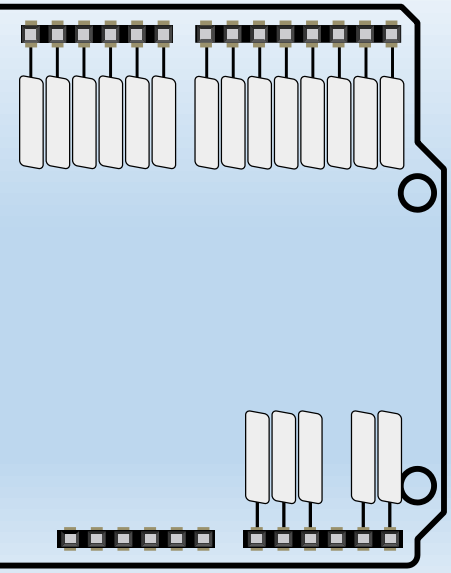
Programação de periféricos

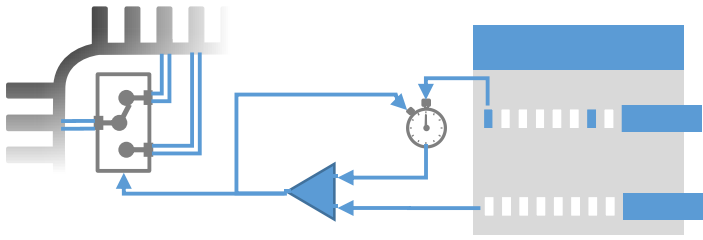


Em teoria, não há diferença entre teoria e prática; mas na prática sim.

Jan van de Snepscheut

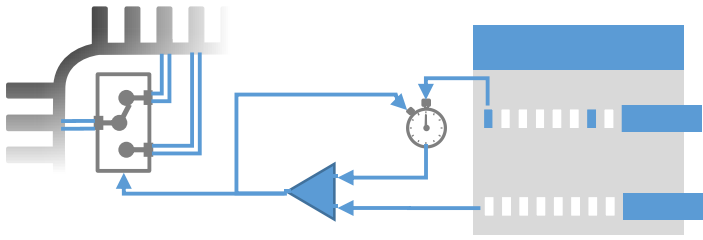
Periféricos

- Os periféricos podem ser
  - internos ao microcontrolador, embutidos no mesmo chip,
  - circuitos externos com funções próprias.
- Os periféricos internos podem executar atividades dedicadas de modo independente do processador
  - converter um sinal analógico num valor digital,
  - realizar a contagem de um tempo,
  - preparar uma mensagem para ser enviada por um sistema serial.



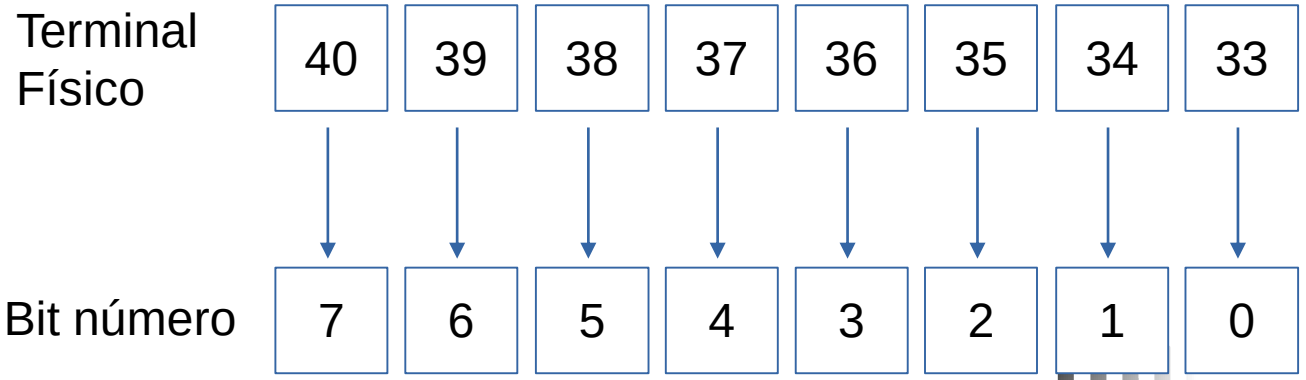
Periféricos

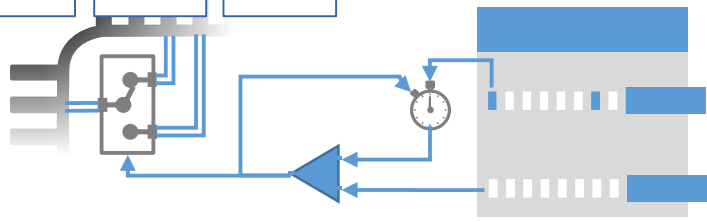
- Os periféricos de interface fazem com que os sinais gerados pelo processador cheguem aos terminais físicos do microcontrolador.
- Os periféricos externos adicionam funcionalidades
  - como interfaces de entrada e saída para o usuário, como teclado ou um display de LCD.
  - complementam as funções internas do chip, provendo recursos que o processador não apresentam



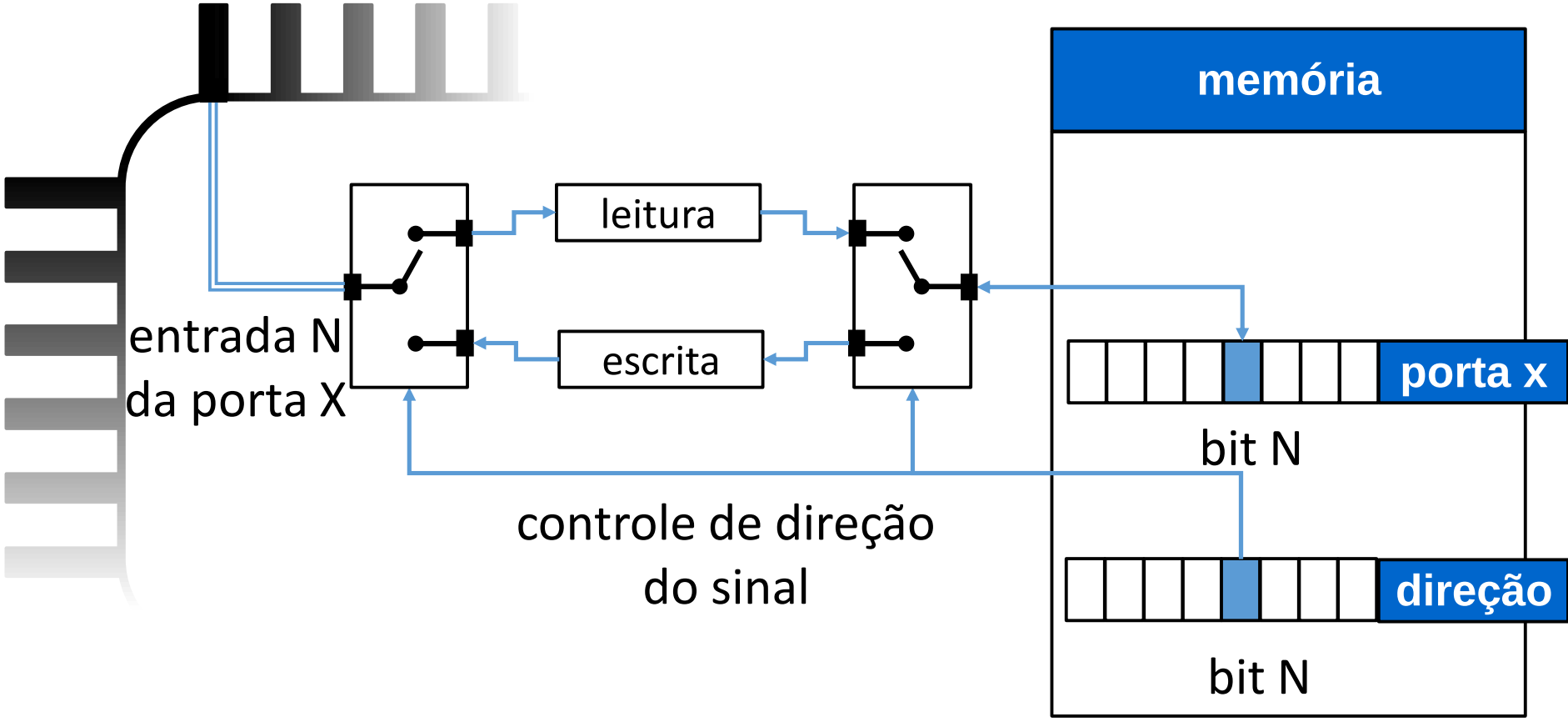
Periféricos de entrada/saída

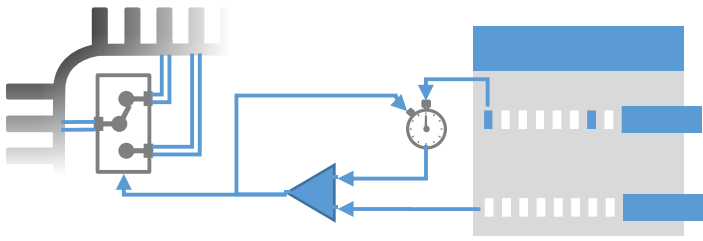
- As portas são registros que repassam a informação da memória para os terminais físicos.
  - Cada um dos bits representa um terminal.
  - Ligar o bit -> terminal com tensão alta (3v3 ou 5v).
  - Desligar o bit -> o terminal com tensão zero.
- Por exemplo
  - os oito bits (0 à 7) estão diretamente conectados aos terminais físicos de número 33 à 40.



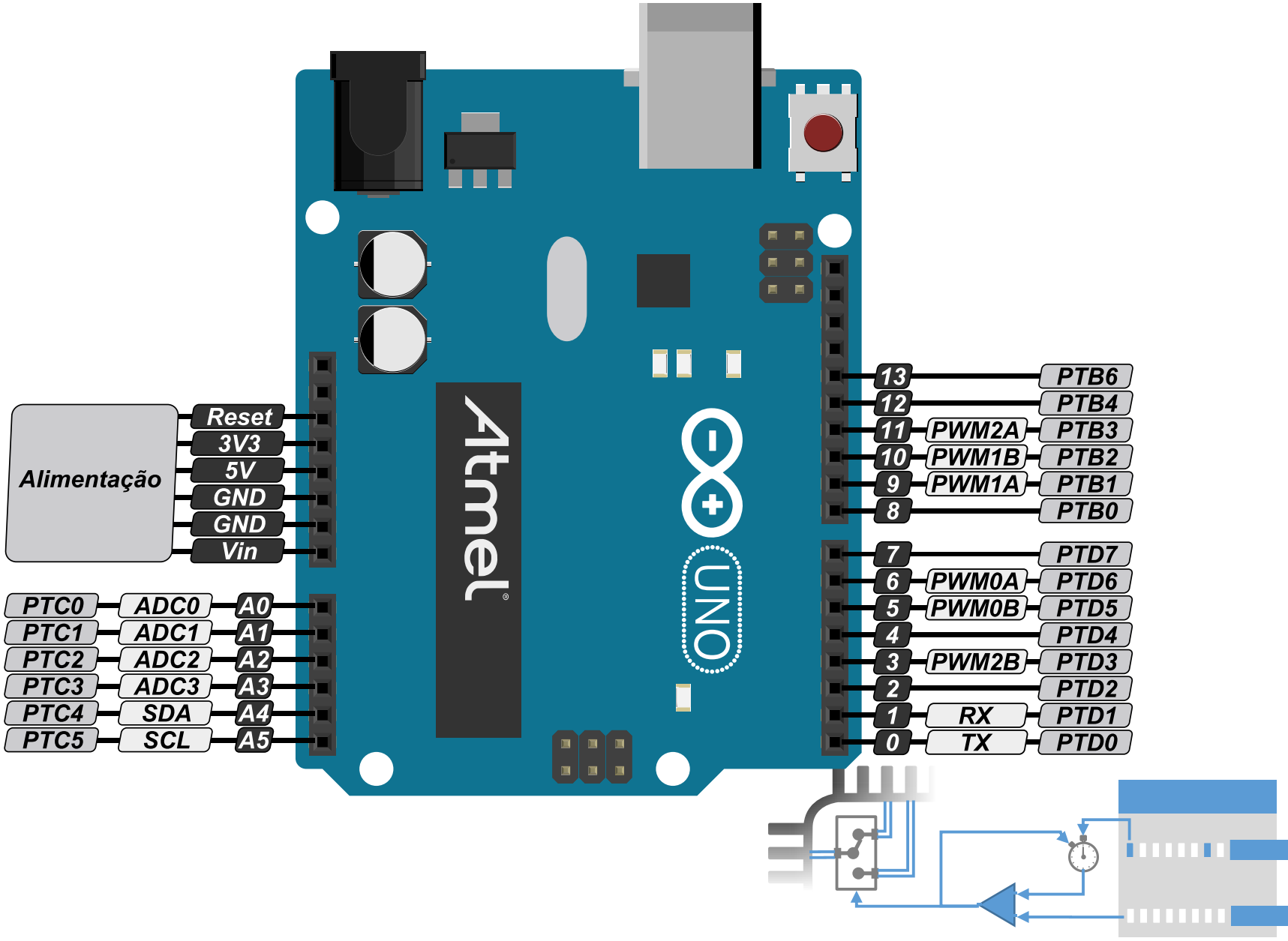


Periféricos de entrada/saída

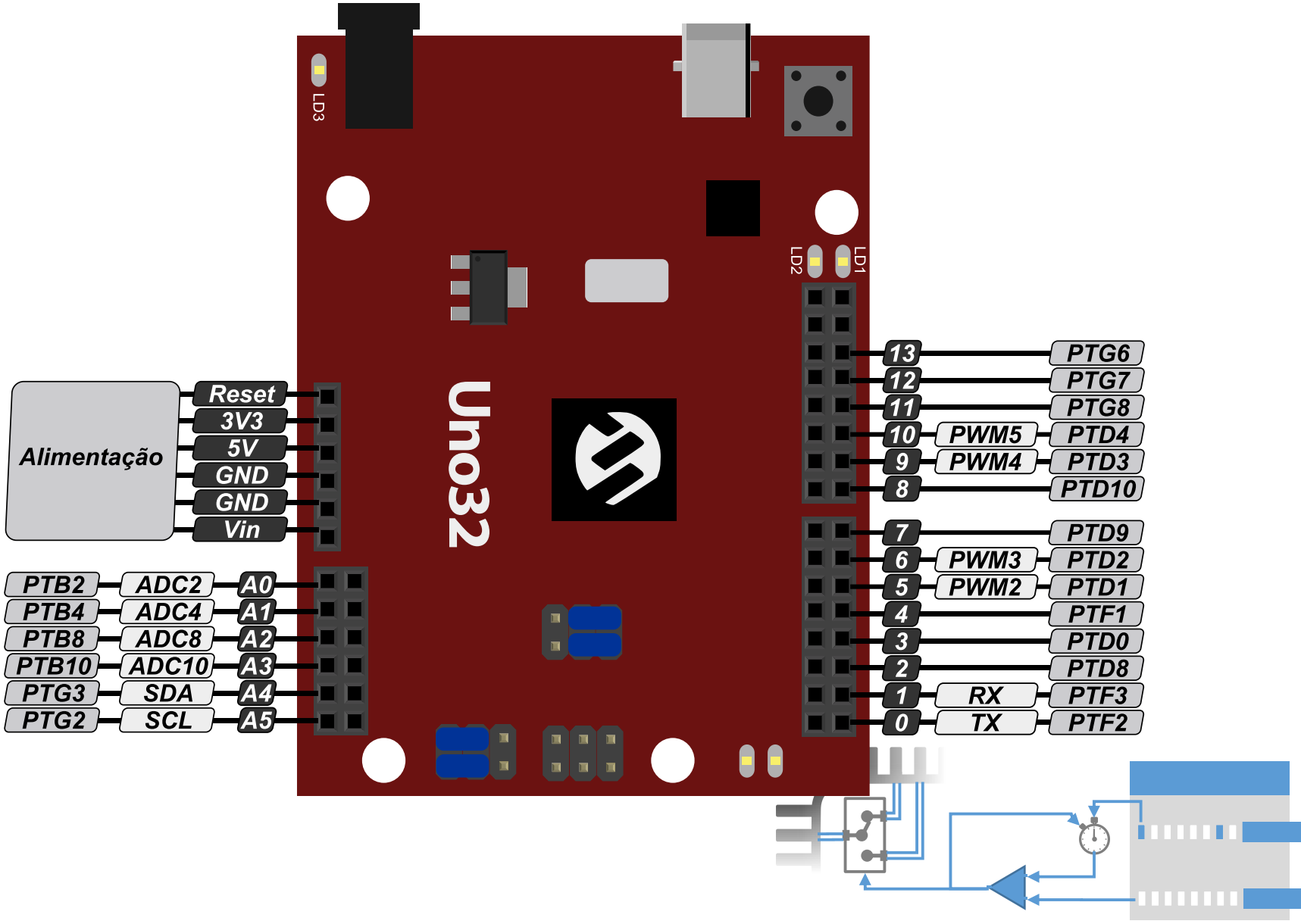




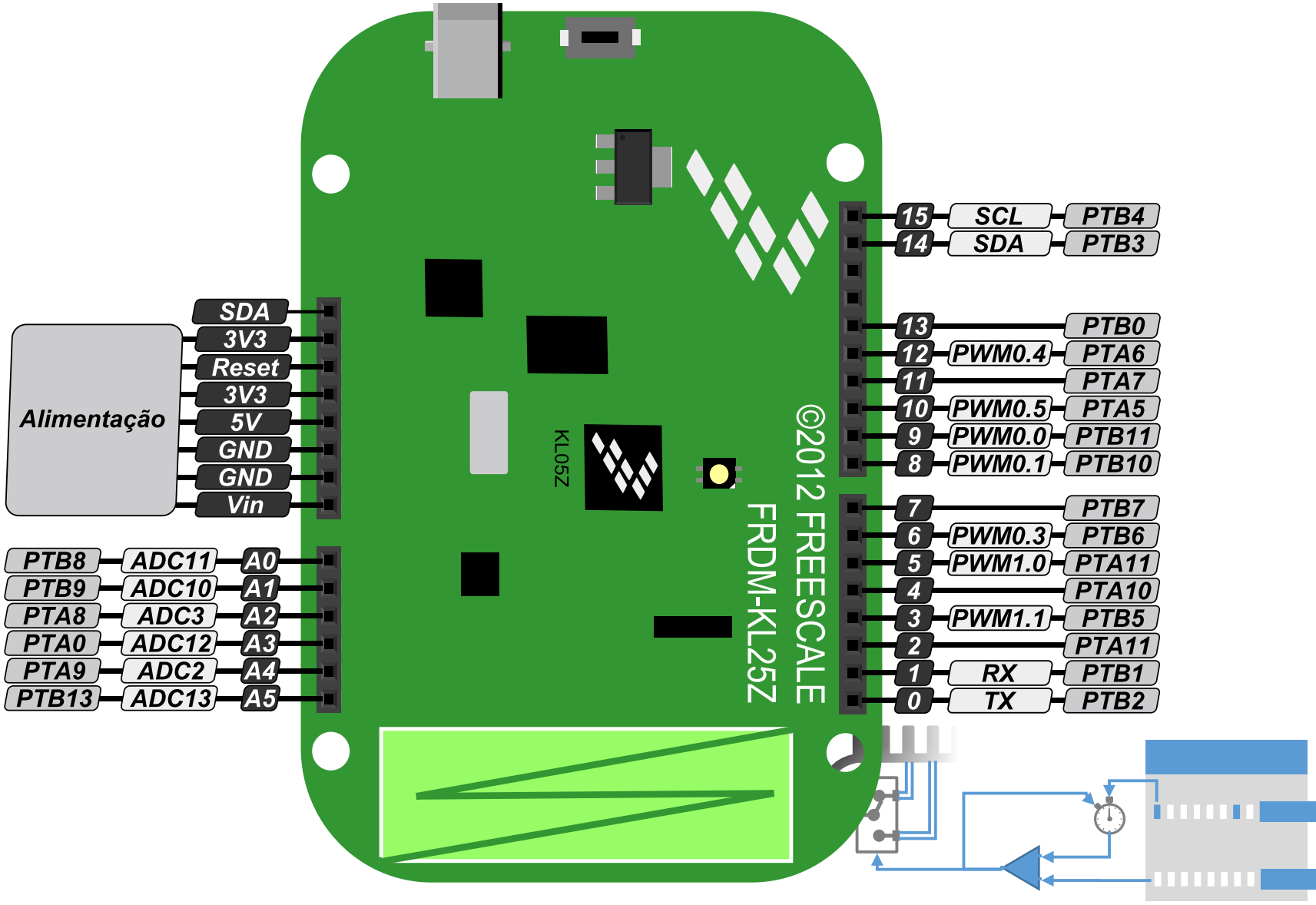
Conexões dos terminais



Conexões dos terminais



Conexões dos terminais

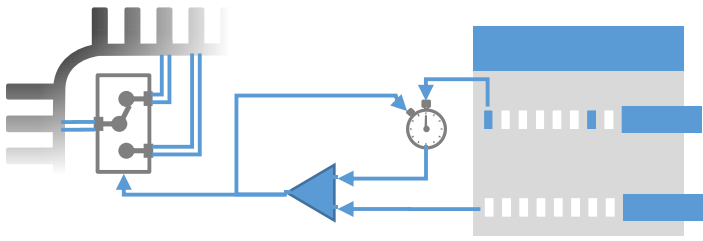


Registros de acesso aos terminais

Registro	Endereço	Microcontrolador
DDRB	0x04	ATmega328
PORTB	0x05	ATmega328
DDRC	0x07	ATmega328
PORTC	0x08	ATmega328
DDRD	0x0A	ATmega328
PORTD	0x0B	ATmega328
TRISB	0xBF886040	PIC32MX320
PORTB	0xBF886050	PIC32MX320
TRISD	0xBF8860C0	PIC32MX320
PORTD	0xBF8860D0	PIC32MX320
TRISF	0xBF886140	PIC32MX320
PORTF	0xBF886150	PIC32MX320
TRISG	0xBF886180	PIC32MX320
PORTG	0xBF886190	PIC32MX320
PORTA_PDOR	0x4004F000	KL05z32
PORTA_PDIR	0x4004F010	KL05z32
PORTA_PDDR	0x4004F014	KL05z32
PORTB_PDOR	0x4004F040	KL05z32
PORTB_PDIR	0x4004F050	KL05z32
PORTB_PDDR	0x4004F054	KL05z32

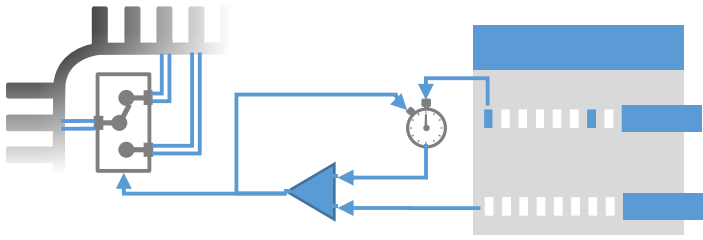
Registros de acesso aos terminais

```
//inicio do programa
void main(void) {
//definimos como:
//(unsigned char) pois os 8 bits representam valores
//(volatile) as variáveis podem mudar a qualquer momento
        volatile unsigned char *PORTD = 0xF83;
        volatile unsigned char *TRISD = 0xF95;
        //configurando todos os pinos como saídas
        // 0 = saída (Output)
        // 1 = entrada (Input)
        *TRISD = 0b00000000;
        //liga apenas os quatro últimos leds
        *PORTD = 0b11110000;
        //mantém o sistema ligado indefinidamente
        for(;;);
}
```

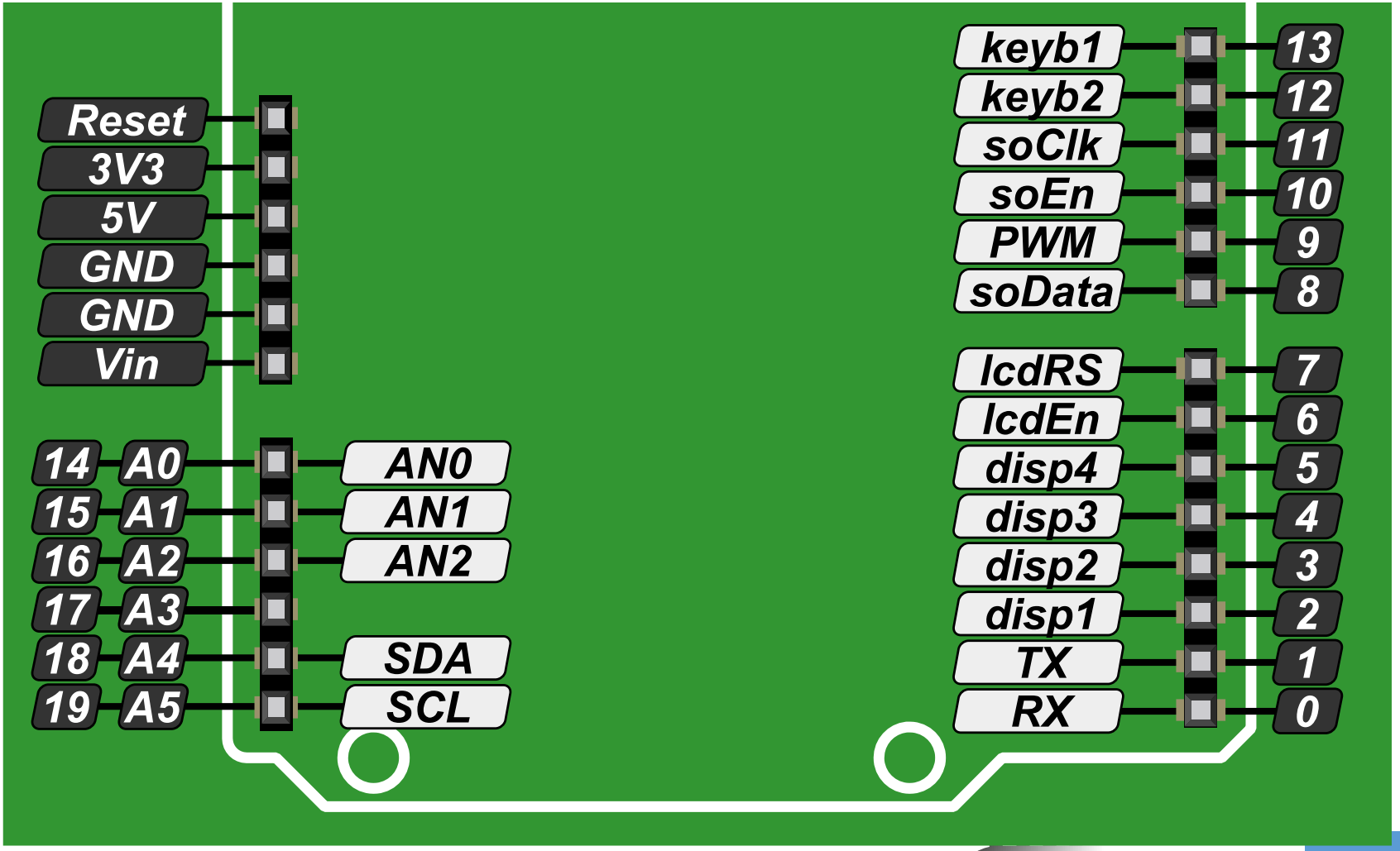


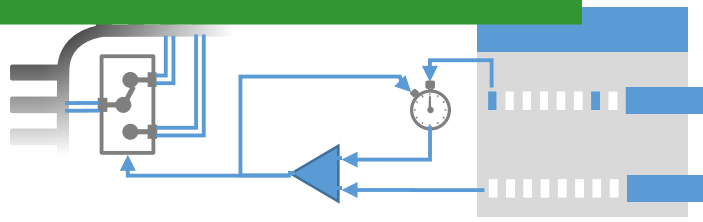
Registros de acesso aos terminais

```
//defines para portas de entrada e saída
#define PORTD (*(volatile unsigned char*)0xF83)
#define TRISD (*(volatile unsigned char*)0xF95)
//inicio do programa
void main(void) {
	//configurando todos os pinos como saídas
	TRISD = 0b00000000;
	//liga apenas os quatro últimos leds
	PORTD = 0b11110000;
	//mantém o sistema ligado indefinidamente
	for(;;);
}
```



Conexão com dispositivos externos





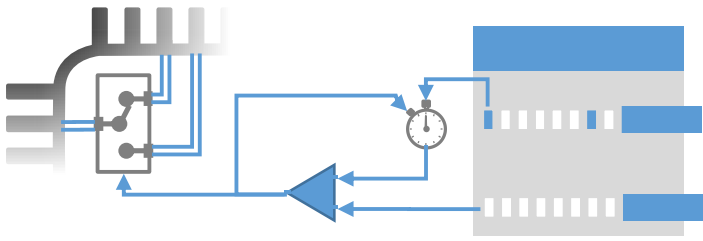
Conexão com dispositivos externos

Terminal	Função	Direção	Freedom	Chipkit	Arduino
D0	RX	Entrada	PTB2	PTF2	PTD0
D1	TX	Saída	PTB1	PTF3	PTD1
D2	disp1	Saída	PTA11	PTD8	PTD2
D3	disp2	Saída	PTB5	PTD0	PTD3
D4	disp3	Saída	PTA10	PTF1	PTD4
D5	disp4	Saída	PTA12	PTD1	PTD5
D6	lcdEn	Saída	PTB6	PTD2	PTD6
D7	lcdRS	Saída	PTB7	PTD9	PTD7
D8	soData	Saída	PTB10	PTD10	PTB0
D9	PWM	Saída	PWM0.0	PWM4	PTB1
D10	soEn	Saída	PTA5	PTD4	PTB2
D11	soClk	Saída	PTA7	PTG8	PTB3
D12	keyb1	Entrada	PTA6	PTG7	PTB4
D13	keyb2	Entrada	PTB0	PTG6	PTB5
A0/D14	AN0	Entrada	ADC11	ADC2	ADC0
A1/D15	AN1	Entrada	ADC10	ADC4	ADC1
A2/D16	AN2	Entrada	ADC3	ADC8	ADC2
A3/D17	---	----	----	---	---
A4/D18	SDA	Entrada/Saída	PTA9	PTG3	PTC4
A5/D19	SCL	Saída	PTB13	PTG2	PTC5

Configuração dos periféricos

- Configuração como entrada/saída
  - pinMode(pin,mode);
- Implementação do Arduino:

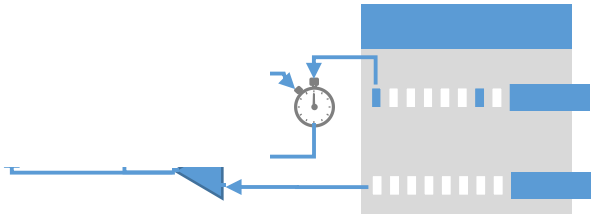
```
void pinMode(uint8_t pin, uint8_t mode){
        uint8_t bit = digitalPinToBitMask(pin);
        uint8_t port = digitalPinToPort(pin);
        volatile uint8_t *reg, *out;

        reg = portModeRegister(port);
        out = portOutputRegister(port);

        if (mode == INPUT) {
                *reg &= ~bit;
                *out &= ~bit;
        } else if (mode == OUTPUT){
                *reg |= bit;
        }
}
```



Configuração dos periféricos

- Implementação para Freescale (trecho da biblioteca **io.c**)

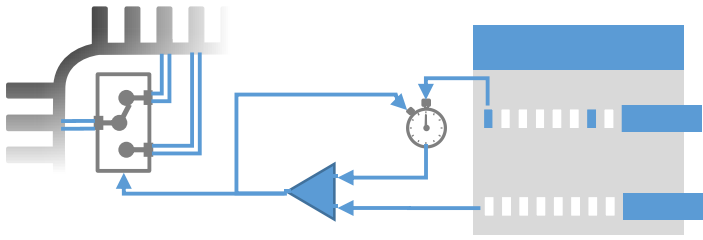
```
void pinMode(int pin, int type) {
        if (type == OUTPUT) {
                switch (pin) {
                case 0: PORTB_PCR(2)  = PRC_V;
                        bitSet(PORTB_PDDR, 2);
                        break;
                 //...
                default: break;
                }
        }
        if (type == INPUT) {
                switch (pin) {
                case 12: PORTA_PCR(6)  = PRC_V;
                        bitClr(PORTA_PDDR, 6);
                        break;
                //...
                default:  break;
                }
        }
}
```



Criação da biblioteca io

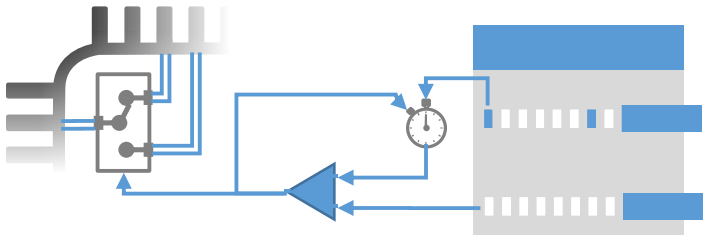
Criação da biblioteca io

- A biblioteca io é responsável por implementar as funções de acesso aos terminais físicos do microcontrolador
- Isso é feito através de 2 funções
  - digitalWrite() : responsável por ligar/desligar um terminal
  - digitalRead(): responsável por ler o estado de um terminal
- Além das funções para manipulação ela também implementa duas funções de configuração
  - pinMode(): define se o terminal será tratado como entrada ou saída
  - systemInit(): inicializa os periféricos básicos do sistema
    - Deve ser a primeira função a ser chamada no main
- Por fim as macros de manipulação de bits também são implementadas na io



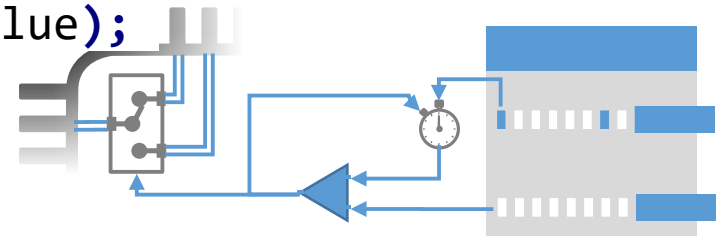
Criação da biblioteca io

- Através desta biblioteca, o acionamento de entradas/saídas na Freedom fica idêntico ao da plataforma Wiring (Arduino/Chipkit)
  - Isto simplifica a geração das demais bibliotecas, que ficarão praticamente idênticas.
- Essa biblioteca também cria um conjunto de defines explicando qual é a função de cada um dos terminais da placa de controle.



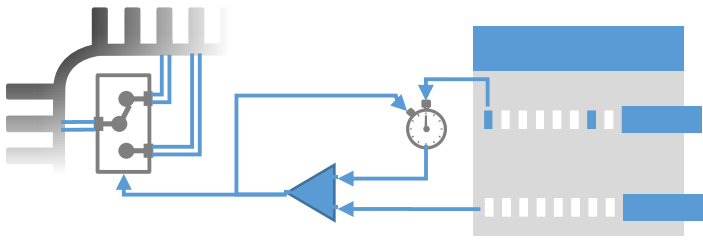
Biblioteca io

```
#ifndef IO_H_
#define IO_H_

#define bitSet(arg,bit) ((arg) |= (1<<bit))
#define bitClr(arg,bit) ((arg) &= ~(1<<bit))
#define bitFlp(arg,bit) ((arg) ^= (1<<bit))
#define bitTst(arg,bit) ((arg) & (1<<bit))
#define OUTPUT 0
#define INPUT  1
#define LOW 0
#define HIGH 1

//definição das funções dos terminais físicos
#define SCL_PIN       19
#define keyb1           13
#define DISP1_PIN     2
//...
        void pinMode(int pin, int type);
        void digitalWrite(int pin, int value);
        int digitalRead(int pin);
        void systemInit(void);
#endif /* IO_H_ */
```



Biblioteca ***io***

```
void digitalWrite(int pin, int value) {
        if (value) {
                switch (pin) {
                case 0:bitSet(PORTB_PDOR, 2);break;
                case 1:bitSet(PORTB_PDOR, 1);break;
                case 2:bitSet(PORTA_PDOR, 11);break;
                //...
                default:break;
                }
        } else {
                switch (pin) {
                case 0:bitClr(PORTB_PDOR, 2);break;
                case 1:bitClr(PORTB_PDOR, 1);break;
                case 2:bitClr(PORTA_PDOR, 11);break;
                //...
                default:break;
                }
        }
}
```



Biblioteca io

```
int digitalRead(int pin) {
        switch (pin) {
        case 0:return bitTst(PORTB_PDIR, 2);
        case 1:return bitTst(PORTB_PDIR, 1);
        case 2:return bitTst(PORTA_PDIR, 11);
        case 3:return bitTst(PORTB_PDIR, 5);
        case 4:return bitTst(PORTA_PDIR, 10);
        //...
        default:break;
        }
        return -1;
}
```