3º Trabalho de Laboratório:

µOscilloscope

Osciloscópio através de um sistema embebido

Curso: Licenciatura em Engenharia Electrotécnica e de Computadores

UC: Sistemas Electrónicos

2º Semestre 2021/2022 (P3)

Docente: José Júlio Paisana

Autores: Grupo nº 2

- (66325) Tomás Marques Videira Fonseca, tomas.mvf@gmail.com

- (96135) Afonso Brito Caiado Correia Alemão, afonso.alemao@tecnico.ulisboa.pt

- (96317) Rui Pedro Canário Daniel, ruipcdaniel@tecnico.ulisboa.pt

Data: 28/04/2022

Índice

1.Introdução

Um sistema embebido é uma combinação de um hardware computacional e um software projetado para uma função específica. No que diz respeito a 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, convertendo-as em dados lidos pelo processador que os converte em informação útil.

A tecnologia da Internet das Coisas, 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. Desta forma, IoT representa um tipo de sistema embebido ligado à internet.

Neste relatório, iremos desenvolver um projeto, através da criação duma aplicação de software para implementar um pequeno osciloscópio, utilizando um hardware para desenvolvimento de soluções IoT.

Os objetivos que pretendemos atingir com a realização deste trabalho são o estudo e aprendizagem do circuito de hardware base para a realização do trabalho, o desenvolvimento do software para implementação do osciloscópio, o teste e ajuste deste software, recorrendo à sua calibração.

Circuito hardware base

No laboratório foi-nos disponibilizado um módulo IoT, representado na figura 1.

Figura 1: Módulo IoT.

Este módulo IoT é composto, entre outros, por uma placa microcontroladora, baseada num processador da família ESP32, possuindo recursos como *Wi-Fi* integrado e conectividade *Bluetooth* para uma ampla gama de aplicações, *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.

A placa de interface possui uma entrada com dois *bornes* por onde se liga o sinal do canal do osciloscópio. Para além disso, possui ainda um conjunto de circuitos eletrónicos para interface com o ADC, como fichas de interligação com a placa microcontroladora; um circuito integrado LM324 com dois AMPOPs para gerar tensões DC de referência de 1 V e de 2 V; dois transístores funcionando como interruptores para implementação de divisores resistivos, permitindo criar várias escalas para o sinal de entrada (Q1 e Q2); um transístor funcionando como interruptor (Q3), para a leitura do sinal de referência (1 V); e dois díodos de proteção ligados ao pino de entrada do ADC para o proteger, onde, caso se considere que os díodos têm V_L = 0,7 V, a tensão do ADC fica limitada a $-0{,}7\text{ V} < V_{ADC} < 2{,}7\text{ V}$.

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 2 V, o terminal negativo do sinal de entrada liga-se a uma fonte de tensão de 1 V. Assim é garantido que a conversão se realiza para tensões de entrada entre -1 V (V_{ADC} = 0 V) e 1 V (V_{ADC} = 2 V).

De modo a permitir gamas de tensão de medida superiores ao intervalo (−1 V, +1 V), o terminal positivo da entrada não é ligado diretamente ao ADC, mas sim a um conjunto de divisores de tensão.

Consoante as zonas de funcionamento de Q1 e de Q2, os divisores resistivos serão constituídos por:

$$R_1 \text{ e } R_5 + R_6 + R_7 \text{ caso } Q_1 \text{ e } Q_2 \text{ ao corte} \Rightarrow Fator_1 = \frac{R_5 + R_6 + R_7}{R_1 + R_5 + R_6 + R_7} = 0{,}509 = \frac{1}{1{,}97} \quad (1)$$

$$R_1 \text{ e } R_5 + R_6 \text{ caso } Q_1 \text{ a conduzir e } Q_2 \text{ ao corte} \Rightarrow Fator_2 = \frac{R_5 + R_6}{R_1 + R_5 + R_6} = 0{,}0342 = \frac{1}{29{,}3} \quad (2)$$

$$R_1 \text{ e } R_5 \text{ caso } Q_1 \text{ ao corte e } Q_2 \text{ a conduzir} \Rightarrow Fator_3 = \frac{R_5}{R_1 + R_5} = 0{,}00239 = \frac{1}{418} \quad (3)$$

Para o nosso caso, pretendemos que o osciloscópio funcione na segunda escala, com Q_1 a conduzir e Q_2 ao corte. Desta forma são permitidas leituras de tensões entre -30 V e 30 V. Desta forma, obtemos a expressão (4).

$$V_{ADC} = Fator_2 \cdot V_i + V_{1V} = Fator_2 \cdot V_i + V_{1V} = Fator_2 \cdot V_i + 1 \quad (4)$$

Visto que $0\text{ V} < V_{ADC} < 2\text{ V}$, tem-se que $-29{,}3\text{ V} < V_i < 29{,}3\text{ V}$.

Calibração

Embora seja necessária uma calibração caso a caso para se obter maior precisão, como primeira aproximação utilizamos a fórmula para obtenção do valor digital lido pelo ADC (D) em função da tensão de entrada no ADC (V_{ADC}) indicada no enunciado de laboratório: "aproximação 1" - dada pela expressão (5).

$$D = 2271{,}27\, V_{ADC} - 207{,}72 \text{ para } 0{,}091455\text{ V} < V_{ADC} < 1{,}8944\text{ V} \quad (5)$$

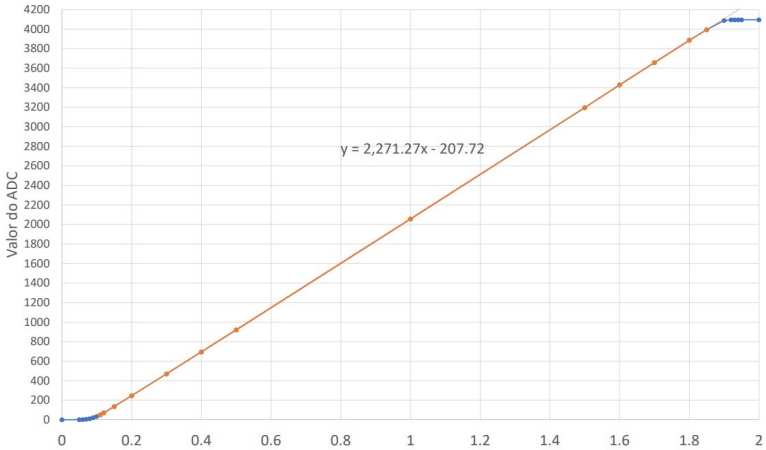

Figura 2: Medidas da resposta do ADC para um módulo IoT para a "aproximação 1".

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

$$V_{ADC} = 0{,}00044028D + 0{,}091455 \text{ para } 0 \leq D \leq 4095 \quad (6)$$

Mas, atendendo aos erros inerentes a cada um dos módulos IoT, particularmente da parte de conversão do ADC, foi aproveitado o laboratório para se realizar uma calibração do circuito, de onde obtivemos a "aproximação 2".

Através da colocação na entrada do circuito de alguns valores DC pré-definidos, para cada tensão DC de entrada (medida com um multímetro) fizemos uma leitura do valor do conversor ADC com precisão, recorrendo à média de uma leitura de 100 amostras de 100 pontos por amostra. Para tal utilizamos o programa *main_exemplo_2.py*, fornecido pelos docentes.

Sendo que o nosso grupo ficou encarregue do módulo *IoT Module 01*, foram realizadas medidas num módulo IoT representadas na tabela 1, que conduziram aos resultados no gráfico da figura 3. Para a obtenção de V_{ADC} foi utilizada a expressão (4).

Tensão DC de entrada [V]	VADC [V]	Valor do ADC
-10,003	0,65860068	1248,23
-4,995	0,82952218	1625,57
-2,006	0,93153584	1854,11
0,001	1,00003413	2006,05
1,993	1,06802048	2161,97
5,007	1,17088737	2389,5
9,998	1,34122867	2776,31

Tabela 1: Valores de calibração medidos no laboratório.

Sendo y a variável que representa "Valor do ADC" e x a variável que representa o "V_{ADC} [V]", obtemos através de uma regressão linear:

$$y = 2238{,}9237151x - 230{,}0491338 \text{ para } 0{,}65860068 < x < 1{,}34122867$$
$$R^2 \simeq 0{,}9999711 \quad (7)$$

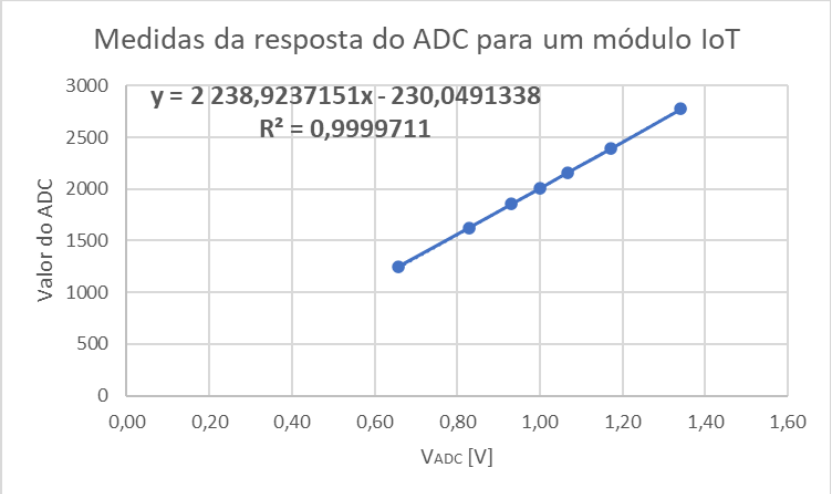

Figura 3: Medidas da resposta do ADC para um módulo IoT para a "aproximação 2".

Podemos então verificar que estamos numa região em que as variáveis y e x se relacionam de forma linear, visto que a partir da regressão linear $y = mx + b$ calculada se obteve $R^2 \approx 1$ e $m \neq 0$, ou seja, cerca de 99,997 % da variação total da variável y é explicada pela variável x através do modelo de regressão linear simples ajustado, donde podemos afirmar que a reta estimada parece ajustar-se bem ao conjunto de dados.

Pela fórmula inversa, obtemos:

$$x = 0{,}0004466y + 0{,}1027499 \text{ para } 1248{,}23 < y < 2776{,}31 \quad (8)$$

A partir dos valores medidos, foram recalculados os parâmetros do gráfico da figura 2 do enunciado, sendo estes introduzidos no nosso programa.

2. Desenvolvimento do programa

De seguida, apresentamos o código do ficheiro *main.py* na sua totalidade, na figura c1. O código apresenta-se com comentários detalhados sobre o seu funcionamento.

main.py

```
# Micro-Osciloscope IOT
# IoT Module 1
# Autores:
# - Afonso Alemão
# - Rui Daniel
# - Tomás Fonseca
# Data: 26/04/2022

# módulo de interface com o display
import T_Display
# módulo de funções matemáticas
import math
import time

# Inicializações de variáveis globais
# Lista com 240 floats, de modo a possuir 240 pontos a imprimir no ecrã
# visto que o display TFT tem uma resolução de 240 pixéis na horizontal
pontos_volt = [0.0] * 240
# Variável apenas usada para display da calibração
width = 160
# Ordenada máxima. Usada para centrar os pontos na grelha.
height = 134
# Grelha possui 6 divisões verticais e 10 divisões horizontais
N_div_verticais = 6
N_div_horizontais = 10

# Função de leitura dos valores do ADC e conversão para tensão de entrada real
# Argumentos de entrada:
# - pontos: número de pontos a ler no ADC
# - escala_horizontal: valor temporal de uma divisão horizontal
# Argumentos de saída:
# - pontos_adc: pontos lidos pelo ADC
# - pontos_volt: valores de tensão de entrada
def read_and_convert(pontos, escala_horizontal):
    if escala_horizontal == 5:
        # Lê pontos do ADC em 50 ms
        pontos_adc = tft.read_adc(pontos, 50)
    elif escala_horizontal == 10:
        # Lê pontos do ADC em 100 ms
        pontos_adc = tft.read_adc(pontos, 100)
    elif escala_horizontal == 20:
        # Lê pontos do ADC em 200 ms
        pontos_adc = tft.read_adc(pontos, 200)
    else:
        # Lê pontos do ADC em 500 ms
        pontos_adc = tft.read_adc(pontos, 500)

    # Inicialização de variáveis a calcular - irrelevante neste trabalho
    # Vmax = 0
    # Vmin = 0
    # Vmed = 0

    for n in range(pontos):
        # Para cada ponto lido pelo ADC, convertemos o valor digital lido
        # pelo ADC (pontos_adc[n]) em Volt (V_ADC)

        # Usado para testes no laboratório: Depois da calibração
        V =  0.0004466 * pontos_adc[n] + 0.1027499
        # Usado para testes no simulador: Antes da calibração
        # V =  0.00044028 * pontos_adc[n] + 0.091455

        # Conversão de V_ADC em Vi
        # Tensão entrada de referência de 1 V
        V = V - 1
        # Entra com o efeito do div. resistivo
        V = V / fator
```

```
        # pontos_volt guarda a tensão de entrada real (Vi)
        pontos_volt[n] = V

        # Cálculos auxiliares para obtenção dos valores máximo, mínimo e
        # médio de tensão: não relevante para este trabalho
        # Caso seja o primeiro ponto, inicializamos as variáveis com o
        # seu valor de tensão
        # if n == 0:
        #     Vmax = Vmin = Vmed = V
        # else:
        #     Vmed += V
        #     if V > Vmax: Vmax = V
        #     if V < Vmin: Vmin = V
    # Para obtenção do valor médio, o somatório das tensões de entrada é
    # dividido pelo número de amostras - irrelevante neste trabalho
    # Vmed /= pontos

    return [pontos_adc, pontos_volt]

# Função para realizar amostras e médias - usada para calibração
# Argumentos de entrada:
# - num_amostras: número de amostras de 100 pontos a serem
#                 utilizadas para o cálculo da média
# def media_amostras(num_amostras):

    # Apaga parte direita do display exceto ícone WiFi
    # tft.display_set(tft.BLACK, width, 0, 240 - width, height - 16)
    # soma = 0
    # for n in range(num_amostras):
        # Lê 100 pontos do ADC em 50 ms
        # pontos_adc = tft.read_adc(100, 50)
        # for j in range(100):
            # soma += pontos_adc[j]

    # Para obtenção do valor médio, o somatório dos pontos lidos do ADC é
    # dividido pelo número total de pontos
    # media = soma / (100 * num_amostras)

    # Escreve valores no display - não usado neste trabalho
    # 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)

# Fator do divisor resistivo constituído por R1 e por R5 + R6
# caso Q1 a conduzir e Q2 ao corte
fator = 1 / 29.3

# Instância um objeto da classe TFT
tft = T_Display.TFT()

# Por defeito, no arranque do programa são utilizados os seguintes valores:
# Escala vertical = 2 V / div; Escala horizontal = 5 ms / div
escala_vertical = 2
escala_horizontal = 5

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

    # Insere no display uma grelha tendo em vista 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 do ecrã.
    tft.display_write_grid(0, 0, 240, 135 - 16, N_div_horizontais, N_div_verticais, tft.GREY1, tft.GREY2)

    # Na horizontal:
    # 240 pixéis corresponde a N_div_horizontais * escala_horizontal (escala_horizontal
    #   por divisão) - Cada pixel: tempo_pixel
    #   tempo_pixel = N_div_horizontais * escala_horizontal / 240

    # Na vertical:
    #   134 pixéis corresponde a 6 * escala_vertical (escala_vertical por divisão):
    #   Cada volt: (134/(6 * escala_vertical)) pixéis

    x = []
    y = []
```

```
# Lê os valores do ADC e obtém tensão de entrada real para cada ponto
[pontos_adc, pontos_volt] = read_and_convert(240, escala_horizontal)

# Tendo em vista o display da tensão na grelha, são criados os
# pontos (x, y) a serem imprimidos no gráfico.
for n in range(len(pontos_volt)):
    # t = n * tempo_pixel
    volt = pontos_volt[n]

    # Para o cálculo da coordenada y de cada ponto, consideramos que o valor y = 0 está a
    # meio da grelha, ou seja, em (height - 16) / 2. A partir desta referência, são calculadas
    # a coordenada y dos restantes pontos de forma a apresentar a tensão pretendida de acordo
    # com a escala vertical
    pixel = (height - 16) / 2 + ((height - 16) / (6 * escala_vertical)) * volt

    # Coordenada x de cada ponto
    x.append(n)

    # Assegurar que não são imprimidos pontos fora da grelha (na vertical):
    # Caso estes excedam o limite superior (ou inferior) da grelha, serão imprimidos
    # neste limite. Utilizámos este método, pois é desta forma que funciona o osciloscópio
    # utilizado nos trabalhos de laboratórios anteriores.
    if pixel > 135 - 16:
        pixel = 135 - 17
    if pixel < 0:
        pixel = 0
    y.append(round(pixel))

# Imprime a tensão de entrada no ecrã
tft.display_nline(tft.YELLOW, x, y)

# É imprimido num espaço com uma altura de 16 pixéis situado na parte de cima
# do display informação sobre as escalas atuais (vertical e horizontal) e sobre
# o estado da ligação Wi-Fi.
str1 = "y : " + str(escala_vertical) + " V / div"
str2 = "x : " + str(escala_horizontal) + " ms / div"
tft.display_write_str(tft.Arial16, str1, 5, 135 - 15)
tft.display_write_str(tft.Arial16, str2, 115, 135 - 15)
tft.set_wifi_icon(240 - 16, 135 - 16)

# Ciclo principal do programa
# Working é um 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.
while tft.working():
    # Lê estado dos botões
    but = tft.readButton()
    if but != tft.NOTHING:
        # Houve um botão que foi pressionado
        print("Button pressed:", but)

        # Botão 1 click rápido - Nova leitura e representação da forma de onda;
        if but == 11:
            break

        # Botão 1 click lento - Envia por mail um ficheiro .csv com os pontos adc
        # outro ficheiro .csv com os pontos volt obtidos
        if but == 12:
            tft.send_mail(pontos_adc,
            "tomas.mvf@gmail.com,afonso.alemao@tecnico.ulisboa.pt,ruipcdaniel@tecnico.ulisboa.pt")
            tft.send_mail(pontos_volt,
            "tomas.mvf@gmail.com,afonso.alemao@tecnico.ulisboa.pt,ruipcdaniel@tecnico.ulisboa.pt")

        # Botão 2 click rápido - Altera a escala vertical, passando para a
        # escala imediatamente acima e de forma circular. De seguida, nova leitura
        # e representação da forma de onda;
        if but == 21:
            if escala_vertical == 1:
                escala_vertical = 2
            elif escala_vertical == 2:
                escala_vertical = 5
            elif escala_vertical == 5:
                escala_vertical = 10
            elif escala_vertical == 10:
                escala_vertical = 1
            break

        # Botão 2 click lento (22) - Altera a escala horizontal, passando para a
        # escala imediatamente acima e de forma circular. De seguida, nova leitura
```

```
        # e representação da forma de onda;
        if but == 22:
            if escala_horizontal == 5:
                escala_horizontal = 10
            elif escala horizontal == 10:
                escala_horizontal = 20
            elif escala_horizontal == 20:
                escala_horizontal = 50
            elif escala_horizontal == 50:
                escala_horizontal = 5
            break
```

Como funcionalidade adicional, de forma a assegurar que não são imprimidos pontos fora da grelha (na vertical), realizámos a seguinte funcionalidade: sempre que os pontos excedam o limite superior (ou inferior) da grelha, serão imprimidos neste limite. Utilizámos este método, pois é desta forma que funciona o osciloscópio utilizado nos trabalhos de laboratórios anteriores.

3.Testes no simulador

Visto que a função *read_adc* no ficheiro *T_Simulator.py* está definida com base na "aproximação 1", foi esta a utilizada nesta secção de testes no simulador.

Deste modo, procedeu-se à testagem e apresentação gráfica do programa, na qual se analisou as variações da forma de onda, da escala horizontal, da escala vertical, do offset, da amplitude e da frequência de uma tensão simulada.

De seguida, encontram-se representados, na figura 4, três sinais com diferentes formas de onda: sinusoidal, triangular e quadrada. Em cada um dos casos a tensão de offset, a amplitude e a frequência do sinal é a mesma, respetivamente, 0 V, 5 V e 25 Hz. A escala horizontal utilizada, 5 ms/div e a vertical, 2 V/div mantêm-se fixas nas 3 simulações.

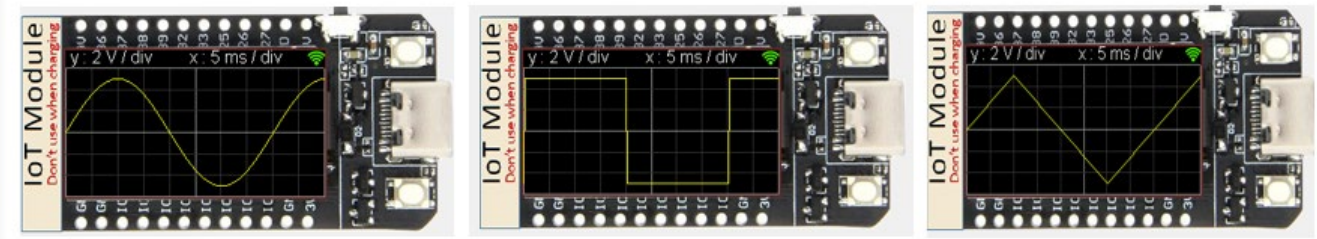

Figura 4: Representação gráfica de um sinal sinusoidal, quadrado e triangular (esquerda para a direita).

Posteriormente, para um sinal sinusoidal com uma amplitude AC de 5 V, frequência de 25 Hz, e mantendo uma escala vertical de 2 V/div, variou-se a escala horizontal, utilizando em cada uma das simulações uma escala de {5; 10; 20; 50} ms/div. Apresentamos os resultados obtidos na figura 5.

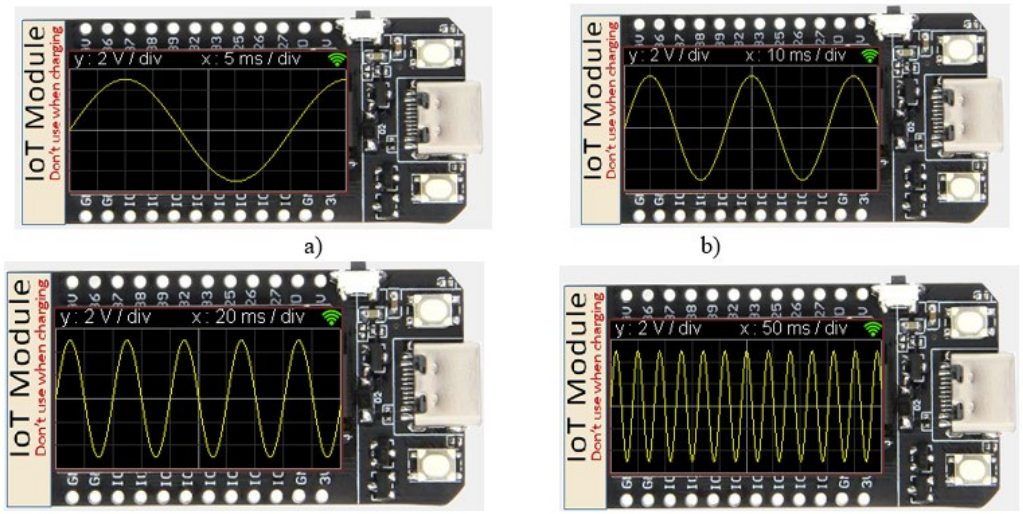

Figura 5: Representação gráfica do sinal sinusoidal, variando a escala horizontal.

Para um sinal sinusoidal com uma amplitude AC de 5 V, 0 V de offset, 25 Hz de frequência, e mantendo uma escala horizontal de 5 ms/div, variou-se a escala vertical, utilizando em cada uma das simulações uma escala de {1; 2; 5; 10} V/div. Apresentamos os resultados obtidos na figura 6.

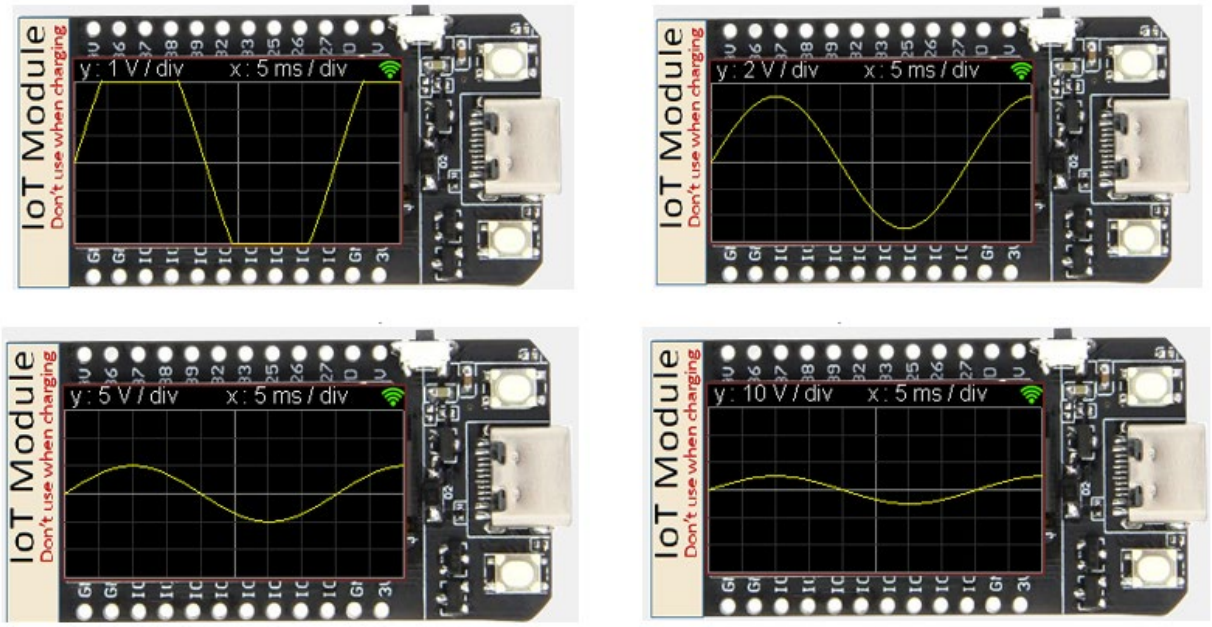

Figura 6: Representação gráfica do sinal sinusoidal, variando a escala vertical.

Para um sinal sinusoidal de frequência de 25 Hz, com tensão de offset de 0 V, e mantendo uma escala horizontal e vertical de, respetivamente, 5 ms/div e 2 V/div, procedeu-se à variação da amplitude AC do sinal, tendo-se testado os seguintes valores de amplitude: {2, 4; 6} V. Apresentamos os resultados obtidos na figura 7.

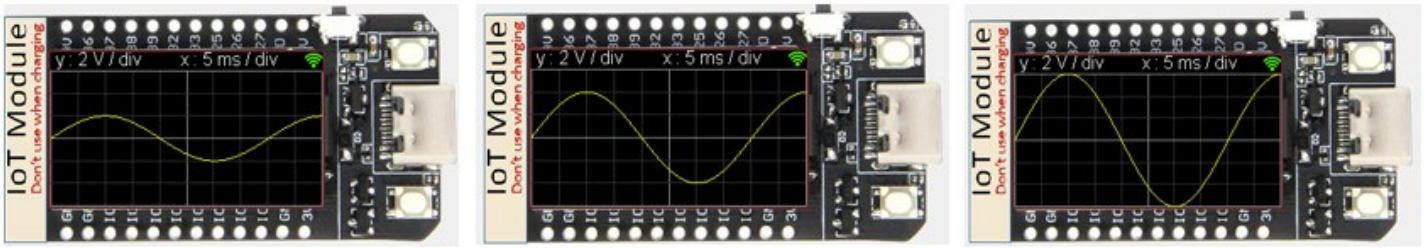

Figura 7: Representação gráfica do sinal sinusoidal, variando a amplitude AC de forma crescente.

Seguidamente, para um sinal sinusoidal com uma amplitude AC de 5 V, 25 Hz de frequência, e mantendo uma escala horizontal e vertical de, respetivamente, 5 ms/div e 2 V/div, variou-se a tensão de offset, tendo sido usado em cada uma das simulações um valor offset de {−5; −1; 0; 1; 5} V. Apresentamos os resultados obtidos nas figuras 8 e 9.

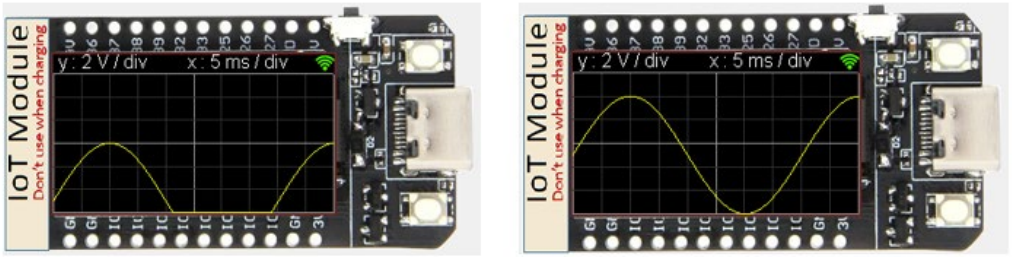

Figura 8: Representação gráfica do sinal sinusoidal, utilizando um offset negativo, −5 V (esquerda) e −1 V (direita).

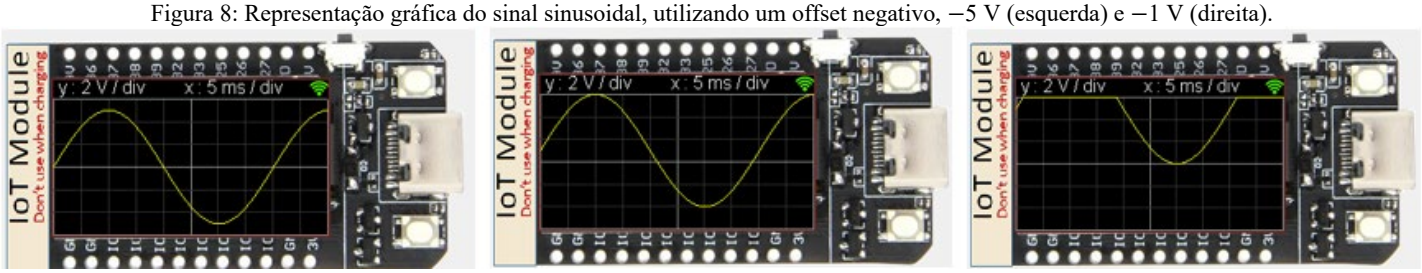

Figura 9: Representação gráfica do sinal sinusoidal, utilizando um valor de offset de 0 V (esquerda), de 1 V (centro) e de 5 V (direita).

Também se testou a imposição de uma tensão constante DC de $\{-10; -5; 0; 5; 10\}$ V, mantendo fixa a escala horizontal em 5 ms/div e a vertical em 5 V/div. Note-se que a amplitude AC pico-a-pico do sinal sinusoidal é nula, sendo que apenas existe componente contínua. Na figura 10, apresentamos os resultados obtidos.

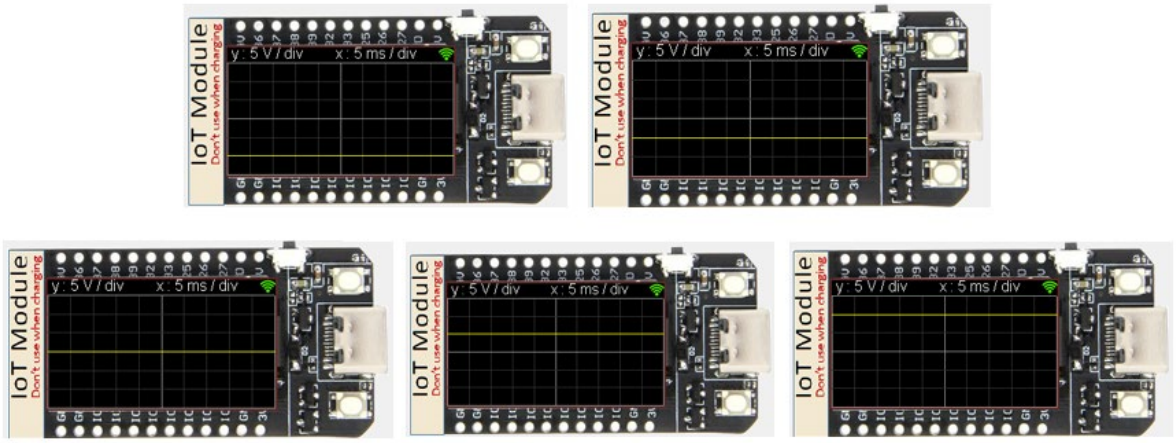

Figura 10: Representação gráfica de uma tensão DC igual a -10 V (em cima à esquerda), -5 V (em cima à direita), 0 V (em baixo à esquerda), 5 V (em baixo ao centro) e 10 V (em baixo à direita).

Por último, para um sinal sinusoidal com uma amplitude AC de 5 V, 0 V de offset, e mantendo fixa uma escala horizontal e vertical de, respetivamente, 5 ms/div e 2 V/div, procedeu-se à simulação do sinal com diferentes frequências. Decidiu-se assim representar 3 sinais com frequências de $\{25; 50; 100\}$ Hz. Apresentamos os resultados obtidos na figura 11.

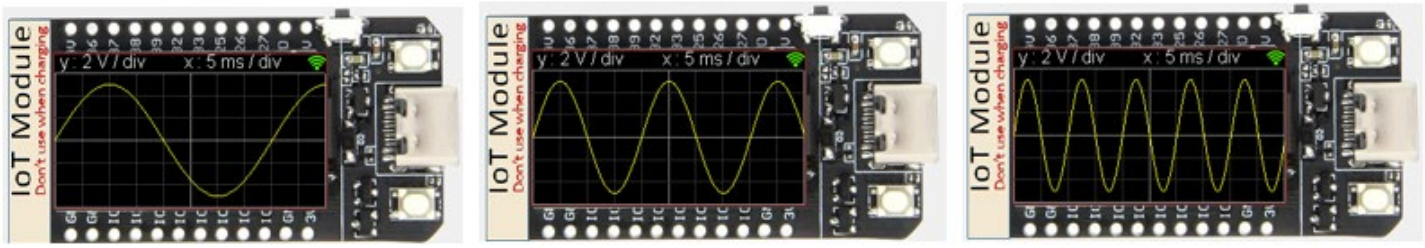

Figura 11: Representação gráfica dos sinais sinusoidais para diferentes valores de frequência: 25 Hz (esquerda), 50 Hz (centro) e 100 Hz (direita).

Ao longo dos testes de simulação obtivemos os resultados esperados teoricamente. Por exemplo, para o sinal representado à esquerda da figura 11, verificamos que o período temporal é igual a 8 divisões da escala horizontal, ou seja, $T = 40$ ms, o que corresponde à frequência $f = T^{-1} = 25$ Hz do sinal aplicado. Na escala vertical, podemos observar que o sinal possui valor médio aproximadamente nulo, e amplitude igual a 2,5 divisões da escala vertical, ou seja, a 5 V, como era expectável.

4.Testes no laboratório

Na secção "1. Introdução" descrevemos detalhadamente o processo de calibração.

Começamos por realizar diversos testes com a "aproximação 1", ou seja, a aproximação antes da calibração.

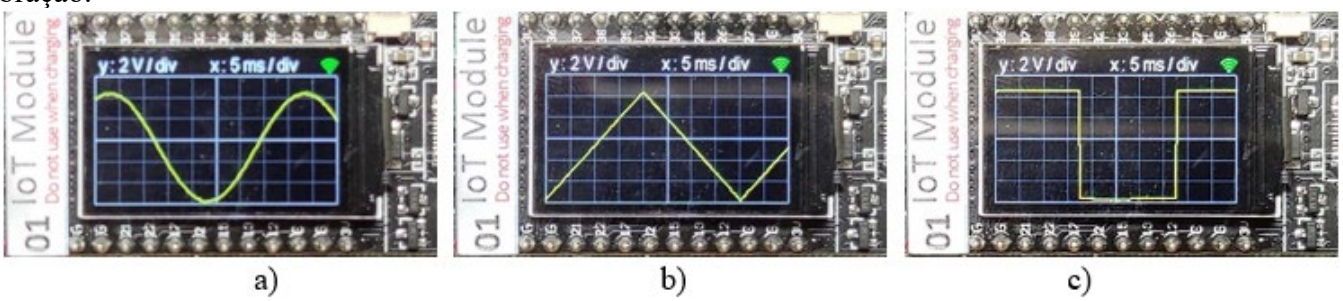

Figura 12: Representação gráfica, antes da calibração, de uma onda:
a) sinusoidal; b) triangular; c) quadrada.

Na figura 12 apresentamos os resultados de três formas de ondas distintas, com 25 Hz de frequência, 5 V de amplitude e 0 V de tensão de offset: sinusoidal, triangular e quadrada.

De seguida, para uma onda sinusoidal com 25 Hz de frequência, 5 V de amplitude e 0 V de tensão de offset, mantendo fixa a escala vertical a 2 V/div, variou-se a escala horizontal utilizando {5; 10; 20; 50} ms/div. Na figura 13 estão representados os resultados obtidos.

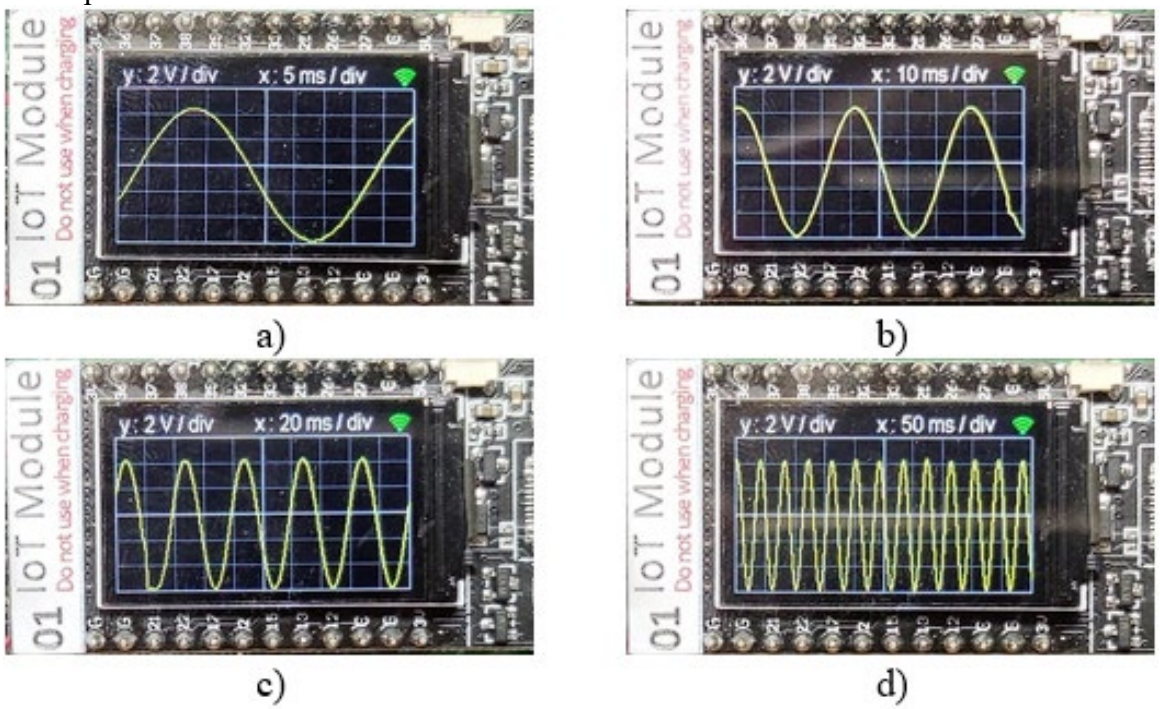

Figura 13: Representação gráfica, antes da calibração, do sinal sinusoidal, variando a escala horizontal:
a) 5 ms/div; b) 10 ms/div; c) 20 ms/div; d) 50 ms/div.

Posteriormente, para uma onda sinusoidal com 25 Hz de frequência, 5 V de amplitude e 0 V de tensão de offset, mantendo fixa a escala horizontal a 5 ms/div, variou-se a escala vertical utilizando {1; 2; 5; 10} V/div. Os resultados obtidos são representados na figura 14.

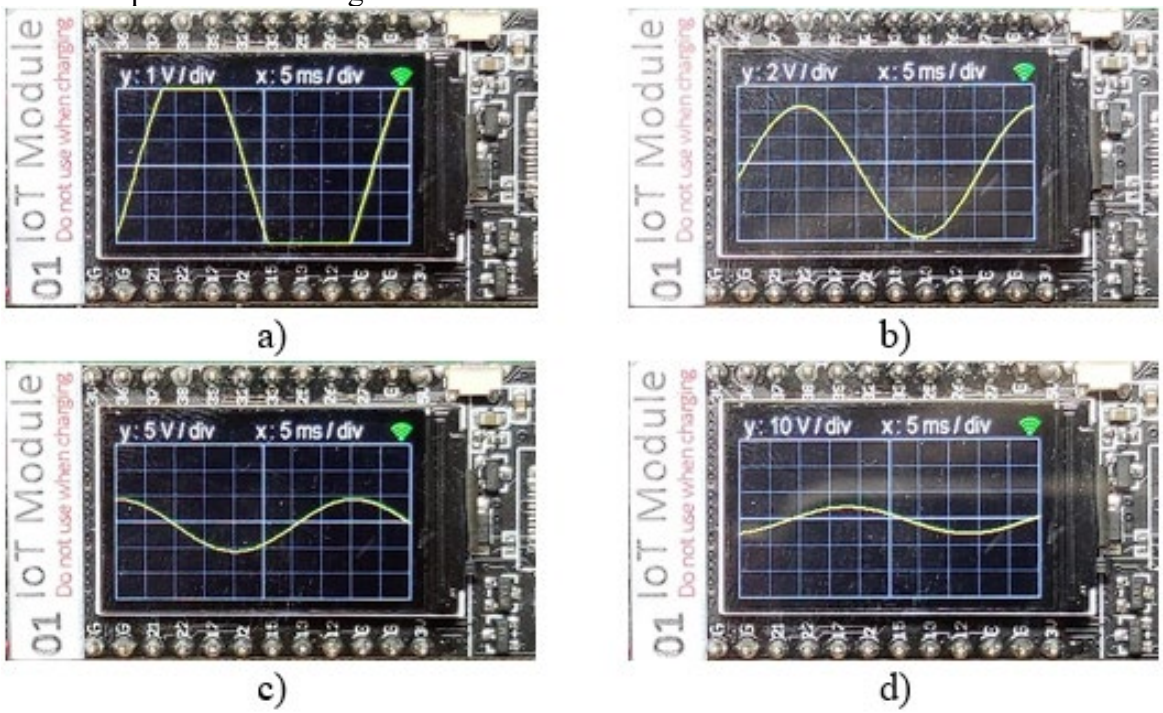

Figura 14: Representação gráfica, antes da calibração, do sinal sinusoidal, variando a escala vertical:
a) 1 V/div; b) 2 V/div; c) 5 V/div; d) 10 V/div.

Para finalizar os testes efetuados antes da calibração, aplicamos uma tensão constante de {−10; −5; 0; 5; 10} V, mantendo fixa a escala horizontal em 5 ms/div e a vertical em 5 V/div. Na figura 15 apresentamos os resultados obtidos.

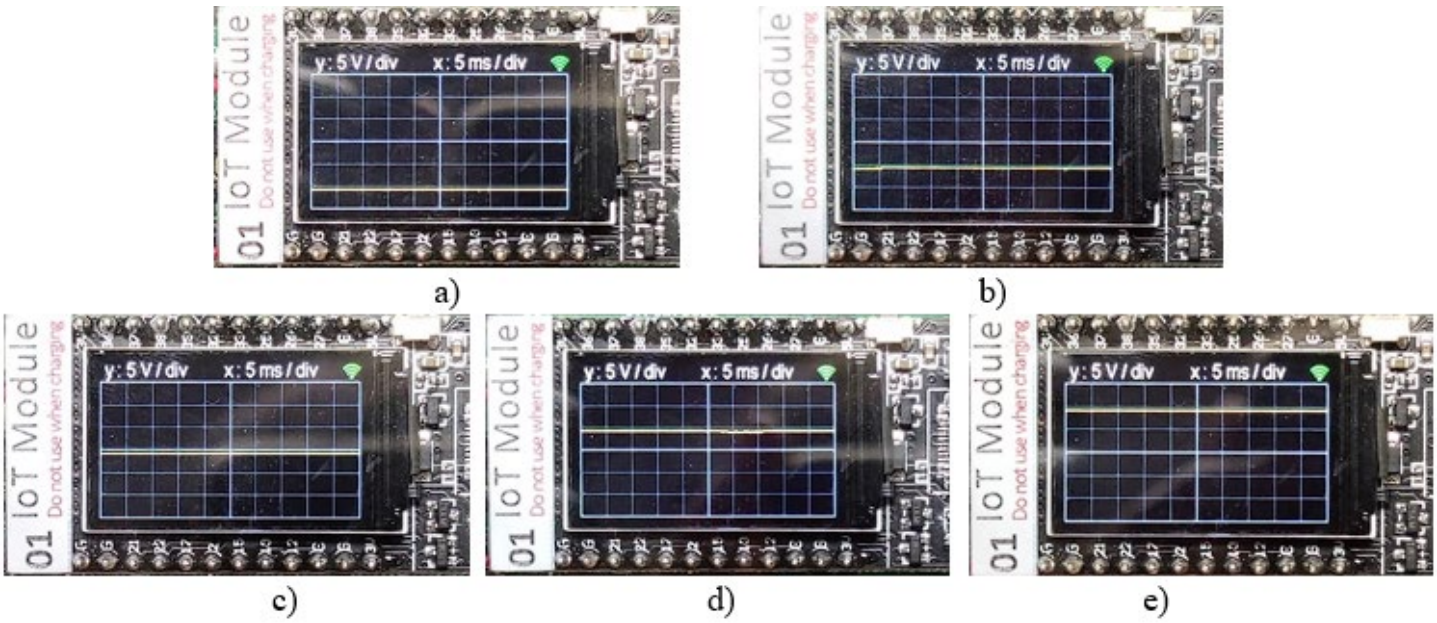

Figura 15: Representação gráfica, antes da calibração, de uma tensão constante:
a) −10 V; b) −5 V; c) 0 V; d) 5 V; e) 10 V.

Atendendo aos erros inerentes a cada um dos módulos IoT, particularmente da parte de conversão do ADC, foi aproveitado o laboratório para se realizar uma calibração do circuito, de onde obtivemos a "aproximação 2", especificada em (8), através de um processo descrito de forma pormenorizada na secção "1. Introdução".

Analisando os resultados obtidos antes da calibração, é notório que as ondas representadas graficamente, no módulo IoT, não estão corretamente centradas na grelha de pontos. Por exemplo, na figura 15 é notório que o valor DC representado se encontra ligeiramente afastado do valor esperado e na figura 12 é evidente que o valor médio dos sinais de entrada representado deveria ser observado como nulo, o que não acontece. Para além disso, os sinais representados estão ligeiramente não concordantes com as escalas selecionadas.

Torna-se assim essencial efetuar a calibração do dispositivo, com a finalidade de corretamente analisar as tensões medidas.

Com o objetivo de comparar a qualidade dos resultados obtidos, realizámos os mesmos testes efetuados anteriormente, mas para a aproximação obtida depois da calibração.

Como primeiro teste, aplicámos novamente uma onda com frequência de 25 Hz, amplitude AC de 5 V e tensão de offset de 0 V, para as formas de onda sinusoidal, quadrada e triangular, obtendo os resultados representados na figura 16.

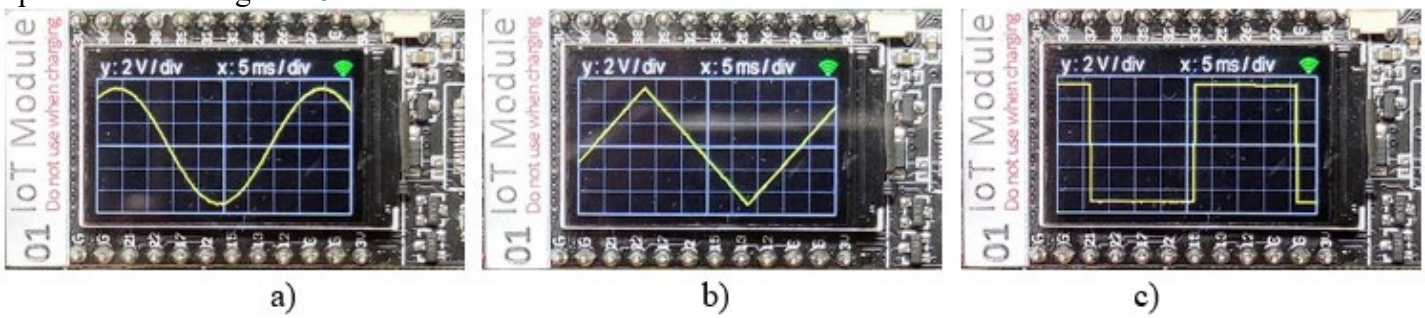

Figura 16: Representação gráfica, depois da calibração, de uma onda:
a) sinusoidal; b) triangular; c) quadrada.

De seguida, para uma onda sinusoidal com 25 Hz de frequência, 5 V de amplitude e 0 V de tensão de offset, mantendo fixa a escala vertical a 2 V/div, variou-se a escala horizontal utilizando {5; 10; 20; 50} ms/div, obtendo assim os resultados representados na figura 17.

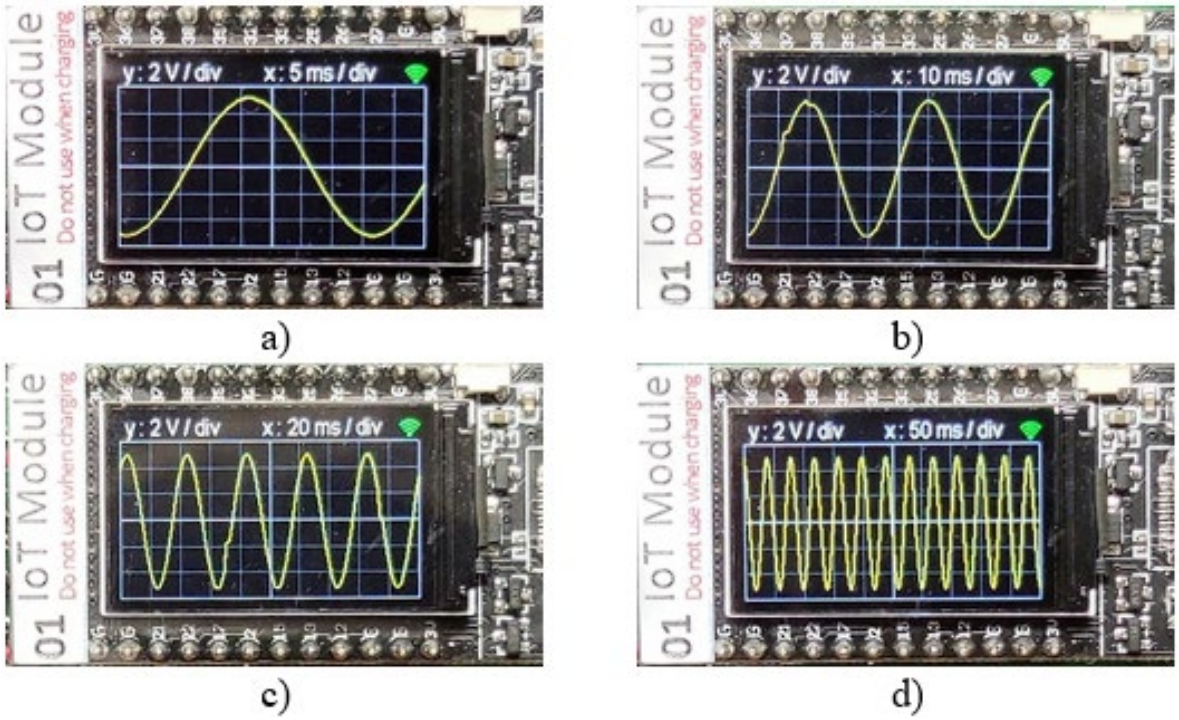

Figura 17: Representação gráfica, depois da calibração, do sinal sinusoidal, variando a escala horizontal:
a) 5 ms/div; b) 10 ms/div; c) 20 ms/div; d) 50 ms/div.

Posteriormente, para uma onda sinusoidal com 25 Hz de frequência e 5 V de amplitude, mantendo fixa a escala horizontal a 5 ms/div, variou-se a escala vertical utilizando {1; 2; 5; 10} V/div. Os resultados obtidos são representados na figura 18.

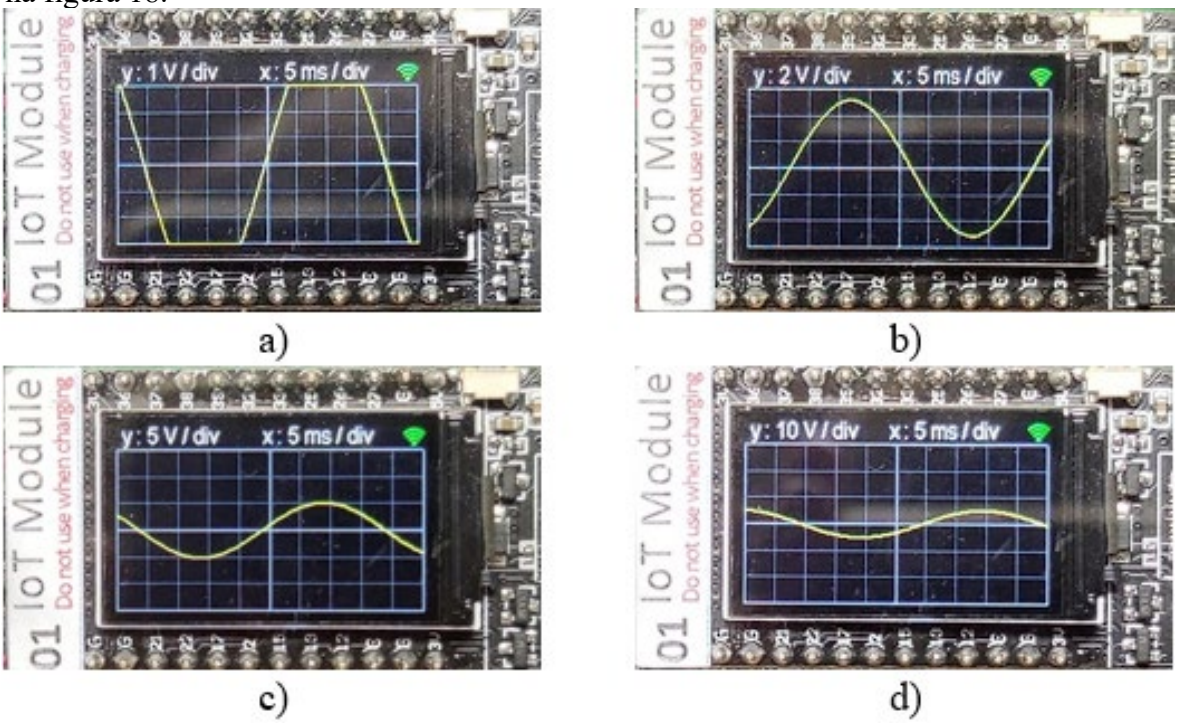

Figura 18: Representação gráfica, depois da calibração, do sinal sinusoidal, variando a escala vertical:
a) 1 V/div; b) 2 V/div; c) 5 V/div; d) 10 V/div.

Para finalizar a série de testes efetuados depois da calibração, aplicamos uma tensão constante de {−10; −5; 0; 5; 10} V, mantendo fixa a escala horizontal em 5 ms/div e a vertical em 5 V/div. Na figura 19 apresentamos os resultados obtidos.

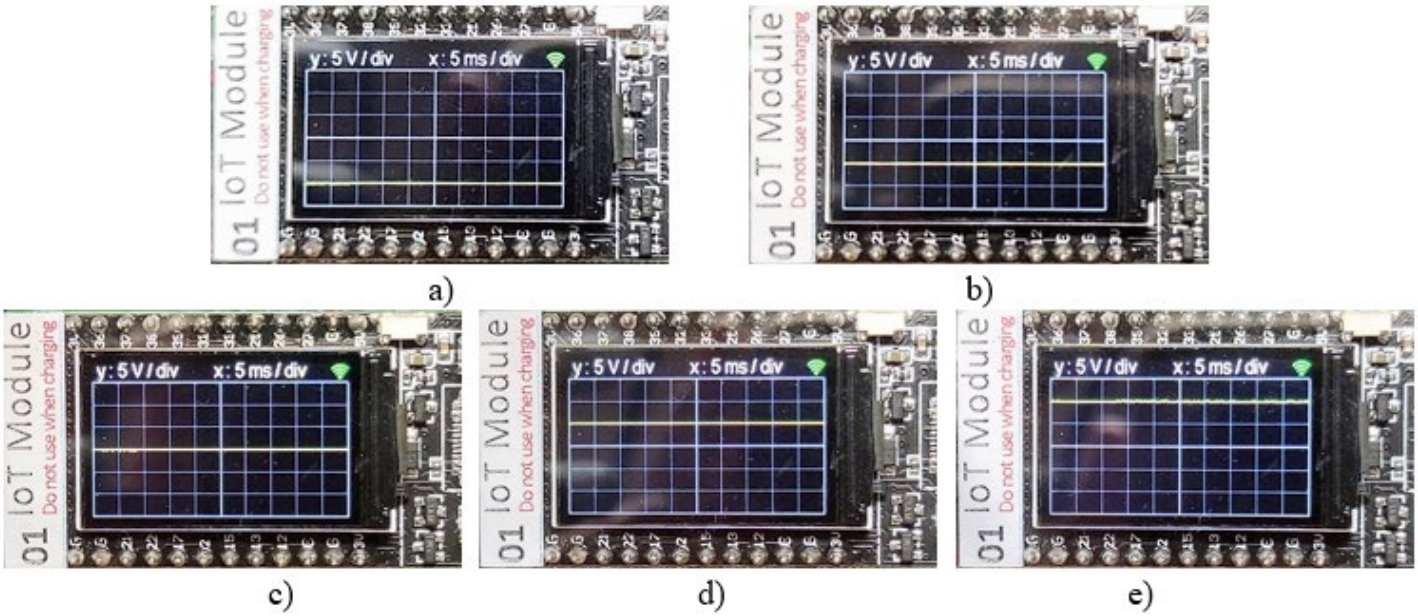

Figura 19: Representação gráfica, depois da calibração, de uma tensão contínua:
a) −10 V; b) −5 V; c) 0 V; d) 5 V; e) 10 V.

Analisando os resultados obtidos depois da calibração, verificamos que as tensões medidas são representadas de forma correta no módulo IoT.

As ondas representadas graficamente, estão corretamente centradas na grelha de pontos. Por exemplo, podemos observar esta exatidão na figura 19 pela sobreposição do sinal medido com a linha horizontal correspondente à escala vertical com o valor da tensão contínua que é medida em cada caso.

Para além disso, os sinais representados estão coerentes com as escalas selecionadas.

De modo a comparar os resultados laboratoriais com os resultados simulados na secção 3 e a demonstrar limitações do módulo IoT, realizámos três testes adicionais.

Começámos por aplicar uma onda sinusoidal de 25 Hz, mantendo fixas as escalas horizontal a 5 ms/div e a vertical a 2 V/div, variando a amplitude entre {2; 4; 6} V. Os resultados obtidos estão representados na figura 20.

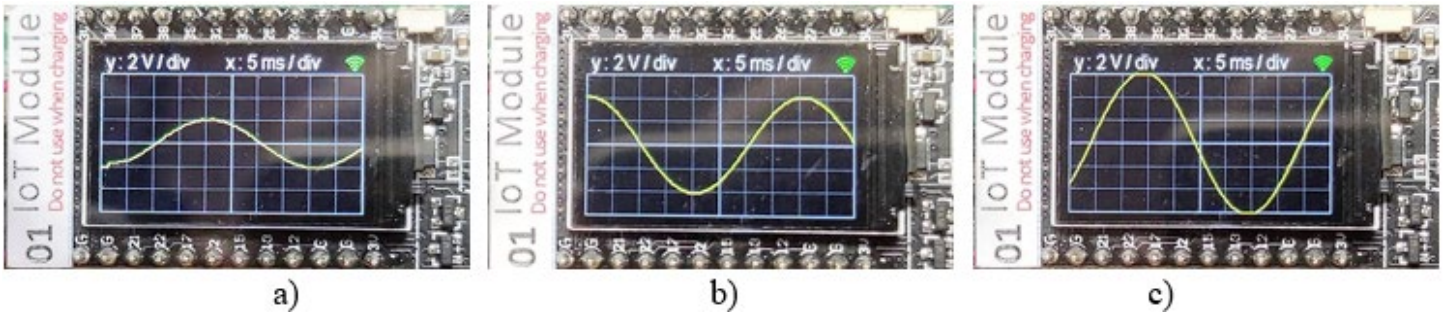

Figura 20: Representação gráfica, depois da calibração, de uma onda sinusoidal, variando a amplitude:
a) 2 V; b) 4 V; c) 6 V.

De seguida, para uma onda sinusoidal com 25 Hz de frequência, 5 V de amplitude e mantendo fixas as escalas horizontal a 5 ms/div e vertical a 2 V/div, aplicamos diversas tensões de offset: {−5; −1; 0; 1; 5} V. Apresentamos, na figura 21, os resultados obtidos.

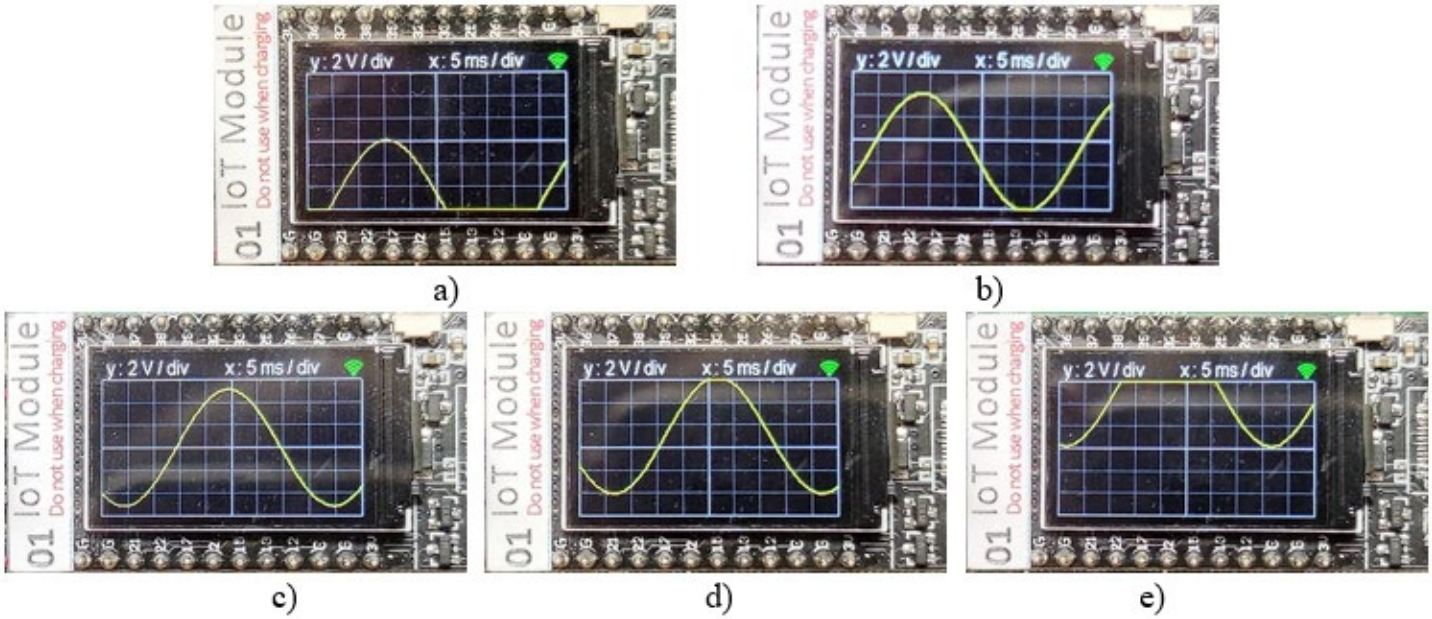

Figura 21: Representação gráfica, depois da calibração, de uma onda sinusoidal, aplicando diferentes tensões de offset: a) −5 V; b) −1 V; c) 0 V; d) 1 V; e) 5 V.

Posteriormente, para uma onda sinusoidal de 5 V de amplitude, com 0 V de tensão de offset, mantendo fixas as escalas horizontal a 5 ms/div e vertical a 2 V/div, variamos o valor da frequência, utilizando 25 Hz, 50 Hz e 100 Hz. Obtemos os resultados representados na figura 22.

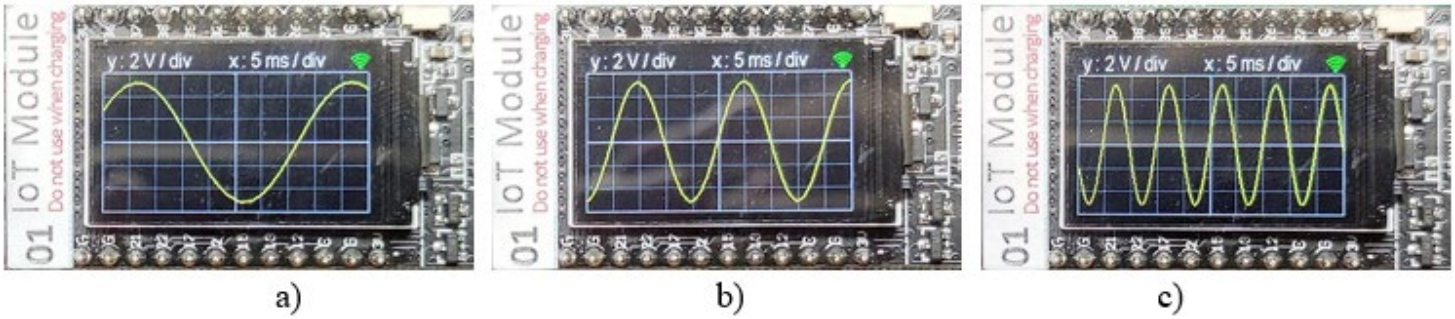

Figura 22: Representação gráfica do sinal sinusoidal, depois da calibração, utilizando diferentes frequências: a) 25 Hz; b) 50 Hz; c) 100 Hz.

Finalmente, de modo a demonstrar limitações do módulo IoT, aplicamos uma onda triangular com 25 Hz de frequência, 5 V de amplitude, escala horizontal igual a 5 ms/div e escala vertical igual a 2 V/div. O resultado obtido encontra-se representado na figura 23.

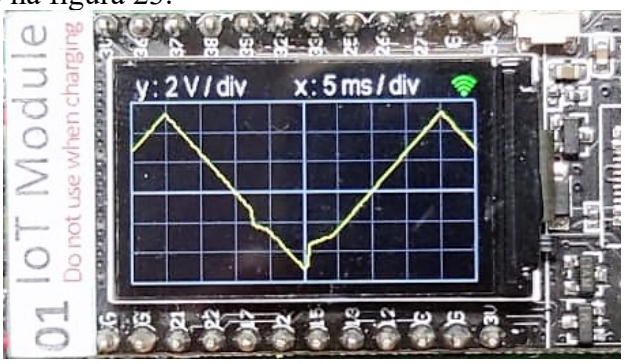

Figura 23: Demonstração das limitações de processamento do módulo IoT.

Devido ao sistema operativo do módulo IoT alternar entre *threads* e ao *python* não ser a linguagem ideal para a implementação da aplicação no módulo, verifica-se uma distorção no desenho da onda. Com o aumento da frequência verificou-se que este fenómeno ocorre com maior probabilidade. Optou-se, no entanto, por medir uma onda de 25 Hz de frequência, uma vez que foi a mais utilizada nos testes efetuados e devido à facilidade de visualização da sua forma de onda no pequeno ecrã do módulo. Por conter porções lineares, que facilitam a visualização da limitação, escolheu-se uma onda triangular.

Também foi possível verificar o bom funcionamento da funcionalidade que assegura que não são imprimidos pontos fora da grelha (na vertical). Sempre que estes excedem o limite superior ou inferior da grelha, são imprimidos neste limite.

Iremos proceder à comparação dos testes de laboratório com os testes de simulação na próxima secção "5. Conclusão".

5.Conclusão

Através da realização deste trabalho experimental, tivemos contacto com um sistema embebido que combina hardware computacional com software projetado para uma função específica.

Desenvolvemos um projeto, através da criação de uma aplicação de software na linguagem de programação *python* tendo em vista implementar um pequeno osciloscópio, utilizando um hardware para desenvolvimento de soluções IoT: módulo IoT representado na figura 1.

Inicialmente estudámos o circuito de hardware base para a realização do trabalho. Posteriormente, desenvolvemos o software para implementação do osciloscópio. Na aula de laboratório, procedemos ao teste e ajuste deste software, recorrendo à sua calibração para obter resultados com melhor exatidão.

Comparando os resultados dos testes efetuados no laboratório com os obtidos na simulação, podemos concluir que estes se assemelham, tal como seria de esperar. As pequenas variações devem-se às incertezas das grandezas medidas, associadas ao uso do gerador de funções e do módulo IoT. Estes aparelhos não são instrumentos ideais, mas sim reais, possuindo uma impedância interna finita e bem definida (para cada frequência).

Desta forma, enquanto que na simulação é possível definir os parâmetros das tensões a medir de forma exata, experimentalmente o mesmo não acontece. Os doze testes experimentais realizados confirmam o correto funcionamento de todas as funcionalidades do osciloscópio, cuja implementação no módulo IoT foi por nós desenvolvida.

Bibliografia

[1] Sistemas Electrónicos - Slides das Aulas Teóricas 2021/2022, 2º Semestre (LEEC), do professor José Gerald, IST-DEEC;
[2] 2º Semestre 2021/2022, Sistemas Electrónicos, 3º Trabalho de Laboratório - µOscilloscope: Osciloscópio através de um sistema embebido - Enunciado;
[3] *Python* 3.8.