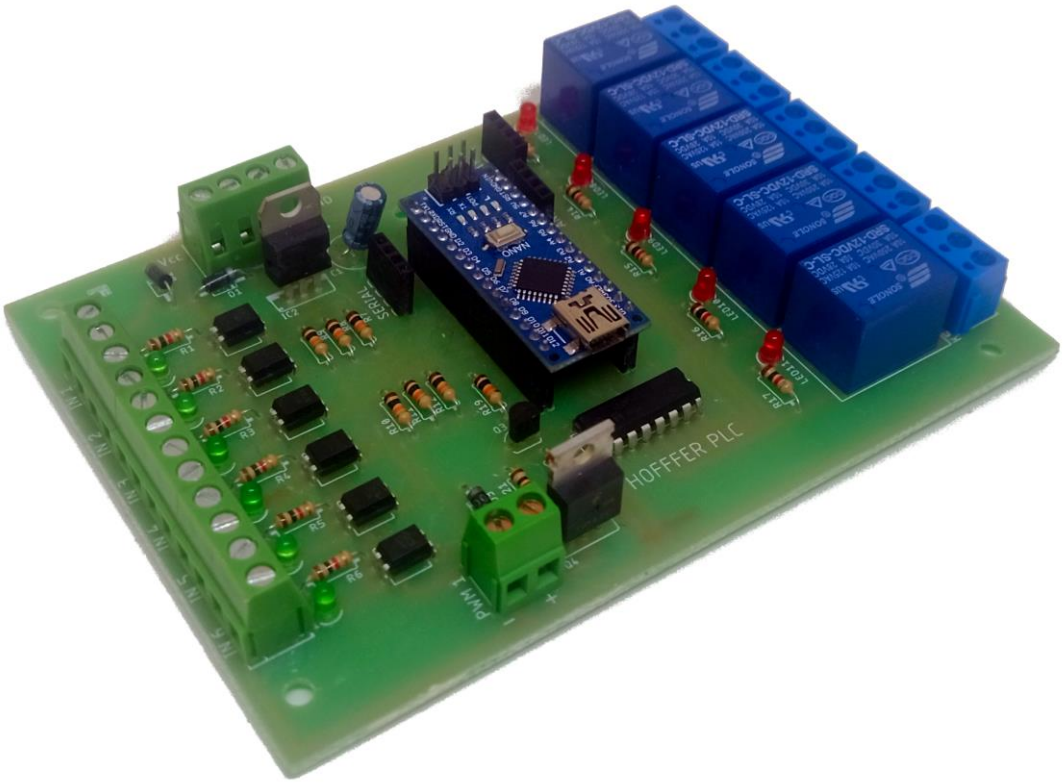
PLC Arduino HF-001



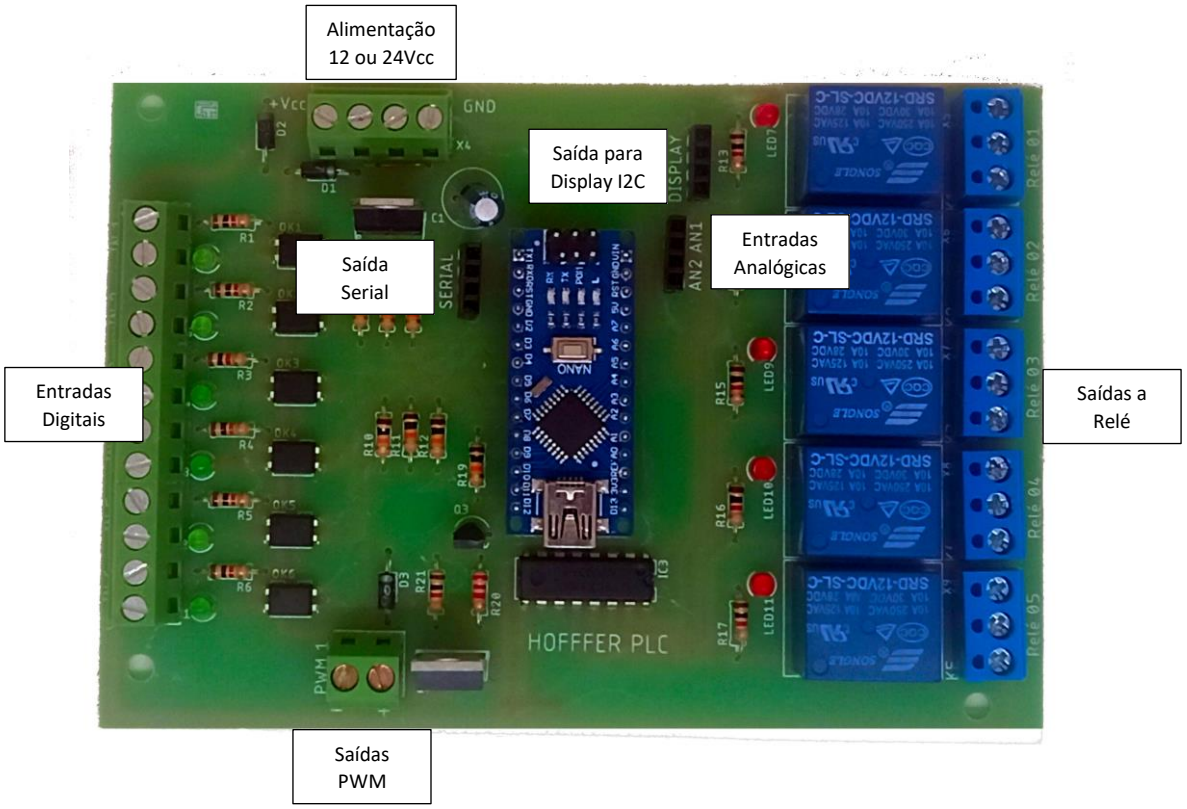
Manual de Programação e Utilização

Por Rafael F. Xidieh

Sumário

Características da placa:

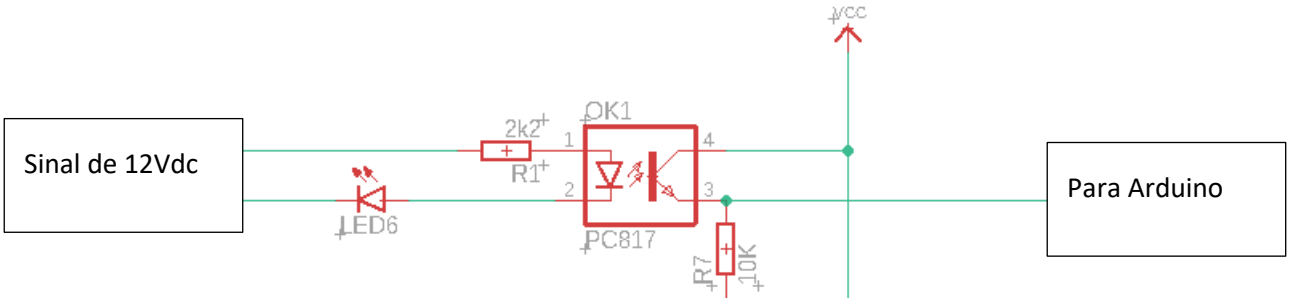
- Alimentação 12Vdc
- 6 entradas digitais isoladas opticamente
- 2 entradas analógicas de 0-5V
- 1 saída I2C para conexão de display
- 1 saída PWM com transistor PNP
- 5 saídas digitais a relé



Descrição do hardware

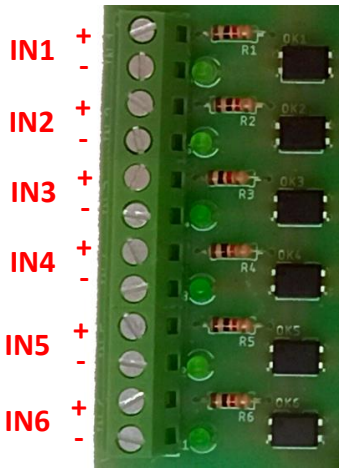
A seguir apresentaremos as características das entradas e saídas do PLC, assim como os endereçamentos a serem utilizados na IDE Arduino ou no software LDMicro.

Diagrama das entradas digitais isoladas:



As entradas digitais devem receber sinais de 12Vdc ou 24Vdc. Para facilitar a instalação pode-se conectar todos os catodos no borne GND da Fonte.

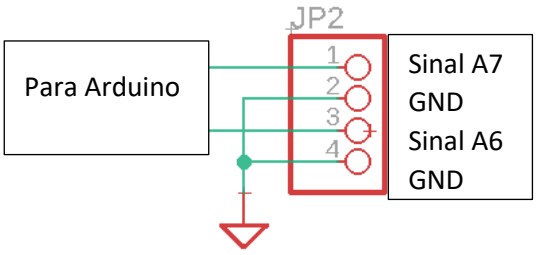
Os dois pinos do acoplador ótico são disponibilizados para permitir a conexão de sensores tipo NPN ou PNP.

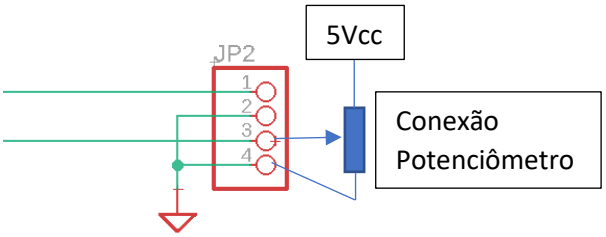


Endereçamento das Entradas digitais

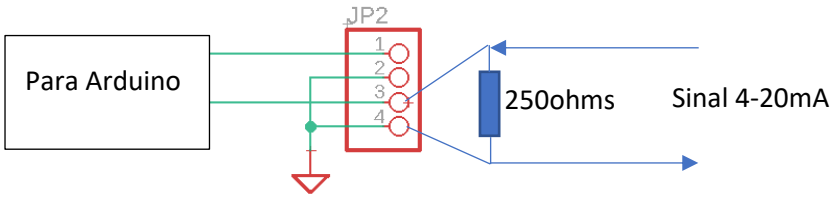
Entrada	Pino do Arduino	Endereço lógico no LDMicro
01	D2	PD2
02	D3	PD3
03	D4	PD4
04	D5	PD5
05	D6	PD6
06	D7	PD7

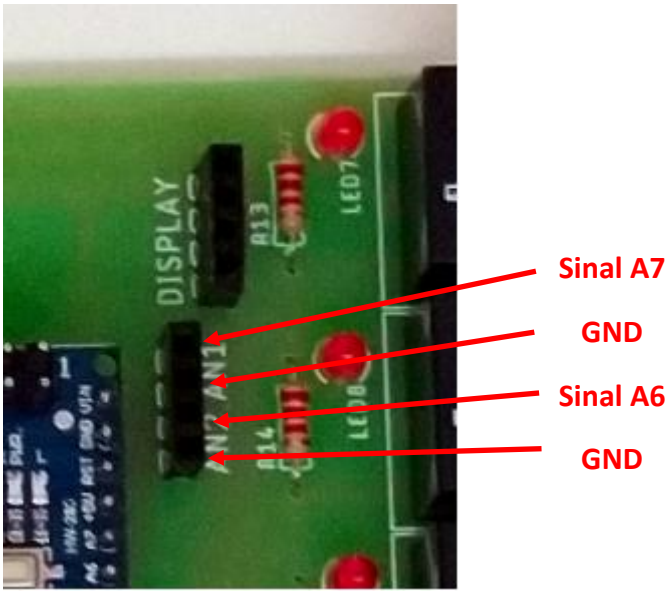
Diagrama das entradas analógicas:





As entradas analógicas devem receber sinais entre 0V e 5Vcc. Elas foram preparadas especialmente para a conexão de potenciômetros facilitando o envio de informações analógicas ao Arduino. Para conexão de sinais de corrente (0 a 20mA ou 4 a 20mA) deve-se utilizar um resistor de 250ohms em paralelo com a entrada:



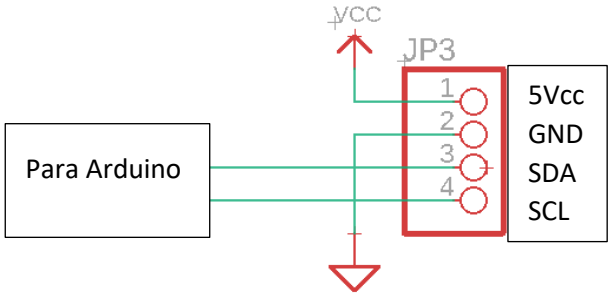


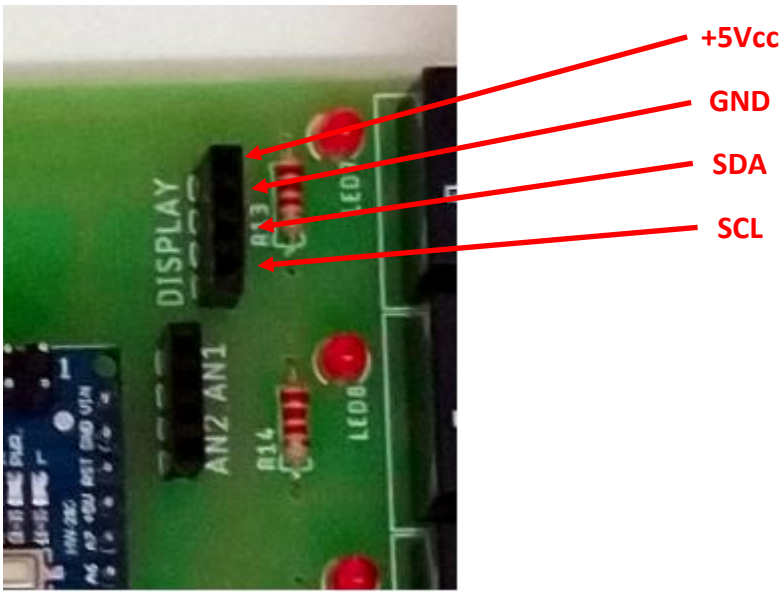
Endereçamento das Entradas Analógicas

Entradas Analógicas	Pino do Arduino	Endereço lógico no LDMicro
01	A7	ADC7
02	A6	ADC6

Diagrama da saída I2C

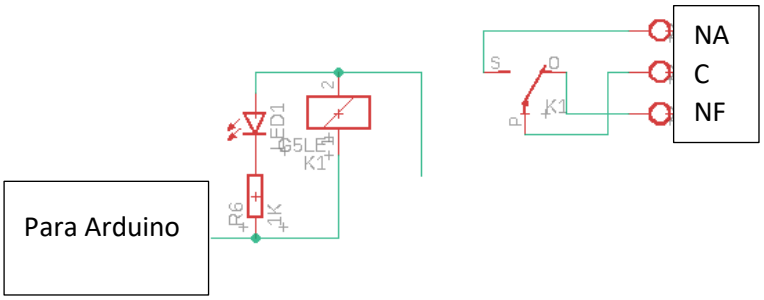
A placa possui uma porta I2C para conexão de displays seriais. A alimentação e comunicação para o LCD estão disponíveis nesta porta



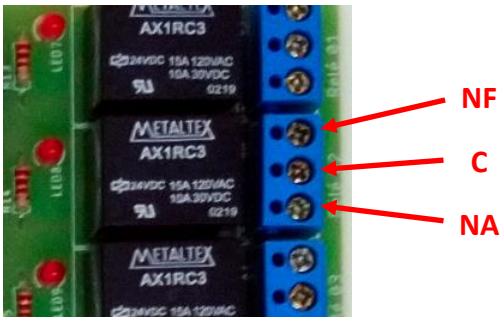


Atenção as ligações correspondentes no LCD!

Diagrama das saídas a relé:



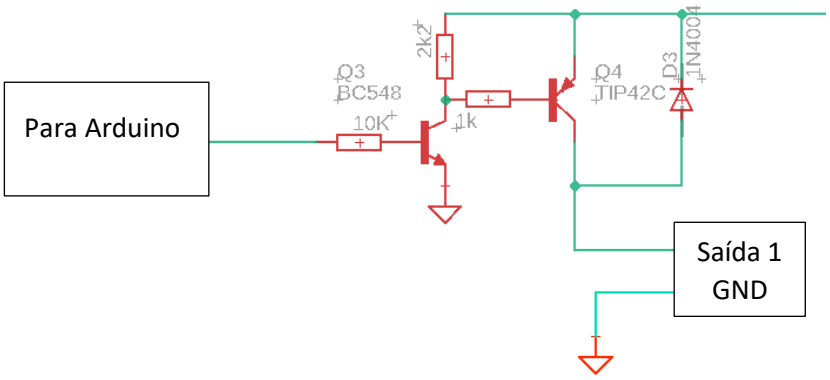
Os relés utilizados suportam até 5A em 240Vac



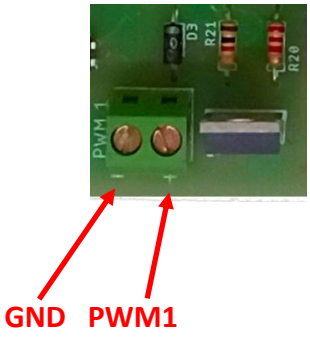
Endereçamento das Saídas Digitais a Relé

Saídas a Relé	Pino do Arduino	Endereço lógico no LDMicro
01	A3	PC3
02	A2	PC2
03	A1	PC1
04	A0	PC0
05	D8	PB0

Diagrama das saída a transistor:



Os transistores utilizados possuem diodo de proteção interno não sendo necessário adicionar esta proteção. A carga máxima recomendada é de até 2A por saída.

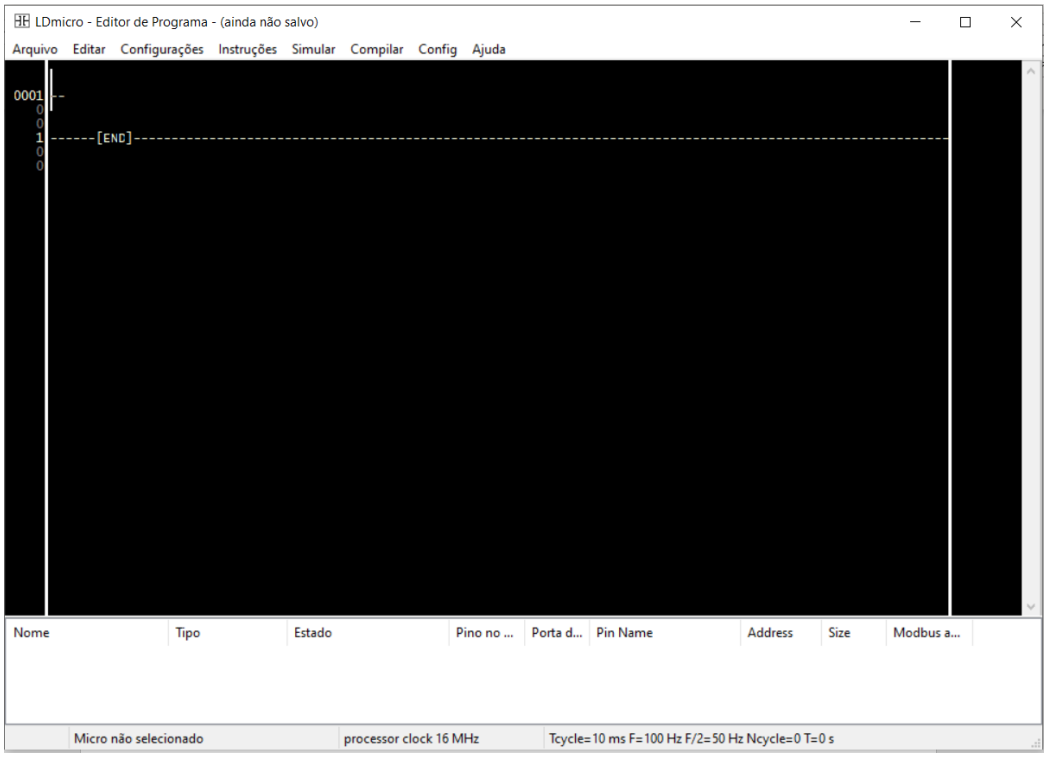


Endereçamento das Saídas Digitais a Transistor

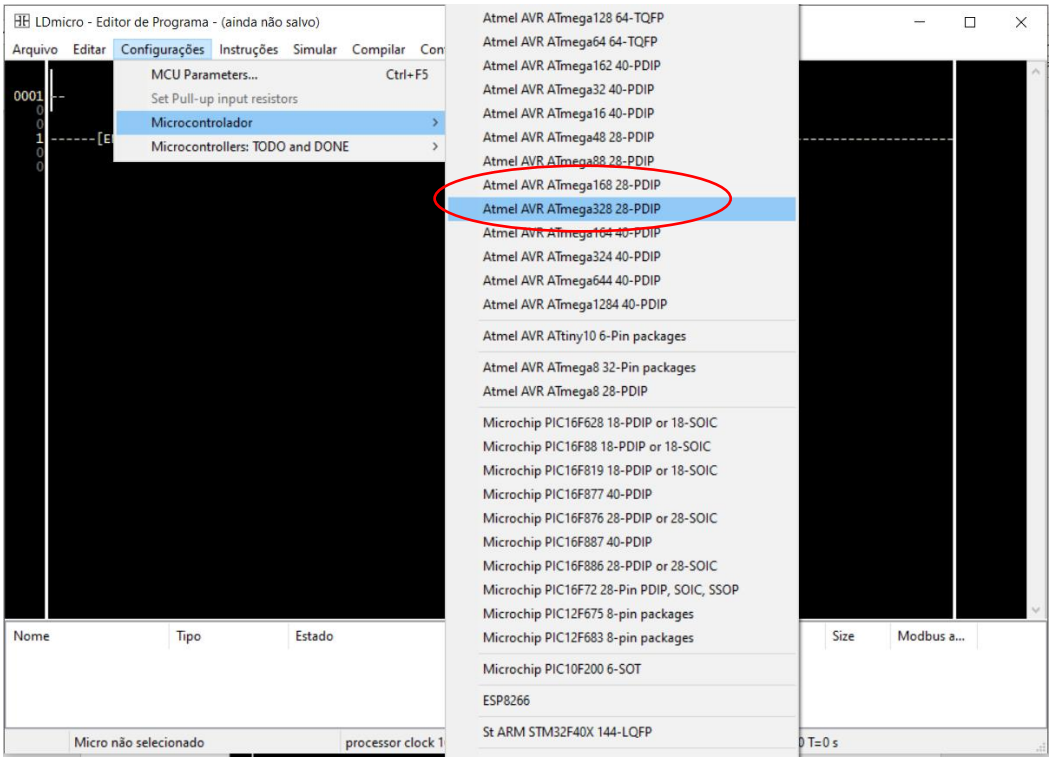
Saída	Pino do Arduino	Endereço lógico no LDMicro
01	D9	PB1

Configurando e utilizando o LDMicro

Copie a pasta LDMicro para o seu computador. Não é necessário instalar o software, basta executar o arquivo **ldmicro-pt.exe**. A seguinte tela será apresentada:



O primeiro passo é configurar o microcontrolador utilizado. Neste caso deve ser configurado o ATMega328. A figura a seguir mostra onde se faz a seleção do Arduino utilizado:



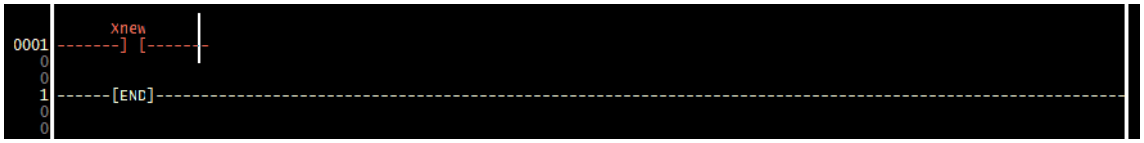
Iniciando a programação

Para iniciar a programação, clique na primeira linha que aparece na área de programa, selecione o menu “Instruções” e depois a função desejada. Pode-se utilizar alguns atalhos de teclado conforme descrito abaixo:

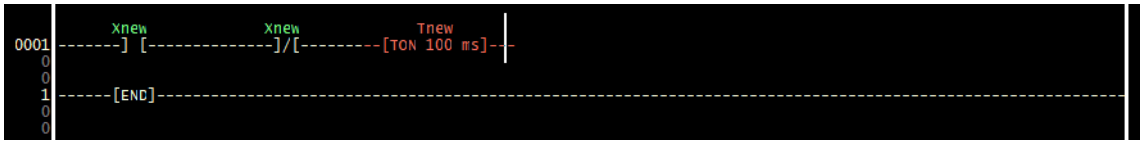
Função	Símbolo	Atalho
Contato de Entrada	----] [----	C
Contato de Entrada Negado	----]/[----	C e depois N
Contato de Saída	----( )----	L
Contato de Saída Negado	----(/)----	L e depois N
Contato de Saída “SET”	----(S)----	L e depois S
Contato de Saída “RESET”	----(R)----	L e depois R
Temporizador TON	-[TON 1.000 s]-	O
Temporizador TOFF	-[TOF 1.000 s]-	F
Detecta Borda de Subida	--[_/ OSR_/ _]--	/
Detecta Borda de Descida	--[_/ OSF_/ _]--	\

A lista completa das instruções está disponível no menu “Instruções”

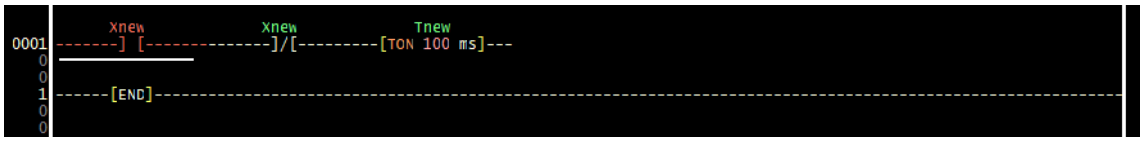
Por exemplo adicionaremos um contato de entrada simples pressionando a tecla “C”, o resultado será:



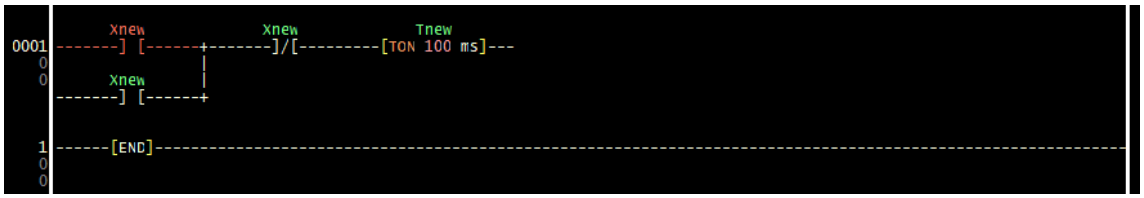
Para adicionar mais contatos, basta acrescentar as funções descritas acima ou as disponíveis no menu “instruções”. As funções são acrescentadas em série:



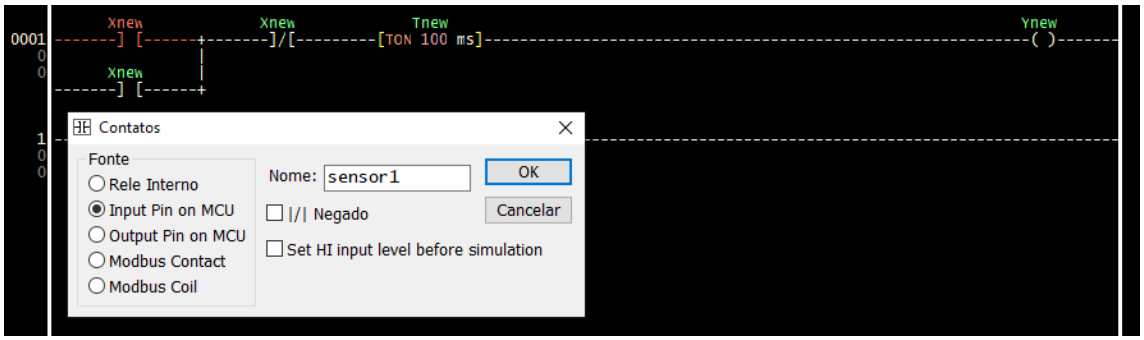
Para acrescentar um contato em paralelo, clique no contato desejado e espere o cursor aparecer na parte inferior da função selecionada:



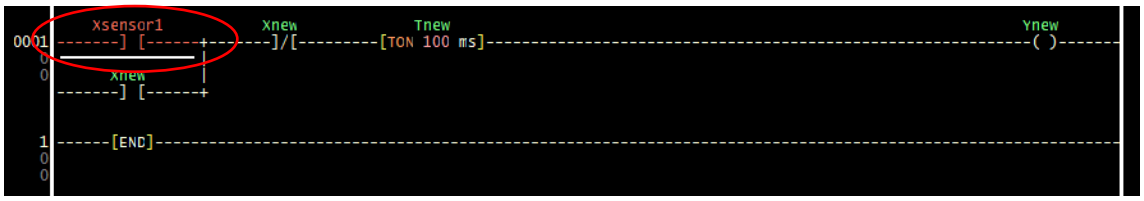
Com o cursor localizado na parte inferior, selecione a função desejada para ser inserida em paralelo:



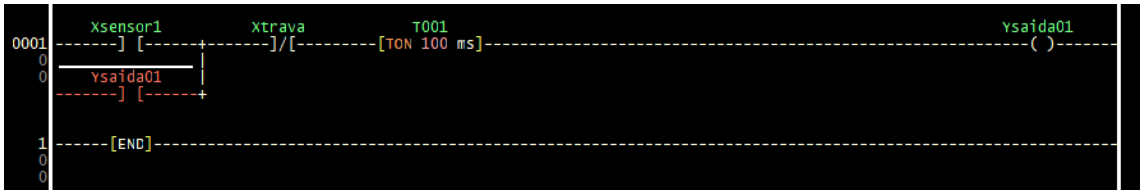
Após completar a lógica, será necessário inserir os nomes dos contatos e funções utilizados, para isso dê um duplo clique na função e insira o nome na caixa de diálogo que aparece:



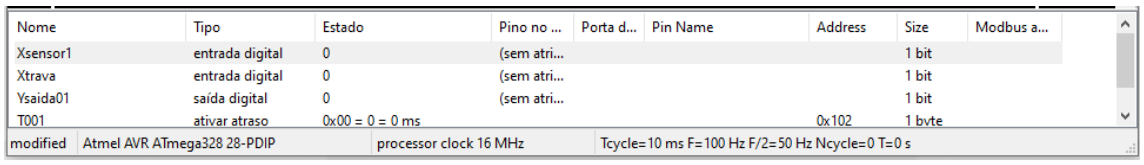
O resultado será este:



Na caixa de diálogo é possível observar outras configurações para um contato, como no caso o “RELE INTERNO” que permite criar uma variável interna ao programa que não está linkada a nenhum pino de IO. Neste exemplo foram criadas as variáveis “sensor1” , “trava”, “saída01” e esta saída faz um selo com a entrada “sensor1”. O resultado é este:



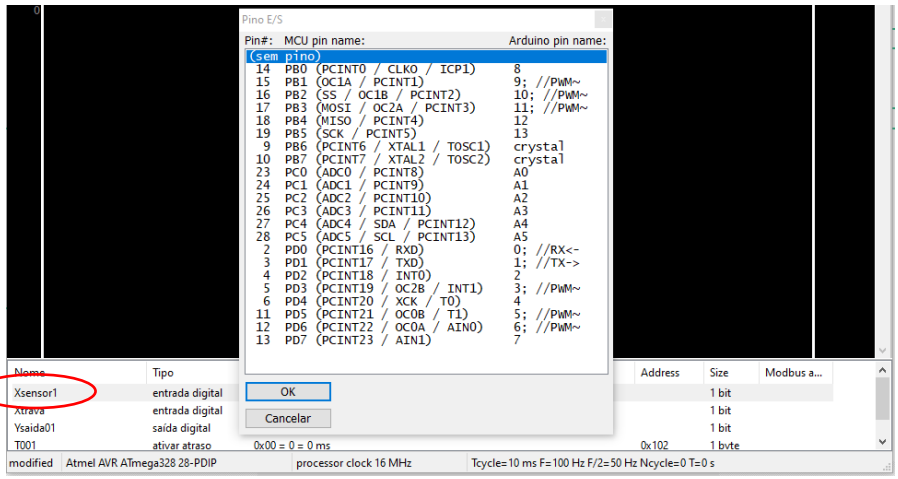
Após concluída a atribuição de nomes, todas as variáveis estarão listadas no rodapé do programa, conforme exemplo abaixo:



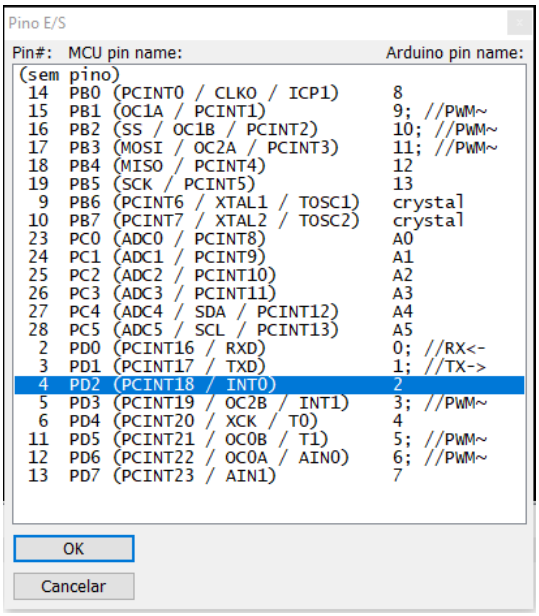
Nome	Tipo	Estado	Pino no ...	Porta d...	Pin Name	Address	Size	Modbus a...
Xsensor1	entrada digital	0	(sem atri...				1 bit	
Xtrava	entrada digital	0	(sem atri...				1 bit	
Ysaida01	saída digital	0	(sem atri...				1 bit	
T001	ativar atraso	0x00 = 0 = 0 ms				0x102	1 byte	

modified | Atmel AVR ATmega328 28-PDIP | processor clock 16 MHz | Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=0 T=0 s

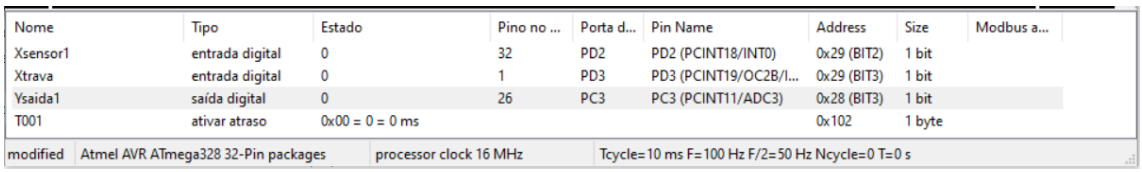
Dê um duplo clique em cada uma das variáveis criadas para atribuir um endereço físico do Arduino, conforme as tabelas de endereçamento apresentadas no início da apostila:



Neste exemplo, selecionamos a Entrada 01 para ser atribuída a variável “sensor1”, conforme a tabela, esta entrada corresponde ao endereço “PD2”:



A configuração das saídas digitais a transistor e relé seguem o mesmo procedimento para atribuição de endereços. A seguir a configuração final dos pinos do microcontrolador:

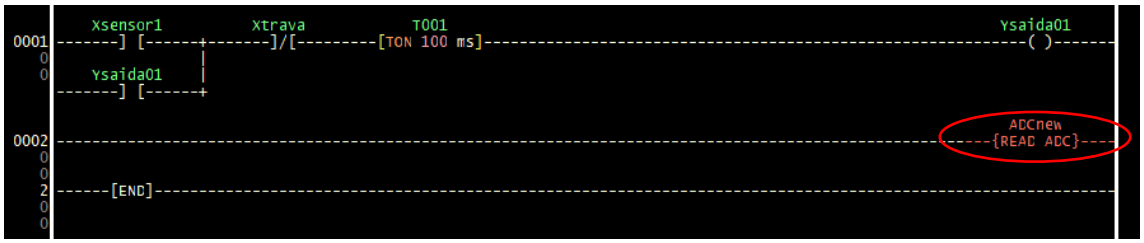


Nome	Tipo	Estado	Pino no ...	Porta d...	Pin Name	Address	Size	Modbus a...
Xsensor1	entrada digital	0	32	PD2	PD2 (PCINT18/INT0)	0x29 (BIT2)	1 bit	
Xtrava	entrada digital	0	1	PD3	PD3 (PCINT19/OC2B/I...	0x29 (BIT3)	1 bit	
Ysaida1	saída digital	0	26	PC3	PC3 (PCINT11/ADC3)	0x28 (BIT3)	1 bit	
T001	ativar atraso	0x00 = 0 = 0 ms				0x102	1 byte	

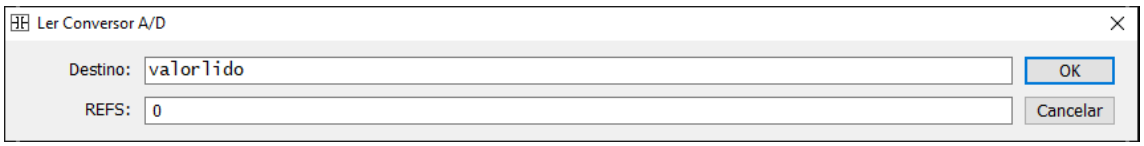
modified | Atmel AVR ATmega328 32-Pin packages | processor clock 16 MHz | Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=0 T=0 s

Utilizando as entradas analógicas

Para utilizar as entradas analógicas é necessário inserir o comando de leitura do conversor AD na lógica. Para isso, insira uma nova linha (Shift + V) e na lista de instruções, selecione o bloco de leitura AD ou simplesmente tecle “A”. O resultado é mostrado a seguir:

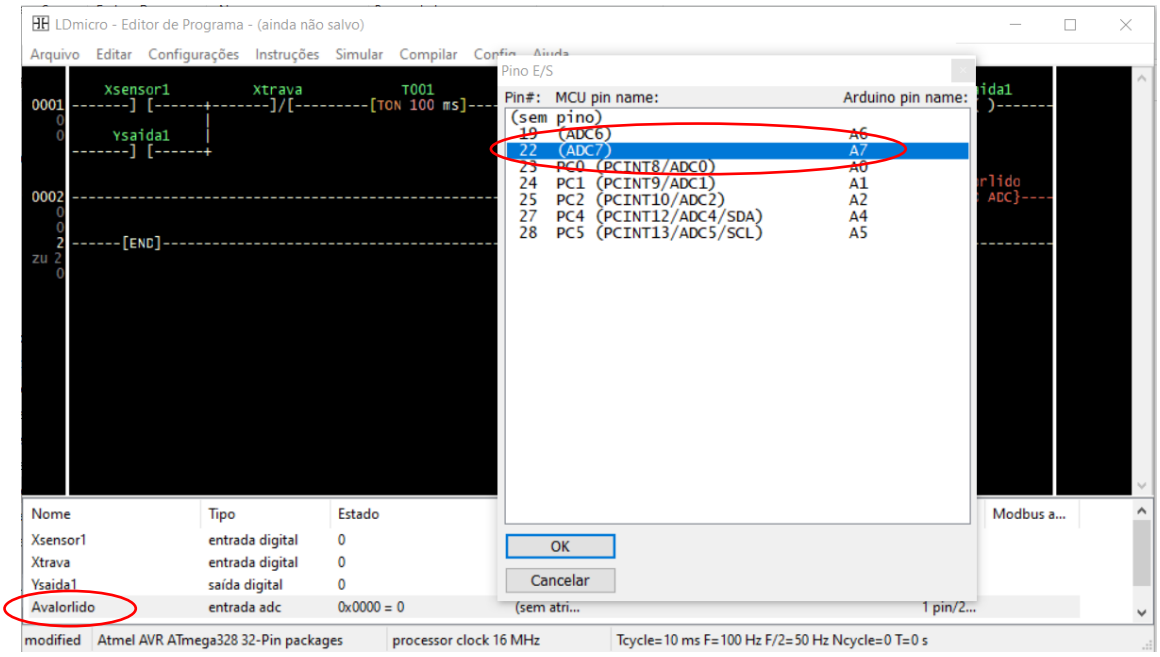


Dê um duplo clique no bloco criado para inserir a configuração da variável para o conversor AD:



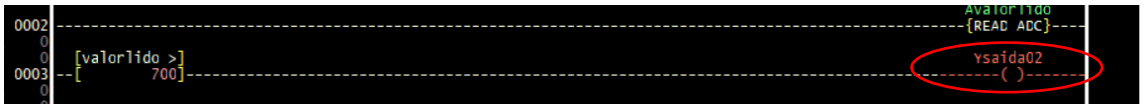
REFS deve ser mantido sempre em 0 e “Destino” contém o nome da variável do conversor AD. Repare que a variável é listada no Rodapé para atribuição do endereço físico no Arduino. Dê

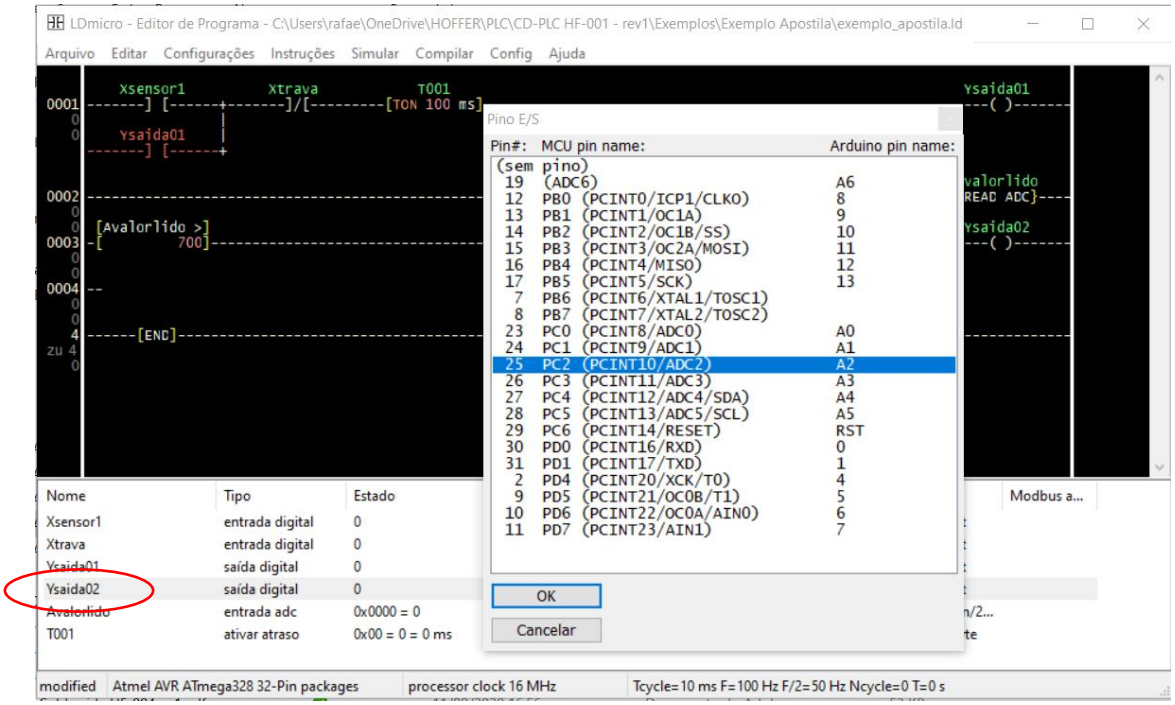
um duplo clique na variável e selecione um endereço de AD conforme a tabela de endereçamento mostrada anteriormente.



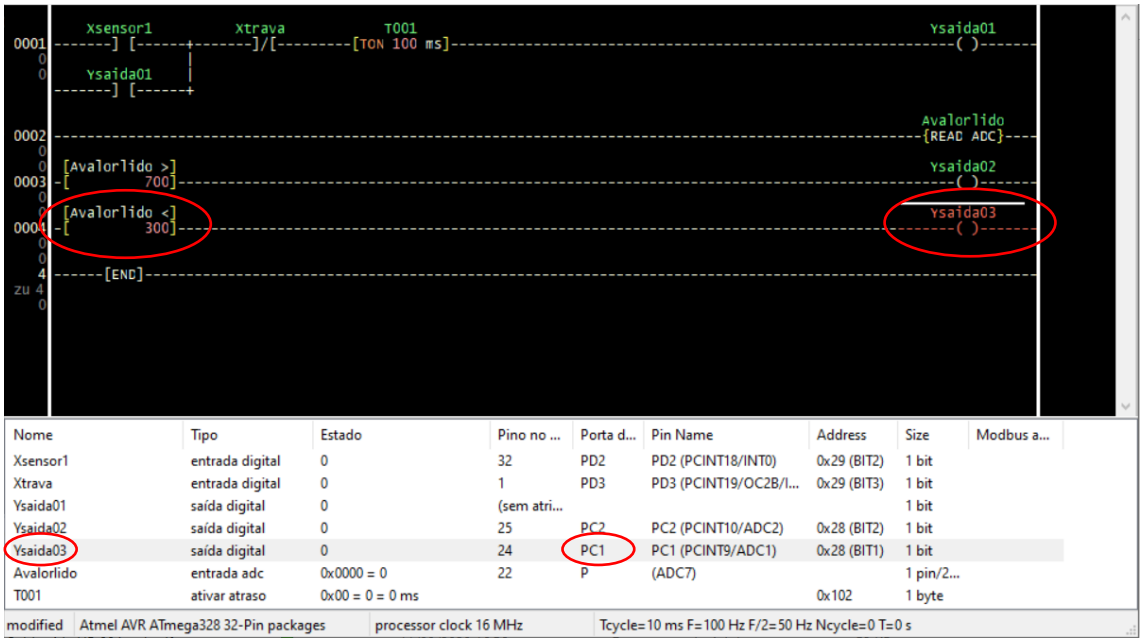
O valor lido da entrada analógica pode ser utilizado em comparadores para acionar as saídas digitais. Para adicionar um comparador “maior que” basta pressionar “>”. Após acrescentar o comparador, com um duplo clique configuramos a entrada a ser usada (Avalorlido) e o valor de comparação, neste exemplo 700.

Com o comparador configurado, acrescentamos a saída a relé 02, que corresponde ao pino PC2



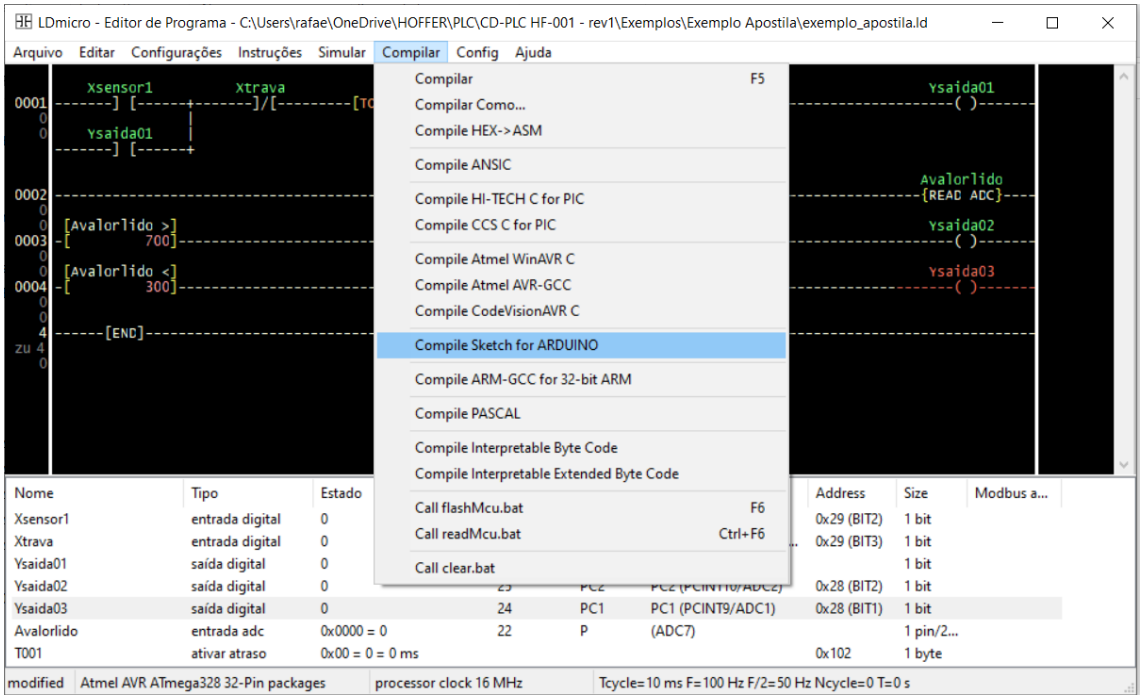


Adicionalmente acrescentamos um outro comparador “menor que” acionando a saída 03 (PC1) e com o valor de 300. Para acrescentar este comparador basta pressionar “<”.

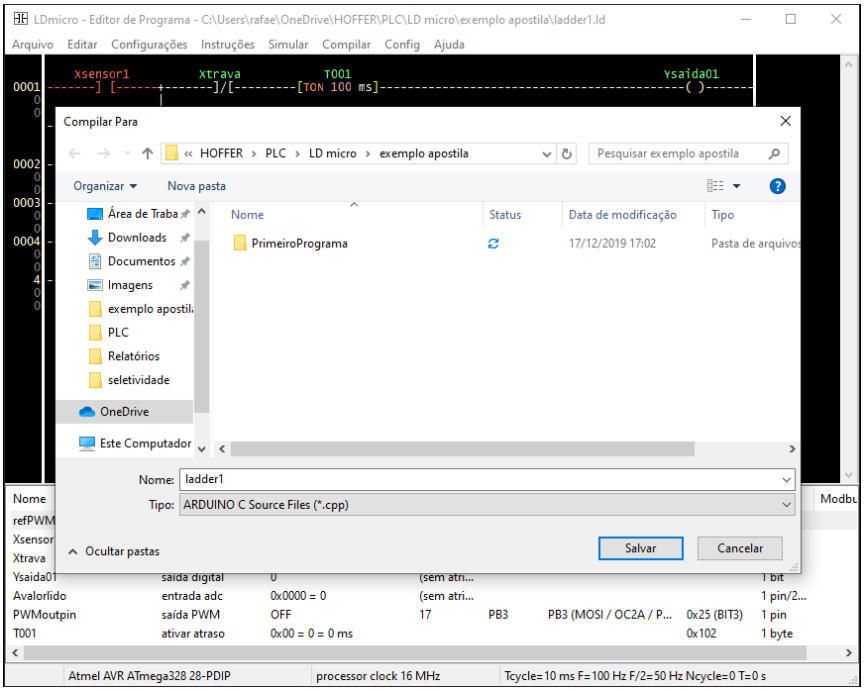


Compilando e carregando o programa

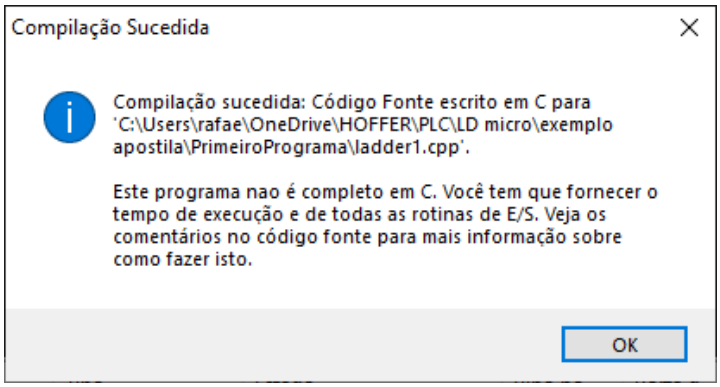
Selecione o menu “Compilar” e “Compile Sketch for ARDUINO”:



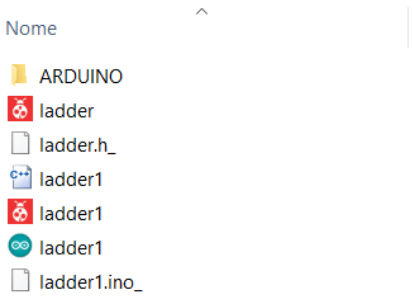
Em seguida será aberta uma janela para selecionar o local onde os arquivos gerados serão gravados. Crie uma pasta para seus arquivos como aqui no exemplo criamos a pasta “PrimeiroPrograma”



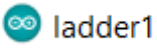
Clique em Salvar e se tudo correr bem a seguinte mensagem aparecerá informando que o código para Arduino foi criado:



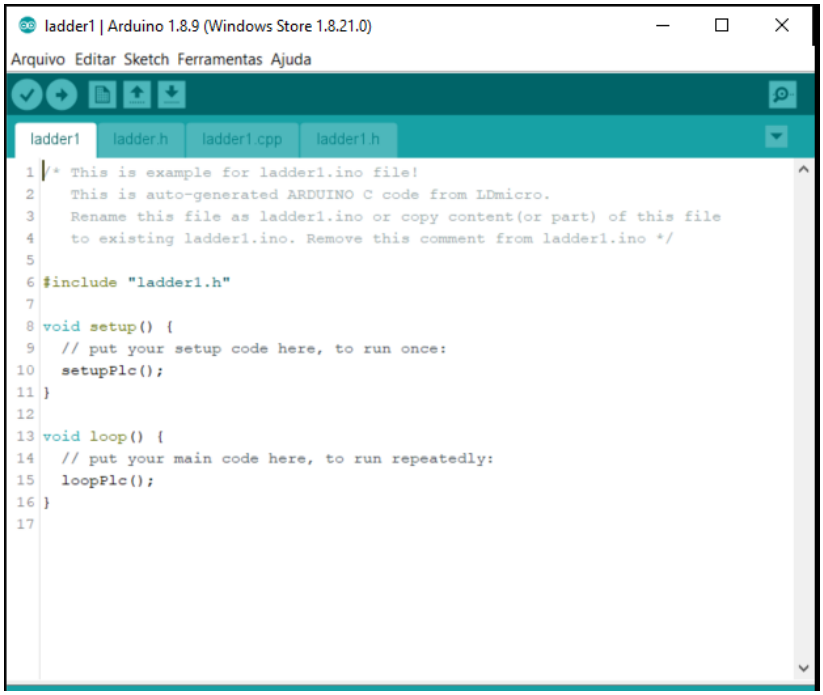
Abra a pasta onde os arquivos foram criados e examine a estrutura criada que deve ser como a apresentada a seguir:



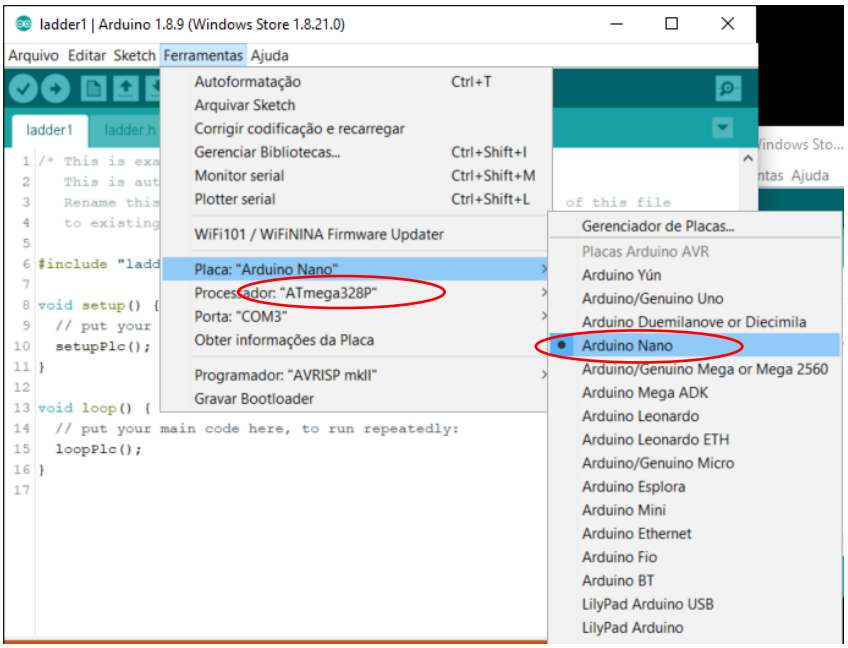
Entre na pasta “ARDUINO” e na subpasta “ladder1” ou na pasta com o nome que você inseriu no LDmicro. Dê um duplo clique no arquivo ladder1.ino ou o arquivo com o nome que você criou com a extensão “.ino”. Este arquivo aparece com o símbolo da IDE Arduino.



Neste momento a IDE Arduino será aberta e apresentará o conteúdo dos arquivos gerados pelo LDmicro:

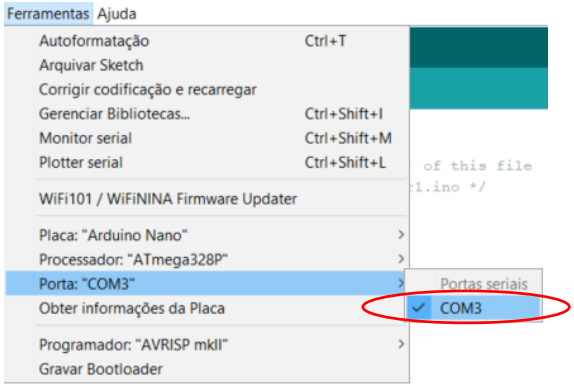


Para configurar o processador utilizado vá em Ferramentas > Placa e Selecione “Arduino Nano”. Depois em “Processador” Selecione o “ATmega328P”.

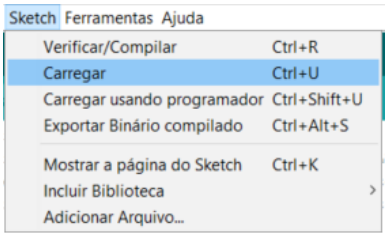


Em seguida, conecte o Arduino na porta USB com um cabo micro USB ou mini USB conforme a placa utilizada. É IMPORTANTE QUE A PLACA ESTEJA DESENERGIZADA!

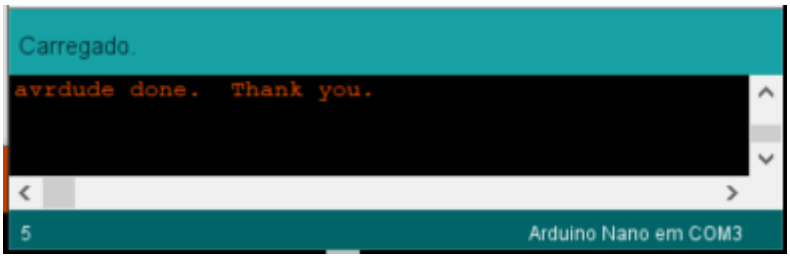
Agora selecione a porta COM correspondente em: Ferramentas > Porta.



Com a porta selecionada, carregue o programa através do menu: Sketch > Carregar.

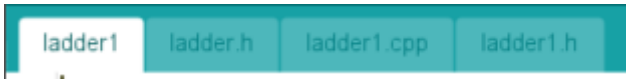


Se tudo correr bem, o programa é carregado para o Arduino e a seguinte mensagem exibida no rodapé:



Acrescentando as Funções do LCD

As funções do LCD devem ser inseridas diretamente nos arquivos criados para o Arduino. No exemplo aqui descrito, ao abrir o arquivo ladder1.ino observamos que as seguintes abas são abertas:



- O arquivo “ladder1” é o principal e será alterado para receber as funções do display
- O arquivo “ladder.h” tem diversas informações e principalmente no final, trás a configuração dos IOs que utilizaremos para comandar as mensagens no display

A seguir apresentamos a configuração de IOS que aparece no LDmicro e a correspondente configuração no fim do arquivo ladder.h.

Configuração no LDmicro:

Nome	Tipo	Estado	Pino no ...	Porta d...	Pin Name	Address	Size	Modbus a...
Xsensor1	entrada digital	0	32	PD2	PD2 (PCINT18/INT0)	0x29 (BIT2)	1 bit	
Xtrava	entrada digital	0	1	PD3	PD3 (PCINT19/OC2B/I...	0x29 (BIT3)	1 bit	
Ysaida01	saída digital	0	(sem atri...				1 bit	
Ysaida02	saída digital	0	25	PC2	PC2 (PCINT10/ADC2)	0x28 (BIT2)	1 bit	
Ysaida03	saída digital	0	24	PC1	PC1 (PCINT9/ADC1)	0x28 (BIT1)	1 bit	
Avalorlido	entrada adc	0x0000 = 0	22	P	(ADC7)		1 pin/2...	
T001	ativar atraso	0x00 = 0 = 0 ms				0x100	1 byte	

modified | Atmel AVR ATmega328 32-Pin packages | processor clock 16 MHz | Tcycle=10 ms F=100 Hz F/2=50 Hz Ncycle=0 T=0 s

Configuração no arquivo ladder.h:

```
// You must provide the I/O pin mapping for ARDUINO board in ladder.h here.
const int pin_Ub_Xsensor1 = 2; // PD2 (PCINT18/INT0)
const int pin_Ub_Ysaida01 = A3;
const int pin_Ub_Xtrava = 3; // PD3 (PCINT19/OC2B/INT1)
const int pin_Ub_Avalorlido = A7; // (ADC7) // Check that it's a ADC pin!
const int pin_Ub_Ysaida02 = A2; // PC2 (PCINT10/ADC2)
const int pin_Ub_Ysaida03 = A1; // PC1 (PCINT9/ADC1)
// You can comment or delete this line after provide the I/O pin mapping for ARDUINO board in ladder.h above.
```

Note que o software adiciona o prefixo “pin_Ub_” em cada uma das variáveis, e devemos tratá-las por estes nomes na configuração do display.

Antes de começar, devemos acrescentar as seguintes bibliotecas na IDE Arduino:

- LCD.h
- LiquidCrystal_I2C.h

Estas bibliotecas podem ser baixadas diretamente no Menu: Ferramentas > Gerenciar Bibliotecas... e no campo “Busca...” digite o nome de cada uma delas para instalação na IDE Arduino.

Outra opção é copiar os arquivos da pasta “Bibliotecas” contido no CD para a pasta Documentos > Arduino > libraries e reiniciar a IDE Arduino

Para nosso exemplo, primeiramente adicionamos no arquivo “ladder1” as seguintes linhas de configuração do display:

Arquivo original:

```
#include "ladder1.h"

void setup() {
```

Arquivo modificado:

```
#include "ladder1.h"
#include <LCD.h>
#include <LiquidCrystal_I2C.h>

//configurações display
#define I2C_ADDR          0x27        //Define I2C Address where the PCF8574A is
#define BACKLIGHT_PIN      3
#define En_pin             2
#define Rw_pin             1
#define Rs_pin             0
#define D4_pin             4
#define D5_pin             5
#define D6_pin             6
#define D7_pin             7
LiquidCrystal_I2C lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

void setup() {
```

Para facilitar o trabalho o código pode ser copiado diretamente dos arquivos do “Exemplo Apostila 1” contido no CD.

Após inseridas estas linhas, a seguinte modificação deve ser feita na função “setup()”:

Arquivo original:

```
void setup() {
  // put your setup code here, to run once:
  setupPlc();
}
```

Arquivo Modificado:

```
void setup() {
    //inicializa lcd
  lcd.begin (16,2);
  lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.setCursor(0,0);
  // put your setup code here, to run once:
  setupPlc();
}
```

Neste ponto o display já estará configurado e podemos nos atentar as mensagens que serão exibidas.

Para nosso exemplo, na primeira linha do LCD exibiremos o status da saída 1. Para isso, inserimos o seguinte código na função “loop()”:

Arquivo original:

```
32 void loop() {
33   // put your main code here, to run repeatedly:
34   loopPlc();
35 }
```

Arquivo Modificado:

```
32 void loop() {
33    if (digitalRead(pin_Ub_Ysaida01)) {       //Verifica se a saída 000 está em "1"
34     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
35     lcd.print("STATUS LED: ON  ");}          //Escreve mensagem "STS LED: ON" no LCD
36   else {                                     //Se saída 000 não for "1"
37     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
38     lcd.print("STATUS LED: OFF ");}          //Escreve mensagem "STS LED: OFF" no LCD
39   // put your main code here, to run repeatedly:
40   loopPlc();
41 }
```

Certifique-se que a mensagem tenha no máximo 16 caracteres. Caso seja menor, complete os caracteres faltantes com espaços.

Na segunda linha exibiremos o percentual do giro de um potenciômetro conectado a entrada analógica 01. Para isso, incluiremos as seguintes linhas dentro da função “loop()” já modificada:

Arquivo original:

```
32 void loop() {
33    if (digitalRead(pin_Ub_Ysaida01)) {       //Verifica se a saída 000 está em "1"
34     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
35     lcd.print("STATUS LED: ON  ");}          //Escreve mensagem "STS LED: ON" no LCD
36   else {                                     //Se saída 000 não for "1"
37     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
38     lcd.print("STATUS LED: OFF ");}          //Escreve mensagem "STS LED: OFF" no LCD
39   // put your main code here, to run repeatedly:
40   loopPlc();
41 }
```

Arquivo Modificado:

```
32 void loop() {
33    if (digitalRead(pin_Ub_Ysaida01)) {       //Verifica se a saída 000 está em "1"
34     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
35     lcd.print("STATUS LED: ON  ");}          //Escreve mensagem "STS LED: ON" no LCD
36   else {                                     //Se saída 000 não for "1"
37     lcd.setCursor(0,0);                      //Seleciona primeira linha do LCD
38     lcd.print("STATUS LED: OFF ");}          //Escreve mensagem "STS LED: OFF" no LCD
39
40   giro = analogRead(pin_Ub_Avalorlido);      // Faz a leitura do potenciometro
41   giro = map(giro,0,1023,1,100);             //Converte 0 a 100%
42   lcd.setCursor(0,1);                        //Seleciona segunda linha do LCD
43   lcd.print("POT.: " + String(giro) + "% "); //Escreve valor do potenciometro no LCD
44
45   // put your main code here, to run repeatedly:
46   loopPlc();
47 }
```

Também devemos declarar a variável “giro” logo após as configurações do lcd:

Arquivo original:

```
20 LiquidCrystal_I2C lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
21
22 void setup() {
```

Arquivo Modificado:

```
20 LiquidCrystal_I2C lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);
21
22 int giro;
23
24 void setup() {
```

Agora compile novamente o Sketch modificado e carregue no Arduino.

DIAGRAMA ESQUEMÁTICO DO PLC ARDUINO

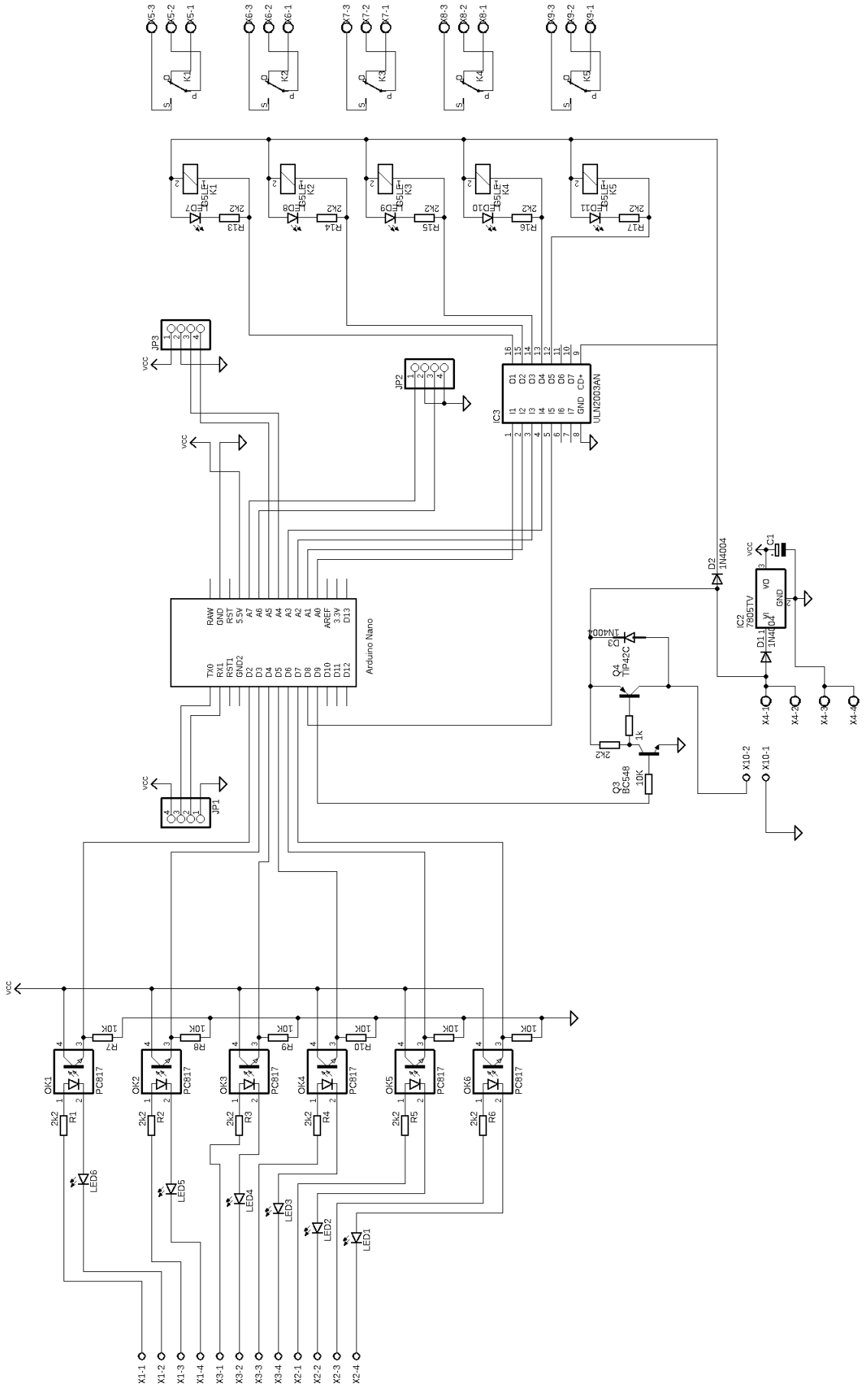


DIAGRAMA DE LIGAÇÃO PARA TESTE DOS EXEMPLOS