SMART ENERGY

Sistemas Embebidos e de Tempo Real

Licenciatura em Engenharia de Sistemas Informáticos

Regime Pós-Laboral

2022/2023

Alunos

Nuno Mendes – Nº 2727

Rosário Silva – Nº 21138

Tiago Azevedo - Nº 21153

Francisco Pereira – Nº 21156

Orientação

Profº Pedro Cunha

Índice

Índice de ilustrações

1. Introdução

No âmbito da UC de Sistema Embebidos e de Tempo Real pretende-se desenvolver um sistema em tempo real que possa ser utilizado como resposta a uma necessidade específica. Este tema provém da proposta apresentada na unidade curricular de Projeto Aplicado, que pretende interligar várias UC's com o mesmo projeto.

A nossa equipa optou por escolher o setor energético considerando que o mesmo tem uma boa base para melhorias em termos de eficiência e, por conseguinte, em termos económicos. Um caso prático do nosso projeto será, por exemplo, otimizar a utilização das lâmpadas dos postes de iluminação das estradas com o objetivo de rentabilizar e prolongar o tempo de vida útil das lâmpadas, evitando assim, manutenções desnecessárias e, por consequência, reduzir a mão de obra de manutenção.

Neste momento não existe qualquer tipo de automatização no que toca ao controlo das luzes nos parques de estacionamento e vias públicas. Uma grande preocupação nos dias de hoje é o gasto excessivo de eletricidade, até porque grande parte dessa energia provém de combustíveis fósseis, levando a uma pegada de carbono significativa.

Este projeto poderá também contribuir para as metas da Comissão Europeia para as Smart Cities em 2030, que atualmente se encontram em risco de não serem alcançadas.

2. Proposta de sistema

Abaixo poderá ser observado o ponto da situação atual, em termos dos Requisitos Funcionais e Não Funcionais que foram prontamente levantados pela nossa equipa para obter, desta forma, a fundação daquilo que é esperado, uma vez que o projeto esteja na fase de lançamento.

2.1. Requisitos Funcionais

É sabido que os Requisitos Funcionais são a definição daquilo que o sistema poderá fazer, ou seja, a materialização de uma ou várias necessidades realizadas em prol do sistema. Estes requisitos terão um impacto substancial no sucesso do projeto. É essencial identificar essas regras o quanto antes, de modo que não haja falhas de comunicação. É importante frisar que estas regras podem sofrer alterações consoante as necessidades/adversidades que eventualmente possam surgir.

- **RF01: Gestão de Energia–**Através de sensores de luz, determinar qual o momento em que os postes de iluminação devem ser ligados/desligados mediante a hora e nível de luminosidade do momento (dia/noite);
- **RF02: Iluminação mediante movimento–**Desligar a iluminação ou reduzir a intensidade para um valor mínimo em caso de ausência de movimento;
- **RF03: Dados em Tempo Real–**Comunicação centralizada dos dados atuais (ID do Arduíno; valor do input da quantidade de luz; valor do output de iluminação e Input de deteção de movimento).

2.2. Requisitos Não Funcionais

Por outro lado, os Requisitos Não Funcionais definem o que é que o sistema fará mais concretamente. São premissas, restrições técnicas e necessidades que não podem ser atendidas através de funcionalidades. Estes Requisitos Não Funcionais (RNF) inerentes ao projeto, estão associados à qualidade e segurança da aplicação que garante o funcionamento otimizado de todo o sistema.

- **RNF01: Base de dados –**O armazenamento de dados deverá ser efetuado recorrendo a uma linguagem de base de dados que envolva SQL;
- **RNF02: Aplicação móvel–**A gestão do sistema, desde consultas à base de dados até ao estado de manutenção e monitorização do mesmo, deverá ser acedido através de uma App;
- **RNF03: Idioma da aplicação –**O sistema deverá ser capaz de ser totalmente traduzido do português para o inglês e vice-versa, de modo a qualquer pessoa ser capaz de compreender todas as funcionalidades do sistema;
- **RNF04: Conta de utilizador–**Qualquer utilizador autorizado a fazer uso do sistema terá de possuir uma conta feita de forma manual pelo responsável da segurança da infraestrutura.

2.3. Arquitetura do Sistema

Tendo em vista o plano final do projeto, foram efetuados alguns protótipos daquilo que o nosso projeto visa atingir. Para o efeito foi feito um esboço em papel, seguido de uma transição para um ambiente mais gráfico.

O primeiro passo para a concessão da arquitetura do sistema foi o desenho de um esboço em papel, criado na primeira reunião oficial do nosso grupo. Aqui podem ser vistos os vários componentes que fazem parte do nosso sistema, desde o caso de uso (uma estrada e respetiva iluminação) até aos componentes que manipulam o sistema (Arduinos, Raspberry Pi, etc…)

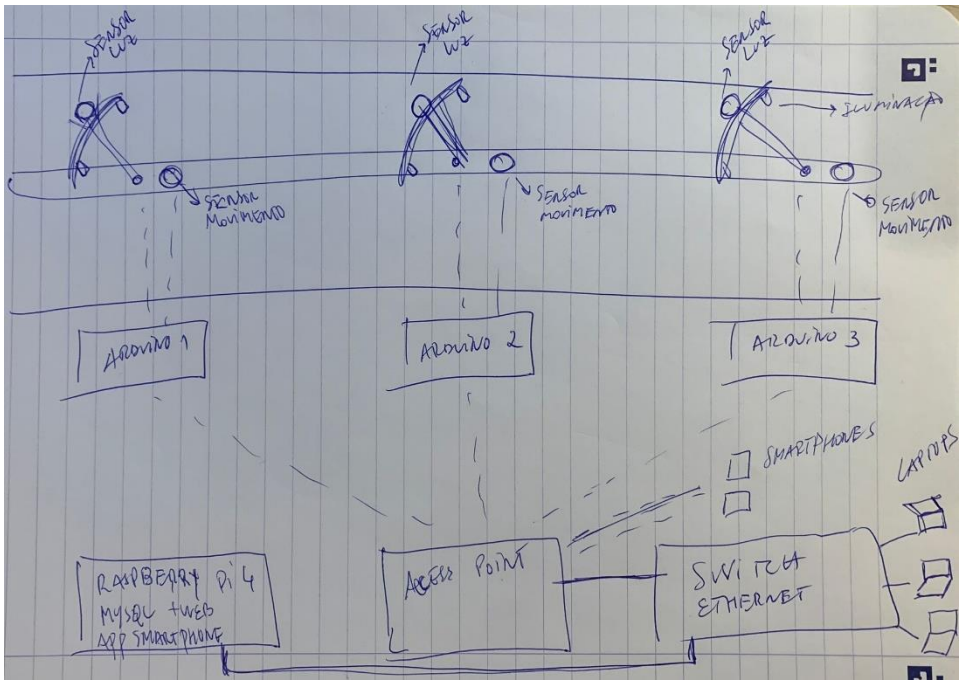

Figura 1 - Esboço do Sistema em papel

Após o esboço em papel ser alvo de análise e aprovação de todos os elementos do grupo, foi passado para um sistema mais organizado e conciso. Este é um diagrama de Hardware que representa o que cada poste necessita de ter de forma a fazer parte do sistema final, nomeadamente:

- 1 Lâmpada LED;
- 1 Sensor de Luz (LDR);
- 1 Sensor de Movimento (PIR);
- 1 Arduíno;
- 1 Shield WiFi (ESP8266).

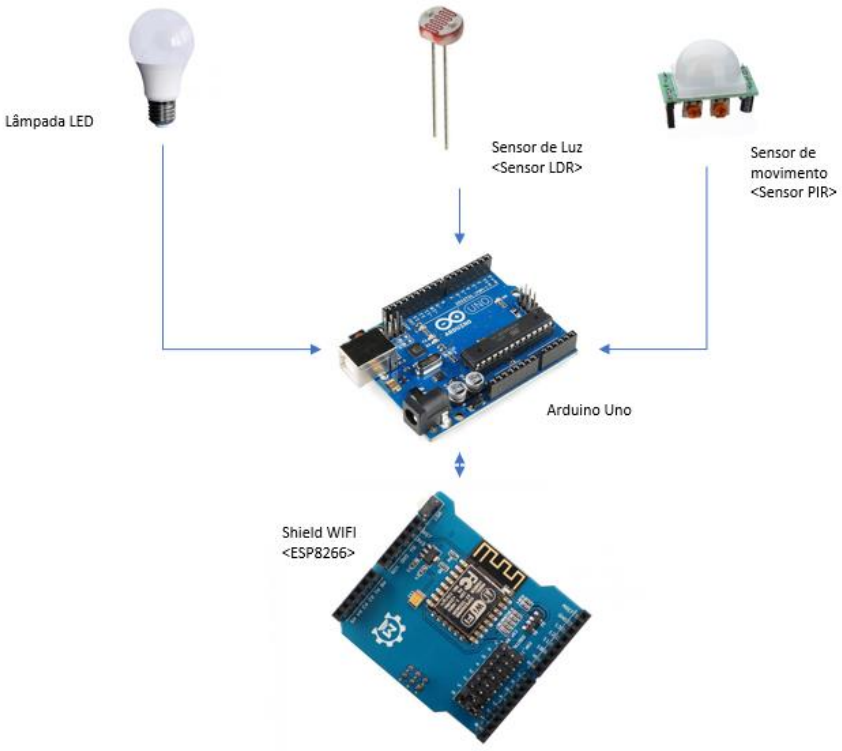

Figura 2 - Diagrama de Hardware de UM poste de iluminação

No próximo esquema conseguimos analisar, de forma completa, todos os componentes que de uma forma ou de outra, irão fazer parte do nosso sistema final.

Aqui estão representados três postes de iluminação, como forma de representação de uma estrada. Na totalidade, a nossa maquete física será constituída por:

- 3 Postes de Iluminação com os componentes descritos acima;
- 1 Access Point;
- 1 Switch Ethernet;
- 1 Raspberry Pi.

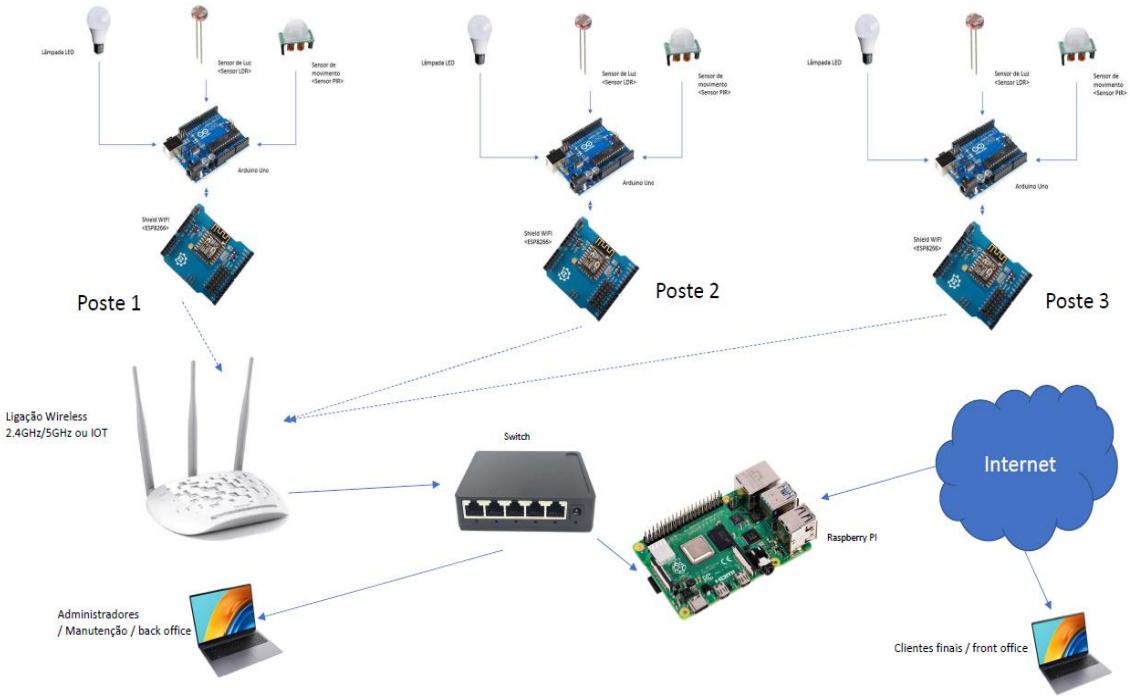

Figura 3 - Diagrama de Hardware de TODO o Sistema

Na próxima figura, efetuado na plataforma *TinkerCad*, podemos verificar uma demonstração visual de todos os componentes eletrónicos para efeitos de prototipagem.

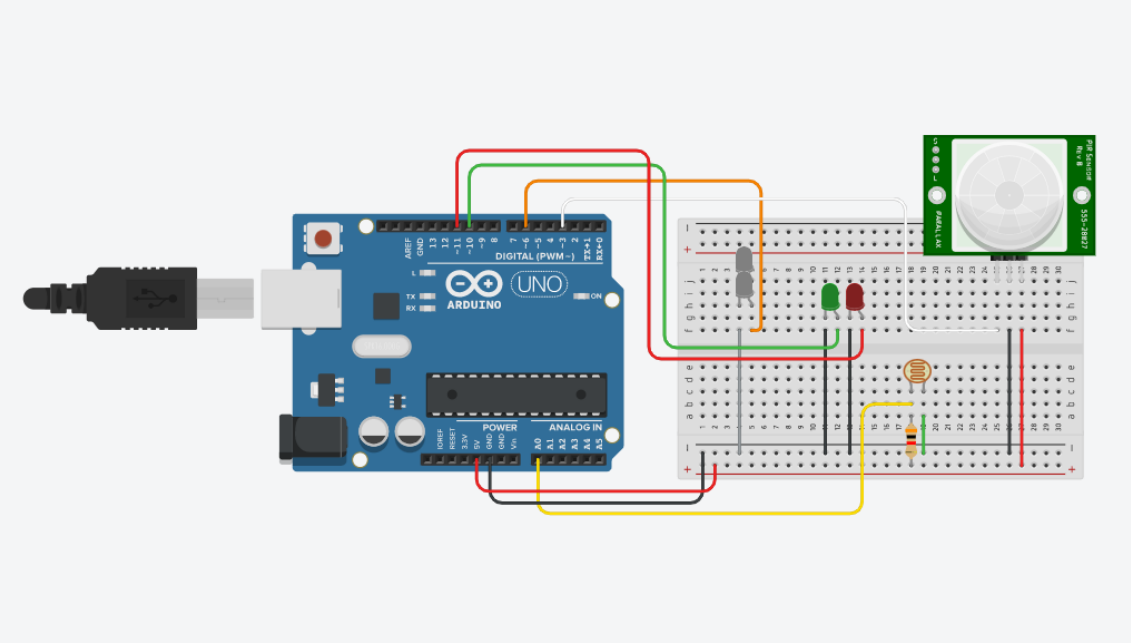

Figura 4 - Circuito eletrónico no TinkerCad

Este próximo esquema foi também efetuado no TinkerCad, mas desta vez para ter uma representação gráfica e simbológica dos circuitos na implementação do projeto, nomeadamente num poste de iluminação.

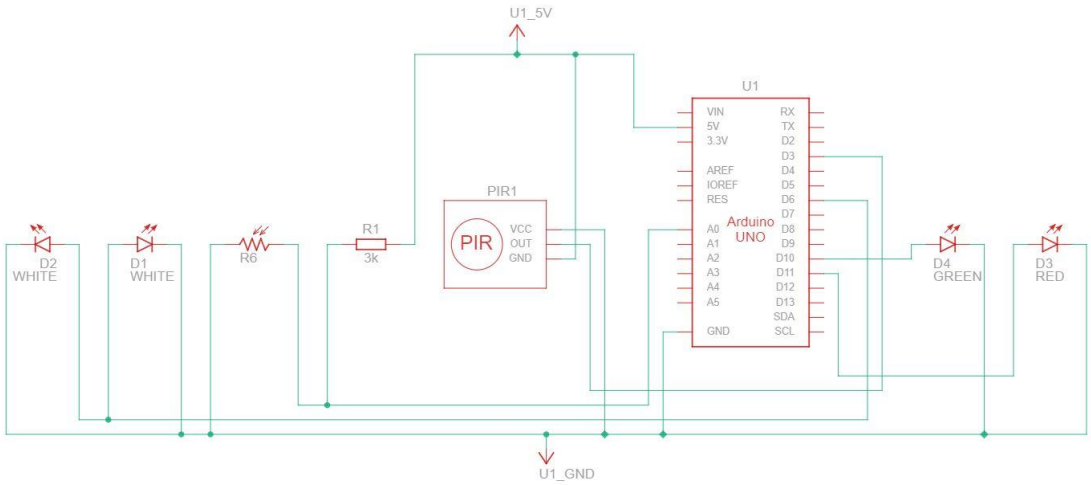

Figura 5 - Esquema do circuito eletrónico

As próximas figuras são maquetes que foram efetuadas fazendo uso da plataforma FIGMA, sendo que estas maquetes irão fazer parte dos recursos a utilizar para a criação da nossa aplicação de gestão do sistema.

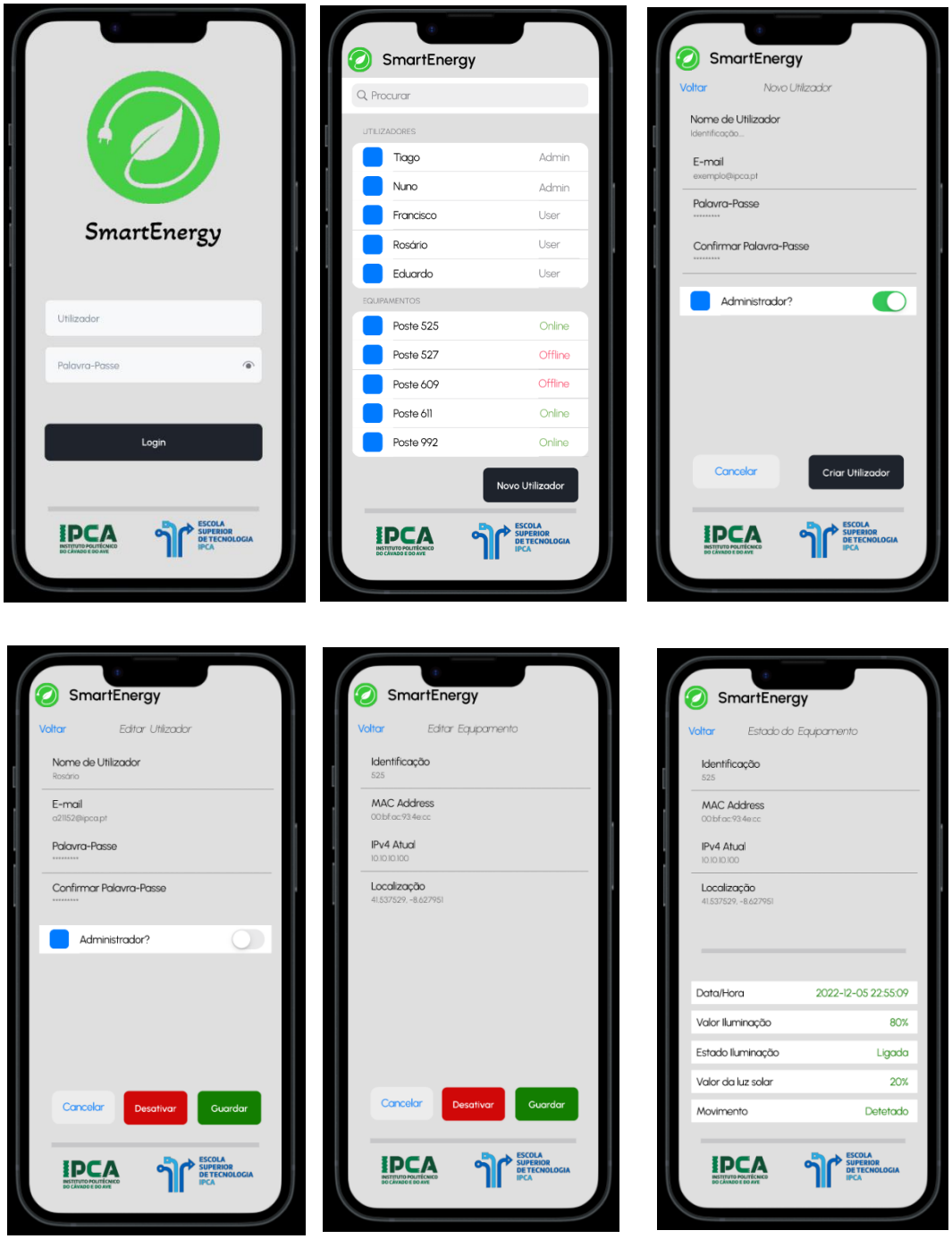

Figura 6 - Mockups da App Móvel

3. Código desenvolvido

Em relação ao código desenvolvido, estará presente na sua totalidade em anexo.

De seguida, está uma explicação mais detalhada sobre as funcionalidades implementadas com o respetivo trecho de código.

A. Interrupt ativado por um sensor:

```
#define interruptPin 3 // pino de input do sensor de movimento para a
funcao de interrupt via detecao de movimento
// inicializacao do Interrupt atraves da detecao de movimento com o sensor PIR
  attachInterrupt(digitalPinToInterrupt(interruptPin), detectionPIR, CHANGE);
void detectionPIR() {
  if (statePIR == LOW) {
    Serial.print("\nInterrupt via detecao de movimento ATIVADO!");
    statePIR = HIGH;            // estado de detecao de movimento passa a TRUE
    sendData = HIGH;            // estado de envio de dados para o servidor passa a
TRUE
  }
}
```

Este código tem como finalidade a execução de um Interrupt, ou seja, o objetivo será executar automaticamente as instruções para enviar os dados em tempo real para o servidor, quando é detetado movimento através do sensor PIR.

B. Interrupt através de temporizador:

```
int counter = 0;      // armazena a quantidade de segundos passados para o
interrupt atraves do timer
```

Primeiramente, é definido um contador para armazenar a quantidade de segundos que o Interrupt timer atuou.

```
// inicializacao do Interrupt atraves de um Timer
  Timer1.initialize(500000);
  Timer1.setPeriod(1000000);            // definido para periodos de 1
segundo
  Timer1.attachInterrupt(periodic);     // funcao que invoca quando e'
atingido o periodo
```

De seguida, é efetuada a inicialização do Interrupt através de um temporizador e este é definido para invocar a função *periodic* a cada segundo.

```
void periodic() {
  if (counter >= periodo) {
       Serial.print("\nInterrupt via timer1 a cada ");
       Serial.print(periodo);
       Serial.print(" segundos ATIVADO!");
       sendData= HIGH;          // estado de envio de dados para o
servidor passa a TRUE
       counter = 0;
  } else {
  ++counter;
  }
}
```

Este trecho de código é referente às instruções da função *periodic*, nomeadamente incrementa o contador *counter* até atingir os 120 segundos, que são definidos na variável *periodo*. Atingindo esta condição, os dados atuais são enviados para o servidor.

C. Implementação do Serviço Web HTTP:

```
/* inicio: serviço http do proprio poste de iluminacao*/

  // fica a espera de ligacoes de clientes
  WiFiEspClient client = server.available();
  if (client) {
    Serial.println("Nova ligacao http");
    // um pedido http termina com uma linha em branco
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
```

```
    // se chegou ao fim da linha (recebeu um caracter de nova linha) e a
linha está em branco,
    // o pedido http terminou, entao ja se pode enviar uma resposta
        if (c == '\n' && currentLineIsBlank) {
          Serial.println("Enviado resposta http");

          // envia um cabeçalho de resposta http padrão
          // use \r\n em vez de muitas instruções println para acelerar o
envio de dados
          client.print(
            "HTTP/1.1 200 OK\r\n"
            "Content-Type: text/html\r\n"
            "Connection: close\r\n"  // a ligacao será fechada após a
conclusão da resposta
            "Refresh: 5\r\n"         // recarrega a pagina
automaticamente a cada 5 segundos
            "\r\n");
          client.print("<!DOCTYPE HTML>\r\n");
          client.print("<html>\r\n");
          client.print("<h4>Smart Energy Campus</h4>\r\n");
          client.print("<h1>Lamp Post</h1>\r\n");
          client.print("<h2>Network</h2>\r\n");
          client.print("Mac Address: ");
          client.print(mac[5],HEX);
          client.print(":");
          client.print(mac[4],HEX);
          client.print(":");
          client.print(mac[3],HEX);
          client.print(":");
          client.print(mac[2],HEX);
          client.print(":");
          client.print(mac[1],HEX);
          client.print(":");
          client.print(mac[0],HEX);
          client.print("<br>\r\n");
          client.print("IP Address: ");
          client.print(ip);
          client.print("<br>\r\n");
          client.print("<h2>Status</h2>\r\n");
          client.print("Light value: ");
          client.print(valLED);
          client.print("<br>\r\n");
          client.print("Light state: ");
          client.print(stateLED);
          client.print("<br>\r\n");
          client.print("LDR value: ");
          client.print(valLDR);
```

```
          client.print("<br>\r\n");
          client.print("LDR %: ");
          client.print(valLDRnew);
          client.print("<br>\r\n");
          client.print("PIR value: ");
          client.print(valPIR);
          client.print("<br>\r\n");
          client.print("PIR state: ");
          client.print(statePIR);
          client.print("<br>\r\n");
          client.print("Timer: ");
          client.print(timer);
          client.print("<br>\r\n");
          client.print("<br>\r\n");
          client.print("Requests received: ");
          client.print(++reqCount);
          client.print("<br>\r\n");
          client.print("</html>\r\n");
          break;
        }
        if (c == '\n') {
          // Iniciando uma nova linha
          currentLineIsBlank = true;
        } else if (c != '\r') {
          // Obteve um caracter na linha atual
          currentLineIsBlank = false;
        }
      }
    }
    // Dando tempo ao navegador da web para receber os dados
    delay(10);

    // Terminar a ligacao
    client.stop();
    Serial.println("Cliente desconetado");
  }
  /* fim: serviço http do proprio poste de iluminacao*/
```

Foi implementado um serviço HTTP em cada poste de iluminação, permitindo que externamente com um equipamento com um navegador acede-se através do endereço IP ao posto e visualiza-se o estado atual do sistema. Durante o desenvolvimento e implementação de novas funcionalidades este código deixou de funcionar por motivos de incompatibilidades do sistema, está previsto numa futura versão a correção e reimplementação deste serviço.

D. Inteligência do poste de iluminação

```
void outputs() {
  // LDRmax - pouca iluminacao, sem sol, escuro
  // LDRmin - muita iluminacao, muito sol
  if (valLDR <= LDRmin ) valLDR=LDRmin;
  if (valLDR >= LDRmax) valLDR=LDRmax;
  valLDRnew = (long) (valLDR * 100 / LDRmax );     // converter para
percentagem 0% a 100%
  int valLEDnew =  (int) (255  * valLDRnew / 100); // atribui ao LED o
valor de iluminacao ideal de acordo com o sensor de input LDR

  if (valLDR >= LDRmed) {

    if (statePIR==HIGH) {   // caso volte a detetar movimento reinicia o
timer
      timer = TIMEmax;      // o tempo de LEDs ligados volta ao maximo
      stateLED = HIGH;      // liga os LEDs
      sendDataToServer();
      statePIR = LOW;       // estado detecao de movimento passa a FALSO
    }

    if (timer > TIMEmax) timer = TIMEmax;
    if (timer > 0) {
      timer = timer - (millis() - timer2);  // atualiza o tempo restante
guardado na variavel timer
      // Ajusta o valor da iluminacao conforme a intensidade de luz
"solar", o novo valor que esta guardado em valLEDnew
      if (valLED < valLEDnew) valLED=valLED + valINCREMENT;
      if (valLED > valLEDnew) valLED=valLED - valINCREMENT;
    } else {                      // reduz o valor da iluminacao dos LEDs
ao valor mínimo, iluminacao de presenca
      timer = 0;
      stateLED = HIGH;
      if (valLED > valLEDmin) {   // reduz a iluminacao até ser igual ao
valor de iluminacao de presenca valLEDmin
        valLED = valLED - valINCREMENT;
      } else {
        valLED = valLEDmin;
      }
    }

    analogWrite(LED, valLED);     // atribui a iluminacao atual aos LEDs
  } else {                        // desliga os LEDs
    stateLED = LOW;
    valLED = 0;                   // atribui a iluminacao a zero...
    analogWrite(LED, valLED);     // ...e desliga os LEDs
```

```
    timer = 0;
    //sendDataToServer();
  }

  // envia para a consola os dados atuais de input e output
  Serial.print("\nLight actual value: "); Serial.print(valLED);
  Serial.print("| Light next value: "); Serial.print(valLEDnew);
  Serial.print("| Light state: "); Serial.print(stateLED);
  Serial.print("| LDR value: "); Serial.print(valLDR);
  Serial.print("| LDR %: "); Serial.print(valLDRnew);
  Serial.print("| PIR value: "); Serial.print(valPIR);
  Serial.print("| PIR state: "); Serial.print(statePIR);
  Serial.print("| Timer: "); Serial.print(timer);
  Serial.print("| Counter: "); Serial.print(counter);
  delay(75);

  timer2 = millis();    // regista o tempo atual
  if (statePIR == HIGH && stateLED == LOW) {
    statePIR = LOW;
  }
}
```

Este código faz as verificações dos dados de entrada obtidos pelos sensores e, de acordo com a quantidade de luz natural existente e a presença de movimento, faz atuar a iluminação artificial por um curto período, ou seja, apenas o essencial para a iluminar a zona onde se enquadra o poste.

E. Envio de dados para o servidor

```
void sendDataToServer() { // funcao que faz o envio dos dados atuais para
o servidor
  Serial.println("\nEnviado dados para o servidor");
  client.stop(); // termina todas as ligacoes e efetua um novo pedido e
liberta o socket do shield WiFi

  // verifica se existe conetividade com o servidor
  if (client.connect("10.10.10.2", 80)) {
    // a ligacao com o servidor foi efetuada
    String s1 = "GET /webservices.php?macaddress=";
    s1 += String(mac[5],HEX); s1 += ":";
    s1 += String(mac[4],HEX); s1 += ":";
    s1 += String(mac[3],HEX); s1 += ":";
    s1 += String(mac[2],HEX); s1 += ":";
    s1 += String(mac[1],HEX); s1 += ":";
    s1 += String(mac[0],HEX);
```

```
    s1 += "&ipaddress=";
    s1 += String(ip[0])+String(".")+String(ip[1])
+String(".")+String(ip[2])+String(".")+String(ip[3]); // endereço IP
atual
    String s2 = "&valled="; s2 += valLED;
    s2 += "&stateled=";     s2 += stateLED;
    s2 += "&valldr=";       s2 += valLDR;
    s2 += "&valldrnew=";    s2 += valLDRnew;
    s2 += "&valpir=";       s2 += valPIR;
    s2 += "&statepir=";     s2 += statePIR;
    s2 += " HTTP/1.1";
    s1 += s2;
    Serial.println((s1));
    client.println((s1));
    client.println(F("Host: 10.10.10.2"));
    client.println("Connection: close");
    client.println();
  }
  else {
    // se a ligacao com o servidor nao for efetuada
    Serial.println(F("A ligacao falhou!"));
  }
  client.stop();
}
```

Nesta função é invocado um webservice do nosso servidor em que passam todos os valores/dados atuais. Estes dados são armazenados numa base de dados (MariaDB) para mais tarde serem tratados. Além disso, na unidade curricular de Programação de Dispositivos Móveis, elaborou-se uma aplicação android para analisar e tratar estes dados de acordo com as maquetes planeadas anteriormente.

F. Acesso HTTP aos postes de iluminação

As próximas três figuras representam os dados em tempo real a serem consultados no Serviço Web de cada poste, através do respetivo endereço de IP. Podem ser verificados dados importantes, tais como o nível de iluminação ou até mesmo a percentagem de luz solar que o sistema está a detetar.

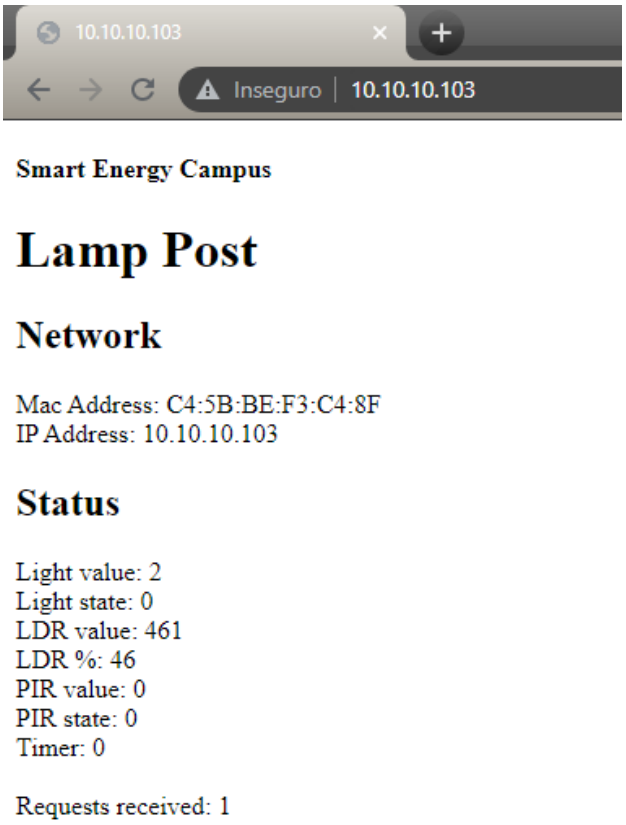

Figura 7 - Serviço Web de um Poste de Iluminação #1

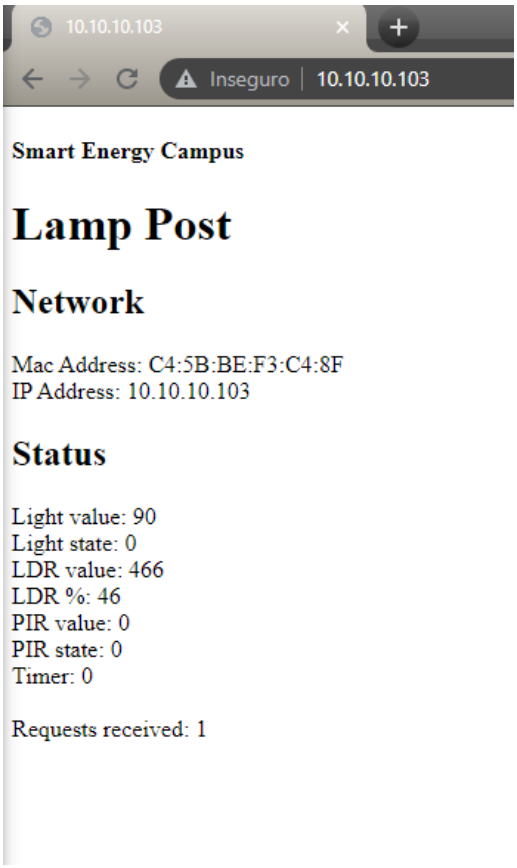

Figura 8 - Serviço Web de um Poste de Iluminação #2

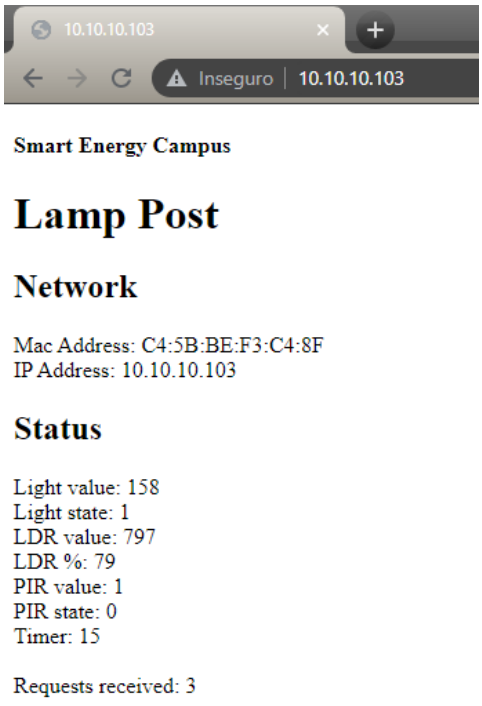

Figura 9 - Serviço Web de um Poste de Iluminação #3

4. Maquete do Sistema

Nas duas próximas figuras conseguimos observar a maquete que já foi prontamente produzida conforme as instruções dos esquemas anteriores.

É de referir que o sensor PIR em futuros trabalhos será colocado num ponto mais estratégico do poste de iluminação de forma que a área de abrangência seja superior e o poste não interfira na deteção de movimento.

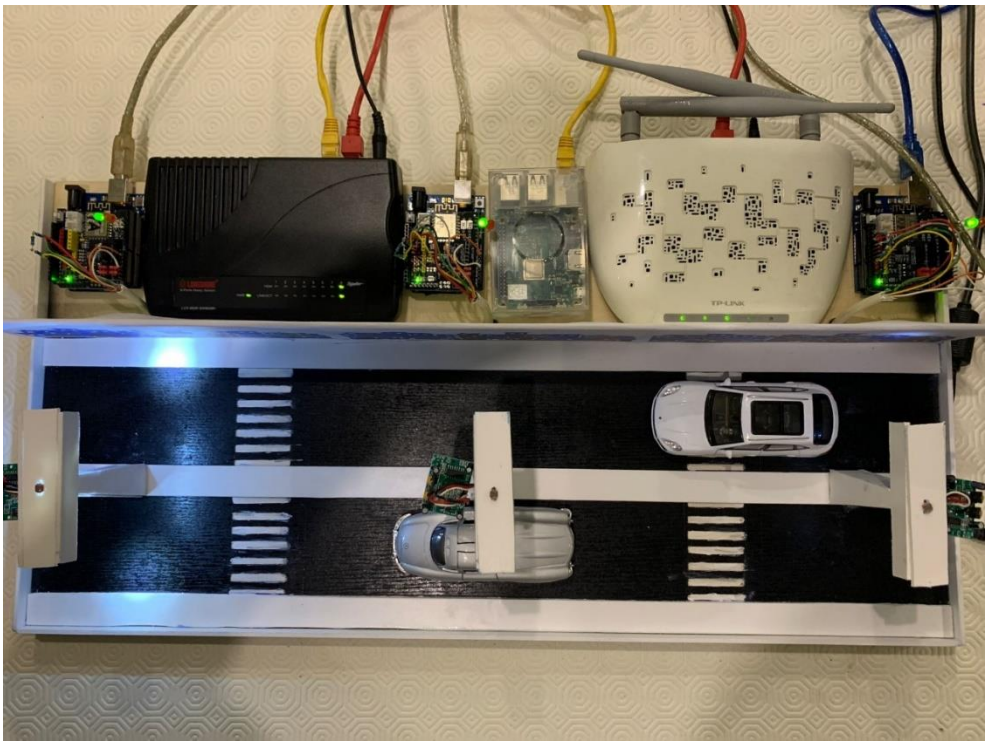

Figura 10 - Maquete do Sistema (visão de cima)

Figura 11 - Maquete do Sistema (visão lateral)

Figura 12 - Maquete do Sistema (visão do Poste #1)

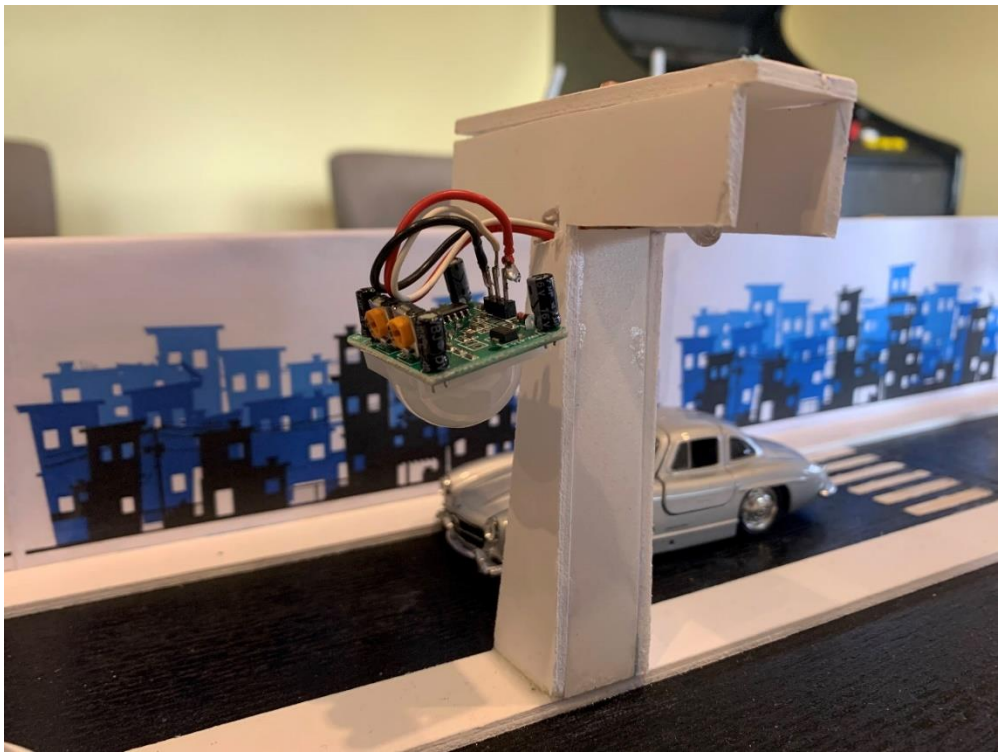

Figura 13 - Maquete do Sistema (visão do Poste #2)

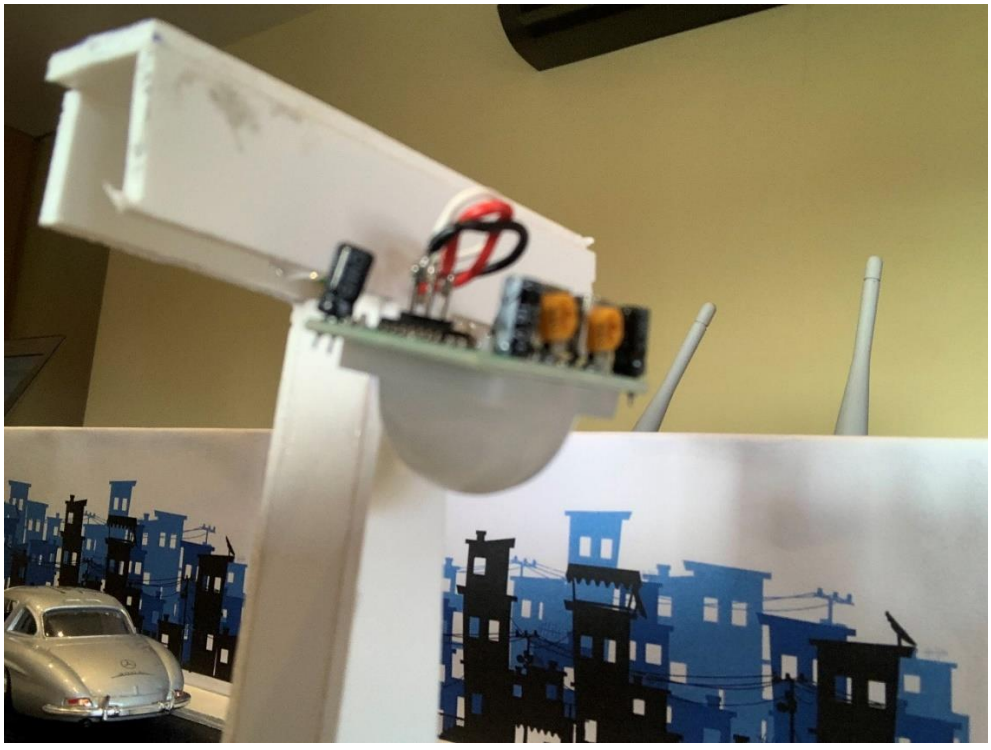

Figura 14 - Maquete do Sistema (visão do Poste #3)

5. Justificação das decisões tomadas

Para o constante progresso deste projeto e, de forma a elaborar uma prova de conceito, foram utilizadas várias tecnologias que nos permitiram estabelecer o que foi planeado através dos diversos conceitos previamente discutidos.

5.1. Hardware

- **HW01: Raspberry Pi 3 Model B+, 1.4GHz, 1GB, cartão SD de 16GB -** O servidor foi implementado num Raspberry Pi para fins de protótipo. É um sistema de baixo custo, consumo reduzido e de grande capacidade para efeitos de prototipagem e desenvolvimento.
- **HW02: Ponto de Acesso de redes sem fios -** Neste equipamento foi configurada uma rede wireless com *essidsmartenergy*segura com encriptação para permitir a comunicação dos *shield wifi* dos *arduinos* com o servidor.
- **HW03: Switch Ethernet -** Este dispositivo foi adicionado à maquete do protótipo com a finalidade de facilitar e interligar diversos computadores portáteis a fim de se poder desenvolver, analisar, etc…
- **HW04: OSOYOO WiFi IoT Learning Kit For Arduino -** Este kit *Arduino* é um clone fiel ao original, além de ser mais económico, inclui os dispositivos necessários para implementação do protótipo do projeto numa maquete. Os componentes essenciais para conceber o protótipo são o *Arduino Uno R3*, o *shield* ESP8266 WIFI, o sensor de movimento por infravermelhos, o sensor LDR foto resistor, resistências e leds. Para efeitos do projeto foram adquiridos 3 conjuntos para simular 3 postes de iluminação público e poder-se obter dados em situações diferentes e mais realísticas.

Em relação aos componentes eletrónicos utilizados, podem ser consultados na tabela seguinte:

Nome	Quantidade	Componente
U1	1	Arduino Uno R3
W1	1	ESP8266 WIFI
PIR1	1	Sensor PIR
D1 D2	2	LED Branco
D3	1	LED Vermelho
D4	1	LED Verde
R1	1	Resistência de 3 kΩ
R6	1	Foto resistor (LDR)

5.2. Software

- **SW01: Debian OS -** O sistema operativo escolhido para a instalação e desenvolvimento no Raspberry Pi.
- **SW02: Web Services -** Foram incluídos alguns pacotes para disponibilização de serviços web, sendo eles o Apache2 e o PHP 8.1.
- **SW03: Base de dados -** Para a realização de toda a estrutura de dados, foi escolhida uma base de dados simples mas eficiente, sendo ela a MariaDB 10.5.15.

Desenvolveu-se na linguagem de programação PHP um serviço web para que o poste de iluminação possa comunicar com o servidor e, por sua vez, os dados transmitidos possam ser guardados na base de dados. Existe um ficheiro no servidor que é invocado pelos Arduinos com a passagem de determinados parâmetros para suportar esta comunicação. O ficheiro em questão denomina-se por *webservices.php*, que pode ser encontrado no diretório:*/var/www/html*.

Para o desenvolvimento do código nos Arduinos, foi escolhida a plataforma open source do fabricante, o Arduino IDE v2.03. A linguagem de programação do Arduino é o C++.Muitas das vezes, estes blocos de código são muito semelhantes à linguagem de programação C, no entanto, é um ambiente relativamente fácil de trabalhar e desenvolver.

6. Conclusão

A realização deste trabalho permitiu-nos esmiuçar os conteúdos lecionados na unidade curricular de Sistemas Embebidos e de Tempo Real, nomeadamente o uso de um Arduino para a realização de um projeto inovador.

O nosso grupo de trabalho compreende o grande poder e usabilidade de um dispositivo deste género e aprecia o facto de nos serem explicadas todas as dúvidas, sempre que necessário, por parte do docente.

Os resultados obtidos apontam para a redução dos consumos de energia, o que irá levar a uma redução da pegada ambiental causada pelo consumo excessivo de energia.

Uma prova de protótipo conceptual já está totalmente funcional e pronta para ser apresentada.

Com base na nossa investigação, a nossa equipa reconhece o problema emergente das emissões de CO2 em Portugal e espera que projetos como este possam proporcionar consciência e inspiração para criar ideias mais inovadoras no futuro e que possam ajudar a comunidade de uma forma positiva.

7. Bibliografia

Raspberry Pi:

https://www.raspberrypi.com/

Sistema Operativo Debian:

https://wiki.debian.org/RaspberryPi

Programação em PHP:

https://www.php.net/

Base de dados MariaDB:

https://mariadb.org/

Clone Arduino OSOYOO:

https://osoyoo.com/

Documentação sobre Arduinos e programação:

https://docs.arduino.cc/software/ide-v2?_gl=1*1x87tgx*_ga*MjUzNjY5NjAxLjE2NjQzMDI4ODQ.*_ga_NEXN8H46L5*MTY3MDQwODU4Ni4xMy4xLjE2NzA0MDg2MDkuMC4wLjA