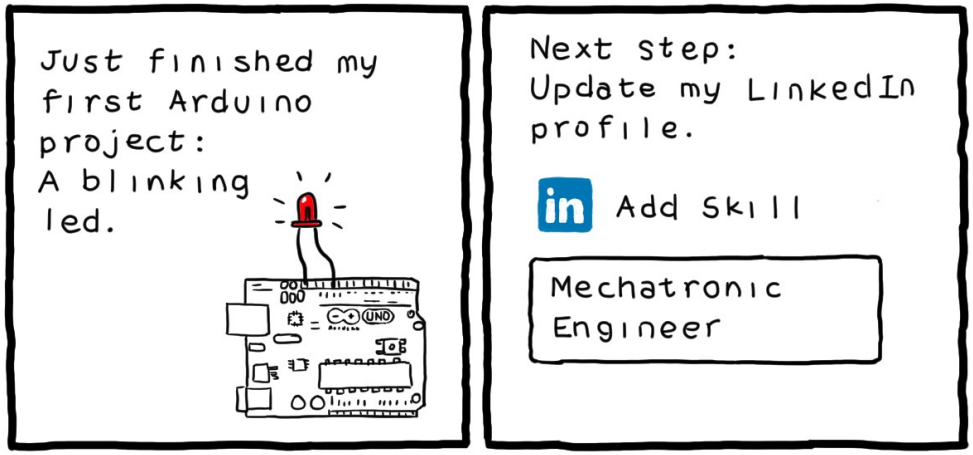


www.turnoff.us

Arquitetura Harvard e Microcontroladores ATmega

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA

Arquitetura Harvard

- Temos memórias para dados e instruções separadas
- **Máquinas com Arquitetura Harvard Pura**
  - Sequem estritamente esse conceito
  - Comum em microcontroladores

Arquitetura Harvard

- **Máquinas com Arquitetura Harvard Modificada**
  - Relaxa a separação física entre a memória de dados e instruções
  - Podemos encaixar a maioria dos PC’s modernos (ex.: x86) nessa arquitetura
  - Nos níveis de memória mais baixos, o processador opera em uma **Arquitetura de Harvard**
    - Temos memórias cache separadas para dados e instruções
  - Em níveis de memória mais altos, temos uma máquina de **Von Neumann**
    - Os dados são acessados por um único barramento até a memória

Exemplo

- Execute o comando *lscpu*
  - Exemplo em minha máquina
    - L1d: cache de dados
    - L1i: cache de instruções

```
...
Nome do modelo:          Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz
Step:                    9
CPU MHz:                 700.020
CPU MHz máx.:            3500,0000
CPU MHz mín.:            400,0000
BogoMIPS:                5799.77
Virtualização:           VT-x
cache de L1d:            64 KiB
cache de L1i:            64 KiB
cache de L2:             512 KiB
cache de L3:             4 MiB
...
```

Verifique você mesmo

- Baixe o datasheet do microcontrolador ATmega328P-PU que utilizaremos

- Verifique que
  - Temos uma Arquitetura Harvard pura
  - Conjunto de instruções RISC
  - Instruções de 16 bits ← Algumas podem ser estendidas para 32 bits
    - Conjunto de instruções AVRe+
  - **Palavras de 8 bits**
  - Capacidades de armazenamento
    - 32Kbytes de memória de programa (Flash)
    - 2Kbytes de memória de dados de trabalho (SRAM)
    - 1Kbytes de memória de dados de armazenamento permanente (EEPROM)

ATmega328P

- 32Kbytes de memória de programa (Flash)
  - Na melhor das hipóteses (todas instruções de 16 bits), seu programa pode ter no máximo **16.384 instruções** em linguagem de máquina
    - Programar em assembly agora pode não ser opcional
- 2Kbytes de memória de dados de trabalho (SRAM)
  - Vai ficar difícil executar um programa em Java ou abrir o Google Chrome!
- 1Kbytes de memória de dados de armazenamento permanente (EEPROM)
  - Pense muito bem qual dado você realmente precisa salvar

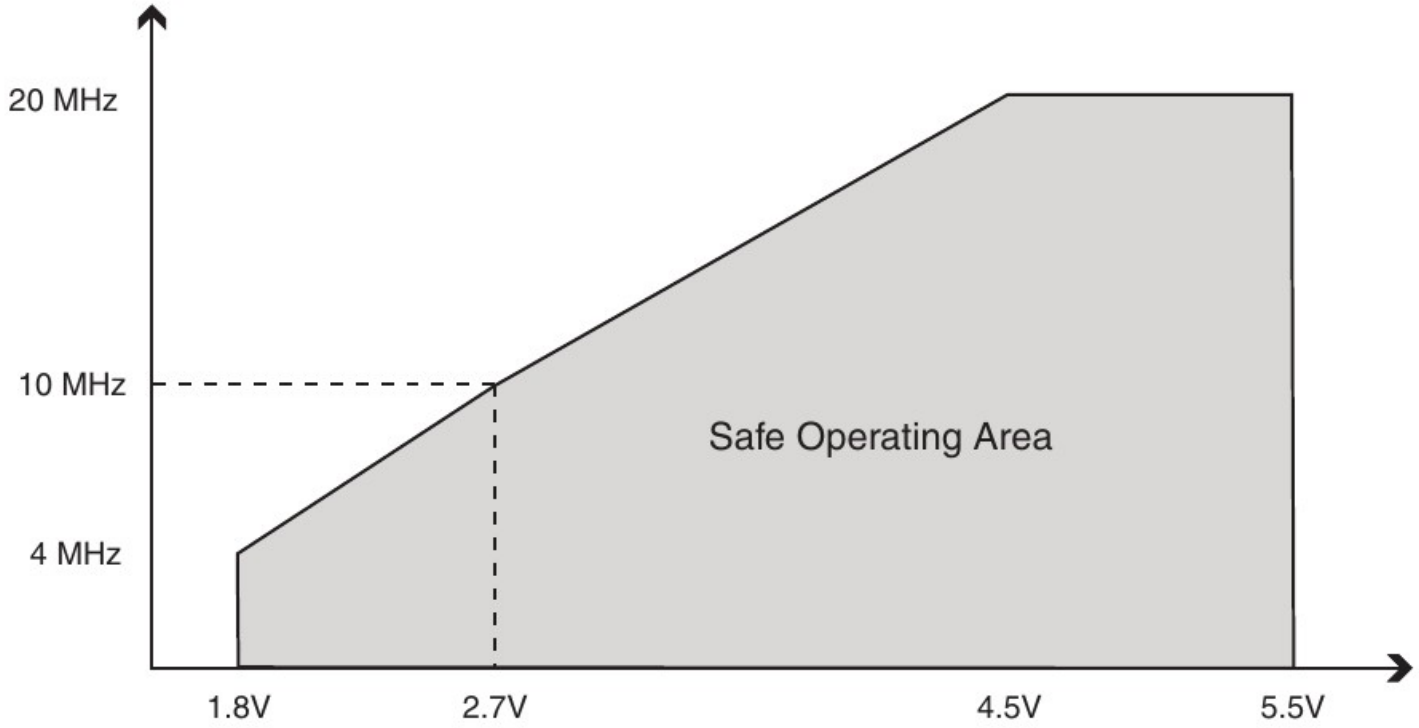
Qual a frequência de operação

- Frequência de operação do ATmega328P
  - Oscilador interno de 8MHz
  - Pode operar a até 20MHz via oscilador externo
- Quantos ciclos de clock cada instrução precisa para ser executada?
  - Temos um pipeline de 2 estágios
  - Para a maioria das instruções, uma instrução é completada a cada ciclo de clock

Microcontroladores são resilientes

- No datasheet do ATmega328P, procure pelas tensões de operação do microcontrolador de acordo com a frequência
  - Seção sobre especificações elétricas

Microcontroladores são resilientes



Tensão de operação aceitável em temperaturas entre -40°C e +85°C (Microchip, 2018)

UDESC

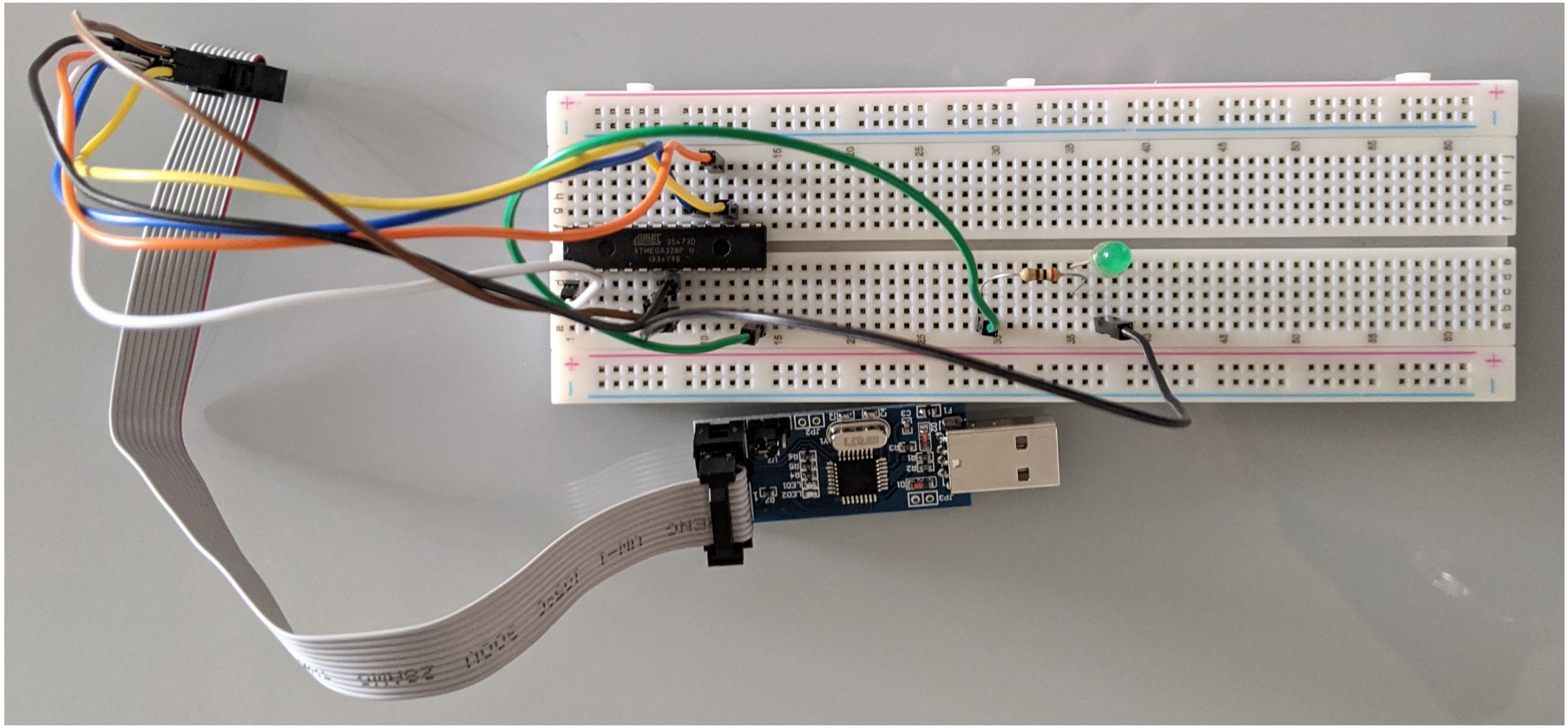
Resiliência

- Os ATmega328P possuem encapsulamentos capazes de suportar temperaturas entre -40 e +125
  - E operar com tensões tão baixas quanto 2.7 Volts de maneira segura nessas temperaturas
- Podem ser utilizados na indústria automotiva
  - Veja detalhes em http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Analise seu microcontrolador

- Seu microcontrolador possui diversos pinos
  - 28 para ser mais preciso
- Cada pino possui uma função específica
- Procure no *datasheet* o “Diagrama de Pinos” do seu microcontrolador

Conectando o ATmega328P ao Gravador



UDESC

Entrada/Saída

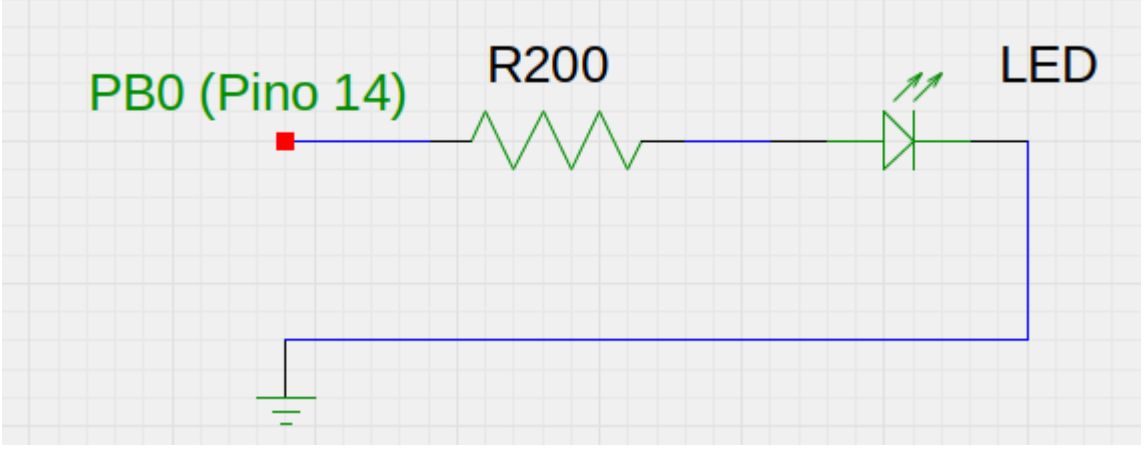
- No ATmega328P temos três portas
  - **Porta B**
    - 8 pinos
  - **Porta C**
    - 7 pinos
  - **Porta D**
    - 8 pinos
  - A maioria dos pinos pode ser configurado como entrada ou saída
    - **Existem exceções, veja no manual**
- Os pinos da Porta B são identificados como PB0, PB1, PB2 …
- Os pinos da Porta C são identificados como PC0, PC1, PC2 …
- Os pinos da Porta D são identificados como PD0, PD1, PD2 …
- **Veja esses pinos no datasheet**

Portas como saída

- Quando um pino é configurado como saída
  - Ao enviar o **nível lógico baixo (0)**, **0 volts** são enviados ao pino
  - Ao enviar o **nível lógico alto (1)**, a tensão utilizada na alimentação – **Vdd**, é enviada ao pino
    - A tensão pode ser muito menor, pois a corrente drenada é limitada
    - Veja os limites de corrente no datasheet
    - As portas são feitas para operar como **acionadores**, e **não como fontes de alimentação**
      - No entanto, como nossos circuitos são simples, poderemos utilizá-las para alimentação dos circuitos
        - **Espere quedas de tensão severas nas portas**

Piscando um LED

- Ligue o ânodo (lado positivo do LED – a “perna” mais longa) em PB0 (pino 14)
  - Insira um resistor de 200Ω entre a PB0 e o LED $(V_{dd} – V_{LED})/I_{led}$ = (5-2)/0,02 = 150Ω (um resistor de 200Ω deve servir)
  - Ligue o cátodo do LED no terra
    - O terra do circuito está ligado na porta GND do ATmega328P



Olá Mundo

- Vamos criar a versão de microcontroladores de um Olá Mundo
  - Crie um arquivo led.c em um diretório de sua preferência
  - Abra com um editor de texto qualquer

Olá Mundo

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main( ){

      DDRB = 0b00000001;

      while(1){
            PORTB = 0b00000001;
            _delay_ms(1000);
            PORTB = 0b00000000;
            _delay_ms(1000);
   }
}
```

UDESC

Olá Mundo

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main( ){

      DDRB = 0b00000001;

      while(1){
            PORTB = 0b00000001;
            _delay_ms(1000);
            PORTB = 0b00000000;
            _delay_ms(1000);
   }
}
```

Demais pinos da porta B são entradas

Pino 0 da porta B é saída

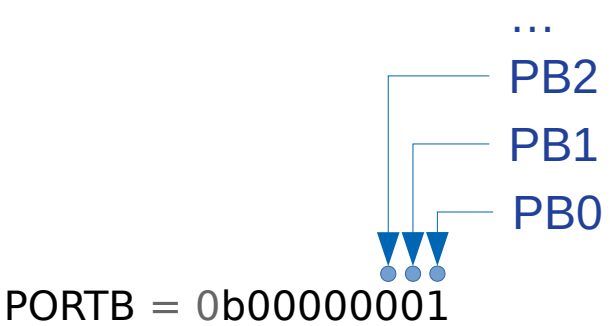
Envie sinal **alto** no pino 0 da porta B, demais pinos da porta B com sinal **baixo**

Envie sinal **baixo** em todos o pinos da porta B.

UDESC

Little-endian

- Atenção com a ordem dos bits
  - *Little-endian* (da direita para a esquerda)



Compilando

- Para compilar, utilize o comando

avr-gcc -Wall -Os -mmcu=atmega328p -o led led.c

Veja detalhes em https://linux.die.net/man/1/avr-gcc

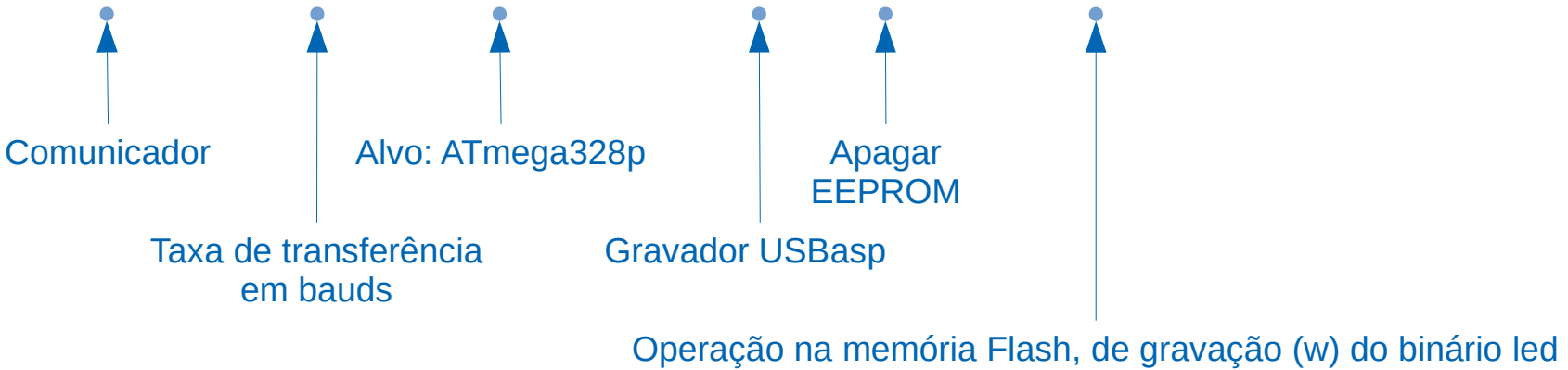
Verificando

- Conecte o USBasp no seu computador
- Para verificar se as conexões estão corretas
  - avrdude -c usbasp -p m328p

Carregando o binário

- Se o comando do slide anterior foi executado corretamente, um binário chamado **led** foi criado no diretório do programa
- Para carregar o binário no microcontrolador
  - Conecte o USBasp no computador

avrdude -b 9600 -p m328p -c usbasp -e -U flash:w:led



Exercícios

1. DDRB e PORTB são registradores do ATMega. Pesquise sobre eles no manual.

2. Adicione um contador no código, e o loop while fará o LED piscar 1x por segundo se o contador for menor que 5 (incremente o contador em 1 a cada iteração do loop), e depois disso piscará 2x por segundo. O contador deverá ser declarado:
   a) Como um **inteiro sem sinal**
   b) Como um **double**

- Submeta os seus códigos fonte no Moodle. Crie um **arquivo txt** indicando o tamanho de cada binário gerado, e indicando o percentual da memória de programa consumida por cada binário
  - você pode verificar o tamanho do binário via *du -h nomeBinario* (menos preciso), ou via *avr-size --mcu=atmega16 nomeBinario*

- Submeta também um vídeo do seu circuito funcionando

Referências

- S. Naimi, S. Naimi, M. Mazidi. **The Avr Microcontroller and Embedded Systems Using Assembly and C.** 2010**.**
- **megaAVR® Data Sheet.** Microchip, 2018.
- **ATmega328P Automotive - Complete Datasheet**. Microchip.
- **AVR Instruction Set Manual**. Microchip, 2016.
- D. Patterson; J. Henessy. **Organização e Projeto de Computadores**: **a Interface Hardware/Software.** 5a Edição. Elsevier Brasil, 2017.