DOI: 10.35598/mcfpga.2020.007

Remote Debugging of Embedded Systems in STM32CubeMonitor

Oleksandr Velihorskyi
ORCID 0000-0002-8256-7339
Biomedical radioelectronic apparatus and system department
Chernihiv National University of Technology
Chernihiv, Ukraine
oleksandr.veligorsky@inel.stu.cn.ua

Ihor Nesterov
Master student, "Telecommunications and radiotechnics" programme
Chernihiv National University of Technology
Chernihiv, Ukraine
ihor.nesterov@inel.stu.cn.ua

Maksym Khomenko
ORCID 0000-0001-9084-3527
Biomedical radioelectronic apparatus and system department
Chernihiv National University of Technology
Chernihiv, Ukraine
mr.homax@gmail.com

***Abstract*—Debugging of embedded systems is one of the most important parts of firmware development. Real-time trace debugging by means of special software and hardware provides the best way to debug the firmware on real equipment. New challenges in higher education, caused by COVID-19 pandemic, require new approaches in courses, oriented on embedded system development. The paper is devoted to experience of STM32CubeMonitor implementation for remote debugging of STM32-based MCU boards during the quarantine and distance learning process, caused by COVID-19 pandemic.**

Keywords—embedded systems, microcontroller, debug, firmware.

I. INTRODUCTION

Debugging of embedded software is one of essential part of radioelectronic devices and systems development. As it pointed in [1], process workflow of embedded software development consists of the next steps: business modelling, requirements, analysis and design, implementation, test and deployment. Debugging is a part of implementation and test parts, when the developer writing the code for microcontroller or signal processor, flashing the device, observing and analyzing the implementation. Various debugging techniques can be used in embedded systems [2], such as traditional (print method, run-time methods), as well as integration testing techniques. Such classic debugging techniques is based on breakpoints, observing of variables, debugging some functions, etc. Any IDE (Keil, Atollic TrueStudio, STM32Cube IDE) has their own functionality to provide debug: variables window, breakpoints (including conditional breakpoints), etc. Another powerful method is used for final tests – so called real-time trace debugging, when special additional software (e.g. *Percepio Tracealyzer* [3] for RTOS- or Linux-based systems), or hardware (e.g. *PCAN-USB* for tracing of CAN interfaces in embedded systems [4]) is used. It should be noted, that usually debugging data performed in digital format as numbers or strings (such as variable values, data packages values, etc.), so it is not so easy to track and visualize changes of data.

To provide unique debugging functions, usually, developer should design his own application (running on Windows on other operation system) that will visualize necessary data in convenient form, like flowcharts, graphs, gauges, etc. In this case, such additional work requests additional time. Moreover, any changes in data format require significant changes in designed application (including compiling, building, etc.). Taking into account the increasing complexity of debugging, some MCU and IDE vendors has already launched additional development tools, aimed to help with complex debug of firmware. One of the first such debugging tools was STM Studio [6] that helps real-time monitoring and visualizing of variables. Unfortunately, this tool has status "NRND – Not recommended for new design" now, but, STM has released a new, more powerful tools family, that will replace STM Studio – STM32CubeMonitor [5]. This paper describes the functionality and the practical experience of using the STM32CubeMonitor in remote Microcontrollers and Embedded Systems Lab on Biomedical radioelectronic and system (BRAS) department of Chernihiv National University of Technology (CNUT).

II. STM32CUBEMONITOR

A. Family of STM32CubeMonitor tools

The STM32CubeMonitor family helps in debugging of embedded software, developed for STM32 microcontrollers [5]. The tools provide real-time reading and visualization of any variables, including remote data.The family consist of the next tools:

- STM32CmonPwr – monitoring on PC power data (currents, voltages and powers) from special X-Nucleo board;
- STM32CmonRF – monitoring RF performance of STM32-based hardware devices, such as Bluetooth Low Energy and 802.15.4 integrated RF transceivers;
- STM32CMonUCPD – monitoring and configuring of USB Type-C and Power Delivery applications for STM32 microcontrollers;
- STM32CubeMonitor – monitoring and visualization of data on Linux, Mac and Windows for STM32 microcontrollers.

In other words, all family consists of specialized tools (power monitor, RF monitor, power delivery monitor), and versatile Cube monitor, that should be used in all other cases.

All mentioned above STM32 debugging tools are free of charge after the authorization on STM web-portal.

STM32CubeMonitor is based on open-source Node-RED [7], flow-based programming tool, oriented on collaboration of Internet of Things devices that has a huge variety of additional modules and strong user's community.

B. *Configuration of STM32CubeMonitor*

STM32CubeMonitor has two main windows – flowchart, containing all nodes for data acquisition, processing and visualization, and dashboard, where user can see all the data during the debug procedure. Examples of flowchart and dashboard are shown on Fig. 1 and Fig. 2, respectively. To add necessary nodes to the flowchart, the left, so called, "nodes" panel should be used (Fig 3). The most important nodes for connection with microcontrollers are collected in group "ST Microelectronics": *acq in* and *acq out* – for receiving and transmitting the data from MCU. Debugger (e.g. ST-Link) should be selected in properties of these nodes for proper connection with target MCU. *Variables* node is used to select the variables or controller registers that will be used to visualize data or control in STM32CubeMonitor. It should be noted, that all data exchange is provided through *elf* or *axf* file (depends on used IDE), that have to be selected in properties window ("executable" field). Data from the board can be collected as fast as possible, or with some sampling frequency, continuously or starting from some trigger point. All these parameters can be configured in properties window of *variables* node.

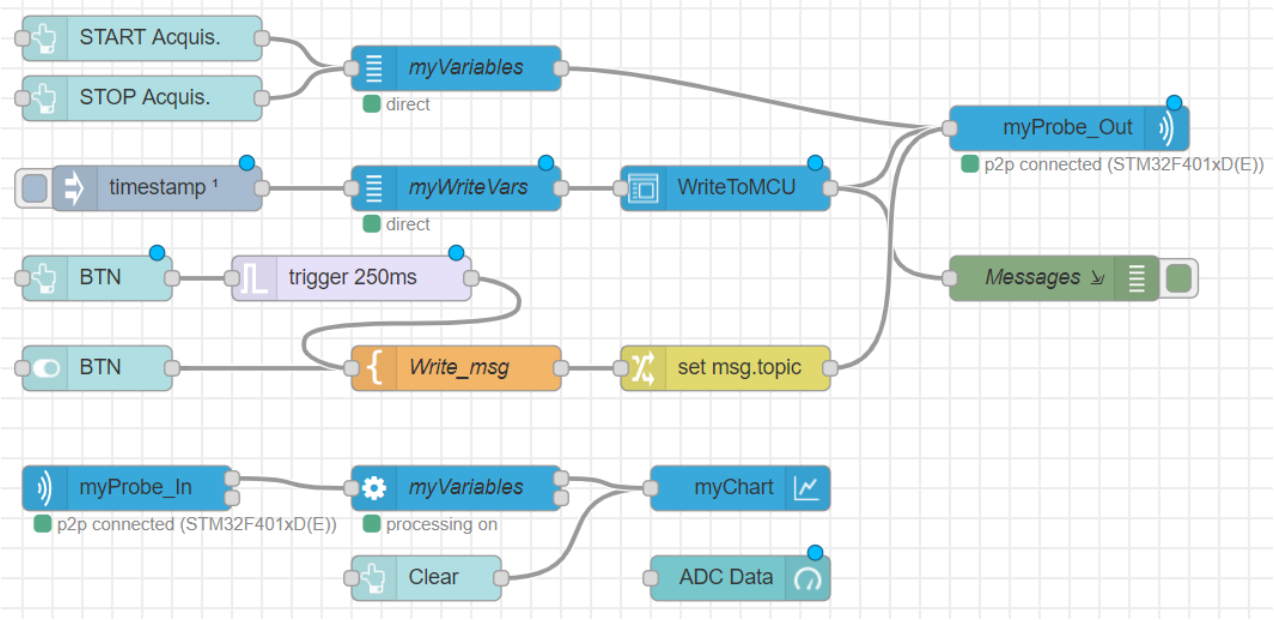

Fig. 1. Example of flowchart diagram in STM32CubeMonitor

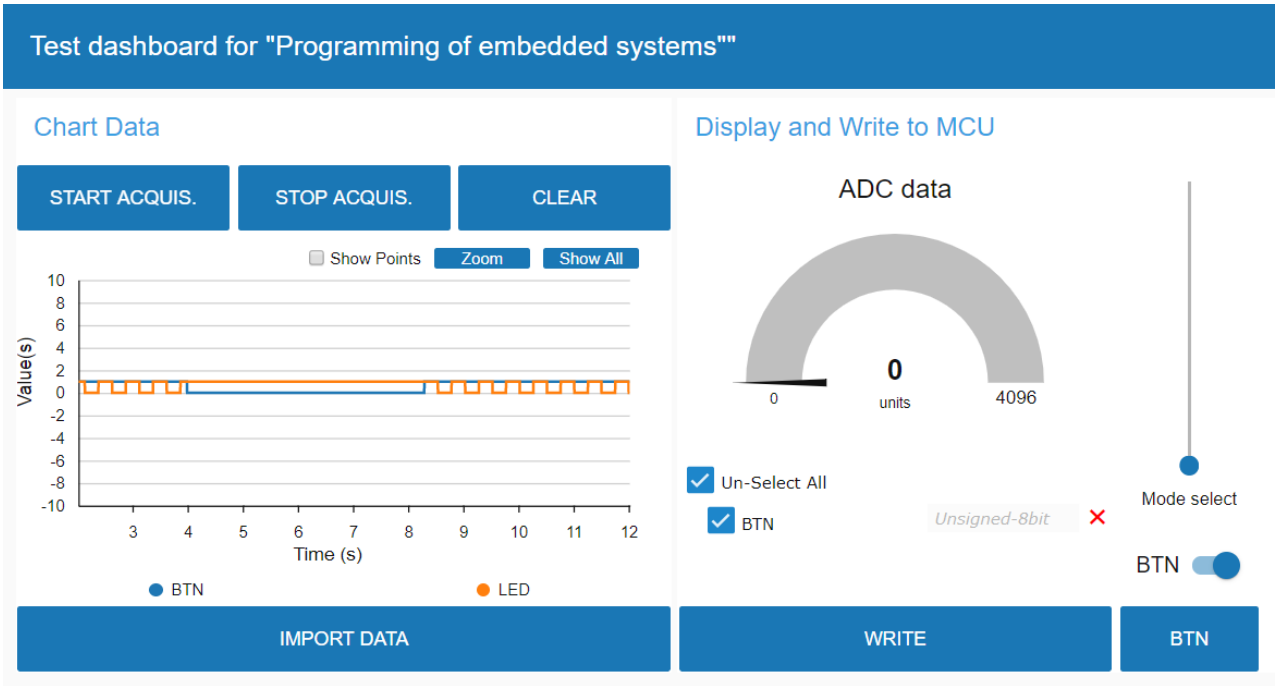

Fig. 2. Dashboard view in STM32CubeMonitor

Starting point of data collection from MCU can be also set by using buttons that will send so called “topic” *start* or *stop* (you can see it in the top left corner on Fig.1 and Fig. 2). Another key topic – *clear* can be used to clear the data on chart. *Processing* node is used for separation of variables to duplets (x and y, for representing time-based charts) and further post-processing (e.g. mathematical calculations, statistical or logical expressions with data). To show the data from MCU various nodes can be used, e.g. *Gauges*, *Charts*, *Numeric* or *Text* windows. These nodes are collected in “dashboard” group in node panel.

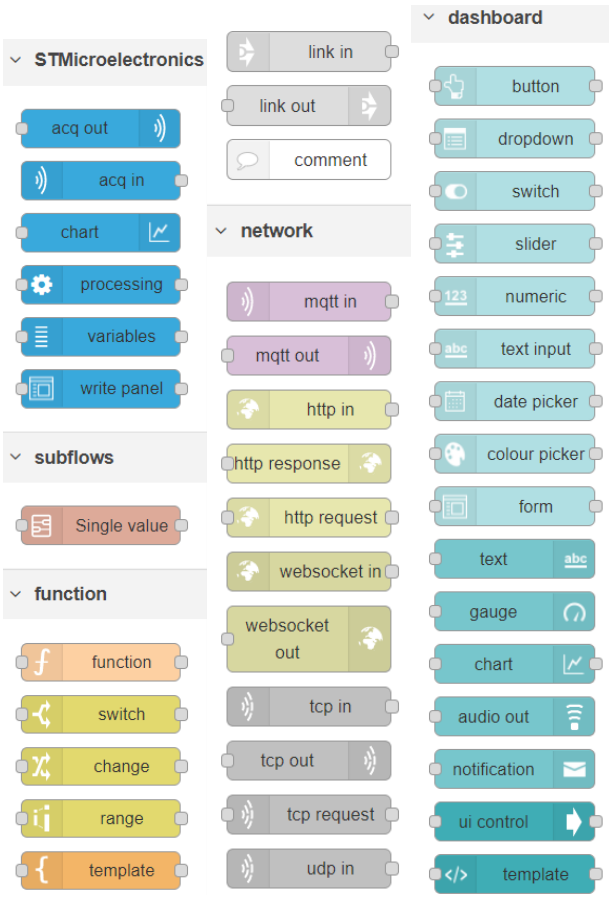

Fig. 3. Node panel in STM32CubeMonitor

One of the most important parts of debugging is external events. As it was described above, classical debugging methods, implemented in IDE has a lot of limitations for external events, so, STM32CubeMonitor provide for developers many useful nodes, such as *button*, *switch*, *slider*, *numeric*, *text input*, *template* and *write panel* (Fig. 3). The easiest way to send the data from CubeMonitor to MCU is *write panel*. As it shown on Fig. 1, series connection of nodes *variables*, *write panel* and *acq out* guaranteed sending some variables from input fields (listed in the *MyNameVars* node) to MCU. The limitation of that way is all variables are represented in *Write panel* as text input fields. It is ok for text or numerical data, but not so good for Boolean data (e.g. buttons, switches, etc.).

To avoid this limitation a series connection of *Button* (or *Switch*), *Template* and *Change* nodes can be used (nodes *BTN*, *Write_msg* and *set.msg_topic* in the center of Fig. 1). The main part is template node, where JSON template should be written, including address of the variable (all addresses of variables can be found in the “executable” field of the *Variables* node), and type (data type, e.g. 1 – uint8_t, etc.). It should be noted, that implementation of *Button* node for sending the state to MCU requires additional trigger (violet block n Fig. 1), ensures resetting the button state to previous value with some delay (e.g. 250 ms). Unfortunately, “long press” can’t be simulated in the current version of STM32CubeMonitor, so *Switch* node will be preferable for such cases.

After the connection of nodes in flowchart mode, user should distribute all widgets (nodes, represented in *dashboard* section of *node* panel – Fig. 3) on the dashboard (Fig. 2). “Dashboard – Layout – Layout” command should be used to organize all widgets on dashboard (Fig. 5). In such mode, size and sequence of widgets can be changed. User can also add groups (block, includes similar widgets, e.g. for visualization of data, or for writing data to MCU), tabs (separated dashboard that can be switched by clicking on Tab header), etc. To start the data acquisition, user should check the connection on flowchart (green marks “p2p connected”, “processing on”, “direct”), then press “Deploy”, “Dashboard” buttons, and then “Start acquisition” in Dashboard window (Fig. 2). It should be noted, that in case of control from STM32CubeMonitor, debug in IDE will not works. Any other connection will lead to disconnection of STM32CubeMonitor (green marks will be replaced by red with error messages), but, user can set “shared mode” (based on TCP protocol instead of default p2p), providing debugging simultaneously in IDE and STM32CubeMonitor [8]. The only one drawback of such connection is lower rate of data acquisition.

In case, when project was changed, *elf* or *axf file* should be updated in *acq in* and *acq out* nodes. Otherwise, linking of variables and their addresses will be lost.

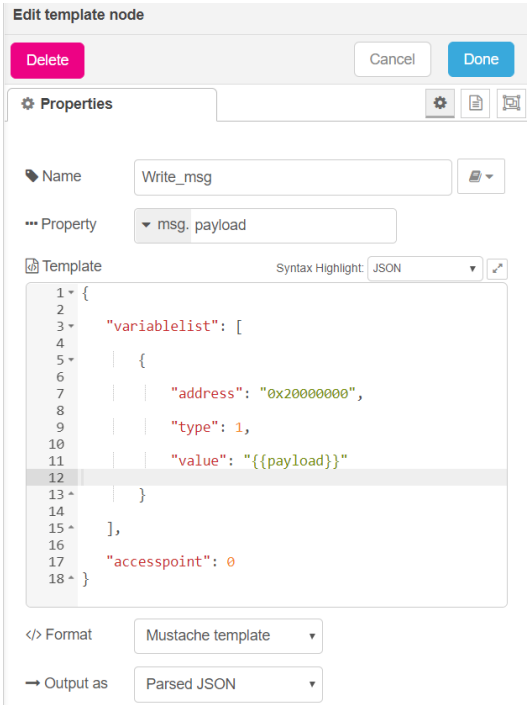

Fig. 4. Node panel in STM32CubeMonitor

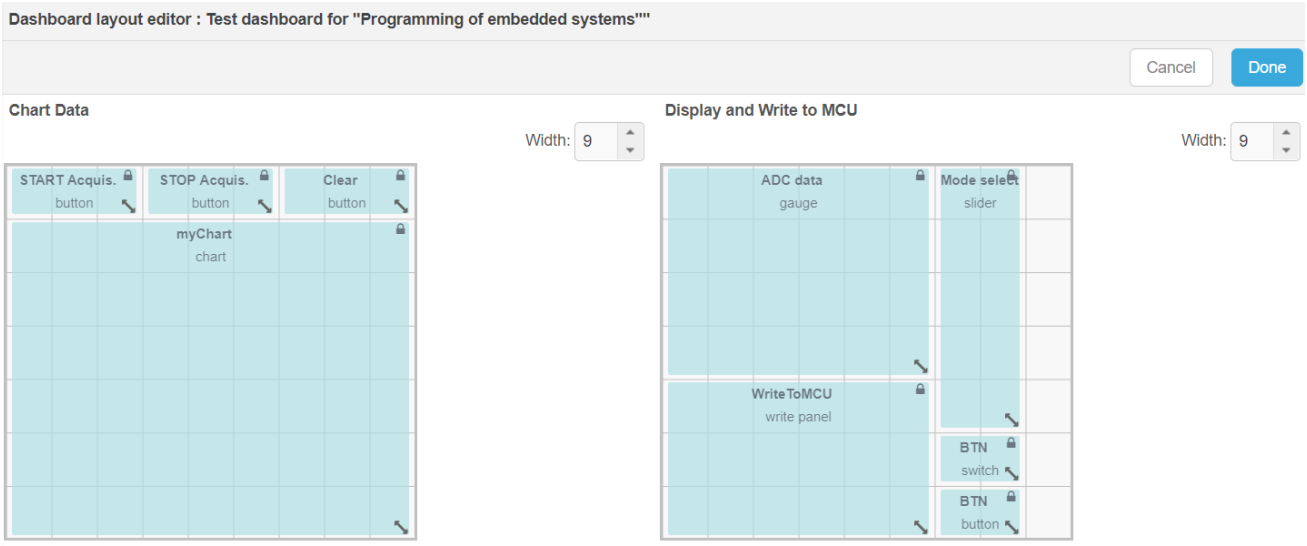

Fig. 5. Layout editor for dashboard in STM32CubeMonitor

III. Implementation of STM32CubeMonitor in Remote Microcontrolers and Embedded Systems Lab

COVID-19 pandemic and quarantine has significantly impacted on higher educational institutions all over the world. To respond to COVID-19 challenges, universities deployed various distance learning technologies, including videoconferences, video lectures, course management systems for distance learning, etc. On master-level educational program "Telecommunications and radiotechnics" in CNUT BRAS department lectures was took place in YouTube streams and Zoom videoconferences modes. Meanwhile, one of the toughest challenges was providing for students all practical-oriented competences and learning outcomes, planned in educational program. To reach planned learning outcomes, on some courses lab equipment was temporarily lent to applicants for higher education during the quarantine, e.g. single-board PCs Raspberry Pi (course «Architecture of modern processors»). Taking into account that STM32 Nucleo boards in spring semester using in different courses for master and bachelor students, it was not possible to distribute these boards. So, taking into account the previous experience of collaboration between CNUT and Bonn-Rhein-Sieg University of Applied Sciences on remote lab [9], it was decided to establish Microcontroller and embedded system remote lab on BRAS department, equipped with PCs, digital multifunctional USB-oscilloscopes ISDS205X (also includes DDS function signal generator and logic analyzer) and STM32 Nucleo development boards. To use this equipment remotely, TeamViewer software was used. Taking into account that PCs in this lab as of now doesn't have a static IP address, it is not possible switch on and switch off them remotely, and head of lab should switch on and off them manually.

One of the essential parts of the remote lab is STM32CubeMonitor, because remote operation doesn't provide possibility to use the input elements: toggle switches, press buttons, etc., whereas change of corresponding MCU registers not so convenient in IDE. So, STM32CubeMonitor was used to monitor the variables values, as well as for emulation of input elements. To provide full functionality of Remote lab from the starting of 2020-21 studying year, a new equipment based on STM32H7 MCUs was installed (with support of AgileVision.Io), and development of dashboards for all laboratory works is in the progress.

Acknowledgment

This work and establishing of remote Microcontrollers and Embedded Systems Lab on BRAS department was supported by company AgileVision.Io, one of the stakeholders of "Telecommunications and radiotechnics" master program in Chernihiv National University of Technology.

References

[1] T Punkka, "Agile methods and Firmware Development," SoberIT 2005, pp.1-21. http://www.ngware.eu/blog/papers/agile_firmware_punkka_V103.pdf

[2] Top Debugging Techniques Used In Embedded Systems, https://www.totalphase.com/blog/2020/03/top-debugging-techniques-used-in-embedded-systems/.

[3] Percepio Tracealyzer https://percepio.com/tracealyzer/.

[4] PCAN-USB. CAN Interface for USB https://www.peak-system.com/PCAN-USB.199.0.html?&L=1.

[5] STM32CubeMonitor https://www.st.com/en/development-tools/stm32cubemonitor.html.

[6] STM-STUDIO-STM32 https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-performance-and-debuggers/stm-studio-stm32.html.

[7] Node-RED. Low-code programming for event-driven applications https://nodered.org/.

[8] How to configure shared mode https://wiki.st.com/stm32mcu/wiki/STM32CubeMonitor:How_to_configure_shared_mode.

[9] K. Pretz, "German University Opens Up Its Hands-on Remote FPGA Lab During the Coronavirus Pandemic," https://spectrum.ieee.org/news-from-around-ieee/the-institute/ieee-member-news/german-university-opens-up-its-handson-remote-fpga-lab-during-the-coronavirus-pandemic.