

Sistemas Operacionais Embarcados

Internet

Internet

- Conteúdo:
 - Conexão à internet
 - Download de arquivos
 - Download de arquivos em serviços em nuvem (Dropbox, Google Drive etc.)
 - Buscas pelo Google
 - Hora mundial
 - Envio de informações (formulários, arquivos etc.)
 - E-mail
 - Servidor web local
 - *Streaming* de video
 - Referências

Conexão à internet

Modelos do Raspberry Pi sem WiFi

- Conecte o cabo ethernet ao RPi. Ele deve se configurar automaticamente para acessar a internet.

Modelos do Raspberry Pi com WiFi

- Execute `sudo iwlist wlan0 scan | grep SSID` para obter os nomes das redes WiFi disponíveis
- Abra o arquivo `/etc/wpa_supplicant/wpa_supplicant.conf`, e acrescente o seguinte texto ao final dele:

```
network={  
    ssid="NOME_DA_REDE"  
    psk="SENHA_DA_REDE"  
}
```

onde `ssid` é o nome da rede, e `psk` é a senha da rede.

Conexão à internet

Modelos do Raspberry Pi com WiFi

- Se quiser criptografar a senha, execute `wpa_passphrase NOME_DA_REDE` e digite a senha da rede. Escreva no arquivo `/etc/wpa_supplicant/wpa_supplicant.conf` o texto retornado no terminal:

```
network={  
    ssid="testing"  
    #psk="testingPassword"
```

```
psk=131e1e221f6e06e3911a2d11ff2fac9182665c004de85300f9cac208a6a80531  
}
```

(Retire o texto precedido pelo símbolo # para maior segurança.)

Conexão à internet

Modelos do Raspberry Pi com WiFi

- Na UnB, utilize a rede eduroam com a seguinte configuração:

```
network={
    ssid="eduroam"
    proto=RSN
    key_mgmt=WPA-EAP
    eap=PEAP
    identity="SEU_EMAIL@unb.br"
    password="SUA_SENHA"
    phase1="peaplabel=0"
    phase2="auth=MSCHAPV2"
}
```

onde os campos `identity` e `password` contém seu e-mail unb.br e sua senha.

Download de arquivos

- A maneira mais simples de baixar arquivos da internet pelo RPi é utilizando o comando `wget`. Por exemplo,

```
wget  
https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Embarcados/master/README.md
```

baixa um arquivo no GitHub, e `wget www.unb.br` baixa o arquivo HTML da página principal da UnB.

Download de arquivos

- O comando `curl` oferece muito mais opções do que o `wget`, sendo bastante interessante para o que veremos mais à frente. Execute

```
curl -O  
https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Embarcados/master/README.md
```

```
curl -O www.unb.br
```

para obter os mesmos resultados.

- Execute `curl -o saida.html www.unb.br` para salvar o resultado no arquivo local `saida.html`

Download de arquivos em serviços em nuvem (Dropbox, Google Drive etc.)

- Para fazer o *download* de arquivos em serviços de armazenamento em nuvem, tais como o Dropbox, o Google Drive e o Microsoft OneDrive, pode ser necessário realizar alguns passos extra. Antes de mais nada, é necessário liberar o acesso ao arquivo no serviço em nuvem.
- Considere o arquivo `Arquivo_na_nuvem.pdf`, disponível em <https://drive.google.com/file/d/11uI5-BAXzIhKO14clUwmQlAo8LyqS-Ui/view?usp=sharing>. É necessário separar nesta URL o identificador do arquivo (uma sequência aparentemente aleatória de caracteres) para fazer o download adequadamente:

```
filename="Arquivo_na_nuvem1.pdf"
```

```
fileid="11uI5-BAXzIhKO14clUwmQlAo8LyqS-Ui"
```

```
curl -L -o ${filename}  
"https://drive.google.com/uc?export=download&id=${fileid}"
```


Download de arquivos em serviços em nuvem (Dropbox, Google Drive etc.)

- Se o arquivo tiver mais de 100MB, o Google Drive pede uma espécie de autenticação antes de permitir o *download* do mesmo (por exemplo, em https://drive.google.com/file/d/1EPVcIVl0UiNtjw9S751ZRiw_FQdTLHf0/view?usp=sharing). Nesse caso, execute:

```
filename="Arquivo na nuvem2.pdf"
fileid="1EPVcIVl0UiNtjw9S751ZRiw_FQdTLHf0"
curl -L -c cookies.txt 'https://docs.google.com/uc?export=download&id='$fileid | sed -rn
's/*.confirm=([0-9A-Za-z ]+).*/\1/p' > confirm.txt
curl -L -b cookies.txt -o $filename 'https://docs.google.com/uc?export=download&id='$fileid'&confirm='
$(<confirm.txt)
rm -f confirm.txt cookies.txt
```

- Procedimento semelhante pode ser necessário com o Dropbox e o OneDrive.

Buscas pelo Google

- Para buscar pelo termo “raspberry pi” em www.google.com, por exemplo, execute

```
curl -o saida.html https://www.google.com/search?q=raspberry+pi
```

e abra em um *browser* o arquivo salvo `saida.html`.

- Provavelmente você verá uma mensagem dizendo que você não teve permissão para fazer esta busca (por exemplo, `Your client does not have permission to get URL /search?q=raspberry+pi from this server.`). Isso acontece porque o Google procura impedir buscas automatizadas como esta que você acabou de tentar.

Buscas pelo Google

- Execute `curl -v www.unb.br > /dev/null`. No pedido HTTP, você verá um campo indicando qual é o agente que está fazendo o pedido. O `curl` geralmente preenche este campo como `curl\VERSAO_DO_CURL` (por exemplo, `User-Agent: curl\7.58.0`).
- O Google provavelmente colocou quaisquer agentes que contenham o nome `curl` na sua “lista negra”.

Buscas pelo Google

- Para conseguir sair da “lista negra”, acrescente a opção `-A USER-AGENT` ao comando anterior:

```
curl -A USER-AGENT -o saida.html  
https://www.google.com/search?q=raspberry+pi
```

e abra em um *browser* o arquivo salvo `saida.html` para ver a busca disponibilizada. (Se você não conseguir permissão novamente, provavelmente o agente `USER-AGENT` também entrou na lista negra do Google. Troque o nome do agente até conseguir.)

Hora mundial

- Para conferir a hora atual via IP da máquina, execute

```
curl http://worldtimeapi.org/api/ip.txt
```

- Para conferir a hora atual via o fuso-horário de Brasília/São Paulo, execute

```
curl http://worldtimeapi.org/api/timezone/America/Sao_Paulo.txt
```

- Para conferir os fuso-horários disponíveis, execute

```
curl http://worldtimeapi.org/api/timezone.txt
```

- Para atualizar a hora do seu Raspberry Pi, execute:

```
sudo date +%s -s @$ (curl -s http://worldtimeapi.org/api/ip.txt |  
grep unixtime | sed "s/unixtime: //" )
```

Envio de informações (formulários, arquivos etc.)

- Páginas web possuem diversos campos destinados a formulários. Por exemplo, a página de login de uma página geralmente possui campos para usuário e senha, além de um botão para envio destas informações. Barras de busca são outro exemplo relevante.
- Quando se baixa uma página na internet, é feita uma requisição HTTP GET. No caso do envio destas informações de formulários, é feita uma requisição HTTP POST. O comando `curl` permite automatizar este tipo de requisição.

Envio de informações (formulários, arquivos etc.)

- Por exemplo, entre na página <https://fga.unb.br>, e faça uma busca na barra correspondente pela palavra “eletronica”. Você será redirecionado para a página <https://fga.unb.br/search/articles?query=eletronica>. Uma forma simples de automatizar essa busca é executando

```
curl -v -o exemplo_POST.txt -d query=eletronica  
https://fga.unb.br/search/articles
```

```
less exemplo_POST.txt
```

- Repare na saída do comando curl que foi feita uma requisição POST.
- Pelo resultado da busca, pode-se perceber que uma requisição GET também é suficiente:

```
curl -v -o exemplo_POST.txt  
https://fga.unb.br/search/articles?query=eletronica
```

Envio de informações (formulários, arquivos etc.)

- Atualização de serviços em nuvem:
 - A instalação de aplicativos de serviços em nuvem pode ser uma solução simples, mas custosa computacionalmente, já que estes aplicativos continuamente conferem se houve mudanças em suas pastas principais e secundárias. Para fazer o upload de arquivos pelo terminal, cada serviço oferece uma solução diferente. Confira algumas delas nos links ao final deste documento.

Envio de informações (formulários, arquivos etc.)

- Comunicação via Telegram

- Com o comando `curl`, é possível criar bots no Telegram. Por exemplo, para mandar uma mensagem “Olá mundo”, execute

```
curl -s -X POST https://api.telegram.org/bot<TOKEN>/sendMessage -d chat_id=<CHAT_ID> -d text="Hello World"
```

onde <TOKEN> é uma string necessária para autorizar o bot a mandar pedidos à API do Telegram (por exemplo, 4334584910:AAEPmjlh84N62Lv3jGWEgOftlxxAfMhB1gs), e <CHAT_ID> é o identificador da conversa.

Envio de informações (formulários, arquivos etc.)

- Outros sistemas de mensagens e notificações
 - Pushbullet: conexão entre dispositivos
 - IFTTT (If This, Than That): conexão entre serviços na nuvem
 - MQTT: protocolo de comunicações máquina/máquina (M2M - Internet of Things)
 - PUSHETTA: notificações em tempo real para diferentes dispositivos (smartphones, browsers, smart TV etc.)
 - yowsup: comunicação via WhatsApp

Envio de informações (formulários, arquivos etc.)

- Armazenamento de dados no Google Sheets
 - No Google Drive, é possível criar formulários com campos específicos, e definir uma planilha online para receber os resultados do preenchimento deste formulário. Por exemplo, confira o formulário em https://docs.google.com/forms/d/e/1FAIpQLSf4wN-l8EwZRKXqAlMODInnRIFZQfEBBmjURL4M_-vgJ1r39A/viewform

Envio de informações (formulários, arquivos etc.)

- Armazenamento de dados no Google Sheets
 - É possível automatizar o preenchimento deste formulário pelo comando curl. Obtenha o identificador do formulário na sua URL de edição, e os nomes dos campos a serem preenchidos usando o inspetor de elementos HTML do seu browser (Ferramentas de desenvolvedor no Google Chrome). No exemplo acima, o identificador é `18YYhW1Dk3xtge66XdG38SfBuPEhm7esfBI4Ajhyh4Bg`, e os campos são `entry.1962235247` e `entry.146553730`. Repare que o identificador não é o mesmo que aquele em <https://docs.google.com/forms/d/e/1FAIpQLSf4wN-l8EwZRKXqAlM ODInnRIFZQfEBBmjURL4M -vgJ1r39A/viewform>

Envio de informações (formulários, arquivos etc.)

- Armazenamento de dados no Google Sheets
 - Para enviar o nome “Eu Mesmo” e a idade “18” para o formulário, execute

```
formid="18YYhW1Dk3xtge66XdG38SfBuPEhm7esfBI4Ajhyh4Bg"
```

```
curl https://docs.google.com/forms/d/$formid/formResponse -d ifg  
-d "entry.1962235247=Eu Mesmo" -d "entry.146553730=18" -d  
submit=Submit
```

- Confira os resultados em
<https://docs.google.com/spreadsheets/d/1-LeRomMtqndDE1nhHrhR7lkyY1sh0eXhOJHJ6dPbc-I/edit?usp=sharing>

E-mail

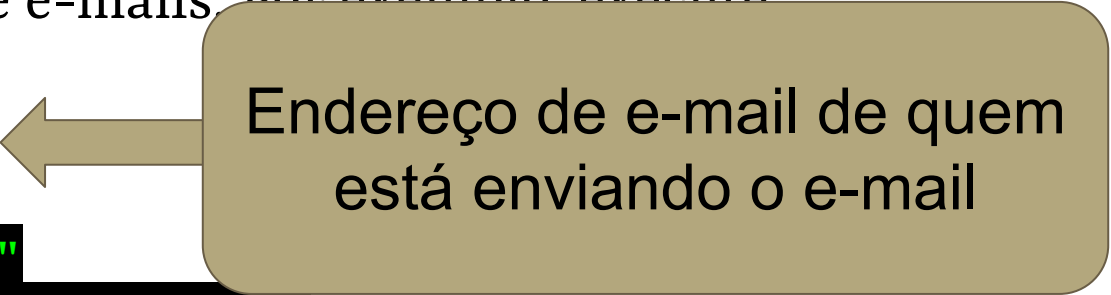
- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"
passw from="SENHA REMETENTE"
email to="EMAIL DESTINATARIO"
servidor envio="smtp://smtp.unb.br:587"
criptografia="--ssl"
email subject="ASSUNTO"
email msg="MENSAGEM"
echo "From: <$email from>" > email.txt
echo "To: <$email to>" >> email.txt
echo Subject: $email subject >> email.txt
echo Date: $(date) >> email.txt
echo $email_msg >> email.txt
curl -u $email_from:$passw_from -n -v --mail-from $email_from
--mail-rcpt $email_to --url $servidor_envio $criptografia -T
email.txt
rm email.txt
```

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp.unb.br:587"  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>" > email.txt  
echo "To: <$email to>" >> email.txt  
echo Subject: $email subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```

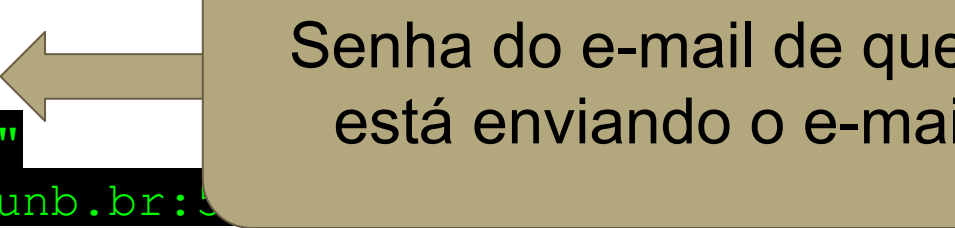


Endereço de e-mail de quem
está enviando o e-mail

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp.unb.br:587"  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>" > email.txt  
echo "To: <$email to>" >> email.txt  
echo Subject: $email subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```




Senha do e-mail de quem está enviando o e-mail

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp.unb.br:"  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>" > email.txt  
echo "To: <$email to>" >> email.txt  
echo Subject: $email_subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```




Endereço de destino do e-mail

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp.unb.br:587"  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>" > email.txt  
echo "To: <$email to>" >> email.txt  
echo Subject: $email subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```



Endereço SMTP do
servidor da UnB e a
porta correspondente

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp"  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>"  
echo "To: <$email to>" >> email.txt  
echo Subject: $email_subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```

Tipo de criptografia a ser usada,
de acordo com as instruções de
configuração de email da UnB
(<http://www.cpd.unb.br/cpd-ser-e-mail>, “Normas e Tutoriais”)

E-mail

- O comando `curl` trabalha com diversos protocolos, incluindo SMTP. Assim, ele permite o envio de e-mails. Por exemplo, execute

```
email from="EMAIL REMETENTE"  
passw from="SENHA REMETENTE"  
email to="EMAIL DESTINATARIO"  
servidor envio="smtp://smtp.  
criptografia="--ssl"  
email subject="ASSUNTO"  
email msg="MENSAGEM"  
echo "From: <$email from>"  
echo "To: <$email to>" >> email.txt  
echo Subject: $email_subject >> email.txt  
echo Date: $(date) >> email.txt  
echo $email_msg >> email.txt  
curl -u $email_from:$passw_from -n -v --mail-from $email_from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt
```

Ou seja, este exemplo funciona para EMAIL_REMETENTE com final @unb.br. Cada provedor de email utilizará um servidor diferente, bem como uma porta e um tipo de criptografia

E-mail

- Repare que o email enviado anteriormente foi escrito no arquivo `email.txt` antes de ser enviado com a opção `-T email.txt` do `curl`. Para acrescentarmos arquivos em anexo ao email, devemos inserir mais informações a este arquivo:

```
nome imagem="imagem.png"
email from="EMAIL REMETENTE"
passwd from="SENHA REMETENTE"
email to="EMAIL DESTINATARIO"
servidor envio="smtp://smtp.unb.br:587"
criptografia="--ssl"
email subject="ASSUNTO"
email_msg="MENSAGEM"
curl -o $nome_imagem
https://fga.unb.br/articles/0001/7219/guia-unb-gama.png
echo "From: <$email_from>" > email.txt
echo "To: <$email_to>" >> email.txt
echo Subject: $email_subject >> email.txt
```

E-mail

```
echo Date: $(date) >> email.txt
echo Content-Type: multipart/mixed\; boundary=corpo_msg >>
email.txt
echo >> email.txt
echo --corpo msg >> email.txt
echo Content-Type: text/plain\; charset=UTF-8 >> email.txt
echo >> email.txt
echo $email msg >> email.txt
echo >> email.txt
echo --corpo msg >> email.txt
echo Content-Type: image/png\; name=\"$nome_imagem\" >>
email.txt
echo Content-Transfer-Encoding: base64 >> email.txt
echo Content-Disposition: attachment; filename=\"$nome_imagem\"
>> email.txt
echo >> email.txt
```

E-mail

```
cat $nome imagem | base64 >> email.txt  
echo --corpo msg-- >> email.txt  
curl -u $email from:$passw from -n -v --mail-from $email from  
--mail-rcpt $email_to --url $servidor_envio $criptografia -T  
email.txt  
rm email.txt $nome_imagem
```

E-mail

- Neste exemplo, baixamos uma imagem de <https://fga.unb.br/articles/0001/7219/guia-unb-gama.png>, e a acrescentamos ao arquivo `email.txt` utilizando o comando `base64`
- Também indicamos no arquivo `email.txt` que este anexo é uma imagem do tipo PNG em

```
echo Content-Type: image/png\; name=\"$nome_imagem\" >>  
email.txt
```

Para outros tipos de arquivos (JPEG, PDF etc.), este campo deve ser modificado.

Servidor web local

- É possível criar um servidor web no Raspberry Pi para que seus dados estejam acessíveis via internet, seja na rede local ou globalmente. No segundo caso, é necessário adquirir um domínio. Dentre os diversos servidores web disponíveis no Linux, vejamos o `apache`

```
sudo apt-get update  
sudo apt-get install apache2 -y
```


para instalar o `apache2` no Raspberry Pi. Acesse <http://localhost/> em um *browser* no Raspberry Pi, ou [http://IP DO RASPBERRY PI](http://IP_DO_RASPBERRY_PI) em um computador, *tablet* ou celular ligado à mesma rede para visualizar a seguinte página:

Servidor web local

- É possível
este
seg
serv

```
sudo apt-get install apache2  
sudo apt-get install apache2-doc
```

para instalar
browser
computador
seguinte



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in** [/usr/share/doc/apache2/README.Debian.gz](#). Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/  
|-- apache2.conf  
|   |-- ports.conf  
|-- mods-enabled  
|   |-- *.load  
|   |-- *.conf  
|-- conf-enabled  
|   |-- *.conf  
|-- sites-enabled  
|   |-- *.conf
```

dados
e. No
rsos

n um

Servidor web local

- Quando você acessa esta página, você faz um requerimento HTTP GET da página HTML localizada em `/var/www/html/index.html` no Raspberry Pi com o `apache2` instalado e executando.
- Mude esta página HTML para apresentar o que quiser ao usuário.
- (Por precaução, execute `sudo cp /var/www/html/index.html /var/www/html/index_original.html` para manter uma cópia desta página no seu Raspberry Pi.)

Servidor web local

- Você deve ter percebido que a página web funcionou logo após a instalação do `apache2`. Isso quer dizer que ele está executando continuamente. Execute

```
sudo /etc/init.d/apache2 stop  
sudo update-rc.d apache2 disable
```

para parar sua execução, e

```
sudo update-rc.d apache2 enable  
sudo /etc/init.d/apache2 start
```

para reinicia-lo.

Servidor web local

- **Envio de dados do cliente para o servidor:**
 - O envio de informações de um cliente para um servidor pode ser feito através de requisições HTTP GET e POST. Com o `apache2` em execução, execute

```
sudo curl -o /var/www/html/index.html  
https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Em  
barcados/master/Code/24_Internet/formulario_RPi1.html  
sudo curl -o /var/www/html/obrigado.html  
https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Em  
barcados/master/Code/24_Internet/obrigado.html
```

Servidor web local

- **Envio de dados do cliente para o servidor:**
 - Acesse o servidor para visualizar uma página com um formulário contendo três campos de preenchimento e um menu *dropdown*. Digite qualquer coisa nestes campos, pressione o botão *Submit* e execute `tail -1 /var/log/apache2/access.log` para ver o último acesso feito ao arquivo `/var/www/html/index.html` via internet.
 - Repare que o que você digitou nos campos do formulário aparecem após a palavra GET, e você pode utilizar isto para ler o que o usuário mandou de informação para o seu Raspberry Pi.
 - Execute `sudo curl -o /var/www/html/index.html https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Embarcados/master/Code/24_Internet/formulario_RPi2.html` para visualizar um formulário com melhor aparência.

Servidor web local

- **Atualização da página:**
 - Execute

```
cd ~
echo '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html
xmlns="http://www.w3.org/1999/xhtml"><head><meta
http-equiv="Content-Type" content="text/html; charset=UTF-8"
/><title>Temperatura do Raspberry Pi</title></head><body><div
class="container"><h1>' > index.html
echo $(date) >> index.html
echo '</h1><p>' >> index.html
echo '$(/opt/vc/bin/vcgencmd measure temp) >> index.html
echo '</p></div></body></html>' >> index.html
sudo mv index.html /var/www/html/index.html
```

e acesse o servidor para visualizar uma página com informação da temperatura da GPU do Raspberry Pi. Isto é possível graças ao comando `/opt/vc/bin/vcgencmd measure temp`

Servidor web local

- **Atualização da página:**

- Atualize a página do servidor, e perceba que o valor indicado na página não muda. Isto acontece porque a página HTML não foi atualizada. Execute

```
curl -o update servidor.sh  
https://raw.githubusercontent.com/DiogoCaetanoGarcia/Sistemas_Emb  
arcados/master/Code/24 Internet/update_servidor.sh  
chmod +x update servidor.sh  
./update_servidor.sh
```

para que a página seja atualizada a cada segundo. (Aperte CONTROL-C para parar esta atualização.)

Streaming de video

- Existem diversas soluções para oferecer *streaming* de video via web no Raspberry Pi, usando o *Raspberry Pi Camera Module* ou *webcams* USB:
 - <https://www.linux-projects.org/uv4l/>
 - https://github.com/silvanmelchior/RPi_Cam_Web_Interface
 - <https://motion-project.github.io/index.html>
 - <https://www.videolan.org/vlc/streaming.html>

Referências

- <https://weworkweplay.com/play/automatically-connect-a-raspberry-pi-to-a-wifi-network/>
- <https://curl.haxx.se/book.html>
- <https://www.hostinger.com.br/tutoriais/comando-curl-linux/>
- <https://www.matthuisman.nz/2019/01/download-google-drive-files-wget-curl.html>
- <http://worldtimeapi.org/>
- <https://www.dropbox.com/developers/documentation/http/documentation>
- <https://riptutorial.com/dropbox-api/example/1356/uploading-a-file-via-curl>
- <https://olivermarshall.net/how-to-upload-a-file-to-google-drive-from-the-command-line/>
- <https://www.shellhacks.com/telegram-api-send-message-personal-notification-bot/>

Referências

- <https://thepihut.com/blogs/raspberry-pi-tutorials/using-ifttt-with-the-raspberry-pi>
- <https://eureka.ykyuen.info/2014/07/30/submit-google-forms-by-curl-command/>
- <https://stackoverflow.com/questions/14722556/using-curl-to-send-email>
- <https://www.commandlinefu.com/commands/view/6716/send-email-with-curl-and-gmail>
- https://raspberrypi-projects.com/pi/software_utilities/email/ssmtp-to-send-emails
- <https://stackoverflow.com/questions/30351465/html-email-with-inline-attachments-and-non-inline-attachments>
- <https://en.wikipedia.org/wiki/MIME>
- <https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md>

Referências

- <https://raspberrypi.stackexchange.com/questions/64115/how-to-stop-apache-from-running-a-local-web-server-on-boot>
- https://www.w3schools.com/howto/howto_css_contact_form.asp
- <https://raspberrypi.stackexchange.com/questions/75248/apache-server-monitoring>
- <https://www.e-tinkers.com/2018/04/how-to-control-raspberry-pi-gpio-via-http-web-server/>
- <https://www.filipeflop.com/blog/streaming-com-raspberry-pi/>
- <https://medium.com/@gonzalovazquez/raspberry-pi-and-motioneye-setting-up-your-own-video-surveillance-95444e0faad>

Referências

- <https://stackoverflow.com/questions/49846400/raspberry-pi-use-vlc-to-stream-webcam-logitech-c920-h264-video-without-transcode>
- <https://raspberrypi.stackexchange.com/questions/4412/streaming-h264-with-logitech-c920>
- <https://raspberrypi-projects.com/pi/pi-hardware/raspberry-pi-camera/streaming-video-using-vlc-player>