# Part 1

1. Introduction

A picture containing diagram

Description automatically generated

* 1. The RTU’s controls the actuators and retrieves sensor readings for the plant. These produce data sent to the OPC server, describing the state of the system, as well as receiving instructions on how to control the system. The OPC server servers as an interconnection between the RTU’s and the SCADA system, displaying data from the plant, as well as controlling it. Data that’s used and/or produced in the SCADA system gets historized in a database. The data stored in the database, is the basis for the ERP.
  2. For this system, det OPC server would serve as the Data engine. It receives data from the RTU’s and other sources. With the data used by the SCADA components. The data engine makes sure production data is accessible throughout the network, regardless of system and manufacturer.
  3. A ERP system makes it easier for management to take business decisions, with the help of data from the plant.
  4. One should avoid connecting the network to the internet. Additionally, the ERP system should be split from the rest of the network with a firewall for access control.

1. Analysis and Design
   1. Five functional requirements would be Level control, temperature control, Alarm on tank level, flow control of hot water and flow control of cold water.
   2. Three non-functional requirements would be easy access of data for development of ML , realistic disturbances, intuitive ui on smart components.
2. Digitalization
   1. With JSON and XML its possible to build structures if you want to store custom objects, and the have to be accessed by a programming language. The CSV format works simply by separating data with a delimiter, and having a header. This format can be read by Excel and could be easier read for people without a technical background. For this system CSV should be used.

|  |
| --- |
| Timestamp, Instr1, Instr2, Instr3, Instr4, Instr5  2022-05-12 09.00, 4.36, 18.666, -42.2, 1015.3, -0.7 |

|  |
| --- |
| 1. using System; 2. using System.Collections.Generic; 3. using System.Linq; 4. using System.Text; 5. using System.Threading.Tasks; 6. using System.Data.SqlClient; 7. using CsvHelper; 8. using System.IO; 9. using System.Globalization; 10. namespace SqlToCsv 11. { 12. internal class Program 13. { 14. static void Main(string[] args) 15. { 16. string DataSource = @"Data source= OSL-0687\SQLEXPRESS"; 17. string database = "SCADA"; 18. string integratedSec = "true"; 19. string ConnectionString = "Data Source=OSL-0687\\SQLEXPRESS;Initial Catalog=Kemira;Integrated Security=True"; 20. SQL sql = new SQL(ConnectionString); 21. string dataQuery = "Select Top 20 from IOData ORDER BY Timestamp DESC"; 23. } 24. } 25. public class SQL 26. { 27. public SqlConnection conn; 28. public SQL(string ConnectionString) 29. { 31. Console.WriteLine("Opening Connection..."); 32. //Opens Connection 33. conn = new SqlConnection(ConnectionString); 34. try 35. { 36. conn.Open(); 37. Console.WriteLine("Connection Succesfull"); 38. } 39. catch (Exception e) 40. { 41. Console.WriteLine("Error: " + e.Message); 42. } 43. conn.Close(); 44. Console.ReadLine(); 45. } 46. public List<IOData> QueryingData(string query) 47. { 48. List<IOData> list = new List<IOData>(); 49. SqlCommand cmd = new SqlCommand(query, conn); 50. SqlDataReader dr = cmd.ExecuteReader(); 51. while (dr.Read()) 52. { 53. IOData data = new IOData() 54. { 55. Timestamp = dr["TimeStamp"].ToString(), 56. Tag1 = (float)dr["Tag1"], 57. Tag2 = (float)dr["Tag2"], 58. Tag3 = (float)dr["Tag3"], 59. Tag4 = (float)dr["Tag4"], 60. Tag5 = (float)dr["Tag5"] 61. }; 62. list.Add(data); 63. } 64. return list; 65. } 67. } 68. public class CSV 69. { 70. public string path = "path\\to\\file.csv"; 71. public void Write(List<IOData> data) 72. { 73. using (var writer = new StreamWriter(path)) 74. using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture)) 75. { 76. csv.WriteRecords(data); 77. } 78. } 79. public List<IOData> Read() 80. { 81. var list = new List<IOData>(); 82. using (var reader = new StreamReader(path)) 83. using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture)) 84. { 85. var records = csv.GetRecords<IOData>(); 86. } 87. return list; 88. } 89. } 90. public class IOData 91. { 92. public string Timestamp { get; set; } 93. public float Tag1 { get; set; } 94. public float Tag2 { get; set; } 95. public float Tag3 { get; set; } 96. public float Tag4 { get; set; } 97. public float Tag5 { get; set; } 98. } 99. } |

1. Communication

a. The smart instruments should have automatic adjustment from wear. Indication when maintenance is needed. Displays with graphs and good ui.

b. ??

1. Multitasking
   1. Or is it grouped into temperature, level and/or flow?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hot Water inlet | Cold Water Inlet | Tank temperature | Outlet | Tank Level | Mixer |
| PT02011 | PT02010 | LZT02003 | P02009 |  |  |
| TT02001 | TT02002 | TT02004 | HV02013 |  |  |
| FT02005 | FT02006 | LZT02009 | HZV02014 |  |  |
| HZV02010 | TZV02009 | LIC02009 | T02007 |  |  |
| TZV02003 | HZV02011 |  |  |  |  |
| GT02003 | GT02009 |  |  |  |  |
| TIZ02003 |  |  |  |  |  |
| TT02003 |  |  |  |  |  |

* 1. An application could either be running, not running, or idle when waiting on resources. OR. A multitasking application has a timer, IRS and tasks running. It could either be running the timer, the IRS or the task.
  2. See reference
  3. The total running time was much shorter for the threads in code 0. This is because them threads runs at the same time . While for Code 1, the threads are querying to run, the next cant start running before the previous one is finished. For the system at hand the need for responsiveness depends on how large the tank is compared to the inlet flows. Regardless, a responsive control system is always wanted. The code structure from Code 0 would therefore be the best choice.

1. Alarm System
   1. Alarm state diagram

Diagram

Description automatically generated

* 1. The alarm philosophy document should explain how to make it easy for operators to understand what the issue is, the alarms should be easily acknowledged and shelved by the operators. The alarms are generated by the SCADA system and stored in the SQL database. For this system alarms are needed for level in tank, and temperature in tank.
  2. Alarm Pseduo Script:

|  |
| --- |
| 1. public class Alarm 2. { 3. public AlarmTemplate AlarmGenerate(int TagId,float value) 4. { 5. //Gets low Level for specific tag 6. int lowLevel = AlarmConfig.LowLevel(TagId); 7. AlarmTemplate \_alarm = new AlarmTemplate(); 8. if (lowLevel > value) 9. { 11. \_alarm.TagId = TagId; 12. \_alarm.AlarmType = "Low"; 13. \_alarm.TimeStamp = DateTime.Now.ToString(); 14. //This needs to be checkes in DB 15. \_alarm.Active = AlarmConfig.Active(TagId, "Low"); 16. \_alarm.severity = AlarmConfig.Severity(TagId); 17. } 18. return \_alarm; 19. } 20. } 21. public class AlarmTemplate 22. { 23. public string AlarmType { get; set; } 24. public string TagId { get; set; } 25. public string TimeStamp { get; set; } 26. public bool Active { get; set; } 27. public int severity { get; set; } 28. } |