LNA LABORATÓRIO NACIONAL DE ASTROFÍSICA

Notas Técnicas do Laboratório Nacional de Astrofísica

Desenvolvimento ASCOM Alpaca para Dispositivos do Observatório do Pico dos Dias

Ramon Carlos Gargalhone

Orlindo Wagner Soares Pereira

LNA-NT-2024-27

Jun/2024

Desenvolvimento ASCOM Alpaca para Dispositivos do Observatório do Pico dos Dias

Ramon Carlos Gargalhone[1]

Orlindo Wagner Soares Pereira[1]

[1]*Laboratório Nacional de Astrofísica, rgargalhone@lna.br, osoares@lna.br*

Resumo: Este documento explora a aplicação do protocolo ASCOM Alpaca, uma extensão do padrão ASCOM utilizando conexão via rede entre dispositivos, para o controle de dispositivos astronômicos no Observatório do Pico dos Dias, com foco em componentes como o focalizador e a cúpula. O protocolo Alpaca utiliza técnicas RESTful e TCP/IP, permitindo a comunicação eficiente entre dispositivos ASCOM em redes modernas. A independência do Windows é uma vantagem significativa, tornando a tecnologia versátil e aplicável em diversas plataformas. A implementação desta tecnologia no observatório visa a total automação e integração, facilitando operações mais eficientes e precisas, além de abrir caminho para novas possibilidades e inovações tecnológicas no campo da instrumentação astronômica.

Abstract: *This document explores the application of the ASCOM Alpaca protocol, the newer ASCOM which uses network communication between apps and devices, for controlling astronomical devices at the Pico dos Dias Observatory, focusing on components such as the focuser and dome. The Alpaca protocol uses RESTful techniques and TCP/IP, enabling efficient communication between ASCOM devices in modern networks. Independence from Windows is a significant advantage, making the technology versatile and applicable across various platforms. The implementation of this technology at the observatory aims for full automation and integration, facilitating more efficient and precise operations, as well as paving the way for new possibilities and technological innovations in the field of astronomical instrumentation.*

Palavras-chave/Keywords: ASCOM, Alpaca, *drivers*, instrumentação, comunicação / *ASCOM, Alpaca, drivers, instrumentation, communication.*

Submetido em: Jun/2024

Revisado por:

Eder Martioli
Orlando Verducci Junior

Sumário

1 Introdução

Neste documento, será discutida a utilização do protocolo ASCOM Alpaca para controlar dispositivos astronômicos, como o focalizador e a cúpula no Observatório do Pico dos Dias. O protocolo ASCOM Alpaca é uma tecnologia inovadora que permite a comunicação e controle de equipamentos astronômicos de forma padronizada e eficiente.

O ASCOM (*Astronomy Common Object Model*) é um conjunto de padrões e interfaces criado para padronizar a comunicação entre equipamentos astronômicos. O Alpaca utiliza protocolos de rede amplamente utilizados para permitir que os aplicativos aproveitem as mesmas propriedades e métodos já definidos nas interfaces ASCOM.

A API Alpaca utiliza técnicas RESTful e TCP/IP, proporcionando uma interface robusta e moderna para que aplicativos e dispositivos ASCOM se comuniquem em ambientes de rede contemporâneos. Isso possibilita uma integração perfeita entre diferentes componentes de um sistema de controle de instrumentos astronômicos, promovendo a interoperabilidade e a flexibilidade.

Quando pensamos no futuro automatizado do Observatório do Pico dos Dias, sua total integração e robotização, a chegada de novos telescópios e novas técnicas a serem implementadas, este tipo de aplicação se torna bastante significativa, abrindo inúmeras possibilidades.

Além disso, uma das grandes vantagens do protocolo Alpaca é sua independência do sistema operacional Windows. Isso amplia ainda mais a versatilidade e a aplicabilidade da tecnologia, permitindo sua utilização em diferentes plataformas e ambientes operacionais, o que é essencial para a evolução e modernização dos observatórios astronômicos.

A implementação do protocolo ASCOM Alpaca no Observatório do Pico dos Dias representa uma etapa significativa, facilitando o controle remoto e automatizado dos equipamentos.

2 ASCOM

O ASCOM, que significa *Astronomy Common Object Model*, é um conjunto de padrões e interfaces desenvolvidos para a astronomia amadora e profissional. Ele foi criado para padronizar a comunicação entre equipamentos astronômicos, como telescópios, câmeras, montagens e outros dispositivos, e os aplicativos de controle que os operam. O ASCOM fornece uma base comum para desenvolvedores de *software* e fabricantes de equipamentos astronômicos, permitindo que eles criem aplicativos e *drivers* que possam interagir de forma harmoniosa e padronizada.

As principais características do ASCOM incluem:

1. Padronização: O ASCOM define um conjunto de interfaces padronizadas que os dispositivos astronômicos devem seguir. Isso torna possível para os desenvolvedores de *software* criar programas de controle que sejam compatíveis com uma ampla variedade de equipamentos.

2. Interoperabilidade: Os dispositivos que seguem os padrões ASCOM podem ser controlados por uma variedade de programas de *software* que também seguem esses padrões. Isso permite que os astrônomos usem seus programas favoritos para operar seus telescópios e acessórios.

3. Abstração de *Hardware*: O ASCOM fornece uma abstração de *hardware* que oculta os detalhes específicos de cada dispositivo, permitindo que os desenvolvedores de *software* criem interfaces consistentes e fáceis de usar.

3 ALPACA

O Alpaca é uma extensão do padrão ASCOM que foi desenvolvida para melhorar a conectividade e a acessibilidade dos dispositivos astronômicos. Em vez de depender de uma conexão direta com um computador local, o Alpaca possibilita a esses dispositivos a comunicação pela rede, o que é especialmente útil para observatórios remotos, observadores itinerantes e a automação de observações.

As principais características do Alpaca incluem:

1. Comunicação pela rede: O Alpaca permite que dispositivos astronômicos sejam controlados e monitorados pela rede, tornando possível operar observatórios de forma remota, seja através da internet ou de uma rede local.

2. Acesso a partir de várias plataformas: Como o Alpaca utiliza protocolos de rede padrão, ele pode ser acessado a partir de uma variedade de dispositivos e sistemas operacionais. Isso significa que você pode controlar seu telescópio a partir de um PC, *tablet* ou *smartphone*, independentemente do sistema operacional que estiver usando.

3. Segurança: O Alpaca inclui medidas de segurança para proteger os dispositivos astronômicos contra acessos não autorizados, garantindo que apenas usuários autorizados possam controlar e monitorar os equipamentos.

4 *Roadmap*

Este roteiro destina-se a orientar os desenvolvedores na criação de um *driver* Alpaca para um dispositivo ASCOM e uma única instância desse dispositivo, utilizando o exemplo de *driver* da cúpula, desenvolvido no Observatório do Pico dos Dias, como guia. Baseado no guia fornecido pela *ASCOM Initiative* [1].

4.1 Solicitação ao *Server*

Quando o servidor recebe solicitações HTTP/REST dos clientes, essas solicitações são encaminhadas para as classes *responder* da API do dispositivo. Toda documentação da API pode ser encontrada online [2]. Por exemplo, um aplicativo pode enviar a seguinte solicitação HTTP para verificar se o focalizador está atualmente em movimento:

```
/api/v1/focuser/0/ismoving?ClientID=xxx\&ClientTransactionID=yyy
```

Aqui, demonstramos como lidar com uma simples solicitação para verificar o *status* da propriedade *"IsMoving"* do focalizador. Essa abordagem permite que os clientes interajam com dispositivos astronômicos de maneira padronizada e remota por meio da infraestrutura Alpaca.

4.2 *GET Responder*

Uma vez que a solicitação seja considerada legal pelo pré-processador do Alpaca, o método **on_get**() do *responder* é chamado se a solicitação da API for um GET (obter o valor de uma propriedade). A primeira coisa que você verá é uma chamada para o dispositivo do *rotator* para verificar se ele está conectado. Se não estiver, é uma *NotConnectedException* do Alpaca. Em seguida, ele tenta ler a posição do dispositivo *rotator*. Se uma exceção for gerada no código do dispositivo, ela é capturada com um *except* genérico e resulta em uma *DriverException* do Alpaca. Caso contrário, ele usa um objeto *PropertyResponse* para construir o JSON de uma resposta de propriedade do Alpaca, incluindo o valor da posição obtida. Por exemplo:

```
class athome:
    def on_get(self, req: Request, resp: Response, devnum: int):
        if not dome.connected:
            resp.text = PropertyResponse(None, req,
                        NotConnectedException()).json
            return
        try:
            val = dome.at_home
            resp.text = PropertyResponse(val, req).json
        except Exception as e:
            resp.text = PropertyResponse(None, req,
                        DriverException(0x500, 'Athome fail', e)).json
```

Script 1: Código Python de solicitação para a propriedade AtHome da cúpula

```
    {
        "Value": true,
        "ClientTransactionID": 321,
        "ServerTransactionID": 1,
        "ErrorNumber": 0,
        "ErrorMessage": ""
    }
```

Script 2: Resposta da propriedade Alpaca

4.3 *PUT Responder*

As chamadas de método da API Alpaca, aquelas que fazem alguma coisa, usam o método HTTP PUT. O código abaixo mostra a resposta para o método *SlewToAzimuth*:

```

class slewtoazimuth:
    def on_put(self, req: Request, resp: Response, devnum: int):
        if not dome.connected:
            resp.text = PropertyResponse(None, req,
                            NotConnectedException()).json
```

```
            return
        az = get_request_field('Azimuth', req)
        try:
            azimuth = float(az)
        except:
            resp.text = MethodResponse(req,
                        InvalidValueException('Invalid az')).json
            return
        try:
            if 0 > azimuth > 360:
                resp.text = MethodResponse(req,
                            InvalidValueException('Invalid az')).json
                return
            dome.slew_to_azimuth(azimuth)
            resp.text = MethodResponse(req).json
        except Exception as e:
            resp.text = MethodResponse(req,
                        DriverException(0x500, 'Slew failed', e)).json
```

Script 3: Código Python para movimentação da cúpula

A principal observação a destacar aqui é que o parâmetro para o método é enviado pelo *body* do PUT como "*form data*". A função get_request_field() trata de obter o texto do parâmetro do PUT do corpo, incluindo requisitos de capitalização, gerando uma exceção *HTTPBadRequest* se algo der errado. A resposta ao método PUT retorna a resposta no formato JSON.

4.4 Erros em Tempo de Execução

Essas exceções são usadas para relatar problemas ou erros que podem ocorrer durante a comunicação ou operação dos dispositivos.

Seguindo o exemplo anterior, é possível observar como a exceção Alpaca *NotConnectedException* é tratada e retornada. O construtor *PropertyResponse* recebe o objeto de solicitação como seu primeiro parâmetro e, em seguida, utilizando a classe de exceção Alpaca *NotConnectedException* como segundo parâmetro. Isso permite obter o número de erro Alpaca e uma mensagem de erro associada, usados para construir a resposta JSON.

Aqui está uma breve descrição de algumas das exceções Alpaca mais comuns.

4.4.1 *NotConnectedException*

Esta exceção é acionada quando um dispositivo não está conectado ou não está disponível para comunicação. Geralmente, isso ocorre quando um aplicativo tenta interagir com um dispositivo que não está ativo.

4.4.2 *DriverException*

A *DriverException* é uma exceção genérica que abrange uma variedade de erros ou falhas que podem ocorrer durante a operação de um dispositivo Alpaca. Ela é usada para relatar erros não específicos que não se encaixam em outras categorias de exceções.

4.4.3 *ValueNotSetException*

Esta exceção ocorre quando uma propriedade de um dispositivo Alpaca não possui um valor válido ou não foi configurada corretamente. Por exemplo, se uma propriedade de posicionamento não foi inicializada, ela pode gerar essa exceção.

4.4.4 *InvalidOperationException*

Essa exceção indica que uma operação solicitada não é permitida no estado atual do dispositivo. Por exemplo, tentar mover um telescópio quando ele já está em movimento pode acionar essa exceção.

4.4.5 *PropertyNotImplementedException*

Esta exceção é usada quando uma propriedade específica não está definida para o dispositivo Alpaca. Por exemplo, um dispositivo pode não oferecer suporte a certos recursos ou configurações, resultando nessa exceção quando essas propriedades são acessadas.

4.4.6 *ActionNotImplementedException*

Semelhante à exceção de propriedade não implementada, esta exceção ocorre quando uma ação específica não é realizável pelo dispositivo. Isso pode acontecer quando um aplicativo tenta executar uma operação que o dispositivo não pode realizar.

É crucial que qualquer problema encontrado pelo seu dispositivo seja comunicado de volta ao aplicativo por meio de uma das exceções Alpaca. Para a maioria dos problemas, a exceção *DriverException* é a escolha adequada para essa finalidade.

5 Dispositivos do OPD

Os dispositivos instalados no OPD utilizam sistema embarcado com os microcontroladores C8051. Os comandos são padronizados no seguinte formato:

CTR DISP CMD = PARAM

1. **CTR**: Controlador (AH, DEC, TUBO, CUP, MEADE)
2. **DISP**: Dispositivo a ser controlado (EIXO, FOCO, TRAPEIRA, etc)
3. **CMD**: Ação a ser executada (MOVER, LER, ABRIR, etc)
4. **PARAM**: Parâmetro do comando, caso houver (MOVER = 80)

Existe também o dispositivo comum a todos os controladores, denominado **PROG**.

1. **STATUS**: Retorna a *string* de estado do microcontrolador, no formato *value******0...000
2. **PARAR**: Interrompe qualquer ação que está em execução no momento
3. **RESET**: Reseta o microcontrolador via *software*

5.1 Cúpula

Nesse artigo, será explorado o dispositivo Cúpula, utilizado como referência no desenvolvimento Alpaca para os dispositivos do OPD, foi testado utilizando a cúpula do Telescópio Meade 40 do OPD.

Este item abordará o desenvolvimento em Python do *drive* e será dividido em *Device* (comunicação com o dispositivo), *Interface* (*Middleware* - interação entre servidor e Device) e *Application* (Servidor).

5.1.1 *Device*

Nesta seção, apresentaremos uma visão geral do código Python desenvolvido, entender a estrutura geral do código e como ele se encaixa no contexto do controle dos dispositivos astronômicos do OPD. O código é projetado para se comunicar com a cúpula e permitir operações como monitoramento de estado, controle de trapeira e movimentação em azimute.

O código está organizado em uma classe chamada Dome, que encapsula todas as funcionalidades relacionadas ao seu controle. Uma parte crítica do código está relacionada à comunicação serial, fundamental para enviar comandos à cúpula e receber informações de estado.

Funcionalidades Principais:

1. **Monitoramento do estado da cúpula**: Isso permite que o código saiba a posição atual da cúpula, se a trapeira está aberta ou fechada e se a cúpula está em movimento.
2. **Controle da trapeira**: É possível abrir e fechar a trapeira.
3. **Movimentação da cúpula**: O código permite que a cúpula seja direcionada para uma posição específica em azimute.

Tratamento de Exceções: Uma parte importante do código é o tratamento de exceções. Isso garante que o código seja capaz de lidar com situações inesperadas, como falhas na comunicação ou erros no controle da cúpula. Exceções personalizadas, como *RuntimeError*, são usadas para relatar problemas específicos que podem ocorrer durante a operação.

Customização: Dependendo da aplicação ou de particularidades, o código pode ser adaptado.

Abaixo, está o método *close_shutter*(), responsável por fechar a trapeira. Vale notar que ele envia comandos utilizando a estrutura **CTR DISP CMD**, padrão dos dispositivos do OPD.

```
def close_shutter(self)-> None:
        if not self._can_set_shutter:
            raise RuntimeError('Cannot set Shutter')
        ret = 'ACK' in self._write(f"MEADE TRAPEIRA FECHAR\r")
```

```
        if not ret:
            ret = 'ACK' in self._write(f"MEADE TRAPEIRA FECHAR\r")
            if not ret:
                self._shutter_status = 4
                raise RuntimeError('Dome close FAIL!')
        self._lock.acquire()
        self._shutter_status = 3
        self._lock.release()
```

Script 4: Método para fechar a trapeira

5.1.2 Interface - *Middleware*

Nesta seção, apresentaremos uma visão geral do código Python desenvolvido para criar uma API Alpaca para controle do dispositivo Cúpula. O código serve como um intermediário entre os clientes e o dispositivo real, permitindo que os usuários controlem e monitorem remotamente.

O código é organizado em um conjunto de classes que correspondem a diferentes ações e propriedades relacionadas à cúpula. Cada classe define como responder a solicitações HTTP para acessar ou modificar essas ações e propriedades.

Funcionalidades Principais:

1. **Ações de comando**: Responde a comandos como abrir ou fechar o obturador da cúpula, encontrar a posição inicial (*home*), mover a cúpula para uma determinada posição em azimute, entre outros.

2. **Propriedades**: Permite a leitura de informações sobre a cúpula, como a descrição do dispositivo, a versão do *driver*, a posição atual do obturador, a capacidade de encontrar a posição inicial (*home*) entre outras.

3. **Gerenciamento de conexão**: Controla o estado de conexão do dispositivo.

Tratamento de exceções: O código implementa tratamento de exceções para lidar com erros que podem ocorrer durante a execução. Isso ajuda a garantir que a API seja robusta e fornece mensagens de erro significativas em caso de problemas.

Roteamento de solicitações HTTP:

O código faz uso da biblioteca Falcon para rotear solicitações HTTP para as classes apropriadas que lidam com ações e propriedades específicas da cúpula. Cada classe define como responder a solicitações HTTP, retornando as informações necessárias ou executando as ações solicitadas.

Abaixo, está a classe *findhome*, responsável por enviar a cúpula para a posição Home.

```
class findhome:
    def on_put(self, req: Request, resp: Response, devnum: int):
        if not dome.connected:
            resp.text = PropertyResponse(None, req,
```

```
                        NotConnectedException()).json
            return
        try:
            dome.find_home()
            resp.text = MethodResponse(req).json
        except Exception as e:
            resp.text = MethodResponse(req,
                        DriverException(0x500, 'Home failed', e)).json
```

Script 5: A Classe FindHome movimenta a cúpula para a posição *Home*

5.1.3 *Application - Server*

Este código é um aplicativo Python que usa o *framework* Falcon para criar um servidor *web* para controlar dispositivos Alpaca. O aplicativo cria rotas para cada tipo de dispositivo (*focuser*, *rotator* e *dome*) e lida com solicitações HTTP para controlar esses dispositivos. Ele também inclui configuração de *log*, manipulação de exceções e funcionalidades de descoberta.

O código abaixo inicializa as rotas da API com base em um módulo de dispositivos específico. Ele examina o módulo em busca de classes que são consideradas controladores e cria rotas para cada uma delas.

```
    app = App()
    init_routes(app, 'focuser', focuser)
    init_routes(app, 'dome', dome)

    # Initialize routes for Alpaca support endpoints
    app.add_route('/management/apiversions', management.apiversions())
    app.add_route(f'/management/v{API_VERSION}/description', management.
    description())
    app.add_route(f'/management/v{API_VERSION}/configureddevices',
    management.configureddevices())
    app.add_route('/setup', setup.svrsetup())
    app.add_route(f'/setup/v{API_VERSION}/focuser/{{devnum}}/setup',
    setup.devsetup())
    app.add_route(f'/setup/v{API_VERSION}/dome/{{devnum}}/setup', setup.
    devsetup())

    app.add_error_handler(Exception, falcon_uncaught_exception_handler)
```

Script 6: Inicialização das rotas da API

6 Validação

A validação Alpaca é realizada por meio da ferramenta ConformU, desenvolvida pela ASCOM *Initiative*.

6.1 *ConformU*

O *ConformU* (*Conform Universal*) é uma ferramenta versátil que realiza testes em dispositivos Alpaca em todas as plataformas e em *drivers* COM na plataforma Windows.

Os dispositivos Alpaca que suportam o protocolo *discovery*, que varre a rede procurando dispositivos Alpaca, são automaticamente apresentados para seleção, mas também é possível configurar manualmente a URL, o número da porta e o número do dispositivo para direcionar dispositivos que não são automaticamente encontrados na rede.

A aplicação pode ser executada no navegador padrão do usuário como uma aplicação de página única *Blazor* ou como uma utilidade de linha de comando sem interface gráfica.

Esta ferramenta realiza todas as operações que o dispositivo deve executar. No caso da cúpula, ele faz as operações de movimentação a cada 45 graus. Verifica posição e estado da trapeira, posição Home, funções da trapeira e demais condições de operação.

6.2 Resultados

Os resultados dos testes são relatados tanto na interface do navegador quanto no console, além de serem registrados em um arquivo de *log* legível por humanos e em um arquivo JSON estruturado legível por máquinas. O *ConformU* também oferece uma visualização dos dispositivos Alpaca e dos dispositivos ASCOM descobertos por meio do Mapa de Descoberta Alpaca.

Aqui estão algumas linhas do *log* final de validação:

```
15:15:28.632 OpenShutter       OK       Shutter opened successfully
15:16:14.263 FindHome          OK       Dome homed successfully
15:16:19.281 Park              OK       Optional member returned a
   MethodNotImplementedException error.
15:16:19.284 SetPark           OK       Optional member returned a
   MethodNotImplementedException error.
15:16:19.285
15:16:19.285 Post-run Checks
15:16:19.286 DomeSafety        INFO     Attempting to close shutter...
15:16:22.715 DomeSafety        OK       Shutter successfully closed
```

Script 7: Resultado de validação ConformU

Ao final da execução de todas as tarefas de teste, é informado se a validação obteve sucesso ou se há problemas a serem corrigidos.

No errors, warnings or issues found: your driver passes ASCOM validation!!

Quando traduzido: Nenhum erro, aviso ou problema encontrado: seu *driver* passa na validação ASCOM!!

Pronto, validação concluída.

7 Aplicação

Este código Python permite controlar uma cúpula usando a biblioteca Alpaca, permitindo que seja possível a automatização do posicionamento e o movimento da cúpula de forma programática. Ele utiliza o pacote **Alpyca**, uma biblioteca cliente API Python para ASCOM Alpaca, produzida pela ASCOM *Initiative*.

A seguir, estão os principais pontos explicados:

```
import time
from alpaca.dome import Dome  # Importando a classe Dome
from alpaca.exceptions import AlpacaException
from alpaca import discovery

# Procurando dispositivos Alpaca disponiveis na rede local
devices = discovery.search_ipv4()
print(devices)

# Criando uma instancia do Dome com o IP e numero do dispositivo
dome = Dome('127.0.0.1:5555', 0)

try:
    # Estabelecendo conexao com a cupula
    dome.Connected = True
    print(f'Conectado a cupula: {dome.Name}')
    print(f'Descricao da cupula: {dome.Description}')

    print(f'Posicao da cupula: {dome.Azimuth}')

    # Movendo
    nova_posicao = dome.Azimuth + 30  # Exemplo: mover 30 graus
    dome.SlewToAzimuth(nova_posicao)

    # Aguardando ate que a cupula termine de se mover
    while dome.Slewing:
        print(f'Posicao atual da cupula: {dome.Azimuth}')
        time.sleep(1)

    print('Movimentacao concluida.')
    print(f'Nova posicao da cupula: {dome.Azimuth}')

except AlpacaException as e:
    print(f'Falha na movimentacao da cupula: {str(e)}')

finally:
    # Desconectando
    print('Desconectando...')
    dome.Connected = False
```

Script 8: Aplicação via Alpyca

O *script* 9 apresenta a implementação de uma rotina de *tracking* onde, dadas as coordenadas de ângulo horário e declinação da montagem do telescópio, são feitos os cálculos de conversão do sistema de coordenadas equatorial para o sistema de coordenadas horizontal (azimute e altura). Para este exemplo, foi adotado como referência uma montagem altazimutal, uma vez que, diferente de montagens equatoriais, não necessitam de cálculos mais complexos de geometria em relação à cúpula.

```
import math

def calc_alt_az(ha, dec):
    """Calcula Azimute e Altitude"""
    RAD2DEG = 180 / math.pi # Converte radianos para graus
```

```
    DEG2RAD = math.pi / 180.0 # Converte graus para radianos
    latitude = -22.53 # Latitude do Observatorio
    H = ha * 15 # Converte hora para graus

    # Calculo da Altitude
    sinAltitude = (math.sin(dec * DEG2RAD)) * (math.sin(latitude *
   DEG2RAD)) + (math.cos(dec * DEG2RAD) * math.cos(latitude * DEG2RAD) *
    math.cos(H * DEG2RAD))
    altitude = math.asin(sinAltitude) * RAD2DEG #altura em graus

    # Calculo do Azimute
    y = -1 * math.sin(H * DEG2RAD)
    x = (math.tan(dec * DEG2RAD) * math.cos(latitude * DEG2RAD)) - (math
   .cos(H * DEG2RAD) * math.sin(latitude * DEG2RAD))
    azimuth = math.atan2(y, x) * RAD2DEG

    # Correcao para valores negativos
    if (azimuth < 0) :
        azimuth = azimuth + 360

    return altitude, azimuth

def _get_telescope_position():
    """Le posicao dos eixos do telescopio"""
    ...

def adjust_position():
    # Le posicao dos eixos Angulo Horario e Declinacao
    ha, dec = _get_telescope_position() # Exemplo
    altitude, azimute = calc_alt_az(ha, dec)
    # Levando em consideracao um telescopio altazimutal
    # sem os calculos de geometria das equatoriais
    dome.SlewToAzimuth(azimute)

# Loop do programa enquanto conectado
while dome.Connected:
    ...
    # Variavel de controle tracking
    if _flag_tracking:
        adjust_position()
    ...
    time.sleep(30)
```

Script 9: Exemplo de aplicação do *tracking*

8 Conclusão

O uso do protocolo ASCOM Alpaca para controlar dispositivos astronômicos como o focalizador e a cúpula no Observatório do Pico dos Dias oferece uma maneira conveniente e padronizada de operar e automatizar as atividades de observação. A flexibilidade e a facilidade de controle remoto são vantagens significativas que contribuem para uma experiência de observação mais eficiente.

Para obter mais detalhes sobre a implementação específica no Observatório do Pico dos Dias, consulte a documentação fornecida com os *drivers* Alpaca para os dispositivos individuais.

Referências

[1] Introduction to alpaca drivers. https://ascom-standards.org/alpycadevice/introduction.html. Accessed: 2023-09-26.

[2] Ascom alpaca device api. https://ascom-standards.org/api. Accessed: 2023-09-26.