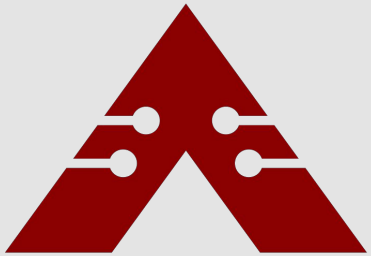


ADA

PROJETOS EM ENGENHARIA
DE COMPUTAÇÃO

Desconstruino o Arduino

Uma introdução ao Atmega

Motivação

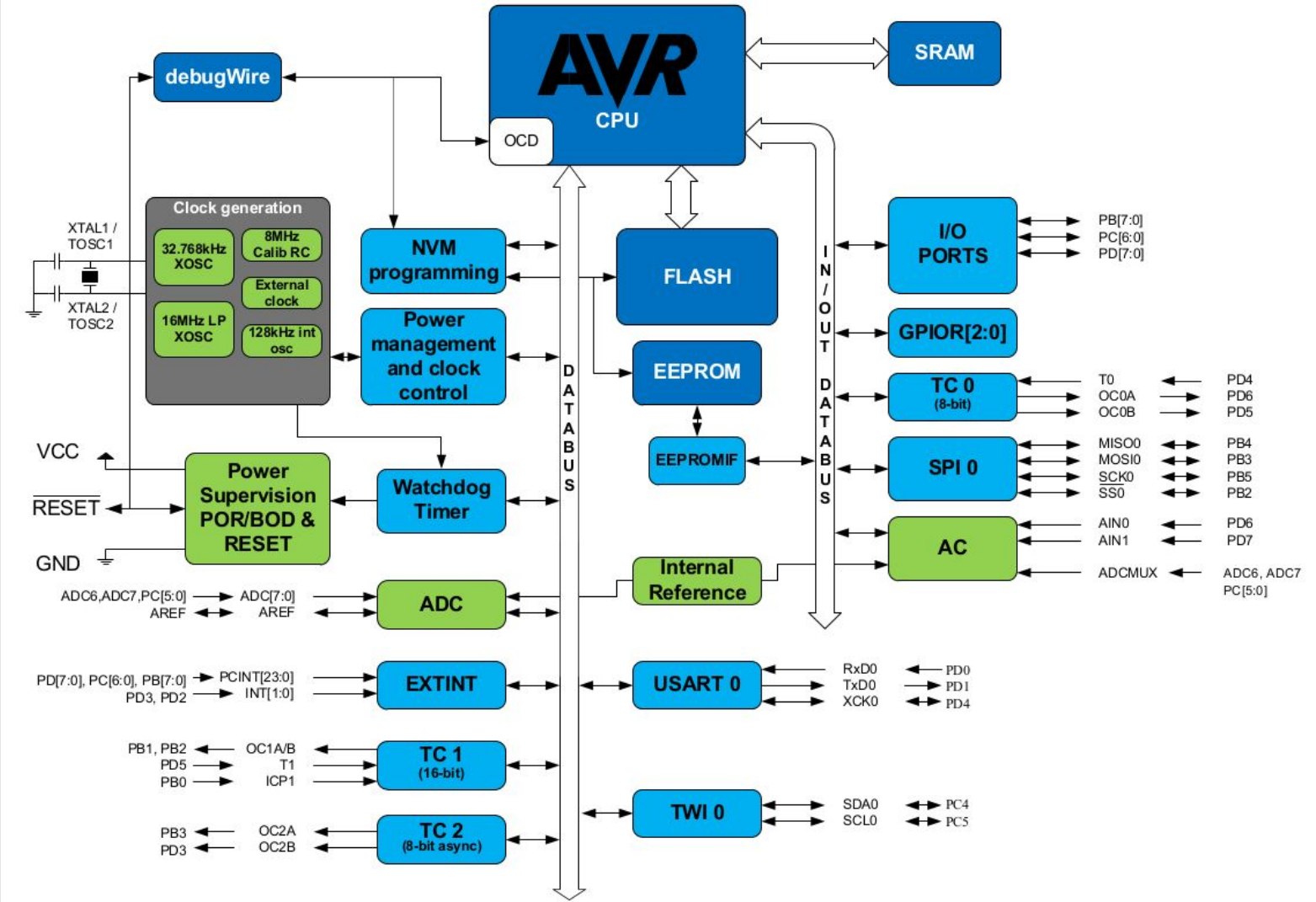
- Maior controle
- Baixo custo
- Abre um leque de possibilidades

Atmega328

- Microcontrolador de 8 bits
- Arquitetura Harvard RISC (131 instruções)
- Até 20MHz
- 2 multiplicadores de dois ciclos

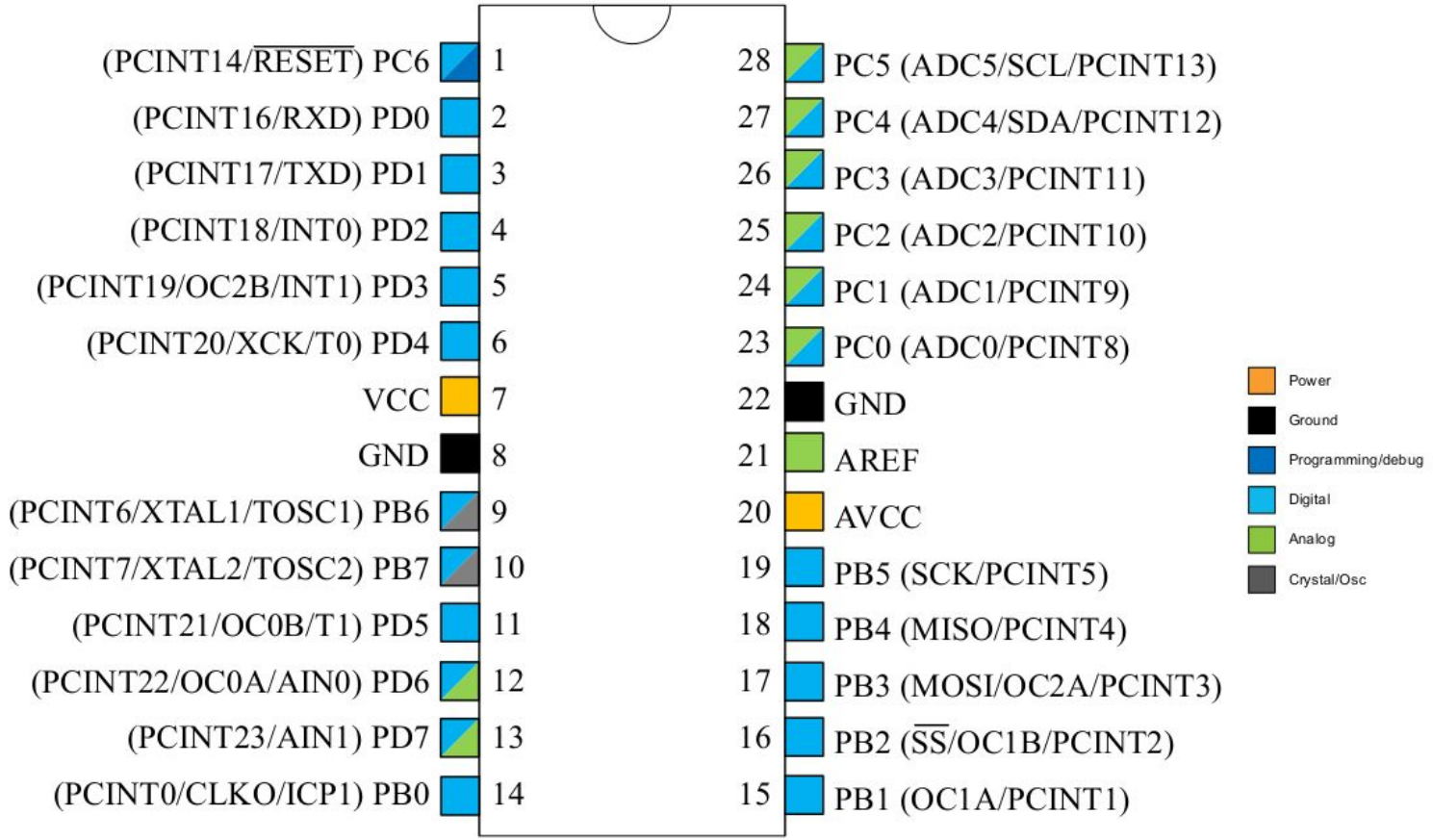
Atmega328

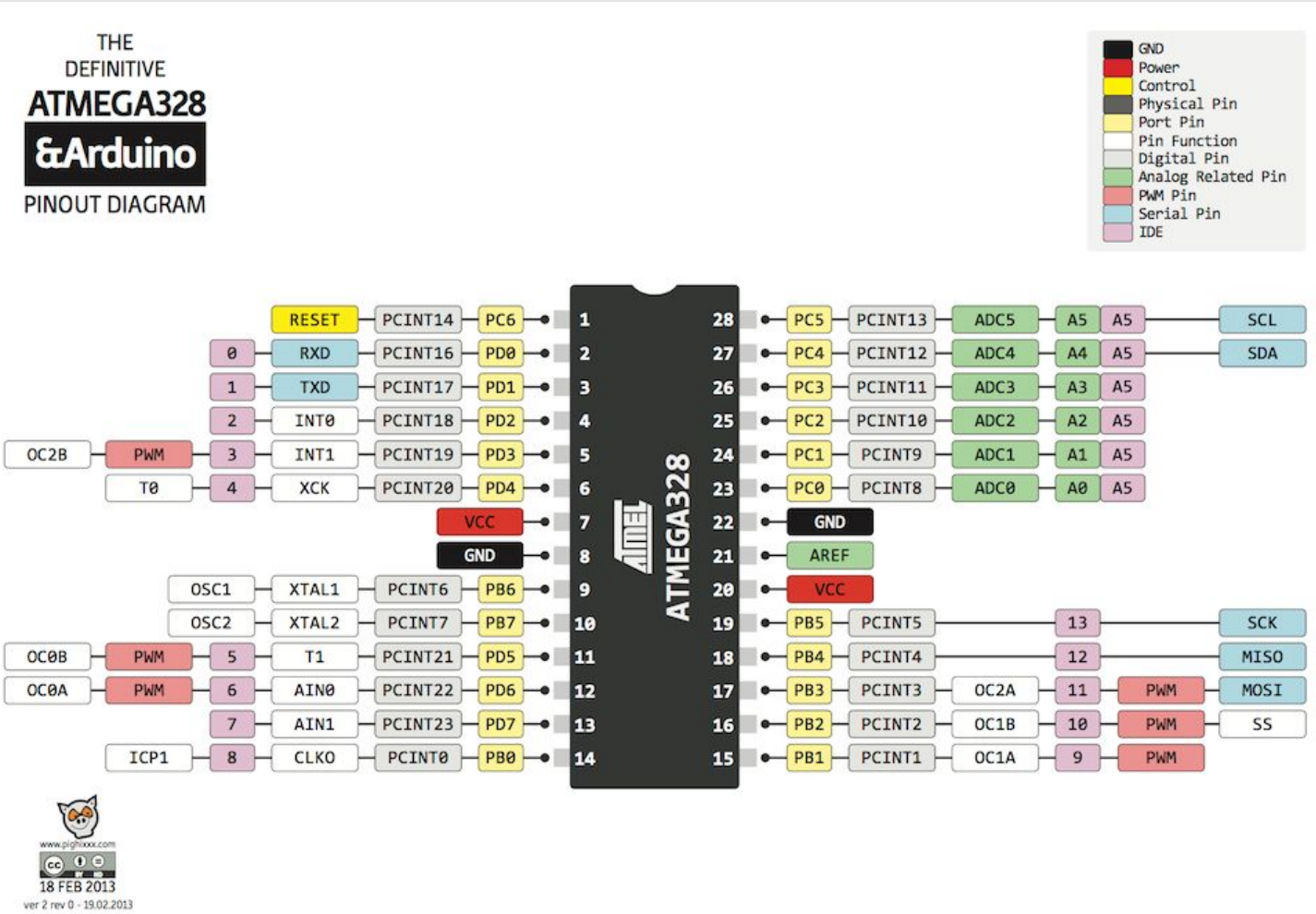
- Aproximadamente 2 dólares nos Estados Unidos.
- No Brasil em torno de 10 reais.
- Na China é possível achar por 5 reais!!

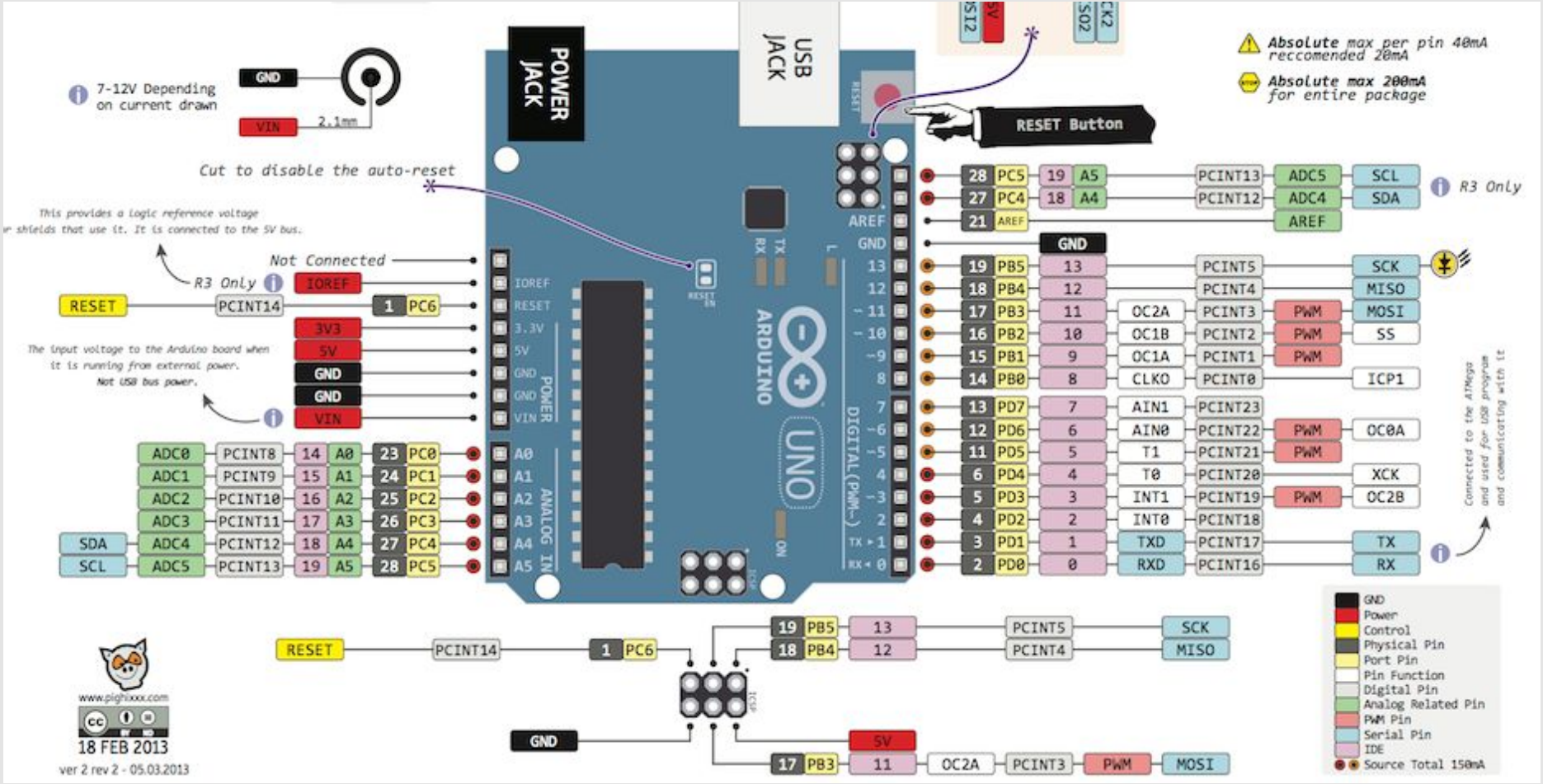

AVR
CPU
OCD
debugWire
SRAM
Clock generation
32.768kHz XOSC
8MHz Calib RC
External clock
16MHz LP XOSC
128kHz int osc
XTAL1 / TOSC1
XTAL2 / TOSC2
NVM programming
Power management and clock control
FLASH
EEPROM
EEPROMIF
DATABUS
I/O PORTS
PB[7:0]
PC[6:0]
PD[7:0]
GPIOR[2:0]
TC 0 (8-bit)
T0
OC0A
OC0B
PD4
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

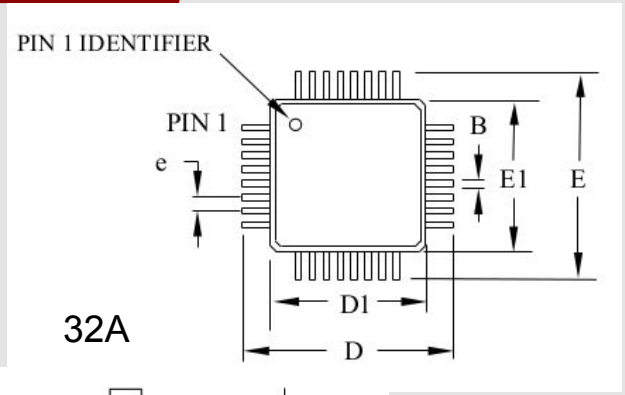



THE
DEFINITIVE
ATMEGA328
&Arduino
PINOUT DIAGRAM
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
ATMEGA328
RESET PCINT14 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
28 PC5 PCINT13 ADC5 A5 A5 SCL
27 PC4 PCINT12 ADC4 A4 A5 SDA
26 PC3 PCINT11 ADC3 A3 A5
25 PC2 PCINT10 ADC2 A2 A5
24 PC1 PCINT9 ADC1 A1 A5
23 PC0 PCINT8 ADC0 A0 A5
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 13 SCK
18 PB4 PCINT4 12 MISO
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
www.pighixxx.com
18 FEB 2013
ver 2 rev 0 - 19.02.2013

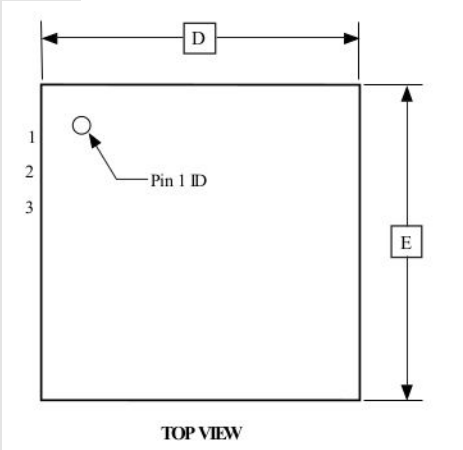

POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
GND
VIN
2.1mm
Cut to disable the auto-reset
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
R3 Only
Not Connected
This provides a Logic reference voltage for shields that use it. It is connected to the 5V bus.
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
Connected to the ATMega and used for USB program and communicating with it
ARDUINO
UNO
POWER
ANALOG IN
DIGITAL(PWM~)
IOREF
RESET
3.3V
5V
GND
VIN
AREF
SCL
SDA
SCK
MISO
MOSI
SS
ICP1
OC0A
XCK
OC2B
TX
RX
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013

Atmega 328

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



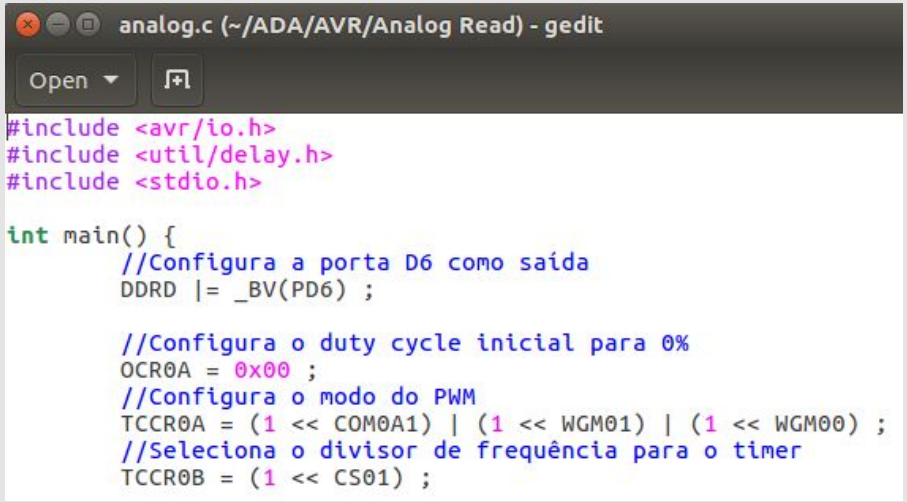
32A

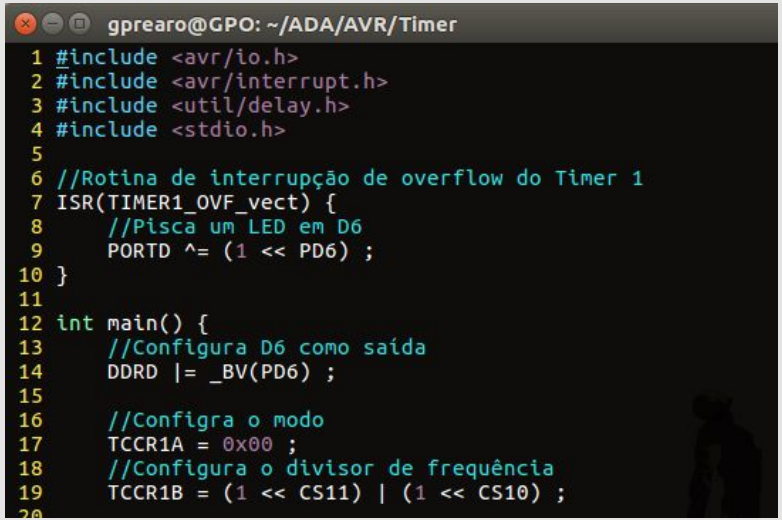


Ferramentas

Editor de texto

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





Compilador

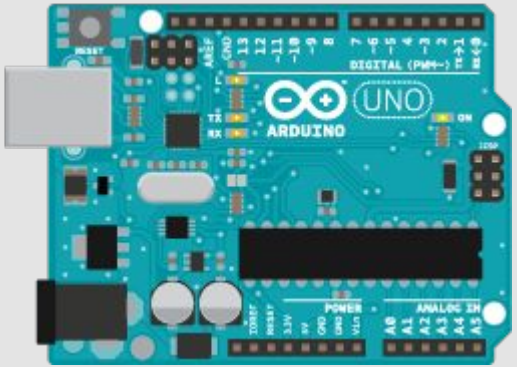
- Leva nosso código de C para o arquivo binário que será gravado no Atmega
- Iremos utilizar o gcc-avr
  - `sudo apt-get install gcc-avr gdb-avr  binutils-avr avr-libc`
  - `avr-gcc -Os -DF_CPU=$(CPU_F) -mmcu=atmega328p -c -o $(NAME).o $(NAME).c`
  - `avr-gcc -mmcu=atmega328p $(NAME).o -o $(NAME)`
 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

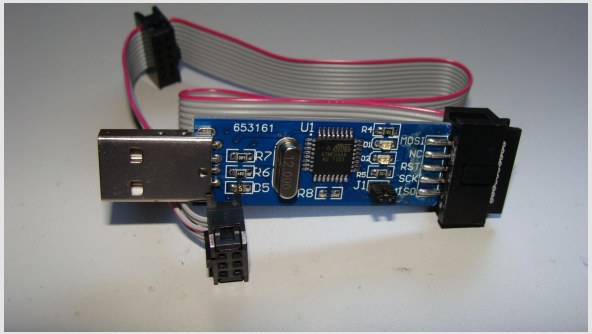
Gravador

- Utilizado para gravar o programa no Atmega
- Iremos utilizar o avrdude
  - `sudo apt-get install avrdude avrdude-doc`
  - `avrdude -c arduino -p m328p -P /dev/ttyACM0 -b 115200`
  - `avrdude -c arduino -p m328p -P /dev/ttyACM0 -b 115200 -U flash:w:$(NAME).hex`

Gravador físico

- Faz a comunicação entre o computador e o Atmega
- Usaremos a própria placa do Arduino





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

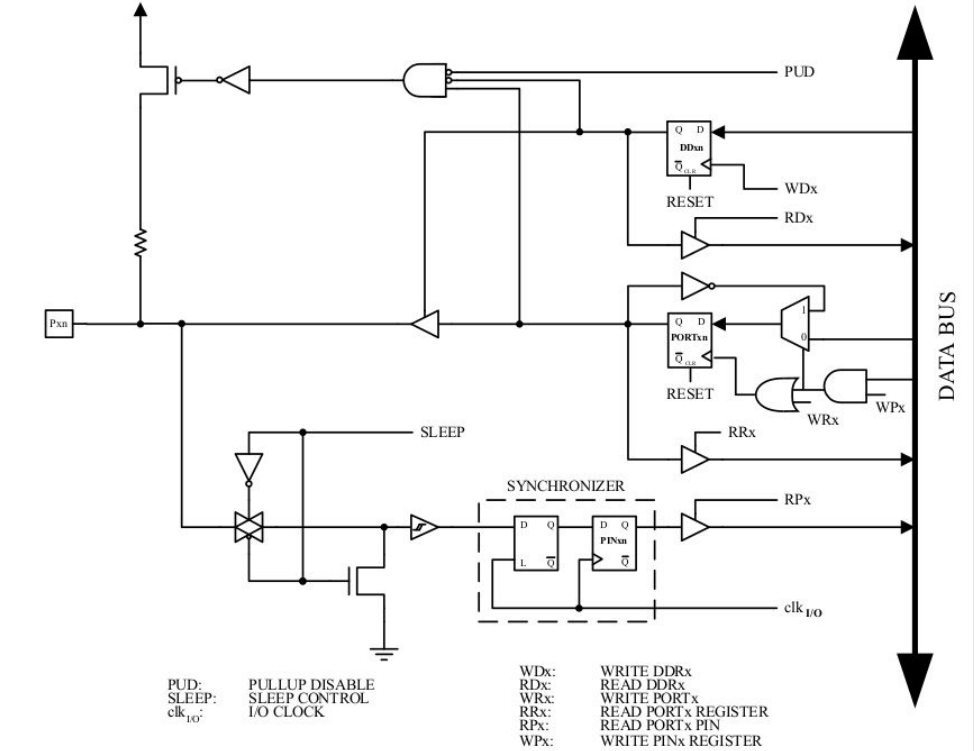
- Como modificar?
- Podemos utilizar o avrdude e nosso gravador físico
- Existem calculadoras de fusíveis
  - http://www.engbedded.com/fusecalc/
  - A calculadora acima gera a linha do comando do avrdude

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)) ;`

Portas digitais - Saída

- `pinMode(13, OUTPUT)` → `DDRB |= (1 << DDB5)`
- `digitalWrite(13, HIGH)` → `PORTB |= (1 << PORTB5)`
- `digitalWrite(13, LOW)` → `PORTB &= ~(1 << PORTB5)`

Manipulação bit-a-bit

- Setando um bit em um byte “DDRB |= (1<< DDB5)”

DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
(1 << DDB5)	0	0	1	0	0	0	0	0
| (ou)	DDB7	DDB6	1	DDB4	DDB3	DDB2	DDB1	DDB0

Manipulação bit-a-bit

- Limpando um bit em um byte “DDRB &= ~(1<< DDB5)”

DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
(1 << DDB5)	0	0	1	0	0	0	0	0
~(1 << DDB5)	1	1	0	1	1	1	1	1
& (e)	DDB7	DDB6	0	DDB4	DDB3	DDB2	DDB1	DDB0

Programa Blink

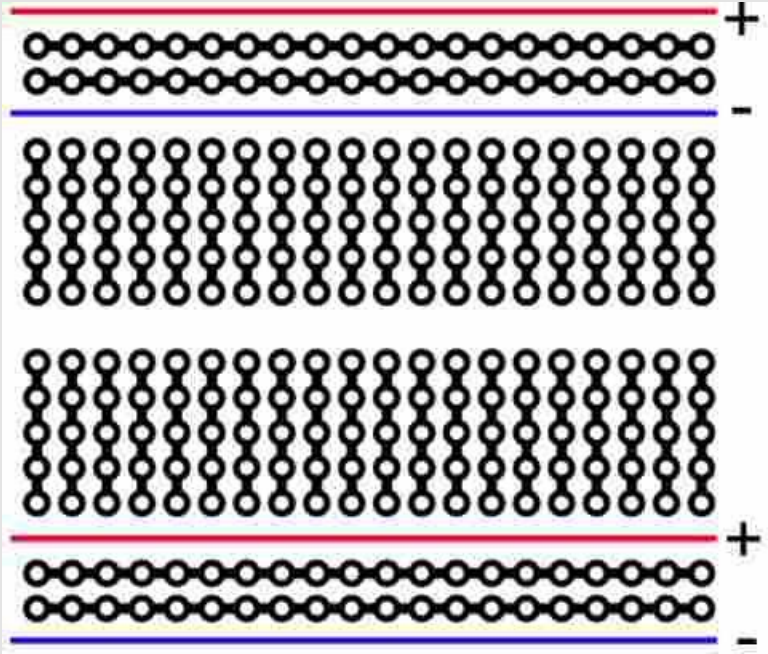
- Entrem na pasta Blink pelo terminal
- Executem o comando `make` e então `make upload`
- Abram o código `blink.c`

```
void setup() {
  pinMode(13, INPUT);
}


void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int main() {
    //Configura a porta D6 como output
    //O macro _BV(DDB5) equivale a (1 << DDB5)
    DDRB |= 1 << DDB5 ;

    while (1) {
        //Define a saída da porta D6 como nível lógico alto
        PORTB |= _BV(PORTB5) ;
        //Espera por 1000ms
        _delay_ms(1000) ;
        //Define a saída da porta D6 como nível lógico baixo
        PORTB &= ~_BV(PORTB5);
        //Espera mais 1000ms
        _delay_ms(1000) ;

    }
    return 0 ;
}
```

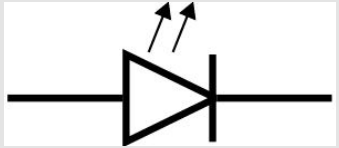
Conectando um LED externo

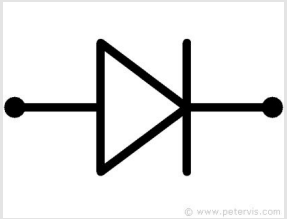
- Iremos precisar de uma protoboard!
  - É uma placa para prototipagem de circuitos
  - Os furos do meio são interligados na vertical
  - Os furos nas extremidades são ligados na vertical



Conectando um LED externo

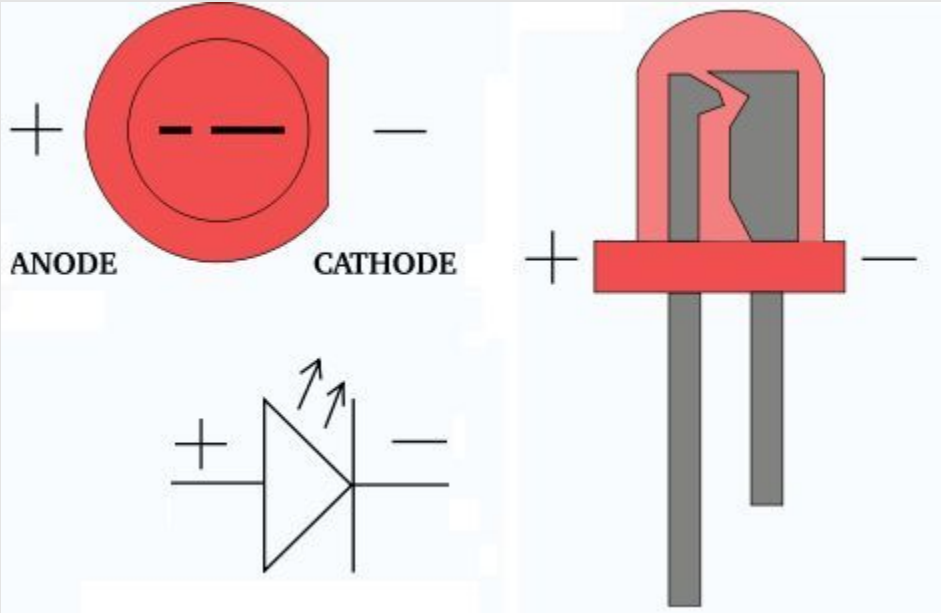
- O que é um LED?
  - Um diodo que emite luz





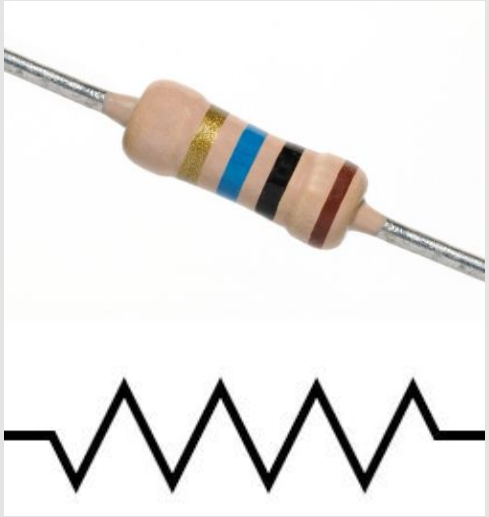
- Mas o que é um diodo?
  - É um componente que permite a passagem de corrente apenas em um sentido
- Como saber qual é o lado certo?

Conectando um LED externo

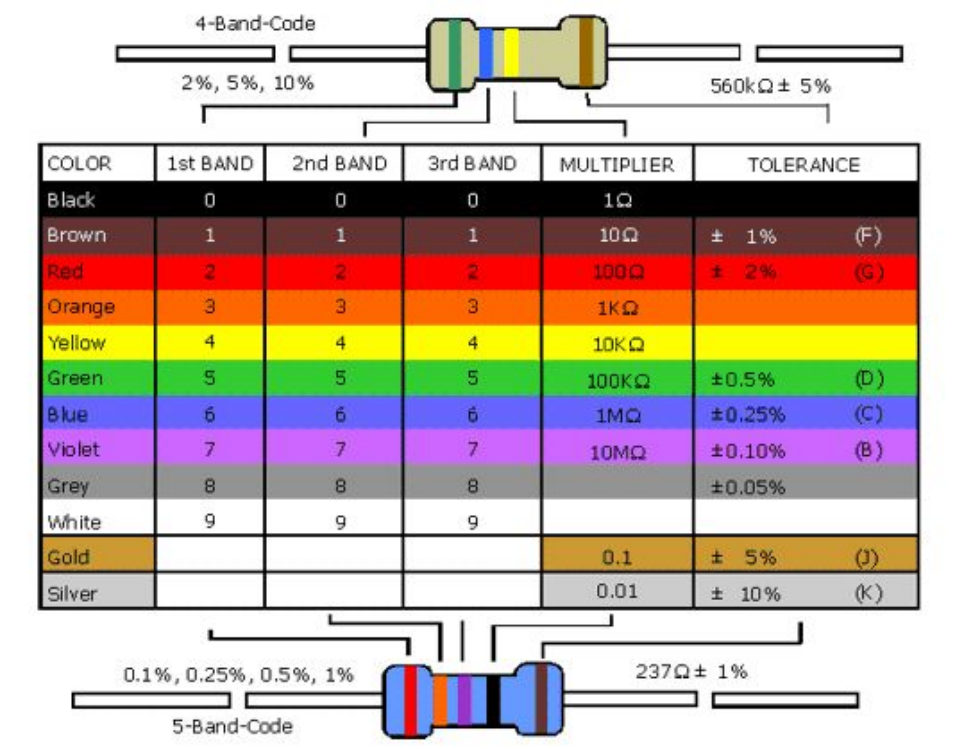


Conectando um LED externo

- Também precisamos limitar a corrente que passa pelo LED com um resistor
- As faixas coloridas nele indicam o valor (em Ohm) e a precisão



Conectando um LED externo



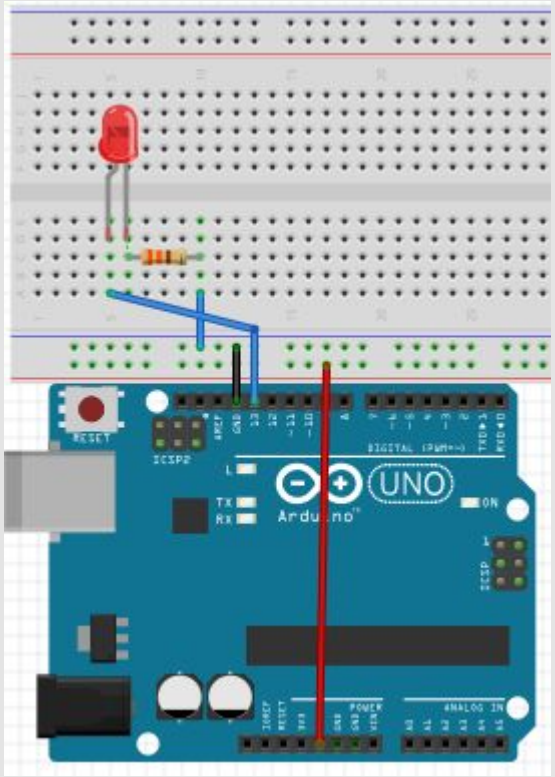
COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
Black	0	0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	±0.5%	(D)
Blue	6	6	6	1MΩ	±0.25%	(C)
Violet	7	7	7	10MΩ	±0.10%	(B)
Grey	8	8	8		±0.05%	
White	9	9	9			
Gold				0.1	± 5%	(J)
Silver				0.01	± 10%	(K)

Conectando um LED externo

- E como calcular o resistor ideal para o LED?
  - A conta é um pouquinho complicada, mas é mais ou menos assim:
  - Deve passar por volta de 20mA no LED
  - Com esta corrente passando, ele dissipa uma tensão de aproximadamente 2V
  - Alimentamos o circuito com 5V
  - Então (5V-2V)/20mA deve dar o valor da resistência

Conectando um LED externo

- Ok. Desse jeito o Led fica sempre aceso
  - Como podemos controlar?
- Ligando o LED em uma porta digital podemos controlar quando ele fica aceso ou apagado

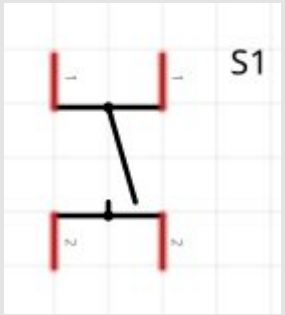


Exercício 01

- Ligar 3 LEDs em 3 portas digitais diferentes
- Acender o primeiro, depois o segundo, o terceiro e assim sucessivamente
- Apenas um aceso por vez

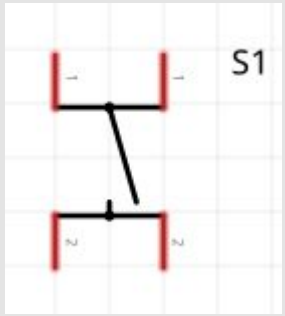
Entrada digital

- Já aprendemos como controlar uma saída digital
- Agora vamos ver como ler uma entrada
- Vamos utilizar um *push button*



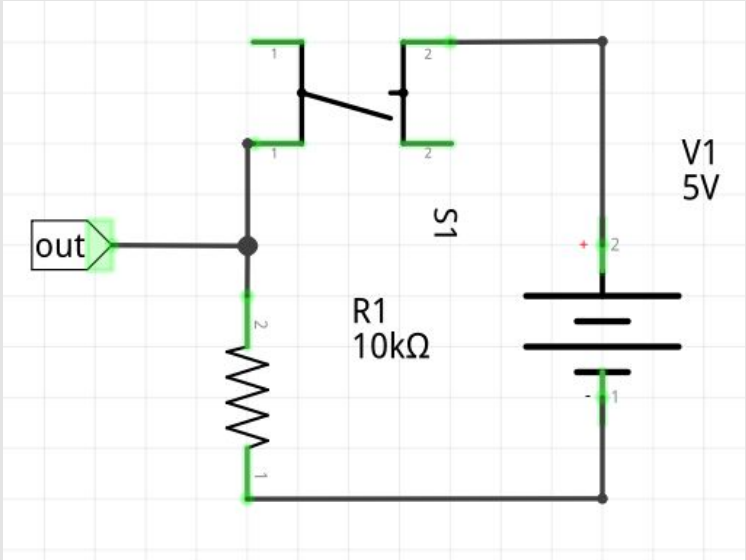
Entrada digital

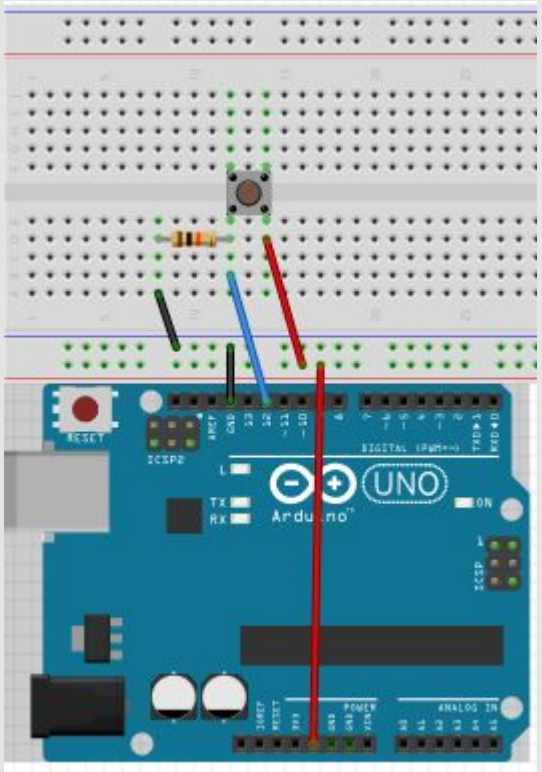
- Como funciona?
  - É bem simples:
  - Enquanto o botão não está sendo apertado não há contato entre os pares de terminais (perninhas)
  - Enquanto pressionamos o contato é fechado



Entrada digital

- E como ligamos ele no Arduino?
  - (Para que serve este resistor?)





Portas digitais - Entrada

- `pinMode(13, INPUT)` → `DDRB &= ~(0 << DDB5)`
- `digitalRead(13)` → `int i = (PINB & (1 << PINB5))`
  - Se há tensão no pino PB5 `i ≠ 0`, caso contrário `i = 0`

Manipulação bit-a-bit

- Lendo um bit em um byte “`PINB & (1 << PINB5)`”

PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
(1 << PINB5)	0	0	1	0	0	0	0	0
& (e)	0	0	PINB5	0	0	0	0	0

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, INPUT) ;
}


void loop() {
  if (digitalRead(12)) {
    digitalWrite(13, HIGH) ;
  } else {
    digitalWrite(13, LOW) ;
  }
}
```

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int main() {
    //Configura a porta B5 como output
    //O macro _BV(DDB5) equivale a (1 << DDB5)
    DDRB |= 1 << DDB5 ;
    //Configura a porta B4 como input
    DDRB &= ~(0 << DDB4) ;

    while (1) {
        //Se a entrada for nível lógico alto acende o LEd
        if (PINB & (1 << PINB4)) {
            PORTB |= (1 << PORTB5) ;
        } else {
            //Caso contrário, apaga
            PORTB &= ~(1 << PORTB5) ;
        }
    }
    return 0 ;
}
```

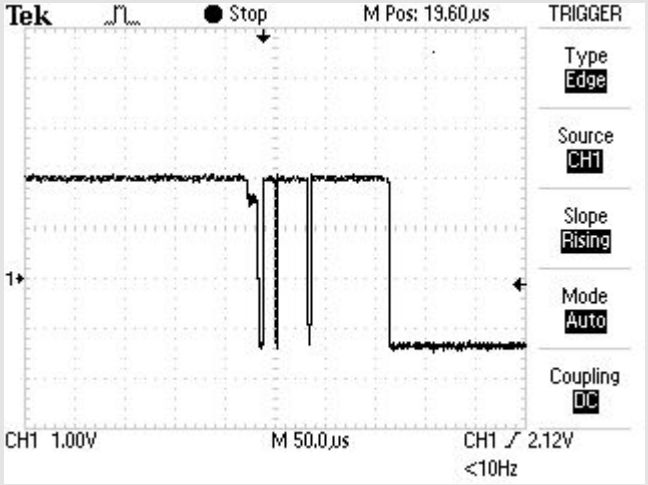
Exercício 02

- Montar um LED e um botão no Arduino (pode-se utilizar o LED da própria placa)
- Ao apertar o botão o LED muda de estado (aceso → apaga; apagado → acende)

Entrada digital

- Debouncing
  - Como apertar o botão é um processo mecânico, é gerado ruído
  - Este ruído pode ser interpretado de maneira errada pelo Arduino

- Sugestões?

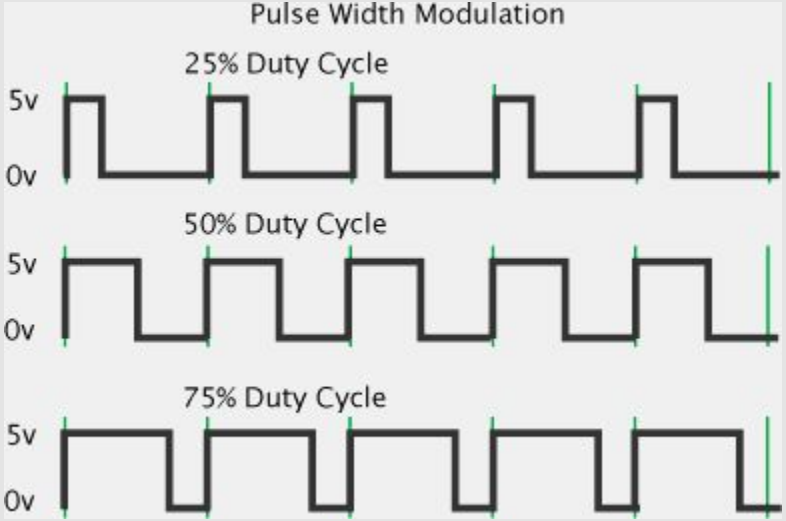


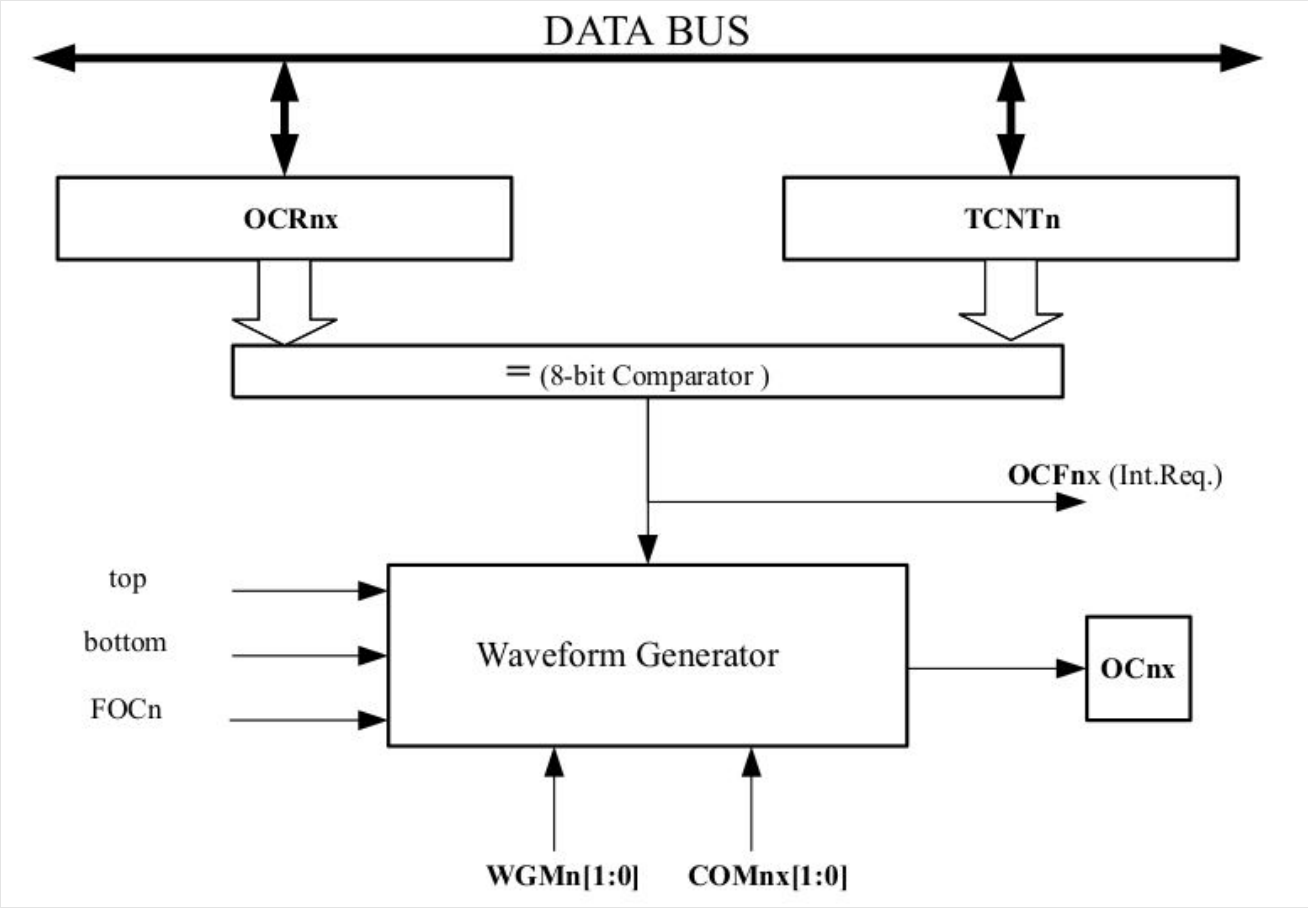
Exercício 03

- Fazer um contador binário com três LEDs e dois botões
- Um botão incrementa
- O outro decrementa

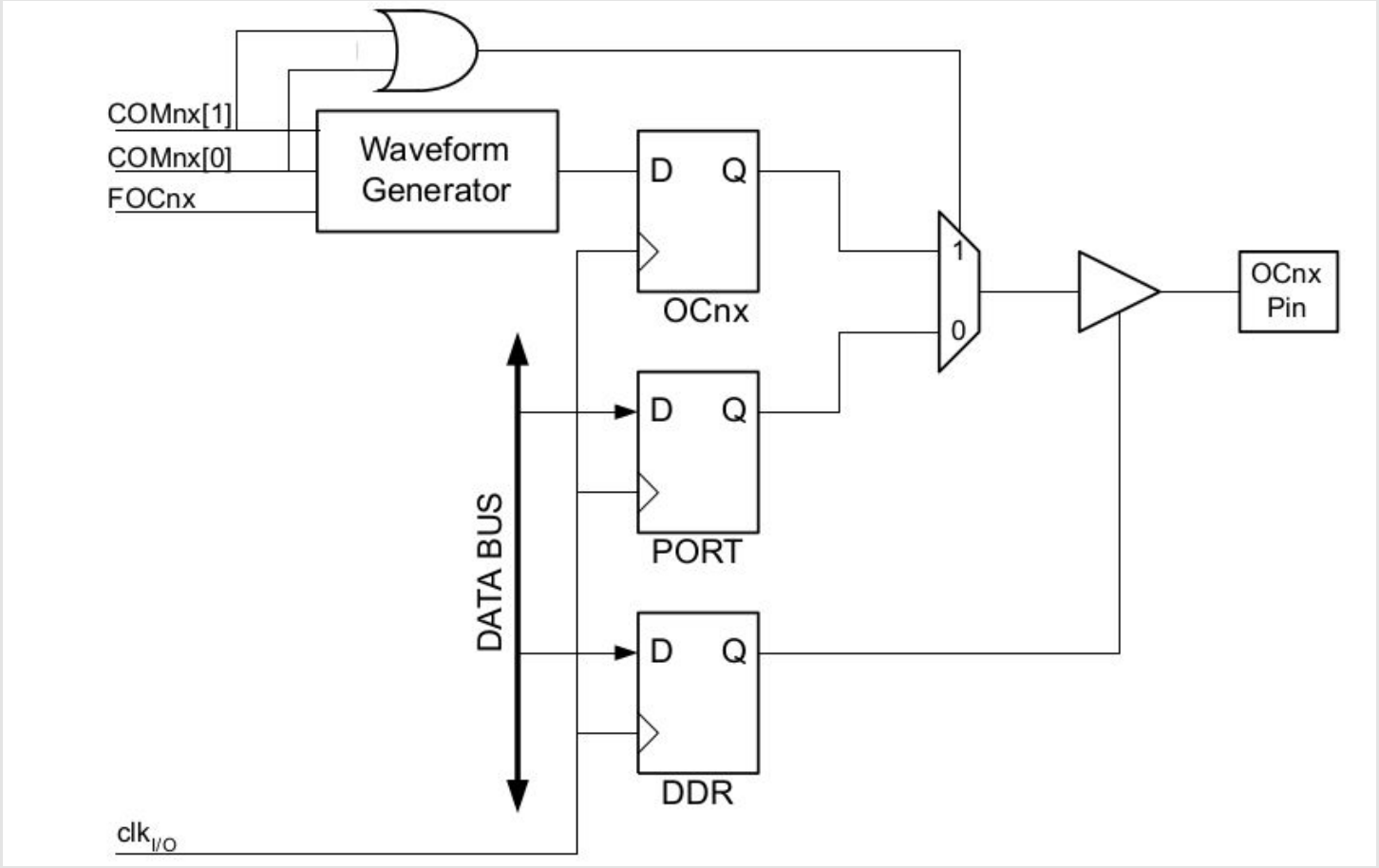
Saída analógica

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





Pag. 129
Datasheet



Pag. 130
Datasheet

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 6 - 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`

Name: TCCR0A
Offset: 0x44
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x24

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Table 19-4. Compare Output Mode, Fast PWM(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

Pag. 138
Datasheet

Name: TCCR0B
Offset: 0x45
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Table 19-9. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on (1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Pag. 140
Datasheet

CA02	CA01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Pag. 142
Datasheet

```
unsigned char dutyCycle = 0 ;

void setup() {
  pinMode(6, OUTPUT);
}


void loop() {
  analogWrite(6, dutyCycle) ;
  dutyCycle += 1 ;
  delay(20) ;
}
```

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>


int main() {
    //Configura a porta D6 como saída
    DDRD |= _BV(PD6) ;

    //Seta o duty cycle para 0%
    OCR0A = 0x00 ;
    //Configura o modo do PWM
    TCCR0A = (1 << COM0A1) | (1 << WGM01) | (1 << WGM00) ;
    //Configura a frequência do timer
    TCCR0B = (1 << CS01) ;


    while (1) {
        //Aumenta o duty cycle a cada 20ms
        OCR0A += 1 ;
        _delay_ms(20) ;

    }
    return 0 ;
}
```

Exercício 04

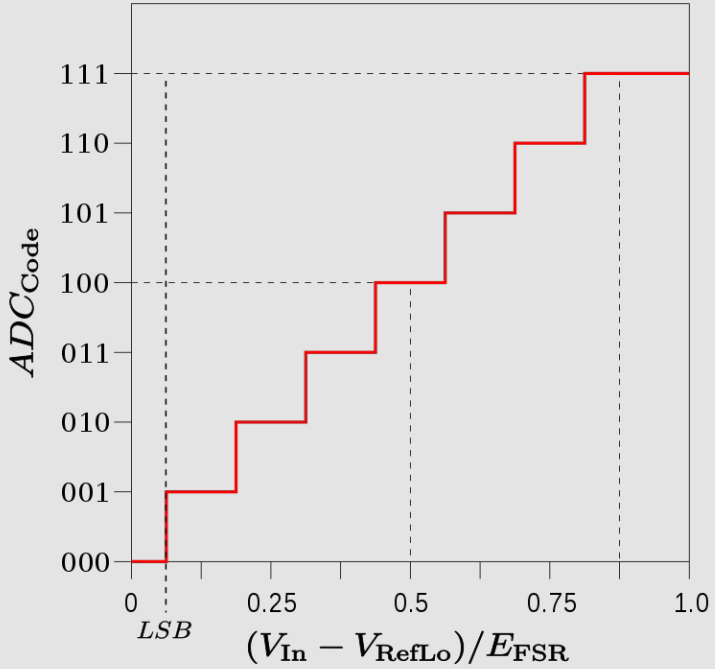
- Ligar um LED na porta 6 e dois botões
- Um botão incrementa a intensidade do LED
- O outro decrementa

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



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!!

Name: ADMUX
Offset: 0x7C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR		MUX3	MUX2	MUX1	MUX0
Access	R/W	R/W	R/W		R/W	R/W	R/W	R/W
Reset	0	0	0		0	0	0	0

REFS[1:0]	Voltage Reference Selection
00	AREF, Internal V_{ref} turned off
01	AV_{CC} with external capacitor at AREF pin
10	Reserved
11	Internal 1.1V Voltage Reference with external capacitor at AREF pin

Pag. 318
Datasheet

MUX[3:0]	Single Ended Input
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
0110	ADC6
0111	ADC7
1000	Temperature sensor

Pag. 318
Datasheet

Name: ADCSRA
Offset: 0x7A
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

- ADEN - Habilita a conversão
- ADSC - Começa a conversão
- ADIF - Sinaliza o fim da conversão

Pag. 319
Datasheet

ADPS[2:0]	Division Factor
000	2
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Quanto menor o clock mais precisão!

Pag. 320
Datasheet

Resultado

- ADLAR = 0

ADCH	0	0	0	0	0	0	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

Resultado

- ADLAR = 1

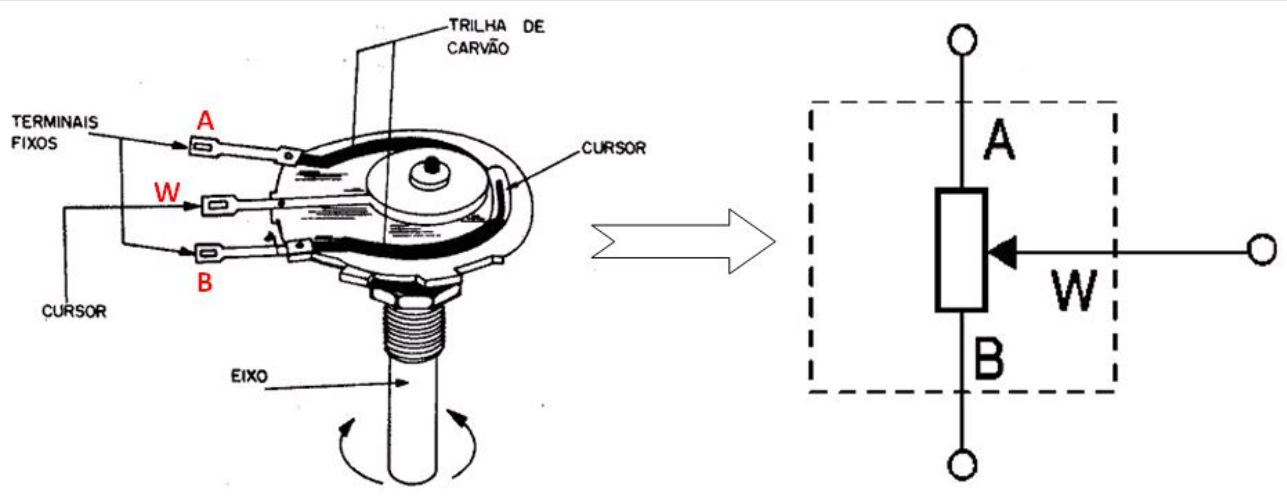
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	0	0	0	0	0	0

Qual é a diferença?

- O dado convertido não é atualizado até o registrador ADCH ser lido
- Se não for necessário mais do que 8 bits de precisão pode-se ajustar o resultado para esquerda
- Assim não é necessário ler o ADCL

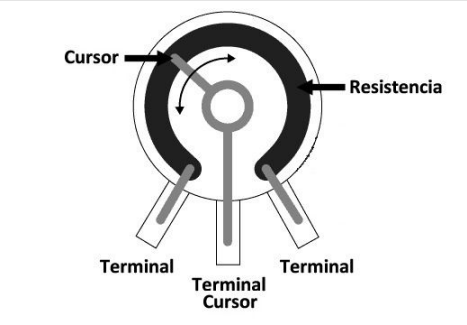
Entrada analógica

- O que é um potenciômetro?
  - Componente que permite variação na resistência



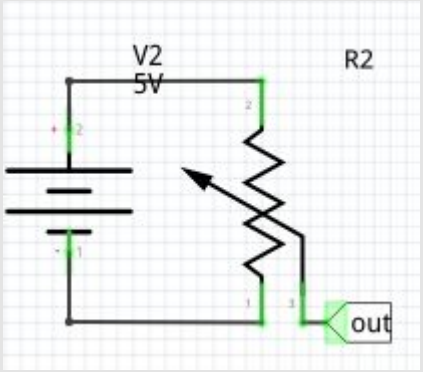
Entrada analógica

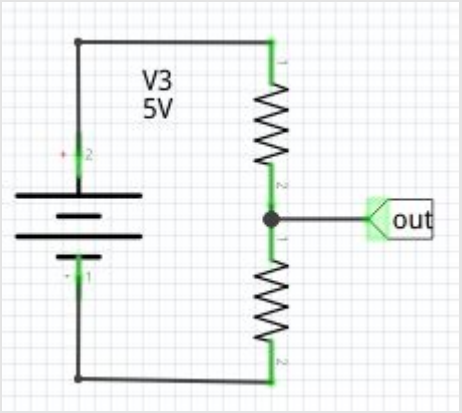
- Como utilizar:
  - O potenciômetro tem três terminais
  - A resistência entre as duas extremidades é sempre fixa
  - A resistência entre o pino do meio e qualquer uma das extremidades é variável
  - Conseguimos variar essa resistência girando o pino central



Entrada analógica

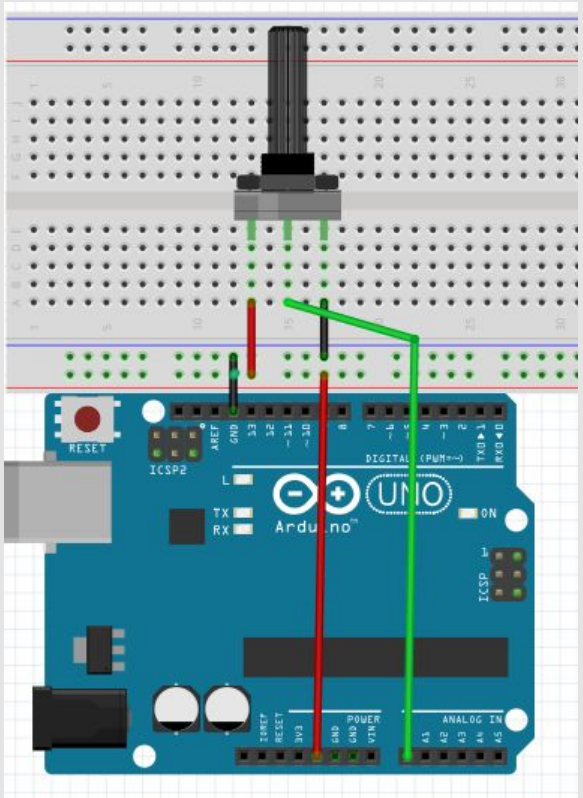
- Utilizamos o potenciômetro como um divisor de tensão
  - Conforme variamos a resistência do terminal central, a tensão neste terminal varia





Entrada analógica

- Conectamos o terminal central a uma das portas analógicas (A0 ~ A5)



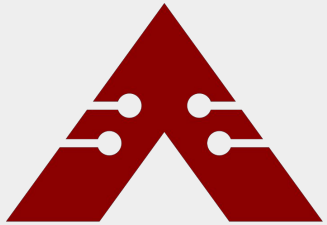
```
void setup() {
  pinMode(13, OUTPUT);
}


void loop() {
  int result = analogRead(1) ;
  if (result > 512) {
    digitalWrite(13, HIGH) ;
  } else {
    digitalWrite(13, LOW) ;
  }

}
```

```
int main() {
    //Configura a porta D6 como saída
    DDRB |= _BV(PB5) ;

    //Vref = AVcc; Bits ajustados a esquerda; Canal 0
    ADMUX = (1 << REFS0) | (1 << ADLAR) | (1 << MUX0);
    //Habilita a conversão AD
    ADCSRA  = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0) ;

    unsigned char result = 0 ;
    while (1) {
        //Inicia a conversão AD
        ADCSRA |= (1 << ADSC) ;
        //Espera a conversão terminar
        while (!(ADCSRA & (1 << ADIF))) ;
        //Configura o duty cycle conforme a entrada analógica
        result = ADCH ;

        if (result > 127) {
            PORTB |= (1 << PORTB5) ;
        } else {
            PORTB &= ~(1 << PORTB5) ;
        }
        //Espera um pouco antes da próxima conversão
        _delay_ms(10) ;
    }
    return 0 ;
}
```

Exercício 05

- Ler uma entrada analógica de um potenciômetro
- Mudar a intensidade de um LED conforme o valor lido

Obrigado!



ADA

PROJETOS EM ENGENHARIA
DE COMPUTAÇÃO