

Sistemas Eletrónicos

2º Semestre 2023/2024 (P3)

μOscilloscope

Osciloscópio através de um sistema embebido

(3º trabalho de laboratório)

INSTITUTO SUPERIOR TÉCNICO

Departamento de Engenharia Eletrotécnica e de
Computadores Área Científica de Eletrónica

Histórico de revisões

Data	Versão	Descrição	Autores
Março 2022	1.0	Versão inicial	Pedro Vítor
Março 2023	1.1	Versão de 2022/2023	Pedro Vítor
<u>Março 2024</u>	1.2	Versão de 2023/2024	Pedro Vítor

Índice

1	Introdução	3
1.1	Enquadramento	3
1.2	Objetivos do trabalho	3
2	Hardware base	3
2.1	Módulo IoT	3
2.2	Placa de interface	4
2.3	ADC do módulo IoT	5
2.4	Display TFT	6
3	Software base	6
3.1	Linguagem de programação e bibliotecas	6
3.2	Programa a desenvolver e bibliotecas	7
3.3	Organização do programa	8
3.4	Módulo de simulação	9
3.5	Exemplos de programas	10
4	Especificação do trabalho	13
4.1	Especificações do programa a desenvolver	13
4.2	Desenvolvimento e teste do programa	18
4.3	Relatório	18
4.4	Classificação	18
5	Referências	19
	Anexo 1 – Esquema da placa microcontroladora	20
	Anexo 2 – Esquema da placa de interface	21
	Anexo 3 – Resumo linguagem Python – “Python Cheat Sheet”	22
	Anexo 4 – Instalação do python e das bibliotecas	23
	1. Procedimento de instalação do Python 3.9.9 num PC Windows	23
	2. Instalação das bibliotecas essenciais	23
	3. Descrição do módulo de interface com o display.	24

1 Introdução

1.1 Enquadramento

Um sistema embebido (*embedded system*) é uma combinação de um hardware computacional e um software projetado para uma função específica. Em termos de hardware um sistema embebido consiste essencialmente em sensores, processadores, portas I/O digitais, portas série, conversores A/D e conversores D/A.

Um sensor lê entradas externas, converte-as em dados lidos pelo processador e o processador converte esses dados em informação útil.

A tecnologia da Internet das Coisas (IoT – *Internet-of-Things*), que se afigura com enorme potencial no futuro, corresponde a uma rede coletiva de dispositivos ligados, que facilitam a comunicação entre os dispositivos e a nuvem, bem como entre os próprios dispositivos. Por isso a IoT representa um tipo de sistema embebido ligado à internet.

1.2 Objetivos do trabalho

O projeto μOscilloscope consiste no desenvolvimento de uma aplicação de software para implementar um pequeno osciloscópio, utilizando um hardware para desenvolvimento de soluções IoT.

Na realização deste trabalho pretendem-se atingir os seguintes objetivos:

1. Aprendizagem do circuito de hardware base para a realização do trabalho;
2. Desenvolvimento do software para implementação do μOscilloscope;
3. Teste e ajuste do software;
4. Calibração do μOscilloscope.

2 Hardware base

2.1 Módulo IoT

O módulo IoT é composto, entre outros, por uma placa micro-controladora, baseada num

processador da família ESP32 [1], chipset ESPRESSIF-ESP32 240MHz Xtensa de 32 bit, com flash de 4MB, memória SRAM de 560kB, três botões (um de *reset* e dois programáveis), um conjunto de interfaces modulares (UART, SPI, SDIO, I2C, LED PWM, I2S, IRGPIO, sensor “capacitive touch”, ADC e DAC), wi-fi, Bluetooth, bateria de iões de lítio (*Li-ion*) recarregável, ligação USB Tipo-C para carregamento do software/debug e carregamento da bateria, display TFT a cores de 1.14” e uma placa de interface para entrada de sinais (Figura 1).

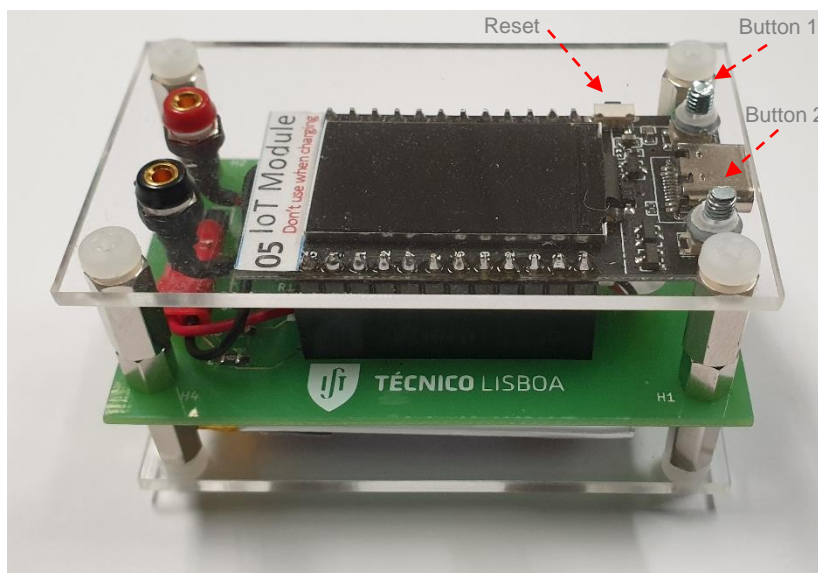


Figura 1 – Módulo IoT.

Nos Anexos 1 e 2 apresentam-se os esquemas elétricos da placa microcontroladora e da placa de interface.

2.2 Placa de interface

A placa de interface possui como entrada a ficha J3 com dois bornes por onde se liga o sinal do canal do μOscilloscope, conforme esquemático apresentado no Anexo 2:

- Pino 1 de J3 positivo/vermelho,
- Pino 2 de J3 negativo/preto.

Possui ainda um conjunto de circuitos eletrónicos para interface com o ADC:

- Fichas de interligação com a placa microcontroladora (J1 e J2);
- Circuito integrado LM324 com dois AMPOPs para gerar tensões DC de referência de 1V (U1A) e 2V (U1B);
- Dois transístores funcionando como interruptores para implementação de divisores resistivos, permitindo criar várias escalas para o sinal de entrada (Q1 e Q2);

- Um transistor funcionando como interruptor (Q3), para a leitura do sinal de referência (1 Volt);
- Dois díodos de proteção (D1 e D2) ligados ao pino de entrada do ADC para o proteger, onde, caso se considere que os díodos têm um $V_f = 0.7V$, a tensão do ADC fica limitada a $-0.7V < V_{ADC} < 2.7V$.

Para que o circuito permita a leitura de sinais de tensão positiva e negativa, e considerando que o conversor ADC tem 12 bits, convertendo tensões entre 0 e 2V, o terminal negativo do sinal de entrada liga-se a uma fonte de tensão de 1V, garantindo que a conversão realiza-se para tensões de entrada entre -1V ($V_{ADC} = 0V$) e +1V ($V_{ADC} = 2V$). De referir que o terminal positivo da entrada não é ligado diretamente ao ADC, mas sim a um conjunto de divisores de tensão, para permitir gamas de tensão de medida superiores ao intervalo $(-1V, +1V)$.

Os divisores resistivos serão constituídos por:

1. R_1 e $R_5 + R_6 + R_7$ caso Q_1 e Q_2 ao corte $Fator_1 = \frac{R_5+R_6+R_7}{R_1+R_5+R_6+R_7} = 0.509 = \frac{1}{1.97}$
2. R_1 e $R_5 + R_6$ caso Q_1 a conduzir e Q_2 ao corte $Fator_2 = \frac{R_5+R_6}{R_1+R_5+R_6} = 0.0342 = \frac{1}{29.3}$
3. R_1 e R_5 caso Q_2 a conduzir e Q_1 ao corte $Fator_3 = \frac{R_5}{R_1+R_5} = 0.00239 = \frac{1}{418}$

Pretende-se que o μOscilloscope funcione na segunda escala (Q1 a conduzir e Q2 ao corte), permitindo leituras de tensões entre $-30V$ e $+30V$ aproximadamente. Esta função é selecionada automaticamente através do módulo de software de biblioteca, sendo:

$$V_{ADC} = Fator_2 V_I + V_{1V} = Fator_2 V_I + 1$$

e V_{ADC} pode variar entre 0 e 2V, ou seja, $-29.3V < V_I < +29.3V$.

2.3 ADC do módulo IoT

Foram realizadas medidas num módulo IoT que conduziram ao gráfico da Figura 2.

Embora seja necessária uma calibração caso a caso para se obter maior precisão, como primeira aproximação, a fórmula para obtenção do valor digital lido pelo ADC (D) em função da tensão de entrada no ADC (V_{ADC}) será:

$$D = 2271.27V_{ADC} - 207.72 \quad 0.091455V < V_{ADC} < 1.8944V$$

A fórmula inversa, para obter a tensão do ADC (V_{ADC}) em função do valor digitalizado (D) é:

$$V_{ADC} = 0.00044028D + 0.091455 \quad 0 \leq D \leq 4095$$

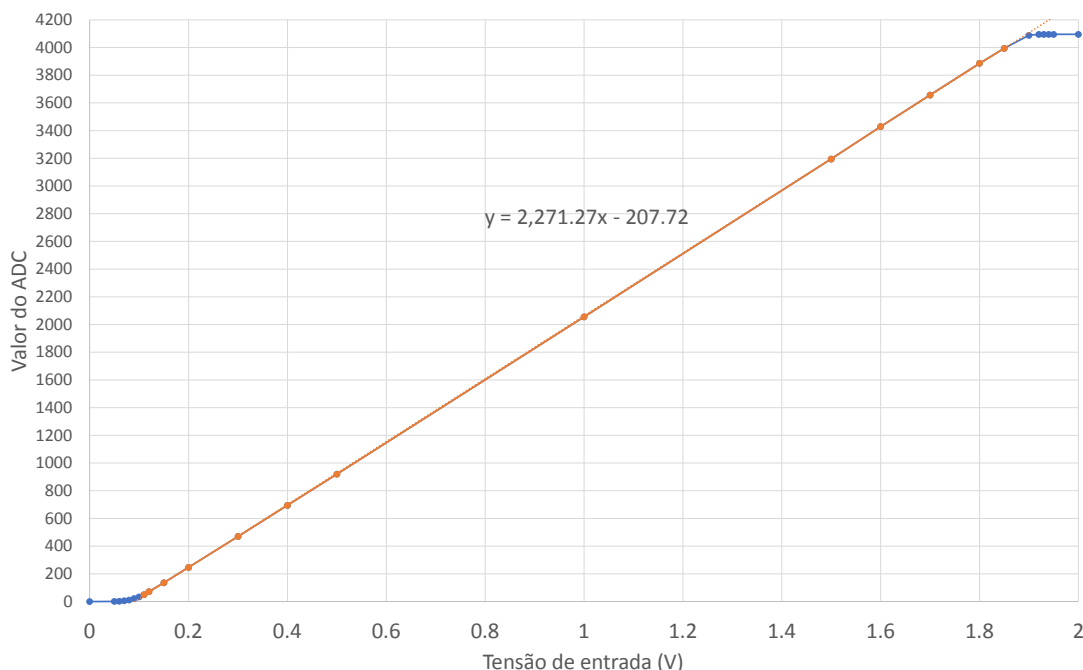


Figura 2 – Medidas da resposta do ADC para um módulo IoT.

2.4 Display TFT

O display TFT tem uma resolução de 240x135 pixéis, em que cada pixel é endereçado através de coordenadas x e y, conforme representado na Figura 3.

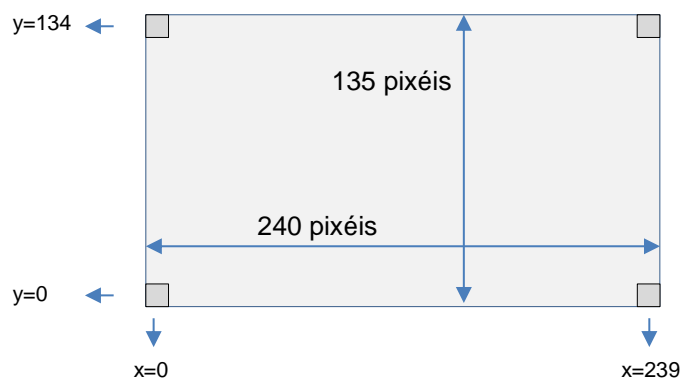


Figura 3 - Display TFT – Endereçamento dos pixéis.

3 Software base

3.1 Linguagem de programação e bibliotecas

O software a desenvolver deverá ser realizado em linguagem Python 3 [2], que face à sua proximidade com a linguagem “C” e pelo facto de o programa a desenvolver ser bastante simples, será muito facilitado o desenvolvimento da aplicação. O programa irá correr no módulo IoT que tem como sistema operativo uma forma simplificada da linguagem Python,

designada micropython, e para facilitar o desenvolvimento de programas foi realizado um simulador que permite que o programa corra num PC contendo Python 3.

O Anexo 3 contém um documento resumo com as principais instruções da linguagem Python, designado “Python Cheat Sheet”. Para consulta de informação mais aprofundada recomendam-se ainda as seguintes páginas Web:

- Tutorial do organismo python.org – <https://docs.python.org/3.9/tutorial/index.html>
- Python Syllabus, lições de temas específicos – <https://www.guru99.com/python-tutorials.html>
- Questões e respostas sobre Python – <https://stackoverflow.com/questions/tagged/python>

O Anexo 4 descreve como instalar a linguagem de programação e as bibliotecas utilizadas, estando organizado da seguinte forma:

1. Procedimento de instalação do Python 3.9.9 (versão recomendada) num PC Windows;
2. Instalação das bibliotecas essenciais;
3. Descrição do módulo de interface, que é a biblioteca que faz a ligação com o hardware do módulo IoT (display, botões e WiFi).

O módulo de interface baseia-se num objeto denominado TFT, que é acedido através dos seguintes métodos:

- **working()** – Devolve True enquanto o programa estiver a correr e False no final;
- **readButton()** – Leitura do estado dos botões;
- **display_set()** – Preenche retângulo com uma cor;
- **display_pixel(), display_npixel()** – Preenche um pixel ou n pixels com uma cor;
- **display_write_str()** – Representa uma string em Arial 16 com uma cor;
- **display_write_grid()** – Representa grelha retangular vertical/horizontal para gráfico;
- **get_color()** – Obtém cor a partir de um valor de RGB (vermelho/verde/azul);
- **read_adc()** – Lê conjunto de valores do conversor analógico digital (ADC);
- **send_mail()** – Envio de pontos lidos pelo ADC por mail para endereço ou endereços;
- **set_wifi_icon()** – Coloca no display o icon representando o estado do WiFi.

3.2 Programa a desenvolver e bibliotecas

O programa a desenvolver deve basear-se num ficheiro com a designação “main.py”, o qual deve estar no computador simulador num diretório em conjunto com os ficheiros de biblioteca e auxiliares. Para correr no ambiente real do laboratório será apenas este ficheiro “main.py” (sem qualquer alteração) que deve ser carregado para o módulo IoT, através da ligação WiFi.

No ambiente de simulação o diretório deverá conter os seguintes ficheiros:

main.py	Programa Principal (igual no módulo e no simulador)
main_exemplo1.py, main_exemplo2.py	Dois exemplos de ficheiro para o Programa Principal
T_Display.py	Módulo de interface com o display
T_Simulator.py	Simulador para o PC
T_Display1.dat ... T_Display5.dat	Ficheiros de dados auxiliares
arial_16.py	Fonte Arial com 16 pixéis por caracter (a única fonte que pode ser usada)

3.3 Organização do programa

O módulo de interface com o display deve ser carregado no início do programa principal, através de uma linha de importação de biblioteca:

```
import T_Display
```

Na sequência deverão ser importadas outras bibliotecas utilizadas pelo programa, compatíveis com o software micropython (sistema operativo do módulo IoT), nomeadamente as bibliotecas `time` e `math` (funções matemáticas):

```
import time  
  
import math
```

De seguida deve ser instanciado um objeto da classe `TFT`, que está definido no módulo `T_Display` (ficheiro `T_Display.py`), por exemplo usando a variável `tft`:

```
tft = T_Display.TFT()
```

Opcionalmente pode ser usado um argumento na criação desta classe, o qual será o código de autorização utilizado por defeito no módulo de simulação para upload do programa para o módulo IoT (apresentado em §3.4), como no exemplo em que o código de autorização por defeito seja `01_Test`:

```
tft = T_Display.TFT("01_Test")
```

Toda a informação de e para o módulo IoT é acedida através deste objeto, com recurso a um conjunto de métodos, cujo detalhe será apresentado no ponto 3. do Anexo 4. Tal como referido no ponto 2. do Anexo 4 **não poderão ser instaladas nem utilizadas outras bibliotecas além das duas indicadas (PySide2 e requests), pelo que caso sejam utilizadas outras bibliotecas, nomeadamente numpy, pandas e Matplotlib, poderão correr no ambiente do PC mas não correrão no módulo IoT.**

Atendendo a que o módulo IoT tem uma memória RAM limitada (menos de 100kB disponíveis)

recomenda-se uma utilização moderada de variáveis de memória, o uso da função `gc.collect()`, importando previamente este módulo (`import gc`) e apagar variáveis quando já não são utilizadas (Ex: `del x,y,z`, para apagar as três variáveis).

3.4 Módulo de simulação

De forma a se poder fazer uma preparação prévia do trabalho fora do laboratório e sem o módulo IoT, foi desenvolvido um módulo Python de simulação designado `T_Simulator.py`.

A Figura 4 mostra a janela da aplicação do simulador, com 3 áreas principais:

1. **Módulo IoT** – Zona de simulação do módulo IoT, formada pelo display onde é representada a informação e pelos botões (simulados através do rato do PC);
2. **Gerador de sinais** – Gerador de sinais usado como entrada do ADC do simulador;
3. **Upload** – Envio do ficheiro em execução para a cloud (usando um código de autorização) e após reset do módulo IoT o ficheiro irá substituir o ficheiro `main.py` existente no módulo.

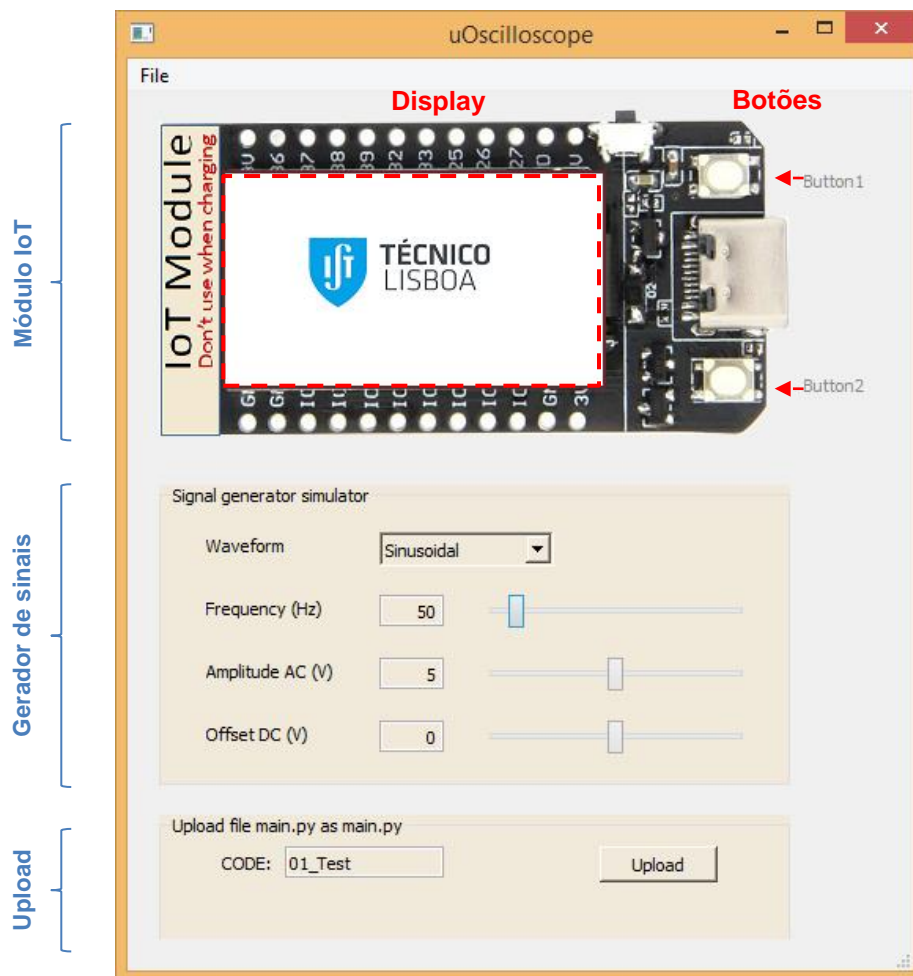


Figura 4 - Janela do simulador.

O software principal, do ficheiro main.py, funciona de modo transparente, sem qualquer alteração em ambos os ambientes:

1. No módulo IoT funciona em modo real, em que os sinais lidos pelo ADC são uma divisão ($Fator_2 = 0.0342 = 1/29.3$) do sinal colocado na entrada do μOscilloscope, entre o Pino 1 de J3 (positivo/vermelho) e o Pino 2 de J3 (negativo/preto).
2. No simulador funciona em modo de simulação, sendo o sinal de entrada do ADC dado pelo “Gerador de sinais” (Figura 4).

Durante o projeto e teste do programa deverá ser utilizado um editor de texto, recomendando-se o notepad++ [3], ou em alternativa um IDE que já conheça, como é o caso do PyCharm [4] Spyder [5], Visual Studio Code [6] ou Thonny [7]. Utilizando um editor de texto para correr o programa main.py, deverá executar na linha de comandos:

```
C:\>python main.py
Start of T_Display: Version 1.03
Starting T-Display
...
```

3.5 Exemplos de programas

De seguida apresentam-se dois exemplos de programas de demonstração.

Exemplo 1 (main_exemplo1.py) – Quando é premido o Botão 1 (click) lê uma amostra de 240 pontos, num intervalo total de 200ms, converte-os em tensões, determina os valores máximo, mínimo e médio, representando-os no display.

```
# Exemplo 1 (main_exemplo1.py)
import T_Display

# Inicializações
pontos_volt = [0.0]*240 # Lista com 240 floats

# Função de leitura dos valores do ADC e representação no display
def read_and_display():
    pontos_adc = tft.read_adc(240, 200) # Lê 240 pontos do ADC em 200ms
    Vmax=0
    Vmin=0
    Vmed=0
    for n in range(240):
        V = 0.00044028 * pontos_adc[n] + 0.091455 # Converte valor do ADC em Volt
        V = V - 1 # Tensão entrada de referência de 1V
        V = V / fator # Entra com o efeito do div. resistivo
        pontos_volt[n] = V
        if n==0: # Caso seja o primeiro ponto
            Vmax = Vmin = Vmed = V
        else:
            Vmed += V
            if V>Vmax: Vmax=V
            if V<Vmin: Vmin=V
    Vmed /= 240 # Divide pelo número de amostras

# Escreve os valores em strings com 2 casas decimais
```

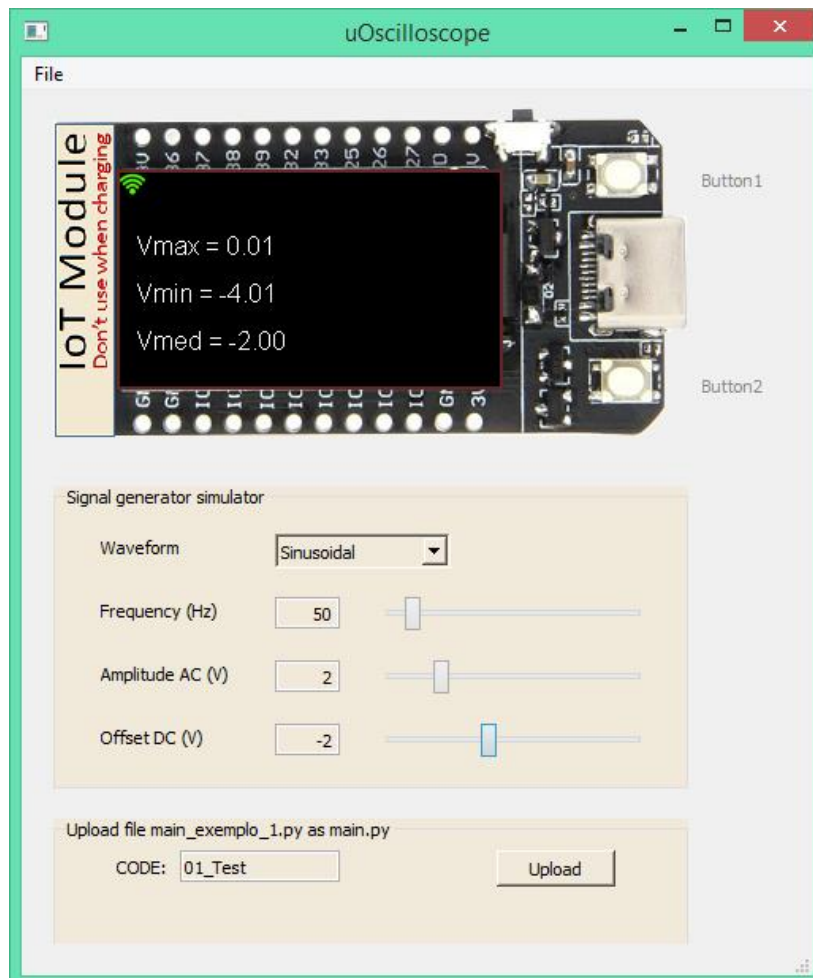
```
str1 = "Vmax = %.2f" % Vmax
str2 = "Vmin = %.2f" % Vmin
str3 = "Vmed = %.2f" % Vmed

tft.display_set(tft.BLACK, 0, 0, 240, 135)    # Apaga display

# Escreve as strings no display
tft.display_write_str(tft.Arial16, str1, 10, 20+60)
tft.display_write_str(tft.Arial16, str2, 10, 20+30)
tft.display_write_str(tft.Arial16, str3, 10, 20)
tft.set_wifi_icon(0,135-16)                  # Coloca o icon wifi no display

# Programa principal (main)
fator = 1/29.3                               # Fator do divisor resistivo
tft = T_Display.TFT()                        # Instancia um objeto da classe TFT
read_and_display()                           # Função de leitura e representação

while tft.working():                          # Ciclo principal do programa
    but=tft.readButton()                     # Lê estado dos botões
    if but!=tft.NOTHING:
        print("Button pressed:",but)
        if but==11:                          # Button 1 click - Repete função
            read_and_display()
        if but==21:                          # Button 2 click - Envia mail
            tft.send_mail(0.2/240,pontos_volt,"Lista de 240 pontos em 0.2 segundos.",
                           "pvitor@ist.utl.pt")
```



Exemplo 2 (main_exemplo2.py) – Desenha uma grelha de 160x134 pixéis, com 8 intervalos na horizontal (10ms/) e 8 intervalos na vertical (1V/), representando, com uma linha a amarelo, uma onda sinusoidal de 50Hz, com 3V de amplitude e coloca o icon WiFi no canto superior direito. Premindo o Botão 1 (click) faz 10 amostras de 100 pontos (50ms) cada e representa o valor médio com 2 casas decimais no display. Caso seja premido o Botão 2 (click) faz o mesmo mas com 100 amostras.

```
# Exemplo 2 (main_exemplo_2.py)
import T_Display
import math                # módulo de funções matemáticas

# Inicializações de variáveis globais
width = 160
height = 134

# Função para realizar amostras e médias
def media_amostras(num_amostras):
    # Apaga parte direita do display excepto icon WiFi
    tft.display_set(tft.BLACK, width, 0, 240-width, height-16)
    soma=0
    for n in range(num_amostras):
        pontos_adc = tft.read_adc(100, 50)    # 10 amostras num total de 50ms
        for j in range(100):
            soma += pontos_adc[j]
        media = soma / (100*num_amostras)

    # Escreve valores no display
    tft.display_write_str(tft.Arial16, "media", width+5, 90)
    tft.display_write_str(tft.Arial16, "%d" % num_amostras, width+5, 70)
    tft.display_write_str(tft.Arial16, "amostras", width+5, 50)
    tft.display_write_str(tft.Arial16, "%.2f" % media, width+5, 30)

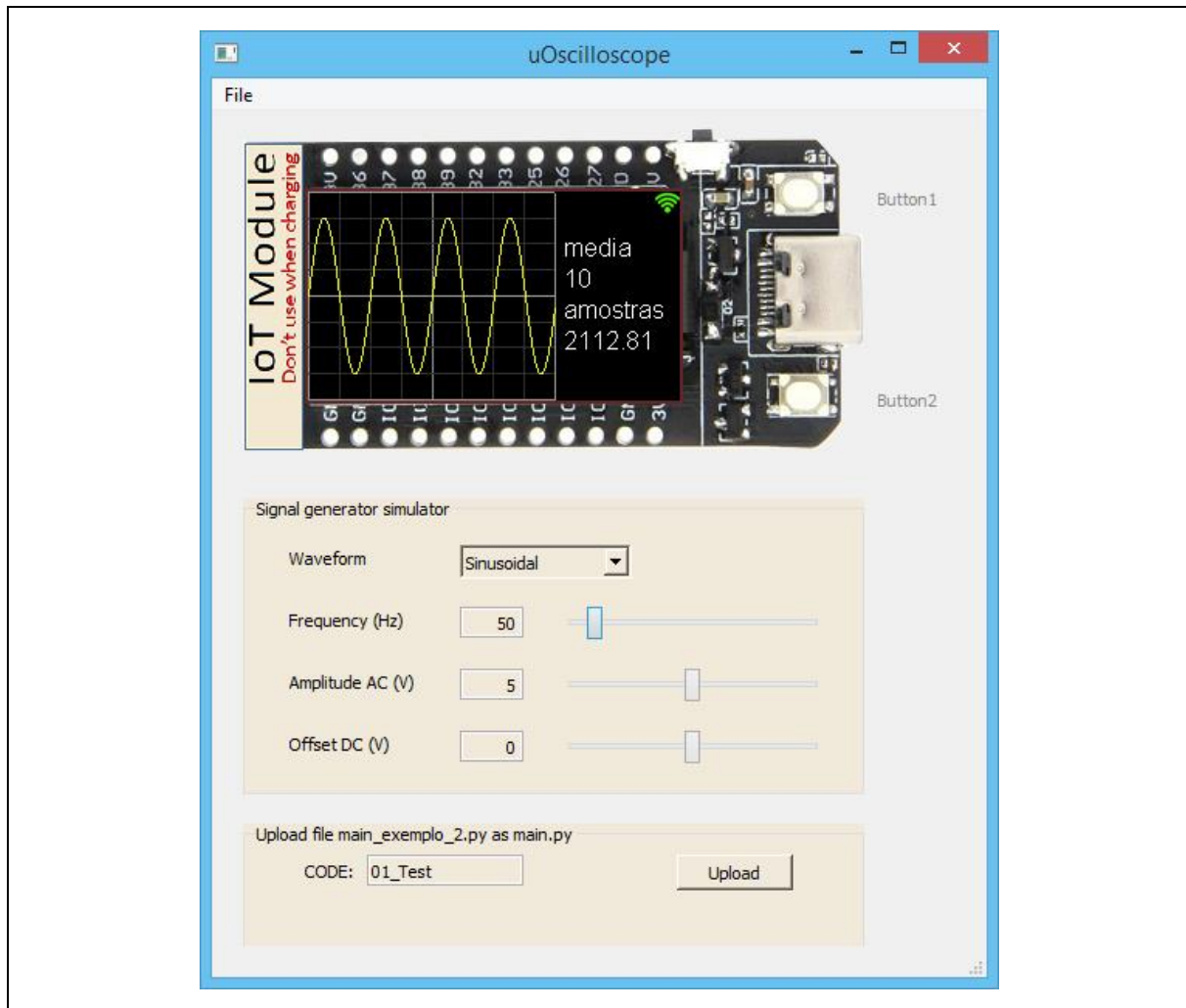
# Programa principal (main)
tft = T_Display.TFT()                # Instancia um objeto da classe TFT
tft.display_set(tft.BLACK, 0, 0, 240, 135)    # Apaga display
tft.display_write_grid(0, 0, 160, 135, 8, 8, True) # Desenha grelha (c/ linhas centrais)

frequencia = 50
amplitude = 3

# HORIZONTAL:
# 160 pixéis corresponde a 80ms (10ms por divisão) - Cada pixel 0.5ms
# VERTICAL:
# 134 pixéis corresponde a 8V (1V por divisão) - Cada volt (134/8) pixéis
x = []
y = []

# Desenha onda sinusoidal
for n in range(width):
    t = n * 0.0005
    volt = amplitude*math.sin(2*math.pi*frequencia*t)    # tensão em volt
    pixel = height/2 + (height/8)*volt
    x.append(n)
    y.append(round(pixel))
tft.display_nline(tft.YELLOW, x, y)
tft.set_wifi_icon(240-16,135-16)

while tft.working():
    but=tft.readButton()                # Ciclo principal do programa
    if but!=tft.NOTHING:
        print("Button pressed:",but)
        if but==11:                    # Button 1 click (10 amostras)
            media_amostras(10)
        if but==21:                    # Button 2 click (100 amostras)
            media_amostras(100)
```



4 Especificação do trabalho

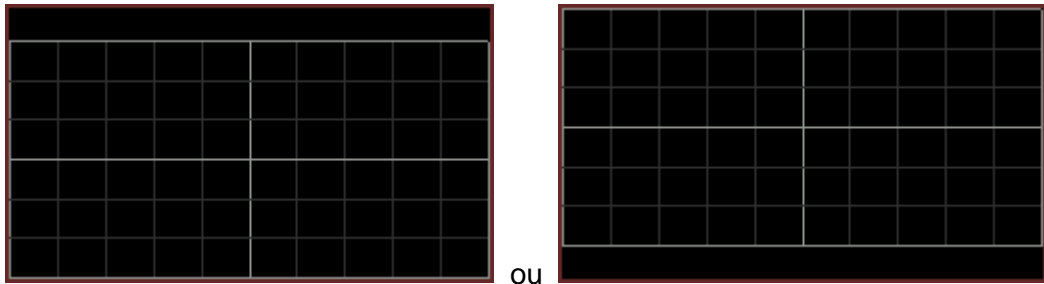
4.1 Especificações do programa a desenvolver

O programa deve realizar um pequeno osciloscópio com as seguintes especificações:

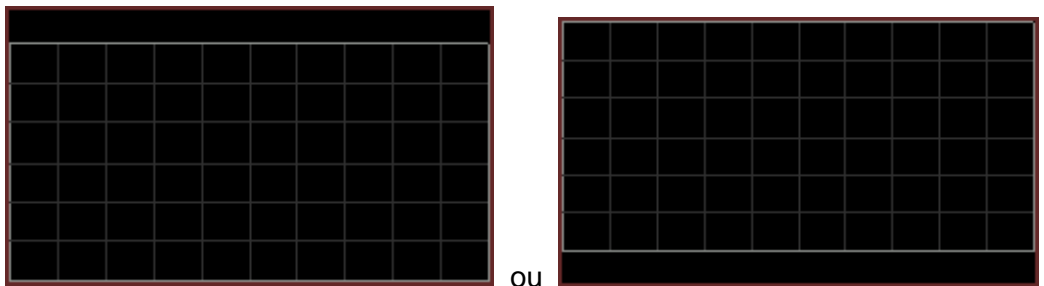
- O display deve conter:
 - Espaço com uma altura de 16 pixéis situado na parte de cima ou na parte de baixo do display para permitir a escrita de uma linha de texto, para representar a informação sobre as escalas atuais (vertical e horizontal) e sobre a ligação Wi-Fi.
 - Grelha para apresentar a forma de onda, com 10 intervalos na horizontal e 6 na vertical, utilizando todo o display exceto um espaço de 16 pixéis situado no

topo ou na parte de baixo do ecrã. A grelha será diferente conforme se pretende apresentar:

Função do tempo (forma de onda) – Grelha com as linhas centrais (horizontal e vertical)



Função da frequência (transformada Fourier) – Grelha sem as linhas centrais (horizontal e vertical)



- Devem ser utilizadas escalas com os seguintes valores por divisão:

Função do tempo (forma de onda):

- Vertical: 1V/, 2V/, 5V/ e **10V/**, sendo 10V/ o valor de arranque do programa (defeito);
- Horizontal: 5ms/, 10ms/, **20ms/** e 50ms/, sendo 20ms/ o valor de defeito.
- A escala vertical de 5V/ significa que as partes de cima e de baixo da grelha correspondem a tensões de 15V e -15V, respetivamente. A escala vertical de 10ms/ significa que o tempo total na horizontal são 100ms.

Função da frequência (transformada de Fourier):

- Vertical: Sendo os valores apresentados como o módulo da transformada de Fourier, serão sempre positivos, pelo que a escala deverá ser dupla da escala que estiver selecionada na representação em função do tempo (0.5V/, 1V/, 2.5V/, 5V/);
- Horizontal: O valor total da escala horizontal decorre do ritmo de Nyquist, ou

seja metade da frequência de amostragem, em que cada um dos 10 intervalos, correspondem a 1/10 desse valor (considerando 240 pontos de amostragem as escalas serão: 240Hz/, 120Hz/, 60Hz/, 24Hz/).

- A escala vertical de 5V/ significa que as partes de cima e de baixo da grelha correspondem a tensões de 30V e 0V, respetivamente. A escala vertical de 60Hz/ significa que a frequência máxima na horizontal é 600Hz.

Para a obtenção do gráfico em função da frequência deve-se utilizar a transformada de Fourier discreta (DFT – Discrete Fourier Transform), onde para uma amostra de N pontos x_n :

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi kn}{N}} = \sum_{n=0}^{N-1} x_n \left[\cos\left(\frac{2\pi kn}{N}\right) - j\sin\left(\frac{2\pi kn}{N}\right) \right] \quad 0 \leq k \leq N-1$$

apresentando a série DFT uma simetria conjugada, em que $X_k = X_{N-k}^*$, resultando que metade dos valores são redundantes pois o espetro obtém-se a partir dos módulos da transformada de Fourier. Assim, define-se o espetro X_{SS_k} do sinal amostrado x_k :

$$X_{SS_k} = \begin{cases} \frac{|X_k|}{N} & k = 0 \quad \rightarrow \text{DC} \\ 2 \frac{|X_k|}{N} & 0 < k < N/2 \\ \frac{|X_k|}{N} & k = N/2 \quad \rightarrow \text{Nyquist} \end{cases}$$

Que tem metade dos pontos mais um ($N/2 + 1$), recomendando-se que, para efeitos de representação no display do módulo IoT, que tem 240 pixéis na horizontal, se utilize uma lista com o mesmo número de elementos que os obtidos a partir do ADC. Assim, tendo o espetro dos pontos ($N=240$), os pixéis deverão ser iguais dois a dois, ignorando-se o último ponto ($k = N/2$):

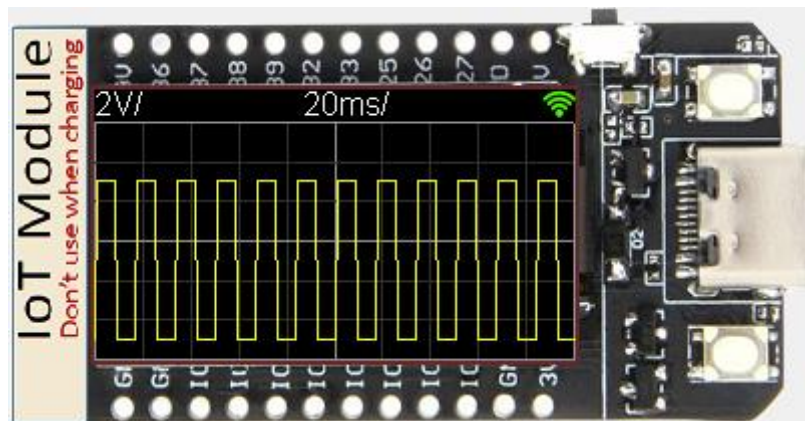
$$P_0 = P_1 = X_{SS_0}, \quad P_2 = P_3 = X_{SS_1} \quad \dots \quad P_{N-2} = P_{N-1} = X_{SS_{N/2-1}}$$

- Em termos funcionais o programa deve seguir os seguintes passos:
 1. Inicializar o display apagando-o;
 2. Desenhar a grelha, as escalas e o icon WiFi;
 3. Realizar uma leitura de valores do ADC, convertê-los para tensões e representar a forma de onda sobre a grelha;
 4. Aguardar que o utilizador prima um botão:

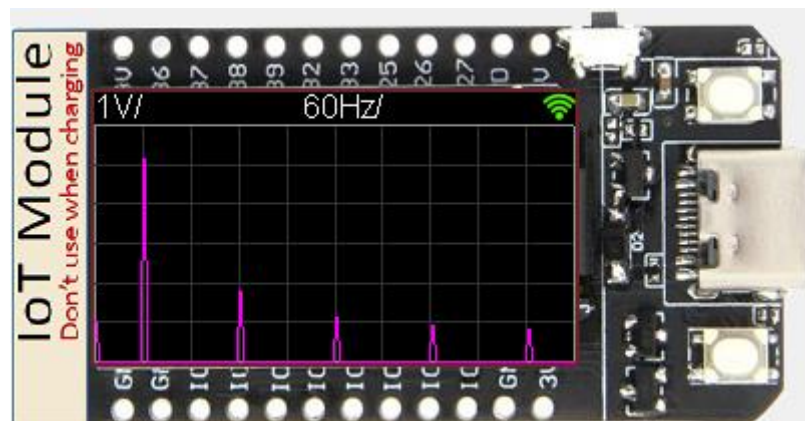
- 4.1. Botão 1 click rápido (11) – Voltar ao passo 1, iniciando uma nova leitura e representando da forma de onda (função do tempo);
 - 4.2. Botão 1 click lento (12) – Enviar por mail os pontos obtidos no passo 3 e voltar ao passo 4, sendo uma tabela de duas colunas, em que a primeira coluna é os tempos em segundos e a segunda as tensões em volt;
 - 4.3. Botão 1 duplo click (13) – Apagar o display e apresentar os valores do conjunto de medidas descritas adiante;
 - 4.4. Botão 2 click rápido (21) – Alterar a escala vertical, passando para a escala imediatamente acima e de forma circular (se for a primeira escala de 1V/ passar para 2V/ e assim sucessivamente, e caso seja a última, voltar à primeira) e seguir o passo 1;
 - 4.5. Botão 2 click lento (22) – Alterar a escala horizontal, passando para a escala imediatamente acima e de forma circular (se for a primeira escala de 5ms/ passar para 10ms/ e assim sucessivamente, e caso seja a última, voltar à primeira) e seguir o passo 1;
 - 4.6. Botão 2 duplo click (23) – Calcular a transformada de fourier da forma de onda obtida na última leitura, apresentando graficamente o espetro (função da frequência).
- Atendendo aos erros inerentes a cada um dos módulos IoT, particularmente da parte de conversão do ADC, deverá ser aproveitado o laboratório para se realizar uma calibração do circuito:
 - A calibração pode ser iniciada através da colocação na entrada do circuito de 5 valores DC pré-definidos (-10V, -5V, 0V, +5V e +10V);
 - Para cada tensão DC de entrada (medida com precisão utilizando um multímetro) deve ser feita uma leitura do valor do conversor ADC com precisão, recorrendo a médias, podendo ser utilizado o programa do Exemplo 2 (ou outro programa desenvolvido pelos alunos), recomendando-se a leitura de 100 amostras de 100 pontos por amostra, correspondendo o valor final a uma média de 10000 pontos.
 - A partir dos valores medidos, devem ser recalculados os parâmetros do gráfico da Figura 2 e introduzidos nas fórmulas de cálculo das tensões a partir dos valores do conversor ADC, completando-se a calibração;
 - O Conjunto de medidas representadas pelo "Botão 2 click rápido", são (esta

informação também deve ser enviada no corpo do mail descrito no ponto 4.2):

- Tensão máxima (Vmax):
- Tensão mínima (Vmin):
- Valor médio (Vav);
- Valor eficaz (Vrms);
- O programa realizado deve assegurar o funcionamento do seguinte exemplo:
 - Onda quadrada de 60Hz, com 4V de amplitude (8Vp.p.) e com uma tensão DC de -1V;
 - Resultado obtido na apresentação da forma de onda em função do tempo, nas escalas de 2V/ e 20ms/:



- Resultado obtido na apresentação do espectro em função da frequência, obtido por duplo click no Botão 2, na sequência da apresentação representada acima:



$$X_{SS_0} = 1, X_{SS_{60Hz}} = 5.113, X_{SS_{180Hz}} = 1.762, X_{SS_{300Hz}} = 1.131, X_{SS_{420Hz}} = 0.898, X_{SS_{540Hz}} = 0.810$$

- Podem ser criadas outras funcionalidades simples, que melhorem o programa, valorizando o trabalho desenvolvido.

4.2 Desenvolvimento e teste do programa

O programa deve ser projetado e testado previamente à aula de laboratório, utilizando o simulador. Na aula de laboratório, deve ser utilizado o módulo IoT para:

1. Testar a apresentação gráfica do programa e as variações das escalas, colocando na entrada uma onda sinusoidal com 10Vp.p. e 25Hz;
2. Obter as imagens de leitura para tensões DC -10V, -5V, 0V, +5V e +10V na escala de 5V/;
3. Realizar a calibração do programa e repetir os pontos 1. e 2.

Quaisquer resultados, sejam através do simulador ou do módulo IoT no laboratório, devem ser recolhidos e apresentados no relatório, seja através das imagens da janela do simulador ou dos ficheiros enviados por mail a partir do módulo IoT, devendo para tal ser utilizados os endereços email de um ou de vários alunos do grupo (separados por vírgulas).

4.3 Relatório

O relatório deve obrigatoriamente respeitar a seguinte estrutura sequencial de secções:

1. **Introdução** – Breve enquadramento e objetivos do trabalho;
2. **Desenvolvimento do programa** – Descrição do programa, o qual deve estar devidamente documentado através de comentários;
3. **Testes no simulador** – Resultados obtidos através do simulador;
4. **Testes no laboratório** – Medidas obtidas no laboratório antes e depois da calibração. Comparação dos dois resultados e comentários quanto à necessidade da calibração;
5. **Conclusões** – Resumo sintético do trabalho, resultados obtidos e principais comentários aos resultados.

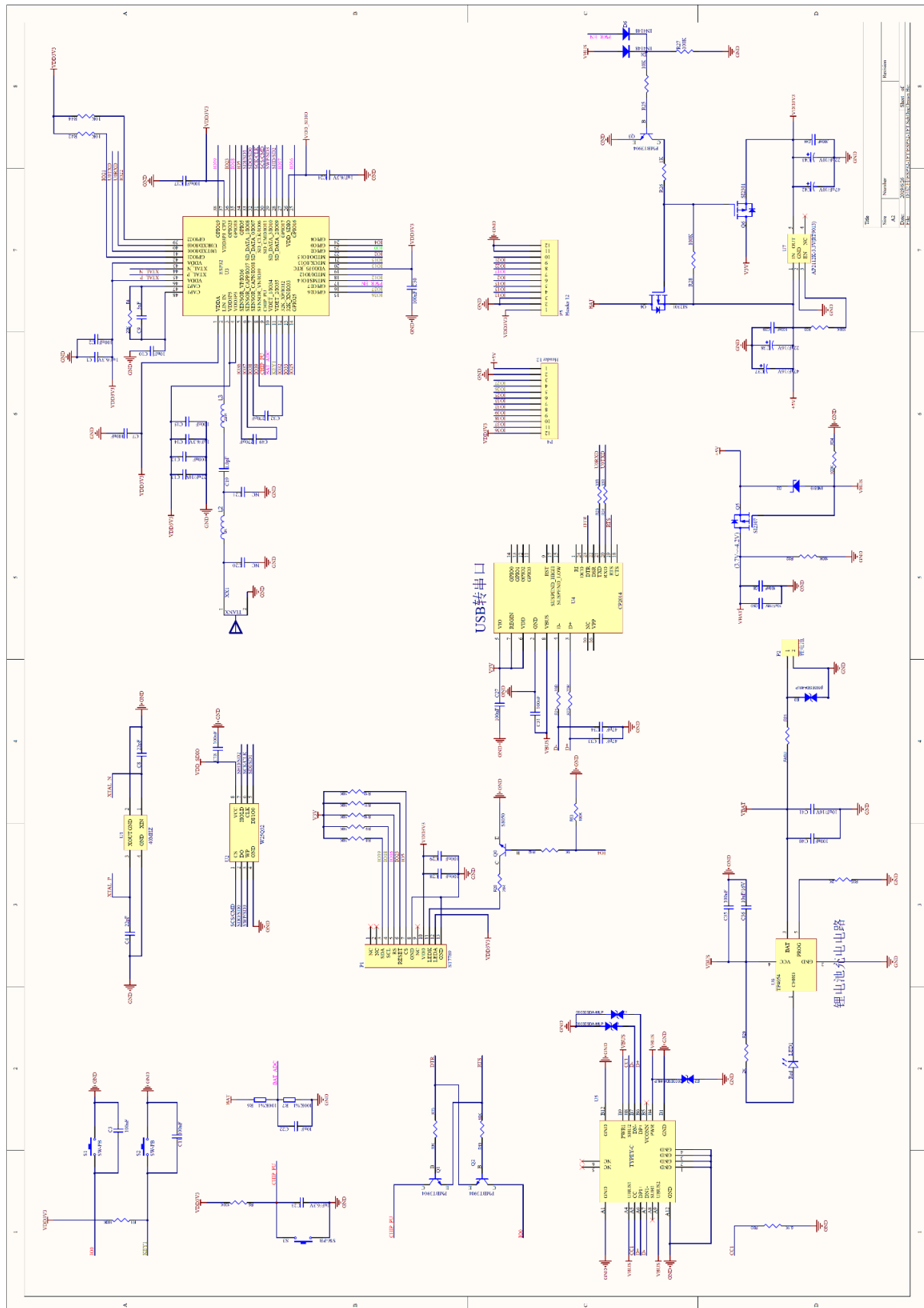
4.4 Classificação

O relatório deve conter no máximo 10 páginas e ter uma estrutura e apresentação cuidada, que corresponderá a **10%** na sua avaliação. A parte de desenvolvimento do programa e dos seus comentários corresponde a **30%**, os testes no simulador **20%**, a apresentação dos testes no laboratório e seus comentários **30%**, e as conclusões **10%**.

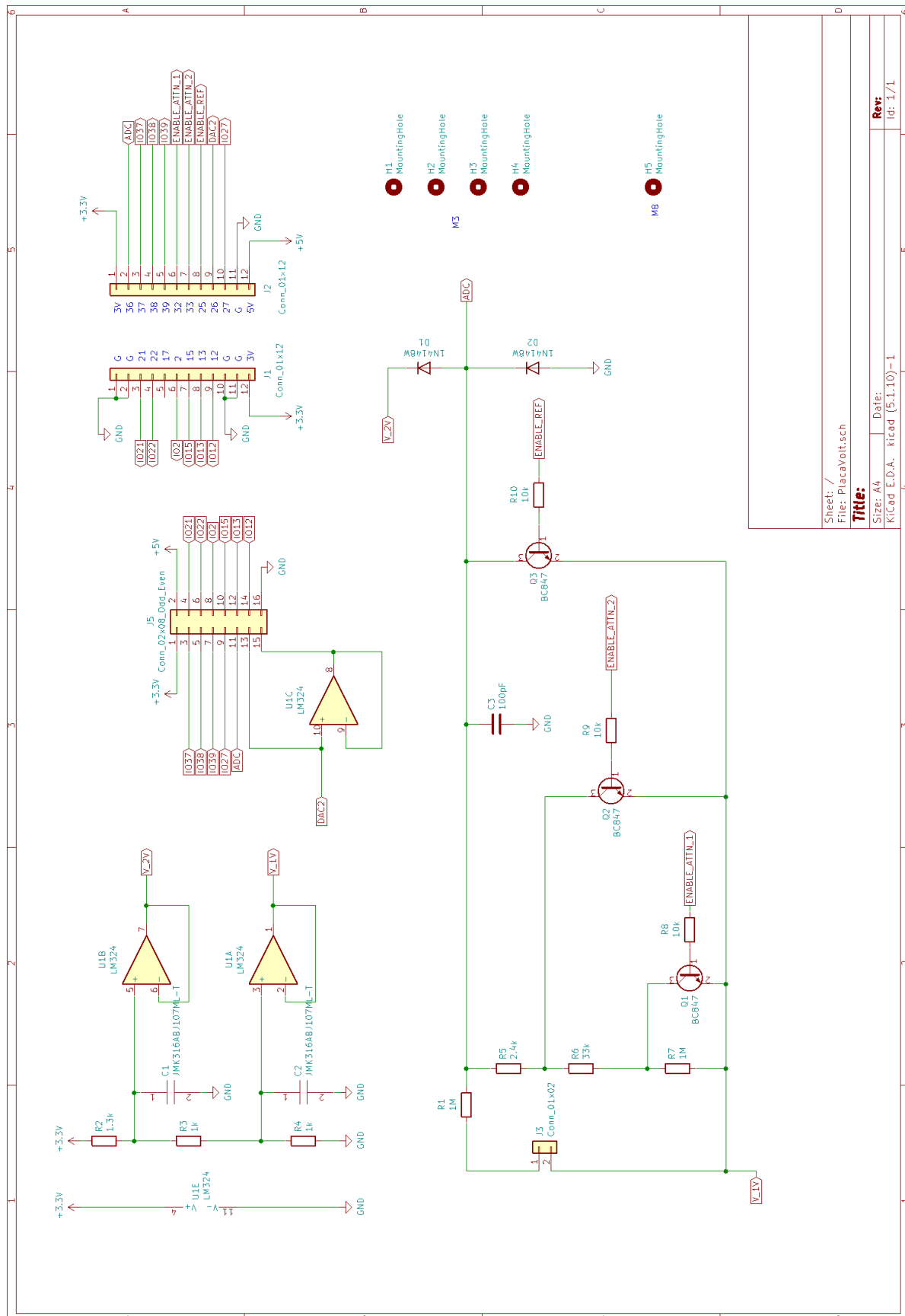
5 Referências

- [1] esp32, <https://www.espressif.com/en/products/socs/esp32>
- [2] python <https://www.python.org/downloads/>
- [3] notepad++ <https://notepad-plus-plus.org/downloads/>
- [4] pycharm <https://www.jetbrains.com/pycharm/>
- [5] spyder <https://www.spyder-ide.org/>
- [6] Visual Studio Code <https://code.visualstudio.com/>
- [7] Thonny <http://thonny.org/>

Anexo 1 – Esquema da placa microcontroladora



Anexo 2 – Esquema da placa de interface



Anexo 3 – Resumo linguagem Python – “Python Cheat Sheet”

Anexo 4 – Instalação do python e das bibliotecas

1. Procedimento de instalação do Python 3.9.9 num PC Windows

Recomenda-se a utilização da versão Python 3.9.9, a qual deve ser instalada a partir do site <https://www.python.org/downloads/release/python-399/>. No caso da instalação em ambiente Windows:

1. Comece por fazer o download do ficheiro de instalação (por. ex. python-3.9.9-amd64.exe);
2. Após execução do ficheiro, marcar a opção “Add Python 3.9 to PATH”, para que possa correr o Python a partir de uma janela de comandos (CMD) e de seguida clicar em “Install Now”;
3. Aceitar a autorização do sistema operativo para instalação do software;
4. Aguardar pela conclusão da instalação.

Para testar se o Python está instalado, pronto para correr e qual a versão instalada:

1. Abrir uma janela de comandos (CMD no caso do Windows: Tecla Windows ou Start e escrever cmd – “Command Prompt”;
2. Executar o comando “python --version”:

```
C:\>python --version
Python 3.9.9
C:\>
```

ou apenas “python”, entrando neste caso na consola de execução do programa, onde pode executar instruções de código, por exemplo para realizar pequenos testes de programação, onde no exemplo abaixo criaram-se duas variáveis (um inteiro e uma string) e imprimiu-se o seu valor na consola através da função print():

```
C:\>python
Python 3.9.9 (tags/v3.9.9:ccb0e6a, Nov 15 2021, 18:08:50) [MSC v.1929 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=12345
>>> b="Variável de teste"
>>> print("Impressão das variáveis a e b:",a,b)
Impressão das variáveis a e b: 12345 Variável de teste
>>> ^Z

C:\>
```

Para sair da consola de execução, fazer Ctrl. Z seguido de Enter.

2. Instalação das bibliotecas essenciais

De seguida deve instalar o módulo gráfico para ser utilizado pelo simulador, o PySide2 e o

requests, utilizando para tal o comando “pip install pyside2 requests”, que no caso do Windows instala-se da seguinte forma (neste caso foi instalada a versão mais recente 5.15.2):

```
C:\>pip install pyside2 requests
WARNING: Ignoring invalid distribution -yside2 (c:\python39\lib\site-packages)
WARNING: Ignoring invalid distribution -yside2 (c:\python39\lib\site-packages)
Collecting pyside2
  Downloading PySide2-5.15.2.1-5.15.2-cp35.cp36.cp37.cp38.cp39.cp310-none-win_amd64.whl
  (137.4 MB)
  ...
Collecting requests
  Downloading requests-2.27.1-py2.py3-none-any.whl (63 kB)
  ...
Installing collected packages: requests, pyside2
Successfully installed pyside2-5.15.2.1 requests-2.27.1
C:\>
```

De realçar que embora no ambiente de simulação (PC) possam ser instalados outros módulos Python de outras bibliotecas, não é possível utilizar essas bibliotecas no módulo IoT, pelo que **não poderão ser instaladas nem utilizadas outras bibliotecas além do PySide2 e do requests**. Por outro lado, também não poderão ser utilizadas outras funcionalidades da biblioteca PySide2 além daquelas que são apresentadas no subcapítulo seguinte.

Sendo o sistema operativo do módulo IoT o micropython, apenas poderão ser utilizadas funções e bibliotecas que fazem parte do micropython associado ao processador ESP32.

Atendendo a que o módulo IoT tem uma memória RAM limitada (menos de 100kB disponíveis) **recomenda-se uma utilização moderada de variáveis de memória, o uso da função gc.collect(), importando previamente este módulo (import gc) e apagar variáveis quando já não são utilizadas** (Ex: del x,y,z, para apagar as três variáveis).

3. Descrição do módulo de interface com o display.

O módulo de interface com o display baseia-se num objeto com a designação TFT, acedido através dos seguintes métodos:

working()	Método que devolve True enquanto o programa está a correr e devolve False quando o utilizador acionar o menu exit ou clicar para fechar a janela do programa. Assim o utilizador deverá ter o programa inserido num loop principal, testando esta variável, saindo deste loop quando a função devolver False, como por exemplo: <pre>while tft.working(): # código enquanto a janela está ativa (loop) # código após saída do loop caso seja necessário # antes do fim do programa</pre>
------------------	---

readButton()

Lê estado do botão e forma é premido, devolvendo um inteiro cujo significado é (o objeto TFT possui variáveis com nomes que facilitam o entendimento do valor devolvido pela função, conforme coluna do meio):

0	NOTHING	Nenhum botão premido
11	BUTTON1_SHORT	Botão 1 click rápido (<1segundo)
12	BUTTON1_LONG	Botão 1 click lento (>1segundo)
13	BUTTON1_DCLICK	Botão 1 duplo click
21	BUTTON2_SHORT	Botão 2 click rápido (<1segundo)
22	BUTTON2_LONG	Botão 2 click lento (>1segundo)
23	BUTTON2_DCLICK	Botão 2 duplo click

Por exemplo os dois primeiros valores são obtidos através de `tft.NOTHING` e `tft.BUTTON1_SHORT`

display_set(...)

Carrega no display um retângulo de uma determinada cor, sendo necessários 5 argumentos:

Cor – Cores pré-definidas (BLACK, BLUE, RED, GREEN, CYAN, MAGENTA YELLOW, WHITE, GREY1, GREY2)

X – Coordenada x do ponto inferior esquerdo

Y – Coordenada y do ponto inferior esquerdo

W – Largura do retângulo

H – Altura do retângulo

Caso se pretenda alterar apenas a cor de um pixel deverá ser `W=H=1` e caso se pretenda colocar todo o display a branco a instrução deverá ser:

```
tft.display_set(tft.WHITE,0,0,240,135)
```

display_pixel(...)

Carrega no display um pixel de uma determinada cor, sendo necessários 3 argumentos:

Cor – Cores pré-definidas (BLACK, BLUE, RED, GREEN, CYAN, MAGENTA YELLOW, WHITE, GREY1, GREY2)

X – Coordenada x do ponto do pixel

Y – Coordenada y do ponto do pixel

Para colocar o pixel do canto superior direito a azul deverá dar a instrução:

```
tft.display_pixel(tft.BLUE,239,134)
```

display_npixel(...) Carrega no display uma lista de pixéis de uma determinada cor, sendo necessários 3 argumentos:

COR – Cores pré-definidas (BLACK, BLUE, RED, GREEN, CYAN, MAGENTA, YELLOW, WHITE, GREY1, GREY2)

X – Lista com as coordenadas x dos pixéis

Y – Lista com as coordenadas y dos pixéis

Para desenhar uma linha diagonal amarela:

```
x=[]  
y=[]  
for n in range(240):  
    x.append(n)  
    y.append(round(n*134/239))  
tft.display_npixel(tft.YELLOW,x,y)
```

ou alternativamente de uma forma mais compacta

```
x=[n for n in range(0,240,1)]  
y=[round(n*134/239) for n in range(0,240,1)]  
tft.display_npixel(tft.YELLOW,x,y)
```

display_write_str(...) Escreve uma string no display utilizando a fonte Arial 16:

STR – String (conjunto de caracteres a ser representados no display)

X – Coordenada x do ponto do pixel (canto inferior esquerdo do primeiro caracter)

Y – Coordenada y do ponto do pixel (canto inferior esquerdo do primeiro caracter)

COR_F – Cor do caracter (opcional)

COR_B – Cor do fundo (opcional)

Para escrever o seguinte texto no ponto de coordenadas (x=20,y=50):

```
tft.write_str("Exemplo de texto",20,50)
```

display_write_grid(...) Desenha uma grelha retangular com um dado número de divisões na horizontal e na vertical, sendo necessários 9 argumentos:

X – Coordenada x do canto inferior esquerdo da grelha

Y – Coordenada y do canto inferior esquerdo da grelha

WIDTH – Largura da grelha

HEIGHT – Altura da grelha

NX – Número de divisões na horizontal (tem de ser par)

NY – Número de divisões na vertical (tem de ser par)

CENTER_LINES – Desenha linhas centrais (True/False)

COR1 – Cor do interior da grelha

COR2 – Cor das linhas centrais e exteriores da grelha

Para desenhar uma grelha com 10 divisões na horizontal e 6 divisões na vertical em dois tons de cinzento (GREY1 – escuro e GREY2 – claro), utilizando a largura total do display e deixando uma linha superior de 16 pixéis para escrever caracteres com e sem linhas centrais:

```
tft.display_write_grid(0,0,240,135-16,10,6,True,
tft.GREY1,tft.GREY2)
```



```
tft.display_write_grid(0,0,240,135-16,10,6,False,
tft.GREY1,tft.GREY2)
```



get_color (...)

Retorna a cor correspondente aos valores dos três argumentos:

R – Vermelho (entre 0 e 255)

G – Verde (entre 0 e 255)

B – Azul (entre 0 e 255)

Para desenhar um pixel em $x=20$ e $y=25$, com uma cor $R=125, G=0$ e $B=80$:

```
tft.display_pixel(tft.get_color(125,0,80),20,25)
```

read_adc(...)

Lê um dado número de pontos do ADC, durante um período especificado e retorna uma lista com os valores lidos (sendo o ADC de 12 bits, cada leitura situa-se entre 0 e 4095). São necessários 2 argumentos:

NPOINTS – Número de pontos a ler ($160 \leq NPOINTS \leq 240$)

INTERVAL – Intervalo de tempo total em milissegundos, em que é feita a leitura (apenas são aceites 50, 100, 200 e 500)

Para ler 240 pontos com um intervalo total de 50ms ($208.33\mu s$ por ponto), para uma variável amostras e imprimir na consola essas leituras:

```
Pontos=tft.read_adc(240,50)

for n in range(240):

    print(n,Pontos[n])
```

Os valores do print serão, por exemplo:

```
0 2064
1 2089
2 2114
3 2139
4 2164
...
239 2089
```

send_mail(...)

Envia por mail um ficheiro formato CSV (*Comma-separated values*) em anexo, com os pontos de uma lista para o endereço especificado:

DELTA_T – Float com o intervalo de tempo entre cada ponto em segundos,

PONTOS_V – Lista de pontos (pode ser por exemplo a lista devolvida pela função `read_adc()`, convertida em tensões em Volt), que podem ser inteiros ou float,

CORPO_MENSAGEM – String contendo informação a colocar no corpo da mensagem,

ENDEREÇO – Endereço para onde é enviada a mensagem e para o

caso de vários endereços, estes devem ser separados por vírgula (ex:
"add1.google.com,add2.google.com"),

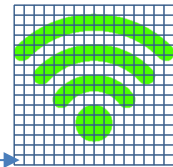
Por exemplo caso se pretenda enviar um mail, com cada um dos pontos vindos do ADC multiplicados por 0.00072, em que os 240 pontos correspondem a um intervalo total de 0.5 segundos:



```
delta pontos_adc=tft.read_adc(240,50)
pontos_volt = [0.0]*240    # Lista com 240 floats
for n in range(240):
    pontos_volt[n] = 0.00072 * pontos_adc[n]
corpo = "Texto adicional:\n240 pontos."
address = "exemplo@gmail.com"
tft.send_mail(0.5/240,pontos_volt,corpo,address)
```

set_wifi_icon(...)

Coloca o icon com o estado da ligação Wi-Fi na posição dada pelos dois argumentos:

X, Y – Coordenadas do canto inferior esquerdo



O icon tem 16x16pixels, sendo verde caso a ligação Wi-Fi esteja estabelecida , ou vermelho caso não tenha sido conseguida .

Por exemplo, caso se pretenda colocar o icon no canto superior direito do display a instrução deverá ser (sendo $224 = 240 - 16$ e $119 = 135 - 16$):

```
tft.set_wifi_icon(224,119)
```