Embedded Platform for Automation of Medical Devices

A. Mendoza Garcia[1], M. Rodriguez Huizar[2], B. Baumgartner [1], U.Schreiber [3], A. Knoll [1]

[1]Technische Universität München, Munich, Germany
[2]Hochschule Mannheim, Mannheim, Germany
[3]German Heart Center, Munich, Germany

Abstract

Embedded systems are becoming of great interest in the medical field. With the reduction of size and increase in processing power, small devices capable of capturing patient data and making control decisions may help in providing patients with better treatments. This paper describes an embedded platform that may be used for the automation of medical devices. A description is given of the hardware and software of this platform. The system may be easily adapted to specific applications with the use of XML based configuration files. At the end a case study of the automation of a heart-lung machine is presented with some preliminary results.

1. Introduction

The use of embedded systems in the medical field is becoming of great interest since it allows the creation of small portable devices that help in the treatment of patients. With the increase of processing power in such devices it is possible to integrate control mechanisms capable of adjusting parameters and take decisions depending on what is sensed from the patient. With the increasing need of medical control systems, the characteristics may be similar for several applications, by using a general purpose reconfigurable platform the time of development may be greatly reduced. This device must be robust and fully reliable, capable of working for extended periods of time without failure.

To enable the system to be easily reconfigurable individual components were designed to be used as building blocks for the complete system. For a specific application, the components required are assembled with configuration files without the need of compilation, eliminating the risk of introducing bugs in the system. Each component can be individually tested for errors reducing the time used for verification.

The time and costs required for the development of an embedded system is significant due to all the components involved, prototyping and testing required. To overcome this, a commercially available embedded platform was considered as a main module of the embedded system. An additional processing unit was included serving as a supervisor of the main module.

On the software layer a Linux open source operating system was used for the main module, providing an abstraction of the hardware with already available features such as file read/write support, network, serial transmission, among others. Only a reduced number of drivers are needed which take care of specific hardware components. This allows the system to be portable between different hardware platforms and may be easily upgraded.

As a control mechanism fuzzy logic was considered since it provides a straight forward way of using rules provided by specialists dictating how the system should respond to the given inputs. A case-study is shown how the platform is configured to control the speed of a centrifugal pump of a heart-lung machine. This is done depending on the current mean arterial pressure (MAP) and extracorporal flow rate (EFR) of the patient.

2. System Description

A medical device requires real-time patient data to be able to take decisions. Vital signals are obtained through various sensors. The sensors signal are then acquired from different interfaces such as analog to digital converters, serial ports, ethernet among others.

The system should be capable of visualizing the different signals to verify correct acquisition and provide user interfaces for sensor calibration and system configuration and control. Logging information is also crucial for future analysis of system behavior and patient reactions. Remote access to the system is possible with the use of an ethernet connection.

Reliability is of great importance. The system must be capable of detecting any failure from the sensors, the hardware components or software. An independent supervisor unit was included, in charge of monitoring the main system and give an alarm in case of failure. Low power consumption was considered to allow the system to be portable and

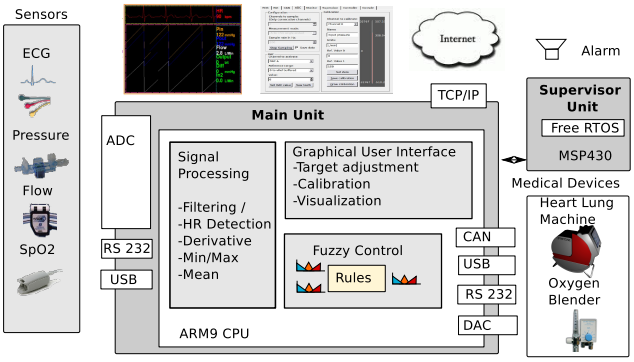

Figure 1. Embedded Platform

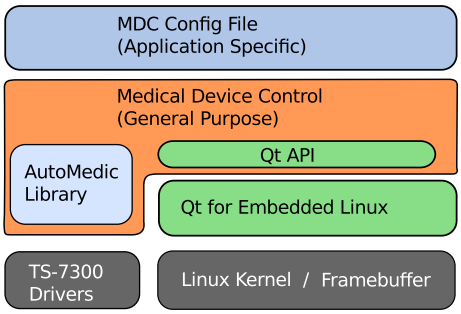

Figure 2. Software Layer

battery powered.

The system is intended to be easily configurable for different applications. A control mechanism is provided where different rules may be programmed to determine the behaviour of the system.

3. Hardware Layer

We used an already built embedded system which complied with the system requirements. Several options where found as evaluation boards [1–3] however they did not fulfill all of the requirements. From Technologic Systems [4] an embedded system was found covering all of the previous requirements. This consists of a TS-7300 board with an 200Mhz ARM920T processor, 2 ethernet ports, 2 USB, 2 UARTs, 2 SD cards, among with a PC/104 Bus for additional daughter boards. The TS-ADC16 board was connected to the PC/104 bus, with 16 ADC channels with 16-bit resolution and four 12-bit DAC channels.

4. Software Layer

The TS-7300 board comes with a pre-configured open source Linux debian based distribution with Kernel version 2.4.26. Having this operating system provides functionality such as ethernet configuration, ssh remote access, sd read/write access. Figure 2 shows the different components of the software layer.

4.1. Main System

Embedded Qt version 4.7 [5] was used to provide a user interface. The source code may be downloaded and configured to be compiled for a specific platform. A gcc cross-compiler provided by Technologic Systems was used to compile the Qt source code.

Using Qt brought the advantage of using the same code to run on the embedded platform and on a normal x86 PC; the only change is the compiler and some of the hardware specific drivers. This reduced the development process greatly since most of the tests and debugging was done on a fast computer and after the program functionality was completed this could be tested in the embedded platform.

4.1.1. Medical Device Control Program

Considering the requirement of configurability a program called medical device control (MDC) was created. The AutoMedic library designed in our group[6] was used as part of the MDC program. A BaseObject abstract class was defined as a fundamental component. This BaseObject is composed of OutputPorts containing a name and a value, and a ConnectPort used as inputs to the BaseObject. The ConnectPort contains a name and a pointer to the OutPort from where the value is obtained as input. All the classes that derive from this object must implement a calculate() method which contains the operations required to generate the output signals. These objects may be used as construction blocks for the control module.

Table 1 lists the different objects that may be used for specific applications.

Input/Output Parameters
Signal Intputs: ADC,ECG,SpO2,Network
Operators: Derivative,Sum,Gain Saturation,Filter,Reference Model,Delay
Control: Basic Controller,Adaptive Controller
Outputs: DAC,Scope,Network,CAN,RS232

Table 1. MDC Objects

The specific parameters for all of the objects are described in XML files that may be easily changed.

Fuzzy logic was used as a control mechanism. This allows the creation of rules that may be provided by specialized medical doctors on how the system should behave [7].

4.2. Supervisor System

The safety and reliability of the system should be guaranteed at all times. System failure may come from different parts of the system. From the input signals obtained from the different sensors safety checks of critical minimum and maximum values are checked at all times. At the same time the control signals are checked to be under a predefined range.

To verify the correct operation of the MDC program and the complete system two supervising components were considered. The first one is based on a software monitoring system that runs in parallel with the MDC. An open-source project called SMART (Smart Monitoring and Rebooting Tool) was adapted to monitor the MDC program. In case a service is not present or stops working SMART automatically restarts the process. The second supervising component consists of an additional micro processor capable of running independently of the main system. Both systems communicate periodically sending each other an alive signal and acknowledgement. A state machine was created to allow both of the systems to communicate with each other. If one of the systems stops responding alarms are generated to inform the user of a system failure.

For the supervising unit a real time operating system was used. FreeRTOS provided the functionality needed and was easily ported to the selected hardware.

5. Case Study: Heart-Lung Machine Automation

This platform was developed as part of a project to automate a heart-lung machine, in previous work we describe the process of automation, and simulation in a mathematical model[8] and tests with a hydraulic model[9]. This embedded platform together with the hydraulic model mentioned was used as a case study to analyze the adaptation and functionality of the system.

An ultrasonic flow probe together with a pressure sensors, were used as analog input signals to the system. The system was configured to use the analog to digital converter to receive these signals. Target values are introduced by the operator. Two fuzzy controllers were used for this application. One containing the rules to reach the target MAP and another for the EFR. The inputs of the controllers are the difference between the target values and the current values and the derivative of each input. The rules determine if the centrifugal pump speed should be increased or decreased. The output of both controllers are added to generate one single value. This value is introduced into an analog to digital converter after minimum and maximum ranges have been verified.

5.1. File configuration

The creation of the configuration files are done by using predefined templates of the objects needed. These are then verified for correct notation.

Figure 3 shows the xml file descriptor for our case study. This generates two input parameters to control Mean Arterial Pressure (MAP) and Extra-corporal Flow Rate (EFR), for these parameters a safe range of maximum and minimum are specified. The controller will try to reach the normal value of MAP and EFR by adjusting the speed of a centrifugal pump, given as PumpSpeed. Following this is the description of the sensor device, in this case using the TS7300 DAQ board, with 2 analog inputs acquired through a pressure sensor and a flow probe and an analog output as a voltage connected to a motor drive of the centrifugal pump. Two fuzzy controllers are specified, both considering the current value and the derivative value. For each input and output the number of sets of the controller are specified and the type of rules to create for each controller. In this case the Fuzzy controller acts as a proportional-

```
<?xml version="1.0" encoding="UTF-8" ?>
<PumpControl>
 <Parameters>
  <Control NumParam="2">
   <Var1 Name="MAP" Units="mmHg"
    Min="50" Normal="70" Max="110" ... />
   <Var2 Name="EFR" Units="L/min" M
    Min="0" Normal="4" Max="5" ... />
  </Control>
  <Output NumParam="1">
   <Var1 Name="PumpSpeed" Units="rpm"
    Min="1000" Normal="1000" Max="3900" ... />
  </Output>
 </Parameters>
 <Sensors NumSensors="1">
  <Sensor1 Name="DAQ" Type="TS7300" Inputs="3"
   Freq="50" Outputs="1" ... >
    <Input1 Name="MAP" Units="mmHg" />
    <Input2 Name="EFR" Units="L/min" />
    <Input3 Name="cPumpSpeed" Units="rpm" />
    <Output1 Name="PumpSpeed" Units="rpm" />
   </Sensor1>
  </Sensors>
 <FuzzyControl NumControllers="2" Type="Mamdani" >
  <Control1 Name="MAP" Inputs="2" Outputs="1" >
   <Input1 Name="eMAP" NumSets="7" Gain= "0.025" .../>
   <Input2 Name="deMAP" NumSets="5" Gain= "0.1" ... />
   <Output1 Name="dPump0" Gain="0.03" RuleType="PI"/>
  </Control1>
  <Control2 Name="EFR" Inputs="2" Outputs="1" >
   <Input1 Name="eEFR" NumSets="7" Gain= "0.25" ... />
   <Input2 Name="deEFR"  NumSets="5" Gain= "0.1" ... />
   <Output1 Name="dPump1" Gain="0.03" RuleType="PI"/>
  </Control2>
 </FuzzyControl>
 <Operators NumOperators="1">
  <Operator1 Name="dPumpSpeed" Type="SUM" Inputs="2">
   <Input1 Name="dPump0" WeightName="wMAP"/>
   <Input2 Name="dPump1" WeightName="wEFR"/>
  <Operator1>
 </Operators>
</PumpControl>
```

Figure 3. MDC Configuration File

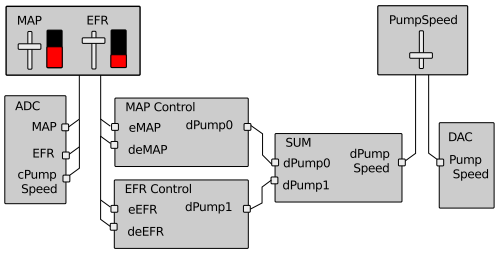

Figure 4. HLM Pump Controller in MDC

integral (PI) controller. At the end a sum operator is used to add the result of both controllers and increase or decrease the pump speed within its upper and lower limits.

5.2. Results

Figure 5 shows the response of the embedded system loaded with the previous configuration file. The controller was set to reach different targets of EFR. A reference signal was used to tell the controller how fast these targets should be reached. The current value was obtained as an analog signal from a ultrasonic flow sensor. The graph in the bottom shows the actual output of the controller. This was converted to an analog signal that was introduced into the pump driver.

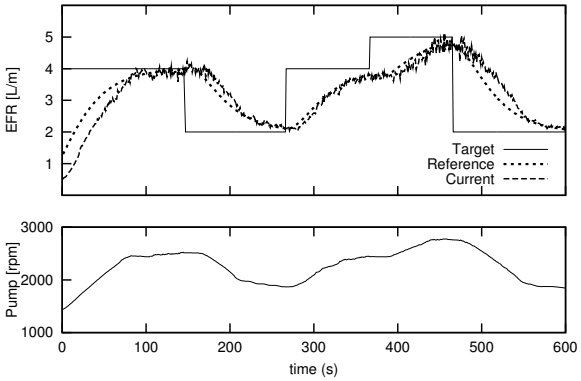

Figure 5. Pump speed control at different flow rates.

6. Discussion and Conclusion

Additional methods for testing the embedded platform are important, One of the tests considered is Hardware-in-the-loop (HIL). This test requires specific models that represent the system to control. For the case study mentioned a simulation model of the cardiovascular system has already been implemented, capable of generating pressure and flow signals and responding depending on changes to pump speeds [7].

The use of an already available embedded solution and an open source operating system reduced significantly the time required for the development of this work. The high speed at which technology advances may bring more optimal options by the time this work has been presented, however by using general components and an independent software layer it is possible to easily port the software to a faster system, using the same configuration files and just adapting the required drivers. The use of Qt brings the possibility of easily porting the software layer to a more robust RTOS. The results of the case study show that the embedded platform can be easily configured and was capable of controlling flow and pressure as expected. Further tests are considered to test for robustness and reliability.

Acknowledgements

This work has been supported by an unrestricted educational grant from the Bayerische Forschungsstiftung.

References

[1] Freescale. MCF51QE128 evaluation board. URL `http://www.freescale.com`.

[2] Keil. MCB2300 evaluation board. URL `http://www.keil.com/mcb2300/`.

[3] IAR. STR912FA development board. URL `http://www.iar.com`.

[4] Technologic Systems. TS-7300. URL `http://www.embeddedarm.com`.

[5] Qt. Embedded linux. URL `http://qt.nokia.com`.

[6] Mendoza G A, Baumgartner B, Schreiber U, Krane M, Knoll A, Bauernschmitt R. Automedic: Fuzzy control development platform for a mobile heart-lung machine. volume 25/7 of IFMBE Proceedings. 2009; 685–688.

[7] Mendoza G A, Baumgartner B, Schreiber U, Eichhorn S, Krane M, Bauernschmitt R, Knoll A. Design of a fuzzy controller for the automation of an extracorporeal support system with the use of a simulation environment 2010;6698 – 6701.

[8] Mendoza G A, Baumgartner B, Schreiber U, Krane M, Bauernschmitt R, Knoll A. Simulation of extracorporeal circulation for the design of a fuzzy controlled perfusion. In IASTED Biomedical Engineering, volume 1,2. 2010; .

[9] Schreiber U, Eichhorn S, Mendoza A, Baumgartner B, Bauernschmitt R, Lange R, Knoll A, Krane M. A new fuzzy controlled extracorporeal circulation system. first results of an in-vitro investigation. CINC 2009;36:497–500.

Address for correspondence:

Alejandro Mendoza Garcia
Technische Universität München
Informatics 6
Boltzmannstrasse 3
85748 Garching
Germany
mendozag@in.tum.de