Terminal++ for Robot Researcher Using C#

Sudip Chakraborty[1] & P. S. Aithal[2]
[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India
OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in
[2]ViceChancellor, Srinivas University, Mangalore, India
OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

Subject Area: Computer Science.
Type of the Paper: Experiment based Research
Type of Review: Peer Reviewed as per |C|O|P|E| guidance.
Indexed In: OpenAIRE.
DOI:
Google Scholar Citation: IJAEML

How to Cite this Paper:
Sudip Chakraborty & Aithal, P. S., (2021). Terminal++ for Robot Researcher Using C#. *International Journal of Applied Engineering and Management Letters (IJAEML)*, *5*(2), 175-182. DOI:

International Journal of Applied Engineering and Management Letters (IJAEML)
A Refereed International Journal of Srinivas University, India.

Crossref DOI : https://doi.org/10.47992/IJAEML.2581.7000.0111

Terminal++ for Robot Researcher Using C#

Sudip Chakraborty[1] & P. S. Aithal[2]

[1]Post-Doctoral Researcher, College of Computer science and Information science, Srinivas University, Mangalore-575 001, India

OrcidID: 0000-0002-1088-663X; E-mail: sudip.pdf@srinivasuniversity.edu.in

[2]ViceChancellor, Srinivas University, Mangalore, India

OrcidID: 0000-0002-4691-8736; E-Mail: psaithal@gmail.com

ABSTRACT

Purpose: *Sometimes our robot researcher needs a terminal program to exchange the data with the robot or automation device. Nevertheless, the readily available terminal program lacks some functionality that is most relevant to the researcher. We feel that a featured rich terminal program can handle lots of communication overhead for the researcher and relieve them from repetitive and time-consuming tasks. In mind for this, we researched and developed a utility program. We added extra features like automatic send, change dynamic data, etc., so our robot researcher can test the system communication better. In this paper, we demonstrated the utility program in detail. It is built using C#, which is under the Microsoft dot net framework. The code is uploaded to GitHub. Anyone can download and use it. It can be customized for their need. All used classes are available in .cs format.*

Design/Methodology/Approach: *This is the software utility program built by the dot net framework of Microsoft visual studio. It has a graphical user interface (GUI) and some object classes. It has a serial and ethernet interface to test the channel. Once the medium is selected, the application will send whatever is written in the input text box. The Data sending may be an automatic or manual process. In manual mode, after typing the command, we need to press the "Enter" key to send the data. In automatic mode, it will send automatically within the preset interval. The transmit and receive content is displayed inside the list box.*

Findings/results: *sometimes, our project goes into a critical phase. We need to have good tools to overcome the situation immediately. This is a helpful tool to trace the communication-related issue. Using this tool, we can observe the outgoing and incoming data traffic. The robot researcher can use it for their communication-related debug purposes.*

Originality/Value: *Using this terminal program, our robot researcher will get lots of benefits where readily available utility programs cannot provide them. It has some unique features like automatic sending, changing dynamic content, etc. It has a serial and ethernet interface channel so that most of the device communication can be debugged through this interface software. It is entirely free and open source. Anyone can download and use it for personal as well as commercial purposes.*

Paper Type: *Experiment-based Research.*

Keywords: Command Terminal, robot debug interface, serial communication interface, ethernet client terminal, Enhanced terminal program. Open source terminal program

1. INTRODUCTION :

We use the terminal program to send the data to the device and receive the response from the connected device. For a one-shot, We type and send through the command line. But for repetitive static and dynamic content, sending data creates lots of issues. It takes time to send repetitively. For dynamic range, send data leads prone to error only through command terminal. To avoid the scenario, most of the time, we create the software based on our needs. To develop software from scratch is time-consuming and costly. Our robot researcher may not afford time and cost. So the only way is to use the currently available free terminal program. The most presently available terminal program has some lack of features that are most relevant for debugging purposes. A few of them are as below:

(1) The used command list is not visible easily.

(2) We navigate commands through the up/down key only.

(3) Automatic send is not available.
(4) A limited number of packet savings options are available.

Through our research, we developed a utility program, which has the following features :-

(1) Transmit and receive packets are displayed into a separate list box
(2) All incoming and outgoing packets are shown with a timestamp.
(3) The savings option for All log data.
(4) Automatic send to the connected device.
(5) The scan data is selected from command history by double click.
(6) The scanning interval is settable.
(7) All application configuration is saved by pressing the “Save config” button or automatically saved on application close.

2. RELATED WORKS :

X. Lu, W. Liu et al. their works Based on sending the command to the robot over the Internet of things, Through the TCP socket, the service-oriented robots get the command via a wireless medium. For different activities [1]. B. Shimano worked on an automatic manufacturing robot. It is interacted with both a supervisory control network and with various sensory devices. It includes high-level computer programming facilities that allow professional programmers to develop special-purpose application programs which simplify the operator interface [2]. L. Gong *et al*. their paper proposes a human-robot cooperation scheme with wearable augmented glasses. The commands like STOP, PULLOVER, TURN-LEFT are compiled and programmed as simplified patterns to control the robot body/hand pantomime. Experimental results show that the proposed methodology and control scheme is feasible in real-time applications with high real-time performance of less than 0.5s latency and open possibilities of easing the traffic jam via simultaneously scheduling multiple traffic cop robots [3]. R. H. Taylor and D. D. Grossman, in their paper, describe the architecture of a robot system designed both to work in an integrated manufacturing environment and to support continuing research in programmable automation. The system design objectives, major functional components, and the AML language are described [4]. In the paper, P. Plaza et al. review development platforms to present one alternative for students as part of an educational environment. They cover several applications based on different development platforms used currently [5]. Wei-Po Lee et al. present their work on the Khepera miniature robot. They evolve a control system for a moderately complicated task. They first evolved the controllers in a simulation and then transferred them to the miniature robot [6]. W. Jie et al. proposed a novel control used to solve the trajectory tracking problem on a seven-degree-of-freedom robot manipulator. In their article, a sliding perturbation observer (SPO) is used to estimate the disturbance from the environment and modeling uncertainties. The stability is analyzed based on the Lyapunov functions for general and fractional-order systems. Implementing a real robot manipulator demonstrated the effects and performance of the new control method [7]. J. Zhang et al., in their paper, present the design and implementation of a new indoor security system with a jumping robot as the surveillance terminal. The captured photo is transmitted to the gateway in 3.68s with five hops at a 0.1% loss rate [8]. J. Choi et al. proposed An adaptive model-free control with non-singular terminal sliding-mode (AMC-NTSM) for high precision motion control of robot manipulators [9]. In their paper, J. O. Hamblen et al. describe a new approach for a course and laboratory designed to allow students to develop low-cost robotic prototypes. the other embedded devices feature Internet connectivity, I/O, networking, a real-time operating system (RTOS), and object-oriented C/C++ [10].

The above all research works are most relevant in the current era. The send commands from GUI to robots are either wired or wireless through various protocols like Zigbee or TCP/IP. The device or robot is controlled through customized software. For the prototyping, customized software leads to some delays. That is why for rapid prototyping, most of the cases need some readily available utility tolls to debug the system. So, this utility tool is for those who want to debug communication systems as soon as possible. So, our work may help the researcher with rapid prototyping.

3. OBJECTIVES :

The objective of this research is to provide an enhanced utility command-line interface. Our robot researcher can send the command and receives the response efficiently. It has some extra functionality that is suitable to test and debug the automation device or robot easily. When we debug any automation device, we need to send data repetitively, which will take unnecessary delay if we do it manually. And also, error-prone. So, we need to send commands autonomously. That is why we created a command-line interface with some extended features to research made easy.

4. APPROACH AND METHODOLOGY :

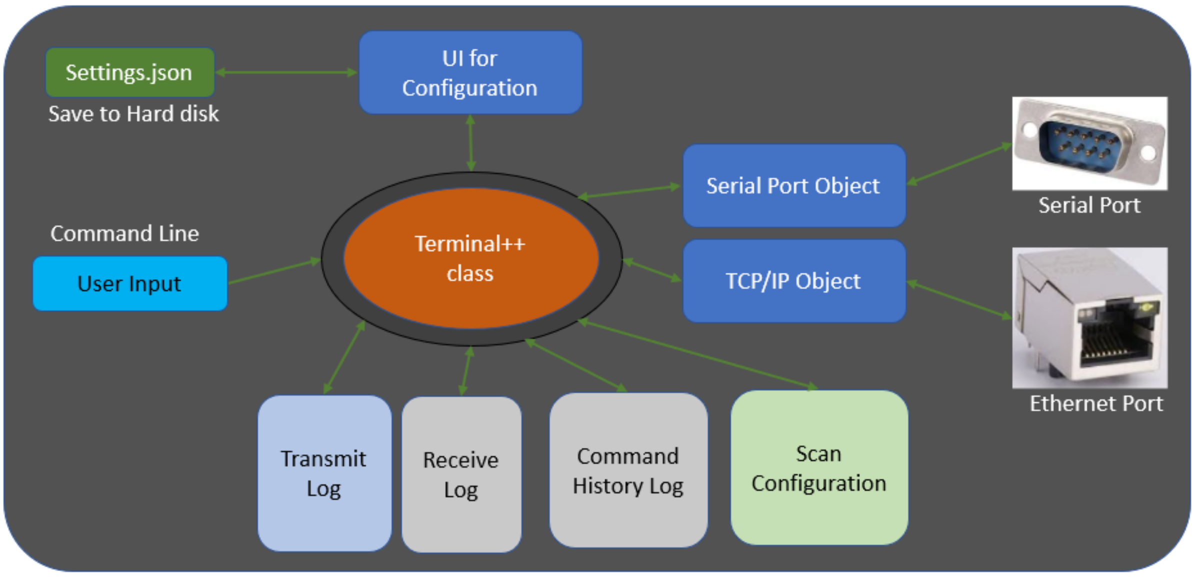
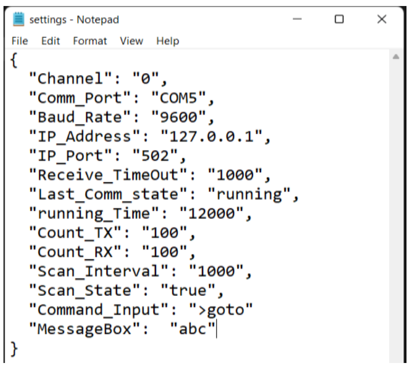


Fig.1 : The block diagram of the application

Figure 1 depicts the block diagram of our application. The main object or coordinator object is the Terminal ++ object. It manages all the connected classes. The user inputs the command through the Command Input Text box. As we are familiar command line, we provide a textbox to send the data to the remote device in the application. After typing the command, press the "SEND" button on the right side of the command button or press enter key, which is trets as accept. There are several text boxes around the application which are used to set up some communication parameters. Once set, all the parameter is saved by pressing the save config button.

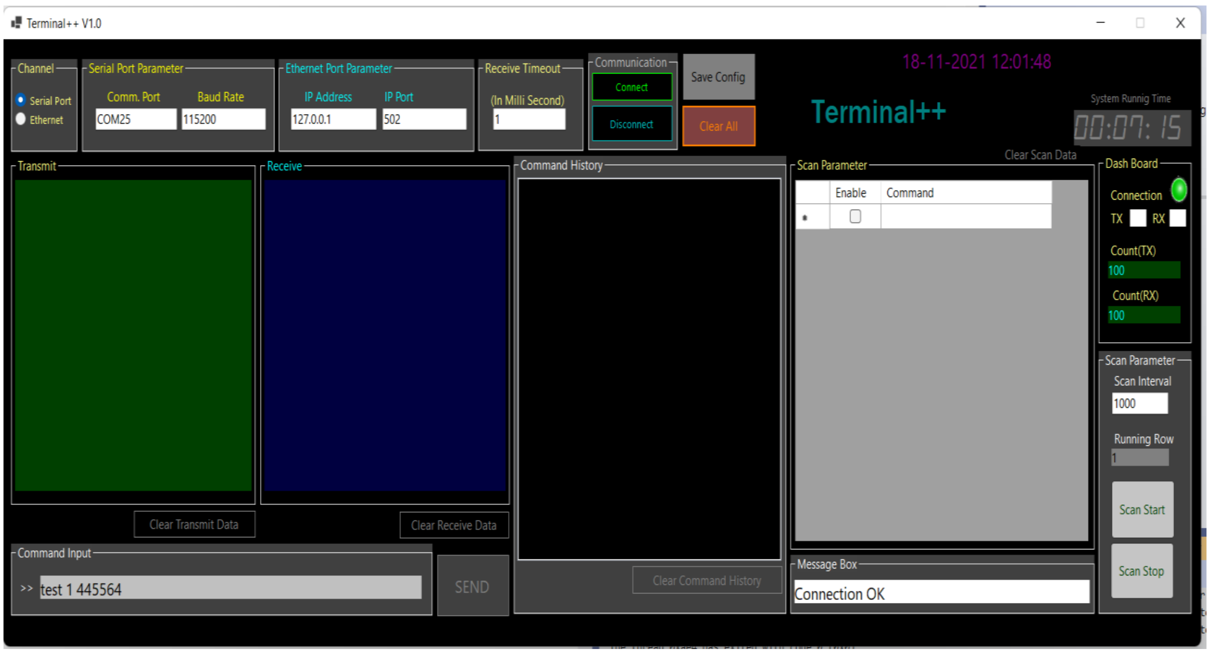
When the application is closed, it automatically saves the data—no need to repeat after reopening the application. Figure 2 depicts the configuration data is held in the Settings.json file. We can find it inside the project folder. It should not alter. May cause error. When the application is opening, reading the data, fill the associated textbox. And when the application is closing, the data save one by one into the file. For our audit purpose, we provide log data. Whatever we send will go to the transmit log list box. The receive log is filled upon receiving the data from the remote or connected device. The command History log list box is filled with a unique command. When we enter a command and press the enter key or click on the send button, it checks from the command history list box. If the same is present in the command history list box, it will not append to the list box. Scan configuration is used to send various commands automatically. The Serial port object and TCP/IP object is the channel to communicate with the remote device.



```
{
  "Channel": "0",
  "Comm_Port": "COM5",
  "Baud_Rate": "9600",
  "IP_Address": "127.0.0.1",
  "IP_Port": "502",
  "Receive_TimeOut": "1000",
  "Last_Comm_state": "running",
  "running_Time": "12000",
  "Count_TX": "100",
  "Count_RX": "100",
  "Scan_Interval": "1000",
  "Scan_State": "true",
  "Command_Input": ">goto"
  "MessageBox":  "abc"
}
```

Fig. 2: The Saving Parameter

After the application overview through a block diagram, let introduce the GUI elements depicted in Figure 3.



Channel: [✓ Serial Port] Ethernet

Serial Port Parameter: Comm. Port COM25, Baud Rate 115200

Ethernet Port Parameter: IP Address 127.0.0.1, IP Port 502

Receive Timeout (In Milli Second): 1

18-11-2021 12:01:48

System Runnig Time 00:07: 15

Count(TX) 100

Count(RX) 100

Scan Parameter: Scan Interval 1000, Running Row 1

Command Input: >> test 1 445564

Message Box: Connection OK

Fig. 3: Application Interface

Channel: In the top left, there is a “channel” group box. If we want to communicate with a remote device through the serial port, we must select the “serial port” radio button. For TCP/IP enabled devices, select the “Ethernet” radio button.

Serial Port Parameter: In this group box, two text box available. In the “Comm. Port” text box, we need to enter which comm port is connected with the device with the running system. The available comm port can be visible inside the device manager. The baud rate text box is the serial communication speed that we can get from connected device specifications.
Ethernet Port Parameter: In this group box, there is two text field. One is the connected device IP address, and another one is the IP port.
Receive time Out: The receive time out is one of the critical parameters. When we send the command to the device, our application waits for the reply. How much time will it remain? That depends on receive time-out text field value. It is preset data. Any time we can change this parameter. After changing the parameter need to press enter. Then it will affect after subsequent fetch data. It may be delayed if we do not enter a significant time because it is a blocking function. It will not return in the main loop until any data is received or time out occurs.
Communication: Inside the communication group box, there are two buttons. One is “Connect,” which is used to connect with the remote device. The “Disconnect” button is used to disconnect from the connected device.
Save Config: The next UI element is the “Save Config” Button. Whatever settings are entered are saved. It is minimized to repeat the same configuration when reopening the application. If we do not press the Save config button, we should not worry. All settings are automatically saved upon closing the application. The abnormal termination has not preserved the settings. In this scenario, the form closing event is not fired before the application is stopped.
Clear All: This button is used to clear the transmit data, receive data, and command history content from their respective list box.
Transmit Listbox: The transmit data is displayed here.
Receive Listbox: The received data is displayed here.
Command History List box: when we type some command, and press enter, the system checks the typed content. If it is not entered as before, it will append to this history box.
Scan Parameter: This is the scan data interface. If we want to send data automatically, we need to enter it here. Double click on the line from the command history list box that we want to add in scanning.

After inserting the data, we can also modify it. In Every line, there is a enable check box. If the check box is not checked, the application will not scan the data.
Command Input: it is the prime user input. In this text box, we enter the command to send.
Send button: after writing the command, press the "enter" key to send the connected device.
Clear Transmit Data button: This button clears the transmit data content.
Clear Receive Data Button: This button clears the receive data content.
Clear Command History Button: This button clears the command history.
Message box: it shows different notification content.
Clear Scan Label: One label, "clear Scan Data," is on the top right side of the scan parameter group box. It clears the scan data.
DashBoard: In the dashboard from the top, the first one is Connection LED. When we try to connect with the connected device, this LED will turn green; if it fails, turn it into the RED. After successfully joining, when we send the data to the remote device, the TX LED with glow, and if any data is received from the connected device, the RX LED will glow. The "Count(TX)" read-only text box displays the number of transmitting data packets. The "Count(RX)" read-only text box is displayed the receive packet count of the remote or connected device.
Scan Parameter: Under the scan Parameter group box from the top, the first one is "Scan Interval." Text box. It is the scanning interval between lines. When we start the scan, from the top, the system begins scanning one by one line. If the check box is not checked, the scanner skips the lines. The interval unit is in a millisecond. The next one is "Running Row," the read-only text box displays the running scan row. The next one is the "Scan Start" button. It is used to start the scan. And "Scan stop" button is used to stop the scan.
System Running Time: On the top-right text box, display the system running time.

5. EXPERIMENT :

Now is the time to do some practical experiments. Download the project from the Github repository. The link is available from the recommendation section. We need to follow the below steps:-

1. Open the application.
2. Select channel from the channel-group box.
3. If the selection is the serial port, fill the "Serial port parameter." For Ethernet, fill in the Ethernet port parameter.
4. Then fill in the suitable receive time out parameter. By default, we can set 1000 milliseconds.
5. Press the connect button. If the device is successfully commenced, the Dashboard Connection LED will go green and turn red if the error.
6. In the command Input textbox, enter the command and observe, the data is transmitted to the remote/connected device. The transmitted information is displayed inside the transmit log. If any reply is received against receive data, it will display inside the receive text box.
7. We can send the command automatically. From the command history list box, double click on any line, adding to the scan Datagrid. Then set scan interval for the speed of the scanning of each line. Then press the "Scan start" button. The scan line will scroll down one by one, depending on the enabled check box. It will scan that line is enabled by a check box. To stop the scan, press the "Scan stop" button.

That's all.

6. RECOMMENDATIONS :

- ❖ The complete source code is available from https://github.com/sudipchakraborty/Terminal-plus-plus-for-Robot-Researcher-Using-C-sarp
- ❖ For Microsoft visual studio 2019 edition download link https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16
- ❖ The above research work is reference only. Taking this project, we need to customize it. We wanted to provide some ideas on building an enhanced terminal program for our research need.

- This application is not bug-free. We debug what we get at experiment time. May persists several bugs, which can be debugged under more tests.
- We developed core functions only. The researcher can add some additional functionality for better usability.
    - We can include more communication channels like Bluetooth, Wi-Fi, Zigbee.
    - In the real scenario, noise is imposed on actual sensor data. We can simulate here by sending dynamic values.
    - To make this robust, Error Handling may be reviewed.
- Any application-related suggestions are appreciated to sudip.pdf@srinivasuniversity.edu.in

7. CONCLUSION :

We observe how we can quickly build a featured rich terminal program for our robot researchers to their work through this research work. It is a small but powerful application. Using this utility program, we can send data to the connected device very efficient way. For repeated send, this tool is much more effective. Source code is available, and anyone can customize and integrate it into their project without any issue.

REFERENCES :

[1] Lu, X., Liu, W., Wang, H., and Sun, Q. (2013). Robot control design based on the smartphone. *25th Chinese Control and Decision Conference (CCDC)*, pp. 2820-2823. DOI: 10.1109/CCDC.2013.6561425.
CrossRef/DOI

[2] Shimano, B., Geschke, C., and Spalding, C. (1984). VAL-II: A new robot control system for automatic manufacturing. *Proceedings. 1984 IEEE International Conference on Robotics and Automation*, pp. 278-292. DOI: 10.1109/ROBOT.1984.1087187.
CrossRef/DOI

[3] Gong, L. *et al*., (2017). Real-time human-in-the-loop remote control for a life-size traffic police robot with multiple augmented reality aided display terminals. *2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, 420-425. DOI: 10.1109/ICARM.2017.8273199.
CrossRef/DOI

[4] Taylor R. H. and Grossman, D. D. (1983). An integrated robot system architecture. *Proceedings of the IEEE*, *71*(7), 842-856. DOI: 10.1109/PROC.1983.12682.
CrossRef/DOI

[5] Plaza, P. Sancristobal, Fernandez, G. Castro M, and C. Pérez, (2016). Collaborative robotic educational tool based on programmable logic and Arduino. *Technologies Applied to Electronics Teaching (TAEE)*, 1-8. DOI: 10.1109/TAEE.2016.7528380.
CrossRef/DOI

[6] Wei-Po Lee, J. Hallam and H. H. Lund, (1997). Applying genetic programming to evolve behavior primitives and arbitrators for mobile robots. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, 501-506. DOI: 10.1109/ICEC.1997.592362.
CrossRef/DOI

[7] Jie, W., Yudong, Z., et al. (2020). Trajectory Tracking Control Using Fractional-Order Terminal Sliding Mode Control with Sliding Perturbation Observer for a 7-DOF Robot Manipulator. *IEEE/ASME Transactions on Mechatronics*, *25*(4), 1886-1893. DOI: 10.1109/TMECH.2020.2992676.
CrossRef/DOI

[8] Zhang, J., Song, G. Qiao, G., Meng, T. and Sun, H. (2011). An indoor security system with a jumping robot as the surveillance terminal. *IEEE Transactions on Consumer Electronics*, *57*(4), 1774-1781. DOI: 10.1109/TCE.2011.6131153.
CrossRef/DOI

[9] Choi, J., Baek, J., Lee, W., Lee, Y. S. and Han, S. (2020). Adaptive Model-Free Control with Non-singular Terminal Sliding-Mode for Application to Robot Manipulators. *IEEE Access*, *8*(1), 169897-169907. DOI: 10.1109/ACCESS.2020.3022523.
CrossRef/DOI

[10] Hamblen, J. O. and van Bekkum, G. M. E. (2013). An Embedded Systems Laboratory to Support Rapid Prototyping of Robotics and the Internet of Things. *IEEE Transactions on Education*, *56*(1), 121-128. DOI: 10.1109/TE.2012.2227320.
CrossRef/DOI

****