

SISTEMAS EMBARCADOS

Desenvolvimento para sistemas
embarcados

Desenvolvimento para sistemas embarcados

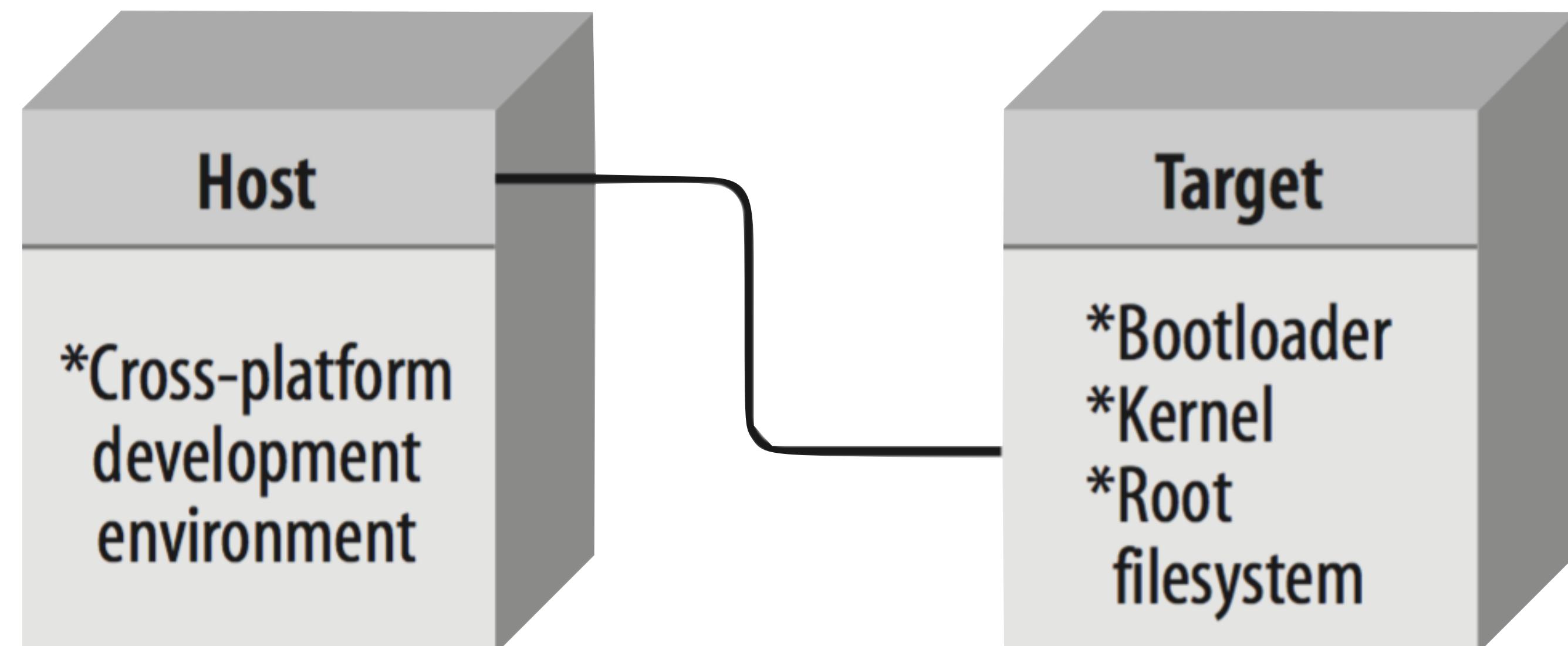
- Conceito de cross-platform
- Componentes e suas funções (host, target, interface de comunicação, etc)
- Processo de geração de imagens;
- Utilização de *makefiles*;

Host (PC)

- **Linux**
 - Debian
 - Fedora
 - OpenSuse
 - Red Hat Enterprise Linux
 - CentOS
 - Ubuntu
 - Slackware
 - Gentoo
- **Unix**
 - Solaris
 - OpenSolaris
 - FreeBSD
 - OpenBSD
- **Mac OS X**
- **Windows**

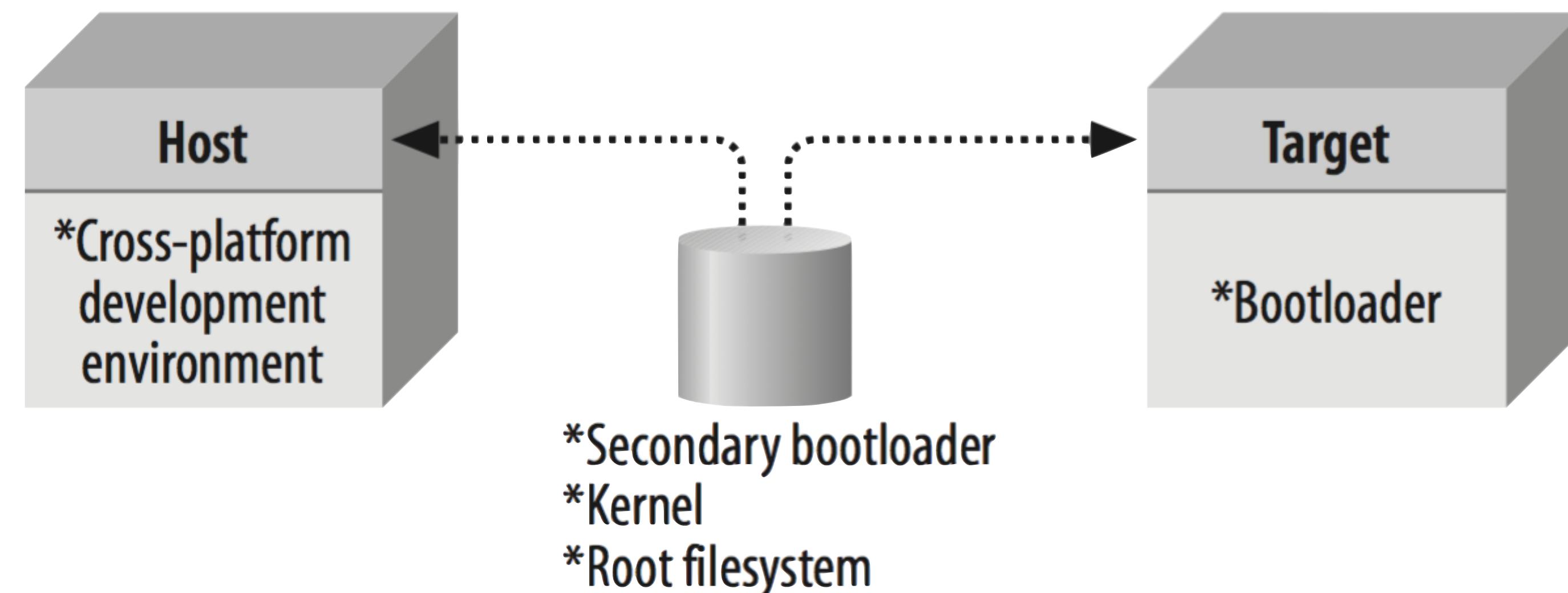
Configurações Host/Target

- **Conectado (Linked Setup)**
 - Cabo físico conectado (Serial, JTAG, Ethernet, USB) por onde trafega toda a comunicação.
 - É possível manter o sistema de arquivos do *target* montado no *host* (*NFS*).



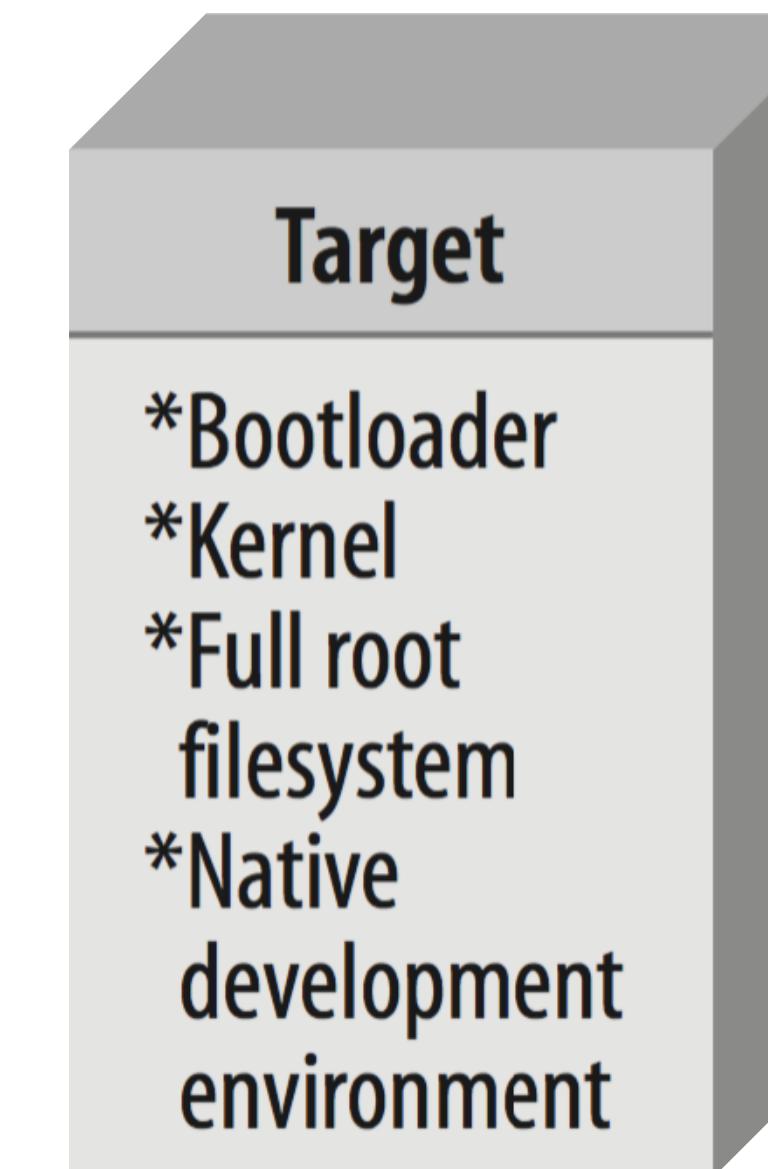
Configurações Host/Target

- **Media Removível (Removable Storage Setup)**
 - Não há conexão física entre os sistemas.
 - O target possui somente um *bootloader* mínimo permanente.



Configurações Host/Target

- **Independente (Standalone Setup)**
 - Todo o ambiente de desenvolvimento está no *target*
 - Similar a uma workstation mas tipicamente com menor poder computacional

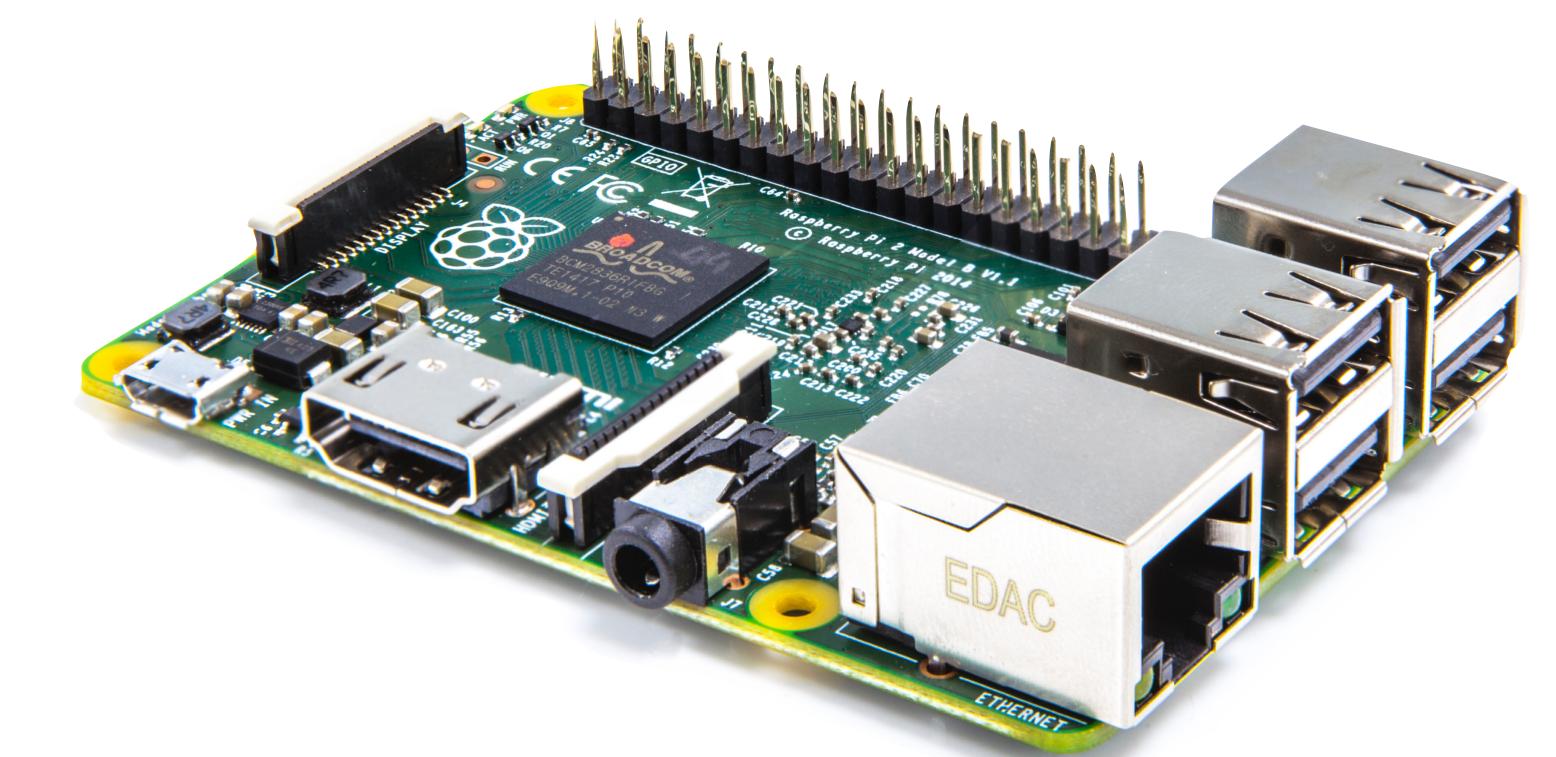


Configurações Host/Target - Debug

- **Serial Link**
- **Ethernet**
- **JTAG**

Desenvolvendo no Raspberry Pi

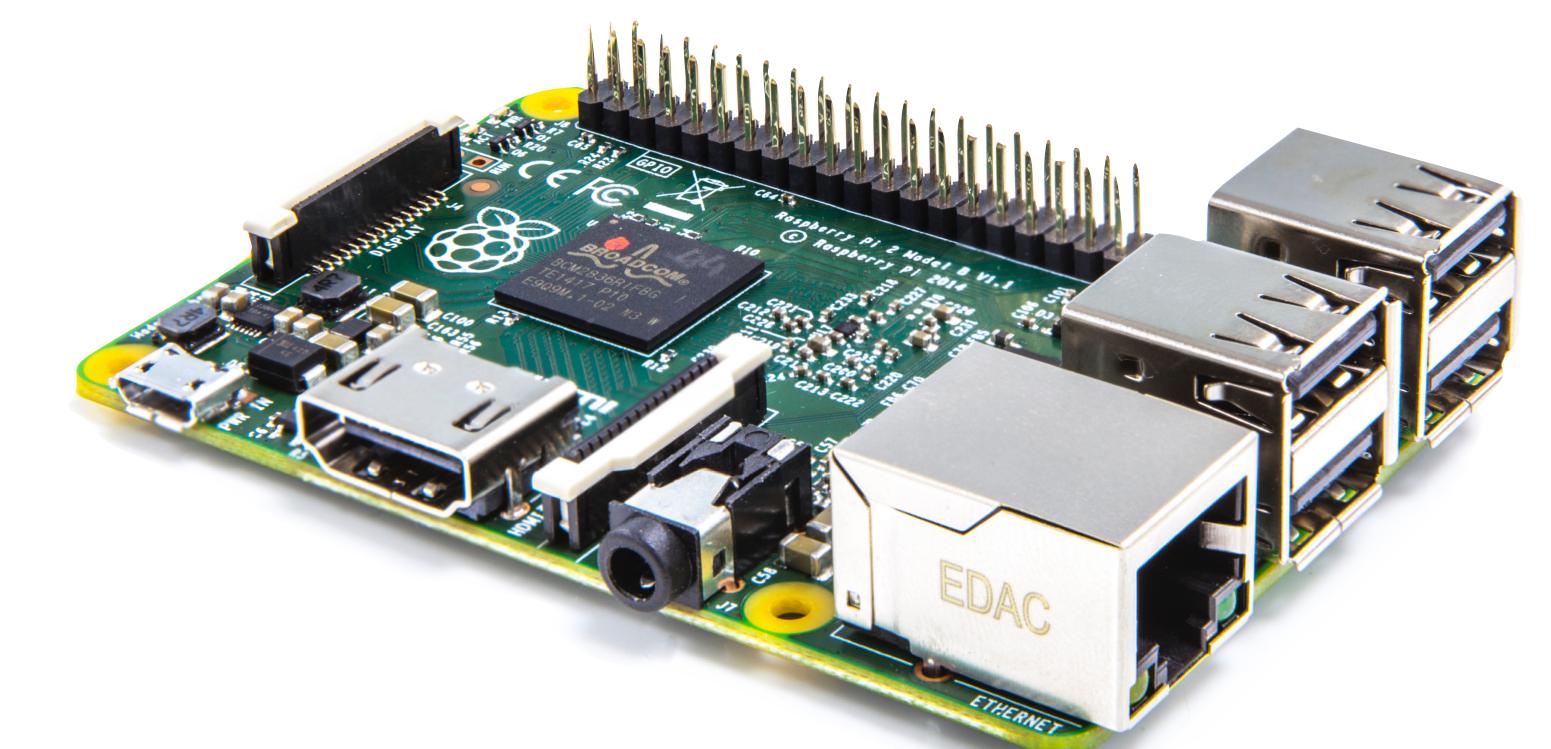
- Configuração Host/Target conectados via Ethernet
- Conexão via protocolo SSH (Secure Shell)
- Crosscompilar o software no *Host* e transferir para o *target* para teste.



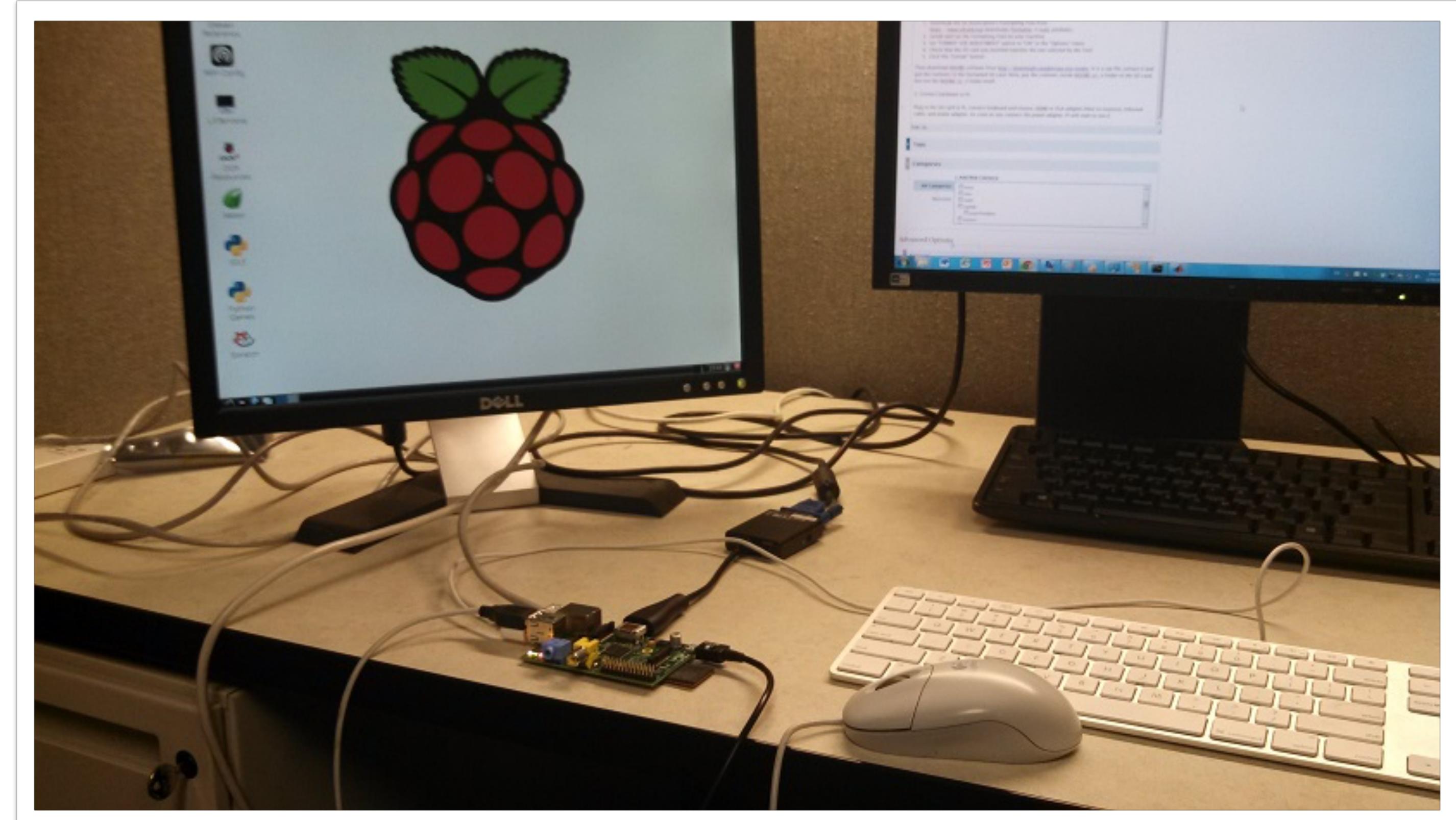
Desenvolvendo no Raspberry Pi

Target - Raspberry Pi 2 B

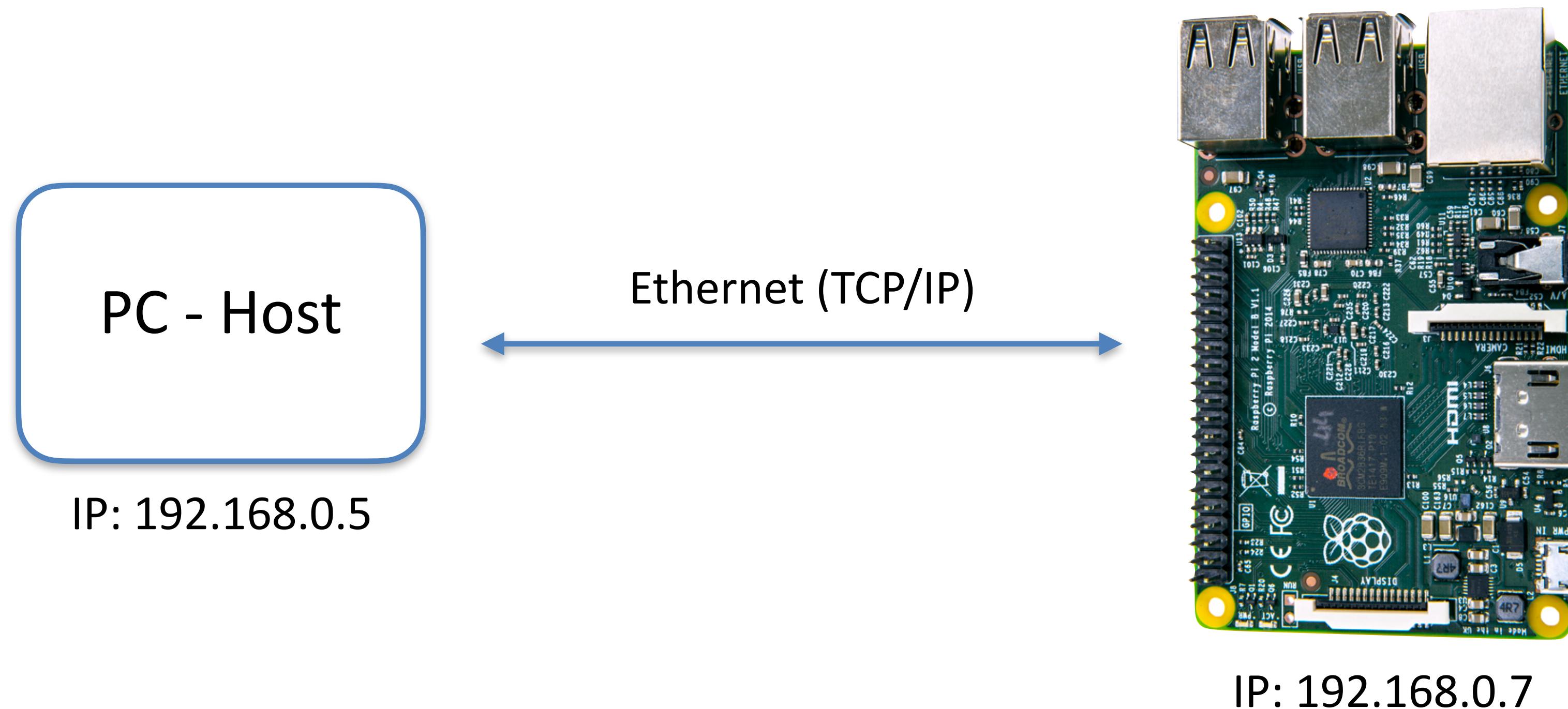
- Processador: Quad-Core ARM Cortex-A7
- Clock 900 MHz
- GPU: Dual Core VideoCore IV Multimedia Co-Processor (OpenGL ES 2.0), OpenVG, H.264 Decoder
- Memory 1GB LPDDR2
- OS: Linux / Windows (SD Card)
- Ethernet 10/100
- Video: HDMI (1.3 e 1.4)
- Áudio out
- USB 2.0 x 4
- GPIO (40 pin) (+3.3V e +5V)
- Camera and Display Connector



Desenvolvendo no Raspberry Pi



Desenvolvendo no Raspberry Pi



```
$ ssh <usuario>@<endereço ip>  
$ ssh pi@192.168.0.7
```

Desenvolvendo no Raspberry Pi

- Acesso via SSH

```
$ ssh <usuario>@<endereço_ip>
$ ssh pi@192.168.0.7
```

- Cópia de arquivos remotos via SSH

```
$ scp <caminho local>
<usuario>@<endereço_ip>:<caminho remoto>
```

```
$ scp arquivo1.txt aluno@192.168.0.7:/home/aluno
```

Ferramentas de Desenvolvimento

- GCC: Gnu C Compiler;
- O `gcc` foi um dos primeiros passos na criação de um sistema operacional livre;
- É por meio dele que os códigos fonte são criados em binários executáveis.

Ferramentas de Desenvolvimento

- *Cross-compiler no Linux*
 - Instalar as ferramentas essenciais (gcc, g++, git, etc):
 - `sudo apt-get install build-essential git`
 - Baixar a *Toolchain* do Raspbian disponível no Github:
 - `mkdir ~/rpi; cd rpi`
 - `git clone git://github.com/raspberrypi/tools.git`
 - `cd ~/rpi/tools/arm-bcm2708`

Ferramentas de Desenvolvimento

- *Cross-compiler no Linux*
 - Na pasta rpi/tools/arm-bcm2708 encontram-se os seguintes diretórios:
 1. **arm-bcm2708-linux-gnueabi**
 2. **arm-bcm2708hardfp-linux-gnueabi**
 3. **gcc-linaro-arm-linux-gnueabihf-raspbian**
 4. **gcc-linaro-arm-linux-gnueabihf-raspbian-x64**
 - Para Linux 32 bits utilizar o #3, para 64 bits utilizar o #4:

Ferramentas de Desenvolvimento

- *Cross-compiler no Linux*
 - Inserir o endereço da pasta com os binários do compilador desejado na variável de ambiente PATH do Linux
 - `cd ~/`
 - `nano .bashrc`
 - Adicionar a linha abaixo ao final do arquivo
 - `export PATH=$PATH:$HOME/rpi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin`
 - Carregar as novas configurações na seção de terminal
 - `source .bashrc`

Ferramentas de Desenvolvimento

- *Cross-compiler no Linux*
 - Testar o compilador
 - **arm-linux-gnueabihf-gcc -v**

Ferramentas de Desenvolvimento

- *Cross-compiler utilizando o Docker*
 - Uma outra opção ao invés de instalar o compilador nativo, é usar uma imagem Docker com as ferramentas pré-instaladas.
 - Neste caso, há uma imagem pronta disponível em:
 - <https://desertbot.io/blog/how-to-cross-compile-for-raspberry-pi>