**DESCRIÇÃO DO ALGORITMO QUE IMPLEMENTA O PROTOCOLO DE COMUNICAÇÃO**

Este documento tem por finalidade descrever como o algoritmo de comunicação entre o PC e o arduino foi implementado.Esse protocolo foi implementado utilizando os ambientes de desenvolvimento da IDE do arduino e o programa Visual Studio 2015 no formato *console aplication*. O algoritmo funciona de modo similar a um sistema de mestre e escravo, onde o mestre é o PC e o escravo é o arduino. Dessa forma, o PC enviará os comandos para o arduino executar. Tais comandos estão descritos no documento “PROTOCOLO DE COMUNICAÇÃO ENTRE O PC E O ARDUINO - DESCRIÇÃO”.

**FUNCIONAMENTO DO CÓDIGO DO ARDUINO**

Já que esse protocolo não possui byte de fim de mensagem, o sistema de comunicação no arduino foi feito com base na contagem de quantos bytes chegaram, e análise do primeiro byte para definição do comando enviado. Para isso, é utilizada uma variável como contadora. Essa variável é inicializada com valor zero, e sempre que um comando é executado, o valor zero é atribuído novamente a ela. O Fluxograma 1 representa o funcionamento do protocolo no arduino.

**Fluxograma 1.**

O programa começa esperando chegar algum byte vindo do PC. Quando chega o primeiro byte da mensagem, é feita a verificação de qual é o byte, pois a partir do primeiro já é possível saber qual o comando está sendo solicitado, ou se não foi um comando (algum byte flutuante).

**Comando SET:** Caso o primeiro byte seja um ‘W’, o arduino irá esperar chegar os 5 bytes restantes do comando SET. Após isso, irá verificar se o checksum está correto, caso esteja correto, será enviado um comando ACK para o PC, e será executado o comando SET normalmente. Caso o checksum não esteja correto, o arduino enviará um NACK para o PC, irá esperar pelo reenvio da string de comando SET. Enquanto o PC mandar mensagens com erros o arduino permanecerá nesse loop de enviar NACK e receber a string até que o checksum da mensagem venha correto.

**Comando POWER:** Caso o primeiro byte da mensagem seja um ‘P’, o arduino irá enviar um ACK para o PC e inverter o status atual da potência do sistema: se estiver ligado irá desliga-la, e se estiver ligado irá desliga-la.

**Comando STATE:** Caso o primeiro byte da mensagem seja um ‘S’, o arduino irá formar a string de STATE, irá enviar para o PC e aguardar a resposta, um ACK ou NACK. Caso receba um NACK, ele ficará em um loop de reenvio da string de STATE, aguardo da resposta do PC enquanto a resposta for NACK. Caso o PC responda ACK, o arduino irá esperar o próximo comando.

**Comando ACK e NACK:** A única possibilidade na qual o arduino receberá esses comandos do PC será quando o arduino enviar a string de STATE. Esses comandos serão tratados como foi descrito no **Comando STATE**.

Quando o primeiro byte de alguma mensagem for um byte diferente dos anteriores, o arduino não executará nenhum comando, e irá desativar a *flag “*endMsg”, a qual quando é verdadeira juntamente com a flag de algum comando (SET, STATE ou POWER), permite a execução do respectivo comando. Isso é feito para garantir que nenhum comando aleatório seja executado no momento indevido.

**FUNCIONAMENTO DO ALGORITMO DO PC**

Como já dito anteriormente, o PC irá agir como mestre enviando os comandos ao arduino. Para enviar esses comandos, foram criadas funções que serão executadas durante o loop de rastreio.

Para carregar um arquivo de texto com as efemérides, foi utilizada a biblioteca “Efem. Essa biblioteca possui uma classe com métodos como: armazenar efeméride, armazenar informações de passagens, obter valores de azimute e elevação da linha atual da tabela, verificar se ela acabou, entre outras.

Para realizar a comunicação serial entre o arduino e o PC, foi utilizada uma biblioteca obtida na internet no site Code Project, desenvolvida por Ramon de Klein. A biblioteca é bastante completa, permitindo a configuração de vários parâmetros como: taxa de transmissão, ter ou não bit de paridade, bit de parada, se o PC deve esperar determinada quantidade de bytes no buffer para fazer a leitura ou se ele deve simplesmente ler na hora que a função for chamada, entre outras opções. A biblioteca foi testada e funcionou perfeitamente para os objetivos do protocolo em questão.

O programa inicia abrindo a porta serial e configurando-a. O nome da porta serial utilizada é inserido no código do programa no formato *const wchar\_t*, e é um argumento da função serial::open(). Neste ponto, é criado o objeto da porta serial e feito algumas configurações como: a taxa de transmissão para 9600 bps, tamanho do byte para 8 bits, sem bit de paridade e 1 bit de parada. Além disso, é configurada a função de leitura da porta para esperar até que chegue a quantidade de bytes esperada, já que é um valor bem determinado (esse valor é um parâmetro da função de leitura). Feito isso, verifica-se se a porta serial realmente foi aberta. Caso tenha sido, o programa continuará normalmente. Caso contrário, o programa encerrará exibindo mensagem de erro.

Para realizar a comunicação serial foram elaboradas funções que implementam o protocolo de comunicação, as quais são utilizadas durante o “loop de rastreio”. Esse loop consiste em: receber as posições da linha atual da tabela de efemérides, através da função Efem::get(); aguardar a hora da efeméride carregada; envia-la ao arduino através da função send\_set(); receber o status atual da antena através da função send\_state(); avançar uma linha na tabela de efemérides e verificar se a tabela chegou ao fim através da função Efem::next(). Caso a tabela tenha acabado, o programa avançará para a próxima passagem, e caso todas as passagens tenham acabado, o programa será encerrado.

A seguir será feita uma descrição funcional das funções elaboradas para enviar os comandos do protocolo ao arduino.

**Função send\_set():** Essa função recebe como parâmetros: os ultimos valores de azimute e elevação carregado na função Efem::get(), a string onde será formado o comando set a ser enviado, e a referência para um contador de erros de comunicação. Com isso, a função codifica os ângulos em 16 bits, os separa em dois bytes cada um, forma a string de mensagem incluindo o checksum e envia para o arduino. Após o envio ela esperará o a resposta do arduino, que pode ser um ACK ou NACK. Caso seja um NACK, ela irá incrementar o contador de erros, reenviar a string de comando para o arduino, e irá esperar outra resposta dele. Permanecerá nesse ciclo até que o arduino responda com um ACK (ou seja, até que a mensagem seja enviada corretamente).

**Função send\_state():** Essa função recebe como parâmetros: uma string que receberá a string de state, referências para as variáveis que receberão a posição de azimute e elevação atual da antena, e a referência para o contador de erros. A função envia para o arduino o byte ‘S’ (comando STATE do protocolo), e espera para ler a string que ele enviar. Feita a leitura, o PC irá verificar se o checksum está correto. Caso não esteja, irá incrementar o contador de erros e enviar NACK para o arduino. Irá então esperar para ler novamente a string de STATE. Ficará nesse ciclo até que o checksum venha correto. Feito isso, a função decodificará os ângulos contidos nos bytes a1, a0, e1, e0, e decodificará o status do sistema contido no byte C.

**Função send\_power():** Essa função não recebe parâmetros. Após ser chamada, ela escreve o byte ‘P’ (comando POWER do protocolo) na porta serial e aguarda uma resposta (ACK ou NACK). Caso seja um NACK, ela irá reenviar o byte para o arduino, e irá esperar outra resposta dele. Permanecerá nesse ciclo até que o arduino responda com um ACK.

**Função verifica\_checksum():** Essa função faz a verificação se o checksum na string de STATE está correto. Caso esteja, ela retornará 0, caso contrário retornará 1. É utilizada na função send\_state().

**Função envia\_reconhecimento():** Essa função recebe como parâmetro o código de erro retornado na função verifica\_checksum(). Caso o valor seja zero, a função escreverá ACK na porta serial, caso contrário escreverá NACK.

**Função reconhecimento\_arduino():** Essa função recebe a resposta do arduino a algum comando enviado pelo PC, um ACK ou NACK. Caso a leitura seja um ACK, ela retornará 0, caso contrário retornará 1.