ladikvadim / **Telegram-Bot**

Telegram bot implementation using LabVIEW

19 commits | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Find File | Clone or download

ladikvadim Fixed bug (Issue: Groups do not work #1) ... Latest commit 8cc2dbb on 26 Jan

CommandProcessing	modified and refactored some VI's	7 months ago
Docs	Added Russian Readme	2 years ago
FPGA	Added Project files	2 years ago
FPGABitfiles	Added Project files	2 years ago
Libraries	Added Project files	2 years ago
OpenWeatherMap	modified and refactored some VI's	7 months ago
PC	modified and refactored some VI's	7 months ago
ProjectFiles	modified and refactored some VI's	7 months ago
Telegram	Fixed bug (Issue: Groups do not work #1)	7 months ago
LICENSE	Initial commit	2 years ago
README.md	Update README.md	2 years ago
SimpleTelegramBot.aliases	modified and refactored some VI's	7 months ago
SimpleTelegramBot.lvlps	Added Project files	2 years ago
SimpleTelegramBot.lvproj	modified and refactored some VI's	7 months ago
main.vi	Added Project files	2 years ago

README.md

Telegram-Bot-EN

These examples demonstrate the use of TelegramBotAPI in LabVIEW. Beginners will be useful to start read Bots: An introduction for developers.
The description in Russian is available by reference.

Getting Started

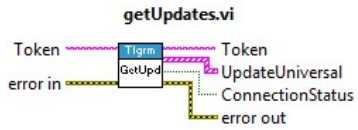
Software and hardware

Example **TelegramBotSimpleExample** requires installed LabVIEW 2015+ and Internet access.
Example **myRIO+TelegramBOT Demo** in addition, requires the presence of modules Real-Time and FPGA, and hardware - myRIO 1900. Instead of myRIO with a small change of the project, you can use any device on the RIO platform (sbRIO, CompactRIO).

Methods description

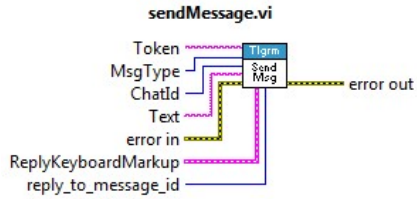
A bot receives updates using method **getUpdates** (long polling) and sends messages using method **sendMessage**. Currently, only these two methods work. Each update is an array of JSON-serialized *Update* objects.

getUpdates



- Input terminals:
  - Token - bot token.
  - error in - input error cluster.
- Output terminals:
  - Token - bot token.
  - UpdateUniresal - array of *Update* objects.
  - ConnectionStatus - connection status indicator.
  - error out - output error cluster.

sendMessage



- Input terminals:
  - Token - bot token.
  - MsgType - message type (Text, ReplyKeyboardMarkup, ReplyKeyboardRemove).
  - ChatId - active chat indicator.
  - Text - message text.
  - ReplyKeyboardMarkup - a custom keyboard with reply options.
  - ReplyToMessageId - parent message ID (in group chats).
  - error in - input error cluster.
- Output terminals:
  - error out - output error cluster.

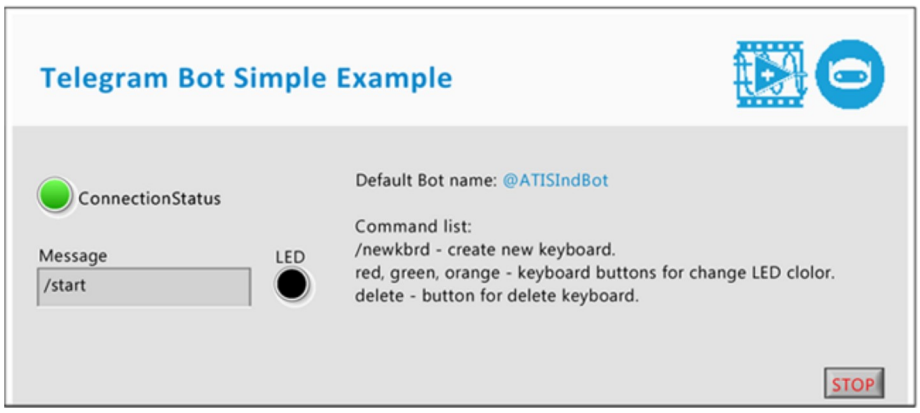
Bot examples

For a correct operation of each example:

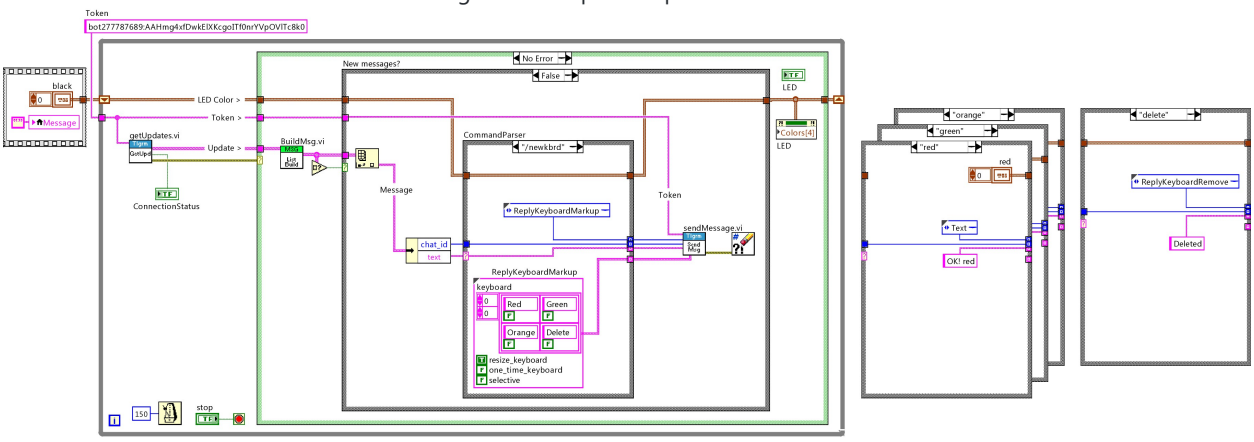
- create a bot in Telegram and write in the field "Token" a token, which sent @Botfather. The token must start with "bot", for example "**bot275887199:AAHmg4xfDwkElXKcgoITf0nrYVpOVlTc8k0**"
- On your device, start a private chat with the bot created in the previous step.
  - For testing, you can use the ***@ATISIndBot*** bot.

TelegramBotSimpleExample

Consider the example of the simplest bot in LabVIEW. To do this, open **TelegramBotSimpleExample.vi**. The following figures show its front panel, which is simple and does not require explanations and a block diagram.



TelegramBotSimpleExample.vi Front Panel



TelegramBotSimpleExample.vi Block Diagramm

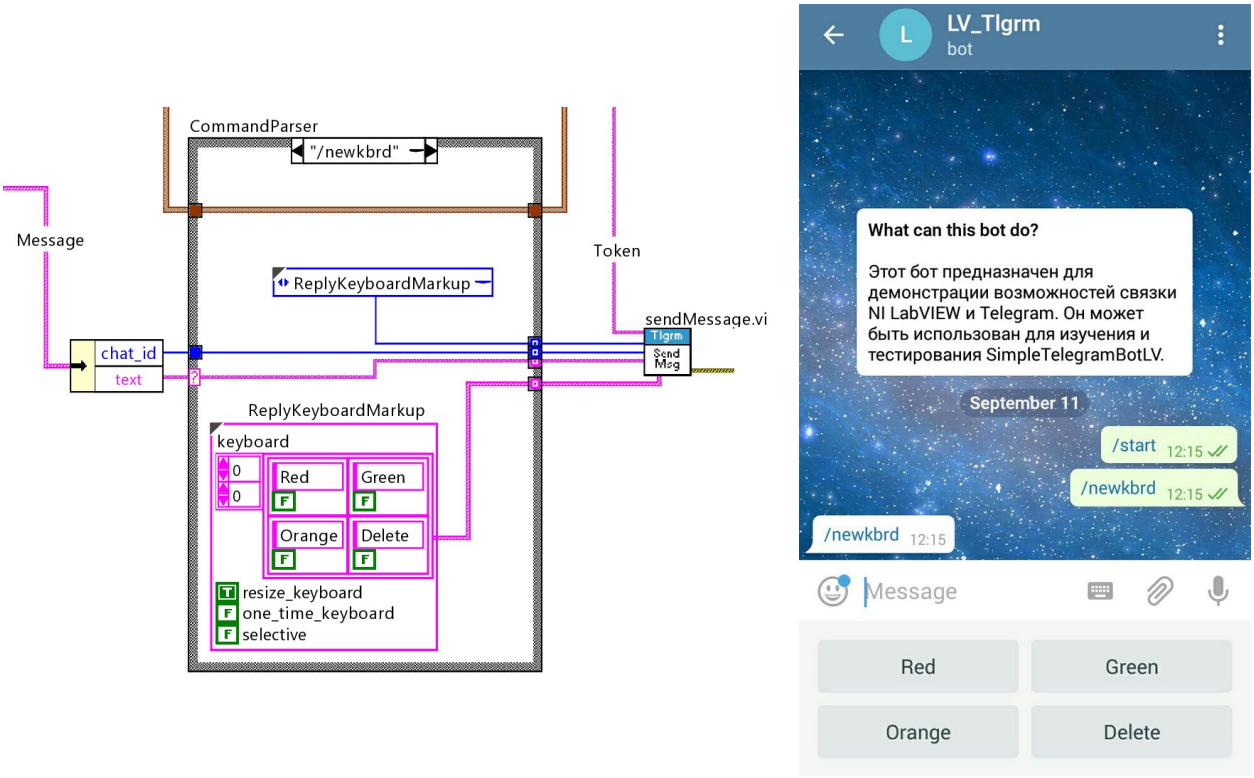
Algorithm of the bot is very simple. First, an initialization takes place. Then, in an infinite loop with a specified period, occur poll the Telegram servers for new updates. SubVI BuildMsg.vi converts JSON Update objects to messages. If there are new messages, then their text is compared to variants of the Case structure. At the last stage, the reply message is sent to the active chat.

How to create a custom keyboard.

In order to send a keyboard to the chat you need to perform two actions:

1. Specify the type of message to send as ***ReplyKeyboardMarkup***.
2. Fill the ***ReplyKeyboardMarkup*** cluster.
   To delete the keyboard, you must specify the message type as ***ReplyKeyboardRemove***.

The following figure shows an example of creating a custom keyboard consisting of four buttons (Red, Green, Orange, Delete) and result that is displayed in chat.



Example of creating a custom keyboard:
A — block diagram fragment that is responsible for creating and sending the keyboard.
Б — result displayed in chat.

myRIO+TelegramBOT Demo

This is a more complex example of implementing the logic of bot, although the meaning of his work remains the same. In this example, using messages in telegram, you can change the glow of custom LED1-3 LEDs on myRIO 1900 and perform some other functions.

To change LED glow, use next commands::
`/led#,Period(ms/us),DutyCycle(%)` - changes the character of user LED (1-3) . For example, ***/led1,1000,50***. For LED1, 2 the period is set in ms, forLED3 in us.
`/led#` (without parameters), # - LED number (1-3). For example, ***/led1***.As a result, bot will return user's keyboard with several buttons, allowing you to on/off the LED.

To obtain weather data in a certain city, use a command::
`/getweather,City,Region` , где City — required parameter, Region — optional parameter. For example ***/getweather,miami,us*** — bot will return a message with weather data in Miami, USA.

To get the current time of myRIO, use a command:
`/gettime` — the bot will return a message about myRIO current time.

Authors

- **Vadim Ladik** - *Initial work*

License

This project is licensed under the MIT License - see the LICENSE file for details

Acknowledgments

- Hat tip to anyone who's code was used

- Inspiration
- Sorry for my bad English
- etc