Ministério da Educação – MEC
Secretaria de Educação Superior – SESu
Programa de Educação Tutorial – PET

Universidade Federal Fluminense – UFF
Escola de Engenharia – TCE
Grupo PET do Curso de Eng. de Telecomunicações – PET-Tele

Apostilas PET-Tele

Introdução ao Raspberry Pi

(Versão: A2021M01D29)

Autores: Raphael Miranda
João Luiz de Amorim Pereira Neto

Tutor: Alexandre Santos de la Vega

Niterói – RJ
Janeiro / 2021

Prefácio

Tendo em vista as diretrizes do MEC em pesquisa, ensino e extensão, o Programa de Educação Tutorial (PET) do curso de Engenharia de Telecomunicações da Universidade Federal Fluminense (UFF) desenvolveu em parceria com o Laboratório de Drenagem, Irrigação e Saneamento Ambiental (LaDISan) [3], um projeto de sensoriamento e atuação remota em um sistema físico utilizando a plataforma Raspberry Pi [5]. Com o desenvolvimento do projeto o grupo adquiriu experiência e conhecimento com a plataforma Raspberry Pi. Assim, surge a ideia da presente apostila como uma ramificação do projeto de atuação remota. O intuito deste trabalho é auxiliar os estudantes no aprendizado de temas importantes para sua formação, que não estão diretamente presentes no currículo, e, servir de material didático para cursos de capacitação ministrados pelos bolsistas do Programa.

Essa apostila, tem como objetivo servir de base para o leitor iniciar seus estudos com a plataforma Raspberry Pi [11], o documento foi utilizado no curso de Raspberry Pi ofertado na Semana de Telecomunicações (SeTeL 2019) ocorrido na Universidade Federal Fluminense e organizado pelo grupo PET-Tele [9]. Espera-se um leitor com algum conhecimento prévio de informática e que já tenha estudado lógica de programação. Além disso, é interessante, para melhor aprendizado, que o leitor esteja com alguma versão do dispositivo em mãos. Conforme ocorra a leitura, comandos e procedimentos devem ser testados no dispositivo.

Não é requisito ter conhecimento prévio específico em Linux ou Python. O material busca contemplar aspectos básicos de sintaxe da linguagem Python e de comandos nativos do sistema Linux aplicados ao dispositivo. Para uma melhor compreensão da linguagem e dos comandos mencionados acima, o grupo PET-Tele disponibiliza apostilas, que podem ser encontradas na página do grupo por meio do seguinte URL:

`www.telecom.uff.br/pet/petws/index.php?pagina=downloads/apostilas .`

Sumário

Capítulo 1

Introdução

Raspberry Pi é uma série de computadores de placa única com tamanho reduzido do tipo *System on Chip* (SoC) [15]. A partir dessa placa são conectados, monitores, teclados, mouses e demais periféricos. Seu desenvolvimento ocorreu no Reino Unido pela Raspberry Pi Foundation [11] e tem como principal missão a promoção do ensino em ciência da Computação em escolas, inclusão tecnológica e empoderamento social [14].

1.1 Materiais necessários para utilização

1. Para atuação no ambiente virtual
   (a) Fonte DC 5V | 2.5A e cabo Micro-USB (pode ser Carregador de *smartphone*)
   (b) Cartão SD Mínimo 8-GB (essencial)
   (c) Teclado & Mouse (essencial)
   (d) Cabo HDMI ou Cabo VGA com conversor HDMI | VGA (essencial)
   (e) Cabo Ethernet ou adaptador Wi-Fi (opcional, para acesso à internet)
   (f) Extensor de portas USB (opcional)
   (g) Fones de ouvido (opcional)
2. Para atuação no ambiente físico
   (a) *Protoboard* (essencial para prática ao final da apostila)
   (b) 2 Jumpers Macho - Fêmea (essencial para prática ao final da apostila)
   (c) 1 Resistor 220Ω e 1 LED (essencial para prática ao final da apostila)
   (d) Sensores e atuadores de acordo com suas necessidades

1.2 Instalação e configuração do sistema operacional

O primeiro passo para começar a operação é fazer a instalação do sistema operacional (SO). Para iniciantes a Raspberry Pi Foundation, recomenda a utilização do sistema próprio, o Raspberry Pi OS, também chamado de Raspbian. O Raspbian é um sistema Linux baseado em Debian [13]. Esse procedimento de instalação é chamado de gravação *flash* do sistema operacional no cartão de memória, ou em jargão, *flash* do SO.

Os seguintes requisitos precisam ser contemplados para a instalação correta do sistema operacional; um cartão de memória do tipo SD-Card ou Micro-SD, com capacidade de armazenamento de no mínimo 8GB e um outro computador operando Linux, Windows ou MAC OS com entrada para SD-Card ou Micro-SD. A depender do modelo de Raspberry Pi e computador será preciso um adaptador de encaixe para adaptar as entradas Micro-SD e SD-Card que possuem tamanho diferentes. Certifique-se de que todas as conexões (computador-cartão e cartão-Raspberry Pi) são compatíveis.

A formatação do cartão de memória pode ser feita utilizando uma gama de *softwares*. O curso vai utilizar o programa chamado Etcher por sua facilidade, interface intuitiva e disponibilidade para Linux, Windows e MAC.

Para realizar a instalação, deve-se inserir o SD-Card no devido leitor do computador rodando Linux, Windows ou MAC. Em seguida, realize o *download* do sistema operacional do Raspberry Pi por meio do seguinte URL:

`https://www.raspberrypi.org/downloads/raspberry-pi-os/ .`

O passo seguinte é fazer o *download* e instalação do programa Etcher compatível com seu sistema operacional. O mesmo pode ser encontrado por meio do seguinte URL:

`https://www.balena.io/etcher/ .`

Ao executar o programa siga as instruções da interface para carregar o sistema operacional Raspbian no programa Etcher, movimentos *drag and drop* de arquivos são permitidos, basta arrastar os arquivos até seu local de destino na interface. A Figura 1.1 mostra a ordem dos processos até a conclusão da gravação.

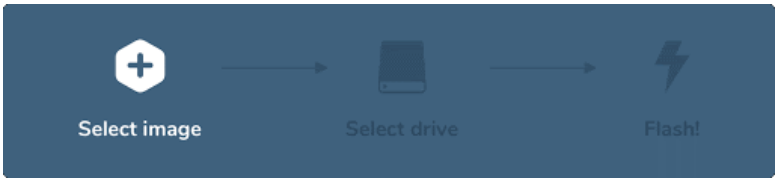


Figura 1.1: Etcher Software passo a passo.

Terminado o procedimento acima retire com segurança o cartão de memória do dispositivo e está pronto para uso no Raspberry Pi. A partir de agora se faz necessário a conexão de mouse, teclado e monitor para configuração da plataforma. A Figura 1.2 mostra os locais de conexão dos periféricos citados no modelo de dispositivo que pertence ao grupo PET-Tele. Pode existir a possibilidade do seu monitor não ter saída HDMI, então poderá ser utilizado um adaptador HDMI-VGA, HDMI-DVI, HDMI-DisplayPort. Observe atentamente se as respectivas entradas são compatíveis antes de forçar um acoplamento.

Ao ligar o dispositivo na rede elétrica observe também se sua fonte de tensão DC fornece energia suficiente para o dispositivo, normalmente fontes com corrente nominal abaixo de 1A não possuem potência suficiente para ligar o dispositivo. Mesmo assim caso sua fonte ligue o dispositivo e não seja a recomendada a placa irá detectar e o aviso *undervoltage* será mostrado no canto superior direito da tela inicial após inicialização da interface. Tenha em mente que utilizar o dispositivo com esse aviso pode ocorrer desligamento imediato ou repentino ao mínimo de estresse no processador ou adição de um dispositivo extra. Após essas observações é possível começar a utilização do Raspberry Pi. Além dos periféricos básicos citados acima, existem outros que podem ser utilizados para obter novas funcionalidades, como por exemplo, um adaptador Wi-Fi ou cabo Ethernet para acesso à rede, extensor USB ou caixas de som.

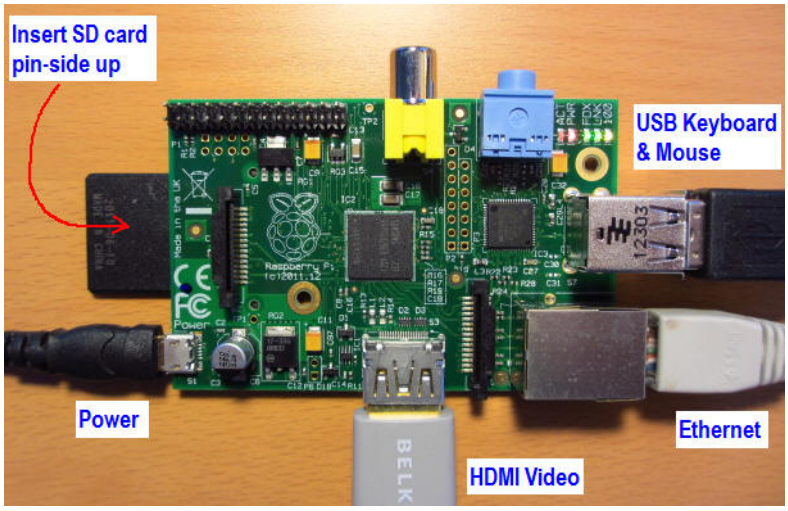


Figura 1.2: Foto do Raspberry Pi e suas conexões.

Como o sistema operacional já está devidamente gravado no cartão de memória podemos iniciar o dispositivo. Em seu estado padrão de fábrica o usuário padrão é definido como “pi”, e a senha padrão “raspberry”. É recomendado fazer alteração dessas credenciais no primeiro uso para impedir invasões quando conectado em uma rede. O processo de inicialização é automático e ocorre assim que o dispositivo é ligado na tomada elétrica. Após a inicialização, se o sistema operacional tiver sido instalado corretamente uma interface semelhante a representada na Figura 1.3 deverá surgir. Pode ocorrer alterações devido a versão do sistema utilizado.

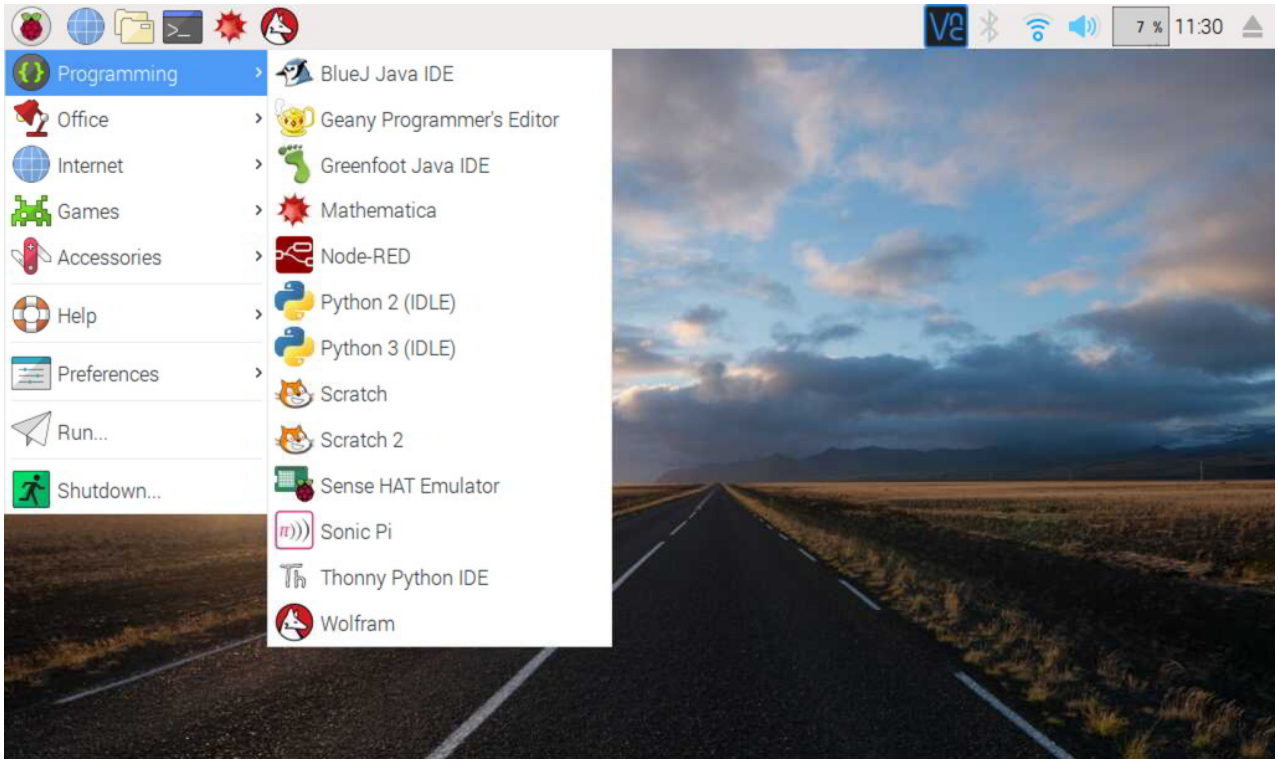


Figura 1.3: Tela inicial Raspberry Pi.

Capítulo 2

Especificações técnicas

2.1 Raspberry Pi Rev. 2

Uma das características marcantes do Raspberry Pi é a facilidade de reprogramar seus terminais para conexão com outros dispositivos. Esses terminais são denominados de pinos de entradas e saídas de propósito geral conhecidos pelo acrônimo GPIO (*General Purprose Input/Output*). O PET-Tele atualmente utiliza o modelo de Raspberry Pi "Revision 2", que conta com 17 desses pinos de entradas e saídas. A Tabela 2.1, apresenta os detalhes técnicos do modelo "Raspberry Pi Revision 2 - Element 14" que é o modelo pertencente ao Grupo PET-Tele.

Neste modelo existem 26 pinos dentre eles, 17 podem ser usados como entradas ou saídas de propósito geral, 5 como terminais comuns (GND), 2 como fontes de tensão +5V e 2 como fontes de tensão +3.3V. A Figura 2.1 pertencente à página Pinballsp [1] e a Tabela 2.2 mostram em detalhes características de cada um dos pinos. Na Figura 2.1 alguns pinos, que estão marcados com um retângulo vermelho, mostram os tipos de comunicações suportadas.

Chip	Broadcom BCM2835 SoC Full HD processador de aplicações Multimídia
CPU	700 Mhz ARM1176JZ-F Processador de Baixa Potência de aplicações
GPU	Dois Núcleos, VideoCore IV, Co-Processador de Multimídia
Memória	512MB SDRAM
Ethernet	Onboard 10/100 Ethernet com conector RJ-45
USB 2.0	Dois Conectores USB 2.0
Saída de Vídeo	HDMI e Composição RCA (PAL e NTSC)
Sáida de Áudio	Conector 3.5 mm ou HDMI
Armazenamento	SD, MMC, SDIO Card Slot
Dimensões	8.6 cm X 5.4 cm X 1.7 cm

Tabela 2.1: Especificações "Raspberry Pi Revision 2 - Element 14".

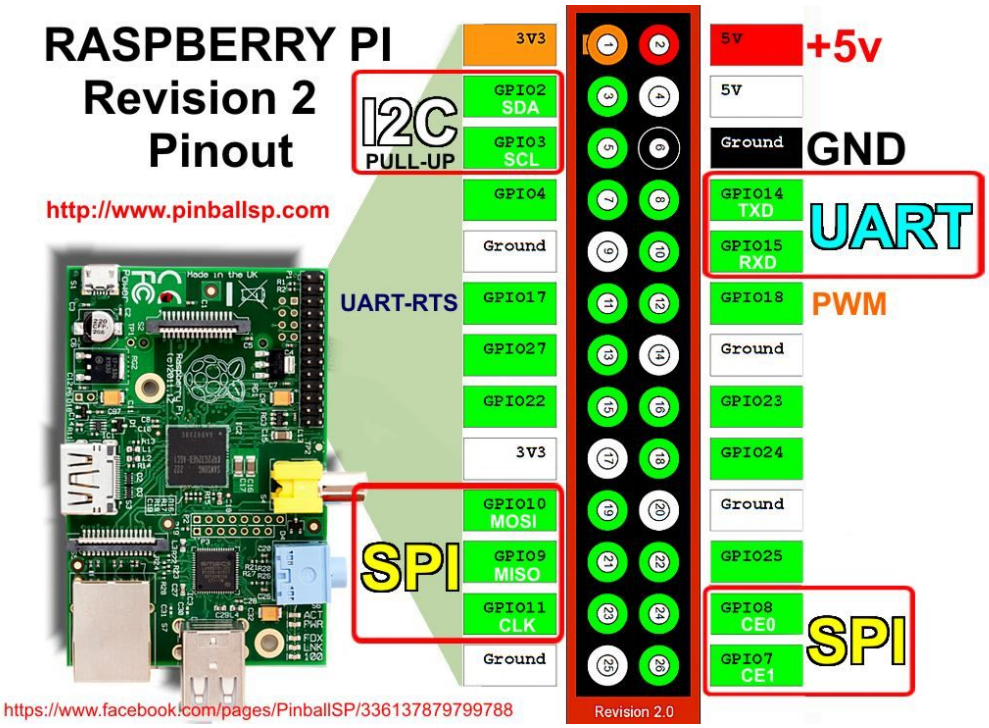


Figura 2.1: Descrição dos pinos do Raspberry |*Pinout*.

N	Descrição	N	Descrição
1	Saída de +3.3V	14	Terminal Comum GND
2	Saída de +5V	15	GPIO22
3	GPIO2 | Pull-Up | I2C-SDA	16	GPIO23
4	Saída de +5V	17	3V3
5	GPIO3 | Pull-Up | I2C-SCE	18	GPIO24
6	Terminal Comum GND	19	GPIO10 | SPI | MOSI
7	GPIO4	20	Terminal Comum GND
8	GPIO14 | UART | TXD	21	GPIO9 | SPI | MISO
9	Terminal Comum GND	22	GPIO25
10	GPIO15 | UART | RXD	23	GPIO11 | SPI | CLK
11	GPIO17 | UART-RTS	24	GPIO8 | SPI | CE0
12	GPIO18 | PWM	25	Terminal Comum GND
13	GPIO27	26	GPIO7 | SPI | CE1

Tabela 2.2: Descrição dos Pinos.

2.2 Resistores de Pull-Up/Pull-Down

O Raspberry Pi possui diversos pinos que podem ser utilizados como entradas ou saídas. Esses pinos normalmente são utilizados para conectar sensores ou atuadores. Uma característica dos modelos de Raspberry Pi e normalmente presentes em microcontroladores são os resistores de *Pull-Up* e *Pull-Down*. Esses resistores são importantes para alcançar uma definição do estado lógico do pino quando se encontram em estado flutuante. Por exemplo, um pino definido como entrada que está em circuito aberto, antes de se conectar a algum componente ocorre o fenômeno nó flutuante. Esse fenômeno é a incapacidade da controladora em definir o nível lógico de tensão do pino, que em eletrônica digital é codificado por *High* ou *Low* ou pela lógica binária "1" ou "0".

Para solucionar o problema do estado flutuante, a seguinte abordagem é praticada. Coloca-se um resistor com resistência elevada em série com esse pino em duas configurações distintas como mostrado na Figura 2.2 e Figura 2.3.

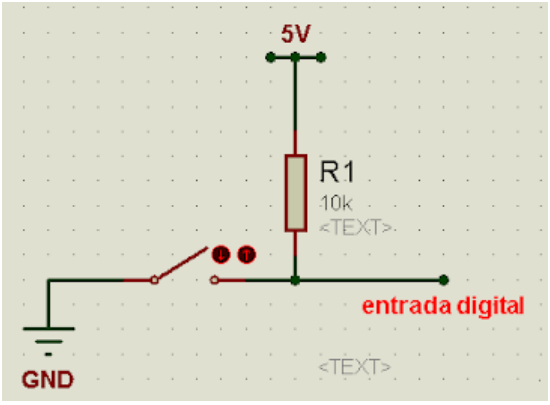


Figura 2.2: Circuito de Pull Up.

O circuito acima é chamado de circuito de *Pull-Up* pois ele "empurra" o pino de entrada para o estado *High* por padrão. Se colocarmos um multímetro com a chave aberta teremos o valor de tensão muito próximo de +5V. Mudando o estado da chave, temos o valor 0V diretamente aplicado na entrada digital.

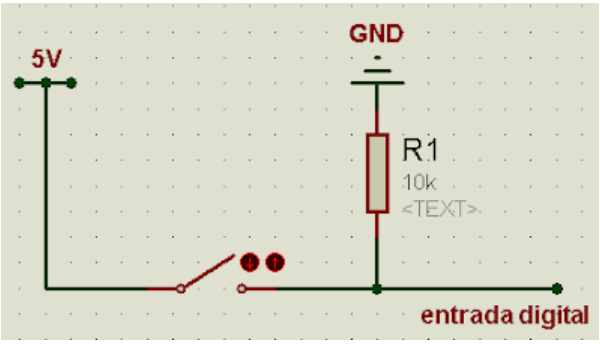


Figura 2.3: Circuito de Pull Down.

Já este outro circuito da Figura 2.3 é chamado de *Pull-Down* pois ele empurra o pino de entrada para o estado *Low* por padrão. Se colocarmos um com a chave aberta, teremos o valor de leitura de 0V, enquanto que se colocarmos o instrumento medidor com a chave fechada teremos +5V.

2.3 Tipos de comunicações *On board*

Uma comunicação compreende em uma série de etapas e premissas para que realmente possa ser dito que existe uma comunicação. A princípio é necessário que existam pelo menos dois interlocutores e uma mensagem. Esses interlocutores precisam ter regras previamente estabelecidas que sejam mutualmente entendíveis. Satisfeitas as condições, podemos abstrair e expandir esse conceito para diversas esferas. O exemplo mais palpável é uma comunicação entre duas pessoas (interlocutores) que trocam uma conversa (mensagem com informação) em uma língua de conhecimento dos dois (regras mutualmente conhecidas). O outro exemplo é quando esses dois interlocutores são dispositivos eletrônicos, e então podemos dizer que é uma comunicação do tipo D2D (*Device-to-Device*, ou Dispositivo a Dispositivo). Quando referimos a uma comunicação D2D podemos assumir que principalmente e não exclusivamente a "língua" utilizada são sinais elétricos e esses sinais podem ser representados pela notação binária. Como exemplo imagine um celular que se comunica com uma estação rádio base sem que um ser humano participe do evento, podemos caracterizar esse exemplo como uma comunicação D2D. O Raspberry Pi possui assim como qualquer outro equipamento eletrônico seus tipos e formatos de comunicação. Em seguida, vão ser apresentados alguns padrões que a plataforma implementou e disponibilizou ao usuário, para criar e modelar o fluxo de informações.

2.3.1 Interface Periférica Serial - SPI

Interface Periférica Serial (*Serial Peripheral Interface*) é uma forma de comunicação do tipo Serial, ou seja, os bits são enviados de maneira sequencial. A Figura 2.4 é uma representação de como se dá esse envio sequencial. Outra característica é ser do tipo síncrona, depende de uma entidade para fazer o controle temporal do fluxo de informações, que no caso é o "Clock" um pulso de intervalo predeterminado. Uma de suas regras de Comunicação é a ótica mestre-escravo. Normalmente é utilizada para um controlador se comunicar com diversos dispositivos e enviar/receber informações. Podemos imbuir que na ótica mestre-escravo o mestre possui privilégios administrativos. Assim, um escravo não deve iniciar nenhum tipo de comunicação no meio físico enquanto não tiver sido requisitado pelo mestre.

2.3.2 Transmissor e Receptor Assíncrono Universal - UART

Transmissor e Receptor Assíncrono Universal (*Universal Asynchronous Receiver/Transmitter*) é um sistema de comunicação misto. Tomado em contraponto ao SPI em que a transmissão totalmente serial as UARTs possuem a finalidade de converter dados paralelos para a forma serial e vice versa, as UARTs estão principalmente presentes nos modems mas também são encontradas em muitos outros equipamentos. No diagrama de funcionamento apresentado na Figura 2.5, por simplificações de representação pode até parecer simples, converter os dados simultâneos que chegam nas entradas paralelas para transmissão sequencial por uma única linha de dados. No entanto, precisamos levar em conta que essa transmissão deve ser feita de modo seguro, com um controle de paridade para garantir a integridade dos dados e formas de sinalização do lado transmissor para o lado receptor saber onde começa e onde termina uma transmissão, além do que existe o processo reconversão para a forma de apresentação paralela na saída do receptor.

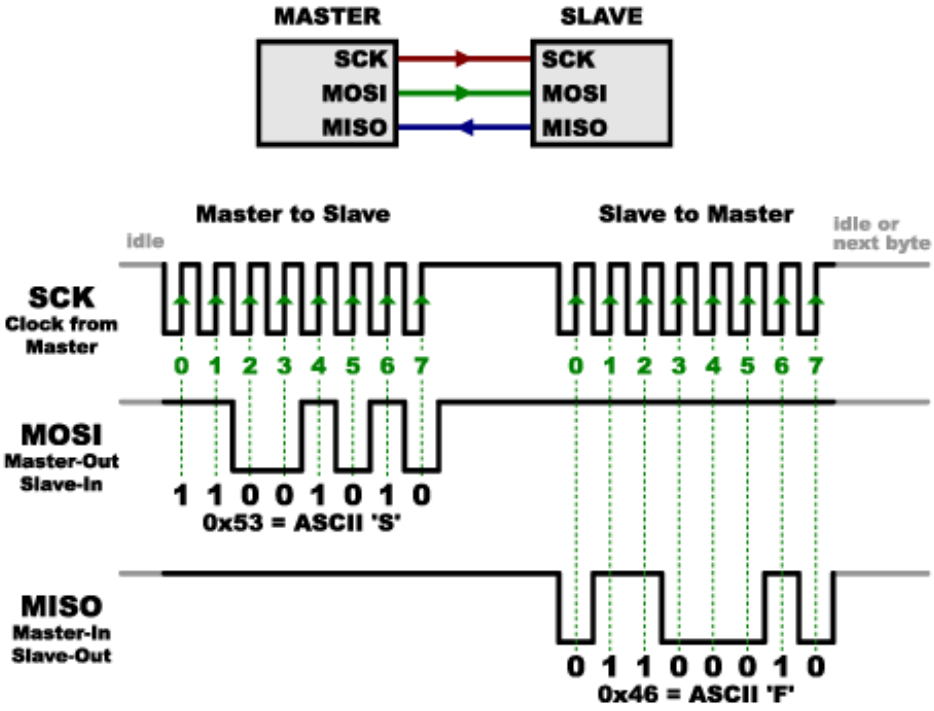


Figura 2.4: Diagrama SPI.

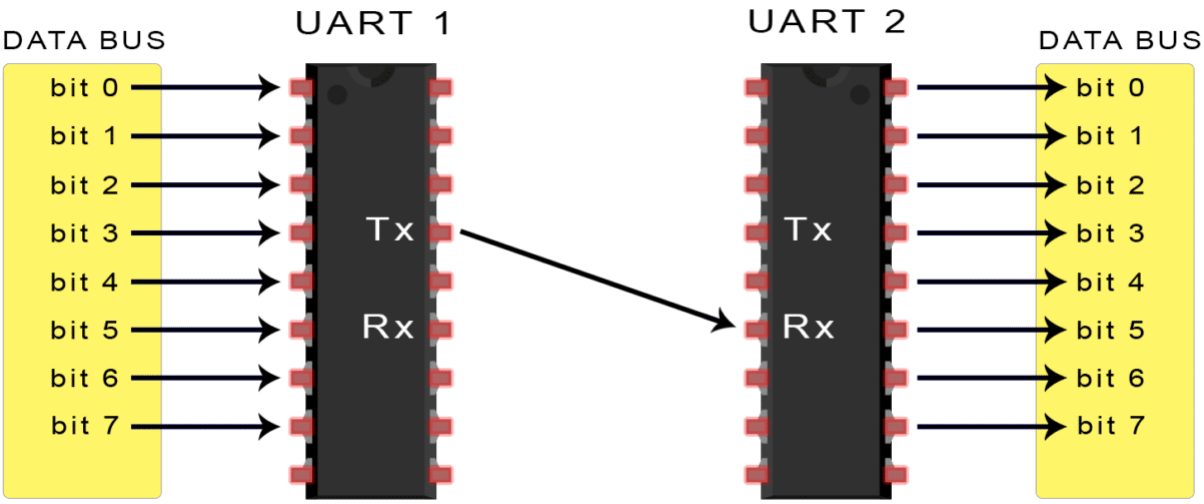


Figura 2.5: Diagrama da comunicação por UART.

2.4 Técnica de Modulação por Largura de Pulso PWM

A técnica *Pulse Width Modulation* (PWM), ou modulação por largura de pulso é amplamente utilizada em microcontroladores quando há a necessidade de controle por tensão média. Normalmente microcontroladores só trabalham com sinais digitais, ou seja, limita-se a apenas dois estados, ligado ou desligado. Alguns projetos de eletrônica podem exigir funções de controle mais complexas, por exemplo, obter uma variação de luminosidade linear em LEDs, obter uma variação de potência em motores e atuadores, obter 40% da potência de um componente, etc. Essas possibilidades são de difícil obtenção em microcontroladores sem o uso de técnicas particulares. Para isso existe a técnica de modulação PWM, nesta técnica, existem um pulso de largura variável (entre um valor mínimo e um valor máximo) e uma cadência de repetição deste pulso, definida por um período ou uma frequência. Um sinal PWM pode ser parametrizado em função da frequência da onda retangular e do ciclo de trabalho dessa onda. O ciclo de trabalho (*duty cycle*) é um valor percentual da duração do pulso em nível alto em relação à duração de um período. A Figura 2.6 apresenta, de forma gráfica, diferentes ciclos de trabalho de um sinal PWM. Para utilizar a técnica PWM no Raspberry Pi, os processos adotados são

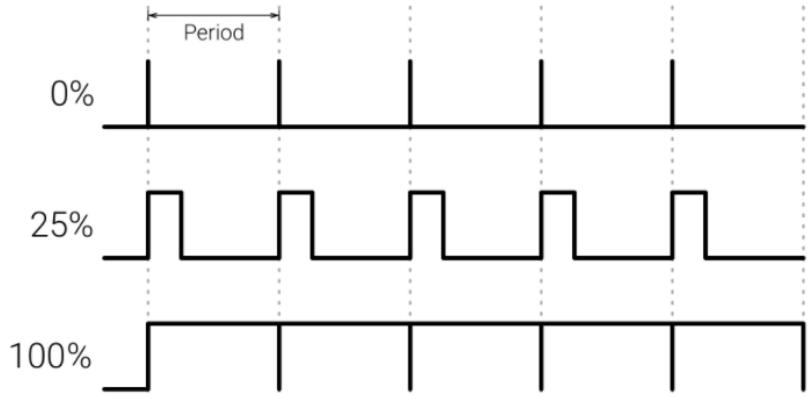


Figura 2.6: Diferentes ciclos de trabalho de um sinal PWM.

semelhantes aos aplicados ao Arduino. A diferença está em algumas especificações da placa e na linguagem utilizada no Raspberry Pi, que trabalha nativamente com Python [10], enquanto o Arduino trabalha com sua linguagem própria. No Arduino modelo UNO a frequência PWM é 976 Hz. A frequência de PWM presentes em placas Raspberry Pi de modelos mais recentes pode ser um valor escolhido de 10Hz a 8 kHz. Além disso, existem bibliotecas da linguagem Python para utilização no controle do servomotor. As duas mais utilizadas são: a PIGPIO [4] e a RPI.GPIO [2].

Capítulo 3

Sistema operacional Linux

O grupo PET-Tele possui uma apostila de Linux [7], que tem por objetivo servir de principal meio de aprendizado para o leitor desta apostila. Os assuntos deste capítulo vão se limitar a um guia-rápido das principais funções do Linux e direcionar para implementações envolvendo o Raspberry Pi.

3.1 História

Diferentemente do que se é levado a pensar, o Linux é uma implementação independente do sistema operacional UNIX. Linux é um termo popularmente empregado para se referir a sistemas operacionais que utilizam o *kernel* Linux [17]. O Linux propriamente dito é um *kernel* ("núcleo"), não um sistema operacional completo.

Sistemas completos construídos em torno do kernel do Linux usam o sistema GNU que oferece um shell, utilitários, bibliotecas, compiladores e ferramentas, bem como outros programas como os editores de texto.

O desenvolvimento do Linux iniciou a partir de um projeto pessoal de um estudante da Universidade de Helsinki, na Finlândia, chamado Linus Torvalds. Ele pretendia criar um sistema operacional mais sofisticado do que o Minix, um UNIX relativamente simples cujo código fonte ele tinha disponível.

O Linux obedece ao padrão estabelecido pelo governo norte americano, POSIX. O POSIX é o padrão da API (*Application Programming Interface*) UNIX, referências para desenvolvedores da família UNIX-like. Desde a apresentação do Linux em 5 de outubro de 1991, por Linus Torvalds, um grande número de pessoas envolvidas com programação começou a desenvolver o Linux. O nome Linux deriva da junção Linus + UNIX = Linux.

3.2 Abertura de sessão, usuários, acesso e proteção

Para usar o Linux é preciso em primeiro lugar, que o usuário digite seu nome e sua senha, que são lidos e verificados pelo programa *login*. No UNIX um arquivo de senha é usado para guardar informações possuindo uma linha para cada usuário, contendo sua identificação alfabética e numérica, sua senha criptografada, seu diretório *home*, além de outras informações. Quando o usuário se identifica, o programa *login* criptografa a senha que acabou de ser lida do terminal e a compara com a senha do arquivo de senhas para dar permissão ao usuário.

O princípio da segurança do sistema de arquivo UNIX está baseado em usuários, grupos e outros usuários. Usuário é a pessoa que criou o arquivo. Grupo é uma categoria que reúne vários usuários. Cada usuário pode fazer parte de um ou mais grupos, que permitem acesso a

arquivos que pertencem ao grupo correspondente. Outros é a categoria de usuários que não se encaixam como donos ou não pertencem aos grupos do arquivo. As permissões de acesso para donos, grupos e outros usuários são independentes uma das outras, permitindo assim um nível de acesso diferenciado.

A conta *root* é também chamada de super usuário. Este é um *login* que não possui restrições de segurança. A conta *root* somente deve ser usada para fazer a administração do sistema. Utilize a conta de usuário normal ao invés da conta *root* para operar seu sistema. Uma razão para evitar usar privilégios *root* é devido à facilidade de se cometer danos irreparáveis ao sistema.

3.3 Sistema de arquivos

Os arquivos no sistema UNIX são organizados em diretórios ou listagens de arquivos. Cada diretório poderá conter um subdiretório, originando assim uma lista hierárquica. Os diretórios são organizados na estrutura de árvore monolítica. Uma aplicação monolítica é autônoma e independente de outras aplicações [16]. O mais alto dos diretórios é chamado de diretório raiz. Esse diretório é determinado pelo símbolo "/". Em sistemas UNIX, todo espaço em disco disponível é combinado em uma única árvore de diretório abaixo do "/".

A adição de nomes e "/" especificam novos subdiretórios. Quando os usuários abrem uma sessão, são trazidos no diretório pessoal chamado de seu diretório de entrada. Esse diretório é tipificado com um til "∼".

Ao abrir um terminal Linux, por padrão,
você irá se deparar com a seguinte linha:

```
Nome_Usuario@Nome_Computador:~$ .
```

3.4 Permissões para acesso a arquivos

O sistema diferencia três classes de usuários. Primeiro, todo arquivo possui um dono, um proprietário, designado no sistema por *user*. O proprietário tem controle total sobre a restrição ou permissão de acesso ao arquivo a qualquer hora. Além da posse individual do arquivo, é possível que um ou mais usuários do sistema possuam o arquivo coletivamente, em um tipo de propriedade de grupo, conforme citado na seção 3.2.

O sistema Unix permite ainda três modos de acesso aos arquivos: leitura, escrita e execução. Os três modos de acesso são relativamente lógicos, porém o significado desses três modos de acesso é diferente para arquivos de diretórios. O usuário com permissão de leitura pode ler o conteúdo do diretório, por exemplo com o comando "ls". O usuário com permissão de escrita pode usar alguns programas privilegiados para gravar em um diretório. A permissão de gravação é necessária para criar ou remover arquivos do diretório. Um usuário deve ter permissão de execução em um diretório para ter acesso aos arquivos ali alocados. Se um usuário tem permissão para leitura e escrita em um arquivo comum que está listado em um diretório mas não tem permissão de execução para aquele diretório, o sistema não o deixa ler nem gravar o conteúdo daquele arquivo comum. A Figura 3.1 ilustra todas as permissões ativadas.

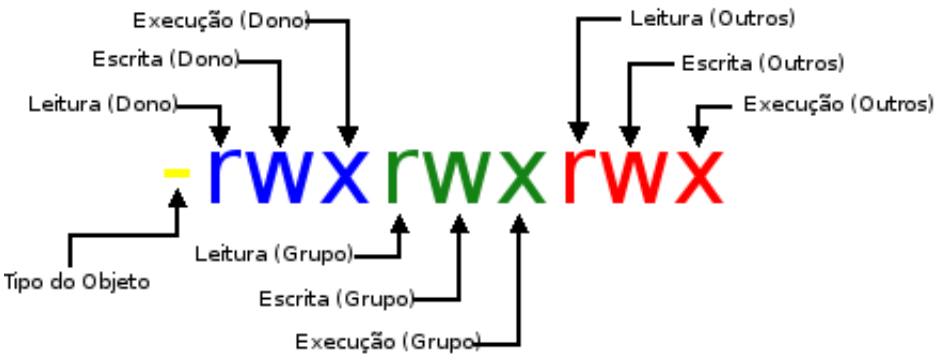


Figura 3.1: Ilustração contendo todas as permissões do UNIX.

Para alterar as permissões, utiliza-se o comando chmod. Observe a sintaxe para uso desse comando:

```
chmod [modo] nome_arquivo .
```

As permissões de leitura, escrita e execução podem ser alteradas pelo modo octal ou modo literal (simbólico). As Tabelas 3.1 e 3.2 mostram respectivamente a codificação para cada um desses modos.

r	w	x	Permissão	Valor octal
0	0	0	Sem permissão	0
0	0	1	Execução	1
0	1	0	Gravação	2
0	1	1	Gravação e execução	3
1	0	0	Leitura	4
1	0	1	Leitura e execução	5
1	1	0	Leitura e gravação	6
1	1	1	Permissão total	7

Tabela 3.1: Relações entre permissões e valor em octal.

Símbolo	Funcionalidade
u	Usuário proprietário
g	Grupo do arquivo
o	Outros
a	Todos (u, g, o)
+	Adicionar permissão
-	Remover permissão
=	Configurar permissão como
r	Leitura
w	Gravação
x	Execução

Tabela 3.2: Relações entre símbolo e funcionalidade para o comando chmod.

3.5 Diretórios

Os diretórios são organizados pela "árvore de diretórios", a estrutura de árvore é muito utilizada em ciência da computação e faz alusão às ramificações de uma árvore desde sua "raiz" até uma "folha". Nesse caso o diretório "raiz" possui símbolo "/" e nele estão contidos todos os outros subdiretórios.

Para navegar de um diretório para outro utilizamos o comando "cd" *change directory* que em seu uso mais simples basta utilizar o comando seguido pelo *path* do diretório de destino. O *path* consiste na lista de diretórios que precisa ser atravessada, desde o diretório raiz até o arquivo, com barras separando os componentes. O *path* pode ser absoluto que é o percurso completo desde a raiz até o arquivo de destino ou pode ser relativo que é o percurso da pasta atual até o arquivo de destino. As mudanças de diretório precisam seguir um *path* válido, ou seja, o diretório descendente precisa estar contido no diretório ascendente anterior e não pode haver saltos no percurso.

Alguns dos principais diretórios do sistema Linux, estão listados abaixo:

- `/dev - Contém arquivos especiais ou arquivos de dispositivos.`
- `/bin e /usr/bin - Contém comandos-padrão de Linux.`
- `/lib e /usr/lib - Possui as bibliotecas-padrão de Linux.`
- `/var - Possui arquivos de configuração e de log.`
- `/etc - Possui arquivos padrão de configuração.`
- `/media - Informações de dispositivos removíveis.`
- `/opt - Possui software comercial.`
- `/tmp - Armazena arquivos temporários.`
- `/home - Contém os diretórios e arquivos pessoais do usuário.`

3.6 Principais comandos

Os sistemas operacionais que operam Linux possuem uma estrutura geral da seguinte forma:

```
comando -opções -argumentos .
```

Podem ser inseridos diferentes comandos na mesma linha separando-os por ponto e vírgula, como, por exemplo:

```
cd /home; ls -all .
```

Nesse exemplo o comando "cd" navega da raiz até o diretório *home* e o "ls" lista todos os diretórios e arquivos presentes. Dentre os comandos básicos do Linux, os mais importantes são os de navegação nos diretórios, criação de novos arquivos e diretórios, alteração de permissões, cópia de arquivos e listagem de arquivos. Abaixo encontra-se uma lista com os principais comandos e suas funções.

- man [Nome do Comando] - "Manual" - Abre o manual do comando específico.

- cd [Nome do Diretório] - “Change Directory” - Muda o diretório atual.
- ls - “List” - Lista arquivos e subdiretórios no diretório atual.
- mkdir [Nome do Diretório] - “Make Directory” - Cria um diretório.
- rmdir [Nome do Diretório]- “Remove Directory” - Remove o diretório se estiver vazio.
- rm [Nome do arquivo]- “Remove” - Remove o arquivo ou diretório.
- mv [Nome do Arquivo] [Local para Mover] - “Move” - Move o arquivo.
- cp [Nome do Arquivo] [Local para Copiar] - “Copy” - Cria uma cópia do arquivo.
- clear - “Clear” - Limpa o texto do terminal.
- cat [Nome do Arquivo] - “Concatenate” - Mostra o conteúdo do arquivo.

Capítulo 4

Python aplicado a Raspberry Pi

A linguagem de programação Python [10] tem revolucionado o aprendizado de programação. A linguagem é também amplamente utilizada no mercado corporativo pela rápida implementação de aplicações completas. Nesta Seção vamos abordar uma perspectiva prática do Python no Raspberry Pi e sua utilização como interface de atuação e controle. A linguagem será aplicada para iniciar o leitor no controle de dispositivos eletrônicos como LED's e motores. Os ensinamentos poderão servir de base para diversas outras aplicações com outros sensores e dispositivos eletrônico fazendo as adaptações necessárias.

O grupo PET-Tele possui uma Apostila de Python [8], com todos os aspectos e elementos da linguagem. Neste tópico nosso foco é a aplicação prática e efetiva no Raspberry Pi.

4.1 Características básicas da linguagem

Python é uma linguagem de programação interpretada, de código fonte aberto e disponível para vários sistemas operacionais. Diz-se que uma linguagem é interpretada se esta não precisar ser diretamente compilada (traduzida para linguagem de máquina), e sim "lida" por um outro programa (chamado de interpretador) que traduzirá para a máquina. O interpretador de Python é interativo, ou seja, é possível executá-lo sem o fornecimento de um *script*. Ao invés disso, o interpretador disponibilizará uma interface interativa onde é possível inserir os comandos desejados um por um e ver o efeito de cada um deles.

4.2 Aspectos da sintaxe

Python usa indentação como delimitação de bloco, portanto devemos fazer indentação corretamente no código fonte. É boa prática fazermos comentários ao longo do programa que estamos criando seja para facilitar a leitura por outras pessoas ou então anotações rápidas como lembretes. Para criar comentários em Python colocamos uma cerquilha antes do texto do comentário. Observe o seguinte exemplo: # Este é um comentário que será ignorado pelo interpretador.

4.3 Declaração de variáveis

Uma variável não pode ser utilizada em uma expressão sem ter sido inicializada. Uma inicialização pode ser feita, por exemplo, por meio de uma atribuição. Não existe "criação

automática" de variáveis. Outro ponto a respeito das variáveis é a não obrigatoriedade de especificação do tipo da variável, ou seja, inteira, float, booleana, char, string.

Observe o seguinte exemplo de uma atribuição:

```
taxa = 5.2
reais = euros * taxa
```

A variável taxa foi inicializada pela atribuição de um valor. No entanto, a variável "euros" não foi inicializada em momento algum, nem mesmo com um valor nulo. Então no código acima será exibido o seguinte erro:

```
name 'euros' is not defined.
```

Pode-se fazer atribuições para mais de uma variável em uma mesma linha basta seguir a seguinte sintaxe.

```
x, y, z = 'Grupo', 'PET', 'Tele'
```

Outra alternativa pode ser a atribuição de um mesmo valor para mais de uma variável.

```
x = y = z = 'Hello World'
```

Um importante no aprendizado da linguagem é como se dá a estrutura da impressão das variáveis na tela. Observe a seguir as diferentes formas de impressão de texto.

```
y = 'Esse_e_o_grupo:'
x = 'PET_Tele'
#diversas formas de mostrar essa variavel na tela.
print(x) #imprime: PET_Tele
print('Esse_e_o_grupo:_' + x) #imprime: Esse_e_o_grupo:_PET_Tele
print(y + x) #imprime: Esse_e_o_grupo:_PET_Tele
```

Apesar de não ser obrigatório, podemos especificar o tipo de variável. A seguir são mostrados alguns dos principais tipos.

```
x = str('Hello World')
x = int(20)
x = float(20.5)
x = complex(1j)
x = list(('Grupo','PET','Tele'))
x = range(6)
x = bool(5)
```

4.4 Estruturas condicionais e iterativas

Python suporta condicionais lógicos usuais da matemática: Igual (a == b), Não igual (a != b), Menor que (a < b), Maior que (a > b), Menor ou igual que (a <= b), Maior ou igual que (a >= b).

```
if variavel = = True:
    print('true')
else:
    print('false')
```

Um laço [for] é usado para iterar sobre uma sequência que pode ser uma lista uma string ou outras estruturas. os laços [for] não necessitam de inicialização de uma variável de indexação antecipadamente.

```
lista = ['p', 'y', 't', 'h', 'o', 'n']
for item in lista:
    print item
```

Com um laço de [while] podemos executar um conjunto de expressões enquanto a condição seja verdadeira.

```
count = 0
while count <= 5:
    print(count)
    count += 1
```

4.5 Processo de criação e chamada de uma função

As linguagens de programação em geral têm o intuito de automatizar ações tornando-as mais rápidas. Se houver alguma ação que seja grande e utilizada com frequência, temos a opção de criar uma função que cumpra o seu objetivo, reduzindo o espaço ocupado pelo programa, além de deixá-lo com uma aparência mais limpa, em vista da diminuição do comprimento do código. Lembre-se da necessidade de indentação para pertencer à função.

```
def minha_funcao():
    print('Ola Mundo implementdo por funcao')
```

A declaração do nome da função em conjunto com os parenteses delimitadores dos argumentos denotam o procedimento de chamada da função para execução.

```
def minha_funcao():
    x=5
    print('Ola a partir da funcao')
return x

minha_funcao()
```

O retorno de uma função pode ser uma variável, uma constante, uma outra função, nada, ou então, a própria função. Quando a função retorna ela mesma é chamada de função recursiva e devemos ter cuidado para não criar retornos infinitos. Para obter algum dos tipos de retornos de uma função basta adicionar a expressão *return* e o elemento a ser retornado.

4.6 Funções para entrada e saída de dados

Python permite entradas a partir de usuários, isso significa que podemos receber do usuário uma ou mais *strings*. Operações matemáticas não podem ser feitas com *strings*, apenas com *floats* e inteiros, porém se somarmos *strings*, Python as juntará, num processo chamado concatenação e ao multiplicarmos uma *string* ela será repetida. Python 3.6 utiliza o comando *input()*. Python 2.7 utiliza o comando *raw_input()*.

```
Nome  = input('Qual seu nome?')
Cor = input('Qual sua cor favorita?')

print('Entao seu nome e %s, ' \
'e sua cor favorita e %s.' % (Nome, Cor) )
```

4.7 PIP - Gerenciador de pacotes e módulos

PIP é um gerenciador de pacotes e módulos, se você possui Python 3.4 ou posterior, PIP já está incluído por padrão. Um pacote contém todos os arquivos que você precisa para um módulo e um módulo são Bibliotecas de códigos em Python que podem ser incluídas em seu projeto. Pode-se verificar se o PIP está instalado navegando até seu diretório do *script* por linha de comando e digitar "pip –version". O gerenciador de pacotes PIP pode ser baixado por meio do seguinte URL :

`https://pypi.org/project/pip/}{https://pypi.org/project/pip/ .`

A instalação de um pacote com o PIP, é feita através da navegação até pasta do seu projeto e execução do comando: "pip install nome_do_pacote". A lista completa de pacotes disponíveis para instalaçao pode ser encontrada por meio do seguinte URL:

`https://pypi.org/ .`

Para remoção de um pacote é utilizado quase o mesmo processo acima, apenas há mudança na substituição do comando "install" por "remove". O comando "pip list" lista todos os pacotes atualmente instalados naquele diretório.

4.8 Importação de módulos

Pensando na reutilização de código, a linguagem Python possui um conjunto de funções e métodos prontos para serem usados ou agregados em seus programas. Essas funções estão agrupadas em estruturas denominadas módulos, agrupamento similar as bibliotecas. Para a utilização desses módulos é preciso utilizar o comando "import nome_do_modulo".

```
import math
print(math.sqrt(25))
```

O código acima importará todos os módulos de *math*. Caso o desenvolvedor queira importar apenas o necessário e não sobrecarregar desnecessariamente o programa utilizamos *from*.

```
from math import sqrt
```

Python possibilita a rápida criação de módulos pelo usuário. Para isso, basta criar um arquivo Python e salvá-lo. Por exemplo o trecho de código abaixo será salvo em um arquivo com nome "meumodulo.py".

```
def saudacao(nome):
    print('bem–vindo', + name)
```

Em um outro *script* contido na mesma pasta é possível importar o módulo criado.

```
import meumodulo
meumodulo.saudacao('Jose')
```

Existe também a possibilidade de renomear um módulo no momento da sua importação.

```
import meumodulo as modulo
modulo.saudacao('Jose')
```

4.9 Função *range()*

A função *range()* retorna um vetor contendo números inteiros sequenciais. O conteúdo deste vetor é definido de acordo com os possíveis argumentos:

```
range(Final)
range(Inicio, Final)
range(Inicio, Final, Passo)
```

4.10 Introdução à Programação Orientada a Objetos

Programação orientada a objetos em sua definição formal é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como atributos, e códigos, na forma de procedimentos, também conhecidos como métodos. Uma forma organizar o pensamento para lidar com o estilo de programação orientada a objetos pode ser a seguinte.

Uma classe é um construtor de objetos e age como se fosse um "projeto" para criar objetos. Toda classe Possui um nome de classe > Um objeto pode pertencer a uma classe > Um objeto possui características (atributos) e procedimentos (métodos) > Um dado pertencente a essa classe herda esses atributos e métodos.

A expressão *class* diz que existe a classe Triangle. Para executarmos operações nessa classe, precisamos definir funções que atuem sobre ela. Por exemplo, funções que armazenem os ângulos do triângulo:

```
class Triangle(object):

def __init__(self, angle1, angle2, angle3):
    self.angle1 = angle1
    self.angle2 = angle2
    self.angle3 = angle3
```

A seguir apresenta-se um caso prático de orientação a objeto. Costuma-se chamar o primeiro parâmetro de *self* que é uma referência a instância atual da classe e é usado para acessar posteriormente variáveis que pertencem a classe. Não precisa ser chamado de *self*, é apenas um padrão adotado por programadores que deve ser o primeiro parâmetro de qualquer função na classe. Um exemplo de classe em Python é mostrado a seguir.

```
class Triangle(object):

def __init__(self, angle1, angle2, angle3):
```

```
        self.angle1 = angle1
        self.angle2 = angle2
        self.angle3 = angle3

    def sum_triangle(self):
        return self.angle1 + self.angle2 + self.angle3

    def check_angles(self):
        if(self.triangle() == 180):
            return True
        else:
            return False
```

Confira um exemplo simples de como instanciar a classe definida acima em Python.

```
figure = Triangle(30, 40, 50)
```

As facilidades envolvendo a criação de classes e objetos reside no fato da possibilidade de alteração dos parâmetros desses objetos criados. O exemplo abaixo mostra esssa facilidade de alteração dos parâmetros:

```
figure = Triangle(30, 40, 50)
figure.angle3 = 60 #Modifica objeto
del figure #deleta objeto
```

4.10.1 Tratamento de erros *try*, *except*, *finally* e *raise*

O primeiro bloco de código *try* permite você testar um bloco de código para erros. O segundo bloco *except* permite lidar com esse erro. No exemplo abaixo o bloco *try* vai apresentar uma exceção, porque x não está definido. Então antes que o programa dê um erro e termine a execução o bloco de exceção é executado e a mensagem "Uma exceção aconteceu" é impressa na tela.

```
try:
    print(x)
except:
    print('Uma Excecao aconteceu')
```

No trecho abaixo o bloco *try* será executado, e passará no teste porque x está definido. O valor de x será exibido na tela.

```
try:
    x = 'Hello World'
    print(x)
except:
    print('Uma Excecao aconteceu')
```

O bloco de código *finally*, se especificado, é executado independentemente do bloco de código *try* levantar um erro ou não. Observe esse caso de uso no trecho abaixo.

```
try:
    print(x)
except:
```

```
        print('Uma Excecao aconteceu')
    finally:
        print('O teste try except foi finalizado')
```

Um outro método que pode ser usado é o levantamento de uma exceção se uma determinada condição ocorrer. O trecho abaixo levanta um erro e força uma parada no programa se a variável "x" for menor que 0.

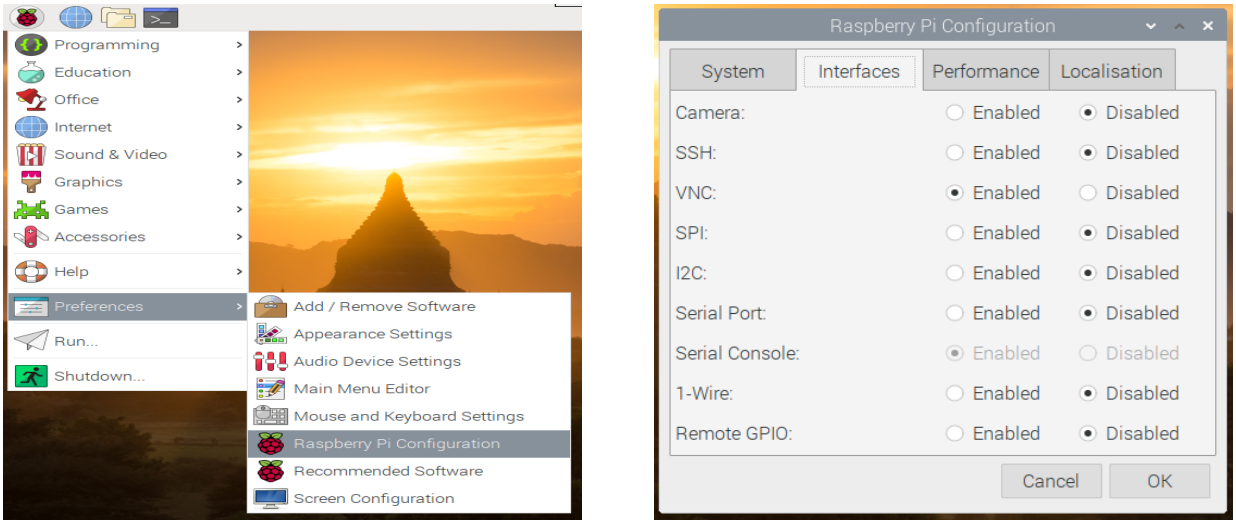
```
x = -1
if x < 0:
    raise ('Erro, numeros abaixo de zero nao aceitos')
```

4.11 Configuração do acesso remoto por SSH

Antes darmos início aos experimentos com componentes eletrônicos de bancada, talvez seja útil saber que o Raspberry Pi pode ser utilizado sem o uso monitor realizando o acesso ao dispositivo remotamente por meio do protocolo SSH [12]. Para isso será necessário conectar o dispositivo a uma rede local ou internet. O acesso por protocolo SSH é limitado a linha de comando do Raspberry Pi, não existe interface gráfica. O primeiro passo é verificar se o Raspberry Pi está apropriadamente configurado e conectado na rede, utilizando o comando "ifconfig" no terminal do Raspberry Pi. Visualize e anote o endereço IP exibido na tela.

Em seguida é preciso habilitar o "Server SSH", que por padrão está desativado. Para isso localize no menu superior, através da interface gráfica o item "preferences" e "Raspberry Pi Configuration". Na nova janela que será exibida localize a guia "interfaces" e habilite a opção "SSH" confirmando a seleção.

Aproveite para entrar na guia *System*, e troque sua senha, bem como visualizar e anotar o *Hostname* para futura identificação do dispositivo. Esses passos são mostrados conforme a Figura 4.2. Feito isso, estará pronto para acessar o Raspberry Pi de um outro computador dispensando o uso de monitor. Caso opte por esse método a navegação somente será feita através de linha de comando.



(a) Lista de opções. (b) Aba interfaces.

Figura 4.1: Menu configurações Raspberry Pi.

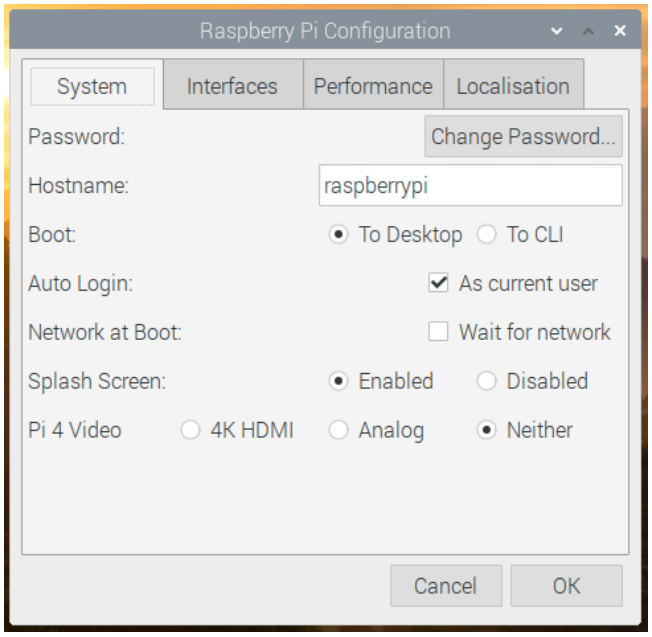


Figura 4.2: Aba *System*.

Uma maneira alternativa de acessar essas configurações é através do terminal pelo comando "raspi-config". Seguindo os passos abaixo também será possível efetuar a habilitação do SSH atráves de linha de comando.

1. Entre com "sudo raspi-config" em uma janela do terminal.
2. Selecione "interfacing Options".
3. Navegue para "SSH" e faça a seleção.
4. Escolha "Yes" e confirme.
5. Escolha "Finish".

Capítulo 5

Atividades com componentes eletrônicos

A primeira atividade pode ser considerada o *Hello, World* da prototipagem eletrônica, o acendimento de LEDs. Para fazermos a ligação do Raspberry Pi com o LED, vamos utilizar os pinos GPIO e uma placa de prototipagem (*Protoboard*). O ânodo do LED, se conecta ao GPIO27 (Pino 13) e o cátodo é conectado em série com um resistor de 220Ω no terminal comum, conforme a Figura 5.1.

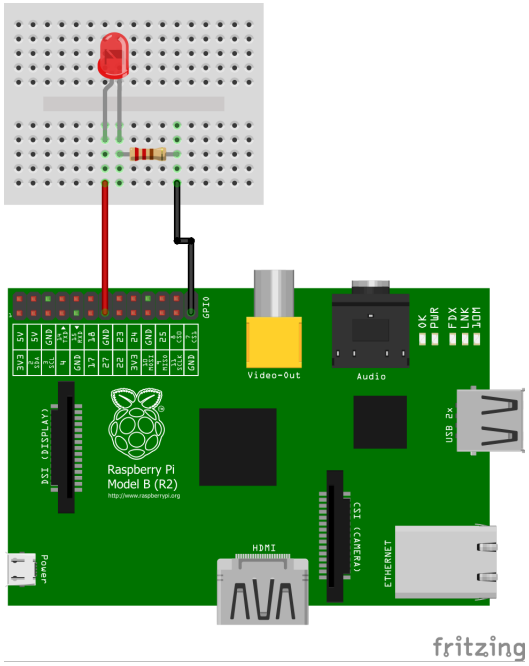


Figura 5.1: Conexões do LED.

5.1 Piscar um LED

O primeiro exemplo vai tratar da criação de um programa para acender e apagar um LED. Antes de começarmos a criar programas em Python precisamos instalar as bibliotecas para manipulação dos GPIO.

```
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio .
```

Utilize os comandos de Linux que aprendidos para criação de um diretório, criação de um arquivo Python e abertura desse arquivo em um editor de textos. Dentro desse arquivo Python

devemos importar dois módulos. O "rpi.GPIO" para fazer o Python reconhecer os pinos e o módulo *time*, para o uso das funções *sleep*, responsáveis pelo tempo em que o LED fica ligado e desligado.

```
import RPi.GPIO as GPIO
from time import sleep
```

O próximo passo é fazer a declaração de uma variável para armazenar o número do pino em que se encontra o LED e a utilização das funções *setmode* e *setup*, para definir o código de numeração e o modo de operação dos pinos respectivamente.

```
led_pin = 13
GPIO.setmode(GPIO.BOARD) #alternativo GPIO.BCM led_pin = 27
GPIO.setup(led_pin, GPIO.OUT, initial=GPIO.LOW)
```

Em seguida basta implementar um laço de repetição com infinitas alterações do estado da saída, ora alta (*High*), ora baixa (*Low*).

```
while True: # Execucao em laco infinito
    GPIO.output(led_pin, GPIO.HIGH)
    sleep(1) # Sleep for 1 second
    GPIO.output(led_pin, GPIO.LOW)
    sleep(1)
```

5.2 Controle PWM em LEDs

A técnica de PWM já conceituada na seção 2.4 é empregada em diversas áreas da eletrônica, talvez a mais comum seja a utilização em fontes chaveadas mas também pode ser utilizada para controle de velocidade de motores, controle de luminosidade, controle de servomotores e diversas outras aplicações. Através da largura do pulso de uma onda quadrada é possível o controle de potência e velocidade, ou seja, tensão média [6].

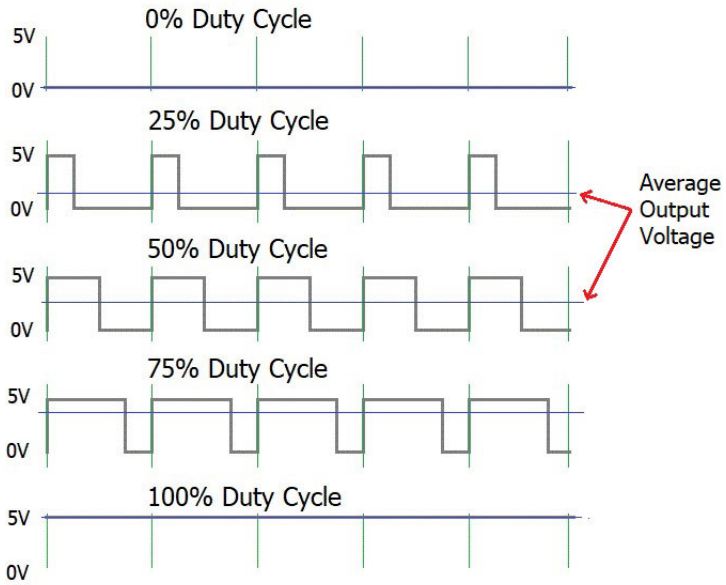


Figura 5.2: Diagrama PWM.

Conforme mostra a Figura 5.2, variando a largura do pulso, existe uma tensão média para cada ciclo de trabalho. Esses ciclos podem ser calculados conforme as Equações 5.1 e 5.2.

$$DutyCycle = 100 \cdot \frac{LarguraPulso}{Periodo} \tag{5.1}$$

$$Periodo = LarguradoPulso + Tempodesligado \tag{5.2}$$

Existem dois tipos de modulação PWM no Raspberry Pi, *hardware* PWM e *software* PWM. No primeiro tipo a temporização dos pulsos e a geração do sinal PWM é controlada por circuito integrado responsável embarcado na placa. Pode ser gerada apenas no GPIO18 e por possuir maior precisão, é mais adequada para controlar servos. O *software* PWM está disponível em todos os pinos. A temporização dos pulsos é controlada pelo escalonador *Scheduler* Linux. Seu uso é mais flexível e é adaptável para uma variedade maior de frequências. É ideal para exercer tarefas que não necessitem muita precisão.

- GPIO.PWM (Pino, Frequência) → Define um Pino e uma frequência em hertz para utilizar PWM.
- NomeVar.start(Ciclo de trabalho [0-100]) → Define um ciclo de trabalho de início para variável escolhida.
- NomeVar.ChangeDutyCycle(Ciclo de trabalho [0-100]) → Muda o ciclo de trabalho para a variável escolhida.

Observe o exemplo abaixo que inicializa um pino em PWM e ciclo de trabalho zero (Desligado).

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)
pwm = GPIO.PWM(led_pin, 100)
pwm.start(0)
```

Exercício 01 - Crie um programa em Python que Ligue um LED conectado no Raspberry Pi em *Fade In* e desligue o em *Fade Out*, utilizando a técnica PWM e um *loop FOR* para passe por cada cada valor de ciclo de trabalho. Se estiver com dificuldades siga os seguintes passos:

1. Abra o terminal.
2. crie um diretório, por exemplo, “SeTeL2019”.
3. Entre no diretório.
4. Crie um arquivo Python.
5. Abra-o em um editor de textos de sua escolha.
6. Escreva o programa em Python para ligar e desligar um LED.
7. Salve o arquivo.
8. Execute o arquivo pelo terminal.

5.3 Controle de servomotor por PWM

Um servomotor é um equipamento eletro-eletrônico que apresenta movimento proporcional a um sinal de comando. Possui circuitos integrados para verificar sua posição atual e realizar movimentos para novas posições em contraste com os motores contínuos que giram indefinidamente. O eixo dos servomotores possui liberdade de 180° e são precisos quanto à sua posição.

O Grupo PET-Tele possui o documento "Novas implementações para automatização e acesso remoto no sistema de reservatórios de detenção aplicados à drenagem urbana do LaDISan/T-CE/UFF utilizando a plataforma Raspberry Pi" [5] que detalha e explica o funcionamento de um servomotor em uma aplicação. Os circuitos de controle presentes no servomotor possibilitam sua utilização com técnicas de modulação PPM ou PWM. A seguir é mostrado um exemplo de como implementar um controle de posição com PWM.

Para início pode ser feita a conexão entre o servomotor e o Raspberry Pi. O servomotor possui 3 fios, tensão que pode ser de +5V à +12V dependendo do modelo, terminal comum (GND) e sinal que irá receber o sinal de controle. A Figura 5.3 mostra como devem ser feitas as ligações.

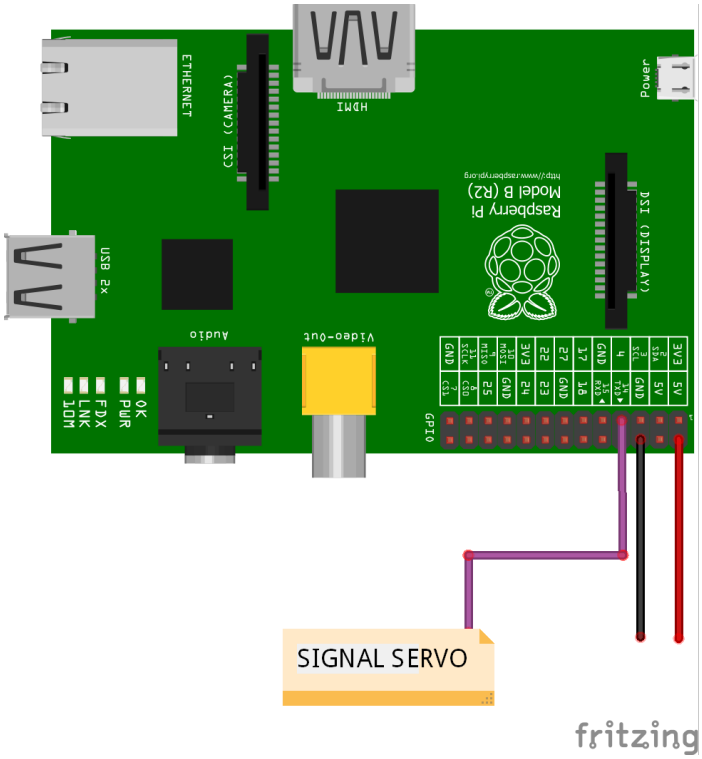


Figura 5.3: Conexão entre servomotor e Raspberry Pi.

Começamos importando respectivamente as bibliotecas "RPi.GPIO" e "time" para trabalhar com os GPIO do Raspberry pi e as funções temporais.

```
import RPi.GPIO as GPIO
import time
```

Em seguida, definimos algumas variáveis com estados iniciais. Os valores são baseados nas especificações da maioria dos servomotores, ou seja, são constantes para boa parte dos servomotores. Um *duty cycle* de 2 % corresponde a um ângulo de 0°, um *duty cycle* de 12 % corresponde a um ângulo de 180°. A partir disso, podemos encontrar que 2 % de *duty cycle*

equivale a um pulso de 0.5 ms e 12 % de *duty cycle* a um pulso de 2.5 ms. A equação 5.1 mostra essa relação entre largura do pulso e *duty cycle*.

```
servo_pin = 4 # definicao do pino
f = 50.0 # frequencia do pino PWM em (Hz)
deg_0_pulse   = 0.5 # largura do pulso para 0
deg_180_pulse = 2.5 # largura do pulso para 180
```

No trecho abaixo aplicam-se as devidas conversões de grau para *duty cycle* e demais expressões para determinar a faixa de todos os ângulos entre 0 e 180°.

```
period = 1000/f # periodo do sinal PWM
deg_0_duty = 100 * deg_0_pulse/period # conversao grau-duty cycle
pulse_range = deg_180_pulse - deg_0_pulse # faixa de pulsos
duty_range = 100 * pulse_range/period # faixa de duty cycle
```

Os métodos *setmode*, *setup* e *GPIO.PWM* configuram o pino para desempenhar o controle do servomotor. O *setmode* está relacionado ao tipo de numeração que será utilizado, o *setup* indica se é um pino de entrada ou saída e o *GPIO.PWM* define que será um pino de PWM a uma determinada frequência escolhida.

```
GPIO.setmode (GPIO.BOARD)
GPIO.setup (servo_pin, GPIO.OUT)
pwm = GPIO.PWM(servo_pin, f)
pwm.start(0)
```

Conforme visto na Subseção 4.5 sobre criação de funções. Podemos definir uma função para ajustar um ângulo para o servomotor realizando a operação de ajuste de um *duty cycle*. Essa operação de mudança *duty cycle* está presente na biblioteca GPIO.

```
def set_angle (angle):
    duty = deg_0_duty + (angle/180.0)* duty_range
    pwm.ChangeDutyCycle(duty)
```

Utiliza-se o método try conforme visto na Subseção 4.10.1 para realizar uma ação de teste. O teste consiste na execução da função "set_angle" de acordo com o valor dado como entrada pelo usuário repetidas vezes, até uma interrupção.

```
try:
    while True:
        angle = input ('Enter angle (0 to 180):')
        set_angle(angle)
```

O encerramento do código é marcado com a execução da limpeza das definições do pino e impressão de uma mensagem de finalização.

```
finally:
    print('cleaning up')
    GPIO.cleanup()
```

Exercício 02 - Escreva um programa em Python que movimente um servomotor conectado a um Raspberry Pi começando em 0° e movendo-se continuamente em intervalos de 100 ms até 180°. Quando o servomotor chegar em 180° mova diretamente o ângulo para 0° e reinicie o movimento. Utilize a técnica PWM e um *loop* com 5 repetições.

Referências Bibliográficas

[1] 2020 PINBALLSP. **Página Pinballsp**. Disponível em: <https://www.pinballsp.com/>. Online; Accesso em: 12 Jan. 2021.

[2] BEN CROSTON. ***biblioteca em python rpi.gpio 0.7.0***. Disponível em: <https://pypi.org/project/RPi.GPIO/>, 2019. Online; Accesso em: 29 Dez. 2019.

[3] DEPARTAMENTO DE ENGENHARIA AGRÍCOLA E MEIO AMBIENTE. **Laboratório de Drenagem, Irrigação e Saneamento Ambiental**. Disponível em: <http://www.uff.br/?q=setor/laboratorio-de-drenagem-irrigacao-e-saneamento-ambiental-ladisan>, 2019. Online; Accesso em: 12 Nov. 2019.

[4] JOAN2937. ***biblioteca em python pigpio***. Disponível em: <https://github.com/joan2937/pigpio>, 2019. Online; Accesso em: 29 Dez. 2019.

[5] JOÃO LUIZ DE AMORIM PEREIRA NETO, RAPHAEL MIRANDA. **Novas implementações para automatização e acesso remoto no sistema de reservatórios de detenção aplicados à drenagem urbana do LaDISan/TCE/UFF utilizando a plataforma Raspberry Pi**. Disponível em: <http://www.telecom.uff.br/pet/petws/index.php?pagina=downloads/projetos>, 2020. Online; Accesso em: 12 Jan. 2021.

[6] MECAWEB EDUCATION SITE. **PWM - Modulação por Largura de Pulso**. Disponível em: <http://www.mecaweb.com.br/eletronica/content/e_pwm>, 2019. Online; Accesso em: 15 out. 2019.

[7] PET-TELE. **Apostilia Linux**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/apostilas/LINUX.pdf>, 2004. Online; Accesso em: 15 out. 2019.

[8] PET-TELE. **Apostila de Python**. Disponível em: <http://www.telecom.uff.br/pet/petws/downloads/apostilas/PYTHON.pdf>, 2011. Online; Accesso em: 15 out. 2019.

[9] PET-TELE. **URL do Grupo**. Disponível em: <https://www.telecom.uff.br/pet/petws/index.php>, 2019. Online; Accesso em: 24 Set. 2019.

[10] PYTHON SOFTWARE FOUNDATION. **WebSite Python**. Disponível em: <https://www.python.org/>, 2001-2020. Online; Accesso em: 15 Abr. 2020.

[11] RASPBERRY PI FOUNDATION. **Pi Foundation Site**. Disponível em: <https://www.raspberrypi.org/>. Online; Accesso em: 24 Set. 2019.

[12] RASPBERRYPI FOUNDATION. **SSH (Secure Shell)**. Disponível em: <https://www.raspberrypi.org/documentation/remote-access/ssh/>, 2020. Online; Accesso em: 18 Jun. 2020.

[13] SOFTWARE IN THE PUBLIC INTEREST, INC. Debian os. Disponível em: <https://www.debian.org/>, 2021. Online; Accesso em: 18 Jan. 2021.

[14] WIKIPEDIA MEMBERS. **Raspberry Pi Wikipedia**. Disponível em: <https://pt.wikipedia.org/wiki/Raspberry_Pi>. Online; Accesso em: 24 Set. 2019.

[15] WIKIPEDIA MEMBERS. **System on a chip - SoC**. Disponível em: <https://pt.wikipedia.org/wiki/System-on-a-chip>. Online; Accesso em: 24 Set. 2019.

[16] WIKIPEDIA MEMBERS. **Aplicação Monolítica**. Disponível em: <https://pt.wikipedia.org/wiki/Aplicaçao_monolítica>, 2019. Online; Accesso em: 15 out. 2019.

[17] WIKIPEDIA MEMBERS. **Linux**. Disponível em: <https://pt.wikipedia.org/wiki/Linux>, 2019. Online; Accesso em: 15 out. 2019.