Prototipagem e Montagem de Placa de Circuito Impresso

Aula 06 - Roteiro Prático nº 03 - Desenvolvimento da PCI: ARES

Apresentação

Esta aula prática foi preparada com o Proteus 7.8 SP2, opções, ícones ou características podem ser diferentes de acordo com a versão.

Objetivos

Ao final desta aula, você será capaz de:

- Simular circuitos eletrônicos no Proteus, inclusive circuitos digitais.
- Criar o l*ayout do circuito eletrônico.*
- Criar o padrão de arquivos comerciais usados na produção de circuitos eletrônicos, os chamados *Gerber*.

Proteus (ISIS)

Na aula anterior, criamos um circuito básico de semáforo. No nosso circuito tínhamos o microcontrolador, alguns resistores e diodos LED. Vocês devem estar lembrados que o circuito ficou igual ao mostrado na **Figura 1**.

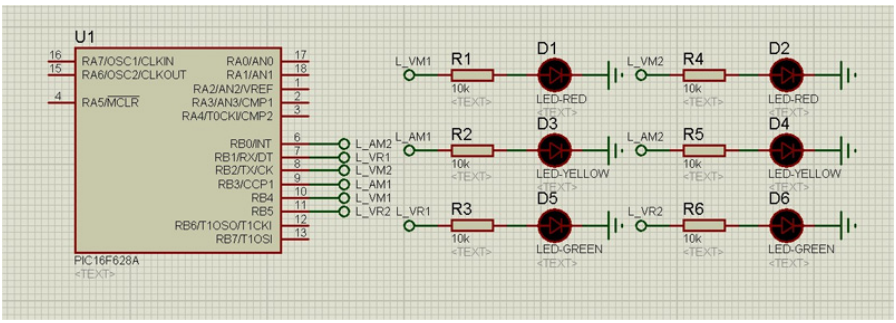

Figura 1 - Tela inicial do Proteus

Nesta aula, vamos adicionar mais 4 chaves DIP, também conhecido como *DIP-Switch*. Para adicionar esse componente ao projeto, basta seguir os passos da aula anterior, se for necessário faça uma revisão.

Dica

Existem diversos tipos de chave *DIP-Switch* disponíveis na biblioteca de componentes do Proteus, como podemos observar na categoria *Switches* e *Relays*, ou ainda, na subcategoria *Switches*. Clique em algumas delas e observe o esquemático de cada uma.

O componente que devemos adicionar é o *DIPSW_4* junto com ele vamos adicionar alguns resistores, lembrem que já fizemos isso na aula anterior. O resistor que vamos usar é chamado de *pull-up*, ele serve para manter a tensão fixa naquele ponto. Caso esses resistores não sejam usados, tal tensão pode sofrer variações.

No nosso circuito, as chaves são ligadas aos resistores que, por sua vez, se conectam diretamente ao VCC do circuito. Dessa forma, conseguimos garantir que no pino de entrada do microcontrolador não ocorram variações de tensão, as quais podem ser provocadas por campos magnéticos ou cargas eletrostáticas.

O valor de um resistor de *pull-up* deve ser bastante alto, entre 1Kohm e 10Kohm. Dessa forma, garantimos que a corrente será baixa em cima desse componente, assim, nossa placa também irá consumir pouca corrente.

O funcionamento do circuito é simples, quando acionamos a chave *DIPSW_4,* o GND fica em comum com o pino do microcontrolador, fazendo com que o sinal daquele pino mude de estado: nível alto para nível baixo. Quando voltamos com a chave para a posição inicial, o resistor que está ligado em VCC faz com que o estado do pino do microcontrolador mude mais uma vez: nível baixo para nível alto. Um exemplo desse tipo de circuito é mostrado na **Figura 2**.

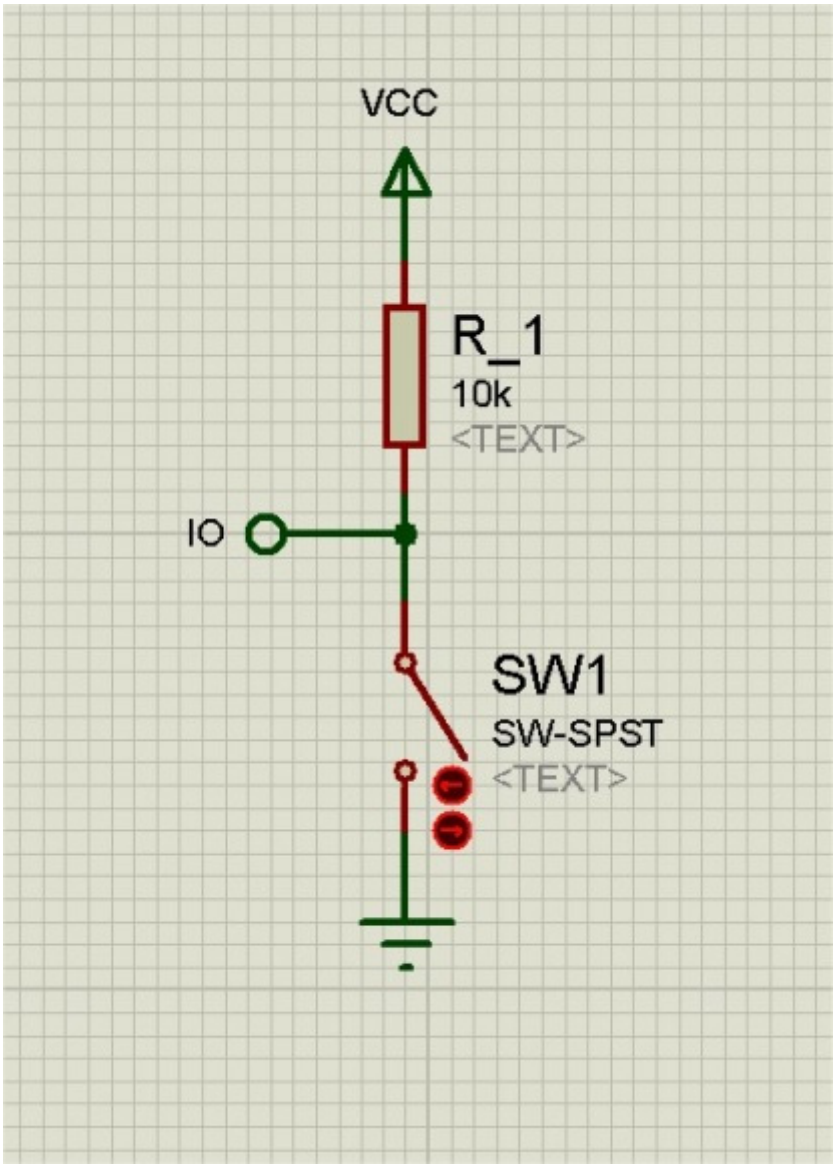

Figura 2 - Exemplo de um circuito com resistor de *pull-up*

Agora que sabemos a função de um resistor de *pull-up,* vamos ligar a nossa chave *DIPSW_4* com o microcontrolador. Para isso, construa, no circuito que foi implementado na aula passada, o módulo mostrado na **Figura 3**. O objetivo desse *DIP switch* é utilizá-lo para modificações lógicas no comportamento dos semáforos. Por exemplo, a depender da palavra configurada nele, os semáforos obedecerão a um comportamento específico. Portanto, há que se fazer modificações no código para atingir o fim desejado.

Por enquanto, não nos preocuparemos em modificar o código desenvolvido em aulas anteriores.

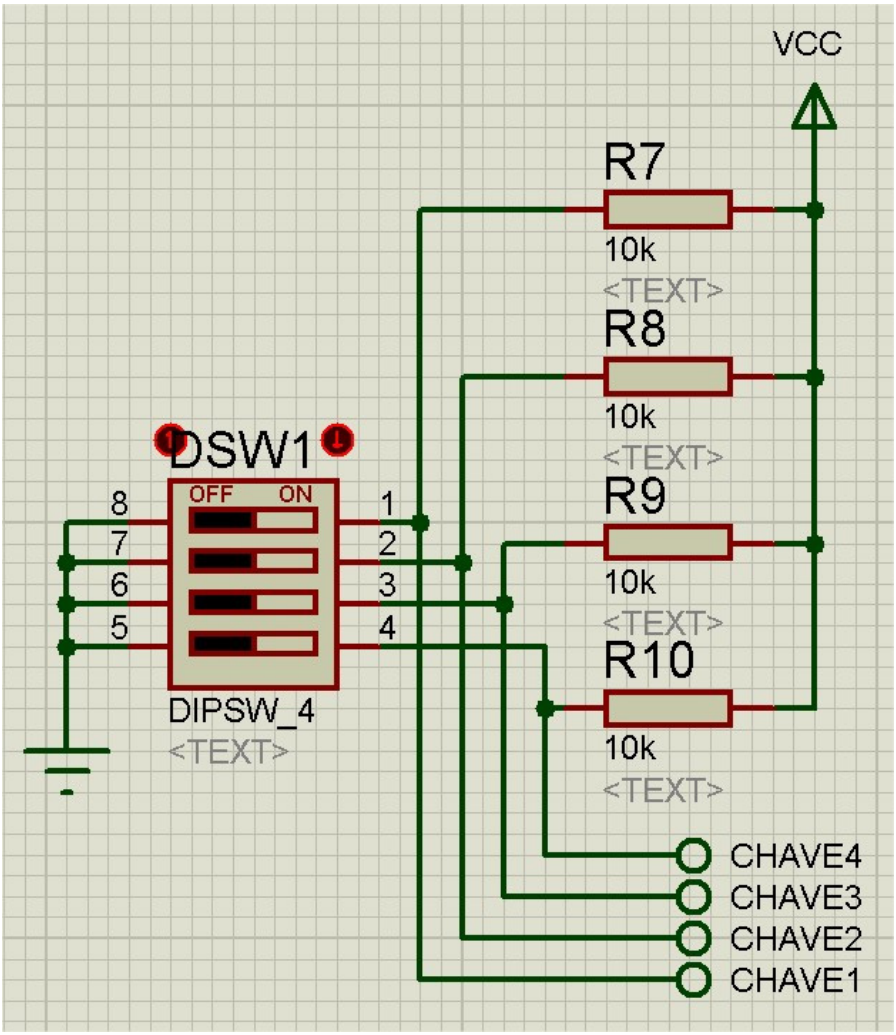

Figura 3 - Circuito com *DIPSW_4*, resistor de *pull_up* e chaves de I/O

Dica

O termo I/O é usado na eletrônica para determinar entrada ou saída, do inglês *input/output*, pode ser que também seja encontrado IO, IN/OUT ou 1/0, mas todos significam a mesma coisa: indicam que naquele ponto os sinais podem ser de entrada ou saída.

Observem que no circuito da **Figura 3** conectamos os pinos 5, 6, 7 e 8 do *DIPSW_4* ao GND e os pinos 1, 2, 3 e 4 aos nomes: CHAVE1, CHAVE2, CHAVE3 e CHAVE4, respectivamente. Além disso, adicionamos nossos resistores de *pull-up* ao VCC, mas o componente VCC é novo e até o momento não tínhamos trabalhado com ele. Vocês devem lembrar de como fazer para adicionar o componente GROUND. deve seguir o mesmo

procedimento como fosse adicionar o GROUND, mas escolher a opção POWER no lugar de GROUND. Caso não se lembre, retorne a aula anterior e revise o assunto. Para efeito de organização do circuito, renomeie o *terminal* inserido com a *String* VCC.

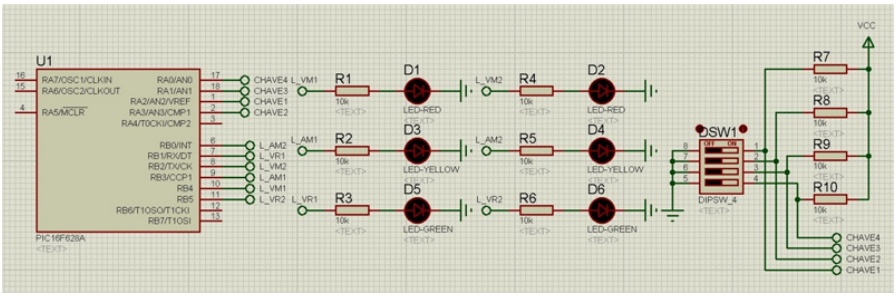

Figura 4 - Circuito com as ligações de I/O

Observe que foram adicionados terminais DEFAULT aos pinos RA0, RA1, RA2 e RA3 do microcontrolador e foram nomeados, respectivamente, de CHAVE4, CHAVE3, CHAVE1, CHAVE2. Dessa maneira, relembrando, nós conectamos, um a um, os terminais do microcontrolador aos terminais do *DIP-Switch* que apresentam a mesma denominação.

Simulação

A partir de agora, vamos testar o código e o circuito que criamos. O objetivo é validá-los antes de partir para a criação do layout da placa. Nesse ponto, teremos a criação física do circuito.

Para a validação, precisamos do código no formato HEX criado na aula 04 (roteiro prático 01) após uma compilação sem erros. Devemos fazer o seguinte procedimento: clicar com o botão direito no microcontrolador e escolher *Edit Properties*. A janela mostrada na **Figura 5** será aberta.

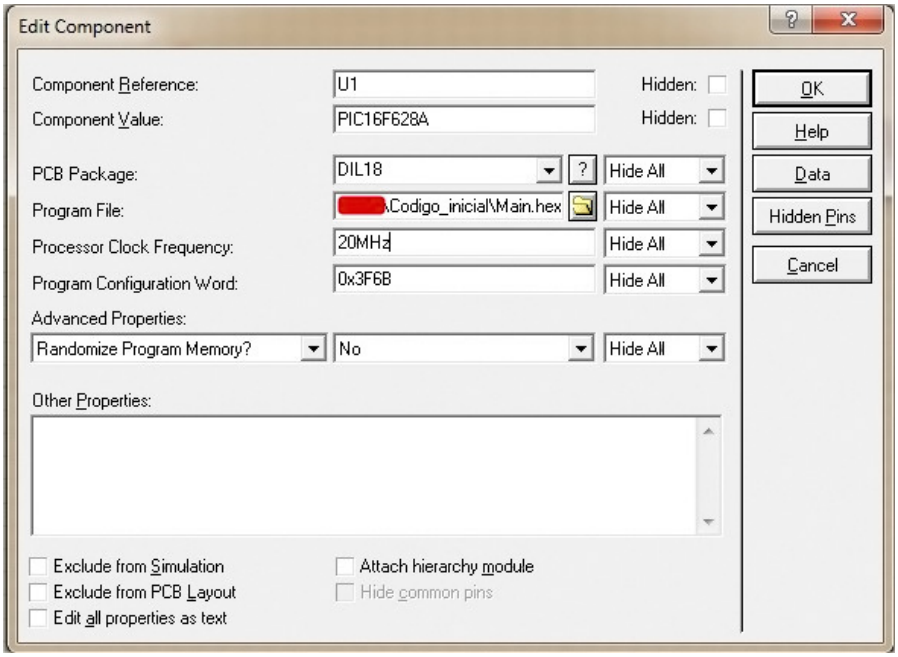

Figura 5 - Janela mostrando as propriedades iniciais do microcontrolador

Inicialmente, devemos indicar onde está nosso arquivo HEX criado anteriormente. Localize o arquivo HEX após clicar no ícone cujo desenho é uma pasta da opção *Program File*. Em seguida, devemos mudar a frequência do *clock* do processador, também chamado de relógio do microcontrolador. Ele deve ser igual ao que colocamos no nosso código: "#use delay(clock=20M)" Então, devemos configurar o *Processor Clock Frequency* para 20MHz, por fim, basta clicar no botão OK. A **Figura 6** mostra a configuração final.

Observação

Você já viu nesse curso que o clock do processador é o que determina a velocidade com que o microcontrolador irá trabalhar. Um exemplo é colocar o clock para $10MHz$, isso quer dizer que o microcontrolador é capaz de realizar 10 milhões de instruções por segundo. Lembre-se de que $1Hz$ quer dizer um ciclo por segundo.

$$10MHz = 10000000Hz = 0,0000001S = 100nS$$

Cada operação leva $100nS$ para $1S$, temos:

$$\frac{1S}{100nS} = 10M$$

Logo, serão 10 milhões de instruções.

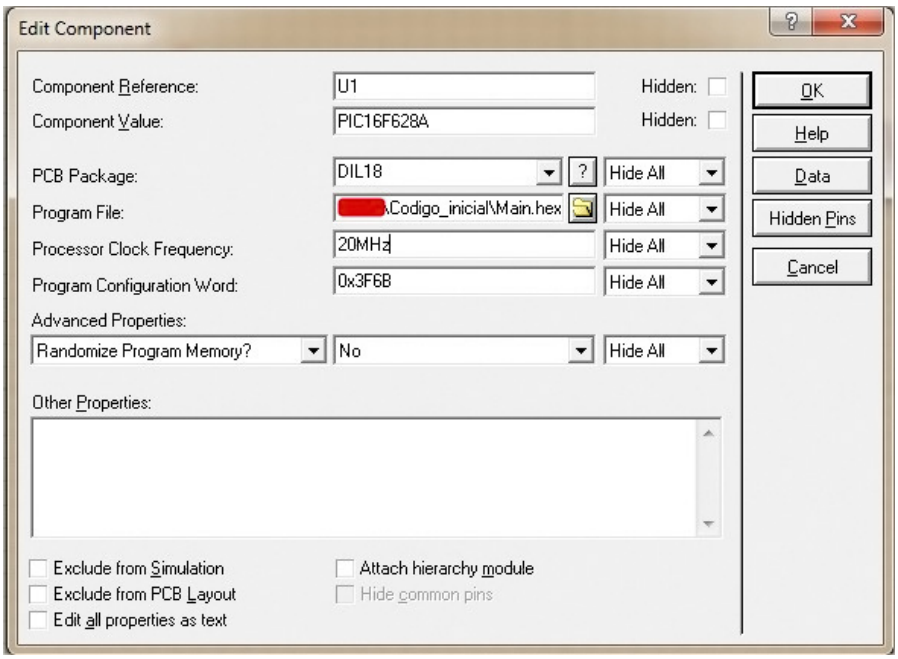

Figura 6 - Janela mostrando as propriedades do microcontrolador após a configuração

A partir de agora, podemos iniciar a simulação, para isso, vamos procurar a barra de simulação, ela fica na parte inferior do ISIS. Nela, vamos fazer uso dos botões Iniciar e Parar, a **Figura 7** mostra essa barra.

Figura 7 - Barra de simulação

Observação

A operação de simulação vai testar o circuito antes de ele ser finalizado, se o circuito tiver erros, o Proteus irá abrir uma janela informando o erro, se isso ocorrer mantenha calma, pois o tutor e monitores estarão presentes para auxiliá-lo.

Se o circuito estiver correto, a simulação irá começar, mas observe que nada acontece. O motivo é porque até agora usamos os valores de resistência considerados padrão no ISIS, esses valores são de 10K. Devido a isso, os LED não irão acender, pois a corrente é muito baixa, logo, temos que parar a simulação e mudar os valores dos resistores (somente os 6 resistores dos LED do semáforo) para 220R, basta clicar sobre o valor e mudá-lo.

O motivo de deixar os resistores com o valor padrão é mostrar que a simulação não funciona, pois se na prática os resistores tivessem esses valores altos, os LED também não irão acender, nosso esquemático deve ficar como mostrado na **Figura 8**.

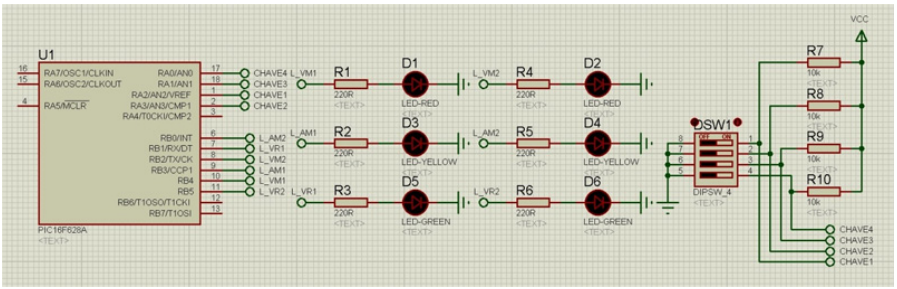

Figura 8 - Esquemático com os valores dos resistores atualizados

Enfim, podemos voltar à simulação, vejam o que acontece! Se o circuito se comportar como mostrado na **Figura 9**, parabéns! Você conseguiu, se não tente novamente, refazendo os passos.

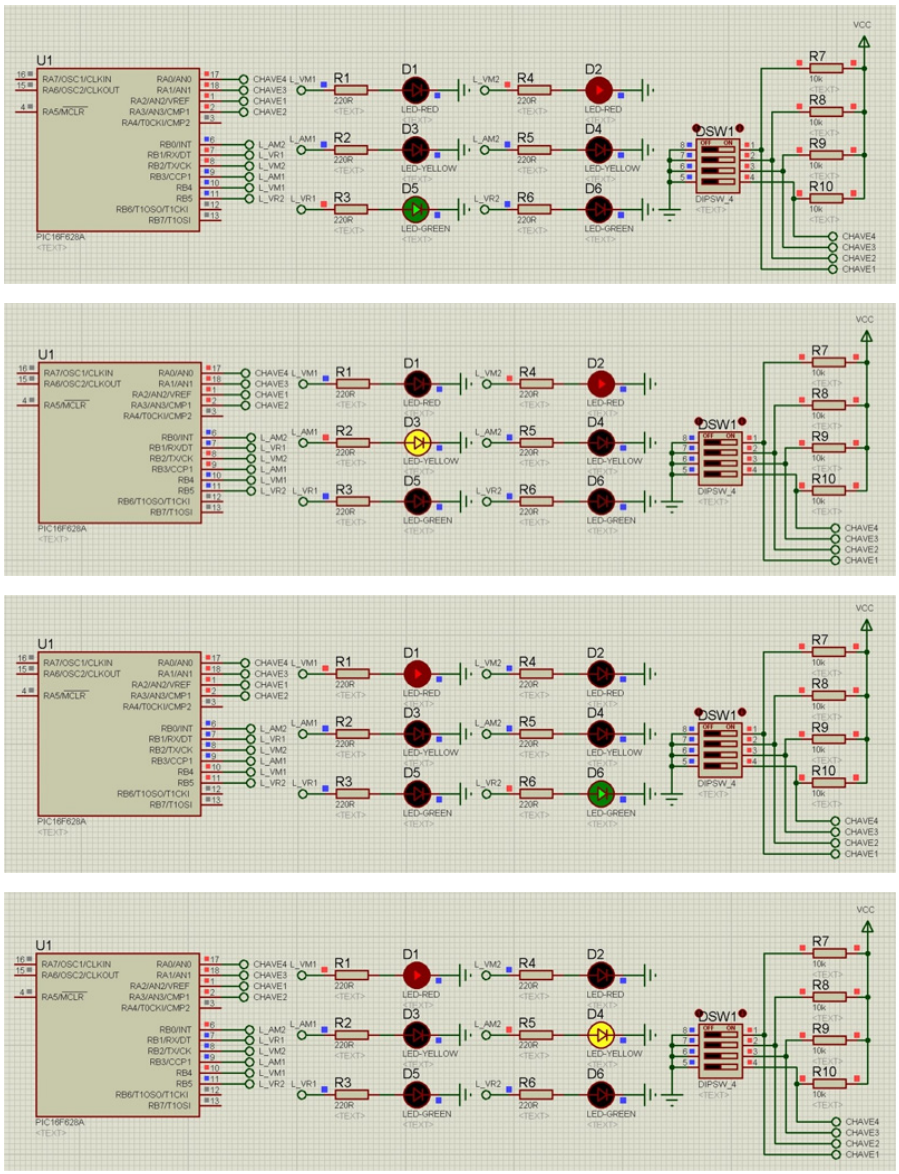

Figura 9 - Simulação do circuito com o código inicial

Adicionando os Outros Circuitos

Vamos agora adicionar os outros circuitos essenciais para nossa placa. Eles não foram necessários até esse ponto, pois na simulação não é obrigatório que estejam presentes. São eles: circuito do relógio, circuito do gravador, circuito do *reset* e circuito da fonte.

Circuito do Relógio

Consiste basicamente em um cristal e de alguns capacitores de filtragem. Para saber como esse circuito será montado, devemos analisar o *datasheet* do microcontrolador. Ele irá dizer qual cristal deve ser usado como mostrado na **Figura 10**.

Monte o circuito de forma que fique como mostrado na **Figura 10** Os componentes são CRYSTAL, CERAMIC15P, Terminal DEFAULT e Terminal GND.

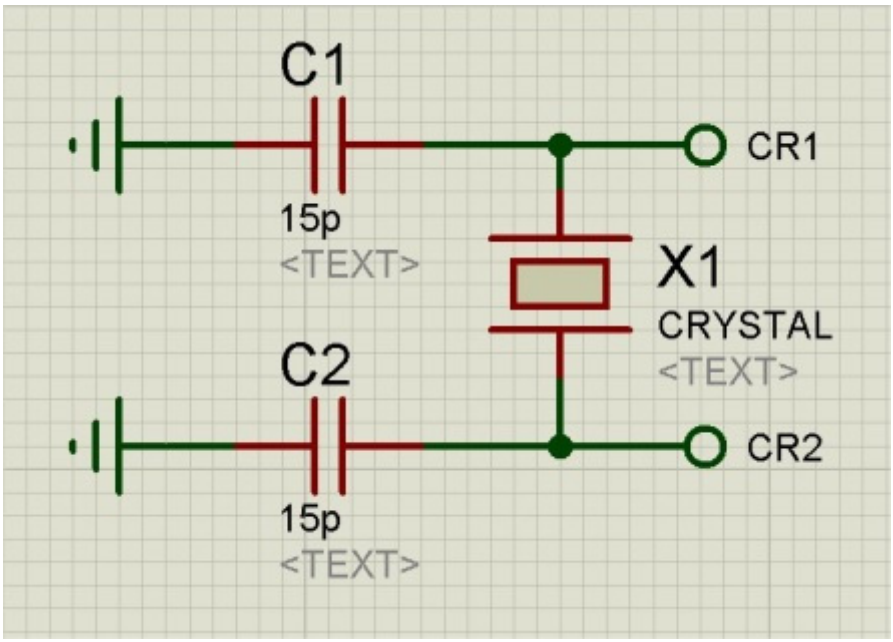

Figura 10 - Circuito do cristal

Para modificar o valor do cristal, basta clicar duas vezes sobre ele. Uma janela será aberta igual à mostrada na **Figura 11**. Na opção *Frequency* coloque o valor para 20MHz.

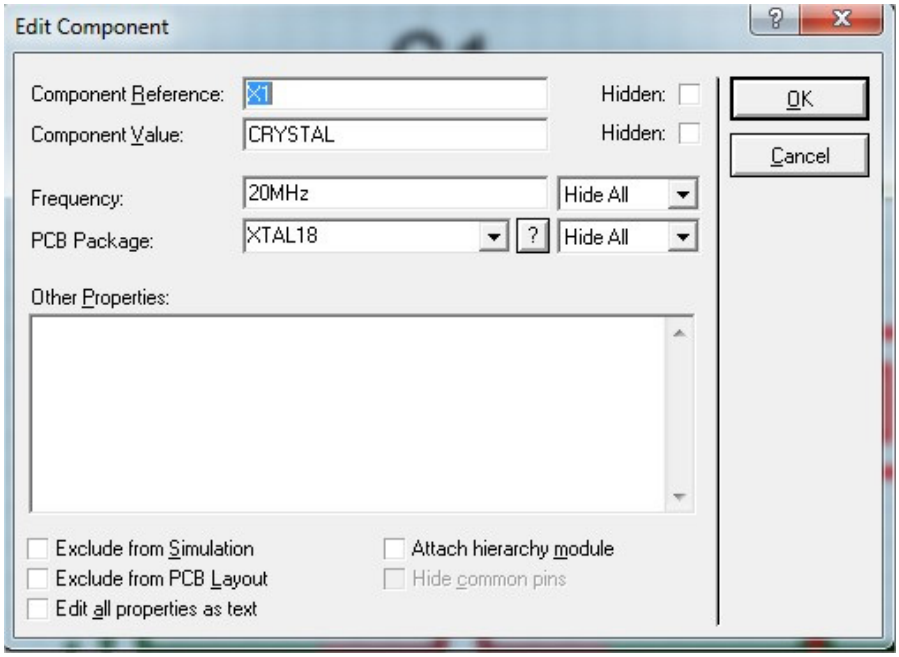

Figura 11 - Propriedades do componente CRYSTAL

Observação

Se o seu circuito for usar um cristal com valor diferente, fique atento para modificá-lo, pois quando a placa for ser soldada por uma empresa, o valor do cristal que será usado é o que foi determinado aqui.

Circuito do Gravador

O próximo circuito a ser montado é o do gravador, ele consiste apenas no Conector (CONN-SIL5) de terminais. Vamos montá-lo conforme a **Figura 12**. Preste atenção para não inverter a ordem dos pinos, se não o componente irá ficar invertido no ARES.

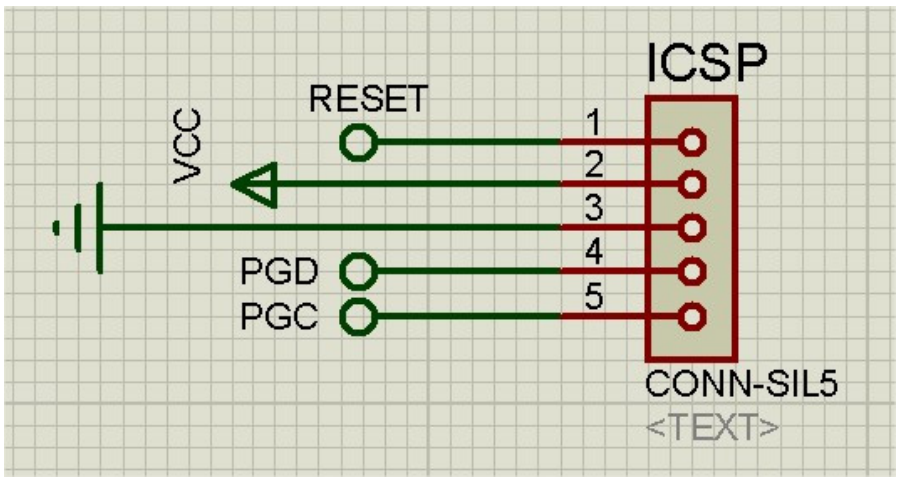

Figura 12 - Circuito do gravador

Esse circuito irá ser útil na hora de gravar o nosso *firmware* no microcontrolador.

Circuito do *reset*

O circuito do *reset* consiste em um resistor de *pull-up*, um botão (BUTTON) e alguns terminais. O resistor é para prevenir *reset* indesejável e o botão serve para reiniciar nossa placa quando for necessário. Observe na **Figura 13** como o circuito deve ficar.

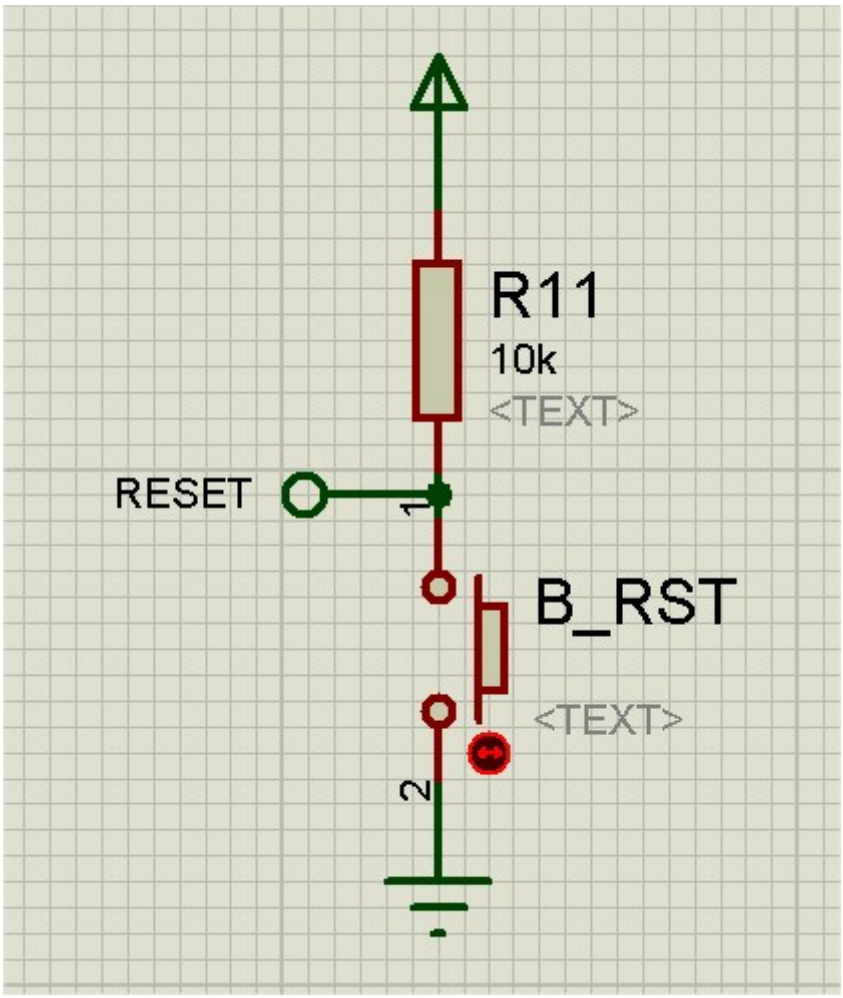

Figura 13 - Circuito do *reset*

Circuito da Fonte

Para algumas pessoas o circuito da fonte é considerado o mais complicado, mas basta dividi-las em módulos e fica fácil de entender. Os módulos da fonte que estamos usando são: proteção contra inversão de polaridade, filtragem na entrada, regulagem, filtragem da saída e status.

A proteção contra inversão de polaridade serve para prevenir que fontes com pinagem diferente do padrão escolhido sejam usadas. A **Figura 14** mostra o padrão de uma fonte com o positivo interno, esse é o modelo usado no nosso projeto. A **Figura 15** mostra uma fonte com o positivo externo. Quem define qual tipo de fonte usar é o projetista, como no mercado temos os dois modelos, devemos usar o módulo de proteção para evitar danos na placa causados pela troca de fontes.

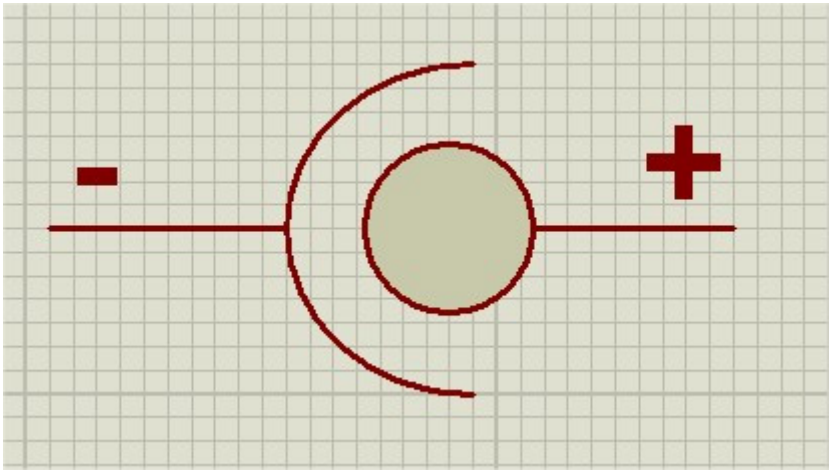

Figura 14 - Padrão com o positivo interno

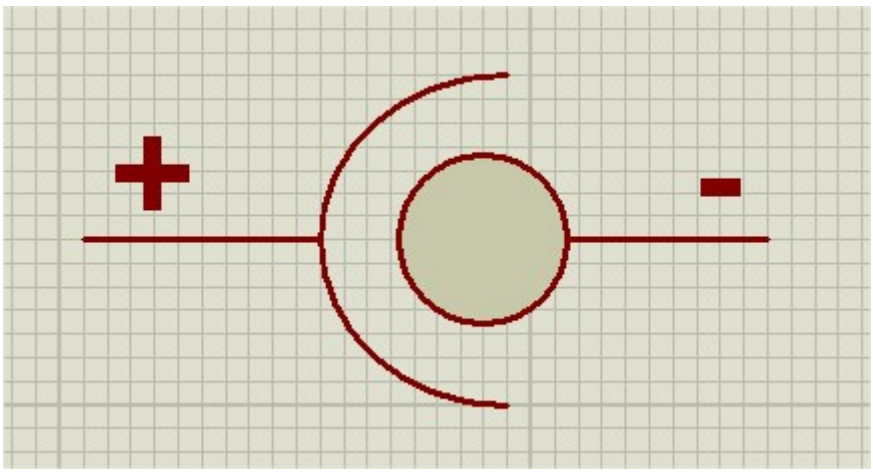

Figura 15 - Padrão com o positivo externo

Para identificar qual padrão é a fonte que você está usando, basta procurar no corpo da fonte o símbolo como mostrado na **Figura 14** ou **Figura 15**.

A filtragem, tanto na entrada quanto na saída, serve para evitar que ruídos atinjam nossa placa e prejudiquem funcionamento dela. Na entrada, geralmente temos ruídos de alta frequência, para isso, vamos usar um capacitor pequeno, aproximadamente, 100nF. Já para a saída, temos ruídos gerados pelo regulador, nesse caso, vamos usar um

capacitor de valor maior, cerca de 10uF. Também usaremos na saída um capacitor com um valor de capacitância pequeno para os casos de ruídos de alta frequência atravessarem o primeiro estágio.

Para o circuito regulador, vamos usar um componente bem conhecido no mercado, o regulador de tensão positiva 7805, cuja fabricação é realizada por várias empresas. Com o uso desse regulador podemos conectar na entrada do circuito fontes que gerem tensões de 7,5 até 12V. O regulador se encarregará de disponibilizar ao circuito 5V. Dessa forma, nosso microcontrolador não terá problemas com variações de tensão.

Na natureza nada se perde, tudo se transforma. Portanto, a energia relacionada a sobre-tensão da fonte deve ir pra algum lugar, mais especificamente, é transformada em calor. Logo, devemos ter cuidado com o valor de saída da fonte escolhida, pois quanto maior for ele, maior será a corrente que atravessará o circuito, e, consequentemente, maior será a energia térmica dissipada, podendo levar à queima de alguns componentes. Nós iremos utilizar uma fonte de tensão com saída de 9V. Em alguns casos, é necessário utilizar um dissipador no circuito devido a altas temperaturas de trabalho.

A **Figura 16** mostra o circuito da fonte completo e a **Figura 17** mostra o circuito da fonte com os módulos em destaque. Verifique que o módulo de status é composto por um LED e resistor, e, basicamente, serve para indicar quando a placa está energizada.

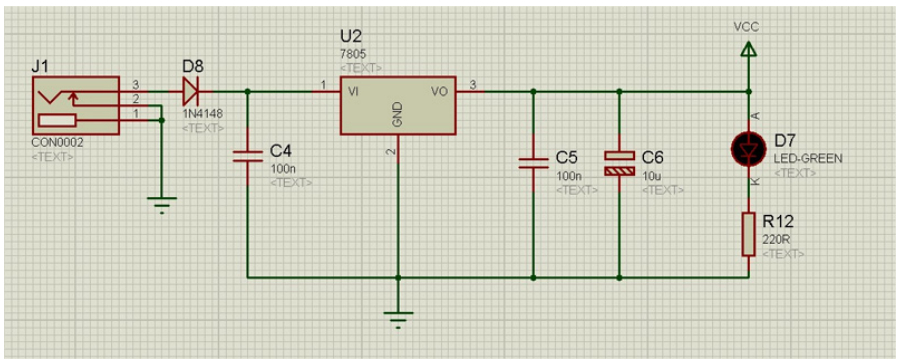

Figura 16 - Circuito da fonte

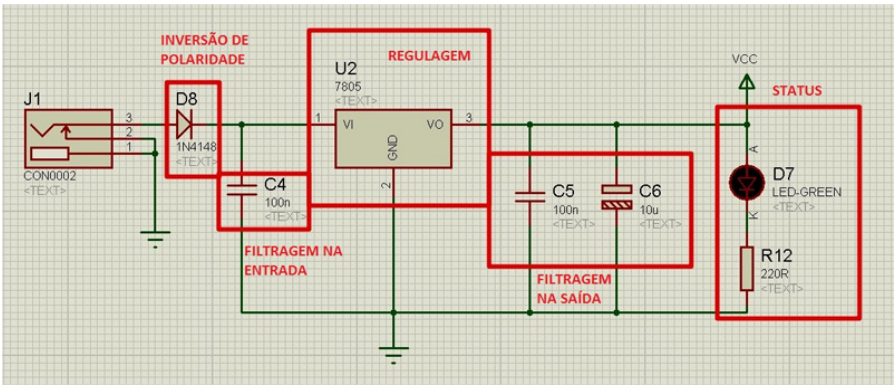

Figura 17 - Circuito da fonte modularizada

Para o circuito da fonte, os componentes usados são: CON0002, 1N4148, CERAMIC15P (com o valor alterado para 100n), 7805, GENELECT10U50V, LED-GREEN, RES e terminais. Quando for criar esse circuito, lembre-se da polaridade de alguns componentes para que ela não fique invertida. É nesse circuito que definimos a origem da tensão de VCC, ela irá alimentar todo o restante do circuito.

Circuito do microcontrolador

Para finalizar os trabalhos dentro do ISIS, temos algumas modificações no microcontrolador, dessa maneira, devemos adicionar mais um capacitor para ele e os terminais correspondentes. O circuito do microcontrolador atualizado é mostrado na **Figura 18**.

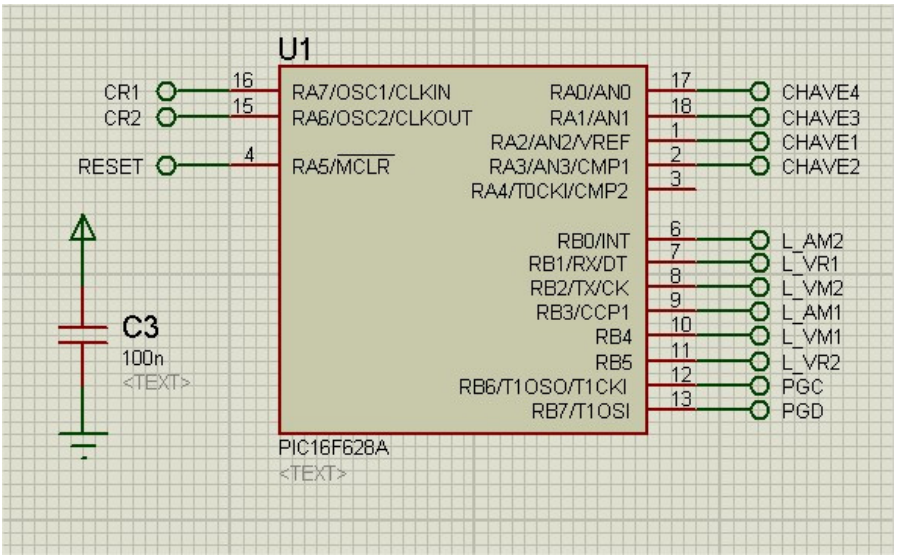

Figura 18 - Circuito do microcontrolador

Circuito final da placa

Depois de adicionar todos esses circuitos, o projeto no ISIS deve ficar como mostrado na **Figura 19**. Se você conseguiu, parabéns! Se não peça ajuda ao tutor ou monitor presente.

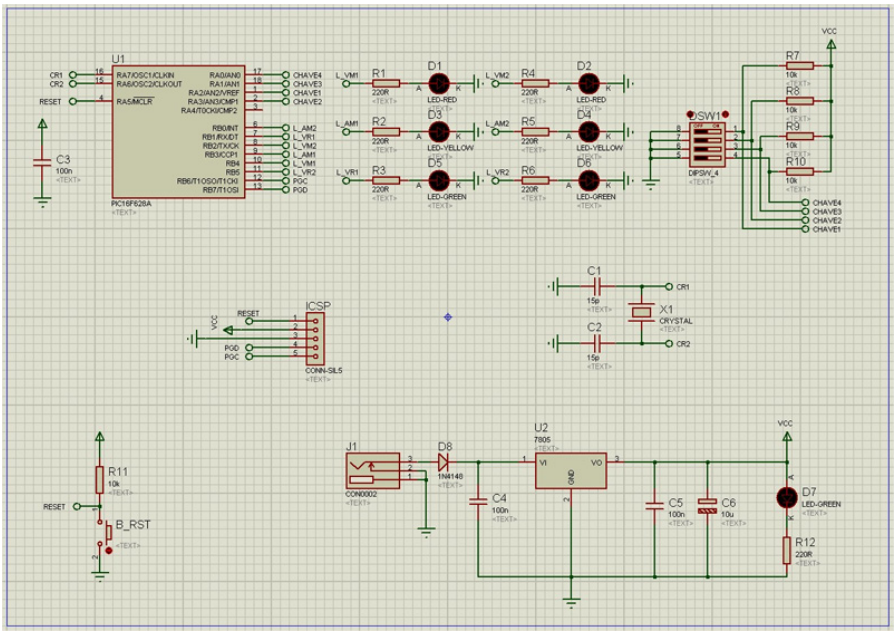

Figura 19 - Circuito final do ISIS

Dica

Conforme mostrado na **Figura 19**, tente sempre trabalhar modularizando os circuitos, isso irá ajudar bastante quando for criar a placa.

Proteus (ARES)

Vamos trabalhar com o ARES. Como já vimos, ele é o responsável por criar o projeto físico da placa, criar as trilhas e definir as regras de roteamento. Para abrir o ARES, basta clicar (dentro do ISIS) no botão

, automaticamente um outro aplicativo será aberto. Inicialmente, pode aparecer uma janela perguntando qual o tipo de *layout* que devemos escolher, basta selecionar *DEFAULT*. A **Figura 19** mostra a tela inicial do ARES.

Caso o ARES não tenha o encapsulamento de algum componente, será solicitado que você indique qual tipo de encapsulamento deve-se utilizar.

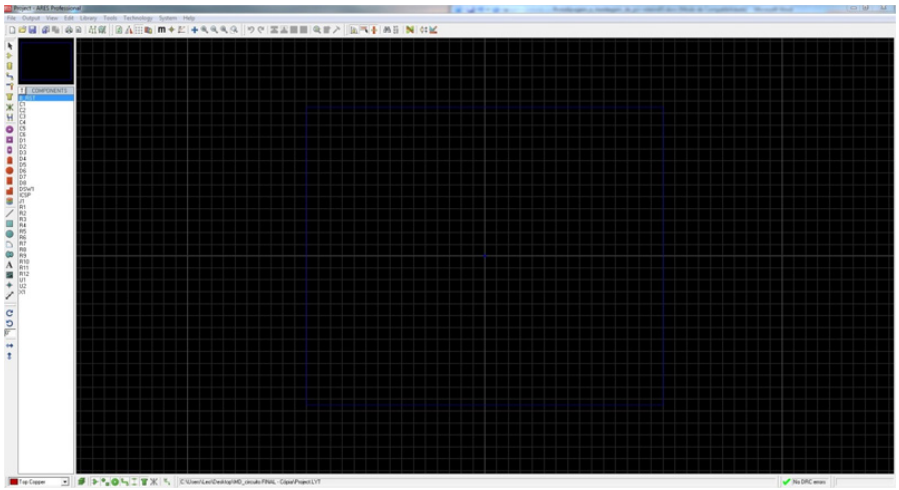

Figura 20 - Tela inicial do ARES

Ao construir o esquemático do circuito no ISIS usando os componentes é gerado o 'netlist' do circuito, que é responsável por descrever a conectividade de um circuito eletrônico. Ao iniciar o ARES, já se tem boa parte das informações para construção da placa final: em particular, comumente, os encapsulamentos já estão relacionados aos símbolos do esquemático, o que possibilita ao ARES pré-selecionar estes para inserção na área de design do layout da placa. Isso nos trás uma importante distinção a ser feita: a diferença entre componente e encapsulamento.

Um componente no Proteus é uma instância de um modelo eletrônico de dispositivo ou componente eletrônico cujo símbolo/desenho não representa necessariamente o modelo físico. É tanto, que no ISIS, devido ao modelo matemático do circuito gerado, é possível fazer a simulação do mesmo, como vimos anteriormente, podendo-se testar se a construção dele está correta.

Um encapsulamento é a representação física de um componente eletrônico de um fabricante específico e/ou modelo específico, com as mesmas dimensões e características. Esses modelos físicos ficam

armazenados na biblioteca do ARES.

No ARES, ao ativar o ‘Componente Mode’,

, estará disponível os mesmos componentes que foram utilizados no ISIS para montar o circuito, trazendo com eles as informações de conectividade, que poderão ser visualizadas no modo gráfico do ARES por meio de fios verdes à medida que os componentes são inseridos na área de desenho da placa. Por outro lado, no ‘Package Mode’,

, acessa-se apenas às instâncias físicas, não existindo a obrigatoriedade de inserção dos mesmos componentes que foram utilizados no circuito implementado no ISIS, podendo-se fazer, inclusive, outras conexões. Portanto, quando se trabalha com um layout obtido a partir de um esquemático, é mais comum se utilizar exclusivamente o modo componente.

Portanto, observe que vários recursos são parecidos com os do ISIS, mas com uma observação: aqui não podemos criar componente, apenas usá-lo ou criar um novo encapsulamento.

Adicionando Componente

Veja que no modo de componente aparecem todos os nossos componentes criados no ISIS, são eles que devemos adicionar ao nosso projeto. Para isso, basta clicar no componente, clicar uma vez na tela que o componente irá aparecer e, em seguida, escolhemos o local, posicionamos o componente e clicamos novamente. À medida que os componentes são adicionados vão aparecendo também as conexões entre eles identificadas por um fio verde, aparecem também o chamado vetor de força (em amarelo), o qual serve para indicar um possível local daquele componente, a **Figura 20** mostra alguns componentes adicionados.

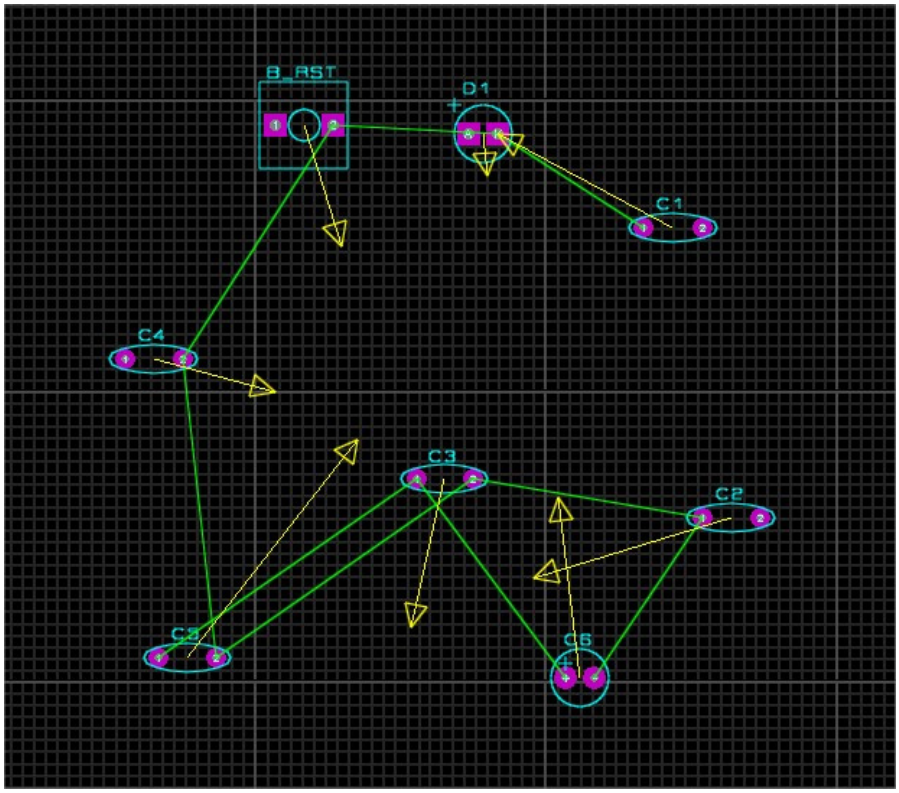

Figura 21 - Adicionando componentes

Raramente os projetistas usam o recurso de vetor de força, caso queiram, podem desabilitá-lo. Para isso, basta ir em View => Layers e desmarcar a opção Force Vectors.

Agora, vamos posicionar todos os componentes, inicialmente, vamos separá-lo por módulos, igual como fizemos no ISIS. Para o microcontrolador vamos colocá-lo no centro do circuito, pois esse componente liga-se a todos os outros módulos. A **Figura 22** mostra todos os componentes posicionados para facilitar a identificação dos módulos. Vejam também que a opção vetor de força foi desabilitada.

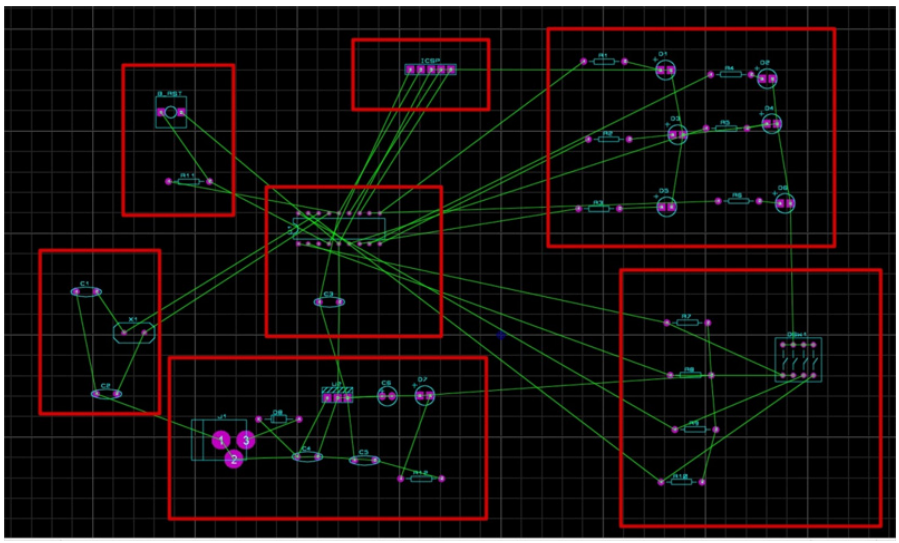

Figura 22 - Circuito com todos os componentes separados por módulos

No ARES, podemos rotacionar os componentes, assim como no ISIS. Para isso, basta clicar com o botão direito e selecionar a opção desejada. Quando adicionamos um componente na placa, ele sempre fica do lado chamado TOP que corresponde à parte de cima da placa. Se escolhermos espelhar, isto é, escolher a opção X-Mirror ou Y-Mirror, o componente vai para o lado do BOTTOM que é a parte de baixo da placa. Não façam isso, pois em nosso projeto nós utilizaremos uma placa de face simples (componentes ficam no TOP da placa) e a depender do tipo de componente, não se terá como soldá-lo à placa.

Se um componente for deletado no ARES, ele somente será excluído da placa, mas irá continuar no projeto. Caso queiram modificar os componentes, devemos fazer isso no ISIS. Os comandos são simples: mover, rotacionar, deletar, dentre outros.

Vamos posicionar nossos componentes da forma como mostra a **Figura 23**. Esse é o padrão de ordem dos componentes escolhido pelo projetista. Caso queira fazer outras versões, fique à vontade.

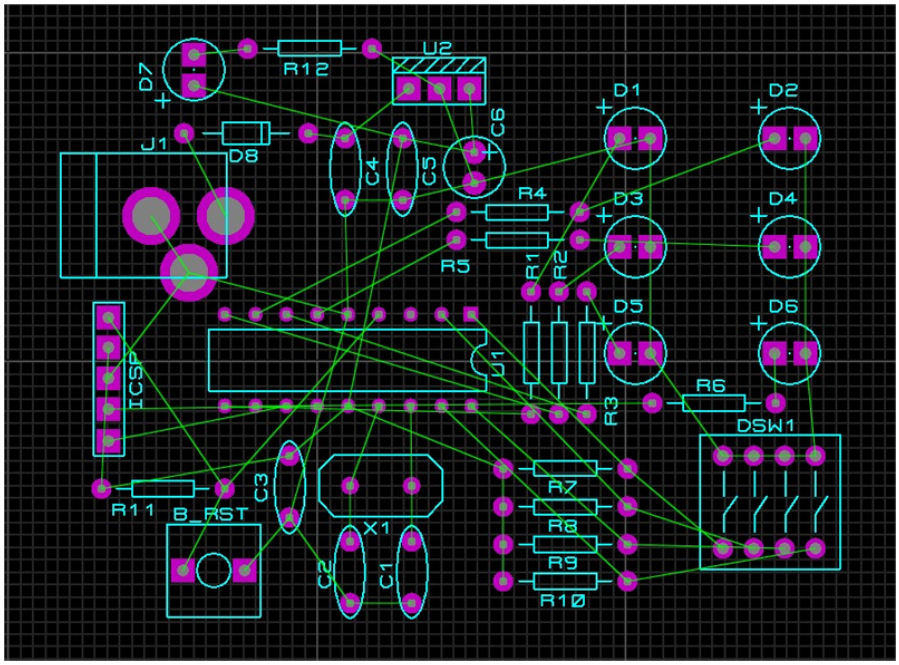

Figura 23 - Circuito com os componentes na posição final

Lembre-se de que o local onde colocamos cada componente corresponde à posição física que ele será soldado. Para o posicionamento dos componentes, algumas regras básicas devem ser seguidas: colocar chaves e botões de forma acessível; não colocar gerador de ruído (regulador) próximo ao relógio do circuito; posicionar o gravador na borda externa; o capacitor de filtro do microcontrolador deve ser colocado próximo ao microcontrolador e afastar componentes que trabalham em temperaturas maiores. Todas essas regras foram seguidas para esse nosso projeto, inclusive a posição de cada Led para o semáforo. Caso seja necessário, podem ser criados pontos de fixação para apoiar a placa.

O próximo passo é criar a borda externa, ela será os limites da nossa placa. Para isso, vamos em 2D Graphics BoX Mode

, no *layer* que fica do lado esquerdo. Na parte inferior, escolhemos *Board Edge,* conforme mostrado na **Figura 24** e criamos um retângulo cobrindo todos os componentes. Esse retângulo pode ser redimensionado posteriormente. Os outros *Layers* que aparecem são específicos, os mais usados são: TOP (parte de cima da placa), BOTTOM (parte de baixo da placa) e BOARD EDGE (corte externo da placa). Tanto para o TOP quanto para o BOTTOM existem outras opções como COPPER (cobre da placa), SILK (nomenclatura da placa) e MASK (usado para criar a máscara de solda). Algumas dessas opções serão usadas mais adiante nesse curso.

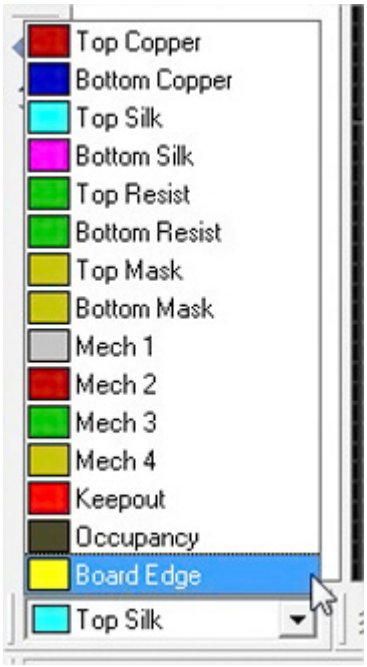

Figura 24 - *Layer* que podemos trabalhar no ARES

Após adicionar a borda externa da placa, o circuito deve ficar como mostrado na **Figura 25**.

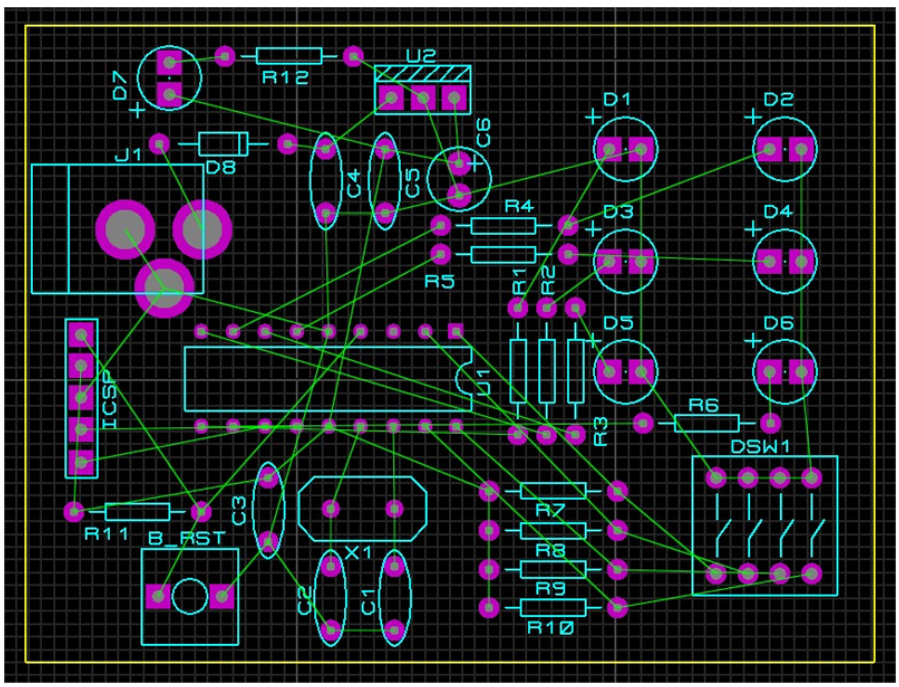

Figura 25 - Circuito com os componentes e a borda externa

Criando as trilhas

Vamos criar as trilhas do nosso circuito. A regra básica é que para tensões de alimentação as trilhas devem ser maiores; para sinais, podemos usar o tipo de trilha padrão. Para atingir uma boa experiência em roteamento de circuito deve-se praticar bastante. Nesse curso, criaremos somente as trilhas do tipo padrão e, mesmo assim, faremos as mais próximas, o restante será feito automaticamente. Para aperfeiçoar a técnica, vocês podem praticar futuramente. Um roteamento manual de uma placa de simples face pode demorar dias, dependendo do tamanho e da quantidade de componentes.

Para criar as trilhas, vamos clicar no *Track Mode*

, em seguida, clicamos no pino do componente que queremos criar a trilha e arrastar até o próximo pino do outro componente. A **Figura 26** mostra esse passo a passo. Nesse ponto, o *scroll* do mouse ajuda bastante quando queremos aplicar um zoom na tela que estamos trabalhando.

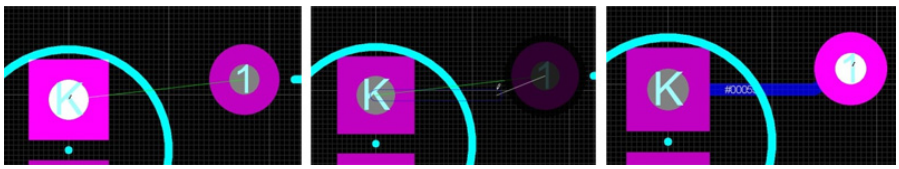

Figura 26 - Criando trilhas

Dica

Em larga escala, quanto menor for o tamanho da placa, mais barato será sua fabricação. Outros fatores que também influenciam na fabricação de placas é a quantidade de *layer* e o tipo de material que será utilizado.

Conforme explicado anteriormente, vamos criar as trilhas mais próximas, use toda a placa para criar várias trilhas, elas devem ficar na cor azul representando que são trilhas do BOTTOM, pois os componentes

estão no TOP e são soldados no BOTTOM. Se a trilha ficar na cor vermelha é porque o *layer* selecionado está incorreto, para desfazer, basta apagar a trilha errada, selecionar o layer correto e continuar.

Para criar o roteamento automático, vamos em Tools => Auto Router... e clicamos em Begin Routing. Observe que o roteamento automático será aplicado, usando trilhas no BOTTOM e no TOP, mas as trilhas que foram criadas manualmente irão permanecer intactas. Ao final, o circuito deve está parecido com o mostrado na **Figura 27**.

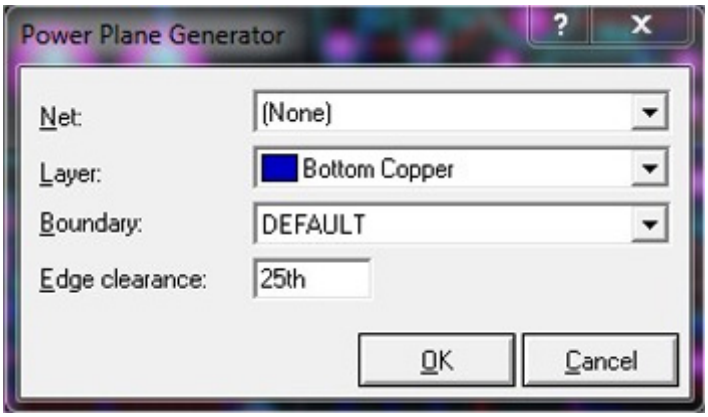

Figura 27 - Circuito com roteamento automático

Em seguida, vamos aplicar a malha de terra no nosso circuito. Ela serve para economizar material e proteger a placa contra descargas estáticas. A descarga estática geralmente acontece no manuseio da placa, principalmente quando tocamos. Se na placa existir uma maior quantidade de cobre conectado ao seu neutro, essa descarga irá para o sinal neutro da placa, o que causa um dano menor.

Apesar da nossa placa não ter aterramento, vamos chamar de malha de terra e fazemos a ligação com o sinal neutro ou GND da placa. Para isso, vamos em Tools => Power Plane Generator, uma janela (**Figura 28**) será mostrada.

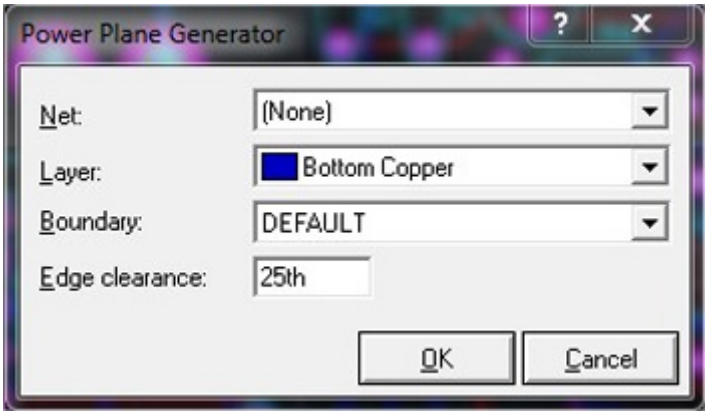

Figura 28 - Circuito com roteamento automático

Na opção Net, escolhemos GND = POWER e, na opção Layer, escolhemos BOTTOM. Se necessário, podemos fazer também no TOP, quando apertamos em OK a malha será aplicada. A **Figura 29** mostra o resultado da placa com TOP e BOTTOM e malha de terra nos dois lados.

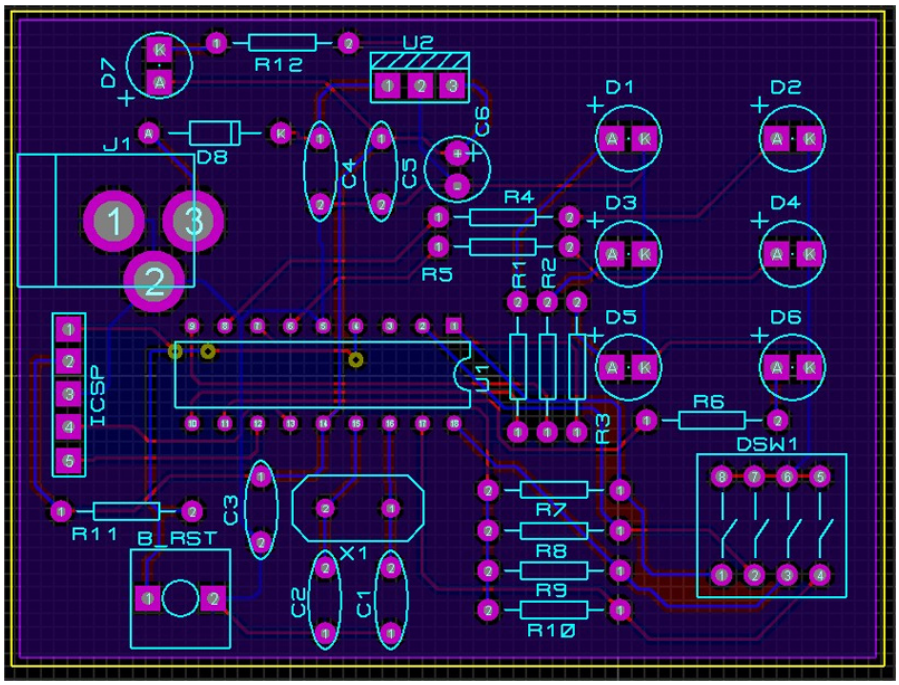

Figura 29 - Circuito com dois *layers*

Se o circuito não tiver erro, uma mensagem no canto inferior direito do ARES terá uma indicação igual à **Figura 30**. Se existir erro, a mensagem será igual à **Figura 31**, ela informa a quantidade de erro encontrada e mostra no circuito o local do erro com um círculo vermelho.

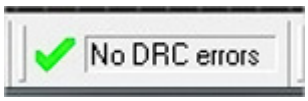

Figura 30 - Circuito sem erros

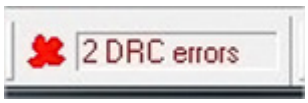

Figura 31 - Circuito com erros

Circuito final da placa com as trilhas

Agora, o circuito está finalizado, o desafio é montar o mesmo circuito usando o roteamento somente no layer do BOTTOM. A Figura 32 mostra o mesmo circuito feito dessa forma, compare-o com a **Figura 27**. e veja a diferença. O circuito da **Figura 32** é o que será usado a partir de agora.

Observe que as trilhas de alimentação tem uma espessura maior do que as trilhas de sinal, conforme já foi explicado nesta aula.

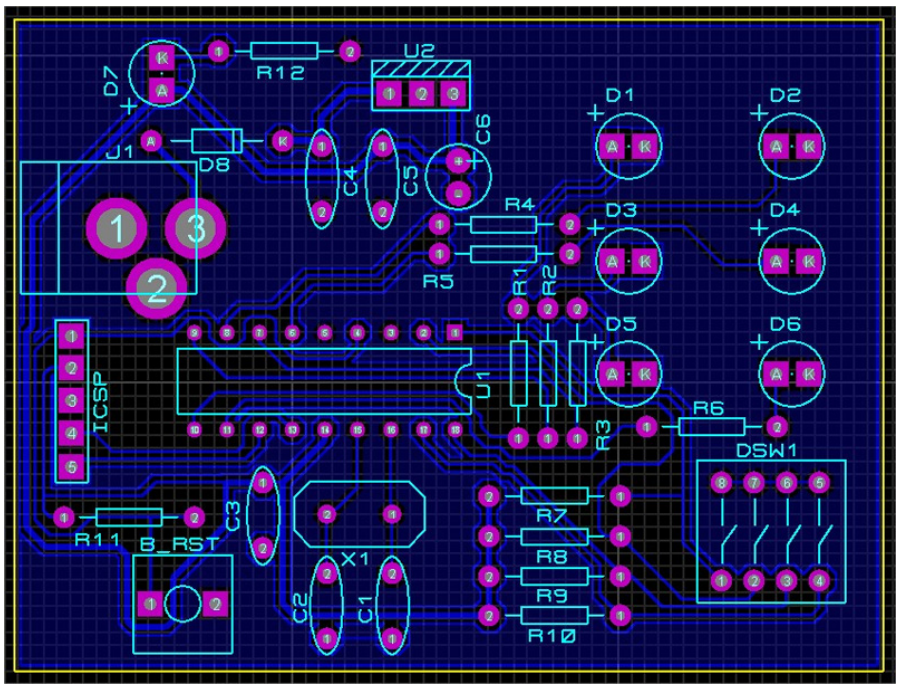

Figura 32 - Circuito com um *layer*

Imagem em 3D

O ARES do Proteus tem um recurso muito interessante, ele permite visualizar em 3D como será a placa. Para isso, vamos em Output => 3D Visualization, basta clicar na tela e começar a rodar com o mouse que a

placa será movimentada. A Figura 33 mostra a movimentação da placa em 3D. Para sair da visualização, basta clicar no botão ESC.

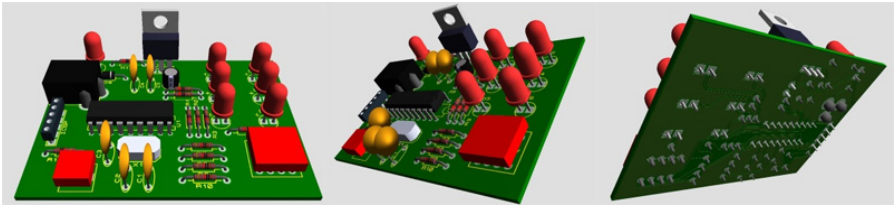

Figura 33 - Circuito com um *layer*

Criando arquivos *gerber*

O último passo, nesta aula, será gerar os arquivos *gerber*. Esses arquivos são necessários para a produção da placa, seja ela para fins comerciais ou de prototipagem. Como nossa placa possui somente o *layer* do BOTTOM e os componentes estão no *layer* do TOP, vamos gerar somente os arquivos necessários para esse fim.

Para gerar esses arquivos, vamos em Output → Gerber/Excellon Output... uma janela irá aparecer pedindo para buscar por erros, desse modo, podemos clicar em Yes. Após a checagem, aparecerá um relatório, se não tiver nenhum erro, podemos continuar. A **Figura 34** mostra a janela de relatório da checagem por erros.

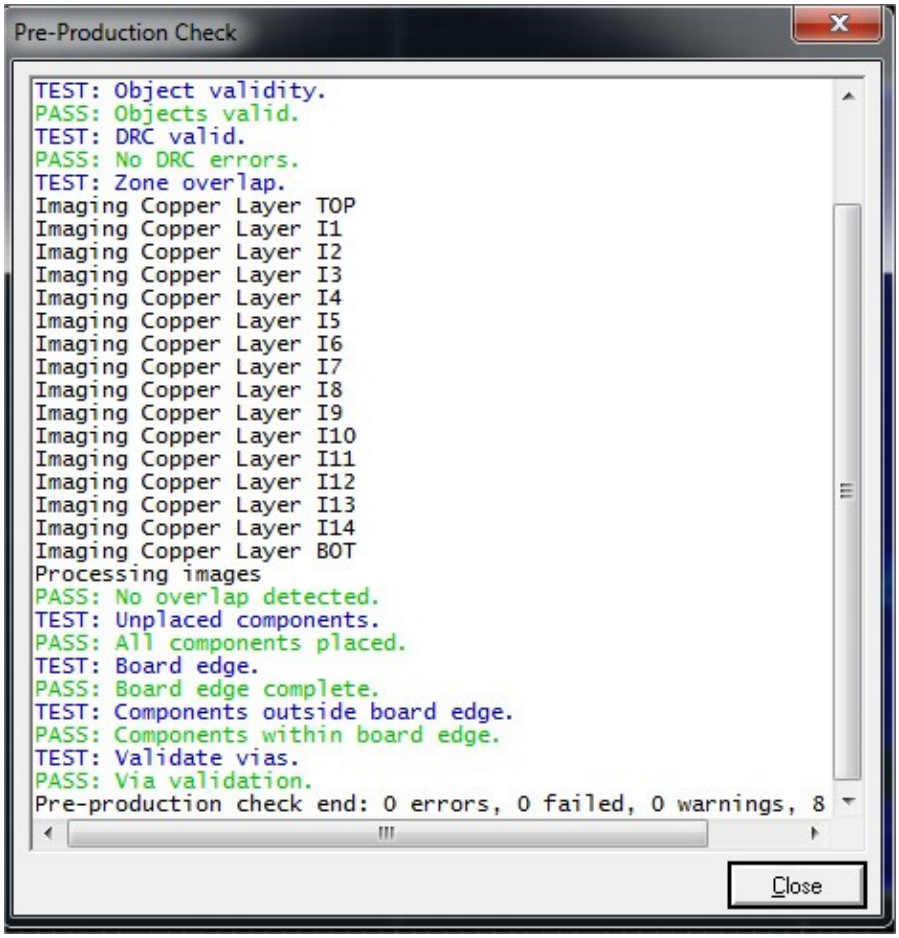

Figura 34 - Checagem por erros

Quando clicamos em Close, a janela de configuração do *gerber* será aberta. Em *Layers/Artworks,* habilitamos: Bottom Copper, Top Silk, Bottom Resist, Drill e Edge. Se nossa placa fosse de dupla face, a configuração seria diferente. As outras configurações da janela não são modificadas, apenas observe que, em Gerber Format, o padrão selecionado é RS274X, esse é o padrão utilizado atualmente por grande parte das empresas de criação de placas. Por fim, em Folder, escolhemos o local de destino dos arquivos *gerber*. A **Figura 35** mostra a configuração do *gerber*.

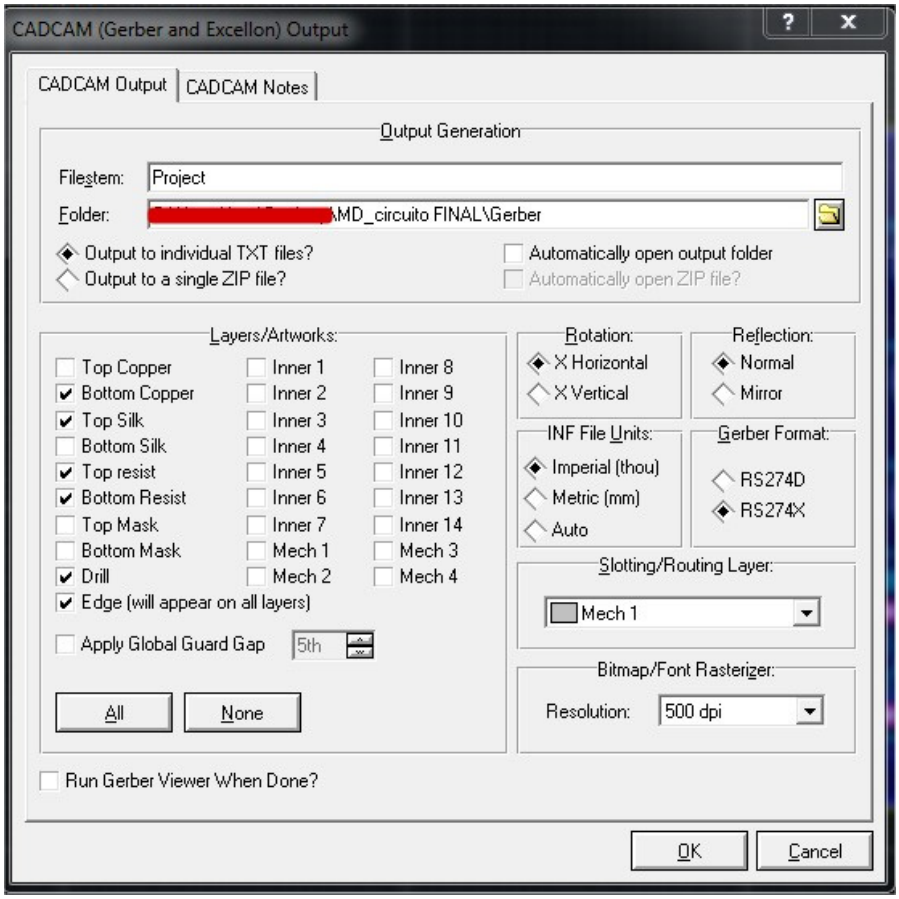

Figura 35 - Configuração do *gerber*

Após a criação do arquivo *gerber*, finalizamos o nosso projeto no Proteus. Esses arquivos são necessários para as próximas aulas. Se tiver algum dúvida, chame o tutor ou monitor presente.

Lembre-se, nesta aula, foi apresentado o básico para usar o Proteus, mas ele possui muitos outros recursos que podem ser explorados futuramente.

Referências

Proteus **Versão do Proteus 6.2 Arturo Sandoval Bermúdez**. 2010. Apostila Completa

Proteus PCB Design Packages. Disponível em: <http://www.labcenter.com/products/pcb_overview.cfm>. Acesso em: 4 set. 2012.

WIKIPÉDIA. **Proteus (programa de computador)**. Disponível em: <http://pt.wikipedia.org/wiki/Proteus_(programa_de_computador)>. Acesso em: 4 set. 2012.