TUPV039

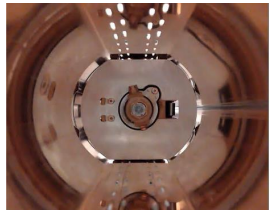
A RELIABLE MONITORING AND CONTROL SYSTEM FOR VACUUM SURFACE TREATMENTS

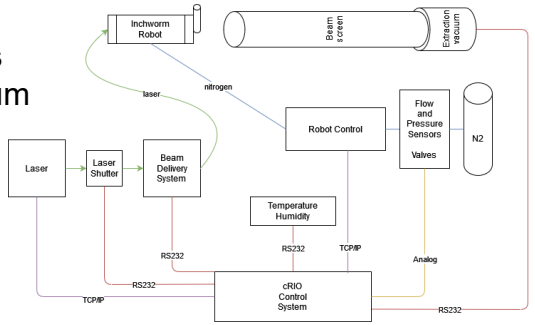
J. Tagg, E. Bez, M. Himmerlich, A. K. Reascos Portilla, CERN, Switzerland

ICALEPCS 2021

Project Description

The objective of the LESS (Laser Engineered Surface Structures) project is the commissioning of an in-situ **laser surface treatment** conceived to mitigate electron clouds in the Large Hadron Collider (LHC) at CERN. **Secondary electrons** are multiplied when they interact with the vacuum chamber walls of the accelerator and consequently form electron clouds that can negatively affect its performance.

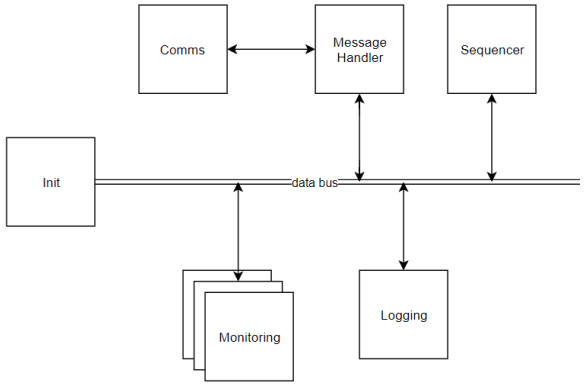




Architecture and Implementation

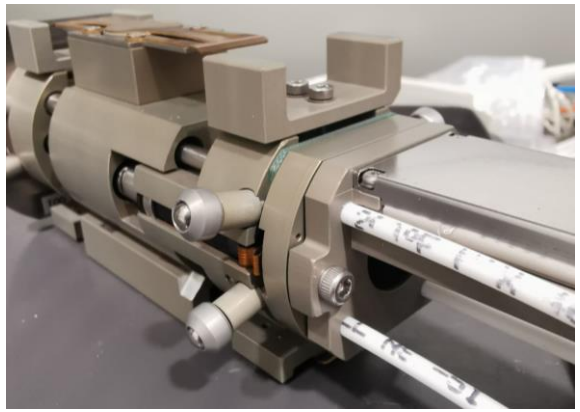
The software is based on a **custom design** which favors 'locality' and emphasizes the use of standard debugging tools. Actions are always run close to where they triggered.

Data is shared among all running processes to give them easier access to any useful information they may need.



Sequences

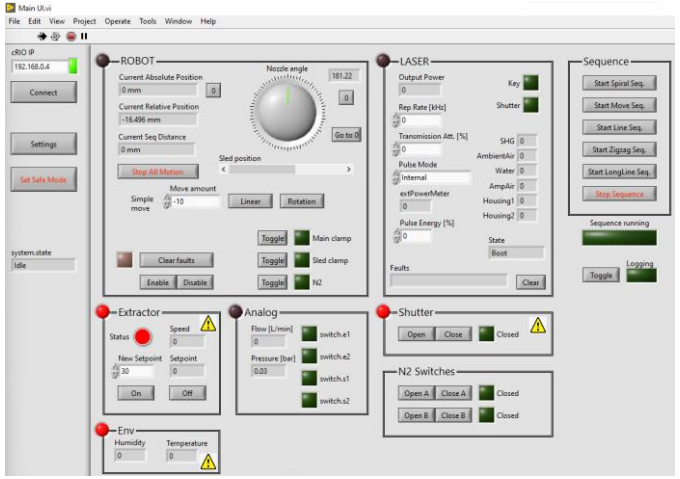
A simple **state machine architecture** is used to run the sequences. One of the main roles of a sequence is to control the inchworm robot as it alternates clamping and movement. The sequence state machine encapsulates all knowledge of how to perform the sequence and can be easily modified to tweak its details and logic.



Conclusions

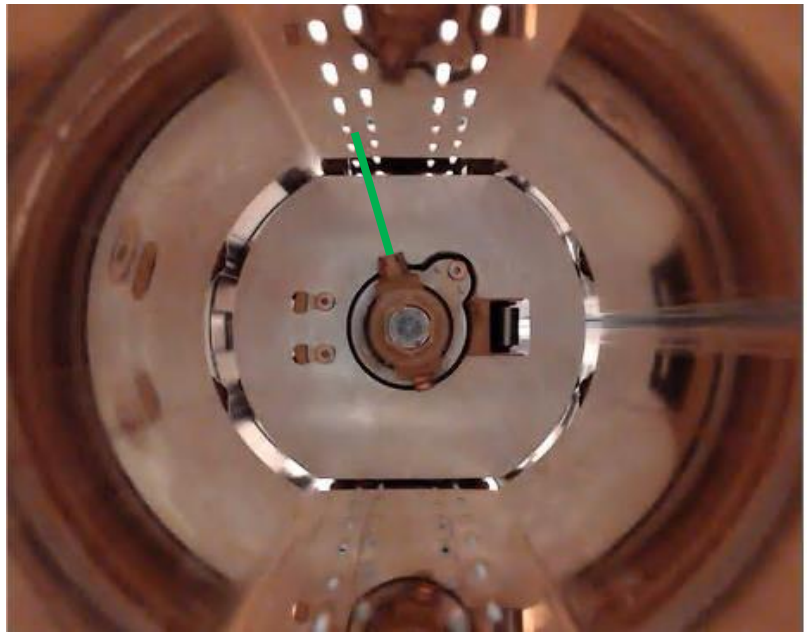
Reliability is ensured through:

- Good practices
- Reliable hardware
- Redundancy
- Safe state on any error that isn't explicitly handled

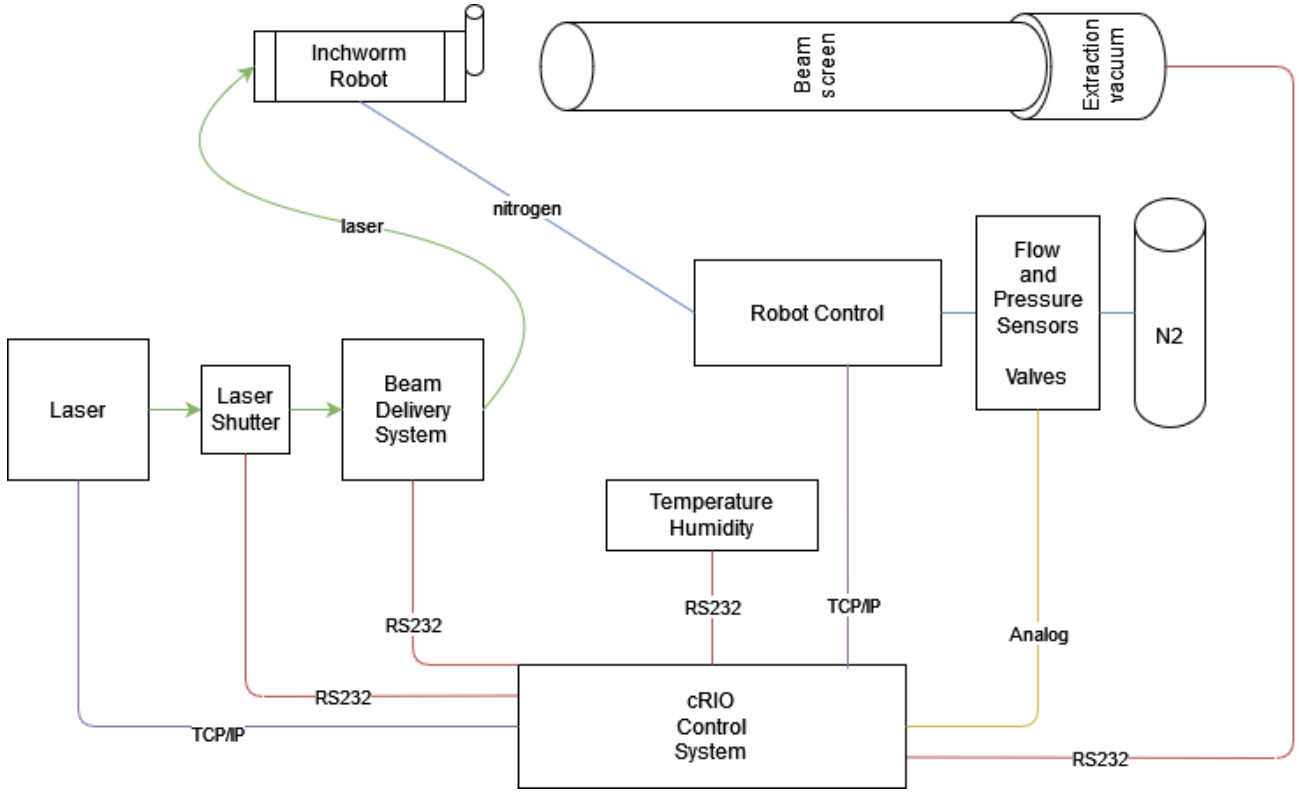


Project Description

The objective of the LESS (Laser Engineered Surface Structures) project is the commissioning of an **in-situ laser surface treatment** conceived to mitigate electron clouds in the Large Hadron Collider (LHC) at CERN. **Secondary electrons** are multiplied when they interact with the vacuum chamber walls of the accelerator and consequently form electron clouds that can negatively affect its performance.
The secondary electron emission of a surface can be reduced by **surface roughening**.



Robot inside the beamscreen (longitudinal view) with a simulated laser beam.



Main components

- Pulsed laser
- Inchworm robot
- Nitrogen atmosphere
- Environmental readouts
- Extraction vacuum

Interfaces

- RS-232
- Ethernet / Telnet
- Analog and digital IO

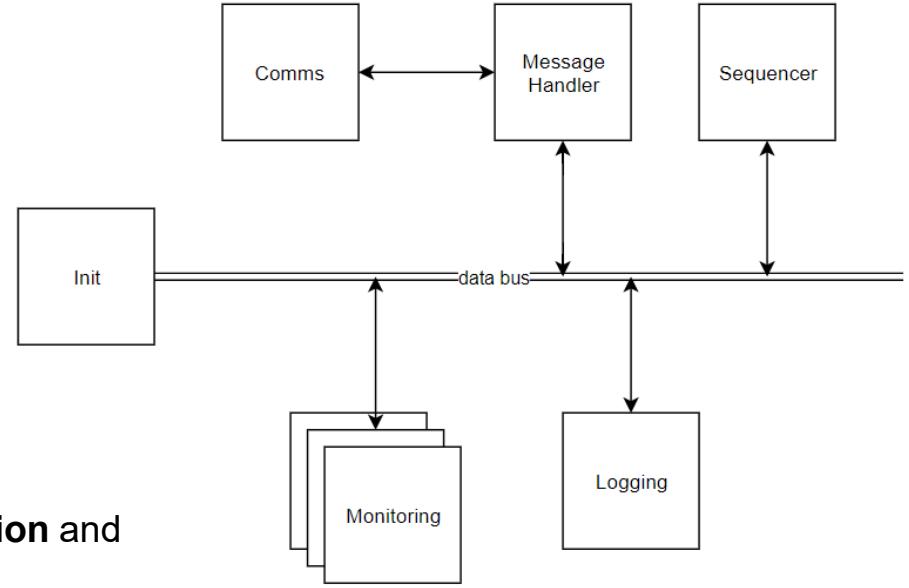
Architecture and Implementation

Programming frameworks considered

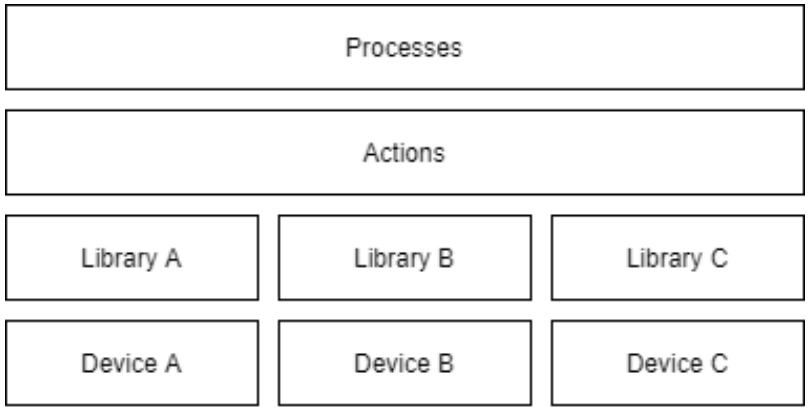
	Readability	Debuggability	Prototyping	Multiple instances
CVT	yes	yes	yes	No
DCAF	With experience	Not directly	Takes planning	Yes
QMH	Can be	With experience	Nothing built-in	Yes
DQMH	Lots of boilerplate code	Good testing tools	Scripting tools for quick creation of functions	Cloneable modules share some resources
Actor Framework	With experience	Difficult in LabVIEW real-time	Slow to deploy	Yes

All frameworks impose ideas and structures, which in small projects can be constraining. We went with a custom solution to optimize the 'locality' of the code. The main goal was to make it more **readable** by removing intermediate steps between cause and effect, and to make it **debuggable** using LabVIEW's built-in tools.

The program is structured around the idea of a **central, shared data set**, which is passed to all running processes. While this gives great freedom to processes to use any data they need to perform their tasks, care is taken to avoid race conditions. This is done by carefully deciding which process writes to which data point, and keeping to a **single writer** wherever possible.
Most processes acquire data only and make it available to other processes.



Software architecture

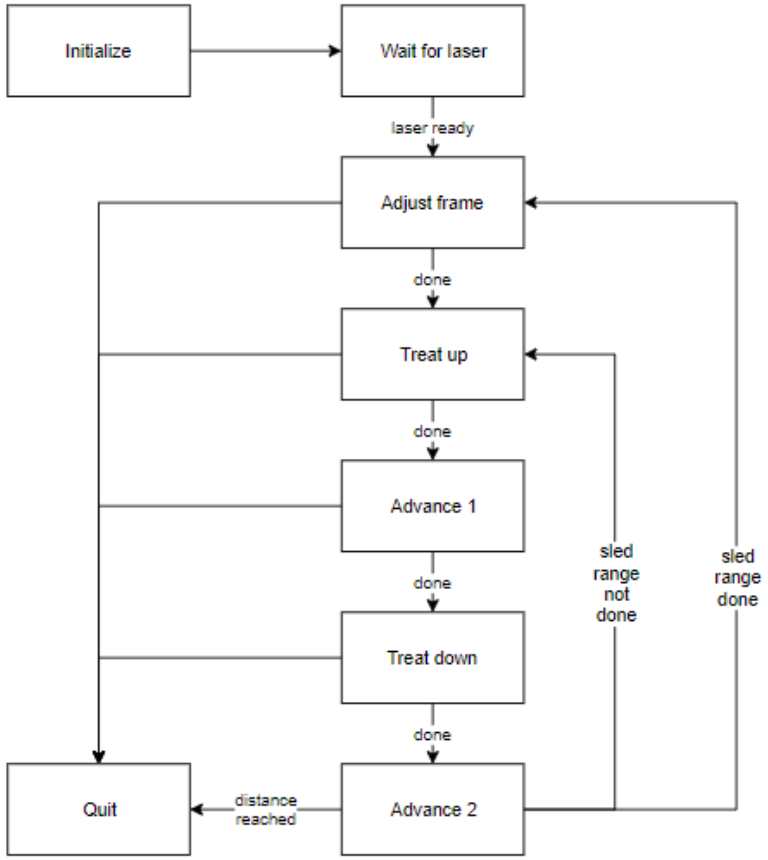


Software layers

The program is designed in layers.
The lower layers are **hardware abstraction** and are unaware of the wider program state.
Actions are state-aware snippets of processing that the application can run within processes. Actions and processes work with the **full dataset as input** and can be nested; they represent the main integration layer.

Sequences

The program is able to run different sequences, each one moving the robot in a specific way. The purpose is to measure their relative performance by analyzing the samples after treatment.



State diagram of the line sequence

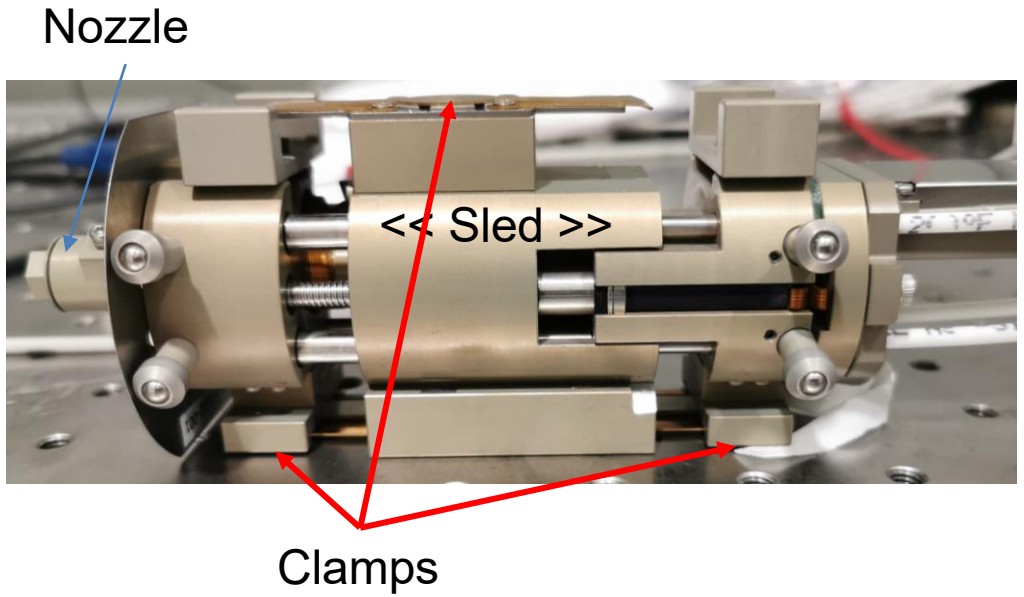
Sequences

Spiral

Line

Longitudinal

Zigzag



Main parameters used for sequences

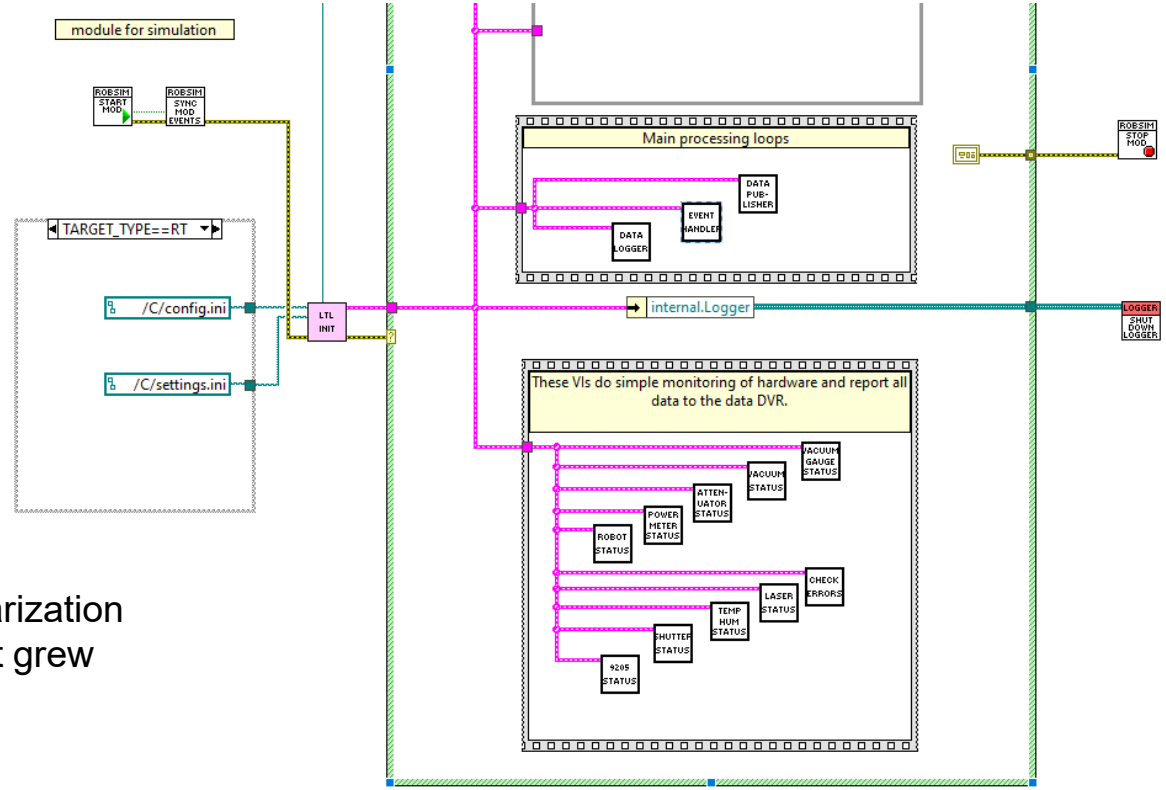
- Closeness of laser tracks (pitch)
- Speed of treatment
- Speed of adjusting movement
- Length to treat
- Range of internal sled movement
- Radius of beamscreen
- Treatment angle

Conclusions

Reliability is ensured through:

- **Good practices**: limit writers of data, modularization, simplify where possible
- **Reliable hardware**: NI CompactRIO running NI Linux RT OS
- **Redundancy**: duplicate important hardware (switches, laser shutter)
- **Safe state** on any error that isn't explicitly handled
- **GUI locked** during sequence to only allow minimal interaction

Delays on hardware integration mean we were not able to fully test the system yet. Focus was on testing the sequences and the robot movement.



Benefits of the custom architecture

- Readable: when something happens, it happens right there
- Ease of debugging using standard LabVIEW tools
- Access to all data wherever needed
- Very fast to deploy
- Small changes and tweaks can be tested in a very short time

Limits of custom architecture

- Limited complexity: race conditions will appear without stricter modularization (manageable on small project, but would need to change if the project grew bigger).