
PRODUTO EDUCACIONAL - MNPEF - PHOTOGATE
COLETOR PHOTOGATE
Opções de ensaios
Opção 01
Ensaio de queda livre
Opção 02
Ensaio de pêndulo
Opção 03
Ensaio de plano inclinado
Posições dos sensores
Sensor 01
cm
Sensor 02
cm
Sensor 03
cm
Sensor 04
cm
Sensor 05
cm
Importar
Resultados
Teste
Distância
Tempo
Limpar
Exportar

Sumário

LISTA DE FIGURA

LISTA DE TABELA

APRESENTAÇÃO

A física é uma área de conhecimento estudado desde sociedade antiga até a moderna por centenas de anos no intuito de compreender os fenômenos naturais existentes no Universo e partir daí construir modelos que descrevem diversas complexidades que contribui para a humanidade em benefícios de valor inestimável no desenvolvimento de tecnologia moderna. No entanto é de suma importância que o processo científico esteja presente nos cotidianos das pessoas, e particularmente na formação dos estudantes do ensino básico para desenvolver o senso crítico e investigador, e neste sentido ter contato com a práticas cientificas necessária para compreender o e interpretar o mundo como todo, no qual o ensino da física possui esse papel de trabalhar os conceitos fundamentais da natureza, como ferramenta pedagógica e relacionar teoria e prática com a vida do aluno em situações reais.

Ao pensar nessas questões desenvolvemos um material que possa relacionar os conceitos de física teórica e prática aos alunos do ensino médio no formato de um manual de atividades experimentais que utiliza placa microprocessadora Raspberry PI nos estudos da gravidade na Mecânica Clássica e através dos experimentos do Plano Inclinado, Queda Livre e Pêndulo Simples.

Uma característica significativa deste manual é a poder integrar todos os alunos, com ou sem necessidades especiais nas atividades e assim garantir a acessibilidade em todas as etapas do experimento.

INTRODUÇÃO

Este manual tem como objetivo devolver experimentos de baixo custo para pessoas com habilidades distintas e que possa atuar individualmente de forma simples, interativo, fácil compreensão, e que obtenhas informações necessárias independente das condições ambientais e sensório motoras e com baixo esforço físico.

Os ensaios experimentais deste manual estão voltados para os estudos da gravitação dos corpos, tais como o pêndulo simples, plano inclinado e queda livre com auxílio do microprocessador Raspberry PI, esquema de montagem de sensores e dispositivos para coleta de dados e aplicação baseado na linguagem Python de alto nível. Além de adquirir os conceitos essenciais dos fenômenos físicos, também tem como alvo empregar meios de comunicação para todos os alunos dos alunos presentes nas salas de aulas comum, em explorar sensações táteis, auditivas e visuais, sendo assim capaz de produzir um novo significado aos fenômenos físicos.

RASPBERRY PI

A escolha do microprocessador Raspberry pi[1] é devido a sua facilidade de aquisição no mercado e pela quantidade de informações já disponíveis em vídeoblogues existente na internet e fórum especializados na funcionalidade do microprocessador. O ponto importante da escolha do módulo é a sua facilidade em manuseio na preparação de conteúdo pelo professor e necessita de pouco conhecimento em linguagem de programação e eletrônica no qual facilita a sua inserção no ensino fundamental e médio.

FIGURA 1 -RASPBERRYPI.ORG

1 Raspberry Pi é uma série de computadores de placa única do tamanho reduzido, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, desenvolvido no Reino Unido pela Fundação Raspberry Pi. Todo o hardware é integrado numa única placa.

Em 2006, os primeiros conceitos do Raspberry Pi foram baseados no microcontrolador Atmel ATmega644. Nesse tempo seus esquemas e layout de PCB foram disponíveis ao público.

A Fundação trustee Eben Upton reuniu professores, acadêmicos e admiradores da computação para criar um computador que motivasse as crianças a desenvolverem algo criativo. O computador é inspirado da Acorn BBC Micro de 1981 Modelo A, Modelo B e Modelo B+ são referências aos modelos originais de escolaridade do computador BBC Microbritânico, desenvolvido pela Acorn Computers. A primeira versão de arquitetura ARM do Raspberry Pi foi montado em um pacote do mesmo tamanho de uma memória stick USB.Tinha uma porta USB em uma extremidade e uma porta HDMI na outra.

GPIO

Um recurso poderoso do Raspberry Pi é a linha de pinos GPIO (entrada / saída de uso geral) ao longo da borda superior da placa. Um cabeçalho GPIO de 40 pinos é encontrado em todas as placas Raspberry Pi atuais. Antes do Pi 1 Modelo B + (2014), as placas eram compostas por um cabeçalho mais curto de 26 pinos.

Esta pinagem GPIO foi projetada para ser uma referência rápida e interativa para os pinos GPIO do Raspberry Pi, além de um guia abrangente para as interfaces GPIO do seu Raspberry Pi.

Utiliza-se o link para orientação dos pinos: https://pinout.xyz/

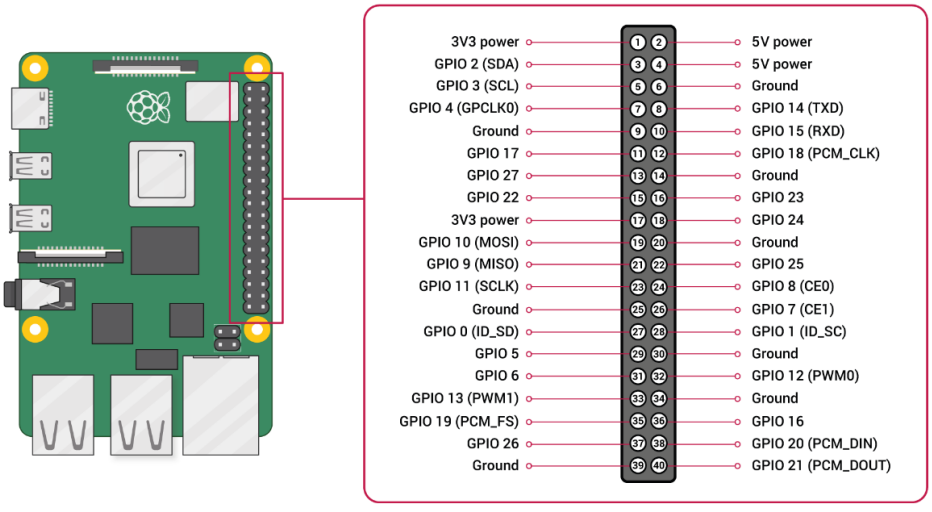


FIGURA 2 – FONTE: HTTPS://WWW.RASPBERRYPI.ORG/DOCUMENTATION/USAGE/GPIO/:

Qualquer um dos pinos GPIO pode ser designado (no software) como um pino de entrada ou saída e usado para uma ampla gama de propósitos.

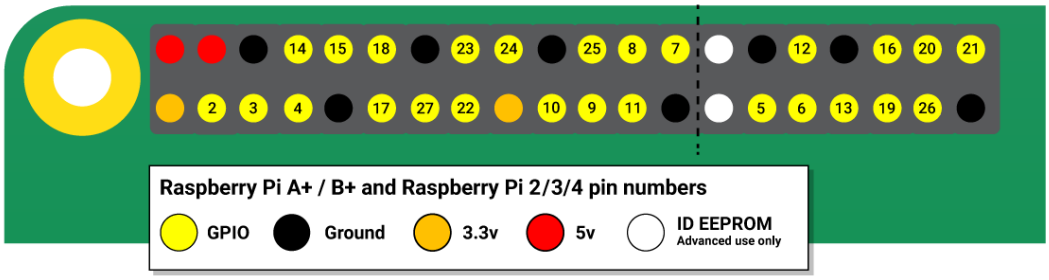


FIGURA 3 - FONTE: HTTPS://WWW.RASPBERRYPI.ORG/DOCUMENTATION/USAGE/GPIO/

Nota: a numeração dos pinos GPIO não está em ordem numérica; os pinos GPIO 0 e 1 estão presentes na placa (pinos físicos 27 e 28), mas são reservados para uso avançado (veja abaixo).

Tensões

Dois pinos de 5V e dois pinos de 3V3 estão presentes na placa, bem como vários pinos de aterramento (0V), que não são configuráveis. Os pinos restantes são todos pinos 3V3 de uso geral, o que significa que as saídas são definidas como 3V3 e as entradas são tolerantes a 3V3.

Saídas

Um pino GPIO designado como um pino de saída pode ser definido como alto (3V3) ou baixo (0V).

Entradas

Um pino GPIO designado como um pino de entrada pode ser lido como alto (3V3) ou baixo (0V). Isso é facilitado com o uso de resistores internos pull-up ou pull-down. Os pinos GPIO2 e GPIO3 têm resistores pull-up fixos, mas para outros pinos isso pode ser configurado no software.

Assim como dispositivos simples de entrada e saída, os pinos GPIO podem ser usados com uma variedade de funções alternativas, algumas estão disponíveis em todos os pinos, outras em pinos específicos.

PWM (modulação por largura de pulso)

- Software PWM disponível em todos os pinos
- Hardware PWM disponível em GPIO12, GPIO13, GPIO18, GPIO19

SPI

- SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
- SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)

Pinagem GPIO

- Dados: (GPIO2); Relógio (GPIO3)
- Dados EEPROM: (GPIO0); Relógio EEPROM (GPIO1)

Serial

- TX (GPIO14); RX (GPIO15)

SISTEMA OPERACIONAL (Raspberry Pi OS)

O Raspberry Pi utiliza o Raspberry Pi OS como o sistema operacional oficial para todos os seus modelos fabricado e para obter o primeiro contado com o sistema operacional é necessário instalar a imagem oferecida pelo próprio site da raspberrypi.org. O site informará todas as ações necessária para instalar o sistema operacional no cartão SD com êxito.

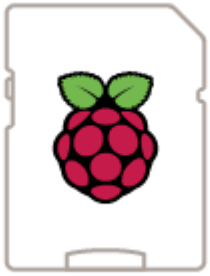


FIGURA 4 - FONTE:HTTPS://WWW.RASPBERRYPI.ORG/DOWNLOADS/RASPBERRY-PI-OS/

O Raspberry Pi OS vem pré-instalado com muitos softwares para educação, programação e uso geral. Possui Python, Scratch, Sonic Pi, Java e outros. Segue abaixo a versão sugerida até a data presente:

- Raspberry Pi OS (32 bits) com desktop e software recomendado
- Imagem com desktop e software recomendado baseado em Debian Buster
- Versão: maio de 2020
- Data de lançamento: 27/05/2020
- Versão do kernel: 4,19
- Tamanho: 2523 MB
- Link de acesso ao sistema operacional: https://downloads.raspberrypi.org/raspios_full_armhf_latest

FIGURA 5 - FONTE: O AUTOR

Após concluir a instalação do sistema operacional é necessário executar as linhas do programa na linguagem Python de acordo com seu compilador de interesse.

PYTHON

A linguagem de programação Python possui um leque maior de recursos para os sistemas embarcados como, por exemplo, interagir com um GPIO (General Purpose Input Output), ou melhor, os pinos programáveis de entrada e saída da placa Raspberry PI.

Ambiente de desenvolvimento integrado (IDE)

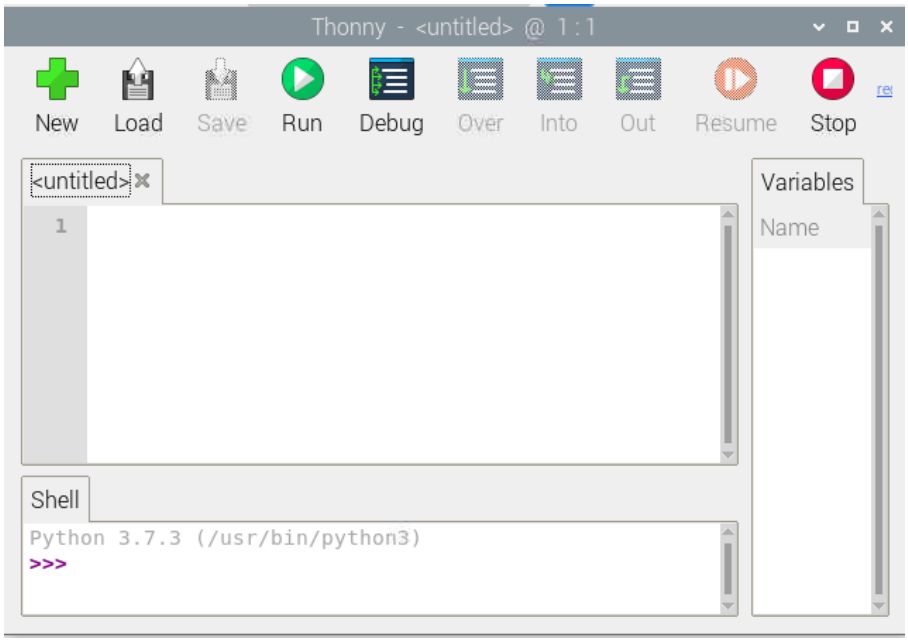


FIGURA 6 - FONTE: O AUTOR

No sistema operacional Raspberry PI OS já vem com IDE[2] Thonny é um ambiente de desenvolvimento integrado para Python projetado para iniciantes.

[2] IDE, do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

GPIO em Python

O uso da biblioteca GPIO Zero torna mais fácil começar a controlar dispositivos GPIO com Python. O GPIO Zero é instalado por padrão na imagem Raspberry PI OS e na imagem Raspberry Pi Desktop para PC / Mac, ambos disponíveis em raspberrypi.org.

Siga estes guias para instalar o sistema operacional, incluindo para PCs que usam o recurso GPIO remoto.

Primeiro, atualize sua lista de repositórios ao digitar no terminal para atualizar e instale os pacotes:

FIGURA 7 - FONTE:HTTPS://GPIOZERO.READTHEDOCS.IO/EN/STABLE/INSTALLING.HTML

MATERIAIS

A lista abaixo apresenta quais materiais e quantidade necessários para elaboração dos ensaios experimentais de fácil acesso em lojas físicas e virtuais especializadas em eletroeletrônica e de baixo custo.

Lista de Materiais

TABELA 1 - MATERIAIS

Quantidade	Materiais
1	Protoboard (mínimo 400 pontos)
2	resistores de 12K ohm ou 10K ohm
1	resistor de 470K ohm
50 cm	fio jumper amarelo de 5mm
50 cm	fio jumper azul de 5mm
50 cm	fio jumper vermelho de 5mm

50 cm	fio jumper preto de 5mm
50 cm	fio jumper branco de 5mm
50 cm	50 cm de fio jumper verde de 5mm
1	Vibracall de 3V
1	circuito integrado – CI 401406
1	1Buzzer ativo 5V ou 12V
5	Led transmissor infravermelho
5	Led receptor infravermelho
1	Led amarelo ou branco
1	fonte de bancada com saída de 3V e 5V
3	conectores machos de fio jumper
3	conectores fêmeas de fio jumper
1	Multímetro
1	Placa processadora Raspberry PI
1	rolo de fita isolante
1	Pastilha Piezo elétrico – PZT
1	eletroímã 12V

FONTE: O AUTOR

ESQUEMA DE MONTAGEM

Para facilitar o experimento e na prática de ensino é sugerido que o esquema de montagem seja utilizado na placa protoboard ou também conhecido como placa de prototipagem, fácil manuseio na identificação dos pontos e também possuir conexões internas para que você possa conectar os dispositivos e jumpers e sua vantagem é de não necessitar de solda para conectar os circuitos.

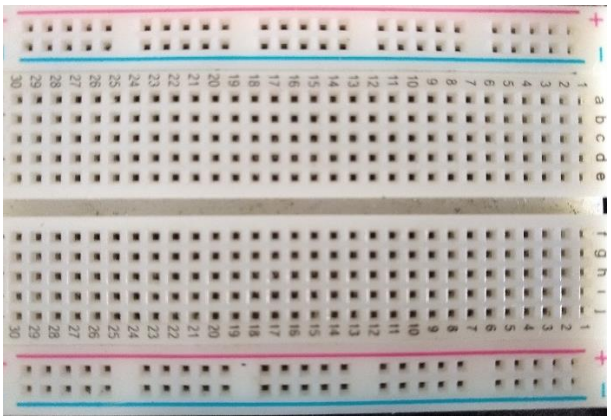
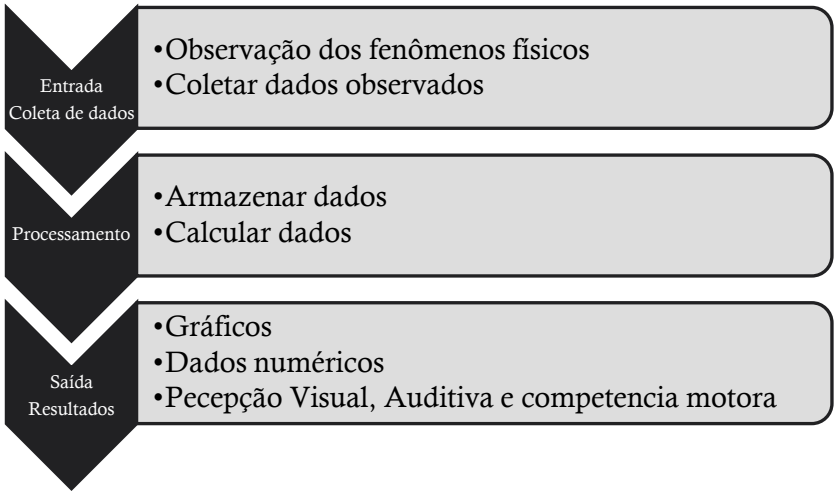


FIGURA 8- PROTOBOARD - FONTE: O AUTOR

O desenvolvimento da aplicação é baseado em três processos que compõe o experimento:



- Coleta de dados com uso de sensores acoplado nas portas físicas (GPIO) do Raspberry PI junto ao esquema de montagem do experimento.
- Processamento de dados que tem a finalidade de registrar os eventos ocorrido no experimento e controlar as funções necessárias e armazenamento de dados.
- Saída de dados para registro de resultados em tabelas, planilhas, saídas sonoras (Bip), visuais (LED).

Sensores e Dispositivos

Nas elaborações dos ensaios experimentais será necessário desenvolver dois esquemas de fácil montagem no qual utiliza sensores de entrada e saída.

Entrada:

- Led emissor e receptor de infra vermelho
- Pastilha piezoelétrica ou micro switch

Saída:

- Led difuso
- Buzzer (5V-12V)
- VibraCall 3V

Led Emissor IR

O photogate utiliza um Led emissor infravermelho de 5 mm e comprimento de onda de 940 nm, o componente funciona a partir da tecnologia infravermelha, não é perceptível ao olho humano, contudo, a partir de uma câmera é possível ver nitidamente o sinal infravermelho sendo emitido pelo Led.

Especificações e características (Led Emissor IR):

- Tensão de operação: 1,2VDC
- Corrente de operação: 20mA
- Comprimento da onda: 940nm
- Ângulo: (15º - 20°)

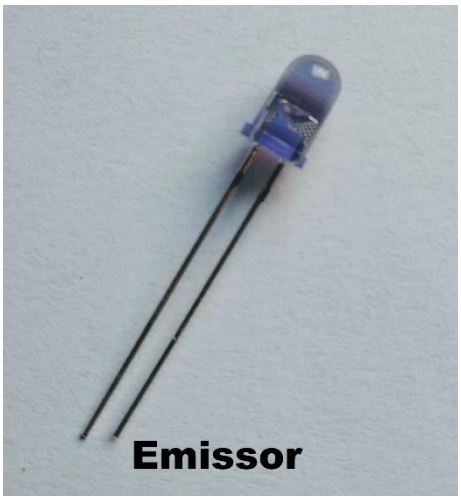


FIGURA 9- FONTE: O AUTOR

Fototransistor IR

Também é necessário um receptor infravermelho Fototransistor IR de 5 mm, componente eletrônico que também funciona através de tecnologia infravermelha que é ativada pela incidência de luz infravermelha e o transistor passa a conduzir, o que permite a passagem de corrente elétrica do coletor para o emissor.

Especificações e características (Fototransistor IR):

- Tensão de operação: 1,1 a 1,4VDC
- Corrente de operação: 10mA

- Potência máxima: 70mW
- Comprimento da onda: 940nm
- Ângulo: (15º - 30°)

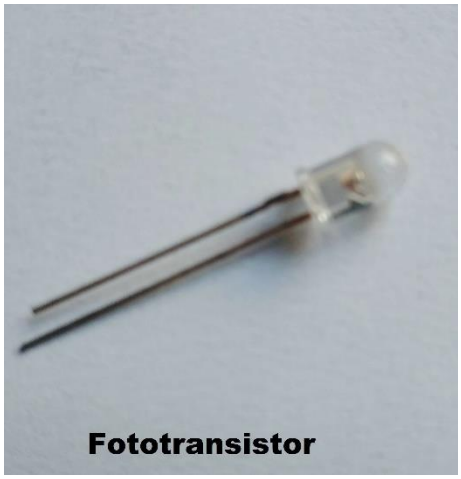


FIGURA 10 - FONTE: O AUTOR

Pastilha Piezoelétrica

O Módulo Piezoelétrico Sensor de Vibração e Toque é um dispositivo que varia a tensão de saída proporcionalmente à força (pressão) aplicada sobre o transdutor piezoelétrico.

O Módulo Piezoelétrico Sensor de Vibração e Toque tem a capacidade de informar a extensão da vibração detectada. Quando a pastilha piezo é pressionada, uma tensão elétrica é gerada e o pino digital do microprocessador Raspberry PI vai ler este valor de tensão e o usuário poderá definir uma ou várias ações para serem executadas com base nas leituras.

Especificações e características:

- Tensão de operação: 3,3 ou 5VDC
- Corrente de operação: <1mA
- Temperatura de operação: -10º a 70º celsius
- Tipo de saída do sinal: analógica
- Diâmetro da pastilha piezo: 20mm

FIGURA 11 - FONTE: O AUTOR

Buzzer Ativo

O Buzzer Ativo é um pequeno alto-falante destinado a emitir sinais sonoros a partir do oferecimento de energia DC ao módulo, não variando a frequência de emissão e a principal finalidade do Buzzer Ativo é a emissão de sinais sonoros como forma de alerta para que o operador fique informado que algo está ocorrendo.

FIGURA 12 - FONTE: O AUTOR

Vibracall

O Vibracall ou Micro motor de vibração é semelhante àqueles responsáveis pelo efeito de vibração em celulares.

Especificações:

- Tensão de operação: 2,5 – 4V
- Corrente de operação: 90mA máx
- Velocidade de rotação: 9000 RPM
- Comprimento do cabo: 3cm
- Dimensão total: 40 x 10 x 3mm

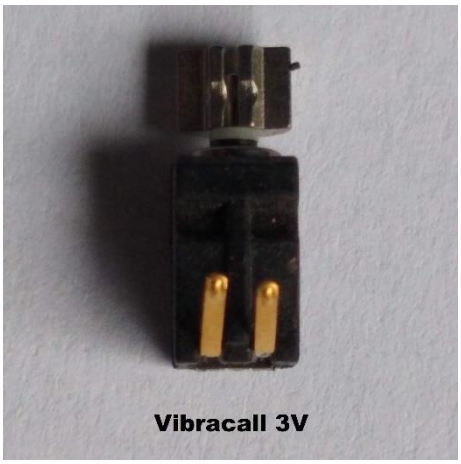


FIGURA 13 - FONTE: O AUTOR

Esquema de montagem 1

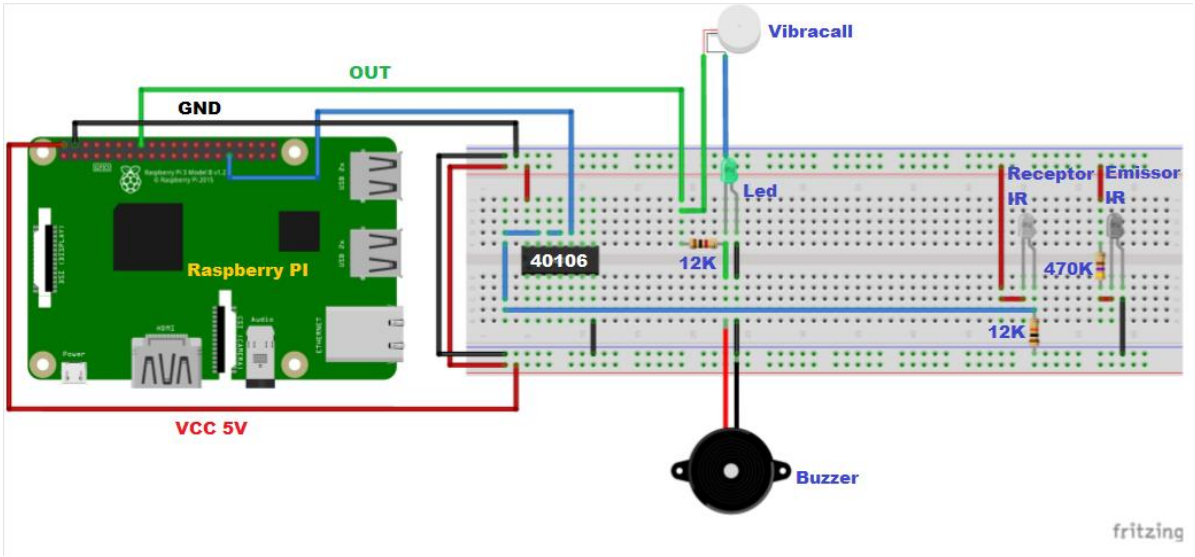


FIGURA 14 - FONTE: O AUTOR

Esquema de montagem 2

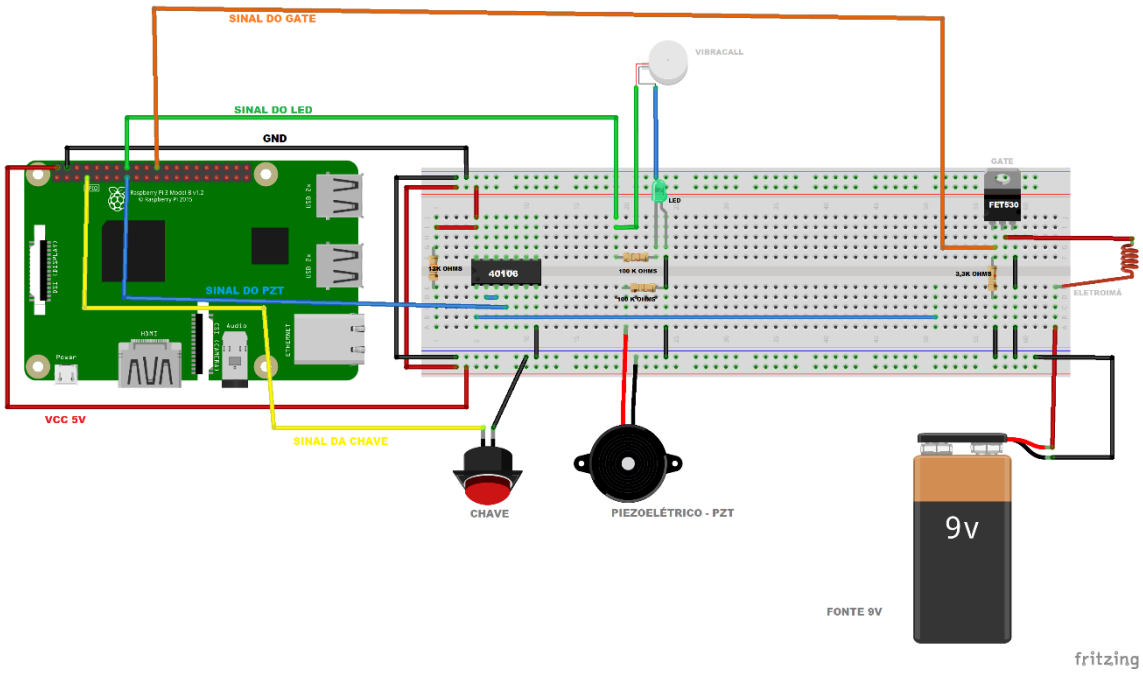


FIGURA 15: FONTE DO AUTOR

Fluxo do programa

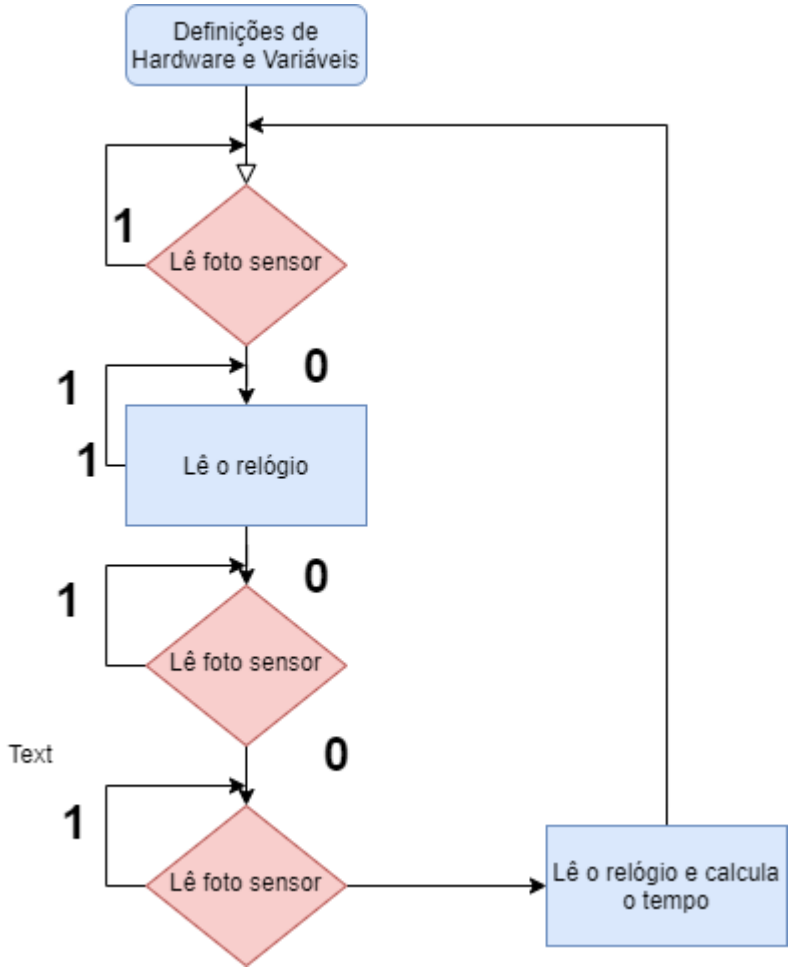


FIGURA 16 -FONTE: O AUTOR

PROCEDIMENTOS

Queda livre

Este procedimento dependerá de duas grandezas, é necessário medir a posição da altura em metros com auxílio de uma régua e posteriormente o microprocessador marcará o tempo de queda, em segundos, com auxílio de um eletroímã e um dispositivo para fim de curso (micro switch ou piezoelétrico).

Para altura do objeto será ajustado cinco posições distintas que será registrado, no instante em que o objeto é abandonado em queda livre, o sinal será medido pelos fotosensores, também será emitido sinalizações sonoras do Buzzer, visuais pelo Led e um sinal tátil gerado pelo Vibracall.

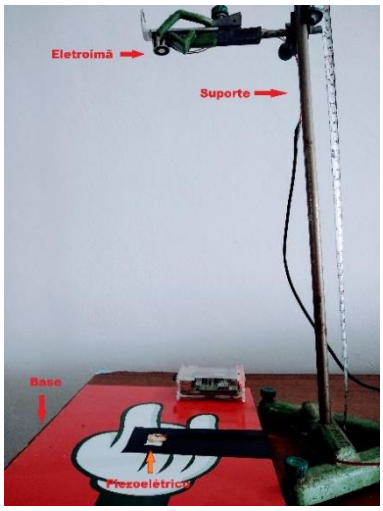


FIGURA 17 - FONTE: O AUTOR

O eletroímã é o gatilho para liberação do objeto, quando ligado produz uma força magnética que fixa uma esfera metálica, o dispositivo faz a leitura de um botão, quando desligado libera a esfera e aciona o cronometro na Raspberry Pi até atingir um piezo elétrico para finalizar o cronometro.

Será registrado cada posição no microprocessador e para cada posição será armazenada automaticamente o tempo de queda na elaboração da tabela da posição (m) x tempo (s):

Os dados coletados na tabela serão utilizados para plotar o gráfico de posição em função do tempo Tempo(s)x Posição(m):

TABELA 2 - FONTE: O AUTOR

Tempo(s)	Posição(m)
t1	h1
t2	h2
t3	h3
t4	h4
t5	h5

- Para linearização dos gráficos é necessário plotar gráfico ((tempo)²(s²) x posição(m)):

TABELA 3 - FONTE: O AUTOR

(Tempo)²(s²)	Posição(m)
$t1^2$	h1
$t2^2$	h2
$t3^2$	h3
$t4^2$	h4
$t5^2$	h5

Plano inclinado

Assim como o experimento da queda livre o eletroímã é o gatilho para liberação do objeto, quando ligado produz uma força magnética que o objeto, o dispositivo faz a leitura de um botão, quando desligado libera a esfera e aciona o cronometro na Raspberry Pi até atingir um piezo elétrico ou micro switch para finalizar o cronometro.

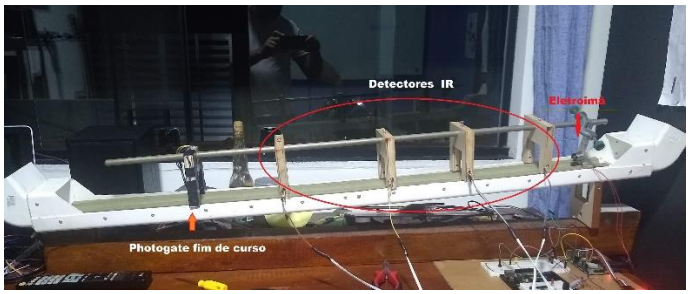


FIGURA 18 - FONTE: O AUTOR

- Dependerá de duas grandezas (distância (m), Intervalo de tempo (s)).
- Armazenar em até 5 posições distintas com relação a origem.
- Um botão iniciará a contagem do tempo de queda até o fim do curso.
- Registrar o tempo para cada distância.

TABELA 4 - FONTE: O AUTOR

Distância (m)	Tempo (s)
d1	t1
d2	t2
d3	t3
d4	t4
d5	t5

- Plotar gráfico da distância em função do tempo
- Plotar gráfico de velocidade em função do tempo

TABELA 5 - FONTE: O AUTOR

Tempo (s)	Velocidade (m/s)
t1	v1=d1/t1
t2	v2=d2/t2
t3	v3=d3/t3
t4	v4=d4/t4
t5	v5/t5/d5

Pendulo simples

O experimento utiliza um peso de massa m e um fio de comprimento L fixo ao suporte universal, que por sua vez possui o Photogate anexado na haste, assim que inicio o movimento do pêndulo o cronometro é iniciado a contagem a partir de um gatilho que armazena o período de oscilação do pendulo e o comprimento L do fio, essas informações são tabeladas posteriormente para plotagem gráfica para modelagem.

FIGURA 19 - FONTE: O AUTOR

- Dependerá de duas grandezas (comprimento do fio L(m), período completo T(s).
- Altura do objeto será ajustado para cinco posições que será guardado.
- Um botão iniciará a contagem do tempo de queda até o fim do curso.
- Armazenar 5 comprimentos distintos L(m)

- Registrar o período de oscilação T(s)
- Plotar gráfico $T^2(s^2)$ x L(m)

TABELA 6: FONTE: O AUTOR

Ensaio	Comprimento L (m)	T1 (s)	T2 (s)	T3 (s)	T4 (s)	T5 (s)	T médio	Tmédio² (s²)	gravidade (m/s²)
1									
2									
3									
4									
5									

CÓDIGO DO PROGRAMA

Segue abaixo as alinhas de programação para criação da aplicação do software dos ensaios de queda livre, plano inclinado e pendulo simples, basta seguir o passo a passo no compilador para criação do programa executável:

20200724_EnsaioFisica.py

```
import sys
import tkinter as tk
import tkinter.ttk as ttk
import EnsaioFisica_support
from gpiozero import LED, Button #controlar pinos de entrada e saida do RPi
from time import time, sleep #biblioteca de marcador de tempo
import matplotlib.pyplot as plt #plotar dados de entrada
import pygame #produzir beeb sonoros
import threading #programações em paralelo

def registraTempo():
    global tempos, ensaio
    if ensaio > 0:
        tempos.append(int(round(time() * 1000)))
        som.blink(on_time=0.008, off_time=0.001,n=1)

def marcaPhotogate01():
    registraTempo()

def marcaPhotogate02():
    registraTempo()

def marcaPhotogate03():
    registraTempo()

def marcaPhotogate04():
    registraTempo()
```

20200724_EnsaioFisica.py

```
def marcaPhotogate04():
    registraTempo()

def marcaPhotogate05():
    global ensaio
    registraTempo()
    if ensaio == 1:
        ensaio = 0

#variaveis globais
photo01 = Button(pin=5, pull_up=None, active_state=False) #GPIO5 -> conetor J8 pino 29
photo02 = Button(6) #GPIO6 -> conetor J8 pino 31
photo03 = Button(13) #GPIO13 -> conetor J8 pino 33
photo04 = Button(19) #GPIO19 -> conetor J8 pino 35
photo05 = Button(pin=26, pull_up=None, active_state=False) #GPIO26 -> conetor J8 pino 37
gatilho = Button(12) #GPIO12 -> conetor J8 pino 32
som = LED(21) #GPIO21 -> conetor J8 pino 40
vibra = LED(20) #GPIO20 -> conetor J8 pino 38
led1 = LED(16) #GPIO16 -> conetor J8 pino 36
eletroima = LED(25) #GPIO25 -> conetor J8 pino 22 Eletroímã
imagem = None


photo01.when_pressed = marcaPhotogate01
photo02.when_pressed = marcaPhotogate02
photo03.when_pressed = marcaPhotogate03
photo04.when_pressed = marcaPhotogate04
photo05.when_pressed = marcaPhotogate05

def ensaioQuedaLivre(top):
    global tempos, ensaio, historico, graficoQueda
```

20200724_EnsaioFisica.py

```

photo01.when_pressed = marcaPhotogate01
photo02.when_pressed = marcaPhotogate02
photo03.when_pressed = marcaPhotogate03
photo04.when_pressed = marcaPhotogate04
photo05.when_pressed = marcaPhotogate05

def ensaioQuedaLivre(top):
    global tempos, ensaio, historico, graficoQueda
    print('Ensaio de queda livre. Pressione o gatilho para iniciar!')
    top.lbl_Status.config(text='Iniciando ensaio de queda livre...')
    eletroima.on()
    pygame.init()
    pygame.mixer.music.load('EnsaioQuedaLivre.mp3')
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    pygame.mixer.music.load("PressioneGatilho01.mp3")
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    top.lbl_Status.config(text='Pressione o gatilho para iniciar')
    while not gatilho.is_pressed:
        ensaio = 0
    while gatilho.is_pressed:
        ensaio = 0
    tempos.clear()
    eletroima.off()
    ensaio = 1
    while ensaio == 1:
        print('Aguardando...')
```

20200724_EnsaioFisica.py

```
        ensaio = 0
    tempos.clear()
    eletroima.off()
    ensaio = 1
    while ensaio == 1:
        print('Aguardando...')
        sleep(1.5)
    som.blink(on_time=0.02, off_time=0.02,n=1)
    resultado = list()
    print(tempos)
    tempoinicial = tempos[0]
    for x in range(len(tempos)):
        tempos[x] = (tempos[x] - tempoinicial)/1000

    #tempos.insert(0,0.0)
    #tempos.pop()
    #print(tempos)
    top.lbl_Status.config(text='Fim do ensaio. Apresentando resultados...')
    pygame.mixer.music.load("FimEnsaio.mp3")
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    print('Final do ensaio de queda livre.')
    print('{};{}'.format(top.posSensor01.get(), type(top.posSensor01.get())))
    if len(top.posSensor01.get()) > 0: resultado.append(float(top.posSensor01.get()))
    if len(top.posSensor02.get()) > 0: resultado.append(float(top.posSensor02.get()))
    if len(top.posSensor03.get()) > 0: resultado.append(float(top.posSensor03.get()))
    if len(top.posSensor04.get()) > 0: resultado.append(float(top.posSensor04.get()))
    if len(top.posSensor05.get()) > 0: resultado.append(float(top.posSensor05.get()))
    top.LblTempos.config(text='')

```

20200724_EnsaioFisica.py

```
    print('Final do ensaio de queda livre.')
    print('{};{}'.format(top.posSensor01.get(), type(top.posSensor01.get())))
    if len(top.posSensor01.get()) > 0: resultado.append(float(top.posSensor01.get()))
    if len(top.posSensor02.get()) > 0: resultado.append(float(top.posSensor02.get()))
    if len(top.posSensor03.get()) > 0: resultado.append(float(top.posSensor03.get()))
    if len(top.posSensor04.get()) > 0: resultado.append(float(top.posSensor04.get()))
    if len(top.posSensor05.get()) > 0: resultado.append(float(top.posSensor05.get()))
    top.LblTempos.config(text='')

    if len(resultado) == len(tempos):
        plt.ylabel('Distância [cm]')
        plt.plot(tempos, resultado)
        historico['Queda livre_{:02}'.format(len(historico)+1)] = (resultado, tempos)
        #top.LblTempos.config(text=historico['Queda livre_{:02}'.format(len(historico))])
        dadosensaio = top.ListaResultados.insert('', 'end', text='QL_{:02}'.format(len(historico)),
                                                 values=('[cm]','[s]',''))
        for x in range(len(historico['Queda livre_{:02}'.format(len(historico))][0])):
            #print('{};{}'.format(historico['Queda livre_{:02}'.format(len(historico))][0][x],historico['Queda livre_{:02}'.format(len
            top.ListaResultados.insert(dadosensaio, "end", text='Ponto {:02}'.format(x),
                                       values=(historico['Queda livre_{:02}'.format(len(historico))][0][x],
                                               historico['Queda livre_{:02}'.format(len(historico))][1][x]))
    else:
        plt.ylabel('Tomada de tempo [unidade]')
        plt.plot(tempos)
        historico['Queda livre_{:02}'.format(len(historico)+1)] = (tempos)
        #top.LblTempos.config(text=historico['Queda livre_{:02}'.format(len(historico))])

    plt.title('Ensaio de Queda livre')
    plt.xlabel('Intervalo [s]')
    print(historico)
    plt.show()
    #top.lbl_Status.config(text='')
```

20200724_EnsaioFisica.py

```
    plt.xlabel('Intervalo [s]')
    print(historico)
    plt.show()
    #top.lbl_Status.config(text='')

def ensaioPlanoInclinado(top):
    global tempos, ensaio, historico
    print('Ensaio de plano inclinado. Pressione o gatilho para iniciar!')
    top.lbl_Status.config(text='Iniciando ensaio de plano inclinado...')
    eletroima.on()
    pygame.init()
    pygame.mixer.music.load('EnsaioPlanoInclinado.mp3')
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    pygame.mixer.music.load("PressioneGatilho01.mp3")
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    top.lbl_Status.config(text='Pressione o gatilho para iniciar...')
    while not gatilho.is_pressed:
        ensaio = 0
    while gatilho.is_pressed:
        ensaio = 0
    tempos.clear()
    eletroima.off()
    ensaio = 1
    while ensaio == 1:
        print('Aguardando...')
        sleep(1.5)
    som.blink(on_time=0.02, off_time=0.02,n=1)
```

20200724_EnsaioFisica.py

```
        print('Aguardando...')
        sleep(1.5)
    som.blink(on_time=0.02, off_time=0.02,n=1)
    resultado = list()
    print(tempos)
    top.lbl_Status.config(text='Fim do ensaio. Apresentado resultados...')
    tempoinicial = tempos[0]
    for x in range(len(tempos)):
        tempos[x] = (tempos[x] - tempoinicial)/1000

    #tempos.insert(0,0.0)
    #tempos.pop()
    print(tempos)
    pygame.mixer.music.load("FimEnsaio.mp3")
    pygame.mixer.music.play(1)
    while pygame.mixer.music.get_busy() == True:
        continue
    print('Final do ensaio de plano inclinado.')
    print('{};{}'.format(top.posSensor01.get(), type(top.posSensor01.get())))
    if len(top.posSensor01.get()) > 0: resultado.append(float(top.posSensor01.get()))
    if len(top.posSensor02.get()) > 0: resultado.append(float(top.posSensor02.get()))
    if len(top.posSensor03.get()) > 0: resultado.append(float(top.posSensor03.get()))
    if len(top.posSensor04.get()) > 0: resultado.append(float(top.posSensor04.get()))
    if len(top.posSensor05.get()) > 0: resultado.append(float(top.posSensor05.get()))
    top.LblTempos.config(text='')

    if len(resultado) == len(tempos):
        plt.ylabel('Distância [cm]')
        plt.plot(tempos, resultado)
        historico['Plano inclinado_{:02}'.format(len(historico)+1)] = (resultado, tempos)
        #top.LblTempos.config(text=historico['Queda livre_{:02}'.format(len(historico))])
```

20200724_EnsaioFisica.py

```
179          plt.plot(tempos, resultado)
180          historico['Plano inclinado_{:02}'.format(len(historico)+1)] = (resultado, tempos)
181          #top.LblTempos.config(text=historico['Queda livre_{:02}'.format(len(historico))])
182          dadosensaio = top.ListaResultados.insert('', 'end', text='PI_{:02}'.format(len(historico)),
183                                                   values=('[cm]','[s]',''))
184          for x in range(len(historico['Plano inclinado_{:02}'.format(len(historico))][0])):
185              #print('{};{}'.format(historico['Queda livre_{:02}'.format(len(historico))][0][x],historico['Queda livre_{:02}'.format(len
186              top.ListaResultados.insert(dadosensaio, "end", text='Ponto {:02}'.format(x),
187                                         values=(historico['Plano inclinado_{:02}'.format(len(historico))][0][x],
188                                                 historico['Plano inclinado_{:02}'.format(len(historico))][1][x]))
189      else:
190          plt.ylabel('Tomada de tempo [unidade]')
191          plt.plot(tempos)
192          historico['Plano inclinado_{:02}'.format(len(historico)+1)] = (tempos)
193          #top.LblTempos.config(text=historico['Queda livre_{:02}'.format(len(historico))])
194
195      plt.title('Ensaio de Plano inclinado')
196      plt.xlabel('Intervalo [s]')
197      print(historico)
198      plt.show()
199      #top.lbl_Status.config(text='')
200
201  def ensaioPendulo(top):
202      global tempos, ensaio, imagem
203
204      print('''Ensaio de Pêndulo. Pressione o gatilho para iniciar o registros de tempo e
205            pressione novamente para terminar os registros de tempo''')
206      top.lbl_Status.config(text='Iniciando ensaio de pêndulo...')
207      pygame.init()
208      pygame.mixer.music.load('EnsaioPendulo.mp3')
209      pygame.mixer.music.play(1)
```

20200724_EnsaioFisica.py

```
210      while pygame.mixer.music.get_busy() == True:
211          continue
212      pygame.mixer.music.load('PressioneGatilho02.mp3')
213      pygame.mixer.music.play(1)
214      while pygame.mixer.music.get_busy() == True:
215          continue
216      top.lbl_Status.config(text='Pressione o gatilho para iniciar...')
217
218      gatilho.wait_for_press()
219      gatilho.wait_for_release()
220      tempos.clear()
221      ensaio = 2
222      top.lbl_Status.config(text='Pressione o gatilho para finalizar.')
223      gatilho.wait_for_press()
224      ensaio = 0
225      resultado = list()
226      for x in range(len(tempos)-1):
227          if (x % 2) == 0:
228              resultado.append((tempos[x+1] - tempos[x])/1000)
229      #tempos.pop()
230      print(resultado)
231
232      pygame.mixer.music.load("FimEnsaio.mp3")
233      pygame.mixer.music.play(1)
234      while pygame.mixer.music.get_busy() == True:
235          continue
236
237      print('Final do ensaio de pêndulo.')
238
239      plt.plot(resultado)
```

20200724_EnsaioFisica.py

```
237      print('Final do ensaio de pêndulo.')
238
239      plt.plot(resultado)
240
241      plt.title('Pendulo Simples')
242      plt.xlabel('Tomada de tempo [unidade]')
243      plt.ylabel('Intervalo [s]')
244      plt.show()
245      top.lbl_Status.config(text='')
246
247  def importarPosicoes(top):
248      try:
249          file1 = open("test.txt","r")
250          param = dict()
251          for x in file1.readlines():
252              y = x.split('=')
253              param[y[0]] = float(y[1].strip())
254          top.posSensor01.delete(0, 'end')
255          top.posSensor02.delete(0, 'end')
256          top.posSensor03.delete(0, 'end')
257          top.posSensor04.delete(0, 'end')
258          top.posSensor05.delete(0, 'end')
259          if 'S1' in param: top.posSensor01.insert(0, float(param['S1']))
260          if 'S2' in param: top.posSensor02.insert(0, float(param['S2']))
261          if 'S3' in param: top.posSensor03.insert(0, float(param['S3']))
262          if 'S4' in param: top.posSensor04.insert(0, float(param['S4']))
263          if 'S5' in param: top.posSensor05.insert(0, float(param['S5']))
264      except:
265          print('Arquivo não carregado')
266
267  def exportarDados(top):
```

20200724_EnsaioFisica.py

```
267 def exportarDados(top):
268     global historico
269     print('Exportar')
270     arquivo = open('/share/EnsaiosFisica/dados_exportados.txt', 'w') #Caminho da armezanagem do arquivo exportado
271     for conjunto in historico:
272         arquivo.write('Ensaio: {}\n'.format(conjunto))
273         arquivo.write('Distâncias [cm];Tempo [s]\n')
274         try:
275             for linha in range(len(historico[conjunto][1])):
276                 if len(historico[conjunto]) == 2:
277                     arquivo.write('{:15.3f};{:9.3f}\n'.format(historico[conjunto][0][linha],historico[conjunto][1][linha]))
278         except:
279             arquivo.write('{}\n'.format(str(historico[conjunto])))
280     arquivo.close()
281
282 def limparDados(top):
283     global historico
284     for registro in top.ListaResultados.get_children():
285         top.ListaResultados.delete(registro)
286     historico.clear()
287     print('Registros removidos')
288
289 def vp_start_gui():
290     '''Starting point when module is the main routine.'''
291     global val, w, root, finalizar, imagem
292     root = tk.Tk()
293     top = janela (root)
294     EnsaioFisica_support.init(root, top)
295     while True:
296         try:
297             root.update_idletasks()
```

20200724_EnsaioFisica.py

```
294     EnsaioFisica_support.init(root, top)
295     while True:
296         try:
297             root.update_idletasks()
298             #top.Label14.config(image=imagem)
299             #root.widget['Label14'].config(text='Teste')
300             root.update()
301             #root.mainloop()
302         except:
303             print('Fim da interface')
304             finalizar = True
305             break
306
307 w = None
308 def create_janela(rt, *args, **kwargs):
309     '''Starting point when module is imported by another module.
310        Correct form of call: 'create_janela(root, *args, **kwargs)' .'''
311     global w, w_win, root
312     #rt = root
313     root = rt
314     w = tk.Toplevel (root)
315     top = janela (w)
316     EnsaioFisica_support.init(w, top, *args, **kwargs)
317     return (w, top)
318
319 def destroy_janela(top):
320     top.destroy()
321
322 class janela:
323     def __init__(self, top=None):
324         '''This class configures and populates the toplevel window.
```

20200724_EnsaioFisica.py

```
322 class janela:
323     def __init__(self, top=None):
324         '''This class configures and populates the toplevel window.
325            top is the toplevel containing window.'''
326         _bgcolor = '#d9d9d9'  # X11 color: 'gray85'
327         _fgcolor = '#000000'  # X11 color: 'black'
328         _compcolor = '#d9d9d9' # X11 color: 'gray85'
329         _ana1color = '#d9d9d9' # X11 color: 'gray85'
330         _ana2color = '#ececec' # Closest X11 color: 'gray92'
331
332         top.geometry("850x600+250+60")
333         top.minsize(120, 1)
334         top.maxsize(2730, 749)
335         top.resizable(1, 1)
336         top.title("PRODUTO EDUCACIONAL - MNPEF - PHOTOGATE")
337         top.configure(background="black", highlightbackground="gray", highlightcolor="white")
338
339         self.lblTextoCabecalho = tk.Label(top)
340         self.lblTextoCabecalho.place(relx=0.05, rely=0.02, height=25, relwidth=1.0, bordermode='ignore')
341         self.lblTextoCabecalho.configure(text='''COLETOR PHOTOGATE''', font='impact 20 bold',
342                                  background="black", foreground="orange")
343
344         self.lblFrame_Opcoes = tk.LabelFrame(top)
345         self.lblFrame_Opcoes.place(relx=0.02, rely=0.07, relheight=0.4, relwidth=0.5)
346         self.lblFrame_Opcoes.configure(text='''Opções de ensaios''',font="impact 20 bold", foreground="orange",
347                                        background="black", relief='groove')
348
349         self.lblFrame_Posicoes = tk.LabelFrame(top)
350         self.lblFrame_Posicoes.place(relx=0.54, rely=0.07, relheight=0.4, relwidth=0.44)
351         self.lblFrame_Posicoes.configure(text='''Posições dos sensores''',font="impact 20 bold", foreground="orange",
352                                        background="black", relief='groove')
```

20200724_EnsaioFisica.py

```
352                                     background="black", relief='groove')
353
354         self.lblFrame_Resultados = tk.LabelFrame(top)
355         self.lblFrame_Resultados.place(relx=0.02, rely=0.47, relheight=0.45, relwidth=0.96)
356         self.lblFrame_Resultados.configure(text='''Resultados''',font="impact 20 bold", foreground="orange",
357                                     background="black", relief='groove')
358
359         self.frm_Status = tk.Frame(top)
360         self.frm_Status.place(relx=0.02, rely=0.93, relheight=0.05, relwidth=0.96)
361         self.frm_Status.configure(borderwidth=2, background="black", relief='groove')
362
363
364         self.posSensor01 = tk.Entry(self.lblFrame_Posicoes)
365         self.posSensor01.place(relx=0.35, rely=0.13, height=20, relwidth=0.42, bordermode='ignore')
366         self.posSensor01.configure(background="white", foreground="#000000",disabledforeground="#a3a3a3",
367                                     font="TkFixedFont", highlightbackground="#d9d9d9", highlightcolor="black",
368                                     insertbackground="black", selectbackground="blue", selectforeground="white")
369
370         self.posSensor02 = tk.Entry(self.lblFrame_Posicoes)
371         self.posSensor02.place(relx=0.35, rely=0.286, height=20, relwidth=0.42, bordermode='ignore')
372         self.posSensor02.configure(background="white", foreground="#000000",disabledforeground="#a3a3a3",
373                                     font="TkFixedFont", highlightbackground="#d9d9d9", highlightcolor="black",
374                                     insertbackground="black", selectbackground="blue", selectforeground="white")
375
376         self.posSensor03 = tk.Entry(self.lblFrame_Posicoes)
377         self.posSensor03.place(relx=0.35, rely=0.457, height=20, relwidth=0.42, bordermode='ignore')
378         self.posSensor03.configure(background="white", foreground="#000000",disabledforeground="#a3a3a3",
379                                     font="TkFixedFont", highlightbackground="#d9d9d9", highlightcolor="black",
380                                     insertbackground="black", selectbackground="blue", selectforeground="white")
381
382         self.posSensor04 = tk.Entry(self.lblFrame Posicoes)
```

20200724_EnsaioFisica.py

```
382         self.posSensor04 = tk.Entry(self.lblFrame_Posicoes)
383         self.posSensor04.place(relx=0.35, rely=0.629, height=20, relwidth=0.42, bordermode='ignore')
384         self.posSensor04.configure(background="white", foreground="#000000",disabledforeground="#a3a3a3",
385                                     font="TkFixedFont", highlightbackground="#d9d9d9", highlightcolor="black",
386                                     insertbackground="black", selectbackground="blue", selectforeground="white")
387
388         self.posSensor05 = tk.Entry(self.lblFrame_Posicoes)
389         self.posSensor05.place(relx=0.35, rely=0.8, height=20, relwidth=0.42, bordermode='ignore')
390         self.posSensor05.configure(background="white", foreground="#000000",disabledforeground="#a3a3a3",
391                                     font="TkFixedFont", highlightbackground="#d9d9d9", highlightcolor="black",
392                                     insertbackground="black", selectbackground="blue", selectforeground="white")
393
394         self.Label1 = tk.Label(self.lblFrame_Posicoes)
395         self.Label1.place(relx=0.05, rely=0.13, height=25, relwidth=0.3, bordermode='ignore')
396         self.Label1.configure(text='''Sensor 01''', font='impact 12 bold',
397                                 background="black", foreground="white")
398
399         self.Label2 = tk.Label(self.lblFrame_Posicoes)
400         self.Label2.place(relx=0.05, rely=0.286, height=25, relwidth=0.3, bordermode='ignore')
401         self.Label2.configure(text='''Sensor 02''', font='impact 12 bold',
402                                 background="black", foreground="white")
403
404         self.Label3 = tk.Label(self.lblFrame_Posicoes)
405         self.Label3.place(relx=0.05, rely=0.457, height=25, relwidth=0.3, bordermode='ignore')
406         self.Label3.configure(text='''Sensor 03''', font='impact 12 bold',
407                                 background="black", foreground="white")
408
409         self.Label4 = tk.Label(self.lblFrame_Posicoes)
410         self.Label4.place(relx=0.05, rely=0.629, height=25, relwidth=0.3, bordermode='ignore')
411         self.Label4.configure(text='''Sensor 04''', font='impact 12 bold',
412                                 background="black", foreground="white")
```

20200724_EnsaioFisica.py

```
412                                 background="black", foreground="white")
413
414         self.Label5 = tk.Label(self.lblFrame_Posicoes)
415         self.Label5.place(relx=0.05, rely=0.8, height=25, relwidth=0.3, bordermode='ignore')
416         self.Label5.configure(text='''Sensor 05''', font='impact 12 bold',
417                                 background="black", foreground="white")
418
419         self.Label6 = tk.Label(self.lblFrame_Posicoes)
420         self.Label6.place(relx=0.8, rely=0.13, height=21, relwidth=0.1, bordermode='ignore')
421         self.Label6.configure(text='''cm''', font='impact 12 bold',
422                                 background="black", foreground="white")
423
424         self.Label7 = tk.Label(self.lblFrame_Posicoes)
425         self.Label7.place(relx=0.8, rely=0.286, height=21, relwidth=0.1, bordermode='ignore')
426         self.Label7.configure(text='''cm''', font='impact 12 bold',
427                                 background="black", foreground="white")
428
429         self.Label8 = tk.Label(self.lblFrame_Posicoes)
430         self.Label8.place(relx=0.8, rely=0.457, height=21, relwidth=0.1, bordermode='ignore')
431         self.Label8.configure(text='''cm''', font='impact 12 bold',
432                                 background="black", foreground="white")
433
434         self.Label9 = tk.Label(self.lblFrame_Posicoes)
435         self.Label9.place(relx=0.8, rely=0.629, height=21, relwidth=0.1, bordermode='ignore')
436         self.Label9.configure(text='''cm''', font='impact 12 bold',
437                                 background="black", foreground="white")
438
439         self.Label10 = tk.Label(self.lblFrame_Posicoes)
440         self.Label10.place(relx=0.8, rely=0.8, height=21, relwidth=0.1, bordermode='ignore')
441         self.Label10.configure(text='''cm''', font='impact 12 bold',
442                                 background="black", foreground="white")
```

20200724_EnsaioFisica.py

```
                                background="black", foreground="white")

        self.cmdOpcao01 = tk.Button(self.lblFrame_Opcoes)
        self.cmdOpcao01.place(relx=0.038, rely=0.171, height=24, relwidth=0.33, bordermode='ignore')
        self.cmdOpcao01.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
                                  highlightcolor="black", pady="0", command=lambda: ensaioQuedaLivre(self))
        self.cmdOpcao01.configure(text='''Opção 01''', font='impact 12 bold')

        self.cmdOpcao02 = tk.Button(self.lblFrame_Opcoes)
        self.cmdOpcao02.place(relx=0.038, rely=0.4, height=24, relwidth=0.33, bordermode='ignore')
        self.cmdOpcao02.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
                                  highlightcolor="black", pady="0", command=lambda:ensaioPendulo(self))
        self.cmdOpcao02.configure(text='''Opção 02''', font='impact 12 bold')

        self.cmdOpcao03 = tk.Button(self.lblFrame_Opcoes)
        self.cmdOpcao03.place(relx=0.038, rely=0.629, height=24, relwidth=0.33, bordermode='ignore')
        self.cmdOpcao03.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
                                  highlightcolor="black", pady="0", command=lambda: ensaioPlanoInclinado(self))
        self.cmdOpcao03.configure(text='''Opção 03''', font='impact 12 bold')

        self.Label11 = tk.Label(self.lblFrame_Opcoes)
        self.Label11.place(relx=0.423, rely=0.171, height=21, relwidth=0.55, bordermode='ignore')
        self.Label11.configure(text='''Ensaio de queda livre''', font='impact 12 bold',
                               background="black", foreground="white", justify = 'left')

        self.Label12 = tk.Label(self.lblFrame_Opcoes)
        self.Label12.place(relx=0.423, rely=0.4, height=21, relwidth=0.55, bordermode='ignore')
        self.Label12.configure(text='''Ensaio de pêndulo''', font='impact 12 bold',
```

20200724_EnsaioFisica.py

```
        self.Label12.configure(text='''Ensaio de pêndulo''', font='impact 12 bold',
                               background="black", foreground="white", justify = 'left')

        self.Label13 = tk.Label(self.lblFrame_Opcoes)
        self.Label13.place(relx=0.423, rely=0.629, height=21, relwidth=0.55, bordermode='ignore')
        self.Label13.configure(text='''Ensaio de plano inclinado''', font='impact 12 bold',
                               background="black", foreground="white", justify = 'left')

#         self.cmdFechar= tk.Button(top)
#         self.cmdFechar.place(relx=0.80, rely=0.94, height=24, relwidth=0.15, bordermode='ignore')
#         self.cmdFechar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
#                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
#                                  highlightcolor="black", pady="0", command=destroy_janela)
#         self.cmdFechar.configure(text='''Fechar''', font='impact 12 bold')

        self.cmdImportar= tk.Button(self.lblFrame_Posicoes)
        self.cmdImportar.place(relx=0.72, rely=0.9, height=25, relwidth=0.25, bordermode='ignore')
        self.cmdImportar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
                                   disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
                                   highlightcolor="black", pady="0", command=lambda:importarPosicoes(self))
        self.cmdImportar.configure(text='''Importar''', font='impact 12 bold')


        self.LblTempos = tk.Label(self.lblFrame_Resultados)
        self.LblTempos.grid(row = 0, column = 0)
        self.LblTempos.configure(text=None, font='impact 12 bold',
                                 background="black", foreground="white", justify = 'left')

        self.ListaResultados = ttk.Treeview(self.lblFrame_Resultados)
        self.ListaResultados['columns'] = ('Distância','Tempo')
        self.ListaResultados.column('#0', width=180, minwidth=180, stretch=tk.NO)
```

20200724_EnsaioFisica.py

```
#         self.cmdFechar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
#                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
#                                  highlightcolor="black", pady="0", command=destroy_janela)
#         self.cmdFechar.configure(text='''Fechar''', font='impact 12 bold')

        self.cmdImportar= tk.Button(self.lblFrame_Posicoes)
        self.cmdImportar.place(relx=0.72, rely=0.9, height=25, relwidth=0.25, bordermode='ignore')
        self.cmdImportar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
                                   disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
                                   highlightcolor="black", pady="0", command=lambda:importarPosicoes(self))
        self.cmdImportar.configure(text='''Importar''', font='impact 12 bold')


        self.LblTempos = tk.Label(self.lblFrame_Resultados)
        self.LblTempos.grid(row = 0, column = 0)
        self.LblTempos.configure(text=None, font='impact 12 bold',
                                 background="black", foreground="white", justify = 'left')

        self.ListaResultados = ttk.Treeview(self.lblFrame_Resultados)
        self.ListaResultados['columns'] = ('Distância','Tempo')
        self.ListaResultados.column('#0', width=180, minwidth=180, stretch=tk.NO)
        self.ListaResultados.column('Distância', width=100, minwidth=100, stretch=tk.NO)
        self.ListaResultados.column('Tempo', width=100, minwidth=100, stretch=tk.NO)
        self.ListaResultados.heading('#0', text="Teste", anchor = tk.W)
        self.ListaResultados.heading('Distância', text="Distância", anchor = tk.W)
        self.ListaResultados.heading('Tempo', text="Tempo", anchor = tk.W)
        self.ListaResultados.place(relx=0.01, rely=0.01, relheight=0.98, relwidth=0.44)

        self.cmdLimpar = tk.Button(self.lblFrame_Resultados)
        self.cmdLimpar.place(relx=0.46, rely=0.75, height=24, relwidth=0.15, bordermode='ignore')
        self.cmdLimpar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
```

20200724_EnsaioFisica.py

```
511         self.cmdLimpar.place(relx=0.46, rely=0.75, height=24, relwidth=0.15, bordermode='ignore')
512         self.cmdLimpar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
513                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
514                                  highlightcolor="black", pady="0", command=lambda: limparDados(self))
515         self.cmdLimpar.configure(text='''Limpar''', font='impact 12 bold')
516 
517         self.cmdExportar = tk.Button(self.lblFrame_Resultados)
518         self.cmdExportar.place(relx=0.46, rely=0.85, height=24, relwidth=0.15, bordermode='ignore')
519         self.cmdExportar.configure(activebackground="#ececec", activeforeground="#000000", background="#d9d9d9",
520                                  disabledforeground="#a3a3a3", foreground="#000000", highlightbackground="#d9d9d9",
521                                  highlightcolor="black", pady="0", command=lambda: exportarDados(self))
522         self.cmdExportar.configure(text='''Exportar''', font='impact 12 bold')
523 
524         self.lbl_Status = tk.Label(self.frm_Status)
525         self.lbl_Status.place(relx=0.01, rely=0.15, height=21, relwidth=0.55, bordermode='ignore')
526         self.lbl_Status.configure(text='''Teste''', font='impact 12 bold',
527                                   background="black", foreground="black", anchor = 'w')
528 
529 
530 ############PROGRAMA PRINCIPAL#########################################
531 ensaio = 0 # 0 = Nenhum ensaio em andamento
532            # 1 = Ensaio de Queda livre
533            # 2 = Ensaio de Pêndulo
534 tempos = list()
535 finalizar = False
536 historico = dict()
537 thread = threading.Thread(target=vp_start_gui)
538 thread.start()
539 opcoes = ['Selecione_Opcao01.mp3',
540           'Selecione_Opcao02.mp3',
541           'Selecione_Opcao03.mp3']
```

20200724_EnsaioFisica.py

```
522         self.cmdExportar.configure(text='''Exportar''', font='impact 12 bold')
523 
524         self.lbl_Status = tk.Label(self.frm_Status)
525         self.lbl_Status.place(relx=0.01, rely=0.15, height=21, relwidth=0.55, bordermode='ignore')
526         self.lbl_Status.configure(text='''Teste''', font='impact 12 bold',
527                                   background="black", foreground="black", anchor = 'w')
528 
529 
530 ############PROGRAMA PRINCIPAL#########################################
531 ensaio = 0 # 0 = Nenhum ensaio em andamento
532            # 1 = Ensaio de Queda livre
533            # 2 = Ensaio de Pêndulo
534 tempos = list()
535 finalizar = False
536 historico = dict()
537 thread = threading.Thread(target=vp_start_gui)
538 thread.start()
539 opcoes = ['Selecione_Opcao01.mp3',
540           'Selecione_Opcao02.mp3',
541           'Selecione_Opcao03.mp3']
542 #plt.plot(3)
543 # for x in opcoes:
544 #     pygame.init()
545 #     pygame.mixer.music.load(x)
546 #     pygame.mixer.music.play(1)
547 #     while pygame.mixer.music.get_busy() == True:
548 #         continue
549 
550 while not finalizar:
551     dummy = time()
552 print('Agora acabou...')
```