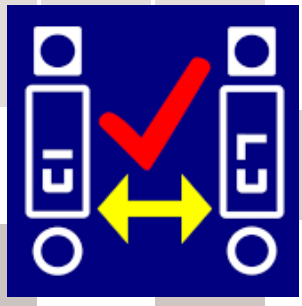
Aplicações Práticas de sistemas embarcados Linux utilizando Raspberry Pi

Sandro Jucá e Renata Pereira



Sandro Jucá e Renata Pereira

Aplicações Práticas de sistemas embarcados Linux utilizando Raspberry Pi

PoD editora
Rio de Janeiro
2018

Aplicações Práticas de sistemas embarcados
Linux utilizando Raspberry Pi

Impressão e Acabamento:
Pod Editora
Rua Imperatriz Leopoldina, 8/1110 – Pça Tiradentes
Centro – 20060-030 – Rio de Janeiro
Tel. 21 2236-0844 • atendimento@podeditora.com.br
www.podeditora.com.br

Projeto gráfico:
Pod Editora

Revisão:
Pod Editora

Capa:

Os autores

CIP-BRASIL. CATALOGAÇÃO-NA-FONTE
SINDICATO NACIONAL DOS EDITORES DE LIVROS, RJ

J84a

Juca, Sandro

Aplicações Práticas de sistemas embarcados Linux utilizando Raspberry Pi [recurso eletrônico] / Sandro Jucá e Renata Pereira. 1ª ed. - Rio de Janeiro: PoD, 2018. recurso digital ; 21MB.

Formato : epdf
Requisitos do sistema : Adobe Acrobat Reader
Inclui bibliografia e índice
ISBN 978-85-8225-212-3 (recurso eletrônico)

1. Raspberry PI (Computador). 2. Linux (Sistema operacional de computador). 3.Livros eletrônicos. I. Pereira, Renata. II. Título.

18-53985 CDD: 004.16
CDU: 004.382.4 30/11/2018

Leandra Felix da Cruz - Bibliotecária - CRB-7/6135

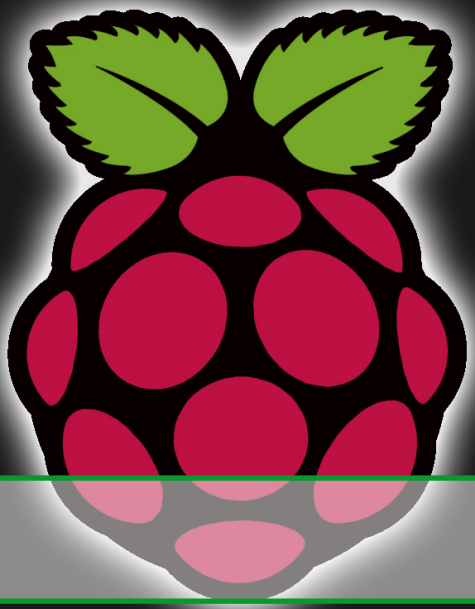
Comitê Editorial

Financiamento da presente obra provém de recurso do Edital Proinfra 2017, do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), do Programa de Pós-Graduação em Educação Profissional e Tecnológica do IFCE (ProfEPT), em articulação com o Programa de Pós-Graduação a nível de Mestrado Acadêmico strictu sensu em Ensino de Ciências e Matemática (PGECM). Os Organizadores da presente obra agradecem a Deus, ao IFCE, como também aos amigos e familiares.

Sobre os autores:

Sandro César Silveira Jucá possui Nivelamento Universitário (Studienkolleg) na Technische Hochschule Köln - Alemanha (1996 a 1998), Graduação em Tecnologia Mecatrônica pelo Instituto Federal do Ceará (IFCE) em 2002. É Especialista em Automação Industrial (2003), Licenciado em Física, em 2005, pela Universidade Estadual do Ceará (UECE), Mestre (UFC, em 2004) e Doutor em Engenharia Elétrica (UFC, em 2014) com pesquisa realizada na Universität Paderborn - Alemanha, pelo programa de Doutorado Sanduíche do Deutscher Akademischer Austauschdienst (DAAD). Atualmente é professor, pesquisador e coordenador de Pesquisa, Pós-Graduação e Inovação do IFCE - Campus Maracanaú e membro do corpo docente do ProfEPT (Mestrado em educação profissional e tecnológica em rede nacional com Qualis Ensino) e do Mestrado Acadêmico em Energias Renováveis (PPGER com Qualis Interdisciplinar). Pesquisa na área de Mecatrônica e Engenharia Elétrica, dentro dos seguintes temas: Energias Renováveis, Sistemas Embarcados, IoT, EaD, Robótica e Educação Profissional.

Renata Imaculada Soares Pereira concluiu Doutorado (2018) com bolsa de doutorado sanduíche na Technische Hochschule Köln - Alemanha e Mestrado (2014) em Engenharia Elétrica pela Universidade Federal do Ceará (UFC). No Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE - Campus Maracanaú) concluiu Tecnólogo em Manutenção Industrial (2012) e curso técnico em Automação Industrial (2009). Áreas de atuação: Sistemas embarcados IoT, Aquisição de dados e Monitoramento online em Nuvem, Energias Renováveis.



Conteúdo

1. Introdução

1.1 Sobre o Raspberry Pi

O Raspberry Pi, na época de lançamento considerado o menor computador do mundo, possui o tamanho de um cartão de crédito, conexões USB para conectar o teclado e o mouse utilizado em computadores de mesa. É possível conectá-lo a TVs com saída HDMI, como pode ser visto na Figura 1.1-A juntamente com a descrição das demais conexões. Além destas vantagens, pode-se destacar o baixo custo do hardware, além do custo zero do software embarcado, baseado em Linux.

Todo o hardware é integrado em uma única placa. O principal objetivo dos desenvolvedores foi promover o ensino em Ciência da Computação básica em escolas, principalmente públicas. É um pequeno dispositivo que permite que pessoas de todas as idades possam explorar a computação para aprender a programar em linguagens como C e Python. É capaz de desenvolver tudo que um computador convencional faz como navegar na internet, reproduzir vídeos de alta definição, fazer planilhas, processar textos, brincar com jogos, além de processar tarefas mais complexas como monitoramento online. Dessa forma, é utilizado por crianças de todo o mundo para aprender como funcionam os computadores, como manipular o mundo eletrônico ao redor deles, e como programar. Versões do Raspberry Pi com vídeo-aulas e outros materiais de treinamento poderiam ser úteis em projetos de inclusão digital, já que o baixo custo permitiria não apenas que eles fossem usados em laboratórios, como também fornecidos aos estudantes para aprender programação em

domicílio juntamente com materiais didáticos.

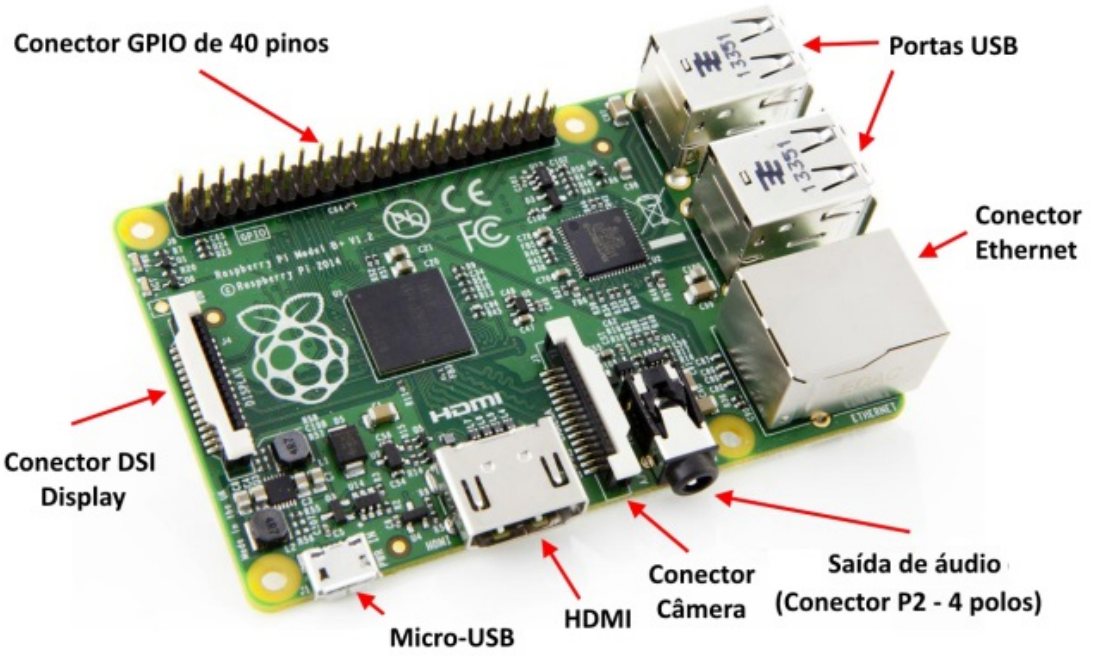


Figura 1.1: Raspberry Pi Modelo B+

Além disto, o RPi possui a capacidade de interagir com o mundo exterior através de sensores. Indo além, trabalhando em conjunto com microcontroladores, pode ser usado em uma ampla gama de projetos digitais. Aqui trabalharemos com microcontroladores PIC, através da placa da Ferramenta SanUSB disponível em http://sanusb.org.

O computador RPi (versão B+) utilizado como base nos projetos descritos nessa apostila é baseado em um *system on a chip* (SoC) Broadcom BCM2835 [1] que inclui um processador ARM1176JZF-S de 700 MHz com uma GPU VideoCore IV,9 operando a 250 MHz e 512 MB de memória RAM em sua última revisão. Apesar da frequência de processamento (clock) parecer baixa em comparação às GPUs para desktops, esta é uma GPU que oferece um poder de processamento superior à Power VR SGX 535 usada no iPhone e em outros dispositivos, inclusive com suporte à decodificação de vídeos 1080p via hardware. A placa não inclui uma memória não-volátil, como um disco rígido, mas possui uma entrada de cartão microSD para armazenamento de dados.

A alimentação elétrica fica por conta de uma porta micro-USB localizada ao lado do cartão de memória. Esta foi escolhida para simplificar e baratear o projeto, já que permite que ele seja alimentado por qualquer carregador de celular (ou por um carregador veicular ligado a uma bateria de 12V e/ou placa solar) e permite que os 5V recebidos sejam enviados diretamente para componentes que usam 5V, como dispositivos USB plugados e a porta HDMI.

Embora possua duas portas USB (no modelo B), o Rpi é limitado em relação à quantidade de energia que pode ser fornecida a dispositivos conectados à porta USB, já que ele mesmo também é

alimentado através de uma porta USB. Os conectores são destinados a dispositivos como teclados e mouses, bem como pendrives e outros dispositivos de baixo consumo. Para usar dispositivos de maior consumo, como HDs externos, é necessário usar um extensor (hub) USB com alimentação própria. Mesmo placas WiFi podem ser um problema, demandando que a fonte de alimentação seja capaz de fornecer pelo menos 700 mA. Normalmente é necessária uma fonte coma a capacidade maior que 1.500 mA, quando é utilizada a interface Ethernet. É possível visualizar na Figura 1.2 as especificações dos quatro primeiros modelos de Rpi. O modelo B apresentava as mesmas conexões do B+, porém com 1 GB de RAM e processador de 900 MHz.

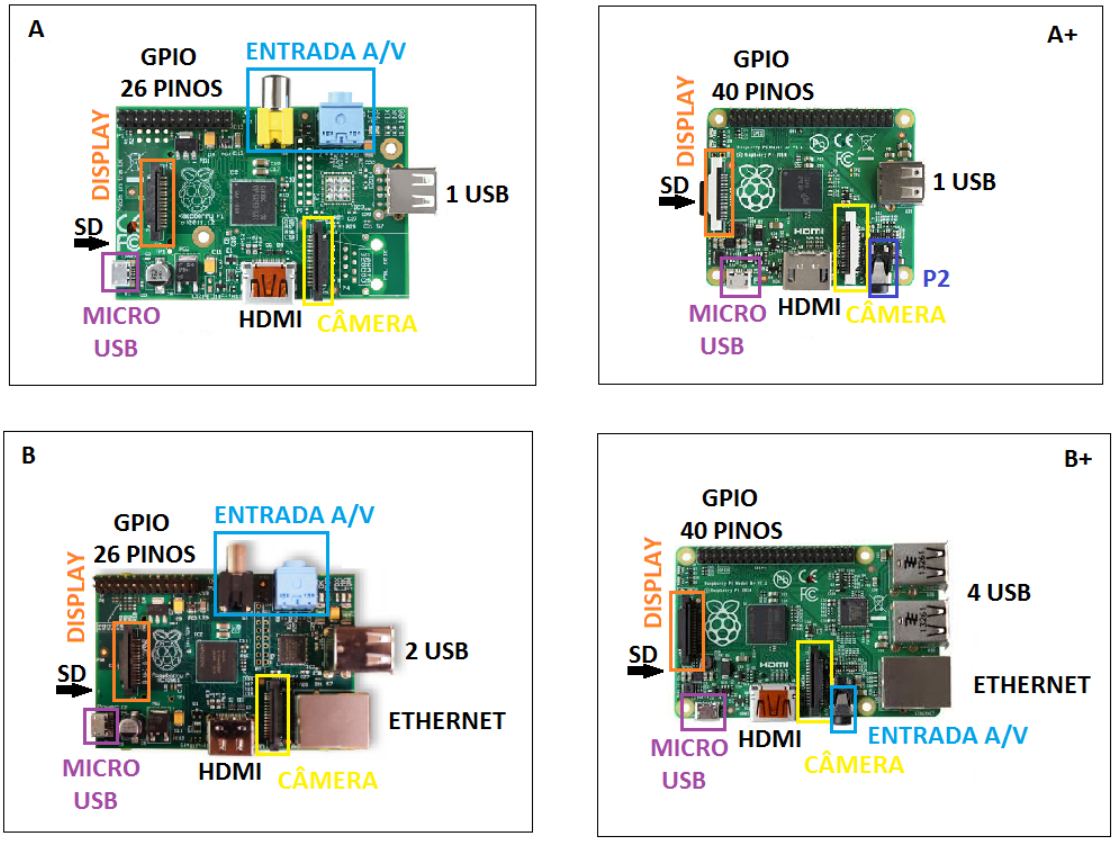


Figura 1.2: Quatro modelos iniciais de Raspberry Pi.

O modem Raspberry Pi 2 apresentava o mesmo tamanho e a mesma configuração de 40 pinos do modelo B+ com a seguinte configuração:

- 900MHz quad-core 32 bits ARM Cortex-A7 CPU;
- 1GB RAM;
- 4 portas USB;
- Porta Full HDMI;
- Interface de camera (CSI);
- Interface de Display (DSI);
- Slot Micro SD card.

A memória RAM passou a ser de 1GB (contra 512MB da versão anterior) e a CPU passou a ser 900MHz quad-core e a versão inicial era um CPU 700 MHz single-core ARMv6.

Já o modelo Raspberry Pi 3 apresenta as mesmas conexões físicas do Raspberry Pi 2, assim como a quantidade de memória RAM (1GB). Como adicional é apresentado modem Wi-Fi 802.11 b/g/n (2.4GHz) e Bluetooth 4.1 (BCM43438) integrados na placa e um processador BCM2837 de 4 núcleos de 64 bits ARM Cortex-A53 com a frequência de 1.2GHz. Existe também um modelo reduzido com dimensões de 6,5cm x 3cm e melhor custo-benefício para aplicações sem processamento de imagens, que é a versão Raspberry Pi Zero W ilustrado na Figura 1.3.

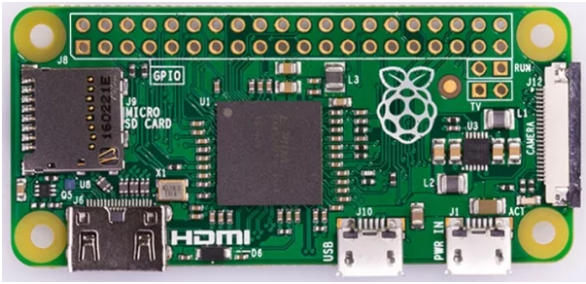


Figura 1.3: Raspberry Pi Zero W

O modelo Raspberry Pi zero WH é o mesmo que o Raspberry Pi Zero W só que vem com os pinos já soldados. O dispositivo possui uma CPU Arm 11 single-core, com velocidade de 1 GHz, memória RAM de 512 MB, GPIO de 40 pinos e suporte para cartão microSD. O Rpi zero W também já vem com Wi-Fi 802.11n, Bluetooth Low Energy (BLE) 4.1, GPU Dual Core VideoCore IV, além de saídas mini HDMI, micro USB e CSI para conectar uma câmera.

O último modelo lançado até o momento é o Raspberry Pi 4. É mais rápido que seus antecessores com um processador *quad-core* da Broadcom com núcleos Cortex-A72 com frequência 1,5 GHz. Possui três opções de memória RAM: 4 GB (modelo vendido no Brasil), 2 GB ou 1 GB, como ilustrado na Figura 1.4.

Além disso, possui duas portas USB 3.0, duas USB 2.0, porta Gigabit Ethernet, como também duas saídas micro-HDMI e alimentação é feita através da entrada USB-C. Uma tabela comparativa entre os modelos de Raspberry Pi é mostrada na Figura 1.5.

Diferente de um PC, o Rpi não possui BIOS ou Setup. Em vez disso, todas as configurações relacionadas ao hardware e ao processo de boot são feitas em um arquivo de texto localizado no diretório raiz do cartão, o "config.txt", que engloba muitas opções que em um PC estariam disponíveis no Setup, incluindo a frequência de operação do processador.

Embora venha sem sistema operacional, o Rpi é compatível com várias distribuições Linux, incluindo o Debian (Raspbian), Arch Linux e Fedora. Diferente do que se tem na plataforma PC, não existe uma imagem única para dispositivos ARM, já que a plataforma carece de BIOS, enumeração de dispositivos plug-and-play e outras funções, o que dificulta a detecção do hardware

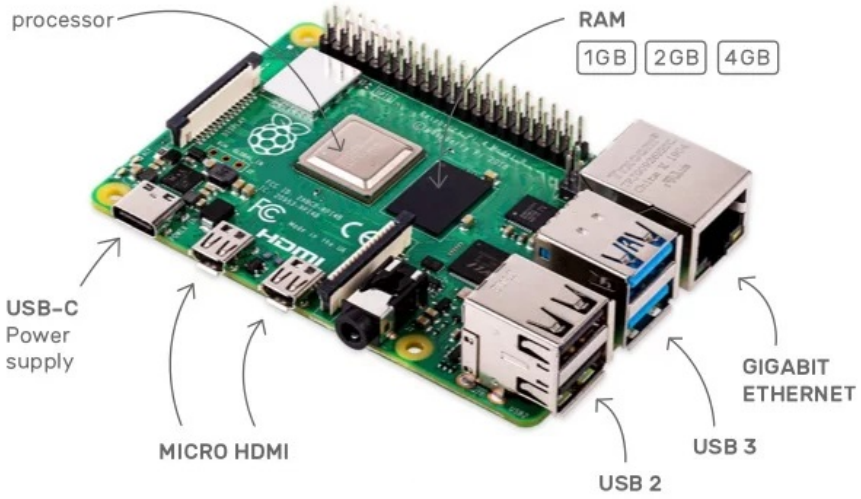


Figura 1.4: Raspberry Pi 4

automaticamente durante o boot. Vale salientar que a gravação via USB de microcontroladores PIC proposta nesse trabalho é baseada no protocolo HID (*Human Interface Device*) e este protocolo plug-and-play é reconhecido automaticamente no boot.

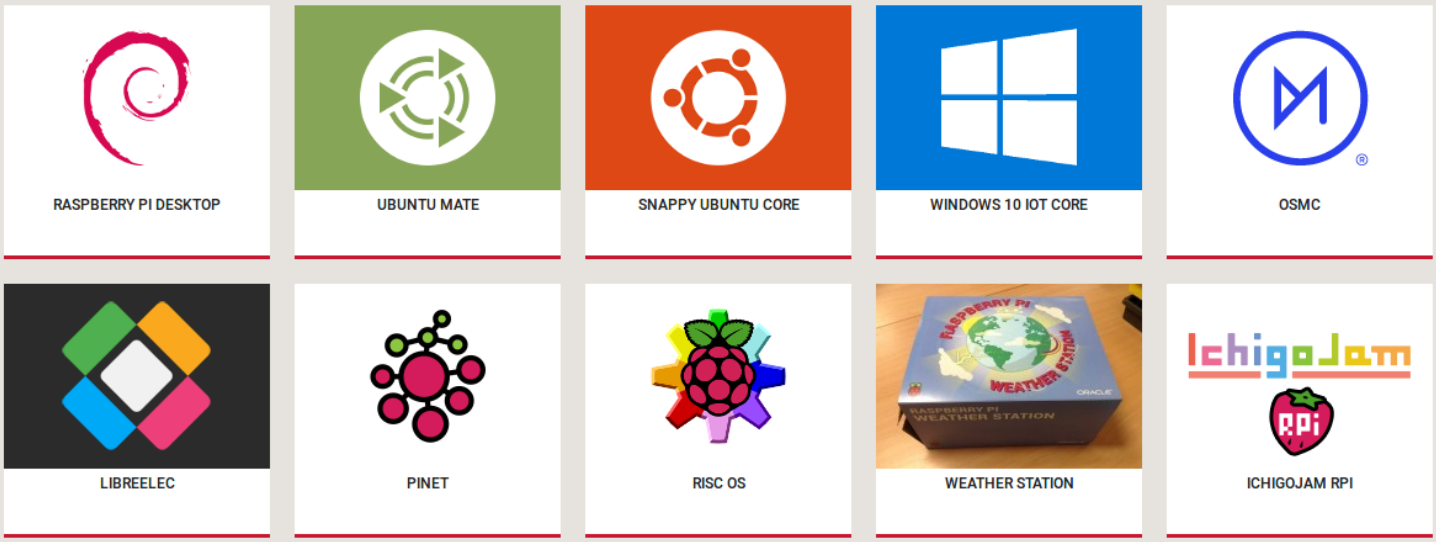
Nos processadores ARM é necessário que uma imagem específica do sistema operacional seja desenvolvida para o dispositivo. Dessa forma, uma das vantagens é que vários voluntários já estão fazendo isso, disponibilizando imagens que estão disponíveis para download [Raspberry 2015]. A instalação destas imagens é simples, consistindo apenas em gravar a imagem no cartão SD usando um software ou outro utilitário de cópia bit a bit, inserir o cartão SD no Rpi e reiniciar o sistema operacional.

Para prover comunicação entre o Rpi e computadores é possível conectá-lo em rede utilizando o padrão Ethernet, Wifi ou realizar comunicação serial que utiliza a porta serial (UART).

	Raspberry Pi A+	Raspberry Pi model B	Raspberry Pi 2	Raspberry Pi Zero	Raspberry Pi Zero W	Raspberry Pi 3	Raspberry Pi 3 model B+	Raspberry Pi 4 Model B
Lançamento	10/11/2014	15/02/2012	01/02/2015	30/11/2015	28/02/2017	29/02/2016	2018	2019
Preço US$	US$20.00		US$35.00	**US$5.00**	**US$10.00**	US$35.00	US$35.00	**US$35.00 a US$55.00**
Chip	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2837	Broadcom BCM2837B0	Broadcom BCM2711
Core	ARM1176JZF-S	ARM1176JZF-S	Cortex-A7	ARM1176JZF-S	ARM1176JZF-S	Cortex-A53 64-bit	Cortex-A53 (ARMv8) 64-bit	Cortex-A72 (ARM v8) 64-bit
Nº Core	1	1	4	1	1	4	4	4
Clock CPU	700 MHz	700 MHz	900 MHz	1 GHz	1 GHz	1.2 GHz	1.4 Ghz	1.5GHz
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore VI
RAM	256 MB	512 MB (256 - model A)	1 GB	512 MB	512 MB	1 GB	1GB LPDDR2 (900 MHz)	**1GB, 2GB ou 4GB**
Wireless	X	X	X	X	802.11n	802.11n	802.11n 2.4GHz/5.0GHz	802.11ac 2.4 GHz/5.0 GHz
Bluetooth	X	X	X	X	4.1	4.1	4.1	5.0, BLE
Consumo	180 mA	500 mA	350 mA	100 mA	150 mA	350 mA	500 mA	**600 mA**

desde o mais limitado -> até o mais potente

Figura 1.5: Comparação dos modelos de Raspberry Pi



2. S.O. no Raspberry Pi

Sistemas Operacionais (S.O.) podem ser definidos como uma coleção de programas que atua como uma interface entre os programas do usuário e o hardware. Sua maior finalidade é proporcionar um ambiente em que o usuário de um microsistema possa executar programas no hardware de forma eficiente. São atribuições de um SO:

1. Gerenciar recursos e dispositivos I/O;
2. Oferecer uma simples interface para aplicativos e usuários.

Dentre os sistemas operacionais com Raspberry Pi, o mais utilizado é o Raspbian (necessário cartão SD de 4 GB). Para projetos de eletrônica, robótica, servidores e outros já que deixa livres no Raspberry Pi o máximo de recursos possível como memória, processador e consumo de corrente, está disponível também o Minibian (MINImal raspBIAN) que cabe em um cartão microSD de apenas 1 GB.

2.1 Raspberry Pi OS

O Raspberry Pi OS (anteriormente chamado Raspbian) é atualmente o sistema operacional oficial para todos os modelos de Raspberry Pi. Esta é a distribuição ideal para quem tem menos conhecimentos dos sistemas Linux ou simplesmente necessite de um sistema operacional pronto. O Raspbian é um sistema quase completo, já que vem com diversas aplicações pré-instaladas, os drivers mais usuais, ferramentas para facilitar algumas configurações necessárias, entre outros. Muitas aplicações e módulos dedicados à programação já vêm incluídos na imagem do Raspbian, bastando iniciar o sistema para acessá-las. Para iniciantes com o Rpi, que desejam experimentar as potencialidades ou começar a programar e desenvolver projetos de sistemas embarcados, o Raspbian é o mais recomendado. No endereço http://downloads.raspberrypi.org/raspbian/images/,

é possível acessar todas as versões de imagens de Raspbian até hoje. A Figura 2.1 abaixo mostra a tela gerada pelo Raspberry para interface com o usuário.

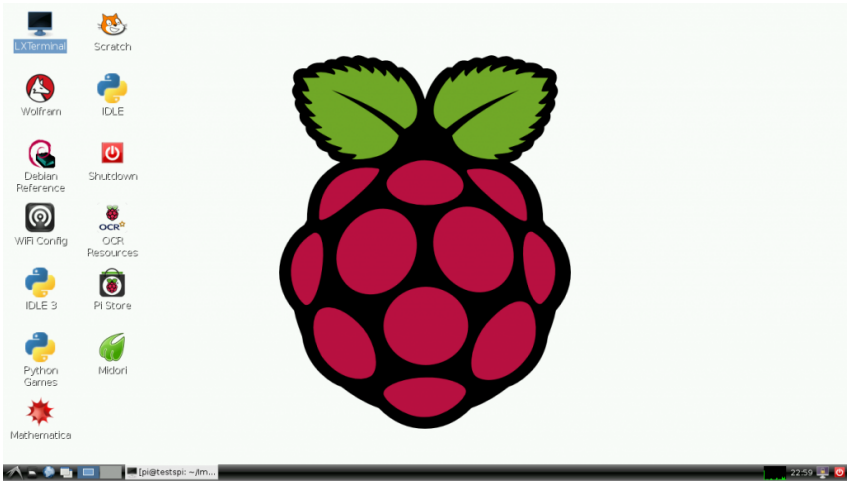


Figura 2.1: Tela gerada pelo Raspberry para interface com o usuário

2.2 Minibian

O **Minibian** é uma versão "enxuta" do **Raspbian**. Destina-se àqueles que precisam de um sistema o mais leve possível e sem ferramentas e aplicações desnecessárias.

O **Minibian** é excelente, por exemplo, para projetos de eletrônica, robótica, servidores e outros já que deixa livres no **Raspberry Pi** o máximo de recursos possível como memória, processador, consumo de corrente, etc.

Esta distribuição não traz sequer GUI (o ambiente de janelas) e cabe num cartão de 1GB.

Se precisa de um sistema leve ou quer construir o seu próprio, o Minibian (Figura 2.2) é uma opção viável.

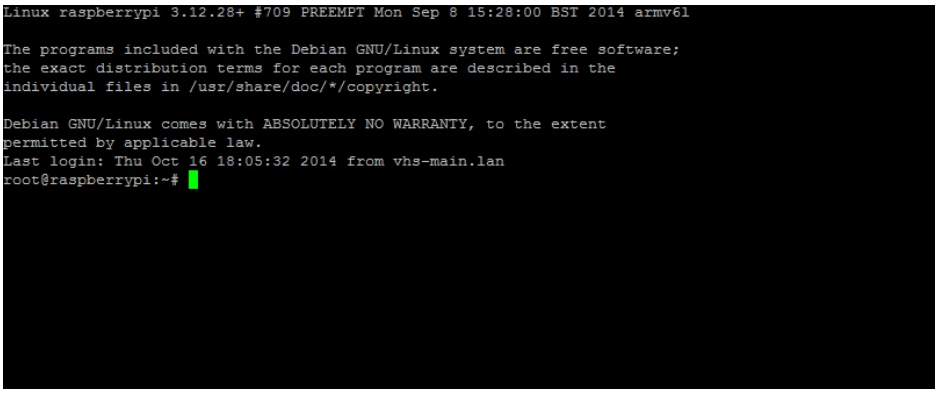


Figura 2.2: Minibian

Download: http://sourceforge.net/projects/minibian/

3. Instalação

3.1 Material Necessário

Para começar as práticas com o Raspberry Pi, segue uma lista do material necessário:

- Raspberry Pi;
- Fonte de Alimentação: 5v. Necessário ao menos 1500 mA;
- Cartão SD ou Micro SD (mínimo de 4GB).

3.2 Instalação do Software

Após baixar o Raspbian baseado em https://distrowatch.com/?newsid=09398 proveniente do site https://www.raspberrypi.org/downloads/raspbian/, instale o **Win32DiskImager** em https://sourceforge.net/projects/win32diskimager/files/latest/download e transfira a imagem do sistema operacional para SD card. Vale salientar que o Raspbian apresenta versões de atualização, em que esse material didático foi descrita e atualizada utilizando várias versões do Raspbian.

Além disso, a *Raspberry Pi Foundation* desenvolveu o Raspberry Pi Imager, disponível em https://www.raspberrypi.org/documentation/installation/installing-images/README.md, que é uma ferramenta gráfica para gravação do cartão SD e que funciona no Mac OS, Ubuntu e Windows. É a opção mais simples para a maioria dos usuários, pois faz o download da imagem e a instala automaticamente no cartão SD, como ilustrado na Figura 3.1.

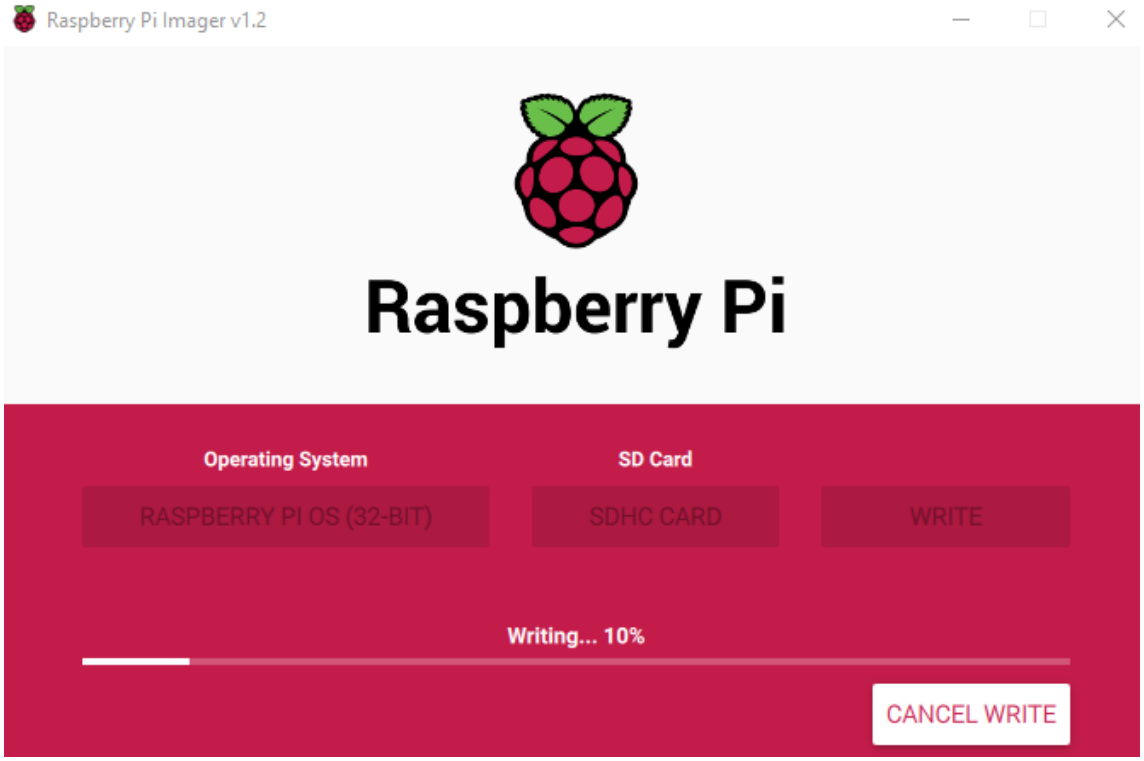


Figura 3.1: Raspberry Pi Imager.

3.3 Instalar o Sistema Operacional no cartão

- Execute como administrador clicando com botão direito no ícone Win32DiskImager (Figura 3.2).
- Em *Device*, assegure-se que seleciona a letra correspondente ao seu cartão SD;
- Clique no ícone com a imagem de uma pasta e escolha o arquivo dentro da pasta para onde extraiu o sistema operacional no passo 1 deste tutorial;
- Clique no botão "Write", aceite o aviso na janela seguinte e aguarde até terminar (Figura 3.3). Este processo irá levar alguns minutos.

Caso occra erro na gravação do SD card, formate o cartão integralmente utilizando o SD Memory Card Formatter e selecione a opção *overwrite format* (formatação completa).

3.4 Habilitando o SSH

O SSH (*Secure SHell*) é um protocolo de rede criptográfico para operação de serviços de rede de forma segura. Esse protocolo que permite a você acessar virtualmente o servidor como se estivesse em um terminal.

Usando o gerenciador de arquivos do Windows, abra a unidade do cartão microSD com nome "boot". Dentro dessa unidade crie um arquivo sem extensão chamado "ssh". clicando com o botão

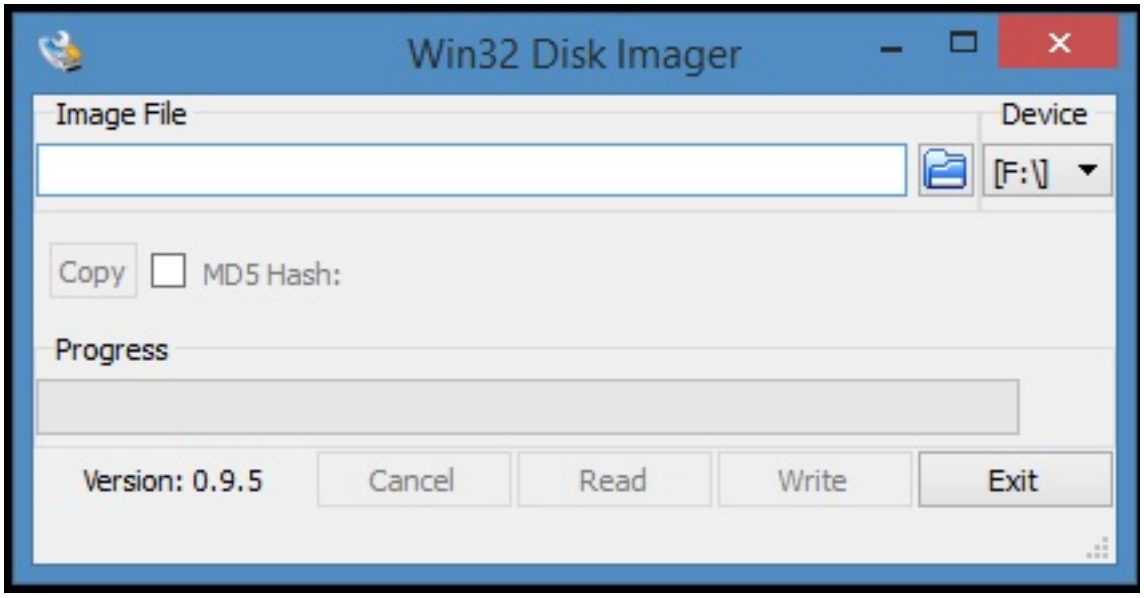


Figura 3.2: Win32DiskImager.

direito do mouse, indo em "Novo -> Arquivo de Texto". Aí no novo arquivo criado, renomeie para ssh e apague a extensão .txt, como ilustrado na Figura 3.4.

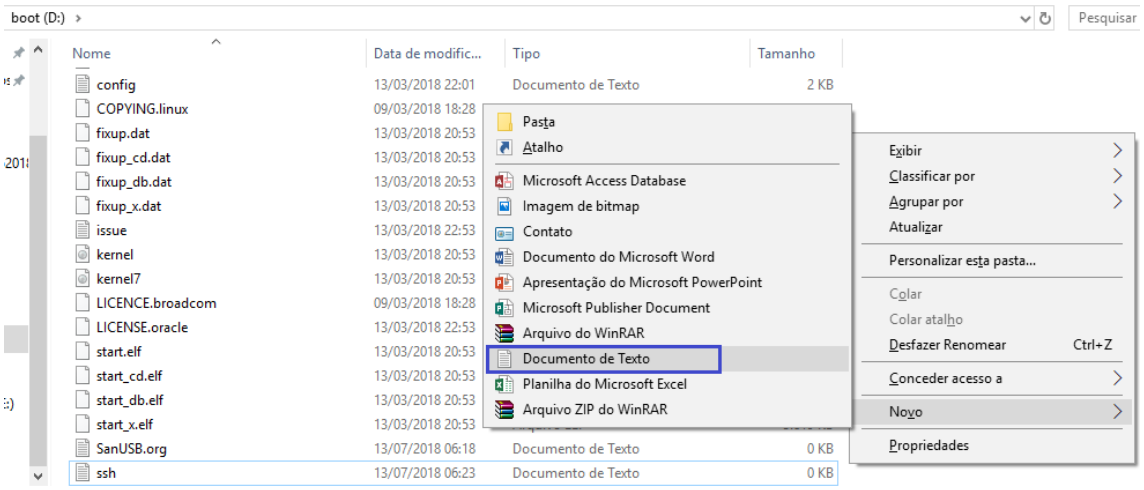


Figura 3.4: Habilitação do ssh no Windows

Caso a extensão não apareça no gerenciador de Arquivos do Windows Explorer, clique em Exibir e selecione a opção Extensões de nomes de arquivos como na Figura 3.5.

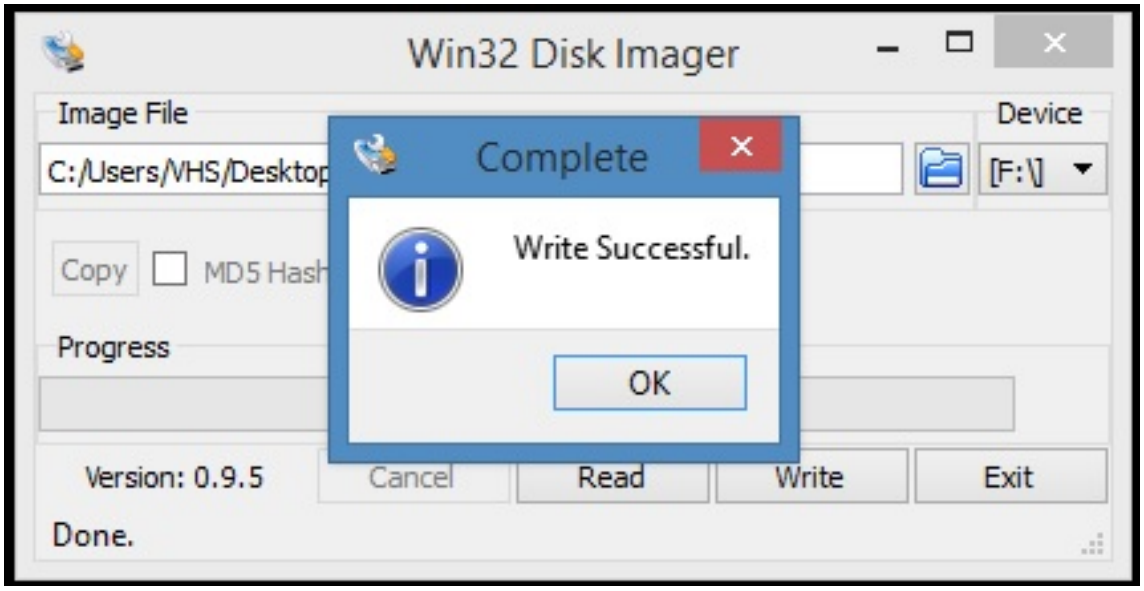


Figura 3.3: Confirmação de gravação da imagem no cartão SD.

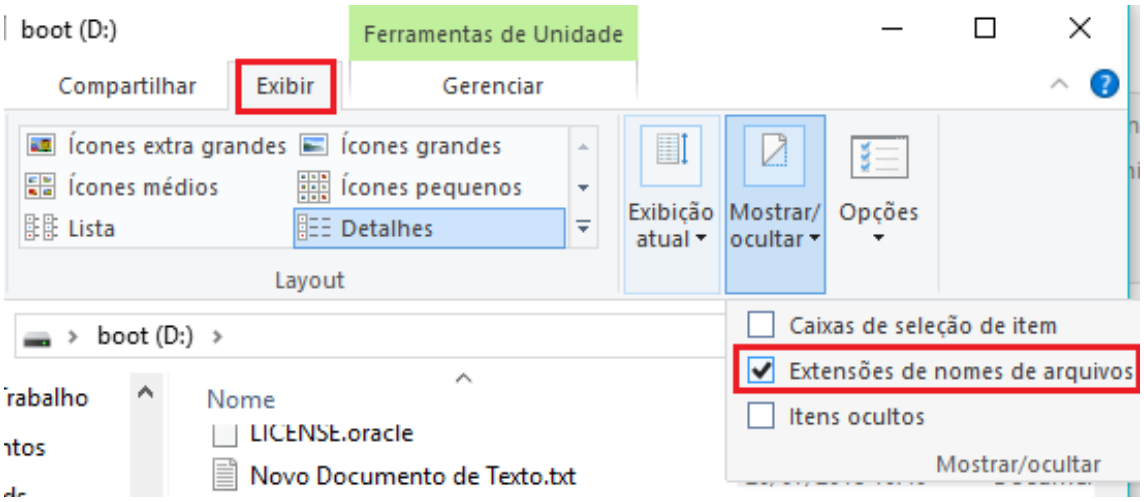


Figura 3.5: Exibir extensão de arquivos

O mesmo procedimento pode ser realizado em um computador no Linux para criar um arquivo "ssh"sem extensão na raiz boot, como mostrado na Figura 3.6.

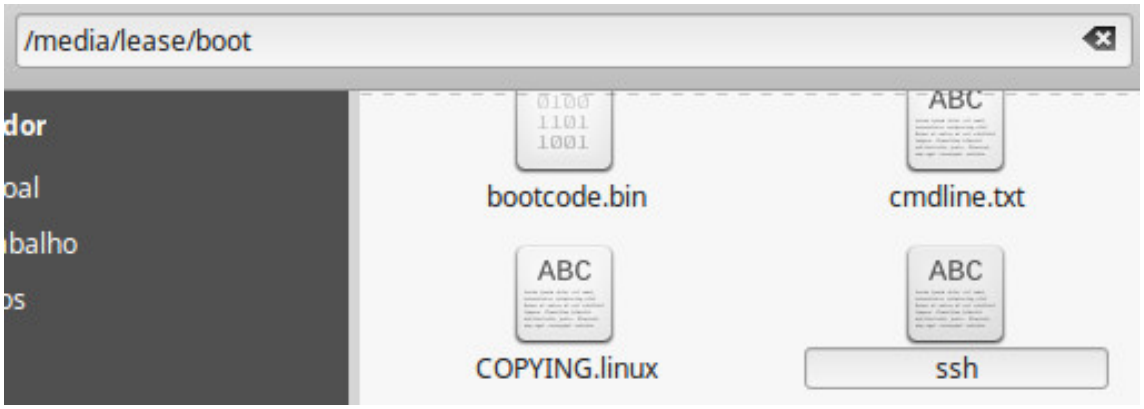


Figura 3.6: Habilitação do ssh no Linux

Outra forma é através do Raspi-Config. Conecte um teclado na porta USB do Rpi e um cabo HDMI para um monitor ou TV para configurar o Rpi no sistema operacional. Para isso, abra uma janela de terminal e digite o seguinte comando:

sudo raspi-config

A tela principal do Raspi-Config será aberta (Figuras 3.7, 3.8, 3.9). Selecione a opção 9 - Advanced Options:

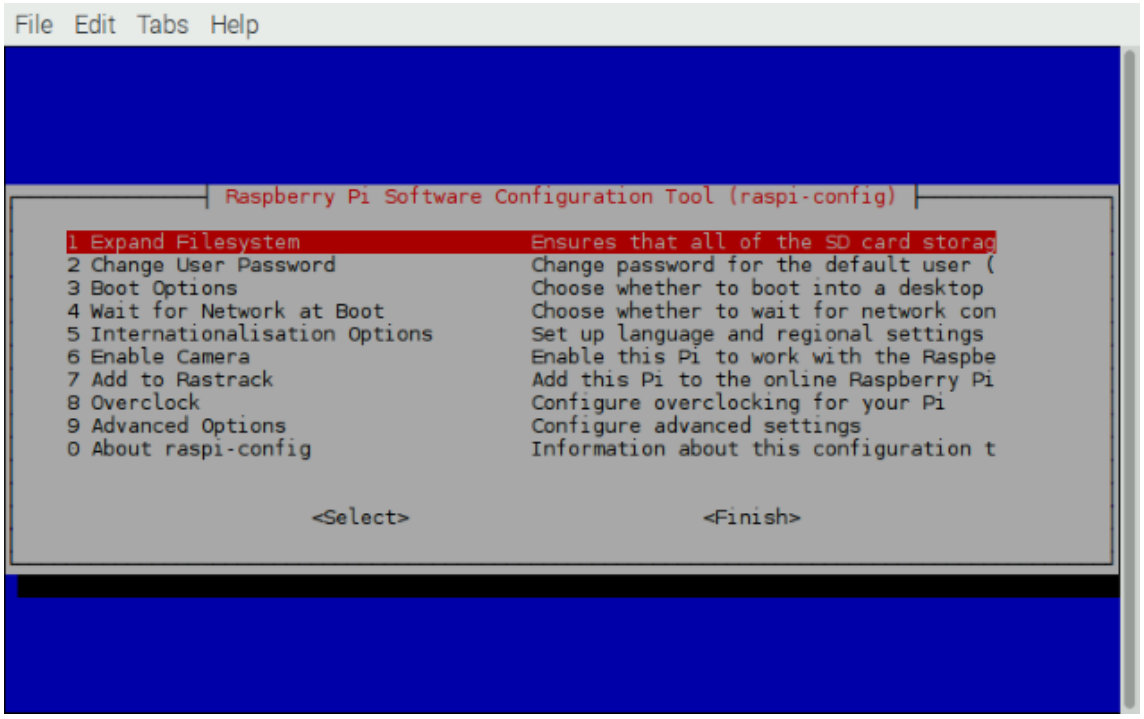


Figura 3.7: Tela do RaspiConfig

Em seguida escolha a opção A4 SSH:

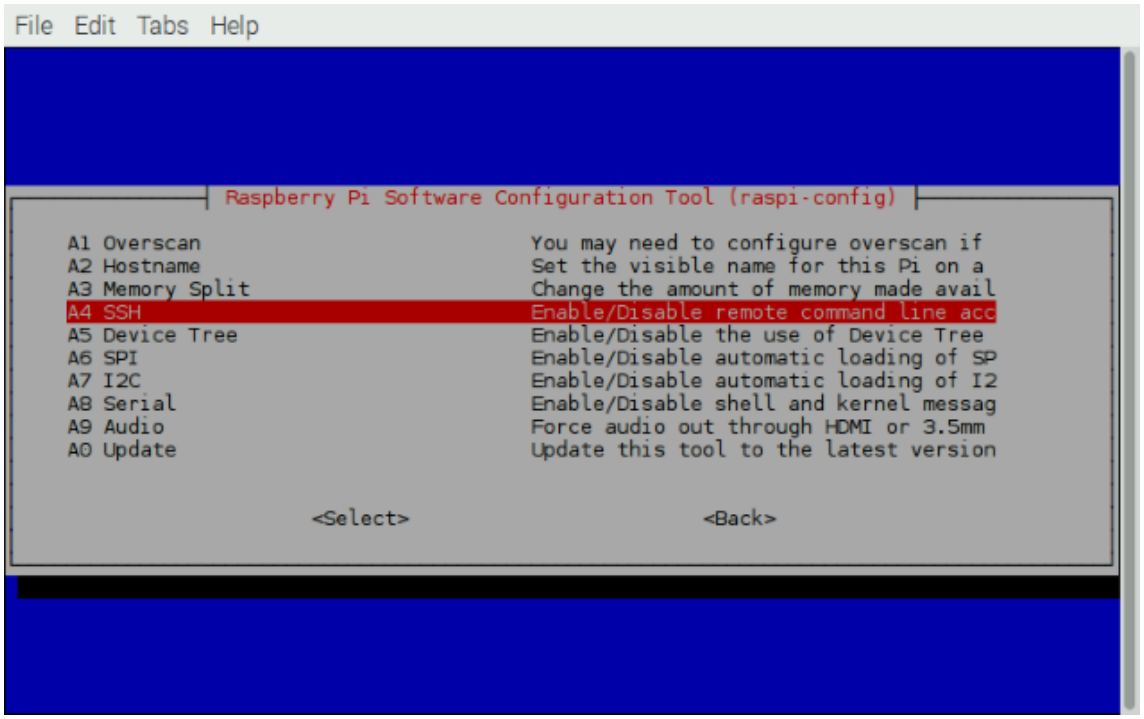


Figura 3.8: Tela do RaspiConfig

Selecione Enable para habilitar o SSH no Raspbian:

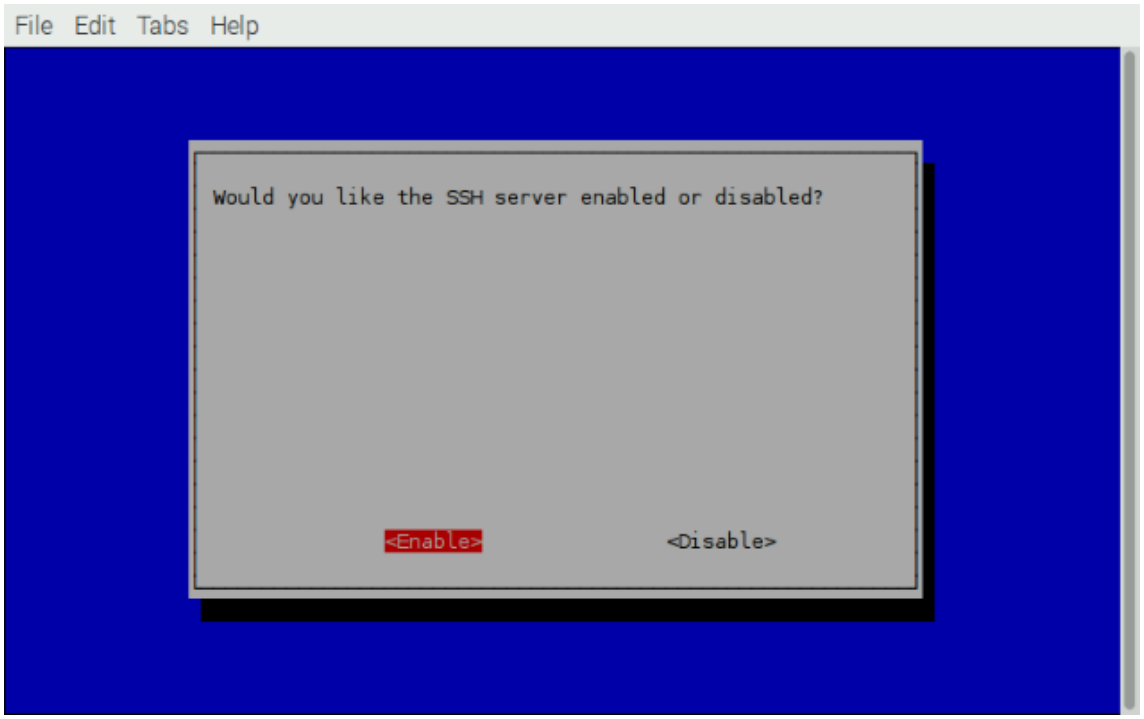


Figura 3.9: Tela do RaspiConfig

3.5 Varredura e conexão com Redes WiFi

Como a comunicação e a programação do Rpi é realizada em modo online, é necessário que o Rpi tenha acesso às interfaces Ethernet e/ou Wifi. Dessa forma, para que o Rpi possa verificar possíveis redes Wifi (selecionadas antes da interface Ethernet), é possível fazer uma varredura nas redes WiFi com o comando:

```
sudo iwlist wlan0 scan
```

Depois de encontradas, é necessário inserir as configurações de uma ou mais redes WiFi no final do arquivo com **nano /etc/wpa_supplicant/wpa_supplicant.conf** como abaixo.

```
#Arquivo: /etc/wpa_supplicant/wpa_supplicant.conf

network={
ssid="SSID-HERE"
psk="WIFI-PASSWORD-HERE"
}
```

Caso a rede esteja oculta, basta inserir **scan_ssid=1**, como abaixo, para não varrer as possíveis redes.

```
network={
ssid="LAESE"
scan_ssid=1
psk="********"
}
```

É possível configurar a conexão WiFi do Rpi através da plataforma de serviço gerada automaticamente em **sanusb.org/rs/sanusb.php** em que ao executar o script *Sloader.sh* no Rpi, basta acessar o link *Commands*, inserir o comando wf e prrencher o SSID e o password nos campos indicados. Assim, *Sloader* irá adicionar a nova rede WiFi automaticamente.

NOTA:

Para ter acesso direto ao arquivo **wpa_supplicant.conf** e configurar a rede Wifi do Rpi imediatamente após a gravação do S.O. Raspbian, sem a necessidade anterior de conexão SSH, Ethernet ou cabo HDMI, que é **situação comum para um Rpi zero W**, por exemplo, basta abrir o SD card novamente pelo gerenciador de arquivos, criar um arquivo **wpa_supplicant.conf**, também disponível em https://github.com/SanUSB/SmallestRpiSetup e inserir a configuração Wifi abaixo como mostra a Figura 3.10. Mais detalhes em: https://youtu.be/ZmEgfkwnEcU.

```
country=BR
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
```

```
  ssid=" "
  psk=" "
}
```

A instrução ctrl_interface=DIR=/var/run/wpa_supplicant cria um link simbólico e replica o conteúdo do arquivo wpa_supplicant do *boot* para /var/run/wpa_supplicant e para /etc/wpa_supplicant.

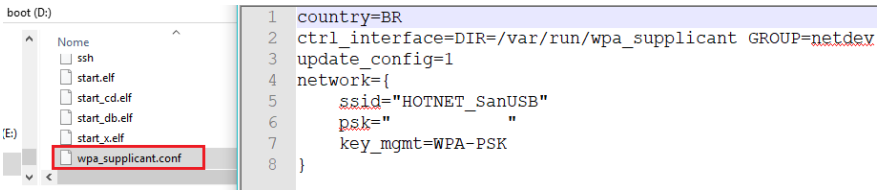


Figura 3.10: Acesso direto pelo SD card ao arquivo wpa_compplicant.conf no Windows

Em uma máquina Linux, ao inserir o SD card na porta USB ou SD do PC, será montado um drive *rootfs* que permite o acesso ao arquivo wpa_compplicant.conf pelo terminal através do caminho ilustrado na Figura 3.11, com o comando:

```
nano /media/USUARIO/rootfs/etc/wpa_supplicant/wpa_supplicant.conf
```

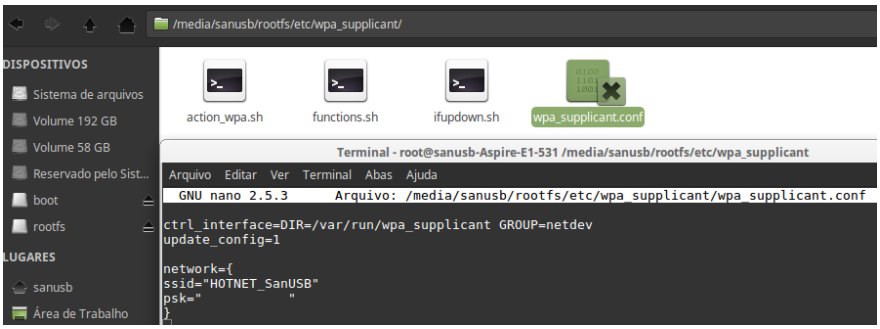


Figura 3.11: Acesso direto pelo SD card ao arquivo wpa_compplicant.conf no Linux

A Figura 3.12 mostra também que é possível a inserção em sequência de duas redes Wifi preconfiguradas via plataforma de serviço que serão selecionadaspara conexão de cima para baixo em sequência. Se não houver modem WiFi, mesmo com as redes WiFi preconfiguradas, a conexão será estabelecida com a rede Ethernet após reinicialização.

3.6 Acesso Remoto SSH (Secure SHell) por linha de comando

É possível utilizar o protocolo SSH, *default* no Raspbian, para acessar o Raspberry Pi de um computador remoto é necessário que através de um programa, como o Putty, conectado na porta 22. O Putty é um programa de fácil execução, sendo necessário somente inserir o IP do Rpi, além de

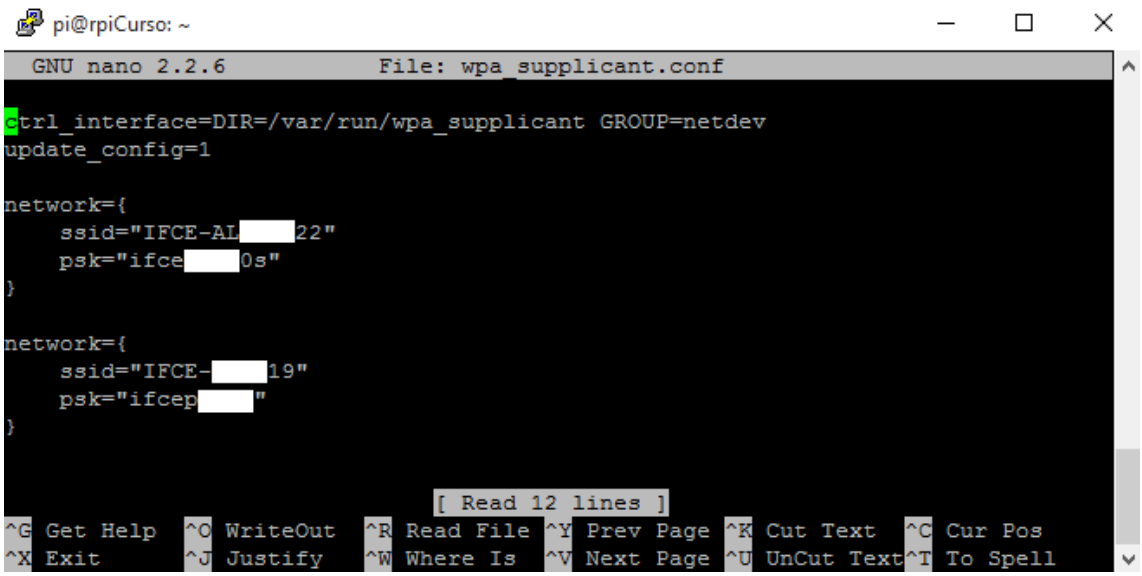


Figura 3.12: Inserção em sequência de duas redes Wifi

satisfazer qualquer usuário. Para fazer download do programa clique no link http://www.putty.org ou em http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

Para verificar o IP do Rpi cadastrado na rede, pode ser inserido o seguinte comando no terminal:

```
ifconfig wlan0
```

O IP estará contido na indicação INET, como por exemplo, INET 192.168.1.21. Também é possível utilizar um software de busca livre e multiplatafoma com interface gráfica chamado *Angry IP Scanner* em http://angryip.org. Para os IPs ativos, basta selecionar em preferências, indicado com um círculo vermelho na Figura 3.13 e clicar em *Alive hosts (responding to pings) only*.

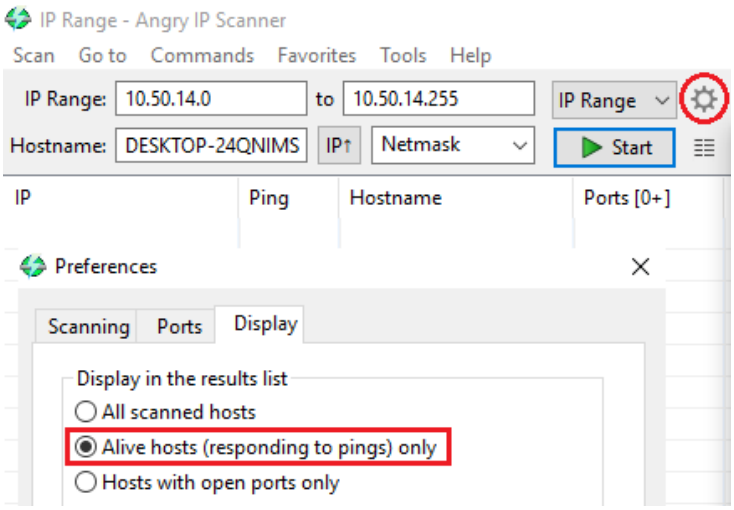


Figura 3.13: Ips ativos com *Angry IP Scanner*

Ao abrir o programa, é aberta uma janela como essa da Figura 3.14. É possível colocar nesta janela as informações de conexão como endereço IP (no nosso caso, 192.168.0.31), a porta (22) e seleciona também o tipo de conexão (SSH).

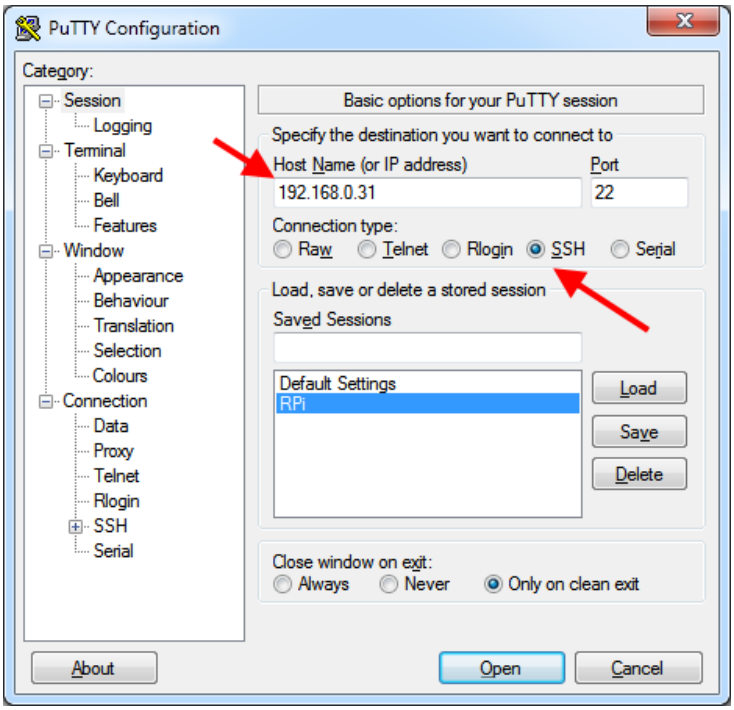


Figura 3.14: Configurações do PuTTY

3.7 Acesso remoto SSH com IPV6 local fixo

O principal motivo para a implantação do IPv6 na Internet é a necessidade de mais endereços, porque a disponibilidade de endereços livres IPv4 acabou. O endereçamento no IPv6 é de 128 bits (o quádruplo do IPv4) e é normalmente escrito como grupos de 4 dígitos hexadecimais. Por exemplo, fe80::abb5:6bbf:29a5:fa28.

O número ipv6 é derivado do endereço físico MAC, dessa forma, ele garante um endereço de IP fixo para o Rpi em qualquer rede local, mesmo que o DHCP da rede, para atribuição de IPs dinâmicos, esteja ativo.

Para acesso remoto do Rpi, por exemplo de um terminal Linux, basta digitar os comandos abaixo:

Listing 3.1: Acesso SSH com ipv6

```
#Conectar ao ipv6 do raspberry pi pelo terminal

#encontrar interface de rede do computador remoto, por exemplo, enp2s0 e
    o ipv6 do Rpi com ifconfig, por exemplo, fe80::abb5:6bbf:29a5:fa28.
```

```
ifconfig

#digitar: ssh -6 usuario @ ipv6 % interface de rede do computador (tudo
    junto)
# -6 solicita conecão por ipv6

ssh -6 pi@fe80::abb5:6bbf:29a5:fa28%enp2s0

#depois de conectado basta digitar o usuário, por exemplo, pi e a senha,
    por exemplo, raspberry.
```

A partir da versão Raspbian stretch é possível utilizar somente *ssh -6 pi@<ipv6>* , como por exemplo, *ssh -6 pi@fe80::d12a:9c1d:47f4:ed87*.

3.8 Shell script e o interpretador de comandos Bash

shell é uma interface de usuário, chamado de shell, ou seja, de casca, porque ele é a camada mais externa em torno do núcleo do sistema operacional.

Por outro lado, shell script é uma linguagem de script usada em vários sistemas operacionais, com diferentes dialetos, dependendo do interpretador de comandos.

Nesse sentido, bash (ou GNU Bourne Again SHell) é o interpretador de comandos mais utilizado nas distribuições Unix/Linux que apresenta recursos e características de uma linguagem de programação de alto nível.

Existem outros interpretadores de comandos, como o bash, que são o CMD prompt e o PowerShell do Windows. Por isso que ao criar scripts é recomendável adicionar o header indicando qual interpretador está sendo utilizado. Exemplo:

!/bin/bash

[comandos ...]

A diretiva *!/bin/bash* indica que os comandos são feitos para serem executados com o interpretador bash.

3.9 Instalação de programas iniciais necessários

Após instalar o Raspbian basta seguir a sequência básica:

1. Conectar o Rpi na rede (network), Abrir o terminal SSH putty e digitar o IP do Rpi.
2. Após entrar no terminal shell, basta digitar:

```
# entrar como root
```

```
sudo su
# baixar o script de instalacao
wget sanusb.org/arquivos/SanUSBlink.sh
# Permitir execucao
chmod 777 SanUSBlink.sh
# executar o script de instalacao
./SanUSBlink.sh
```

Pronto. Será instalado automaticamente todos os programas iniciais necessários. Recomendo que façam em casa. Caso a rede esteja muito lenta pode ocorrer a falha de instalação em algum dos programas.

Listing 3.2: Instalação dos programas necessários para as práticas desse material didático

```
#!/usr/bin/env bash

# install curl
if [ ! -f "/usr/bin/curl" ] && [ ! -f "/bin/curl" ]; then
    apt-get upgrade
    apt-get update
    apt-get install curl
fi

# install sanusb (sanusb.org)
if [ ! -f "/usr/share/sanusb/sanusb" ]; then
    mkdir /home/share/
    cd /home/share
    wget http://sanusb.org/tools/SanUSBrpi.zip
    unzip SanUSBrpi.zip
    cd SanUSBrpi
    sudo dpkg -i sanusb_raspberry.deb
    chmod +x sanusb
    cp sanusb /usr/share/sanusb
fi
# install Wpi
if [ ! -f "/usr/include/wiringPi.h" ]; then
    apt-get install wiringpi
fi


# install serial minicom e serialconfigtest
if [ ! -f "/etc/inittab.out" ]; then
```

```
    sudo apt-get install minicom
    wget sanusb.org/tools/serialconfigtest.deb
    dpkg -i serialconfigtest.deb
fi

# install Samba #smbd.service must last to start
if [ ! -f "/etc/samba/smb.conf.old" ]; then
    kill $(ps | grep serialtest | cut -d' ' -f2)
    apt-get upgrade
    apt-get update
    apt-get install samba samba-common-bin
    mkdir /home/share/
    cd /home
    chmod 777 share
    cd /etc
    chmod 777 samba
    cd /etc/samba
    mv smb.conf smb.conf.old
    wget http://sanusb.org/arquivos/smb.zip
    unzip smb.zip
    service smbd restart
fi
```

3.10 Raspberry Pi – Como fazer um backup do seu sistema

Depois de ter o seu **Raspberry Pi** com o sistema operacional ajustado de acordo com as suas necessidades e preferências, a última coisa que vai querer é ter que recomeçar do zero caso falhe o cartão SD, por exemplo. Felizmente é muito fácil fazer uma cópia de segurança completa do cartão e repô-la sempre que necessário. Em poucos passos vamos criar um arquivo com a imagem do cartão que inclui todo o sistema operacional e mostrar-lhe como pode facilmente utilizar essa imagem para replicar o sistema noutros **Raspberry Pi** ou recuperar o seu de um acidente. Para este tutorial vamos utilizar um PC com sistema operacional Windows e o programa grátis "Win32 Disk Imager" em http://sourceforge.net/projects/win32diskimager/

3.11 Criação do arquivo de imagem do sistema

- Se ainda não instalou o Win32 Disk Imager faça-o agora clicando no instalador.
- Coloque o cartão com o sistema operacional do seu **Raspberry Pi** no leitor de cartões e ligue ao PC.
- Inicie o Win32 Disk Imager, clicando com o botão direito no ícone e executando como

administrador. A partir daí irá surgir a tela inicial como ilustrado na Figura 3.15

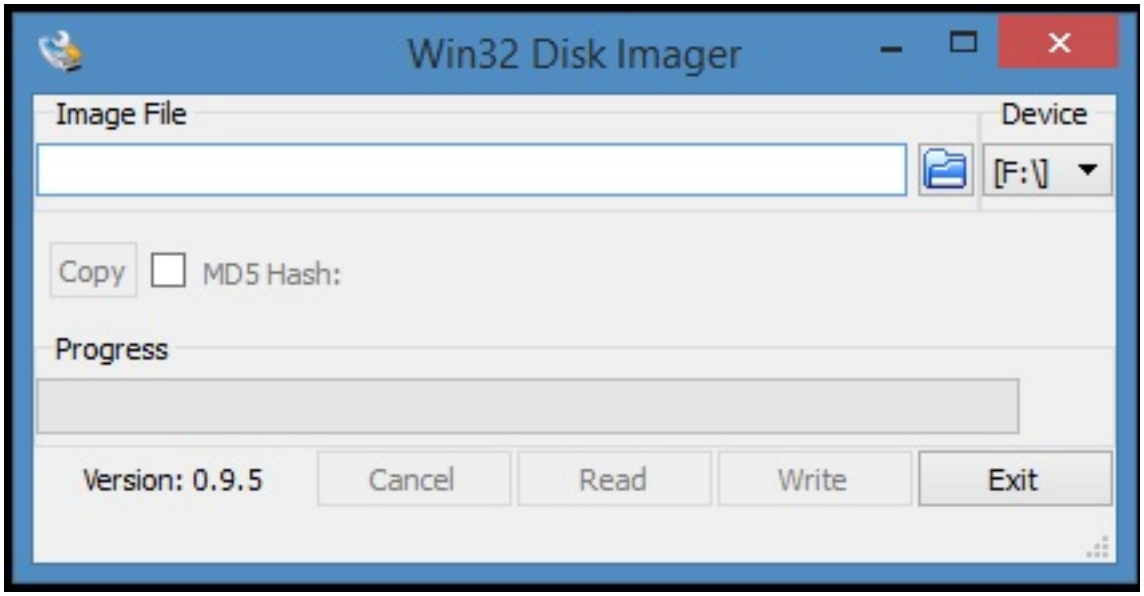


Figura 3.15: Win32DiskImager

- Em *"Device"* selecione a letra correspondente ao seu cartão SD do qual quer fazer backup.
- Clique no ícone com uma pasta, selecione uma localização para guardar o arquivo a ser criado, atribua-lhe um nome e clique em "Abrir" (Figura 3.16)

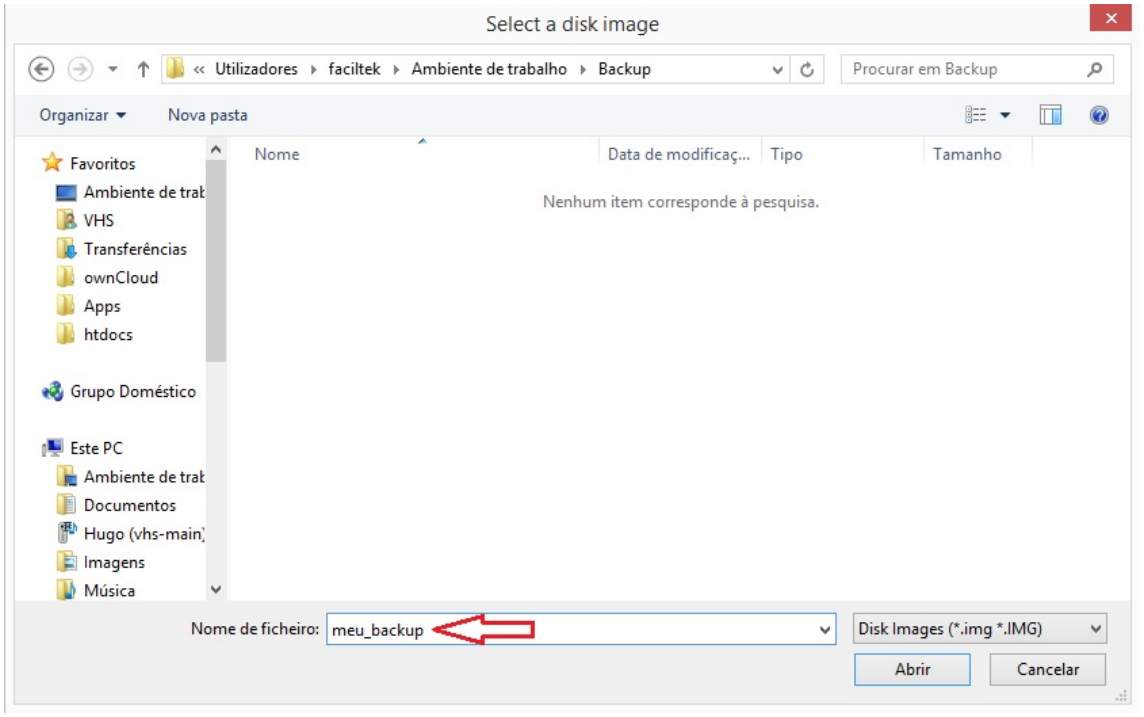


Figura 3.16: Abrir Local do Arquivo

- Finalmente clique no botão *"Read"* (Figura 3.17), confirme e aguarde o processo ser concluído.

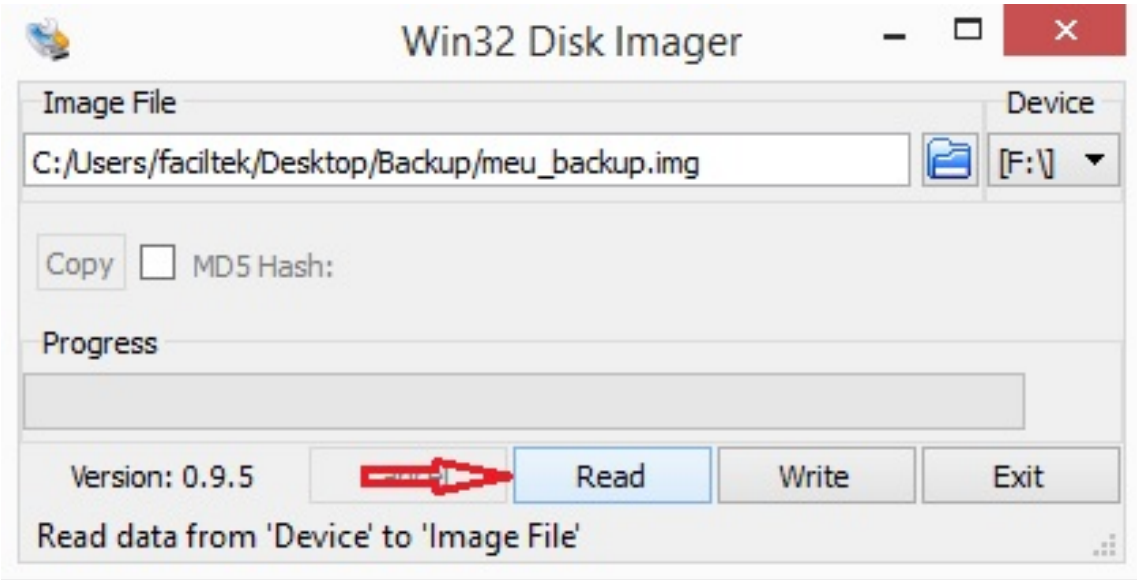
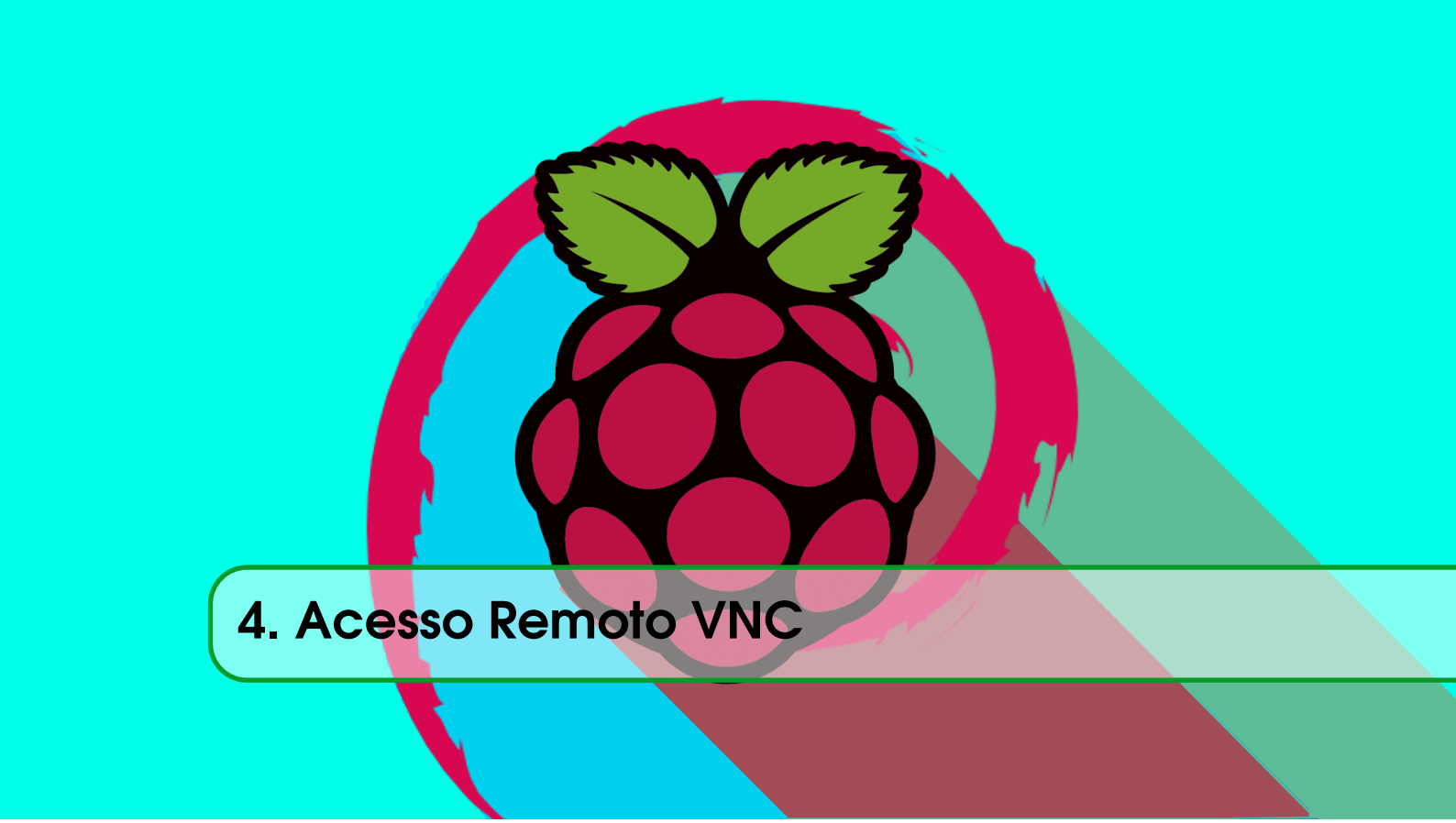


Figura 3.17: Concluir backup.

- Abra a pasta que selecionou para guardar o backup e confirme, na pasta selecionada, se o arquivo foi criado.



4. Acesso Remoto VNC

4.1 Acesso a um Raspberry remoto via interface gráfica

Gerenciar um Raspberry Pi (Rpi) por linhas de comando através de interface SSH, como o programa Putty, é uma forma eficiente de acesso remoto. Caso seja necessário utilizar interface gráfica, a versão Jessie normal já dispõe para computadores com S.O. Linux um modo de acesso gráfico automático através do comando.

Outra forma possível para máquinas Windows ou *smartphones*, é através do protocolo VNC (*Virtual Network Computing*) utilizando o programa RealVNC, para versões mais recentes a partir do Raspbian Stretch permitindo inclusive acesso através de aplicativos de smartphones, e também o programa tightvnc.

Para instalar o RealVNC, execute os seguintes comandos para garantir que a versão mais recente do VNC:

```
sudo apt update
sudo apt install realvnc-vnc-server realvnc-vnc-viewer
```

É possível habilitar o servidor VNC graficamente ou por linha de comando. Para habilitar graficamente, inicialize a área de trabalho do Rpi e selecione **Menu > Preferences > Raspberry Pi Configuration > Interfaces**, e verifique se o VNC está ativado.

Para habilitar o servidor VNC por linha de comando, utilize o comando

```
sudo raspi-config
```

Agora, é necessário ativar o servidor VNC seguindo os seguintes passos:

Navegue para **Interfacing Options**.

Role para baixo e selecione **VNC > Yes**.

Para estabelecer a conexão VNC, encontre o IP local do Rpi executando *ifconfig* no terminal ou utilize o aplicativo *Advanced IPScanner*. No computador ou *smartphone* onde será realizada a visualização gráfica, instale o RealVNC disponível em https://www.realvnc.com/pt/connect/download/viewer/. Digite o endereço IP do seu Raspberry Pi no RealVNC e pronto, já é possível visualizar e controlar o Rpi remotamente.

Se aparecer o erro "*Cannot currently show desktop*", digite novamente no terminal ***raspi-config*** *-> Advanced Options -> Resolution -> DMT Mode 82 1920 x 1080 60Hz 16:9* (selecione a maior resolução). Para habilitar as configurações digite ***reboot*** no terminal.

Mais detalhes em:

https://www.raspberrypi.org/documentation/remote-access/vnc/.

Já o TightVNC é um software livre de código aberto (GPL2) que fornece compressão para permitir que ele trabalhe sobre redes lentas de Internet e está prontamente disponível nos repositórios de software (sudo apt-get install tightvncserver) para a versão Raspbian Jessie normal (**não funciona na versão Lite**).

O VNC segue o modelo cliente-servidor tradicional. O software do servidor é executado no host e software cliente é executado na máquina local que deseja controlar o host.

4.2 Instalando o software do servidor TightVNC

Primeiro é recomendável atualizar as informações do repositório de software digitando no terminal:

- **sudo apt-get update**

Instale o software do servidor a partir dos repositórios **como usuário pi (sem ser usuário root, ou seja, sem digitar sudo su)**. Se estiver como root, basta digitar exit no terminal.

- **sudo apt-get install tightvncserver**

4.3 Iniciando o servidor e configurando uma senha

```
# Digite o codigo abaixo no terminal para iniciar o servidor.
vncserver :1
# Depois digite o comando abaixo pedir para definir uma senha de 8
   digitos (padrao 12345678).
vncpasswd
# Crie um arquivo de configuracao
nano /etc/systemd/system/vncserver@.service
```

Com o seguinte conteúdo:

```
[Unit]
Description=Remote desktop service (VNC)
After=syslog.target network.target

[Service]
Type=forking
User=pi
PAMName=login
PIDFile=/home/pi/.vnc/%H:%i.pid
ExecStartPre=-/usr/bin/vncserver -kill:%i > /dev/null 2>&1
ExecStart=/usr/bin/vncserver -depth 24 -geometry 1280x800:%i
ExecStop=/usr/bin/vncserver -kill:%i

[Install]
WantedBy=multi-user.target
```

Indique que existe um novo serviço daemon e inicie o serviço vnc: `sudo systemctl daemon-reload sudo systemctl enable vncserver@1.service`

Reinicie o Rpi com o comando reboot e verifique se o serviço vnc está ativo com o comando: `sudo systemctl is-enabled vncserver@1.service`

Para conectar com o cliente remoto Windows, depois que reiniciar o Rpi, execute o comando abaixo manualmente no terminal: `tightvncserver`

Aparecerá a indicação da configuração do serviço vnc e a tela disponível 1.

```
pi@raspberrypi:~ $ sudo su
root@raspberrypi:/home/pi# tightvncserver

New 'X' desktop is raspberrypi:1

Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/raspberrypi:1.log
```

Figura 4.1: Configuração do TightVNCserve

Finalmente, acesse o Rpi pelo Windows com o programa TightVNCclient, digitando o IP:5901, exemplo: `Remote Host:192.168.25.8:5901`.

Cada vez que se inicia o tightvncserver, ele captura o próximo Desktop disponível, que no nosso caso remoto automático será o 1. Quando for requerido uma senha durante a instalação, digite uma senha de 8 dígitos no terminal como, por exemplo 12345678.

Essa senha será a mesma utilizada quando conectar no cliente Windows.

Para instalar o tightvnc no computador cliente Windows, basta baixá-lo em TightVNC.com. Após instalar o cliente Windows, aparecerá a interface da Figura 4.3

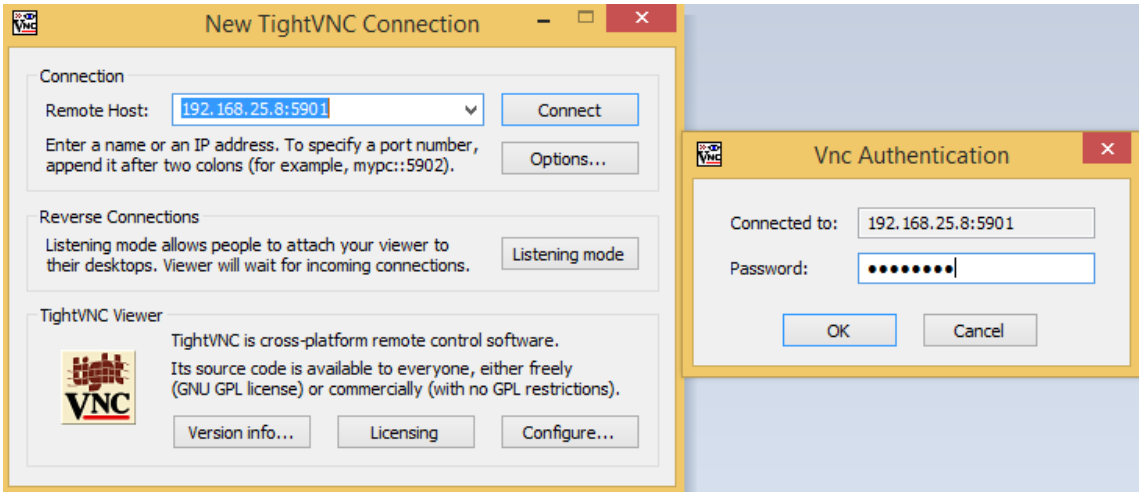


Figura 4.2: TightVNC

Note que a porta VNC é 5901, ou seja, o desktop virtual instalado será o 1. Ao clicar em Connect, basta digitar a mesma senha cadastrada na instalação no Rpi (12345678). Para modificar o Desktop virtual basta digitar **tightvncserver** no terminal SSH, e será mostrado qual o Desktop disponível para conexão no cliente Windows.

4.4 Script de do tightvnc para inicialização automática na versão Raspbian Wheezy

Para ter a inicialização automática do tightvnc, é possível executar os seguintes passos para instalação como usuário pi(sair do modo root –> exit):

```
#Baixar o script de instalacao
pi@raspberrypi ~ $ sudo wget sanusb.org/arquivos/VncConfig.sh
#Permitir execucao
@raspberrypi ~ $ sudo chmod 777 VncConfig.sh
#Entra na pasta que contem o VncConfig.sh pelo usuario pi (se estiver
    root digitar exit) e executar o script de instalacao:
@raspberrypi ~ $ ./VncConfig.sh
```

Abaixo o script *VncConfig.sh*.

```
#!/usr/bin/env bash
# Install as pi user (exit -without sudo su)
```

```
  sudo apt-get update
sudo apt-get install tightvncserver
  sudo wget http://sanusb.org/tools/VncConfigUserPi.deb
  sudo dpkg -i VncConfigUserPi.deb
  sudo reboot
```

Para instalar manualmente basta seguir passo a passo as linhas comando abaixo:

- **wget http://sanusb.org/vnc/tightvncserver-init.txt**

A partir das linhas de comando abaixo, scripts ou arquivos binários podem ser configurados para tornar-se Default no Raspbian:

- **sudo mv tightvncserver-init.txt /etc/init.d/tightvncserver**

A variável da linha 23, que define o Desktop virtual, foi configurada como :2, ou seja, na porta 5902. O valor original é :1 Altere o arquivo para que ele seja de propriedade root (é a propriedade padrão para arquivos de inicialização):

- **sudo chown root:root /etc/init.d/tightvncserver**

Permita que o arquivo seja executável:

- **sudo chmod 755 /etc/init.d/tightvncserver**

Adicione o comando abaixo para habilitar a execução no início do sistema operacional:

- **sudo update-rc.d tightvncserver defaults**

O TightVNC está instalado e vai ser carregado na inicialização do Raspbian. Neste momento, é recomendado a reinicialização (reboot) para se certificar de que está funcionando, mas é possível simplesmente parar e iniciar manualmente digitando:

- **sudo /etc/init.d/tightvncserver stop**
- **sudo /etc/init.d/tightvncserver start**

4.5 Mudar senha do VNC

O VNC só aceita senha com oito caracteres, caso seja digitada uma senha com mais de oito caracteres o vnc irá considerar somente os oito primeiros.

```
pi@raspberrypi ~ $ vncpasswd
Using password file /home/pi/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
pi@raspberrypi ~ $
```

```
sudo su
reboot
```

5. Instalação de servidores de arquivos

Nesse projeto, é demonstrado a instalação e configuração de um software livre servidor de arquivos em rede para auxiliar na comunicação, registro e compartilhamento de informações em segurança. Para exemplificar um servidor convencional que pode ser utilizado de forma gratuita é descrito inicialmente o servidor arquivos Samba, em que os arquivos compartilhados podem ser visualizados em computadores Windows, Linux e Mac OSX, bem como em *smartphones*, quando for utilizado o segundo serviço proposto em software livre chamado de *Owncloud.*

5.1 Configurando endereço de IP fixo

Caso seja necessário que o servidor tenha um endereço de IP fixo, é necessário considerar os seguintes conceito: Os endereços IPv4 consistem em endereços de 32 bits divididos em 4 octetos e uma máscara de sub-rede do mesmo tamanho. É importante salientar que máscara de sub-rede é um número de 32 bits usado em um IP para separar a parte correspondente à rede e aos hosts, de acordo com as classes, como ilustra a Figura 5.1

Assim, a quantidade de endereços IP de redes e de hosts que podem ser conectados a essas redes depende máscara de sub-rede. A notação da máscara de sub-rede utilizada normalmente é a CIDR (abreviação de *Classless Inter-Domain Routing*) que identifica a quantidades de bits da máscara de sub-rede de tamanho variável que pode ser /8, /16 e /24 bits, conforme a Figura 5.2

Para configurar um IP fixo de um Rpi, digite ***ifconfig*** no terminal e verifique as informações:

```
interface {interface de rede que está utilizando}
static ip_address={seu futuro ip fixo}/{máscara de rede (CIDR)}
```

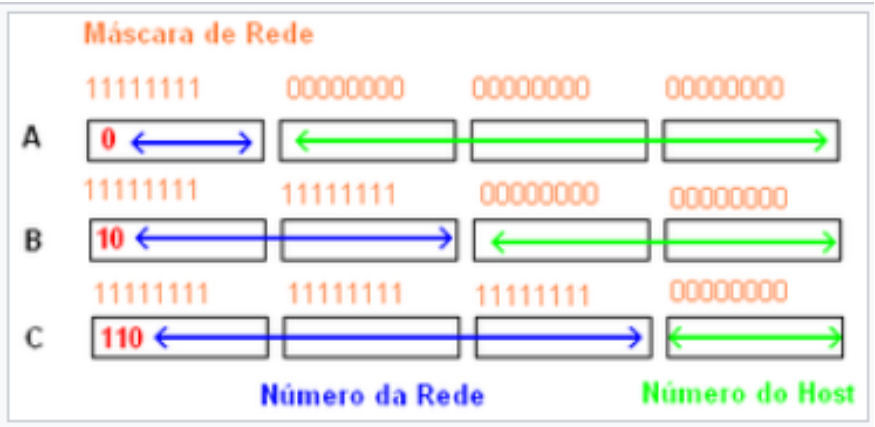


Figura 5.1: Máscara de rede.

Classe	Bits iniciais	Início	Fim	Máscara de sub-rede padrão	Notação CIDR
A	0	1.0.0.1	126.255.255.254	255.0.0.0	/8
B	10	128.0.0.1	191.255.255.254	255.255.0.0	/16
C	110	192.0.0.1	223.255.255.254	255.255.255.0	/24

Figura 5.2: Notação CIDR.

```
static routers={ip de seu roteador}
static domain_name_servers={ip do seu roteador}
```

Depois edite o arquivo ***dhcpcd.conf***, digite no terminal: **sudo nano /etc/dhcpcd.conf**

e insira os dados de IP Fixo no início do arquivo, como no exemplo abaixo:

```
interface wlan0
static ip_address=192.168.11.9/24
static routers=192.168.11.1
static domain_name_servers=192.168.11.1
```

5.2 O servidor Samba

O SAMBA é um servidor de arquivos remotos compatíveis com o Windows. Com ele é possível compartilhar arquivos e possibilitar a ediçãod e programas. Para essa instalação iremos usar como sistema operacional no Raspberry Pi o Raspbian, a versão embarcada do Debian.

Para não ter que ficar usando o 'sudo' a cada comando digite:

sudo bash ou **sudo su**

Para instalar automaticamente, execute o script em `http://sanusb.org/arquivos/SanUSBlink.sh`. Para instalar manualmente pelo terminal, digite a seguinte linha de comando para baixar o SAMBA:

apt-get install samba samba-common-bin

Confirme a instalação quando solicitado.

Vamos configurar um compartilhamento, usaremos o sub-diretório home para isso.

Crie o sub-diretório share dentro do diretório home:

mkdir /home/share/

Dê permissões totais para sua pasta share (digitar cd /home):

chmod 777 share

Note que esse tipo de ação pode comprometer a segurança do seu sistema, válido apenas para testes gerais e fins didáticos.

Crie uma cópia de segurança do arquivo de configurações original do samba. Com isso em qualquer necessidade podemos voltar às configurações originais:

cp /etc/samba/smb.conf /etc/samba/smb.conf.old

Edite o arquivo /etc/samba/smb.conf:

nano /etc/samba/smb.conf

Vá com o cursor até o final do arquivo **smb.conf** e insira as seguintes linhas:

```
[share]
comment = Laese Compartilhamento Geral
path = /home/share
create mask = 0777
directory mask = 0777
writable = yes
security = share
browseable = yes
public = yes
```

Depois salvar o aquivo: **Ctrl+X -> Yes -> <Enter>**

Com isso foi criado um compartilhamento chamado share que dará acesso irrestrito aos arquivos lá constantes utilizando o Samba (http://www.samba.org). Agora reinicie o serviço para que a nova configuração seja efetivada como na Figura 5.3

service smbd restart

```
root@raspberrypi:/home# service samba restart
[ ok ] Stopping Samba daemons: nmbd smbd.
[ ok ] Starting Samba daemons: nmbd smbd.
```

Figura 5.3: service samba restart.

Agora se o computador estiver no Windows, abra o Windows Explorer e na barra de endereços digite (barra invertida): \ **endereço ip do Raspberry****share**, por exemplo: \ \ **192.168.1.3****share** e adicione ou visualize arquivos compartilhados com o Raspberry como é demonstrado na Figura 5.4.

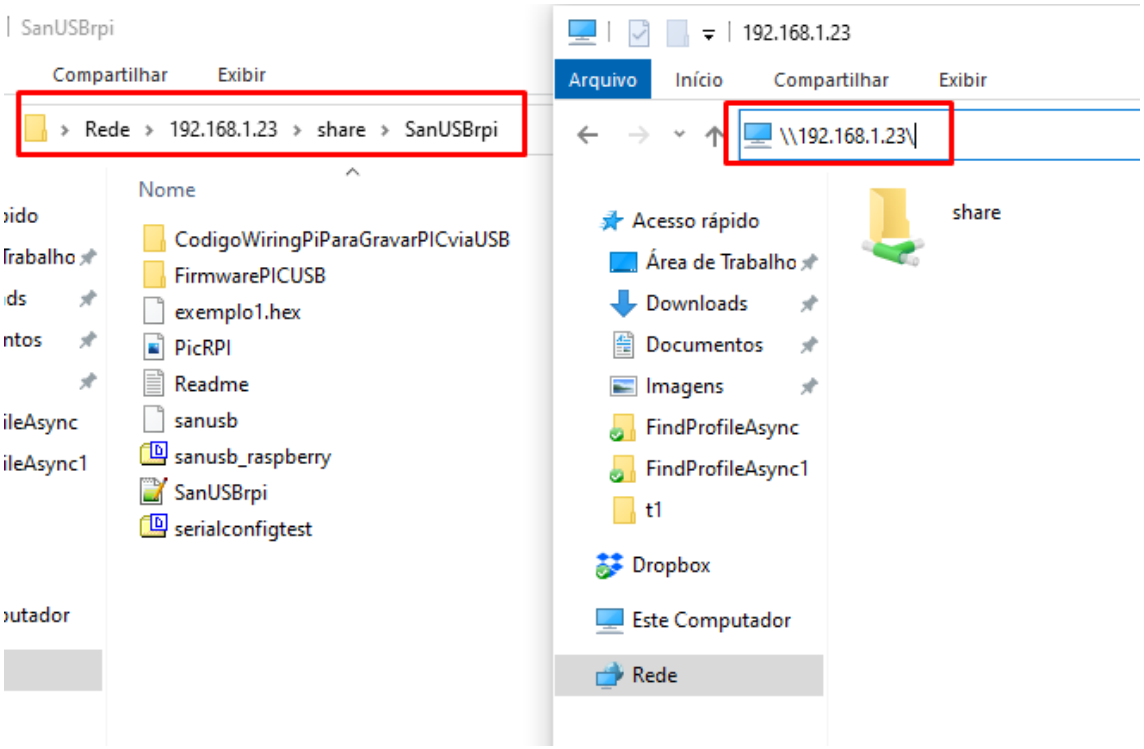


Figura 5.4: Arquivos compartilhados na pasta *share*.

Veja como acessar o Raspberry Pi na Figura 5.5

```
root@RPI3SanUSB:/home/share/SanUSBrpi# ls
CodigoWiringPiParaGravarPICviaUSB  PicRPI.jpg  sanusb_raspberry.deb
exemplo1.hex                       Readme.txt  SanUSBrpi.sh
FirmwarePICUSB                     sanusb      serialconfigtest.deb
root@RPI3SanUSB:/home/share/SanUSBrpi# cd ..
root@RPI3SanUSB:/home/share# chmod 777 SanUSBrpi -R
```

Figura 5.5: Acesso ao Rpi

Para editar algum arquivo específico conceda anteriormente permissão de edição, como super-usuário (*sudo su*). Exemplo:

chmod 777 arquivo

Para editar arquivos de alguma pasta específica conceda anteriormente permissão recursiva (-R) de edição para a pasta, como por exemplo:

chmod 777 pasta -R

Utilizar o Samba para editar arquivos no Desktop através do Explorer ou atarvés de uma IDE como o Visual Studio Code que é gratuito, multiplataforma e disponível em https://code.visualstudio.com/Download. Estas são formas simples de programar uma aplicação no Rpi. A Figura 5.6 mostra a edição de um arquivo.

É possível também no no **Prompt de comando** do Windows ou no próprio VScode (*New Terminal*), abrir um terminal e acessar o Rpi via ssh com o comando ***ssh pi@IPdoRpi***.

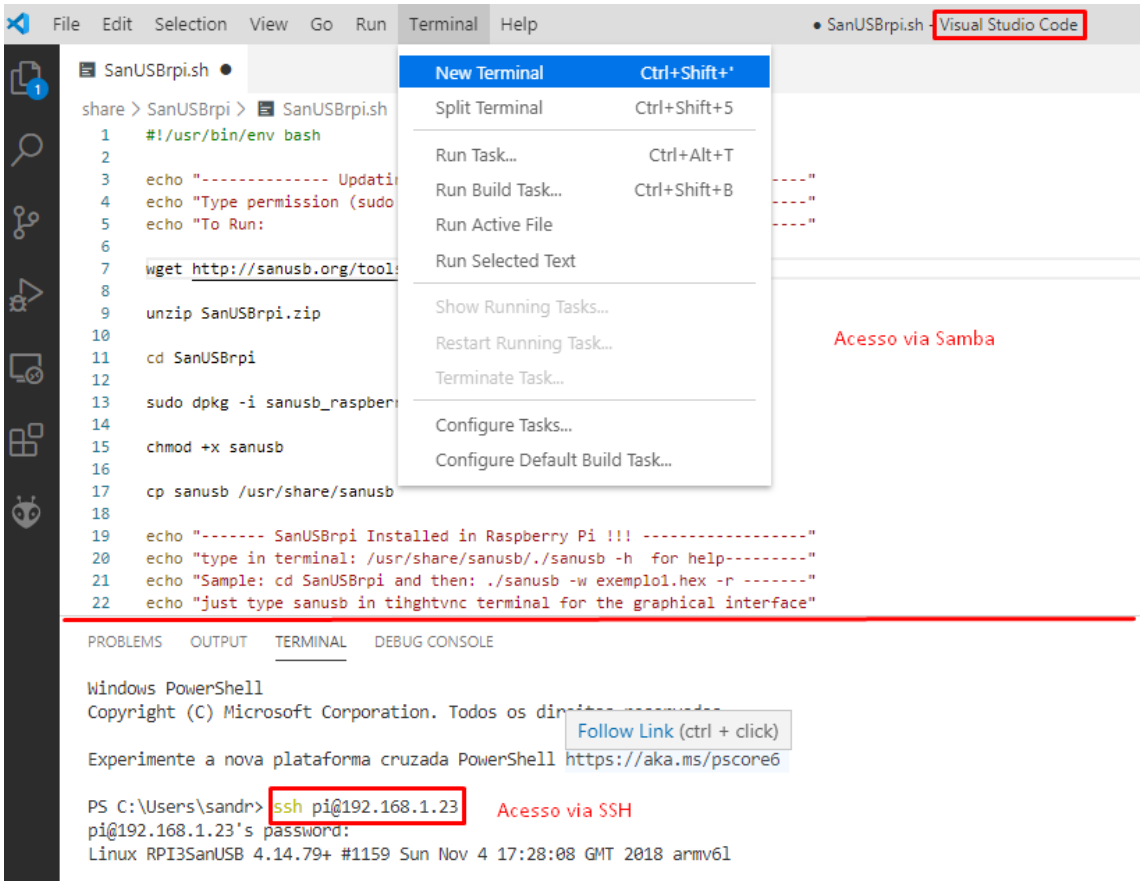


Figura 5.6: Edição com IDE VScode

Outra forma de editar e programar arquivos do RPi após conceder as permissões é utilizar o sftp, que é padrão nas versões atuais do Windows, digitando simplesmente no Explorer:

sftp://*IPdoRpi*

Após inserir Login (pi) e senha (raspberry), o WinSCP é aberto o WinSCP mostrando os arquivos do Rpi conforme ilustrado em 5.7. Como os arquivos já tem permissão de edição, depois de salvar, bastar clicar em ***Pular*** que os arquivos serão atualizados no Rpi.

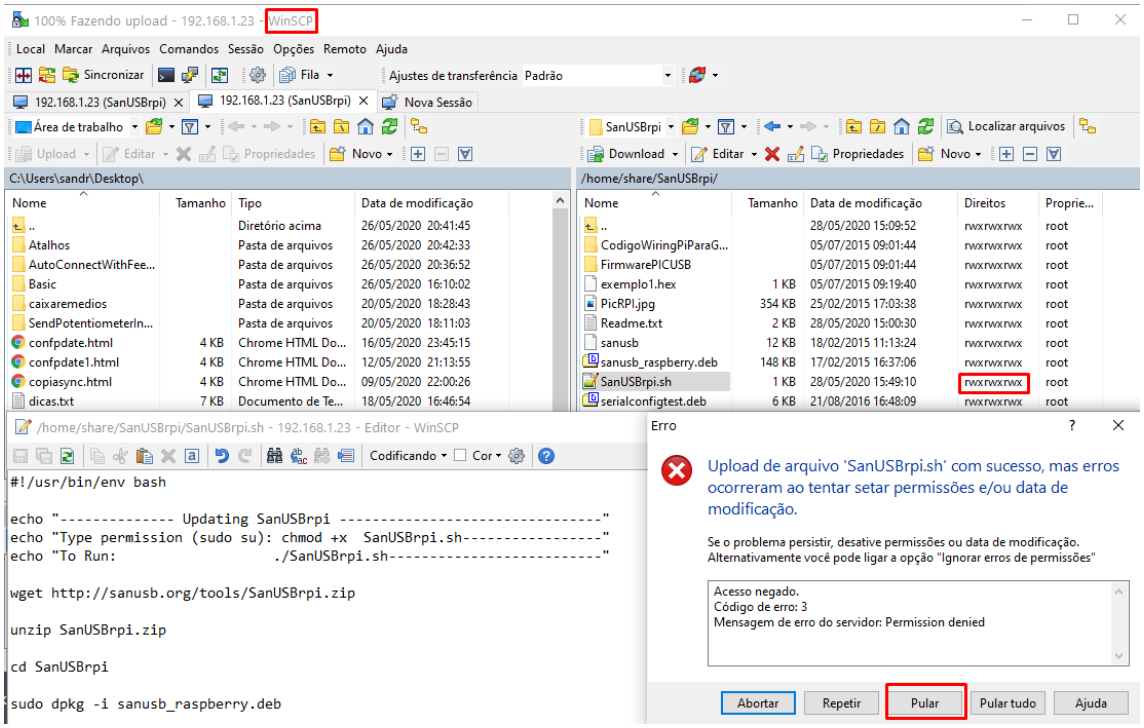


Figura 5.7: Edição com sftp

Se o S.O. do computador for MacOSX, o IP do Rpi, para edição de arquivos com o Samba, irá surgir no Finder. Se for Linux, instale o gerenciador de arquivos Dolphin ou golfinho para acessar os arquivos, conforme a Figura 5.8.

Figura 5.8: Instalação do Dolphin

Antes de iniciar, clique na chave no canto superior direito, depois em barra de localização, ilustrada na Figura 5.9, e marque localização editável para aparecer o endereço dos Rpis (em remote:/).

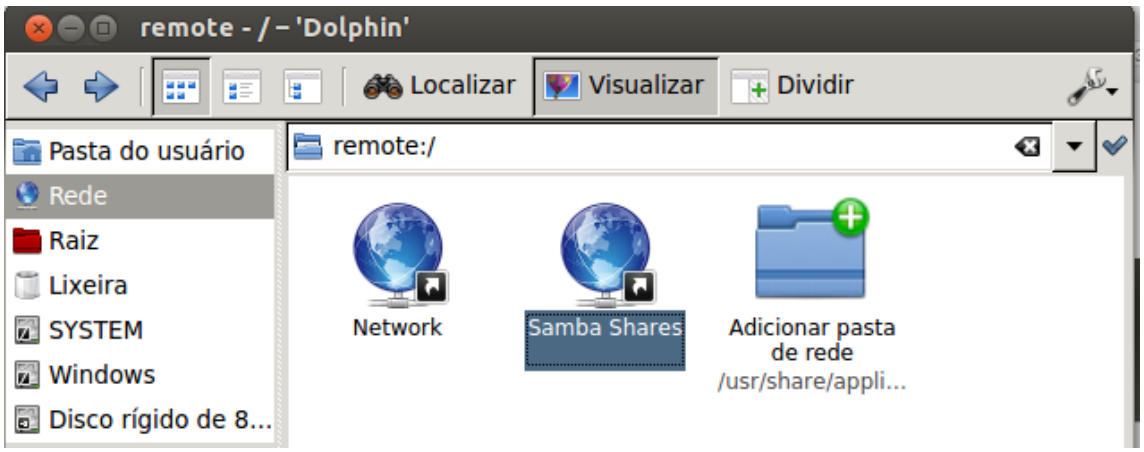


Figura 5.9: Localização da rede no Dolphin

Ao clicar em Samba shares aparecerão todos as máquinas que têm o servidor Samba de compartilhamento de aquivos. Após selecionar a máquina desejada, no caso o Rpi, aparecerá a mesma pasta share configurada anteriormente como mostra a Figura 5.10.

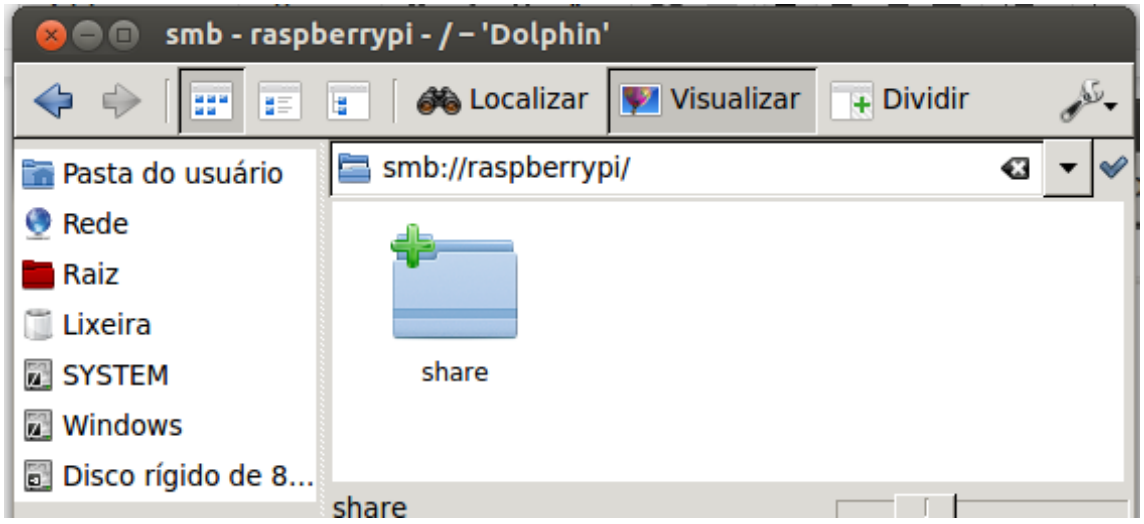


Figura 5.10: Localização da pasta share no Dolphin

5.3 Criar senha para acesso ao Samba

É importante criar uma senha de acesso à pasta share, de forma que somente o usuário autorizado poderá visualizar e trabalhar com os arquivos. O passo a passo é descrito a seguir:

USUÁRIO VÁLIDO PI

Caso não se queira criar um novo usuário, é possível utilizar o usuário válido padrão "pi". Para que o Samba saiba que o "pi"também é um usuário do servidor Samba, digite:

sudo smbpasswd -a pi

Então digite a senha padrão do usuário pi duas vezes (padrão: **raspberry**), ou outra senha caso a senha do usuário pi tenha sido alterada anteriormente pelo comando **pi@raspberrypi: S** passwd).

Depois digite:

nano /etc/samba/smb.conf

Vá com o cursor até o final de share e modifique o public (public=no):

[share]

...

public=no

Depois basta reiniciar, fechar o Explorer e, da próxima vez, o Samba pedirá a nova senha cadastrada pelo usuário pi.

```
CRIAR NOVO USURIO E SENHA PARA O SAMBA (exemplo de usuário: laese)
root@raspberrypi:/home/pi# adduser laese (Para remover: userdel usuário)
Adding user laese ...
Adding new group laese (1002) ...
Adding new user laese (1001) with group laese ...
Creating home directory /home/laese ...
Copying files from /etc/skel ...
Enter new UNIX password: (exemplo: sanusb) ou <Enter> até o final (sem
    password)
Retype new UNIX password: (exemplo: sanusb)
No password supplied
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Enter the new value, or press ENTER for the default

root@raspberrypi:/home/pi# smbpasswd -a laese
New SMB password: ifcelaese (Escolha a senha)
Retype new SMB password: ifcelaese
Added user laese.

root@raspberrypi:/home/pi#
Digte:
nano /etc/samba/smb.conf

Vá com o cursor até o final share e inclua:
[share]

public=no (observe que antes o acesso estava liberado, com a opão public
    =yes)
valid user=laese #caso seja usuário pi não énecessário essa linha

root@raspberrypi:/home/pi/scripts# service samba restart
[ ok ] Stopping Samba daemons: nmbd smbd.
[ ok ] Starting Samba daemons: nmbd smbd.
```

Pronto. Já é possível entrar pelo Window Explorer e visualizar os arquivos compartilhados /home/share/. Se for a primeira vez, e for marcado “Lembrar minhas credenciais”, após o login e senha, o Explorer abrirá nas próximas vezes normalmente sem o pedido de senha como na Figura 5.11

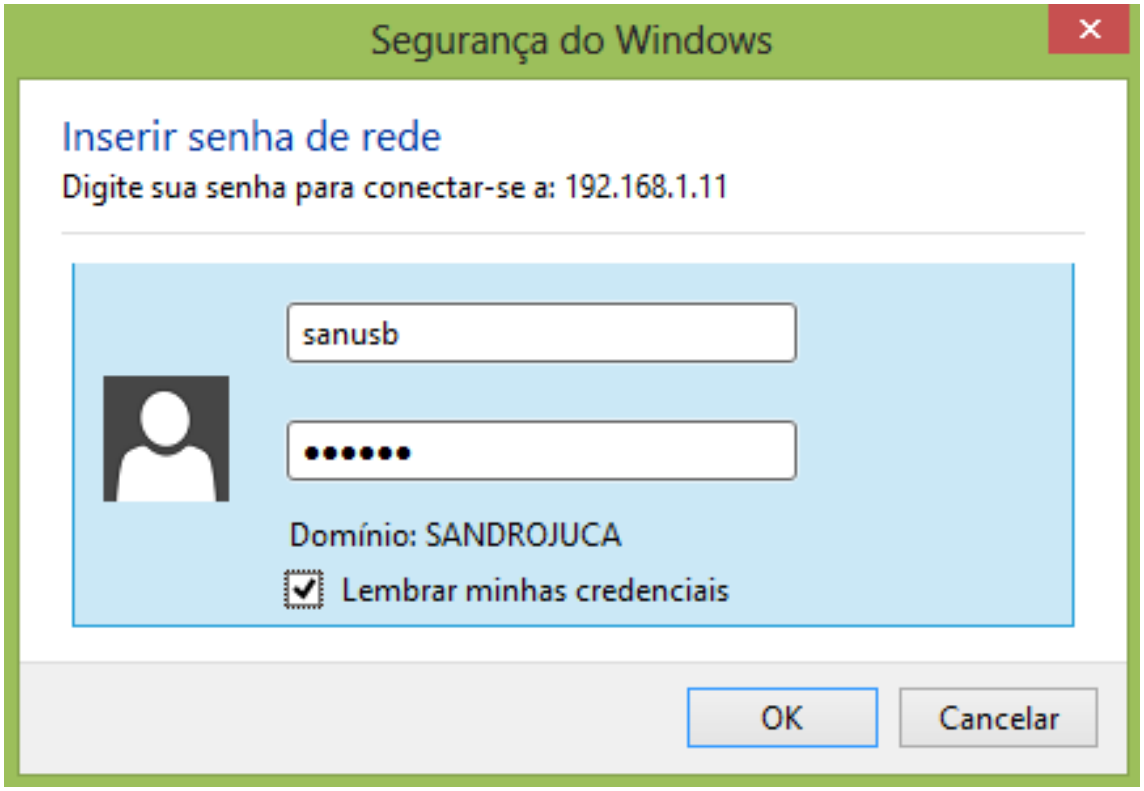


Figura 5.11: Login de segurança do Windows

Diretório compartilhado em: **cd /home/share/**
Para trocar a senha do Samba, é possível digitar novamente (**smbpasswd -a usuário**) e inserir a nova senha ou remover o usuário anterior (se não for o pi), inserir um novo usuário e senha (com addusr) e reiniciar o Rpi (reboot) e o computador cliente.

Vale salientar que a senha inserida nessa Janela de segurança do Windows® fica armazenada na memória RAM do PC. Dessa forma, caso a senha seja modificada, é necessário reiniciar também o PC para apagar a senha da RAM do PC e poder inserir a nova senha cadastrada pelo usuário do Rpi. Assim é recomendado desligar o PC quando não for mais necessário acessar remotamente uma pasta de segurança do Rpi.

5.4 Compartilhador de arquivos Owncloud

O OwnCloud, é um serviço robusto de código aberto para armazenamento e compartilhamento online de arquivos localizado em uma máquina com S.O. Linux, como é o caso do Raspberry Pi. O serviço foi desenvolvido por responsáveis pelo KDR com o objetivo de disponibilizar uma forma de sincronia e armazenamento online com controle de versões, senhas, permissão de edição, histórico de atividades e data de expiração do arquivo disponibizado por URL.

Dessa forma, o servidor torna-se privado onde o usuário pode saber onde se localiza fisicamente. Para baixar os instaladores de clientes Linux, Windows, Mac OSX ou clientes mobile, basta acessar

https://owncloud.org/install/

Para instalar o owncloud no raspbian, ou seja para realizar a etapa 1 (*Get Owncloud server*) do site **https://owncloud.org/install/** no Raspbian, siga os passos abaixo:

Atualize o sistema:
sudo apt-get update
sudo apt-get upgrade

Instale os pacotes necessários:
sudo apt-get install apache2 php5 php5-json php5-gd php5-sqlite curl libcurl3 libcurl3-dev php5-curl php5-common php-xml-parser sqlite

Baixe uma versão do Owncloud:
sudo wget https://download.owncloud.org/community/owncloud-8.1.3.tar.bz2
É possível verficar a última versão em https://owncloud.org/changelog/

Descompacte o arquivo
sudo tar -xjf owncloud-8.1.3.tar.bz2

Copie a pasta descompactada para a pasta do servidor apache /var/www/html

sudo cp -r owncloud /var/www/html

Configure como proprietário o diretório do Apache

sudo chown -R www-data:www-data /var/www

Renicie o Rpi para concluir a instalação

reboot

Após o reinicio, acesse o site do servidor Owncloud pelo IP do Rpi, como por exemplo: http://192.168.25.5/owncloud/index.php.
Assim, deve aparecer uma tela inicial para configurar login e senha do servidor mostrado a seguir na Figura 5.12.

Figura 5.12: Tela inicial do Owncloud

Pronto basta ir para o passo 2 (Sync your data) do site **https://owncloud.org/install/** e baixar aplicação para clientes de acesso aos arquivos do Owncloud, fazer o login no computador do cliente, seja desktop ou mobile, e os dados já serão sincronizados e visualizados via URL na Figura 5.13 e via Explorador de aquivos na Figura 5.14. Na visualização dos arquivos do servidor via URL é possível controlar e restaurar versões anteriores dos arquivos, gerar URL de compartilhamento de arquivo local com senha e data de expiração.

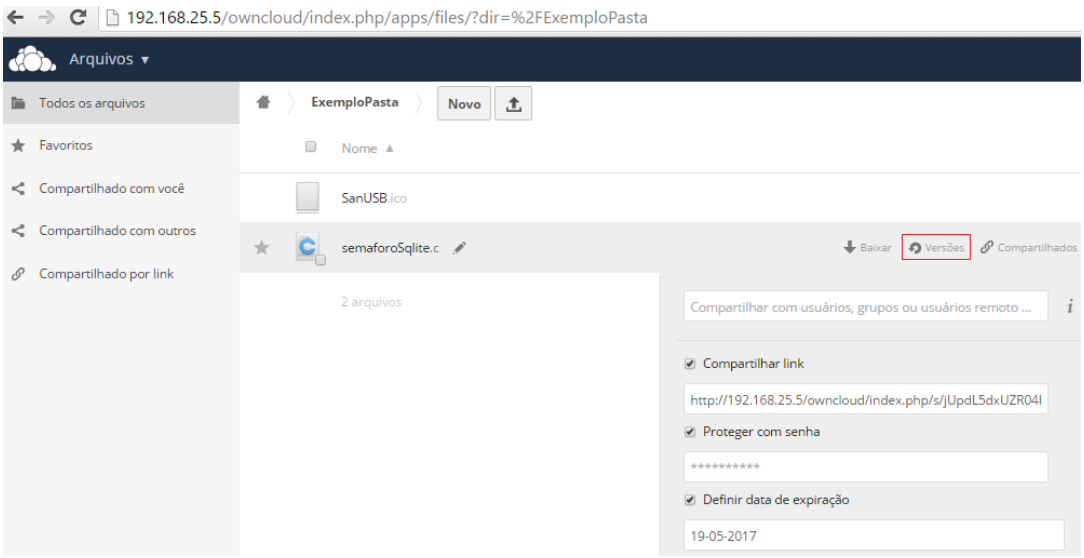


Figura 5.13: Visualização dos arquivos do servidor via URL

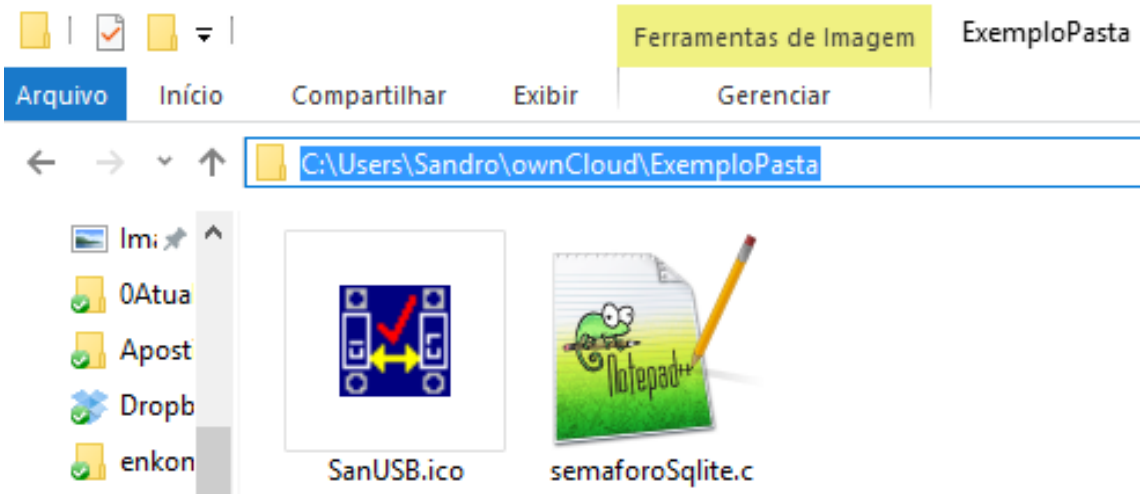


Figura 5.14: Visualização dos arquivos do servidor via explorer

5.5 Copiar arquivos do PC para o Raspberry Pi com Filezilla e vice-versa

Uma das melhores ferramentas é utilizar o SSH *File Transfer Protocol* ou SFTP (Figura 5.15). Nesse caso, SSH significa Secure Shell). Para isso, se utiliza o Filezilla com configuração mostrada na figura abaixo a apartir do IP do Rpi na rede. Nesse caso, o Host é o IP, o Nome do usuário é pi, a senha é raspberry e a Porta SFTP é 22.

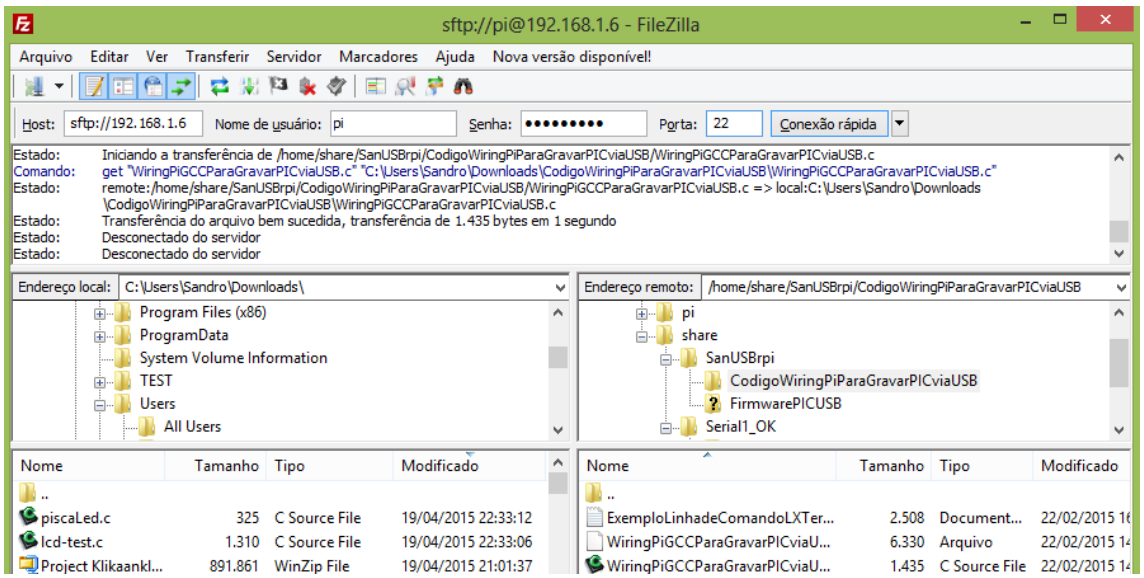


Figura 5.15: Filezilla

6. Principais comandos para o Raspberry Pi

6.1 Listagem de Arquivos e Movimentação

6.1.1 Localização atual

O primeiro dos comandos Linux que vamos apresentar é o comando '**pwd**'. Ao abrir o terminal, queremos saber em que 'pasta' estamos a navegar. Este comando mostra-lhe em que diretório se encontra atualmente na navegação.

- No terminal introduza o seguinte comando: pwd

Poderá ver o caminho para o diretório onde se encontra na navegação mostrado abaixo na Figura 6.1

```
rpi@RaspberryPi:~$ pwd
/home/rpi
rpi@RaspberryPi:~$
```

Figura 6.1: Localização atual no Rpi..

6.2 Listagem de arquivos e diretórios

O comando para listar o conteúdo da diretoria em que se encontra é '**ls**'.

- No terminal introduza o comando:
ls

Surgirá uma listagem com o conteúdo da sua localização atual mostrado a seguir na Figura 6.2). Pode ver todas as 'pastas' (chamadas normalmente de diretórios no Linux) e arquivos que se encontram nessa localização.

Figura 6.2: Arquivos locais

6.3 Navegação – Endereços absolutos

Para navegar para outras localizações, o comando Linux utilizado é '**cd**' seguido da localização para onde quer navegar. A raíz do sistema é representado por uma barra no início do caminho especificado '/'.

- Navegue para a raiz do sistema com o seguinte comando:

cd /

- Para conferir onde se encontra, utilize novamente o comando '**pwd**'.

pwd

Verá que na Figura 6.3 se encontra em '/' (raíz do sistema).

Figura 6.3: Arquivos locais

Se quiser ir para a sua pasta desejada que se encontra normalmente em '*/home/pastadesejada*', terá que utilizar o comando:
cd /home/pastadesejada

Para saber o seu nome da pasta atual digite o comando '**whoami**'.

- Com estes comandos já será capaz de navegar entre pastas, saber a sua localização e listar o conteúdo dos diretórios.

Assim foi aprendido como navegar pelo sistema Linux utilizando o comando 'cd' seguido de um **endereço absoluto** do tipo '**/home/pastadesejada**'. São chamados **endereços absolutos** porque está a indicar um caminho completo a partir da raíz '/'.

Outra forma de navegar é utilizando caminhos relativos ao diretório em que se encontra. Vamos

ver um exemplo:
- Endereço relativo – Subir um nível

- No terminal digite o comando '**cd /**' para ir para a raiz
  cd /
  Vamos agora até à pasta desejada, 'subindo' um diretório de cada vez em alternativa a introduzirmos o caminho completo como vimos anteriormente.
- Digite agora o comando para ir para o diretório seguinte (**/home**).
  cd home

Encontra-se agora em '/home' como pode verificar como comando '**pwd**'.
Falta então subir mais um diretório para a pasta desejada. Por exemplo rpi. - Digite o comando:

cd rpi

Novamente com o comando '**pwd**' pode verificar que se encontra em '/home/rpin, mostrado na Figura 6.4

```
rpi@RaspberryPi:/home$ cd /
rpi@RaspberryPi:/$ cd home
rpi@RaspberryPi:/home$ pwd
/home
rpi@RaspberryPi:/home$ cd rpi
rpi@RaspberryPi:~$ pwd
/home/rpi
rpi@RaspberryPi:~$
```

Figura 6.4: Diretórios Rpi.

6.4 Endereço relativo – Descer um nível

Para descer um nível a partir da localização atual, o comando Linux que vamos utilizar é '**cd** ..'.

- No terminal introduza '**cd** ..'. *cd* ..

Com o comando '**pwd**' pode verificar que desceu um nível de '*/home/rpi*' para '/home', como mostra a Figura 6.5.

```
rpi@RaspberryPi:~$ pwd
/home/rpi
rpi@RaspberryPi:~$ cd ..
rpi@RaspberryPi:/home$ pwd
/home
rpi@RaspberryPi:/home$
```

Figura 6.5: Diretórios Rpi.

Para finalizar pode limpar o terminal com o comando '**clear**'.

6.5 Criação de diretórios

Iremos agora criar um novo diretório na pasta de utilizador e para isso vamos recorrer ao comando:

mkdir

Para criar um diretório dentro daquele em que se encontra basta introduzir o comando "**mkdir**" seguido do nome do diretório a criar (*mkdir nomediretorio*).

Também pode criar um diretório indicando o caminho completo do mesmo (mkdir /home/pasta-desejada/nomediretorio).

- Comece por navegar até à sua pasta de utilizador com os comandos anteriormente aprendidos.
  cd /home/pastadesejada
- Utilize agora o comando "mkdir" para criar o novo diretório.
  mkdir meudiretorio
- Com o comando "ls" confirme que o diretório foi criado.
  ls

6.6 Criação de arquivos

A criação de um novo arquivo é feita através do comando "**touch**" seguido do nome e extensão desejados. Tal como para os diretórios, também podemos criar arquivos dentro do diretório onde nos encontramos, bastando para isso indicar o nome do arquivo ou fazê-lo indicando o caminho completo para o arquivo (touch /. . . /nomedapasta/nomedoarquivo). Note que a pasta onde vai criar o arquivo já tem que existir!

- Navegue até à sua pasta de utilizador e crie um novo arquivo com o comando "*touch*".
  touch meuarquivo.txt
- Com o comando "ls" verifique que o arquivo foi criado.
  ls

Depois de aprender como movimentar-se pela estrutura do sistema operacional e como criar diretórios e arquivos em Linux, vamos agora mostrar como pode copiar, mover ou renomar arquivos através da linha de comandos.

Abra uma sessão no terminal e vamos começar a partir da nossa pasta desejada (/home/-pastadesejada).

6.7 Copiar arquivos

A cópia de arquivos em Linux é feita através do comando "cp" seguido do nome do arquivo a copiar e do caminho que indica o destino para a cópia. Se seguiu o artigo anterior, deverá ter na sua pasta de utilizador o arquivo "*meuarquivo.txt*" e o diretório "*meudiretorio*" que criámos. Vamos utilizá-los neste tutorial.

- Vamos copiar o arquivo "*meuarquivo.txt*" da sua pasta atual para dentro do diretório "*meu-diretorio*" também na mesma localização através do comando "cp". (**cp arquivo-a-copiar destino-da-cópia**).

cp /home/pastaatual/meuarquivo.txt /home/ pastaatual/meudiretorio/

Também pode usar caminhos relativos para este comando, tal como vimos para os comandos "*ls*" e "*cd*", por exemplo.

- Verifique que o arquivo foi copiado com o comando "ls".

ls /home/ pastaatual/meudiretorio/

6.8 Mover e renomear arquivos

A forma de mover e renomear arquivos é praticamente igual ao que acabamos de ver para copiar arquivos. Apenas temos que substituir o comando "*cp*" por um novo comando: "*mv*"
Melhor ainda é que o comando "mv" serve para mover e renomear arquivos.
De repente pode parecer estranho, mas, se pensar bem, se mover um arquivo de uma localização para essa mesma localização, mas dando outro nome ao arquivo, está no fundo a renomear o arquivo.

Embora tecnicamente tenha sido criada uma cópia e eliminado o original, o resultado prático é o mesmo.

- Crie um novo arquivo "outroarquivo.txt" na pasta de utilizador.
  touch /home/meudiretorio/outroarquivo.txt
- Para mover o arquivo que criou para pasta " minhapasta" utilize o comando "mv" seguido arquivo a copiar e localização do destino.
  mv /home/meudiretorio/outroarquivo.txt /home/meudiretorio/minhapasta/
- Verifique que o arquivo foi movido e já não se encontra na localização original.
- Para renomear o arquivo "meuarquivo.txt" no diretório de utilizador utilize "*mv nomeantigo nomenovo*".
  mv /home/meudiretorio/meuarquivo.txt /home/meudiretotio/novonome.txt

6.9 Listar e remover processos em execução

Para listar e encerrar processos em execução basta inserir o comando "*ps*" para listar processos e "kill processo" para encerrar processos em execução.

6.10 Inicialização automática de scripts

6.10.1 Systemd

Uma das possibilidades de inicialização automática de scripts, além do crontab, é utilizando a aplicação systemd. Considere um script para piscar um Led em *Bash shell* chamado de *Blink.sh* abaixo:

```
#!/bin/sh
#sanusb.org # pino BCM 21 e Pino fsico 40

echo 21 > /sys/class/gpio/export #wPi 29
echo out > /sys/class/gpio/gpio21/direction
while :
do
   echo Blink shell
   #gpio write 1 1
   echo "1" > /sys/class/gpio/gpio21/value
   sleep 1.5

   #gpio write 1 0
   echo "0" > /sys/class/gpio/gpio21/value
   sleep 1.5
done
```

Vamos agora iniciar o processo criando um arquivo ***.service*** chamado de *Blink.service* dentro da pasta /lib/systemd/system, onde estão todos os serviços systemd, utilizando:

```
pi@raspberrypi:~ $ sudo nano /lib/systemd/system/Blink.service
```

E depois inserindo o conteúdo abaixo:

```
[Unit]
Description=Bora piscar um Led

[Service]
Type=simple
ExecStart=/home/pi/Blink.sh

[Install]
WantedBy=multi-user.target
```

Para um arquivo em python o *Execstart* seria ***ExecStart=/usr/bin/python /home/pi/Blink.py***.

Após criar o arquivo Blink.service, é necessário habilitá-lo com o comando **sudo systemctl**

enable Blink.service e depois reiniciar com reboot como abaixo.

```
pi@raspberrypi:~ $ sudo systemctl enable Blink.service
Created symlink /etc/systemd/system/multi-user.target.wants/Blink.
   service  /lib /systemd/system/Blink.service.
pi@raspberrypi:~ $ sudo reboot
```

Para iniciar, parar e ver o status de execução contínua do script após o *boot*, basta digitar respectivamente:

```
pi@raspberrypi:~ $ sudo systemctl start Blink.service
pi@raspberrypi:~ $ sudo systemctl stop Blink.service
pi@raspberrypi:~ $ sudo systemctl status Blink.service
```

Para parar o serviço de execução automática basta desabilitar com o comando:

```
sudo systemctl disable Blink.service
```

6.10.2 Bashrc

Outra forma de inicialização automática de scripts ou aplicações, é através do arquivo oculto chamado ***bashrc*** localizado na pasta /home/pi. Para visualizar, basta digitar:

```
ls -la
```

No final do arquivo bashrc editado como **nano .bachrc**, insira o comando abaixo de execução em segundo plano (&) do script desejado. Exemplo:

```
./Blink.sh&
```

Mantenha o script Blink.sh também em **/home/pi** ou em outra pasta indicando o endereço absoluto do local.

Para parar o processo, verifique o número utilizando o comando **ps** e depois execute o comando *kill*. Exemplo:

```
sudo kill 5941
```

6.10.3 rc.local

Outra forma de inicialização automática de scripts ou aplicações, é através da edição so arquivo chamado **rc.local**.

```
sudo nano /etc/rc.local
```

Tendo em vista a execução de uma aplicação python Blinkpy.py para piscar um Led em ***home/pi***, descrita abaixo:

```
import RPi.GPIO #sanusb.org/
import time

RPi.GPIO.setwarnings(False)
RPi.GPIO.setmode(RPi.GPIO.BCM)
RPi.GPIO.setup(21, RPi.GPIO.OUT)

# pisca GPIO21 -Pino fsico 40
print "Blink Python"

while(True):
       RPi.GPIO.output(21,RPi.GPIO.HIGH)
       time.sleep(0.3)
       RPi.GPIO.output(21,RPi.GPIO.LOW)
       time.sleep(0.3)
```

Após abrir o arquivo, insira a execução da inicialização automática da aplicação no final do arquivo, como por exemplo:

```
sudo python /home/pi/Blinkpy.py &
```

6.10.4 init.d

Nessa aplicação de inicialização automática, é necessário que a aplicação esteja na pasta ***/etc/init.d***. Dessa forma, vamos criar um aplicação python, como exemplo:

```
pi@raspberrypi:~ $ sudo nano /etc/init.d/Blinkpy.py
```

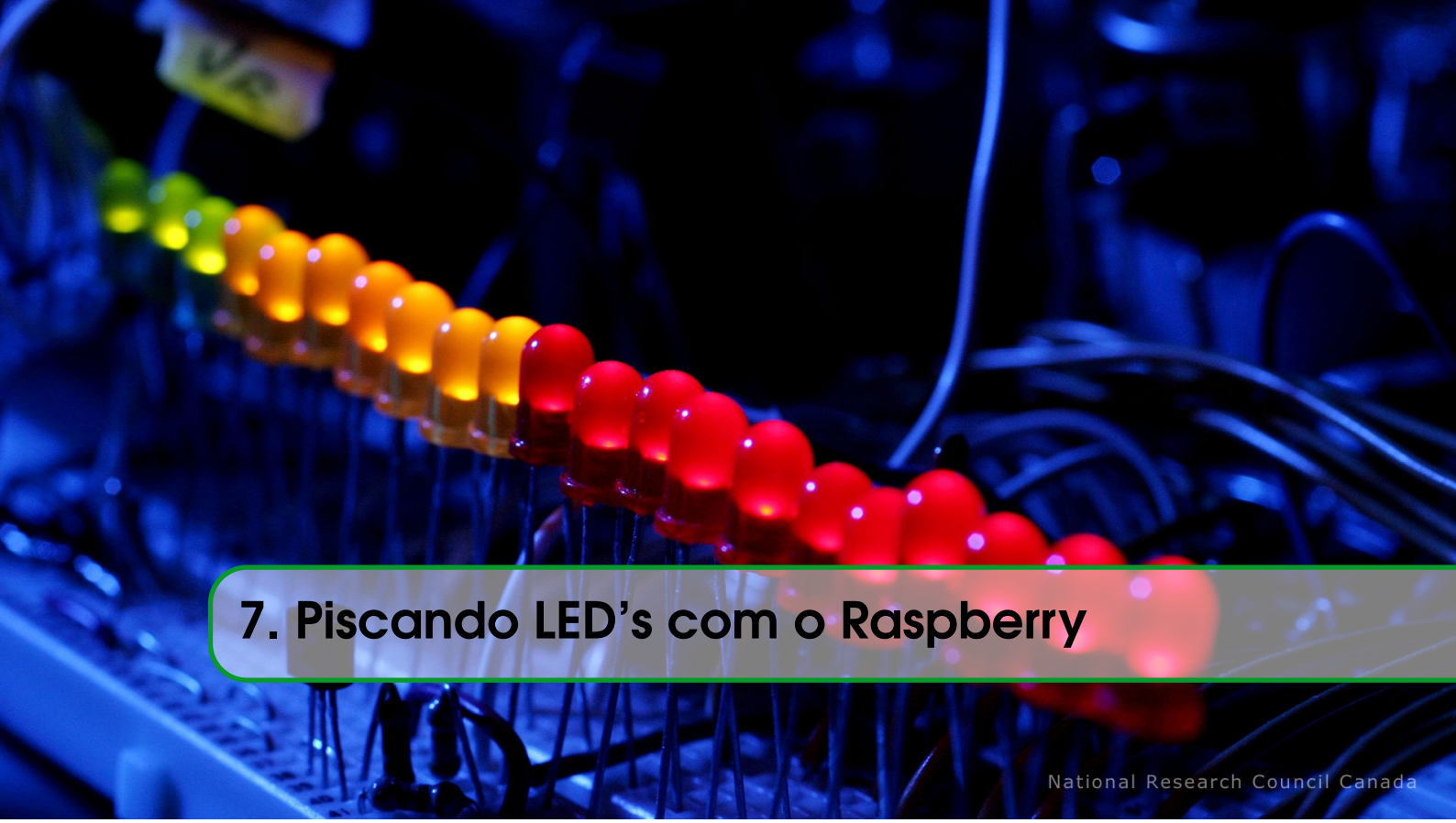
Conceder permissão de execução:

```
pi@raspberrypi:~ $ sudo chmod +x /etc/init.d/Blinkpy.py
```

Entre na pasta ***cd /etc/init.d*** e execute o comando:

```
pi@raspberrypi:~ $ sudo update-rc.d Blinkpy.py defaults
```

Após o ***reboot***, o processo estará funcionando.



7. Piscando LED's com o Raspberry

Finalizada a instalação do sistema operacional Raspbian, é possível agora testar a parte do hardware da placa utilizando o terminal pelo modo gráfico ou SSH utilizando o programa putty.

Para facilitar a programação é utilizada uma biblioteca desenvolvida em C para acesso das portas GPIO do Raspberry chamada "wiringPi". As informações completas sobre a biblioteca está em http://wiringpi.com/. Mais detalhes em: https://www.youtube.com/watch?v=J6KsTz6hjfU . Como é acesso ao hardware, é recomendável estar como usuário root digitando sudo su no terminal remoto putty ou VNC.

A partir da versão Jessie do Raspbian, o WiringPi já vem pré-instalado. Para verificar a instalação basta digitar no terminal *gpio readall* e deve aparecer a configuração dos pinos de I/O. Se não aparecer a descrição dos pinos, instale o wiringpi normalmente:

sudo apt-get install wiringpi

A partir da versão do Raspberry Pi 4, é necessário instalar a última versão do wiringpi digitando no terminal os comandos:

cd /tmp

wget https://project-downloads.drogon.net/wiringpi-latest.deb

sudo dpkg -i wiringpi-latest.deb

gpio -v

Neste tópico é mostrado também como acessar os pinos GPIO, ilustrado na Figura 29, utilizando linguagem em C através da biblioteca"***wiringpi***"e fazer um LED conectado ao GPIO 17 piscar. A Figura 7.1 demonstra o mapeamento dos pinos.

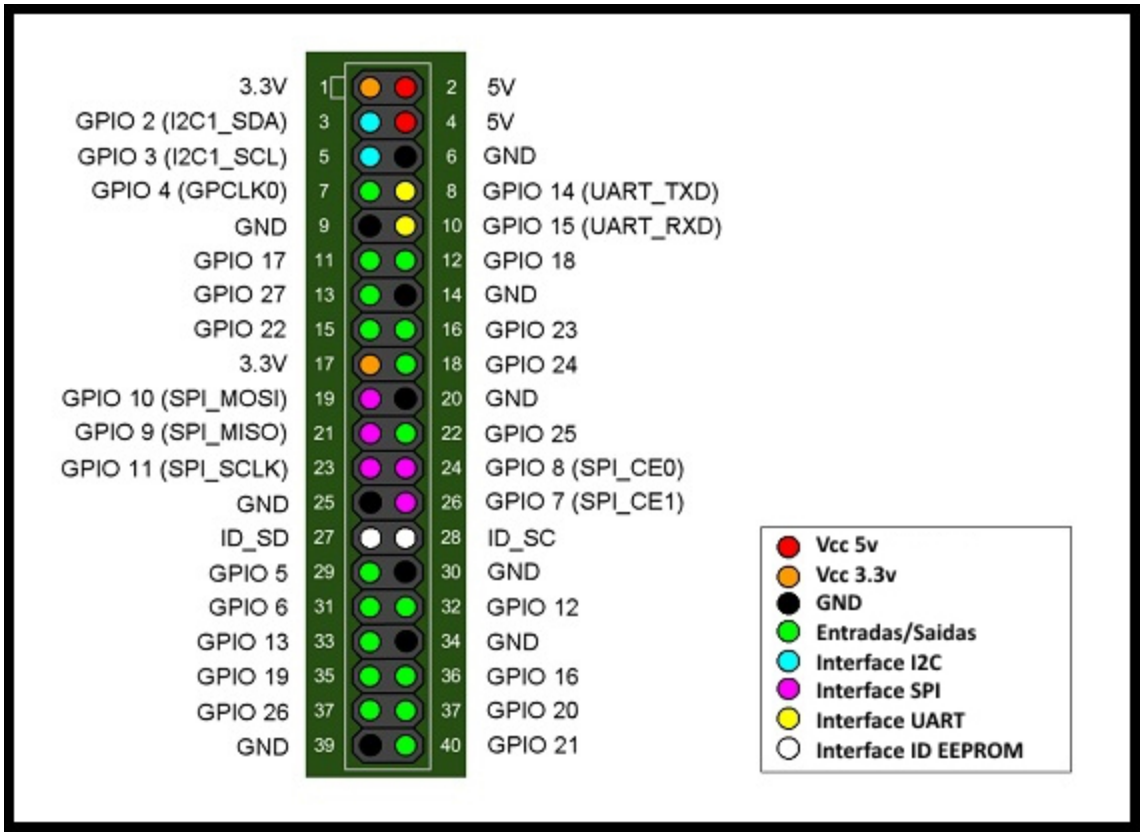


Figura 7.1: Pinos GPIO.

O circuito do LED abaixo Figura 7.2 é bastante simples.

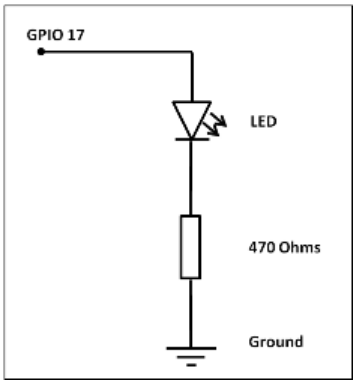


Figura 7.2: Conexão de LED ao Rpi.

A indicação física de conexão do LED nos pinos GPIO é indicada na Figura 7.3:

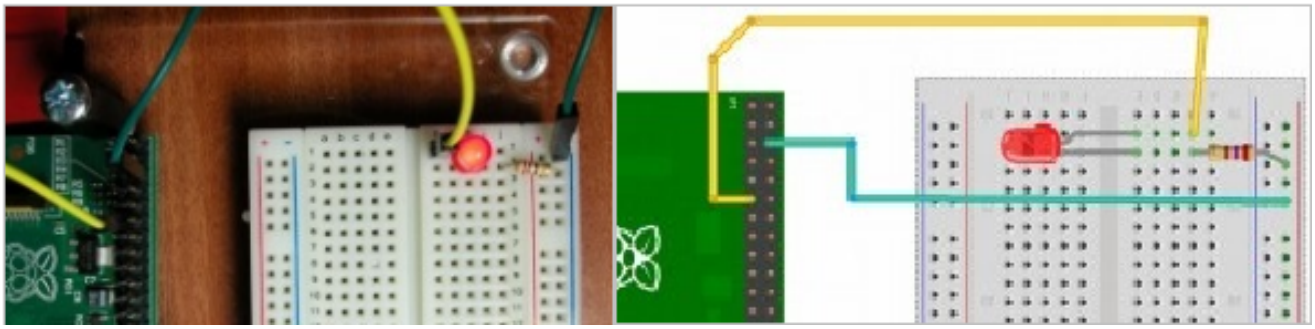


Figura 7.3: Ligação de um pino GPIO com um LED..

Após a instalação da bibloteca wiringpi, são habilitadas as seguintes comandos do modo GPIO que podem ser digitados no terminal:

7.1 Comandos de entrada e saída no LXTerminal

gpio mode <pin> in / out / pwm / clock / up / down / tri

Esse comado define o modo de um pino para ser de entrada, saída PWM ou clock, e, adicionalmente, pode definir os resistores internos de *pull-up* e *pull-down* ou nenhum.

gpio write <pin> 0/1

Define um pino de saída para alto (1) ou baixo (0)

gpio pwm <pin> <valor>

Define o pino para um valor PWM (de 0-1023)

gpio read <pin>

Lê e imprime o valor lógico do pino de dado. Ele irá imprimir 0 (baixo) ou 1 (alto).

gpio readall

Esse comando lê todos os pinos normalmente acessíveis e imprime uma tabela de seus números (wiringPi, BCM-GPIO e números de pinos físicos), por isso torna-se um gráfico de referência cruzada útil), juntamente com os seus modos e valores atuais.

gpio wfi <pin> subida / descida / ambos

Isso faz com que GPIO para realizar uma espera não-ocupada em um único pino GPIO até que muda de estado para que indicado.

gpio edge <pin> rising/falling/both/none

(subida / queda / ambos / nenhum)

Isso permite que o pino para borda interrupção seja habilitado na subida, descida ou ambas as bordas ou nenhum para desativá-lo.

Nota: **Os números de pinos no modo sys são sempre números de pinos BCM-GPIO.**

Exemplos: Os comandos abaixo utilizam os números de pinos wiringPi para definir pino 0 como uma saída e, em seguida, define o pino para nível lógico 1.

gpio mode 0 out

gpio write 0 1

Os comandos abaixo utilizam esquema de numeração pin BCM-GPIO e realiza a mesma operação como acima.

gpio -g mode 17 out

gpio -g write 17 1

Este outro abaixo utiliza o esquema de numeração física (**gpio menos um**) dos mesmos pinos dos comandos anteriores e executam a mesma operação como acima.

gpio -1 mode 11 out

gpio -1 write 11 1

7.2 Resistores internos de pull-up e pull-down

Os pinos GPIO têm resistências internas **pull-up** que podem ser controladas via software quando um pino está em modo de entrada.

gpio mode 0 up

gpio mode 0 down

gpio mode 0 tri

Neste exemplo, são definidos os resistores para *pull-up*, *pull-down* e nenhum (*tristate*), respectivamente, em wiringPi no pino 0. Em linguagem C, o comando WiringPi utilizado é o **pullUpDnControl(int pino, int PUD)**; //PUD pode ser PUD-UP, PUD-DOWN ou PUD-OFF.

Um circuito tipíco de botão ligado a um pino do Rpi com pull-up é ilustrado na Figura 7.4.

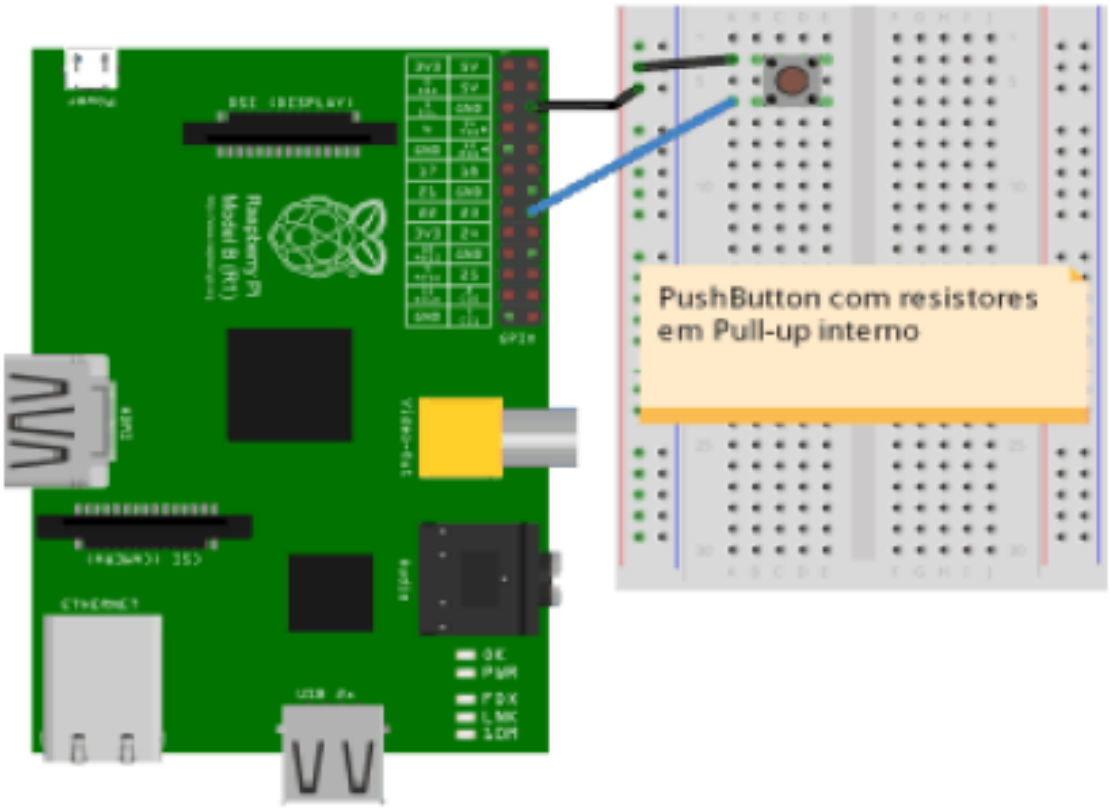


Figura 7.4: Conexão de botão ao Rpi.

Quando o botão for pressionado o nível logico do pino passa de 1 (alto) para o nível lógico baixo (0), pois o curto-circuito do botão conecta o pino ao Gnd.

7.3 Comandos shell para acionamento dos pinos

Para atuar os pinos físicos, o processador olha para os registros disponíveis em /sys/class/gpio. Então para modificar o estado físico dos pinos, é necessário inicialmente exportá-los para esse diretório /sys/class/gpio/export.

echo 17 > /sys/class/gpio/export

Após a exportação, é possível verificar o surgimento do diretório /sys/class/gpio/gpio17 em /sys/class/gpio/. Em seguida, este pino BCM deve ser configurado para saída, ecoando **out** para *direction*:

fecho out > /sys/class/gpio/gpio17/direction

E finalmente comandar o valor 1 ou 0 para ligar e desligar o pino BCM:

fecho 1 > /sys/class/gpio/gpio17/value

Dessa forma, é possível fazer um led piscar através de um script *shell*:

```
#!/bin/sh
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction

while [ 0 ]
do
echo 1 > /sys/class/gpio/gpio17/value
sleep 1
echo 0 > /sys/class/gpio/gpio17/value
sleep 1
done
```

Outro exemplo para comandos shell é descrito a seguir. Crie um diretório exemplo para inserir um script.

- cd
- mkdir scripts
- cd scripts

Crie o seguinte script com um arquivo chamado **lightswitch**

```
#!/bin/bash
if [ $# > 1 ]
then
/usr/local/bin/gpio mode 4 out
if [[ "$1" = "on" ]] # $1 inserted parameter
then
/usr/local/bin/gpio write 4 on
```

```
fi
if [[ "$1" = "off" ]]
then
/usr/local/bin/gpio write 4 off
fi
fi\textbf{}\textbf{}
```

Declare o script como executável:

chmod +x lightswitch

Execute o script com os comandos abaixo.

./lightswitch on

./lightswitch off

Se tudo ocorrer corretamente, o pino gpio 4 vai varias com os comandos acima.

7.4 Comandos WiringPi

gpio export <pin> in/out

Exporta o comando WiringPi, realizado por software, para um pino físico (BCM-GPIO) configurado em /sys/class/GPIO e o define como uma entrada ou saída tornando o pino disponível para os usuários.

gpio unexport <pin>

Remove a exportação de um pino.

gpio unexportall

Remove todas as exportações /sys/class/GPIO.

exports GPIO

Isso imprime uma lista de todos os pinos GPIO que foram exportados através da interface em /sys/class/GPIO e seus modos.

Levando em consideração que a biblioteca "**wiringpi**" já esteja instalada, vamos digitar o seguinte comando para ver a situação das portas GPIO do Raspberry disponível em /sys/class/GPIO. É necessário estar como usuário root (*sudo su*).

gpio readall

Aparecerá a tabela conforme a Figura 7.5.

```
pi@raspberrypi ~/code/wiringPi $ gpio readall
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 0 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 0 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 0 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 0 | 27 || 28 | 0 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 0 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 0 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
```

Figura 7.5: Tabela de pinos do comando GPIO.

Veja que a GPIO 17 (primeira linha da tabela) está em modo OUT, ou seja, é possível enviar comandos através dela. Caso a porta esteja em modo IN, deve-se mudar sua configuração para OUT, da seguinte forma:

gpio mode 0 out

Veja que o valor dessa porta está como LOW. Para mudar o valor manualmente, podemos digitar o seguinte comando:

gpio write [wiringpi port] [0/1] ,ou seja,

gpio write 0 1

Nesse caso o valor dessa porta passará a ser HIGH. Mas o que queremos é mudar o valor (0/1) dessa porta através de um código C para piscar o LED. Para isso, você pode utilizar o diretório compartilhado /home/share para criar o arquivo blink.c.

Para configuração do número dos pinos é possível utilizar os seguintes comandos em int main():

wiringPiSetup();//Para configurar os pinos na sequência wPi

wiringPiSetupGpio();//Para configurar os pinos na sequência GPIO (BCM)

wiringPiSetupPhys();//Para configurar os pinos na sequência física P1

Segue abaixo o código em C (blink.c) para fazer o LED piscar a cada 0,5 segundo:

```
#include <stdio.h>
#include <wiringPi.h>
```

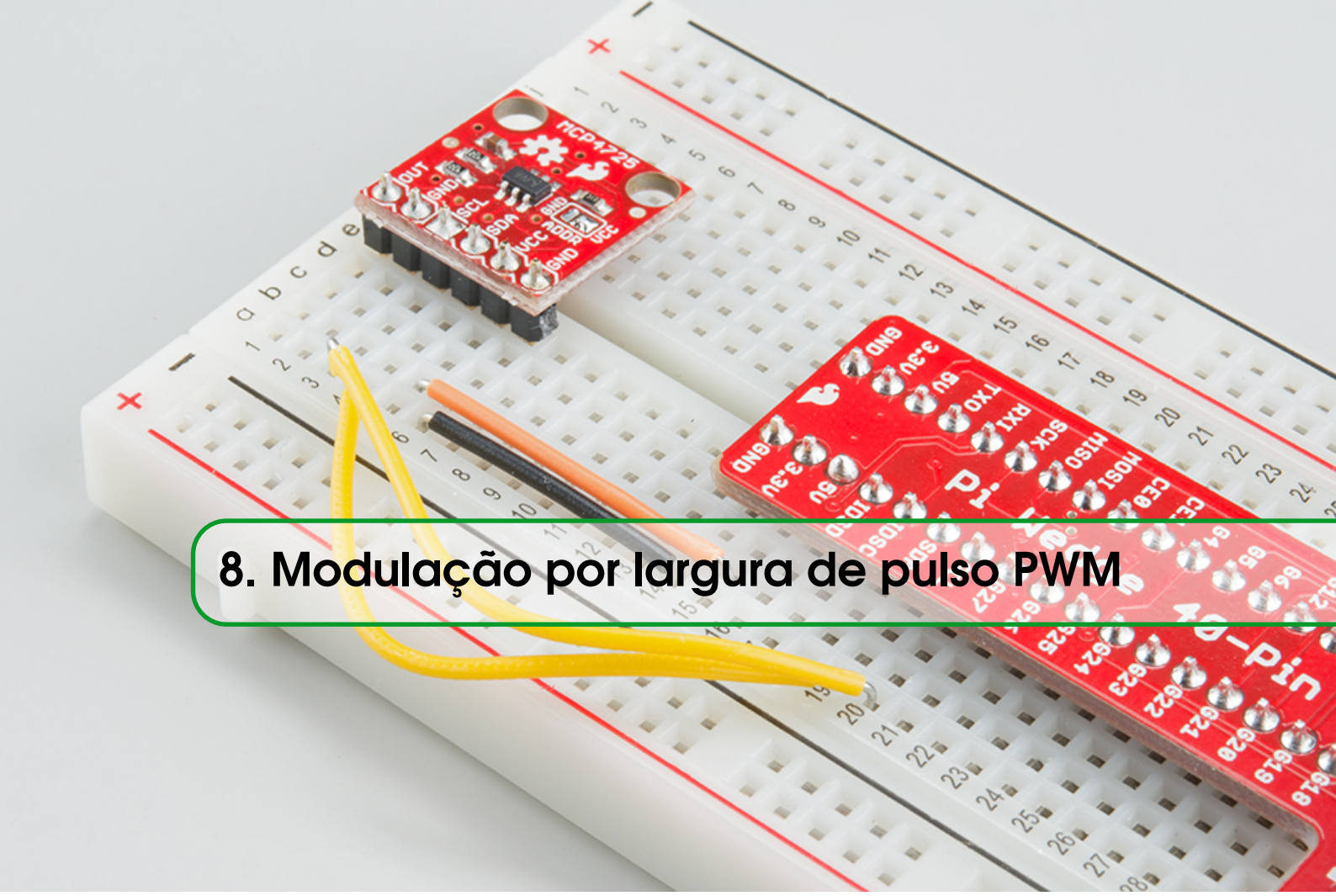
```
//LED-pino 1 do wiringPi éo BCM_GPIO 18 e o pino fisico 12
#define LED 1
int main (void) {
    //printf ("Raspberry Pi blink\n") ;
    wiringPiSetup () ; // configuraão wPi
    pinMode (LED, OUTPUT) ;

    while(1) {
        digitalWrite (LED, HIGH) ; // On
        delay (500) ; // mS
        digitalWrite (LED, LOW) ; // Off
        delay (500) ;
    }
    return 0 ;
}
```

Para compilar o código, é necessário digitar a seguinte linha de comando:

gcc -o blink blink.c –l wiringPi

Para executar, digite o comando:

./blink



8. Modulação por largura de pulso PWM

O código abaixo escreve o valor para o registo PWM para o pino. O PWM definido pelo WiringPi utiliza um pino de PWM, o pino 1 (BMC-GPIO 18, físico 12) e o intervalo fixo do ciclo de trabalho é de 0 a 1024. Outros dispositivos PWM podem ter outras faixas de PWM.

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
int main (void) {
    int bright ;
    printf ("Raspberry Pi wiringPi PWM test program\n") ;
    if (wiringPiSetup () == -1) // configuraão wPi
    exit (1) ;

    pinMode (1, PWM_OUTPUT) ;

    for (;;) {
        for (bright = 0 ; bright < 1024 ; ++bright) {
            pwmWrite (1, bright) ;
            delay (1) ;
```

```
        }
        for (bright = 1023 ; bright >= 0 ; --bright) {
            pwmWrite (1, bright) ; delay (1) ;
        }

    }
    return 0 ;
}
```

8.1 Modulação por largura de pulso (PWM) por Hardware

Para configurar a frequência do sinal PWM por **hardware** é necessário utilizar as funções pwmSetClock() e pwmSetRange(). O intervalo válido para pwmSetClock() varia de 2 a 4095, enquanto o intervalo válido para pwmSetRange() é de até 4096. Nesse caso, para configurar o PWM por hardware igual ao PWM wiringPi poderiam ser utilizados *pwmSetClock(2)*, *pwmSetRange(1024)* e *pwmSetMode*(PWM-MODE-MS).

O clock do PWM por Hardware do Raspberry Pi tem uma frequência base de 19,2 MHz. Essa frequência, divida pelo argumento de pwmSetClock(), é a frequência em que o contador PWM é incrementado. Quando o contador atinge um valor igual ao intervalo definido em pwmSetRange(), ele reinicia para zero. Por exemplo, se for usado o intervalo de 4, ou seja, pwmSetRange(4), você poderia alcançar altas frequências, mas estaria habilitado apenas a configurar o ciclo de trabalho para 0/4, 1/4, 2/4, 3/4 or 4/4.

Enquanto o contador é menor do que o ciclo de trabalho especificado, a saída é alta, caso contrário, a saída é baixa. Para definir o PWM com uma frequência específica, é possível usar a seguinte relação:

Frequência PWM por passo in Hz = 19.2 MHz / pwmClock

Dessa forma, por exemplo, para pwmSetClock(192), a frequência total de PWM por passo será de 100 KHz (19.200 KHz/192), ou seja, o ciclo de trabalho pode ser incrementado a cada 10 us. Para pwmSetRange(100), O período total do ciclo de trabalho será de 1000 us (100x10us).

Os modelos A e B têm um PWM por hardware no pino BCM 18 (wPi 1 ou pino físico 12).

Modelos A+ e B+ têm uma segunda saída PWM por hardware nos pino BCM 18 (wPi 1 ou pino físico 12) e o pino BCM 13 (wPi 23 ou pino físico 33), testado no exemplo abaixo

Vídeo auxiliar: https://www.youtube.com/watch?v=hvH2pO4IsH4

Exemplo:

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

```
int main (void){
    int bright;
    printf ("Gerando sinais PWM por configuracao de Hardware\n") ;

    // Inicializa o sistema wiringPi para usar o pinos de GPIO BCM
    if (wiringPiSetupGpio() == -1) // configuraão BCM GPIO
        exit (1) ;
    // seta o modo do pino BCM GPIO para INPUT, OUTPUT ou PWM_OUTPUT
    pinMode(18,PWM_OUTPUT);
    pinMode(13,PWM_OUTPUT);//Segundo PWM

    // Configura o PWM para o modo mark:space // (PWM_MODE_MS)
    pwmSetMode(PWM_MODE_MS);

    // seta o divisor para o clock do PWM
    pwmSetClock(192);//frequncia total de PWM = 100 KHz

    // seta o registrador de intervalo no gerador de PWM
    pwmSetRange (5);

    while (1) {
        //testar: pwmSetRange (100); bright < 100 e delay(10);
        //testar: pwmSetRange (10); bright < 10 e delay (100);
        //testar: pwmSetRange (5); bright < 5 e delay (200);
        for (bright = 0 ; bright < 5 ; ++bright){
            pwmWrite (18, bright) ; pwmWrite (13, bright) ;
            delay (200) ;
        }
        for (bright = 5 ; bright >= 0 ; --bright){
            pwmWrite (18, bright) ; pwmWrite (13, bright) ;
            delay (200) ;
        }
    }
return 0;
}
```

O gerador de PWM por hardware pode ser executado em dois modos - "balanced"e "mark:space", que é mais comum, no entanto, o modo padrão do Pi é "balanced". É possível alternar entre os modos, fornecendo o parâmetro:

PWM-MODE-BAL ou PWM-MODE-MS.-> pwmSetMode(PWM-MODE-MS);

9. Comunicação Serial entre Pic e Raspberry

Neste tópico, são descritas duas das formas de comunicação mais utilizadas em sistemas embarcados, sendo elas a comunicação serial e a interface USB. Para realizar a comunicação serial entre um microcontrolador PIC e Raspberry Pi, é preciso utilizar os pinos Ground, o GPIO 14 (TX ou pino físico 8) e o GPIO 15 (RX ou pino físico 10), conforme as Figura 9.1 e Figura 9.2.

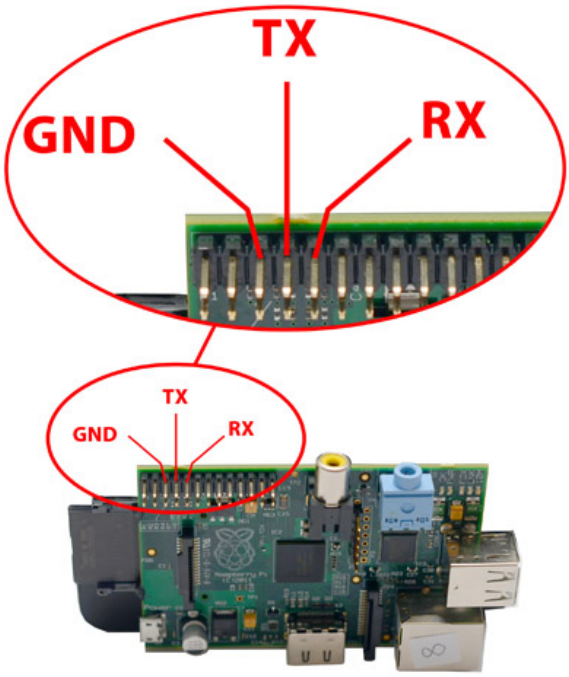


Figura 9.1: Pinos necessários para comunicação serial.

A Figura 9.2 ilustra o circuito básico para comunicação serial entreo o Rpi e o PIC SanUSB. Esse circuito funciona também para gravação online do microcontrolador PIC através da interface USB do Rpi. Mais detalhes no capítulo 11 (Gravando o PIC via USB utilizando RPi).

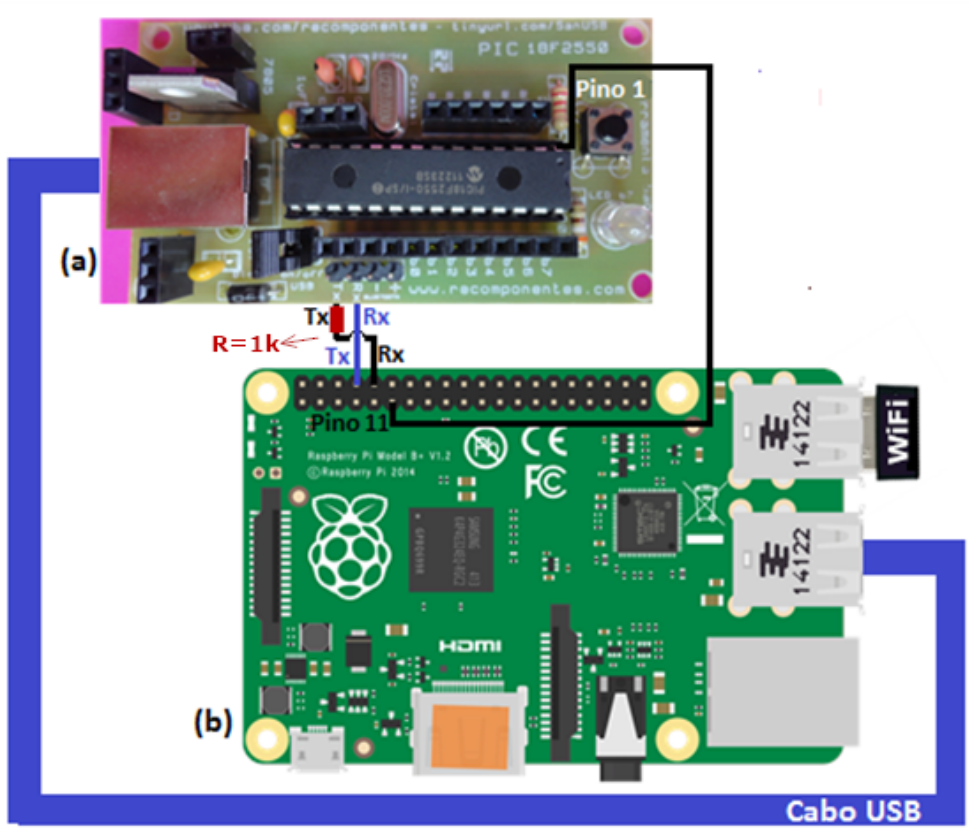


Figura 9.2: Conexão serial entre a placa SanUSB (a) e o Raspberry Pi (b).

No Raspberry Pi 1 e 2, a porta serial equivalente COM1 é encontrado nos pinos GPIO 14 e GPIO 15 e é endereçada por /dev/ttyAMA0. No Rpi 3, essa interface de hardware serial (*uart*), endereçada por /dev/ttyAMA0, foi colocada para o *Bluetooth Low Energy*. Dessa forma, a porta serial dos pinos GPIO 14 e GPIO 15 foi endereçada para uma segunda porta serial /dev/ttyS0, também chamada de "*mini uart*", que é emulada por software. As taxas de transmissão da /dev/ttyS0 são calculadas por software a partir da frequência núcleos de CPU. Assim, se a CPU for sobrecarregada, é possível corromper a comunicação serial. Resumindo, as portas em um Raspberry Pi 3 na configuração inicial são:

/dev/ttyAMA0 -> serial1 (Bluetooth).

/dev/ttyS0 -> serial0 (porta serial GPIO)

Se o usuário quiser a porta serial /dev/ttyAMA0 de volta no padrão GPIO, o modo mais simples é através de uma sobreposição com o comando "pi3-miniuart-bt", ou seja, a mini uart (/dev/ttyS0) será utilizada para o Bluetooth e a porta serial /dev/ttyAMA0 será redirecionada para nos pinos GPIO 14 e 15. Resumindo:

/dev/ttyAMA0 -> serial0 (porta serial GPIO padrão no Rpi 1 e Rpi 2).

/dev/ttyS0 -> serial1 (Bluetooth).

Para implementar, edite:

sudo nano /boot/config.txt

e adicione no final:

dtoverlay=pi3-miniuart-bt

Salve e reinicie para que as alterações para que tenham efeito. É possível verificar as alterações realizadas em:

ls -l /dev

Para utilizar o bluetooth low energy (BLE) do Rpi3, é recomendável comentar ou retirar a linha dtoverlay=pi3-miniuart-bt do arquivo /boot/config.txt, salvar e reiniciar (reboot) o Rpi. Para configurar e testar automaticamente a porta serial do Rpi instale o minicom e descompacte o serialconfigtest.deb, como abaixo, conecte o pino Tx (pino físico 8) do Rpi ao pino Rx (pino físico 10) do Rpi e execute o firmware disponível em **/home/share/SerialUSB/./serialtest** após a instalação.

sudo apt-get install minicom

sudo wget sanusb.org/tools/serialconfigtest.deb

sudo dpkg -i serialconfigtest.deb

Conecte o pino Tx ao Rx por um fio (*jumper* em cuirto-circuito), como ilustrado na Figura 9.3.

Figura 9.3: Conexão do pinos Tx e Rx do Rpi.

Se aparecer *L received* e *D received* quando o pino Tx estiver conectado com o Rx por um fio, como abaixo, significa que o Rpi enviou o byte "L" pelo Tx e recebeu o mesmo "L" pelo próprio Rx, apresentando no terminal essas indicações, mostrando que a comunicação serial funcionou como ilustrado na Figura 9.4.Mais detalhes em https://github.com/SanUSB-grupo/RpiSerialConfigTest.

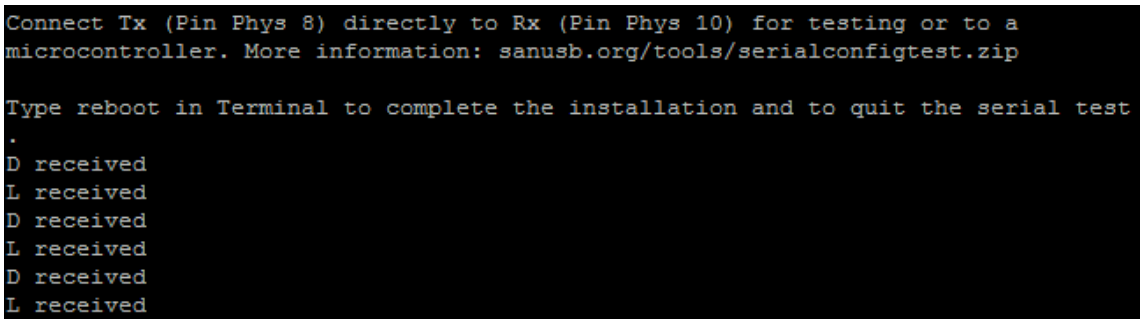


Figura 9.4: Exemplo de teste de comunicação serial.

Quando for conectar o pino Rx do Rpi com tensão de 3,3V e o pino de um outro microcontrolador ou modem bluetooth, atentar para verificar se a tensão dos pinos desse outro dispositivo é 5V. Se for, é recomendável colocar um resistor de 1 K entre o Rx do Rpi e o Tx do modem ou microcontrolador, pois como todos os pinos do Rpi têm um diodo grampeador interno para 3,3V, então caso haja diferença de tensão na comunicação, esta ficará aplicada sobre o resistor de 1K. Um exemplo de aplicação com bluetooth, Rpi e pic pode ser visto em https://www.youtube.com/watch?v=jFxrw3wCvwI.

9.1 Configuração manual serial do Raspberry Pi

1. Faça um backup do arquivo original /boot/cmdline.txt.
   sudo cp /boot/cmdline.txt /boot/cmdline.out.txt
2. Edite o arquivo /boot/cmdline.txt:
   sudo nano /boot/cmdline.txt
   Este arquivo contém:
   dwc_otg.lpm_enable=0 console=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
   Remover os parâmetros que fazem referências a taxa de transmissão default da porta serial UART de 115200 bps, como **console = ttyAMA0,115200** , ficando assim:
   dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
3. Comente a próxima linha em **/etc/inittab**, adicionando na frente dele:
   T0: 23: de respawn: / sbin / getty -L ttyAMA0 115200 vt100
   Agora, Instale o terminal serial minicom para configurar e testar a taxa de comunicação serial:

sudo apt-get install minicom

4. Reinicie o Raspberry Pi para carregar a nova configuração
   sudo reboot

9.2 Configurando manualmente a porta serial com minicom

Para configurar o terminal serial minicom (apt-get install minicom) com outra taxa de transmissão, abra o minicom no terminal (Figura 6.5) com o seguinte comando:

sudo minicom -s

Selecione a opção **Serial Port Setup**(Figura 9.5)

Figura 9.5: Serial Port Setup..

Pressione o botão "**A**" para mudar o dispositivo para "**ttyAMA0**«Enter>, e "**E**«Enter> para modificar a taxa (baud rate) para, por exemplo, 19200 como ilustrado na Figura 9.6.

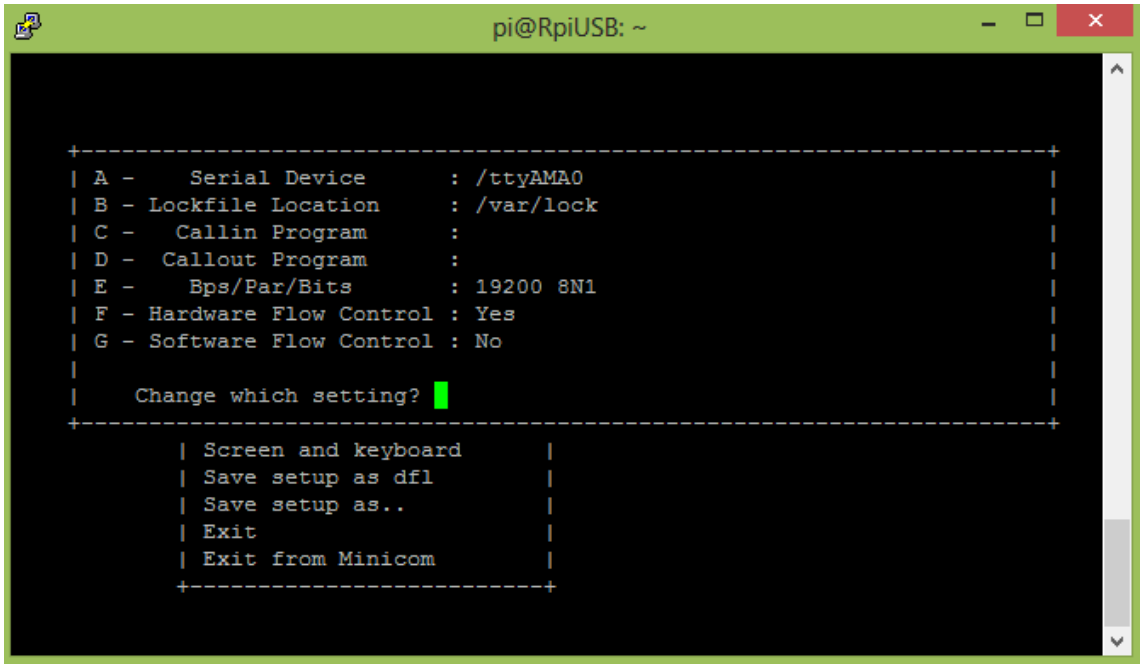


Figura 9.6: Configurações no LXTerminal.

Selecione **Save setup as dfl** para alterar permanentemente a taxa de transmissão e depois pressione **Enter** para voltar ao menu principal (Figura 9.7). Selecione **Exit** para sair do minicom com a nova configuração. Pronto, a interface serial está configurada com a nova taxa de transmissão podendo a partir de agora utilizar os comando wiringPi da biblioteca **wiringSerial.h.**

Figura 9.7: Salvando configurações no LXTerminal.

Para testar a conexão de um dispositivo como um microcontrolador ou um modem bluetooth via porta serial com o terminal do Rpi, abra um terminal serial em um dispositivo Android e execute a linha de comando abaixo no terminal do Rpi.

minicom -b 19200 -o -D /dev/ttyAMA0

Minicom rodando em 19200 bps (Figura 9.8).

Figura 9.8: Configurando taxa de transmissão do terminal.

Quando for digitado um caractere no terminal minicom, ele vai ser recebido pelo dispositivo ou terminal serial Android, que por sua vez, ao enviar um caractere ASCII será recebido pela serial do Rpi e mostrado no minicom. Para sair, pressione “Ctrl+A” liberação em seguida, tecle X. Surgirá a tela da Figura 9.9 e pressione “Yes”.

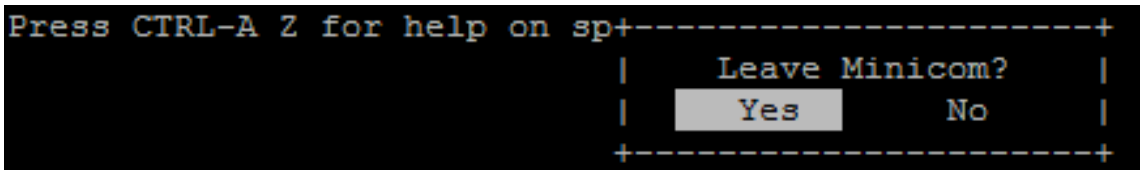


Figura 9.9: Saindo do minicom.

9.3 Funções seriais WiringPi

Para usar, você precisa se certificar que seu programa inclui o seguinte arquivo:

include <wiringSerial.h>

Em seguida, as seguintes funções estão disponíveis:

int serialOpen (char *device, int baud);

Isso abre e inicializa o dispositivo serial e define a taxa de transmissão. Ele define a porta, a taxa, e define o tempo limite de leitura para 10 segundos. O valor de retorno é o descritor de arquivo ou -1 por qualquer erro, caso em que o erro será definido conforme o caso:

void serialClose (int fd);

Fecha o dispositivo identificado pelo descritor de arquivo dado:

void serialPutchar (int fd, unsigned char c) ;

Envia o único byte para o dispositivo serial identificado pelo descritor de arquivo dado:

void serialPuts (int fd, char *s);

Envia a string terminada em nulo ao dispositivo serial identificado pelo descritor de arquivo dado:

void serialPrintf (int fd, char *message, ...);

Emula a função printf sistema ao dispositivo serial:

int serialDataAvail (int fd);

Retorna o número de caracteres disponíveis para leitura, ou -1 para qualquer condição de erro, caso em que errno será definido de forma adequada:

int serialGetchar (int fd);

Retorna o próximo personagem disponível no dispositivo serial. Esta chamada vai bloquear por até 10 segundos, se não há dados disponíveis (quando ele retornará -1):

void serialFlush (int fd);

Isso descarta todos os dados recebidos, ou esperando para ser enviados.

O primeiro exemplo abaixo é um firmware para testar se a comunicação serial está funcionando e para testar um comando do teclado “q” para *quit*:

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <termios.h>
#include <fcntl.h>

char charRecebido;
int fd;
int flagL=0;
int flagD=0;
int pinLed=1;
char x;

int getch(void)
```

```
{
    int ch;
    struct termios oldt, newt;
    long oldf, newf;

    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~(ICANON | ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);
    oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
    newf = oldf | O_NONBLOCK;
    fcntl(STDIN_FILENO, F_SETFL, newf);
    ch = getchar();
    fcntl(STDIN_FILENO, F_SETFL, oldf);
    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    return ch;
}

PI_THREAD (recebe_serial) {
   (void)piHiPri (10) ;

   while(1) {
      charRecebido = serialGetchar(fd);
      fflush(stdout);

      if(charRecebido == 'A' || charRecebido == 'L'){
         flagL = 1;
         digitalWrite (pinLed, HIGH);
         printf("%c received\n",charRecebido);
         serialFlush (fd) ;
      }
      if(charRecebido == 'B' || charRecebido == 'D'){
         flagD = 1;
         digitalWrite (pinLed, LOW);
         printf("%c received\n",charRecebido);
         serialFlush (fd) ;
      }

   }
}
int exists(const char *fname)
```

```
{
    FILE *file;
    if (file = fopen(fname, "r"))
    {
        fclose(file);
        return 1;
    }
    return 0;
}

int main(void){

   if ((fd = serialOpen ("/dev/ttyAMA0", 19200)) < 0){
      fprintf (stderr, "Unable to open serial device: %s\n", strerror (
          errno)) ;
      return 1 ;
   }

   wiringPiSetup() ;
   pinMode(pinLed, OUTPUT) ;

   piThreadCreate (recebe_serial) ;
   digitalWrite (pinLed, HIGH);
   strcat(str1,">/dev/null 2>&1"); //quiet the response
   system(str1);
   fflush(stdout);

   if (!exists("/usr/share/sanusb/sanusb")){
         system("dpkg -i /home/share/SanUSBrpi/sanusb_raspberry.deb");
         system("cp /home/share/SanUSBrpi/sanusb /usr/share/sanusb");
         system("rm /home/share/SanUSBrpi.zip");
                                        }


   fflush(stdout);
   printf("\n\nEstablished serial connection!!!\n");
   printf("\nSerialConfigTest was successfully installed.\n");
   printf("\nConnect Tx (Pin Phys 8) directly to Rx (Pin Phys 10) for
       testing or to a\n");
   printf("microcontroller. More information: sanusb.org/tools/
       serialconfigtest.zip\n\n");
```

```
  fflush(stdout);
  //system("kill $(ps | grep serialtest | cut -d' ' -f2)");

do {
        printf("Type reboot in Terminal to complete the installation and
             to quit the serial test.\n");
        serialPutchar(fd,'D');
        delay (2000);
        serialPutchar(fd,'L');
        delay (2000);
        serialPutchar(fd,'D');
        delay (2000);
        serialPutchar(fd,'L');
        delay (2000);
        serialPutchar(fd,'D');
        delay (2000);
        serialPutchar(fd,'L');
        delay (2000);

        x = getch();
   } while (x != 'q'); //q -> quit

  return 0;
}
```

O vídeo em https://www.youtube.com/watch?v=rfffzww1k6g demonstra um exemplo de comunicação serial entre o Pic e o Rpi, como ilustra a Figura 9.10. Mais detalhes em: sanusb.org/tools/serialconfigtest.zip.

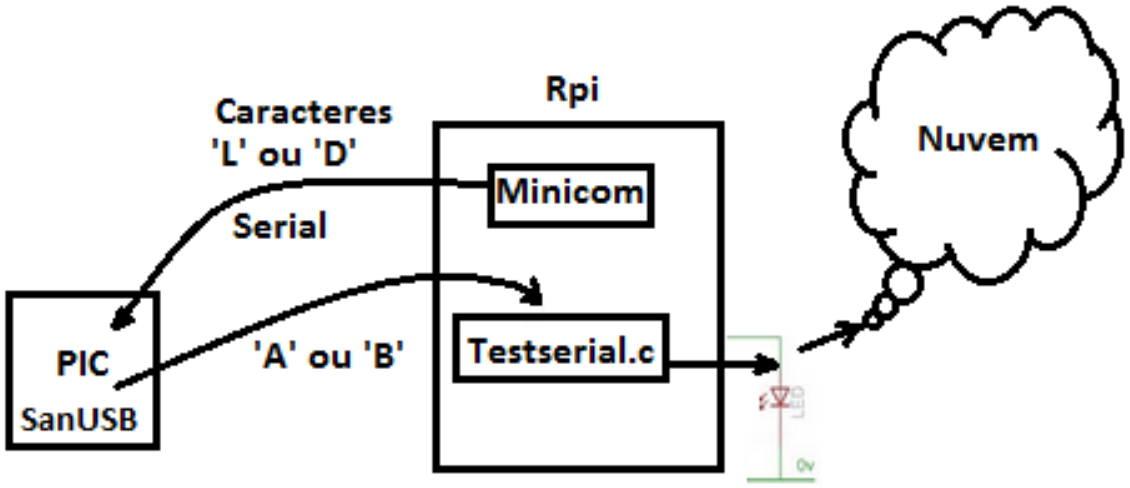


Figura 9.10: Ilustração da comunicação serial entre PIC e RPI.

O Minicom é utilizado, como teste em um terminal do Rpi, para enviar comandos via serial para o PIC (‘L’ ou ‘D’), que por sua vez, retorna via serial outros caracteres (‘A’ ou ‘B’) para o Rpi que também está executando um firmware (Testserial.c). Este firmware recebe os comandos do PIC para acender ou apagar um Led (wPi 1) e postar o estado do Led na nuvem. O código Testeserial.c é mostrado abaixo:

O link seguinte auxilia no próximo código:

https://docs.google.com/spreadsheets/d/1GswcRQUjO6BA2X7jy1LUypzOI2OzVwdZw8Pynnz9FjQ/edit#gid=943056195

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
char charRecebido;
int fd;
int flagL=0;
int flagD=0;
int pinLed=7;
int tempo[7];
int flagPos=0;
int flagInicial=1; //Flag Inicial para iniciar o processo

PI_THREAD (recebe_serial) {
```

```
    (void)piHiPri (10) ;
    while(1) {
        charRecebido = serialGetchar(fd);
        fflush(stdout);
        if(charRecebido == 'A'){
            printf("%c received\n",charRecebido);
            flagL = 1;
            serialPutchar(fd,'D');
            serialFlush (fd) ;
        }
        if(charRecebido == 'B'){
            printf("%c received\n",charRecebido);
            flagD = 1;
            serialPutchar(fd,'L');
            serialFlush (fd) ;
        }
    }
}
int main(void){
    char str1[150] = "curl 'https://docs.google.com/forms/d/19
        K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse?ifq%20&
        entry.289099442=Ligado&submit=Submit'";
    char str2[150] = "curl 'https://docs.google.com/forms/d/19
        K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse?ifq%20&
        entry.289099442=Desligado&submit=Submit'";
    if ((fd = serialOpen ("/dev/ttyAMA0", 19200)) < 0){
        fprintf (stderr, "Unable to open serial device: %s\n", strerror (
            errno)) ;
        return 1 ;
    }
    fflush(stdout);
    printf("Conexão aberta!!!\n");
    fflush(stdout);
    wiringPiSetup() ;
    pinMode(pinLed, OUTPUT) ;
    piThreadCreate (recebe_serial) ;
    digitalWrite (pinLed, LOW);
    while (1) {
        if(flagInicial == 1){
            flagInicial=0;
            serialPutchar(fd,'L');
```

```
        }
        if(flagL == 1){
            flagL = 0;
            digitalWrite (pinLed, HIGH);
            //strcat(str1,">/dev/null 2>&1"); //quiet the response
            //system(str1);
            fflush(stdout);
        }
        if(flagD == 1){
            flagD = 0;
            digitalWrite (pinLed, LOW);
            //system(str2);
            //strcat(str2,">/dev/null 2>&1"); //quiet the response
            fflush(stdout);
        }
        delay (100);
    }
    return 0;
}
```

Firmware do PIC abaixo:
video auxiliar: https://www.youtube.com/watch?v=-9Gf5K0SZ2Y

```
#include "SanUSB48.h"
char flag_on=0, flag_off=0;
#pragma interrupt interrupcao
void interrupcao(){
    unsigned char c;
    if (serial_interrompeu) {
        serial_interrompeu=0;
        c = le_serial();
    }
    switch (c){
        case 'L':
        nivel_alto(pin_b7);
        flag_on=1; //Não imprimir dentro da interrupão
        break;
        case 'D':
        nivel_baixo(pin_b7);
        flag_off=1;
        break;
    }
```

```
}

void main(){
    clock_int_48MHz();
    habilita_interrupcao(recep_serial);
    taxa_serial(19200);
    while(1){
        if(!entrada_pin_e3){Reset();}//pressionar o botão estado para
            gravaão
        if (flag_on){
            flag_on=0;
            swputc('A');
            //sendrw((rom char *)"A\r\n"); //envia string
        }
        if (flag_off){
            flag_off=0;
            swputc('B');
            //sendrw((rom char *)"B\r\n"); //envia string
        }
        sendrw((rom char *)"Pic "); //envia string
        tempo_ms(2000);
        sendrw((rom char *)"serial\r\n"); //envia string
        tempo_ms(2000);
    }
}
```

Agora veja um exemplo de como concatenar os bytes em forma de string do PIC pro Raspberry. Além disso, observe posteriormente como é feito passo a passo para enviar dados via WiFi do microcontrolador para a Google.

Firmware do Raspberry abaixo:

O link seguinte auxilia no próximo código:

https://docs.google.com/spreadsheets/d/1GswcRQUjO6BA2X7jy1LUypzOI2OzVwdZw8Pynnz9FjQ/edit#gid=943056195

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <string.h>
#include <stdint.h>
```

```
int main(void){
    //piThreadCreate (recebe_serial);
    int fd;
    char temp;
    char str1[50]="";
    if ((fd = serialOpen ("/dev/ttyAMA0", 19200)) < 0){
        fprintf (stderr, "Unable to open serial device: %s\n", strerror (
            errno)) ;
        return 1 ;
    }
    fflush(stdout);
    printf("Conexão aberta!\n");
    fflush(stdout);
    while(1) {
        char charRecebido = serialGetchar(fd);
        if(charRecebido == '!'){
            char strPost[200] = "curl \"https://docs.google.com/forms/d
                /19K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse
                ?ifq%20&entry.289099442=
            strcat(strPost, str1);
            strcat(strPost, "&submit=Submit\"");
            //postando
            system(strPost);
            strcpy(str1, "" );
        }
        else{
            temp = charRecebido;
            char str2[3];
            sprintf(str2, "%c", temp);
            strcat(str1,str2);
        }
        fflush(stdout);
    }
}
```

Firmware PIC abaixo:

```
////Resistor de 1K entre os pinos C1 (PWM) e A0(analógico) e um
    capacitor de 10uF
#include "SanUSB1.h"// https://www.youtube.com/watch?v=lB21b3zA4Ac
#pragma interrupt interrupcao //Tem que estar aqui ou dentro do firmware.
    c
```

```
void interrupcao(){
}
unsigned long int resultado, Vresult; //16 bits
unsigned char i=0;

void main(){
    clock_int_4MHz(); //Funão necessaria para o dual clock
    taxa_serial(19200);
    habilita_canal_AD(AN0);
    while(1){
        resultado = le_AD10bits(0);//L canal 0 da entrada analógica com
            resoluão de 10 bits (ADRES)
        Vresult= (resultado *420)/1023;
        //sendrw((rom char *)"Tensao Result= ");
        //swputc('*');
        sendnum(Vresult);
        swputc('!');
        tempo_ms(4000);
        inverte_saida(pin_b7);
    }
}
```

Vale salientar, que no exemplo acima, o PIC ficará enviando os dados do potenciômetro (primeiramente o dado é convertido de analógico para digital) para o Rpi, logos após isso, o Raspberry tratará de enviar essa palavra para o Google Drive. Detalhes para envio de dados para uma planilha do google drive são mostrados nos apêndices.

Para depurar a interface serial do Rpi utilizando comunicação serial com outro dispositivo, basta utilizar o CuteCom (**apt-get install cutecom**) com a versão do Raspbian com interface gráfica, ilustrado na Figura 9.11. Dessa forma é possível visualizar os dados enviados e recebidos pela serial. Vale salientar que a baud rate (taxa de tarnsmissão)utilizada, nesse exemplo, é 19.200 bps (bits por segundo).

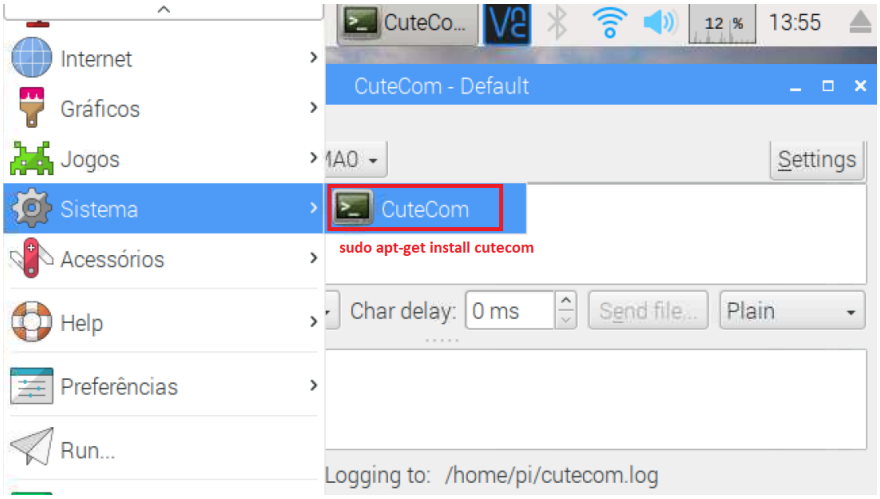


Figura 9.11: Ilustração da interface gráfica do CuteCom

10. Transmissão de vídeo utilizando Ustream e RPi

Neste capítulo é mostrado como realizar transmissão de vídeos via Internet utilizando Rpi.

10.1 Instalação dos pacotes necessários

Primeiramente é recomendável verificar se o seu Raspbian (OS) está atualizado utilizando os comandos:

sudo apt-get update

sudo apt-get upgrade

Caso queira inicialmente testar sua câmera batendo uma foto para comprovar se está funcionando antes de iniciar a transmissão de vídeo, instale as bibliotecas:

sudo apt-get install libav-tools

sudo apt-get install fswebcam

Para tirar foto digite no terminal:

fswebcam photo.jpg

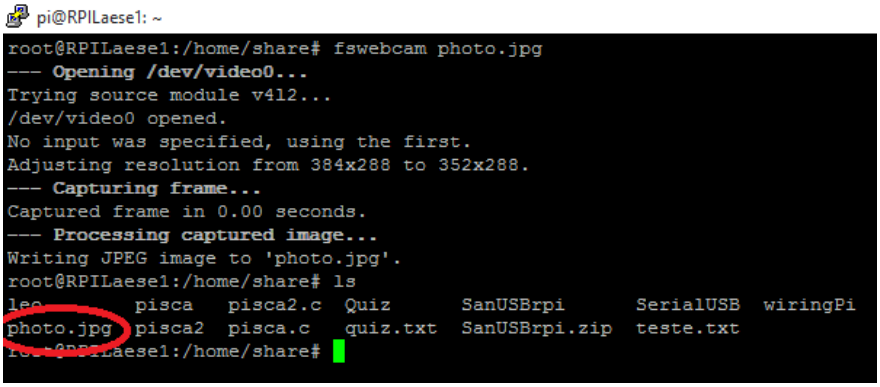


Figura 10.1: Terminal Raspbian

Como pode ser visto na Figura 10.1 a sua câmera bateu a foto com o nome photo.png com sucesso, caso queira bater a foto com outro nome, basta digitar 'fswebcam nome$_da_foto.png'no terminal$.

10.2 Criação da conta no Ustream

Primeiramente acesse o site www.ustream.tv/ e clique na opção sign up localizado no canto superior esquerdo.

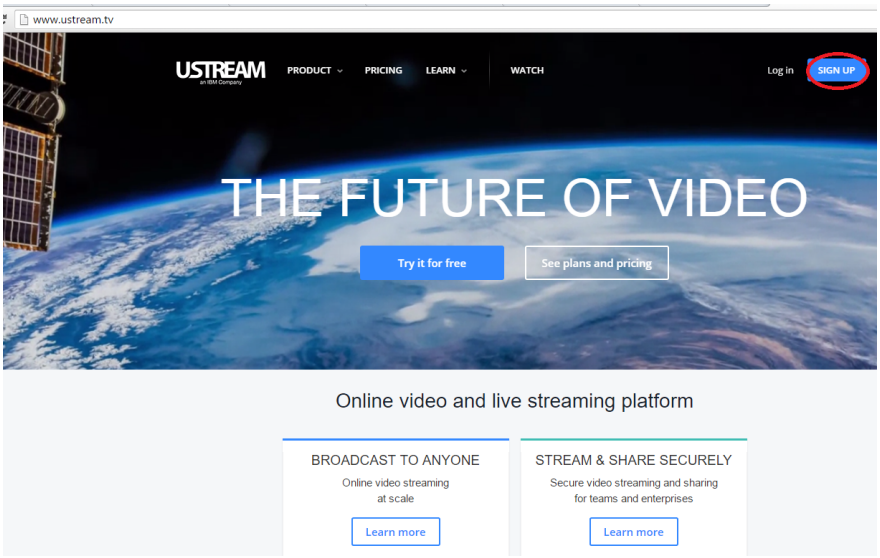


Figura 10.2: Campo Ustream

Logo após essa etapa preencha os campos exigidos que sua conta no Ustream será feita. Há uma outra possibilidade de acessar o Ustream pelo Facebook, caso não queria fazer uma conta.

10.3 Acessando a conta e criando canal

Após feito a conta no Ustream, clique na opção Channel settings para a criação do canal.

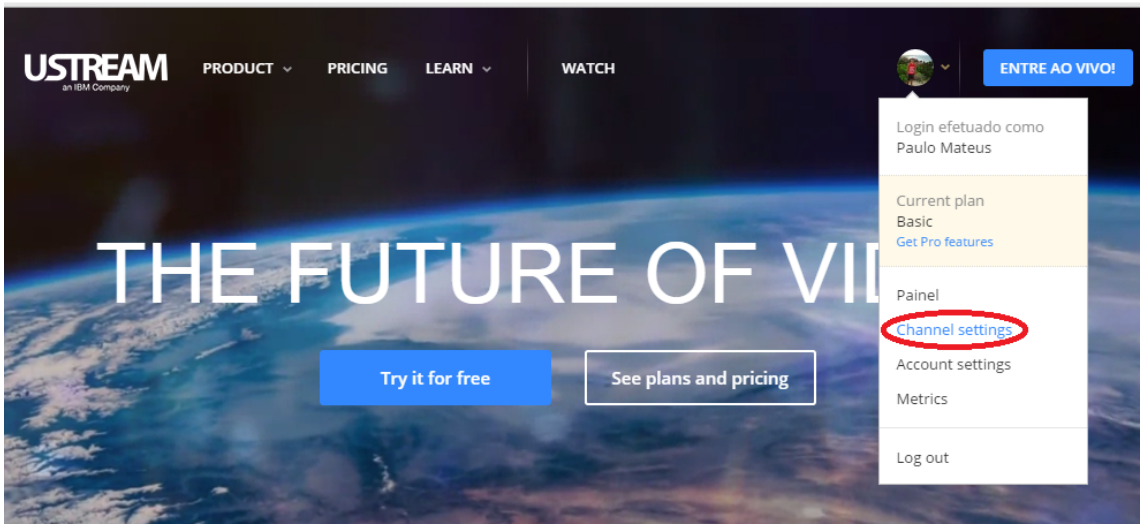


Figura 10.3: Criação de canal Ustream

Em seguida crie o canal e preencha os campos pedidos. A partir desse momento já se tem um canal criado.

10.4 Shell script para filmagem de vídeo

Para a filmagem online do vídeo basta salvar em um arquivo shell o seguinte código abaixo:

!/usr/bin/env bash
avconv -f video4linux2 -s 320x240 -r 15 -i /dev/video0 -metadata title="RPILaese-f flv rtmp:// 0.12345.fme.ustream.tv/ustreamVideo/0.12345/6789

Onde ‘rtmp://0.12345.fme.ustream.tv/ustreamVideo/0.12345’ e ‘6789’ podem ser acessados pela sequencia de passos a seguir:

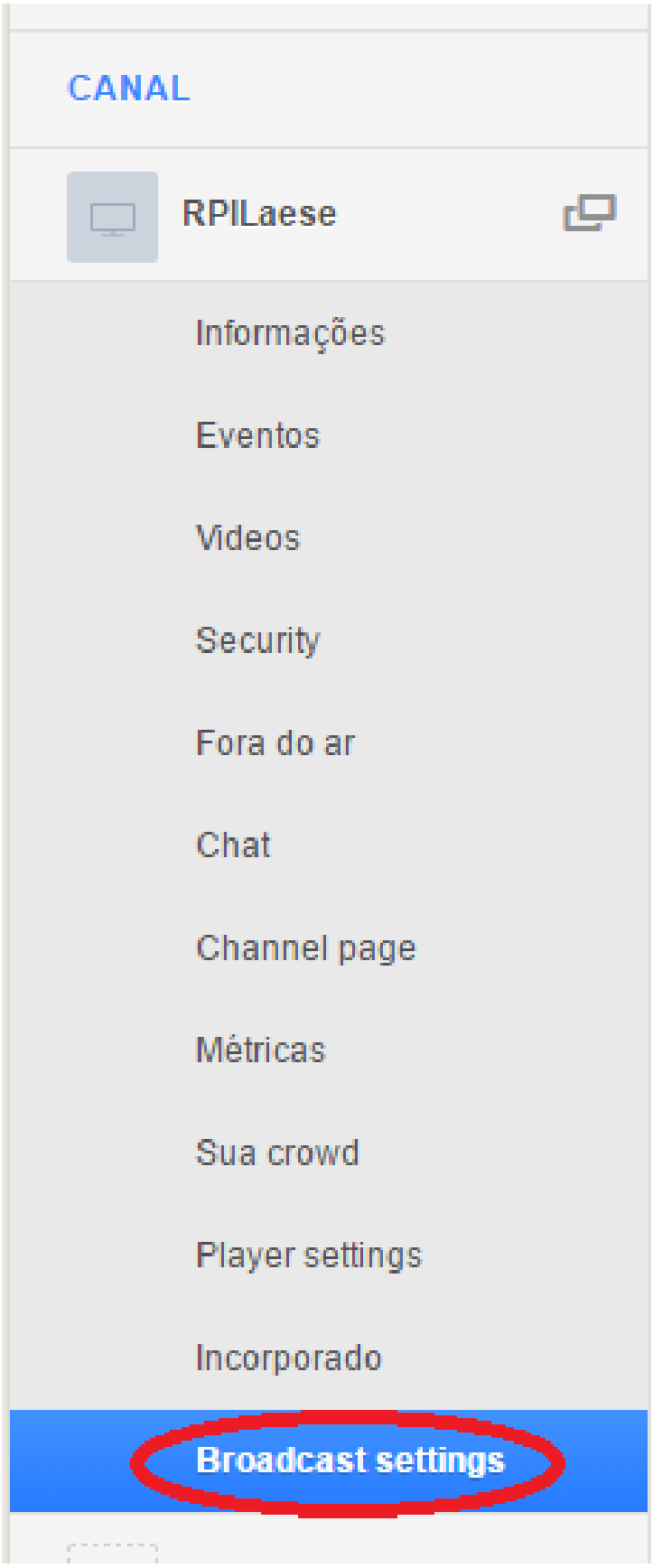


Figura 10.4: Configurações de transmissão

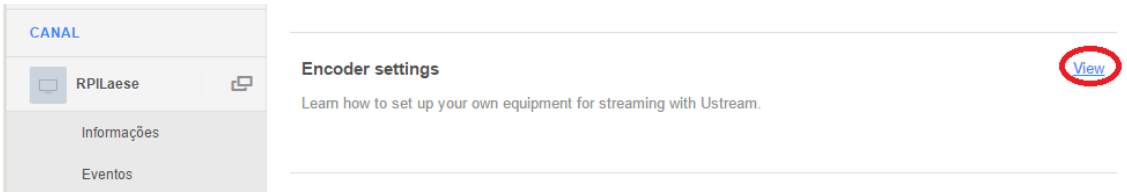


Figura 10.5: Configuração do Codificador

Figura 10.6: Configurações do canal

Lembre-se que antes de executar o programa acima, entre em modo administrador pelo terminal e altere permissões para o programa usando 'chmod 755 nome_do_programa'. Por fim basta executar o programa utilizando o comando 'sh nome_do_programa.sh'. Outro exemplo de script, acessando o ustream pelo facebook:

!/usr/bin/env bash

avconv -f video4linux2 -s 320x240 -r 15 -i /dev/video0 -metadata title="RPILaese-f flv rtmp://1.22699898.fme.ustream.tv/ustreamVideo/22699898/GnLuJnE5AZFAheFpyCDbHd99aTfZNjPt

11. Sistema Embarcado em Jogo Interativo

11.1 Sistema de Jogo Educacional

O sistema do jogo educativo foi elaborado para que possa interagir com o usuário de três formas: através de um circuito com três botões e uma chave desenvolvido para o jogo e acoplado ao RPI, por meio de um teclado de computador fazendo uso de teclas específicas, e através de uma página web.

Nos tópicos abaixo serão descritos a lógica utilizada para o desenvolvimento do jogo, bem como o funcionamento do sistema com as alternativas de interação com o usuário.

11.2 Software do jogo

Para a elaboração do jogo, utilizou-se uma linguagem de programação comum em ambiente de sistemas embarcados, a linguagem C, fazendo o uso da biblioteca wiringPi que permite o acesso à interface GPIO (General Purpose Input/Output) do Raspberry Pi. Essa biblioteca inclui um utilitário de linha de comando que pode ser usado para programar e configurar os pinos de entrada e saída do microcomputador que serão necessários para a comunicação do RPI com o circuito desenvolvido para o jogo [Wiring Pi 2015]. No diagrama abaixo (Figura 11.1) pode-se observar o funcionamento do software utilizado para o desenvolvimento do jogo.

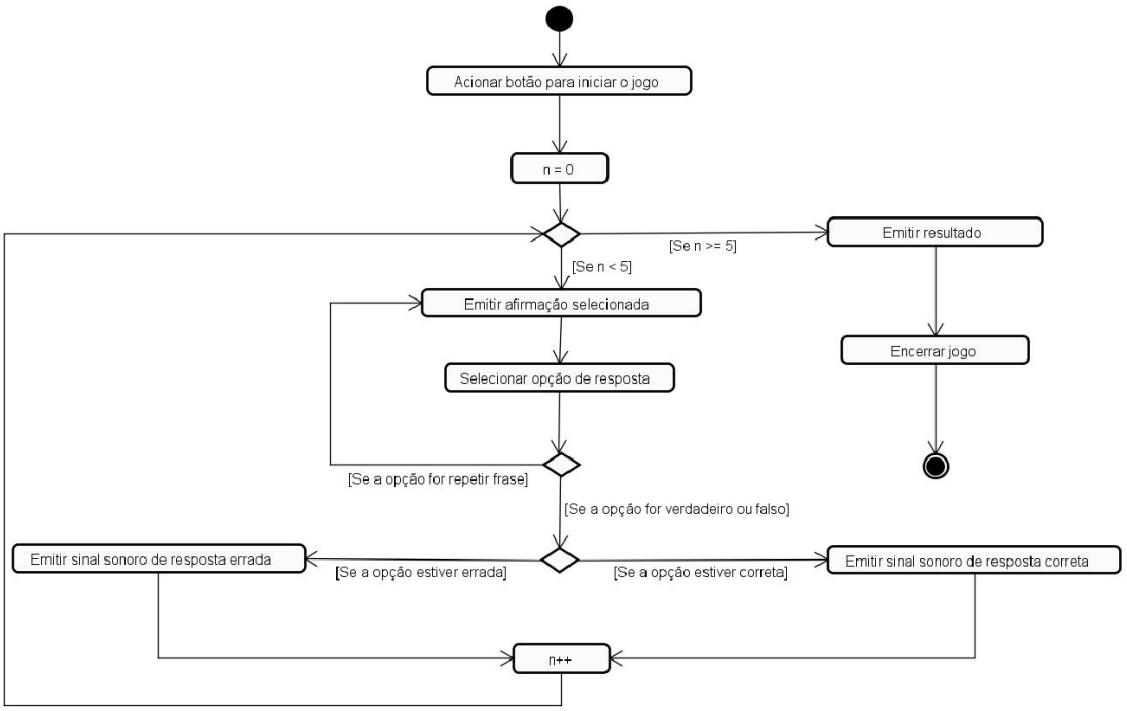


Figura 11.1: Diagrama de funcionamento do sistema.

código em C, segue abaixo:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
//declarando variáveis para os trs botoes do sistema e o sensor de
   movimento
int botaoRepetir = 0;//pino11
int botaoFalso = 2;//pino13
int botaoVerdadeiro = 3;//pino15
int sensorMovimento = 1;//pino12
int contRespostaCorreta = 0;
//vetor de caracteres com o gabarito das afirmaes
char gabarito[100]="
   fvvfvfvvvfvfvfvfffvfffvfvffvfvvvfvfffvvfvvvvvvfffffvfvvfvfffff";
"fvvvvfvvvvvvfvvfvfvvvvvffvfvvvffvvvvfvv";//unir com a linha anterior

//Funcao que analiza a resposta do usuario
void analizarResposta(char strSom[], int afirmacao){
```

```
    char resposta; //Variavel para armazenar a resposta do usuario
    //Loop para esperar a respsota do usuario
    while(1){
        if (digitalRead(botaoRepetir) == 0){
            system(strSom);
        }
        if (digitalRead(botaoFalso) == 0){
            resposta = 'f';
            break;
        }
        if (digitalRead(botaoVerdadeiro) == 0){
            resposta = 'v';
            break;
        }
    }
    //Verifica se a resposta esta correta ou errada
    if (resposta == gabarito[afirmacao-1]){
        system("mpg321 -g 100 /home/share/Quiz/Sons/respostaCorreta.mp3")
            ;
        contRespostaCorreta++;
    }
    else{
        system("mpg321 -g 100 /home/share/Quiz/Sons/respostaErrada.mp3");
    }
}
//Funcao que inicia o jogo.
void quiz(){
    //Contador de afirmaes. Contabiliza o numero de afirmacoes feitas
    int contAfirmacoes = 0;
    //vetor de inteiros utilizado para verificar se uma afirmacao já foi
        dita
    int verifcarRepeticao[100];
    //zera o vetor "verifcarRepeticao"
    memset(verifcarRepeticao,0,sizeof(int)*100);
    //O loop termina quando 5 afirmacoes ja foram feitas
    while(contAfirmacoes < 5){
        //Seleciona aleatoriamente uma afirmacao para ser dita
        int afirmacao = rand()%100 + 1;
        //Verifica se a afirmacoa ja foi dita.
        //Se sim seleciona outra afirmacao.
        //Se nao, armazena a afirmacao no vetor "verifcarRepeticao",
```

```
            incrementa o contador de afirmacoes e reproduz o audio da
            afirmacao
        if (verifcarRepeticao[afirmacao-1]!=afirmacao){
            contAfirmacoes++;
            verifcarRepeticao[afirmacao-1] = afirmacao;
            char strAfirmacao[4];//Variavel para armazenar a afirmacao
                como uma string
            sprintf(strAfirmacao, "%i", afirmacao);//Converte o int "
                afirmacao" na string strAfirmacao
            char strSom[200] = "mpg321 -g 100 /home/share/Quiz/Sons/";//
                string que representa o comando para reproduzir o audio
            //Concatenacao de strings
            strcat(strSom, strAfirmacao);
            strcat(strSom, ".mp3");
            system(strSom);//Executando o comando strSom no terminal
            analizarResposta(strSom, afirmacao);
            //O sistema espera 2s para fazer outra afirmacao
            delay(2000);
        }
    }
}
int main (void){
    wiringPiSetup();
    srand( (unsigned)time(NULL) );
    pullUpDnControl(botaoRepetir, PUD_UP);
    pullUpDnControl(botaoFalso, PUD_UP);
    pullUpDnControl(botaoVerdadeiro, PUD_UP);
    pullUpDnControl(sensorMovimento, PUD_UP);
    while (1){
        printf("%d\n",digitalRead(sensorMovimento));
        if (digitalRead(sensorMovimento) == 0){
            system("mpg321 -g 100 /home/share/Quiz/Sons/inicio.mp3");
            delay(2000);
            quiz();
            char strContRespostaCorreta[2];//Variavel para armazenar a
                afirmacao como uma string
            sprintf(strContRespostaCorreta, "%i", contRespostaCorreta);//
                Converte o int "afirmacao" na string strAfirmacao
            char strSom[200] = "mpg321 -g 100 /home/share/Quiz/Sons/
                resultado";//string que representa o comando para
                reproduzir o audio
```

```
            //Concatenacao de strings
            strcat(strSom, strContRespostaCorreta);
            strcat(strSom, ".mp3");
            system(strSom);//Executando o comando strSom no terminal
            contRespostaCorreta = 0;
        }
    }
}
```

Ao iniciar o jogo, o sistema escolhe aleatoriamente uma afirmação a ser emitida através de uma caixa de som anexada ao RPI. As afirmações são arquivos em formato MP3 armazenados no RPI e pode abordar assuntos de várias áreas de conhecimento como história, matemática, informática, português, geografia, entre outras. Após a reprodução da frase, o programa aguarda a resposta do usuário, que pode ser verdadeiro, falso ou repetir. No caso da resposta ser verdadeiro ou falso, o programa confere a resposta com um gabarito existente na sua memória e informa ao usuário se a resposta está correta ou errada. Se a resposta for repetir, o sistema simplesmente reproduz a afirmação novamente. Após cinco afirmações reproduzidas o sistema finaliza o jogo informando ao usuário o resultado final, a quantidades de respostas corretas.

11.3 Circuito Metareciclado para o Jogol

A metareciclagem consiste na desconstrução de materiais tecnológicos para a reconstrução da tecnologia. Os princípios da metareciclagem têm por base o reuso do hardware, a aplicação de softwares livres, o uso de licenças abertas e a ação em rede, buscando a formação de uma ideia sobre a reapropriação de tecnologia objetivando a transformação social. Tal conceito abrange uma gama diversificada de possíveis formas de ações como: captação de computadores usados, operacionalização de laboratórios de Metareciclagem, o uso de softwares livres, e a criação de ambientes de circulação da informação através da internet, passando por todo tipo de experimentação e apoio estratégico e operacional a projetos socialmente engajados [Guimarães e Maurer 2011].

Na Figura 11.2 pode ser observado o esquema do circuito desenvolvido exclusivamente para o jogo. Nele faz presente três botões e uma chave. Cada botão representa uma resposta do usuário ao jogo e a chave é utilizada para inicializar o jogo. Na construção do circuito foi utilizado somente materiais que estavam disponíveis, sendo utilizados os princípios da metareciclagem visando minimizar os custos. O botão utilizado para repetir a afirmação foi substituído por um tipo de chave que realiza a mesma função.

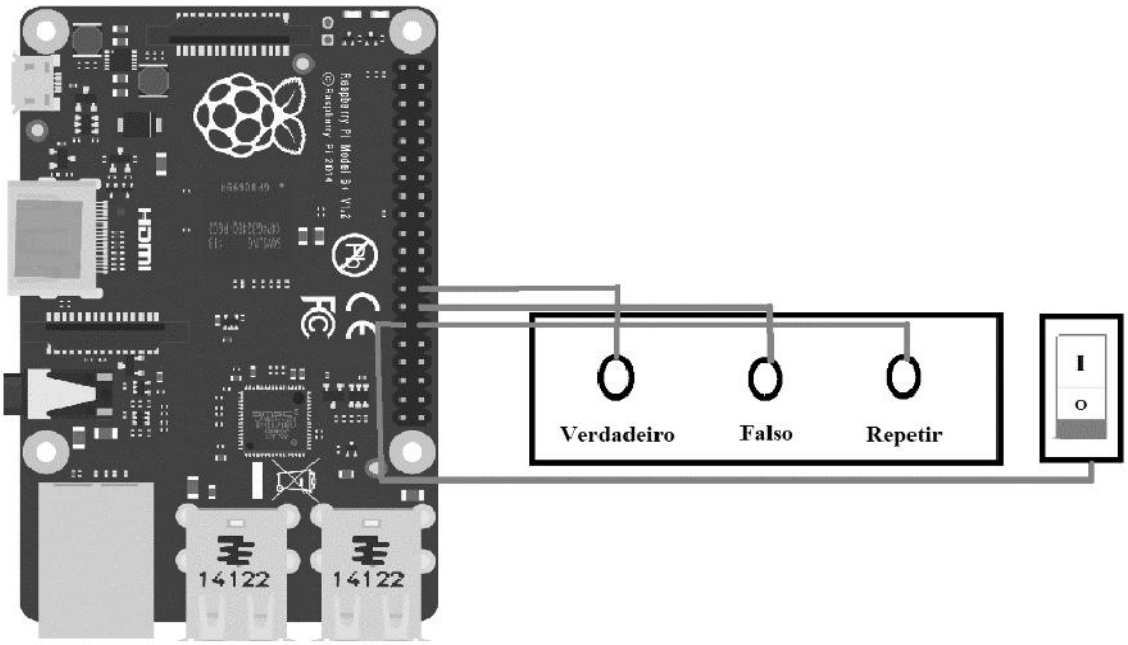


Figura 11.2: Esquema do circuito Jogo educacional.

12. Robô utilizando sistema embarcado

O conceito de robô data dos inícios da história, quando os mitos faziam referência a mecanismos que ganhavam vida. Começando na civilização grega, os primeiros modelos de robô que encontramos eram figuras com aparência humana e/ou animal, que usavam sistemas de pesos e bombas pneumáticas. As civilizações daquele tempo não tinham nenhuma necessidade prática ou econômica, nem nenhum sistema complexo de produtividade que exigisse a existência deste tipo de aparelho.

Mais tarde, cientistas árabes acrescentaram um importante e novo conceito à ideia tradicional de robôs, concentrando as suas pesquisas no objetivo de atribuir funções aos robôs que fossem ao encontro das necessidades humanas. A fusão da ideia de robôs e a sua possível utilização prática marcaram o início de uma nova era.

Leonardo DaVinci abriu caminho a uma maior aproximação ao complexo mundo dos robôs. DaVinci desenvolveu uma extensiva investigação no domínio da anatomia humana que permitiu o alargamento de conhecimentos para a criação de articulações mecânicas. Como resultado deste estudo desenvolvido, surgiram diversos exemplares de bonecos que moviam as mãos, os olhos e as pernas, e que conseguiam realizar ações simples como escrever ou tocar alguns instrumentos.

Com base no contexto acima citado, com o auxílio de um sistema Linux embarcado foi construído um robô com um que desvie de obstáculos utilizando sensor de ultrassom com o auxílio de uma webcam é possível visualizar as imagens obtidas a partir de qualquer computador, dessa forma, é possível realizar um monitoramento com base nas imagens.

12.1 Descrição de Componentes

12.1.1 Sensor de Ultrasom

As utilidades dos sistemas ultrassônicos (Figura 12.1) aliados ao seu baixo custo indicam o seu emprego em muitas aplicações. Os dispositivos baseados no método de pulso-eco, por transdutores elétricos foram utilizados amplamente na medição de distâncias sendo historicamente sua aplicação mais habitual.

Na técnica de pulso-eco, os métodos de medida de distância baseiam-se na determinação do tempo de trânsito que gasta uma onda ultrassônica em percorrer o trajeto de ida e volta. Trata-se então de medir, com a maior precisão possível, o intervalo de tempo transcorrido entre o momento da emissão da onda ultrassônica e o instante de detecção do eco refletido. Por outro lado, para converter o tempo em distância é necessário conhecer a velocidade de propagação das ondas no meio circundante.

Figura 12.1: Sensor Ultrasom.

12.1.2 Sensor de Ultrasom

O servo motor é alimentado com tensões de 5 V e recebe um sinal no formato PWM (Pulse Width Modulation). Este sinal é 0 V ou 5 V. O circuito de controle do servo fica monitorando este sinal em intervalos de 20 ms.

Se neste intervalo de tempo, o controle detecta uma alteração do sinal na largura do sinal, ele altera a posição do eixo para que a sua posição coincida com o sinal recebido. Um sinal com largura de pulso de 1 ms corresponde a posição do servo todo a esquerda ou 0 grau. Um sinal com largura

de pulso de 1,5 ms corresponde a posição central do servo ou de 90 graus. Um sinal com largura de pulso de 2 ms corresponde a posição do servo todo a direita ou 180 graus.

Internamente, como ilustrado na figura 3, um servo motor é composto por diversos elementos:

- Motor – responsável pelo acionamento das engrenagens e eixo principal do servo motor;
- Engrenagens – responsáveis pela redução da rotação do motor e aumento do torque;
- Encaixe de saída – conexão de saída para controle;
- Potenciômetro – usado para monitorar a posição do servo motor;
- Circuito de controle – base do funcionamento do servo motor, monitora a saída do potenciômetro e a ativação do motor interno para manter a posição determinada pela entrada.

O controle do servo motor é obtido por um sinal de entrada que apresenta níveis de tensão TTL e que especifica a sua posição. O formato deste sinal segue a modulação PWM (Pulse Width Modulation ou Modulação por largura de pulso), conforme ilustrado na figura 12.2. Uma informação é codificada em modulação PWM através da largura do pulso em nível alto em relação ao período total de oscilação, ou seja, através do seu fator de forma (duty cycle).

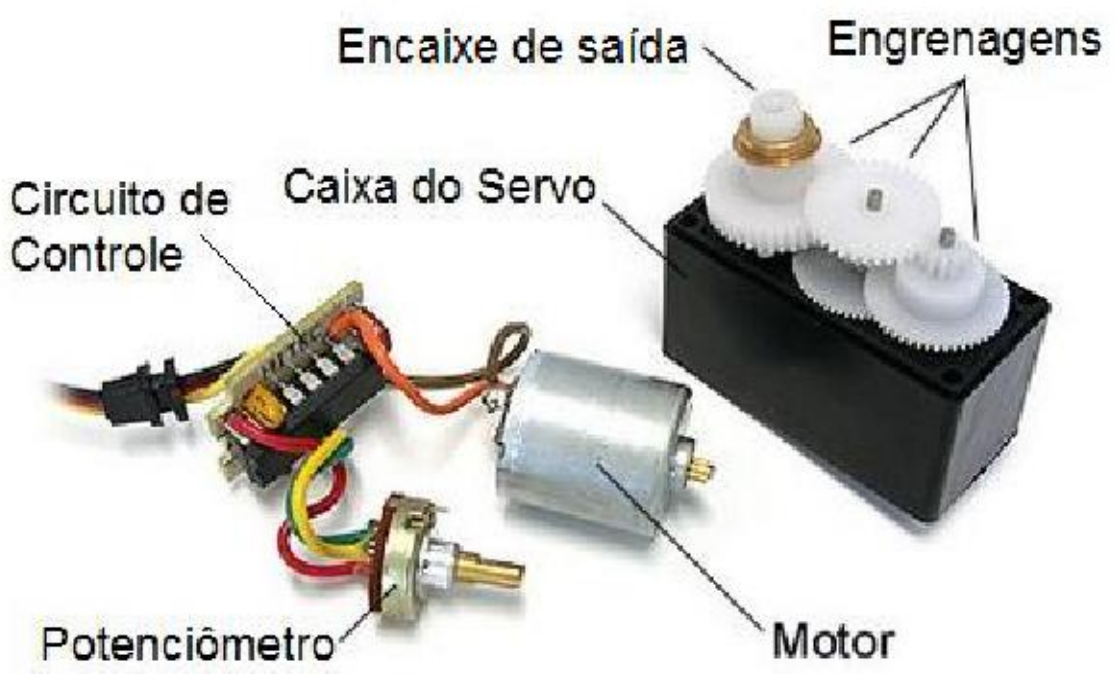


Figura 12.2: Servo Motor

Nesse sistema está sendo utilizado o PWM por software com a utilização da biblioteca wiringPi, utilizando uma frequência de 50Hz e pulsos com largura entre 1 e 2 ms, como mostra a figura 12.3.

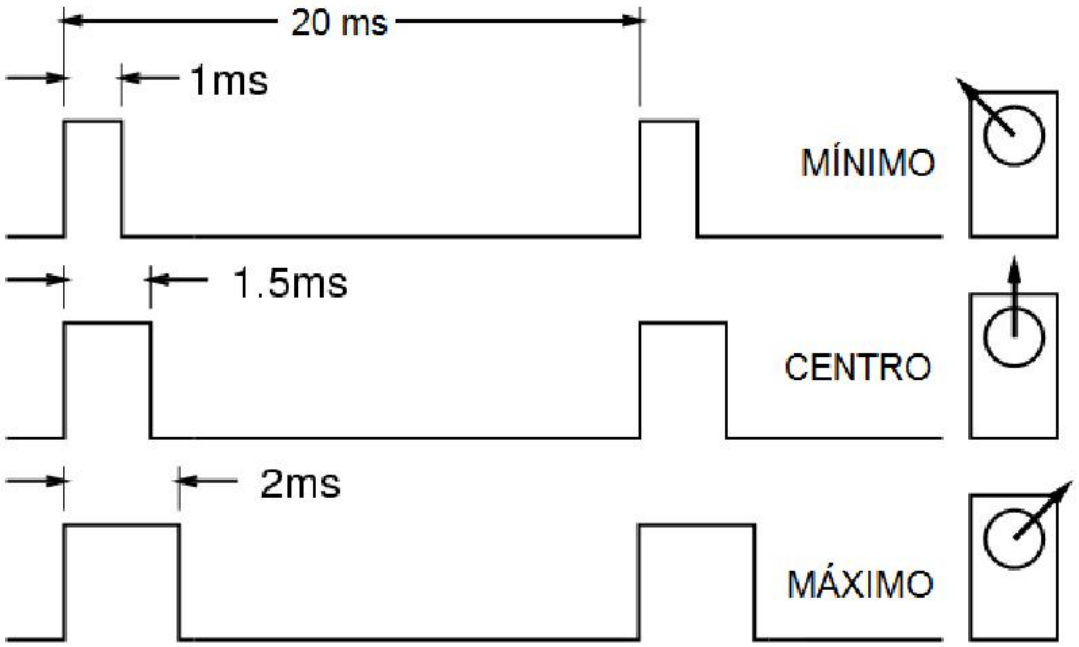


Figura 12.3: Sinais de controle de um Servo Motor.

12.2 Funcionamento do Sistema

Inicialmente o raspberry pi lê a distância entre o robô e um objeto mais próximo, dessa forma, se a distância encontrada for maior que a distância mínima estabelecida, o robô continua sua trajetória para a frente, já se a distância encontrada for menor que a mínima estabelecida, significa que um obstáculo foi encontrado ativando o servo motor que controla a posição do ultrassom, é realizada a medição das distâncias da esquerda e da direita, sendo assim, o robô irá identificar a maior distância e realizar um desvio para o lado escolhido e em seguida continua sua trajetória para a frente até que se encontre um novo obstáculo.

Para realizar o monitoramento por imagem foi utilizada uma webcam comum conectada ao raspberry para que se possa visualizar de um computador as imagens do ambiente, utilizando o pacote denominado motion.

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define PWM_MOTOR_ESQUERDA 2
#define PWM_MOTOR_DIREITA 0
#define PWM_MOTOR_ULTRASOM 1
#define ECHO_PIN 5
```

```
// gpio 24
#define TRIG_PIN 4
#define SPEED_OF_SOUND 34029
double distancia;

float i;
double distanciaUltrasom();

roboFrente(){
    softPwmWrite(PWM_MOTOR_ESQUERDA,20);
    softPwmWrite(PWM_MOTOR_DIREITA,10);
}
roboTras(){
    softPwmWrite(PWM_MOTOR_ESQUERDA,10);
    softPwmWrite(PWM_MOTOR_DIREITA,20);
}
roboParar(){ //Depois verificar se consegue fazer por hardware!!!
    softPwmWrite(PWM_MOTOR_ESQUERDA,0);
    softPwmWrite(PWM_MOTOR_DIREITA,0);
}
roboEsquerda(){
    softPwmWrite(PWM_MOTOR_DIREITA,13);
    //delay(500);
}
roboDireita(){
    softPwmWrite(PWM_MOTOR_ESQUERDA,18);
    //delay(1000);
}
ultrasomEsquerda(){
    softPwmWrite(PWM_MOTOR_ULTRASOM,20);
    delay(3000);
}
ultrasomCentro(){
    softPwmWrite(PWM_MOTOR_ULTRASOM,15);
    delay(1000);
    softPwmWrite(PWM_MOTOR_ULTRASOM,0);
}
ultrasomDireita(){
    softPwmWrite(PWM_MOTOR_ULTRASOM,10);
    delay(3000);
}
```

```
double distanciaUltrasom(){
    (void)piHiPri (10) ;
    //double distanciaUltra;
    digitalWrite(TRIG_PIN, LOW);
    //delay(1000);
    double distanciaUltra;
    int x=0,flagTimeout=0,z=0,cont=0,a=0;
    struct timeval time_start1, time_end1, time_inicio;
    double time_elapsed1,timeout;
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    while (digitalRead(ECHO_PIN) == LOW) {
        gettimeofday(&time_start1, NULL);
    }
// Uma vez que um sinal érecebido, o valor éalterado a partir de baixo (
    LOW) e alta (HIGH), e o sinal irá permanecer elevada durante a duraã
    o do impulso de eco. portanto, precisamos também da ltima alta
    timestamp para o ECHO (time_end).
    while (digitalRead(ECHO_PIN) == HIGH) {
        gettimeofday(&time_end1, NULL);
    }
// Calculamos a diferenca do tempo
    time_elapsed1 = (time_end1.tv_sec + time_end1.tv_usec / 1000000.0) -
        (time_start1.tv_sec + time_start1.tv_usec / 1000000.0);
// calcula a distancia em cm, como tempos o comprimento da ida e volta
    do sinal, e necessario a divisão por 2, pois queremos a distancia do
     ultrasônico até o objeto.
    distanciaUltra = (time_elapsed1 *SPEED_OF_SOUND) / 2;
    //printf("Distancia : %.2f cmn \n", distanciaUltra);
    delay(100);
    /*if(cont<=10){
        distanciaUltra+=distanciaUltra;
    }
    else{
        printf("Distancia : %.2f cmn \n", distanciaUltra/10);
        return (distanciaUltra/10);
    }*/
    return distanciaUltra;
}
PI_THREAD (robo){
```

```
    (void)piHiPri (20) ; //Prioridade da Thread
    while(1){
        distancia=distanciaUltrasom();
        printf("Distancia : %.2f cmn \n", distancia);
        if(distancia<=50){
            roboParar();
            delay(1000);
        }
        if((int)distancia%2==0){
            roboDireita();
            delay(500);
            roboDireita();
            delay(500);
            //roboFrente();
            delay(500);
        }
        else{
            roboEsquerda();
            delay(500);
            roboEsquerda();
            delay(500);
            //roboFrente();
            delay(500);
        }
        //obstaculoEncontrado();
        }else{
            roboFrente();
        }
    }
}

int main (void){
    printf ("Gerando sinais PWM na Raspberry Pi\n") ;
    // Inicializa o sistema wiringPi para usar o pinos de GPIO
    if (wiringPiSetup () == -1) //using wPi pin numbering
        exit (1) ;
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    softPwmCreate(PWM_MOTOR_ESQUERDA,0,200); //Cria ciclos de 20ms
    softPwmCreate(PWM_MOTOR_DIREITA,0,200); //Cria ciclos de 20ms
    softPwmCreate(PWM_MOTOR_ULTRASOM,0,200); //Cria ciclos de 20ms
```

```
    piThreadCreate (robo) ;
    ultrasomCentro();
    while(1){
    }
return 0;
}
```

13. Gravando o PIC via USB utilizando RPi

Para instalação dos programas necessários para a gravação via USB, baixe o arquivo disponível em http://sanusb.org/tools/SanUSBrpi.deb e extraia para a pasta compartilhada /home/share (se não houver a pasta, basta criar com mkdir /home/share). Depois instale o pacote sanusb-raspberry.deb seguindo o tutorial em https://www.youtube.com/watch?v=S30wVi9RWEs. A pasta SanUSBrpi já contém o binário executável válido sanusb para a gravação do microcontrolador via USB com Raspberry Pi utilizando linhas de comando no Terminal.

Para instalar, basta executar diretamentamente o script (./SanUSBrpi.sh), dentro da pasta SanUSBrpi.zip, com permissão anterior de execução (chmod +x SanUSBrpi.sh).

Ou instalar pelo Terminal, utilizando putty ou tightVNC, digitando:

- wget sanusb.org/prog/SanUSBrpi.zip
- unzip SanUSBrpi.zip
- cd SanUSBrpi
- sudo dpkg -i sanusb_raspberry.deb
- cp sanusb /usr/share/sanusb

Para testar se a instalação ocorreu corretamente, verifique o comando de help (./sanusb -h) no LXTerminal(mais exemplos de comandos na Figura 13.1):

- /usr/share/sanusb/./sanusb –h

Resposta:

pi@raspberrypi /usr/share/sanusb/./sanusb –h

sanusb : Free PIC USB Programmer (sanusb.org)

z	Description	Default
-w <file>	Write hex file to device (will erase first)	None
-e	Erase device code space (implicit if -w)	No erase
-r	Reset device on program exit	No reset
-h	Help	

Figura 13.1: Tabela de Comandos

Antes de iniciar a gravação, verifique se o jumper de tensão da porta USB (+5V) está removido, pois a alimentação do microcontrolador vem do pino do 3,3V do Rpi, como o circuito abaixo, para que os dois possam se comunicar na mesma tensão (3,3V) e não ocorrer danos nos pinos por sobretensão em (5V), veja Figura 13.2.

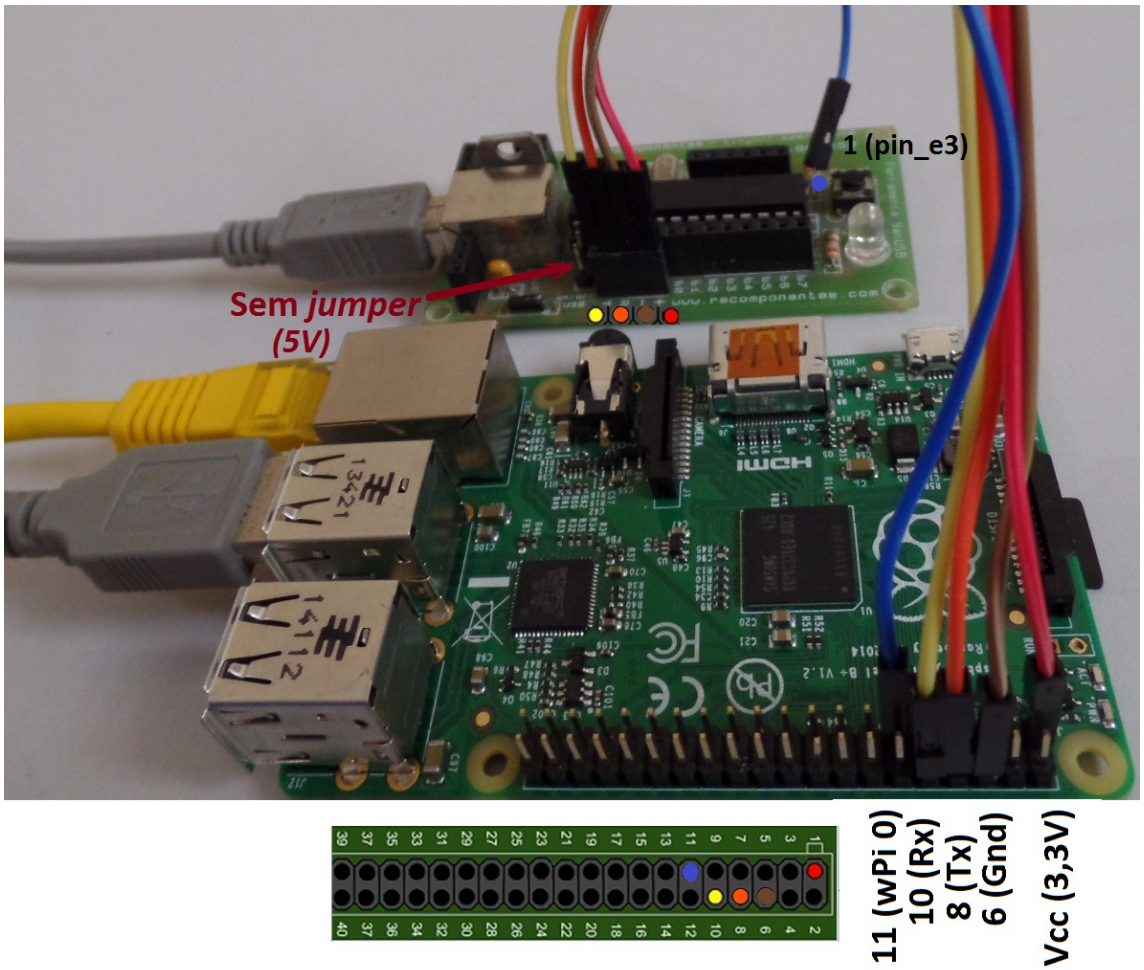


Figura 13.2: Conexão entre Rpi e PIC.

Com o pino físico 11 do Rpi, ou seja, wPi 0 ligado por um fio no pino 1 (pin-e3) do microcontrolador, execute o código **WiringPiGCCParaGravarPICviaUSB** para gravar e regravar o microcontrolador, como mostrado abaixo:

```
pi@raspberrypi ~ $ sudo su
```

```
root@raspberrypi:/home/pi# cd /home/share/SanUSBrpi/
    CodigoWiringPiParaGravarPICviaUSB/
root@raspberrypi:/home/share/SanUSBrpi/CodigoWiringPiParaGravarPICviaUSB
    # gcc -o WiringPiGCCParaGravarPICviaUSB
    WiringPiGCCParaGravarPICviaUSB.c -l wiringPi
root@raspberrypi:/home/share/SanUSBrpi/CodigoWiringPiParaGravarPICviaUSB
    # ./WiringPiGCCParaGravarPICviaUSB
Digite Ctrl+c para sair e gravar o microcontrolador via USB
^C
root@raspberrypi:/home/share/SanUSBrpi/CodigoWiringPiParaGravarPICviaUSB
    # /home/share/SanUSBrpi/./sanusb -w /home/share/SanUSBrpi/exemplo1.
    hex -r
SanUSB device found
Writing '/home/share/SanUSBrpi/exemplo1.hex':....
OK....
```

Em substituição ao código *WiringPiGCCParaGravarPICviaUSB* para regravação do microcontrolador, é possível digitar diretamente no LXTerminal:

gpio mode 0 out <Enter> (configura o pino wPi 0 como saída), depois:

gpio write 0 0 <Enter> (escreve 0 no pino wPi 0 conectado ao pin-e3 do PIC para gravação)

e finalmente:

gpio write 0 1 <Enter> (escreve 1 no pino wPi 0 conectado ao pin-e3 do PIC para operação do microcontrolador após a gravação). Dessa forma, dentro da pasta /home/share/SanUSBrpi/CodigoWiringPiParaGravarPICviaUSB, digite:

```
gpio mode 0 out

gpio write 0 0

gpio write 0 1

#Para gravar digite o "end. do executável" -w "end. do arquivo .hex" -r:

/home/share/SanUSBrpi/./sanusb -w /home/share/SanUSBrpi/exemplo1.hex -r

SanUSB device found

Writing '/home/share/SanUSBrpi/exemplo1.hex':....

OK....
```

Para gravação via USB, basta conectar o cabo USB (5V) e ligar apenas o fio azul entre o pino físico Rpi 11 (Wpi 0) de 3,3 V ao pino e3 do microcontrolador PIC (pino 1 de entrada). Essa conexão, com proteção de tensão (5V <-> 3,3 V), é possível porque os pinos de I/O do Rpi têm um diodo grampeador interno para 3,3V.

Além disso, o pino e3 da placa PIC SanUSB tem um resistor de 2k2 ligado à fonte USB (5V), o que permite a aplicação da tensão de 3,3 V em um lado do resistor conectado ao pino e3 de entrada do PIC, com tensão proveniente do pino Wpi 0 do Rpi através do fio azul, e permite também a tensão de 5V (fonte USB da placa PIC) do outro lado deste resistor de 2k2. Dessa forma, é aplicado a queda de tensão (5V - 3,3 V) no resistor protegendo o pino Rpi 11 (Wpi 0), quando o pino e3 de entrada do PIC recebe nível lógico alto.

Caso seja utilizada comunicação serial, é necessário verificar se o jumper de tensão da fonte USB (+5V) do PIC está removido, pois a alimentação do microcontrolador deve vir do pino de 3,3V do Raspberry pi (Rpi), como o circuito abaixo, para que os dois possam se comunicar na mesma tensão (3,3V) e não ocorrer danos nos pinos do Rpi de 3,3V por sobretensão (5V).

Mais detalhes em SanUSB

A placa PIC SanUSB pode ser construída seguindo o tutorial e os programas estão disponíveis aqui e pode ser adquirida aqui .

14. Instalação de Firmware em nuvem

Após inserir um nome de um perfil no site http://sanusb.org/rs/sanusb.php e realizar o upload de um arquivo a ser compilado e executado no RPI, como os arquivos exemplos da pasta http://sanusb.org/rs/examples.zip, é necessário baixar o SloaderInstall.sh, ou seja, o arquivo SloaderInstall.sh será gerado pelo servidor em http://sanusb.org/rs/sanusb.php após inserir o nome do seu perfil em Insert the plattform name e clicar em Enviar, onde surgirá o link para *download wget sanusb.org/rs/SEU-PERFIL/SloaderInstall.sh*.

Configure o crontab para que o shell script SloaderInstall.sh baixado seja executado na inicialização do sistema. Para isso, siga os passos abaixo:

1. Abra o terminal do *Raspberry Pi* e entre como super-usuário sudo.
2. Para que os comandos que foram copiados para a página do crontab funcionem, é necessário que o shell script nomeado de **SloaderInstall.sh** esteja dentro do diretório **CronSloader**. Se o diretório não existir, crie-o dentro da pasta share, para isso basta digitar no terminal o comando **mkdir/home/share/CronSloader**.
3. Digite o comando **crontab –e**
4. Umas página de edição será aberta, dessa o cursor até o final da página (todas as linhas que começam com são apenas comentários).

5. Copie o texto (comandos) abaixo e cole na página do crontab:
   * * * * * sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 5; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 10; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 15; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 20; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 25; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 30; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 35; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 40; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 45; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 50; sh /home/share/CronSloader/SloaderInstall.sh
   * * * * * sleep 55; sh /home/share/CronSloader/SloaderInstall.sh
6. Agora dê Ctrl + x para sair da página de texto, e depois aperte Y para salvar as mudanças.
7. Depois execute um reboot no RPI para que o cron realize automaticamente as chamadas do script SloaderInstall.sh a cada cinco segundos.Mais detalhes sobre a ferramenta crontab no capítulo 18.

Para verificar os arquivos agendados no crontab, basta digitar *crontab –l.*

15. Interrupções com WiringPi

Com um kernel mais recente de manipulação de interrupção com o GPIO, é possível realizar por uma interrupção pelo wiringpi. Isso libera o processador para fazer outras tarefas enquanto não há uma interrupção. Os GPIOs pode ser configurado para interromper em uma subida, descida ou ambas as bordas do sinal de entrada.

int waitForInterrupt (int pino, int timeOut);

Quando chamado, ele vai esperar por um evento de interrupção para acontecer nesse pino e seu programa será parado. O parâmetro de tempo limite é dado em milissegundos, ou pode ser -1, o que significa que esperar para sempre.

Ex.: if (waitForInterrupt (BUTTON-PIN, -1) > 0) // aguarde até interromper dentro

{...} //de uma thread

O valor de retorno é -1 se ocorreu um erro (e errno será definido apropriadamente), 0 se esgotou, ou 1 em um evento de interrupção bem-sucedido.

Antes de chamar waitForInterrupt, primeiro você deve inicializar o pino GPIO e, atualmente, a única maneira de fazer isso é usar o programa GPIO, seja em um script, ou usando o system de dentro do firmware.

Ex.: system ("gpio edge 17 falling") ; // interrupcao por borda de descida

Outro exemplo. Para uma interrupção de queda de no pino 0, precisamos executar:

system ("gpio edge 0 falling") ;

int wiringPiISR (int pin, int edgeType, void (*function)(void)) ;

Esta função registra uma função de interrupções recebidas no pino especificado. O parâmetro

é edgeType ou INT-EDGE-FALLING, INT-EDGE-RISING, INT-EDGE-BOTH ou INT-EDGE-SETUP. Se for INT-EDGE-SETUP então não inicialização do pin vai acontecer - é assumido que você já tenha configurado o pino em outros lugares (por exemplo, com o programa de GPIO), mas se você especificar um dos outros tipos, então o pino será exportado e inicializado como especificado. Isto é conseguido através de uma chamada adequada para o programa utilitário GPIO, por isso, precisam estar disponíveis.

O número PIN é fornecido no modo atual - modos wiringPi, BCM-GPIO ou Sys nativas. Esta função irá trabalhar em qualquer modo, e não precisa de privilégios.

A função será chamada quando a interrupção é disparada. Quando ele é acionado, é desmarcada na despachante antes de chamar sua função, por isso, se um disparo subsequente de interrupção antes de terminar o seu manipulador, então não vai ser desperdiçada. (No entanto, só pode acompanhar mais uma interrupção, se mais de uma interrupção de incêndios, enquanto um está sendo tratado em seguida, eles serão ignorados).

Esta função é executada em uma prioridade alta (se o programa é executado usando sudo, ou como root) e executa simultaneamente com o programa principal. Ele tem acesso total a todas as variáveis globais, arquivos abertos e assim por diante. Veja o exemplo de programa wiringPi **isr.c** para obter mais detalhes sobre como utilizar este recurso.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <wiringPi.h>

// Use GPIO Pin 17, which is Pin 0 for wiringPi library

#define BUTTON_PIN 0

// the event counter
volatile int eventCounter = 0;

// ----------------------------------------------------------------------

void myInterrupt(void) {
   eventCounter++;
}

// ----------------------------------------------------------------------

int main(void) {
  // sets up the wiringPi library
```

```
  if (wiringPiSetup () < 0) {
      fprintf (stderr, "Unable to setup wiringPi: %s\n", strerror (errno)
          );
      return 1;
  }

  // set Pin 17/0 generate an interrupt on high-to-low transitions
  // and attach myInterrupt() to the interrupt
  if ( wiringPiISR (BUTTON_PIN, INT_EDGE_FALLING, &myInterrupt) < 0 ) {
      fprintf (stderr, "Unable to setup ISR: %s\n", strerror (errno));
      return 1;
  }

  // display counter value every second.
  while ( 1 ) {
    printf( "%d\n", eventCounter );
    eventCounter = 0;
    delay( 1000 ); // wait 1 second
  }

  return 0;
}
```

16. Tarefas concorrentes com wiringPi

16.1 PI-THREADS NO LINUX

As bibliotecas de thread POSIX são APIs baseadas em padrões para C/C++. Estas permitem gerar um novo fluxo de processo concomitante. É mais eficaz em sistemas multiprocessador ou multi-core, onde o fluxo do processo pode ser programado para serem executadas em outro processador ganhando assim velocidade através de processamento paralelo real. Para mais detalhes acesse os links abaixo:

http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html#CREATIONTERMINATION

http://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example/

Pthreads são uma maneira simples e eficaz de criar uma aplicação multi-threaded. Esta introdução aos pthreads mostra a funcionalidade básica de executar duas tarefas em paralelo e fusão de volta para um único segmento em que o trabalho foi feito.

Aplicações multi-thread permitem que duas ou mais tarefas a serem executadas simultaneamente (ou seja: ao mesmo tempo). Quando uma thread é criada usando piThreadCreate, tanto a thread original quanto a nova thread compartilham a mesma memória; é como fazer duas chamadas de função, ao mesmo tempo.

16.1.1 Processamento simultâneo (multi-threading)

O wiringPi tem uma interface simplificada para a implementação Linux de threads POSIX, bem como um mecanismo (simplificado) para acessar (exclusões Mútuas) do mutex.

Usando essas funções você pode criar um novo processo (uma função dentro de seu programa

principal), que decorre em simultâneo com o seu programa principal e utilizando os mecanismos mutex, passar com segurança variáveis entre eles.

int piThreadCreate (name);

Esta função cria um segmento que é uma outra função em seu programa previamente declaradas usando a declaração PI-THREAD. Esta função é então executado simultaneamente com o programa principal. Um exemplo pode ser a de ter essa função de espera para uma interrupção enquanto o programa carrega em fazer outras tarefas. A linha pode indicar um evento, ou ação por uso de variáveis globais para nos comunicarmos com o programa principal, ou de outros tópicos.

Funções de Tópicos são declaradas como segue:

PI_THREAD (myThread){

.. o código aqui é executado simultaneamente com

o programa principal em um loop infinito

}

Abaixo está um firmware que alterna três leds em modo independente e concorrente através de três threads criadas.

Funções: https://projects.drogon.net/raspberry-pi/wiringpi/functions/

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

PI_THREAD (Task_1)
{
  (void)piHiPri (10) ; // Seta como alta prioridade
  while(1)
  {
        digitalWrite(1,HIGH); delay(1000); //pino fsico 12
        digitalWrite(1,LOW); delay(1000);
  }
}
PI_THREAD (Task_2)
{
  (void)piHiPri (10);
    while(1)
  {
        digitalWrite(4,HIGH); delay(300); //pino fsico 16
        digitalWrite(4,LOW); delay(300);
  }
}
```

```
PI_THREAD (Task_3)
{
  (void)piHiPri (10);
    while(1)
  {
       digitalWrite(5,HIGH); delay(100); //pino fsico 18
       digitalWrite(5,LOW); delay(100);
  }
}

int main (void)
{
  wiringPiSetup() ;

   pinMode(1, OUTPUT); // system ("gpio mode 1 out") ;
   pinMode(4, OUTPUT); // system ("gpio mode 4 out") ;
   pinMode(5, OUTPUT); // system ("gpio mode 5 out") ;

  piThreadCreate (Task_1) ;
  piThreadCreate (Task_2) ;
  piThreadCreate (Task_3) ;

  printf ("Exemplo de Multithread \n") ; fflush (stdout) ;

  while(1){//main fica parado aqui
          }
 return 0 ;
}
```

Considerndo a aplicação de tarefas concorrentes em um microcontrolador PIC USB, através de um RTOS (sistema operacional em tempo real), o firmware abaixo também pode ser utilizado para processar três tarefas.

```
#include "SanUSB48.h" //Exemplo com 4 tarefas (task) concorrentes (
    paralelas)
#include <osa.h> //Vdeo-aula: https://www.youtube.com/watch?v=
    s6BG8ZN0aDk
//Evitar uso de outros laos dentro das tasks (como for, do -while, etc
    .!)
```

```
#pragma interrupt interrupcao
void interrupcao(){

    if (PIR1bits.TMR1IF)
    {
        PIR1bits.TMR1IF = 0;
        TMR1H = 0xD8;
        TMR1L = 0xF0;
        OS_Timer();
    }
}

void PIC_Init(void)
{
    LATB= 0x00;
    TRISB= 0x00;

    T1CON= 0x80; // modo 16 bits
    TMR1H= 0xD8; // 1ms
    TMR1L= 0xF0;

    INTCON = 0;
    INTCONbits.PEIE = 1;
    PIR1bits.TMR1IF = 0; // Flag interrupcao Timer1
    PIE1bits.TMR1IE = 1; // Habilita interrupcao Timer1
    T1CONbits.TMR1ON= 1; // Liga Timer1
}

void Task_1(void)
{
    while(1)
    {
        inverte_saida(pin_b7);
        OS_Delay(1000);
    }
}

void Task_2(void)
{
    while(1)
    {
```

```
        OS_Delay(200);
        LATBbits.LATB6^=1;
    }
}

void Task_3(void)
{
    while(1)
    {
        OS_Delay(300);
        LATBbits.LATB5^=1;
    }
}
/* // OSAcf.h configurado com 4 tasks
void Task_4(void)
{
    while(1)
    {
        OS_Delay(400);
        LATBbits.LATB4^=1;
    }
} */

void main(void)
{
    clock_int_48MHz();

    PIC_Init(); // Configuraes gerais do PIC

    OS_Init();

    OS_Task_Create(1,Task_1); // Criando uma tarefa, prioridade 1
    OS_Task_Create(2,Task_2); // Criando uma tarefa, prioridade 2
    OS_Task_Create(3,Task_3); // Criando uma tarefa, prioridade 3
    //OS_Task_Create(4,Task_4); // Criando uma tarefa, prioridade 4

    OS_EI(); // Habilita interrupcoes
    OS_Run(); // Executa o RTOS
}
```

Abaixo está um firmware que emula um semáforo com passagem de pedestre:

Exemplo de semáforo com passagem de pedestre - Duas tarefas

Funções: https://projects.drogon.net/raspberry-pi/wiringpi/functions/

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>

int flag=0;

PI_THREAD (Task_1)
{
  (void)piHiPri (10) ; // Seta como alta prioridade

  while(1)
  {
    delay(50);
         if (!digitalRead(16))
          flag=1;
  }

}

PI_THREAD (Task_2) // LEDs
{
  (void)piHiPri (10);

  while(1)
  {
    if (flag==0){
        digitalWrite(10,HIGH); //pino fsico 24 VERDE CARRO
        digitalWrite(1,HIGH); delay(1000); //pino fsico 12 VERMELHO
            PEDRESTRE

    }else if (flag==1){
        flag = 0;
        digitalWrite(10,LOW);

        digitalWrite(6,HIGH); delay(3000); //pino fsico 22 AMARELO
        digitalWrite(6,LOW);

        digitalWrite(5,HIGH); //pino fsico 18 VERMELHO CARRO
```

```
      digitalWrite(1,LOW);

      digitalWrite(4,HIGH); delay(3000); //pino fsico 16 VERDE PEDESTRE

      digitalWrite(4,LOW);
      digitalWrite(5,LOW);
    }
  }
}


int main (void)
{
  wiringPiSetup() ;

   pinMode(1, OUTPUT); // system ("gpio mode 1 out") ;
   pinMode(4, OUTPUT); // system ("gpio mode 4 out") ;
   pinMode(5, OUTPUT); // system ("gpio mode 5 out") ;
   pinMode(6, OUTPUT); // system ("gpio mode 6 out") ;
   pinMode(10, OUTPUT); // system ("gpio mode 10 out") ;

   pinMode(16, INPUT); // system ("gpio mode 16 input") ;

   pullUpDnControl(16,PUD_UP);

  piThreadCreate (Task_1) ;
  piThreadCreate (Task_2) ;
  //piThreadCreate (Task_3) ;

  printf ("Exemplo de Multithread \n") ; fflush (stdout) ;

  while(1){//main fica parado aqui
          }

 return 0 ;
}
```

piLock (int keyNum); e piUnlock (int keyNum);

Estes sistemas permitem sincronizar atualizações variáveis de seu programa principal para quaisquer threads em execução em seu programa. keyNum é um número de 0 a 3 e representa uma "chave". Quando um outro processo tenta bloquear a mesma chave, será parado até que o primeiro

processo tenha desbloqueado a mesma chave.

Você pode precisar usar essas funções para garantir que você obtenha dados válidos quando a troca de dados entre o seu programa principal e um fio - caso contrário, é possível que o segmento poderia acordá-up até meio durante a copiar seus dados e alterar os dados - para que os dados que acabam -se a cópia é incompleta, ou inválido. Veja o programa wfi.c no diretório de exemplos para um exemplo.

16.2 Funções Auxiliares

piBoardRev (void);

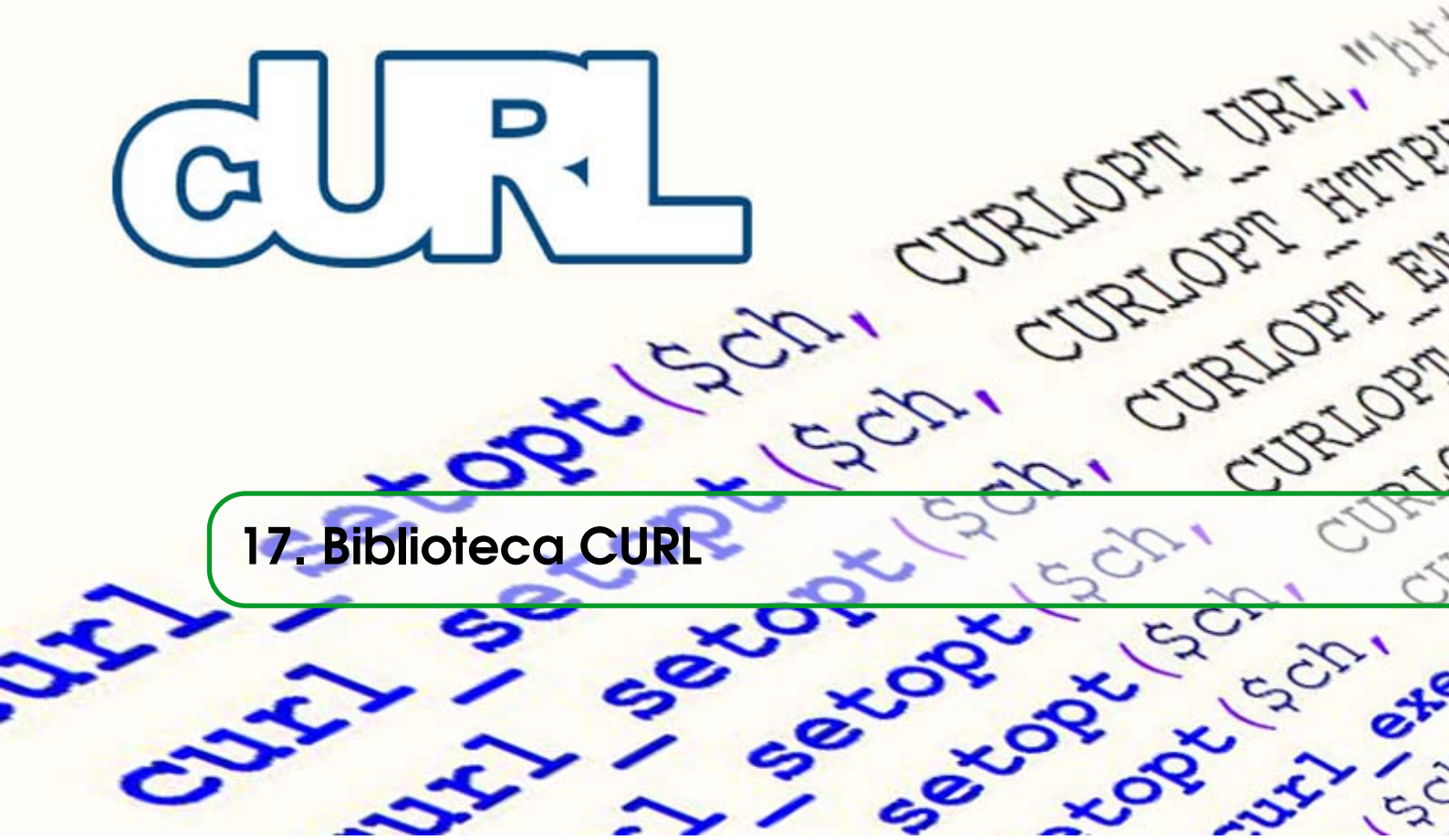
Isso retorna a revisão da placa do Raspberry Pi. Vai ser 1 ou 2. Alguns dos pinos BCM-GPIO mudaram número e função quando se deslocam de bordo revisão 1-2, por isso, se você estiver usando números de pinos BCM-GPIO, então você precisa estar ciente das diferenças.

wpiPinToGpio (int wPiPin);

Isso retorna o número de pinos BCM-GPIO do pino wiringPi fornecido. Leva a revisão da placa em conta.

setPadDrive (grupo int, int valor);

Isso define, ou seja, "força"a criação de drivers para um determinado grupo de pinos. Existem 3 grupos de pinos e a unidade é de 0 a 7.



17. Biblioteca CURL

O curl é uma biblioteca que permite fazer requisições e enviar dados em shell para outros sites e serviços da web. Para instalar basta digitar **sudo apt-get install curl**

17.1 Exemplos: Acessar um siste para post no LXTerminal

Exemplos: Acessar um siste para post no LXTerminal

Curl

https://docs.google.com/forms/d/19K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse?ifq%20&entry.289099442=9&entry.711178561=30&entry.648964283=29&submit=Submit

Ver resultados aqui

Abaixo um código em C para armazenar valores em uma planilha do google drive.

```
#include <stdio.h>
#include <wiringPi.h>
//sudo apt-get install curl
//Ver os dados:
//https://docs.google.com/spreadsheets/d/1
    GswcRQUjO6BA2X7jy1LUypzOI2OzVwdZw8Pynnz9FjQ/edit#gid=943056195
// LED -pino 1 do wiringPi éo BCM_GPIO 18 e o pino fsico 12
#define LED 1

int main (void)
```

```
{
   printf ("Raspberry Pi blink\n") ;

   wiringPiSetup () ;
   pinMode (LED, OUTPUT) ;

   while(1)
   {

system ("curl \"https://docs.google.com/forms/d/19
    K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse?ifq%20&entry
    .289099442=9&entry.711178561=30&entry.648964283=29&submit=Submit \"")
    ;

   digitalWrite (LED, HIGH) ; // On
   delay (5000) ; // ms
   digitalWrite (LED, LOW) ; // Off
   delay (5000) ;
}
   return 0 ;
}
```

Fazer o download de arquivo em um diretório desejado

curl -L –output /home/share/sanswerscontent.hex

http://sanusb.org/rs/$answer/$answer$content.hex

Fazer o upload de arquivo em um servidor online

curl -F "password=sanusb-F "arquivo=@/home/share/blink.c"

http://sanusb.org/rs/upload.php

-F preenche em sequência os dados do formulário upload.php do servidor.

Capturar o conteúdo de um arquivo online em uma variável:

content=$(curl -sl sanusb.org/rs/$answer/view.txt)

18. Ferramentas de sistemas embarcados Linux

18.1 Executando scripts na inicialização do Debian/Ubuntu

Primeiramente, vamos criar um script chamado **meuScript** em /etc/init.d com uma estrutura semelhante à abaixo:

```
#!/bin/bash
#
# /etc/init.d/meuScript

case "$1" in
    start)
        echo "Iniciando servio..."
        # comando para iniciar o servio
        ;;

    stop)
        echo "Parando servio..."
        # comando para parar o servio
```

```
        ;;

    restart)
        echo "Reiniciando servio..."
        # comando para reiniciar o servio
        ;;

    *)
        echo "Operaão inválida"
        ;;
esac
```

Dê permissão de execução para o script:

t chmod +x meuScript

A partir desse momento você já pode executar os comandos abaixo:

/etc/init.d/meuScript start

/etc/init.d/meuScript stop

/etc/init.d/meuScript restart

Para inserir o script na inicialização do sistema, usamos o comando update-rc.d da seguinte maneira:

update-rc.d meuScript defaults

O "meuScript" é o nome do script em /etc/init.d, não o caminho completo a ele, ou seja, não importa em qual diretório você está, apenas coloque o nome do script.

O "defaults" indica que desejamos inserir o script nos runlevels padrões do sistema.

Ao executar esse comando, um warning será mostrado, apesar de o comando ter funcionado. Aparecerá uma mensagem como esta:

"update-rc.d: warning: /etc/init.d/meuScript missing LSB information update-rc.d: see< http://wiki.debian.org/LSBInitScripts**»**

É possível ignorá-lo ou inserir as informações de LSB, o que é mostrado no link exibido no warning. É algo bem simples.

Para remover um script da inicialização:

update-rc.d -f meuScript remove

Agora, é mostrado um exemplo que uso para uma instalação manual do Apachearquivo :

```
#!/bin/bash
#
# /etc/init.d/apache2

### BEGIN INIT INFO
# Provides: apache2
# Required-Start: $local_fs $syslog
# Required-Stop: $local_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start Apache at boot
# Description: Start Apache HTTP Server
### END INIT INFO

# Path to Apache controller
BIN=/usr/local/apache2/bin/apachectl

function startApache()
{
  echo "Starting Apache..."
  $BIN start
}
function stopApache()
{
  echo "Stopping Apache..."
  $BIN stop
}
case "$1" in
  start )
    startApache
    if [ $? -eq 0 ]
    then
      echo "OK."
    fi
  ;;
 stop )
    stopApache
    if [ $? -eq 0 ]
    then
      echo "OK."
```

```
    fi
  ;;
restart )
    stopApache
    sleep 1
    startApache
    if [ $? -eq 0 ]
    then
      echo "OK."
    fi
  ;;
*)
    echo "Invalid option"
    ;;
esac
```

18.2 Raspberry como Servidor

O diretório /var/www/html funciona como o Public-html de servidores online. Após colocar a webpage lá basta acessá-la pelo **IPdoRPI/nomedoarquivo.**

18.3 Instalação de um servidor web e transferência de um Website

Para simplificar este assunto, são divididos os sites em duas categorias: Dinâmicas e Estáticas.

Páginas Estáticas: Paginas que são feitas apenas utilizando HTML e CSS. São construídas utilizando somente a linguagem HTML (com CSS ou Java Script). Essas páginas ficam armazenadas no servidor e quando um internauta chama a página, através de uma solicitação feita ao servidor onde está hospedada, o mesmo entrega a página possibilitando o browser ler conteúdo do html, css, os textos e as imagens e apresentar o conteúdo ao internauta.

Páginas Dinâmicas: Páginas que usam algum tipo de linguagem dinâmica como PHP, .NET entre outras.

Páginas Dinâmicas com código, em PHP por exemplo, funcionam da seguinte forma: O programador monta o site em PHP, os arquivos PHP do site são publicados no servidor. Quando um internauta solicita a página, o servidor processa o PHP e entrega o código já processado em HTML. Isso acontece porque o PHP é pré-processado no servidor e não no browser da máquina do internauta.

Controlar os LEDs remotamente com SSH é interessante, mas a interface (console) não é muito amigável, pois é necessário digitar linhas de comandos. Por isso que é muito utilizado interface gráfica para os projetos.

Programar front-ends para comunicar com os diversos SOs (IOS, Android, Windows Phone, etc.) a um Rpi seria necessário conhecer diferentes linguagens. É por isso que um site é a melhor solução, que é compatível com todos os dispositivos e é necessário "apenas"conhecer um pouco de quatro linguagens: HTML (para o esqueleto da página), CSS (o estilo de página), PHP (linguagem interpretada no servidor) e JavaScript (Interações com o usuário). É importante salientar que JavaScript não tem nada a ver com Java. JavaScript é uma linguagem de programação client-side para auxilio na criação de sites. Ela é utilizada para controlar o HTML e o CSS e manipular comportamentos na página como, por exemplo, o surgimento de submenus ao passar o mouse.

As funções escritas em JavaScript podem ser embutidas dentro do documento HTML, possibilitando o incremento das funcionalidades do seu documento HTML com elementos interessantes. Sendo possivel responder facilmente a eventos iniciados pelo usuário, incluir efeitos que tornem páginas dinâmicas. Dessa form, é possível dizer que JavaScript é mais uma extensão do HTML do que uma linguagem de programação propriamente dita.

Em documentos HTML, a utilização da linguagem JavaScript, se dá sob a forma de funções (applets), as quais são chamadas em determinadas situações ou em resposta a determinados eventos, estas funções podem estar localizadas em qualquer parte do código HTML, a única restrição é que devem começar com a declaração **<SCRIPT>** e termina com o respectivo **</SCRIPT>.**

```
//http://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst

<!DOCTYPE html>
<html>
<body>

<script>
teste=prompt("Digite a senha","");
alert(teste+" seja Bem Vindo!");
</script>

<h1>Meu Primeiro JavaScript</h1>

<script>
    // Comentário dentro de script para uma linha de texto.
    /* Comentário de várias linhas de texto,
    continuaão do comentário de várias linhas */
valor=50
document.write("O Resultado do calculo eh ",(10*2)+valor)
</script>
</br>

<button type="button"
```

```
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

<img id="ImagemBulbo" onclick="changeImage()" src="images/pic_bulboff.
    gif" width="100" height="180">
<p>Clique na Lampada para on/off.</p>

<script>
alert('Clique na Lampada:\nOn -Ligada\nOff -Desligada');
function changeImage() {
    var image = document.getElementById('ImagemBulbo');
    if (image.src.match("bulbon")) {
        image.src = "images/pic_bulboff.gif";
    } else {
        image.src = "images/pic_bulbon.gif";
    }
}
</script>

</body>
</html>
```

Dessa forma, quando o Rpi é configurado como servidor HTML, é necessário instalar apenas o software servidor HTTP Apache (apt-get install apache2), quando é configurado como

servidor PHP, é necessário instalar também o pré-processador PHP (apt-get install php5 libapache2-mod-php5), ou seja, para instalação completa basta, depois de atualizar o Rpi com o comando "sudo apt-get upgrade", digitar "sudo apt-get install apache2 php5 libapache2-mod-php5". Agora é possível testar se o servidor está funcionando, digitando o IP do Rpi em algum browser e surgirá uma página indicando "It works!".

Se não funcionou, em seguida, verifique o IP, tente reinstalar o Apache ou reiniciar o Raspberry Pi. A página com "It´s work!"está mostrando que o servidor Apache está funcionando corretamente, mas não sua extensão PHP. Para verificá-lo, navegue até o diretório "/var/www/"usando o comando "cd /var/www/html". O diretório /var/www/html funciona no Rpi como o Public-html nos servidores convencionais.

Se você usar o comando "ls", você deve ter apenas um arquivo denominado "index.html". Este arquivo corresponde à página"It´s work!". É possível excluí-la com excluí-lo (*"sudo rm index.html"*) e criar um outro chamado "index.php"(use *"sudo nano index.php "*). Em seguida, digite o seguinte texto:

<?
php
phpinfo();
?>

Depois de salvar o editor nano usando o comando (Ctrl + O), e saindo com (Ctrl + X). Agora, se você atualizar seu navegador, você deve ver uma longa página com muita informação sobre o seu servidor e PHP. Se você não fizer isso, verifique o arquivo index.php, tente reinstalar o PHP ou tentar entender o erro exibido na página (Google)

Se Ambas as páginas foram corretamente exibido, então agora você tem um servidor Apache/PHP totalmente funcional, mas usando nano toda vez que é irritante e não muito confortáveis.

Nós realmente precisamos para transferir arquivos de seu computador para o Raspberry Pi. Você pode querer instalar um servidor FTP, mas não é necessário, você já pode transferir arquivos usando o protocolo SFTP. Tudo que você precisa é de SFTP cliente em seu computador como o WinSCP ou Filezilla para Windows, Cyberduck para Mac ou Filezilla para Linux. Se você tentar transferir arquivos antes de ler o que está próximo, você provavelmente vai ter problemas tais como: "Acesso recusado"ou "não pode escrever aqui". É devido ao factthat o pi usuário não possuir o diretório www. Na verdade, se você tentar o comando "ls -l /var/www/html", você verá que somente o root (superusuário) é possuir o diretório html. Você pode (como eu fiz) usar o comando **"sudo chown -R pi /var/www/html"** para alterá-lo ou criar um grupo chamado www-data em que coloca o usuário PI, em seguida, usar o comando **"sudo chown -R www-data /var/www/html "**. A flag -R é recursiva, isso significa que o usuário/grupo não só é proprietária do diretório em si, mas sim para tudo qu estiver dentro (como exemplo o index.php). Para editar o arquivos dentro de /var/www/html digite o comando de permissão **sudo chmod 755 /var/www/html.**

Agora se tem o servidor pronto para trabalhar e receber páginas da web.

Dica:
- Para corrigir acentuação dos caracteres no php, basta inserir no início do código:

```
<?php
ini_set('default_charset','UTF-8'); // no php.
//header('Content-Type: text/html; charset=utf-8'); // no html.
```

Para ligar e desligar os pinos físicos do Rpi via PHP com o commando shell_exec, crie um arquivo .php dentro de /var/www/html (*cd /var/www/html*) e depois digite *nano pinon.php* e com seguinte código para ligar o pino 40:

```
<?php
  shell_exec("gpio -1 mode 40 out"); // seta o pino fsico 40 (gpio -1 )
      como sada (out)
  shell_exec("gpio -1 write 40 1"); // Liga o pino fsico 40

  echo "Physical pin 40 is on!";
```

```
?>
```

Para executar o arquivo, acesse o endereço no browser http://IPdoRPi/pinon.php , exemplo: http://192.168.25.8/pinon.php e o pino físico 40 será ligado. Da mesma forma, para desligar, crie um arquivo *nano pinoff.php* escreva o conteúdo abaixo e realize o mesmo acesso pelo browser http://IPdoRPi/pinoff.php

```
<?php

  shell_exec("gpio -1 mode 40 out"); // seta o pino fsico 40 (gpio -1 )
      como sada (out)
  shell_exec("gpio -1 write 40 1"); // Liga o pino fsico 40

  echo "Physical pin 40 is on!";
?>
```

Para executar o arquivo, acesse o endereço no browser http://IPdoRPi/pinon.php , exemplo: http://192.168.25.8/pinon.php e o pino físico 40 será ligado. Da mesma forma, para desligar, crie um arquivo nano pinoff.php escreva o conteúdo abaixo e realize o mesmo acesso pelo browser http://IPdoRPi/pinoff.php

```
<?php

  shell_exec("gpio -1 write 40 0"); // Desliga o pino fsico 40

  echo "Physical pin 40 is off!";
?>
```

Para conferir a alteração do estado do pino basta digitar no terminal *gpio readall.*

18.4 CURL no PHP

Para testar o resgate de conteúdo de um arquivo utilizando PHP basta utilizar a biblioteca cURL.Para verificar se ela já não está habilitada no seu servidor teste com uo comando phpinfo(); em um arquivo .php.
Se aparecer cURL support, a bilbioteca está instalada. Se não, procure por ;extension=php_curl.dll no arquivo php.ini do servidor e retire o comentário (;). Abaixo um exemplo de código testado em php que resgata o conteúdo de um arquivo .txt.

```
<?php
   //código em: http://sanusb.org/exemplos/CurlPhp.php
  // Inicia o cURL acessando uma URL
  $cURL = curl_init('sanusb.org/test1/view.txt'); //sanusb.org/test1/
      registro.txt
```

```
  // Define a opão que diz que voc quer receber o resultado encontrado
  curl_setopt($cURL, CURLOPT_RETURNTRANSFER, true);
  // Executa a consulta, conectando-se ao site e salvando o resultado na
       variável $resultado
  $resultado = curl_exec($cURL);
  //echo "$resultado";
  echo $resultado;
    // Encerra a conexão com o site
  curl_close($cURL);
?>
```

18.5 Copiar arquivos do PC para o Raspberry Pi com Filezilla e vice-versa

Uma das melhores ferramentas é utilizar o SSH *File Transfer Protocol* ou SFTP. Nesse caso, SSH significa *Secure Shell*). Para isso, se utiliza o Filezilla (Figura 18.1) com configuração mostrada na figura abaixo a apartir do IP do Rpi na rede. Nesse caso, o Host é o IP, o Nome do usuário é pi, a senha é raspberry e a Porta SFTP é 22.

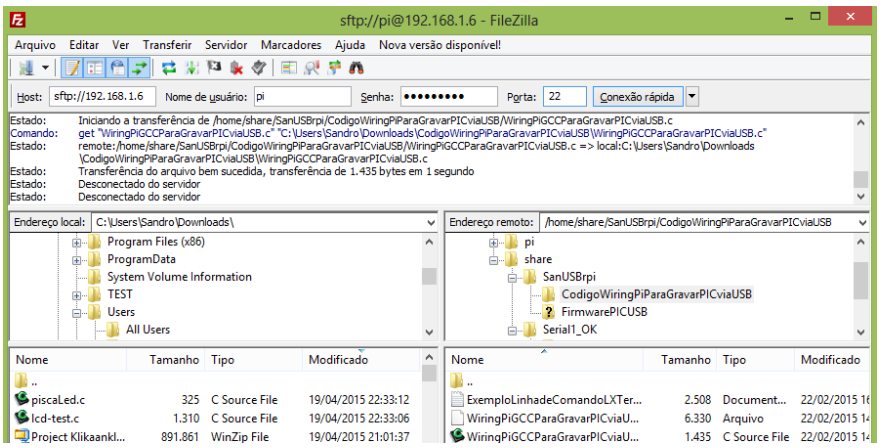


Figura 18.1: Filezilla.

Há uma série de programas que você pode baixar para ajudá-lo com isso, mas é recomendado o FileZilla.

Programando em shell script

http://www.devin.com.br/shell_script/

http://www.devmedia.com.br/introducao-ao-shell-script-no-linux/25778

Quem usa Linux conhece bem o prompt de comando sh, ou variações como o bash. O sh ou o bash têm uma "poderosa" linguagem de script embutido nelas mesmas – shell script. Diversas pessoas utilizam-se desta linguagem para facilitar a realização de inúmeras tarefas administrativas

no Linux, ou até mesmo criar seus próprios programas. Patrick Volkerding, criador da distribuição Slackware, utiliza esta linguagem para toda a instalação e configuração de sua distribuição. Você poderá criar scripts para automatizar as tarefas diárias de um servidor, para efetuar backup automático regularmente, procurar textos, criar formatações, e muito mais. Para você ver como esta linguagem pode ser útil, vamos ver alguns passos introdutórios sobre ela.

Interpretadores de comandos são programas feitos para intermediar o usuário e seu sistema. Através destes interpretadores, o usuário manda um comando, e o interpretador o executa no sistema. Eles são a "Shell" do sistema Linux. Usaremos o interpretador de comandos bash, por ser mais "extenso" que o sh, e para que haja uma melhor compreensão das informações obtidas aqui, é bom ter uma base sobre o conceito de lógica de programação.

Uma das vantagens destes shell scripts é que eles não precisam ser compilados, ou seja, basta apenas criar um arquivo texto qualquer, e inserir comandos à ele. Para dar à este arquivo a definição de "shell script", teremos que incluir uma linha no começo do arquivo (#!/bin/bash) e torná-lo "executável", utilizando o comando chmod. Vamos seguir com um pequeno exemplo de um shell script que mostre na tela: "Nossa! Estou vivo!":

#!/bin/bash
echo 'Nossa! Estou vivo!'

Fácil, hein? A primeira linha indica que todas as outras linhas abaixo deverão ser executadas pelo bash (que se localiza em /bin/bash), e a segunda linha imprimirá na tela a frase "Nossa! Estou vivo!", utilizando o comando echo, que serve justamente para isto. Como você pôde ver, todos os comandos que você digita diretamente na linha de comando, você poderá incluir no seu shell script, criando uma série de comandos, e é essa combinação de comandos que forma o chamado shell script. Tente também dar o comando 'file arquivo' e veja que a definição dele é de Bourne-Again Shell Script (bash script).

Contudo, para o arquivo poder se executável, você tem de atribuir o comando de executável para ele. E como citamos anteriormente, o comando chmod se encarrega disto:

$ chmod +x arquivo

Pronto, o arquivo poderá ser executado com um simples "./arquivo".
Veja mais aqui.

18.6 Dropbox no Raspberry

Dropbox Uploader é um script que pode ser usado para carregar, descarregar a excluir arquivos do Dropbox e para compartilhamento online de arquivos, sincronização e serviço de *backup*. Dessa forma, este aplicativo pode funcionar como um compartilhador de arquivos no modo internet em conjunto com o Samba que compartilha no modo intranet.
O Dropbox Uploader é também:

- Portátil: Está escrito em script BASH e Uploader necessita apenas que o usuário tenha

instalado a biblioteca cURL (sudo apt-get install curl).

- Seguro: Não é obrigado fornecer o nome de usuário/senha para esse script, porque ele usa o sistema de API do Dropbox para o processo de autenticação.

Características:

- Multiplataforma:
- Suporte para a API oficial Dropbox
- Não é necessário senha
- Simples passo-a-passo assistente de configuração
- Exclui, move, renomeia, copiar e lista arquivos
- Cria link de compartilhamento

18.6.1 Comandos

- **upload** – Para fazer upload de ficheiros locais para o serviço
- **download** – Download de ficheiros do Dropbox para directorio local
- **delete** – Apagar um ficheiro remoto no Dropbox
- **mkdir** – Criar diretórios dentro da pasta do App do Dropbox
- **list** – Listar conteúdo de um diretório no Dropbox
- **share** – Partilhar conteúdos
- **info** – Informações sobre a conta do Dropbox
- **unlink** Desconecta da API/pasta atual do dropbox

Instalando:

root@raspberrypi: # **git clone https://github.com/andreafabrizi/Dropbox-Uploader.git**

Cloning into 'Dropbox-Uploader'...
remote: Counting objects: 654, done.
remote: Total 654 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (654/654), 201.07 KiB | 277 KiB/s, done.
Resolving deltas: 100% (328/328), done.

root@raspberrypi: # ls
root@raspberrypi: /Dropbox-Uploader# **chmod +x dropbox_uploader.sh**
root@raspberrypi: /Dropbox-Uploader# **./dropbox_uploader.sh**

Esta é a primeira vez que você executar este script.

1. Abra a seguinte URL no seu navegador e fazer login usando sua conta: https://www.dropbox.com/developers/apps
2. Clique em "Criar App", em seguida, selecione "app API Dropbox"
3. Selecione "app API Dropbox"
4. Agora, vá em frente com a configuração, escolha as permissões de aplicativos e restrições de acesso a sua pasta DropBox.

Pode seu app ser limitada a sua própria pasta?

Sim. Meu aplicativo só precisa de acesso a arquivos que ele cria. Este é caracterizado por App Folder em https://www.dropbox.com/developers/apps.

Que tipo de arquivos que seu aplicativo precisa de acesso?

Tipos de arquivos específicos. Meu aplicativo só precisa de acesso a determinados tipos de arquivo, como texto ou fotos.

Todos os tipos de arquivos. Acesso completo ao Dropbox.

5. Digite o "App Name"que você prefere (por exemplo Rpi3SanUSB)
   Agora, clique no botão "Criar App".
   É gerada automaticamente uma pasta em C:/Users/Sandro/Dropbox/Aplicativos/Rpi3SanUSB

Clique em após executar no terminal **./dropbox_uploader.sh** :
Insira o acces token gerado no App do dropbox: #Access token: WXWS_1-mdZsAAAAAAAAXLMSb9D3-DhJHK7axeuycAeILP29fnUdKhr_XtLG1ABKm > **Access Token request... OK**
Para vincular uma nova pasta criada no app basta digitar **./dropbox_uploader.sh unlink**

Exemplos:
Upload com criação de uma nova pasta no dropbox:
//endereço absoluto do script dropbox_uploader.sh + upload + endereço absoluto do arquivo + nome da pasta e arquivo para o dropbox dentro da pasta do App

root@rpiusb:/home/pi/Dropbox-Uploader ./dropbox_uploader.sh upload /home/pi/SanUSBlink.sh SanUSBlink.sh
> Uploading "/home/pi/SanUSBlink.sh"to "/SanUSBlink1.sh"... DONE

Se a escolha de configuração inicial foi: Sim. Meu aplicativo só precisa de acesso a arquivos que ele cria.O arquivo será alocado em aplicativos: **C:/Users/Dropbox/Aplicativos/.**

Se a escolha de configuração inicial for: **Não. Meu aplicativo precisa de acesso a arquivos que já participam do Dropbox.** A pasta/arquivo gerada será alocada na pasta padrão do dropbox **C:/Dropbox/**. Foi o caso do exemplo em questão. Este é caracterizado por Full Dropbox em https://www.dropbox.com/developers/apps.

Download de um arquivo:
//endereço absoluto do script dropbox_uploader.sh + download + endereço absoluto da pasta/arquivo do dropbox a partir do App + nome da pasta e arquivo no Rpi

root@rpiusb:/home/pi/Dropbox-Uploader ./dropbox_uploader.sh download SanUSBlink.sh /home/share/link.sh
> Downloading "/SanUSBlink.sh"to "/home/share/link.sh"... DONE

list – listar conteúdo de um diretório no Dropbox
//endereço do script executável dropbox_uploader.sh + list + endereço absoluto da pasta do dropbox a partir do App

root@raspberrypi:/var/www# ./dropbox_uploader.sh list /firmware > Listing "/firmware"... DONE [F] 1322 exemplo.sh

mkdir – cria nova subpasta pasta na pasta do App
//endereço do script executável dropbox_uploader.sh + list + endereço absoluto da pasta do dropbox a partir do App

root@rpiusb:/home/pi/Dropbox-Uploader# ./dropbox_uploader.sh mkdir exemplopasta > Creating Directory "/exemplopasta"... DONE root@rpiusb:/home/pi/Dropbox-Uploader

Para linkar a API gerada com outros usuários, basta instalar e executar o ./dropbox_uploader.sh em outros RPIs e inserir a App key: kh18rq3keosqd95 e App secret: z9igzvhf35vgp47 , como por exemplo da API SyncUSB19, e depois confimar abrindo pelo browser a URL da API sugerida pela instação como https://www.dropbox.com/developers/apps/info/kh18rq3keosqd95

18.7 Banco de dados estruturado MySQL

MySQL é um banco de dados relacional, e um servidor multiusuário, multitarefa, compatível com o padrão SQL (Structured Query language – Linguagem de Consulta estruturada), linguagem essa amplamente utilizada para manipulação de dados em RDBMS (Banco de dados Relacionais), sendo considerada um ferramenta de manipulação de base de dados de tamanho moderado.

As principais características que destacam MySQL são: sua velocidade proporcionada pela sua implementação leve que não inclui na totalidade o suporte as instruções SQL; sua natureza de distribuição gratuita; facilidade de integração com servidor Web e linguagens de programação de desenvolvimento de sites dinâmicos, especialmente a linguagem PHP.

Principais características:

- Permite operações e funções nas cláusulas select e where, bem como suporte as funções SQL(group by , order by), além de funções de grupo como: Count(), avg(), sum(), std(), max(), min().
- Permite a seleção de diferentes tabelas de diferentes bases de dados em uma mesma query.
- Suas características de privilégio de password são bastante flexíveis, permitindo inclusive a validação por “host”.
- Possui algoritmos de criptografia de password, fornecendo assim segurança aos dados gravados nas tabelas.
- Permite a utilização de até 16 índices por tabela.
- Capacidade para manipular bancos com até 50 milhões de registros.
- MySQL foi escrito em C e C++.
- Permite conexões via TCP/IP.

O banco de dados MySQL suporta a utilização de procedures e triggers. A utilização de procedures possibilita que vários processos de programação sejam efetuados no próprio banco de dados, permitindo assim a diminuição do tráfego cliente-servidor bem como benefícios no que diz respeito a independência de aplicativos, pois os processos implementados no banco ficam transparentes para a aplicação que acessa os dados. Pode-se definir trigger como sendo um ou vários procedimentos armazenados que são executados quando determinado evento ocorre, por exemplo, a cada vez que um registro é atualizado dispara-se a execução de certos procedimentos. Trecho de código abaixo com trigger para guardar nova senha inserida:

```
mysql> CREATE TRIGGER user_au AFTER UPDATE
ON user FOR EACH ROW
BEGIN
   IF NEW.Password != OLD.Password THEN
   INSERT INTO mysql.password_history (Host, User) VALUES (NEW.Host, NEW.
      User) ON DUPLICATE KEY UPDATE LastChange=CURRENT_TIMESTAMP;
   END IF;
END;
```

Todas as informações referentes à controle de acessos e privilégios ficam armazenas em uma base de dados (**show databases**;) dentro do MySQL chamado mysql (**use mysql**;) , com as informações sobre usuários especificadas dentro de uma tabela (**show tables**;) com o nome de user. Esta tabela possui a seguinte estrutura de dados assim explicada:

Campo	Tipo	Funçao
Host	Char(60)	Host a qual se dará o acesso ao banco
User	Char(16)	Nome do usuário
Password	Char(16)	Senha do usuário (Campo pode ser criptografado)
Select_priv	Enum('n','y ')	Possibilidade de executar o comando select
Insert_priv	Enum('n','y ')	Possibilidade de executar o comando insert
Update_priv	Enum('n','y ')	Possibilidade de executar o comando update
Delete_priv	Enum('n','y ')	Possibilidade de executar o comando delet
Create_priv	Enum('n','y ')	Possibilidade criar banco de dados
Drop_priv	Enum('n','y ')	Possibilidade de apagar banco de dados
Reload_priv	Enum('n','y ')	Possibilidade de atualizar os privilégios definidos
Shutdown_priv	Enum('n','y ')	Possibilidade de finalizar o servidor
Process_priv	Enum('n','y ')	Possibilidade de gerenciar tarefas em execução
File_priv	Enum('n','y ')	Possibilidade de usar comandos que executam sobre arquivos
Grant_priv	Enum('n','y ')	Possibilidade de definir privilégios para outros usuários
Index_priv	Enum('n','y ')	Possibilidade de criar e apagar índices
Alter_priv	Enum('n','y ')	Possibilidade de executar o comando ALTER TABLE

Figura 18.2: Tabela user do MySQL

Outras tabelas completam o controle total de permissões e privilégios:

DB – a tabela na Figura 18.2 possui as informações de qual usuário pode executar quais comandos, de determinada máquina, exatamente sobre um banco de dados específico. Esta tabela se faz necessária, pois a tabela user não possui qualquer referência sobre quais bancos de dados eram definidos os privilégios, assim com a tabela db pode ter um controle mais específico.

Host – esta tabela tem por objetivo controlar o acesso a banco levando em consideração em

qual máquina esta sendo efetuada a conexão, assim se a tabela db não identificar o host para um banco de dados e usuário, a tabela host define se o banco de dados em questão pode ser acessado de determinado host.

18.7.1 Conexão ao MySQL de um servidor online

O processo de conexão com o banco requer que alguns parâmetros sejam passados ao servidor MySQL:

- Hostname – Qual computador deseja efetuar a conexão.
- Usuário – O nome do usuário que deseja se conectar ao banco
- A Senha – Senha do usuário que requisitou a conexão.

Esses dados são passados para o servidor através da aplicação cliente, por exemplo um página desenvolvida em PHP ou um módulo de administração.

Abaixo o exemplo de uma conexão através da linguagem PHP embutida em uma página HTML:

a) $conexao = mysql_conect (‘localhost’,’ sanusbor _1’,’xxxyyy’);

b) mysql_select_db(‘conexao’,’ sanusbor _ms’);

Com a função nº 1 nativa da linguagem PHP, solicita-se uma conexão ao servidor MySQL, sendo o lugar da conexão o próprio servidor(localhost), o usuário (a6132965_1) e a sua senha(xxxyyy). Caso executado com sucesso a função, a variável $conexao recebe a validação da conexão. A segunda função seleciona qual o banco de dados que a conexão acima realizada deseja conectar, nesse caso foi conectado ao banco de dados a6132965_ms. Exemplo:

```
<?php
//*
$hostname = "localhost";
$usuario = "sanusbor_1";
$senha = "laese";
$banco  = "sanusbor_ms";
//http://www.kinghost.com.br/wiki/doku.php/bancos_de_dados
//*/

/*
$hostname = "mysql4.000webhost.com";
$usuario = "a6132965_1";
$senha = "silveir1";
$banco = "a6132965_ms";
//*/

$conn = mysql_connect($hostname,$usuario,$senha);
```

```
mysql_select_db($banco) or die("Não foi possivel conectar ao banco MySQL
    ");

if (!$conn) {echo "Não foi possvel conectar ao Host."; exit;}
else {echo "Parabens!!! A conexao ao banco de dados ocorreu normalmente
    !";}

mysql_close();
?>
```

O vídeo exemplo em https://www.youtube.com/watch?v=pwuXOmpGJoU demonstra um acesso remoto a um led de um raspberry através de uma página php que é atualizada a cada 10segundos para conectar a um banco na nuvem e verificar uma flag desse banco de dados de um servidor online. Essa flag pode ser alterada por qualquer software na internet de acesso ao banco, como o PhpMyAdmin. O símbolo % na configuração do MySQL remoto permite que qualquer IP de cliente, como o Rpi, possa acessar o banco criado no servidor online. Script PHP a seguir:

```
<?php
error_reporting(0);
?>
<html>
    <head>
  <meta http-equiv="refresh" content="10;URL=http://192.168.1.4/ipmysql/
      index.php">
    </head>
    <body>
    <?php

    $conecta = mysql_connect("sanusb.org", "sanusbor_1", "laese") or
        print (mysql_error());
    mysql_select_db("sanusbor_ms", $conecta) or print(mysql_error());
    $sql = "SELECT comando1, comando2 FROM Rasp where id='1'";
    $result = mysql_query($sql, $conecta);
    $ler = mysql_fetch_array($result);
    $sensor = $ler["comando1"];
    echo `gpio mode 0 out`;
    echo `gpio mode 1 out`;

    if($sensor==1){
      echo "Ativado, valor $sensor";
      echo `gpio write 0 1`;
      echo `gpio write 1 1`;
```

```
  }else if ($sensor==0){
    //shell_exec("shutdown -a");
    echo "Desativado, valor $sensor";

    echo `gpio write 0 0`;
    echo `gpio write 1 0`;
  }
  mysql_close($conecta);
?>
  </body>
  </html>
```

Testando a Liberação do IP para acesso remoto ao MySQL remoto do servidor online: https://www.youtube.com/watch?v=sodbMGrxdcE. O banco MySQL server pode ser instalado no servidor online com MySQL client no Rpi, mais adequado para aplicação com ioT, ou MySQL server no Rpi, adequado para apliações locais. Para se se conectar ao MySQL remoto via shell, é necessário fornecer também o nome de usuário, senha e o nome do servidor em que o banco está hospedado:

mysql -u nome-do-usuario -p senha -h nome-do-servidor nome-do-banco-de-dados

Seguindo o exemplo acima, ficaria assim:

mysql -u sanusbor_1 -p laese -h sanusb.org sanusbor_ms

Instalando o MySQL Server no Raspbian Para a instalação é só seguir os passos a baixo: http://thobias.org/doc/shell_bd.html

1. No teminal do Raspbian digite como root os comandos a baixo:

apt-get update
apt-get install mysql-server
apt-get install php5-mysql

Obs.: Para desinstalar basta digitar:**sudo apt-get remove –purge mysql**
Remova todas as pastas (**rm –r pasta**) encontradas com **find / -name "mysql".**

O php5-mysql adiciona as bibliotecas mysql para permitir que o PHP possa acessar o banco de dados mysql. Durante a instalação será pedido para criar uma senha (ex. 1234).

18.7.2 Comando do MySQL

Para alterar o nível dos pinos no banco de dados é só utilizar os comandos:

1. Conecte com o banco de dados usando o comando
   mysql -p -u root ou
   mysql -u root –pSENHADOROOT

Importe o arquivo fonte para a base de dados:
 # source "local de origem"/"arquivo".sql
Exemplo: *source /var/www/gpio/gpio.sql*

2. Para mostrar todos os bancos de dados digite o comando a seguir conforme Figura 18.3
show databases;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| gpio               |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)
```

Figura 18.3: Database

Para modificar a base de dados (database), basta digitar: use db.
Exemplo:
use mysql; use gpio;
Para mostrar todas as tabelas do banco importado gpio.sql digite conforme Figura 18.4:
show tables;

```
mysql> show tables;
+----------------+
| Tables_in_gpio |
+----------------+
| pinDescription |
| pinDirection   |
| pinStatus      |
| users          |
+----------------+
4 rows in set (0.00 sec)
```

Figura 18.4: Tabelas do banco MySQL.

3. Para alterar quaisquer status dos pinos gpio do raspberry no banco de dados é só usar o comando:
UPDATE pinStatus SET pinStatus = '0 ou 1' WHERE pinNumber = 'nº bcm do pino

desejado';
UPDATE pinDirection SET pinDirection = 'out' WHERE pinNumber = '18'
UPDATE pinStatus SET pinStatus = '1' WHERE pinNumber = '18'

4. Para mostrar uma tabela de estatus dos pinos use:
   SELECT * FROM pinStatus

18.7.3 Atualizando o Raspberry Pi

Para para copiar a pasta completa para dentro de www devemos executar o comando:

cp –r "/endereço da pasta gpio" /var/www/

Lembrando que é necessário dar permissões de execução na pasta www e que é necessário alterar o usuário e senha do banco de dados nos arquivos .php e .sh para a senha criada no momento da instalação do mysql, pois os mesmos estão com valores padrão e vão precisar de conexão com o banco de dados instalado:

cd /var

chmod 777 www

Para que as alterações do banco de dados sejam realizadas no raspberry é necessario rodar um script GPIOServer.sh, para isso basta executá-lo:

chmod +x GPIOServer.sh

./ GPIOServer.sh

Esse script é responsavel por verificar o banco de dados passando as alterações realizadas no banco de dados para o RPI. O script pode continuar rodando enquanto são feitas alterações no banco de dados, usando os comandos do mysql. Para o exemplo em questão ilustrado em http://youtu.be/s67GjlMeITc . Se acessar o endereço **IPdoRpi/gpio/index.php** ou simplesmente **IPdoRpi/gpio** irá solicitar uma senha que foi cadastrada como admin (usuário) e gpio (senha).

18.8 Instalando o MySQL Client

Instalando o MySQL Client

1. Para instalar o MySQL client que é muito mais leve que o server, utilize o comando:
   apt-get install mysql-client
2. Para prover suporte às bibliotecas em C no MySQL instale:
   sudo apt-get install libmysqlclient-dev
3. Para compilar, insira include <my_global.h> e include <mysql.h> no cabeçaho e considere as bibliotecas adicionais.
   gcc -o exemploBD exemploBD.c -l wiringPi **$(mysql_config –libs –cflags)**

Abaixo um exemplo de conexão direta com um banco MySQL de um servidor na nuvem:

```
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <unistd.h>
#include <termios.h>
#include <fcntl.h>
#include <my_global.h> // gcc -o exemploBD exemploBD.c -l wiringPi $(
    mysql_config --libs --cflags)
#include <mysql.h> // apt-get install mysql-client libmysqlclient-dev
#include <stdio.h> // http://zetcode.com/db/mysqlc/

char charRecebido;
int fd;
int flagL=0;
int flagD=0;
int pinLed=1, numquant=0;
char x;
char value[10];
char guardarPic[10];
char *resultado; //string
char str1[10];
char buf1[100];
int num = 0, res,pin4,pin17,pin18,pin21,pin22,pin23,pin24,pin25;
     //strings para selecao de valor nico (uma linha uma coluna -> row
         [0])
     char *status4= "SELECT pinStatus FROM pinStatus WHERE pinNumber='4'
         ";
     char *status17= "SELECT pinStatus FROM pinStatus WHERE pinNumber
         ='17'";
     char *status18= "SELECT pinStatus FROM pinStatus WHERE pinNumber
         ='18'";
     char *status21= "SELECT pinStatus FROM pinStatus WHERE pinNumber
         ='21'";
     char *status22= "SELECT pinStatus FROM pinStatus WHERE pinNumber
         ='22'";
     char *status23= "SELECT pinStatus FROM pinStatus WHERE pinNumber
```

```
        ='23'";
     char *status24= "SELECT pinStatus FROM pinStatus WHERE pinNumber
        ='24'";
     char *status25= "SELECT pinStatus FROM pinStatus WHERE pinNumber
        ='25'";

//---------------------------------------------------------------
// Function for processing error from MySQL operation
//---------------------------------------------------------------
void finish_with_error(MYSQL *con)
{
  fprintf(stderr, "%s\n", mysql_error(con));
  mysql_close(con);
  exit(1);
}

int main(void){

char *server = "sanusb.org"; /* set me first */
char *user = "sanusbor_1";
char *password = "laese";
char *database = "sanusbor_ms";

wiringPiSetup(); //Para configurar os pinos na sequncia GPIO (BCM)
                              //BCM
     system("gpio mode 7 out"); //pin 4
     system("gpio mode 0 out"); //pin 17
     system("gpio mode 1 out"); //pin 18 //Led aqui
     system("gpio mode 29 out");//pin 21
     system("gpio mode 3 out"); //pin 22
     system("gpio mode 4 out"); //pin 23
     system("gpio mode 5 out"); //pin 24
     system("gpio mode 6 out"); //pin 25
//---------------------------------------------------------------
// Creating dataTable
/*---------------------------------------------------------------

if (mysql_query(con, "CREATE TABLE IF NOT EXISTS sensordata(temperature
    INT,time DATETIME)"))
  {
       finish_with_error(con); //only one remote database: sanusbor_ms
```

```
  }//*/
//------------------------------------------------------------------
// Inserting a record into dataTable
//------------------------------------------------------------------
/*
if (mysql_query(con, "INSERT INTO temp VALUES(75, 19)"))
   {
      finish_with_error(con);
   }
//*/
//------------------------------------------------------------------
// Inserting a data into dataTable with sprintf
/*------------------------------------------------------------------
         ++num;
        sprintf(buf1, "INSERT INTO sensordata VALUES(%d,NOW())", num);
        if (mysql_query(con, buf1)) {mysql_close(con);}
        //*/
    //------------------------------------------------------------------
// Deleting dataTable
/*------------------------------------------------------------------
if (mysql_query(con, "DROP TABLE IF EXISTS sensordata"))
   {
      finish_with_error(con);
   }//*/

  MYSQL *con = mysql_init(NULL);
  MYSQL_RES *result;
  MYSQL_ROW row;
  int i;

   // Get the number of fields of dataTable
  int num_fields = mysql_num_fields(result);

  /* Iterate through all rows selected from dataTable and print them
  while ((row = mysql_fetch_row(result)))
  {
      for (i = 0; i < num_fields; i++) //Iterate through all fields of
          each record
      {
          printf("%s ", row[i] ? row[i] : "NULL");
      }
```

```
      printf("\n");

  }

  mysql_free_result(result); //*/ //serve para limpar a variável de
     resultado retornado por mysql_store_result
  //mysql_close(con);
  //-------------------------------------------------------------
// Connecting to Database
//*-------------------------------------------------------------
  if (con == NULL)
   {
     fprintf(stderr, "%s\n", mysql_error(con));
     exit(1);
   }

if (mysql_real_connect(con, server, user, password, database, 0, NULL,
    0) == NULL)
   {
     finish_with_error(con);
   }

 while(1) {

//-------------------------------------------------------------
// Function for selecting records from dataTable
//*-------------------------------------------------------------
  if (mysql_query(con, status18)) {
                              finish_with_error(con);
                               }
         delay (1000);
        //selected data from dataTable to the variable row[0]
         result = mysql_store_result(con);
         delay (1000);
         if(result) {
         row = mysql_fetch_row(result);
         resultado = row[0]; //Como a consulta foi filtrada em apenas
            uma coluna e uma linha row[0]
         delay (1000);
         pin18 = atoi(resultado); //string para inteiro
```

```
        if(pin18==1){
        printf("BCM Pin18 = %d\r\n", pin18);
        system("gpio write 1 1"); //Wpi 1

        }else if (pin18==0){
        printf("BCM Pin18 = %d\r\n", pin18);
        system("gpio write 1 0"); //Wpi 1
                        }
                }

  delay (3000);
   //mysql_close(con);
        }
   return 0;
}
```

18.9 Banco de dados estruturado SQlite

SQLite é uma boa opção de banco de dados para sistemas embarcados Linux por uma série de motivos:

- É fácil de instalar.
- É livre e *open source*.
- Os bancos de dados são armazenados em um único arquivo que é possível se conectar diretamente a partir de seu código, sem a necessidade de um processo de servidor em execução.
- A bibliotecas são customizadas, ou seja, ocupa menos espaço no disco.
- E ainda um software que está sendo usado em muitos produtos bem conhecidos, tais como o Mozilla Firefox, Apple, Dropbox e Google.

O uso do SQLite é recomendado onde a simplicidade da administração, implementação e manutenção são mais importantes que incontáveis recursos que SGBDs mais voltados para aplicações complexas possivelmente implementam. As situações onde a simplicidade é a melhor escolha são muito mais freqüentes do que pode-se imaginar.

Exemplos de uso do SQLite são:

- Sites com menos de cem mil requisições por dia
- Dispositivos e sistemas embarcados
- Aplicações desktop
- Ferramentas estatísticas e de análise
- Aprendizado de banco de dados
- Implementação de novas extensões de SQL

Não se recomenda o uso do SQLite para sites com:

- Muitos acessos
- Grande quantidades de dados (talvez maior que algumas dúzias de gigabytes)
- Sistemas com grande concorrência
- Aplicações cliente/servidor

SQLite:

- É Software Livre/domínio público e multiplataforma
- É um mecanismo de armazenamento seguro com transações ACID
- Não necessita de instalação, configuração ou administração.
- Implementa a maioria do SQL92
- Permite guardar o banco de dados em um único arquivo
- Suporta bases de dados abaixo de 2 terabytes
- Não tem dependências externas

18.9.1 Instalação

Para instalar, basta digitar:

sudo apt-get install sqlite3
sudo apt-get install libsqlite3-dev

Criando seu primeiro banco de dados

O SQLite trata as bases de dados num único arquivo, ou seja, o nome do banco de dados corresponderá ao nome do arquivo que o SQLite criará nesta fase. Para este exemplo iremos chamá-lo MyFirstDatabase.db e nós vamos guardá-lo na pasta de início padrão. Então, ainda no mesmo tipo de linha de comando:

sqlite3 MyFirstDatabase.db

Ao digitar aparece o prompt do SQLite, como abaixo na Figura 18.5:

```
root@RPIUSB:/home/pi# sqlite3 MyFirstDatabase.db
SQLite version 3.7.13 2012-06-11 02:05:22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
```

Figura 18.5: Prompt do SQLite

Agora é possível adicionar dados ao banco.

18.9.2 Criando sua primeira Tabela

No SQLite, os dados são armazenados em tabelas que, conceitualmente, parece muito com tabelas de dados que você iria encontrar no mundo real (ver Figura 18.6).

Fruta	Calorias	Preco
Framboesa	60	4.99
Banana	52	0.79
Laranja	85	2.5

Figura 18.6: Tabela exemplo banco de dados.

Primeiro, é necessário criar a tabela e, em seguida, adicionar os dados à essa tabela. Para fazer ambas as tarefas podemos usar uma linguagem chamada SQL, que é abreviação de "Structured Query Language".Embora SQL não seja estritamente uma linguagem de programação, mas adaptado especificamente para consultar dados,o SQLite é um programa que permite executar comandos tradicionais SQL, como é o caso Oracle, MySQL, entre outros. Cada um deles tem particularidades, mas os comandos básicos são os mesmos em todos os programas de banco de dados estruturados. Para criar a tabela, digite ao prompt **sqlite>**, como mostrado abaixo, e pressione Enter:

CREATE TABLE fruta (nome TEXT, calorias INT, preco NUMERIC);

Para facilitar, todos os comandos em cor azul são comandos SQL e devem ser inseridos no prompt **sqlite>**. Todos os comandos em cor preta devem ser introduzidos no prompt de comando Linux padrão $.

É necessário o ponto e vírgula no final do comando SQL. Se não for inserido basta colocá-lo na linha seguinte e pressionar Enter. É isso aí, você acabou de criar uma tabela chamada de fruta.

Bem, fruta é o nome da tabela que criamos e que iremos utilizar para se referir à tabela quando se quiser inserir e ler os dados, nome, calorias e preço são os nomes das três colunas; TEXTO, INT e NUMERIC são os tipos de dados para as colunas correspondentes.

Para qualquer um que tenha usado quaisquer outros bancos de dados SQL, no passado, é importante notar aqui que SQLite usa um sistema de tipo de dados dinâmicos. Isto significa que você pode realmente armazenar qualquer tipo de dados em qualquer coluna. Para sair SQLite do banco criado **MyFirstDatabase.db**, digite:

.exit

e depois listar os arquivos(Figura 18.7):

ls

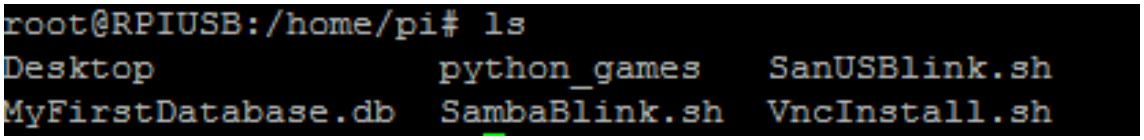


Figura 18.7: Database

e é possível ver nosso banco de dados recém-criado.

Inserção de dados

Bem, agora temos uma tabela. O próximo passo é inserir alguns dados usando SQL.
É necessário reconectar-se ao banco de dados com o mesmo comando como antes.

sqlite3 MyFirstDatabase.db

Em seguida, para inserir a primeira linha de dados você pode ver na tabela acima, nós digitamos:

INSERT INTO fruta values("Framboesa", 60, 4.99);

Repita para as próximas duas linhas:

INSERT INTO fruta values("Banana", 52, 0.79);
INSERT INTO fruta values ("Laranja", 85, 2.5);

Consultando os Dados

Para ver todos os dados (*) que está numa tabela em particular, usamos o comando de seleção SQL.

SELECT * FROM fruta; É mostrado as três linhas de dados que acabamos de adicionar(Figura 18.8):

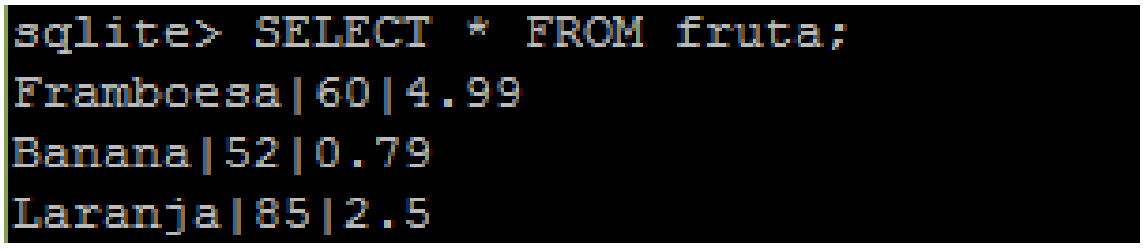


Figura 18.8: Linhas de dados

Assim é visto todas as colunas(Figura 18.9). Para ver um subconjunto de colunas, como nome e preço, basta digitar (antes de FROM) :

SELECT nome, preco FROM fruta;

```
root@RPIUSB:/home/pi# sqlite3 MyFirstDatabase.db
SQLite version 3.7.13 2012-06-11 02:05:22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> SELECT nome, preco FROM fruta;
Framboesa|4.99
Banana|0.79
Laranja|2.5
```

Figura 18.9: Colunas de dados

Para filtrar um subconjunto específico de linhas precisamos da cláusula **WHERE** (Figura 18.10). Por exemplo, o comando:

SELECT * FROM fruta WHERE preco > 3;

```
sqlite> SELECT * FROM fruta WHERE preco > 3;
Framboesa|60|4.99
sqlite>
```

Figura 18.10: Demonstração do comando WHERE

Mostra a linha em que o preço é maior do que 3. E o comando abaixo(Figura 18.11) mostra a linha com o nome Banana:

SELECT * FROM fruta WHERE name = "Banana";

```
sqlite> SELECT * FROM fruta WHERE nome = "Banana";
Banana|52|0.79
```

Figura 18.11: Demonstração do comando SELECT

O comando SELECT é realmente um dos principais comandos da linguagem SQL.

Excluindo Dados

Para remover dados da tabela, em seguida(Figura 18.12), é possível utilizar o comando DELETE. que é muito semelhante ao comando SELECT assim:

DELETE FROM fruta WHERE nome = "Framboesa";

```
sqlite> DELETE FROM fruta WHERE nome = "Framboesa";
sqlite> SELECT * FROM fruta;
Banana|52|0.79
Laranja|85|2.5
```

Figura 18.12: Remoção de dados da tabela

DICA: Se você quiser adicionar novamente os dados que você acabou de excluir, uma maneira rápida de fazer isso é para pressionar a seta para cima na linha de comando para voltar a inserir comandos que você inicialmente utilizou para adicionar os dados e apenas pressione Enter.

Usando phpLiteAdmin com Raspberry Pi

O phpLiteAdmin que é uma ferramenta *front-end* baseada em PHP para gerenciar e manipular bancos de dados com SQLite. Para utilizar o phpLiteAdmin no Rpi como servidor é necessário ter instalado anteriormente:

apt-get install apache2 php5 libapache2-mod-php5
apt-get install php5-sqlite

E seguir os passos abaixo:

```
cd /var
chmod 755 www
cd www
mkdir admin
cd admin
wget sanusb.org/arquivos/phpliteAdmin_v1-9-5.zip
unzip phpliteAdmin_v1-9-5.zip
```

Agora é necessário renomear e reconfigurar, se necessário, *password e directory*:

```
cp phpliteadmin.config.sample.php phpliteadmin.config.php
chmod 755 php*
nano phpliteadmin.config.php
...
$password = 'admin'; //YourPassword
$directory = '.'; // AbsolutePathToDatabase -> . criar db no diretório /
    var/www/admin
```

Agora, crie um banco de dados exemplo como o **MyFirstDatabase.db** do tópico anterior em **/var/www/admin.**

Para dar permissão de escrita no banco de dados gerado digite:

chown www-data /var/www/admin/
chown www-data /var/www/admin/ MyFirstDatabase.db

Pronto, para acessar e manipular dados com o Phpliteadmin basta abrir com um navegador o endereço IPdoRpi/admin/phpliteadmin.php, exemplo: 192.168.1.7/admin/phpliteadmin (Figura 18.13). Mais detalhes no vídeo: https://youtu.be/0adJoqDS9eE.

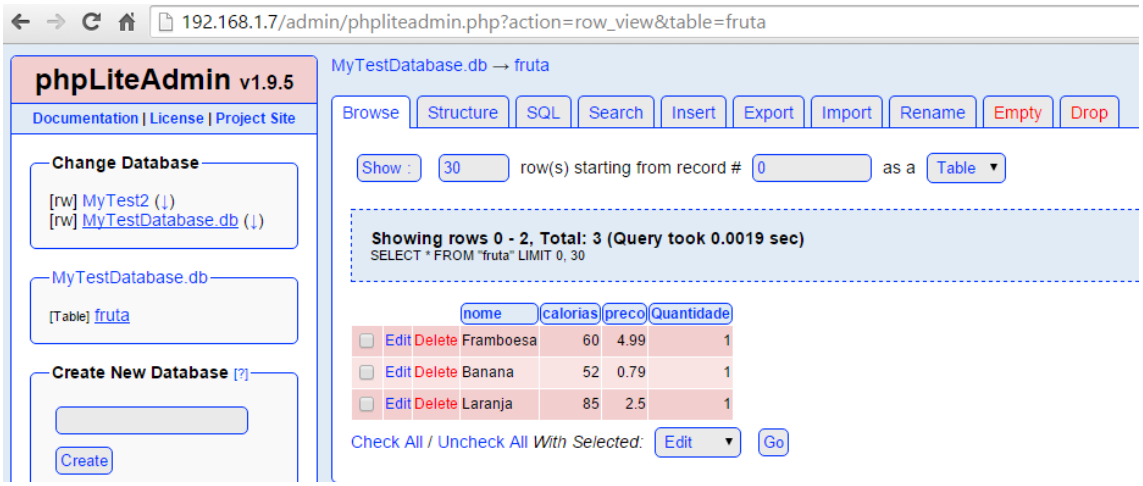


Figura 18.13: Tela do phpLiteAdmin

É possível notar no vídeo que a aba *Structure* insere e configura as colunas da tabela fruta, e a aba Insert é utilizada para inserir as linhas de dados da tabelas.

Veja um exemplo de criação de um novo banco de dados :

O SQLite trata as bases de dados num único arquivo, ou seja, o nome do banco de dados corresponderá ao nome do arquivo que o SQLite criará nesta fase. Para este exemplo será chamado Temperatura.db e vai ser guardado na pasta compartilha no samba.

Então, ainda no mesmo tipo de linha de comando:
- Sqlite3 Temperatura.db

A tabela será criada no banco de dados utilizando o phplite admin e ficará da seguinte forma (Figura 18.14):

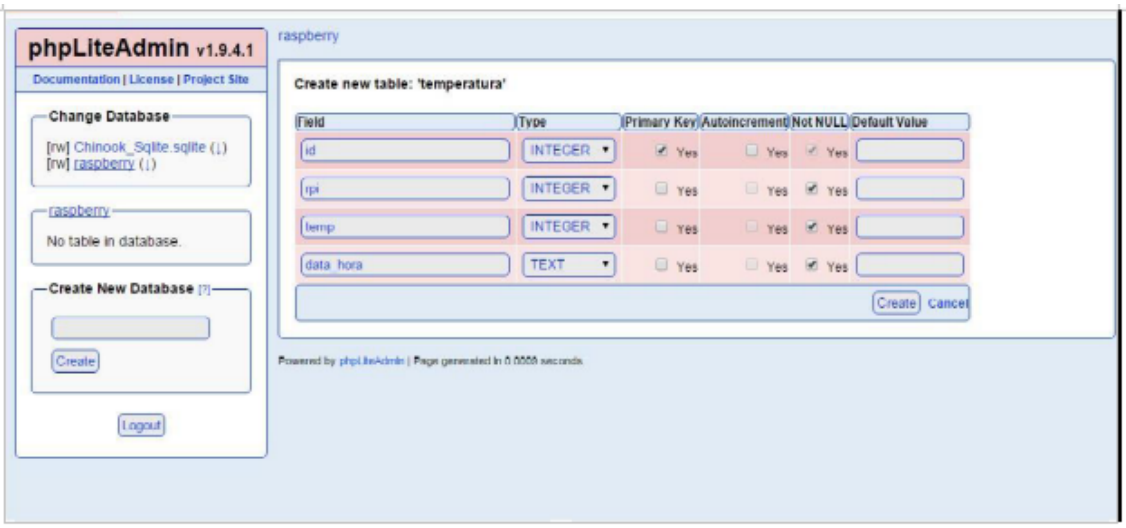


Figura 18.14: Tabela criada usando o banco de dados

Para finalizar iremos inserir dados na base e imprimir através do phpLite admin(Figura 18.15):

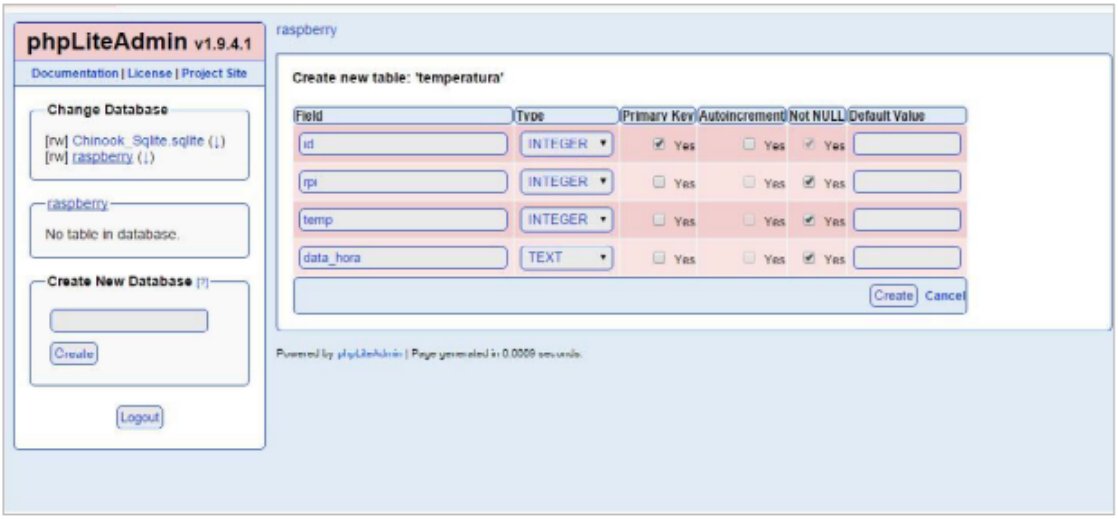


Figura 18.15: Inserção e impressão de dados

Como um segundo exemplo, é possível verificar um semáforo com cruzamento de pedestres em que o momento de solicitação de passagem do pedestre é guardado no banco SQLite. Vale salientar que para compilar um código em C utilizando as bibliotecas wiringPi e SQLite pode-se utilizar a seguinte linha de comando:

gcc -o nomearquivo nomearquivo.c -lsqlite3 -std=c99 -l wiringPi

Abaixo o código em C do referido exemplo:

```
CRIAR A BASE DE DADOS test.db DA TABELA semaforo:
sqlite3 test.db
```

```
*/
/*COMO COMPILAR COM WIRINGPI E SQLITE:
gcc -o nomearquivo nomearquivo.c -lsqlite3 -std=c99 -l wiringPi
*/
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#include <sqlite3.h>

#define LED 7
#define BOTAO 2
#define CVERMELHO 4
#define CVERDE 1
#define CAMARELO 5
#define PVERMELHO 0
#define PVERDE 3
#define SEG_A 21
#define SEG_B 22
#define SEG_C 23
#define SEG_D 24
#define SEG_E 25
#define SEG_F 26
#define SEG_G 27
#define DEZ 28
#define UNI 29
int flagBotao=0;

//Variáveis globais para conexão com o banco
sqlite3 *db;
char *err_msg = 0;

int digitos[200]={
    1,1,1,1,1,1,0,
    0,1,1,0,0,0,0,
    1,1,0,1,1,0,1,
    1,1,1,1,0,0,1,
    0,1,1,0,0,1,1,
    1,0,1,1,0,1,1,
    1,0,1,1,1,1,1,
```

```
    1,1,1,0,0,0,0,
    1,1,1,1,1,1,1,
    1,1,1,1,0,1,1
};

int segmentos[10]={SEG_A,SEG_B,SEG_C,SEG_D,SEG_E,SEG_F,SEG_G};

void display(){
    int i,j,num,unidade,dezena,dec;

  for(dec=99 ; dec >=0; dec --){

    for (j = 0; j < 15; j++){
      digitalWrite(DEZ, LOW); //liga display unidade
      digitalWrite(UNI, HIGH); //desliga display dezena

      dezena= dec/10;
      unidade = dec %10;

      for(i=0;i<7;i++){ //escreve o dgito da dezena no display
        digitalWrite(segmentos[i],digitos[7*dezena+i]);
      }

      delay(5);
      digitalWrite(DEZ, HIGH); //desliga o display da dezena
      digitalWrite(UNI, LOW); //desliga o display da unidade

      for(i=0;i<7;i++){ //escreve o dgito da unidade no display
        digitalWrite(segmentos[i],digitos[7*unidade+i]);
      }

      delay(5);
    }
  }
}

void semaforoPedestre(){
  flagBotao=0;
  // sinal vermelho do carro
  digitalWrite(CVERMELHO,HIGH);
  // sinal vermelho pedestre
```

```
  digitalWrite(PVERMELHO,LOW);
  // sinal verde pedestre
  digitalWrite(PVERDE,HIGH);
  display();
  digitalWrite(PVERDE,LOW);
  // sinal vermelho pedestre
  digitalWrite(PVERMELHO,HIGH);
  // sinal vermelho carro
  digitalWrite(CVERMELHO,LOW);
}

void semaforoCarro(){
  int rc = sqlite3_open("test.db", &db);
  digitalWrite(CVERDE,HIGH);
  delay(2000);
  digitalWrite(CVERDE,LOW);
  //sinal amarelo
  digitalWrite(CAMARELO,HIGH);
  delay(2000);
  digitalWrite(CAMARELO,LOW);
  //sinal vermelho
  digitalWrite(CVERMELHO,HIGH);
  if(flagBotao==0){
    delay(2000);
    digitalWrite(CVERMELHO,LOW);
  }else{
    //Insere no banco de dados a data e o horário que o botão foi
       pressionado
    char *sql = "insert into Semaforo(data) VALUES (datetime('now'));";
    rc = sqlite3_exec(db, sql, 0, 0, &err_msg);
    semaforoPedestre();
  }

}

PI_THREAD (botao_pedestre) {
    //(void)piHiPri (10) ;

    while(1){

      if(digitalRead(BOTAO) == 0){
```

```
      flagBotao=1;
    }
  }
}

PI_THREAD (semaforo) {
  (void)piHiPri (10) ;

  while(1){
    semaforoCarro();
  }
}

int main ()
{

   //Comeando a trabalhar com sqlite
   int rc = sqlite3_open("test.db", &db);

   if (rc != SQLITE_OK) {

       fprintf(stderr, "Cannot open database: %s\n", sqlite3_errmsg(db))
           ;
       sqlite3_close(db);

       return 1;
   }

   /*char *sql = "DROP TABLE IF EXISTS Semaforo;"
               "CREATE TABLE Semaforo(Id INT, Name TEXT);"
               "INSERT INTO Semaforo VALUES(1, '10:45');"
               ;*/

   char *sql = "create table if not exists Semaforo(id integer primary
       key autoincrement, data DATE);";
   rc = sqlite3_exec(db, sql, 0, 0, &err_msg);

  char *sql2 = "insert into Semaforo(data) VALUES (datetime('now'));";
   rc = sqlite3_exec(db, sql2, 0, 0, &err_msg);
```

```
    if (rc != SQLITE_OK ) {

        fprintf(stderr, "SQL error: %s\n", err_msg);

        sqlite3_free(err_msg);
        sqlite3_close(db);

        return 1;
    }

    sqlite3_close(db);
  wiringPiSetup();
  pinMode(LED, OUTPUT);
  pinMode(CVERMELHO, OUTPUT);
  pinMode(CVERDE, OUTPUT);
  pinMode(CAMARELO, OUTPUT);
  pinMode(PVERMELHO, OUTPUT);
  pinMode(PVERDE, OUTPUT);
  pinMode(SEG_A, OUTPUT);
  pinMode(SEG_B, OUTPUT);
  pinMode(SEG_C, OUTPUT);
  pinMode(SEG_D, OUTPUT);
  pinMode(SEG_E, OUTPUT);
  pinMode(SEG_F, OUTPUT);
  pinMode(SEG_G, OUTPUT);
  pinMode(DEZ, OUTPUT);
  pinMode(UNI, OUTPUT);

  pullUpDnControl(BOTAO, PUD_UP);
  piThreadCreate (botao_pedestre) ;
  piThreadCreate (semaforo) ;
  while(1){
  }
}
```

O print do resultado de uma consulta, após compilar o firmware acima e pressiona o botão de pedestre, é mostrado na Figura 18.16:

```
root@raspberrypi:/home/share# ./semsqlite
^C
root@raspberrypi:/home/share# sqlite3 test.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> SELECT * FROM semaforo;
1|2016-08-22 19:50:49
2|2016-08-22 19:58:19
3|2016-08-22 19:58:35
sqlite>
```

Figura 18.16: Resultado da compilação do código ao apertar o botão de pedestres

Um outro exemplo para plotar pontos armazenados no SQLite no google maps através de uma API da google demonstrado no vídeo: https://youtu.be/ZnYQ7rn5Wc0.

Arquivos em: https://drive.google.com/drive/u/1/folders/1DhfSYt_6p-1qDeq6CVxLhl-EFfsYsC8i . O arquivo data.php encapsula os dados do banco em json para que a página html possa plotar os pontos do banco com a função *.getJSON*.

Importante conferir permissão de gravação a todos os arquivos do SQlite (chmod 777).

Para armazenar bancos e atributos ou características, é necessário também utilizar um banco de dados, como o SQlite, e acessar esses dados através de um arquivo .php que, por sua vez, envia esses dados para o mapa .html via json. O resultado do exemplo pode ser visto em: sanusb.org/sig ou http://confirma.tk/map$_sql/$

O SQLite é uma biblioteca de código aberto (open source) recomendável em aplicativos básicos desktop/mobile e Web Sites que permite a disponibilização de um banco de dados estruturado na própria aplicação, sem a necessidade de acesso a um sistema de gerenciamento de banco de dados (SGDB) separado.

Dessa forma, é importante salientar que, geralmente, não é necessário instalar o SQlite no servidor, pois o PHP já possui bibliotecas nativas para suporte ao SQLite e que os bancos e tabelas criados pelo phpliteadmin.php, por exemplo pontos.bd, podem ficar na mesma pasta dos arquivos WEB.

O SQlite contém uma interface gráfica que é um arquivo .php chamado phpliteadmin.php que possibilita a inserção e configuração interativa de dados e tabelas no Banco sem utilizar a linguagem SQL.

sanusb.org/sig/phpliteadmin.php?action=row$_viewtable = pontosouhttp://confirma.tk/map_sql/phpliteadmin.ph$
$row_viewtable = pontosSenha : admin$

Veja também: https://www.youtube.com/watch?v=0adJoqDS9eE (PhpLiteAdmin) https://www.youtube.com/watch?

(Criar uma Google maps API)

18.10 Modificar o nome do Rpi

Para alterar o nome basta inserir as três linhas de comando abaixo no terminal:

nano /etc/hostname (mudar o hostname para o novo nome)
nano /etc/hosts (mudar o hostname para o novo nome)
/etc/init.d/hostname.sh (executar o script para mudança do nome)

reboot (reiniciar o sistema)

18.11 Modificar a senha do Usuário pi

pi@raspberrypi: $ passwd
(current) UNIX password: raspberry
Enter new UNIX password: NOVA SENHA
Retype new UNIX password: NOVA SENHA

18.12 Modificar data, hora e fuso horário do Rpi

É possível, através do "raspi-config", alterar a data, hora e fuso horário do Rpi. Basta digitar este comando com o usuário root (sudo su). Siga o passo a passo mostrado nas figuras abaixo:

1. Selecione a opção 5 (Figura 18.17):

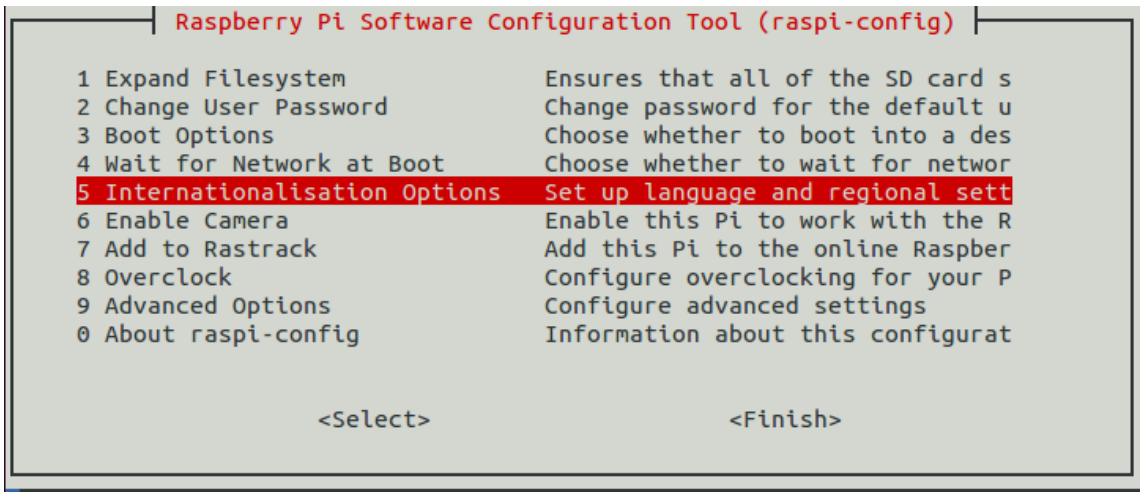


Figura 18.17: Tela do RaspiConfig

2. Selecione a opção I2 para configurar o fuso horário(Figura 18.18):

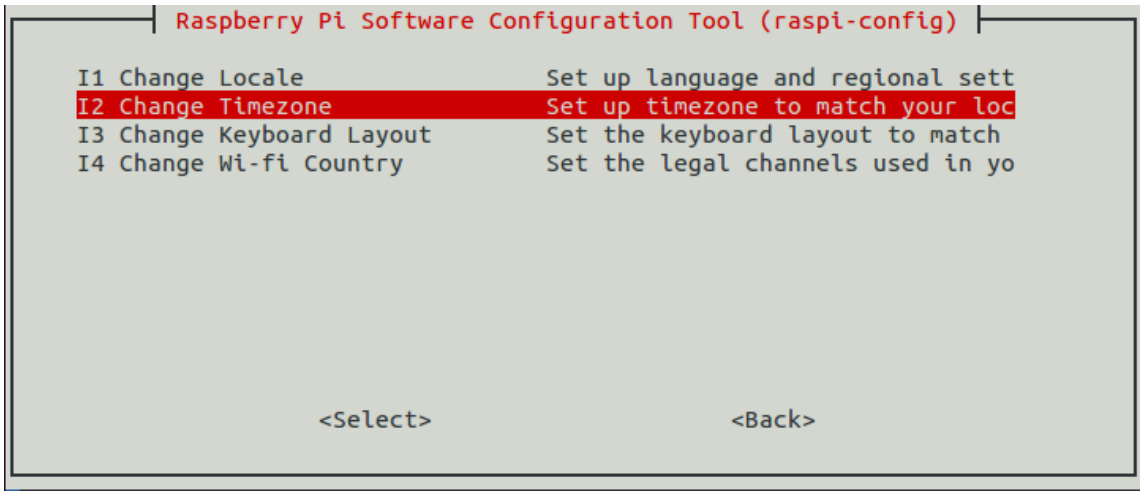


Figura 18.18: Tela do RaspiConfig

3. Selecione América(Figura 18.19):

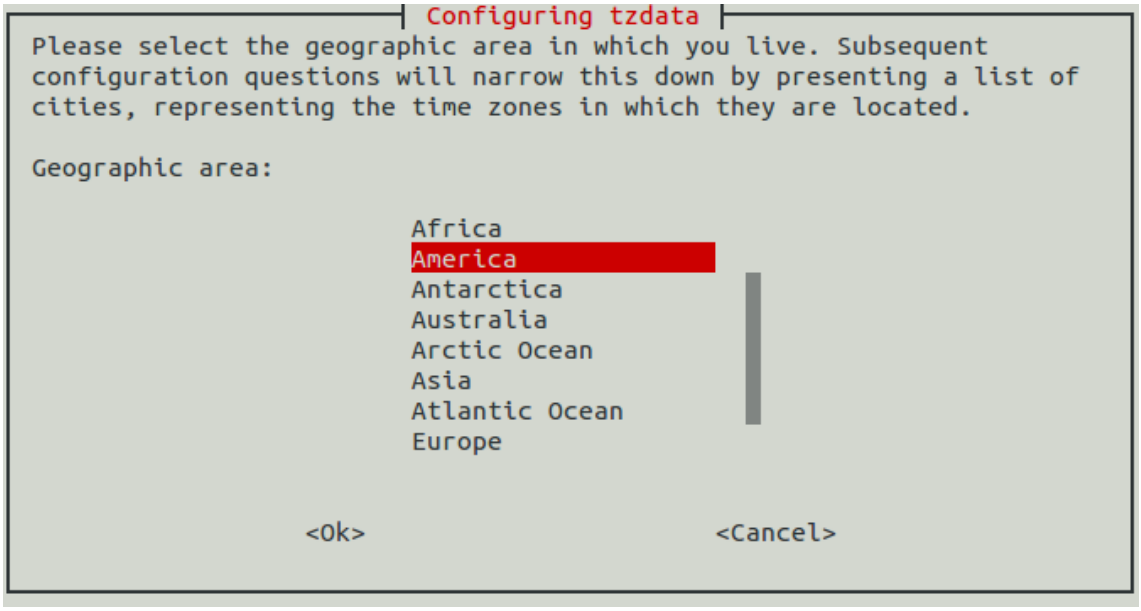


Figura 18.19: Tela do RaspiConfig

4. E por último, Fortaleza. Para sair, OK (Figura 18.20).

Figura 18.20: Tela do RaspiConfig

18.13 Configurar o teclado para uso local

Para configurar o teclado, caso use o Rpi de forma local, sem acesso remoto, com monitor, mouse e teclado acoplado a este, basta selecionar a opção I3(18.21) dentro do menu 4 (Internationalization Options).

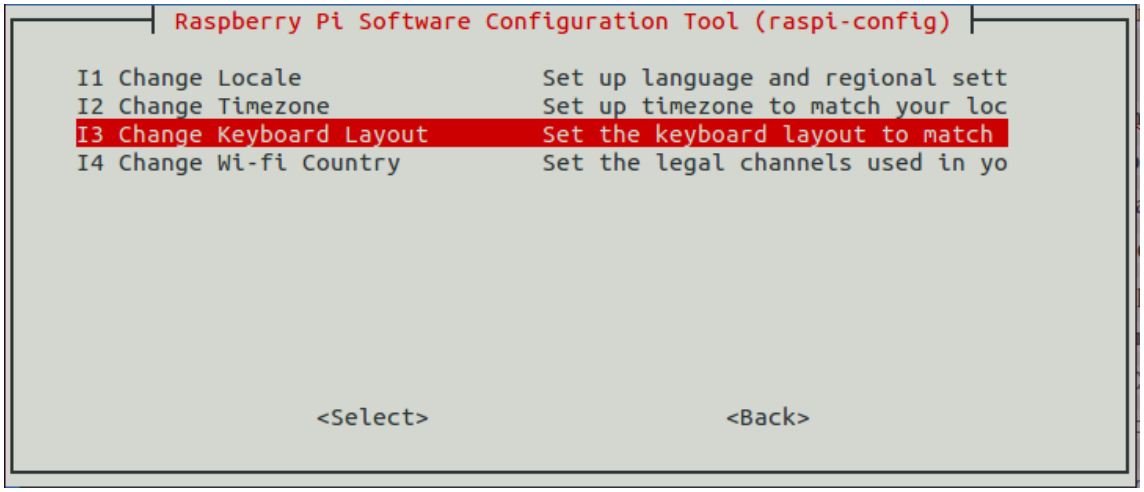


Figura 18.21: Tela do RaspiConfig

18.14 Criar, ler e escrever em um arquivo via Shell

O exemplo abaixo mostra um código exemplo em shell script para criar, ler e escrever em um arquivo shell.

```
#!/bin/bash
   # verify the psw
  if [ ! -f "./psw" ]; then
    read -p 'Digit the password of your profile: ' answer;
  echo $answer > ./psw
  #to insert new password of your profile delete the file psw (rm psw).
  fi
      answer=$(cat ./psw) # read the file content
  if [[ $answer == "" ]]; then
        rm psw
        echo -ne "Please, insert the password: \n"
    read answer
    echo $answer > ./psw
    fi
echo 1 -mostrar a senha
read op;
if [ $op -eq "1" ]; then
              cat ./psw
fi
echo "--------------------------------------------------"
```

18.15 CRON

é uma ferramenta para configurar tarefas agendadas em sistemas Unix, usado para agendar comandos ou scripts para serem executados periodicamente e em intervalos fixos; tarefas vão desde o backup de pastas base dos usuários todos os dias à meia-noite, para registrar informações, por exemplo, da CPU a cada hora.

O comando crontab (tabela *cron*) é usado para editar a lista de tarefas agendadas em operação, e é feito em uma base por usuário; cada usuário (incluindo root) tem sua própria crontab.

EDIÇÃO CRONTAB

O comando crontab dar ao usuário a capacidade de agendar tarefas para serem executadas em um momento específico ou com um intervalo específico. crontab é uma concatenação de "tabela cron"porque ele usa o cron agendador (scheduler) de tarefas para executá-las. cron é nomeado após "Cronos", a personificação grega de tempo.

Como exemplo, considere que queremos executar um script Python todos os dias às 06:00. O seguinte comando vamos editar o crontab; Execute crontab com a indicação -e para editar a tabela de cron: **crontab -e**

A primeira vez que executar crontab você será solicitado a selecionar um editor; se você não tiver certeza de qual usar, escolha o **nano**.

O layout de uma entrada cron é composta por seis componentes: minuto (m), hora(h), dia do mês (mon), mês do ano (mon), dia da semana (dow), mostrado na Figura 18.22, e o comando a ser executado.

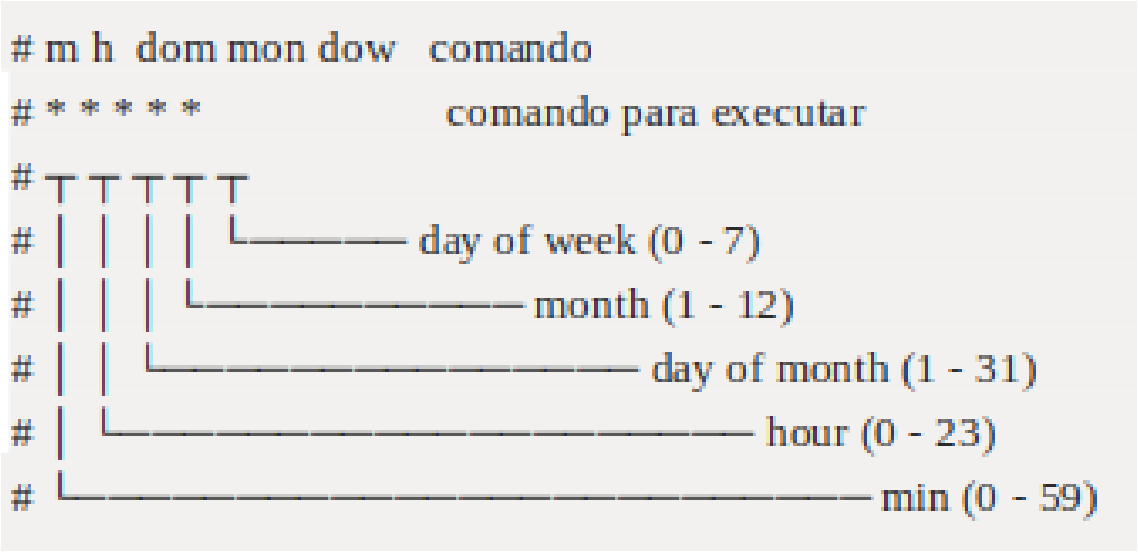


Figura 18.22: Layout de entrada CRON

Expressões cron são compostas de seis campos obrigatórios (segundos, minutos, horas, dia do mês, mês, dia da semana) e um campo opcional (ano) separados por espaços em branco(Figura 18.23):

Field Name	Valores permitidos	Caracateres especiais permitidos
Segundos (não default no Rpi)	0-59	- * /
Minutos	0-59	- * /
Horas	0-23	- * /
Dias (do mês)	1-31	* ? / L W
Mês	1-12 ou JAN-DEC	- * /
Dias (da semana)	1-7 ou SUN-SAT	- * ? / L #
Ano (opcional)	vazio, 1970-2199	- * /

Figura 18.23: Tabela de expressões CRON

Existem vários caracteres especiais que são usados para especificar os valores(Figura 18.24):

Por exemplo: Adicione a seguinte linha ao final da crontab. Ele vai salvar um timestamp, a temperatura e a humidade a cada minuto em temp.log.

*** * * * * echo ‘date +%Y%m%d%H%M%S‘,‘/home/share/dht11‘ » /home/share/temp.log**

Outro exemplo: **0 0 * * * /home/pi/backup.sh**

Esta entrada cron irá executar o script backup.sh todos os dias à meia-noite (em 0 minutos). Veja mais detalhes no capítulo 14 de aplicação do *crontab*.

Outros exemplos de configuração (somente do tempo faltando inserir o comando):
m h d m a
1 * * * * (cada hora no minuto xx: 01)
2 5 * * * (todos os dias às 05:02)
0 4 3 * * (cada terceiro dia do mês às 04:00)
* 2 * * 5 (todo minuto 2:00 - 02:59 às sextas-feiras)
*/2 * * * * (A cada 2 minutos)
00 * * * * (Todo dia de hora em hora (hora cheia))
00-59/5 * * * * (De cinco em cinco minutos todos os dias (divisão por 5 do intervalo 00-59)
15 10,12,16,18,22 * * * (Nas seguintes horas: 10, 12, 16, 18, 22 aos 15 minutos da hora)
*/1 * * * * (Script para ser executado a cada minuto).

Caracter e	Specificações	Notas
*	Todos os valores	* no campo minuto significa cada minuto.
?	Nenhum valor específico no dia do mês e dia da semana campos.	? especifica um valor de um campo, mas não o outro.
-	Uma faixa	10-12 no campo hora significa que o script será executado em 10, 11 e 12 (meio-dia)
,	Valores adicionais	Digitando "Mon, Qua, Sex" no campo Dia-de-semana, o script será executado apenas na segunda-feira, quarta-feira, e sexta-feira.
/	Incrementos.	0/15 no campo de minutos, serão incluídos os minutos 0, 15, 30 e 45. Inserindo * antes do '/' é equivalente a especificar 0 é o valor para começar.
#	Dia da semana	6 # 3 (no campo dia de semana), significa a terceira sexta-feira (dia 6 é sexta-feira; # 3 é a terceira sexta-feira do mês).
L	O último dia de um mês ou semana.	L significa o último dia do mês. Se utilizado no dia da semana, por si só campo, significa 7. Se usado no dia do campo de semana após um outro valor, isso significa que o último dia do mês.
W	O dia útil (Seg-Sex) mais próximo do dia especificado.	Especificando 15W significa que o trabalho cron irá disparar sobre o dia da semana mais próximo do dia 15 do mês. Se o dia 15 é um sábado, o gatilho disparar na sexta-feira dia 14. Se o 15 é um domingo, o gatilho é acionado na segunda-feira dia 16.

Figura 18.24: Caractéres especiais CRON

18.16 Simulação de Comunicação GPIO Raspberry usando php e mysql com Raspian no VirtualBox

Muitas vezes desenvolvemos sistemas baseados em um CI, e precisamos de portas de sinalização extra, que não estão disponíveis por padrão no CI. Como as portas GPIO não tem função definida e por padrão não são usadas, podemos usa-las para prover essa sinalização. Um exemplo, são os chips Realtek ALC260(codec de áudio) eles tem 4 pinos GPIO, que não são usados por padrão. A Acer utiliza o primeiro pino GPIO (GPIO0) para ligar o amplificador utilizado para os alto-falantes internos do laptop e fone de ouvido externo.

GPIOs são utilizados em:

- Dispositivos com pinos de escassez: circuitos integrados, tais como dispositivos lógicos programáveis system-on-a-chip, embutido e hardware personalizado, e (por exemplo, FPGAs).

- Chips Multifuncionais: gestores de energia, codecs de áudio e placas de vídeo.
- Aplicações embutidas (por exemplo, Arduino, BEAGLEBONE, kits de PSoC e Raspberry Pi) fazem uso pesado de GPIO para a leitura de vários sensores ambientais (IR , vídeo, temperatura , orientação de 3 eixos , e aceleração) , e para escrever a saída para motores DC (via PWM) , áudio, monitores LCD , ou LEDs para status.

GPIO capacidades podem incluir:

- GPIO pinos podem ser configurados para estarem de entrada ou de saída.
- GPIO pinos podem ser ativados / desativados.
- Os valores de entrada são legíveis (normalmente alto=1, baixo=0).
- Os valores de saída são graváveis / legível.
- Valores de entrada muitas vezes podem ser usados como IRQs (tipicamente para os eventos de ativação).

Periféricos GPIO variar muito amplamente. Em alguns casos, eles são muito simples, um grupo de pinos que podem ser comutados como um grupo , quer de entrada ou de saída . Em outros, cada pino pode ser configurado de forma flexível para aceitar ou fonte de tensões diferentes de lógica, com pontos fortes da unidade configurava e puxe ups / baixos. As tensões de entrada e de saída são tipicamente, embora não universalmente, limitado à tensão do dispositivo com os GPIOs sobre a oferta e pode ser danificado por maiores tensões.
Um estado do pino GPIO pode ser exposto para o desenvolvedor de software através de um de um número de diferentes interfaces, como uma memória mapeada periférica, ou através de instruções dedicadas porta IO.
Alguns têm GPIOs 5 V entradas tolerantes : mesmo quando o dispositivo tem uma tensão de alimentação baixa (tal como 2 V) , o dispositivo pode aceitar 5 V sem danos.
Uma porta GPIO é um grupo de pinos de GPIO (tipicamente oito GPIO pinos) dispostas num grupo e controlados como um grupo.

18.16.1 Prática

Descrição

Foi instalado no Oracle VirtualBox o Sistema raspbian para emular o uso do raspberry, sendo necessária a instalação do apache, mysql e php. Após as configurações referentes aos serviços citados anteriormente serem realizadas bastou publicar na pasta “www” do apache o arquivo .php que irá receber uma tela de login e comandos para inserir dados no banco mysql, juntamente com as imagens que o mesmo utiliza para ilustrar na web o acendimento do led. Também se deve executar o arquivo “.sql” que está anexado ao projeto, pois o mesmo irá criar a base de dados e toda estrutura de tabelas e informações que o sistema precisa para iniciar o funcionamento, além de executar o arquivo “.sh” que este irá coletar as informações referentes aos pinos no banco de dados e irá executar comandos para o GPIO ativar ou desativar pinos.

Preparando o ambiente

Primeiramente deve-se importar o appliance do raspbian no oracle virtualBox. O link para o passo-a-passo deste processo pode ser encontrado nas referências deste documento.

Devemos agora instalar o apache. Vejamos a seguir:

1. Atualizando o repositório
   # apt-get update
2. Instalando pacote apache2
   # apt-get install apache2
3. Acessando a pasta onde está o Apache:
   # cd /etc/apache2
4. Podemos verificar se foi criada a pasta www, também podendo testar acessando no navegador de algum cliente o ip do raspbian:
   # cd /var/www
5. Com o apache instalado iremos instalar o mysql. Instalando o Mysql:
   # apt-get install mysql-server
   Diga sim para a pergunta que será exibida na tela e espere até o final do download, ao iniciar a instalação, lhe será pedido senha de usuário root para o Mysql, crie uma senha e na sequencia confirme a mesma, após isso a instalação continuará:
6. Finalizada a instalação, agora vamos nos conectar ao MySQL e fazer um teste.
   Digite o seguinte comando para acessar o Mysql como usuário root, substitua "SENHADOROOT" pela senha que criou para o Mysql:
   # mysql -u root –pSENHADOROOT

   Será-lhe retornado algo parecido com isto(Figura 18.25:

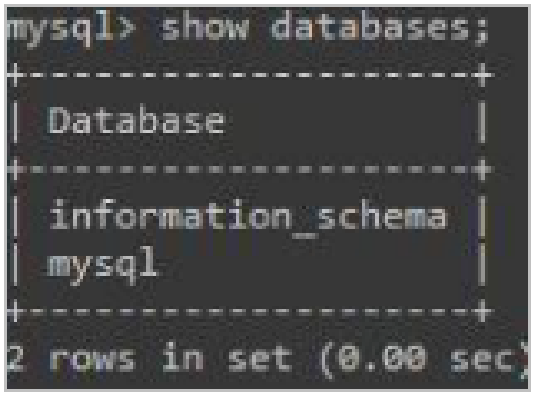


Figura 18.25: Teste de comando ao conectar no MySQL

Neste momento é interessante importar os arquivos do projeto para dentro do sistema raspbian, para que posteriormente possam ser copiados para a pasta /www e publicados. Nesta prática será utilizado o samba para compartilhar pastas e transferir os arquivos do projeto por

este meio até o raspbian.

Com todas estas configurações realizadas, falta pouco para o projeto começar a funcionar, basta agora executarmos o arquivo .sql dentro do servidor mysql para que possamos importar todo a base de dados necessária para este sistema, e após isso, copiar a pasta com os arquivos de publicação em uma pasta dentro de /var/www/, nesta prática utilizo a pasta gpio.

7. Para executar arquivos .sql devemos utilizar o seguinte comando:
   # mysql -u root –pSENHADOROOT
   # source "local de origem"/"arquivo".sql
8. E para copiar a pasta completa para dentro de www devemos executar o comando, lembrando que temos que alterar o usuário e senha do banco de dados nos arquivos .php e .sh para a que criamos no momento da instalação do mysql, pois os mesmos estão com valores padrão e vão precisar de conexão com o banco de dados:
   # cp –r "pasta de origem" /var/www/
9. Após todas estas tarefas o Sistema estará online e poderá ser visualizado através do navegador de alguma máquina cliente com o endereço "ip do raspbian"/gpio ou para visualizar através do próprio sistema pode ser usado também localhost/gpio.
10. Também deve ser executado o arquivo .sh, pois o mesmo conecta com o banco para verificar informações e alterar estado de pinos, mas como estamos utilizando uma máquina virtual ele apresentará erro por não ser possível instalar a biblioteca wiringPi. Mas para executar basta.
    # sh "nomedoarquivo".sh

18.16.2 Firmware

Os arquivos necessários para execução deste projeto sempre deverão estar postados em anexo a este documento. Segue abaixo nas Figuras18.26,18.27,18.28 e 18.29, o projeto em funcionamento:

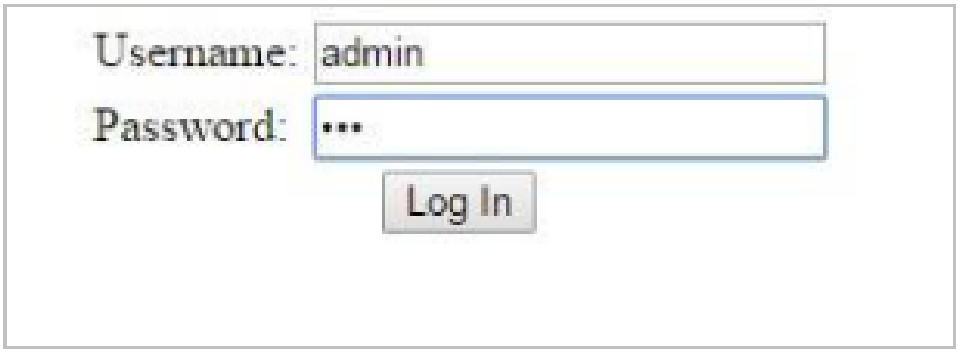


Figura 18.26: Tela inicial do projeto para login na ferramenta.

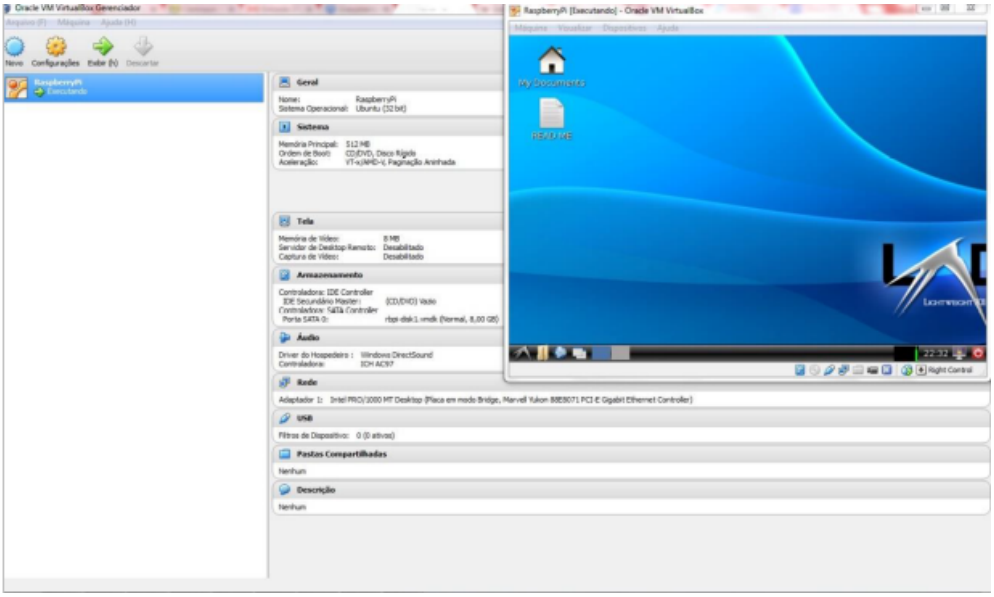


Figura 18.27: Sistema raspbian emulado no Oracle Virtual Box com 512 de memória ram.

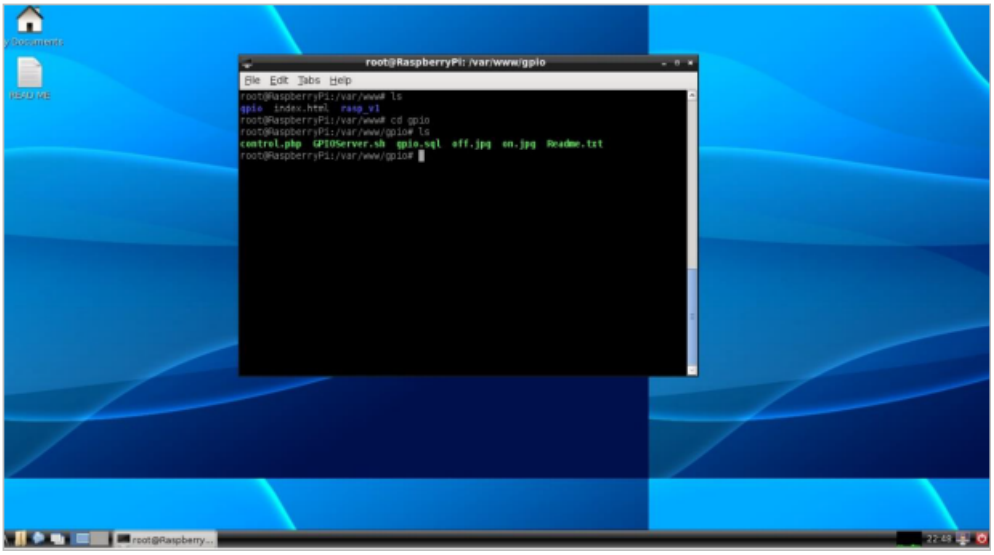


Figura 18.28: Sistema raspbian emulado no Oracle Virtual Box com 512 de memória ram.

GPIO Test Page Change Password

GPIO #	GPIO Description	Status	Action	Edit
4	Pin 4		Turn Off	Edit
17	Pin 17		Turn On	Edit
18	Pin 18		Turn On	Edit
21	Pin 21		Turn On	Edit
22	Pin 22		Turn Off	Edit
23	Pin 23		Turn On	Edit
24	Pin 24		Turn On	Edit
25	Pin 25		Turn Off	Edit

Log out

Figura 18.29: Tela principal do projeto, onde pode-se comandar os leds através de botões.

18.17 Semáforo utilizando pinos GPIO do raspberry pi e MySql

18.17.1 Prática

Descrição

Foi implementado um código em c para controlar todo o funcionamento do semáforo, onde serão utilizadas as bibliotecas wiringPi.h e mysqlclient para enviar comandos para os pinos GPIO e ler dados na base MySql respectivamente. O botão que o pedestre utiliza para solicitar passagem está em uma página web php e publicada na internet, onde ao ser acionado é atualizado um campo para 1 no banco de dados e ao ler este valor o raspberry pi irá realizar os procedimentos para tal, e após um tempo o trânsito de veículos é liberado novamente.

Preparando o ambiente

Primeiramente deve-se ter o ambiente do raspbian já embarcado no raspberry. O link para o passo-a-passo deste processo pode ser encontrado nas referências deste documento.
Devemos agora instalar a biblioteca libmysql-dev, pois neste exemplo não teremos banco de dados interno, e sim em uma hospedagem na internet. Vejamos a seguir:

1. Atualizando o repositório
   apt-get update
2. Instalando pacote libmysql-dev
   apt-get install libmysql-dev
3. Agora deve-se instalar a biblioteca wiringPi.
   sudo apt-get install git-core

18.17.2 Firmware

Para compilar e executar o código deve-se utilizar a seguinte estrutura:
gcc c_mysql.c –o c_mysql $(mysql_config –libs)
./c_mysql

```
#include <stdio.h>
#include <wiringPi.h>
#include <mysql/mysql.h>

//semaforo dos carros
int cVerd = 17; //pino 11
int cAmar = 18; //pino 12
int cVerm = 27; //pino 13

//semaforo dos pedestres
int pVerd = 22; //pino 15
int pVerm = 23; //pino 16
```

```
//botao do pedestre
int botao=0;

int verifica_botao(){
  int b;
  MYSQL conexao;
  MYSQL_RES *resp;
  MYSQL_FIELD *campos;
  MYSQL_ROW lin;
  mysql_init(&conexao);
  mysql_real_connect(&conexao, "mysql.andersonfarias.kinghost.net", "
      andersonfarias", "fco470470", "andersonfarias", 0, NULL, 0);
  if (mysql_query(&conexao,"SELECT comando2 FROM Rasp WHERE id='1';")){
        printf("Erro: %s\n",mysql_error(&conexao));
      }else
      {
     //scanf(mysql_store_result(&conexao), "%d", &botao);
        resp = mysql_store_result(&conexao);//recebe a consulta
     campos = mysql_fetch_fields(resp);
     lin = mysql_fetch_row(resp);
     sscanf(lin[0], "%d", &b);
      }
  mysql_close(&conexao);
  return b;
}

int liberar_transito(){
  int b = 0;
  MYSQL conexao;
  mysql_init(&conexao);
  mysql_real_connect(&conexao, "mysql.andersonfarias.kinghost.net", "
      andersonfarias", "fco470470", "andersonfarias", 0, NULL, 0);
  if (mysql_query(&conexao,"UPDATE Rasp SET comando2='0' WHERE id='1';")
      ){
        printf("Erro: %s\n",mysql_error(&conexao));
      }else
      {
     b=1;
     printf("Trnsito Liberado.");
      }
```

```
  mysql_close(&conexao);
  return b;
}


int pedestre(){
  printf("BOTO FOI PRECIONADO.\n");
  printf("Pedestre VERDE: VERDE\n");
   digitalWrite(cVerd,LOW);
   digitalWrite(cAmar,HIGH);
   digitalWrite(pVerm,HIGH);
   delay(3000);
   digitalWrite(pVerd,HIGH);
   digitalWrite(cAmar,LOW);
   digitalWrite(cVerm,HIGH);
   digitalWrite(pVerm,LOW);
   delay(9000);
   digitalWrite(pVerm,HIGH);
   digitalWrite(pVerd,LOW);
  digitalWrite(pVerm,LOW);
  delay(1000);
  digitalWrite(pVerm,HIGH);
  delay(1000);
  digitalWrite(pVerm,LOW);
  delay(1000);
  digitalWrite(pVerm,HIGH);
  delay(1000);
  digitalWrite(cVerm,LOW);
  digitalWrite(cVerd,HIGH);

   if(liberar_transito()==0){
     liberar_transito();
   }

   delay(2000);
}

int main(void){

   wiringPiSetupGpio();
```

```
  pinMode(cVerd, OUTPUT);
  pinMode(cAmar, OUTPUT);
  pinMode(cVerm, OUTPUT);
  pinMode(pVerd, OUTPUT);
  pinMode(pVerm, OUTPUT);



  while(1){

    digitalWrite(cVerd,HIGH);printf("Carro: VERDE\n");
    digitalWrite(pVerm,HIGH);printf("Pedreste: VERMELHO\n");
   printf("\n\n\n\n\n\n\n\n");
    digitalWrite(cAmar,LOW);
    digitalWrite(cVerm,LOW);
    digitalWrite(pVerd,LOW);

   while(1){
   delay(1000);
   botao = verifica_botao();
     if(botao==1){
       pedestre();
       break;
     }
   }


  }
  return 0;
}
```

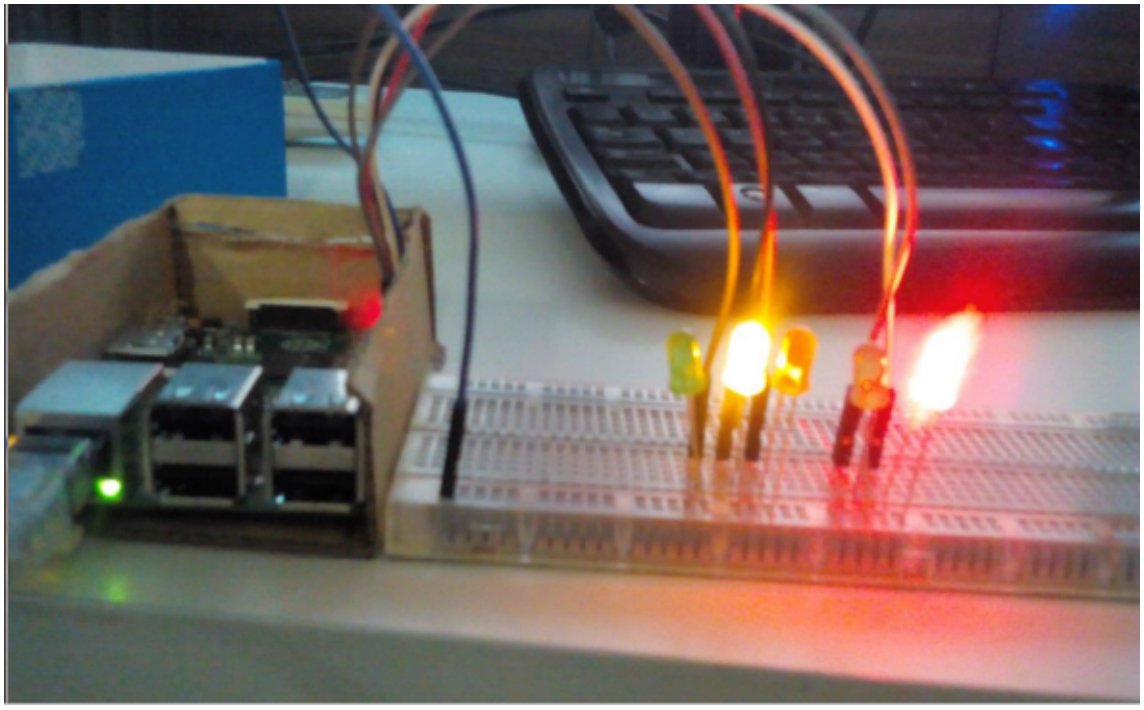


Figura 18.30: Prática em funcionamento com protoboard e raspberry pi 2 conectado à internet.

18.18 Controle WiFi de Robô móvel (motor CC) pelo Rpi via SSH

//Controle WiFi de Robô móvel (motor CC) pelo Rpi via SSH

```
//Controle WiFi de Robô móvel (motor CC) pelo Rpi via SSH
#include <wiringPi.h>
#include <stdio.h>

#include <unistd.h>
#include <termios.h>
#include <fcntl.h>

int getch(void)
{
    int ch;
    struct termios oldt, newt;
    long oldf, newf;

    tcgetattr(STDIN_FILENO, &oldt);
    newt = oldt;
    newt.c_lflag &= ~(ICANON | ECHO);
    tcsetattr(STDIN_FILENO, TCSANOW, &newt);
```

```
    oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
    newf = oldf | O_NONBLOCK;
    fcntl(STDIN_FILENO, F_SETFL, newf);
    ch = getchar();
    fcntl(STDIN_FILENO, F_SETFL, oldf);
    tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
    return ch;
}

//make global variables to store the wiring pi pin numbers for the
    motors
int m1a = 0;
int m1b = 1;
int m2a = 3;
int m2b = 4;

//make robot go forwards
void forwards()
{
  digitalWrite(m1a, HIGH);
  digitalWrite(m1b, LOW);
  digitalWrite(m2a, HIGH);
  digitalWrite(m2b, LOW);
}

//all motors off
void stop()
{
  digitalWrite(m1a, LOW);
  digitalWrite(m1b, LOW);
  digitalWrite(m2a, LOW);
  digitalWrite(m2b, LOW);
}

//Make both motors turn backwards
void reverse()
{
  digitalWrite(m1a, LOW);
  digitalWrite(m1b, HIGH);
  digitalWrite(m2a, LOW);
  digitalWrite(m2b, HIGH);
```

```
}

//Make motors turn fwd, back
void left()
{
  digitalWrite(m1a, HIGH);
  digitalWrite(m1a, LOW);
  digitalWrite(m1a, LOW);
  digitalWrite(m1a, HIGH);
}

//Make motors turn fwd, bak
void right()
{
  digitalWrite(m1a, LOW);
  digitalWrite(m1a, HIGH);
  digitalWrite(m1a, HIGH);
  digitalWrite(m1a, LOW);
}

//main function
int main(void)
{
    int x;
    do {
        delay(2000);
        if ((x = getch()) != EOF) {
            switch(x) {
                case 'L':
                case 'l':
                    forwards();
                    printf("L was pressed \n");
                    break;

                case 'D':
                case 'd':
                    right();
                    printf("D was pressed \n");
                    break;

                case 'P':
```

```
            case 'p':
                left();
                printf("P was pressed \n");
                break;

            case \'T':
            case 't':
                reverse();
                printf("T was pressed \n");
                break;
        }
    } else {
        //printf("stop");
        stop();
    }
} while (x != 'q'); //q -> quit

return 0;
}
```

Método passo a passo para enviar dados para o Google Drive

Veja agora como é feito o passo a passo para enviar dados via WiFi para o Google drive atrvés de acesso a URL:

1. Inicialmente é necessário criar um formulário de página do *Google Drive* (você deve estar conectado) que armazena os dados inseridos em uma planilha gerada automaticamente.
2. Selecione "Novo-> "Formulários"do Google Drive menu (Figura 18.31).
   Feche se aparecer *lay-out* personalizado.

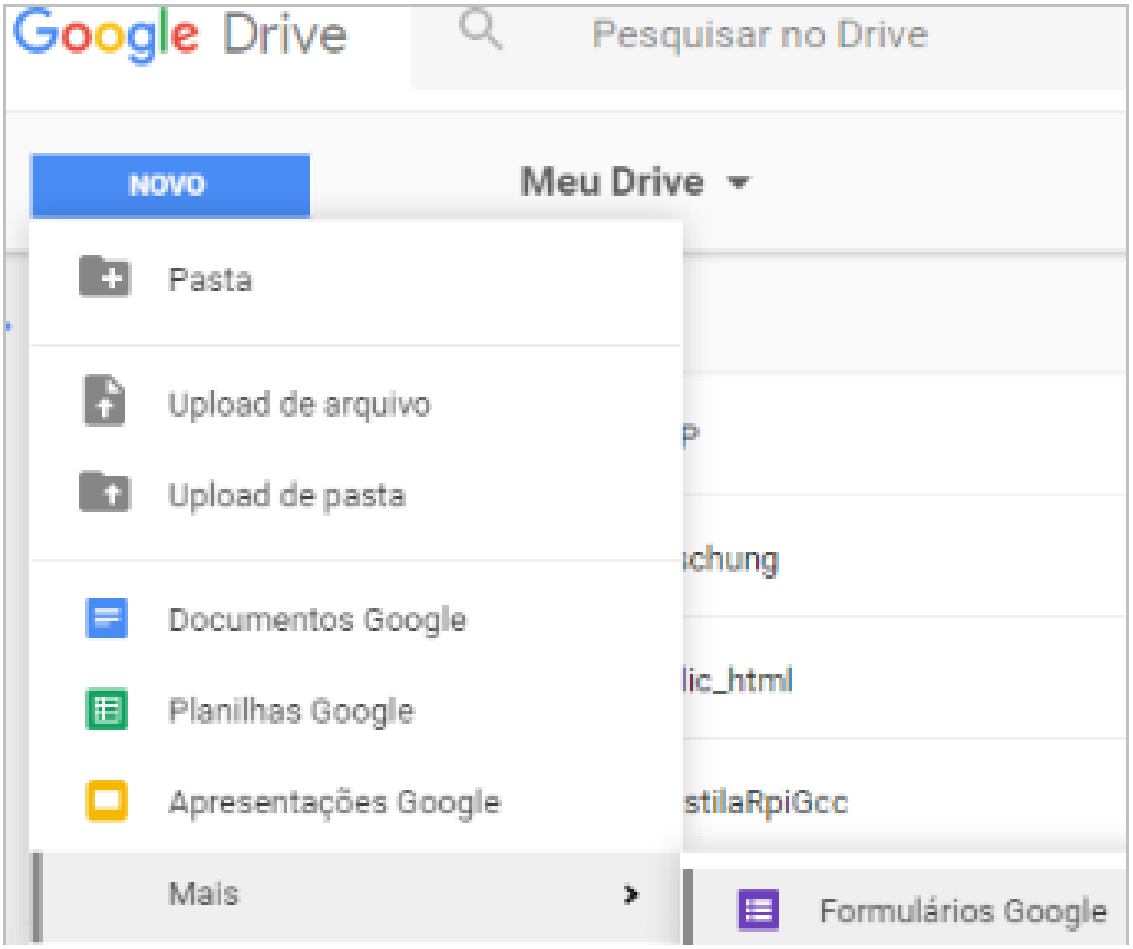


Figura 18.31: Opções do Google Drive

3. **Criar o formulário selecionado resposta curta**. Insira quantos Itens seja necessário, clicando em **Duplicar**. Dê um nome ao formulário e às perguntas **(os títulos das perguntas serão os nomes das colunas da tabela de sensoriamento)**:
4. Clique em **Visualizar e o formulário publicado será visualizado** (Figura 18.32).

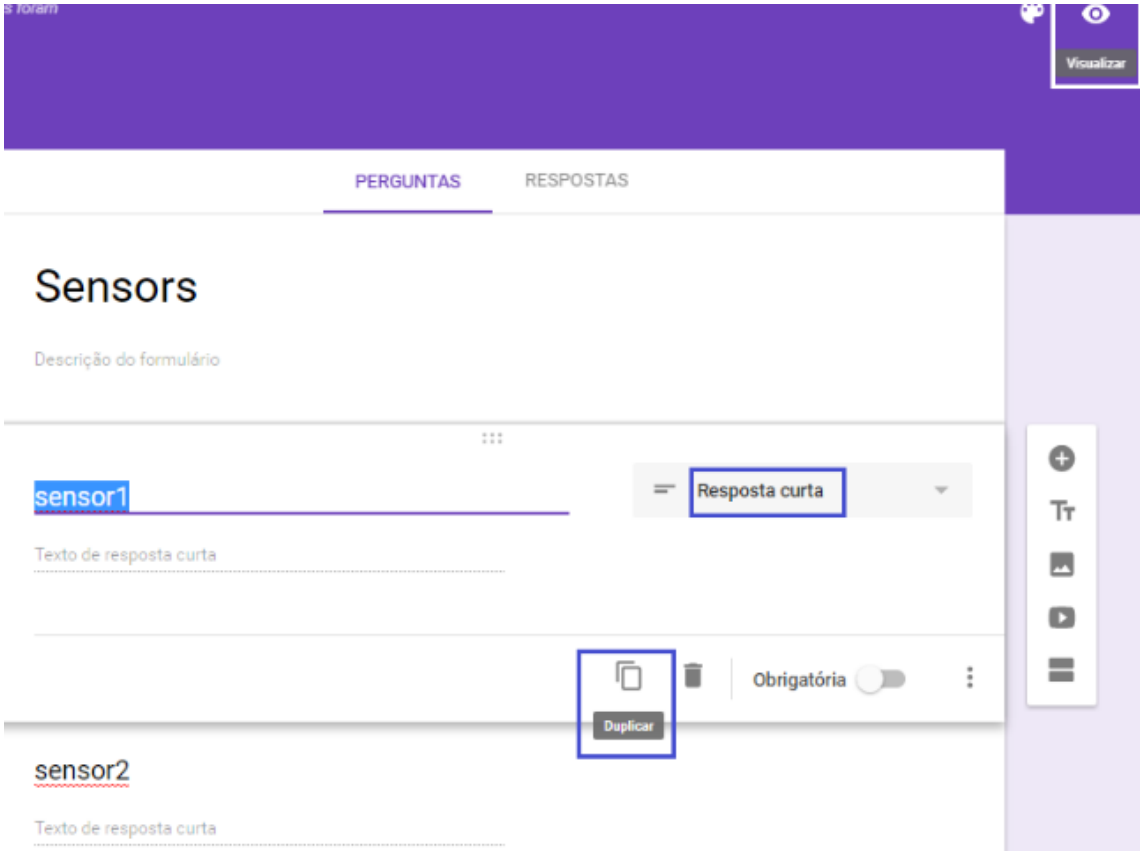


Figura 18.32: Criação de formulário no Google Drive

Clicando na Aba **RESPOSTAS**, aparece a opção de criar uma planilha vinculada para armazenar os dados através do nome do formulário (Figura 18.33).

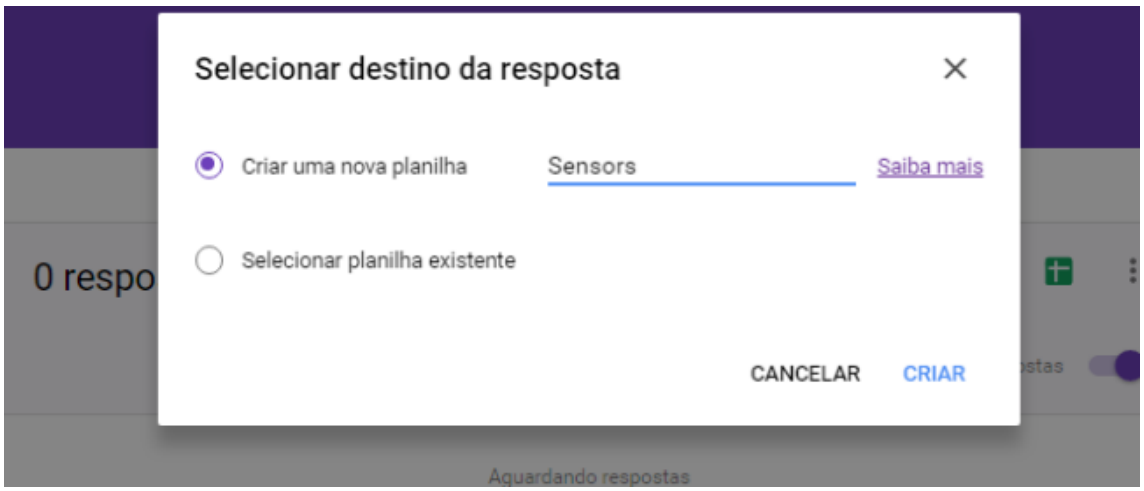


Figura 18.33: Endereçamento de respostas

Clicando em **Visualizar** e clicar em Mais e **gerar link preenchido automaticamente**, aparecerá o formulário com o endereço como no exemplo abaixo em azul:
https://docs.google.com/\forms/d/19K_BHbwr03gC232UbLgqOrDzY4j6cG0wN9Ictdzebts/

viewform

Assim, o endereço do formulário criado é https://docs.google.com/forms/d/19K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts.

5. Cada entrada de sensor recebe um nome (name), por exemplo, **entry.955491578, entry.1225247694**, etc. que pode ser visto explorando o código HTML, clicando com o botão direito dentro da caixa de texto e em inspecionar elemento e levante a aba para ver o script (Figura **??**.

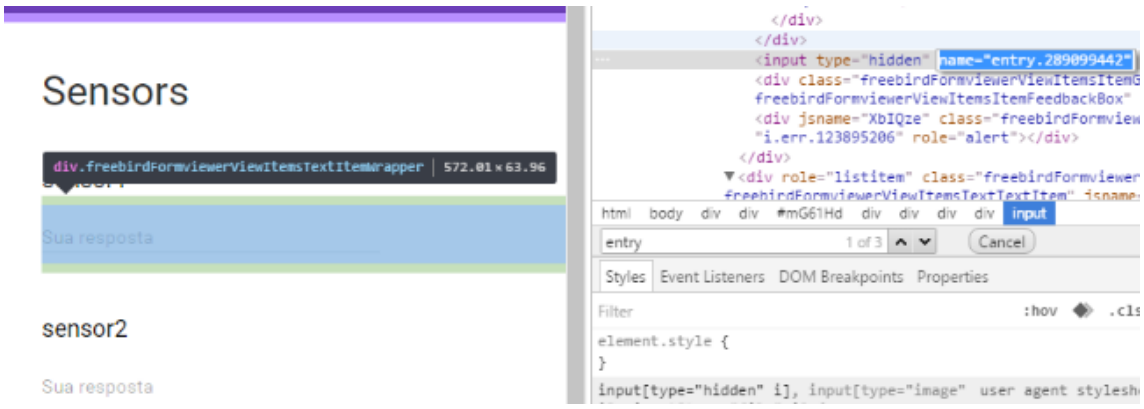
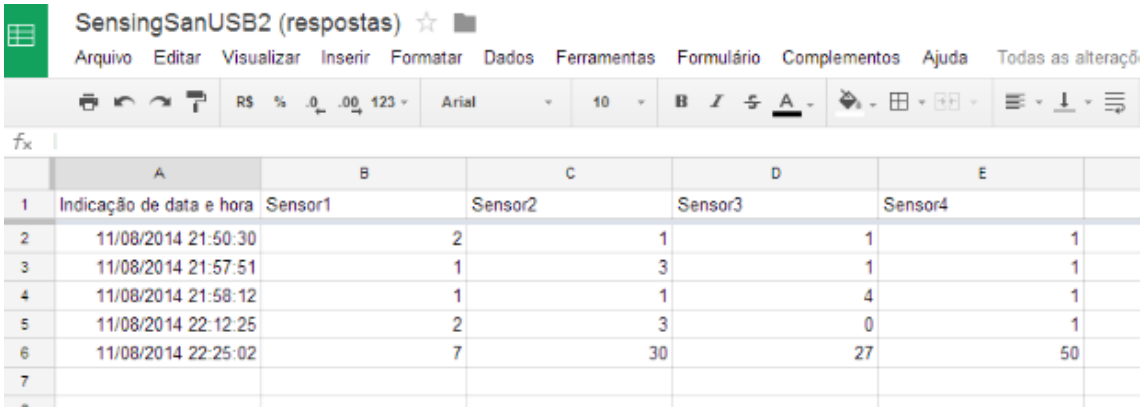


Figura 18.34: Nome das entradas em código HTML

6. Dessa forma, os nomes das entradas definidas por entry são **entry.289099442** de sensor1, **entry.711178561** do sensor2 e **entry.648964283** do Sensor3. Para enviar dados para o formulário, é necessário utilizar esta sintaxe:
lhttps://docs.google.com\/forms/d/Endereçodoformulario/formResponse?&ifq%20&entrydosensor=Valor&submit=Submit

   Exemplo: https://docs.google.com\/forms/d/19K_BHbwrO3gC232UbLgqOrDzY4j6cGOwN9Ictdzebts/formResponse?ifq%20&entry.289099442=9&entry.711178561=30&entry.648964283=29&submit=Submit
7. Os dados são copiados na planilha da Google criada automaticamente anteriormente (18.35)
https://docs.google.com/spreadsheets/d/1GswcRQUjO6BA2X7jy1LUypzOI2OzVwdZw8Pynnz9FjQ/edit#gid=943056195



SensingSanUSB2 (respostas)

Arquivo Editar Visualizar Inserir Formatar Dados Ferramentas Formulário Complementos Ajuda Todas as alteraçõ

	A	B	C	D	E
1	Indicação de data e hora	Sensor1	Sensor2	Sensor3	Sensor4
2	11/08/2014 21:50:30	2	1	1	1
3	11/08/2014 21:57:51	1	3	1	1
4	11/08/2014 21:58:12	1	1	4	1
5	11/08/2014 22:12:25	2	3	0	1
6	11/08/2014 22:25:02	7	30	27	50
7					

Figura 18.35: Armazenamento de código em forma de planilha

8. É utilizado o método POST para enviar dados para o formulário. Este código armazena no banco de dados da Google, em forma de planilha, o valor da entrada dos sensores.

   Outro exemplo aqui

   Visualização: https://docs.google.com/spreadsheets\/d/1Yc-vqcrtvjtn-ufhCZyTQ1_U33xQs02IB8qh7Hw4SVY/edit#gid=618836785

19. Plataforma Firebase

O Firebase é uma plataforma *Backend as a Service* (*BaaS*) que permite armazenar e sincronizar dados entre produtos da mesma plataforma. Além de serviços como autenticação, banco de dados e hosting, comumente mencionadas para aplicativos web, smartphones e serviços js, o Firebase também provê suporte para o desenvolvimento de aplicativos de jogos, por meio de SDKs.

Geralmente, a linguagem padrão para consulta de banco de dados é aStructured Query Language (SQL), a qual não é usada pelos bancosNot Only SQL (NoSQL), que no caso do Firebase utiliza notações em JavaScript Object Notation (JSON).

Essa plataforma, gratuita até 1GB de dados e 5GB de armazenamento, disponibiliza atualmente os seguintes produtos:

Authentication - O Firebase ajuda a autenticar e gerenciar usuários que acessam os aplicativos.

Realtime Database - O banco de dados em tempo real do Firebase, como o nome sugere, permite armazenar e consultar dados JSON em tempo real.

Hosting - é um recurso de hospedagem de conteúdo estático e dinâmico (js) na Web em nível de produção para desenvolvedores. O Firebase Hosting hospeda todos os tipos de conteúdo, desde seus arquivos CSS e HTML até microsserviços Express.js e APIs.

Storage - permite fazer upload e armazenar conteúdo gerado pelo usuário, como arquivos e imagens.

Cloud Firestore - é um banco de dados flexível e escalável para desenvolvimento em dispositivos móveis, aplicações web e no Google Cloud Platform.

Cloud Messaging - é uma solução de mensagens entre plataformas que permite entregar mensagens de forma confiável, sem nenhum custo.

19.1 Criando um Projeto Firebase

Um projeto Firebase dá-lhe um espaço de nome exclusivo para seus dados.Atualmente, Firebase não é apenas um banco de dados em tempo real, mas é todo um conjunto de serviços que torna mais fácil para nós para escrever aplicações móveis. Dispõe de uma base de dados, notificações, ferramentas de desenvolvimento, autenticação, os relatórios de falhas, análise e muito mais.

No nosso caso, no entanto, será focada no banco de dados único, ou seja, uma área de armazenamento na nuvem para se autenticar os nossos valores de temperatura. Em outras palavras, você pode pensar no projeto como um logger de Temperatura onde eu quero registrar a temperatura a ser recolhidos a partir de várias estações.

Ao acessar `https://console.firebase.google.com` aparecerá uma tela de projeto, como mostrado abaixo, onde você precisará fornecer um nome de projeto.Note que você precisará fornecer um nome único para o seu projeto. Selecione um país / região onde você gostaria que seu banco de dados a ser hospedado.Clique em *Create Project* conforme a figura 19.1

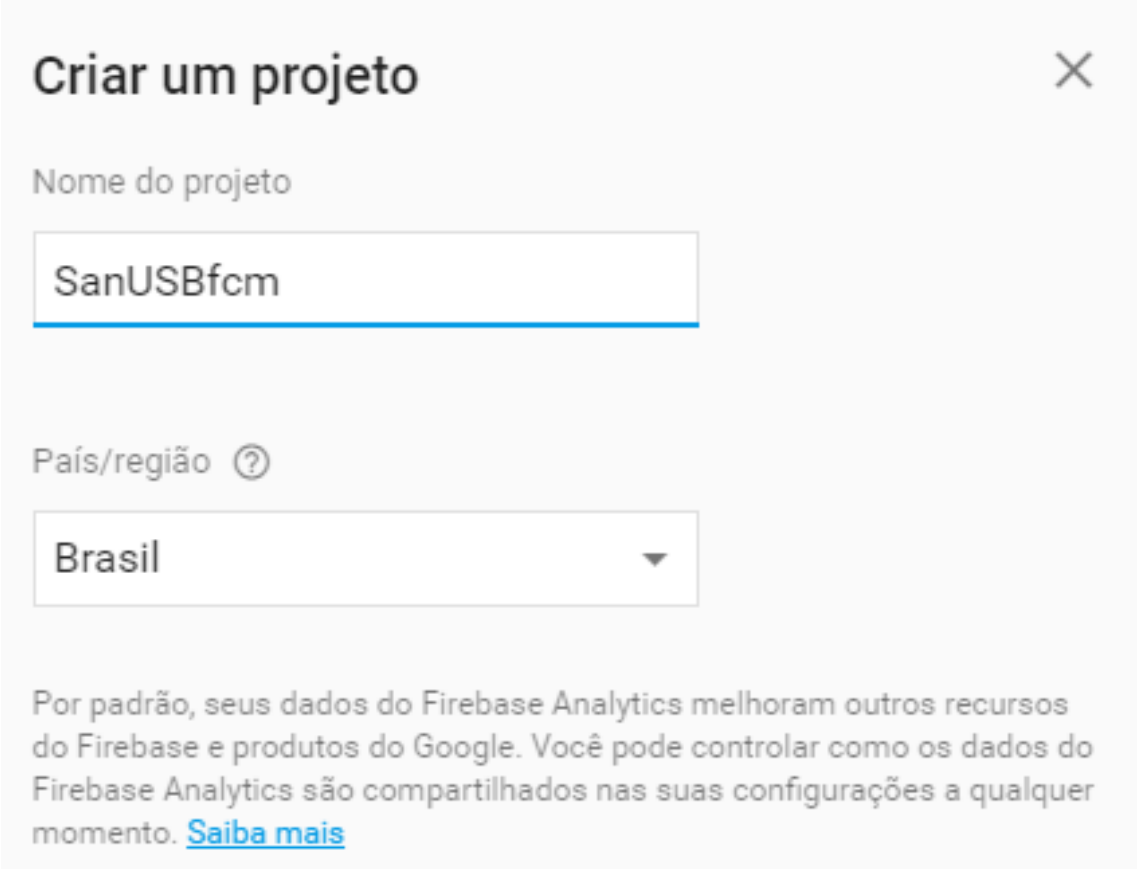


Figura 19.1: Tela de projeto Firebase

no link Database no lado esquerdo. Isto irá mostrar-se alguns detalhes sobre seu banco de dados, conforme mostrado na figura 19.2

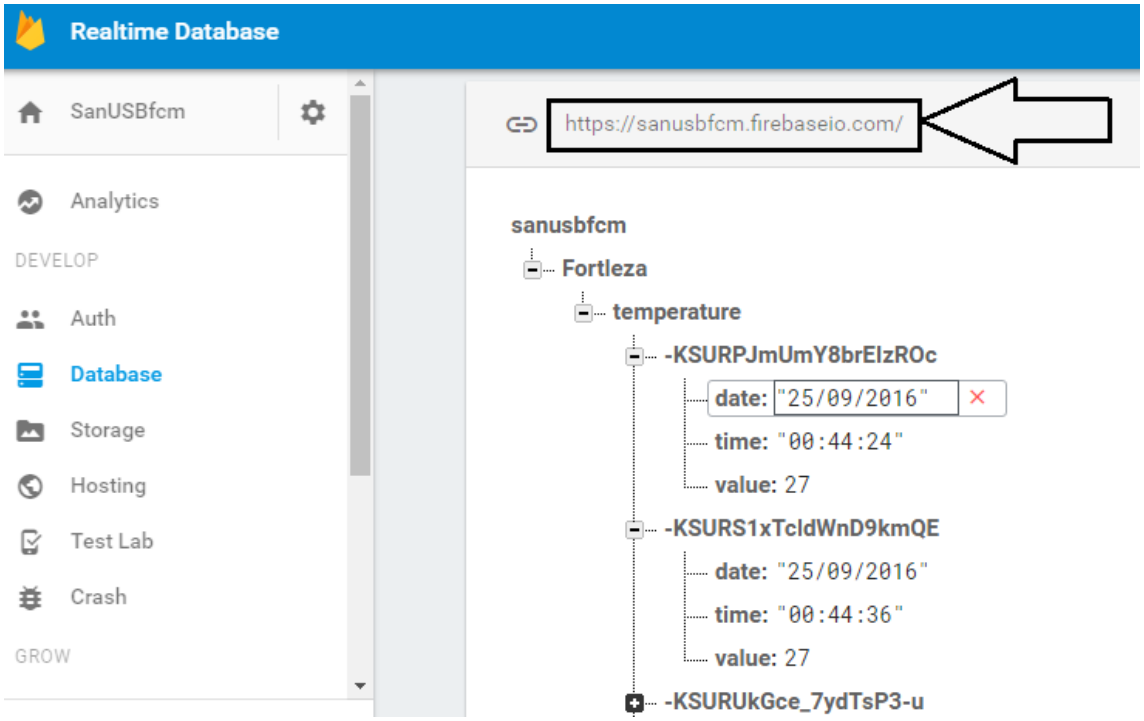


Figura 19.2: Detalhes do banco de dados

Importante: Anote o seu URL do banco de dados como destacado na tela acima. Este é o banco de dados de URL original que está disponível para o mundo exterior para a integração.

Clique na guia Regras. O Firebase suporta múltiplos mecanismos de autenticação, mas vamos mantê-lo simples por agora e não usar qualquer tipo de autenticação. Note-se que o que vamos fazer não é uma boa prática, mas é ok para o nosso tutorial aqui. O que vamos fazer é abrir acesso a esta base de dados Firebase permitir que qualquer pessoa possa ler e escrever. Para fazer isso, basta editar o texto como é mostrado na figura 19.3

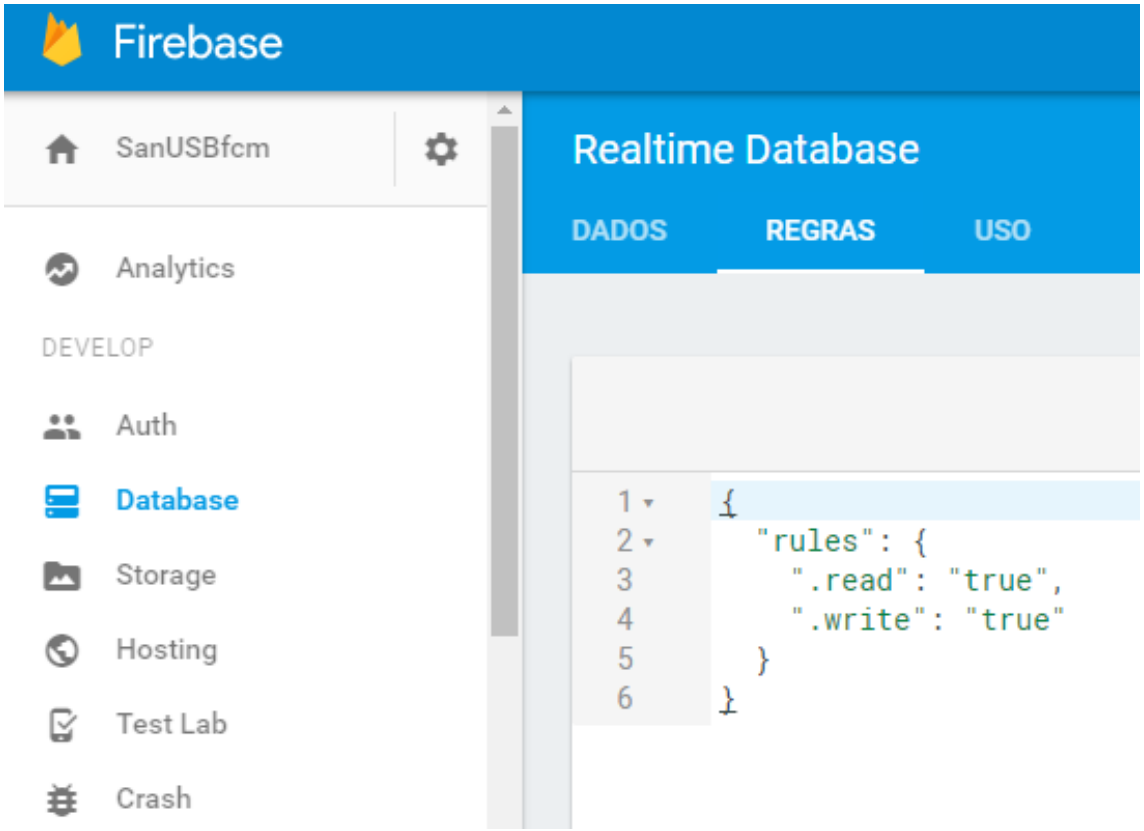


Figura 19.3: Liberando o acesso do banco de dados

Clique em Publicar. Isto irá publicar (associado) as regras com o seu banco de dados. Ele também indica claramente que definimos nosso banco de dados a ser acessado pelo público (ler e escrever).

É aconselhável ler a API REST do Firebase, uma vez que ajuda a entender um pouco melhor sobre o que está acontecendo. Confira adocumentação para uma lista de bibliotecas do lado do cliente que estão disponíveis.

Código Python

Dê uma olhada no programa cliente Python no Rpi abaixo:

```
#sudo apt-get install python-dev
#sudo apt-get install python-pip
#sudo pip install requests==1.1.0
#import serial
import time
import requests
import json
firebase_url = 'https://sanusbfcm.firebaseio.com/'
#Connect to Serial Port for communication
#ser = serial.Serial('ttyAMA0', 19200, timeout=0)
```

```
#Setup a loop to send Temperature values at fixed intervals
#in seconds
fixed_interval = 5
while 1:
  try:
    #temperature value obtained from microcontroller + LM35 Temp Sensor
    temperature_c = 27 #ser.readline()

    #current time and date
    time_hhmmss = time.strftime('%H:%M:%S')
    date_mmddyyyy = time.strftime('%d/%m/%Y')

    #current location name
    temperature_location = 'Fortaleza';

    #insert record
    data = {'date':date_mmddyyyy,'time':time_hhmmss,'value':
        temperature_c}
    result = requests.post(firebase_url + '/' + temperature_location + '/
        temperature.json', data=json.dumps(data))

    print 'Record inserted. Result Code = ' + str(result.status_code) +
        ',' + result.text
    time.sleep(fixed_interval)
  except IOError:
    print('Error! Something went wrong.')
  time.sleep(fixed_interval)
```

Verificando os nossos dados

Supondo que todo o hardware está ligado e você tem o seu código Python pronto, executar o código é simples. Um exemplo de execução do código Python no Rpi é mostrado na figura 19.4

```
root@rpiusb:/home/share# python firebaseclient.py
Record inserted. Result Code = 200,{"name":"-KSUdUPoNqgP2EqWfAie"}
Record inserted. Result Code = 200,{"name":"-KSUdX7vTZIbIScdCrIn"}
Record inserted. Result Code = 200,{"name":"-KSUdZsZDsn3jw-uiPDE"}
```

Figura 19.4: Execução do código no RPI

O passo final foi para validar se os nossos dados passaram a ser transmitidos com sucesso noFirebase. Vamos visitar o Painel do Firebase e será visto algo semelhante à tela abaixo:

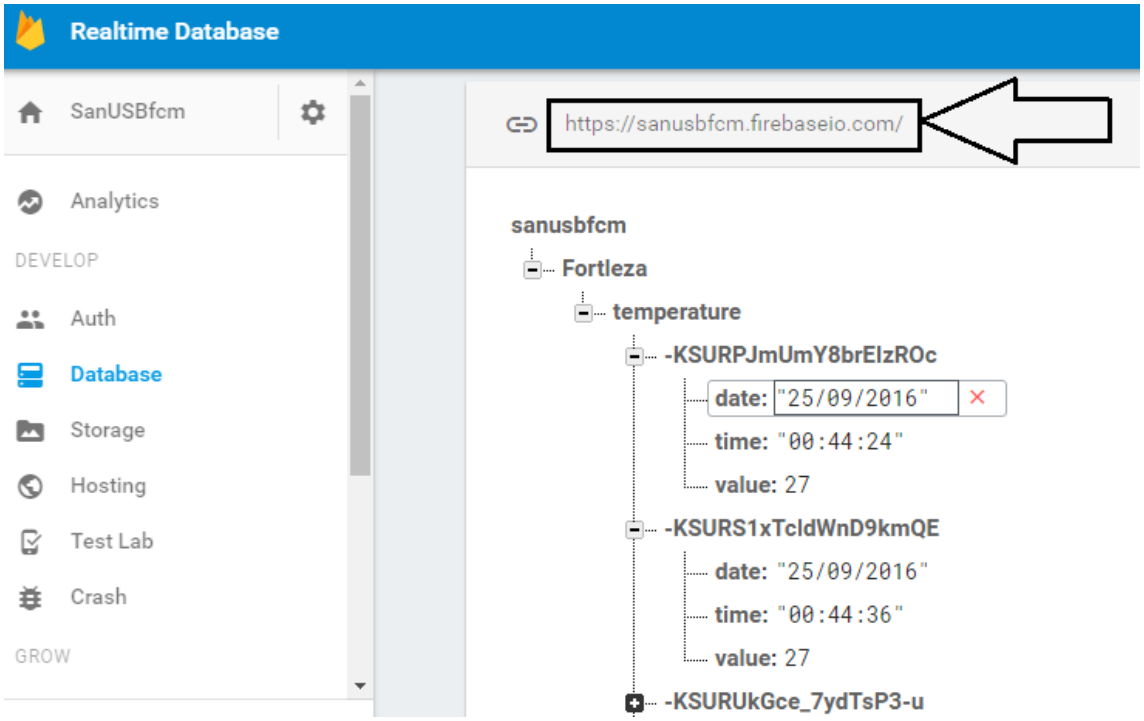


Figura 19.5: Painel do Firebase

A figura 19.5 acima mostrada quando um registro é inserido. Observe que Firebase criada automaticamente o nome do local (Fortaleza) e o nó para o registro da temperatura sob ele e, finalmente, os atributos (data, hora, valor) como descrito no script.

Para visualizar a resposta JSON dos dados inseridos no banco Firebase, basta escrever a URL em um navegador que será retornado a árvore de valores.

Resposta: 15

Executando um json no diretório anterior aparecerá todos os dados do diretório em formato json:

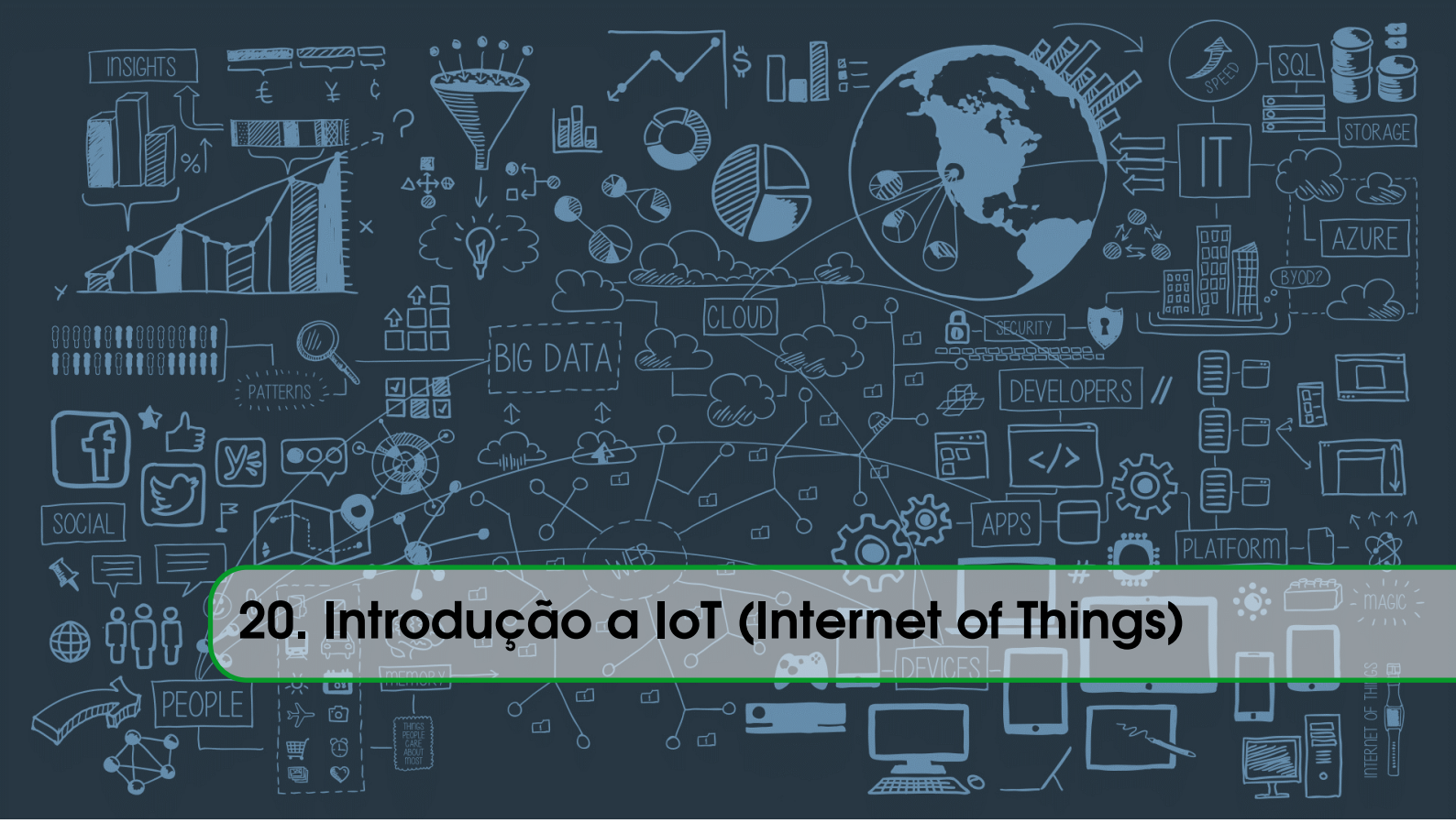
Resposta: "Corrente":15,"Luminosidade":"Lux":"350

,"Lux":"350","name":"first":"SanUSB","last":"Laese

Para ler e escrever no Firebase utilizando javascript (escrever) e php (ler) é mostrado um exemplo a seguir:

```
<html>
  <head>
              <link rel="icon"
    href="http://sanusb.org/images/ico16.ico"/>
    <script type="text/javascript" src="https://cdn.firebase.com/js/
        client/2.4.2/firebase.js"></script>

<!-  http://sanusb.org/exemplos/testeFireBase.php e https://console.
```

```
    firebase.google.com/project/sanusbfcm/database/data ->
  </head>
  <body>

    <script>

      var config = {
       // apiKey: "AIzaSyAoo6QhxZSmHnjTAKKvCleeNRp2Cpkne-c",
        authDomain: "sanusbfcm.firebaseapp.com",
        databaseURL: "https://sanusbfcm.firebaseio.com",
        storageBucket: "",
        //messagingSenderId: "892183778812"
      };
      firebase.initializeApp(config);
    </script>

    <script>

      var candidato = new Firebase('https://sanusbfcm.firebaseio.com/
          Votacao3');
      candidato.child('Candidato1').set(8);
      candidato.child('Candidato2').set(10);
      candidato.child('Candidato3').set(9);

    </script>

    <?php
      $Candidato3 = file_get_contents('https://sanusbfcm.firebaseio.com/
          Votacao3/Candidato3.json'); //Para ler valores
      echo $Candidato3;
    ?>

  </body>
</html>
```



20. Introdução a IoT (Internet of Things)

20.1 Conceito de IoT e Introdução ao MQTT

No modelo convencional de comunicação WEB, toda nova notificação somente será realizada se houver uma requisição do cliente. Isto torna o modelo ineficiente, pois em diversas situações é necessário atualizar o cliente no momento em que ocorreram modificações nos dados do servidor, sob pena da informação ficar depreciada, principalmente em redes sociais, bolsa de valores, informações climáticas entre outros. Outro problema do modelo, seria de que o usuário necessitaria enviar diversas requisições para o servidor, com a intenção de saber se existe alguma atualização da informação (pulling), aumentando assim o processamento do dispositivo do usuário e também gerando um desconforto, pois na maioria dos casos não há um tempo exato para a informação ser atualizada.

Uma solução é inverter os papéis, fazendo o servidor acordar o cliente, sempre que uma nova mensagem chegar. Este modelo é conhecido por Push. **Push refere-se a um modelo de comunicação entre cliente-servidor onde é o servidor que inicia a comunicação. Neste modelo, é possível que um servidor envie dados para um cliente sem que ele tenha explicitamente solicitado.**

Esta tecnologia de envio assíncrono de notificações por parte do servidor é o modelo mais implementado em IoT e também utilizado em diversos sistemas operacionais como o iOS, utilizando o *Apple Push Notification Service* (APNS); no Windows Phone com o *Microsoft Push Notification Service (MPNS)* e no Android usando o *Google Cloud Messaging (GCM)* ou o *Firebase Cloud Messaging (FCM)* que é a nova versão do GCM.

O MQTT (***significa Message Queuing Telemetry Transport***) é um protocolo leve, criado pela IBM em 1999, que permite a comunicação bidirecional (não como o protocolo HTTP, mas a seu

modo específico) para coleta de dados e gerenciamento de dispositivos da Internet das Coisas.

O **MQTT** perimite o envio de mensagens para **filas**, e estas podem ser escutadas por outros dispositivos inscritos. O protocolo utiliza uma arquitetura publish/subscribe, em contraste ao request/response do protocolo web HTTP. A insustentável leveza desse ser consagrou-lhe como o protocolo oficial da Internet das Coisas (IoT) e das conversas entre máquinas (M2M), com seu uso largamente estimulado por gigantes como a Intel e IBM.

Ele segue o padrão de projetos chamado observer, onde um grupo de objetos interessados assinam (subscribe) um determinado tópico. Outros objetos publicam informações nesses tópicos e os assinantes recebem. De acordo com a Figura 20.1

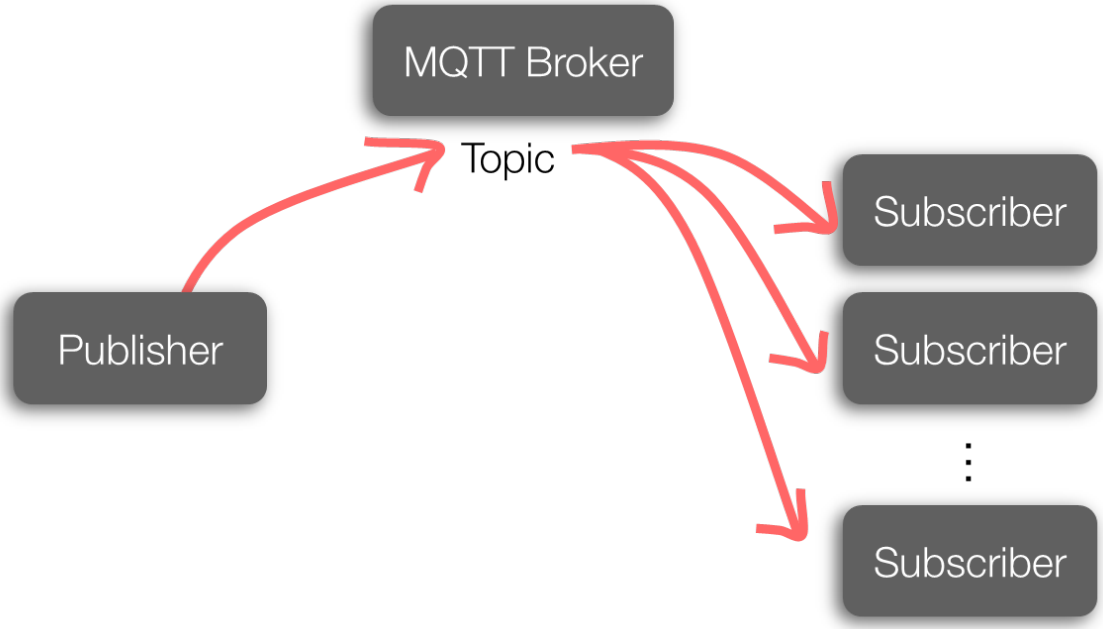


Figura 20.1: Recepção e transmissão de mensagens pelo Broker

O centralizador ou servidor de notificação do MQTT é chamado de broker que funciona como o servidor de notificação no método push. Ele que recebe e envia mensagens para todos os clients que estabelecerem conexão e todos os dispositivos devidamente inscritos para receber a notificação ou *payload.*

O protocolo MQTT é baseado no TCP/IP e ambos, cliente e broker, necessitam da pilha TCP/IP para o seu funcionamento. O MQTT está na mesma camada OSI que o HTTP, porém a maior diferença entre os dois protocolos é o tamanho do payload. No HTTP, o payload é maior, o que inviabiliza o seu uso em conexões de baixa qualidade, como GSM por exemplo.

O publisher é responsável por se conectar ao broker e publicar mensagens. Já o subscriber é responsável por se conectar ao broker e receber as notificações ou mensagens que ele tiver interesse. Na figura 20.2 é possível observar a arquitetura do paradigma pub/sub.

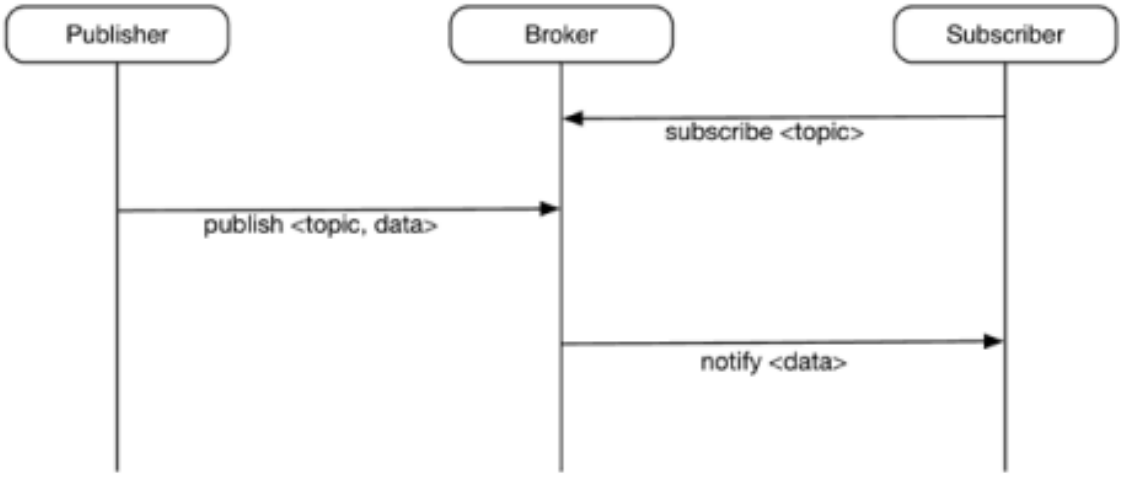


Figura 20.2: Conexão Publisher-Broker-Subscriber

Para os nossos testes usaremos o broker mqtt.eclipse.org. É muito fácil usá-lo, basta acessar o endereço mqtt.eclipse.org na porta 1883 com um cliente MQTT como o paho e pronto. Na realidade é um broker Mosquitto aberto ao público.

O servidor mosquito está para o MQTT assim como o Apache está para o HTTP, ele fará a conexão como servidor broker entre o pub/sub.

Caso prático de uso:

Como foi visto, o MQTT (Message Queue Telemetry Transport) é um leve e eficiente protocolo voltado para IoT. Possui header extremamente compacto e exige pouquíssimos recursos de hardware, processamento e baixíssima banda para funcionar. Como é calcado no TCP, possui também portanto garantia de entrega das mensagens (algo essencial em IoT). Além disso, utilizando-se de um servidor (broker) na nuvem, a comunicação torna-se a nível global. Ou seja, é possível interagir via MQTT com um Raspberry Pi de qualquer lugar do planeta que possua acesso à Internet. Outro ponto interessante é que este protocolo possui implementações nos mais diversos sistemas operacionais, permitindo assim que dispositivos totalmente diferentes "conversem"pela Internet.

Servidor Broker utilizado

Há muitos servidores brokers disponíveis na Internet (tanto grátis quanto pagos). Neste exemplo, utilizaremos o broker oficial do Eclipse (iot.eclipse.org), pois consiste de um broker grátis e com baixo downtime.

Preparação

Antes de codificar o projeto, primeiro é necessário baixar o Paho-MQTT. O Paho-MQTT trata-se do cliente MQTT oficial, sendo este disponível para as mais diversas linguagens (incluindo Python).

Para tal, primeiramente abra o terminal/console na Raspberry Pi. Uma vez aberto, clone o repositório do Paho-MQTT para Python utilizando o seguinte comando:

git clone https://github.com/eclipse/paho.mqtt.python

Este processo pode demorar alguns minutos, dependendo da velocidade de sua conexão com a Internet. Uma vez clonado o repositório, deve-se entrar na pasta com o conteúdo do mesmo e instalar o Paho-MQTT para Python. Para tal, utilize os seguintes comandos:

cd paho.mqtt.python

python setup.py install

Após a instalação, está tudo pronto para se começar a desenvolver aplicações em Python que utilizem MQTT.

O projeto consistirá de uma aplicação Python capaz de escrever na tela todas as mensagens recebidas pelo broker em um determinado tópico.É chegada a hora de fazer código! Para isto, salve o seguinte código editando com **nano MQTT.py** em alguma pasta conhecida:

```
import paho.mqtt.client as mqtt
import sys

#definicoes:
Broker = "mqtt.eclipse.org" #"test.mosquitto.org"
PortaBroker = 1883
KeepAliveBroker = 60
TopicoSubscribe = "sanusbmqtt" #dica: troque o nome do topico por algo "unico
    ",
                                #Dessa maneira, ninguem ira saber seu topico
                                    de
                                #subscribe e interferir em seus testes

#Callback -conexao ao broker realizada
def on_connect(client, userdata, flags, rc):
    print("[STATUS] Conectado ao Broker. Resultado de conexao: "+str(rc))

    #faz subscribe automatico no topico
    client.subscribe(TopicoSubscribe)

#Callback -mensagem recebida do broker
def on_message(client, userdata, msg):
  MensagemRecebida = str(msg.payload)

  print("[MSG RECEBIDA] Topico: "+msg.topic+" / Mensagem: "+MensagemRecebida)


#programa principal:
try:
       print("[STATUS] Inicializando MQTT...")
       #inicializa MQTT:
       client = mqtt.Client()
```

```
        client.on_connect = on_connect
        client.on_message = on_message

        client.connect(Broker, PortaBroker, KeepAliveBroker)
        client.loop_forever()
except KeyboardInterrupt:
        print "\nCtrl+C pressionado, encerrando aplicacao e saindo..."
        sys.exit(0)
```

Para executar no terminal, vá a pasta onde o arquivo **MQTT.py** está salvo e utilize o seguinte comando:

python MQTT.py

A aplicação será iniciada e todas as mensagens recebidas no tópico **sanusbmqtt** irão aparecer na tela dos clientes inscritos.

Para testar, utilize qualquer cliente MQTT como, por exemplo, o para PC em ou o disponível para Android. Depois de instalar o cliente MQTT, conecte-se ao servidor broker mqtt.eclipse.org e publique no tópico sanusbmqtt uma mensagem qualquer. A mensagem irá aparecer na tela do terminal do Raspberry Pi, com ilustrados nas Figuras 20.3,20.4 e20.5

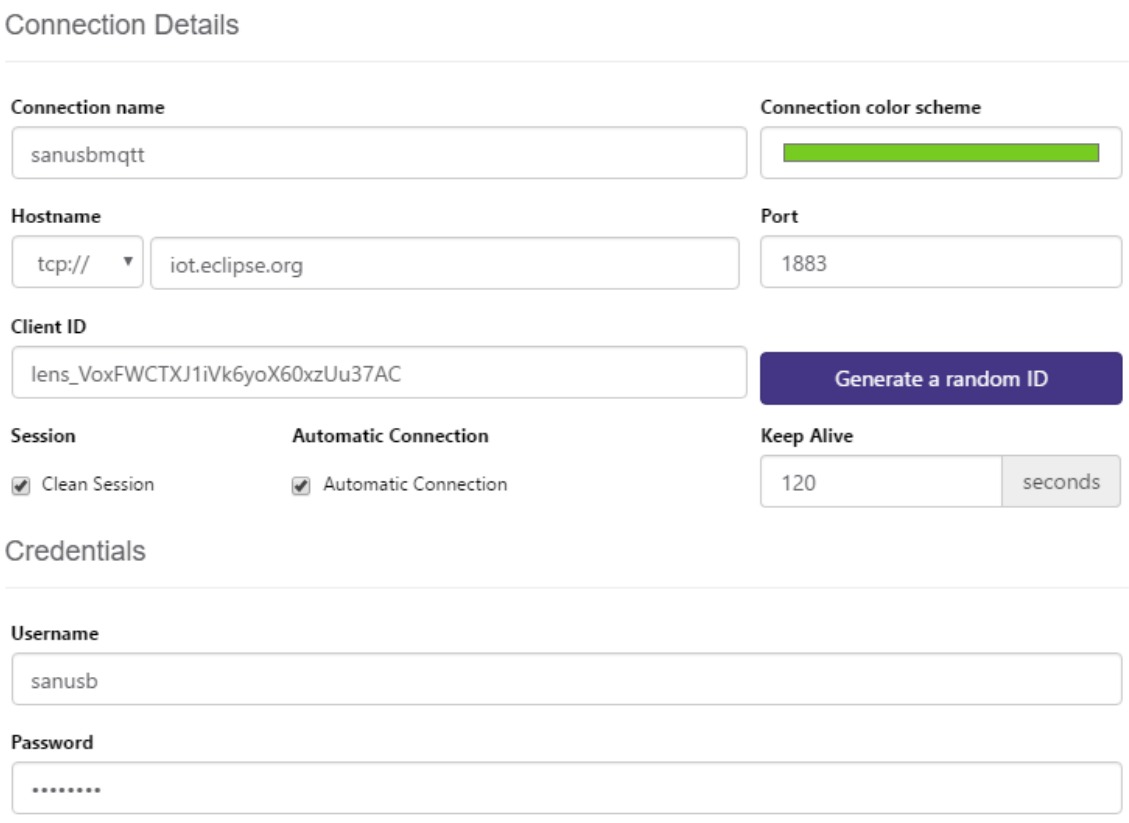


Figura 20.3: Detalhes de conexão

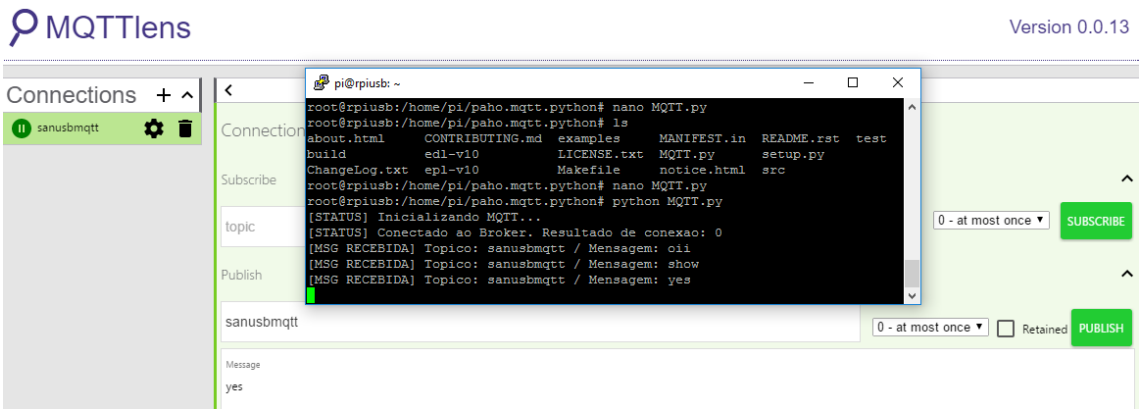


Figura 20.4: Mensagem no terminal do RPI

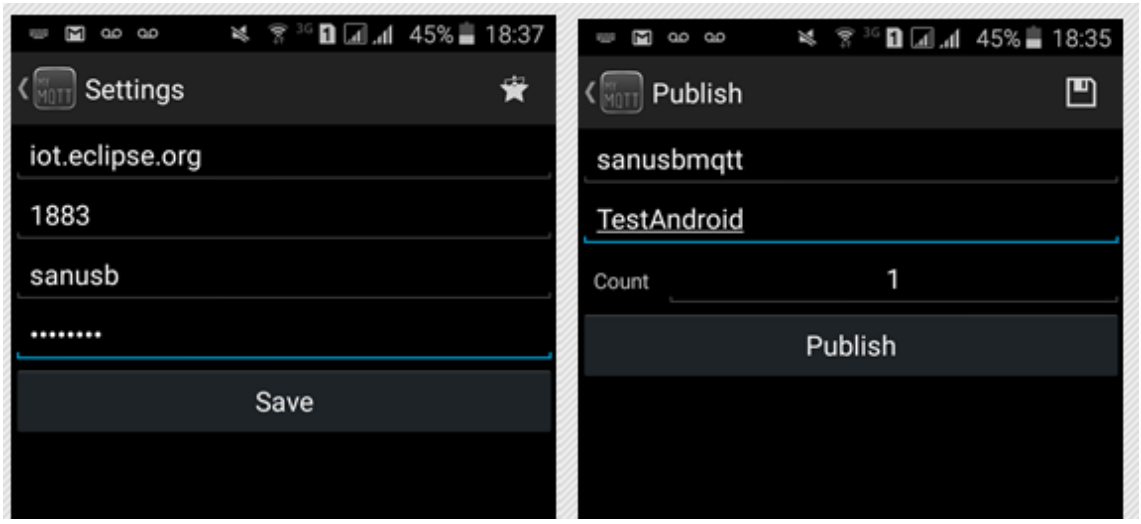


Figura 20.5: Configurações MQTTlens

Um Exemplo para publicar em um tópico:

```
#!/usr/bin/env python

import paho.mqtt.client as mqtt

# This is the Publisher
client = mqtt.Client()
client.connect("mqtt.eclipse.org",1883,60) # "test.mosquitto.org"
\textbf{client.publish("sanusbmqtt", "on");}
client.disconnect();
```

Outro exemplo mostrado abaixo é para comandar o acionamento de um Led no pino físico 7 (GPIO 4). Para isso recomendável instalar a biblioteca gpiozero (**sudo apt-get install python3-gpiozero python-gpiozero**)

```
#!/usr/bin/env python
```

```
import paho.mqtt.client as mqtt
from gpiozero import LED

led = LED(4)

def on_connect(client, userdata, flags, rc):
  print("Connected with result code "+str(rc))
  client.subscribe("sanusbmqtt")

def on_message(client, userdata, msg):
  if (msg.payload == "on"):
    print("on!")
    led.on()
  elif (msg.payload == "off"):
    print("off!")
    led.off()

# client.disconnect()

client = mqtt.Client()
client.connect("mqtt.eclipse.org",1883,60) # "test.mosquitto.org"

client.on_connect = on_connect
client.on_message = on_message

client.loop_forever()
```

20.2 Teste de Subscribe/Publish usando o Python

Abra o terminal e instale o Mosquitto com o seguinte comando:

sudo apt-get install mosquitto mosquitto-clients

Abra outro terminal para ter dois abertos um do lado do outro, o objetivo do teste é abrir um tópico se inscrevendo nele, para depois publicar neste tópico usando o segundo terminal aberto e verificar se a mensagem foi enviada corretamente. A seguir o comando subscribe:

mosquitto_sub -h mqtt.eclipse.org -t test

test é o tópico de acordo com a tag -t e mqtt.eclipse.org é o host do servidor MQTT com a tag -h. Após a execução não há saída pois o inscrito fica aguardando mensagens que foram publicadas para mostrar. Agora no outro terminal, insira o seguinte comando, sendo -m é a tag de mensagem:

mosquitto_pub -h mqtt.eclipse.org -t test -m "hello world"

Com esse comando, no terminal em que foi realizado o subscribe, deve está exibindo a mensagem Hello World. Usuário e senha: Para implementar um sistema de contas o Mosquitto oferece um comando simples para a configuração.

sudo mosquitto_passwd -c /etc/mosquitto/passwd user

user é o nome que você quiser para se referir ao usuário a ser configurada a senha. Agora temos que configurar um arquivo para implementar a senha criada em todas as conexões. Segue:

sudo nano /etc/mosquitto/conf.d/default.conf

O arquivo deve estar vazio, adicione os trechos a seguir nele:

allow_anonymous false password_file /etc/mosquitto/passwd

Agora basta dá um restart e a implementação da senha para o usuário está concluída.

sudo systemctl restart mosquitto

Para testar basta realizar o teste anterior, mas com ou sem a tag -u que representa o usuário e com ou sem a tag que representa a senha -P

mosquitto_pub -h mqtt.eclipse.org -t test -m "hello world"

A saída esperada é:

Connection Refused: not authorised.

Error: The connection was refused.

Isso aconteceu porque não foi especificado nem o usuário e nem a senha Agora refaça o teste com o usuário e senha criados como ilustrado na Figura 20.6:

mosquitto_sub -h mqtt.eclipse.org -t test -u "user"-P "senha"

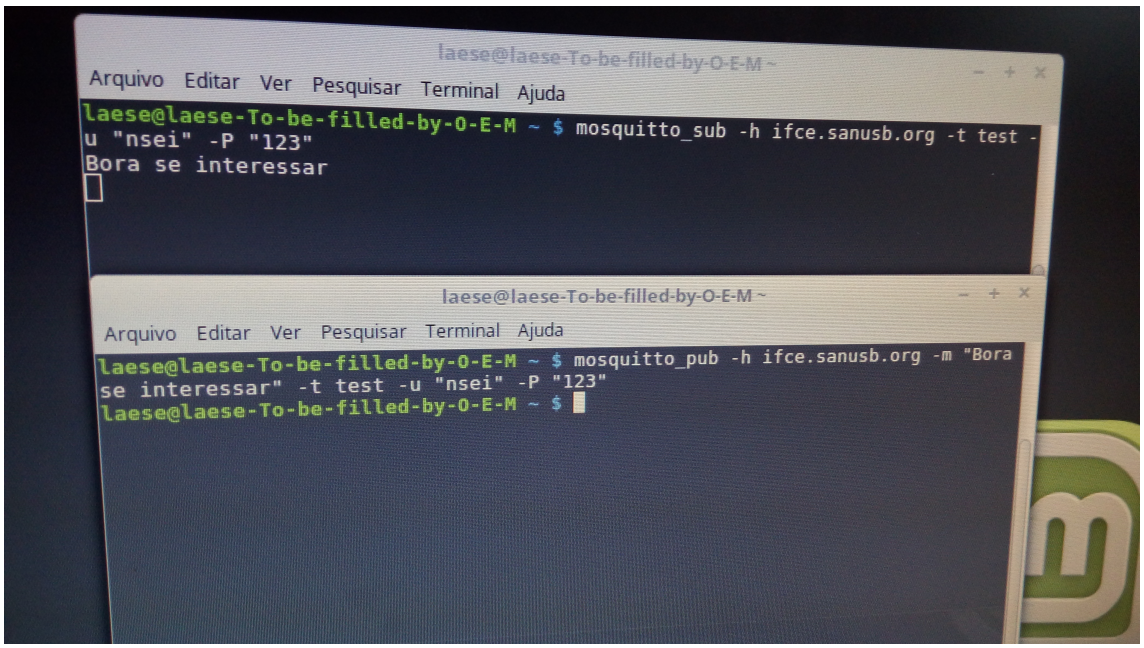


Figura 20.6: Teste MQTT Shell

Ele vai funcionar tanto para a comunicação entre terminais como mostrado acima, como entre máquinas diferentes.

20.3 Instalando o Mosquitto servidor Broker no Raspbian

Para instalar o Mosquitto Broker, digite os seguintes comandos:

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
```

É necessário digitar Y e pressionar Enter para confirmar a instalação. Para fazer o Mosquitto iniciar automaticamente na inicialização do Raspbian, digite:

```
sudo systemctl enable mosquitto.service
```

Para saber a versão instalada do mosquitto e o IP do servidor broker Raspberry digite, respectivamente:

```
mosquitto -v
hostname -I
```

Após instalar o MQTT Broker no Raspberry, é recomendável instalar um MQTT Cllient para testar a instalação do Broker e publicar mensagens de teste.

```
sudo apt-get install mosquitto-clients
```

Agora execute o Mosquitto em segundo plano como um daemon:

```
mosquitto -d
```

Testando a comunicação

Para se inscrever em um tópico do MQTT com o Mosquitto Client, abra uma janela do terminal e digite o comando:

```
mosquitto_sub -d -t topicsanusb
```

Agora você está inscrito em um tópico chamado topicsanusb.

Para publicar uma mensagem de teste no **topicsanusb** , abra uma segunda janela do terminal e publique uma mensagem com o comando:

```
mosquitto_pub -d -t testTopic -m "Olá, mundo!"
```

O resultado da comunicação é ilustrado na Figura 20.7.

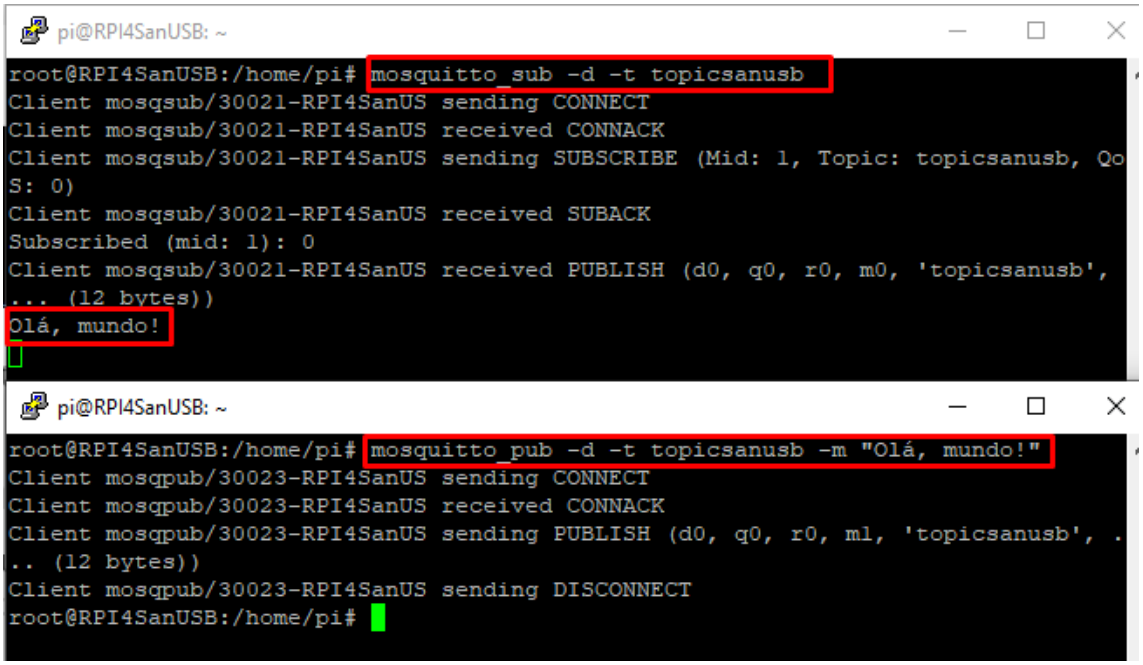


Figura 20.7: Tela de resultados da comunicação com o broker instalado no Rpi

Outra forma de realizar o teste, é escrever como host (-h) o localhost ou o IP encontrado no comando *hostname -I*, como ilustrado na Figura 20.8.

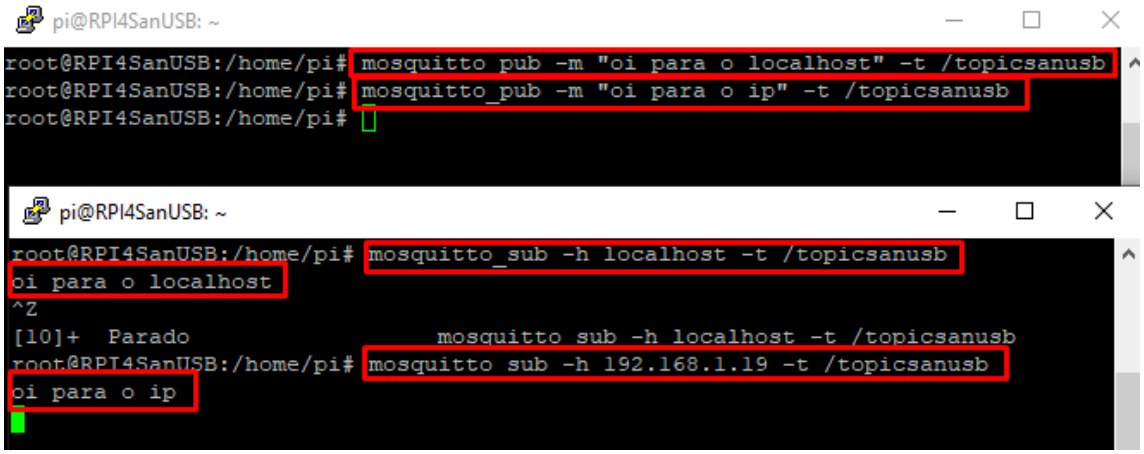


Figura 20.8: Tela de resultados da comunicação com o broker instalado no Rpi

20.4 Acionamento de um pino do Rpi utilizando aplicação IoT *pulling*

O objetivo desta seção é apresentar os resultados da prática acionamento de LED com Raspberry Pi 3 via Web, usando o Raspbian Lite integrado ao servidor SanUSB (`sanusb.org/gpio`) ilustrado na Figura 20.9.

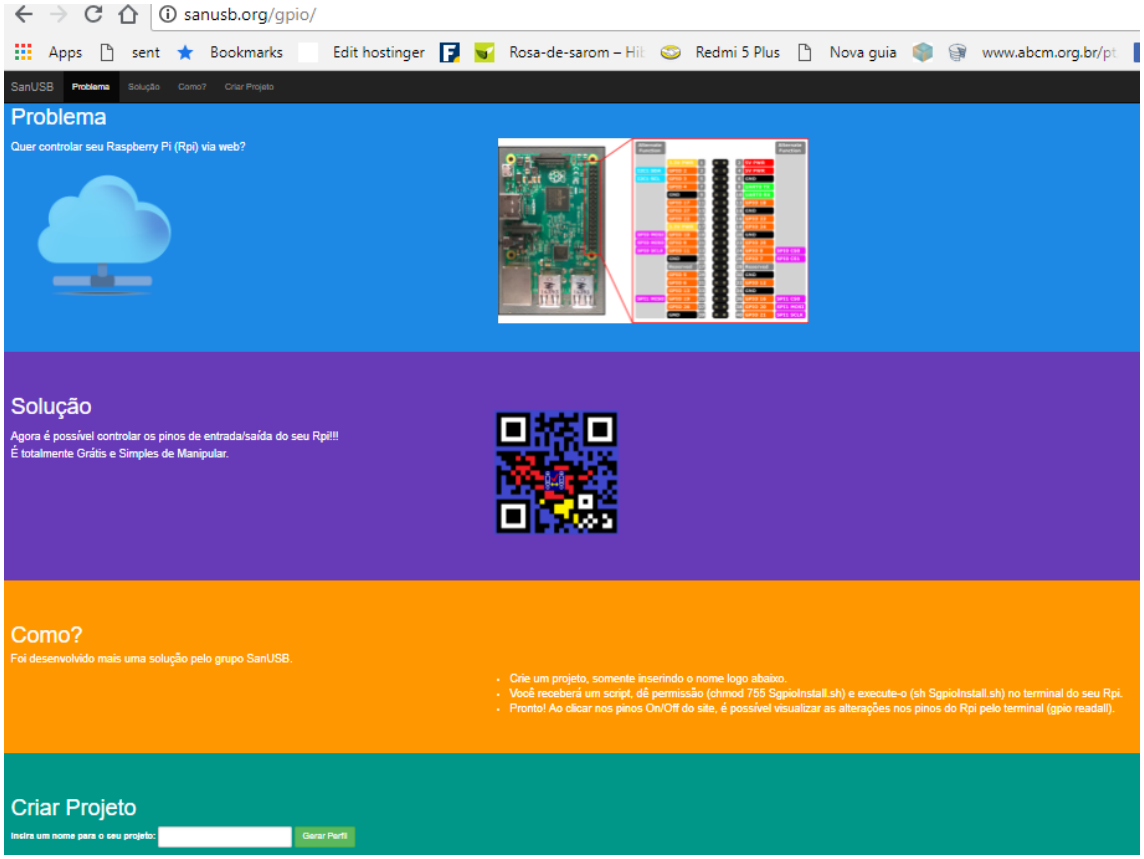


Figura 20.9: Conexão Tela da plataforma Gpio

Para isso, o passo seguinte foi criar um perfil de projeto no no final da página *sanusb.org/gpio*, como é ilustrado na Figura 20.11, no intuito de gerir o controle de acionamento dos pinos do Raspberry Pi via página Web gerada pelo perfil. No exemplo, o perfil gerado foi *laese*.

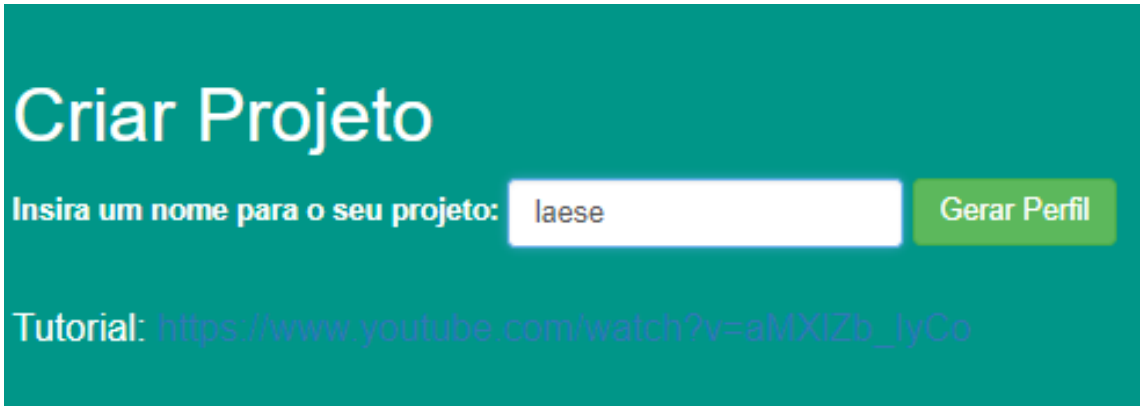


Figura 20.10: Criação do perfil online do usuário

Após a criação do perfil em *sanusb.org/gpio*, como é ilustrado na Figura **??**, é gereado uma página com o Links da página com os botões dos pinos e dos programas Rpi para a aplicação *pulling* e para a aplicação *push* em *bash shell*.

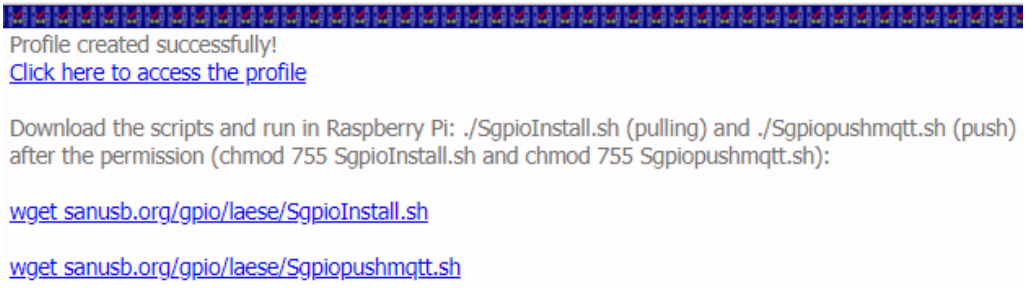


Figura 20.11: Links da página com os botões dos pinos e dos programas Rpi em bash *pulling* e *push*

Tanto para a aplicação *pulling*, quanto para a aplicação *push* é necessário, após baixar os programas, é necessário conceder no terminal, como super-usuário (sudo su), permissão de execução com os comandos *chmod 755 SgpioInstall.sh*, para aplicação *pulling*, e *chmod 755 Sgpiopushmqtt.sh*, para aplicação *push*.

Ao clicar no link do perfil, é gerada uma página com botões que representam cada um dos pinos de I/O de um Rpi, como é ilustrado na Figura 20.12.

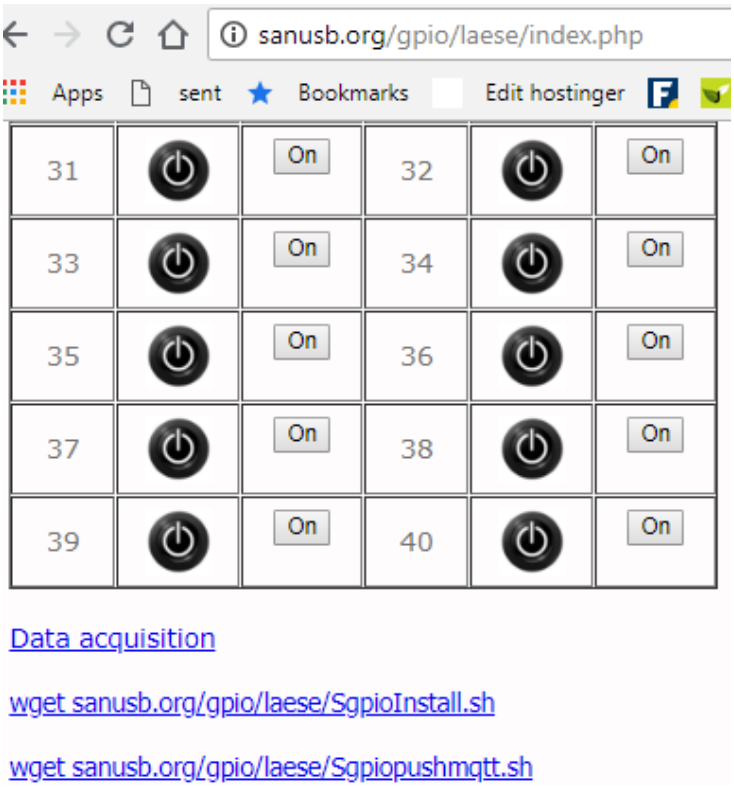


Figura 20.12: Página com botões que representam cada um dos pinos de I/O de um Rpi

No caso *pulling*, ao clicar no botão de um determinado pino físico, é enviado um comando para um arquivo contido no perfil online que é consultado periodicamente pelo script *Sgpiointall.sh* que está operando no Rpi. No caso *push mqtt*, ao clicar no botão é enviado um comando *publish* para o broker mqtt.eclipse.org no tópico com o mesmo nome do perfil do usuário, no caso do exemplo, *laese*, e esse comando é "empurrado"no Rpi que está operando o script *Sgiopushmqtt.sh* com inscrição (*subscribe*) no

mesmo tópico *laese*. O script em shell bash da aplicação *pulling* para o perfil *laese* é mostrado abaixo:

```
#!/bin/bash
# Created site: http://sanusb.org/gpio/laese
# 15/07/18 14:55:16. Tutorial: youtu.be/aMXlZb_IyCo
profile=laese

# install Wpi
  if [ ! -f "/usr/include/wiringPi.h" ]; then
    sudo apt-get install wiringpi
  fi

# install sanusb (sanusb.org)
   if [ ! -f "/usr/share/sanusb/sanusb" ]; then
   cd /home/share
   wget http://sanusb.org/tools/SanUSBrpi.zip
   unzip SanUSBrpi.zip
   cd SanUSBrpi
   sudo dpkg -i sanusb_raspberry.deb
   chmod +x sanusb
   cp sanusb /usr/share/sanusb
   fi

##************************************************************
contini=$(curl -sl sanusb.org/gpio/$profile/view)
sleep 1
echo contini=$contini
##************************************************************

#Start Loop
while :
do
  #Verifica o ndice content
  content=$(curl -sl sanusb.org/gpio/$profile/view)
  #echo content=$content
  sleep 2

if (( "$content" != "$contini" )) ; then
## && (( "$content" != "" ))
     ((e=2))
     cont=$content

      for i in {40..1}
      do
      g=$((e**$i)); if (("$cont" >= "g")); then ((cont=cont-g)); gpio -1
          mode $i out; gpio -1 write $i 1; else gpio -1 write $i 0; fi
      done
```

```
    gpio readall
    contini=$(curl -sl sanusb.org/gpio/$profile/view)
    sleep 1
  fi
done
```

O script em shell bash da aplicação *push* para o perfil *laese* é mostrado abaixo:

```
#!/bin/bash
# Created site: http://sanusb.org/gpio/laese
# 15/07/18 14:55:17. Tutorial: youtu.be/aMXlZb_IyCo
profile=laese

# install Wpi
  if [ ! -f "/usr/include/wiringPi.h" ]; then
    sudo apt-get install wiringpi
  fi

# install mqtt client
   if [ ! -f "/usr/share/doc/libc-ares2/changelog.gz" ]; then
   sudo apt-get install mosquitto-clients
   fi

# install sanusb (sanusb.org)
   if [ ! -f "/usr/share/sanusb/sanusb" ]; then
   cd /home/share
   wget http://sanusb.org/tools/SanUSBrpi.zip
   unzip SanUSBrpi.zip
   cd SanUSBrpi
   sudo dpkg -i sanusb_raspberry.deb
   chmod +x sanusb
   cp sanusb /usr/share/sanusb
   fi

p="backpipe"
pid=0

#Sugerido digitar Control+C para sair eliminando os processos concorrentes
    do mqtt
ctrl_c() { #ps para ver os processos ativos
  printf "\nLimpando..."
  rm $p
  kill $pid # mata o processo mqtt
  if [[ "$?" -eq "0" ]];
  then
   echo "Sucesso";exit 0
```

```
  else
   exit 1
  fi
}

listen(){ #loop que escuta o mqtt
  ([ ! -p "$p" ])
  (mosquitto_sub -h mqtt.eclipse.org -t $profile >$p 2>/dev/null) &

  echo "pid $!"
  echo "host mqtt.eclipse.org"
  echo "topic $profile"

  pid=$!

contini=$(cat $p)
sleep 1

  while : # read line <$p
  do
    content=$(cat $p)
 if [[ "$content" != "$contini" ]]; then
    truncate -s 0 $p
    echo "$content"
    contini=$(cat $p)

  ((e=2))
  cont=$content

  for i in {40..1}
    do
    g=$((e**$i)); if (("$cont" >= "g")); then ((cont=cont-g)); gpio -1 mode
        $i out; gpio -1 write $i 1; else gpio -1 write $i 0; fi
    done

  gpio readall

 fi
    sleep 1
  done
}

trap ctrl_c INT #marca funcao de interrupcao ctrl_c()
listen #executa mqtt_sub e escuta
```

Após colocar os scripts no Rpi, foi realizado a montagem de um circuito básico de acionamento IoT, que pode ser vista na Figura 20.14, onde mostra a conexão de um LED nos pinos 40 e GND. Nessa prática, o

LED está sendo comandado pela página online do perfil *laese* ilustrada na Figura 20.13.

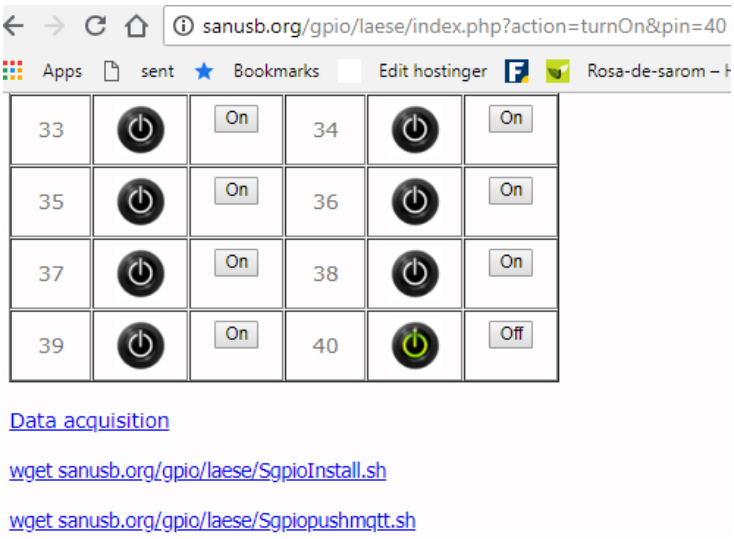


Figura 20.13: Página online de acionamento dos pinos do Rpi

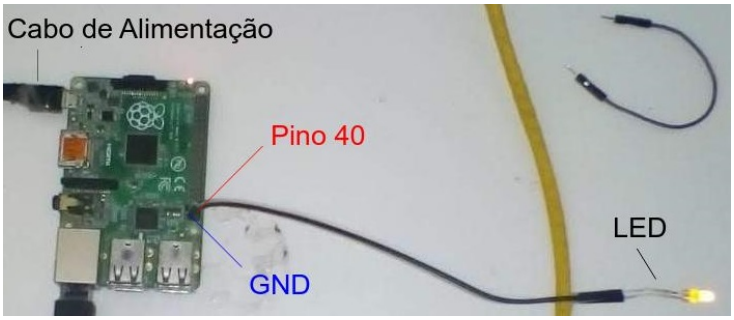


Figura 20.14: Circuito báasico de acionamento IoT de um LED do Rpi

Como exemplo, o vídeo `https://youtu.be/NaAf3I0Ej6Y` mostra uma aplicação internet das coisas utilizando um notebook ou um smartphone possa ser possível controlar o acionamento de eletrodomésticos como, por exemplo, ligar um ventilador. Nesse caso, é utilizado um Raspberry Pi, que é um sistema embarcado que pode acessado pela internet. Para utilizar o ambiente em `sanusb.org/gpio`, é necessário criar Inicialmente um perfil. Após a criação de um perfil, é gerado um link para acessar uma página com vários botões e cada botão corresponde a um pino do dispositivo IoT Raspberry Pi. A Figura 20.15 ilustra o circuito de acionamento IoT do ventilador.

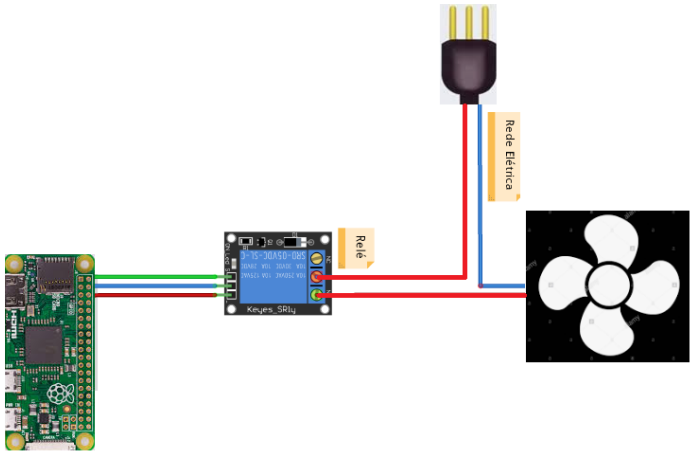


Figura 20.15: Circuito de acionamento IoT do ventilador

20.5 Monitor MQTT

No intuito de testar qualquer *broker* e qualquer comunicação via MQTT, com a possibilidade de se inscrever e publicar em um mesmo tópico, foi desenvolvido o Monitor Web MQTT em `sanusb.org/mqtt`, como ilustrado na Figura 20.16.

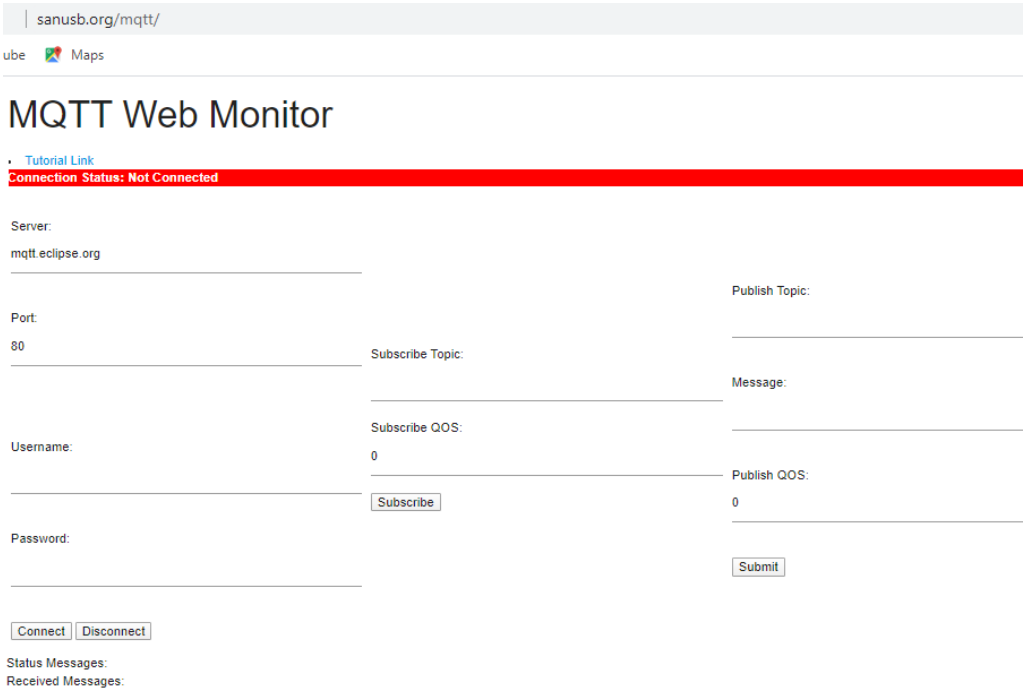


Figura 20.16: Monitor WEB MQTT

A porta padrão MQTT é 1883 (para terminais e firmware de microcontroladores), porém as portas 80, 8000 e 8080 são portas utilizadas por ambientes *websocket*, como o monitor proposto, que empacotam e também se comunicam com inscrições em porta padrão MQTT (1883). Um tutorial

de utilização do Monitor MQTT pode ser acessado no link https://youtu.be/0Wj-leEoyZQ.

Os *Brokers* listados abaixo são de acesso livre e gratuito, sem a necessidade de inscrição anterior, e podem ser utilizados por quaisquer aplicações MQTT, como também pelo Monitor Web MQTT proposto:

mqtt.eclipse.org (porta 80);

broker.hivemq.com (porta 8000);

broker.mqttdashboard.com (porta 8000);

test.mosquitto.org (porta 8080).

20.6 Minificando páginas Web para aplicações IoT utilizando Raspberry Pi

Na verdade, a minificação é mais que uma compressão (.zip, .rar, etc.), pois remove do código Web (HTML, CSS, JavaScript, etc.) tudo que não é obrigatório para que o arquivo seja interpretado e executado e, ao contrário das técnicas de compressão convencionais, os arquivos minificados não precisam retornar ao estado anterior, mais sim serem executados corretamente depois da descompressão. Além disso, a linguagem Web é uma linguagem interpretada, ou seja, é lida linha por linha. Logo, quanto menos linhas de código tiver a página, mais rápido será o tempo de execução.

Dessa forma, quando um usuário faz uma requisição para uma página Web, a versão minificada é enviada ao invés da versão completa e com isso é possível obter uma resposta mais rápida e a um custo menor da banda de internet, ou seja, ter menos tráfego de banda sem sacrificar a funcionalidade do código.

Nesse sentido, o gulp é um automatizador de tarefas js para minificar, organizar e controlar arquivos HTML, CSS e JavaScript. Esta seção mostra como compactar páginas Web em formato zip para aplicações IoT utilizando Raspberry Pi. Para iniciar é necessário baixar o script de instalação, dentro do Rpi, por exemplo em */home/pi*, digitando a linha de comando:

```
sudo wget http://sanusb.org/arquivos/gzipusb.sh
```

Após baixar o arquivo, é neessário dar permissão de execução e executar:

```
sudo chmod 777 gzipusb.sh
sudo ./gzipusb.sh
```

O script em shell gzipusb.sh baixa uma pasta configuradda gzipusb.zip com bibliotecas para a minificação (compressão) de arquivos, baixa e instala o Node.js (ambiente de execução Javascript *server-side*), o npm (*Node Package Manager*) e, depois, baixa e instala o *gulp* que é um automatizador de tarefas js para minificar (compactar) os arquivos Web, como descrito abaixo:

```
#!/bin/bash
# Link: http://sanusb.org/arquivos/gzipusb.sh
```

```
# Tutorial: https://youtu.be/HwZVmrcntKo

# Instalar o node (https://nodejs.org/dist/v9.9.0/). Outras verses em
    https://nodejs.org/dist/:
  apt install unzip
  cd /home/pi
    wget http://sanusb.org/tools/gzipusb.zip
  unzip gzipusb.zip
  cd gzipusb
  curl -o node-v9.9.0-linux-armv6l.tar.gz https://nodejs.org/dist/v9
      .9.0/node-v9.9.0-linux-armv6l.tar.gz
  tar -xzf node-v9.9.0-linux-armv6l.tar.gz
  sudo cp -r node-v9.9.0-linux-armv6l/* /usr/local/
  node -v
  npm -v
  npm i

# Instalar gulp globalmente dentro da pasta. Para compilar digite gulp:
  npm install -g gulp
  gulp -v

# Para compilar digite gulp, eos arquivos .html da pasta entrada estarão
     compactados na pasta saida

# install sanusb (sanusb.org)
   if [ ! -f "/usr/share/sanusb/sanusb" ]; then
     cd /home/share
     wget http://sanusb.org/tools/SanUSBrpi.zip
   unzip SanUSBrpi.zip
   cd SanUSBrpi
   sudo dpkg -i sanusb_raspberry.deb
   chmod +x sanusb
   cp sanusb /usr/share/sanusb
   fi
```

No vídeo https://www.youtube.com/embed/HwZVmrcntKo é mostrado um tutorial para a minificação (compressão) de páginas html com gulp para aplicações IoT utilizando Raspberry Pi e no vídeo https://www.youtube.com/embed/oc0Ox725sGk é mostrado um tutorial para a minificação da aplicação IoT Monitor Web MQTT (Figura 20.16), embarcada em um microcontrolador ESP32.

21. Referências

Broadcom (2015) “BCM2835 ARM Peripherals” Abril.https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf

Gordon Projects (2013) “WiringPi”, Maio. https://projects.drogon.net/raspberry-pi/wiringpi/

Grupo SanUSB (2015) “Ferramenta SanUSB” Abril. http://sanusb.org

Jucá, S. C. S., Carvalho, P. C. M. and Brito, F. T. (2009) “SanUSB: software educacional para o ensino da tecnologia de microcontroladores”, Ciências Cognição, Rio de Janeiro, v. 14, p. 134-144.

Raspberry (2015) Abril. http://www.raspberrypi.org/.

Sampaio, F., Jucá, S. C. S., and Pereira, R. I. S. (2014) “Aplicação WEB para Monitoramento Online de Microgeração Elétrica via Modem WiFi utilizando Fontes Renováveis de Energia”, V EATI – Encontro Anual de Tecnologia da Informação.

SanUSB (2015) “Gravando PIC online via porta USB de um Raspberry Pi” Abril. https://www.youtube.com/watch?v=S30wVi9RWEs,

Viva o Linux (2010) “Gravação de microcontroladores PIC via porta USB”, Agosto. http://www.vivaolinux.com.br/.