SISTEMAS COMPUTACIONAIS EMBEBIDOS

ANO LECTIVO: 2020/2021

CURSO: ENG. Eletrotécnica

REGIME: D

TRABALHO DE AVALIAÇÃO 1

AUTORES:

NOME: Gonçalo Lopes

Nº ALUNO: 2181775

NOME: Mateo Romero

Nº ALUNO: 2182076

Índice

Índice de Figuras

Introdução

Este relatório é realizado no âmbito da disciplina de Sistemas Computacionais Embebidos e tem como objetivo dar a entender o projeto da disciplina e implementar grande parte da matéria abordada na cadeira.

O projeto trata-se do controlo de uma estufa, em que este tem de garantir condições climáticas regularizadas para o crescimento de plantas. Para conseguir manter e monitorizar estas condições climáticas especificas, é proposto a utilização de:

- Controlador:
  - ESP32;
- Sensores (*PySense*):
  - Temperatura (Si7006-A20);
  - Luminosidade (LTR-329ALS-01);
  - Humidade (Si7006-A20);
- Atuadores:
  - Motor de Rega;
  - Servomotor para o controlo de abertura de janela;
  - Resistência de aquecimento;
  - LED sinalizador;

A comunicação feita entre o microcontrolador e os sensores será feita a partir do protocolo I2C.

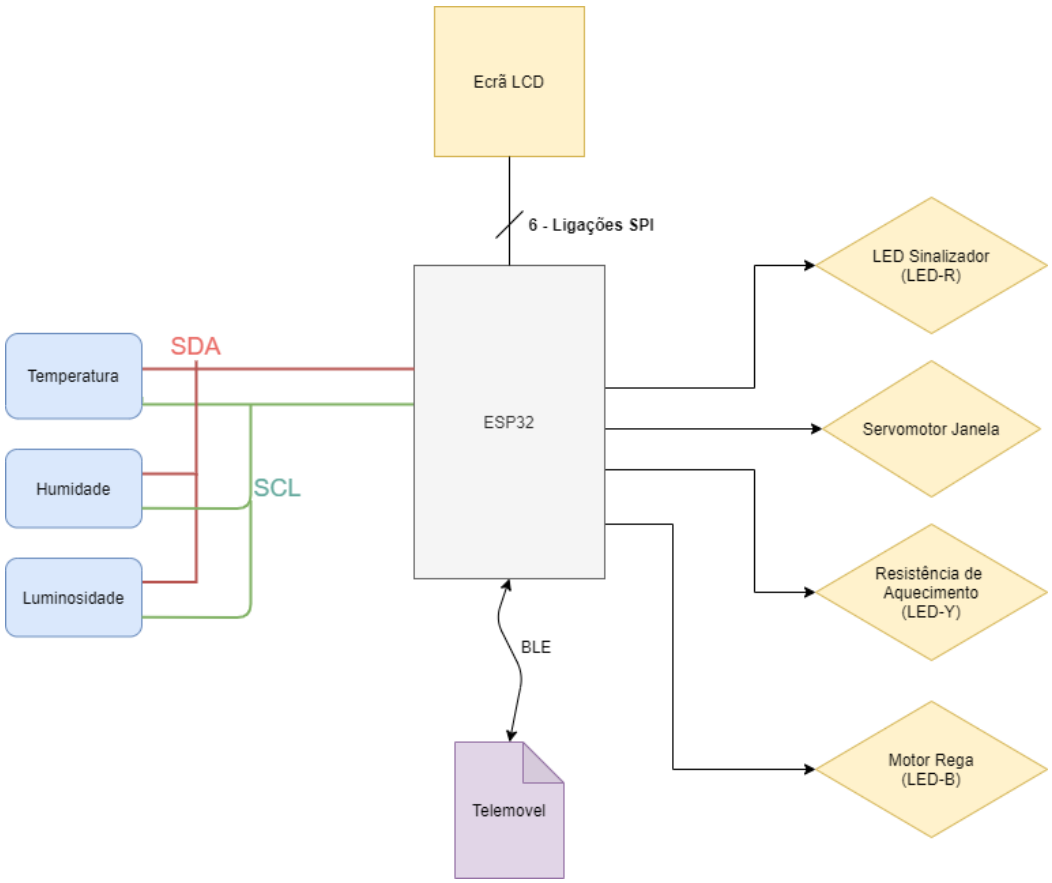
Para o controlo da temperatura serão utilizadas duas saídas, um servomotor para o controlo de uma janela, permitindo ventilação da estufa e uma resistência de potência para aquecer a mesma, neste caso vai ser representado por um LED. O controlo de humidade será utilizado um motor de rega, também representado por um LED.

Os valores dos sensores e dos estados dos atuadores estarão visíveis num LCD utilizando um protocolo de comunicação SPI.

Este projeto também abrange comunicação por via *Bluetooth Low Energy* (BLE) com um dispositivo móvel, para supervisionar todo o sistema.

Ligações

- **Saídas:**
  - LED Rega – Pino 14;
  - LED Resistência – Pino 12;
  - LED Luz – Pino 2;
  - Servomotor – Pino 25.
- **I2C:**
  - SDA – Pino 21;
  - SCL – Pino 22.
- **SPI:**
  - CS – Pino 15;
  - Reset – Pino 27;
  - MOSI/SDI – Pino 26;
  - MISO/SDO – Pino 13;
  - SCK – Pino 19;
  - D/C – Pino 18.



Visão Geral

Para este trabalho decidiu-se repartir as funções em tarefas distintas para aproveitar melhor o processador do microcontrolador e tronar o código mais eficiente (ver Figura 1).

As tarefas foram organizadas da seguinte forma:

TAREFA	DESCRIÇÃO	PRIORIDADE
T0	**Atualização LCD**	**1**
T1	**Medição dos sensores**	**5**
T2	**Cálculos das medições**	**4**
T3	**Atuadores**	**7**
T4	**Envio/Receção BLE**	**7**
T5	**Servomotor/Janela**	**4**

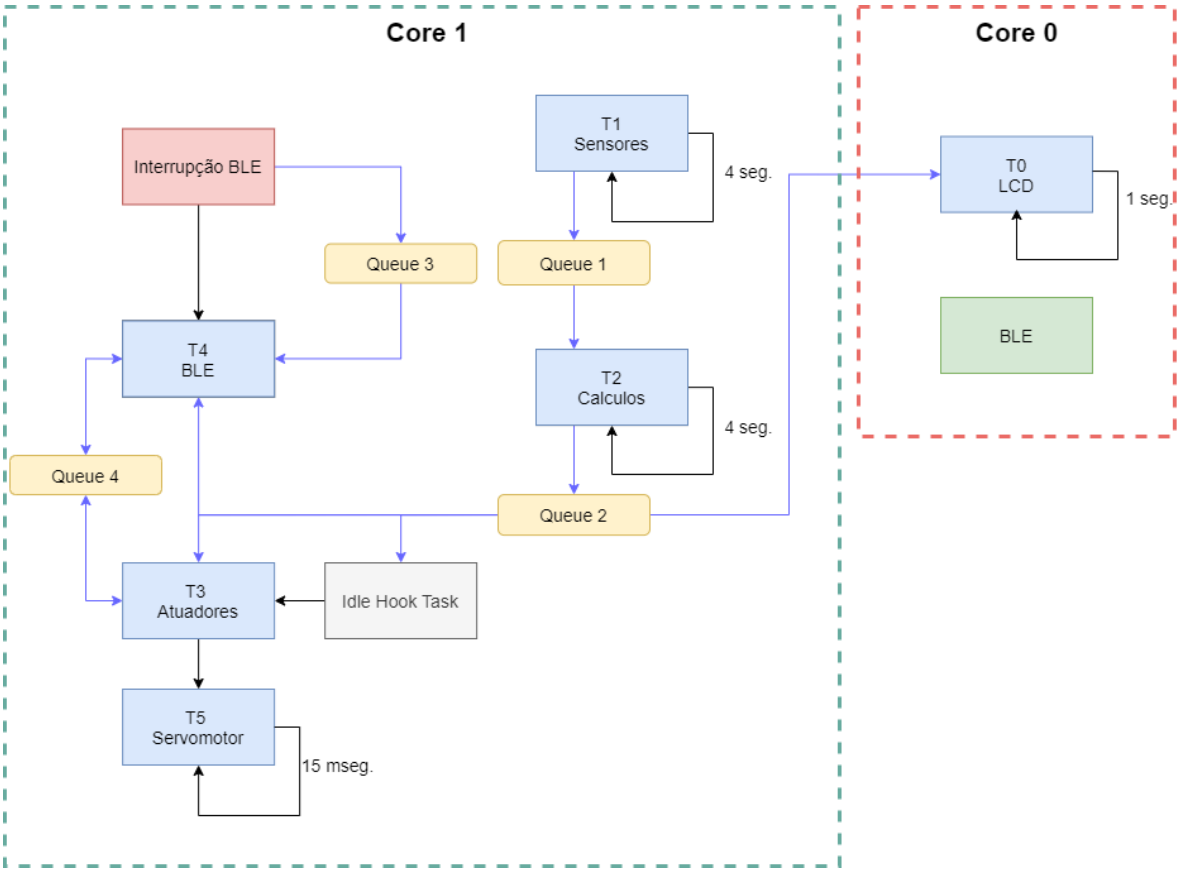


Figura 1 - Esquema Software

Queues

Os queues utilizados têm como máximo 5 espaços, mas têm dados de diferentes tamanhos.

- O *Queue 1* utiliza a estrutura *thlr*, esta estrutura guarda o valor retirado dos sensores.
- O *Queue 2* utiliza outra estrutura, *thl*, onde guarda o valor dos sensores já processados para a grandeza física.
- O Queue 3 utiliza um *array* de *char*[5], onde guarda a informação da comunicação BLE.
- O Queue 4 utiliza uma estrutura de variáveis de tipo *bool*, e armazena as flags dos estados dos atuadores.

As estruturas *thlr* e *thl* estão visíveis na Figura 2.

```
typedef struct temphumlumraw{
    uint32_t rawtemp = 0;
    uint32_t rawhum =0;
    uint32_t rawlum1 =0;
    uint32_t rawlum2= 0;
}thlr;

typedef struct temphumlum{
    float temp =0.0;
    float hum =0.0;
    float lum =0.0;
}thl;

struct flags{
    bool t29 = false, t25 = false, t20 = false, t26 = false;
    bool hl75400 = false, hl50400 = false;
};
```

Figura 2 - Estruturas dos Queues

Idle Hook Task

Foi utilizado o *Idle Hook Task* que verificar se as variáveis dos sensores são superiores ou inferiores a uns limites pré-definidos, junto com as *flags* da *queue 4*. Aqui é criada uma tarefa dos atuadores (T3) que, sendo a tarefa com maior prioridade de todas, corre imediatamente.

Callbacks

Para a comunicação BLE são utilizados dois diferentes *callbacks*. O primeiro *callback* é relativamente simples é causado quando um dispositivo liga ou desliga do ESP32. O segundo *callback* é utilizado para enviar e receber dados.

Tarefas

1. Tarefa 0 – Atualização LCD

A T0 está a correr periodicamente a cada segundo, dentro do *core 0.* É daqui que é atualizado o LCD (Figura 3). Esta tarefa corre no *core* 0 porque o LCD mostra horas, minutos e segundos e se esta tarefa não poder correr por causa de outras tarefas com maior prioridade o tempo a ser mostrado não iria ser correto.

Para atualizar os estados dos atuadores a ser mostrados no LCD são verificadas *flags* dos atuadores ou é verificado o estado do pino correspondente (*HIGH* ou *LOW*).

Esta tarefa também limpa o *Queue 2*, ela verifica se o *Queue 2* tem dados novos e recebe (elimina) os antigos, mantendo dentro do *Queue 2* sempre uma estrutura de dados para ser utilizada nas outras tarefas.

Figura 3 - LCD

2. Tarefa 1 - Medição dos sensores

A T1 é uma tarefa periódica com um período de 4 segundos, aqui é lido o valor dos sensores de temperatura, humidade e luminosidade a partir do protocolo *I2C* e os valores dos mesmos são colocado no *Queue 1*.

Nesta tarefa também é aumentada a prioridade da tarefa 2 (Cálculos das medições) para 6 se a *Queue 2* estiver vazia, ou seja, que não há valores calculados na *Queue 1* e desta maneira a tarefa 2 tem mais prioridade que a tarefa 1.

3. Tarefa 2 - Cálculos das medições

A T2 também é uma tarefa periódica com um período de 4 segundos. Nesta tarefa é realizado os cálculos de cada grandeza. A tarefa começa por ir buscar variáveis ao *Queue 1* em que a tarefa 1 colocou os dados medidos e converte-os para graus Celsius, humidade relativa e lux. Se a *Queue 1* estiver vazia a tarefa 2 baixa a sua prioridade para a original, assim a tarefa 1 volta a ter mais prioridade que a tarefa 2.
Os cálculos foram colocados numa zona critica.

- **Temperatura:**

$$T\ (^0C) = \frac{175{,}72 * Temp\_Sensor}{65536} - 46{,}85$$

- **Luminosidade:**

$$LUX = \frac{(Ch1\_sensor) + (Ch2\_sensor)}{2}$$

Para facilitar os cálculos decidiu-se fazer uma média dos dois canais de luz, as variáveis ALS_GAIN e ALS_INT (ganhos internos) são definidas como *default*.

- **Humidade:**

$$RH\% = \frac{125 * RH\_Sensor}{65536} - 6$$

4. Tarefa 3 – Atuadores

A T3 é criada na *Idle Hook Task* como já antes foi descrito com a prioridade de 7.
Dependendo de qual grandeza se encontra fora do seu limite superior ou inferior é ativado/desativado o atuador físico correspondente.
São utilizadas condições e as *flags* do *queue* para evitar que o sistema continuamente este a ligar ou desligar algum atuador se o mesmo já se encontra no estado desejado.
No final, são enviados os novos estados das *flags* ao *queue*.

Para melhor compreensão do funcionamento dos atuadores da estufa, o sistema é descrito com os seguintes diagramas (Figura 4 e Figura 5):

- **Sistema de regulação da temperatura:**

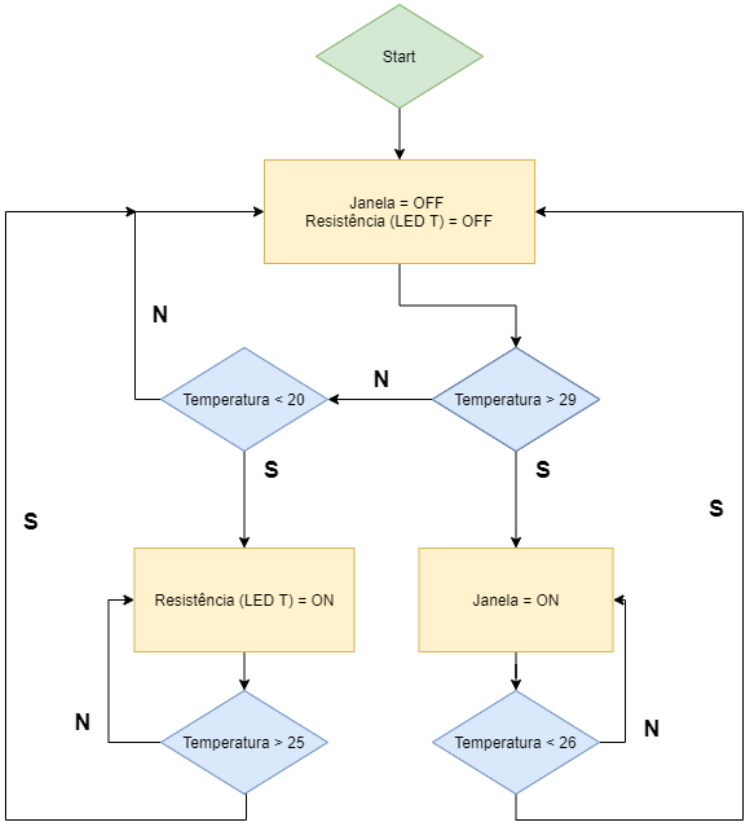


Figura 4 - Diagrama Regulação de Temperatura

A atuação do servomotor e da resistência de aquecimento só dependem do valor da temperatura, se esta estiver abaixo de 20º C ativa a resistência e quando subir para os 25ºC desliga. Se a temperatura estiver acima dos 29ºC ativa o servomotor, dando-lhe um grau de abertura (ex. 180º), quando a mesma baixar para os 26ºC fecha a janela.

- **Sistema de regulação da humidade:**

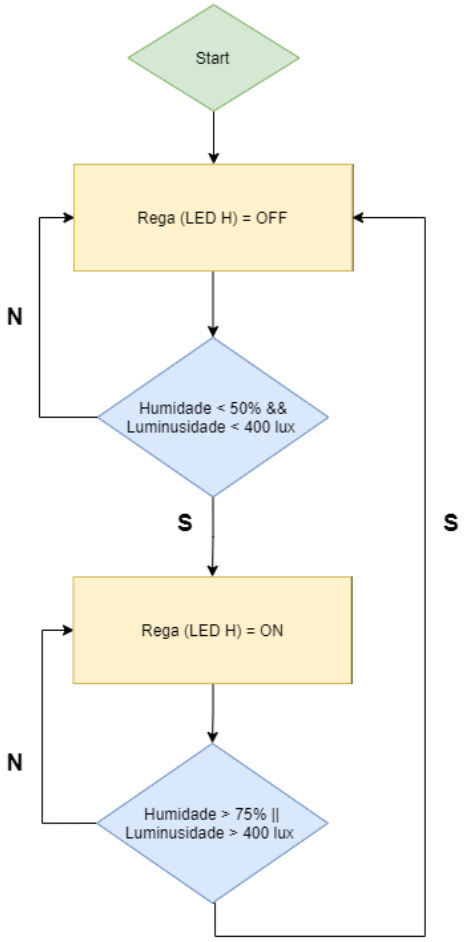


Figura 5 - Diagrama Regulação Humidade

Para ativar o sistema de rega, será preciso que a humidade da terra esteja abaixo de 50% e que a luminosidade não seja superior a 400 lux (luminosidade de amanhecer), porque não se deve regar à tarde. Para desativar o motor de rega, basta que a humidade suba para os 75%.

5. Tarefa 4 - Envio/Receção BLE

A tarefa 4 é criada numa interrupção BLE, que acontece quando há alguma comunicação a partir do dispositivo móvel. Dentro da interrupção, o ESP32 pode escrever ou ler dados dependendo do comando recebido. Se há um comando para receber dados, dentro da interrupção são transformados numa *array* de *char*, e colocados na *Queue 3*. A seguir é utilizado um *semaphore* para correr a tarefa do BLE e controlar o sistema com base no código recebido.

O sistema pode enviar os dados de temperatura, humidade e luminosidade sempre que recebe *'t'*, *'h'*, ou 'l' respetivamente. Para receber os dados das grandezas esta tarefa utiliza a função *QueuePeek* para receber e não eliminar os dados do *Queue2*.

Para poder controlar os atuadores de forma manual é necessário primeiro bloquear o sistema de controlo automático (bloquear a criação da tarefa 3), para isso tem de ser enviado o comando *'m'* para ativar o controlo manual, e a seguir podem ser utilizados os comandos *'ton'* ou *'toff'* para abrir ou fechar a janela, *'ron'* ou '*roff'* para ligar ou desligar o sistema de rega, e finalmente *'aon'* ou '*aoff'* para controlar a resistência de aquecimento. Para sair do modo manual, o dispositivo volta a enviar um *'m'*, e logo são colocadas as flags do *queue* 4 a *false*.

O envio dos comandos é realizado por parte da app android (Figura 6).

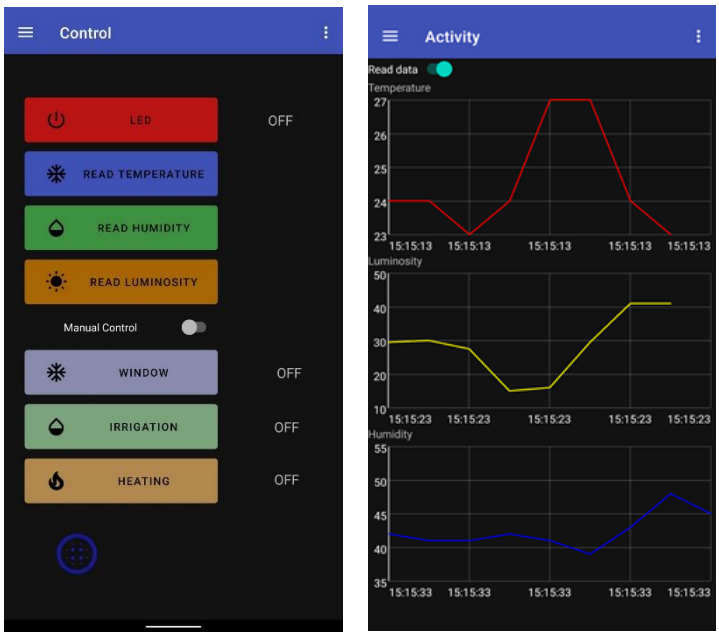


Figura 6 - Aplicação Android

6. Tarefa 5 – Servomotor

Para o controlo do servomotor é utilizada outra tarefa para permitir que o sistema interrompa o seu processo, se for necessário. O servomotor utiliza um ciclo para aumentar periodicamente os graus de abertura/feixe com um período de *15ms* até atingir os limites (0 ou 180). Quando chegar a um dos limites a tarefa 5 elimina-se.

O seu controlo é realizado com a biblioteca *"ESP32Servo.h".*

Controlo dos Queues

Para controlar a quantidade de dados dentro dos *queues*, especificamente dos *Queue 1* e 2 é utilizado *uxQueueMessagesWaiting()* para controlar a prioridade da tarefa 2.
Se há mais do que 1 dado dentro do *Queue 1* a tarefa 2 não reduz sua prioridade para a seguir correr novamente com alta prioridade e retirar os dados extra do *Queue 1* e colocá-los no *Queue 2*.
Noutro caso, se há mais do que 1 dado dentro do *Queue 2*, não é necessário aumentar a prioridade da tarefa 2, porque primeiro as outras tarefas têm que retirar os dados do *Queue 2*, antes da tarefa 2 colocar mais dados dentro da mesma.
Quando é criada a tarefa 3, retira uma estrutura da *Queue 4* (*queue* com as *flags* para os atuadores) e utiliza-a para comparar com o estado atual dos sensores, após ter efetuado as mudanças necessárias, atualiza a mesma com as *flags* atuais. A *Queue* 4 também é utilizada na tarefa 4 (BLE), quando alteramos do estado manual para o automático é retirado uma estrutura da *queue* e atualizado com uma estrutura de *flags* a false.

Exemplo de funcionamento

Na Figura 7 mostra o funcionamento teórico do sistema. (A duração de cada tarefa não é referenciada em termos práticos)

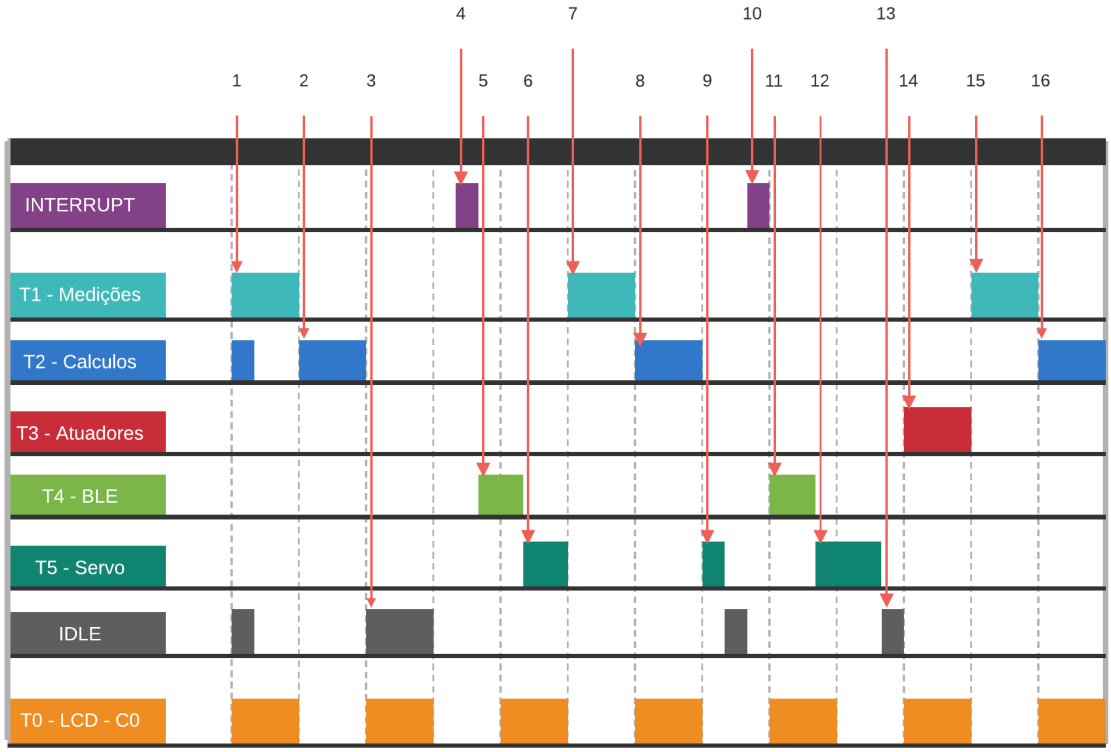


Figura 7 - Exemplo de funcionamento

Legenda Figura 7:

1. A tarefa com maior prioridade corre, neste caso é a tarefa 1 e a tarefa 2 fica á espera.
2. A tarefa 2 pode correr.
3. Não há mais tarefas que precisam correr, então a tarefa *Idle* corre.
4. Acontece uma interrupção por o BLE, é entregue ao semáforo.
5. A tarefa 4 pode correr, e ativa o servomotor criando a tarefa 6.
6. Tarefa 6 inicia a abrir a janela.
7. A tarefa 6 é interrompida por a tarefa 1, fica a espera para terminar seu processo.
8. A tarefa 2 corre com prioridade elevada pela tarefa 1.
9. A tarefa 6 pode completar seu processo.
10. Há outra interrupção, entregue ao semáforo.
11. A tarefa 4 corre novamente, e ativa o servomotor criando a tarefa 6.

12. A tarefa 6 corre completamente, fechando a janela.
13. A tarefa Idle verifica que os dados estão fora dos limites, cria a tarefa 3
14. A tarefa 3 corre.
15. O sistema mede novos dados. A tarefa 1 corre.
16. A tarefa 2 corre.

Anexos

Código
Vídeo: https://youtu.be/WT9ojvMCGUo