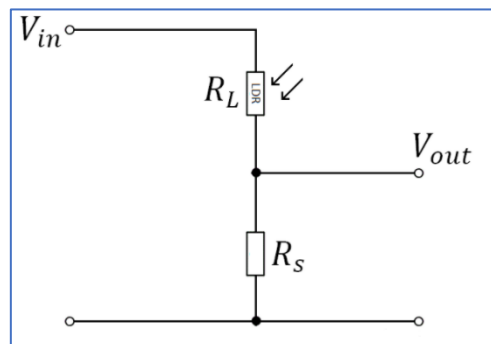


Figura 16 - Divisor de tensão.

$$V_{out} = \frac{R_S}{R_L + R_S} * 3.3 \quad (4)$$

$$V_{out} = \frac{ADC * 3.3}{4095} \quad (5)$$

$$R_L = \frac{(3.3 - V_{out})}{V_{out} * R_S} \quad (6)$$

De modo a passar para luxs, como não tínhamos acesso a um *datasheet* fiável optamos por fazer a caraterização do sensor. Para isso baseamo-nos num projeto elaborado por David Williams intitulado “Design a Luxmeter Using a Light Dependent Resistor” [6].

Para caraterizar o sensor coletamos vários valores de resistência para vários valores de lux de modo a podermos observar a curva caraterística do sensor através de gráficos apresentados na página seguinte.

Na tabela 3 apresentamos os valores da resistência do LDR, retirados com o auxílio do multímetro na primeira coluna e os valores em lux retirados de uma aplicação móvel na segunda coluna com o objetivo de visualizar a relação entre eles.

Tabela 3 - Valores da resistência vs valores de luminosidade.

Resistência LDR (Ω)	Luminosidade (lux)
142000	1
39000	6
12550	11
9500	18
5500	40
4350	73
3050	80
2330	128
1575	310
1440	343
1280	510
1150	560
930	680
823	740
805	970
769	1544
463	3075
313	3244
270	3808

O gráfico da figura 17 é feito a partir da relação entre os luxs e a resistência da tabela anterior. Através deste gráfico conseguimos visualizar que a resistência diminui exponencialmente com o aumento da luminosidade.



Figura 17 - Gráfico dos luxs em relação à resistência.

Com estes valores elaboramos o gráfico da figura 18 que relaciona os logaritmos dos luxs com os logaritmos da resistência através da fórmula 7.

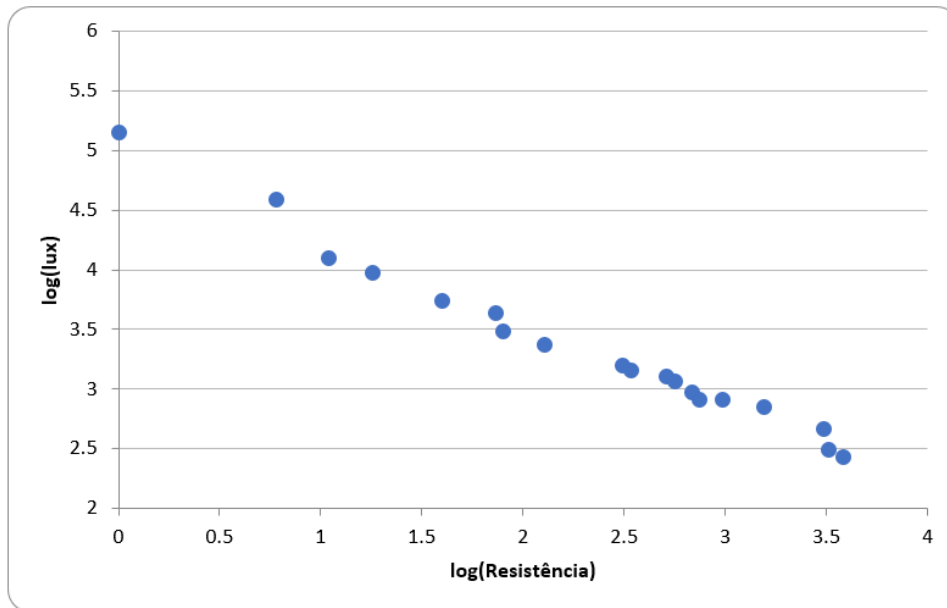


Figura 18 - Gráfico do log dos luxs em relação ao log das resistências.

$$\log_{10}(lux) = m * \log_{10}(Resistência) + b \quad (7)$$

Como o objetivo é obter os valores de luxs chegamos à equação 8.

$$lux = b * R^m \quad (8)$$

No nosso caso em particular a fórmula calculada foi a apresentada em 9, esta que foi colocada no concentrador para o cálculo dos luxs.

$$lux = 9594799.679 * R^{-1.397610133} \quad (9)$$

4. Implementação

4.1. Sistema sensor

Relativamente à implementação do sistema sensor primeiramente decidimos elaborar o circuito necessário para conectar os sensores com a placa ESP32. Decidimos utilizar as portas 16, 25, 18 e 32 para conectar o LED que simula uma lâmpada, o LED que controla a luminosidade/movimento da área sendo este ativado quando são atingidos valores específicos, o sensor de movimento e o sensor de luminosidade respetivamente. Para ligar os LEDs e o LDR foi necessário o dimensionamento de resistências para limitar a corrente drenada.

4.1.1. Código

O código elaborado para o sistema sensor foi realizado em C com o auxílio do ambiente de desenvolvimento Arduino IDE. Para a implementação do código no Arduino foram utilizadas 2 bibliotecas particulares, a biblioteca <NTPClient.h> [4] que foi utilizada na implementação do protocolo NTP que nos permite obter o *timestamp* através de WiFi e a biblioteca <WiFi.h> [5] que nos permite utilizar *sockets* para a comunicação entre a placa ESP32 e o concentrador.

Numa fase inicial do código foram definidas algumas variáveis globais presentes na figura 19, o *timeSeconds* corresponde ao tempo entre medições do sensor de movimento. O *MAX* corresponde ao tamanho máximo de um pacote periódico que no nosso caso decidimos utilizar 512 bytes que corresponde a um máximo de aproximadamente 72 amostras por pacote. *MAXMOV* corresponde também ao tamanho máximo, mas neste caso ao do pacote de movimento que é enviado quando o sensor é ativado. Outra variável que controla o tamanho máximo de um pacote que decidimos implementar foi o *DATAPACKETEMPTY* que diz respeito ao tamanho do pacote periódico sem as amostras inseridas. Finalmente temos o identificador de sistema sensor da placa utilizada, o *ISS*.

```
#define timeSeconds 2.5
#define MAX 512
#define MAXMOV 7
#define ISS 1
#define DATAPACKETEMPTY 6
```

Figura 19 - Variáveis globais.

De seguida inicializamos variáveis globais como os pinos de conexão aos sensores e LEDs e os *arrays* que utilizamos para os pacotes periódicos e de movimento. Após isso na função *setup()* é aberta a porta série com um *baudrate* de 115200, é feita a conexão entre a placa e o WiFi, a configuração do interruptor para o sensor de movimento e a inicialização dos LEDs como demonstrado na figura 20.

```
void setup() {  
  Serial.begin(115200);  
  delay(100);  
  WiFi.begin(ssid, password);  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.println("...");  
  }  
  timeClient.begin();  
  //timeClient.setTimeOffset(3600);  
  pinMode(motionSensor, INPUT_PULLUP);  
  attachInterrupt(digitalPinToInterrupt(motionSensor), detectsMovement, RISING);  
  pinMode(light, OUTPUT);  
  pinMode(led, OUTPUT);  
  digitalWrite(light, LOW);  
  digitalWrite(led, LOW);  
}
```

Figura 20 - Função *setup()*.

Na função *loop()*, que é considerada a função *main* do código e está constantemente à escuta por tramas vindas do concentrador começamos por conectar com sucesso a placa ao socket através do método *connect* da biblioteca <WiFi.h>. Após isso a placa espera pela chegada do pacote *start* para poder começar a recolher dados e através dos parâmetros recebidos envia o pacote periódico de x em x tempo. Relativamente ao sensor de movimento, quando é detetado movimento é enviado um pacote que contém apenas o identificador do sistema sensor, o *timestamp* e o valor 1 se foi detetado. De modo a simplificar o código criamos várias funções para diferentes tarefas como a função *createDATA()* onde criamos o pacote periódico sem as amostras visto este ter um tamanho pré-definido. Para adicionar as amostras ao pacote utilizamos a função *insertValuesDataPacket()*. Para receber os vários tipos de tramas também decidimos utilizar diferentes funções, a função *getStart()* para receber a trama *START*, a função *getStop()* para a trama *STOP* e a função *getLight()* para receber a trama *LIGHT* que controla o estado da lâmpada. Para além destas funções de receber pacotes também temos 2 para a criação de pacotes, a função *samplesMov()* que cria o pacote de movimento e a função *samplesLDR()* o pacote periódico.

4.1.2. Fluxograma

Na figura seguinte apresentamos o fluxograma relativo ao código realizado no Arduino IDE onde resumidamente o programa começa com a configuração dos sensores e conecta-se ao socket. Após isso fica à espera de uma trama para começar a recolher, parar ou acender/desligar uma lâmpada.

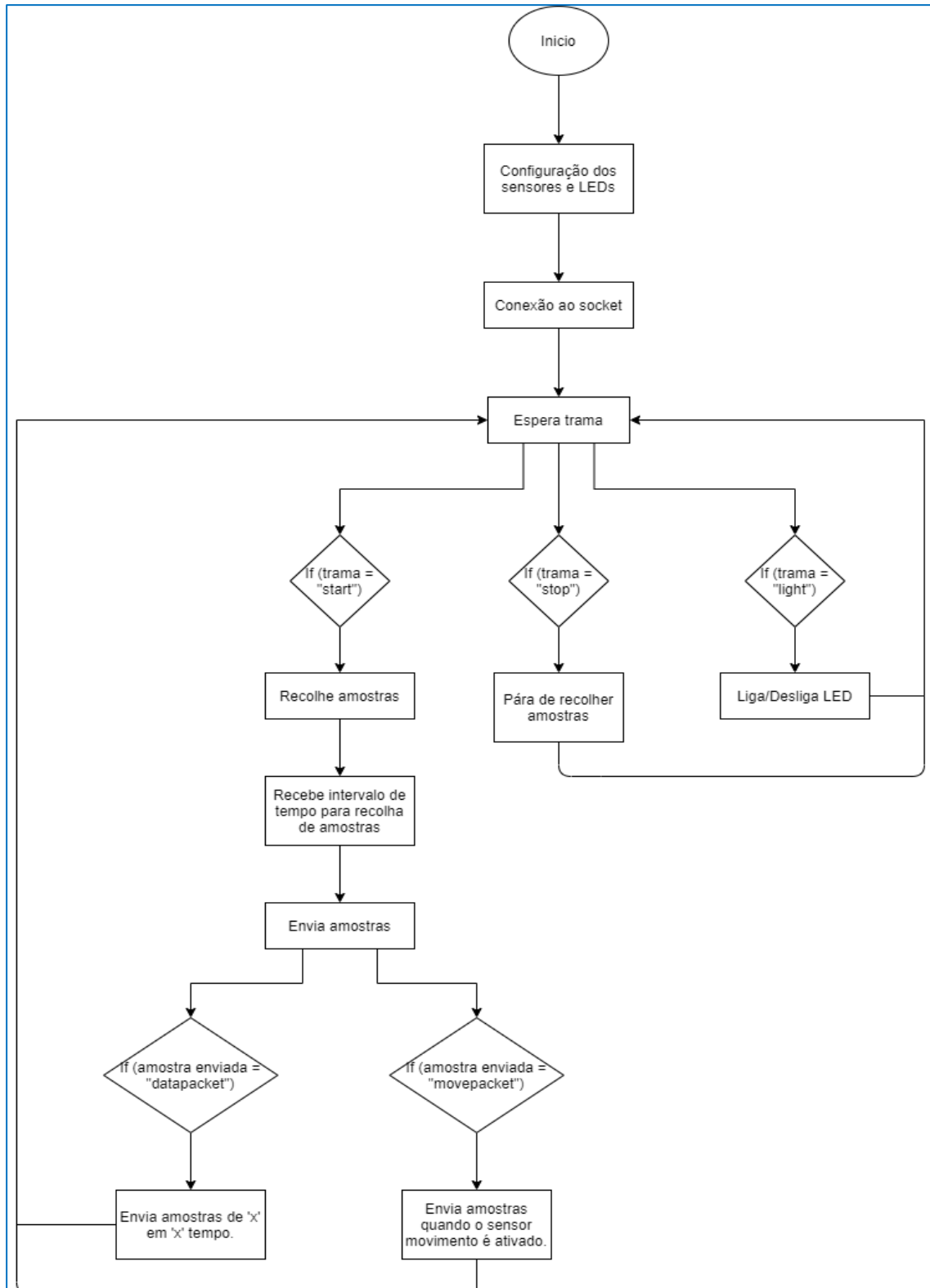


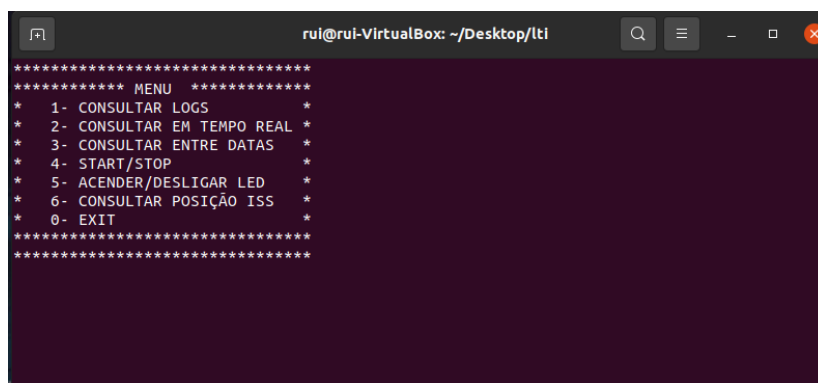
Figura 21 - Fluxograma ESP32.

4.2. Concentrador

4.2.1. Código

Para programar o concentrador utilizamos a linguagem C. Para além das bibliotecas normalmente utilizadas usamos algumas em específico, a biblioteca `<sys/socket.h>` para fazer a conexão com o ESP32 através de sockets, usamos também a `<pthread.h>` que nos permitia usar múltiplas *threads*, uma com o intuito de lidar com a interface do utilizador e outra comunicar com o ESP32.

No inico do código são carregados os valores de configuração do concentrador, através da função `getConfigFile()`, onde é retirada a porta a usar na comunicação com o ESP32, a frequência de envio de amostras e o intervalo de tempo de recolha das mesmas. Depois de carregados os valores é aberto o *socket* para a comunicação com o ESP32 e apresentada a interface ao utilizador (figura 22).



```
***** MENU *****
* 1- CONSULTAR LOGS *
* 2- CONSULTAR EM TEMPO REAL *
* 3- CONSULTAR ENTRE DATAS *
* 4- START/STOP *
* 5- ACENDER/DESLIGAR LED *
* 6- CONSULTAR POSIÇÃO ISS *
* 0- EXIT *
*****
```

Figura 22 - Interface do utilizador.

A *thread* onde se encontra a comunicação fica à espera que o utilizador interaja com a interface. Se este selecionar a opção para começar a recolher amostras, é criado o pacote *start* através da função `creatStartPacket()` e este é enviado. Caso selecione a opção para parar de recolher amostras é criado o pacote *stop* através da função `creatStopPacket()` e este é enviado. Este processo é idêntico para a opção de acender/desligar o led, onde é criado o pacote *light* através da função `creatLightPacket()`.

A mesma *thread* fica à “escuta” de informação recebida pela placa ESP32. Ao receber informação verifica o tipo do pacote e consoante o mesmo tem diferentes comportamentos. Se o tipo de pacote for 2(DATA LDR) é retirados os valores das amostras e guardados no ficheiro `logSamples.csv`, se o tipo for 3(DATA MOV) é retirado o valor da amostra e guardado no ficheiro `logSamplesMov.csv`, por fim se o tipo for 5(ERROR) é guardado no ficheiro `logError.csv` o valor do erro.

4.2.2. Fluxograma

Na figura seguinte apresentamos o fluxograma relativo ao código do concentrador. Este começa com o carregamento do ficheiro de configuração e criação e abertura do *socket*. Após isso apresenta a interface ao utilizador e espera o envio ou receção de dados com o ESP32.

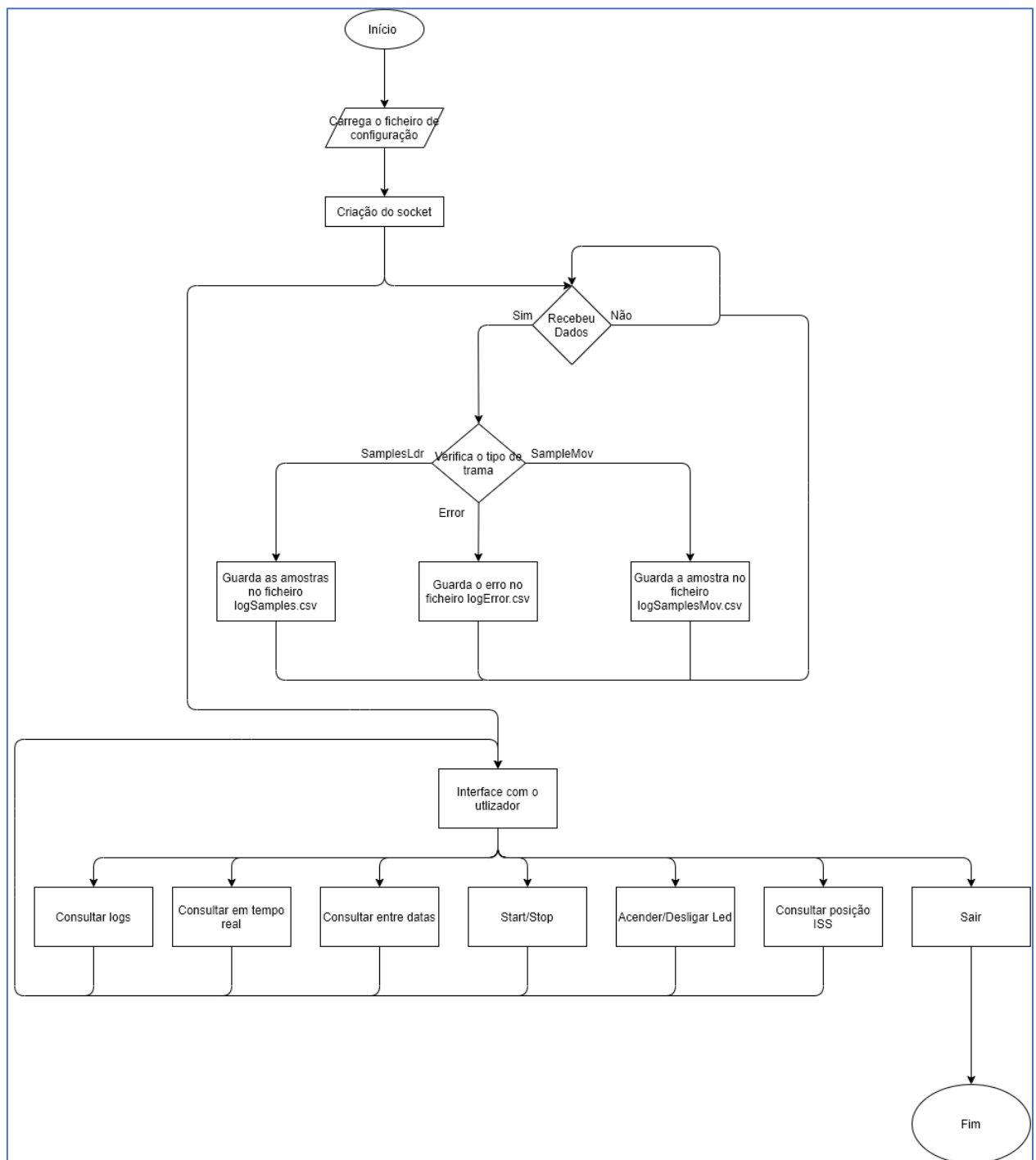


Figura 23 - Fluxograma do concentrador.

5. Testes e discussão de resultados

Após o desenvolvimento e implementação do nosso programa tanto a nível de *hardware* como *software* tivemos de realizar vários testes para ver se tudo funcionava como era desejado, ou seja, se os dados gerados pelos sensores eram enviados com sucesso por parte da placa ESP e recebidos eficazmente no concentrador, garantindo fiabilidade na comunicação do sistema.

Na figura 25 demonstramos os valores retirados do Arduino antes do envio para o concentrador. Na figura podemos observar o valor do ADC, a resistência do LDR, a tensão do LDR, o valor dos luxs e da lâmpada. Estes valores exceto os luxs são enviados para o concentrador. Os valores recebidos por parte do concentrador são apresentados na figura 24. O campo ISS corresponde ao sistema sensor que está a enviar os dados e o valor dos luxs são calculados no concentrador. Os restantes valores são os recebidos do sistema sensor.

```
-> *****
-> Trama Enviada
-> ADC :1001
-> Resistencia LDR:3090.91
-> Tensão LDR: 2.49
-> Lux: 127.93
-> LAMPADA: 0
-> *****
-> Trama Enviada
-> ADC :986
-> Resistencia LDR:3153.14
-> Tensão LDR: 2.51
-> Lux: 124.41
-> LAMPADA: 0
```

Figura 25 - Valores enviados pela placa ESP.

```
-----
ISS: 1
ADC: 1001
Resistencia LDR: 3090.91
Tensão LDR: 2.49
LUX: 127.9
LAMPADA: 0
Luz apagada
-----

-----
ISS: 1
ADC: 986
Resistencia LDR: 3153.14
Tensão LDR: 2.51
LUX: 124.4
LAMPADA: 0
Luz apagada
-----
```

Figura 24 - Valores lidos no concentrador.

Estes valores recebidos pelo concentrador são guardados num ficheiro *log* com os campos: ISS (Identificação do sistema sensor), valor do ADC, o valor da tensão no LDR, o valor da resistência do LDR, o valor dos luxs, o estado da lâmpada e o *timestamp*. Os valores recebidos pelo concentrador da imagem 24 estão guardados no ficheiro como demonstrado na figura 26.

→	1	1083	2.43	2781.16	148.3	0	Wed Apr 28 21:31:06 2021
→	1	1038	2.46	2945.09	136.9	0	Wed Apr 28 21:31:06 2021
→	1	1001	2.49	3090.91	127.9	0	Wed Apr 28 21:31:12 2021
→	1	986	2.51	3153.14	124.4	0	Wed Apr 28 21:31:12 2021
	1	1139	2.38	2595.26	163.4	0	Wed Apr 28 21:31:27 2021
	1	1098	2.42	2729.51	152.3	0	Wed Apr 28 21:31:27 2021

Figura 26 - Verificação de escrita no ficheiro log.

Após termos realizados vários testes tanto para testar a comunicação entre a placa e o concentrador como a medição da luminosidade sentimos que chegamos a um resultado satisfatório na realização do projeto. Os valores obtidos do LDR mostraram-se dentro da gama aceitável comparados com valores medidos no multímetro, no entanto como não eram valores exatos e tinham um erro de aproximadamente 10% no valor da tensão, o cálculo tanto da resistência como dos luxs não foi o melhor possível.

Relativamente aos valores medidos pelos luxs quando o ambiente era pouco iluminado, ou seja, quando os luxs eram baixos o nosso sensor media com uma precisão desejável. No entanto quando incidia muita luz no sensor e os valores de lux eram elevados a precisão não era a melhor. Achamos que isto se possa dever ao facto de não termos retirado valores suficientes de amostras para valores elevados de luminosidade.

Conclusão

Após a conclusão desta fase A podemos afirmar que nos sentimos satisfeitos com o resultado e com o nosso desempenho uma vez que cumprimos com os objetivos propostos. O facto de termos elaborado um planeamento antes da realização do projeto ajudou na coordenação e organização do que cada elemento do grupo tinha de fazer.

Relativamente às dificuldades que foram surgindo ao longo da realização desta fase estas foram ultrapassadas com sucesso apesar do desconhecimento inicial de alguns recursos e tecnologias.

A comunicação entre o sistema sensor e o concentrador que era a parte mais crucial desta primeira fase foi implementada com sucesso assim como a transferência dos valores medidos pelos sensores. Na próxima fase já iremos passar para o desenvolvimento dos sistemas sensores simulados assim como a criação de um sistema central que comunica com os vários concentradores.

Referências

- [1] Last Minute Engineers, <https://lastminuteengineers.com/pir-sensor-arduino-tutorial/>.
- [2] <http://mundoprojetado.com.br/ldr-o-que-e-e-como-funciona/>
- [3] <https://www.geeksforgeeks.org/relational-database-from-csv-files-in-c/>
- [4] Taranais, <https://github.com/taranais/NTPClient>
- [5] Arduino, <https://www.arduino.cc/en/Reference/WiFi>
- [6] David Williams, <https://www.allaboutcircuits.com/projects/design-a-luxmeter-using-a-light-dependent-resistor/>

Autoavaliação

Rui Filipe Ribeiro Freitas:

Nesta primeira fase do projeto ajudei no desenvolvimento do código do concentrador e do sistema sensor e na pesquisa e elaboração do relatório.

Sandro Teixeira Ribeiro:

Nesta fase ajudei na pesquisa teórica, escrita e revisão do relatório.

Tiago João Pereira Ferreira:

Nesta fase ajudei no desenvolvimento do código do concentrador e sistemas sensores.