Use of cross-compiling in the programming of an intelligent device for measuring electrical quantities

Jakub Przybylski[1,*] and *Paweł* Idziak[2]

[1] Faculty of Mechanical Engineering and Management, Poznań University of Technology, 60-965 Poznań, ul. Piotrowo 3, Poland
[2] Faculty of Electrical Engineering, Poznań University of Technology 60-965 Poznań, ul. Piotrowo 3A, Poland

Abstract. A device for voltage and direct current measurement, built by coupling a Raspberry Pi microcomputer with an Arduino series microcontroller is presented in this paper. The structure of the device is presented and the method of implementing proprietary programs for both devices is described. A touch screen of the device allows the user to decide on the measurement mode (automatic or manual) and on the method of archiving and visualizing the effects of measurements. The application uses the so-called cross-compiling, thanks to which it is possible to effectively couple such different platforms as Raspberry Pi and Arduino.

1 Introduction

The digitization of practically all life areas forces the search for such techniques to measure various physical quantities that the measurement results could be: digitally processed into a form enabling their direct or deferred visualization, such as "stored" digital signals transformed (changed) into an analogue form and archived in an efficient way.

Modern instruments usually allow for measurement of several different physical quantities, e.g. current, voltage, temperature, frequency, sometimes power, condenser capacitance, resistance, etc. Depending on the measurement principle and device design, the measurement results are made available to the operator either in an analogue or in a digital form [1]. In a typical multimeter, the control elements placed on the front panel of the multimeter allow selection of the measurement mode and range of measured quantities, and the measurement configuration of the instrument remains unchanged [2].

2 Design of an intelligent multimeter

The intelligent multimeter presented here was built using the Raspberry Pi and Arduino platforms. In the original version, it allows you to measure DC voltage or current in one of three available modes [1]. Two of them will ensure an automatic measurement of the controlled quantity and registering it. In the automatic mode, the user has to declare the number of planned measurements - detecting changes in values and "capturing" data is the task of the multimeter system. In the manual mode, the user decides themselves on the number and time of measurements to be taken and recorded [1]. The touch screen is used to operate the instrument and to visualize the measurement values. The program realizing all functions of the multimeter allows you to export the measurements to a file and save the latter to a SD card.

An important feature of such an instrument is the possibility to reconfigure it and adapt it to the user's individual needs (change of the screen appearance, change of the measured quantities, etc.). This requires, however, the use of sufficiently efficient software.

The use of a touch screen requires the use of a suitably efficient control platform that allows you to design and program the user's interface in a relatively simple way.

In this project, a Raspberry Pi 3 microcomputer in version B was used. As none of the models of this device offers analogue inputs necessary to implement the foreseen algorithms, the measurement functions were separated from the screen operation.

In the project, a microcontroller from the Arduino family - Arduino Mega 2560, was used to measure DC voltage and current. It supports analogue inputs and is able to communicate with the Raspberry Pi platform via I2C or USB type communication modules. An additional advantage is the wide range of additional components and modules such as, for example, current sensors (required for the ammeter).

2.1 Block diagram of the device

For designing the architecture of the device using two coupled platforms, the priority was their reliable, mutual communication. Two different communication interfaces are used to control and transmit measured values. In the case of one common interface, the device would have to interrupt transmission of measured values to change the system's operating mode. This could result in problems at the stage of programming and developing the

* Corresponding author: pawel.idziak@put.poznan.pl

application, as well as undesirable, misleading measurements during the operation of the device.

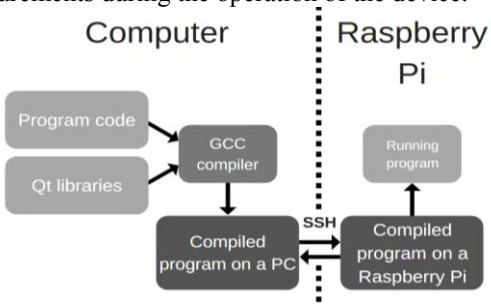

Fig. 1. Block diagram of an intelligent multimeter.

At a given moment, the Raspberry Pi provides an image display on only one output. It is not possible to run the application on the touch screen while operating the Raspbian system (an operating system edited for the microcomputer) via the image from the HDMI port. For this reason, for the purposes of developing and modifying application, a remote access to microcomputer files was provided through the SSH (Secure Shell) communication protocol, and the application itself was designed from the PC level using cross-compiling. The software used for it is open-source software.

3. Cross-compiling

The cross-compiling, supported by the Qt environment, allows programming and compiling applications in architecture other than the target platform architecture. This allows you to design applications from the level of a desktop computer, which is more efficient, and to reduce the share of the less-efficient portable target platform for testing the program code functioning. The realization of the same process on a desktop computer shortens the time of device configuration from a dozen or so to a few hours [3].

The program code along with the Qt libraries is compiled by the GNU Compiler Collection (GCC) and then sent via SSH protocol to the Raspberry Pi. It is possible to install Qt libraries together with the Qt Creator utility program in the Raspbian system, but the configuration process (realized partly in the Terminal) takes then a very long time [4].

Performing cross-compiling in the Windows environment succeeds only for the Raspberry Pi 1 and 2 models. Therefore, the tools from the Linux-Ubuntu system were additionally used for the design process. The structure is shown in Figure 2.

The key steps in configuring the cross-compilation are as follows: build a so-called "Qt Base" containing all basic libraries and upload them to the device by running scripts in Terminal. The whole process may last up to several hours, depending on the computer hardware possibilities. The information about the original and target architecture of platform, path names of previously created folders with Qt files for both of these platforms, and the name of the device for which the program is to be compiled are to be included into the project launch command.

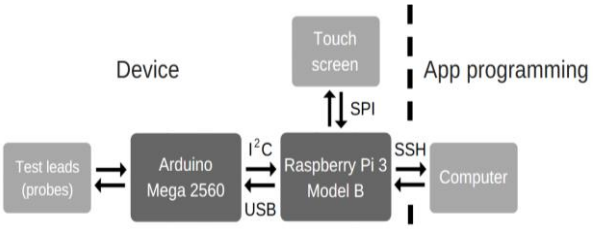

Fig. 2. Structure of the system during cross-compiling.

The command for the configuration of cross-compilation from Ubuntu 64-bit system for Raspberry Pi 3 Model B with 32-bit Raspbian system used in the multimeter structure is as follows [4]:

/configure -release -opengl es2 -device linux-rasp-pi3-g++-device-option CROSS_COMPILE=~/raspi/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin/arm-linux-gnueabihf- -sysroot ~/raspi/sysroot -opensource -confirm-license -make libs -prefix /usr/local/qt5pi -extprefix ~/raspi/qt5pi -hostprefix ~/raspi/qt5 –v.

After setting up the GCC copmiler and debugger in Qt Creator as described in Qt's technical documentation, whole process has been completed, the application can be programmed from the computer level.

4. Concluding remarks

There are technical reasons to continue work on the development of an intelligent multimeter, to be used to measure not necessarily electrical quantities. The used separation of measurement and control functions to separate systems makes it possible to build a device with various measurement possibilities, operated by means of a touch screen.

This requires the use of adequately software, both in the area of communication inside the instrument as well as at the stage of design or modification of the latter.

The use of the mentioned cross-compiling significantly (several times) shortens the configuration process. By using this method the time necessary to perform the modification of the device can be shortened up to 10 times. The vast majority of used and necessary software is an open source environment.

References

1. The official Raspberry Pi documentation is available at https://www.raspberrypi.org/documentation/ (access 10/09/2017)
2. A. Kumar, What is a digital multimeter (DMM) and its working principle details, https://analyseameter.com/2015/09/digital-multimeter-dmm-working-principle.html (access 09.09.2017)
3. The official documentation of Qt libraries for version 5.10 is available at https://doc.qt.io/qt-5.10/ (access 24.01.2018)
4. M. Summerfield, Biblioteki Qt. Zaawansowane programowanie przy użyciu C++ (*Advanced programming using C ++*), Gliwice: Helion (2014)