APRENDA AUTOMAÇÃO INDUSTRIAL COM ARDUÍNO

JORGE CÂNDIDO

APRENDA AUTOMAÇÃO INDUSTRIAL COM ARDUÍNO

INCLUI O PROJETO DE UM CONTROLADOR LÓGICO PROGRAMÁVEL COMPLETO COM ARDUÍNO.

DO PROJETO À MONTAGEM E PROGRAMAÇÃO

SOBRE O AUTOR:

JORGE CÂNDIDO É MESTRE EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO, COM MAIS QUE 30 ANOS DE EXPERIÊNCIA EM PROJETOS DE AUTOMAÇÃO INDUSTRIAL.

ENTUSIASTA DA PLATAFORMA ARDUÍNO.

CRIADOR DO SITE:

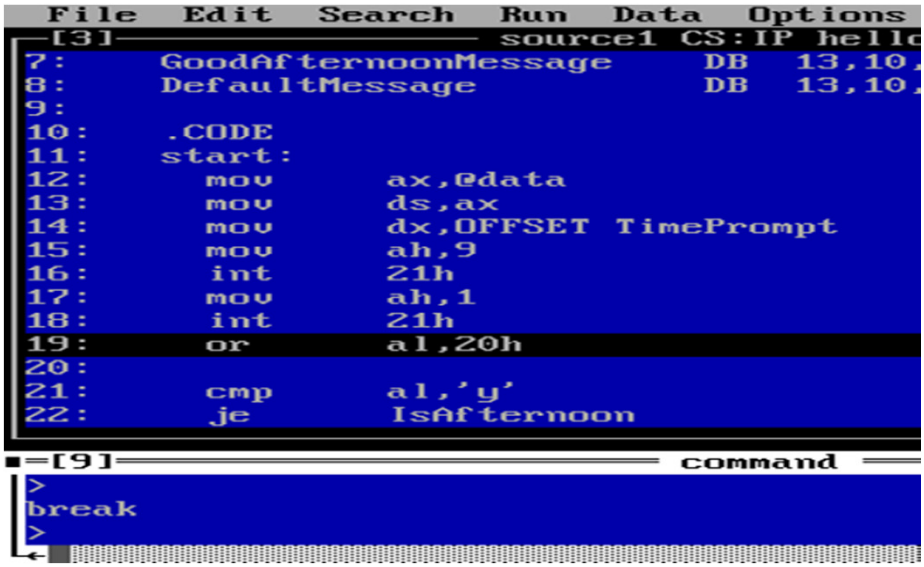
www.cursoarduino.com

SUMÁRIO

PREFÁCIO

Minha primeira impressão ao conhecer o Arduino e sua ferramenta de desenvolvimento (IDE) foi de total desprezo e indiferença. Depois de passar mais de 15 anos fazendo projetos com microcontroladores com variados níveis de complexidade, eu tinha a convicção que aprender a projetar qualquer sistema microcontrolado, era uma tarefa que exigia nada menos que um curso de graduação completo. Ou seja, "o cara tem que ser engenheiro formado", era o que eu pensava na época.

Quando eu comecei minha carreira como engenheiro (há pelo menos 20 anos atrás), para poder programar microprocessadores (microcontroladores não existiam ainda) o sujeito tinha que escrever o programa em assembler. O assembler é a linguagem nativa de qualquer processador e cada processador tem seu conjunto de instruções e registradores. Mesmo o programador mais experiente sempre tinha que consultar um manual contendo explicações de como usar cada registrador e instruções disponíveis para o processador para o qual está sendo escrito o programa.



Hoje eu reconheço que esta realidade é coisa do passado. A linguagem C virou um padrão de linguagem disponível para qualquer tipo de processador (microcontroladores inclusive). A linguagem assembler ainda existe como alicerce para programação. Mas os compiladores transformam o programa escrito em linguagem C para o equivalente em linguagem assembler para que o programa possa rodar em um microcontrolador específico.

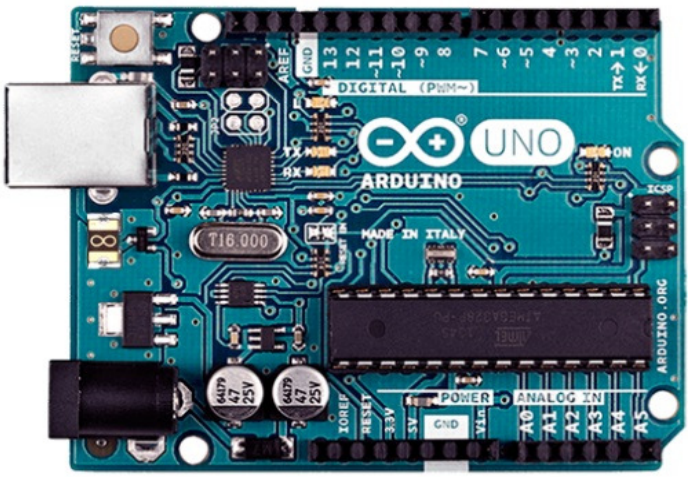
A popularização da linguagem C democratizou a etapa de desenvolvimento de programas para microcontroladores. Faltava uma plataforma de hardware que fosse acessível a quem não é projetista de hardware. Afinal de contas, desenhar um esquema eletrônico, elaborar o traçado de trilhas em um programa da CAD, mandar fabricar o PCB, comprar os componentes e montar o circuito são tarefas muito dispendiosas (em termos de tempo e dinheiro), para não dizer inviáveis fora de uma empresa de desenvolvimento de placas eletrônicas.

O Arduino veio para cobrir esta lacuna do desenvolvimento de projetos eletrônicos. A ideia é disponibilizar uma plataforma padrão com os requisitos mínimos para funcionar um projeto com microcontrolador. Estes requisitos mínimos são: fonte de alimentação, gerador de clock, circuito de reset, interface para conexão com um computador (para descarregar os programas) além do microcontrolador. Além da placa (hardware) que tem um custo muito baixo, temos a disposição uma plataforma de desenvolvimento de programas (IDE) muito simples de ser usada e mais importante, gratuita.

Estas são apenas algumas das razões que fizeram que eu mudasse de ideia em relação ao Arduino. Hoje eu sou usuário e entusiasta do Arduino. Acredito que com o auxílio desta plataforma, hobistas, técnicos e engenheiros têm condições de aprender a projetar equipamentos de uma forma divertida, fácil e muito barata. E depois de conviver com pessoas deste tipo, que anseiam por aprender eletrônica, dentro das empresas que trabalhei, eu decidi me dedicar a ensinar um pouco do que sei.

E porque ensinar automação industrial com o Arduino? Basicamente por 3 razões:

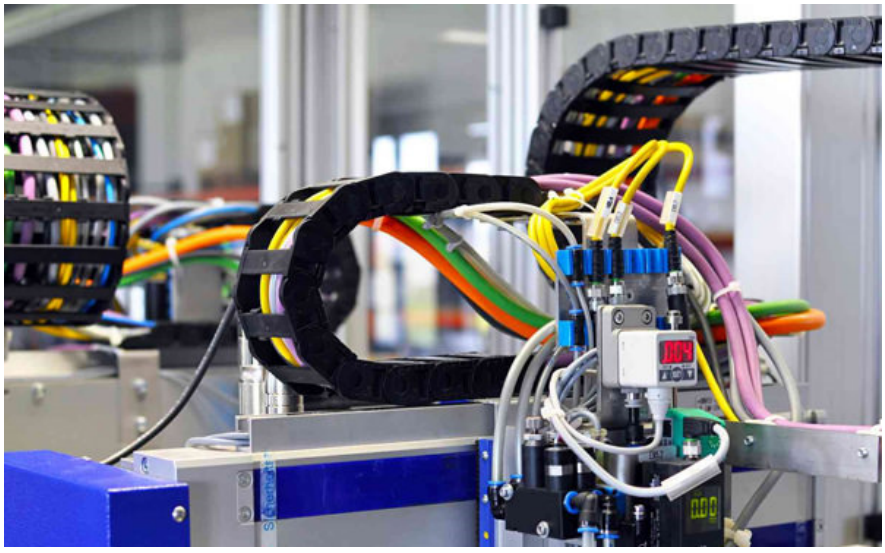
Existem bons tutoriais e cursos sobre Arduino disponíveis na internet. Quem desejar aprender o básico terá muitas opções para atingir este objetivo, até de forma gratuita. Eu mesmo criei um curso online que tem como principal objetivo fundamentar as noções básicas de eletrônica e programação usando o Arduino (www.cursoarduino.com). Neste sentido, a intenção deste livro é mostrar alguns tópicos mais avançados para aqueles que já conhecem o Arduino e já tiveram a oportunidade de montar alguns circuitos básicos.



Outra razão para o tema deste livro é que grande parte de minha experiência como engenheiro, foi trabalhando exatamente com projetos de automação industrial. Desde que eu comecei a "brincar" com o Arduino, eu enxerguei logo a oportunidade de aplicação desta ferramenta na área de automação industrial.

Assim eu me sinto muito confortável para ensinar sobre este tema e consigo detectar várias boas aplicações nesta área.

Não foi fácil escolher os tópicos a serem abordados neste livro, devido as várias possibilidades existentes. Assim escolhi detalhar como seria um projeto de controlador lógico programável feito com Arduino. A partir desses projeto o leitor será capaz de imaginar outras aplicações usando as mesmas ideias aqui apresentadas.



A última razão para escolher automação industrial como tema deste livro foi que em minha trajetória profissional sempre observei a enorme dificuldade em encontrar profissionais com experiência em projetos de automação industrial.

Anualmente centenas de recém-formados em tecnologia e engenharia são lançados no mercado de trabalho trazendo na bagagem uma pequena experiência adquirida nos laboratórios das escolas e faculdades que finalizaram. Eu sei muito bem como é valorizado nas empresas um candidato a emprego com um conhecimento diferenciado nesta área.

Espero com este livro ajudar tanto os leitores, a adquirir um conhecimento que normalmente só se consegue dentro das empresas, quanto as empresas que poderão ter a disposição candidatos mais bem preparados em uma seleção de emprego.

02

CLP CONTROLADOR LÓGICO PROGRAMÁVEL

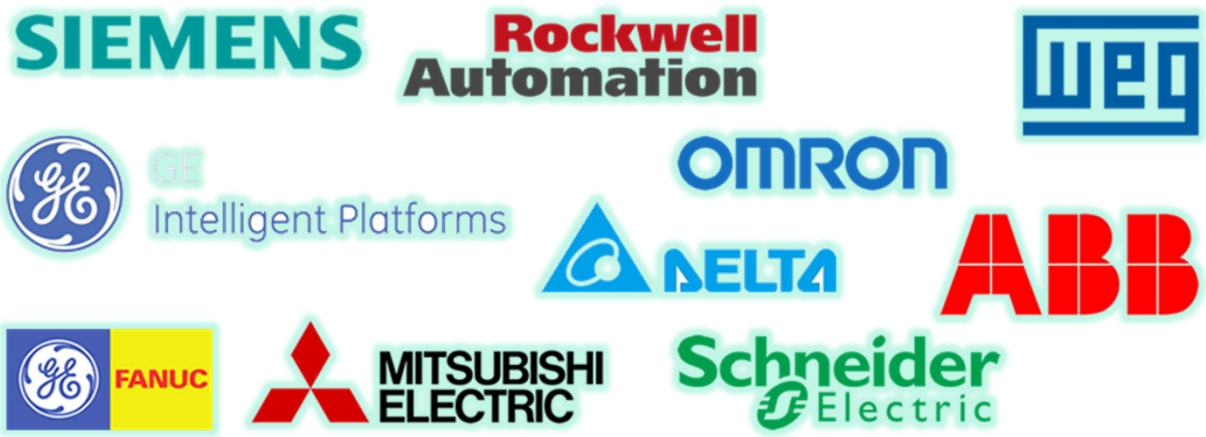
O controlador lógico programável (CLP) é o componente principal de quase todos os projetos de automação industrial. É um computador industrial com capacidade de executar programas criados com a finalidade de automatizar algum processo industrial. Um exemplo de um processo deste tipo seria uma esteira onde são transportados frascos de perfumes vazios e em algum ponto do transporte o frasco deve parar e uma válvula deve ser acionada por um tempo pré-definido para que o frasco seja preenchido de perfume.

Parece complicado, mas um CLP consegue realizar esta tarefa com relativa facilidade. Esse tipo de automação tem um valor muito grande para a indústria já que seria uma tarefa repetitiva para ser realizada por humanos exigindo vários períodos de descanso onde a produção pararia. Com a automatização do processo, a produção não para e o custo de implementação é rapidamente pago pela alta produtividade.

O modelo mais simples de CLP possui apenas a CPU e uma interface de entradas e saídas. Por meio das entradas o CLP recebe sinais de sensores da máquina. Por meio das saídas o CLP atua em algum dispositivo da máquina, por exemplo, válvulas pneumáticas. Por padrão, os sinais de entradas e saídas em qualquer máquina tem tensão (voltagem) 0V e 24V. Normalmente quando a tensão está em 24V o sinal de entrada ou saída está ligado (atuado).

Existem várias outras funções e características que um CLP mais completo possuem, como por exemplo: controle de eixo, gerenciamento de segurança, supervisório, interface de comunicação com o usuário (IHM), controle de variáveis analógicas (temperatura, pressão, vazão).

O projeto que vamos mostrar aqui será bem mais simples com apenas algumas entradas e saídas. Mas não é porque o Arduino não seja capaz de executar as tarefas mais complexas. É simplesmente para que o leitor possa aprender de forma gradual e robusta. A flexibilidade que a ferramenta de programação do Arduino proporciona, permite a criação de controles industriais que não ficam devendo nada frente a grandes marcas.



03

CONCEPÇÃO DO PROJETO CLP

Vamos dividir a concepção do projeto em duas partes. Na primeira parte vamos estabelecer as características do hardware necessário para construção do nosso CLP. Na segunda parte vamos definir as funções do software do CLP.

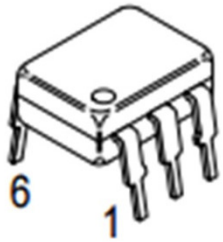
HARDWARE

Vamos usar o Arduino UNO como base para nosso projeto. Esta placa possui 14 pinos de entradas e saídas digitais e 6 pinos de entradas analógicas, além de porta serial RS232 e I2C. De todos estes recursos, vamos neste projeto usar somente 8 pinos de sinais digitais, 4 como saídas e 4 como entradas.

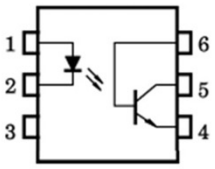
Além da CPU (Arduino) vamos precisar da interface de entradas e saídas (shield). O circuito de interface transforma a tensão padrão de máquina que iremos automatizar para o nível de tensão da placa do Arduino. A tensão padrão de máquinas é 0V e 24V. O nível de tensão da placa do Arduino é 0V e 5V.

Em um projeto de automação industrial o ideal é que a interface de entradas e saídas seja isolada galvanicamente. Isto é, a parte interna do CLP deve estar isolada eletricamente do circuito da máquina. Isso é importante porque a máquina gera muitos ruídos elétricos que podem interferir no processamento do programa e alterar seu funcionamento. Com o isolamento galvânico entre o circuito interno do CLP e o circuito da máquina, minimizamos este efeito.

O isolamento das entradas e saídas pode ser feito por acoplador-ótico. Estes componentes são formados por um par "LED-FOTOTRANSISTOR" de forma que o sinal é transmitido por luz dentro do componente. A imagem abaixo mostra o esquema dentro de um acoplador-ótico do tipo 4N25.



PIN CONFIGURATIONS (Top view)



1 : ANODE
2 : CATHODE
3 : N.C.
4 : EMITTER
5 : COLLECTOR
6 : BASE

Neste ponto o leitor terá que montar um circuito com acoplador-ótico fazendo o nivelamento de tensão para o Arduino. Eu não conheço nenhum shield para Arduino que tenha este circuito pronto. Na próxima sessão eu vou passar o esquema eletrônico para esta montagem. Felizmente um circuito bem simples é suficiente para funcionar como interface de entradas e saídas e permite que o leitor faça a montagem proposta sem muita dificuldade.

Para que nosso CLP fique completo e autônomo vamos precisar de uma fonte de alimentação para o Arduino. Para este fim podemos usar qualquer fonte de alimentação entre 9 e 15 V.

SOFTWARE

Podemos dividir o software de nosso CLP em 2 partes. Uma delas vamos chamar de básico. O básico é o esqueleto do programa que vai definir e configurar os elementos básicos do CLP: entradas, saídas, variáveis, temporizadores. O básico vai ser sempre igual em nosso CLP independentemente da aplicação.

A aplicação é a segunda parte do software e é onde é feita a lógica de controle da máquina ou processo que iremos automatizar.

Dentro da parte do software que chamamos de básico, iremos definir também o “scan” do CLP.

Este é um conceito muito importante em automação industrial. O “scan” do CLP define um intervalo de tempo para o CLP executar as rotinas de controle. É como se fosse um gerente de uma empresa que deve circular pelos departamentos de hora em hora para ver se tudo está sendo feito conforme o planejado e eventualmente tomar alguma decisão. Todos sabem que o gerente só passa de hora em hora, então mesmo que uma tarefa tenha sido finalizada logo após o gerente ter passado, o funcionário só terá uma nova tarefa na hora seguinte. Desta forma todas as etapas do processo ficam sincronizadas e o controle fica mais organizado.

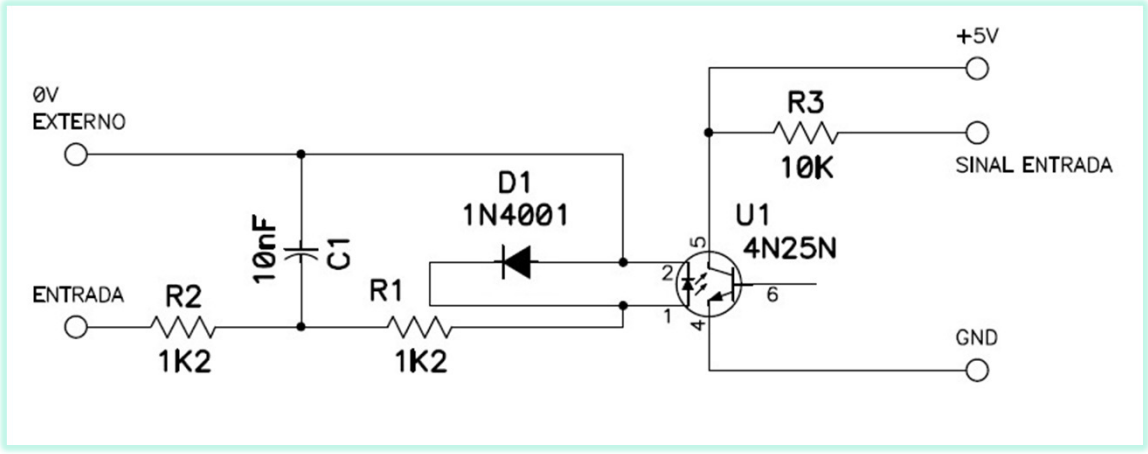
Em nosso CLP vamos definir um tempo de scan de 100 ms, ou seja a cada 100 ms o software vai ler o estado das entradas, atualizar as variáveis e atualizar o estado das saídas.

04

SHIELD DE ENTRADAS E SAÍDAS

Nosso circuito de interface de entradas e saídas poderá ser montado em uma placa padrão. Esta placa que iremos montar será encaixada no Arduino como um shield qualquer. Para isso vamos colocar conectores do tipo barra de pinos nas laterais da placa casando com os conectores do Arduino.

Para o caso da interface de entradas vamos montar um circuito bem simples que terá todas as características necessárias ao nosso projeto. A figura abaixo mostra o esquema eletrônico de uma entrada. O ponto definido como “ENTRADA” é o ponto que recebe o sinal da máquina, lembrando que este sinal tem níveis 0V (desligada) e 24V (ligada). O ponto “0V EXTERNO” vai ser ligado no negativo da fonte de alimentação da máquina. O ponto “+5V” e o “GND” serão ligados na fonte da placa Arduino. E o ponto “SINAL ENTRADA” vai ser ligado em um dos pontos de IO da placa Arduino.

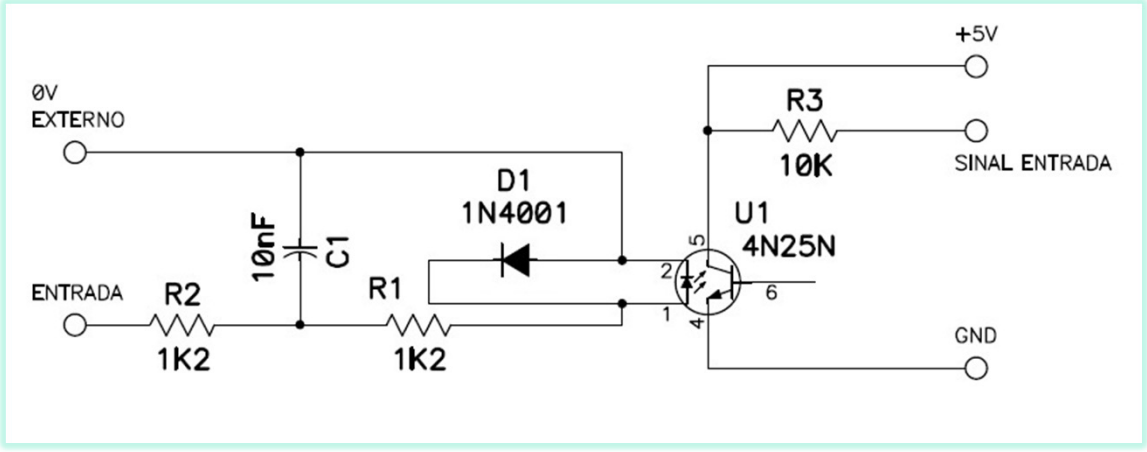


Os resistores R1 e R2 fazem um circuito de filtro juntamente com o capacitor C1. É importante usar 2 resistores de 1/3W porque eles vão ter que dissipar uma potencia razoável já que temos que baixar a tensão de entrada de 24V para o nível de excitação do acoplador-ótico.

O diodo D1 serve para proteger o acoplador-ótico contra eventuais ligações erradas. O pino 6 do acoplador-ótico fica sem ligação mesmo.

E finalmente o resistor R3 serve para o sinal de entrada ficar em 5V quando a entrada está desligada.

Para fazer nosso CLP com 4 entradas vamos ter que montar uma placa com 4 circuitos iguais ao da figura.

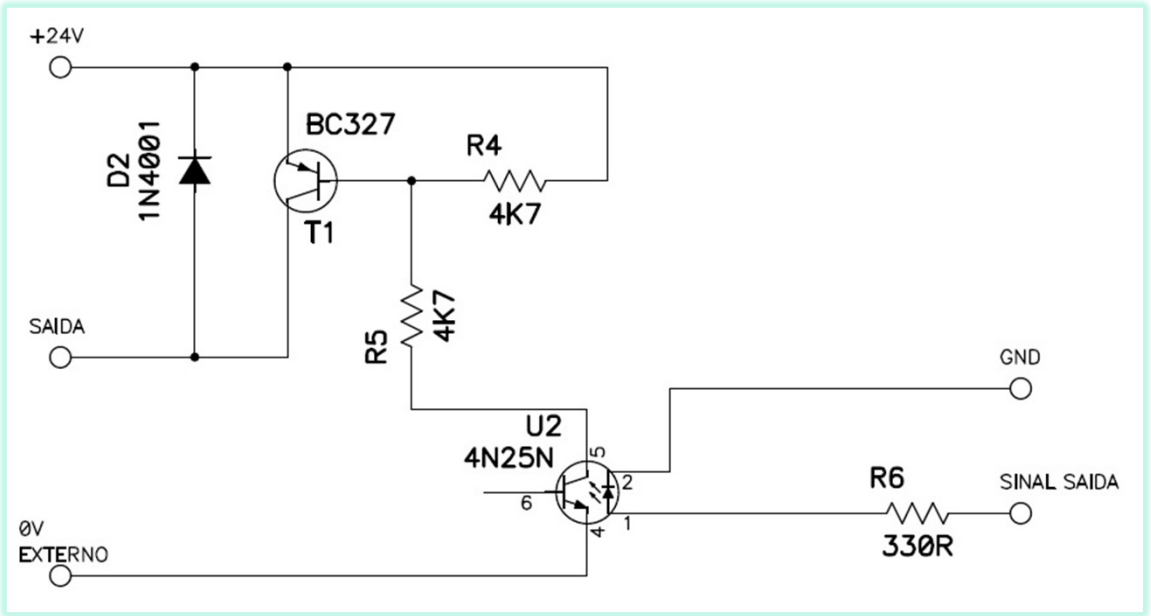


Para o caso da interface de saídas vamos usar um circuito com um transistor para poder ter uma capacidade de fornecimento de corrente maior nas saídas.

Desta forma nosso CLP será capaz de acionar reles, chaves contatoras e válvulas pneumáticas. O esquema da interface de saídas está na figura abaixo.

Novamente temos do lado direito o sinal que comanda a saída e que vai ligado a um pino do Arduino, juntamente com o GND.

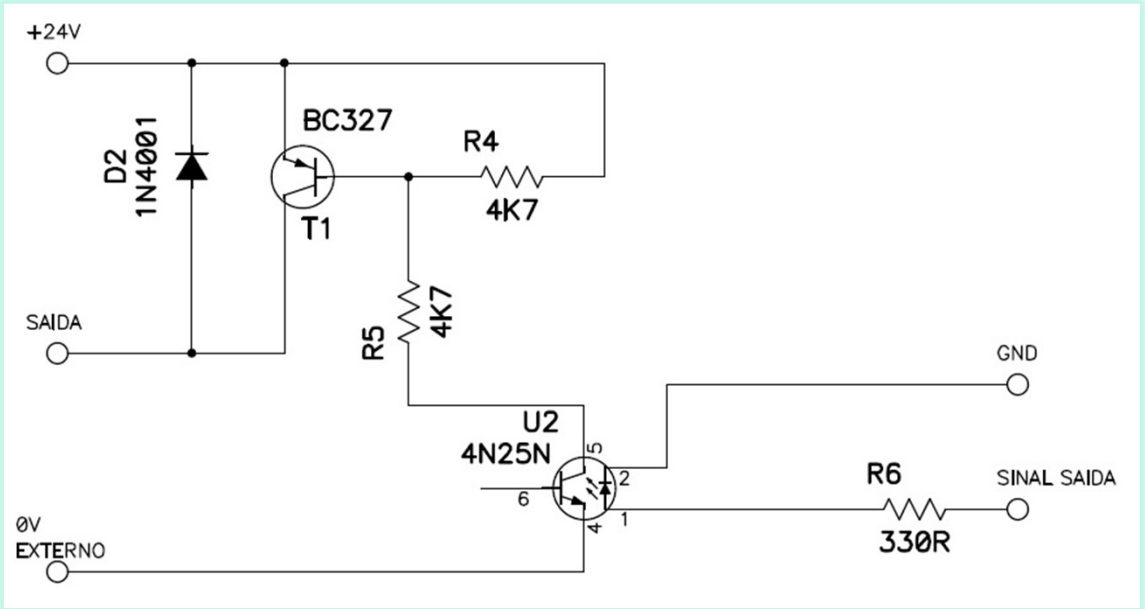
Do lado esquerdo temos a parte que está isolada do circuito interno do CLP e que está conectada a máquina.



O +24V e o 0V externo são ligados a fonte de 24 volts da máquina e o ponto “SAIDA” é o comando que vai atuar para ligar ou desligar algum componente da máquina, por exemplo um motor, uma válvula pneumática, uma bomba hidráulica ou uma lâmpada indicativa de emergência.

Com o transistor na saída podemos atingir uma capacidade de fornecimento de corrente de até 300mA sem problemas.

O diodo D2 serve para suprimir a tensão reversa gerada em cargas indutivas e que pode danificar o circuito.



05

MONTAGEM

Os circuitos propostos aqui para fazer as interfaces de entradas e saídas isoladas são bem simples e podem ser montados em uma placa de circuito impresso de montagem padrão (aquele tipo com vários furinhos para montagem de qualquer circuito).

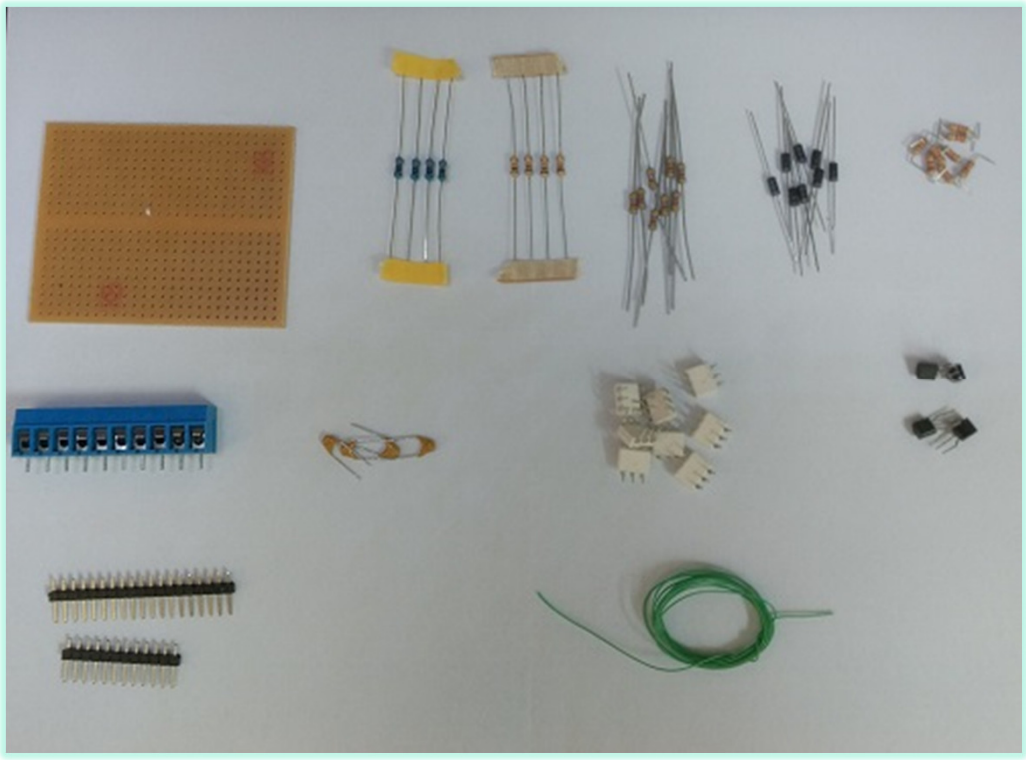
Para o leitor que queira fabricar sua própria placa de circuito, irei disponibilizar no anexo deste livro o desenho das trilhas e furação.

Os pinos do Arduino que iremos utilizar para acionar entradas e saídas são os seguintes:

Entrada 1 – pino 5
Entrada 2 – pino 4
Entrada 3 – pino 3
Entrada 4 – pino 2
Saída 1 – pino 6
Saída 2 – pino 7
Saída 3 – pino 8
Saída 4 – pino 9

A lista completa de material é a seguinte:

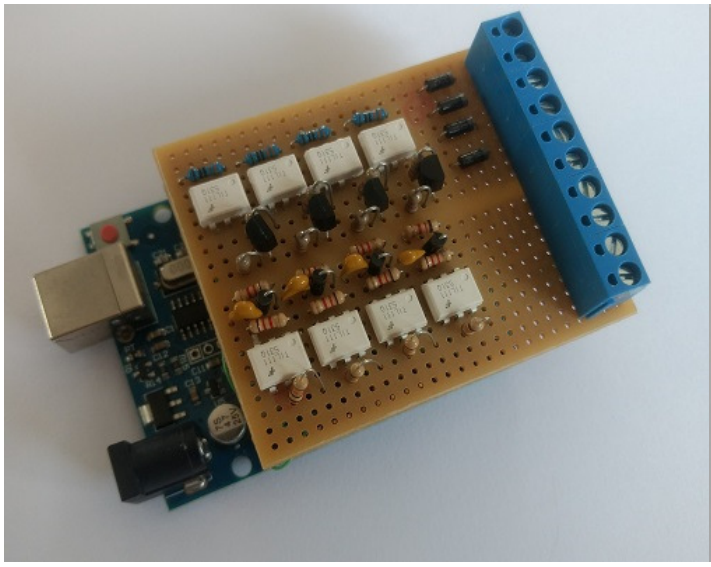
- 1 placa padrão de 70x55mm
- 4 resistores de 10K-5%
- 4 resistores de 330R-5%
- 8 resistores de 4K7-5%
- 8 resistores de 1K2-5%
- 4 capacitores cerâmicos de 10nF
- 4 transistores BC327
- 8 diodos 1N4001
- 8 acopladores óticos 4N25
- Conector de parafusos com 10 vias
- Barra de pinos
- Fio para ligação entre os componentes

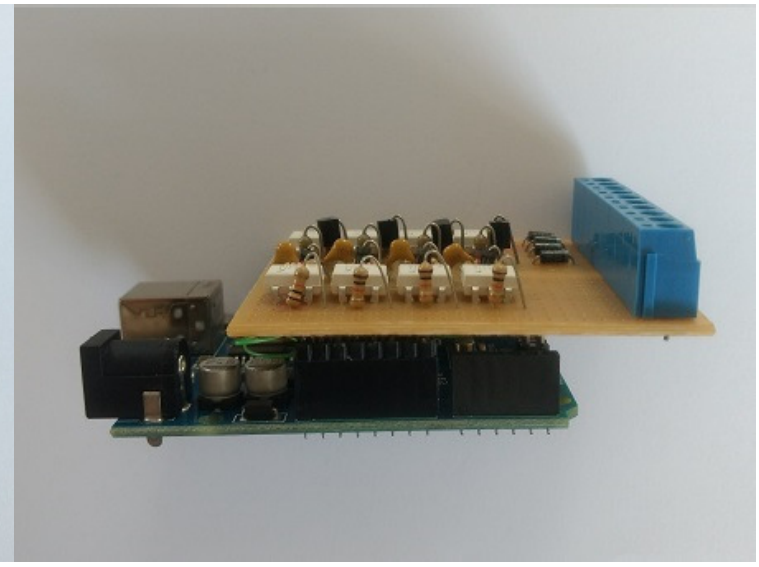


Os cuidados que o leitor deve tomar ao realizar esta montagem se resumem basicamente na hora da soldagem. Nesta hora é melhor fazer o serviço com bastante calma para minimizar a possibilidade de ocorrer problemas de solda fria ou mau contato. Um problema destes pode provocar uma grande perda de tempo durante a fase de desenvolvimento do programa. Muitas vezes ficamos procurando um problema nas linhas do programa e a causa real é um problema na placa eletrônica.

A figura ao lado mostra a placa finalizada e encaixada no Arduino. Vamos usar bornes de parafusos para fazer a interligação da interface com os circuitos da máquina. Desta forma a montagem do CLP na máquina fica muito mais fácil.

São 4 bornes para as entradas, 4 bornes para saídas e 2 bornes para a ligação de +24V e 0V.





06

SIMULADOR DE ENTRADAS E SAÍDAS

Uma ferramenta muito útil para o desenvolvimento de programas de automação industrial é um simulador de entradas e saídas de máquina. Esta ferramenta é composta de uma caixinha com chaves e LEDs.

As chaves são ligadas nas entradas do CLP e os LEDs são ligados nas saídas do CLP.

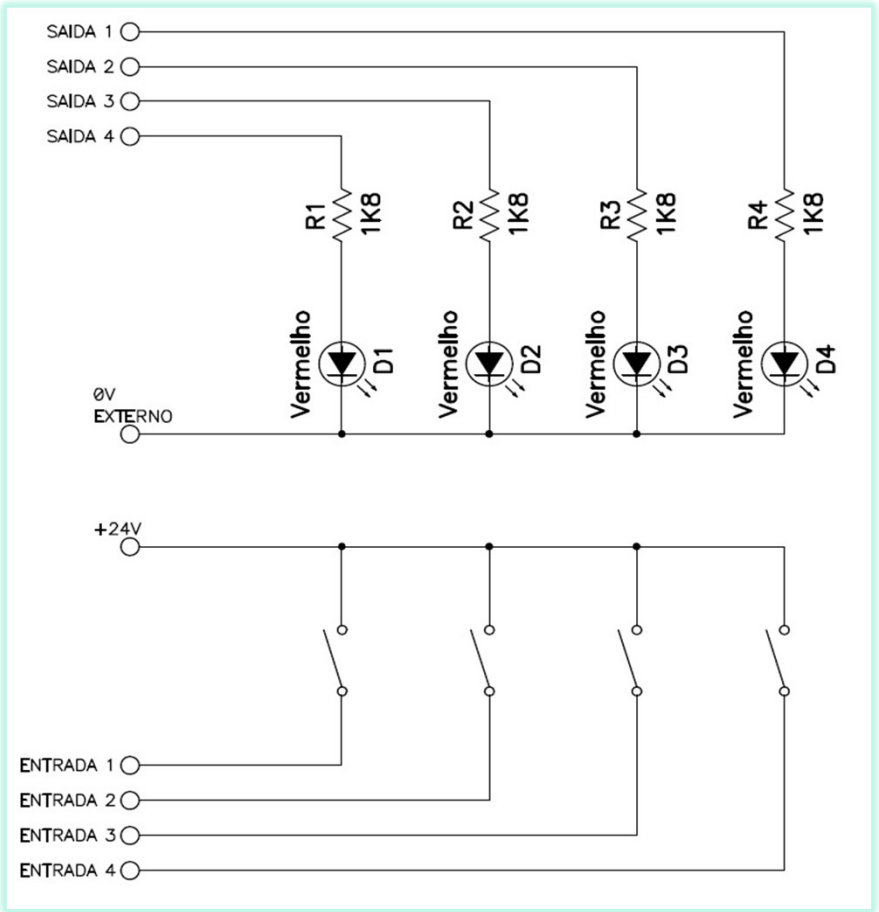
Desta forma podemos simular o acionamento de sensores e dispositivos da máquina durante o desenvolvimento e teste do programa.

Nem sempre temos a máquina à disposição para trabalharmos durante muito tempo. Com o simulador podemos testar o funcionamento do programa e fazer a instalação do CLP na máquina com o programa praticamente pronto.

A figura abaixo mostra o esquema do circuito de um simulador de entradas e saídas.

Neste esquema podemos ver que cada LED está conectado a uma saída do CLP de forma que ao atuarmos uma saída no CLP, o LED correspondente no simulador vai acender.

Para o teste das entradas usamos 4 chaves tipo alavanca de forma que ao ligarmos uma chave, a entrada do CLP correspondente vai receber a tensão de 24V e será atuada.



Para completar o kit de testes do CLP vamos precisar de uma fonte de alimentação de 24V.

Vamos usar uma fonte de alimentação independente da fonte de alimentação do Arduino. Assim teremos os circuitos internos e externos do CLP realmente isolados eletricamente, exatamente como deve ser em uma máquina real.

A figura abaixo mostra o simulador de entradas e saídas finalizado. Vamos usar o simulador para testar o programa desenvolvido para o CLP.

07

PROGRAMAÇÃO

Vamos começar o desenvolvimento da parte de programação do CLP criando um programa base que poderá ser usado em várias outras aplicações.

Neste programa inicial vamos colocar as definições de pinos para os sinais de entradas e saídas do CLP, vamos criar uma base de tempo que será o “scan” de varredura do CLP, vamos colocar as instruções para atuação das saídas e também as instruções para a leitura das entradas do CLP. Esta será a base para qualquer aplicação (programa) de nosso CLP.

A transcrição completa do programa base está logo abaixo. Acompanhe as explicações das funções de cada trecho do programa pelos números de linha.

Logo após o cabeçalho do programa iniciamos a área de "#defines". A diretiva "#define" é usada para que o compilador substitua no corpo do programa toda palavra que for igual a palavra que vem logo após o #define pelo número que vem na sequência.

Em nosso caso, por exemplo, o compilador substitui as palavras ENTRADA1 pelo número 5 no corpo do programa. Uma das razões para se usar o #define é para facilitar uma eventual troca de pinos do Arduino. Se for o caso de uma troca de pino após o programa estar pronto, só precisamos alterar o número que aparece após a palavra da definição, e não no corpo do programa.

Outra razão para o uso de #define é facilitar a compreensão do programa pelo uso de nomes ao invés de números no corpo do programa. Vamos usar palavras em com letras maiúsculas para definição de constantes dentro do programa, como no caso dos #define, e palavras em letras minúsculas para definição de memórias variáveis. Desta forma logo saberemos identificar dentro do programa quando estamos trabalhando com constantes ou variáveis.

As definições das memórias variáveis estão entre as linhas 28 e 36 do programa.

Em seguida temos a função SETUP(), iniciando na linha 39. Esta função é executada no inicio do programa apenas para que sejam feitas as inicializações de variáveis e periféricos do Arduino.

Em nosso programa vamos usar a função SETUP() para inicializar os pinos do Arduino que serão usados como entradas e saídas. Além disso, vamos ler o valor atual da contagem de milissegundos e guardar em nossa variável "tempoUltimoScan" na linha 49.

A variável "tempoUltimoScan" é usada no programa para contar o tempo predefinido de 100 ms. Usamos este tempo para controlar o "scan" do CLP. Mas isso estamos fazendo dentro da função LOOP() a partir na linha 53. A função LOOP é a função principal de qualquer programa do Arduino, e fica em execução continuamente em um looping infinito.

Dentro da função LOOP(), na partir da linha 56, temos uma instrução "if" que verifica quando completa um período de tempo igual a 100ms. Vamos explicar detalhadamente o que acontece no teste de condição da instrução if:

if (millis()-tempoUltimoScan>TEMPO_SCAN)

A expressão que está dentro dos parênteses na instrução “if” é o teste da condição. Se o resultado do teste da condição for “true” (verdadeiro) as instruções que vem a seguir entre as chaves { } serão executadas. Se a condição for “false” (falso) o programa não executa as instruções dentro das chaves.

Chamamos a expressão dentro dos parênteses do “if” de condicional. Para saber o resultado desta expressão condicional primeiramente é executada a função millis(). Esta função retorna um valor que é a atual contagem de milissegundos da CPU. A seguir, deste valor é subtraído o valor de milissegundos da última leitura de tempo que está armazenada na variável tempoUltimoScan. O resultado desta subtração nos revela o tempo em milissegundos transcorrido desde a última execução de scan. Se este tempo transcorrido for maior que o valor na constante TEMPO_SCAN, o resultado da expressão condicional é verdadeiro e as instruções dentro das chaves serão executadas.

Dentro do scopo da instrução “if”, temos 2 linhas. A primeira faz a leitura atual da contagem de milissegundos preparando para a próxima contagem de tempo de “scan”. A segunda linha chama a rotina controleCLP(). É dentro da rotina controleCLP que iremos fazer toda a lógica de controle da automação. Desta forma o programa base do CLP fica sempre igual mudando apenas o conteúdo da rotina controleCLP().

```
/*********************************************************
  ArduinoCLP
  Controlador lógico programável com Arduino

  Possui 4 entradas 24V e 4 saídas 24V

  Scan de 100ms para executar as rotinas de controle

  curso gratuito de programação com Arduino
  www.cursoarduino.com

  criado em 02/2018
  por Jorge Candido
 *********************************************************/

//area para definicao de constantes
#define ENTRADA1  5
#define ENTRADA2  4
#define ENTRADA3  3
#define ENTRADA4  2
#define SAIDA1    6
#define SAIDA2    7
#define SAIDA3    8
#define SAIDA4    9
#define TEMPO SCAN 100

//area para definicao de variaveis
int tempoUltimoScan;
bool estadoEntrada1;
bool estadoEntrada2;
bool estadoEntrada3;
bool estadoEntrada4;
bool estadoSaida1;
bool estadoSaida2;
bool estadoSaida3;
bool estadoSaida4;

// A função de setup é executada uma vez ao ligar a placa ou apos reset
void setup() {
  // Inicializa todos os pinos de IO.
  pinMode(ENTRADA1, INPUT);
  pinMode(ENTRADA2, INPUT);
  pinMode(ENTRADA3, INPUT);
  pinMode(ENTRADA4, INPUT);
  pinMode(SAIDA1, OUTPUT);
  pinMode(SAIDA2, OUTPUT);
  pinMode(SAIDA3, OUTPUT);
  pinMode(SAIDA4, OUTPUT);
  tempoUltimoScan = millis();
}
```

```
51
52 // A funcao loop fica rodando continuamente
53 void loop() {
54
55   //aguarda completar o tempo de Scan para chamar rotina de controle
56   if (millis()-tempoUltimoScan>TEMPO_SCAN)
57   {
58     tempoUltimoScan = millis();
59     controleCLP();
60   }
61 }
62
63 void controleCLP(void)
64 {
65   //atualizacao das variaveis internas de estado das entradas
66   estadoEntrada1= !digitalRead(ENTRADA1);
67   estadoEntrada2= !digitalRead(ENTRADA2);
68   estadoEntrada3= !digitalRead(ENTRADA3);
69   estadoEntrada4= !digitalRead(ENTRADA4);
70
71   //execucao do controle
72   //altera variaveis estadoSaida confome a necessidade do controle
73   //para teste vamos apenas copiar o estadoEntradas para estadoSaidas
74   estadoSaida1= estadoEntrada1;
75   estadoSaida2= estadoEntrada2;
76   estadoSaida3= estadoEntrada3;
77   estadoSaida4= estadoEntrada4;
78
79   //atualiza as saidas conforme memorias internas
80   digitalWrite(SAIDA1,estadoSaida1);
81   digitalWrite(SAIDA2,estadoSaida2);
82   digitalWrite(SAIDA3,estadoSaida3);
83   digitalWrite(SAIDA4,estadoSaida4);
84 }
```

Com este programa base podemos fazer vários projetos de automação somente alterando a função controleCLP().

A rotina controleCLP() que está transcrita acima, serve apenas para teste da nossa placa de entradas e saídas. O que esta rotina faz é copiar o estado das entradas nas saídas do CLP.

Nas linhas 66 a 69 fazemos a leitura das entradas e guardamos o estado de cada uma em uma variável do programa chamada estadoEntrada.

Nas linhas 74 a 77 apenas copiamos as variáveis estadoEntrada para as variáveis estadoSaida. Ao carregar este programa em nosso CLP podemos fazer o teste das entradas e saídas com o simulador de entradas e saídas.

Nas linhas 80 a 83 fazemos a atualização das saídas segundo as variáveis do programa estadoSaida.

Quando conectamos o simulador ao CLP vemos que ao acionar uma das chaves no simulador, o led da saída correspondente acende no simulador. Fazendo todos os leds acenderem e apagarem teremos concluído o teste.

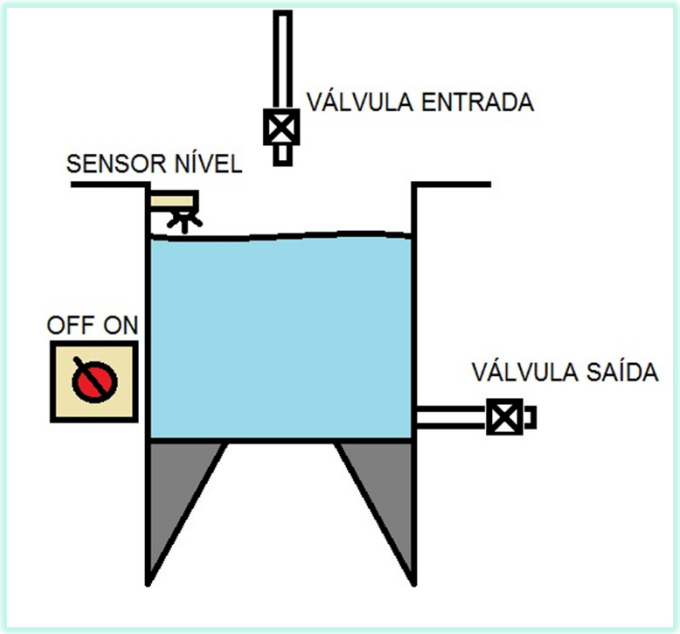
08

APLICAÇÃO: CONTROLE DE NÍVEL

Neste exemplo de aplicação do CLP que construímos eu vou propor um típico controle de nível de líquidos.

Nesta aplicação teremos um tanque de líquido que deve ser mantido sempre com sua capacidade máxima para que a vazão do líquido permaneça com mínima variação possível.

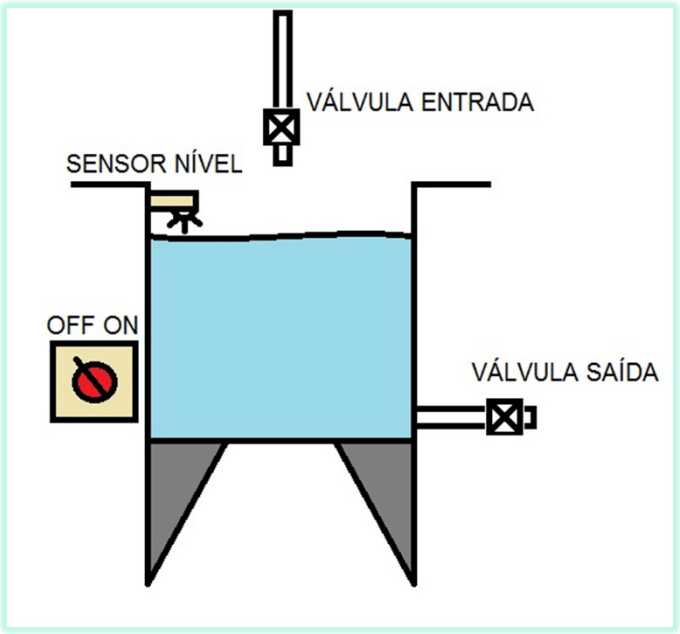
A válvula de saída é controlada por outro processo independente, que pode ser uma outra máquina, um circuito temporizador ou até mesmo uma pessoa que abre e fecha a válvula de saída para usar o líquido.



O que importa para nosso projeto é que a válvula de entrada deve abrir para encher o tanque sempre que o sensor de nível detectar que o nível de líquido no tanque baixou. Mas a válvula de entrada somente pode abrir se a chave ON-OFF estiver na posição ligado. No caso de uma limpeza ou manutenção do tanque, vamos deixar a chave na posição desligado para poder esvaziar o tanque.

Primeiramente vamos definir os sinais de controle do tanque:

- A válvula de entrada será controlada pela saída 1 do CLP.
- O sensor de nível será ligado na entrada 1 do CLP.
- A chave ON-OFF será ligada na entrada 2 do CLP.

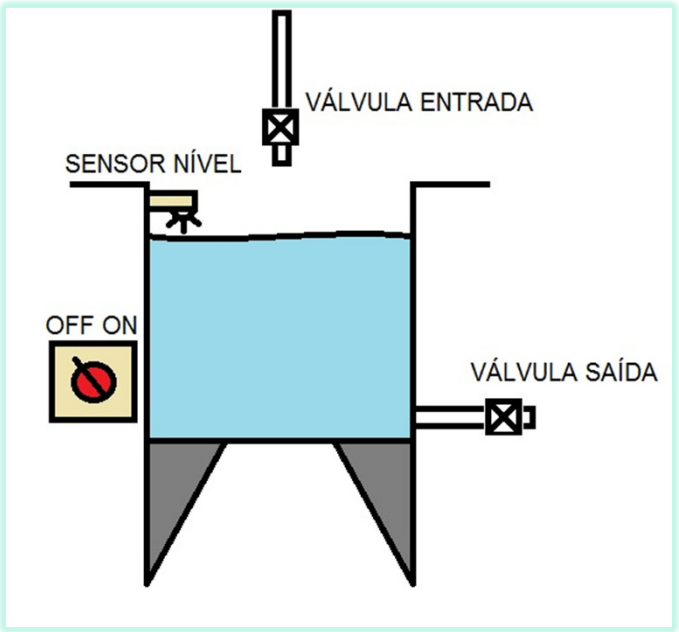


Também temos que descobrir ou definir como vai se comportar cada um desses sinais.

A válvula de entrada vai abrir quando ligarmos a saída 1 do CLP.

O sensor de nível vai enviar tensão para a entrada 1 do CLP quando o líquido atingir o nível desejado. Ou seja, a entrada 1 vai estar ligada quando o nível for atingido e vai estar desligada quando o nível estiver baixo.

A chave ON-OFF vai enviar tensão para a entrada 2 do CLP quando estiver na posição ON. Ou seja, a entrada 2 vai estar ligada quando a chave estiver na posição ON e vai estar desligada quando a chave estiver na posição OFF.

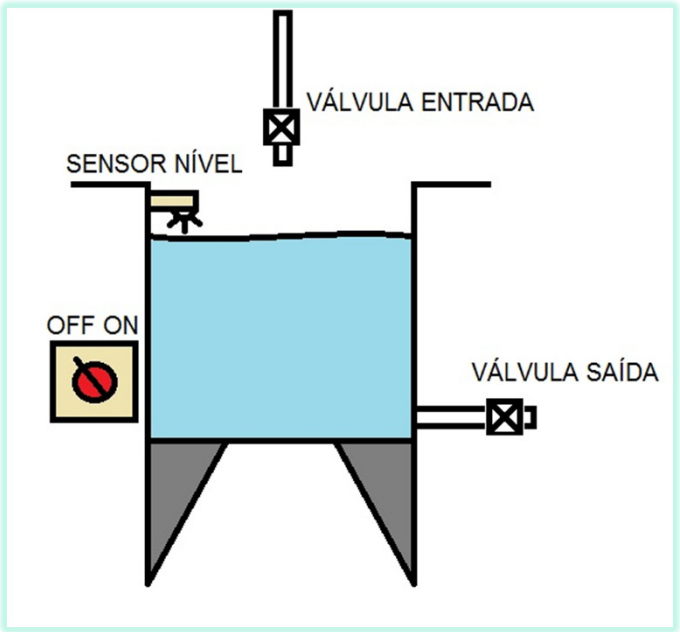


Para fazer o programa de controle de nível do tanque vamos usar o mesmo programa base que já utilizamos e vamos alterar somente a rotina controleCLP().

Na próxima página está transcrito a parte do programa que contem a rotina controleCLP().

Dentro da rotina a primeira parte nas linha 66 e 67, fazemos a leitura das 2 entradas que iremos usar, guardando o estado dessas entradas em memórias.

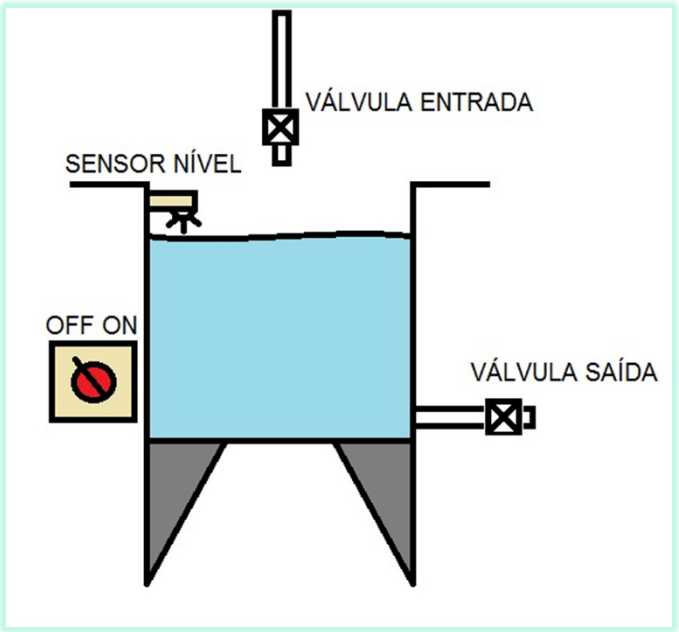
Na parte intermediária da rotina, linhas 70 a 84, temos o controle sendo feito com as instruções “if”.



Na linha 70 é testada a condição da chave ON-OFF. Caso entrada 2 ligada (chave na posição ON), fazemos o teste da condição do sensor de nível na linha 72.

Caso sensor de nível esteja com sinal desligado, deixamos a memória estadoSaida1 em “true”. Caso contrário, deixamos a memória estadoSaida1 em “false”.

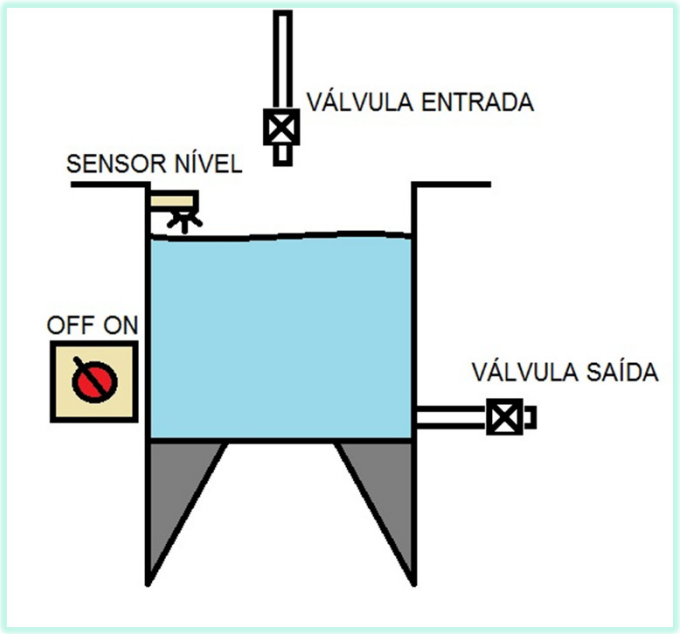
Na linha 81 temos a instrução “else” correspondente da instrução “if” da linha 70. Portanto, caso a condição do estadoEntrada2 seja “false” (chave na posição OFF) deixamos a memória estadoSaida1 em “false”.



Na terceira parte da rotina controleCLP() a partir da linha 87, temos a atualização da saída 1 segundo a memória estadoSaida1.

É muito importante manter esta estrutura de programa para realizar o controle do CLP:

1. Uma primeira parte onde é feita a leitura de todas as entradas utilizadas no controle.
2. Uma parte intermediária onde é construída a lógica do controle.
3. Uma parte final onde todas as saídas utilizadas no controle são atualizadas praticamente ao mesmo tempo.



```
63 void controleCLP(void)
64 {
65   //atualizacao das variaveis internas de estado das entradas
66   estadoEntrada1= !digitalRead(ENTRADA1);
67   estadoEntrada2= !digitalRead(ENTRADA2);
68
69   //execucao do controle
70   if (estadoEntrada2)
71   {
72     if (!estadoEntrada1)
73     {
74       estadoSaida1= true;
75     }
76     else
77     {
78       estadoSaida1= false;
79     }
80   }
81   else
82   {
83     estadoSaida1= false;
84   }
85
86   //atualiza as saidas conforme memorias internas
87   digitalWrite(SAIDA1,estadoSaida1);
88 }
```

09

ANEXO: ARQUIVOS PARA FABRICAÇÃO DO SHIELD

O leitor que tiver alguma experiência em fabricação de placas de circuito impresso poderá aproveitar os desenhos contidos neste anexo para confeccionar seu próprio shield de entradas e saídas isoladas.

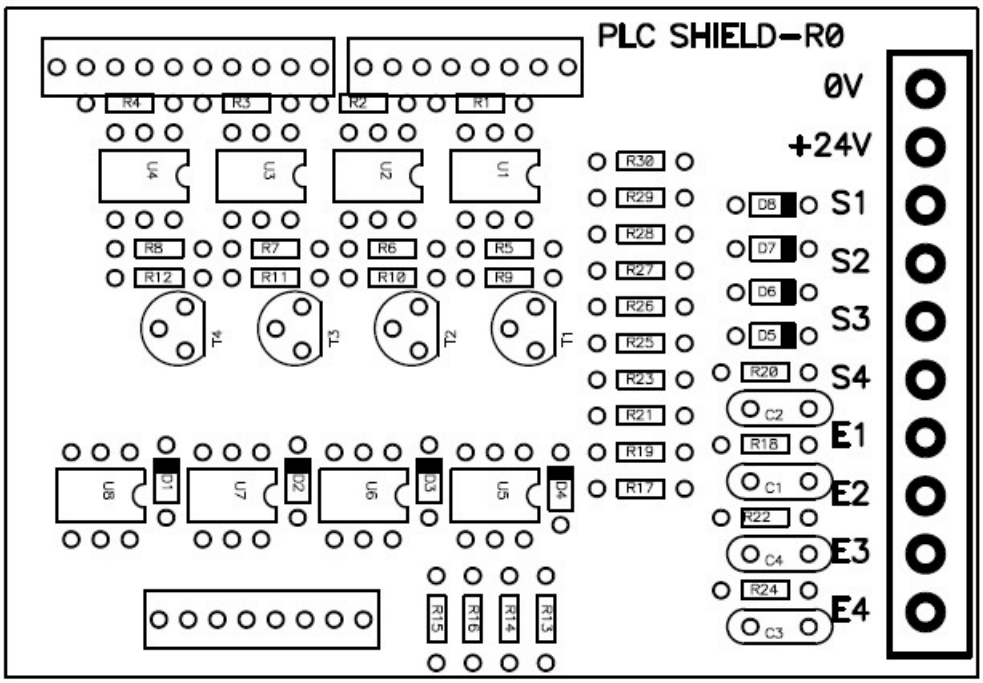
Para que a placa fique nas dimensões corretas o leitor deve imprimir os desenhos mudando a escala para que o desenho fique nas dimensões de 84mm por 58 mm.

O desenho das trilhas está espelhado para que as trilhas fiquem na parte de baixo da placa, enquanto que os componentes serão montados na parte de cima da placa.

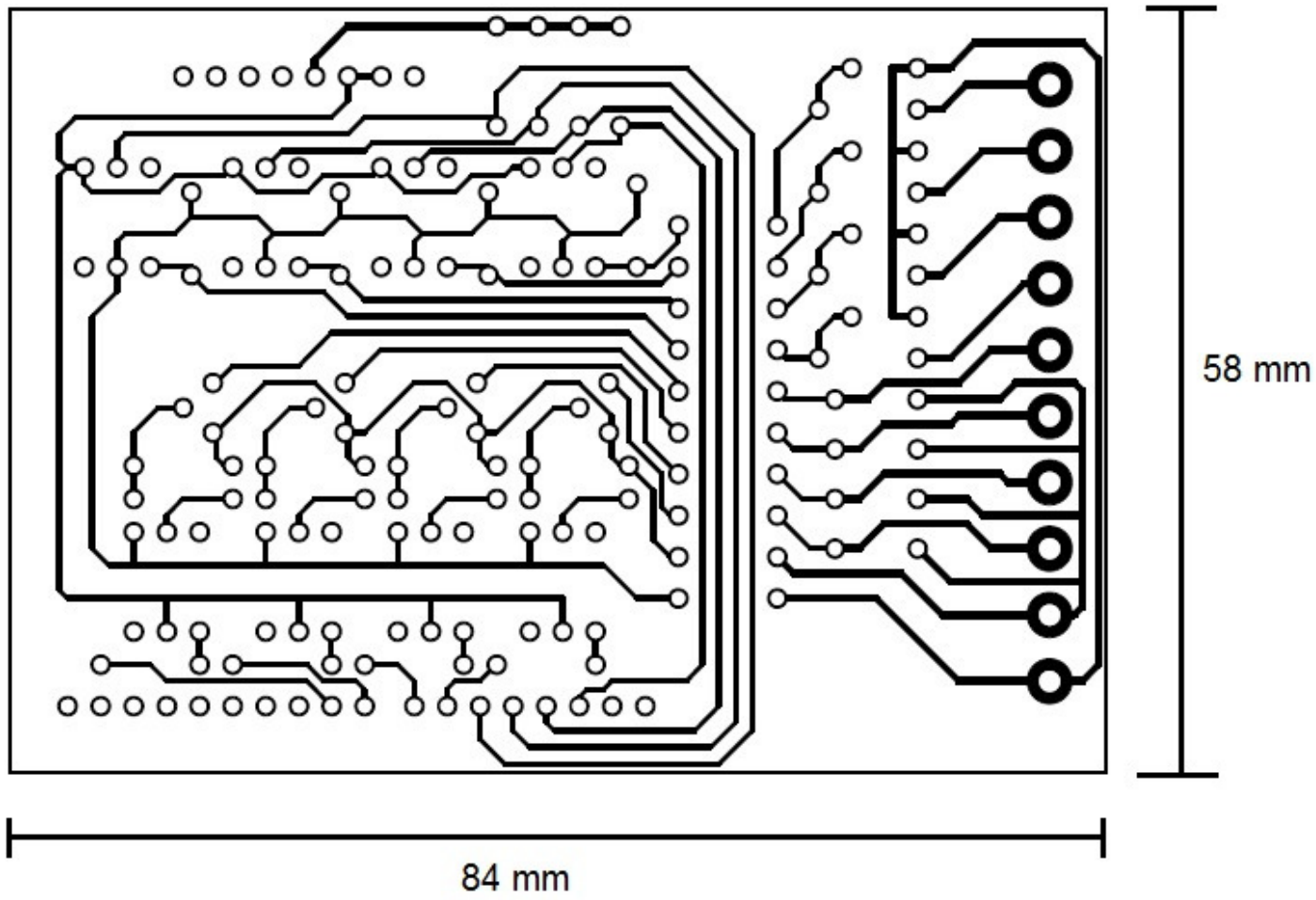
Os conectores tipo barra de pinos devem ser montados no lado contrário ao dos outros componentes. Isso porque esses conectores serão encaixados na placa do Arduino.

LISTA DE MATERIAIS

REFERÊNCIA	CÓDIGOS
R13, R14, R15, R16	Resistores 10K
R1, R2, R3, R4	Resistores 330R
R5, R6, R7, R8, R9, R10, R11, R12	Resistores 4,7K
R17, R18, R19, R20, R21, R22, R23, R24	Resistores 1,2K
R25, R26, R27, R28, R29, R30	Resistores 1R
C1, C2, C3, C4	Capacitores cerâmicos 10nF
T1, T2, T3, T4	Transistores BC327
D1, D2, D3, D4, D5, D6, D7, D8	Diodos 1N4001
U1, U2, U3, U4, U5, U6, U7, U8	Acoplador ótico 4N25



LAYOUT:



SOBRE O AUTOR:

JORGE CÂNDIDO É MESTRE EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO, COM MAIS QUE 30 ANOS DE EXPERIÊNCIA EM PROJETOS DE AUTOMAÇÃO INDUSTRIAL.

ENTUSIASTA DA PLATAFORMA ARDUÍNO.

CRIADOR DO SITE:

www.cursoarduino.com