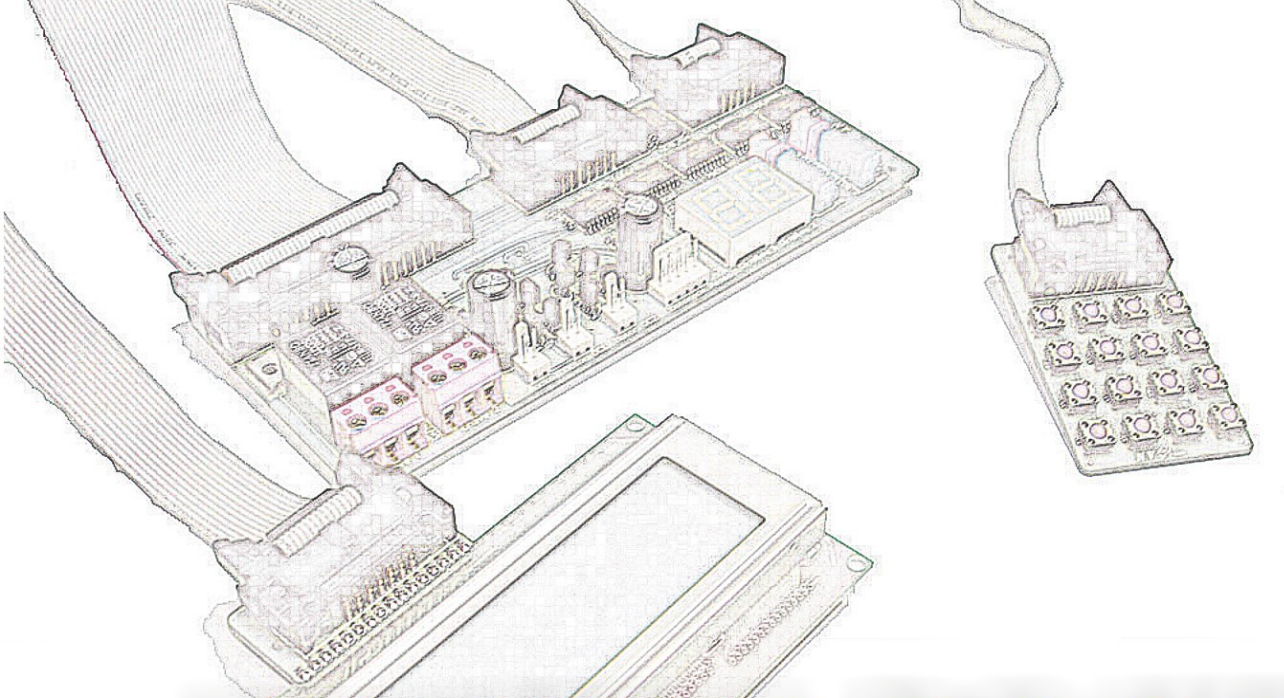
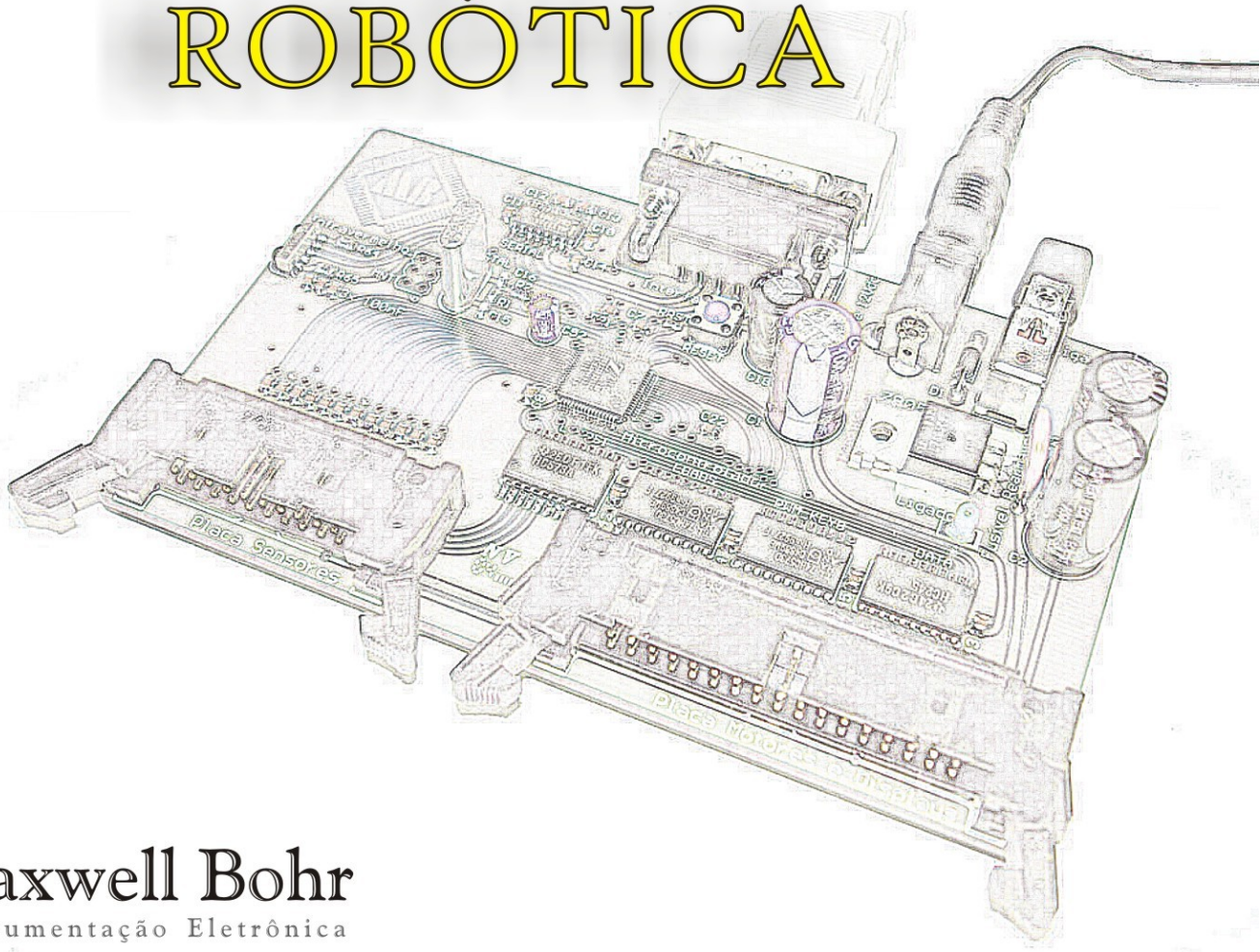


MANUAL DO KIT DIDÁTICO DE ROBÓTICA



Maxwell Bohr
Instrumentação Eletrônica

http://www.maxwellbohr.com.br
contato@maxwellbohr.com.br

Sumário

Introdução

Esse documento tem como objetivo guiar o usuário durante a montagem e o uso do Kit Didático de Robótica. Estão descritas em detalhes as partes que compõem o equipamento, de modo a permitir a sua montagem e o entendimento de todas as suas funcionalidades.

Além da descrição física e funcional do equipamento, também será visto como criar programas que sejam capazes de controlar o Kit Didático de Robótica a partir de um computador. Para isso, serão apresentados vários projetos práticos e a partir deles serão discutidos os vários tópicos relacionados à programação do Kit.

Com esses conhecimentos já será possível criar projetos muito interessantes de automação e robótica, tendo como limitante, apenas a criatividade de quem está desenvolvendo os projetos.

O Kit Didático de Robótica

O Kit Didático de Robótica é um sistema projetado para auxiliar o aprendizado de automação e robótica. Seu principal objetivo é permitir a criação de projetos práticos que possam ser utilizados como base para o estudo das diversas áreas do conhecimento envolvidas nas atividades de automação e robótica, principalmente eletrônica, mecânica e programação.

Basicamente, o Kit Didático de Robótica é um conjunto de módulos que possibilita expandir a capacidade de interação do computador com o ambiente. Ele é conectado ao computador através de uma porta serial e possibilita que o computador leia parâmetros externos tais como temperatura, luminosidade, peso, entre outros. Além dessas leituras, é possivel também que se atue no ambiente, por exemplo, através do acionamento de algum motor ou display.

As operações realizadas pelo Kit são controladas por programas que rodam no computador. Esses programas enviam comandos de controle para o Kit através da porta serial do computador. São esses programas que possuem toda a lógica de controle em qualquer projeto.

Para permitir uma maior flexibilidade, de modo que o Kit pudesse ser utilizado em uma grande gama de projetos, não foi feito nenhum vínculo lógico a nível de hardware entre os ítens que compõe o Kit. Esses vínculos são criados exclusivamente no programa de controle no computador, conforme a necessidade de cada projeto.

Por exemplo, um vínculo que pode ser criado em um programa de controle é o de que um dos motores controlados pelo Kit seja ligado assim que o valor da leitura do sensor de temperatura ultrapasse certo valor. Para o Kit Didático de Robótica o motor e o sensor de temperatura são dois ítens totalmente independentes, sem nenhum vínculo. Se não houver um programa de controle que crie um vínculo entre os dois, a variação na temperatura não influi de nenhuma maneira no funcionamento do motor.

No entanto podemos criar um programa de controle, que roda no computador, para monitorar a leitura do sensor de temperatura e quando este ultrapassar um certo valor, o programa envia um comando para o Kit solicitando que um motor seja ligado. Dessa forma teríamos um vínculo entre o motor e o sensor de temperatura, no entanto, esse vínculo existiria apenas em função do programa de controle. Uma vez esse programa fosse interrompido o vínculo deixaria de existir.

Em outros programas poderiam ser criados vínculos diferentes. Isso permite que o Kit

seja utilizado nos mais diversos projetos sendo necessário apenas adaptar a lógica no programa de controle. De certa forma podemos considerar que o Kit Didático de Robótica fornece várias "pecinhas" que podem ser unidas no programa de controle de modo a permitir a criação das mais diversas aplicações.

Assim, utilizando o poder de processamento dos computadores aliado às capacidades extras que o Kit Didático de Robótica fornece, é possível criar desde projetos simples até projetos de alta complexidade para o ensino de automação e robótica. Veremos agora as partes que compõe o Kit Didático de Robótica.

Composição do Kit Didático de Robótica

O Kit Didático de Robótica foi projetado de modo que fosse um sistema modular e permitisse expansões. O sistema possui um *Módulo Principal* que possui uma conexão com o computador através de um cabo serial e dois conectores para a conexão de módulos de expansão. Atualmente existem quatro módulos de expansão disponíveis, são eles:

- **Módulo de Sensores**: Possibilita a leitura de parâmetros através de vários sensores. É de fácil uso e não exige conhecimentos avançados. Ideal para um primeiro momento de aprendizagem ou montagens rápidas que não necessitem de sensores diferentes dos oferecidos por esse módulo, uma vez que seus sensores já são pré-estabelecidos.

- **Módulo de Sensores Genérico**: Possui várias entradas genéricas para sensores. É necessário algum conhecimento de eletrônica para utilizar essas entradas, no entanto, permite grande flexibilidade na escolha dos sensores, uma vez que esses não são pré-estabelecidos e podem ser escolhidos conforme a necessidade. Ideal para montagens mais complexas ou avançadas.

- **Módulo de Motores e Displays**: Possui vários atuadores e displays. É de fácil uso e não exige conhecimentos avançados. Ideal para um primeiro momento de aprendizagem ou montagens que não necessitem de atuadores ou displays diferentes dos oferecidos por esse módulo uma vez que ele não é muito flexível para a adição de ítens extras.

- **Módulo de Entradas, Saídas e Servo-Motores**: Possui portas para entradas e saídas digitais, conectores para conexão de servo-motores e conexão para um display de cristal líquido. O uso pleno desse módulo exige algum conhecimento de eletrônica. No entanto, apesar desse requisito, assim como ocorre com o Módulo de Sensores Genérico, esse módulo é muito flexivel. Ele permite a adição e o controle de ítens extras conforme a necessidade dos projetos.

A figura a seguir mostra o *Módulo Principal* do Kit Didático de Robótica com o *Módulo de Sensores* e o *Módulo de Motores e Displays* totalmente montados e com todos os seus ítens.

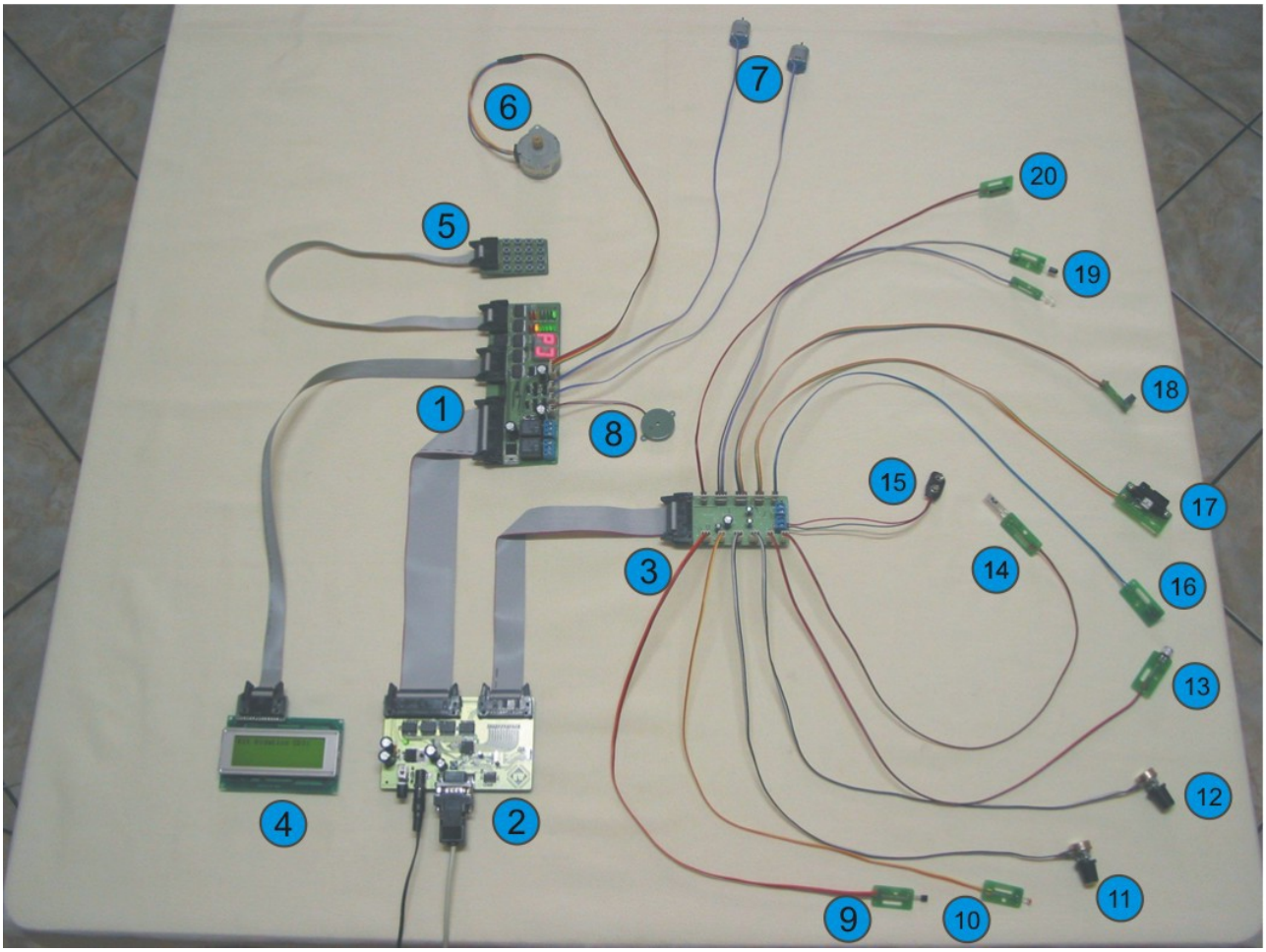


Figura 1. Módulo Principal, Módulo de Sensores e Módulo de Motores e Displays montados e com todos os seus ítens. (1) Módulo Principal, (2) Módulo de Motores e Displays, (3) Módulo de Sensores, (4) Display de Cristal Líquido, (5) Teclado, (6) Motor de Passo, (7) Motores DC, (8) Buzzer(Audio), (9) Sensor de Temperatura, (10) Sensor de luminosidade, (11) Potenciômetro I, (12) Potenciômetro II, (13) Microfone, (14) Sensor de vibração, (15) Clip para bateria, (16) Sensor de Peso, (17) Sensor de distância, (18) Par óptico I, (19) Par óptico II, (20) Chave magnética.

Os tópicos a seguir descrevem detalhadamente todos os módulos que compõe o Kit Didático de Robótica.

Módulo Principal

No *Módulo Principal* se encontra o "cérebro" do equipamento. Esse módulo possui um microcontrolador, que é um componente eletrônico que pode ser considerado como um mini computador. Esse componente executa um programa, denominado firmware, que fica gravado em sua memória interna. Esse programa interpreta os comandos enviados por um computador através da porta serial e faz com que o Kit responda adequadamente.

Uma funcionalidade interessante que esse módulo permite é a atualização do firmware. Essa operação poderia ser comparada a uma atualização de BIOS nos computadores. Isso permite correção de falhas e adição de novas funcionalidades, como por exemplo o suporte à algum módulo de expansão novo.

Existe um botão de reset que reinicia a execução do firmware e funciona de forma

similar ao botão de reset dos computadores. Ele pode ser utilizado para forçar que o Kit retorne ao seu estado inicial sem a necessidade de desligar e ligar o equipamento.

Para possibilitar a comunicação entre um computador e esse módulo é necessário que os dois estejam conectados por um cabo serial. Para isso, existe um conector para cabos seriais nesse módulo. Encontramos também uma entrada para fonte de energia e ao seu lado um botão de liga e desliga. Um LED indica se o equipamento está ligado ou desligado.

Outra funcionalidade que o *Módulo Principal* oferece é a comunicação sem fio através de uma porta infravermelho (IrDA). Utilizando essa porta é possível que o Kit Didático de Robótica seja controlado remotamente sem o uso de fios. Para isso é necessário enviar os comandos de controle a partir de algum outro dispositivo que possua uma porta infravermelho. Alguns exemplos de dispositivo que podem ser utilizados para enviar comandos remotamente são computadores de mão, notebooks, computadores de mesa que possuam portas infravermelho e até mesmo outro Kit.

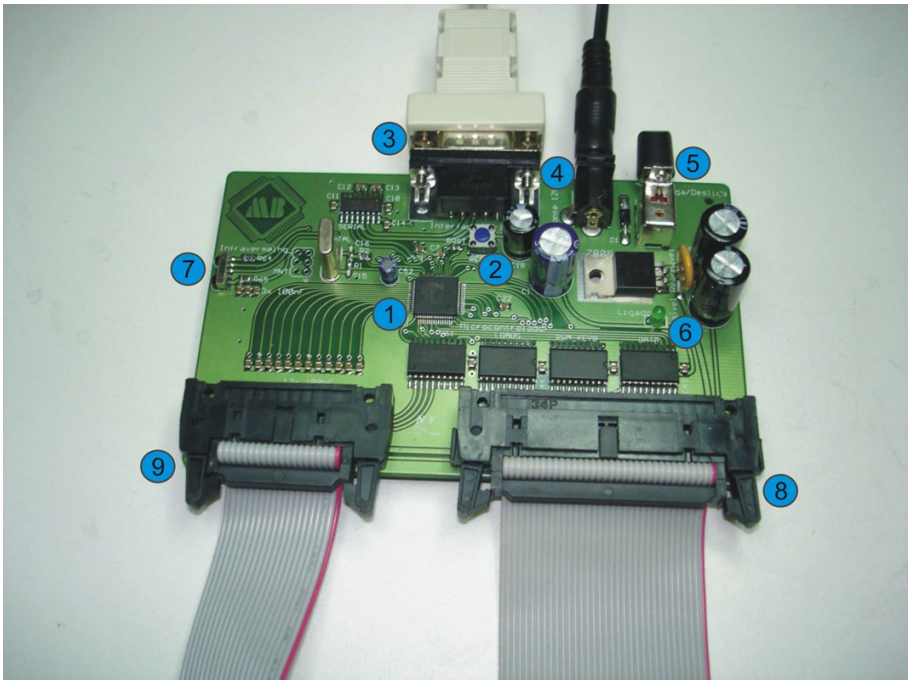


Figura 2. *Módulo Principal.* (1) Microcontrolador, (2) Botão de reset, (3) Conector para cabo serial, (4) Conector da fonte de energia, (5) Botão liga/desliga, (6) LED de indicação de liga/desliga, (7) Porta de comunicação infravermelho, (8) Conector para cabo flat de 34 vias, (9) Conector para cabo flat de 20 vias.

Por fim, para permitir a conexão dos módulos de expansão, o *Módulo Principal* possui dois conectores, um para cabos flat de 20 vias e outro para cabos flat de 34 vias. Dessa forma podemos ter dois módulos de expansão conectados ao mesmo tempo.

Módulo de Sensores

O M*ódulo de Sensores* faz a interface entre o *Módulo Principal* e vários sensores. Ele possui uma placa com vários conectores para os sensores e um conector para cabo flat de 20 vias que permite sua conexão com o *Módulo Principal*. Próximo de todos os conectores existe um texto

na placa especificando que sensor deve ser ligado no conector. Esse texto também existe nas placas dos sensores, de modo que a montagem seja facilitada até que se tenha uma maior familiaridade dos conectores corretos para cada sensor. Na figura a seguir podemos ver o M*ódulo de Sensores* com todos os seus ítens.

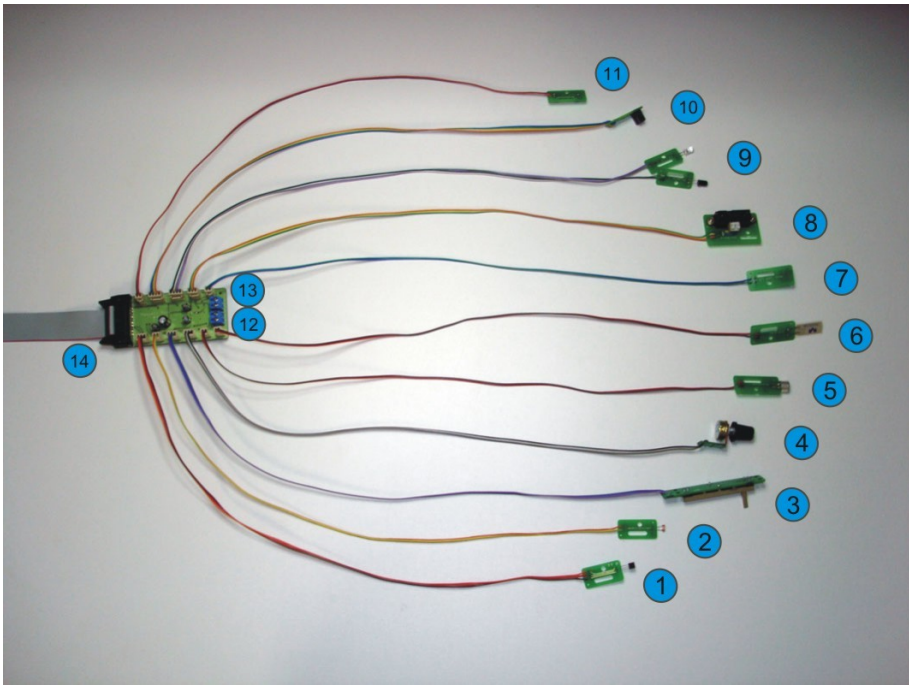


Figura 3. *Módulo de Sensores.* (1) Sensor de Temperatura, (2) Sensor de Luminosidade, (3) Potenciômetro Linear, (4) Potenciômetro Angular, (5) Microfones, (6) Sensor de Vibração, (7) Sensor de Peso, (8) Sensor de Distância, (9) Par Óptico I, (10) Par Óptico II, (11) Chave Magnética, (12) Entrada auxiliar I, (13) Entrada auxiliar II, (14) Conector flat de 20 vias para conexão com o *Módulo Principal.*

Basicamente a função desse módulo é adequar e padronizar o sinal de diferentes sensores analógicos para que eles possam ser lidos no *Módulo Principal*. É no microcontrolador do *Módulo Principal* que o sinal elétrico dos sensores, já adequado pelos componentes eletrônicos do *Módulo de Sensores* é lido. Para isso, esse sinal deve ser convertido de uma tensão para um valor digital de 10 bits. Essa conversão é necessária porque tanto o microcontrolador como o computador conseguem processar apenas valores digitais. O componente que faz essa conversão se chama "conversor analógico-digital" ou simplesmente conversor AD, e faz parte do microcontrolador do *Módulo Principal*.

Conversores AD transformam uma grandeza analógica, no caso a tensão elétrica, em um valor digital discreto. A faixa de tensão de leitura do conversor AD do microcontrolador do *Módulo Principal* é de 0 à 2,2 Volts, isso quer dizer que ele pode converter apenas tensões nessa faixa. Por isso, existe a necessidade de que o *Módulo de Sensores* converta o sinal de todos os sensores para um valor tensão nessa faixa. Outra característica do conversor AD utilizado é que ele possui uma resolução de 10 bits, o que possibilita a representação de 1024 estados diferentes (2^{10}). Dessa forma, podemos representar 1024 níveis de tensão diferentes. Assim teremos o valor mínimo de tensão representado por 0 e o máximo por 1023. Os valores intermediários representam os níveis de tensão entre 0 e 2,2 Volts, sendo que cada nível representa uma variação de 2,2/1024 Volts, ou seja, 2,15 milivolts aproximadamente. Este valor é aproximado devido a variações da escala de leitura do

conversor AD do microcontrolador, que pode variar até 10% para mais ou para menos.

O significado do valor lido pelo conversor AD irá depender do sensor lido. Por exemplo, no sensor de luminosidade, o valor indica o grau de luminosidade do ambiente onde o sensor se encontra. Um valor 0 indica escuridão total e conforme o grau de luminosidade aumenta o valor da leitura aumenta até um ponto em que a luminosidade está muito grande e satura a leitura do sensor, que irá responder um valor 1023. Outro exemplo é o valor de leitura dos potenciômetros. Essa leitura indicam a posição em que o knob do potenciômetro se encontra, 0 no início do curso e 1023 no fim.

Além de entradas para os sensores analógicos, ainda existem três entradas para sensores digitais. Esses sensores possuem sinais bem definidos, 0 ou 1 e não precisam ser lidor por um conversor AD pois ja possuem um sinal digital.

Para conhecer os sensores e permitir a interpretação do significado de suas respostas é feita a seguir uma descrição de todos os sensores que podem ser conectados no *Módulo de Sensores*.

Sensor de Temperatura

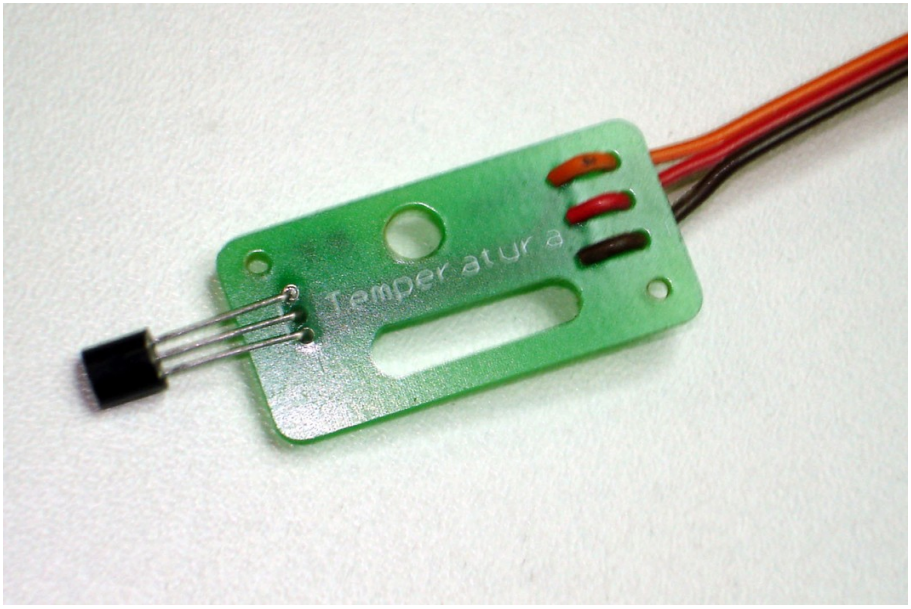


Figura 4. Sensor de temperatura.

A temperatura é um parâmetro muito importante em inúmeras aplicações práticas. É muito utilizada no monitoramento ambiental e metereológico. Na indústria esse é um dos parâmetros mais utilizados no monitoramento de diversos tipos de processos. Mesmo em casa, a temperatura influencia muito no dia a dia das pessoas e é monitorada em diversos equipamentos cotidianos tais como computadores, geladeiras, celulares, fornos, entre outros.

O sensor de temperatura do Kit pode medir temperatura no intervalo de -40ºC à 125ºC. Seu sinal apresenta um comportamento linear e quanto menor esse valor menor é a temperatura. Uma fórmula aproximada de conversão da resposta do conversor AD para um valor em graus

Celsius (ºC) é a dada a seguir:

$$T = (X-296)/4,4$$

Onde:

T = Temperatura em ºC.

X = Valor de tensão convertido pelo conversor AD (0-1023).

Substituindo o valor de X na fórmula pelo valor lido pelo conversor AD vamos obter um valor de temperatura em graus Celsius correspondente. Podemos fazer a operação contrária também e converter um valor em graus Celsius para o valor que seria lido pelo conversor AD. É importante ressaltar que esse valor é aproximado devido a inúmeros fatores que influenciam a leitura. A seguir um gráfico com a resposta típica desse sensor.

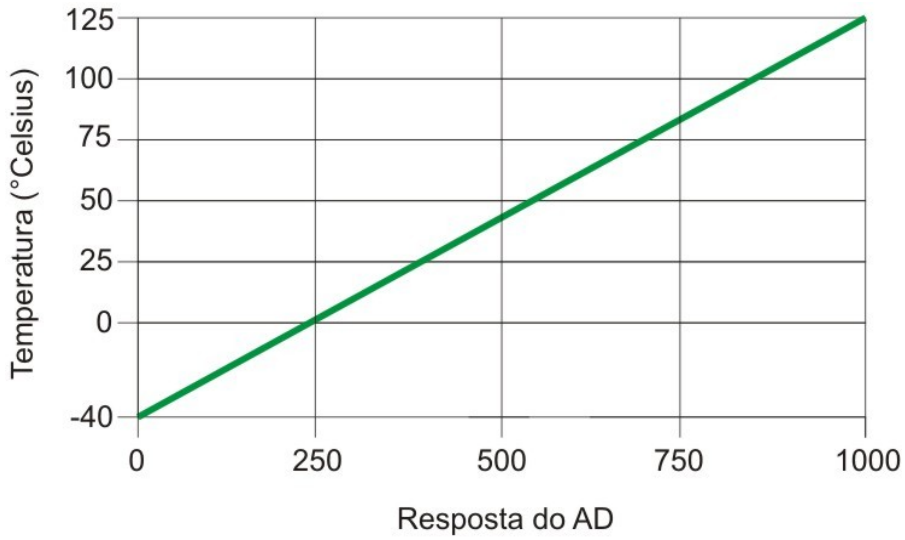


Figura 5. Gráfico de resposta do sensor de tempetatura.

Sensor de Luminosidade

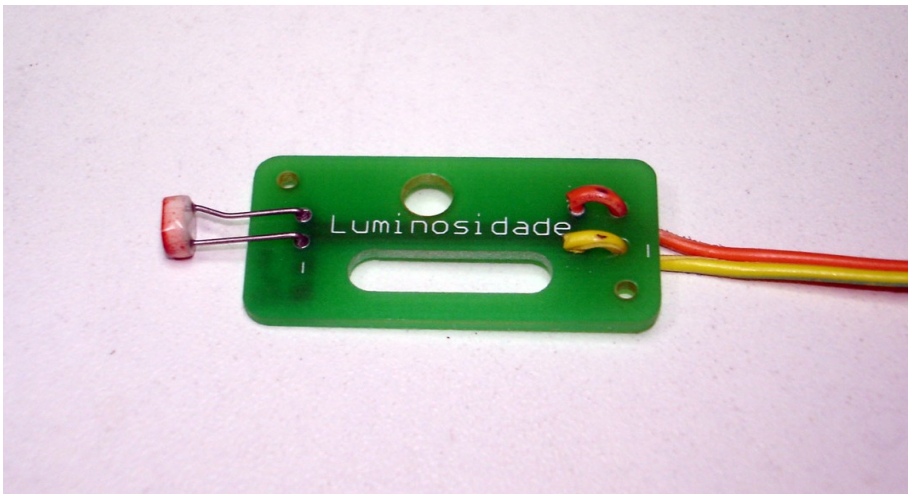


Figura 6. Sensor de luminosidade.

Esse sensor detecta o nível de luminosidade em um ambiente. Esse tipo de sensor é utilizado, por exemplo, nos postes de rua para acender ou apagar as luzes da iluminação pública. Dessa forma não é necessária a intervenção de ninguém para que quando estiver escuro as luzes sejam ligadas e assim que a iluminação natural do sol seja suficiente elas sejam apagadas.

Quanto menor for a iluminação sobre esse sensor menor será o seu sinal de resposta. Um valor de leitura do conversor AD igual a 0 indica escuridão total e conforme o grau de luminosidade aumenta o valor da leitura aumentará até um ponto em que a luminosidade é muito grande e satura a leitura do sensor, que irá responder um valor 1023.

Potenciômetros

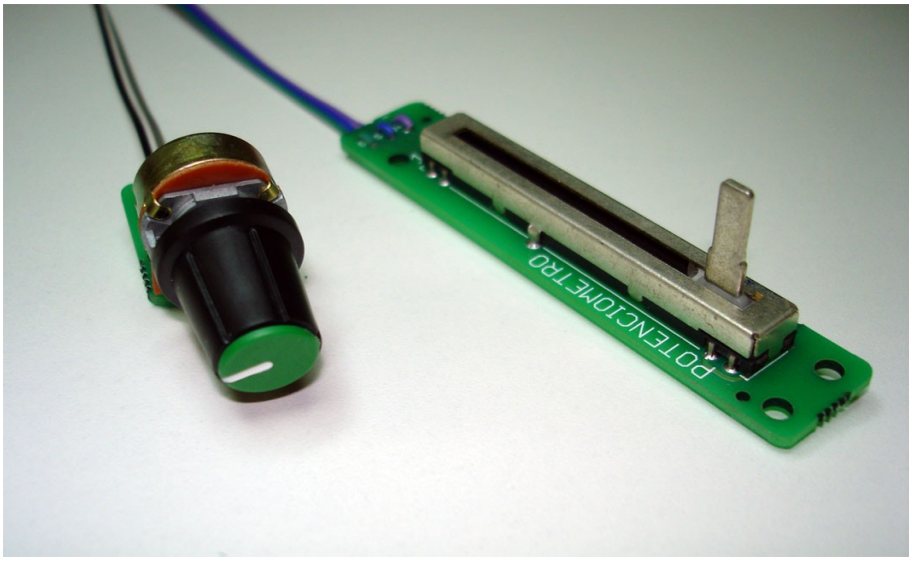


Figura 7. Potenciômetros.

Potenciômetros são normalmente utilizados para definir algum parâmetro como, por exemplo, o volume do audio de uma televisão ou rádio. Além desse tipo de função os potenciômetros podem também ser utilizados como sensores de posição. Nesse caso eles poderiam ser utilizados para identificar a posição de alguma peça móvel. O Kit Didático de Robótica contém dois potenciômetros, sendo um deles angular e o outro linear.

Os potenciômetros angulares podem ser utilizados, por exemplo, para identificar o ângulo de inclinação de uma rampa se um desses for colocado na base da rampa. Outro exemplo seria utilizá-los para registrar a posição angular de um braço mecânico, sendo necessário apenas posicioná-los nos "cotovelos" do braço.

Os potenciômetros lineares são muito utilizados em equalizadores de som. O funcionamento deles é muito similar ao dos potenciômetros angulares, no entanto, os lineares poden ser utilizados de forma mais fácil para identificação de posição em movimentos lineares enquanto que os potenciômetros angulares são mais apropriados para identificação de posições angulares.

A resposta desse sensor representa à posição do knob do potenciômetro. Será lido um valor 0 quando esse estiver no início do curso e 1023 quando ele estiver no final. Os valores entre 0 e 1023 representam as posições intermediárias.

Microfone

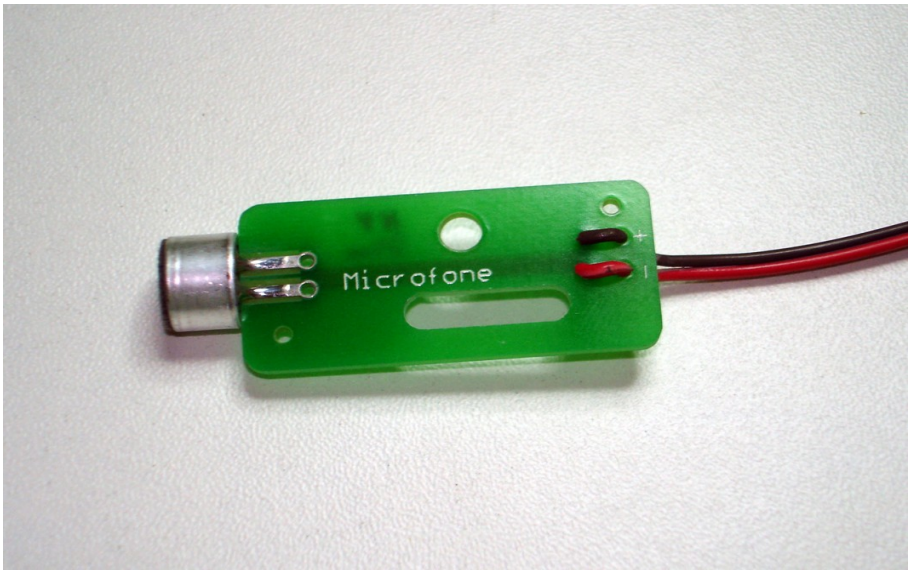


Figura 8. Microfone.

Microfones são sensores de sons. Com eles é possível identificar, por exemplo, o nível de ruído em um ambiente. Apesar da maioria dos computadores já possuírem um microfone, foi adicionado um no Kit Didático de Robótica principalmente para mostrar que podem ser utilizados como um sensor.

A resposta desse componente será correspondente à vibração gerada pelo som que ele está captando. Para uma análise dessa vibração são necessárias várias leituras em um certo período de tempo. O intervalo entre essas leituras não podem ser muito grande porque a variação do sinal

sonoro é muito rápida, e se as aquisições forem feitas com intervalos muito grandes, essa variação será perdida impossibilitando uma análise. Serão dados exemplos de como deve ser feita a leitura desse sensor ainda nesse documento.

Sensor de Vibração

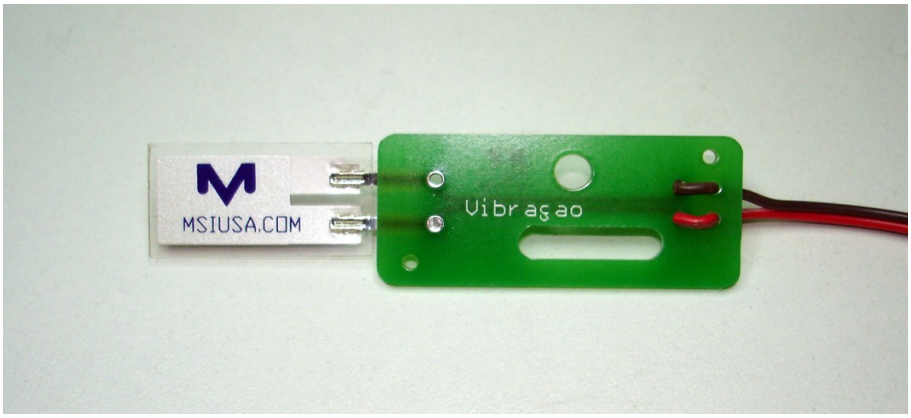


Figura 9. Sensor de vibração.

A vibração em estruturas é muito comum e o monitoramento dessas vibrações pode ser muito útil. Por exemplo, em ambientes industriais, onde o uso de motores é muito comum, através da análise da vibração deles é possível detectar algum defeito em um estágio inicial, quando é mais fácil e barato fazer o reparo. Isso pode evitar perdas financeiras significativas.

A resposta desse sensor é muito parecida com a resposta do microfone. No fundo os dois registram vibrações, mas no caso do microfone a vibração é provocada por ondas sonoras, enquanto que o sensor de vibração registra vibração de estruturas.

Por ter uma resposta parecida com a do microfone, a leitura do sinal também deve ser semelhante, isto é, várias leituras em um certo período de tempo para possibilitar uma análise e o intervalo entre essas leituras não pode ser muito grande.

Entradas Auxiliares

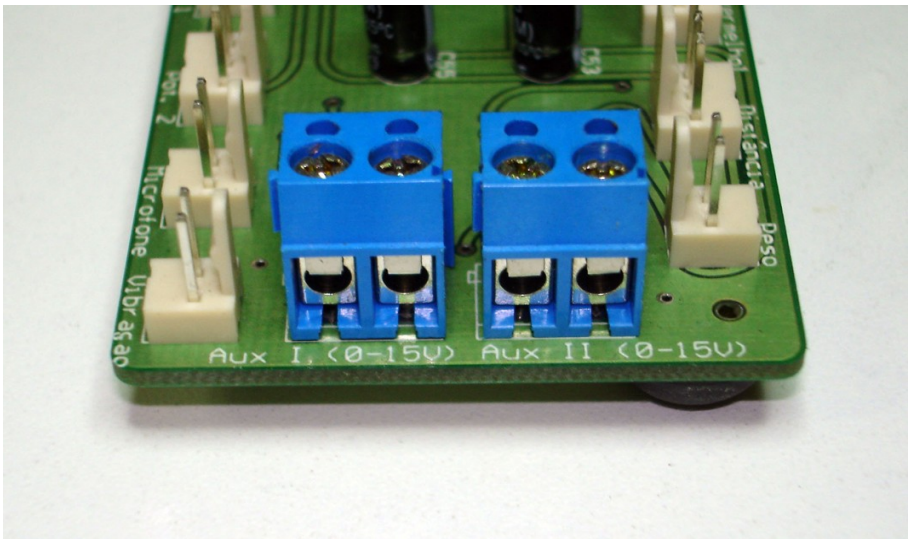


Figura 10. Entradas auxiliares.

As duas entradas auxiliares são entradas genéricas para tensões de 0 à 15 Volts. Essas entradas podem ser utilizadas para a ligação de algum sensor extra ou para monitorar alguma tensão como, por exemplo, de uma bateria ou pilha que possua uma tensão menor do que 15 Volts. A seguir, o esquema para ligação nessas entradas.

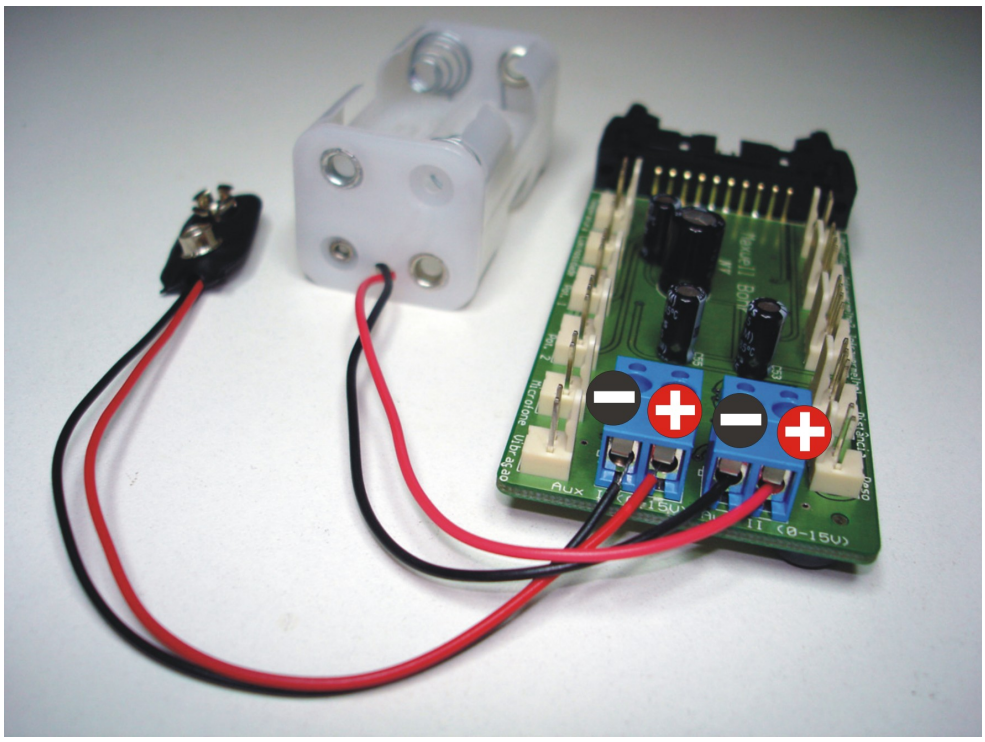


Figura 11. Esquema de ligação das entradas auxiliares.

A resposta lida pelo conversor AD será correspondente a tensão aplicada na entrada auxiliar, sendo que 0 irá corresponder a 0 Volts e 1023 à 15 Volts. Os valores intermediários representam os níveis de tensão entre 0 e 15 Volts, sendo que cada nível representa uma variação de 15/1024 Volts, ou seja, 14,6 milivolts aproximadamente. A fórmula que converte um valor lido pelo conversor AD no valor em Volts aplicado na porta auxiliar é:

$$V = 0{,}0146*X$$

Onde:

V = Tensão em Volts.

X = Valor de tensão convertido pelo conversor AD (0-1023).

Substituindo o valor de X na fórmula pelo valor lido pelo conversor AD obteremos o valor da tensão na entrada auxiliar em Volts. Podemos fazer a operação contrária também e converter um valor de tensão na entrada auxiliar para o valor que seria lido pelo conversor AD. É importante resaltar que esse valor é aproximado devido a inúmeros fatores que influênciam a leitura.

Sensor de Peso

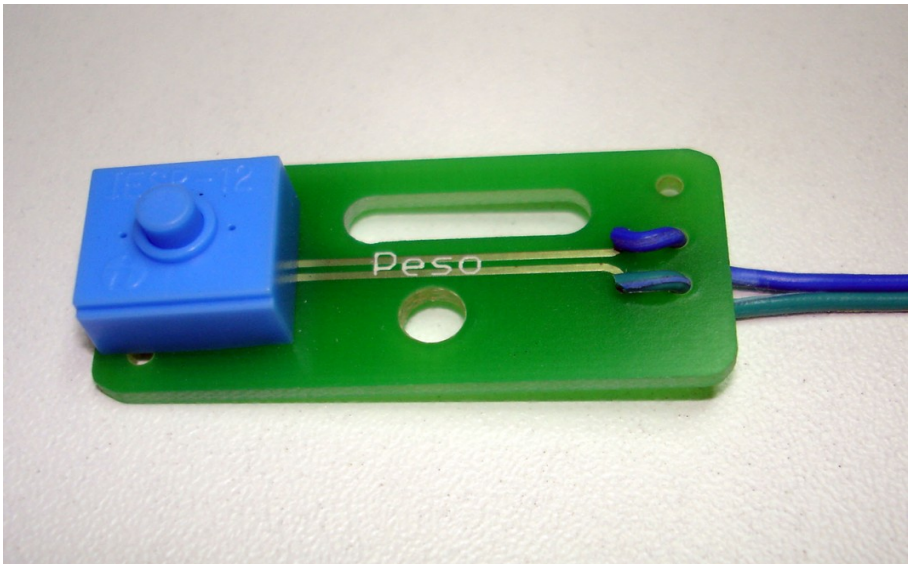


Figura 12. Sensor de Peso.

Peso é um outro parâmetro muito utilizado no dia a dia das pessoas. Vemos balanças para medição de peso em vários estabelecimentos comerciais tais como mercados, farmácias, restaurantes e até mesmo em casa. No ambiente indústrial a medição do peso de produtos é muito importante, desde o início do processo produtivo até o seu término.

O sensor de peso que acompanha o Kit Didático de Robótica pode medir pesos no intervalo de 200g à 4Kg. No entanto é recomendável que o peso máximo utilizado seja de 1,5 Kg para que o sensor tenha uma vida útil maior e, além do mais, opere na sua faixa de maior sensibilidade, que ocorre principalmente entre 200 e 600 gramas.

A resposta desse sensor não é linear e apresenta um comportamento logarítimico. Esse comportamento faz com que em um gráfico de resposta por peso o gráfico cresça rapidamente a partir de 250 gramas e se estabilize com um crescimento pequeno por volta de 500, 600 gramas. A seguir um gráfico aproximado da relação resposta por peso do sensor de peso IESP-12.

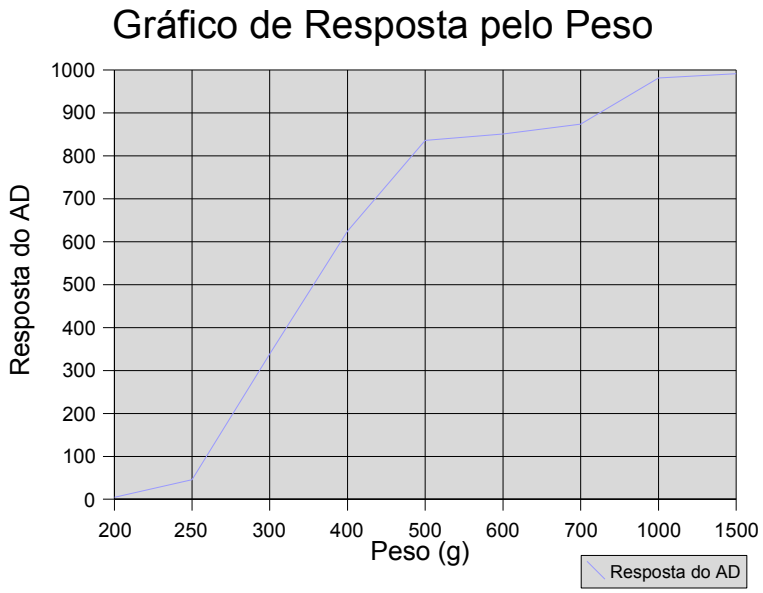


Figura 13: Gráfico de resposta pelo peso.

Sensor de Distância

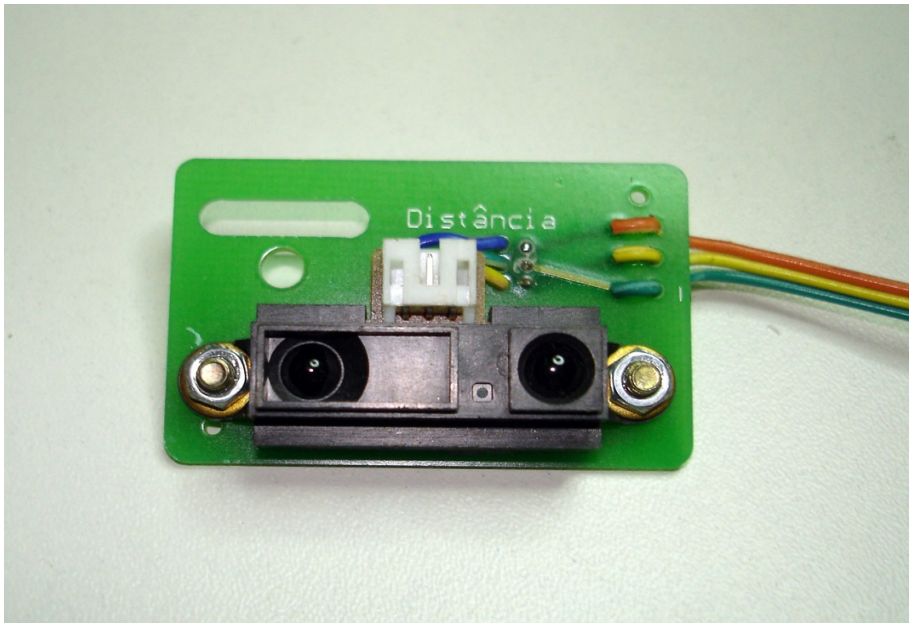


Figura 14. Sensor de Distância.

A identificação da distância de um objeto pode ser muito útil. Esse tipo de recurso é utilizado para detecção de obstáculos em robôs móveis, identificação de presença, abertura automática de portas, auxílio ao motorista para estacionar carros, entre outras aplicações.

O sensor utilizado no Kit Didático de Robótica identifica a distância de objetos através da emissão e reflexão de luz infravermelha. Ele possui um emissor desse tipo de luz, que é invisível para o ser humano, e um receptor. Através do cálculo do tempo entre a emissão e a recepção da luz após essa ser refletida por algum objeto, é possível saber a distância de um objeto. Com esse sensor é possível identificar objetos à uma distância de 5 cm à 80 cm. O gráfico abaixo é uma aproximação da resposta lida no conversor AD em função distância de um objeto.

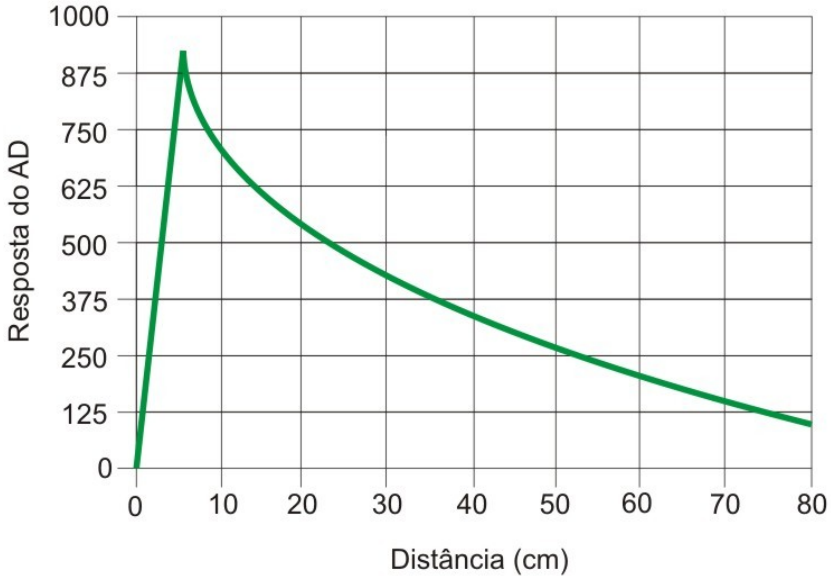


Figura 15. Resposta do Sensor de Distância.

Pares Ópticos

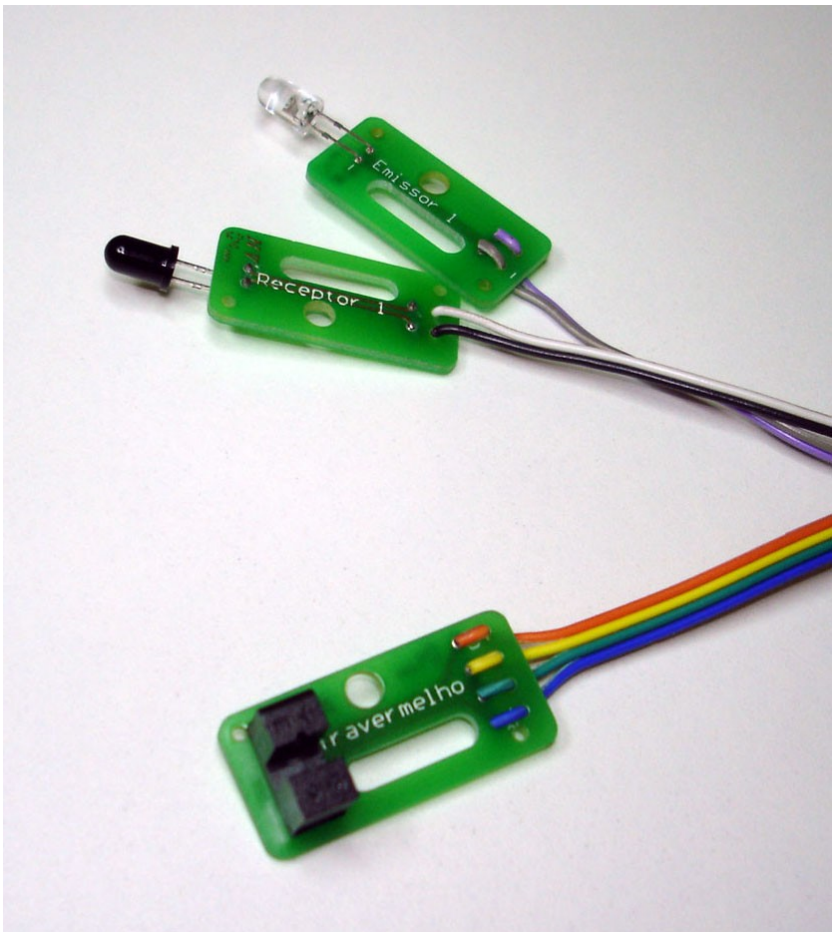


Figura 16. Pares Ópticos

Pares ópticos são um conjunto de um emissor e um receptor de luz infravermelho. Em aplicações práticas eles são posicionados de forma que um fique apontado para o outro. Dessa forma, se algum objeto interromper o feixe de luz infravermelho que existe entre o emissor e o receptor será possivel a detecção. Esse tipo de sensor é utilizado para identificar final de curso em impressoras, identificar presença de discos em drives de disquetes, contagem de rotações, entre outras aplicações.

Existem dois modelos acompanhando o Kit Didático de Robótica. Um deles contém o emissor e o receptor de infravermelho em placas separadas, de modo a serem utilizados na distância adequada à sua aplicação. O outro modelo já contém o emissor e o receptor na mesma placa, separados por uma pequena distância, para serem utilizados como uma chave óptica. Esse modelo é semelhante ao utilizado em drives de disquete, para reconhecer se há algum disquete presente.

Esse é um sensor com resposta digital e seu sinal não precisa ser lido pelo AD. Se o emissor estiver apontado para o receptor e não houver nenhum obstáculo entre eles entao a resposta será 1. Se houver algum obstáculo impedindo que a luz infravermelho chegue até o receptor então a resposta lida será 0.

Chave Magnética

Figura 17. Chave Magnética.

Uma chave magnética é uma chave elétrica que é acionada por um campo magnético. É muito utilizada em sistemas de alarmes para identificar a abertura de portas e janelas. Normalmente ficam em caixinhas cinza, uma com um imã e outra com a chave magnética. São fixadas de forma que quando a janela ou a porta estão fechadas o imã fique perto da chave e assim a deixe acionada. Se essa porta ou janela for aberta o imã irá se distanciar da chave magnética e fará com que ela seja desligada acionando o alarme.

Podemos ver, através do vidro de proteção desse componente, que esse dispositivo nada mais é do que duas chapas de metal que ficam próximas e com a aproximação de um campo magnético elas se atraem, fechando o contato elétrico. Assim como os pares ópticos, esse é um sensor com resposta digital. Se o contato da chave estiver fechado a leitura será 1, caso contrário será 0.

Tensões de Alimentação

Além de monitorar os sensores, também é possível monitorar a tensão de alimentação que chega até a o *Módulo de Sensores*. Podemos monitorar duas tensões distintas, uma de 12 Volts não estabilizada e outra de 5 Volts estabilizada. Esse tipo de monitoramento é normalmente encontrada em equipamentos eletrônicos, como por exemplo os computadores. A finalidade dele é, em geral, prevenção e identificação de falhas na alimentação do equipamento.

Módulo de Sensores Genérico

Como vimos anteriormente, o *Módulo de Sensores* possui uma lista pré-estabelecida de sensores. Essa lista tem seu foco principal na variedade dos sensores. Essa característica é muito interessante para que se conheça vários tipos de sensores e possibilita a criação de inúmeros projetos com facilidade. No entanto, isso também pode limitar o seu uso.

Se algum projeto necessitar de algum sensor que não está presente na sua lista pré-estabelecida ou se forem necessários vários sensores de um mesmo tipo torna-se difícil o uso do *Módulo de Sensores*. Poderíamos adaptar sensores para que fossem ligados em alguma das portas auxiliares. No entanto temos apenas duas portas auxiliares e dependendo das necessidades isso não seria suficiente para solucionar o problema.

Para contornar essa situação, foi desenvolvido o *Módulo de Sensores Genérico*. Seu objetivo é permitir uma maior liberdade no uso de sensores de modo que seja possível customizar a seleção de sensores de acordo com o projeto que se deseja trabalhar. Podemos selecionar os tipos de sensores e a quantidade de cada um de maneira que possam ser aproveitados todos os recursos que o Kit Didático de Robótica oferece.

Esse módulo possui uma placa com 4 conectores idênticos. Em cada um desses conectores é possível ligar 3 sensores analógicos e 1 digital. Assim podemos conectar um total de 12 sensores analógicos e 4 digitais. Essas entradas são genéricas, isto é, podemos ligar sensores de qualquer tipo, desde de que o sinal desses seja convertido para as especificações dessas entradas genéricas. Além dos conectores para sensores, esse módulo possui uma saída de alimentação para circuitos externos. A seguir, uma figura do *Módulo de Sensores Genérico*.

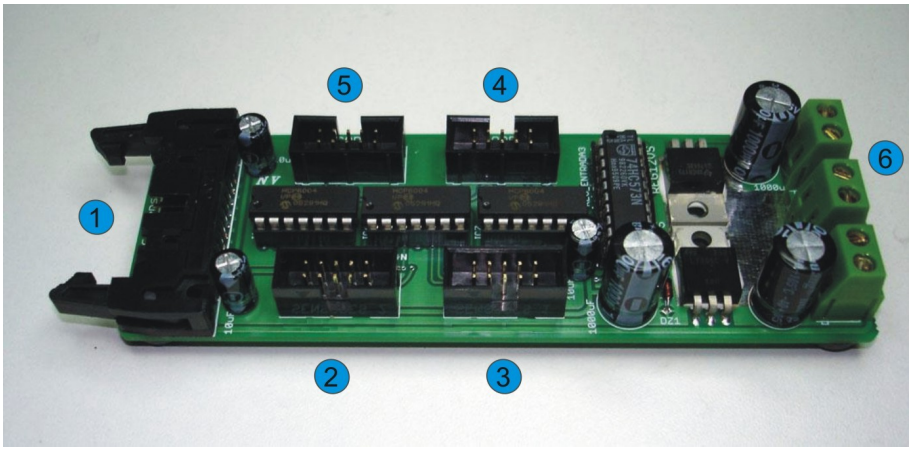


Figura 18. Módulo de Sensores Genérico. (1) Conector flat de 20 vias para conexão com o Módulo Principal, (2) Entrada Genérica 0, (3) Entrada Genérica 1, (4) Entrada Genérica 2, (5) Entrada Genérica 3, (6) Saída de alimentação.

Conectores para Sensores Genéricos

Cada um dos 4 conectores para sensores do *Módulo de Sensores Genérico* possui entradas para 3 sensores analógicos e 1 digital. A pinagem desses conectores é identica. A seguir a configuração dos pinos desses conectores:

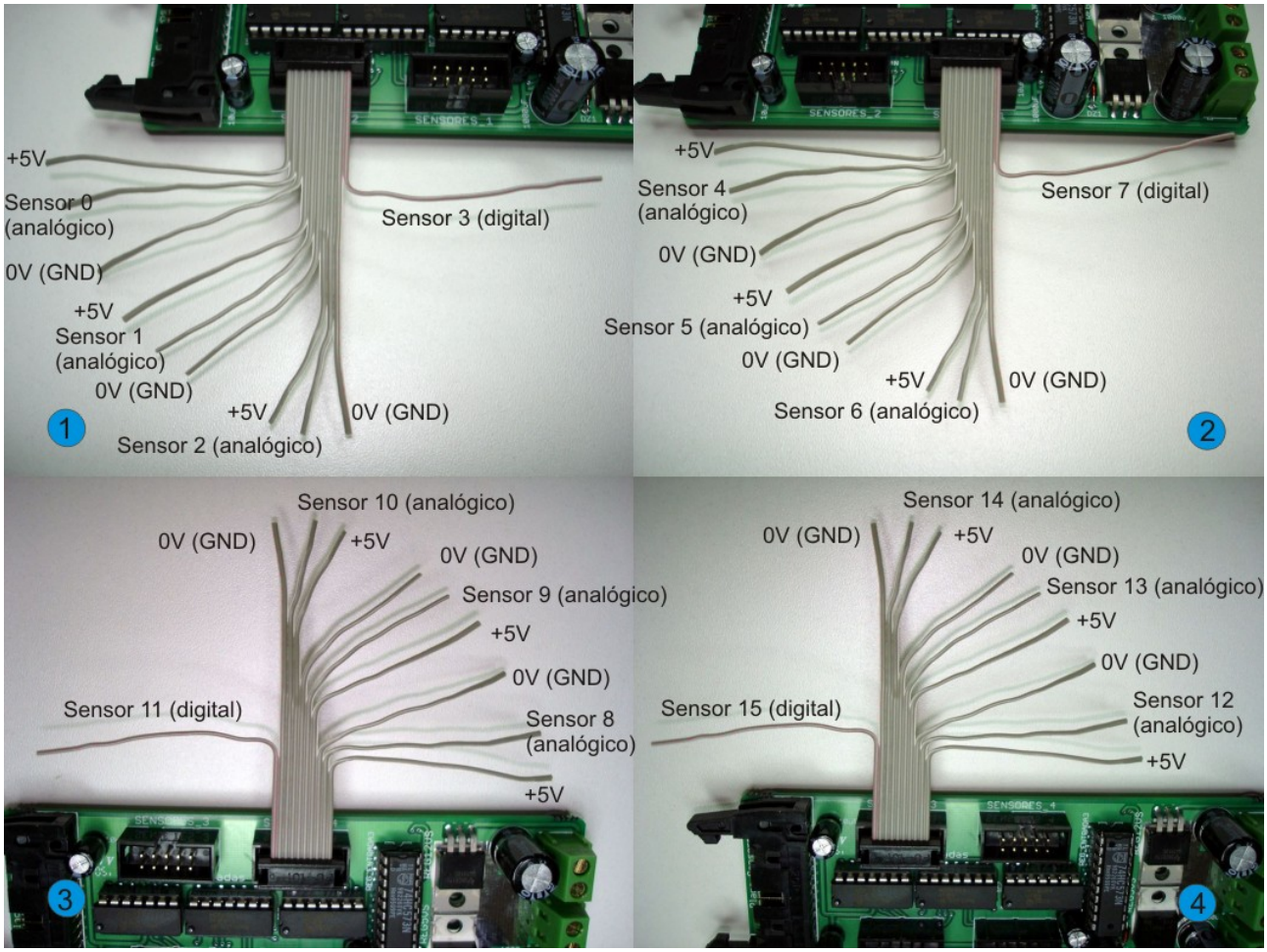


Figura 19. *Esquema de conexão dos conectores de sensores genéricos. (1) Entrada genérica 0. (2) Entrada genérica 1. (3) Entrada genérica 2. (4) Entrada genérica 3.*

A figura acima mostra detalhadamente a disposição dos conectores de sensores genéricos e a nomenclatura de suas entradas. Podemos perceber que cada conector possui 3 entradas para sensores analógicos e uma entrada para o sinal de um sensor digital, além das alimentações destinadas a esses sensores.

Os pinos para sinais analógicos devem receber uma tensão entre 0 e 5 Volts. Se o sensor que queremos utilizar possuir algum outro tipo de sinal, esse deve ser adequado eletronicamente de forma que seja convertido em uma tensão de 0 à 5 Volts. Já o sinal de um sensor digital deve ser 5 Volts para representar o nível lógico “1” e 0 Volts para representar o nível lógico “0”.

Como podemos perceber, para utilizar as entradas genéricas, é necessário o conhecimento das especificações técnicas do sensor e um conhecimento básico de eletrônica para que se possa fazer a adequação de seu sinal corretamente quando necessário. O uso de sinais fora das especificações pode danificar o equipamento.

Para servir de exemplo de ligação de um sensor analógico vamos mostrar como ligar um potenciômetro. Abaixo o esquema de ligação de um potenciômetro.

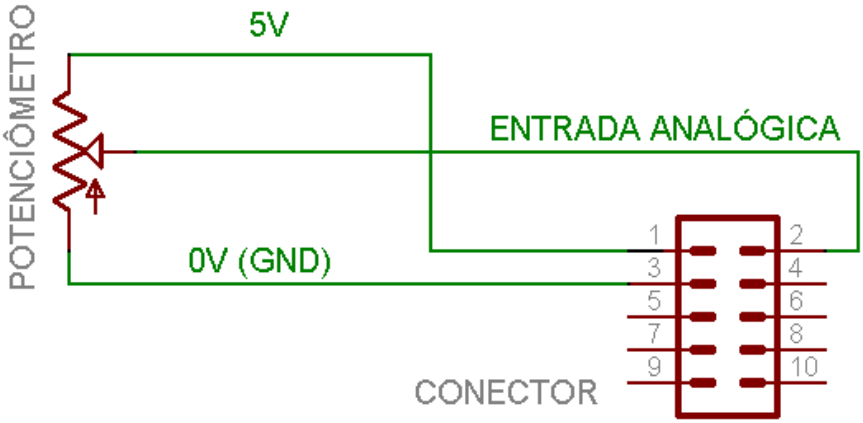


Figura 20. Ligação de um potenciômetro em um conector do Modulo de Sensores Genérico.

Ligamos os pinos de 0 e 5 Volts nos terminais nas extremidades do potenciômetro. Com isso temos no terminal central do potenciômetro um sinal que irá variar de 0 à 5 Volts dependendo apenas da posição do knob. Dessa forma podemos ligar o terminal central do potenciômetro diretamente no pino de sinal analógico. Se quisessemos ligar 3 potenciômetros em um mesmo conector seguiriamos o seguinte esquema:

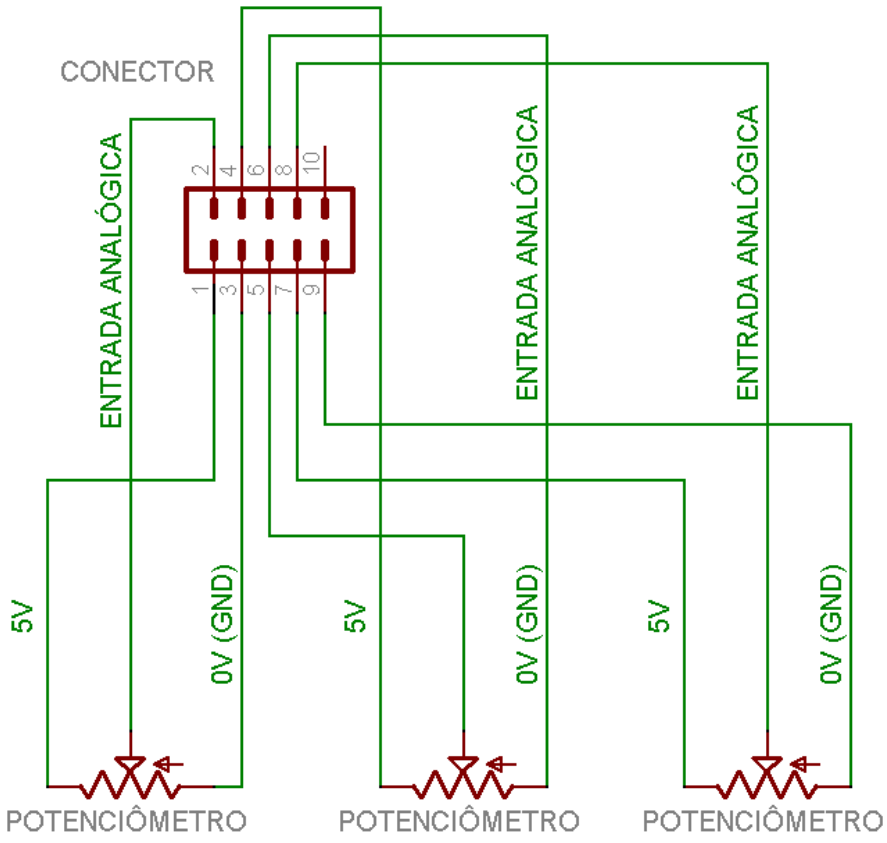


Figura 21. Ligação de três potenciômetros em apenas um conector do Modulo de Sensores Genérico.

Para demonstrar como ligar um sensor digital, vamos utilizar uma chave como exemplo. O esquema de ligação seria:

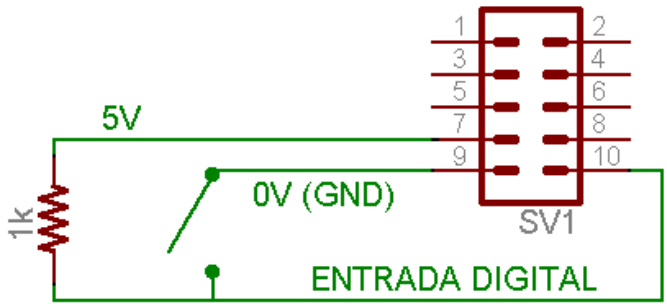


Figura 22. Ligação de uma chave.

Com essa ligação, quando a chave estiver aberta, temos 5 Volts no pino de sinal, logo um nível lógico "1", se a chave estiver fechada temos 0 Volts no pino de sinal, o que significa um nível lógico "0".

Saída de Alimentação

Para auxiliar o desenvolvimento de circuitos externos, foi adicionada no *Módulo de Sensores Genérico* uma saída para alimentação de circuitos externos simples. Essa saída pode ser utilizada para alimentar circuitos montados em uma matriz de contatos ou mesmo em placas, desde que não consumam muita corrente. Ela possui três terminais, um com 0 Volts, outro com 5 Volts estabilizados e outro com 12 Volts não estabilizados. A seguir o esquema dos terminais.

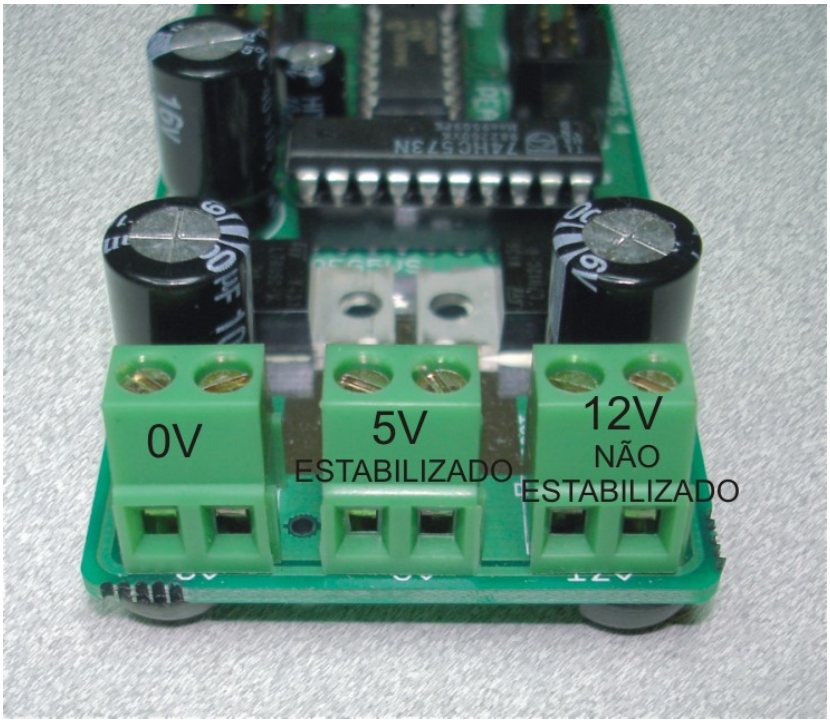


Figura 23. Esquema da saída de alimentação.

Módulo de Motores e Displays

Além da leitura de sensores, o Kit Didático de Robótica também pode controlar dispositivos, esse é o objetivo do *Módulo de Motores e Displays*. Como o seu próprio nome já sugere, os principais ítens de controle são motores e displays. Esse módulo possui uma placa com toda a eletrônica necessária para a ligação e controle desses dispositivos. Eles são considerados dispositivos de saída, uma vez que, apresentam informações ou modificam o ambiente em que se encontram. A única exceção é um teclado numérico, que assim como os sensores, é um dispositivo de entrada de informações, pois enviam informação externas para o Kit.

O *Módulo de Motores e Displays* possui uma placa eletrônica com três conectores para cabo flat. Um de 34 vias que é utilizado para conexão com o *Módulo Principal*, outro de 14 vias para ligação de um display de cristal líquido (LCD) e por fim um de 10 vias para conexão de um teclado. Existem também conectores para um buzzer, um motor de passo e dois motores DC.

Além desses conectores essa placa do *Módulo de Motores e Displays,* possui soldados diretamente nela, dois relés, que são chaves eletromagnéticas, dois displays de 7 segmentos e duas barras de LEDs. A seguir uma imagem do *Módulo de Motores e Displays* montado com todos os seus ítens.

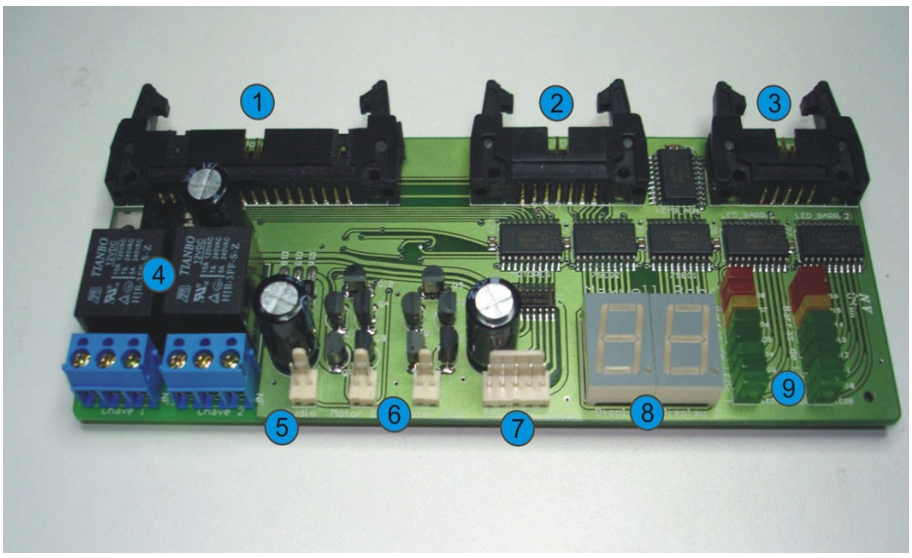


Figura 24. Módulo de Motores e Displays. (1) Conector flat de 34 vias para conexão com o Módulo Principal, (2) Conector de 14 vias para conexão do display de cristal líquido, (3) Conector de 10 vias para conexão do teclado, (4) Relés, (5) Conector para buzzer, (6) Conectores para motores DC, (7) Conector para motor de passo, (8) Displays de 7 segmentos, (9) Barras de LEDs.

Display de Cristal Líquido

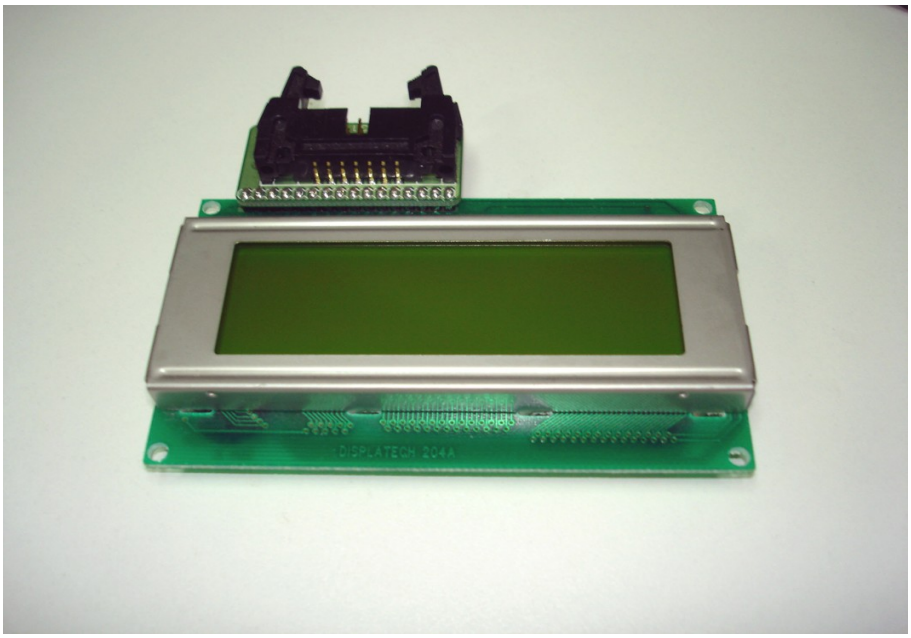


Figura 25. Display de cristal líquido.

Os displays de cristal líquido ou simplesmente LCD, são muito utilizados em equipamentos eletrônicos para criar a interface com o usuário. Seu baixo consumo de energia, comparado com o de outras tecnologias de display, faz com que sejam muito utilizados em aparelhos portáteis.

O displays de cristal líquido que acompanha o Kit Didático de Robótica permite a

apresentação de 4 linhas de texto, sendo que cada linha dessas pode apresentar 20 caracteres. Esse display também possui uma luz de fundo que permite uma melhor visualização dos caracteres, principalmente em ambientes sem muita iluminação.

Esse ítem também pode ser utilizado com o M*ódulo de Entradas, Saídas e Servo-Motores.*

Teclado

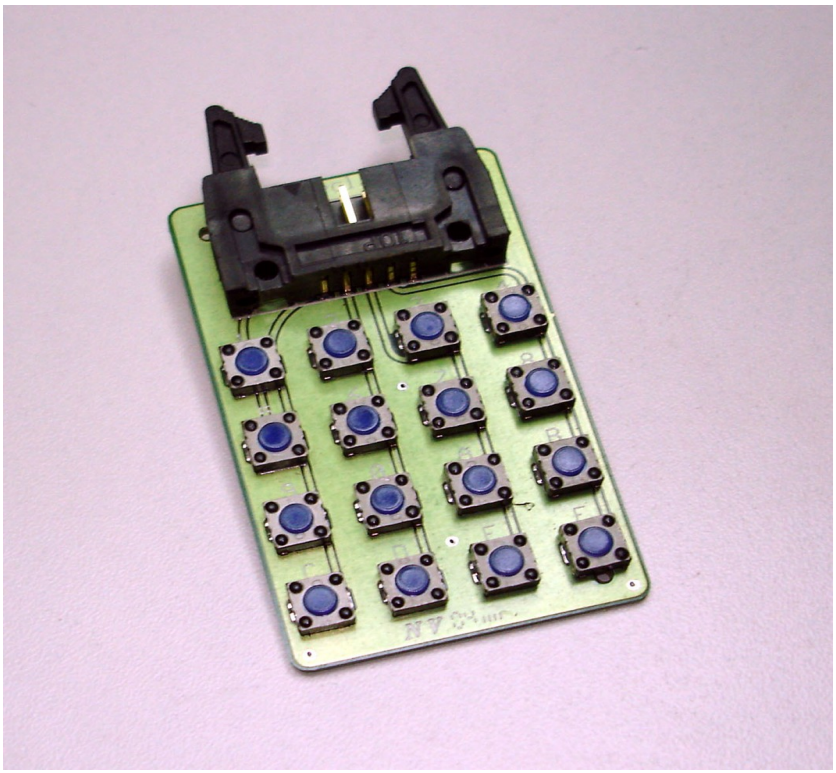


Figura 26. Teclado.

Teclados são um dos principais meios de entrada de informações nos computadores e em muitos outros equipamentos eletrônicos, tais como, caixas eletrônicos nos bancos, calculadoras, painéis de controle industrial, telefones, entre outros.

O teclado é o único ítem de entrada de informações que acompanha o *Módulo de Motores e Displays*. Ele possui 16 botões distribuidos em 4 linhas por 4 colunas. O significado de cada tecla irá depender da aplicação, podendo ser utilizado como um teclado numérico, um controle direcional, botões de controle, ou qualquer outra função que seja necessária pela aplicação.

Barras de LEDs

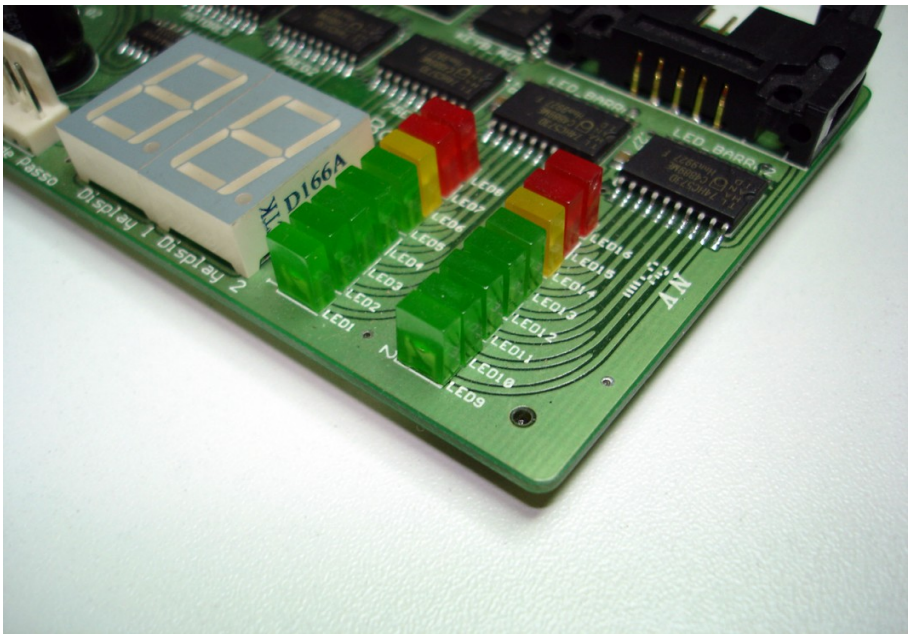


Figura 27. Barras de LEDs.

LEDs são frequentemente utilizados em interface de equipamentos. São, em geral utilizados como indicação de algum estado, como por exemplo se um equipamento está ligado ou se está realizando alguma operação. Uma outra forma muito comum é utilizá-los em barra para indicar alguma progressão ou nível, o exemplo mais comum desse tipo de interface pode ser vista em aparelhos de som, no entanto, também é utilizada em painéis industriais, controle de progressão de operações, entre outros.

O *Módulo de Motores e Displays* possui 2 barras com 8 LEDs. Em cada uma existem 5 LEDs verdes, 1 amarelo e 2 vermelhos. Essa diferenciação de cor foi feita para possibilitar a indicação de níveis com importâncias diferentes. Por exemplo, em um montitoramento de temperatura, podemos fazer uma lógica de controle que enquanto a temperatura estiver normal o nível dos LEDs fica apenas nos verdes, se a temperatura aumentar até um valor aceitável, mas acima do normal, então o nível de LEDs chegaria até o amarelo, e por fim se a temperatura atingir um nível perigoso os LEDs vermelhos se acenderiam.

Apesar dos LEDs estarem posicionados em forma de barra, eles podem ser utilizados para outros tipos de indicações, isso irá depender da aplicação e do programa de controle no computador.

Displays de 7 Segmentos

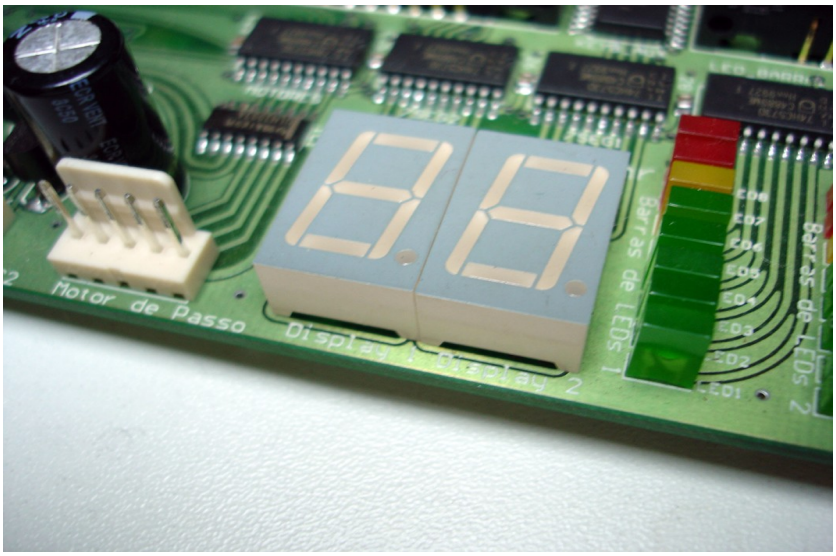


Figura 28. Displays de 7 segmentos.

Outra forma de apresentar informações é através do uso de displays de 7 segmentos. Esses displays recebem esse nome porque possuem 7 segmentos de LEDs dispostos de maneira que torna possível apresentar números de 0 à 9 e algumas letras.

O *Módulo de Motores e Displays* possui 2 displays de 7 segmentos. Esses displays possuem um oitavo segmento que serve para apresentar um ponto.

Motor de Passo

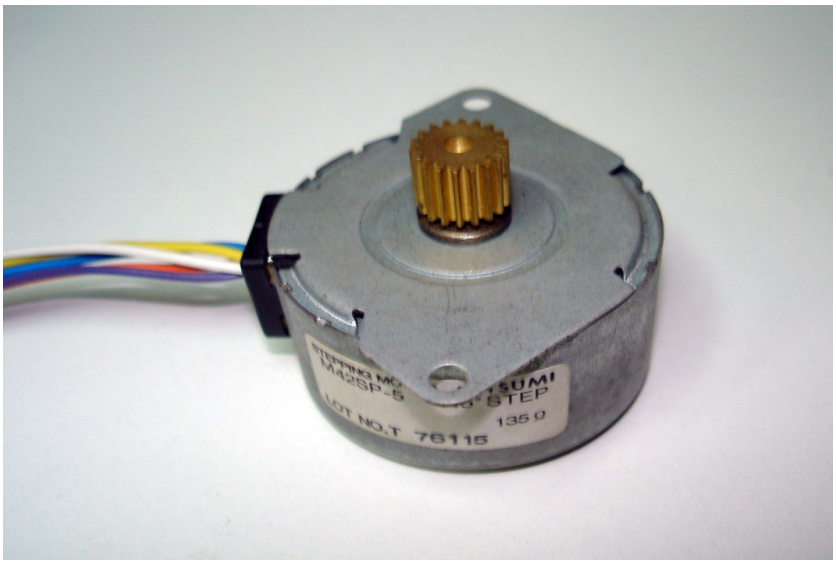


Figura 29. Motor de passo.

Motores de passo são utilizados principalmente quando a precisão nos movimentos é importante. Nesses motores uma volta de seu eixo é dividida em vários passos e podemos controlar o movimento do eixo de passo em passo. Dessa forma conseguimos um movimento muito preciso. Quanto mais passos por volta o motor possuir mais preciso será o seu movimento. Esse tipo de motor é muito utilizado em impressoras, drives de disquete, carros, sistemas industriais entre outras aplicações.

O motor de passo que acompanha o *Módulo de Motores e Displays* possui 48 passos por volta, dessa forma cada passo movimenta o eixo do motor em 7.5º. Podem ser controlados ainda o sentido da rotação e a velocidade. Comparados com os motores DC, esse motor não permite uma velocidade de rotação muito alta e nem uma força tão grande como os servo-motores, no entanto, em aplicações de precisão ele é uma ótima opção, pois, de forma simples, oferece uma boa precisão o que não ocorre com os outros motores que acompanha o Kit.

Motores DC

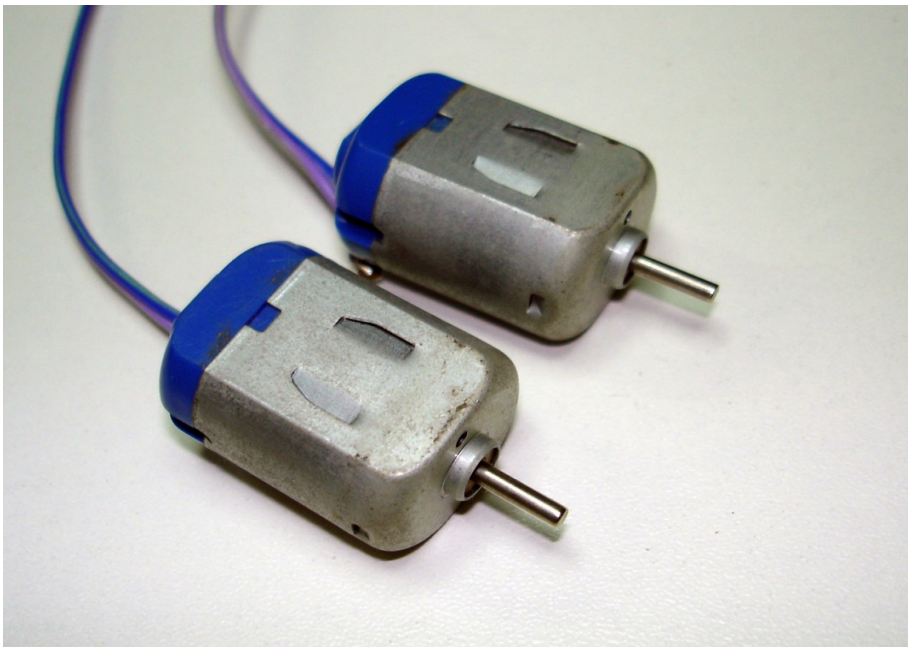


Figura 30. Motor DC.

Os motores DC são alimentados por uma fonte de energia de corrente contínua. Eles suportam uma alta rotação e tem características semelhantes a de motores AC, que são alimentados por uma fonte de energia de corrente alternada.

O modelo de motor DC que faz parte do *Módulo de Motores e Displays* suporta uma rotação máxima de aproximadamente 10.000 RPM e essa velocidade pode ser controlada. Esse motor é de pequeno porte e é muito utilizado em brinquedos.

Justamente por ser de pequeno porte esse motor não proporciona uma força muito grande. Para aplicações que necessitem de mais força seriam necessários motores de maior porte. Existe a possibilidade de alguma montagem mecânica com polias ou engrenagens para que se

tivesse uma aumento na força desses motores, que é justamente o que é feito nos servo-motores em que um motor é acoplado à uma caixa de redução, que é um conjunto de engrenagens que aumenta a força do motor.

O uso dos motores DC que acompanham o *Módulo de Motores e Displays* é mais recomendado para aplicações que necessitem de velocidade sem exigir muita força.

Buzzer

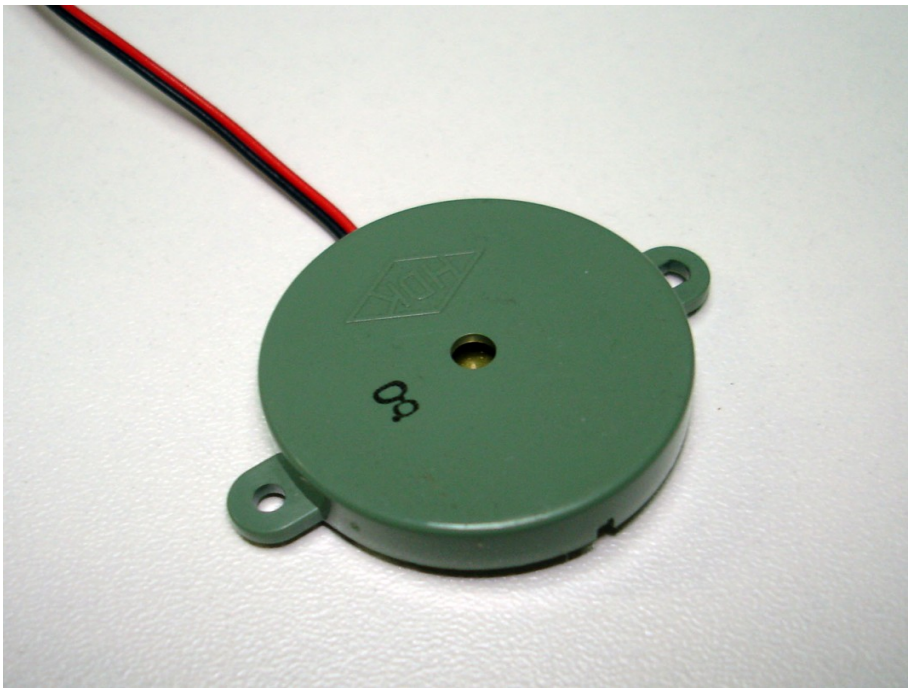


Figura 31. Buzzer.

Buzzer é um dispositivo que emite sons. É utilizado principalmente para emitir avisos sonoros simples, no entanto, é possível utiliza-los para emitir seqüências complexas de sons.

O buzzer que acompanha o *Módulo de Motores e Displays* pode emitir sons em uma faixa de freqüência de 20Hz à 20KHz, que é a faixa audível ao ser humano. A resposta em freqüência típica de um buzzer é irregular, ou seja, varia muito dependendo da freqüência do som. Na prática isso significa que em certas freqüências o som será mais alto do que em outras.

Relés

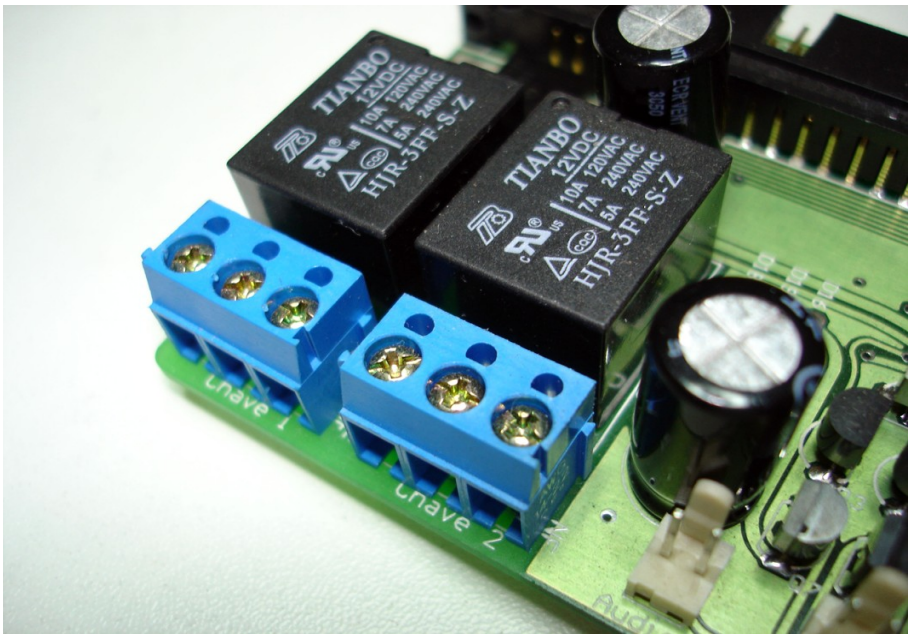


Figura 32. Relés.

Relés são chaves eletromecânicas acionadas através de um sinal elétrico. No *Módulo de Motores e Displays* encontramos dois desses componentes que podem ser utilizados como interruptores para acionar circuitos externos.

Apesar dos relés utilizados nesse módulo suportarem tensão e corrente elevadas, o Kit Didático de Robótica foi projetado para ser utilizado apenas com tensões baixas, em função dos riscos do uso de tensões mais elevadas. Por isso, esses relés devem ser utilizados apenas para acionar circuitos que utilizem uma tensão baixa, de no máximo 20 Volts. A seguir um esquema para ligação de circuitos externos aos relés.

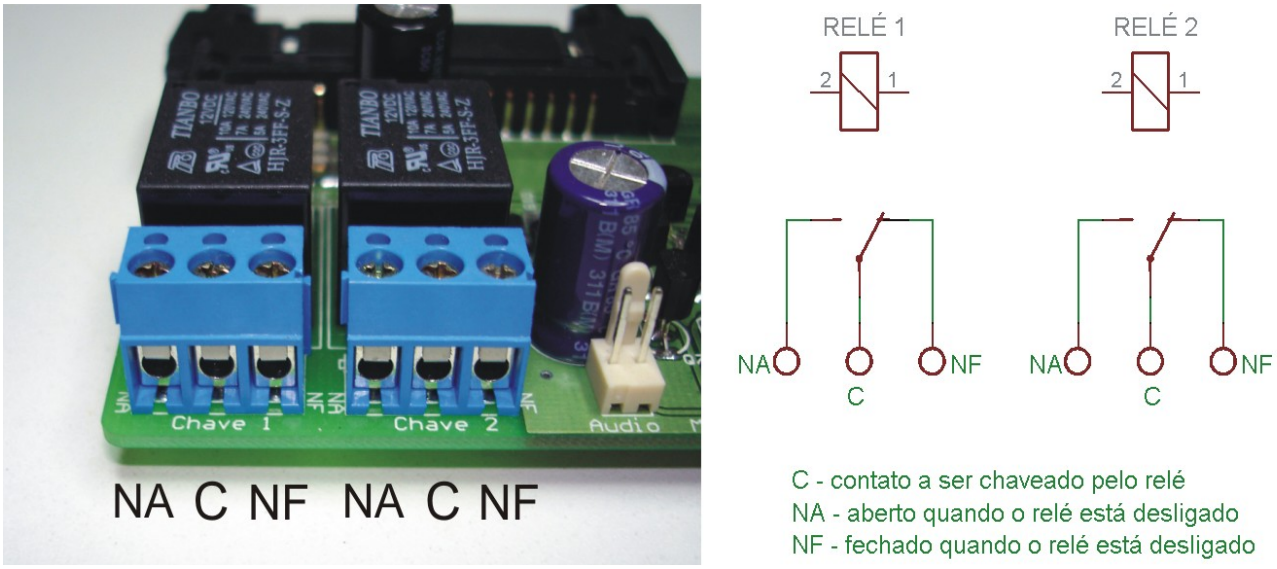


Figura 33. Esquema de ligação dos relés.

Módulo de Entradas, Saídas e Servo-Motores

O M*ódulo de Entradas, Saídas e Servo-Motores* foi desenvolvido com dois objetivos principais. Primeiro, controlar servo-motores, muito importantes em aplicações que dependem de alguma movimentação mecânica, uma vez que esse tipo de motor possui um bom torque. O segundo objetivo foi flexibilizar os tipos de dispositivos de entrada e saída que o Kit Didático de Robótica pode controlar. Isso é possível através de portas digitais de entrada e saída genéricas. Podemos dizer que esse módulo seria para o *Módulo de Motores e Displays* o que o *Módulo de Sensores Genérico* é para o *Módulo de Sensores.*

Esse módulo possui um conector de 34 vias para conexão com o *Módulo Principal* e um de 14 para conexão de um display LCD, o mesmo que também é utilizado no *Módulo de Motores e Displays*. Além desses conectores de cabo flat, o módulo possui outros seis, sendo dois para portas de entradas digitais, dois para portas de saídas digitais e dois para controle de servo-motores. Assim como no *Módulo de Sensores Genérico,* existe uma saída para alimentação de circuitos externos.

É importante citar que como todos os módulos do Kit são alimentados pela fonte do *Módulo Principal*, uma fonte que fornece 800 miliamperes, é preciso que as extensões dos módulos não consumam muita corrente. Se isso for realmente necessário será preciso utilizar uma fonte externa. Essa limitação aplica-se tanto a saida de alimentação quanto à portas, principalmente a de ligação de servo-motores que podem consumir muita corrente dependendo da montagem.

A seguir uma foto do M*ódulo de Entradas, Saídas e Servo-Motores*.

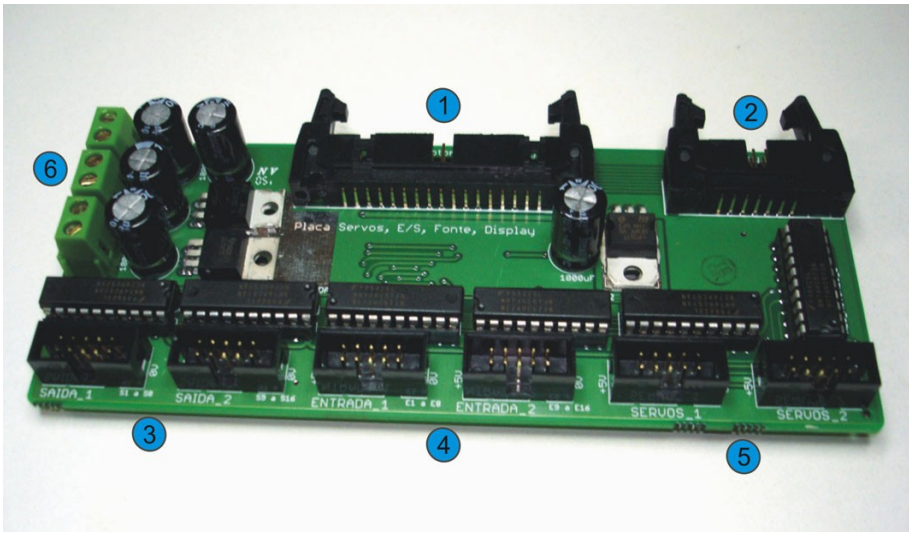


Figura 34. Módulo de Entradas, Saídas e Servo-Motores. (1) Conector flat de 34 vias para conexão com o Módulo Principal, (2) Conector de 14 vias para conexão do display de cristal líquido, (3) Saídas digitais, (4) Entradas digitais, (5) Conectores para servo motores, (6) Saída de alimentação.

Display de Cristal Líquido

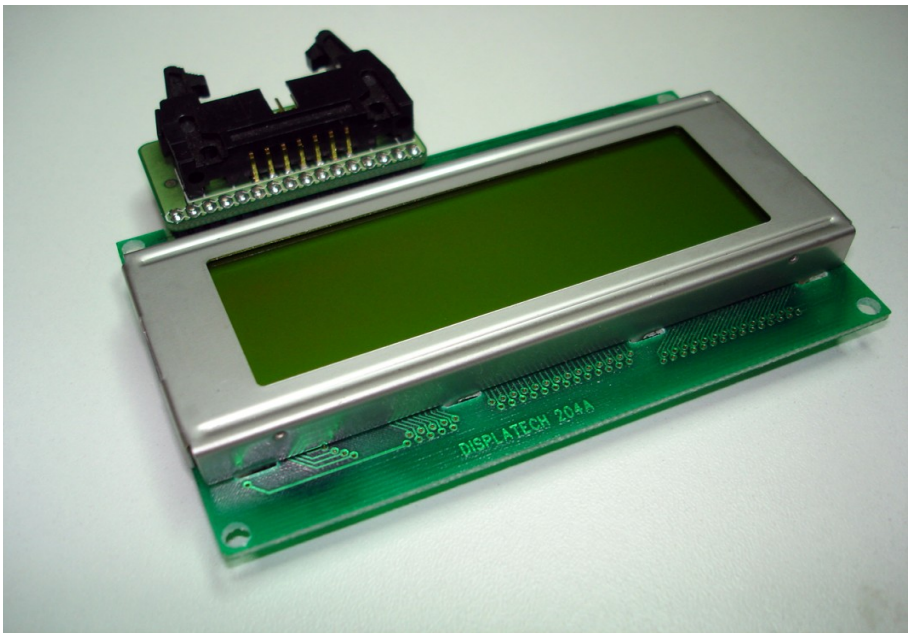


Figura 35. Display de cristal líquido.

O mesmo display de cristal líquido que é utilizado no *Módulo de Motores e Displays* podemos conectar nesse módulo. Para maiores informações veja a descrição desse ítem no *Módulo de Motores e Displays*.

Entradas Digitais

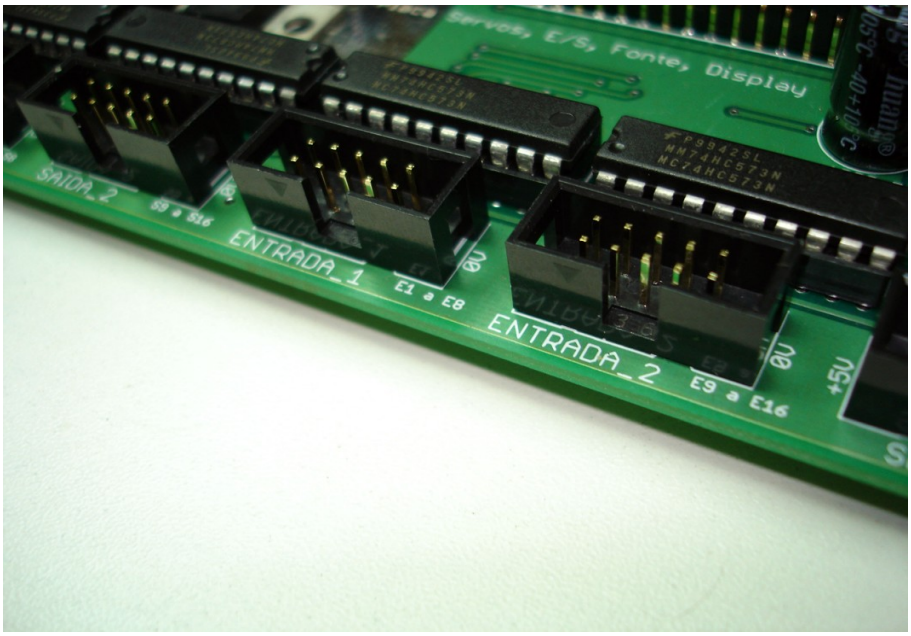


Figura 36. Portas de entradas digitais.

Para possibilitar a ligação de dispositivos de entrada de dados que não fazem parte do Kit Didático de Robótica foram disponibilizadas duas porta genéricas de entradas digitais nesse módulo, sendo que, cada uma possui oito bits, isto é, um byte. Essas portas podem ser utilizadas também para a leitura de sensores digitais, da mesma forma que é feito no *Módulo de Sensores Genéricos.* A pinagem dos conectores dessas portas são identicas e é dada a seguir:

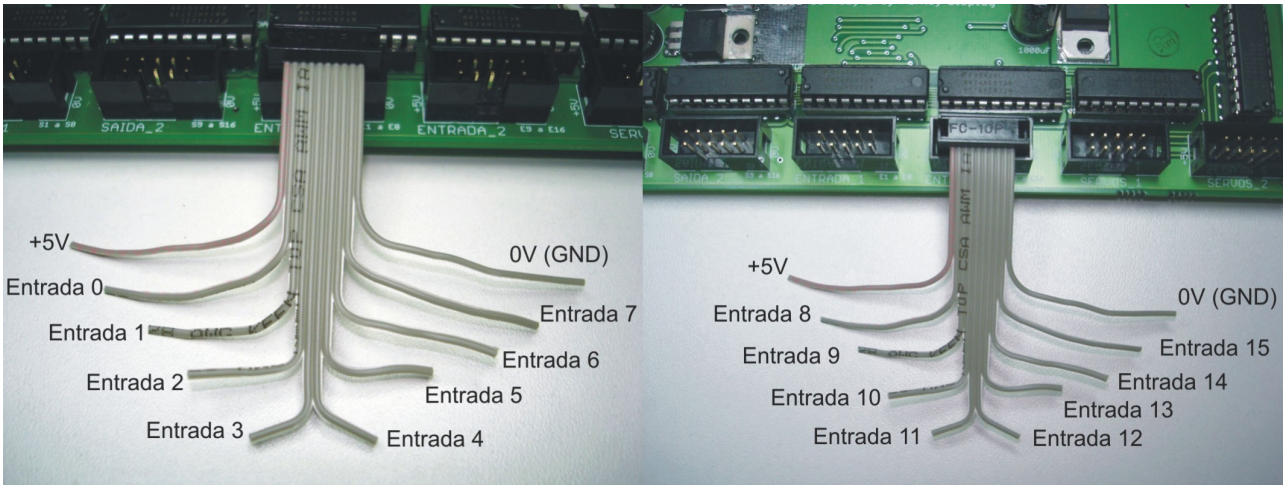


Figura 37. Esquema dos conectores das portas de entradas digitais.

Como podemos ver, cada conector possui pinos para 5 Volts, 0 Volts e oito pinos para entradas de sinal digitais. Para indicar um nível lógico “1” em algum dos pinos de entrada devemos aplicar 5 Volts, e para indicar um nível lógico “0” devemos aplicar 0 Volts. O uso de tensões fora da faixa de 0 à 5 Volts pode danificar o equipamento.

Para utilizar essas entradas é necessário um conhecimento básico de eletrônica. Em grande parte das aplicações a interface com essas portas será simples, principalmente se elas forem utilizadas para leitura da resposta de sensores digitais. Para aplicações mais complexas é necessário um conhecimento de eletrônica digital. A seguir um exemplo de ligação de oito chaves que poderiam ser utilizadas como sensores digitais.

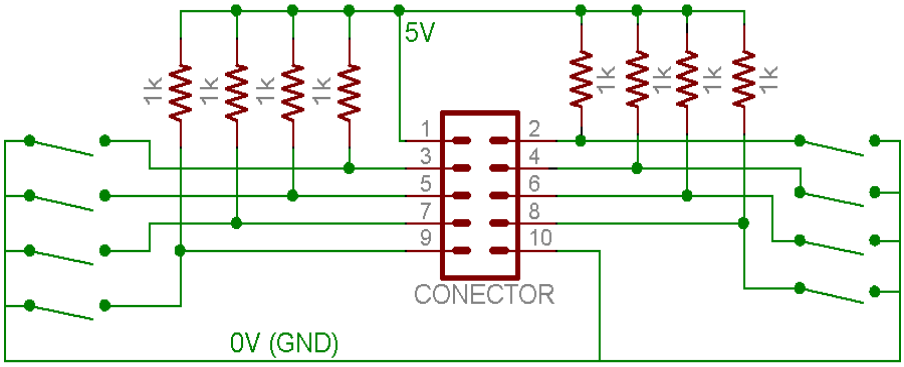


Figura 38. Esquema de ligação de oito chaves em uma porta de entrada digital

Saídas Digitais

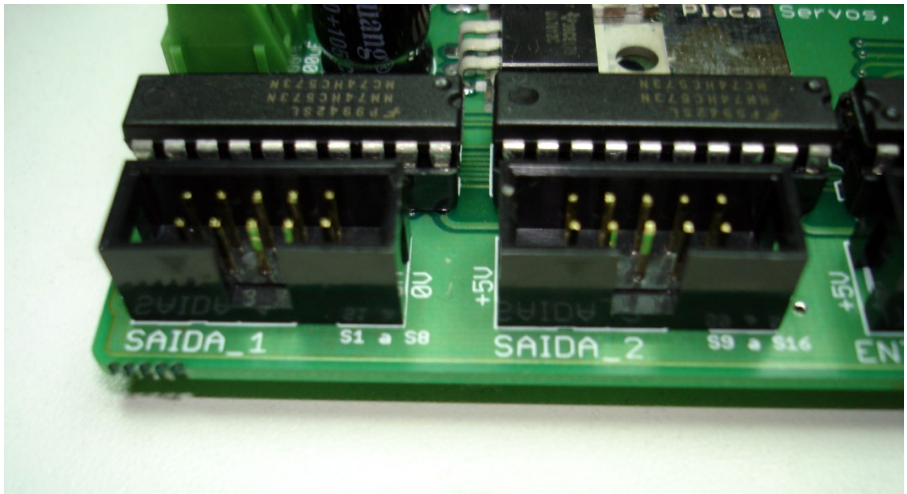


Figura 39. Portas de saídas digitais.

Além de portas para entradas digitais, o M*ódulo de Entradas, Saídas e Servo-Motores* possui duas portas de saídas digitais. Assim como as portas de entrada, as de saída possuem oito bits cada uma. A pinagem das duas portas de saidas são identicas e é dada a seguir:

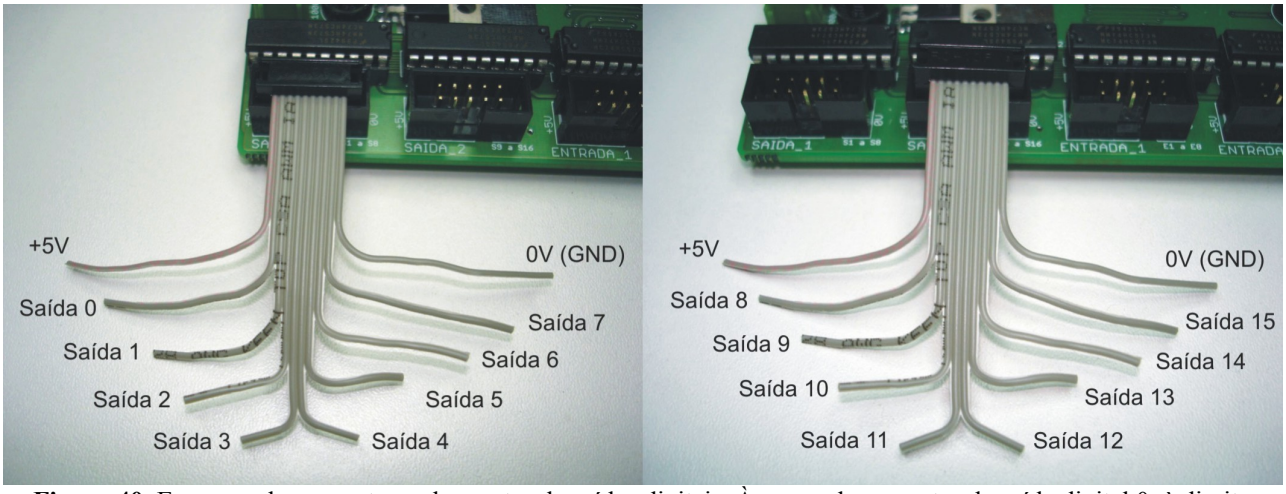


Figura 40. Esquema dos conectores das portas de saídas digitais. À esquerda, conector de saída digital 0; à direita, conector de saída digital 1.

Os conectores dessas portas são muito parecidos com os conectores das portas de entradas digitais. Cada conector possui pinos para 5 Volts, 0 Volts e oito pinos para saídas digitais. Uma tensão de 3,3 Volts nesses pinos indica um nível lógico "1" e uma tensão de 0 Volts indica um nível lógico "0".

Para utilizar essas saídas é necessário um conhecimento básico de eletrônica para permitir a interface delas com dispositivos externos. O uso incorreto dessas portas pode danificar o equipamento. A seguir um exemplos de uso em que oito LEDs são ligados em uma porta de saída digital.

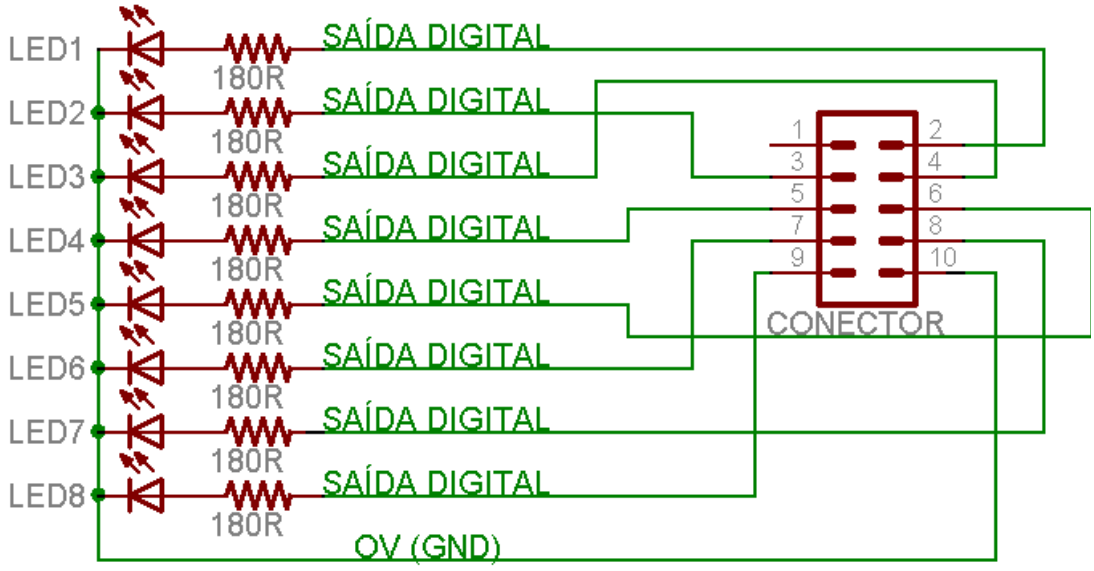


Figura 41. Esquema de ligação de oito LEDs em uma porta de saída digital

Com essa ligação simples podemos controlar o estado dos LEDs através de uma das portas de saída digital. Um outro exemplo interessante é dado abaixo.

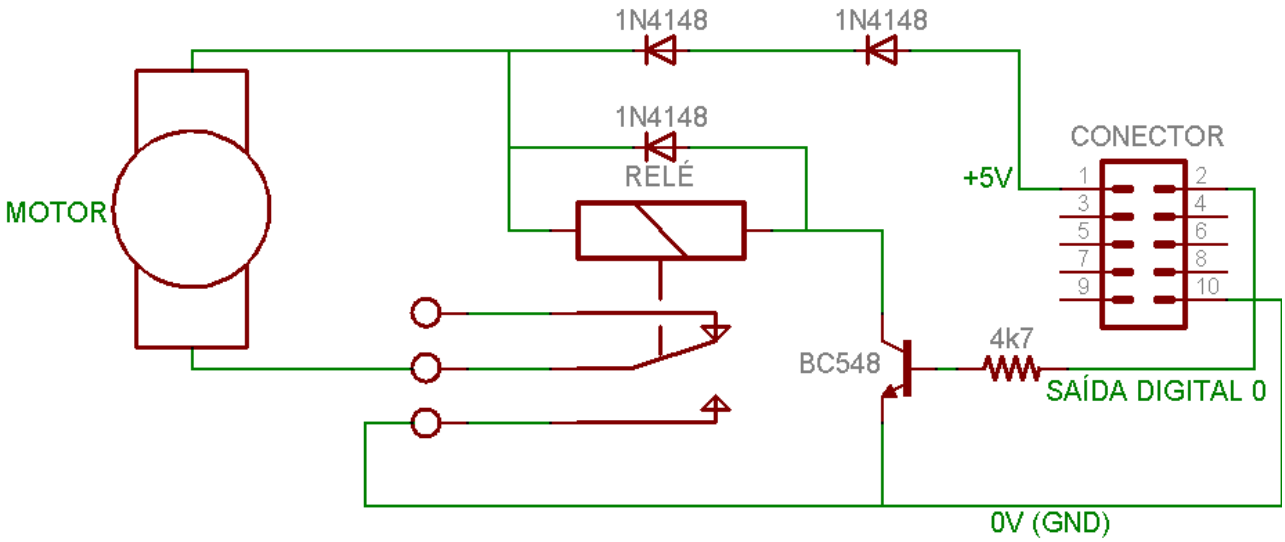


Figura 42. Esquema para acionamento de um motor através de uma porta de saída digital.

Nesse exemplo controlamos o acionamento de um motor utilizando apenas um bit que indica o estado do motor, isto é, ligado ou desligado.

Servo-Motores

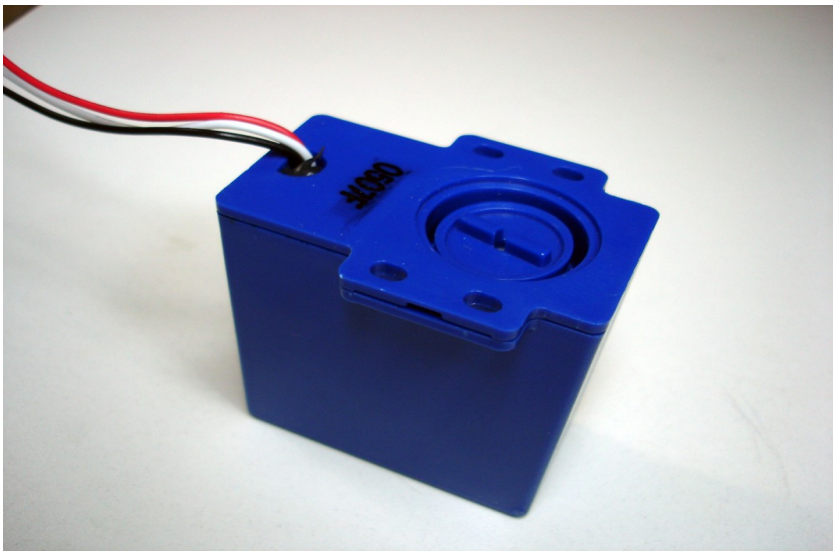


Figura 43. Servo-Motor.

Um servo-motor é um dispositivo que possui um motor conectado a engrenagens com a função de aumentar a força do movimento final. Existe um controle da posição de seu eixo de saída, através de um sinal elétrico. Para executar esta função, os servo-motores possuem uma placa de controle que faz a leitura da posição do eixo de saída e compara com a posição desejada, que depende do sinal elétrico enviado para o servo. Servo-motores são utilizados em muitos sistemas de automação, brinquedos, robos, entre outros.

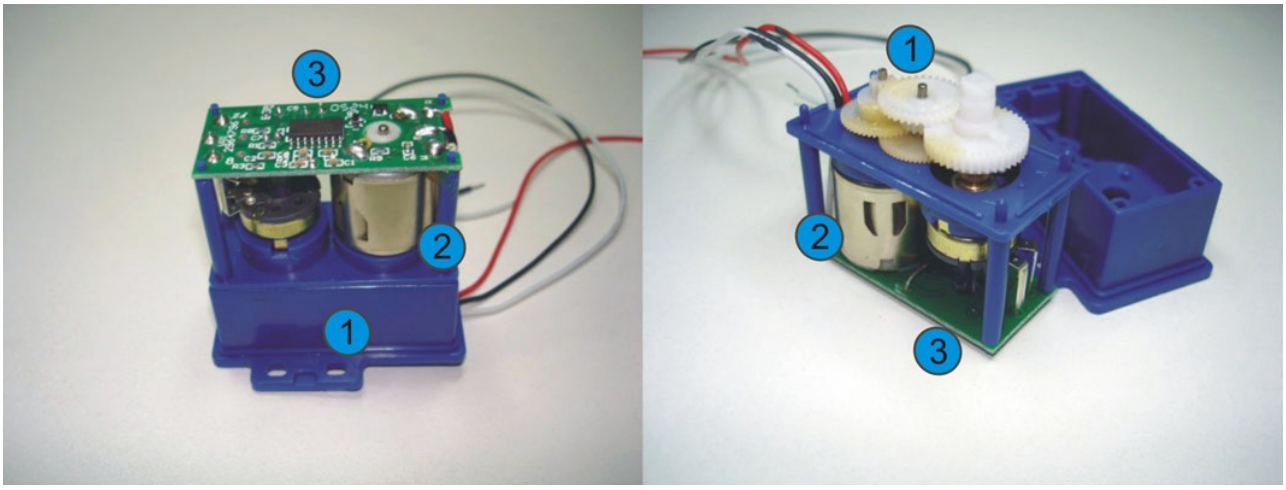


Figura 44. Servo-Motor desmontado. (1) Caixa de Redução, (2) Motor, (3) Placa de Controle.

Em geral os servo-motores são feitos para permitir o controle do movimento de seu eixo em 180º. No entanto é possível fazer uma modificação neles de modo a permitir giro contínuo. Mas com essa modificação não será mais possível controlar a posição do eixo do servo, apenas seu sentido da rotação.

O M*ódulo de Entradas, Saídas e Servo-Motores* possui dois conectores que possibilitam o controle de servo-motores. Esse módulo consegue controlar um total de oito servo-motores distintos. A seguir o esquema da pinagem dos dois conectores utilizados para controle de servo-motores.

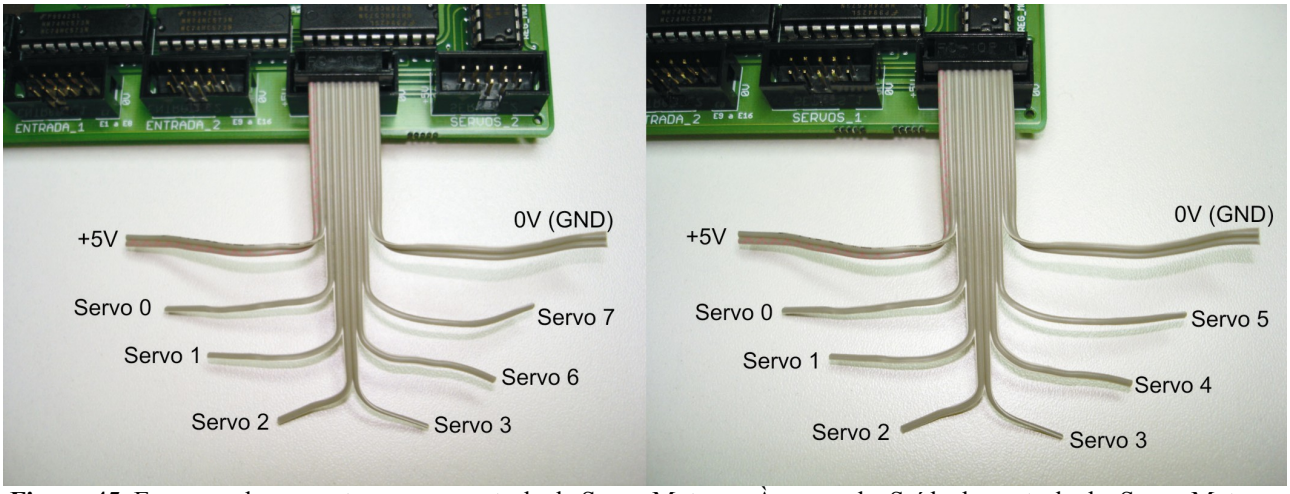


Figura 45. Esquema dos conectores para controle de Servo-Motores: À esquerda, Saída de controle de Servo-Motores 0; à direita, Saída de controle de Servo-Motores 1.

Note que existem pinos com os mesmos sinais de controle nos dois conectores. Cada conector possui dois sinais de controle únicos e quatro compartilhados com o outro conector. Os quatro sinais de controle compartilhados pelos dois conectores podem ser utilizados tanto em um conector quanto no outro. Abaixo o esquema de ligação de seis servo-motores em um único conector.

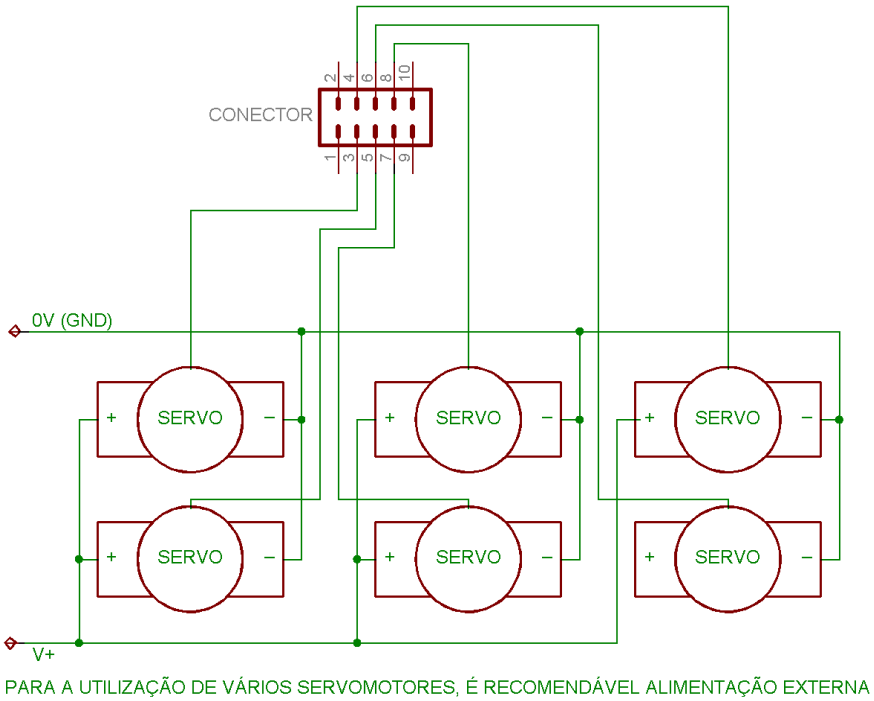


Figura 46. Esquema de ligação de seis servo-motores em um único conector de controle de servos.

Saída de Alimentação

Para auxiliar o desenvolvimento de circuitos externos, assim como no *Módulo de Sensores Genérico*, foi adicionada no *Módulo de Entradas, Saídas e Servo-Motores* uma saída para alimentação de circuitos externos simples. Essa saída pode ser utilizada para alimentar circuitos montados em uma matriz de contatos ou mesmo em placas, desde que não consumam muita corrente. Ela possui três terminais, um com 0 Volts, outro com 5 Volts estabilizados e outro com 12 Volts não estabilizados. A seguir o esquema dos terminais.

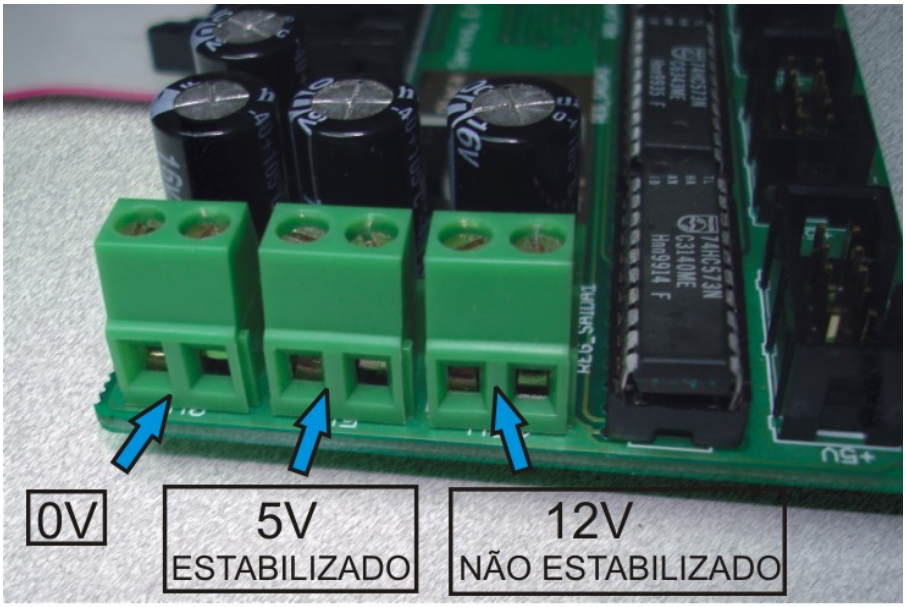


Figura 47. Esquema da saída de alimentação.

Atualização de Firmware

Uma funcionalidade muito interessante que o Kit Didático de Robótica oferece é a possibilidade de atualização do firmware do microcontrolador no *Módulo Principal*. Isso possibilita a adição de funcionalidades e a correção de erros no firmware.

Para possibilitar essa atualização, toda vez que houver correção de erros no firmware ou a adição de novas funcionalidades, serão disponibilizados programas para que essa atualização seja feita por um computador. A seguir será dado como exemplo toda a sequência de atualização do Kit da versão 1.0.0 para a 1.1.0. Esse processo será idêntico ou muito parecido em qualquer atualização.

Primeiramente o Kit que será atualizado deve ser conectado ao computador e ligado. Em seguida deve ser executado o programa de atualização e a seguinte tela será apresentada:

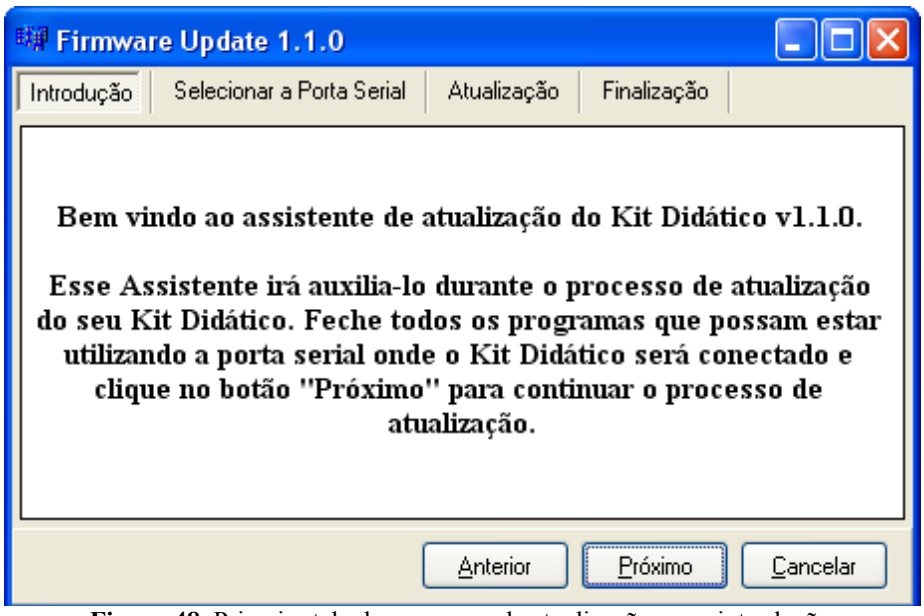


Figura 48. Primeira tela do programa de atualização, uma introdução.

Essa é apenas uma tela de introdução, no entanto ela apresenta uma informação importante, a de que todos os programas que possam estar utilizando a porta serial, onde o Kit que será atualizado está conectado, devem ser fechados. Isso é necessário porque se algum programa já estiver utilizando essa porta não será possivel que o programa de atualização a utilize para comunicação com o Kit, inviabilizando a atualização. Clicando no botão "próximo", será apresentada a seguinte tela.

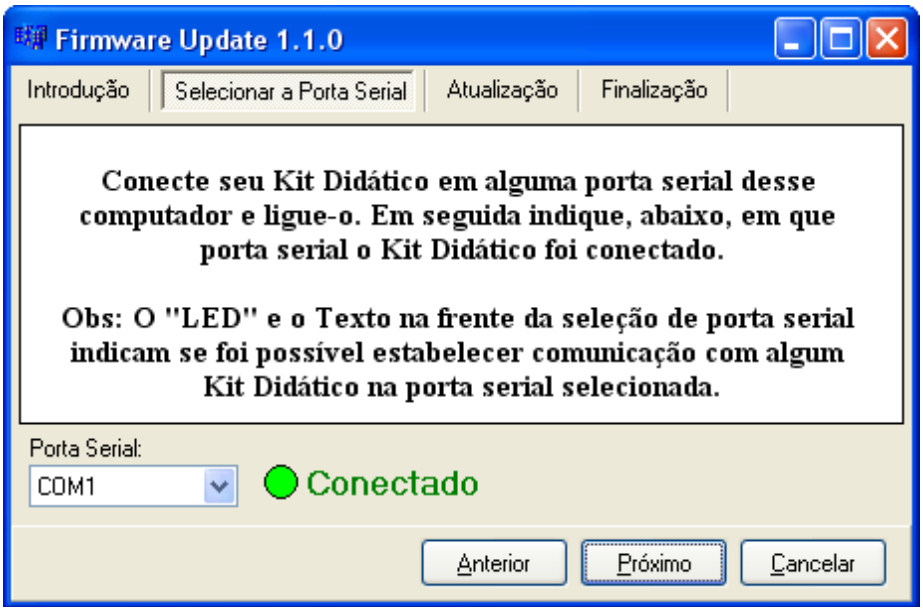


Figura 49. Segunda tela do programa de atualização, seleção de porta serial.

Nessa tela é necessário indicar em que porta serial o Kit está conectado. Ao lado da seleção existe um "LED" e um texto que indica se foi possível estabelecer uma comunicação com algum Kit naquela porta. Caso não seja possível pode ser que exista algum problema. Se o botão "próximo" for pressionado mesmo sem que seja possível estabelecer essa comunicação, o programa de atualização mostrará um aviso, no entanto é possível prosseguir mesmo assim.

Os motivos mais comuns que podem fazer com que não seja possível estabelecer a comunicação é a seleção da porta serial errada, algum programa utilizando a porta selecionada ou o Kit estar desligado. Nesses casos a atualização não irá funcionar até que se resolva o problema. Existe ainda a possibilidade de algum problema com o Kit, no entanto, nesse caso ele não funcionaria com nenhum outro programa de controle.

Existe a possibilidade de que mesmo sem a indicação de comunicação seja possível fazer a atualização. Isso pode ocorrer por uma incompatibilidade entre o programa de atualização e o firmware do Kit. Por exemplo, o programa de atualização da versão 1.0.0 para a 1.1.0, que estamos utilizando como exemplo, irá indicar comunicação apenas se o Kit conectado na porta serial possuir a versão 1.0.0 do firmware. No entanto, podemos fazer o processo de atualização completo mesmo com um Kit que tenha a versão 1.1.0. Para isso, temos apenas que ignorar o aviso de que não foi possível estabelecer a comunicação, apresentado quando apertamos o botão "próximo" nessa segunda tela sem que o programa tenha estabelecido a comunicação.

Outra possibilidade é a de que a atualização de um Kit tenha sido interrompida no meio. Esse tipo de interrupção faz com que o Kit não responda mais a comandos ou que funcione de forma incorreta uma vez que seu firmware não esta completo. Nesses casos é necessário que o

processo de atualização seja refeito do início ao fim sem interrupções. Se houver, nessa segunda tela, a indicação de que não foi possível estabelecer a comunicação com o Kit, esse aviso pode ser ignorado e o processo deve ser continuado.

Após feita a seleção de porta serial o botão “próximo” deve ser pressionado e a seguinte tela será apresentada.

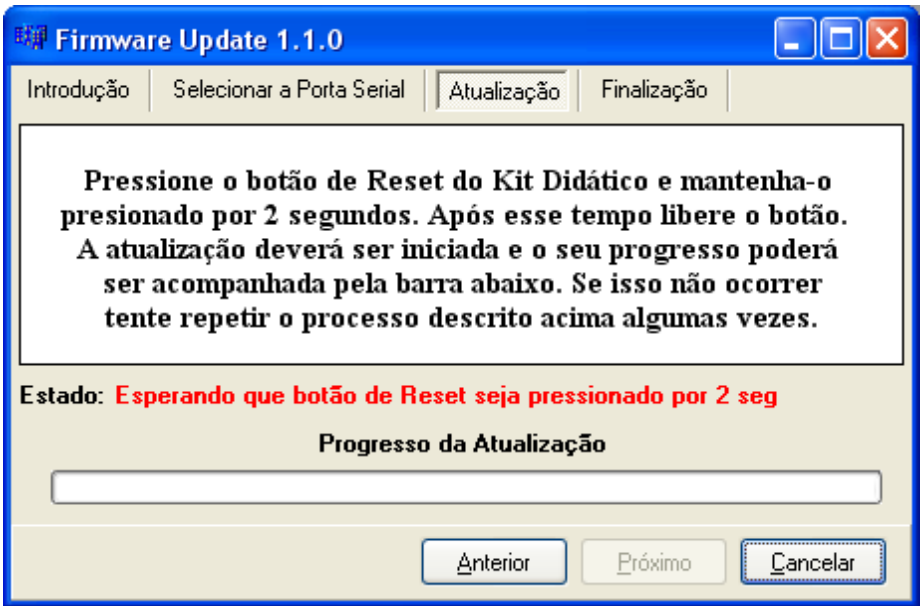


Figura 50. Terceira tela do programa de atualização, atualização.

É nessa tela que o processo de atualização será iniciado. Nela existem as instruções necessárias para o início, uma barra de progresso e um texto indicando o estado da atualização. Para iniciar o processo é necessário pressionar o botão de reset do Kit por aproximadamente 2 segundos e liberá-lo. Se o programa de atualização identificar essa operação o processo será iniciado e o andamento poderá ser acompanhado pela barra de progressão.

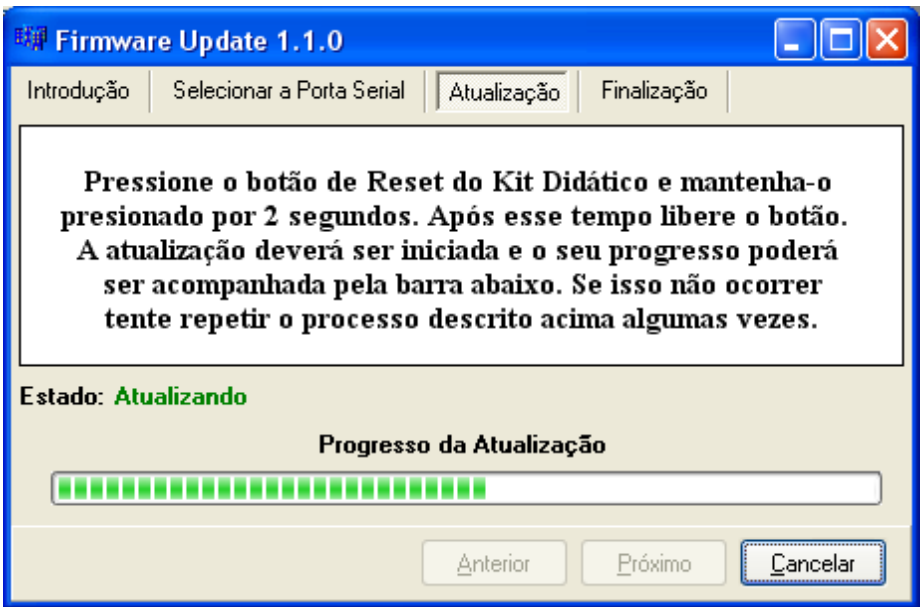


Figura 51. Atualização em andamento.

Caso o processo não inicie, a operação de pressionar o botão de reset e liberá-lo deve ser repetida, pois pode ser que a operação anterior não tenha sido identificada. A atualização demora alguns minutos e se tudo correr bem o Kit estará atualizado ao término desse processo.

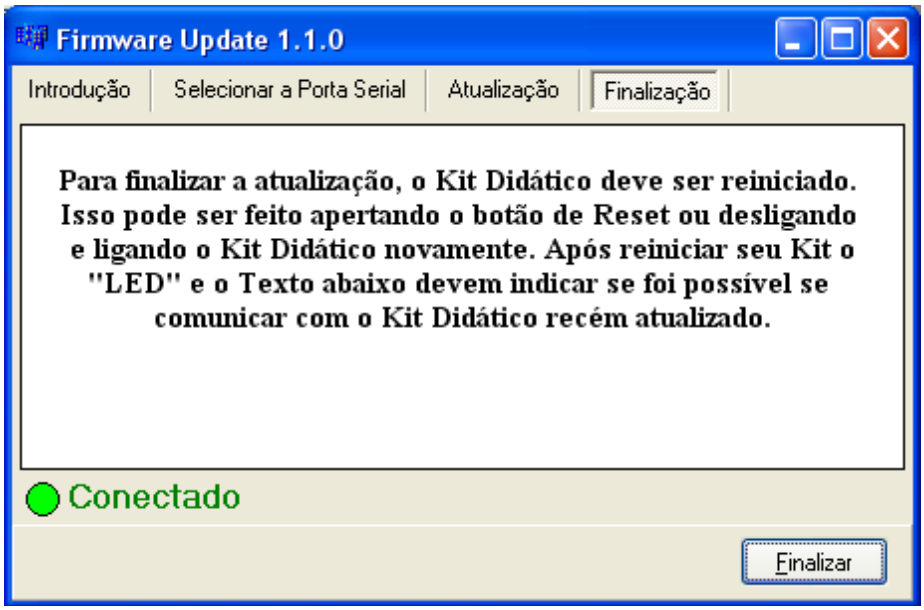


Figura 52. Finalização do processo de atualização.

A quarta e última tela que será apresentada pede que o Kit seja reiniciado. Ela também possui um "LED" e um texto que indicam se foi possível estabelecer uma comunicação com o Kit recém atualizado. Essa comunicação só será possível após o Kit ser reiniciado. Se for possível estabelecer a comunicação com o Kit o processo ocorreu com sucesso e o Kit já estará pronto para ser utilizado com a nova versão de firmware.

É importante lembrar que caso ocorra algum erro durante o processo de atualização ele deverá ser reiniciado.

Montagem e Operação

O Kit Didático de Robótica é um sistema modular formado por vários ítens. Sua montagem é simples e a seguir vamos ver todos os passos necessários para montá-lo corretamente.

É importante lembrar que toda vez que algum ítem do Kit for removido ou adicionado é necessário que o Kit esteja desligado, caso contrário existe a possibilidade de que algum componente seja danificado. Por isso, sempre deve ser verificado se o Kit está desligado antes de qualquer modificação na sua montagem e ele deve ser ligado novamente somente após que a montagem ou a modificação esteja completa.

Durante a montagem e operação do Kit Didático de Robótica é preciso tomar outro cuidado muito importante. Como as placas e cabos dos módulos ficam expostos, é importante cuidar para que os contatos elétricos dos ítens não entrem em curto circuito, o que poderia danificar o Kit.

O primeiro passo na montagem do Kit é montar o *Módulo Principal*. Esse módulo possui uma única placa e para montá-lo é necessário apenas ligar a fonte de alimentação no plug ao

lado do botão “liga-desliga”. Em seguida é necessário conectar esse módulo a um computador através de um cabo serial. Com isso o Kit já poderia ser ligado e as funcionalidades do *Módulo Principal* já poderia ser utilizado.

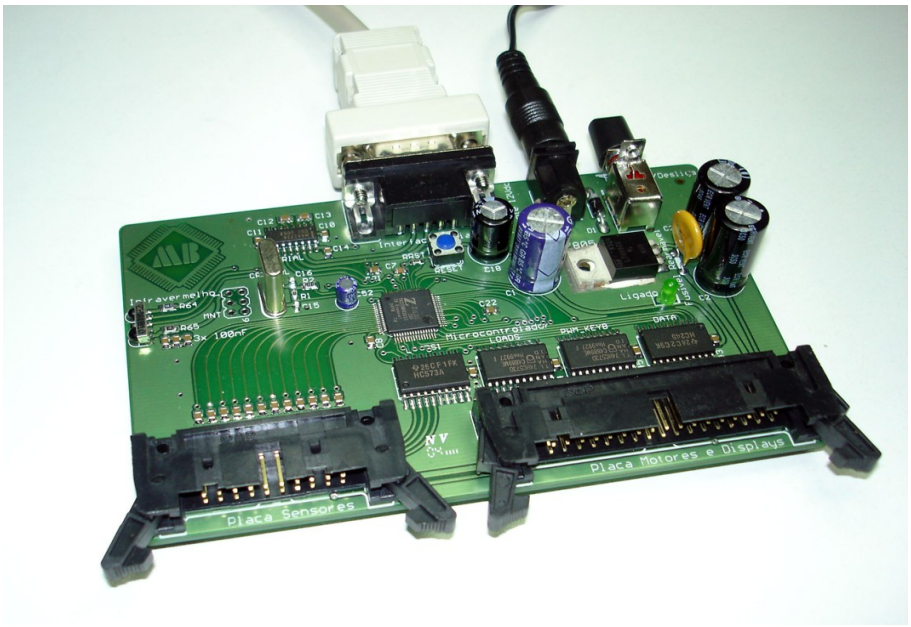


Figura 53. *Módulo Principal* com uma fonte e um cabo serial conectados.

O Kit não oferece muitas funcionalidades apenas com o *Módulo Principal* montado. Por isso nosso próximo passo é montar os outros módulos do Kit. Vamos iniciar pelo *Módulo de Sensores* que utiliza o conector para cabo flat de 20 vias do *Módulo Principal*.

É importante lembrar que o Kit deve estar desligado durante essa montagem. O *Módulo de Sensores* possui uma placa com vários conectores onde são conectados os sensores. Essa placa também possui um conector para cabo flat de 20 vias. Esse conector deve receber o cabo que liga esse módulo ao *Módulo Principal*. A seguir podemos ver na imagem a conexão da placa do *Módulo de Sensores* ao módulo principal através de um cabo flat.

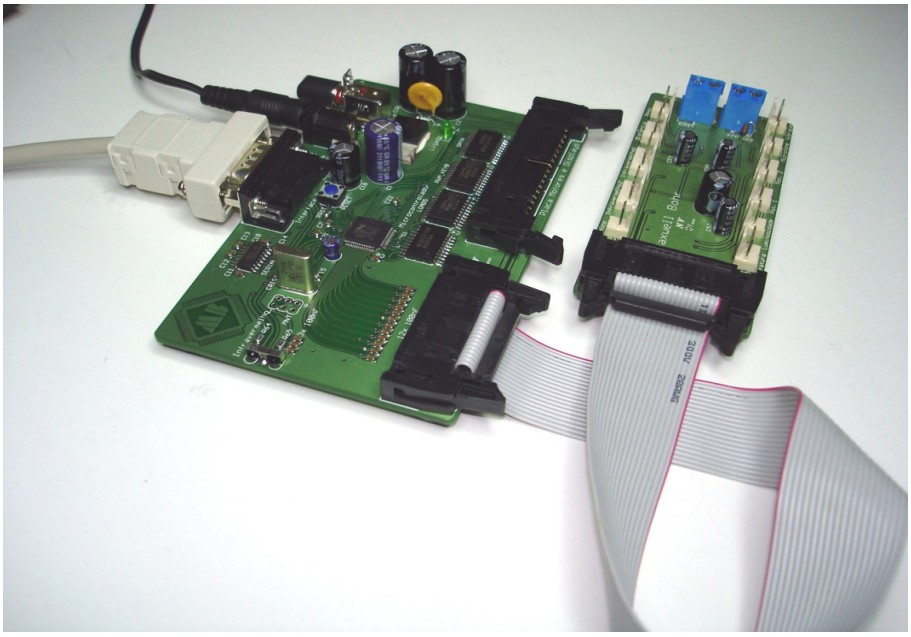


Figura 54. *Módulo de Sensores* conectado ao *Módulo Principal* por um cabo flat de 20 vias.

Agora temos que ligar os sensores. Todo conector de sensor possui próximo um texto com o nome do sensor que deve ser conectado nele. Existe um texto em todos os sensores também. Dessa forma não é necessário decorar a posição dos sensores, é necessário apenas verificar os textos para descobrir o conector correto. Veja na imagem a seguir o texto no conector e no sensor. Todos os conectores e sensores seguem esse padrão.

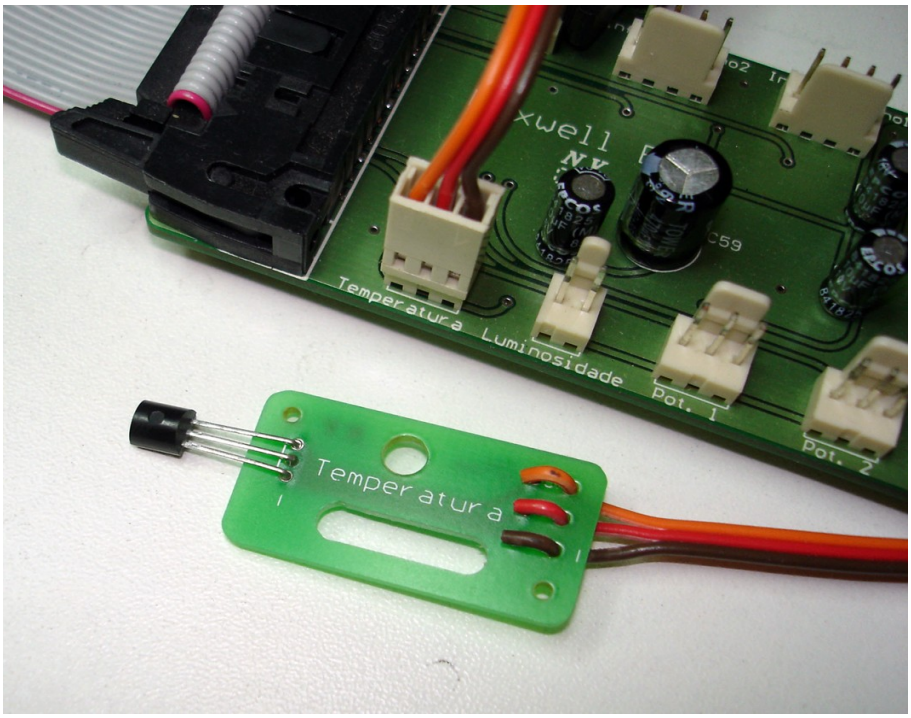


Figura 55. Conector e sensor de temperatura com texto de identificação.

Os conectores são feitos de modo que é possível conecta-los apenas em um sentido, evitando dessa forma a possibilidade de ligação no sentido errado. A seguir uma imagem com o *Módulo de Sensores* com todos os seus sensores conectados.

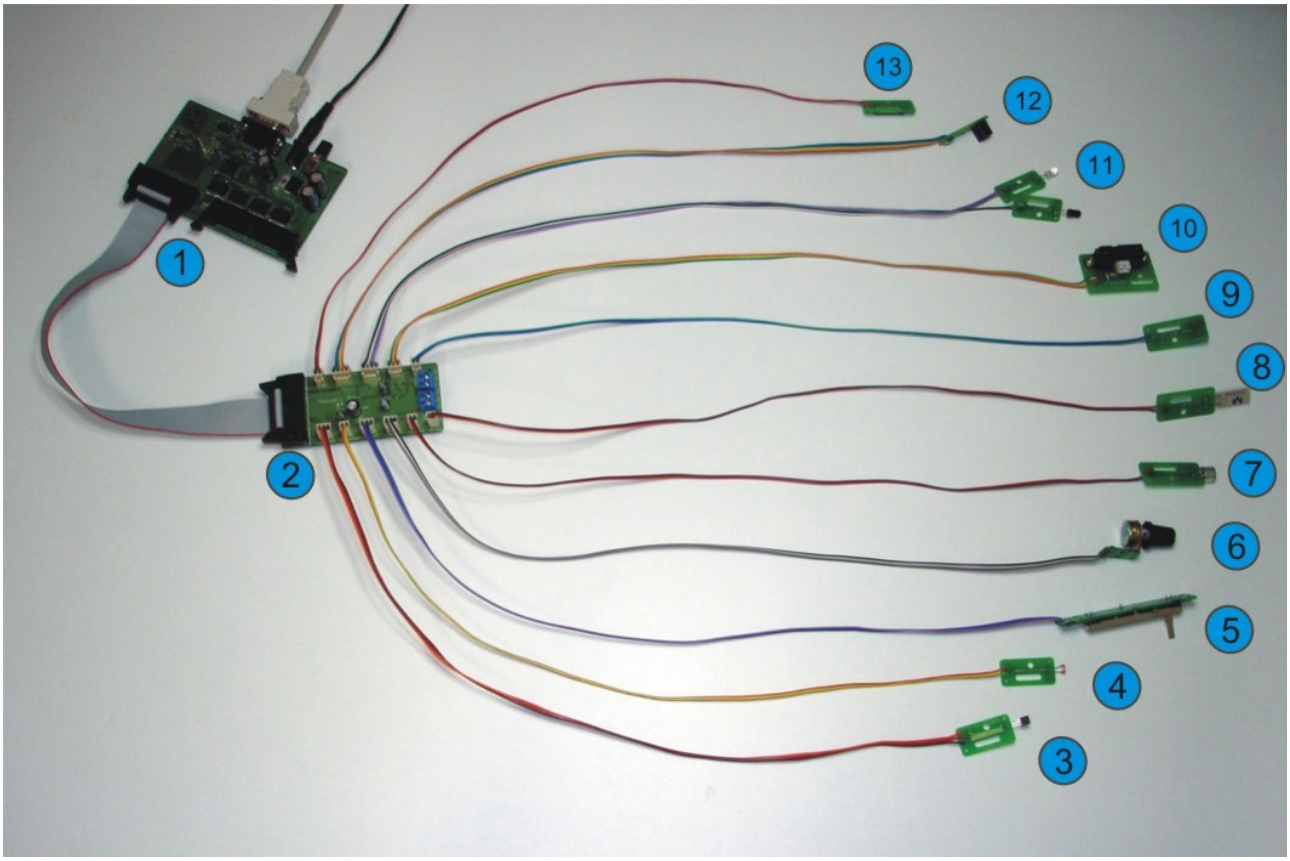


Figura 56. *Módulo de Sensores* conectado ao *Módulo Principal* com todos os seus sensores. (1) *Módulo Principal*, (2) *Módulo de Sensores,*(3) Sensor de Temperatura, (4) Sensor de Luminosidade, (5) Potenciômetro Linear, (6) Potenciômetro Angular, (7) Microfones, (8) Sensor de Vibração, (9) Sensor de Peso, (10) Sensor de Distância, (11) Par Óptico I, (12) Par Óptico II, (13) Chave Magnética.

A figura anterior mostra a montagem completa do *Módulo de Sensores* já conectado ao *Módulo Principal*. Com o Kit montado dessa forma já podemos utilizar as funcionalidades do *Módulo de Sensores* além das presentes no *Módulo Principal*. Essas funcionalidades extras podem ser resumidas em leitura dos valores dos sensores do *Módulo de Sensores*. Com isso já é possível criar projetos práticos com o Kit.

É interessante lembrar que não é necessário que todos os sensores estejam ligados ao mesmo tempo. Se estivermos construindo algum projeto que utiliza apenas um tipo de sensor, os outros não precisam ser conectados, o que é aconselhável, pois as chances de ocorrer algum curto-circuito são reduzidas.

O próximo módulo que veremos como deve ser montado é o *Módulo de Sensores Genérico*. Esse módulo possui um conector para cabos flat de 20 vias que é utilizado para conectá-lo ao *Módulo Principal*. Como o *Módulo Principal* possui apenas um conector para cabos flat de 20 vias, então apenas um módulo com esse tipo de conexão pode ser ligado de cada vez. Deste modo para conectar o *Módulo de Sensores Genérico* ao *Módulo Principal* é necessário desconectar o *Módulo de Sensores* que também utiliza o conector flat de 20 vias.

O *Módulo de Sensores Genérico* possui uma placa com vários conectores genéricos para sensores. Ele não possui ítens extra, logo sua montagem é bem simples, sendo necessária apenas a conexão com o *Módulo Principal*. A seguir uma imagem do Kit com o *Módulo de Sensores Genérico* montado.

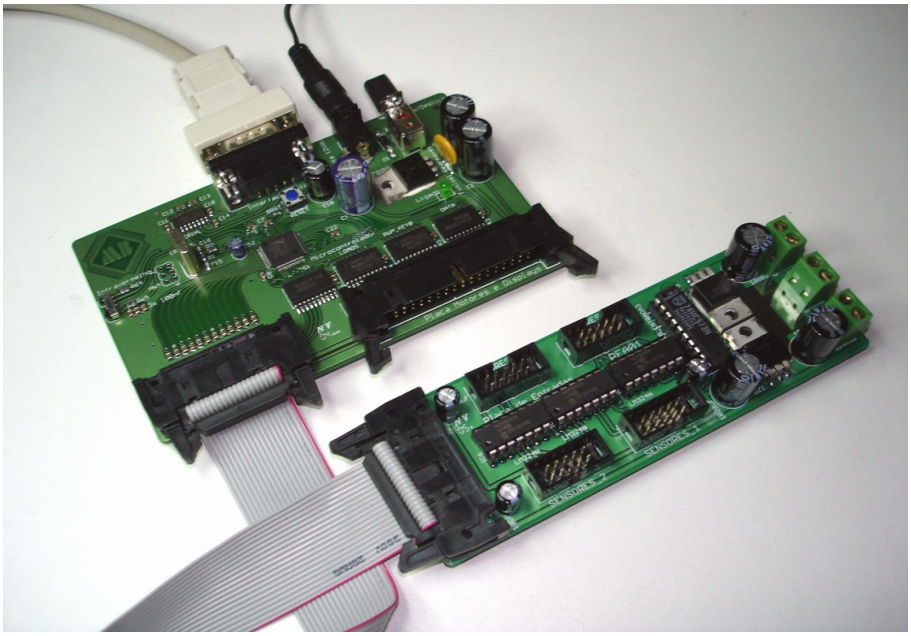


Figura 57. *Módulo de Sensores Genérico* conectado ao *Módulo Principal*.

Apesar de não conter ítens extra esse módulo possui conectores para que sejam conectados sensores montados à parte. É importante que se tenha cuidado para que nenhuma ligação seja feita de forma incorreta nesses conectores, pois isso poderia danificar o equipamento. No tópico que descreve o *Módulo de Sensores Genérico* existe a especificação dos conectores e dicas para a interface eletrônica com sensores.

Toda extensão ao Kit deve atender as especificações do módulo correspondente. Isso aplica-se principalmente ao *Módulo de Sensores Genérico* e ao *Módulo de Entradas, Saídas e Servo-Motores* onde essas extensões serão mais frequentes.

Agora veremos a montagem do *Módulo de Motores e Displays*. Esse módulo possui um conector de 32 vias para cabo flat que é utilizado para conexão com o *Módulo Principal*. Como ele utiliza o conector de 32 vias, qualquer módulo que utilize o conector de 20 vias pode ser conectado simultaneamente a ele no *Módulo Principal*. A seguir uma imagem desse módulo conectado ao *Módulo Principal*.

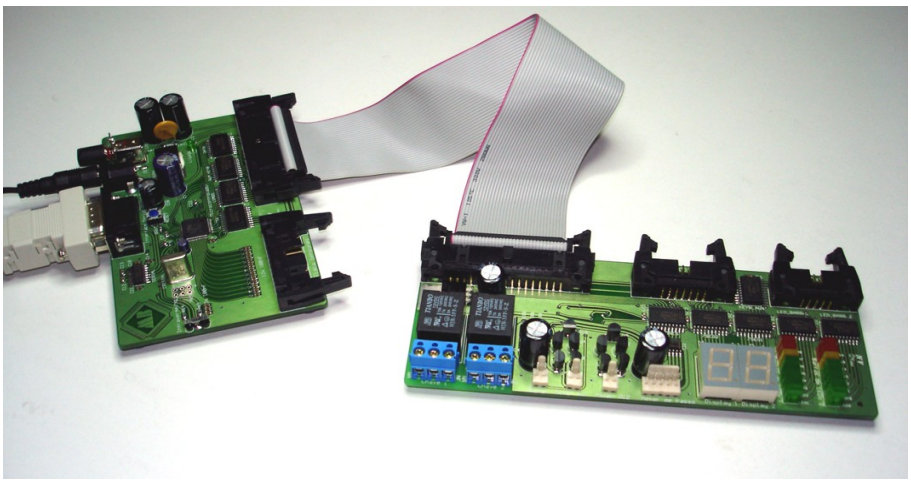


Figura 58. *Módulo de Motores e Displays* conectado ao *Módulo Principal*.

Além dessa conexão de cabo flat para o *Módulo Principal*, esse módulo possui conectores para alguns ítens extras, três motores e um buzzer. Os conectores possuem um texto indicando o ítem que deve ser conectado nele. A seguir uma imagem do módulo com os complementos conectados.

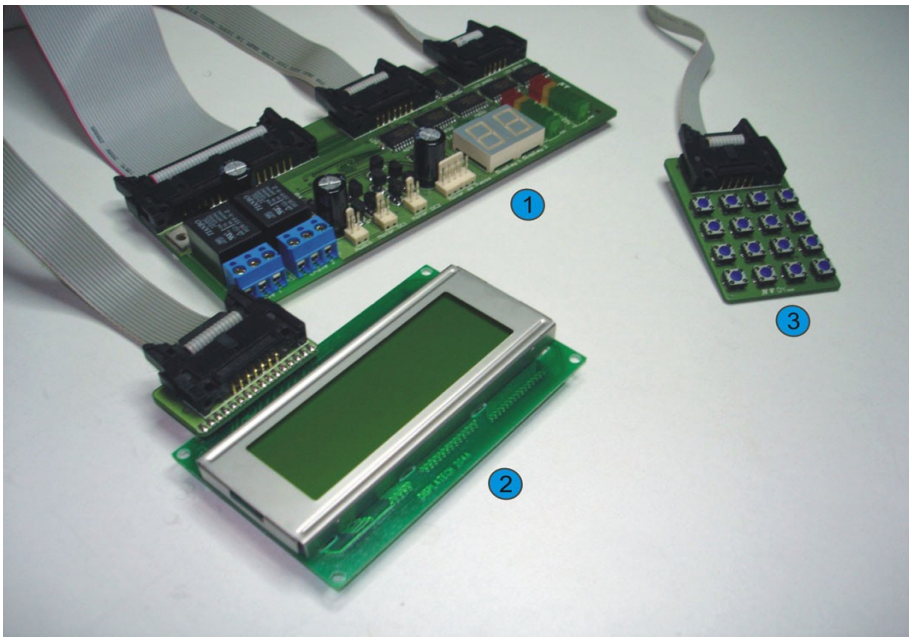


Figura 59. *Módulo de Motores e Displays* com seus ítens extras. (1) *Módulo de Motores e Displays*, (2) Display de Cristal Líquido, (3) Teclado

O último módulo que veremos é o *Módulo de Entradas, Saídas e Servo-Motores*. Esse módulo conecta-se ao *Módulo Principal* através de um cabo flat de 32 vias, ou seja, utilizando o conector destinado ao *Módulo de Motores e Displays*. Dessa forma ele pode estar conectado ao mesmo tempo que algum outro módulo que utilize o conector flat de 20 vias tais como os dois módulos de sensores.

Assim como *Módulo de Sensores Genérico*, esse módulo possui apenas conectores para extensões para o Kit. Dessa forma sua montagem inicial é simples, sendo necessário apenas conecta-lo ao *Módulo Principal*. No entanto, quando alguma extensão for conectada é necessário que essas atendam os padrões desse módulo. No tópico sobre esse módulo é dada todas as especificações do módulo e dos conectores que ele disponibiliza. O uso de extensões for a dos padrões pode danificar o equipamento. A seguir uma imagem com esse módulo montado e conectado ao *Módulo Principal*.

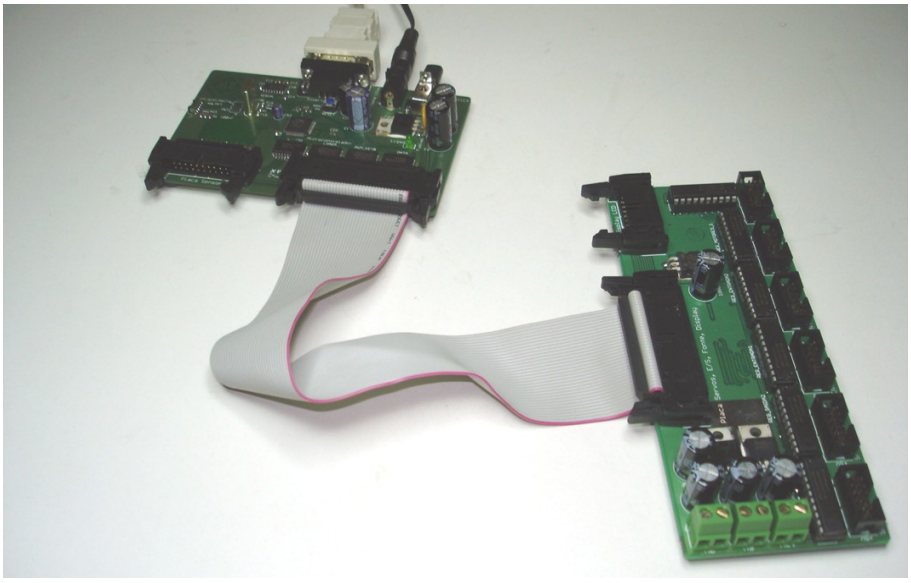


Figura 60. *Módulo de Entradas, Saídas e Servo-Motores* conectado ao *Módulo Principal*.

Programação para Controle do Kit Didático de Robótica

O Kit Didático de Robótica não realiza nenhuma operação a não ser que um programa de controle envie um comando para tal. Isso permite que o Kit seja muito flexível, uma vez que ele não fica preso a nenhuma programação pré-definida o que limitaria suas aplicações. Logo a criação de programas de controle torna-se muito importante. Por isso, nesse tópico, vamos ver como criar esses programas e o primeiro passo para isso é conhecer a biblioteca de controle para o Kit.

Biblioteca de Controle

O Kit Didático de Robótica responde a comandos enviados a partir de uma porta serial. Para isso é necessário abrir uma comunicação serial entre o computador e o Kit e enviar os bytes que codificam o comando desejado. Isso pode ser demorado e criar uma dificuldade desnecessária.

Por isso, foi criada uma biblioteca de controle que cuida de todos os detalhes de comunicação e codificação dos comandos para envio ao Kit, facilitando muito a tarefa de programação para o Kit.

Para permitir uma maior gama de escolhas na forma de criação de programas de controle, essa biblioteca foi desenvolvida em três linguagens de programação. Essas linguagens são Delphi, C++ e C# sendo que essa última versão pode ser utilizada com qualquer linguagem .NET, o que abre ainda mais a gama de escolhas uma vez que existem dezenas de outras linguagens .NET.

Mesmo com linguagens diferentes o uso da biblioteca é muito semelhante em todas elas. Basicamente a biblioteca oferece uma classe que agrupa vários métodos. Essa classe é denominada TKit na versão para Delphi e C++ e apenas Kit na versão para C#. Os métodos dessa classe realizam todas as possíveis operações com o Kit. Por exemplo, existe um método para leitura de sensores, outro para acionar motores, outro para interagir com os displays e assim por diante.

Quando necessário, esses métodos possuem parâmetros como, por exemplo, o texto que deve ser escrito em um display de cristal líquido. Com isso não é necessário saber os detalhes de codificação dos parâmetros no comando enviado ao Kit através da porta serial, tudo isso é tratado automaticamente.

Se houver algum retorno ele também é recebido e retornado pelo método sem a necessidade de conhecimento de nenhum detalhe de transmissão ou codificação dos valores.

Para as explicações será utilizada a linguagem Delphi, no entanto, o uso da biblioteca em outras linguagens será muito semelhante.

Grupos de Métodos

Os métodos da classe de controle da biblioteca são divididos em alguns grupos. O principal critério dessa divisão é a funcionalidade que os métodos fornecem. A seguir uma lista com esses grupos:

- **Métodos de Baixo Nível**: Os métodos desse grupo tratam da transmissão e da codificação dos comandos e são destinados principalmente para uso interno na biblioteca. Esse grupo de

métodos está acessível para serem utilizadas diretamente nos programas de controle, no entanto serão raramente utilizados.

- **Métodos de Sistema**: Agrupa os métodos utilizados para inicializar uma comunicação e dar informações úteis sobre o sistema em que o programa esta rodando.
- **Métodos do *Módulo Principal***: Como o próprio nome sugere, esse grupo contém os métodos que fornecem as funcionalidades do *Módulo Principal*.
- **Métodos dos *Módulos de Sensores***: Os métodos desse grupo tratam da leitura de sensores tanto do *Módulo de Sensores* quanto do *Módulo de Sensores Genérico*.
- **Métodos do *Módulo de Motores e Displays***: Métodos que controlam as funcionalidades oferecidas pelo *Módulo de Motores e Displays*.
- **Métodos do Display de Cristal Líquido**: Métodos de controle do display de cristal líquido.
- **Métodos do *Módulo de Entradas, Saídas e Servo-Motores***: Fornece os métodos para controle do *Módulo de Entradas, Saídas e Servo-Motores.*

É importante citar que os métodos específicos para um módulo necessitam que esse esteja conectado para que seu uso funcione corretamente. A seguir uma lista com as principais funções da biblioteca de controle separadas por grupos. Uma descrição mais detalhada dessas funções será dada ainda nesse documento.

WriteByte	KeyboardReadNow
Write	KeyboardReadBuffer
ReadByte	KeyboardClearBuffer
Read	LEDBarSetLevel
Purge	LEDBarSetStatus
SendCommand	Disp7SegSetNum
SendCommandAndRecv	Disp7SegSetStatus
	AudioOn
OpenCommunication	DCMotorOn
CloseCommunication	StepMotorOn
GetInstalledSerialPorts	RelayOn
IsConnected	LCDOn
NOP	LCDWriteText
DeviceName	LCDGotoXY
DeviceStatus	LCDClear

IrdaOn	**LCDCursorOn**
FlashWrite	**LCDBacklightOn**
FlashRead	
FlashErase	**DigitalPortRead**
	DigitalPortWrite
SensorRead	**ServoMotorOn**
SensorReadAll	**ServoMotorOff**
SensorBufferOn	
SensorReadBuffer	
SensorClearBuffer	

Legenda:

- Baixo Nível
- Sistema
- Módulo Principal
- Módulos de Sensores
- Módulo de Motores e Displays
- Display de Cristal Líquido
- Módulo de Entradas, Saídas e Servo-Motores

Métodos de Baixo Nível

Os métodos desse grupo tratam da transmissão e da codificação dos comandos e são destinados principalmente para uso interno na biblioteca. Esse grupo de métodos está acessível para serem utilizadas diretamente nos programas de controle, no entanto serão raramente utilizados. A seguir uma tabela com os métodos que fazem parte desse grupo com uma pequena descrição da funcionalidade que eles implementam.

Método	Descrição
WriteByte	Envia um byte pela porta serial.
Write	Envia vários bytes pela porta serial.
ReadByte	Recebe um byte pela porta serial.
Read	Recebe vários bytes pela porta serial.
Purge	Limpa o buffer de entrada e saída da porta serial.

Método	Descrição
SendCommand	Envia um comando para um Kit.
SendCommandAndRecv	Envia um comando para um Kit e recebe um retorno.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

WriteByte

Declaração	`Procedure WriteByte(b : Byte);`
Descrição	Envia um byte pela porta serial.
Parâmetros	B **:** O valor do byte que deve ser enviado pela porta serial.
Retorno	Nenhum.

Write

Declaração	`Procedure Write(data : DynByteArray);`
Descrição	Envia um array de bytes pela porta serial.
Parâmetros	Data : Array de bytes que devem ser enviado pela porta serial.
Retorno	Nenhum.

ReadByte

Declaração	`Function ReadByte() : Byte;`
Descrição	Recebe byte pela porta serial.
Parâmetros	Nenhum.
Retorno	Retorna o byte lido.

Read

Declaração	`Procedure Read(var data : DynByteArray; num : Integer);`
Descrição	Recebe um determinado número de bytes pela porta serial.
Parâmetros	Data : Array onde os bytes lidos devem ser armazenados.

	Num : Número de bytes que devem ser lidos.
Retorno	Nenhum.

Purge

Declaração	`Procedure Purge();`
Descrição	Limpa o buffer de entrada e saída da porta serial.
Parâmetros	Nenhum.
Retorno	Nenhum.

SendCommand (1)

Declaração	`Procedure SendCommand(cmd : Byte); overload;`
Descrição	Envia um comando para um Kit Didático.
Parâmetros	Cmd : Comando que deve ser enviado.
Retorno	Nenhum.

SendCommand (2)

Declaração	`Procedure SendCommand(cmd : Byte; parm : DynByteArray); overload;`
Descrição	Envia um comando com parâmetros para um Kit Didático.
Parâmetros	Cmd : Comando que deve ser enviado. Parm : Parâmetros do comando.
Retorno	Nenhum.

SendCommandAndRecv (1)

Declaração	`Procedure SendCommandAndRecv(cmd : Byte; var ret : DynByteArray; num : Integer); overload;`
Descrição	Envia um comando para um Kit Didático e recebe um retorno.
Parâmetros	Cmd : Comando que será enviado. Ret : Array que armazena o retorno do comando.

	Num : Número de bytes do retorno. Se esse valor for negativo será considerado que o comando tem retorno de tamanho variável. Nesse caso serão lidos pela serial dois bytes que formarão um valor de 16 bits que indicará o número de bytes total do retorno.
Retorno	Nenhum.

SendCommandAndRecv (2)

Declaração	`Procedure SendCommandAndRecv(cmd : Byte; var ret : DynByteArray); overload;`
Descrição	Envia um comando para um Kit Didático e recebe um retorno.
Parâmetros	Cmd : Comando que deve ser enviado. Ret : Array que armazena o retorno do comando.
Retorno	Nenhum.

SendCommandAndRecv (3)

Declaração	`Procedure SendCommandAndRecv(cmd : Byte; parm : DynByteArray; var ret : DynByteArray; num : Integer); overload;`
Descrição	Envia um comando com parâmetros para um Kit Didático e recebe um retorno.
Parâmetros	Cmd : Comando que deve ser enviado. Parm : Parâmetros do comando. Ret : Array que armazena o retorno. Num : Número de bytes do retorno. Se esse valor for negativo será considerado que o comando tem retorno de tamanho variável. Nesse caso serão lidos pela serial dois bytes que formarão um valor de 16 bits que indicará o número de bytes total do retorno.
Retorno	Nenhum.

SendCommandAndRecv (4)

Declaração	`Procedure SendCommandAndRecv(cmd : Byte; parm : DynByteArray; var ret : DynByteArray); overload;`

Descrição	Envia um comando com parâmetros para um Kit Didático e recebe um retorno sem tamanho definido.
Parâmetros	Cmd : Comando que deve ser enviado. Parm : Parâmetros do comando. Ret : Array que armazena o retorno.
Retorno	Nenhum.

Métodos de Sistema

Esse grupo contém os métodos utilizados para inicializar uma comunicação e dar informações úteis sobre o sistema em que o programa esta rodando.

Alguns desses métodos serão utilizados em todos os programas que controlam um Kit, pois, para que os métodos de controle possam ser chamados é necessário primeiramente abrir um canal de comunicação com o Kit, o que pode ser feito com o método OpenCommunication que pertece a esse grupo. A seguir uma tabela com as funções desse grupo.

Método	Descrição
OpenCommunication	Inicializa a comunicação com um Kit através de uma porta serial.
CloseCommunication	Finaliza a comunicação com um Kit.
GetInstalledSerialPorts	Detecta todas as portas seriais instaladas no sistema operacional.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

OpenCommunication

Declaração	`Procedure OpenCommunication(port : String);`
Descrição	Inicializa comunicação utilizando a porta serial passada como parâmetro.
Parâmetros	Port : Porta serial que será utilizada na comunicação com o Kit Didático. ("COM1", "COM2", ...)
Retorno	Nenhum.

CloseCommunication

Declaração	`Procedure CloseCommunication();`
Descrição	Finaliza a comunicação.
Parâmetros	Nenhum.
Retorno	Nenhum.

GetInstalledSerialPorts

Declaração	`Procedure GetInstalledSerialPorts(list : TStrings);`
Descrição	Detecta todas as portas seriais instaladas no sistema operacional.
Parâmetros	List : Classe do tipo TStrings que irá armazenar a lista de portas seriais instaladas no sistema.
Retorno	Nenhum.

Métodos do Módulo Principal

Como o próprio nome sugere, esse grupo contém os métodos que fornecem as funcionalidade do *Módulo Principal*. A seguir uma tabela com esses métodos com uma pequena descrição.

Método	Descrição
IsConnected	Verifica se existe algum Kit conectado e ligado na porta serial.
NOP	Envia um comando 'NOP'. Não realiza nenhuma operação.
DeviceName	Retorna uma String com o nome do Kit Didático.
DeviceStatus	Retorna a estrutura de dados que armazena o estado do Kit Didático no momento.
IrdaOn	Liga ou desliga o modo de envio de comandos através da porta de comunicação IrDA.
FlashWrite	Grava dados na memória não volátil do Kit Didático.
FlashRead	Lê dados gravado na memória não volátil do Kit Didático.
FlashErase	Apaga todo o conteúdo da memória não volátil do Kit Didático.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

IsConnected

Declaração	`Function IsConnected() : Boolean;`
Descrição	Verifica se existe algum Kit Didático conectado e ligado na porta serial atual.
Parâmetros	Nenhum.
Retorno	Verdadeiro se houver um Kit Didático conectado e ligado à porta serial atual ou Falso caso contrário.

NOP

Declaração	`Procedure NOP();`
Descrição	Envia um comando 'NOP'. Não realiza nenhuma operação. É utilizado para verificar a presença de algum Kit Didático ligado à uma porta serial. Emite uma exceção caso não exista resposta.
Parâmetros	Nenhum.
Retorno	Nenhum.

DeviceName

Declaração	`Function DeviceName() : String;`
Descrição	Retorna uma String com o nome do Kit Didático.
Parâmetros	Nenhum.
Retorno	Descrição do Kit Didático.

DeviceStatus

Declaração	`Procedure DeviceStatus(var struct_status : DynByteArray);`
Descrição	Retorna a estrutura de dados que armazena o estado do Kit Didático no momento.
Parâmetros	Struct_status : Estrutura que irá armazenar o estado do Kit Didático.
Retorno	Nenhum.

IrdaOn

Declaração	`Procedure IrdaOn(tf : Boolean);`
Descrição	Liga ou desliga o modo de envio de comandos através da porta de comunicação IrDA. Quando esse modo está ativado todos os comandos enviados serão redirecionados para a porta IrDA para execução remota por algum outro Kit Didático que receba esse comando. A porta IrDA tem um alcance de aproximadamente 1 metro e para que seja possível a comunicação entre dois Kits Didático suas portas IrDA devem estar direcionadas uma para a outra e sem nenhum objeto interrompendo a visada entre elas.
Parâmetros	On : Especifica se o modo IrDA deve ser ligado ou desligado.
Retorno	Nenhum.

FlashWrite

Declaração	`Procedure FlashWrite(data : DynByteArray);`
Descrição	Grava dados na memória não volátil do Kit Didático. Esses dados poderão ser recuperados mesmo após o Kit Didático ser desligado. O conteúdo anterior da memória é apagado.
Parâmetros	Data : Array com dados que devem ser gravados na memória. O número máximo de bytes que podem ser gravados é 100.
Retorno	Nenhum.

FlashRead

Declaração	`Procedure FlashRead(var data : DynByteArray; num : Integer);`
Descrição	Lê dados gravado na memória não volátil do Kit Didático.
Parâmetros	Data : Array que irá armazenar os dados lidos da memória. Num : Número de bytes que devem ser lidos. O número máximo de bytes é 100.
Retorno	Nenhum.

FlashErase

Declaração	`Procedure FlashErase();`
Descrição	Apaga todo o conteúdo da memória não volátil do Kit Didático.
Parâmetros	Nenhum.

Retorno	Nenhum.

Métodos dos Módulos de Sensores

Os métodos desse grupo tratam da leitura de sensores tanto do *Módulo de Sensores* quanto do *Módulo de Sensores Genérico*. Esses métodos serão utilizados toda vez que se desejar ler o sinal de algum sensor. A seguir uma tabela com os métodos desse grupo.

Método	Descrição
SensorReadNow	Lê um canal de sensor.
SensorReadAll	Lê todos os canais de sensores em uma única operação.
SensorBufferOn	Liga ou desliga a aquisição bufferizada.
SensorReadBuffer	Lê todos os valores de leituras buferizadas de sensores.
SensorClearBuffer	Limpa o buffer de leituras de sensor.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

SensorReadNow

Declaração	`Function SensorReadNow(sensor : Integer) : Integer;`
Descrição	Lê um canal de sensor. Existem 16 canais para sensores que podem ser lidos com esse comando. Abaixo uma lista com o número dos canais, os sensores conectados a cada canal no Módulo de Sensores e o conector onde se encontra o canal no Módulo de Sensores Genérico.
Parâmetros	Sensor : Número do canal que deve ser lido.
Retorno	A leitura do canal.

Canal	Módulo de Sensores	Módulo de Sensores Genérico
00	Potenciômetro II	Conector 1: Primeira entrada analógica
01	Microfone	Conector 1: Segunda entrada analógica

Canal	Módulo de Sensores	Módulo de Sensores Genérico
02	Vibração	Conector 1: Terceira entrada analógica
03	Chave Magnética	Conector 1: Entrada digital
04	Temperatura	Conector 2: Primeira entrada analógica
05	Luminosidade	Conector 2: Segunda entrada analógica
06	Potenciômetro I	Conector 2: Terceira entrada analógica
07	Não utilizada	Conector 2: Entrada digital
08	Auxiliar I	Conector 3: Primeira entrada analógica
09	5V	Conector 3: Segunda entrada analógica
10	12V	Conector 3: Terceira entrada analógica
11	Par Óptico I	Conector 3: Entrada digital
12	Peso	Conector 4: Primeira entrada analógica
13	Distância	Conector 4: Segunda entrada analógica
14	Auxiliar II	Conector 4: Terceira entrada analógica
15	Par Óptico II	Conector 4: Entrada digital

SensorReadAll

Declaração	`Procedure SensorReadAll(var data : DynIntegerArray);`
Descrição	Lê todos os canais de sensores em uma única operação. Os canais são lidos na ordem em que aparecem na tabela encontrada na descrição da função SensorReadNow.
Parâmetros	Data : Array com os valores de todos os canais.
Retorno	Nenhum.

SensorBufferOn

Declaração	`Procedure SensorBufferOn(tf : Boolean; sensor, interval : Integer);`
Descrição	Liga ou desliga a aquisição bufferizada. Quando este modo de aquisição está ligado, o Kit Didático faz leituras de um canal de sensor em intervalos determinados e guarda a leitura em um buffer. Esse buffer pode conter até 90 leituras e quando

	cheio as leituras mais novas vão sobrescrevendo as mais antigas. O intervalo entre as aquisições é dado em milisegundos e pode ser de 1ms ou até mesmo de várias horas. Nesse modo, a aquisição não depende do computador, ela continuará sendo feita mesmo que o Kit Didático seja desconectado da porta serial. Dessa forma, se for utilizado um intervalo de por exemplo 1 minuto é possível fazer um monitoramento de algum parâmetro sem a presença de um computador por 1 hora e meia, se esse intervalo for de 10 minutos podemos ter um monitoramento por 15 horas. O Kit Didático suporta a leitura de apenas um sensor por vez nesse modo.
Parâmetros	Tf : Determina se a aquisição deve ser ligada ou desligada. Se esse parâmetro for false, que significa que a aquisição bufferizada deve ser desligada, os outros parâmetros serão ignorados. Sensor : Define qual canal de sensor será monitorado. Interval : Intervalo entre as aquisições em milésimos de segundos. Esse valor pode variar de 1 à 4.294.967.296 milisegundos.
Retorno	Nenhum.

SensorReadBuffer

Declaração	`Procedure SensorReadBuffer(var data : DynIntegerArray);`
Descrição	Lê todos os valores de leituras buferizadas de sensores.
Parâmetros	Data : Array com os valores lido do buffer de aquisições.
Retorno	Nenhum.

SensorClearBuffer

Declaração	`Procedure SensorClearBuffer();`
Descrição	Limpa o buffer de leituras de sensor.
Parâmetros	Nenhum.
Retorno	Nenhum.

Métodos do Módulo de Motores e Displays

Métodos que controlam as *funcionalidades* oferecidas pelo *Módulo de Motores e Displays*. Esses métodos dependem desse módulo para funcionarem. A seguir uma tabela com esses métodos e uma

pequena a descrição.

Método	Descrição
KeyboardReadNow	Lê que tecla do teclado está pressionada no momento.
KeyboardReadBuffer	Lê o valor mais antigo armazenado no buffer do teclado.
KeyboardClearBuffer	Limpa o buffer do teclado.
LEDBarSetLevel	Define o nível de uma barra de LEDs.
LEDBarSetStatus	Define quais LEDs individuais da barra de LEDs serão ligados.
Disp7SegSetNum	Define o número que será apresentado em uma display de 7 segmentos.
Disp7SegSetStatus	Define os segmentos de um display de 7 segmentos que deverão ser ligados.
AudioOn	Liga ou desliga o buzzer que emite um beep.
DCMotorOn	Liga ou desliga um motor DC.
StepMotorOn	Liga ou desliga o motor de passos.
RelayOn	Liga ou desliga uma chave eletromagnética (Relé).

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

KeyboardReadNow

Declaração	`Function KeyboardReadNow() : Byte;`
Descrição	Lê que tecla do teclado está pressionada no momento.
Parâmetros	Nenhum.
Retorno	O valor da tecla pressionada que pode variar de 0 à 15. Se não houver nenhuma tecla pressionada retorna 255(0xFF).

KeyboardReadBuffer

Declaração	`Function KeyboardReadBuffer() : Byte;`
Descrição	Lê o valor mais antigo armazenado no buffer do teclado. O buffer possui 10

	posições.
Parâmetros	Nenhum.
Retorno	O valor lido no buffer. Se o buffer estiver vazio retorna 255(0xFF).

KeyboardClearBuffer

Declaração	`Procedure KeyboardClearBuffer();`
Descrição	Limpa o buffer do teclado.
Parâmetros	Nenhum.
Retorno	Nenhum.

LEDBarSetLevel

Declaração	`Procedure LEDBarSetLevel(bar,level : Integer);`
Descrição	Define o nível de uma barra de LEDs.
Parâmetros	Bar : Barra de LEDs que se deseja definir o nível. Level : Nível da barra de LEDs. Pode assumir um valor de 0 à 8.
Retorno	Nenhum.

LEDBarSetStatus

Declaração	`Procedure LEDBarSetStatus(bar : Integer; status : Byte);`
Descrição	Define quais LEDs individuais da barra de LEDs serão ligados.
Parâmetros	Bar : Barra que se deseja definir estado. Status : Define quais LEDs serão ligados ou desligados. Os 8 bits desse parâmetro definem os LEDs que ficarão ligados ou desligados. Cada LED da barra de LEDs está relacionado a 1 bit do valor de 8 bits do parâmetro 'status'. Um bit de valor 1 significa que esse LED será ligado e um bit 0 que ele será desligado. Assim podemos ligar todos os LEDs da barra com um status igual a 255(decimal), que em binário é representado por 11111111b. Se o valor de 'status' fosse 85, que é representado em binário por 10101010b teríamos os LEDs ligados alternadamente.
Retorno	Nenhum.

Disp7SegSetNum

Declaração	`Procedure Disp7SegSetNum(disp, num : Integer);`
Descrição	Define o número que será apresentado em uma display de 7 segmentos. Além dos digitos de 0 à 9, suporta também apresentar os digitos hexadecimais A, B, C, D, E e F.
Parâmetros	Disp : Display numérico que se deseja definir o valor. Num : Um valor de 0 à 15 indicando o número que deve ser apresentado.
Retorno	Nenhum.

Disp7SegSetStatus

Declaração	`Procedure Disp7SegSetStatus(disp : Integer; status : Byte);`
Descrição	Define os segmentos de um display de 7 segmentos que deverão ser ligados. Esse comando permite o controle total sobre o display.
Parâmetros	Disp : Display numérico que se deseja definir estado. Status : Os 8 bits desse parâmetro indicam qual segmento de LED do display será ligado ou desligado. A seguir o esquema de um display de 7 segmento e os bits que representam seus segmentos. **Esquema** / **Segmento** / **Descrição** / **Bit** ---(a)--- / a / Superior / 0 (f) (b) / b / Superior Direito / 1 ---(g)--- / c / Inferior Direito / 2 (e) (c) / d / Inferior / 3 ---(d)--- .(h) / e / Inferior Esquerdo / 4 f / Superior Esquerdo / 5 g / Central / 6 h / Ponto / 7 Para apresentar o número 1 é necessário que os segmentos 'b' e 'c' sejam acesos. Dessa forma o parâmetro que devemos utilizar é 6 , que em binário é representado por 00000110b. Como podemos ver sem problemas pela representação binária, os bits 1 e 2 (a contagem é feita da direita para a esquerda)

	são 1, o que indica que os segmentos representados por esses devem ser acesos. Os outros segmentos ficam apagados pois os bits que os representam são 0.
Retorno	Nenhum.

AudioOn

Declaração	`Procedure AudioOn(time, freq : Integer);`
Descrição	Liga ou desliga o buzzer que emite um beep.
Parâmetros	Time : Tempo que o som permanecerá ligado. Pode variar de 0 à 65536. Um valor 0 nesse parametro interrompe a emissão de som imediatamente. Freq : Freqüência do som. Pode variar de 0 à 65536. Um valor 0 nesse parametro interrompe a emissão de som imediatamente.
Retorno	Nenhum.

DCMotorOn

Declaração	`Procedure DCMotorOn(motor, dir, speed : Integer);`
Descrição	Liga ou desliga um motor DC.
Parâmetros	Motor : Motor que deve ser ligado. Dir : Direção do movimento. Pode assumir os valores 0 ou 1. Speed : Velocidade. Um valor 0 nesse parametro interrompe o movimento do motor e é utilizado para desligar-lo.
Retorno	Nenhum.

StepMotorOn

Declaração	`Procedure StepMotorOn(dir, speed, steps : Integer);`
Descrição	Liga ou desliga o motor de passos.
Parâmetros	Dir : Direção do movimento. Pode assumir os valores 0 ou 1. Speed : Velocidade. Um valor 0 nesse parametro interrompe o movimento do motor de passos e é utilizado para desligar-lo.

	Steps : Número de passos. Caso o número de passos seja 255(0xFF em hexadecimal) então o motor irá rodar continuamente até que se execute novamente a função StepMotorOn que irá sobrepor os parâmetros anteriores. Um valor 0 nesse parametro interrompe o movimento do motor de passos e é utilizado para desligar-lo.
Retorno	Nenhum.

RelayOn

Declaração	`Procedure RelayOn(relay : Integer; tf : Boolean);`
Descrição	Liga ou desliga uma chave eletromagnética (Relé).
Parâmetros	Relay : Determina qual chave deve ser ligada ou desligada. Tf : Determina se a chave deve ser ligada(true) ou desligada(false).
Retorno	Nenhum.

Métodos do Display de Cristal Líquido

Nesse grupo encontramos os métodos de controle do display de cristal líquido. Esses métodos podem ser utilizados em conjunto com os dois módulos que suportam o displa yde cristal líquido, são eles, o *Módulo de Motores e* Displays e o *Módulo de Entradas, Saídas e Servo-Motores*. A seguir uma tabela com esses métodos e uma pequena a descrição.

Método	Descrição
LCDOn	Liga ou desliga o display de cristal líquido(LCD).
LCDWriteText	Escreve um texto no dispaly LCD na posição atual do cursor.
LCDGotoXY	Posiciona o cursor do LCD.
LCDClear	Limpa a tela do LCD.
LCDCursorOn	Liga ou desliga a visualização do cursor do LCD.
LCDBacklightOn	Liga ou desliga a iluminação de fundo do LCD.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e

retorno.

LCDOn

Declaração	`Procedure LCDOn(tf : Boolean);`
Descrição	Liga ou desliga o display de cristal líquido(LCD).
Parâmetros	Tf : Determina se o display deve ser ligado(true) ou desligado(false).
Retorno	Nenhum.

LCDWriteText

Declaração	`Procedure LCDWriteText(txt : String);`
Descrição	Escreve um texto no dispaly LCD na posição atual do cursor. O cursor pode ser posicionado em qualquer linha e coluna do display com o função LCDGotoXY.
Parâmetros	Txt : String com texto que deve ser escrito.
Retorno	Nenhum.

LCDGotoXY

Declaração	`Procedure LCDGotoXY(lin, col : Integer);`
Descrição	Posiciona o cursor do LCD. O display possui 4 linhas e 20 colunas.
Parâmetros	Lin : Linha em que o cursor deve ser posicionado. Pode assumir o valor de 0 à 3. Col : Coluna em que o cursor deve ser posicionado. Pode assumir o valor de 0 à 19.
Retorno	Nenhum.

LCDClear

Declaração	`Procedure LCDClear();`
Descrição	Limpa a tela do LCD.
Parâmetros	Nenhum.
Retorno	Nenhum.

LCDCursorOn

Declaração	`Procedure LCDCursorOn(tf : Boolean);`
Descrição	Liga ou desliga a visualização do cursor do LCD.
Parâmetros	Tf : Determina se o cursor deve ser ligado(true) ou desligado(false).
Retorno	Nenhum.

LCDBacklightOn

Declaração	`Procedure LCDBacklightOn(tf : Boolean);`
Descrição	Liga ou desliga a iluminação de fundo do LCD.
Parâmetros	Tf : Determina se a iluminação de fundo do LCD deve ser ligada(true) ou desligada(false).
Retorno	Nenhum.

Métodos do Módulo de Entradas, Saídas e Servo-Motores

Fornece os métodos para controle do *Módulo de Entradas, Saídas e Servo-Motores.* A seguir a descrição de seus métodos. . A seguir uma tabela com esses métodos e uma pequena a descrição.

Método	Descrição
DigitalPortRead	Lê o valor de uma porta digital.
DigitalPortWrite	Grava um valor em uma porta digital.
ServoMotorOn	Liga um servo-motor.
ServoMotorOff	Desliga um servo motor.

A seguir uma descrição detalhada de todos os métodos com declaração, descrição, parâmetros e retorno.

DigitalPortRead

Declaração	`Function DigitalPortRead(port : Integer) : Byte;`

Descrição	Lê o valor de uma porta digital.
Parâmetros	Port : Porta que se deseja ler.
Retorno	Valor lido na porta.

DigitalPortWrite

Declaração	`Procedure DigitalPortWrite(port : Integer; value : Byte);`
Descrição	Grava um valor em uma porta digital.
Parâmetros	Port : Porta na qual deseja-se gravar o valor. Value : Valor a ser gravado.
Retorno	Nenhum.

ServoMotorOn

Declaração	`Procedure ServoMotorOn(motor : Integer; pos : Byte);`
Descrição	Liga um servo-motor. Atualmente o Módulo de Entradas, Saídas e Servo-Motores pode controlar 8 servo-motores. Desses, os quatro primeiros, numerados de 0 à 3, tem um controle preciso, possibilitando a especificação de 256 posições. Já o controle dos quatro ultimos servomotores, numerados de 4 à 7, é menos precisa e consegue controlar apenas 3 posições, que são inicio meio e fim.Para esses quatro último motores, se for passado como parâmetro de posição um valor de 0 à 85, ele permanecerá na posição de início, um valor de 86 à 171 fará com que o motor permaneça no meio do curso e um valor de 172 à 255 faz com que o motor permaneça no final do curso. Para controlar a direção do giro de servo-motores que foram modificados para giro contínuo é necessário apenas passar como parâmetro de posição o valor 0 que fará com que o motor gire em uma direção ou o valor 255 para que ele gire para a direção contrária.
Parâmetros	Motor : Motor que se deseja ligar. Pos : Posição em que o motor deve permanecer.
Retorno	Nenhum.

ServoMotorOff

Declaração	`Procedure ServoMotorOff(motor : Integer);`

Descrição	Desliga um servo motor.
Parâmetros	Motor : Motor que se deseja desligar.
Retorno	Nenhum.

Exemplo de Programa

Vamos ver rapidamente um exemplo de programa de controle desenvolvido em Delphi. Nosso exemplo será uma janela com dois botões. Um desses botões quando pressionado liga o primeiro motor DC e envia um texto ao LCD dizendo que o motor está ligado. O outro botão desliga o motor e envia um texto ao LCD dizendo que o motor está desligado. A aparência do programa será a seguinte.

Figura 1: *Aparência final do programa exemplo.*

Antes de iniciar é importante dizer que o objetivo desse manual não é ensinar Delphi, por isso não será dado muitos detalhes sobre as tarefas básicas de criação desse programa. Por isso, é importante que se tenha um conhecimento básico sobre programação Delphi ou mesmo sobre alguma outra linguagem de programação, pois, os procedimentos de criação desse programa exemplo seria bem semelhante nas outras linguagens de programação que a biblioteca tem suporte, isto é, C++ e todas as linguagens .NET.

Outra fonte interessante de informações são os tutoriais que acompanham a biblioteca de controle. Eles apresentam vários exemplos com muito mais detalhes do que esse programa que vamos dar agora. No final desse tópico, sobre programação do Kit, existe um sub-tópico que fornece mais detalhes sobre os tutoriais. Vamos ao exemplo agora.

Primeiramente precisamos criar um projeto que possa utilizar a biblioteca de controle. Em Delphi, o modo mais fácil de fazer isso é incluir os arquivos com o código fonte da biblioteca de controle no projeto do nosso programa exemplo. Para maiores detalhes de como criar o projeto veja os tutoriais que acompanham a biblioteca de controle. Existe um tutorial denominado “Base” em que é explicado detalhadamente como se deve proceder para criar um projeto que utilize a biblioteca de controle.

Temos que incluir quatro arquivo ao projeto, são eles o Robotica.pas e três arquivos que implementam a comunicação serial. Após incluir esses arquivos o gerenciador de projetos no Delphi ficaria assim.

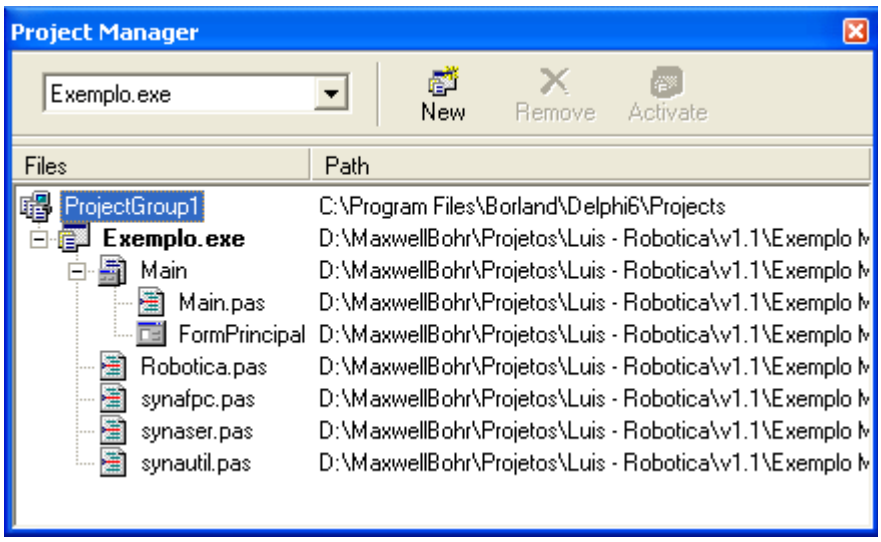


Figura 2: *Gerênciador de projetos após a adição dos arquivos da biblioteca de controle.*

Agora adicionamos, na seção **uses** do código do **Form** principal de nosso programa, uma referência a biblioteca de controle. Isso é feito adicionando "Robotica" à lista de **Units**. A seguir o código do **Form** principal nesse momento.

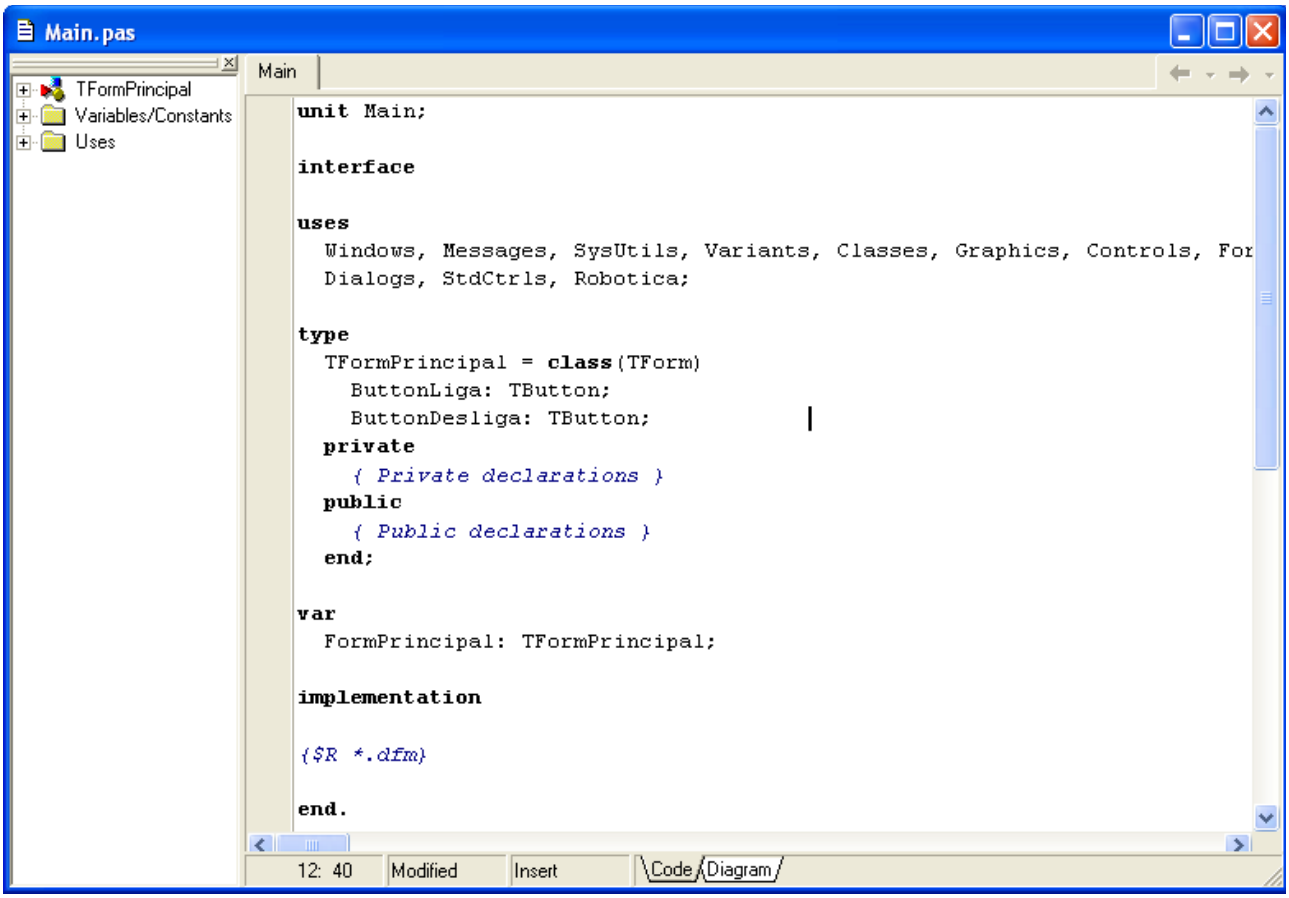


Figura 3: *Código do Form principal após inclusão da biblioteca de controle à lista de Units.*

Com nosso projeto criado e já podendo utilizar as rotinas de controle do Kit, implementaremos o código dos dois botões. A seguir o código completo do botão ligar.

```
procedure TFormMain.ButtonLigaClick(Sender: TObject);
var
  kit : TKit;
begin
  // Cria uma instância da classe TKit
  kit := TKit.Create;

  // Abre a comunicação através da porta serial 1(COM1)
  kit.OpenCommunication('COM1');

  // Envia comandos para o Kit
  kit.DCMotorOn(0, 0, 255);
  kit.LCDWriteText('Motor Ligado!');

  // Fecha a comunicação
  kit.CloseCommunication;
end;
```

Esse código é bem simples, primeiro criamos uma instância da classe TKit. Essa classe contém todos os métodos de controle do Kit, então sempre que quisermos enviar um comando para o Kit teremos que ter uma instância válida dessa classe. Em seguida abrimos a comunicação com um Kit conectado na COM1 e enviamos os comandos para ligar o primeiro motor DC com velocidade máxima e para enviar uma mensagem de texto para o display de cristal líquido. Por fim finalizamos a comunicação.

O código do botão para desligar o motor é muito semelhante. Mudamos apenas os parâmetros do método “DCMotorOn” para que o motor seja desligado e o texto que é enviado ao LCD. A seguir como fica o código.

```
procedure TFormMain.ButtonDesligaClick(Sender: TObject);
var
  kit : TKit;
begin
  // Cria uma instância da classe TKit
  kit := TKit.Create;
```

```
    // Abre a comunicação através da porta serial 1(COM1)
    kit.OpenCommunication('COM1');

    // Envia comandos para o Kit
    kit.DCMotorOn(0, 0, 0);
    kit.LCDWriteText('Motor Desligado!');

    // Fecha a comunicação
    kit.CloseCommunication;
end;
```

Com esse exemplo já é possível ter um noção de como é a programação para o Kit. Como podemos ver, com o uso da biblioteca, essa programação é muito simples, sendo necessário apenas chamar os métodos com os parâmetros desejados sem se preocupar com detalhes de comunicação ou codificação de comandos. A seguir vamos ver um pouco sobre tratamento de erros, um assunto importante para criação de programas que utilizam a biblioteca de controle, alias em programação em geral.

Tratamento de Erros

Quando algum erro ocorre em algum dos métodos da biblioteca de controle, é emitida uma exceção. A causa mais comum para esse tipo de situação é a falha na comunicação, no entanto outros erros podem ocorrer.

Para tratar as exceções emitidas é necessário utilizar os recursos específicos de tratamento de exceção da linguagem de programação utilizada. Vamos dar um exemplo em Delphi, no entanto, outras linguagens utilizam estruturas de tratamento de exceção semelhantes.

Vamos supor que tivéssemos um programa onde enviamos o texto “Teste” para o display de cristal líquido. Supondo que tivéssemos uma instância já inicializada da classe TKit chamada “kit”, a linha de código que envia o texto seria:

```
kit.LCDWriteText('Teste');
```

Se não houver erro na execução desse método teríamos o texto escrito no LCD. No entanto, se houver algum erro, provavelmente de comunicação, esse método emite uma exceção e uma caixa de diálogo com um erro seria apresentada ao usuário.

Sem um código para tratamento de exceções não temos controle sobre a mensagem que é apresentado e nem conseguimos executar algum código para tratamento do erro. Para permitir esse tipo de ação existe uma estrutura em Delphi que permite que as exceções não sejam tratadas automaticamente, mas sim pelo programador.

Essa estrutura é o **try-except**. Para utilizar essa estrutura temos que colocá-la em volta do nosso código, dessa forma qualquer exceção que ocorra no nosso código será capturada e o controle do programa será passado para uma parte do código que será responsável pelo tratamento da exceção. Abaixo o esquema de como essa estrutura é utilizada.

```
try
  ...
  Código que queremos executar
  ...
Except
  ...
  Código que é executado somente se houver uma exceção
  ...
end;
```

No nosso exemplo, com o uso da estrutura **try-except** em volta do código teríamos:

```
try
  kit.LCDWriteText('Teste');
except
  ShowMessage('Erro no envio do texto para o LCD');
end;
```

Dessa forma, se ocorrer alguma exceção durante a execução do bloco try, que no nosso caso só possui uma chamada ao método de escrita no LCD, a execução do programa é desviada para o bloco except onde devemos colocar o código para o tratamento da exceção. No exemplo acima colocamos o código para a apresentação de uma mensagem de erro customizada quando houver uma exceção.

Em muitos casos os erro podem ser ignorados, como por exemplo quando estamos verificando se algum Kit esta respondendo em uma porta serial, mas não queremos que o programa seja interrompido se não houver um Kit respondendo. Nesse caso podemos deixar o bloco except vazio. Dessa forma a exceção será ignorada. Nosso código ficaria da seguinte forma se quiséssemos ignorar as exceções geradas por ele.

```
try
  kit.LCDWriteText('Teste');
except
end;
```

Dessa forma nenhum erro no envio do comando seria informado. Para maiores informações sobre exceções procure na documentação da linguagem de programação que está sendo utilizada. Abaixo alguns links onde mais informações sobre exceções podem ser encontradas.

Delphi:

http://www.delphibasics.co.uk/Article.asp?Name=Exceptions

C++:

http://cplus.about.com/od/beginnerctutorial/l/aa122202a.htm

C#:

http://www.codersource.net/csharp_tutorial_exceptions.html

Tutoriais

Para auxiliar o aprendizado da programação para a criação de programas de controle para o Kit Didático de Robótica foram criados vários tutoriais onde são explicados, passo a passo, a criação de vários programas exemplo. O conjunto de todos esses tutoriais e programas exemplos tem o intuito de cobrir todos os detalhes da biblioteca de controle de forma que através dos tutoriais seja possível aprender como desenvolver programas que possam aproveitar ao máximo as funcionalidades que o Kit oferece. Os tópicos a seguir apresentam uma lista com o nome dos tutoriais e os assuntos abordados por eles. Esses tópicos estão divididos por grau de complexidade dos tutoriais.

Básicos

Esses tutoriais abordam apenas tópicos isolados. Utilizam o computador e o Kit com apenas o *Módulo de Sensores* ou o *Módulo de Motores e Displays*, não sendo necessária nenhuma montagem eletrônica ou mecânica. Veja a lista desses tutoriais a seguir.

Nome do Tutorial	***Descrição***
Base	Demonstra como criar um projeto base que possa utilizar a biblioteca de controle do Kit. Esse projeto fornece as estruturas básicas necessárias para um programa de controle. Esse projeto poderá ser utilizado como base para a criação da maioria dos outros programas de controle.
Leitura de Sensores	Descreve as formas básicas de leitura de sensores.
Display de Cristal Líquido	Explica como utilizar as funcionalidades do display de cristal líquido.
Teclado	Mostra como se faz para utilizar o teclado.

Nome do Tutorial	*Descrição*
Barra de LEDs	Esse tutorial explica como interagir com as barras de LEDs.
Display de 7 Segmentos	Descreve como se faz para utilizar os displays de 7 segmentos.
Buzzer	Mostra como controlar o buzzer.
Motor DC	Ensina a controlar os motores DC.
Motor de Passo	O objetivo desse tutorial é apresentar como se faz para utilizar o motor de passo.
Relé	Explica como controlar os relés.
Memória Flash	Ensina a utilizar a memória flash.
Informações sobre o Kit	Mostra como obter informações sobre o Kit.

Intermediários

Esses tutoriais consideram que os assuntos tratados nos tutoriais básicos já são conhecidos. Por isso eles tratam de mais de um tópico básico em conjunto ou um tópico isolado de maior complexidade. Utilizam o computador e o Kit com todos os seus módulos, isto é, o *Módulo de Sensores,* o *Módulo de Sensores Genérico,* o *Módulo de Motores e Displays* e o *Módulo de Entradas, Saídas e Servo-Motores.* Em alguns exemplos são necessárias montagens eletrônicas. A seguir uma lista com esses tutoriais.

Nome do Tutorial	*Descrição*
IrDA	Explica como utilizar a porta IrDA para o envio de comandos remotamente.
Integração entre os componentes	Demonstra como os vários componentes do Kit podem ser integrados em projetos mais complexos.
Leitura Bufferizada de Sensores	Mostra como fazer leitura de sensores utilizando o buffer interno do Kit.
Entradas Genéricas de Sensores	Explica o que deve ser feito para utilizar as portas genéricas de sensores
Entradas Digitais	Ensina a utilizar as portas digitais de entrada.
Saídas Digitais	Ensina a utilizar as portas digitais de saída.
Servo-Motores	Mostra como utilizar servo-motores
Uso de Threads	Demonstra como utilizar threads nos programas de controle.

Avançados

Os tutoriais avançados abordam vários tópicos em conjunto. Esses tópicos já foram tratados nos tutoriais básicos e intermediários e esse conhecimento é utilizado nesses tutoriais para a construção de projetos completos com software de controle, motangem mecânica e montagem eletrônica. São eles:

Nome do Tutorial	*Descrição*
Plotter	Explica como fazer um programa de controle para um plotter.
Bípede	Descreve a criação de um controle para um bípede.
Inseto	Criação do programa de controle de um inseto.
Separador de Grãos	Mostra como fazer o controle de um separador de grãos

Métodos da Biblioteca de Controle

O conjunto total de tutoriais aborda todos os métodos oferecidos pela biblioteca de controle. A seguir, uma tabela com a indicação de qual tutorial apresenta um determinado método da biblioteca de controle. Como podemos observar, todos os métodos da biblioteca são abordados em algum ponto.

Nível	*Nome do Tutorial*	*Métodos Apresentados*
Básico	Base	• OpenCommunication • CloseCommunication • IsConnected • GetInstalledSerialPorts
Básico	Leitura de Sensores	• SensorReadNow • SensorReadAll
Básico	Display de Cristal Líquido	• LCDOn • LCDWriteText • LCDGotoXY • LCDClear • LCDCursorOn • LCDBacklightOn
Básico	Teclado	• KeyboardReadNow • KeyBoardReadBuffer • KeyboardClearBuffer
Básico	Barra de LEDs	• LEDBarSetLevel • LEDBarSetStatus
Básico	Display de 7 Segmentos	• Disp7SegSetNum • Disp7SegSetStatus
Básico	Buzzer	• AudioOn
Básico	Motor DC	• DCMotorOn
Básico	Motor de Passo	• StepMotorOn
Básico	Relé	• RelayOn
Básico	Memória Flash	• FlashWrite • FlashRead • FlashErase

Básico	Informações sobre o Kit	• DeviceName • DeviceStatus
Intermediário	IrDA	• IrdaOn
Intermediário	Integração entre os componentes	-
Intermediário	Leitura Bufferizada de Sensores	• SensorBufferOn • SensorReadBuffer • SensorClearBuffer
Intermediário	Entradas Genéricas de Sensores	-
Intermediário	Entradas Digitais	• DigitalPortRead
Intermediário	Saídas Digitais	• DigitalPortWrite
Intermediário	Servo-Motores	• ServoMotorOn • ServoMotorOff
Intermediário	Uso de Threads	-
Avançado	Plotter	-
Avançado	Bípede	-
Avançado	Inseto	-
Avançado	Separador de Grãos	-

Estrutura de diretorios

Os tutoriais estão organizados em uma estrutura de pastas que os divide por nível de dificuldade. Dentro de cada pasta de um tutorial são armazenados todos os arquivos relacionados ao tutorial. Na maioria dessas pastas são armazenados um arquivo com o texto do tutorial e uma sub-pasta com o código fonte do projeto criado no tutorial. A seguir a estrutura de diretórios que armazena os tutoriais.

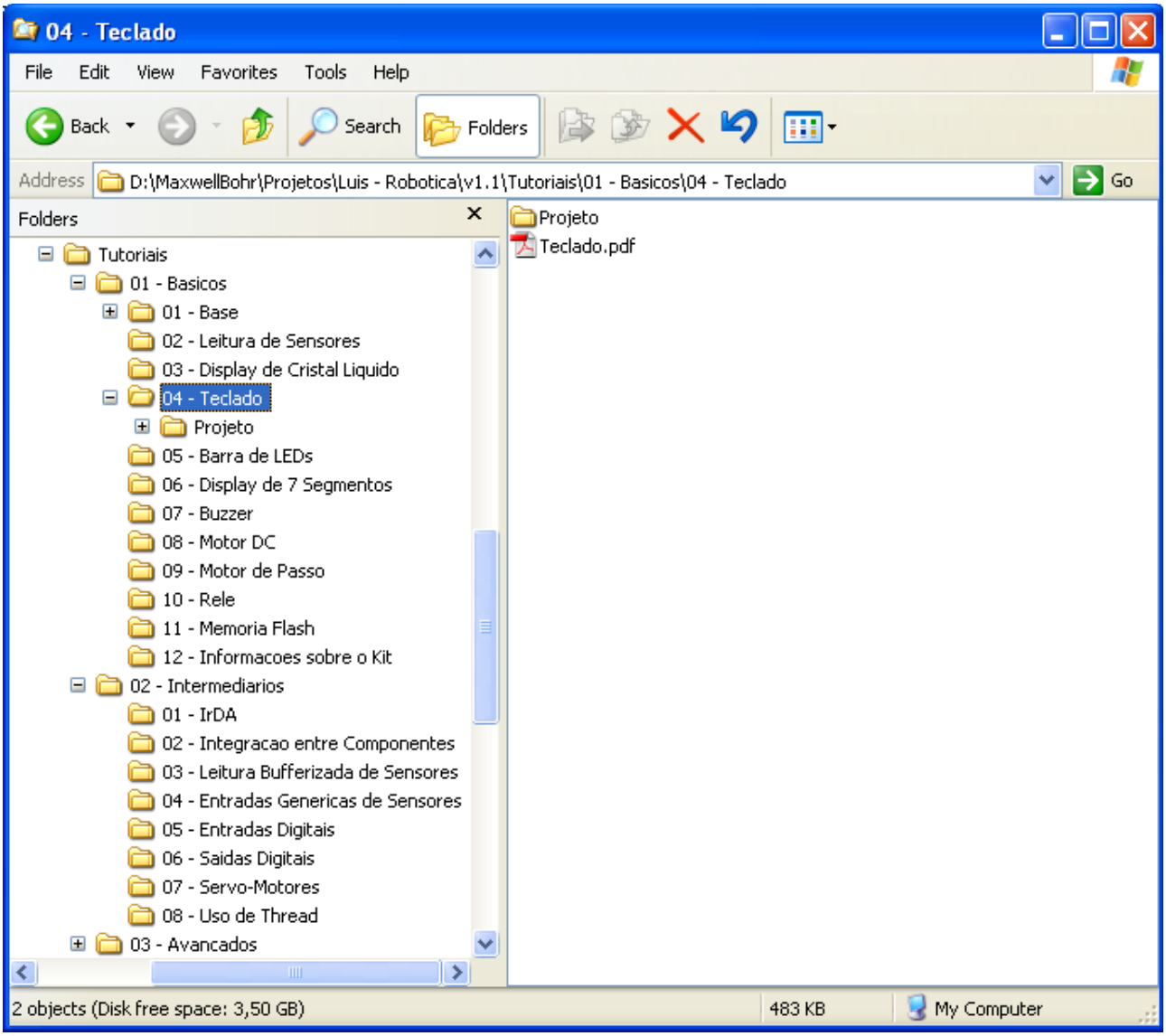


Figura 4: *Estrutura de diretórios para armazenar os tutoriais. Do lado direito o diretório do tutorial sobre teclado com o documento de texto e uma pasta com o código fonte do projeto criado.*

Comunicação Direta sem o Uso da Biblioteca de Controle

Esse tópico descreve como os comandos utilizados para controlar o Kit Didático são enviados para execução sem que se faça uso da biblioteca de controle. É descrito como gerar as seqüências de bytes que codificam os comandos e que devem ser enviadas ao Kit Didático para execução. São explicadas também as configurações que devem ser feitas para possibilitar a comunicação entre um Kit Didático e um programa de computador.

Envio de Comandos

Todas as operações no Kit Didático são realizadas a partir do envio de comandos através da porta serial de um computador. Esses comandos são formados por uma seqüência de bytes que determinam que operação o Kit Didático deverá realiza. Essa seqüência de bytes segue um formato padrão, que é apresentado a seguir:

BEGIN_CMD -> Código do comando -> [*parâmetros*]

Primeiramente é enviado um byte indicando o início do envio de um novo comando. Esse byte é denominado *BEGIN_CMD*. Em seguida deve ser enviado um byte com o código do comando que será executado. Logo após o código do comando são enviados os parâmetros. Os parâmetros variam dependendo do comando sendo executado, podendo até mesmo não haver parâmetros.

Uma vez enviada essa seqüência o Kit Didático irá processar o comando e realizar a tarefa requisitada. Se houver algum tipo de retorno esse será enviado de volta para o computador pela porta serial. Ao final de tudo o Kit Didático envia um byte indicando o término da execução do comando. Esse byte é denominado *END_CMD*. A seguir a seqüência completa da execução de um comando.

BEGIN_CMD -> Código do comando -> [*parâmetros*] <- [*retorno*] <- END_CMD

Os bytes denominados BEGIN_CMD e END_CMD são constantes e possuem sempre o mesmo valor que não depende do comando enviado e servem apenas para indicar o início e o término do envio de um comando. Abaixo a definição do valor dessas constantes:

Constante	*Valor Decimal*	*Valor Hexadecimal*
BEGIN_CMD	240	0xF0
END_CMD	241	0xF1

Todo comando possui um código único que o identifica. Esse código é formado por 8 bits, ou seja, um byte. Esse byte é enviado após o byte *BEGIN_CMD*, permitindo que o Kit Didático determine que comando deve ser executado. A seguir uma lista com os comandos reconhecidos pelo Kit Didático, seus códigos na forma decimal e hexadecimal e uma pequena descrição da função de cada um.

Comando	*Código(Dec/Hex)*		*Descrição*
• *CMD_NOP*	*00*	*0x00*	*Nenhuma operação*
• *CMD_DEVICE_NAME*	*01*	*0x01*	*Retorna uma descrição do Kit Didático*
• *CMD_DEVICE_STATUS*	*02*	*0x02*	*Retorna o estado atual do Kit*

• *CMD_IRDA_ON*	*03*	*0x03*	*Lig/Des envio de comandos pela porta IrDA Didático*
• *CMD_FLASH_WRITE*	*04*	*0x04*	*Gravar dados na memória Flash*
• *CMD_FLASH_READ*	*05*	*0x05*	*Ler dados na memória Flash*
• *CMD_FLASH_ERASE*	*06*	*0x06*	*Apagar dados na memória Flash*
• *CMD_SENSOR_READ_NOW*	*16*	*0x10*	*Ler um sensor imediatamente*
• *CMD_SENSOR_READ_ALL*	*17*	*0x11*	*Ler todos os sensor imediatamente*
• *CMD_SENSOR_BUFFER_ON*	*18*	*0x12*	*Lig/Des leitura de sensores buferizada*
• *CMD_SENSOR_READ_BUFFER*	*19*	*0x13*	*Ler buffer de leituras de sensor*
• *CMD_SENSOR_CLEAR_BUFFER*	*20*	*0x14*	*Apagar buffer de leituras de sensor*
• *CMD_LCD_ON*	*32*	*0x20*	*Lig/Des o LCD*
• *CMD_LCD_WRITE_TEXT*	*33*	*0x21*	*Escrever texto no LCD*
• *CMD_LCD_GOTOXY*	*34*	*0x22*	*Posicionar cursor do LCD*
• *CMD_LCD_CLEAR*	*35*	*0x23*	*Limpar tela do LCD*
• *CMD_LCD_CURSOR_ON*	*36*	*0x24*	*Lig/Des cursor do LCD*
• *CMD_LCD_BACKLIGHT_ON*	*37*	*0x25*	*Lig/Des luz de fundo do LCD*
• *CMD_KEYBOARD_READ_NOW*	*48*	*0x30*	*Ler tecla pressionada no teclado*
• *CMD_KEYBOARD_READ_BUFFER*	*49*	*0x31*	*Ler tecla do buffer do teclado*
• *CMD_KEYBOARD_CLEAR_BUFFER*	*50*	*0x32*	*Limpar buffer do teclado*
• *CMD_LEDBAR_SET_LEVEL*	*51*	*0x33*	*Definir nível de barra de LEDs*
• *CMD_LEDBAR_SET_STATUS*	*52*	*0x34*	*Definir estado de barra de LEDs*
• *CMD_DISP_7SEG_SET_NUM*	*53*	*0x35*	*Definir número de display de 7seg.*
• *CMD_DISP_7SEG_SET_STATUS*	*54*	*0x36*	*Definir estado de display de 7seg.*
• *CMD_AUDIO_ON*	*55*	*0x37*	*Lig/Des som (Buzzer)*
• *CMD_DCMOTOR_ON*	*56*	*0x38*	*Lig/Des motor DC*
• *CMD_STEP_MOTOR_ON*	*57*	*0x39*	*Lig/Des motor de passo*
• *CMD_RELAY_ON*	*58*	*0x3A*	*Lig/Des Chave Eletromagnética (Relé)*
• *CMD_DIGITAL_PORT_READ*	*64*	*0x40*	*Ler porta digital*
• *CMD_DIGITAL_PORT_WRITE*	*65*	*0x41*	*Gravar em porta digital*
• *CMD_SERVO_MOTOR_ON*	*66*	*0x42*	*Ligar Servo Motor*
• *CMD_SERVO_MOTOR_OFF*	*67*	*0x43*	*Desligar Servo Motor*

Atualmente os comandos estão divididos em cinco grupos. Os quatro primeiro bits que formam o código de um comando indicam o grupo a que ele pertence. Podemos identificar facilmente o grupo de um comando através da representação em hexadecimal de seu código. Na representação hexadecimal cada dígito do valor representa quatro bits, logo o primeiro dígito representa o grupo. Por exemplo, o comando *CMD_SENSOR_READ_NOW* com código em hexadecimal 0x10 pertence ao grupo de número 1, o comando *CMD_RELAY_ON* com código 0x3A pertence ao grupo de número 3 e o comando *CMD_SERVO_MOTOR_ON* com código 0x42 pertence ao grupo de número 4. A seguir uma pequena descrição dos cinco grupos de comando:

Grupo	***Descrição***
0	Comandos que dependem apenas do *Módulo Principal*. Não é necessário a conexão de nenhum outro módulo para o funcionamento desses comandos.
1	Comandos de manipulação de sensores. A execução desses comandos depende da

Grupo	*Descrição*
	conexão do *Módulo de Sensores* ou do *Módulo de Sensores Genérico* para serem executados.
2	Comandos de controle do display de cristal líquido(LCD). Dependem da conexão do *Módulo de Motores e Displays* ou do *Módulo de Entradas, Saídas e Servo-Motores* para serem executados.
3	Comandos de controle do *Módulo de Motores e Displays*. Dependem da conexão desse módulo para funcionarem.
4	Comandos de controle do *Módulo de Entradas, Saídas e Servo-Motores*. Dependem da conexão desse módulo para funcionarem.

Após o envio do código do comando, se existirem, são enviados os parâmetros. Seu significado e formato depende do comando sendo enviado. Por exemplo, o comando `CMD_LCD_BACKLIGHT_ON`, que liga ou desliga a luz de fundo do Display de Cristal Líquido(LCD), possui um parâmetro que indica se a luz de fundo deve ser ligada ou desligada. Esse parâmetro é formado por um único byte que se tiver o valor 0 indica que a luz de fundo deve ser desligada e se for 1 que ela deve ser ligada. Um outro exemplo é o comando `CMD_LCD_WRITE_TEXT`, que escreve um texto no LCD. Esse comando tem como parâmetro o texto que será escrito no display. No entanto, como esse texto possui um tamanho variável, temos que indicar para o Kit Didático qual é o tamanho do texto. Isso é feito enviando primeiramente dois bytes que indicam esse tamanho. Esses dois bytes formam um valor de 16 bits que indica quantos bytes serão enviados em seguida, no caso o tamanho do texto. O envio de todo parâmetro que possua um tamanho variável terá essa forma, em que dois bytes são enviado primeiramente indicando o tamanho do parâmetro e somente após esses bytes o parâmetro propriamente dito é enviado.

Além de parâmetros, os comandos podem ter algum tipo de retorno de resultado. Assim como acontece com os parâmetros, o significado do retorno irá depender do comando executado. Por exemplo, o comando `CMD_SENSOR_READ_NOW`, que lê um sensor tem como retorno dois bytes, que formam um valor de 16 bits, que é a leitura do sensor.

O retorno pode ter um tamanho variável e do mesmo modo que ocorre com os parâmetros, esse tamanho deve ser indicado de alguma forma para que o programa no computador saiba quantos bytes ele deve ler como retorno. Essa indicação é feita da mesma maneira que foi feita com os parametros. Para indicar o número de bytes que devem ser lidos como retorno o Kit Didático envia dois bytes, formando um valor de 16 bits, que especifica o número de bytes que serão enviado em seguida.

É importante esclarecer a forma como devem ser enviados os valores de 16 bits ou 32 bits. Esses valores devem ser enviados iniciando-se do byte mais significativo para o menos. O Kit Didático também utiliza essa mesma lógica no envio dos retornos.

Por exemplo, o comando `CMD_FLASH_READ`, utilizado para ler valores da memória não volátil do Kit Didático, possui como parâmetro um valor de 16 bits que indica o número de bytes que devem ser lidos da memória. Vamos dizer que queremos ler 100 bytes da memória do Kit Didático. Para isso enviaríamos a seguinte seqüência de bytes pela porta seria:

```
0xF0 -> 0x05 -> 0x00 -> 0x64
```

Como podemos ver, enviamos primeiramente o byte `BEGIN_CMD` seguido do código do comando `CMD_FLASH_READ` e por fim o parâmetro de 16 bits com valor 100, ou em hexadecimal 0x0064. O envio do parâmetro de 16 bits é feito na ordem do byte mais significativo, no caso 0x00, para o menos significativo, ou seja 0x64 no nosso exemplo.

Podemos continuar utilizando o mesmo exemplo para mostrar como o Kit Didático utiliza a mesma lógica de envio em seus retornos. No comando, *CMD_FLASH_READ* como o número de bytes que se deseja ler da memória varia, o tamanho do retorno também é variável. Logo o retorno desse comando segue o formato dos retornos variáveis, em que, antes do retorno propriamente dito, é enviado um valor de 16 bits indicando o tamanho do retorno. Os bytes de retorno no nosso exemplo seria o seguinte:

```
<- 0x00 <- 0x64 <- 100 bytes lidos na memória
```

Os dois primeiros bytes, 0x00 e 0x64, indicam o tamanho do retorno, que é 100 bytes. No comando *CMD_FLASH_READ,* o tamanho do retorno é igual ao número de bytes que se requisitou que fossem lidos. No nosso exemplo requisitamos a leitura de 100 bytes, como retorno temos 0x00, que é o byte mais significativo do tamanho do retorno, seguido do byte 0x64 que é o byte menos significativo do tamanho do retorno. Com issu formamos o valor em hexadecimal 0x0064, ou 100 em decimal. Após esse valor são enviados os 100 bytes lidos da memória, que é o retorno propriamente dito.

Outra informação importante é que a contagem dos valores de parâmetros e retornos iniciam a partir de 0 e não 1. Por exemplo, no comando *CMD_LCD_GOTOXY*, utilizado para posicionar o cursor do display de cristal líquido, existem dois parâmetros, um com a linha e outro com a coluna onde o cursor será posicionado. Para posicionar o cursor na primeira linha temos que passar como parâmetro de linha o valor 0 e não 1. O mesmo acontece para posicionar o cursor na primeira coluna. Nesse caso o valor do parâmetro de coluna deveria ser 0.

Outro exemplo é o comando *CMD_DCMOTOR_ON*, utilizado para controlar os motores DC. Esse comando possui um parâmetro que indica qual motor DC queremos controlar. Para controlar o primeiro motor DC temos que passa como parâmetro o valor 0 e para controlar o segundo o valor 1.

Configuração da Comunicação

A comunicação entre o Kit Didático e o computador é feita por uma porta serial. Um ponto importante é a configuração dos parâmetros dessa comunicação serial. Os dois lados da comunicação devem utilizar as mesmas configurações. O valor dos parâmetros de configuração utilizados no Kit Didático são listados abaixo e para que um programa inicie uma comunicação com o Kit Didático ele deverá configurar a porta serial do computador com esses mesmos valores.

Taxa de transmissão:	38400 Bauds
Número de bits de dados:	8 bits
Paridade:	Sem Paridade
Número de bits de stop:	1 StopBit

Os detalhes para que um programa configure uma porta serial e possa enviar e receber bytes através dela depende principalmente do sistema operacional e da linguagem de programação utilizados. A seguir algum sites com informações para programação de porta serial em Windows e Linux e para as linguagens de programação C++, Delphi em ambiente da Borland e para a plataforma .NET.

Comunicação serial no Win32:

http://msdn.microsoft.com/library/en-us/dnfiles/html/msdn_serial.asp

Comunicação serial no Linux:
http://www.tldp.org/HOWTO/Serial-Programming-HOWTO

Comunicação serial para Delphi, Kylix e C++ Builder:
http://synapse.ararat.cz
http://sourceforge.net/projects/comport
http://sourceforge.net/projects/tpapro
http://www.deepsoftware.ru/nrcomm (Componente Comercial)

Programação serial para .NET:
http://www.codeproject.com/dotnet/DotNetComPorts.asp
http://msdn.microsoft.com/msdnmag/issues/02/10/NETSerialComm/default.aspx

Programação serial para .NET 2.0:
http://msdn2.microsoft.com/en-us/library/30swa673(en-us,vs.80).aspx
http://msmvps.com/coad/archive/2005/03/23/39466.aspx

Descrição Detalhada dos Comandos

A seguir todos os comandos do Kit Didático v1.1.0 serão vistos em detalhes. São apresentadas todas as informações, específicas de cada comando, necessárias para a formação da sua seqüência de bytes enviada ao Kit Didático. Essas informações podem ser resumidas em código do comando, descrição do comando e descrição da forma de envio, tamanho e significado dos seus parâmetros e retorno. Ao final é apresentado um quadro de resumo que pode ser utilizado para referência rápida.

CMD_NOP: Nenhuma Operação

Código: 0x00

Descrição: Não realiza nenhuma operação. É utilizado para verificar a presença de algum Kit Didático ligado à uma porta serial.

Parâmetros: Nenhum.

Retorno: Nenhum.

CMD_DEVICE_NAME: Retorna uma descrição do Kit Didático

Código: 0x01

Descrição: Retorna uma "string" com o nome do Kit Didático.

Parâmetros: Nenhum.

Retorno: Retorno de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de retorno que deverão ser recebidos. Os bytes que vem em seguida são caracteres ASCII e formam a "string" com o nome.

CMD_DEVICE_STATUS: Retorna o estado atual do Kit Didático

Código: 0x02

Descrição: Retorna a estrutura de dados que armazena o estado do Kit Didático no momento. Essa estrutura é formada pelos seguintes campos:

Campo	*Tamanho*	*Descrição*
ID	2 bytes	ID utilizado para identificar o modelo do Kit Didático.
FIRMWARE_VERSION	4 bytes	Versão do programa de controle(Firmware) no *Módulo Principal*. Primeiro byte indica versão, segundo byte

Campo	*Tamanho*	*Descrição*
		sub-versao e os dois últimos revisão.
IRDA	1 byte	Indica se possui IrDA. Um valor 0 indica que não possui porta de comunicação IrDA e um valor 1 indica que possui.
IRDA_ON	1 byte	Indica se o modo de envio de comandos pela porta IrDA está ligado. Um valor 0 indica que esse modo esta desligado e 1 indica que está ligado.
BOARD	1 byte	Indica o módulo que está conectado ao barramento de 40 vias. Quando não há modulos conectados assume o valor 0, se encontrar o *Módulo de Motores e Displays* assume o valor 1 e se encontrar o *Módulo de Entradas, Saídas e Servo-Motores* assume o valor 2.
REGISTERS	6 bytes	Valor dos registradores internos de controle.
CMD_ERROR	2 bytes	Numero de comandos que falharam.
CMD_SUCCESS	2 bytes	Numero de comandos executados com sucesso.
CLOCK	4 bytes	Clock de operação do módulo de Kit Didático.
RESERVED	-	Os bytes a partir desse ponto são reservados ou de uso interno.

Parâmetros: Nenhum.

Retorno: Retorno de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de retorno que deverão ser recebidos. Os bytes que vem em seguida são a estrutura com o estado atual do Kit Didático e segue o formato apresentado na descrição desse comando.

CMD_IRDA_ON: Lig/Des envio de comandos pela porta IrDA

Código: 0x03

Descrição: Liga ou desliga o modo de envio de comandos através da porta de comunicação IrDA. Quando esse modo está ativado todos os comandos enviados serão redirecionados para a porta IrDA para execução remota por algum outro Kit Didático que receba esse comando. A porta IrDA tem um alcance de aproximadamente 1 metro e para que seja possível a comunicação entre dois Kits Didático suas portas IrDA devem estar direcionadas uma para a outra e sem nenhum objeto interrompendo a visada entre elas.

Parâmetros: Um valor 0 para desligar o modo de envio de comandos através da porta de comunicação IrDA e um valor 1 para ligar.

Retorno: Nenhum.

CMD_FLASH_WRITE: Gravar dados na memória Flash

Código: 0x04

Descrição: Grava dados na memória não volátil do Kit Didático. Esses dados poderão ser recuperados mesmo após o Kit Didático ser desligado. Antes de gravar dados na memória, ela deve ser apagada com o comando `CMD_FLASH_ERASE`, caso contrário a gravação poderá não funcionar corretamente. O número máximo de bytes que podem ser gravados é 100.

Parâmetros: Parâmetro de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de parâmetro que serão enviados em seguida. Atualmente esse valor pode ser no máximo 100. Após o envio do tamanho do parâmetro são enviados os bytes dos dados que serão gravados.

Retorno: Nenhum.

CMD_FLASH_READ: Ler dados na memória Flash

Código: 0x05

Descrição: Lê dados gravados na memória não volátil do Kit Didático. O número máximo de bytes que podem ser lidos é 100.

Parâmetros: O número de bytes que deverão ser lidos. O primeiro byte deverá ser a parte mais significativa seguido de outro com a parte menos significativa formando um valor de 16 bits que indica o número total de bytes que se deseja ler. Atualmente esse valor pode ser no máximo 100.

Retorno: Retorno de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de retorno que deverão ser recebidos. Logo depois são enviados os bytes de dados lidos.

CMD_FLASH_ERASE: Apagar dados na memória Flash

Código: 0x06

Descrição: Apaga todo o conteúdo da memória não volátil do Kit Didático.

Parâmetros: Nenhum.

Retorno: Nenhum.

CMD_SENSOR_READ_NOW: Ler um sensor imediatamente

Código: 0x10

Descrição: Lê um canal de sensor. Existem 16 canais para sensores que podem ser lidos com esse comando. A seguir uma lista com o número dos canais, os sensores conectados a cada canal no *Módulo de Sensores* e o conector onde se encontra o canal no *Módulo de Sensores Genérico*.

Canal	*Módulo de Sensores*	*Módulo de Sensores Genérico*
00	Potenciômetro II	Conector 1: Primeira entrada analógica
01	Microfone	Conector 1: Segunda entrada analógica
02	Vibração	Conector 1: Terceira entrada analógica
03	Chave Magnética	Conector 1: Entrada digital
04	Temperatura	Conector 2: Primeira entrada analógica
05	Luminosidade	Conector 2: Segunda entrada analógica
06	Potenciômetro I	Conector 2: Terceira entrada analógica
07	Não utilizada	Conector 2: Entrada digital
08	Auxiliar I	Conector 3: Primeira entrada analógica
09	5V	Conector 3: Segunda entrada analógica
10	12V	Conector 3: Terceira entrada analógica
11	Par Óptico I	Conector 3: Entrada digital
12	Peso	Conector 4: Primeira entrada analógica
13	Distância	Conector 4: Segunda entrada analógica
14	Auxiliar II	Conector 4: Terceira entrada analógica
15	Par Óptico II	Conector 4: Entrada digital

Parâmetros: Um byte indicando canal que deve ser lido.

Retorno: A leitura do canal, que é um valor de 16 bits. O primeiro byte com a parte mais significativa do valor e o segundo com a parte menos significativa.

CMD_SENSOR_READ_ALL: Ler todos os sensores imediatamente

Código: 0x11

Descrição: Lê todos os canais de sensores em uma única operação. Os canais são lidos na ordem que aparecem na tabela encontrada na descrição do comando `CMD_SENSOR_READ_NOW`.

Parâmetros: Nenhum.

Retorno: Retorno de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de

bytes de retorno que deverão ser recebidos. Em seguida um conjunto de valores de 16 bits que são as leituras dos canais. O primeiro byte de cada um desses valores é partes mais significativa da leitura do canal, e o segundo a parte menos significativa.

CMD_SENSOR_BUFFER_ON: Lig/Des leitura de sensores buferizada

Código: 0x12

Descrição: Liga ou desliga a aquisição bufferizada. Quando este modo de aquisição está ligado, o Kit Didático faz leituras de um canal de sensor em intervalos determinados e guarda a leitura em um buffer. Esse buffer pode conter até 90 leituras e quando cheio as leituras mais novas vão sobrescrevendo as mais antigas. O intervalo entre as aquisições é dado em milisegundos e são utilizados 4 bytes para indicar esse valor. Isso permite que o intervalo entre as aquisições possa ser de 1ms ou até mesmo de várias horas. Nesse modo, a aquisição não depende do computador, ela continuará sendo feita mesmo que o Kit Didático seja desconectado da porta serial. Dessa forma se for utilizado um intervalo de, por exemplo, 1 minuto é possível fazer um monitoramento de algum parâmetro sem a presença de um computador por 1 hora e meia, se esse intervalo for de 10 minutos podemos ter um monitoramento por 15 horas. O Kit Didático suporta a leitura de apenas um sensor por vez nesse modo.

Parâmetros: Um byte com valor 1 para indicar que o modo de aquisição buferizada deve ser ligado ou com valor 0 para ser desligado. Em seguida um byte indicando o canal que será lido. Por fim os 4 bytes que determinam o intervalo entre as aquisições. Esses bytes devem ser enviados do mais para o menos significativo.

Retorno: Nenhum

CMD_SENSOR_READ_BUFFER: Ler buffer de leituras de sensor

Código: 0x13

Descrição: Lê todos os valores de leituras buferizadas de sensores.

Parâmetros: Nenhum.

Retorno: Retorno de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de retorno que deverão ser recebidos. Em seguida um conjunto de valores de 16 bits que são as leituras do canal buferizado. O primeiro byte de cada um desses valores é partes mais significativa da leitura do canal, e o segundo a parte menos significativa.

CMD_SENSOR_CLEAR_BUFFER: Apagar buffer de leituras de sensor

Código: 0x14

Descrição: Limpa o buffer de leituras buferizadas.

Parâmetros: Nenhum.

Retorno: Nenhum.

CMD_LCD_ON: Lig/Des o LCD

Código: 0x20

Descrição: Liga ou desliga o display de cristal líquido (LCD).

Parâmetros: Um byte indicando se o LCD deve ser ligado ou desligado. Um valor 0 indica que ele deve ser desligado e 1 que deve ser ligado.

Retorno: Nenhum.

CMD_LCD_WRITE_TEXT: Escrever texto no LCD

Código: 0x21

Descrição: Escreve um texto no dispaly LCD na posição atual do cursor. O cursor pode ser posicionado em qualquer linha e coluna do display com o comando `CMD_LCD_GOTOXY`.

Parâmetros: Parâmetro de tamanho variável. O primeiro byte enviado é a parte mais significativa seguido de outro que é a parte menos significativa de um valor de 16 bits que indica o número de bytes de parâmetro que serão enviados em seguida. Após o envio do número de bytes que se deseja enviar de texto são enviados os bytes ASCII dos caracteres do texto.

Retorno: Nenhum.

CMD_LCD_GOTOXY: Posicionar cursor do LCD

Código: 0x22

Descrição: Posiciona o cursor do LCD. O display possui 4 linhas e 20 colunas.

Parâmetros: Um byte com a linha com um valor de 0 à 3 e um byte com a coluna com um valor de 0 à 19.

Retorno: Nenhum.

CMD_LCD_CLEAN: Limpar tela do LCD

Código: 0x23

Descrição: Limpa a tela do LCD.

Parâmetros: Nenhum.

Retorno: Nenhum.

CMD_LCD_CURSOR_ON: Lig/Des cursor do LCD

Código: 0x24

Descrição: Liga ou desliga a visualização do cursor do LCD.

Parâmetros: Um byte indicando se o cursor do LCD deve ser ligado ou desligado. Um valor 0 indica que esse deve ser desligado e 1 que deve ser ligado.

Retorno: Nenhum.

CMD_LCD_BACKLIGHT_ON: Lig/Des luz de fundo do LCD

Código: 0x25

Descrição: Liga ou desliga a iluminação de fundo do LCD.

Parâmetros: Um byte indicando se a iluminação de fundo do LCD deve ser ligada ou desligada. Um valor 0 indica que essa deve ser desligada e 1 que deve ser ligada.

Retorno: Nenhum.

CMD_KEYBOARD_READ_NOW: Ler tecla pressionada no teclado

Código: 0x30

Descrição: Lê que tecla do teclado está pressionada no momento.

Parâmetros: Nenhum.

Retorno: Tecla pressionada. Essa tecla é representada por um byte com valor que pode variar de 0 à 15. Se não houver nenhuma tecla pressionada retorna 0xFF.

CMD_KEYBOARD_READ_BUFFER: Ler tecla do buffer do teclado

Código: 0x31

Descrição: Lê o valor mais antigo armazenado no buffer do teclado. O buffer possui 10 posições.

Parâmetros: Nenhum.

Retorno: Valor lido do buffer do teclado. Esse valor é um byte com valor que pode variar de 0 à 15. Caso o buffer esteja vazio retorna 0xFF.

CMD_KEYBOARD_CLEAN_BUFFER: Limpar buffer do teclado

Código: 0x32

Descrição: Limpa o buffer do teclado.

Parâmetros: Nenhum.

Retorno: Nenhum.

CMD_LEDBAR_SET_LEVEL: Definir nível de barra de LEDs

Código: 0x33

Descrição: Define o nível de uma barra de LEDs.

Parâmetros: Um byte indicando a barra que será definido o nível seguido de um byte com o nível. Esse valor pode estar no intervalo de 0 (Todos os LEDs apagados) à 8 (Todos os LEDs ligados).

Retorno: Nenhum.

CMD_LEDBAR_SET_STATUS: Definir estado de barra de LEDs

Código: 0x34

Descrição: Define o estado de uma barra de LEDs. Com esse comando é possível ter controle total sobre o estado de todos os LEDs da barra.

Parâmetros: Um byte indicando a barra que se deseja definir o estado seguido de um byte que indica o estados dos 8 LEDs da barra. Cada bit desse byte indica o estado de um LED, sendo que um bit 0 indica que o LED esta desligado e um bit 1 indica o LED ligado. Ex: Um byte com valor 0, isto é, 00000000b em binário, indica que todos os LEDs estão apagados, já um byte com valor 255, que em binário é representado por 11111111b, indica que todos os LEDs estão ligados. Um valor 85, que em binário é representado por 01010101b, indica que os LEDs da barra alternam entre um ligado e o seguinte desligado. Um valor 15, em binário 00001111b, indica que os 4 primeiros LEDs da barra estão ligados e os 4 últimos desligados.

Retorno: Nenhum.

CMD_DISP_7SEG_SET_NUM: Definir número de display de 7 segmentos

Código: 0x35

Descrição: Define o número que será apresentado em um display de 7 segmentos. Além dos digitos de 0 à 9, suporta também apresentar os dígitos hexadecimais A, B, C, D, E e F.

Parâmetros: Um byte indicando o display de 7 segmentos que se deseja definir o número seguido de um byte com valor de 0 à 15 indicando o número que deve ser apresentado.

Retorno: Nenhum.

CMD_DISP_7SEG_SET_STATUS: Definir estado de display de 7 segmentos

Código: 0x36

Descrição: Define os segmentos de um display de 7 segmentos que deverão ser ligados. Esse comando permite o controle total sobre o display.

Parâmetros: Um byte indicando o display de 7 segmentos que se deseja definir o estado seguido de um byte indicando que segmentos devem ser ligados. Os bits desse byte indicam se o segmento estará ligado ou desligado. A seguir o esquema de um display de 7 segmento e os bits que representam seus segmentos.

Esquema	Segmento	Descrição	Bit
---(a)---	a	Superior	0
(f) (b)	b	Superior Direito	1
---(g)---	c	Inferior Direito	2
(e) (c)	d	Inferior	3
---(d)--- .(h)	e	Inferior Esquerdo	4
	f	Superior Esquerdo	5
	g	Central	6
	h	Ponto	7

Para apresentar o número 1 é necessário que os segmentos 'b' e 'c' sejam acesos. Dessa forma o parâmetro que devemos utilizar é 6 , que em binário é representado por 00000110b. Como podemos ver sem problemas pela representação binária, os bits 1 e 2 (a contagem é feita da direita para a esquerda) são 1, o que indica que os segmentos representados por esses devem ser acesos. Os outros segmentos ficam apagados pois os bits que os representam são 0.

Retorno: Nenhum.

CMD_AUDIO_ON: Lig/Des som (Buzzer)

Código: 0x37

Descrição: Liga ou desliga o áudio.

Parâmetros: Um valor de 16 bits indicando o tempo que o som permanecerá ligado em milisegundos seguido de outro valor de 16 bits que é um parâmetro de freqüência do som emitido. Os dois parâmetros de 16 bits são enviados da mesma forma, primeiro o byte mais significativo seguido do menos. Se algum desses parâmetros for 0 o audio será desligado imediatamente.

Retorno: Nenhum.

CMD_DCMOTOR_ON: Lig/Des motor DC

Código: 0x38

Descrição: Liga ou desliga um motor DC.

Parâmetros: Um byte indicando qual motor deve ser ligado. Outro byte indicando a direção que pode assumir dois valores, 0 ou 1. Por fim um byte que indica a velocidade de rotação do motor. Para parar um motor em movimento envie como parâmetro de velocidade o valor 0.

Retorno: Nenhum.

CMD_STEP_MOTOR_ON: Lig/Des motor de passo

Código: 0x39

Descrição: Liga ou desliga o motor de passo.

Parâmetros: Um byte com a direçao que pode assumir o valor 0 ou 1. Outro byte com a velocidade com um valor de 0 à 255. Por fim um byte indicando o número de passos. Caso o número de passos seja 255 (0xFF em hexadecimal) então o motor irá rodar continuamente até que se envie novamente um comando `CMD_STEP_MOTOR_ON` que irá se sobrepor ao anterior. Se o parâmetro de velocidade ou número de passos for 0 o motor irá parar imediatamente.

Retorno: Nenhum.

CMD_RELAY_ON: Lig/Des Chave Eletromagnética (Relé)

Código: 0x3A

Descrição: Liga ou desliga uma chave eletromagnética (Relé).

Parâmetros: Um byte indicando qual chave deve ser controlada seguido de um byte indicando se a chave deve ser ligada ou desligada. Um valor 0 no último parâmetro indica que a chave deve ser desligada e 1 que deve ser ligada.

Retorno: Nenhum.

CMD_DIGITAL_PORT_READ: Ler porta digital

Código: 0x40

Descrição: Lê valor de uma porta digital.

Parâmetros: Um byte indicando a porta que será feita a leitura.

Retorno: Um byte com a leitura da porta.

CMD_DIGITAL_PORT_WRITE: Gravar em porta digital

Código: 0x41

Descrição: Grava um valor em uma porta digital.

Parâmetros: Um byte indicando a porta seguido de outro byte com o valor que deve ser gravado.

Retorno: Nenhum.

CMD_SERVO_MOTOR_ON: Ligar Servo Motor

Código: 0x42

Descrição: Liga um servo-motor. Atualmente o *Módulo de Entradas, Saídas e Servo-Motores* pode controlar 8 servo-motores. Desses, os quatro primeiros, numerados de 0 à 3, tem um controle preciso, possibilitando a especificação de 256 posições. Já o controle dos quatro ultimos servo-motores, numerados de 4 à 7, é menos precisa e consegue controlar apenas 3 posições, que são inicio meio e fim. Para esses quatro último motores, se for passado como parâmetro de posição um valor de 0 à 85, ele permanecerá na posição de início, um valor de 86 à 171 fará com que o motor permaneça no meio do curso e um valor de 172 à 255 faz com que o motor permaneça no final do curso. Para controlar a direção do giro de servo-motores que foram modificados para giro contínuo é necessário apenas passar como parâmetro de posição o valor 0 que fará com que o motor gire em uma direção ou o valor 255 para que ele gire para a direção contrária.

Parâmetros: Um byte indicando o servo-motor que deve ser ligado seguido de outro byte que indica a posição.

Retorno: Nenhum.

CMD_SERVO_MOTOR_OFF : Desligar Servo Motor

Código: 0x43

Descrição: Desliga um servo-motor.

Parâmetros: Um byte indicando que servo-motor deve ser desligado.

Retorno: Nenhum.

Comando	*Cód(Dec)*	*Cód(Hex)*	*Parametros*	*Retorno*
CMD_NOP	00	0x00	-	-
CMD_DEVICE_NAME	01	0x01	-	Descritivo**
CMD_DEVICE_STATUS	02	0x02	-	Estado Atual**
CMD_IRDA_ON	03	0x03	Ligar/Desligar	-
CMD_FLASH_WRITE	04	0x04	Dados*	-
CMD_FLASH_READ	05	0x05	Número de bytes	Dados Lidos**
CMD_FLASH_ERASE	06	0x06	-	-
CMD_SENSOR_READ_NOW	16	0x10	Sensor	Leitura
CMD_SENSOR_READ_ALL	17	0x11	-	Leituras**
CMD_SENSOR_BUFFER_ON	18	0x12	Lig/Des, Sensor, Intervalo	-
CMD_SENSOR_READ_BUFFER	19	0x13	-	Leituras**
CMD_SENSOR_CLEAR_BUFFER	20	0x14	-	-
CMD_LCD_ON	32	0x20	Ligar/Desligar	-
CMD_LCD_WRITE_TEXT	33	0x21	Texto*	-
CMD_LCD_GOTOXY	34	0x22	Linha, Coluna	-
CMD_LCD_CLEAR	35	0x23	-	-
CMD_LCD_CURSOR_ON	36	0x24	Ligar/Desligar	-
CMD_LCD_BACKLIGHT_ON	37	0x25	Ligar/Desligar	-
CMD_KEYBOARD_READ_NOW	48	0x30	-	Tecla
CMD_KEYBOARD_READ_BUFFER	49	0x31	-	Tecla
CMD_KEYBOARD_CLEAR_BUFFER	50	0x32	-	-
CMD_LEDBAR_SET_LEVEL	51	0x33	Barra, Nível	-
CMD_LEDBAR_SET_STATUS	52	0x34	Barra, Status	-
CMD_DISP_7SEG_SET_NUM	53	0x35	Display, Número	-
CMD_DISP_7SEG_SET_STATUS	54	0x36	Display, Status	-
CMD_AUDIO_ON	55	0x37	Tempo, Freqüência	-
CMD_DCMOTOR_ON	56	0x38	Motor, Direção, Velocidade	-
CMD_STEP_MOTOR_ON	57	0x39	Direção, Velocidade, Passos	-
CMD_RELAY_ON	58	0x3A	Chave, Ligar/Desligar	-
CMD_DIGITAL_PORT_READ	64	0x40	Porta	Valor
CMD_DIGITAL_PORT_WRITE	65	0x41	Porta, Valor	-
CMD_SERVO_MOTOR_ON	66	0x42	Motor, Posição	-
CMD_SERVO_MOTOR_OFF	67	0x43	Motor	-

* Parâmetros com tamanho variável. Antes de enviar os dados do parâmetro, é necessário enviar um valor de 16 bits que indica quantos bytes serão enviados como parâmetro.

** Retornos com tamanho variável. Os dois primeiros bytes do retorno são um valor de 16 bits que indicam o número de bytes do retorno.

Obs: Configuração da porta serial - 38400 Bauds, 8 bits, Sem Paridade e 1 Stop Bit.

1 - Principais Mudanças da Versão 1.0.0 para 1.1.0

- Constante *REQUEST* foi renomeada para *BEGIN_CMD*.
- Constante *ACK* foi renomeada para *END_CMD*.
- Modificação na seqüência de envio de comando. Após o término do processamento de todo comando será enviado um *END_CMD*. A seguir a nova seqüência para envio de comando:

```
BEGIN_CMD -> Código do comando -> [parâmetros] <- [retorno] <- END_CMD
```

- Configuração da Porta Serial: 38400 Bauds, 1 StopBit, Sem Paridade e 8 bits de dados.
- Padronização do envio de parâmetros e recebimento de retornos com tamanho variável.
- Vários comando do tipo "LIGA/DESLIGA" modificados para que exista apenas um comando e que a função de ligar ou desligar seja definada por parâmetro. Ex. *CMD_LCD_ON*, *CMD_LCD_CURSOR_ON*, *CMD_RELAY_ON*
- Renomeação dos comandos para termos em inglês.
- Remoção de 16 comandos.
- Modificação de funcionamento em 13 comandos.
- Adição de 9 comandos

Comandos apenas renomeados

CMD_PING	->	*CMD_NOP*
CMD_ENVIAR_INFRAVERMELHO	->	*CMD_IRDA_ON*
CMD_GRAVAR_TEXTO_NA_FLASH	->	*CMD_FLASH_WRITE*
CMD_LER_TEXTO_DA_FLASH	->	*CMD_FLASH_READ*
CMD_APAGAR_MEMORIA_FLASH	->	*CMD_FLASH_ERASE*
CMD_LER_SENSORES	->	*CMD_SENSOR_READ_NOW*
CMD_LER_SENSORES_PACOTE	->	*CMD_SENSOR_READ_ALL*
CMD_LER_TECLADO_AGORA	->	*CMD_KEYBOARD_READ_NOW*
CMD_LER_BUFFER_TECLADO	->	*CMD_KEYBOARD_READ_BUFFER*
CMD_LIMPAR_BUFFER	->	*CMD_KEYBOARD_CLEAR_BUFFER*
CMD_GRAVAR_TEXTO_LCD	->	*CMD_LCD_WRITE_TEXT*
CMD_GOTOXY_CURSOR_LCD	->	*CMD_LCD_GOTOXY*
CMD_LIMPAR_TELA_LCD	->	*CMD_LCD_CLEAR*
CMD_LIGAR_LCD	->	*CMD_LCD_ON*
CMD_LIGAR_CURSOR	->	*CMD_LCD_CURSOR_ON*
CMD_LIGAR_LUZ_DE_FUNDO	->	*CMD_LCD_BACKLIGHT_ON*
CMD_GRAVAR_LEVEL_BARRA1	->	*CMD_LEDBAR_SET_LEVEL*
CMD_GRAVAR_DIRETO_BARRA1	->	*CMD_LEDBAR_SET_STATUS*
CMD_GRAVAR_NUMERO_7SEG1	->	*CMD_DISP_7SEG_SET_NUM*
CMD_GRAVAR_DIRETO_7SEG1	->	*CMD_DISP_7SEG_SET_STATUS*
CMD_LIGAR_AUDIO	->	*CMD_AUDIO_ON*
CMD_LIGAR_MOTORDC1	->	*CMD_DCMOTOR_ON*

- *CMD_LIGAR_MOTOR_PASSO* -> *CMD_STEP_MOTOR_ON*
- *CMD_LIGAR_CHAVE1* -> *CMD_RELAY_ON*

Comandos removidos

- *CMD_LER_VERSAO_PROGRAMA*
- *CMD_DESLIGAR_LCD*
- *CMD_DESLIGAR_CURSOR*
- *CMD_DESLIGAR_LUZ_DE_FUNDO*
- *CMD_GRAVAR_LEVEL_BARRA2*
- *CMD_GRAVAR_DIRETO_BARRA2*
- *CMD_GRAVAR_NUMERO_7SEG2*
- *CMD_GRAVAR_DIRETO_7SEG2*
- *CMD_DESLIGAR_AUDIO*
- *CMD_DESLIGAR_MOTORDC1*
- *CMD_LIGAR_MOTORDC2*
- *CMD_DESLIGAR_MOTORDC2*
- *CMD_DESLIGAR_MOTOR_PASSO*
- *CMD_DESLIGAR_CHAVE1*
- *CMD_LIGAR_CHAVE2*
- *CMD_DESLIGAR_CHAVE2*

Comandos Modificados

- *CMD_ENVIAR_INFRAVERMELHO* *(CMD_IRDA_ON)*
- *CMD_LER_TEXTO_DA_FLASH* *(CMD_FLASH_READ)*
- *CMD_LER_SENSORES_PACOTE* *(CMD_SENSOR_READ_ALL)*
- *CMD_LIGAR_LCD* *(CMD_LCD_ON)*
- *CMD_LIGAR_CURSOR* *(CMD_LCD_CURSOR_ON)*
- *CMD_LIGAR_LUZ_DE_FUNDO* *(CMD_LCD_BACKLIGHT_ON)*
- *CMD_GRAVAR_LEVEL_BARRA1* *(CMD_LEDBAR_SET_LEVEL)*
- *CMD_GRAVAR_DIRETO_BARRA1* *(CMD_LEDBAR_SET_STATUS)*
- *CMD_GRAVAR_NUMERO_7SEG1* *(CMD_DISP_7SEG_SET_NUM)*
- *CMD_GRAVAR_DIRETO_7SEG1* *(CMD_DISP_7SEG_SET_STATUS)*
- *CMD_LIGAR_MOTORDC1* *(CMD_DCMOTOR_ON)*
- *CMD_LIGAR_MOTOR_PASSO* *(CMD_STEP_MOTOR_ON)*
- *CMD_LIGAR_CHAVE1* *(CMD_RELAY_ON)*

Mudanças no funcionamento dos comandos

- *CMD_ENVIAR_INFRAVERMELHO (CMD_IRDA_ON)*: Modificado para que seja um comando do tipo "LIGA/DESLIGA". Controla o envio de comandos pela porta IrDA, uma vez ligado o modo IrDA todos os comando enviados serão encaminhados para a porta IrDA, sendo necessário enviar esse comando novamente indicando que se deseja desligar o modo IrDA para que os comandos sejam executados no Kit Didático local.

- *CMD_LER_TEXTO_DA_FLASH (CMD_FLASH_READ)*: Os dois primeiros bytes do retorno desse comando indicam quantos bytes serão enviados.

- *CMD_LER_SENSORES_PACOTE (CMD_SENSOR_READ_ALL)*: Os dois primeiros bytes do retorno desse comando indicam quantos bytes serão enviados no retorno.

- *CMD_LIGAR_LCD (CMD_LCD_ON):* Adicionado parâmetro para indicar se o display LCD deve ser ligado ou desligado.

- *CMD_LIGAR_CURSOR (CMD_LCD_CURSOR_ON):* Adicionado parâmetro para indicar se o cursor do display LCD deve ser ligado ou desligado.

- *CMD_LIGAR_LUZ_DE_FUNDO (CMD_LCD_BACKLIGHT_ON):* Adicionado parâmetro para indicar se a iluminação de fundo do display LCD deve ser ligada ou desligada.

- *CMD_GRAVAR_LEVEL_BARRA1 (CMD_LEDBAR_SET_LEVEL):* Adicionado parâmetro para indicar a barra de LEDs que se deseja definir o nível.

- *CMD_GRAVAR_DIRETO_BARRA1 (CMD_LEDBAR_SET_STATUS):* Adicionado parâmetro para indicar a barra de LEDs que se deseja definir o estado dos LEDs.

- *CMD_GRAVAR_NUMERO_7SEG1 (CMD_DISP_7SEG_SET_NUM):* Adicionado parâmetro para indicar o display de 7 segmentos que se deseja definir o número exibido.

- *CMD_GRAVAR_DIRETO_7SEG1 (CMD_DISP_7SEG_SET_STATUS):* Adicionado parâmetro para indicar o display de 7 segmentos que se deseja definir o estado dos segmentos de LED.

- *CMD_LIGAR_MOTORDC1 (CMD_DCMOTOR_ON):* Adicionado parâmetro para indicar o motor que se deseja manipular.

- *CMD_LIGAR_CHAVE1(CMD_RELAY_ON):* Adicionado parâmetro para indicar o relè que se deseja manipular.

Comandos adicionados

- *CMD_DEVICE_NAME*
- *CMD_DEVICE_STATUS*
- *CMD_SENSOR_BUFFER_ON*
- *CMD_SENSOR_READ_BUFFER*
- *CMD_SENSOR_CLEAR_BUFFER*
- *CMD_DIGITAL_PORT_READ*
- *CMD_DIGITAL_PORT_WRITE*
- *CMD_SERVO_MOTOR_ON*
- *CMD_SERVO_MOTOR_OFF*

Resumo das Modificações de Comandos da Versão 1.0.0 para 1.1.0

Comando v1.0.0	*Mudança*	*Comando v1.1.0*
REQUEST	(=)	BEGIN_CMD
ACK	(=)	END_CMD
CMD_PING	(=)	CMD_NOP
CMD_ENVIAR_INFRAVERMELHO	(*)	CMD_IRDA_ON
CMD_GRAVAR_TEXTO_NA_FLASH	(=)	CMD_FLASH_WRITE
CMD_LER_TEXTO_DA_FLASH	(*)	CMD_FLASH_READ
CMD_APAGAR_MEMORIA_FLASH	(=)	CMD_FLASH_ERASE
CMD_LER_VERSAO_PROGRAMA	(-)	-
CMD_LER_SENSORES	(=)	CMD_SENSOR_READ_NOW
CMD_LER_SENSORES_PACOTE	(*)	CMD_SENSOR_READ_ALL
CMD_LER_TECLADO_AGORA	(=)	CMD_KEYBOARD_READ_NOW
CMD_LER_BUFFER_TECLADO	(=)	CMD_KEYBOARD_READ_BUFFER
CMD_LIMPAR_BUFFER	(=)	CMD_KEYBOARD_CLEAR_BUFFER
CMD_GRAVAR_TEXTO_LCD	(=)	CMD_LCD_WRITE_TEXT
CMD_GOTOXY_CURSOR_LCD	(=)	CMD_LCD_GOTOXY
CMD_LIMPAR_TELA_LCD	(=)	CMD_LCD_CLEAR
CMD_LIGAR_LCD	(*)	CMD_LCD_ON
CMD_DESLIGAR_LCD	(-)	-
CMD_LIGAR_CURSOR	(*)	CMD_LCD_CURSOR_ON
CMD_DESLIGAR_CURSOR	(-)	-
CMD_LIGAR_LUZ_DE_FUNDO	(*)	CMD_LCD_BACKLIGHT_ON
CMD_DESLIGAR_LUZ_DE_FUNDO	(-)	-
CMD_GRAVAR_LEVEL_BARRA1	(*)	CMD_LEDBAR_SET_LEVEL
CMD_GRAVAR_DIRETO_BARRA1	(*)	CMD_LEDBAR_SET_STATUS
CMD_GRAVAR_LEVEL_BARRA2	(-)	-
CMD_GRAVAR_DIRETO_BARRA2	(-)	-
CMD_GRAVAR_NUMERO_7SEG1	(*)	CMD_DISP_7SEG_SET_NUM
CMD_GRAVAR_DIRETO_7SEG1	(*)	CMD_DISP_7SEG_SET_STATUS
CMD_GRAVAR_NUMERO_7SEG2	(-)	-
CMD_GRAVAR_DIRETO_7SEG2	(-)	-
CMD_LIGAR_AUDIO	(=)	CMD_AUDIO_ON
CMD_DESLIGAR_AUDIO	(-)	-
CMD_LIGAR_MOTORDC1	(*)	CMD_DCMOTOR_ON
CMD_DESLIGAR_MOTORDC1	(-)	-
CMD_LIGAR_MOTORDC2	(-)	-
CMD_DESLIGAR_MOTORDC2	(-)	-
CMD_LIGAR_MOTOR_PASSO	(*)	CMD_STEP_MOTOR_ON
CMD_DESLIGAR_MOTOR_PASSO	(-)	-
CMD_LIGAR_CHAVE1	(*)	CMD_RELAY_ON
CMD_DESLIGAR_CHAVE1	(-)	-

Comando v1.0.0	*Mudança*	*Comando v1.1.0*
CMD_LIGAR_CHAVE2	(-)	-
CMD_DESLIGAR_CHAVE2	(-)	-

(=)Comando não foi alterado (-)Comando Removido (*)Comando alterado