



Unidade Curricular de
Sistemas Electrónicos para a Internet das Coisas
Engenharia Eletrotécnica e de Computadores

Sistema IoT

Autores:

Gonçalo Gaspar Lopes nº 2181775
Mateo Daniel Rodriguez Romero nº 2182076

Leiria, junho de 2021

Conteúdo

1	Introdução	1
1.1	Estrutura do sistema	1
2	Projeto	3
2.1	Círculo de Monitorização do Ambiente	3
2.1.1	Monitorização local a partir do <i>LCD</i>	4
2.2	Círculo de Segurança	5
2.3	Servidor Local	7
3	Componentes	11
	Conclusão	13

Lista de Figuras

1.1	Diagrama do sistema	2
2.1	Diagrama do Circuito de Mon. do Ambiente	4
2.2	LCD - Sistema Ativo	4
2.3	LCD - Intruso	4
2.4	Diagrama do circuito de segurança	5
2.5	Dashboard Node-Red	8
2.6	FLows Node-Red	9
2.7	Email recebido	9

Capítulo 1

Introdução

Este relatório é referente à unidade curricular de Sistemas Electrónicos para a Internet das Coisas, com o objectivo de desenvolver e representar um sistema IoT utilizando várias componentes e frameworks estudadas. Para este projeto decidiu-se implementar um sistema domótico, constituído por 3 circuitos:

- Circuito de monitorização do ambiente;
- Circuito de segurança;
- Servidor local.

1.1 Estrutura do sistema

Na figura 1.1 está representado o sistema completo do projeto e os pinos de ligação de cada componente. No capítulo 2 são descritos todos os sub-sistemas e o seu funcionamento.

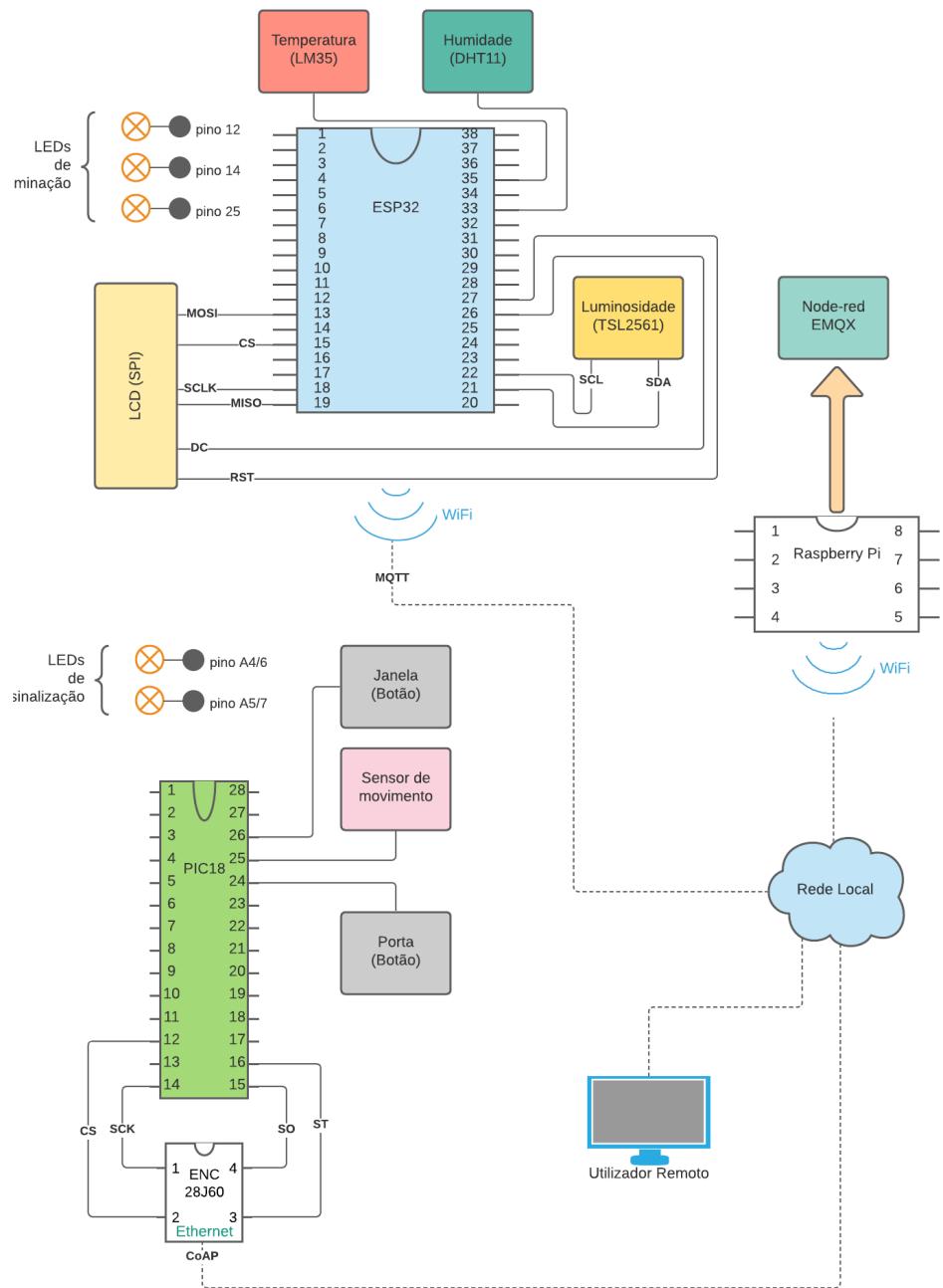


Figura 1.1: Diagrama do sistema

Capítulo 2

Projeto

2.1 Circuito de Monitorização do Ambiente

Este circuito é controlado a partir de um *ESP32* e tem como objetivo reunir dados sobre o meio ambiente com alguns sensores e atuar sobre 3 *LEDs* (ver figura 2.1). Os sensores presentes neste sistema são:

- *LM35* - Sensor de temperatura;
- *DHT11* - Sensor de Humidade;
- *TSL2561* - Sensor de Luminosidade.

Para complementar este sistema agregou-se um *LCD*, comunicando via *SPI*, para monitorizar as variáveis do local, o estado de cada *LED* e o estado de segurança da casa (ver secção 2.1.1).

O *ESP32* está conectado a um *broker* na rede local via *WiFi* e comunica através do protocolo *MQTT*. No *broker* ele publica em vários canais o valor de cada sensor e está subscrito a um canal de segurança, um de alarme e a três canais de controlo de iluminação.

O funcionamento do *ESP32* segue uma sequência sempre em *loop*. Primeiro lê os valores dos sensores, o sensor de temperatura tem de passar por um conversor *ADC*, o de humidade segue uma comunicação serial (onde recebe 8bits de valor inteiro e 8 decimais) e o sensor de luminosidade segue uma comunicação *I2C*. De seguida publica cada valor no tópico correspondente e por fim atualiza o *LCD*. Se houver alguma publicação nos tópicos subscritos, entra numa interrupção *mqtt_callback*, onde filtra de onde veio a publicação e atua conforme a mensagem.

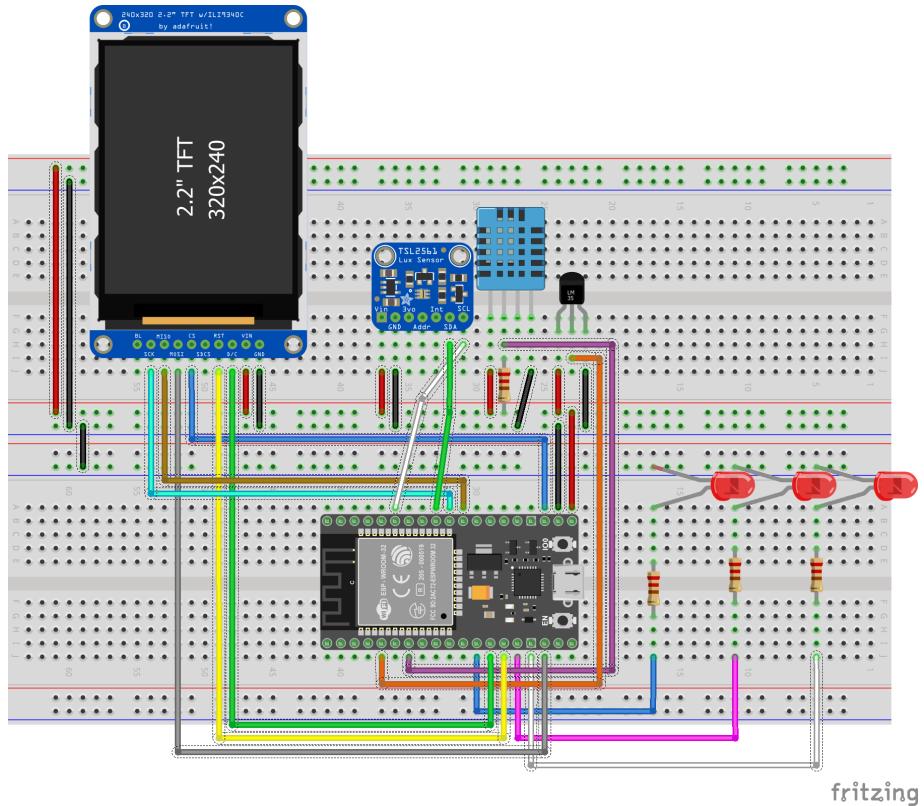


Figura 2.1: Diagrama do Circuito de Mon. do Ambiente

2.1.1 Monitorização local a partir do LCD



Figura 2.2: LCD - Sistema Ativo

Figura 2.3: LCD - Intruso

Nas figuras 2.2 e 2.3 estão representados 2 exemplos da monitorização do sistema no local.

Na primeira figura está representado o *LCD* com todos os valores medidos nos sensores, os estados das lâmpadas, em que a 2^a está ligada, e o sistema de segurança está ativo. Na segunda figura, retira-se que a 3^a lâmpada também está ligada e que alguém despoletou o alarme.

2.2 Circuito de Segurança

Para o sistema de segurança foi utilizado o microcontrolador *PIC18LF26K40*, a comunicar por Ethernet. Tem 2 sensores representados por botões de pressão a monitorizar o estado de uma porta e uma janela, um sensor de movimento e dois LED's que sinalizam o estado do alarme e se o sistema está ativo.

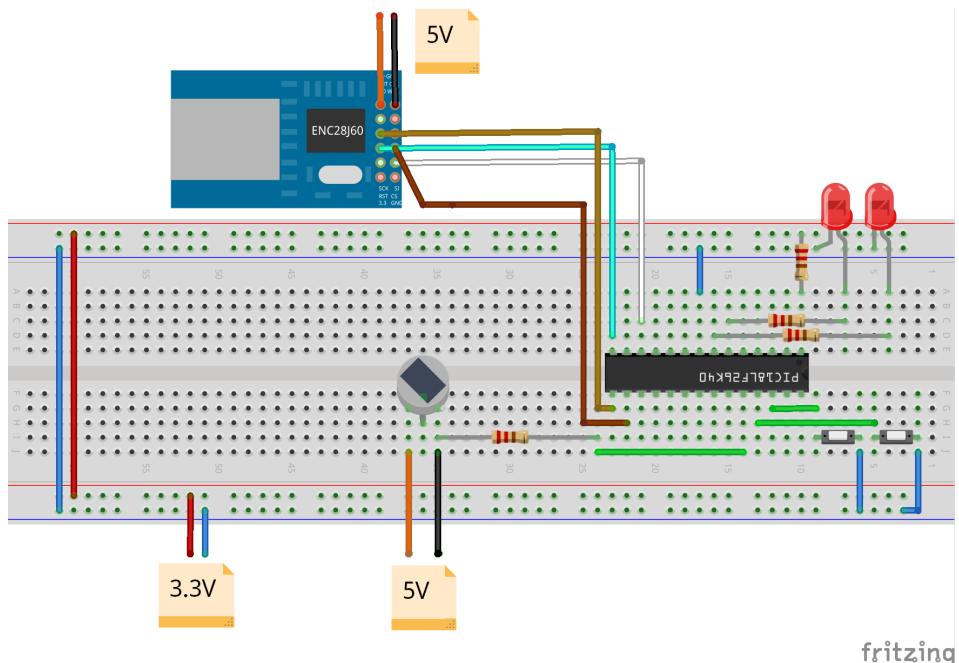


Figura 2.4: Diagrama do circuito de segurança

Para a monitorização do sensor e dos botões foram utilizadas 3 interrupções. As interrupções dos botões foram configuradas para serem acionados num flanco descendente, em conjunto com o uso de WPU. Para o sensor foi utilizada uma interrupção em flanco ascendente, porque o sensor A1626311 envia 3V se detetar movimento e 0V se não o detetar.

O sistema utiliza o protocolo *CoAP* para comunicar com o *Node-red*, em que os recursos encontram-se organizados no código com a seguinte ordem:

1. Sistema
2. Alarme (só pode ser desligado)
3. Janela
4. Porta
5. Sensor de movimento

Em relação à utilização do *CoAP* no microcontrolador, foi utilizada a biblioteca *CoAP Utility*, e para poder ligar por meio de *SPI* ao *ENC28J60* foi utilizada a biblioteca *MAC*.

O sistema inicia por *default* desativado, para ativá-lo é enviado o comando *put coap://ipi.pip.ipi.pip/mchp/Led1 Led,1* a partir do *Node-red*. Agora com o sistema ativo, se algum dos botões for pressionado, ou se for detetado movimento, o microcontrolador atualiza os estados correspondentes de cada "sensor". Com isto o *Node-red* envia pedidos *CoAP* cada segundo para saber o estado de cada sensor, e ativa o alarme.

Uma vez que o alarme foi ativado, o utilizador poderá desligar o alarme ou desligar o sistema inteiro a partir do *Node-red*. Se o alarme for desligado o sistema coloca os estados dos sensores a 0. Se o sistema for desligado, os estados dos sensores passa a 0 e só enviarão informação de volta se o sistema for ligado novamente.

Para que seja possível enviar os estados dos sensores nos pedidos *CoAP*, foi necessário adaptar o código do ficheiro *Led.c*:

```
bool LedGetter(uint8_t idx){  
    if (idx+1 == 2) { //ALARME  
        if(alarmeAt)  
            Led[idx].ledStatus = 1;  
        else  
            Led[idx].ledStatus = 0;  
    }  
    if (idx+1 ==3) { //JANELA  
        if(janAberta)
```

```

        Led[idx].ledStatus = 1;
    else
        Led[idx].ledStatus = 0;
    }

    if (idx+1 ==4) { //PORTA
        if(portAberta)
            Led[idx].ledStatus = 1;
        else
            Led[idx].ledStatus = 0;
    }

    if (idx+1 ==5) { //MOVIMENTO
        if(movdet)
            Led[idx].ledStatus = 1;
        else
            Led[idx].ledStatus = 0;
    }

    bool ret = ERROR;
    Get_ToEthernet((char *) "led", TEXT_STRING);
    Get_ToEthernet(Led[idx].ledStatus, UNSIGNED_INTEGER);
    printf("\n\rEnviado: ,led,%d,",Led[idx].ledStatus);
    return SUCCESS;
}

```

O estado do sistema não é alterado internamente pelo *P/C* por tanto, não foi necessário criar condições extra para saber se o sistema está ligado ou não. As variáveis de tipo *bool* dos sensores e do alarme são partilhadas com a *main.c*.

2.3 Servidor Local

Este módulo funciona como um servidor na rede, aqui é implementado o *Node-Red* e o *EMQx* a correr num *Raspberry Pi*, todos os valores dos sensores e dos alertas do circuito de segurança são enviados para este servidor. O sistema de monitorização do ambiente envia os dados por meio de *MQTT*, e o sistema de segurança utiliza o protocolo *CoAP*. De seguida os dados são colocados num *dashboard*, onde o utilizador pode visualizá-los e tomar ações (figura 2.5).

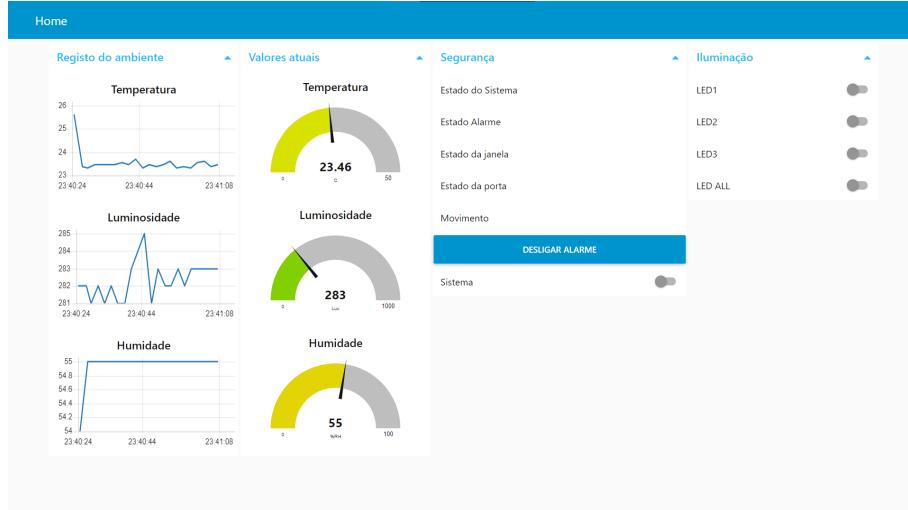


Figura 2.5: Dashboard Node-Red

A estrutura dos canais *MQTT* são *local/ambiente/* e a seguir especifica-se o tipo da variável com *temp*, *lum*, ou *hum* para a informação vinda dos sensores. Para o sistema de segurança a estrutura segue *local/seguranca/* e a seguir *sistema*, *alarme*, *janela*, *porta*, ou *movimento* conforme o sensor ou atuador. Finalmente para o sistema de iluminação é *local/iluminacao/* junto com *led1*, *led2* ou *led3*. Foi escolhida esta estrutura para permitir a subscrição em grupo dos diferentes tópicos, por exemplo para todas as luzes com *local/iluminacao/#*.

As mensagens enviadas pelo sistema do ambiente são diretamente colocadas nos diferentes *widgets* do *dashboard*, enquanto que a informação obtida pelos pedidos *GET* do *CoAP* são processadas antes de serem visualizadas (figura 2.6). O *dashboard* encontra-se dividido em 4 grupos, “Registo do ambiente” mostra gráficos da temperatura, humidade e luminosidade, o grupo “Valores atuais” mostra as medições das grandezas em tempo real, o grupo “Segurança” tem os estados de cada um dos sensores, do alarme e do sistema, e o último grupo “Iluminação” controla os 3 *LEDs* que simulam a iluminação da habitação.

O controlo da iluminação, a desativação do alarme e a ativação o sistema de segurança é tudo feito a partir do *dashboard*, por meio de *switches* e botões.

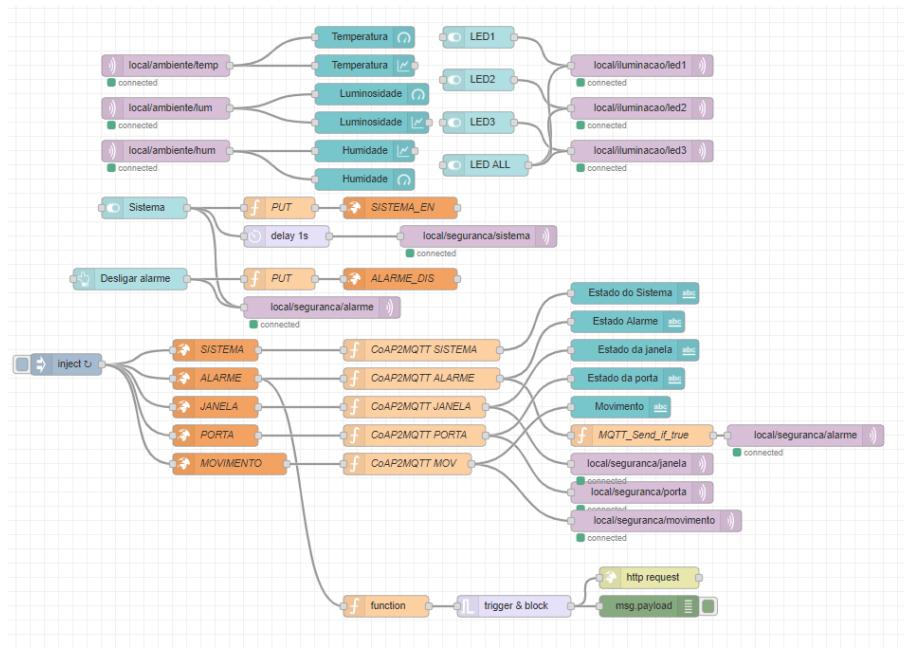


Figura 2.6: Flows Node-Red

Os dados do sistema de segurança também são publicados nos tópicos *MQTT* específicos para que eles sejam visíveis no *LCD* do sistema do ambiente. Além disto, se houver algum alerta no circuito de segurança, é enviado um email ao utilizador. Isto é realizado por meio do *If This Then That (IFTTT)*, em que o *Node-red* ao ativar o alarme, envia um *http request* para o *IFTTT* e neste é gerada uma mensagem personalizada com destino ao email do utilizador (ver figura 2.7).

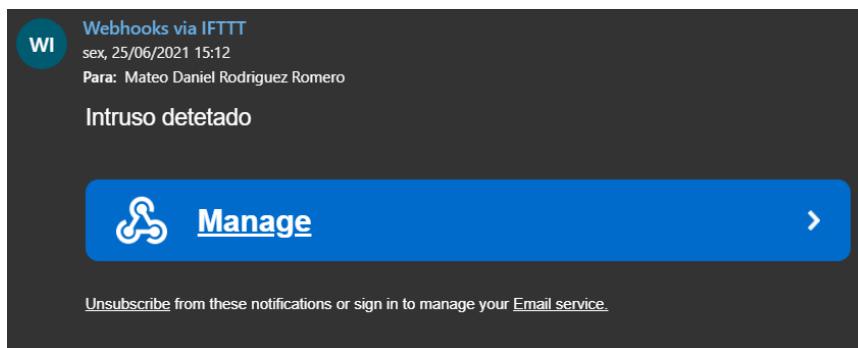


Figura 2.7: Email recebido

Capítulo 3

Componentes

- *PIC18LF26K40*;
- *EPS32*;
- *Raspberry Pi3 B*;
- *TFTSPI(LCD240x320)*;
- *LM35*;
- *DHT11*;
- *TSL2561*;
- 2 botões de pressão;
- *A1626311*;
- *ENC28J60*;
- 5 *LEDs*;

Conclusão

Para concluir, o resultado final deste trabalho está completamente funcional, todos os requisitos propostos para o projeto foram cumpridos:

- Um servidor a correr num PC ou nuvem - broker *EMQx* a correr no *Raspberry*;
- Utilizar dois sistemas locais baseados em hardware diferente - *ESP32* e *PIC*;
- Incluir *PIC* com comunicação via rede fixa - *ENC28J60*;
- Permitir a monitorização remota - *Node-Red*;
- Pelo menos uma configuração remota em cada sistema - Ativar LEDs de iluminação e ativar sistema de segurança;
- 1 notificação via email - Quando o alarme é ativo;
- Permitir o registo de dados local e remoto - Os dados estão acessíveis no *LCD* e no *Node-Red*;
- 2 botões de pressão;
- Incluir pré-processamento local - Processamento *ADC* do *LM35*;
- Incluir pré-processamento remoto - Transformar mensagem *CoAP* para *MQTT*;

Este projeto deu a entender a matéria abordada na disciplina, a importância que o *IoT* tem e terá nesta geração, a importância da ligação de dispositivos com diferentes protocolos de comunicação, no nosso caso entre *CoAP* e *MQTT*, e como toda a informação dos vários controladores pode ser agregada, neste caso por meio do *Node-red*.