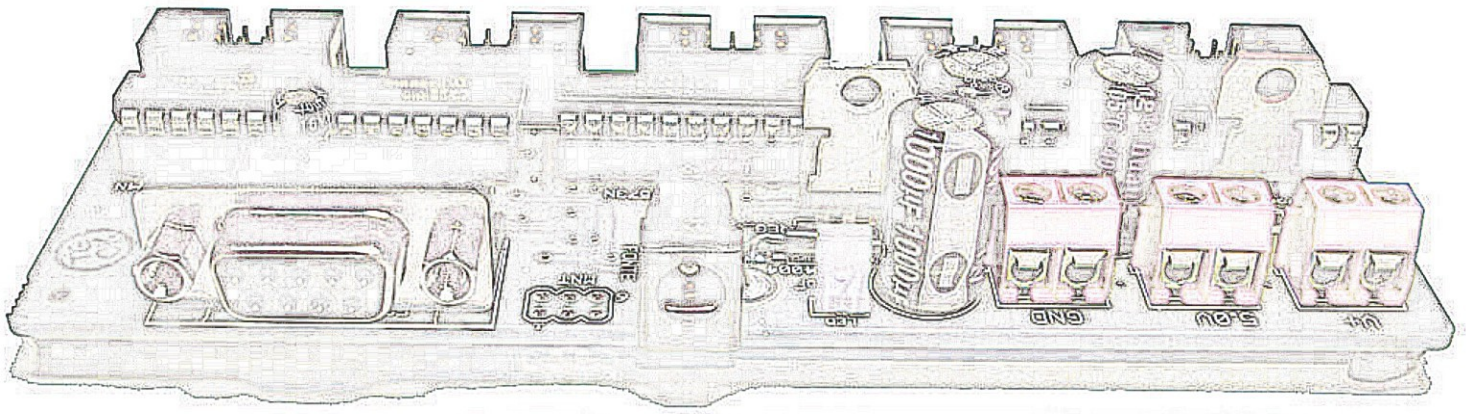
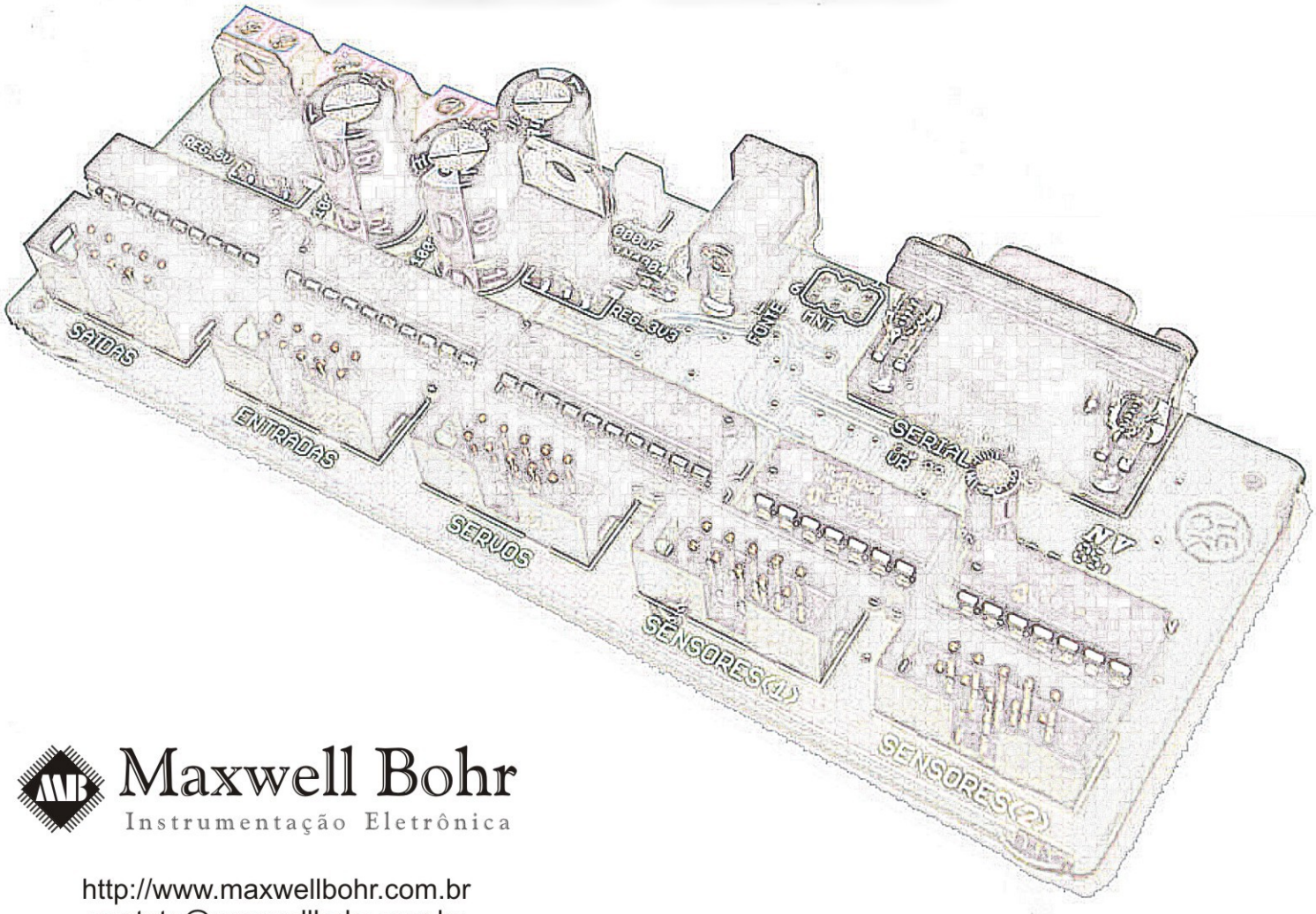


MANUAL DO MEC 1000



Maxwell Bohr
Instrumentação Eletrônica

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

Introdução

Esse documento tem como objetivo guiar o usuário durante a montagem e o uso do *MEC1000*. Os detalhes que compõem o equipamento são descritos de forma objetiva, de modo a facilitar a compreensão sobre o seu funcionamento.

A descrição física e funcional do equipamento visa esclarecer ao usuário o funcionamento do *MEC1000,* suas capacidades e funcionalidades, de modo que possa ser aproveitada toda sua estrutura para a utilização em projetos de automação.

Com esse equipamento, o usuário poderá aplicar seus conhecimentos de mecatrônica e robótica em projetos reais, analisar seu funcionamento e com isso, expandir sua habilidade na área, tendo como limitante apenas sua criatividade.

O *MEC1000*

O *MEC1000* 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 *MEC1000* é um equipamento 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.

As operações realizadas pelo *MEC1000* são controladas por programas que rodam no computador. Esses programas enviam comandos de controle para o *MEC1000* 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 *MEC1000* pudesse ser utilizado em uma grande gama de projetos, as entradas digitais, saídas digitais, controle de servo-motores e entradas de sinais de sensores suportam vários tipos de componentes e circuitos, desde que estejam dentro das especificações do *MEC1000*. Por exemplo, na porta de sensores podem ser utilizados vários tipos de sensores simultaneamente, não estando limitados a um modelo específico, bastando apenas serem compatíveis ao *MEC1000*.

Também não há nenhum vínculo lógico a nível de hardware entre os itens que podem ser conectados ao *MEC1000***.** Esses vínculos são criados exclusivamente no programa de controle no computador, conforme a necessidade de cada projeto, proporcionando uma maior versatilidade.

Por exemplo, um vínculo que pode ser criado em um programa de controle é o de que um componente controlado pelo *MEC1000,* escolhido e conectado pelo usuário, por exemplo um relé, um motor ou um LED, seja ligado assim que o valor da leitura de um sensor, como por exemplo um sensor de temperatura, ultrapasse um certo valor. Para o *MEC1000,* o componente 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 *MEC1000* solicitando que um componente seja ligado. Dessa forma teríamos um vínculo entre esse componente 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 *MEC1000* seja utilizado nos mais diversos projetos sendo necessário apenas adaptar a lógica no programa de controle. Podemos considerar que o *MEC1000* fornece vários recursos independentes que podem ser unidos 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 *MEC1000* oferece, é possível criar desde projetos simples até projetos de alta complexidade na área de mecatrônica. Veremos agora a composição do *MEC1000.*

Composição do MEC1000

O *MEC1000* é composto por uma única placa que foi desenvolvida de modo a poder realizar inúmeras operações de acordo com o programa desenvolvido pelo usuário. O "cérebro" de todo o sistema localiza-se na parte inferior da placa. Trata-se do microcontrolador, um componente eletrônico que pode ser considerado como um mini computador em um chip, otimizado para controlar dispositivos eletrônicos. Ele também possui uma memória interna, dentro da qual está o firmware, um programa que intercepta os comandos enviados por um computador e faz com que o *MEC1000* responda adequadamente.

Para permitir a comunicação e intercâmbio de dados entre o computador e o *MEC1000* é necessário que ambos estejam conectados através de um cabo serial. Há nessa placa um conector de cabos seriais para realizar tal conexão. Também está presente um conector para a fonte de alimentação e um LED que indica se o equipamento está ligado.

O *MEC1000* foi produzido de modo a poder interagir com diversos tipos de componentes, entre eles motores, sensores e até mesmo circuitos externos. Toda a interação do *MEC1000* com esses componentes é realizada através de portas de comunicação. Cada tipo de porta possui uma especificação, sendo destinada a um propósito diferente. Os quatro tipos diferentes de portas são os seguintes:

- **Saídas Digitais.** As saídas digitais possibilitam que o MEC1000 se comunique com circuitos externos, como por exemplo uma barra de LEDs ou um motor DC. Ela é composta por 2 pinos de alimentação (5V e 0V) e 8 bits de saída de sinais digitais.

- **Entradas Digitais.** As entradas digitais são responsáveis pela recepção de sinais digitais oriundos de circuitos externos, tais como sensores digitais e chaves. Ela é composta por dois pinos de alimentação (5V e 0V) e 8 bits de entrada de sinais digitais.

- **Saídas de Sinais de Controle de Servo-motores.** Essas saídas são destinadas ao controle de servo-motores.Através delas, é possível controlar um sistema mecânico automatizado com até 6 servo-motores operando simultaneamente.

- **Entradas de Sinais de Sensores .** São destinadas à conexão com sensores. Através delas, o *MEC1000* pode comunicar-se simultaneamente com 6 sensores analógicos e 2 sensores digitais através dos dois conectores.

A figura a seguir mostra o MEC1000 com a indicação de todos os seus ítens.

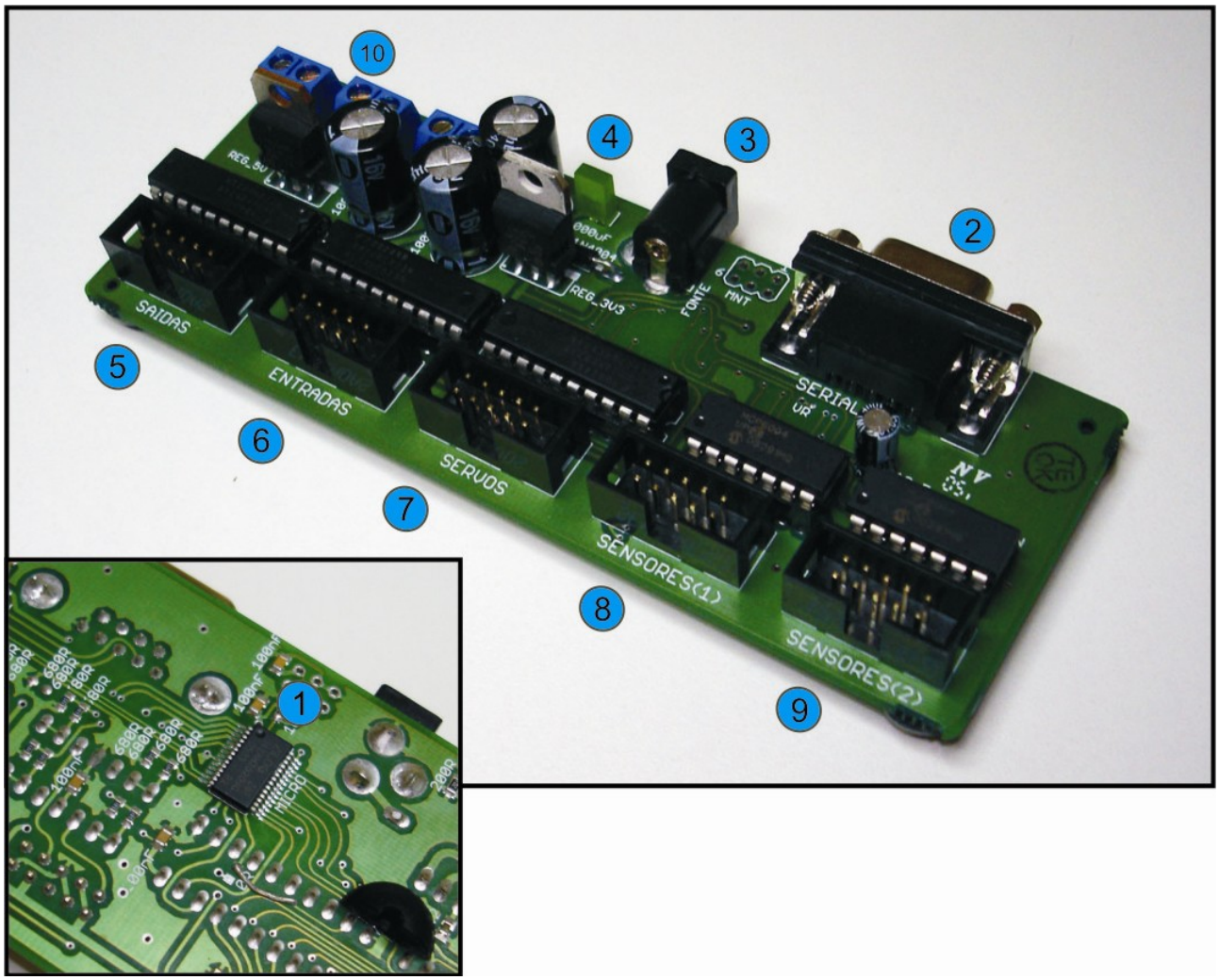


Figura 1: 1-Microcontrolador. 2-Conector para cabo serial. 3-Conector da fonte de energia. 4-LED de indicação liga/desliga. 5-Porta de saídas digitais. 6-Porta de entradas digitais. 7-Porta de controle de servo-motores. 8-Porta de sensores 0. 9-Porta de sensores 1. 10-Saídas de Alimentação.

Saídas Digitais

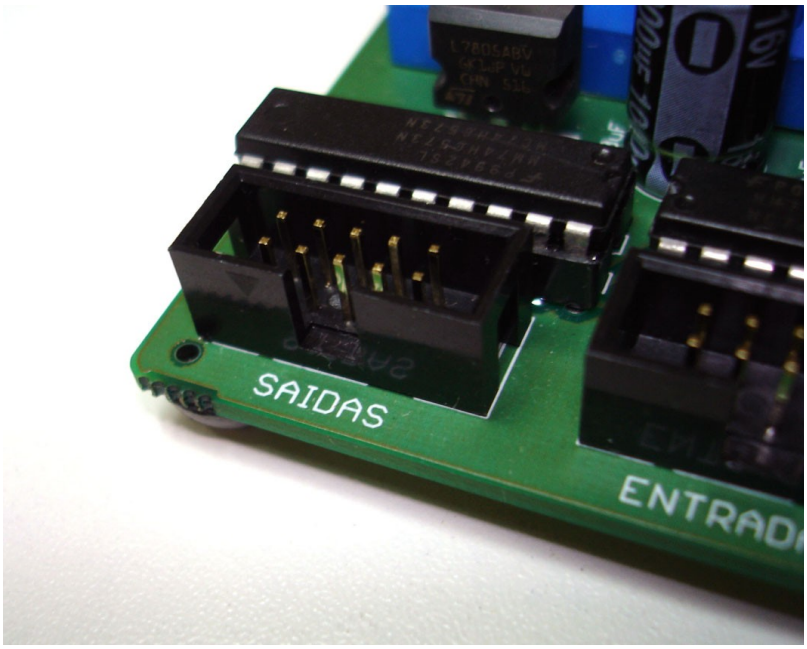


Figura 2: Porta de saídas digitais.

As saídas digitais são responsáveis pela conexão do *MEC1000* com dispositivos externos. Estão presentes 8 saídas digitais, cada uma com um bit. Ao todo, podem ser enviados 8 níveis lógicos diferenres simultaneamente. A imagem abaixo mostra a pinagem do conector de saídas digitais.

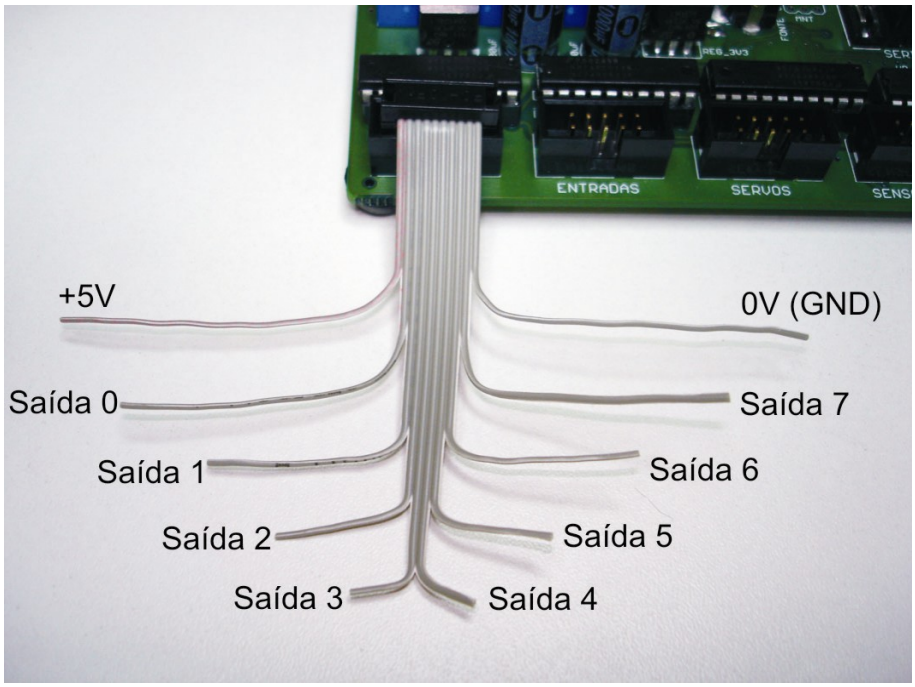


Figura 3: Conexão da porta de saídas digitais.

Observando a imagem, podemos ver que os pinos das extremidades são responsáveis pela alimentação (5 Volts e 0 Volt), enquanto que os oito pinos centrais enviam os sinais digitais. Para um nível lógico “1”, a tensão de saída é 3,3 Volts. Para um nível lógico 0, a tensão de saída é 0 Volt.

É possível utilizá-la em inúmeras aplicações, especialmente para controlar circuitos externos, como LEDs, motores, displays. O usuário deverá ter cuidado com a capacidades do circuito a ser conectado, de modo a não causar danos tanto ao equipamento quanto ao *MEC1000.*

Entradas Digitais

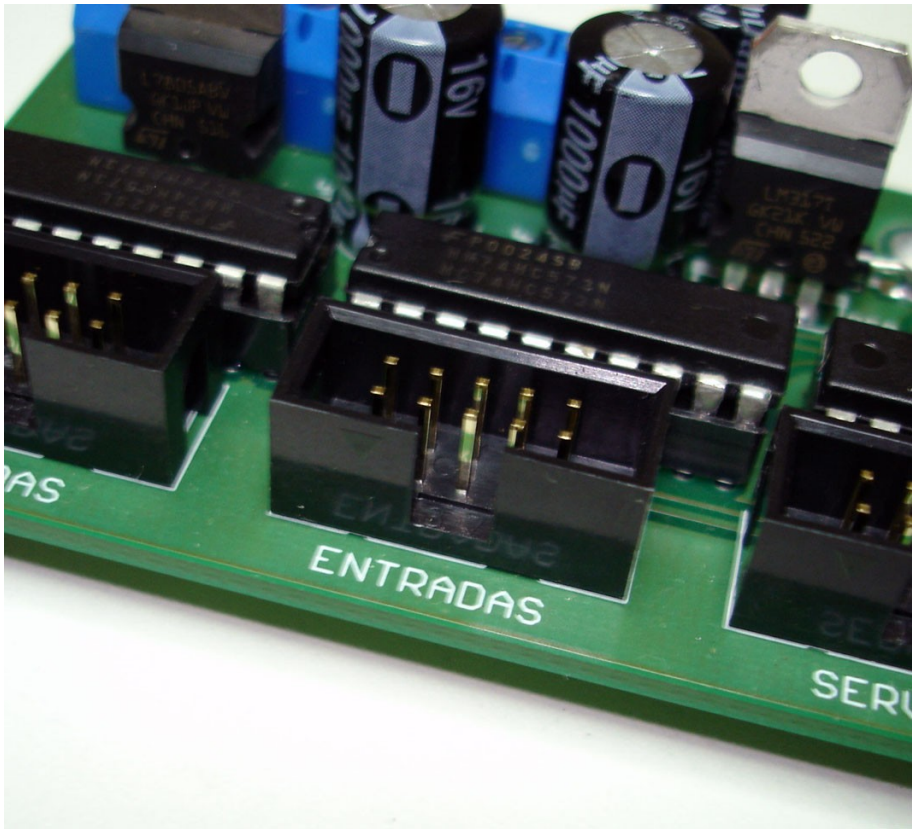


Figura 4: Porta de entradas digitais.

Para que o *MEC1000* possa receber dados oriundos de circuitos externos, existem as entradas digitais. Existem oito entradas digitais, cada uma recebendo um nível lógico. Ao todo, o *MEC1000* pode receber 8 bits simultaneamente de suas 8 entradas. Como podemos ver na figura abaixo, a pinagem desse conector é muito semelhante à da porta de saídas digitais.

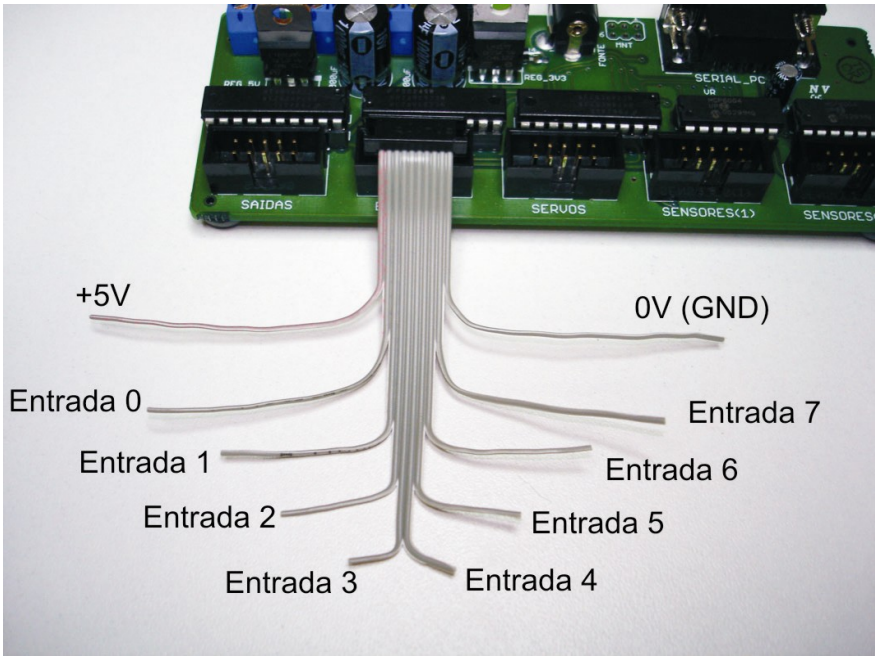


Figura 5: Esquema da porta de entradas digitais.

Observando a configuração da porta, podemos ver que, do mesmo modo que a porta de saídas digitais, as extremidades são destinadas à alimentação (5 Volts e 0 Volts). Os oito pinos centrais são destinados à entrada de sinais digitais. Uma tensão de 5 Volts corresponde a um nível lógico "1" e uma tensão de 0 Volt corresponde a um nível lógico "0".

Essas entradas podem ser usadas para receber sinais de sensores digitais, do mesmo modo que as entradas digitais do conector de sensores, possibilitando uma maior versatilidade ao MEC1000.

Em grande parte das aplicações, as conexões com essas portas serão simples, não causando tantas dificuldades ao usuário, principalmente se elas forem destinadas à leitura de sensores digitais. Para utilizações mais complexas, é necessário o conhecimento de eletrônica digital.

Saídas de Sinais de Controle de Servo-motores

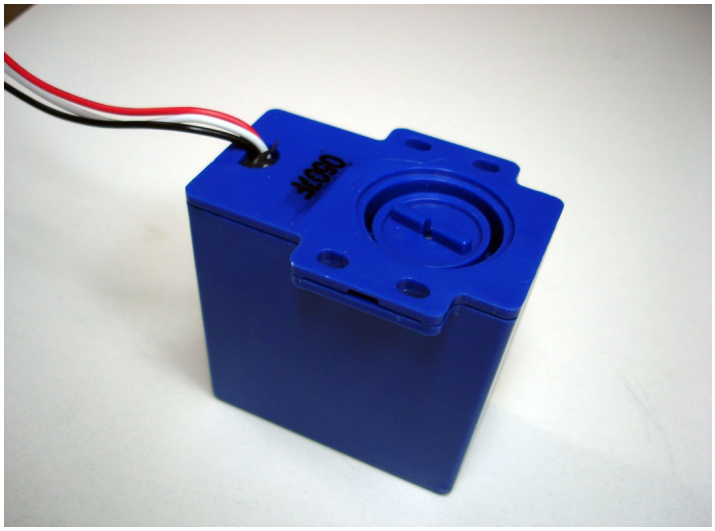


Figura 6: Servo-motor.

O servo-motor é um dispositivo constituído por um motor elétrico acoplado a uma caixa de redução, com o objetivo de aumentar o torque do movimento final. Uma placa de controle localizada no seu interior permite que, com um sinal externo, seja possível controlar a posição do seu eixo. A placa de controle faz a leitura da posição do eixo e compara com a posição desejada, fazendo os ajustes necessários. Devido ao torque disponível, os servo-motores são muito utilizados em muitos sistemas de robótica e automação, brinquedos, entre outros.

Em geral, os servo-motores possuem uma limitação no seu giro, que pode variar entre 90 e 270 graus. Podemos modificá-los para que possam exercer giro contínuo, embora a capacidade de controlar a posição do seu eixo seja perdida e só seja possível controlar a rotação.

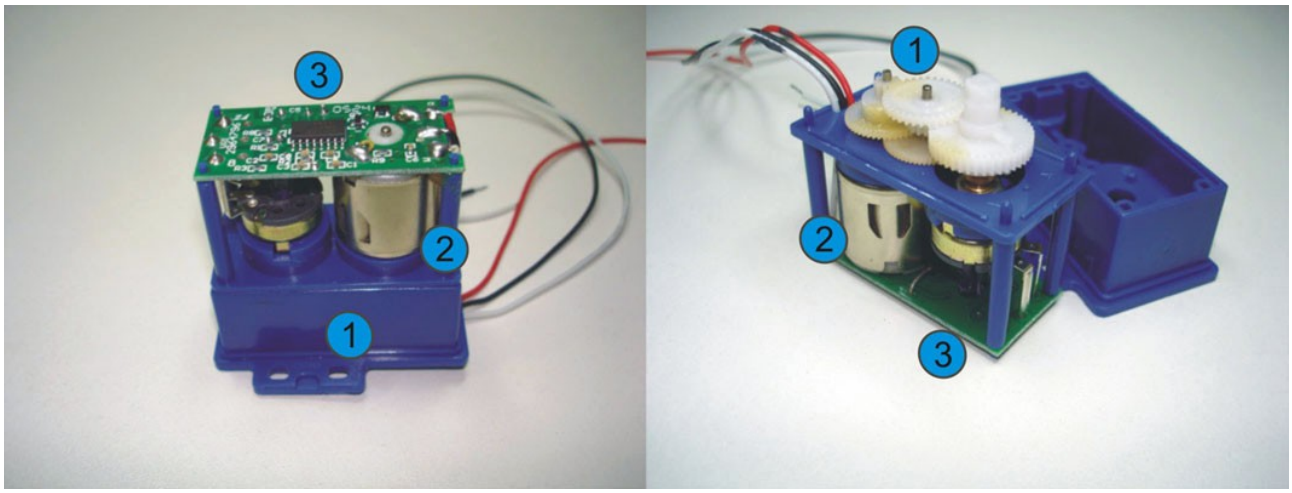


Figura 7: Servo-motor aberto, deixando à mostra seus componentes. (1) Caixa de redução, (2) Motor elétrico, (3) Módulo de controle.

O *MEC1000* possui uma porta destinada ao controle de servo-motores. A imagem abaixo mostra a pinagem dessa porta. Observe que os dois cabos de cada extremidade são destinados à alimentação (5 Volts e 0 Volt) e que os 6 pinos centrais são para o controle dos servos.

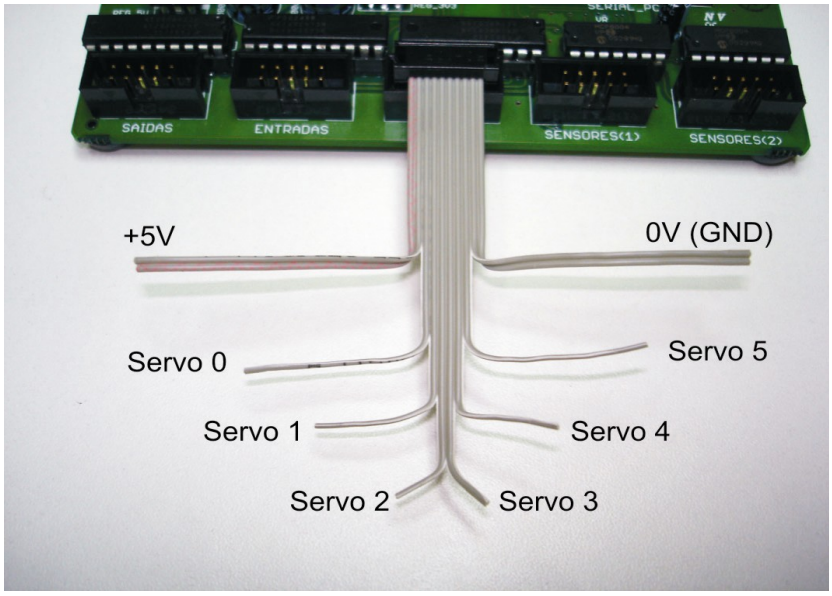


Figura 8: Configuração da porta de controle de servos.

A utilização dos servo-motores é limitada pela fonte de alimentação do *MEC1000* . É necessário o uso de alimentação externa quanto for necessário conectar mais de um servo-motor, pois o seu consumo não é suportado pela fonte de alimentação, que é capaz de fornecer apenas algumas centenas de miliamperes. Como consequência da ligação de vários servo-motores, a fonte poderá superaquecer e danos podem ser causados ao equipamento.

Entradas de Sinais de Sensores

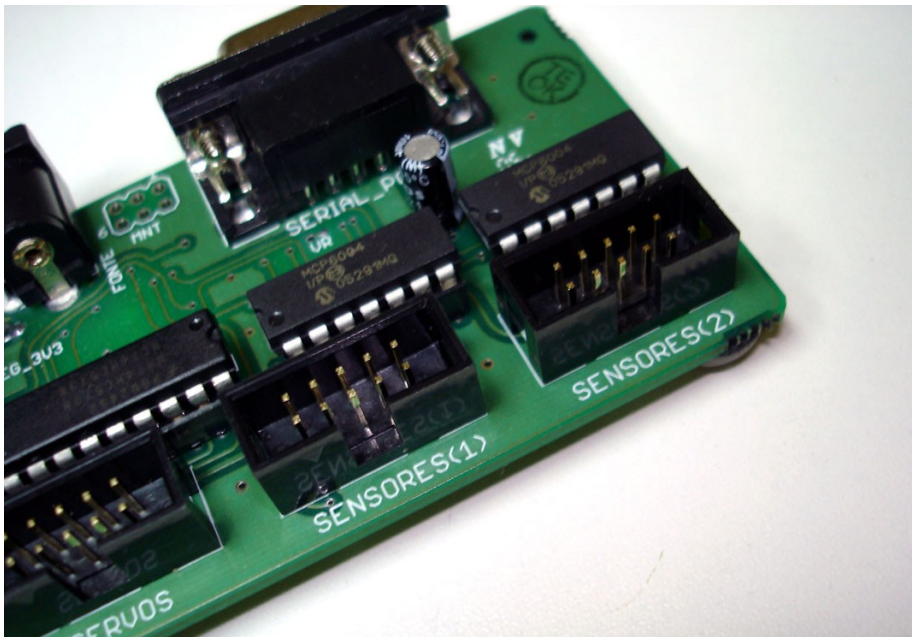


Figura 9: Portas de sensores.

Essas são as entradas que permitem a conexão de sensores ao *MEC1000.* Através dos sensores ligados nelas, o equipamento pode fazer a leitura das características do ambiente e, de acordo com os vínculos estabelecidos pelo usuário, responder a certas mudanças nessas características.

As entradas de sensores foram desenvolvidas de modo a serem muito flexíveis e dar liberdade ao usuário no momento de escolher os componentes adequados ao seu projeto. Basta ao usuário respeitar as especificações dos sensores e do *MEC1000* ao estabelecer as ligações.

O *MEC1000* possui dois conectores para entradas de sensores, cada um podendo ser conectado a três sensores analógicos e um digital. A pinagem desses conectores é idêntica. A imagem a seguir mostra a configuração da ligação nesses conectores.

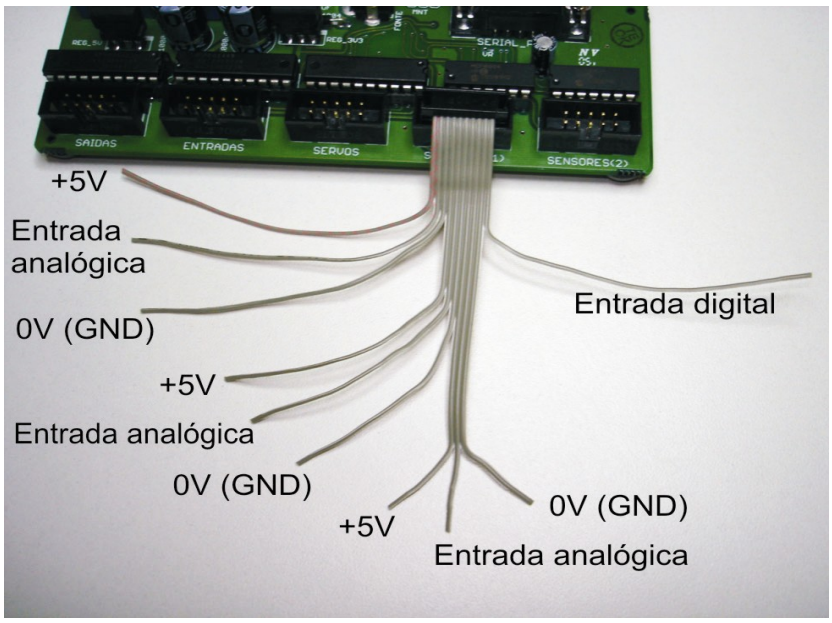


Figura 10: Configuração das portas de sensores.

Como podemos ver, os pinos do conector podem ser divididos em 3 conjuntos distintos. Cada conjunto é formado por um pino de 5V, um pino de 0V e uma entrada de sinal analógico. Além desses conjuntos, há uma entrada extra para o sinal de um sensor digital. Caso seja necessário utilizar mais sensores digitais que o suportado pelas Entradas de Sinais de Sensores, é possível conectá-los às Entradas Digitais.

Os pinos analógicos devem receber um sinal entre 0 e 5 Volts. Caso seja necessário usar um sensor que não se enquadre nessa especificação, é necessário convertê-lo a uma tensão entre 0 e 5 Volts. Já o sinal de um sensor digital deve ser de 5 Volts para representar nível lógico "1" e 0 Volt para representar nível lógico "0".

É importante lembrar que o uso de um componente que não seja adequado às portas de sensores do *MEC1000* ou algum erro no momento de estabelecer as conexões pode danificar tanto o componente quanto o *MEC1000.* Por isso, é essencial verificar muito bem as especificações e as pinagens antes de estabelecer qualquer ligação.

Saídas de alimentação

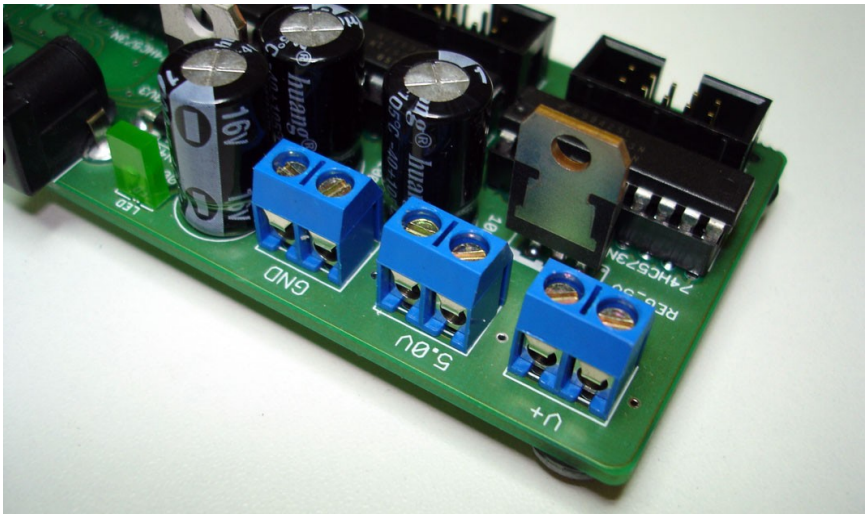


Figura 11: Saídas de alimentação.

Para auxiliar no desenvolvimento de projetos eletrônicos, foi adicionada ao *MEC1000* uma saída de alimentação para circuitos externos simples. Essa saída pode ser usada para alimentar circuitos montados numa matriz de contatos ou até mesmo em placas, com a condição de que não consumam muita corrente. Ela possui três terminais, um com 0 Volt (GND), um com 5 Volts estabilizados (5V) e um com a tensão da fonte na qual está ligado o *MEC1000* (V+).

Uma funcionalidade interessante da saída de alimentação é que ela pode ser utilizada para alimentar o *MEC1000.* Ligando o pólo positivo de uma fonte de alimentação ou de uma bateria no terminal V+ e o pólo negativo no GND o *MEC1000* será alimentado. A fonte a ser usada deve ter uma tensão entre 5 e 12 Volts. Mas é preciso muito cuidado ao realizar essa operação. Um erro na conexão dos pólos pode acarretar em danos irreversíveis ao *MEC1000.*

Montagem e operação

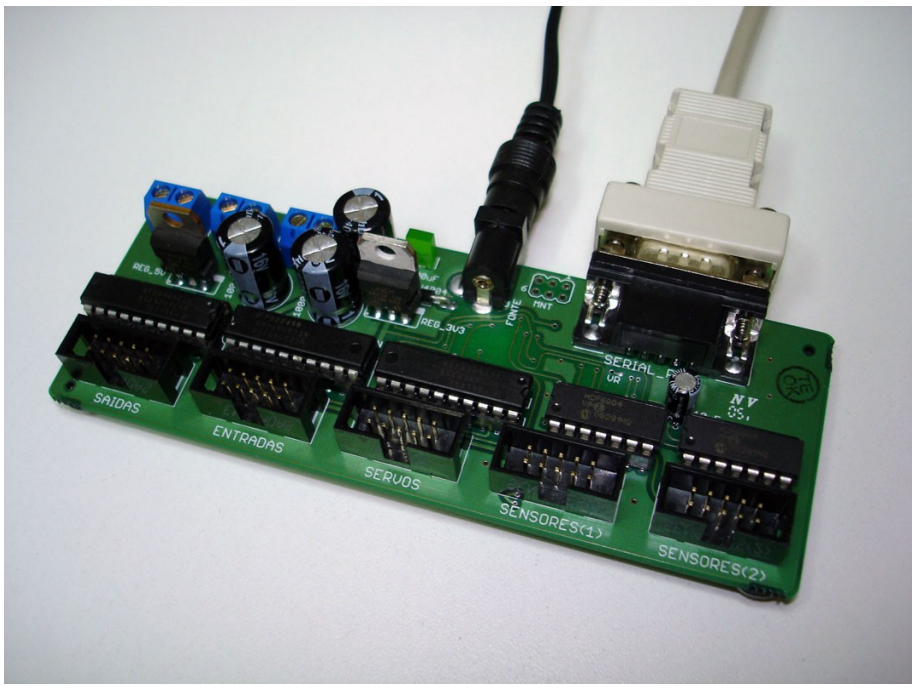


Figura 12: *MEC1000* com fonte de alimentação e cabo serial conectados.

O *MEC1000* foi desenvolvido de modo a ser um equipamento simples de montar e operar. Acompanhe abaixo a descrição de sua montagem.

A princípio, é muito importante que o *MEC1000* esteja desligado no momento de estabelecer as conexões e adicionar os componentes, caso contrário, poderá acarretar em danos a algum componente. Por isso, é importante verificar se o equipamento está desligado antes de realizar qualquer modificação na estrutura do *MEC1000,* ligando-o novamente depois que a montagem estiver pronta.

Durante a montagem e operação do *MEC1000* é preciso tomar outro cuidado muito importante. Como as placas e cabos do módulo ficam expostos, é importante cuidar para que os contatos elétricos dos ítens não entrem em curto circuito, o que poderia danificá-lo.

Também é importante verificar a tensão da tomada a qual a fonte será ligada e ajustar a chave para que a tensão selecionada seja a mesma da tomada. A chave da fonte ajusta a entrada para 127 ou 220V.

O primeiro passo é a montagem do módulo do *MEC1000.* Trata-se de um processo extremamente simples. Inicialmente, o *MEC1000* deve ser ligado à fonte de alimentação, bastando ligá-la no conector correspondente. O LED próximo ao conector deverá acender, indicando que o equipamento está ligado. Depois de ligá-lo, basta conectá-lo ao computador utilizando um cabo serial, que também possui um conector adequado no módulo.

Para conectar os sensores, motores e outros componentes, é necessário o conhecimento do método de crimpagem de cabos flat. A conexão desses equipamentos deve ser feita com muita atenção, pois a pinagem dos conectores deve ser respeitada, sabendo que os riscos de cometer um erro são reais e podem causar danos ao equipamento. Também é necessário cuidado com os cabos, de modo a não causar nenhum curto-circuito, o que também seria muito prejudicial ao *MEC1000* e aos componentes nele conectados.

É interessante notar que os programas desenvolvidos para o *KDR5000* são compatíveis

com o *MEC1000*. Devido às diferenças estruturais, nem todas as funcionalidades do *KDR5000* estão disponíveis no *MEC1000*, mas os programas que utilizam componentes presentes em ambos os modelos podem ser utilizados sem problema nos dois equipamentos. Por exemplo, um programa que controle os servo-motores pode ser utilizado tanto no *KDR500*0 quanto no *MEC1000*.

Programação para Controle do MEC1000

O MEC1000 não realiza nenhuma operação a não ser que um programa de controle envie um comando para tal. Isso permite que o ele 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 MEC1000.

Biblioteca de Controle

O MEC1000 responde a comandos enviados a partir de uma porta serial. Para isso é necessário abrir uma comunicação serial entre o computador e o MEC1000 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 MEC1000, facilitando muito a tarefa de programação do equipamento.

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 MEC1000.

Quando necessário, esses métodos possuem parâmetros como, por exemplo, qual servomotor será controlado. Com isso não é necessário saber os detalhes de codificação dos parâmetros no comando enviado ao MEC1000 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. Também agrupa métodos destinados a inicialização da comunicação do MEC1000 com um computador.
- **Métodos de Alto Nível:** Agrupa os métodos utilizados para o controle do MEC1000. Estão presentes nesse grupo métodos utilizados para inicializar uma comunicação, dar informações úteis sobre o sistema em que o programa esta rodando, ler um canal de sensores, controlar um servomotor e ler ou enviar dados através das portas de Entradas e Saídas digitais.

A seguir uma lista com as funções da biblioteca de controle separadas por grupos. Uma descrição mais detalhada dessas funções será dada ainda nesse documento.

Métodos de Baixo Nível

Os métodos desse grupo tratam da transmissão e codificação dos comandos e da comunicação com um computador e são destinados principalmente para uso interno na biblioteca. Esse grupo de métodos está acessível para serem utilizadas diretamente nos programas de controle, mas alguns métodos serão raramente utilizados.

Entretanto, existem métodos que serão utilizados em todos os programas que controlam um MEC1000, 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
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.
SendCommand	Envia um comando para um MEC1000.
SendCommandAndRecv	Envia um comando para um MEC1000 e recebe um retorno.
OpenCommunication	Inicializa a comunicação com um MEC1000 através de uma porta

Método	Descrição
	serial.
CloseCommunication	Finaliza a comunicação com um MEC1000.
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.

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 MEC1000.
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 MEC1000.
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 MEC1000 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 MEC1000 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 MEC1000 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 MEC1000 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.

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 MEC1000. ("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 de Alto Nível

Esse grupo contém os métodos que fornecem as funcionalidades do MEC1000. Esses métodos podem ser utilizados para a leitura dos canais de sensores, leitura e envio de dados através das Entradas e Saídas digitais ou controle de servomotores. A seguir uma tabela com esses métodos apresentando uma pequena descrição.

Método	Descrição
IsConnected	Verifica se existe algum MEC1000 conectado e ligado na porta serial.
NOP	Envia um comando 'NOP'. Não realiza nenhuma operação.
SensorReadNow	Lê um canal de sensor.
SensorReadAll	Lê todos os canais de sensores em uma única operação.
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.

IsConnected

Declaração	`Function IsConnected() : Boolean;`
Descrição	Verifica se existe algum MEC1000 conectado e ligado na porta serial atual.
Parâmetros	Nenhum.
Retorno	Verdadeiro se houver um MEC1000 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 MEC1000 ligado à uma porta serial. Emite uma exceção caso não exista resposta.
Parâmetros	Nenhum.
Retorno	Nenhum.

SensorReadNow

Declaração	`Function SensorReadNow(sensor : Integer) : Integer;`
Descrição	Lê um canal de sensor. Existem 8 canais para sensores que podem ser lidos com esse comando. Abaixo uma lista com o número dos canais e conector onde se encontra cada canal.
Parâmetros	Sensor : Número do canal que deve ser lido.
Retorno	A leitura do canal.

Canal	Função
00	Conector 1: Primeira entrada analógica
01	Conector 1: Segunda entrada analógica
02	Conector 1: Terceira entrada analógica
03	Conector 1: Entrada digital
04	Conector 2: Primeira entrada analógica
05	Conector 2: Segunda entrada analógica
06	Conector 2: Terceira entrada analógica
07	Conector 2: 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.

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. O MEC1000 pode controlar 6 servo-motores. Cada servomotor pode ser deslocado em 8 posições diferentes, sendo que um valor 0 no parâmetro pos indica que ele permanecerá no início do curso e um valor 255 fará com que ele permaneça no final do curso. Além dessas, existem 6 posições intermediárias. 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.