Problemstellung

- Ein Teleskop mit Montierung, Kamera, Filterrad, Motorfokus und Kuppel/Dach fernzusteuern. Darüber hinaus auch Abläufe automatisieren. Bildersequenzen erstellen, Zugriffe auf Datenbanken und Internet. Sowie der Download der Ergebnisse zu ermöglichen.
- Teleskope die selbständig Beobachtungen durchführen und die Ergebnisse am nächsten Tag zur Verfügung stellen. Die Allskyskamera ist so ein Beispiel, die natürlich auch zum Equipment gehört.
- Kommerzielle und freie Programme wie SharpCap oder N.I.N.A. ermöglichen dies in der Vergangenheit. Den Durchbruch brachte das Platesolving, die astrometrische Lösung der Aufnahme, es werden die Koordinaten in RA und DE des Bildes ermittelt. Das Programm ASTAP kann u.a. das Platesolving durchführen. Damit weiß man wo das Teleskop hinzeigt und kann es ggfs. korrigieren. ASTAP wird in den beiden oben genannten Programmen benutzt.
- Den Zugriff über das Internet geht heute mit teils kostenfreien Programmen wie RemoteDesktop-Verbindung, AnyDesk oder Teams.
- Voraussetzung ist eine stabile und schnelle Internetverbindung.

Anforderungen

- Eine eigene speziell für die eigenen Anwendungen eine Lösung selber zu entwickeln.
- Für jemanden der „nur“ Astrofotografie macht, reicht N.I.N.A. vollkommen aus.
- Mit dem Programm Carte du Ciel kann man auch Kometen, Kleinplaneten und Veränderliche aufsuchen. Dieses Programm wird u.a. von SharpCap benutzt.
- Werden weitere Kataloge benötigt reichen diese beiden Programme nicht aus.
- Die Eigenentwicklung sollte plattformunabhängig und quelloffen sein.

Umsetzung

Skriptsprache Python

- Plattformunabhängige Skriptsprache mit Interpreter
- Kostenlos
- Viele Anwendungen: Astronomie, Numerik, Datenanalyse, Graphiken plotten usw.
- Python ist die „Hülle“ die die Komplexität verhüllt.
- Realisierung z.B. von Kameratreibern muss C/C++ mit installiert werden.
- Auch SharpCap und N.I.N.A. ? haben eine Python-Schnittstelle

Entwicklungsumgebung

- Visual Studio
- Pycharm
- Eclipse?

Beispiel

Seestar_alp

Für den Seestar S50 gibt es ein Steuerungsprogramm mit dem PC/Notebook in Python geschrieben. Dazu muss das Seestar im „Station Mode“ ins Heimnetzwerk eingebunden werden.

- Eine ausführbare Version für den reinen Anwender mit Win/Linux/Raspberry/MacOS System. Nach dem Start ein von seestar_alp.exe wird im Internetbrowser die Adresse localhost:5432 eingegeben und der SimpleSeestarController steuert dann den Seestar, oder auch mehrere davon.
- Eine Entwicklerversion als Framework um neueste Entwicklungen mitzukriegen und selber zu entwickeln, z.B. eine andere Benutzeroberfläche.

Seestar mit N.I.N.A. gesteuert.

Seestar S50 unterstützt auch die Alpaca-Schnittstelle. Das ist ASCOM für WLAN und läuft unter Windows/Linux. Alle Steuerprogramme die Alpaca unterstützen sollten auch damit den Seestar steuern. Mit N.I.N.A. geht das. Dazu muss der Seestar äquatorial mit EQ-Plattform aufgestellt sein und lässt sich dann fernsteuern. N.I.N.A. läuft nur unter Windows und hat nur Deepsky-Objekte in der Datenbank.

Fazit

- Mit dem Seestar_alp kann man seine eigene Steuerungsprogramm erstellen und anpassen. Als Steuerrechner geht der Raspberry (Linux) oder ein Mini-PC unter Windows.
- Mit dem Seestar hätte man die kleinste Remote-Sternwarte zum Testen. Für Erfahrungen für andere Teleskope der Sternwarte.
- Hier lassen sich Abläufe wie Zugangsberechtigung, IT-Sicherheit und automatische Abläufe testen.

Pyobs

-Quelle https://docs.pyobs.org/en/latest/

Pyobs ist ein modulares Framework für professionelle Teleskope entwickelt von der Uni Göttingen. Es werden auch Komponenten aus dem Amateurbereich verwendet, wie z.B. ASI, QHY und SBIG.

Übersicht

pyobs ist ein Python-Framework für den Aufbau und Betrieb **autonomer, robotischer oder entfernter astronomischer Observatorien**. Es bietet die Software-Grundlage für die Integration und Steuerung von Teleskopen, Kameras, Kuppeln, Wettersensoren und Planungssystemen – alles in einer einheitlichen und modularen Umgebung.

Die Designphilosophie des Frameworks betont *Flexibilität* und *Erweiterbarkeit*: Anwender konfigurieren Observatoriumssysteme deklarativ (typischerweise in YAML) durch die Kombination von wiederverwendbaren Modulen für die Hardwaresteuerung, Datenverarbeitung, Terminplanung und Überwachung. Pyobs ermöglicht es einem Teleskop, vollautomatische Beobachtungszyklen durchzuführen, von der Kalibrierung bis zur Datenerfassung und Archivierung.

Das Hauptprojekt, pyobs-core, bietet die zentrale Infrastruktur und abstrakte Schnittstellen. Zusätzliche Repositorys (z. pyobs-asi, pyobs-gui, pyobs-alpaca) bestimmte Hardwaretreiber oder Benutzeroberflächen implementieren.

Hauptmerkmale

- **Hardware-Abstraktion und** Treiber — Einheitliche Schnittstellen für Kameras, Teleskope, Kuppeln, Filterräder und andere Geräte.
- **Automatisierung und Planung** – Ermöglicht den Roboterbetrieb, die Überwachung von Aufgaben und die Integration mit externen Schedulern wie dem Las Cumbres Observatory.
- **Datenerfassung und -verarbeitung** – Unterstützt die Bildkalibrierung, Photometrie, Quellenextraktion und Katalogerstellung durch konfigurierbare Verarbeitungspipelines.
- **Verteilter Betrieb** — Module können über mehrere Maschinen laufen und über ein vernetztes Messaging-System kommunizieren.
- **Konfigurationsbasierte Architektur** – Observatory-Setups werden in YAML-Dateien definiert, wodurch die Notwendigkeit einer benutzerdefinierten Programmierung reduziert wird.
- **Erweiterbarkeit** – Anwender können eigene Module entwickeln oder bestehende für neue Instrumente, Sensoren oder Observatoriumslogik erweitern.
- **Integrations-Ökosystem** — Companion-Pakete bieten Unterstützung für spezifische Kameras (ASI, SBIG, QHYCCD), Fokussiermotoren, GUIs, Wetterschnittstellen und Astrometriedienste.

Architektur

Pyobs folgt einem **modularen, ereignisgesteuerten** Design. Jedes *Modul* stellt eine Komponente des Observatoriums dar - beispielsweise einen Kameracontroller, einen Scheduler oder einen Wettermonitor. Module kommunizieren asynchron und tauschen Befehle, Daten und Ereignisse über eine Netzwerkschnittstelle aus.

Die Architektur wird von mehreren Prinzipien geleitet:

- **Trennung von Anliegen:** Hardwaretreiber, Steuerungslogik und Datenverarbeitung werden als eigenständige Module implementiert.
- **Asynchroner Betrieb:** Beobachtungsworkflows (z.B. Belichtungen, Auslesen, Kuppelbewegung) sind nicht blockierend und gleichzeitig.
- **Abstraktionsschichten:** Standardisierte Schnittstellen definieren Verträge für Hardware und Services und sorgen für Hardwareunabhängigkeit.
- **Deklarative Konfiguration:** Modulverbindungen und Verhaltensweisen werden über Konfigurationsdateien und nicht über Code definiert.

Ökosystem und Repositorys

Die Pyobs-Organisation auf GitHub hostet eine Familie von Repositorys:

- **pyobs-core** — Das Hauptrahmenwerk, das Schnittstellen, Modulverwaltung und Kommunikationsinfrastruktur bietet.

- **pyobs-asi**pyobs-asi, **pyobs-qhyccd**pyobs-qhyccd, **pyobs-sbig**pyobs-sbig — Treiber für verschiedene astronomische Kameras.
- **pyobs-aravis** — Unterstützung für Aravis-kompatible Industriekameras.
- **pyobs-zwoeaf** — Modul für ZWO EAF Fokusmotoren.
- **Pyobs-Pilar** — Schnittstelle zum Pilar Teleskop-Steuerungssystem.
- **pyobs-alpaca** — ASCOM Alpaca-Brücke für Interoperabilität mit anderer Software.
- **pyobs-gui** — Grafische Benutzeroberfläche zur Steuerung und Überwachung von Observatorien.
- **pyobs-astrometry**Pyobs-Astrometrie — Webservice-Wrapper für *astrometry.net* Solve-Feld-Operationen.

Zusammen bilden diese Komponenten ein komplettes, erweiterbares Observatoriums-kontrollsystem.

Anwendungsfälle

Pyobs können in einer Vielzahl von astronomischen Kontexten eingesetzt werden:

1. **Vollständig autonome Teleskope** Führen Sie ganze Beobachtungsnächte automatisch aus: Wählen Sie Ziele aus, nehmen Sie Bilder auf, führen Sie Kalibrierungen durch und archivieren Sie Daten.
2. **Fernbedienung** Ermöglichen Sie es menschlichen Bedienern, Observatoriumskomponenten von entfernten Standorten über Netzwerkschnittstellen oder GUIs zu steuern.
3. **Instrument Prototyping** Integrieren Sie schnell neue Instrumente oder Geräte durch die Implementierung kleiner Schnittstellenmodule.
4. **Datengesteuertes Feedback** Integrieren Sie Echtzeit-Bildanalysen (z. B. Quellenerkennung oder Qualitätsmetriken), um die Planung oder Instrumentensteuerung zu beeinflussen.
5. **Pädagogische oder kleine Observatorien** Führen Sie vereinfachte Setups für studentische Projekte oder Forschung mit minimalem Overhead aus.

Einschränkungen und Outlook

Obwohl es mächtig ist, bleibt Pyobs ein sich entwickelndes System:

- Konfigurationsdateien können in großen Installationen komplex werden.
- Hardwarespezifisches Verhalten kann weiterhin eine benutzerdefinierte Handhabung erfordern.
- Die vollständige Testabdeckung über alle Module hinweg ist ein kontinuierlicher Aufwand.
- Für Hochdurchsatzumgebungen kann eine Leistungsabstimmung erforderlich sein.

Trotz dieser Vorbehalte reifen Pyobs weiterhin als robuste Open-Source-Plattform für die Automatisierung und Steuerung von Observatorien.

Teleskope

Pyobs läuft derzeit auf fünf Teleskopen auf der ganzen Welt:

- **MONET/North** (McDonald Observatory, Texas) und **MONET/South** (SAAO, Südafrika)
- **IAG 50cm** (Göttingen, Deutschland)
- **IAG VTT** (Göttingen, Deutschland)
- **ROTSE Namibia** (Hess-Seite, Namibia)

Literatur

- Projektseite: https://github.com/pyobs/
- Kern-Framework: https://github.com/pyobs/pyobs-core
- Dokumentation: https://docs.pyobs.org/
- Wissenschaftliche Referenz: *Grenzen in der Astronomie und Weltraumwissenschaften* (2022), „pyobs: Ein modulares Kontrollsystem für astronomische Observatorien“