EPICS-BASED TELEGRAM INTEGRATION FOR CONTROL AND ALARM HANDLING AT TEX FACILITY

D. Moriggi[*], S. Pioli, F. Cardelli, C. Di Giulio, P. Ciuffetti
INFN - Laboratori Nazionali di Frascati, Italy

Abstract

TeXbot is a Telegram bot developed in python language used to notify in asynchronous way event in TEX (TEst stand for X-band) facility at Frascati National Laboratories. The application has been realized making use of framework such as telepot and pysmlib, to interface with Telegram and with EPICS environment respectively.

The bot make able the user to subscribe to multiple topic in order to be automatically notified in case of different set up of the machine or when an interlock occurs on a single component. Furthermore the user can request accurate information about subsystem of the accelerator by simply make use of special commands and token in Telegram app.

INTRODUCTION

In any industrial context it is present the need to have information delivered in the fastest way as possible.

In this environment the Telegram TeXbot solution can be deployed, since it can control and handle alarm when a subsystem goes in an interlock state or certain event occurs to the system.

The bot is developed at National Laboratory of Frascati and running at TEX (EuPRAXIA TEst stand for X-band) facility and is capable of sending automatic live messages through Telegram app displaying information on particle accelerator system state.

The final user can start joining private chat and once inserted the correct password can initiate conversation with bot asking for precise information about the TEX facility. Through text recognition the bot can answer by sending information directly or can eventually submit different kind of menu used to let user choose the appropriate content.

Moreover the consumer can handle subscription to specific categories in order to receive accurate notifications on certain devices in real-time, or also ask for the chart representing the history for a particular component in a given time span.

FRAMEWORK

This system is realized thanks to the integration of multiple framework each devoted to a different type of functionalities. The first to be mentioned is Telepot [1], a Python repository downloadable from pip, the packet manager for Python, that encapsulate Telegram API (Application Program Interface) [2] making able to build up and control the bot.

The second Python library used to create state machine capable of handling events during system operations and making able the bot to communicate with EPICS is pysmlib [3], which is built on top of PyEpics and therefore guarantees a perfect integration with EPICS Channel Access.

The last is flask [4] used to realize the back-end component responsible to make available API to retrieve and manipulate data coming from the EPIC S Archiver Appliance [5, 6] used to store data of the TEX facility.

ARCHITECTURE

The main four components that constitute the overall architecture are shown in Fig. 1 and it will be described in the following:

- the EPICS Channel Access
- TeXbot, the Telegram bot
- Flask back-end for retrieval
- EPICS Archiver Appliance

The first fundamental requirement for this kind of setup is the need of EPICS CA (Channel Access), that once a connection it's established with it, make the system able to retrieve live information on PVs (Process Variables) of the TEX facility.

Then obviously comes the TeXbot that could be thought and refer to it as a listener in the framework, because it is distributed in reading-mode respect to data, interrogating the CA and sending raw information to Telegram user.

Another component is represented by the back-end service that behave in its own container and it is responsible to exposes APIs used to retrieve manipulated information coming not only from live data, but even from the archived ones.

The last independent service that runs in its dedicated container is the EPICS Archiver Appliance, which is devoted to handle a layered data storage by saving information in three different area depending on its date of acquisition.

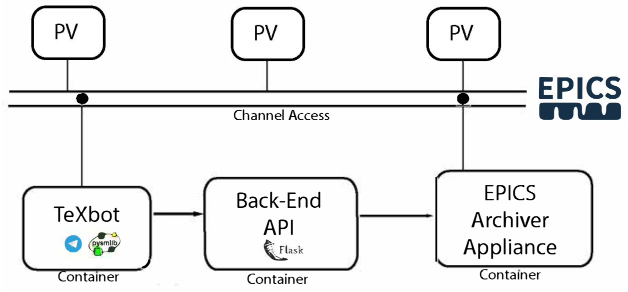


Figure 1: Architecture.

[*] daniele.moriggi@lnf.infn.it

FUNCTIONALITIES

TeXbot

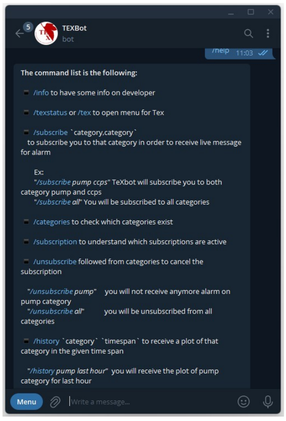
Once registered to the TeXbot inside Telegram App, it is possible to start interacting with it. Then the first thing to do it is to authenticate yourself by submitting a pre-shared password to the application, in order to be authorized to send commands and visualize data.

The interaction with bot is performed through the implementation of token recognition, to perform common operation of visualizing general information, retrieve data, register to particular categories or produce plot for certain variables.

Moreover multiple menu are implemented to make a collection of different portion of the data, helping the end user to visualize detailed information on the particle accelerator of TEX facility.

Recognized Tokens In the following it will be described a list of token words that can be sent to the TeXbot in order to perform a specific action.

- /password - used as a first command when starting the TeXbot, authenticate the user to the bot
- /info - shows general information on the system and the developer
- /help - shows an help to make clear how use the commands (Fig. 2a)
- /tex - it shows a menu, let the user the possibility to choose among different collection of information (Fig. 2b):
  - Status Tex - shows overall information of the facility, modulator status, vacuum status on components
  - Source Status RF - shows precise information about RF (Radio Frequency) source
  - LLRF - shows information about Low Level Radio Frequency signal
  - Vacuum - shows another sub-menu for each subsystem: klystron, Wave Guide and RF-loads to monitor the respective vacuum
  - Safety - show the operational state and if the patrol has been performed
- /categories - shows which categories of information can subscribe to
- /subscribe /unsubscribe - open another sub-menu used to subscribe/unsubscribe to categories and receive/cancel live message when certain event occurs (Fig. 2d)
- /subscription - shows active user's subscriptions (Fig. 2d)
- /history <category> last <timeSpan> - it will provide a downloadable chart representing the history of the PVs belonging to selected category in a given time span (hours, days, months) (Fig. 2c)



/help 11:03

The command list is the following:

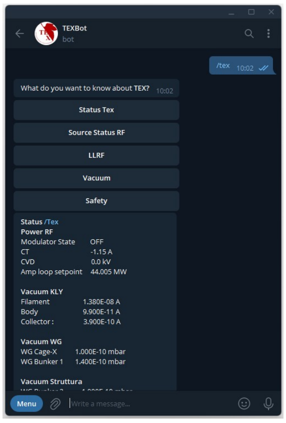
- /info to have some info on developer
- /texstatus or /tex to open menu for Tex
- /subscribe \`category,category\`
  to subscribe you to that category in order to receive live message for alarm

  Ex:
  "/subscribe pump ccps" TeXbot will subscribe you to both category pump and ccps
  "/subscribe all" You will be subscribed to all categories
- /categories to check which categories exist
- /subscription to understand which subscriptions are active
- /unsubscribe followed from categories to cancel the subscription

  "/unsubscribe pump" you will not receive anymore alarm on pump category
  "/unsubscribe all" you will be unsubscribed from all categories
- /history \`category\` \`timespan\` to receive a plot of that category in the given time span

  "/history pump last hour" you will receive the plot of pump category for last hour

(a) /help command



/tex 10:02

What do you want to know about TEX? 10:02

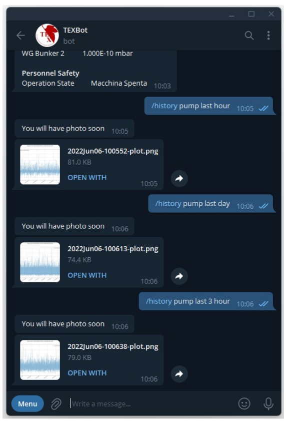
```
Status /Tex
Power RF
Modulator State      OFF
CT                   -1.15 A
CVD                  0.0 kV
Amp loop setpoint    44.005 MW

Vacuum KLY
Filament         1.380E-08 A
Body             9.900E-11 A
Collector :      3.900E-10 A

Vacuum WG
WG Cage-X      1.000E-10 mbar
WG Bunker 1    1.400E-10 mbar

Vacuum Struttura
```

(b) /tex command



```
WG Bunker 2      1.000E-10 mbar

Personnel Safety
Operation State      Macchina Spenta    10:03
```

/history pump last hour 10:05

You will have photo soon 10:05

2022Jun06-100552-plot.png 81.0 KB OPEN WITH 10:05

/history pump last day 10:06

You will have photo soon 10:06

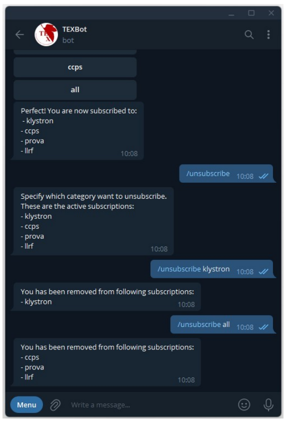
2022Jun06-100613-plot.png 74.4 KB OPEN WITH 10:06

/history pump last 3 hour 10:06

You will have photo soon 10:06

2022Jun06-100638-plot.png 79.0 KB OPEN WITH 10:06

(c) /history command



Perfect! You are now subscribed to:
- klystron
- ccps
- prova
- llrf
10:08

/unsubscribe 10:08

Specify which category want to unsubscribe.
These are the active subscriptions:
- klystron
- ccps
- prova
- llrf
10:08

/unsubscribe klystron 10:08

You has been removed from following subscriptions:
- klystron
10:08

/unsubscribe all 10:08

You has been removed from following subscriptions:
- ccps
- prova
- llrf
10:08

(d) /subscription command

Figure 2: Screenshot.

In Fig. 2 are presented some features of TeXbot.

EPICS State Machine The core functionalities responsible for the interconnection with EPICS Channel Access are delegated to pysmlib, that is devoted to realize a FSM (Finite State Machine).

At the start the FSM is in a basic state and with high priority continuously poll specific PVs belonging to a category to be supervisioned given that represent an important part of the system that can raise an alarm.

As long as no alarm occurs it waits for message arriving from TeXbot and when this occurs, the FSM will compose the message to be redirected to the user who made the request.

The logic implemented by the state machine can be seen on the flowchart in Fig. 3. From the idle state the first check is about interlock, if one or more occurs the system then evaluate if anyone is subscribed to that category to which ilk belong, if so the message is sent else the system return to idle state.

If no ilk occurs, the state evaluate if any messages arrived from Telegram, if so then the data are retrieved from backend or Channel Access and sent to telegram user in form of message or image.

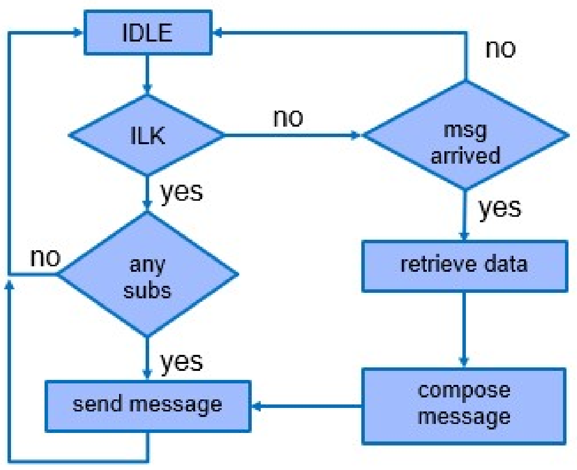


Figure 3: Logic flowchart.

Back-End

To perform more complex operations on data, a back-end service has been created making use of Flask, that exposing APIs to other services can extend in a powerful way the possibility to aggregate, manipulate information and control subsystem. This kind of service behaves as a bridge establishing connection among different component in the architecture and making this framework fully extendable in terms of services and capabilities.

The back-end is mainly used, at this stage, for the interaction with the EPICS Archiver Appliance, that is used to store data and from which can be query the database to retrieve raw data.

EPICS Archiver Appliance

The EPICS Archiver Appliance by Shankar et al. is the subsystem that has been deployed in its own container and it is responsible for archiving pre-defined set of PVs, following for each of them a precise policy for storaging.

In order to store data, the EPICS Archiver Appliance uses ProtocolBuffers (PB) serialization mechanism by Google, moving the information from the closest layer, the Short Term Store (STS), dedicated to last data in order of arrival, to the farther layer, the Long Term Store (LTS), dedicated to the oldest data produced. Moreover it also make you able to query the database through its APIs.

CONCLUSION

TeXbot can represent a possible solution integrated and developed upon Telegram and EPICS, respectively one of the most downloaded application on mobile application market nowadays and one of most used framework for controlling system, solving different kind of problem in a given technological research and industrial environment.

It in fact can handle in real time alerting messages and can be extremely expandable under a newly presented architecture, representing a powerful solution deployable in a relative low time and guaranteeing high reliability and stability.

REFERENCES

[1] Telepot, `https://telepot.readthedocs.io/en/latest/`

[2] Telegram, `https://core.telegram.org/`

[3] D. Marcato *et al.*, “Pysmlib: A Python Finite State Machine Library for EPICS”, presented at the ICALEPCS’21, Shanghai, China, Oct. 2021, paper TUBL05, unpublished.

[4] Flask, `https://flask.palletsprojects.com/en/2.1.x/`

[5] EPICS Archiver Appliance, `https://slacmshankar.github.io/epicsarchiver_docs/index.html`

[6] M. V. Shankar, L. F. Li, M. A. Davidsaver, and M. G. Konrad, “The EPICS Archiver Appliance”, in *Proc. ICALEPCS’15*, Melbourne, Australia, Oct. 2015, pp. 761–764. doi:10.18429/JACoW-ICALEPCS2015-WEPGF030