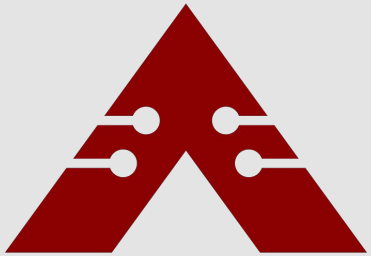


ADA

PROJETOS EM ENGENHARIA
DE COMPUTAÇÃO

Atmega328

O cérebro do arduino

Motivação

- Maior controle
- Escalabilidade
- Empreendedorismo
- Baixo custo
- Abre um leque de possibilidades

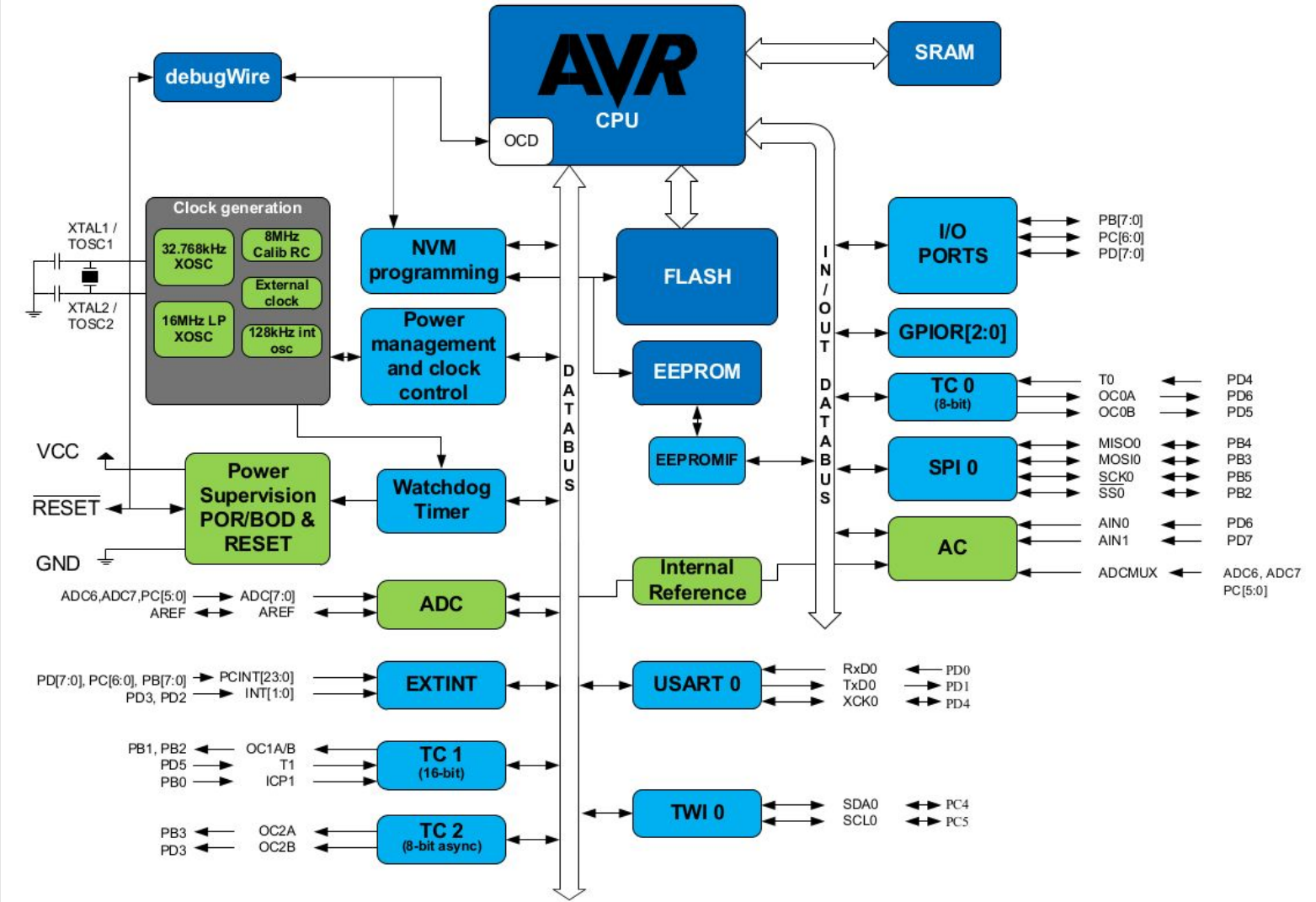
Atmega328

- Microcontrolador de 8 bits
- Arquitetura Harvard RISC (131 instruções)
- 32 x 8 registradores de uso geral
- Até 20MHz
- 2 multiplicadores de dois ciclos

Atmega328

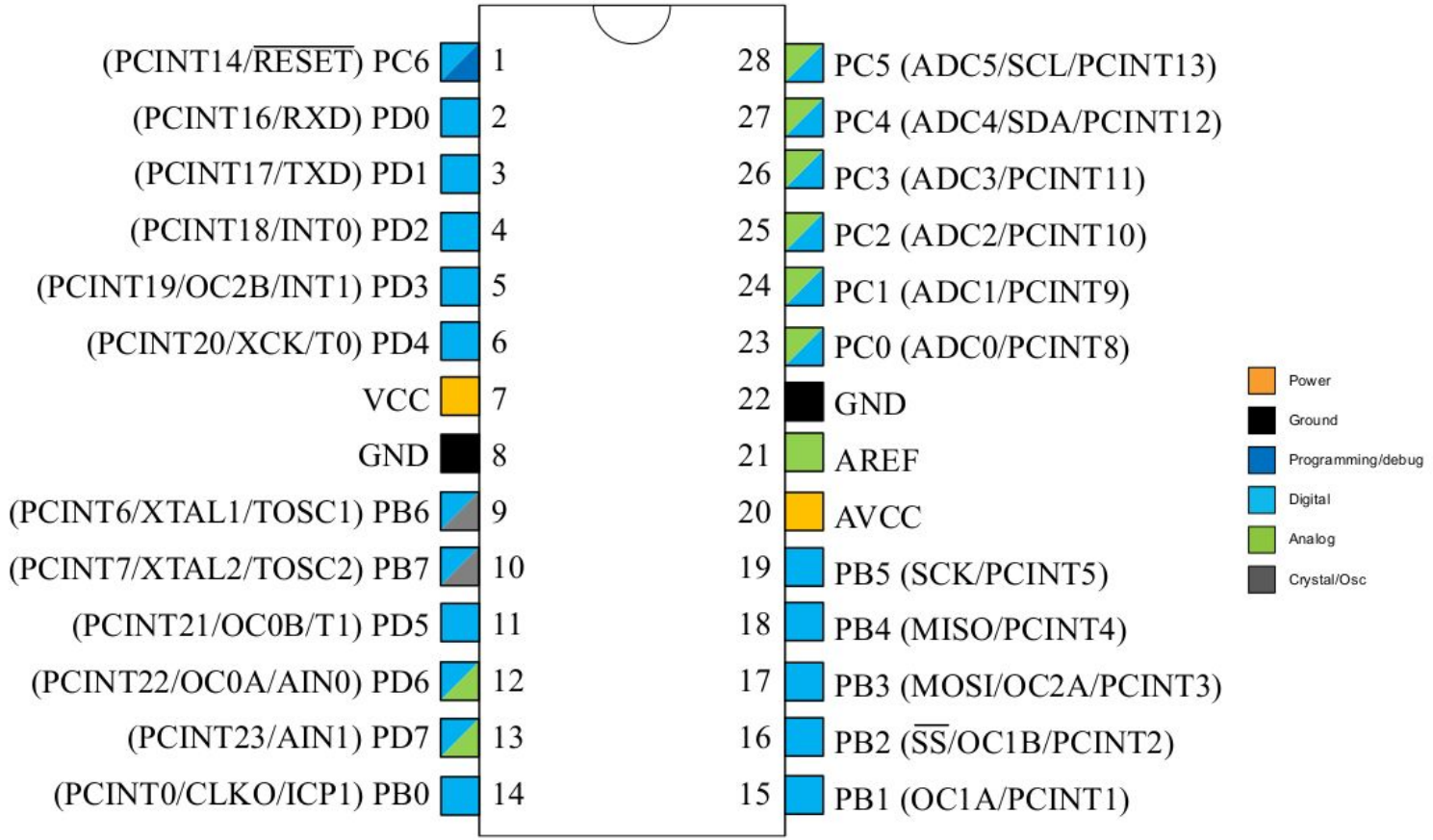
- 32KB de memória de programa Flash
- 1KB de EEPROM
- 2KB de memória SRAM
- 2 temporizadores/contadores de 8 bits
- 1 temporizador/contador de 16 bits

Aproximadamente 2 dólares.
No Brasil em torno de 10 reais.
Na China é possível achar por 5 reais!!


AVR
CPU
OCD
debugWire
SRAM
Clock generation
32.768kHz XOSC
8MHz Calib RC
External clock
16MHz LP XOSC
128kHz int osc
XTAL1 / TOSC1
XTAL2 / TOSC2
NVM programming
Power management and clock control
FLASH
EEPROM
EEPROMIF
DATABUS
I/O PORTS
PB[7:0]
PC[6:0]
PD[7:0]
GPIOR[2:0]
TC 0 (8-bit)
T0
OC0A
OC0B
PD4
PD6
PD5
SPI 0
MISO0
MOSI0
SCK0
SS0
PB4
PB3
PB5
PB2
AC
AIN0
AIN1
PD6
PD7
ADCMUX
ADC6, ADC7
PC[5:0]
VCC
RESET
GND
Power Supervision POR/BOD & RESET
Watchdog Timer
Internal Reference
ADC6,ADC7,PC[5:0]
AREF
ADC[7:0]
AREF
ADC
PD[7:0], PC[6:0], PB[7:0]
PD3, PD2
PCINT[23:0]
INT[1:0]
EXTINT
USART 0
RxD0
TxD0
XCK0
PD0
PD1
PD4
PB1, PB2
PD5
PB0
OC1A/B
T1
ICP1
TC 1 (16-bit)
TWI 0
SDA0
SCL0
PC4
PC5
PB3
PD3
OC2A
OC2B
TC 2 (8-bit async)

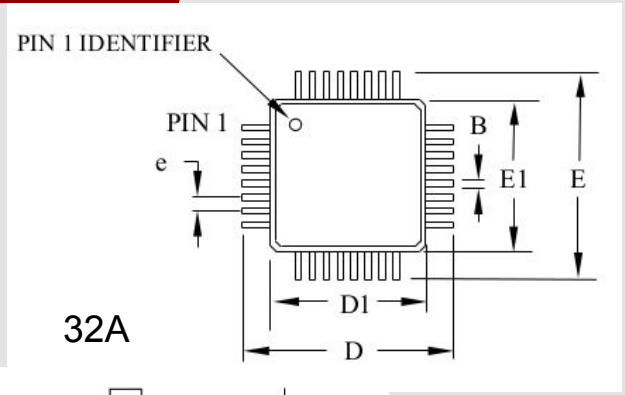
Pin-out

Figure 5-1. 28-pin PDIP

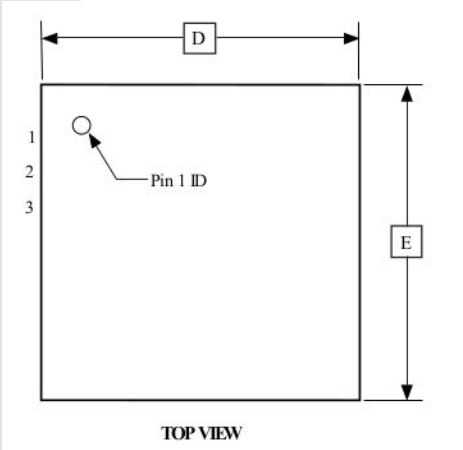


Atmega 328

- Outros encapsulamentos
  - 32A
    - E = 9mm
    - D = 9mm
  - 28M1
    - E = 4mm
    - D = 4mm



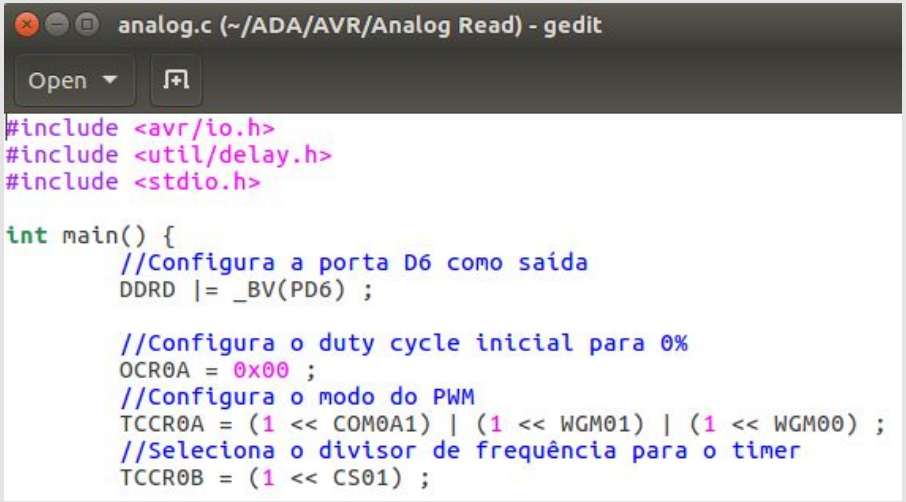
32A

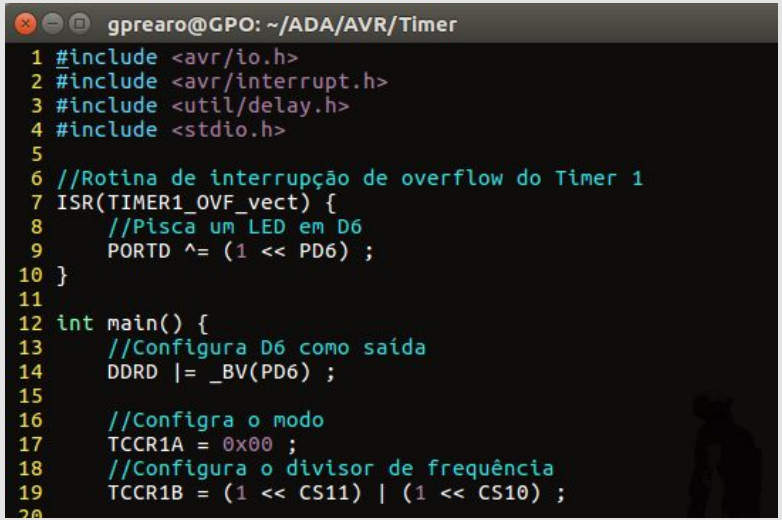


Ferramentas

Editor de texto

- Utilizado para criar nossos programas em C
- Podemos utilizar o `gedit` ou o `vim`, por exemplo.





Compilador

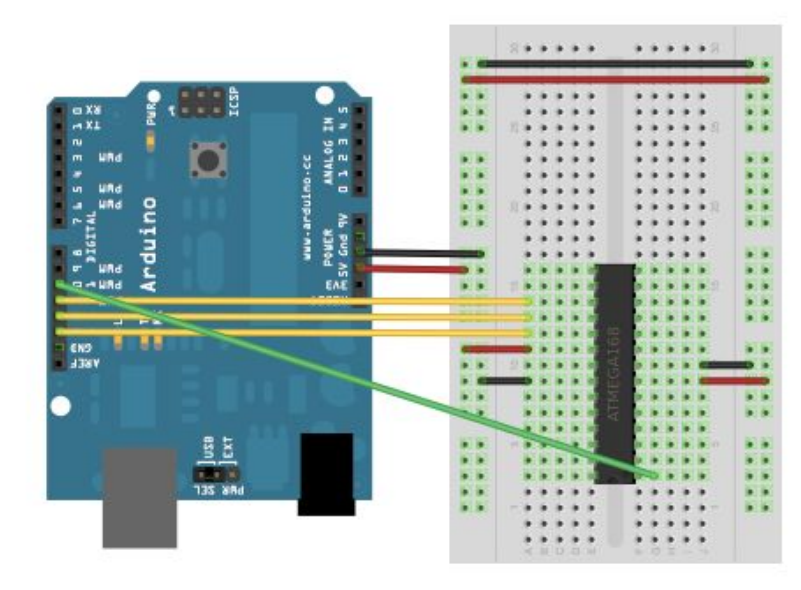
- Leva nosso código de C para o arquivo binário que será gravado no Atmega
- Iremos utilizar o gcc-avr
  - `sudo apt-get install gcc-avr gdb-avr  binutils-avr avr-libc`
  - `avr-gcc -Os -DF_CPU=$(CPU_F) -mmcu=atmega328p -c -o $(NAME).o $(NAME).c`
  - `avr-gcc -mmcu=atmega328p $(NAME).o -o $(NAME)`
  - `avr-objcopy -O ihex -R .eeprom $(NAME) $(NAME).hex`

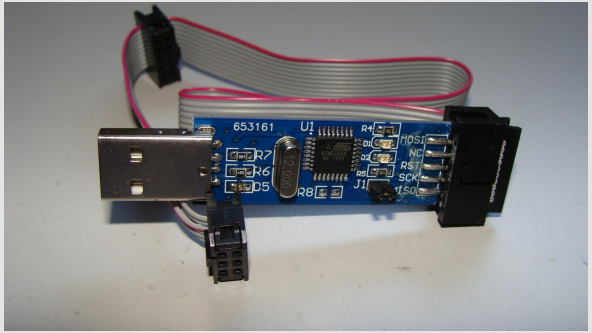
Gravador

- Utilizado para gravar o programa no Atmega
- Iremos utilizar o avrdude
  - `sudo apt-get install avrdude avrdude-doc`
  - `avrdude -c avrisp -p m328p -P /dev/ttyACM0 -b 19200`
  - `avrdude -c avrisp -p m328p -P /dev/ttyACM0 -b 19200 -U flash:w:$(NAME).hex`

Gravador físico

- Faz a comunicação entre o computador e o Atmega
- Usaremos um arduino configurado como gravador ISP





https://www.arduino.cc/en/Tutorial/ArduinoISP

Mão na massa

Fusíveis

- Memória não volátil
- No caso do ATmega328 são 3 bytes
- Determinam algumas configurações
  - Origem do clock
  - Frequência do clock
  - Debug Wire
  - Watch-dog timer

Fusíveis

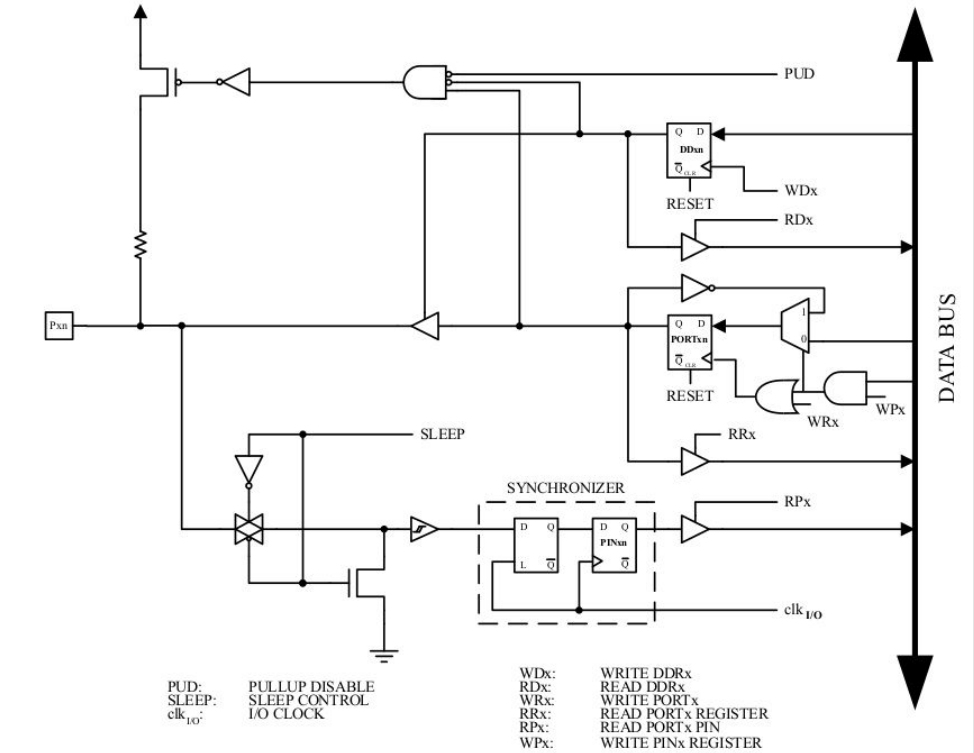
- Como modificar?
- Podemos utilizar o avrdude e nosso gravador físico
- Existem calculadoras de fusíveis
  - http://www.engbedded.com/fusecalc/
  - A calculadora acima gera a linha do comando do avrdude

Portas digitais

- Todas as portas são de entrada e saída
- Portas B (8 bits), C (6 ou 7 bits) e D (8 bits)
- Direção da porta configurada pelo `DDRx` (Data Direction Register)
  - `DDRB, DDRC, DDRD`
- Saída configurada pelos registradores `PORTx`
- Entrada lida pelos `PINx`

Portas digitais

Figure 18-2. General Digital I/O(1)



Portas digitais

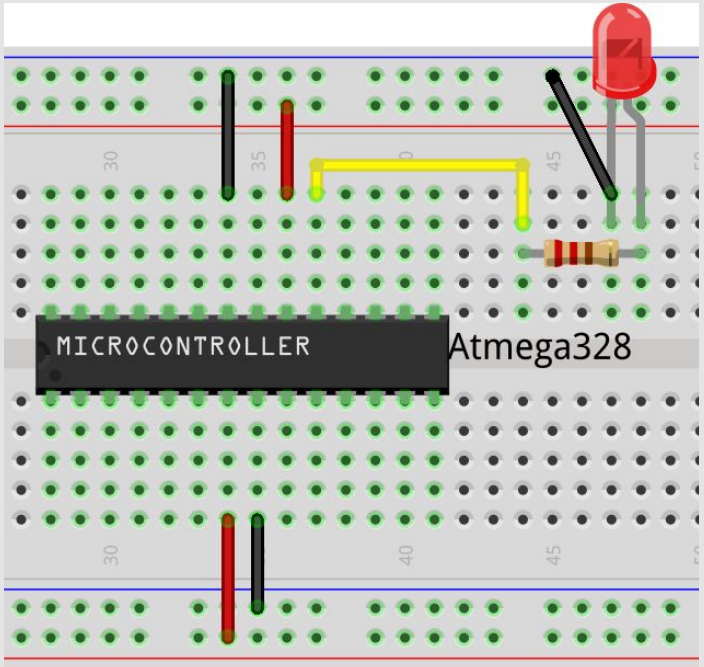
Exemplo de código:

- Nível lógico alto em PB5
  - `DDRB = (1<<DDB5) ;`
  - `PORTB = (1<<PB5) ;`
- Ler nível lógico da porta B2
  - `unsigned char i ;`
  - `i = (PINB & (1 << PINB2)) ;`

Programa Blink

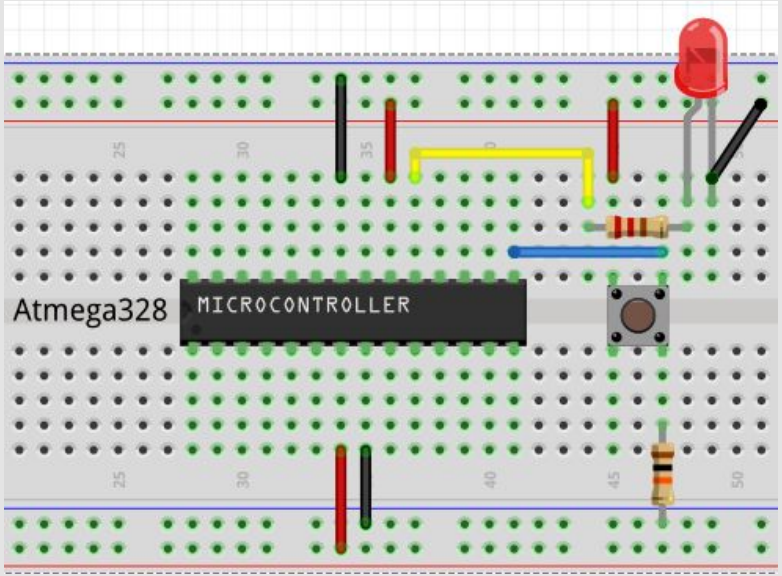
- Fazer um LED piscar utilizando um bit de umas das portas
- 1s ligado e 1s desligado

Dica: _delay_ms(1000) ;



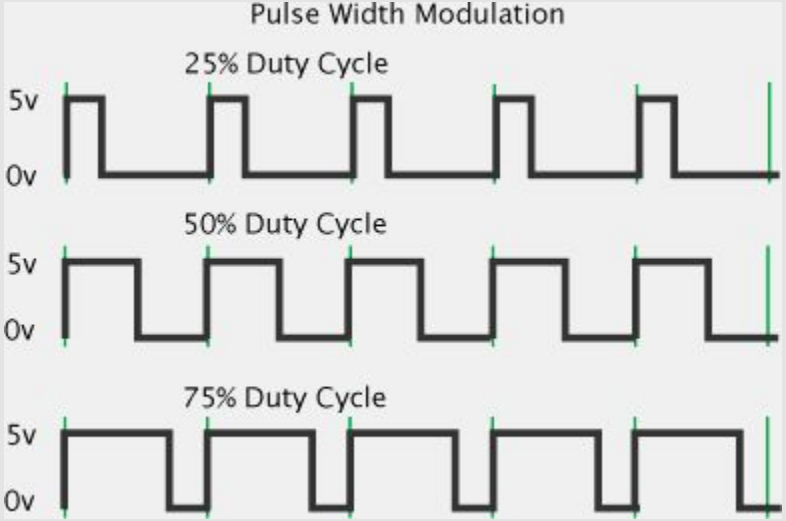
Acionar um LED com um botão

- Ao pressionar um botão o LED acende
- Ao pressionar novamente ele apaga



Saída analógica

- PWM
  - Saída digital pulsada
  - Controle através do *duty cycle*



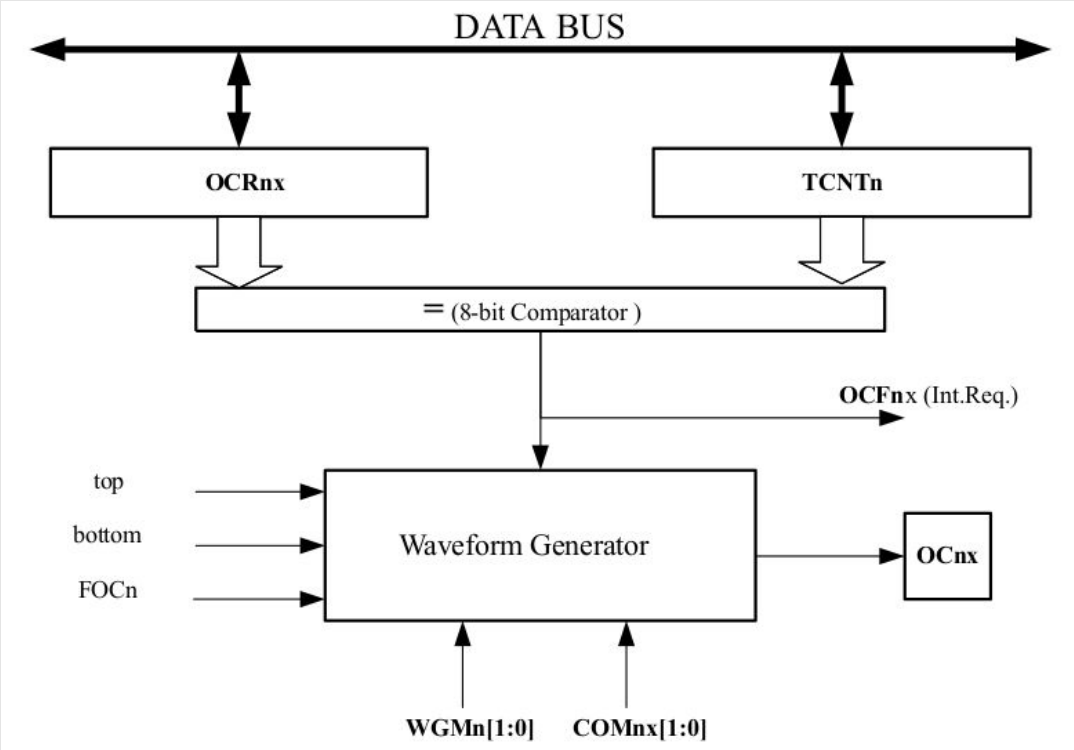
Saída analógica

- Através de PWM por temporizador
- Dois temporizadores de 8 bits (dois canais) e um de 16
- Usaremos o Timer 0, canal A (porta PD6)
- Registradores importantes:
  - `OCR0A (Output Compare Register)`
  - `TCCR0A (Timer/Counter 0 Control Register A)`
  - `TCCR0B (Timer/Counter 0 Control Register B)`

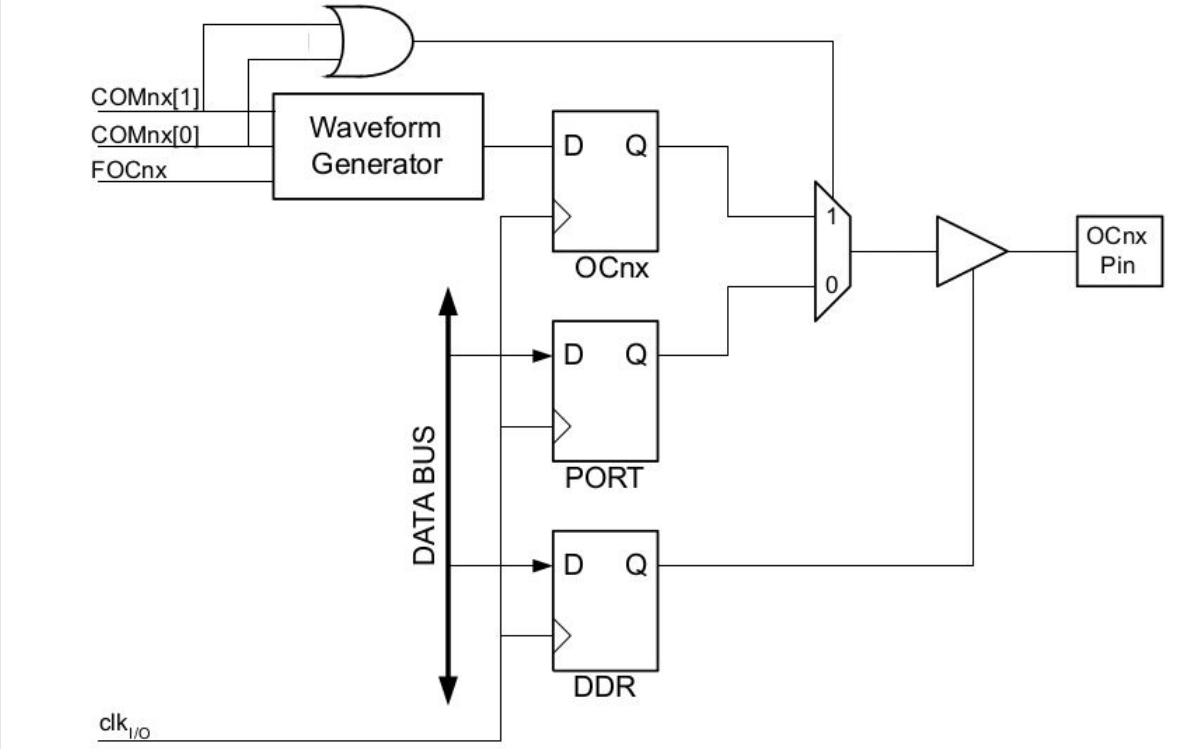
Saída analógica

- No Atmega:
  - Configura-se um temporizador para contar até certo número
    - Os bits 0 e 1 de `TCCR0A` definem disso
    - Usaremos ambos em 1, que equivale a contar até `0xff (Modo Fast PWM)`
  - Configura-se o que acontece quando há *match*
    - Bits 6 e 7 de `TCCR0A`, usaremos 0 e 1 respectivamente
  - Configura-se qual é a frequência de contagem
    - Os bits 0, 1 e 2 de `TCCR0B` definem isso
    - Usaremo 010, que equivale a ⅛ frequência do clock
  - Configura-se até qual número a saída deve ser 5V
    - Este número deve ser atribuído à `OCR0A`

Saída analógica



Saída analógica



Entrada analógica

- Feito através do conversor Analógico/Digital
- Resolução de 10 bits
- Tensão de referência selecionável

$$d = \frac{V_{in} \cdot 1024}{V_{ref}}$$

Entrada analógica

- Registradores importantes
  - ADMUX
    - Os bits 6 e 7 selecionam a tensão de referência
    - O bit 5 ajusta a posição do resultado a conversão
    - Os bitis de 0 a 3 selecionam qual canal deve-se realizar a leitura
  - ADCSRA
    - Bit 7 - Habilita a conversão Analógico-Digital
    - Bit 6 - Começa a conversão
    - Bit 4 - Sinaliza quando a conversão termina
    - Bits 0..2 - Frequência de conversão

Entrada analógica

- Registradores importantes
  - ADCH
    - Parte mais significativa do valor convertido
  - ADCL
    - Parte menos significativa
  - Os dois são alterados com ADLAR!!

Entrada analógica

- Controlar PWM com entrada analógica

Interrupções

- Executa um determinado código quando determinado evento ocorre
- Outros códigos podem ser executados enquanto o evento não ocorre
- A interrupção pode ser externa ou interna
  - Exemplos
    - Externa: borda de subida de um sinal em uma das portas
    - Interna: o temporizador terminou a contagem

Interrupções Externas

- Dois tipos de interrupções externas
  - PCINT e INT
- PCTINT
  - Registradores importantes
  - PCICR: Pin Change Interrupt Control Register
    - Bit 0 - Habilita interrupções PCINT0 ~ PCINT7
    - Bit 1 - Habilita interrupções PCINT8 ~ PCINT14
    - Bit 2 - Habilita interrupções PCINT16 ~ PCINT23
  - PCMSK: Pin Change Mask Register (de 0 a 2)
    - Habilita interrupções de cada pino

Interrupções

- Interrupção do tipo INT
- Registradores importantes
  - SREG: O bit 7 habilita interrupções
  - EICRA: External Interrupt Control Register A
    - Define os eventos de interrupção da INT0 (PD2) e INT1 (PD3)
  - EIMSK: External Interrupt Mask Register
    - Bit 0 habilita a interrupção INT0
    - Bit 1 habilita a interrupção INT1

Temporizadores

- Mesmo princípio do PWM
- Agora o timer gera uma interrupção

$$t = \frac{N \cdot D_f}{f_{clk}}$$

Blink com timer

- Usaremos o Timer1. Por que?
  - Os timers 0 e 2 são de 8 bits (N máximo = 255)
  - A máxima divisão de frequência é 1024
  - O ATmega trabalha a 8MHz
  - Então, o t máximo é 1/30 seg.
- O Timer1 tem 16 bits
  - N máximo = 65535
  - Com uma divisão de frequência de 64, conseguimos um t de aproximadamente 0.5 segundos!

Blink com timer

- Registradores importantes
  - TCCR1A: Timer/Counter 1 Control Register A
    - Bits de 4 a 7: Configuração da saída para porta do timer (não usaremos saída para porta)
    - Bits 0 e 1: Modo de operação
  - TCCR1B: Timer/Counter 1 Control Register B
    - Bits 6 e 7: Configuração de contagem por entrada (não usaremos)
    - Bits 3 e 4: Modo de operação
    - Bits de 0 a 2: Divisor de frequência
  - TIMSK1: Timer/Counter 1 Interrupt Mask Register
    - Bits 0 a 2: Habilitam interrupções (cada um em um evento diferente)

Obrigado!

ADA

PROJETOS EM ENGENHARIA
DE COMPUTAÇÃO