Prototipagem e Montagem de Placa de Circuito Impresso

Aula 06 - Roteiro Prático nº 03 - Desenvolvimento da PCI: ISIS - Parte II

Apresentação

Nesta aula, vamos continuar o desenvolvimento do projeto da PCB utilizando a ferramenta ISIS do Proteus. Alguns componentes como o circuito do clock (oscilador) do microcontrolador, circuito com o conector para gravação do microcontrolador, circuito de alimentação e botão de reset serão adicionados no projeto nesta aula. Ao final da aula, o esquemático do circuito da placa estará finalizado.

Objetivos

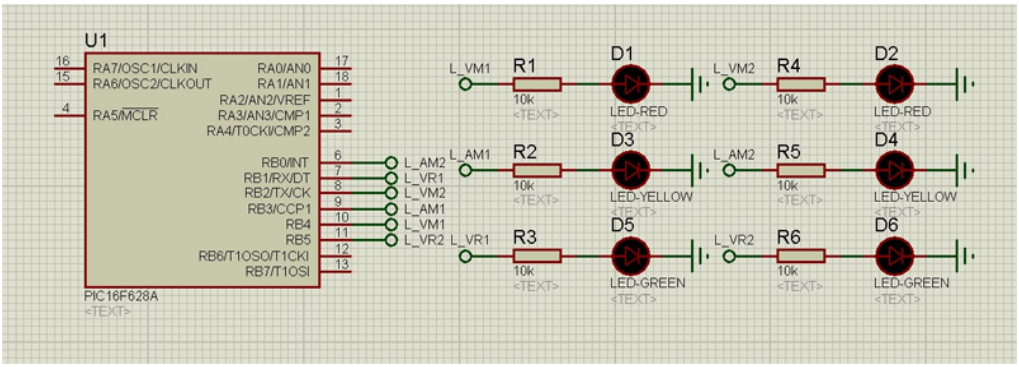
Ao final desta aula, você será capaz de:

- Simular circuitos eletrônicos no Proteus, inclusive circuitos digitais.
- Inserção dos circuitos auxiliares do microcontrolador no esquemático.
- Criação do esquemático final do projeto da placa.

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 01 - 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.

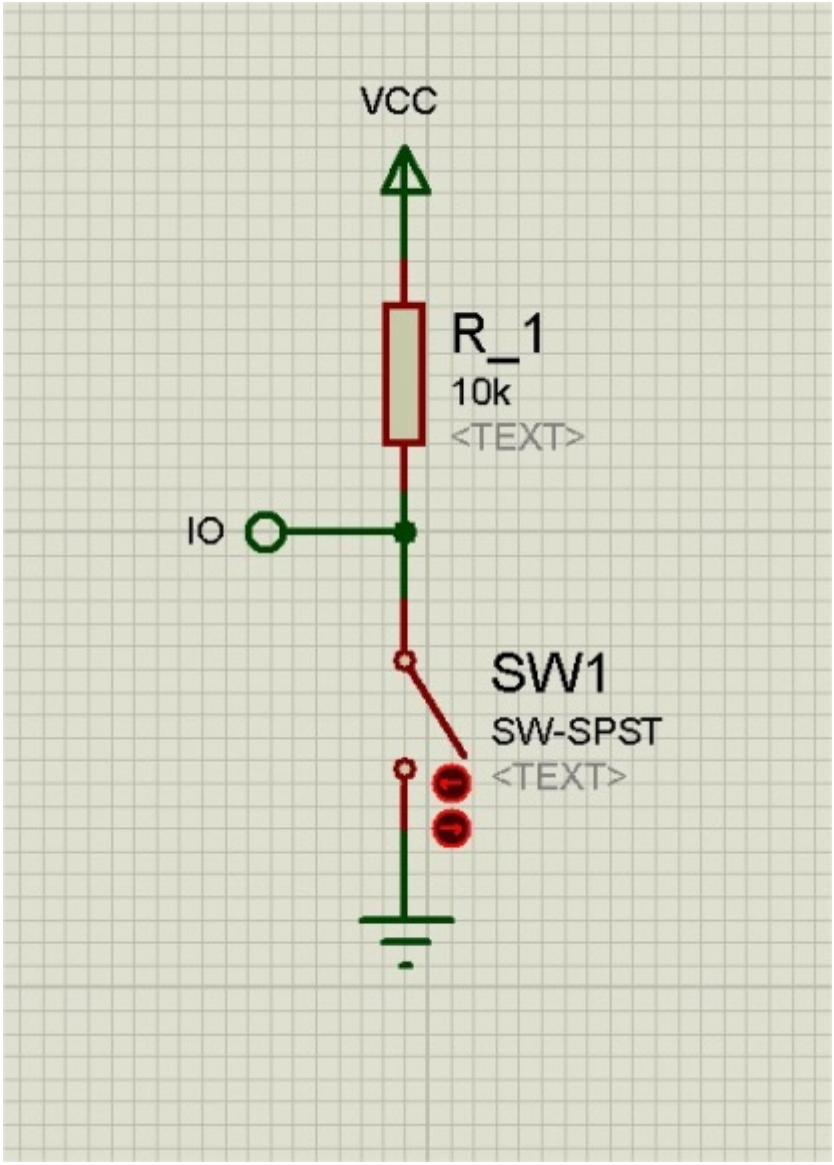
O componente que devemos adicionar é o *DIPSW_4* junto com ele vamos adicionar alguns resistores, lembrem que já fizemos isso na aula anterior. O resistor que vamos usar é chamado de *pull-up*, ele serve para manter a tensão fixa naquele ponto. Caso esses resistores não sejam usados, tal tensão pode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

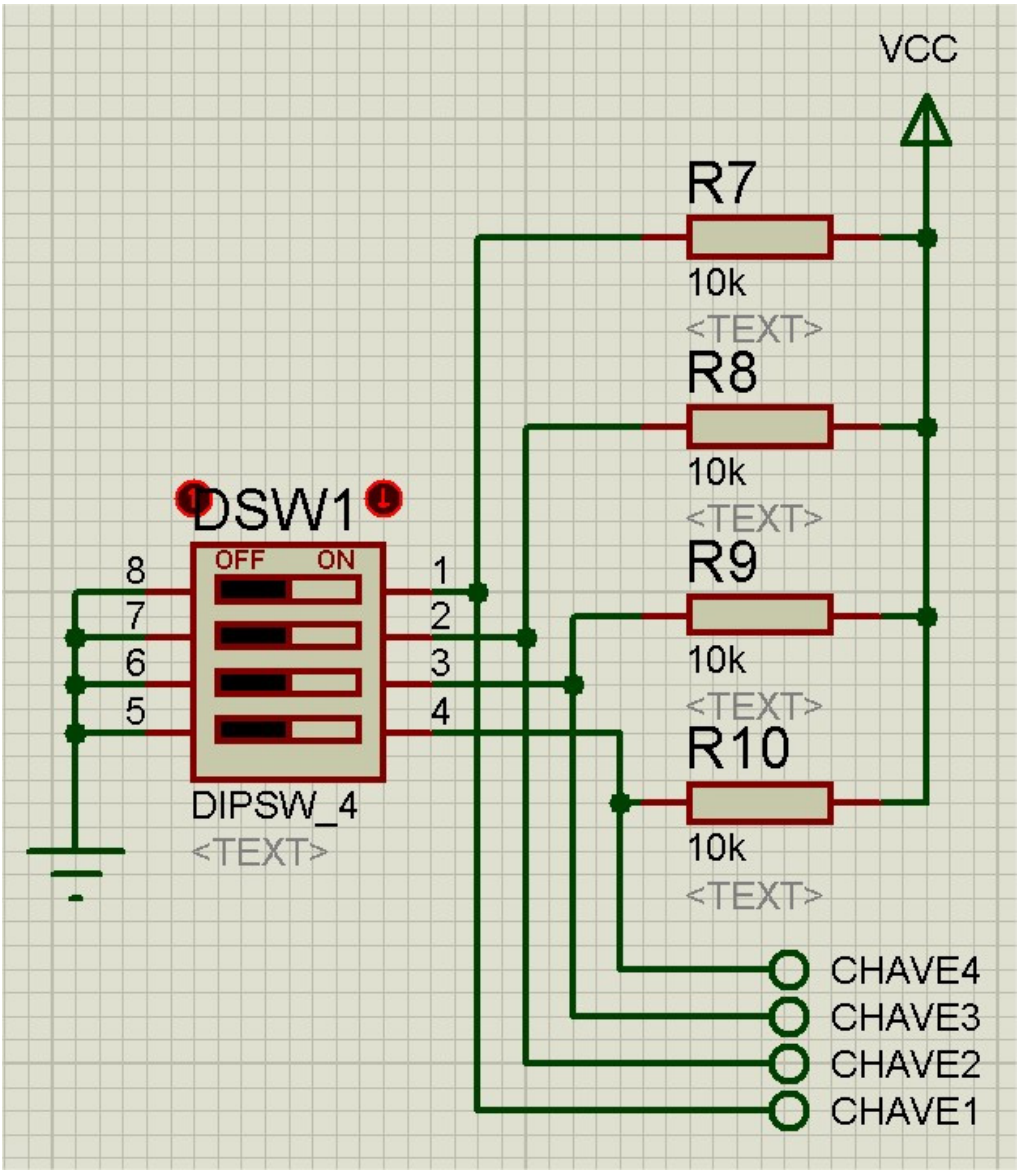
Figura 02 - Exemplo de um circuito com resistor de *pull-up*



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



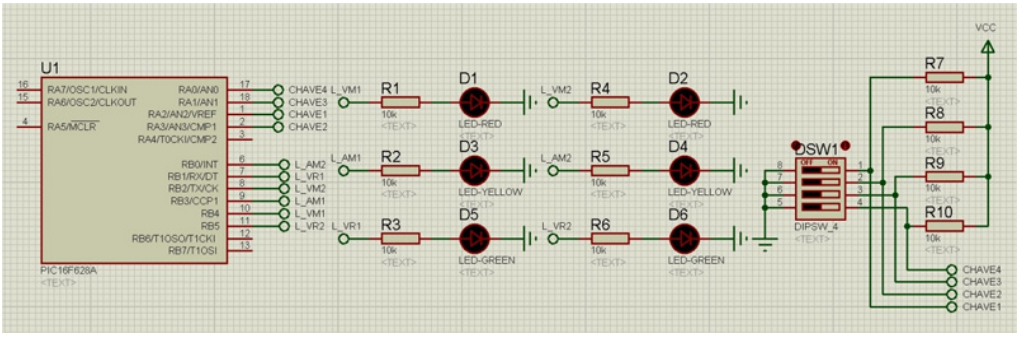
Dica

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.

Para adicionar o componente VCC deve seguir o mesmo procedimento de adição do 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 04 - Circuito com as ligações de I/O



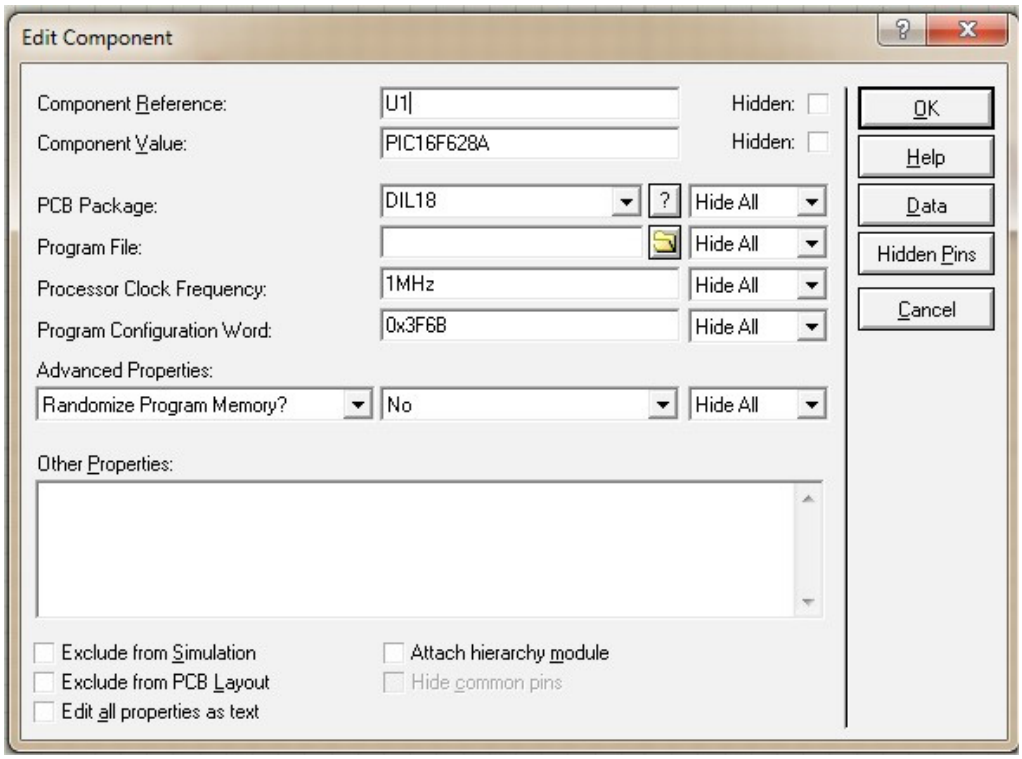
Observe que foram adicionados terminais DEFAULT aos pinos RA0, RA1, RA2 e RA3 do microcontrolador e foram nomeados, respectivamente, de CHAVE4, CHAVE3, CHAVE1, CHAVE2. Dessa maneira, relembrando, nós conectamos, um a um, os terminais do microcontrolador aos terminais do *DIP-Switch* que apresentam a 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 05 - 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.

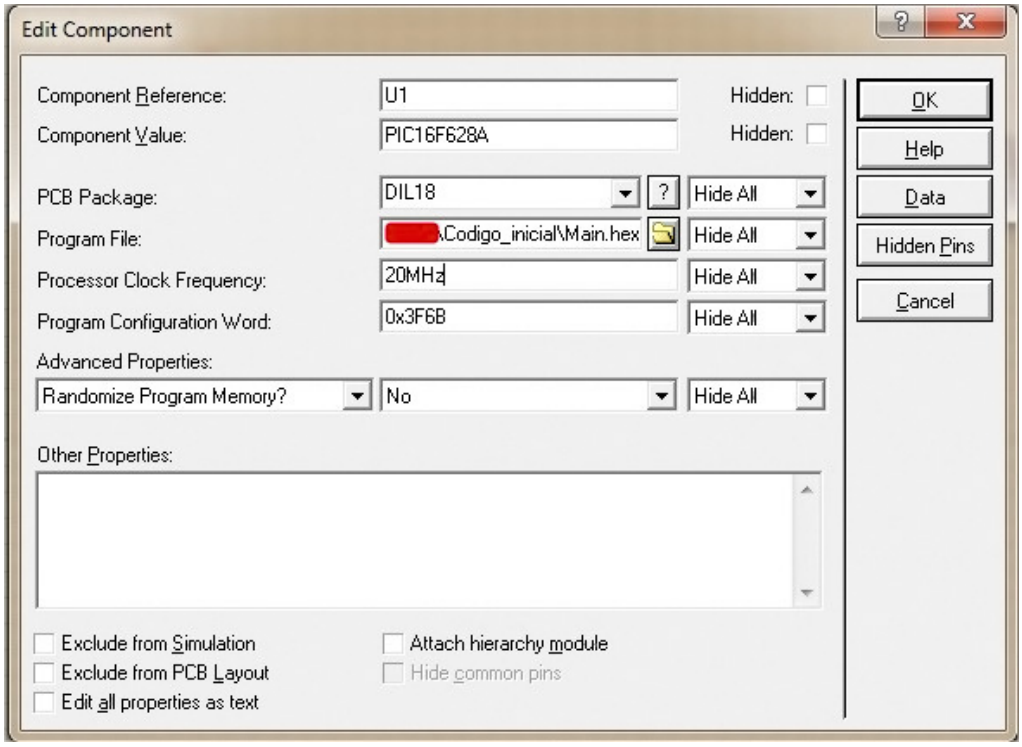
$$10MHz = 10000000Hz = 0,0000001S = 100nS$$

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

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

Logo, serão 10 milhões de instruções.

Figura 06 - 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 07 - Barra de simulação

Observação

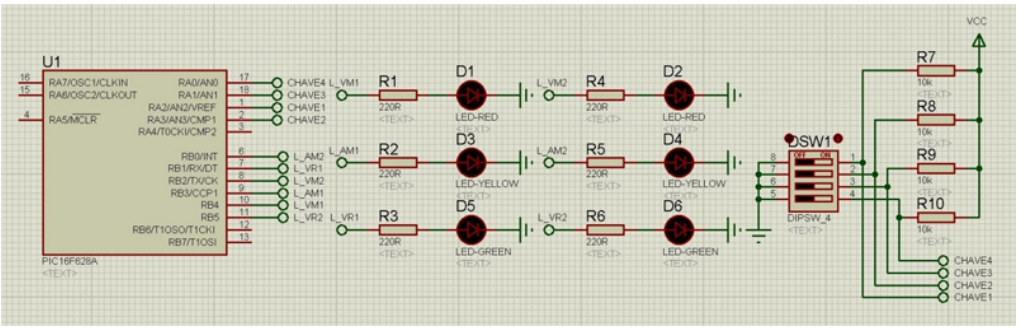
A operação de simulação vai testar o circuito antes de ele ser finalizado. Se o circuito apresentar erros, o Proteus irá abrir uma janela informando o erro. Se isso ocorrer mantenha a calma, pois o professor mediador estará presente para auxiliá-lo.

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.

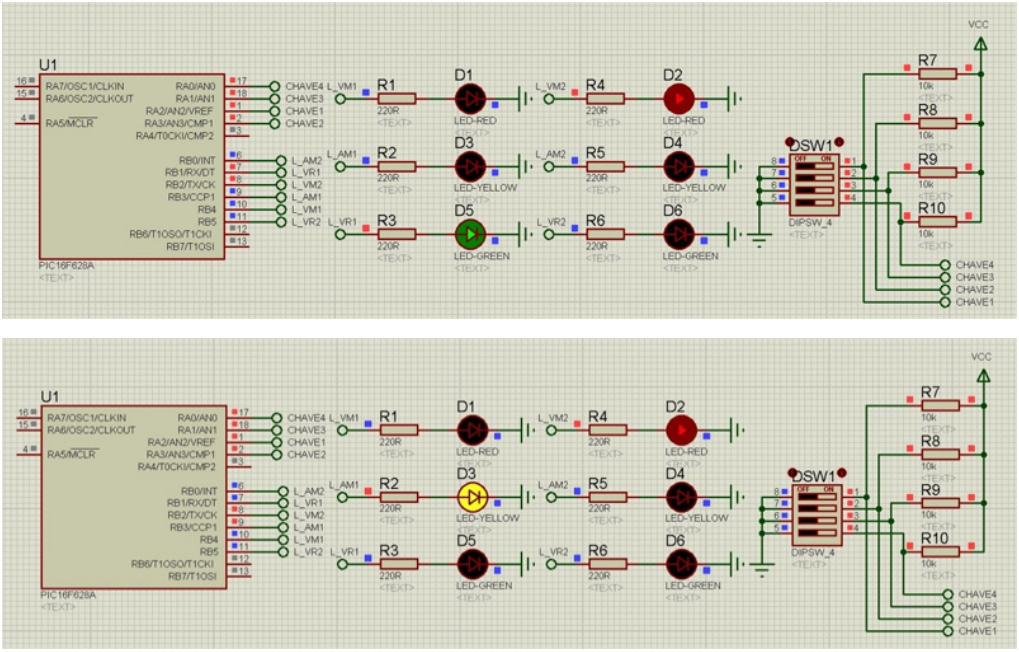
O motivo de deixar os resistores com o valor padrão é mostrar que a simulação não funciona, pois se na prática os resistores tivessem esses valores altos, os LED também não irão acender, nosso esquemático deve ficar como mostrado na **Figura 8**.

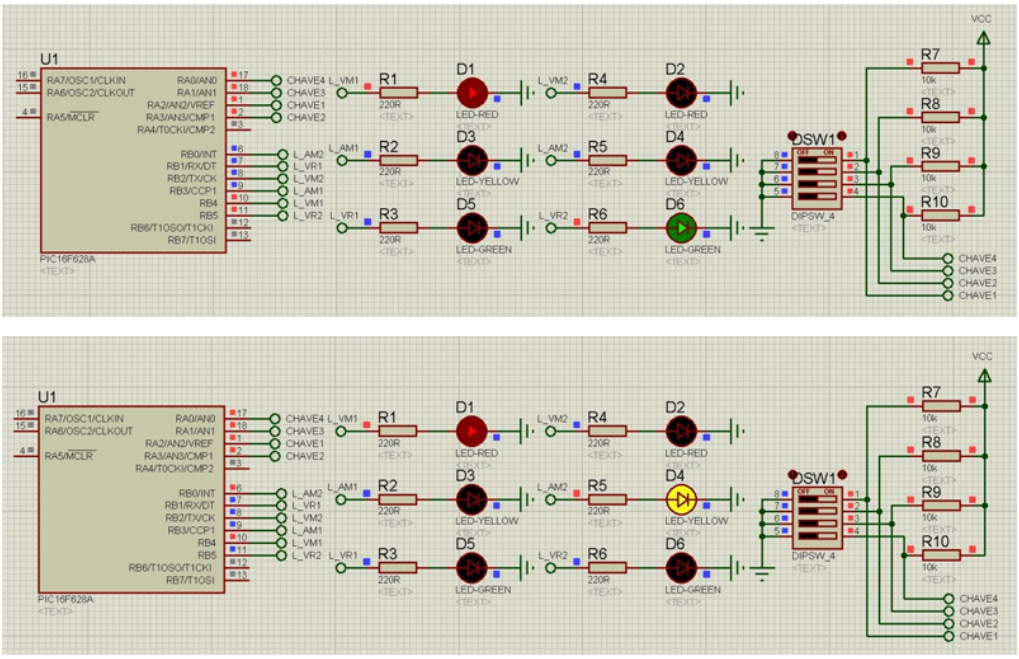
Figura 08 - Esquemático com os valores dos resistores atualizados



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 09 - Simulação do circuito com o código inicial





Adicionando os Outros Circuitos

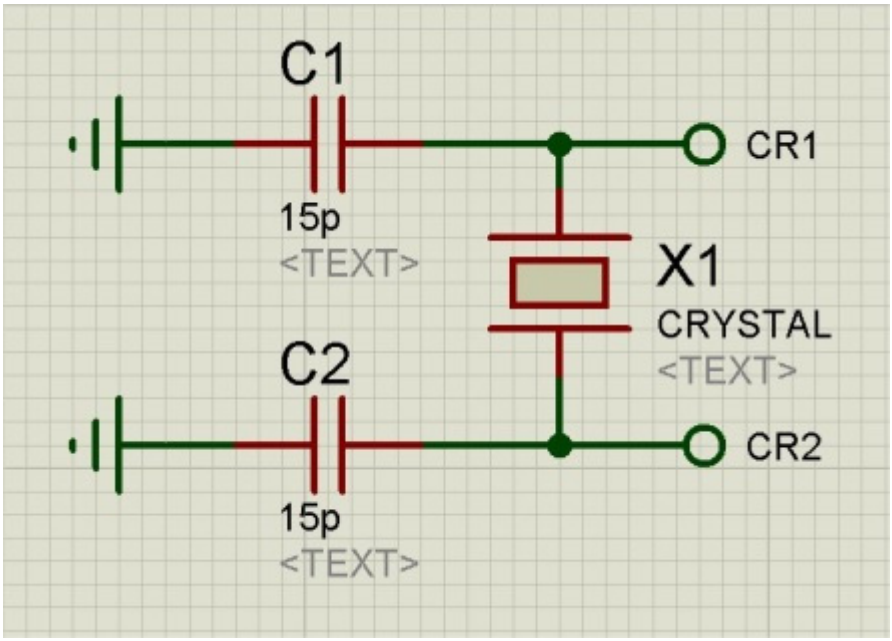
Vamos agora adicionar os outros circuitos essenciais para nossa placa. Eles não foram necessários até esse ponto, pois na simulação não é obrigatório que estejam presentes. São eles: circuito do relógio, circuito do gravador, circuito do *reset* e circuito da fonte.

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**.

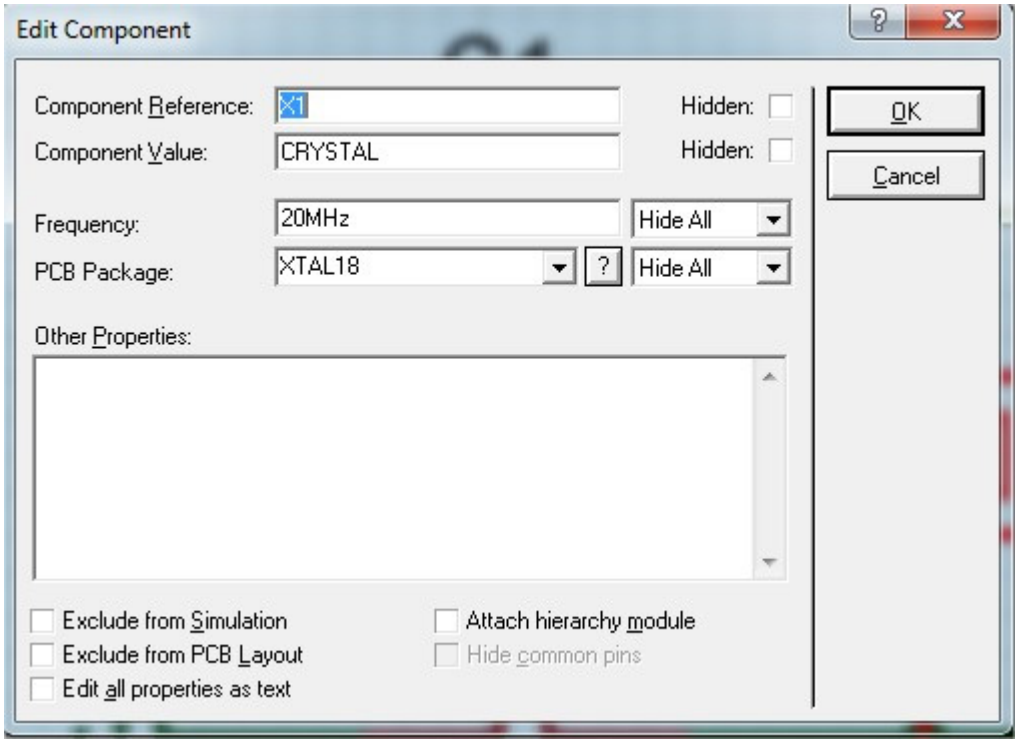
Monte o circuito de forma que fique como mostrado na **Figura 10** Os componentes são CRYSTAL, CERAMIC15P, Terminal DEFAULT e Terminal GND.

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



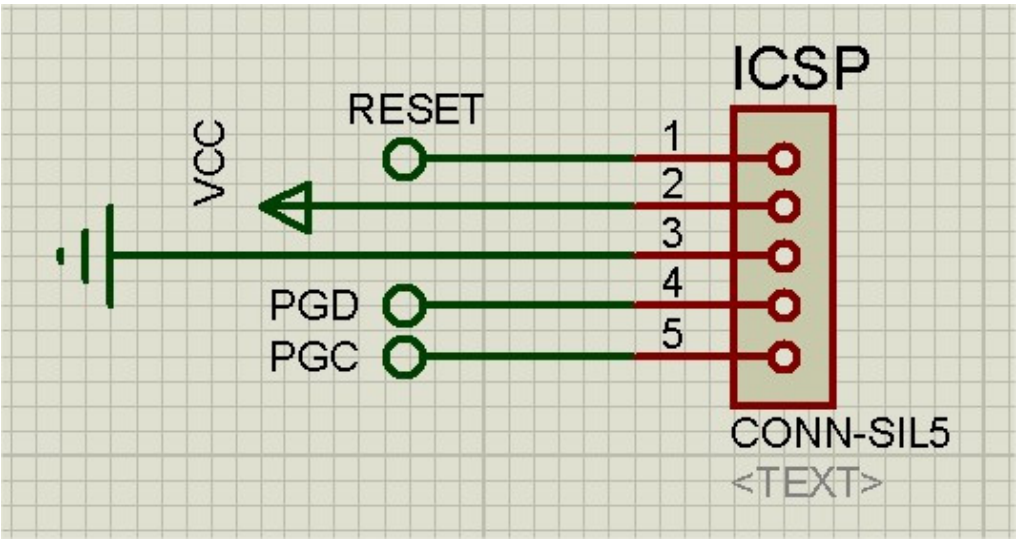
Observação

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

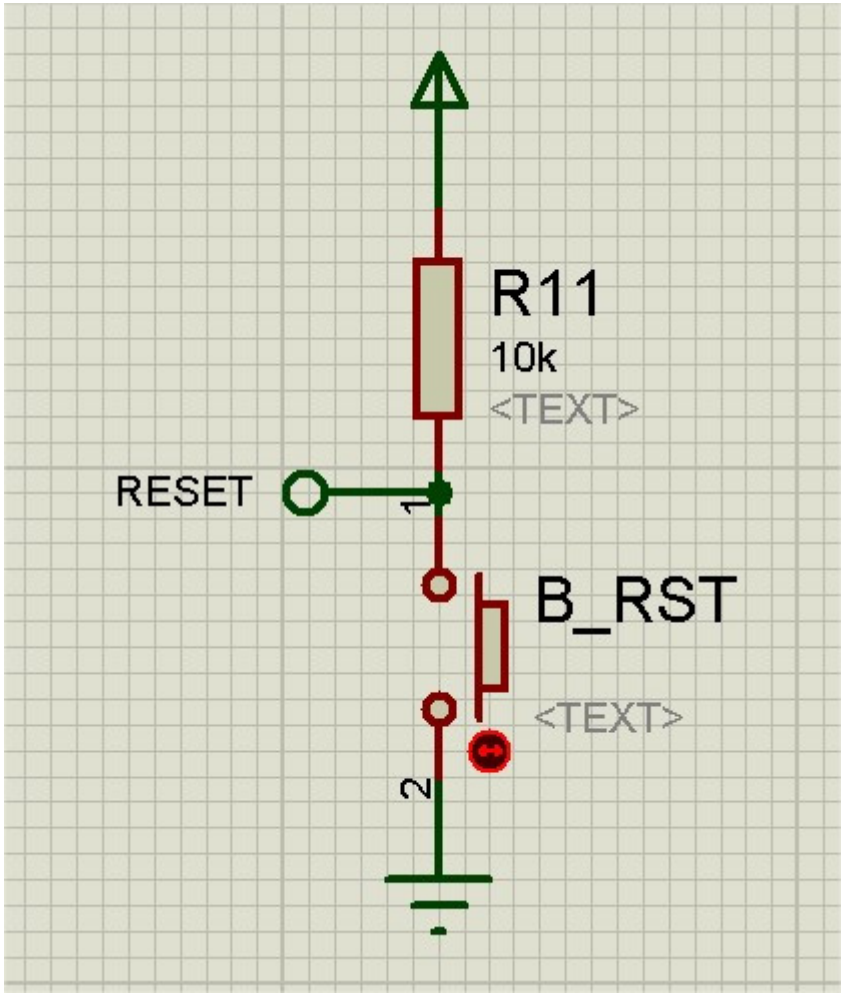


Esse circuito irá ser útil na hora de gravar o nosso *firmware* no microcontrolador.

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*



O pino de *reset* será conectado ao pino RA6/MCLR do microcontrolador.

Circuito da Fonte

Para algumas pessoas o circuito da fonte é considerado o mais complicado, mas basta dividi-lo 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

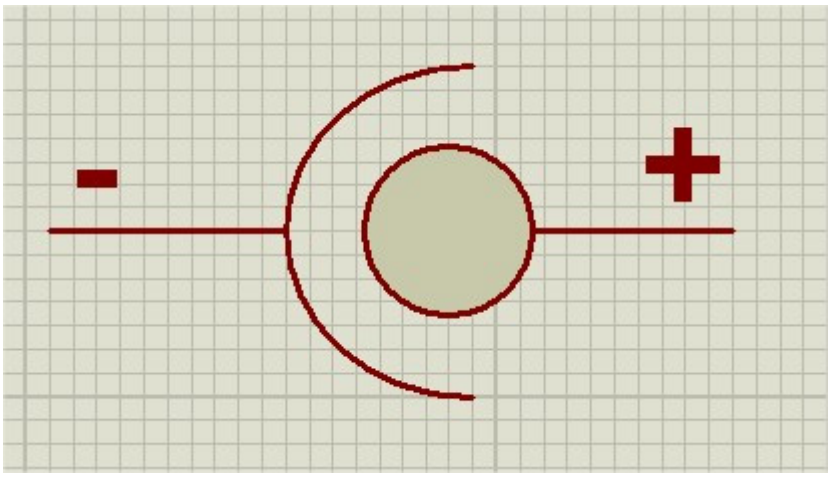
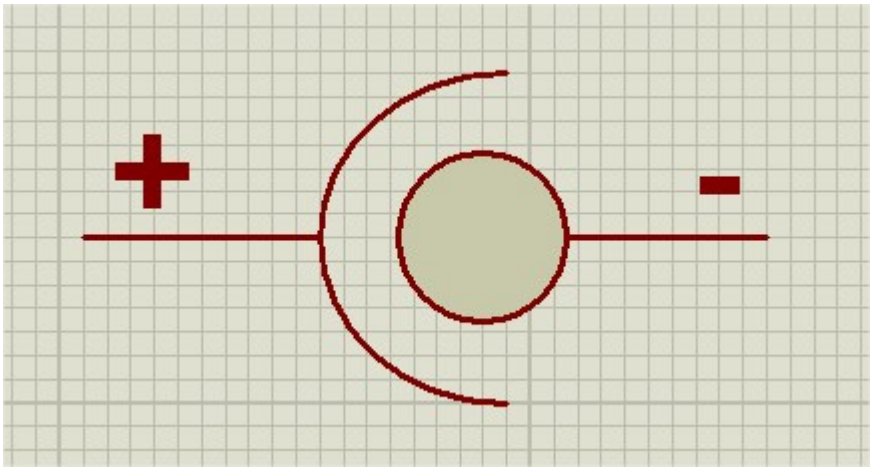


Figura 15 - Padrão com o positivo externo



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 projetar um circuito de alimentação para utilizar uma fonte de tensão com saída de 9 V, conforme mostra a **Figura 16**. Em alguns casos, é necessário utilizar um dissipador no circuito devido a altas temperaturas de trabalho.

Figura 16 - Circuito da fonte

A **Figura 17** mostra o circuito da fonte completo e a **Figura 18** mostra o circuito da fonte 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 17 - Circuito da fonte

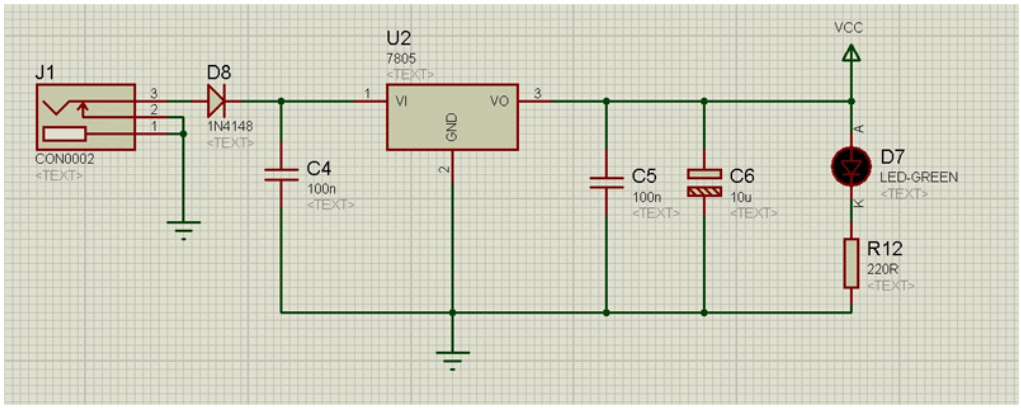
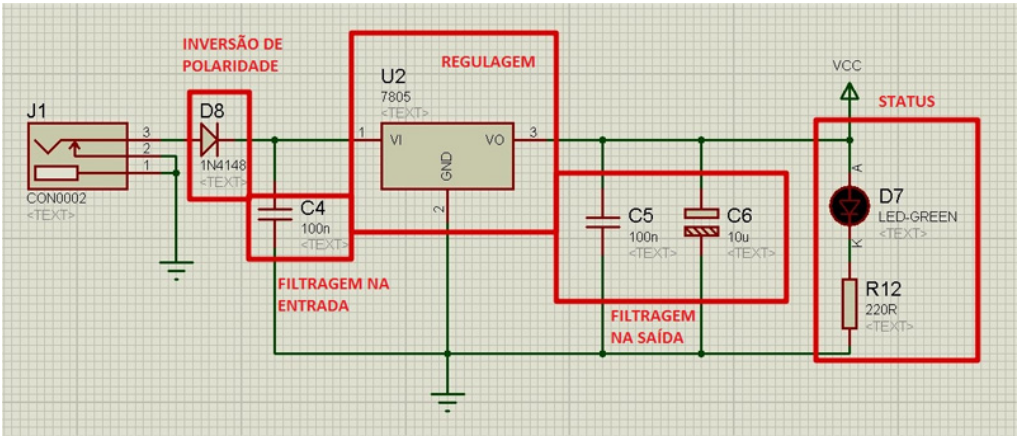


Figura 18 - Circuito da fonte modularizada



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.

Atenção

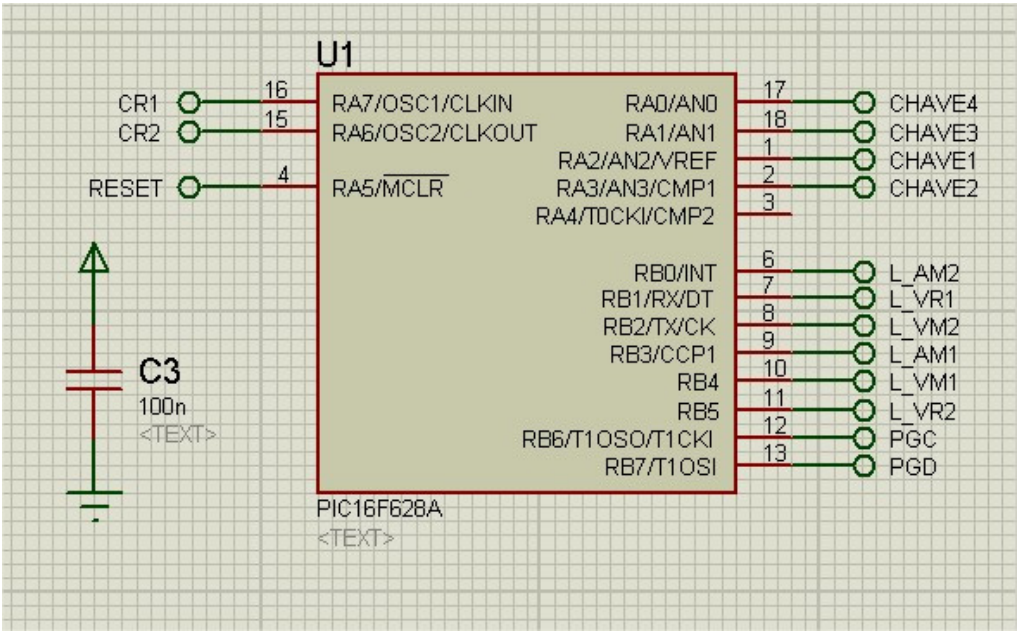
O componente CON0002 não é padrão das bibliotecas do ISIS. Em alguns casos, quando não existe o design do componente, é possível criar um novo componente e inseri-lo na biblioteca do Proteus.

Use os arquivos “CON0002.LYT” e “CON0002.DSN”, fornecidos no ambiente virtual, para adicionar ao Proteus através da opção “Make Device”.

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 19**.

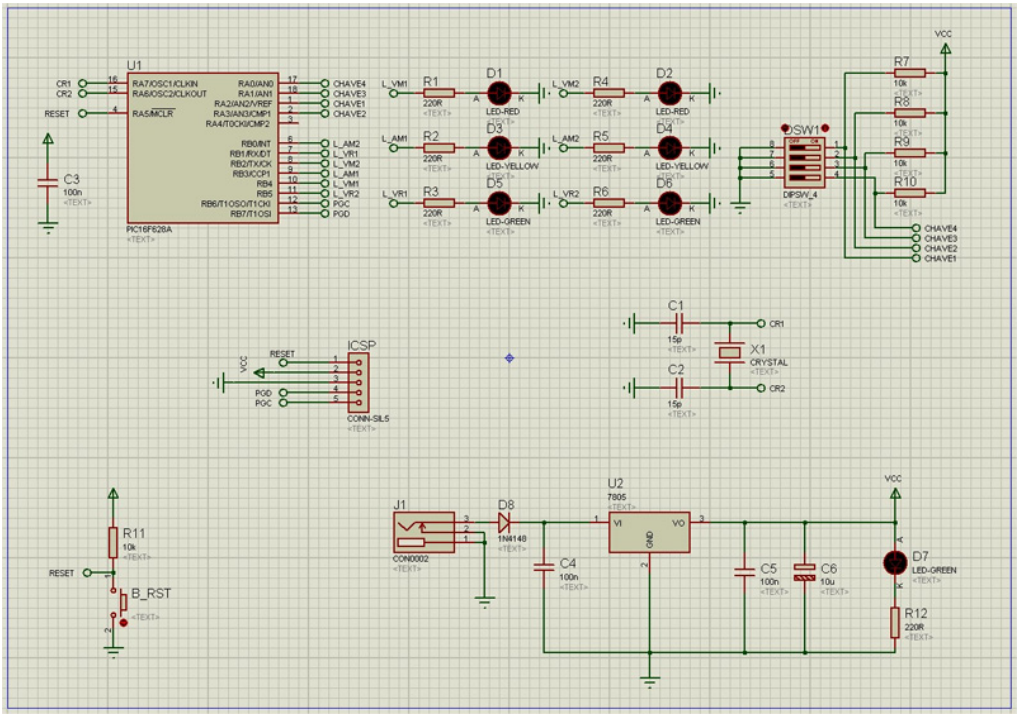
Figura 19 - Circuito do microcontrolador



Circuito final da placa

Depois de adicionar todos esses circuitos, o projeto no ISIS deve ficar como mostrado na **Figura 20**. Se você conseguiu, parabéns! Se não conseguiu, entre em contato com seu professor.

Figura 20 - Circuito final do ISIS



Atenção

Conforme mostrado na **Figura 20**, tente sempre trabalhar modularizando os circuitos, isso irá ajudar bastante quando for criar a placa.

Nesta aula, foram adicionados os últimos blocos do esquemático da Placa de Circuito Impresso em desenvolvimento. Os circuitos de alimentação, do *clock* (relógio), de *reset* e de gravação foram os últimos blocos inseridos no projeto do circuito. O esquemático finalizado nesta aula será utilizado na próxima aula para realizar o design da placa, como posicionamento dos componentes, roteamento das trilhas e outras características.

Referências

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

Proteus PCB Design Packages. Disponível em: <>. Acesso em: 4 set. 2012.

WIKIPÉDIA. **Proteus (programa de computador)**. Disponível em: <>. Acesso em: 4 set. 2012.