

Implementação de um *Device Driver* para Controle de *Hardware Reconfigurável*

Orientado: Rodolfo Labiapari Mansur Guimarães,
Orientador: Otávio de Souza Martins Gomes,
Coorientador: Diego Mello da Silva.



Apresentação dia 11 de novembro de 2015

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Objetivos

Objetivo Geral

- **Pesquisa e Desenvolvimento** de um *Driver* em plataforma **GNU/Linux** para tornar possível a **comunicação com um dispositivo externo** ao computador:
 - No qual executa um algoritmo de criptografia simétrica.

Objetivo Específico

- Comunicar por meio do protocolo **USB**.
- Componentes de **código-fonte aberto** junto com **hardware re-configurável**.
- Descrever uma metodologia para o entendimento e desenvolvimento de um *Driver*.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

- 1 Algoritmos desenvolvidos em *hardware* **necessitam** de uma interface de controle para a comunicação com o usuário.
- 2 Os *Drivers* atuais são genéricos [1]:
 - Tornando suficientemente para serem extremamente complexos.
- 3 Documentação sobre desenvolvimento de um *Driver*.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Sumário

1 Objetivo

2 Justificativa

3 Metodologia

- Materiais
- Métodos

4 Referencial Teórico

- Materiais
- Protocolos

5 Desenvolvimento

- FPGA
- Linux
- Arduino

6 Conclusão

- Itens utilizados para a realização da pesquisa:
 - 1 **Dispositivo Lógico Programável:** FPGA;
 - 2 **Plataforma de Prototipagem:** Arduino;
 - 3 **Sistema Operacional:** GNU/Linux.

Sumário

1 Objetivo

2 Justificativa

3 Metodologia

- Materiais
- Métodos

4 Referencial Teórico

- Materiais
- Protocolos

5 Desenvolvimento

- FPGA
- Linux
- Arduino

6 Conclusão

Método - Software

- Pesquisa no livro **Linux Driver Driver 3º Edição** disponível gratuitamente para leitura e impressão¹ [1].
- Utilizou-se de *mailing lists* do sítio linux-usb.org para suprimir dúvidas.
- Documentação do Linux Kernel².
- Estudo de *Drivers* do Kernels.

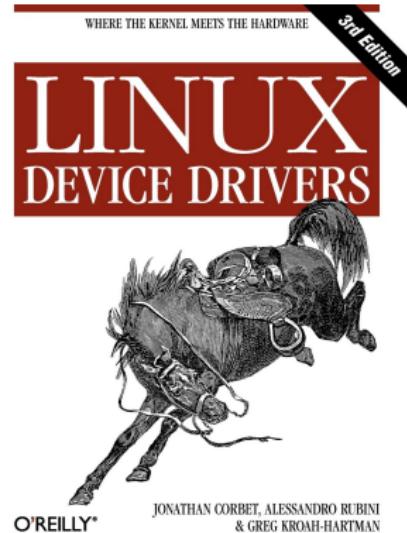


Figura 1: Linux Device Driver 3º Edição [1].

¹<https://www.makelinux.net/ldd3/>.

²<https://www.kernel.org/doc/Documentation/driver-model/>. Navigation icons

- Cursos introdutório sobre **Linguagem VHDL e Sintetização de Projetos em Hardware.**

1 Pesquisa sobre **desenvolvimento de Drivers**:

- Características principais;
- Tipos;
- Escrita;
- Compilação.

2 FPGA:

- Escrever procedimentos de envio e recebimento de dados;
- Máquina de estado controladora.

3 Interconectar as duas extremidades usando o **Arduino**.

Método - Estrutura do Projeto

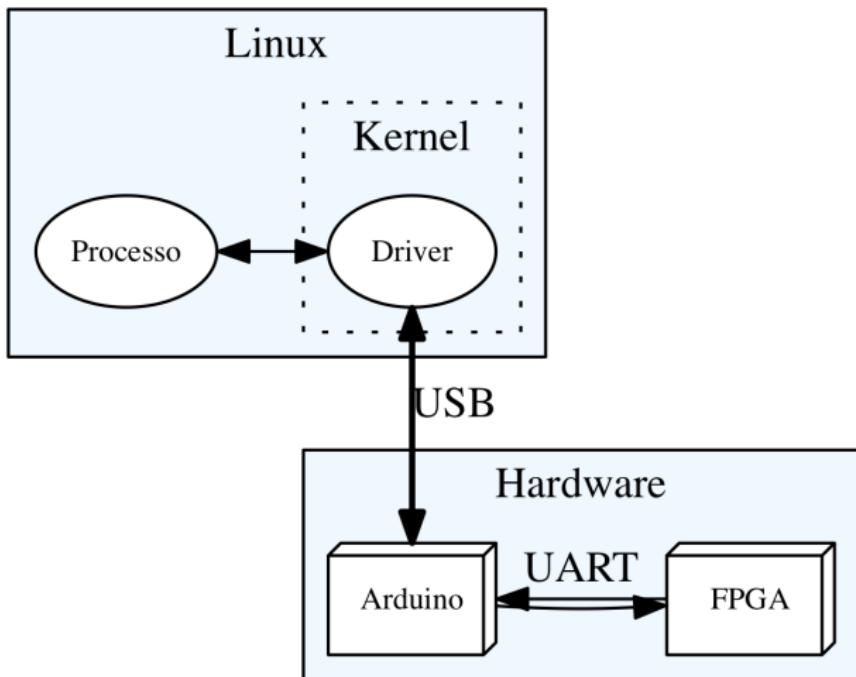


Figura 2: Representação do projeto numa visão geral.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

FPGA

- Tocci (2003) cita que os dispositivos lógicos programáveis são as “*maravilhas de flexibilidade de projeto*”.
- FPGA (*Field Programmable Gate Array*, ou seja, Arranjo de Portas Programável em Campo) consiste num **arranjo de blocos lógicos e canais de roteamento**.
- Significa que é capaz de alterar seus caminhos de dados/fluxos **habilitando/desabilitando módulos** [3].

FPGA - Roteamento de Blocos

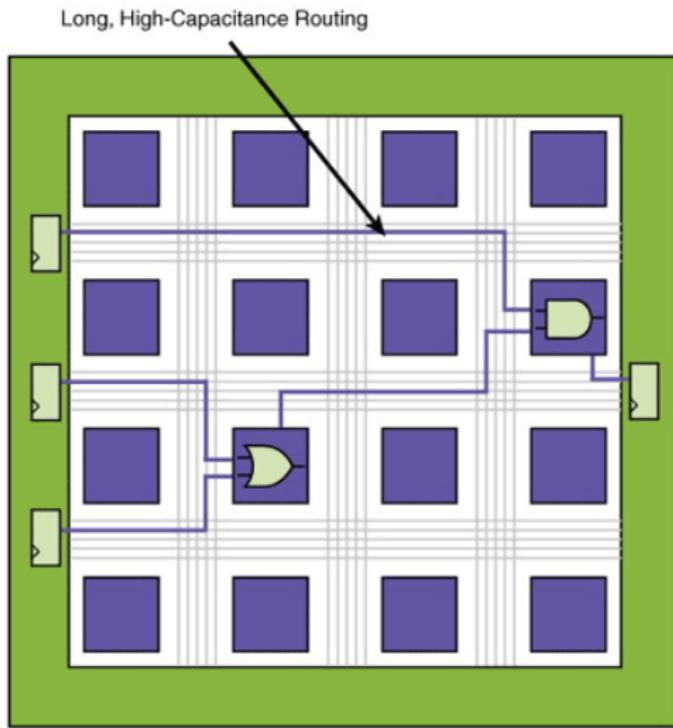


Figura 3: Exemplo de roteamento interno no FPGA.

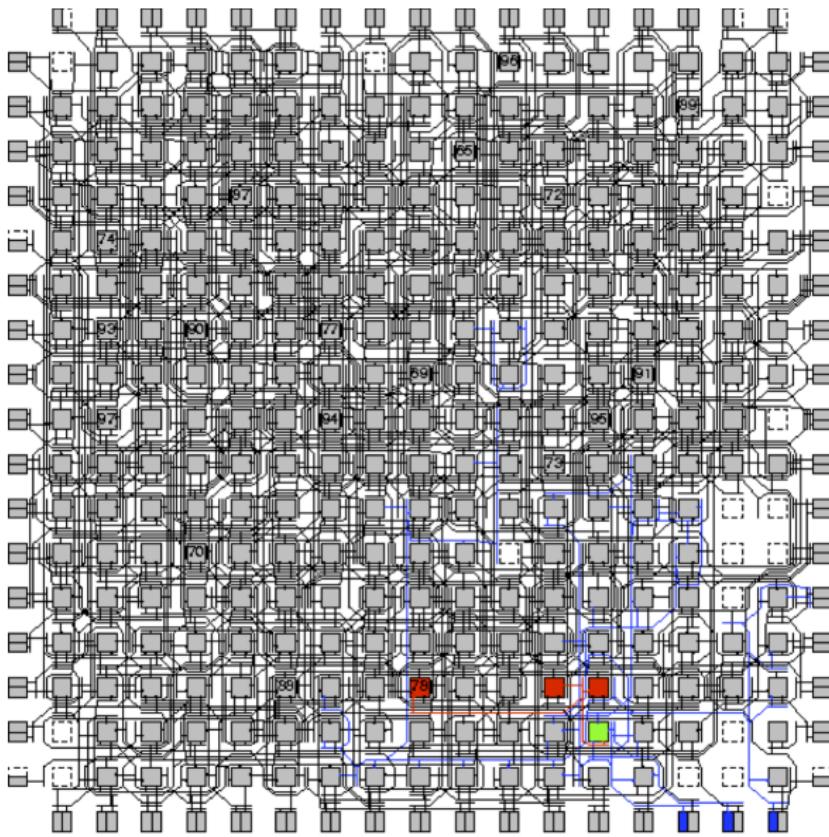


Figura 4: Exemplo de roteamento interno bastante complexo no FPGA..

FPGA - Placa

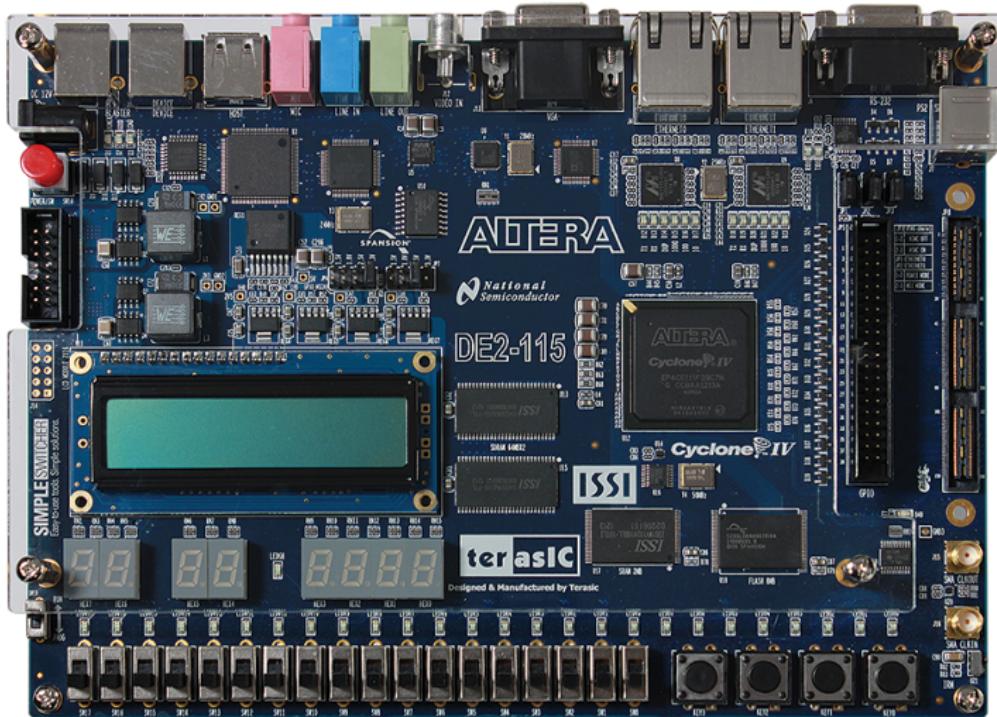


Figura 5: FPGA de Desenvolvimento Altera DE2-115.

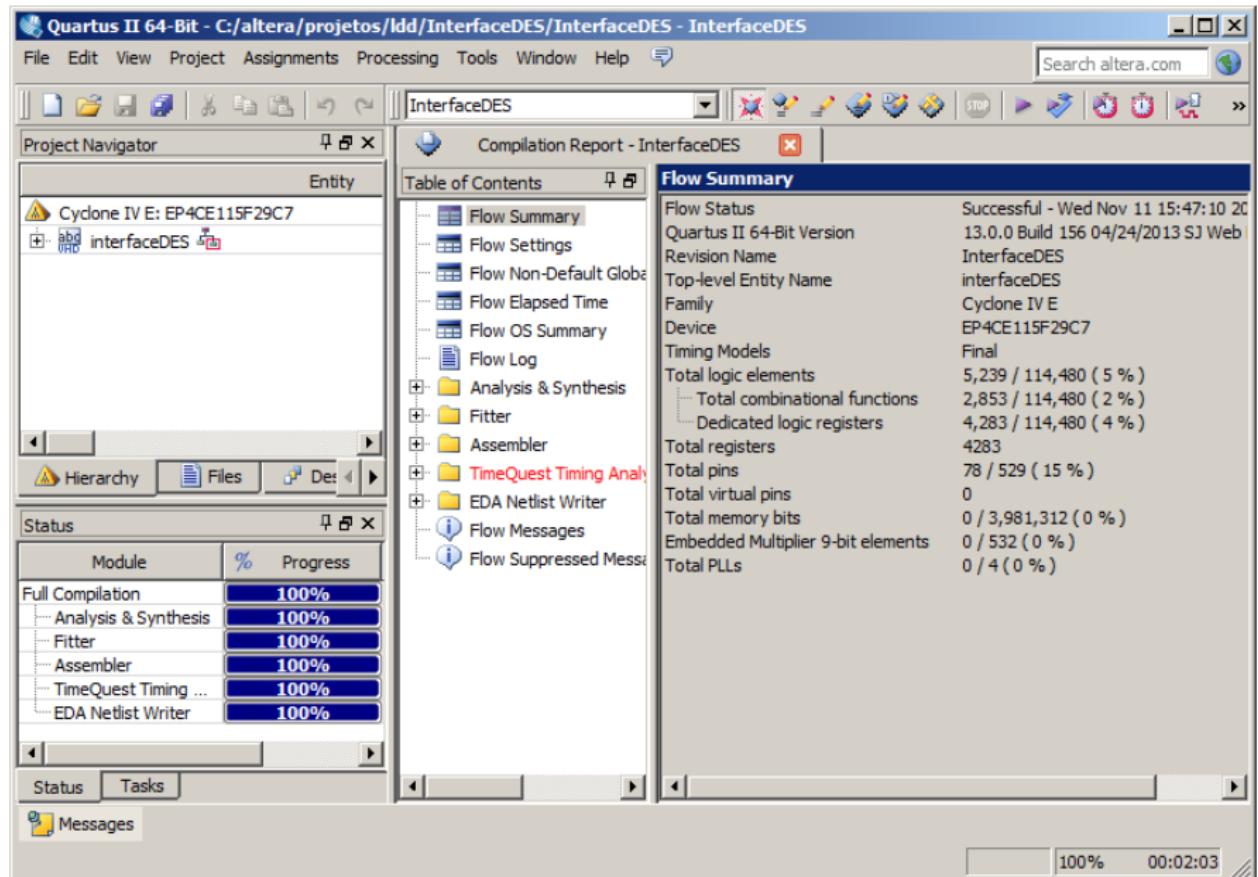


Figura 6: Altera Quartus II.

- Características importantes da placa utilizada:
 - Altera DE2-115 utilizada para ensino e desenvolvimento.
 - Quantidade de elementos lógicos: 114.480;
 - IDE³ Quartus II;
 - Utilizou-se a linguagem de descrição de *hardware* VHDL;
 - *VHSIC Hardware Description Language*, ou seja, uma linguagem de descrição de hardware VHSIC⁴.
 - Clock de 50Mhz;
 - **Várias interfaces** para acomodar as mais diversas aplicações.
- São muito utilizados para **desenvolver protótipos e realização de testes** antes da fabricação em massa [4].

³Integrated Development Environment ou Ambiente de Desenvolvimento Integrado.

⁴Very High Speed Integrated Circuits.

Arduino

- Plataforma de prototipagem eletrônica aberta baseada em micro-controlador.
- Utiliza linguagem de programação Arduino além da IDE Arduino.
- Facilidades:
 - **Reduz** a complexidade da montagem da infraestrutura do projeto;
 - Possui vários meios de obter informações de uso e sem custo;
 - É uma plataforma de **computação física**⁵.

⁵Sistemas digitais podem mensurar variáveis no ambiente físico e tomar decisões sobre tal.

Arduino Mega 2560



Figura 7: Arduino Mega 2560.

The screenshot shows the Arduino IDE 1.6.5 interface. The title bar reads "Blink | Arduino 1.6.5". The toolbar contains icons for file operations (New, Open, Save, Upload, Download) and a search function. The main code editor window displays the "Blink" sketch. The code is as follows:

```
/*
Blink
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

The status bar at the bottom indicates "9" and "Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/cu.Bluetooth-Incoming-Port".

Figura 8: IDE Arduino 1.6.5.

- Características do Arduino utilizado no trabalho:
 - **Arduino Mega 2560** com microcontrolador ATmega2560;
 - Possui 54 portas de entrada e saída Digital:
 - Sendo 15 podendo ser utilizadas para saída PWM (*Pulse-Width Modulation*).
 - Outras 16 portas para entrada analógica;
 - Frequência de *Clock* de 16MHz.

Linux

- Linux é um termo utilizado para referir sistemas operacionais que utilizam o **kernel Linux**.
- O sistema operacional como um todo tem como nome **GNU/Linux**.
- Linux Kernel foi desenvolvido por Linus Torvalds em 1991.
- Reimplementação reelaborada do MINIX⁶ obedecendo o padrão POSIX [5].
- Hoje é desenvolvido/estudado em projetos colaborativos de código-fonte aberto com principalmente:
 - Programadores individuais/Entusiastas;
 - Colaborações de grandes empresas como *IBM, Sun Microsystems, HP, Oracle, Google*, entre outras.

⁶Baseado no UNIX.

- Linux é um kernel **monolítico** no qual os *Drivers* podem:
 - Executar com acesso total ao *hardware*;
 - Configurados como módulos e carregados/descarregados **enquanto o sistema está ligado**.
- Executa em arquiteturas como: *Intel, StrongARM, PowerPC, Alpha*, etc. além de embarcados.
- Hoje, é possível encontrar distribuições Linux com coleções de *software* (por exemplo o GNU) pronto para o uso [6].

A distribuição Linux utilizada para a realização do desenvolvimento:

- **Linux Mint 17.2** com codi-
nome Rafaela, 64-bits;



Figura 9: Logo Linux Mint.

Sendo executado na máquina virtual:

- **Oracle VM VirtualBox 5.0.8** instalada no OS X El Capitan.



Figura 10: Logo Oracle VM
VirtualBox.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - **Protocolos**
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Comunicação Serial

- Utilizadas a mais de 40 anos para a comunicação entre dispositivos [5].
- São representadas por arquivos de dispositivo no diretório /dev.
- Foram utilizados dois protocolos de comunicação serial sendo eles:
 - UART;
 - USB.

Protocolos - UART

- **Universal Asynchronous Receiver/Transmitter.**

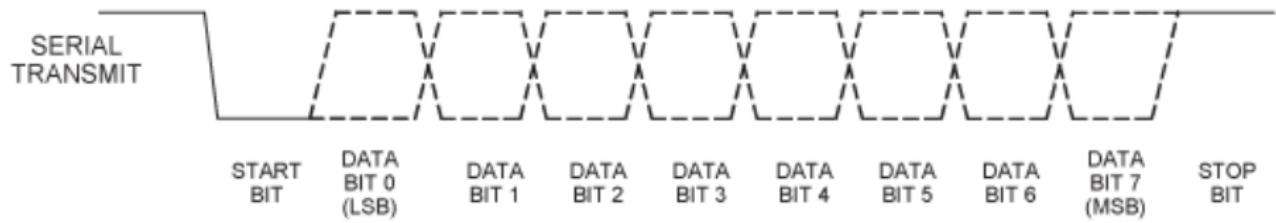


Figura 11: Protocolo de comunicação UART.

- **Atributos:** *idVendor*, *idProduct*, Números de *Interfaces*, *Endpoints*...
- **Um dispositivo USB possui:** *Configurations*, *Interfaces* e *Endpoints*.
- Modo de envio de pacotes **por meio (ou não)** de **URB⁷**:
 - **Tipo:** Interrupt, Bulk, Control, Isochronous;
 - **Operações:** Criar, destruir, enviar, callback handler, cancelamento.
- Entre muitas outras características.

⁷ **USB Request Block.**

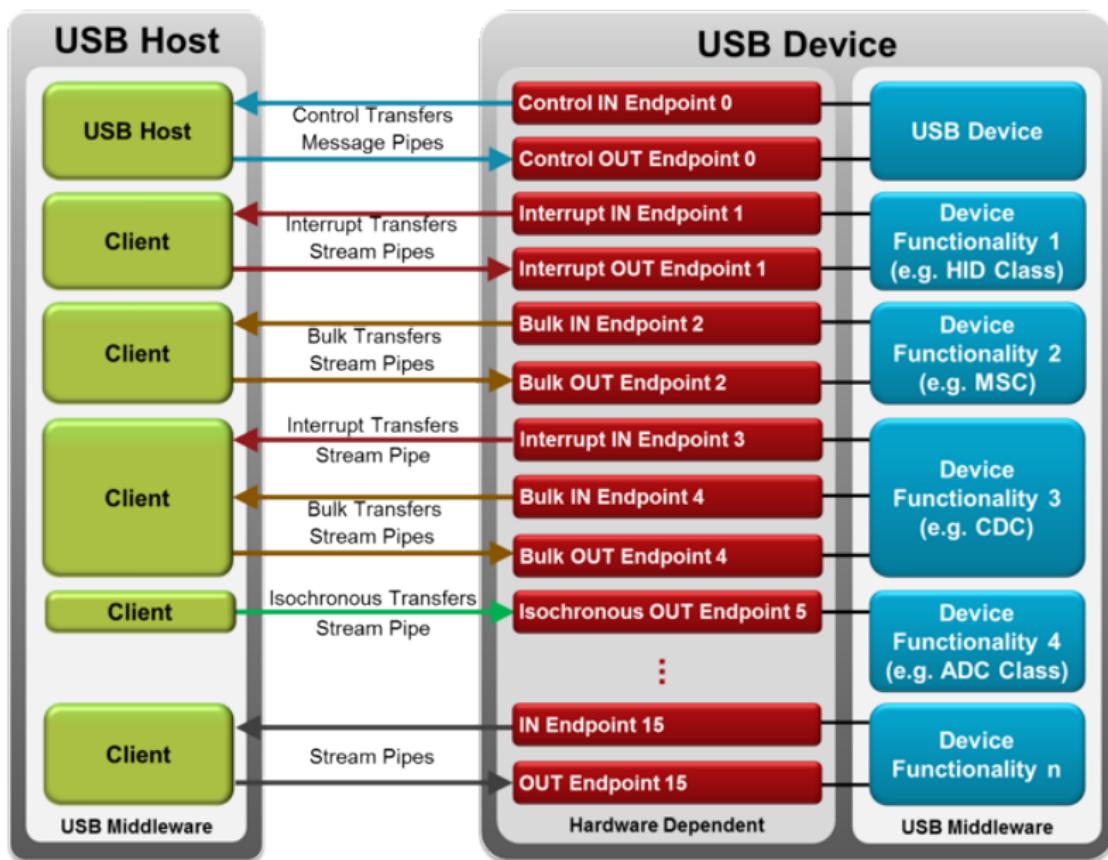


Figura 12: Exemplo de comunicação USB.

■ Vantagens:

- *Universal Serial Bus* é um sistema de **interconexão de periféricos genéricos**.
- **Cabos e conectores padronizados** além de vários **adaptadores** para extender suas funcionalidades.
- É um excelente sistema e que acredita-se durar ainda por vários anos [5].
- O Linux tem um **amplo e sólido suporte** ao USB.

■ Desvantagem:

- Protocolo complexo.

- Abreviatura de **teletypewriter**.
- Interligava um terminal (físico ou virtual) a uma máquina Unix.
- Meio de nomear qualquer dispositivo de porta serial [1]:
 - **Físicos:**
 - Portas Seriais;
 - Conversores USB/serial;
 - *Modems* que necessitam de um processamento especial.
 - **Virtuais:**
 - Consoles virtuais utilizados para fazer login em um computador por meio de uma conexão de rede;
 - Entre outros.
- Vive **sob** o *Driver de Caractere* e oferece **recursos de interface**:
 - Controla o fluxo de dados;
 - Formato dos dados.
- Foco na **transmissão e recebimento de dados com o hardware**:
 - Ao invés da interação com o espaço do usuário.

Driver TTY

- O **Driver TTY**:

- Não comunica diretamente com o TTY *Line Discipline*;
- Formata os dados pro envio ao *hardware* e também os recebe.

- **TTY Line Discipline**:

- Formatar os dados recebidos pelo usuário ou o *hardware*, de forma específica (PPP, Bluetooth por exemplo).

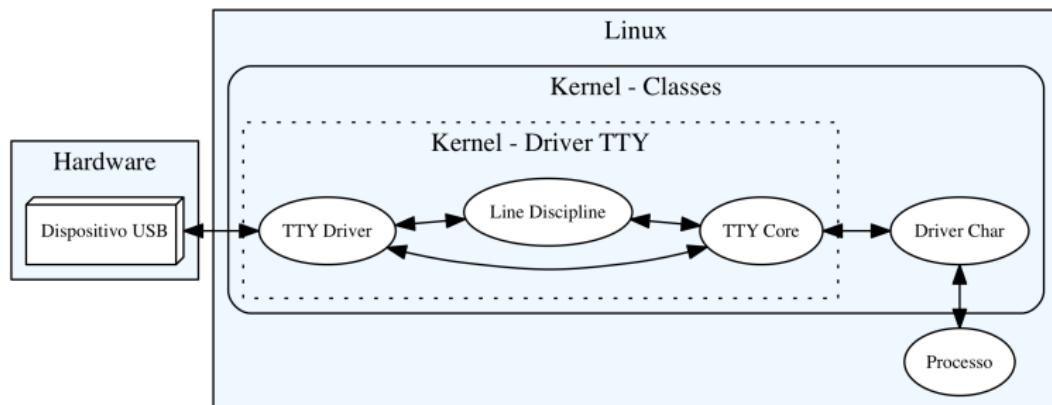


Figura 13: Comunicação utilizando TTY.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - **FPGA**
 - Linux
 - Arduino
- 6 Conclusão

FPGA

- Síntese do algoritmo **3DES**, no qual utiliza:
 - Três chaves de 64-bits;
 - Blocos de texto de 64-bits.
- Escrita do método de **envio e recebimento de informações** utilizando protocolo **UART**⁸:

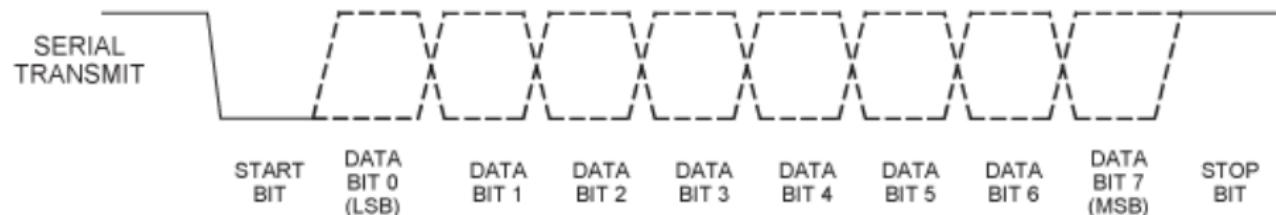


Figura 14: Protocolo de comunicação UART.

⁸Universal Asynchronous Receiver/Transmitter

Desenvolvimento - FPGA - Máquina de Estados Controladora

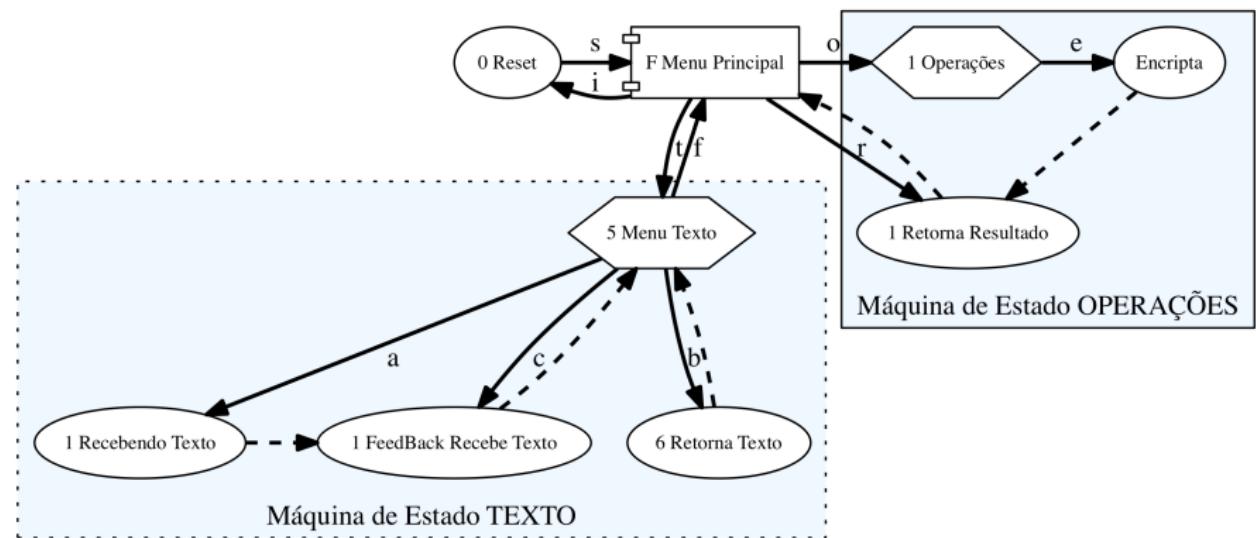


Figura 15: Exemplo da máquina de Estados do FPGA.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Linux

- Percebeu-se que existe **3 classes** de um *Driver* genérico:
 - **Bloco:** 512 bytes ou maiores, em potência de 2, mapeado em /dev;
 - **Interface de Rede:** tratamento de pacote de redes, mapeado como eth0;
 - **Caractere:** mapeado em /dev.
- De todos, o *Driver* pertencentes à classe Caracteres mostrou-se mais **simples e eficaz** para a comunicação USB.

- Com pesquisas mais profundas, descobriu-se que:
 - Unir a comunicação **TTY** (*Driver Char*) com bibliotecas do Kernel específicas para USB torna a escrita mais facilitada;
- Utilizado o TTY com a biblioteca USB Serial:
 - Utilizar **Macros** tornando o processo automatizado;
 - O **desenvolvimento** é facilitado;
 - E consequentemente a **compreensão**.

Comunicação utilizando TTY

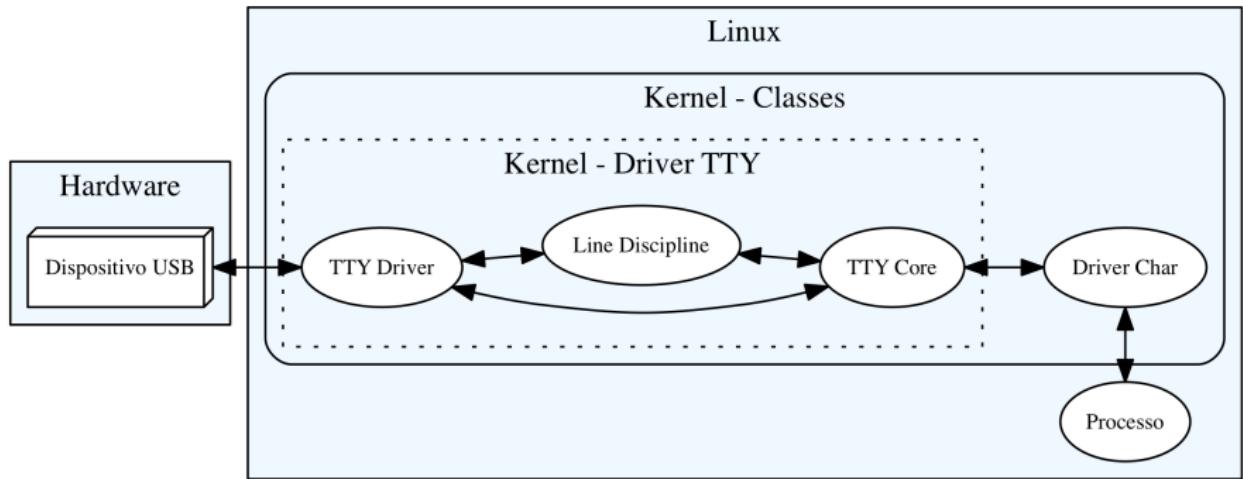


Figura 16: Comunicação utilizando TTY.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Arduino

- **Função:** Interconectar o FPGA com o *Driver*.
 - 1 O FPGA não tem interface ‘amigável’ para o usuário;
 - 2 E o *Driver* só realiza a comunicação entre o espaço do usuário com o *hardware*.
- Possui uma máquina de estados no qual:
 - **Controla** as informações a serem exibidas para o usuário;
 - **Instrui** quais passos podem ser executados nos momentos certos;
 - **Armazena** as informações para operação de forma organizada.

Desenvolvimento - Estrutura do Projeto

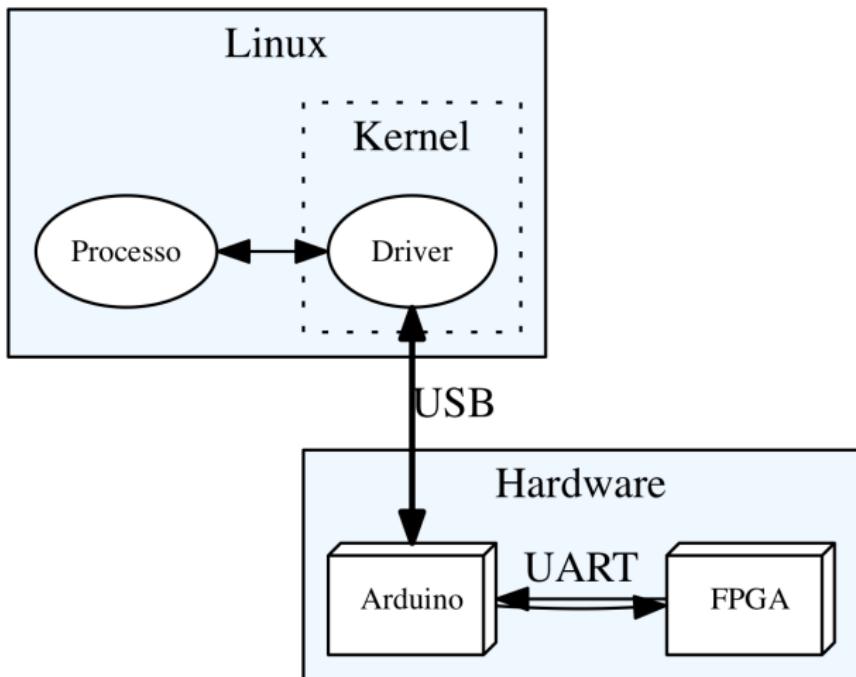


Figura 17: Representação do projeto numa visão geral.

Sumário

- 1 Objetivo
- 2 Justificativa
- 3 Metodologia
 - Materiais
 - Métodos
- 4 Referencial Teórico
 - Materiais
 - Protocolos
- 5 Desenvolvimento
 - FPGA
 - Linux
 - Arduino
- 6 Conclusão

Tarefas Cumpridas

1 Driver:

- Concede a comunicação com o Arduino via USB por meio de funções de manipulação de arquivos situados no diretório /dev.

2 Arduino:

- Realiza seu papel de intermediário entre as extremidades:
 - Tanto com o FPGA quanto com o *Driver*.
 - Exibe as informações corretamente.

3 FPGA:

- É possível realizar a encriptação de **apenas 1 bloco de 64-bits por vez**:
 - Sendo este alterado manualmente pelo usuário.

Exemplo Prático

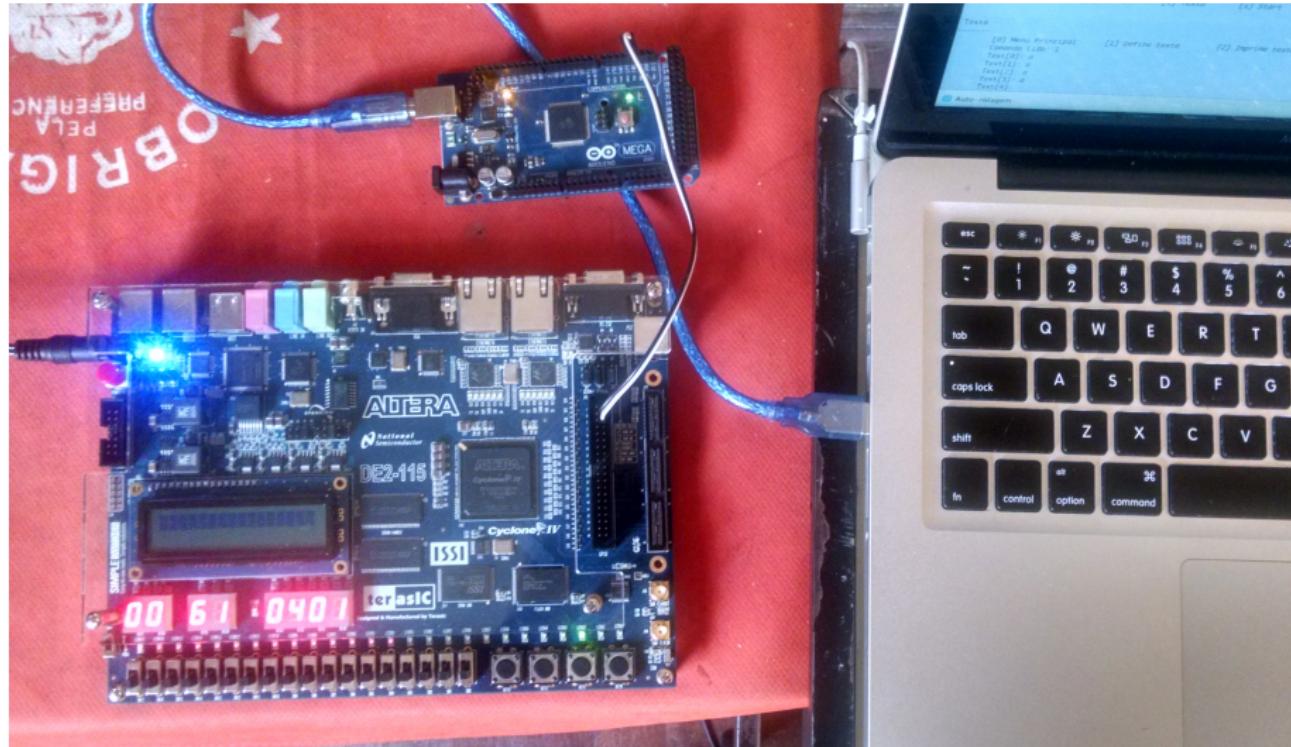


Figura 18: Foto real do projeto em atuação.

Tarefas Pendentes

- 1 Além da **encriptação**, adicionar o algoritmo que realiza a **decodificação**:
 - Adicionar os procedimentos pra que a decriptação seja possível.
- 2 Atualmente encripta-se somente 1 bloco por vez (alterado manualmente pelo usuário):
 - Permitir que o envie o texto do tamanho que quiser desde que:
 - Seu limite seja a capacidade de armazenamento interno do Arduino.
- 3 Desenvolver um conhecimento maior para aprimorar o *Driver TTY*.

Referências I

-  CORBET, J.; RUBINI, A.; KROAH-HARTMAN, G.
Linux device drivers.
"O'Reilly Media, Inc.", 2005.
-  TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L.
Sistemas digitais: princípios e aplicações.
Prentice Hall, 2003.
v. 8.
-  MOREIRA, V. R.; CARDOSO, F. A.; ARANTES, D. S.
Plataforma reconfigurável para ensino e pesquisa em laboratório
de sistemas digitais a distância.
In: .
c2008.
v. 1.
p. 542–551.

Referências II

-  SKLIAROVA, I.; FERRARI, A. B.
Introdução à computação reconfigurável.
Electrónica e Telecomunicações, v. 4, n. 1,
p. 103–119, 2003.
-  NEMETH, E.; HEIN, T. R.; SNYDER, G.
Manual completo do linux: guia do administrador.
2004.
-  CAMPOS., A.
O que é linux. br-linux., Março 2006.
Disponível em [j<http://br-linux.org/faq-linux>](http://br-linux.org/faq-linux). Acessado dia
22/10/2015.

Implementação de um *Device Driver* para Controle de *Hardware Reconfigurável*

Orientado: Rodolfo Labiapari Mansur Guimarães,
Orientador: Otávio de Souza Martins Gomes,
Coorientador: Diego Mello da Silva.



Apresentação dia 11 de novembro de 2015

- Alterar o funcionamento da comunicação USB para que tenha uma via só para envio de controle e outra só para envio de dados⁹.
- Adicionando o protocolo USB no FPGA:
 - Mensagens de informações ao usuário deverão ser trabalhadas no próprio *Driver* eliminando o dispositivo intermediário¹⁰; ou
 - Criação de um *software* a nível de usuário para todo o controle do *hardware*.

⁹Se possível

¹⁰Hipótese.

- 1 Algoritmo 3DES de criptografia simétrico desenvolvido em hardware onde:
 - 3 chaves de 64 bits (o mesmo que 8 caracteres ASCII de 8 bits);
 - Texto formado em blocos de 64 bits completos.
- 2 Algoritmos de envio e recebimento de dados através de UART.
 - $Clock_{FPGA} = 50.000.000$ $Clock_{requerido} = 9.600$
 - Cada intervalo teria: $\frac{50.000.000}{9.600} = 5.208,33333$ clocks do $Clock_{FPGA}$
 - Ou seja, a cada 5.208,33333 pulso de clock de 50Mhz, deve pular 1 do *NovoClock_{FPGA}*.
 - Erro: $\frac{|Clk_{Experi} - Clk_{Correto}|}{Clk_{Correto}} = \frac{0,333}{5.208,0} = 0,06400409626\%$

- 1 Algoritmo 3DES de criptografia simétrico desenvolvido em hardware onde:
 - 3 chaves de 64 bits (o mesmo que 8 caracteres ASCII de 8 bits);
 - Texto formado em blocos de 64 bits completos.
- 2 Algoritmos de envio e recebimento de dados através de UART.
 - $Clock_{FPGA} = 50.000.000$ $Clock_{requerido} = 9.600$
 - Cada intervalo teria: $\frac{50.000.000}{9.600} = 5.208,33333$ clocks do $Clock_{FPGA}$
 - Ou seja, a cada 5.208,33333 pulso de clock de 50Mhz, deve pular 1 do *NovoClock_{FPGA}*.
 - Erro: $\frac{|Clk_{Experi} - Clk_{Correto}|}{Clk_{Correto}} = \frac{0,333}{5.208,0} = 0,06400409626\%$
- 3 Máquina de estados controladora do sistema em hardware.

- Então, o ‘novo’ driver possui:
 - Macro de inicialização (`init()`, `exit()`, `probe()`, `disconnect()`);
 - Funções de envio de pacotes;
 - Funções como `open()` e `close()` para a inicialização e término da comunicação;
 - Uma tabela de devices para possíveis conexões.
- E deixou de existir:
 - `Init();`
 - `Exit();`
 - `Probe();`
 - `Disconnect()...`

Como é o envio de um URB?

- 1 No driver, associa um específico endpoint a um específico device.
- 2 Envia do driver para o USB Core
- 3 Envia do USB Core para um específico driver Host Controlador USB.
- 4 Este driver Host Controlador processa essa informação e envia para o device.
- 5 Quando o envio é completado, o driver Controlador Host notifica o driver do desenvolvedor.

Major e Minor Numbers

- Drivers de Char possuem dois números como parte de suas características.
 - **Major:** Identifica que o driver está associado com o device.
 - **Minor:**
- Por exemplo /dev/null e /dev/zero:
 - Ambos tem o número Major igual;
 - Ele é utilizado para associar o driver ao dispositivo;
 - Entretanto, possuem o número Minor diferentes.
 - É usado pelo kernel para determinar qual device específico ele está referenciando.
 - ...

Device

- Endpoints:
 - Control, Interrupt, Bulk e Isochronous.
- Interfaces:
 - Mouse, Teclado possuem somente 1 via;
 - Podem ter configurações diferentes;
 - Áudio em Linux, por exemplo teria 2 interfaces (Áudio e Controle).
- Configurations:
 - Em Linux, não é comum utilizar mais que 1 configuração no dispositivo.
- Ou seja:
 - 1 Devices usually have one or more configurations.
 - 2 Configurations often have one or more interfaces.
 - 3 Interfaces usually have one or more settings.
 - 4 Interfaces have zero or more endpoints.

Linguagens de Programação - VHDL

- **VHSIC (Very High Speed Integrated Circuits) Hardware Description Language.**

```
1 — importa std_logic da IEEE library
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4
5 — Declara uma entidade
6 entity ANDGATE is
7     port (
8         IN1 : in std_logic;
9         IN2 : in std_logic;
10        OUT1: out std_logic);
11 end ANDGATE;
12 architecture RTL of ANDGATE is
13
14 begin
15    OUT1 <= IN1 and IN2;
16 end RTL;
```

Linguagens de Programação - Arduino

■ Linguagem Arduino.

```
1 // the setup function runs once when you press reset or power the board
2 void setup() {
3     // initialize digital pin 13 as an output.
4     pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)
10    delay(1000);             // wait for a second
11    digitalWrite(13, LOW);   // turn the LED off by making the voltage low
12    delay(1000);             // wait for a second
13 }
```

Compilação do Driver

```
1 obj-m := usbSerial.o
2
3 KERNELDIR ?= /lib/modules/$(shell uname -r)/build
4 PWD      := $(shell pwd)
5
6 all:
7     rm -f *.o *~ core .depend *.cmd *.ko *.mod.c *.symvers *.order
8     #sudo /usr/sbin/ntpdate ntp.on.br time.nist.gov pool.ntp.org
9
10    # Compilando os arquivos atraves do kernel
11    $(MAKE) -C $(KERNELDIR) M=$(PWD)
12
13    # Copiando para a pasta temporaria
14    sudo cp usbSerial.ko /tmp/
15
16    sudo rmmod usbSerial          # Removendo o driver antigo
17
18    sudo insmod /tmp/usbSerial.ko # Adicionando o novo driver
```

Implementação de um *Device Driver* para Controle de Hardware Reconfigurável

Orientado: Rodolfo Labiapari Mansur Guimarães,
Orientador: Otávio de Souza Martins Gomes,
Coorientador: Diego Mello da Silva.



Apresentação dia 11 de novembro de 2015