

## Microcontroladores AVR – Conceitos básicos

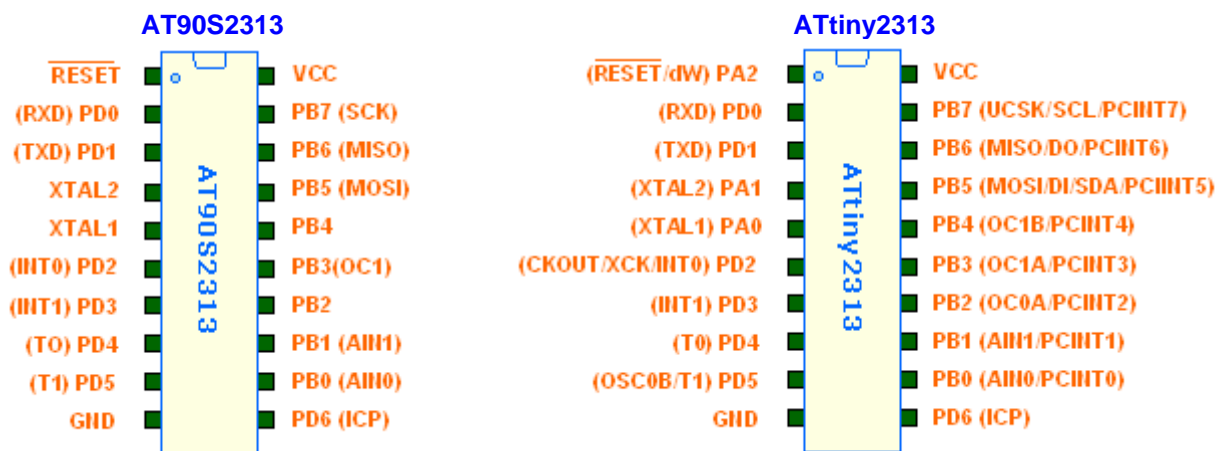
### Breve introdução

O surgimento da microeletrônica no início da década de 70 possibilitou a criação de processadores em uma única pastilha ou chip. Porém, só era possível processar as informações com o auxílio de periféricos como dispositivos de entrada e saída (I/O), memórias etc. Surge o computador.

A crescente integração e a necessidade logo permitiu encapsular um microprocessador, memórias e periféricos de entrada e saída em um único chip, surgindo assim o microcontrolador e uma nova era para eletrônica, a era dos microcontroladores.

### O microcontrolador AVR

Nosso curso terá início com o microcontrolador AT90S2313 ou Attiny2313 (versão melhorada) e com o tempo será substituído por outros de maior porte como o ATmega 8 e o ATmega16.

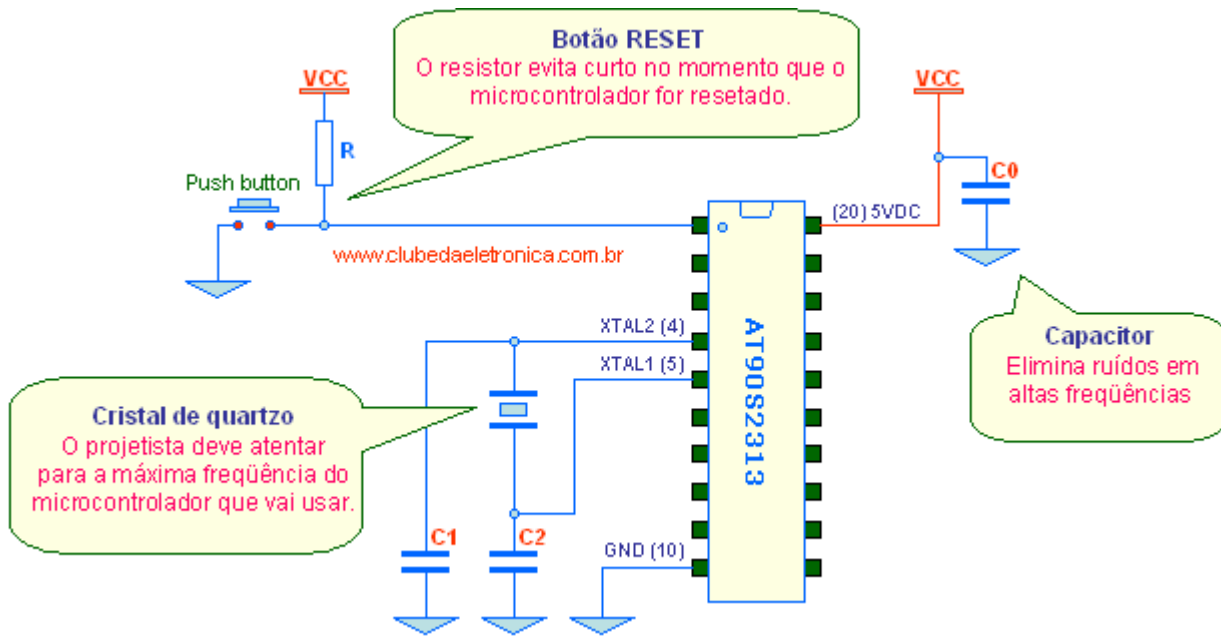


### Descrição e conexão de alguns pinos:

- **VCC** ⇒ Alimentação positiva 5,0 VDC  
 Lembre-se: o microcontrolador é um componente que trabalha com grandezas digitais (1 e 0). Por isso não ultrapasse os valores determinados, se não poderá danificar o componente.
- **GND** ⇒ Terra ou referência
- **RESET** ⇒ Em nível lógico baixo, o micro controlador para de executar as instruções em nível alto ele continua. Lembrando: Baixo = 0 VDC e Alto = 5,0 VDC  
 Importante: Para efetuar o reset o pino deve estar em nível 0 em pelo menos 50 ns ou mais.
- **XTAL1 e 2** ⇒ Nestes pinos deve ser conectado um cristal oscilador que dará a base de tempo para o processamento das informações.  
 Os valores dos capacitores C1 e C2 estão entre 17pF e 33pF. Lembrar que estes capacitores são cerâmicos.

Descrever todos os pinos seria uma tarefa cansativa, tanto para o leitor como para o autor, então eles serão descritos à medida que estiverem sendo usados.

**Conectando a alimentação, o clock e o reset do microcontrolador.**



**Explorando os I/O dos microcontroladores AVR**

Os microcontroladores possuem portas ou "PORTS" que são pinos bidirecionais, ou seja, podem ser configuradas, via software, como entrada ou saída. O AT90S2313 tem dois conjuntos de portas que são:

**PORTB ⇒ Corresponde a oito pinos que vão do PB0 ao PB7**

Três registros de oito bits são responsáveis pelo controle e configuração destas portas e estão associados a cada pino do PORTB. Estes registros são:

❑ **PORTB - Data register (Registro de dados)**

Comanda os pinos configurados como saída, ou seja, envia "0" ou "1".

BIT	7	6	5	4	3	2	1	0
\$18 (\$38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

❑ **DDRB Data Direction Register (Registro de direção de dados)**

Configura a porta se "0" será entrada, se "1" será saída.

BIT	7	6	5	4	3	2	1	0
\$17 (\$37)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

❑ **PINB Input Pins Address (Registro de entrada de endereço)**

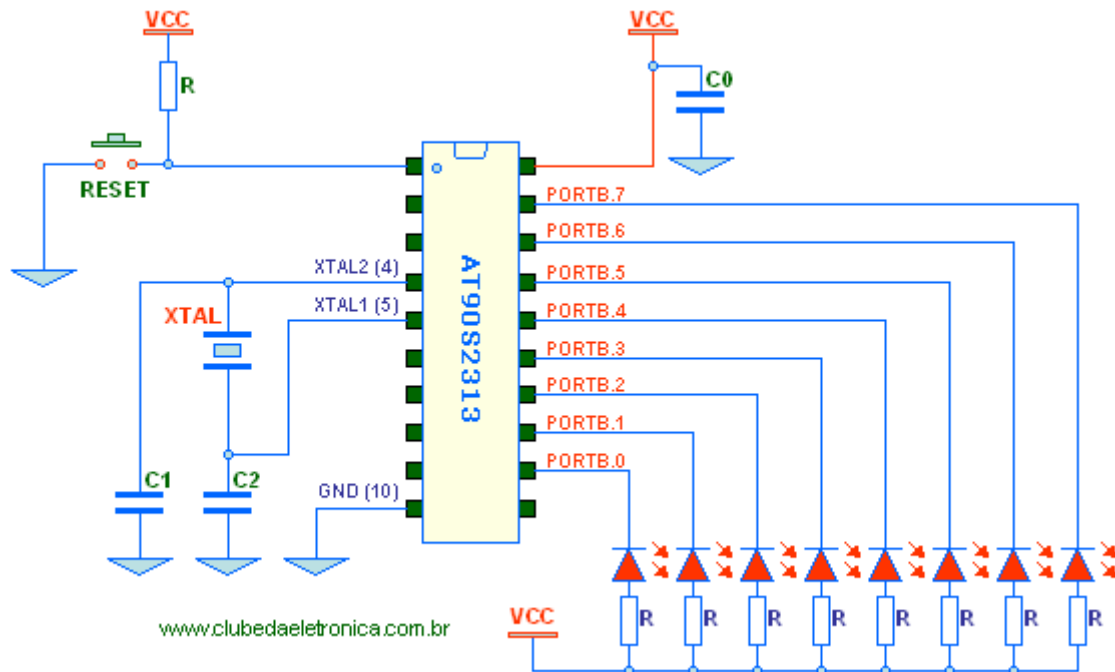
Comanda os pinos configurados como entrada, ou seja, recebe "0" ou "1".

BIT	7	6	5	4	3	2	1	0
\$16 (\$36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Read/write	R	R	R	R	R	R	R	R
Initial value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

## Uma explicação mais prática.

Para maior clareza sobre os registros é necessário conhecer a placa de desenvolvimento, neste caso, adicionaremos ao PORTB um conjunto de oito LEDs.

Placa de testes



Os LEDs, por razões óbvias, são saídas. Logo, todo o PORTB deve ser configurado como saída.

$DDRB=0b11111111$

Notem a polaridade dos LEDs e observe que já existe uma tensão de 5VDC no anodo de cada um deles, ou seja, para acendê-los deveremos enviar 0 para este PORT. Assim, se o projetista deseja acender todos os LEDs deverá enviar "0" para todo o PORTB.

$PORTB=0b00000000$

Se desejar acender somente o LED conectado ao pino 12 (PORTB. 0) somente este PORT deverá estar em "0".

$PORTB=0b11111110$

Programa exemplo 01:

Acendendo todos os LEDs simultaneamente.

```
#include <90S2313.h>           // chama biblioteca do microcontrolador utilizado.

void main (void)
{
  DDRB=0b11111111;           // configura todo o PORTB como saída.
  PORTB=0b00000000;         // envia "0" para todo o PORTB acendendo os LEDs
}
```

Se o projetista desejar acender somente o LED conectado ao PORTB.4 (pino 16) basta modificar o código. Vejamos:

```
#include <90S2313.h>           // chama biblioteca do microcontrolador utilizado.

void main (void)
{
  DDRB=0b11111111;           // configura todo o PORTB como saída.
  PORTB=0b11101111;         // envia "0" somente para o PORTB.4 acendendo somente este LED
}
```

Praticando ...

- 1- Elabore um programa que acenda os LEDs somente os PORT pares.
- 2- Elabore um programa que aciona o PORTB.6

### PORTD ⇒ Corresponde a oito pinos que vão do PD0 ao PD6

Três registros de oito bits são responsáveis pelo controle e configuração destas portas e estão associados a cada pino do PORTD. Estes registros são:

#### PORTD - Data register (Registro de dados)

Comanda os pinos configurados como saída, ou seja, envia "0" ou "1".

BIT	7	6	5	4	3	2	1	0
\$12 (\$32)	---	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTB1	PORTD0
Read/write	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	---	0	0	0	0	0	0	0

#### DDRD Data Direction Register (Registro de direção de dados)

Configura a porta se "0" será entrada, se "1" será saída.

BIT	7	6	5	4	3	2	1	0
\$11 (\$31)	---	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Read/write	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	---	0	0	0	0	0	0	0

## PIND Input Pins Adress (Registro de entrada de endereço)

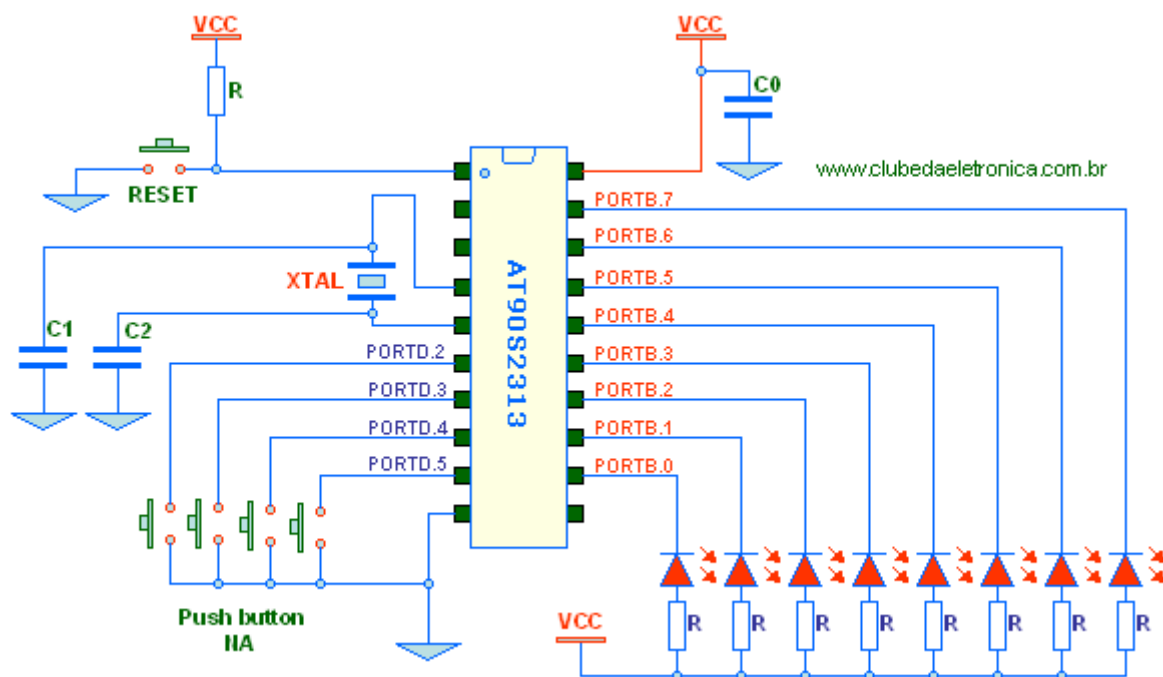
Comanda os pinos configurados como entrada, ou seja, recebe "0" ou "1".

BIT	7	6	5	4	3	2	1	0
\$10 (\$30)	---	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Read/write	---	R	R	R	R	R	R	R
Initial value	---	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Lembrando que o bit 7 desses registros não tem valor relevante algum, pois o controle e configurações são do bit 0 ao bit 6.

### Uma explicação mais prática.

Reforçando que só é possível programar conhecendo a placa de desenvolvimento ou de aplicação, uma vez que já adicionamos LEDs ao PORTB adicionaremos agora quatro botões push buttons normalmente abertos ao PORTD.



Os botões são entradas de informações, ou seja, os pinos onde elas estão conectadas devem ser entradas, neste caso do PORTD.2 ao PORTD.5 devem ser configurados, via software, como entrada.

$DDRD.2=0$   
 $DDRD.3=0$   
 $DDRD.4=0$   
 $DDRD.5=0$

Estão conectadas chaves nos PORTD do 2 ao 5.  
Assim, somente estas serão configuradas como entrada.

As chaves estão conectadas ao (GND) terra do microcontrolador e a alguns pinos do PORTD, ou seja, se pressionarmos, por exemplo, o botão conectado ao PORTD.5 este recebe um sinal de terra ou "0". Porém, se o botão estiver em sua condição, normalmente aberto, o PORTD.5 não recebe nem "0" (terra) nem "1" 5VCC. Assim, devemos programá-lo no estado inicial "1" e se receber "0" executará uma função.

<b>PORTD.2=1</b> <b>PORTD.3=1</b> <b>PORTD.4=1</b> <b>PORTD.5=1</b>	}	Configurados inicialmente como "1", ou seja, Chave não pressionada "1". Chave pressionada "0".
--	---	--

Programa exemplo 02:

Pressionado o botão conectado ao PORTD.2, acende o LED conectado ao PORTB.7.

Antes de iniciarmos o programa devemos recapitular alguns comandos básicos de linguagem C.

**O comando if – else** ⇒ O comando if é utilizado quando se deve optar entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.

**Estrutura:**

```
if (condição)
  comando A
else
  comando B
```

**Descrição:**

Se a condição proposta for verdadeira o comando A será executado, senão o comando B será executado.

Agora, vamos ao programa:

```
#include <90S2313.h>           // chama biblioteca do microcontrolador utilizado.

void main (void)
{
  // configurando os pinos

  DDRD. 2=0;                  // configura o PORTD.2 como entrada.
  PORTD. 2=1,                 // comanda o estado inicial do PORTD.2.

  DDRB. 7=1;                  // configura o PORTB.7 como saída entrada.
  PORTB. 7=1;                 // garante que a saída do PORTB.7 será inicialmente 1 (LED apagado)

  // fim das configurações

  While (1)                   // loop infinito.

  // Programa principal.

  {
    if (PIND.2==0)            // Se o PIND.2 for igual a "0" ( botão pressionado).
      PORTB.7=0;              // O PORTB.7 será "0" e o LED acenderá
    else                       // Senão
      PORTB.7=1;              // O PORTB.7 será "1" e o LED não acenderá
  }
}
```

**Praticando ...**

- 3- Elabore um programa que pressionando o botão 3 acende o LED 7.
- 4- Elabore um programa que pressionando o botão 2 acende todos os Leds.

## O ATtiny2313 (uma versão melhorada)

O AT90S2313 acabou sendo substituído pelo Attiny2313, porém nada do que já foi dito deve ser mudado, somente acrescentado, por exemplo, ao Attiny2313 foi acrescentado um PORTA que vai do PORTA.0 ao PORTA.2, assim, de 15 passou-se para 18 portas de I/O

**PORTA ⇒ Corresponde a três pinos que vão do PA0 ao PB2**

Três registros de oito bits são responsáveis pelo controle e configuração destas portas e estão associados a cada pino do PORT A. Estes registros são:

### PORTA - Data Register (Registro de dados)

Comanda os pinos configurados como saída, ou seja, envia “0” ou “1”.

BIT	7	6	5	4	3	2	1	0
	----	----	----	----	----	PORTA2	PORTA1	PORTA0
Read/write	----	----	----	----	----	R/W	R/W	R/W
Initial value	----	----	----	----	----	0	0	0

### DDRA Data Direction Register (Registro de direção de dados)

Configura a porta se “0” será entrada, se “1” será saída.

BIT	7	6	5	4	3	2	1	0
	----	----	----	----	----	DDA2	DDA1	DDA0
Read/write	----	----	----	----	----	R/W	R/W	R/W
Initial value	----	----	----	----	----	0	0	0

### PINA Input Pins Adress (Registro de entrada de endereço)

Comanda os pinos configurados como entrada, ou seja, recebe “0” ou “1”.

BIT	7	6	5	4	3	2	1	0
	----	----	----	----	----	PINA2	PINA1	PINA0
Read/write	----	----	----	----	----	R/W	R/W	R/W
Initial value	----	----	----	----	----	N/A	N/A	N/A

Outros microcontroladores estudados neste curso, como o Atmega16 chegam ter 32 portas de I/O o que sem dúvida auxilia na elaboração de projetos de maior complexibilidade.

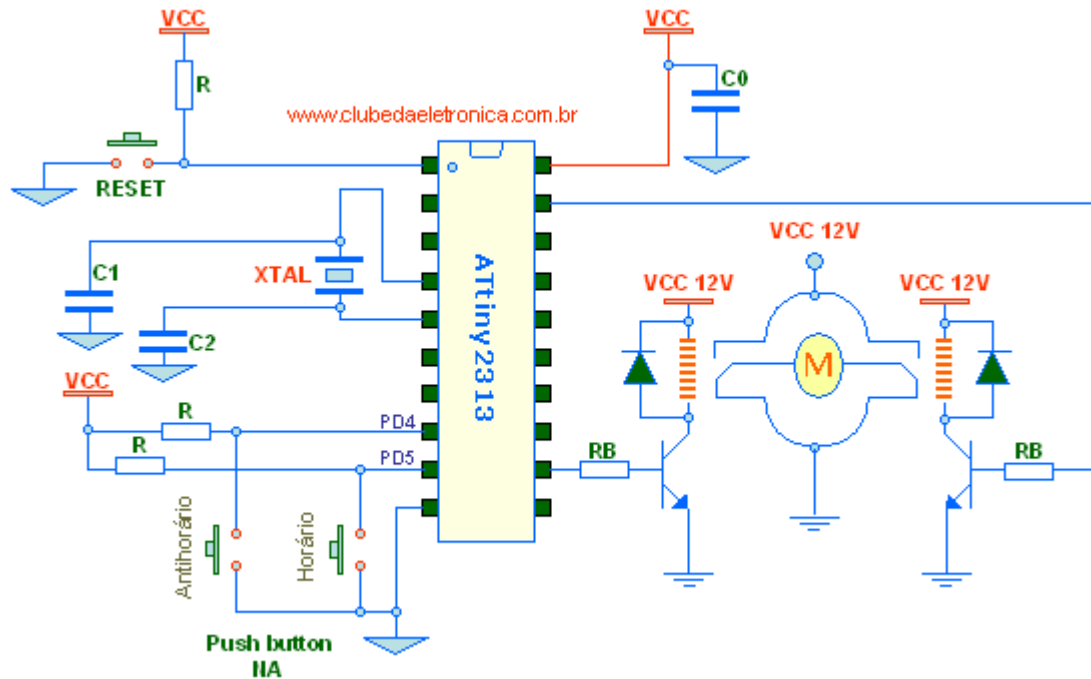
## Reforçando conceitos sobre o hardware

Como já foi dito, só é possível o projetista programar um microcontrolador se ele conhecer o hardware de aplicação ou desenvolvimento. Pensando nisso, o hardware e o microcontrolador serão mudados, a fim de reforçar o conjunto software hardware.

### A Idéia

Usaremos agora o attiny2313 para controlar o sentido de um motor DC, ou seja, se o botão horário for pressionado ele gira no sentido horário se o botão anti-horário for pressionado ele gira no sentido anti-horário.

## A nova placa de aplicação



Notem que, somente as saídas PB0 e PB7 do Attiny2313 foram utilizadas e nelas estão conectados transistores NPN (necessitam receber "1" para chaveamento). As entradas estão ligadas diretamente a um resistor e a fonte VCC, ou seja, já estão recebendo "1" os botões se pressionados enviarão "0".

Agora, vamos ao programa:

```
#include <tiny2313.h>           // chama biblioteca do microcontrolador utilizado.

void main (void)
{
    // configurando os pinos

    DDRD. 4=0;                 // configura o PORTD.4 como entrada.
    DDRD. 5=0;                 // configura o PORTD.5 como entrada.

    DDRB. 0=1;                 // configura o PORTB.0 como saída
    DDRB. 7=1;                 // configura o PORTB.7 como saída

    // fim das configurações

    While (1)                  // loop infinito.

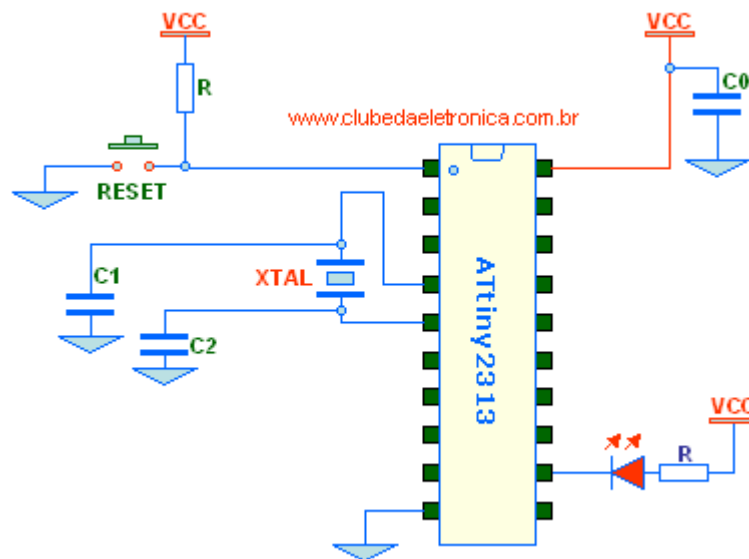
    // Programa principal.
    {
        if (PIND.4==0)         // Se o PIND.4 for igual a "0" (botão pressionado).
            PORTB.7=1;         // O PORTB.7 será "1" e o sentido anti-horário.
        else                    // senão
            PORTB.7=0;         // o PORTB.7 será "0" motor parado.

        if (PIND.5==0)         // Se o PIND.5 for igual a "0" (botão pressionado).
            PORTB.0=1;         // O PORTB.5 será "1" e o sentido horário.
        else                    // senão
            PORTB.0=0;         // o PORTB.7 será "1" motor parado.
    }
}
```

## Mais conceitos sobre linguagem C

A função `delay ()` ⇒ Permite ao programador inserir uma pausa entre uma execução e outra, porém para que a função seja reconhecida devemos inserir a biblioteca `delay.h`. Vejamos um exemplo.

Gerando Clock com o Attiny2313 (Pisca-Pisca)



```
#include <tiny2313.h>           // chama biblioteca do microcontrolador utilizado.
#include <delay.h>             // chama biblioteca delay.

void main (void)
{
  // configurando os pinos.

  DDRB.0=1;                    // configura o PORTB.0 como saída.

  // Programa principal.

  While (1)                    // loop infinito.
  {
    PORTB.0=1;                 // O PORTB.7 será "1" e o LED apagado.
    Delay_ms(200);             // Se mantém apagado por 200 milisegundos.

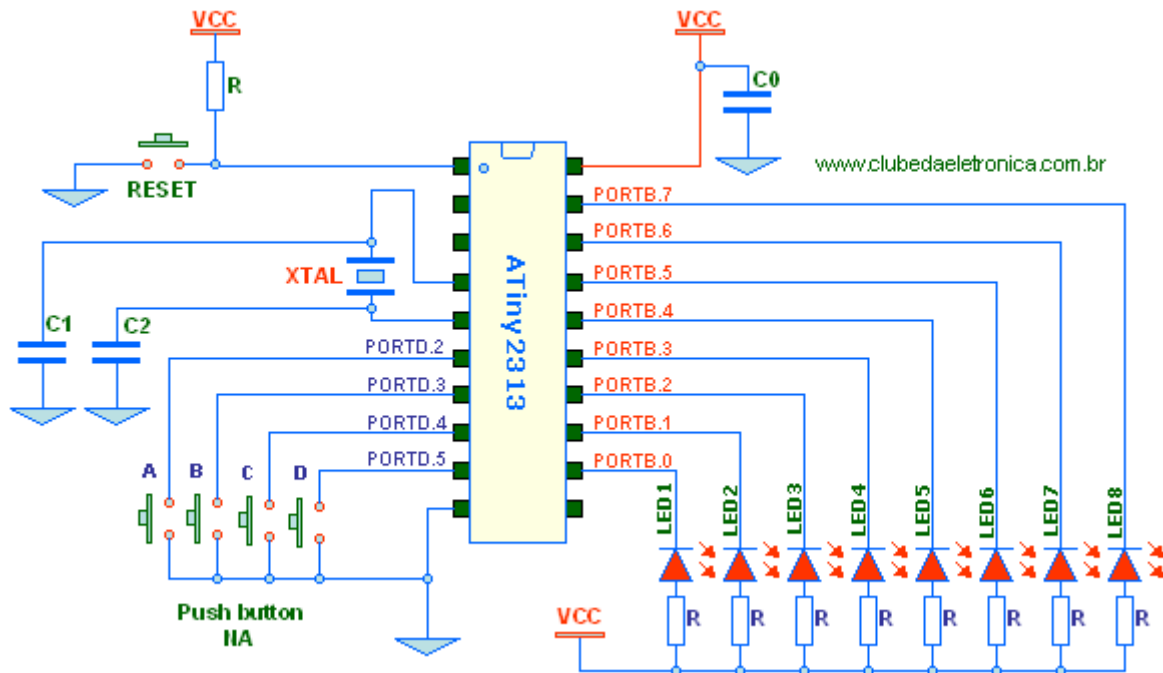
    PORTB.0=0;                 // O PORTB.7 será "0" e o LED aceso.
    Delay_ms(200);             // Se mantém aceso por 200 milisegundos.
  }
}
```

### Praticando ...

- 5- Elabore um pisca-pisca em que todos os LEDs ficam piscando num intervalo de 400ms.
- 6- Elabore um programa que os LEDs fiquem alternando entre pares e impares num intervalo de 100ms.

## Um seqüencial de 8 LEDs com Atiny2313

Agora que já conhecemos mais uma função da linguagem C podemos montar um circuito seqüencial, sempre atentando para o hardware.



```
#include <tiny2313.h> // chama biblioteca do microcontrolador utilizado.
#include <delay.h> // chama biblioteca delay.

void main (void)
{
    // Configurações iniciais

    DDRB=0b11111111; // configura todo o PORTB como saída.
    PORTB=0b11111111; // garante que os leds estejam inicialmente todos desligados.

    // programa principal

    While (1) // loop infinito.
    {
        PORTB=0b11111110; // envia "0" para o PORTB.0 e o LED1 acende .
        Delay_ms(100);

        PORTB=0b11111101; // envia "0" para o PORTB.1 e o LED2 acende .
        Delay_ms(100);

        PORTB=0b11111011; // envia "0" para o PORTB.2 e o LED3 acende .
        Delay_ms(100);

        PORTB=0b11110111; // envia "0" para o PORTB.3 e o LED4 acende .
        Delay_ms(100);

        PORTB=0b11110111; // envia "0" para o PORTB.4 e o LED5 acende .
        Delay_ms(100);

        PORTB=0b11101111; // envia "0" para o PORTB.5 e o LED6 acende .
        Delay_ms(100);
    }
}
```

```

PORTB=0b10111111; // envia "0" para o PORTB.6 e o LED7 acende .
Delay_ms(100);

PORTB=0b01111111; // envia "0" para o PORTB.7 e o LED8 acende .
Delay_ms(100);
}
}

```

### Praticando ...

- 7- Elabore um programa para que os LEDs acendam um a um num intervalo de 100ms.
- 8- Elabore um programa para os LED acenderem um a um e apaguem um a um.

### Um seqüencial de 8 LEDs com Attiny2313

Agora que já conhecemos mais uma função da linguagem C podemos montar um circuito seqüencial, porém, utilizaremos botões, assim, se o **Botão A** for pressionado a seqüência será do LED1 ao LED8 e se o **Botão A** não for pressionado os LEDs deverão permanecer todos apagados.

Solução:

Agora será necessário o comando **if-else** e a função **delay**. Vejamos o programa:

```

#include<tiny2313.h>
#include<delay.h>

void main (void)
{
// configurando os pinos

DDRB=0b11111111;           //configura todo o PORTB como saída
PORTB=0b11111111;         //garante que a saída seja inicialmente 1 (apagados)

DDRD.2=0;                 //configura o PORTD.2 como entrada
DDRD.3=0;                 //configura o PORTD.2 como entrada

// configurando os pinos

While(1)

if(PIND.2==0)             // se o PIND.2 for "0" executa o que esta entre as chaves
{
PORTB=0b11111110;
delay_ms(100);

PORTB=0b11111101;
delay_ms(100);

PORTB=0b11111011;
delay_ms(100);

PORTB=0b11110111;
delay_ms(100);

PORTB=0b11101111;
delay_ms(100);

PORTB=0b11011111;
delay_ms(100);
}
}

```

```

PORTB=0b10111111;
delay_ms(100);

PORTB=0b01111111;
delay_ms(100);
}

else // senão todo o PORTB vai a zero

PORTB=0b00000000;
}

```

### Praticando ...

9- Elabore um programa que:

- Pressionando o botão 1, o seguinte código será executado: (PORTB.0 ao PORTB.3) 0001, 0010, 0100 e 1000.
- Pressionando o botão 2, o seguinte código será executado: (PORTB.4 ao PORTB.7) 1000, 0100, 0010 e 0001.
- Pressionando o botão 3, o seguinte código será executado: (PORTB.0 ao PORTB.7) 1000 0001, 0100 0010, 0010 0100 e 0001 1000.
- Pressionando o botão 4, o seguinte código será executado: (PORTB.0 ao PORTB.7) 0001 1000, 0010 0100, 0100 0010 e 1000 0001.

**O comando for** ⇒ É um comando de looping (repetição) sua estrutura é bastante semelhante à utilizada em outras linguagens.

#### Estrutura:

```

for (
  Início da variável;
  Terminado desejado para variável;
  Incremento ou decremento da variável
);
Comando;

```

#### Descrição:

Uma variável de controle, geralmente um contador, recebe um valor inicial. O trecho de programa que pertence ao laço é executado e ao final a variável de controle é incrementada ou decrementada e comparada com o valor final que ela deve alcançar. Caso a condição de término tenha sido atingida o laço é interrompido.

### Pisca utilizando e o Attiny e comando for

```

#include<tiny2313.h>

void main (void)
{
  // Declarando as variáveis

  unsigned int tempo; //declara uma variável de 16 bit chamada tempo

  // Configurando as portas

  DDRB=0b11111111; //configura todo o PORTB como saída
  PORTB=0b00000000; //comanda o PORTB para que seja inicialmente "0"

  // Fim das configurações

  while(1) //looping infinito

```

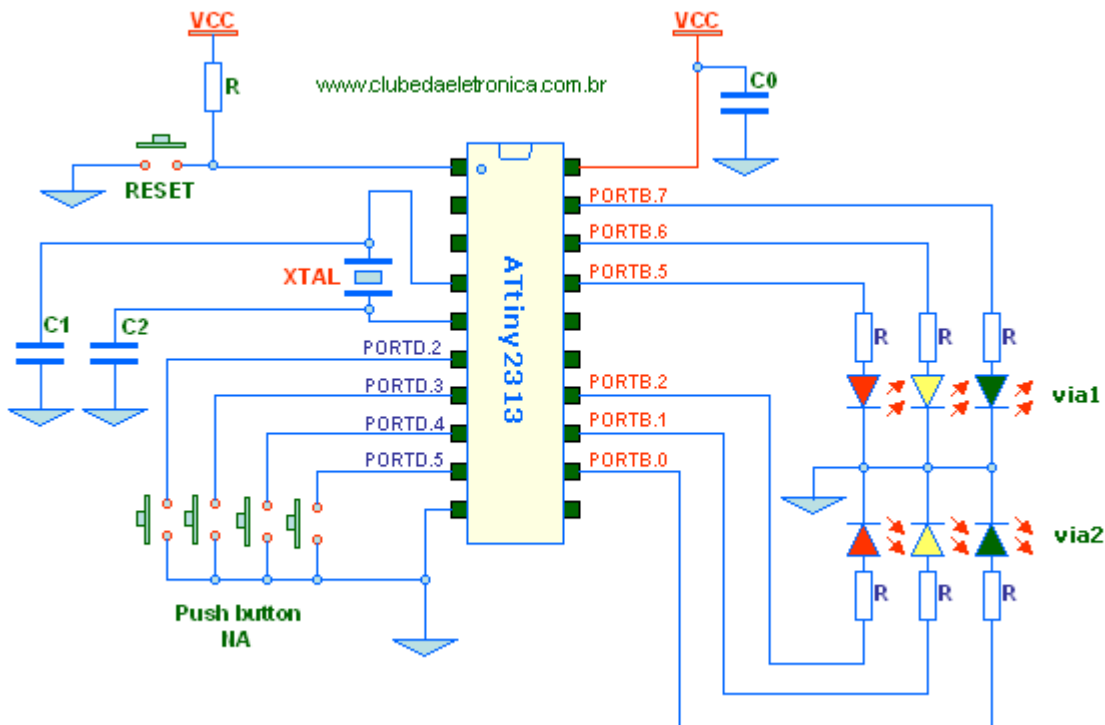
```

{
    for (tempo=0; tempo<=60000; tempo++); //Para o tempo entre 0 e 15ms a saída será 1
    PORTB.1=1;                               //entre 0 e 60000 (15ms) o PORTB.0 será 1

    for (tempo=0; tempo<=60000; tempo++); //Para o tempo entre 0 e 15ms a saída será 0
    PORTB.1=0;                               //entre 0 e 60000 (15ms) o PORTB.0 será 0
}
}

```

### Um semáforo utilizando o Attiny 2313 e comando for



Acima o hardware para desenvolvimento e teste com o Attiny 2313, observem a polarização dos LEDs, neste caso, eles estão com seus catodos ligados à terra (**0**), ou seja, para acende-los devemos mandar via software 5V (**1**) para o PORT correspondente.

```

#include<attiny2313.h>

void main (void)
{
    // Declarando as variáveis

    unsigned int tempo;           //declara uma variável de 16 bit chamada tempo

    // Configurando as portas

    DDRB=0b11111111;             // configura todo o PORTB como saída
    PORTB=0b00000000;           // comanda o PORTB para que seja inicialmente "0"

    // Inicio do programa

    while(1)                      // looping infinito

    {

```

```

for(tempo=0; tempo<=60000; tempo++); // Para o tempo entre 0 e 15ms a saída será 1
PORTB=0b00100001; // Vermelho 2 e Verde 1

for(tempo=0; tempo<=60000; tempo++); // Para o tempo entre 0 e 15ms a saída será 0
PORTB=0b00100010; // Vermelho 2 e Amarelo 1

for(tempo=0; tempo<=60000; tempo++); // Para o tempo entre 0 e 15ms a saída será 0
PORTB=0b10000100; // Verde 2 e Vermelho1

for(tempo=0; tempo<=60000; tempo++); // Para o tempo entre 0 e 15ms a saída será 0
PORTB=0b01000100; // Amarelo 2 e Vermelho1
}
}

```

### Praticando ...

11- Implemente um Botão Liga e um Botão Desliga ao semáforo.

### Mais sobre C - Operadores

Os operadores indicam ao compilador a necessidade de se fazer manipulações matemáticas ou lógicas.

Aritméticos ⇒ São usados para calcular expressões matemáticas. Sendo classificados em duas categorias: os binários ou unários. Os operadores unários atuam na inversão de valores. Veja a tabela abaixo.

Operadores	Descrição
=	Atribuição
+	Soma
-	Subtração
/	Divisão
%	Modulo (obtém o resto da divisão)

Operador unitário	Ação
-	Sinal negativo
+	Sinal positivo

**Incremento e decremento** ⇒ O operador de incremento (++) soma um ao seu operando enquanto que o de decremento (--) subtrai um. Eles podem ser pré-fixados ou pós-fixados conforme mostra a tabela abaixo:

Operadores	Ação
++	Incremento de 1 à variável
--	Decremento de 1 da variável

**Relacionais** ⇒ São responsáveis pelas **comparações** de expressões nos programas. A lista completa se encontra abaixo:

Operadores	Descrição
>	Maior
>=	Maior igual
<	Menor
<=	Menor igual
==	Igualdade
!=	Diferente

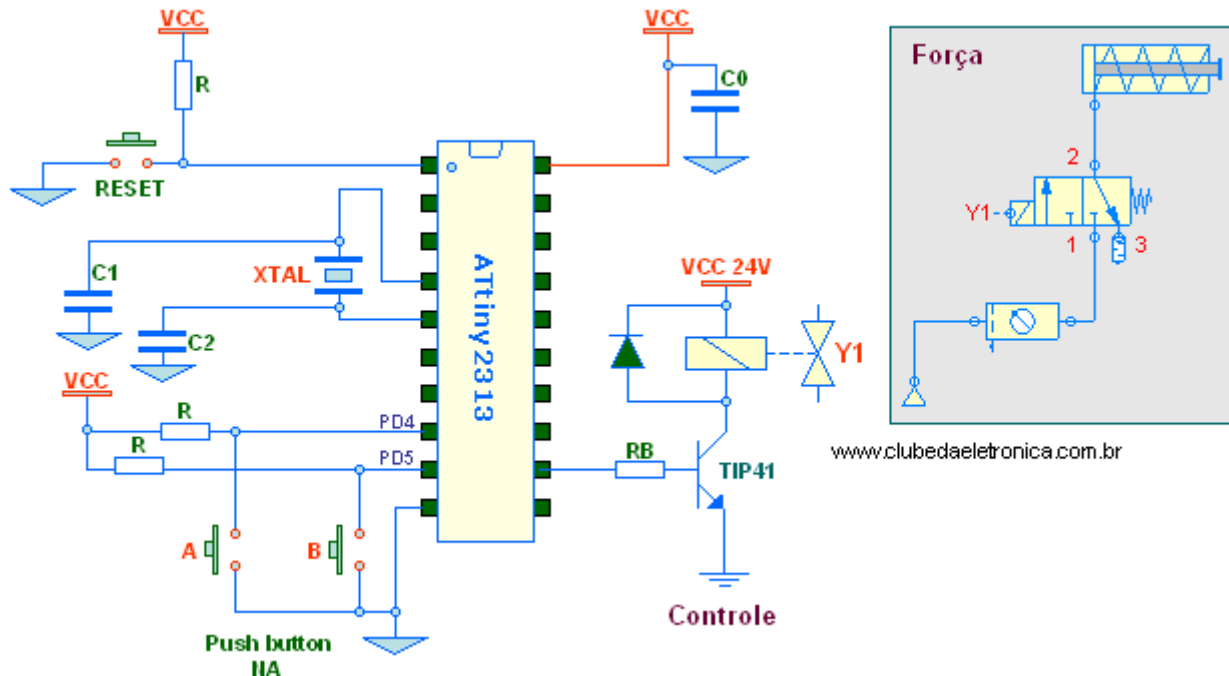
**Lógicos** ⇒ Os operados lógicos servem para interligar mais de uma operação relacional. E assim como os relacionais retornam zero para falso e um para verdadeiro.

Operadores	Descrição
&&	Lógica AND
	Lógica OR
!	Lógica NOT

## Aplicações lógicas - AND

Conhecendo os operadores lógicos, podemos elaborar programas capazes de executar as mesmas funções que as portas lógicas, isso sem o inconveniente de todas aquelas fios para ligação.

### A placa de aplicação e esquema de força



### A aplicação lógica - Prensa

Implemente um sistema de prensagem que, por questões de segurança, deverá haver dois botões B1 e B2 que deverão ser pressionados ao mesmo tempo para atuação.

```
#include<tiny2313.h>

void main (void)
{
  // configurando os pinos

  DDRB.0=1;           //configura o PORTB.0 como saída
  PORTB.0=1;         //garante que a saída seja inicialmente 1 (desativada)

  DDRD.4=0;           //configura o PORTD.2 como entrada
  DDRD.5=0;           //configura o PORTD.2 como entrada

  // Programa principal

  While(1)

  If((PIND.4==0)&& (PIND.5==0)) // se o PIND.4 for "0" e PIND.5 for "0"

    PORTB.0=1;           // envia 1 e satura o transistor energizando a válvula

  else                   // senão

    PORTB.0=0;           // envia 0 e abre o transistor não energizando a válvula

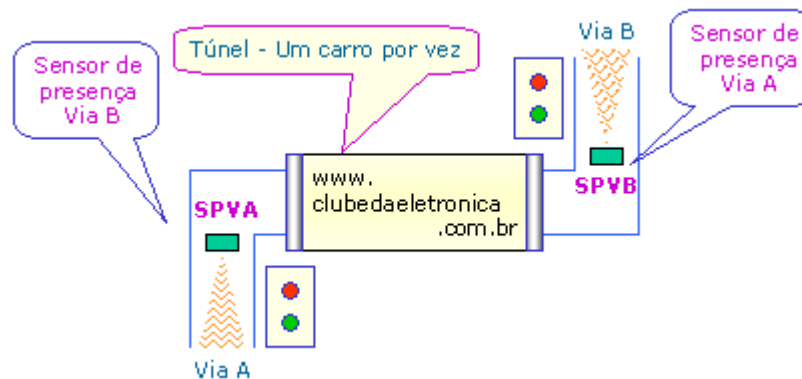
}
```

**Praticando ...**

- 12 – Elabore um programa que execute a função ou exclusivo.  
 13 – Elabore um programa que execute a função ou coincidência.

**Projeto prático 01 – Controle de trafego (resolvido)**

Deseja-se implementar um controle de trafego para um túnel que só permite a passagem de um carro por vez. Veja ilustração:



A prefeitura que encomendou o projeto tem os seguintes critérios:

Quando os sensores detectarem a presença do carro, um nível lógico alto (ON) será enviado ao seu respectivo dispositivo de atuação.

Situação dos  
sensores

Crítérios de projeto

SPVA	SPVB	
OFF	OFF	Se não houver nenhum carro, a <b>via B</b> deverá ser liberada (verde) e a <b>via A</b> bloqueada (vermelho).
OFF	ON	Se o sensor detectar carro na <b>via B</b> , esta será liberada (sinal verde) e a <b>Via A</b> bloqueada (sinal vermelho).
ON	OFF	Se o sensor detectar carro na <b>via A</b> , esta será liberada (sinal verde) e a <b>Via B</b> bloqueada (sinal vermelho).
ON	ON	Se ambos os sensores detectarem carros, a via A deverá ser liberada (sinal verde) e a <b>via B</b> bloqueada (sinal vermelho).

1º Passo – montar a tabela verdade a partir de todas as condições possíveis

SPVA	SPVB
0	0
0	1
1	0
1	1

VMA	VDA	VMB	VDB
1	0	0	1
1	0	0	1
0	1	1	0
0	1	1	0

2º Passo – extrair a tabela verdade das expressões verdadeiras

$$VMA = (SPVA' \cdot SPVB') + (SPVA' \cdot SPVB)$$

$$VMB = (SPVA \cdot SPVB') + (SPVA \cdot SPVB)$$

3º Passo – elaborar o programa

```

/*****
/*      www.clubedaeletrônica.com.br - Microcontroladores AVR */
/*      Projeto: controle de trafego - Autor: Clodoaldo Silva      */
/*      Revisão: 15/04/2009                                       */
*****/

#include<mega8.h>
#include<delay.h>

void main (void)
{
//CONFIGURAÇÕES INICIAIS

DDRB=0b11111111;      // configura port B como saída
PORTB=0b00000000;    // configura o port B para iniciar DESLIGADO

DDRD=0b00000000;    // configura o port D como entrada
PORTD=0b00000000;    // configura o port B para iniciar ABERTA

// DEFININDO VARIÁVEIS

// entradas

#define SPVA PIND.2
#define SPVB PIND.3

//saídas

#define VMA PORTB.5
#define VDA PORTB.4

#define VMB PORTB.2
#define VDB PORTB.1

// INICIO DO PROGRAMA

    while(1)
    {
        if(((SPVA==0)&&(SPVB==0))||((SPVA==0)&&(SPVB==1)))
        {
            VMA=1;      // vermelho 1 aceso
            VDA=0;      // verde 1 apagado
        }
    }
}

```

```
VMB=0;      // vermelho 2 apagado
VDB=1;      // verde 2 aceso
}

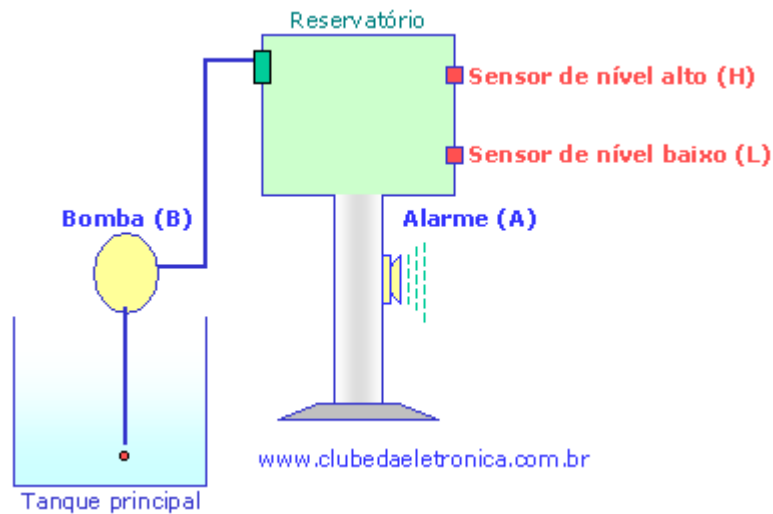
if(((SPVA==1)&&(SPVB==0))|((SPVA==1)&&(SPVB==1)))
{
  VMA=0;      // vermelho 1 apagado
  VDA=1;      // verde 1 aceso
  VMB=1;      // vermelho 2 aceso
  VDB=0;      // verde 2 apagado
}
}
```

**Praticando...**

15- Elabore o hardware para o controle de trafego, utilizando o Atmega8 e Leds.

## Projeto prático 02 – Controle de nível

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:



### Descrição de funcionamento:

- O reservatório deve estar sempre cheio, ou seja,  $H=1$ ;
- Se  $H=0$ , a bomba deverá ser acionada;
- Se a bomba não atender a demanda e o reservatório esvaziar, ou seja,  $L=0$ , um alarme deverá ser acionado.

### 1- Construa o hardware e elabore o software seguindo as etapas.

a. Defina as entradas e saídas

Entradas		Saídas	

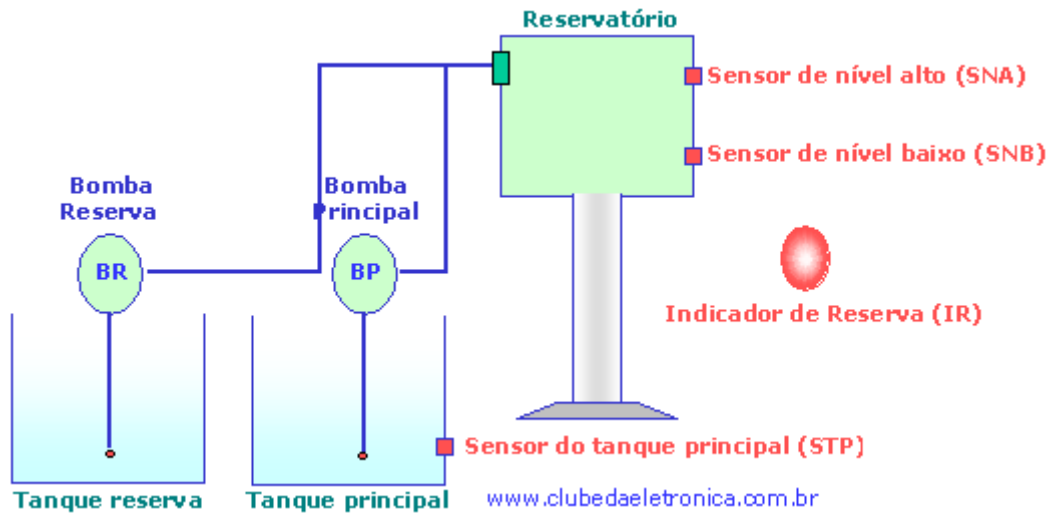
b. Complete a tabela verdade de maneira que atenda as exigências.

H	L	B	A	Expressões booleanas
				A=

- Extraia a expressão lógica
- Elabore o programa
- Construa o circuito utilizando o attiny2313

**Projeto prático 02 – controle de nível com tanque reserva**

Deseja-se controlar o nível de água de um reservatório, conforme ilustração:



Seu funcionamento deve ser o seguinte:

- O reservatório deve estar sempre cheio, ou seja, SNA=1;
- Se SNA=0, a bomba principal BP deverá ser acionada, mas somente se houver água no tanque principal, ou seja, STP =1, se STP =0, a bomba reserva deve ser acionada;
- Se a bomba reserva BR for acionada, um indicador de reserva (IR) deverá ser acionado.

Construa o hardware e elabore o software seguindo as etapas.

a. Defina as entradas e saídas

Entradas		Saídas	

b. Elabore a tabela verdade de maneira que atenda as exigências.

SNA	SNB	STP	BP	BS	IR	Expressões lógicas
						BP =  BS =  IR =

- c. Extraia a expressão lógica
- d. Elabore o programa
- e. Construa o circuito utilizando o attyni2313 (com driver de potência)

**Projeto prático 03 – Controle de abertura de Porta de elevador**

Deseja-se implementar um sistema lógico que controla a abertura da porta (**P**) de um elevador de três andares. O circuito apresenta 4 entradas, sendo:

**SM** = Sensor que indicará se o elevador esta em movimento (1) ou parado (0).  
**SA1, SA2 e SA3** são os sensores dos andares, se (1) esta no andar e (0) não.

Importante: não abrir a porta se o elevador estiver em movimento.

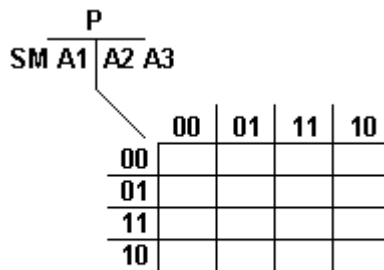
**Construa o hardware e elabore o software seguindo as etapas.**

a. Defina as entradas e saídas

Entradas		Saídas	

b. Complete a tabela verdade de maneira que atenda as exigências.

SM	A1	A2	A3	P
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



P =

- c. Complete o mapa para simplificação
- d. Extraia e expressão lógica
- f. Elabore o programa
- e. Construa o circuito utilizando o atmega16 (utilize LEDs para simular)

**Projeto prático 03 – Sistema de votação**

Deseja-se implementar um sistema lógico simplificado para um sistema de votação de uma empresa, que tem sua diretoria constituída pelos seguintes elementos: **Diretor**, **Vice-diretor**, **Secretário** e **Tesoureiro**.

Uma vez por mês esta diretoria se reúne para discutir sobre os mais diversos assuntos, sendo que as propostas são ou não **Aceitas**. Devido o número de elementos da diretoria ser par, o sistema adotado é o seguinte:

- Maioria → A proposta é aceita
- Minoria → A proposta é rejeitada
- Empate → Vence o voto dado pelo diretor

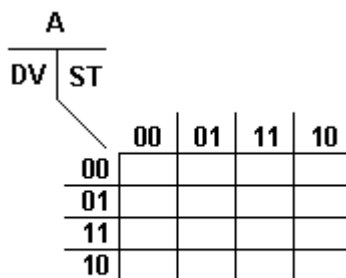
**Construa o hardware e elabore o software seguindo as etapas.**

a. Defina as entradas e saídas

Entradas		Saídas	

- b. Complete o mapa para simplificação
- c. Extraia e expressão lógica
- g. Elabore o programa
- d. Construa o circuito utilizando o attiny2313 (utilize LEDs para simular)

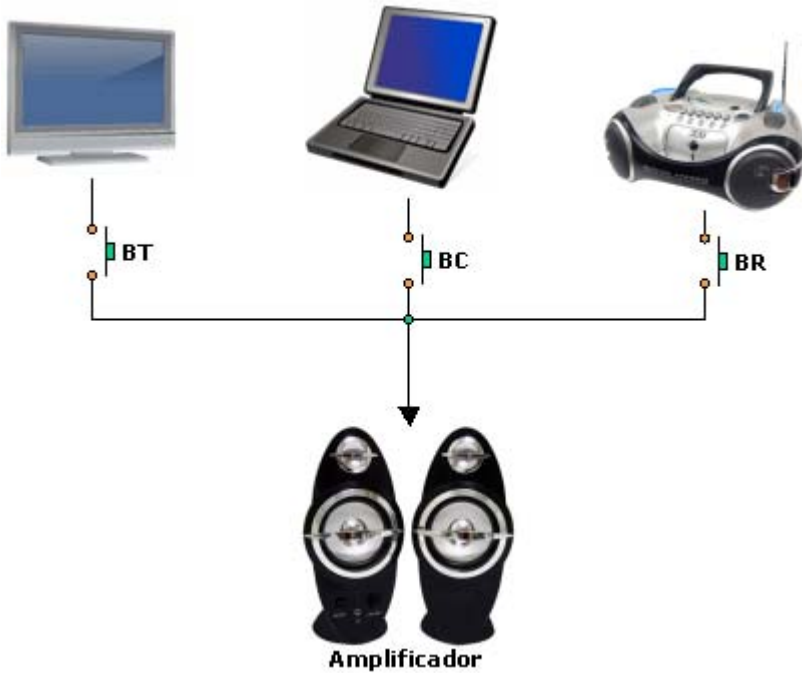
D	V	S	T	A
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



A =

**Projeto prático 04 – Controle de saída de áudio**

Deseja-se implementar um circuito lógico que controla a saída de áudio para um dispositivo amplificador. O circuito apresenta 3 entradas, sendo: **BT** (botão da TV), **BC** (botão do computador) e **BR** (Botão do rádio) todos ligados à uma única saída, o Amplificador **A**.



**Critérios de projeto:**

Só amplifica um por vez, estabelecendo as seguintes prioridades:

- 1- Televisão
- 2- Computador
- 3- Rádio.

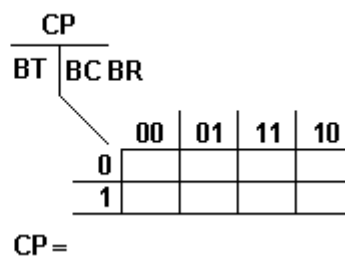
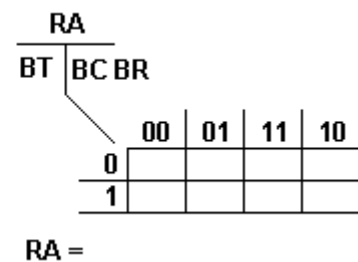
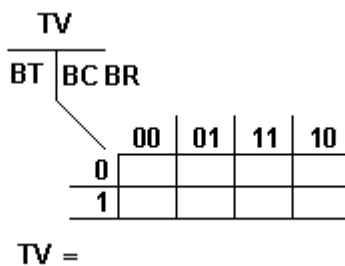
Construa o hardware e elabore o software seguindo as etapas.

a. Defina as entradas e saídas

Entradas		Saídas	

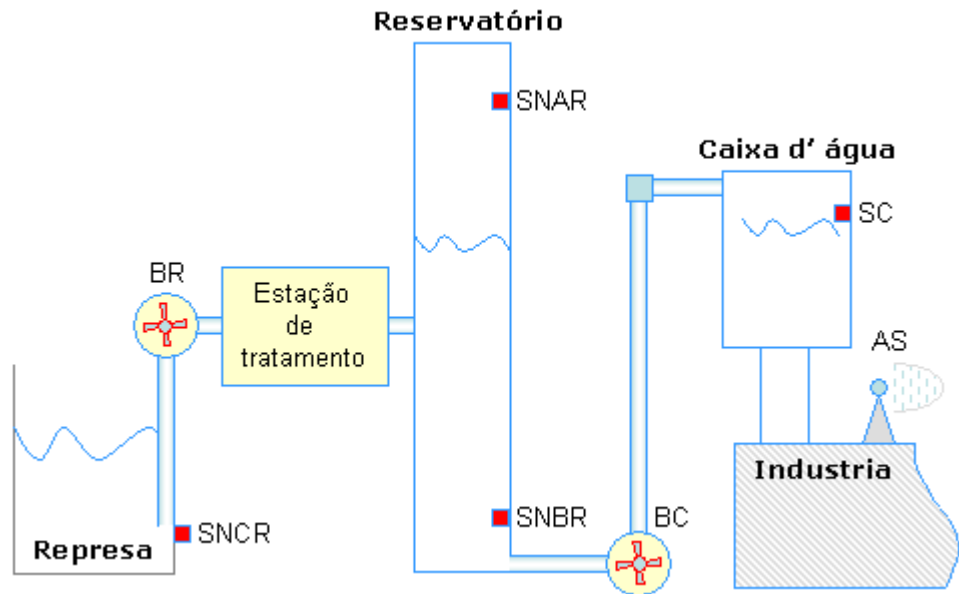
- b. Complete o mapa para simplificação
- c. Extraia e expressão lógica
- h. Elabore o programa
- d. Construa o circuito utilizando o attiny2313 (utilize LEDs para simular)

BT	BC	BR	TV	CP	RA
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			
1	1	1			



## Projeto prático 05 – Sistema de abastecimento

Uma indústria capta toda água que precisa de uma represa local. Esta água é bombeada para uma estação de tratamento e em seguida armazenada em um reservatório e esta por sua vez deve ser bombeada à uma caixa de água de menor porte, a fim de alimentar a indústria.



### Descrição do funcionamento

Sempre que o sensor de nível alto do reservatório (SNAR) estiver desacionado (0), a bomba do rio (BR) deve ser ligada (1) para encher o reservatório até o sensor de nível alto (SNAR) ser acionado (1).

A indústria esta em uma região de baixo índice pluviométrico e o rio, as vezes, fica baixo não sendo possível captar a água. Então o sensor de nível crítico do rio (SNCR) estiver desacionado (0), um alarme (AS) deverá ser ligado (1) para avisar o operador que a bomba do rio (BR) deve ficar desligada (0).

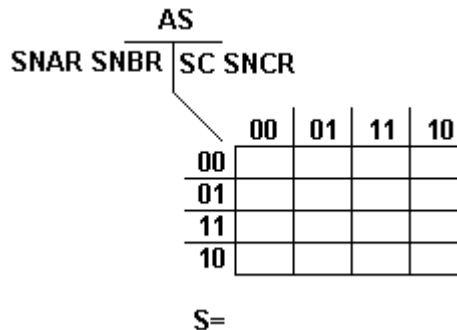
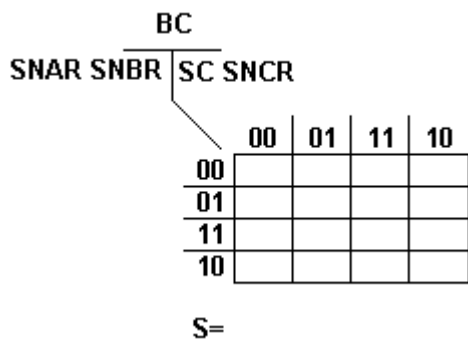
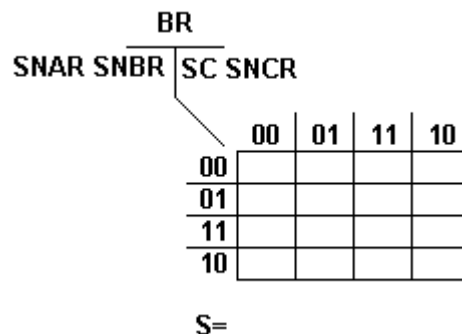
Ao mesmo tempo a caixa d'água da industria deve ficar com seu nível sobre o sensor da caixa (SC), ou seja,  $SC = 1$ .

Se o nível da caixa d'água ficar abaixo de SC, ou seja,  $SC = 0$  a bomba da caixa (BC) deve ser ligada (1), mas somente se  $SNBR = 1$ .

### Construa o hardware e elabore o software seguindo as etapas.

- Defina as entradas e saídas
- Complete o mapa para simplificação
- Extraia e expressão lógica
- Elabore o programa
- Construa o circuito utilizando o atmega16 (utilize driver para acionamento de contatores)

SNAR	SNBR	SC	SNCR	BR	BC	AS
0	0	0	0			
0	0	0	1			
0	0	1	0			
0	0	1	1			
0	1	0	0			
0	1	0	1			
0	1	1	0			
0	1	1	1			
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			



Continua ...