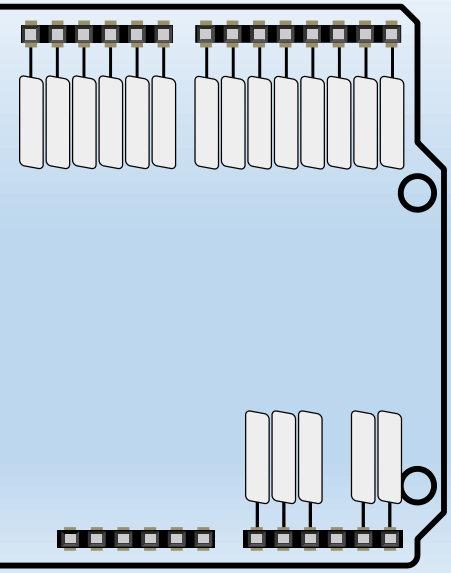
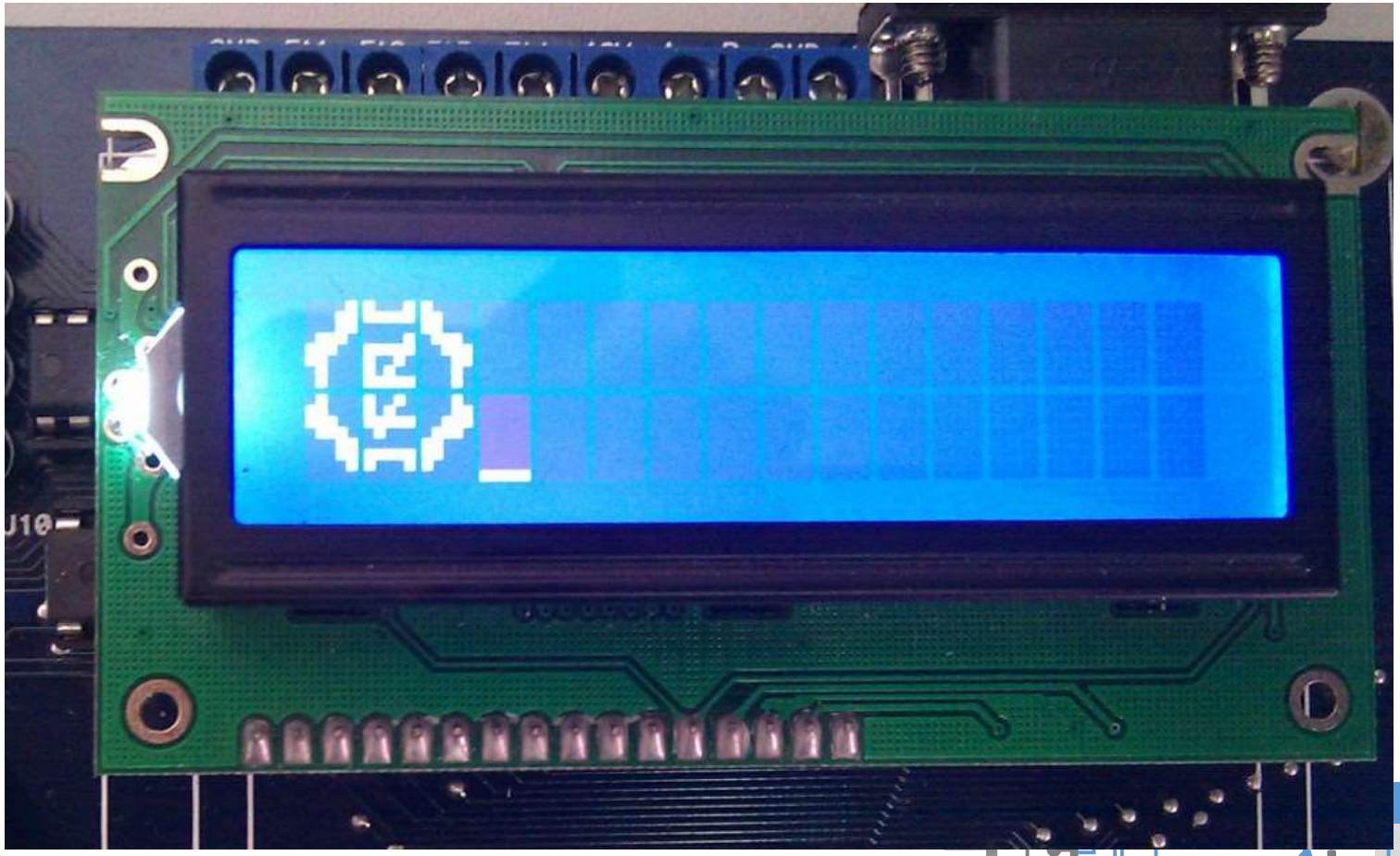
Display de LCD



O que me deixou orgulhoso foi que usei poucas peças para construir um computador que poderia realmente exibir palavras numa tela e digitar palavras num teclado e rodar uma linguagem de programação que poderia executar jogos. E eu fiz tudo isso sozinho.

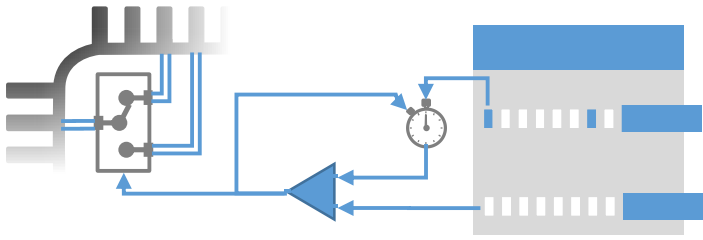
Steve Wozniak

Display de LCD



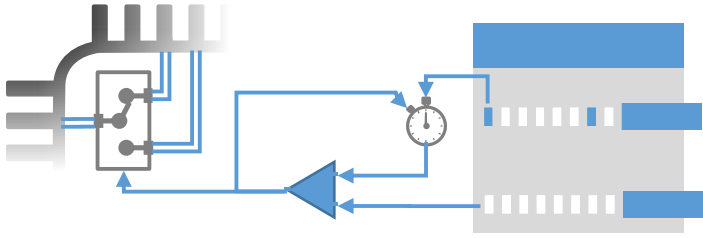
Display de LCD

- Utiliza-se do conceito de polarização da luz para formar a imagem
- Os pixels são formados por retenção de luz.
  - No caso dos displays com leds, os pixels são formados por emissão de luz.
- Geralmente possui um microcontrolador integrado
- São encontrados em formato de 7 segmentos, matricial ou com ícones dedicados
  - Os formatos matriciais mais comuns são 5x7 e 5x8
- No formato matricial é possível representar as letras, números e diversos símbolos



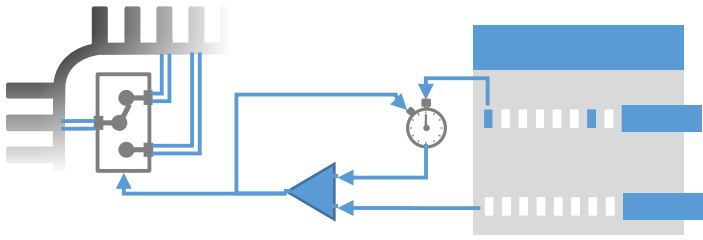
Display de LCD

- Este tipo de display pode ser visto como uma solução de exibição completa:
  - O microcontrolador faz o papel da placa de vídeo
  - O display faz o papel do monitor
  - A comunicação é feita através de um barramento paralelo
  - O barramento é comandado por 3 sinais de controle

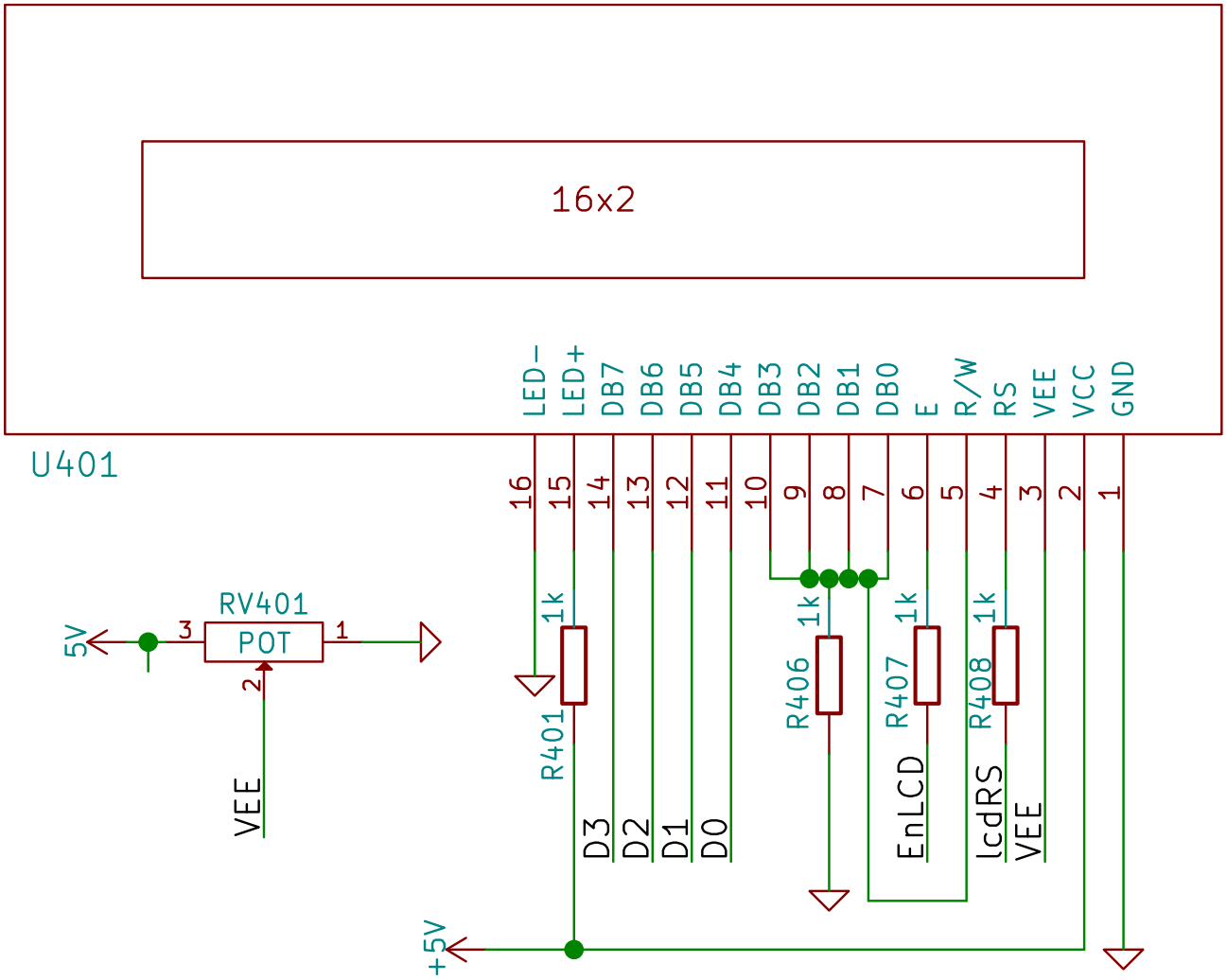


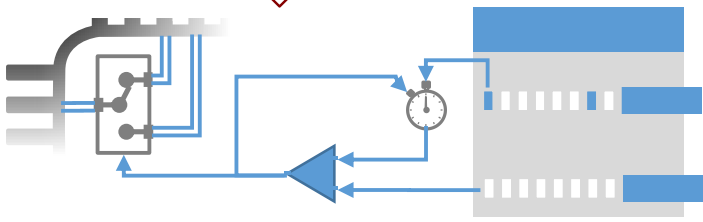
Função dos terminais

1. GND/Terra (0v)
2. VCC (+5v)
3. Ajuste do contraste
4. Seleção de registro(RS)
5. Read/Write (RW)
6. Clock, Enable (EN)
7. Bit 0
8. Bit 1
9. Bit 2
10. Bit 3
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight +
16. Backlight Gnd



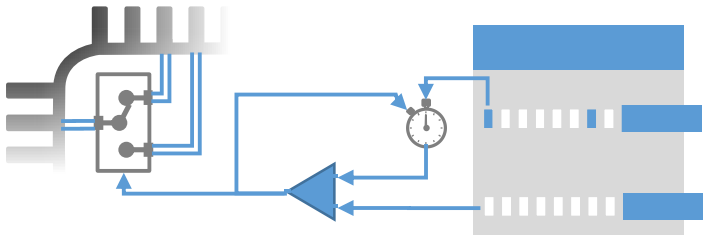
Função dos terminais



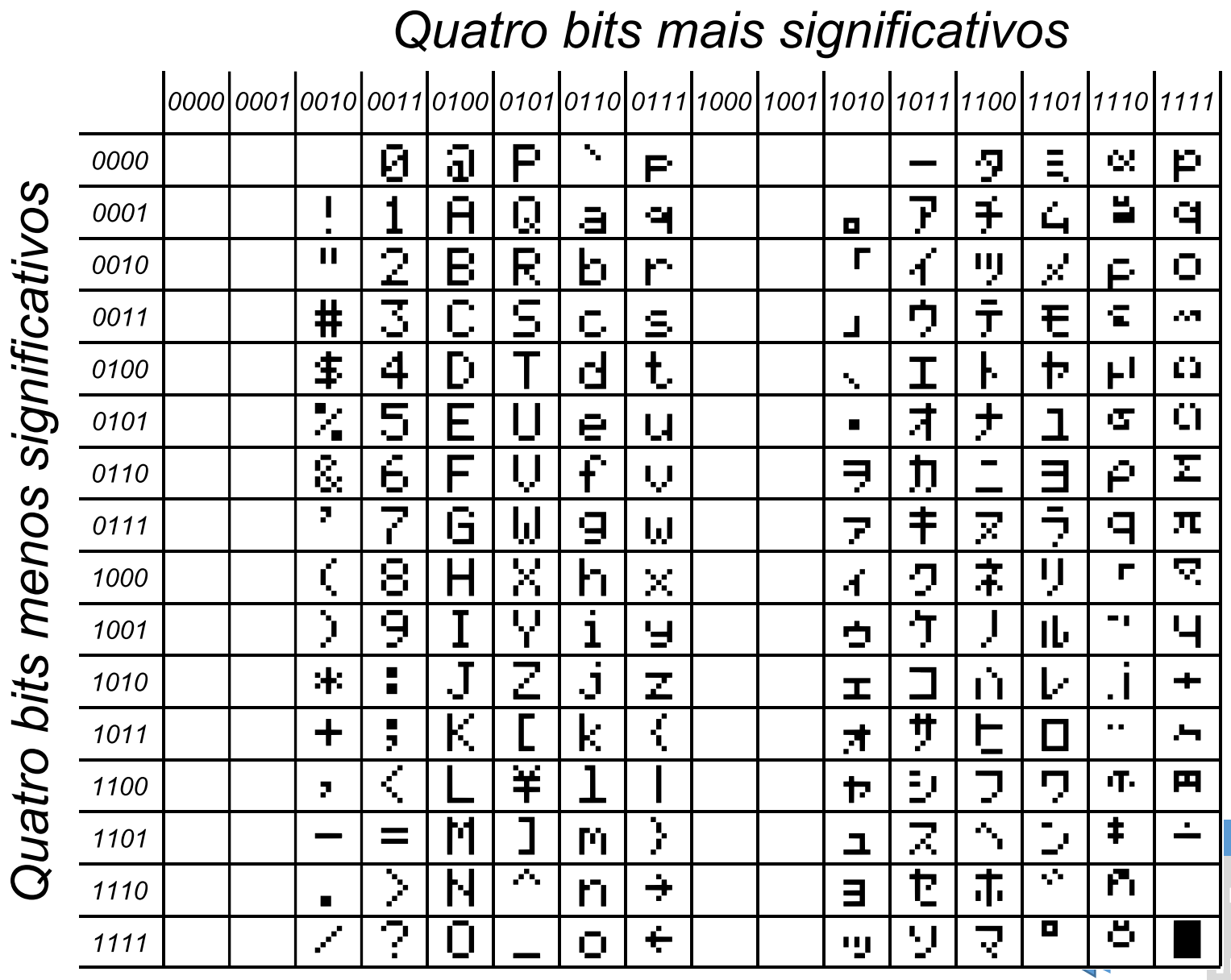


Rotina de envio de dados ou comandos

1. Para comando RS = 0; Para dados RS = 1;
2. Habilitar a escrita (RW=1) ou leitura (RW=0)
3. Colocar a informação no barramento (8 ou 4 bits)
4. Acionar terminal EN (enable)
5. Desligar o terminal EM
6. Se for comunicação de 4 bits voltar para o item 3 e colocar os últimos 4 bits
7. Delay para o LCD entender o comando/caracter

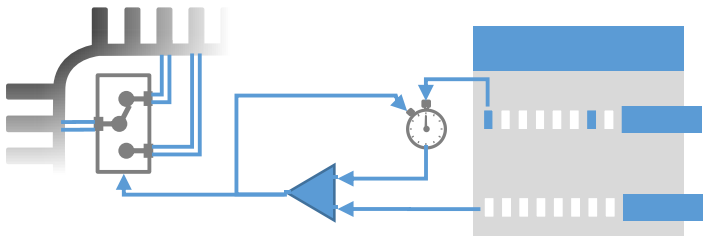


Caracteres conhecidos (ROM A00)



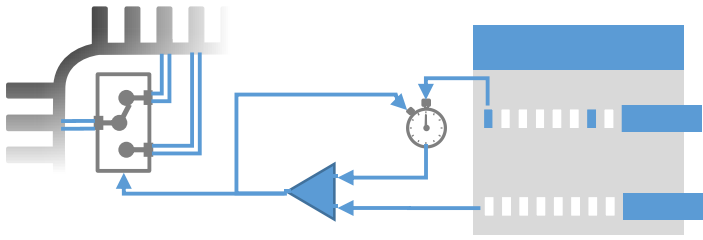
Comandos

Instrução	RS	RW	Bits								Tempo
			7	6	5	4	3	2	1	0	
Limpa	0	0	0	0	0	0	0	0	0	1	37 us
Reset	0	0	0	0	0	0	0	0	1	-	1.5ms
Config.	0	0	0	0	0	0	0	1	ID	S	37 us
Config.	0	0	0	0	0	0	1	D	C	B	37 us
Movim.	0	0	0	0	0	1	SC	RL	-	-	37 us
Config.	0	0	0	0	1	DL	N	F	-	-	37 us
Movim (l,c)	0	0	1	X	0	0	Coluna				37 us
Ocup.	0	1	BF	-	-	-	-	-	-	-	10 us



Opções dos comandos

- ID: 1 -- Incrementa, 0 -- Decrementa
- S: 1 -- O display acompanha o deslocamento
- SC: 1 -- Desloca o display, 0 -- Desloca o cursor
- RL: 1 -- Move para direita, 0 -- Move para esquerda
- DL: 1 -- 8 bits, 0 -- 4 bits
- N: 1 -- 2 linhas, 0 -- 1 linha
- F: 1 -- 5x10 pontos, 0 -- 5x8 pontos
- BF: 1 -- Ocupado, 0 -- Disponível
- X: 1 -- 2a linha, 0 -- 1a linha
- Coluna: nibble indicativo da coluna

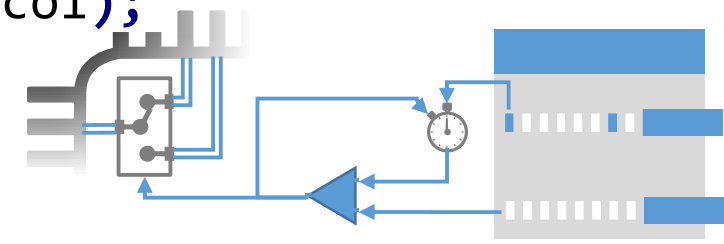


Criação da biblioteca LCD

Desenvolvimento da biblioteca

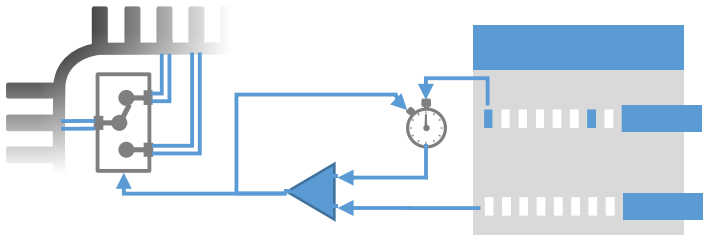
- Função para enviar comandos
- Função para enviar dados
- Função de inicialização
- Funções de impressão pré-formatada
  - Texto (string) e Número (inteiros)
- Função de controle de posicionamento

```
#ifndef LCD
#define LCD
  void lcdCommand(char value);
  void lcdChar(char value);
  void lcdInit(void);
  void lcdString(char msg[]);
  void lcdNumber(int value);
  void lcdPosition(int line, int col);
#endif
```



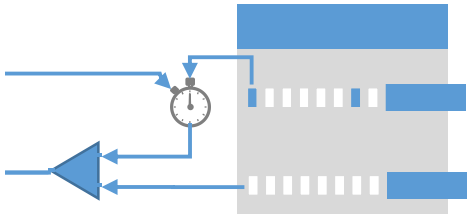
Desenvolvimento da biblioteca

- Além das funções disponibilizadas pela biblioteca, foram criadas funções internas:
  - Duas rotinas de delay
  - Geração de pulso de clock
  - Envio de 4 bits de dados
  - Envio de 8 bits de dados
- Estas funções simplificam a criação das rotinas de controle do LCD



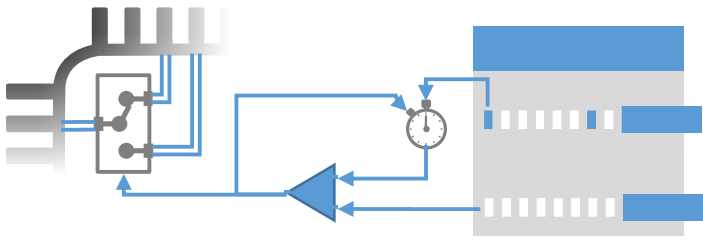
Desenvolvimento da biblioteca

```
#include "so.h"
#include "io.h"
#include "lcd.h"

void delayMicro(int a) {
        volatile int i;
        for (i = 0; i < (a * 2); i++);
}
void delayMili(int a) {
        volatile int i;
        for (i = 0; i < a; i++) {
                delayMicro(1000);
        }
}
//Gera um clock no enable
void pulseEnablePin() {
        digitalWrite(LCD_EN_PIN, HIGH);
        delayMicro(5);
        digitalWrite(LCD_EN_PIN, LOW);
        delayMicro(5);
}
```



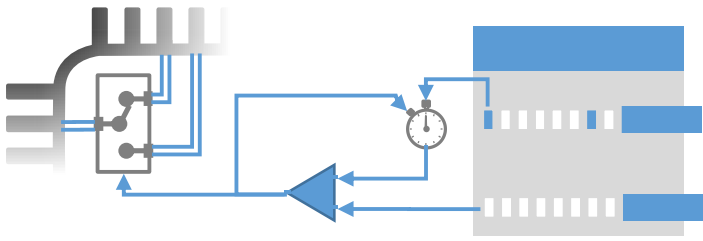
Desenvolvimento da biblioteca

```
//Envia 4 bits e gera um clock no enable
void pushNibble(char value, int rs) {
        soWrite(value);
        digitalWrite(LCD_RS_PIN, rs);
        pulseEnablePin();
}
//Envia 8 bits em dois pacotes de 4
void pushByte(char value, int rs) {
        soWrite(value >> 4);
        digitalWrite(LCD_RS_PIN, rs);
        pulseEnablePin();

        soWrite(value & 0x0F);
        digitalWrite(LCD_RS_PIN, rs);
        pulseEnablePin();
}
```



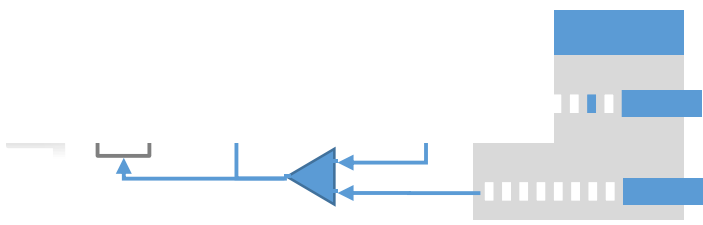
Desenvolvimento da biblioteca

```
void lcdCommand(char value) {
        pushByte(value, LOW);
        delayMili(2);
}
void lcdChar(char value) {
        pushByte(value, HIGH);
        delayMicro(80);
}
```



Desenvolvimento da biblioteca

```
void lcdPosition(int line, int col) {
        if (line == 0) { lcdCommand(0x80 + (col % 16)); }
        if (line == 1) { lcdCommand(0xC0 + (col % 16)); }
}
//Imprime um texto (vetor de char)
void lcdString(char msg[]) {
        int i = 0;
        while (msg[i] != 0) {
                lcdChar(msg[i]);
                i++;
        }
}
void lcdNumber(int value) {
        int i = 10000; //Máximo 99.999
        while (i > 0) {
                lcdChar((value / i) % 10 + 48);
                i /= 10;
        }
}
```



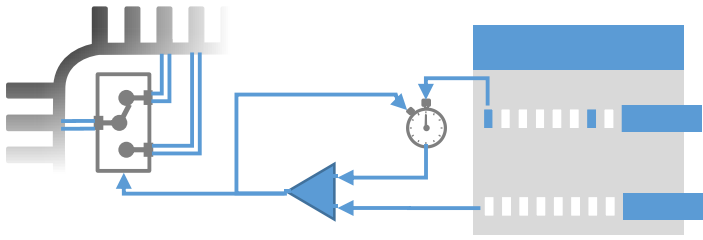
Desenvolvimento da biblioteca

```
// Rotina de incialização
void lcdInit() {
        pinMode(LCD_EN_PIN, OUTPUT);
        pinMode(LCD_RS_PIN, OUTPUT);
        soInit();
        delayMili(15);
        // Comunicação começa em estado incerto
        pushNibble(0x03, LOW);
        delayMili(5);
        pushNibble(0x03, LOW);
        delayMicro(160);
        pushNibble(0x03, LOW);
        delayMicro(160);
        // Mudando comunicação para 4 bits
        pushNibble(0x02, LOW);
        delayMili(10);
        // Configura o display
        lcdCommand(0x28);          //8bits, 2 linhas, fonte: 5x8
        lcdCommand(0x08 + 0x04);   //display on
        lcdCommand(0x01);          //limpar display, posição 0
}
```

Desenhando imagens no LCD

Desenhando imagens no LCD

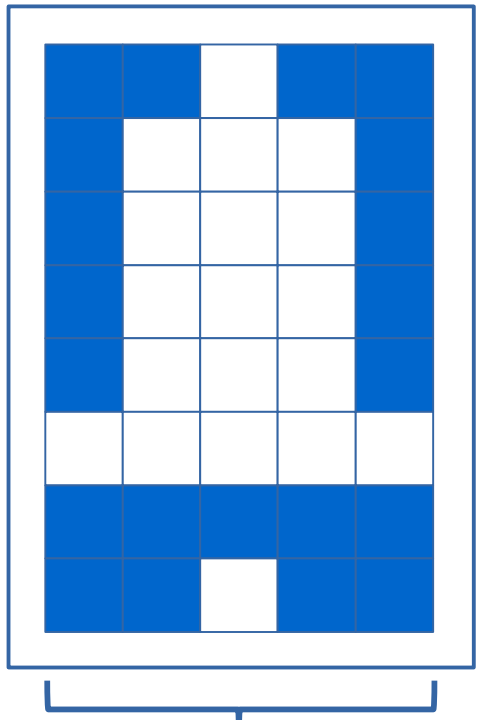
- A maioria dos LCD's permite a criação de caracteres customizados
- Para os displays compatíveis com a controladora HD44780:
  - 8 caracteres disponíveis
  - Formato binário e matricial de 8*5
  - Armazenados a partir do endereço 0x40
  - Os valores são salvos por linha de caracter



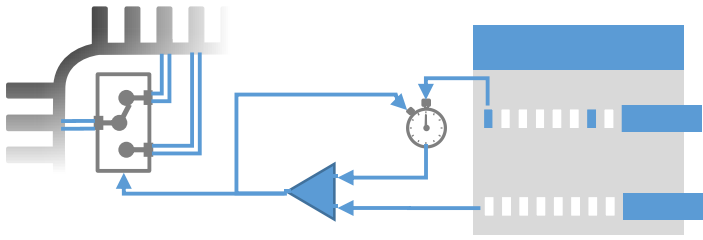
Desenhando imagens no LCD

0x40 0x47	Caracter 0
0x48 0x4F	Caracter 1
0x50 0x57	Caracter 2
	...
0x70 0x77	Caracter 6
0x78 0x7F	Caracter 7

0x50	1a linha	0x04
0x51	2a linha	0x0E
0x52	3a linha	0x0E
0x53	4a linha	0x0E
0x54	5a linha	0x0E
0x55	6a linha	0x1F
0x56	7a linha	0x00
0x57	8a linha	0x04

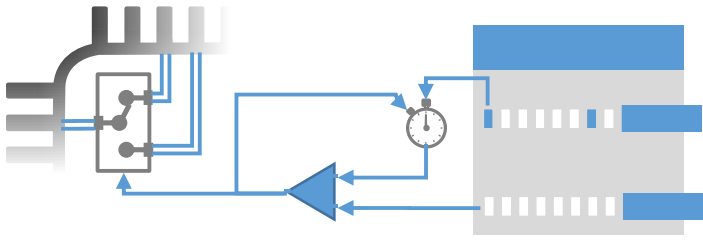


Apenas 5 bits por linha!



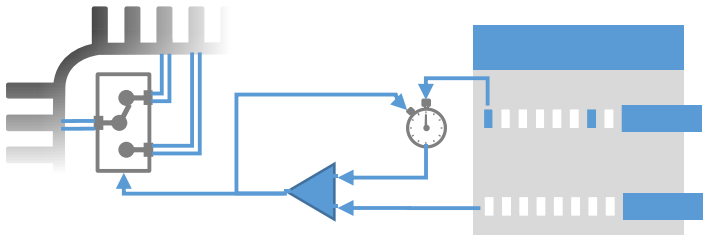
Enviando o símbolo para o lcd

```
//Cada linha é representada por um caracter
char sino[8] =  {0x04, 0x0E, 0x0E, 0x0E, 0x0E,
0x1F, 0x00, 0x04};
//Configura para a primeira posição de memória
lcdCommand(0x40);
//Envia cada uma das linhas em ordem
for(i=0; i<8; i++){
   lcdChar(sino[i]);
}
```



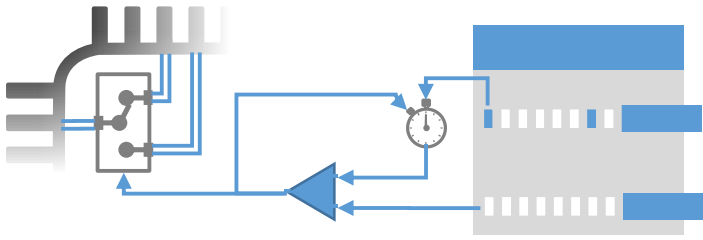
Desenhando imagens no LCD

- É possível criar um desenho/imagem de até 20*16 pixels (4*2 caracteres)
- A imagem será binária, apenas pixels brancos ou pretos
- Existe uma separação entre os caracteres, que pode prejudicar de certa maneira a imagem em questão



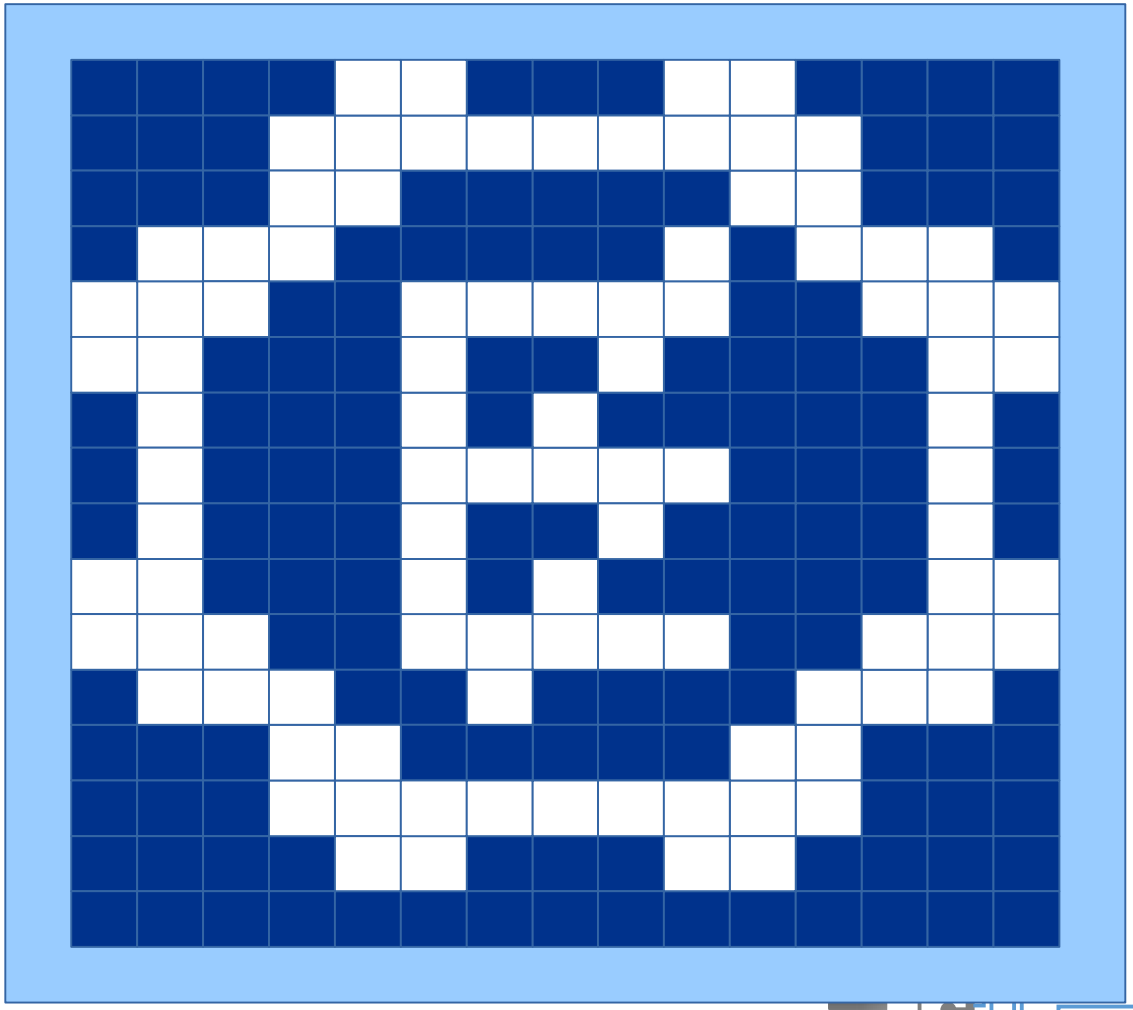
Passos para geração de uma imagem

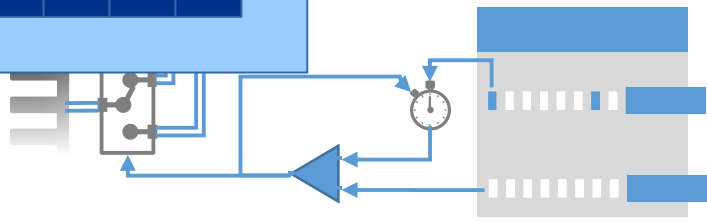
1. Criar uma imagem binária com o desenho desejado;
2. Segmentar a imagem em retângulos de 8x5;
3. Transcrever cada linha em binário/hexadecimal;
4. Gerar o código fonte.



Passos para geração de uma imagem

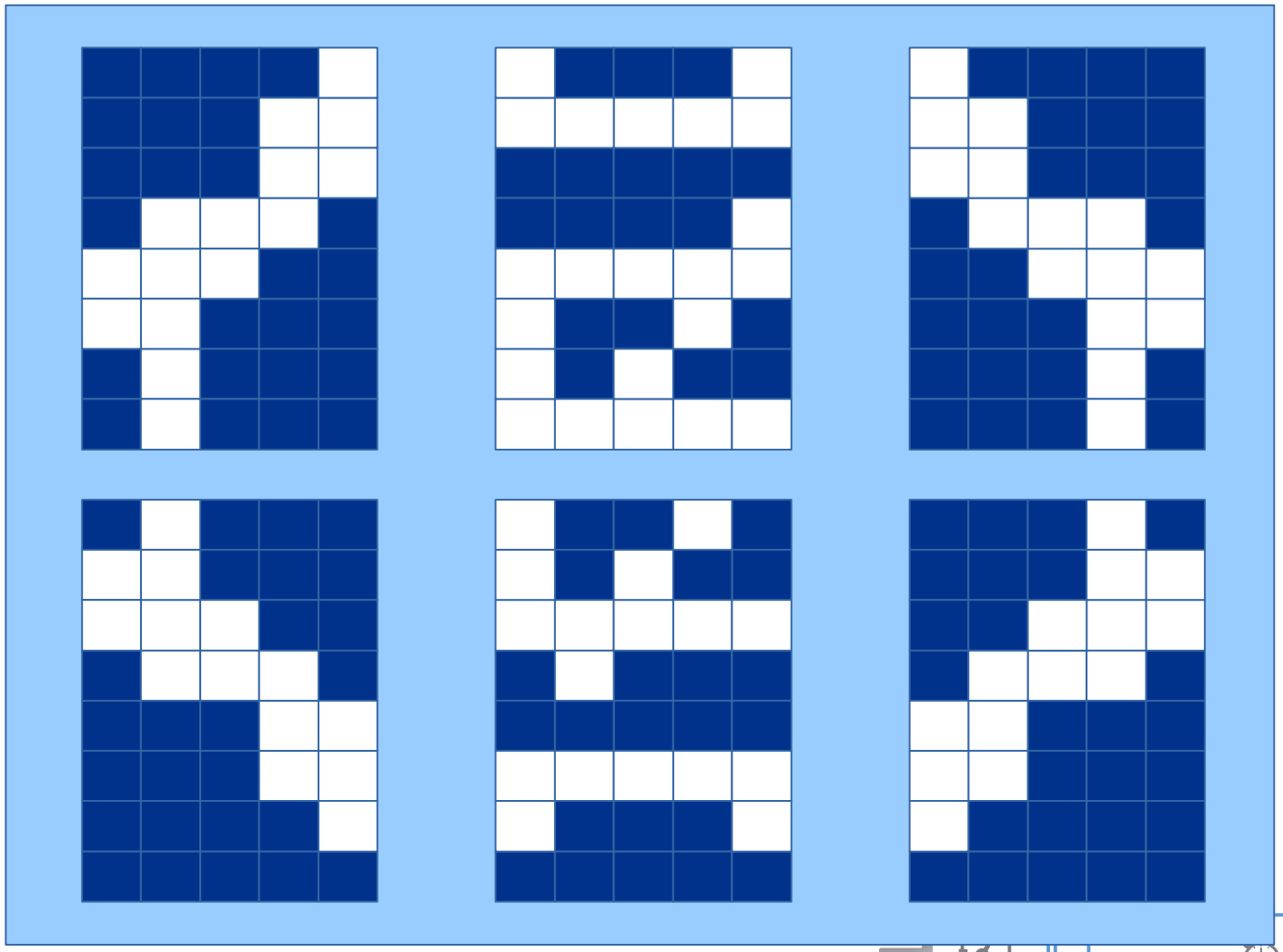
- 1 - Criar uma imagem binária com o desenho desejado;





Passos para geração de uma imagem

- 2 - Segmentar a imagem em retângulos de 8x5;



Passos para geração de uma imagem

- 3 - Transcrever cada linha em binário/hexadecimal;

0x01	0	0	0	0	1
0x03	0	0	0	1	1
0x03	0	0	0	1	1
0x0E	0	1	1	1	0
0x1C	1	1	1	0	0
0x18	1	1	0	0	0
0x08	0	1	0	0	0
0x08	0	1	0	0	0

0x11	1	0	0	0	1
0x1F	1	1	1	1	1
0x00	0	0	0	0	0
0x01	0	0	0	0	1
0x1F	1	1	1	1	1
0x12	1	0	0	1	0
0x14	1	0	1	0	0
0x1F	1	1	1	1	1

0x10	1	0	0	0	0
0x18	1	1	0	0	0
0x18	1	1	0	0	0
0x0E	0	1	1	1	0
0x07	0	0	1	1	1
0x03	0	0	0	1	1
0x02	0	0	0	1	0
0x02	0	0	0	1	0

0x08	0	1	0	0	0
0x18	1	1	0	0	0
0x1C	1	1	1	0	0
0x0E	0	1	1	1	0
0x03	0	0	0	1	1
0x03	0	0	0	1	1
0x01	0	0	0	0	1
0x00	0	0	0	0	0

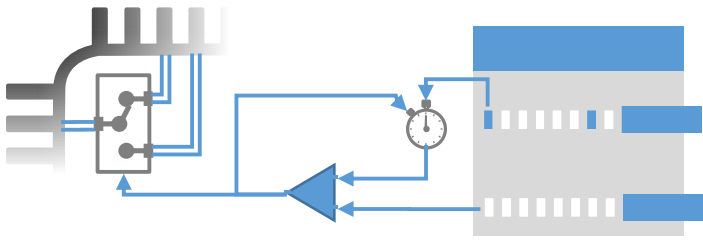
0x12	1	0	0	1	0
0x14	1	0	1	0	0
0x1F	1	1	1	1	1
0x08	0	1	0	0	0
0x00	0	0	0	0	0
0x1F	1	1	1	1	1
0x11	1	0	0	0	1
0x00	0	0	0	0	0

0x02	0	0	0	1	0
0x03	0	0	0	1	1
0x07	0	0	1	1	1
0x0E	0	1	1	1	0
0x18	1	1	0	0	0
0x18	1	1	0	0	0
0x10	1	0	0	0	0
0x00	0	0	0	0	0

Passos para geração de uma imagem

- 4 - Gerar o código fonte.

```
//Cada linha é representada por um caracter
char logo[48] = {
        0x01, 0x03, 0x03, 0x0E, 0x1C, 0x18, 0x08, 0x08, //0,0
        0x11, 0x1F, 0x00, 0x01, 0x1F, 0x12, 0x14, 0x1F, //0,1
        0x10, 0x18, 0x18, 0x0E, 0x07, 0x03, 0x02, 0x02, //0,2
        0x08, 0x18, 0x1C, 0x0E, 0x03, 0x03, 0x01, 0x00, //1,0
        0x12, 0x14, 0x1F, 0x08, 0x00, 0x1F, 0x11, 0x00, //1,1
        0x02, 0x03, 0x07, 0x0E, 0x18, 0x18, 0x10, 0x00  //1,2
};

lcdCommand(0x40); //Configura para a primeira posição de memória
//Envia cada uma das linhas em ordem
for(i=0; i<48; i++){
        lcdChar(logo[i]);
```



Passos para geração de uma imagem