TDR7

Control Systems
Technical Design Report

Turkish Accelerator Center
IR-FEL and Bremsstrahlung Facility

Authored By: Teoman Soygul
Date: 8/8/2010

Table of Contents

List of Tables

List of Figures

Acronyms

FEL	Free Electron Laser
IO	Input/output
IOC	Input/output Controller
PLC	Programmable Logic Controller
IPC	Industrial PC
CA	Channel Access
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
LAN	Local Area Network
GUID	Globally Unique Identifier
VS	Microsoft Visual Studio
WCF	Windows Communication Foundation
TPL	Task Parallel Library
ADO	ActiveX Data Objects
ASP	Active Server Pages
PLINQ	Parallel Language Integrated Query
CLR	Common Language Runtime
CLI	Common Language Infrastructure
CPU	Central Processing Unit
BPM	Beam Position Monitor
DAQ	Data Acquisition
ADC	Analog Digital Converter
RAID	Redundant Array of Independent Disks

1 Introduction

Turkish Accelerator Center is anticipated to have one of the most modern implementations of distributed control systems used in any nuclear research facility in Europe. The complete control architecture is implemented as a soft real-time distributed control system based on EPICS (Experimental Physics and Industrial Control System) software with various EPICS compliant hardware ranging from IPCs, PACs, to PLCs. With a distinguished two-tier approach, the backbone of the control network infrastructure is fiber optic 10 Gigabit Ethernet while gateways provide data access for LAN workgroups and web clients. This two-tier approach completely isolates the stable control network from the other workgroup networks where possible software glitches, viruses, and other problems may clog the data links. As a backup to fiber optic 10GbE, a separate copper Ethernet is deployed with redundant switches for all the distributed control nodes. Whenever necessary, custom control software is developed (and deployed) using native EPICS .NET Library on the client workstations at the control room or the IOC (input output controller) nodes throughout the accelerator hall. Thus software architecture is standardized on a typical client-server model on each node, backed up with publish/subscribe messaging paradigm throughout the control network. With the integration of all the subsystems, the project is expected to deliver exceptional performance, scalability, and reliability in less than twelve months' time. When completed, the project will mark a milestone for future nuclear research laboratories in terms of the industrial grade reliability and IT level of technology of its control system.

1.1 Scope

The purpose of this document is to provide an overview of Turkish Accelerator Center's IR-FEL and Bremsstrahlung facility's distributed control systems. Starting with the technical requirements, anything materializes the control systems is discusses up to a certain extend. The system's hardware components, network infrastructure, and software architecture are all described in an orderly fashion. On the other hand, since this document is intended to be a report on the technical design and not the technical design itself, separate design drafts provide a more in-depth view of the problems discussed here and implemented solutions. Low-level details into each components design are provided in separate documents. This document is intended to be the starting point of any research on internal structure of the control systems of IR-FEL & Bremsstrahlung facility, by the responsible personnel.

1.2 Overview

The technical design of the control systems is completed with partial implementation on the injector portion of the accelerator. The technical design is not conceptual only, meaning that control hardware and software are intact. This way, implementation of the control system on the remaining sections of the facility will be nothing more than replicating the work that was done for the injector sector. It is expected to transform this complete design into a complete implementation in the original 12 months' time allocated for the project. Anticipated cost of realizing this design is around €80.000, which is much lower than €500.000 budget allocated for this project. At this point, to account for iterative development methodology, implementation of the project and attendance to all related conferences will be carried out in parallel to provide a constant stream of feedback from industry experts around the world.

2 Project Requirements Analysis

This project requires two major components to function, which are the hardware systems that make up the control network, and the distributed control and computing software to be used along with the required hardware. While the accelerator hall requires intensive control and supervision, control room requires crystal-clear process visualization to provide a friendly user interface to a complex system. The prime concern of this project is to provide industrial grade long-term stability and reliability while making use of the innovative technology. This way, the project can be a breakthrough in technology amongst other nuclear research facilities. Any requirement that is a direct outcome of the given objectives can be grouped either as technical or functional requirement. Functional requirements are the desired set of behaviors of all the components of the control system. On the other hand, technical requirements are the design goals that are to be met to achieve the desired system behavior with regard to scalability, stability, performance and other factors, which govern the system response.

2.1 Functional Requirements

The first and the foremost functionality of all the control related hardware and software is to supervise the ongoing processes within the facility, which are particle acceleration and nuclear experiments in this case. This supervision is provided via control loops, which control the dynamic behavior of the system. Even though almost all the control loops are distributed and implemented on large scale via distributed computing architecture, they still closely resemble traditional electronic control circuitry. Essentially all the internal control logic is one big workflow, which is divided into subsections and developed with Windows Workflow Foundation 4.0. The following image describes such a control feedback loop, which is implemented as workflows as explained in detail at section 3.2.2.

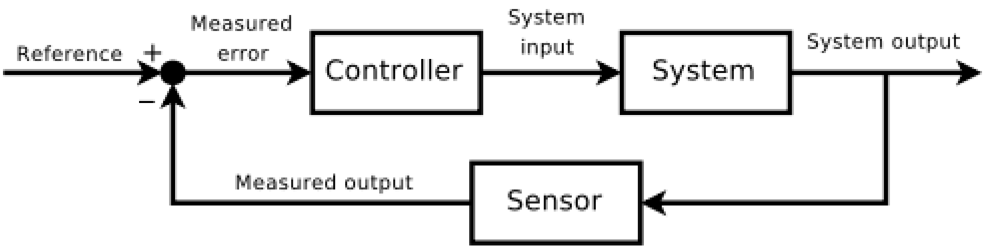


Figure 2-1 Control feedback loop [1]

Probable control system scenarios and behavior expected in specific situations are modeled using UML activity and use case diagrams [2]. While some certain responses are expected of the system in certain cases, the most important of these responses are informing the operators in case of dangerous situations, protecting the machines from damage, and providing safe shutdown sequences. These expectations can be summarized in a simple activity diagram, such as the following.

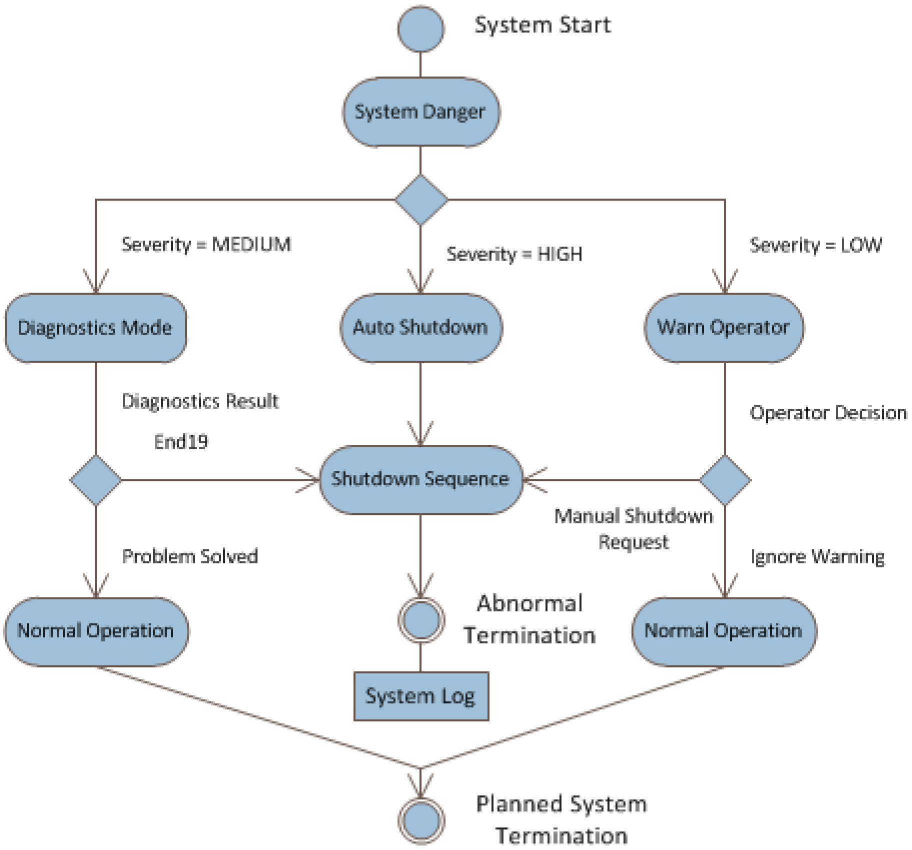


Figure 2-2 Machine protection activity - response diagram

As the diagram exhibit the behavior of system in cases of different threat levels, user-machine interaction may be much complicated than it is shown in the activity diagram. In these sorts of cases, possible interaction scenarios were modeled via use case diagrams. Below is such an example of user-machine interaction, depicting the system response in a hazardous situation.

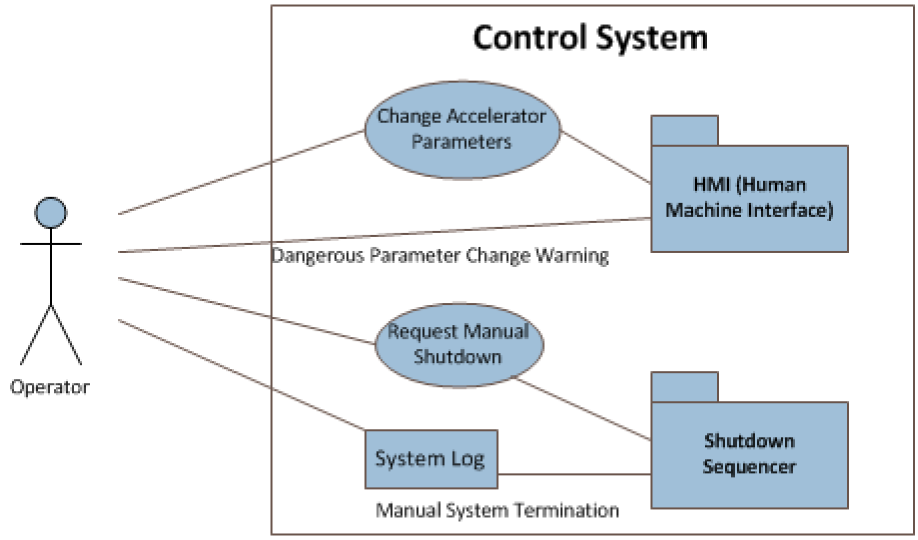


Figure 2-3 UML use case diagram modeling a sample user-machine interaction of the control system

In this section of the document, only the methods used in capturing the functional requirements are given and neither the diagrams nor the full lists of requirements are presented since they would span a couple of hundred pages. On the other hand, most important functional requirements captured during the analysis can be listed as:

- Providing fast responses to system warnings
- Providing good crisis management mechanisms
- Implementing intrusion protection and prevention
- Implementing self-diagnosis
- Having large bandwidth and QoS implementation for the network
- Integrating LDAP for user and group management
- User security certificate management
- Providing historical operator logs

2.2 Technical Requirements

As it was mentioned earlier, once all the hardware requirements are categorized under the control network and all the software requirements are put under control software category, all the other requirements naturally drive from these two.

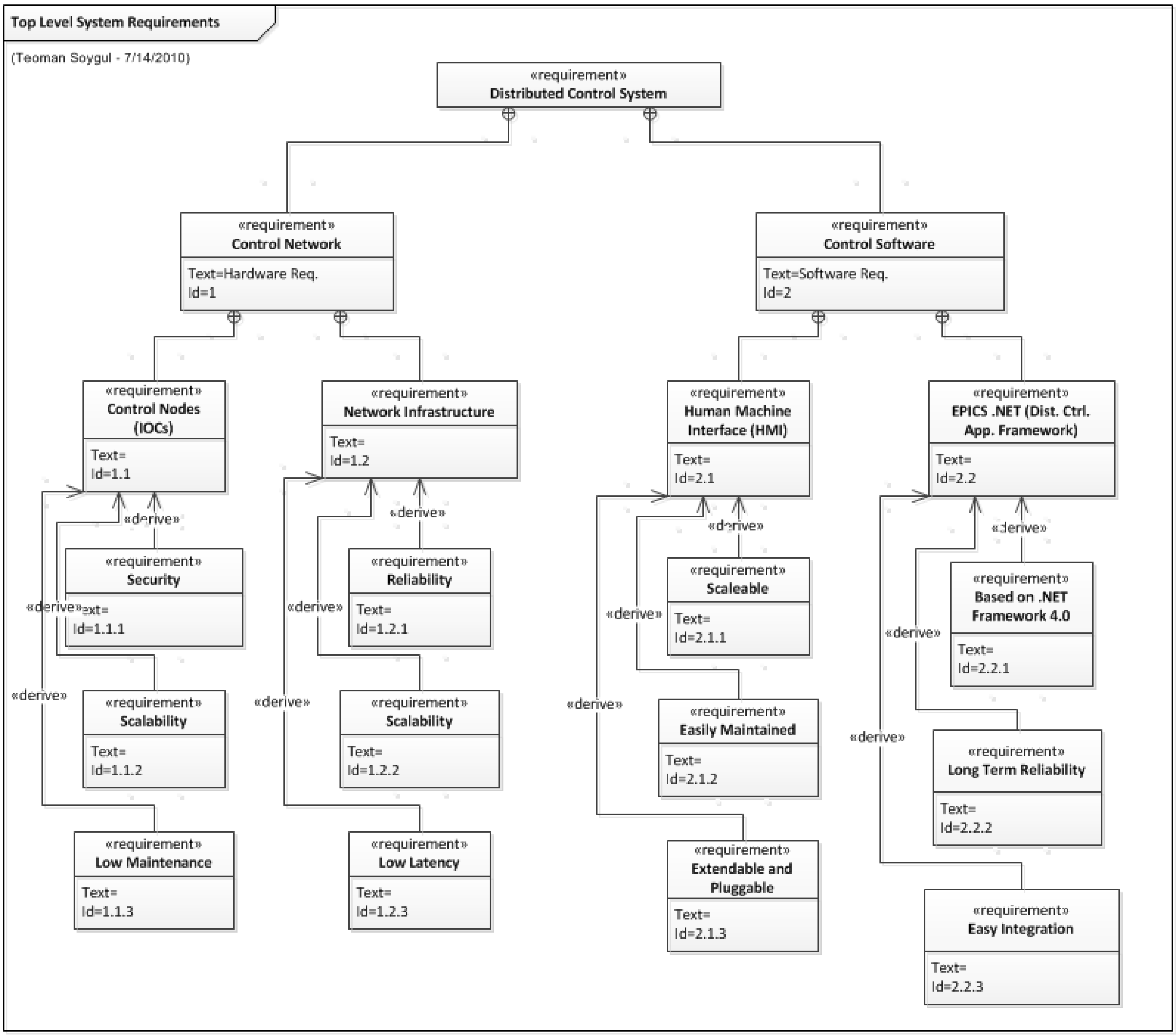


Figure 2-4 Top level system requirements

Taking the functional requirements into account, the above figure summarizes the technical requirements of this project while following sections provide a more in-depth analysis of these requisites.

2.2.1 Architecture

The definite requisite of the system architecture is scalability, which can be achieved through distributed computing and control models. Decentralizing the operation center (i.e. the brain) eliminates one of the most common single point-of-failure challenges hence removing the need for a centralized control server. What replaces a control server in this case is a reference server where each control node connects to get the latest software updates and such. Naturally, a control server would have limited system resources (i.e. processing power, hard-disk space, and memory) and running into those limitations in long term is almost unavoidable. In addition, trying to add more resources to a centralized server is much more costly than adding same amount of resources to distinct computing units. Even Google favors cheap and distributed computing over centralized and expensive servers [3]. Once the processing power and control responsibility is distributed over a farm of computers, scaling the whole system is as easy as adding new control computers to the network along with new network switches.

2.2.2 Reliability

Reliability is paramount. Considering the sentence "nuclear research" is in the project definition, high availability is of essence. Even though there are secondary and tertiary machine and personnel protection measures, primary control system must be in charge at every step of the way. Due to this, redundancy of the chosen control hardware is crucial. Choosing computers with redundant power supplies, fans, and hard disks in Raid 1 sort of a failsafe operation mode is vital. Even having a secondary network is necessary for both load balancing and fail safety. In addition, distributing the system to distinct control nodes is important for reliability as well as scalability. If a local control node fails, it will not be affecting the operation of the rest of the nodes, which are always autonomous and self-contained.

2.2.3 Performance

Performance bottlenecks can bring a system to a halt, which contradicts reliability. In addition, in case of system crisis retaining extra system resources plays a great role in faster system recovery. Another need for speed is for the times when the system is extraordinarily overburdened due to an unexpected number of node failures. In this situation, another advantage of using distributed control would be the performance benefit, which is very cost effective due to using higher quantities of cheaper components. While cost effective hardware is adequate solution for hardware-level performance issues, at the software level things need to be handled by application profilers. Profilers are the ultimate solution to performance bottlenecks in software whenever throwing more processing power at the solution may not be the cure. Also there are various other types of system performance analyzer tools that can be used to predict possible performance shortcomings.

2.2.4 Security

Both hardware and software level security is important for a control system that is intended to be operational 24/7. Any control network should be safe of any hacker intrusion or virus infections. This can certainly be achieved through separating the control network from LAN workgroups. Software security needs to be provided through either code access security, using permission tables or role-based security implementations.

2.2.5 Safety

Due to environmental constraints, not every control hardware can be fitted in any desired corner. Especially the accelerator hall is a high radiation area and gamma and x-ray radiation are known to decay silicon chips. This means that control nodes to be used within the accelerator hall needs to be compact and should fit in-between the protective lead blocks. In addition, high energy radiation is known to deteriorate the transparency of fiber-optic cables so fiber cables used in high radiation areas should be replaced in every 2 to 3 years or special shielded cables should be opted in for specific sectors.

2.2.6 Integration

Whenever possible, any control hardware or software library used in the implementation of the system should be compliant with third party tools. This way, preexisting equipment can easily be integrated into the control system.

2.2.7 Maintenance

As usual, low maintenance is always a plus. Especially, choosing the hardware systems with regard to technical personnel in hand will always prove advantageous. Low maintenance hardware will mean less need for technicians and less training and retraining sessions.

3 System Design

Overall system design is crafted to satisfy all the functional and technical requirements, which were previously mentioned in the project requirements analysis phase. Many of the diagrams used in capturing the requirements were altered and then combined to transform them into a single components diagram, which is seen below.

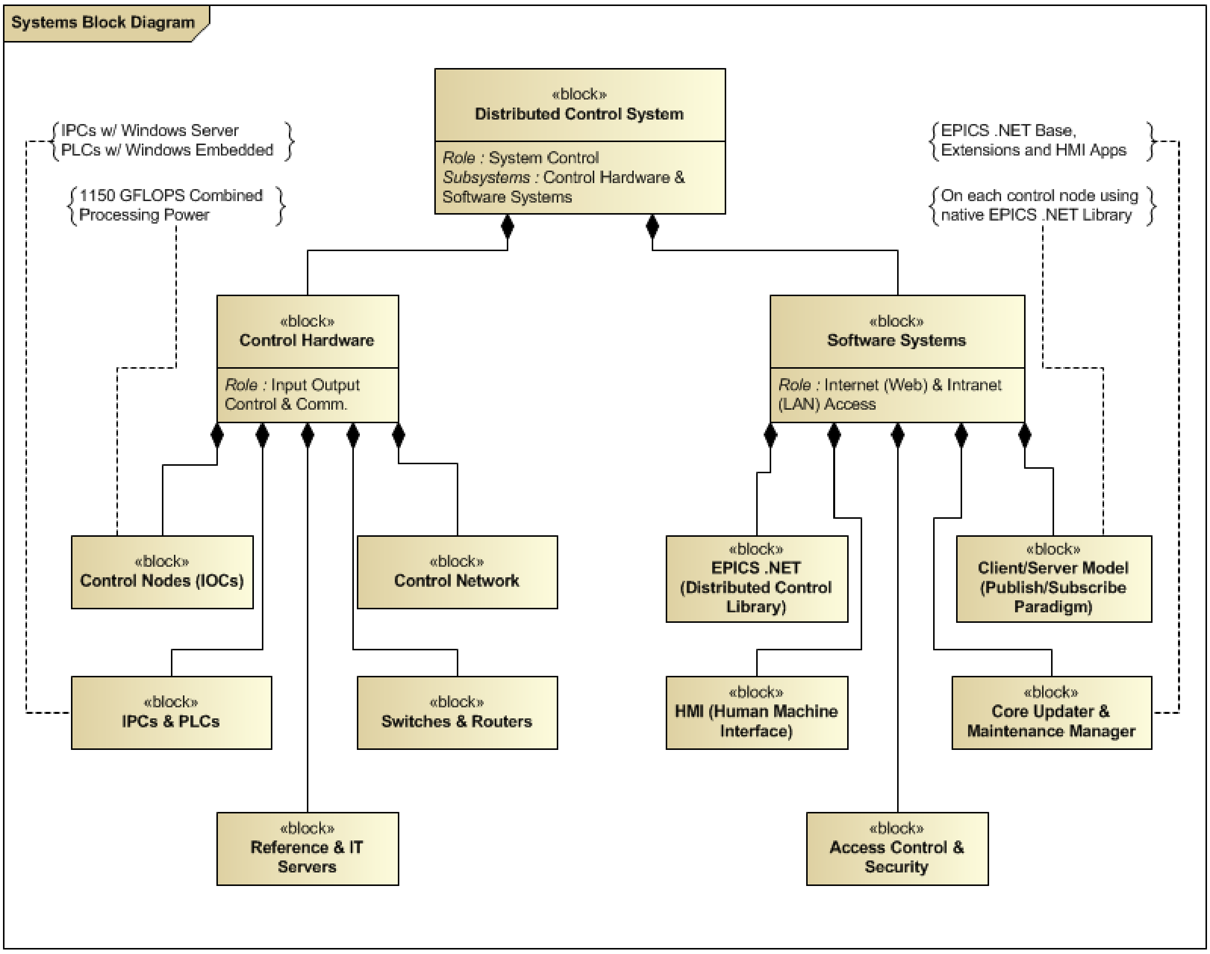


Figure 3-1 Overall system components

This illustration visualizes all the major components of the design of distributed control and computing system. To keep the design as simple as possible, not all the technical requirements are transformed into system components, rather they show up as separate considerations at the design and implementation of each component. Following sections provide a more elaborate insight into the systems design. All of the designs involving the control hardware and related technologies are group under "*Distributed Control Systems Architecture"* title while the all the software used for the purposes of distributed control and computing are listed under the "*Software Systems Architecture*" title.

3.1 Distributed Control Systems Architecture

For any industrial grade control system, PLCs (programmable logic controllers) are invaluable. Their reliability is unmatched. They also provide a very cost effective solution to many classical control challenges (like high voltage switching). With the advance of technology, some of the new PLCs are now equipped with Embedded PCs which are equipped with a general purpose CPUs in addition to conventional ladder logic processors. This sort of PLCs were chosen the task in this project due to their ability run Windows Embedded OS on their general purpose CPUs, which provides the capability of running required software for distributed control and computing tasks. In conjunction to the safety requisites, these PLC boxes can be used within the accelerator hall inside lead boxes. Implementing PLC units as input/output controllers was successfully accomplished for the first time at J-PARC [4] then at KEKB [5] [6] at Japan. A sample configuration is illustrated below.

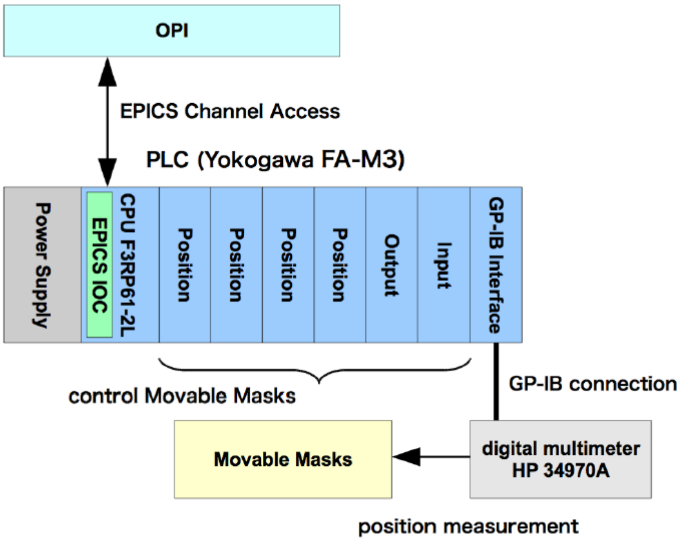


Figure 3-2 Configuration of PLCs as IOCs at KEKB [7]

In addition, industrial PCs with slot count 14 and more were chosen for the tasks that require a high amount of I/O and plenty of processing power. The details of these selections are discussed in the coming sections but their importance is that they make up the control nodes of the distributed control system, which make up the kernel of all the control operations. Thinking about the fact that in distributed systems there is no central control server, these control nodes make up the control server entity as a whole.

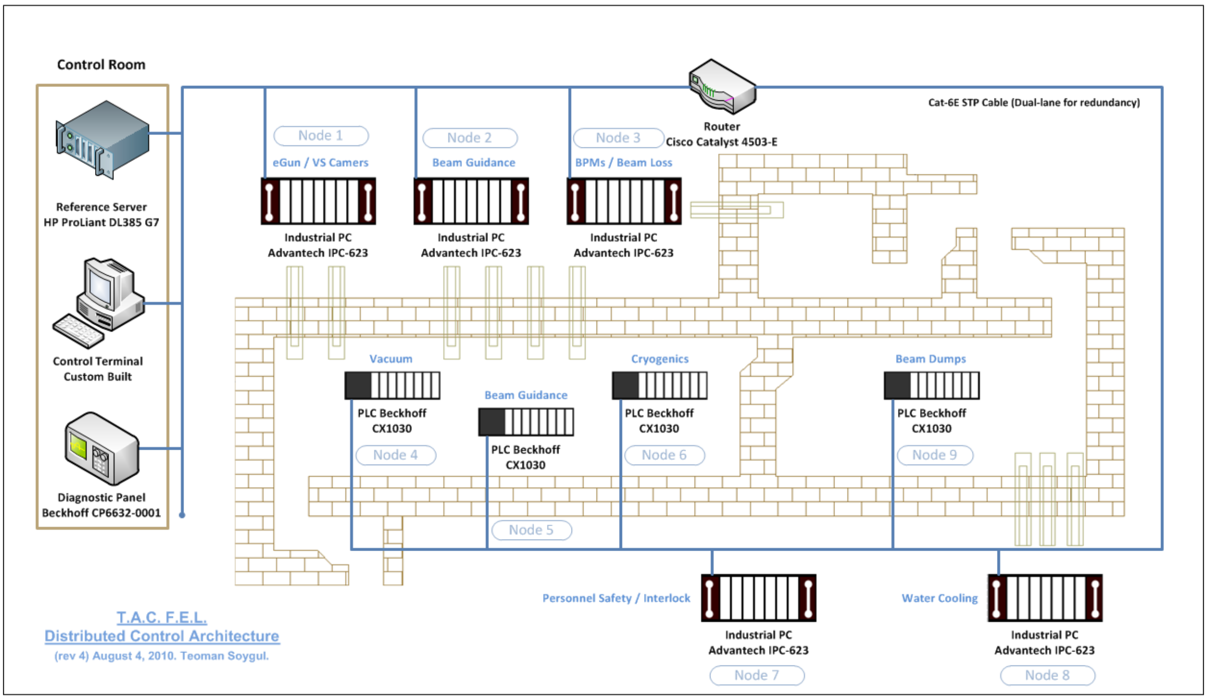


Figure 3-3 TAC Distributed control architecture

On the figure above, all the control nodes are distributed and the existence of reference server is only to provide the nodes with software updates and static reference data. Network redundancy is achieved through the usage of double lane Cat-6E STP (shielded twisted pair) cables in dual lane configuration. Additional fiber-optic high-speed network dedicated to the industrial PCs are illustrated in detail in the upcoming sections. In addition, the reasoning behind the hardware choices is given below.

3.1.1 Input Output Controllers and DAQ Hardware

In this distributed architecture, input output controllers (IOCs) are the computing units with attached data acquisition modules (DAQ). These computing units may be full-blown computers or PLCs with general-purpose processors, as it was indicated beforehand. For this project, two distinct types of controllers were chosen to function as IOC boxes. These two types of units were chosen for different configuration scenarios, and their location within the facility is described by the previous Figure 3-3. In essence, operation of the control nodes only depends on the core router. Failure of the reference server or some of the nodes does not affect the functionality of the rest of the system. Below is a very simple representation of the control network architecture along with the chosen units of controllers. Following sections build on this architecture to describe and define the control network infrastructure.

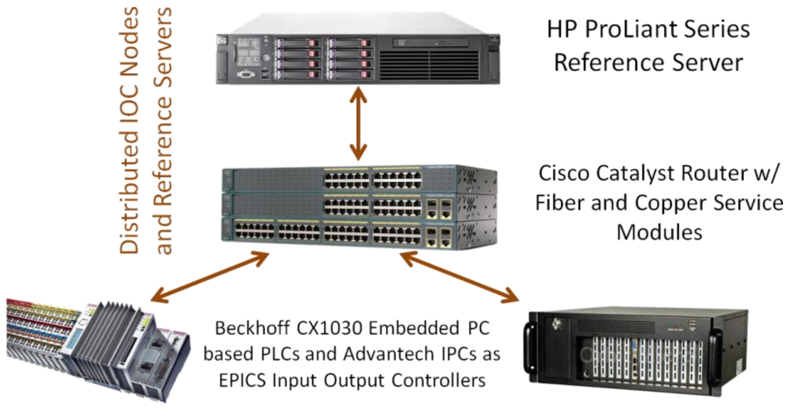


Figure 3-4 Basic control network architecture along with IOCs

Above basic schema describes how IOC boxes communicate with each other and how they retrieve software updates from the reference server. The details of chosen input output controller unit types that are shown on the schema, is given below.

Table 3-1 IOC Based on Embedded-PC PLC

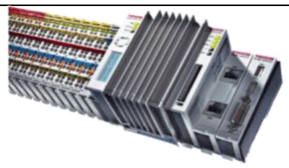
	Beckhoff Embedded-PC PLC With CX1030 CPU Module [8]
Processor	Intel® Pentium® M, 1.8 GHz clock frequency
RAM	1 GB Max
Storage	8 GB Compact Flash
Interfaces	2 x RJ 45 (Ethernet, internal switch), 10/100 Mbit/s
Diagnostics LED	1 x power, 2 x LAN link/activity, TC status, 1 x flash access
Expansion slot	1 x Compact Flash type I+II insert with eject mechanism
Operating system	Microsoft Windows Embedded Standard
Dimensions (W x H x D)	96 mm x 121 mm x 98 mm
Average cost of CPU units	~2000$
Average cost of modules	50$ to 300$

Table 3-2 IOC Based on Industrial PC

	Advantech IPC-623 Chassis [9]
PCI Express Backplane	BPX6806 | PICMG 1.3 | 20-Slot PCI Express Backplane [10]
System Host Board	MCXT-E | PICMG 1.3 System Host Board (SHB) [11]

CPU	Two Quad-Core Intel® Xeon® Processors - 5400 series with 1333MHz FSB, 2x6MB L2 cache
Memory	8 GB four-channel Fully Buffered DIMM (FBDIMM) DDR2-667
Ethernet Interfaces	Intel® 82563EB Ethernet controller - Two 10/100/1000Base-T Intel® 82563E Ethernet controller - One 10/100/1000Base-T
Hard Disks	2x Western Digital Enterprise 1 TB in RAID 1 Configuration
Power Supply	2x Redundant 750W
Fiber Optic Network Adapter	Emulex OneConnect OCe10102-N 10GbE NIC [12]
Average Cost	2000$ to 3000$

Table 3-3 DAQ Hardware

	Various
Average Cost for IPCs	200$ to 1000$
Average Cost for PLC Modules	50$ to 300$

Along with the IOC and DAQ hardware, additional network components and their details are listed below.

Table 3-4 Reference Server

	HP ProLiant DL385 G7 6172 1P 8GB-R P410i/256 Hot Plug 1 SFF 460W PS Server (573088-001) [13]
Processor	AMD Opteron™ Model 6172 (12 core, 2.1 GHz, 12MB L3, 80W)
Memory	16GB Advanced ECC
Network Controller	(2) 1GbE NC382i Multifunction 4 Ports
Storage Controller	(1) Smart Array P410i/256MB
Form Factor	2U Rack
Projected Price	3500$
Turkey (Ankara) Branch	Servus Ankara [14]

Table 3-5 Core Router

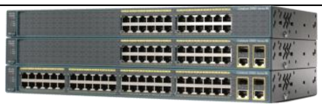
	Cisco Catalyst 4503-E [15]
Total Number of Slots	3
Supervisor Engines Supported	Supervisor Engine 6-E, 6L-E, V-10GE, V, IV, II-Plus-10GE, II-Plus, II-Plus-TS
Number of Power Supply Bays	2
Integrated PoE	Yes
Slots	2x 24 100Base-TX, 1x 5 10GBase-SR
Projected Price with Units	~6500$

Table 3-6 Workstations and Control Terminal

CPU	AMD Phenom II X6 1090T Black Edition Thuban 3.2GHz Six-Core
CPU Cooler	Tuniq Tower 120 Extreme
Thermal Paste	OCZ Freeze Max Performance Termal Gel (3.5g) - OCZTFRZTC
Mainboard	Gigabyte GA-890FXA-UD5 AM3 AMD 890FX SATA 6Gb/s USB 3.0 ATX AMD Motherboard
RAM	Kingston 2GB 1333MHz DDR3 Non-Ecc CL9 Ram KVR1333D3N9/2G
Hard Drive (System)	Western Digital Caviar Black WD5001AALS 500GB 7200RPM 32MB Cache SATA3.0 Gb/s 3.5"
Hard Driver (Secondary)	Seagate Barracuda 7200.11 ST31500341AS 1.5TB 7200 RPM 32MB Cache SATA 3.0Gb/s 3.5"
Monitor	Samsung T260 25.5"
Graphics Card	PowerColor HD 4850 1GB GDDR3
Power Supply	Ocz Fatality Serisi 550W
Chassis	Cooler Master RC-690-KKP2-GP
Chassis Fan (Front)	Zalman ZM-F3 120 mm
Chassis Fan (Exhaust)	Arctic Cooling Arctic Fan 12 PWM
DVD Drive	LG GH22NS50 22X SATA
Keyboard	Microsoft Natural Ergonomic Keyboard 4000 (Wired) - B2M-00021
Mouse	Microsoft Comfort Mouse 4500 - 4FD-00003
Mousepad	Addison White Optic Mousepad - Black
Power Cord	A-Link Power Cable Black 1,8M

Direct Power Cord	Digitus Power Cable - AK-503
Ethernet Cord	Cat6 UTP Ethernet Cable 3M
DVI Cable	Inca DVI to DVI 1.8MT IDD-01
Aggregate Price	2500$

Table 3-7 Diagnostics Panel

	Beckhoff CP6632-0001 [16]
Display	15-inch with alphanumeric keyboard and touch screen
Slots	1 Mini PCI slot
Processor	Intel IXP420 with XScale 533 MHz
Ports	2 USB, 1 RS232
Ethernet	2 RJ 45 Ethernet connectors 10/100 Mbits
Average Cost	700$

3.1.2 Network Infrastructure

In accordance with the technical requirements, network security and performance along with reliability is taken to be the priorities of the design. Counting for the performance requirements, backbone of the control network is planned to be a fiber optic 10 Gigabit Ethernet while the LAN workgroups are to communicate on standard Gigabit copper Ethernet over Cat-6E cables. To provide redundancy in control network, along with fiber optic Ethernet, a two-lane Gigabit Ethernet is deployed with Cat-6E STP (shielded twisted-pair) cables. The STP cables are more expensive but they are shielded from electromagnetic and RF interferences. The complete design is illustrated with the below figure.

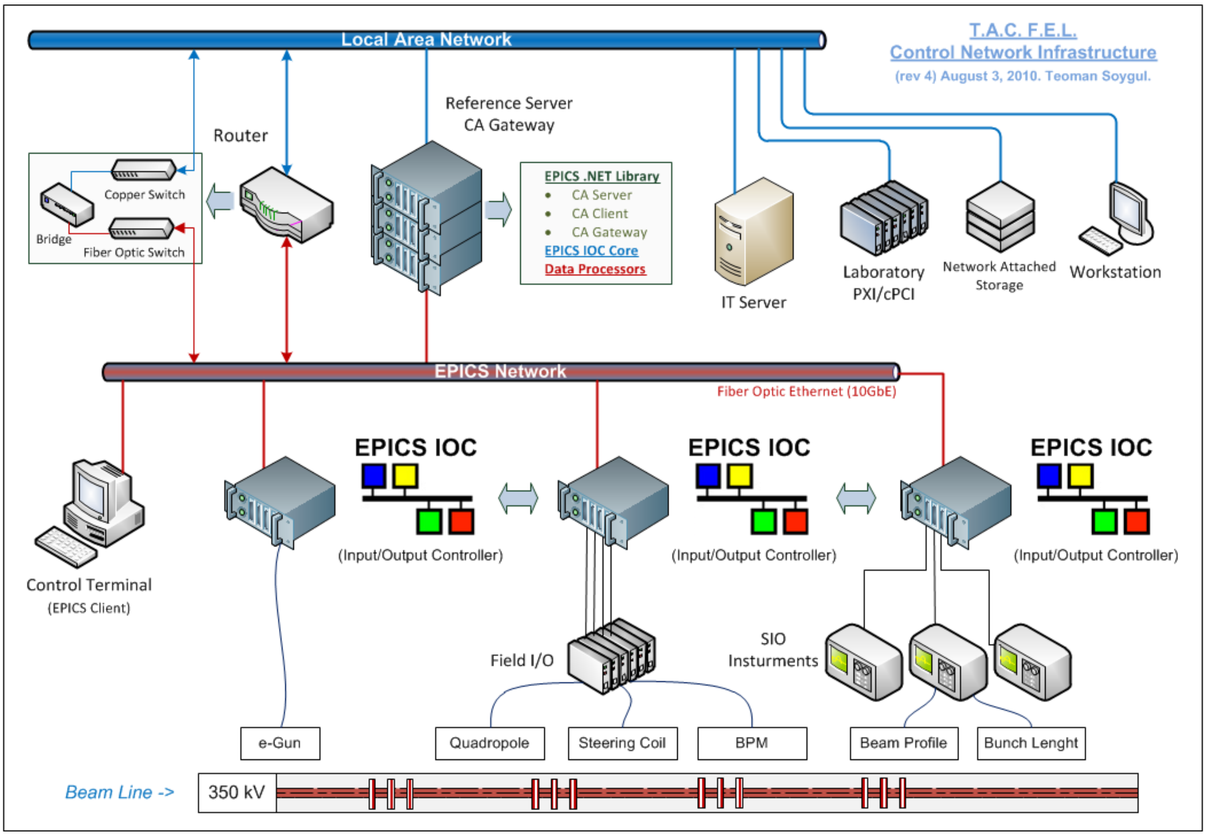


Figure 3-5 Control network infrastructure diagram

The figure depicts a distinguished two-tier approach to separating the control network from the rest of the workgroups, which are on a separate LAN. Any computer that is on the LAN workgroup is meant to access data on the control network through the gateways. This two-tier approach completely isolates the stable control network from the other workgroup networks where possible software glitches, viruses, and other problems may clog the data links.

3.1.3 Communication Protocols

Since the control network is separated from the workgroup LAN, interfaces to these networks are also separate. Control network can easily be accessed making use of the EPICS .NET Library which will be described in the upcoming sections. On the other hand, workgroup LAN access does not require a special library as it is a standard TCP/IP network. Finally, the internetwork communications can be made by the means of gateways. Although direct access can be granted for performance reasons, gateways provide the ultimate security and separation between the networks. In addition, any Fieldbus controller can be added to control nodes to bridge a CAN Bus or PROFIBUS link with the control network. One thing to note here is that unlike legacy systems, PLCs, which are used as input/output controllers in the control system, does not require any Fieldbus technology to communicate with the control network. Rather they join the EPICS network directly, making use of their general purpose CPUs and the control library.

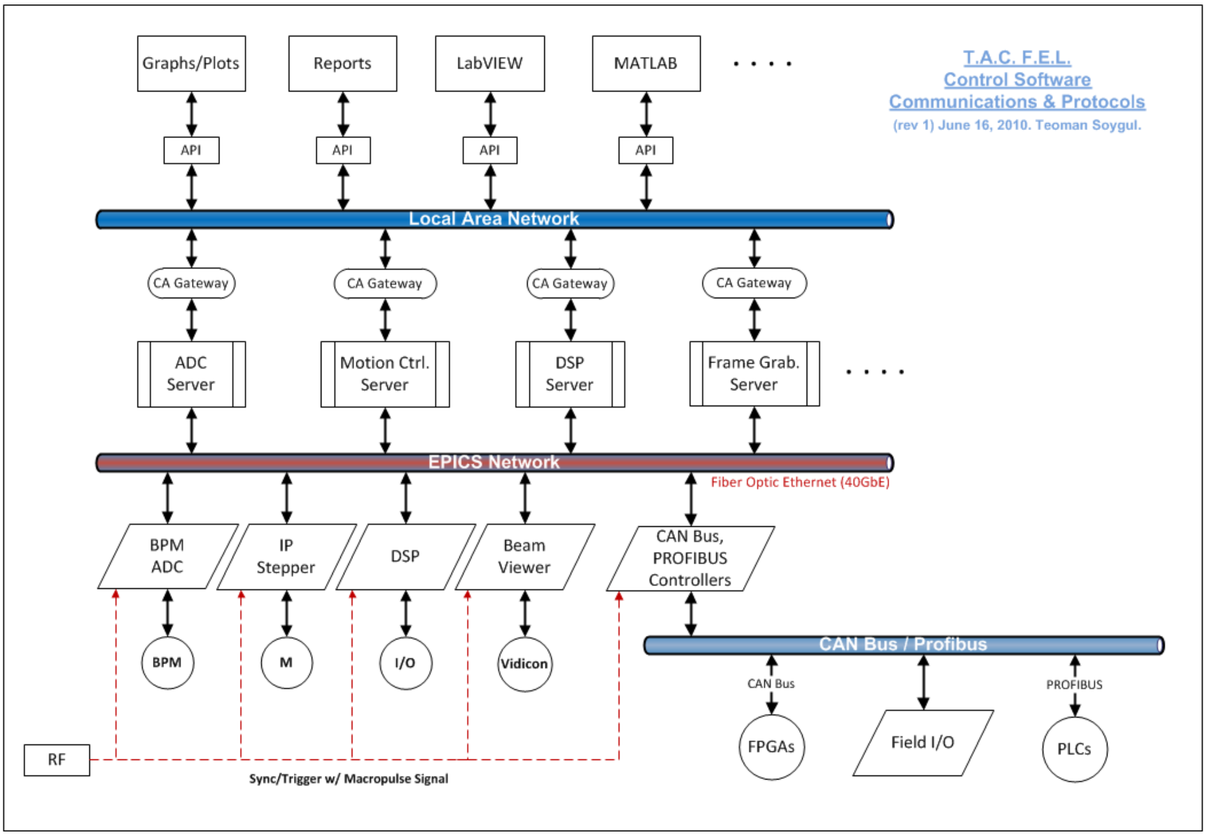


Figure 3-6 Control software communications and protocols diagram

On a special note, applications like MATLAB or LabVIEW has the ability to directly access the EPICS network with their own implementations of channel access. Due to this, they are not bound to run on workstations but they may also be installed on control nodes whenever necessary.

3.2 Software Systems Architecture

Due to the reasons listed at the technical requirements section (and just like the hardware systems), the control software architecture must be a distributed computing standard while not sacrificing performance for scalability. While each autonomous control node should be able to communicate with each other through network, nodes should not be dependent on each other for operation, which makes them truly autonomous, and failsafe. Although each node should have the ability to broadcast the network with certain information, most of the communications should be through direct point-to-point links. On the other hand, for security reasons the workstations may not be able to get point-to-point links with the control nodes if not mandatory, rather the preferred way to communicate with the control network should be through a gateway whenever possible. It is essential to note that, this type of indirect communication adds considerably to the network overhead so it should be used with care. Below diagram clearly describes the intended distributed computing architecture.

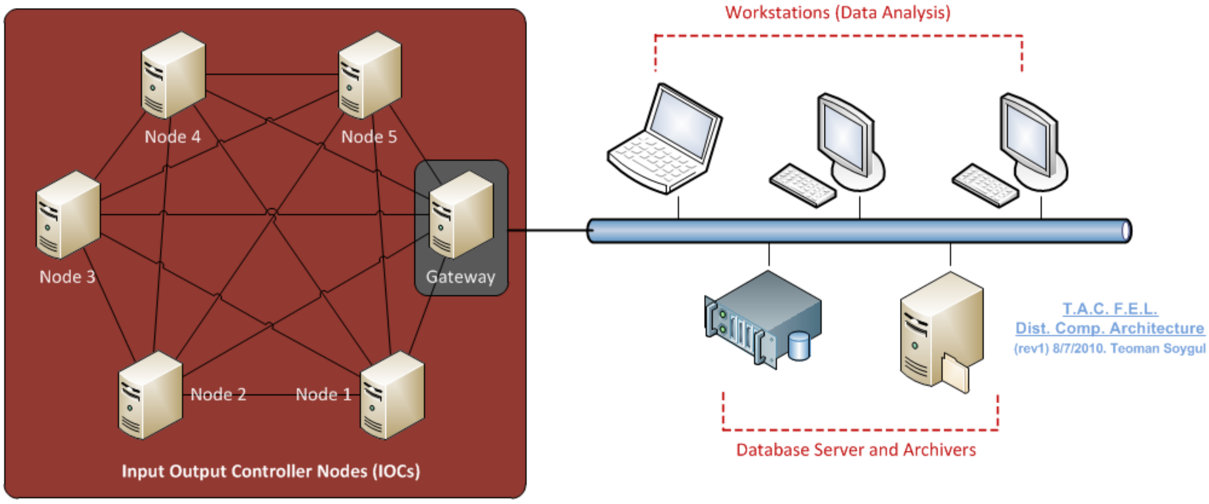


Figure 3-7 Intended distributed computing architecture for the control system

It is important to note that the development of the required software for this sort of a complicated architecture is a long-term business and may take a considerable amount of time. Development process should adhere to the basic rules of software development life cycle. Although a typical waterfall model [17] is not the prime model that is used at in-house software development, it is used in conjunction with the iterative model for most of the time. Thinking about the fact that most of the software developed at this facility is also intended for open-sourcing, added input from other users is thought to accelerate the development cycle for any new version of the applications released under GNU General Public License v3 [18]. The reasoning behind this choice of releasing most of the software development efforts as open source is to provide a notable contribution to the software used nuclear studies, especially in the particle acceleration field. In addition, the core software used in many of the distributed computing operations is already open source and needs community help, which this project also aims to provide. Input to the open-source community and feedback from it is essential at every step of the software development process.

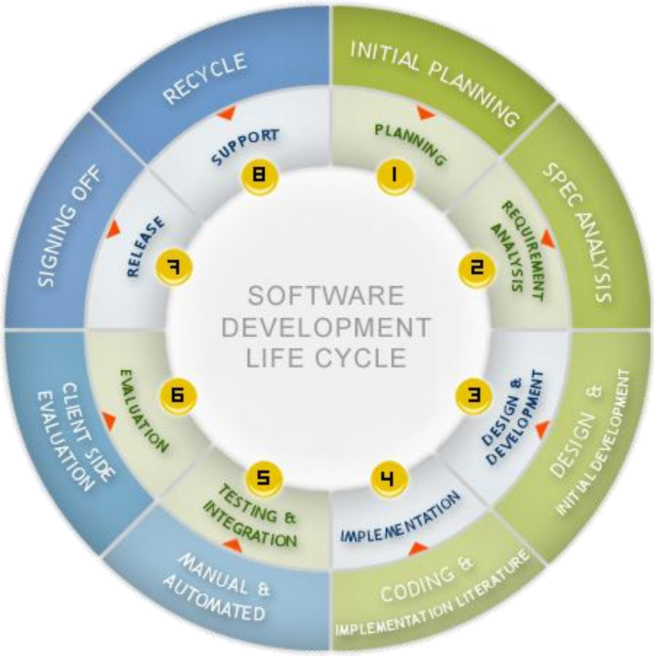


Figure 3-8 Base model for software development life cycle used for in-house software development [19]

Each planned release for software is proposed to go through a complete development cycle to ensure quality. The planning stage always depends on Microsoft Project Pro 2010 [20] while the bug reporting and issue tracking is handled by another open-source project Trac [21]. As it will be mentioned extensively in following pages, implementation is done using Microsoft Visual Studio 2010 [22] with .NET Framework 4.0 [23] and Developer Express UI components [24]. Software testing depends on built-in unit testing capabilities of Visual Studio Ultimate edition thus, no additional tool is necessary. ReSharper [25], a Visual Studio add-on, is used to ensure code quality and the code documentation is prepared with Sandcastle documentation compiler [26] while the users manuals are authored with Help & Manual [27]. Finally, the deployment is done using Microsoft ClickOnce deployment technology [28].

3.2.1 EPICS .NET Library

Distributed computing requires low latency and high-speed node-to-node communication. Normally this would require source and destination node IPs to be known by the application at the run time. To remove the burden of dealing with node IPs and resource IDs (or the GUIDs) of the programmer, EPICS .NET Library was developed for this project. This library deals with all the low-level details of TCP/IP and UDP connections between nodes, beaconing, broadcasting, packet transmission and losses, and many more of small but critical functions of distributed computing. Abstracting away all the low-level technical details, developers can focus on the functionality rather than heft technical issues. This library is essentially a .NET Framework compatible version of conventional EPICS (Experimental Physics and Industrial Control System) [29] libraries and tools. On the other hand, it is not a direct port of legacy EPICS. Since the latest major release of EPICS, software was almost 10 years ago and continuous advances in computer sciences resulted in new and innovative technologies. One of the most important of those technologies, Microsoft .NET Framework, makes up the foundation of this brand new distributed control and computing library. Leveraging the full potential of .NET Framework, EPICS .NET Library makes use of many essential tools like Windows Communication Foundation for communication routines, PLINQ and Task Parallel Library for task and data parallelism, ADO.NET and Entity Framework for distributed database tasks, ASP.NET for data publishing, and many more of libraries that are offered natively with the framework. In addition, many of the programming techniques offered by the CLR (like generic programming for type safety or extension methods and classes for backwards compatibility) and advanced C# programming constructs and language features are used throughout the library. The library is written purely in C# with 100% managed code thus maintaining full CLI compatibility. This way, along with this library any available .NET programming language can be used (C#, Visual Basic, C++/CLI, LISP .NET, IronRuby, IronPhthon, and many more) [30]. This library is fully compliant with .NET Framework 4.0 and upcoming Mono 2.8 [31] with minor modifications.

Figure 3-9 New EPICS .NET Logo

Initial development version of the EPICS .NET Library was based on Paul Scherrer Institut's semi complete implementation [32] with .NET 2.0. Even though this library did not have a proper implementations for CA server or gateway, it served well as a technology preview during the initial planning phase to get a better understanding of system requirements and possible challenges that would arise during the development phase. With its new logo, new home page [33] [34], and complete implementation, Turkish Accelerator Center's EPICS .NET Library will be fully ready for community use as of this year. It is also in full production level use at the free electron laser facility for all of the control tasks as it is described in the following chapters of this document.

EPICS .NET Library uses a typical soft real-time distributed computing model. One major concern of building a real-time system on a system that runs on CLR which behaves just like the Java virtual machine, with lots of memory garbage collection going on randomly, was unpredictable pauses in the runtime engine due to garbage collection. This concern is pretty much eased with the introduction of garbage collector notifications in .NET 3.5 SP1, which gives the opportunity to pass on the control tasks to other control nodes or local applications instances in the face of an upcoming garbage collection session. In addition, the background garbage collector introduced in .NET 4.0 eliminates the random pauses in the applications threads with some minor limitations.

The distributed computing model used by the EPICS .NET Library should not be confused with parallel computing. In a parallel computing system, a single workload is distributed to a farm of computers, which is totally separate from what this library uses. EPICS .NET distributes the center of operation (i.e. the brain of the system) to separate nodes in order to maintain scalability and fail-safety. The task parallelism is achieved locally on each node using the System.Threading.Tasks.Parallel library thus quad core processors of nodes are utilized at full capacity. Note that each node runs its own copy of the library at its private memory space and processor. There is no shared memory system is involved in the system, this way the failure of a node does not disturb the operation of the rest of the system. With the absence of shared memory, each node communicates via passing messages to each other.

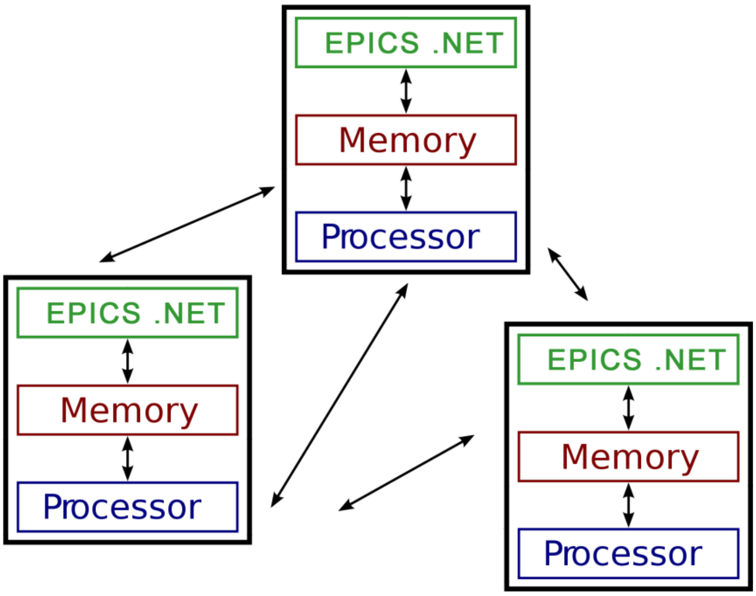


Figure 3-10 Distributed computing model with EPICS .NET Library

EPICS .NET Library fall in the client-server model of distributed computing architectures. Each input/output controller node is turned separately into clients and servers so each node is capable of requesting or sending information. Each system resource can be published on the EPICS network (which is a TCP/IP connection for node-to-node communications and UDP connection for beaconing and broadcasting). What is described by a resource may be the CPU usage data or the memory usage statistics or a signal flowing through an analog/digital converter attached to the system. Since the library uses publish/subscribe messaging paradigm for passing messages on the network any input or output signal collected or created by the input/output controllers can be published on the network with a unique identifier. Consequently, any node wishing to monitor a data for changes just needs to be aware of that resource's unique identifier. This removes the need from the accessors to know the IP address of the target node or the GUID or the resource to be accessed. Therefore, within the EPICS network, each resource is a freely floating entity where each node may monitor for changes or use the resource's data for internal calculations. Once the network is flooded with freely floating resources, it closely resembles a database with freely accessible records. Due to this, EPICS network is sometimes called a real-time distributed database. To take things a step further, any resource published on the network is also called an EPICS record! To access records on the network, channels are used (which are a part of the channel access protocol). Extensive information on the mechanics of resource publishing through the input/output nodes is given in the flowing sections.

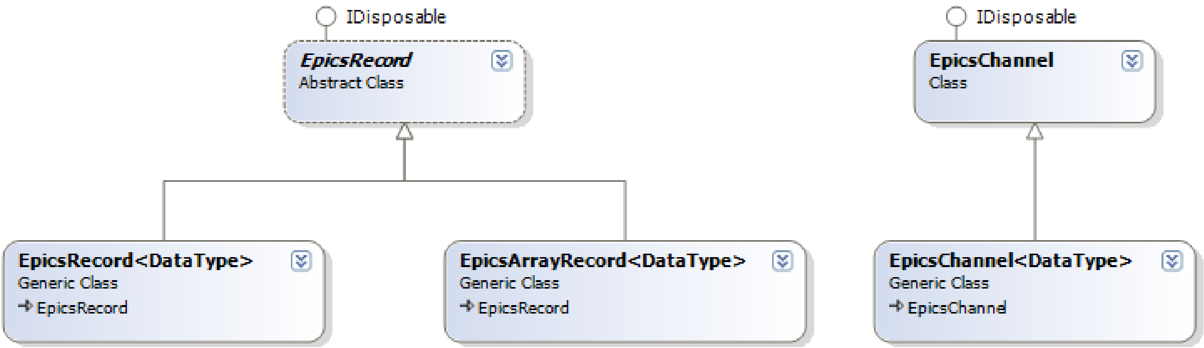


Figure 3-11 EpicsChannel type which is used in accessing EpicsRecord types

The class diagram above depicts the EpicsRecord and EpicsChannel types and their inheritance hierarchy. EPICS channels provide the node-to-node communication between the server (the node that is publishing a resource) and the client (a node that is requesting a specific resource). Thinking about the fact that each input/output controller node is a server as well as a client, lots and lots of channels are creating in each node. To remove some of the overhead of creating and destroying channels, each channel is recycled and reused for queued operations. Once created, a node can use a channel to access any resource published at the network. A very basic example to usage of a channel is:

```
var localClient = new EpicsClient();
var memoryUsageResource = localClient.CreateChannel<int>("MyIOCNode:MemoryUsage");
Console.WriteLine(memoryUsageResource.Get());
```

Here an EpicsClient type is used to create an EpicsChannel, which then accesses a remote resource (memory usage of the remote computer in this case) and displays it at the system console. As it is obvious, there is no information present in the code either about the IP address of the remote node or the GUID of the system resource that is being accessed. Knowledge about the unique EPICS record name, which acts as a unique identifier, and the node name is enough. Publishing a specific resource at the network is also as simple as accessing it.

```
var myIOCNode = new EpicsServer();
var memoryUsage = myIOCNode.GetEpicsRecord<int>("MyIOCNode:MemoryUsage");
memoryUsage.VAL = System.Diagnostics.PerformanceCounter("Memory", "Available MBytes");
```

Here the control node is named as “MyIOCNode” and an EPICS server is created on the node and a resource” MemoryUsage” is created and published on the EPICS network. The memory usage statistic is fetched easily via .NET Framework’s System.Diagnostics class. Based on this very simple client/server architecture, a very complex distributed system can be created with minimal effort. As it is apparent, all of the low-level details on the ongoing message passing processes are hidden from the programmer. All the complex details of network communications are handled by codecs which are the deciphers of the channel access protocols which is how EPICS based applications understand requests from each other.

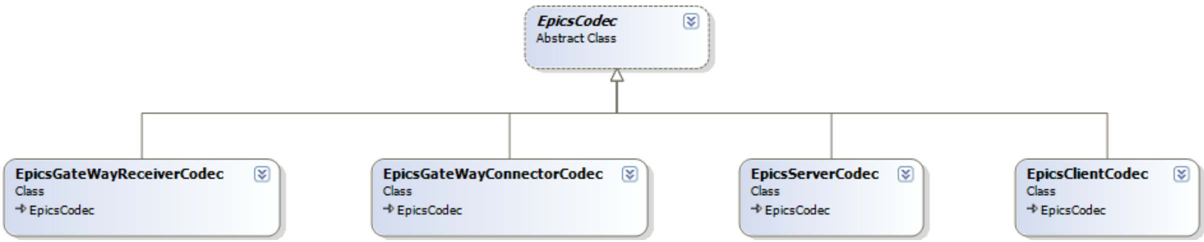


Figure 3-12 EpicsCodec type which is the basis of all channel access protocol communications

The EpicsCode class makes connection two types of network connection protocols that are available within the EPICS network. First is the UDP connection, which provides very low latency communications. This sort of a connection is fast is only suitable for small data packets because packet losses are unavoidable with UDP which sacrifices reliability for performance. UDP connections are used for beaconing, which deals witch nodes letting each other know of their existences and broadcasting, which deals with nodes lettings each other know of the resources that they publish at the network. Once each node is aware of each other, they may subscribe to each other’s resources using an EpicsChannel type as a brief example was given before. Once a data channel is established between two nodes, rest of the communications occurs using the TCP connection. TCP connections are suitable for long running reliable communications where there is little tolerance to packet losses and large chunks of data are passed around.

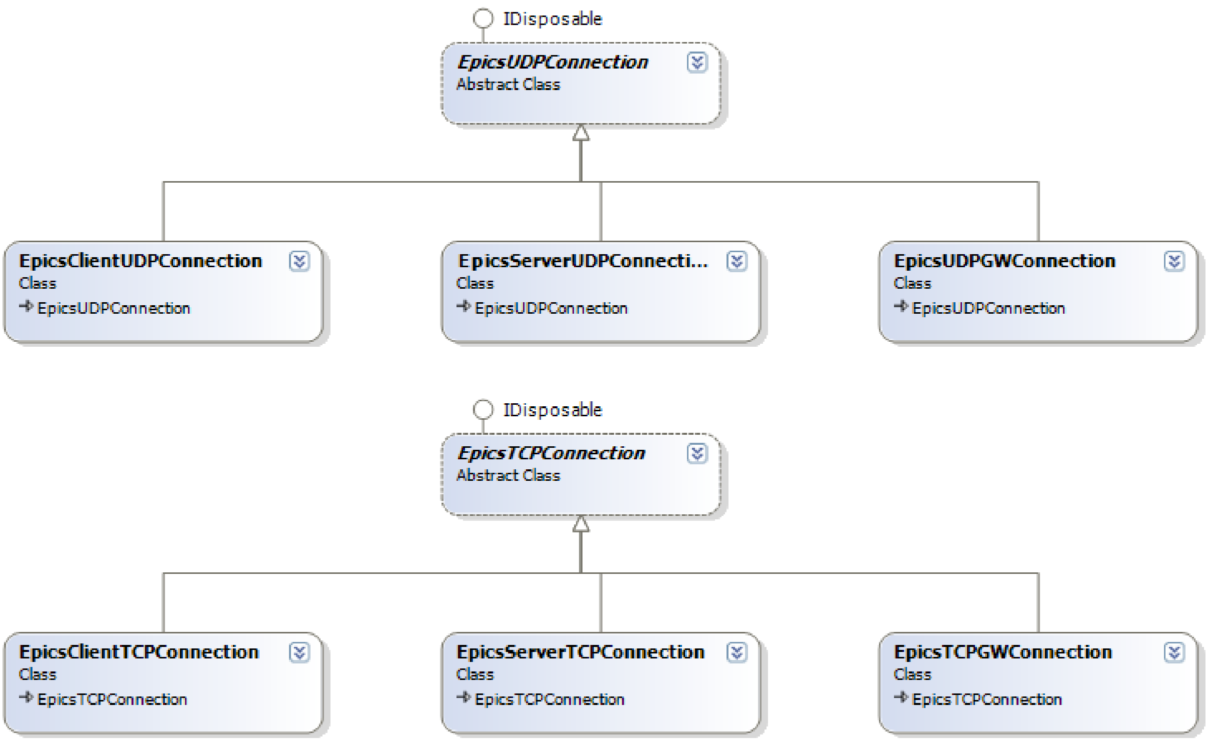


Figure 3-13 EpicsUDPConnection and EpicsTCPConnection types

Although EPICS .NET Library's implementation is based solely on these two connection types, extendable nature of the library allows for any desired connection type to be added according to user needs like ModbusTCP or connections based on raw Ethernet frames. This is possible through the extension methods, which are a part of .NET Framework 3.5 and above (again the benefit of being 100% native .NET assembly).

On the network, sharing primitive types of data like numbers, decimal numbers, strings, and other types is quite simple. While it is possible to retrieve any desired resource that is published on the network, some types of resources carry specific type of data that may need special treatment. For instance, publishing a whole image that is coming from a frame grabber that is attached to a nearby camera is easily possible but requires the use of extended EPICS types. Again, the extendable nature of the library allows the end users to add any specific type, as they need.

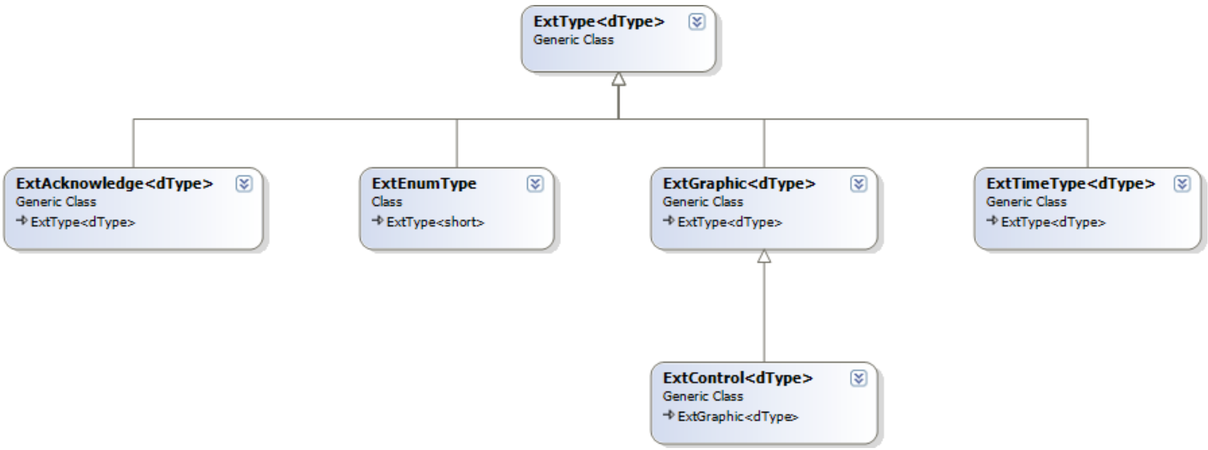


Figure 3-14 Default extended types that come with the library

Speaking of the basic building blocks of EPICS .NET Library, documentation of more advanced classes that are available for developers use can be used in the MSDN style code documentation that is accompanying the source code. Below is an overview of the rest of the classes that ship with the current version of the library.

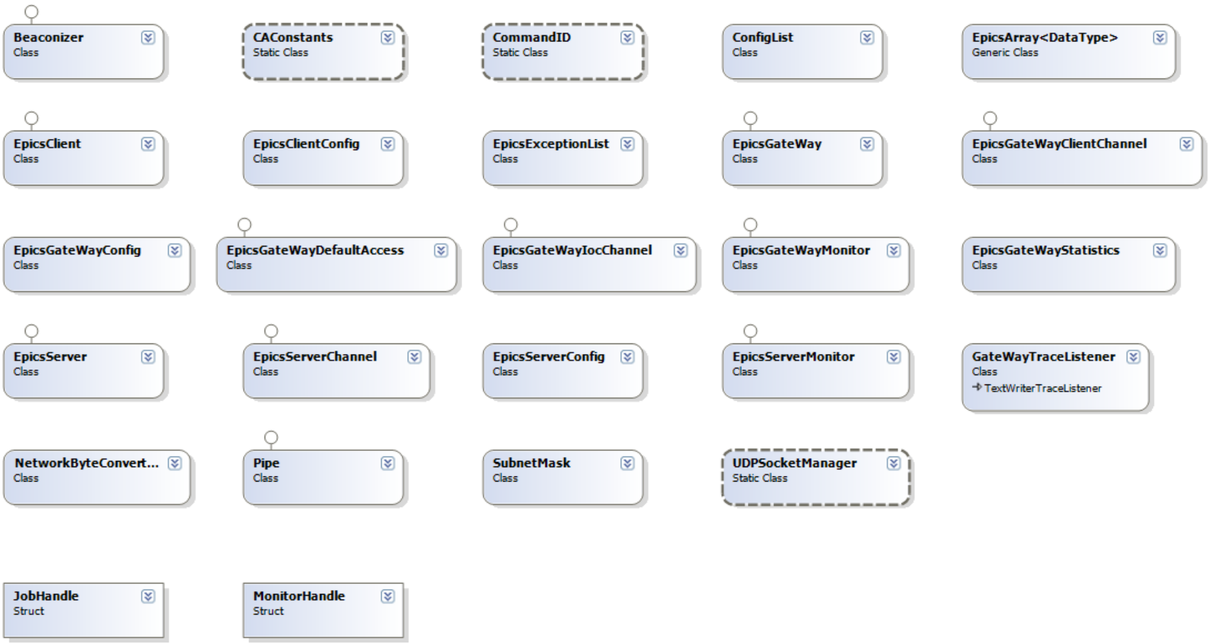


Figure 3-15 Complete list of types that make up the library

Whenever necessary, the documentation of the source code is filled with sample usages and other many educational system scenarios, which makes reading the source code a lot easier and meaningful.

3.2.2 Control Nodes and the Workflow

The control nodes make up the core of the control network but they are only functional when they start executing their internal control logic (the control workflow). In a typical configuration, control nodes like industrial PCs or PLCs have data acquisition hardware that collects analog or digital signals from the machines. This DAQ hardware then converts these signals into digital data, which is processed by the control node's processor, which is finally published on the EPICS network using the control library. Any other computer requiring to access this data will simply subscribe to it again using the control library.

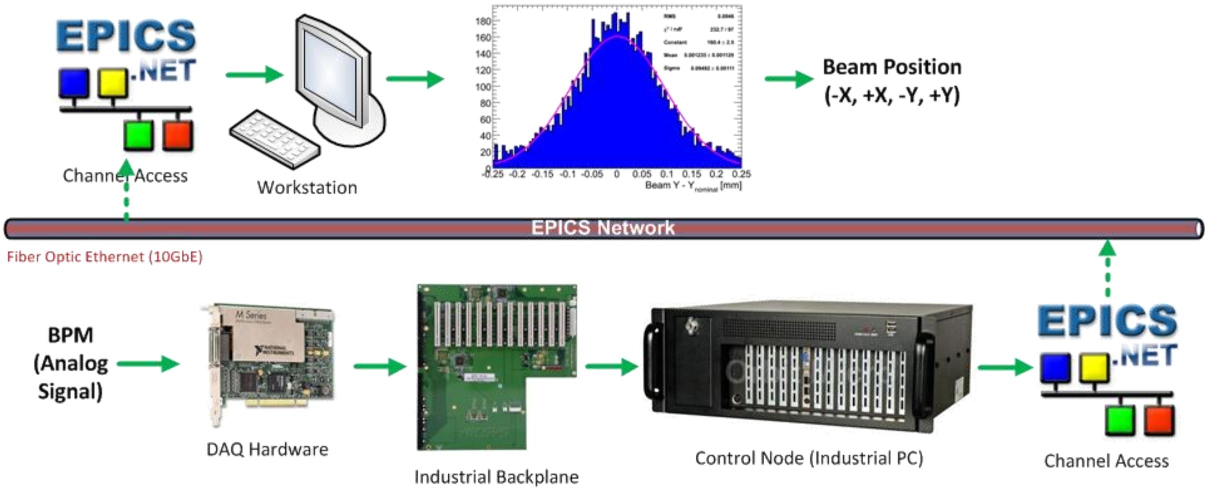


Figure 3-16 Control nodes and data flow

The above figure summarizes the control node configuration that is used with the control system. Once desired DAQ hardware is put in the node chassis, the EPICS .NET Library takes care of publishing the required signals acquired from the DAQ hardware on the control network. Then the node-to-node or node-to-workstation data flow is regulated with the previously mentioned publish/subscribe messaging paradigm. On the other hand, internal control logic is controlled mainly with Windows Workflow Foundation [35] which is again a part of .NET Framework. Workflow Foundation controls the internal data flow within the nodes at the higher logical level. Low-level details are all taken care of in the code to achieve maximum performance with the overall workflow.

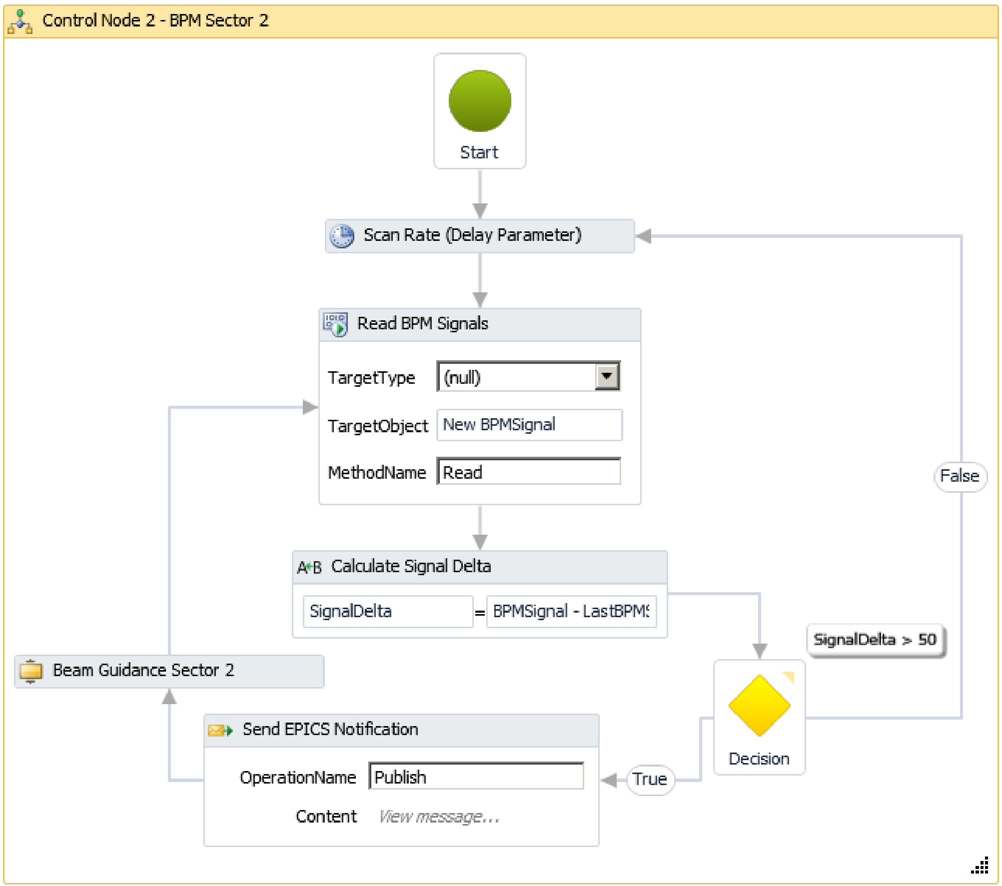


Figure 3-17 BPM logic at the control node 2

The preceding workflow diagram clearly describes the series of events taking place internally at the control node that is depicted as "Control Node (Industrial PC)" at Figure 3-16. Note that this is not just a pretty illustration; this is the actual code that controls the workflow! With the new features of Visual Studio 2010, creating the workflows completely in the designer is now possible, without manually writing XAML. The above diagram illustrates how the scan rate for the DAQ hardware is set, how BPM signal is evaluated and published on the EPICS network, and finally how the beam guidance code is notified of the current situation. The beam guidance is again automatized by another workflow, all internally at the control node.

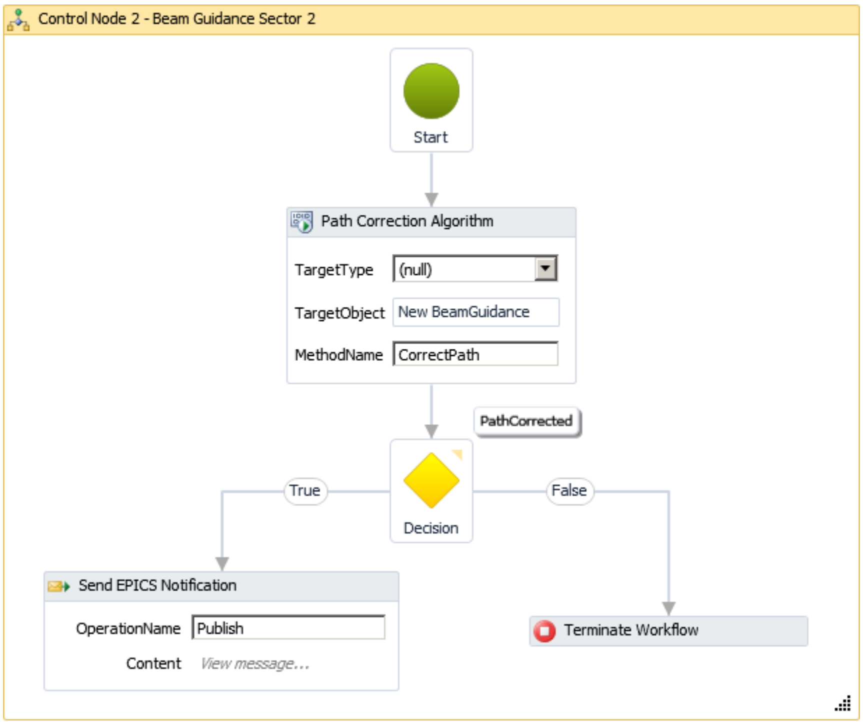


Figure 3-18 Beam guidance logic at the control node 2

The preceding workflow diagram controls the beam guidance in the least complicated way. This is possible via calling the pre-written path correction algorithm to help. This workflow corrects the beam path and places the beam back at the center of the drift tube. This automatized beam correction is done as fast as 10.000 times per second (previously configured by the scan rate parameter of BPM workflow). This workflow and the path correction algorithm are reusable so implementing BPM control and beam guidance in other nodes and sectors is a breeze. While the above workflow logic is truly autonomous, meaning that beam guidance is taken care of by the control nodes all automatically, the operator can get involved in the process using the HMI, which is introduced at the following section.

3.2.3 Human Machine Interface (HMI)

Almost all of the human-machine interactions occur at software level thus providing the users with a solid interface is essential. The main goal of this user interface is to provide the means for effective operation and control of the machines and gather the feedback signals on a single screen, which aids the operator in making operational decisions. Although almost all of the control procedure is automated, variations in the scientific experiment scenarios compel constant supervision of the diagnostic monitors where the information presented must be crystal-clear. Again, consistency and usability is up to par with the rest of the system so at every step of the control, the users will be met with a familiar interface. It is important to note that just like any other control software developed for the IR-FEL facility, this HMI software makes extensive use of our EPICS .NET Library to communicate with the rest of the distributed control system.

Figure 3-19 Initial screen of the main HMI software: “The Virtual Accelerator”

The name “Virtual Accelerator” is given to the application that is controlling the beam line, cryogenics, and vacuum systems. This is due to the fact that the application is primarily designed to fully simulate the accelerator as a whole, while providing human-machine interface to the control facilities. As of writing this report, not all the systems are in place so simulating the accelerator components is crucial in development of software systems. Considering the amount of effort put forth for developing this sort of a large-scale simulator, it was also significant to publish this application as an open-source software. Doing so, any interested parties (i.e. scientists, developers) can contribute to this great open-source project, while using it to train their own operators or jump start creating their own control systems. While interested groups can benefit from this project, they will also be providing invaluable feedback, which will accelerate the debugging and development of this great software. Just like any open-source project, this software will evolve reasonably fast, so it is important to provide a consistent interface with every new version of the software so that 3rd party users will not be taken by surprise. The persistent layout that will be used throughout the software is as below.

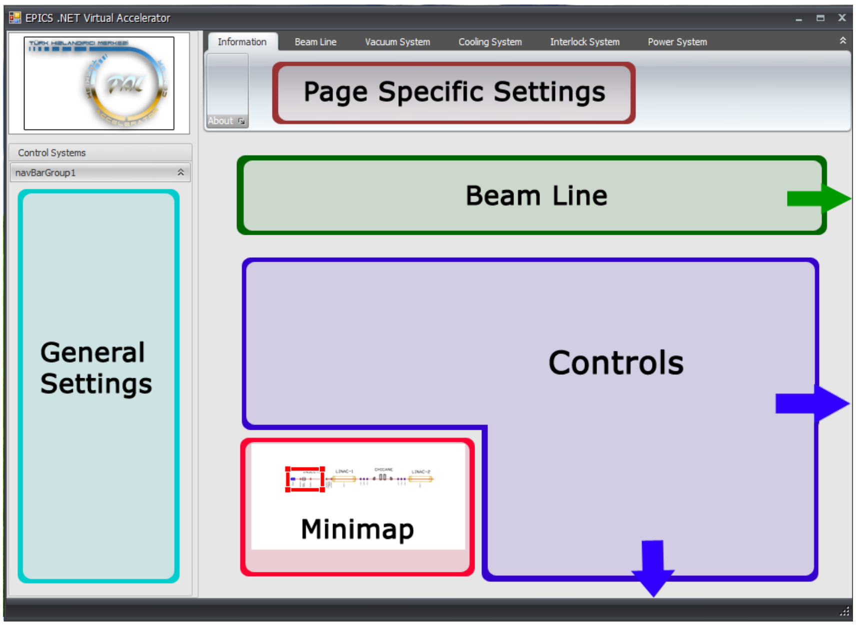


Figure 3-20 Main layout of control and simulation software

Each tab on the top of the screen is used to change the view to other components of the control system and the accompanying ribbon bar is populated with page specific controls. The left hand side is reserved for general settings, which are always visible regardless of the tab being displayed. This is because controls such as emergency switches should always be visible on the page as the rule of thumb. Just below the ribbon bar is the overview of what is being controlled, like the beam line, and rest of the screen is filled with control panels. With respect to these guidelines, below is an application of this design, which concerns with the control of the beam line, with minor modifications like the location of the mini-map and the application logo.

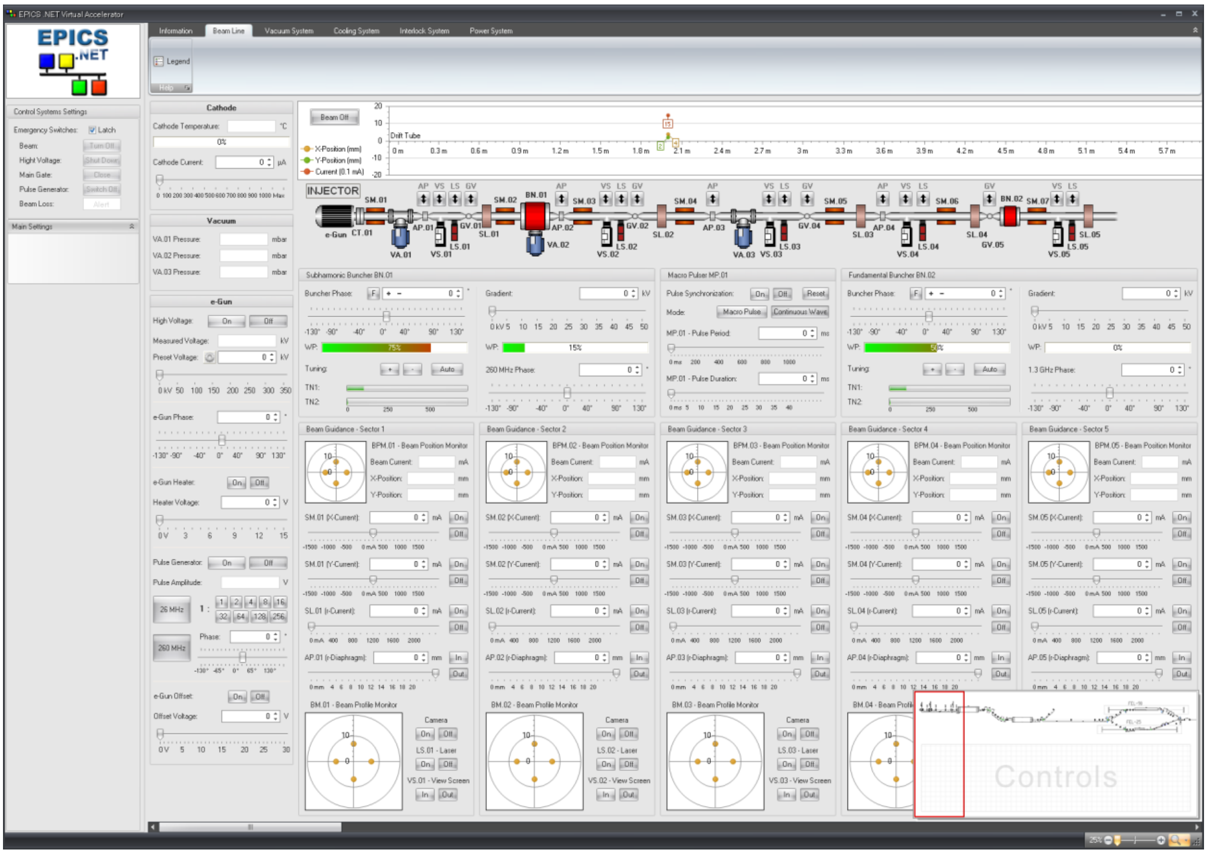


Figure 3-21 Main screen of the beam line control tab visualizing the injector portion of the accelerator

One of the reasons for developing an EPICS library natively in C# programming language with Microsoft .NET Framework 4.0 [23] was to be able to benefit from the unmatched UI capabilities of Microsoft Visual Studio 2010 [22] in combination with Developer Express Windows Forms UI components [24]. As a side note, Windows Forms was chosen as the UI base rather than the brand new Windows Presentation Foundation 4.0, due to its maturity and suitability for industrial grade applications with its proven stability. Developer Express UI components were used to enhance the user experience, regarding the fact that they are considered to be the best UI components available for WinForms, for their stability and functionality. Continuing the investigation of the HMI software, below is the line chart that visualizes the process inside of the drift-tube.

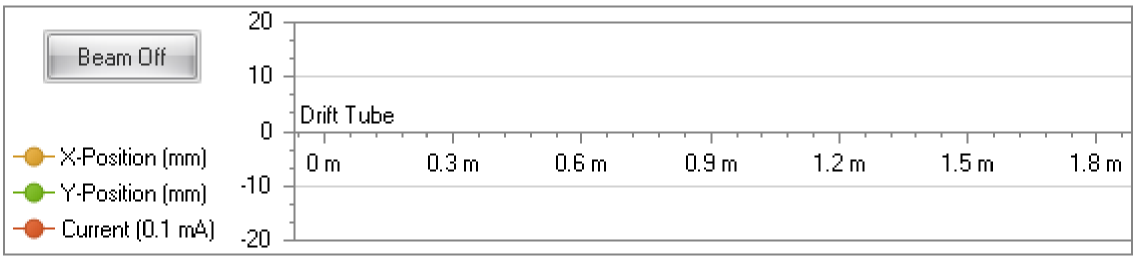


Figure 3-22 Line chart visualizing the drift tube in terms of beam location and current

The provided line chart, which extends all through the beam line, gives a quick overview of the beam location in the drift tube, while providing a real-time measurement of the beam current at the given locations on the beam line. This line chart is populated with the data retrieved from the beam position monitors through the data acquisition hardware. Beam position on two axes (-X, +X and -Y, +Y) and the beam current are measured at each sector and the exact position of the measurement is specified by the y-axis of the line chart in terms of meters of distance from the beam source (e-gun). The complete beam line is visualized by the graphics found just below the chart.

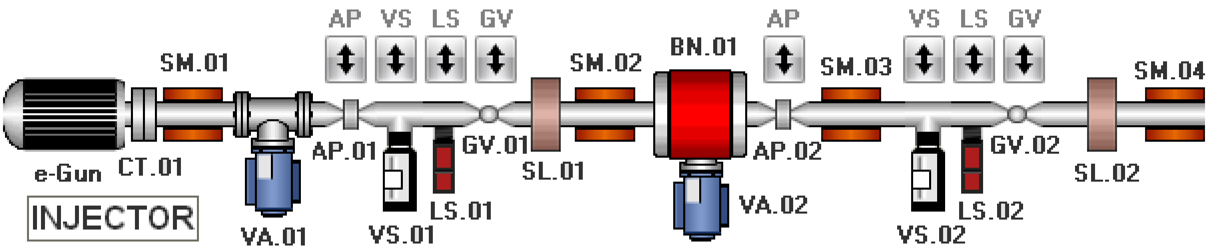


Figure 3-23 Visualization of beam line elements

Each beam line component is visualized by its unique set of graphics describing the status of that component/equipment at a given time. Some of the beam line elements require manual activation to gain functionality. To enable quick access to the provided functionalities, small buttons are places on top of some components so the operator can quickly close a gate valve, set the aperture or drive the beam profile monitor mirrors into the beam line. Once activated, separate panels display detailed information and control for the selected element as seen from the following screenshots.

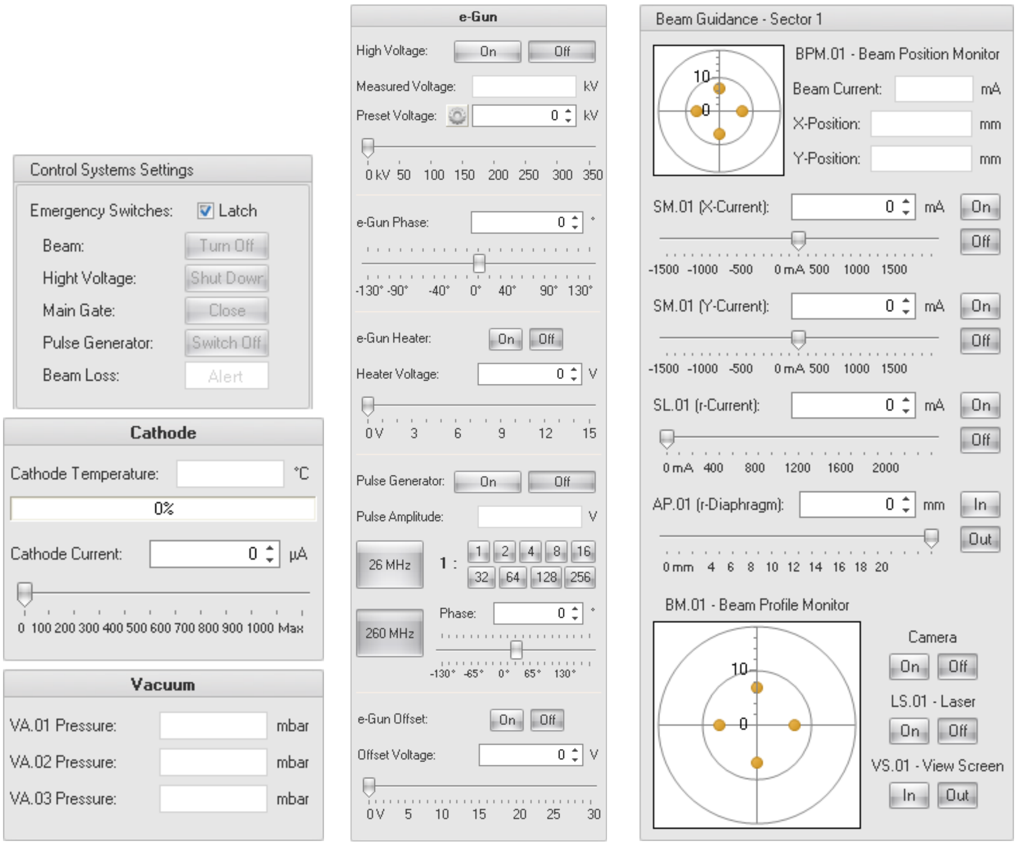


Figure 3-24 Various control panels with detailed data from beam line elements

Each control panel exhibit real-time data collected throughout the beam line and feature specific functionalities. Some control panels, like the "Beam Guidance" panel, are specific to particular sections of the beam line. These sections are generally divided based on their location to the nearest solenoid. Background watermarks indicating the exact location of the each sector can be enabled from the settings panel but are disabled by default. At some sectors, some of the beam guidance elements may have been omitted on purpose so they are disabled at their respective panels. Some control panels, like the buncher controls, were positioned closer to the beam line chart to highlight their importance because they affect the operation of the whole beam line and not just some local equipment.

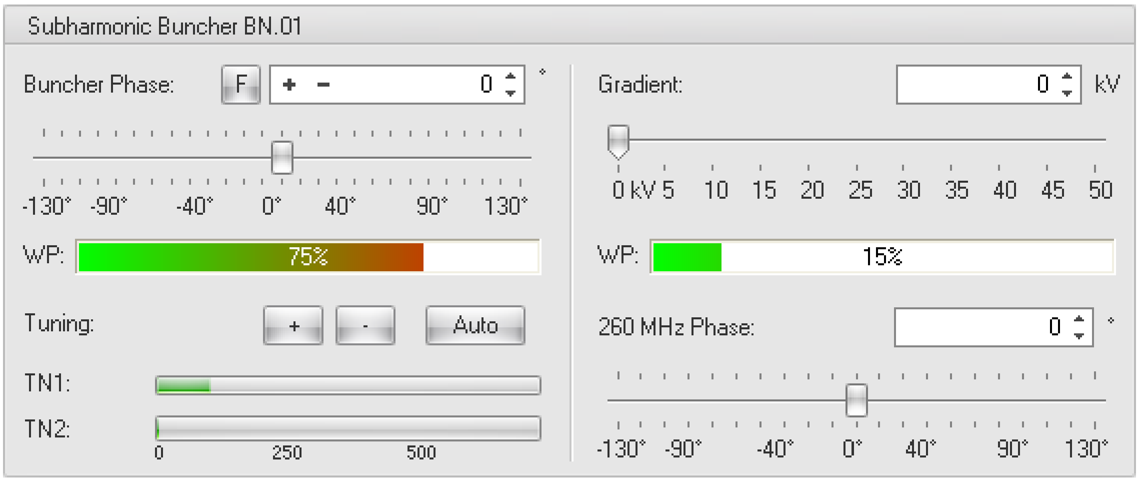


Figure 3-25 Buncher controls panel

As it is obvious from the above figure, each control panel is quite self-explanatory to anyone with a science or engineering background. Even though this, help is available for each panel with a single hit on “F1” key or via the ribbon bar which is available on top of each control page. In addition to this, the ribbon status bar that is at the bottom of the screen is equipped with zoom control and a mini-map.

Figure 3-26 The ribbon status bar of the main form with the mini-map enabled.

The zoom control on the status bar adjusts the zoom level on the main form, thus reveling more detailed controls or hiding the details to give an overview of the beam line. The mini-map provides an easy method for scrolling to the desired portion of the beam line without even dealing with the scroll bars. The red borders on the mini-map indicate the currently selected area of control on the beam line and it can be dragged through the map using the mouse. It dynamically adapts to the screen size and resizes itself to accurately reflect the new width and height of the application window. Being obligated to scroll through pages and pages of controls is one of the most common difficulties that challenge the operators every day so; this feature was implemented to overcome this specific hurdle, making the lives of end users less complicated.

3.2.4 Process Visualization

Process visualization is done along the beam line, for cryogenics, for vacuum system, and for the interlock mechanism. A sample of beam line visualization can be seen at Figure 3-23. Each visualization is accompanied with process animations informing the operator about the current situation of the ongoing processes.

3.2.5 Data Management and Storage

During the experiments, data acquisition hardware produces massive amounts of data. To reproduce and analyze the experiment scenarios, selected portions of these data must be stored. In addition, EPICS .NET Library only publishes real-time values on the network thus when operator requests detailed graphs on histograms of this data, it needs to be retrieved from an immediate storage. EPICS Archiver takes care of storing the values for future graphing, capturing the selected resources published on the control network. All these data captured from the network by the EPICS Archiver is stored with Microsoft SQL Server 2008 R2 Enterprise on the reference server of which the specs are given at Table 3-4.

3.3 IT Systems

Not all the data produced during the experiments may be for internal use. These data may be shared over the internet with other research labs for collaboration. In addition, staff of the facility needs collaboration over the internet as not everyone is present at the laboratory at all times. Due to this, a separate IT server is deployed to serve the needs that are not directly related to the control system. This way, control system needs no internet connection for sharing internal data. All the security concerns are dealt by the IT server and related software on a separate layer.

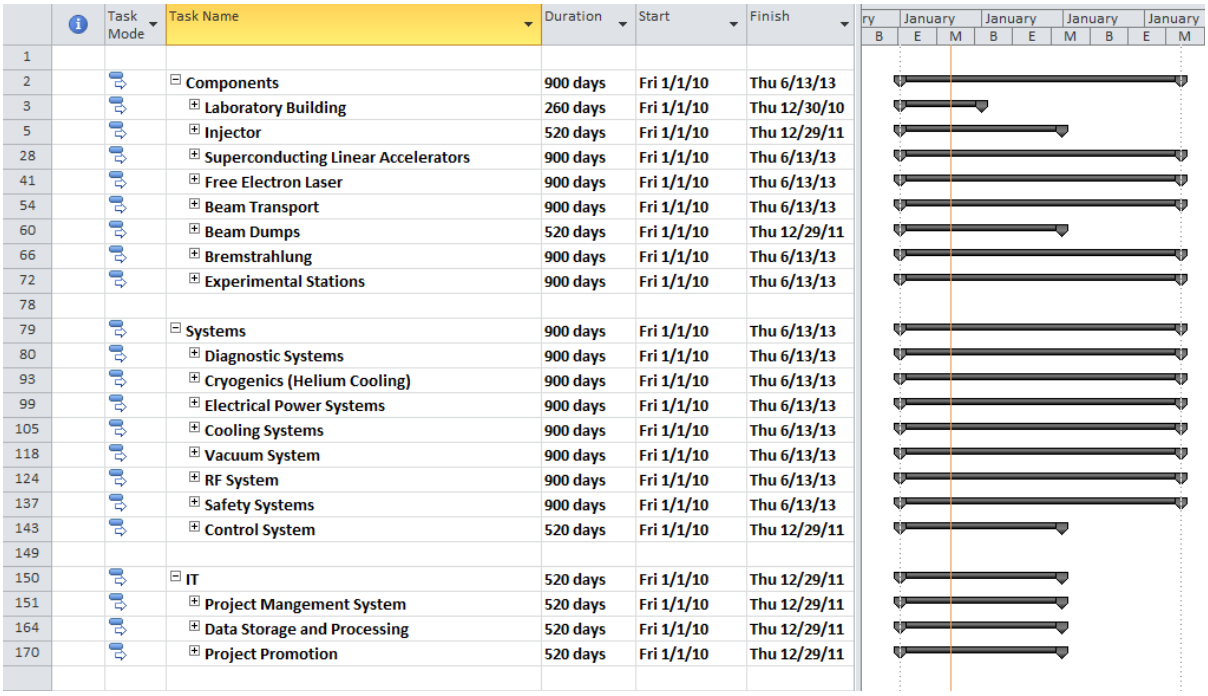
A notable feature of IT systems is that nearly all of the server software used in the IT servers are either professional, developer or enterprise editions. Normally this would be a very costly solution to a rather simple problem for a corporation. However, since this facility is tied to Ankara University and its Institute of Accelerator Technologies, nearly all of the Microsoft software used in the systems are acquired via Microsoft Academic Alliance program with developer status. This provides the opportunity of using the listed Microsoft software [36] at almost no cost. Apart from their functionality, this is one of the reasons why Microsoft server and client software are used throughout the facility, almost exclusively. List of 3rd part software used and their uses are listed on Appendix A.

3.3.1 IT Assets Management

As usual, Microsoft Visio 2010 takes care of IT assets management with the assets management diagram templates. Selected diagrams will be added here as all the sections are detailed.

3.3.2 Project Management System

As a part of the IT systems, project management is done using Microsoft Project Server 2010. Although not yet fully implemented, project tasks and resources are easily matched to calculate the projected timeline and budget for the project. Below is the screenshot from the main screen of the project management panel.



	Task Mode	Task Name	Duration	Start	Finish
1					
2		Components	900 days	Fri 1/1/10	Thu 6/13/13
3		Laboratory Building	260 days	Fri 1/1/10	Thu 12/30/10
5		Injector	520 days	Fri 1/1/10	Thu 12/29/11
28		Superconducting Linear Accelerators	900 days	Fri 1/1/10	Thu 6/13/13
41		Free Electron Laser	900 days	Fri 1/1/10	Thu 6/13/13
54		Beam Transport	900 days	Fri 1/1/10	Thu 6/13/13
60		Beam Dumps	520 days	Fri 1/1/10	Thu 12/29/11
66		Bremstrahlung	900 days	Fri 1/1/10	Thu 6/13/13
72		Experimental Stations	900 days	Fri 1/1/10	Thu 6/13/13
78					
79		Systems	900 days	Fri 1/1/10	Thu 6/13/13
80		Diagnostic Systems	900 days	Fri 1/1/10	Thu 6/13/13
93		Cryogenics (Helium Cooling)	900 days	Fri 1/1/10	Thu 6/13/13
99		Electrical Power Systems	900 days	Fri 1/1/10	Thu 6/13/13
105		Cooling Systems	900 days	Fri 1/1/10	Thu 6/13/13
118		Vacuum System	900 days	Fri 1/1/10	Thu 6/13/13
124		RF System	900 days	Fri 1/1/10	Thu 6/13/13
137		Safety Systems	900 days	Fri 1/1/10	Thu 6/13/13
143		Control System	520 days	Fri 1/1/10	Thu 12/29/11
149					
150		IT	520 days	Fri 1/1/10	Thu 12/29/11
151		Project Mangement System	520 days	Fri 1/1/10	Thu 12/29/11
164		Data Storage and Processing	520 days	Fri 1/1/10	Thu 12/29/11
170		Project Promotion	520 days	Fri 1/1/10	Thu 12/29/11

Figure 3-27 Project management plan

Each bullet shown on the screenshot can be extended with a simple click to reveal the details about the subsections of the project.

3.3.3 Collaboration Web Site

Working with the rest of the IT systems, the collaboration web site makes sharing information between the project staff a breeze. The provided screenshot, which is from the homepage of the web site features project logbook, news, notifications, reports, and events makes up only a small portion of information sharing capabilities of the web site.

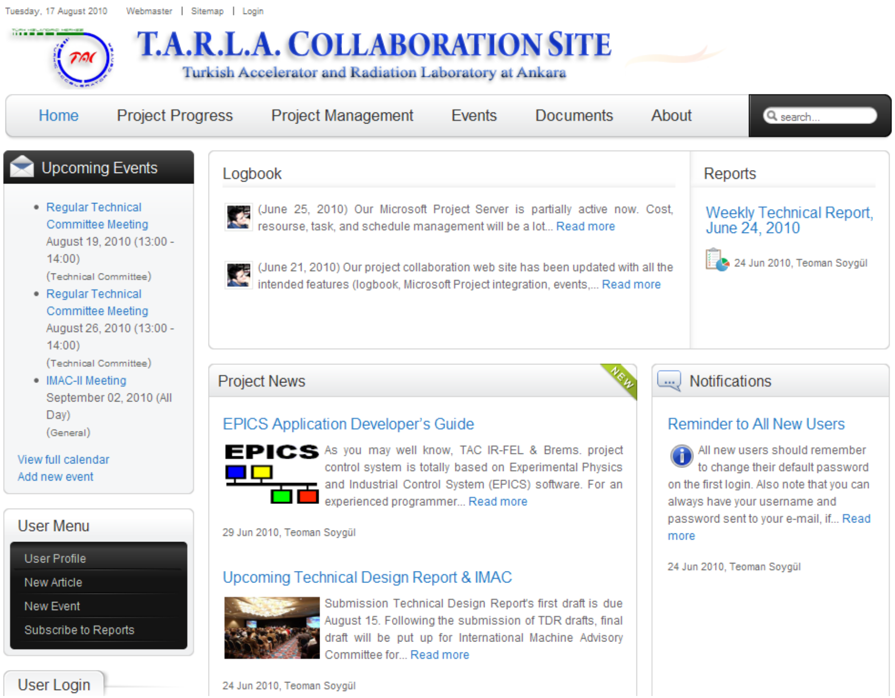


Figure 3-28 Collaboration web site home page

Almost all of the information presented at the web site is also sent as weekly, monthly, or quarterly reports and newsletters where project members can subscribe in order to be notified of the latest project news and events.

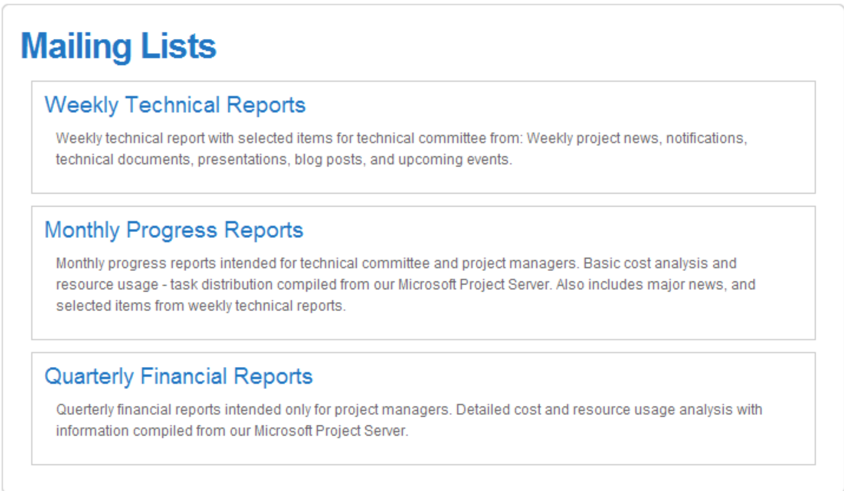


Figure 3-29 Newsletter and reporting system of the web site

One of many notable features of the collaboration web site is its in page editing capability. This feature provide the users with ability to change the web site content without requiring any knowledge of HTML or other web authoring technologies. This way, any nontechnical personnel will be subject to a very familiar Word like interface when editing content.

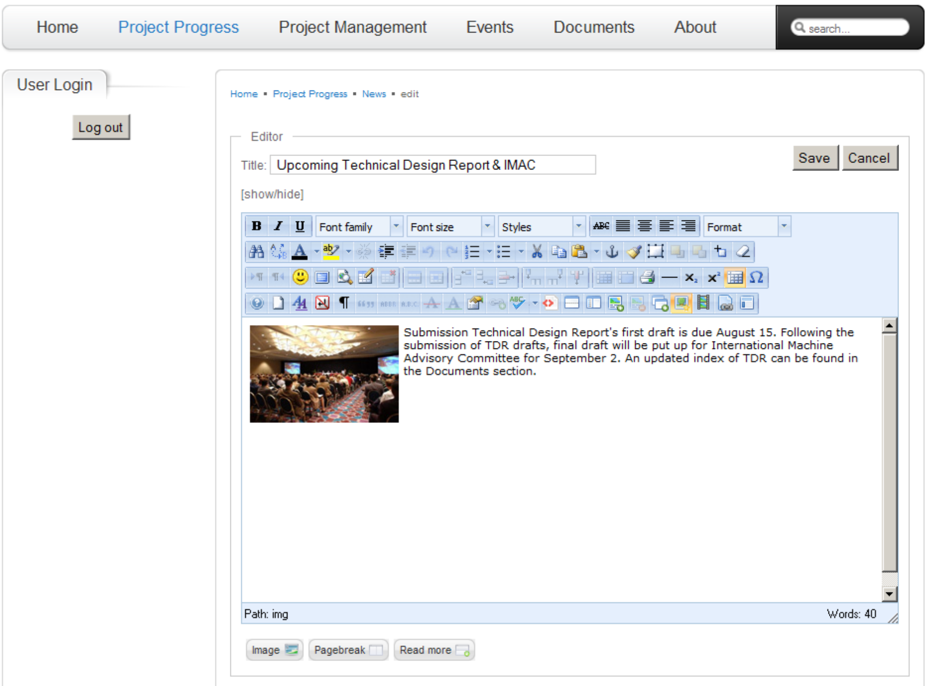


Figure 3-30 Web site's in page editing capability

4 System Implementation

The system implementations that are done according to the project plan are listed below._The roles and locations of the currently implemented configurations are as follows.

4.1 Configurations

All the system configurations are stored on Microsoft System Center Configuration Manager [37]. Additional information is not presented here since access to system configurations is classified. Additional details can be retrieved via logging into the system, once the IT servers are fully deployed.

4.2 Floor Plan

Control room is design for a typical 5-workstation configuration. One master terminal has the ultimate power to override any control while the workstations are for general-purpose control and data processing. The control room is equipped with two 19-inch racks of 20U size. One is for the servers while the other is for network equipment and for future growth.

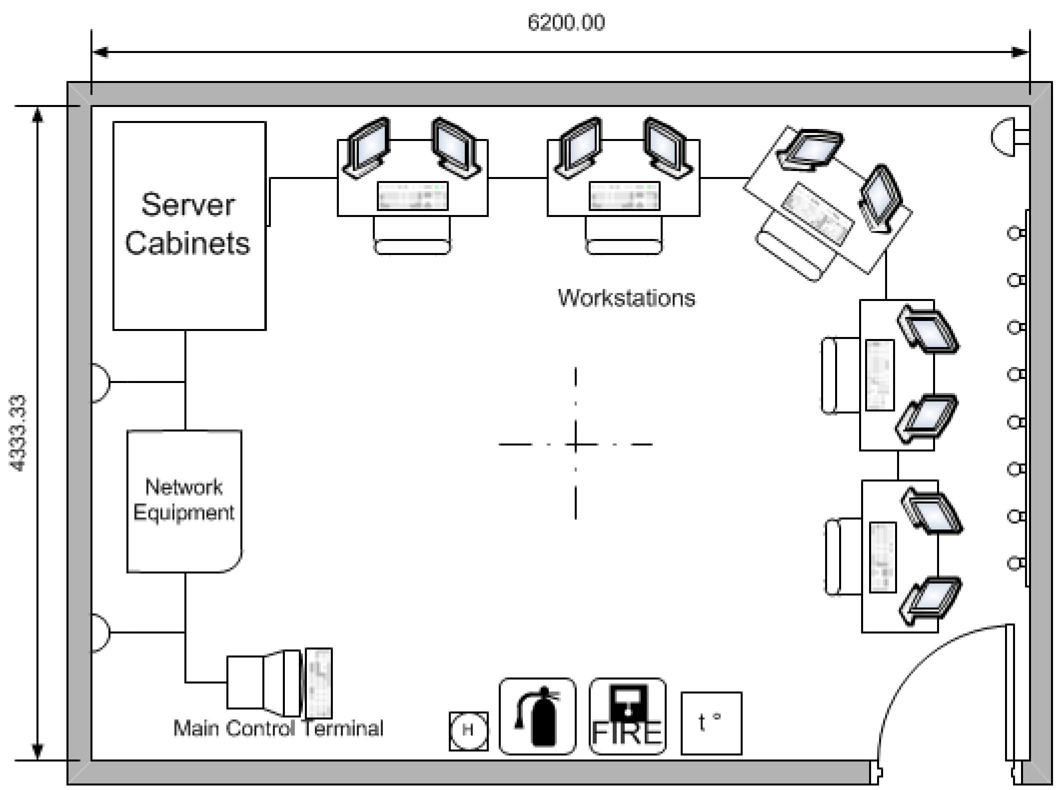


Figure 4-1 Floor plan for the control room

When completed, control room will be equipped with workstations featuring a comfortable 2-monitor setup for each operator or scientist. Below is the illustration of the intended setup.

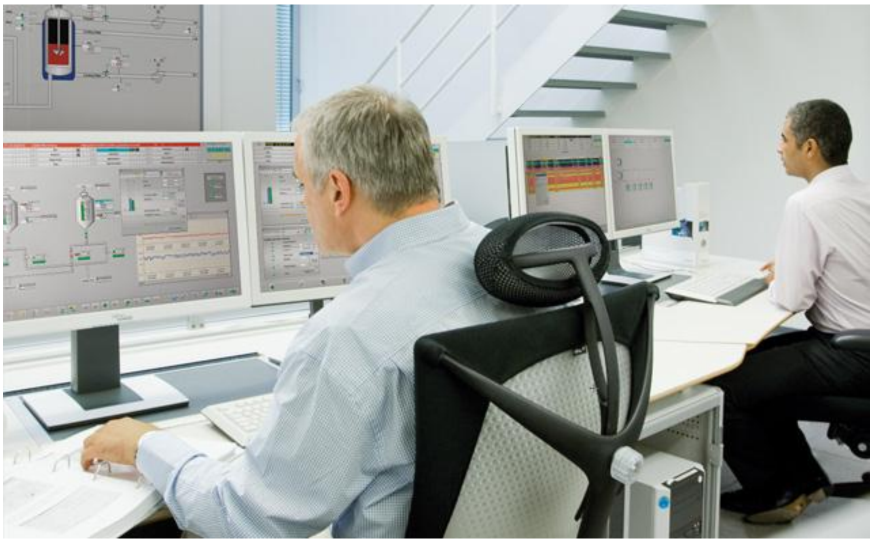


Figure 4-2 Intended control room configuration with two monitor workstation setups (courtesy of Siemens [38])

Fitted with comfortable chairs, control room should provide a friendly environment, especially for night shift operators.

4.3 Grid Control

The electron gun grid is a general part of the system responsible for pulse generation and the filament, which produces the electron cloud for initial acceleration. The pulses are generated with a pulse generator modified for 60V pulse amplitudes. A trigger generator is used to trigger the pulse generator to initiate a continuous wave or a pulsed output. These two equipment are controlled with a PLC based IOC which are all fitted inside a structure called the "hamburger". The naming is due to the structure's shape and it is responsible for providing HV (350kV) isolation from the environment. This insulation dictates that no conducting cable should intersect the insulation barriers thus all outside communication is done via "Black Box Media Converter Switch 10/100Mbp Copper to 100Mbps Fiber" [39], which is connected to the PLC unit. As there can be no power cable penetrating the insulation barrier, all the required power is transmitted with a 230 VAC to 230 VAC transformer with an isolation voltage of 350kV. An illustration of this structure is given below.

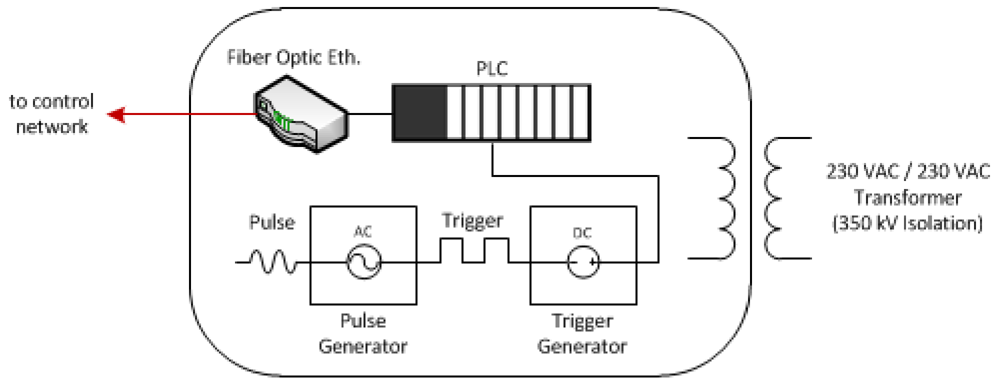


Figure 4-3 Hamburger structure for grid control

Apart from pulse generation, grid control requires feeding the filament with 12V Max voltage and the gun gate with a 10V Max bias voltage. Filament voltage adjusts the density of the electron cloud thus generating the initial beam current. On the other hand, gun gate regulates this beam current through the supplied bias voltage. This bias voltage is supplied with a redundant power supply to make sure that beam current is always under control via the bias voltage.

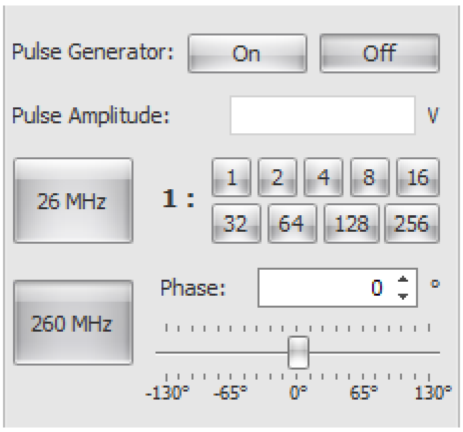


Figure 4-4 Grid control panel of the HMI

The above image belongs to the e-gun control panel of the HMI software. Again, the pulse generation parameters are easily modified using this custom interface.

4.4 Beam line Control

Beam line is equipped with devices like solenoid for beam focusing, steering magnets for beam guidance, gate valves for sector division, and many more equipment used in accelerating and delivering the electron beams to the free electron laser paths. The low-level control of all these equipment is done by the control nodes using the EPICS .NET Library as it was described before. As it is illustrated in Figure 3-3, the control software interacts with these pieces of equipment through the input output controls that are located on selected spots in the facility. Finally, the operator interacts with the control software using the HMI software that is described in section 3.2.3 on page 36.

4.5 Cryogenics Control

Cryogenics control is yet to be implemented so this section will be filled, as the implantation is complete.

4.6 UHV Control

Vacuum control is yet to be implemented so this section will be filled, as the implantation is complete.

4.7 Control of Diagnostic Systems

Unlike usual industrial plant control systems, diagnostic equipment are not considered a natural part of the control system in accelerator facilities due to their extremely complex nature. In addition to this, and unlike the LLRF system, diagnostic systems are not self-contained. Rather, they are designed to warn the supervisory systems in case of danger. Due to this, constant interaction between diagnostic and control systems is required. Just like interactions with all other machines in the facility, these interactions are also carried out with the control nodes, namely the input/output controllers. For instance, a beam position monitor probe is a delicate piece of equipment that is designed to measure the beam position within the drift-tube with a very high level of accuracy (in the order of micrometers). The signals coming through these probes are carried with RF cables to the nearest IOC that has an analog digital converter card with 16 bits of measurement accuracy and 250 KSPS measurement rate or higher. Once the analog signal is converted to digital by the ADC, the EPICS .NET Library publishes the real-time data on the control network. Then this data is used as necessary to guide the beam in its correct path within the drift-tube. Figure 3-24 right hand side column illustrates how this data is displayed on the HMI beam guidance panel for operator supervision.

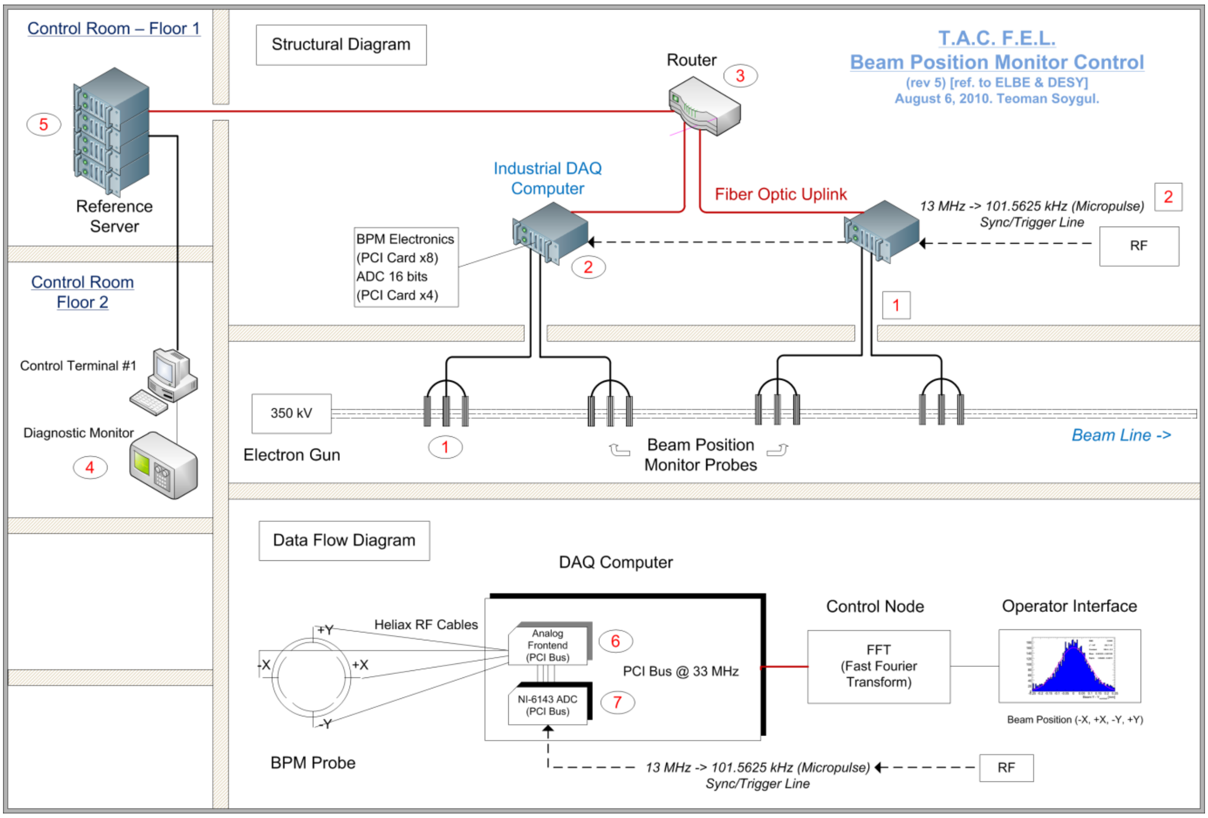


Figure 4-5 Beam position monitors data acquisition, control, and supervision

The data flow diagram at the bottom part of the above figure demonstrates an important detail. According to this diagram, the data flow starts with the BPM probe and is carried out right up to the HMI screen (the operator interface). The mechanics of this data flow is detailed on Figure 3-16. One thing to note here is that the conversion of analog data to digital, fast Fourier transform operation on this data is all carried out on the control node. This is because the unprocessed data may flood the control network so only after required transformation the data is released on the network.

5 Conclusion and Future Projections

Near future objective of the project is to extend the current implementation of the distributed control systems to include remaining portions of the facility. Even though it is relieving to have a complete design and a partial implementation on the injector section, which can be taken as a reference, some special parts of the system like the FEL lines is expected to prove challenging to realize. Even in the face of the aviating obstacles, the control system for the injector, linac-1 and linac-2, and the mid/far FEL paths is expected to be delivered in the original 12 months' time frame. The implementation of the control system for the bremsstrahlung room, machine and human protection, and interlock systems will be delayed until those sub-projects are complete.

6 References

[1] Anonymous. (2010, August) Wikipedia: Control Theory. [Online]. http://en.wikipedia.org/wiki/Control_theory

[2] Anonymous. (2010, August) Wikipedia: Use Case Diagrams. [Online]. http://en.wikipedia.org/wiki/Use_case_diagram

[3] Anonymous. (2010, August) Wikipedia: Google Platform. [Online]. http://en.wikipedia.org/wiki/Google_platform

[4] M. Takagi S. Motohashi, "Data Acquisition System of Beam Loss Monitors of the J-PARC Main Ring," in *Proceedings of the 4th Annual Meeting of Particle Accelerator Society of Japan*, Wako, 2008, pp. 1-3.

[5] K. Furukawa, T. Mimashi, T. T. Nakamura, J. Odagiri K. Mikawa, "Embedded EPICS Controller for KEKB Pulsed Quadrupole," in *Proceedings of EPAC08*, Italy, 2008, pp. 1-3.

[6] K. Furukawa, N. Kamikubota, H. Nakagawa A. Uchiyama, "Development Of Embedded EPICS on F3RP61-2L," in *PCaPAC08*, Ljubljana, Slovenia, 2008, pp. 145-147.

[7] Takuya Nakamura, "Upgrading the Control System of the Movable Masks for KEKB," in *Proceedings of PCaPAC08*, Slovenia, 2008, pp. 1-3.

[8] Beckhoff Automation. (2010, August) Beckhoff Embedded PC series CX1020, CX1030. [Online]. http://www.beckhoff.com/english.asp?embedded_pc/embedded_pc_series_cx1020_cx1030.htm?id=97337455619

[9] Advantech Co. (2010, August) 4U 20-slot Rackmount Chassis. [Online]. http://www.advantech.com.tw/products/4U-20-slot-Rackmount-Chassiswith-Multi-system-and-Front-accessibleRedundant-Power-Supply/mod_1-2JKJWD.aspx

[10] Trenton Technology Inc. (2010, August) Trenton 20-Slot PCI Express Backplane. [Online]. http://www.trentontechnology.com/backplanes/picmg-13-backplanes/pci-express-backplane-bpx6806

[11] Trenton Technology Inc. (2010, August) Trenton MCXT-E SHB. [Online]. http://www.trentontechnology.com/single-board-computers/picmg-13/mcxt-e6685

[12] Emulex Corp. (2010, August) Emulex OneConnect OCe10102-N. [Online]. http://www.emulex.com/products/oneconnect-ucnas/emulex-branded/oce10102-n/overview.html

[13] Hewlett-Packard. (2010, August) HP ProLiant DL385 G7 6172. [Online]. http://h10010.www1.hp.com/wwpc/us/en/sm/WF06b/15351-15351-3328412-241644-241475-4132832-4161661-4161662.html

[14] Hewlett-Packard. (2010, August) HP Sales Partners. [Online]. http://h20465.www2.hp.com/gpl/PartnerDetails.aspx?intPartnerId=1-2Z9-973&long=0.0000000000000000000&lat=0.0000000000000000000&PName=SERVUS-ANKARA&Fax=3124721850&Phone=3124721838&Cnry_Code=TR&lng_Code=tr&SLprofile=&prefShopIcon=goldres.gif%2cais.gif%2cap

[15] Cisco Systems. (2010, August) Cisco Catalyst 4503-E Switch. [Online]. http://www.cisco.com/en/US/products/ps9295/index.html

[16] Beckhoff Automation. (2010, August) CP66xx built-in Control Panel with Ethernet interface. [Online]. http://www.beckhoff.com/english.asp?industrial_pc/cp66xx.htm

[17] Anonymous. (2009, February) C2 Knowledge Base. [Online]. http://www.c2.com/cgi/wiki?WaterfallModel

[18] Free Software Foundation. (2007, June) GNU General Public License. [Online]. http://www.gnu.org/licenses/gpl.html

[19] Notetech. (2009, August) Notetech Software Development Life Cycle. [Online]. http://www.notetech.com/software-development-life-cycle.html

[20] Microsoft Corporation. (2010, August) Microsoft Project Professional 2010. [Online]. http://www.microsoft.com/project/en/us/project-professional-2010.aspx

[21] Edgewall Software. (2010, August) The Trac Open Source Project. [Online]. http://trac.edgewall.org/

[22] Microsoft Corporation. (2010, August) Microsoft Visual Studio 2010. [Online]. http://www.microsoft.com/visualstudio/en-us

[23] Microsoft Corporation. (2009, December) Microsoft.NET Framework. [Online]. http://www.microsoft.com/net/

[24] Developer Express Inc. (2010, August) Developer Express.NET WinForms Components & Tools. [Online]. http://www.devexpress.com/Products/NET/Controls/WinForms/

[25] JetBrains. (2010, August) ReSharper: Developer Productivity Tool for Microsoft Visual Studio. [Online]. http://www.jetbrains.com/resharper/

[26] Microsoft Corporation. (2010, June) Microsoft CodePlex Sandcastle Project. [Online]. http://sandcastle.codeplex.com/

[27] EC Software GmbH. (2009, June) Help & Manual Product Overview. [Online]. http://www.ec-software.com/products_hm_overview.html

[28] Microsoft Corporation. (2010, August) Microsoft Developer Network (MSDN). [Online]. http://msdn.microsoft.com/en-us/library/t71a733d%28VS.80%29.aspx

[29] Argonne National Laboratory. (2009, January) Experimental Physics and Industrial Control System. [Online]. http://www.aps.anl.gov/epics/

[30] Brian Ritchie. (2010, August) Dotnet Powered. [Online]. http://www.dotnetpowered.com/languages.aspx

[31] Novell. (2010, June) Mono Project. [Online]. http://www.mono-project.com/Roadmap

[32] Paul Scherrer Institut. (2009, July) EPICS Sharp. [Online]. http://gfa-it.web.psi.ch/epicsSharp/index.php?site=about

[33] Turkish Accelerator Center. (2010, August) Turkish Accelerator and Radiation Facility at Ankara. [Online]. http://www.tarla-fel.org/epics.net/

[34] Teoman Soygul. (2010, August) EPICS.NET Library Home. [Online]. http://www.soygul.com/project/epics.net/

[35] Microsoft Corporation. (2010, August) Windows Workflow Foundation. [Online]. http://msdn.microsoft.com/en-us/netframework/aa663328.aspx

[36] Microsoft Corporation. (2010, August) MSDN Academic Alliance. [Online]. http://msdn.microsoft.com/en-us/academic/bb676724.aspx

[37] Microsoft Corporation. (2010, August) Microsoft System Center Configuration Manager. [Online]. http://www.microsoft.com/systemcenter/en/us/configuration-manager.aspx

[38] Siemens AG. (2010, August) Siemens Automation. [Online]. http://www.automation.siemens.com/mcms/automation/en/automation-systems/process-control-systems/Pages/Default.aspx

[39] Black Box Corporation. (2010, August) Hardened Media Converter Switches, 10/-100-Mbps Copper to 100-Mbps Fiber, 100–240-VAC, SC. [Online]. http://www.blackbox.com/Store/Detail.aspx/Hardened-Media-Converter-Switches-10-100-Mbps-Copper-to-100-Mbps-Fiber-100-240-VAC-SC/LBH100A%C4%82H%C4%82SC

7 Appendix A: List of Software

<table>
<tr><td>Microsoft</td><td>

Software acquired through MSDN Developer Academic Alliance:

- Microsoft Visual Studio 2010 Ultimate x86
- Microsoft Visual Studio Team Foundation Server 2010 x64
- Microsoft Windows Server 2008 R2 Enterprise x64
- Microsoft SQL Server 2008 R2 Enterprise (=Developer) x64
- Microsoft Office Project Server 2010 x64
- Microsoft Windows 7 Enterprise (=Ultimate) x64

Software bought separately with Academic Licenses:

- Microsoft Office Project Professional and Visio Professions x86
- Microsoft Office Professional Plus x86
- Microsoft Systems Center Configuration Manager 2007 R2 x86

</td></tr>
<tr><td>intel</td><td>

Software bought with Academic Licenses:

- Intel® C/C++ Compiler Suite Professional Edition for Windows x86/x64
- Intel® Integrated Performance Primitives (Intel® IPP) x86/x64

</td></tr>
</table>

	• Intel® Math Kernel Library (Intel® MKL) x86/x64 • Intel® Cluster Toolkit x86/x64
NATIONAL INSTRUMENTS	*NI Academic Site License – Research Option 2010 Upgrade + 3 Year SSP* • LabVIEW Core Software • LabVIEW Controls and Embedded Software • LabVIEW Signal Processing and Communications Software
Other	• Developer Express Inc. Developer Express.NET WinForms Components & Tools • JetBrains ReSharper • EC Software Help & Manual