

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 9 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

- Desenvolver um **driver em plataforma GNU/Linux** para tornar possível a **comunicação com um dispositivo externo** ao computador.

que execute um algoritmo de cripto
metodologia

Objetivo Específico

- Tornar possível a conexão de um *algoritmo de criptografia desenvolvido em hardware* **com** o espaço de usuário por meio de um driver no Kernel Linunx.
- Protocolo **USB**.
- Componentes de **código-fonte aberto** e com **hardware reconfigurável**.

Sumário

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

Justificativas

- 1 Algoritmos desenvolvidos em hardware **necessitam** de uma interface de controle para a comunicação com o usuário.
- 2 Desenvolver um driver simples e de fácil entendimento.
- 3 Criar uma metodologia para 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

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 - Estrutura do Projeto

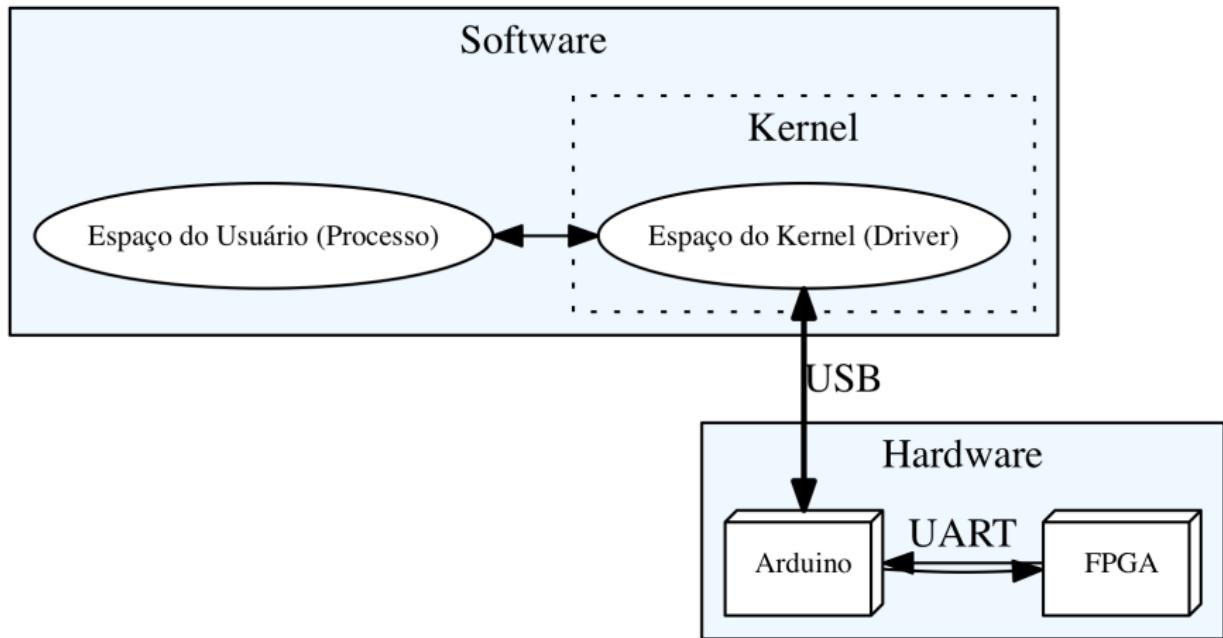


Figura: Representação do projeto numa visão geral.

- 1 Obtenção do **algoritmo de criptografia** desenvolvido em pesquisa aplicada.
- 2 Pesquisa sobre **desenvolvimento de drivers**.
- 3 Utilização do Arduino para **interconectar as duas extremidades**.

Método - Software

- Pesquisa por meio das informações contidas no livro **Linux Device Driver** 3º Edição disponível gratuitamente para leitura e impressão em <http://www.makelinux.net/ldd3/>.
- Utilizou-se de ***mailing lists*** do sítio linux-usb.org para suprimir dúvidas.
- Estudo de drivers nos kernels atuais.

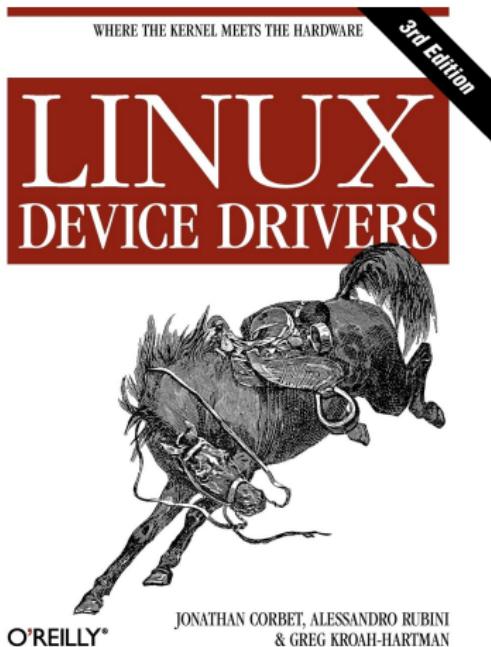


Figura: Linux Device Driver 3º Edição.

- Cursos introdutório sobre **linguagem VHDL e sintetização de projetos em hardware.**
- Em seguida, com a pesquisa aplicada, aprimorou-se o desenvolvimento de projetos de maior porte e mais complexos.

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 [1] cita que os dispositivos lógicos programáveis são as “*márvilhas 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** [2].

FPGA - Roteamento de Blocos

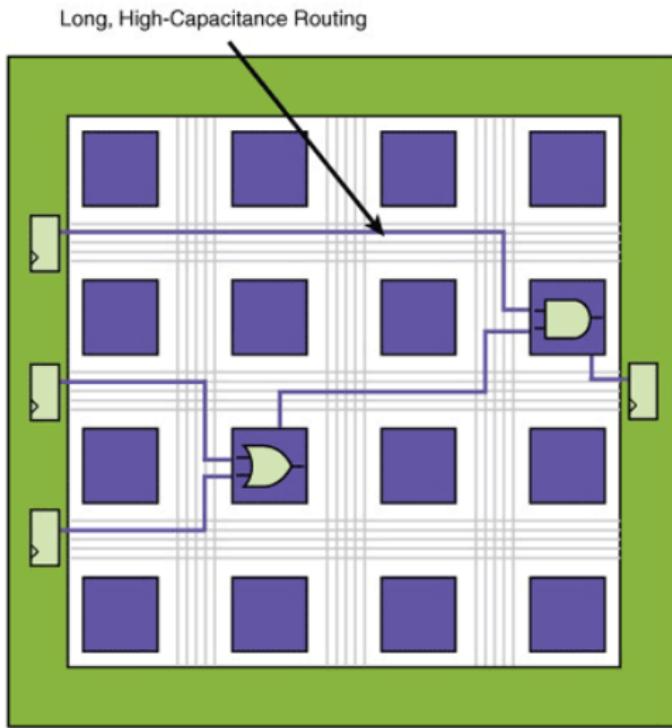


Figura: Exemplo de roteamento interno no FPGA.

FPGA - Roteamento de Blocos

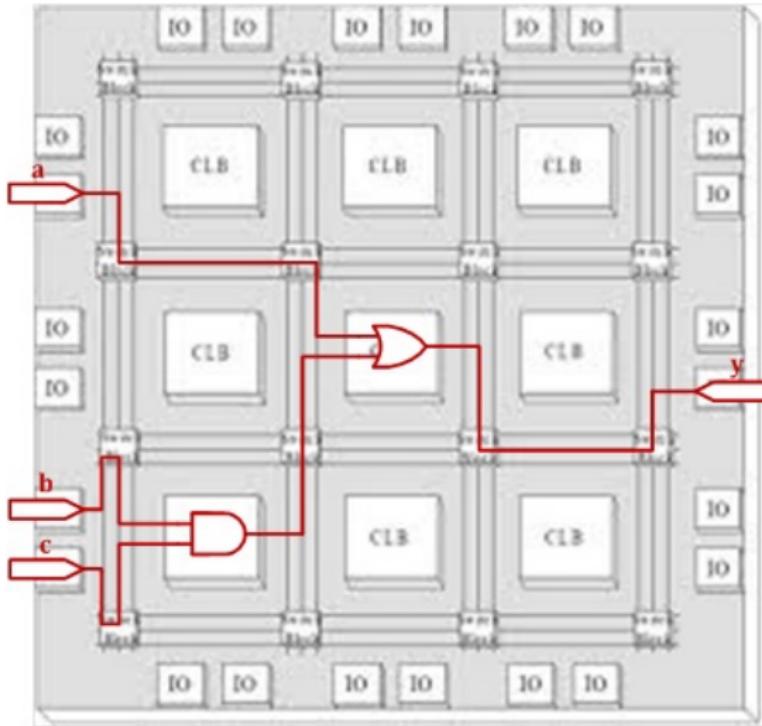


Figura: Outro exemplo de roteamento interno no FPGA.

FPGA - Roteamento de Blocos

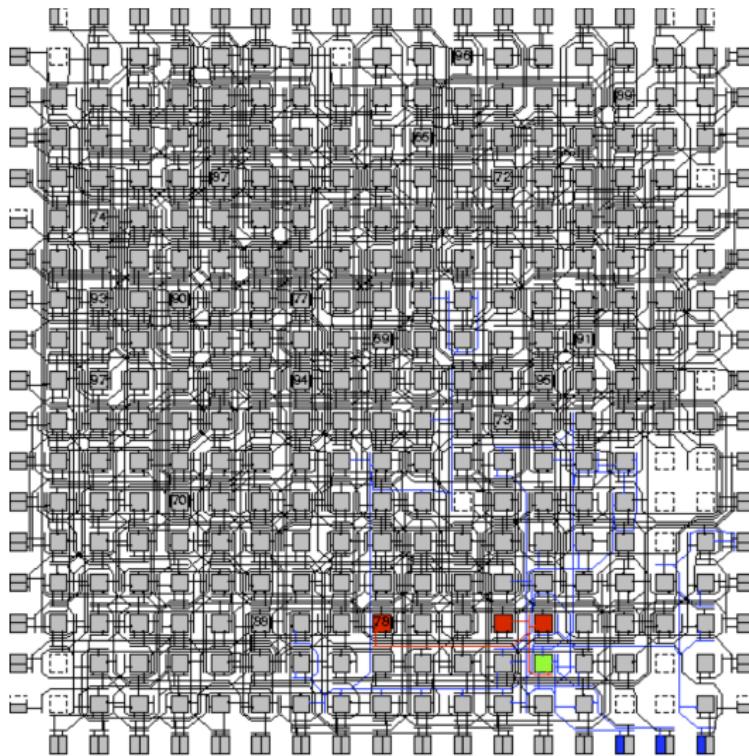


Figura: Exemplo de roteamento interno bastante complexo no FPGA..

FPGA - Placa

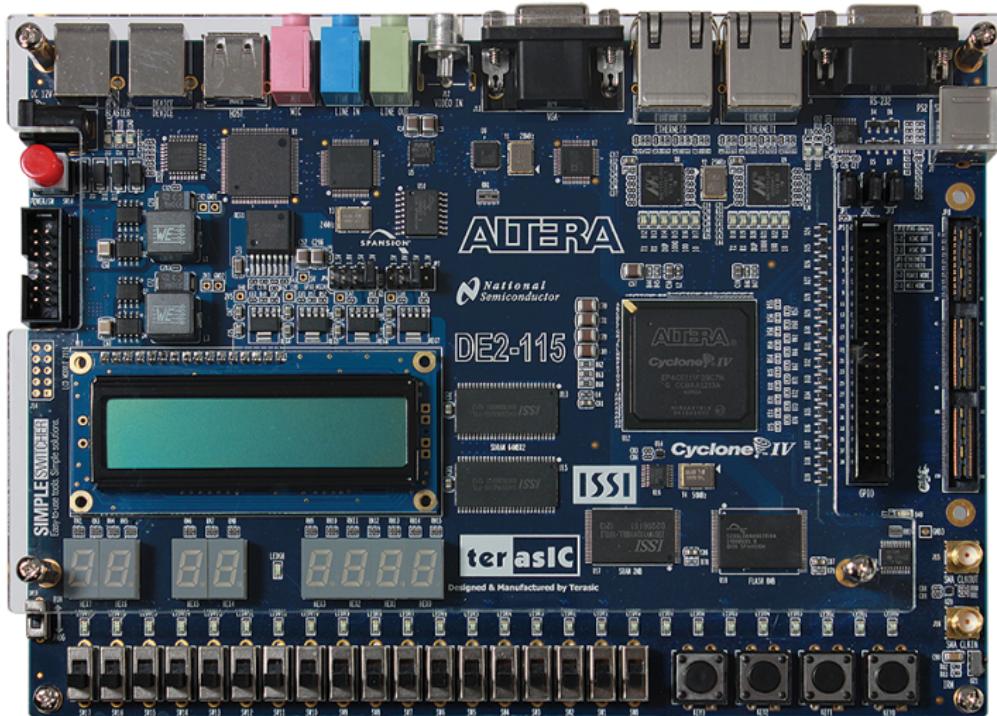


Figura: FPGA de Desenvolvimento Altera DE2-115.

TODO

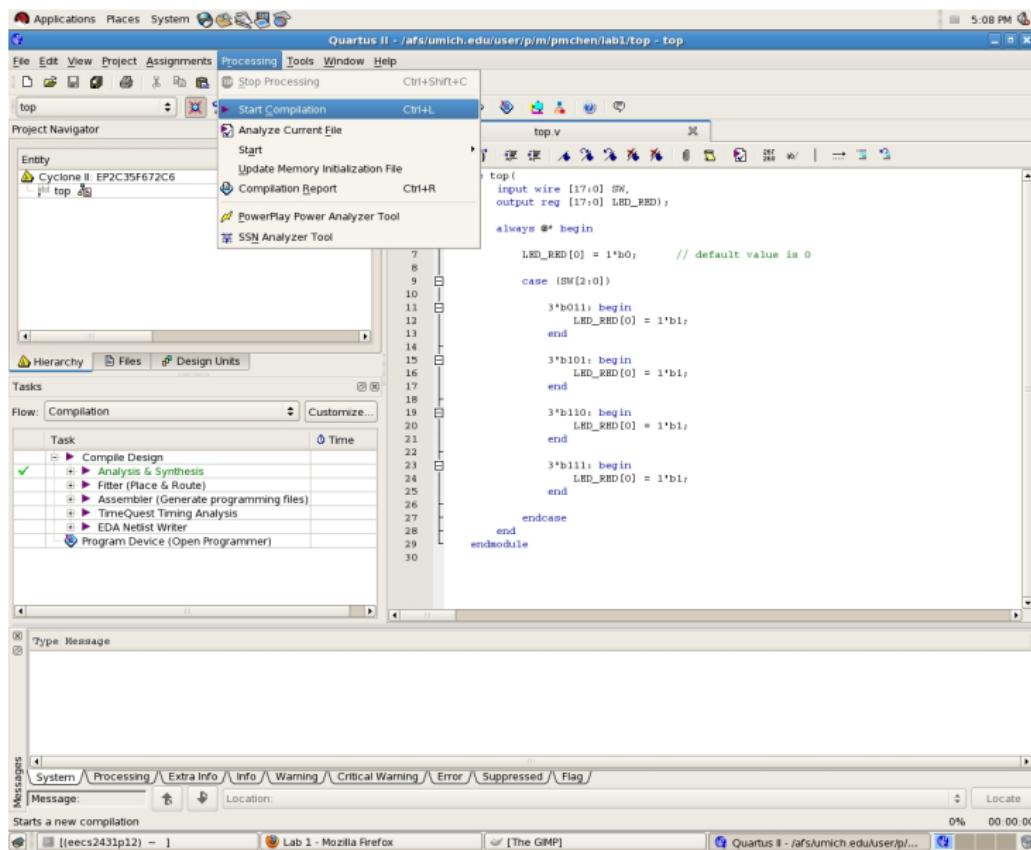


Figure: Altera Quartus II

- Características importantes da placa utilizada:
 - **Altera DE2-115** utilizada para ensino e desenvolvimento.
 - Quantidade de elementos lógicos: 114.480;
 - 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 [3].

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**¹.

¹Sistemas digitais podem mensurar variáveis no ambiente físico e tomar decisões sobre tal.

Arduino Mega 2560

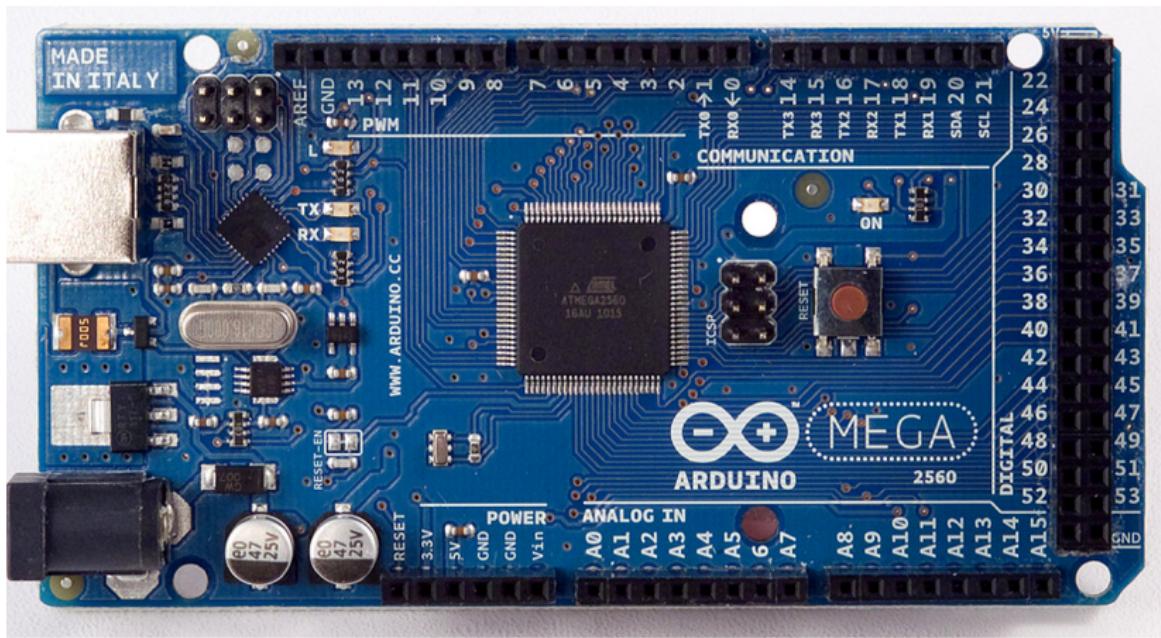


Figura: Arduino Mega 2560.

TODO

The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.6.5". The main window displays the "Blink" sketch. The code is as follows:

```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeating
 * this sequence forever.
 * modified 8 May 2014
 * 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
    delay(1000);                // wait for a second
}
```

The status bar at the bottom indicates "Arduino/Genuine Uno on COM1".

Figure 1. 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 16 MHz.

Linux

Definição de 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 MINIX² obedecendo o padrão POSIX [4].
- 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.

²Baseado no UNIX.

- Linux é um kernel *monolítico* no qual os drivers podem:
 - Executar com acesso total ao hardware;
 - Ser facilmente 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 [5].

A distribuição Linux utilizada para a realização do desenvolvimento:

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



Figura: Logo Linux Mint.

Sendo executado na máquina virtual:

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



Figura: 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

- Utilizadas a mais de 40 anos para a comunicação entre dispositivos [4].
- 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.**

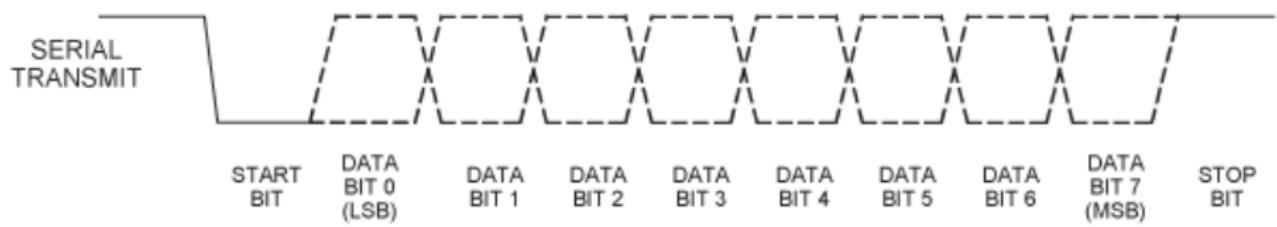


Figura: Protocolo de comunicação UART.

- Um dispositivo USB possui: *Endpoints*, *Interfaces* e *Configurations*;
- Atributos: *idVendor*, *idProduct*, Números de *Interfaces*, *Endpoints*...;
- Modo de envio de pacotes **por meio (ou não)** de URB³:
 - **Tipo:** Interrupt, Bulk, Control, Isochronous;
 - **Operações:** Criar, destruir, enviar, callback handler, cancelamento.
- Entre muitas outras características.

³ **USB Request Block.**

■ 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 [4].
- O Linux tem um **amplo e sólido suporte** ao USB.

■ Desvantagem:

- Protocolo complexo.

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

- Desenvolveu-se o método de encriptação do algoritmo **3DES** no qual utiliza:
 - Três chaves de 64-bits;
 - Blocos de texto de exatos 64-bits;
 - Ele opera 1 bloco de texto e retorna o resultado;
- Método de **envio e recebimento de informações** utilizando protocolo **UART**⁴:

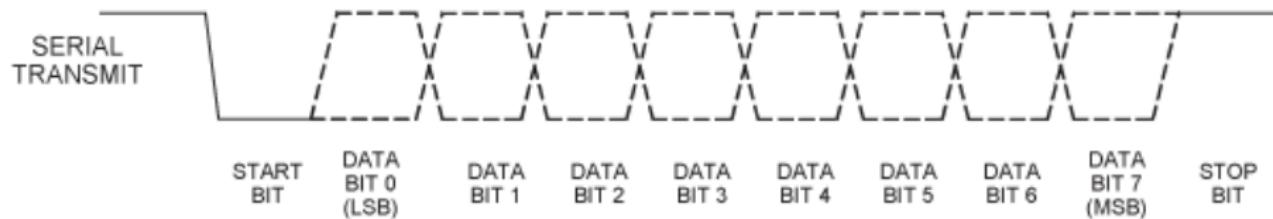


Figura: Protocolo de comunicação UART.

⁴Universal Asynchronous Receiver/Transmitter

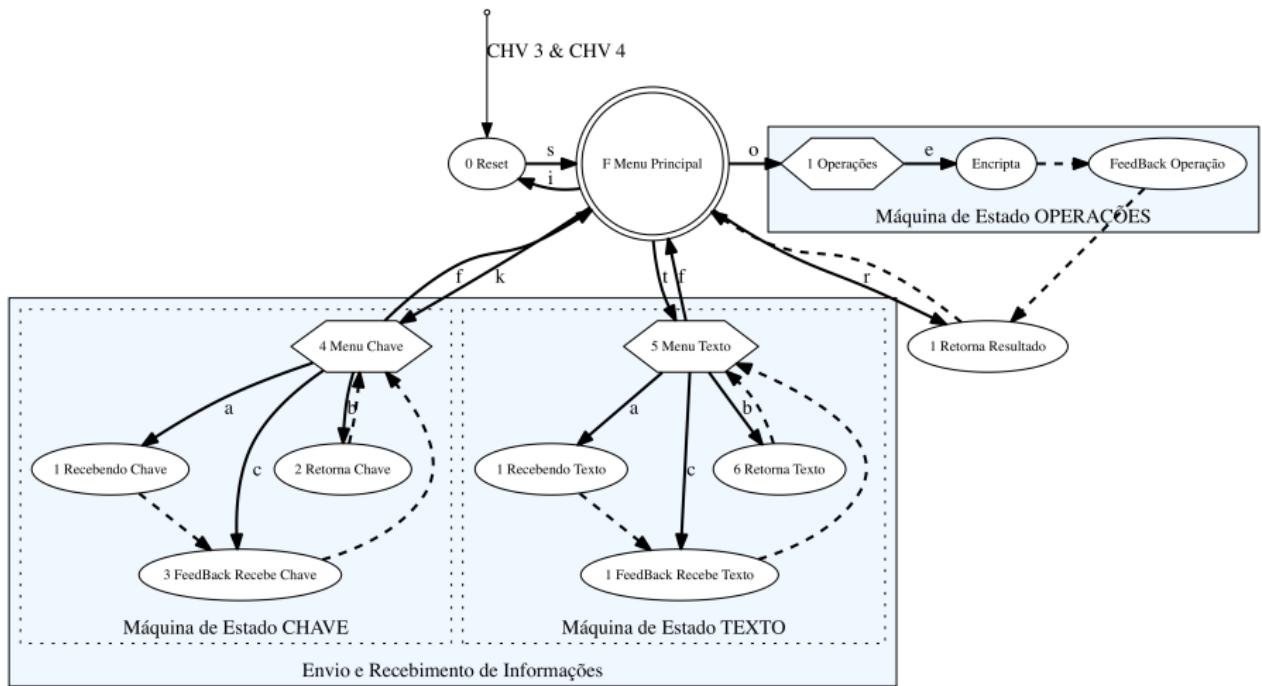


Figura: Máquina de Estados do FPGA.

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** possibilidades de construção de um driver USB Serial:
 - **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 que utiliza **Caracteres (Char)** mostrou-se ser mais simples e de fácil entendimento para a comunicação USB.

- Com pesquisas mais profundas, descobriu-se que:
 - Um driver Char com comunicação **TTY** pode utilizar bibliotecas específicas de USB;
- Utilizado o TTY com a biblioteca USB Serial:
 - TTY utiliza **Macros** para que o processo seja mais automatizado;
 - O **desenvolvimento** é facilitado;
 - E consequentemente a **compreensão**.

Comunicação utilizando TTY

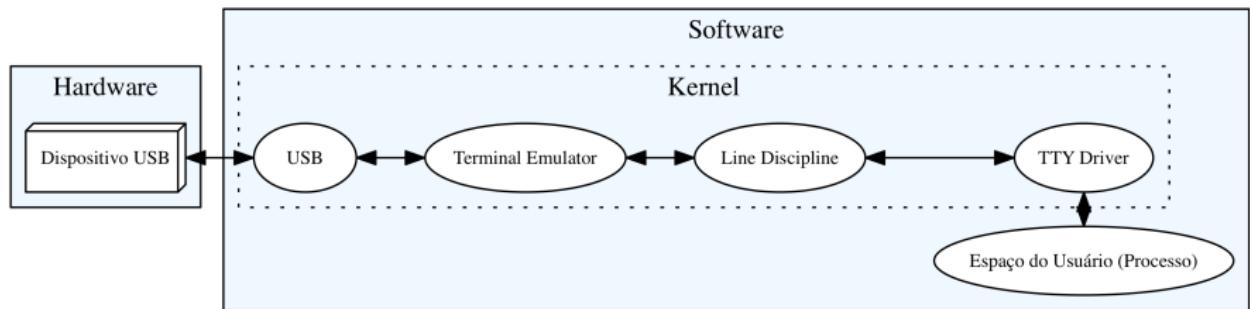


Figura: Comunicação utilizando TTY.

Sumário

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

Arduino

Desenvolvimento - Estrutura do Projeto

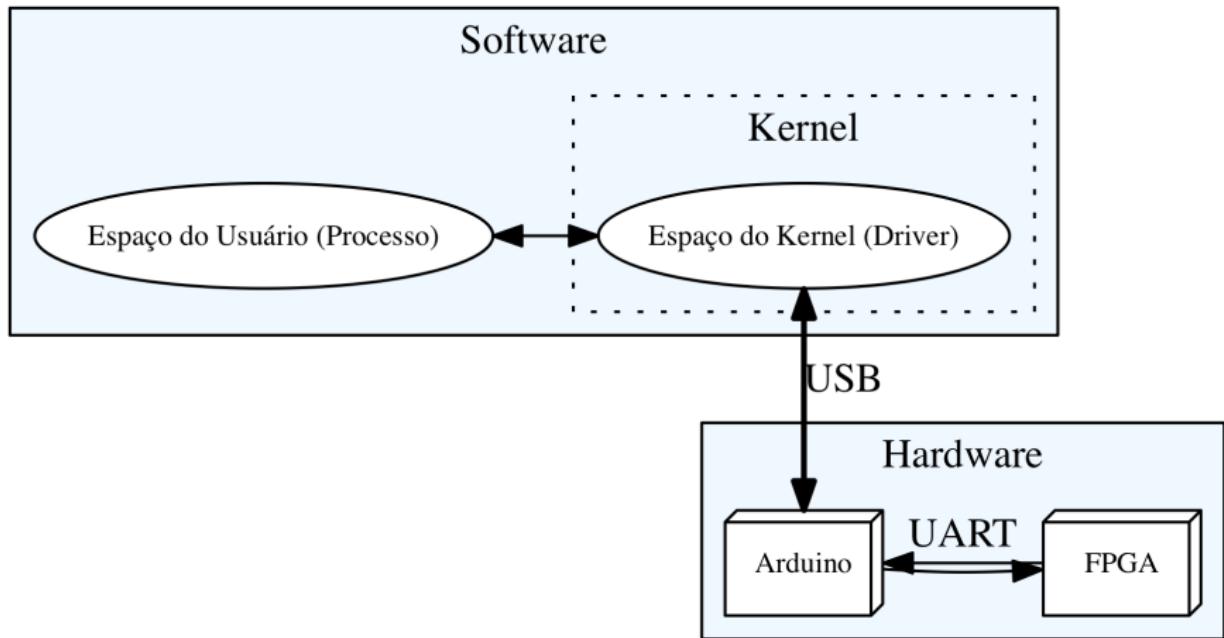


Figura: Representação do projeto numa visão geral.

- **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 controladora 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.

Sumário

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

- 1 Driver realiza sua função de permitir a comunicação com o Arduino via USB por meio de manipulação de arquino no diretório /dev.
- 2 O Arduino faz o papel de intermediário entre as extremidades exibindo as informações corretamente.
- 3 É possível realizar a encriptação de **apenas 1 bloco de 64-bits por vez**:
 - Sendo este alterado manualmente pelo usuário.

Tarefas Pendentes

- 1 Adicionar todos os procedimentos pra que a decriptação seja possível de ser executada.
- 2 Tornar o limite do texto limitado somente pelo **tamanho de armazenamento disponível no Arduino**.
- 3 Aprender mais sobre os conceitos de comunicação por TTY.

TODO

- Alterar o funcionamento da comunicação USB para que tenha uma via só para envio de controle e outra só para envio de dados⁵.
- Adicionar o protocolo USB no FPGA:
 - Mensagens de auxílio ao usuário sejam tratadas no driver eliminando o dispositivo intermediário;
 - Ou, criação de um software a nível de usuário para todo o controle do hardware.

⁵Se possível

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 9 de novembro de 2015

Referências I

-  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 [jhttp://br-linux.org/faq-linux¿](http://br-linux.org/faq-linux). Acessado dia
22/10/2015.

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 }
```

■ Linguagem C.

```
1 // t           // wait for a second
2 digitalWrite(13, LOW);    // turn the LED off by making the voltage
3 delay(1000);          // wait for a second
4 }
```

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 9 de novembro de 2015