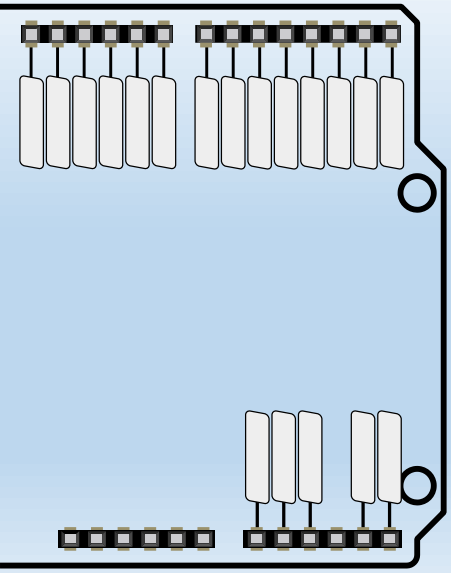
Saídas Digitais

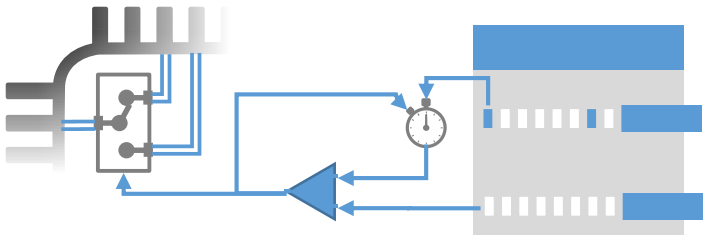


A revolução digital é muito mais significativa do que a invenção da escrita, ou mesmo da prensa de impressão.

Douglas Engelbart

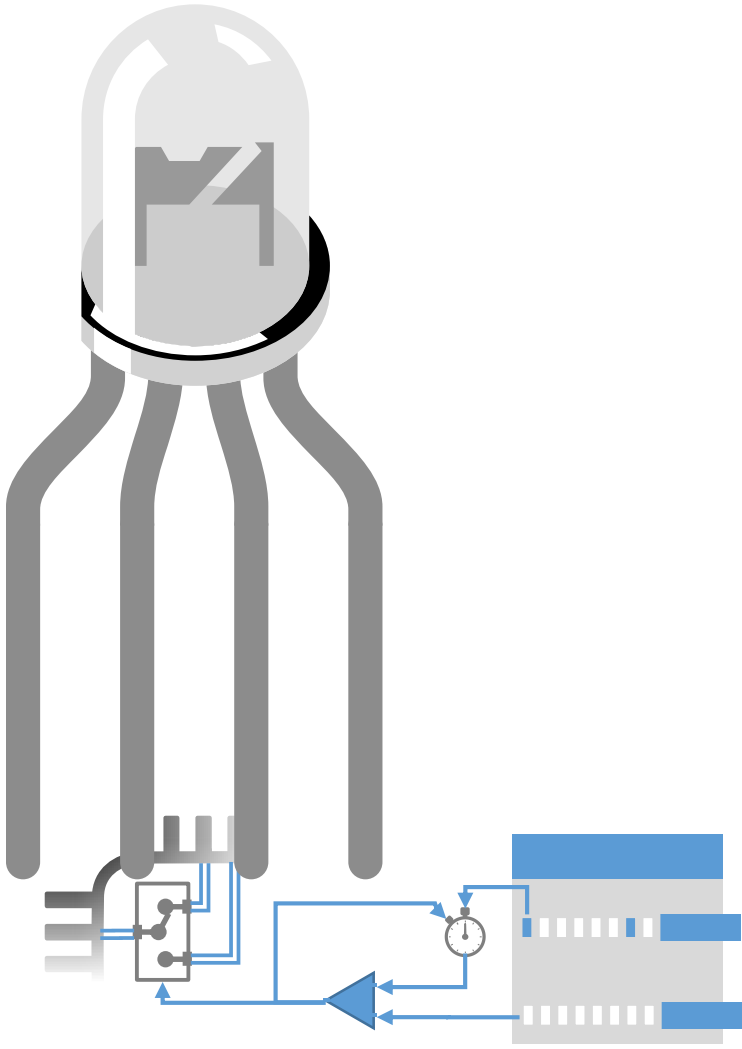
Saídas Digitais

- Diversos componentes eletrônicos possuem apenas dois tipos de estados: ligados ou desligados.
  - Para realizar o controle destes dispositivos são utilizadas saídas digitais.
- As saídas digitais apresentam dois estados distintos.
  - Na maioria das vezes estas saídas estão mapeadas em uma determinada região da memória.
  - Ao acessar essa memória e fazer com que o bit tenha o valor 1, a saída passa para um estado alto.
  - Fazendo o bit receber o valor 0, faz com que o estado da saída seja baixo.
  - Para isso podemos utilizar as rotinas bitClr() e BitSet().



Saídas Digitais

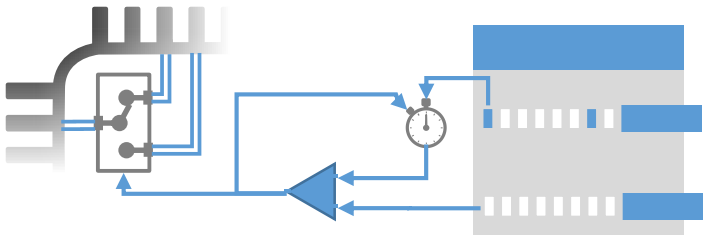
- Um led RGB é composto internamente de três leds:
  - Vermelho
  - Verde
  - Azul



Criação da biblioteca do Led RGB

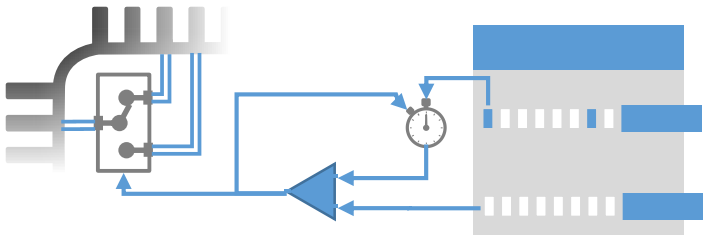
Criação da biblioteca do Led RGB

- O led RGB é um dispositivo simples de saída digital
- A biblioteca foi criada com o intuito de simplificar a geração das cores
  - Cada cor foi implementada como um define
  - Existem funções prontas para ligar/desligar o led ou configurar a cor a ser exibida



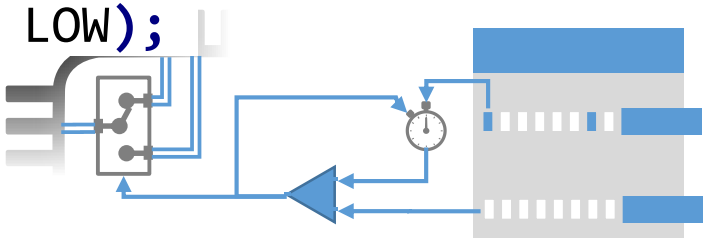
*Biblioteca **RGB***

```
#ifndef RGB
#define RGB
//todos desligados
#define OFF    0
//cores primárias
#define RED    1
#define GREEN  2
#define BLUE   4
//cores secundárias
#define YELLOW (RED+GREEN)
#define CYAN   (GREEN+BLUE)
#define PURPLE (RED+BLUE)
//todos acesos
#define WHITE  (RED+GREEN+BLUE)
       void rgbColor(int led);
       void turnOn(int led);
       void turnOff(int led);
       void rgbInit(void);
#endif
```



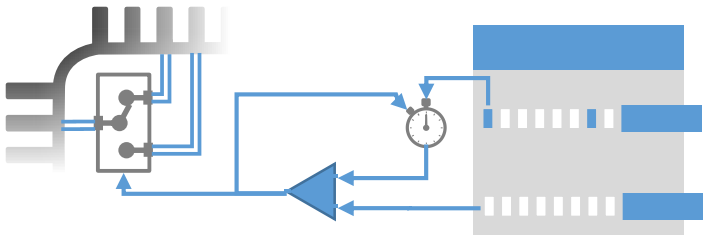
Biblioteca de acesso ao LED RGB

```
#include "io.h"

void rgbColor(int led) {
	if (led & 1) {
		digitalWrite(LED_R_PIN, HIGH);
	} else {
		digitalWrite(LED_R_PIN, LOW);
	}
	if (led & 2) {
		digitalWrite(LED_G_PIN, HIGH);
	} else {
		digitalWrite(LED_G_PIN, LOW);
	}
	if (led & 4) {
		digitalWrite(LED_B_PIN, HIGH);
	} else {
		digitalWrite(LED_B_PIN, LOW);
	}
}
```



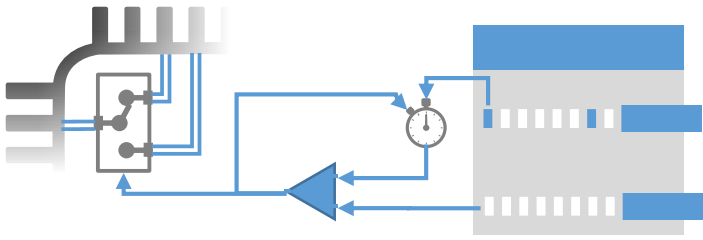
Biblioteca de acesso ao LED RGB

```
void turnOn(int led) {
      if (led & 1) { digitalWrite(LED_R_PIN, HIGH); }
      if (led & 2) { digitalWrite(LED_G_PIN, HIGH); }
      if (led & 4) { digitalWrite(LED_B_PIN, HIGH); }
}

void turnOff(int led) {
      if (led & 1) { digitalWrite(LED_R_PIN, LOW); }
      if (led & 2) { digitalWrite(LED_G_PIN, LOW); }
      if (led & 4) { digitalWrite(LED_B_PIN, LOW); }
}
```



Biblioteca de acesso ao LED RGB

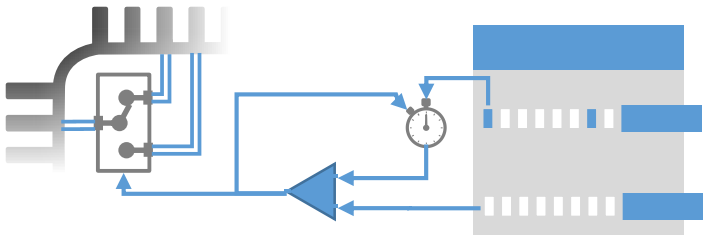
```
void rgbInit(void) {
      pinMode(LED_R_PIN, OUTPUT);
      pinMode(LED_G_PIN, OUTPUT);
      pinMode(LED_B_PIN, OUTPUT);
}
```



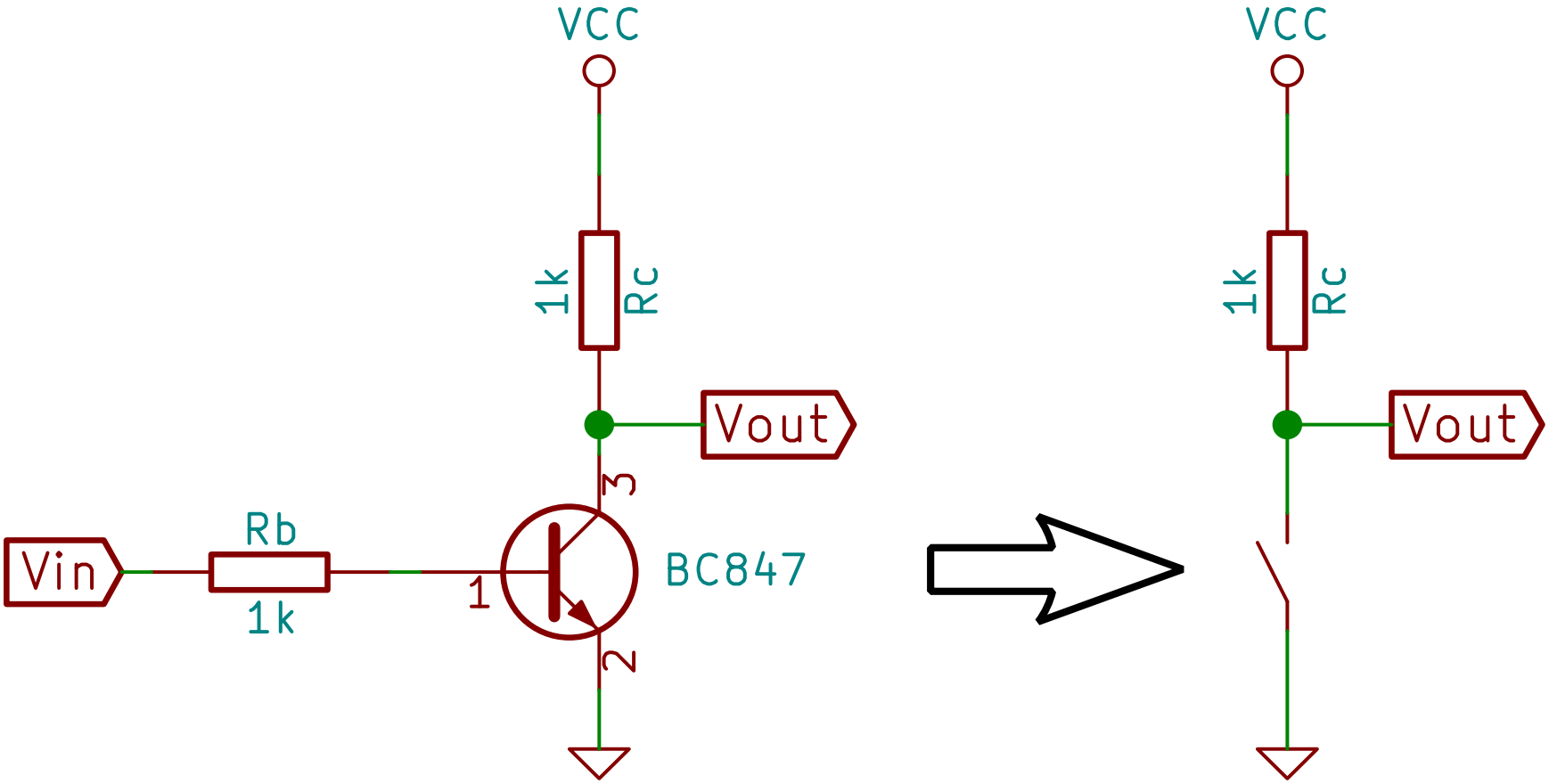
Outras saídas digitais

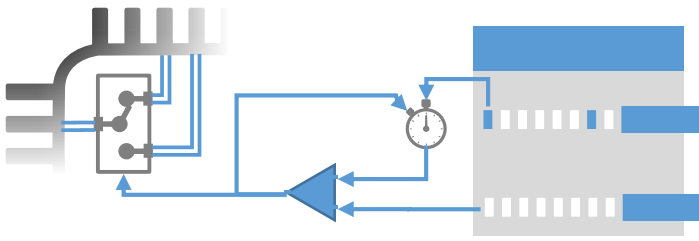
Outras saídas digitais

- Transistores
  - Pode funcionar como um amplificador ou chave.
  - No modo amplificador ele possui a capacidade de ampliar o nível de tensão.
  - Como chave ele permite ligar cargas em sua saída.
  - Apenas tensão contínua
- Relés
  - Similar ao transistor operando como chave, mas permite DC e AC
- Relés de estado sólido
  - Circuitos eletrônicos que funcionam como um relé.
  - Dependendo do modelo pode permitir AC

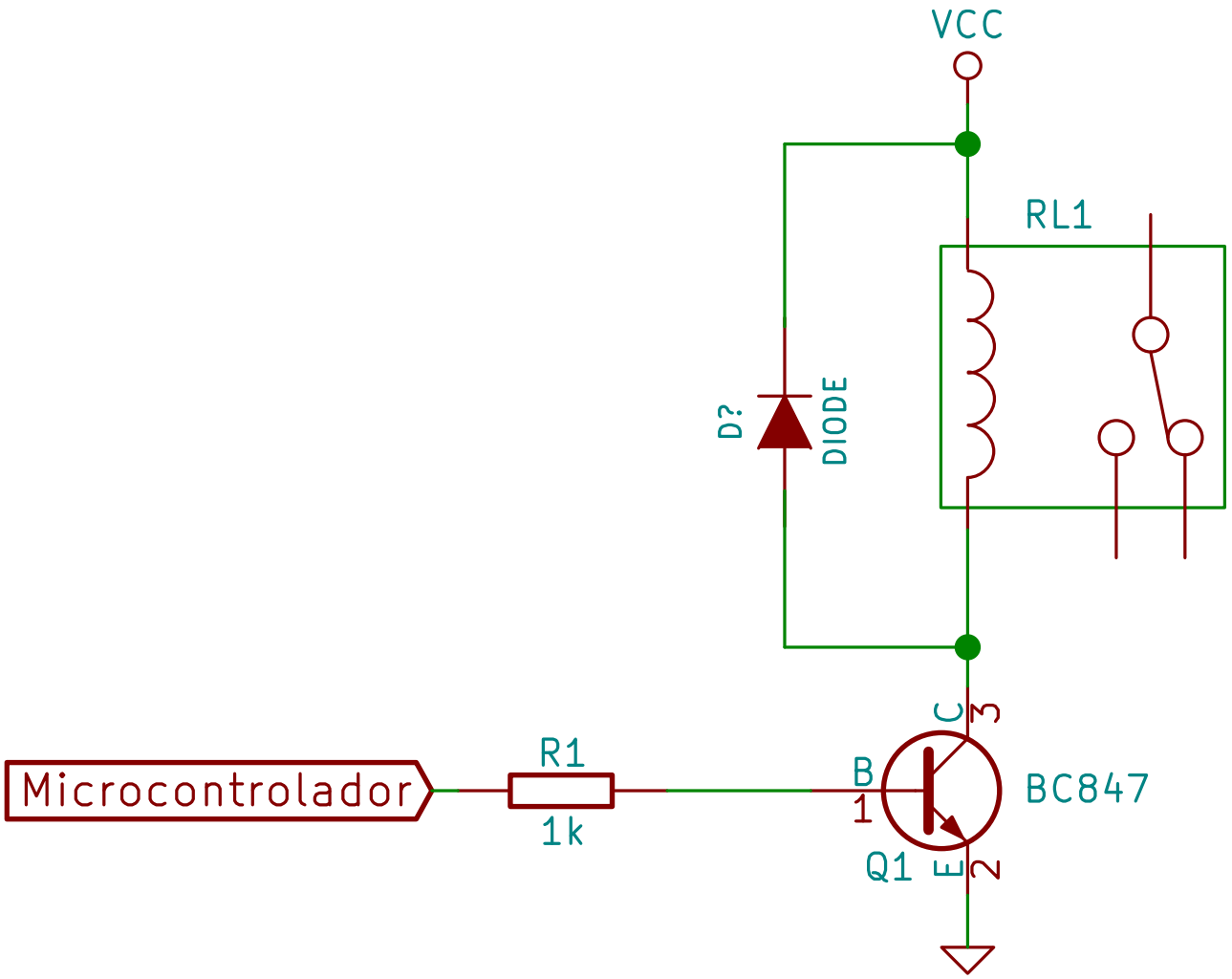


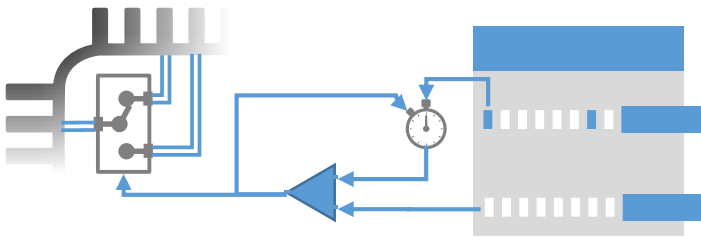
Transistor operando como chave





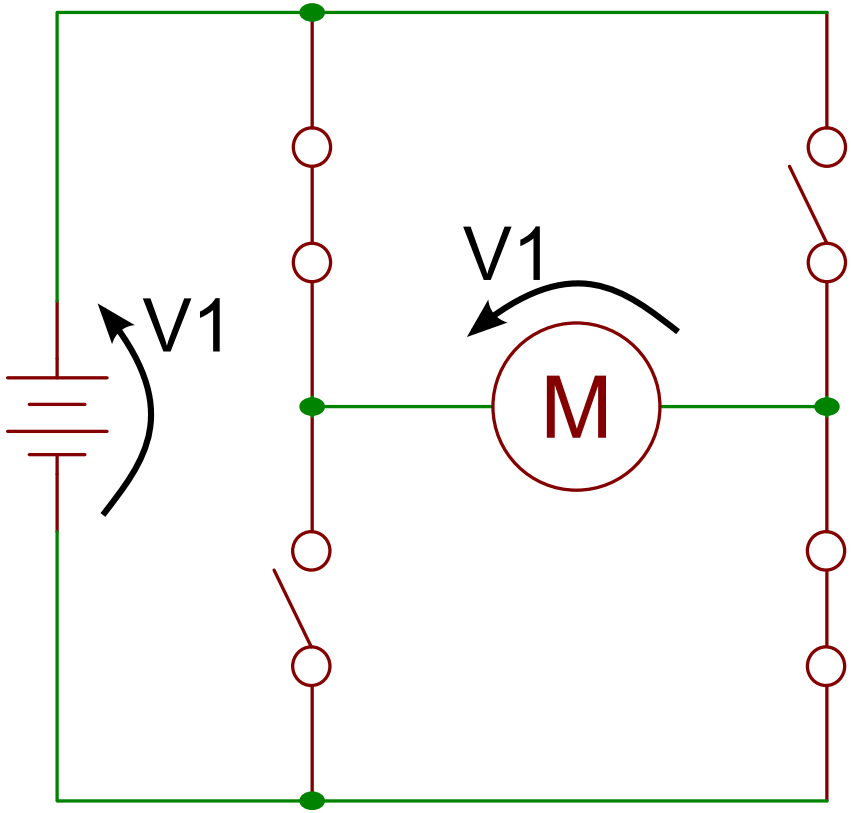
Acionamento por relé

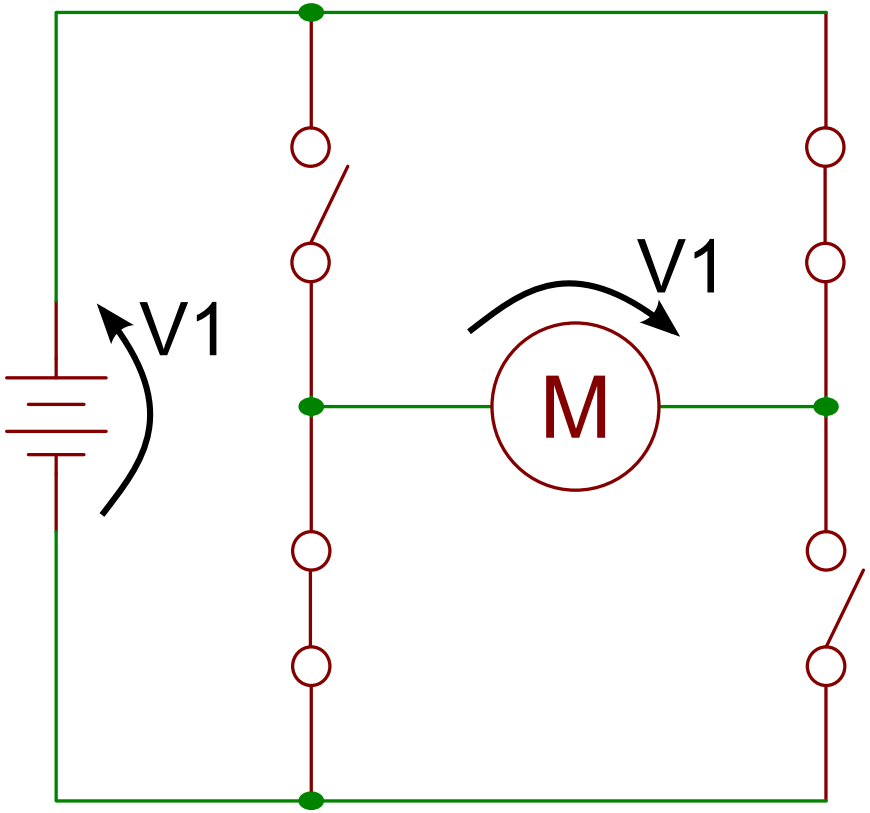


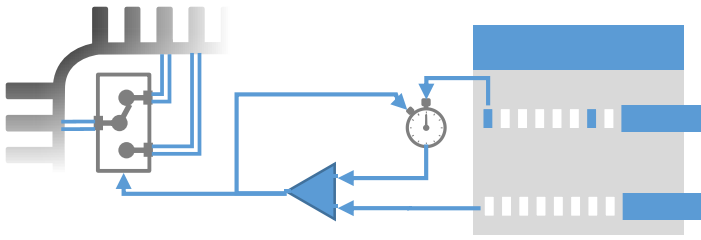


Acionamento de uma ponte H

Exemplo de uso ponte H



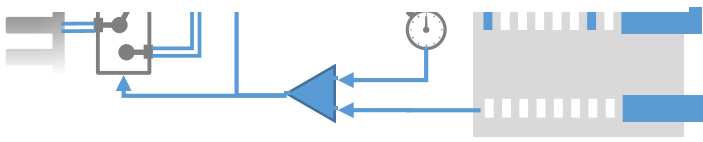




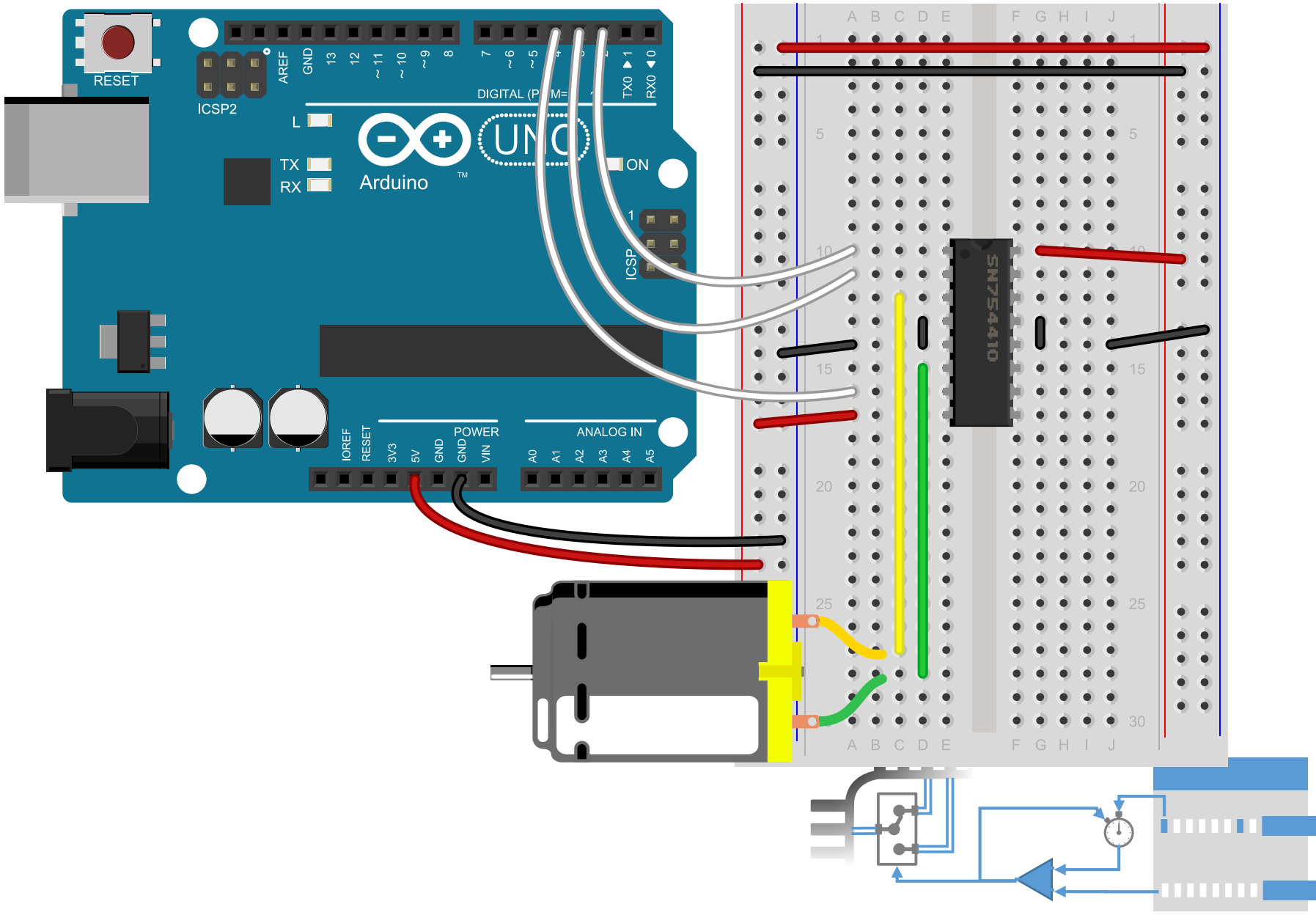
Acionamento (terminais ligam as chaves)

```
#define sw1A 3
#define sw1B 4
#define sw2A 5
#define sw2B 6

void initMotorControl(void){
//configura 4 terminais como saída
    pinMode(sw1A, OUTPUT);
    pinMode(sw1B, OUTPUT);
    pinMode(sw2A, OUTPUT);
    pinMode(sw2B, OUTPUT);
}

void motorOff(void){
    digitalWrite(sw1A, LOW);
    digitalWrite(sw1B, LOW);
    digitalWrite(sw2A, LOW);
    digitalWrite(sw2B, LOW);
}
```

```
void motorOnLeft(void){
     //sempre desliga primeiro
     digitalWrite(sw2A, LOW);
     digitalWrite(sw2B, LOW);
     digitalWrite(sw1A, HIGH);
     digitalWrite(sw1B, HIGH);
}

void motorOnRight(void){
     //sempre desliga primeiro
     digitalWrite(sw1A, LOW);
     digitalWrite(sw1B, LOW);
     digitalWrite(sw2A, HIGH);
     digitalWrite(sw2B, HIGH);
}
```



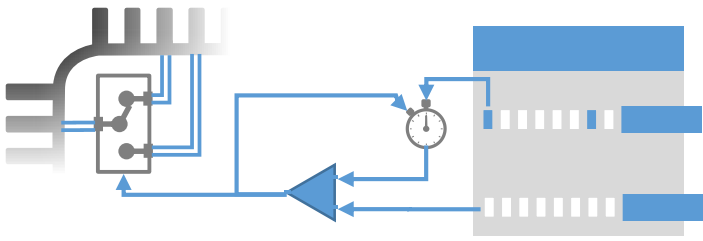
Ponte H com circuito dedicado



Acionamento com circuito dedicado

```
#define motor1Pin  3
#define motor2Pin  4
#define enablePin  2
void initMotorControl(void){
        pinMode(motor1Pin, OUTPUT);
        pinMode(motor2Pin, OUTPUT);
        pinMode(enablePin, OUTPUT);
}
void motorOff(void){
        digitalWrite(enablePin, LOW);
}
void motorOnLeft(void){
        digitalWrite(enablePin, HIGH);
        digitalWrite(motor1Pin, LOW);
        digitalWrite(motor2Pin, HIGH);
}
void motorOnRight(void){
        digitalWrite(enablePin, HIGH);
        digitalWrite(motor1Pin, HIGH);
        digitalWrite(motor2Pin, LOW);
}
```

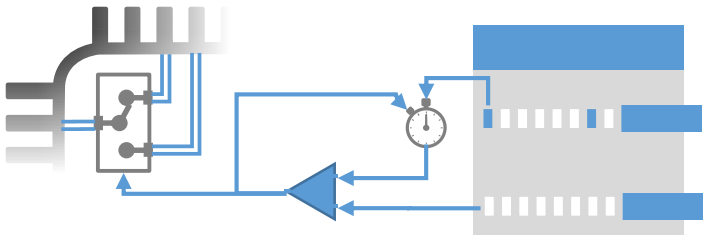
- Acionamento mais simples
- Sem problema de curto circuito



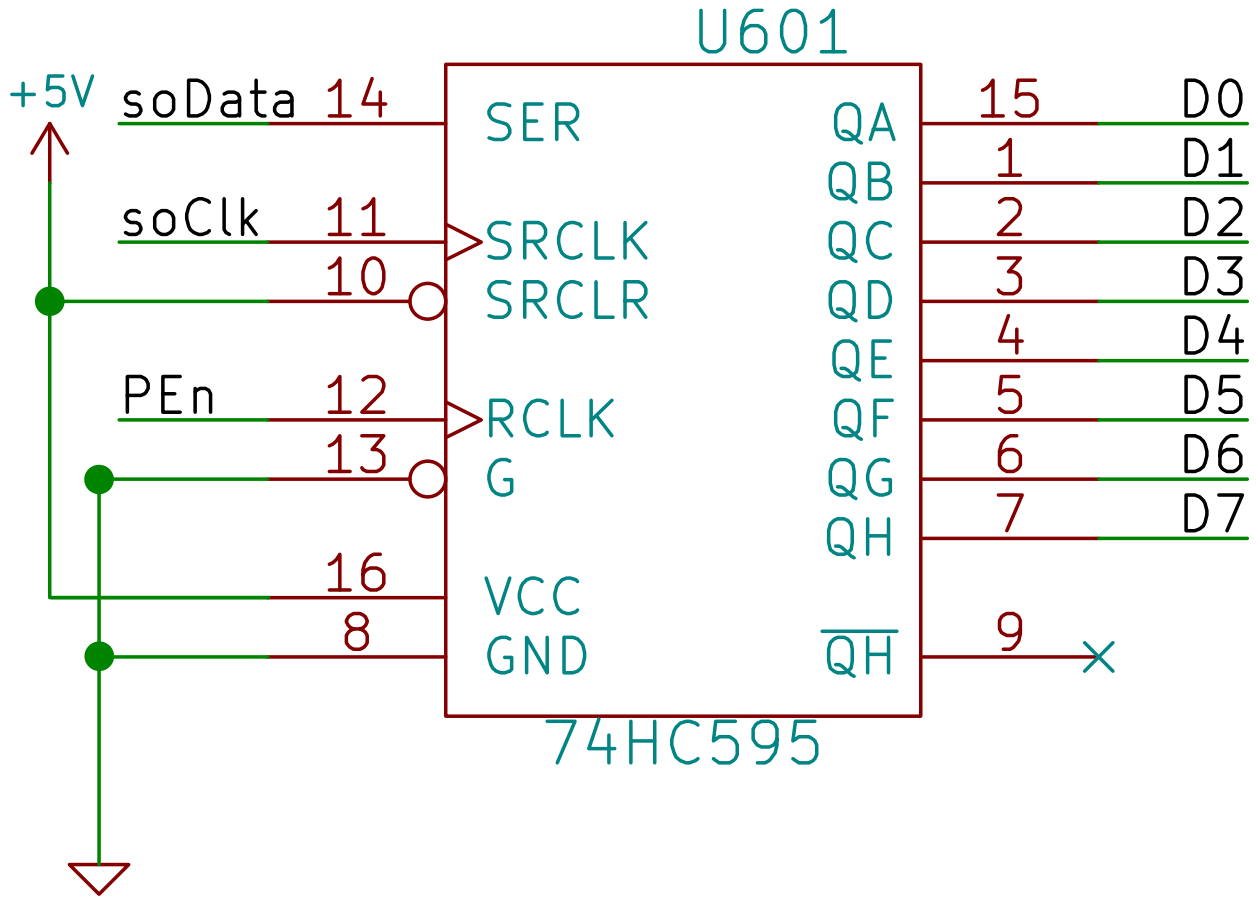
Expansão de saídas digitais

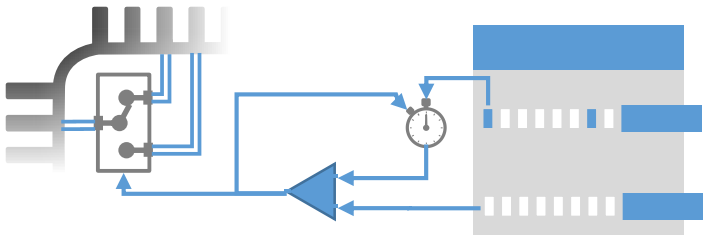
Expansão de saídas

- O custo de um microcontrolador é dependente da quantidade de saídas disponíveis.
- Se o micro não possui saídas suficientes podemos utilizar circuitos para expansão de saídas
  - conversores de serial para paralelo
  - expansores de IO
  - multiplexação temporal dos terminais
  - multiplexação em frequências diferentes.
- Toda abordagem insere algum problema
  - custo no sistema;
  - atraso na resposta;
  - aumento da complexidade.



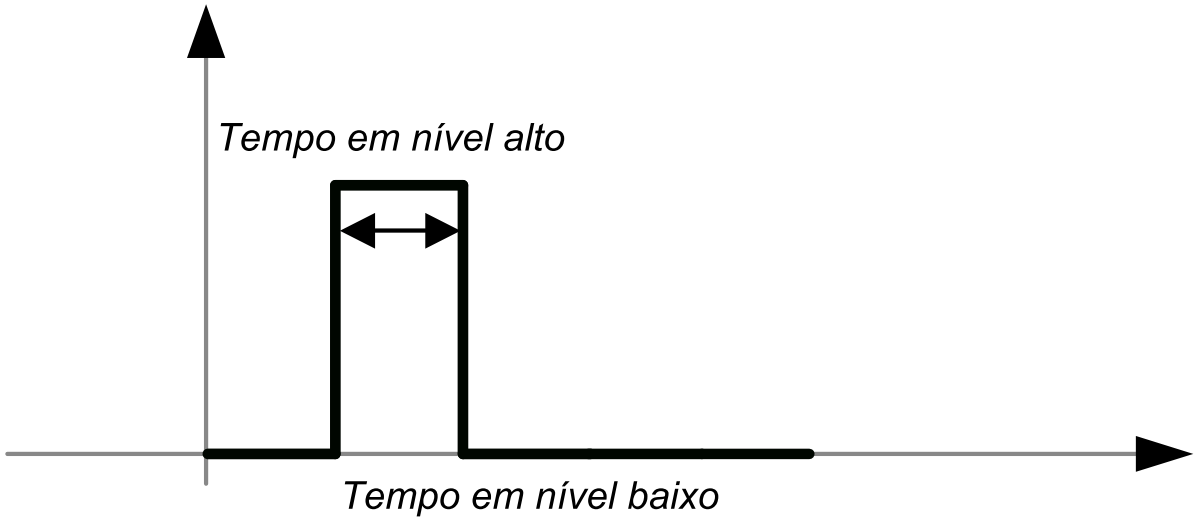
Expansão de saídas

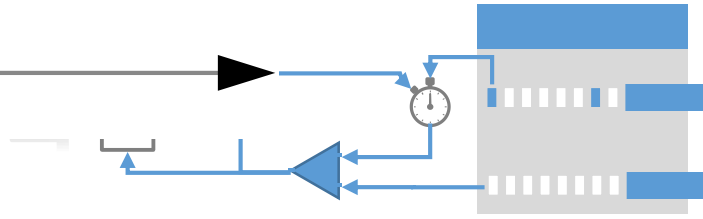




Funcionamento do 595

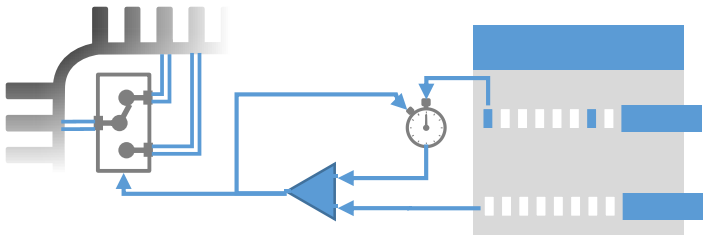
- Cada bit é enviado de modo serial
  - Os bit são enviado no terminal soData
  - Para indicar o “fim” da transmissão de cada bit deve ser enviado um pulso no terminal soClk
- Os oito últimos bits são apresentados na saída paralela quando houver um pulso no pino Pen
- Um pulso é o ato de ligar e desligar um terminal gerando uma pequena alteração no sinal





Gerando um pulso nos terminais

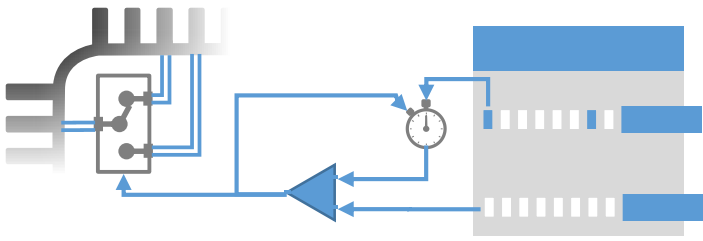
```
void PulseEnClock(){
      digitalWrite(SO_EN_PIN, HIGH);
      digitalWrite(SO_EN_PIN, LOW);
}
void PulseClockData(){
      digitalWrite(SO_CLK_PIN, HIGH);
      digitalWrite(SO_CLK_PIN, LOW);
}
```



Conversor serial paralelo 74hc595

- Para simplificar o envio é possível criar uma função que recebe uma variável de 8 bits e envia cada bit serialmente.

```
void soWrite(int value) {
      int i;
      digitalWrite(SO_CLK_PIN, LOW);
      for (i = 0; i < 8; i++) {
            digitalWrite(SO_DATA_PIN, value & 0x80);
            PulseClockData();
            value <<= 1;
      }
      PulseEnClock();
}
```



Biblioteca **so**

- A biblioteca só possui duas funções
  - Inicializar os terminais
  - Escrever os oito bits através do conversor
- O conversor será utilizado como uma porta extra para acionar o LCD, display de 7 segmentos e o teclado.

```
#ifndef SO_H_
#define SO_H_

        void soInit (void);
        void soWrite(int value);

#endif
```