Universidade Estadual do Sudoeste da Bahia – UESB
Vitória da Conquista
Ciência da Computação

Painel de LED baseado em Arduino

Marcos Ricardo Santos da Silva

Vitória da Conquista/BA
2013

Marcos Ricardo Santos da Silva

Painel de LED baseado em Arduino

Monografia apresentada como exigência para obtenção do grau de Bacharelado em Ciência da Computação da Universidade Estadual do Sudoeste da Bahia – UESB.

Orientador: Prof. Roque Mendes

Vitória da Conquista/BA
2013

RESUMO

Apesar da popularização dos painéis LED como meio de disseminação na informação, estes ainda tem um custo elevado, com produção restrita a alguns países. Microcontroladores como o Arduino, tem possibilitado ao grande público de projetistas e empresas dedicarem esforços para definir o ambiente de desenvolvimento de software e hardware necessário para obter os benefícios desta tecnologia. O presente trabalho apresenta o desenvolvimento de um painel de LED simplificado, baseado na plataforma Arduino. Com esta tecnologia é possível encontrar soluções de baixo custo, fácil adaptação e de excelente relação custo x benefício. Além do mais, todo o produto desenvolvido é hardware livre, permitindo que qualquer um com interesse possa adaptá-lo à sua necessidade ou melhorá-lo.

Palavras-chave: Painel de LED, computação gráfica, microcontroladores, Arduino.

ABSTRACT

Despite the popularity of LED panels as a means of dissemination of information, these still have a high cost, with production restricted to few countries. Microcontrollers such as Arduino, has enabled large audience of designers and companies devote efforts to set the environment for the development of software and hardware needed to get the benefits of this technology. This paper presents the development of an LED panel simplified, based on the Arduino platform. With this technology it is possible to find solutions of low cost, easy adjustment and excellent cost-benefit ratio. Furthermore, any product developed is open source hardware, allowing anyone with an interest can adapt it to your needs or improve it.

Keywords: LED Panel, graphic computation, microcontrollers, Arduino.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As ideias de liberdade oferecidas pelo software livre podem ser estendidas para o hardware livre, ideia que é quase tão antiga quanto a GPL. Atualmente o hardware livre é uma tendência, tendo como seu maior ícone o projeto Arduíno.

O Arduíno é uma plataforma open-source de protótipos eletrônicos baseados em hardware e software, flexível e fácil de usar. Trata-se de uma plataforma de prototipagem eletrônica de hardware livre, projetada com um microcontrolador Atmel AVR de placa única, com suporte de entrada/saída embutido e uma linguagem de programação padrão. O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de usar. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e de ferramentas mais complicadas.

O Arduíno e o movimento de hardware aberto que têm convertido coisas difíceis em coisas fáceis, e as impossíveis em coisas simplesmente difíceis. Até agora, chegar a uma fase de protótipo de projeto de hardware era difícil, pelo menos para a maioria das pessoas, e ir além de um protótipo bruto era impossível para muitos.

A crescente popularidade do Arduíno fez surgir uma enorme variedade de projetos, desde instalações de arte a controle de dirigíveis. Graças ao Arduíno, muitos problemas relacionados a hardware foram diluídos. Os projetos eletrônicos complexos foram empurrados de volta para as áreas mais especializadas, surgindo novos projetistas com visões diferentes para as tecnologias atuais, mas também com novas ideias para se aplicar esta ferramenta.

1.1 OBJETIVO GERAL

Desenvolver um dispositivo baseado na plataforma Arduíno que permita o controle de painéis de LED, incluindo hardware e software, que seja capaz de exibir texto com caracteres.

1.2 OBJETIVOS ESPECÍFICOS

- Compreender a arquitetura básica e conceitos teóricos que envolvem o funcionamento dos painéis de LED.
- Pesquisar como o Arduíno pode ser utilizado como microcontrolador de painel de LED.
- Implementar o hardware necessário para ter uma interface com o Arduíno, assim como o software que o controle.
- Construir o painel de LED, interligando o hardware desenvolvido à estrutura de LED responsável por exibir as informações no formato texto.

2 REFERENCIAL TEÓRICO

Para implementar este projeto é necessário entender alguns conceitos e o funcionamento dos componentes envolvidos no mesmo, o referencial teórico diz respeito a este ponto do projeto onde é explicado as etapas do processo de aprendizagem para se atingir o objetivo.

2.1 OS PAINÉIS DE LED

Os painéis de LED são dispositivos de saída que fazem o uso de diodos emissores de luz (LED) para exibir dados e informações. Este tipo de display é confundido com as tela de LCD retroiluminado por LED, atualmente bastante utilizado em laptops ou monitores, erroneamente chamados de telas de LED.

Figura 1 - Painel de LED RGB
Fonte: (http://www.madeinasia.com/led-display/LED-Display-325577.html).

Estes dispositivos consistem em painéis ou módulos de LED (Diodo Emissor de Luz – ver figura 1) monocromáticos (LEDs de cor única) ou policromáticos (LEDs RBG ou outras configurações). Em um painel de LED monocromático um único LED é acionado de modo a formar um pixel. Já nos painéis policromáticos, um conjunto

de LEDs vermelho, verde e azul ou um único LED RBG que são acionados para conseguir formar um pixel, geralmente em uma forma triangular ou quadrada. Estes pixels são uniformemente separados e distribuídos numa superfície de maneira a obter uma resolução absoluta de pixel. Dessa maneira, é possível exibir caracteres de texto, imagens e até mesmo vídeos, dependendo da complexidade do dispositivo.

Um problema relacionado às telas de LED é a resolução: Enquanto em um monitor de computador podemos ter uma resolução de 1366x768 pixels, em um display de LED de 4X3 metros tem-se uma resolução de 192x144. Para solucionar esta questão, desenvolve-se uma tecnologia conhecida como pixel virtual, que oferece maior resolução de imagem com a mesma configuração física, fazendo uso de alguns conceitos geométricos básicos.

2.1.1 Principais Usos

Alguns dos principais painéis de LED são: letreiros luminosos, sinalizações de trânsito, mostradores de destino em transportes coletivos, painéis de controle de senha para atendimento, painéis de publicidade e vídeo de alta resolução full-color (em shows, eventos públicos), dentre outras. Na Figura 2 é possível visualizar alguns modelos de painéis de LED.

Figura 2 - Exemplos de painéis de LED

Fonte: (http://www.madeinasia.com/led-display/LED-Display-325577.html).

Os painéis de alta resolução, devido a sua alta resistência ao ar livre, fácil fabricação, manutenção e baixo consumo, se tornaram uma tendência para a disseminação da informação. Seu principal propósito é a exibição de informações referentes às publicidades. Comparado aos cartazes, estes dispositivos oferecem uma informação mais dinâmica, mais fácil de ser substituída, além de configurar a mensagem mais atraente para o destinatário. Ademais, a troca do conteúdo do painel é realizada de maneira centralizada e remotamente, o que é enxergado como grande vantagem para a publicidade de marketing.

Os sistemas computadorizados associados aos painéis proporcionam uma gestão automática do painel. É possível, por exemplo, trocar o conteúdo de acordo com uma data e hora específica, um recurso útil em áreas como cinema, transportes público, estradas e rodovias.

Outro destaque para o emprego de painéis LED de alta resolução (Ver figura 3) ocorre em eventos de massa, da forma a ser impossível usufruir uma boa visão do que acontece. Exemplo de shows, comícios, concursos, estádios, etc. Ainda há os telões de LED decorativos, que muitas vezes fazem parte da iluminação local, como em casa noturnas, bares, pisos e paredes em pistas de danças.

Os painéis de LED de alta resolução são uma nova tendência em mídia exterior. Eles estão impregnados nos grandes centros mundiais e já avançam nas pequenas cidades. Painéis de LEDs em cidades menores, imediatamente se transformam em uma sensação. No Brasil os painéis de LED possuem um alto custo, apesar de já ser possível encontrá-los, não apenas nas grandes cidades, mas também em cidades de médio e pequeno porte.

Figura 3 - Painel de LED usado em um estádio esportivo
Fonte: (http://commons.wikimedia.org/wiki/File:09-02-06-RRS-PigScreen.jpg).

2.1.2 A propriedade Pitch

Uma propriedade importante dos painéis de LED é o Pitch, descrita como a medida da distância entre os centros dos pixels de uma tela de LED. O valor dessa propriedade é diretamente proporcional à distância que um espectador do dispositivo deve ficar. Quanto maior for o Pitch maior deve ser a distância do usuário em relação ao painel de LED. Segue abaixo uma tabela listando algumas distâncias e seus respectivos valores de Pitch:

Tabela 1 - Relação entre distância e valores de pitch

Distância do público	Altura do texto recomendado (polegadas)	Pitch recomendado
25 m	0.5“ (12,7 mm)	12 milímetros
50 m	1" (25,4 mm)	16 milímetros
75 m	1.5" (38,1 mm)	20 milímetros
100 m	2" (50,8 mm)	20 milímetros
150 m	3" (76,2 mm)	23 milímetros
200 m	4" (101.6 mm)	25 milímetros
250 m	5" (127 mm)	28 milímetros

Fonte: Autoria própria.

2.2 O USO DE MATRIZES EM COMPUTAÇÃO GRÁFICA

As matrizes bidimensionais são estruturas utilizadas para representar caracteres, símbolos e imagens. Através destas pode-se realizar diversas operações para manipular objetos gráficos. Nesse contexto, as matrizes são elementos fundamentais para a computação gráfica.

Como exposto por Eduardo Azevedo (2003), em computação gráfica todas as transformações geométricas podem ser representadas na forma de equações. O autor afirma também que este modo de representação faz com que a manipulação de objetos gráficos normalmente envolva muitas operações aritméticas. As matrizes são comumente empregadas nessas manipulações pelo fato de serem mais fáceis de aplicar e entender do que as próprias equações algébricas, o que explica o fato de os programadores e engenheiros as utilizarem extensivamente.

> Devido ao padrão de coordenadas usualmente adotado para representação de pontos no plano (x, y) e no espaço tridimensional (x, y, z), pode ser conveniente manipular esses pontos em matrizes quadradas de 2x2 ou 3x3 elementos. Através de matrizes e de sua multiplicação, podemos representar todas as transformações lineares 2D e 3D. Várias transformações podem ser combinadas resultando em uma única matriz denominada matriz de transformação. (AZEVEDO, 2003, p.33)

Na representação matricial, a imagem é descrita por um conjunto de células em um arranjo espacial bidimensional; uma matriz propriamente dita. Cada célula representa os pixels (ou pontos) da imagem matricial. Graças às matrizes de transformação geométrica é possível que as imagens mudem de posição (translação e rotação) e de tamanho (escala) além de ser possível alterá-las em outros atributos. Portanto, as imagens virtuais que vemos na televisão ou no cinema, mudando de posição, tamanho, cor, se tornando invisível, se dividindo e se compondo novamente, são resultados da utilização de muitas matrizes.

A Figura 4 apresenta a forma de descrição de uma imagem na forma matricial.

Essa é a representação usualmente empregada para formar a imagem nas memórias e telas dos computadores e na maioria dos dispositivos de saída gráficos (impressoras e vídeos).

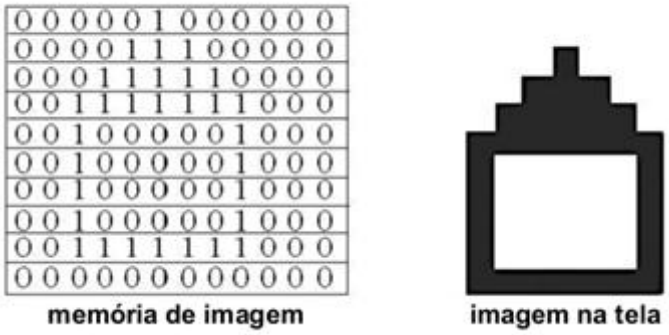


Figura 4 - Representação de uma imagem através de matrizes
Fonte: (AZEVEDO, 2003, p.15).

Embora a saída dos computadores modernos seja geralmente sob a forma de matrizes de ponto, os computadores podem armazenar dados internamente quer como matriz de ponto, ou como um vetor padrão de linhas e curvas. Codificação de dados como um vetor requer menos memória e menos armazenamento de dados, e possui vantagens em situações onde as formas podem precisar ser redimensionadas. Para obter alta qualidade de imagem fazendo uso apenas de fontes matriciais seria necessário armazenar um padrão de matriz de pontos separados para diversos tamanhos de pontos potencialmente diferentes a serem usados. Em vez disso, um único grupo de formas de vetor é usado para processar todos os padrões matriciais específicos necessários para a exibição atual ou tarefa de impressão.

Diversos tipos de tecnologias modernas utilizam matrizes de ponto para exibição de informações, incluindo telefones celulares, TVs e impressoras. Nas impressoras, os pontos são geralmente áreas escuras do papel. Nos monitores de computadores os pontos podem ser luminosos, como nos monitores de LED, CRT ou plasma, ou podem ser escurecidos como um monitor LCD.

2.3 RASTERIZAÇÃO

A Rasterização é a tarefa de tomar uma imagem descrita em uma imagem vetorial e convertê-la em uma imagem raster (pixels ou pontos) para a saída em vídeo ou impressora.

> Rasterização é o processo de conversão da representação vetorial para a matricial. Ela permite realizar a conversão de um desenho tridimensional qualquer em uma representação inteira possível de ser armazenada na memória (de vídeo ou impressão) de um dispositivo raster. Grande parte dos dispositivos de entrada e saída, tal como filmadoras digitais, scanners, vídeos e impressoras, usam uma tecnologia matricial, também denominada tecnologia raster. Esses dispositivos possuem uma memória na qual é composta a imagem a ser posteriormente exibida no dispositivo. (AZEVEDO, 2003, p.33)

Um processador de imagem raster (RIP) é o componente utilizado para produzir uma imagem raster (pixels ou pontos) também conhecida como bitmap. Esta última é então enviada para um dispositivo onde a mesma vai ser exibida.

Um bitmap é uma estrutura de dados matricial disposta em uma grade retangular de pixels, ou pontos de cor, que podem ser visualizadas através de um monitor, papel ou outro meio de exibição. As imagens raster são armazenadas em arquivos de imagens com formatos variados.

De acordo Eduardo Azevedo (2003), um bitmap corresponde à representação binária de uma imagem a ser exibida em uma tela, geralmente no mesmo formato utilizado para o armazenamento em memória de vídeo. Um mapa de bits é tecnicamente caracterizado por uma largura e altura da imagem em pixels e pelo número de bits por pixel (são usados bits adicionais para representar as cores de um pixel).

Um bitmap pode ser monocromático, em escala de cinza ou colorido. Normalmente os pixels são formados no padrão RGB (do inglês: Red, Green, Blue), que utiliza três números inteiros para representar as cores vermelho, verde e azul ou RGBA, quando o formato possui transparência (sendo A o nível de alfa de cada pixel). Para mídias impressas, as imagens bitmap ou raster utilizam o modo de cor CMYK (Ciano, Magenta, Amarelo e Preto).

Na figura 5 temos um bitmap RBG. Observe que cada pixel é formado pela

combinação de valores RBG.

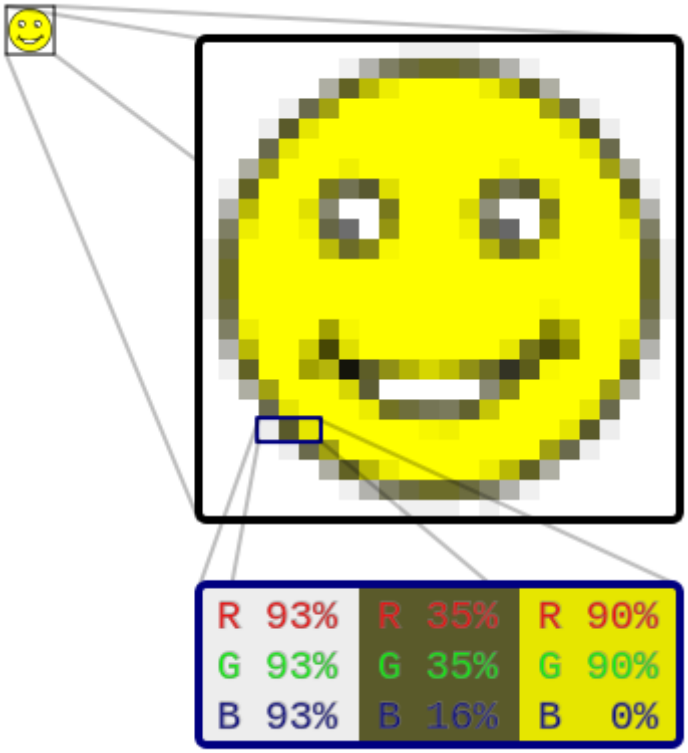


Figura 5 - Imagem no formato bitmap
Fonte: (http://commons.wikimedia.org/wiki/File:Rgb-raster-image.svg).

No RIP, a entrada pode ser uma descrição da página em um alto nível, chamada de linguagem de descrição de página, como PostScript, Portable Document Format, XPS ou outro bitmap de maior ou menor resolução que o dispositivo de saída. Neste último caso, aplica-se um alisamento ou algoritmos de interpolação para gerar, a partir do mapa de bits de entrada, outro mapa de bits de saída com a resolução correta.

Basicamente, o processamento de imagem raster é o processo de transformar um vetor digital de informações, como um arquivo PostScript, em uma imagem raster de alta resolução.

Os estágios de um processador RIP são:

- Interpretação: Fase onde as PDLs suportadas (linguagens de descrição de página) são convertidas para uma representação interna privada de

cada página. A maioria dos RIPs processa páginas em série de modo que o estado atual da máquina é apenas para a página atual, ou seja, uma página de uma vez. Visto que a página atual tenha sido processada, o sistema é preparado para a próxima página.

- Rendering: Um processo através do qual a representação interna privada de uma página é transformada em um bitmap de tom contínuo. Na prática, a interpretação e renderização são frequentemente feitos em conjunto. Linguagens simples foram projetadas para funcionar no próprio hardware, acionando diretamente o renderizador.

- Screening: Para exibição, um bitmap de tom contínuo é convertido em um meio-tom (padrão de pontos). Dois métodos de exibição ou tipos são Amplitude Modulada (AM) e estocástica ou Frequência Modulada (FM). Em exibição AM, o tamanho do ponto varia dependendo da densidade do objeto - valores tonais; pontos são colocados em uma grade fixa. Na FM, o tamanho dos pontos permanece constante e estes são colocados aleatoriamente para criar áreas mais escuras ou mais claras da imagem. O posicionamento de pontos é precisamente controlado por algoritmos matemáticos sofisticados.

A figura 6 apresenta o diagrama do funcionamento básico de um processador RIP.

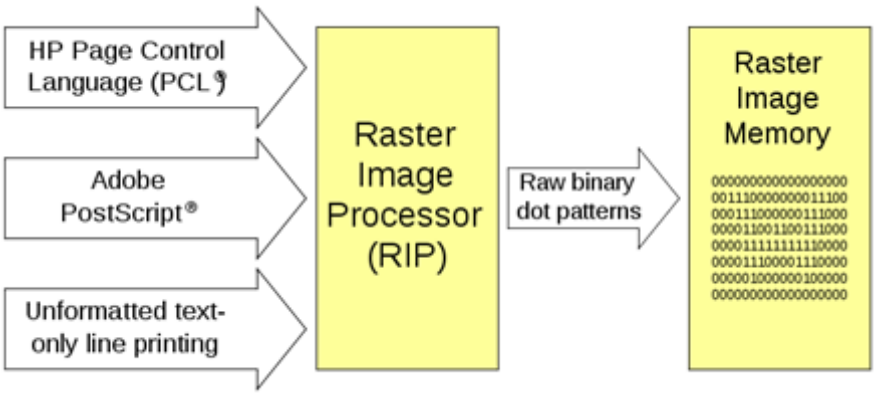


Figura 6 - Funcionamento de processador RIP
Fonte: (https://commons.wikimedia.org/wiki/File:RIP_Data_Flow.svg).

2.4 MATRIZES DE LED

Esta matriz é uma estrutura básica de um painel de LED onde os LEDs são dispostos. Ao controlar o fluxo de eletricidade através das colunas e linhas, é possível controlar cada LED individual. Ao digitalizar as linhas da matriz rapidamente é possível criar caracteres ou imagens para serem exibidas a um usuário. Para conseguir tal efeito é necessário que a taxa de atualização das linhas seja rápida o suficiente para evitar que o olho detecte a transição das mesmas.

2.4.1 O LED

O LED (Light Emitting Diodo - Diodo Emissor de Luz) é um componente importantíssimo no mundo da eletrônica. Sua principal funcionalidade é a emissão de luz em equipamentos eletrônicos, sejam eles produtos de microeletrônica, como sinalizador de avisos, ou em algum equipamento maior, como um painel de LED. Trata-se de um componente feito com materiais semicondutores (mesma tecnologia utilizada nos chips de computadores), que tem a propriedade de transformar energia elétrica em luz. Tal transformação é diferente da encontrada nas lâmpadas convencionais que utilizam filamentos metálicos, radiação ultravioleta e descarga de gases, dentre outras. Nos LEDs, a transformação de energia elétrica em luz é feita na matéria, sendo, por isso, chamada de Estado Sólido (Solid State).

Na estrutura de um LED encontramos algo chamado diodo que nada mais é um tipo simples de semicondutor. De modo geral, um semicondutor é um material com capacidade variável de conduzir corrente elétrica. A maioria dos semicondutores é feita de um condutor pobre que teve impurezas (átomos de outro material) adicionadas a ele.

O processo de adição de impurezas é chamado de dopagem. Um semicondutor com elétrons extras é chamado material tipo-N, uma vez que tem partículas extras carregadas negativamente. No material tipo-N, elétrons livres se movem da área

carregada negativamente para uma área carregada positivamente.

Um semicondutor com buracos extras é chamado material tipo-P, uma vez que ele efetivamente tem partículas extras carregadas positivamente. Os elétrons podem pular de buraco em buraco, movendo-se de uma área carregada negativamente para uma área carregada positivamente. Como resultado, os próprios buracos parecem se mover de uma área carregada positivamente para uma área carregada negativamente.

Um diodo é composto por uma seção de material tipo-N ligado a uma seção de material tipo-P, com eletrodos em cada extremidade. A figura 7 (a) dá uma ideia do arranjo dos semicondutores do tipo n e p para formar um diodo e em 7 (b) o símbolo que é usado para representar o diodo.

Figura 7 - (a) arranjo de um diodo (b) símbolo que representa o diodo
Fonte: (http://www.mspc.eng.br/eletrn/semic_210.shtml).

A polarização do diodo é dependente da polarização da fonte geradora. A polarização é direta quando o pólo positivo da fonte geradora entra em contato com o lado do cristal P (chamado de ânodo) e o pólo negativo da fonte geradora entra em contato com o lado do cristal N (chamado de cátodo). Um diodo só permite a condução de eletricidade quando está polarizado de forma direta, permitindo o fluxo de eletricidade em um só sentido. A figura 8 apresenta uma polarização direta de um diodo.

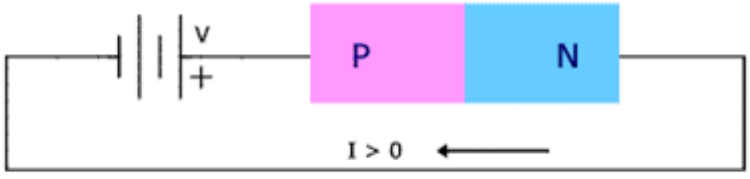


Figura 8 - Diodo polarizado diretamente
Fonte: (http://www.mspc.eng.br/eletrn/semic_210.shtml).

Quando polarizado diretamente, um diodo tem a capacidade de irradiar energia. Nos diodos comuns essa energia é dissipada em forma de calor. Num LED, a energia é irradiada em forma de luz. Os LEDs substituem as lâmpadas incandescentes em várias aplicações por causa de sua baixa tensão, longa vida e por terem um bom funcionamento em circuitos de chaveamento (liga-desliga).

Os diodos comuns são feitos de silício, um material opaco que bloqueia a passagem da luz. Os LEDs são diferentes. Pelo uso de elementos como o gálio, arsênico e fósforo, um fabricante pode produzir LEDs que irradiam as luzes vermelha, verde, amarela, azul, laranja ou infravermelha (luz invisível). Os LEDs que produzem irradiação de luz visível são úteis nos instrumentos como calculadoras, relógios, celulares, etc. O LED infravermelho encontra aplicações nos sistemas de alarme contra ladrão e outras áreas que necessitam de irradiação infravermelha. (MALVINO, Albert Paul, 1997, p 161)

Os materiais usados na fabricação de diodo e transistores, como silício e germânio, a maior parte da energia é liberada em forma de calor, sendo insignificante a luz emitida. No LED, os materiais utilizados para a construção são elementos como o arsenieto de gálio ou fosfeto de gálio, onde o número de fótons de luz emitidos é suficiente para constituir fontes de luz bastante eficientes.

A cor que um LED possui depende do cristal e da impureza usada na dopagem do semicondutor. Aqueles que utilizam o arsenieto de gálio emitem uma radiação infravermelha, e se forem dopados com fósforo a emissão pode ser vermelha ou amarela, dependendo da concentração. Quando se faz uso do fosfeto de gálio dopado com nitrogênio será emitida luz verde ou amarela. Os LEDs ainda podem emitir luz azul, violeta e branca, utilizando fósforo, como nas lâmpadas fluorescentes, onde a luz azul é adsorvida e a luz branca é emitida. Na figura 9 pode ser visto a estrutura básica de LED.

Figura 9 - Representação da estrutura básica de um diodo LED
Fonte: (http://oficinabrasilvirtual.blogspot.com.br/2010/12/led-diodo-emissor-de-luz.html).

2.4.2 Como Funciona uma Matriz de LED

Uma matriz é um arranjo de elementos, neste caso, os LEDs, dispostos em uma grade de linhas e colunas. Trata-se de uma topologia elétrica, que fisicamente pode ser organizada em qualquer padrão ou grade. Com este arranjo de matriz podemos controlar um grupo maior de LEDs com apenas alguns pinos de saída. O número de LEDs que se pode controlar é o produto do número de linhas e colunas. Ou seja, em uma matriz com 8 colunas e 8 linhas, poderíamos controlar o total de 8 X 8 = 64 LEDs.

Na figura 10 temos dois diagramas esquemáticos de matrizes de LED 8x8. Como pode ser observado, os LEDs de uma determinada linha são interconectados por um terminal específico dos LEDs. Da mesma maneira, os LEDs de uma determinada coluna são interconectados por um terminal específico dos LEDs. Os LEDs destas matrizes podem ser vistos como as coordenadas X e Y.

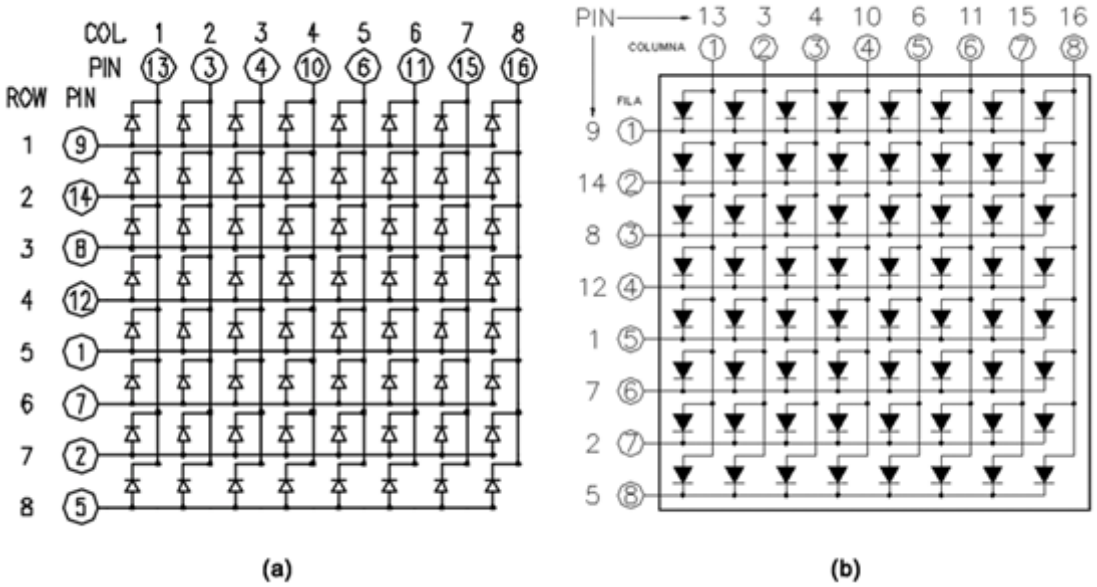


Figura 10 - Diferentes configurações de matrizes de LED
Fonte: Autoria própria.

As Matrizes LED podem ser de dois tipos: Ânodo comum ou cátodo comum. Na figura 10 temos dois diagramas esquemáticos de duas matrizes 8x8, sendo a figura 10(a) do tipo ânodo comum e a figura 10(b) do tipo cátodo comum.

A diferença entre essas duas configurações de matrizes de LED é como se pode acender um determinado LED. Na matriz de ânodo comum os LEDs são acionados aplicando uma tensão positiva em suas linhas e uma negativa em suas colunas. Já nas matrizes de cátodo comum os LEDs são acionados aplicando uma tensão negativa nas linhas e uma positiva em suas colunas. Por exemplo, para acender o LED (4,4) da matriz de cátodo comum (Figura 10b), seria necessário aplicar uma tensão positiva na coluna 4 e uma negativa na linha 4.

O motivo de as linhas e as colunas serem todas conectadas é para minimizar o número de pinos necessários. Se não o fizéssemos, uma única unidade de matriz de pontos 8x8 necessitaria de 65 pinos: 64 para cada LED e um para um ânodo ou cátodo comum. Ligando a fiação das linhas e colunas, são necessários apenas 16 pinos.

Entretanto, isso representa um problema se você deseja que um LED específico acenda em certa posição. Caso, por exemplo, tivéssemos uma matriz de ânodo comum (Figura 10a) e quiséssemos acender um o LED na posição X, Y de valores 5,3 (quinta coluna, terceira linha), aplicaríamos uma corrente à terceira linha e o terra

ao pino da quinta coluna. O LED correspondente acenderia conforme a figura 11.

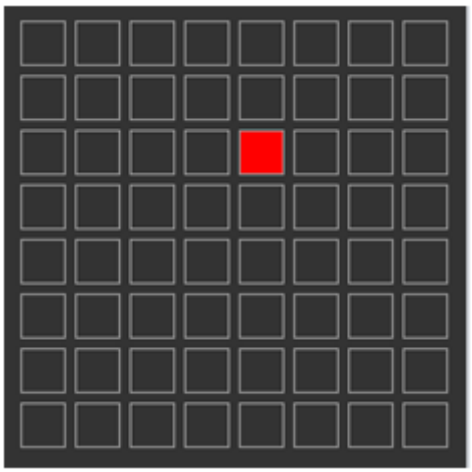


Figura 11 - LED na quinta coluna e na terceira linha acesso
Fonte: Autoria própria.

Agora, imagine que quiséssemos acender também o LED da coluna 3, linha 5. Então aplicaríamos uma corrente à quinta linha e o terra ao pino da terceira coluna. O LED correspondente agora seria iluminado. Mas também acenderia os LEDs da coluna 3, linha 3 e coluna 5, linha 5, causando uma inconveniência vista na figura 12.

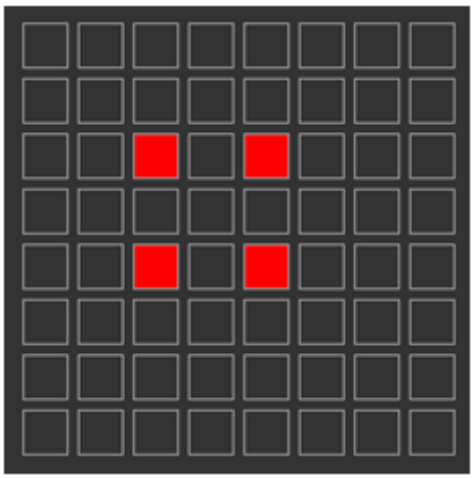


Figura 12 - LEDs acesso nas posições 5,3; 3,5; 3,3; 5,5
Fonte: Autoria própria.

Isso ocorre porque está sendo aplicado uma tensão positiva nas linhas 3 e 5, e uma negativa nas colunas 3 e 5. Deste modo não poderíamos apagar os LEDs indesejados sem apagar também aqueles que devem estar acessos. Esse problema poderia ser resolvido se cada LED possuísse uma pinagem separada, mas como já foi citado, faria com que o número de pinos pulassem de 16 para 65. Uma matriz de LED com 65 pinos teria uma fiação muito complexa e seria muito difícil de controlar, pois você necessitaria de uma microcontrolador de ao menos 64 saídas digitais.

Para solucionar esse problema nas matrizes de LED, faz-se uso do conceito de multiplexação.

2.4.2.1 Multiplexação Nas Matrizes De LED

Como citado acima, os LEDs são dispostos em uma grade de linhas e colunas possibilitando uma diminuição na quantidade de número de pinos necessários para controlar a matriz de LED. Entretanto essa estrutura apresenta um problema quando tentamos acender um LED individual sem que os outros da mesma coluna também acendam. Para resolver esta questão fazemos o uso de multiplexação.

Multiplexação é processo de codificar informações de duas ou mais fontes de dados num único canal. Este conceito se aplica perfeitamente nas matrizes de LED, pois a informação necessária para acionar os LEDs é transmitida por uma quantidade de fios menor do que a quantidade de terminais existentes na matriz. Por exemplo, em uma matriz de 8x8 (figura 10), devemos transmitir a informação de 64 LEDs por 16 pinos de conexões.

A multiplexação em matrizes de LED é uma técnica de acender uma linha de cada vez. Selecionando a coluna que contém a linha, que, por sua vez, contém o LED que se deseja acender, e ligando a alimentação para essa linha (ou da forma oposta, para matrizes de cátodos comuns), os LEDs escolhidos nessa linha serão iluminados. Essa linha será, então, apagada, e a próxima acessa, novamente com as colunas apropriadas escolhidas, e fazendo com que os LEDs da segunda linha agora sejam iluminados. Esse processo é repetido em todas as linhas até chegar em sua base, e, então, o processo volta ao topo.

Se isso for feito a uma velocidade suficiente (30 Hz, ou 30 vezes por segundo ou mais), o fenômeno de persistência da visão (em que uma pós-imagem permanece na retina por, aproximadamente, 1/25 de um segundo) fará com que o painel de LED se acender por inteiro, mesmo que cada linha acenda e apague em sequência.

Utilizando essa técnica, pode-se solucionar o problema de exibição de LEDs individual, sem que outros LEDs na mesma coluna também acendam.

Na figura 13 podemos ver esse processo de multiplexação usado na exibição do numeral ‘2’ em matriz 8x8.

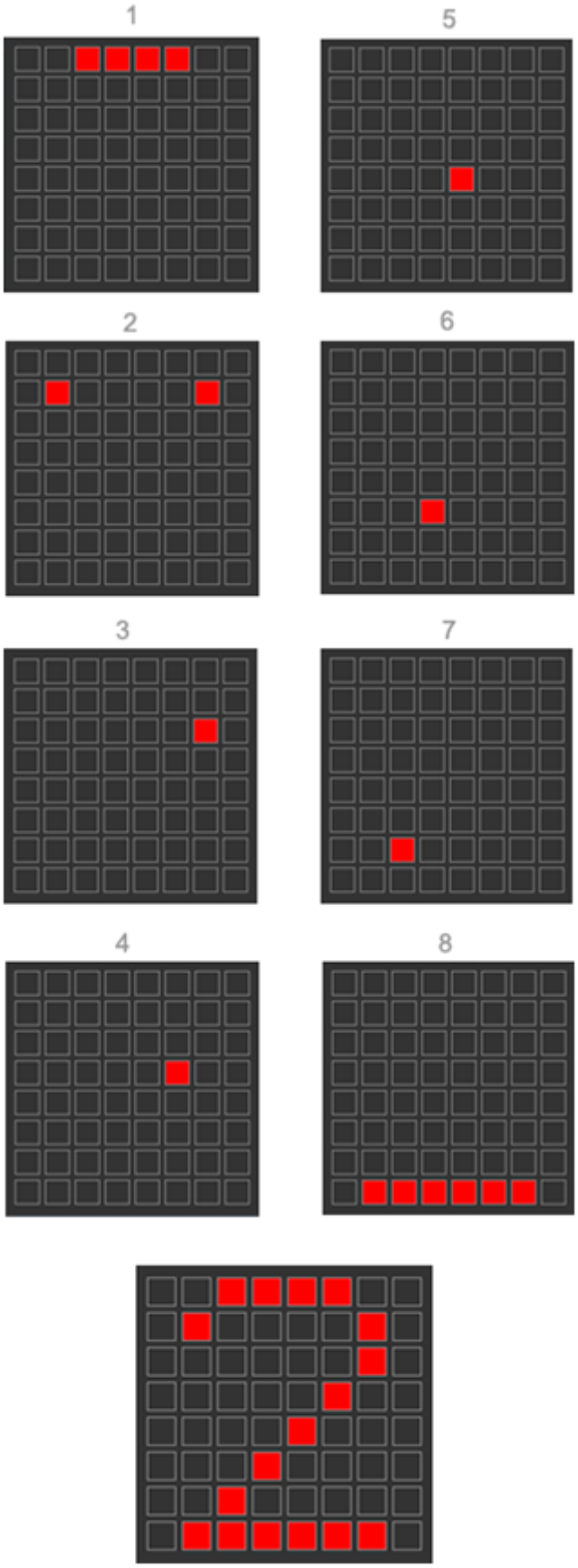


Figura 13 - Processo de multiplexação usado na formação do numeral 2 (dois)
Fonte: Autoria própria.

2.4.3 Matriz De Led Disponíveis No Mercado

Podemos encontrar no mercado diferentes tipos de matrizes de LED, variando o preço de acordo com: sua resolução, as cores dos LEDs (monocromática, bicolor ou RBG) e pela potencia dos seus LEDs.

Para este projeto vamos usar módulos de matriz ânodo comum com 64 LEDs (8x8) de cor vermelha. Na figura 14 pode ser visto o aspecto físico da matriz, e na figura 15 seu datasheet.

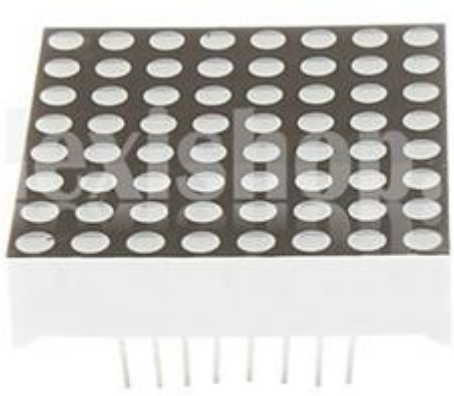


Figura 14 - Matriz de LED LD1088BS
Fonte: (http://pt.aliexpress.com/item/5pcs-8x8-Dot-Matrix-1-9mm-Diameter-Red-LED-Display-free-shipping/594892927.html).

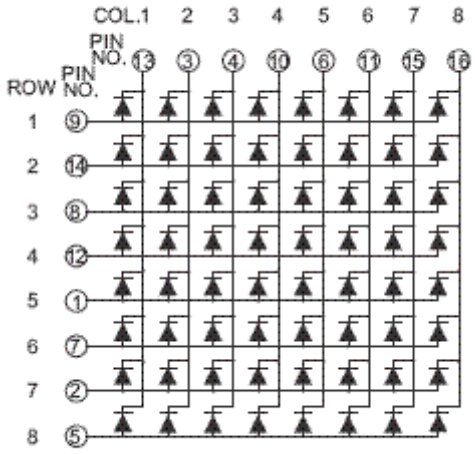


Figura 15 - Datasheet da atriz de LED LD1088BS
Fonte: (http://pt.aliexpress.com/item/5pcs-8x8-Dot-Matrix-1-9mm-Diameter-Red-LED-Display-free-shipping/594892927.html).

2.5 REGISTRADORES DE DESLOCAMENTO (SHIF-REGISTER)

Como caracterizado por Newton Braga (2013), em eletrônica digital os registradores de deslocamento (shift-register) são o resultado da utilização de um conjunto de registradores de uma forma especial de maneira que a informação é deslocada pelo circuito conforme o mesmo é ativado. Este conjunto de flip-flops é ligado em cascata partilhando o mesmo clock, onde a saída Q de dados de cada flip-flop está ligada à entrada D do próximo flip-flop, conforme pode ser visto na figura 16 utilizando flip-flops do tipo D (figura 16 a) e tipo JK (figura 16 b).

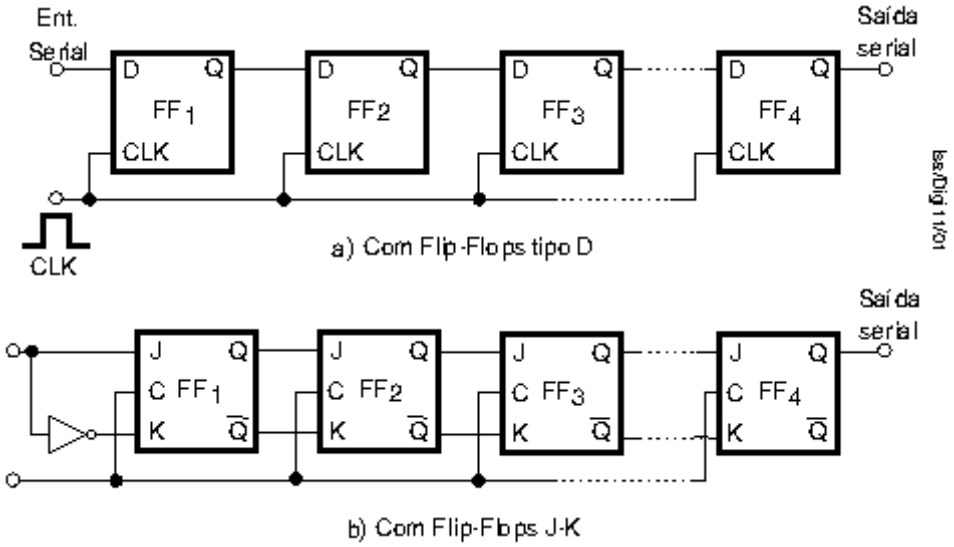


Figura 16 - Registradores de deslocamento com flip-flops D e J-K.
Fonte: (BRAGA, 2013, p.86).

Estes circuitos podem deslocar uma informação (bit), aplicada na entrada, em uma posição a cada pulso de clock. Por exemplo, o bit 1 aplicado na entrada aparece na saída do primeiro flip-flop no primeiro pulso de clock, depois desloca-se, aparecendo na saída do segundo flip-flop no segundo pulso de clock e assim por diante, até aparecer na saída do final da sequência, figura 17

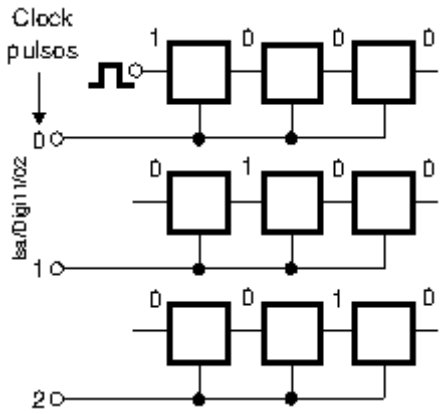


Figura 17 - Deslocamento dos bits pelos flip-flops do registrador
Fonte: (BRAGA, 2013, p.86).

Na configuração mostrada na figura 16(a), cada flip-flop tipo D tem sua saída conectada à entrada do flip-flop seguinte e todos eles são controlados pelo mesmo

clock.

Para entender como funciona este circuito, vamos partir da situação inicial em que todos eles estejam desativados ou com suas saídas Q no nível baixo.

Inicialmente vamos aplicar à entrada de dados um nível alto (1). Conforme podemos ver, esta entrada é feita pela entrada J do primeiro flip flop (FF1).

Com a chegada do pulso de clock a este flip-flop, ele muda de estado e com isso “armazena” o pulso aplicado à entrada, o qual aparece em sua saída depois de um curto intervalo de tempo. Veja que este sinal é armazenado com o flanco positivo do sinal de clock, quando então o nível alto deve estar presente na entrada do flip-flop.

O intervalo de tempo, que decorre entre a aplicação do sinal na entrada de dados e seu aparecimento na saída do flip-flop, é da ordem de alguns nanossegundos nos circuitos integrados das famílias lógicas comuns. Mas é importante que em muitas aplicações mais rápidas ele seja levado em conta.

No próximo pulso de clock, ocorre algo interessante: a entrada do primeiro flip-flop já não tem mais o nível alto, e, portanto FF1 não muda de estado. No entanto, na saída de FF1, temos nível alto, e esta saída está ligada à entrada do segundo flip-flop (FF2).

Isso significa que, com a chegada do segundo pulso de clock, o nível lógico da saída do primeiro se transfere para a saída do segundo, depois é claro, de um pequeno intervalo de tempo, conforme a figura 18.

clock	entrada	FF1	FF2	FF3	FF4	Saída
0	1	0	0	0	0	0
1	1	1	0	0	0	0
2	A 0	1	1	0	0	0
3	0	0	1	1	0	0
4	1	0	0	1	1	0
5	0	1	0	0	1	1
6	0	0	1	0	0	1
7	0	0	0	1	0	B 0
8	0	0	0	0	1	0
9	0	0	0	0	0	1

Figura 18 - Deslocamento dos bits pelos flip-flops do registrador
Fonte: (BRAGA, 2013, p.87).

A sequência de bits aplicados à entrada (a) aparece na saída (b) depois de certo número de clock. Isso significa que o bit 1 aplicado na entrada se “deslocará” no circuito, passando para a saída do segundo flip-flop. É claro que, se nessa segunda passagem, tivermos aplicado um novo nível 1 na entrada do circuito, ao mesmo tempo que o primeiro se transfere para o segundo flip-flop, o segundo se transfere para a saída do primeiro flip-flop. Veja a figura 19.

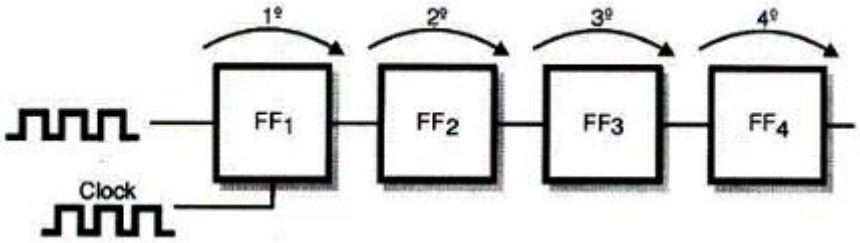


Figura 19 - Sequência de transferências dos bits conforme os pulsos de clock
Fonte: (BRAGA, 2013, p.88).

Chegando agora um terceiro pulso de clock, teremos nova transferência e o nível alto ou bit 1 se transfere para a saída do flip-flop seguinte, ou seja, FF3.

Em outras palavras, a cada pulso de clock, os níveis existentes nas saídas dos flip-flops, sejam eles 0 ou 1, se transferem para o flip-flop seguinte.

Assim, supondo que apliquemos, em sequência, na entrada de um shift register como o indicado, os níveis 0101, teremos a seguinte sequência de condições de saída para os flip flops de um shift-register que use 4 deles, veja a tabela abaixo.

Tabela 2 - Sequência de condições de saída para os flip flops

Clock	Entrada	FF1	FF2	FF3	FF4
inicio	0	0	0	0	0
0	1	0	0	0	0
1	0	1	0	0	0
2	1	0	1	0	0
3	0	1	0	1	0
4	0	0	1	0	1

Fonte: (BRAGA, 2013, p.88).

Podemos perceber então que no quinto pulso de clock, o primeiro nível lógico referente ao primeiro pulso de clock, aparece na saída do último flip-flop (FF4) e que, se lermos a saída dos flip-flops teremos registrado os níveis aplicados na entrada: 0101.

Conclui-se que aplicando um dado binário num shift register, depois do número apropriado de pulsos de clock, ele pode armazenar este dado. Para retirar o dado em sequência, basta continuar aplicando pulsos de clock ao circuito, conforme a tabela a seguir.

Tabela 3 - Retirando dados em sequência de registrador de deslocamento

clock	FF1	FF2	FF3	FF4	Saída
início(4)	0	1	0	1	1
5	0	0	1	0	0
6	0	0	0	1	1
7	0	0	0	0	0

Fonte: (BRAGA, 2013, p.88).

A figura 20 mostra o que ocorre de maneira simplificada:

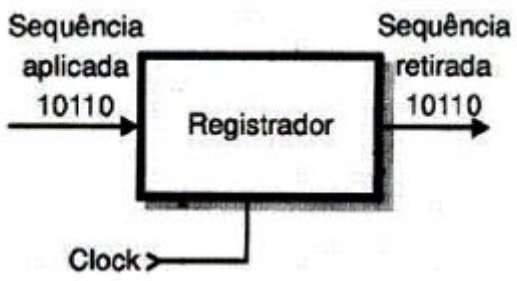


Figura 20 - Entrada e saída de dados num registrador de deslocamento
Fonte: (BRAGA, 2013, p.88).

Então para armazenar um dado de 4 bits num registrador devemos aplicar 4 pulsos de clock. E para ler em sequência, mais 4 pulsos de clock.

Para “apagar” os dados registrados em um registrador de deslocamento, como o indicado, basta aplicar um pulso na entrada CLEAR. Todos os flip-flops terão suas

saídas levadas ao nível baixo ou 0.

2.5.1 Tipos De Registradores De Deslocamento

Dependendo da maneira como a informação entra e como ela pode ser obtida num registrador de deslocamento, podemos ter diversas configurações que nos levam a muitos tipos de circuitos.

Assim, existem circuitos em que temos uma entrada serial ou duas, e também podemos ter uma ou duas linhas de saída.

A seguir, veremos os principais tipos, bem como suas denominações.

2.5.1.1 SISO - Serial-in/Serial-out

No exemplo citado acima, os dados foram aplicados à entrada do registrador na forma de níveis lógicos um atrás do outro, acompanhando o sinal de clock. Dizemos que este registrador opera com a carga de dados “serial” ou em série. Em outras palavras, este circuito tem entrada serial ou serial-in.

Exatamente como ocorre com a porta serial de um computador, os dados são “enfileirados” e entram um após outro e vão sendo armazenados em flip-flops, conforme o circuito da figura 21.

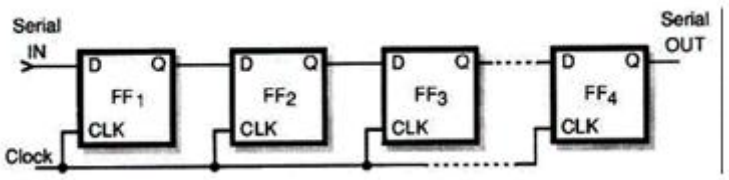


Figura 21 - Os dados são enfileirados na entrada e saem enfileirados.
Fonte: (BRAGA, 2013, p.89).

2.5.1.2 PISO - Parallel-in/Serial-out

Outra possibilidade de operação para os registradores de deslocamento é a de operar com a entrada de dados em paralelo e saída de dados em série. Dizemos que se trata de um registrador de deslocamento com entrada paralela e saída serial.

Na figura 22 temos um diagrama que usa 4 flip-flops tipo D e que tem entrada de dados paralela e saída serial.

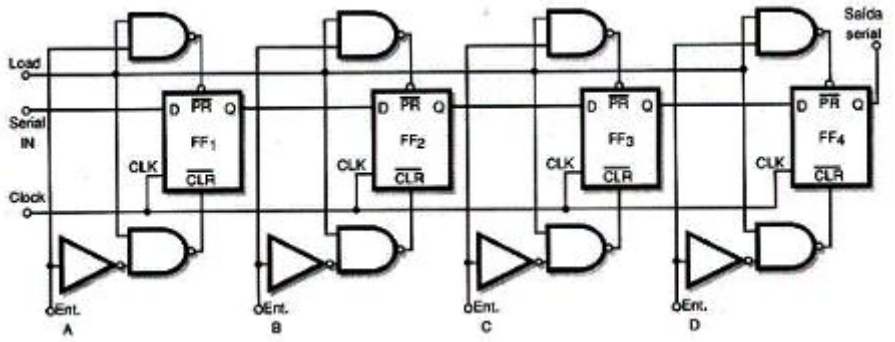


Figura 22 - Os dados são enfileirados na entrada e saem enfileirados.
Fonte: (BRAGA, 2013, p.89).

Analisemos como ele funciona:

Os dados são colocados ao mesmo tempo na entrada, pois ela opera em paralelo. Por exemplo, se vamos armazenar o dado 0110, esses dados são aplicados ao mesmo tempo nas entradas correspondentes dos flip flops.

No primeiro pulso de clock, os flip flops “armazenam” esses dados. Assim, os flip-flops que possuem nível 1 em sua entrada passam esse nível à saída (FF2, FF3). Por outro lado, os que possuem nível 0 na sua entrada, mantém este nível na saída (FF1 e FF4).

Isso significa que, após o pulso de clock, as saídas dos flip-flops apresentarão os níveis 0110.

2.5.1.3 SIPO - Serial-In/Parallel-out

Da mesma forma, como verificamos na figura 23, podemos carregar os dados em

série e fazer sua leitura em paralelo.

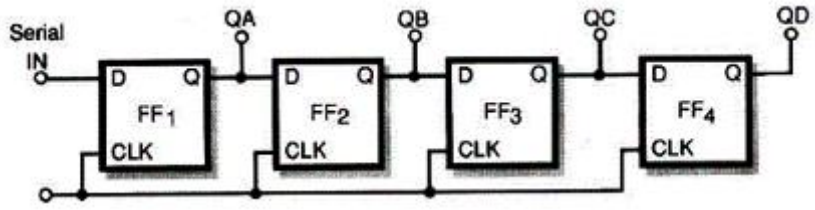


Figura 23 - Os dados são enfileirados na entrada e saem enfileirados.
Fonte: (BRAGA, 2013, p.90).

Os registradores que operam desta forma podem ser também denominados conversores série-paralelo ou paralelo-série, conforme o modo de funcionamento.

Esse tipo de registrador é muito importante na transmissão de dados através de meios físicos (transmissão serial), já que ele pode fazer sua conversão para forma digital normal de dados que chegam serialmente, transmitidos por um modem.

2.5.1.4 PIPO - Parallel-in/Parallel-out

Estes são circuitos em que os dados são carregados ao mesmo tempo e depois lidos ao mesmo tempo pelas saídas dos flip-flops, veja a figura 24.

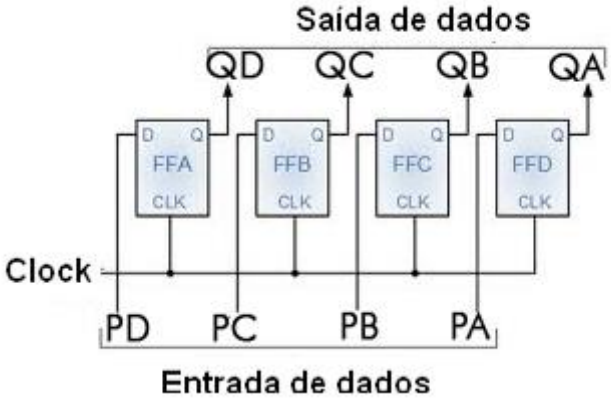


Figura 24 - Os dados são enfileirados na entrada e saem enfileirados.

Fonte: (BRAGA, 2013, p.91).

Os registradores de deslocamento podem ainda ser classificados quanto à direção em que os dados podem ser deslocados.

Dizemos que se trata do tipo Shift-Right, quando os dados são deslocados para a direita e que se trata de um tipo Shift-Left, quando os dados são deslocados somente para a esquerda.

Existem ainda os tipos bidirecionais, como o mostrado na figura 25, em que os dados podem ser deslocados nas duas direções. Este é um registrador do tipo SISO.

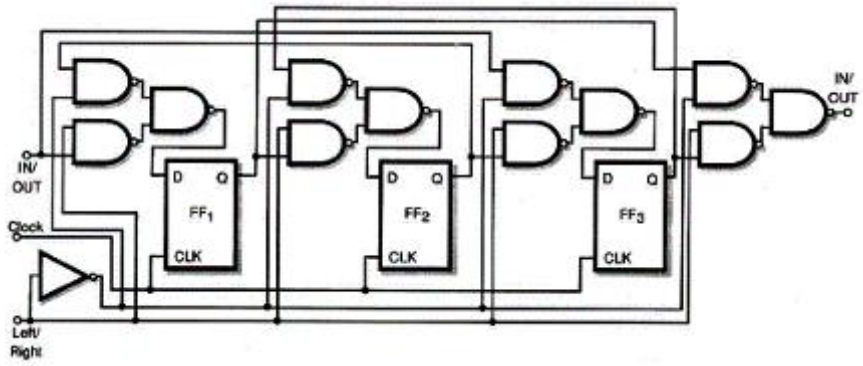


Figura 25 - Os dados são enfileirados na entrada e saem enfileirados.
Fonte: (BRAGA, 2013, p.91).

O sentido de deslocamento é determinado por uma entrada que atua sobre portas que modificam o ponto de aplicação dos sinais em cada flip-flop, exatamente como funciona os contadores up e down.

Com a aplicação de um nível lógico conveniente na entrada LEFT/ RIGHT, podemos determinar o sentido de deslocamento dos dados no circuito.

2.5.2 Operando Com Binários

Pode-se perceber que os registradores de deslocamento podem memorizar

números binários, recebendo-os em série ou paralelo e entregando-os depois em série ou paralelo.

Nos computadores, esta configuração é bastante usada tanto na conversão de dados de portas como nas próprias memórias e outros circuitos internos.

É interessante observar que na configuração que tomamos como exemplo, em que são usados 4 flip flops, os bits armazenados seguem uma determinada ordem.

Assim, quando representamos o número 5 (0101), cada um dos bits tem um valor relativo, que depende da sua posição no dado. MSB significa bit mais significativo, ou seja, de maior peso, enquanto que LSB significa bit menos significativo ou de menor peso.

Nos exemplos mostrados, estamos trabalhando com dados de 4 bits, e não 8, como é comum nos computadores, obtendo assim o “byte”, para maior facilidade de entendimento. Ligando então 4 flip-flops de modo a obter um shitf-register, como observamos na figura 26, e entrando com os dados de tal forma que o bit menos significativo (LSB) seja o primeiro, depois de 4 pulsos de clock, ele vai aparecer, na saída do último flip-flop.

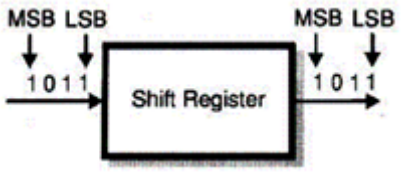


Figura 26 - A ordem de entrada é a ordem de saída.
Fonte: (BRAGA, 2013, p.92).

Da mesma forma, se o registrador de deslocamento for carregado em paralelo, o bit menos significativo (LSB) deve entrar no último, de modo que na leitura ele seja o primeiro a sair.

Nos projetos que usam shift-register, a ordem de entrada e saída dos bits é muito importante para se obter o funcionamento correto do aparelho.

2.5.3 Registradores De Deslocamento Integrados

Podemos encontrar registradores de deslocamento nas famílias TTL ou CMOs. A seguir alguns exemplos de circuitos integrados comuns que podem ser usados em projetos, analisando suas principais características.

2.5.3.1 7495 - Shift-Register De 4 Bits (Da Esquerda Para A Direita - Entrada E Saída Em Paralelo)

Este circuito integrado TTL pode operar de duas formas: Shift ou Load. Na figura 27 temos sua pinagem.

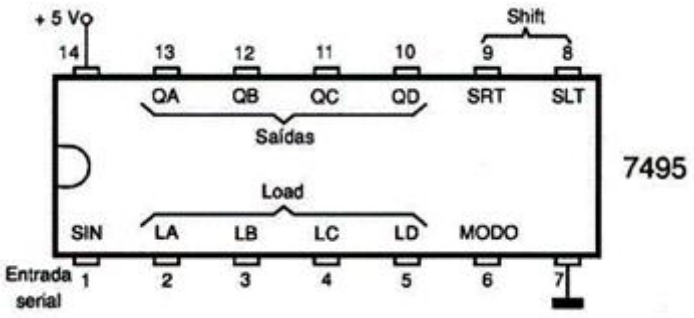


Figura 27 - Shift-register de 4 bits (PIPO).
Fonte: (BRAGA, 2013, p.93).

Para operar no modo shift, basta colocar a entrada MODE no nível baixo. Uma transição do nível alto para o nível baixo na entrada de clock SRT movimenta os dados de uma etapa para a direita.

Uma transição do nível alto para o baixo na entrada SLT movimenta o dado no sentido inverso. É interessante observar que este circuito usa dois clocks, um para movimentar os dados para a direita e outro para a esquerda.

No modo Load, esta entrada deve ir ao nível alto, e a informação carregada nas entradas LA, LB, LC e LD entram no circuito na transição do nível alto para o baixo da entrada de comando na entrada shift-left (SLT).

A frequência máxima de operação de um 7495 standard é de 36 MHz. Velocidades maiores de operação podem ser conseguidas com os tipos LS.

2.5.3.2 74164 - Shift-Register De 8 Bits (Entrada Serial, Saída Paralela)

Na figura 28 temos a pinagem deste shift register TTL.

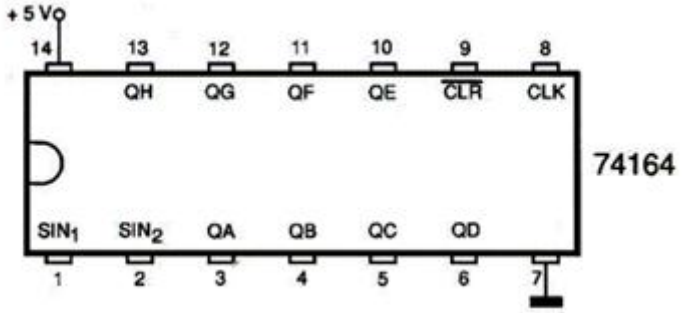


Figura 28 - Shift-Register De 8 Bits (Entrada Serial, Saída Paralela)
Fonte: (BRAGA, 2013, p.94).

Este circuito pode ser usado na configuração de serial-in/serial-out ou serial in/parallel-out, ou seja, entrada e saída de dados em série, ou entrada de dados em série e saída em paralelo.

Na operação normal, uma das saídas seriais é mantida no nível alto e os dados são aplicados à segunda entrada serial. A entrada Clear é mantida no nível alto e, a cada pulso do nível baixo para o alto do clock os dados movem-se de um estágio no circuito. O conteúdo do shift pode ser zerado levando-se a entrada clear por um instante ao nível baixo. A frequência máxima de operação deste circuito na série Standard é de 36 MHz.

2.5.3.3 74165 - Shift- Register De 8 Bits (Entrada Paralela, Saída Serial)

Este circuito integrado TTL contém um shift-register de 8 bits com entrada paralela e saída de dados serial. A pinagem é mostrada na figura 29.

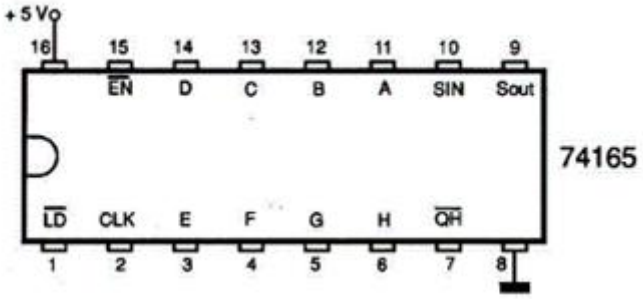


Figura 29 - Shift-register de 8 bits (PISO).
Fonte: (BRAGA, 2013, p.94).

Para operação normal, EN deve ficar no nível baixo e LOAD no nível alto. Nestas condições, os dados são deslocados um estágio na transição positiva do sinal de clock.

Quando a entrada LOAD é levada ao nível baixo, o conteúdo das entradas de A até H é carregado no registrador.

Fazendo EN=0 e LOAD=1 os dados são deslocados uma etapa no circuito a cada transição positiva do sinal de clock. A última etapa do circuito dispõe de um acesso para a saída complementar.

2.5.3.4 4014 - Shift- Register Estático De 8 Bits (Entrada Paralela E Saída Em Série)

Este circuito integrado CMOS tem a pinagem mostrada na figura 30.

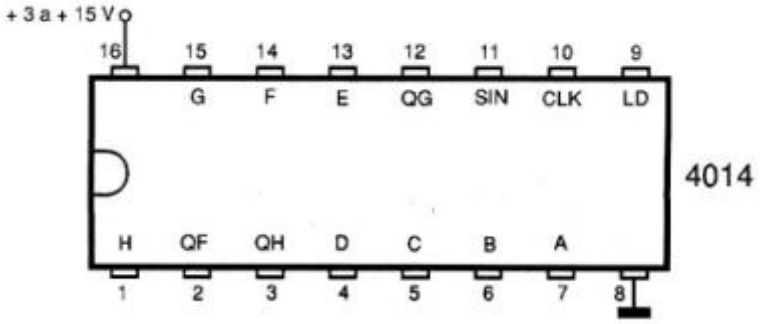


Figura 30 - Shift-register de 8 bits (PISO).
Fonte: (BRAGA, 2013, p.).

Um controle série/paralelo controla a entrada e habilita as etapas individuais de cada um dos 8 estágios. As saídas Q são disponíveis nos estágios 6, 7 e 8. Todas as saídas podem fornecer ou drenar a mesma intensidade de corrente.

Quando a entrada de controle paralelo/série está no nível baixo, os dados são deslocados pelo circuito a cada transição positiva do sinal de clock. Quando a entrada de controle está no nível alto, os dados são aplicados a cada etapa do shift-register com a transição positiva do clock.

A frequência máxima de operação deste tipo de circuito depende da tensão de alimentação. Para uma alimentação de 10 V, esta frequência é da ordem de 5 MHz, caindo para 2,5 MHz com uma alimentação de 5 V.

2.5.3.5 595 - Shift- Register De 8 Bits (Entrada Serial, Saída Serial Ou Paralela

A pinagem deste é mostrada na figura 31

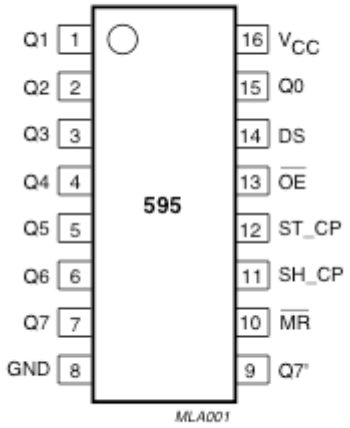


Figura 31 - Datasheet do registrador de deslocamento 595
Fonte: (http://www.arduino.cc/en/Tutorial/ShiftOut).

Este tipo de registrador de deslocamento de 8 bits, com entrada serial, saída serial ou paralela e com latches (travras) de saída. Os dados entram quando o LACTCH está definido como low, e saem como o LATCH está definido como HIGH.

Este circuito opera muito rapidamente, em 100 MHz, ou seja, podemos enviar dados através deste a uma velocidade aproximada de 100 milhões de vezes por segundo. Isso significa que podemos enviar sinais PWM via software para os CIs.

O PWM é recurso necessário para o controle de painéis de LEDs coloridos, por esse motivo, este circuito foi o escolhido pelo autor para ser utilizado neste projeto. Além disso, ele é um circuito fácil de ser encontrado e de baixo preço, cerca de R$ 1,00.

2.5.4 Conversão Série/Paralelo E Paralelo Série

Uma das aplicações mais comuns dos registradores de deslocamento é a conversão de informações da forma paralela para serial e da forma serial para paralela.

Na figura 32 temos um exemplo de como podemos converter uma sequência de bits, transmitidos em série por uma linha, em um conjunto de saídas paralelas que

correspondam exatamente a estes bits.

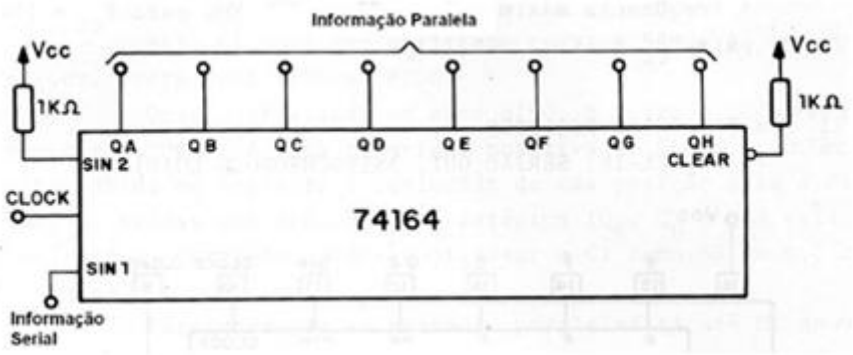


Figura 32 - Convertendo uma informação serial (dados) em paralelo
Fonte: (BRAGA, 2013, p.98).

Este tipo de aplicação pode ser encontrada nos modems, e outros sistemas de transmissão de dados seriais, que recebem um fluxo de bits em uma única linha e que devem ser transferidos para uma saída paralela de 8 linhas, que é o modo de operação dos computadores.

Este circuito emprega um shift-register do tipo SIPO (Entrada serial e saída paralela). Na figura 33 temos o gráfico de tempos que ilustra como ele funciona e de que modo a sequência de dados aparece serialmente na sua saída.

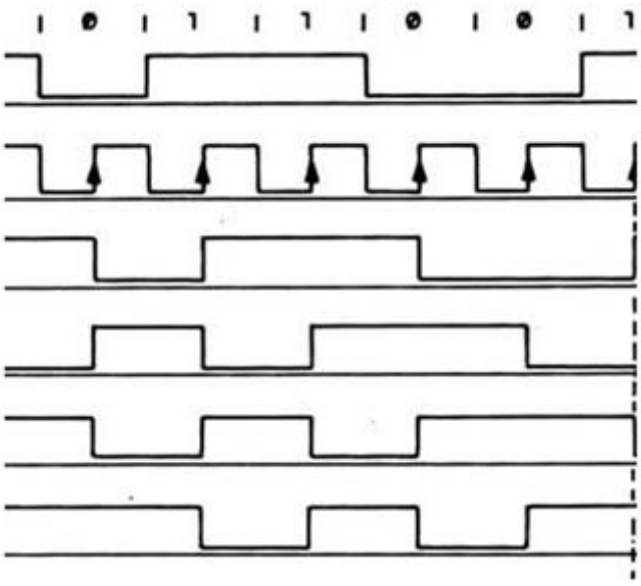


Figura 33 - Sequência de transferência dos sinais no circuito dado como exemplo
Fonte: (BRAGA, 2013, p.98).

Na primeira linha do gráfico de tempos, temos a sequência de níveis lógicos que é aplicada à entrada do circuito e que corresponde justamente à informação serial que devemos tornar paralela. Dois bits de valor 1 seguidos não possuem uma "separação", mas são distinguidos pelo tempo que corresponde a dois ciclos.

Na segunda linha temos o sinal de clock que vai sincronizar a transferência desses sinais ao longo do registrador de deslocamento.

Nas linhas seguintes temos então a carga dos sinais que vão se deslocando pelo shift-register até que ao final do oitavo ciclo do clock o sinal está presente em todas as saídas do circuito na forma paralela.

A transformação de dados da forma paralela para forma serial também é importante em aplicações digitais que envolvem informações na forma digital. Os dados das linhas paralelas dos computadores devem ser serializados para serem transmitidos por uma linha telefônica, por exemplo.

Podemos usar um shift-register como o 74165 para fazer isso. A figura 34 mostra o modo de ligação deste componente.

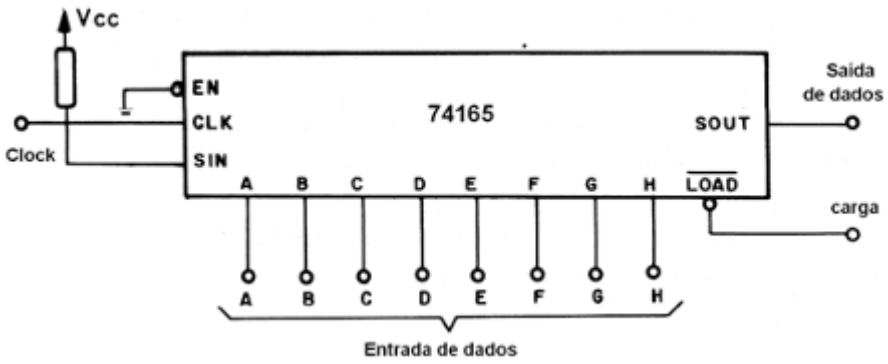


Figura 34 - Usando um 74165 na conversão da forma paralela para serial.
Fonte: (BRAGA, 2013, p.39).

Os sinais são aplicados nas entradas de dados e, com um comando LOAD, eles são carregados no shift-register. Depois disso, a cada pulso de clock os sinais são transferidos, bit por bit, para a saída. A forma de sinais para sequência de dados

01011100 é mostrada na figura 35.

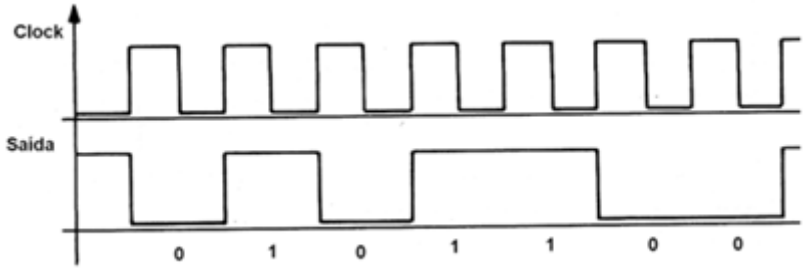


Figura 35 - Conversão da sequência 0101100
Fonte: (BRAGA, 2013, p.99).

2.6 MICROCONTROLADORES

Um microcontrolador é um computador completo dentro de um circuito integrado. “Ele possui em um único encapsulamento, uma unidade central de processamento, memória de programa, memórias auxiliares, sistema de entrada/saída e vários periféricos que variam entre os modelos” (LUPPI e SCUNK, 2001, p. 20).

> O termo microcontrolador é usado para descrever um sistema mínimo que inclui uma CPU, memória e circuitos de entrada e saída, tudo montando num único circuito integrado, que pode ter de 8 a até mais de 100 pinos. Alguns microcontroladores podem vir com contadores decimais internos, conversores analógico-digitais, comparadores de tensão e circuitos de comunicação serial, tudo embutido no mesmo encapsulamento. (SILVEIRA, 2011, p.20)

O microcontrolador é o componente chave para a correta tomada de decisões no sistema de controle do projeto a ser desenvolvido e, graças à capacidade de ser programado, dá ao projeto extrema flexibilidade. Seu uso se faz necessário quando um circuito precisa realizar operações e procedimentos que variam conforme estímulos ou condições de ambiente, comando e procedimento, análises e

correções, etc.

Atualmente o uso de microcontroladores é muito abrangente e comum, pois muitos equipamentos de nosso uso diário (tais como celulares, alarmes entre outros), usam microcontroladores para execução de suas funções básicas.

2.6.1 Modelos Disponíveis No Mercado

Existem no mercado dezenas de fabricantes de microcontroladores, com milhares de modelos disponíveis. Os principais fatores que influenciam a compra de microcontroladores são:

- Tamanho da memória RAM: A memória de acesso randômico (RAM) é utilizada pelo programa principal para executar suas rotinas. Todo processador precisa de uma memória RAM para rodar;
- Capacidade de memória FLASH: Uma vez desenvolvido o programa responsável por controlar o circuito eletrônico desenvolvido, ele precisa ser armazenado permanentemente de forma rápida e eletrônica. Tudo isso é possível graças à memória FLASH, que combina velocidade com gravação através de um meio elétrico, dispensando o uso de luz ultra-violeta;
- Número de entradas e saídas (I/Os): Todo sistema computacional precisa receber dados do mundo externo, processá-los e gerar uma saída. Em função disto, a entrada e saída são feitas através de portas específicas conectadas diretamente ao processador. Em muitos projetos esse fator é determinante;
- Tempo de atuação do fabricante no mercado: Pensando em larga escala, se um projeto foi desenvolvido com uma família específica de microcontroladores e essa linha sai de produção, o projeto precisará ser readaptado, gerando mais custos. Por esse motivo sempre é bom verificar a experiência e o tempo de atuação do fabricante escolhido;
- Suporte ao desenvolvedor: Tão importante quanto o tempo de atuação no mercado é o apoio que o fabricante deve conceder ao desenvolvedor, fornecendo a ele ferramentas que facilitem o desenvolvimento de aplicações usando sua tecnologia.

Entre as principais famílias disponíveis no mercado encontram-se: PIC, AVR E ARM.

2.6.2 A família PIC

Atuando no Brasil desde 1990, a Microchip, fabricante de microcontroladores PIC, já está consolidada e tem seus produtos usados em muitos eletroeletrônicos conhecidos, desde alarmes para casa até no-breaks de computadores.

Em relação à memória interna e I/Os tanto PIC quanto AVR possuem diversos modelos semelhantes, fazendo esses quesitos não influenciarem na escolha de uma ou outra família.

Uma grande diferença em se tratando da medida de velocidade está nos MIPS (milhões de instruções por segundo). Afinal, “Nos microcontroladores PIC, o sinal do clock é internamente dividido por quatro. Portanto, para um clock externo de 4MHz, temos um clock interno de 1 MHz e, consequentemente, cada ciclo de máquina dura 1us (microssegundo)” (SOUSA, 2009, p. 24). Já em um microcontrolador AVR, onde não ocorre a divisão por quatro, com o mesmo clock externo de 4MHz, teríamos 4 ciclo de máquinas por microssegundo.

O Integrated Developer Environment (IDE) da família PIC mais conhecido é o PLAB, que agora roda em Windows, Linux e Mac. Em geral os kits de estudante para esta plataforma custam em torno de R$ 200,00.

2.6.3 A plataforma de Desenvolvimento Arduino

O microcontrolador da família AVR chegou de forma massiva recentemente no Brasil, com o codinome “Arduino”. É uma plataforma de prototipagem eletrônica open-source, baseado em um microcontrolador que se conecta ao computador pessoal através de uma porta serial ou USB, a depender do modelo utilizado. Ele possui uma linguagem de programação própria baseada em Wiring que é similar à linguagem C, e é implementada em um ambiente de desenvolvimento (IDE), também

próprio, baseado em Processing, e pode ser utilizada em vários sistemas.

A placa do Arduino é composta por um microprocessador Atmel AVR, um oscilador, um regulador linear de 5 volts e pinos de entrada/saída para serem conectados a outros circuitos ou sensores. A figura 36 apresenta o Arduino Uno, versão esta aplicada neste trabalho.

Figura 36 - Placa Arduino
Fonte: (http://www.hobbyist.co.nz/?q=taxonomy/term/1).

Essa plataforma como é conhecida, se tornou um conceito entre os estudantes, afinal reúne as melhores características para se começar em um universo novo: Facilidade em desenvolver o “hello world”, preço acessível (o kit inicial custa R$ 40,00), muitos exemplos de códigos fonte disponíveis na internet, grande comunidade de desenvolvedores, por fim, o software e o hardware usam o conceito open-source.

O microcontrolador Atmega328 é o mais comumente usado nas placas Arduino. Seu custo está na faixa de R$ 20,00 no Brasil, e possui o benefício da compatibilidade com a plataforma Arduino. Suas principais características são:

- 2 KB de memória RAM;
- 32 KB de memória FLASH (dos quais 30KB podem ser usados pelo programador);
- 32 registradores internos de uso geral;

- 2 interrupções externas;
- 2 Timers de 8 bits;
- 1 Timer de 16 bits;
- 14 Entradas/Saídas digitais, 6 permitem operar com PWM;
- 6 Entradas/Saídas analógicas(que podem ser usadas como digitais)
- Frequência de Clock de 16MHz;

O Atmel AVR será o microcontrolador usado neste projeto. Este é o microcontrolador padrão da plataforma Arduino e pode ser facilmente acoplado ao circuito elétrico desenvolvido, não precisando de processos de solda industriais.

Na figura 37 são mostrados os blocos que compõem o hardware do Arduino.

Figura 37 – Arquitetura do Arduino
Fonte: (www.inf.ufes.br/~erus/arquivos/erus_minicurso arduino.pdf).

A seguir daremos mais detalhes sobre suas entradas e saídas digitais e analógicas.

2.6.3.1 Entrada e saídas

Como foi dito acima, toda a eletrônica ativa está dentro do chip microcontrolador. Para entender melhor essa eletrônica, considere o chip mais simples usado no Arduino: ATMega8 (ver figura 38).

Figura 38 – Microcontrolador ATMega8
Fonte: (www.inf.ufes.br/~erus/arquivos/erus_minicurso arduino.pdf).

Como descrito por Silveira (2011), o chip ATMega8 possui 28 pinos de conexões elétrica, 14 de cada lado. Através desses pinos que podemos acessar as funções do microcontrolador, enviar dados para dentro de sua memória e acionar dispositivos externos.

No Arduino, os 28 pinos deste micro controlador são divididos da seguinte maneira:

- 14 pinos digitais de entrada ou saída (programáveis)
- 6 pinos de entrada analógica ou entrada/saída digital (programáveis)
- 5 pinos de alimentação (gnd, 5V, ref analógica)
- 1 pino de reset
- 2 pinos para conectar o cristal oscilador

Os pinos digitais (14) e os analógicos (6) são utilizados pelo usuário por meio da programação dos mesmos. Estes pinos possuem mais de uma função, podendo ser de entrada ou de saída, alguns podem servem para leitura analógica e também como entrada digital.

Diferente de uma entrada digital, que nos informa apenas se existe ou não uma tensão aplicada em seu pino, a entrada analógica é capaz de medir a tensão aplicada. Através da entrada analógica, conseguimos utilizar sensores que convertem alguma grandeza física em um valor de tensão que depois é lido pela entrada analógica.

Nas saídas digitais podemos fazer com que um pino libere 0 volts ou 5 volts. Com um pino programado como saída digital, podemos acender um led, ligar um relé, acionar um motor, dentre diversas outras coisas. Podemos programar o Arduino para no máximo 20 saídas digitais, porém podemos utilizar um ou mais pinos para

controlar um bloco de pinos.

Existem também pinos que possuem funções especiais que podem ser usado através da programação adequada. A seguir algumas dessa funções e seus respectivos pinos:

- **PWM**: Uma saída digital que gera um sinal alternado (0 e 1) onde o tempo que o pino fica em nível 1 (ligado) é controlado. Pinos 3,5,6,9,10 e 11.
- **Porta Serial USART**: Pode-se usar um pino para transmitir e um pino para receber dados no formato serial assíncrono. Pino 0 (rx recebe dados) e pino 1 (tx envia dados).
- **Comparador analógico**: O uso de dois pinos para comparar duas tensões externas. Pinos 6 e 7.
- **Interrupção externa**: Pode-se programa um pino para avisar o software sobre alguma mudança em seu estado. Pinos 2 e 3.
- **Porta SPI**: Um padrão de comunicação serial síncrono, bem mais rápido que a USART. Pinos 10, 11, 12 e 13.

2.6.4 Programação De Microcontroladores

Um microcontrolador necessita de uma camada de software para fazer a correta tomada de decisões, baseado na programação e em suas entradas. Programar um microcontrolador nada mais é que inserir em seu chip um conjunto de instruções que permita seu controle de acordo com a necessidade do desenvolvedor.

2.6.4.1 Programação em Assembly

Existem níveis de programação para computadores, e Assembly pode ser considerado o nível mais baixo de programação, pois não possui nenhum comando, instrução ou função além daqueles definidos no conjunto de instruções do processador utilizado. Portanto, o programador fica limitado a usar as instruções do processador selecionado para resolver seu problema. “Qualquer operação mais complexa deve ser traduzida em um conjunto de comandos Assembly” (PEREIRA,

2007, p. 16-17)

> Assembly consiste em uma forma alternativa de representação dos códigos de máquina usando mnemônicos, ou seja, abreviações de termos usuais que descrevem a operação efetuada pelo comando em código de máquina. A conversão dos mnemônicos em códigos binários executáveis pela máquina é feita por um tipo de programa chamado Assembler (montador). (SILVEIRA, 2011, p.15)

A Assembly possui um alto grau de complexidade, e exige do programador experiência e um bom conhecimento na área.

2.6.4.2 Programação em C

A linguagem de programação em C é de nível médio, afinal está muito próxima da máquina, porém ao mesmo tempo permite criar aplicações de alto nível.

> C é frequentemente chamada de linguagem de médio nível para computadores. Isso não significa que C é menos poderosa, difícil de usar ou menos desenvolvida que uma linguagem de alto nível como BASIC e Pascal, tampouco implica que C é similar à linguagem assembly e seus problemas correlatos aos usuários. C é tratada como uma linguagem de médio nível porque combina elementos de linguagens de alto nível com a funcionalidade da linguagem assembly. (SCHILDT, 1991, p. 4)

Utilizando a linguagem C, o processo de compilação será muito eficiente. “Eficiência no jargão dos compiladores é a medida do grau de inteligência com que o compilador traduz um programa em C para o código de máquina. Quanto menor e mais rápido o código gerado, maior será a eficiência da linguagem e do compilador”. (PEREIRA, 2007, p. 18). Ela permite que o programador preocupe-se mais com a programação da aplicação em si, já que o compilador assume para si tarefas como o controle e localização das variáveis, operações matemáticas e lógicas, verificação de bancos de memória, etc.

A IDE do Arduíno é totalmente compatível com a linguagem C, e todos os programas exemplos encontrados na comunidade do Arduino são escritos em C.

2.7 SISTEMAS DE CORES

Isaac Newton em seus estudos sobre as propriedades refratoras dos corpos transparentes decompôs a luz branca utilizando um prisma triangular de cristal e depois fez o processo inverso utilizando um segundo prisma invertido. Neste experimento ele constatou que as cores do espectro visível (vermelho, laranja, amarelo, verde, azul, anil e violeta), antes tida como uma qualidade do material refrator que alterava a cor da luz, eram as componentes da luz que as gerava. A luz branca, composta pelas sete cores do arco-íris, conforme demonstrado por Isaac Newton, pode ser sintetizada em suas três componentes básicas ou cores primárias que combinadas duas a duas, formam as secundárias.

As cores podem ser produzidas pela combinação das primárias, ou então, da composição de suas combinações (secundárias e terciárias). Não há um conjunto finito de cores primárias visíveis que produza realmente todas as cores, mas grande parte delas pode ser gerada realmente a partir de três cores primárias escolhidas das extremidades e centro do espectro de luzes visíveis como, por exemplo, as cores vermelha, azul e verde, conhecidas como as cores primárias RBG (figura 37).

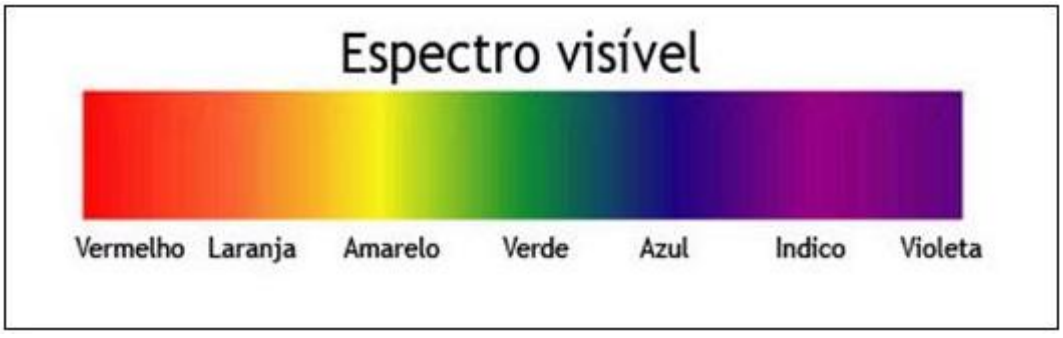


Figura 39 - Espectro visível do homem
Fonte: (http://educar.sc.usp.br/otica/mf4_2.htm).

O motivo de se usar três cores primárias é pelo fato dos olhos humanos possuírem somente três tipos diferentes de sensores coloridos, sensíveis a

diferentes partes do espectro de luz visível. Estes sensores são chamados de fotopigmentos azul, vermelho e verde.

Um sistema de cores nada mais é do que um modelo matemático que descreve de maneira abstrata as cores. Neste, as cores são representas como tuplas de números, geralmente com três ou quatro valores ou componente de cor (cores primárias). Existem vários tipos de sistemas de cores que classificam e analisam os seus efeitos de acordo o ambiente em que as mesmas atuam.

> Um sistema de cores é um modelo que explica as propriedades ou o comportamento das cores num contexto particular. Não há um sistema que consiga explicar todas as características relacionadas às cores, por isso, são usados sistemas diferentes para auxiliar a descrever os diferentes aspectos das cores e sua percepção pelo ser humano. Os sistemas de cores mais conhecidos são XYZ, o RGB, o HSV e o HLS. (AZEVEDO, 2003, p.189)

Todas as cores que podem ser produzidas por um sistema são chamadas de espaço de cores (color space ou color gamut). Um espaço de cor é um método necessário para se quantificar as sensações visuais das cores, que podem assim ser precisamente especificadas. A introdução de uma representação matemática no processo de especificação de cor gera muitos benefícios já que permite a especificação de uma cor através de um sistema de coordenadas geralmente cartesiano.

2.7.1 Sistema De Cores Aditivas e Subtrativas

Os sistemas de cores disponíveis são dependentes de qual ambiente se pretender usar as cores, sendo portando classificados em sistemas de cores subtrativas ou aditivas.

2.7.1.1 Sistema De Cores Subtrativas

Como caracteriza Eduardo Azevedo (2003), os sistemas de cores subtrativas são processos usados nas impressoras e pinturas. Uma pintura é diferente de um monitor que, por ser uma fonte de luz, pode criar cores. As cores subtrativas são as cores usadas em objetos que não possuem fontes de luz própria, e têm como cores primárias as cores: magenta, amarelo e ciano; tidas como as cores primárias subtrativas, pois seu efeito é subtrair, isto é, absorver alguma cor da luz branca.

Quando uma luz branca atinge um objeto, ela é parcialmente absorvida pelo objeto. A parte da luz que não é absorvida é refletida, e ao atingir o olho humano pode-se determinar a cor refletida.

Na figura 38 são apresentadas as cores primárias subtrativas e suas respectivas combinações para produção das cores secundárias (verde, azul e vermelho).

Figura 40 - Processo subtrativo das cores
Fonte: (http://luzeled.com.br/luz).

Se um objeto absorve a componente vermelha da luz branca refletida, a cor é vista como ciano. Já se a componente retirada da luz branca for a verde, a cor vai ser vista como magenta. Por fim, se subtrai a componente azul da luz branca refletida, a cor gerada vai ser amarela.

As cores terciárias podem ser obtidas através da combinação de duas primárias em proporções diferentes.

2.7.1.2 Sistema Cores Aditivas

Este é o tipo de sistema usado nos monitores de vídeos e televisores, onde a cor é gerada pela mistura de vários comprimentos de onda luminosa, provocando uma sensação de cor quando atinge e sensibiliza o olho.

As cores primárias aditivas são: vermelho, verde e azul. Como é descrito por Eduardo Azevedo (2003), no processo aditivo, o preto é gerado pela ausência de qualquer cor, indicando que nenhuma luz está sendo transmitida, e a cor branca é a mistura de todas elas, indica que as suas componentes estão no valor máximo.

Na figura 39 temos as cores primárias aditivas e suas respectivas combinações para produção das cores secundárias (amarela, ciano, magenta).

Figura 41 - Processo aditivo das cores
Fonte: (Fonte: http://luzeled.com.br/luz).

Ao contrário das cores subtrativas, as cores deste sistema são formadas a partir da soma das cores primárias deste. Em uma imagem coloria, a representação da cor C de cada pixel da imagem pode ser obtida matematicamente por:

$$C = r.R + g.G + b.B$$

Onde R, G e B são as três cores primárias e r, g, b são os coeficientes de mistura correspondentes a cada uma das intensidades associadas a cada um dos canais

RGB. Os coeficientes de mistura podem ser números reais ou inteiros. Os valores inteiros são usados no processo de aquisição de imagens raster e em sua armazenagem na forma de arquivos de imagem. Assim a cor C de cada pixel da imagem pode ser plotada no espaço de cores RGB usando-se os coeficientes de mistura (r, g, b) como coordenadas. Portando, uma cor, pode ser considera como um ponto em um espaço tridimensional (modelada como um subconjunto do espaço R^3 ou N^3), onde cada cor possui uma coordenada (r, g, b).

2.7.2 O Modelo RBG

O modelo RBG é um modelo de cores aditivas no qual a cores vermelha, azul e verde são somadas de várias maneiras para reproduzir uma variedade de cores. O nome do modelo vem das iniciais dos nomes das cores primárias em inglês (Red, Blue e Green).

RGB é um modelo que dependente do dispositivo em uso. Dispositivos diferentes podem detectar ou reproduzir um determinado valor de RGB de forma diferente, uma vez que os elementos usados na fabricação dos mesmos podem ter respostas diferentes ao mesmo nível de RBG. Tal variação pode ser de um fabricante para outro, ou até mesmo de um dispositivo para outro ao longo do tempo. Assim, um valor RGB não define a mesma cor em todos os dispositivos sem algum tipo de gerenciamento de cores.

Os dispositivos que usam o modelo RGB são dispositivos de entrada como: TVs, câmeras de vídeos, scanners de imagens e câmeras digitais; ou os dispositivos de saída como: televisores de várias tecnologias (CRT, LCD, plasma, etc.), monitores de computador e telefones celulares, displays de LED multicoloridos e projetores de vídeos.

O sistema RBG, usa um sistema de coordenadas cartesianas R, G, B, cujo subespaço de interesses é o cubo unitário é mostrado na figura 40.

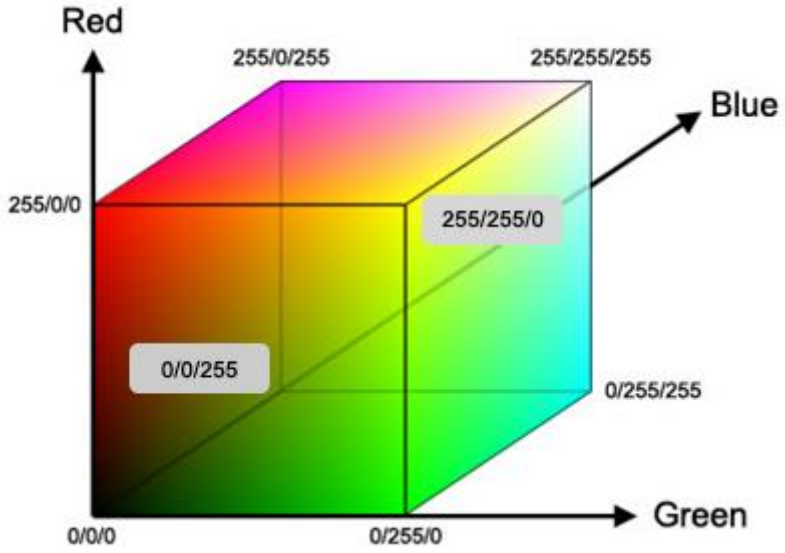


Figura 42 - Subespaço do modelo RGB a partir dos eixos XYZ
Fonte: (Fonte: http://www.laguerradelosmundos.net/sistema-de-color-rgb).

A diagonal principal do cubo, que vai do preto ao branco, possui quantidade iguais de cores primárias e representa a escala de cinza. Cada ponto colorido, dentro dos limites do cubo, pode ser representado por (R,G,B), onde os valores de R,G,B variam entre zero a um valor máximo. Por conveniência, a maioria dos arquivos digitais atuais usam números inteiros entre 0 e 255 para especificar estas quantidades. O número 0 indica ausência de intensidade e o número 255 indica intensidade máxima.

Assim, cada cor no sistema RGB é identificado por uma tripla ordenada (R, G, B) de números inteiros, com $0 \leq R \leq 255$, $0 \leq G \leq 255$ e $0 \leq B \leq 255$. Sendo assim, podemos associar cada cor do sistema RGB com pontos com coordenadas inteiras de um cubo com aresta de tamanho 255. Podemos observar que nos vértices do cubo temos as cores primárias: vermelho (255,0,0), verde (0,255,0), azul (0,0,255); as cores secundárias: amarela (255,255,0), ciano (0,255,255) e magenta (255,0,0); além do branco (0,0,0) e o preto (255,255,255).

2.8 MULTIPLEXAÇÃO

A multiplexação consiste na operação de codificar informações de duas ou mais fontes de dados num único canal. São utilizados em situações onde o custo de implementação de canais separados para cada fonte de dados é maior que o custo e a inconveniência de utilizar as funções de multiplexação/demultiplexação. O dispositivo usado para este tipo de operação chama-se multiplexador (multiplexer ou apenas mux).

Alguns diferentes modos de realizar a multiplexação:

- Multiplexação por Divisão de Frequência (FDM - Frequency Division Multiplexing): o espectro de frequências é dividido em diversas faixas, uma para cada transmissão ou comunicação distinta.
- Multiplexação por Divisão de Tempo (TDM - Time Division Multiplexing): o tempo de transmissão de um canal é dividido em pequenos slots de tempo (iguais ou de acordo com uma proporção de estatística), atribuindo-se um slot a cada uma das várias transmissões que estão acontecendo ao mesmo tempo.
- Multiplexação por Largura de Pulso (PWM - Pulse Width Modulation): técnica para a obtenção de resultados análogos com meios digitais. Modula-se largura do pulso, formalmente, a duração do pulso, com base na informação do sinal modulador.

2.8.1 Multiplexação por Largura de Pulso (PWM - Pulse Width Modulation)

PWM é uma técnica de multiplexação que permite simular uma saída analógica em uma digital, através da modulação da largura do sinal digital. Embora seja uma técnica de modulação utilizada para codificar a informação para transmissão, a sua utilização principal é permitir o controle da energia fornecida aos dispositivos elétricos, especialmente para as cargas de inércia, tais como motores e no controle de luminosidade de componentes luminosos, por exemplo, o LED.

Os sistemas digitais baseiam o seu funcionamento na lógica binária, em que toda a informação é guardada e processada sob a forma de zero (0) e um (1). Esta representação é conseguida usando dois níveis discretos de tensão elétrica, representados em uma forma básica de onda, chamada de Onda Quadrada.

Na figura 41 temos o gráfico representativo de uma onda quadrada, com o nível baixo no valor -1 e o nível alto no valor +1.

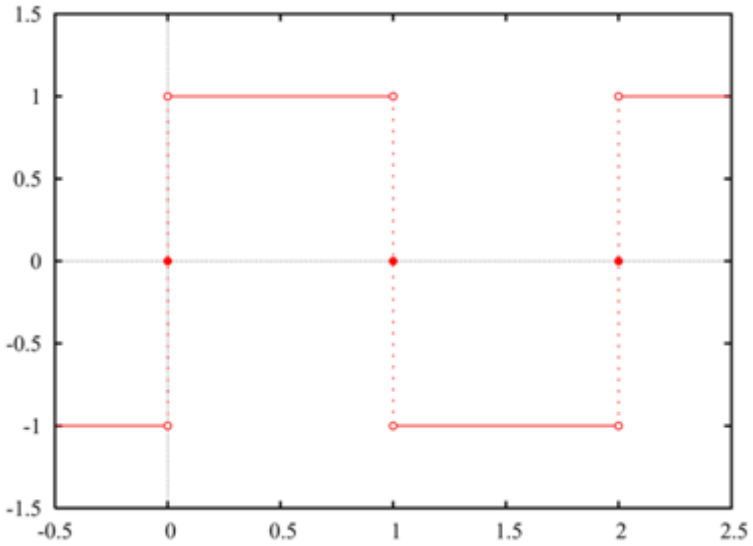


Figura 43 - Gráfico de uma onda quadrada
Fonte: (http://commons.wikimedia.org/wiki/File:Square_wave_closeup.png).

Uma onda quadrada ideal alterna regularmente e instantaneamente entre os dois níveis, que podem ou não incluir o zero. Ou seja, variar de um valor mínimo diretamente para o valor máximo, sem passar por valores intermediários. Podemos simular uma porta analógica em uma digital alterando a porção do tempo que o sinal alto possui em relação à porção de tempo do sinal baixo. A porcentagem do tempo que um pulso permanece em nível alto em comparação ao tempo em que permanece em nível baixo é chamada de ciclo de trabalho.

A figura 42 mostra quatro sinais PWM, através de gráficos de tensão ao longo do tempo. Os sinais A e B mostram ciclos de trabalhos de 50%. A linha C mostra cerca de 10%, e a D mostra cerca de 80% no ciclo de trabalho. Note que conforme o ciclo de trabalho aumenta, a quantidade de tempo em que o sinal é alto é mais extensa. Em outras palavras isso significa que a corrente fluirá na célula 80% do tempo na linha D, mas apenas 10% do tempo na linha C.

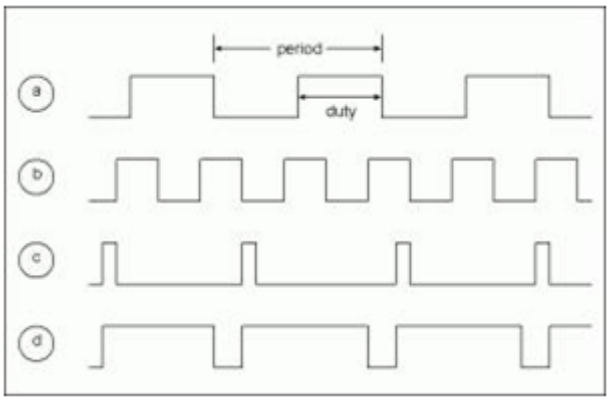


Figura 44 - Diferentes sinais digitais
Fonte: (http://www.hho-eccosaver.com/blog/o-que-e-o-pwm-como-funciona-e-para-que-serve-2).

Através do PWM, podemos, por exemplo, controlar a luminosidade de LED, modulando o seu pulso digital, que vai de 0 volts (apagado) a 3 volts (luminosidade máxima). Combinando LEDs de cor vermelha, azul e verde, podemos simular as cores do sistema do sistema RBG.

2.9 TRABALHOS RELACIONADOS

Nesta seção será apresentando um trabalho similar que se mostra relevante ao projeto, visando facilitar o entendimento analisando a suas dificuldades e extrair possíveis soluções para os problemas encontrados ao longo do desenvolvimento deste projeto.

2.9.1 Matriz de LEDs sensores

Os autores (VIOLADA, DA SILVA E MEURER, 2011) deste projeto têm como objetivo a manipulação e a verificação de propriedades pouco exploradas dos diodos emissores de luz (LED). Faz-se uso da característica capacitiva que um LED adquire quando polarizando reversamente e a geração de foto-corrente para diminuição do tempo de descarga de tal capacitor. Com esses dois fatores pode-se acionar um

LED, por exemplo, apenas apontando uma lanterna em sua direção.

Após atingir este objetivo, implementou-se uma pequena matriz de LED onde foi possível “desenhar” diferentes formar sobre a superfície à medida que o usuário desloque a lanterna sobre os LEDS.

Neste projeto foi utilizada uma matriz controlada por uma microcontrolador Arduino. Inicialmente a matriz de LED deste projeto estava organizada como mostrada na figura 10b da seção 2.4.2, ou seja, uma matriz de cátodo comum.

O principal problema encontrado neste projeto foi devido ao grande número de portas sendo alteradas. Com matrizes de tamanhos superiores a 4x4, o microcontrolador não possuía velocidade suficiente para que fosse feita a varredura da matriz sem que o olho humano percebesse que, na verdade, os LEDs estavam piscando.

Este problema foi resolvido implementando outro tipo de matriz (ver figura 45), onde os LEDs são ligados independentes um do outro, cada um deste exigindo uma porta digital do microcontrolador. Assim, seria necessário somente verificar se há luz incidida em cada LED individualmente. Quando há luz incidida o LED acende e não torna a apagar até ser inicializada novamente a matriz.

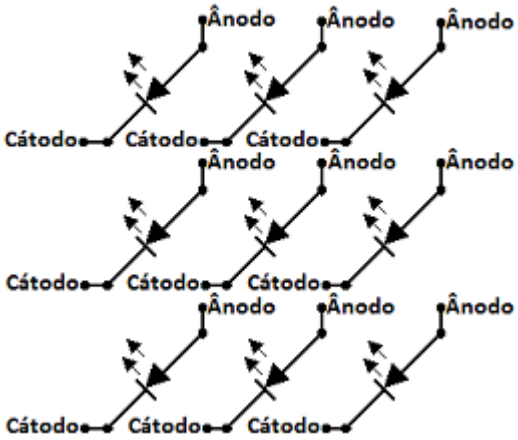


Figura 45 – Matriz com LEDs ligados maneira independentes
Fonte: (VIOLADA, DA SILVA e MEURER, 2011, p. 19).

Os autores concluem que para uma continuação do projeto seria necessário à utilização de um microcontrolador com maior velocidade de processamento e mais portas, para que mais LEDs pudessem ser incluídos no projeto.

3 METODOLOGIA

Neste capítulo será apresentando a parte prática deste trabalho, detalhando a implementação do circuito usado para o controle da matriz de LED associado ao Arduino. Serão expostas também algumas funcionalidades da camada de software do Arduino, juntamente com o algoritmo responsável em ler um mapa de bits e multiplexar a informação para os registradores de deslocamento, por meio das portas digitais necessárias do microcontrolador.

3.1 PROCEDIMENTOS E TECNOLOGIA

Para realizar este trabalho, foi feito um estudo bibliográfico a respeito do assunto a ser desenvolvido, neste caso, os painéis de LED, a fim de obter o embasamento teórico antes de partir para a modelagem em si.

Primeiramente analisamos os tipos de painéis de LED, principais aplicações, estrutura básica e funcionamento do mesmo. Após este estágio conclui-se que um projeto de um painel de LED poderia ser divido em três etapas básicas:

- Etapa 1: responsável por pegar a informação a ser exibida no painel e gerar o sinal elétricos correspondentes à informação;
- Etapa 2: estrutura que transfere os sinais elétricos gerados na etapa 1 para a estrutura responsável em exibir a informação propriamente dita;
- Etapa 3: parte do painel de LED responsável por exibir a informação proveniente dos pulsos elétricos.

Na etapa 1, foi feito um estudo sobre as matrizes e suas importância na computação gráfica, no processo de manipular da informação gráfica. Assim como, foi realizado um estudo no processo de Rasterização, responsável por transformar

de imagens gráficas codificadas vetorialmente para um padrão matricial conhecido como bitmap (mapa de bit).

Nesta etapa também, vimos à necessidade de microcontrolador, no caso deste projeto, o Arduino. Este seria composto por uma camada de software, responsável pelo tratamento da informação, a ser exibida pelo LED. Basicamente, o microcontrolador, recebe um mapa de bits, e partir desse gera os sinais digitais, responsáveis em exibir a informação no painel de LED.

Na etapa 2, estudamos uma maneira de transferir a informação digital gerada pelo microcontrolador para a matriz de LED, responsável por exibir a informação ao usuário do mesmo. Surgiu então a seguinte questão: como pegar a informação digital, proveniente do microcontrolador Arduino e distribuir para os LEDs individualmente.

O Arduino possui 19 saídas que podem ser usadas como portas digitais, quantidade insuficiente para controlar uma matriz de LED que exige muitas conexões. Tomando como exemplo uma um painel de LED, com uma matriz de LED de resolução 800x600, seriam exigidas 1400 conexões do microcontrolador.

Para resolver este problema, fez-se um estudo sobre os registradores de deslocamento, os seus tipos, e suas aplicações. O uso de registradores de deslocamento permite que os sinais digitais sejam multiplexados em uma única porta do Arduino, ou seja, de maneira serial. A informação binária entra e percorre os registradores de deslocamento de forma serial, e saem na forma paralela. Dessa forma podemos expandir as portas digitais do Arduino com um baixo custo, pois portas lógicas de microcontrolador são mais caras que os registradores de deslocamento integrado.

A etapa 3 é a responsável por exibir a informação ao usuário. Fez-se um estudo sobre as matrizes de LED, estrutura fundamental do painel de LED. Estudamos o LED e suas propriedades, e como estes podem ser ligados numa estrutura matricial de maneira a reduzir o número de conexões. Através de um processo de multiplexação que ocorre na própria matriz, acendendo uma linha por vez, é possível diminuir substancialmente a quantidade de conexões necessárias da matriz de LED.

Por exemplo, em uma matriz de LED de 800x600 possui 481 mil LEDs, ter uma conexão individual para cada LED tornaria não só construção de um painel de caro, mas também muito trabalhoso. Pelo processo de multiplexação que ocorre na própria matriz de LED, é possível diminuir essa quantidade para 1400 conexões

provenientes dos registradores de deslocamento.

A figura 43 apresenta a arquitetura básica do painel de LED deste projeto.

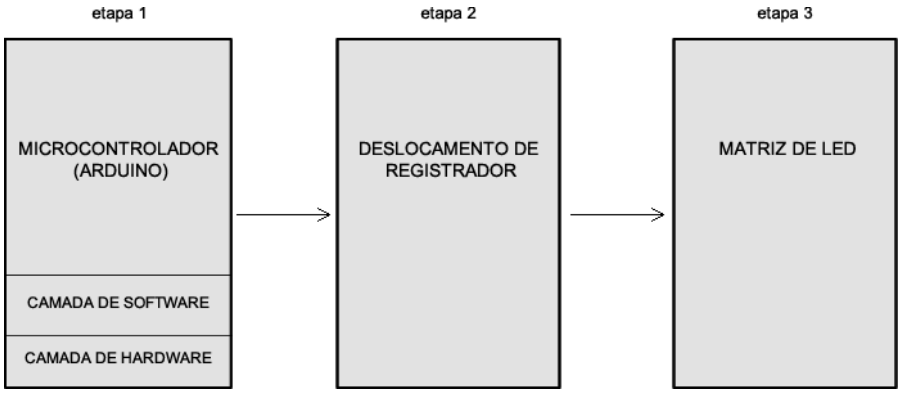


Figura 46 - Arquitetura do painel de LED
Fonte: Autoria própria.

Foram feitas também pesquisas bibliográficas, que possibilitem em futuras modificações trabalhar com painéis de LED RBG. Para isso, foi realizado um estudo sobre os sistemas matemático de cores RBG e na multiplexação do pulso digital por PWM.

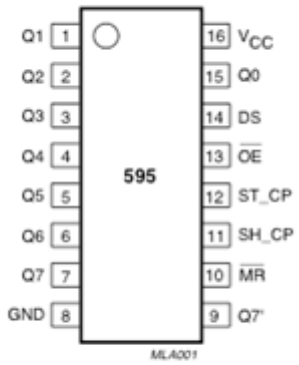
3.2 O HARDWARE

Este projeto foi implementado com seguintes componentes físicos: Microcontrolador Arduino, dois CIs de registrador de deslocamento do tipo 595 (74HC595) de 8 bits, oito resistores de 220 Ohm para controlar a voltagem que vai para os LEDs da matriz e uma matriz ânodo comum monocromática com 64 LEDs (8x8).

No apêndice A pode ser visto simulação da interligação dos componentes. E no apêndice B o diagrama esquemático do hardware.

Um ponto chave neste projeto é o uso dos registradores de deslocamento para realizar a expansão das portas do Arduino. Na figura 44, temos o datasheet do registrador de deslocamento usado no projeto, o 74HC595, juntamente com a

descrição de cada pino.



PINS 1-7, 15	Qo " Q7	Output Pins
PIN 8	GND	Ground, Vss
PIN 9	Q7"	Serial Out
PIN 10	MR	Master Reclear, active low
PIN 11	SH_CP	Shift register clock pin
PIN 12	ST_CP	Storage register clock pin (latch pin)
PIN 13	OE	Output enable, active low
PIN 14	DS	Serial data input
PIN 16	Vcc	Positive supply voltage

Figura 47 - Pinagem do 74HC595
Fonte: (http://www.arduino.cc/en/Tutorial/ShiftOut).

Como pode ser visto nos apêndices A e B, usamos somente três saídas digitais do Arduino para controlar a matriz de LED: Os pinos 8, 11 e 12. Estas saídas estão ligadas aos registradores de deslocamento nos pinos ST_CP (pino data trava), DS (pino de entrada de dados) e SH_CP (pino do clock) consequentemente. A alimentação elétrica também é proveniente do Arduino através de duas saídas: 5V (polo positivo de 5 volts) e Gnd (polo negativo), com uma amperagem de 20 mA.

As saídas digitais das portas 8 e 12 do Arduino são ligadas a todos os registradores de deslocamento, pois todos possuem o mesmo clock e controle de travas. Já a saída digital 11 é somente ligada ao primeiro registrador, que é responsável por receber os dados provenientes do Arduino inicialmente.

A saída serial (pino Q7'') do registrador U1 é ligada à entrada de dados (pino DS) do registrador de deslocamento U2. Todos os registradores de deslocamento do tipo 595 possuem uma saída serial de dados, que ligada à entrada de dados de outro registrador de deslocamento (exceto no primeiro), possibilita o deslocamento da informação em um efeito cascata.

Cada CI 74HC595 possui oito saídas paralelas (Q0 a Q7), que estão que conectadas às colunas e linhas da matriz de LED para controlar o fluxo elétrico

sobre a mesma. O registrador U1 é responsável por enviar o sinal digital às colunas, enquanto U2 envia sinais digitais às linhas.

Podemos observar também que as saídas paralelas do registrador de deslocamento U1 estão ligadas as colunas por um resistor de 220 Ohms. O uso dos resistores se deve ao fato do CIs precisarem de 5 volts para funcionar corretamente, enquanto a matriz de LED pode ser alimentada no máximo por 3,3 volts. Os resistores fazem essa redução de voltagem, evitando que os LEDs das matrizes se danifiquem.

Através de cálculo simples, podemos encontrar o valor da resistência necessária para reduzir à tensão necessária. Veja a fórmula abaixo:

$$R = (Vs - V_l) / I$$

Onde Vs é voltagem fornecida, V_l é a voltagem da matriz e I a corrente da matriz de led.

Em nossa matriz de LED de ânodo comum, os LEDs são acionados aplicando uma tensão positiva em suas linhas e negativa em suas colunas. Se desejarmos acender os LEDs das colunas 3 e 6 da linha 3. Para o registrador U2, responsável por manda sinais para as linhas, mandaríamos a sequência binária 00100000. Desse modo, todos os LEDs da linha 3 estariam com uma corrente positiva em seus ânodos e prontos para receber a corrente negativa em seu cátodo.

No registrador de deslocamento U1, que é responsável por mandar sinais para as colunas, mandaríamos a sequência binária 11011011. Assim, os LEDs das colunas 3 e 6 receberiam uma corrente elétrica negativa em seus cátodos e consequentemente iriam acender.

Note que na sequência binária 11011011, os LEDs a serem acesos nas colunas são representados na sequência binária pelo valor 0, correspondente ao nível baixo. Na prática, a informação que vai ser exibida numa matriz de LED, é proveniente de mapa de bits. Logo essa sequência estaria representada como 00100100. Se essa sequência de bits fosse enviada para o registrador de deslocamento, não teria efeito, pois em nossa matriz precisamos de um sinal negativo para as colunas. Podemos resolver isso aplicando a operação lógica de negação na sequência de bits, desse modo not(00100100) = 11011011.

Apesar deste projeto usar uma matriz de 8x8, os conceitos usados nesta servem

para matriz de qualquer dimensão. Matrizes maiores exigiriam mais registradores de deslocamento, mas o controle da matriz seria semelhante. Podemos encontrar no mercado registradores de deslocamento de 16 bits, o que facilitaria trabalhar com matrizes de maiores resoluções.

A escolha de uma matriz 8x8 se justifica por esta simplificar o entendimento da teoria que envolver os painéis de LED e pelo seu custo reduzido.

3.3 O SOFTWARE

O software que controla o painel de LED foi implementado na linguagem própria do Arduino. O Arduino utiliza uma IDE escrita em Java derivado do que foi feito para Processing e Wiring. Seu principal objetivo é facilitar a programação de controle do hardware. Após escrever o programa, com apenas um clique do mouse é possível fazer um upload do programa para o Arduino. Um dos fatos que evidencia a simplicidade de seu uso.

Existem duas funções básicas para criar um programa:

- setup() função chamada apenas uma vez no início do programa, geralmente utilizada para a inicialização de variáveis e/ou parâmetros.
- loop() função que fica em um laço infinito de repetição enquanto o dispositivo permanece ligado. Permite que o Arduino receba dados, mude de estado e as responda conforme programado.

A seguir será explicado brevemente o funcionamento do programa, cujo código pode ser consultado no apêndice C desta monografia.

O código deste projeto utiliza um recurso do chip ATmega do Arduino, conhecido como Hardware Timer: basicamente, um timer que pode ser utilizado para disparar um evento. No caso deste projeto, estamos definindo um ISR (Interrupt Service Routine, ou rotina de serviço de interrupção) para que dispare a cada 10 mil microssegundos, o equivalente a um centésimo de segundo.

Para realizar o ISR, fazemos uso de uma biblioteca que facilita o uso de Interrupções, a TimerOne. Simplesmente informamos qual o intervalo da interrupção (neste caso 10 mil microssegundos) e a função que desejamos ativar cada que vez a interrupção for disparada.

Não função setup(), definimos as portas digitais que vão ser usados como saída por meio da função pinMode(pino,tipo), “tipo” pode ser INPUT ou OUTPUT. Com o uso da biblioteca TimerOne, definimos na função setup, o tempo da interrupção pela a instrução Timer1.initialize(10000).

Determinamos também em no escopo de setup(), qual função vai ser executada a cada interrupção pelo comando Timer1.attachInterrupt(screenUpdate), neste casado a função screenUpdate() que será descrita adiante.

Como já foi citado, nosso objetivo é implementar um painel de LED que seja capaz de exibir mensagens com rolagem horizontal. Para isso declaramos uma variável com o nome “font” do tipo array que contem todos os mapas de bits dos caracteres imprimíveis da tabela ASCII (32 a 126). Assim o caractere 65, correspondente à letra ‘A’, terá o seguinte mapa de bits:

B00001110
B00010001
B00010001
B00010001
B00011111
B00010001
B00010001
B00010001

A partir de um string de entrada, os mapas de bits são montados pela função scroll(char myString[], int speed) no buffer de saída, sendo esta a variável byte buffer[8]. Além de atualizar o buffer, a função scroll é responsável por realizar o processo de rolagem na mensagem pela matriz de LED.

Para utilizar o registrador de deslocamento, o pino latch e o pino clock devem estar definidos como LOW. O pino do latch permanecerá LOW até que todos os oitos bits tenham sido definidos. Isso permite que os dados entrem no registrador de armazenamento (um local no CI para armazenamento de 1 ou 0).

Em seguida, definimos um sinal HIGH ou LOW no pino de dados, de acordo com o bit da sequência binária, e define o pino do clock como HIGH. Esse processo faz com que o sinal apresentado no pino de dados seja armazenado no registrador. Feito isso, definimos novamente o clock como LOW, e apresentamos no pino de dados o próximo bit. Fazendo isso 8 vezes, teremos enviados o número completo de 8 bit ao 595. O pino latch será definido com HIGH, o que transferirá os dados do

registrador de armazenamento para o registrador de deslocamento e suas saídas paralelas apresentaram os sinais correspondentes. Na tabela abaixo temos essa sequência de eventos:

Tabela 4 - Enviando dados para o 595

Sequência	Pino	Estado	Descrição
1	Latch	LOW	Permite que dados entrem
2	Dados	HIGH	Primeiro bit de dados (1)
3	Clock	HIGH	Clock vai para HIGH. Dados são armazenados
4	Clock	LOW	Pronto para receber o próximo bit
5	Dados	HIGH	2º bit de dados (1)
6	Clock	HIGH	2º bit de dados é armazenando
...	...	...	
n-3	Dados	HIGH	8º bit de dados (0)
n-2	Clock	HIGH	8º bit de dados é armazenando
n-1	Clock	LOW	Impede que novo dados sejam armazenados
n	Latch	HIGH	Envia os 8 bits em paralelo

Fonte: Autoria própria.

Esse processo de controle dos registradores de deslocamento é feito por duas funções: screenUpdate() e shiftIt(). A função screenUpdate é executada a cada 10000 microssegundos pelo processo de interrupção definido na função setup(). Ela é responsável por consultar o buffer de saída, enviando a sequência binária de cada linha para a função shiftIt(), e esta envia os sinais digitais aos registradores de deslocamento de acordo com sequência de bits processada.

Na função screenUpdate() um laço percorre todas as linhas do buffer. No inicio do bloco do laço o comando digitalWrite(latchPin, LOW), abre o latch, deixando os registradores de deslocamento prontos para receberem os dados. Após isso temos duas chamada para a função shiftIt: shiftIt(~buffer[k]) e shiftIt(row). A primeira envia para o registrador U1 a sequência binária que está na linha k do buffer, a segunda chamada da função shiftIt envia para o registrador U2 a sequência binária corresponde ao número da linha do buffer que está sendo processada. Como já foi explicado na seção 3.2 a negação dos bits em shiftIt(~buffer[k]) se justifica porque a matriz de LED do projeto é tipo ânodo comum e precisa de uma tensão negativa nas colunas para funcionar. Após isso a instrução digitalWrite(latchPin, HIGH), fecha o latch, enviando os dados no registrador para a matriz de LED.

A função shiftIt é responsável por realizar os eventos da sequência 2 até n-1 vistos na tabela 4. Um laço percorre o conteúdo binário da linha do buffer armazenada na variável dataOut. O bloco do laço inicia-se com o comando digitalWrite(clockPin, LOW), após isso uma máscara de bits (8 bits) verifica se o bit analisado no laço é 1 ou 0, e armazena o valor HIGH ou LOW na variável pinState. O comando digitalWrite(dataPin, pinState) apresenta no pino dados o valor do bit presente na variável pinState. Na linha seguinte o clock é defino como HIGH pela instrução digitalWrite(clockPin, HIGH), e o valor presente no pino de dados é armazenado no CI.

Em resumo, o software de controle do painel de LED pode ser divido em duas partes principais: Uma parte que atualiza o buffer dada uma mensagem e consultando os mapas de bits correspondente dos caracteres. E uma segunda parte que consulta a informação contida no buffer para gerar os sinais digitais para os registradores de deslocamento. A figura 45 simplifica o funcionamento do software responsável por controlar o painel de LED.

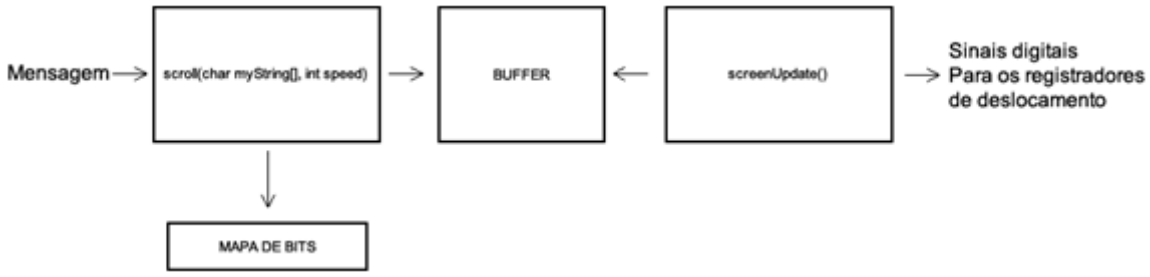


Figura 48 - Funcionamento do software
Fonte: Autoria própria.

3.4 TESTES

O teste realizado no painel consistiu em verificar que dado um texto de entrada, os LEDs corretos sejam acionados. Além disso, verificou-se também se o fenômeno de persistência da visão ocorreu da maneira esperada, através do processo de multiplexação que ocorre na matriz de LED do painel.

Como foi dito na seção 3.3, os textos tem sua entrada a partir da função *scroll.* Na figura 49 temos uma chamada para a função scroll tendo como argumento um texto (“A”) e um inteiro referente à velocidade de rolagem do texto.

```
void loop() {
  clearDisplay();
  scroll(" A ", 70);
}
```

Figura 49 – Texto aplicado na função scroll
Fonte: Autoria própria.

Na figura 50 temos a letra “A” sendo exibida no painel de LED.

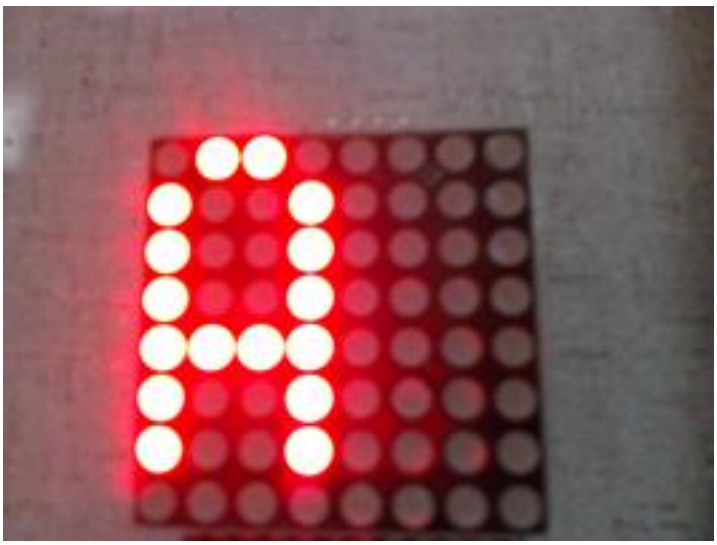


Figura 50 – Exibindo “A” no painel
Fonte: Autoria própria.

Na figura 51 temos a exibição do texto “ABC” no painel. Neste caso, como o tamanho do texto é maior do que a resolução da matriz, o processo de rolagem é acionado.

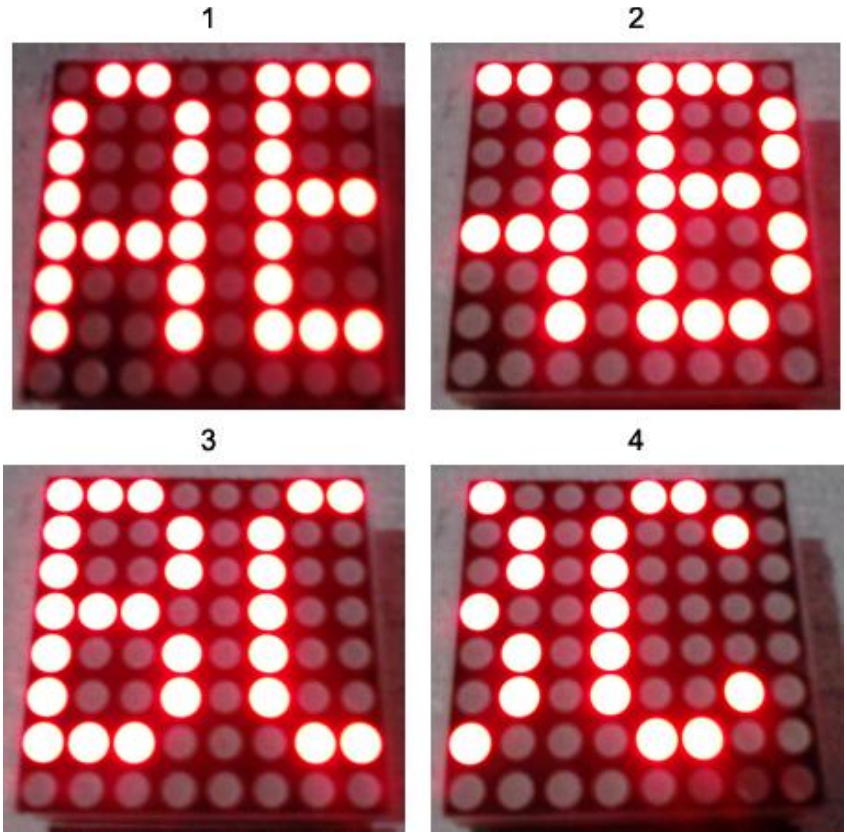


Figura 51 – Exibindo “ABC” no painel
Fonte: Autoria própria.

4 CONCLUSÃO

Este trabalho apresentou um projeto de um painel de LED, baseado na plataforma Arduíno. Esta plataforma permitiu a implementação deste projeto de hardware com nível de complexidade substancialmente menor.

Através da camada de hardware e software do Arduíno tinha-se tudo que seria necessário para processar as informações de entrada e gerar os sinais lógicos necessários a serem usado no projeto. A partir disto, coube-se criar uma interface entre o Arduíno e a estrutura que continha os LEDs responsáveis por exibir a informação. Tal interface foi construída através dos registradores de deslocamento, um importante conceito em eletrônica digital, essencial neste projeto.

Através destes tipos de registradores de deslocamento, é possível expandir as portas digitais do Arduíno. Portanto, a aplicação deste conceito não fica restrita somente neste projeto, mas podem ser aplicados em projetos maiores que necessitem de vários controles digitais.

Por meio de uma metodologia, embasada em conceitos teóricos necessários, foi

possível alcançar o objetivo do projeto de construir um painel de LED capaz de exibir mensagens em textos.

4.1 TRABALHOS FUTUROS

Embora tenhamos ficado satisfeitos com os resultados obtidos, afirmamos também que melhorias podem ser feitas neste projeto deixando-o mais completo. Em futuras modificações, pode-se aumentar a resolução do painel e/ou transforma-lo em painel colorido.

A questão da resolução da tela do painel é relativamente mais simples do que do a das cores. Como foi mencionado, os conceitos aplicados aqui servem para um projeto de maior escala, alterando poucos fatores. Talvez a maior alteração e a mais importante seriam na quantidade de registradores para controlar os terminais de uma matriz de LED maior, neste caso, ligaríamos mais registradores de deslocamento no arranjo dos circuitos.

Transformar o painel de uma cor para um painel colorido envolve não só o uso de mais registradores, mas também a aplicação de um conceito importante, a Multiplexação por Largura de Pulso.

No projeto de painel de LED, cada pixel é formado por apenas um LED, o que significa que em uma matriz de LED de 8x8, é preciso apenas dois registradores pra controlar a matriz de LED: um para as linhas e outro para as colunas. Em uma matriz colorida, cada pixel seria formando por três LEDs, no intuito de formar um LED RBG. Neste caso uma matriz de LED de 8x8, precisaria de quatro registradores para ser controlada: Um para as linhas e três para as colunas (cada um controlando uma cor). No apêndice D, pode ser visto o diagrama esquemático dos registradores de deslocamento para controlar uma matriz de LED RBG.

Outra questão importante é como gerar as cores em cada pixel. Em nosso projeto, os LEDs da matriz só recebem dois valores elétricos discretos, que por consequência deixam um LED especifico no estado aceso ou apagado. Somente com esses dois valores elétricos, iríamos conseguir somente oito cores em cada pixel RBG (2^3), uma quantidade muito pequena pra se trabalhar com elementos gráficos coloridos. Para conseguimos mais cores, tínhamos que variar o valor do

pulso elétrico do LED a valores intermediários indo do nível baixo ao nível alto, que por consequência iria variar o brilho do LED. Esses valores poderiam ser baseados no sistema de cores RBG, onde poderíamos mandar tuplas com valores correspondentes a cada LED em um pixel.

Essa variação do pulso do LED é feita através do PWM. A partir de um mapa de bits RBG, poderíamos gerar pulsos digitais modulados para os LEDs, e obter uma combinação que permitirá produzir até 16 milhões de cores.

REFERÊNCIAS

ANDREY, J. M. **Eletrônica básica: teoria e prática**. São Paulo: Rideel, 1999. 416p.

AZEVEDO, E. **Computação Gráfica - Teoria e Pratica**. 1ª. ed. São Paulo: CAMPUS, 2003. 368p.

SOUZA, A. R. de; et al. **A placa Arduino: uma opção de baixo custo para experiências de física assistidas pelo PC**. Revista Brasileira de Ensino de Física, São Paulo, v. 33, n. 1, 1702, 2011.

VIOLADA, G. C.; DA SILVA, L. M.; MEURER R. F. **Matriz de LEDs sensores.** 2011. Monografia (Engenharia de Computação). Disponível em: < http://www.pessoal.utfpr.edu.br/msergio/Monog-11-1-Matriz-LEDS.pdf> Acesso em: 12 set. 2013

BRAGA, N. C.**Curso de Eletrônica - Volume 3 - Eletrônica Digital**. São Paulo: Instituto Newton C. Braga, 2013. 179p.

BRAGA, N. C. **Curso de Eletrônica - Volume 1 - Eletrônica Básica**. São Paulo: Instituto Newton C. Braga, 2013. 219p.

KÜHNEL, C. **BASCOM Programming of Microcontrollers with Ease: An Introduction by Program Examples**. Estado Unidos: Universal Publisher, 2001. 236p.

LUPPI, A.; SCHUNK, L. M. **Microcontroladores AVR: Teorias e aplicações práticas**. São Paulo: Érica, 2001. 180p.

MALVINO, A. P. **Eletrônica**. 4ª. ed. São Paulo: Pearson Makron Books, 1997. 746p.

MCROBERTS, M. **Arduino Básico**. São Paulo: Novatec, 201. 453p.

SCHILDT, H. **C Completo e Total**. São Paulo: McGraw-Hill Ltda, 191. 827p.

SILVEIRA, J. A. **Experimentos com o Arduino**. São Paulo: Ensino Profissional, 2011. 39p.

SOUZA, D. J. **Desbravando o PIC - Ampliado e Atualizado para PIC 16F628A**.

São Paulo: Érica, 2009. 272p.

Minicurso Arduino JACEE 2012. Disponível em: <www.inf.ufes.br/~erus/arquivos/erus_minicurso arduino.pdf> Acesso em: 12 set. 2013

Aliexpress. Disponível em: <http://pt.aliexpress.com/item/5pcs-8x8-Dot-Matrix-1-9mm-Diameter-Red-LED-Display-free-shipping/594892927.html> Acesso em: 7 abr. 2013

Arduino Learning ShiftOut. Disponível em: <http://playground.arduino.cc/Portugues/LearningShiftOut> Acesso em: 12 jul. 2013

Arduino PWM. Disponível em: <http://arduino.cc/en/Tutorial/PWM> Acesso em: 20 jun. 2013

Arduino ShiftOut. Disponível em: <http://www.arduino.cc/en/Tutorial/ShiftOut> Acesso em: 12 jul. 2013

Arduino ShiftOut Reference. Disponível em: <http://arduino.cc/en/Reference/ShiftOut> Acesso em: 12 jul. 2013

Educar. Disponível em: <http://educar.sc.usp.br/otica/mf4_2.htm> Acesso em: 10 mai. 2013

Hho-Eccosaver. Disponível em: <http://www.hho-eccosaver.com/blog/o-que-e-o-pwm-como-funciona-e-para-que-serve-2> Acesso em: 4 ago. 2013

Laguerra de los mundos. Disponível em: <Fonte: http://www.laguerradelosmundos.net/sistema-de-color-rgb> Acesso em: 4 ago. 2013

Led matrix with pwm using na Arduino. Disponível em: <http://francisshanahan.com/index.php/2009/how-to-build-a-8x8x3-led-matrix-with-pwm-using-an-arduino> Acesso em: 25 jun. 2013

Luz e Led. Disponível em: <http://luzeled.com.br/luz> Acesso em: 22 jul. 2013

Luz e Led. Disponível em: <http://www.laguerradelosmundos.net/sistema-de-color-

rgb> Acesso em: 4 ago. 2013

Macao Communications. Disponível em: <http://macao.communications.museum/por/exhibition/secondfloor/moreinfo/FlipFlop.html> Acesso em: 11 mai. 2013

MadeInAsia. Disponível em: <http://www.madeinasia.com/led-display/LED-Display-325577.html> Acesso em: 4 abr. 2013

MSPC. Disponível em: <http://www.mspc.eng.br/eletrn/semic_210.shtml> Acesso em: 21 mai. 2013

Oficina Brasil Virtual. Disponível em: <http://oficinabrasilvirtual.blogspot.com.br/2010/12/led-diodo-emissor-de-luz.html> Acesso em: 18 jul. 2013

Thebox. Disponível em: <http://www.thebox.myzen.co.uk/Workshop/LED_Matrix.html> Acesso em: 15 mai. 2013

Unicamp. Disponível em: <http://www.iar.unicamp.br/lab/luz/dicasemail/led/dica36.htm> Acesso em: 20 mai. 2013

Hobbyist. Disponível em: <http://www.hobbyist.co.nz/?q=taxonomy/term/1> Acesso em: 11 abr. 2013

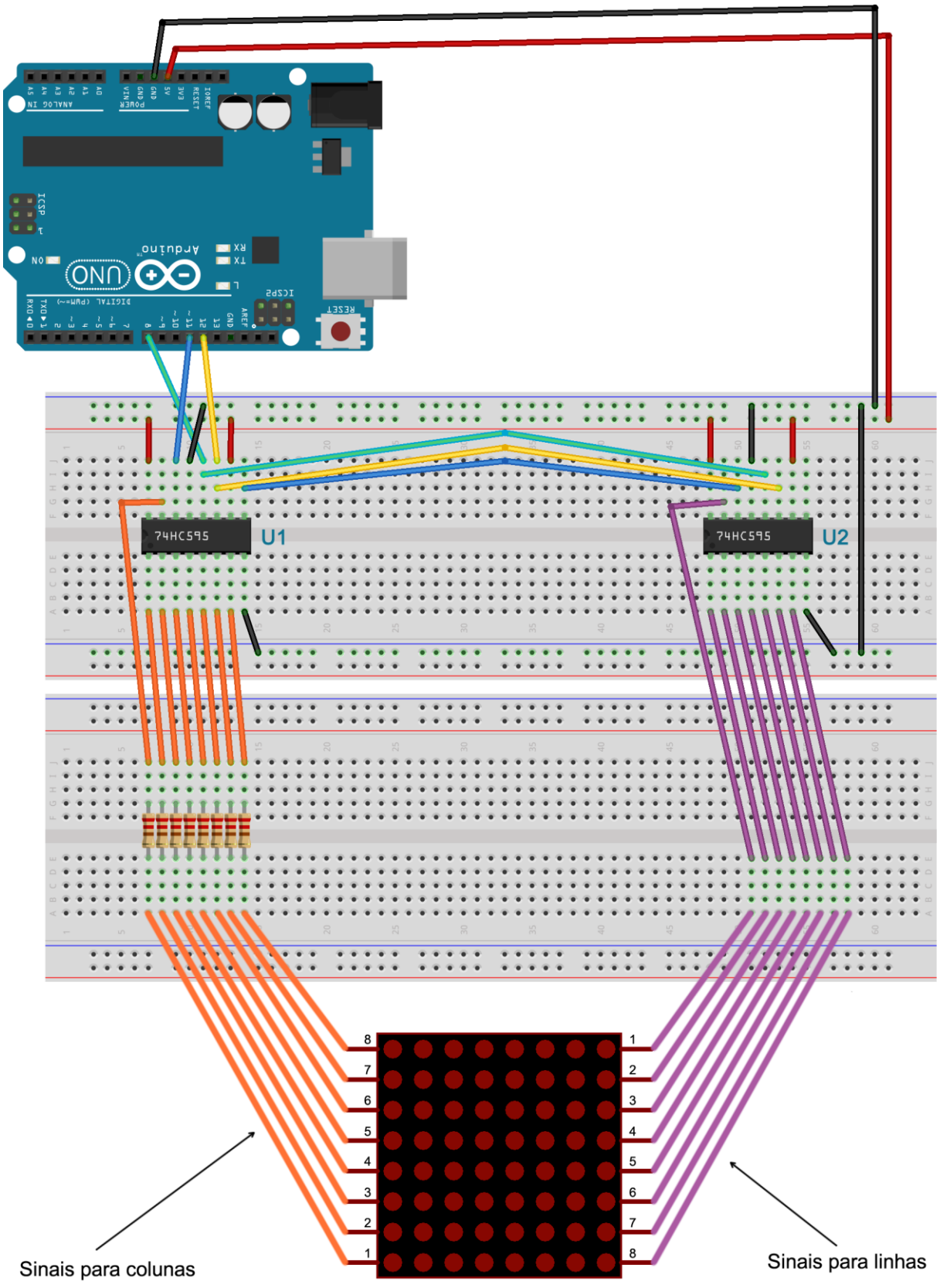
Commons. Disponível em: <http://commons.wikimedia.org/wiki/File:09-02-06-RRS-PigScreen.jpg> Acesso em: 11 abr. 2013

Commons. Disponível em: <https://commons.wikimedia.org/wiki/File:RIP_Data_Flow.svg> Acesso em: 11 abr. 2013

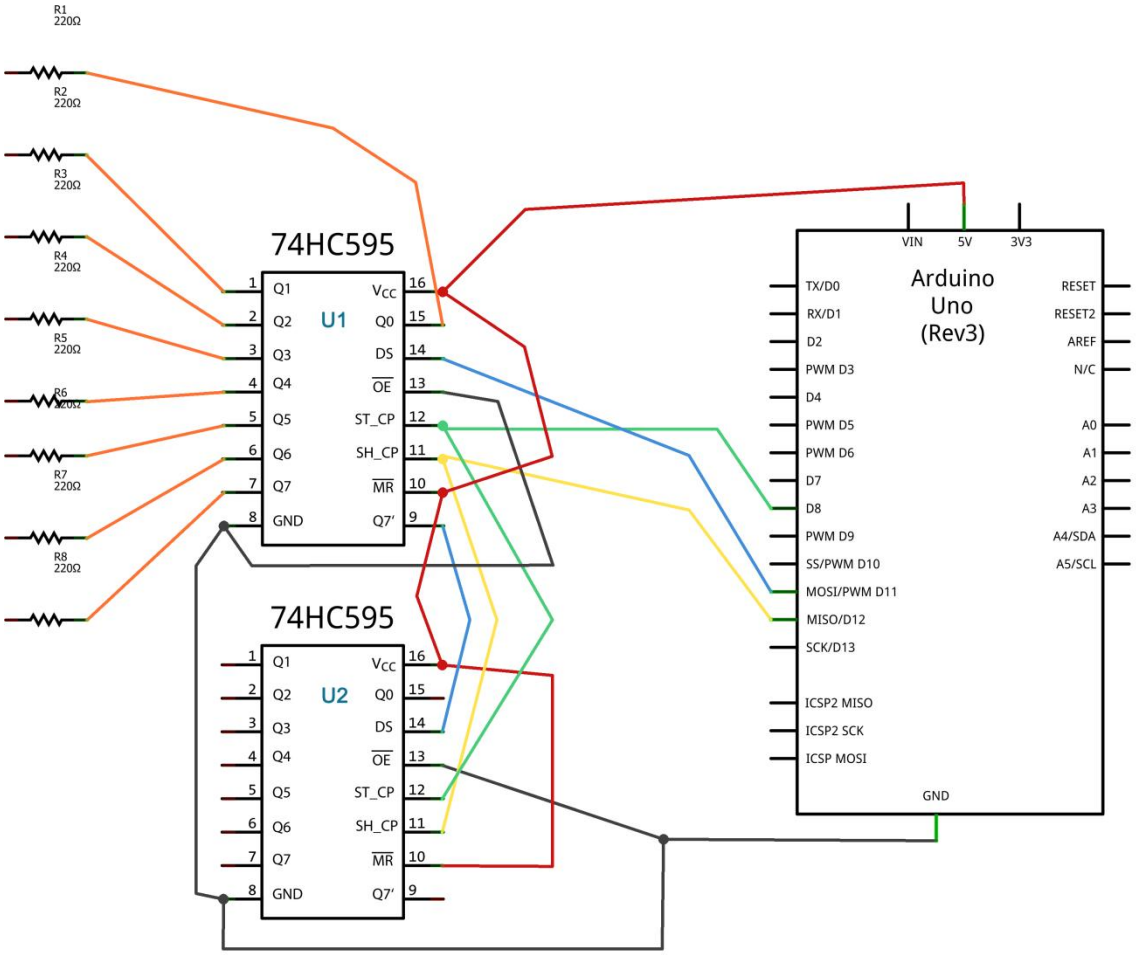
Commons. Disponível em: <http://commons.wikimedia.org/wiki/File:Square_wave_closeup.png> Acesso em: 4 ago. 2013

Commons. Disponível em: <http://commons.wikimedia.org/wiki/File:Rgb-raster-image.svg> Acesso em: 11 abr. 2013

APÊNDICE A – SIMULAÇÃO DO HARDWARE



APÊNDICE B – DIAGRAMA ESQUEMÁTICO DO HARDWARE



APÊNDICE C – PROGRAMA PARA CONTROLAR O PAINEL

```
#include <avr/pgmspace.h>
#include <TimerOne.h>

int starter = 1;
char charMessage[40];
int latchPin = 8;
int clockPin = 12;
int dataPin = 11;
byte buffer[8];
static byte font[][8] PROGMEM = {

  {
   B00000000, B00000000, B00000000, B00000000, B00000000, B00000000,
   B00000000, B00000000}
  ,
  {
   B00000100, B00000100, B00000100, B00000100, B00000100, B00000100,
   B00000000, B00000100}
  ,
  {
   B00001010, B00001010, B00001010, B00000000, B00000000, B00000000,
   B00000000, B00000000}
  ,
  {
   B00000000, B00001010, B00011111, B00001010, B00011111, B00001010,
   B00011111, B00001010}
  ,
  {
   B00000111, B00001100, B00010100, B00001100, B00000110, B00000101,
   B00000110, B00011100}
  ,
  {
   B00011001, B00011010, B00000010, B00000100, B00000100, B00001000,
   B00001011, B00010011}
  ,
  {
   B00000110, B00001010, B00010010, B00010100, B00001001, B00010110,
   B00010110, B00001001}
  ,
  {
   B00000100, B00000100, B00000100, B00000000, B00000000, B00000000,
   B00000000, B00000000}
  ,
  {
   B00000010, B00000100, B00001000, B00001000, B00001000, B00001000,
   B00000100, B00000010}
  ,
  {
   B00001000, B00000100, B00000010, B00000010, B00000010, B00000010,
   B00000100, B00001000}
  ,
  {
   B00010101, B00001110, B00011111, B00001110, B00010101, B00000000,
   B00000000, B00000000}
  ,
  {
   B00000000, B00000000, B00000100, B00000100, B00011111, B00000100,
   B00000100, B00000000}
  ,
```

```
{
 B00000000, B00000000, B00000000, B00000000, B00000000, B00000110,
 B00000100, B00001000}
,
{
 B00000000, B00000000, B00000000, B00000000, B00001110, B00000000,
 B00000000, B00000000}
,
{
 B00000000, B00000000, B00000000, B00000000, B00000000, B00000000,
 B00000000, B00000100}
,
{
 B00000001, B00000010, B00000010, B00000100, B00000100, B00001000,
 B00001000, B00010000}
,
{
 B00001110, B00010001, B00010011, B00010001, B00010101, B00010001,
 B00011001, B00001110}
,
{
 B00000100, B00001100, B00010100, B00000100, B00000100, B00000100,
 B00000100, B00011111}
,
{
 B00001110, B00010001, B00010001, B00000010, B00000100, B00001000,
 B00010000, B00011111}
,
{
 B00001110, B00010001, B00000001, B00001110, B00000001, B00000001,
 B00010001, B00001110}
,
{
 B00010000, B00010000, B00010100, B00010100, B00011111, B00000100,
 B00000100, B00000100}
,
{
 B00011111, B00010000, B00010000, B00011110, B00000001, B00000001,
 B00000001, B00011110}
,
{
 B00000111, B00001000, B00010000, B00011110, B00010001, B00010001,
 B00010001, B00001110}
,
{
 B00011111, B00000001, B00000001, B00000001, B00000010, B00000100,
 B00001000, B00010000}
,
{
 B00001110, B00010001, B00010001, B00001110, B00010001, B00010001,
 B00010001, B00001110}
,
{
 B00001110, B00010001, B00010001, B00001111, B00000001, B00000001,
 B00000001, B00000001}
,
{
 B00000000, B00000100, B00000100, B00000000, B00000000, B00000100,
 B00000100, B00000000}
,
{
```

```
 B00000000, B00000100, B00000100, B00000000, B00000000, B00000100,
 B00000100, B00001000}
,
{
 B00000001, B00000010, B00000100, B00001000, B00001000, B00000100,
 B00000010, B00000001}
,
{
 B00000000, B00000000, B00000000, B00011110, B00000000, B00011110,
 B00000000, B00000000}
,
{
 B00010000, B00001000, B00000100, B00000010, B00000010, B00000100,
 B00001000, B00010000}
,
{
 B00001110, B00010001, B00010001, B00000010, B00000100, B00000100,
 B00000000, B00000100}
,
{
 B00001110, B00010001, B00010001, B00010101, B00010101, B00010001,
 B00010001, B00011110}
,
{
 B00001110, B00010001, B00010001, B00010001, B00011111, B00010001,
 B00010001, B00010001}
,
{
 B00011110, B00010001, B00010001, B00011110, B00010001, B00010001,
 B00010001, B00011110}
,
{
 B00000111, B00001000, B00010000, B00010000, B00010000, B00010000,
 B00001000, B00000111}
,
{
 B00011100, B00010010, B00010001, B00010001, B00010001, B00010001,
 B00010010, B00011100}
,
{
 B00011111, B00010000, B00010000, B00011110, B00010000, B00010000,
 B00010000, B00011111}
,
{
 B00011111, B00010000, B00010000, B00011110, B00010000, B00010000,
 B00010000, B00010000}
,
{
 B00001110, B00010001, B00010000, B00010000, B00010111, B00010001,
 B00010001, B00001110}
,
{
 B00010001, B00010001, B00010001, B00011111, B00010001, B00010001,
 B00010001, B00010001}
,
{
 B00011111, B00000100, B00000100, B00000100, B00000100, B00000100,
 B00000100, B00011111}
,
{
 B00011111, B00000100, B00000100, B00000100, B00000100, B00000100,
```

```
 B00010100, B00001000}
,
{
 B00010001, B00010010, B00010100, B00011000, B00010100, B00010010,
 B00010001, B00010001}
,
{
 B00010000, B00010000, B00010000, B00010000, B00010000, B00010000,
 B00010000, B00011111}
,
{
 B00010001, B00011011, B00011111, B00010101, B00010001, B00010001,
 B00010001, B00010001}
,
{
 B00010001, B00011001, B00011001, B00010101, B00010101, B00010011,
 B00010011, B00010001}
,
{
 B00001110, B00010001, B00010001, B00010001, B00010001, B00010001,
 B00010001, B00001110}
,
{
 B00011110, B00010001, B00010001, B00011110, B00010000, B00010000,
 B00010000, B00010000}
,
{
 B00001110, B00010001, B00010001, B00010001, B00010001, B00010101,
 B00010011, B00001111}
,
{
 B00011110, B00010001, B00010001, B00011110, B00010100, B00010010,
 B00010001, B00010001}
,
{
 B00001110, B00010001, B00010000, B00001000, B00000110, B00000001,
 B00010001, B00001110}
,
{
 B00011111, B00000100, B00000100, B00000100, B00000100, B00000100,
 B00000100, B00000100}
,
{
 B00010001, B00010001, B00010001, B00010001, B00010001, B00010001,
 B00010001, B00001110}
,
{
 B00010001, B00010001, B00010001, B00010001, B00010001, B00010001,
 B00001010, B00000100}
,
{
 B00010001, B00010001, B00010001, B00010001, B00010001, B00010101,
 B00010101, B00001010}
,
{
 B00010001, B00010001, B00001010, B00000100, B00000100, B00001010,
 B00010001, B00010001}
,
{
 B00010001, B00010001, B00001010, B00000100, B00000100, B00000100,
 B00000100, B00000100}
```

```
,
{
 B00011111, B00000001, B00000010, B00000100, B00001000, B00010000,
 B00010000, B00011111}
,
{
 B00001110, B00001000, B00001000, B00001000, B00001000, B00001000,
 B00001000, B00001110}
,
{
 B00010000, B00001000, B00001000, B00000100, B00000100, B00000010,
 B00000010, B00000001}
,
{
 B00001110, B00000010, B00000010, B00000010, B00000010, B00000010,
 B00000010, B00001110}
,
{
 B00000100, B00001010, B00010001, B00000000, B00000000, B00000000,
 B00000000, B00000000}
,
{
 B00000000, B00000000, B00000000, B00000000, B00000000, B00000000,
 B00000000, B00011111}
,
{
 B00001000, B00000100, B00000000, B00000000, B00000000, B00000000,
 B00000000, B00000000}
,
{
 B00000000, B00000000, B00000000, B00001110, B00010010, B00010010,
 B00010010, B00001111}
,
{
 B00000000, B00010000, B00010000, B00010000, B00011100, B00010010,
 B00010010, B00011100}
,
{
 B00000000, B00000000, B00000000, B00001110, B00010000, B00010000,
 B00010000, B00001110}
,
{
 B00000000, B00000001, B00000001, B00000001, B00000111, B00001001,
 B00001001, B00000111}
,
{
 B00000000, B00000000, B00000000, B00011100, B00010010, B00011110,
 B00010000, B00001110}
,
{
 B00000000, B00000011, B00000100, B00000100, B00000110, B00000100,
 B00000100, B00000100}
,
{
 B00000000, B00001110, B00001010, B00001010, B00001110, B00000010,
 B00000010, B00001100}
,
{
 B00000000, B00010000, B00010000, B00010000, B00011100, B00010010,
 B00010010, B00010010}
,
```

```
{
 B00000000, B00000000, B00000100, B00000000, B00000100, B00000100,
 B00000100, B00000100}
,
{
 B00000000, B00000010, B00000000, B00000010, B00000010, B00000010,
 B00000010, B00001100}
,
{
 B00000000, B00010000, B00010000, B00010100, B00011000, B00011000,
 B00010100, B00010000}
,
{
 B00000000, B00010000, B00010000, B00010000, B00010000, B00010000,
 B00010000, B00001100}
,
{
 B00000000, B00000000, B00000000, B00001010, B00010101, B00010001,
 B00010001, B00010001}
,
{
 B00000000, B00000000, B00000000, B00010100, B00011010, B00010010,
 B00010010, B00010010}
,
{
 B00000000, B00000000, B00000000, B00001100, B00010010, B00010010,
 B00010010, B00001100}
,
{
 B00000000, B00011100, B00010010, B00010010, B00011100, B00010000,
 B00010000, B00010000}
,
{
 B00000000, B00001110, B00010010, B00010010, B00001110, B00000010,
 B00000010, B00000001}
,
{
 B00000000, B00000000, B00000000, B00001010, B00001100, B00001000,
 B00001000, B00001000}
,
{
 B00000000, B00000000, B00001110, B00010000, B00001000, B00000100,
 B00000010, B00011110}
,
{
 B00000000, B00010000, B00010000, B00011100, B00010000, B00010000,
 B00010000, B00001100}
,
{
 B00000000, B00000000, B00000000, B00010010, B00010010, B00010010,
 B00010010, B00001100}
,
{
 B00000000, B00000000, B00000000, B00010001, B00010001, B00010001,
 B00001010, B00000100}
,
{
 B00000000, B00000000, B00000000, B00010001, B00010001, B00010001,
 B00010101, B00001010}
,
{
```

```
   B00000000, B00000000, B00000000, B00010001, B00001010, B00000100,
   B00001010, B00010001}
  ,
  {
   B00000000, B00000000, B00010001, B00001010, B00000100, B00001000,
   B00001000, B00010000}
  ,
  {
   B00000000, B00000000, B00000000, B00011111, B00000010, B00000100,
   B00001000, B00011111}
  ,
  {
   B00000010, B00000100, B00000100, B00000100, B00001000, B00000100,
   B00000100, B00000010}
  ,
  {
   B00000100, B00000100, B00000100, B00000100, B00000100, B00000100,
   B00000100, B00000100}
  ,
  {
   B00001000, B00000100, B00000100, B00000100, B00000010, B00000100,
   B00000100, B00001000}
  ,
  {
   B00000000, B00000000, B00000000, B00001010, B00011110, B00010100,
   B00000000, B00000000}
};

/**
  Objetivo: limpar todo o contéudo da matriz de LED
  Entrada: nenhuma
  Retorno: nenhum
**/
void
clearDisplay ()
{
  for (byte x = 0; x < 8; x++)
    {
      buffer[x] = B00000000;
    }
  screenUpdate ();
}

/**
  Objetivo: montar os mapas de bits no buffer
  Entrada: Um array de char com a mensagem de entrada
           e um inteiro referente a velocidade de rolagem do texto
  Retorno: nenhum
**/
void
scroll (char myString[], int speed)
{
  byte firstChrRow, secondChrRow;
  byte ledOutput;
  byte chrPointer = 0;
  byte Char1, Char2;
  byte scrollBit = 0;
  byte strLength = 0;
  unsigned long time;
  unsigned long counter;
  while (myString[strLength])
```

```
    {
      strLength++;
    }
  counter = millis ();
  while (chrPointer < (strLength - 1))
    {
      time = millis ();
      if (time > (counter + speed))
      {
        Char1 = myString[chrPointer];
        Char2 = myString[chrPointer + 1];
        for (byte y = 0; y < 8; y++)
          {
            firstChrRow = pgm_read_byte (&font[Char1 - 32][y]);
            secondChrRow = (pgm_read_byte (&font[Char2 - 32][y])) << 1;
            ledOutput =
            (firstChrRow << scrollBit) | (secondChrRow >>
                                          (8 - scrollBit));
            buffer[y] = ledOutput;
          }
        scrollBit++;
        if (scrollBit > 6)
          {
            scrollBit = 0;
            chrPointer++;
          }
        counter = millis ();
      }
    }
}

/**
  Objetivo: Atualizar a matriz de LED de acordo os mapas de bits do buffer
  Entrada: nenhuma
  Retorno: nenhum
**/
void
screenUpdate ()
{
  byte row = B10000000;
  for (byte k = 0; k < 9; k++)
    {
      digitalWrite (latchPin, LOW);
      shiftIt (~buffer[k]);
      shiftIt (row);
      digitalWrite (latchPin, HIGH);
      row = row >> 1;
    }
}

/**
  Objetivo: Gera os sinais digitais para os registradores de deslocamento
            a partir de uma determinada linha do buffer
  Entrada: uma variável do tipo byte correspondente a uma linha do buffer
  Retorno: nenhum
**/
void
shiftIt (byte dataOut)
{
  boolean pinState;
  digitalWrite (dataPin, LOW);
```

```
  for (int i = 0; i < 8; i++)
    {
      digitalWrite (clockPin, LOW);
      if (dataOut & (1 << i))
      {
        pinState = HIGH;
      }
      else
      {
        pinState = LOW;
      }
      digitalWrite (dataPin, pinState);
      digitalWrite (clockPin, HIGH);
      digitalWrite (dataPin, LOW);
    }
  digitalWrite (clockPin, LOW);
}

/**
  Objetivo: função padrão utlizada para fazer configurações iniciais
  Entrada: nenhuma
  Retorno: nenhum
**/
void
setup ()
{
  Serial.begin (9600);
  pinMode (latchPin, OUTPUT);
  pinMode (clockPin, OUTPUT);
  pinMode (dataPin, OUTPUT);
  Timer1.initialize (10000);
  Timer1.attachInterrupt (screenUpdate);
}

/**
  Objetivo: função padrão que permanece em um laço infinito.
            Permite mudar de estado em tempo de execução.
  Entrada: nenhuma
  Retorno: nenhum
**/
void
loop ()
{
  clearDisplay ();
  scroll (" TCC - PAINEL DE LED ", 70);
}
```

APÊNDICE D – DIAGRAMA ESQUEMÁTICO PARA UM PAINEL RBG