Capítulo 4

Desenvolvendo Sensores de Vídeo para a Internet das Coisas com o Raspberry Pi

Daniel G. Costa[1]

[1]Universidade Estadual de Feira de Santana (DTEC-UEFS)

danielgcosta@uefs.br

Abstract

The increasing interest for Internet of Things (IoT) technologies has brought a lot of attention to microelectronics and sensors development. With the availability of affordable embedded platforms for countless applications, it is possible to develop low-cost programmable sensors to provide different types of data, specially when cameras are employed. This Chapter covers the fundamentals of visual sensing and presents several technical details when using the Raspberry Pi platform for development of visual sensors. Basic configurations of the Raspberry Pi along with the most popular programs to take image snapshots and to make video streams will be presented. Moreover, the development of Python programs using the Raspberry Pi Camera will be discussed, as well as real-time transmissions of visual data over wireless networks. At the end of this Chapter, readers should know the fundamentals for the creation of highly programmable visual sensors with Raspberry Pi.

Resumo

O crescente interesse por tecnologias ligadas à Internet das Coisas (Internet of Things - IoT) tem trazido muita atenção para o desenvolvimento de sensores e dispositivos microeletrônicos. Com a disponibilidade de plataformas acessíveis para desenvolvimento embarcado, torna-se possível criar soluções de baixo custo para um número incontável de aplicações, especialmente quando câmeras são utilizadas. Este Capítulo aborda os fundamentos do monitoramento por sensores visuais, apresentado diversos detalhes técnicos relacionados à utilização do Raspberry Pi como um sensor visual. Para tanto, configurações básicas do Raspberry Pi e detalhes da utilização de ferramentas para captura de imagens e vídeos serão apresentados. Mais ainda, será discutido o desenvolvimento de programas Python explorando a câmera do Raspberry Pi, além de questões relacionadas a transmissão em tempo real dos dados visuais capturados. Assim, espera-se que ao final deste Capítulo os leitores conheçam os fundamentos para a criação de sensores visuais altamente programáveis.

4.1. Introdução

O desenvolvimento de tecnologias eficientes de sensoriamento e o constante barateamento de dispositivos eletrônicos embarcados vêm permitindo a inserção crescente de milhões de nós inteligentes às redes de comunicação digitais, com previsões de dezenas de bilhões de dispositivos conectados até o final desta década. De fato, já existem mais dispositivos que pessoas conectadas à Internet, permitindo o surgimento de uma nova era para troca e processamento de informações [Atzori et al. 2010, Lin et al. 2017].

De maneira geral, a Internet das Coisas (*Internet of Things - IoT*) é uma das principais fronteiras para o desenvolvimento das redes de comunicação, com potencial para grandes revoluções sociais e econômicas. Nesse novo cenário, além das inúmeras redes privadas que devem ser criadas, governos e organizações irão implantar diversos tipos de dispositivos que poderão ser acessados publicamente na Internet, como sensores pluviométricos, câmeras de segurança, sensores de temperatura, lâmpadas inteligentes, entre muitos outros [Costa et al. 2017]. Esse grande conjunto de informações, gerado e processado diariamente e de forma ininterrupta, irá revolucionar a forma como interagimos com as pessoas, com as cidades e com os dispositivos que nos cercam [Perera et al. 2014, Costa et al. 2018, Andrade et al. 2017]

Um dos elementos centrais do universo IoT é o "sensor", um dispositivo eletrônico com funções bem definidas e que estará frequentemente conectado a uma rede. Em linhas gerais, um sensor é um dispositivo eletrônico desenvolvido para coletar informações do ambiente monitorado, podendo as informações coletadas serem de tipos diferentes de acordo com as unidades de monitoramento utilizadas para a construção do sensor e de sua programação. Os sensores podem então ser interligados de diferentes formas e à diferentes redes, dependendo do seu objetivo no contexto da aplicação desenvolvida [Costa et al. 2015].

Na última década, Redes de Sensores Sem Fio (RSSF) constituíram-se em um dos principais tópicos de pesquisa na área de redes de computadores e comunicação digital. As RSSF são redes *ad hoc* tipicamente compostas por muitos dispositivos operados por bateria e de baixo custo (sensores e/ou atuadores), cada um com recursos de processamento e comunicação sem fio que permitem que operem de forma cooperativa e auto organizável. Como alguns ou todos esses dispositivos podem possuir capacidade para um determinado tipo de monitoramento, as RSSF podem ser utilizadas para uma grande variedade de aplicações inovadoras, permitindo monitoramento em regiões sem infraestrutura, perigosas para acesso humano ou de difícil acesso, revolucionando a forma como dados são obtidos de um ambiente monitorado [Kuo et al. 2018]. Entre as aplicações de monitoramento das RSSF, podemos destacar: controle industrial, monitoramento ambiental e climático, operações de resgate, automação residencial, detecção de poluição, planejamento militar, entre muitas outras [Costa et al. 2017].

Quando sensores são equipados com câmeras de baixo custo e reduzido consumo de energia, dados visuais podem ser obtidos do ambiente monitorado, permitindo um novo escopo de aplicações quando comparado com RSSF tradicionais compostas por sensores de grandezas escalares (temperatura, pressão, luminosidade, umidade, etc). Essas novas Redes de Sensores Visuais Sem Fio (RSVSF) enriquecem a percepção do ambiente monitorado com imagens e/ou vídeos, aperfeiçoando aplicações de monitoramento tradi-

cionais ou mesmo permitindo o surgimento de uma nova gama de aplicações, como em controle de tráfego, localização e rastreamento, segurança pública, previsão climática e monitoramento de desastres, apenas para citar alguns exemplos [Leone et al. 2017].

Independente da forma como serão interligados, da programação relacionada às funções de monitoramento e da quantidade empregada, os sensores serão elementos centrais do universo IoT. E, portanto, disponibilizar formas eficientes e flexíveis de criar sensores torna-se bastante desejável. Nesse cenário, a utilização de plataformas para desenvolvimento embarcado permite que uma extensa gama de sensores possa ser criada de forma facilitada, acelerando atividades de experimentação e prototipação na área de IoT. Este Capítulo, portanto, está destinado a fornecer o *background* necessário para que sensores visuais (também referenciados comumente como sensores de vídeo) sejam facilmente criados na plataforma Raspberry Pi.

De fato, com o advento de plataformas embarcadas para o desenvolvimento de dispositivos eletrônicos genéricos, como, por exemplo, o Raspberry Pi, o BeagleBone e o Intel Edison, é possível criar dispositivos multimídia de monitoramento de forma facilitada [Kruger and Hancke 2014]. Assim, pode-se desenvolver aplicações de monitoramento visual de baixo custo para a Internet das Coisas. Contudo, há diversos detalhes de configuração e operação desses dispositivos que devem ser considerados antes de se escolher qual é a plataforma mais adequada para determinado projeto [Nikhade 2015, Patil et al. 2017]. A Tabela 4.1 apresenta alguns exemplos de plataformas disponíveis para desenvolvimento embarcado que podem ser consideradas para a construção de sensores visuais IoT. Essas plataformas possuem diferentes recursos e preços que podem guiar a escolha da solução mais apropriada para determinado projeto.

Tabela 4.1. Algumas plataformas para construção de sensores visuais IoT.

Plataforma	Lançamento	CPU	RAM	Website
Raspberry Pi 2	2015	900 MHz	1 GB	http://raspberrypi.org
BeagleBone Black	2013	1 GHz	512 MB	http://beagleboard.org
Orange Pi One	2016	1.2 GHz	512 MB	http://www.orangepi.org
CubieBoard 4	2015	2 GHz	2 GB	http://cubieboard.org

Com o crescente aumento do interesse por dispositivos embarcados microprocessados em uma única placa (*single-board computers*), o mercado foi inundando por diferentes modelos de diferentes fabricantes, com significativas variações na capacidade de processamento e armazenamento, na quantidade e tipo de interfaces de entrada/saída e no custo final. Nesse rico cenário, a plataforma Raspberry Pi vem se destacando por sua versatilidade, baixo custo e ampla comunidade de suporte, que coloca essa plataforma como um potencial candidato para o desenvolvimento de dispositivos IoT. Tendo sido inicialmente lançado em 2012 com propósitos educacionais, o Raspberry Pi evoluiu e continua evoluindo, com diferentes modelos sendo lançados e com extensos recursos de software disponíveis. No cenário atual, a plataforma Raspberry Pi oferece amplo suporte para o desenvolvimento de sensores para aplicações da Internet das Coisas [Gupta et al. 2015, Sandeep et al. 2015, Patchava et al. 2015].

A Tabela 4.2 apresenta os principais modelos do Raspberry Pi que foram lançados até meados de 2018. Todos os modelos do Raspberry Pi (com exceção do modelo original,

com 26 pinos) possuem 40 pinos de GPIO (*General Purpose Input Output)* e realizam armazenamento por cartão microSD (os modelos do Raspberry Pi não possuem unidade de armazenamento integrada).

Tabela 4.2. Principais modelos lançados do Raspberry Pi.

Placa	Lançamento	CPU	RAM	Rede	Interfaces E/S
Raspberry Pi Model B	2012	700 MHz	512 MB	Ethernet	HDMI, 2 USB
Raspberry Pi Model A+	2014	700 MHz	256 MB	-	HDMI, 1 USB
Raspberry Pi Model B+	2014	700 MHz	512 MB	Ethernet	HDMI, 4 USB
Raspberry Pi 2 Model B	2015	900 MHz	1 GB	Ethernet	HDMI, 4 USB
Raspberry Pi Zero	2015	1 GHz	512 MB	-	HDMI mini, 1 micro USB
Raspberry Pi 3 Model B	2016	1.2 GHz	1GB	Wi-Fi, Bluetooth, Ethernet	HDMI, 4 USB
Raspberry Pi Zero W	2017	1 GHz	512 MB	Wi-Fi, Bluetooth	HDMI mini, 1 micro USB
Raspberry Pi 3 Model B+	2018	1.4 GHz	1GB	Wi-Fi, Bluetooth, Ethernet	HDMI, 4 USB

A Figura 4.1 apresenta três modelos diferentes do Raspberry Pi. Embora todos os três modelos possam ser utilizados para construir um sensor visual, diferenças na capacidade de processamento e armazenamento podem diretamente interferir na qualidade efetiva do sensor desenvolvido.

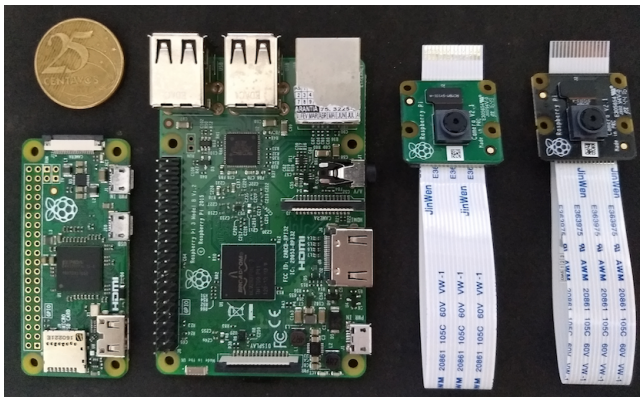

Figura 4.1. Da esquerda para a direita: Raspberry Pi Zero, Raspberry Pi 3 Model B, Pi Camera v2 e Pi Camera v2 NoIR.

4.2. Iniciando no Raspberry Pi

O Raspberry Pi é um pequeno computador construído em uma única placa de circuito, tão pequeno que cabe na palma da mão. Além do tamanho reduzido, o que permite sua utilização prática em diversos cenários da Internet das Coisas, o Raspberry Pi consome pouca energia e possui baixo custo de aquisição, tornando-o um candidato promissor como plataforma versátil para a construção de sensores visuais. De fato, existem muitos detalhes relacionados ao Raspberry Pi e muitos projetos estão sendo criados utilizando essa plataforma, nas mais diversas áreas. Contudo, será dada atenção aos detalhes relacionados a transformar o Raspberry Pi em um sensor de vídeo (que neste Capítulo será considerado como um sensor capaz de transmitir imagens e/ou *streams* de vídeo).

O modelo de referência adotado neste Capítulo é o Raspberry Pi 2 model B (rasp2b), embora os detalhes apresentados também sejam válidos para outros modelos

lançados a partir deste. Para alimentação, recomenda-se utilizar uma fonte fornecendo entre 4.8 V e 5.2 V e 2.5A, sobretudo quando diversos periféricos estejam conectados ao rasp2b. Além do Raspberry, serão necessários neste Capítulo um cartão microSD (classe 10) com pelo menos 4 GB de capacidade de armazenamento e uma câmera HDMI oficial. Periféricos como monitor, teclado e mouse podem ser utilizados de acordo com o nível de conhecimento do leitor e as funcionalidade desejadas, uma vez que o rasp2b pode ser configurado diretamente a partir de um computador auxiliar.

4.2.1. Escolhendo o sistema operacional

Sendo um computador, o Raspberry precisa de um sistema operacional para oferecer serviços aos usuários e coordenar a interação entre processador, memória e unidades de entrada e saída. Para tanto, há diversos sistemas operacionais disponíveis para serem utilizados com o Raspberry Pi, utilizando diferentes sistemas "tradicionais" como referência (por exemplo, Linux, Windows e Android). Entre os sistemas disponíveis, o Raspbian é um sistema operacional baseado no Linux Debian, sendo o sistema operacional oficialmente suportado pela Raspberry Pi Foundation, para todos os modelos Raspberry Pi. Este Capítulo considera o Raspbian como sistema operacional oficial, mais especificamente a versão Raspbian Stretch (lançado em 2017).

Há diversos tutoriais gratuitos online sobre como realizar a instalação do Raspbian. Adicionalmente, a Raspberry Pi Foundation suporta o NOOBS (New Out Of the Box Software), um instalador de sistemas operacionais (algumas opções de SO são disponibilizadas) para facilitar o uso do Raspberry por iniciantes. De qualquer forma, será considerado neste Capítulo a utilização do Raspbian como sistema operacional padrão, sendo o Raspbian Stretch completo (*Desktop*) a versão utilizada como referência para os experimentos realizados.

4.2.2. Configurações iniciais

Há algumas configurações que devem ser feitas logo após a primeira execução do sistema operacional. Essas configurações são necessárias para a correta execução dos exemplos deste Capítulo. Embora a indicação da instalação seja feita quando necessário, nas seções apropriadas, podem-se instalar todos os pacotes necessários logo de início. Assim, a sequência de comandos a seguir instala todos os programas, bibliotecas e dependências necessárias neste Capítulo (permissão de "root" é exigida para execução desses comandos):

```
$ apt-get update
$ apt-get upgrade
$ apt-get install gpac vlc mplayer
$ apt-get install hostapd dnsmasq apache2
```

4.2.3. Comandos básicos

Como o Raspbian é baseado no sistema Debian, a maioria dos comandos disponíveis para o *bash* do Linux estão também disponíveis no Raspberry. A interface de interação por linha de comando na versão simplificada do Raspbian (*Lite*) é, por padrão, o *bash*. Já para a versão completa (*Desktop*), o terminal de linha de comando pode ser facilmente acessado.

De qualquer forma, a utilização de comandos diretamente no *bash* garante mais flexibilidade de configuração, além de permitir acesso e controle remoto do Raspberry através do protocolo SSH. Portanto, conhecer os comandos disponíveis pode facilitar sobremaneira a utilização do Raspberry como sensor de vídeo.

A Tabela 4.3 apresenta alguns dos principais comandos de configuração utilizados na construção de sensores de vídeo com o Raspberry Pi. Para facilitar a organização de tais comandos, a Tabela 4.3 define a seguinte classificação: Sistema e Redes.

Tabela 4.3. Alguns comandos do *bash* para a construção de sensores visuais.

Comando	Tipo	Descrição
apt-get	Sistema	Baixa e instala novos programas e bibliotecas
chmod	Sistema	Altera as permissões de acesso e execução de arquivos
date	Sistema	Exibe informações de data e hora
df	Sistema	Apresenta informações de uso do cartão microSD
hostname	Sistema	Exibe o nome definido para o sistema
free	Sistema	Apresenta o uso da memória do sistema
ip	Rede	Usado para exibir e alterar informações das interfaces de comunicação em rede (Ethernet e Wi-Fi) e da tabela de rotas
iwconfig	Rede	Usado para exibir e alterar informações das interfaces de comunicação em rede sem fio
iwgetid	Rede	Exibe o id da rede sem fio que está oferecendo conectividade ao Raspberry
iwlist	Rede	Lista as redes Wi-Fi atualmente disponíveis
lsusb	Sistema	Lista os dispositivos USB conectados ao Raspberry
netcat (nc)	Rede	Comando versátil utilizado para abrir conexões em rede
nslookup	Rede	Comando utilizado para interação com o serviço DNS
ping	Rede	Utilizado para teste de conectividade (mensagens ICMP)
poweroff	Sistema	Desliga imediatamente o sistema
reboot	Sistema	Reinicia imediatamente o sistema
scp	Rede	Comando utilizado para realizar transferências seguras de arquivos pela rede (protocolo SSH)
ssh	Rede	Permite o acesso seguro via terminal à um dispositivo remoto (protocolo SSH)
tar	Sistema	Programa utilizado para agregar e compactar arquivos
uname	Sistema	Exibe informações do sistema operacional em execução
wget	Rede	Realiza o download de arquivos presentes ma Web

Alguns comandos serão mais frequentemente utilizados para a construção e operacionalização de sensores visuais em aplicações IoT. Por exemplo, os seguintes comandos apresentam igualmente o endereço IP atualmente associado ao Raspberry, mudando apenas a quantidade de informações exibidas:

```
$ ip -4 addr show
$ hostname --all-ip-address
$ ip addr list
```

Já o seguinte comando lista as redes sem fio disponíveis:

```
$ iwlist wlan0 scan
```

Uma rede sem fio pode ser definida diretamente no arquivo de configuração /etc /wpa_supplicant/wpa_supplicant.conf, no caso de uma configuração realizada, por exemplo, dentro de um *script*.

Uma configuração importante para sensores visuais é a definição de um endereço IP fixo. Embora tal configuração possa ser definida em um servidor DHCP, por exemplo associando endereços MAC à endereços IP, usuários podem estabelecer no próprio Raspberry qual endereço IP deve ser utilizado. Uma forma usual é alterando o arquivo "/etc/dhcpcd.conf". Outra forma que pode ser utilizada é apresentada no exemplo a seguir, que altera o endereço IP da interface conectada à rede Wi-Fi (wlan0).

```
$ ip addr add 10.1.1.100 dev wlan0
```

Embora esses e muitos outros comandos sejam bastante úteis, muitos usuários podem se sentir mais confortáveis com a utilização de ferramentas integradas de configuração, como o comando *raspi-config*. Esse comando irá abrir uma tela de configuração para importantes configurações do sistema.

Por fim, assim como ocorre com comandos no Linux, pode-se automatizar a execução de diversos comandos através de *scripts*, diretamente no *bash* ou utilizando a linguagem de programação Python. Contudo, deve-se lembrar que muitos do comandos necessários podem necessitar de acesso privilegiado de super usuário (root). O usuário padrão "sudo" do Raspbian Stretch é *pi* e a senha padrão desse usuário é *raspberry*. Como muitos comandos exigem acesso privilegiado, deve-ser ter bastante cuidado ao executar comandos com permissão de "root".

4.2.4. Câmera do Raspberry Pi

Após o lançamento da plataforma Raspberry e com o rápido sucesso obtido, foi lançado o Raspberry Pi Camera Module em 2013. Essa é uma câmera de baixo consumo de energia porém com "boa" definição, sendo pequena e flexível o suficiente para acompanhar a filosofia de desenvolvimento do Raspberry. Como foi desenvolvida pelo mesmo fabricante do Raspberry, a Raspberry Pi Camera (chamada aqui apenas de RaspCamera) é 100% compatível e pode ser facilmente conectada através de uma interface CSI (*Camera Serial Interface*), presente em todos os modelos atuais. Em 2016, a segunda versão da RaspCamera foi lançada, juntamente com a versão infra-vermelha (NoIR).

A RaspCamera V2 é a referência para os exemplos descritos neste Capítulo. A Figura 4.1 apresenta a RaspCamera V2 "padrão"e a infra-vermelha (NoIR).

Para conectar a RaspCamera, deve-se primeiramente desligar o Raspberry. A câmera é conectada diretamente na interface CSI, removendo o ajuste de plástico e encaixando a câmera de modo que os conectores do cabo e do Raspberry entrem em contato. Por fim, deve-se habilitar a câmera no Raspbian (a câmera é desabilitada por padrão), utilizando o painel de controle ou o comando *raspi-config*. A RaspCamera deve ser habilitada na opção "interface".

A Figura 4.2 apresenta um Raspberry Pi 2 model B com uma RaspCamera V2

conectada à interface CSI.

Figura 4.2. Câmera conectada ao Raspberry pela interface CSI.

Embora câmeras USB convencionais possam ser conectadas ao Raspberry, ou mesmo qualquer tipo de câmera através da GPIO, a RaspCamera trás diversas vantagens para aplicações baseadas na captura de imagens e/ou vídeos. De fato, a comunicação através da CSI, o tamanho reduzido, a "boa" resolução e o baixo consumo de energia colocam essa câmera como uma excelente opção para a construção de sensores visuais com o Raspberry Pi.

A Tabela 4.4 apresenta algumas das especificações da RaspCamera V2.

Tabela 4.4. Características básicas da RaspCamera V2.

Resolução máxima de imagem	3280 x 2464 (8 MP)
Resolução máxima de vídeo	1920 x 1080 (HDMI)
Codecs de imagem	JPEG, GIF, BMP, PNG, YUV420 e RGB888
Codec de vídeo	h.264 (*raw*)
FoV horizontal	62.2°
FoV vertical	48.8°

Como comentário final, para o Raspberry Pi Zero deve-se adaptar o cabo da câmera, porque a interface CSI é menor que no Raspberry Pi 2 e 3. Para tanto, pode-se adquirir adaptadores específicos para esse fim.

4.3. Trabalhando com imagens

Para iniciar no uso da câmera do Raspberry, que é um recurso fundamental para a construção de sensores visuais, serão apresentados exemplos de uso do programa *raspistill*. Esse programa é padrão no Raspbian e possui diferentes parâmetros de configuração para diversas funções relacionadas a obtenção de imagens pela câmera do Raspberry.

Digitando o comando a seguir, é apresentada um tela de “preview” por tempo indeterminado. Essa opção é útil para ver se a câmera está funcionando corretamente e qual é a imagem que será capturada.

```
$ raspistill -t 0
```

Para recuperar uma imagem instantânea atraves da câmera (“foto”), deve-se especificar o nome do arquivo que irá salvar a imagem, através da opção “-o”. O comando a seguir apresenta um exemplo do uso dessa opção. A imagem salva pode depois ser facilmente visualizada clicando sobre ela ou através de qualquer programa de reprodução de imagens.

```
$ raspistill -o foto.jpg
```

Quando a opção “-o” é utilizada da maneira apresentada, a tela de *preview* é exibida e a imagem é salva após 5 segundos, na pasta atual. Para que a imagem seja salva após um tempo específico, sem apresentar a tela de *preview*, deve-se executar o programa *raspitill* como descrito a seguir (para 1 segundo de “intervalo”):

```
$ raspistill -n -t 1000 -o foto.jpg
```

Há diversas operações que ocorrem quando a câmera é iniciada, preparada e utilizada para a captura de uma imagem. Com a opção *verbose* habilitada (“-v”) é possível ver todas as configurações estabelecidas (valores padrões são apresentados, quando não explicitamente definidos). A Figura 4.3 apresenta o *print* de uma captura de uma imagem através do comando *raspistill* no Raspbian *Desktop*.

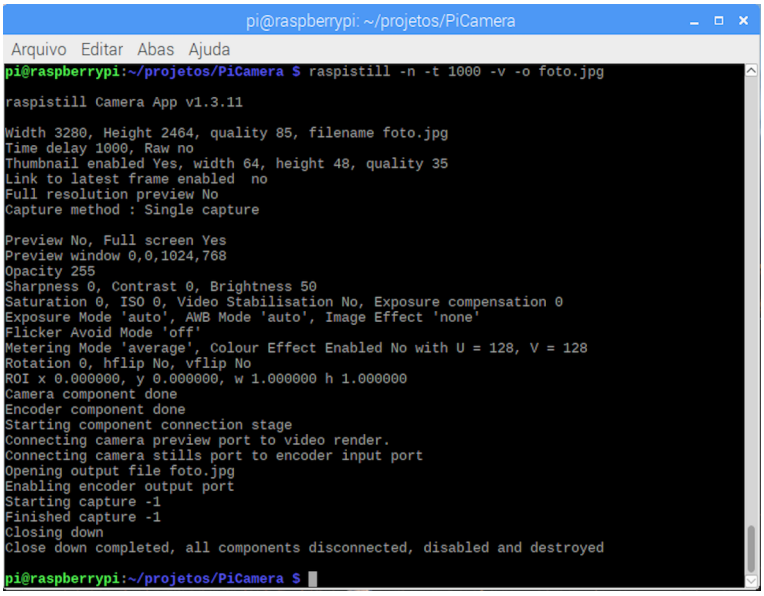

Figura 4.3. Configurações e procedimentos para captura de uma imagem.

Por padrão, o comando *raspistill* salva a imagem na resolução máxima possível, a menos que indicado de outra forma através de parâmetros específicos. O comando a seguir altera a resolução da imagem que será capturada, definindo a imagem no formato VGA.

```
$ raspistill -n -w 640 -h 480 -t 2000 -o foto.jpg
```

Há ainda parâmetros para funções mais complexas. O comando a seguir salva uma imagem P&B no formato PNG (o parâmetro "-cfx" altera o padrão de cores).

```
$ raspistill -n -cfx 128:128 -t 1 -o foto.png -e png
```

Caso a câmera esteja invertida, a imagem pode ser facilmente rotacionada. O comando a seguir apresenta um exemplo que rotaciona a imagem em 180^o.

```
$ raspistill -n -rot 180 -t 500 -o foto.jpg
```

A Tabela 4.5 apresenta uma lista com alguns parâmetros do programa *raspistill*. Para a lista completa de opções, a documentação oficial pode ser consultada gratuitamente online.

Tabela 4.5. Parâmetros de configuração do programa *raspistill*.

Parâmetro	Funcionalidade
-k	Exibe a tela de *preview* e apenas salva a imagem após o usuário pressionar a tecla <ENTER>
-e	Define o codec para a imagem a ser capturada (opções: jpg, bmp, gif e png)
-t	Altera o tempo de *delay* antes de uma imagem ser capturada
-rot	Rotaciona a tela de *preview* e a imagem capturada de acordo com o ângulo informado
-p	Estabelece a dimensão e posição da tela de *preview*
-n	Desabilita a tela de *preview*
-f	Apresenta a tela de *preview* em modo de "tela cheia" (*full screen*)
-ex	Estabelece o modo de exposição da lente (algumas opções: *night*, *backlight*, *spotlight*, *sports*, *snow*, *beach*, *antishake* e *fireworks*)
-q	Indica a qualidade da compressão das imagens JPEG (varia de 0 a 100). Impacta a qualidade da imagem e o tamanho do arquivo
-r	Salva a imagem sem compressão (*raw*)
-br	Altera o brilho da imagem (varia de 0 a 100)
-co	Altera o contraste da imagem (varia de -100 a 100)
-sh	Altera a nitidez da imagem (varia de -100 a 100)
-ifx	Estabelece um filtro para a imagem (algumas opções: *negative*, *solarise*, *whiteboard*, *sketch*, *emboss*, *oilpaint*, *pastel*, *film* e *saturation*)
-v	Exibe na tela informações de configuração e captura

As opções de efeitos na imagem são úteis para diversas aplicações e devem ser consideradas como um recurso poderoso. O exemplo a seguir apresenta o comando para salvar uma imagem com o efeito *watercolour*:

```
$ raspistill -ifx watercolour -o imagem.jpg
```

Um exemplo de uma imagem salva após a execução desse comando é exibida na Figura 4.4.

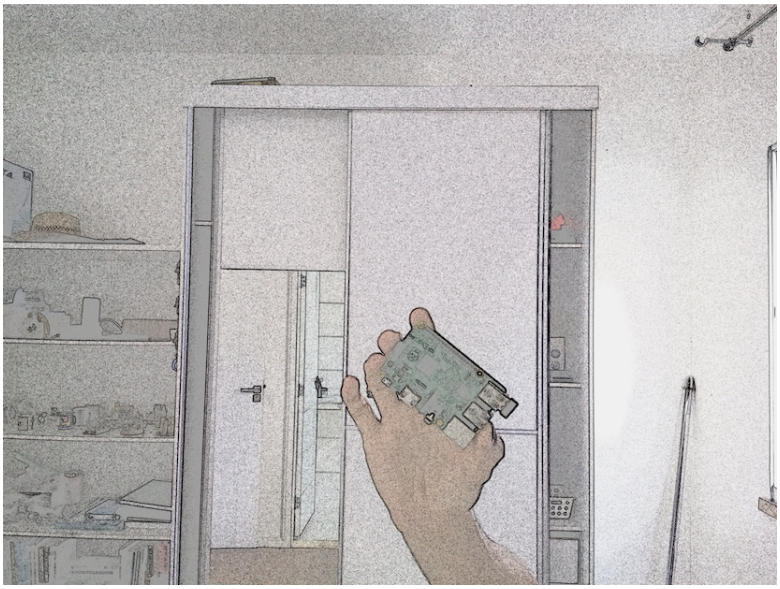

Figura 4.4. Exemplo de imagem com aplicação de filtro.

O comando *raspistill* também pode ser utilizado para capturar diversas imagens em sequência. Para tanto, deve-se utilizar a opção *timelapse* ("-tl"). O comando a seguir captura uma imagem a cada 5 segundos, durante 1 minuto, sendo as imagens salvas em arquivos numerados sequencialmente. Após a execução desse comando, 13 arquivos de imagens serão salvos, de "img0.jpg" a "img12.jpg" (a opção "%2d" define uma contagem com 2 dígitos). A resolução estabelecida é 300 x 300 (a menor resolução possível é 64 x 64).

```
$ raspistill -w 300 -h 300 -t 60000 -tl 5000 -o img%2d.jpg
```

Outro comando que pode ser utilizado para captura de imagens é o *raspiyuv*. Esse comando possui muitas opções semelhantes àquelas do comando *raspistill*, com a diferença que o comando *raspiyuv* produz saídas sem formatação e sem processamento.

4.4. Trabalhando com vídeos

O Raspbian também disponibiliza um programa para gravar vídeo, o *raspivid*. Por padrão, o comando *raspivid* salva vídeos na resolução 720p, a menos que indicado de outra forma através de parâmetros específicos. De fato, há diversos parâmetros de configuração para esse programa, sendo muitos desses parâmetros semelhantes aos disponíveis para o programa *raspistill* (Tabela 4.5).

Vamos iniciar com um exemplo. A seguir é apresentado um comando para gravar 15 segundos (15000 milisegundos) de vídeo no arquivo "video.h264". Esse vídeo é gravado como um *raw stream* h.264 e, portanto, não está inserido em um "container" (como avi ou mpg). Assim, o arquivo de vídeo não contém informações adicionais, como uma trilha de áudio, por exemplo.

```
$ raspivid -t 15000 -o video.h264
```

A Figura 4.5 apresenta o *print* de uma captura de um vídeo através do comando *raspivid* no Raspbian *Desktop*, quando a opção "-v" é utilizada. É interessante notar as configurações realizadas automaticamente para a captura do vídeo: resolução 1920 x 1080 e 30fps.

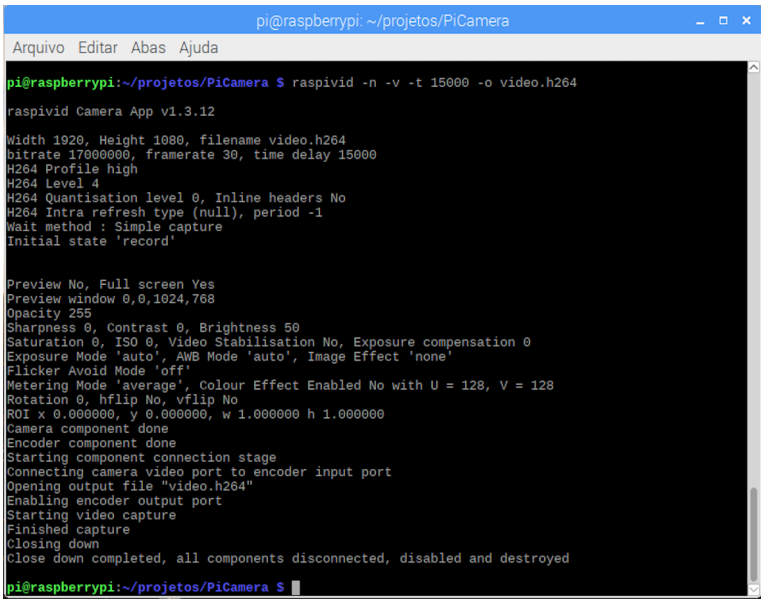

Figura 4.5. Configurações e procedimentos para captura de um vídeo.

As opções "-w" e "-h" podem ser utilizadas para alterar a resolução do vídeo que será gravado, naturalmente alterando o tamanho do arquivo final. O exemplo a seguir apresenta a captura de um vídeo de 20s na resolução 200 x 200.

```
$ raspivid -t 20000 -w 200 -h 200 -o video2.h264
```

Outros parâmetros bastante úteis é "-b", para alterar a taxa de bit do arquivo de vídeo, e "-fps", para alterar o número de quadros por segundo.

Após salvar um arquivo de vídeo, ele pode ser transmitido para outro dispositivo, para reprodução e/ou processamento, ou mesmo ser considerado localmente. Entre as opções para reproduzir o vídeo localmente, a ferramenta "omxplayer" apresenta-se como uma boa opção, além de já estar disponível nativamente no Raspbian.

Um exemplo da utilização da ferramenta "omxplayer" é apresentado a seguir. Há diversas opções que podem ser ativadas apertando determinados botões, por exemplo pausando a reprodução do vídeo ou alterando sua velocidade de reprodução.

```
$ omxplayer video.h264
```

Como o vídeo gravado com o programa *raspivid* é *raw*, ele poderá ser reproduzido com restrições e perdas de qualidade em determinadas ferramentas. Uma solução para esse "problema" é encapsulá-lo em algum "container" (por exemplo, mpg). O programa "MP4Box" permite realizar tal processamento. Para tanto, deve-se instalar o pa-

cote gratuito “gpac”. O seguinte comando encapsula o vídeo gerado pelo *raspivid* em um “container” específico (MPEG4).

```
$ MP4Box -add video.h264:fps=30 video.mp4
```

Além do “omxplayer”, podem-se utilizar ferramentas mais robustas. Um bom exemplo nesse sentido é a ferramenta VLC.

4.5. Criando sensores visuais com Python

Embora os programas *raspistill* e *raspivid* sejam bastante úteis e possam ser inclusive automatizados em *scripts* Bash ou Python, por exemplo, há diversas ações relativas a um sensor visual que podem ser melhor desempenhadas com recursos de programação mais completos. Nesse sentido, a câmera do Raspberry Pi se torna tão poderosa quanto for sua API de utilização. Neste Capítulo iremos apresentar a API “picamera”, disponível para a linguagem Python e instalada por padrão no Raspbian.

De fato, a linguagem Python é bastante utilizada pela comunidade em torno do Raspberry Pi, principalmente devido a sua flexibilidade, facilidade de programação e extensa gama de API disponíveis, permitindo um controle profundo dos recursos do Raspberry. A referência neste Capítulo é o Python 3.

4.5.1. Automatizando sensores visuais

Com a flexibilidade de programação trazida com o uso da linguagem Python, um número incontável de possibilidades são apresentadas ao Raspberry Pi e aos hardwares conectados. Essa subseção apresenta diversos exemplos para auxiliar na construção de sensores visuais explorando a API picamera.

O código a seguir rotaciona a câmera em 180^o e captura uma imagem, salvando-a no arquivo “pic.jpg”.

```
1 import picamera
2
3 camera = picamera.PiCamera()
4 camera.rotation = 180
5 camera.capture ("pic.jpg")
```

Esse pequeno código é bastante simples, porém apresenta características interessantes do uso da API picamera. Inicialmente, na linha 1, é realizada a “importação” da API para o código. Na linha 3 é criado um objeto do tipo PiCamera(): é através desse objeto que a câmera é acessada. Na linha 5, uma imagem é capturada e salva no arquivo “pic.jpg” (não é exibida a tela de *preview*).

A seguir é apresentado um código para exibir a tela de *preview* por 15 segundos, sem salvar qualquer imagem. Nesse código é utilizado o tratamento de exceções nativo da linguagem Python.

```
1 import picamera
2 import time
3
4 camera = picamera.PiCamera()
```

```
5  try:
6      camera.start_preview()
7      time.sleep(15)
8      camera.stop_preview()
9  finally:
10     camera.close()
```

Uma sequência de fotos é capturada no código a seguir, definindo para tanto a resolução 640 x 480 para as imagens salvas. Ao final da execução desse código, 10 arquivos de imagens serão salvos, cada um sendo criado com intervalo de 1 segundo entre as capturas.

```
1  from picamera import PiCamera
2  from time import sleep
3
4  #2 segundos de preparacao
5  sleep(2)
6
7  with PiCamera() as cam:
8          cam.resolution = (640, 480)
9      for n in range(1,11):
10         cam.capture("pic" + str(n) + ".jpg")
11         sleep (1)
```

Utilizando a API picamera, diversos métodos podem ser empregados para interação com a câmera do Raspberry. A Tabela 4.6 apresenta alguns desses métodos que podem ser utilizados.

Seguindo a ideia de explorar os métodos disponíveis para interação com a câmera, o exemplo a seguir apresenta diversas variáveis que alteram dinamicamente propriedades da captura de dados visuais, modificando a forma como a imagem é exibida na tela de *preview*. Para cada uma de quatro propriedades (*brightness*, *contrast*, *saturation* e *sharpness*), o código uniformemente varia o valor possível entre o mínimo e o máximo, em intervalos de 10 segundos. Deve-se lembrar que alguns atributos, como por exemplo *resolution*, só podem ser alterados quando a câmera estiver ociosa.

```
1  from picamera import PiCamera
2  from time import sleep
3
4  with PiCamera() as camera:
5      camera.resolution = (640, 480)
6      camera.start_preview()
7
8      try:
9          for i in range(101):
10             camera.brightness = i
11             sleep(0.1)
12
13         camera.brightness = 50
```

Tabela 4.6. Alguns métodos da API picamera.

Método	Descrição
capture()	Captura uma imagem e salva no arquivo especificado
capture_continuous()	Captura uma sequência contínua de imagens
capture_sequence()	Captura uma sequência de imagens
close()	Libera os recursos alocados para a câmera
start_recording()	Inicia a gravação de um novo *stream* de vídeo
stop_recording()	Interrompe a gravação de vídeo corrente
wait_recording()	Define o tempo de duração de uma gravação de vídeo
resolution()	Configura a resolução de imagem ou vídeo
image_effect()	Define um efeito para a imagem
start_preview()	Exibe a janela de *preview*
stop_preview()	Fecha a janela de *preview*

```
14
15          for i in range(-100, 101):
16              camera.contrast = i
17              sleep(0.05)
18
19          camera.contrast = 0
20
21          for i in range(-100, 101):
22              camera.saturation = i
23              sleep(0.05)
24
25          camera.saturation = 0
26
27          for i in range(-100, 101):
28              camera.sharpness = i
29              sleep(0.05)
30
31      finally:
32          camera.stop_preview()
33          camera.close()
```

É importante chamar o método *close()* quando a câmera não for mais necessária, uma vez que a câmera consome em média 250mA quando está em execução, o que inicia com a criação do objeto "PiCamera".

Outras configurações interessantes são realizadas no código a seguir. Nesse có-

digo é alterado o tamanho da janela de *preview* e o modo de exposição da câmera. Além disso, é aplicado um filtro à imagem. Por fim, a imagem é salva no formato PNG.

```
1  from picamera import PiCamera
2  from time import sleep
3
4  with PiCamera() as camera:
5      camera.resolution = (320, 320)
6
7      camera.exposure_mode = "beach"
8      camera.image_effect = "pastel"
9
10     camera.preview_fullscreen = False
11     camera.preview_window = (100, 100, 320, 320)
12
13     camera.start_preview()
14
15     #Preparar a camera
16     sleep (2)
17     camera.capture ("foto.png", format="png")
18     camera.close()
```

Uma sequência de imagens também pode ser criada utilizado o método *capture_sequence()*, como apresentado a seguir.

```
1  from picamera import PiCamera
2  from time import sleep
3
4  with PiCamera() as camera:
5      camera.resolution = (320, 320)
6      sleep (1)
7
8      #sequencia 1
9      camera.capture_sequence
       (["img1.jpg","img2.jpg","img3.jpg"])
10
11     sleep (1)
12     #sequencia 2
13     camera.capture_sequence
       (("img%1d.jpg" % i for i in range (4,11)))
14
15     camera.close()
```

Já o código a seguir salva imagens de forma ininterrupta, a cada 0,5 segundos, gravando a descrição da hora de captura no nome de cada arquivo.

```
1  from picamera import PiCamera
2  from time import sleep
3
```

```
4  with PiCamera() as camera:
5
6      sleep (1)
7
8      for arquivo in camera.capture_continuous
       ('pic_{timestamp}.jpg',use_video_port=False):
9          print ("Arquivo %s salvo" % arquivo)
10         sleep (0.5)
```

Vídeos também podem ser facilmente criados com a API picamera. No geral, as configurações necessárias são semelhantes, com a diferença que devem-se chamar outros métodos para isso. O código a seguir cria um vídeo de 10 segundos de duração, com 15fps e com resolução de 128 x 128.

```
1  from picamera import PiCamera
2  from time import sleep
3
4  camera = PiCamera()
5  camera.framerate = 15
6
7  camera.resolution = (128,128)
8  camera.preview_fullscreen = False
9  camera.preview_window = (0, 0, 128, 128)
10
11 camera.start_preview()
12
13 #Inicia a captura do video
14 camera.start_recording("video.h264")
15
16 sleep(10)
17
18 #Encerra a captura do video
19 camera.stop_recording()
20
21 camera.stop_preview()
22
23 camera.close()
```

No exemplo apresentado, a janela de *preview* é exibida no canto superior esquerdo da tela, exatamente do mesmo tamanho do vídeo criado.

O próximo exemplo grava um vídeo usando o método *wait_recording()*, na resolução padrão (1080p) e por 20 segundos. Esse método é mais interessante por retornar um erro caso o vídeo não possa ser gravado (por exemplo, for falta de espaço de armazenamento), sendo uma opção melhor que utilizando o método *time.sleep()*.

```
1  from picamera import PiCamera
2
3  camera = PiCamera()
```

```
4  camera.framerate = 30
5
6  camera.start_preview()
7  camera.start_recording("video2.h264")
8
9  #grava um video de 20 segundos
10 camera.wait_recording(20)
11
12 camera.stop_recording()
13
14 camera.stop_preview()
15 camera.close()
```

Há ainda muitos outros métodos úteis para utilizar a API picamera. Contudo, os métodos e exemplos apresentados já oferecem uma boa base para a construção de sensores visuais IoT.

4.5.2. Interagindo com a GPIO

Um dos recursos mais poderosos do Raspberry é a disponibilidade de 40 pinos de interação com outros hardwares, chamados simplesmente de GPIO (*General Purpose Input/Output*). Com esse pinos, o Raspberry pode se conectar com uma infinidade de outros dispositivos, ampliando significativamente as possibilidades no desenvolvimento de sensores de vídeo com o Raspberry Pi. Com a adoção do modelo de GPIO com 40 pinos, que se mantém nas versões mais recentes, inclusive no pequeno Raspberry Pi Zero, diversos fabricantes puderam desenvolver hardwares adicionais ao Raspberry com maior facilidade.

Cada pino da GPIO possui uma funcionalidade bem específica, que é padrão, como apresentado na Figura 4.6. Para facilitar as atividades que serão desenvolvidas nesta seção, os pinos GPIO estão apresentados de forma simplificada, não diferenciando os recursos adicionais que podem estar presentes em alguns pinos. Deve-se notar que a identificação numérica dos pintos (de 1 a 40) é diferente da sua utilização. Por exemplo, o pino 11 é o pino GPIO 17. A Figura 4.6 é uma ótima referência para utilizar corretamente cada pino.

Dos 40 pinos da GPIO, os pinos 2 e 4 fornecem 5V, os pinos 1 e 17 fornecem 3.3V e os pinos 6, 9, 14, 20, 25, 30, 34 e 39 são pinos "terra"(GND). Os demais pinos possuem diversas funcionalidades, podendo não apenas interagir diretamente com outros hardwares, mas também realizarem comunicações utilizando os padrões UART, I2C e SPI (pinos em “azul” na Figura 4.6), devendo esses recursos serem consultados em material complementar. Contudo, os pinos em verde na Figura 4.6 são exclusivos para entradas e saídas genéricas, sendo esses pinos utilizados prioritariamente neste Capítulo.

De fato, cada um dos pinos GPIO podem ser definidos como pinos de entrada ou saída, operando todos eles com apenas dois valores bem definidos: LOW (0V) ou HIGH (3.3V). Basicamente, todos os pinos estão no modo de “entrada” por padrão quando o sistema é iniciado, sendo a configuração correta dos pinos fundamental para sua operação apropriada.

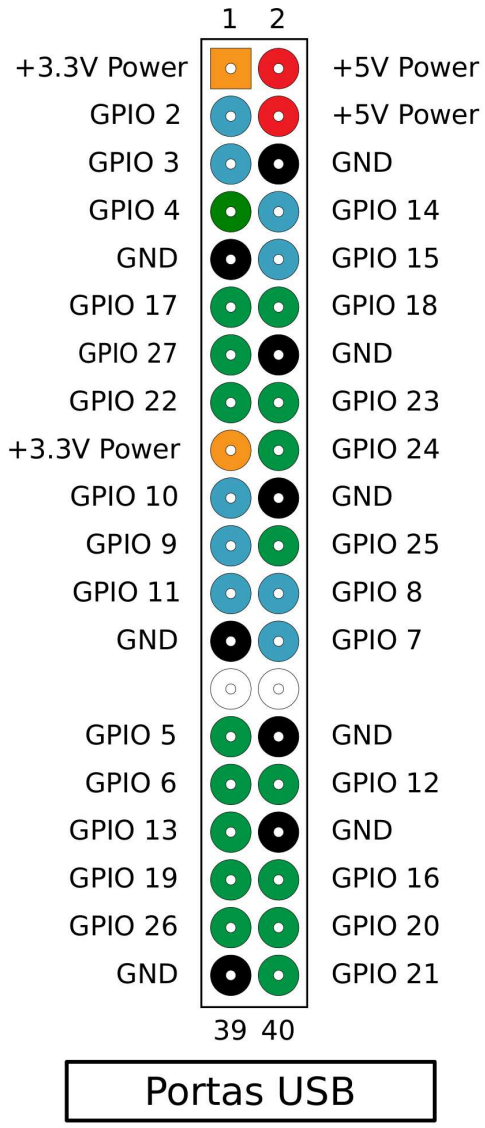

Figura 4.6. Funções de cada pino da GPIO.

Um lembrete importante é que como a entrada é feita com 3.3V no estado HIGH, deve-se ter cuidado com a utilização de componentes que utilizam uma voltagem maior, sobretudo porque muitos componentes operam com 5V. Para tanto, deve-se dimensionar apropriadamente o circuito desenvolvido, usando resistores quando necessário.

Neste Capítulo serão apresentados alguns exemplos de utilização da GPIO através da programação em Python, integrando as funcionalidades apresentadas com o uso da câmera. De fato, existem algumas bibliotecas no Python para interação com a GPIO, com níveis diferentes de recursos e complexidades. Contudo, durante toda esta seção será utilizada a API "gpiozero", que é uma API que facilita sobremaneira o uso de diversos componentes comuns da eletrônica, como LEDs e botões (*push-buttons*). Dessa forma, ao invés de se preocupar em configurar individualmente cada pino que será usado (por exemplo, indicando se o pino será de entrada ou saída), abstrações de alto nível serão

utilizadas.

A Tabela 4.7 apresenta alguns dos principais componentes modelados pela API gpiozero.

Tabela 4.7. Componentes modelados pela API gpiozero.

Classe	Tipo	Descrição	Métodos comuns
LED	Saída	LED padrão	on() off() blink()
RGBLED	Saída	LED rgb	on() off() blink() color()
Buzzer	Saída	Componente *buzzer*	on() off() beep()
Motor	Saída	Motor genérico	forward() backward() stop()
Button	Entrada	Botão (*push-button*)	wait_for_press() wait_for_release()
MotionSensor	Entrada	Sensor de movimento	wait_for_motion() wait_for_no_motion()
LightSensor	Entrada	Sensor de luminosidade	wait_for_light() wait_for_dark()
DistanceSensor	Entrada	Sensor de distância	wait_for_in_range() wait_for_out_of_range()

4.5.2.1. Controlando a câmera com botões

O acesso aos pinos da GPIO será feito diretamente a partir de um programa Python, utilizando para tanto a biblioteca gpiozero (instalada por padrão no Raspbian). Essa biblioteca permite o acesso facilitado há diversos pinos da GPIO, com abstrações de alto nível.

O primeiro exemplo a ser considerado irá utilizar um *push-button* conectado a uma *breadboard*, que será então conectado ao pino GPIO 18. A conexão é bastante simples: o pino 6 (GND) é conectado a uma linha do botão, sendo a linha oposta conectada ao pino GPIO 18. Para interagir com o botão criado, pode-se utilizar o código apresentado a seguir, que define o método "fotografar()".

```
1 import gpiozero
2 import picamera
3 import signal
4
5 camera = picamera.PiCamera()
6
7 def fotografar():
```

```
8           camera.capture ("imagem.jpg")
9
10  botao = gpiozero.Button (18)
11
12  botao.when_pressed = fotografar
13
14  signal.pause()
```

A classe gpiozero.Button já encapsula todos os detalhes necessários para interagir com um botão, bastando apenas informar qual é o pino que será associado ao botão desejado no processo de criação do objeto, como realizado na linha 10. A partir daí, o botão real já pode ser "acessado" através desse objeto.

A instrução na linha 14, "signal.pause()", evita que o programa seja encerrado antes do usuário apertar o botão. Quando em execução, toda vez que o usuário apertar o botão, uma nova imagem será salva no arquivo "imagem.jpg".

O próximo exemplo não apenas salva imagens com o clique do botão, mas também acende um LED. Para tanto é considerado o circuito apresentado na Figura 4.7.

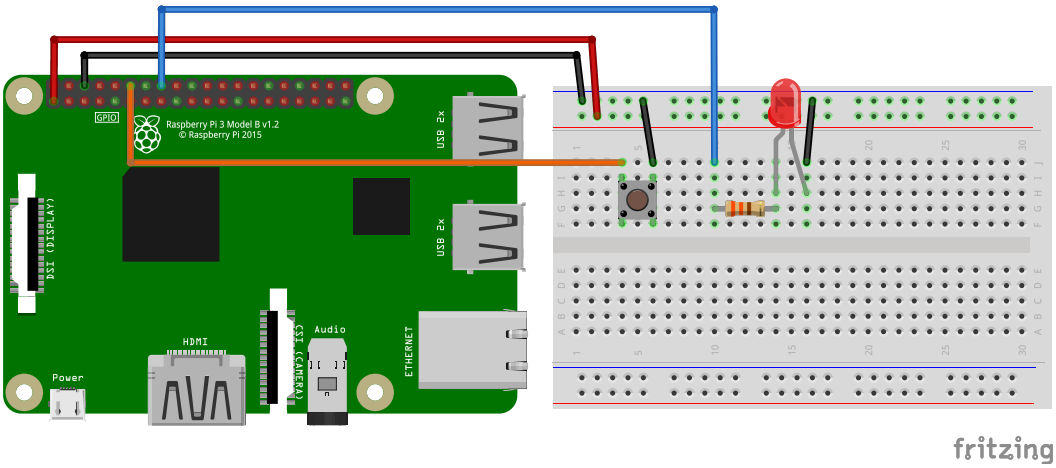

Figura 4.7. Circuito para captura de imagem a partir de um botão.

Para o circuito apresentado, pode-se utilizar o código descrito a seguir. Enquanto o botão está automaticamente configurado para a GPIO 18, como um *input*, o LED está configurado no pino GPIO 23, com um *output*. Além disso, nas funções definidas, será executado um trecho de código quando o botão for pressionado ("when_pressed") e outro código quando o botão for liberado ("when_released").

```
1  from picamera import PiCamera
2  from gpiozero import Button, LED
3  from signal import pause
4  from time import sleep
5
6  camera = PiCamera()
```

```
7
8  #acessando via GPIO
9  botao = Button (18)
10 led = LED (23)
11
12 #capturar imagem
13 def fotografar():
14     camera.capture("imagem.jpg")
15
16 #acender LED por 2 segundos
17 def ligarled():
18     led.on()
19     sleep(2)
20     led.off()
21
22 botao.when_pressed = ligarled
23 botao.when_released = fotografar
24
25 pause()
```

No próximo exemplo são utilizados dois botões e dois LEDs. O primeiro botão é utilizado para disparar a captura de uma imagem, acionando em seguida um LED e um *buzzer* (para emitir um sinal sonoro). Já o segundo botão dispara a captura de um vídeo de 10 segundos, também deixando um LED específico ligado por 10 segundos. O código para esse exemplo é apresentado a seguir.

```
1  from picamera import PiCamera
2  from gpiozero import Button, LED
3  from signal import pause
4  from time import sleep
5
6  camera = PiCamera()
7
8  botao1 = Button (18)
9  botao2 = Button (23)
10
11 led1 = LED (24)
12 led2 = LED (25)
13
14 def filmar():
15     camera.start_recording("video.h264")
16     camera.wait_recording (10)
17     camera.stop_recording()
18
19 def fotografar():
20     camera.capture("imagem.jpg")
21
```

```
22 def ligarledVideo():
23     led1.on()
24     sleep(10)
25     led1.off()
26
27 def ligarledCamera():
28     led2.on()
29     sleep(1)
30     led2.off()
31
32 botao1.when_pressed = ligarledCamera
33 botao1.when_released = fotografar
34
35 botao2.when_pressed = ligarledVideo
36 botao2.when_released = filmar
37
38 pause()
```

O circuito relacionado a esse exemplo é apresentado na Figura 4.8.

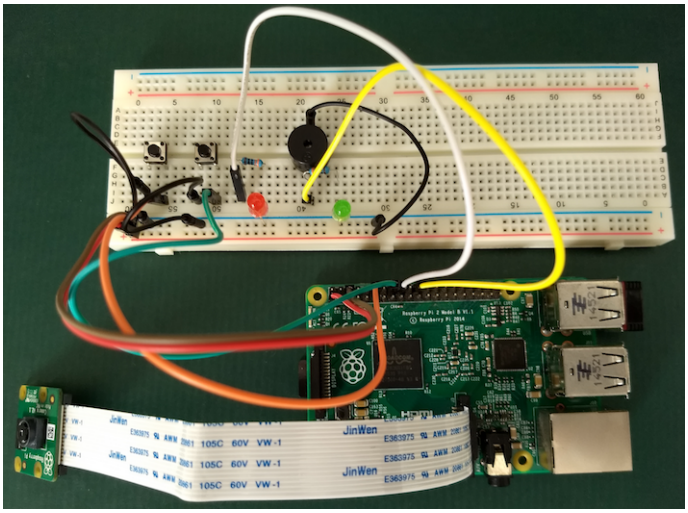

Figura 4.8. Circuito para captura de imagem e vídeo a partir de diferentes botões.

Por fim, para exemplificar o uso de botões através da GPIO, a tabela 4.8 apresenta métodos comuns que podem ser utilizados por objetos do tipo Button.

4.5.2.2. Tirando fotos com um sensor de distância

Outro exemplo interessante do uso da GPIO é a conexão com um sensor de distância (proximidade). Para tanto, foram escritos dois códigos de exemplo para conexão a um sensor ultra-sônico HC-SR04, que é um sensor de distância barato e fácil de operar. Esse sensor possui quatro pinos, sendo um deles utilizado para controle do sensor (*trigger*) e outro para receber informações sobre a distância dos objetos localizados (*echo*). Contudo,

Tabela 4.8. Alguns métodos de gpiozero.Button.

Método	Descrição
wait_for_press()	Pausa a execução do *script* até o botão ser pressionado (*timeout* também pode ser definido)
wait_for_release()	Pausa a execução do *script* até o botão ser liberado (*timeout* também pode ser definido)
held_time()	Tempo em segundos que o botão esteve pressionado
is_pressed()	Retorna "True" se o botão estiver pressionado
when_held()	Indicação da função que será chamada quando o botão for pressionado pelo tempo indicado
when_pressed()	Indicação da função que será chamada quando o botão for pressionado (sai do estado "inativo" para o estado "ativo")
when_released()	Indicação da função que será chamada quando o botão for liberado (sai do estado "ativo" para o estado "inativo")

como esse sensor opera com 5V, deve-se utilizar resistores para a entrada de dados no Raspberry, uma vez que o nível HIGH da GPIO é 3.3V. A Figura 4.9 apresenta o esquema de interconexão, utilizando dois resistores de $1K\Omega$.

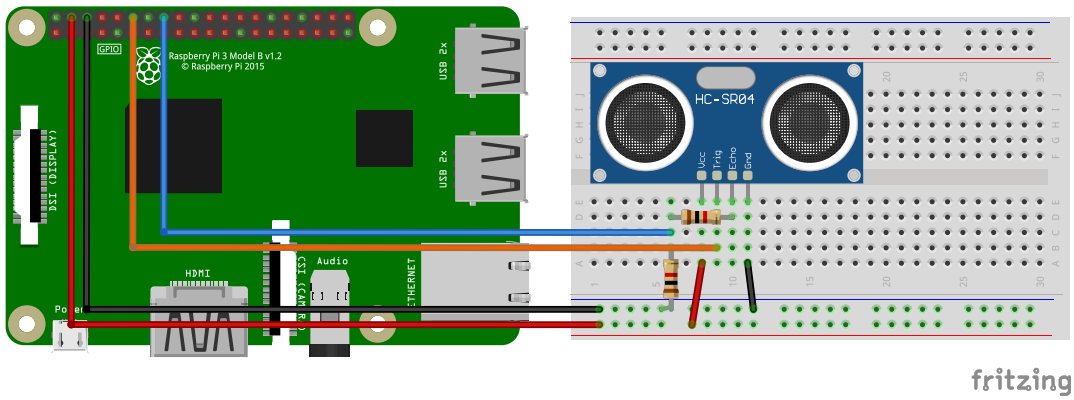

Figura 4.9. Conectando um sensor HC-SR04 ao Raspberry.

O código apresentado a seguir captura uma imagem quando um objeto está em uma distância de até 80cm em relação ao sensor. Para tanto, o pacote gpiozero cria a abstração DistanceSensor, que facilita bastante a utilização desse tipo de sensor.

```
1 from gpiozero import DistanceSensor
2 from picamera import PiCamera
3 from time import sleep
4 from datetime import datetime
5
6 camera = PiCamera()
```

```
7
8  def fotografar():
9      hora = datetime.now()
10     camera.capture(str(hora) + ".jpg")
11     print ("Imagem capturada: " + str(hora))
12
13 sensor = DistanceSensor (echo=23, trigger=18,
   max_distance=2.0)
14
15 while True:
16         distancia = sensor.distance
17
18         #distancia em metros
19         if (distancia <= 0.8):
20             fotografar()
21
22         sleep (1)
```

Quando um objeto gpiozero.DistanceSensor é criado, deve ser especificado quais são os pinos *trigger* e *echo*. A partir daí, pode-se acessar dinamicamente o valor da distância de objetos localizados pelo sensor. Há diversos métodos disponíveis que são úteis quando é utilizado o objeto DistanceSensor, o que oferece mais flexibilidade ao processo de programação. Por exemplo, o código a seguir possui a mesma funcionalidade do código anteriormente apresentado (detecção de objetos na distância de até 80cm), estando a diferença nos métodos que são empregados.

```
1  from gpiozero import DistanceSensor
2  from picamera import PiCamera
3  from time import sleep
4  from datetime import datetime
5
6  camera = PiCamera()
7
8  def fotografar():
9      hora = datetime.now()
10     camera.capture(str(hora) + ".jpg")
11     print ("Imagem capturada: " + str(hora))
12
13 sensor = DistanceSensor (echo=23, trigger=18,
   max_distance=0.8)
14
15 while True:
16         sensor.wait_for_in_range()
17         fotografar()
```

A Figura 4.10 apresenta o circuito criado seguindo o modelo definido.

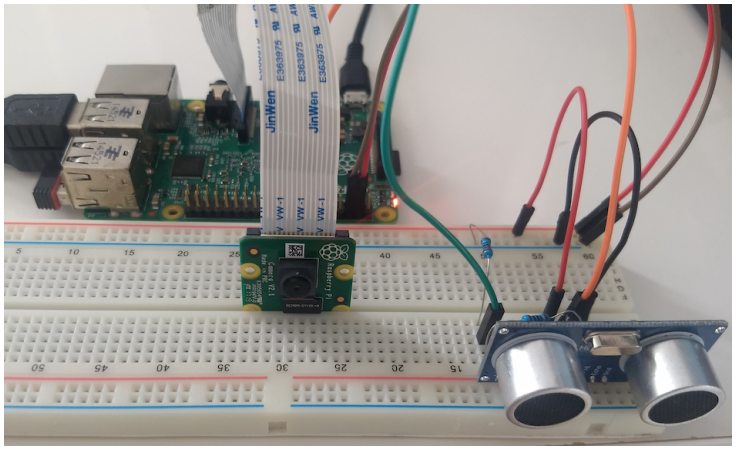

Figura 4.10. Sensor de vídeo ativado por presença.

4.5.3. Automatizando a execução de *scripts*

Em diversas situações, a execução de *scripts* pode ser automatizada para que seja realizada em momentos definidos pelo usuário. E esse tipo de serviço é útil em diversas situações, garantindo maior flexibilidade e controle avançado em inúmeras aplicações desenvolvidas com o Raspberry Pi, incluindo aquelas destinadas ao monitoramento por câmera.

A execução automatizada de *scripts* e/ou programas em geral é um serviço oferecido pelo sistema operacional. Como este Capítulo é baseado no Raspbian, recursos básicos do Linux Debian podem ser utilizados para essa função.

Considerando a necessidade de automatizar a execução de *scripts* que seja válida para o sistema como um todo e que ocorra sempre que o Raspberry for iniciado, pode-se utilizar o arquivo "/etc/profile". As instruções presentes nesse arquivo serão então executadas na inicialização do sistema, sendo isso ideal quando sensores visuais precisam entrar em operação automaticamente após serem ligados.

O exemplo a seguir apresenta um código Python que vai ligar um LED (GPIO 18) toda vez que o sistema for iniciado. O circuito que deve ser montado é muito simples, sendo responsável apenas pela ativação desse LED.

```
1 from gpiozero import LED
2 from time import sleep
3
4 led = LED (18)
5
6 #Por padrao, o pino GPIO18 esta LOW quando iniciado
7 led.on ()
8 sleep (10)
9 led.off ()
```

Considerando o nome do arquivo como sendo "ligarled.py", estando localizado na pasta *home* do usuário padrão do sistema, podemos inserir a seguinte linha de comando ao final do arquivo "/etc/profile":

```
python /home/pi/ligarled.py
```

Após a configuração do arquivo "/etc/profile", toda vez que o sistema for iniciado ou quando houver um novo *login*, o LED indicado será ligado por 10 segundos, sendo apagado em seguida. Deve-se perceber que como há uma chamada ao método "time.sleep()", o fluxo de processamento só será devolvido ao sistema após a execução do *script* indicado.

Já para executar *scripts* ou programas em geral em uma frequência definida, pode-se utilizar o serviço "crontab". O comando *$ crontab -e* irá permitir a configuração do arquivo de configuração desse serviço. A partir daí é possível alterar diretamente o arquivo de configuração, que possui um padrão de formatação simples que indica a frequência de execução de determinado comando (1 vez a cada hora, 3 vezes ao dia, 1 vez por semana, etc). O exemplo a seguir define a execução do *script* "ligarled.py"a cada minuto, para todos os dias.

```
* * * * * python /home/pi/ligarled.py
```

Há, de fato, diversas opções para configuração do crontab, sendo elas bastante úteis na configuração de sensores visuais. Por exemplo, a opção "@reboot"pode ser usada para indicar que o comando indicado será executado a cada inicialização do sistema. Para maiores informações sobre a configuração desse serviço, bibliografia complementar deve ser consultada, ou mesmo a documentação oficial do Raspbian.

4.6. Transmitindo a partir do Raspberry

Frequentemente, a captura de imagens e vídeos feitas por um Raspberry deverá ser transmitida para outro computador, para processamento e/ou reprodução. Esse cenário é, de fato, muito comum em aplicações IoT [Jyothi and Vardhan 2016, Sruthy and George 2017]. Para tanto, há diversos recursos disponíveis, sendo alguns deles descritos nesta seção.

De fato, *streaming* pode ser realizado de diferentes formas possíveis, podendo utilizar como base o TCP (confiabilidade) ou UDP (tempo-real). Os próximos exemplos apresentam soluções com diferentes abordagens.

4.6.1. Transmissões com programas padrões

O vídeo capturado a partir da câmera do Raspberry pode ser facilmente transmitido utilizando o comando *raspivid*. Para tanto, utilizam-se recursos já disponíveis no Raspbian. Aqui será apresentado um exemplo simples dessa possibilidade.

No exemplo apresentado a seguir, considerando uma comunicação cliente-servidor, o Raspberry irá se comportar como um cliente (que produz o vídeo e se conecta à um servidor para enviar esse vídeo), enquanto o servidor irá apenas reproduzir o vídeo recebido. O comando a seguir, que deve ser executado no lado Servidor, aguarda uma conexão na porta 15000 (usa o comando "nc"para isso). Quando a conexão é iniciada, o fluxo de bytes é direcionado ao *player* de vídeo "vlc", que é configurado para processar um fluxo de vídeo no formato h.264.

```
$ nc -l 15000 vlc –demux h264 -
```

No lado do cliente (Raspberry), pode-se digitar o comando apresentado a seguir. O comando *raspivid* é usado para transmitir por 60 segundos um pequeno vídeo com resolução 128 x 128, que é então direcionado para a conexão criada pelo comando "nc". Neste exemplo é considerado o endereço IP do servidor como sendo "10.0.0.10".

```
$ raspivid -w 128 -h 128 -t 60000 -o - nc 10.0.0.10 15000
```

Deve-se perceber que o comando *netcat* ("nc") está criando um fluxo de dados com o protocolo TCP, o que garante confiabilidade na transmissão em detrimento do tempo da comunicação. Dessa forma, o vídeo recebido é armazenado em cache (bufferização) e só é reproduzido quando isso for possível de forma contínua. Contudo, para diversas aplicações, um fluxo de dados sem perdas de transmissão pode ser desejado.

4.6.2. Transmissões com o MJPEG-streamer

O MJPEG-streamer é um conjunto de bibliotecas e programas para permitir a transmissão de imagens e vídeos em rede, de forma facilitada. De fato, essa opção é bastante flexível e permite transmissões de diversas formas possíveis. Nesta seção será apresentado um exemplo de como utilizar esse recurso.

Inicialmente, algumas bibliotecas e dependências devem ser instaladas, como apresentado a seguir:

```
$ apt-get install build-essential libjpeg8-dev
imagemagick libv4l-dev cmake
```

O MJPEG-streamer pode ser baixado do endereço "http://lara.uefs.br" (há outras fontes na web, inclusive no Github). O arquivo ".zip" que será baixado deve ser descompactado e compilado da seguinte forma:

```
$ make
$ make install
```

Se tudo estiver correto e não houver problemas na compilação, o programa mjpeg-streamer será instalado na pasta "/usr/local/bin/", que é uma pasta que já está no $PATH do sistema. A partir daí, esse programa poderá ser acessado livremente.

O exemplo a seguir inicia o *streaming* a partir da câmera do Raspberry Pi.

```
$ mjpg_streamer -i "input_raspicam.so -d /dev/video0
-fps 15 -q 80" -o "output_http.so -p 8080
-w /usr/local/share/mjpg-streamer/www"
```

Na execução desse comando é indicado como entrada (*input*, "-i") a interface representada no arquivo "/dev/video0", sendo a transmissão feita com 15 frames por segundo e "qualidade" jpeg estabelecida em 80. A câmera do Raspberry deve está indicada pelo sistema operacional no arquivo /dev/video0, o que faz como que ela se comporte como uma câmera USB qualquer. Caso isso não ocorra, pode-se digitar o comando "modprobe bcm2835-v4l2", carregando o módulo necessário. Por fim, a opção de saída (*output*, "-o") é estabelecida como um fluxo HTTP na porta 8080. Assim, pode-se acessar facilmente o fluxo transmitido em um navegador Web.

A Figura 4.11 apresenta o acesso em um computador ao fluxo transmitido pelo Raspberry. O acesso é feito digitando o endereço IP do Raspberry e indicando a porta 8080 (por exemplo, http://10.3.1.1:8080).

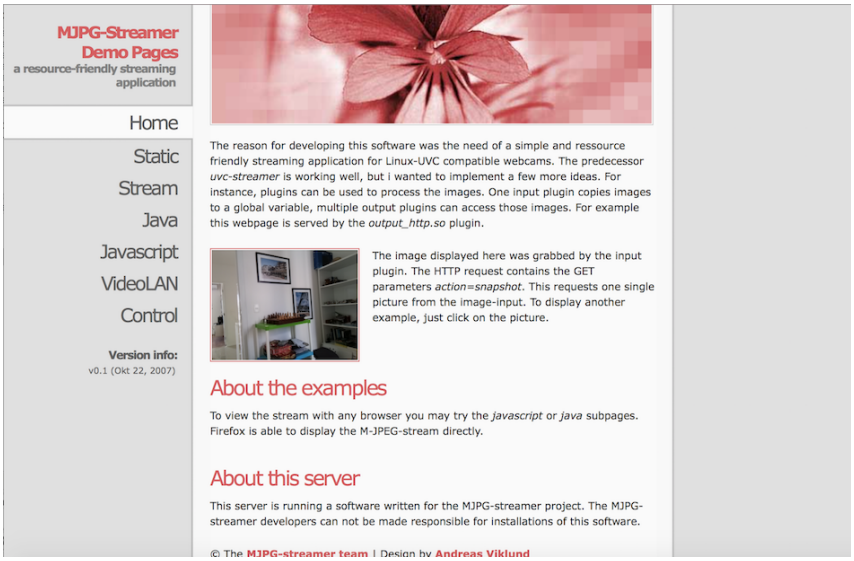

Figura 4.11. Tela principal quando o Raspberry é acessado pelo navegador Web na porta 8080.

Na tela apresentada há algumas opções para interação com o fluxo transmitido. Nas opções apresentadas, “Static” permite visualizar uma imagem instantânea, capturada naquele momento, enquanto “Stream” exibe em tempo real um fluxo de vídeo.

4.6.3. Transmissões pela API picamera

A API picamera também oferece recursos para realizar a transmissão de imagens e vídeos em tempo real, como alternativa ao armazenamento local de arquivos. O exemplo a seguir apresenta um código para realizar *streaming* por 60 segundos, enviando o vídeo capturado para o endereço “10.0.0.10” na porta 15000.

```
1  import socket
2  import picamera
3  import time
4
5  ip = "10.0.0.10"
6  porta = 15000
7
8  socket_cliente = socket.socket()
9  socket_cliente.connect ((ip, porta))
10
11 conexao = socket_cliente.makefile ('wb')
12
13 try:
14     camera = picamera.PiCamera()
15     camera.resolution = (320, 320)
```

```
16      camera.framerate = 15
17
18      time.sleep(2)
19
20      camera.start_recording (conexao,format="h264")
21      camera.wait_recording (60)
22      camera.stop_recording ()
23
24  except:
25
26      print ("Erro no envio do video")
27
28  finally:
29      camera.close()
30      socket_cliente.close()
```

Nesse exemplo é utilizado o pacote "socket", que é um pacote para conexões diversas na Internet (pilha TCP/IP). Através de um objeto "socket" é iniciada uma conexão com o endereço IP e a porta TCP especificados e, portanto, a parte "servidor' da conexão já deve estar esperando novas conexões. De fato, o lado servidor pode ser criado como na subseção anterior, utilizando o comando a seguir (o programa "vlc" é substituído pelo programa "mplayer"):

```
$ nc -l -p 15000 mplayer -fps 15 -cache 1024 -
```

Outra opção viável para o lado servidor é criar um *script* Python para reproduzir o vídeo. Para este exemplo, será feita uma comunicação direta entre dois Raspberry conectados à mesma rede Wi-Fi: um deles (cliente) irá transmitir o vídeo para o outro Raspberry (servidor), que irá reproduzir o vídeo recebido. O código do lado cliente pode ser o mesmo do apresentando anteriormente. Já o código para o lado do servidor é apresentando a seguir.

```
1  import socket
2  import subprocess
3
4  porta = 15000
5
6  socket_servidor = socket.socket()
7  socket_servidor.bind(('0.0.0.0', porta))
8  socket_servidor.listen(0)
9
10 conexao = socket_servidor.accept()[0].makefile('rb')
11 try:
12     comando = ['vlc', '--demux', 'h264', '-']
13
14     player = subprocess.Popen(comando,
       stdin=subprocess.PIPE)
15
```

```
16      while True:
17          # Ler 1k de dados e enviar ao player
18          data = conexao.read(1024)
19
20          if not data:
21              break
22
23          player.stdin.write(data)
24
25  finally:
26      conexao.close()
27      socket_servidor.close()
28      player.terminate()
```

O exemplo apresentado utiliza o programa "vlc", que é externo ao Python, para reproduzir o vídeo recebido. Essa é, de fato, uma estratégia simples e fácil de implementar, porém bibliotecas adicionais poderiam ser utilizadas para reproduzir o vídeo pelo próprio Python.

4.7. Copiando arquivos do Raspberry

Frequentemente, os arquivos de imagem e vídeo salvos em um Raspberry deverão ser copiados para outro Raspberry, computador convencional ou mesmo um celular, permitindo assim que os arquivos possam ser reproduzidos, processados, armazenados ou mesmo distribuídos. E isso pode ser feito de diferentes formas, como apresentado nesta seção.

4.7.1. Conexão direta via SSH

Uma solução simples para obter dados visuais capturados pelo Raspberry é conectando-se a ele diretamente. A partir daí, arquivos de imagem e vídeo podem ser facilmente copiados. De fato, o acesso direto a um Raspberry através do protocolo SSH, por exemplo, ou mesmo usando soluções de acesso remoto de Desktop (como o VNC), é uma tarefa simples caso seja conhecido o endereço IP do Raspberry e este esteja acessível.

Considerando que o Raspberry Pi em questão e o computador que irá acessá-lo estão na mesma rede (ou acessíveis através de uma rede IP), pode-se realizar uma conexão SSH direta para permitir a cópia de arquivos. Deve-se lembrar, contudo, que o acesso SSH deve estar habilitado no Raspberry (por exemplo, usando o comando *raspi-config*).

A Figura 4.12 apresenta uma conexão via SSH à um Raspberry, que neste caso está acessível pelo endereço "192.168.1.12".

Para acessar os arquivos através do protocolo SSH, existem diversos programas que podem ser utilizados. Uma forma simples é realizar a cópia de arquivos por linha de comando usando programas como o *scp*, nativamente presente em diversos sistemas operacionais. Outra opção é através de programas populares com diversos recursos disponíveis em interfaces interativas, como o "PuTTY" e o "WinSCP".

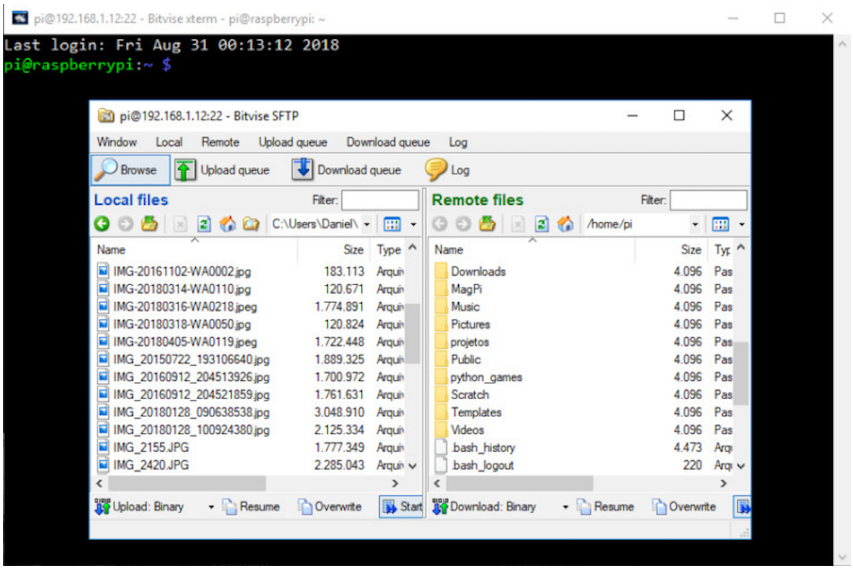

Figura 4.12. Conexão SSH à um Raspberry.

4.7.2. Transformando o Raspberry num AP

Quando houver muitos Raspberry operando de forma concorrente em determinado ambiente, uma solução simples e prática é transformar cada um dos dispositivos em um Wi-Fi Access Point (AP). Assim, para se conectar a um determinado Raspberry, basta selecionar o ID da rede Wi-Fi desejada e fazer a associação à rede. Para diversos cenários de monitoramento, quando for necessário se conectar individualmente à cada Raspberry, está será uma opção prática e rápida, sobretudo porque não é necessário configurar a adesão dos Raspberry à uma mesma rede.

Para configurar o Raspberry como um AP, serão utilizados dois serviços: "hostapd" e "dnsmasq'. Ambos os serviços podem ser facilmente instalados com o comando *apt-get*.

O primeiro passo para configurar um AP é definir um endereço IP estático para a interface Wi-Fi (wlan0). Para isso, considerando o sistema operacional Raspbian Stretch, deve-se alterar o arquivo /etc/dhcpcd.conf e incluir/alterar a seguinte informação (os dados utilizados são apenas exemplos e devem ser substituídos pelos valores apropriados).

```
interface wlan0
static ip_address=10.3.1.1/24
```

Como o objetivo não é conectar o Raspberry à Internet mas sim agir apenas como um Access Point offline, pode-se utilizar qualquer endereçamento válido na faixa de endereços IP. Por questões de conformidade com redes IP já existentes e por padronização, optou-se aqui por usar um endereço na rede privada 10.3.1.0/24, mas qualquer endereçamento poderia ser usado aqui. Neste exemplo, escolheu-se a seguinte padronização: o bloco de endereço "10", seguido pelo número "3" (uma vez que está sendo utilizado um Raspberry Pi 3 B) e o número "1", indicando que esse é o sensor número 1. Dessa forma, pode-se facilmente saber qual o sensor de vídeo que está sendo acessado.

Após a instalação do serviço "hostapd", deve-se configurar o arquivo /etc/hostapd/hostapd.conf. É neste arquivo que serão descritas as configurações da rede sem fio sendo criado. A seguir é apresentado um exemplo de configuração desse arquivo.

```
interface=wlan0
hw_mode=g
channel=1
wpa=2
wpa_key_mgmt=WPA-PSK
ssid=rasp3_1
wpa_passphrase=rasp3sensor1
```

No exemplo apresentado, apenas algumas configurações foram estabelecidas, deixando diversas outras configurações com valores padrões. Entre as opções definidas, é interessante notar a indicação do padrão IEEE 802.11g para comunicação, uso do canal 1 do padrão Wi-Fi (poderia ser automaticamente definido) e o padrão de criptografia baseado em WPA. Para o ID da rede (ssid), foi definido o nome "rasp3_1", sendo a senha definida como "rasp3sensor1". Como as redes Wi-Fi podem implementar criptografia, que é, inclusive, um serviço comumente utilizado, a adesão à determinada rede é geralmente controlada por senha. Contudo, pode-se configurar uma rede aberta (sem senha), porém se for utilizado WPAv2, a senha definida deve ter no mínimo 8 caracteres.

A terceira configuração necessária é definir o serviço DHCP, para que qualquer computador, celular ou dispositivo que se conecte ao Raspberry AP receba automaticamente um endereço IP para comunicação. Para tanto, deve-se adicionar ao arquivo /etc/dnsmasq.conf a seguinte informação.

```
interface=wlan0
dhcp-range=10.3.1.2,10.3.1.5,255.255.255.0,24h
```

Na configuração definida, o range de IP foi estabelecido entre 10.3.1.2 e 10.3.1.5, sendo então possíveis apenas quatro endereços IP (10.3.1.2, 10.3.1.3, 10.3.1.4 e 10.3.1.5).

Por fim, devem-se iniciar o serviços "hostapd" e "dnsmasq", como apresentado a seguir.

```
$ systemctl start hostapd
$ systemctl start dnsmasq
```

Após a realização das configurações apresentadas, a rede "rasp3_1" pode ser acessada por qualquer dispositivo habilitado para a rede Wi-Fi. A rede fica disponível como uma rede convencional, como apresenta a Figura 4.13.

Figura 4.13. Rede disponível para conexão ao Raspberry.

Após a conexão à rede criada, pode-se facilmente acessar o Raspberry e fazer cópia de arquivos, alterar configurações e/ou executar comandos. Pensando no cenário de

cópia de arquivos obtidos pelo sensor (imagens e vídeos), há duas abordagens bem interessantes. A primeira delas é usando o serviço SSH, como descrito na seção anterior. Uma outra opção é instalar um servidor Web no Raspberry, a exemplo do Apache. Instalando o servidor, os arquivos podem ser acessados diretamente através de um navegar convencional, como o "Firefox", o "Chrome" ou o "Safari". No caso do Apache, ele poder ser configurado de várias formas para exibir os arquivos de imagem e/ou vídeo capturados pelo Raspberry. Como uma solução viável, arquivos podem ser armazenados na pasta "/var/wwww/html", permitindo fácil e rápido acesso aos arquivos de imagem e/ou vídeo produzidos pelo Raspberry.

A Figura 4.14 apresenta um acesso via um *smartphone* ao endereço "10.3.1.1/imagens", considerando que o celular está conectado à rede "rasp3_1" definida previamente.

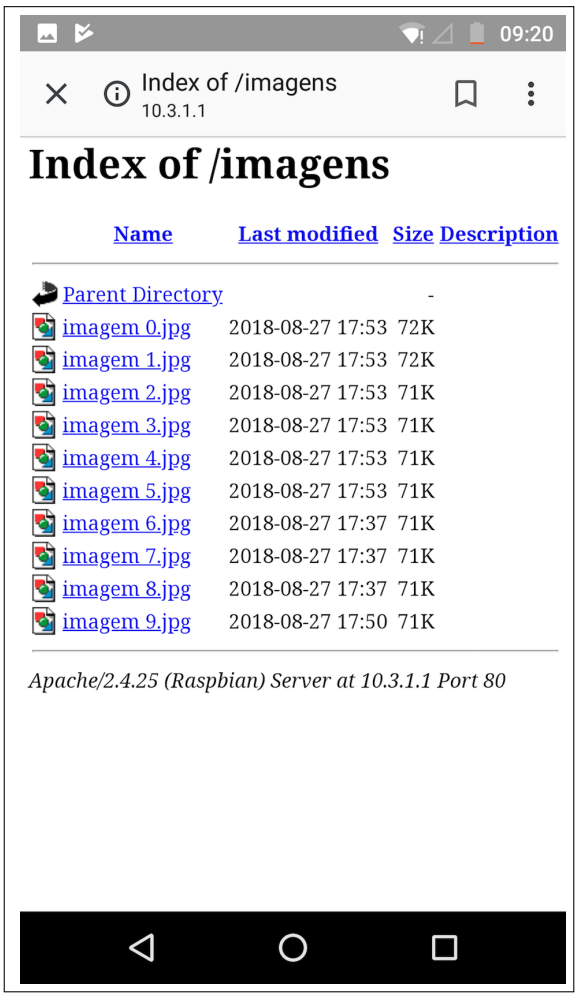

Figura 4.14. Acessando arquivos de imagens a partir de um celular.

4.8. Conclusão

O desenvolvimento tecnológico em diversas áreas, como microeletrônica, redes de comunicação, processamento paralelo, inteligência artificial e ciência de dados, vêm transformando significativamente o cenário da computação e das tecnologias da informação em geral. Nesse contexto, a Internet das Coisas surge como um grande escopo que integra

diversas tecnologias, revolucionando a forma como dados são capturados, distribuídos e processados. Ao final desta década, a Internet das Coisas já será o cenário dominante.

No mundo IoT, encontrar plataformas acessíveis para desenvolvimento e prototipação é de grande importância. O Raspberry surge como uma dessas plataformas, permitindo uma ampla gama de possibilidades de desenvolvimento. Com o uso de uma câmera, sensores de vídeo podem ser facilmente criados, o que aumenta sobremaneira as possibilidades de utilização do Raspberry em aplicações IoT.

Este Capítulo abordou a construção de um sensor de vídeo com o Raspberry Pi, apresentando alguns dos mais importantes detalhes de configuração e operação. Além disso, muitos exemplos mostraram usos práticos do Raspberry como sensor visual, permitindo sua utilização prática em diversas aplicações. Contudo, há ainda muitos detalhes que não foram abordados. Para o leitor, há muito material gratuito na Web, com inúmeros tutoriais e projetos constantemente mostrando novos usos promissores para o Raspberry. Apenas como referência, o site da Raspberry Foundation, "https://www.raspberrypi.org/", e da revista MagPi, "https://www.raspberrypi.org/magpi/", podem ser gratuitamente consultados para esse propósito.

Referências

[Andrade et al. 2017] Andrade, D. C., Costa, D. G., and ao B. Rocha-Junior, J. (2017). Adaptive sensing relevance exploiting social media mining in smart cities. In *Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web*, pages 405–408.

[Atzori et al. 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805.

[Costa et al. 2017] Costa, D. G., Collotta, M., Pau, G., and Duran-Faundez, C. (2017). A fuzzy-based approach for sensing, coding and transmission configuration of visual sensors in smart city applications. *Sensors*, 17(1):1–17.

[Costa et al. 2018] Costa, D. G., Duran-Faundez, C., Andrade, D. C., Rocha-Junior, J. B., and Just Peixoto, J. P. (2018). Twittersensing: An event-based approach for wireless sensor networks optimization exploiting social media in smart city applications. *Sensors*, 18(4).

[Costa et al. 2015] Costa, D. G., Guedes, L. A., Vasques, F., and Portugal, P. (2015). Research trends in wireless visual sensor networks when exploiting prioritization. *Sensors*, 15:1760–1784.

[Gupta et al. 2015] Gupta, M. S. D., Patchava, V., and Menezes, V. (2015). Healthcare based on iot using raspberry pi. In *Green Computing and Internet of Things (ICGCIoT)*, pages 796–799.

[Jyothi and Vardhan 2016] Jyothi, S. N. and Vardhan, K. V. (2016). Design and implementation of real time security surveillance system using iot. In *2016 International Conference on Communication and Electronics Systems (ICCES)*, pages 1–5.

[Kruger and Hancke 2014] Kruger, C. P. and Hancke, G. P. (2014). Benchmarking internet of things devices. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 611–616.

[Kuo et al. 2018] Kuo, Y., Li, C., Jhang, J., and Lin, S. (2018). Design of a wireless sensor network-based iot platform for wide area and heterogeneous applications. *IEEE Sensors Journal*, 18(12):5187–5197.

[Leone et al. 2017] Leone, G. R., Moroni, D., Pieri, G., Petracca, M., Salvetti, O., AzzarÃ, A., and Marino, F. (2017). An intelligent cooperative visual sensor network for urban mobility. *Sensors*, 17(11).

[Lin et al. 2017] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., and Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142.

[Nikhade 2015] Nikhade, S. G. (2015). Wireless sensor network system using raspberry pi and zigbee for environmental monitoring applications. In *Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pages 376–381.

[Patchava et al. 2015] Patchava, V., Kandala, H. B., and Babu, P. R. (2015). A smart home automation technique with raspberry pi using iot. In *Smart Sensors and Systems (IC-SSS)*, pages 1–4.

[Patil et al. 2017] Patil, N., Ambatkar, S., and Kakde, S. (2017). Iot based smart surveillance security system using raspberry pi. In *Communication and Signal Processing (ICCSP)*, pages 0344–0348.

[Perera et al. 2014] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25:81–93.

[Sandeep et al. 2015] Sandeep, V., Gopal, K. L., Naveen, S., Amudhan, A., and Kumar, L. S. (2015). Globally accessible machine automation using raspberry pi based on internet of things. In *Advances in Computing, Communications and Informatics (ICACCI)*, pages 1144–1147.

[Sruthy and George 2017] Sruthy, S. and George, S. N. (2017). Wifi enabled home security surveillance system using raspberry pi and iot module. In *2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–6.