THOMPSON RIVERS UNIVERSITY

SENG 4630 – Safety Critical Software Systems

Failure Alert System for Nuclear Power Plants

Alex Bepple (T00619983)

Ben Lea (T00597297)

Sanyam Gupta (T00650151)

Table Of Contents

List Of Figures

List Of Tables

1 Introduction

A nuclear power plant is an incredibly environmentally friendly and efficient way to source energy. Nuclear power when harnessed correctly can be incredibly beneficial to man however this comes with the risks that come with nuclear fuels. Fallout and radiation as demonstrated in Chernobyl and Fukushima can cause the destruction of large areas for many years after their demise. This raises the importance that these systems have on being safety critical as failures can have drastic consequences. With the use of the Ada programming language a safety critical software can be created that can send early alerts if any anomalies may occur and how best to rectify those issues and take steps to make sure they don't occur.

2 Design Problem

2.1 Problem Definition

Nuclear power facilities are required to increase efficiency, safety, and overall public confidence in nuclear power to bring down cost and remain operational. The current monitoring systems are outdated and result in unnecessary down time, decreased efficiency and pose unnecessary safety risk to the environment and nearby communities.
In order to mitigate the problems described, the task is to design and develop a failure alert system for nuclear power plants. The alert system is to monitor a variety of the powerplants conditions (radiation levels, temperature, ect) and generate warning events when the system is out of normal operating condition. Each warning event has an associated action which varies from taking autonomous measures to warning the operators to manually prevent the occurrence of a failure.

2.2 Design Requirements

2.2.1 Functions

Table 1: User Interface	
ID	Description
1_Real Time Data	The system must provide real time monitoring of the following plant parameters: temperature, pressure, radiation level, and power output.
2_Security	The system must have measures in place to prevent unauthorized access.
3_Hardware Interfaces	The system provides interfaces which allow it to connect to external hardware systems.
4_Software	The system's software interfaces must have a reasonable learning curve.

Interfaces	
5_Communication Protocols	Communication is available between computers and sensors.
6_Saftey	The systems must deliver important information to the user clearly

Table 2: Alarms	
ID	Description
1_Real Time Data	The system must provide real time monitoring of the following plant parameters: temperature, pressure, radiation level, and power output.
2_Alarms and Events	The system must be able to generate alarms containing severity level and information regarding the alarm.
3_UserInterface	The system must provide a user interface which will allow users to monitor the data and acknowledge alarms.
4_Security	The system must have measures in place to prevent unauthorized access.
5_Hardware Interfaces	The system provides interfaces which allow it to connect to external hardware systems.
6_Software Interfaces	The system's software interfaces must have a reasonable learning curve.
7_Communication Protocols	Communication is available between computers and sensors.
8_Safety	Alarms must indicate clearly what is wrong, and do so in real time to reduce safety risks associated with delays in relaying this information.
9_Risk	Failure to sound an alarm can cause a possible meltdown or disaster

Table 3: Automated plant monitoring and control	
ID	Description

1_Real Time Data	The system must provide real time monitoring of the following plant parameters: temperature, pressure, radiation level, and power output.
2_Alarms and Events	The system must be able to generate alarms containing severity level and information regarding the alarm.
3_UserInterface	The system must provide a user interface which will allow users to monitor the data and acknowledge alarms.
4_Security	The system must have measures in place to prevent unauthorized access.
5_Hardware Interfaces	The system provides interfaces which allow it to connect to external hardware systems.
6_Software Interfaces	The system's software interfaces must have a reasonable learning curve.
7_Communication Protocols	Communication is available between computers and sensors.
8_Safety	Redundancies must be in place to handle errors and prevent critical failure.
9_Risk	Errors in code can cause catastrophic failure of the entire system.

Table 4: Low Level Software Requirements	
ID	Description
Language	ADA 8.0
Control interface	Must have a concurrent connection between the sensors and the alarms. Must detect when connections get dropped Must handle disconnections gracefully and attempt automated fix with warning. Must poll the sensor data and send result of logic to alarms
Alarm	Must interface with the controller Must understand and handle the received messages.

Data Exchanged

Table 5: Data Exchanged	
ID	Description
Temperature	TCP ● UDP Controller Port ● Frequency of transmission UDP ● Current temperature data
Pressure	TCP ● UDP Controller Port ● Frequency of transmission UDP ● Current pressure data
Alarm for coolant water pressure	Sound alarm when coolant water pressure is above 21 MPa and below 16MPa (max is 22.064 MPa), minimum(needed to have a working reactor) is 15 MPa

2.2.2 Objectives

- Fault Tolerant: The system must be fault tolerant such that a single point of failure should not result in a catastrophic failure.
- Scalability: The system must be scalable to accommodate adding and removing sensors.
- Safety: The system must be designed so that it does not pose a large risk to the general public.
- Environmental Sustainability: The system must be designed with environmental sustainability in mind. The software should aim to detect any abnormalities that may lead to accelerated wear on the system which may lead to a reactor melt down. The software should help improve the reactor's efficiency and therefore minimize the nuclear waste.
- Societal impacts: The system must efficiently power the homes and buildings of nearby cities.
- Economic factors: The system must be designed to be cost effective throughout its life span
- Security: The system must have measures in place to limit access to unauthorized individuals.
- Maintainability: The code should be designed to be maintained easily with good design.

2.2.3 Constraints

- The software must be written in Ada programming language.
- The system must be compliant to NSCA (nuclear safety and control act)
- Must keep temperatures below 2700 degrees celsius to avoid meltdown
- The system must be at least 30 kilometers from the nearest city.
- The software must be completed by 31/03/2023.
- The system is designed without hardware implementation and uses simulated data to simulate the power plant.

3 Solution

In this section we will be discussing the various design decisions we considered and giving the rationale on why we chose our final solution. The main considerations are the system architecture and communication protocol. These are the high level designs which have a large impact on the overall design of the project. Each of the solutions will be weighted on the following criteria: safety, fault tolerance, security, maintainability, and scalability.

To give a fair comparison for each of the solutions it is assuming that each of the sensors have an equal probability of failure and that the cost of a failure is equal for each sensor.

Solution 1

Architectural Design 1:

The architecture design specifies the relationship between sensors, controller(s), and monitoring system. In our initial solution the team considered having each of the physical sensors communicate to a single main controller which would gather the information and relay it to the monitoring system. This design is simple to implement and is resource efficient as there is no duplicated effort.

However, the risk analysis shows that the solution has a high probability of failure and is thus lacking fault tolerance and safety requirements, because if a single component fails the entire system would go down.

Communications Protocol 1 : Shared Memory

In solution 1 the team considered using shared memory. Shared memory is memory that can be simultaneously accessed by multiple programs to provide communication between them. This method is extremely low latency as it is just reading and writing to a memory block. The concept is simple and easy to implement.

However managing access to a shared memory is quite complex, careful synchronization measures must be taken to ensure the memory is not accessed by two programs at the same

time. Additionally shared memory does not scale easily when there are a large number of devices which need to access the memory. Lastly, this method becomes very impractical when the physical devices must be geographically far from each other, as the bits will be required to be sent over a network, which will require additional communication methods.

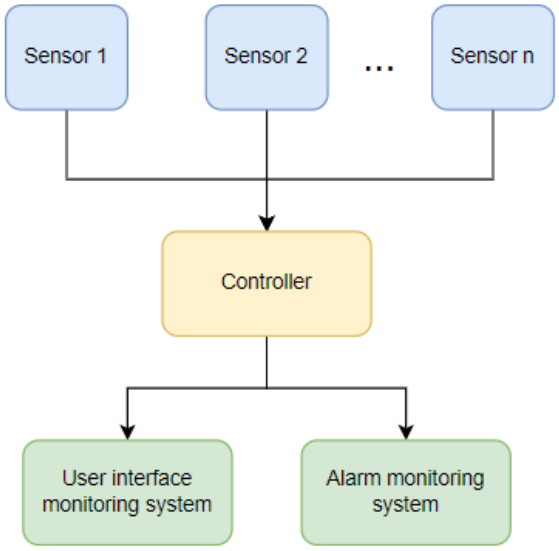


Fig 1 Solution 1 System Architecture

Solution 2

Architectural design 2:

With our second solution we considered connecting each sensor to its own controller. Connecting each sensor to its own controller would relieve the major failure point in the previous system, the single controller. However if a single sensor, or controller fails, it will only affect that sensor, which can then relay the alarms and warnings to the UI and the alarm system that a component failed.

Communication Protocol 2: TCP (Transmission Control Protocol)

TCP is a time tested popular communication protocol. It is designed to be very reliable, such that messages are delivered with no errors and in the correct order. The protocol uses checksums to ensure error detection, lastly the protocol and mechanism in place to control congestion and thus give steady communication times. However, the trade off for TCP high reliability is higher overhead and latency.

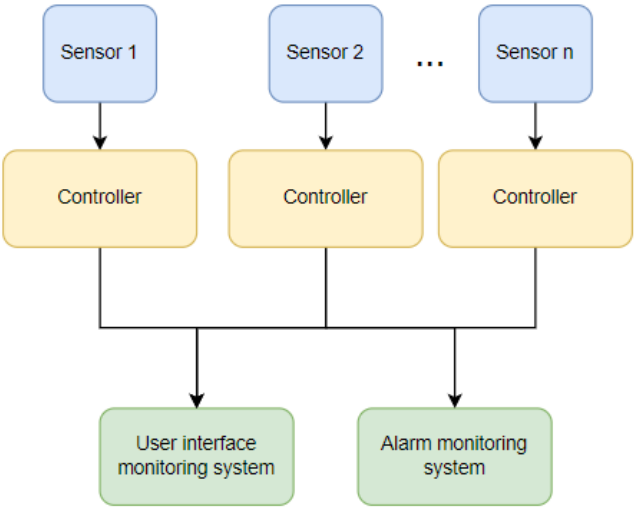


Fig 2 Solution 2 System Architecture

Final Solution

Architectural Design 3:

For our final solution we decided on having 2 redundant sensors for each sensor with each of the sensors hooked in parallel with each of the redundant controllers. This solution provides increased reliability, having multiple sensors and multiple controllers measuring the same physical parameter increases the overall reliability of the system, as shown by the risk analysis. Better fault tolerance as having multiple sensors in place, if one sensor has failed, it is easy to detect and isolate the fault. This solution increases accuracy, having redundant sensors helps improve the accuracy of the system.

However, the system has an increased cost to purchase the components and implement the system. Adding more components increases the complexity of the system, each of the components needs to communicate with multiple other components. The controller logic becomes more complex as it will require implementing a voting system. Since the system is a safety critical environment we are more concerned with robustness than cost so we will implement this solution.

Communication Protocol 3: TCP + UDP (User Datagram Protocol)

UDP is another commonly used communication protocol. Compared to TCP it has very low overhead and low latency, however it is less reliable than TCP. The sensors and controllers in our application are measuring and communicating the sensor data many times a second.

Because of this, UDP provides a significant advantage as we can process the data closer to real time. The unreliability of UDP is not as large of a concern for this application, because if a small percentage of the data packets are dropped, we will immediately be receiving more data packets with the up to date information, and with the advanced logic in the controller we can detect if a sensor has failed to send and attempt to reconnect.

Our final communication design features the leveraging of the strengths of UDP and TCP to provide fast UDP transmission of sensor data, while having dependable TCP connection for sensor setup and alarm communication. This uses Adas tasks package to allow for both socket styles to be used simultaneously to achieve this goal. Something we have never seen implemented before.

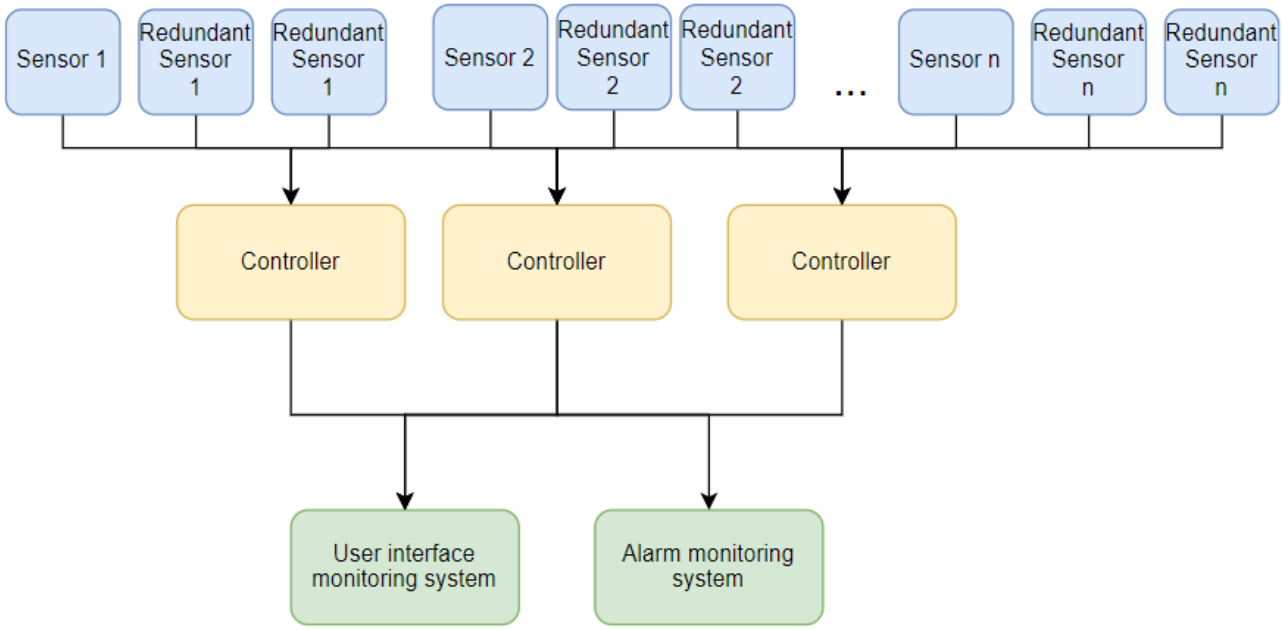


Fig 3 Final Solution System Architecture

Decomposition Design

The decomposition design describes the breakdown of each of the subsystems software design. The figure below shows each module which is designed to run on an isolated machine: sensor,

controller, and alarm. Each module is broken down into packages to encapsulate the related data and functions.

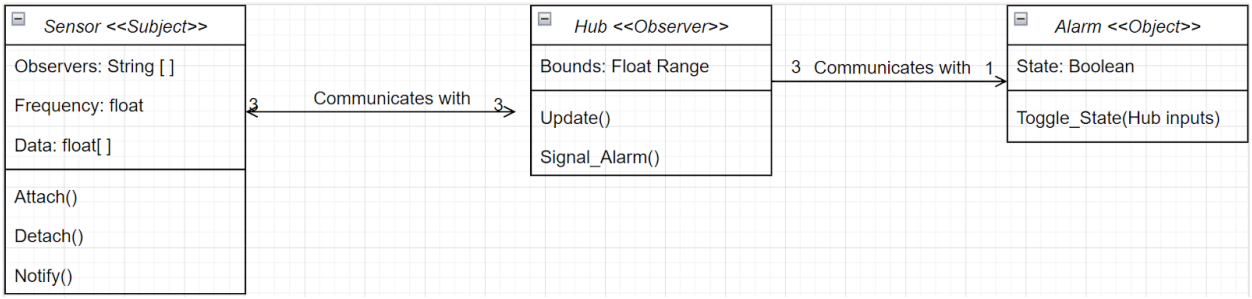


Fig 4 Decomposition Design

Analysis:

To analyze the risk of each solution we calculated the probability of failure for each of the design architectures. We did this by giving each component a probability of failure, then based on the layouts of the components calculated the likelihood of failure which would result in bad reading which would cause the system to potentially give a false alarm. The formula used for calculating the probability are as followed:

$$P_f = g(\mathbf{F}) = 1 - \prod_{i=1}^{n} (1 - F_i)$$

Fig 5 Probability of failure of components in series

$$P_f = g(\mathbf{F}) = \prod_{i=1}^{n} F_i$$

Fig 6 Probability of failure of components in parallel

Risk Levels

0	No Consequence
1	Added task
2	Minimal Loss

3	Lose money
4	Near meltdown
5	Catastrophe

Table 6 Definition of Risk Levels

Definition:

True Positive: Relaying no cause for concern and the reactor is functioning properly.

False Positive: Relaying no cause for concern and the reactor is malfunctioning.

True Negative: Relaying a cause for concern and the reactor is malfunctioning.

False Negative: Relaying a cause for concern and the reactor is functioning properly.

Probability is the likelihood of the event in a 1 year period.

Component values:

Sensor	Risk	Probability	Result
True Positive	0	0.90	0
False Positive	5	0.01	0.05
True Negative	4	0.08	0.32
False Negative	2	0.01	0.02

Table7 Risk values of Sensor

Controller	Risk	Probability	Result
True Positive	0	0.90	0.0
False Positive	5	0.005	0.025
True Negative	4	0.09	0.36
False Negative	3	0.005	0.015

Table 8 Risk values of Controller

Alarm	Risk	Probability	Risk Result
True Positive	0	0.005	0.00
False Positive	5	0.03	0.15
True Negative	3	0.94	0.00
False Negative	2	0.025	0.05

Table 9 Risk values of Alarm

Component Totals:

Component	Risk Result total	Probability of failure
Sensor	0.39	0.10
Hub Controller	0.4	0.10
Alarm	0.2	0.05

Table 10 Total Risk values

From these results we can see that the redundancy of the sensors and hubs should be prioritized to reduce the systems overall risk. Our solutions will reflect the addition of these redundancies.

	Solution 1	Solution 2	Final Solution
Probability of failure	0.230500	0.145855	0.060441

Table 11 Probabilities of failure

These probabilities show the effect of the redundancy at reducing the failure probability of a system.

Features

- Decoupled design separating the logic center from the data collection and alarm systems.

- Polling system that sounds the alarm as soon as a majority of sensors detect a dangerous reading. This will be completed in through the compilation of the sensor, hub, and controller in ADA 2012,

- The logic is able to detect when measurements are out of an acceptable range through the use of the logic coded in ADA 2012.

- The system has a display that alerts the users of warnings and errors of which situation is failing.

- Communication between the sensors, controllers, and alarms facilitated by simultaneous UDP and TCP transmission across the subnet.

- Concurrent tasking to facilitate dependable communication between each layer of the system, and concurrent runtimes of messaging and computation.

Environmental, Societal, Safety, and Economic Considerations

Environmental: The final design is very robust minimizing the possibility of failure thus ensuring any damage to the environment that could occur from a failure is minimized as well. The environment was a consideration that raised the importance of our failure handling in our system, and was thought of during our risk mediation technique brainstorming.

Societal: Our setting of developing for a nuclear reactor brought up the obvious risks to nearby society with the parallels that can be drawn from the well known Chernobyl disaster. Its impact on the society it was a part of can still be seen today years later with abandoned cities in its radiation radius. This consideration increased the importance of a robust system to reduce these risks to society.

Safety: The robustness of the design ensured that the amount of negative results is minimized and if any were to come up they would be easy to solve. Safety was a large concern during our brainstorming and factoring of adding redundancy. This redundancy would provide a much higher guarantee of safety to the system.

Economic: The economic impacts were a heavy factor in our consideration of the risk factors involved in component failures. We had to consider that a failure of the system would not only be potentially harmful for nearby users, but would also implicate a large economic burden to fix. This raised the importance of robustness in the system to deter these economical catastrophes from occurring on failure.

Limitations

Due to our limitation of not being able to implement the system into a nuclear reactor environment we will have to take some liberties. Some such liberties will be the use of a data generator to simulate the action of a real system, and the use of a text file as our output of the alarm. This handles the issue of input and output mediums. We also found a limit on our computing power. Attempting to simultaneously simulate an increasingly high volume of docker images on one machine caused unforeseen issues with computational speed. To circumvent these issues we tested a proof of concept that is less robust with a reduced number of concurrent redundant systems. These temporary solutions were to be designed in such a way with low coupling so that portability can be high when their successors become available.

4 Decision Matrix

Objective (scale factor)	**Solution 1**	**Solution 2**	**Final Solution**
Safety(*2)	0	1	2
Cost (*0.25)	2	1	0
Maintainability(*1)	2	1	1
TOTAL	2.5	3.25	5

Table 12 Decision Matrix

The above decision matrix shows how we considered objectives while planning our solution. With a system like this it is more desirable to have a safe system at the expense of cost and maintainability due to the coupled complexity that redundancy and robustness requires. Cost is the least of our worries as we have lives at stake if catastrophic circumstances occur so funding should be made available to make it safe. Maintainability is considered as a downside of the system being more redundant. This is included to be able to see the natural tradeoff of this factor with the safety factor. The Weighting of safety in this way skews the selection towards the more robust system which is what is desired. While maintainability got lower it did not become impossible to maintain due to our modular implementation.

5 Team Work

Meeting 1

Time: 2023-01-23 10:00 am to 11:00 am

Agenda:

1. Discuss project introduction.
2. Discuss the project's problem definition.
3. Assign tasks to each team member.

Team Member	Previous Task	Completion State	Next Task
Ben Lea	N/A	N/A	Collaboratively write problem definition and introduction. Record meeting notes. Contribute 2 functions, 2 objectives and 1 constraint.
Alex Bepple	N/A	N/A	Collaboratively write problem definition and introduction. Review and edit problem definition and introduction. Contribute 2 functions, 2 objectives and 1 constraint.
Sanyam Gupta	N/A	N/A	Collaboratively write problem definition and introduction. Review and edit problem definition and introduction. Contribute 2 functions, 2 objectives and 1 constraint.

Meeting 2

Time: 2023-02-07 12:00 pm to 12:45 pm

Agenda:

1. Discuss project introduction.
2. Discuss the project's problem definition.
3. Assign tasks to each team member.

Team Member	Previous Task	Completion State	Next Task
Ben Lea	Collaboratively write problem definition and introduction. Review and edit problem definition and introduction. Contribute 2 functions, 2 objectives and 1 constraint.	100%	Work on low level requirements
Alex Bepple	Collaboratively write problem definition and introduction. Review and edit problem definition and introduction. Contribute 2 functions, 2 objectives and 1 constraint.	100%	Work on low level requirements
Sanyam Gupta	Collaboratively write problem definition and introduction. Review and edit problem definition and introduction. Contribute 2 functions, 2 objectives and 1 constraint.	100%	Work on low level requirements

Meeting 3

Time: 2023-02-14 12:00 pm to 12:45 pm

Agenda:

1. Discuss project introduction.
2. Discuss the project's problem definition.
3. Assign tasks to each team member.

Team Member	Previous Task	Completion State	Next Task
Ben Lea	Work on low level requirements	40%	Update the gannt chart
Alex Bepple	Work on low level requirements	40%	Work on features,limitations and considerations,solution analysis
Sanyam Gupta	Work on low level requirements	40%	Work on features,limitations and considerations, solution analysis

Meeting 4

Time: 2023-03-2 2:30 pm to 3:45 pm

Agenda:

1. Discuss project introduction.
2. Discuss the project's problem definition.
3. Assign tasks to each team member.

Team Member	Previous Task	Completion State	Next Task
Ben Lea	Work on low level requirements	70%	Work on low level requirements
Alex Bepple	Work on features,limitations	50%	Work on low level requirements

	and considerations, solution analysis		
Sanyam Gupta	Work on features,limitations and considerations, solution analysis	50%	Work on low level requirements

Meeting 5

Time: 2023-03-2 2:30 pm to 11:00 am

Agenda:

1. Discuss project design
2. Decide design objectives to include in decision matrix
3. Create decision matrix

Team Member	Previous Task	Completion State	Next Task
Ben Lea	Work on low level requirements	100%	Design selection
Alex Bepple	Work on features,limitations and considerations, solution analysis	100%	Design selection
Sanyam Gupta	Work on features,limitations and considerations, solution analysis	100%	Design selection

Meeting 6

Time: 2023-03-30-2:30 pm to 2023-03-31 11:59 pm

Agenda:

1. Marathon Code Session

Team Member	Previous Task	Completion State	Next Task
Ben Lea	Design selection	100%	Alarms and controller logic
Alex Bepple	Design selection	100%	Communication between sensors, controller, and alarms.
Sanyam Gupta	Design selection	100%	Create data generation code

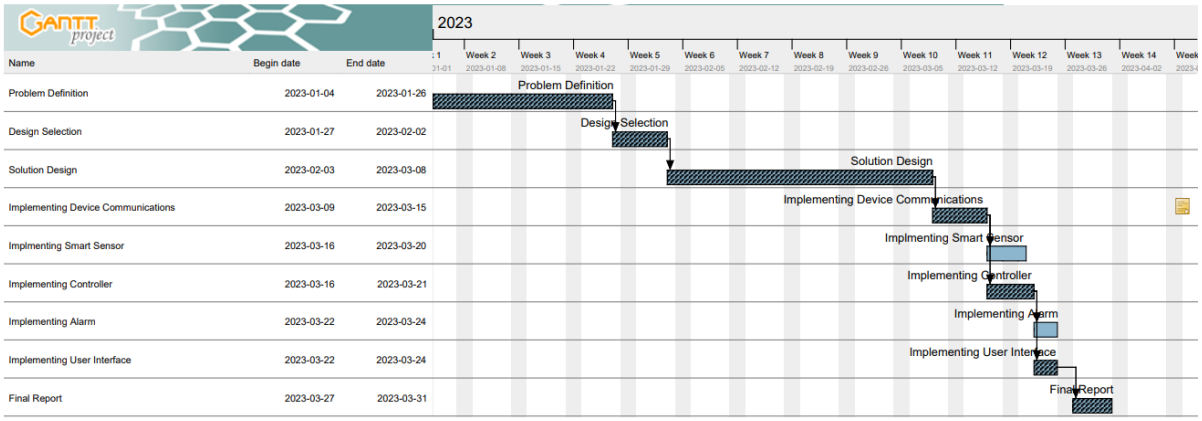
6 Project Management

Failure Alert System for Nuclear Power Plants

Mar. 8, 2023

Gantt Chart

3



Tasks

Name	Begin date	End date
Problem Definition	2023-01-04	2023-01-26
Design Selection	2023-01-27	2023-02-02
Solution Design	2023-02-03	2023-03-08
Implementing Device Communications	2023-03-09	2023-03-15
Implmenting Smart Sensor	2023-03-16	2023-03-20
Implementing Controller	2023-03-16	2023-03-21
Implementing Alarm	2023-03-22	2023-03-24
Implementing User Interface	2023-03-22	2023-03-24
Final Report	2023-03-27	2023-03-31

Fig 7 Project Gantt Chart

The critical path is shown by the black and blue bars. Due to the short time period, small team, and nature of the project, we had very little activities with slack time, any delays meant features of the project had to be cut.

The ***Implementation of the Communications*** task took an extra 2 days to complete which cut into the ***Implementation of User Interface*** leaving us only 1 day to complete this task. The team agreed that the user interface was not as critical of a task and that it could be simplified to save time.

7 Conclusion and Future Work

This safety critical software was made in Ada with fault tolerance, scalability and maintainability in mind. To replicate sensor data a function would randomly generate data. Multiple sensor simulators of varying kinds could be run at once with each communicating with multiple controllers over UDP and TCP. The controller would then poll the sensors and give warnings as needed to the alarms, data between controllers and alarms is again communicated using TCP. Finally alarms poll controllers and alert the user as needed.

Controllers check to see if sensor readings are outside of the stated safe boundaries and send a warning if the polling indicates there is a need for concern.Through the use of tasking it's possible to have concurrent messaging between layers and computation within layers. Redundancies of sensors, controllers and alarms were created and allowed for more to be added so that if ever one of any components was to fail the system itself could still function. This is a key requirement for safety critical projects such as this where a single point of failure is unacceptable.

When looking at future work connecting and using actual sensors would most likely be the next natural step. This would help take the prototype from purely software to an integration of software and hardware. Currently, this could not be done because we did not have access to the sensor and alarm materials necessary for this functionality.