Curing Cancer, on the Back of LabVIEW

Author
Tim Nolan, Data Science Automation, Inc.

NI Product(s) Used:
sbRIO-9636
LabVIEW 2012 SP1
LabVIEW FPGA
LabVIEW Real Time
LabVIEW Structured Error Handler
SQLITE Toolkit
TestStand 2012

Category:
Advanced Research

Application Area:
Medical Device development

The Challenge
Performing extracorporeal blood treatment on human patients while complying with European Union regulations for Class C medical devices, adjusting to user feedback, and meeting delivery deadlines.

The Solution
Patient risks were minimized by using a single board RIO to manage the system with critical functions being handled by the FPGA. Code was developed with agile software development techniques, and validated delivery with automated regressive testing.

Introduction

For 25 years, Data Science Automation® (DSA) has been a premier automation systems integrator, leveraging commercial off-the-shelf tools in the design and implementation of custom-engineered, complete, and highly-adaptive solutions in laboratory automation, embedded/new product development, manufacturing and test automation. The company provides an extensive array of automation engineering, programming, consulting & training services to dramatically improve research, manufacturing, government & business operations. DSA is fast and methodical, staffed with exceptional, multi-disciplinary, NI Certified professionals that consistently apply CSIA-certified best practices to deliver the lowest total cost of ownership.

It's Not Every Day You Get to Work on a Cancer Treatment

Immunotherapy is a rapidly developing area of cancer treatment and provides an alternative to patients that are untreatable by traditional methods. There are a variety of immunotherapy techniques and each attempts to modify the human immune system in ways that allow it to combat cancer cells. This treatment offers an alternative to individuals who can tolerate traditional treatments but whose cancers cannot be targeted, vital for the millions of individuals who cannot tolerate chemotherapy.

Our client is developing a system that performs extracorporeal treatment on blood, which means that blood is removed from a patient, treated outside of the body, and returned to the patient. This particular method of immunotherapy is designed to remove certain molecules generated by cancer cells that actually suppress the body's own immune response to those cells. By releasing these molecules, cancer cells are trying to protect themselves. By removing these molecules from the bloodstream, our client intends to strip away the cancer cells' protection, and allow the body to fight the cancer more effectively on its own.

Working on Cancer Treatments Requires a Lot of Regulatory Compliance

Our client's system is connected to a patient and fluid supply bags. Blood is circulated through a tubing set and treatment components via motor-driven pumps. The direction and path of flow is controlled by the pump direction and pinch clamps. At certain points the tubing passes through pressure sensors, a bubble detector, and a hemoglobin sensor, all to monitor for conditions that could damage the blood or threaten patient safety (Figure 1). The software is required to control the motors and pinch clamps, and log data from the pressure sensors, bubble detector, hemoglobin sensor, and load cells which monitor the volume of fluid supply bags. The software must also stop flow and close the clamps when an abnormal condition (hereafter referred to as a "fault") is sensed. The software interface must enable the user to switch between several flow configurations that correspond to performing treatment, priming for treatment, pausing, etc.

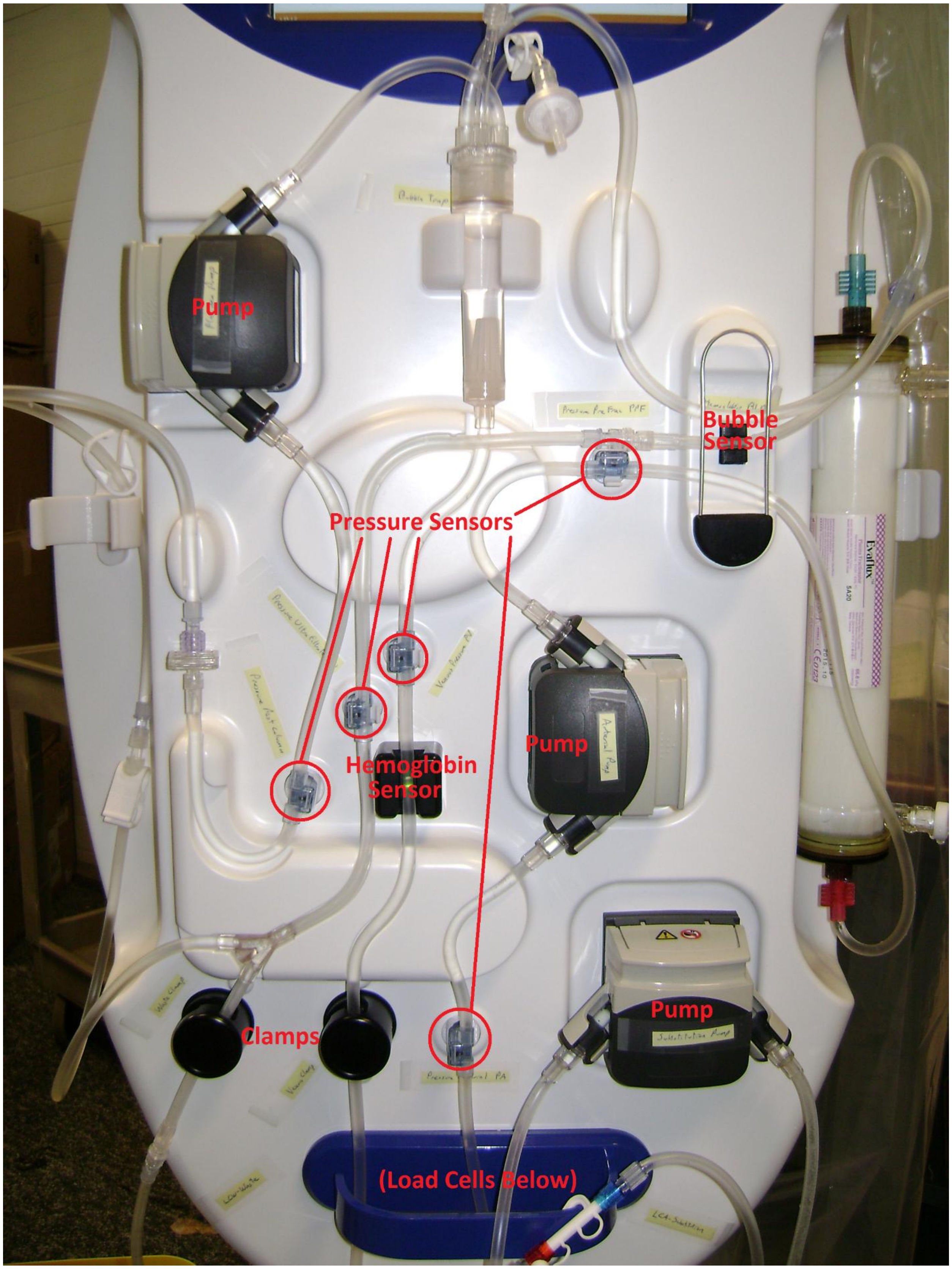

Figure 1 Blood Pumps and patient-protective systems.

These functions alone constitute a fairly generic flow control system. But, to preserve patient safety and comply with medical device regulations, this system requires deterministic responses to faults, as well as the highest possible level of system health and error monitoring. Our client's initial system design – which involved a Windows-based touch-panel PC and USB-based DAQ - was not adequate to meet these requirements, so a number of options were considered: 1) Adding hardware fail safes for every condition that threatens patient safety; 2) Using a real-time OS for critical tasks in conjunction with Windows; 3) Utilizing a sbRIO to handle critical tasks while using the touch-panel PC for user interaction (Figure 2).

Medical devices have of course classically been hardware-centric, but have increasingly incorporated software as the reliability of software platforms has matured, and can be demonstrated. In the case of our customer's device, the hardware only option was rejected because it offered the least flexibility in design. The development path for this product included multiple cycles of system modifications in response to clinical feedback, which would have been nearly impossible if using solid state circuitry only. *Rapid software updates offered a cost-effective method of iterative release superior to a hardware implementation.*

A sbRIO was implemented because of the maturity of the technology and the track record of NI RIO products. The high reliability of FPGA (in addition to the determinism of the real-time software) also contributed to judgment that a sbRIO would be most likely to assure regulatory compliance and maximize patient safety. Due to regulatory restrictions, we had to lock the release version of LabVIEW to the initial version of **2012 SP1** for the entirety of the project.

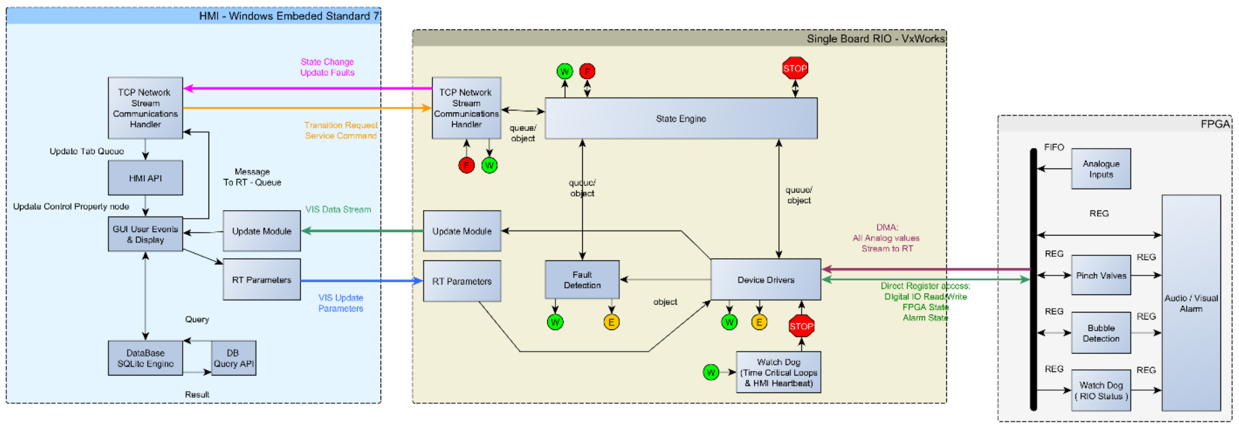

Figure 2 Software modularization and distributed computing.

Multiple Targets to Target Multiple Risks

The system software resides on three targets: the FPGA, the RIO, and the touch-panel PC (hereafter referred to as the HMI). The RT code manages the main functions of the system, and schedules all tasks for each treatment. The FPGA code handles the majority of the hardware interactions and critical patient safety functions. The HMI code is reserved for only the least critical functions and user interaction. This architecture fits well with the system's paradigm for patient safety and regulatory compliance. All of the highest risk faults were handled at the FPGA level, with immediate patient protection in clamping blood returns and stopping pumps. The RT and Windows system displaying lower priority faults, which did not need immediate hardware action.

This distributed control was managed by independent code modules for each functionality, the main State Machine, Analog IO, Logging, Reporting etc. The Data Science Automation Non-Blocking Variable library was used for seamless data transfer between parallel modules.

The RT code contains the majority of the systems functions including a central state machine. Data is exchanged with the FPGA via DMA's and the FPGA front panel. This includes RT drivers for each piece of hardware that obtain raw data via the

FPGA. The FPGA and RT also monitor each other's health via watchdog signals. The RT code includes a deterministic loop whose primary purposes are fault monitoring and data collection. This handles fault prioritization and reporting. The RT Fault monitor also has access to the FPGA for FPGA-level faults, so that mitigating action can be taken on the RT level (changing to a safe state and informing the user via the HMI, and resetting faults on the FPGA after user intervention). The remainder of the RT code including the central state machine are non-deterministic. Finally, the HMI code handles user interaction and other low-priority tasks such as logging system data to a database.

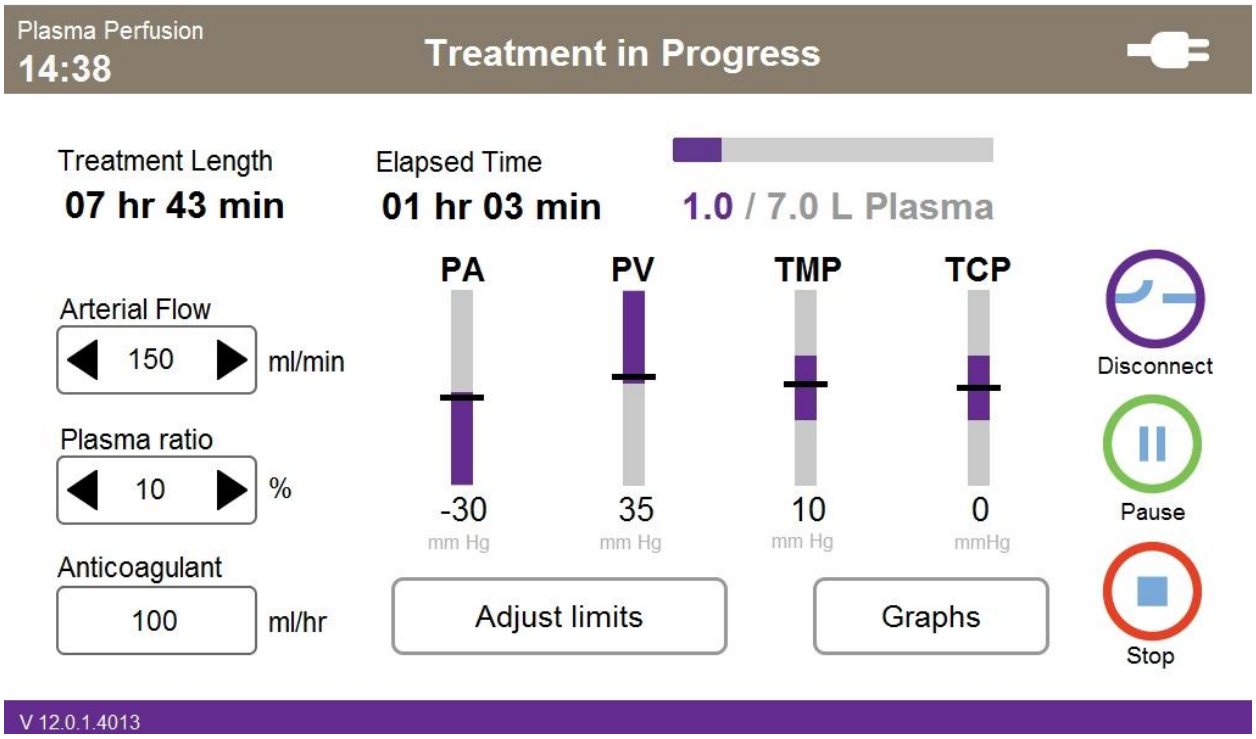

Figure 3 Commercial User Interface designed in LabVIEW. 60601 compliant for alarming and safety.

The Payoff

The system is currently being delivered after successive development releases. In each of those cycles certain sets of functionality were delivered to the customer and clinical feedback was provided, so that corrections could be made in the following cycle in addition to adding new sets of functionality. The design paradigm allowed multiple developers to work on the software simultaneously: at least 1 developer on both the FPGA and HMI source code, and at least 2 developers on the RT code. This ability of a multi-developer team to deliver incremental functional designs in two-week cycles allowed DSA to respond to the customer's aggressive delivery schedule. The device has passed Class C Medical device approval for the EU, and is undergoing commercialization.

Contact Information

Timothy Nolan,
Manager, Product Engineering
Data Science Automation Inc., 375 Valley Brook Road, Suite 106, McMurray PA 15317
(724) 942-6330
tdn@dsautomation.com
www.DSAutomation.com