Arduino

Arduino é uma plataforma de desenvolvimento *open-source*. Tal plataforma é composta tanto de hardware, na forma placas de microcontroladores, quanto de software, na forma de uma IDE.

Existem diversos modelos de hardware disponíveis, como o Arduino UNO, o Arduino MEGA, o arduino Nano e outros. Tais modelos apresentam como características em comum a presença de pinos de entrada e saída e um microcontrolador, de modo que tenham a capacidade de receber informações de sensores e transmitir informação para atuadores. As diferenças entre os modelos estão no número de pinos, modelo de microcontrolador, tamanho da placa, interface de comunicação serial, entre outras.

Neste projeto o modelo de placa utilizado foi o Arduino Pro Mini. As características deste modelo são:

|  |  |
| --- | --- |
| Microcontolador | ATmega328p |
| Voltagem de operação | 5V |
| Voltagem de entrada | 5 – 12V |
| Pinos de entrada/saída digital | 14 (sendo que 6 podem ter saída em PWM) |
| Pinos de entrada analógica | 8 |
| Corrente DC por pino de entrada/saída | 40mA |
| Memória Flash | 32kB |
| SRAM | 2kB |
| EEPROM | 1kB |
| Velocidade de Clock | 16MHz |
| Dimensões | 17,8mm x 33mm x 1.6mm |

Além das características listadas acima, um fator importante na decisão pelo uso deste modelo foi o custo. Atualmente o Arduino Pro Mini é vendido por $9,95 no fabricante oficial e por menos de $2 em fabricantes alternativos. Comparado ao valor de $24,95 do modelo Arduino UNO temos uma redução significativa nos custos.

A maior limitação do modelo se encontra no fato de que ele não possui interface USB integrada, sendo necessário o uso de um adaptador USB-serial externo para comunicação entre um computador e o módulo. Como a intenção do projeto é de instalar os módulos em pontos dispersos de uma residência e realizar comunicação via Bluetooth tal limitação é mitigada e não apresenta obstáculo para a execução do mesmo.

Função Connect

A função connect tem o objetivo de permitir a conexão entre duas unidades de forma confiável a partir de um único comando. Para atingir tal objetivo a função não apenas envia os comandos adequados para o módulo bluetooth como também recebe e analisa a resposta de modo a determinar se a conexão foi efetuada ou não.

Quando chamada, a função deve receber dois argumentos: “MAC” e “err”. O argumento “MAC” é a sequência de 12 caracteres alfanuméricos que identifica de forma única cada módulo HM10 e tem como objetivo permitir a identificação do módulo ao qual se deseja conectar. O argumento “err” é um valor inteiro que indica para a função se já houveram tentativas de conexão anteriores. O valor passado pelo usuário deve sempre ser “0”, sendo que outros valores são enviados pela própria função em caso de falha na conexão.

Após a execução da função existem três possíveis cenários, listados a seguir:

1. Módulos estão conectados. Função retorna inteiro “1”
2. Módulos não estão conectados pois o módulo alvo não foi encontrado. Função retorna inteiro “0”
3. Módulos não estão conectados pois o módulo local não foi capaz de interpretar o comando de conexão. Função retorna inteiro “-1”

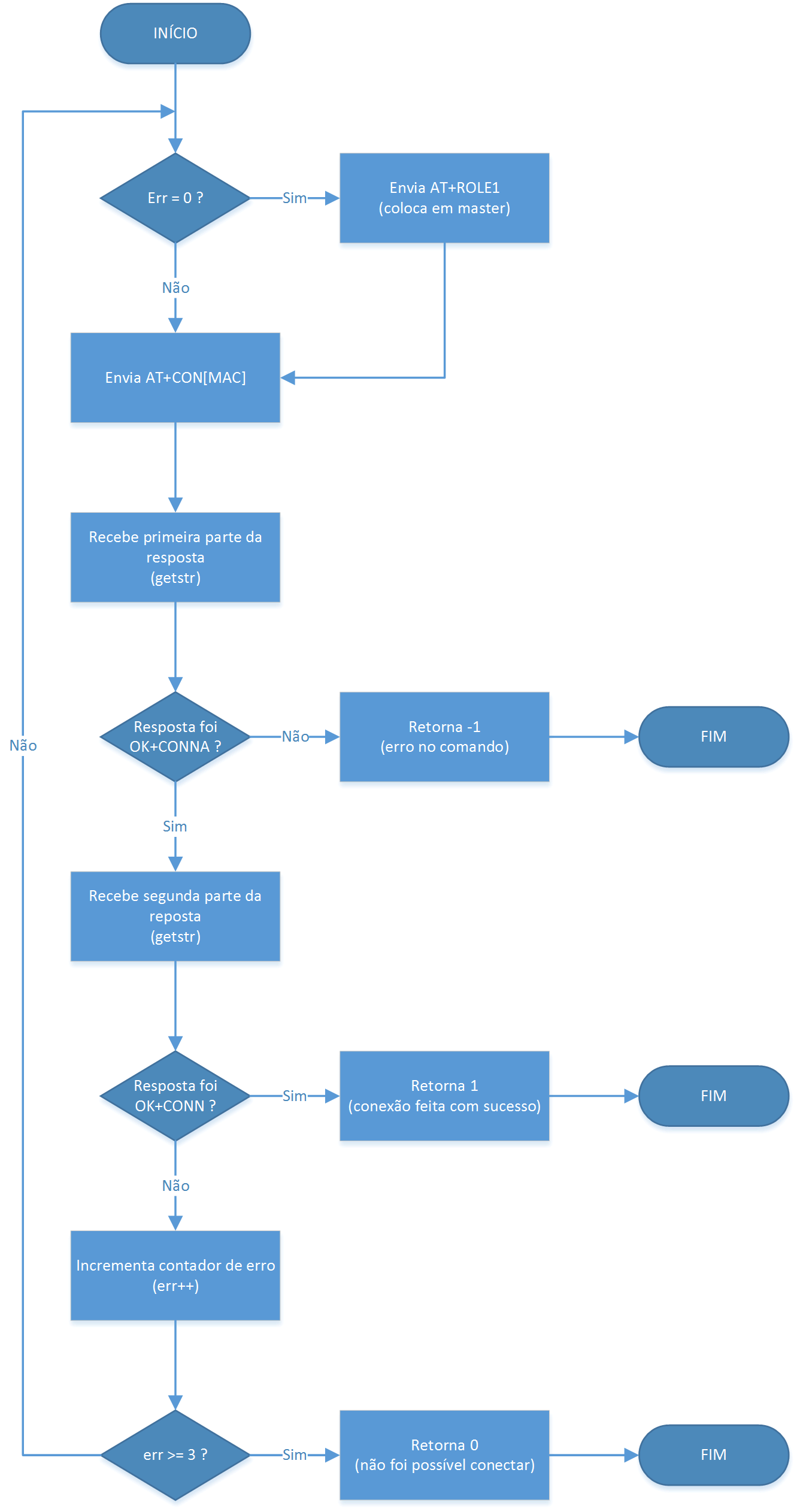
Um fluxograma representando o funcionamento da função pode ser observado na Figura X, na página seguinte.

Inicialmente a função verifica se está executando a primeira tentativa de conexão e em caso afirmativo envia o comando necessário para configurar o módulo como “master”, permitindo que o mesmo se conecte a outros módulos.

A seguir, o comando de conexão é enviado para o módulo, juntamente com o MAC do módulo ao qual se deseja conectar. Imediatamente a função recebe a analisa a resposta do módulo, de modo a identificar se o comando foi aceito e uma tentativa de conexão será efetuada (OK+CONNA). Em caso negativo a função retorna o valor “-1”, indicando o erro. Em caso afirmativo a função prossegue para o recebimento da segunda parte da mensagem, que indica se a conexão foi de fato efetuada.

O tempo de execução dessa etapa varia conforme a realização da conexão, sendo que o módulo permanece buscando por até 10s. A função está preparada para seguir com sua execução imediatamente após a chegada da segunda parte da resposta, independente do tempo transcorrido até sua chegada.

Após receber a segunda parte da resposta a função avalia se a conexão foi efetuada (OK+CONN) ou não. Se os módulos estiverem conectados a função retorna o valor “1” e encerra sua execução. Caso a conexão tenha falhado a função verifica quantas tentativas já foram feitas e decide entre tentar novamente, no caso de menos de 3 tentativas, ou retornar “0”, indicando que a conexão não foi possível.



Função Disconnect

A função disconnect atua de forma oposta à função connect, permitindo que duas unidades sejam desconectadas com um simples comando. A desconexão é feita de modo mais simples, porém ainda de forma robusta, analisando a resposta do módulo para determinar se a desconexão foi efetuada como desejado.

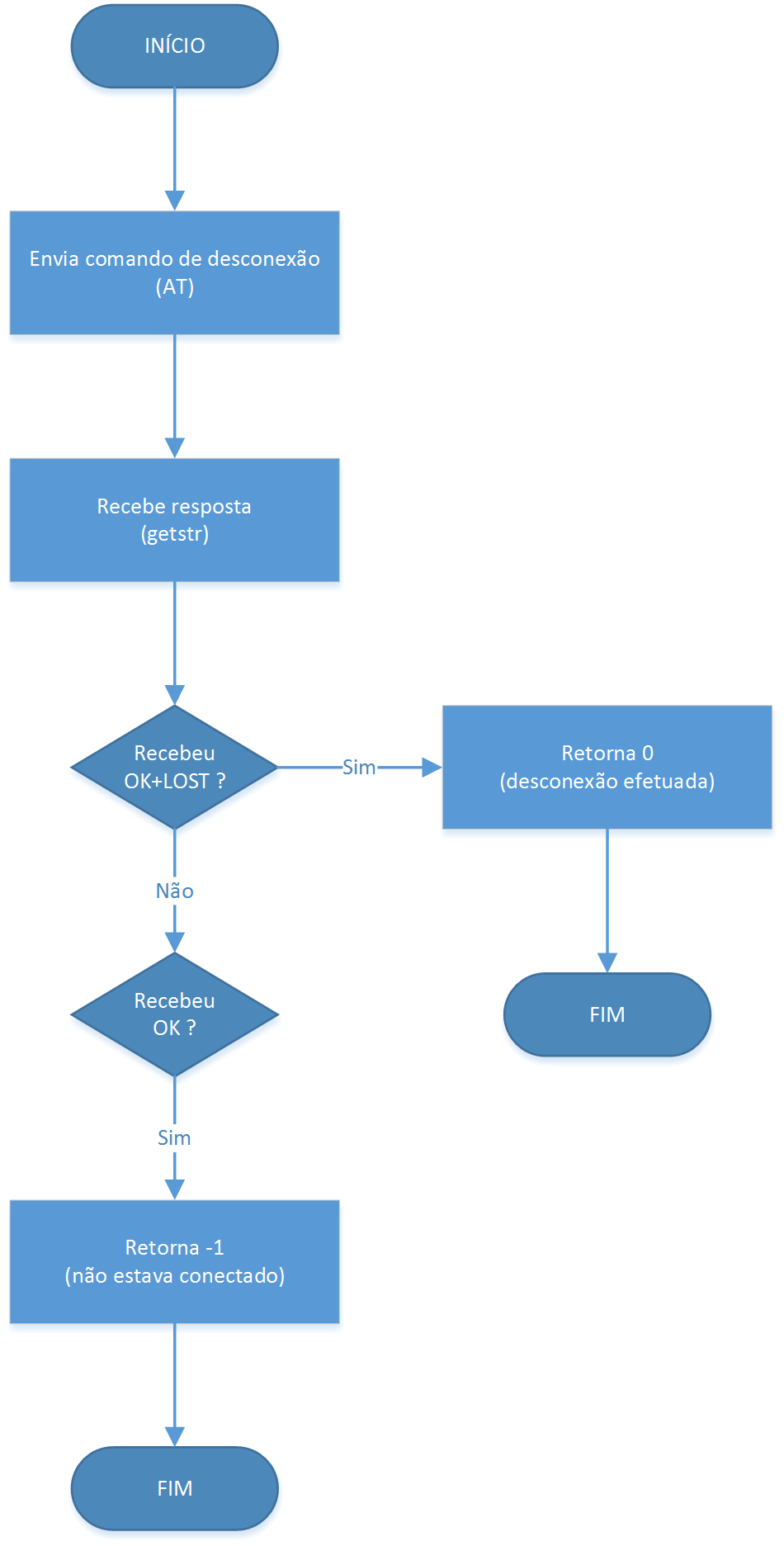
Por não necessitar de nenhum parâmetro específico a função disconnect não recebe nenhum valor como entrada. A execução da função independe de qual unidade esteja conectada, ou até mesmo de o módulo estar ou não conectado.

Após a execução da função existem dois possíveis cenários, listados a seguir:

1. Conexão terminada com sucesso. Função retorna inteiro 0
2. Módulo já não apresentava conexão antes da execução. Função retorna inteiro -1

Um fluxograma representando o funcionamento da função pode ser observado na Figura X, na página seguinte.

Quando chamada, a função envia ao módulo o comando “AT”, de modo a efetuar a desconexão. Em seguida, a função recebe a resposta do módulo e a analisa para verificar em qual das duas possibilidades ela se encontra. A primeira resposta, “OK+LOST”, indica que uma conexão estava presente e que a mesma foi devidamente terminada. Se verificada, a resposta leva o módulo a retornar o valor 0, indicando sucesso. A segunda resposta possível, “OK”, indica que o módulo não estava conectado mesmo antes da execução da função, o que pode indicar que algum erro tenha ocorrido na operação da unidade. Dessa forma, a função retorna o valor -1, indicando erro.



Função GetString

A função getstring tem como objetivo permitir que a unidade receba mensagens através da porta serial de forma simples. O uso da função é adequado quando sabe-se que a mensagem provém do módulo local, e não de outra unidade, tendo em vista que a mensagem é simplesmente interpretada como uma string, sem a formatação proposta.

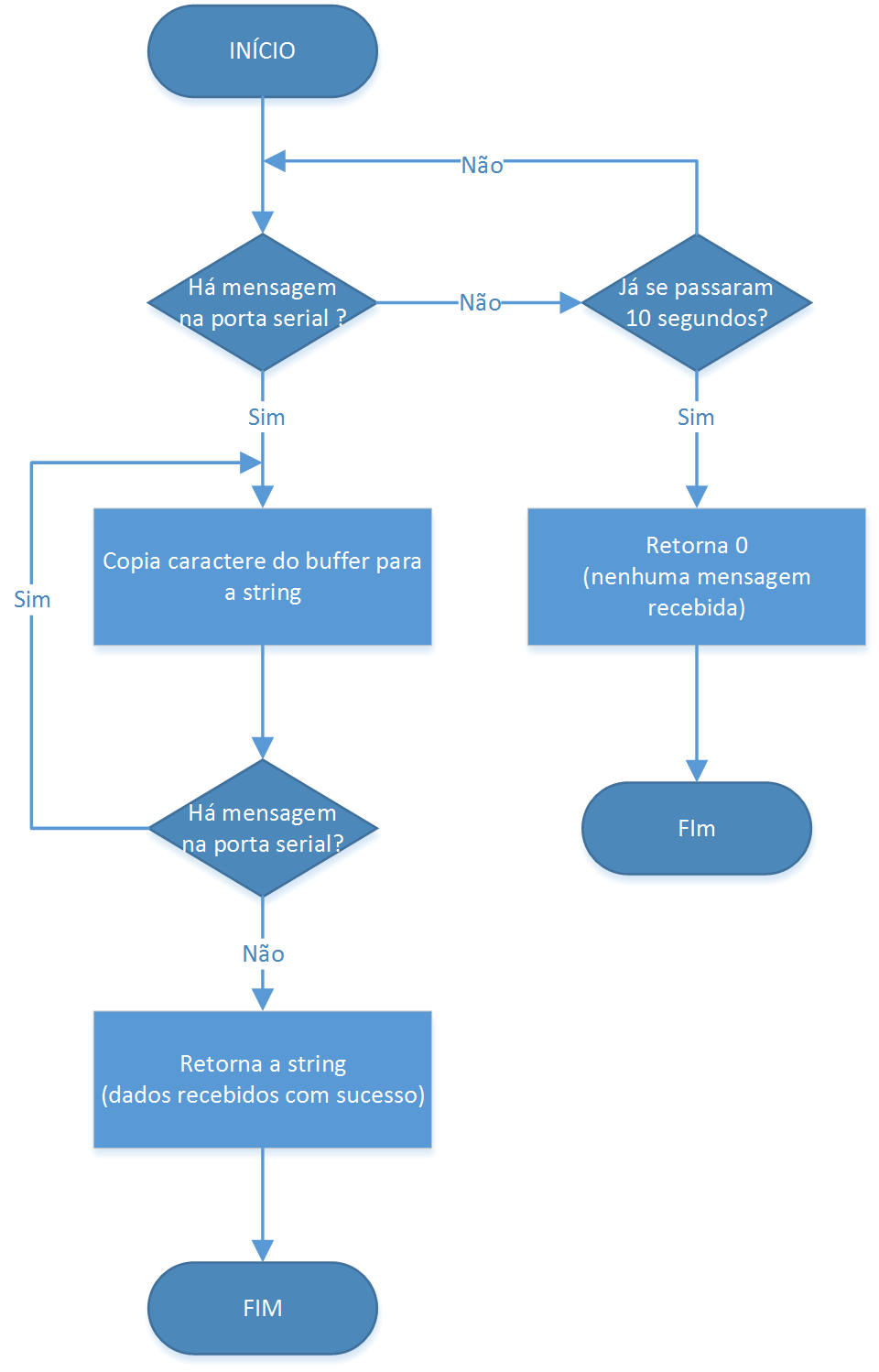
A função independe de parâmetros de entrada, tendo como retorno uma string de caracteres. Devido a limitações da comunicação serial tanto o arduino quanto a Raspberry Pi somente podem acessar um caractere por vez, de modo que a organização de todos os caracteres de uma mensagem permite que a mesma seja interpretada como um todo, facilitando a comunicação. Esse processo é usado múltiplas vezes dentro de outras funções, como connect e disconnect, sendo de grande importância para o funcionamento das mesmas.

Após a execução da função existem dois possíveis cenários, listados a seguir:

1. Informação recebida com sucesso. Função retorna a informação em forma de string.
2. Tempo limite de 10s excedido sem que informação seja recebida. Função retorna string com um único caractere 0.

Um fluxograma representando o funcionamento da função pode ser observado na Figura X, na página seguinte.

Ao ser chamada a função getstring inicia um contador e aguarda um de dois cenários: A porta serial acusa que existem bytes disponíveis no buffer ou o timer atinge 10 segundos sem que nenhuma mensagem seja recebida. No primeiro caso a função copia a mensagem byte a byte para uma string temporária, aplicando um pequeno delay entre as cópias para garantir que nenhum byte se perca, finalmente retornando a string como resultado. No segundo caso, a função entende que nenhuma mensagem foi recebida e retorna uma string com um único caractere, 0, de modo a indicar que nada foi recebido.



Função GetPacket

A função getpacket funciona de modo similar à função getstring, porém em vez de apenas receber os dados da porta serial e transferi-los para uma string ela organiza essa informação em um pacote de comunicação padronizado.

A organização leva em conta o tamanho da string recebida e o tamanho padrão das partes do pacote. Desse modo, a string é quebrada em quatro partes conforme o seguinte critério:

Caracteres 0 a 11 – ids (identificador do remetente)

Caracteres 12 a 23 – idr (identificador do receptor)

Caracteres 24 a length-1 – data (mensagem em si)

Caracteres length-1 a length – checksum

No critério acima “length” indica o tamanho da string recebida, de modo que a parte de dados da string é identificada como os caracteres do 24 ao penúltimo e o checksum é identificado como o último caractere da mesagem.

