Feedback control system based on a remote operated PID controller implemented using mbed NXP LPC1768 development board

Emil Pricop, Florin Zamfir, Nicolae Paraschiv
Petroleum-Gas University of Ploiesti
39, Bucuresti Blvd., Ploiesti, 100680, Prahova, Romania

E-mail: emil.pricop@gmail.com

Abstract. Process control is a challenging research topic for both academia and industry for a long time. Controllers evolved from the classical SISO approach to modern fuzzy or neuro-fuzzy embedded devices with networking capabilities, however PID algorithms are still used in the most industrial control loops. In this paper, we focus on the implementation of a PID controller using mbed NXP LPC1768 development board. This board integrates a powerful ARM Cortex-M3 core and has networking capabilities. The implemented controller can be remotely operated by using an Internet connection and a standard Web browser. The main advantages of the proposed embedded system are customizability, easy operation and very low power consumption. The experimental results obtained by using a simulated process are analysed and shows that the implementation can be done with success in industrial applications.

1. Introduction
Process control is a challenging topic of research for both academia and industry, being present in almost each industry and human activity from energy production to flying planes. Automatic process control refers to all the methods used to control the process variable in order to achieve an objective without any human intervention. The process control objectives are related to process efficiency and safety.

Industrial process control infrastructure evolved significantly in the last decades along with electronics and computer science. The analog devices used for implementation of basic controllers were replaced by digital devices with significant processing power such as PLCs and process computers. The control algorithms also evolved and the research shows how to use fuzzy or neuro-fuzzy approaches in various industrial facilities, but PID algorithm is still used in the majority of industrial control loops. Also the devices are now exchanging data using various protocols, starting from specific ones such as Modbus, Profibus or OPC and even on the global Internet, exposing the industrial infrastructure to various security threats [1].

In this paper we focus on the implementation of a PID controller using mbed NXP LPC1768 development board. This board integrates a powerful ARM Cortex-M3 core and has networking capabilities. The controller can be operated locally, from a PC, or remotely by using a TCP/IP or Internet connection and a simple Web browser. The development board characteristics are described in the second section of this paper.

The novelty of the implementation consists mainly in the development board used. This development board can be utilized in industrial applications due to its very low-cost and very low power consumption.
The controller implemented on the development board is very versatile, the control algorithm can be changed without any hardware modifications.

The control system was implemented in an experimental setup. The process and disturbances are simulated using Simulink software. The development board is connected to the simulated process using a National Instruments NI-6008 data acquisition equipment. In this way the controller can apply the command to the simulated process and receive the feedback in order to compute the next value of the command. This approach is presented in the third section of the paper.

The objective of the paper is to develop a networked controller. In order to reach this objective a RPC over HTTP server was implemented and a web based operational interface was designed. The server could be accessed from anywhere in the world where an Internet connection is available using a standard computer or a smartphone with a simple Web browser. The implementation and testing of the network components are presented in the fourth section of this paper.

The proposed networked controller was extensively tested using various process models in Simulink. The simulation results are presented in the last section of this paper.

2. Mbed NXP LPC1768 development board

Mbed NXP LPC1768 is a powerful and user-friendly development board based on an ARM Cortex-M3 microcontroller. It integrates the microcontroller unit together with peripherals such as SPI, I2C, USB and Ethernet interfaces. The development board has the main objective of rapid prototyping, supporting even an on-line compiler for writing code quickly and efficiently. In this section we will present the main features provided by mbed NXP LPC1768 development board.

Mbed board is built around an LPC1768 microcontroller produced by NXP semiconductor company. This microcontroller integrates an ARM Cortex-M3 core running at 100 MHz frequency with 63 KB of SRAM and 512 KB of Flash memory. The core architecture uses a multi-layer AHB bus that permit to use high-bandwidth peripherals such as Ethernet interface and USB to run simultaneously without having an impact on the board performance [2].

LPC1768 is a very versatile and powerful microcontroller. It can be used not only for implementing basic computing tasks but also it allows rapid prototyping for complex devices. The peripherals are divided in three categories: serial connected, analog and general purpose ones. The serial peripherals are:

- an 10/100 Ethernet controller offering IEEE 802.3 standard compatibility and full TCP/IP stack. The controller also supports WOL – Wake on LAN function, allowing the start-up of the device from network.
- USB 2.0 full-speed interface allowing a bandwidth of maximum 12 Mbps;
- three SPI controllers, full-duplex with a maximum bandwidth of 12,5 Mbps;
- three I2C bi-directional interfaces with a 1 Mbps bandwidth and the possibility to connect multiple equipment for master/slave communication;
- two CAN 2.0B controllers, with 1 Mbps bandwidth and 32-bit address space;
- one UART serial interface allowing 6,25 Mbps speed and full support of RS-232 and RS-485 communication standards [3].

The analog peripherals are represented by a 12-bit, 8 channels ADC (analog to digital converter) and a 10-bit DAC (digital to analog converter). The analog to digital converter has support for dynamic memory access. The conversion is made at 200 Hz multi-channel. Each converter channel has its own memory register for conversion result temporary storage. The conversion step is 0,806 mV with conversion limits between 0 and 3,3V [3].

In the general purposes category of peripherals, there are found a DMA controller with eight channels, PWM interfaces, 32-bit general timers or counters. For real-time application LPC1768 has a real time clock (RTC) with a 32kHz oscillator. In order to avoid microcontroller blockage there is implemented a Watch-dog timer that can reset the whole system in case of an error.
In order to interact easily with this great variety of peripherals mbed provides open-source libraries and APIs. In this way, the prototyping time is significantly reduced since the need to write low-level code is eliminated. If open-source stacks or third-party protocols are needed they can be added and reused in all the projects. All the mbed libraries comply to Cortex Microcontroller Software Interface Standard (CMSIS) allowing the user to migrate the project to other programming tools or to implement custom code for higher performance.

Mbed NXP LPC1768 is very easy to setup and use on any type of computer. The novelty of the device consists in the on-line compiler, available on http://developer.mbed.org. There is no need to install supplementary software apart form a standard web browser. Practically the user can login to mbed compiler from anywhere in the world, write a program in C++ using standard libraries and download the compiled firmware that can be copied on the development board. By using this approach, the user always has access to the latest version of compiler and to an update libraries set. Along with using the on-line compiler the user can program the device by using Keil MDK (Microcontroller Development Kit) that support ARM Cortex devices along with uVision Integrated Development Editor [4].

3. Feedback control system design

The feedback control system is based on a PID controller implemented on mbed NXP LPC1768 development board. The block diagram of the proposed system is presented in the figure 1. All the components listed in the block diagram are integrated on the mbed board.

The implemented PID controller has the standard inputs and output of an analogic PID controller. The set point, i, and the tuning parameters – Kp – proportional gain, Ti – integral gain and Td – derivative gain should be set by using the network interface via the web browser. The feedback from the process (r), also named process value is acquired using an analog input of the board and then is converted to a digital signal using the integrated ADC. The converted value is used to calculate the next control action (u) which is converted from digital to analog by using the internal 10-bit DAC of LPC1768. The limits for analog input/output are 0-3,3V.

![Figure 1. Proposed PID Controller system diagram](image)

The controller is implemented in the mbed NXP LPC1768 board using the standard PID library from mbed repository. The PID loop functioning is not very complex. The controller read the process value and compare it to the desired value, which is also the set point. The following step is to compute an output value in order to adjust the process value to be closer or even to accede to the set-point. The algorithm is based on the following numerical form of PID command law:

\[
  u(t) = K_p \left( e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right)
\]  

The signification of the notations in (1) is presented below:

- \( u(t) \) is the control signal (command);
• \( e(t) \) is the deviation (error), computed as the difference between \( i \) – set point and \( r \) – process value;
• \( K_p \) is proportional gain;
• \( T_i \) is the integral time constant;
• \( T_d \) is the derivative time constant [5].

The control signal is computed as a sum of three terms: the proportional value which is proportional to the current value of the error, an integral term that is an average of past error and an derivative value which linearly interpolate the future error values [5][1].

The PID library available on mbed repository implement the homonym class. The main object defined is the controller, which has a large number of specific methods. The programmer can set the input and output limits, the tuning parameters (\( K_p, T_i, T_d \)), the set point, manual or automatic operation mode and can call the command computing routine. The control action is computed based on a numeric implementation of (1) [6].

The source code for the mbed program starts with preprocessor directives and variable declarations: the two pins used for data transfer from/to PC, the tuning parameter \( s \), the set point variable and the special variable PID controller as it is seen in figures 2 and 3. In the main function, all the variables are initialized, the controller limits are set and the web server is started. The controller enters an infinite loop where it listen for new tuning values or set point modification on a thread and on another thread calculates the control signal value periodically. For debugging purposes, the microcontroller writes the process values and corresponding control signal on serial port.

```
#include "PID.h"
#include "mbed.h"
float sp, kp, ti, td;
RPCVariable<float> rpc_sp(sp, "sp");
RPCVariable<float> rpc_kp(kp, "kp");
RPCVariable<float> rpc_ti(ti, "ti");
RPCVariable<float> rpc_td(td, "td");
```

**Figure 2.** PID controller declarations

**Figure 3.** RPC variable declarations

In order to obtain good precision, the output from Simulink should be filtered. The numerical filter is a nonlinear method used to reduce the noise of the input process variable. This can be done with a median filter that reorder the last \( n \) values acquired and return the value from the center of the string. The implementation is made with pointers and FIFO method, where the data is sorted in the acquisition order and the output is computed based of the value of element placed in the middle of the sorted buffer. The program contains two structures that force the values to be stored only if the data stays in certain limits.

4. Remote operator console

The PID controller designed in this paper should be remotely operated. This objective is accomplished by using the mbed TCP/IP stack in order to implement an HTTP server that can be accessed from anywhere in the world by using a simple web browser.

The HTTP implementation on mbed has mainly two components: one webserver engine programmed in the mbed firmware and HTML files that are located in the mbed file system. Practically the HTTP server is a process running on the mbed along with the PID routine, which represent another process running on the MCU, and the files are stored on a file system like a hard drive on a PC.

The server start when mbed board power on. The IP address can be configured to be static or dynamic, meaning that it can be obtained automatically form a DHCP server on the network if available. For debugging purposes, the IP address is printed on serial port. The server listens for connections on TCP port 80, allowing any user that has a standard browser like Firefox, Chrome, Safari or Internet Explorer to view or set parameters as it is shown in figure 6.

The web page programming uses RPC JavaScript library. RPC is a protocol that allows client and server communication by replacing network protocols and communication methods. This is an inter
process communication where one program can use services from a remote program. Once compiled, the RPC function can be used by any program with client/server capability [7].

The set point and the tuning parameters are defined as RPC variables, which can be read, written or deleted using RPCVariable class, included in RPCVariable.h library. The RPC variables are defined in two places – the microcontroller program (firmware) and also in the JavaScript stored in the mbed file system. The code section containing variable definition in the mbed program is presented in figure 4.

The JavaScript links standard variable to the RPC variables defined in the microcontroller program, using functions located in the RPCVariable.h library. For example, reading Set Point variable (Fig. 4) uses “readp” function that has two major roles. First task is to store in a variable “f” the content acquired through “sp” RPCVariable reading. The “sp” is declared in the HTML header along with the rest of the RPCVariable. Second task imply using the print function to display the “sp” variable in the location referred by “id” with the same name.

Modifying the Set Point can be done with the function “writep” (Fig. 5). The value placed by the operator in the text box identified by wsp id is sent to “csp” variable when Set button is used only. This time the RPC write function send the value to the set point RPCVariable stored in mbed memory.

The operator interface can be accessed using an URL in format http://x.y.z.t/index.htm, where x.y.z.t is the IP address allocated to the microcontroller unit. The operator can retrieve the set point, process value and tuning parameters by pressing the “View Values” button. If the operator wants to change any parameter or the set point, he should write the value in the corresponding text box and press the “Set” button. If the operator does not write a value in the text box corresponding to one parameter, its value does not change.

Figure 4. Source code used to read Set Point variable.

Figure 5. Source code used to modify Set Point variable.

Figure 6. Web based operator interface for the PID controller.
5. Experimental results

Experimental testing of the proposed PID controller require the existence of a process to interact. The controller was tested with first and second order processes configured by the authors and simulated using Simulink.

In order to connect the physical PID controller implemented on the mbed NXP1768 development board it is necessary to acquire and transmit the signals from Simulink through a data acquisition board. We have used National Instruments USB-6008 data acquisition device for interconnecting the PID controller with the PC. NI USB-6008 has 8 analog inputs and 2 analog outputs. The board is supplied with NI DAQmx software drivers for operation on Microsoft Windows Operating System. The block diagram of the laboratory experimental setup is shown in figure 7.

![Block diagram of the experimental system](image)

**Figure 7.** Block diagram of the experimental system

The experimental setup includes a PC where the process is simulated in Simulink. In order to transmit and acquire signals the Simulink model should integrate the NI DAQmx drivers for USB-6008 board. The board is connected to the PC by USB port. AI0 – analog input channel from USB-6008 is used for command acquisition from the PID controller and AO0 – analog output channel is designated for transmitting the feedback process value from simulator to the controller. The connections are made as shown in the following table:
Table 1. Pins assignment on USB-6008 and mbed NXP LPC1768.

| USB-6008 pin | USB-6008 Simulink signal | mbed NXP LPC1768 pin | mbed Controller signal |
|--------------|--------------------------|----------------------|------------------------|
| A10+         | Controller command signal for the process | P18 | Controller output (computed command signal) |
| A10-         | GND | |
| AO0          | Process value – feedback signal | P15 | Controller input – feedback process value |
| GND          | GND | |

Figure 8. Experimental system for PID controller testing

The connection to NI USB-6008 board is made by an Analog Input block, from the Data Acquisition Simulink toolbox. The block is configured to acquire data from nidaq driver’s channel 0, as shown in figure 9. The block used for analog output of the process value consists of a MATLAB function that uses the programming environment standard data acquisition and signal generation functions. The function define an analog output using nidaq driver, setup the desired output channel and write data at a desired sampling rate [8].

The process is simulated as a first order model with the transfer function

$$G(s) = \frac{1.5}{2s + 1}$$  \hspace{1cm} (2)

The disturbance is generated using the same model as shown in the block diagram in figure 9. There are two display block that show the current command (control signal) and the simulated process value (output). The time scope block has the role to generate a graphical representation of the two signals.

Figure 9. Block diagram of a simulated process with external PID controller
A second simulation was made in order to test the performances of the PID controller. The external controller was replaced by a simulated PID block included in Simulink libraries as shown in the block diagram from figure 10.

**Figure 10.** Block diagram of a simulated process with internal PID controller

The simulation results for the two systems are shown in figures 11 and 12. The PID controller parameters are set point sp=1.7, proportional gain kp=1.3, integral gain ti=2.2 and derivative gain td=0. The command signal is represented with blue and the process value is the red curve.

**Figure 11.** Control signal and process value for simulated PID controller

**Figure 12.** Control signal and process value for mbed NXP LPC1768 PID controller
A combined representation of the two systems can be seen in Fig.13. The number of signals on the time scope is increased to five: set point, acquired control signal from the microcontroller, output sent to microcontroller, simulated control signal and output from Simulink PID controller.

Figure 13. Simulation results of the simulated and experimental PID controllers

Analyzing the graphs presented in figures 11, 12 and 13 we can conclude that the controller work good on a first order process. The control signal (command) has approximately the same shape as the one generated by simulated Simulink controller.

The performances of the controller can be analyzed by using two parameters: overshoot and settling time. The results obtained in case of the first order process are shown in the table below.

Table 2. Overshoot and settling time values for the first order process – set point = 1,7

| Parameter      | Simulated controller in MATLAB | Mbed implemented controller |
|----------------|--------------------------------|-----------------------------|
| Overshoot      | 0,43                           | 0,28                        |
| Maximum PV     | 2,13                           | 1,98                        |
| Settling time  | 8 sec. (0.98 criteria)         | 16 sec.                     |

The MATLAB simulated PID controller has a better settling time that the one implemented on mbed development board, being reduced by half, but the overshoot is better in the case of the implemented controller.

The experimental testing has been conducted also with a second order process as shown in the figures below.

Figure 14. Block diagram of a second order process with external PID controller
The simulation results for a second order process show that the mbed NXP LPC1768 PID controller works very good. The results obtained with the two controllers (simulated and implemented) are presented in Table 3. The values were computed using graphical methods.

Table 3. Overshoot and settling time values for the second order process – set point = 1,7

| Parameter       | Simulated controller in MATLAB | Mbed implemented controller |
|-----------------|--------------------------------|-----------------------------|
| Overshoot       | 0,84                           | 0,59                        |
| Maximum PV      | 2,54                           | 2,29                        |
| Settling time   | 13 sec.                        | 23 sec.                     |

The overshoot value is better with the mbed implemented PID controller that with the PID controller simulated in MATLAB. The settling time is higher when the implemented controller is used.

We can conclude that the controller designed and implemented using the microcontroller has lower overshoot that a simulated controller with the same tuning parameters, but the settling time is approximatively double.
6. Conclusion
In this paper is presented an innovative approach in implementing a networked PID controlled based on NXP LPC1768 development board. The controller main advantages are the versatility, easy remote operation, low cost and very low power consumption.

The implemented controller was laboratory tested by connecting it with a simulated process in Simulink. The communication was realized by using a National Instruments USB-6008 board for interfacing with the PC. The experimental results for first and second order processes are convincing and performances achieved by the PID controller are significant and comparable to the ones of a simulated controller with the same parameters. The experimental results show that the overshoot is lower when using the NXP LPC1768 implemented controller that the one obtained with the MATLAB simulated controller. Meanwhile the settling time is approximatively double in case our implementation than the MATLAB one. This result is good for industrial implementation since the controller does not action brutally on the controlled installation.

After securing the operator interface the controller can be used in any industrial setup. In case of the industrial implementation the cost of the development board solution is smaller than using a conventional PID controller with network interface.

The research should be continued in order to implement a remotely operated multi-channel controller using the same development board which is capable to run a real-time operating system – RTOS. Also the security of the remote interface should be improved in order to authenticate the operator and to log all the commands or changes issued.

References
[1] Ray Chaudhri U and Ray Chaudhri U 2013 Fundamentals of Automatic Process Control CRC Press (USA)
[2] http://www.nxp.com/documents/leaflet/LPC1768.pdf
[3] http://www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf
[4] http://www.nxp.com/documents/user_manual/UM10360.pdf
[5] Aström K J and Murray R 2008 Feedback Systems: An Introduction for Scientists and Engineers Princeton University Press (New Jersey USA)
[6] https://developer.mbed.org/cookbook/PID
[7] https://developer.mbed.org/cookbook/Interfacing-Using-RPC
[8] http://www.mathworks.com/help/daq/ref/analog_input.html