# Industrial Informatics
# Automation Department
# Year Project

# 2022-2023

**Professor**: Honoriu Valean
**Team members**:
Voic Andrei-Florin - Team Leader, Software Developer
Oprea Sergiu-Daniel - Software Developer, Software Tester
Moldovan Ioan-Iulian - Database Developer
**Group**: 30332_2

# Contents

# PLC Reader Application Documentation

## *Chapter 1: Introduction*

### 1.1 Scope

The Programmable Logic Controller (PLC) Reader Application is a Windows Forms application designed to interact with and extract data from PLCs. In the complex world of industrial automation, PLCs form the backbone of many systems, controlling a vast array of processes. However, their utility is only as good as the ability to understand, analyze, and act on the data they generate. That's where the PLC Reader Application comes in.

The PLC Reader Application is designed to interface with PLCs, extract valuable data, store that data for analysis, and then present the user with insightful analytics. This provides a comprehensive solution that enables users to effectively monitor, control, and optimize their industrial processes.

While many applications may only allow for simple read/write operations or barebones data visualization, the PLC Reader Application goes above and beyond. It provides users with robust capabilities for real-time monitoring, historical data storage, and in-depth analytics.

## 1.2 Overview

This application interacts directly with PLCs, utilizing the S7.Net library to facilitate communication. The PLC Reader Application is capable of establishing and managing multiple PLC connections simultaneously, allowing for broad system visibility.

The data collected from the PLCs is stored in a local database, ensuring that historical data is available for analysis. This is crucial for identifying trends, diagnosing problems, and optimizing system performance.

## 1.3 Goals

The primary goals of the PLC Reader Application are to:

-**Improve efficiency**: By providing real-time data and historical trends, the PLC Reader Application allows users to identify inefficiencies and bottlenecks in their systems, enabling them to make changes that optimize system performance.

**-Reduce costs**: With better visibility into system performance, users can identify and address issues more quickly, reducing downtime and maintenance costs.

**-Facilitate better decision-making**: Armed with robust analytics and insights, users can make more informed decisions about their systems, improving overall operational effectiveness.

## 1.4 Functionalities

The PLC Reader Application has several key functionalities:

**-PLC Communication and Display**: The application can establish connections with multiple PLCs, read data from these PLCs, and display this data in real-time.

**-Data Storage**: The application can store PLC data in a local database, preserving historical data for future analysis.

**-Data Analysis and Reporting**: The application provides robust data analysis capabilities, allowing users to generate reports that highlight key performance indicators and trends.

**-Graphical Visualization**: The application provides graphical visualization of data, making it easier to understand trends and patterns.

By combining these functionalities into a single, user-friendly application, the PLC Reader Application serves as a powerful tool for anyone working with PLCs and industrial automation systems.

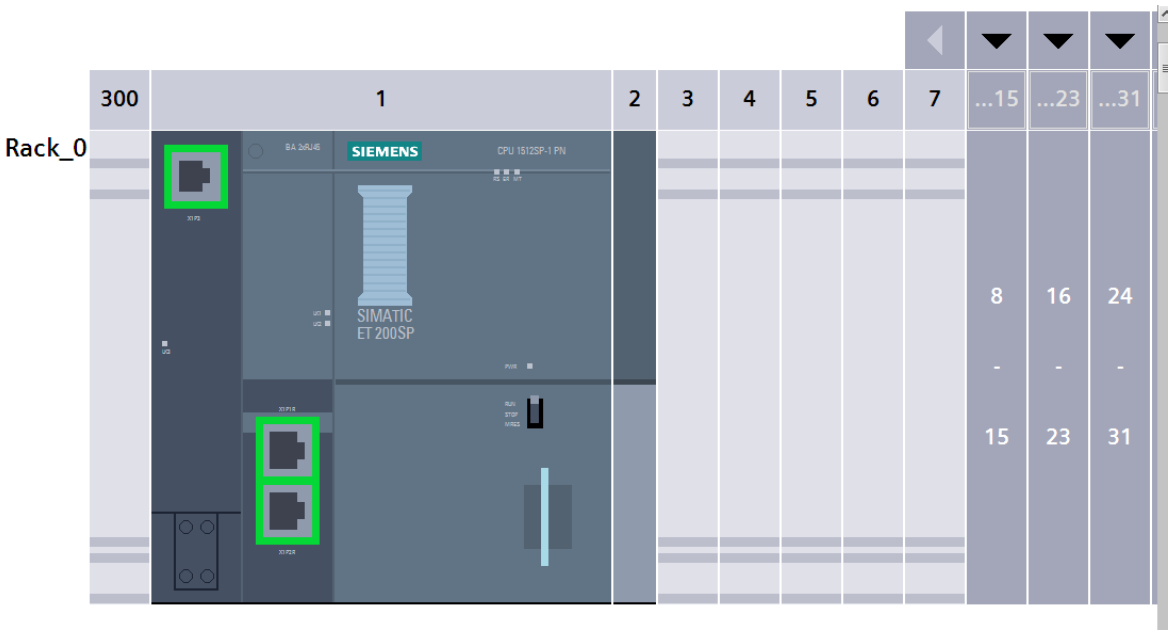
# *Chapter 2: Hardware Component*

## 2.1 Hardware Configuration

The PLC model used for this project is the Siemens Simatic S7-1500, a popular and robust PLC that supports a wide range of automation tasks.

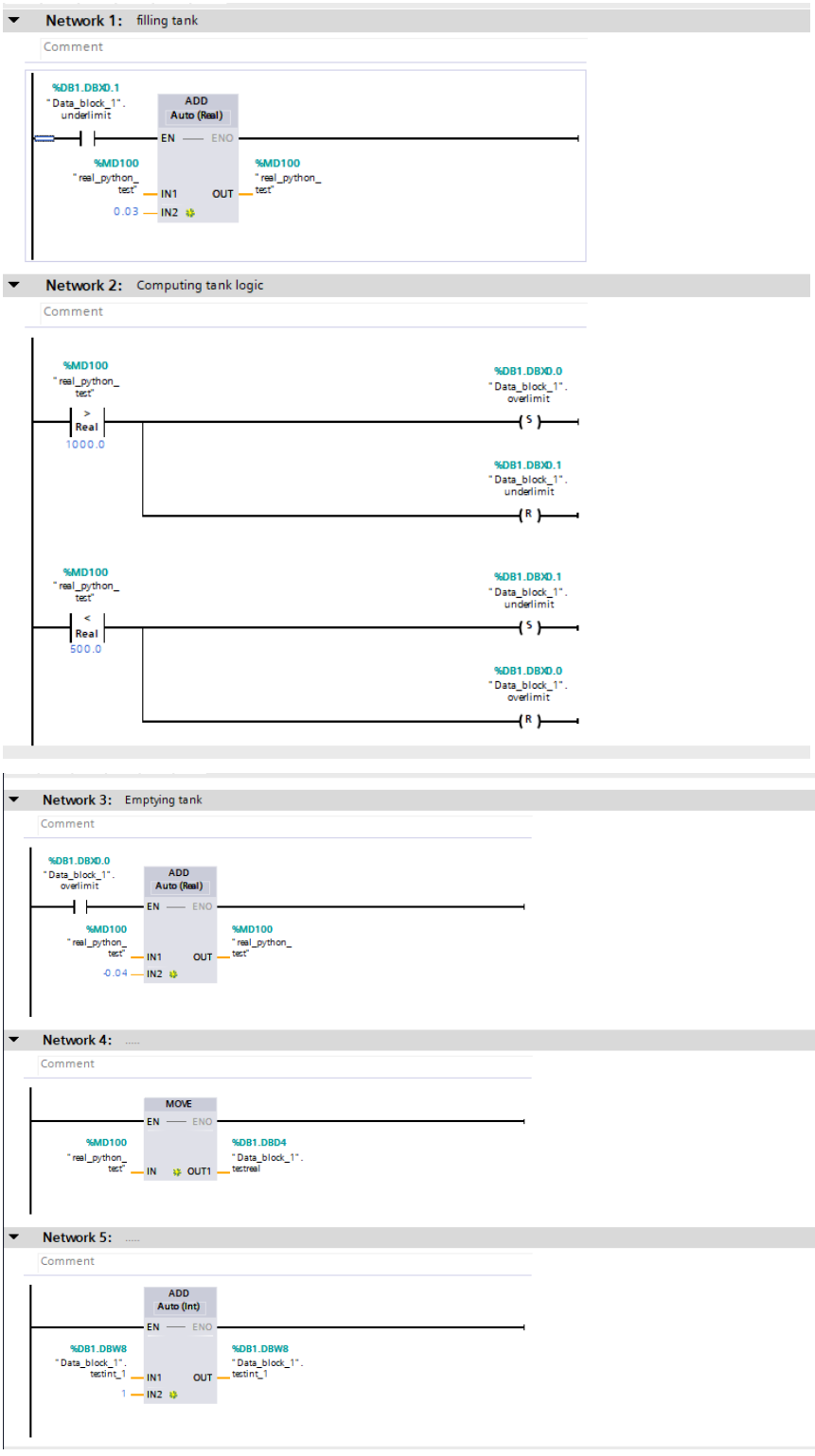
For the purpose of this application, we have created a simple configuration that allows the PLC to communicate with a Windows-based application. The PLC is connected to a network switch which is further connected to a PC where the PLC Reader Application is running. The connection between the PC and the PLC is made via a standard Ethernet cable. The PLC and PC are configured to be on the same subnet to facilitate communication. The specific model is S7-1512SP-1PN.

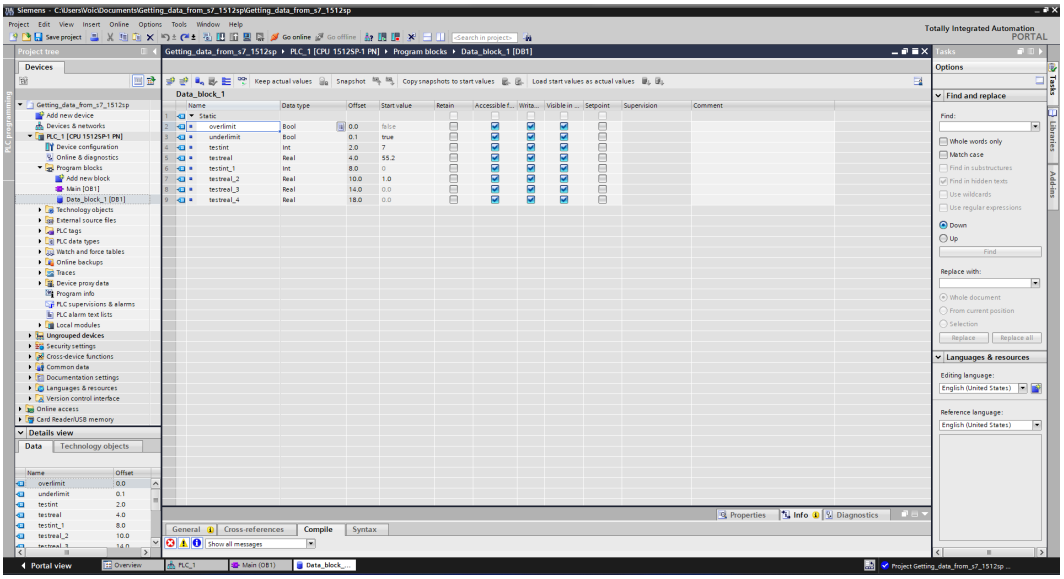

SIEMENS
CPU600
SIEMENS
SIMATIC
ET 200SP


300
1
2
3
4
5
6
7
...15
...23
...31
Rack_0
SIEMENS
CPU 1512SP-1 PN
SIMATIC
ET 200SP
8
-
15
16
-
23
24
-
31

## 2.2 TIA Portal Program

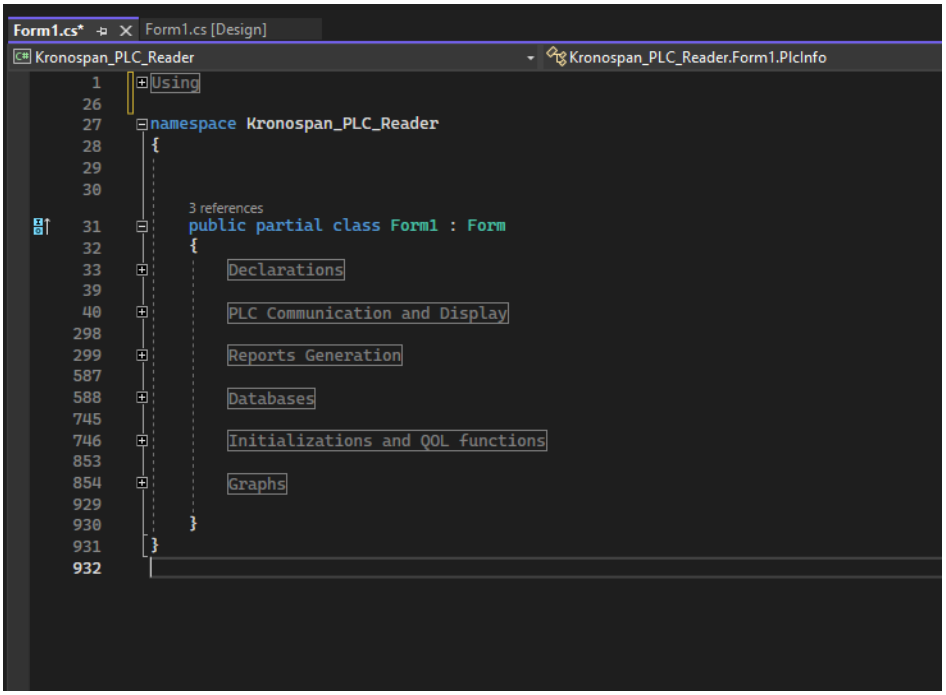
For this PLC Reader Application, a simple PLC program was developed using the TIA Portal for testing purposes. The program involves a data block (DB1) that saves the generated random data for testing. This DB includes the data types Integer and Real.

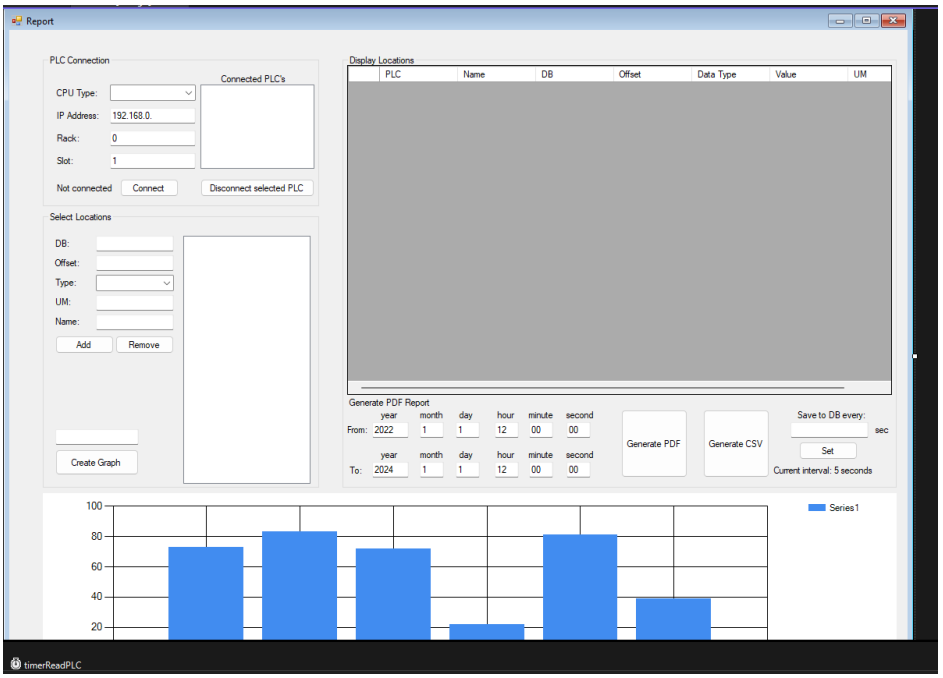



Siemens - C:\Users\Voicl\Documents\Getting_data_from_s7_1512sp\Getting_data_from_s7_1512sp
Totally Integrated Automation
PORTAL
Project tree
Devices
Getting_data_from_s7_1512sp ▸ PLC_1 [CPU 1512SP-1 PN] ▸ Program blocks ▸ Data_block_1 [DB1]
Data_block_1
Name
Data type
Offset
Start value
Retain
Accessible f...
Writa...
Visible in ...
Setpoint
Supervision
Comment
overlimit
underlimit
testint
testreal
testint_1
testreal_2
testreal_3
testreal_4
Tasks
Options
Find and replace
Languages & resources
Details view
Data
Technology objects
Properties
Info
Diagnostics
General
Cross-references
Compile
Syntax
Portal view
Overview

# Chapter 3: Software Component

## 3.1 Main Components of the Program



### 3.1.1 PLC Communication and Display

This region contains classes and methods for managing connections with Programmable Logic Controllers (PLCs), displaying connection status, and reading data from PLCs.

**-PlcInfo**

This class encapsulates information about a PLC, including an instance of the PLC, its name, and a System.Windows.Forms.Timer associated with it. It has a constructor that takes these three parameters to create an instance of PlcInfo.

**-button_Connect_PLC_Click(object sender, EventArgs e)**

This event method is invoked when the "Connect PLC" button is clicked. It reads the CPU type, IP address, rack number, and slot number from the user interface, then creates a new Plc instance and attempts to open a connection to it. If the connection is successful, it updates the connection status label, starts a new Timer with a 1-second interval, and adds a new PlcInfo object to the plcs list. It also adds a new item to the listView_ConnectedPLCs list view.

**-Timer_TickForPlc(Plc plc, string plcName)**

This method is invoked on each tick of a PLC's Timer. If the PLC is connected, it reads data from the PLC for each item in the listView_DB_Locations list view and displays it in the dataGridView_PLCValues data grid view. It also handles starting and stopping the saveDataTimer Timer.

**-button_Disconnect_Selected_PLC_Click(object sender, EventArgs e)**

This event method is invoked when the "Disconnect Selected PLC" button is clicked. It checks whether a PLC is selected in the listView_ConnectedPLCs list view. If so, it stops and disposes of the Timer associated with the selected PLC, closes the PLC connection, removes the PLC from the plcs list, and removes it from the listView_ConnectedPLCs list view. It also removes all rows associated with the PLC from the dataGridView_PLCValues data grid view.

**-UpdateDataGridView(string plcName, string itemName, string db, string startByteAdr, string dataType, string valueRead, string unitOfMeasurement)**

This method updates the dataGridView_PLCValues data grid view with the passed parameters. If a row with the same PLC name and item name is

found, it updates the value read. If no such row is found, it adds a new row with the passed parameters.

**-button_Add_Click(object sender, EventArgs e)**

This event method is invoked when the "Add" button is clicked. It reads the name, DB, location, data type, and unit of measurement from the user interface, then adds a new item to the listView_DB_Locations list view with this information.

**-button_Remove_Click(object sender, EventArgs e)**

This event method is invoked when the "Remove" button is clicked. It checks whether an item is selected in the listView_DB_Locations list view. If so, it removes all rows associated with the selected item from the dataGridView_PLCValues data grid view and removes the selected item from the listView_DB_Locations list view

### 3.1.2 Reports Generation

The Reports Generation component of the PLC Reader Application provides functionality for creating PDF and CSV reports of PLC data over a specified time period.

**-GeneratePdfReport(SysDateTime startTime, SysDateTime endTime)**

This method generates a PDF report of PLC data between the provided startTime and endTime. The report includes a timestamped list of item names and their corresponding values and units of measurement.

This method first reads data from the SQLite database and pivots the data table to create a new table with one row per timestamp and one column per unique item name. The PDF report is then created using the iTextSharp library, with the images and pivoted data table included.

The PDF report is saved to the user's desktop with the filename in the format PLCValues_StartTime_EndTime.pdf.

**-GetImageDataFromResource(string resourceName)**

This method retrieves image data from a specified resource embedded in the assembly. The resourceName parameter should be the name of the image file in the Resources folder.

**-button_generateReport_Click(object sender, EventArgs e)**

This event handler is triggered when the user clicks the "Generate Report" button. It reads the start and end times from the respective text boxes, and then calls the GeneratePdfReport method with these times.

**-button_generateCSV_Click(object sender, EventArgs e)**

This event handler is triggered when the user clicks the "Generate CSV" button. It reads the start and end times from the respective text boxes, and then calls the GenerateCSVReport method with these times.

**-GenerateCSVReport(SysDateTime startTime, SysDateTime endTime)**

This method generates a CSV report of PLC data between the provided startTime and endTime. The report includes a timestamped list of item names and their corresponding values and units of measurement. It first reads data from the SQLite database and pivots the data table to create a new table with one row per timestamp and one column per unique item name. The CSV report is then written to a file using the StreamWriter class. The CSV report is saved to the user's desktop with the filename in the format PLCValues_StartTime_EndTime.csv.

By using these methods, users can easily generate PDF or CSV reports of PLC data for any specified time period, helping improve efficiency and facilitate better decision-making.

SHIPD

kronospan

**2023-04-16 15:10:18 - 2024-01-01 12:00:00**

| Timestamp | motorSpeed | windSpeed | PID | conveyorSpeed |
|---|---|---|---|---|
| | m/s | km/h | % | m/s |
| 2023-04-16 15:10:18 | 539.4011 | -32645 | -16160 | 188.4126 |
| 2023-04-16 15:10:23 | 616.535 | -27778 | -16112 | 216.071 |
| 2023-04-16 15:10:28 | 762.8077 | -22903 | -16320 | 267.3085 |
| 2023-04-16 15:10:33 | 909.2006 | -18032 | -16320 | 318.4935 |
| 2023-04-16 15:10:38 | 925.8499 | -13153 | -16256 | 323.6557 |
| 2023-04-16 15:10:43 | 730.957 | -8279 | -16128 | 255.4152 |
| 2023-04-16 15:10:48 | 536.264 | -3408 | -16112 | 187.3146 |
| 2023-04-16 15:10:53 | 618.9586 | 1462 | 16384 | 216.9403 |
| 2023-04-16 15:10:58 | 765.1412 | 6330 | 16256 | 268.1673 |
| 2023-04-16 15:11:04 | 911.7142 | 11212 | 16656 | 319.5309 |
| 2023-04-16 15:11:09 | 922.3231 | 16093 | 16448 | 322.4213 |
| 2023-04-16 15:11:14 | 727.4302 | 20967 | 16576 | 254.2088 |
| 2023-04-16 15:11:19 | 532.6172 | 25842 | 16256 | 186.0103 |
| 2023-04-16 15:11:24 | 621.8326 | 30716 | 16544 | 217.9357 |
| 2023-04-16 15:11:29 | 768.3456 | -29940 | -16512 | 269.2363 |
| 2023-04-16 15:11:34 | 915.2789 | -25048 | -16160 | 320.6734 |

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Timestamp | motorSpe | windSpee | PID | conveyorSpeed | |
| 2 | | m/s | km/h | % | m/s | |
| 3 | 4/16/2023 15:10 | 539.4011 | -32645 | -16160 | 188.4126 | |
| 4 | 4/16/2023 15:10 | 616.535 | -27778 | -16112 | 216.071 | |
| 5 | 4/16/2023 15:10 | 762.8077 | -22903 | -16320 | 267.3085 | |
| 6 | 4/16/2023 15:10 | 909.2006 | -18032 | -16320 | 318.4935 | |
| 7 | 4/16/2023 15:10 | 925.8499 | -13153 | -16256 | 323.6557 | |
| 8 | 4/16/2023 15:10 | 730.957 | -8279 | -16128 | 255.4152 | |
| 9 | 4/16/2023 15:10 | 536.264 | -3408 | -16112 | 187.3146 | |
| 10 | 4/16/2023 15:10 | 618.9586 | 1462 | 16384 | 216.9403 | |
| 11 | 4/16/2023 15:10 | 765.1412 | 6330 | 16256 | 268.1673 | |
| 12 | 4/16/2023 15:11 | 911.7142 | 11212 | 16656 | 319.5309 | |
| 13 | 4/16/2023 15:11 | 922.3231 | 16093 | 16448 | 322.4213 | |
| 14 | 4/16/2023 15:11 | 727.4302 | 20967 | 16576 | 254.2088 | |
| 15 | 4/16/2023 15:11 | 532.6172 | 25842 | 16256 | 186.0103 | |
| 16 | 4/16/2023 15:11 | 621.8326 | 30716 | 16544 | 217.9357 | |
| 17 | 4/16/2023 15:11 | 768.3456 | -29940 | -16512 | 269.2363 | |
| 18 | 4/16/2023 15:11 | 915.2789 | -25048 | -16160 | 320.6734 | |
| 19 | 4/16/2023 15:11 | 917.5469 | -20164 | -16224 | 320.7357 | |
| 20 | 4/16/2023 15:11 | 722.6541 | -15289 | -16128 | 252.5091 | |
| 21 | 4/16/2023 15:11 | 527.5213 | -10407 | -16160 | 184.2127 | |
| 22 | 4/16/2023 15:11 | 625.6877 | -5530 | -16512 | 219.285 | |
| 23 | 4/16/2023 15:11 | 771.9904 | -658 | -16128 | 270.5225 | |
| 24 | 4/16/2023 15:12 | 918.4433 | 4218 | 16608 | 321.7389 | |
| 25 | 4/16/2023 15:12 | 913.8102 | 9102 | 16256 | 319.2739 | |
| 26 | 4/16/2023 15:12 | 718.4376 | 13976 | 16544 | 251.0614 | |
| 27 | 4/16/2023 15:12 | 523.1849 | 18861 | 0 | 182.7229 | |
| 28 | 4/16/2023 15:12 | 628.6218 | 23724 | 16384 | 220.2909 | |
| 29 | 4/16/2023 15:12 | 774.8644 | 28595 | 16576 | 271.5074 | |
| 30 | 4/16/2023 15:12 | 921.3774 | -32062 | -16384 | 322.7764 | |
| 31 | 4/16/2023 15:12 | 909.8337 | -27189 | -16512 | 318.078 | |
| 32 | 4/16/2023 15:12 | 715.1807 | | | | |
| 33 | 4/16/2023 15:12 | | -22320 | -16512 | 249.9214 | |
| 34 | 4/16/2023 15:12 | 520.4477 | -17448 | -16112 | 181.7649 | |

### 3.1.3 Databases

The Databases region contains methods for interacting with a SQLite database that stores PLC data.

**-CreateDataTable()**

This method establishes a connection to a SQLite database and creates a table named PLCValues, if it does not exist. The table has columns for Id, PLCName, ItemName, DB, StartByteAdr, DataType, ValueRead, UnitOfMeasurement, and Timestamp.

**-listView_DB_Locations_SelectedIndexChanged(object sender, EventArgs e)**

This event method is invoked when the selected item in listView_DB_Locations is changed. It populates form fields with data from the selected item.

**-CreateDatabaseFile()**

This method checks if a SQLite database file with the name PLCValues.db exists. If it doesn't, the method creates the file.

**-SaveDataToDatabase(string plcName, string itemName, string db, string startByteAdr, string dataType, string valueRead, string unitOfMeasurement)**

This method saves a record to the PLCValues table in the SQLite database. It inserts a new row with the provided PLC name, item name, DB, start byte address, data type, value read, unit of measurement, and the current timestamp.

**-SaveAllDataToDatabase(Plc plc, string plcName)**

This method loops over all the items in the listView_DB_Locations, reads the values from the PLC, and saves them to the SQLite database by invoking the SaveDataToDatabase method.

**-FetchTimeAndValueData(string itemName)**

This method fetches timestamped value data for a given item name from the PLCValues table. It returns a list of tuples, where each tuple contains a timestamp and a value.

### 3.1.4 Graphs

The Graphs region contains methods for visualizing the data stored in the SQLite database as a line chart.

**-buttonCreateChart_Click(object sender, EventArgs e)**

This event method is invoked when the "Create Chart" button is clicked. It reads the ItemName from a textBox, connects to the SQLite database, and retrieves all rows with the specified ItemName, ordered by Timestamp. If data is found, a DataTable is loaded with the data and passed to the CreateChart method. If no data is found, a message box is displayed to the user.

**-CreateChart(DataTable dataTable)**

This method creates a line chart in the chart1 Chart control. It first clears any existing series, sets the X-axis label to "Timestamp" and the Y-axis label to "ValueRead", and configures the X-axis to display dates in the "yyyy-MM-dd

HH:mm:ss" format. The X and Y axes are set to auto-scale based on the data points.

A new Series named "ValueRead" is created and added to the chart. The XValueType property of the series is set to DateTime, and the ChartType property is set to Line.

Next, the method loops through the rows of the passed DataTable, parsing the Timestamp and ValueRead values, and adds these as points to the series. Finally, the method forces the chart to recalculate the axis scales.

## 3.2 Database Structure

The PLCValues table in the SQLite database has the following structure:

**Id**: An integer primary key that auto-increments.

**PLCName**: A text field that holds the name of the PLC from which the value was read.

**ItemName**: A text field that holds the name of the item.

**DB**: A text field that stores the DB number.

**StartByteAdr**: A text field that stores the start byte address.

**DataType**: A text field that stores the data type.

**ValueRead**: A text field that stores the value read from the PLC.

**UnitOfMeasurement**: A text field that stores the unit of measurement for the value.

**Timestamp**: A text field that stores the timestamp at which the value was read.

# Chapter 4. Application Testing

## 4.1 Testing Procedure

The developing team chose a test-driven approach, which means that some possible problems were thought of before the implementation of the application, and the intent of avoiding them gave shape to the validation part. The user is in this way warned about the incorrect format of the information provided.

## 4.2 Results

The testing of the application followed the model of different actions that can be performed on a page. The tests display what was done in the application, the expected output for a successful application, and the actual output. Since the approach of the implementation of the application was in order to avoid problems and bugs, the majority of the tests have passed.

# Chapter 5. Conclusion

In conclusion, the PLC Reader Application is a powerful tool for interacting with and extracting data from Programmable Logic Controllers (PLCs) in industrial automation systems. It provides extensive functionalities, including real-time monitoring, historical data storage, data analysis, and reporting. By improving efficiency, reducing costs, and facilitating better decision-making, the application enhances the overall operational effectiveness of industrial processes. The hardware component of the application utilises the Siemens Simatic S7-1500 PLC model, while the software component encompasses PLC communication and display, reports generation, database management, and graphical visualisation.

The application has undergone thorough testing, resulting in a robust and reliable solution. Overall, the PLC Reader Application proves to be a valuable asset for anyone working with PLCs and seeking to optimise their industrial automation systems.